Understanding the popularity of reporters and assignees in the Github

Joicy Xavier, Autran Macedo, Marcelo de A. Maia
Computer Science Department
Federal University of Uberlândia
Uberlândia, Minas Gerais, Brasil
joicyxavier@mestrado.ufu.br, autran@fc.ufu.br, marcmaia@facom.ufu.br

***Abstract*—Github has evolved from traditional version control systems to incorporate the wave of the Web 2.0. Intensive collaboration among developers is one of the main goals of Github beyond traditional version control. Understanding how those developers collaborate is a key issue to enhance the outcomes of individuals and of the ecosystem as a whole, as well. Developers activity during the collaboration may be partially registered in the Github database. The analysis of this database can help to answer important questions about different facets of collaboration. In this work, our interest is to understand which factors can influence developers' popularity and provide insights for individuals to enhance their own popularity. We measure popularity with the number of developer followers. We have analyzed a subset of the Github database in order to explain the high popularity phenomenon. Although, we have found that commit activity is an important factor for high popularity, we also have observed developers with low activity (reports and assigns) but with a high number of followers. We present external factors that can explain this dichotomy and they should be considered as key factors in the ecosystem of open-source development.**

***Keywords*—Github, popularity, profiling**

I. Introduction

The data availability in an unprecedented scale imposes a major challenge on our capacity to extract relevant information from data-intensive repositories. In the software development scenario, the situation is similar, for instance, *Ohloh.net* accounts for near 30 billion stored lines of code in February, 2014. The Web 2.0 phenomenon has influenced how software engineers manage projects. Collaboration has been considered a key success factor for the software lifecycle and service providers are increasingly offering more support for software development collaboration. The service providers for software development sharing has been moving from the classical repository supporting version control to more sophisticated services such as, issues trackers and collaborative code review. They also offer some support to extract open data from the repository. In this scenario, mining software repositories plays a key role to shed light on the intricate relationships that manifest in the different artifacts produced during the development process.

A major service provider for project hosting is Github, which hosts not only source code and respective commits, but also the life cycle of issues and the events raised from the collaboration among their users. Github users may be viewed according to their role during the software life cycle. The user roles which we are interested are based on the activities around the life cycle of an issue. A user is not necessarily a developer that commits to the repository to fix known bugs. Users can also be those that report the issues without necessarily having to fix them, although it is possible that the user can assume both roles. It is important to be aware of the specific role of the user in the software life cycle because different user profiles may induce different collaboration patterns and consequently different popularity profile. It is also important to understand different patterns and profiles in this collaboration process in order to improve our behaviour as a community [5]. Popularity is already an important issue for studies in social networks [2], [4], [9]. However, social networks of developers may have their own specific issues and should be understood on their own.

In this work, we aim at understanding the behaviour of these kind of roles using a partial dataset from selected projects of Github. Our goal was the establishment of users that typically report bugs and of users that typically fix bugs. We assessed variables such as registration time, number of commits, number of participating projects, and number of followers. In order to discover the possible patterns concerning that data, specially which is typical profile of popular users in the Github, we used descriptive statistics and a data mining algorithm [1].

The remaining of the paper is organized as follows: in Section 2, we propose the used methodology. In Section 3, we have shown the results and finally in Section IV, the concluding remarks are presented.

II. Methodology

We used a subset of the dataset provided by the GHTorrent group [6][1]. This dataset contains data from 90 projects, including the top-10 starred software projects for the top programming languages on Github. The dataset accounted for more than 150,000 issues in 13 different programming languages. Our subset considered only issues that were associated to bugs, then the number of issues was drop to 56,959. (From now on, every time we mention issue or dataset, we mean the issue of our subset or our subset, respectively). Both reporters and assignees can be associated to the issues. A reporter is a Github user, which creates an issue (record) relating to a bug information. Whenever a bug is fixed or a bug is allocated to some developer, the respective issue record is related to

[1] http://2014.msrconf.org/challenge.php

TABLE I. RANGES USED IN THE DATA ANALYSIS

Time	Followers	Commits
0-1 years	0-50	0-100
1-2 years	50-100	100-300
2-3 years	100-150	300-500
3-4 years	150-200	500-1000
+4 years	+200	+1000

that user (a.k.a assignee). For each user (reporter or assignee) associated to issues, we computed how long time the user has been registered in the Github (*Time*), the number of followers (*Followers*), the total number of commits in all participating projects (*Commits*), and the number of collaborating projects (*Projects*). These variables were computed in order to understand the user profile concerning the relationship between activity and popularity.

Firstly, this data was analyzed separately for reporters and assignees in order to understand their profiles concerning issues. For this was done a data analysis to identify the numeric ranges of each attribute and were created scales in order to highlight the most significant intervals.The data was classified into different ranges for the variables *Time*, *Followers*, and *Commits*, which are described in Table I. We defined the ranges according to the mean values to identify the skewness in the data more clearly. A more common approach would be defining ranges according quartiles, but in this case the ranges would have the same number of elements, and we were interested in showing which range had possibly lower or higher number of elements.

The data was analyzed using Weka[2] in order to find association rules. We defined the minimum support equals to 0.1 and the mininum confidence greater equals to 0.4. These values were combined to obtain a reasonable number of rules, as we are working with few variables. A larger confidence and support would generate obvious rules, and this would not be interesting for our study. For instance, if we consider support=0.5 we would have only 52 rules, which has little to say about relationships on the followers. Example of rules with support = 0.5 is given below. The rule concerning followers do not provide significant value saying that reporters has 50 to 100 followers with confidence 0.54. Using support = 0.1 and confidence = 0.4, we obtained 326 rules that were more interesting to be analyzed and will be shown in the sequel.

```
1. Commits=0-100,  Projects=0  ==>
    Time=4+ Usertype=reporter        conf:(0.75)
2. Projects=0 ==>
    Time=4+ Commits=0-100            conf:(0.72)
3. Usertype=reporter  ==>
    Time=4+ Commits=0-100 Projects=0 conf:(0.64)
4. Usertype=reporter ==>
    Followers=50-100                 conf:(0.54)
```

With support = 0.5 the confidence values were also larger, however, only a few rules that say something about the popularity are found, so it was decided to consider a support = 0.1.

[2]www.cs.waikato.ac.nz/ml/weka/index.html

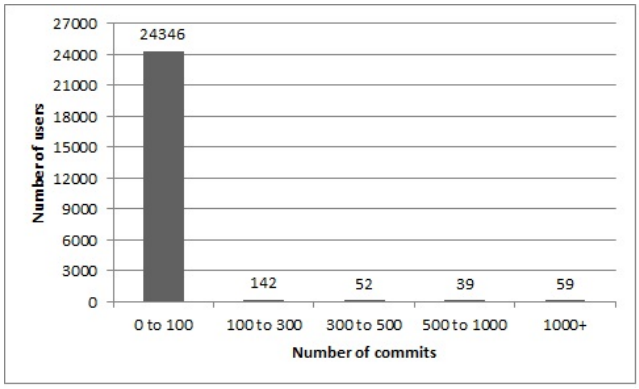

Fig. 1. Number of reporter commits

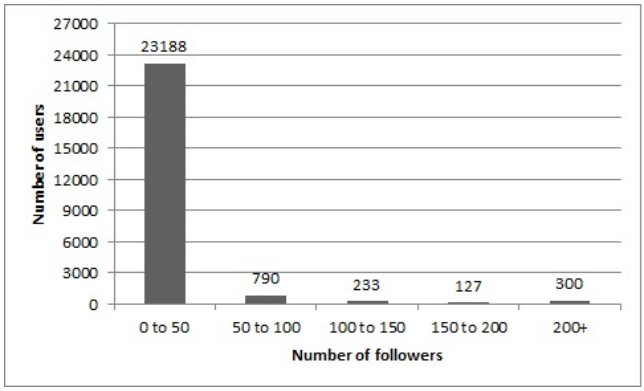

Fig. 2. Number of reporter followers

III. RESULTS

In this section, we present the data analysis for reporters associated with the filtered bugs and for the users that worked on those bugs (assignees).

A. Reporters

Users that create issue records in the Github are called reporters. These users contribute to bug detection and publish them to the community in order to get them solved. The data analysis was driven to establish the typical profile of reporters. We found 24,638 different users that reported the 56,959 bugs, that is, a mean of 2.31 bugs reported per user. For each of these users, we computed the previously described variables *Commits*, *Followers*, *Projects*, and *Time*.

Because reports do not necessarily need to commit, the minimum number of commits was zero and the maximum number was 7,903 commits. As expected, the large majority of the reporters (24,346) have small number of commits (0-100), as shown in Figure 1.

The number of followers for reporters ranged from zero to 12,340. We can observe that most of reporters (94.11%) have small number of followers (0-50). We can observe that there are some reporters that are intensively followed, but only 5.88% have more than 50 followers, as presented in Figure 2.

Github users may collaborate with more than one project. In Figure 3, we have shown the number of projects that each

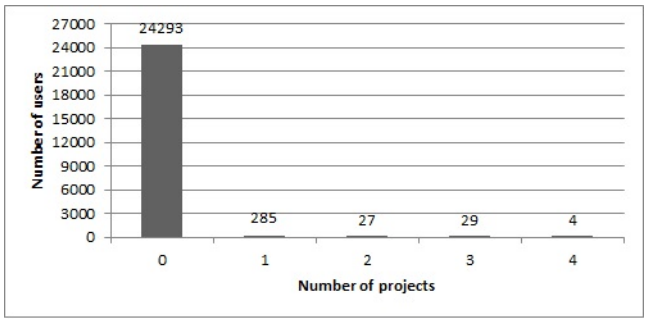

Fig. 3. Number of projects reporters participate

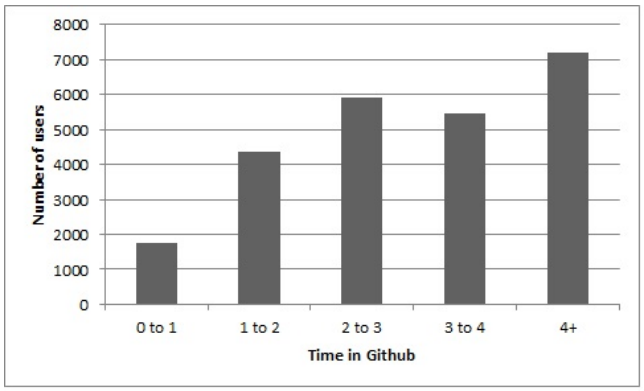

Fig. 4. Registration time for reporters

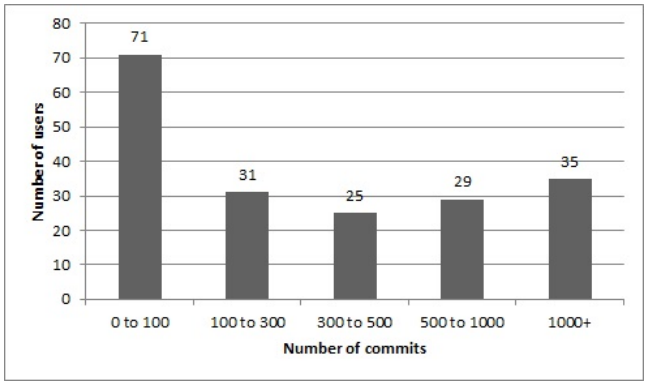

Fig. 5. Number of assignee commits

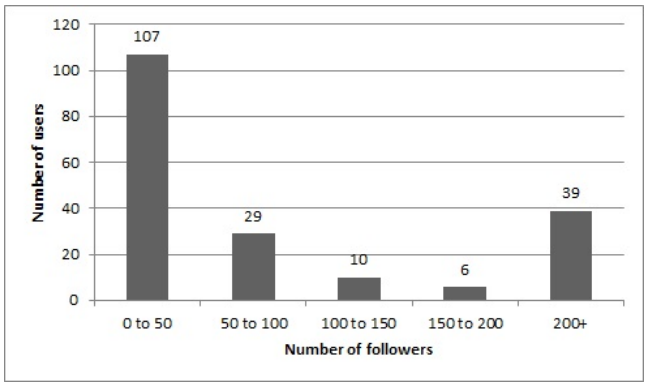

Fig. 6. Number of assignee followers

of the reporters participate. Interestingly, we can observe that 24,293 reporters (98.6%) participate in zero project, once it is possible to report a bug without being attached to the respective project.

The period of time that the reporters are registered in Github varied from 72 days to six years. A significant part of the users (7,181 – ∼30%) have been registered in Github for more than four years. Nonetheless, the other ranges are quite uniformly distributed, as can be observed in Figure 4, except for the range from zero to one year which is the least frequent.

B. Assignees

Assignees are the Github users who fix bugs. Once an assignee fixes a bug the correspondent issue is assigned to him. From the 56,959 bugs, only 4,425 had an associated assignee. For those assigned bugs, 191 different users were involved in their fixing, accounting for a mean of 23 bugs per user. In the same way as we did with reporters, we evaluated the variables *Commits*, *Followers*, *Projects*, and *Time*.

The predominant range for assignee commits was zero to 100. However, the difference was not as high as was for reporters. Moreover, the remaining ranges were quite uniformly distributed, as shown in Figure 5. This was somewhat expected since assignees are responsible for commiting the fixes for bugs. One interesting point is that there is a considerable number of very active assignees (those with more than 1000 commits). This result reinforces the fact that the different assignees' profiles are more uniform among themselves concerning commit activity .

With respect to the number of followers 56% of users (107) have from zero to 50 followers and 20% (39 users) have more than 200 followers, as shown in Figure 6 . The analysis of the number of followers of both reporters and assignees have shown that the 0-50 range is predominant. However, the ranges with higher number of followers have much more higher relative frequency for assignees than for reporters, specially in the range with more than 200 followers. This result shows that although popularity is for very few reporters, this is not exactly the case for assignees. It suggests that assignees tend to be more popular than reporters in Github, which is quite expected because Github is mainly focused on developers.

The group of assignees is distinct from reporters also in the number of projects they participate. While most of reporters do not participate in any project, most of assignees (135 assignes = 70.6%) participate in at least one project, as shown in Figure 7. Only 1% of reporters participates in at least one project.

With respect to the registration time, 51.8% of the assignees are registered in the Github for more than four years and only 2.6% are registered for less than one year, as shown in Figure 8. This could be explained by the sample that considered the top-10 starred projects, which presumably are assumed to have a established team. We can also observe that the larger the registration time, the large is the user contribution in the fixing of bugs.

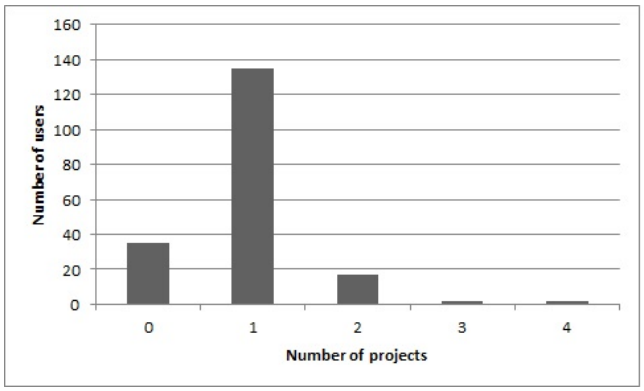

Fig. 7. Number of projects assignees participate

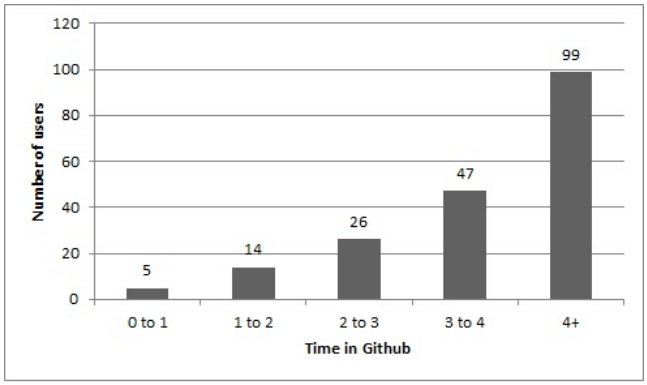

Fig. 8. Registration time for assignees

C. Mining relationships on user variables

We have opted to use association rules to find possible relationships between the activity of users in the repository in order to unveal possibly non-intuitive relationships. The activities can be measured with the attributes time of register, number of commits, number of participating projects, number of followers and that if the user is reporter or assignee of the projects. Thus, we chose the Apriori algorithm because it is a widely used algorithm to find association rules.

We have used the Apriori algorithm to mine association rules using the variables *Commits*, *Followers*, *Projects*, and *Time*.

With respect to reporters, we found rules, shown below, indicating that reporters are typically users with low activity, independently of the registration time.In other words, the productivity of reporters little popular are not likely to improve over time.

```
1. Time=1-2 Followers=0-50 Projects=0  ==>
     Commits=0-100 conf:(1)
2. Time=2-3 Followers=0-50 Projects=0  ==>
     Commits=0-100 conf:(1)
3. Time=3-4 Followers=0-50 Projects=0 ==>
     Commits=0-100 conf:(1)
4. Time=4+ Followers=0-50 Projects=0  ==>
     Commits=0-100  conf:(1)
```

With respect to assignees, the encountered rules have shown that not only assignees tend to have a larger registration

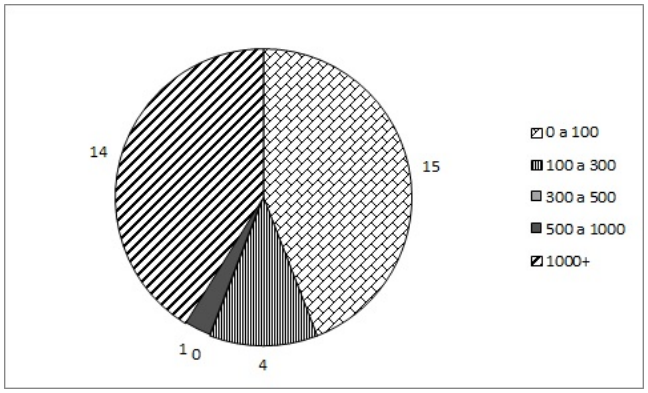

Fig. 9. Number of commits for users with +1000 followers

time, but also they tend to be more active and popular. In this case, the registration time is also related to the number of followers and the number of commits. As we can observe in the following rules, users with registration time greater than four years have higher number of commits and followers, whereas users with registration time less than four years have lower number of followers. We can observe in *rule 6* that users with low number of commits tend to have low number of followers.

```
1. Commits=1000+     ==>
   Time=4+           conf:(0.83)
2. Followers=200+    ==>
   Time=4+           conf:(0.82)
3. Commits=500-1000 ==>
   Time=4+           conf:(0.66)
4. Time=2-3          ==>
    Followers=0-50   conf:(0.81)
5. Time=3-4          ==>
    Followers=0-50   conf:(0.49)
6. Commits=0-a-100   ==>
     Followers=0-50 conf:(0.76)
```

So, we could observe that assignees are generally users with higher registration time, have higher commit activity, and so, have higher visibility among their peers. However, there are still other users (including reporters) that do not have high commit activity and still have very high number of followers. So, we decided to manually inspect those users with the highest number of followers to identify other factors that could promote their popularity. The obtained association rules could not provide that information, so we deepened the analysis with descriptive statistics.

We have filtered in the +200 commits range, those users with more than 1000 commits. We found 34 users, from those none were only assignees. Indeed, 20 were only reporters and 14 were both reporter and assignee. Out of these 34 users, 31 have registration time greater than 4 years and the other 3 users have from 3 to 4 years. Curiously, the number of commits has divided this set of users with more than 1000 followers into two predominant subsets shown in Figure 9. In one subset, we can observe users with more than 1000 commits, and the other subset, we have users with less than 100 commits. In other words, although the commit activity can help to achieve the highest levels of popularity, it is not necessarily the major factor because almost half of the highest popular have low

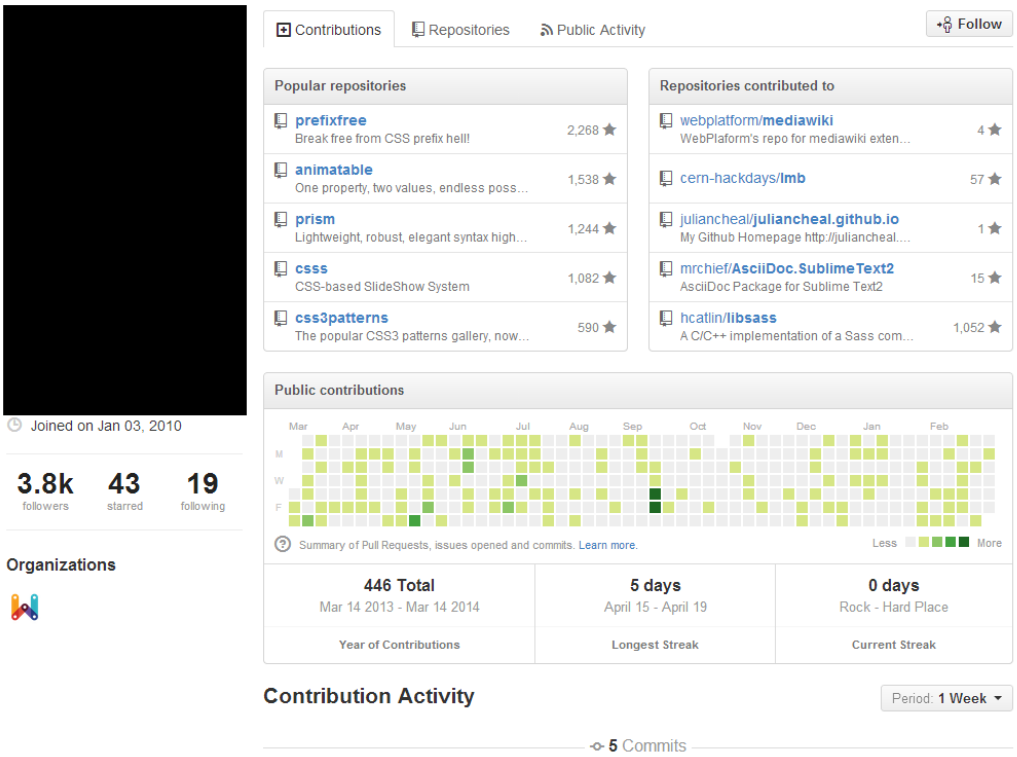

Fig. 10. An example of user with +1000 followers and <100 commits

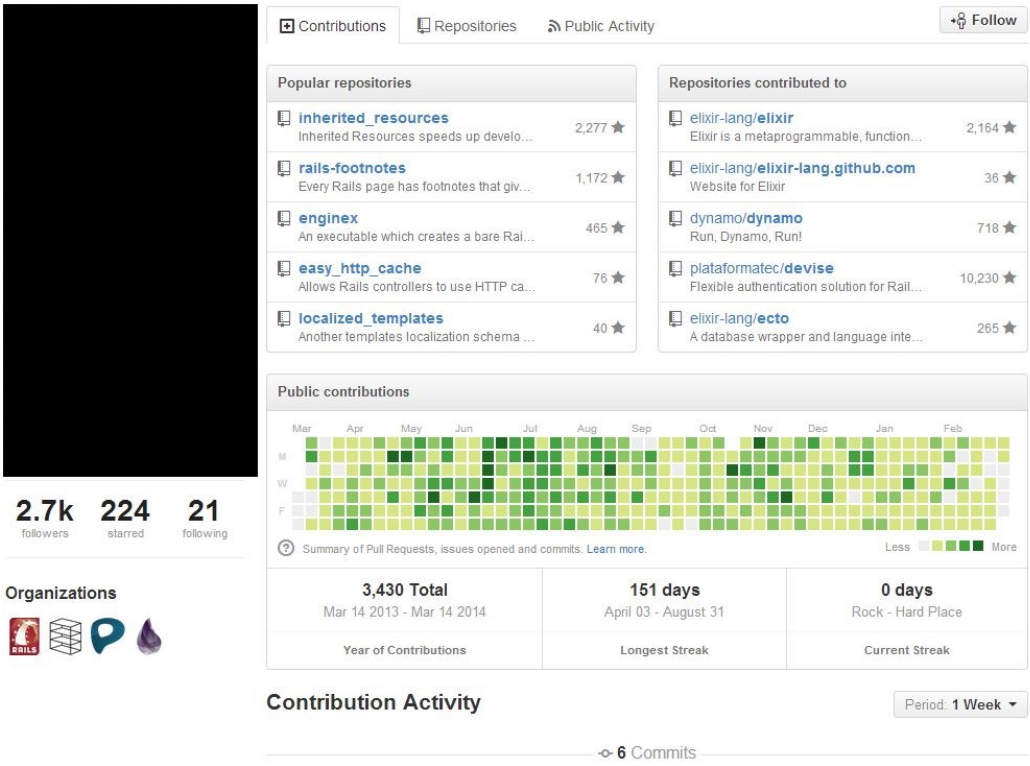

Fig. 11. An example of user with +1000 followers and +1000 commits

commit activity. Indeed, we could find a significant part ($\sim 30\%$) of users with more than 1000 commits that have few followers (0-100).

So, we decided to investigate those top-level popular users that are low profile commiters to understand what could explain their popularity. Out of those 15 users, four are owners of important projects: html5 - boilerplate, jquery, homebrew e rails. So, the ownership of important projects could help to explain high popularity. However, we could not find in the dataset information that could help to explain the high popularity of the other 11 users. So, we decided to manually investigate the activity of those users outside the Github. We have observed that those users have distinguished participation in the web with their blogs, their personal social network, and even with publication of printed books. One of these users did not fix any bug, is not a project owner, follows just a few users, and has more than 2000 followers. However, this user writes blog, acts in several social networks and is a book author. Other user, shown in Figure 10 also has low activity, but has more than 3,000 followers, maybe because of the established blog and high activity in social networks. On the other hand, in Figure 11, we can observe a user with more than 1000 commits and 2,700 followers. In this case, the number of commits represents the importance of the developer in the respective project and consequently, enhance the number of followers.

IV. Related Work

The concern of being popular is somehow related to reputation scores available in other social-techno sites. For instance, in Stackoverflow, users are scored according down-votes and upvotes on their questions and answers. In [3], they conducted an analysis of how users pursue their reputation, and achieved that new contributors who want to earn high reputation scores quickly may have some rules to follow, such as, answering questions related to tags with lower expertise density, answering questions promptly, being the first one to answer a question, being active during off peak hours, and contributing to diverse areas.

The mechanism for achieving reputation can also be viewed as a kind of gamification. In [8] they conduct a literature review to understand how much does gamification works in a general sense. In the case of Stackoverflow, Grant and Betts [7], has shown that badges can be used to influence user behaviour. They have shown an increase in user activity related to a badge immediately before it is awarded when compared to the period after the grant.

Other study has analyzed the impact of the activity in the reputation [11]. Interestingly, they found many users participating in non-core actitvities that had greater than one badge and/or high reputation score. This result is similar to ours in the sense that it unveals that there are other factors that impact reputation, as we have shown.

In [10], the authors have shown that *rockstars*, i.e., highly popular users affect the way and the intensity that followers act in their projects, corroborating with our claim that gaining high popularity is very important to the individuals and their respective projects. Moreover, our results suggest that the activity outside the Github is very important to enhance rockstars popularity and thus enhancing the activity on their projects.

V. Final Remarks

In this work, we have analyzed the profile of bug reporters and assignees in a Github dataset with the top starred projects of Github. We have used descriptive statistics and association rule mining to support our findings.

We have observed that the number and the profile of reporters and assignees are quite different. Because assignees have more specialized activity their are present in lower number, as expect. On the other hand, reporters are users with relative lower registration time because they can more easily contribute in bug detection.

With respect to the popularity, we could observe that the commit activity may influence on the popularity. However, there are other factors outside the Github activity that influence the popularity. We observe that popularity outside the Github, for example, in blogs or social networks, is likely important to improve the popularity inside the Github and should be considered as an important factor to the project success.

VI. Acknowledgments

This work was partially supported by FAPEMIG grant CEXAPQ-2086-11 and CNPQ grant 475519/2012-4.

References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[2] R. Baeza-Yates and D. Saez-Trumper. Online social networks: Beyond popularity. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 489–490, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[3] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft. Building reputation in stackoverflow: An empirical investigation. In *Proc. of the MSR'2013* pages 89–92, Piscataway, NJ, USA, 2013. IEEE Press.

[4] Y. Cha, B. Bi, C.-C. Hsieh, and J. Cho. Incorporating popularity in topic models for social network analysis. In *Proceedings of the 36th International ACM SIGIR Conference*, pages 223–232, New York, NY, USA, 2013. ACM.

[5] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in github: Transparency and collaboration in an open software repository. In *Proc. of CSCW'12*, pages 1277–1286, New York, NY, USA, 2012. ACM.

[6] G. Gousios. The GHTorrent dataset and tool suite. In *Proc. of MSR'13*, pages 233–236, 2013.

[7] S. Grant and B. Betts. Encouraging user behaviour with achievements: an empirical study. In *Proc. of MSR'2013*, pages 65–68. IEEE, 2013.

[8] J. Hamari, J. Koivisto, and H. Sarsa. Does gamification work? – a literature review of empirical studies on gamification. In *Proceedings of the 47th Hawaii International Conference on System Sciences. HICSS*, 2014.

[9] G. Kazai and N. Milic-Frayling. Trust, authority and popularity in social information retrieval. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 1503–1504, New York, NY, USA, 2008. ACM.

[10] M. J. Lee, B. Ferwerda, J. Choi, J. Hahn, J. Y. Moon, and J. Kim. Github developers use rockstars to overcome overflow of news. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 133–138, New York, NY, USA, 2013. ACM.

[11] V. S. Sinha, S. Mani, and M. Gupta. Exploring activeness of users in qa forums. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 77–80. IEEE Press, 2013.