Analysing the 'Biodiversity' of Open Source Ecosystems: The GitHub Case

Nicholas Matragkas, James R. Williams, Dimitris S. Kolovos, and Richard F. Paige
Department of Computer Science, University of York, UK
{nicholas.matragkas;james.r.williams;dimitris.kolovos;richard.paige}@york.ac.uk

ABSTRACT

In nature the diversity of species and genes in ecological communities affects the functioning of these communities. Biologists have found out that more diverse communities appear to be more productive than less diverse communities. Moreover such communities appear to be more stable in the face of perturbations. In this paper, we draw the analogy between ecological communities and Open Source Software (OSS) ecosystems, and we investigate the diversity and structure of OSS communities. To address this question we use the MSR 2014 challenge dataset, which includes data from the top-10 software projects for the top programming languages on GitHub. Our findings show that OSS communities on GitHub consist of 3 types of users (core developers, active users, passive users). Moreover, we show that the percentage of core developers and active users does not change as the project grows and that the majority of members of large projects are passive users.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Measurement

Keywords

Data mining; Data and knowledge visualization

1. INTRODUCTION

Biodiversity is defined as the "totality of genes and species" of an ecosystem [4] and it is considered a primary factor of an ecosystem's stability and productivity. Similar to an ecological ecosystem, research suggests that member diversity of Open Source Project (OSS) communities is an indication of a project's health and longevity. More particularly, [2] performed a diversity-related analysis of the users of 337 open source projects hosted on SourceForge[1] and they found out that diversity in project roles and experience positively affects project success, which is measured by user participation.

Following [1], a community around an OSS product is not a uniform collection of individuals, but it is rather a collection of heterogeneous actors with different goals and motivations. More particularly, an OSS project community has an onion-shaped structure. At the core of this structure are the founders of the project and the project's leaders, who take important decisions. The next layer of the structure consists of the core developers of the project, who have high commit privileges on the source code repository. Surrounding the core developers are the co-developers, who do not have commit rights to the source code repository. These developers usually submit patches, which are reviewed by the core developers and then they are applied to the code base. The next layer of the onion-shaped structure comprises the active users of the project. Active users report new bugs, write documentation and answer questions of user newsgroups. Finally, at the rim of the structure are the passive users of the OSS product. It is the existence of all these roles and their interactions that leads to the success of a project.

In this paper, we investigate whether the GitHub[2] OSS community demonstrates high diversity of user roles and whether there is an emergent structure as a result of the various user activities. More particularly our research questions are the following:

- **RQ1**: Do GitHub communities exhibit diversity and structure? If so, how does this structure look like?
- **RQ2**: Is the structure of GitHub communities similar to the structure of other open source software communities hosted in other forges?
- **RQ3**: Does the size of a community affect its structure?

2. METHODOLOGY

To address the above research questions, we used the GHTorrent MongoDB dataset provided by the MSR challenge [3]. The dataset contains data related to 90 popular GitHub projects and their forks. These projects (totalling 108,710)

MSR'14, May 31 – June 1, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2863-0/14/05...$15.00
http://dx.doi.org/10.1145/2597073.2597119

[1] `http://sourceforge.net`
[2] `https://github.com/`

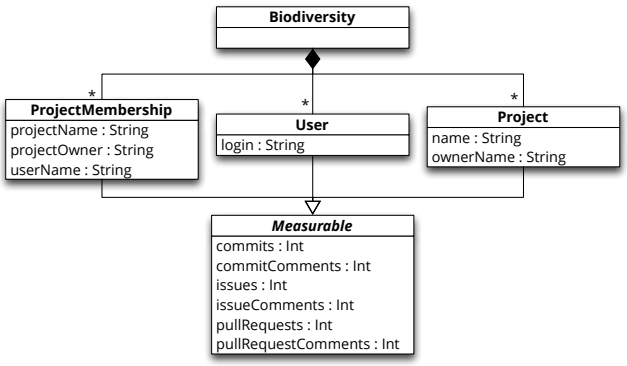

Figure 1: The data model of the transformed data.

are stored in a MongoDB collection. Each commit, commit comment, issue, issue comment, pull request, and pull request comment related to these projects are grouped into MongoDB collections of the same kind. Details of nearly 500,000 users associated with these projects are also stored in their own collection. The dataset also includes data related to project members and user followers.

In order to analyse user behaviour so as to determine each user's role in a project, we need to collate all information related to that user. Due to the fact that this information is contained in separate MongoDB collections, querying can be challenging. To address this we have implemented a preprocessing script to convert the data into a state better suited to the kind of analysis we wish to perform. We now present the data model of the transformed data, briefly overview the transformation, and describe our analysis methods.

2.1 Transforming the Data for Analysis

To transform the data from the disjointed form into an integrated arrangement that captures the activity of users, we need to read and summarise every activity-related entry in the GHTorrent dataset. Rather than interacting directly with the database at a low level of abstraction (using one of MongoDB's many drivers), we utilise *Pongo*[3], a Java-based object-document mapper for MongoDB. From the textual definition of a data model, Pongo generates strongly-type Java classes, allowing users to work with the database at a more convenient level of abstraction. Figure 1 shows the data model, represented as a class diagram, that we have defined using Pongo to collate the data needed for our analysis. We define three *measurable* types: users, projects, and the contribution a single user makes on a single project. Each of these stores counts related to user activity. The `User` class stores the total activity a user has over every project they participate in. The `Project` class stores the total activity of all users on that project. Finally, the `ProjectMembership` class stores an individual's contribution to a single project. Each of the measurable types has its own MongoDB collection for storage, and they are contained in a single MongoDB database (represented by the `Biodiversity` class in figure 1). From this data model definition, Pongo generates the equivalent Java code. An example of using the generated code to interact with the database is shown in listing 1.

[3] `https://code.google.com/p/pongo`

The transformation process involves iterating through a number of the GHTorrent dataset collections and creating or updating entries in the biodiversity database. Initially, we create a `User` for every entry in the GHTorrent 'users' collection, and a `Project` for every entry in the 'repos' collection. We then iterate through the 'commits', 'commit_comments', 'issues', 'issue_comments', 'pull_requests', and 'pull_request_comments' collections updating the counts for associated `User`s and `Project`s whilst simultaneously creating or updating `ProjectMembership` entries.

Due to the fact that we only need to iterate through every collection in the GHTorrent dataset once, we use the MongoDB Java driver directly for reading the dataset, but use the Pongo-generated classes for inserting data into the analysis database. The code used for this transformation can be found at: `https://github.com/ossmeter/msr14-challenge`. As our focus is on user activity, the other parts of the GHTorrent dataset are ignored.

```
1  // Connect to database
2  Biodiversity bio = new Biodiversity();
3
4  // Create a new user and add to database
5  User user = new User();
6  user.setLogin("octocat");
7  bio.getUsers().add(user);
8  bio.sync();
9
10 // Query the database
11 user = bio.getUsers().findOne(
12           User.LOGIN.eq("octocat"));
```

Listing 1: An example using the Pongo-generated code to create a new user and then query the database.

2.2 Analysing Biodiversity of Projects

Although the GHTorrent dataset has now been summarised for biodiversity analysis, it still contains some entries that we need to exclude. After the dataset is cleaned, we perform *cluster analysis* on the data to try to discover natural clusters of user behaviour.

2.2.1 Cleaning the Data

As mentioned, the GHTorrent dataset consisted of 90 popular projects and their forks. There are a total of 108,620 forks, many of which have no activity – i.e. all of the count fields in the database are zero. As such, the vast majority of the 496,519 users also exhibited zero activity. It appears that a common trend on GitHub is to fork a project but make no changes. To inhibit these users from distorting our analysis, we exclude them from the dataset. As our goal is to answer research questions related to the structure of communities on GitHub, we also exclude projects that consist of less than five members (as is also done in [5]).

Table 1 shows a summary of the number of projects, users, and project memberships in the cleaned dataset in comparison to the original. The dramatic decrease in the number of projects is largely due to the removal of many of the inactive forks. Table 1 also shows an insight into the size of the communities around the projects in the GHTorrent dataset, with 21 projects having over 1,000 members.

Table 1: A summary of the dataset. #U is the number of users of a project.

Dataset	Projects	Users	Project Memberships
GHTorrent	108,710	496,519	NA
Biodiversity	108,710	496,519	189,225
Cleaned	20,510	54,659	97,894
Cleaned (#U≥5)	161	40,013	73,270
Cleaned (#U≥10)	99	39,806	72,890
Cleaned (#U≥100)	72	39,341	71,712
Cleaned (#U≥1000)	21	23,774	50,087

2.2.2 Clustering the Data

In order to determine whether there is structure in GitHub communities, and if so, how this structure looks, we need to classify users into roles based on their behaviour. To do this we analyse the project memberships collection to look for commonalities. The project membership collection is the most suitable as it breaks down user behaviour on a per-project basis. A single user may be a member of multiple projects, but may play different roles in each. This means that we cannot analyse the users collection, as this is a cumulative total of each users' behaviour.

Absolute counts of user activity (i.e. number of commits on a project, etc.) is misleading in the context of communities: it provides an unfair comparison between users. For example, a user who has committed 10 times on a project with only 15 commits should be seen as having contributed more than a different user who has committed 10 times on a project with 15,000 commits. As such, during the cleaning of the biodiversity dataset, we convert the absolute counts into proportional values with respect to each project's overall activity. Our first user above will therefore have contributed 66% of the project commits, whereas the second user will only have contributed 0.0007% of their project's commits.

To group similar users together, we apply *cluster analysis* to the dataset. Cluster analysis [6] splits data into groups that are similar in some sense. It is commonly used to define taxonomies in large datasets. In particular, we apply the *k-means* clustering algorithm to the proportionally-logged project membership dataset. K-means divides a dataset into a user-defined (k) number of clusters. In brief, k-means initially selects k random data points (non-random initial selection techniques exist, see [6]) called *centroids* and forms clusters of data points around them based on their Euclidean distance to the rest of the data points. Points are assigned into a cluster with the centroid nearest to them. The centroids are then recalculated by computing the centre of each cluster. This process of assigning clusters and recomputing centroids repeats until the algorithm converges on a fixed set of centroids.

3. RESULTS

In this section we report the results of our analysis aimed at answering the three research questions formulated in Section 1.

To find a satisfactory clustering result, a number of iterations were needed, where we executed the algorithm for different numbers of clusters (k). The validity of the clustering result was then assessed through visualisation. The resulting number of clusters was three.

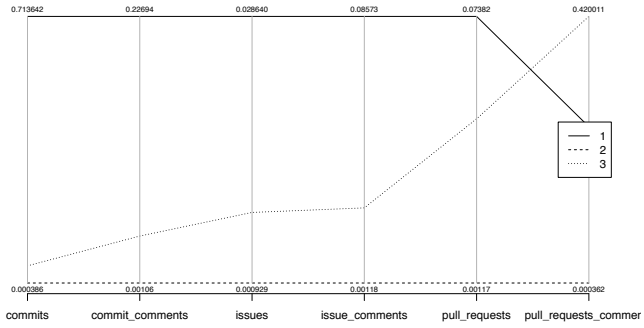

Figure 2: Visualisation of the clusters found by k-means for users working on projects with more than 10 members.

The centroid plot illustrated in Figure 2 shows the exemplar profile for each of the 3 clusters resulting from the k-means clustering applied to the dataset of projects consisting of more than 10 members. The average member in cluster 1 is quite active in commiting but not in the rest of the categories. She is responsible for 71% of the commits, 23% of the commit comments, 2% of the issue reports, 8% of the issue comments, and 7% of the pull requests. Such behavior can be demonstrated by core developers, who focus mainly on development activities and do not communicate often with the rest of the community. On the other hand, the average member of cluster 3 is active in issue reporting and discussing, as well as in commenting on pull requests, but she is less active on development related activities. Such behavior can be demonstrated by active users. Finally, the average member of cluster 2 is not active in any of the categories. Such behavior can be demonstrated by passive users who just follow the project but are not participating. Table 2 lists the size of the three aforementioned clusters. The biggest proportion corresponds to cluster 2 which is the cluster of the passive users (99.8%). The other two clusters, namely cluster 1 and 3, correspond to 0.1% each.

To investigate how project size affects the results, we applied clustering to larger groups as well. Figure 3 illustrates the centroid plot resulting from the k-means analysis of projects with more than 1000 members. The member behaviours derived for large projects are similar to the member behaviours derived from the analysis of projects with more than 10 members but less than 1000. Namely, the average member belonging to cluster 2 is very active in every aspect of the project. Her source code commits correspond to 77% of all the commits, her issue reports correspond to 14% of

Table 2: The sizes of each of the clusters for users working on projects with more than 10 team members.

Group ID	Cluster Size	Proportion
1	65/72806	0.001
2	72668/72806	0.998
3	73/72806	0.001

Table 3: The sizes of each of the clusters for users working on projects with more than one thousand team members.

Group ID	Cluster Size	Proportion
1	40/50067	0.0008
2	11/50067	0.0002
3	50016/50067	0.999

the total issue reports and her pull requests correspond to 20% of the total pull requests. On the other hand, the average user of cluster 1 does not commit source code a lot but she is very active in reporting issues, in discussing about this issues and in creating pull requests. Finally, the average user of the last cluster, namely cluster 3, is very inactive in every activity and therefore can be considered as an observer or passive user of the OSS project.

In addition to the clusters derived for larger projects being similar to the ones derived for small ones, their size as a percentage of the entire populations is quite similar. This size is shown in Table 3. For popular projects only 0.02% of the entire population are core developers, while 0.08% are active users. Finally the vast majority of project members (99%) are just passive users or observers. Based on this observation we can say that the percentage of core developers and active users does not change as the project grows.

The analysis of the other dataset partitions listed in Table 1 gives similar results. The centroid plots for this analysis can be found on GitHub[4].

4. DISCUSSION AND CONCLUSION

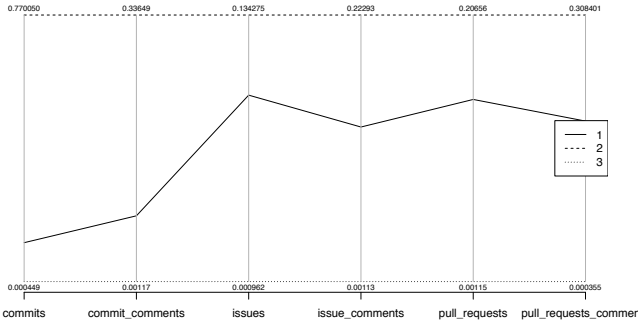

Figure 3: Visualisation of the clusters found by k-means for users working on projects with more than one thousand members.

[4]https://github.com/ossmeter/msr14-challenge

We have performed a cluster analysis on the GHTorrent dataset in order to understand how GitHub communities are structured. We address the research questions set out in section 1:

RQ1: According to our clustering analysis we found that GitHub communities have a structure that consists of three main types of users, namely core developers, active users, and passive users. In these three categories we can add a fourth one, which consists of all the users who fork a project on GitHub and then have no activity on the fork. This activity was excluded from the cluster analysis, in order to avoid distorting the results.

RQ2 The three-layer structure revealed by our cluster analysis is in accordance with the onion-shaped structure proposed by [1]. Moreover, our results are consistent with the results of [2], who find that core developers correspond to a small percentage of all the members of SourceForge OSS projects.

RQ3 The structure of OSS communities is not affected by project size. Moreover, the consistency of such communities does not change the larger a project gets. More particularly, as projects get larger the percentage of core developers working on the project remains stable. This could potentially have implications on the sustainability and longevity of the project. However, the large passive user base may allow for new core members to arise should existing core members leave. Further work is required to determine whether this occurs in practice.

5. ACKNOWLEDGEMENTS

This research was supported by the EU through the Automated Measurement and Analysis of Open Source Software (OSSMETER) FP7 STREP project (318736).

6. REFERENCES

[1] K. Crowston and J. Howison. Assessing the health of open source communities. *Computer*, 39(5):89–91, 2006.

[2] S. Daniel, R. Agarwal, and K. J. Stewart. The effects of diversity in global, distributed collectives: A study of open source project success. *Information Systems Research*, 24(2):312–333, 2013.

[3] G. Gousios. The ghtorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR'13, pages 233–236, 2013.

[4] T. Larsson. Biodiversity evaluation tools for european forests. *Criteria and Indicators for Sustainable Forest Management at the Forest Management Unit Level*, page 75, 2001.

[5] M. Loreau, S. Naeem, P. Inchausti, J. Bengtsson, J. Grime, A. Hector, D. Hooper, M. Huston, D. Raffaelli, B. Schmid, et al. Biodiversity and ecosystem functioning: current knowledge and future challenges. *science*, 294(5543):804–808, 2001.

[6] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.