# On Clusters in Open Source Ecosystems

Shaheen Syed and Slinger Jansen

Department of Information and Computer Sciences, Utrecht University
Princetonplein 5, 3508 TB Utrecht, the Netherlands
{s.a.s.syed, slingerjansen}@uu.nl

**Abstract.** This paper seeks to find characteristics of relationships between developers within various clusters of FLOSS ecosystems. We have mined the repository of the open source programming language Ruby, and linked developers and projects on the basis of collaboration. We used Social Network Analysis, and more specifically, the concept of modularity to expose underlying clusters or sub-communities. A survey was constructed to aid in the qualitative part of this research. The data shows that Ruby's ecosystem consist mostly of single developers who work independently. Developers within clusters of a few developers often have personal relationships formed through friendship, work and the open source community. Personal relationships formed through the open source community grow as clusters consist of more developers. Developers in clusters with large number of developers are often unaware of friend-of-friend relationships. Project administrators, however, fail to list developers that contribute through pull/request issues as authors, making data on Ruby's repository incomplete.

**Keywords:** Software Ecosystems, Clusters, Ruby, FLOSS

## 1 Introduction

Free/Libre Open Source Software (FLOSS) developers have to work collaboratively in order to maintain, improve or extend their software. Especially when their collaboration is underpinned by a common technology or market, and operate through the exchange of information, resources and artifacts, Software Ecosystems (SECOs) start to emerge [1]. Jansen et al. state that getting insights into a SECO requires the quantification and measurement of specific SECO characteristics. Possible characteristics are the number of sub-communities, the reciprocity of the ecosystem or the out degree of keystone actors. Moreover, understanding SECO characteristics can aid in maximizing its profitability, and more concretely, SECO orchestrators can develop strategies to keep a SECO vibrant and profitable for other organizations in the SECO.

This paper addresses the characteristics of sub-communities or clusters within the Ruby SECO. Clusters are defined as sets of nodes which are connected together by some path, that is, there exists a sequence of links that can be traversed to go from one node to any other node. There exist, however, no paths between two nodes in different clusters [3]. As SECOs can be seen as large

social networks, where actors (developers) are connected through relationships (collaborations) that hold them together [4], numerous clusters are present.

The study of SECOs, or social networks, can be conducted by applying Social Network Analysis (SNA). Several studies have applied SNA in the context of open source. For example, Gao, Freeh and Madey used SNA to study core developer networks on SourceForge.net hosted projects [5]. Lopez-Fernandez et al. used SNA to analyze source code repositories of open source projects [6]. Madey et al. applied SNA to model the open source network as collaborative networks and studied the growth of these networks [7]. Previous studies are, however, focused on the developer or project level, and not specifically on the cluster or sub-community level. Concretely put, this paper uses SNA to reveal underlying clusters within the Ruby SECO and strives to define the characteristics of relationships between developers. The ability to find and analyze such clusters can provide invaluable help in understanding and visualizing the structure of networks and thus understanding the SECO. Several algorithms exist to find sub-communities or clusters within networks [8–10]. Popular algorithms for large networks are based on the concept of modularity, which take the structure of a network and decompose it into modular communities, i.e. sub-communities or clusters. It looks for dense connections (relations) within groups, and sparse connections between them. Furthermore, it requires less computational complexity in contrast to other methods, e.g. graph partitioning. Our study uses the Louvain method [11], a modularity algorithm that has successfully been applied to find sub-communities in large networks, such as Twitter, LinkedIn, and Flickr. To extend our analysis, a survey is constructed to aid in the qualitative part of relationships between developers. Hence, Ruby developers were asked numerous questions related to other developers that were clustered together. Doing so enabled us to classify qualitative data by cluster sizes.

The remainder of the paper is structured as follows: Section 2 elaborates on the research method and provides insights into the various steps undertaken from data extraction to cluster identification. Furthermore, it elaborates on the survey and how it was constructed. Section 3 discusses the first findings of the data that were extracted from the Ruby repository and provides an overview of distributions that define the Ruby ecosystem. Section 4 makes a first classification of survey results based on various cluster sizes. Finally, Section 5 concludes our classification of results and provides improvements for future work.

## 2 Research Design

Our study has taken as object of study the Ruby open source SECO. Ruby is an object oriented programming language designed and developed in 1995 by Yukihiro Matsumoto in Japan. The syntax is influenced by Perl and Smalltalk features and is similar in many respects. It was created to balance functional programming with imperative programming. Ruby's most popular framework, named Ruby on Rails, is an open source web framework that enables developers to create web applications more easily. These applications, best described

as utilities or libraries, are called *gems* and are hosted on Ruby's repository Rubygems.org. Several gems can be combined to create specific features or extend existing work.

### 2.1 Research Question

This paper is an extension of our previous explorative study, in which we presented elements, characteristics, descriptives, roles, cliques and relationships of the Ruby ecosystem [12]. Our previous work was quantitative in nature and lacked essential qualitative information. We replicated the study and focused on relationships between developers within clusters. Thus, the main research question answered in this paper is: *What are the defining characteristics of relationships between developers within clusters of FLOSS ecosystems?* To answer this question, we looked at relationships between developers within the Ruby SECO of various cluster sizes with respect to the number of developers residing within them.

Developers are connected by strong or weak ties. For example, developers have strong ties if they are friends, classmates or working at the same company (i.e. they know each other personally to some extent), and have coded together to produce a working gem. Weak ties, by contrast, involve infrequent and limited interaction. For example, developers are connected together through a friend-of-friend relationship, that is, developer $a$ has worked with developer $b$, who in turn has worked with developer $c$, developer $a$ and $c$ are now connected through developer $b$. Our study further looks at these friend-of-friend relationships and to what extent they constitute strong or weak ties.

Strong ties are important because it improves the exchange of information with other strong ties and reinforce companionship and support. The study of strong ties can help us understand the structure and local information of a social group [3]. Weak ties bring us more new opportunities and resources which cannot be obtained from close relations [13]. The study of weak ties provides a global view of the community as a whole.

### 2.2 Data Gathering

We collected our data from Rubygems.org as it provided us with an API to mine the data. Because integral collection was not possible, we developed several scripts to extract the data one by one from the API's XML output. The data was then stored in a relational database. The API of Rubygems.org provided us with the following data on each gem:

1. Gem name
2. Total downloads
3. Current version
4. Downloads current version
5. Authors
6. Info
7. Project uri
8. Gem uri
9. Homepage uri
10. Wiki uri
11. Documentation uri
12. Mailing list uri

13. Source code uri
14. Bug tracker uri
15. Dependencies

As our research was targeted at developers (authors), we distilled this information from the complete dataset. Unfortunately, Rubygems.org stored author information as a string, causing problems when multiple authors have collaborated on a gem. Several formats for entering multiple authors were used by project administrators, such as "Peter and John", "Peter & John", "Peter, John". This information was decoupled using several SQL statements which ultimately resulted in database records with single entities. Each gem was given a unique id and was linked to one or multiple authors. Data extraction was performed on May 7th 2012, providing us with a snapshot of the Ruby ecosystem as it was then.

### 2.3 Cluster Identification

After data collection we visualized the data using *Gephi* [14], which is a free to use application for social network analysis. Furthermore, Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs.

A common property within networks is *community structure*. It is the division of network nodes into groups within which the connections are dense, but between which they are sparse [9]. Newman and Girvan defined the measure of *modularity* to quantify the strengths of communities by using the denseness and sparsity of the group's inner and outer connections. Their modularity measure is a scalar value between -1 and 1, where 1 indicates networks with clusters that are disconnected from each other, i.e. no nodes overlap two clusters. Their algorithm is especially useful for undirected and unweighted graphs to reveal underlying clusters.

To identify clusters within the Ruby ecosystem we used the algorithm proposed by Blondel, Guillaume, Lambiotte and Lefebvre [11], known as the Louvain algorithm. Their algorithm is based on the work of Newman and Girvan but improved in terms of computer time and modularity. We used the complete dataset to create the graph where each node consists of a developer or gem. Developers were connected to other developers if they both contributed to the same gem. The above mentioned algorithm was then performed on the graph.

### 2.4 Survey

To get insights in the qualitative aspects of clusters, we constructed an online survey that was personalized for Ruby developers registered on Rubygems.org. The online survey[1] was able to retrieve information on the respondent's cluster from the database that contained information on developers and gems. For example, if developer $a$ was clustered in cluster $k$, all other gems and developers in cluster $k$ were shown to developer $a$ when answering the survey.

[1] http://www.ruby-research.com

Data was collected between November 23rd 2012 and February 16th 2013. Invitations to the survey were posted on forums, message boards, and developers were invited to participate by contacting them via email. Email addresses were obtained from public profiles on Rubygems.org.

The variables measured in the survey fall into two categories. The first category measured relationships (ties) towards other developers and other gems residing in the respondent's cluster, and if they had collaborated in producing gems. Furthermore, it measured to what extent they had personal relationships to other developers and, if any, how they were established. The second category measured motivations for creating new gems and the various contributions/roles developers have when creating gems with other developers.

To measure characteristics of relationships (e.g. tie strength), answers could be given on an ordinal scale (e.g. I know no - few - half - almost all - all developers in my cluster). Because our research was explorative of nature, we were not concerned about interval or ratio data to perform statistical analysis. The ordinal scale was sufficient in the sense that it enabled us to make some sort of classification based on the *a priori* grouping of cluster sizes based on the number of developers. Other questions were measured on a nominal scale, such as the various types of relationships or motivations. Moreover, several checkboxes could be selected that were processed with multiple answer analysis.

## 3 Data

At the time of analysis, the Ruby ecosystem consists of 37,551 gems and 15,679 developers. On average, each gem is created by 2.39 developers. The gems vary in their total downloads from just 48 downloads ("Whitelabel" by Peter Schröder) to a number as high as 12,623,891 downloads ("Rack" by Christian Neukirchen), with a mean value of 16,477.58 and a standard deviation of 238,668.06. Only 113 gems have total downloads exceeding one million, which constitute for 0.3% of the total ecosystem. In contrast, 18,783 gems have equal or less than 1,000 total downloads, representing 50% of the total ecosystem. Out of all the gems, 3.8% had an additional uri to a wiki of their gem, 7.9% provided an uri to additional documentation and only 1.5% provided the option to subscribe to their mailing list.

### 3.1 Clusters

We have identified 10,586 valid clusters by performing the modularity algorithm proposed by Blondel et al. [11]. The modularity achieved by performing modularity optimization (optimizing the modularity to find distinct clusters) was 0.985, an indication that nearly all clusters are unique, i.e. no developers exist in more than one cluster. These clusters are derived from all 37,551 gems and 15,679 developers. During data analysis, a total of 328 clusters contained a single node, which constituted a gem. These clusters were excluded from our study as, apparently, project administrators failed to register its corresponding

developer(s). A manual search showed that the majority of these clusters are registered as version 0.1, had just a few downloads and provided no additional information. This can be caused by the free nature of its repository, where registering a gem requires no strict rules or formats. An illustration of a random cluster with 7 developers and 13 gems is shown in Fig. 1

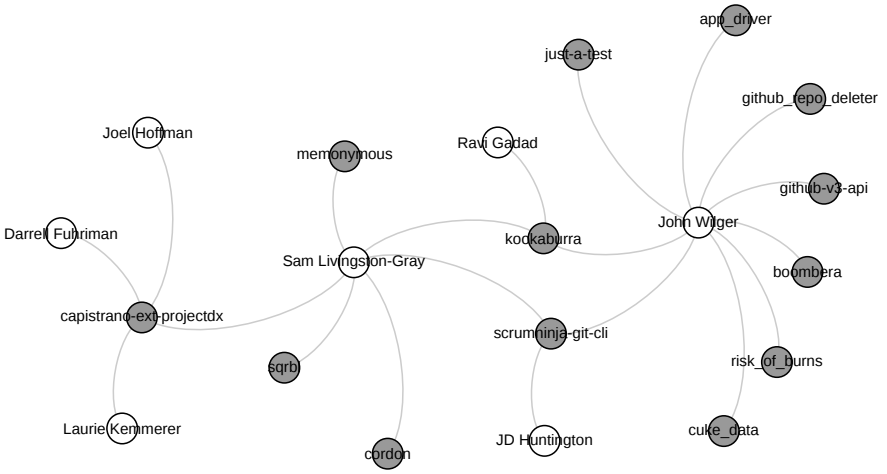


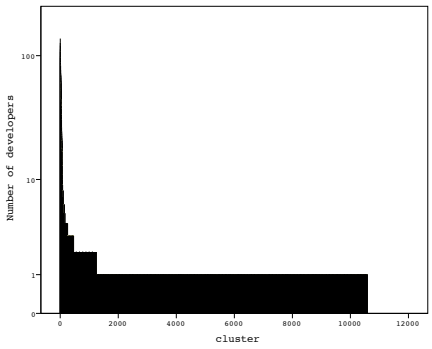

**Fig. 1.** Cluster with 7 developers (white nodes) and 13 gems (grey nodes) and the relations between them.

**Number of Nodes** Most clusters encompass just two nodes (5,819 clusters that constitute 55% of the total ecosystem), that is, one developer that is linked to its own gem. A smaller, but still relatively large part, consists of three nodes (2,035 clusters, 19.2%), being a single developer who created two gems, or two developers that have worked on the same gem. The distribution of nodes across clusters shows that a small portion of just 0.5% are clusters with over 100 nodes. In addition, the largest cluster consist of 600 nodes: 136 developers who created 464 gems, which account for less than 0.01% of the total ecosystem. Interestingly, none of Ruby's top 10 most downloaded gems were part of the top 6 largest clusters. "Activesupport", "Activerecord", "Actionpack", "Actionmailer", "Activeresource" and "Rails", created by David Heinemeier Hansson, were part of the 7th largest cluster (107 developers and 370 gems).

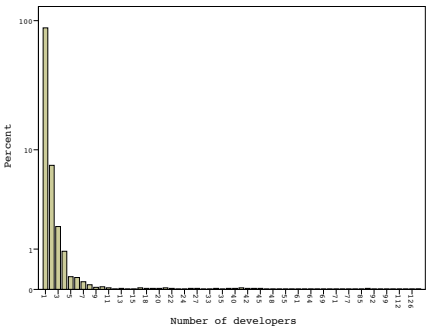
**Number of Developers/Gems** Figures 2(a), 2(b), 2(c), 2(d) show the highly skewed developer and gem distribution across clusters. The skewed distributions further show that the vast majority of clusters consist of just one or two developers. We identified 9,341 clusters with 1 developer (88.2%), 786 clusters with 2

developers (7.4%), and 206 clusters with 3 developers (1.9%). This shows that Ruby's ecosystem, when looking at the number of developers within a cluster, consists of 97.6% of clusters that are not larger than 3 developers. Moreover, similar numbers are found when focusing on the number of gems. Approximately 90% of all clusters have not produced more than 4 gems.

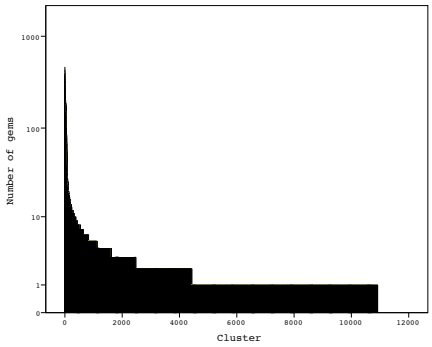
The ample part of the Ruby ecosystem consists of single developers working on one or multiple gems. A smaller, but still relatively large part, consists of two or three developers working together to produce gems. There are few clusters that encompass tens or even +100 developers that are connected through collaboration.



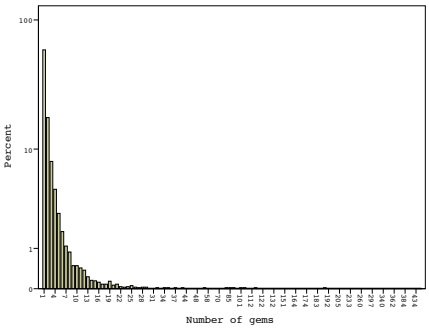

(a) Developer distribution




(b) Number of developers as percentage of the complete ecosystem




(c) Gem distribution




(d) Number of gems as percentage of the complete ecosystem

**Fig. 2.** Distribution of developers and gems across clusters: y-axis logarithmically transformed

## 4 Survey

A total of 782 respondents filled in the survey, being approximately 5% of all registered Ruby developers on Rubygems.org. To classify the responses, we grouped clusters by the number of developers within them. As the distribution of developers within a cluster is highly skewed, we created groups with minor increments up to 10 developers. The remainder of clusters, being that with 10+ developers are grouped together. This enabled us to classify survey data on cluster sizes with respect to the number of developers. We excluded clusters of a single developer for questions related to ties, as none were present. The classification of responses are as follows: Clusters with one developer 403 responses, clusters with 2-3 developers 95 responses, clusters with 4-5 developers 35 responses, clusters with 6-7 developers 13 responses, clusters with 8-9 developers 4 responses, and finally, all clusters that have 10 or more developers 232 responses.

### 4.1 Ties through Collaboration

Tables 1 and 2 show survey data with respect to relationships with other developers that are clustered together. The data of Table 1 provides an overview of ties to other developers. These ties are a mixture of strong and weak ties, for example, developers who have collaborated, have emailed, used each other's gems etc. In addition, Table 2 shows the presence of strong ties, i.e. personal relationships, classified into various cluster sizes.

Two-third of the respondents in clusters with 2-3 developers know all other developers with more than 50% of them having a personal relationship. When looking at large clusters with over 10 developers, the vast majority (84.5%) of developers know just a few other developers, 61.6% of these are personal of nature. Comparing both tables reveals that the relations of strong ties in contrast to ties in general do not drop considerable.

Additionally, we asked, if personal relationships were present, how they were established. The results are shown in Table 3. The large part of these personal relationships are formed through friendships, work mates and through the open source community. In all cluster groupings, they constitute the majority of the responses. Personal relationships that are formed through the open source community seem to grow from 16.5% in clusters with 2-3 developers, to 33.7% in clusters with over 10 developers. In contrast, the percentage of personal relationships that are formed through work associates (work at the same company) seems to diminish from 41% in clusters with 2-3 developers to 25.4% in clusters with 10+ developers. Furthermore, the Ruby ecosystem consists of very few personal relationships that are bounded by family or schoolmates. The respondents that answered "other" were given the option to write out an alternative. More than half of them answered "conference" as a mean to establish personal relationships.

The relationships between developers and gems are based on the author field of the Rubygems.org repository. The algorithm, therefore, seeks for connections

based solely on this information. We asked developers if they would have expected other developers or gems to be clustered within their cluster. From all responses, 36.8% had expected other gems, and 30.8% had expected other developers to be residing in their cluster. The gems that were expected were mostly related to the Ruby on Rails framework, for example, Active Support, Rack, Rails. Additionally, developers seem to have expected other gems on which they have worked through pull requests. Pull requests are a common way of collaborating with others, examples are adding updates or fixing bugs, that are then sent to the project administrators. This does not imply that they list them as an author. Several developers seem to have committed changes to various other projects in which they were not mentioned as an author. Similarly, they would have expected developers that have contributed via pull requests and that were accepted in the codebase.

**Table 1.** *To what extent do you know the other developers we have identified within your cluster?*

| Number of developers | I know a few other developers | I know half of the other developers | I know almost all other developers | I know all other developers | I know only developers with whom I created a gem |
|---|---|---|---|---|---|
| 2-3 | 11.6% ($n$=11) | 1.1% ($n$=1) | 2.1% ($n$=2) | 66.3% ($n$=63) | 18.9% ($n$=18) |
| 4-5 | 22.9% ($n$=8) | 8.6% ($n$=3) | 14.3% ($n$=5) | 45.7% ($n$=16) | 8.6% ($n$=3) |
| 6-7 | 46.2% ($n$=6) | - | - | 30.8% ($n$=4) | 23.1% ($n$=3) |
| 8-9 | 25.0% ($n$=1) | 50.0% ($n$=2) | 25.0% ($n$=1) | - | - |
| 10+ | 84.5% ($n$=196) | 4.7% ($n$=11) | .9% ($n$=2) | - | 9.9% ($n$=23) |

**Table 2.** *To what extent do you personally know the other developers we have identified within your cluster?*

| Number of developers | I do not know any of the other developers personally | I personally know a few other developer | I personally know half of the other developers | I personally know almost all other developers | I personally know all other developers | I personally know only developers with whom I created a gem |
|---|---|---|---|---|---|---|
| 2-3 | 18.9% ($n$=18) | 8.4% ($n$=8) | 4.2% ($n$=4) | 2.1% ($n$=2) | 52.6% ($n$=50) | 13.7% ($n$=13) |
| 4-5 | 28.6% ($n$=10) | 11.4% ($n$=4) | 11.4% ($n$=4) | 20.0% ($n$=7) | 25.7% ($n$=9) | 2.9% ($n$=1) |
| 6-7 | 38,5% ($n$=5) | 7.7% ($n$=1) | - | - | 30.8% ($n$=4) | 23.1% ($n$=3) |
| 8-9 | 25.0% ($n$=1) | 50.0% ($n$=2) | - | 25.0% ($n$=1) | - | - |
| 10+ | 23.3% ($n$=54) | 6.6% ($n$=143) | 2.2% ($n$=5) | - | - | 12.9% ($n$=30) |

**Table 3.** *How did you establish your personal relationship (if any) with other developers within your cluster?*

| Number of developers: | 2-3 | 4-5 | 6-7 | 8-9 | 10+ |
|---|---|---|---|---|---|
| Friend | 20.9% | 16.9% | 11.1% | 16.7% | 19.1% |
| Work at same company | 41.0% | 39.0% | 22.2% | 50.0% | 25.4% |
| Family | 2.2% | - | - | - | 0.2% |
| Open Source Community | 16.5% | 25.4% | 27.8% | 33.3% | 33.7% |
| Business partner | 5.8% | 5.1% | 11.1% | - | 4.8% |
| Schoolmates | 0.7% | 5.1% | 5.6% | - | 2.2% |
| I work alone / no personal relationship | 6.5% | 8.5% | 5.6% | - | 9.2% |
| Other | 6.5% | - | 16.7% | - | 5.3% |

### 4.2 Motivations to Create New Gems

We asked developers what motivates them when creating new gems (projects). The results are shown in Table 4 with the percentage of responses (adjusted for multiple answers). In each grouping of cluster sizes, the majority of developers are motivated by personal needs, i.e. they need to create a certain gem for their own goals. This does not seem to increase or decrease when clusters consist of more developers. Two other motivations that contribute for a some what smaller part are "helping others" and "improving programming skills". The percentage of responses for "helping others" seems to increase as clusters are getting larger. Furthermore, 'fun' was a frequently given answer in the open text box.

**Table 4.** *What motivation(s) do you have for creating new gems?*

| Number of developers: | 1 | 2-3 | 4-5 | 6-7 | 8-9 | 10+ |
|---|---|---|---|---|---|---|
| Personal needs | 65.3% | 58.6% | 66.7% | 48.1% | 57.1% | 57.9% |
| To help others | 13.5% | 14.6% | 13.7% | 22.2% | | 18.9% |
| Monetary rewards | 1.8% | 2.5% | 2.0% | 3.7% | | 3.4% |
| Improve programming skills | 10.8% | 11.5% | 7.8% | 18.5% | 14.3% | 10.6% |
| Recognition | 6.3% | 8.9% | 9.8% | 7.4% | 14.3% | 7.0% |
| Other | 2.2% | 3.8% | - | - | 14.3% | 2.3% |

### 4.3 Contribution/role when Collaborating

Lastly, we asked developers about their contribution when creating gems with other developers. Results are shown in Table 5 together with the percentage of cases, as roles vary among different projects. Sixty percent of the cases create gems solely by themselves, an indicator that large parts of the Ruby ecosystem consist of single developers. This was also noticed from the skewed distribution as discussed in Section 3. The remainder of roles are diverse and make up an almost equal part of the cases. Project leaders are, however, in minority. Possibly an indicator that Ruby projects are small in nature and do not consist of a clear hierarchy of developers that are led by a project leader.

**Table 5.** *What is your contribution when creating gems with other developers?*

| Contribution / role | N | Percentage of responses | Percentage of cases |
|---|---|---|---|
| I create gems solely by myself | 469 | 22.40% | 60.00% |
| Testing | 275 | 13.10% | 35.20% |
| Debugging | 252 | 12.00% | 32.20% |
| Documenting | 207 | 9.90% | 26.50% |
| Lending technical knowledge | 213 | 10.20% | 27.20% |
| Project leader | 133 | 6.40% | 17.00% |
| Core developer | 239 | 11.40% | 30.60% |
| Co-developer | 282 | 13.50% | 36.10% |
| Other | 24 | 1.10% | 3.10% |

## 5 Discussion and Conclusion

In this paper, we have mined data on relationships between developers and gems from Ruby's repository Rubygems.org. We have used Social Network Analysis to uncover underlying sub-communities, or clusters as we have labeled them. Based on this data, we performed qualitative analysis by means of a survey to make a first attempt to find characteristics of various cluster sizes. These clusters were grouped together by the number of developers residing within them. The survey was aimed at finding characteristics of connections or ties between developers and gems. We analyzed if these connections were personal and how they were established. Furthermore, we made an attempt to classify motivations amongst various cluster sizes and the contributions or roles Ruby developers have when collaborating with other developers.

Clusters with few developers make up a large part of the Ruby Ecosystem. Within these small clusters, developers often have personal relationships that are mainly formed through friendship, work and through the open source community. As clusters are increasing in developer size, the relationships (ties) to other developers, personal and non-personal, seem to decline. It seems that Ruby developers collaborate with other developers, but are not aware of collaborations of friend-of-friend relationships. Additionally, conferences are important to stimulate collaborations between developers, as developers have stated that it facilitated in new collaborations. Personal relationships that are formed through the Open Source community seem to grow as clusters are getting larger. Furthermore, the ample part of the Ruby ecosystem consist of single developers who create gems for personal use and do not engage in interaction.

Our explorative study made a first attempt into the classification of FLOSS data in cluster sizes, but it is however far from complete. As developers have stated, relationships can not solely be modeled by looking at information given by project administrators, as this information lacks detailed information on the actual number of developer contributions. A substantial amount of developers are motivated by altruism and like to help others. As a consequence, developers contribute by making pull requests/issues in which they are necessarily not listed as an author by project administrators. This data is, however, available in other common repositories such as Github. Clustering developers and gems

by incorporating this information would yield more information and a better representation of the actual ecosystem structure.

Another important aspect of the Ruby ecosystem is the existence of dependencies between gems. These dependencies can either be runtime dependencies or development dependencies. The first are other gems that are essential to work in real time for end-users, the latter are gems that are necessary for development purposes [12]. Although developers have not directly collaborated with other developers from gem dependencies, incorporating these dependencies would increase the complexity of the ecosystem and possibly provide a more in-depth view of the actual structure.

The data we collected by mining rubygems.org will be uploaded to Flossmole.org to assist other researchers in the study of FLOSS ecosystems.

## 6 Acknowledgement

We would like to thank all the Ruby developers who have filled in the survey and especially Jon-Michael Deldin for his comments to this work.

## References

1. Jansen, S., Brinkkemper, S., Finkelstein, A.: A sense of community: A research agenda for software ecosystems. In: 31st International Conference on Software Engineering, New and Emerging Research Track. (2009)
2. Messerschmitt, D., Szyperski, C.: Software ecosystem: understanding an indispensable technology and industry. The MIT Press (2005)
3. Xu, J., Christly, S., Madey, G.: Application of social network analysis to the study of open source software. In Bitzer, J., Schroder, P.J.H., eds.: The Economics of Open Source Development. Elsevier (2006)
4. Haythornthwaite, C.: Tie strenghts and the impact of new media. In: Proceedings of the 34th Hawaii International Conference on Systems Science, Maui, Hawaii (2001) 1019
5. Gao, Y., Freeh, V., Madey, G.: Analysis and modeling of the open source software community. In: Proceedings of North American Association for Computational Social and Organizational Science Conference (NAACSOS 2003). (2003)
6. Lopez-Fernandez, L., Robles, G., Gonzalez-Barahona, J.M., et al.: Applying social network analysis to the information in cvs repositories. In: International Workshop on Mining Software Repositories. (2004) 101–105
7. Madey, G., Freeh, V., Tynan, R.: The open source software development phenomenon: An analysis based on social network theory. In: Americas conference on Information Systems (AMCIS2002). (2002) 1806–1813
8. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D., eds.: Defining and identifying communities in networks. Number 101, USA, Proc. Natl. Acad. Sci. USA (2004)
9. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69**(2) (2004) 26113
10. Wu, F., Huberman, B.A.: Finding communities in linear time: A physics approach. Eur. Phys. J. **38** (2004) 331–338

11. Blondel, V.D., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment **10** (October 2008) P10008
12. Kabbedijk, J., Jansen, S.: Steering insight: An exploration of the ruby software ecosystem. In: Proceedings of the Second International Conference on Software Business. (2011)
13. Granovetter, M.: The strength of weak ties. American Journal of Sociology **78**(6) (1973) 1360–1380
14. Bastian, M., Heymann, S., Jacomy, M.: Gephi: An open source software for exploring and manipulating networks. In: International AAAI Conference on Weblogs and Social Media. (2009)