Towards the roles and motives of open source software developers

Ruvar Spauwen and Slinger Jansen

Utrecht University
Department of Information and Computing Sciences
Princetonplein 5, 3584 CC, Utrecht, Netherlands
r.a.spauwen@students.uu.nl,s.jansen@cs.uu.nl

Abstract. The software ecosystems of current web browsers include thousands of extensions, which provide additional and customized features for end users and which can generate stronger loyalty towards the browser. Not much research has been done on the development of browser extensions and, in particular, why developers chose to develop for a certain browser. Hence, this research tries to do so by (1) investigating and proposing a set of single platform developer roles and (2) looking more closely at the subset of developers that have developed for multiple platforms, including their behavioral patterns. The selection of browsers consists of Chrome, Firefox, Opera, and Safari, and the researched dataset includes all identified browser extension projects on the web-based hosting service Github between the years 2007 and 2012. The goal of this research is to propose a set of methods that enable clear and meaningful categorization of open source software developers. Consequently, these methods could be seen as stepping stones towards more qualitative-based research, such as: investigating the motives of specific category of developers for contributing to an ecosystem, which can lead to more effective input for controlling or at least influencing an ecosystem's health.

Key words: software ecosystems, internet browser extensions, open source software development, behavioral patterns, developer roles, repository mining

1 Introduction

These days, many companies realize that the total value of a software product can not be defined only by the value of the product itself. Instead, the total value of a software product can be better defined as the sum of a product's value and the values of other products that are interacting with the specific software product [17]. This creates a network of mutual dependencies in which products, companies and services work together to keep the network alive, and such type of network was first described by Messerschmitt and Szyperski in 2003 [14] as a software ecosystem (SECO). In 2009, Jansen, Finkelstein and Brinkkemper proceeded by defining a SECO as "a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the

relationships among them" [10]. Free and Open source software (FOSS) projects tend to have a SECO that forms itself around the projects in which their growth is determined by the amount of developers that are willing to contribute to projects and create new projects [12]. Companies could provide support by offering toolkits and developer platforms, but the success of these types of software projects still depends on the effort of the developers themselves. This research project focusses on free and publicly available software development that is liberally licensed to grant users the right to use, copy, study, change and improve the source code [4].

An important question concerning FOSS development relates to why developers participate in these projects. Research has shown several motives for participation, for instance: "pleasure", "a form of personal rewarding" and "an opportunity to improve technical skills" [2,16]. Other research names reasons like "building trust and reputation", "showing creativity", "advancement through increasingly challenging technical roles" [11], as well as "being generous in sharing time expertise and source code" [1]. Additionally, it often occurs that FOSS developers are involved in several projects: a study of 2002 showed that 5% of the examined FOSS developers were involved in 10 or more projects [6]. Madey et al. [13] examined the importance of these so-called "hubs" or "linchpins", through social network analysis, and showed that the absence of these developers can result in the segregation of networks. However, these studies do not specify if developers are related to one or several platforms and, more specifically, whether developer perform the same role in every SECO or that they might perform different kind of roles.

To the authors' knowledge, in-depth research on the behaviour of these multiple platform developers does not exist. Idu, van de Zande, and Jansen [9] performed an interesting case study on Apple's App Store ecosystem, but their scope is more focused on a single company with multiple sub-ecosystems. On the other hand, Hyrynsalmi et al. [7] do consider the SECOs from different mobile software companies and discuss characteristics of multiple platform developers. Only their research does not elaborate on how developers might behave over time. Therefore, this research attempts to go a step further by using the developer behaviour perspective over time to work towards, possibly valuable, motives of developers for participating in certain FOSS projects.

More specifically, knowledge of behavioral patterns might be useful in a way that it can lead to new methods for influencing a SECO's health; i.e. information about developers' relationships with platforms and other developers over time can be used as an indicator of the robustness of a SECO, which is one of the determinants of measuring SECO health [8]. Also, it could be used to define group stability, connectedness and outbound links, which are part of the metrics Den Hartigh defines to measure robustness [3]. If companies have access to developers' motives, for instance, for choosing to contribute to a specific platform or not, they can use this knowledge for more accurate improvements to their platform, which in return might lead to stronger relationship with current developers as well as more future developers.

Finally, the accessibility of source code, alterations over time and information about contributing developers make FOSS projects suitable for (dynamic) network analysis and they provide new ways of exploring large software ecosystem, as shown by [12]. Additionally, the large number of FOSS projects that are being hosted on open source software hosting websites like Github, Sourceforge and Google Code, increase the reliability of statistical analysis results [18].

The remainder of this paper is organized as follows. In the following section, we present the main research questions and their related sub questions. Subsequently, the research method is discussed in section 3, providing argumentation on the chosen type of research, the selection of cases and collection of data. Section 4 contains the analysis on the gathered dataset and the following results are presented in section 5. Finally, we conclude this paper by giving an overview of the things discussed, point out important limitations and propose several opportunities for future work.

2 Research Questions

The main research questions answered in this paper is: **"How can we divide open source developers of browser extensions into distinctive categories and roles?"**. From the main research question, the following sub questions are derived:

1. **What are the characteristics on which we can categorize developers in the separate browser ecosystems?** - By looking closely at the developers and their activities per platform, we will be able to extract the most important characteristics on which the developers can be categorised.
2. **How can we combine the single platform developer characteristics into roles** - In order to get a better overview of the distributions of characteristics per browser, we have to combine them into meaningful roles.
3. **What are the combinations in which developers are or have been connected to more than one ecosystem?** - Before we can research multiple platform developers and their behaviours, we first need to locate them and determine if different behavioral patterns exist.
4. **How can we quantify the different types of multiple platform developers?** - We need to design a model and visual representation that can give meaning to the discovered behavioral patterns and that enables clear and valuable categorization of multiple platform developers.

3 Research Method

In the following sections, we elaborate on the applied research method for gathering the data and performing the analysis. This includes: descriptions of the selected cases and the data source, explanation on how we performed the actual data gathering and on our attempts on ensuring the validity of the research.

3.1 Case selection and description

The selection of cases for the research consisted of the following three criteria. First, the cases had to be part of the same type of software ecosystem to be able to compare them. Second, access to case related software development projects had to be available (e.g. source code, developers and change history) and, finally, these projects needed to be recognizable during the data mining process.

The selection of cases consist of the following web browsers: Chrome (Google), Firefox (Mozilla), Opera (ASA) and Safari (Apple). The reason why Internet Explorer is not selected, will be clarified in the following section. Browser extensions are small software programs that can modify and enhance the functionality of the browser and can be created by internal (e.g. the browser company) and external developers, like commercial companies or (collaborating) independent developers. Unfortunately, there is no current and complete overview of the total number of available extensions per browser and, although all browser and their extensions are freely available and the browser manufacturers actively support FOSS development, not all extensions have public source code. Therefore, to ensure objectiveness and completeness of the research, the collection of extensions is narrowed down to only those that are developed and stored online in Github repositories. Github is a website that provides source code hosting for repositories that use the Git revision control system and provides several social networking features to improve collaboration among developers. Additionally, the website offers both paid plans for private repositories as well as free accounts for FOSS projects. As mentioned in section 1, alternatives to Github and the Git revision control system exist, but due to the time scope of this project and the differences in systems (e.g. what kind of data is stored per repository), this research only includes the most popular combination [5, 15]. Furthermore, Github provides the necessary structure to be able to divide the dataset into extensions, unique developers and commits, including details about the size and the time of the commit and its author. Consequently, the following relationships among the entities are ensured: developers can create multiple commits for multiple extensions and an extension can be related to one or more developers.

3.2 Data Gathering

In order to be able to perform the analysis, quantitative data needed to be collected for each of the cases. Although Github provides access to sufficient data, the repositories are stored in such a manner that they can not be categorised instantly. For instance: the search-string “Firefox extension” produces many false-positives, because the search function of Github includes a repository description field, which often consists of misleading data (e.g. “this is an alternative to the Firefox extension”). Fortunately and in contrary to Internet Explorer, each of the selected browsers have specified a so called “manifest” file, that needs to be included in order to make the extension executable for the browser. Table 1 provides an overview of the manifest files considered for determining if a repository contains a manifest file. Unfortunately, in practice

it appeared that not all repositories have included a manifest file (e.g. yet, anymore or never had one). Also, it appeared that older versions may have used different manifest file types (e.g. Firefox), and that for some platforms only the file type and not the combination of filename and type is mandatory. The last case might increase the chance for false-positives if a file type is also used in other programs than browser extensions. Therefore, when we considered a possible manifest file, we also looked at values inside the file, as can be seen in the third column of Table 1. Because not all of these values are specified by the browser manufacturers as being mandatory, a manifest file was considered valid if and only if it contained at least two of its related values. The manifest files that have no values specified were considered as special cases, for instance: if the scraper script did not find a Firefox "install.rdf" or "manifest.json" file but it did find a "chrome.manifest",".manifest" or ".xpi" file, then the repository would be checked manually.

Platform	**Files**	**Values**
Chrome	manifest.json	"name", "version", "description"
Firefox	install.rdf	"xlmns:em=http://www.mozilla.org/2004/em-rdf#"
		"<em:id>", "<em:version>", "<em:name>", "<em:description>"
	chrome.manifest	n/a
	manifest.json	"name", "version", "description", "author"
	.manifest/.xpi	n/a
Opera	config.xml	"xlmns=http://www.w3.org/ns/widgets", "<name", "<name>", "<description>", "<author>"
Safari	*.plist	"<plist>", "<plist", "<dict>", "<key>"
	*.safariextz	n/a

Table 1: Description of manifest files and related values per platform

The list of candidate repositories was created by searching on Github with different combinations of keywords, like: "Chrome extension 2011-01-01...2011-12-31". Because there is no consensus on the term "extension", alternatives like "add(-)on" and "plug(-)in" were considered as well and subsequently combined with the four platforms and different date intervals. To clarify, a date interval relates to those repositories that were created in that specific period. It was necessary to include this because Github only returns a thousand repositories per search query. Additionally, only repositories that are no Forks of other repositories were considered, because these repositories would skew the data since it is impossible to combine their commits into unique contributions. Finally, due to the design of Github it was decided to count a commit in a repository containing extension for multiple platforms (e.g. Chrome and Firefox) for both platforms.

Different methods are used for collecting the actual data. Due to the extensive checks for manifest files, it was not possible to use Github's developer API for the initial filtering of repositories and so HTML scraping methods were used for this. When this was finished, we were able to use the quicker Github API to retrieve a vast amount of additional information on the developers and commits.

4 Analysis

This section provides the analysis performed on the dataset retrieved from Github, so each question posed in section 2 can be assessed. But first, we provide a description on how the final dataset was established.

The combinations of keywords used to search on Github produced a result of more than 30.000 separate repositories of which almost 9.000 passed one of the platform specific manifest-file checks. Next, this number was reduced to 7.749 extensions from 7.564 unique repositories, by first automatically removing the extensions that had no commits in the period of interest and the repositories that were incomplete. The period of interest was set between 2007 and 2012, because there were only a few (Firefox) manifest-containing repositories before that period. In Chapter 6 we argue that this selection, which is also implemented in the metrics and normalization, can be improved due to the fact that the other three browsers started supporting extensions in 2009 and 2010, which is more than five years after Firefox. Then, a second filter was applied on the dataset by manually controlling an arbitrary selection of outstanding cases, like: odd date values (e.g. "1/1/1970"), extremely large commits (e.g. bulk import) and the default developer accounts "invalid-email-address", "unknown" and "(no author)". Finally, the remaining set of extensions was as follows: 4.746 (Chrome), 2.032 (Firefox), 772 (Safari) and 199 (Opera). Furthermore, more than 340.000 commits and nearly 10.000 developers were collected. The performed analysis is mostly focused on relative values because there is no complete information on the total number of extensions outside of Github and, despite all measures during the gathering of the data, it is not certain whether every record is a finalized extension (i.e. available at one of the official extension markets). Fortunately, the obtained dataset is sufficiently large for dividing developers into different categories and for observing relationships among developers and different platforms.

4.1 Single Platform Developer Roles

This sections explains the design of the single platform developer roles and includes an overview of the distributions per platform. The roles are based on a set of metrics which are aggregated into the following three scores: **T**, **N** and **C**. We argue that a valuable method to categorize developers is provided by a combination of the developer's connectedness with other developers and extensions (**N**), the frequency and intervals over which the developer creates commits (**T**) and the number and size of these commits (**C**). The base metrics are inspired on metrics from related literature, but they are modified and combined in such a manner that further research is necessary to validate their design and the results.

The three scores range from 0 to 1 and they can be visually represented by a 3-dimensional graph divided into eight same-sized cubes, or "octants". Each one of these cubes has distinctive properties that can be related to a specific role. Table 2 gives an overview of these eight roles, including their label and specific properties. The first property of the roles relates to the score **T**, where the value *Occasional* or *Regular* is selected depending on a developer's average

contribution time per extension combined with the total number of days between a developer's first and last commit. Next, the scores **N** and **C** relate to the second property of the role, where specific combinations of scores lead to the values *Adjuster*, *Investor*, *Networker* or *Collaborator*. For instance, the values *Adjuster* and *Investor* are selected when the developer has a low **N** score combined with a low, respectively high **C** score, and the values *Networker* and *Collaborator* are assigned when the **N** is high and the score **C** is low, respectively high.

Labels	**Roles**	**T**	**C**	**N**
a	Occasional Adjuster	0,0 - 0,5	0,0 - 0,5	0,0 - 0,5
b	Occasional Investor	0,0 - 0,5	0,5 - 1,0	0,0 - 0,5
c	Occasional Networker	0,0 - 0,5	0,0 - 0,5	0,5 - 1,0
d	Occasional Collaborator	0,0 - 0,5	0,5 - 1,0	0,5 - 1,0
e	Regular Adjuster	0,5 - 1,0	0,0 - 0,5	0,0 - 0,5
f	Regular Investor	0,5 - 1,0	0,5 - 1,0	0,0 - 0,5
g	Regular Networker	0,5 - 1,0	0,0 - 0,5	0,5 - 1,0
h	Regular Collaborator	0,5 - 1,0	0,5 - 1,0	0,5 - 1,0

Table 2: Overview of roles and related scores

After defining the roles and calculating the scores, we selected for each platform the top 10 percent of developers having the highest combined **T**, **N** and **C** scores. Despite the significant differences in the resulting number of developers per platform (e.g. 576, 357, 21 & 98), these subsets provide the most interesting developers and a correct comparison is ensured by selecting the same relative number of developers for each platform. Next, the developers were given a role based on their **T**, **N** and **C** scores and the resulting distributions are shown in Figure 1. It can be seen that Chrome, Firefox and Safari appear to be surprisingly similar: they all have the *Occasional Adjuster* as the most occurring role, followed by the *Occasional Networker*, the *Regular Adjuster* and a small portion is taken by the *Regular Networker* or *Occasional Collaborator* roles. Regarding Opera, we can see that by far the largest part is occupied by the *Occasional Networker* role. This might be caused by the fact that many of the Opera extensions in our dataset share their repository with extensions for other platforms and this results in a more than average number of connections with other developers.

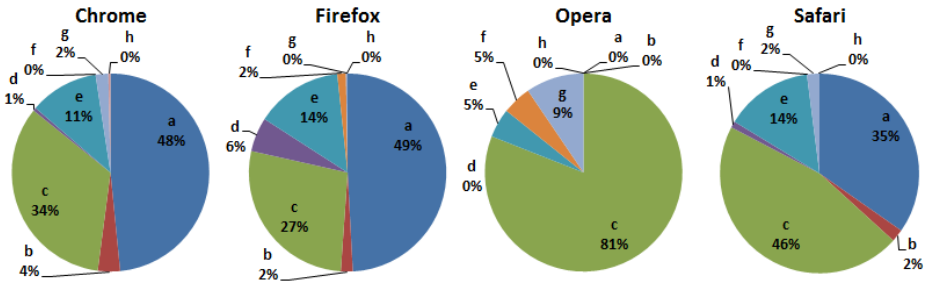

Fig. 1: Top 10 percent distribution of single platform developer roles per platform

4.2 Multiple platform Developers

The second part of the research only considers the subset of multiple platform developers, or MPDs, which include the developers that have contributed to different extensions for more than one type of browser platform, to a single repository that contains extensions for more than one type of browser platform, or both. The dataset contains a total of 743 MPDs of which there are 629, 69 and 45 developers having a connections with 2, 3 and 4 different platforms respectively. An overview of the different combinations and their occurrences can be seen in Table 3. It is clear that the combination Chrome-Firefox is the most occurring, but we can also see that the combination Chrome-Safari is quite significant with 136 developers. It is clear that combinations of platforms containing Opera occur the least often, but percentage-wise, the set of Opera developers consist of the largest number of multiple platform developers (38%) compared to Chrome (12%), Firefox (16%) and Safari (31%).

Platform	**Combinations**										
Chrome	*	*	*		*	*	*				*
Firefox	*		*	*	*		*	*		*	
Opera					*	*	*	*	*	*	*
Safari		*	*	*	*				*	*	*
#	409	136	59	59	45	19	7	3	3	2	1

Table 3: Number of developers per multiple platform combination

A quadrant-based model was designed that rates developers upon the following two criteria: *Fluctuation over Time Active* and *Amount of Contribution*. The first criterium relates to how often a developer switches to another platform or a combination of platforms: this is a combined score that looks at the fluctuation per month, fluctuation per year and the total length of activity (e.g. first and last commit) of the developer during the previously mentioned main scope of this research. Two different time interval levels were considered, because: analysis showed that not every developer creates a commit every month, but if you would only look at intervals per year the granularity would be too low to discover valuable fluctuations. The total time of activity of a developer is included, so that fluctuations of developers that have a longer activity time are weighed heavier. The second criteria relates to the number of commits combined with the size of these commits, regarding number of additions and deletions. A logarithmic normalization was used on the number of additions and deletions, because the dataset contains a small collection of high outliers which often consist of less valuable bulk import commits, and it enables a better distinction between the large number of developers with relatively low "addition" and "deletion" values. The metrics used in this model are partly based on existing metrics, but the combination of them and output of results are based mostly on instinct and have not yet been extensively validated. Therefore, the analysis presented in section 5.2 is solely meant for giving an indication of the model's possibilities.

5 Results

The following sections elaborate upon the results in the context of the research questions posed in section 2. The first two sub questions are depicted in section 5.1 and section 5.2 elaborates on the findings related to the remaining two.

5.1 Single Platform Developer Roles

The distributions of the developer roles, introduced in section 4.1, provide an overview of certain properties of developers and their occurrences as well as similarities and differences between platforms. Figure 1 shows that most Chrome, Firefox and Safari developers score low on the length and size of their contributions. Furthermore we can see that for both Safari and Opera the most occuring role is the *Occasional Networker*. But as mentioned in section 4.1, we believe that the set of Opera developers is skewed because a significantly less amount of repositories containing only an Opera extension is present in the dataset. The reason for this is uncertain and therefore interesting to further investigate upon.

5.2 Multiple Platform Developers

The model introduced in section 4.2 has been applied to the collection of multiple platform developers in the dataset. A graphical representation of the results can be seen in Figure 2. To ensure the clearness of the graph, the developers are grouped based on similar ranges (e.g. $0,1 < x \leq 0,2$ and $0,5 < y \leq 0,6$) and represented as a bubble in the graph, with a size relative to the number of developers in the group. Additionally, the bubbles that contain more than one developers have a label showing the number of developers inside them and, for explanatory purposes, IDs of several outstanding developers have been included.

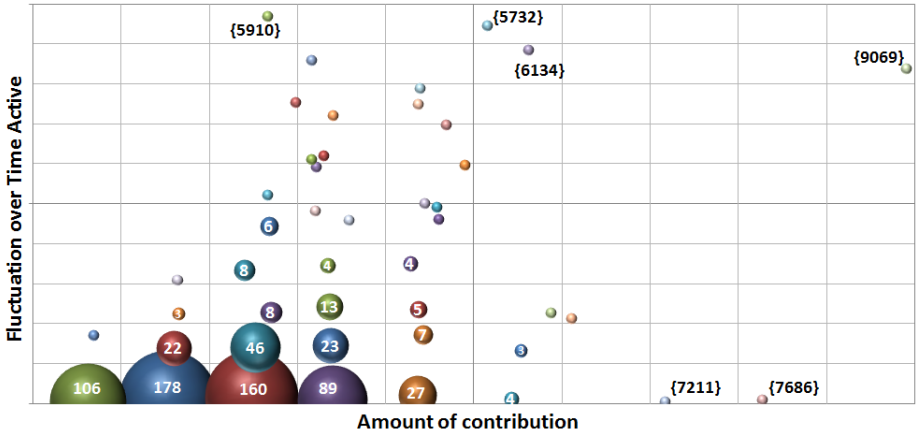

Fig. 2: Distribution of the multiple platform developers

Figure 2 shows that most developers score low on both criteria (e.g. located in the bottom left quadrant). This implies that these developers have a relatively low contribution combined with a stable loyalty to a certain combination of platforms. More interesting are the developers positioned in the other quadrants. For instance, if we look more closely at the three most right developers, named "jcutler", "2d1" and "johnjbarton". Table 4 shows that "johnjbarton" has a average number of commits, but his total number of additions and deletions is significantly higher than the other two and therefore his *Amount of Contribution*, despite the logarithmic normalization, is valued higher. The table shows also that on the level of **Years**, both "jcutler" and "2d1" did not change their combination of platforms. Developer "johnjbarton" does have fluctuations on the level of **Years**, namely: between 2007 and 2010 he developed only for Firefox, then from 2011 both for Firefox and Chrome and, finally, in 2012 only for Chrome. More specific, it shows that 2011 is divided into two periods, as the **Months** value also shows: until July he developed for Firefox and later on only for Chrome. Given his high scores on both criteria, it would be interesting to investigate why "johnjbarton" switched to Chrome after four years of loyalty towards Firefox.

Developer		**Fluctuation over Time**			**Commits**		
ID	**Name**	**Years**	**Months**	**Time (d)**	**Commits**	**Additions**	**Deletions**
5732	jrburke	2	3	2017	2087	1126540	264572
5910	georgebrock	4	3	82	15236	263	79431
6134	Ajnasz	2	6	1699	3805	91528	109521
7211	jcutler	0	0	650	2087	609714	425510
7686	2d1	0	2	286	6957	110735	21584
9069	johnjbarton	2	1	1933	3805	3918354	2095710

Table 4: Overview of multiple platform developers from right quadrants

Next, some remarks can be made regarding the top three developers from Table 4, namely: {5732}, {5733} and {6134}. These IDs relate to the developers named "jrburke", "slightlyoff" and "Ajnasz" respectively, and they are part of the developers with the highest *Fluctuation over Time Active* values combined with significantly high *Amount of Contribution* values. They all have been active developers for at least 5 years and they all have created a commit in the last month of 2012. Furthermore, although they have only developed for the Chrome and Firefox platform, they are all related to multiple different extensions. For instance, "jrburke" has created commits for a total of 8 different extensions. Another remarkable observation from the dataset, is that the developer "slightlyoff" was not active during the years 2009 and 2010 and when he started developing again, his loyalty changed from Chrome to Firefox. Regarding the developer "Ajnasz", we observed that his main loyalty is towards Firefox, but we saw that in 2010 and in 2011, during different periods, he created commits for Chrome extensions, but none of these periods lasted longer than one month. This behaviour is quite different than the behaviour found at the other mentioned developers.

6 Discussion

First of all, the amount of collected research data and the number of sources can be increased: as mentioned in section 3 only repositories hosted on Github have been considered, while there are several other sources (e.g. SourceForge, GoogleCode and CodePlex). Furthermore, the dataset could be extended within the context of Github: although multiple combinations of keywords have been used, it is possible that not all significant extensions have been retrieved. For example, names and documentation in non-English languages could withhold extensions from being discovered; Also, plural forms of keywords could be included and a list of extensions names available on each of the official Extension web stores could be created, although there are some additional challenges to this method (e.g. differences in names between project development name and web store version). To improve the completeness of the dataset and results, extension for Microsoft Internet Explorer, the second most used web browser in the world, should be included. If this was possible it would provide additional comparison materials and most likely more multiple platform developers.

Further improvements can be made regarding the contents of the current dataset. For example, when looking at repositories containing multiple extensions for the same platform: currently, a maximum of one extension per repository per platform combination is considered. Accepting more extensions would greatly increase the data gathering time, because then the scraper script has to consider all the directories of a repository. Another issue is that we are not certain whether all extensions in the dataset are actual finished extension. An effective method for validating all the extensions with the browsers' official markets is needed to improve the ratio of installable extensions in the dataset. One can argue that unfinished, failed or test projects are some kind of contribution to a certain platform, but it would be wise to additionally investigate manifest-containing repositories and verify that they are truly related to a web browser.

7 Conclusion

As discussed, this research is mostly intended as a step towards qualitative research: due to designs of the data source and type of SECOs, many validity issues remain which negatively influence quantitative analysis. Besides providing extensive insight into these issues, the dataset was analysed in such a manner that the results could lead the way to succeeding qualitative research. This was done by suggesting different developer roles and how these were distributed per platform. For the group of multiple platform developers we presented an overview of platform combinations, a quadrant based on specific criteria and we discussed several outstanding examples of developers and their behavior patterns. Based on these examples, we proposed several questions on which future work can be based on. Additionally, the following questions can be considered when including the single platform roles: "How do the single platform roles of a multiple platform developer compare to each other?" and "Does a developer changes roles if you

look at specific periods?". These questions should be interpreted as guidelines: guidelines on how roles, relationships among developers and behavioral patterns could be used to extract the motives of certain developers for participating in and contributing to certain development projects. We hope that these guidelines lead to more reliable methods for influencing the health of open source SECOs.

References

1. M Bergquist and J Ljungberg. The power of gifts: organizing social relationships in open source communities. *Information Systems Journal*, 11(4):305–320, 2001.
2. K Crowston and B Scozzi. Open source software projects as virtual organisations: competency rallying for software development. *IEE Proceedings - Software*, 149(1):3–17, February 2002.
3. E Den Hartigh, M Tol, and W Visscher. The Health Measurement of a Business Ecosystem. *Ecosystems*, 2783565:1–39, 2006.
4. J Feller and B Fitzgerald. *Understanding open source software development*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
5. K Finley. Github Has Surpassed Sourceforge and Google Code in Popularity. `http://readwrite.com/2011/06/02/github-has-passed-sourceforge`, 2011.
6. A Hars and S Ou. Working for free? Motivations of participating in open source projects. *International Journal of Electronic Commerce*, 6(3):25–39, 2002.
7. S Hyrynsalmi, T Mäkilä, A Järvi, A Suominen, M Seppänen, and T Knuutila. App Store, Marketplace, Play! An Analysis of Multi-Homing in Mobile SECOs. In *Proc. 4th Intern. Workshops on SECOs*, volume 879, pages 59–72. CEUR, 2012.
8. M Iansiti and R Levien. Keystones and Dominators: Framing the Operational Dynamics of Business Ecosystems. page 84, 2004.
9. A Idu, T Van De Zande, and S Jansen. Multi-homing in the Apple ecosystem: Why and how developers target multiple Apple App Stores. In *Proc. of the Intern. Conf. on Management of Emergent Digital EcoSystems*, pages 122–128. ACM Press, 2011.
10. S Jansen, A Finkelstein, and S Brinkkemper. A Sense of Community: A Research Agenda for Software Ecosystems. *31st International Conference on Software Engineering Companion Volume*, (C):187–190, 2009.
11. C Jensen and W Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *Proceedings of the 29th international conference on Software Engineering*, pages 364–374, 2007.
12. J Kabbedijk and S Jansen. Steering insight: An exploration of the ruby software ecosystem. In B Regnell, I Weerd, and O Troyer, editors, *Software Business*, volume 80, pages 44–55. Springer Berlin Heidelberg, 2011.
13. G Madey, V Freeh, and R Tynan. *Free/Open Source Software Development*. IGI Global, July 2004.
14. D Messerschmitt and C Szyperski. *Software Ecosystem: Understanding an Indispensable Technology and Industry*, volume 1. The MIT Press, 2003.
15. I Skerrett. Eclipse Community Survey Result for 2012. `http://ianskerrett.wordpress.com/2012/06/08/`.
16. G Von Krogh, S Spaeth, and K Lakhani. Community, joining, and specialization in open source software innovation. *Research Policy*, (7):1217–1241.
17. L Xu and S Brinkkemper. Concepts of product software. *European Journal of Information Systems*, 16(5):531–541, 2007.
18. RK Yin. *Case Study Research: Design and Methods*, volume 5. Sage Pub., 2009.