The Social Side of Software Platform Ecosystems

Cleidson R. B. de Souza
Vale Institute of Technology
and Federal University of Pará
cleidson.desouza@acm.org

Fernando Figueira Filho
Federal University of Rio
Grande do Norte
fernando@dimap.ufrn.br

Müller Miranda
Federal University of Pará
mulgsm@gmail.com

Renato Pina Ferreira
Federal University of Pará
renpina@gmail.com

Christoph Treude
University of São Paulo
ctreude@ime.usp.br

Leif Singer
University of Victoria
lsinger@uvic.ca

ABSTRACT

Software ecosystems as a paradigm for large-scale software development encompass a complex mix of technical, business, and social aspects. While significant research has been conducted to understand both the technical and business aspects, the social aspects of software ecosystems are less well understood. To close this gap, this paper presents the results of an empirical study aimed at understanding the influence of social aspects on developers' participation in software ecosystems. We conducted 25 interviews with mobile software developers and an online survey with 83 respondents from the mobile software development community. Our results point out a complex social system based on continued interaction and mutual support between different actors, including developers, friends, end users, developers from large companies, and online communities. These findings highlight the importance of social aspects in the sustainability of software ecosystems both during the initial adoption phase as well as for long-term permanence of developers.

ACM Classification Keywords

H.5.3 Information Interfaces and Presentation (e.g. HCI): Group and Organization Interfaces—*Computer-supported cooperative work*

Author Keywords

Software ecosystems; Social aspects

INTRODUCTION

Software ecosystems, as an approach for large-scale software development, have widened the focus of previous efforts to encompass the relationships among different actors that all rely on services provided by a common software platform. They involve a much broader set of actors, such as service and infrastructure providers, end users and external developers. An example of a software ecosystem is the iOS operating system. In this case, Apple provides iOS with a very large set of application programming interfaces (APIs) on top of which extensions (apps) can be built. External software developers use Apple's APIs to develop their own apps that will run on iOS, and consequently on iPhones and/or iPads. Other examples of successful ecosystems in different domains include Android, Apache, Eclipse, Linux, GNOME, Facebook, SAP, and Windows Kinect.

In a systematic mapping of the literature, Barbosa and colleagues [3] argue that software ecosystems include three dimensions: the technical, the business and the social dimension. The *technical* dimension is related to the "common software" or the software platform in which the ecosystem is embedded. The *business* dimension is related to business models, licensing and partnering strategies, benefits from getting involved in an ecosystem, among other aspects. Finally, the *social* dimension focuses on how the different ecosystem actors relate to each other in order to achieve their goals.

However, not all software ecosystems are alike. In a different study, Manikas and Hansen [22] classify software ecosystems in: (i) *proprietary* and (ii) *free and open source software - FOSS* ecosystems. In a proprietary ecosystem, source code and other artifacts are protected because they are products that generate income. This often means that new developers who want to join the ecosystem might need to go through a verification process. Furthermore, an organization governs the evolution of the ecosystem and defines the rules of access to the platform and the availability of the generated product (apps). In a FOSS ecosystem, source code is freely available since actors do not necessarily participate in the ecosystem to obtain financial returns, therefore, no formal verification is necessary for new developers. One example of a proprietary ecosystem that requires verification is the iOS, that demands the payment of a fee to publish apps at the App Store, while examples of FOSS ecosystems include Apache and Eclipse. In practice, most ecosystems combine aspects of both types of ecosystems [22], but this distinction is important as we will discuss in the rest of the paper.

Manikas and Hansen [22] conclude that most publications about *FOSS* ecosystems focus on their technical and social aspects, whereas publications about *proprietary* ecosystems focus mostly on their business aspects. In other words, there is a gap in the literature about the social aspects of *proprietary* software ecosystems. Previous work [14] on FOSS ecosystems suggest that social aspects (e.g., previous ties among software developers) have an impact on their attractiveness, i.e, "the

CHI'16, May 07-12, 2016, San Jose, CA, USA
 ISBN 978-1-4503-3362-7/16/05$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858431

existence and the amount of prior collaborative relations ... [among software developers] ... do increase the probability that an OSS project will attract more developers". This ability to attract new developers to an ecosystem is crucial for its sustainability, i.e. to its survival. Therefore, studying the social side of proprietary software ecosystems is important because it can shed light on aspects that might influence their sustainability. A sustainable software ecosystem is one "that can increase or maintain its user/developer community over longer periods of time and can survive inherent changes such as new technologies or new products (e.g., from competitors) that can change the population (the community of users, developers etc)" [10]. Sustainability then includes the adoption and permanence of software developers in a particular ecosystem [30]. By *adoption*, we mean a developer's decision to start developing for a particular ecosystem [20], while *permanence* means "the continued participation of developers in a software ecosystem".

This paper fills a gap in the literature by focusing on how social aspects contribute to the sustainability of proprietary software ecosystems. We collected data using different research methods (semi-structured interviews and an online survey) with different groups of informants. Interviews and survey answers were coded using Grounded Theory techniques [5, 7]. Our results point at a complex social system based on continued interaction and mutual support between different actors, thus highlighting the importance of social aspects in software ecosystems both during the initial adoption phase and the long-term permanence of developers, therefore contributing to the sustainability of a software ecosystem.

The rest of this paper is organized as follows. In the next section, we present an overview of software ecosystems. Then, we describe the methodology we used for our data collection and analysis. After that, our results are presented, followed by a discussion in the context of the literature. Finally, we present our final remarks.

BACKGROUND

Defining Software Ecosystems

In the software development literature one can find several definitions of software ecosystems [22]. For instance, Bosch and Bosch-Sijtsema [4] argue that: "a software ecosystem consists of a software platform, a set of internal and external developers and a community of domain experts in service to a community of [end] users that compose relevant solution elements to satisfy their needs". This is the definition that we will use in this paper as it clearly describes the actors involved in the ecosystem. In the iOS ecosystem, the software platform is the iOS operating system and its associated APIs, *internal* or *keystone* developers are Apple employees who develop the iOS platform [16], while *external* developers and domain experts are responsible for developing apps to the platform which will be purchased by end users.

A different definition is presented by Scacchi [29] who talks about multi-project software ecosystems, whereby "ongoing development and evolution of one free and open source software system gives rise to propagated effects, architectural dependencies, or vulnerabilities in one or more of the projects linked to it". Examples in this case include the Apache ecosystem [17]. Other authors even use the term ecosystem for a set of loosely interrelated projects that are hosted on the same web-domain like the GNOME ecosystem [23].

Types of Software Ecosystems

The different definitions of software ecosystems can be better understood when types of ecosystems are discussed. As mentioned previously, a very simple and effective classification distinguishes between *proprietary* and *FOSS* ecosystems [22]. It should be noted that this distinction between proprietary and FOSS ecosystems neglects a higher number of variations since most real software ecosystems could be categorized as hybrid ecosystems, i.e., ecosystems that combine elements of both types [22]. In other words, to label an ecosystem as FOSS or proprietary is rarely a binary decision, in fact it is generally a matter of how much the platform is open (FOSS) or closed (proprietary). Openness is the degree to which a platform owner allows the platform users to interact with the platform, as well as to view, extend or change its components. Openness depends on different technical and business aspects such as platform architecture, platform accessibility, platform transparency, licensing, marketing policy, among other factors [2]. And, more importantly, as we will discuss, openness directly influences the social aspects of software ecosystems.

Software Ecosystem Dimensions

In a systematic mapping of the literature Barbosa et al. [3] argue that software ecosystems, despite being proprietary or FOSS, need to be understood in three dimensions: the technical, the business and the social dimension. The technical dimension is related to the software platform and the overall infrastructure in which the ecosystem is embedded. An example of this approach is Bosch and Bosch-Sijtsema [4] who focus on software architecture challenges for ecosystems and solutions for these challenges. The business dimension is related to business models, licensing and partnering strategies, among other financial aspects. In this case, an example is Adner's work [1] who discusses tactical aspects and strategies that should be taken into account when designing a software ecosystem. Finally, the social dimension focuses on how the different ecosystem actors relate to each other in order to achieve their goals. For instance, Fricker [13] proposes a model based on negotiation and network theory for analyzing and designing requirements in a software ecosystem.

The Sustainability of Software Ecosystems

Another important aspect in the study of software ecosystems is their sustainability or health [10, 22]. Attracting external developers to an ecosystem can make a crucial difference, but only if the participation of developers is sustained, that is, if developers join an ecosystem and keep developing for it. Dhungana et al. [10] define a *sustainable* software ecosystem as one "that can increase or maintain its user/developer community over longer periods of time and can survive inherent changes such as new technologies or new products (e.g., from competitors) that can change the population (the community of users, developers etc)". Sustainability includes

both *adoption* and *permanence* of software developers in an ecosystem. By *adoption*, we mean a developer's decision to start developing for a particular ecosystem, while *permanence* means "the continued participation of developers in a software ecosystem"[30].

Sustainability and Social Aspects of Software Ecosystems

Sustainability is a major aspect of software ecosystems. Using Barbosa et al.'s dimensions, we identified related work that focuses on the *business* (e.g., [1]) and *technical* (e.g., [8]) aspects that are being studied in the context of sustainable ecosystems. These previous works can be applied to both proprietary and FOSS ecosystems. However, previous research on the influence of *social* aspects in the sustainability of an ecosystem has largely focused solely on *FOSS* ecosystems. An example is Jergensen et al. [18] who studied progressive developers' participation in the FOSS ecosystem GNOME. In addition, Shah [30] found that, for some developers, motivation to work in a FOSS project evolves over time and participation becomes a hobby, but these developers who see their participation as a hobby are critical to the long-term sustainability of the software ecosystem as they take on tasks that might otherwise go undone and work to improve the source code. Similarly, Draxler and colleagues [11] studied software developers from small software enterprises who were searching, installing and configuring new Eclipse extensions (plug-ins). They focused on the potential of collaboration for learning about new plug-ins and tailoring software tools to developers' needs in the context of the FOSS Eclipse ecosystem. Finally, recently Steinmacher et al. [32] presented 13 social barriers that make it difficult for newcomers to join free or open software projects. In short, in *FOSS* ecosystems, there is a clear relationship between the sustainability of an ecosystem and its social aspects.

Research on social aspects of *proprietary* ecosystems is very limited. One of the few examples is Yu's and Deng's [36] study about the "strategic dependencies [relationships] between software vendor, third party developers, and end-users" aiming to explore and reason about alternate ways for achieving strategic goals for each actor. However, this is done at the conceptual level, i.e., without empirical data. More recently, Karhu et al. [19] suggest that different mobile ecosystems choose to establish different relationships with their stakeholders: some of them are based more on competition, while others are more collaborative. Note that this work is both about the business and the social aspects of proprietary ecosystems, but, the social is related to the relationships among *companies*, not individual developers. Similarly, Koch and Kerschbaum [20] focus on developers' motivations for joining a particular mobile ecosystem. They report that their major motivations are the experience of fun, the intellectual stimulation and the opportunity to learn new skills. The potential financial gain from participating in a mobile ecosystem is not as relevant. In other words, social (personal) aspects are implicitly addressed in this work, since its major focus is on the business aspects.

The difference between the number of studies about the social aspects of FOSS vs proprietary ecosystems might be explained by the openness of the ecosystem. As mentioned before, openness is the degree to which a platform owner allows the platform users to interact with, view, extend or change its components [2]. Openness influences the extent to which an ecosystem is closer to an FOSS or a proprietary ecosystem. In a closed (proprietary) ecosystem, developers are not able to extend or change the components of the software platform, while in an open (FOSS) ecosystem this is possible. This has important implications: in a FOSS ecosystem, internal or external developers are likely to interact because they need to coordinate changes in the platform since their code is interdependent, i.e., changes in one part of the platform might impact other parts. Meanwhile, in a proprietary ecosystem, no changes are allowed in the platform, therefore, developers' code will only depend on the software platform but not on each other's, which means that they are unlikely to interact among themselves.

Research Contribution

The previous paragraphs suggest that despite the importance of social aspects for the sustainability of software ecosystems, these aspects have been studied mostly in the context of *FOSS* ecosystems. There is a lack of knowledge about social aspects in *proprietary* ecosystems and, as a consequence, their impact in the *sustainability* of these ecosystems. The goal of this paper is exactly to address this gap by answering the following research question: *how do social aspects influence the sustainability of proprietary ecosystems?*. We will answer this question through an empirical study that focuses on software developers' adoption and permanence in a particular software ecosystem. The detailed approach used to answer this research question is presented in the following section.

METHODOLOGY

In order to examine our research question, we conducted a qualitative study. Initially, we investigated the many different aspects that could influence developers' decisions to join and remain in a software ecosystem. These initial results suggested the important role of the social aspects in the adoption and permanence of a developer in a particular ecosystem, in addition to other business and technical aspects. Then, we focused on the social aspects through additional data collection and analysis. Our data is based on semi-structured interviews and on responses from a survey we distributed to software developers. A summary of our data collection methods is presented in Table 1. Each data collection step and the data analysis used in this research are presented in details below.

Initial interviews

Initially, we conducted semi-structured interviews with nine mobile developers (eight men and one woman) who were active mobile developers. Our interviewees were between 19 and 29 years old, and they were all from Brazil. Two were students (one undergraduate and one graduate) with less than two years of experience developing mobile apps. The others were professionals with over two years of experience in mobile development. The interviews were conducted in the period of August to December 2013, through a convenience sample based on our personal contacts.

Study	Geography	Sample size	Platform	Format
Initial interviews	Brazil	9	Android (8), iOS (3), both (1)	Semi-structured interviews
Survey	USA (24), Canada (4), Brazil, India, Netherlands, UK (3)	83	Android (50), Web (40), iOS (39)	15 questions: 7 about work experience, 5 about specific app, 3 about the software development process (open ended)
Additional interviews with Brazilian developers	Brazil	9	Android (9)	Semi-structured interviews focusing on social aspects
Additional interviews with international developers	US, Brazil, Venezuela	7	Android (4), iOS (1), both (1), Windowsphone (1)	Semi-structured interview focusing on social aspects (same guide as previous interviews)
Validation interview	Confidential	1	Confidential	Semi-structured interview by email

Table 1. Summary of our data collection methods.

We used Rogers' diffusion of innovations theory [28] as a starting point for our initial interviews. Rogers explains the diffusion of innovations through four different aspects namely, (i) the innovation itself, (ii) the communication channels used to transmit information about the innovation, (iii) the social system in which the innovation and the adopters are embedded, and (iv) the time scale in which the innovation is adopted. Based on these aspects, we created an interview guide. All interviews were recorded and later transcribed. We used coding techniques from Grounded Theory for data analysis [7]. Open and axial coding resulted in the identification of 265 different categories that covered technical, business and social aspects of software ecosystems. The results of this initial set of interviews have been published previously in [24].

Survey

To address the limitations of our initial set of interviews we conducted a survey with a larger sample of software developers from around the world. We selected mobile developers using a parser that extracted information from GitHub [1]. More precisely, we analyzed the documentation of various mobile platforms and projects from GitHub and observed that projects for the Android, Windows Phone and FirefoxOS platforms all contained files with the same unique name, while in iOS projects the word LSRequiresIPhoneOS occurs in some files. Based on this information, we wrote a parser that accessed GitHub through its API and identified projects with these indicators. This way, we were able to identify projects that included mobile software development. We then fetched information about the contributors of these projects, randomly selected 400 developers out of this larger pool and invited them to take our survey.

We designed the survey with 15 questions divided into three sections. The first section asked questions about the work experience of the respondent in developing mobile applications. The second section collected information about the most important application developed by the respondent, whether the application was available in the app store and what tools were used to develop it. Finally, the third section asked about developers' perceptions regarding mobile application development including what were the most positive and negative aspects they had to deal with during the development of mobile apps and the hardest problem they faced. All questions of this last section were open-ended.

We received 83 answers for our survey (response rate 20.75%). The most frequent locations from our survey respondents were: 1st - USA (24); 2nd - Canada (4); 3rd - Brazil, India, the Netherlands and the United Kingdom (3); 4th - Belgium, Cambodia, China, France, Spain and Taiwan (2). Other countries indicated by developers included: Bolivia, Denmark, Finland, Georgia, Indonesia, Israel, Japan, Mexico, Sweden and Switzerland. The location information is not required by GitHub, which explains the amount of missing information about the location of some informants (19). The three most cited platforms were Android (50), Web (40), and iOS (39), but informants could select multiple platforms. Developers' experience ranged from less than one year to more than five years. Survey respondents had published between one and more than six apps in mobile app stores.

Similarly to our previous dataset, we analyzed the open ended questions using coding techniques from Grounded Theory [7]. Furthermore, we integrated the data from both datasets (interviews and survey) adopting a unique set of codes. We present our results in an integrated way, although there were some differences between the survey results and the initial interviews. For instance, survey answers mentioned the possibility of receiving feedback from users through comments in the "app store" (E9's quote later in the paper).

Additional interviews with Brazilian developers

We then designed a new interview guide that aimed to gather information about the *social* aspects of software ecosystems in more details. For instance, questions included the influence of friends and co-workers in joining and participating in an ecosystem, the perceived benefits or drawbacks of being a member of an online community about a mobile platform, how important it is to be part of that community, how that community can help beginners to start developing for the plat-

[1] http://github.com

form, and whether there is interaction with other community members, for instance by blogging, answering questions on message boards etc.

Using this new interview guide, we conducted nine additional semi-structured interviews with mobile developers (8 men and 1 woman). Our interviewees were between 20 and 28 years old with experience between seven months and six years, and they were all from Brazil. The interviews were again based on a convenience sample through our personal contacts. Interviews lasted between 14 and 28 minutes and were recorded. The shortest interview was conducted with an undergraduate student with limited experience.

Additional interviews with international developers

One of the questions of our survey asked whether the informant could be contacted for an interview and asked for the informant's email address. We received 25 positive answers from the 83 survey respondents. We then sent an email invitation to these 25 informants asking whether they would still be willing to be interviewed. We interviewed 7 informants using the same interview guide from our previous data collection phase. Interviewees were from different countries including United States, Brazil and Venezuela and had different levels of experience developing for mobile ranging from one to four years. Interviews ranged from 14 to 45 minutes and were again recorded.

Data Analysis

In our data analysis we transcribed all interviews and integrated all four datasets into one unique dataset. After that, the data collected was jointly analyzed by using coding techniques from Grounded Theory [7].

In our first step—open coding—-our data was micro-analyzed (line-by-line) to identify categories. We identified two major categories related to social aspects of mobile development: adoption and permanence, i.e., the social aspects that influence a software developer to adopt a particular mobile platform, and the social aspects that influence a developer to keep developing for that platform. As discussed in the previous section, adoption and permanence are essential to ecosystem sustainability.

In the next step—axial coding—categories were broken into subcategories and we identified properties and dimensions of those categories. Initially, different authors coded different datasets all focusing on the social aspects of ecosystem development. As one would expect this resulted in similar ideas being coded differently. Therefore, we held a 3-day workshop with the authors in which the integrated dataset was analyzed again and coded using one unique coding scheme. Our coding scheme created 25 different categories including "adoption as influenced by the social network", "the relationship between a developer and the keystone company", etc., as well as subcategories such as the community aspect and its size, scope (local vs. global) and communication channels used.

Validation interview

In the last step of our research, we presented our results to a business platform development leader from a major software company. This was done through the presentation of a executive summary over email. According to this leader:

> *The "social aspect" as you call it plays a big role IMO, both for getting "infected" with an ecosystem preference and for staying with it. From my perspective, this is part of daily business for platform biz dev [business development] teams, either on the marketing side for tackling individual developers or on the key account side for tackling large ISVs [independent service providers]. Also on-site events are important and virtually every platform [keystone] company has them.*

In other words, this platform leader validated our results suggesting that social aspects are indeed very important for the sustainability of a software ecosystem.

FINDINGS

In this section we answer our research question—*how do social aspects influence the sustainability of proprietary ecosystems?* We found that the sustainability of a software ecosystem depends on two different processes, i.e. (i) *adoption*, which is concerned with the process of joining a particular ecosystem and (ii) *permanence*, in which developers decide to keep developing for that ecosystem. To illustrate the different aspects of each theme, we provide a selection of quotes from the initial interviews and survey (indicated by A#), as well as from the additional interviews (B#).

Social influences on ecosystem adoption

In our study, we asked developers how they were introduced to mobile development and whether someone influenced their decision to *adopt* a particular platform ecosystem. We found that developers are often influenced by others, including friends, undergraduate colleagues and co-workers:

> *"Actually the idea came out of nowhere but had the influence of a friend of ours, who also worked with development for people with disability. And then we decided: let's make an app for that." [A2]*

> *"Yes, I was influenced by a friend, which is even more active in the Android community. After I bought my Android device, I looked at my phone and thought about creating an app. [...] But I was influenced by other friends who also are developers." [B3]*

> *"Some friends of mine already coded for Android and gave me tips .. it's cool, it's interesting, the ability to be able to do several things, and for being a low-cost device influenced a lot." [A12]*

Having friends or co-workers who are actively developing for a particular platform brings many advantages to the developer, including being exposed to new experiences and acquiring new skills. In addition to technical skills, one can also learn about business aspects through these interactions:

> *"I saw with the experience of X and Y [A3's friends] they released a very similar app, an app about Naruto [a game character] and you could see that the App Store was selling ten, twenty times more than the Android Market." [A3]*

A similar advantage is the increasing perception a developer acquires about the job market, especially the local one. Thus, (s)he is able to identify new opportunities:

> *"I started [learning about this platform] after joining my current job... my co-workers needed an iOS developer, and I became interested in learning." [B7]*

Several of our informants reported the logic behind their decisions about the adoption of mobile platforms: they were ambitious to reach a larger number of users[2].

The next quote presents another result from our dataset. It illustrates how the broader social context in which a developer is embedded influences his/her decision to adopt a platform:

> *"iOS was more complicated due to its [programming] language. I did some things but did not really understand how it worked. Android was simpler because it is Java, which I studied in my undergraduate course." [A4]*

In this case, the context is related to the university the informant attended and, consequently, the prior knowledge (s)he possessed before deciding to adopt an ecosystem. This can be explained by Rogers' diffusion of innovations theory, which defines compatibility as the degree to which an innovation is perceived as consistent with the existing values, past experience, and the needs of the potential adopters [28]. Thus, participants who learned Java previously had greater compatibility with the Android platform, which made it easier for them to adopt that platform.

Our results suggest that social aspects are important factors that contribute to attract developers to join a particular platform ecosystem. In other words, developers are influenced by other developers, but also by their own previous experiences, and their expectations and perceptions of the local market.

Finally, it is important to mention that increasing the adoption of a software ecosystem for developers is an important step in building a successful software ecosystem. However, the social aspects that influence the permanence of developers within a software ecosystem after its initial *adoption* are also important. This aspect, *permanence*, will be discussed in the next section.

Social influences on ecosystem permanence

We have previously defined software ecosystem permanence as *"the continued participation of developers in a software ecosystem"*. So, throughout our additional interviews we specifically asked our informants how social aspects influenced them in remaining in a specific platform ecosystem.

The role of developer communities

Permanence in an ecosystem is influenced by both local developers and online communities. Both help nourish the software ecosystem by answering questions, providing updated information, feedback, motivation, etc. According to one of our informants:

> *"I think it's really important to join an [online] group, get your questions answered and have people you can count on. The response time is fast, sometimes you post and receive an answer in an hour!" [A7]*

The quote above also illustrates the efficiency of these communities, where after posing a question, one might get answers in a very short time-frame.

Despite the different advantages of online communities, some informants prefer to participate in local communities or help colleagues through face-to-face groups. This does not mean they do not like or do not participate in online communities, but only that they treat face-to-face relationships more carefully. The following is a quote in which the informant said his contribution to the community is more related to his personal time in face-to-face events.

> *"My contribution is higher for my company. To the outside community I do not have greater collaboration, only a few tutorials. In general I am more contributory to my colleagues at work... I also like teaching, doing DOJOs,... [I prefer] something live more than participating in an online community. " [A9]*

Participation in online communities is quite varied, i.e., a developer has a variety of channels that (s)he can use to get the information (s)he needs. For instance, informant E1 reported that he prefers to use Facebook groups to address his questions. On the other hand, informant E16 participates in events to stay updated regarding new features and versions of the platform ecosystem. In this case, the event E16 was referring to is Google I/O, an annual conference held by Google focusing on software development. Differently, E9 refers to YouTube channels as an important source of information for him. These different channels provide useful information about programming tips, update informations, among other topics. In short, one can observe a variety of means used to keep in touch with the online community of developers: social networks like Facebook and Google Plus, events, YouTube channels, etc. In addition, Q&A sites are also important:

> *"Of course I do several queries ... many of them I get the answer from Stack Overflow, sometimes the GUJ [a Java user group in Brazil] which also has Android questions [...] Company's[3] channel also has people who answer questions there or Google Plus which is a more technical network to find new things that are emerging." [A9]*

The rapid evolution of techniques and development of technologies made teaching materials (e.g. books) become obsolete at a much faster rate and, as a consequence, software development increasingly challenging [33]. By participating in online communities, developers are able to keep up with this fast-paced environment. In short, this is an important advantage of these communities. This can be confirmed by the quote below:

> *"It's hard to find solutions to everyday problems. In general, the community is the main source [of information].*

[2]The Android operating system was present in 89% of all smartphones in Brazil until the end of the second quarter of 2014, according to a study by Kantar Worldpanel ComTech.

[3]Famous company in Brazil that provides training in software development.

The moment you buy a book, chances are it is already out of date." [B1]

While online communities provide advantages like efficiency and access to updated information, our respondents also reported at least one limitation of these online communities. E14 mentioned that the support provided by these communities at times is *not* ideal, since most of the information available in these communities is more suitable for advanced users, falling short in supporting novice users. This quote also suggests that information on those sites should be organized differently to address this aspect.

"I rank as good the help [one finds in an online community] but not outstanding. As it [the information one finds in these communities] should be organized at all levels. I see a lot of benefits for those who are advanced but not for who's starting." [A14]

One important aspect that we have identified in our findings is the role of what we call *internal developers*, i.e., developers who are employees of keystone companies and are involved in developing SDKs, APIs, libraries, emulators and other tools for software development [16]. As developers working for keystone companies, they have access to important information about the ecosystem. To be more specific, in our study, we found that respondents value the participation and engagement of these internal developers on social networking sites and other social media. This participation provides higher levels of social transparency [34] about the identity of internal developers therefore allowing external developers to interact with them. To illustrate this point, our data shows that Android developers (who work for Google) use specific communication channels to disseminate information about the documentation and official releases [35], so by taking advantage of their level of social transparency in social media they can promote the Android ecosystem to "external" developers.

"The Android community is in Google Groups or Google Plus. [...] There is even the participation of members of the Google team, what motivates us a lot." [B3]

By displaying their identities and activities on social networks, internal developers are also showing their interest and commitment to the community of software developers. This attitude towards the community increases developers' confidence in the platform and motivates them to continue participating in the ecosystem.

In our study, no informant reported how Apple (developers) communicates with external developers to release official information, to inform where to find documentation, or any other aspect. This could be a limitation of our study, because only a minority of informants developed for the iOS ecosystem.

In summary, developers' social relationships (face to face or online) are important for the persistence of a developer in a given software platform. Online communities can be found in several ways—online groups, social networking sites, events, etc. In addition, *keystone* developers help even more in this process by serving as a source of motivation for developers,

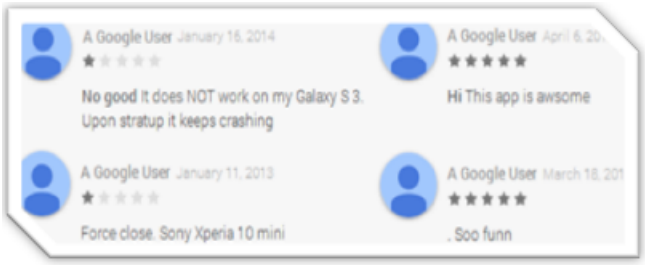

Figure 1. Different user experiences for the same application based on the Android operating system.

as well as answering specific questions about the details of the platform ecosystem.

The role of end users

In most of the major ecosystems, the distribution of apps is usually done through app stores such as Play Store, App Store and Windows Phone Store. These stores, however, are not only used as channels for the distribution of mobile applications, *but they also serve as a communication channel between end users and application developers*. In other words, app users provide feedback to developers by suggesting new features to be implemented and/or by reporting bugs. Figure 1 illustrates different feedback for the same app: while some users complain that the application does not work perfectly on their smartphones, others comment that the application is "fun".

The quote below illustrates how our informants reported this communication between developers and end users. It also illustrates how app stores provide useful information about app usage, which therefore influences the continuous improvement of apps. On one hand, getting negative feedback from end users is something important to the improvement of the app. On the other hand, it is also harmful, because the more negative reviews it has, the lower its rating. And the lower its rating, the less likely it is to be bought.

"I can see the number of users who are using my app, how many people are uninstalling, how many people have downloaded the same and which countries [they are from]. If a user rates my application and reports an error, I can answer it. This allows direct interaction with the user." [A9]

In addition to using app stores as the primary means of communication with end-users, developers reported using their personal relationships to distribute their applications for testing. This flexibility in the distribution process was mentioned several times in our interviews as a factor influencing the continued participation of developers in the ecosystem. For example, in the Android ecosystem, there is a way to distribute an app for a user group selected by the developer. Given the great importance of having positive feedback before making their applications available in an app store for all end users, many developers create test groups with a small number of selected users. Furthermore, one can select user groups to test different versions of the app and release updates as tests are concluded. After this testing, the app can be made available for all users who wish to download it. This is a potential way

to improve the quality of the app and reduce the amount of negative scores. The quote below illustrates these aspects:

> *"The best way is to link an application with a group. So you can bind them as testers. They receive e-mail and they install the application. They accept that they want to test and install the application." [B3]*

A different way to distribute applications, and thus receive feedback from users, is through the direct distribution of the apps directly to end users using an installation file. Again, this is done before publishing the final version of the app in the app store.

> *"We tested the application by giving [it] to some friends and family, then we waited for feedback. We distributed [the app] using e-mail and pen drives." [A1]*

In short, promoting the relationship between developers and end users is an essential aspect of a software ecosystem because it allows developers to improve the quality of applications based on the feedback from their end users. This can be done by making the app available on the app store (i) to end users (in this case, communication between developers and end users occurs through comments) (ii) to test groups, and finally (iii) to end users (personal contacts and / or user groups) using installation files. In the latter cases, communication takes place directly between the developer and the end users.

DISCUSSION

In our results, we observed that social aspects influence the adoption and the permanence of software developers in a particular software ecosystem. Adoption is influenced by a broader context in which developers are embedded which includes personal friends and co-workers and, at the same time, previous experiences by the developers. Permanence is influenced in many ways. Friends and co-workers who are in close physical proximity play an important role in answering questions, providing feedback, etc. Meanwhile, online communities provide support for distributed developers through different channels including social networking sites, groups, Q&A sites, and so on. Finally, informants reported that the participation and engagement of keystone developers on social networking sites and other social media is an important source of motivation and information for them. In this section, we discuss our findings in the context of previous literature and conclude it by providing implications from our study for companies and developers interested in the sustainability of software ecosystems.

Social Aspects of Proprietary vs. FOSS Ecosystems

As mentioned in Section 2, FOSS ecosystems allow external developers to change their software platform, since they are more open than proprietary ecosystems [2]. This means that developers dealing with FOSS ecosystems need to interact to coordinate their changes in the software platform since their code is interdependent. In fact, several studies focus on the social aspects of software developers in this context pointing out to the importance of the social aspects for the sustainability of these ecosystems. Meanwhile, proprietary ecosystems are closed [2], i.e., developers can not change the software platform, but only use its services through APIs. Therefore, since their code is independent, developers rarely need to interact among themselves.

The lack of interaction among developers in proprietary ecosystems could lead one to believe that the social aspects of these ecosystems are not important. However, our results suggest the opposite, i.e., they illustrate how the interaction among different social actors (friends, family, developers working for the software platform company etc.) allows a developer to make a decision about *joining* and *remaining* in a particular software ecosystem, therefore creating a sustainable ecosystem. Furthermore, our results also show *how* this interaction among social actors takes place: in some cases, face to face in informal conversations or events, while in others it is mediated by different tools including social networking sites, Q&A sites, forums, and others. Finally, our results illustrate the important role of developers who are employees of keystone companies: by displaying their identities and activities online, these developers are also displaying their commitment to a particular ecosystem, which motivates developers to continue participating in this ecosystem. Permanence in an ecosystem is then influenced by the degree to which keystone companies allow their co-workers to be "transparent" [34] about their work and engage with external community members.

In the context of previous work, this paper highlights the role of *keystone* developers, an aspect overlooked in the FOSS ecosystem research. Our findings also show that developers are attracted to software ecosystems by their colleagues, but in different ways of those reported by Hahn et al. [14]. According to Hahn et al., social factors influence developers to adopt and remain developing for a particular *project* in a FOSS ecosystem, while in proprietary ecosystems the influence is to adopt and remain in the same *platform ecosystem*, not necessarily in the same project. This means that community support, interactions among developers and other social factors are means of socialization for building potentially distinct applications (projects) for that platform, i.e., developers work *independently*. In short, in a FOSS ecosystem, interactions are needed to coordinate artifact changes in a common project and, consequently, developers' work is *interdependent*. It is possible that social factors in FOSS projects also influence independent work, but to the best of our knowledge, this has not been reported in the literature.

In general, the focus on the *platform*, instead of *projects* is a major difference from our work compared to previous work on FOSS ecosystems. For instance, Steinmacher et al. [32] presented 13 social barriers that make difficult to newcomers to join FOSS *projects*, and consequently, their sustainability. Exceptions to this are the work by Draxler and colleagues [11] and Jergensen et al. [18]. Draxler looked at the collaboration among software developers who participate in the same software ecosystem, while Jergensen et al. [18] studied progressive developers' participation in the FOSS ecosystem GNOME. In both cases, the focus was on FOSS ecosystems, not on proprietary ones. Despite the differences between our findings and previous work on FOSS, further research is

needed to understand social factors' impact on the sustainability of software ecosystems when "hybrid" development models are adopted (e.g. Eclipse is a FOSS project with IBM developers working on it), in which there might be both independent and interdependent work involved.

Our results also illustrate how these social aspects directly influence the sustainability of a proprietary software ecosystem. As pointed out earlier, a sustainable software ecosystem is "one that can increase or maintain its user/developer community over longer periods of time and can survive inherent changes such as new technologies or new products" [10]. For instance, we discussed how online communities and keystone developers using social media helped software developers to stay current regarding both the ecosystem in which they participate and the technology landscape in general. Staying current is important because it helps developers to prepare for upcoming changes in a fast-paced environment.

Software ecosystems as online communities

Our results suggest that the permanence in a software ecosystem is largely influenced by local and remote (online) communities of developers. Therefore, it is important to contrast our results with previous work in online communities.

In 1995, Sproull and Faraj introduced a view of the Internet as "a social technology that allows people with common interests to find each other, gather, and sustain connections over time" [31]. Following this interpretation, online communities—defined as "people with shared interests or goals for whom electronic communication is a primary form of interaction" [9]—have developed for different topics. In these communities, information exchange is the main goal [26].

The centerpiece of most online communities is a single website to which members are supposed to contribute. For example, the online community powering Wikipedia has the clear goal of improving and adding to the content of the online encyclopedia. Social aspects are often realized in such online communities as a side-effect. For example, Forte et al. [12] found that WikiProjects—groups of contributors who want to work together as a team to improve Wikipedia—not only help produce articles, but also provide a place to find collaborators and facilitate socializing and networking. Since these online communities need users to create content for their community website, their success is determined through sociability and usability [25]. In a study on the collaborative writing community Everything2, Lampe et al. [21] found that social and cognitive factors seemed to be more important in predicting contributions to the site than usability issues. Similarly, Ridings and Gefen [26] found social aspects to be among the reasons why individuals join online communities.

While software ecosystems are similar to online communities, their main goal is not the creation of content for the community website. Instead, their goal is multi-faceted, i.e. developers' goal is the development of apps, keystone companies' goals are more business-oriented including market share, profits, etc. As became evident in our interviews, software developers use multiple means of communication and knowledge exchange when they participate in an ecosystem, ranging from the Q&A website Stack Overflow and other social media channels to face to face interaction with friends and colleagues. In addition, they use various tools and development environments to design, develop, and release their applications. They value the social transparency and the information up-to-dateness their communities provide and use these communities as an essential tool for their daily work as professional developers.

Software ecosystems as a developer choice

It is important to keep in mind that software developers constantly have to make decisions about the tools to be used including what libraries or tools (e.g., IDE) to use and what platforms to develop for. Depending on the decision to be made, previous research was inconclusive or illustrated the importance of social aspects. For instance, when choosing an API, Robillard [27] suggested that social aspects do *not* play a crucial role. In his research on obstacles encountered by developers trying to learn an API, Robillard identified problems related to inadequate or absent resources, the structure or design of the API, as well as other technical and process issues. In addition, he noted obstacles caused by the respondents background and prior experience—a category that also emerged in our study. Other researchers have theorized that social aspects are important: Chen et al. [6] identified eleven themes that play a role when developers choose an API, such as learnability and interoperability. Among others, they identified an active community as an important factor when choosing APIs, especially when official documentation was lacking. In those cases, the community could answer specific questions and provide up-to-date information on an API—a result which is mirrored by our findings in the context of a software ecosystem instead of APIs.

A different type of decision to be made is whether to become an active contributor of an open source project or to adopt a mobile ecosystem. When making these decisions, developers consider a mix of social, technical, and business factors. Hertel et al. [15] compared joining an open source project to participating in a social movement, such as the civil rights movement or the labor movement. In their survey on motivation of software developers in open source projects, they found that developers' engagement was particularly determined by their identification as a developer for a particular project as well as by pragmatic motives to improve their own software tools and career chances. Our study revealed similar findings: developers' preferences towards ecosystems are influenced by their personal identification with the governance strategies of the keystone companies.

To summarize, while previous research was somewhat inconclusive (especially in the context of APIs), our paper provides additional evidence of the importance of social aspects in the decision-making process of software developers.

Organizational Implications

In order to attract developers to their ecosystem and to sustain them as active contributors of apps and other content, keystone companies need to be aware of the importance of the social aspects of the software ecosystems. Therefore, we argue that these implications follow from our findings:

- When choosing a software ecosystem, local market share (e.g., among friends and family), can be just as important as overall market share. In addition, developers are biased towards ecosystems for which they are somehow already familiar. Therefore, it is important for keystone companies to not only promote the ecosystem as a whole, but also the programming languages and tools that are part of it.
- We found that developers value the active participation and engagement of keystone developers in social network sites and social media. This social transparency can be achieved by explicitly recognizing the effort from keystone developers who participate in these sites.
- Social network sites and social media resources are also an important source for developers to learn about new releases or development tools and techniques. For companies trying to get information out to their developers, it is important to make use of the channels available. It is also important that keystone companies are careful with the different degrees of expertise, therefore providing information for both beginners and advanced developers.
- Finally, encouragement for live events is also important since some developers reported that they prefer this form of participation instead of online events. As they reported, they feel more active in face to face events. Thereby, the existence of these events gives these developers opportunities to contribute to the ecosystem.

CONCLUSION

The simple model in which a single company develops a product and then sells these products to its clients has been slowly replaced by a complex ecosystem composed by different companies, internal and external developers, online developer communities, and end users. These software ecosystems can be studied from a technical, business, or social point of view. While the technical and business aspects of such ecosystems have been investigated in past research, the social side of software ecosystems is less well understood, especially in the context of *proprietary* ecosystems (i.e., not free and open source).

We reported on a qualitative empirical study based on 25 interviews with mobile software developers and an online survey with 83 respondents from the mobile development community to understand the social aspects of proprietary ecosystems and their impact on the sustainability of such ecosystems. Our data analysis reveals a complex social context that includes the socialization of developers (and the associated learning), a form of "social capital" (knowing who to ask for help [11]) that is enabled by the social transparency of keystone developers. These social aspects in general do not only play an important role when software developers initially choose to adopt a particular software ecosystem, but are also crucial in sustaining a developer's ties with that ecosystem.

In future work, we will continue to investigate the social aspects of software ecosystems from the perspective of the different actors involved. In particular, we plan to study which communication and coordination channels are the most effective for different tasks in the ecosystem, and we plan to investigate the consequences of different strategies of keystone companies in an ecosystem in more detail. In addition, we believe an important aspect to be studied is how social aspects influence "hybrid" ecosystems, i.e., ecosystems which are both FOSS and proprietary.

ACKNOWLEDGMENTS

The first author would like to thank the funding from CNPq (process numbers 440880/2013-0 and 310468/2014-0).

REFERENCES

1. Ron Adner. 2012. *The Wide Lens: A New Strategy for Innovation*. Portfolio Hardcover.
2. Mohsen Anvaari and Slinger Jansen. 2010. Evaluating Architectural Openness in Mobile Software Platforms. In *Proc. of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10)*. ACM, New York, NY, USA, 85–92.
3. Olavo Barbosa, Rodrigo Santos, Carina Alves, Claudia Werner, and Slinger Jansen. 2013. *Software Ecosystems: Analyzing and Managing Business Networks in Software Industry*. Edward Elgar, Chapter A Systematic Mapping Study on Software Ecosystems through a Three-dimensional Perspective.
4. Jan Bosch and Petra Bosch-Sijtsema. 2010. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software* 83, 1 (2010), 67–76.
5. Kathy Charmaz. 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. SAGE Publications.
6. Xiang 'Anthony' Chen, Matthew Dunlap, Richard Fung, and Túlio Souza. 2011. How Do Developers Choose APIs? (2011). University of Calgary, Canada.
7. Juliet Corbin and Anselm Strauss. 2008. *Basics of qualitative research: Techniques and procedures for developing grounded theory* (3rd ed.). Sage Publications.
8. Simone da Silva Amorim, John D. McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez. 2014. Flexibility in Ecosystem Architectures. In *Proc. of the European Conf. on Software Architecture Workshops*. Article 14, 6 pages.
9. Alan R. Dennis, Sridar K. Pootheri, and Vijaya L. Natarajan. 1998. Lessons from the Early Adopters of Web Groupware. *J. Manage. Inf. Syst.* 14, 4 (1998), 65–86.
10. Deepak Dhungana, Iris Groher, Elisabeth Schludermann, and Stefan Biffl. 2010. Software Ecosystems vs. Natural Ecosystems: Learning from the Ingenious Mind of Nature. In *Proc. of the 4th European Conf. on Software Architecture: Companion Volume*. 96–102.
11. Sebastian Draxler and Gunnar Stevens. 2011. Supporting the Collaborative Appropriation of an Open Software Ecosystem. *Computer Supported Cooperative Work* 20, 4-5 (2011), 403–448.

12. Andrea Forte, Niki Kittur, Vanessa Larco, Haiyi Zhu, Amy Bruckman, and Robert E. Kraut. 2012. Coordination and Beyond: Social Functions of Groups in Open Content Production. In *Proc. of the Conf. on Computer Supported Cooperative Work*. 417–426.

13. Samuel Fricker. 2009. Specification and Analysis of Requirements Negotiation Strategy in Software Ecosystems. In *Proc. of the First Intl. Workshop on Software Ecosystems*. 19–33.

14. Jungpil Hahn, Jae Y. Moon, and Chen Zhang. 2008. Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties. *Information Systems Research* 19, 3 (2008), 369–391.

15. Guido Hertel, Sven Niedner, and Stefanie Herrmann. 2003. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* 32, 7 (2003), 1159 – 1177.

16. Marco Iansiti and Roy Levien. 2004. *The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability*. Harvard Business Press.

17. Chris Jensen and Walt Scacchi. 2007. Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study. In *Proc. of the 29th Intl. Conf. on Software Engineering*. 364–374.

18. Corey Jergensen, Anita Sarma, and Patrick Wagstrom. 2011. The Onion Patch: Migration in Open Source Ecosystems. In *Proc. of the 19th SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering*. 70–80.

19. Kimmo Karhu, Tingan Tang, and Matti Hämäläinen. 2014. Analyzing Competitive and Collaborative Differences Among Mobile Ecosystems Using Abstracted Strategy Networks. *Telemat. Inf.* 31, 2 (2014), 319–333.

20. Stefan Koch and Markus Kerschbaum. 2014. Joining a smartphone ecosystem: Application developers' motivations and decision criteria. *Information and Software Technology* 56, 11 (2014), 1423–1435.

21. Cliff Lampe, Rick Wash, Alcides Velasquez, and Elif Ozkaya. 2010. Motivations to Participate in Online Communities. In *Proc. of the Conf. on Human Factors in Computing Systems*. 1927–1936.

22. Konstantinos Manikas and Klaus Marius Hansen. 2013. Software ecosystems – A systematic literature review. *Journal of Systems and Software* 86, 5 (2013), 1294–1306.

23. Tom Mens, Maëlick Claes, and Philippe Grosjean. 2014. ECOS: Ecological studies of open source software ecosystems. In *Proc. of the Conf. on Software Maintenance, Reengineering and Reverse Engineering*. 403–406.

24. Müller Miranda, Renato Ferreira, Cleidson R. B. de Souza, Fernando Figueira Filho, and Leif Singer. 2014. An Exploratory Study of the Adoption of Mobile Development Platforms by Software Engineers. In *Proc. of the 1st ACM Intl. Conf. on Mobile Software Engineering and Systems*. 50–53.

25. Jenny Preece. 2001. Sociability and usability in online communities: determining and measuring success. *Behaviour & IT* 20, 5 (2001), 347–356.

26. Catherine M. Ridings and David Gefen. 2004. Virtual Community Attraction: Why People Hang Out Online. *J. Computer-Mediated Communication* 10, 1 (2004).

27. Martin P. Robillard. 2009. What Makes APIs Hard to Learn? Answers from Developers. *IEEE Softw.* 26, 6 (2009), 27–34.

28. Everett M. Rogers. 2003. *Diffusion of innovations* (5th ed.). Free Press. 576 pages.

29. Walt Scacchi. 2007. Free/Open Source Software Development: Recent Research Results and Methods. In *Architectural Issues*, Marvin V. Zelkowitz (Ed.). Advances in Computers, Vol. 69. Academic Press Inc., 243–295.

30. Sonali K. Shah. 2006. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Manage. Sci.* 52, 7 (2006), 1000–1014.

31. Lee Sproull and Samer Faraj. 1995. Atheism, Sex, and Databases: The Net As a Social Technology. In *Public Access to the Internet*, Brian Kahin and James Keller (Eds.). MIT Press, 62–81.

32. Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proc. of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 1379–1392.

33. Margaret-Anne Storey, Leif Singer, Brendan Cleary, Fernando Figueira Filho, and Alexey Zagalsky. 2014. The (R) Evolution of Social Media in Software Engineering. In *Proceedings of the on Future of Software Engineering (FOSE 2014)*. ACM, New York, NY, USA, 100–116.

34. Colleen Stuart, Laura Dabbish, Sara Kiesler, Peter Kinnaird, and Ruogu Kang. 2012. Social Transparency in Networked Information Exchange: A Theoretical Framework. In *Proc. of the Conf. on Computer Supported Cooperative Work*. 451–460.

35. Christoph Treude and Margaret-Anne Storey. 2011. Effective Communication of Software Development Knowledge Through Community Portals. In *Proc. of the 19th SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering*. 91–101.

36. Eric S. K. Yu and Stephanie Deng. 2011. Understanding Software Ecosystems: A Strategic Modeling Approach. In *Proc. of the 3rd Intl. Workshop on Software Ecosystems*. 65–76.