Start-up acquisitions in the open source software space : What is the effect on community dynamics?

David Roegiers

23rd of May 2016

Master Thesis

Tutored by prof. Georg von Krogh

Supervised by Yash Raj Shrestha

Chair of Strategic Management and Innovation

Department of Management, Technology, and Economics

ETH Zurich


venture-funded
public
dyad
data
archival
commercial-OSS
IPO
effect
turnover
sponsorship
soft-services
motivation
firm
leadership
contributions
acquiree
master
technical
entry-exit
GitHub
relationship
dual-licensing
capital
users
quantitative
permissive
acquisition
pre-acquisition
PyGithub
ANOVA
loss-leader
hard-services
venture
OSS
exit
participant
founder
research
restrictive
repositories
open-innovation
commit
extrinsic-motivation
Python
community
acq-hire
dynamics
startup
comments
developer
entrepreneur
dyad
practice
code-releasing
symbiotic
licensing
issues
press
CrunchBase
research-gap
business
academic
productivity
post-acquisition
acquirer
empirical
attractiveness
intrinsic-motivation
snowball
pull-request

Abstract

We tell the story of entrepreneurial programmers that launch an open source software community and simultaneously found a startup to serve as the commercial face of the project. With support from venture capitalists the software is scaled sufficiently, after which the business is subject to acquisition by established OSS players. But how does the community react to such a change?

In this master thesis we study dyads of venture-funded OSS startups and their communities. We investigate what effect a firm acquisition has on the community in terms of entry-exit dynamics and contribution rates. After deductive hypothesis formulation from literature, we perform quantitative tests by gathering empirical `GitHub` data from a `CrunchBase` sample of 16 cases. Using 1,363,933 contributions from 25,518 contributors over a period of 11 years, we found in contrary to expectations that technical contributions decrease after the acquisition event. Non-technical contributions on the other hand experience mixed effects. Post-acquisition exit rates of developers and users & visitors increase, while a similar effect for entry rates is only observed for users & visitors. These results imply that community attractiveness decreases upon acquisition. We provide potential explanations and propositions for future research. To support that, we release a `Python` module specially written for the purpose of collecting data from `GitHub` organisations, using efficient multi-threading and data enriching heuristics. Implications of our findings on practice and academics are discussed.

Acknowledgements

My gratitude goes out to Yash Raj Shrestha for his distinguished supervision. He has been a constant input of good ideas, all the while offering the freedom to make up my own mind.

This work would have not been possible without professor von Krogh and his chair, which offered a backbone in innovation literature and knowledge on which this thesis was built. Additionally, I wish to recognise Vincent Jacques and the PyGitHub community for the great GitHub API library that they provide.

Thanks to Omid, Aldo, and Yash for the refreshing breaks, and Natassja for the brain-feeding chocolate. Above all, I extend profound gratitude to Manja for the mental support.

Contents

Chapter 1

Introduction

There is a theory which states that if ever anyone discovers exactly what the universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened...
- Douglas Adams

For the past two decades, open source software has been a hot research topic in various academic fields (von Krogh and von Hippel, 2006). Software systems such as `Linux` and `Apache`, triggered widespread attention to the novel innovation practice. Initially, opinions were scattered about whether these projects could catalyse substantial market value. But the emergence of commercial firms that successfully capitalise on OSS with a tremendous market share, quickly created a positive consensus. These players (e.g. `Red Hat`) gave birth to several business models that combine OSS development with market revenues in a sustainable way (Hecker, 1999).

This momentum led to the initiation of many new projects coming from developers, universities, and companies, in an attempt to find popularity, supporting resources, and constructive ideas for their software code. However, it turned out to be quite challenging to keep projects and their corresponding communities alive (Santos et al., 2012). Developers that freely contribute to open source software need motivations to do so. These can vary from career and pay to altruism and fun (von Krogh et al., 2012). The presence or absence of these factors can determine the fate of a community.

Lately, there have been many OSS projects founded and sponsored by commercial firms. The firms are essential components, as they manage the infrastructure, pay developers, organise conference events, and provide community leadership. There are cases of "co-opetition" where multiple firms engage in a single community, such as `Open Stack`. More often though,

a one-to-one relationship takes hold where the community and OSS firm (e.g. `MySQL`) live "symbiotically" (Dahlander and Magnusson, 2005).

In the past decade there have been many venture-funded startups appearing that demonstrate such a relationship. The goal of these ventures is to find an entrepreneurial exit. Although some initial public offerings (IPO) of pure OSS companies have taken place (`Red Hat`, `HortonWorks`, ...), the acquisition channel seems to be the favoured option so far. On the acquirer's side, high-tech startup acquisitions often seem to be a human resource strategy for innovation, termed "acqhire" (Selby and Mayer, 2013). The interest lies in the highly-skilled personnel and tacit knowledge, which presents a strong buy-in into OSS communities (Lerner and Tirole, 2005a).

Many of the observed cases tell the story of entrepreneurial software developers that launch an OSS community and simultaneously found a startup to serve as the commercial face of the project. With support from venture capitalists the software is scaled sufficiently, which in turn makes their business ideal subjects for acquisition by established OSS players. But how does the community react to such a change? Will it maintain its size or growth with the new profit-seeking governance?

Now, it is known that the firm attributes perceived by the community will have an effect on the motivation of contributors (Spaeth et al., 2015). Misalignment in the social practice or person-organisation fit can deter the relationship between the community and the sponsor, which influences the developer turnover and overall success of the OSS project (Sharma et al., 2010; von Krogh et al., 2012).

This thesis seeks to identify community effects, when such acquisitions take place. We base ourselves on "dyads" of OSS communities and their respective startup sponsor, where typically one can't live without the other. When the OSS startup is acquired changes may occur in the development team and decision-making, the commercial entity is replaced by a new image, and the open source software can become embedded as part of a larger portfolio. We expect this to have an influence on the dynamics of the community in terms of participant's arrival, departure, and contribution rate, and profile attributes.

A deductive study is presented, where our hypotheses are investigated using quantitative archival data from `GitHub`. Repository activity is registered as (non)-technical contributions, made by either developers, or users & visitors. Accordingly, our research adds to the many academic papers that further discover open source software by focusing on `GitHub` since the beginning of this decade. In addition, we release a `Python` module as a tool to gather activity of entire `GitHub` projects.

For the past 5 to 10 years, market consolidation in the OSS space has actively been reported. Given the large amount of new and upcoming ventures, we do not expect this trend to stagnate.[1] In essence, this thesis beholds a right timing in the practitioner's point of view: our findings consider sufficient cases, without being too late to inform decision-makers that take part in a later phase of the acquisition trend. Practical implications are bounded by the open source software discipline. Nevertheless, the highly-specific topic of this work allows managers in the field to directly apply take-aways, all the while providing insights to generalise findings for the IPO exit strategy.

Our thesis contributes to the very long list of literature around open source software. More in particular, it lays at the intersection of the research streams of community motivation and commercial OSS (a.k.a. OSS 2.0), by building on research papers that look at the relationship between the community and organisational sponsor. Hauling in the academic fields of *venture-funded startups* and *acquisitions*, we fill a clear research gap. Thus, this work represents an initial academic move to a growing industry phenomenon: OSS startup exits. Moreover, our results potentially feed the knowledge-base to further theorise on how open source software communities perceive and are influenced by neighbouring commercial entities.

In the following, we start off with a literature review in Chapter 2. Afterwards, theory and methodology are discussed in Chapter 3 & 4. Results are explained in Chapter 5 and, lastly, we conclude our findings in Chapter 6.

[1] If not, it would rather be traded off for the other exit; more IPO's are expected towards the future (Waters, 2015).

Chapter 2

Literature Review

As a starting point, this thesis revolves around open source software literature. For further exploration, we briefly discuss the intersection with two other research streams: venture-funded startup and acquisition literature. Additionally, we will touch upon some peripheral research concepts to better contextualise the work. We finish off by synthesizing and discussing the research question based on the research gap.

2.1 Open Innovation

The research exploration of this study can partly be designated under the umbrella term of *Open Innovation*. As originally explained by Henry Chesbrough: *"Open Innovation is a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology. Open Innovation combines internal and external ideas into architectures and systems whose requirements are defined by a business model."* (Chesbrough, 2003). In other words, the *Open Innovation* paradigm treats R&D as an open system, which adds external in- and outflows to the picture.

Concurrent with the rise of *Open Innovation* there has been a parallel research body focused on the role of the community in the innovation process (West and Lakhani, 2008). In essence, developing open source software through the help of a community that crosses the firm boundaries fits the definition of Open Innovation. As such, West and Gallagher (2006) state that open source software is a great example of *Open Innovation*. Nonetheless, both topics look at innovation with different perspectives and, therefor, their research bodies focus on different mechanisms (e.g. *Open Innovation* mainly studies contractual agreements). Additionally, there is critique stating that open source software is not always a form of innovation (West and Lakhani, 2008). Taking this into account, we do not view open source software as a refined model of *Open Innovation*, like Wilson (2010), per se, but do perceive the concepts

as highly interconnected.

One element of *Open Innovation* is external technology acquisition. Lichtenthaler (2008) described this as *"...the absorption of external technologies e.g., by means of in-licensing agreements or strategic alliances."* As a pecuniary agreement that can increase or broaden technological knowledge, Granstrand et al. (1992) mention the acquisition of innovative firms as a typical strategy to achieve this. Thus, we categorise the acquisition of venture-backed high-tech startups as an approach to *Open Innovation*.

Without detaining any further, we discussed the concept for its inherent, definition-based similarities with the focus literature streams. As such, the research combination seems sensible as the two focal activities - developing open source software and acquiring high-tech startups - seem to be stemming from the same philosophy: opening up firm innovation boundaries.

2.2 Open Source Software

The literature body around open source software is (OSS) immense. von Krogh and von Hippel (2006) already pointed this out a decade ago, while the trend has not diminished since.[1] In the following we give a thorough overview of the subject. Even though it can not be fully exhaustive, we believe it represents a solid examination of the research field.

For readers with a limited knowledge about OSS, we strongly recommend the first chapter of (von Hippel and von Krogh, 2003), which summarises the history and clarifies important aspects in a very understandable way.

2.2.1 Defining (F)OSS

Free and open-source software (FOSS) is computer software that can be classified as both free and open-source software.[2]

The *Free Software Foundation*[3] represents a social movement spreading an ideology: *"Free software is software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, free software is a matter of liberty, not price. To understand the concept, you should think of free as in "free speech", not as in "free beer". We sometimes call it "libre software" to show we do not mean it is gratis."*[4,5]

[1] For example, looking at the search term "open-source software" in Google Trends, we don't see a big change since 2004. https://www.google.com/trends/

[2] https://en.wikipedia.org/wiki/Free_and_open-source_software

[3] http://www.fsf.org/about

[4] http://www.gnu.org/philosophy/free-sw.html

[5] It is important to note that this is not the same as *freeware*, which does not disclose source code and therefor permit modifications. https://en.wikipedia.org/wiki/Freeware

The *Open Source Initiative* arose later as a pragmatic and business-focused approach that mitigated the philosophic and political label associated with free software.[6] It released the *Open Source Definition* in the form of 10 criteria that give rise to a type of license that can be attributed to the distribution of software code.[7] In summary, it defines the free redistribution in absence of fees, the obligation of source code access, the possibility to redistribute derivative works under the same license, and a non-discrimination policy. In principle, open-source software imposes slightly weaker conditions than free software. Most open-source licenses in use today are free as well, with a few exceptions such as, e.g. `Open Watcom`. *"The two terms describe almost the same category of software, but they stand for views based on fundamentally different values. Open source is a development methodology; free software is a social movement."*[8]

Given the direction that the following research is aiming towards, i.e. management research, the focus will lie on open source software, and not free software, unless stated otherwise. This decision will increase the number of licenses in the research scope and makes this study take distance from idealistic bias.

2.2.2 The OSS Community

Towards the end of the 20$^{\text{th}}$ century there was a rise of projects that developed software under open source license agreements. These are self-sustained by a community[9], as described by Santos et al. (2012): *"...open source software projects comprise groups of developers and users geographically dispersed but connected together through shared values and the Internet (Herbsleb and Mockus, 2003; Stewart and Gosain, 2006)."*

The open source community has been extensively researched mainly due to its unique nature and as a novel model of innovation and organisation (von Krogh and Spaeth, 2007; von Hippel and von Krogh, 2003; Puranam et al., 2014; Zammuto et al., 2007). It represents a remarkable decentralised and cooperative form of self-organization, showing traits of high efficiency and enormous user participation (Koch, 2004), as opposed to conventional commercial software development. The Linus law presented by Raymond (1999) implies a higher output quality of this "bazaar-like" software development, compared to traditional methodologies.

Individual Motivations

von Hippel and von Krogh (2003) released a literature review and theoretical proposition to incite and give common ground to the emerging literature wave around the open source soft-

[6] https://opensource.org/history
[7] https://opensource.org/
[8] https://www.gnu.org/philosophy/open-source-misses-the-point.html
[9] On-line communities in general are a popular research topic (Butler et al., 2014, 2003; Faraj and Johnson, 2011).

ware community. An interesting question they address is one suggested by (Lerner and Tirole, 2003): *"Why should thousands of top-notch programmers contribute freely to the provision of a public good?"* It has been observed that adequate rewards can be provided to participants in open source software projects in a variety of forms. On the one hand, open source can be viewed in the collective-action innovation model, as a public good.[10] On the other hand, it also shows traits of the private innovation model, where developers (and firms) reap private benefits. So, accordingly, their answer lies in re-framing it as a "private-collective" innovation model, where incentives for private investment and collective action can coexist.[11]

This brings us to the question of what actually motivates developers & other participants to contribute to open source projects. In an attempt to pour the contributor's motivation in an equation, Lerner and Tirole (2003) see the net benefit of the contributor, as the immediate payoff (current benefit minus current cost), plus the delayed payoff (delayed benefit minus delayed cost). A large body of literature has addressed elements in- and outside that equation, which we summarised in Table 2.1.

So, there is a broad set motives that drive developers to contribute to OSS projects. Different kind of contributors are driven by diverging motives, which in turn determine the nature of their contributions in terms of quality, necessity, and challenge. It is also important to realise that for a given individual the motivation to remain active in a project can change over time, and will depend on certain project characteristics. In general, a line is drawn to separate intrinsic from extrinsic motivational factors, as they tend to polarise the aforementioned characteristics.

von Krogh et al. (2012) performed an exhaustive literature review of articles that cover the topic of individual motivations in OSS communities. They classify contribution motives as such:

- **Intrinsic**: ideology, altruism, kinship, and fun.
- **Internalised Extrinsic**: reputation, reciprocity, learning, and own-use.
- **Extrinsic**: career and pay.

On top of this review, a motivation-practice framework was proposed, where motivation in OSS is expanded from its isolation context to social sciences. They argue that the antecedents and consequences of contributor motivation are more complex than what current theories suggest. The social practice of OSS gives shape to standards of excellence, which has the strong potential to mutually influence contribution behaviour. Over time, increased participation in

[10] OSS is often considered through an economical perspective as a public good, being non-excludeable and non-rivalrous. However, according to Bessen (2006), this tends to be an oversimplification.

[11] https://en.wikipedia.org/wiki/Private-collective_model_of_innovation

(Raymond, 1999)	provides a threefold classification of contributor motivations: 1) benefits for own use of the content they provide, 2) the pleasure of programming itself, and 3) the reputation prospects of making good contributions.
(Stallman, 2001)	argues that the free software ideology feeds the open source movement.
(Hars and Ou, 2002)	share an article that categorises the motivation in: 1) internal factors such as intrinsic motivation and altruism, and 2) external rewards such as expected future returns and personal needs. They state that the external factors have a bigger weight and are more likely to be found with salaried and contracted programmers, instead of students and hobbyists.
(Kuan, 2001)	models open source as consumer integration into production, in which users organise to produce a good for themselves.
(Franke and Von Hippel, 2003)	show that heterogeneity in market demand can bring the creation of innovation toolkits (such as for the `Apache` security software). This allows users to modify the product to their specific needs, which leads to higher satisfaction in comparison to traditional market customization.
(Lerner and Tirole, 2003)	identify the delayed benefits as, the career concern and ego gratification, which are grouped together as signaling incentives, towards peers, potential employers, and the venture capital community. According to economic theory these are stronger (1) when the visibility of the performance to the relevant audience is higher, (2) with increased impact of effort on performance, and (3) when the performance is more informative about talent. The articles state that these 3 effects are indeed stronger for open source, in comparison to closed source. Under these conjectures, programmers will want to work on projects that attract a large audience. Additionally, an OSS project may well lower the cost for the programmer, due to the following phenomena: (1) the Alumni effect; programmers are often familiar with the code, as it is more likely that they saw the material in university, (2) customisation and bug-fixing; which brings about a private benefit for the developer (and her firm).
(Lakhani et al., 2002)	performed with the *Boston Consulting Group (BCG)* a broad `SourceForge` survey with the following key findings: increasing knowledge is the biggest reported benefit for contributors, and losing sleep the biggest cost.
(Hertel et al., 2003)	performed a study on 141 contributors of the `Linux` kernel project, in order to investigate the willingness to contribute, both at the community and team level. The main motivational factors were: identification factors as `Linux` users/developers, pragmatic motives for self-improvement and career advantages, norm-oriented motives related to reactions of relevant others (friends, family, colleagues), social and political motives to support independent software, and hedonistic motives out of enjoyment.
(Lakhani and Von Hippel, 2003)	take a deep-dive into the motivations behind the mundane, but necessary support and help tasks. They discovered that information providers will in most cases gain valuable information by studying other answers. In addition, the public posting of the provider's name gave the potential of reputation-building through helping. These effects strongly depend on specific features of site design.
(Haruvy et al., 2003)	present a survey analysis - building on (von Hippel and von Krogh, 2002) -, where motives for developer contribution were categorised in two channels. One is based on private incentives as future rewards and self-fulfillment, and will influence direct contributions. On the other hand, social considerations seem to dominate as drivers of product quality and product development speed.
(Shah, 2006)	makes an important distinction between the majority of developers who leave the community once their needs are met and a small group of hobbyists that remain active and handle unpopular, organisational tasks, who are vital for the long-term viability of the project. They found that motives tend to change over time and that reciprocity or fairness is an important factor driving the contribution of code to the community. Also, the governance structure, in terms of code control and ownership clauses, dramatically affects the participation choices of volunteers.
(Roberts et al., 2006)	explain that developers' motivations are not independent but rather related in complex ways. People who are motivated for the status, are more likely to get paid for contributing, compared to developers that are motivated by value-use. Status motivation enhances intrinsic motivation, which is not diminished by extrinsic motivation. Finally, they find that paid participation and status motivations lead to better contribution performance.
(Mehra et al., 2008)	found that some programmers receive bonuses by their employers for working on independent OSS projects, as it may result in strategic value for the firm of learning by doing progress for the programmer.

Table 2.1: Overview of Research Papers on Individual Motivation in OSS Community Participation

the social practice, can diminish the role of extrinsic factors and institutions. This opens up a new avenue for exploring the developer's motivation to take part in OSS development.

OSS Communities as Social Organisations

As communities grow, tasks become complex and require increased decomposition and management. So, there is the need for governance, control, and leadership to guide the development process. As these differ quite a lot from hierarchical methodologies in traditional organisations, it spiked another interesting research topic in the context of OSS, that views the community as a self-governing entity.

O'Mahony and Ferraro (2007) view the OSS community as a social system that coordinates, manages, and sustains itself. It investigates in the `Debian`[12] community how governance systems are designed and implemented. They propose a mix of bureaucratic authority on the

[12]https://en.wikipedia.org/wiki/Debian

one hand, with democratic mechanisms to delimit that basis of authority. Their research suggests that democracy plays a dual role: it ensures that the governance system represents the communities' interests, and provides the possibility to adapt the conceptualization of authority.

Dahlander and O'Mahony (2011) study an open source project in the perspective of lateral authority, which, unlike horizontal or vertical authority that are managerial, can be described as a task-based authority. In this paradigm, progressing towards the center is equivalent to the traditional "moving up the hierarchy". The main takeaway is that gaining lateral authority requires technical contribution at first, but coordination work is more critical at a subsequent stage. After gaining that authority, individuals spend significantly more time on coordinating project work and less on coding.

Di Tullio and Staples (2013) identified as much as 19 governance mechanisms to ensure productive participation. Based on this, they categorised three community configurations (*Open, Defined, and Authoritarian*), mainly characterised by the extent of development process management, and bottom-up goal setting.

Nakakoji et al. (2002) propose to classify OSS communities in three types based on their control styles and community structure. Exploration-oriented projects focus on sharing innovations and knowledge (e.g. `GNU`[13] and `Perl`[14]). Utility-oriented projects are there to satisfy individual needs (e.g. `Linux`). Service-oriented projects focus on supplying a reliable and stable service (e.g. `Apache`). Interestingly, evolution patterns are discussed where projects often change from one type to another over time.

Lerner and Tirole (2003) explain that the governance structure can be one person (e.g. `Linux`) or a group of people working with consensus (e.g. `Apache`).[15] The leader(s) usually commit the initial code and carry a form of real authority (Aghion and Tirole, 1997), opposed to traditional formal authority. As the project grows, leaders will spend less time with programming contributions, and more with recommendations, project management tasks, and the project's vision. In order to avoid forking and loss of developers, it is important that the leader communicates goals and evaluation procedures, and stays open for changes and improvements on the merit of the developers.

Additionally, O'Mahony and Ferraro (2007) mention that developers who engaged in organization-building behaviours, and made popular technical contributions, were more likely to become leaders. As such, they describe `Debian` as a meritocracy, where merit is not solely determined by technical contributions. Faraj et al. (2015) further explores the topic of leadership, expanding it to on-line communities in general. There, in the context of OSS, it is emphasised that one must also be able to dialogue about the communities' core subject. So,

[13] https://www.gnu.org/home.en.html

[14] https://www.perl.org/

[15] The paper does not explicitly mention a commercial firm. We believe that at the time there were not many examples of firms leading a community (rather firms taking part in the community). In this thesis, we will include that case.

high participation and sociability may not be sufficient for leadership, but involvement in technical topics is required. Similarly, Huffaker (2010) states that credibility and the use of affective, assertive, and linguistic diversity is typical for leadership positions.

Another study (Stewart, 2005) looks at how community peers evaluate others' reputation and how their own status gets established by the use of public references. Findings stated that as the number of peers who gave high (or low) status ratings of an actor increased, the likelihood of that actor receiving a future high (or low) increased. Moreover, as time passes the odds that an actor receives a public rating of any type is reduced. This implies that over time social status gets institutionalised and that for newcomers speed matters for the establishment of their status.

von Krogh et al. (2003) propose more detailed findings about the joining and specialization dynamics. Leveraging data from one OSS project they identify 4 elements to model newcomer behavioural strategies:

- Joining scripts: the type and amount of activity a joiner needs to go through to become a recognised developer.
- Specialization: how does the developer distribute his or her contributions over the software modules or components.
- Contribution barriers: this represents technical variation in modules, which is related to newcomer specialization.
- Feature gifts: entire features/modules that are written by newcomers, these are related to the specialisation in open source projects.

Looking at on-line communities in general, participant turnover is an important attribute. Ransbotham and Kane (2011) teaches us that on `Wikipedia` a balanced mix of new and experienced participants is the best way to create qualitative content. In other words, participant turnover should not be too high, nor too low, which, respectively, has a negative impact on knowledge retention and knowledge creation. Also, the technological infrastructure that supports the processes of the community, has an influence on the community size and resilience (Butler et al., 2014). Even though `GitHub` amounts to a strong technological infrastructure, participant turnover in open source software development is a critical problem in terms of talent retention. It increases for the worse, when there is less person-organisation fit for contributors (Sharma et al., 2010).

Theories, simulations and empirical studies in literature also draw a solid line between the self-organizing, open-source community and complex adaptive systems (Maillart et al., 2008; Agnihotri et al., 2012; Xu and Madey, 2004). Graves (2013) highlights this under the mechanism of preferential attachment, which shares similarities to the social network perspective of Xu and Madey (2004); Xu et al. (2005). Also, the growth of a project can be modelled

based on its size and developer entry-exit dynamics (Schweitzer et al., 2014).

In conclusion, communities are more than a group of equal peers. In the absence of traditional authority, governance mechanisms emerge to avoid anarchy and guide development, such as democratic bureaucracy, lateral authority, and leadership. These are driven by differences in developer social status throughout the community.

Success Recipes

There is a huge amount of open source projects out there, but only few are paired with an active community. So, what are typical characteristics of successful projects? Several papers offer suggestions about catalysts and must-haves for OSS success stories.

Before a project gets momentum, the founder must upload a critical mass of code, as an initial condition (Lerner and Tirole, 2003). This body of code must be substantial enough to depict the merit of the software, but leave a sufficient amount of challenges to attract developers. Additionally, it is important that the code be divided into distinct components, to allow parallel management and operations. A similar study confirmed these architectural requirements, called modularity and option value, and the need for their management throughout the project's life cycle (Baldwin and Clark, 2006). Another important antecedent is obviously the target audience of the software: the larger the size of the niche, to which the project belongs, the more developers and users that project can attract (Chengalur-Smith et al., 2010).

Now, the liability of smallness, tells us that the legitimacy of a OSS project is predicted by the size of its community. Also based on the liability of newness, a OSS project's age is positively correlated with the attraction for active users (Chengalur-Smith et al., 2010). Thus, many effects of success take place after initiation and are more difficult to control. Singh et al. (2011) investigate open source project success based on network social capital. It appears that teams with a high level of internal cohesion are more successful, but more external contacts with some technological diversity brings the project forward. As it is not trivial to define and measure success, Lee et al. (2009) proposes a model for OSS success based on 5 determinants: software quality, community service quality, user satisfaction, OSS use, and individual net benefit. The results indicate that these elements have specific reinforcing effects on each other, and thus form important ingredients for a OSS success recipe.

Looking into 4000 FOSS projects on `SourceForge`, Santos et al. (2012) noticed that community agents seemed to be drawn to only a few successful projects due to preferential attachment. This phenomenon is explained through project attractiveness, which significantly influences activeness, effectiveness, likelihood of task completion, and time for task completion

in the project. In turn, license type, type of user, application domain, and stage of development are identified as the main conditions that determine the perceived attractiveness of the project. Finally, Zhou and Mockus (2014) highlight the importance of long-term contributors for completing project-critical tasks. The initial behaviour and experience of newcomers is very determining for the conversion rate to long-term participants. They propose some empirically identified approaches to help with this recruitment process and contributor effectiveness in general.

2.2.3 Licensing

An important characteristic of open source software projects is the license(s), which addresses use of the copyright and roughly-said determines the freedoms for involved parties. As mentioned before, open source software is defined as having a license that complies with a set of requirements (see Section 2.2.1).

The copyright or code ownership is a complicated attribute of open source software. By default the copyright is given to the author of a piece of work. Projects can maintain a list of authors, but some administrative bodies and foundations explicitly ask to assign the contributions' copyright to them, or request a re-licensing agreement between them and the developer. This tends to become complicated when legal complaints or violations occur. Another aspect of intellectual property are the ideas and algorithms behind the code, which fall into the separate domain of patents and are usually not covered by OSS licenses. For gaining a better understanding about the legal aspects and OSS licensing in general, we refer to (Wilson, 2005; Laurent, 2004).

Software licenses[16] are hard to understand due to their legal complexity and jargon, and, being default contracts with no negotiation costs (Parker and Van Alstyne, 2005), are easily overlooked. The OSS license body is continuously developing, as it is being put to the test in court gradually and in turn receives improvement modifications, which leads to new versions and forking. Many different categorizations have been used throughout literature and less-formal writings (Lerner and Tirole, 2005b; Rosen, 2004; Sen et al., 2009). For the sake of overview, we will use the following:

– **Permissive licenses**: These are licenses with little restrictions. They simply require you to attribute the code to its original developers in modified code and documentation. Popular examples are the `MIT`[17], `BSD`[18], and `Apache`[19] licenses. This category is also referred to as academic, non-restrictive, or non-copyleft licenses.

[16] For an overview about the history of software licensing we refer to Section 2 of (Lerner and Tirole, 2005b)
[17] https://en.wikipedia.org/wiki/MIT_License
[18] https://en.wikipedia.org/wiki/BSD_licenses
[19] https://en.wikipedia.org/wiki/Apache_License

- **Moderately restrictive licenses**: These licenses require that when modified versions of the code are distributed, the source code must be made generally available. This mechanism is called the copyleft (a wordplay on the well-known copyright[20]). Typical is the `LGPL` license. This is also referred to as weak copyleft licenses.
- **Highly restrictive licenses**: This class refers to such licenses that do not allow (modified) versions of the code to mingle with other software that does not employ such a license. This mechanism is called "reciprocal" or "viral". The bible of this category is the well-known `GPL`.[21] This category is also referred to as reciprocal or strong copyleft licenses.
- **Corporate licenses** represent a family of alternative licenses, that are employed by commercial companies that have "opened up" their proprietary code. Typically, these licenses have specialised provisions to address copyright[22] and liability concerns of the corporate sponsor. Examples are the *Mozilla Public License*[23], *Sun Public License*[24], *IBM Public License*[25], *Eclipse Public License*[26] (Fitzgerald, 2006; West and O'Mahony, 2008). See Section 2.2.4 for further explanation.

Lerner and Tirole (2005b) kicks off the OSS licensing literature stream by proposing some impacts and issues that license choice can have on different groups:

- Commercial vendors and service providers: their opportunities are affected by the license. A permissive license will usually attract commercial software developers to write value-adding applications.[27] In this case "hijacking" might occur, where the company adds proprietary code to the OSS software and distributes the result with commercial fee licenses. Although this disrupts the dynamics of the open source community, it can offer a significant boost to the project. Also, if a programmer contributes code that infringes on a patent, lawsuits may of course hamper further development. The `GPL` offers a provision that restricts further distribution of the code, when such an event occurs.
- The community: the benefit that developers reap from contributing can depend on the license. For example, using a popular license will require less exploration transaction costs for the contributor, compared to using an innovative, less-familiar one. Also, when the project gets hijacked, it can demotivate contributors as they are deprived of some

[20] https://en.wikipedia.org/wiki/Copyleft
[21] https://en.wikipedia.org/wiki/GNU_General_Public_License
[22] In autonomous OSS communities (absence of a corporate sponsor) copyright is usually collectively owned by all contributors.
[23] https://en.wikipedia.org/wiki/Mozilla_Public_License
[24] https://en.wikipedia.org/wiki/Sun_Public_License
[25] https://en.wikipedia.org/wiki/IBM_Public_License
[26] https://en.wikipedia.org/wiki/Eclipse_Public_License
[27] Companies tend to take a different approach when they release their proprietary code to the public. We will discuss this in the next Section (2.2.4)

of the benefits (they need to pay for the extended version of the software, and will not be able to customise its source to their needs). Opposed to unsophisticated end users, software that is aimed at developers and system administrators is better released under a permissive license, to foster a strong community appeal.

- The end user: the license type will also influence the likelihood of forking. This can negatively impact the welfare of the end user due to incompatibilities and number of versions.
- Complementing or competitive projects: projects who wish to set standards (e.g. code geared towards the internet) are better off with a permissive license. But there are also dynamic strategic complementarities: if existing projects in the field have restrictive licenses, the licensor is recommended to use a similar one, so that the products can be combined. Reversely, future projects might base their licenses on precedent ones.

The paper looked to confirm this framework by analysing 40,000 open source projects on the `SourceForge` database. The results were that restrictive licenses are less common for: projects operating in commercial environments, aimed towards software developers and technical users. Restrictiveness is more common for: applications geared towards end-users, system administrators, and consumers e.g., desktop tools and games.

Fershtman and Gandal (2007) made an analysis on the top 71 projects of *SourceForge* and found that the output per contributor is higher when licenses are less restrictive. This has to do with the fact that restrictive-licensed projects have much more contributors. These results suggest that developers have a stronger intrinsic signaling motivation for restrictive projects; they are rewarded for small contributions by having their names added to the publicly available list of developers.

Stewart et al. (2006) investigates how the license choice influences the user interest and development activity. Based on a diverse sample of 138 projects, they found that users seem to be more interested in projects that have non-restrictive licenses. Developers tend to prefer non-restrictive-licensed projects as well when a corporate sponsor is present. However, when the latter is not established, this relation no longer holds.

Sen et al. (2009) contributes an empirical analysis of licensing by focusing on the developers' motivation and attitude. They found that developers who are driven by the cognitive challenge of programming are more driven towards copyleft licenses over permissive licenses, because they might wish to protect their intellectual contribution. Developers motivated by an extrinsic status factor are more likely to prefer the permissive license compared to moderately restrictive licenses. Those least restrictive licenses tend to attract a bigger audience, which in turn is beneficial to improve their status. However, it is mentioned that developers may see both non- and strong-copyleft licenses as beneficial for their status, as the latter attracts more serious developers. A second stream of observations state that programmers

who believe in freedom of redistribution are more likely to choose less restrictive licenses. In contrast, developers that see OSS participation as a social obligation, are attracted by more restrictive licenses. In summary, developers prefer licenses that are closely aligned with their motivations and beliefs, and thus it is important for OSS administrators to consider which kind of developers it wishes to attract upon selecting a license.

Finally, Singh and Phelps (2009) propose a social influence model of the OSS license choice. Looking at more than 5,000 projects hosted at `SourceForge`, they observed that the most important factor of influence to the license of a new project, are the license types that are more socially proximate in the licensor's social network. A new project is more likely to select a certain license type, if that license has previously been chosen for similar projects and if those projects are large and successful. Additionally, project managers with longer tenure in the OSS environment are less susceptible to social influence.

In conclusion, we find that licensing is an important foundation of an OSS project, it can determine many dynamics in terms of commercial complementary activity and contributor motivation. We proposed a classification with increasing level of permissiveness. There is no absolute right or wrong on that scale, rather the choice should be determined case by case. Some projects will even go as far as creating a new one or license multiple components of the code differently, in order to obtain optimal dynamics.

2.2.4 OSS 2.0

As OSS attracted great attention in research and media, opinions on its business potential and market success were scattered (Fitzgerald, 2005). This led to the coining of the term OSS 2.0 (Fitzgerald, 2006), that took an outward perspective on the matter, focusing less on the community[28] and more on the surrounding ecosystem. Organisations are trying to implement open source success on traditional development projects in the climate of outsourcing and off-shoring. These initiatives have been labeled as "inner source", "corporate source", "community source", and "gated source" (Gurbani et al., 2005; Dinkelacker and Garg, 2001; Shah, 2006).

Much discussion in literature has been centered around the relative benefit of open source software versus proprietary code. Many papers highlight the benefit of OSS, as superior software quality and robustness (Raymond, 1999; Krishnamurthy, 2003), decreased error and bugs (Kuan, 2001), better customization and flexibility (Bessen, 2006; Krishnamurthy, 2003), better development process through increased integrity (Johnson, 2006), and strong community support (Krishnamurthy, 2003). On the other hand, proprietary software might be better documented (Lerner and Tirole, 2003), have improved security, provide better usabil-

[28] Much of the preceding research merely focused on project characteristics and developer motivation, due to the massive amount of public repository information that is easy to obtain.

ity and versioning control (Krishnamurthy, 2003), and give a more transparent total cost of ownership (Ven et al., 2008). Conclusions remain elusive, but as it seems both models can engage in a balanced competition, where each addresses a particular target market (Sacks, 2015; Gaudeul, 2007). Bonaccorsi et al. (2006) confirm that hybrid strategies[29] for firms is not merely a transient phenomenon, which supports the idea that the market permits the survival of both paradigms. This equilibrium is once more defended by Campbell-Kelly and Garcia-Swartz (2010) as a so-called "move to the middle", where proprietary vendors have embraced the OSS methodology, and open source firms have resorted to traditional commercial practices. In the context of the software stack, these players have learned to co-operate as well as compete with one and other.

The Relationship between Commerce & Community

OSS 2.0 introduces commercial activity around open source software. This gives rise to a paradigm that splits on the governance characteristic: the distinction between commercially and community-focused projects. We relate this to four approaches in which firms establish relationships with a community. Based on the 2*2 matrix of Dahlander (2007), there are two typical characteristic dimensions. The first is distinguished by two cases: whether the project was firm-initiated or community-initiated. The second represents the level of firm participation in the community. The managerial ladder in (Grand et al., 2004) proposes four increasing levels to which extent a firm may be participating in a community.[30]

Dahlander and Magnusson (2005) interpret the relationship between a firm and a OSS community as a difficult interaction, because both entities have a different rationale for existence and of the absence of formal agreements or contracts. They derived three approaches: symbiotic[31], commensalistic[32], and parasitic. In the parasitic approach the firm focuses on its own benefits, without taking into account that its actions might influence the community. They refer to a symbiotic approach when the firm tries to co-develop itself and the community. This requires that the firm is directly involved in community development, by contributing code and offering infrastructure for the project. The challenge typically lies in managing the opposing needs of openness and control. The commensalistic approach, as an intermediate way to relate, is to benefit from the co-existence with an entity while leaving it without harm. The idea is to thrive on community resources, while keeping the direct involvement in the development of those resources to a minimum. How the legal mechanisms, that the firm uses for the code, relate to the norm and values of the community is of importance. The symbiotic approach clearly offers more possibility to influence the development, but require from

[29] In a hybrid strategy a company develops software under proprietary licenses, as well as open source licenses.
[30] This paper actually focuses more on the business side of OSS, which will be discussed in the next section.
[31] As it seems, the word "symbiotic" is used quite differently by Lerner and Tirole (2005a). We prefer the currently provided definition.
[32] Has also been referred to as "altruistic".

the firm flexible management to address several community issues. To mitigate this cost, a commensalistic approach seems more appropriate, but it may be significantly harder to get acceptance from the community for the firms' commercial use of the communal resources. This can deteriorate the relationship to a parasitic one with bad firm reputation. The findings are visualised in Figure 2.1.

Lerner and Tirole (2005a) refer to firm-initiated projects, as the "code releasing" strategy, where a firm transforms proprietary software into a public OSS repository. In this case the firm will usually introduce a governance structure where they maintain a significant amount of control. This approach makes sense, when software profitability is little compared to the complementary offering, and the product could use a boost in market share. Concerning community-initiated projects, there are quite some challenges when a commercial entity tries to obtain leadership in an OSS project. It is hard to obtain community credibility and trust, as developers fear that objectives are not aligned in such a situation.

Shah (2006) takes the level of firm control one step further, by studying contributor's motives for an independent open source project and a "gated source"[33] project that is governed by a firm. In the latter case, some risks of OSS are mitigated, meanwhile gaining from the community development benefits. The paper mentions that such a level of code control and email lists domination will inhibit the motivation of need-driven and hobbyist developers. Also, the private firm ownership of the source code dismantles the community development process, as developers fear that the code might no be accessible at a later point in time. Similarly, the degrees of freedom for the contributors are diminished, due to restrictions on use, modification and distribution.

West and O'Mahony (2008) look at the same distinction, but considering cases that don't cross the border of the OSS definition. They ask how OSS communities that have been founded by firms, differ from autonomous counterparts, by looking at 12 open source communities designed by corporate sponsors. Three design dimensions were considered: intellectual property (IP) rights, development approach, and community governance model. Decisions in this area give rise to a specific participation architecture with two types of openness: transparency and accessibility. Transparency describes the possibility to follow and understand a communities' production efforts, accessibility represents to what extent an external participant can influence that production. The observation was that sponsors were more likely to offer transparency, than accessibility. However, restricting access to development processes, limited the community's ability to attract new members and grow. So, firms face a control

[33]The license is not open source. Participants need to explicitly accept the license by the click of a button and may then download, use, and modify the source code. For commercial purposes a royalty fee needs to be paid to the firm.

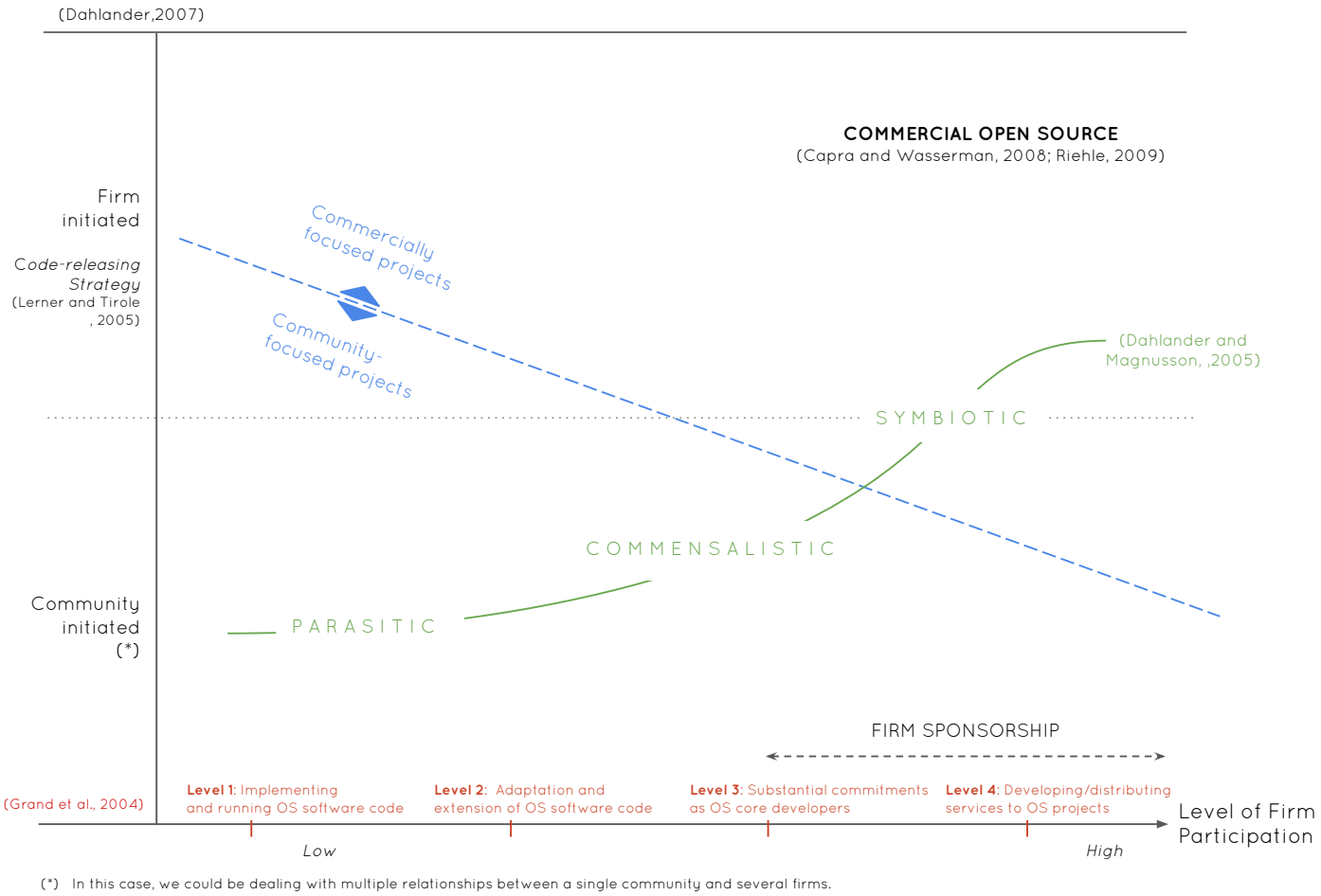


Figure 2.1: Visualisation of relationship ontology between firm and OSS community. The blue annotations refer to the paradigm we described in the beginning of this section.

versus growth trade-off in community design.

Capra and Wasserman (2008); Riehle (2009) make a clear distinction between commercial and community open source.[34] Commercial open source projects are controlled by a single firm that (almost[35]) fully owns the copyright of the code, which allows it to attribute multiple licenses to the code. Consequently, they distribute the code under an OSS license and a commercial license - which is referred to as the dual-licensing strategy.[36] Community open

[34]Capra and Wasserman (2008) proposes an interesting framework, allowing to position a project on a scale, ranging between these two extremes.

[35]In some cases external contributions are re-licensed without transmission of full ownership, or so small that property right issues are neglected.

[36]http://commons.oreilly.com/wiki/index.php/Open_Sources_2.0/Open_Source:_Competition_and_Evolution-/Dual_LicensingDual_Licensing

source projects are autonomous and controlled by a community of stakeholders in the form of a social structure. The code ownership is attributed to original authors, i.e. it is distributed throughout the contributors. The software is released under a single[37] OSS license.

Dahlander and Magnusson (2008) additionally adds three themes and six tactics for how firms relate to communities:

- Accessing: extending the resource base. A first tactic is to establish a new community. Although it is initially difficult to attract outsiders (especially in niche markets) and this approach carries high upfront (management) costs, it works as a marketing tool to create brand awareness. A second tactic is identifying and using existing communities, which offers greater flexibility, but is difficult to predict strategically. For both tactics, license problems are common.
- Aligning: connecting the firm's strategy with the community. A first tactic is adopting licensing practices to clarify ownership of the product. This entails extensively researching potential legal consequences, and being specific and not too demanding in license practices. This is crucial to gain trust from community participants. A second tactic is influencing the direction of development in communities. Firms can not enforce direct control over community actions, so they use more subtle means, such as financial incentives for individuals (Dahlander and Wallin, 2006) and maintaining task attractiveness by internalizing less interesting activities.
- Assimilating: integrating and sharing results. A first tactic is to devote resources to evaluate and select source code from the community. A second tactic is feeding back non-strategic source code developed within the firm to communities, which increases legitimacy of the firm, as they are "giving something back" to the community.

According to Stewart and Gosain (2006), programmers are more motivated when the sponsoring organisation[38] is a non-market organisation, like a university, instead of a firm. The relationship between a sponsoring firm and the community is strongly influenced by how the OSS developers perceive the commercial entity in terms of openness and credibility (Spaeth et al., 2015). West and O'Mahony (2004) says that creating a spin-out, a firm-initiated open source project, shares many of the same issues as a community managed one, such as building a collaboration infrastructure, designing governance mechanisms, and making key decisions about licensing and other external factors. The ongoing relationship between the sponsor and the community will raise questions of ownership, decision rights, and control. For the

[37] The authors excluded the case of a single project that is split up in different code components, receiving different licenses, although all open source.

[38] Organizational sponsorship indicates a publicly displayed affiliation between an OSS project and an organization.

sponsoring firm there is an essential tension/trade-off between appropriating the returns versus providing incentives for adoption. In this case a sponsor's desires to guide the project to meet its own needs must be weighed against providing incentives for participants to join and contribute to the community.

Finally, we finish off by lifting the OSS community and its symbiotic firm to the general context of *Organization Science*. Gulati et al. (2012) gives conceptual foundations to study novel organisations comprising of multiple legally autonomous entities. These are referred to as meta-organisations, that are characterised by permeable boundaries, less authority hierarchy, fewer employment contracts, and fluidity of structures.[39] They confirm us that the meta-organisation's designer - the OSS firm - must recognise and accommodate members' organisation-level strategies and provide them a guiding economic logic for collective action.

OSS Business Models

After extensively describing the relationship between communities and firms, naturally comes the question: how do firms make a profit from open source software? OSS firm strategies, business models, revenue models, offerings and product features have been studied in depth. In an attempt to get an exhaustive overview about how firms create business around and with OSS, we propose 6 categories of business drivers based on terms from previous research. It is important to realise that these categories are not distinct, and may share characteristics. As such, a firm's business model or specific product could be assigned to multiple categories. The goal of this categorization is to represent the focus and highlight found throughout literature. A summary is given in Table 2.2.

Market business drivers represent indirect tactics, where companies invest in open source software to drive revenue in other channels. A common term to describe this is the "loss-leader" approach. For example, `HP` promotes open source to facilitate the sales of their hardware products (Fitzgerald, 2006). `SOT` decided to release their source code as a marketing device (Dahlander and Magnusson, 2008). In a different perspective, the OSS stack can be seen as a platform ecosystem. Large companies take part in commoditizing the platform to sell complementary code (patronage (Koenig, 2006)). Smaller companies will seek to capture value as well in the complementary channel, which is typical in the case of `Linux`.

Resource drivers address newly emerging concepts that permit more efficient resource use for firms. Software development is often partly outsourced to the community, which reduces R&D costs and increases innovative capability (Andersen-Gott et al., 2012). Additionally,

[39] Following the article's taxonomy, we see the aforementioned case as a combination of "open community" and "managed ecosystem".

networks of vertically or geographically dispersed firms can more effectively support themselves in the OSS context (Ghosh, 2006).

Licensing represents the well-known dual-licensing approach for usage rights, warranty indemnification, maintenance, release management, and re-branding (Weikert and Riehle, 2013).

Distribution revenue streams are typically obtained for combining/integrating open source software, packaging, distributing on several media, and selling accessories such as translated documentation, branded merchandise, and other ancillary products.

Software upgrade business drivers, is a typical tactic where added functionality, utilities, plugins, performance, and certification are not included in the open source licensed software. For the customer to access those, a fee must be paid to the firm.

Hard services is a category we define as services that require additional programming and configuration effort in the process of bringing the open source software in a product to the customer. These include customization, tailoring, hosting, ad hoc solutions, and configuration.

Soft services are supportive services, such as training, education, consulting and custom certification.

A firm's decision to take part in the OSS game, is not a "all or nothing" option. There are many examples of "hybrid" companies out there, not all follow a pure play strategy (Scacchi et al., 2006; Rolandsson et al., 2011). This is clearly laid out by Grand et al. (2004) as a four-level management model of increasing private resource allocation. Level 1 sees the firm as a user, contribution limits itself to occasional feedback to the community by deployment developers. Level 2 firms deliver OS software as a complementary asset to a proprietary product, and therefor develop adaptations to the source code (typically for new hardware). Level 3 represents open source software as a design choice for certain projects, although the revenues are still based on commercial licenses. Level 4, identifies OSS as a full part of the business model, where the firms distributes services to the OS projects. These firms usually contribute significantly to the software in order to get close to the community, but seem to experience less resistance than dual-licensors for example. We included this ladder in Figure 2.1.

We can clearly see that the market has found numerous creative ways to capitalise with and around OSS. However, in the analysis above only opportunities are presented - and no

perils. Not all products have high profit potential, as they strongly depend on niche and utility characteristics (Krishnamurthy, 2003). Riehle (2009) introduces an entire new business function for companies who wish to reap resource drivers, called community management, which requires significant management resources and know-how to function properly. Also, the internal software development processes require thorough adaptation. Other conditions include an in-depth understanding of the property rights implications and even a sensitivity to the social norms of the community (Grand et al., 2004). Finally, not any revenue model can be chosen, as its feasibility is determined through business model conditions, set by community, firm and software characteristics (Rajala et al., 2006).

2.3 Venture-Funded OSS Startups

(Ye, 2008) formulated a clear definition for a venture-funded OSS startup. This is not a popular study, but we shall reuse and slightly adapt it for the sake of guidance. A venture-funded OSS startup is a company that:

- was established after 2000
- has no public offering
- is not a subsidiary of a parent company
- generates its revenue on market offerings that rely on OSS
- has received institutional financing from one or more independent VC firms and bank-based VC firms

2.3.1 Open Source Software & Venture-Funded Startups

Open source software is without a doubt a considerable approach as a business model for high-tech IT startups. There are enough successful examples out there (see Appendix A). For venture capitalists, however, it has been quite a volatile ride. Starting out around the mid-end nineties, it got heavily set back by the *Dotcom* bubble (Sterne and Herring, 2006). After a few years, copious investments started to reappear (Blau, 2006; Aslett, 2008; Byfield, 2008), even though at the time there were no scientific methods available and many uncertainties in (e)valuating this new breed of companies (Puhakka et al., 2007). Some articles raised concerns for a new techno-economic bubble (LaMonica, 2005; Cook, 2005; Greenemeier, 2005). However, as a couple of years passed, the hype regained balance and was set off for a realistic growth phase (Phipps, 2010; O'Grady, 2010).

Wood (2005) outlines some of the business benefits of OSS for startups which are under strong time and resource pressure. The paper distinguishes between companies simply using open source software and those releasing open source software. The focus of our study lies in

Business Driver Categories	Market	Resource	Licensing	Distribution	Softw. Upgrade	Hard Services	Soft Services
(Hecker, 1999; Rajala et al., 2006)	Loss-Leader; Sell it, Free it; Software Franchising; Widget-Frosting		Brand licensing	Accessorizing; Branded Merchandise		Support & Service Enabler	
(Spiller and Wichmann, 2002)	Interest Enablers; Retailers of Complementary Products			Retailers of OSS Distributions (general/niche/speciality/Linux)	OSS Development	Service & Support Providers	
(Krishnamurthy, 2003)				The Distributor	The Software Producer	3rd-Party Service Provider	
(Kooths et al., 2003)	Selling Complementary Hardware		Dual-licensing	Packaging	Software Add-ons	Customizing; Implementation	Training; Consulting
(Dahlander, 2005)	Black-boxing		Licensing				Support; Education; Consultancy
(Koenig, 2006)	Embedded; Optimization; Patronage		Dual-Licensing				Support; Consulting; SaaS; Transaction; Advertising
(Bonaccorsi et al., 2006)		R&D	Maintenance	Distribution		Development of ad hoc Solutions	Support; Training; Consulting
(Ghosh, 2006)	Extending Hardware; Promotion	Network	Dual-Licensing	Integration		Customization	Support; Training: Consulting
(Gruber and Henkel, 2006)	Embedded Linux						
(Fitzgerald, 2006)	Loss-leader; Hardware-Facilitation; Platform Bootstrapping; Commodification	Development Cost-Reduction	Dual-Licensing	Accessorizing			
(Feller et al., 2006)		Network					
(Henkel, 2006)	Embedded Linux						
(Comino and Manenti, 2011)			Dual-Licensing				
(Ågerfalk and Fitzgerald, 2008)		Outsourcing					
(Dahlander and Magnusson, 2008)	Marketing via OSS	Minimizing R&D Costs	Dual-License	Translation	Extra Functionality		
(Riehle, 2009)	The OSS Sales Funnel		Core Product		Whole Product	Operational Comfort; Consulting Services	
(Campbell-Kelly and Garcia-Swartz, 2010)			Dual-Licensing; Maintenance	Packaging		Solutions	After-sales Support; Training; Consulting
(Lakka et al., 2011)	Marketing; Hardware Manufacturers; Embedded	R&D Cost Savings; Network	Dual-License	Distributor; Bundled Software; Ancillary Markets & Editions	Added-value Editions; Commercial on OSS	Host-based Service	
(Rolandsson et al., 2011)			Dual-Licensing			Market-Driven Tailoring; Solutions Provider	Support; Education; Customer-Oriented Consulting
(Andersen-Gott et al., 2012)		Improved Innovative Capability; Cost Reduction			Certification	Complementary Services	
(Weikert and Riehle, 2013)	Complementary Hardware		Non-Copyleft Usage Rights; Warranty; Indemnification; Maintenance; Release Management; Re-branding; Perpetual License	Documentation; Software Distribution	Advanced Core Product; Utilities & Plugins; Improved Behaviour; Certification	Custom Implementation/Certification; Installation; Configuration; Data Migration; Hosting	Support; Training; Consulting

Table 2.2: Summary of OSS revenue models

the latter. For that group, open source software can offer a quick and cheap path to market and testing/development cost minimization. Nonetheless, they warn that these benefits will only occur if a community arises, which in turn requires significant management and software attractiveness. For releasing software as open source, permissive/BSD-style licenses are discouraged. Additionally, it is mentioned that traditional GNU-style licenses are not ideal either: *"Releasing under GNU-style licenses is very popular in some circles..., but those licenses fail to address certain issues of importance for startups. In particular, startups wishing to pursue a patent protection strategy must ensure that their open source license does not invalidate their patent applications. Two particular licenses are recommended to provide startups adequate intellectual property and market protections... The Mozilla Public License (MPL) version 1.1* [40] *and the Open Software License (OSL) version 2.1* [41]*... Version numbers of these licenses are important and the newer versions should generally be used."*

Schaarschmidt and von Kortzfleisch (2014) presents a study focusing more on the venture capital (VC) investment in OSS. As commercial use of OSS is still in an early stage, there lacks evidence on how effectively different commercialization approaches work. Furthermore, the investment often - but not always (Wood, 2005) - lacks traditional protection mechanisms such as patenting, and investment in public goods, to which open source software shares many characteristics, is considered risky. So, venture capitalists seek to minimise risk through non-traditional strategies. The paper set out to find differences in risk-affinity of investors. The risk was estimated based on community tenure and previous venture investment round number. These propositions were not confirmed in a analysis of 45 OSS project investments. However, it concludes that venture capitalists have by tendency similar perceptions about the potential and risks of OSS, as many agents have a low share of OSS in their portfolio managed through heterogeneous risk strategies.

Gruber and Henkel (2006) presents a study on the managerial implications of venturing in the OSS space under the context of e-entrepreneurship. A combination of in-depth interviews and large-scale surveys from industry participants in the field of embedded *Linux* is analysed to shine light on differences to the challenges reported in general entrepreneurship literature. Three key challenges for new venture management are categorised as liabilities of newness and smallness, and market entry barriers. Results showed that those three key challenges were reduced by the open source software development model. Although this effect facilitates the venture management process, another main challenge arose: the need for in-depth industry knowledge or technical expertise. This was mentioned in the context of embedded *Linux*, which is a very complicated and huge code environment. We wish to add that other challenges, such as community building in the case of code-releasing, might have been overlooked, as the data set relies on the massive, existing *Linux* community.

[40] https://www.mozilla.org/en-US/MPL/1.1/
[41] https://opensource.org/licenses/osl-2.1.php

In Gorman (2006), VC common sense states as a general rule that for early stage startups in the consumer Internet space and/or the 'open source' area being 1 in market share is much more important than profitability. Now, open source is a powerful market diffuser, which has more potential to obtain market leadership. Nerney (2013) provides some more insights by an interview with *Salil Deshpande*, a *Bain Capital* OSS venture capitalist. He tends to see a grid of software problems that companies face. Some of the boxes have been disrupted by open source, others haven't. Startups who target areas of the latter, form promising leads.

2.4 High-Tech Startup Exits: Acquisition

Brau et al. (2003) points out that a venture exit boils down to a choice between an initial public offering (IPO) or a takeover by a public or private acquirer.[42] Agarwal and Helfat (2009) mentions acquisitions as an avenue for strategic renewal taken by larger technological firms. As such, it represents a transaction bringing together two different industry players, matching their needs. Technology-based industries, among which computer software and hardware, typically ranks amongst the most active sectors in the Securities Data Corporations annual filings of MA (Ranft and Lord, 2000). In the following we will touch the surface of the heterogeneous world of acquisition literature.[43]

Acquirer & Implementation

Most acquisition research has been framed in the perspective of the buyer. The following papers look at the transaction conditions (pre-acquisition decision-making) and the implementation aspects (post-acquisition performance). An important characteristic of acquisitions is whether the target firm is public or private. Shen and Reuer (2005); Capron and Shen (2007) investigate this distinction. One of the common findings is that acquirers favor private targets in familiar industries, and turn to public targets to enter new business domains. In the OSS domain most acquisitions take place within its industry (Appendix A), which suits our direction of looking at privately-owned ventures.

Benson and Ziedonis (2005, 2009) published two papers that investigate returns and performance of startup acquisitions through the channel of corporate venture capital (CVC). Building on trends in IT-related sectors, they identify the acquisition as an important mode of entrepreneurial-firm exit: *"Based on our calculations using proprietary data from VentureOne, acquisitions are three times more likely than IPOs as an initial exit event for IT startups;..."* The first paper, identifies share value destruction patterns, probably stemming

[42] We will not further study the initial public offering, because it is a quite different process than a acquisition and for our application area (OSS startups) there are insufficient examples to allow thorough analysis of the affair.

[43] We do not look into merger literature. We talk about a merger when two similar-sized organisations are melted into one. In this thesis, acquisitions are in order as we look at small venture startups that get bought by more dominant firms.

from managerial overconfidence or agency problems at the CVC program level. The second one presents a study that looked at 242 technology startup acquisitions. They discovered that firms consistently engaging in CVC activities, reap higher returns from startup acquisitions, and simultaneously obtain a window on upcoming innovative technologies.

Grandstrand and Sjölander (1990) looked at acquisitions of small technology-based firms by large technology-based firms. Although perhaps obsolete, it reveals some interesting findings to be mentioned:

- Competition among buyers leads to reduced transaction times, higher prices, and 'underdeveloped' firms with unfinished technology being acquired.
- Continuity in top management and key R&D personnel of the small firm is associated with the acquirer's success.

Puranam et al. (2006) states that upon acquisition of small, technology-based firms, structural integration decreases the likelihood of introducing new products for target firms immediately after acquisition and for target firms with limited innovation development trajectories. In a subsequent study (Puranam et al., 2009), conditions were proposed under which structural integration is considered more or less necessary. Although interdependence between acquirer and acquiree increase the motivation for structural integration, preexisting common ground can support alternative ways of achieving coordination.

Ranft and Lord (2002) developed an empirically-grounded model of technology and capability transfer during acquisition implementation. It states that the greater the tacitness and social complexity of knowledge underlying an acquired firm's technologies and capabilities, the more difficult and longer is the transfer during acquisition implementation. Also, retention of key acquired employees facilitates the preservation of that knowledge. The relative size and relative performance of the acquired firm are positively associated with autonomy and retention of key employees but negatively associated with communication between the two organisations.

Acquiree

Graebner and Eisenhardt (2013) set out to look at the seller's perspective, which was poorly understood. It challenges the assumption that being acquired is a sign of weakness and looks to rectify the image of the seller as unimportant, unsuccessful, and reluctant. They detected that private equity sellers are often pushed toward acquisition by difficult, albeit normal strategic hurdles, such as chief executive search or funding round and by strong personal motivations for sale. Nonetheless, sellers are more likely to be pulled towards acquisition by attractive buyers that offer synergistic combination potential and organizational rapport.

Acqhire

In the following we wish to highlight a novel acquisition phenomenon brought to us by Coyle and Polsky (2013); Selby and Mayer (2013): the 'acqhire' a portmanteau of the words 'acquire' and 'hire'.[44] They investigate the mechanism where larger dominant firms acquire smaller startups in order to obtain highly skilled personnel as a resource bundle instead of normal hiring procedures.

Coyle and Polsky (2013) addressed this literature gap as first. They wondered why firms take so much trouble and expense instead of simply hiring the people. One reason is to eliminate litigation risks by investors of the venture who would dread the loss of engineering talent. Naturally this argument only holds if there are valid legal claims to uphold such a lawsuit. Other reasons are nonlegal mechanisms, such as reputational concern, self-image, and a desire to avoid social sanctions, that hold the entrepreneur back to defect for short-term financial benefits to the loss of the venture's stakeholders. In this category also belongs the prestige of a "sale" to a leading technology company. Finally, for the entrepreneur tax benefits occur in such a transaction as ordinary income can be classified as capital gain.

Selby and Mayer (2013) mention three distinct benefits as: *"...1) the preservation of dynamic capabilities and tacit knowledge embedded in the startup's team dynamics; 2) the prevention of knowledge leaks which might hasten the decay in value of the new human capital; and 3) the protection of the acquired firm's innovation potential."* Two conditions are recognised for companies to take this approach: *"...when human capital activities are closely tied to core competencies and address issues facing high exchange hazards, and when the innovation problems they face require solutions in a relatively short time frame and may benefit from hiring a team of people who work well together and bring a different approach to the problem."*

OSS

Goth (2005) released an article correctly predicting some acquisition activity, which realigns strengths and liabilities in the vendor community. Tunguz (2015) mentions that VC's are motivated in OSS by substantial exits in the sector, mostly in the form of strategical acquisitions. Phipps (2014) talks about the maturity of the open source phenomenon. He says that giant businesses of the future don't monetise open source; they monetise innovations made possible by open source. Now, if that statement holds, it still justifies OSS startup acquisitions by giants, due to the "acqhire" phenomenon.

[44]The term appears in many different spellings throughout popular press. Ben Zimmer summarises a clear history of the word route to its original source: http://www.visualthesaurus.com/cm/wordroutes/buzzword-watch-acq-hire/

According to Gorman (2006), the primary challenge for OSS is that many potential acquirers may come from traditional business models and have to accept this new open source model that does not include ownership of the intellectual property.

2.5 Addressing the Gap

In the previous sections we visited several literature streams. The goal of this master thesis is to explore the cross-section. We identify it as a research gap, as no papers were found that focus on that research field combination: OSS community, startups, and acquisition. Figure 2.2 represents a visual summary of the literature streams and gap. Going through the literature of open source software we went from the community, through licensing, and ending with a commercial focus.[45] In this review, the acquisition was mainly looked at in the context of venture-funded startups, but obviously general acquisition literature encompasses more than that.

For a long time now OSS startups have been entering the market (Wood, 2005; Rosenberg, 2010), and the movement is maintained with more acquisitions appearing (Tunguz, 2015). In the case of sponsored OSS projects, we have learned that communities are influenced by the perceived firm attributes (Spaeth et al., 2015). We also know that developers can be motivated to contribute to OSS for career opportunities and reputation (von Krogh et al., 2012).

Our proposal is based on the fact that an acquisition represents an event that changes certain dynamics in the firm's ecosystem, which includes the OSS community that it sponsors. The motivational landscape of the community will shift due to changes in attraction and repulsion motives of contributors, according to existing research. We want to capture this transformation via this study and seek to confirm specific conjectures that take place. This leads us to a causal research question, that will further be dissected in the next chapter.

Research question: **What changes arise in the dynamics of an open source software community, when its sponsoring startup undergoes an acquisition?**

2.5.1 Validity

In the following we raise and defend some questions to challenge the validity of our research question.

– *Does it appropriate the current literature landscape?*

[45] (von Krogh and von Hippel, 2006) gave a similar categorization: 1) motivations for contributions, 2) governance, organisation and innovation, and 3) competitive dynamics. Now, we have taken a different approach, which seemed more suitable for this thesis. The most important distinction is the focus on the community versus the firm. Additionally, we pinpointed licensing, as an important mediator for the entire innovation process.

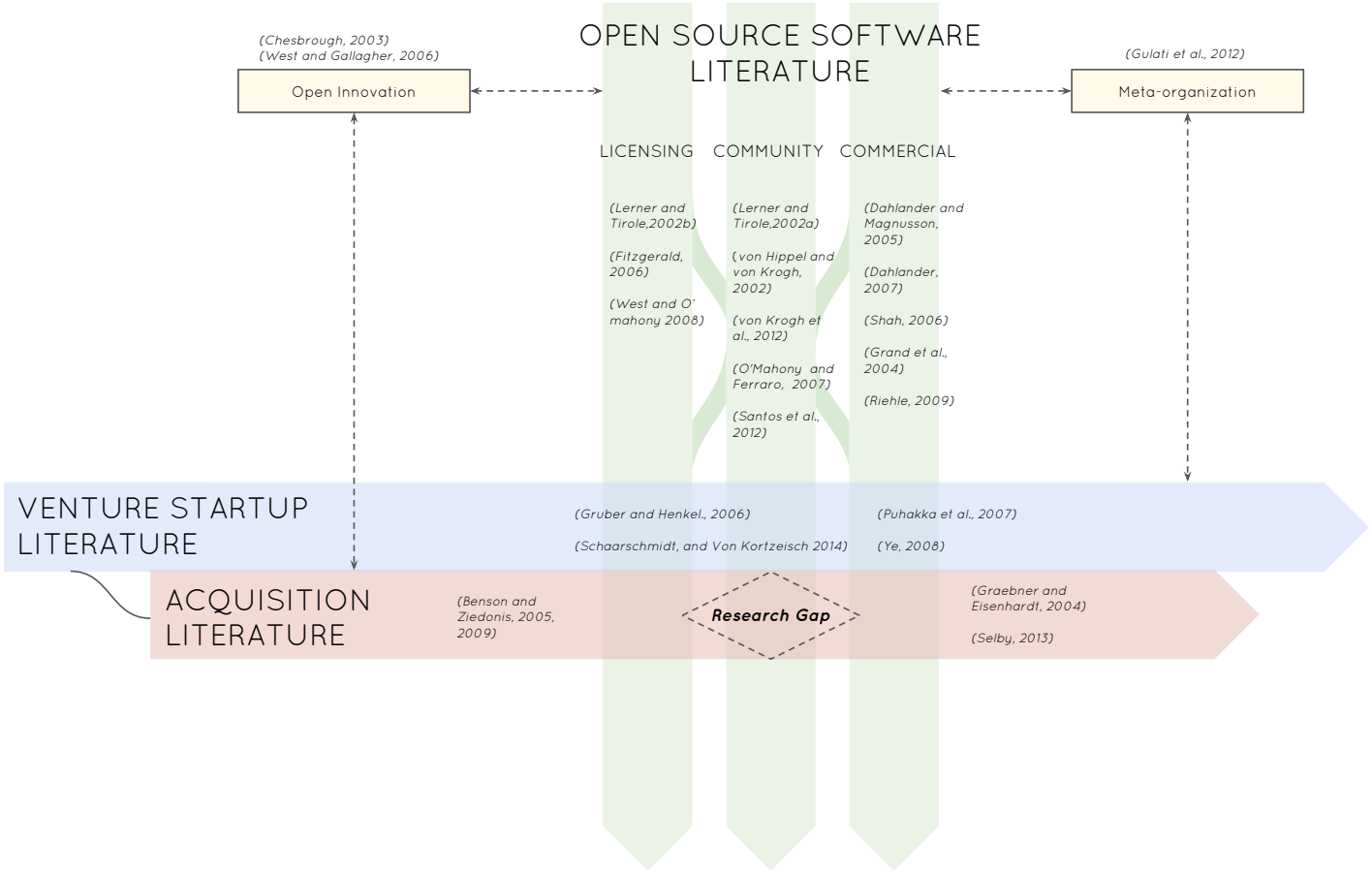


Figure 2.2: Summary of literature review. The cited articles in the image are illustrative selections and non-exhaustive. The thick colored lines represent literature streams that define the research gap, while the yellow boxes are peripheral concepts from literature that were simply referred to.

We believe it does, as it clearly addresses a research gap.

– *Is the question not too far ahead in the context of previous studies?*

The neighboring research fields are well covered. Assuming we are the first to look at this specific field, our research question remains broad enough to capture extensive observations and give general insights to future research.

– *Are the imposed boundaries not too constraining? In other words, is it worth looking at? Are we not zooming in too much?* OSS startups have been acquired for more than ten years and the trend is not decreasing. In 2008, such transactions amounted to more than half a billion dollars (Aslett, 2009; Rosenberg, 2010).

– *Who cares?*

We want to inform executives about community risks when they acquire an OSS startup. Given the size of these investments, we are pretty convinced they would care.

Finally, we wish to confirm this proposal with a web article that exactly fits our research question: Babock (2008). It identifies the risks of an OSS acquisition as, weakened developer and user community relationships, leadership exit, development moving from open source software toward proprietary product, and slow innovation. On the other hand rewards are more developers and access to a larger customer base. Although we will not dwell any further on this practitioner's view, it shows us that others have thought about this issue.

2.6 Conclusion

In this chapter, we ran through several research fields that circumvent our area of interest. We uncovered three core literature streams: open source software development, venture-funded startups, and acquisitions. We made connections to peripheral research frameworks, such as *Meta-organisations*, and *Open Innovation*. In doing so, there were difficulties to fully cover all the literature bodies in depth.[46] Nonetheless, we believe to have found an appropriate balance between depth and breadth, and identified the papers of importance to form our hypotheses in the next chapter. We finished this chapter by formulating the research question that will guide this work and intends to cover the gap we recognised. In the end some critical questions were raised to validate it.

[46] For example, in the case of OSS venture-funded startups we believe to have identified all published material of importance. For open source software community, however, this could not be accomplished, as the number of papers out there is simply too high.

Chapter 3

Theory & Model

In this chapter we will transform the research question into multiple hypotheses based on conclusions gathered from our literature review (Chapter 2). In the end, these will be summarised in a visual model (Section 3.2).

3.1 Theory

3.1.1 Research Question

In the previous chapter we formulated the research question as follows.

What changes arise in the dynamics of an open source software community, when its sponsoring startup undergoes an acquisition?

In an attempt to answer this, we split it up in two smaller themes: sponsoring startup acquisition, and OSS community dynamics. In essence, we are studying the open source software community as a system by looking at its dynamics. Concretely, we want to observe changes triggered by a very specific event: the acquisition of the sponsoring startup. The rationale behind this model is that a firm acquisition is an action that can easily be controlled or forced by a small entity, such as an executive team with sufficient capital. Community dynamics on the other hand are aggregate behavioural observations of a complex system[1], consisting of many interdependent agents. These dynamics can not be controlled directly, although they are very determining for the success of an open source project. The goal is to better understand the system by capturing the influence of the acquisition effort on community gestures.

[1]OSS communities or projects have been studied as complex systems in the past (Schweitzer et al., 2014; Maillart et al., 2008; Agnihotri et al., 2012).

3.1.2 Topic: Sponsoring Startup Acquisition

Previous studies have investigated how a sponsor firm influences the OSS community and its shape (West and O'Mahony, 2004; Spaeth et al., 2015). Our main objective is to complement this literature, by offering the different perspective of a deep-dive in a well-defined use case. As a design choice, to focus on sponsoring startup acquisitions, we narrow the scope of this study, but in turn make the take-aways more applicable in practice.

The startups that are in the scope of this study are named 'venture-funded OSS Startups" (Ye, 2008). This is, however, insufficient to fully support our use case. In the literature review, a firm sponsorship refers to a broad set of affiliations a firm may have with a OSS community: a sponsor offers support by infrastructure and salary provision, marketing activities, event/conference organisation, etc. In turn they enjoy commercial opportunities of the software built by the community (Dahlander and Magnusson, 2005; Dahlander, 2007; Grand et al., 2004; Riehle, 2009). We specifically mention that it should be of a "symbiotic" nature (Dahlander and Magnusson, 2005): both the community and the startup should be well aware of and reap benefits from each other. We argument this decision by stating that for the community to be moved by an acquisition event, a logical assumption is that the community cares and maintains interest about the commercial firm.

Looking at Dahlander (2007); West and O'Mahony (2004), we will not make the distinction between firm- and community-initiated projects, as many new communities arise as a mix of both. It is quite typical for an experienced and popular developer (or several) to start an OSS project on their own from which a community is spawned, and simultaneously found a small company to represent the commercial face of that project.[2]

Concerning the business model, the startups typically have a pure OSS business model, rather than a hybrid one. Nonetheless, we do not exclude that latter case: proprietary code may also be developed, as long as the OSS is a required complement to the private software.

In this thesis, we tend to look at "dyads" of communities and startups, both depend on each other to a similar extent. Generally speaking, a dyad represents two entities that share a bi-directional, significant relationship. Thus, we filter out companies that commercialise on multiple distinct OSS communities on one hand (such as *Red Hat*), and big communities on which multiple firms focus and engage in co-opetition (such as *Linux* or *OpenStack*[3]). This will allow to ground our research and mitigate more complex and hidden effects.

We have pinpointed the entrepreneurial exit (acquisition or IPO) as a disruptive event in a startup's history. We decide to drop the IPO case and focus on acquisitions for two main

[2] Although no proof is delivered here, we have read several written testimonies of this phenomenon.
[3] https://www.openstack.org/

reasons:

- To our knowledge only a few OSS IPO's occurred (see Appendix B), compared to a large numbers of acquisitions. This can be partly explained by the public scepticism around the profitability of OSS. However, Waters (2015) predicts a coming wave of IPOs, as numerous OSS firms are reporting rising revenues and indicate plans for such an exit.
- An initial public offering is quite a different process than an acquisition. According to Burst (2014), many startups are not built for profitability (an important criteria for IPO success), but rather for being bought by larger companies.

Additionally, Coyle and Polsky (2013); Selby and Mayer (2013) confirm the acquisition mechanism in high tech industry under the "acqhire" phenomenon, as a human resource strategy. This especially fits the case for OSS, where the technology is free and open, but the challenge lies in obtaining competencies to handle those. Dahlander and Magnusson (2008) explicitly mention that firms need to develop sufficient capacity and competences to assimilate external knowledge. A specialised OSS startup offers the skills, tacit knowledge, and dynamic capabilities of the core development team, which is valued very highly by the acquiring firm.

3.1.3 Hypotheses: OSS Community Dynamics

We have delineated the population and antecedents under which we want to observe OSS community dynamics, above. The goal is to compare the dynamics pre- and post-acquisition.

Santos et al. (2012) highlights different groups of agents in a community: developers, users, and visitors.[4] Developers are programmers that contribute to the project with valid code. Users and visitors can contribute to the project without code, by filing in bug reports, proposing new features, suggesting code insights, indicating installation problems, etc. Even if they don't contribute directly, they aid the diffusion project by adoption and promotion. We will focus on productivity success and entry-exit dynamics (Schweitzer et al., 2014) looking at one distinction: whether the participant makes a technical contribution (developer) or a non-technical one (users and visitors) (Dahlander and O'Mahony, 2011).

New Joiners

An important aspect of an OSS community is its capability to attract new joiners. Whether a considerate participant will end up joining an online community depends on the *initial fit expectations*, the expected degree to which an individual believes that their interests match the community (Butler et al., 2014). These interests can vary strongly between individuals (von Krogh et al., 2012).

[4]Sponsors are also mentioned. As that has been covered in the previous section, we leave it out.

Quite typical for developers are *signaling incentives*, mainly career concern and ego gratification (Lerner and Tirole, 2003). We expect both these incentives to be reinforced by an acquisition, as the community suddenly receives attention from a larger company, bringing in more career opportunities and watching peers. We thus deduce that more developers will join the community.

There is a continuous tension between a community and its commercial dyad in terms of governance and control. Sometimes firms decide to loosen up the development and give more autonomy to the community, which attracts more developers (West and O'Mahony, 2008). We believe that an acquisition increases the odds of this happening, as it brings in a new hierarchy of decision-making (that of the acquiring firm). It is also said that projects with a sponsor, attract greater user interest over time (Stewart et al., 2006). We extrapolate this statement to have effect when the sponsor becomes part of a larger whole.

Dahlander and Magnusson (2005) identify 7 critical managerial challenges for firms when attending an OSS community. One of those is attracting new developers and users. As the acquiring firm typically has more experience with OSS communities, than the acquired startup[5], they can bring in better practices for these managerial challenges.

So, we predict a higher likeliness for prospective developers and users to join an open source project after acquisition. This is furthermore reinforced by the fact that there will be more prospective developers and users, due to the large media coverage that is accompanied by the acquisition. In other words, much more people hear about the OSS project/community, which triggers their curiosity to look into it.

Finally, from *preferential attachment* (Graves, 2013) we expect the acquisition to be an indication of developer growth, which in turn will attract more developers as well.

Hypothesis 1.1: A post-acquisition community will attract more new developers than before.

Hypothesis 1.2: A post-acquisition community will attract more new users and visitors than before.

[5]We notice that most of the acquisitions take place inside the same industry sector. In other words, the acquiring firm usually already has experience handling several large communities. We refer to Appendix A

Exits

According to Sharma et al. (2010), the value fit between a developer and the OSS organisation (consisting of the community and in part also the startup dyad) is important in predicting turnover intention. There is an *essential tension* in the relationship between a community and its sponsor, trading off appropriation of the returns versus providing incentives for adoption (West and O'Mahony, 2004). The new commercial entity prompted by the acquisition, creates a risk for a more closed development process. Developers could react on this with decreased inclination to contribute, as "making their mark" is a critical incentive.

When a firm does not sustain the standards of excellence sought by the software developers, the latter will seek to change institutions that do serve the social practice, e.g. by joining another project or forking (von Krogh et al., 2012). Moreover, if a firm is not perceived open and credible by the developers, they can find a lacking social identification with the firm, which doesn't help for their motivation (Spaeth et al., 2015).

No matter how little organisational change is introduced in the startup upon acquisition, new decisions will be taken that are reflected on the community. We predict that this effects in a loss of developers that are on the edge of the community-sponsor tension. Even without those new control actions, some developers might decide to leave based on reputation grounds; if the acquiring firm is not perceived as one that supports their needs.

The influx of users in the community is highly dependent on project attractiveness (Santos et al., 2012). Building on that, we expect the acquisition to decrease project attractiveness for some users & visitors, which in turn will repel more of these existing agents.

Lastly, if the acquisition introduces changes in governance and/or licensing, there will be time-consuming transition costs and work interference for community participants, which upsets community acceptance (Dahlander and Magnusson, 2008). This may come in the form of sponsor-specific legal terms and ambiguous community leadership (West and O'Mahony, 2004).

Hypothesis 2.1: A post-acquisition community will repel more existing developers than before.

Hypothesis 2.2: A post-acquisition community will repel more existing users and visitors than before.

Project Success

The success of a project can be explained in the context of network social capital. For example, teams with higher internal cohesion, moderate technological diversity, and number of direct and indirect contacts are positively associated with success in terms of productivity (Singh et al., 2011). Having a OSS startup suddenly embedded in a larger firm can create knowledge spillovers, and reinforce the aforementioned social effects.

In terms of preferential attachment (Santos et al., 2012), we hypothesise that an acquisition is a public indication of project attractiveness, which will reinforce itself systematically.

This brings the question of how is OSS success measured? Crowston et al. (2003) presents a summary on this topic. As our scope is on users, visitors and developers, we select a process-based measure: the level of activity, which can be measured as contributions. Technical contributions are additions to the technical core of the project work, in the case of open source that would be working code (Dahlander and O'Mahony, 2011). All other contributions are categorised as non-technical.

Hypothesis 3.1: A post-acquisition community will experience more technical contributions.

Hypothesis 3.2: A post-acquisition community will experience more non-technical contributions.

3.2 Model

Having identified our hypotheses above, we construct a conceptual model to frame our propositions in a less literary way. A visual presentation can be found in Figure 3.1.

Using this model, we reformulate our hypotheses.[6]

$\mathcal{H}_{1.1}$: a_d increases in the post-acquisition period.

$\mathcal{H}_{1.2}$: a_{uv} increases in the post-acquisition period.

$\mathcal{H}_{2.1}$: d_d increases in the post-acquisition period.

$\mathcal{H}_{2.2}$: d_{uv} increases in the post-acquisition period.

[6] a_d is the entry rate of new developers in the community, while a_{uv} is for visitors and users combined. The same goes for $d_{d/uv}$, as the exit rate. CCR represents the technical contribution rate and $nCCR$ the non-technical contribution rate.

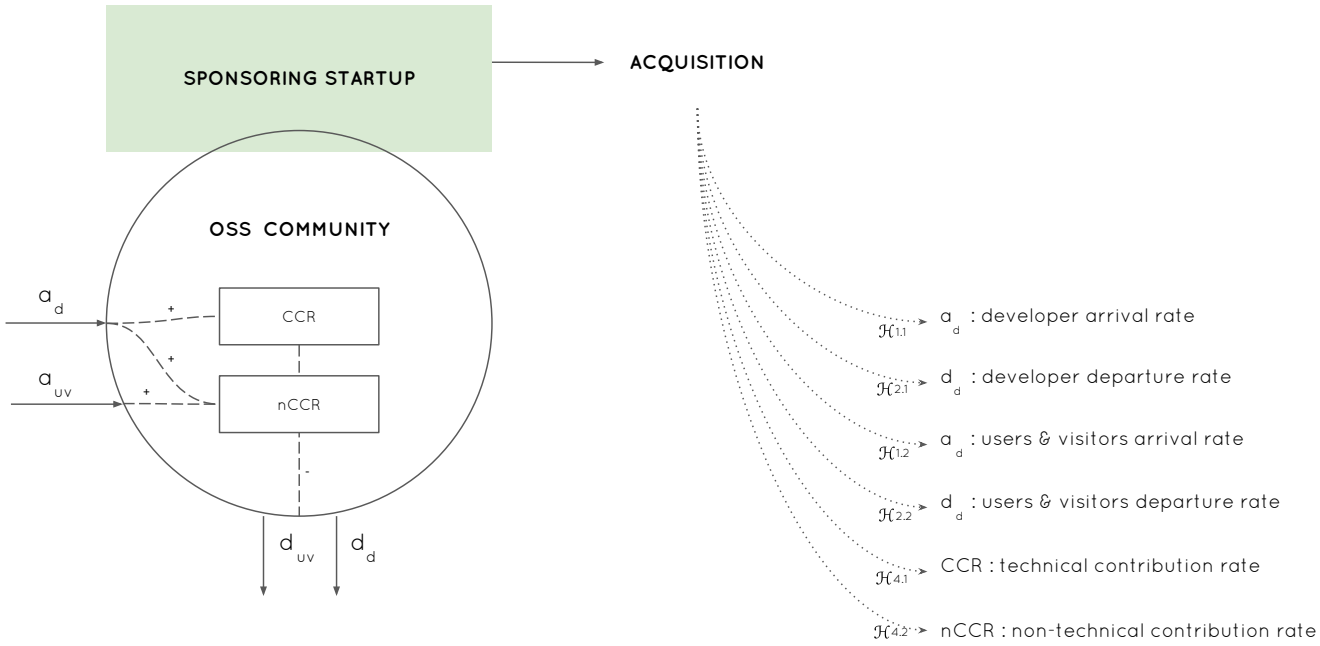


Figure 3.1: Model of community dynamics and sponsoring startup acquisition.

$\mathcal{H}_{3.1}$: CCR increases in the post-acquisition period.

$\mathcal{H}_{3.2}$: $nCCR$ increases in the post-acquisition period.

Chapter 4

Research Methods

In the previous chapters we have explored the literature landscape, defined our research question, and streamlined it into several hypotheses that were summarised in an abstract model. In this chapter we will tackle the research methods used for testing our hypotheses.

4.1 Research Design

So far, a deductive research approach is taken. Now, we want to empirically test the hypotheses that were deducted from our literature exploration. We will perform a statistical quantitative analysis by leveraging the freely available OSS data. Nonetheless, apart from a lack of resource, we have no valid reason to leave out a survey, content analysis, or qualitative approach.[1] OSS communities will be sampled using primarily a snowball and purposive approach. For each of those a longitudinal analysis will be performed.

4.2 Sampling OSS Acquisitions

In order to collect the cases that we want to analyse, we need to define a population. We previously quoted the definition of a "venture-funded OSS startup" (Ye, 2008). It states that the firm must have been established after 2000, have no public offering, not be a subsidiary of a parent company, gather revenues on OSS, and have received financing from one or more independent venture capital (VC) firms. We use the same criteria for our sample. In order to capture sufficiently recent effects, we look at firms established after 2000. Additionally, in order to talk about a startup firm, it should be no older than 11 years at the time of acquisition (Hellmann and Puri, 2002). To ensure original behavior, the community's acquisition virginity must be preserved. In other words, by taking over the public offering and subsidiary criterion, the acquisition will be the community's first one. The VC criterion, characterises the pressure on the business to seek for an exit. Obviously, the startup must collect revenues based on

[1] See propositions for future research in Chapter 6.

OSS offerings.
Additionally, we look at "dyads" of startups and communities (see Chapter 3). We categorise this term with the following conditions:

- When community and firm carry the same name, e.g. Ansible.
  [Sufficient], as trademarks are shared and by using that name the firm & community are named under one word.
- When the community leader or founder, has a crucial position (CTO/CEO) inside the startup.
  [Sufficient], as every decision will be taken in the eyes of the firm, as well as the community.
- If the website of the community is hosted on the same domain as that of the sponsor.
  [Sufficient], as the startup manages (part of) the community infrastructure.
- If the community website shares a link to its commercial dyad and vice versa.
  [Required], as both must at least publicly recognise the existence of each other.
- If the business model of the OSS startup is highly dependent on the open source software project (so-called "pure" strategy). And the development of the software depends highly on human resources of the startup.
  [Sufficient], both can hardly survive without the other.
- The commercial startup must focus its offering on the community dyad.
  [Required], definitely a must-have.
- The OSS community's must have a single principal sponsor, which is the commercial startup dyad.
  [Required], as we want to look at community-firm pairs and not co-opetitions such as `OpenStack`.

4.2.1 CrunchBase Population

To start collecting elements of our population we look at acquisitions in the open source software space. In an attempt to be as exhaustive as possible, we took a two-phased approach. Most of the sample collection is done using `CrunchBase`[2], which is a public database from `TechCrunch`[3] one of the major startup blogs. It mainly gathers funding and exit information from global startups in general. It provides an almost complete overview of the recent IT and Internet Industry in the U.S. and has been used previously as a academic data source (Ter Wal et al., 2016; Alexy et al., 2012; Block and Sandner, 2009).

[2] www.crunchbase.com
[3] www.techcrunch.com

First we snowballed cases on `CrunchBase`, based on three web articles[4], by looking at acquisition connections. This resulted in 71 acquisitions of which 37 were identified as being part of our population.

In a second phase, to test the exhaustiveness of this set, we used a `CrunchBase` data export.[5] It had a total of 101,053 funded companies of which 117 were categorised as "open source".[6] It accounted 22,849 acquisitions, from which 57 had an acquiree categorised as "open source". Now, 35 of those 57 acquisitions were not covered by our set from phase one. Of those 35, 12 additional cases belonged to our population. So the second phase, increased our population sample by about 30%. A list of all acquisitions can be found in Appendix A.

In Figure 4.1 we can see that the OSS acquisition is a growing trend, especially in 2014 and 2015. Also, we notice that the population, starting from 2005, shows no relevant visual time-bias, vis-a-vis all OSS acquisitions.

4.2.2 GitHub Sample

Much of the past open source software research is based on publicly available data from `SourceForge`. However, since 2011, `GitHub` has taken over as the market leader of on-line repository hosting[7] and since then many papers switched to that as the platform to mine OSS data (Thung et al., 2013; Tsay et al., 2014; Marlow et al., 2013). In this study we will take the same approach to investigate the community. In turn, this fixates a purposive sample of our population. We bound it with the following conditions:

- The community's development must be hosted on `GitHub` before, at, and after the time of acquisition, so that we are able to extract the data. We don't select mirror projects, as they don't contain non-technical contributions.
- The total `GitHub` activity should show more than 1000 commits and 15 contributors, for it to be considered a community. We know that more than 90% of `GitHub` repositories have less than 50 commits (Kalliamvakou et al., 2015). Assuming a venture-funded startup will have at least 5 employees, we believe 15 participants to be a realistic lower bound to consider the residual group a community.
- We look at acquisitions taking place during or after 2008, which is the founding year of `GitHub`. This is a necessary condition if non-technical contributions on the `GitHub`

[4](Schireson, 2016; Tunguz, 2015)
http://gemsres.com/story/apr06/209227/linux-venture-capital-fig3.gif

[5]Academic license: https://data.crunchbase.com/

[6]On the `CrunchBase` website there are 752 search results for the category "open sourceÂ´Â´. The discrepancy between that number and what we found in the data export comes from the fact that the export only includes companies that have been funded. As it is one of our conditions that the startup must be venture-funded, we drop these search results. Most of them offer small complementary services to larger established communities.

[7]https://en.wikipedia.org/wiki/GitHubHistory

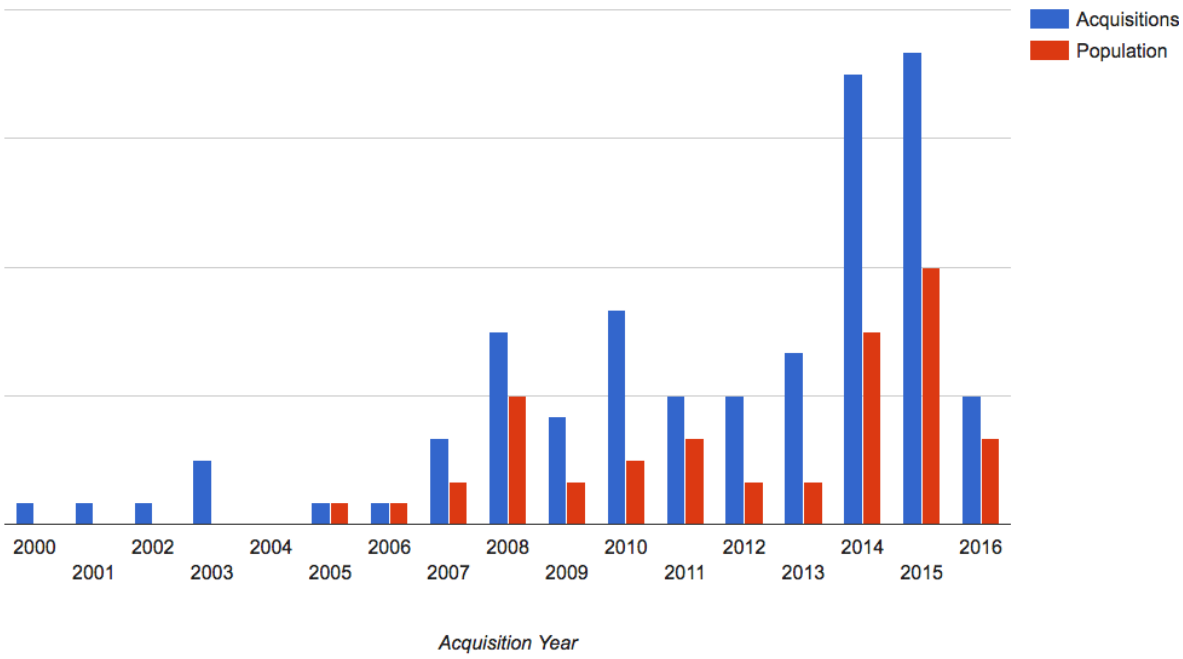


Figure 4.1: Timeline of OSS acquisitions and the population. The number for the year 2016 is not trend-representative, because at the time of collection we were only in the month of April of 2016, and therefor could obviously not capture all deals of that year.

infrastructure should be present. As a cut off date, we take only acquisitions that take place before 2016 - else we would not be able to gather sufficient data due to its newness.

To investigate these conditions we researched our population sample on-line.[8] We have identified a total of 16 dyads (a close pair of acquired startup and OSS community) as our `GitHub` sample.[9] Their time distribution is presented in Figure 4.2.

As `GitHub` is a relatively new on-line repository host, the conditions we laid on the sample will probably bias our representation of the population towards recent acquisitions.[10] Despite this statistical non-probabilistic bias, gathering young cases will make our findings more contemporary and relevant for practice.

Finally, we have summarised our sampling approach in Figure 4.3.

[8] Mainly looking at `Wikipedia`, the websites of the companies/communities, and studying the `GitHub` repositories for each project if given.

[9] In contrast to a population sample of 49.

[10] Although `GitHub` was only founded in 2008. It is possible that it hosts git repositories with historic data from before. Nonetheless, the "hot" project repository used to be `SourceForge`, thus we expect some older projects to be remaining there.

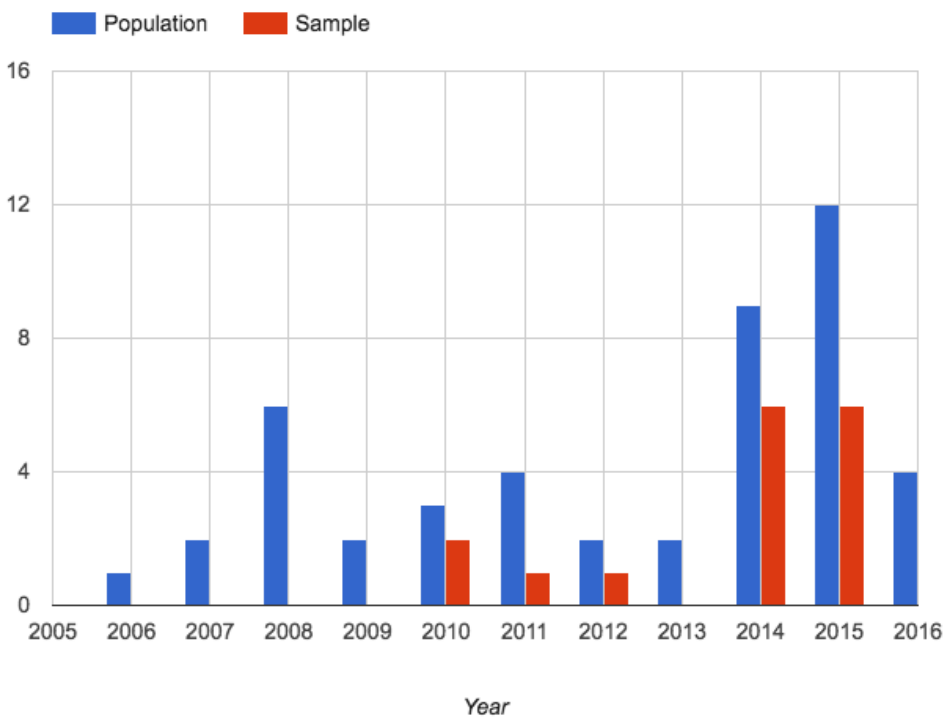


Figure 4.2: Timeline of our purposive GitHub sample of dyads.

4.2.3 Sample Description

In Appendix C, we summarise general findings of the final `GitHub` sample. Here we can see, that the startup does not always carry the same name as the community. On average these venture-funded OSS startups get acquired at an early age: 3 years on average, varying from 11 to less than 1. We find 7 permissive, 6 restrictive, 3 corporate, and 1 moderately restrictive licenses. Unfortunately, the acquisition price is often not disclosed. From what we have seen it usually dangles in double- and triple digit million dollars. We see that in 7 cases the OSS product gets renamed, re-branded, or swallowed by the acquirers portfolio. In the remaining 7 the product stays pretty much untouched.

Not to our surprise, the most popular business model is services (hard soft). Second come in dual-licensors and functionality up-graders. Finally, we note that “Infrastructure” is the most popular IT sector, where the consolidation takes place.

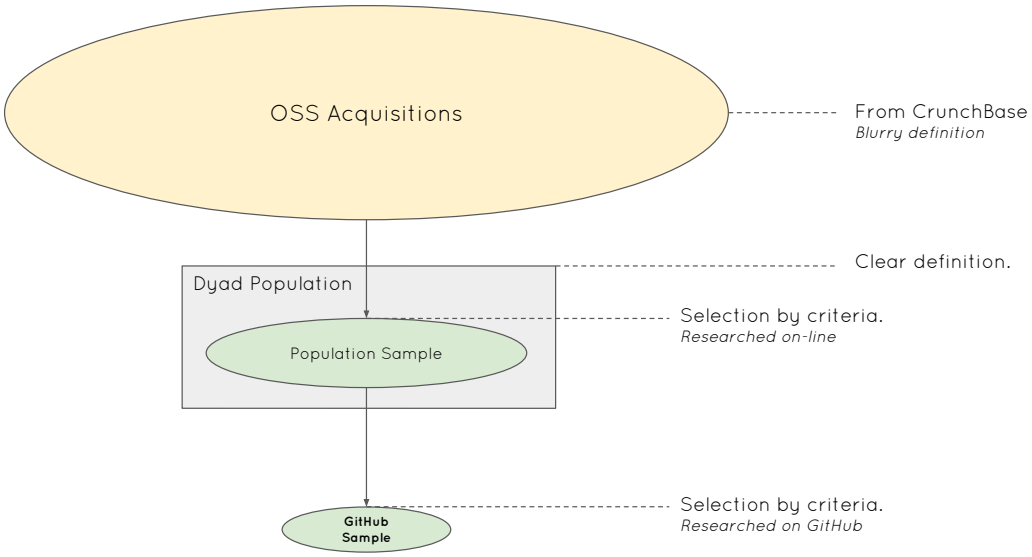


Figure 4.3: Timeline of our purposive GitHub sample of dyads.

4.3 Data Collection

4.3.1 GitHub

`GitHub` is a SaaS[11] repository host based on the `git`[12] technology. It provides a user interface that visualises and adds a layer of functionality on the common `git` features, such as commit, branch, master, fork, pull request,... Moreover, it offers the possibility to manage issues with comments, which can serve as bug reports, feature requests, or any general discussion. Finally, it gives a user the ability to manage their profile and connect with other users, making the entire platform into a sort of social media site for developers. Dabbish et al. (2012) value the platform as a micro supply chain between consumer and producer, all the while providing visible signals of attention and communication to overcome the downsides of transparency on a large-scale network.

One of the strengths of the platform is the pull request. It is a mechanism, that allows outsiders to merge contributions, that they made in their own fork of a project, back to the master. It permits easy inclusion of code coming from programmers without `git`-access. The number of pull requests and failure rates typically grow as the project increases in size (Rahman and Roy, 2014).

[11] Software as a service
[12] https://en.wikipedia.org/wiki/Git_(software)

To fully understand the reasoning behind the coming technical decisions, we suggest the reader that a good understanding of `git` and `GitHub` will help.[13]

Projects on GitHub

Software content and meta data in `GitHub` is stored under repositories. These repositories can be either public or private. Private repositories, whose creation require a paid subscription, are fully hidden for non-permitted users and therefor omitted in this study, as they no longer can be considered as open source. Repositories are grouped under either a user account or an organisation, which can have users registered as members. Popular projects with large communities are usually stored under an organisation, which handles several repositories for different integrations or modules around the core software. To analyse the community we will look at these several repositories of the organisation, unless for large foundations, such as `Eclipse` that places the `BIRT` community under a single repository. For any organisation we distinguish between source repositories and forks. The latter is ignored, as they represent copies from external development.

Contributions in GitHub

We select 5 kind of functions in `GitHub` that sustain the development process: commit, pull request, issue, comment and merge.[14] Although the intention of the user actions behind these features seems rather straightforward, we notice some workarounds and inconsistencies (partly reported by Kalliamvakou et al. (2015)). For that reason we shall re-map these functions to 5 contributions, which better reflect the nature of the contributions: Code Contribution, Code Review, Pull Request, Issue Creation, and Comment. We discuss the proceedings and summarise it in Table 4.1.

Code Contribution: In essence, a commit is an action where a user writes code and uploads it to the repository. However, there are cases where this no longer holds.

- For example, when a user merges a `GitHub` pull request, it is registered as a separate commit in `GitHub` without any programming occurring. Instead, the code is verified and included by someone who has permissions to do this.
- Another approach is the so-called branch merge. This functions similarly to the pull request, but the branch merge takes place inside the current repository, instead of an

[13] Some documentation:
http://www.instructables.com/id/Introduction-to-GitHub/
http://ftp.newartisans.com/pub/git.from.bottom.up.pdf

[14] We don't discuss forking, although it can account for project popularity, not every fork can be considered as a contribution. In the end those who want to contribute, will get noticed by the submission of the pull request.

external one, without a request (second user). In most cases it represents an organisational restructuring in the project by one of the leaders. The actual commits will then be included in the master branch after the merge. For those reasons we will ignore this event, as we feel it doesn't support our hypotheses.[15]

These two kind of commits are rare and not considered as *Code Contributions*.[16] For normal commits, we count a code contribution in the name of the author at the time stamp given by the `git` author header.[17]

Code Review: Code review is an action where no code is created, but code is accepted and merged into the master branch of the repository. This can be done in a variety of ways. A commit representing the merge of a pull request is the most known example. It is an important task of the community leaders. Although no content has been created, significant effort needs to be allocated in reviewing the code, which is quite similar to the activity of coding itself.

Quite often, we notice that the committer and author of a commit are not the same person. This occurs if someone authors some content to the index, which then gets committed by another `git` account. That commit is also a *Code Review* action, because we expect developers to not publicly commit code in their name without reviewing it. In this case, next to the *Code Contribution* count for the author, we also register a *Code Review* event by the committer. The differing time stamps in the `git` headers are then assigned to each action accordingly.

Pull Request: *"GitHub made the Fork & Pull development model popular, but pull requests are not unique to GitHub; In fact, git includes the git-request-pull utility, which provides the same functionality at the command line. GitHub and other code hosting sites improved this process significantly by integrating code reviews, discussions and issues, thus effectively lowering the entry barrier for casual contributions. Combined, forking and pull requests create a new development model, where changes are pushed to the project maintainers and go through code review by the community before being integrated."* (Kalliamvakou et al., 2015).

`GitHub` treats a pull request as an issue with added functionality for commit inclusion and merging for project maintainers. When a developer opens up a pull request, we will register this as a *Pull Request* action. In itself it is not a Code Contribution (that is reflected by the commits embedded within), but rather a motion from an external developer to include and defend their code.

[15] Even if it does, this does not occur frequently, so the impact won't be too large.

[16] They are identified by a regular expression in the Title.

[17] Every commit has a committer and an author assigned to it. To identify the user that wrote the code, we look at the name of the author. Mostly these fields are the same. However, sometimes they differ, in that case we create a "Code Review" action in the name of that committer.

Issue Creation: When any community participant opens up an issue that is not a pull request, anything from technical to organizational topics could be discussed. Thus, a new normal issue is seen recorded as a *Issue Creation* action.

Comment: All `GitHub` users have the possibility to give comments to either commits, issues, or pull requests of any public repository. This will be recorded as a *Comment* action.

The contributions are re-mapped to the definition of Dahlander and O'Mahony (2011). Code Contribution and Code Review are classified as technical contributions, and Pull Request, Issue Creation, and Comment as non-technical contributions. According to Santos et al. (2012), we define developers as contributors that at least once perform a Code Contribution or Code Review, all other participants belong to the group of users & visitors.

Developer Profiles in GitHub

Each user account on `GitHub` has a public profile where the number of followers, followings and favorite (starred) repositories are displayed and any associations to `GitHub` organisations. Additionally, the user has the possibility to display other data such as name, email address, company, profile picture, and location.

When `GitHub` interprets commits from `git` it reads the committer and author e-mail address and tries to link it to an active user on their platform. When such a hit occurs the commit will be linked to the profile of that user. If not, the name from the `git` commit is taken without a profile hyperlink. Unfortunately, the latter situation often occurs, when developers do `git` operations under a `git` account whose e-mail address is not linked to their `GitHub` account. The problem with this is that for a part of the commits no profile analysis can be concluded for those authors or committers, in order to identify their employer.

We have identified some heuristics that allow us to link a `git` commit to a `GitHub` profile, when `GitHub` fails at doing so:

GitHub Functions			Technical Contributions		Non-technical Contributions		
			Code Contribution	**Code Review**	**Pull Request**	**Issue Creation**	**Comment**
Commit	Merge of Pull Request			x			
	Branch Merge						
	Normal	Committer = Author	x				
		Committer != Author	x	x			
Issue	Pull Request				x		
	Normal Issue					x	
Comment							x
					Users & Visitors		
			Developers				

Table 4.1: Mapping of `GitHub` functions to Contributions.

- An essential piece of data from the `git` commit that `GitHub` does not use for profile linkage, is the author/committer name. We use the name to do a global search for users on `GitHub`.[18] If the results deliver 20 hits or less and a single one of those users has forked the focus repository, we find it safe to assume that is the profile we are looking for.
- Another approach is to look through the list of organisation members and repository contributors, and compare the `GitHub` full name and account name with the `git` commit name by removing spaces and converting upper-case characters to lower-case. A single hit in this heuristic, allows us to link the commit with an active profile.
- The third heuristic works as follows: some commits have a pull request number attached to it. When the commit has a unspecified committer/author, but is part of a pull request. Then it is safe to assume that the creator of that pull request is the person who wrote that commit. We only apply this heuristic, when author and committer have the same email address.

These heuristics might give errors, but based on manual observations we believe that the benefits outweigh the costs. Another option would be to manually assess email-profile links, but this work is too cumbersome. Even if fully successful, this will not solve all cases, as some internal developers working with the `git` repository might not have a `GitHub` account (e.g. the `Mono` project).

Validation

To defend our use of `GitHub` to source our research data, we discuss the perils given by Kalliamvakou et al. (2015):

- *A repository is not necessarily a project.*
  We only look at OSS communities with active projects.
- *Most projects have very few commits.*
  We have set a minimal amount of required commits to define our population.
- *Most projects are inactive.*
  We look at projects who were active at the time of acquisition.
- *A large portion of the projects are not for software development.*
  Our population targets OSS projects.
- *Two thirds of projects are personal.*
  Our population must have a minimum number of multiple contributors.

[18] This only makes sense if the name is somewhat longer than 4 characters. Often, the 4 or less character names are called "root" as a system default or deliver too many search results. For example, "Yash" hits 3,003 results at the time of writing.

- *Only a fraction of projects use pull requests. And of those that use them, their use is very skewed.* The `Mono` project for example does not work with pull requests, but gives contribute access directly to the remote `git` repository. This is not a problem as we are doing a longitudinal study for each repository, instead of a cross-sectional one where repositories are compared to each other.
- *If the commits in a pull-request are reworked (in response to comments) `GitHub` records only the commits that are the result of the peer-review, not the original commits.* As suggested in the paper, we will only look at the commits of the master and not the commits of the pull request.
- *There are three different ways to merge pull requests: so sometimes pull request are actually merged, although they appear not to be.* We will not look at the status of the pull requests. The three different ways of merging are identified and registered in our data extraction.
- *Many active projects do not conduct all their software development in `GitHub`: some repositories are simply mirrors, many commits are done by non-`GitHub` users, and issue-tracking is sometimes externalised.* We only look at projects that are actually developed on `GitHub`, not mirrors. Heuristics will be coded to identify the profile of the developer. Mailing-lists, `Google` groups, and forums are often used in parallel to guide development. So, part of the non-coding contributions will not be captured for some projects. However, since we perform a longitudinal study, it does not have to be an obstacle to detect differences over time.

4.3.2 Python

Research papers employ several channels for collecting `GitHub` data. The platform releases all activity through its application programming interface (API) service[19], allowing any user account to fetch an event timeline or detailed repository information via HTTP requests. The `GitHubArchive`[20] collects and stores all events[21], as the number of past events that can be fetched through the API are limited to 90 days. It also provides the possibility to query them on `Google BigQuery`. We discard this database as the event payloads do not offer the level of detail needed. This led us to `PyGitHub`[22] which offers all API functionality as a `Python` library. Due to feasibility reasons we chose that approach.

There exists also `GHTorrent` (Gousios and Spinellis, 2012), which mirrors all the data that can be requested through the `GitHub` API from a server. Access permission needs to be

[19] The `GitHub` API v3
https://developer.github.com/v3/

[20] https://www.githubarchive.org/

[21] https://developer.github.com/v3/activity/events/

[22] http://pygithub.github.io/PyGithub/v1/index.html

requested to the research group, and performance issues might be a problem. Nonetheless, we recommend looking into this option for future studies, as we expect the `MongoDB` database to provide a more user-friendly data collection experience without loss of quality. The `git-GitHub`-profile heuristics we leverage through `PyGitHub`, however, would not be possible with solely this approach.

Our extended script is available as its own repository on `GitHub`, more details can be found in Appendix E.

4.4 Descriptive Data Analysis & Transformation

So far, we have a collection of contributions as represented in Table 4.2. The *User Account* normally represents the user name of the developer on `GitHub`, but if no such profile was found the `git` e-mail address is used. The last field, *Other*, is used to store supplementary information such as the number of additions or deletions of a code contribution.

Gathering all the `GitHub` activity of our samples, we get a total of 1,363,933 contributions over 11 years by 25,518 contributors. About three quarters (%72) of these contributions are technical, and half (47%) of all community participants are developers. This ratio, however, varies a lot per community, as some repositories decide to use `GitHub` to guide their non-technical interactions and others host it elsewhere. The different sample projects vary quite a lot: e.g. in size (μ=85,264; σ=149,328), and developer concentration (min=12%; max=95%). More details can be found in Appendix D.

In order to set up the data for hypothesis testing, we need to transform it to a format where we can extract the community's (non-)technical contributions, and contributor-type entry-exit dynamics. Firstly, we use a productivity time window in which we aggregate contributions (Adams et al., 2009). Scholtes et al. (2015) selects 7 days as ideal to average out weekend-workday periodicity. Thus, we calculate the number of technical and non-technical contributions in consecutive non-overlapping time windows based on ISO week dates.[23]

For productivity measure we use a contribution event count (number of commits/issues/...), as described previously. (Scholtes et al., 2015) argues that including additions and deletions,

[23]The ISO week date is ideal for programming and scripting the data. https://en.wikipedia.org/wiki/ISO_week_date

Time Stamp	Contribution Type	User Account	Organisation/Repository	Other
12h45m12s 06/12/2009	code contribution	ltorvalds	ansible/ansible	12 add; 6 del

Table 4.2: Information header of contribution (incl. example).

give a more realistic result. However, we decided not to include this, as we want to keep a unified approach. Treated as events, a commit can be compared to an issue (which doesn't hold if we look at the text-vs-code content). We justify this decision by stating that this study interests itself in the intent behind a contribution and not necessarily its efficiency.

For each participant we know in what week their first contribution takes place, which in turn allows us to easily compute the number of contributor entries (new joiners) per week. For exits (contributor departure) it is a bit more difficult. In principal, contributors, who seem to have left the community, can always return and make new contributions in the future. We, thus, need a measurement to predict whether or not a community participant has actually left the community and will not return. For this we compute - similar to Scholtes et al. (2015) - the cumulative density function of the aggregate of all participant's inter-contribution time. From Figure 4.6 we notice that after 19 weeks of idle time, there's only a 5% chance that the participant will recontribute in the future. We thus take this value and use it to compute departures.[24]

The acquisition event will be defined by a date. In principal, an acquisition has several milestones and phases: public announcement, signing the contract, the transfer of control, integration, completion, etc. (Very and Schweiger, 2001). In this study we will use the date of public announcement for two reasons: information about the other milestones often remain undisclosed, especially for smaller acquisitions of start-ups, and we expect the public announcement, with its instant accompanying media coverage, to have the largest influence on the agents that underlie our hypotheses. Additionally, we usually read that acquisitions were fully completed not more than one month after announcement.

For our observations we will look at a symmetrical interval around the acquisition announcement. By cutting the data one year (52 weeks) before and after the acquisition, seasonality effects will be evened out. This time frame will be long enough to observe delayed effects, such as new management/portfolio decisions from the acquirer, or rolling-off an old project before becoming a new joiner of the project in scope. Additionally, we drop the time window of the week in which the acquisition is announced, as pre- and post-acquisition effects are mixed during this interval. By doing this only a few days are left out, which is negligible compared to the full observation period. This reduces our data set down to 324,150 contributions - although the other contributions are needed to compute entry-exit dynamics (Appendix D). So, we now have a bundle of weekly rates[25], consisting of the following:

[24] This gives the requirement for 19 additional weeks, measured beyond the observation interval, to properly compute all departure times.

[25] For the acquisitions that take place in 2015, we were not able to gather a full observation period of 2 times 52 weeks, as the event was too recent. In these cases we reduced the symmetric observation period to the highest value possible.

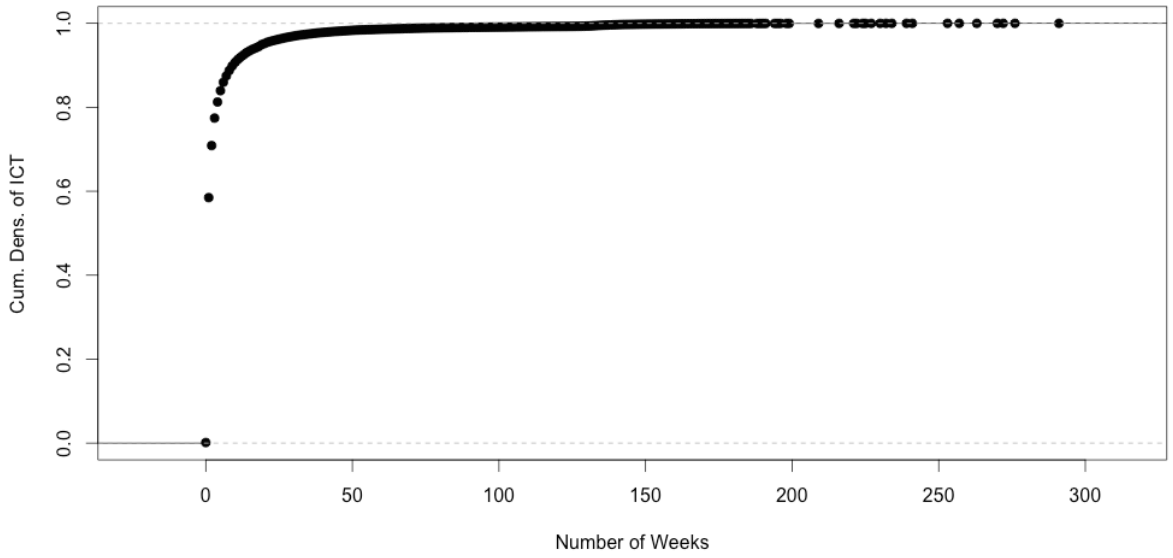


Figure 4.4: Full

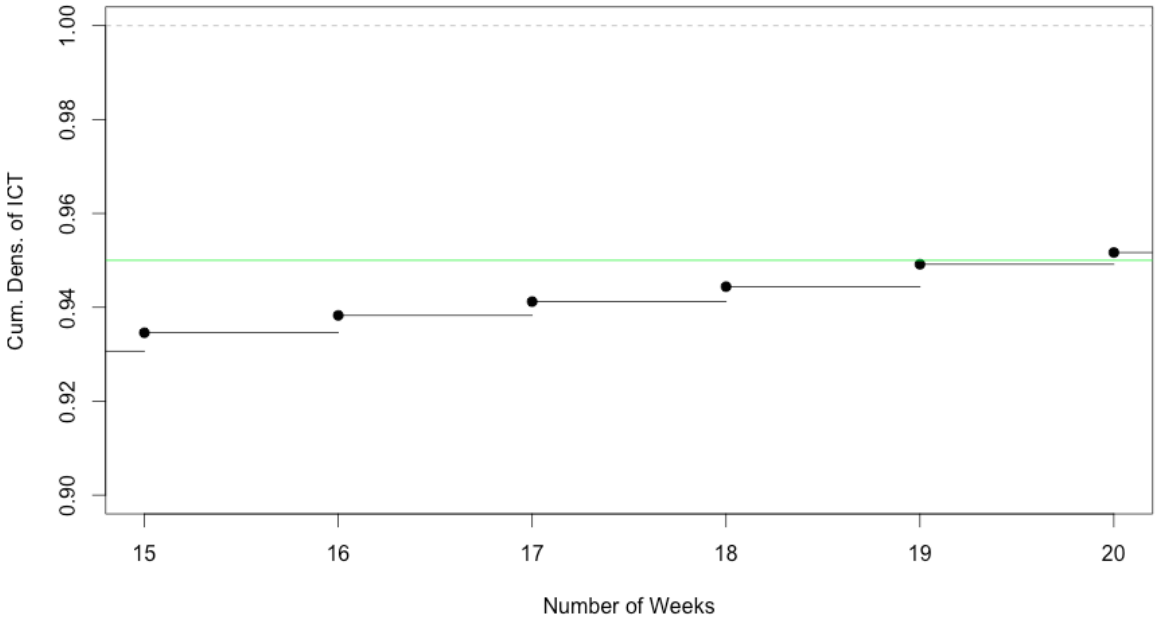


Figure 4.5: Zoomed

Figure 4.6: Cumulative density function of inter-contribution times (in weeks).

- Description: Week Number *(YYYYWW)*
- Dummy Categorical Variable: pre- or post-acquisition
- Numeric Variable: Nr. of Contributions
- Numeric Variable: Nr. of Technical Contributions
- Numeric Variable: Nr. of Non-technical Contributions
- Numeric Variable: Nr. of Entries
- Numeric Variable: Nr. of Developer Entries
- Numeric Variable: Nr. of Users & Visitors Entries
- Numeric Variable: Nr. of Exits
- Numeric Variable: Nr. of Developer Exits
- Numeric Variable: Nr. of Users & Visitors Exits

4.5 Methodology Reflections

Now that the methodology is laid out, we defend the 4 critical questions about the quality of research from Shipman (1988):

1. **Reliability**: *If the investigation had been carried out again by different researchers using the same methods, would the same results have been obtained?*
   If this would to be done tomorrow, yes we believe little variation with our results would be detected. As mentioned before, we are not too sensitive to the non-probabilistic bias of the sample as it is quite exhaustive. It might be, however, that the same study, in 10 years can lead to quite different results, as a new population of firms or communities might take shape towards the future.

2. **Validity**: *Does the evidence reflect the reality under investigation? Has the researcher found out what he/she claims it's about?*
   This is the advantage about open source. If one is trying to see how many people are developing or using the software, all the information is available in the public domain. However, we don't capture all of the community activity, such as mailing lists, forums, website analytics, and non-official communication (e.g. two developers discussing in person at a conference). This problem is partly mitigated by the fact that we are performing a longitudinal study and not cross-comparing organisations directly.

3. **Generalisability**: *What relevance do the results have beyond the situation investigated?*
   Over time, we do believe our results to be generalisable for future OSS startup acquisitions. We would not be surprised that in some cases it even concerns the same developers

and/or users, jumping from one project to the other before they get acquired or joining one acquired project after the other to refresh their exposure. In terms of mechanism, the question to ask is whether an IPO or acquisition of non-startup firms will show the same effects. We do not predict big differences, as the logic used in deriving our hypotheses, could easily be reiterated for those cases.

4. **Credibility**: *Is there sufficient detail on the way the evidence was produced for the credibility of the research to be assessed?*
   Due to the quantitative and factual characteristic of our research design, and the fact that we strongly utilise the depth provided by the `GitHub` API, we confirm the credibility of this work.

4.6 Conclusion

In this chapter we explained the path of how we got to our final sample of projects. We then showed how we gathered the data and described how it was transformed to its final shape. The next step is to observe this data with statistical analysis in order to support our hypotheses. This will be explored in the next chapter.

Chapter 5

Results & Discussion

In this chapter we put our hypotheses to the test by statistically analysing our collected data. Consequently, we discuss the results and findings in a theoretical context, by leveraging the descriptive data of our sample.

5.1 Data Analysis

In Chapter 4 we gathered measurements of community dynamics in weekly rates. The use of longitudinal, rather than cross-sectional data, allows us to draw inferences about the causal relationships we hypothesise, which is not new in studying OSS projects (Roberts et al., 2006; Chengalur-Smith et al., 2010; Ransbotham and Kane, 2011). Based on the need to compare the rates before and after the acquisition, we compute one-way analyses of variance (ANOVA) (Earley, 1991; Shalley, 1995; Bansal and Clelland, 2004). As independent variable the categorical variable, that describes the acquisition precedence of the measurement (pre- or post-acquisition), is selected. In essence, means are compared between two groups: weekly rates pre- and post-acquisition. As dependent variables we will use the total amount, the number of technical, and the number of non-technical contributions per week. Moreover, we also look into the weekly entry and exit rates for all contributors, developers, and users & visitors.

5.1.1 Contributions

Table 5.1 summarises the ANOVA results of the contributions. For all contributions combined, 12 out of the 16 communities show significant differences. However, these differences seem to go in different directions: 8 projects experience a decreased contribution rate after acquisition ($\bar{\Delta}$=-34.4%, σ_{Δ}=10.3%)[1], while 4 of them increase ($\bar{\Delta}$=72.8%, σ_{Δ}=54.7%). This might imply mixed effects, so we break down the contributions in technical and non-technical

[1] $\bar{\Delta}$ represents the mean of the differences expressed in percentages.
σ_{Δ} represents the standard deviation.

(O'Mahony and Ferraro, 2007). With this decision two projects are dropped (`FuseSource` and `RabbitMQ`), as they show only a tiny bit of non-technical activity after the acquisition and, thus, don't permit longitudinal data analysis.

For the technical contributions we notice a clear post-acquisition decrease (significant for 10 out of 14 projects, $\bar{\Delta}$=-38.8%, σ_{Δ}=20.0%), which offers strong support to reject the null hypothesis (i.e. that no effect takes place), but in the opposite direction of *Hypothesis 3.1*. As an exception however, `NuCivic` shows a moderately significant increase.

The non-technical contributions show diverging behaviours. 11 out of 14 project show significant effects, where 4 projects decrease in activity ($\bar{\Delta}$=-32.8%, σ_{Δ}=6.4%), and 8 projects increase ($\bar{\Delta}$=114.8%, σ_{Δ}=93.0%[2]). This offers partial support for *Hypothesis 3.2*, although for some projects it has an adverse effect.

	All Contributions					**Technical Contributions**					**Non-technical Contributions**				
Community	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value
Tutumcloud	89.3	40.1	-55%	17.19	.000***	42.9	11.1	-74%	22.07	.000***	46.43	29.04	-37%	8.52	.006†
Ansible	925.6	1017.1	10%	1.44	.236	211.7	172.6	-18%	4.48	.039*	713.96	844.54	18%	4.27	.044*
Spree	143.7	95.9	-33%	6.91	.011*	28.8	25.9	-10%	0.29	.592	114.94	70.00	-39%	11.22	.001**
Sequenceiq	176.2	160.2	-9%	0.79	.327	113.9	68.8	-40%	18.27	.000***	64.32	95.11	48%	13.84	.000***
Thinkaurelius	53.2	32.7	-38%	12.37	.001***	27.4	8.8	-68%	31.13	.000***	25.76	23.96	-7%	0.07	.643
Feedhenry	36.8	34.4	-7%	0.20	.642	28.2	15.2	-46%	12.27	.001**	6.48	17.19	165%	24.03	.000***
Elgg	226.0	158.9	-30%	3.94	.050†	225.7	150.8	-33%	4.88	.029*	0.27	8.13	2921%	36.85	.000***
Wiredtiger	129.9	138.8	7%	0.69	.408	61.7	70.8	15%	2.73	.102	68.19	67.98	0%	0.00	.975
Ceph	537.4	712.4	33%	21.56	.000***	431.1	382.4	-11%	4.77	.031*	106.31	330.06	210%	74.35	.000***
Eucalyptus	236.0	160.4	-32%	20.60	.000***	178.6	120.1	-33%	22.16	.000***	57.42	40.29	-30%	8.59	.004*
Gluster	75.2	45.2	-40%	15.34	.000***	74.8	44.5	-40%	15.70	.000***	0.31	0.62	100%	1.02	.316
Fusesource	103.6	80.9	-22%	3.98	.049*	-	-	-	-		-	-	-	-	
NuCivic	52.9	111.5	111%	32.35	.000***	31.9	44.8	40%	8.15	.005†	21.00	66.70	218%	45.65	.000***
Pandas	525.3	394.0	-25%	26.81	.000***	50.3	37.7	-25%	9.63	.002*	475.00	356.29	-25%	27.67	.000***
Pentaho	294.6	350.9	19%	5.58	.020*	159.3	175.0	10%	1.00	.318	135.29	175.92	30%	11.68	.001**
Rabbitmq	69.7	159.0	128%	80.86	.000***	-	-	-	-		-	-	-	-	

Table 5.1: Results of ANOVA for Contributions (p < .1 † ; p < .05 * ; p < .01 ** ; p < .001 ***).

5.1.2 Contributors

For analysing the entry-exit dynamics of the community we perform ANOVA on the weekly rates at which developers and users & visitors, join and leave the community.

Entries

Table 5.2 summarises the results. Looking at all entries half of the projects show significant differences in the means before and after acquisition. Of those, all except one (`Spree`) show an increase in entry rates ($\bar{\Delta}$=201.6%, σ_{Δ}=312.3%). We further delve into the observations, by making a distinction between developers, and users & visitors.[3]

[2] We excluded the 2921% outlier.

[3] Again, two projects are dropped in this case, as they lack the pre-acquisition activity to feed the analysis.

For the developers, 4 projects got significant results of which 3 showed an increase (`Elgg`, `Ceph`, and `Pandas`, $\bar{\Delta}$=176%, σ_{Δ}=196%) and one a decrease (`Ansible`). Here, we fail short in proof to support *Hypothesis 1.1.* In this longitudinal analysis, data might fail short in size to statistically leverage the effect of the acquisition. The 4 significant projects are the largest in terms of *velocity.*[4] Some projects carry a lot of their community traffic on `GitHub` (e.g. `Ansible`), while others not as much: e.g. around the time of acquisition `RabbitMQ` reflects `git` development on `GitHub`, but carried issues and discussions on mailing lists and other tools. Nonetheless, we notice that on average, less than half of the contributors are developers, and for `Ansible` this ratio goes down to 25%, so the mean developer entry rates are usually quite lower than for users & visitors. We believe these are reasons why we fail short to support *Hypothesis 1.1.*

For the users & visitors, there are 7 (out of 14) projects with statistically significant results, and all except one (`Spree`), indicate an increase in the number of entries after acquisition ($\bar{\Delta}$=145.6%, σ_{Δ}=145.1%). This provides partial support for *Hypothesis 1.2.* Once more we see low average values for the entry rates in Table 5.2, which can explain the lack of significance in half of the projects.

	All Entries					**Developer Entries**					**Users & Visitors Entries**				
Community	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value
Tutumcloud	8.13	8.09	-1%	0.00	.971	1.65	1.26	-24%	0.67	.417	6.48	6.83	5%	0.11	.742
Ansible	72.25	81.21	12%	3.06	.086†	14.14	10.11	-29%	14.90	.000***	58.11	71.11	22%	7.45	.009**
Spree	8.56	5.41	-37%	13.29	.001**	1.94	1.44	-26%	1.77	.188	6.63	3.97	-40%	12.72	.001**
Sequenceiq	2.37	3.55	50%	7.31	.007**	0.57	0.68	19%	0.40	.531	1.86	2.92	57%	9.07	.004**
Thinkaurelius	2.75	2.55	-7%	0.00	.566	0.27	0.25	-7%	0.04	.846	2.46	2.50	2%	0.01	.904
Feedhenry	0.39	0.90	132%	6.20	.013*	0.27	0.42	57%	1.00	.320	0.10	0.42	340%	8.53	.004**
Elgg	0.08	0.77	900%	21.57	.000***	0.08	0.38	400%	9.66	.002**	0.00	0.38	++	14.91	.000***
Wiredtiger	0.35	0.38	11%	0.09	.761	0.08	0.13	75%	0.75	.388	0.27	0.25	-7%	0.04	.852
Ceph	3.17	7.04	122%	47.78	.000***	2.62	5.02	92%	24.05	.000***	0.56	2.02	262%	30.16	.000***
Eucalyptus	0.58	0.67	17%	0.36	.550	0.48	0.46	-4%	0.02	.889	0.10	0.21	120%	2.68	.105
Gluster	0.62	0.67	9%	0.11	.740	0.60	0.60	0%	0.00	1.000	0.02	0.08	300%	1.89	.172
Fusesource	0.37	0.42	16%	0.24	.625	-	-	-	-		-	-	-	-	
NuCivic	1.27	1.62	28%	1.54	.218	0.63	0.67	6%	0.05	.825	0.63	0.94	48%	2.70	.103
Pandas	11.54	16.67	45%	34.33	.000***	2.48	3.37	36%	5.31	.023*	9.06	13.31	47%	32.54	.000***
Pentaho	2.44	2.44	0%	0.00	1.000	1.10	0.94	-14%	0.51	.477	1.35	1.50	11%	0.52	.473
Rabbitmq	0.27	0.67	150%	7.77	.006**	-	-	-	-		-	-	-	-	

Table 5.2: Results of ANOVA for weekly Entries (p < .1 † ; p < .05 * ; p < .01 ** ; p < .001 ***).

Exits

We follow an analogous approach for the community exits. First we drop 5 projects as can be seen in Table 5.3, because they were acquired in 2015, which means departures can not be observed in a part of the observed period, as we decided on minimum 19 weeks of inactivity

[4]We define the velocity as the number of development actions observed during the observation period, divided by the length of the observation period (Appendix D). It is a quite similar to the averages from the ANOVA studies.

as an exit condition.

For all exits, we notice 6 of 11 projects that significantly increase ($\bar{\Delta}$=147.3%, σ_{Δ}=140.3%), while no decrease in exit rate is observed after acquisition. For the developers 5 out of 9 projects show a significant increase ($\bar{\Delta}$=193.4%, σ_{Δ}=228.6%), and for users & visitors also 5 out of 9 ($\bar{\Delta}$=189.4%, σ_{Δ}=122.9%). For all 11 projects all the mean exit rates increase after acquisition. Thus, solid support was found for *Hypotheses 2.1 & 2.2*.

	All Exits					**Developer Exits**					**Users & Visitors Exits**				
Community	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value	$\bar{x}_{pre}$	$\bar{x}_{post}$	Δ (%)	F-value	p-value
Tutumcloud															
Ansible															
Spree															
Sequenceiq															
Thinkaurelius															
Feedhenry	0.29	0.63	120%	4.91	.029*	0.21	0.29	36%	0.50	.481	0.08	0.35	350%	7.00	.009**
Elgg	0.10	0.50	420%	12.90	.001**	0.10	0.15	60%	0.58	.449	0.00	0.35	200%	16.07	.000***
Wiredtiger	0.25	0.40	62%	1.73	.191	0.02	0.13	600%	5.02	.027*	0.23	0.27	17%	0.14	.706
Ceph	2.29	5.94	160%	47.62	.000***	1.81	4.21	133%	29.09	.000***	0.48	1.73	260%	29.77	.000***
Eucalyptus	0.67	0.83	23%	0.84	.362	0.58	0.67	17%	0.45	.502	0.10	0.15	60%	0.66	.417
Gluster	0.38	0.56	45%	1.26	.265	0.37	0.50	37%	0.87	.352	0.02	0.06	200%	1.03	.312
Fusesource	0.27	0.35	29%	0.60	.439	-	-	-	-		-	-	-	-	
NuCivic	0.90	1.65	83%	8.66	.004**	0.40	0.75	86%	4.65	.033*	0.50	0.90	81%	5.73	.019*
Pandas	10.83	17.44	61%	49.68	.000***	2.06	3.73	81%	18.89	.000***	8.77	13.71	56%	41.37	.000***
Pentaho	1.87	2.62	40%	7.60	.007**	0.63	1.06	67%	4.71	.032*	1.23	1.56	27%	2.25	.137
Rabbitmq	0.25	0.37	46%	0.99	.323	-	-	-	-		-	-	-	-	

Table 5.3: Results of ANOVA for weekly Exits (p < .1 † ; p < .05 * ; p < .01 ** ; p < .001 ***).

5.2 Results & Discussion

As summarised in Table 5.4, our findings were in part confirmed and somewhat diverged from the theoretical propositions.

Hypothesis	**Conclusion**
Hypothesis 1.1	Not supported
Hypothesis 1.2	Partially supported
Hypothesis 2.1	Supported
Hypothesis 2.2	Supported
Hypothesis 3.1	Inversely supported
Hypothesis 3.2	Partially supported

Table 5.4: Summary of results and hypotheses support.

Our observations did not capture a reinforcing effect in developer arrival by acquisition (*Hypothesis 1.1*). Among the few projects where it did show present, `Ansible` experienced a negative effect. That community has the highest velocity (average number of contributions by week), and maybe `Red Hat`, it's acquirer, exercised a higher control on incoming code, because it planned on open-sourcing the software's commercial extension: `Tower`. That particular observation period is also limited due to the short tenancy of the new owner.

We observed that many of the acquisitions function as "acq-hires" Coyle and Polsky (2013); Selby and Mayer (2013). The core developers of the project are employees/founders of the startup dyad, and, thus, embedded as a small development team, whose shape does not change much through the acquisition event. Other community developers are in a peripheral position (e.g. `WiredTiger` has a total of only 22 code contributors for its entire development). It might be that together with a lack of openness and transparency (West and O'Mahony, 2008; Spaeth et al., 2015) of the startup the size of the developer group is limited, which in turn doesn't allow to measure an acquisition effect in terms of community arrivals/entries.

For new-joining users & visitors our expectations were partly confirmed (*Hypothesis 1.2*). Once again, the significance of our results seem to struggle with low weekly rates, which is probably due to the fact that we were not able to measure the non-`GitHub` user & visitor interactions. We noticed that `Spree` did not follow this trend. This makes sense when reading the following excerpt from the `SpreeCommerce` blog: "*We will be spending significantly less time as a company on the Spree project ... This means that you can expect a decline in contributions from employees who were previously being paid to work on open source as part of their day job.* (Schofield, 2015). Stewart and Gosain (2006) pointed out that sponsorship attracts great user interest. In this case we notice the inverse effect, where users & visitors are repelled by an announcement of the sponsor taking distance.

We learn that overall, exit rates are increasing for developers, and users & visitors (*Hypothesis 2.1 & 2.2*). This finding might resonate with von Krogh et al. (2012), where the developers changes institutions if his/her social practice is not supported. One could argue that the increase in exit rates is simply an inherent part of systematic community growth (more one-time need-driven committers creates more exits). However, this would require a similar trend for the entries, which is missing for this sample.

Our most remarkable result was that technical contributions decrease after acquisition (in opposition with *Hypothesis 3.1*). For some projects, such as `Spree`, this can be explained by realignment of the core developers, that are members of the startup, towards new strategic goals, which turn away from the OSS community. In other cases, like `Ansible`, the acquirer highlights in the press that they want to change as little as possible of the relation between the acquiree and the community. Nonetheless, less technical activity takes place. One could state that OSS projects naturally show less technical contributions at the start of maturity, which coincides with the acquisition. We reject this argument, as we found no documentation of such an effect (in `GitHub` commit graphs or literature).[5] We expect two reasons behind the decrease of technical contributions: (1) as the higher exit rates shows, less external developers contribute to the project, and (2) internal developers (i.e. employees of the acquired startup)

[5]To the contrary, Santos et al. (2012) points out that project attractiveness increases with the stage of development.

are possibly required to focus on sharing knowledge and integrating the open source and/or private software in the larger portfolio of the acquiring firm, which takes their time away from technical contribution to the project. As an exception to this rule, there is `NuCivic` which enjoys more technical contributions post-acquisition. We believe this has to do with the fact that the acquiring firm delivers specific solutions for government, a special, large and rigid market.

Most of the projects in our sample show an increase in non-technical contributions after the acquisition (*Hypothesis 3.2*). This is supported and explained by a good project-to-project relation with higher entry rates of users & developers. Due to media coverage new people are attracted to the project, and they ignite a surge of non-technical communication. Other projects (`Tutum`, `Spree`, `Eucalyptus`, and `Pandas`) show a decrease. We did not find shared attributes for these communities in our data (Appendix C & D), but we did notice that each of these acquisitions were accompanied with official press releases that stressed the importance of the newly gained human resources ("acq-hire"), and not as much the continuation of the community. It is possible that this might have scared off potential new users, as the future of the community might have been insecure.

5.3 Findings

Based on descriptive statistics, we know that the sample communities are well-diverse in terms of size, `GitHub`-use, velocity, market sector, and licenses (Chapter 4). Despite that, statistical support was found for most of the hypotheses we constructed. The main takeaway is that the acquisition event clearly inhibits the technical contributions, which stands in congruence with the fact that more developers leave the community, while we don't see the same happening for newly joining developers. This can be seen as a reaction to the assumption or fact that the acquiring firm trades the incentives for community adoption off for more appropriation of returns (West and O'Mahony, 2004). That phenomenon makes sense, given that startup ventures often are more focused on reaching critical mass than profitability (Gorman, 2006), while an acquiring firm is more interested in maximising value from the expensive acquisition. Another explanation, would be that a shift in external development resources takes place towards other projects of the acquirer's OSS portfolio, which plays in their benefit.

Non-technical contributions show mixed results, as proof was found for more community entries of users & visitors, as well as exits. Part of this can certainly be explained by the large media coverage, which simply increases popularity of a community and potentially creates a sort of hype. On the other hand, according to preferential attachment (Santos et al., 2012), users & visitors can behave as an extension of the developer community. In other words, if developers jump off the project, visitors will quickly get the word around and users will invest less time in contributing.

In summary, our findings indicate that the dyad acquisition seems to decrease *project attractiveness* (Ågerfalk and Fitzgerald, 2008; Santos et al., 2012).

5.4 Conclusion

In this chapter we went through a statistical analysis in a attempt to support our hypotheses, which succeeded to some extent. The results were discussed, put in the light of sample-specific contexts, and finally summarised. We will elevate these findings to literature/practical implications, limitations, and future work in the next chapter.

Chapter 6

Conclusion

6.1 Implications

6.1.1 For Research

This thesis contributes to the vast literature stream of open source software in several ways.

Firstly, it adds to the growing research stream of OSS organisational sponsorship (Shah, 2006; Stewart et al., 2006; Spaeth et al., 2015). These papers often focus on individual motivations, while we take a community-perspective on the matter, similar to West and O'Mahony (2004, 2008). In many cases, the relationship between firm and community is tackled Lerner and Tirole (2005a); Dahlander and Magnusson (2005, 2008). As it was difficult to position our cases between community- or firm-initiated projects (Dahlander, 2007), we introduced "dyads" of startup ventures and their corresponding OSS community. The emerging conclusion of existing literature is that a private, for-profit organisational sponsor has challenges to maintain the community, due to (perceived) firm openness and other attributes. Our contribution is that this "dyad tension" is probably worsened due to acquisition by a more profit-focused firm, which is supported by our findings about the technical dynamics of the community. We add that through a.o. acquisition, the organisational sponsor might be an additional influence factor on Santos et al. (2012)'s project attractiveness.

Secondly, some papers investigate developers and users/visitors as key community contributors (von Krogh and von Hippel, 2006; Santos et al., 2012). Unlike most literature that merely focuses on programmers, we followed the same approach. Our different results for each role confirms that this split-up makes sense. In the end, programmers and non-programmers occupy quite dissimilar roles in an open source software community, and, thus, behave differently.

Thirdly, we add to the growing number of papers that source research data from `GitHub` (Thung et al., 2013; Marlow et al., 2013; Dabbish et al., 2012; Kalliamvakou et al., 2015; Scholtes et al., 2015). The - relatively - new platform brings the benefit of hosting social-

media-like functionality on top of traditional `git`-interactions, which brings more potential to study the community beyond development traffic. We have written a `Python` module that can collect in-depth data from large repositories, enrich it in meta data, and prepare it for data analysis. This code contributes in a unique way to the set of available tools for research around OSS, in which `GitHub` plays an ever increasing role.

One literature stream looks into participant turnover of online communities (Sharma et al., 2010; Ransbotham and Kane, 2011). Our post-acquisition observations imply a higher turnover, which might be anteceded by the lack of person-organisation fit (Butler et al., 2014).

An extensive sample of OSS startup acquisitions was presented using a procedural approach, and showed an increasing trend. That finding confirms motivations to venture in the OSS space, as presented by Schaarschmidt and von Kortzfleisch (2014); Gruber and Henkel (2006). Also, the structural integration of an acquired technology firm brings the risk of disrupting the innovative capability (Puranam et al., 2009), which might be an antecedent for decreased community activity.

The cases investigated in our sample showed many traces of the "acq-hire" phenomenon (Coyle and Polsky, 2013; Selby and Mayer, 2013). This mechanism makes particular sense in the OSS game, as not only dynamic capabilities and tacit knowledge are preserved, but also the social network and leadership of the core developers. Although we expect that the retention of key personnel is essential to maintain the community, it is possible that (part of) their effort is assigned to new activities for the benefit of the acquiring firm, which results in less technical productivity of the OSS community.

The scope of this research is rather specific. Therefor, it serves well as a use case for existing theories from literature. On the other hand, we hope to ignite a new avenue of academic work based on the research gap we recognised: startup exits of OSS communities. It should be generalised further to non-startup acquisitions and IPO's, in order to understand the robustness or sensitivity of an OSS community when (one of) its sponsor(s) or the dyad firm experiences a governance-shifting event. We believe such a research stream would complement well to the existing literature around OSS organisational sponsorship and take-aways for the active industry.

6.1.2 For Practice

The utility of our findings for the practice of commercialising open source software is evident. As we provided proof that there is an effect on the community upon firm acquisition, decision-makers in the industry can inform themselves about our findings.

Firstly, acquisition activity in the OSS space has been registered for longer than the

past decade, and shows an increasing trend up to date. Although, rumours of future IPO's reached the surface (Waters, 2015), their numbers would remain small compared to acquisitions. Thus, we expect that practice can benefit in the future from research insights about purchasing commercial firms in the open source sector (such as those that we provide).

It is important to realise that community wealth might not be the main priority of the acquiring firm, but exploiting market opportunities is. As such, the inhibiting effect of an acquisition on the community, is merely collateral damage in favor of commercialisation. In terms of the "acqhire", where a lot of importance is attached to the human resources gain of the startup, the acquirer could favor organisation-wide use of their knowledge and capabilities over maintaining the community leadership and productivity.

So, we propose that acquisitions of pure OSS startups have a negative effect on the community technical productivity and developer group size. However, we believe that must not necessarily be the case, if the acquirer let's the startup and its personnel continue with "business as usual". That would require a low level of structural integration. For non-technical contributors and contributions, there seems to be growth potential probably due to media coverage. However, there are cases where this does not hold, which might be a reflection of the decreased developer dynamics in the community.

An important caveat, is that higher entry-exit rates do not correspond with more growth, but a higher turnover, which is not always a good thing. If community participants' tenure is too short, important knowledge might get lost or essential tasks go undone.

One should as well discern acquirers between pure OSS firms (such as `Red Hat` and companies of which OSS is only a minor part of business (like `HP`). For the first case, the acquired OSS product often gets integrated in their larger portfolio. Possibly, the target community might be consumed by the larger portfolio community, which doesn't result in a loss, but a redistribution of resources, in the perspective of the acquiring firm.

The acquiree ought to have its reasons for the acquisition: community/product opportunities, financial exit, career decision, workload reduction. If post-acquisition community wealth is important for the founder(s), it is best to discuss this as part of the acquisition deal, as it might not be ensured by the benevolence of the acquiring firm. Of course, exit pressure from venture capital investors can obstruct the required leeway to negotiate such conditions.

In conlusion, we observe that a startup acquisition is bad for project attractiveness.

6.2 Limitations

Our research methods carry several limitations. We defined a startup-community dyad as two entities that rely heavily on each other. So, it was not easy to frame the population,

which would be subjected to analysis. Although we listed clear selection criteria, some of them are prone to subjectivity. In other words, a slight, but justified change could influence our population size significantly.

By using `GitHub` as a purposive sample criterion, we might have gathered a biased representation of our population. Looking at the time line, this problem was not observed. However, one could imagine industry sector preferences for other platforms (such as `BitBucket`[1]) for example.

Focusing only on data from `GitHub`, we miss out on other communication of the community. We noticed that many projects host forums and mailing lists. Events and the public website also represent alternative communication channels. These were not looked at in this study.

Scholtes et al. (2015) defends that the mere number of commits is not a good representation of productivity. Although we do not fully agree (4), it is up to the reader to evaluate the validity of that decision.

A question that remains unanswered is whether the post-acquisition effects are transitional (short-term) or lasting (long-term). With our statistical model we assumed a long-term shift in community dynamics. If the changes are transitional, they will have been averaged out over one year. In that case, another statistical model and observation period might be appropriate.

We mapped our measurements around the public announcement date of the acquisition. However, an acquisition can also be seen as a process, consisting of several phases, which exert diverse effects on community dynamics. This is not taken into account in our thesis.

Lastly, contributor's intentions are not fully captured by just counting contribution events by their type. For example, commits of documentation or project management-related tasks are considered as technical contributions. All source projects of the organisation were included in our data.[2] Some of these repositories might serve for non-technical content management and, thus, their commits incorrectly increase the number of technical contributions.

6.3 Future Work

This work takes a deductive approach, where hypotheses are distilled from literature, and empirically tested using quantitative data. Keeping our research question unaltered, there are alternative ways to go about this. We propose on one hand to enlarge the data set by other communication channels (such as mailing lists and forums), forking behaviour, and/or leveraging the profile information of contributors to extract more explanatory variables. We also believe that there are stronger statistical models out there, than the simple ANOVA, to extract results from our data set and its extension as proposed above.

[1] www.bitbucket.org

[2] For some communities that consisted of more than 50 repositories.

On the other hand, a qualitative approach may be taken, by analyzing some of the textual content of the non-technical contributions (natural language processing). We read up on a lot of contextual information in the acquirer's press releases of the acquisition events. That too would be a very interesting source of qualitative data. Above all, we strongly recommend future studies to include a survey analysis from OSS developers/entrepreneurs, who are in the middle of the acquisition and community. Such data will allow us to confirm the speculations we formulated behind the effects we observed in community dynamics.

Looking at alternative research questions, to further pave the avenue for understanding the influence of firms on communities, we propose to include non-startup acquisitions (e.g. `Novell` by `The Attachmate Group`). Although it diversifies the sample a lot and we stop talking about pure OSS companies, it must be interesting to see if the decreased project dynamics hold under those conditions as well. On the long term, if OSS IPO's start appearing as predicted by Waters (2015), it would certainly be interesting to know if it has similar effects as the acquisition. Therefor, we recommend to elevate the analysis to a more general exit context. Stepping away from the "dyad", one could also look into communities that involve several companies (co-opetition), such as the acquisition of `SuSe` by `Novell`.

6.4 Summary

In this thesis we investigated dyads of open source software startups and their communities. More particular, the effect that a firm acquisition has on the community in terms of entry-exit dynamics and contribution rates. From existing literature we synthesised hypotheses that were quantitatively tested with a sample of 16 cases that were gathered using public `GitHub` data. Our findings showed that technical contributions decrease after the acquisition event, while non-technical contributions experience mixed effects. Developers, and users & visitors exit rates increase, while a similar effect is not observed for entries. We conclude that the acquisition is bad for community attractiveness.

Appendix A

List of Acquisitions

Table A.1 shows an overview of all acquisitions we reviewed for our sample. The *Date* column represents the date of public announcement of the acquisition. The *Pop?*-flag indicates whether or not the acquisition case belongs to the population, and the *Sam?*-flag shows whether or not the case is included in our sample. Finally, the *Source* column shows in which of our sampling phases, the companies were discovered.

Acquiree	**Acquirer**	**Date**	**Pop?**	**Sam?**	**Source**
Altamira Software	Microsoft	27/09/1996	no		Crunchbase Export
AlphaNumerica	CollabNet	21/08/2000	no		Snowball
Linuxcare	TurboLinux	21/02/2001	no		Crunchbase Export
Star Division	Sun Microsystems	02/01/2002	no		Snowball
Ximian	Novell	04/08/2003	no		Snowball
SuSe	Novell	07/11/2003	no		Snowball
Sistina Software	Red Hat	18/12/2003	no		Snowball
Gluecode Software	IBM	10/05/2005	yes	no	Crunchbase Export
JBoss	Red Hat	10/04/2006	yes	no	Snowball
Metamatrix	Red Hat	02/06/2007	no		Snowball
Scalix	Xandros	18/07/2007	yes	no	Snowball
Clam Antivirus	SourceFire	17/08/2007	no		Snowball
Zimbra	Yahoo	17/09/2007	yes	no	Snowball
MySQL	Sun Microsystems	01/01/2008	yes	no	Crunchbase Export; Snowball
Imity	ZYB	22/04/2008	no		Crunchbase Export
Koders	Black Duck Software	28/04/2008	no		Crunchbase Export; Snowball
Identyx	Red Hat	19/06/2008	no		Snowball
Symbian Software	Nokia	24/06/2008	yes	no	Crunchbase Export
Linspire	Xandros	01/07/2008	yes	no	Snowball
Qumranet	Red Hat	04/09/2008	yes	no	Snowball
Projity	Serena Software	23/09/2008	yes	no	Crunchbase Export
XenSource	Citrix	22/10/2008	yes	no	Snowball
SharpForge	Codesion	03/02/2009	yes	no	Snowball
Open Source Food	Tsavo Media	06/02/2009	no		Crunchbase Export
Sun Microsystems	Oracle	20/04/2009	no		Snowball
Ohloh	Geeknet	28/05/2009	no		Snowball
SpringSource	VMware	10/08/2009	yes	no	Snowball
Zimbra	VMware	12/01/2010	no		Snowball
Rabbit Technologies	SpringSource	13/04/2010	yes	yes	Crunchbase Export; Snowball
Layerboom	Joyent	15/07/2010	no		Snowball
Ohloh	Black Duck Software	05/10/2010	no		Snowball
Codesion	CollabNet	19/10/2010	no		Snowball
Novell	The Attachmate Group	10/11/2010	no		Snowball
SpikeSource	Black Duck Software	17/11/2010	no		Snowball
Makara	Red Hat	30/11/2010	no		Snowball
Curverider	Thematic Networks	07/12/2010	yes	yes	Crunchbase Export
All for Good	Points of Light Institute	22/12/2010	yes	no	Crunchbase Export

DimDim	SalesForce	06/01/2011	yes	no	Snowball
Olliance Group	Black Duck Software	10/01/2011	no		Snowball
WaveMaker	VMware	08/03/2011	yes	no	Crunchbase Export
Magento	eBay	06/06/2011	yes	no	Crunchbase Export; Snowball
Tekriti Software	Kellton Tech Solutions	01/08/2011	no		Crunchbase Export
Gluster	Red Hat	07/10/2011	yes	yes	Snowball
MobiPrimo Technologies	PubMatic	13/03/2012	no		Crunchbase Export
Likewise Software	EMC	20/03/2012	no		Crunchbase Export
FuseSource	Red Hat	27/06/2012	yes	yes	Snowball
CureTogether	23andMe	10/07/2012	no		Crunchbase Export
Geoloqi	Esri	16/10/2012	yes	no	Crunchbase Export
Clover	First Data Corporation	01/12/2012	no		Crunchbase Export; Snowball
Arkeia Software	Western Digital Technologies	22/01/2013	no		Crunchbase Export
SourceNinja	Kissmetrics	01/02/2013	no		Crunchbase Export
WaveMaker	Pramati Technologies	02/05/2013	no		Crunchbase Export
Zimbra	Telligent System	15/07/2013	no		Snowball
Standing Cloud	AppDirect	10/09/2013	no		Crunchbase Export
Pandastream	Copper.io	01/10/2013	yes	no	Crunchbase Export
SourceFire	Cisco	07/10/2013	yes	no	Snowball
Interactive Ideas	CMS Distribution	05/11/2013	no		Crunchbase Export; Snowball
CollabNet	Woven Systems	03/02/2014	no		Snowball
Coverity	Synopsys	19/02/2014	no		Crunchbase Export; Snowball
Jaspersoft	TIBCO Software	28/04/2014	yes	no	Crunchbase Export; Snowball
Inktank	Red Hat	30/04/2014	yes	yes	Crunchbase Export; Snowball
OhmData	WANdisco	01/05/2014	no		Crunchbase Export; Snowball
XA Secure	HortonWorks	15/05/2014	no		Snowball
eNovance	Red Hat	18/06/2014	no		Crunchbase Export
Blackbird	Pythian	26/06/2014	no		Crunchbase Export
Gazzang	Cloudera	02/07/2014	yes	no	Snowball
SmartThings	Samsung Electronics	16/07/2014	no		Crunchbase Export; Snowball
Orchard	Docker	23/07/2014	yes	no	Snowball
SmartThings	Samsung Global Innovation Center	14/08/2014	no		Crunchbase Export; Snowball
Eucalyptus Systems	Hewlett-Packard	12/09/2014	yes	yes	Crunchbase Export; Snowball
FeedHenry	Red Hat	18/09/2014	yes	yes	Snowball
DataPad	Cloudera	30/09/2014	yes	yes	Snowball
Alien Blue	Reddit	16/10/2014	no		Crunchbase Export
Code For The People	Automattic	06/11/2014	no		Crunchbase Export; Snowball
Rare Crowds	MediaMath	10/11/2014	no		Crunchbase Export
Actuate	OpenText Corporation	10/12/2014	no		Crunchbase Export; Snowball
WiredTiger	MongoDB	16/12/2014	yes	yes	Snowball
NuCivic	GovDelivery	17/12/2014	yes	yes	Crunchbase Export
EnterpriseDB	Peakequity	19/12/2014	no		Crunchbase Export
ThinkAurelius	DataStax	03/02/2015	yes	yes	Snowball
Pentaho	Hitachi Data Systems	10/02/2015	yes	yes	Crunchbase Export
Mortar Data	Datadog	11/02/2015	yes	no	Crunchbase Export; Snowball
Found	Elastic	10/03/2015	no		Crunchbase Export
Kitematic	Docker	12/03/2015	yes	no	Snowball
SocketPlane	Docker	15/03/2015	no		Snowball
SequenceIQ	HortonWorks	13/04/2015	yes	yes	Snowball
Gitorious	GitLab	03/05/2015	yes	no	Crunchbase Export; Snowball
Packetbeat	Elastic	27/05/2015	no		Crunchbase Export
Piston Cloud Computing,	Cisco	03/06/2015	no		Crunchbase Export; Snowball
Myrrix	Cloudera	16/07/2015	yes	no	Snowball
Zimbra	Synacor	18/08/2015	no		Snowball
Onyara	HortonWorks	25/08/2015	yes	no	Snowball
Spree Commerce	First Data Corporation	18/09/2015	yes	yes	Crunchbase Export
Elastic Cloud	Elastic	03/10/2015	no		Crunchbase Export
Zend Technologies	Rogue Wave Software	06/10/2015	no		Snowball
Ansible	Red Hat	16/10/2015	yes	yes	Snowball
RoboVM AB	Xamarin	21/10/2015	yes	no	Crunchbase Export; Snowball
Tutum	Docker	21/10/2015	yes	yes	Snowball
Pandastream	Telestream	02/11/2015	no		Crunchbase Export
Protecode	Synopsys	06/11/2015	no		Crunchbase Export
Amee	Pi	18/11/2015	no		Crunchbase Export; Snowball
Appcelerator	Axway	18/01/2016	yes	no	Crunchbase Export
Sleepycat Software	Oracle Corporation	14/02/2016	yes	no	Crunchbase Export; Snowball
PredictionIO	Salesforce	19/02/2016	yes	no	Crunchbase Export; Snowball
Xamarin	Novell	24/02/2016	yes	no	Snowball
Bashton	Claranet	29/02/2016	no		Crunchbase Export

Table A.1: List of all acquisitions.

Appendix B

List of IPO's

During our sampling process we, next to acquisitions, also registered all IPO's in the OSS space that we found. Table B.1 gives an overview, and proves to say that the exit mechanism is clearly less common then the acquisition (Appendix A).

Company	Date
Actuate	17/07/1998
Red Hat	20/08/1999
SourceFire	08/03/2007
WANdisco	01/06/2012
Hortonworks	11/12/2014

Table B.1: List of all IPO's.

Appendix C

Sample Description

Table C.1 summarises important data from the set of 13 studied cases. It gives (1) the community name, (2) the name of the acquired startup, (3) in which year the startup was founded, (4) the amount of venture capital that startup collected (if disclosed), (5) the name of the acquiring firm, (6) the acquisition date, (7) the utilised license(s), (8) the license category (as defined in Chapter 2), (9) the price of the acquisition deal (if disclosed), (10) whether the OSS product received an on-line rebranding, (11) the business model exploited by the startup (as defined in Chapter 2), (12) what market section the product is in, (13) what the product is. We defined the market section (12) in three fields : Infrastructure is software that supports network infrastructure (data centers, server clusters, distributed networks, etc.), Development are frameworks and platforms that aid in the development of other software, and Applications refer to solutions that are more focused towards end-user value.

Community	Acquiree	Found.	VC (m$)	Acquirer	Acq. Date	License Type	License Cat.	Price (m$)	Online Merge	Business Model	Market	Product
TutumCloud	Tutum	2013	2.72	Docker	21/10/2015	ASLv2	Permissive	-	No rebranding	Hard & Soft Services	Infrastructure	Docker Container Platform
Ansible	Ansible	2013	6	Red Hat	16/10/2015	GPL	Restrictive	110	No rebranding	Software Upgrade	Infrastructure	Server Management
Spree	Spree Commerce	2011	6.5	First Data Corporation	18/09/2015	New BSD License	Permissive	-	No rebranding	Hard & Soft Services	Development	Web Application Framework
Cloudbreak	SequenceIQ	2014	-	HortonWorks	13/04/2015	ASLv2	Permissive	-	No rebranding	Hard Services	Infrastructure	Big Data on Cloud
Titan	ThinkAurelius	2014	-	DataStax	03/02/2015	ASLv2	Permissive	-	No rebranding	Soft Services (Consulting & Training)	Application	Graph Database
FeedHenry	FeedHenry	2010	9	Red Hat	01/10/2014	FHL/ ASLv2	Corporate	82	Rebranding	(Dual) Licensing/Loss-Leader	Development	Mobile Application Platform
Elgg	CurveRider	2005	0.5	Thematic Networks	07/12/2010	GPLv2 & MIT	Restrictive	-	Rebranding	Soft Services & Software Upgrade	Application	Social Networking
WiredTiger	WiredTiger	2012	-	MongoDB	16/12/2014	GPLv3	Restrictive	-	No rebranding	Soft Services	Infrastructure	MongoDB Storage Engine
Ceph	Inktank Storage	2012	14.4	Red Hat	30/04/2014	LGPLv2.1	Moderately Restrictive	175	Rebranding	Soft Services (Training and Resources)	Infrastructure	Distributed storage
Eucalyptus	Eucalyptus Systems	2009	55	Hewlett-Packerd	12/09/2014	GPLv3	Restrictive	100	Rebranding	(Dual) Licensing	Infrastructure	Hypervisor
GlusterFS	Gluster	2005	8.5	Red Hat	07/10/2011	GPLv2/ LGPLv3	Restrictive	136	Rebranding	Soft Services (Training, Support)	Infrastructure	Distributed File System
FuseSource/ JBoss-Fuse	FuseSource	2010	-	Red Hat	27/06/2012	ASLv2	Permissive	-	Rebranding	Hard & Soft Services (Customization)	Infrastructure	Integration Platform
DKAN	NuCivic	2014	-	GovDelivery	17/12/2014	GPLv2	Restrictive	-	No rebranding	Hard & Soft Services (Cloud Hosting and Support)	Application	Big Data
Pandas	DataPad	2013	1.7	Cloudera	30/09/2014	New BSD	Permissive	-	Rebranding	Hard & Soft Services (Cloud Hosting)	Application	Data Visualisation
Pentaho	Pentaho	2004	72	Hitachi Data Systems	10/02/2015	ASLv2 & Pentaho Community	Permissive & Corporate	550	No rebranding	(Dual) Licensing & Services	Application	Data Analytics
RabbitMQ	Rabbit Technologies	2007	-	SpringSource	13/04/2010	ASLv2 & MPL	Corporate	-	No rebranding	(Dual) Licensing & Soft Services	Infrastructure	Messaging

Table C.1: Description of GitHub Sample

Appendix D

Data Description

Table D.1[1] shows an overview of how much data we collected from our sample and how it breaks down to types of contributions and users. Percentages are always based on total contributions/contributors. The observation window determines which contributions will be used for statistical analysis. Nonetheless, we still require the full data set to calculate the entry-exit dynamics.

[1] For `DataPad` we only collected the `Pandas` repository of the `PyData` organisation.

Community	Total Contributions	Technical Contributions		Non-technical Contributions		Total Contributors	Developers		Users & Visitors		Observation Window (weeks)	Contributions in Observation Window	
Tutumcloud	6 738	3 549	53%	3 189	47%	703	177	25%	526	75%	46	2 978	44%
Ansible	129 513	36 495	28%	93 018	72%	8 980	2 237	25%	6 743	75%	56	54 397	42%
Spree	63 474	27 215	43%	36 259	57%	2 607	971	37%	1 636	63%	64	7 667	12%
Sequenceiq	18 269	11 369	62%	6 900	38%	346	158	46%	188	54%	76	13 840	76%
Thinkaurelius	10 056	4 964	49%	5 092	51%	498	58	12%	440	88%	104	4 521	45%
Feedhenry	7 208	5 050	70%	2 158	30%	110	70	64%	40	36%	104	3 491	48%
Elgg	127 213	79 898	63%	47 315	37%	310	90	29%	220	71%	104	20 017	16%
Wiredtiger	25 694	15 546	61%	10 148	39%	58	21	36%	37	64%	104	13 971	54%
Ceph	624 700	577 567	92%	47 132	8%	6 455	6 160	95%	295	5%	104	64 990	10%
Eucalyptus	42 959	35 148	82%	7 811	18%	213	174	82%	39	18%	104	20 613	48%
Gluster	24 895	23 376	94%	1 519	6%	344	284	83%	60	17%	104	6 256	25%
Fusesource	30 466	27 459	90%	3 007	10%	251	204	81%	47	19%	104	9 592	31%
NuCivic	26 825	21 098	79%	5 727	21%	241	131	54%	110	46%	104	8 553	32%
Pentaho	80 966	56 824	70%	24 124	30%	551	346	63%	205	37%	104	33 569	41%
Pandas	96 444	13 180	14%	33 264	86%	2 933	612	21%	2 321	79%	104	47 803	50%
RabbitMQ	48 513	39 005	81%	9 508	19%	918	236	26%	682	74%	104	11 892	25%
Total Sum	1 363 933	977 824	72%	386 108	28%	25 518	11 929	47%	13 589	53%	-	324 150	24%
Mean	85 246	61 114	-	24 132	-	1 595	746	-	849	-	-	20 259	-
Std Dev.	149 328	13 195	-	29 451		2574	1545	-	1696	-	-	19 470	-

Table D.1: Sizes and breakdown of collected GitHub data.

Appendix E

PyGitHub Script

The `Python` script that we wrote and used is available under the following `GitHub` repository : https://github.com/droegier/ThesisOSS.

E.1 How does it work?

Using a repository or organisation name, it has the ability to gather all (non)-technical `GitHub` activity of that entity. Additionally, it permits lightweight data analysis and transformation for further statistical operations.

Multiple dummy accounts can be created to access the `GitHub` service API. The class modules contain working threads that allow to collect the data more quickly. Locks are used to write the data to `xml`-files, before it is transformed to `csv`-files.

It was written using `PyGitHub v1.25.2`[1] and documentation is included in the *readme* file and as comments in the source code.

E.2 How does it help?

A special benefit of this script is that it implements heuristics (described in Chapter 4) that permit the identification of profiles that are not recognised by `GitHub` from the `git`-data. This helps to improve the completeness of the data, if further analysis needs to be done around profiles characteristics of contributors. Table E.2 shows the data improvement for a randomly selected group of core repositories from our sample. The last two columns indicate which ratio of the `git` developer accounts are linked to a user account/profile in `GitHub` before (as seen on the `GitHub` website) and after our heuristics.

[1] http://pygithub.github.io/PyGithub/v1/

Organisation Name	**Repository Name**	**Total # Developers**	**GitHub Profile Hits** (pre-script)		**GitHub Profile Hits** (post-script)	
PyData	Pandas	612	560	91.5%	593	96.9%
Ansible	Ansible	1 514	1 373	90.7%	1 494	98.7%
Eucalyptus	Eucalyptus	72	47	65.3%	54	75%
Spree	Spree	798	669	83.8%	705	88%
Elgg	Elgg	78	65	83%	70	89.7%

Calling commit-wise data through the public API can be quite slow. For this reason we implemented a multi-threading approach for collecting data using several dummy accounts in parallel. This offers a quicker way of gathering the data.[2]

E.3 Future Improvements

In case other researchers wish to use this code base, we recommend they make their own additions. We identify the following topics as necessary and pending improvements.

- Redesign the class structure, to make them inherited.
- Find the optimal number of dummy accounts (trade-off between API speed and multi-thread memory locking).
- Test the validity of the GitHub profile/account selection heuristics.
- Look into OSS libraries for data analysis inside Python.
- See if integration with GHTorrent is interesting to speed up data collection.

[2] Although we are aware of the `GHTorrent` database (Gousios and Spinellis, 2012), our approach offers a public solution, that does not require access permission from the research group.

Appendix F

Declaration of Originality

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

STARTUP ACQUISITIONS IN THE OSS SPACE : WHAT IS THE EFFECT ON COMMUNITY DYNAMICS?

Authored by (in block letters):

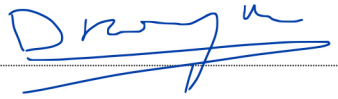
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
ROEGIERS	DAVID

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZURICH 21/05/2016	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Bibliography

Adams, P. J., Capiluppi, A., and Boldyreff, C. (2009). Coordination and productivity issues in free software: The role of brooks' law. In *IEEE International Conference on Software Maintenance, ICSM*, pages 319–328.

Agarwal, R. and Helfat, C. E. (2009). Strategic Renewal of Organizations. *Organization Science*, 20(2):281–293.

Ågerfalk, P. J. and Fitzgerald, B. (2008). Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy. *MIS Quarterly*, 32(2):385–409.

Aghion, P. and Tirole, J. (1997). Formal and Real Authority in Organizations. *Journal of Political Economy*, 105(1):1–29.

Agnihotri, R., Murali, S., and Kothandaraman, P. (2012). Theorization of the open source software phenomenon: a complex adaptive system approach. *Journal of Management & Marketing Research*, 9:1–10.

Alexy, O. T., Block, J. H., Sandner, P., and Ter Wal, A. L. J. (2012). Social capital of venture capitalists and start-up funding. *Small Business Economics*, 39(4):835–851.

Andersen-Gott, M., Ghinea, G., and Bygstad, B. (2012). Why do commercial companies contribute to open source software? *International Journal of Information Management*, 32(2):106–117.

Aslett, M. (2008). VC funding for open source hits an all-time high.

Aslett, M. (2009). VC funding for open source mixed messages from 2008.

Babock, C. (2008). A New Model: Open Source Software After It's Acquired.

Baldwin, C. Y. and Clark, K. B. (2006). The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Management Science*, 52(7):1116–1127.

Bansal, P. and Clelland, I. (2004). Talking Trash: Legitimacy, Impression Management, and Unsystematic Risk in the Context of the Natural Environment. *Academy of Management Journal*, 47(1):93–103.

Benson, D. and Ziedonis, R. H. (2005). Corporate venture capital and the returns to acquiring portfolio companies. *Journal of Financial Economics*, 98(3):478–499.

Benson, D. and Ziedonis, R. H. (2009). Corporate Venture Capital as a Window on New Technologies: Implications for the Performance of Corporate Investors When Acquiring Startups. *Organization Science*, 20(2):329–351.

Bessen, J. (2006). Open Source Software: Free Provision of Complex Public Goods. In *The Economics of Open Source Software Development*, pages 57–81.

Blau, J. (2006). Open source startups speak out.

Block, J. and Sandner, P. (2009). What is the Effect of the Financial Crisis on Venture Capital Financining: Empirical Evidence from US Internet Start-ups. *Venture Capital - An International Journal of Entrepreneurial Finance*, 11(4):295–309.

Bonaccorsi, A., Giannangeli, S., and Rossi, C. (2006). Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. *Management Science*, 52(7):1085–1098.

Brau, J., Francis, B., and Kohers, N. (2003). The Choice of IPO versus Takeover: Empirical Evidence*. *The Journal of Business*, 76(4):583–612.

Burst, A. (2014). Is the Hortonworks IPO a referendum on open source?

Butler, B., Sproull, L., Kiesler, S., and Kraut, R. (2003). Community Effort in Online Groups: Who Does the Work and Why? *Leadership at a Distance*, pages 1–32.

Butler, B. S., Bateman, P. J., Gray, P. H., and Diamant, E. I. (2014). An Attraction-Selection-Attrition Theory of Online Community Size and Resilience. *MIS Quarterly*, 38(3):699–728.

Byfield, B. (2008). VCs regain interest in open source.

Campbell-Kelly, M. and Garcia-Swartz, D. (2010). The Move to the Middle: Convergence of the Open-Source and Proprietary Software Industries. *International Journal of the Economics of Business*, 17(2):223–252.

Capra, E. and Wasserman, A. I. (2008). A framework for evaluating managerial styles in open source projects. *IFIP International Federation for Information Processing*, 275:1–14.

Capron, L. and Shen, J. C. (2007). Acquisitions of private vs. public firms: Private information, target selection, and acquirer returns. *Strategic Management Journal*, 28(9):891–911.

Chengalur-Smith, I., Sidorova, A., and Daniel, S. L. (2010). Sustainability of free/libre open source projects: A longitudinal study. *Journal of the Association for Information Systems*, 11(11):657–683.

Chesbrough, H. W. (2003). *Open Innovation*, volume 2006.

Comino, S. and Manenti, F. M. (2011). Dual licensing in open source software markets. *Information Economics and Policy*, 23(3-4):234–242.

Cook, J. (2005). Venture Capital: Open source startups are hot – too hot?

Coyle, J. F. &. P. and Polsky, G. D. (2013). Acqui-hiring. *Duke Law Journal*, 63(2):280–346.

Crowston, K., Annabi, H., and Howison, J. (2003). Measuring Open Source Software Project Success. *Management Science*, 52(7):1043–1056.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012). Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, (2011):1277–1286.

Dahlander, L. (2005). Appropriation and appropriability in open source software. *International Journal of Innovation Management*, 9(3):259–285.

Dahlander, L. (2007). Penguin in a new suit: A tale of how de novo entrants emerged to harness free and open source software communities. *Industrial and Corporate Change*, 16(5):913–943.

Dahlander, L. and Magnusson, M. (2008). How do Firms Make Use of Open Source Communities? *Long Range Planning*, 41(6):629–649.

Dahlander, L. and Magnusson, M. G. (2005). Relationships between open source software companies and communities: Observations from Nordic firms. *Research Policy*, 34(4):481–493.

Dahlander, L. and O'Mahony, S. (2011). Progressing to the Center: Coordinating Project Work. *Organization Science*, 22(June 2015):961–979.

Dahlander, L. and Wallin, M. W. (2006). A man on the inside: Unlocking communities as complementary assets. *Research Policy*, 35(8 SPEC. ISS.):1243–1259.

Di Tullio, D. and Staples, D. S. (2013). The Governance and Control of Open Source Software Projects. *Journal of Management Information Systems*, 30(3):49–80.

Dinkelacker, J. and Garg, P. (2001). Corporate Source: Applying Open Source Concepts to a Corporate Environment. *1st WS Open Source SE.*

Earley, P. C. (1991). East Meets West Meets Mideast: Further Explorations of Collectivistic and Individualistic Work Groups. *Academy of Management Best Papers Proceedings*, 8(1):205–209.

Faraj, S. and Johnson, S. L. (2011). Network Exchange Patterns in Online Communities. *Organization Science*, 22(6):1464–1480.

Faraj, S., Kudaravalli, S., and Wasko, M. M. (2015). Leading Collaboration in Online Communities. *MIS Quarterly*, 39(2):393–412.

Feller, J., Finnegan, P., and Hayes, J. (2006). Open source networks: an exploration of business model and agility issues. *ECIS 2006 Proceedings*.

Fershtman, C. and Gandal, N. (2007). Open source software: Motivation and restrictive licensing. *International Economics and Economic Policy*, 4(2):209–225.

Fitzgerald, B. (2005). Has Open Source Software a Future? *Perspectives on free and open source software*, (February 1998):570.

Fitzgerald, B. (2006). The Transformation of Open Source Software. *MIS Quarterly*, 30(3):587.

Franke, N. and Von Hippel, E. (2003). Satisfying heterogeneous user needs via innovation toolkits: The case of Apache security software. *Research Policy*, 32(7):1199–1215.

Gaudeul, A. (2007). Do Open Source Developers Respond to Competition? The TEX Study Case. *Review of Network Economics*, 6(2):239.

Ghosh, R. A. (2006). Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU. 26:243.

Gorman, R. (2006). Open-source venture capitalist answers your questions.

Goth, G. (2005). Open source meets venture capital. *IEEE Distributed Systems Online*, 6(6):2.

Gousios, G. and Spinellis, D. (2012). GHTorrent: Github's data from a firehose. In *IEEE International Working Conference on Mining Software Repositories*, pages 12–21.

Graebner, M. E. and Eisenhardt, K. M. (2013). The Seller ' s Side of the as Story : Acquisition and Courtship as Governance in Syndicate Firms Entrepreneurial. *Administrative Science Quarterly*, 49(3):366–403.

Grand, S., von Krogh, G., Leonard, D., and Swap, W. W. (2004). Resource allocation beyond firm boundaries: A multi-level model for open source innovation. *Long Range Planning*, 37(6):591–610.

Grandstrand, O. and Sjölander, S. (1990). The acquisition of technology and small firms by large firms. *Journal of Economic Behavior & Organization*, 13:367–386.

Granstrand, O., Bohlin, E., Oskarsson, C., and Sjöberg, N. (1992). External technology acquisition in large multi-technology corporations. *R&D Management*, 22(2):111–134.

Graves, N. J. D. (2013). *Open Source Software Development as a Complex System*. PhD thesis.

Greenemeier, L. (2005). VCs Back Open-Source Upstarts.

Gruber, M. and Henkel, J. (2006). New ventures based on open innovation an empirical analysis of start-up firms in embedded Linux. *International Journal of Technology Management*, 33(4):356–382.

Gulati, R., Puranam, P., and Tushman, M. (2012). Meta-organization design: rethinking design in interorganizational and community contexts. *Academy of Management Journal*, 51(2):315–334.

Gurbani, V. K., Garvert, A., and Herbsleb, J. D. (2005). A case study of open source tools and practices in a commercial setting. *Proceedings of the fifth workshop on Open source software engineering - 5-WOSSE*, pages 1–6.

Hars, A. and Ou, S. (2002). Working for free? Motivations of participating in open source projects. *International Journal of Electronic Commerce*, 6(3):25–39.

Haruvy, E., Wu, F., and Chakravarty, S. (2003). Incentives for Developers ' Contributions and Product Performance Metrics in Open Source Development : An Empirical Exploration.

Hecker, F. (1999). Setting up shop: The business of open-source software. *IEEE Software*, 16(1):45–51.

Hellmann, T. F. and Puri, M. (2002). Venture Capital and the Professionalization of Start-Up Firms: Empirical Evidence. *The Journal of Finance*, 57(1):169–197.

Henkel, J. (2006). Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy*, 35(7):953–969.

Herbsleb, J. J. D. and Mockus, A. (2003). An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 29(6):481–494.

Hertel, G., Niedner, S., and Herrmann, S. (2003). Motivation of software developers in open source projects: An Internet-based survey of contributors to the Linux kernel. *Research Policy*, 32(7):1159–1177.

Huffaker, D. (2010). Dimensions of Leadership and Social Influence in Online Communities. *Human Communication Research*, 36(4):593–617.

Johnson, J. P. (2006). Collaboration, peer review and open source software. *Information Economics and Policy*, 18(4):477–497.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., and Damian, D. (2015). An in-depth study of the promises and perils of mining GitHub.

Koch, S. (2004). Profiling an Open Source Project Ecology and Its Programmers. *Electronic Markets*, 14(2):77–88.

Koenig, J. (2006). Seven Open Source Business Strategies for Competitive Advantage. *IT Managers Journal*, pages 1–6.

Kooths, S., Langenfurth, M., and Kalwey, N. (2003). Open-Source Software: An Economic Assessment.

Krishnamurthy, S. (2003). An Analysis of Open Source Business MOdels. *Source*, (February):1–21.

Kuan, J. (2001). Open Source Software as Consumer Integration into Production.

Lakhani, K. R. and Von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy*, 32(6):923–943.

Lakhani, K. R., Wolf, B., Bates, J., and Dibona, C. (2002). The Boston Consulting Group Hacker Survey.

Lakka, S., Stamati, T., Michalakelis, C., and Martakos, D. (2011). The Ontology of the OSS Business Model. *International Journal of Open Source Software and Processes*, 3(March):39–59.

LaMonica, M. (2005). Open source, open wallet.

Laurent, A. M. S. (2004). Understanding Open Source and Free Software Licensing. *Ariadne*, page 193.

Lee, S. Y. T., Kim, H. W., and Gupta, S. (2009). Measuring open source software success. *Omega*, 37(2):426–438.

Lerner, J. and Tirole, J. (2003). Some Simple Economics of Open Source. *The Journal of Industrial Economics*, 50(2):197–234.

Lerner, J. and Tirole, J. (2005a). The Economics of Technology Sharing: Open Source and Beyond. *Journal of Economic Perspectives*, 19(2):99–120.

Lerner, J. and Tirole, J. (2005b). The Scope of Open Source Licensing. *Journal of Law, Economics and Organization*, 21(1):20–56.

Lichtenthaler, U. (2008). Open innovation in practice: an analysis of strategic approaches to technology transactions. *IEEE Transactions on Engineering Management*, 55(1):148–157.

Maillart, T., Sornette, D., Spaeth, S., and Von Krogh, G. (2008). Empirical tests of Zipf's law mechanism in open source linux distribution. *Physical Review Letters*, 101(21):1–4.

Marlow, J., Dabbish, L., and Herbsleb, J. (2013). Impression Formation in Online Peer Production : Activity Traces and Personal Profiles in GitHub. In *16th ACM Conference on Computer Supported Cooperative Work*, pages 117–128.

Mehra, A., Dewan, R. M., and Freimer, M. (2008). Firms as Incubators of Open Source Software. *Information Systems Research*, 22(March):1–26.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., and Ye, Y. (2002). Evolution patterns of open-source software systems and communities. *Proceedings of the international workshop on Principles of software evolution - IWPSE '02*, (January):76 – 85.

Nerney, C. (2013). How open source start-ups can get funding and go viral.

O'Grady, S. (2010). The State of Open Source: Startup, Growth, Maturity or Decline?

O'Mahony, S. and Ferraro, F. (2007). The Emergence of a Governance Structure in an Open Source Community. *Academy of Management Journal*, 50(5):1079–1106.

Parker, G. and Van Alstyne, M. (2005). Innovation through optimal licensing in free markets and free software. *Papers.Ssrn.Com*, (September):1–30.

Phipps, S. (2010). Is The "Open Source Bubble" Over?

Phipps, S. (2014). Open source startups: Don't try to be Red Hat.

Puhakka, M., Jungman, H., and Seppänen, M. (2007). *Handbook of Research on Open Source Software.*

Puranam, P., Alexy, O., and Reitzig, M. (2014). What's "new" about new forms of organizing? *Academy of Management Review*, 39(2):162–180.

Puranam, P., Singh, H., and Chaudhuri, S. (2009). Integrating Acquired Capabilities: When Structural Integration Is (Un)necessary. *Organization Science*, 20(2):313–328.

Puranam, P., Singh, H., and Zollo, M. (2006). Organizing for Innovation: Managing the Coordination-Autonomy Dilemma in Technology Acquisitions. *Academy of Management Journal*, 49(2):263–280.

Rahman, M. M. and Roy, C. K. (2014). An insight into the pull requests of GitHub. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, pages 364–367.

Rajala, R., Nissilo, J., and Westerlund, M. (2006). Determinants of OSS revenue model choices. *Proceedings of the Fourteenth European Conference on Information Systems*, pages 1839–1850.

Ranft, A. L. and Lord, M. D. (2000). Acquiring New Knowledge: the Role of Retaining Human Capital in Acquisitions of High-Tech Firms. *Journal of High Technology Management Research*, 11(2):295.

Ranft, A. L. and Lord, M. D. (2002). Acquiring New Technologies and Capabilities: A Grounded Model of Acquisition Implementation. *Organization Science*, 13(4):420–441.

Ransbotham, S. and Kane, G. C. (2011). Membership Turnover and Collaboration Success in Online Communities : Explaining Rises and Falls. *Management Information Systems Quarterly*, 35(3):613–627.

Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49.

Riehle, D. (2009). The Business Model of Commercial Open Source Software. *Proceedings of the Fifteenth Americas Conference on Information Systems*, pages 18–30.

Roberts, J. a., Hann, I.-H., and Slaughter, S. a. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science*, 52(7):984–999.

Rolandsson, B., Bergquist, M., and Ljungberg, J. (2011). Open source in the firm: Opening up professional practices of software development. *Research Policy*, 40(4):576–587.

Rosen, L. (2004). *Open Source Licensing: Software Freedom and Intellectual Property Law.*

Rosenberg, D. (2010). Open source acquisitions - what's the holdup?

Sacks, M. (2015). Competition Between Open Source and Proprietary Software: Strategies for Survival. *Journal of Management Information Systems*, 32(3):268–295.

Santos, C., Kuk, G., Kon, F., and Pearson, J. (2012). The attraction of contributors in free and open source software projects. *The Journal of Strategic Information Systems*, 22(1):26–45.

Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K. (2006). *Understanding free/open source software development processes*, volume 11. Addison-Wesley, 2002.

Schaarschmidt, M. and von Kortzfleisch, H. (2014). Examining Investment Strategies of Venture Capitalists in Open Source Software. *International Journal of Innovation & Technology Management*, 11(4):1–19.

Schireson, M. (2016). The Money in Open Source Software.

Schofield, S. (2015). The Future of Spree Open Source Software.

Scholtes, I., Mavrodiev, P., and Schweitzer, F. (2015). *From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects.* Number December.

Schweitzer, F., Nanumyan, V., Tessone, C. J., and Xia, X. (2014). How Do Oss Projects Change in Number and Size? a Large-Scale Analysis To Test a Model of Project Growth. *Advances in Complex Systems*, 17(07n08):1550008.

Selby, J. and Mayer, K. J. (2013). Startup Firm Acquisitions as a Human Resource Strategy for Innovation: The Acqhire Phenomenon. *Academy of Management Proceedings*, 2013(1):17109–17109.

Sen, R., Subramaniam, C., and Nelson, M. L. (2009). Determinants of the Choice of Open Source Software License. *Journal of Management Information Systems*, 25(3):207–240.

Shah, S. K. (2006). Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science*, 52(7):1000–1014.

Shalley, C. E. (1995). Effects of coaction, expected evaluation, and goal setting on creativity and productivity. *Academy of Management Journal*, 38:483–503.

Sharma, P. N., Daniel, S. L., and Chung, T. T. (2010). The impact of person-organization fit on turnover in open source software projects. *ICIS 2010 Proceedings Thirty First International Conference on Information Systems*, (Kristof 1996).

Shen, J. C. and Reuer, J. J. (2005). Adverse selection in acquisitions of small manufacturing firms: A comparison of private and public targets. *Small Business Economics*, 24(4):393–407.

Shipman, M. D. (1988). *The Limitations of Social Research.*

Singh, P. V. and Phelps, C. (2009). Determinants of Open Source Software License Choice : A Social Influence Perspective. *Social Science Research Network*, pages 1–39.

Singh, P. V., Tan, Y., and Mookerjee, V. (2011). Network effects: The influence of structural capital on open source project success. *MIS Quarterly*, 35(4):813–A7.

Spaeth, S., von Krogh, G., and He, F. (2015). Perceived Firm Attributes and Intrinsic Motivation in Sponsored Open Source Software Projects. *Information Systems Research*, 26(1):224–237.

Spiller, D. and Wichmann, T. (2002). Free/Libre Open Source Software: Survey and Study Basics of Open Source Software Markets and Business Models. *FLOSS Final Report - PArt 3*, (July):58.

Stallman, R. (2001). Philosophy of the GNU Project.

Sterne, P. and Herring, N. (2006). The Open Source Venture Capital Universe.

Stewart, D. (2005). Social Status in an Open-Source Community. *American Sociological Review*, 70(5):823–842.

Stewart, K. J., Ammeter, A. P., and Maruping, L. M. (2006). Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects. *Information Systems Research*, 17(2):126–144.

Stewart, K. J. and Gosain, S. (2006). The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *Source: MIS Quarterly*, 30132208(2):291–314.

Ter Wal, a. L. J., Alexy, O., Block, J., and Sandner, P. G. (2016). The Best of Both Worlds: The Benefits of Open-specialized and Closed-diverse Syndication Networks for New Ventures Success. *Administrative Science Quarterly*.

Thung, F., Bissyandé, T. F., Lo, D., and Jiang, L. (2013). Network structure of social coding in GitHub. *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, pages 323–326.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014). Let's talk about it: evaluating contributions through discussion in GitHub. *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 144–154.

Tunguz, T. (2015). The Disruptive Effect of Open Source Startups.

Ven, K., Verelst, J., and Mannaert, H. (2008). Should you adopt open source software? *Software, IEEE*, 25(3):54–59.

Very, P. and Schweiger, D. M. (2001). The acquisition process as a learning process: Evidence from a study of critical problems and solutions in domestic and cross-border deals. *Journal of World Business*, 36(1):11–31.

von Hippel, E. and von Krogh, G. (2003). Open Source Software and the âPrivate-Collectiveâ Innovation Model: Issues for Organization Science. *Organization Science*, 14(2):209–223.

von Krogh, G., Haefliger, S., Spaeth, S., and Wallin, M. W. (2012). Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *MIS Quarterly*, 36:649–676.

von Krogh, G. and Spaeth, S. (2007). The open source software phenomenon: Characteristics that promote research. *The Journal of Strategic Information Systems*, 16(3):236–253.

von Krogh, G., Spaeth, S., and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32:1217–1241.

von Krogh, G. and von Hippel, E. (2006). The promise of research on open source software. *Management Science*, 52(7):975–983.

Waters, R. (2015). New breed of open source start-ups eye rising revenues and IPOs.

Weikert, F. and Riehle, D. (2013). A Model of Commercial Open Source Software Product Features. *Software Business. From Physical Products to Software Services and Solutions SE - 10*, 150:90–101.

West, J. and Gallagher, S. (2006). *Patterns of Open Innovation in Open Source Software*, volume pp.

West, J. and Lakhani, K. R. (2008). Getting Clear About the Role of Communities in Open Innovation. *Industry and Innovation*, 15(2):223–231.

West, J. and O'Mahony, S. (2004). Contrasting Community Building in Sponsored and Community Founded Open Source Projects. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 00(C):1–10.

West, J. and O'Mahony, S. (2008). The Role of Participation Architecture in Growing Sponsored Open Source Communities. *Industry & Innovation*, 15(2):145–168.

Wilson, R. (2005). Open Source Development - An Introduction To Ownership And Licensing Issues.

Wilson, R. (2010). So What Is Open Innovation ? How Does That Relate To FOSS ?

Wood, D. (2005). Open Source Software Strategies for Venture-Funded Startups.

Xu, J., Christley, S., and Madey, G. (2005). The open source software community structure. *North American Association for Computational Social and Organizational Science (NAACSOS 2005)*.

Xu, J. and Madey, G. (2004). Exploration of the Open Source Software Community Contact.

Ye, H. (2008). *Classifying Venture Capital Backed Open Source Software Startups Using Publicly Available Information*. PhD thesis.

Zammuto, R. F., Griffith, T. L., Majchrzak, A., Dougherty, D. J., and Faraj, S. (2007). Information Technology and the Changing Fabric of Organization. *Organization Science*, 18(5):749–762.

Zhou, M. and Mockus, a. (2014). Who Will Stay in the {FLOSS} Community? Modelling Participant's Initial Behaviour. {*IEEE*} *Transactions on Software Engineering*, PP(99):1.