Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)

The role of firm employees' participation for the success of firm-initiated open source projects

Betreuer: Univ.-Prof. Dr. Frank Thomas Piller
Beratungsassistent: Prof. Dr. Christoph Ihl

vorgelegt and der
Rheinisch-Westfälischen Technischen Hochschule Aachen
- Lehrstuhl der TIME Research Area –

von: Philipp Ständer
Kempener Str. 121
50733 Köln
Matr.-Nr.: 336329

Abgabetermin: 30.03.2016

Acknowledgement

I would like to thank Prof. Christoph Ihl, Christian Maschner, Fabian Mies, Miriam Schäfer, Veit-Henning Köster, David Langer, Jan Krems and Alexander Esser for their suggestions and critical review.

Last but not the least important, I would like to thank my parents and my family for everything.

Contents

1 Introduction

Open source software (OSS) is gaining more and more importance as innovation factor in software businesses. Also the number of firms who have initiated open source (OS) projects has increased quite substantially over time. Therefore, OSS has reached increasing attention by scholars in fields of organization and open innovation over the last past three decades. There are various popular examples of OS projects which demonstrate the feasibility of running businesses with or on initiated OS projects (Google, Twitter, facebook and Microsoft are just a few popular examples). But not only big companies are actively involved in OSS development. A variety of small and medium-sized enterprises have also initiated and maintained OS projects successfully. Some of those OS projects are also part of their core business.

OS communities are technical communities and can help firms to develop and deploy new technical innovations (Rosenkopf et al. (2001), Rosenkopf and Tushman (1998), Simcoe (2006); Fleming and Waguespack (2007)). Thus, firms have credible reasons to collaborate with OS communities. Nonetheless firms that are involved in OS projects have to face various difficulties. One important point is that firms have to care about their relationship to the OS community and have to find a fair way to join commercial (of the firm) and non-commercial interests (of the OS community) without reducing the efficiency and the quality of innovation. Because of this symbiotic relationship between the firm and the OS community (Dahlander and Magnusson (2005)) organizational obstacles have to be solved by a firm if it wants to collaborate with autonomous OS communities (Piezunka and Dahlander (2013)) or sponsored OS communities (West and O'Mahony, 2008).

This leads to the question to what extent firm's internal resources affect external resources. If we define external contributors as individuals who contribute their own ideas (Piller and Walcher (2006)), solutions (Jeppesen and Lakhani (2010)), knowledge (Laursen and Salter (2006)) or innovations (Urban and Von Hippel (1988)) we can regard external developers, who are external contributors, as a resource of knowledge. Thus, external developers will affect the capability of firm's absorptive capacity of prior related knowledge (Cohen and Levinthal (1990)). Furthermore, we can assume that over time external developers may have similar project-related knowledge as firm employed developers (Piezunka and Dahlander (2013)) and do represent their own interests whereas firms employed developers (sponsored contributors) represent the firm' s interests (Dahlander and Wallin (2006)).

As Dahlander and Magnusson (2005) found out, "OSS firms can use symbiotic, commensalistic, or parasitic approaches for interrelating to their communities" and by using the "symbiotic approach, firms have more possibilities to influence the (OS) community" (Dahlander and Magnusson, 2005, p.491-492). A new study from 2014 shows that "OSS projects with large peripheral developer participation [1] are those that are likely to be innovative and offer high-quality" (Krishnamurthy et al., 2016, p.40:22) and that "presence of feature requests is positively associated with peripheral developer participation" (Krishnamurthy et al., 2016, p.40:23).

This study will focus on the impact of firm developers' commitment on external developers' commitment and the long-term success of the projects. Therefore 58 firms with more than 3000 published projects (written in the 10 most popular programming languages) hosted on GitHub [2] will be studied. We will measure code contribution and communication actions (participation on issues) referred to (1) the influence of firm employed developers on external developers (and v.v.) and (2) the final success / acceptance of a project on the platform. Thus, the main questions are [3]:

- To what extent does firm employed developers' commitment encourage external developers' commitment?
- Do external developers more often contribute source code if firm employed developers contribute proportionate more than usual?
- Do external developers participate more actively in issues if firm's developers do as well?
- Does participation in the beginning affect the later success of firms' initiated projects?

The empirical findings mainly confirm all hypotheses to the effect that participation of firm employed developers has a significant positive impact on external commitment and long-term success of projects. Further details, explanations and interpretations of the empirical findings can be found in chapter 5 on page 57.

[1] peripheral developers = external developers

[2] the worldwide largest and most popular open source hosting service (see Hyett (2011); Metz (2015); GitHub (2016b))

[3] the hypotheses are formulated in chapter 2.15 on page 18

2 Theory and Hypotheses

2.1 Introduction and Prior Research

There are several reasons why firms initiate open source (OS) projects. As West (2003) noticed, firms do publish source code in order to get their product widely adopted, to increase the likelihood to attract developers [4] and to achieve faster technological development (Lerner and Tirole (2002)). Beside that, all OS projects can be termed as collective action model which applies to the provision of public goods, where a public good is non-excludable and non-rival (Olson, 2009, p.14). Thus, OS projects can be regarded as foundation for a novel, private-collective model for the motivation of innovation (Hippel and Krogh (2003)). This confirms the assumption that any interested person can voice their opinion when organizations search for innovation by soliciting suggestions from externals (Alexy et al. (2012)). Most OS projects which are initiated by firms are also part of their business. Hence, they utilize resources as knowledge and ideas to maximize their value creation and profit in the long run (Dahlander and Magnusson (2005)).

Because OS projects of firms depend on the symbiotic relationship between internal end external developers, externals' suggestions can lead organizations to novel, useful, and actionable ideas (Hill and Birkinshaw (2009)), develop new concepts (Katila and Ahuja (2002); Jeppesen and Lakhani (2010); Shane (2000)) or find new markets for their technologies (Gruber et al. (2008); Shane (2000)). So the "wisdom of the crowds" (Surowiecki (2005)) depends on external suggestions which facilitates research on external sources of innovation (Dahlander and Magnusson (2005)) as on open innovation (Chesbrough (2003)), crowdsourcing (Afuah and Tucci (2012)) and user-based innovation (Von Hippel (1986)).

Most initiatives to engage external contributors do fail as Dahlander and Piezunka (2014) showed in their study. They developed arguments about what increases the likelihood of getting suggestions from externals by proactive and reactive attention to suggestions. They say that firms and communities have divergent rationales for existing which causes problems on interaction. Moreover, they found out that early organizations' engagements weigh more in the early stage and that firms should pay attention to new contributors. Thus, they suggest to keep entry barriers low since it results in a lower threshold for participation.

Piezunka and Dahlander (2013) developed a theory about the structures that are build around a suggestion after it is posted. They say the more suggestions are related to each other, the greater is the demand of externals for a suggestion. Additionally, the more a suggestion is debated by externals, the higher the likelihood an organization will attend to it. On the other side, the greater the diversity

[4] first-mover advantages

of suggestions and the more suggestions competes with each other, the lower the likelihood that an organization will attend to it.

One of the initial ideas of the OSS community is to work for non-commercial reasons. This means that firms have to deal with the term openness (Dahlander and Gann (2010)) and have to accept that external developers are free to join and work on informal relationships (Dahlander and Magnusson (2005)) whereas firm-based software creation is normally restricted to relations within the firm. To use the taxonomy by Feller et al. (2002), we can distinguish between economic, social and technological motivation factors. As Bonaccorsi and Rossi (2006) found out, firms are driven by economic and technological factors rather than by social motivation factors. Contribution of external developers depends on both extrinsic and intrinsic motivation [5] (Hertel et al. (2003); Hars and Ou (2001); Lakhani et al. (2002)) or even by intellectual challenges (Raymond (2001); Hertel et al. (2003)). This supports the assumption that users often find solutions to their own problems and are willing to share them when the marginal cost of sharing is low (Von Hippel (1976)). Furthermore, users in OS projects are not protected by intellectual property rights (Waguespack and Fleming (2004)), which is converse to intellectual property handling of traditional (in-house) software development by firms.

West and O'mahony (2008) analyzed how corporate sponsorship influences OS communities. They identified three design dimensions for corporate sponsorship when designing OS communities:

1. intellectual property rights
2. development approach, and
3. model of community governance

Overall they came to the conclusion that openness of sponsored OS projects is more likely to offer transparency than accessibility, which has implications for their communities' growth.

Dahlander and Gann (2010) analyzed also the relationship between OS software firms and communities. They figured out that firms who depend on a symbiotic approach have subtle means of control to influence the community and are confronted with challenging managerial issues. For instance, if a firm is well-known and respected in the community, they have more influence on the development activities performed in the community compared to less well-connected ones. They distinguish five mechanism of subtle means of control (Dahlander and Magnusson, 2005, p.489):

1. devote firm employees to work with and in the community
2. reputation of firm employees in the community
3. fringe benefits
4. interaction tools which offers communication channels for the community

[5] which is a social motivation

5. provide interesting tasks ('selling' development tasks)

A greater possibility of influencing the community might result in several benefits, but may also emerge the following managerial issues (Dahlander and Magnusson, 2005, p.489):

- respecting the norms and values of OSS communities
- using licenses in a suitable way
- attracting developers and users
- dealing with the resource consumption involved in community development
- aligning different interests about the nature of work
- resolving ambiguity about control and ownership

2.2 Incentives of Firms to use Open Source

Organizations use OSS to reduce costs and standardization expenses (Wichmann (2002); Ghosh (2007)). Beside that, OSS is less costly because there are no license costs to the contrary of proprietary software (Hawkins (2004) and Wichmann (2002)). Software products create costs for licensing, customization, maintaining and training (Weiss (2005)), but as Wichmann (2002) found out, the licensing costs are normally the main reason for managers to use OSS.

2.3 How can Success of Firms' initiated Open Source Projects be measured?

As mentioned before, firms are driven by economic and technological factors (Feller et al. (2002)) whereas developers are more driven by technological and particularly social motivation factors (Bonaccorsi and Rossi (2006)). So in between we have the technological motivation factor which motivates firms and developers. So we can roughly divide success into economic and social success [6]. Here, social success is popularity in OS communities (i.e. many developers pay attention or actively participate on the project). As shown in a previous research, the number of participants depends on the complexity of the OS project (Piezunka and Dahlander (2013)). In this study we will concentrate on projects' social success, which is closely related to technical success as well. Social and technical success can be analyzed in a more reliable way since most firms simply do not release any business related numbers on their projects for external research.

To compare the impact of internal and external developers on a firms' initiated OS projects we have to analyze the sort and rate of contribution. We can narrow down the process of software development to:

1. participation by submitting source code, and

[6] economic success of a project is not explained in detail, since it is not subject of the study

2. taking part on discussions and support channels [7]

We can regard 1.) as technological and 2.) as social contribution. Social contribution can be proceeded in submitting and discussing ideas or solving problems on forum threads. Thus we should be able to derive a measure of success by these two participation possibilities.

We can request different social interactions to measure the social success of projects through meta data which will be explained in detail in chapter 3.4.

2.4 Open Source Software and Free Software Licenses

The widely used term ”Open Source” must be distinguished in ”free software” and ”open source software”. The term ”free” means zero, as in free of charge and free of defects (Licensing (2004)). The term also guarantees the freedom to run, copy, distribute, study, change and improve the software (Foundation (2016)). Thus, free software must be OS, otherwise you are not able to study and change the software. Whereas the term ”Open Source” just implies that you are able to read the source code but without necessarily the previous mentioned rights to distribute, run and change the software (Stallman (2014)).

Three groups of OS licenses are available (Laurent (2004)): *restricted*, *less restrictive, more permissive* and *very permissive* (see chapter 3.9 for details). Altogether having 10 different licenses:

- The MIT (or X) License
- The BSD License
- The Apache License v1.1 and v2.0
- The Academic Free License
- GNU General Public License
- GNU Lesser General Public License
- The Mozilla Public License 1.1 / MPL 1.1
- The Q Public License
- Artistic License (Perl)
- Creative Commons Licenses

Each license describes specific ideas and rules of contribution, usage, copyright and copyleft.

Regarding to software licenses, *copyleft* ”grants everyone the right to use, modify and distribute the program on the condition that the licensee also grants similar rights over the modifications that was made” (Mustonen, 2003, p.101). As Stallmann (2015) states, the copyleft in OS license keeps the freedom of the initial software project for distribution.

[7] sometimes also called *interaction tools*

According to Black Duck (2015) nowadays the following 3 most used licenses in OS projects are:

- MIT (25%)
- Gnu General Public License (GPL) 2.0 (22%)
- Apache License 2.0 (16%)

The distribution is close in line with Bonaccorsi and Rossi Lamastra (2003), where copyleft licenses in the OS community are dominant but preferred to be mixed (dual-licensing) with other non-copyleft (i.e. more permissive) licenses. This is used by companies to achieve competitive advantages and / or to protect (patented) intellectual property (IP) by keeping parts of the source code closed (e.g. *Google* and *Oracle* on *Android*: Amadeo (2013); Gal (2016); Kuhn (2016)).

In the following we will pool ”free open source software” (FOSS) and ”open source software” to OS (Open Source) and OSS (Open Source Software) since the subject of the study is the contribution of firm employees and external developers rather than the philosophy and legal aspects of software distribution. The considered software projects differ in their licenses regarding to their field of application, company philosophy and long-term strategy. The segmentation of firms' licenses will be considered in chapter 3.9 on page 28.

2.5 Open Source Software as Public Good

OSS in this context must permit non-exclusive commercial exploitation of the licensed work, make the work's source code available and must permit the creation of derivative works from the original work (Laurent (2004)). OS licenses must allow to read, modify, execute and distribute the software which creates a public good. Thus, OSS is a public good with non-revival and to some extend non-excludable attributes ((Eilhard, 2009, p.6), Lerner and Triole (2000)).

As mentioned before, the introduced licenses do differ in terms of their intention of usage and modification. The OS licenses ensure that the rules of the OS community apply on every user of the OSS (Eilhard (2010)). Because OSS is a non-exclusive and non-excludable public good, everyone has the right to use the software. Thus, many actors can benefit by the contribution from a few actors. As stated by the private-collective model of innovation (Hippel and Krogh (2003)) economic parties invest private resources to produce a public good. The private investment expects a return of investment. For that, knowledge spillovers will tried to be avoided and society may grant access with patents, copyrights and trade secrets (Osterloh and Rota (2007)). On the other side the collective action model applies to the production of a public good by having a central agent (e.g. a government subsidy program) to produce a public good.

2.6 Open Source Business Models

In the last three decades OSS has shifted from the hacker scene to the mainstream (Fitzgerald (2006)) and became more relevant as business opportunity for companies.

Because OSS can be regarded as a public good, companies which business rely on their initiated OS projects have to find a way to work profitable. According to Popp and Meyer (2010) the following possibilities exist:

- leverage the OS community as supplier, development, sales, maintenance or support resource
- sell software which is based on the OSS
- provide services for the OSS to the client

As Example: The *Android operating system* was initiated by *Google* in 2008 (Dan Morrill (2008)) and is one example of successful initiated OS projects by a company. According to Bloomberg and Reuters (Joel Rosenblatt (2016); Steve Trousdale (2016)) *Android* generated 31 Billion USD revenues for Google since it's launch. As stated in table 1, *Android* is by far the most installed operating system on shipped devices in 2015. This example demonstrates that firm's initiated OS projects can generate relevant revenue - even for larger companies. Beside that, *Android* is developed by a community of firm employed developers and external developers [8].

	Operating System	2014	2015	2016
1	Android	1,156,111	1,156,111	1,156,111
2	iOS / MacOS	262,615	262,615	262,615
3	Windows	333,017	333,017	333,017
4	Others	626,358	626,358	626,358

Table 1: Share of Operating Systems on worldwide shipped devices (thousands of units); Shipments include mobile phones, ultramobiles (including tablets) and PCs [10]

2.7 Open Source and Software as a Service (SaS)

Many companies are implementing OSS as a part of their software service nowadays. This means, software services are usually offered through a platform (which is the internet by default). The terms *Software as Service (SaS)*, *Cloud Services* and *Cloud Computing* imply that the service is only available

[8] *Android* projects are not part of the study since *Android* is not classified as commercial organization. However, the share on core modules by firm employed developers (i.e. *Google* developers) is roughly estimated between 74 % - 89 % (see `https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/csv/examples/android_ratios.csv` for detailed information)

[10] Source: Gartner (2015)

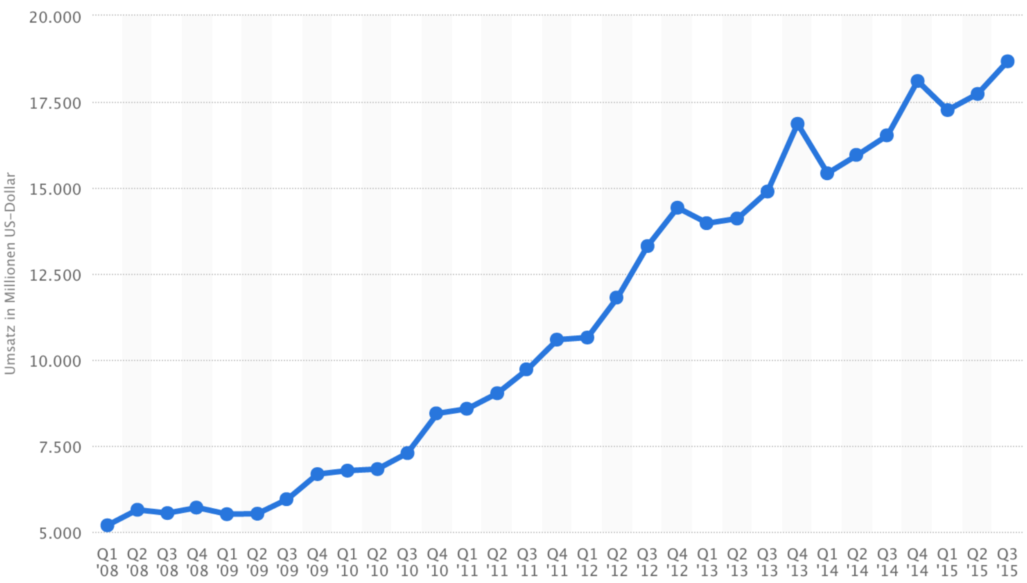

Figure 1: Revenue of Google in Mio. USD, © 2016 Statista, Source: `http://de.statista.com/statistik/daten/studie/154635/umfrage/umsatz-von-google-quartalszahlen/`

via remote rather than installed as local software. Thus, SaS is rather rented than licensed (Buxmann et al. (2008)) and, compared with traditional locally installed software, profit is generated by leasing the software and service to the user.

The range of open and closed source code behind software services differs widely. Server side components and frontend modules often rely on OSS technologies [11]. Because most SaS-APIs only allow receiving and returning (processed) users' data, those closed server-side processes make studying executed code hard or even impossible. Thus, the user is neither able to determine what the software really does nor able to change (and redistribute) it (Richard M. Stallman (2010)). Therefore, this approach is widely against the initial thoughts of free software.

Moreover, cloud computing aims to force people to buy lock-in systems that will increase costs over time (Bobbie Johnson (2008)). To fill the gap of lacking copyleft on SaS, the free software foundation published the GNU Affero GPL in 2007 (Foundation (2007)) which demands a download link to the source code on hosted services running the affected software.

Beyond closed-source SaS-providers other companies like *GitLab* (Hall (2015)) or *ownCloud* (Vaughan-Nichols (2014)) publish their entire server- and client-side software as OS. Profit is generated by providing enterprise editions (Hall (2015); ownCloud (2016); (Van Baarsen, 2014, p.148)) [12] for instance.

[11] for example: many services using OS databases like MariaDB or PostgreSQL for data storage

[12] the enterprise edition does not necessarily differ in features rather than then in support features for the service itself

Other companies publish and maintain specific libraries or components as OS [13]. The latter is common practice for (larger) technology companies like Google, Microsoft, facebook or Amazon.

2.8 Open Source and Open Innovation

OSS can be regarded as process innovation (Bonaccorsi and Rossi (2003)) with a connection to open innovation. Information to solve a problem can be sticky and opening innovation can help to solve the problem of sticky information (Von Hippel (1994)). Open Innovation implies that valuable ideas may come from the inside or outside of a company (Chesbrough, 2006, p.43f) and let to the assumption that exploration and exploitation of internal and external ideas lead to better innovation. Using the term ”openness” according to Chesbrough (2003), ”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 firms look to advance their technology”. Thus, companies can turn (technology regarded) ideas from inside and outside the company to profit (Dahlander and Gann (2010)).

For the need of companies to connect ”Open Innovation” with ”Open Source”, Dahlander and Gann (2010) give the following reasons:

- Social and economic changes in working patterns:
    - today employees are seeking more often portfolio careers than a single employer for lifetime
    - OS commitment builds a better portfolio
- Globalization:
    - allows increasing division of labor
    - enables working on distributed channels on source code
- Intellectual property rights (IPR), venture capital (VC) and technology standards allow for organization to trade ideas
- Software and technology:
    - changes the minimum efficient scale of production
    - allows new ways to collaborate
    - coordinates across geographical distances

As follows, *Open Source* and *Open Innovation* are related terms since programming is a kind of innovation by creating new products (software) to solve (technical) problems.

[13] for example: facebook initiated React (Occhino (2015)) and uses React Native (Facebook (2015)) in multiple production apps (Facebook (2016))

2.9 Open Source Collaboration of Firms with external Developers

As stated in chapter 2.8, OS developers are innovating users (Von Hippel (2005)): they evaluate, change and improve source code for their own purpose and solve problems by creating software for specific use cases. Usually they publish their innovations without claiming intellectual property rights and are driven by extrinsic and intrinsic motivation (Hertel et al. (2003); Hars and Ou (2001); Lakhani et al. (2002)).

Firms' investment in OS software and communities differs from the interest of (external) OS developers since firms expect a return of investment. Their intention of investment is mainly driven by economic, technical and strategic reasons (Wichmann (2002); Henkel (2006)). If firms acting in OS sections, conflicts with OS community members may occur due to different interests of contribution (see Koçulu (2016) as example), license permissions (Hars and Ou (2001)) and commercial intention. Forms of commitment by firms can be participating in software development, communication channels or in general by creating network effects through an "architecture of participation" (O'reilly, 2007, p.22).

But why do firms initiate OS projects and do maintain them even if they do not belong to their core business?

West and Gallagher identified the following benefits when organizations do sponsor OS projects (West and Gallagher, 2006, p.13):

- helping to establish their technology as de facto standards (reduces the likelihood of having to re-implement other products to conform to competing standards)
- attracting improvements and complements that make the technology more attractive
- together, the innovation and complements enable the sale of related products
- generating mindshare and goodwill with the same audience that includes the potential customers for these related products

As Tsay et al. (2014) found out, contributions on GitHub by submitters with high status in the community and a stronger social connection to the project have a higher chance to get accepted. Besides, discussions of contributions have a social and technological factor. Contributions on established projects are harder to be accepted (Tsay et al. (2014)) which may occur trough higher communication costs. In addition, employed developers who work for longer periods on firms' initiated OS projects have a higher reputation in the community, more power of control on contributions and larger influence of project's (technical) design and advancement.

Consequently, firms may receive indirect (nonmonetary) return of investments if they initiate and maintain projects that are not primary relevant for (monetary) revenue. Better reputation and establishment of software technology with more success are such returns as well as coming across new potential employees [14].

2.10 GitHub as Host for observed Projects

GitHub is the largest social coding repository and hosts over 35 Mio. repositories by 14 Mio. people (GitHub (2016b); Metz (2015)). It was successfully used as data source in several other studies before (Vasilescu et al. (2013); Terrell J (2016); Dabbish et al. (2012); Thung et al. (2013)) and allows to query project related metadata. Moreover, all popular firms do publish their OS projects mainly on GitHub because it offers extended enterprise support (Holman (2011)) [15] beside it's popularity for free and public accessible projects.

2.11 Increasing commitment of Companies to Open Source

2.11.1 As an example of developing a common approach: Microsoft and Open Source

Between the late 90s and the beginning of the century Microsoft was widely known for it's rigorous distaste towards OS licenses, OSS and especially the *GNU/Linux* system. At this time Microsoft's core product and cash cow was Microsoft's operating system Windows (*MS Windows* for clients & *MS Windows Server*) and *Microsoft Office* (Reifman, 2007, p.2). In other words, Microsoft got threatened by OSS. This was regarding particular Linux as client and server operating system and *OpenOffice* (nowadays widely known as fork *LibreOffice*). This may be one of the reasons why former Microsoft CEO *Steve Ballmer* regards *Linux* (and it's GPL license) in 2001 as cancer that attaches itself in an intellectual property sense to everything it touches (Greene (2001)).

Since user interaction with the operating system shifted from local machines (i.e. locally installed software on stationary computers) to the internet (using software services in the web browser and on mobile devices, as described in chapter 2.7), Microsoft couldn't defend its dominant position and lost market shares in the last decade (Weinberger (2015)).

By failing to defend the dominant position of their proprietary software, *Microsoft* started to change their strategy over the last decade and increased their commitment in OSS. As *Sam Ramji* already stated in 2008 (Foley (2008)), *Microsoft* has realized to include *Linux* (i.e. other OSS and OS based services) as a vertical product integration in a heterogeneous environment on top of *Windows*. This strategy of mixing OSS with proprietary software (Hanna (2010)) is common practice in technology companies nowadays.

[14] external developers are potential employees since they have project related knowledge and matching skills

[15] against payment of a fee, `https://enterprise.github.com/features`

In 2014 *Microsoft* announced to publish their programming language and enterprise framework *.NET* as open source (Microsoft (2014)). This step was just one of their latest upcoming OS commitments and facing towards the OS community (see Seth (2016) as one of the latest examples).

Finally, *Microsoft* moved in 2015 most of its OS projects from codeplex (their own OS hosting service) to *GitHub* (Uhlenhuth (2015)). With this step *Microsoft* tries to reach the vibrant OS community (of *GitHub*) and indicates that they want to be an active part of it. Currently, Microsoft's GitHub account contains more than 370 projects with 1,138 organizational developer profiles (external developers are excluded) [16].

2.11.2 GitHub's Atom Editor and Microsoft's Visual Studio Code as example for technology exchange between firms and collaboration with the OS community

In 2011 *GitHub Inc.* [17] employee *Corey Johnson* and *GitHub* founder *Chris Wanstrath* started building the *Atom editor* (hereafter referred to as *Atom*) as closed source project inside the *GitHub* company (Ben Ogle (2016)). *GitHub* conceived the *Atom* in February 2014 to the public (Sobo (2014b)). In May 2015 they published its source code (Sobo (2014a)) under the *MIT License*. The idea of the *Atom* is to be a "zero-compromise combination of hackability and usability" and therefore a rival software product to other closed source / commercial editors like *Sublime*, *Apple's XCode* and *Microsoft's Visual Studio*. The feedback of the OS community was widely positive and *Atom* gained quickly considerable attention (Serdar Yegulalp (2015) and Dohm (2016)).

One key feature of *Atom* is its modularity, which allows customization through "packages" (i.e. "plugins"). Each plugin is customized for specific software development needs. The format of packages and themes was previously introduced by *Sublime*, which holds a large repository of plugins. Thus, all *Sublime* plugins can easily be converted to *Atom* [18]. In addition, the introduction of *apm* (*Atom Package Manager* [19] as central software provider for those plugins enabled a new software ecosystem for the editor and accelerated the acceptance of converting, building and deploying new plugins for *Atom* by the community.

A year after the initial release of *Atom*, *Microsoft* announced its free and open source editor *Visual Studio Code* (*VSC*) (Frederic Lardinois (2015)). *VSC* is a code editor with similar look-and-feel, features and ideas of *Atom*, but initiated by *Microsoft*. Moreover, *VSC* is based on *Electron* (GitHub

[16] Retrieved 9th January 2016

[17] to "distinguish" between the company and the community, we will use *GitHub Inc.* for the company; but the transition is seamless sometimes

[18] `https://discuss.atom.io/t/convert-sublime-grammar-to-atom-grammar/14843`

[19] `https://github.com/atom/apm`

[20] Source: `https://atom.io/packages`, `https://packagecontrol.io/stats`, `http://colorsublime.com`

	Themes	Packages	Users	Authors	Downloads
Atom	1,008	3,496	-	-	32,79M
Sublime	295	3,426	4,6M	2,477	-

Table 2: Available Plugins for *Atom Editor* and *Sublime Editor*; Retrieved on: 21st January 2016 [20]

(2016a)) and *React* - both are OS technologies by *GitHub Inc.* and *facebook* respectively and core technologies of *Atom* as well.

This makes *VSC* to a rival product with the potential to substitute *Atom*. Using *Electron* and some other core libraries as foundation, *VSC* guarantees interoperability of *Atom's* plugins on its editor. In November 2015 *Microsoft* finally published *VSC* as open source likewise (Microsoft (2015)).

GitHub Inc. and *Microsoft* are aware that the editor gets only established with a variety of plugins (see table 3.16) and with an active community. The reason is that establishing software standards is to some degree a Winner-Take-All market. Once a standard is established in the community (here an editor with a specific API and package format) the development of "plugins" will be successful, too.

This example demonstrates how two firms published initially closed source developed software as open source: *Atom* (*GitHub Inc.*) and *VSCode* (*Microsoft*). Both editors are heavily using OS libraries as technical foundation.

Altogether, Microsoft's editor is a good example of

- using OS technology of competitors (*GitHub Inc.*, *facebook* and *Instagram*)
- as foundation to build OSS for the community
- with plugins and code contribution by the community and developers by competitors

Interpretation: The reason why both firms decided to publish their projects as open source may be that the OS developers are their target customers (Sobo (2014b)), community members and potential employees. By providing technologies for the community, both firms will improve their reputation in the developer community. Furthermore, upcoming projects will be easier accepted by external developers, more likely successful and establishing software technology in future might be more feasible in a smaller amount of time.

A further analysis of "Atom" and "VSC" ("Microsoft" and "GitHub Inc." respectively) can be found in chapter 3.16 on page 37

2.12 Forms of Participation on GitHub Projects

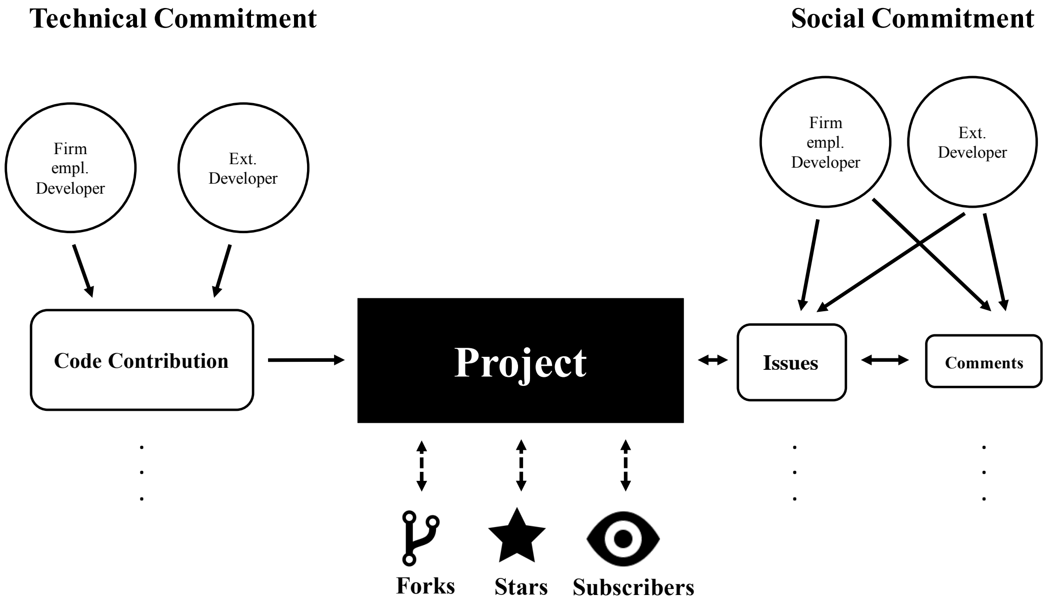

Figure 2: Technical and social Commitment in GitHub Projects

GitHub allows direct and indirect participation. This implies that every user can participate with minimum action (indirect) and active contribution (direct).

The following technical and social participations are possible:

- Code Contribution (direct)
- Creating Issues (direct)
- Participation by commenting on Issues (direct)
- *Fork*, *Subscribe* and *Star* project (indirect)

Use and interpretation of these attributes will be explained in chapter 3.4 on page 23.

2.13 Linear and Logistic Regression Models

The empirical analysis uses *Linear & Logistic Regression Models.* According to Seber and Lee we can describe the linear models of the study as follows (Seber and Lee, 2012, p.35):

$$
\begin{aligned}
\mathrm{lm}_j : y_{\mathrm{j}_i} \qquad & = \beta_{j_0} + \beta_{j_i} x_{j_i} + \cdots + \varepsilon_{j_i} \qquad (1) \\
& x : \textit{explanatory variable} \\
& \varepsilon : \textit{fluctuation} \text{ or } \textit{error} \\
& \beta : \textit{unknown parameter} \\
& \mathrm{lm} : \textit{linear model} \\
& \forall i \in \textit{observations} \\
& \forall j \in \textit{linear regression models}
\end{aligned}
$$

The independent and dependent variables will be different regarding to model assumptions. In particular *Ratio* (share of contribution by internal developers) and *Age* will be set as independent variable, *share of contribution by external developers* and *Top Project* set as dependent variable. *Firms* and *Programming Languages* will be used as dummy variables in some models.

Because *Top Project* is $\in \{0, 1\}$, the *logit model* is used in this case (Hilbe, 2009, p.23):

$$
\mu_{j_i} = \frac{1}{1 + e^{-x_{j_i}\beta_{j_i}}} \qquad (2)
$$

$$
\begin{aligned}
& \mu : \textit{fitted value} \\
& \forall i \in \textit{observations} \\
& \forall j \in \textit{logistic regression models}
\end{aligned}
$$

2.14 Fitting Linear Mixed-Effects Models

To consider effects within *Firms* and effects within *Programming Languages* some analyses will be handled additionally by *Fitting Linear Mixed-Effects Models* (Bates et al., 2015, p.13): [21]

[21] as mentioned in (Bates et al., 2015, p.1), ”mixed effects” denotes a model that incorporates both fixed- and random-effects terms

$$\eta_j = X\beta_j + Z\gamma_j + \varepsilon_j \tag{3}$$

η : *known vector*

γ : *unknown vector of random effects*

β : *unknown vector of fixed effects*

ε : *unknown vector of random errors*

$\forall j \in$ *linear mixed-effects models*

2.15 Hypotheses

Derived from motivation, theory and previous studies [22] we formulate the following hypotheses:

- **H 1** Firm employees' participation affects the participation of external developers
    - **H 1.1** If firm employees contribute source code more often, external developers do as well
    - **H 1.2** If firm employees participate more often on issue threads, external developers do as well
- **H 2** Firm employees' participation (in the beginning) affects the (later) success of firm-initiated open source projects
    - **H 2.1** If firm employees contribute source code more often in the beginning, the project gets more successful
- **H 3** If firm employees' contribution share is higher overall the more likely the project is popular

The analysis will contain code contribution (technical commitment) and participation on issues and comments on issues (social commitment) [23]. Contribution will be measured with a "standardized contribution share" in respect of (social) success metrics and relations between firm employed developers and external developers. All hypotheses will be verified with firms' publicly available OS projects on GitHub.

[22] Hill and Birkinshaw (2009); Dahlander and Piezunka (2014); Piezunka and Dahlander (2013); Dahlander and Magnusson (2005); Alexy et al. (2012); Tsay et al. (2014); West and Gallagher (2006); Krishnamurthy et al. (2016)

[23] whereby issue participation can also be technical in some cases

3 Data

In the following chapter, the collection, processing and validation of the relevant data will be explained. The empirical data is collected from public projects on GitHub until February 2016 [24]. Repositories' related activity data is analyzed from 2011 to 2015.

Beside the selection of languages (see chapter 3.2) and firms (see chapter 3.6) all analyzed observations are unfiltered [25] and contain the full available range of public available firms' OS projects.

Git is an OS distributed version control system. It is published and developed by the *Linux* development community (including Linus Torvalds, the creator of *Linux*) since 2005 (Chacon and Straub, 2014, p.31).

3.1 Data Sources and Open Data Approach

All analyzed data is collected from the following public and free available data sources:

GitHub API: Project data, issues and issue comments
Git repository: Code contribution
Glassdoor API: [26] Firm and employee's ratings
GitHub Archive: [27] Time-referenced data for user events (time period 2011 - 2015)
GHTorrent Project: [28] User data (email, location, company and full-name)
LinkedIn Websearch: Manual classification for developer employments

External data is received as JSON files (Bray (2014)) and converted to the CSV format (Shafranovich (2005)) for better processing possibilities in *R*. All data can be downloaded and observed through a git repository [29].

[24] collected during the period from December 2015 to February 2016

[25] unless stated otherwise

[26] `https://www.glassdoor.com/developer/index.htm`

[27] `https://www.githubarchive.org/`

[28] `http://ghtorrent.org/`

[29] `https://git.zeitpulse.com/philipp/masterthesis-data/tree/master`

3.2 Selection of Programming Languages

To specify a proper selection of relevant OS projects we need to define a list of popular programming languages.

The most used programming languages on *GitHub* [30] are listed in table 3 (GitHub and La (2015)). The figures reflect the overall ranking of programming languages in OS (Cade Metz (2015)) and integrates close enough to the *TIOBE index* (TIOBE (2015)) and *Programming languages used in most popular websites* on Wikipedia (2015). Excluded languages are *Shell* and *CSS* since *Shell* is a UNIX command line interpreter and neither used for building regular web services nor for building software in the proper sense [31] and *CSS* is a markup language and not a programming language.

The replacement of *Shell* and *CSS* with the *GO language* seems to be reasonable (see Paul Krill (2015), Schaaf (2014) and Finnegan (2015) for possible reasons). The selection excludes the ”rising stars” of OS programing languages *Scala, Hack* and *Swift* since the popularity, spreading and age of those languages is too little for the observation.

The following 10 languages (in alphabetically order, grouped by programming language family) are finally selected:

- C, C#, C++, Objective-C
- Go
- Java
- JavaScript
- PHP
- Python
- Ruby

The exact number of projects according to *Stars* and *Forks* can be looked up in table 3 and figure 13 respectively.

The usage of programming languages in projects differ between firms. Some firms promote their own programming languages (e.g. *Google* uses *Go* / *Microsoft* uses *C#* more often than usual) or use specific languages according to their field of application (e.g. *GitHub Inc.* uses *Ruby*). Figure 3 and 4 illustrate the usage of the 10 selected programming languages in ”Top Projects” and ”Residual Projects” [32] according to firms.

[30] *Visited 12.01.2016:* `https://github.com/search?o=desc&q=stars:%3C1`

[31] `http://www.rpi.edu/dept/arc/training/shell/slides.pdf`

[32] terms will explained on page 25

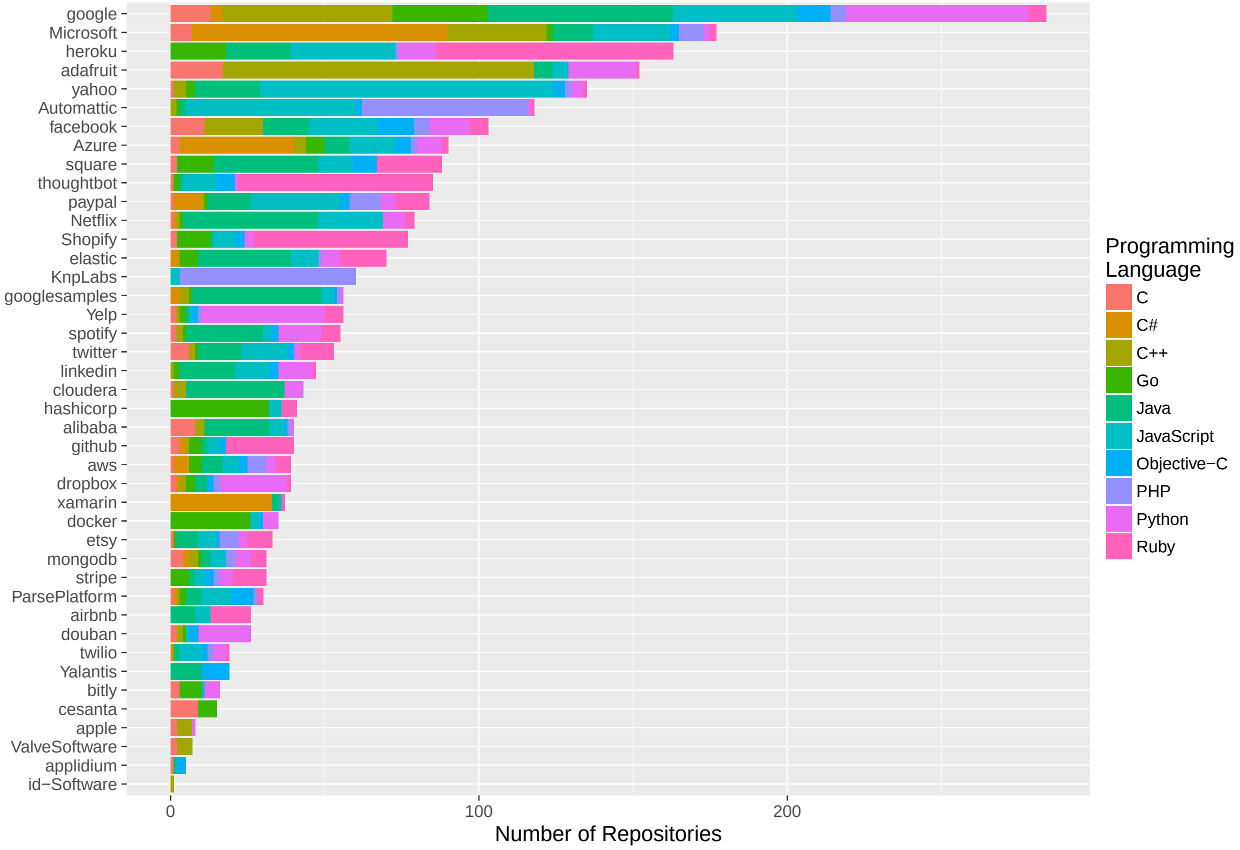

Figure 3: OS Projects of Firms and the Programming Languages they are written in

Programming language	**Stars**	**Forks**
JavaScript	252891	134294
Python	132995	74924
Java	115249	83361
Ruby	105547	55153
PHP	97112	57332
C	59920	36506
C++	53706	33105
Objective-C	44734	24845
C#	35859	22157
Shell	36874	-
CSS	-	21637
Sum	**934887**	**733789**

Table 3: Programming Languages with projects' *Stars* and *Forks* (*Retrieved: 12.10.2016*)

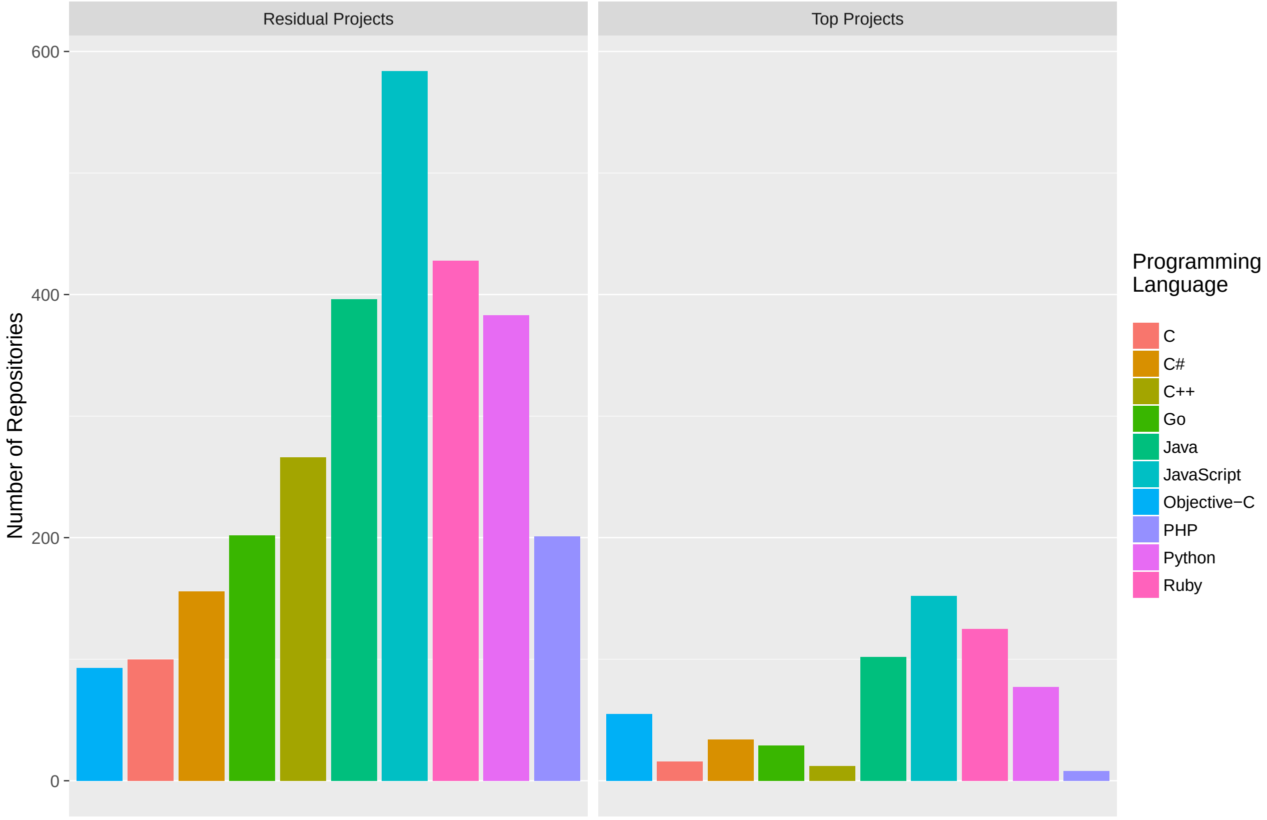

Figure 4: Programming Languages which are used in *Top Projects* and *Residual Projects*.

3.3 Restrictions of Data Collections

All measurements and analyses are made under the following restriction: GitHub repositories' social data (*stargazers*, *forks*, *contributors*, ...) can only be measured with the actual values, i.e. cross-sectional data. For that reason, these attributes are query able through the GitHub API (GitHub (2015b)) with the possibility of sorting. However, querying and analyzing time series of GitHub event data is possible through the GitHubArchive (see Grigorik (2012); Brian Doll (2012); listing 1) having events with the verbs *watch* (mostly called *subscribing*) and *fork*. As mentioned in the beginning, the available time period of user activities is from 2011 to 2015 [33].

3.4 Social-Success-Metrics of git Repositories

A *git repository* (sometimes also called *git project* or *git repo*, where *git* is an optional term) provides the following statistics by default:

- date and time of commits (commits are code contributions)
- name and email of the author (authors are developers)
- time zone location of the contributor (via the date-time property)
- code changes and action [34]

A GitHub project has the following social metrics (terms will be described in the paragraph below):

- Stars / Stargazers
- Subscribers (sometimes also called "Watchers")
- Forks
- Open and closed Issues

Finally, we get the following metrics of interest:

- Number of Stargazers, Subscribers and Forks
- Number of Code Commits
- Number of Contributors
- Number of Issues and Comments on Issues

Stargazers [35] are GitHub users / developers who mark a project with a star (GitHub (2015c)), similar to the ordinary "like button" on *facebook*. The reason may be to mark this as one of your favorite projects or to show the initiator your interest in the project or idea.

[33] GitHub itself exists since 2007 (Weis (2014))

[34] an action may be a regular *code commit*, *code merge* or basic file operation (like rename, delete, copy or create file)

[35] derived "action" is *Star*

Fork is a copy / clone of a git project (GitHub (2014)) containing all previous code contributions until the moment of the fork event. Projects are forked either to collaborate with the origin project or to be resumed as an independent project [36]. The number of forks doesn't explain necessarily the activity of contributors nor the popularity. If you participate on a GitHub project by committing code (called *pull request* [37]) with your GitHub account, this project will be forked into your account by default. The *clone* and *fork* actions are fundamental ideas of the decentralized approach of git (Loeliger (2006)).

Subscribers are users who get notified on important project activities (GitHub (2015d)). The term is similar to *subscribing* on *facebook* or *following* on *twitter*. *Subscribing* may be considered as the most important indicator of active interest in projects (beside code contribution and issue participation).

Issues are communication threads (GitHub and Neath (2007)). Every issue can be *open* or closed and labeled as *feature request*, *bug report*, *idea* or *code proposal* for instance. Issue threads are an exclusive collaboration feature of GitHub [38] and not implemented in git itself.

In terms of **"Forms of Participation"** (see chapter 2.12 and figure 2 respectively) we can roughly classify this (direct and indirect) user actions from "lower participation / activity" to "higher participation / acitivity". *Stars* and *Forks* can be interpreted as the lowest participation level because they do not influence projects' progress directly (everyone can *fork* and *star* a project without any direct "consequences"). Whereas *subscribing* is more active (but still indirect), because it provides constant updates of the current project activities to the subscribing user.

The impact of internal (and external) contribution on these levels of participation has to be considered specifically. In general, higher-active responses are more important than lower-passive responses.

[36] *LibreOffice* is a fork of *OpenOffice* which is continuing as independent project for instance

[37] `https://help.github.com/articles/using-pull-requests/`

[38] `https://guides.github.com/features/issues/`

3.5 Projects' Data Enquiry

Due to GitHub API limitations (GitHub (2015a)) we are only able to query 1000 results for each language. As mentioned before, only firm's initiated projects for the 10 most famous programming languages are interesting for the observation.

By querying the most popular 1000 repositories for each of the programming languages [39] we finally receive 10,000 repositories [40] from 2858 different organizations [41], sorted descending by listed ”Top Repositories” for each organization [42] (see table 20 on page 72).

This results in 2,858 organizations having 78,055 public repositories (see table 4 for details [43]). By selecting the most relevant organizations [44] we get 113 (share of 3.95 %) relevant organizations [45]. We can classify finally 58 relevant commercial firms (see table 22 and chapter 3.6).

Evidently, closer examination of the data shows that nearly every commercial successful firm in the technology area has a public GitHub profile (see table 21 on page 73).

We classify **Repositories** as:

Top Repositories (sometimes also called *Top Projects*), which were found the first 1,000 rank search result

Residual Repositories are the remaining amount of repositories by each firm

All firms' projects that are classified in one of the 10 chosen programming languages and are not tagged as ”fork” are relevant [46].

[39] for each language: `https://api.github.com/search/repositories?q=language:$ProgrammingLanguage$&sort=stars&order=desc&per_page=100&page=1...10`, queried on 19.1.2016

[40] `https://git.zeitpulse.com/philipp/masterthesis-data/tree/master/apidata/top_repos_by_language`

[41] `https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/csv/organizations.csv`

[42] in the following top repositories or top projects are all projects listed in the top 1,000 search matches

[43] `https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/apidata/organizations_top_repos_restructured.json`

[44] i.e. having at least 4 different projects in the search results of most popular projects on all 10 programming languages

[45] data source: `https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/csv/repositories_details.csv`

[46] Remark: Unfortunately, not all projects can be classified absolutely as *fork* and *not fork* since not all projects are tagged as such on GitHub itself

Popular / Top Repositories:	10,000
Repositories by Organizations:	78,055
Organizations:	2,858
Organizations with relevant Number of Top Repositories:	113
Commercial Organizations:	58
Share of relevant Organizations:	**3.954 %**
Share of observed Organizations:	**2.03 %** (51.33 %)

Table 4: Top Repositories for selected Programming Languages on *GitHub*. ”Share of observed Organizations” are finally classified commercial firms (selection criterions will be explained in chapter 3.6)

3.6 Firms are relevant commercial Organizations

A firm is relevant for the study, if

- it is a commercial organization
- uses at least one of its projects actively in their business
- it contributes source code to their own projects and to other OS projects
- it holds and uses a global web domain [47]

The selection was done manually by evaluating the form of company, activity with employed developers and firms' business activity. Since the classification of commercial activity is qualitative, it is also based on ratings by employees on Glassdoor [48] [49], which will be introduced in chapter 3.15.

We will evaluate the repositories of these firms [50] and compare participation of firm employed and external developers.

The assessment of which organization is commercial and builds a business on its OS projects follows the assumption that:

- most projects are used by the firm in commercial context (like *docker, aws* ...)
- the firm offers paid services for some of its project (like *chef, Microsoft,* ...)
- the firm acts commercial - even the OS projects and their service is available for zero costs (like *GitHub, facebook* ...)

3.7 Structure of a git Project

A git repository can roughly be described with the following attributes (Loeliger and McCullough, 2012, p. 32):

- commits, tags and merges
- branches
- trees *(not relevant here)*
- files / blobs *(not relevant here)*

A detailed explanation of those attributes is not necessary because only *commits* and *branches* are relevant for the study and will be explained in the following chapters.

[47] an actively used web domain is important to classify firm employed and external developers

[48] `https://www.glassdoor.com/`

[49] `https://git.zeitpulse.com/philipp/masterthesis-data/tree/master/apidata/glassdoor/employers`

[50] `https://git.zeitpulse.com/philipp/masterthesis-data/blob/master/csv/repositories.csv`

3.8 Branches of Interest

Each git repository contains different branches. Every branch can be a different working subtree (i.e. source code environment) of a project. We will only inspect the default branch of each project which is in most cases the **master branch**. The master branch is by convention the official branch containing all finally submitted and accepted code of a project (Loeliger and McCullough, 2012, p. 89-90). Some projects have different default branches (*production* branch for instance) and will be scope of the observation if it is tagged as such on GitHub.

3.9 Firms' License Usage

Which kind of licenses firms are using might reflect their strategic and projects' commercial long-term intention respectively.

With the following segmentation (based on Bonaccorsi and Rossi Lamastra (2003)) for the most used licenses of the observed repositories / firms, we can describe a spectrum from restrictive to permissive licenses. As figure 5 illustrates: most firms use (very) permissive licenses (from green-blue to magenta) instead of restrictive licenses (red to orange) [51]. The reason might be simple: Permissive license avoid problems of the copy-left idea and enables future commercial (closed source) development.

[51] which is in line with Bonaccorsi and Rossi Lamastra (2003) and Black Duck (2015)

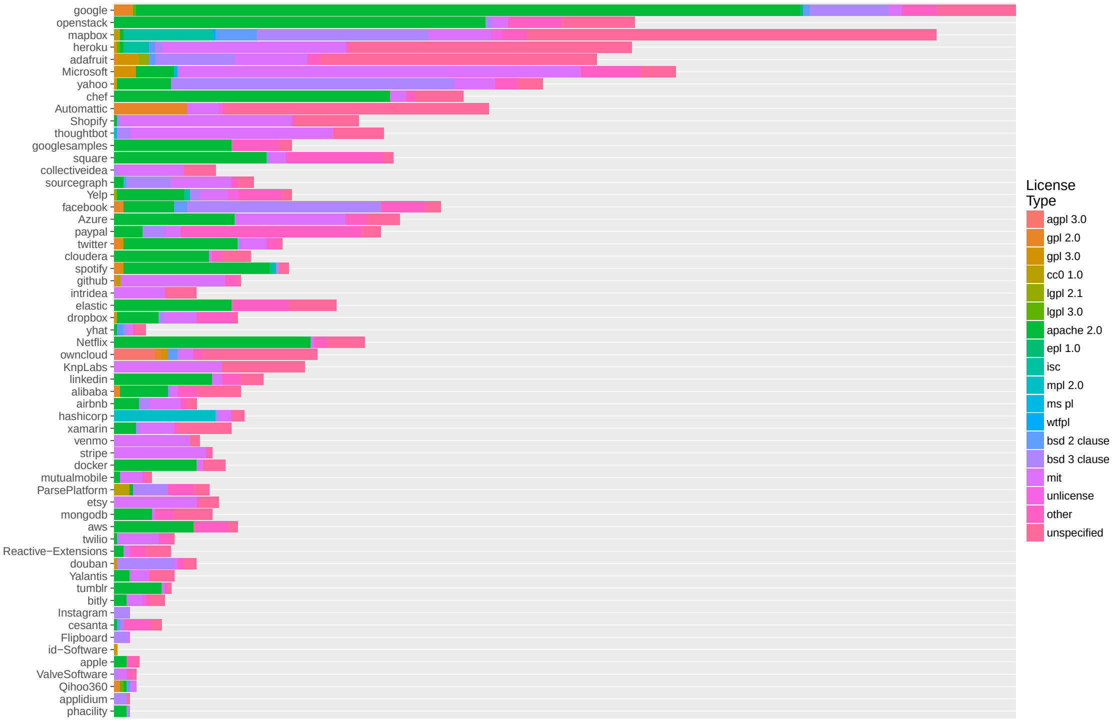

Figure 5: Most OS projects by firms do use (very) permissive licenses (from green over blue to magenta). The x-axis (caption not in the plot) represents the absolute number of repositories (from 15 - 284 repositories).

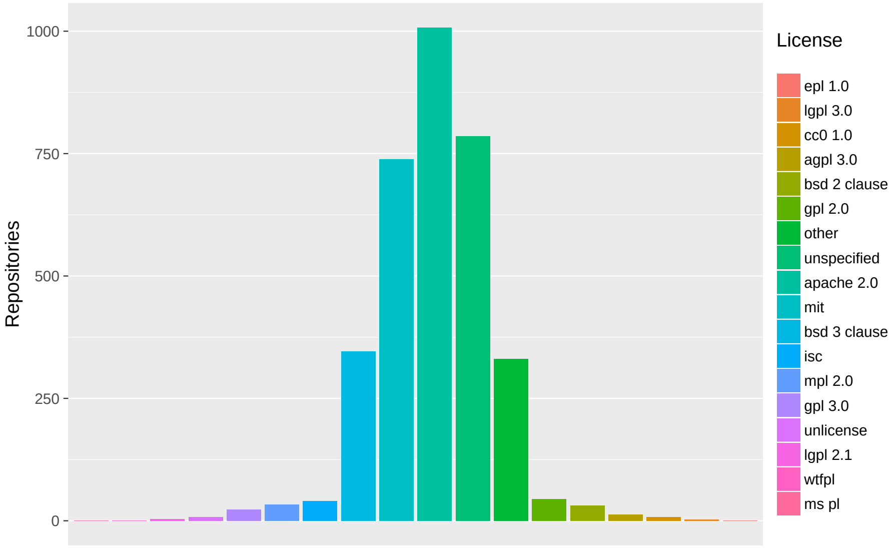

Figure 6: Frequency of licenses in GitHub repositories of firms. Permissive licenses (like Apache, MIT and BSD) are favored. *Unspecified* means not categorized (i.e. in most cases proprietary OS license by firm). Graphs for Top and Residual Projects in figure 14 and 15.

Strong Copyleft (restrictive)

- GNU Affero General Public License (agpl 3.0)
- GNU General Public License (gpl 2.0, gpl 3.0)
- Creative Commons CC0 1.0 Universal (cc0 1.0)
- SIL Open Font License (ofl 1.1)

Mixed (less restrictive, more permissive)

- GNU Lesser General Public License, Version 2.1 (lgpl 2.1, lgpl 3.0)
- Apache License, Version 2.0 (apache 2.0)
- Eclipse Public License (epl 1.0)
- ISC-Licence (isc)
- Mozilla Public License, version 2.0 (mpl 2.0)
- Microsoft Public License (ms pl)

Non-Copyleft (very permissive)

- Do What The Fuck You Want To Public License (wtfpl)
- BSD licenses (bsd 2 clause, bsd 3 clause)
- The MIT License (mit)

Uncategorized

Beside that there are also unlicensed projects (i.e. no license on ”purpose”) and license unspecified projects (i.e. no license detected by GitHub).

3.10 Classification of Developers

We distinguish two groups of participants:

1) Internal Developers (also called *firm employed developers*) are developers / users who

- are currently employed at the company of the observed project, or
- were employed at the company of the observed project in the past

2) External Developers are developers / users who

- never worked at the company of the observed project, and / or
- are independent developers / hobbyist programmer

3.11 Evaluation and Classification of Commits

A *commit* represents the participation of a developer. It includes code changes or new code [52]. In the following, we will regard a commit as a participation activity. We ignore the size of the commit (how many lines and files have changed) because measuring code quality by these simple metrics would be difficult. According to Stamelos et al. (2002) further specialized software is needed to measure code quality reliable.

The number of projects' valid commits of the selected 58 firms is 2,951,188. These commits are used in the final version of the projects (i.e. they are productive code commits). Commits without a valid email address [53] are sorted out. As mentioned before, a commit can also be a **merge** and is taken into account as well [54].

There are 1,111,384 of firm employed developers and 1,839,804 of external developers [55]. Thus, the share of firm employed developers is at least 37.66 % on all observed projects.

To compare the participation of firm employed and not firm employed developers, each commit must be identified by the author's email address and be assigned to internal (i.e. employed by firm) or external (i.e. employed not by firm of the project). The selection of firm employed developers is quite conservative, because we only regard developers with a firm e-mail address or developers which are known to work for the company. Many more developers may work or have been working for the firm in the past.

To maximize the classification of firm employed developers with acceptable effort, a whitelist was generated and checked manually via LinkedIn for the developers with the highest share of participation. The whitelist [56] contains over 550 manually checked developers, everyone with a share of code contribution from 25 % up to 100 % for a specific project.

3.12 Time and Location of Code Contribution

Through almost 3 Mio. observed commits we can measure a representative frequency of code contribution (see figures 7, 8, 16 and 17). The histograms are stacked and not overlaid [57].

[52] detailed code changes can be inspected by *git diff $commithash*

[53] here a valid email address following ABNF with the regular expression
``/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$/``
Source: `http://www.w3.org/TR/html5/forms.html#valid-e-mail-address`

[54] *Merge* is an activity of implementing proposed code changes into the (final) project

[55] for classification see next section

[56] can be received as csv file: `https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/csv/classification/int_ext_developer_classification.csv`

[57] the total number of commits by not firm employees is actually higher (see previous section for numbers)

Interpretation: Most of the code contribution takes place between 9:00 and 19:00 with its peak at 15:00 (see table 8). An interesing observation is that firm employed developers' and external developers' code contributions frequency is very similar. The reasons could be:

a) external developers are employed as well (but at another firm)
b) externals / freelance workers write and commit code in the same time frame because they are most efficient in that time frame
c) externals / freelance workers write and commit code in the same time frame because code contribution is a form of collaboration and "forces" same time frames

According to the manual classification of developers (most of them are or were employed at the particular firm) and the fact that code contribution time is similar regarding to the local time, reason a) seems to be the prime cause: **external developers are employed developers of other / rival firms**.

Beside that, the core hours are in line with independent studies about working hours (see figure 9). Thus, the most effective time period for developers seems to be between midday and late afternoon. The distribution of worldwide code contribution confirms the leading role of Silicon Valley / California / Oregon and USA in the firm managed open source segment (see figure 7 and 16)

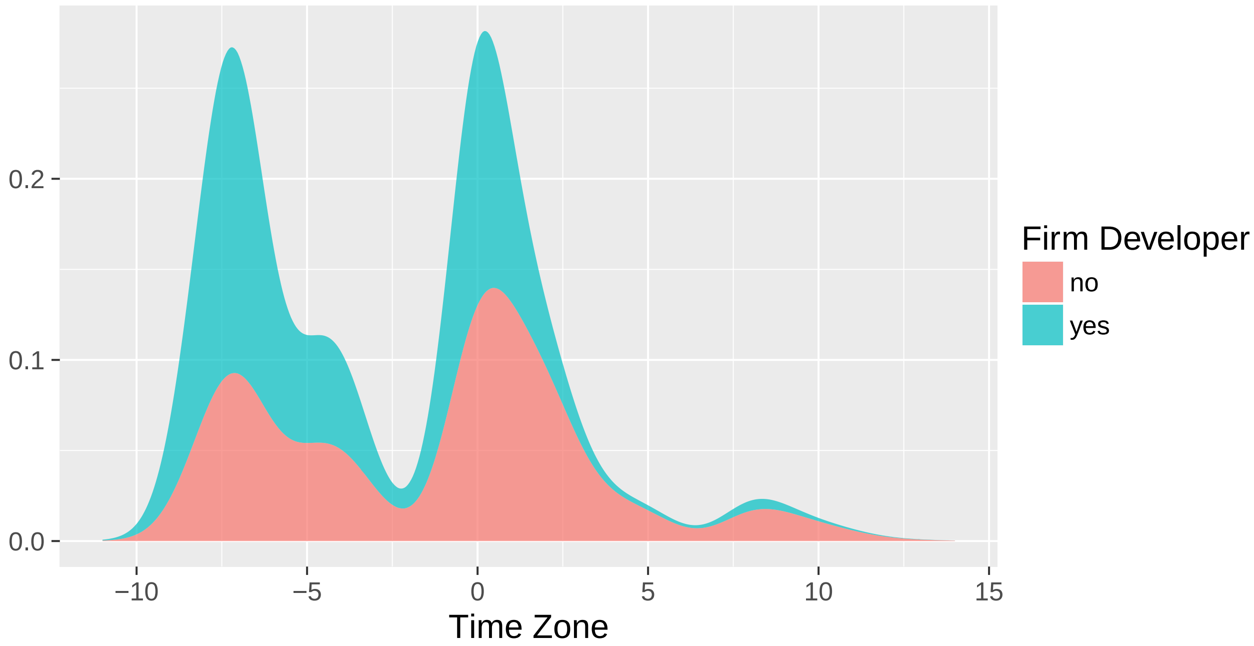

Figure 7: Code contributions by firm employed and external developers over time zones. Assumption: Most code contributions are from Europe (UTC between -1 and +2) and USA / Canada (UTC between -4 and -8) by considering economicaly most successful countries by longitude (and ignoring latitude influences).

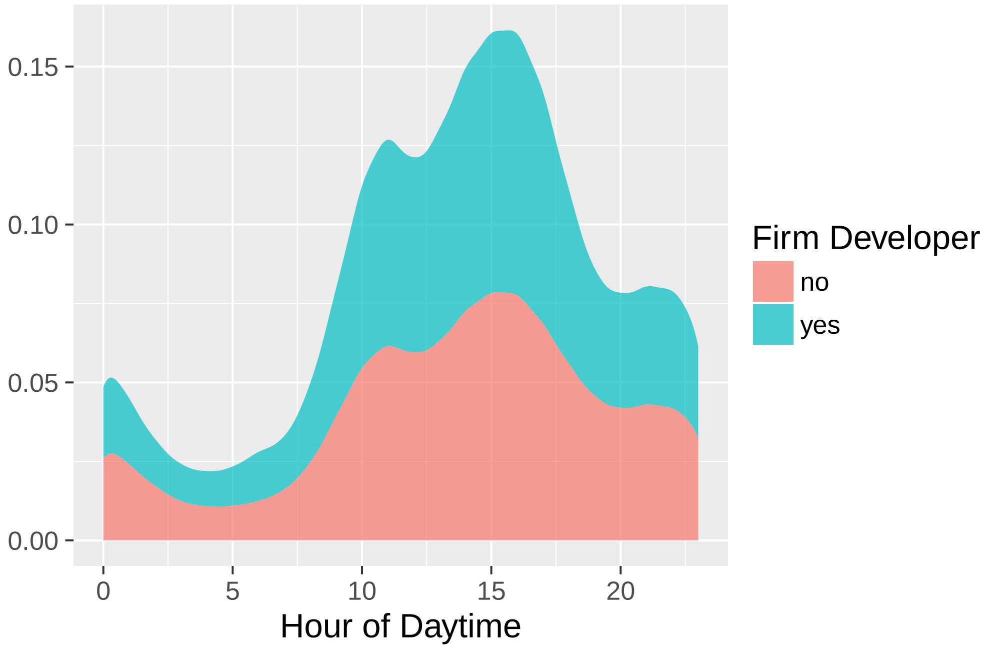

Figure 8: Code contributions of firm employed and external developers by local daytime (workdays, weekend and holidays are included). Most of the code commits are placed between 09:00 - 19:00, with its peak between 15:00 - 16:00

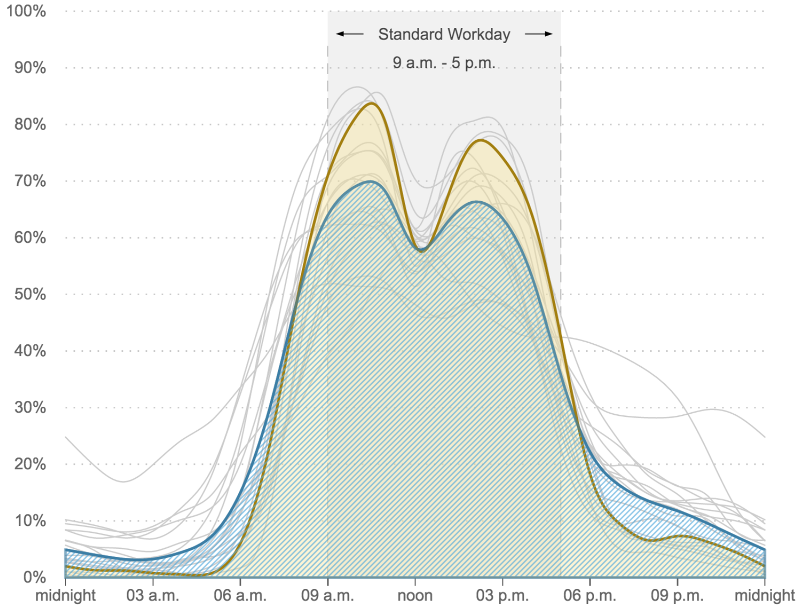

Figure 9: 2011-2012 Annual Averages ("Computer and Mathematical" (yellow) compared to "All Jobs" (green)): *The majority of people are at work from 9:00 to 17:00, with a small break in the middle of the day for lunch* (according to Bui (2014)). Source: BLS, American Time Use Survey; Credit: Quoctrung Bui/NPR

3.13 Observed Repositories

The final number of observed repositories is 3,419, including 610 top projects and 2,809 residual projects (see table 5 for further details).

Statistic	N	Mean	St. Dev.	Min	Max
Age (in days)	3,419	743.790	560.458	0	2,904
Number of Contributors	3,419	17.181	40.053	2	468
Number of Commits	3,419	748.581	8,510.597	2	428,840
Number of Commits by firm employed developers	3,419	250.875	1,501.224	0	56,624
All Issues count	3,419	20.836	75.842	1	1,884
Closed Issues count	3,419	2.817	18.725	0	706
Open Issues count	3,419	18.019	74.182	0	1,884
Stars	3,419	426.290	1,538.234	0	35,214
Subscribers	3,419	75.221	123.100	1	2,617
Forks	3,419	99.204	378.507	0	10,772
Ratio (share of firm employed developers)	3,419	0.521	0.348	0.000	1.000
Mean share of Top Repositories	3,419	0.178	0.383	0	1

Table 5: Statistic of observed GitHub Projects

The number of repositories (i.e. observations) will vary between statistical models [58].

3.14 The Role of Developers on Issues

Issues are communication channels for GitHub repositories. GitHub provides data of users who opened issues and users who participate on issues by commenting / discussing. All comments will be counted and assigned to firm employed and external developers [59].

As mentioned before, issues can:

- be technical (bug reports / source code related)
- include general ideas / feature requests
- be organizational related

Finally, we have 405,163 issues with 1,718,363 comments (6,190 issues and 32,599 issues' comments are by firm employed developers).

[58] some are using ”Top Repositories”, ”Residual Repositories”, ”All Repositories” and ”Repositories older than X days” for instance

[59] some models also include a measurement of weighted comments by its content size

3.15 Firms' OS Commitment as Proxy for Quality of Work Environment

To which extend does contribution of firm employees on firms' OS projects reflect the employer quality? If employed developers get encouraged by their employer to initiated and contribute on OS project, do they stay more likely at their current employer or rate their working climate more positive? If an firm employed developer is allowed to work on OS projects as part of his work, does it increase their incentive to be more creative and try out new ideas by themselves? Or do developers initiate projects to demonstrate and promote a specific technology innovation inside a firm?

This research question itself would probably fill another study (see 5.2). But we will observe briefly a few data collections regarding to work climate and GitHub activities of firms. For that, ratings from employees of their workplace are evaluated. The data is received from Glassdoor. Ratings of 50 firms were found [60] but only 14 firms (see table 23) had enough ratings and enough Top Repositories (i.e. at least 38 ratings and 10 Top Repositories) to be "representative" with partly significant regression results. However, there is a (weak) positive significant relation between the number of popular OS projects and the rating by the employees (and vice versa) which could be measured by *employees rating their firm* and *firms' number of popular Open Source projects on GitHub* through the OLS method (see table 6 for details). Interpretation: (a) if a firm initiates more OS projects, the firm is more valuable as employer for developers (b) if a firm is a good place to work, employees will more likely initiate OS projects through the firm.

Thus, the numbers of observations / ratings are not representative and might be a promising approach for further research.

[60] https://git.zeitpulse.com/philipp/masterthesis-data/tree/master/apidata/glassdoor/employers

	Dependent variable:			
	Rating	OS Projects	Ratings count	Work-Life-Balance
	(1)	(2)	(3)	(4)
Average Ratio Top Projects	−0.115	12.263	4,441.273	0.222
	(0.226)	(21.167)	(5,659.478)	(0.446)
Average Ratio Residual Projects	0.108	−21.155	−386.364	−0.342
	(0.217)	(18.723)	(5,751.773)	(0.410)
Ratings Count	0.00003*	−0.002		−0.00004
	(0.00001)	(0.001)		(0.00003)
Rating		74.312**	17,654.490*	1.638**
		(25.873)	(8,434.416)	(0.486)
Culture and Values	0.634***	−37.375	−13,061.970*	−0.769
	(0.132)	(24.268)	(5,587.065)	(0.510)
Work-Life-Balance	0.424**	−42.472***	−7,375.867	
	(0.126)	(9.910)	(4,863.693)	
Age of Firm on GitHub	−0.0001	0.014**	0.728	0.0003**
	(0.0001)	(0.004)	(2.016)	(0.0001)
OS Projects count	0.008**		−142.543	−0.019***
	(0.003)		(104.811)	(0.004)
Residual Repos. count	−0.001**	0.152***	26.620	0.003***
	(0.0005)	(0.018)	(15.218)	(0.001)
Constant	0.024	−4.067	7,961.749	−0.065
	(0.338)	(31.831)	(7,988.258)	(0.665)
Observations	14	14	14	14
R^2	0.985	0.957	0.891	0.955
Adjusted R^2	0.962	0.889	0.718	0.884
Residual Std. Error (df = 5)	0.076	7.202	1,975.574	0.150
F Statistic (df = 8; 5)	41.895***	14.067***	5.135**	13.407***
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 6: Rating of Work Environment by employees and Firms' activity on Open Source projects

3.16 Microsoft and GitHub Inc.: Contribution and Social Success Metrics of Atom and VSC

	Init. by Firm	Editor	License	Stars	Subscr.	Forks	Contrib.	Published	Op.Issues	Commits	Ratio
1	GitHub	Atom	MIT	23,998	1,524	4,094	260	2014 Q2	1,644	27,373	73.63%
2	Microsoft	VSC	MIT	9,884	679	1,205	56	2015 Q2	774	1,909	58.87%

Table 7: Repository Data of Atom and VSC [61]

The time series 10 and 11 [62] show the code contribution of *GitHub* / *Microsoft* employed developers against *external developers*. **Gold** represents code contribution by external developers, **Black** represents code contribution by firm employed developers. *Atom* was published as open source in May 2015, *VSC* a half year later in November 2015.

If you compare the basic statistics of the competitors *Microsoft* and *GitHub Inc.* (see table 17 and 18 on page 68) you can see that *Microsoft* is a ”youngster” on *GitHub* [63]: Microsoft projects have a mean age of less than 1 year, high share of firm employed developers (over 68 %) and fewer ”Top Projects” (18 % of Microsofts' projects are ”Top Projects”). Whereas *GitHub Inc.'s* ”Top Project” share is 40 % with a lower share of firm employed developers (53 %) and older projects (mean age is almost 2 years).

Interpretation: This basic statistic provides indications of the differences between the two companies *GitHub Inc.* and *Microsoft*: Microsoft is a ”newer” member of the OS community (according to the age of projects) and tries to gain its reputation (by initial high share of firm employed developers) and tries to establish popular projects (share of ”Top Projects”) with the long-range objective to motivate external developers (i.e. the OS community) to participate on their projects.

[61] Retrieved on: 21st January 2016 via `https://github.com/`

[62] Retrieved on 18th January 2016 via 'git log'

[63] the following values are mean values of each firms' projects

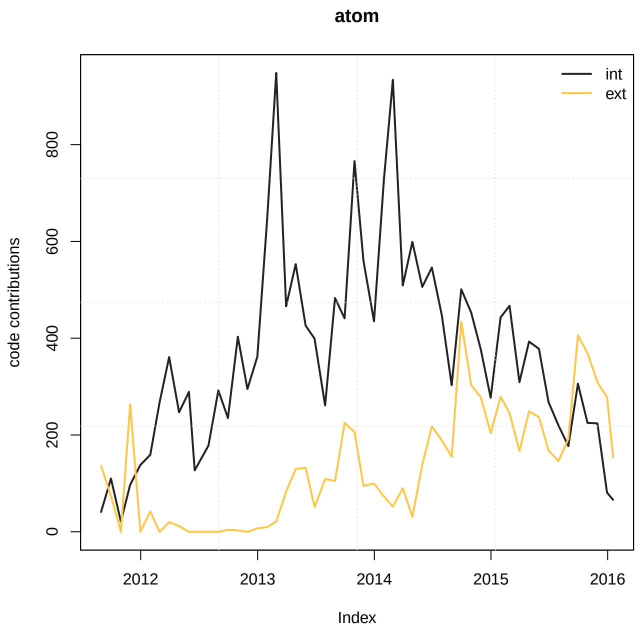

Figure 10: Code Contribution of ”Firm employed Developers” (int) and ”External Developers” (ext) to Atom (GitHub Inc.). Published as open source in May 2015

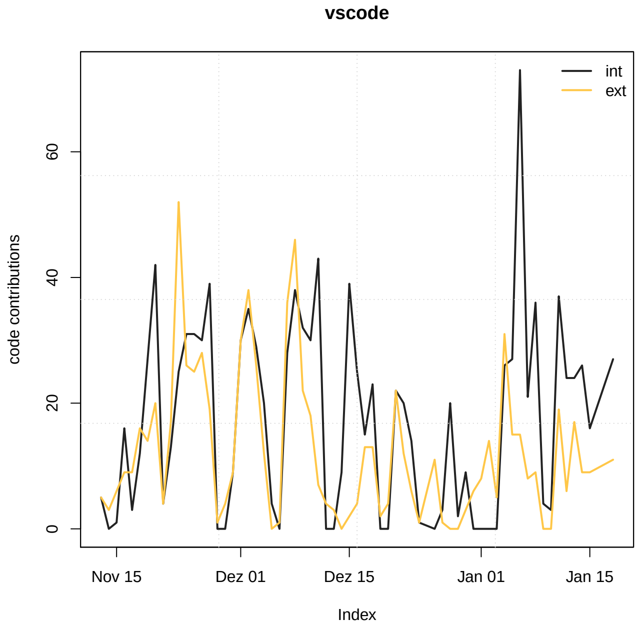

Figure 11: Code Contribution of ”Firm employed Developers” (int) and ”External Developers” to Visual Studio Code (Microsoft). Published as open source in November 2015. (time range of plot: Nov. 2015 - Jan. 2016)

4 Analysis

With the information of code commits, *GitHub* projects' metrics, issue communication and user events (over time) we can verify the following hypotheses:

H 1 Firm employees' participation affects the participation of external developers

  H 1.1 If firm employees contribute source code more often, external developers do as well

  H 1.2 If firm employees participate more often on issue threads, external developers do as well

H 2 Firm employees' participation (in the beginning) affects the (later) success of firm-initiated open source projects

  H 2.1 If firm employees contribute source code more often in the beginning, the project gets more successful

H 3 If firm employees' contribution share is higher overall the more likely the project is popular

4.1 Terms and Definitions

Younger Projects: Projects that are younger than 1 year (<365 days)

Older Projects: Projects that are older than 1 year (>= 365 days)

Top Projects: Projects that were found in the first 1000 most popular projects of a programming language (can be 1 or 0, see below for mathematical definition)

Residual Projects Projects that were *not* found in the 1000 most popular projects for a programming language [64]

Ratio Code commit share of firm employed developers to external developers (normalized value between 0 and 1)

$$Ratio = \begin{cases} 1 & \text{if repository is maintained to 100\% by firm employed developers only} \\ 0 & \text{if repository is maintained to 0\% by firm developers (i.e. 100\% external developers)} \\ \in (0,1) & \text{else} \end{cases} \tag{4}$$

$$Top\ Project = \begin{cases} 1 & \text{if is found in search of 1000 most starred projects of the 10 selected Languages} \\ 0 & \text{else} \end{cases} \tag{5}$$

[64] ”Top Projects” and ”Residual Projects” together build the set of relevant projects

4.2 Slopes and Dots (on plot figures)

Yellow-Solid ／ Correlation coefficient for the influence of code commits from firm employed developers on external developers (via OLS)

Blue-Dashed ⋰ Correlation coefficient for the influence of code commits from external developers on firm employed developers (via OLS) [65]

Grey Dots ⁘ Number of code contribution by firm employed developers (x-axis) and external developers (y-axis)

4.3 H 1: Firm employees' participation affects the participation of external developers

4.4 H 1.1: If firm employees contribute source code more often, external developers do as well

4.4.1 H 1.1: Models and Terminology

The following linear models (using OLS) examine the influence of code contribution by firm employed developers on external developers and vice versa.

Every project is represented by a dot and plotted through number of code contributions by firm employed developers (x-axis) against number of code contributions by external developers (y-axis). The influence of firm employed developers' participation (independent) on external developers' participation (dependent) is represented by the solid yellow slope and defined in the models 1.1, 1.3, 1.5, 1.7, 1.9, 1.11 and 1.13. Conversely, the influence of external developers' participation (independent) on firm employed developers' participation (dependent) is represented by the dashed blue slope and defined in the models 1.2, 1.4, 1.6, 1.8, 1.10, 1.12 and 1.14. For better understanding see plots 27 - 32 [66] and regression results i.a. in table 8 (page 41 and page 77ff).

int. commits / internal commits: Number of code contributions by firm employed developers
ext. commits / external commits: Number of code contributions by external developers

See section 4.1 and 4.2 for further explanation of terms, definitions and plots.

R code will be used to describe the models [67] since it's a bit handier to describe than in mathematical

[65] actually x- and y-axis must be reverted to illustrate external impact on firm employed developers, but for the sake of simplicity blue and yellow slopes share one plot here

[66] plot for model 1.12 and 1.13 (internal and external commits on younger Residual Projects) was omitted since there is no significant correlation, as noted in table 26

[67] all analyses are made in *R*; so the models are taken from the source code

notation. *See listing 3 on page 59 for details.*

4.4.2 H 1.1: Plots

See page 125 for plots.

4.4.3 H 1.1: Regression Tables

Additional regression tables can be found in chapter B.3, page 77 - 79 (table 24, 25 and 26).

	Dependent variable:			
	ext. commits	int. commits	ext. commits	int. commits
	(1.1.1)	(1.1.2)	(1.1.3)	(1.1.4)
int. commits	0.999^{***}		0.794^{***}	
	(0.091)		(0.040)	
ext. commits		0.034^{***}		0.492^{***}
		(0.003)		(0.025)
Constant	247.010^{*}	233.808^{***}	48.746	540.463^{***}
	(138.107)	(25.282)	(141.371)	(109.104)
Observations	3,419	3,419	610	610
R^2	0.034	0.034	0.390	0.390
Adjusted R^2	0.034	0.034	0.389	0.389
Residual Std. Error	7,964.959 (df = 3417)	1,475.494 (df = 3417)	3,368.299 (df = 608)	2,651.187 (df = 608)
F Statistic	121.248^{***} (df = 1; 3417)	121.248^{***} (df = 1; 3417)	388.882^{***} (df = 1; 608)	388.882^{***} (df = 1; 608)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 8: Influence of internal commits on external commits (and v.v.) on ”All Projects” (Model 1.1.1 - 1.1.2) and ”Top Projects” (Model 1.1.3 - 1.1.4)

4.4.4 H 1.1: Results and Interpretation

The regression results confirm the hypothesis: [68] If firm employed developers contribute more source code, external developers contribute more source code as well.

The reversed impact (i.e. influence of code contributions from external developers on contributions of firm employed developers) is also existing - except for (older) top projects - but essentially weaker in all cases. The distance between the impact of internal developers on external developers and the other way round decreases if projects are more popular (i.e. "Top Projects") [69]. This indicates, that the biggest positive impact of internal developers' participation on external developers' participation is reached in not well-known (and younger) projects.

The number of code commits by firm employed developers have a high positive significant influence on the number of code commits by external developers. The impact of code commits by external developers on firm employed developers is also positive, but significant lower - especially in not well-known projects.

[68] Except model 1.1.13 and 1.1.14 (influence of code contribution on young residual projects, i.e. new and unknown projects of firms). But we have to be consider, that model 1.1.13 and 1.1.14 are using the smallest subset of 909 observations, which may cause the poor fitting among other things.

[69] on "Top Projects" externals' participation has a high positive impact on internal developers as well

4.5 H 1.2: If firm employees participate more often on issue threads, external developers do as well

4.5.1 H 1.2.1: Models and Terminology for Issues

We observe the number of created (i.e. opened) issues by *GitHub* users. With linear regression we want to determine the influence of the participation of firm employed developers on the participation of external *GitHub* users and vice versa. Neither we consider the content size (i.e. how much the user actually wrote) nor do we rate the quality of an issue.

issues by int. users: Number of issues created by firm employed developers / *GitHub* users
issues by ext. users: Number of issues created by external developers / *GitHub* users

See section 4.1 and 4.2 (page 39 and 40) for further explanation of terms, definitions and plots.

See listing 4 on page 60 for defined models in R code.

4.5.2 H 1.2.1: Plots for Issues

See page 125 for plots.

4.5.3 H 1.2.1: Regression Tables for Issues

Additional regression tables can be found in 27, page 80 - 82 (table 27, 28 and 29).

	Dependent variable:			
	issues by ext. users	issues by firm empl. users	issues by ext. users	issues by firm empl. users
	(1.2.1)	(1.2.2)	(1.2.3)	(1.2.4)
issues by firm empl. users	23.800^{***}		33.598^{***}	
	(0.867)		(2.443)	
issues by ext. users		0.009^{***}		0.008^{***}
		(0.0003)		(0.001)
Constant	70.842^{***}	0.704^{***}	228.113^{***}	2.252^{***}
	(7.446)	(0.143)	(38.219)	(0.599)
Observations	2,935	2,935	523	523
R^2	0.205	0.205	0.266	0.266
Adjusted R^2	0.204	0.204	0.265	0.265
Residual Std. Error	395.887 (df = 2933)	7.523 (df = 2933)	817.527 (df = 521)	12.557 (df = 521)
F Statistic	754.343^{***} (df = 1; 2933)	754.343^{***} (df = 1; 2933)	189.129^{***} (df = 1; 521)	189.129^{***} (df = 1; 521)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 9: Impact of issue participation by firm employed developers on external users (and v.v.) in ”All Projects” (Model 1.2.1 - 1.2.2) and ”Top Projects” (Model 1.2.3 - 1.2.4)

4.5.4 H 1.2.2: Models for Issues' Comments

Every issue contains comments which enables discussion between users. Beside the number of comments per issue we consider the size of involvement for every user by measuring the size of written text. Every character of a comment is taken into account and finally add up to obtain a share of content contribution. This implies that if (for example) a firm employed developer writes 400 characters long comments in total inside an issue thread with all comments together having a size of 1,200 characters, the content share of the firm employed developer would be $\frac{1}{3}$. We also use the number of issue comments as unit of measurement.

Every dot represents an issue with it's comments and is plotted through *number of issue comments by firm employee / developer* (x-axis) against *number of issue comments by external users* (y-axis). See section 4.1 and 4.2 (on page 39 and 40) for further explanation of terms, definitions and plots.

issues' comments by firm employed developers: Number of issue comments by firm employed developers / GitHub users

issues' comments by external developers / users: Number of issue comments by external developers / GitHub users

content share by firm employed developers: Share (between 0 and 1) of written content on comments by firm employed developers / GitHub users with respect to all written content of the issue thread

content share by external developers / users: Share (between 0 and 1) of written content on comments by external developers / GitHub users with respect to all written content of the issue thread

Note: The actual numbers of observations are much higher: The number of preprocessed issue comments is 1,034,702 (each of them belonging to one of the 2,609 issues which are finally counted as observation).

See listing 5 on page 61 for models defined in R code.

4.5.5 H 1.2.2: Plots for Issues' Comments

See page 127 for plots.

4.5.6 H 1.2.2: Regression Tables for Issues' Comments

Additional regression tables can be found in chapter B.5, page 84 - 85 (table 30 and 31).

	Dependent variable:			
	Number of comments	Comments by ext. developers		Comments by firm employed developers
	(1.3.1)	(1.3.2)	(1.3.3)	(1.3.4)
Content share by firm employed developers	17,616.700	17,210.760		
	(108,998.200)	(105,034.700)		
Comments by firm employed developers			14.362***	
			(0.263)	
Comments by ext. developers				0.037***
				(0.001)
Constant	492.454***	481.881***	330.379***	$-$7.338**
	(92.407)	(89.046)	(60.837)	(3.108)
Observations	2,609	2,609	2,609	2,609
R^2	0.00001	0.00001	0.534	0.534
Adjusted R^2	$-$0.0004	$-$0.0004	0.533	0.533
Residual Std. Error (df = 2607)	4,716.907	4,545.387	3,104.210	157.884
F Statistic (df = 1; 2607)	0.026	0.027	2,982.656***	2,982.656***
Note:				*p$<$0.1; **p$<$0.05; ***p$<$0.01

Table 10: Impact of participation by content share in issues' comments (Model 1.3.1 - 1.3.2) and number of comments by firm employed users on external users (and v.v.) in all Projects (Model 1.3.3 - 1.3.4)

4.5.7 H 1.2.1 and H 1.2.2: Results and Interpretation

If firm employed users [70] participate more actively by writing comments on issues, they have a positive significant impact on the participation number of external users (by writing comments as well).

However, there is no evidence that the share of content has an effect as well. The reason might be that the size of written text is not as important as the action of participating with communication itself. Maybe measurement of the content quality in a more sophisticated way could find a significant correlation.

4.6 H 1: Results and Interpretation

In all participation areas (code commitment, issue opening and issue commenting) the commitment of firm employees has a positive significant impact on the commitment of external developers. The impact is essentially higher than the influence from external developers on internal developers [71].

[70] *GitHub* users can be moderators, representatives or developers of a firm

[71] as stated before, the impact of external developers on internal developers is also significant and positive, but essentially weaker and tending towards zero in most cases

4.7 H 2: Firm employees' participation (in the beginning) affects the (later) success of firm-initiated open source projects

4.7.1 H 2.1: If firm employees contribute source code more often in the beginning, the project gets more successful

All observed repositories (1,409 repositories) are older than 4 years to make them comparable over time [72]. *Today* is the point in time of receiving data [73]

$$\forall \; ratio_t, \; forks_t, \; subscribers_t \tag{6}$$

$$t \in \{1, 2, 3, 4, \{1-2\}, \{2-3\}, \{3-4\}, \{5-6\}, \textit{today}\} \tag{7}$$

$$\tag{8}$$

$t_1 :=$ first 3 months (91 days)

$t_2 :=$ first half year (182 days)

$t_3 :=$ first year (364 days)

$t_4 :=$ first two years (728 days)

$t_{1-2} :=$ first 182 days

$t_{2-3} :=$ between day 183 - 364

$t_{3-4} :=$ between day 365 - 728

$t_{5-6} :=$ after 728 days until ”today”

See listing 6 on page 62 for models defined in *R* code.

[72] All finally observed repositories can be found at: `https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/csv/calculated/time_obervations_in_section_code_and_events.csv`

[73] period of receiving data is from 16th - 20th January 2016

4.8 H 2.1: Regression Tables

Additional regression tables can be found in chapter B.6, page 86 - 93 (see table 32, 33, 34, 35, 36, 37, 38 and 39)

	Dependent variable:			
	Top Project			
	(2.1.1)	(2.1.2)	(2.1.3)	(2.1.4)
Age	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}
	(0.0001)	(0.0001)	(0.0001)	(0.0001)
Ratio_1	1.774^{***}			
	(0.184)			
Ratio_2		1.760^{***}		
		(0.174)		
Ratio_3			1.680^{***}	
			(0.173)	
Ratio_4				1.758^{***}
				(0.178)
Constant	-2.722^{***}	-2.906^{***}	-2.965^{***}	-3.160^{***}
	(0.209)	(0.216)	(0.218)	(0.227)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	−704.245	−699.204	−702.446	−699.158
Akaike Inf. Crit.	1,414.490	1,404.408	1,410.893	1,404.317
Note:		*p<0.1; **p<0.05; ***p<0.01		

Table 11: The higher the ”Ratio” (code contribution share) by firm developers in the beginning the more likely it is a ”Top Project” in the long-run (Model 2.1.1 - Model 2.1.4)

	Dependent variable:					
	Top Project					
	(2.1.5)	(2.1.6)	(2.1.7)	(2.1.8)	(2.1.9)	(2.1.10)
Age	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}
	(0.0001)	(0.0001)	(0.0001)	(0.0001)	(0.0002)	(0.0002)
Subscribers_1	24.942^{***}					
	(5.700)					
Subscribers_2		6.733^{***}				
		(1.453)				
Subscribers_3			5.789^{***}			
			(0.836)			
Subscribers_4				1.231^{***}		
				(0.433)		
Subscribers_5					-1.673^{***}	
					(0.310)	
subscribers.today						0.005^{***}
						(0.0003)
Constant	-2.481^{***}	-2.798^{***}	-3.637^{***}	-2.972^{***}	-2.095^{***}	-4.514^{***}
	(0.203)	(0.231)	(0.290)	(0.321)	(0.193)	(0.366)
Observations	1,409	1,409	1,409	1,409	1,409	1,409
Log Likelihood	−740.587	−740.066	−725.080	−746.645	−734.832	−302.736
Akaike Inf. Crit.	1,487.175	1,486.132	1,456.161	1,499.291	1,475.665	611.471
Note:						*p<0.1; **p<0.05; ***p<0.01

Table 12: The more "Subscribers" a project gains in the beginning (i.e. the first 3 months) the more likely it is a "Top Project" in the long-run (Model 2.1.5 - 2.1.10)

4.9 H 2.1: Interpretation and Conclusion

All models show that the attention a projects gains in the beginning has the biggest impact on later success and popularity. Moreover, commitment of firm employed developers in the beginning has a stronger impact of (later) social success metrics. Thus, early commitments of employees have a stronger positive impact on long-term project's success and popularity than later ones.

The most important time period seems to be the first 12 months (see table 36, 37, 38 and 39). This time period has the biggest impact on early social success ("Stars", "Forks" and "Subscribers"), which in turn have the largest positive impact on the chance that a project becomes a "Top Project".

Nevertheless, the network effect of commitment on time-bound social metrics could not entirely be explained. There is a strong evidence that earlier commitment of firm employed developers has an affirmative influence of later / long-term project's success, but furthers investigation and more sophisticated models are necessary to confirm the final analysis.

4.10 H 3: If firm employees' contribution share is higher overall the more likely the project is popular

In the following, plots and tables will investigate the impact of "Ratio" (i.e. share of firm employed developers on code contributions) on projects' social success ("Stars", "Subscribers", "Forks", "Number of Issues", "Number of contributors" and is "Top Project"). We assume that "Age" (in days) has a positive influence on social success metrics, too [74]. "Number of Contributors" is just a validation of data and model fit, since it should *not* be influenced necessarily by "Ratio" (but "Age").

We assume that firms and programming languages have an impact on the mentioned attributes as well [75]. Thus, we introduce dummy variables for *Programming Languages* (10 in total) and *Firms* (58 in total) to achieve possibly better model fits. Beside the linear regression we use linear mixed-effects models (Bates et al. (2015)) (see table15 and 44) to consider effects between "Firms" and "Programming Languages" (similar approach as dummy variables).

Detailed regression tables regarding to project size and popularity (with dummy variables for *programming languages* and *firms*) can be looked up in table 42 and 43 on page 101.

See listing 7 on page 65 for models defined in R code.

4.10.1 H 3.1: Plots

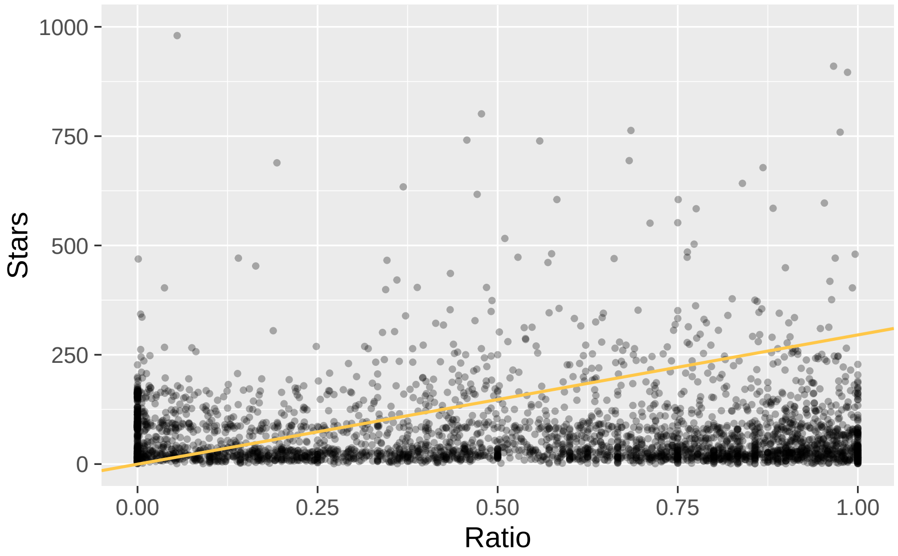

Figure 12: Correlation coefficient (yellow slope) of "Ratio" on "Stars" (Model 3.1)

[74] because the older a project gets, the higher the number of "Stars", "Subscribers" etc. might be

[75] "Top Projects" are selected by programming languages, but "Residual Projects" need to be considered as well

4.10.2 H 3.1: Regression Tables

	Dependent variable:											
	Stars				Subscribers				Forks			
	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)	(3.9)	(3.10)	(3.11)	(3.12)
Ratio	295.609***	278.398***	129.899	152.407*	7.184	7.068	18.616***	17.985***	51.697***	39.361**	26.293	29.970
	(75.069)	(79.011)	(83.121)	(83.466)	(6.010)	(6.284)	(6.284)	(6.322)	(18.318)	(19.329)	(20.865)	(21.002)
Age	0.367***	0.437***	0.443***	0.494***	0.031***	0.036***	0.037***	0.039***	0.129***	0.143***	0.151***	0.158***
	(0.047)	(0.049)	(0.050)	(0.051)	(0.004)	(0.004)	(0.004)	(0.004)	(0.011)	(0.012)	(0.013)	(0.013)
Constant	−0.236	−70.590	−434.208***	−764.217***	48.363***	47.700***	−20.544*	−39.926***	−23.486	−43.139	−113.655***	−192.851***
	(60.212)	(154.968)	(139.704)	(195.611)	(4.821)	(12.325)	(10.562)	(14.817)	(14.692)	(37.910)	(35.069)	(49.222)
Observations	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−29,905.750	−29,877.320	−29,692.650	−29,668.010	−21,273.000	−21,221.860	−20,863.760	−20,845.730	−25,083.110	−25,063.340	−24,966.870	−24,950.480
Akaike Inf. Crit.	59,817.500	59,778.650	59,505.300	59,474.010	42,552.000	42,467.720	41,847.520	41,829.460	50,172.220	50,150.680	50,053.740	50,038.950
Note:												*p<0.1; **p<0.05; ***p<0.01

Table 13: Impact of ”Age” and ”Ratio” on ”Stars”, ”Subscribers” and ”Forks” through *Fitting Generalized Linear Models* with dummy variables for *Firms* and *Programming Languages*. Using (a) dummy variables for each *Firm* in model 3.3, 3.7 and 3.11 (b) dummy variables for each *Language* in model 3.2, 3.6, 3.10 (c) dummy variables for each *Firm and Language* in model 3.4, 3.8 and 3.12

See regression table 45 and 46 (page 112 - 113) for results having ”Ratio” as dependent variable and table 44 (page 111) for *Mixed Model* between Firms and Programming Languages.

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
Ratio	2.147	2.006	7.861*	7.612*	−2.417	−2.647	1.391	1.402	2.025***	1.727***	1.906***	1.753***
	(3.717)	(3.930)	(4.235)	(4.275)	(1.911)	(2.005)	(2.051)	(2.067)	(0.155)	(0.166)	(0.191)	(0.197)
Age	0.015***	0.017***	0.021***	0.022***	0.018***	0.019***	0.020***	0.021***	0.001***	0.001***	0.001***	0.001***
	(0.002)	(0.002)	(0.003)	(0.003)	(0.001)	(0.001)	(0.001)	(0.001)	(0.0001)	(0.0001)	(0.0001)	(0.0001)
Constant	8.443***	−2.780	−19.593***	−34.006***	5.114***	4.488	−14.598***	−20.129***	−3.426***	−2.546***	−5.820***	−6.276***
	(2.981)	(7.707)	(7.118)	(10.018)	(1.533)	(3.933)	(3.447)	(4.845)	(0.140)	(0.251)	(0.437)	(0.529)
Observations	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−19,630.080	−19,616.630	−19,514.800	−19,507.690	−17,355.340	−17,316.430	−17,035.330	−17,023.880	−1,458.752	−1,373.897	−1,208.251	−1,150.787
Akaike Inf. Crit.	39,266.160	39,257.260	39,149.610	39,153.370	34,716.680	34,656.870	34,190.660	34,185.760	2,923.504	2,771.795	2,536.502	2,439.574
Note:												*p<0.1; **p<0.05; ***p<0.01

Table 14: Impact of ”Age” and ”Ratio” on ”Issues”, ”Contributors” and ”Popularity” through *Fitting Generalized Linear Models.* Using (a) dummy variables for each *Firm* in model 3.15, 3.19 and 3.23 (b) dummy variables for each *Language* in model 3.14, 3.18, 3.22 (c) dummy variables for each *Firm and Language* in model 3.16, 3.20 and 3.24

	Dependent variable:					
	Stars	Subscribers	Forks	Number of Issues	Number of Contributors	Top Project
	linear *mixed-effects*	*linear* *mixed-effects*	*linear* *mixed-effects*	*linear* *mixed-effects*	*linear* *mixed-effects*	*generalized linear* *mixed-effects*
	(3.25)	(3.26)	(3.27)	(3.28)	(3.29)	(3.30)
Ratio	149.245^{*}	18.379^{***}	31.783	5.896	1.235	1.910^{***}
	(82.032)	(6.237)	(20.266)	(4.111)	(2.032)	(0.185)
Age	0.420^{***}	0.036^{***}	0.142^{***}	0.020^{***}	0.020^{***}	0.001^{***}
	(0.049)	(0.004)	(0.012)	(0.002)	(0.001)	(0.0001)
Constant	160.468	44.443^{***}	-9.606	2.911	2.225	-3.497^{***}
	(104.513)	(9.753)	(21.187)	(4.280)	(2.974)	(0.235)
Observations	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−29,774.830	−20,966.900	−25,031.110	−19,582.970	−17,137.260	−1,298.686
Akaike Inf. Crit.	59,559.660	41,943.800	50,072.220	39,175.940	34,284.520	2,605.372
Bayesian Inf. Crit.	59,590.340	41,974.490	50,102.910	39,206.620	34,315.210	2,629.921
Note:						*p<0.1; **p<0.05; ***p<0.01

Table 15: Impact of ”Age” and ”Ratio” on projects' social metrics (”Stars”, ”Subscribers”, ”Forks” and ”Issues”) and popularity (is ”Top Project” yes / no) through *Mixed Model* between *Firms*. ”Number of Contributors” is just a control attribute and not a describing model.

4.11 H 3: Results and Interpretation

The ”Ratio” (i.e. the share of code commits by firm employed developers) has a significant positive impact on the popularity of a project.

If we use ”Stars” as a proxy for popularity, we can approve that the ”Ratio” has a high positive influence on the likelihood that a project is (very) popular (correlation coefficient is 295, see model 3.1 in table 13). This extends to ”Subscribers” as well but with a lower impact (see model 3.7 and 3.8 in table 13). ”Forks” are also positively influenced by ”Ratio” but also with a much lower correlation coefficient (see model 3.10 in table 13). Interestingly enough, ”Ratio” has no (significant) impact on the number of issues - but issue participation share has an impact on success and popularity (as shown in chapter 4.5.7).

Unfortunately, dummy variables (for *Firms* and *Programming Languages* respectively) have diverse effects on the regression results. This might be caused by different popularity attributes which are effected by different users' behavior. For example: *JavaScript* developers could be more ”Stargazer” friendly, *C++* developer user more ”Subscriber” friendly (and so on). Firms also may have different effects on specific popularity attributes: Some are more popular and some are less popular in the OS community (see the example of *Microsoft* and *GitHub* in chapter 3.16).

Another broad hint is given by the *logistic regression models*: **The probability that a project is a ”Top Project” depends significant on the size of ”Ratio”** (see model 3.21 - 3.24 in table 14). By using logistics regression we can interpret β_j through $e^{\beta j}$ as effect coefficient (Agresti et al., 2007, p. 71). Thus, the effect coefficient is between $e^{1.727} \approx 5.62$ and $e^{1.91} \approx 6.75$ [76]. **This means that projects with highest ”Ratios” are more likely ”Top Projects” by the factor of 6.**

[76] the problem of multicollinearity might reduce the validity of the mentioned effect coefficients, but seems to be rejected by results in observations over time (see chapter 4.7)

5 Results and Conclusion

All hypotheses are mainly confirmed by the empirical findings:

- Through their code contribution firm employed developers have a strong positive impact on the code contribution of external developers
- Firm employed developers have a strong positive impact by participating on issues and comments in issues by causing more participation of external users
- The first year seems to be the most important time period for firm employed developers' participation: High contribution ratio in the beginning benefits popularity and success of the OS project in the long run
- The higher the overall share of contribution from firm employed developers, the more likely the project is successful and accepted by the OS community

Nevertheless, influence of higher participation of firm employed developers could not be verified ultimately on some continuous social metrics [77] which might be caused by broad difference of popularity and level of awareness between firms and software segments.

Beside the regression results we also receive interesting fringe insights: According to the frequency of commits with respect to daytime we can assume, that most firms and OS developers do work in USA, Canada and Europe. Also do external developers seem to be employed as well. Most developers commit source code at the usual working time period (9 a.m. - 5 p.m.) and "external developers" do not differ significantly from as "firm employed" classified developers. This would imply, that there is a stronger collaboration between firms than between firms and "libertine" developers.

5.1 Implications and Managerial Advice

Firms should initiate and invest [78] in their OS projects. The share of code contribution by firm employees should be very high - especially in the beginning - to achieve a broad acceptance by the OS community and developers on (rival) firms [79]. The return of investment will be a multiplier effect on commitment [80] by external developers and a gain on level of awareness. Beside that, firms' OS commitment might lead to more cross-firm collaboration, improve employment of external developers and bring forward establishing new technologies in smaller amounts of time. If the share of firm employed developers is higher than usual, projects seem more likely to reach a certain degree

[77] *Forks* and *Number of Issues* on some models

[78] with funds and man-power

[79] contribution in the beginning can be interpreted as groundwork that needs to be done by firm employed developers to build a solid software foundation

[80] via source code contribution and participation in communication channels

of quality, maturity and attention. As result the community is willing to contribute their professional expertise for free to the project.

5.2 Further Research

The empirical data implies further research with respect to broader observations of firms and projects. By analyzing more firms and projects the hypotheses could eventually be applied more generally to software technology driven firms. Consideration of financial data of firms could reveal the impact of OS commitment on firms' revenue. And more sophisticated employees firm rating would help to investigate the coherence of OS commitment and quality of work environment. Furthermore, observations of network relations between firms, developers and users would improve the understanding of the importance of collaboration and exchange of knowledge.

A Listings

Listing 1: BigQuery via GitHubArchive

```
-- # Using data from https://www.githubarchive.org/
--   1. Login into the Google Developer Console
--      (https://console.developers.google.com/)
--   2. Create a project
--      (https://developers.google.com/console/help/#creatingdeletingprojects)
--   3. Activate the BigQuery API
--      (https://developers.google.com/console/help/#activatingapis)
--   4. Open public dataset
--      (https://bigquery.cloud.google.com/table/githubarchive:day.
    events_20150101)

-- All data is collected since 2011 - 2014

-- Count GitHub users

SELECT COUNT(DISTINCT actor) FROM [githubarchive:github.timeline]

--→ 4525103
```

Listing 2: Structure of a git commit

```
commit a35c785202ef7351f1e071a0191b201b745da168
Merge: fbb9327 12667fc
Author: Name <email>
Date:   Sun Jul 26 21:10:39 2015 -0400

  Message / description of the commit
```

Listing 3: Linear regression models in R for H 1.1

```
linear.1.1.1 <- lm(formula = `ext. commits` ~ `int. commits`, data =
    codeContributions)
linear.1.1.2 <- lm(formula = `int. commits` ~ `ext. commits`, data =
    codeContributions)
linear.1.1.3 <- lm(formula = `ext. commits` ~ `int. commits`, data =
    codeContributions[codeContributions$is_top_project == TRUE,])
linear.1.1.4 <- lm(formula = `int. commits` ~ `ext. commits`, data =
    codeContributions[codeContributions$is_top_project == TRUE,])
linear.1.1.5 <- lm(formula = `ext. commits` ~ `int. commits`, data =
    codeContributions[codeContributions$is_top_project == FALSE,])
linear.1.1.6 <- lm(formula = `int. commits` ~ `ext. commits`, data =
    codeContributions[codeContributions$is_top_project == FALSE,])
```

```r
linear.1.1.7 <- lm(formula = `ext. commits` ~ `int. commits`, data =
    codeContributions[codeContributions$is_top_project == TRUE &
    codeContributions$age >= 365,])
linear.1.1.8 <- lm(formula = `int. commits` ~ `ext. commits`, data =
    codeContributions[codeContributions$is_top_project == TRUE &
    codeContributions$age >= 365,])
linear.1.1.9 <- lm(formula = `ext. commits` ~ `int. commits`, data =
    codeContributions[codeContributions$is_top_project == FALSE &
    codeContributions$age >= 365,])
linear.1.1.10 <- lm(formula = `int. commits` ~ `ext. commits`, data =
    codeContributions[codeContributions$is_top_project == FALSE &
    codeContributions$age >= 365,])
linear.1.1.11 <- lm(formula = `ext. commits` ~ `int. commits`, data =
    codeContributions[codeContributions$is_top_project == TRUE &
    codeContributions$age < 365,])
linear.1.1.12 <- lm(formula = `int. commits` ~ `ext. commits`, data =
    codeContributions[codeContributions$is_top_project == TRUE &
    codeContributions$age < 365,])
linear.1.1.13 <- lm(formula = `ext. commits` ~ `int. commits`, data =
    codeContributions[codeContributions$is_top_project == FALSE &
    codeContributions$age < 365,])
linear.1.1.14 <- lm(formula = `int. commits` ~ `ext. commits`, data =
    codeContributions[codeContributions$is_top_project == FALSE &
    codeContributions$age < 365,])
```

Listing 4: Linear regression models in R for H 1.2

```r
linear.1.2.1 <- lm(formula = issues_count_by_external_developer ~
    issues_count_by_firm_employed_developer, data = issues)
linear.1.2.2 <- lm(formula = issues_count_by_firm_employed_developer ~
    issues_count_by_external_developer, data = issues)
linear.1.2.3 <- lm(formula = issues_count_by_external_developer ~
    issues_count_by_firm_employed_developer, data =
    issues[issues$is_top_project == TRUE,])
linear.1.2.4 <- lm(formula = issues_count_by_firm_employed_developer ~
    issues_count_by_external_developer, data = issues[issues$is_top_project ==
     TRUE,])
linear.1.2.5 <- lm(formula = issues_count_by_external_developer ~
    issues_count_by_firm_employed_developer, data =
    issues[issues$is_top_project == FALSE,])
linear.1.2.6 <- lm(formula = issues_count_by_firm_employed_developer ~
    issues_count_by_external_developer, data = issues[issues$is_top_project ==
     FALSE,])
linear.1.2.7 <- lm(formula = issues_count_by_external_developer ~
    issues_count_by_firm_employed_developer, data =
    issues[issues$is_top_project == TRUE & issues$age >= 365,])
linear.1.2.8 <- lm(formula = issues_count_by_firm_employed_developer ~
    issues_count_by_external_developer, data = issues[issues$is_top_project ==
     TRUE & issues$age >= 365,])
linear.1.2.9 <- lm(formula = issues_count_by_external_developer ~
    issues_count_by_firm_employed_developer, data =
    issues[issues$is_top_project == FALSE & issues$age >= 365,])
linear.1.2.10 <- lm(formula = issues_count_by_firm_employed_developer ~
    issues_count_by_external_developer, data = issues[issues$is_top_project ==
     FALSE & issues$age >= 365,])
linear.1.2.11 <- lm(formula = issues_count_by_external_developer ~
    issues_count_by_firm_employed_developer, data =
    issues[issues$is_top_project == TRUE & issues$age < 365,])
linear.1.2.12 <- lm(formula = issues_count_by_firm_employed_developer ~
    issues_count_by_external_developer, data = issues[issues$is_top_project ==
     TRUE & issues$age < 365,])
linear.1.2.13 <- lm(formula = issues_count_by_external_developer ~
    issues_count_by_firm_employed_developer, data =
    issues[issues$is_top_project == FALSE & issues$age < 365,])
linear.1.2.14 <- lm(formula = issues_count_by_firm_employed_developer ~
    issues_count_by_external_developer, data = issues[issues$is_top_project ==
     FALSE & issues$age < 365,])
```

Listing 5: Linear regression models in R for H 1.3

```
linear.1.3.1 <- lm(formula = issue_comments_count ~
    content_share_by_firm_employed_developer, data = observedIssuesComments)
linear.1.3.2 <- lm(formula = comments_count_by_ext ~
    content_share_by_firm_employed_developer, data = observedIssuesComments)
linear.1.3.3 <- lm(formula = comments_count_by_ext ~ comments_count_by_int,
    data = observedIssuesComments)
linear.1.3.4 <- lm(formula = comments_count_by_int ~ comments_count_by_ext,
    data = observedIssuesComments)
linear.1.3.5 <- lm(formula = issue_comments_count ~
    content_share_by_firm_employed_developer, data =
    subset(observedIssuesComments, issuesComments$is_top_project == TRUE))
linear.1.3.6 <- lm(formula = comments_count_by_ext ~
    content_share_by_firm_employed_developer, data =
    subset(observedIssuesComments, issuesComments$is_top_project == TRUE))
linear.1.3.7 <- lm(formula = comments_count_by_ext ~ comments_count_by_int,
    data = subset(observedIssuesComments, issuesComments$is_top_project ==
    TRUE))
linear.1.3.8 <- lm(formula = comments_count_by_int ~ comments_count_by_ext,
    data = subset(observedIssuesComments, issuesComments$is_top_project ==
    TRUE))
```

```
linear.1.3.9 <- lm(formula = issue_comments_count ~
    content_share_by_firm_employed_developer, data =
    subset(observedIssuesComments, issuesComments$is_top_project == FALSE))
linear.1.3.10 <- lm(formula = comments_count_by_ext ~
    content_share_by_firm_employed_developer, data =
    subset(observedIssuesComments, issuesComments$is_top_project == FALSE))
linear.1.3.11 <- lm(formula = comments_count_by_ext ~ comments_count_by_int,
    data = subset(observedIssuesComments, issuesComments$is_top_project ==
    FALSE))
linear.1.3.12 <- lm(formula = comments_count_by_int ~ comments_count_by_ext,
    data = subset(observedIssuesComments, issuesComments$is_top_project ==
    FALSE))
```

Listing 6: Linear regression models in R for H 2.1

```
linear.2.1.1 <- glm(formula = is_top_project ~ age + ratio.1,
    family=binomial(link='logit'), data = observations )
linear.2.1.2 <- glm(formula = is_top_project ~ age + ratio.2,
    family=binomial(link='logit'), data = observations )
linear.2.1.3 <- glm(formula = is_top_project ~ age + ratio.3,
    family=binomial(link='logit'), data = observations )
linear.2.1.4 <- glm(formula = is_top_project ~ age + ratio.4,
    family=binomial(link='logit'), data = observations )

linear.2.1.5 <- glm(formula = is_top_project ~ age + subscribers.1,
    family=binomial(link='logit'), data = observations )
linear.2.1.6 <- glm(formula = is_top_project ~ age + subscribers.2,
    family=binomial(link='logit'), data = observations )
linear.2.1.7 <- glm(formula = is_top_project ~ age + subscribers.3,
    family=binomial(link='logit'), data = observations )
linear.2.1.8 <- glm(formula = is_top_project ~ age + subscribers.4,
    family=binomial(link='logit'), data = observations )
linear.2.1.9 <- glm(formula = is_top_project ~ age + subscribers.5,
    family=binomial(link='logit'), data = observations )
linear.2.1.10 <- glm(formula = is_top_project ~ age + subscribers.today,
    family=binomial(link='logit'), data = observations )

linear.2.1.11 <- glm(formula = is_top_project ~ age + forks.1,
    family=binomial(link='logit'), data = observations )
linear.2.1.12 <- glm(formula = is_top_project ~ age + forks.2,
    family=binomial(link='logit'), data = observations )
linear.2.1.13 <- glm(formula = is_top_project ~ age + forks.3,
    family=binomial(link='logit'), data = observations )
linear.2.1.14 <- glm(formula = is_top_project ~ age + forks.4,
    family=binomial(link='logit'), data = observations )
```

```
linear.2.1.15 <- glm(formula = is_top_project ~ age + forks.5,
    family=binomial(link='logit'), data = observations )
linear.2.1.16 <- glm(formula = is_top_project ~ age + forks.today,
    family=binomial(link='logit'), data = observations )

linear.2.1.17 <- glm(formula = is_top_project ~ age + forks.1_2,
    family=binomial(link='logit'), data = observations )
linear.2.1.18 <- glm(formula = is_top_project ~ age + forks.2_3,
    family=binomial(link='logit'), data = observations )
linear.2.1.19 <- glm(formula = is_top_project ~ age + forks.3_4,
    family=binomial(link='logit'), data = observations )
linear.2.1.20 <- glm(formula = is_top_project ~ age + forks.4_5,
    family=binomial(link='logit'), data = observations )

linear.2.1.21 <- glm(formula = forks.1 ~ age + ratio.1, data = observations )
linear.2.1.22 <- glm(formula = forks.2 ~ age + ratio.2, data = observations )
linear.2.1.23 <- glm(formula = forks.3 ~ age + ratio.3, data = observations )
linear.2.1.24 <- glm(formula = forks.4 ~ age + ratio.4, data = observations )
linear.2.1.25 <- glm(formula = forks.2 ~ age + ratio.1, data = observations )
linear.2.1.26 <- glm(formula = forks.3 ~ age + ratio.2, data = observations )
linear.2.1.27 <- glm(formula = forks.4 ~ age + ratio.3, data = observations )
linear.2.1.28 <- glm(formula = forks.5 ~ age + ratio.4, data = observations )
linear.2.1.29 <- glm(formula = forks.3 ~ age + ratio.1, data = observations )
linear.2.1.30 <- glm(formula = forks.4 ~ age + ratio.2, data = observations )
linear.2.1.31 <- glm(formula = forks.5 ~ age + ratio.3, data = observations )

linear.2.1.32 <- glm(formula = subscribers.1 ~ age + ratio.1, data =
    observations )
linear.2.1.33 <- glm(formula = subscribers.2 ~ age + ratio.2, data =
    observations )
linear.2.1.34 <- glm(formula = subscribers.3 ~ age + ratio.3, data =
    observations )
linear.2.1.35 <- glm(formula = subscribers.4 ~ age + ratio.4, data =
    observations )
linear.2.1.36 <- glm(formula = subscribers.2 ~ age + ratio.1, data =
    observations )
linear.2.1.37 <- glm(formula = subscribers.3 ~ age + ratio.2, data =
    observations )
linear.2.1.38 <- glm(formula = subscribers.4 ~ age + ratio.3, data =
    observations )
linear.2.1.39 <- glm(formula = subscribers.5 ~ age + ratio.4, data =
    observations )
linear.2.1.40 <- glm(formula = subscribers.3 ~ age + ratio.1, data =
    observations )
```

```
linear.2.1.41 <- glm(formula = subscribers.4 ~ age + ratio.2, data =
    observations )
linear.2.1.42 <- glm(formula = subscribers.5 ~ age + ratio.3, data =
    observations )

linear.2.1.43  <- glm(formula = forks.today ~ age + ratio.1, data =
    observations )
linear.2.1.44  <- glm(formula = forks.today ~ age + ratio.2, data =
    observations )
linear.2.1.45  <- glm(formula = forks.today ~ age + ratio.3, data =
    observations )
linear.2.1.46  <- glm(formula = forks.today ~ age + ratio.4, data =
    observations )

linear.2.1.47  <- glm(formula = subscribers.today ~ age + ratio.1, data =
    observations )
linear.2.1.48  <- glm(formula = subscribers.today ~ age + ratio.2, data =
    observations )
linear.2.1.49  <- glm(formula = subscribers.today ~ age + ratio.3, data =
    observations )
linear.2.1.50  <- glm(formula = subscribers.today ~ age + ratio.4, data =
    observations )

linear.2.1.51  <- glm(formula = forks.today ~ age + ratio.1_2, data =
    observations )
linear.2.1.52  <- glm(formula = forks.today ~ age + ratio.2_3, data =
    observations )
linear.2.1.53  <- glm(formula = forks.today ~ age + ratio.3_4, data =
    observations )
linear.2.1.54  <- glm(formula = forks.today ~ age + ratio.4_5, data =
    observations )

linear.2.1.55  <- glm(formula = subscribers.today ~ age + ratio.1_2, data =
    observations )
linear.2.1.56  <- glm(formula = subscribers.today ~ age + ratio.2_3, data =
    observations )
linear.2.1.57  <- glm(formula = subscribers.today ~ age + ratio.3_4, data =
    observations )
linear.2.1.58  <- glm(formula = subscribers.today ~ age + ratio.4_5, data =
    observations )

linear.2.1.59 <- glm(formula = is_top_project ~ age + ratio.1_2,
    family=binomial(link='logit'), data = observations )
linear.2.1.60 <- glm(formula = is_top_project ~ age + ratio.2_3,
    family=binomial(link='logit'), data = observations )
linear.2.1.61 <- glm(formula = is_top_project ˜ age + ratio.3_4,
    family=binomial(link='logit'), data = observations )
linear.2.1.62 <- glm(formula = is_top_project ˜ age + ratio.4_5,
    family=binomial(link='logit'), data = observations )
```

Listing 7: Linear regression models in R for H 3.1

```
linear.3.1 <- glm(formula = stargazers_count ˜ ratio + age, data =
    contributions )
linear.3.2 <- glm(formula = stargazers_count ˜ ratio + age + lang_, data =
    contributions )
linear.3.3 <- glm(formula = stargazers_count ˜ ratio + age + firm_, data =
    contributions )
linear.3.4 <- glm(formula = stargazers_count ˜ ratio + age + firm_ + lang_,
    data = contributions )
linear.3.5 <- glm(formula = subscribers_count ˜ ratio + age, data =
    contributions )
linear.3.6 <- glm(formula = subscribers_count ˜ ratio + age + lang_, data =
    contributions )
linear.3.7 <- glm(formula = subscribers_count ˜ ratio + age + firm_, data =
    contributions )
linear.3.8 <- glm(formula = subscribers_count ˜ ratio + age + firm_ + lang_,
    data = contributions )
linear.3.9 <- glm(formula = forks_count ˜ ratio + age , data = contributions)
linear.3.10 <- glm(formula = forks_count ˜ ratio + age + lang_, data =
    contributions)
linear.3.11 <- glm(formula = forks_count ˜ ratio + age + firm_, data =
    contributions)
linear.3.12 <- glm(formula = forks_count ˜ ratio + age + firm_ + lang_, data =
     contributions)

linear.3.13 <- glm(formula = all_issues_count ˜ ratio + age , data =
    contributions)
linear.3.14 <- glm(formula = all_issues_count ˜ ratio + age + lang_, data =
    contributions)
linear.3.15 <- glm(formula = all_issues_count ˜ ratio + age + firm_, data =
    contributions)
linear.3.16 <- glm(formula = all_issues_count ˜ ratio + age + firm_ + lang_,
    data = contributions)
linear.3.17 <- glm(formula = contributors_count ˜ ratio + age , data =
    contributions)
linear.3.18 <- glm(formula = contributors_count ˜ ratio + age + lang_, data =
    contributions)
linear.3.19 <- glm(formula = contributors_count ˜ ratio + age + firm_, data =
    contributions)
linear.3.20 <- glm(formula = contributors_count ~ ratio + age + firm_ + lang_,
     data = contributions)
linear.3.21 <- glm(formula = top_repo ~ ratio + age , family=binomial(link='
    logit'), data = contributions )
linear.3.22 <- glm(formula = top_repo ~ ratio + age + lang_,
    family=binomial(link='logit'), data = contributions )
linear.3.23 <- glm(formula = top_repo ~ ratio + age + firm_,
    family=binomial(link='logit'), data = contributions )
linear.3.24 <- glm(formula = top_repo ~ ratio + age + firm_ + lang_,
    family=binomial(link='logit'), data = contributions )

mixedeffect.3.25 <- lmer(formula = stargazers_count ~ ratio + age + (1 |
    organization_name), data = contributions)
mixedeffect.3.26 <- lmer(formula = subscribers_count ~ ratio + age + (1 |
    organization_name), data = contributions)
mixedeffect.3.27 <- lmer(formula = forks_count ~ ratio + age + (1 |
    organization_name), data = contributions)
mixedeffect.3.28 <- lmer(formula = all_issues_count ~ ratio + age + (1 |
    organization_name), data = contributions)
mixedeffect.3.29 <- lmer(formula = contributors_count ~ ratio + age + (1 |
    organization_name), data = contributions)
mixedeffect.3.30 <- lmer(formula = top_repo ~ ratio + age + (1 |
    organization_name), family=binomial(link='logit'), data = contributions)

mixedeffect.3.31 <- lmer(formula = stargazers_count ~ ratio + age + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.32 <- lmer(formula = subscribers_count ~ ratio + age + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.33 <- lmer(formula = forks_count ~ ratio + age + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.34 <- lmer(formula = all_issues_count ~ ratio + age + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.35 <- lmer(formula = contributors_count ~ ratio + age + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.36 <- glmer(formula = top_repo ~ ratio + age + (1 |
    organization_name) + (1 | language), family=binomial(link='logit'), data =
     contributions)

mixedeffect.3.37 <- lmer(formula = ratio ~ stargazers_count + (1 |
    organization_name), data = contributions)
mixedeffect.3.38 <- lmer(formula = ratio ~ subscribers_count + (1 |
    organization_name), data = contributions)
mixedeffect.3.39 <- lmer(formula = ratio ~ forks_count + (1 |
    organization_name), data = contributions)
mixedeffect.3.40 <- lmer(formula = ratio ~ all_issues_count + (1 |
    organization_name), data = contributions)
mixedeffect.3.41 <- lmer(formula = ratio ~ contributors_count + (1 |
    organization_name), data = contributions)
mixedeffect.3.42 <- lmer(formula = ratio ~ top_repo + (1 | organization_name),
     data = contributions)

mixedeffect.3.43 <- lmer(formula = ratio ~ stargazers_count + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.44 <- lmer(formula = ratio ~ subscribers_count + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.45 <- lmer(formula = ratio ~ forks_count + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.46 <- lmer(formula = ratio ~ all_issues_count + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.47 <- lmer(formula = ratio ~ contributors_count + (1 |
    organization_name) + (1 | language), data = contributions)
mixedeffect.3.48 <- lmer(formula = ratio ~ top_repo + (1 | organization_name)
    + (1 | language), data = contributions)
```

B Tables

B.1 Introduction and Theory

	Linux	FreeBSD	Unknown	MS.Windows
W3Techs (02-2015)	35.9%	0.95%	30.9%	32.3%
W3cook (05-2015)	96.6%	1.7%	0%	1.7%

Table 16: Share of Operating Systems on public Internet Servers; *Sources:* `http://w3techs.com/technologies/overview/operating_system/all`, `http://w3techs.com/technologies/details/os-unix/all/all`, `http://www.w3cook.com/os/summary/`

Statistic	N	Mean	St. Dev.	Min	Max
Top Project (yes/no)	177	0.181	0.386	0	1
Ratio	177	0.684	0.334	0.000	1.000
Age (days)	177	294.260	226.632	20	1,466
Number of Commits	177	1,241.955	10,558.430	3	127,063
Firm Employees' Commits	177	138.073	307.597	0	2,573
External Developers' Commits	177	1,103.881	10,548.850	0	126,912
Stars	177	186.051	580.191	0	4,909
Contributors	177	9.949	27.287	2	262
Subscribers	177	50.655	71.049	2	585
Forks	177	45.153	155.701	0	1,812
Number of Issues	177	15.429	43.100	1	420

Table 17: Statistic of Projects by Microsoft

Statistic	N	Mean	St. Dev.	Min	Max
Top Project (yes/no)	40	0.400	0.496	0	1
Ratio	40	0.528	0.333	0.000	1.000
Age (days)	40	1,039.925	615.875	62	2,449
Number of Commits	40	1,784.450	7,037.635	5	44,486
Firm Employees' Commits	40	290.175	516.544	0	1,905
External Developers' Commits	40	1,494.275	7,021.648	0	44,482
Stars	40	993.175	1,705.672	1	7,922
Contributors	40	56.625	114.295	2	451
Subscribers	40	62.750	62.040	6	233
Forks	40	244.350	518.172	1	2,611
Number of Issues	40	20.500	43.776	1	242

Table 18: Statistic of Projects by GitHub

Statistic	N	Mean	St. Dev.	Min	Max
Top Project (yes/no)	103	0.534	0.501	0	1
Ratio	103	0.591	0.342	0.000	1.000
Age (days)	103	731.204	517.623	79	2,489
Number of Commits	103	1,103.039	4,112.523	5	36,596
Firm Employees' Commits	103	559.748	1,682.788	0	14,499
External Developers' Commits	103	543.291	3,553.457	0	35,872
Stars	103	2,360.272	4,762.077	12	35,214
Contributors	103	36.767	70.711	2	468
Subscribers	103	188.990	347.568	16	2,617
Forks	103	396.107	854.878	2	5,716
Number of Issues	103	56.029	161.550	1	1,043

Table 19: Statistic of Projects by facebook

B.2 Data and Basic Statistics Tables

	Name of Organization on GitHub	Top Repositories	is commercial Organization
1	google	85	yes
2	facebook	56	yes
3	Microsoft	33	yes
4	square	30	yes
5	apache	23	partly
6	aspnet	23	partly
7	thoughtbot	23	yes
8	alibaba	19	yes
9	mozilla	18	partly
10	twitter	18	yes
11	Netflix	17	yes
12	github	16	yes
13	mongodb	16	yes
14	mono	16	no
15	thephpleague	16	no
16	docker	15	yes
17	elastic	15	yes
18	hashicorp	15	yes
19	golang	14	partly
20	Yalantis	13	yes
21	dotnet	12	partly
22	rails	12	partly
23	airbnb	10	yes
24	aws	10	yes
25	etsy	10	yes
26	googlesamples	10	yes
27	linkedin	10	yes
28	xamarin	10	yes
29	zeromq	10	no
30	coreos	9	partly
31	laravel	9	no
32	Shopify	9	yes
33	spring-projects	9	partly
34	adafruit	8	yes
35	bitly	8	yes
36	dmlc	8	no
37	doctrine	8	no
38	facebookarchive	8	partly
39	fastlane	8	no
40	jquery	8	no
41	KnpLabs	8	yes

	Name of Organization on GitHub	Top Repositories	is commercial Organization
42	shadowsocks	8	no
43	Yelp	8	yes
44	apple	7	yes
45	Automattic	7	yes
46	Azure	7	yes
47	cocos2d	7	no
48	couchbase	7	no
49	douban	7	yes
50	FriendsOfPHP	7	no
51	id-Software	7	yes
52	libgit2	7	no
53	stripe	7	yes
54	cloudera	6	yes
55	dropbox	6	yes
56	enormego	6	yes
57	Homebrew	6	no
58	icsharpcode	6	no
59	msgpack	6	no
60	nodejs	6	no
61	openresty	6	no
62	ParsePlatform	6	yes
63	raspberrypi	6	partly
64	spotify	6	yes
65	twilio	6	yes
66	yahoo	6	yes
67	android	5	partly
68	angular	5	partly
69	angular-ui	5	partly
70	applidium	5	yes
71	celluloid	5	no
72	cesanta	5	yes
73	cucumber	5	yes
74	FriendsOfSymfony	5	no
75	heroku	5	yes
76	JetBrains	5	partly
77	paypal	5	yes
78	plataformatec	5	yes
79	RailsApps	5	no
80	rspec	5	no
81	ServiceStack	5	yes
82	socketio	5	no
83	ValveSoftware	5	yes
84	VerbalExpressions	5	no

	Name of Organization on GitHub	Top Repositories	is commercial Organization
85	visionmedia	5	no
86	activerecord-hackery	4	no
87	CakeDC	4	partly
88	castleproject	4	no
89	chef	4	yes
90	collectiveidea	4	yes
91	composer	4	no
92	docopt	4	no
93	documentcloud	4	partly
94	Flipboard	4	yes
95	forkingdog	4	no
96	getsentry	4	yes
97	gliderlabs	4	yes
98	gorilla	4	no
99	Instagram	4	yes
100	intridea	4	yes
101	mapbox	4	yes
102	mutualmobile	4	yes
103	openstack	4	yes
104	owncloud	4	yes
105	phacility	4	yes
106	Qihoo360	4	yes
107	Reactive-Extensions	4	yes
108	sass	4	no
109	sourcegraph	4	yes
110	symfony	4	partly
111	tumblr	4	yes
112	venmo	4	yes
113	yhat	4	yes

Table 20: All Organizations having at least 4 top repositories

	Firm	Repos	JS	ObjC	Go	C	Python	Ruby	Java	PHP	C++	C#	CS	Scala	Hack
1	google	79	6		15	6	8	1	16	3	23	1			
2	facebook	58	10	9		7	4		10	2	14				2
3	twitter	48	3	2		4		3	4		2			30	
4	Microsoft	21	1			2					8	10			
5	github	20	1	1	3	1		8				2	4		
6	alibaba	20				4			12		4				
7	linkedin	12	2	2					7				1		
8	aws [81]	11	1	1	2		1	2	1	2		1			
9	yahoo	6	2		1	1			1					1	
10	groupon	4											4		
11	NetEase	2	1									1			
12	yandex	2					1				1				
13	eBay	2					2								
14	SonyWWS	2										2			
15	sony	1			1										
16	awslabs [82]	1				1									
17	intel-iot-devkit	1				1									
18	forcedotcom [83]	1							1						
19	amazonwebservices [84]	1								1					
20	developerforce [85]	1										1			

Table 21: Commercial most succesfull and most popular companies and their *GitHub* projects' programming languages. *Source: GitHub API, October 2015*

	Name	Name on GitHub	Public Repositories	Top Repositories	On GitHub since
1	Adafruit Industries	adafruit	469	8	2010
2	Airbnb	airbnb	86	10	2011
3	Alibaba	alibaba	87	19	2012
4	Apple	apple	18	7	2015
5	Applidium	applidium	15	5	2010
6	Automattic	Automattic	279	7	2011
7	Amazon Web Services	aws	49	10	2012
8	Microsoft Azure	Azure	148	7	2014
9	Bitly	bitly	26	8	2010
10	Cesanta Software	cesanta	24	5	2013
11	Chef Software, Inc.	chef	284	4	2008
12	Cloudera	cloudera	133	6	2009
13	Collective Idea	collectiveidea	155	4	2008
14	(Cucumber)	(cucumber)	(52)	(5)	2010
15	Docker	docker	64	15	2013
16	Douban Inc.	douban	38	7	2011
17	Dropbox	dropbox	104	6	2011
18	elastic	elastic	107	15	2014
19	(enormego)	(enormego)	(26)	(6)	2009
20	Etsy, Inc.	etsy	56	10	2010
21	Facebook	facebook	149	56	2009
22	Flipboard	Flipboard	19	4	2010
23	(Sentry)	(getsentry)	(79)	(4)	2012
24	GitHub	github	127	16	2008
25	(Glider Labs)	(gliderlabs)	(19)	(4)	2014
26	Google	google	681	85	2012
27	Google Samples	googlesamples	184	10	2014
28	HashiCorp	hashicorp	79	15	2011
29	Heroku	heroku	488	5	2008
30	id Software	id-Software	18	7	2012
31	Instagram	Instagram	25	4	2011
32	INTRIDEA Inc.	intridea	107	4	2008
33	KNP Labs	KnpLabs	92	8	2010
34	LinkedIn	linkedin	87	10	2010
35	Mapbox	mapbox	558	4	2011
36	Microsoft	Microsoft	405	33	2013
37	mongodb	mongodb	54	16	2009
38	Mutual Mobile	mutualmobile	60	4	2009
39	Netflix, Inc.	Netflix	99	17	2011
40	OpenStack	openstack	665	4	2010
41	ownCloud	owncloud	95	4	2012
42	Parse	ParsePlatform	56	6	2011
43	PayPal	paypal	140	5	2010

	Name	Name on GitHub	Public Repositories	Top Repositories	On GitHub since
44	Phacility	phacility	6	4	2013
45	(Plataformatec)	(plataformatec)	(23)	(5)	2009
46	Qihoo 360	Qihoo360	17	4	2013
47	Cloud Programmability Group	Reactive-Extensions	38	4	2011
48	(ServiceStack)	(ServiceStack)	(32)	(5)	2011
49	Shopify	Shopify	265	9	2008
50	Sourcegraph	sourcegraph	150	4	2013
51	Spotify	spotify	131	6	2010
52	Square	square	157	30	2009
53	Stripe	stripe	66	7	2011
54	thoughtbot, inc.	thoughtbot	221	23	2008
55	Tumblr	tumblr	34	4	2010
56	Twilio	twilio	47	6	2009
57	Twitter, Inc.	twitter	135	18	2009
58	Valve Software	ValveSoftware	17	5	2012
59	Venmo	venmo	73	4	2010
60	Xamarin	xamarin	78	10	2011
61	Yahoo Inc.	yahoo	330	6	2008
62	Yalantis	Yalantis	36	13	2011
63	Yelp.com	Yelp	149	8	2009
64	yhat	yhat	102	4	2012

Table 22: Finally selected commercial firms for observation. Bracketed firms are sorted out since they are not using an united webdomain.

Data source:
https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/csv/organizations.csv
https://git.zeitpulse.com/philipp/masterthesis-data/raw/master/csv/commercial_classification/commercial_classification_of_organizations.csv

	Firm	Rating	Ratings count	Level	Culture and Values	Career Opportunities	Work-Life-Balance	OS Repos.	Top Repos. count
1	elastic	4.90	62	Very Satisfied	4.80	4.80	4.50	107	15
2	Facebook	4.50	1297	Very Satisfied	4.50	4.30	3.70	149	56
3	Airbnb	4.50	299	Very Satisfied	4.70	4.30	3.90	86	10
4	Google	4.40	4665	Very Satisfied	4.40	4.00	4.00	681	85
5	Square	4.40	155	Very Satisfied	4.30	4.00	4.10	157	30
6	Google Samples	4.40	4665	Very Satisfied	4.40	4.00	4.00	184	10
7	LinkedIn	4.40	1340	Very Satisfied	4.50	4.10	4.10	87	10
8	Etsy, Inc.	4.30	38	Very Satisfied	4.30	3.50	4.20	56	10
9	Twitter, Inc.	4.00	400	Satisfied	4.10	3.70	4.00	135	18
10	mongodb	4.00	135	Satisfied	4.00	4.00	3.80	54	16
11	Microsoft	3.90	12492	Satisfied	3.70	3.60	3.60	405	33
12	Alibaba	3.90	42	Satisfied	3.90	4.00	3.40	87	19
13	Netflix, Inc.	3.70	544	Satisfied	3.90	3.40	3.40	99	17
14	Amazon Web Services	3.40	7572	OK	3.30	3.40	2.70	49	10

Table 23: Firm ratings by employees (*Source: Glassdoor API, 25.01.2016*)

[81] by Amazon.com

[82] by Amazon.com

[83] by Salesforce

[84] by Amazon.com

[85] by Salesforce

B.3 Regression Tables for H 1.1

	Dependent variable:			
	ext. commits	int. commits	ext. commits	int. commits
	(1.1.5)	(1.1.6)	(1.1.7)	(1.1.8)
int. commits	5.566***		0.453***	
	(0.448)		(0.075)	
ext. commits		0.009***		0.156***
		(0.001)		(0.026)
Constant	$-$145.306	100.309***	381.728**	710.625***
	(166.753)	(6.572)	(167.964)	(93.460)
Observations	2,809	2,809	478	478
R^2	0.052	0.052	0.071	0.071
Adjusted R^2	0.052	0.052	0.069	0.069
Residual Std. Error	8,483.016 (df = 2807)	347.859 (df = 2807)	3,408.818 (df = 476)	1,997.843 (df = 476)
F Statistic	154.275*** (df = 1; 2807)	154.275*** (df = 1; 2807)	36.143*** (df = 1; 476)	36.143*** (df = 1; 476)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 24: Impact of internal commits on external commits (and v.v.) on ”Residual Projects” (Model 1.1.5 - 1.1.6) and ”Top Projects” older than 1 year (Model 1.1.7 - 1.1.8)

	Dependent variable:			
	ext. commits	int. commits	ext. commits	int. commits
	(1.1.9)	(1.1.10)	(1.1.11)	(1.1.12)
int. commits	5.751***		0.936***	
	(0.528)		(0.040)	
ext. commits		0.010***		0.864***
		(0.001)		(0.037)
Constant	$-$238.115	121.186***	$-$318.675	519.123**
	(232.686)	(9.417)	(249.478)	(236.770)
Observations	1,900	1,900	132	132
R^2	0.059	0.059	0.809	0.809
Adjusted R^2	0.058	0.058	0.807	0.807
Residual Std. Error	9,718.195 (df = 1898)	409.982 (df = 1898)	2,805.891 (df = 130)	2,694.891 (df = 130)
F Statistic	118.705*** (df = 1; 1898)	118.705*** (df = 1; 1898)	550.484*** (df = 1; 130)	550.484*** (df = 1; 130)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 25: Impact of internal commits on external commits (and v.v.) on ”Residual Projects” older than 1 year (Model 1.1.9 - 1.1.10) and ”Top Projects” younger than 1 year (Model 1.1.11 - 1.1.12)

	Dependent variable:	
	ext. commits	int. commits
	(1.1.13)	(1.1.14)
int. commits	2.323*	
	(1.225)	
ext. commits		0.002*
		(0.001)
Constant	190.823	58.289***
	(180.326)	(4.483)
Observations	909	909
R^2	0.004	0.004
Adjusted R^2	0.003	0.003
Residual Std. Error (df = 907)	4,983.821	134.877
F Statistic (df = 1; 907)	3.598*	3.598*
Note:	*p<0.1; **p<0.05; ***p<0.01	

Table 26: Impact of internal commits on external commits (and v.v.) on ”Residual Projects” younger than 1 year (Model 1.1.13 - 1.1.14)

B.4 Regression Tables for H 1.2.1

	Dependent variable:			
	issues by ext. users	issues by firm empl. users	issues by ext. users	issues by firm empl. users
	(5)	(6)	(7)	(8)
issues by firm empl. users	5.211***		34.605***	
	(0.448)		(2.815)	
issues by ext. users		0.010***		0.008***
		(0.001)		(0.001)
Constant	40.005***	0.357***	242.496***	2.892***
	(2.717)	(0.125)	(47.815)	(0.724)
Observations	2,412	2,412	414	414
R^2	0.053	0.053	0.268	0.268
Adjusted R^2	0.053	0.053	0.267	0.267
Residual Std. Error	132.237 (df = 2410)	5.851 (df = 2410)	898.292 (df = 412)	13.449 (df = 412)
F Statistic	135.324*** (df = 1; 2410)	135.324*** (df = 1; 2410)	151.164*** (df = 1; 412)	151.164*** (df = 1; 412)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 27: Impact of issue participation by firm employed developers on external users (and v.v.) in ”Residual Projects” (Model 1.2.5 - 1.2.6) and ”Top Projects” older than 1 year (Model 1.2.7 - 1.2.8)

	Dependent variable:			
	issues by ext. users	issues by firm empl. users	issues by ext. users	issues by firm empl. users
	(9)	(10)	(11)	(12)
issues by firm empl. users	5.051***		15.993***	
	(0.526)		(3.744)	
issues by ext. users		0.010***		0.009***
		(0.001)		(0.002)
Constant	47.408***	0.543***	179.710***	-0.124
	(3.813)	(0.180)	(33.079)	(0.892)
Observations	1,679	1,679	109	109
R^2	0.052	0.052	0.146	0.146
Adjusted R^2	0.052	0.052	0.138	0.138
Residual Std. Error	154.470 (df = 1677)	6.985 (df = 1677)	338.349 (df = 107)	8.075 (df = 107)
F Statistic	92.310*** (df = 1; 1677)	92.310*** (df = 1; 1677)	18.248*** (df = 1; 107)	18.248*** (df = 1; 107)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 28: Impact of issue participation by firm employed developers on external users (and v.v.) in ”Residual Projects” older than 1 year (Model 1.2.9 - 1.2.10) and ”Top Projects” younger than 1 year (Model 1.2.11 - 1.2.12)

	Dependent variable:	
	issues by ext. users	issues by firm empl. users
	(13)	(14)
issues by firm empl. users	11.839***	
	(2.545)	
issues by ext. users		0.002***
		(0.001)
Constant	22.359***	0.105***
	(1.890)	(0.029)
Observations	733	733
R^2	0.029	0.029
Adjusted R^2	0.027	0.027
Residual Std. Error (df = 731)	49.898	0.715
F Statistic (df = 1; 731)	21.636***	21.636***
Note:		*p<0.1; **p<0.05; ***p<0.01

Table 29: Impact of issue participation of firm employed users on external users (and v.v.) in ”Residual Projects” younger than 1 year (Model 1.2.13 - 1.2.14)

B.5 Regression Tables for H 1.2.2

	Dependent variable:			
	Number of comments	Comments by ext. developers		Comments by firm employed developers
	(1.3.5)	(1.3.6)	(1.3.7)	(1.3.8)
Content share by firm employed developers	−35,594.540	−35,082.530		
	(400,394.800)	(385,606.700)		
Comments by firm employed developers			14.338^{***}	
			(0.592)	
Comments by ext. developers				0.038^{***}
				(0.002)
Constant	$2,080.726^{***}$	$2,037.550^{***}$	$1,416.162^{***}$	-33.710^{**}
	(477.470)	(459.836)	(311.632)	(16.254)
Observations	498	498	498	498
R^2	0.00002	0.00002	0.542	0.542
Adjusted R^2	−0.002	−0.002	0.541	0.541
Residual Std. Error (df = 496)	10,628.340	10,235.790	6,930.931	355.725
F Statistic (df = 1; 496)	0.008	0.008	585.804^{***}	585.804^{***}
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 30: Impact of participation by content share in issues' comments (Model 1.3.5 - 1.3.6) and numbers of comment by firm employed users on external users (and v.v.) (Model 1.3.7 - 1.3.8) in "Top Projects"

	Dependent variable:			
	Number of comments	Comments by ext. developers		Comments by firm employed developers
	(1.3.9)	(1.3.10)	(1.3.11)	(1.3.12)
Content share by firm employed developers	4,611.045	4,577.814		
	(13,202.160)	(12,945.400)		
Comments by firm employed developers			4.109***	
			(0.318)	
Comments by ext. developers				0.018***
				(0.001)
Constant	119.063***	116.157***	104.293***	0.832
	(9.816)	(9.625)	(9.309)	(0.631)
Observations	2,111	2,111	2,111	2,111
R^2	0.0001	0.0001	0.073	0.073
Adjusted R^2	−0.0004	−0.0004	0.073	0.073
Residual Std. Error (df = 2109)	450.873	442.104	425.589	28.056
F Statistic (df = 1; 2109)	0.122	0.125	166.993***	166.993***
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 31: Impact of participation by content share in issues' comments (Model 1.3.9 - 1.3.10) and numbers of comment by firm employed users on external users (and v.v.) (Model 1.3.11 - 1.3.12) in "Residual Projects"

B.6 Regression Tables and Data Summary for H 2.1

	Dependent variable:					
	Top Project					
	(2.1.11)	(2.1.12)	(2.1.13)	(2.1.14)	(2.1.15)	(2.1.16)
Age	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}	0.0002
	(0.0001)	(0.0001)	(0.0001)	(0.0001)	(0.0002)	(0.0002)
Forks_1	2.125					
	(1.419)					
Forks_2		3.020^{***}				
		(0.711)				
Forks_3			2.883^{***}			
			(0.532)			
Forks_4				0.884^{**}		
				(0.397)		
Forks_5					-2.017^{***}	
					(0.315)	
forks.today						0.007^{***}
						(0.001)
Constant	-2.298^{***}	-2.649^{***}	-3.047^{***}	-2.703^{***}	-2.093^{***}	-2.566^{***}
	(0.194)	(0.218)	(0.249)	(0.281)	(0.192)	(0.238)
Observations	1,409	1,409	1,409	1,409	1,409	1,409
Log Likelihood	−749.642	−741.173	−735.814	−748.297	−728.325	−501.374
Akaike Inf. Crit.	1,505.285	1,488.346	1,477.627	1,502.593	1,462.650	1,008.749
Note:						*p<0.1; **p<0.05; ***p<0.01

Table 32: The more forks a project gains in the beginning (i.e. first 3 - 6 months) the more likely it is a top project in the long-run (Model 2.1.11 - 2.1.16)

	Dependent variable:			
	Top Project			
	(2.1.17)	(2.1.18)	(2.1.19)	(2.1.20)
Age	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}
	(0.0001)	(0.0001)	(0.0002)	(0.0001)
Forks_{1-2}	5.695^{***}			
	(1.271)			
Forks_{2-3}		5.199^{***}		
		(0.804)		
Forks_{3-4}			3.445^{***}	
			(0.684)	
Forks_{4-5}				-5.106^{***}
				(0.773)
Constant	-2.681^{***}	-3.310^{***}	-3.588^{***}	-0.788^{***}
	(0.219)	(0.263)	(0.336)	(0.283)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	−740.234	−729.169	−737.423	−728.274
Akaike Inf. Crit.	1,486.467	1,464.338	1,480.846	1,462.547
Note:	*p<0.1; **p<0.05; ***p<0.01			

Table 33: If a projects gets forked in the beginning more often, the more likely it is a ”Top Project” (Model 2.1.17 - 2.1.20)

	Dependent variable:			
	Forks_2	Forks_3	Forks_4	Forks_5
	(2.1.21)	(2.1.22)	(2.1.23)	(2.1.24)
Age	-0.00003^{***}	-0.0001^{***}	-0.0002^{***}	0.0003^{***}
	(0.00001)	(0.00001)	(0.00001)	(0.00001)
Ratio_1	0.031^{***}			
	(0.007)			
Ratio_2		0.060^{***}		
		(0.009)		
Ratio_3			0.058^{***}	
			(0.012)	
Ratio_4				-0.153^{***}
				(0.016)
Constant	0.113^{***}	0.238^{***}	0.486^{***}	0.168^{***}
	(0.007)	(0.010)	(0.014)	(0.019)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	1,407.889	1,003.989	574.908	127.614
Akaike Inf. Crit.	−2,809.778	−2,001.979	−1,143.817	−249.229
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 34: As the delay shows, number of ”Forks” are rising relatively over time (see table 41 for details). This maybe caused by network effects (the attention of a project in the beginning affects the later attention beneficial). Thus, further analyses would be necessary to proof the pure effect of ”Ratio” on ”Forks” over time. (Model 2.1.21 - 2.1.31).

	Dependent variable:										
	Subscribers$_1$	Subscribers$_2$	Subscribers$_3$	Subscribers$_4$	Subscribers$_2$	Subscribers$_3$	Subscribers$_4$	Subscribers$_5$	Subscribers$_3$	Subscribers$_4$	Subscribers$_5$
	(2.1.32)	(2.1.33)	(2.1.34)	(2.1.35)	(2.1.36)	(2.1.37)	(2.1.38)	(2.1.39)	(2.1.40)	(2.1.41)	(2.1.42)
Age	-0.00000	-0.00001^{***}	-0.00005^{***}	-0.0001^{***}	-0.00001^{***}	-0.00005^{***}	-0.0001^{***}	0.0002^{***}	-0.00005^{***}	-0.0001^{***}	0.0002^{***}
	(0.00000)	(0.00000)	(0.00000)	(0.00001)	(0.00000)	(0.00000)	(0.00001)	(0.00001)	(0.00000)	(0.00001)	(0.00001)
Ratio$_1$	0.004^{***}				0.021^{***}				0.055^{***}		
	(0.001)				(0.004)				(0.007)		
Ratio$_2$		0.023^{***}				0.054^{***}				0.079^{***}	
		(0.003)				(0.006)				(0.011)	
Ratio$_3$			0.067^{***}				0.097^{***}				-0.190^{***}
			(0.006)				(0.011)				(0.016)
Ratio$_4$				0.089^{***}				-0.148^{***}			
				(0.011)				(0.016)			
Constant	0.006^{***}	0.069^{***}	0.200^{***}	0.546^{***}	0.071^{***}	0.207^{***}	0.552^{***}	0.176^{***}	0.212^{***}	0.562^{***}	0.175^{***}
	(0.001)	(0.004)	(0.007)	(0.013)	(0.004)	(0.007)	(0.012)	(0.019)	(0.007)	(0.012)	(0.019)
Observations	1,409	1,409	1,409	1,409	1,409	1,409	1,409	1,409	1,409	1,409	1,409
Log Likelihood	4,296.766	2,453.283	1,565.419	701.254	2,446.640	1,539.586	705.850	108.943	1,535.701	690.719	133.812
Akaike Inf. Crit.	−8,587.532	−4,900.567	−3,124.839	−1,396.507	−4,887.280	−3,073.172	−1,405.699	−211.885	−3,065.401	−1,375.438	−261.624

Note: *p<0.1; **p<0.05; ***p<0.01

Table 35: Influence of ”Ratio” on number of ”Subscribers” (Model 2.1.32 - 2.1.42). See table 34 for possible explanation of effects (”Subscribers” behaves similar to ”Forks”)

	Dependent variable:			
	Forks_{today}			
	(2.1.43)	(2.1.44)	(2.1.45)	(2.1.46)
Age	0.180^{***}	0.186^{***}	0.186^{***}	0.185^{***}
	(0.034)	(0.034)	(0.034)	(0.034)
Ratio_1	254.674^{***}			
	(46.734)			
Ratio_2		194.085^{***}		
		(43.119)		
Ratio_3			217.558^{***}	
			(41.383)	
Ratio_4				135.234^{***}
				(40.749)
Constant	-95.681^{**}	-102.676^{**}	-121.572^{**}	-107.743^{**}
	(46.130)	(46.954)	(47.354)	(48.734)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	−10,916.200	−10,920.850	−10,917.220	−10,925.430
Akaike Inf. Crit.	21,838.410	21,847.700	21,840.430	21,856.870
Note:			*p<0.1; **p<0.05; ***p<0.01	

Table 36: The higher the code contribution share of firm employed developers in the beginning, the higher is the final number of forks (except between month 6 - 12) (Model 2.1.43 - 2.1.46)

	Dependent variable:			
	Subscribers$_{today}$			
	(2.1.47)	(2.1.48)	(2.1.49)	(2.1.50)
Age	0.405**	0.430***	0.424***	0.434***
	(0.163)	(0.163)	(0.163)	(0.163)
Ratio$_1$	1,053.216***			
	(226.797)			
Ratio$_2$		877.369***		
		(208.852)		
Ratio$_3$			753.878***	
			(201.218)	
Ratio$_4$				728.765***
				(197.006)
Constant	77.437	28.895	17.185	−42.648
	(223.868)	(227.429)	(230.250)	(235.611)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	−13,141.840	−13,143.780	−13,145.570	−13,145.740
Akaike Inf. Crit.	26,289.680	26,293.550	26,297.130	26,297.480
Note:	*p<0.1; **p<0.05; ***p<0.01			

Table 37: The higher the code contribution share of firm employed developers in the beginning, the higher is the final number of forks (except between month 3 - 6) (Model 2.1.47 - 2.1.50)

	Dependent variable:			
	Forks$_{today}$			
	(2.1.51)	(2.1.52)	(2.1.53)	(2.1.54)
Age	0.185***	0.188***	0.185***	0.180***
	(0.034)	(0.034)	(0.034)	(0.034)
Ratio$_{1\_2}$	287.570***			
	(50.896)			
Ratio$_{2\_3}$		281.414***		
		(49.170)		
Ratio$_{3\_4}$			135.234***	
			(40.749)	
Ratio$_{4\_5}$				135.903***
				(48.874)
Constant	$-$114.287**	$-$133.835***	$-$107.743**	$-$108.402**
	(46.716)	(47.585)	(48.734)	(50.104)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	$-$10,915.110	$-$10,914.710	$-$10,925.430	$-$10,927.070
Akaike Inf. Crit.	21,836.220	21,835.410	21,856.870	21,860.130
Note:			*p<0.1; **p<0.05; ***p<0.01	

Table 38: Code contribution share of firm employed developers in the first year has the biggest positive impact on the final number of forks (Model 2.1.51 - 2.1.54)

	Dependent variable:			
	Subscribers$_{today}$			
	(2.1.55)	(2.1.56)	(2.1.57)	(2.1.58)
Age	0.424***	0.436***	0.434***	0.402**
	(0.163)	(0.163)	(0.163)	(0.163)
Ratio$_{1\_2}$	1,241.737***			
	(246.863)			
Ratio$_{2\_3}$		1,109.494***		
		(238.874)		
Ratio$_{3\_4}$			728.765***	
			(197.006)	
Ratio$_{4\_5}$				816.083***
				(236.158)
Constant	−10.923	−64.506	−42.648	−81.107
	(226.587)	(231.175)	(235.611)	(242.100)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	−13,140.000	−13,141.840	−13,145.740	−13,146.610
Akaike Inf. Crit.	26,286.000	26,289.670	26,297.480	26,299.210
Note:			*p<0.1; **p<0.05; ***p<0.01	

Table 39: Code contribution share of firm employed developers in the first year has the biggest positive impact on the final number of subscribers (Model 2.1.55 - 2.1.58)

	Dependent variable:			
	Top Project			
	(2.1.59)	(2.1.60)	(2.1.61)	(2.1.62)
Age	0.001^{***}	0.001^{***}	0.001^{***}	0.001^{***}
	(0.0001)	(0.0001)	(0.0001)	(0.0001)
Ratio_{1-2}	2.330^{***}			
	(0.212)			
Ratio_{2-3}		2.408^{***}		
		(0.213)		
Ratio_{3-4}			1.758^{***}	
			(0.178)	
Ratio_{4-5}				2.488^{***}
				(0.230)
Constant	-2.984^{***}	-3.201^{***}	-3.160^{***}	-3.579^{***}
	(0.219)	(0.228)	(0.227)	(0.249)
Observations	1,409	1,409	1,409	1,409
Log Likelihood	−687.044	−682.113	−699.158	−685.443
Akaike Inf. Crit.	1,380.088	1,370.227	1,404.317	1,376.885
Note:	*p<0.1; **p<0.05; ***p<0.01			

Table 40: The higher the code contribution share of firm employed developers in the beginning the more likely it is a ”Top Project” in the long-run (Model 2.1.59 - 2.1.62)

Statistic	N	Mean	St. Dev.	Min	Max
Forks$_{today}$	1,409	178.048	571.655	1	11,670
Subscribers$_{today}$	1,409	773.444	2,744.954	1	56,785
Number of Code Contributions$_{total}$	1,409	1,140,249.000	0.000	1,140,249	1,140,249
Number of Code Contributions$_1$	1,409	13.781	61.867	0	1,040
Number of Code Contributions$_2$	1,409	39.065	156.261	0	2,320
Number of Code Contributions$_3$	1,409	85.478	326.570	0	4,476
Number of Code Contributions$_4$	1,409	171.919	575.452	0	7,054
Number of Code Contributions$_5$	1,409	499.018	2,836.235	0	65,040
Ratio$_1$	1,409	0.169	0.320	0.000	1.000
Ratio$_2$	1,409	0.221	0.348	0.000	1.000
Ratio$_3$	1,409	0.283	0.361	0.000	1.000
Ratio$_4$	1,409	0.359	0.370	0.000	1.000
Ratio$_5$	1,409	0.474	0.359	0.000	1.000
Subscribers$_1$	1,409	0.007	0.012	0.000	0.200
Subscribers$_2$	1,409	0.060	0.043	0.000	0.333
Subscribers$_3$	1,409	0.161	0.086	0.000	0.485
Subscribers$_4$	1,409	0.392	0.165	0.000	1.000
Subscribers$_5$	1,409	0.381	0.248	0.000	1.000
Forks$_1$	1,409	0.012	0.040	0.000	1.000
Forks$_2$	1,409	0.077	0.091	0.000	1.000
Forks$_3$	1,409	0.162	0.125	0.000	1.000
Forks$_4$	1,409	0.305	0.177	0.000	1.000
Forks$_5$	1,409	0.445	0.257	0.000	1.000
Ratio$_{1-2}$	1,409	0.195	0.293	0.000	1.000
Ratio$_{2-3}$	1,409	0.252	0.304	0.000	1.000
Ratio$_{3-4}$	1,409	0.359	0.370	0.000	1.000
Ratio$_{4-5}$	1,409	0.416	0.308	0.000	1.000
Subscribers$_{1-2}$	1,409	0.033	0.024	0.000	0.200
Subscribers$_{2-3}$	1,409	0.110	0.060	0.000	0.333
Subscribers$_{3-4}$	1,409	0.277	0.111	0.000	0.500
Subscribers$_{4-5}$	1,409	0.386	0.062	0.167	0.500
Forks$_{1-2}$	1,409	0.044	0.050	0.000	0.500
Forks$_{2-3}$	1,409	0.119	0.085	0.000	0.500
Forks$_{3-4}$	1,409	0.233	0.113	0.000	0.500
Forks$_{4-5}$	1,409	0.375	0.088	0.000	0.500

Table 41: Trend of social metrics and contribution ratio over time. All observed projects are at least 4 years old. Normalized values (between 0 - 1) for ”Forks”, ”Subscribers” and ”Ratio” are increasing over time in most cases.

B.7 Regression Tables for H 3

	Dependent variable:											
	Stars				Subscribers				Forks			
	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)	(3.9)	(3.10)	(3.11)	(3.12)
Ratio	295.609***	278.398***	129.899	152.407*	7.184	7.068	18.616***	17.985***	51.697***	39.361**	26.293	29.970
	(75.069)	(79.011)	(83.121)	(83.466)	(6.010)	(6.284)	(6.284)	(6.322)	(18.318)	(19.329)	(20.865)	(21.002)
Age	0.367***	0.437***	0.443***	0.494***	0.031***	0.036***	0.037***	0.039***	0.129***	0.143***	0.151***	0.158***
	(0.047)	(0.049)	(0.050)	(0.051)	(0.004)	(0.004)	(0.004)	(0.004)	(0.011)	(0.012)	(0.013)	(0.013)
lang_C#		−214.631		169.676		−4.257		10.249		19.293		59.294
		(178.693)		(196.570)		(14.212)		(14.889)		(43.714)		(49.463)
lang_C++		116.238		332.343**		1.946		25.198**		33.039		83.661**
		(167.244)		(166.608)		(13.301)		(12.620)		(40.913)		(41.924)
lang_Go		239.094		334.448*		−4.575		12.997		41.503		59.721
		(172.819)		(183.499)		(13.745)		(13.899)		(42.277)		(46.174)
lang_Java		157.940		282.315*		25.440**		25.214**		77.153**		109.166***
		(155.887)		(159.385)		(12.398)		(12.073)		(38.135)		(40.106)
lang_JavaScript		208.857		592.722***		14.696		22.251*		33.642		126.760***
		(152.583)		(157.994)		(12.135)		(11.967)		(37.326)		(39.756)
lang_Objective-C		362.103*		363.975*		−0.915		7.949		44.242		72.779
		(187.742)		(191.544)		(14.932)		(14.509)		(45.927)		(48.198)
lang_PHP		−216.619		108.148		1.185		2.580		−28.608		61.594
		(177.471)		(200.353)		(14.115)		(15.176)		(43.415)		(50.415)
lang_Python		−145.320		108.349		−31.810**		−2.574		−12.433		40.406
		(157.632)		(163.375)		(12.537)		(12.375)		(38.562)		(41.110)
lang_Ruby		−230.824		−30.721		−32.958***		−13.470		−50.827		2.041
		(155.093)		(167.303)		(12.335)		(12.673)		(37.940)		(42.098)
firm_airbnb			1,783.590***	1,872.003***			157.228***	170.044***			249.295***	261.275***
			(306.332)	(317.847)			(23.159)	(24.076)			(76.896)	(79.980)

	Dependent variable:											
	Stars				Subscribers				Forks			
	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)	(3.9)	(3.10)	(3.11)	(3.12)
firm_alibaba			721.450***	731.614***			142.658***	142.330***			348.882***	335.851***
			(257.418)	(265.678)			(19.461)	(20.124)			(64.618)	(66.852)
firm_apple			3,988.731***	4,078.281***			308.373***	312.584***			567.897***	586.710***
			(524.956)	(522.366)			(39.687)	(39.567)			(131.776)	(131.443)
firm_applidium			444.278	427.531			20.273	28.301			37.993	41.116
			(655.790)	(660.620)			(49.578)	(50.039)			(164.618)	(166.231)
firm_Automattic			347.271*	285.364			153.892***	159.146***			33.362	13.352
			(183.921)	(207.865)			(13.904)	(15.745)			(46.168)	(52.305)
firm_aws			450.729*	492.689*			47.916**	57.181***			122.577*	125.413*
			(259.128)	(270.254)			(19.590)	(20.471)			(65.047)	(68.004)
firm_Azure			197.388	223.868			36.238**	42.884***			72.173	73.259
			(192.978)	(210.404)			(14.589)	(15.937)			(48.442)	(52.944)
firm_bitly			91.677	114.545			17.437	27.925			−60.302	−40.775
			(380.401)	(388.841)			(28.758)	(29.453)			(95.489)	(97.844)
firm_cesanta			378.819	532.016			26.698	40.404			95.627	144.460
			(391.038)	(402.779)			(29.562)	(30.509)			(98.160)	(101.351)
firm_chef			57.765	342.192			41.216***	71.395***			10.695	75.783
			(180.801)	(212.299)			(13.669)	(16.081)			(45.385)	(53.421)
firm_cloudera			158.901	171.780			23.838	21.199			36.376	11.930
			(249.194)	(260.622)			(18.839)	(19.741)			(62.553)	(65.580)
firm_collectiveidea			−197.327	39.931			−45.061**	−16.310			−113.033	−54.816
			(283.407)	(303.640)			(21.426)	(23.000)			(71.142)	(76.405)
firm_docker			1,833.834***	1,803.627***			154.281***	161.640***			418.454***	429.360***
			(272.022)	(293.341)			(20.565)	(22.219)			(68.284)	(73.813)
firm_douban			208.868	305.613			15.831	31.079			23.678	45.856
			(306.898)	(314.065)			(23.201)	(23.789)			(77.039)	(79.028)

	Dependent variable:											
	Stars				Subscribers				Forks			
	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)	(3.9)	(3.10)	(3.11)	(3.12)
firm_dropbox			469.635*	600.255**			35.674*	50.410**			62.852	84.987
			(260.330)	(269.009)			(19.681)	(20.376)			(65.349)	(67.690)
firm_elastic			611.492***	652.659***			72.232***	77.962***			171.865***	168.103***
			(210.873)	(224.926)			(15.942)	(17.037)			(52.934)	(56.598)
firm_etsy			752.616***	809.325***			42.131**	52.976**			75.912	80.411
			(277.259)	(287.605)			(20.961)	(21.785)			(69.598)	(72.370)
firm_facebook			2,394.111***	2,384.316***			171.279***	176.552***			384.147***	380.714***
			(184.613)	(191.626)			(13.957)	(14.515)			(46.342)	(48.219)
firm_Flipboard			3,845.847***	3,760.156***			182.437***	183.684***			388.654**	363.904**
			(656.115)	(659.318)			(49.602)	(49.941)			(164.700)	(165.904)
firm_github			898.635***	1,037.884***			34.714*	53.426***			187.585***	223.290***
			(256.875)	(269.861)			(19.420)	(20.441)			(64.481)	(67.905)
firm_google			720.257***	727.488***			53.652***	58.067***			133.897***	130.516***
			(146.826)	(155.824)			(11.100)	(11.803)			(36.857)	(39.210)
firm_googlesamples			501.872**	496.091**			43.458**	39.813**			162.108***	132.682**
			(227.162)	(242.156)			(17.173)	(18.342)			(57.023)	(60.934)
firm_hashicorp			814.807***	791.444***			44.781**	53.007**			109.553*	123.786*
			(254.954)	(280.208)			(19.275)	(21.225)			(63.999)	(70.508)
firm_heroku			89.603	177.261			15.988	31.161**			72.618*	92.963**
			(165.134)	(186.053)			(12.484)	(14.093)			(41.452)	(46.816)
firm_id-Software			1,886.755	1,805.205			336.544***	328.951***			594.253	578.166
			(1,447.476)	(1,439.507)			(109.429)	(109.037)			(363.350)	(362.222)
firm_Instagram			752.507	880.403			39.559	56.610			176.467	201.123
			(655.800)	(657.806)			(49.578)	(49.826)			(164.621)	(165.523)
firm_intridea			181.327	321.635			−24.038	−1.633			−36.053	2.153
			(308.743)	(323.098)			(23.341)	(24.473)			(77.502)	(81.301)

	Dependent variable:											
	Stars				Subscribers				Forks			
	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)	(3.9)	(3.10)	(3.11)	(3.12)
firm_KnpLabs			9.014	137.359			−0.787	12.789			−16.603	−12.197
			(223.086)	(267.556)			(16.865)	(20.266)			(56.000)	(67.325)
firm_linkedin			325.648	280.647			24.240	26.410			33.529	14.243
			(241.938)	(252.207)			(18.291)	(19.104)			(60.732)	(63.463)
firm_mapbox			185.569	−48.125			80.295***	78.695***			26.587	−14.349
			(151.975)	(170.937)			(11.489)	(12.948)			(38.149)	(43.013)
firm_Microsoft			401.146**	440.668**			47.493***	52.144***			96.519**	97.017**
			(162.492)	(181.612)			(12.284)	(13.756)			(40.789)	(45.699)
firm_mongodb			572.687**	605.178**			60.054***	69.642***			203.993***	211.318***
			(286.179)	(291.850)			(21.635)	(22.107)			(71.838)	(73.438)
firm_mutualmobile			794.332*	681.476			34.910	40.099			111.917	95.217
			(432.681)	(445.379)			(32.711)	(33.736)			(108.613)	(112.070)
firm_Netflix			509.045**	451.850**			232.128***	230.588***			70.328	40.937
			(200.325)	(215.458)			(15.145)	(16.320)			(50.286)	(54.216)
firm_openstack			162.883	354.423*			13.372	35.155**			64.127	100.846**
			(166.677)	(188.503)			(12.601)	(14.278)			(41.840)	(47.433)
firm_owncloud			245.363	293.141			94.300***	104.324***			84.892	82.323
			(220.658)	(242.050)			(16.682)	(18.334)			(55.390)	(60.907)
firm_ParsePlatform			431.926	353.046			56.053**	60.203***			118.252	103.283
			(289.122)	(298.425)			(21.858)	(22.605)			(72.576)	(75.093)
firm_paypal			88.295	56.179			9.617	15.543			16.517	4.902
			(196.640)	(210.941)			(14.866)	(15.978)			(49.361)	(53.079)
firm_phacility			1,354.084**	1,318.708**			155.553***	159.946***			174.756	158.418
			(658.868)	(662.543)			(49.810)	(50.185)			(165.391)	(166.715)
firm_Qihoo360			943.963*	1,053.359*			122.062***	126.461***			317.453**	339.558**
			(558.211)	(557.352)			(42.201)	(42.217)			(140.124)	(140.246)

	Dependent variable:											
	Stars				Subscribers				Forks			
	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)	(3.9)	(3.10)	(3.11)	(3.12)
firm_Reactive-Extensions			608.570*	411.081			26.069	24.826			48.994	13.200
			(362.725)	(369.390)			(27.422)	(27.980)			(91.052)	(92.949)
firm_Shopify			295.762	467.720**			105.086***	127.295***			14.280	58.401
			(203.489)	(224.093)			(15.384)	(16.974)			(51.081)	(56.388)
firm_sourcegraph			180.876	150.278			0.193	7.189			25.492	36.502
			(247.783)	(273.833)			(18.732)	(20.742)			(62.199)	(68.904)
firm_spotify			264.807	328.380			45.336***	51.815***			33.513	31.464
			(227.333)	(238.452)			(17.186)	(18.062)			(57.066)	(60.002)
firm_square			1,093.104***	1,109.454***			70.649***	76.974***			166.484***	163.711***
			(193.279)	(210.141)			(14.612)	(15.917)			(48.517)	(52.878)
firm_stripe			238.656	323.727			29.079	44.894**			23.239	45.943
			(284.560)	(295.828)			(21.513)	(22.408)			(71.431)	(74.439)
firm_thoughtbot			514.152***	684.286***			9.095	32.676**			29.911	73.026
			(195.882)	(219.319)			(14.809)	(16.613)			(49.171)	(55.187)
firm_tumblr			439.802	445.527			43.491	52.326*			43.340	43.519
			(360.833)	(369.461)			(27.279)	(27.985)			(90.577)	(92.967)
firm_twilio			121.696	53.474			23.703	29.964			21.738	6.643
			(351.376)	(358.265)			(26.564)	(27.137)			(88.204)	(90.150)
firm_twitter			779.237***	762.750***			166.364***	172.280***			69.571	64.321
			(230.516)	(240.739)			(17.427)	(18.235)			(57.865)	(60.577)
firm_ValveSoftware			961.506*	1,006.940*			135.571***	135.791***			205.071	217.620
			(558.226)	(555.752)			(42.202)	(42.096)			(140.128)	(139.844)
firm_venmo			436.297	462.792			34.925	47.095**			46.361	53.875
			(302.225)	(315.025)			(22.848)	(23.862)			(75.865)	(79.269)
firm_xamarin			143.419	239.294			127.241***	134.535***			184.614***	193.921***
			(265.400)	(298.732)			(20.064)	(22.628)			(66.622)	(75.170)

	Dependent variable:											
	Stars				Subscribers				Forks			
	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)	(3.9)	(3.10)	(3.11)	(3.12)
firm_yahoo			291.255*	88.268			15.581	13.410			45.036	4.177
			(172.914)	(190.246)			(13.072)	(14.410)			(43.405)	(47.871)
firm_Yalantis			1,222.287***	1,197.076***			77.274***	79.524***			266.267***	248.290***
			(352.406)	(365.651)			(26.642)	(27.697)			(88.462)	(92.009)
firm_Yelp			183.684	338.035			12.692	32.249*			34.810	65.535
			(225.708)	(239.584)			(17.063)	(18.148)			(56.658)	(60.286)
firm_yhat			710.520	767.158			29.902	44.637			84.530	97.330
			(471.268)	(476.262)			(35.628)	(36.075)			(118.299)	(119.841)
Constant	−0.236	−70.590	−434.208***	−764.217***	48.363***	47.700***	−20.544*	−39.926***	−23.486	−43.139	−113.655***	−192.851***
	(60.212)	(154.968)	(139.704)	(195.611)	(4.821)	(12.325)	(10.562)	(14.817)	(14.692)	(37.910)	(35.069)	(49.222)
Observations	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−29,905.750	−29,877.320	−29,692.650	−29,668.010	−21,273.000	−21,221.860	−20,863.760	−20,845.730	−25,083.110	−25,063.340	−24,966.870	−24,950.480
Akaike Inf. Crit.	59,817.500	59,778.650	59,505.300	59,474.010	42,552.000	42,467.720	41,847.520	41,829.460	50,172.220	50,150.680	50,053.740	50,038.950

Note: *p<0.1; **p<0.05; ***p<0.01

Table 42: Detailed Regression Table: Impact of ”Age” and ”Ratio” on ”Stars”, ”Subscribers” and ”Forks” through *Fitting Generalized Linear Models* with dummy variables for *Firms* and *Programming Languages*. Using (a) dummy variables for each *Firm* in model 3.3, 3.7 and 3.11 (b) dummy variables for each *Language* in model 3.2, 3.6, 3.10 (c) dummy variables for each *Firm and Language* in model 3.4, 3.8 and 3.12

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
ratio	2.147	2.006	7.861*	7.612*	−2.417	−2.647	1.391	1.402	2.025***	1.727***	1.906***	1.753***
	(3.717)	(3.930)	(4.235)	(4.275)	(1.911)	(2.005)	(2.051)	(2.067)	(0.155)	(0.166)	(0.191)	(0.197)
age	0.015***	0.017***	0.021***	0.022***	0.018***	0.019***	0.020***	0.021***	0.001***	0.001***	0.001***	0.001***
	(0.002)	(0.002)	(0.003)	(0.003)	(0.001)	(0.001)	(0.001)	(0.001)	(0.0001)	(0.0001)	(0.0001)	(0.0001)
lang_C#		10.769		15.255		−2.302		−3.798		−0.439		0.301
		(8.887)		(10.067)		(4.535)		(4.869)		(0.272)		(0.368)
lang_C++		14.470*		17.171**		2.457		6.712		−0.320		0.555*
		(8.318)		(8.533)		(4.244)		(4.127)		(0.247)		(0.307)
lang_Go		22.068**		14.360		6.397		2.801		0.151		0.391
		(8.595)		(9.398)		(4.386)		(4.545)		(0.252)		(0.321)
lang_Java		14.278*		17.304**		−5.369		−3.103		−0.678***		−0.717**
		(7.753)		(8.163)		(3.956)		(3.948)		(0.232)		(0.285)
lang_JavaScript		12.686*		17.803**		−2.354		3.703		−1.619***		−1.202***
		(7.588)		(8.092)		(3.872)		(3.913)		(0.254)		(0.302)
lang_Objective-C		4.762		5.631		−6.667		−4.925		−0.120		−0.463
		(9.337)		(9.810)		(4.765)		(4.744)		(0.280)		(0.348)
lang_PHP		16.926*		8.654		−3.341		−0.999		−0.847***		−0.408
		(8.826)		(10.261)		(4.504)		(4.962)		(0.289)		(0.391)

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
lang_Python		3.092		7.609		12.824***		2.975		−1.414***		−1.162***
		(7.840)		(8.367)		(4.000)		(4.047)		(0.259)		(0.318)
lang_Ruby		−0.052		5.067		−1.393		−2.457		−1.655***		−1.517***
		(7.713)		(8.568)		(3.936)		(4.144)		(0.248)		(0.322)
firm_airbnb			17.277	19.858			17.225**	23.634***			3.062***	4.697***
			(15.609)	(16.279)			(7.558)	(7.873)			(0.569)	(0.611)
firm_alibaba			22.277*	22.852*			2.369	7.964			3.094***	3.969***
			(13.116)	(13.607)			(6.351)	(6.580)			(0.514)	(0.554)
firm_apple			23.867	26.692			131.388***	132.286***			6.461***	6.896***
			(26.748)	(26.753)			(12.952)	(12.938)			(1.150)	(1.174)
firm_applidium			5.866	12.765			−3.683	4.679			3.300***	4.150***
			(33.415)	(33.834)			(16.180)	(16.362)			(0.992)	(1.037)
firm_Automattic			37.468***	38.024***			11.555**	15.210***			1.441***	2.388***
			(9.371)	(10.646)			(4.538)	(5.148)			(0.559)	(0.620)
firm_aws			18.783	20.934			15.376**	21.263***			2.162***	3.118***
			(13.203)	(13.841)			(6.393)	(6.694)			(0.539)	(0.580)
firm_Azure			26.153***	26.499**			18.567***	24.545***			1.169**	1.779***

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
			(9.833)	(10.776)			(4.761)	(5.211)			(0.553)	(0.597)
firm_bitly			−7.492	−3.982			−6.882	−4.807			2.773***	3.309***
			(19.383)	(19.914)			(9.386)	(9.631)			(0.646)	(0.684)
firm_cesanta			15.082	23.531			17.458*	21.512**			2.835***	3.213***
			(19.925)	(20.628)			(9.648)	(9.976)			(0.692)	(0.729)
firm_chef			22.877**	31.331***			16.439***	23.568***			−0.379	1.402**
			(9.212)	(10.873)			(4.461)	(5.258)			(0.641)	(0.697)
firm_cloudera			9.800	8.119			6.025	12.201*			1.217**	2.206***
			(12.697)	(13.348)			(6.148)	(6.455)			(0.592)	(0.625)
firm_collectiveidea			−10.260	−2.599			−9.977	−3.410			−0.080	1.598**
			(14.440)	(15.551)			(6.992)	(7.521)			(0.681)	(0.731)
firm_docker			100.533***	101.238***			61.459***	63.974***			3.918***	4.346***
			(13.860)	(15.023)			(6.712)	(7.266)			(0.531)	(0.585)
firm_douban			3.401	8.936			−0.517	2.190			2.503***	3.583***
			(15.637)	(16.085)			(7.572)	(7.779)			(0.615)	(0.661)
firm_dropbox			18.458	23.230*			11.241*	14.929**			2.229***	3.268***
			(13.265)	(13.777)			(6.423)	(6.663)			(0.601)	(0.642)

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
firm_elastic			56.926***	57.240***			26.492***	32.473***			2.499***	3.557***
			(10.745)	(11.520)			(5.203)	(5.571)			(0.492)	(0.540)
firm_etsy			8.063	10.800			7.771	13.172*			2.540***	3.837***
			(14.127)	(14.730)			(6.841)	(7.123)			(0.545)	(0.587)
firm_facebook			55.298***	57.411***			35.655***	39.469***			3.909***	4.961***
			(9.407)	(9.814)			(4.555)	(4.746)			(0.438)	(0.482)
firm_Flipboard			26.656	28.141			18.759	26.441			5.318***	6.601***
			(33.431)	(33.767)			(16.188)	(16.330)			(1.197)	(1.241)
firm_github			13.649	18.961			49.316***	55.222***			2.935***	4.204***
			(13.089)	(13.821)			(6.338)	(6.684)			(0.512)	(0.577)
firm_google			37.187***	38.032***			16.524***	19.939***			2.894***	3.743***
			(7.481)	(7.981)			(3.623)	(3.859)			(0.407)	(0.438)
firm_googlesamples			12.768	10.421			11.100**	18.256***			2.163***	3.243***
			(11.575)	(12.402)			(5.605)	(5.998)			(0.521)	(0.564)
firm_hashicorp			53.445***	54.075***			32.248***	35.202***			3.328***	3.662***
			(12.991)	(14.351)			(6.290)	(6.940)			(0.510)	(0.580)
firm_heroku			7.694	10.957			5.035	10.236**			0.098	1.432**
			(8.414)	(9.529)			(4.074)	(4.608)			(0.595)	(0.633)

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
firm_id-Software			-7.908	-11.441			-8.110	-10.127			15.225	15.063
			(73.753)	(73.724)			(35.713)	(35.654)			(324.744)	(324.744)
firm_Instagram			27.182	32.765			15.720	22.539			5.015***	6.458***
			(33.415)	(33.690)			(16.180)	(16.293)			(1.219)	(1.252)
firm_intridea			2.502	8.195			3.275	9.105			1.146*	2.856***
			(15.731)	(16.547)			(7.618)	(8.003)			(0.682)	(0.729)
firm_KnpLabs			5.080	9.419			7.771	13.198**			1.429***	2.141***
			(11.367)	(13.703)			(5.504)	(6.627)			(0.548)	(0.642)
firm_linkedin			8.258	7.944			3.720	8.295			2.126***	3.300***
			(12.328)	(12.917)			(5.969)	(6.247)			(0.541)	(0.579)
firm_mapbox			18.398**	15.918*			8.194**	9.834**			-0.084	1.051
			(7.744)	(8.755)			(3.750)	(4.234)			(0.635)	(0.663)
firm_Microsoft			23.334***	22.830**			17.604***	23.280***			2.498***	2.927***
			(8.280)	(9.301)			(4.009)	(4.498)			(0.435)	(0.483)
firm_mongodb			3.715	6.473			47.271***	50.914***			2.327***	3.374***
			(14.582)	(14.947)			(7.061)	(7.229)			(0.554)	(0.592)
firm_mutualmobile			15.365	19.738			6.313	14.501			2.640***	3.668***

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
			(22.046)	(22.810)			(10.675)	(11.031)			(0.744)	(0.806)
firm_Netflix			19.113*	17.219			11.074**	16.565***			2.130***	3.333***
			(10.207)	(11.035)			(4.943)	(5.337)			(0.471)	(0.520)
firm_openstack			6.710	13.118			63.274***	66.251***			0.421	1.962***
			(8.493)	(9.654)			(4.112)	(4.669)			(0.639)	(0.689)
firm_owncloud			82.412***	85.365***			23.018***	28.135***			1.799***	2.691***
			(11.243)	(12.397)			(5.444)	(5.995)			(0.655)	(0.710)
firm_ParsePlatform			19.117	20.497			11.729	17.265**			2.467***	3.561***
			(14.732)	(15.284)			(7.133)	(7.391)			(0.606)	(0.646)
firm_paypal			4.931	5.371			5.610	10.504**			0.746	1.847***
			(10.019)	(10.803)			(4.852)	(5.225)			(0.598)	(0.634)
firm_phacility			54.564	55.098			49.630***	52.187***			3.822***	4.409***
			(33.571)	(33.932)			(16.256)	(16.410)			(1.188)	(1.225)
firm_Qihoo360			27.084	31.405			8.399	11.151			4.385***	4.713***
			(28.443)	(28.545)			(13.773)	(13.805)			(0.889)	(0.896)
firm_Reactive-Extensions			12.295	9.249			16.013*	18.364**			3.016***	3.979***
			(18.482)	(18.918)			(8.949)	(9.149)			(0.712)	(0.791)

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
firm_Shopify			8.128	14.190			8.601*	14.477***			1.474***	2.713***
			(10.368)	(11.477)			(5.021)	(5.550)			(0.536)	(0.582)
firm_sourcegraph			9.240	9.569			9.172	11.913*			1.420**	1.657**
			(12.625)	(14.024)			(6.113)	(6.782)			(0.652)	(0.706)
firm_spotify			10.709	12.302			13.646**	19.293***			1.394**	2.593***
			(11.583)	(12.212)			(5.609)	(5.906)			(0.580)	(0.619)
firm_square			9.775	11.049			10.174**	16.483***			2.670***	3.844***
			(9.848)	(10.762)			(4.769)	(5.205)			(0.442)	(0.491)
firm_stripe			1.525	5.807			13.700*	18.910***			2.186***	3.363***
			(14.499)	(15.151)			(7.021)	(7.327)			(0.582)	(0.628)
firm_thoughtbot			4.963	11.521			14.751***	21.141***			1.889***	3.558***
			(9.981)	(11.232)			(4.833)	(5.432)			(0.470)	(0.534)
firm_tumblr			8.418	12.004			5.231	12.260			2.643***	3.808***
			(18.386)	(18.922)			(8.903)	(9.151)			(0.718)	(0.756)
firm_twilio			11.977	13.025			7.517	11.536			2.071***	3.278***
			(17.904)	(18.349)			(8.669)	(8.874)			(0.713)	(0.784)
firm_twitter			10.452	11.960			4.574	9.325			2.334***	3.544***
			(11.746)	(12.329)			(5.687)	(5.963)			(0.486)	(0.525)

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
firm_ValveSoftware			147.585***	149.228***			39.073***	39.290***			5.031***	5.060***
			(28.443)	(28.463)			(13.773)	(13.765)			(0.957)	(0.972)
firm_venmo			11.621	17.151			9.068	16.387**			2.216***	3.375***
			(15.399)	(16.134)			(7.457)	(7.803)			(0.670)	(0.724)
firm_xamarin			8.392	7.062			8.570	17.036**			2.387***	2.533***
			(13.523)	(15.300)			(6.548)	(7.399)			(0.571)	(0.652)
firm_yahoo			12.268	9.708			8.536**	11.367**			0.693	1.850***
			(8.811)	(9.743)			(4.266)	(4.712)			(0.598)	(0.634)
firm_Yalantis			12.839	15.420			11.149	20.419**			4.407***	5.533***
			(17.956)	(18.727)			(8.695)	(9.057)			(0.623)	(0.670)
firm_Yelp			14.280	20.465*			8.881	11.987**			1.416**	2.707***
			(11.501)	(12.270)			(5.569)	(5.934)			(0.556)	(0.613)
firm_yhat			37.694	41.471*			12.393	14.404			2.774***	4.315***
			(24.013)	(24.392)			(11.628)	(11.796)			(0.807)	(0.836)
Constant	8.443***	−2.780	−19.593***	−34.006***	5.114***	4.488	−14.598***	−20.129***	−3.426***	−2.546***	−5.820***	−6.276***
	(2.981)	(7.707)	(7.118)	(10.018)	(1.533)	(3.933)	(3.447)	(4.845)	(0.140)	(0.251)	(0.437)	(0.529)

	Dependent variable:											
	Number of Issues				Number of Contributors				Top Project			
	normal				*normal*				*logistic*			
	(3.13)	(3.14)	(3.15)	(3.16)	(3.17)	(3.18)	(3.19)	(3.20)	(3.21)	(3.22)	(3.23)	(3.24)
Observations	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−19,630.080	−19,616.630	−19,514.800	−19,507.690	−17,355.340	−17,316.430	−17,035.330	−17,023.880	−1,458.752	−1,373.897	−1,208.251	−1,150.787
Akaike Inf. Crit.	39,266.160	39,257.260	39,149.610	39,153.370	34,716.680	34,656.870	34,190.660	34,185.760	2,923.504	2,771.795	2,536.502	2,439.574

Note: *p<0.1; **p<0.05; ***p<0.01

Table 43: Detailed Regression Table: Impact of ”Age” and ”Ratio” on ”Issues”, ”Contributors”, ”Popularity” and ”Top Project” through *Fitting Generalized Linear Models.* Using (a) dummy variables for each *Firm* in model 3.15, 3.19 and 3.23 (b) dummy variables for each *Language* in model 3.14, 3.18, 3.22 (c) dummy variables for each *Firm and Language* in model 3.16, 3.20 and 3.24

	Dependent variable:					
	Stars	Subscribers	Forks	Number of Issues	Number of Contributors	Top Project
	linear *mixed-effects*	*linear* *mixed-effects*	*linear* *mixed-effects*	*linear* *mixed-effects*	*linear* *mixed-effects*	*generalized linear* *mixed-effects*
	(3.31)	(3.32)	(3.33)	(3.34)	(3.35)	(3.36)
Ratio	168.200**	17.839***	33.219	5.788	1.305	1.765***
	(82.283)	(6.258)	(20.382)	(4.133)	(2.042)	(0.189)
Age	0.466***	0.038***	0.149***	0.020***	0.021***	0.001***
	(0.050)	(0.004)	(0.012)	(0.002)	(0.001)	(0.0001)
Constant	102.770	43.266***	−18.116	2.346	1.793	−3.393***
	(119.455)	(10.364)	(24.220)	(4.530)	(3.133)	(0.315)
Observations	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−29,760.380	−20,956.550	−25,023.000	−19,581.640	−17,133.280	−1,252.915
Akaike Inf. Crit.	59,532.760	41,925.100	50,058.000	39,175.290	34,278.570	2,515.830
Bayesian Inf. Crit.	59,569.590	41,961.920	50,094.820	39,212.110	34,315.390	2,546.516

Note: *p<0.1; **p<0.05; ***p<0.01

Table 44: Impact of ”Age” and ”Ratio” on projects’ social metrics (”Stars”, ”Subscribers”, ”Forks” and ”Issues”) and popularity (is ”Top Project” yes / no) through *Mixed Model* between *Firms* and *Programming Languages*. ”Number of Contributors” is just a control attribute and not a describing model.

	Dependent variable:					
	Ratio					
	(3.37)	(3.38)	(3.39)	(3.40)	(3.41)	(3.42)
Stars	0.00000					
	(0.00000)					
Subscribers		0.0001^{**}				
		(0.00005)				
Forks			0.00001			
			(0.00001)			
Number of Issues				0.0001		
				(0.0001)		
Number of Contributors					−0.00004	
					(0.0001)	
Top Project						0.137^{***}
						(0.014)
Constant	0.549^{***}	0.542^{***}	0.550^{***}	0.550^{***}	0.552^{***}	0.515^{***}
	(0.024)	(0.024)	(0.024)	(0.024)	(0.024)	(0.023)
Observations	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−819.663	−814.992	−818.802	−816.630	−816.721	−767.020
Akaike Inf. Crit.	1,647.326	1,637.985	1,645.603	1,641.260	1,641.442	1,542.039
Bayesian Inf. Crit.	1,671.874	1,662.533	1,670.152	1,665.809	1,665.990	1,566.588
Note:	*p<0.1; **p<0.05; ***p<0.01					

Table 45: Impact of projects' social metrics on ”Ratio” using *Mixed-Effects Models* between firms (Model 3.37 - 3.42)

	Dependent variable:					
	Ratio					
	(3.43)	(3.44)	(3.45)	(3.46)	(3.47)	(3.48)
Stars	0.00001					
	(0.00000)					
Subscribers		0.0001^{**}				
		(0.00005)				
Forks			0.00001			
			(0.00001)			
Number of Issues				0.0001		
				(0.0001)		
Number of Contributors					−0.00004	
					(0.0001)	
Top Repo						0.125^{***}
						(0.014)
Constant	0.555^{***}	0.550^{***}	0.557^{***}	0.556^{***}	0.559^{***}	0.521^{***}
	(0.029)	(0.029)	(0.029)	(0.029)	(0.029)	(0.027)
Observations	3,419	3,419	3,419	3,419	3,419	3,419
Log Likelihood	−795.593	−791.577	−794.918	−792.955	−792.904	−751.717
Akaike Inf. Crit.	1,601.185	1,593.154	1,599.837	1,595.910	1,595.808	1,513.433
Bayesian Inf. Crit.	1,631.871	1,623.839	1,630.522	1,626.595	1,626.494	1,544.119
Note:	*p<0.1; **p<0.05; ***p<0.01					

Table 46: Impact of projects' social metrics on ”Ratio” using *Mixed-Effects Models* between firms and programming languages (Model 3.43 - 3.48)

C Graphics and Plots

C.1 Introduction and Theory

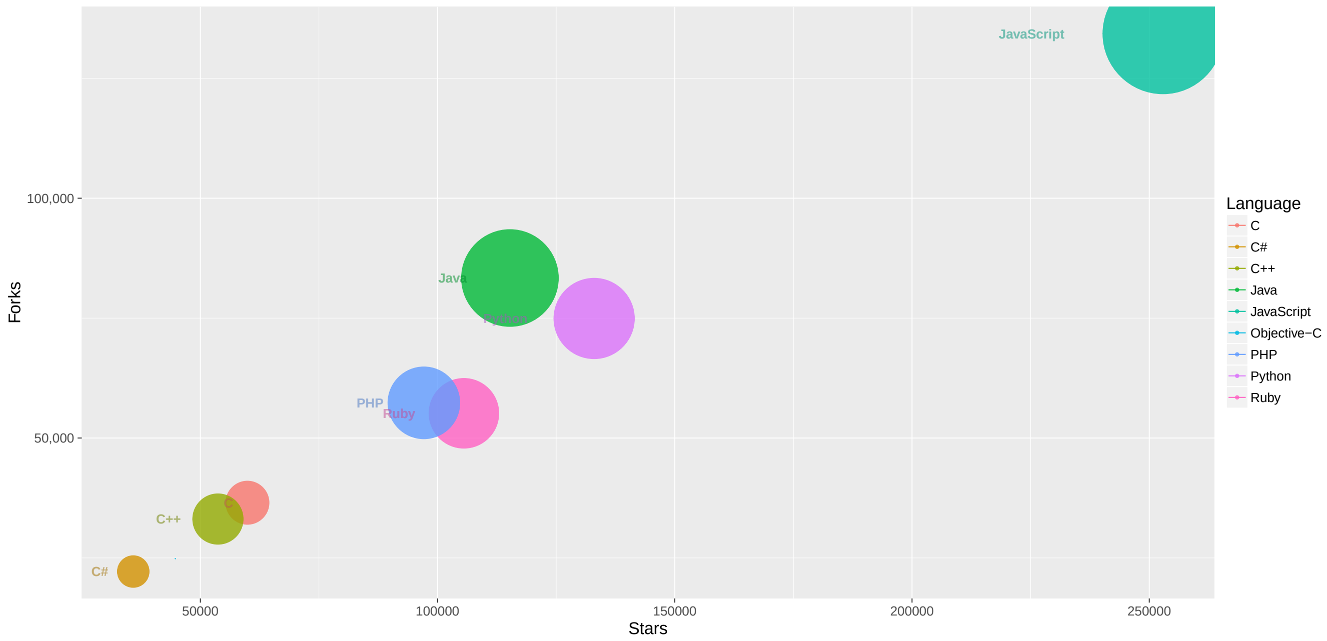

Figure 13: Most popular languages on GitHub in 2014. The cumulated numbers of forks and stars on all projects can be regarded as proxy for the popularity of the programming language. The size of the circles represents the number of existing projects (proportions: JavaScript contains 323,938 projects, C# contains 56,062 projects) [87]

[87] Data origin: `http://githut.info/`

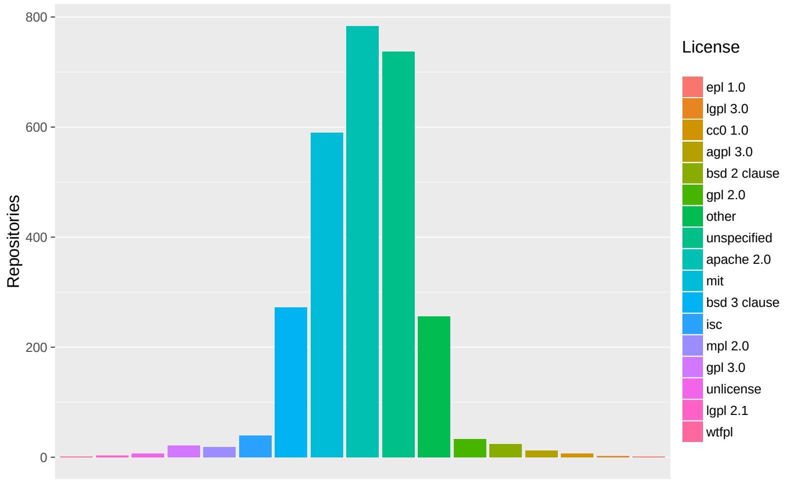

Figure 14: License distribution of firm's residual projects on GitHub.

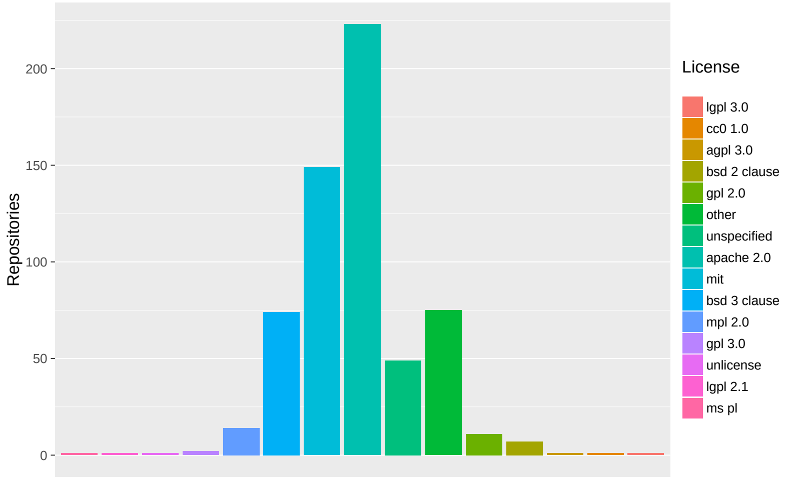

Figure 15: License distribution of firm's top projects on GitHub.

C.2 Plots of Code Contribution

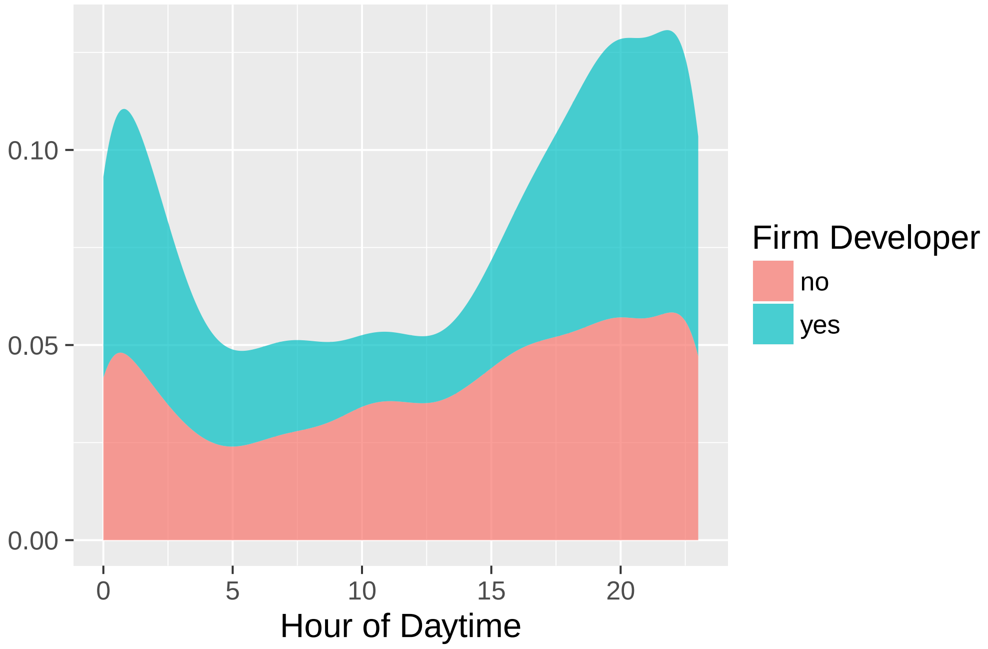

Figure 16: Code contributions by firm employed and external developers based on worldwide consideration (UTC time). Most commits are placed during the late evening which is between morning and midday (9:00 - 16:00) in California, USA (UTC -8)

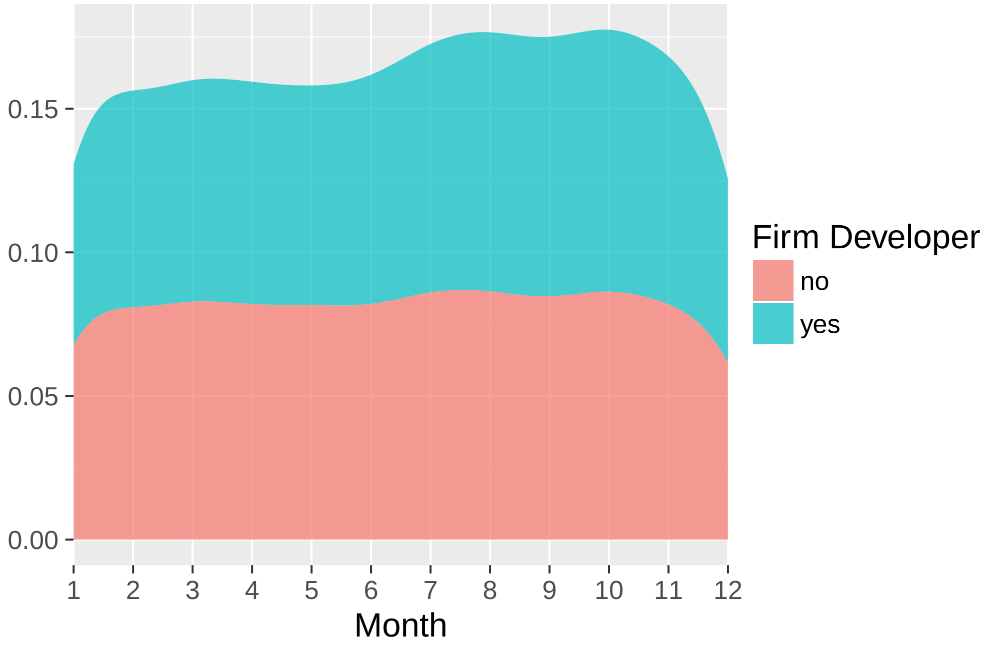

Figure 17: Code contributions by firm employed and external developers over the year (January - December)

C.3 Plots of Repositories

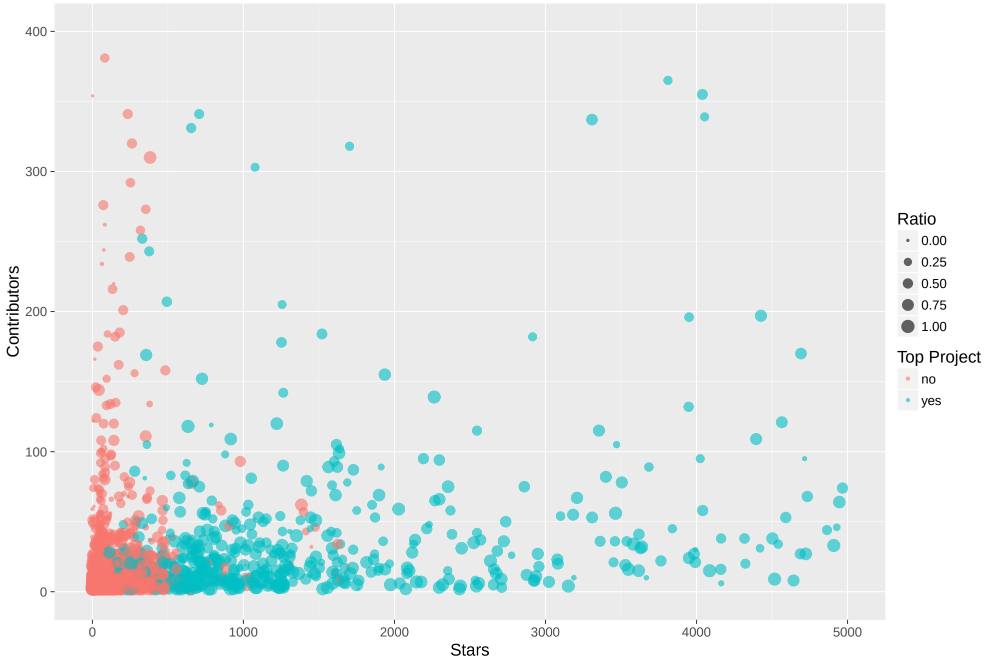

Figure 18: The "ratio" (code commitment share of firm employed developers), represented by size of dots, is quite high on all observed projects (≥ 0.5)

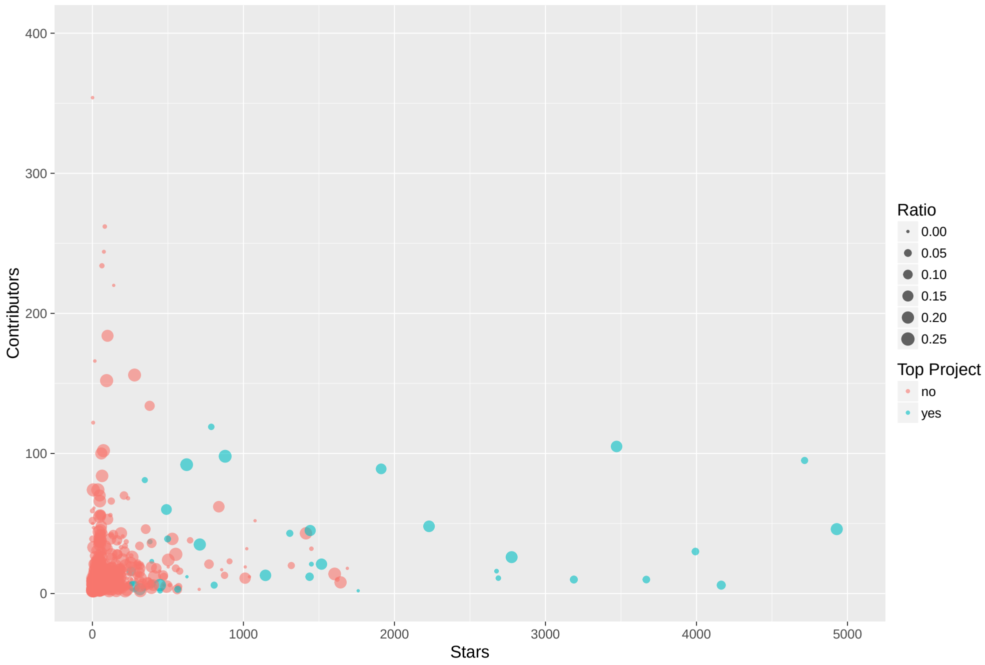

Figure 19: If the ratio is low (≤ 0.25) there are fewer top projects / less stargazers

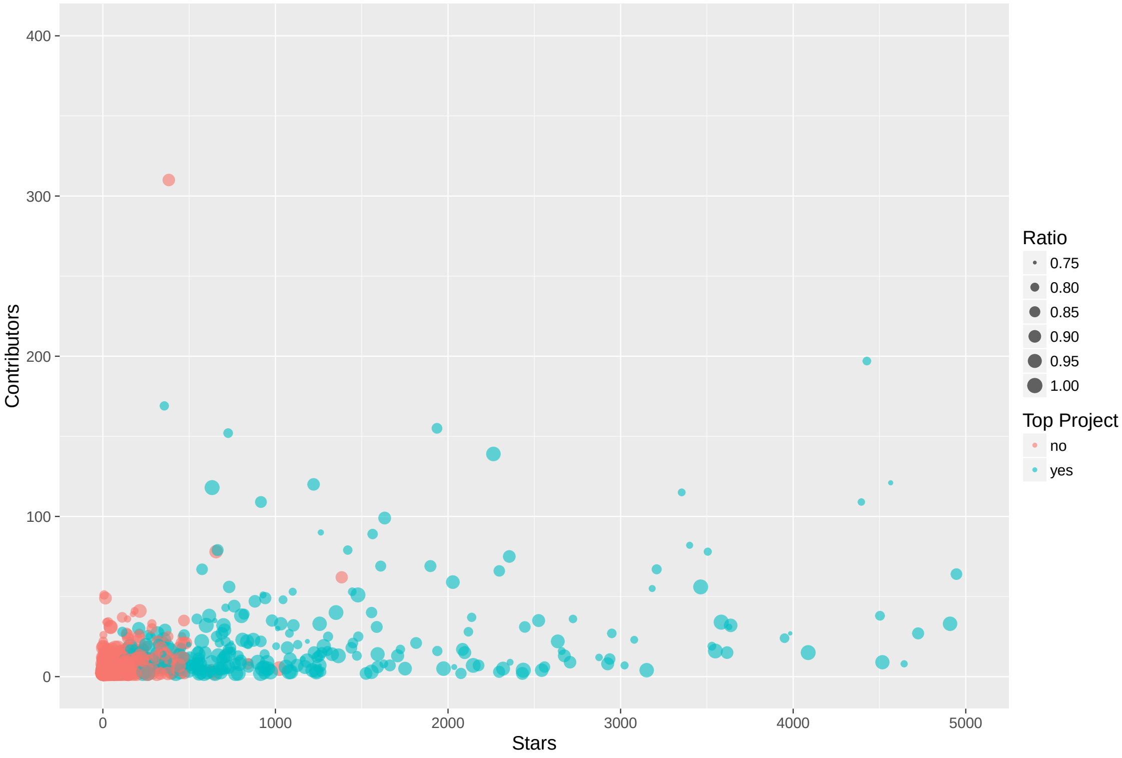

Figure 20: If the ratio is higher (≥ 0.75) there are more top projects / stargazers

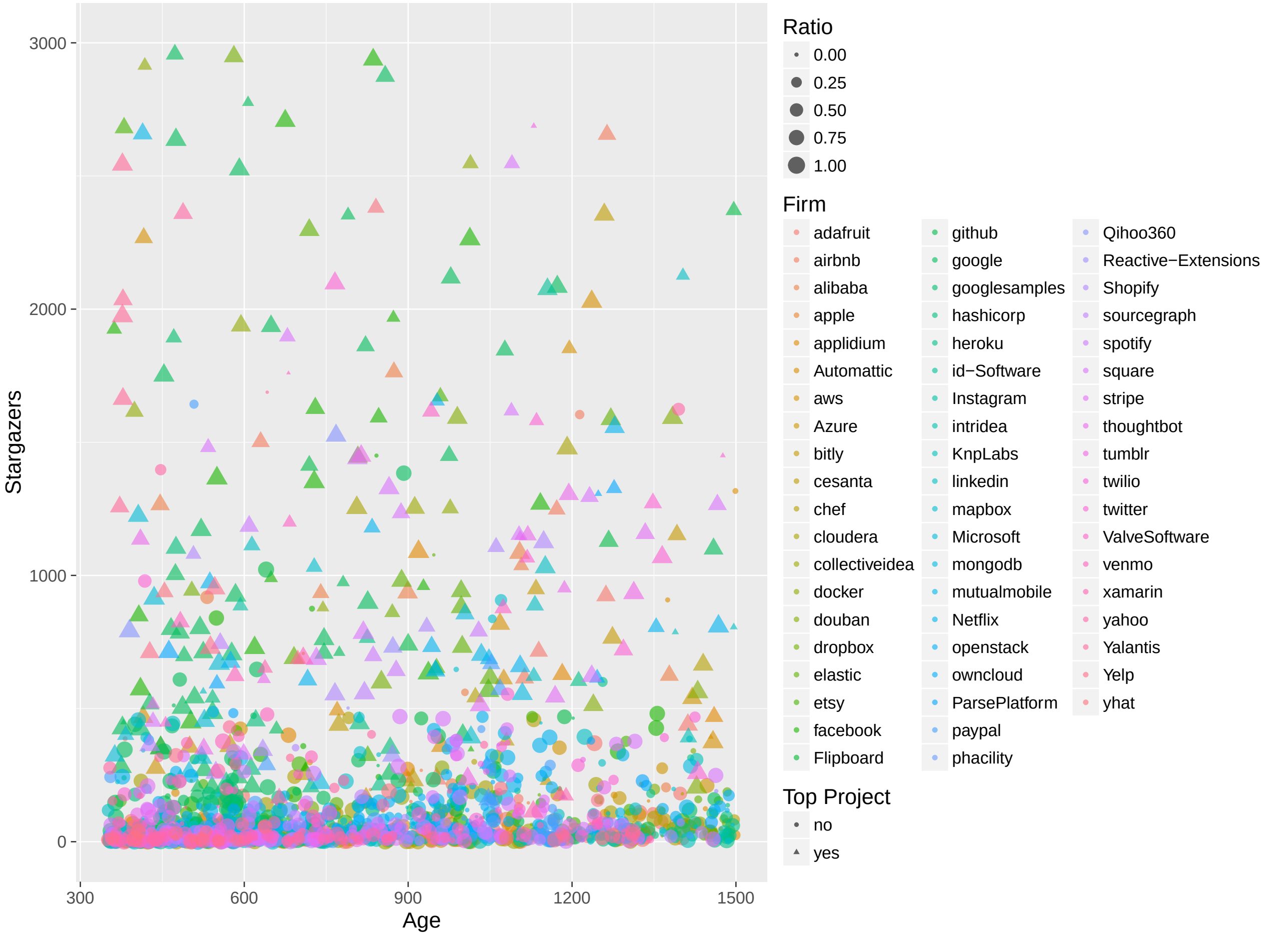

Figure 21: The age of projects (in days), their popularity (number of stars), colored by firms, size of shapes by ”ratio” (code commitment share of firm employed developers) and shape by ”Top Project” or ”Residual Project”. All plotted projects are less than 4.5 years old.

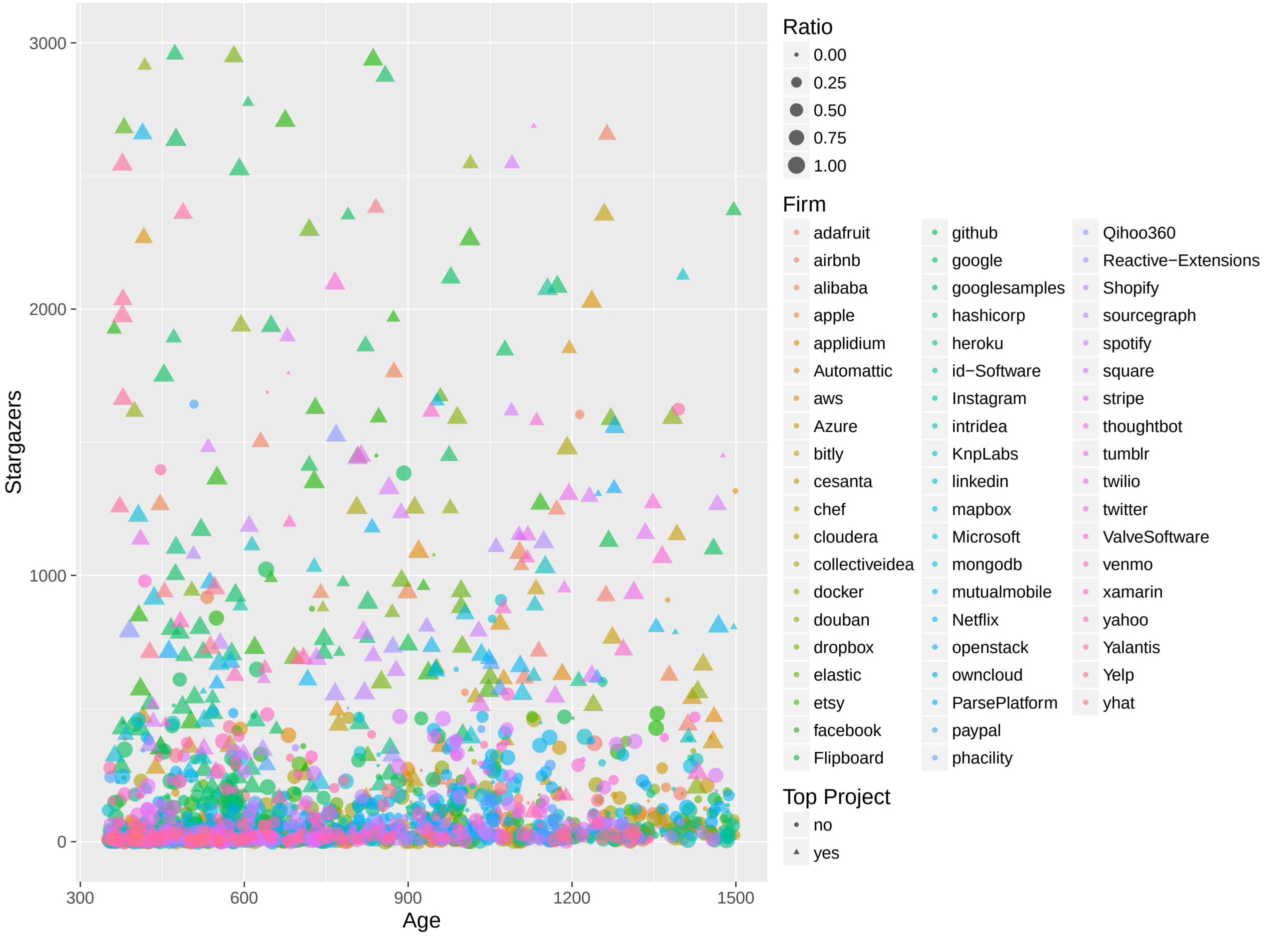

Figure 22: The age of projects (in days), their popularity (number of stars), colored by (all) firms, size of shapes by ”ratio” (code commitment share of firm employed developers) and shape by ”Top Project” or ”Residual Project”. All plotted projects are less than 4.5 years old.

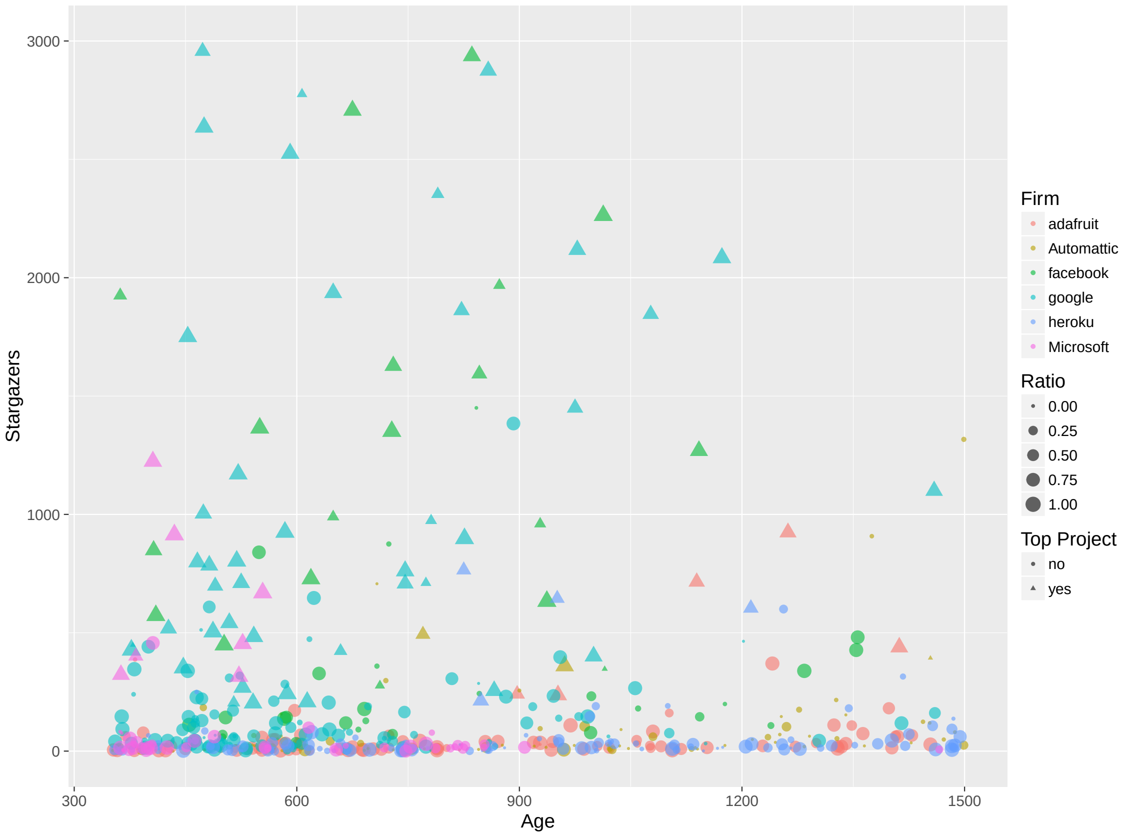

Figure 23: The age of projects (in days), their popularity (number of stars), colored by firms with most popular OS projects, size of shapes by ”ratio” (code commitment share of firm employed developers) and shape by ”Top Project” or ”Residual Project”. All plotted projects are less than 4.5 years old.

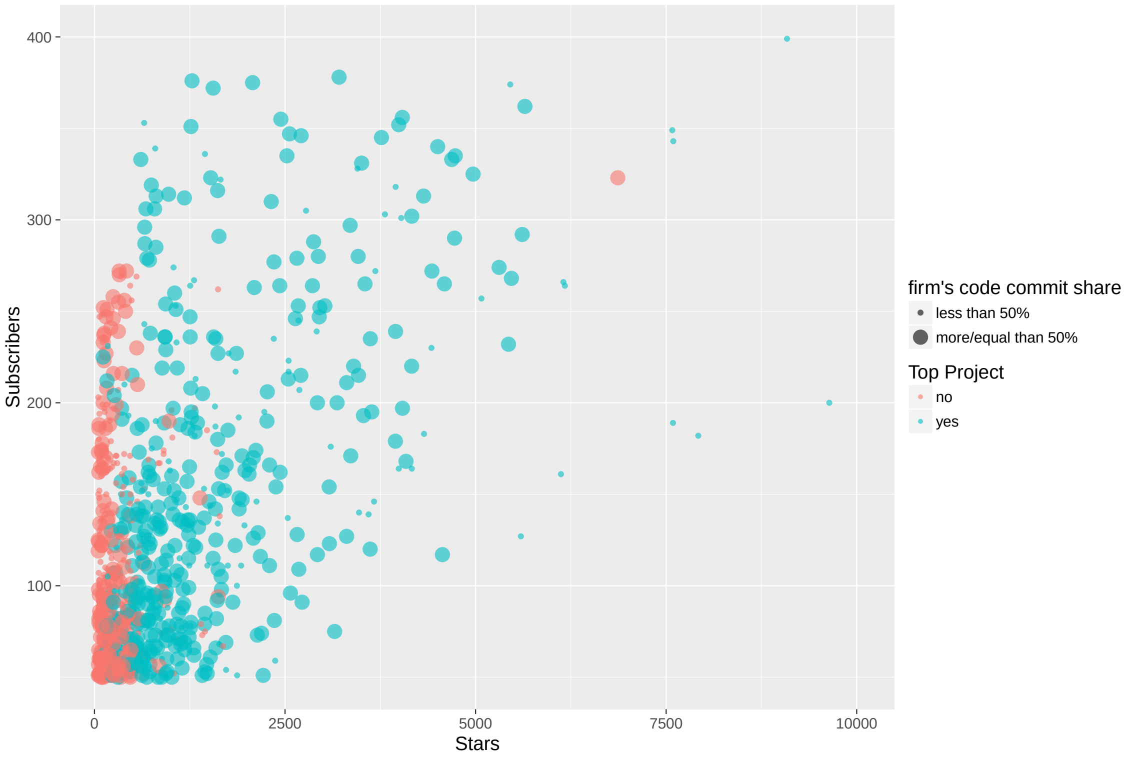

Figure 24: Seperation in ratio < 0.5 and ≥ 0.5; Plotted with Stars against Subscribers (proxies for popularity), colored in Top and Residual Projects and scaled x-axis up to 10,000 stars.

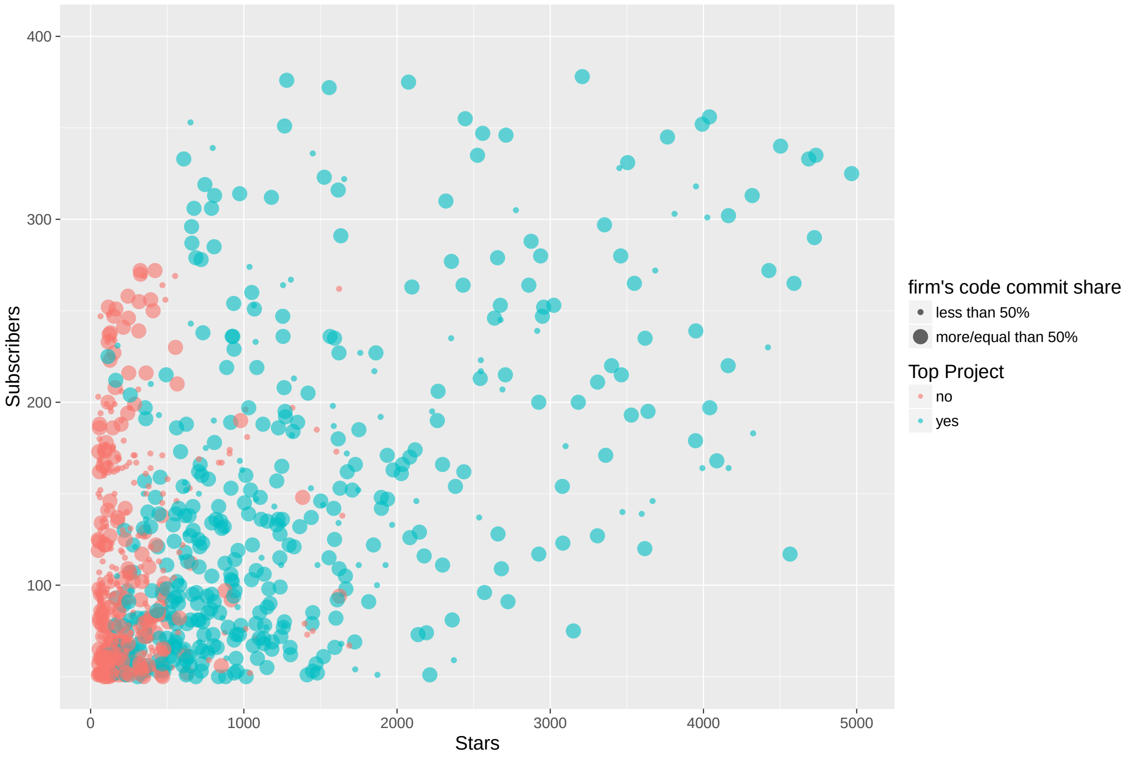

Figure 25: Same as figure 24 but with scaled x-axis up to 5000 stars.

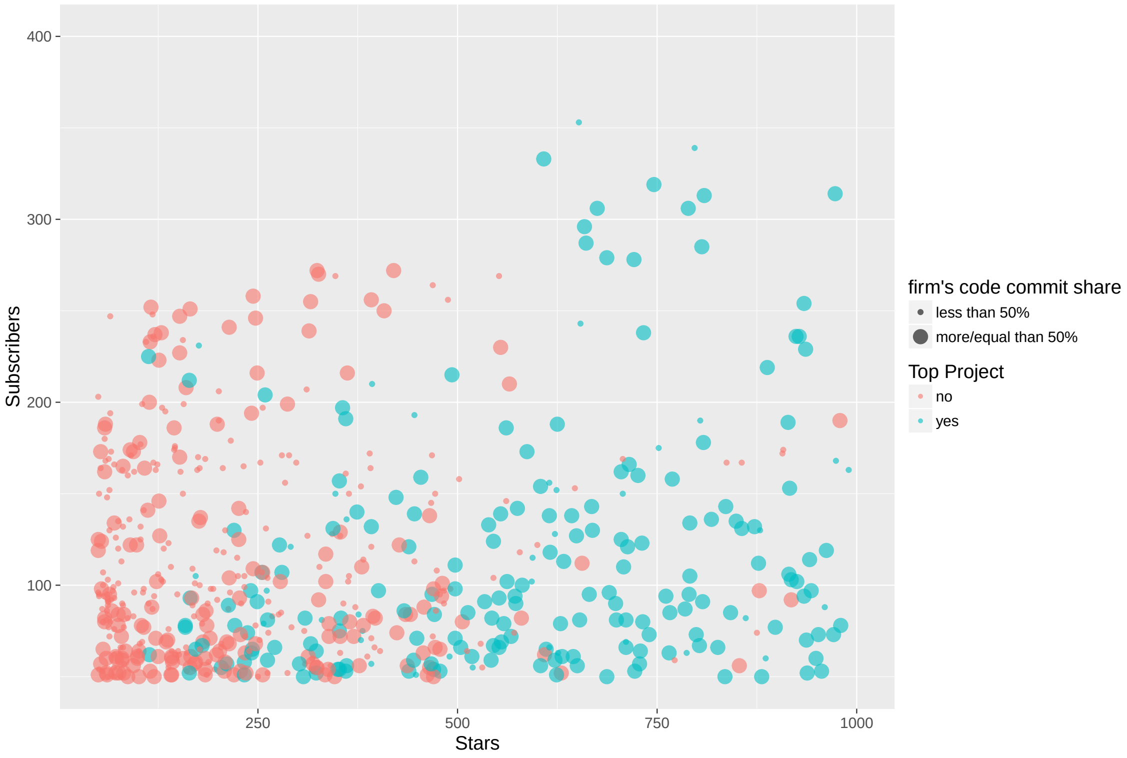

Figure 26: Same as figure 24 but with scaled x-axis up to 1000 stars.

C.4 Plots for H 1.1

See page 40 for explanation of slopes and axes.

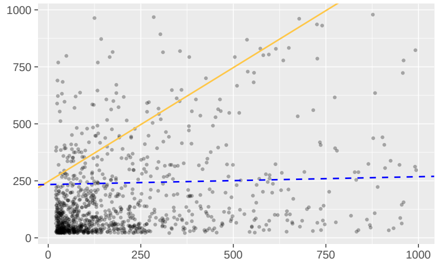

Figure 27: All Projects (Model 1.1.1 - 1.1.2)

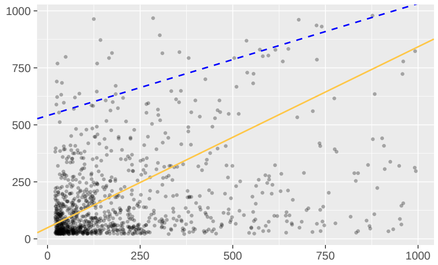

Figure 28: Top Projects (Model 1.1.3 - 1.1.4)

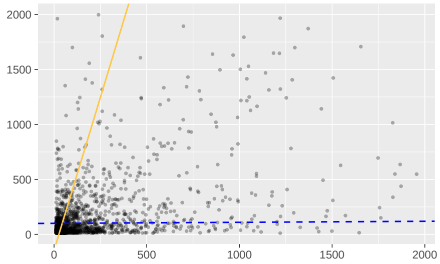

Figure 29: Residual Projects (Model 1.1.5 - 1.1.6)

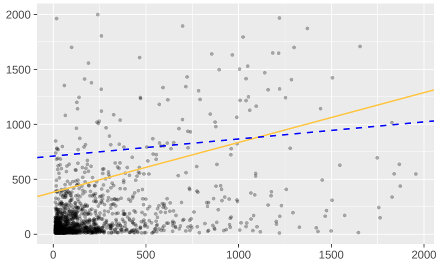

Figure 30: Older Top Projects (Model 1.1.7 - 1.1.8)

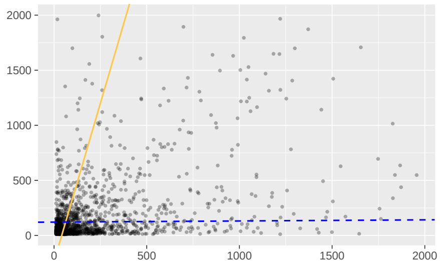

Figure 31: Older Residual Projects (Model 1.1.9 - 1.1.10)

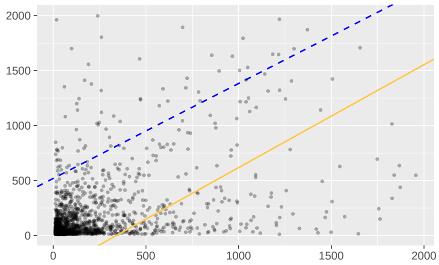

Figure 32: Younger Top Projects (Model 1.1.11 - 1.1.12)

C.5 Plots for H 1.2.1

See page 40 for explanation of slopes and axes.

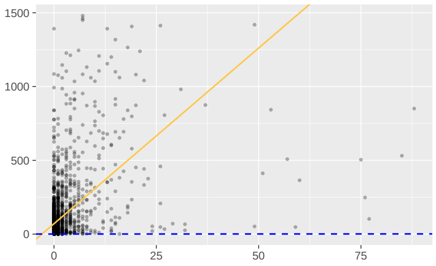

Figure 33: All Projects (Model 1.2.1 - 1.2.2)

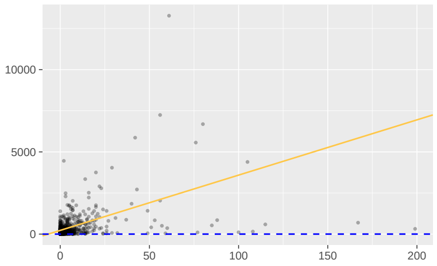

Figure 34: Top Projects (Model 1.2.3 - 1.2.24)

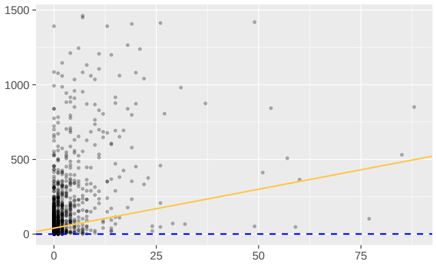

Figure 35: Residual Projects(Model 1.2.5 - 1.2.6)

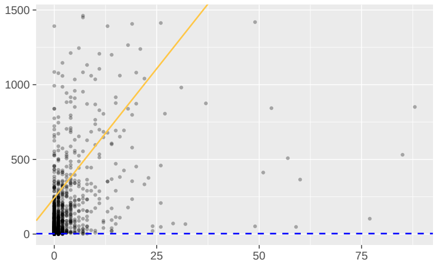

Figure 36: Older Top Projects (Model 1.2.7 - 1.2.8)

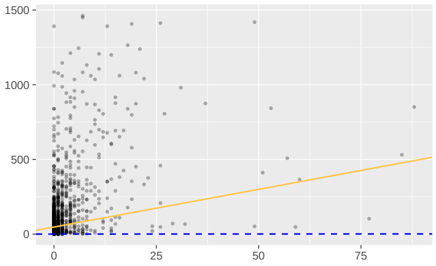

Figure 37: Older Residual Projects (Model 1.2.9 - 1.2.10)

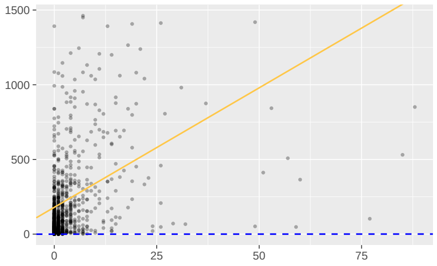

Figure 38: Younger Top Projects (Model 1.2.11 - 1.2.12)

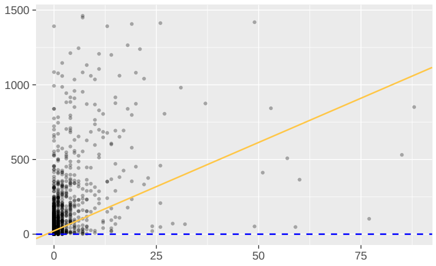

Figure 39: Younger Residual Projects (Model 1.2.13 - 1.2.14)

C.6 Plots for H 1.2.2

See page 40 and 45 for explanation of slopes and axes.

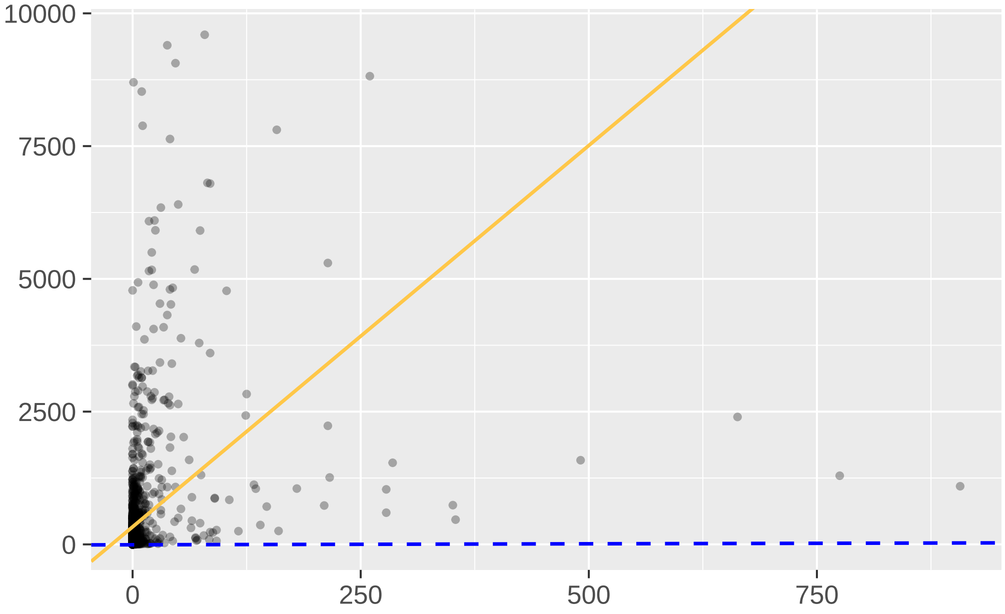

Figure 40: Comments of Issues (Model 1.3.3- 1.3.4)

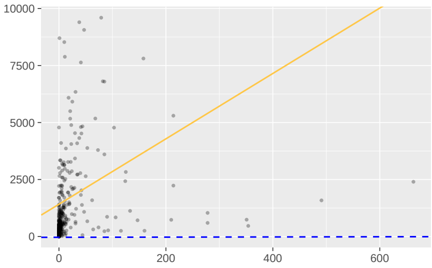

Figure 41: Comments of Issues (Model 1.3.3- 1.3.4)

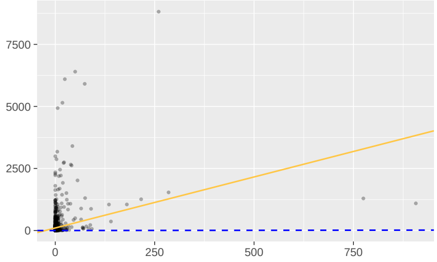

Figure 42: Comments of Issues (Model 1.3.5- 1.3.6)

D Open Source Software used for making this Study

The following open source software is used to receive, process, analyze and present the data (this list is not fully inclusive):

Atom Editor (`https://github.com/atom/atom`)

R Studio (`https://github.com/rstudio/rstudio`)

- foreign (Venables and Ripley (2002))
- MASS (R Core Team (2015))
- xtable (Dahl (2016))
- stargazer (Hlavac (2015))
- stringr (Wickham (2015))
- ggplot2 (Wickham (2009))
- scales (Wickham (2016))
- PerformanceAnalytics (Peterson and Carl (2014))
- xts (Ryan and Ulrich (2014))
- TeachingDemos (Snow (2016))

NodeJS (`https://github.com/nodejs/node`) using the following npm modules:

- csv (0.4.6)
- csv-stringify (v0.0.8)
- coffee-script (v1.9.0)
- fast-csv (v0.6.0)
- github (v0.2.4)
- glob (v6.0.4)
- globs (v0.1.2)
- google (v1.4.0)
- js-yaml (v3.4.3)
- kerberos (v0.0.18)
- minimist (v1.2.0)
- moment (v2.10.6)
- moment-timezone (v0.4.1)
- mongodb (v2.1.7)
- mongoskin (v2.0.0)
- nodemailer (v1.8.0)
- procstreams (v0.3.0)
- progress (v1.1.8)
- request (v2.61.0)

sequence (v3.0.0)

split (v1.0.0)

underscore (v1.8.3)

yargs (v3.27.0)

MariaDB (`https://github.com/MariaDB/server`)

MongoDB (`https://github.com/mongodb/mongo`)

Texmaker (`http://www.xm1math.net/texmaker/download.html`)

LaTeX (`https://www.latex-project.org/`)

References

Afuah, A. and C. L. Tucci (2012): "Crowdsourcing as a solution to distant search," *Academy of Management Review*, 37, 355–375.

Agresti, A. et al. (2007): "An introduction to categorical data analysis. Hoboken," .

Alexy, O., P. Criscuolo, and A. Salter (2012): "Managing unsolicited ideas for R&D," *California Management Review*, 54, 116–139.

Amadeo, R. (2013): "Google's iron grip on Android: Controlling open source by any means necessary," http://arstechnica.com/gadgets/2013/10/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/, (Accessed on 03/26/2016).

Bates, D., M. Mächler, B. Bolker, and S. Walker (2015): "Fitting Linear Mixed-Effects Models Using lme4," *Journal of Statistical Software*, 67, 1–48.

Ben Ogle, G. (2016): "Atom1.0," http://blog.atom.io/2015/06/25/atom-1-0.html, (Visited on 01/20/2016).

Black Duck, m. (2015): "Top 20 Open Source Licenses — Black Duck," https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses, (Visited on 12/25/2015).

Bobbie Johnson, T. G. (2008): "Cloud computing is a trap, warns GNU founder," http://www.theguardian.com/technology/2008/sep/29/cloud.computing.richard.stallman, (Visited on 01/24/2016).

Bonaccorsi, A. and C. Rossi (2003): "Why open source software can succeed," *Research policy*, 32, 1243–1258.

Bonaccorsi, A. and C. Rossi (2006): "Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business," *Knowledge, Technology & Policy*, 18, 40–64.

Bonaccorsi, A. and C. Rossi Lamastra (2003): "Licensing schemes in the production and distribution of open source software: An empirical investigation," *Available at SSRN 432641*.

Bray, T. (2014): "The javascript object notation (json) data interchange format," .

Brian Doll, I. G. (2012): *Analyzing Millions of GitHub Commits*, (Accessed on 03/12/2016).

Bui, Q. (2014): "Who's In The Office? The American Workday In One Graph : Planet

Money : NPR," http://www.npr.org/sections/money/2014/08/27/343415569/whos-in-the-office-the-american-workday-in-one-graph, (Accessed on 03/14/2016).

Buxmann, P., T. Hess, and D.-W.-I. S. Lehmann (2008): "Software as a Service," *Wirtschaftsinformatik*, 50, 500–503.

Cade Metz, w. (2015): "Github's Top Coding Languages Show Open Source Has Won," http://www.wired.com/2015/08/github-data-shows-changing-software-landscape/, (Visited on 01/17/2016).

Chacon, S. and B. Straub (2014): *Pro git*, Apress.

Chesbrough, H. (2003): "The logic of open innovation: managing intellectual property," *California Management Review*, 45, 33–58.

Chesbrough, H. W. (2006): *Open innovation: The new imperative for creating and profiting from technology*, Harvard Business Press.

Cohen, W. M. and D. A. Levinthal (1990): "Absorptive capacity: A new perspective on learning and innovation," *Administrative science quarterly*, 128–152.

Dabbish, L., C. Stuart, J. Tsay, and J. Herbsleb (2012): "Social coding in GitHub: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, ACM, 1277–1286.

Dahl, D. B. (2016): *xtable: Export Tables to LaTeX or HTML*, r package version 1.8-2.

Dahlander, L. and D. M. Gann (2010): "How open is innovation?" *Research policy*, 39, 699–709.

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

Dahlander, L. and H. Piezunka (2014): "Open to suggestions: How organizations elicit suggestions through proactive and reactive attention," *Research Policy*, 43, 812–827.

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

Dan Morrill, G. (2008): "Announcing the Android 1.0 SDK, release 1," http://android-developers.blogspot.de/2008/09/announcing-android-10-sdk-release-1.html, (Visited on 01/23/2016).

Dohm, L. (2016): "Atom Reaches One Million Active Users," http://blog.atom.io/2016/03/28/atom-reaches-1m-users.html, (Accessed on 03/29/2016).

Eilhard, J. (2009): “Open source incorporated,” *Available at SSRN 1360604.*

Eilhard, J. (2010): “Tapping into the source: corporate involvement in open source software,” Ph.D. thesis, École Nationale Supérieure des Mines de Paris.

Facebook (2015): “React Native — A framework for building native apps using React,” `https://facebook.github.io/react-native/`, (Visited on 01/03/2016).

Facebook (2016): “React Native Readme,” `https://github.com/facebook/react-native/blob/d8e138770f20e6fe5c6ba86840c9ec0d74400011/README.md`, (Visited on 01/03/2016).

Feller, J., B. Fitzgerald, et al. (2002): *Understanding open source software development*, Addison-Wesley London.

Finnegan, M. (2015): “Why Google’s programming language can rival Java in the enterprise — Apps — Techworld,” `http://www.techworld.com/apps/why-googles-go-programming-language-could-rival-java-in-enterprise-3626140/`, (Visited on 01/17/2016).

Fitzgerald, B. (2006): “The transformation of open source software,” *Mis Quarterly*, 587–598.

Fleming, L. and D. M. Waguespack (2007): “Brokerage, boundary spanning, and leadership in open innovation communities,” *Organization science*, 18, 165–180.

Foley, M. J. (2008): “Microsoft’s open-source strategy: A picture is worth a thousand words — ZDNet,” `http://www.zdnet.com/article/microsofts-open-source-strategy-a-picture-is-worth-a-thousand-words/`, (Visited on 01/09/2016).

Foundation, F. S. (2007): “GNU Affero General Public License, Version 3.0 - GNU-Projekt - Free Software Foundation,” `http://www.gnu.org/licenses/agpl-3.0.de.html`, (Visited on 01/24/2016).

Foundation, F. S. (2016): “What is free software? - GNU Project - Free Software Foundation,” `https://www.gnu.org/philosophy/free-sw.en.html`, (Visited on 12/25/2015).

Frederic Lardinois, T. (2015): “Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows,” `http://techcrunch.com/2015/04/29/`
`microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-wind`
(Visited on 01/18/2016).

Gal, A. (2016): “Oracle sinks its claws into Android,” `http://andreasgal.com/2016/01/05/oracle-sinks-its-claws-into-android/`, (Accessed on 03/26/2016).

Gartner (2015): "Gartner Says Tablet Sales Continue to Be Slow in 2015," https://www.gartner.com/newsroom/id/2954317, (Visited on 01/05/2016).

Ghosh, R. A. (2007): "Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU," .

GitHub (2014): "Forking Projects · GitHub Guides," https://guides.github.com/activities/forking/, (Visited on 2015.10.07).

GitHub (2015a): "GitHub API Limit on Search," https://developer.github.com/v3/search/#rate-limit, (Visited on 2015.10.06).

GitHub (2015b): "GitHub API v3," https://developer.github.com/v3/, (Visited on 2015.10.07).

GitHub (2015c): "Starring — GitHub API," https://developer.github.com/v3/activity/starring/, (Visited on 2015.10.07).

GitHub (2015d): "Watching — GitHub API," https://developer.github.com/v3/activity/watching/, (Visited on 2015.10.07).

GitHub (2016a): "Electron," http://electron.atom.io/, (Visited on 01/18/2016).

GitHub (2016b): "Press," https://github.com/about/press, (Visited on 01/17/2016).

GitHub and A. La (2015): "Language Trends on GitHub," https://github.com/blog/2047-language-trends-on-github, (Visited on 2015.10.06).

GitHub and K. Neath (2007): "Issues 2.0: The Next Generation," https://github.com/blog/831-issues-2-0-the-next-generation, (Visited on 2015.10.07).

Greene, T. C. (2001): "Ballmer: "Linux is a cancer"," http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer/, (Accessed on 03/20/2016).

Grigorik, I. (2012): "The github archive," .

Gruber, M., I. C. MacMillan, and J. D. Thompson (2008): "Look before you leap: Market opportunity identification in emerging technology firms," *Management Science*, 54, 1652–1665.

Hall, S. (2015): "GitLab Raises $1.5 Million as Code Sharing and Open Source Gains Acceptance - The New Stack," http://thenewstack.io/gitlab-raises-1-5-million-as-code-sharing-and-open-source-gains-acceptance/, (Visited on 01/03/2016).

Hanna, J. (2010): "Mixing Open Source and Proprietary Software Strategies," http://

hbswk.hbs.edu/item/mixing-open-source-and-proprietary-software-strategies, (Accessed on 03/20/2016).

Hars, A. and S. Ou (2001): "Working for free? Motivations of participating in open source projects," in *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, IEEE, 9–pp.

Hawkins, R. E. (2004): "The economics of open source software for a competitive firm," *NETNOMICS: Economic Research and Electronic Networking*, 6, 103–117.

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

Hertel, G., S. Niedner, and S. Herrmann (2003): "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel," *Research policy*, 32, 1159–1177.

Hilbe, J. M. (2009): *Logistic regression models*, CRC Press.

Hill, S. A. and J. M. Birkinshaw (2009): "Idea sets: conceptualizing and measuring a new unit of analysis in entrepreneurship research," *Organizational research methods.*

Hippel, E. v. and G. v. Krogh (2003): "Open source software and the "private-collective" innovation model: Issues for organization science," *Organization science*, 14, 209–223.

Hlavac, M. (2015): *stargazer: Well-Formatted Regression and Summary Statistics Tables*, Harvard University, Cambridge, USA, r package version 5.2.

Holman, Z. (2011): "Introducing GitHub Enterprise," https://github.com/blog/978-introducing-github-enterprise, (Accessed on 03/27/2016).

Hyett, P. (2011): "GitHub Dominates the Forges," https://github.com/blog/865-github-dominates-the-forges, (Accessed on 03/27/2016).

Jeppesen, L. B. and K. R. Lakhani (2010): "Marginality and problem-solving effectiveness in broadcast search," *Organization science*, 21, 1016–1033.

Joel Rosenblatt, B. (2016): "Google's Android Generates $31 Billion Revenue, Oracle Says," http://www.bloomberg.com/news/articles/2016-01-21/google-s-android-generates-31-billion-revenue-oracle-says-ijor8hvt, (Visited on 01/23/2016).

Katila, R. and G. Ahuja (2002): "Something old, something new: A longitudinal study of search behavior and new product introduction," *Academy of management journal*, 45, 1183–1194.

Koçulu, A. (2016): "I've Just Liberated My Modules," https://medium.com/@azerbike/i-ve-just-liberated-my-modules-9045c06be67c#.8ox036r7i, (Accessed on 03/23/2016).

Krishnamurthy, R., V. Jacob, S. Radhakrishnan, and K. Dogan (2016): "Peripheral Developer Participation in Open Source Projects: An Empirical Analysis," *ACM Transactions on Management Information Systems (TMIS)*, 6, 14.

Kuhn, B. M. (2016): "Sun, Oracle, Android, Google and JDK Copyleft FUD," http://ebb.org/bkuhn/blog/2016/01/05/jdk-in-android.html, (Accessed on 03/26/2016).

Lakhani, K., B. Wolf, J. Bates, and C. DiBona (2002): "The boston consulting group hacker survey," *Boston, The Boston Consulting Group.*

Laurent, A. M. S. (2004): *Understanding open source and free software licensing*, O'Reilly Media, Inc.

Laursen, K. and A. Salter (2006): "Open for innovation: the role of openness in explaining innovation performance among UK manufacturing firms," *Strategic management journal*, 27, 131–150.

Lerner, J. and J. Tirole (2002): "Some simple economics of open source," *The journal of industrial economics*, 50, 197–234.

Lerner, J. and J. Triole (2000): "The simple economics of open source," Tech. rep., National Bureau of Economic Research.

Licensing, O. S. (2004): "Software Freedom and Intellectual Property Law, Lawrence Rosen," .

Loeliger, J. (2006): "Collaborating with GIT," *Linux Magazine, June.*

Loeliger, J. and M. McCullough (2012): *Version Control with Git: Powerful tools and techniques for collaborative software development*, " O'Reilly Media, Inc.".

Metz, C. (2015): "How GitHub Conquered Google, Microsoft, and Everyone Else," http://www.wired.com/2015/03/github-conquered-google-microsoft-everyone-else/, (Accessed on 03/27/2016).

Microsoft (2014): "Announcing .NET 2015 Preview: A New Era for .NET - .NET Blog - Site Home - MSDN Blogs," http://blogs.msdn.com/b/dotnet/archive/2014/11/12/announcing-net-2015-preview-a-new-era-for-net.aspx, (Visited on 12/28/2015).

Microsoft (2015): "VS Code is Open Source!" https://code.visualstudio.com/updates/vNovember#_vs-code-is-open-source, (Visited on 01/18/2016).

Mustonen, M. (2003): "Copyleft—the economics of Linux and other open source software," *Information Economics and Policy*, 15, 99–121.

Occhino, T. (2015): "React Native: Bringing modern web techniques to mobile — Engineering Blog — Facebook Code," https://code.facebook.com/posts/1014532261909640/react-native-bringing-modern-web-techniques-to-mobile/, (Visited on 01/03/2016).

Olson, M. (2009): *The logic of collective action*, vol. 124, Harvard University Press.

O'reilly, T. (2007): "What is Web 2.0: Design patterns and business models for the next generation of software," *Communications & strategies*, 17.

Osterloh, M. and S. Rota (2007): "Open source software development—Just another case of collective invention?" *Research Policy*, 36, 157–171.

ownCloud (2016): "ownCloud Server or Enterprise Edition," https://owncloud.com/lp/owncloud-server-or-enterprise-edition/, (Accessed on 03/29/2016).

Paul Krill, I. (2015): "Java reigns, but Go language spikes in popularity," http://www.infoworld.com/article/2981872/application-development/java-reigns-go-language-spikes-in-popularity.html, (Visited on 01/17/2016).

Peterson, B. G. and P. Carl (2014): *PerformanceAnalytics: Econometric tools for performance and risk analysis*, r package version 1.4.3541.

Piezunka, H. and L. Dahlander (2013): "A study of organizations' attention to suggestions by externals over time," .

Piller, F. T. and D. Walcher (2006): "Toolkits for idea competitions: a novel method to integrate users in new product development," *r&D Management*, 36, 307–318.

Popp, K. and R. Meyer (2010): *Profit from Software Ecosystems: Business Models, Ecosystems and Partnerships in the Software Industry*, BoD–Books on Demand.

R Core Team (2015): *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...*, r package version 0.8-66.

Raymond, E. S. (2001): "The cathedral and the bazaar-musings on Linux and open source by an accidental revoltionary (rev. ed.)." .

Reifman, J. (2007): "Microsoft's Sacred Cash Cow," .

Richard M. Stallman, B. R. (2010): "What Does That Server Really Serve?" http://www.bostonreview.net/richard-stallman-free-software-DRM, (Visited on 01/24/2016).

Rosenkopf, L., A. Metiu, and V. P. George (2001): "From the bottom up? Technical committee activity and alliance formation," *Administrative Science Quarterly*, 46, 748–772.

Rosenkopf, L. and M. L. Tushman (1998): "The coevolution of community networks and technology: Lessons from the flight simulation industry," *Industrial and Corporate Change*, 7, 311–346.

Ryan, J. A. and J. M. Ulrich (2014): *xts: eXtensible Time Series*, r package version 0.9-7.

Schaaf, H. (2014): "The Popularity of Go," http://herman.asia/the-popularity-of-go, (Visited on 01/17/2016).

Seber, G. A. and A. J. Lee (2012): *Linear regression analysis*, vol. 936, John Wiley & Sons.

Serdar Yegulalp, I. (2015): "Atom at 1.0: GitHub's Node-based editor is just getting started," http://www.infoworld.com/article/2940432/development-tools/atom-at-10-githubs-node-based-editor-is-just-getting-started.html, (Visited on 01/18/2016).

Seth, G. (2016): "ChakraCore GitHub repository is now open — Microsoft Edge Dev Blog," https://blogs.windows.com/msedgedev/2016/01/13/chakracore-now-open/, (Visited on 01/14/2016).

Shafranovich, Y. (2005): "Common format and MIME type for Comma-Separated Values (CSV) files," .

Shane, S. (2000): "Prior knowledge and the discovery of entrepreneurial opportunities," *Organization science*, 11, 448–469.

Simcoe, R. (2006): "Public and private participation in the development of and governance of the internet. R. Nelson, ed," *The Limits and Complexities of Market Organization.*

Snow, G. (2016): *TeachingDemos: Demonstrations for Teaching and Learning*, r package version 2.10.

Sobo, N. (2014a): "Atom Is Now Open Source," http://blog.atom.io/2014/05/06/atom-is-now-open-source.html, (Visited on 01/18/2016).

Sobo, N. (2014b): "Introducing Atom," http://blog.atom.io/2014/02/26/introducing-atom.html, (Visited on 01/18/2016).

Stallman, R. M. (2014): "Why "Free Software" is better than "Open Source" - GNU Project - Free Software Foundation," https://www.gnu.org/philosophy/free-software-for-freedom.en.html, (Visited on 12/25/2015).

Stallmann, R. M. (2015): "Copyleft: Pragmatic Idealism - GNU Project - Free Software Foundation," https://www.gnu.org/philosophy/pragmatic.en.html, (Visited on 12/25/2015).

Stamelos, I., L. Angelis, A. Oikonomou, and G. L. Bleris (2002): "Code quality analysis in open source software development," *Information Systems Journal*, 12, 43–60.

Steve Trousdale, R. (2016): "Oracle lawyer says Google's Android generated $31 billion revenue," http://www.reuters.com/article/us-oracle-google-lawsuit-idUSKCN0UZ2W9, (Visited on 01/23/2016).

Surowiecki, J. (2005): *The wisdom of crowds*, Anchor.

Terrell J, K. A. e. a. (2016): "Gender bias in open source: Pull request acceptance of women versus men," *Information Systems Journal*.

Thung, F., T. F. Bissyandé, D. Lo, and L. Jiang (2013): "Network structure of social coding in github," in *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, IEEE, 323–326.

TIOBE (2015): "TIOBE Index for October 2015," http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, retrieved 10.06.2015.

Tsay, J., L. Dabbish, and J. Herbsleb (2014): "Influence of social and technical factors for evaluating contribution in GitHub," in *Proceedings of the 36th international conference on Software engineering*, ACM, 356–366.

Uhlenhuth, K. (2015): "We're moving to GitHub! - The Visual Basic Team - Site Home - MSDN Blogs," http://blogs.msdn.com/b/vbteam/archive/2015/01/10/we-re-moving-to-github.aspx, (Visited on 12/28/2015).

Urban, G. L. and E. Von Hippel (1988): "Lead user analyses for the development of new industrial products," *Management science*, 34, 569–582.

Van Baarsen, J. (2014): *GitLab Cookbook*, Packt Publishing Ltd.

Vasilescu, B., V. Filkov, and A. Serebrenik (2013): "StackOverflow and GitHub: Associations between software development and crowdsourced knowledge," in *Social Computing (SocialCom), 2013 International Conference on*, IEEE, 188–195.

Vaughan-Nichols, S. J. (2014): "Do-it-yourself corporate cloud with ownCloud 6 Enterprise Edition — ZDNet," http://www.zdnet.com/article/do-it-yourself-corporate-cloud-with-owncloud-6-enterprise-edition/, (Visited on 01/03/2016).

Venables, W. N. and B. D. Ripley (2002): *Modern Applied Statistics with S*, New York: Springer, fourth ed., iSBN 0-387-95457-0.

Von Hippel, E. (1976): "The dominant role of users in the scientific instrument innovation process," *Research policy*, 5, 212–239.

Von Hippel, E. (1986): "Lead users: a source of novel product concepts," *Management science*, 32, 791–805.

Von Hippel, E. (1994): ""Sticky information" and the locus of problem solving: implications for innovation," *Management science*, 40, 429–439.

Von Hippel, E. A. (2005): "Democratizing innovation," .

Waguespack, D. M. and L. Fleming (2004): "Penguins, camels, and other birds of a feather: The emergence of leaders in open innovation communities," *Retrieved on January*, 28, 2004.

Weinberger, M. (2015): "Microsoft Windows OEM and Office revenue is down as cloud grows - Business Insider," `http://www.businessinsider.com/microsoft-windows-oem-and-office-revenue-is-down-as-cloud-grows-2015-4?IR=T`, (Accessed on 03/13/2016).

Weis, K. (2014): "GitHub CEO and Co-Founder Chris Wanstrath Keynoting Esri's DevSummit!" `https://blogs.esri.com/esri/arcgis/2014/02/10/github-ceo-and-co-founder-chris-wanstrath-keynoting-esris-devsummit/`, (Accessed on 03/27/2016).

Weiss, A. (2005): "Real world open source: The TCO question. Serverwatch," .

West, J. (2003): "How open is open enough?: Melding proprietary and open source platform strategies," *Research policy*, 32, 1259–1285.

West, J. and S. Gallagher (2006): "Challenges of open innovation: the paradox of firm investment in open-source software," *R&d Management*, 36, 319–331.

West, J. and S. O'mahony (2008): "The role of participation architecture in growing sponsored open source communities," *Industry and Innovation*, 15, 145–168.

Wichmann, T. (2002): "Free/libre and open source software: Survey and study (floss), final report, part II: Firms' open source activities-motivations and policy implications," Tech. rep., Technical report.

Wickham, H. (2009): *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.

Wickham, H. (2015): *stringr: Simple, Consistent Wrappers for Common String Operations*, r package version 1.0.0.

Wickham, H. (2016): *scales: Scale Functions for Visualization*, r package version 0.4.0.

Wikipedia (2015): "Programming languages used in most popular websites - Wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites, (Visited on 2015.10.06).