Open Source Project Collapse – Sources and Patterns of Failure

Daniel Ehls
Hamburg University of Technology
daniel.ehls@tuhh.de

Abstract

Why do open source projects fail? Open source projects have gained tremendous momentum, in theory, managerial practice and global economy. However, a large number of projects are now dormant, collapsed, or abandoned. Even celebrated success stories lose developers and fail. Yet, failure is underexplored and our understanding of developer departure is limited. Previous literature has concentrated on prospering projects, attracting contributors, and expanding communities, but it is unclear why even well-integrated members leave and projects fail. This study explores open source project failure by drawing on ten in-depth open source software case studies and netnographic analyses. We identify antecedents of developer departure, discover patterns of project collapse, and reveal where members move. We complement the dominant research logic of how to facilitate membership on-boarding with the aspect of understanding de-boarding. Our results enhance our understanding of why and how open projects fail and involve implications for open organizations.

1. Introduction

"Around the same time we started [the open source project] we knew people at another Swedish company that worked on a great web server. And it was licensed under the GPL! Since this was before the Apache web server dominated the field, you would have thought that they had a glorious future. Well, that future did not work out, since they failed in some basic principles."

The open source (OS) literature knows a number of iconic and highly prospering communities such as Linux or Wikipedia. Projects that have managed to attract a large number of contributors and create a product are usually referred to as successful projects [3, 33, 39]. However, numerous further OS projects are neither crowded, nor generate outputs. Besides success stories, failure is a substantial case [5, 36]. While 1.63% of all hosted projects at SourceForge.com reach the maturity phase, a total of 63% fail [19]. Moreover, failure is not limited to emerging communities and growing projects, but even 'incumbent' communities collapse and fail. Frequently, participants terminate their contribution a few months after joining in, OS projects lacking returns and various OS communities are abandoned [19, 30]. Thus, even established projects that overcame start-up challenges such as license choice [5], agreed on governance structures [26], attracted members [3, 29] and fulfilled success criteria [6, 21, 34] might still fail. Indeed, 50% of registered project members halt their contribution [10], with most contributors quitting within one year [31], and 80% of open source software projects disappearing [5] even with their objective unachieved.

So far, our understanding of failure cases has been limited, and the causes of and patterns for the collapse of open source projects have been missing. For instance, while previous research has focused on explaining member incoming and growth, such as attracting participants and leading them towards the center [12], our understanding of membership departure and collapse is limited. Earlier research recommends a socialization process for sustained member participation; yet, how can we explain integrated core members with strong ties leaving an ongoing, not completed, project? Why do participants no longer contribute, although they have previously done so? What spurs them to change their minds? Additionally, with 80% of OS projects disappearing [5], where do these significant numbers of developers go after their project termination?

Moreover, an emerging stream of membership turnover [3, 12, 24, 28] has demonstrated the substantial consequences of participants' incoming and going such as cognitive dissonance, but the antecedents of participation termination and community collapse are still unclear. This paper aims to explain this puzzling phenomenon. Particularly, we

URI: http://hdl.handle.net/10125/41807
ISBN: 978-0-9981331-0-2

explore three questions: what drives away community members of a working project, where do they go, and what are the patterns of community collapse?

In order to explore this poorly-understood phenomenon, we employ an inductive qualitative grounded approach, draw on multiple intentionally-sampled case studies, and adopt several data sources. Our empirical fields are ten heterogeneous, but firm, sponsored open source projects chosen according to theoretical sampling. Data collection follows a netnographic research approach, reviewing the archival project database, and retrospective data such as existing interviews with project core members.

Our study reveals underpinning factors and trajectories for membership withdrawal and contributes to our understanding of how open projects collapse. We shift the focus away from the dominant research logic on how to facilitate membership on-boarding to the aspect of understanding de-boarding. Instead of looking for supporting mechanism to grow and expand projects, we concentrate on existing projects and lessons how not to run them. In doing so, we complement existing participation and success research by discovering damaging strategies that cause a loss of developers and project failure. The next section provides a more detailed research background, followed by the research strategy employed, and findings. We conclude with implications for theory and executives.

2. Research Background

Open source (OS) initiatives are today a central element of value creation and firm's business strategy [2]. Firms gain access to scarce knowledge, as well as creativity and ideas [20, 25]. The features of OS communities make them distinctive organizational forms [2, 20, 22] which "fundamentally change the approaches and economics" of production [14] and offer "eye-opening examples of novel innovation practices" [36].

Previous research has strongly focused on stimulating contributions. Eliciting external input of participants and increasing the size of the community is a central tenet for prospering projects [8, 35]. Earlier studies provided great insights into participation motives and attraction of members [37]. Later studies explored how to stimulate repeated or long-term contributions [10, 38] and how members progress towards the community center [7, 20, 26]. While research on participation has thus contributed tremendously to our understanding of participant incoming and project growth, results regarding project collapse are missing, as prior research is limited on the building up of communities. Furthermore, literature highlights the motivation to contribute; however, why do contributors lose motivation and quit the project?

An emergent stream of literature has also started to concentrate on membership turnover [3, 12, 24, 28]. This line of research has so far revealed the consequences of membership turnover for projects. Empirical evidence highlights the effects of gaining or losing knowledge sources on production, stimulating herding behavior, introducing cognitive dissonance, or effects on product performance. Although we have thus gained insight into the effects of turnover, research remains inconclusive regarding revealing rationales underpinning membership turnover.

Moreover, OS success literature [6, 21, 34] has identified a range of success factors. However, even this literature calls for additional measures to explain success in more detail. It is also questionable as to whether failure is the opposite of success, and if there is such an assumed dichotomy. Can success factors really explain failure? Indeed, analyzing success cases introduces a success bias and susceptible samples [8, 9]. Failure and success projects might have pursued different strategies of which we are not aware. However, these harmful 'invisible' strategies can also be present in successful projects, but mediated and potentially mitigated by other factors. These hidden factors not only pose a latent threat to projects that are still running, but understanding them also expands theoretical discussions.

In summary, several important questions are unanswered, while research faces controversial observations regarding individual member participation and organizational failure. In other words, although prior research recommends the socialization of members for sustained participation, we observe that even socialized and core members still depart. Why should they leave their 'friends' and abandon an influential community position? Moreover, enabling community leadership structures and governance fosters project prosperity. Nevertheless, how can we explain the collapse of mature communities that not only successfully attracted members and created social bonds, but also established a working basis of authority and decision rights?

Avoiding the collapse of innovation systems is a central issue for policy-makers and executives. Understanding failure pattern facilitates organizational learning [4], reduces sample bias towards successful cases [8], and responds to the call for further research on theorizing membership turnover [12]. In line with these rationales, introduced controversies, observations and open questions, this study explores the failure of OS projects. As explained

above, we particularly pursue three questions: what turns project members away, where they go, and what the patterns of community collapse are. According to previous literature, we focus on developers as being essential for OS projects [8, 29, 34, 35], and define failure cases as projects that are neither formally canceled by project owners nor finished, but have lost a large number of formerly active developers while gaining fewer new ones.

In exploring participant departure and project failure, we take a different look at the dominant participation research logic. Instead of concentrating on managing the member supply side and leading members towards the community center, we concentrate on the member drop-out side and seek to explain what leads developers away from the project, causing community collapse. The participation research focus is extended from encouraging participation and member on-boarding to also understanding the patterns of driving away participants and de-boarding. We thus follow the important rationale to elicit volunteer contribution to firm's problem solving tactics [35] and drawing on a critical number of participants [29, 34], but we acknowledge that contributions can be elicited by attracting new joiners, and already existing members in the community. Particularly, if already attracted members leave, the total contribution base decreases and the project size shrinks, eventually down to the point of community collapse.

3. Research Strategy and Sample Selection

Our research question aims to search for unknown antecedents for developer departure, create an initial model for understanding failure and elucidate an underexplored issue. Accordingly, we follow the literature reference and apply a qualitative grounded approach [11, 13, 23]. This approach is most useful in providing insight into unchartered terrain [23], particularly to identify novel conceptual categories for a poorly-understood phenomenon and find broad possible explanations in the absence of a strong theory [11, 13]. Another crucial research aspect to consider is the heterogeneous project situation. Contrary to empirical evidence that the institutional setting of the community already strongly affects contribution and self-selection behavior [31, 32, 33], prior research has frequently failed to consider the broader project context [36] and interactions between individual and organization level factors [8]. We respond to this call for "messy holistic methods" in fluid organizations [12] and do not rely on a single project analysis. Instead, we pursue a multiple case study analysis [17] that is suitable for exploring central constructs within their context, and increase variance in data [13, 39].

In order to establish our sample, we purposefully selected specific OS projects for analysis, as recommended for inductive qualitative research [17]. We purposefully screened communities, forums and blogs for developer comments pointing to failed projects, according our definition of failure. We triangulated our failure criteria with data. For instance, to validate the projects that are neither formally canceled nor finished, we searched for project status statements by project owners or administrators. Additionally, we looked for projects that were formerly active (measure: create a product) and mature (measure: at least 2 years of being productive). Our identified projects managed to overcome start-up challenges, established working structures, and elicited important member contribution for collaborative production [3, 26], which are important success criteria [6, 21, 34]. In analyzing these special failure cases, we aim to identify clear antecedents causing the change from a flourishing project to failure. Finally, archived project's documents still need to be available for data collection (measure: access to key community traffic given). None of the authors has been involved in any of the selected communities or knew any of the participants beforehand. Table 1 shows the selected cases.

Case	Project Type	Peak
Xara Xtreme	Graphics	2005-2007
Roxen	WebServer	1996-2009
Mambo	CMS	2002-2007
Sodipodi	Graphics	1999-2004
XFree86	X-Windows	1992-2004
Meego	Mobil Op. S.	2010-2011
OpenDarwin	Op. S.	2002-2006
OpenOffice	Office	2000-2011
MySQL	Database	1995-2010
Compiere	ERP	2001-2006

Table 1: Sampled Cases

3.1. Data Collection

Our primary data source for each case is archived project's documents, such as stored forum discussions, e-mail lists, and FAQ pages. These information sources are the central communication tools. They reflect participant behavior, interactions and controversies within the community, as well as key actions taken in the project. A common key concern in

data collection is self-reported data and social desirability bias. To mitigate this challenge our collected data came from retrospective and real projects [27]. For each case, we studied the available documents from at least four months before failure and several weeks afterwards, summing up to several hundred documents per case. Notably, most of the cases still have their websites and documents publicly accessible. Another key general concern in data collection is limited contextual understanding and common method bias. We therefore triangulated our data with supplementary documentation from third-party websites. Drawing on supplementary external information sources increases the perspectives covered and provides greater depth [39]. Main information sites were information hubs, such as open source news platforms, independent opinion pages, including blog articles, as well as media coverage, such as interviews conducted with project participants or leaders. Linking this secondary material to our primary data helped us to compare project information to outside views and account for the project environment.

3.2. Data Analysis

Our data analysis follows the practices of netnographic content coding and analytic induction [13, 18]. We first screened the publicly available project documents for indications of community collapse and extracted original participants' statements. These initial statements were collected and tagged with provisional codes. They were supplemented with 'outsider' statements tagged as well. Thereafter, the provisional coded testimonials were compared, and if applicable combined, resulting in conceptual categories and a more solid description of failure for a specific case. Our analysis proceeded with discussing and aligning the categories across cases and deriving general sources of failures. We further grouped these sources of failure into broader classes. Our analysis elicited several distinct sources of failure, and we differentiated them into two broader classes. In order to safeguard the soundness and robustness of our categories, two independent researchers reviewed our coding scheme and supported our categorization.

4. Findings

This section shows the discovered sources of OS projects' failure and patterns of failure, and reveals the new 'home project' of departed developers.

4.1. Failure Sources

4.1.1. Product Functionality

The functionality of the generated product is a core challenge and indicator of high product quality. For OS projects, it is the software product per se, the delivered product as a black box. Users can observe the features of the product as well as its malfunctioning without inspecting or understanding the source code. This category is relevant not only to developers, who can modify the product and are strongly involved in the community, but also to ordinary users who just use, but do not further develop, the product. Critical elements within product functionality are technical issues, missing features, product compatibility and usability, bugs and malfunctioning. Frequently, users accept a certain amount of limited product functionality, particularly in good faith and hope for future updates and improvements. However, we noticed that, in collapsed communities, the product functionality declined for a longer time, with no action taken to improve the product. Even worse, project managers refused to implement certain features, leaving users and developers frustrated.

"No, but I do miss features which have been availible for a long time in XaraXtreme's Windows version..... Don't get me wrong, I really enjoy using XaraXt<reme and don't think there's an equal program availible for Linux. But I don't like using abandoned programs with missing features (e.g. proper SVG im/export, flash export or the possibility to use the great LiveEffects and other plugins)."

4.1.2. Production Challenges

A core characteristic of open source projects is the collaborative nature of the projects and working as a community towards a goal. The organization, processes and practices describe how the product is developed. Our study discovered that organizing production and developer collaboration is a major reason for project collapse. For developers, the production aspect is important from two perspectives. First, production determines the final product. Thus, confidence in the production can mitigate product flaws as well as short release cycles signal activity of the product and community. Secondly, interaction with other community members is important for collaboration. Challenges in working together slow down production and even upset contributors. Exemplary issues are missing development tracking,

limited development resources, slow release schedule, and lacking support.

"What make MiaCMS different from Mambo? By design we have less formal policies and procedures, fewer teams and team leaders, and have refocused on a core open source principle… release early, release often".

A major aspect within production is the difficulty to align the development direction. Project members feel uncertain about plans of the product and are concerned about the project future. To develop the product together, it is important that developers work in the same direction and their work is aligned along a roadmap. Particularly challenging is a change of the development direction. Developers signed up under the assumption of working towards specific objectives. However, if it later transpires that community management heads towards another direction, developers often go towards other projects.

"The group is a bunch of Roxen users that are unhappy with the direction that the Roxen development is heading. We decided that we could create a server that would better serve our needs than Roxen 2.x...."

4.1.3. Respect OS Ideology and Community The collaborative and open nature is an idiosyncratic trait of OS projects. However, community management and the hosting firm frequently fail to pay due attention to them. Real interest in the project is missing as well as a community focus. Sometimes, the community is 'used' for commercial purposes without being given sufficient benefits. A project thus can provide sufficient code access and the right license, and may even be a bolt corporate name as a sponsor; however, the community is interested in a greater goal.

"Apple failed to build a community around Darwin in the seven years since its original release because it was not a corporate direction, but rather a marketing stunt. Culturally, Apple did not and does not understand what it means to be open source or to build a working community."

4.1.4 Partnerships Eliciting a certain number of developers is a core issue for growing a project. However, it is not only the number of developers and pure market share, but also the right developers and connection to an ecosystem of partnerships. Particularly for mature projects, growing out of the amateur niche and gaining traction among further professional partners is vital. Clearly, having a vital project and attractive product might not be sufficient. Our cases point to a lack of cooperation with other projects and not being supported by commercial partners. Projects are still alive without these partners. However, to enter a larger ecosystem, building interfaces to other projects and gaining a market penetration is required to survive beyond a project scale level, particularly as soon as global competition grows. An essential partnership also represents an NGO, providing a safe harbor for developers and hosting the project.

"Another thing to note is that XFree86 has dramatically less commercial support than just about any "cornerstone" Open Source project. Maybe that's because of our "meritocracy" and focus on individual contributors."

4.1.5 Change of Leadership and Governance Structure Coordinating development efforts and decision processes are important aspects in steering the project. Vivid projects seem to have developed a working leadership structure and a governance system. However, we discovered that, not the current leadership structure, under which developers have signed up for the project, that represents the problem, but changes to the leadership team do. Hosting partners try to alter the leadership team without contacting the community directly, or introduce sudden changes to the project organization. Particularly after 'acquisitions' of an open source project, a new leadership team should take over. However, the community loses its voice, and in turn, feels hijacked.

"The Mambo Foundation was formed without regard to the concerns of the core development teams. We, the community, have no voice in its government or the future direction of Mambo. The Mambo Steering Committee made up of development team and Miro representatives authorized incorporation of the Foundation and should form the first Board. Miro CEO Peter Lamont has taken it upon himself to incorporate the Foundation and appoint the Board without consulting the two development team representatives, Andrew Eddie and Brian Teeman."

4.1.6. Change of Business Strategy The nature of open source projects, including free revealing of its outcomes, is a constant challenge for managers. Offering complementing services or products is a well-functioning strategy for commercialization. However, reflecting and aligning the business strategy might be necessary. Particularly for established and

running communities, the shifting of its business model toward a classical proprietary model is attemptable, as it could allow a direct revenue stream. We found evidence that some of our sampled projects try the shift and force open source participants into commercial products. However, community volunteers reacted strongly to this announcement, and as a result, frequently left the project. Communities overstepped the balance, and lost a major share of their members.

"Compiere certainly did not fail due to its technology. It failed due to lack of sales and marketing expertise, execution and the wrong bet to "upgrade" open source minded partners and customers to a traditional, commercial model. I think that the Commercial Open Source model is still valid, but Compiere overstepped the balance between proprietary and open product components".

An important aspect within the business model represents openness traits. Access to code and license properties are two essential criteria for open source software developers. The two criteria determine the use and modification of the code, and thus the fit with individual needs and employing the code for further purposes. In contrast to the high importance of the criteria, we found evidence that projects try to change given properties. Accordingly, they aim to change two core features of the projects on which volunteers have themselves self-selected into the project. However, changing the base for project selection causes members to rethink their contribution, and stimulate voting with feet and forking.

"we've decided to delay making the source code fully public until we've progressed the port a little further and the program is more functional on Linux".

4.1.7. Trust Issues A final reason for project collapse is trust. Community members at one point realize they cannot trust the project anymore. Thus, community members put trust in the project, but lost it for certain reasons. Broken promises, mistrust in corporations or false advertising are exemplary issues.

"Now, a rational, cynical human being would say corporations around the world say anything to make a buck, and you can't trust them, so why did you believe these open-source claims? Very fair. Some of us aren't so bright and were lured in by the promises that turned out to be false advertising."

4.2 Patterns of OS Failure

While the aspects above are separate sources of failure, they rarely appear isolated. In fact, we notice that certain failure sources add to each other and even build on each other. Community members are loyal to their community's respective product, and try to influence the project to enable a collaboration. However, we find that the leadership team frequently does not listen to complaints and disagrees with the community.

"Oracle's official response to the announcement of The Document Foundation was clear – Oracle will continue OpenOffice.org as usual"

Consequently, a failure cycle starts. The community members' level of confidence and need fulfillment declines, while their dissatisfaction grows. Members react increasingly sensitively to further aspects, and, in parallel, production slows down and further failure sources emerge, leading to the final departure of participants.

"I am developer and when I first heard about opensourcing of Xara, I was really happy. I want to help you with such an excellent product (which is missing in Free Software world, Inkscape doesn't fit my needs), but then I realized that it is not fully Free Software, that there are still some proprietary parts. My willingness for help completely fell off in that exact moment, but I have got still hope that it will be completely opensourced sometime in future (yes, it means including CDraw). I would never invest my time in some partially closed source software. After reading latest discussion on mailing list, my hopes are gone. You clearly doesn't understand what is important for Free Software community. "Free as in beer" is _nothing_ for us, we want "free as in speech" software. Completely free, without closed proprietary parts. I know many good developers in Free Software community and nobody would even consider helping Xara because the fact that some parts of Xara are closed. It doesn't matter that it is only one part, Xara is simply "tainted". And your attitude clearly shown in latest discussion on mailing list proves that you really don't understand Free Software community and development model. This is why Inkscape get all attention and help from community, while you get nearly nothing. It is simple "opensource or die" principle. […] I am really disappointed that such a great software (which Xara clearly is) is sentenced to failure in Free Software community simply because you don't understand principles on which Free Software community stands. […] My interest in Xara

is now gone, I am going to look at Inkscape again, maybe they will be able tu fulfill my needs in the end and maybe I will be possible to help him too."

We further identified two longitudinal patterns of project collapse. Members react with different sensitivity, contingent on the failure source. In particular, we discovered two basic categories of failure (see Table 2): slowly emerging and suddenly erupting. Slowly emergent failure sources are issues that have existed for a long time within the community and grow over time. Frequently, other failure sources reinforce each other and at some point, they overstep a certain threshold. Exemplary sources within this category are production challenges, product functionality, community ideology, and partnerships. In contrast, suddenly erupting failure sources represent issues that become visible within a short period of time and have a huge impact on the community. Examples for erupting failure sources are changes of important aspects, such as business strategy, license or code access characteristics, and realization of missing trust.

Failure categories	
Slowly emerging	Suddenly erupting
• Product features	• Business strategy
• Production challenges	• License and code access restrictions
• Community Ideology	• Trust
• Partnerships	

Table 2: Failure sources

4.3 Departure Destinations

It is so far unknown where developers head, after quitting their engagement. Our study reveals three categories. Members head towards a forked project, competing projects or to a sponsoring partner.
Heading towards a fork represents an option, if the project structure is stable and the code and production are functioning well. As such, sudden changes such as those in the leadership team or openness properties might cause a fork. Members can rely on their previous contributed donations and continue to use their familiar product, as if nothing has happened.

"We want a change to give the community as well as the software it develops the opportunity to evolve. For this reason, from now on we will support The Document Foundation and will—as a team—develop and promote LibreOffice. We hope that many are going to join us on this path."

Another home for developers are competing projects. Developers look to fulfill their needs. Accordingly, they search for similar projects and frequently find a replacement in projects in the same scope. Interestingly, developers even accept inferior technical capabilities, as they are confident about their own development competencies.

"The end result of that thinking was that the community around the product shrank to a few very happy users/customers. The at that time technically inferior Apache web server got all the users and developers. So nowadays hardly anybody knows about the Swedish web server "Spinner/Roxen", but everyone has heard about the Apache web server."

The third place to which developers migrate, which we discovered, is the sponsoring or hosting corporation itself. Hosting projects know core contributors and welcome them as knowledgeable employees. However, this might crowd out and drain the volunteer developer base even further.

"Apple ended up hiring more than half of its most active contributors, which amounted to roughly three or four people. This drained the contributor pool significantly, and effectively muffled the ones that got hired,...."

5. Discussion

Failure of open source projects is widely observed, but our understanding is limited. While young projects face the challenge to attract sufficient participants and set up the organizational structure for community interaction, which was explored in previous research, mature communities have already increased and apparently managed collaboration. Yet, even mature communities fail. Despite earlier research to understand participation motives and the consequences of membership turnover, prior insights are lacking the identification of what causes the collapse of a project. Our study provides early insight into the underexplored phenomenon of community failure. We focus our knowledge in three core areas: (1) revealing antecedents that drive the likelihood of project failure, (2) introducing two different patterns of project collapse, and (3) revealing departure destinations of developers of collapsed communities. More generally, we (I) reinforce the call to watch out for biased samples and highlight failure studies, (II) contribute to the understanding of social practices, and (III) inform the discussion of 'open organizations'.

(1) We disclosed several sources for failure and antecedents of community collapse. Our findings indicate a strong association between the identified failure sources and projects becoming inactive. Indeed, the identified failure sources might exhibit an explanation why participation motivations decline and even socialized members leave. They inform the discussion on OS member turnover, and link it to decreasing motivation. It also calls to attention OS project success research, and the inclusion of 'destroying' factors dampening the prosperity of projects - despite the prevalence of established success factors.

(2) We discovered that volunteer developers raise their voice before deciding to leave the community, but frequently management refuses to listen. Several issues might build up, increasing frustration and finally triggering the departure. Therefore, we discovered two patterns differentiated by the speed at which developers decide to leave and are contingent on the specific failure source. Some failure antecedents drive developers more quickly away than others. Accordingly, we classify failure sources in two broader failure pattern categories leading to project collapse: sudden changes and emerging issues. Emerging issues are present in the community over a longer period and slowly grow until developers leave. Sudden changes bring disrupting impacts for the community and initiate a rapid departure decision by developers. Thus, developers are more sensitive to some failure antecedents, while they tolerate others for a longer time. Some failure sources might even represent hygiene factors. These insights detail the failure discussion. Not all 'faults' are similar, which explains why some OS projects die over a longer period, while other communities collapse rapidly.

(3) We studied the migration destinations of developers after leaving. Developers move to forks, project-hosting firms and competing projects. As such, contributors remain active in the "development" world, but dedicate their efforts to other projects. In doing so, they socialize with new projects, making a return to previous projects harder. Losing members thus hurts failing projects twice. First, the failing community loses its labor. Second, they indirectly empower competing projects because they free up for them the capacity of the departing developers. Developers go to rival projects and contribute to their prosperity. Switching developers therefore means a loss of development power for the failing project as well as a boost or contribution to new projects. As shown in the case of Roxen, developers even turn towards currently inferior projects, for the sake of development freedom and future. The loss of labor not only weakened Roxen, but also, with the developers turning to the new project and increasing it tremendously, drove it out of the market. Revealing developer destinations together with antecedents for departure thus not only increases our understanding of failure, but also offers a more nuanced view of an end-to-end participation lifecycle of participants and community [15].

From a broader point of view, our contribution is threefold. First, (I) from a methodology point of view, our insights inform future research and address a biased sample in previous research. While prior research has advanced our knowledge tremendously, it has frequently concentrated on successful projects. As such, earlier studies claim limitations regarding their sample and their findings. Our study is one of the first to explore and include failed cases, thereby not only enabling a better understanding, but also opening an avenue for stronger generalizations and a more complete picture of the open source ecology.

Second (II), our findings contribute to the social practice view of OS communities [37] and social bonds [7]. Participation and institutional settings are strongly interrelated. Motivation to contribute to software development does not change per se. Yet, motivation to contribute to a specific project changes. Thus, measuring participation to one specific project might not reveal the full picture. One might assume that participation has decreased because a developer does no longer contribute to the project. However, our study has found that participation rationales still exist, but developers focus their contribution efforts towards new endeavors. The institutional setting strongly influences participation rationales and labor donation. Our study has found further evidence that developers group together and head towards new projects collectively. Leaving a community does not mean leaving a social bond. Social ties are still existent, and are frequently transferred to new projects.

(III) Finally, our study informs the discussion beyond OS projects, particularly the conversation of 'open organizations' [2, 7, 22]. Our findings point to two critical challenges for (mature) open organizations so far little understood: solving conflicts and introducing major changes. Major changes might be required to adapt to a changing environment or a new business direction. Also, the acquisition of a project or external influence, such as surrounding ecosystem decisions, is a driver for rapid change. So far, little is known as to how to manage it. Within our cases, major challenges are communicated top-down, without consultation of the community, and without ensuring an acceptable way forward for community members. Consequently, heavy conflicts emerge. Conflicts also grow as a result of identified emergent challenges. While smaller issues do not immediately trigger

departure of project members, they build up and increase conflict. The conflicts observed in our samples are not limited to individual developers, but rather affect the entire community. While individual level conflicts might cause single members departing, the observed failure sources prompt larger groups of developers to leave. Interestingly, observed projects rarely collapse because of the conflict per se, but because of failure to manage the conflict, unite core project stakeholders, and find suitable solutions. Community management reacts with ignorance, instead of interacting with community. Project members complain about flaws, and even provide counterproposals and mitigation strategies. However, community management or the hosting firm frequently reacts with silence, thus undermining the community's confidence in the future. Our study thus not only provides evidence for adjusting a governance model [16], but also supports the standing proposition that member retention is strongly affected by project conflicts [3]. While successful projects might have managed to implement formal and informal settlement processes [3, 20, 26], absence of conflict solving and resolving structures has a fatal impact on open organizations with fluid membership and stimulates a 'leaving script', resulting in community collapse.

6. Outlook

Our study has certain limitations. We used a non-random sample to follow the recommendation for a theoretical case sample. Additionally, the analyzed projects are all firm-hosted projects that are within the scope of research, have gained popularity, and offer insight into firm-community relationships. Future research should use larger and more diverse samples, include success cases, and build on complementing research approaches. Such potential future studies could help determine the most critical failure issues, as well as validate associations between identified factors and context. Second, our failure definition centers on an unintended huge developer loss. Thus, even cases that support millions of websites in the world, such as MySQL which is often referred to as a success case [21], might appear in a new light. Notably, our definition does not require a complete termination of the project. Some projects might even revive after a major loss of developers, for example with new structures – and some do. Certainly, further failure definitions are promising, such as complete bankruptcy or disappointing adoption. The insight of future research and our present study would inform managers. Our study represents a lesson-learnt document on how a community collapses. It highlights critical issues in running an open organization, interacting with the community, and collaborating with unequal partners. Managers should avoid identified pitfalls, otherwise they could risk driving participants away and destroying the community.

10. References

[1] Ars Technica: http://arstechnica.com/tech-policy/2016/01/editors-demand-ouster-of-wikimedia-board-member-involved-in-no-poach-deal/ accessed 10 of May 2016

[2] Baldwin, C., & von Hippel, E. (2011). Modeling a paradigm shift: From producer innovation to user and open collaborative innovation. Organization Science,22(6), 1399-1417.

[3] Butler, B. S. (2001). Membership size, communication activity, and sustainability: A resource-based model of online social structures. Information systems research, 12(4), 346-362.

[4] Cannon MD, Edmondson AC (2005) Failing to learn and learning to fail (intelligently): how great organizations put failure to work to innovate and improve. Long Range Plan 38(3):299–319

[5] Colazo, J., & Fang, Y. (2009). Impact of license choice on open source software development activity. Journal of the American Society for Information Science and Technology, 60(5), 997-1011.

[6] Crowston, K., Annabi, H., Howison, J. (2003). "Defining open source software project success". International conference on information Systems.

[7] Dahlander, L., & O'Mahony, S. (2011). Progressing to the center: Coordinating project work. Organization science, 22(4), 961-979.

[8] Dahlander, L., & Piezunka, H. (2014). Open to suggestions: How organizations elicit suggestions through proactive and reactive attention. Research Policy,43(5), 812-827.

[9] Denrell, J. (2003). Vicarious learning, undersampling of failure, and the myths of management. Organization Science, 14(3), 227-243.

[10] Ducheneaut, N. (2005). Socialization in an open source software community: A socio-technical analysis. Computer Supported Cooperative Work (CSCW),14(4), 323-368.

[11] Edmondson, A. C., & McManus, S. E. (2007). Methodological fit in management field research. Academy of management review, 32(4), 1246-1264.

[12] Faraj, S., Jarvenpaa, S. L., & Majchrzak, A. (2011). Knowledge collaboration in online communities. Organization science, 22(5), 1224-1239.

[13] Glaser, B.G., Strauss AL. 1967. Discovery of Grounded Theory. Strategies for Qualitative Research. Mill Valley, CA: Sociology Press.

[14] Hars, A., & Ou, S. (2002). Working for free? Motivations for participating in open-source projects. International Journal of Electronic Commerce, 6(3), 25–39

[15] Herstatt, C., & Ehls, D. (2015). Open Source Innovation: Phenomenon, Participant Behaviour, Impact (Vol. 37). Routledge.
[16] Kane, G. C., Johnson, J., & Majchrzak, A. (2014). Emergent life cycle: The tension between knowledge change and knowledge retention in open online coproduction communities. Management Science, 60(12), 3026-3048.
[17] Eisenhardt, K. M.: Theory Building from Multiple Cases. In Primer: Qualitative Research in Strategic Management, Strategic Management Journal [originally posted September 2014; updated October 2014]
[18] Kozinets, R. V. (2002). The field behind the screen: Using netnography for marketing research in online communities. Journal of marketing research, 39(1), 61-72.
[19] Krishnamurthy, S. (2002). Cave or community?: An empirical examination of 100 mature open source projects. First Monday.
[20] Lee, G. K., & Cole, R. E. (2003). From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. Organization science, 14(6), 633-649.
[21] Lee, S, T., Kim, H., Gupta, S. (2009). Measuring open source software success. Omega, 37, Pp.426-438
[22] Levine, S. S., & Prietula, M. J. (2013). Open collaboration for innovation: principles and performance. Organization Science.
[23] Anteby, M. Lifshitz, H., Tushman, M (2014): Using Qualitative Research for "How" Questions. In: Primer: Qualitative Research in Strategic Management, Strategic Management Journal [originally posted September 2014; updated October 2014]
[24] Oh, W., & Jeon, S. (2007). Membership herding and network stability in the open source community: The Ising perspective. Management science, 53(7), 1086-1101.
[25] O'Mahony, S., & Bechky, B. A. (2008). Boundary organizations: Enabling collaboration among unexpected allies. Administrative Science Quarterly,53(3), 422-459.
[26] O'Mahony, S., & Ferraro, F. (2007). The emergence of governance in an open source community. Academy of Management Journal, 50(5), 1079-1106.
[27] Podsakoff, P. M., & Organ, D. W. (1986). Self-reports in organizational research: Problems and prospects. Journal of management, 12(4), 531-544.
[28] Ransbotham, S., & Kane, G. C. (2011). Membership turnover and collaboration success in online communities: Explaining rises and falls from grace in Wikipedia. MIS Quarterly-Management Information Systems, 35(3), 613.
[29] Ren Y, Harper FM, Drenner S, Terveen L, Kiesler S, Riedl J, Kraut RE (2012): Building member attachment in online communities: Applying theories of group identity and interpersonal bonds. MIS Quart. 36(3):841–864.
[30] Schweik, C. M., & English, R. (2007). Tragedy of the FOSS commons? Investigating the institutional designs of free/libre and open source software projects. First Monday, 12(2).
[31] Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. Management Science, 52(7), 1000-1014.
[32] Stewart, K. J., & Gosain, S. (2006). The impact of ideology on effectiveness in open source software development teams. Mis Quarterly, 291-314.
[33] Stewart, K., & Ammeter, T. (2002). An exploratory study of factors influencing the level of vitality and popularity of open source projects. ICIS 2002 Proceedings, 88.
[34] Subramaniam, C., Sen, R., & Nelson, M. L. (2009). Determinants of open source software project success: A longitudinal study. Decision Support Systems, 46(2), 576-585.
[35] Terwiesch, C., & Xu, Y. (2008). Innovation contests, open innovation, and multiagent problem solving. Management science, 54(9), 1529-1543.
[36] Von Krogh, G., & Von Hippel, E. (2006). The promise of research on open source software. Management science, 52(7), 975-983.
[37] Von Krogh, G., Haefliger, S., Spaeth, S., & Wallin, M. W. (2012). Carrots and rainbows: Motivation and social practice in open source software development.MIS quarterly, 36(2), 649-676.
[38] Wu, C. G., Gerlach, J. H., & Young, C. E. (2007). An empirical analysis of open source software developers' motivations and continuance intentions.Information & Management, 44(3), 253-262.
[39] Yin, R. (1994). Case study research: Design and methods . Beverly Hills.