Open Source Is A-Changin'

How Qualitative Research Can Help Us Adapt

Daniel Klug[a] and Heather Miller[b]

a University of Basel, Switzerland

b Northeastern University, USA, and EPFL, Switzerland

Abstract In the past five years, industry has overwhelmingly turned to open source as a primary means to build products and services atop of, with around 80% of surveyed companies acknowledging that they run on open source. 66% of companies say they would first consider open source options before proprietary ones. A decade ago, this wasn't the case at all; these numbers were flipped, with a tiny minority of companies preferring open source solutions.

This has changed the dynamics of open source community markedly. In the early days of the open source movement, most people using a project were also contributing back to it in some way. However, the number of users versus contributors has changed by likely uncountable orders of magnitude. With unicorn startups being built on top of this open source shared infrastructure, the incentives and motives behind open source contributors building this valuable infrastructure has also begun to shift.

Meanwhile, software engineering studies focused on open source have been overwhelmingly quantitative; quantitative analyses on codebases, issue trackers, surveys, etc.

In this essay, we argue that with these changing tides comes dynamics that we can't easily quantify. Why then do rely predominantly on quantitative methods when we attempt to understand the dynamics of open source communities? We lay out a number of research questions and qualitative techniques from the social sciences that can help us better understand these trends, and how to adapt to them going forward.

ACM CCS 2012

- *General and reference → Computing standards, RFCs and guidelines;*
- **Applied computing → Publishing;**

Keywords open source software, community, sociology

The Art, Science, and Engineering of Programming

Perspective The Art of Programming

Area of Submission Open Source Community, Social Coding

1 Open Source Has Eaten The World

In the past five to ten years, an enormous shift in industry's attitude towards open source software[1] has taken place. This trend has been tracked year-to-year by the open source security and license compliance firm, Black Duck Software. Black Duck runs an annual survey which asks typically over 1,000 companies about their open source use. In 2010, Black Duck noted that 42% of respondents said their companies run part or all of its operations on open source software [44]. By 2015, 78% of the over 1,300 companies surveyed said that they "run on open source"—a nearly 2x increase in the number of companies saying they build their software products atop open source software [43]. In the same 2015 edition of the survey, a whopping 66% or approximately two-thirds of respondents say that their companies first reach for open source solutions before even considering proprietary ones!

This is a far cry from reality in the early 2000s. Mark Suster, an entrepreneur and venture capitalist provides a glimpse of what it was like to build software products at the turn of the millennium,

> *"When I built my first company starting in 1999 it cost $2.5 million in infrastructure just to get started and another $2.5 million in team costs to code, launch, manage, market and sell our software."* [47]

At that time, infrastructure upon which to build a software product was proprietary and expensive. Infamous players such as Microsoft fought to keep this status quo, noting the threat of open source in leaked memos,

> *"Open source software poses a direct, short-term revenue and platform threat to Microsoft, particularly in the server space. Additionally, the intrinsic parallelism and free idea exchange in open source software has benefits that are not replicable with our current licensing model and therefore present a long term [sic] developer mindshare threat."* [52]

In an effort to counter the feared rise of open source, Microsoft famously adopted an "embrace, extend, extinguish" strategy intent on thwarting the momentum of the open source movement by publicizing usability issues in open source software, and embracing, extending, and extinguishing open protocols [51]. For the better part of a decade, Microsoft was in an open war with open source, and the future of software seemed to be headed in a decidedly different direction.

However, Microsoft's vision for open source never came to pass. Fast-forward again to 2015 where a vast majority of companies build atop open source software and avoid even considering proprietary software in their stacks. Open source has clearly won, despite countless trials and tribulations over the past decade and a half. And now, the modus operandi of many software startups is to build value by building on what can be reused. For example, Mike Krieger, the co-founder of Instagram, in a blog article offering advice to other startup founders recommends to embrace open source,

[1] Throughout this essay, we use the term "open source" but do so with the intention of indiscriminately referring to both *free/libre and open source software* (FLOSS).

"Borrow instead of building whenever possible. There are hundreds of fantastic open-source projects that have been built through the hard experience of creating and scaling companies; especially around infrastructure and monitoring … that can save you time and let you focus on actually building out your product." [33]

Krieger's view is also in line with that of Nadia Eghbal's experience as a software developer in the foreword of her report on the unseen labor behind open source [23], namely that building software products these days is in large part pulling together and building a patchwork of different pieces of open source software,

"I saw the enormous amounts of money being poured into software companies. But as an amateur software developer, I knew that I had never done any of it alone. I used free and publicly available code which I cobbled together and offered up for personal or commercial purposes. Really, the people behind those projects, whoever they were, had done most of the work." [23]

1.1 The Manpower Behind Open Source

Meanwhile, as we continue to promote the merits of reusing existing open source solutions, and as we continue to indiscriminately build products from open source components, we largely miss the unfortunate reality behind some of these open source projects.

A well-known example is OpenSSL, a toolkit for Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols, and a general-purpose cryptography library. In 2014, OpenSSL was used on 66% of all web servers worldwide to secure connections on the internet. In April of 2014, a catastrophic security bug dubbed "Heartbleed" was discovered in OpenSSL's implementation of TLS that affected around 17% of the world's secure web servers at the time of disclosure.

Prior to Heartbleed, OpenSSL was maintained by only a handful of volunteers, despite the fact that much of the global internet and the world's largest companies depended heavily on the security provided by OpenSSL. Steve Marquess, a US Department of Defense security consultant noticed another contributor, Stephen Henson, worked full time on OpenSSL and learned after years of working with him on OpenSSL that Henson earned only one-fifth of Marquess's salary [23]. Nadia Eghbal described the situation on the OpenSSL team as,

"Until that moment, [Marquess] had 'always assumed, (as had the rest of the world) that the OpenSSL team was large, active, and well resourced.' In reality, OpenSSL wasn't even able to support one person's work." [23]

While this is just one anecdotal story, it's likely that there exist many other important open source projects that businesses are built upon, and which suffer from similar or worse manpower and sustainability woes.

Although not a reliable metric for identifying projects with sustainability woes, efforts on estimating the Truck Factor [2, 25] of a given software project could provide a useful metric for identifying projects with manpower woes. The Truck Factor, or Bus Factor as it is sometimes known, is the minimum number of team members that have to suddenly disappear from a project before a project is incapacitated. Tools to calculate the Truck Factor of a code base based on its commit history have been developed [29], and offer a powerful and sometimes frightening glimpse into the health of some of the world's most widely-used open source projects.

While preliminary work, Avelino et. al [3] have applied their Truck Factor calculation to the 133 most popular GitHub projects across several programming languages, and have found that 64% of the top projects on GitHub rely on only 1 or 2 developers to survive—a jarring figure if true. Projects such as Sass, the popular CSS preprocessor, and D3, the popular JavaScript library for visualizing data, are both calculated to have a Truck Factor of 1—meaning that there is only a single developer that guarantees the project's survival. Projects such as Pandas, a widely-used Python data analysis library, the Clojure programming language, and Apache Cassandra, the popular NoSQL database, are all calculated to have a Truck Factor of 2. The remaining 36% of popular GitHub projects analyzed have Truck Factors spread from 3 to 250. The full table of results can be found in Avelino et. al's prelininary study [3]. If accurate, these results imply that manpower shortages for popular open source projects not only exist, but that they may be commonplace!

To make matters worse, reports of contributor distress abound amid the shifting dynamics and rapid uptake of open source by industry. In her report, Eghbal quotes Noah Kantrowitz, a member of the Python Software Foundation, who describes this shift,

> *"In the early days of the open source movement there were relatively few projects and in general most people using a project were also contributing back to it in some way. Both of these have changed by likely uncountable orders of magnitude."*
>
> *"The other problem is the growing imbalance between producers and consumers. In the past, these were roughly in balance. Everyone put time and effort in to the Commons and everyone reaped the benefits. These days, very few people put in that effort and the vast majority simply benefit from those that do. This imbalance has become so ingrained that for a company to re-pay (in either time or money) even a small fraction of the value they derive from the Commons is almost unthinkable."* [23]

Meanwhile, contributor distress and even burnout seem to be on the rise, often at the hands of successful companies that have benefitted from open source. Another quote from Eghbal's report, from Daniel Roy Greenfeld, a Python and Django developer captures this sentiment,

> *"I personally get regular demands for unpaid work (Discussions about payment for work always stall) by healthy high profit companies large and small for [my projects]. If I don't respond in a timely fashion, if I'm not willing to accept a crappy pull request, I/we get labeled a jerk. There is nothing like having core Python/PyPA maintainers working for Redhat [sic] demanding unpaid work while criticizing what they consider your project's shortcomings to ruin your day and diminish your belief in open source."* [23]

Many more often more colorful instances of contributor burnout, rage-quitting, and other types of contributor attrition continue [1, 7, 15, 50].

In the early days of the open source movement, most people using a project were also contributing back to it in some way. However, the number of users versus contributors has changed by likely uncountable orders of magnitude. With unicorn startups being built on top of this open source shared infrastructure, the incentives and motives behind open source contributors building this valuable infrastructure has also begun to shift.

With this shift in the dynamics of open source communities, how are contributors' attitudes changing? In this essay, we argue that with these changing tides comes dynamics that we can't easily quantify. Why then do rely predominantly on quantitative methods when we attempt to understand the dynamics of open source communities?

2 How Have Open Source Communities Been Studied Thus Far?

For almost two decades, open source communities have been the subject of studies on people's motivations or on community structures. These studies generally used quantitative approaches because the goal was to provide universal understanding about this world-spanning, decentralized, and ever growing phenomenon which has people freely working together. Here, quantitative studies want to be—and are—able to generalize findings to make assumptions about a large and dynamic community. Open source communities as well provide easy access to relevant social and work related data that can be mined. Therefore, it seems reasonable to turn to quantitative methods for collecting and analyzing data and to test hypotheses.

Recently, qualitative approaches have begun to be used to analyze characteristics of open source communities. As we will further outline in the following section, it becomes clear that there is a growing need for qualitative methods to look at more detailed aspects of the social dynamics of open source communities.

2.1 Quantitative Approaches

Looking at the ongoing changes in the open source software communities, quantitative research provides generalized insights on intrinsic and extrinsic factors for contributing [34]. Job and career opportunities [35], technical and market success, or licensing and consumer base [38] are equally important motivations as enjoyment, interest, or obligation and community [36] for people to contribute to open source projects. While hybrid forms of extrinsic and intrinsic motivation seem to be most common, psychological factors and as well social characteristics of work design, workplace, and socio-demographic variables and values play important roles regarding social recognition and identification in open source contributing [37].

These findings are based on quantitative approaches, such as surveys, which allow to collect and analyze data about people's motivation to engage and contribute in open source software development [21]. Quantitative studies also look at aspects of labor [28] and problem-solving, project-related behavior, or modes of strategic communication to generate variables for analyzing open source communities in relation to patterns of participation, membership or (self-)regulation [27].

Quantitative network analysis illustrates factors beyond motivation. Self-development and altruism [41], and reputation and positive evaluation are additional factors for open source software contribution, they largely base on preferential attachment, accumulated advantage, homophily, and shared affiliations [32].

In a broad perspective, quantitative research demonstrates, that 'community' is one major quality for open source software contribution and engagement. Learning from peers, creating valuable artifacts, and foremost altruism and personal devotion to create better software environments while at the same time selflessly helping a community are key aspects among paid and unpaid contributors alike [4].

Quantitative studies exemplify community-based interaction and structures as general key elements in the creation and the progress of open source software groups. But they as well hint that many detailed aspects on individual levels define people's decisions in joining, contributing to, and maintaining open source software projects. Given the ongoing changes in software industry and its relation with open source software development, as well as vice versa, individual aspects seem to play an increasing role. This calls for new in-depth research on the changes in motivations, benefits, goals etc. in open source communities from a primarily qualitative perspective.

2.2 Qualitative Approaches

In general, only few approaches have been made so far to analyze open source communities using qualitative methods, though some studies notice a growing need for it [45]. Similar to the quantitative research, early qualitative studies identify comparable motivations and benefits of contributing [9]. However, qualitative analyses are able to connect the characteristics of open source communities to basic social concepts, such as the forms of capital by French sociologist Pierre Bourdieu (1986).

2.2.1 Social Capital

Bourdieu defines the potential of an individual as a combination of different proportions of economic, social, cultural, and symbolic capital. Each form of capital is generated through social interaction and can be accumulated, institutionalized, or even materialized and as well turned into one of the other forms. Bourdieu defines social capital as "the aggregate of the actual or potential resources which are linked to possession of a durable network of more or less institutionalized relationships of mutual acquaintance and recognition – or in other words, to membership in a group – which provides each of its members with the backing of the collectivity-owned capital, a 'credential' which entitles them to credit, in the various senses of the word" [10]. Therefore, social capital includes all current and potential resources of an individual that are linked to participating in social relationships and networks. In this way, social capital refers to the relationship of individuals within a group and not to the individual itself, it defines and enables membership in a community [36]. This is an important aspect when looking at online communities because individuals who are only loosely connected in a network, or not at all (so-called 'weak ties'), need to gather higher social capital in order to participate in networks.

Looking at open source software communities, studies find that, while coding quality is important on a technical and work level, social capital in terms of reputation based on peer feedback and social behavior within open source communites is as well crucial [14]. Socialization and sociality are key factors in building, shaping, and maintaining open source communities as well as for stepping up within open source communities. This connects to further theories of social capital, for example Robert Putnam (2000), who shows that social capital and social interaction are indicators for ways individuals organize themselves in societal structures and how they maintain communities through shared activities [42]. In interviews with open source contributors, studies find strong approval of 'social contributions' and social capital, such as community-orientated behavior, versus professional contributions, like programming or coding, which illustrates that social skills or social capital is not considered to be 'work [16].

In terms of the changing dynamics between open source software communities and industry, the first research question would be:

Research Question 1

In which way do contributors and developers in open source software communities still value social capital, given the context of increasing industry exploitation and interference?

Another relevant aspect within open source software communities, is that, for example, social capital can be achieved through close social relations and solid infrastructures but as well through loose networks with structural holes [40].

2.2.2 Structural Holes

The social theory of structural holes can be another relevant approach to look at the ongoing changes in open source software communities. It derives from American sociologist Ronald Burt [12] who provides explanations for differences in social capital through network analysis. Structural holes are based on a concept of competition, people or parties with complementary resources or information are divided by a 'structural hole' that is filled by a third person or party who connects the other two and therefore creates advantages. In this way, the theory of structural holes deals with the missing of relationships within a network and defines several networks that have no or only little overlaps while each network itself has very strong ties. Usually, structural holes are filled through individuals with weak ties, which means that a high social capital is needed. Whoever fills structural holes, individuals or groups, gains higher visibility and information advantage within networks [13].

Contributors to and developers of open source software are able to fill structural holes between open source community networks and industry networks. This entails numerous aspects, such as connecting unpaid labor to commercial and profit-seeking companies. From an organizational perspective, some studies already use a social network analysis approach to look at strategies of companies in collaborating in open source software projects [49].

Given the dynamics and changes between open source software communities and industry, a second research question would be:

Research Question 2

Under which circumstances are open source software contributors and developers still willing to fill structural holes between open source software communities and industry?

Both research questions imply posing longer-term or self-biographical questions to open source contributors and their relationship to open source software development and communities. Further, this also implies to look beyond only the working environment an open source contributor finds themselves within. Rather, it should also include questions exploring biographical dimensions, social dimensions, and political attitudes that touch upon the foundational philosophies of open source and free software culture.

For example, in their study, Carillo, Huff & Chawner as well propose a "more global and confirmatory approach addressing an individuals' entire socialization experience" [16]. They argue that qualitative research on open source software communities and their members/contributors should not simply rely on data available from within the community, such as emails or forums, but should try to capture people's biographical and professional history, attitudes, decisions etc. to find out about crucial socialization aspects concerning open source activities. In qualitative interviews with open source contributors, Carillo, Huff & Chawner ask about "different instances of citizenship behavior in the FOSS community context" and about "key factors that characterize the socialization experience of newcomers in FOSS communities" [16].

Such an approach could be instructive when applied to better understanding changes in attitudes and motivations related to contributors' desire to gather social capital as well as to fill structural holes.

3 A New Approach

We argue for a new qualitative approach focused on analyzing organizational, professional and social changes in and as well towards open source communities which could as well be enhanced with a mixed-method design. Many studies call for qualitative research as it is so far underrepresented in analyzing open source software development. As previously shown, quantitative studies mainly illustrate reputational, professional, or social dimensions in contributors' motivations to work in open source. They also provide information to outline profiles of contributors to get ideas on the research area of open source. Data mining and big data sets enable analyses and interpretations of general aspects in organizational research on open source communities, their team structures and network structures [17, 53]. This leads to generalizable results to evaluate organizational structures and peer contexts in open source communities. Qualitative research may use existing quantitative data of recent studies or can be based on quantitative results to draft relevant questions. Standardized methods in quantitative research, based on the principle of intersubjectivity, help to measure and generate phenomena of empirical observation based on hypotheses. In our proposed research, qualitative methods are far more suitable because we do not seek to generalize hypothetical assumptions but to evaluate in-depth understanding of social and sociocultural aspects in the changing dynamics of open source communities.

To answer the proposed research questions, an approach based on Grounded Theory [18, 31, 46] should be applied. Grounded Theory is a methodology from the social sciences in which a social theory is systematically constructed out of collected data. For example, to develop a theory about social interaction requires interviewing people or observing social situations. Grounded Theory methodology allows one to inductively frame research questions, as well gathered from quantitative studies, and to collect, transcribe, analyze, and interpret data using a variety of qualitative methods. In Grounded Theory, collecting, extracting, and analyzing data follows a circular process based on a special type of sampling called theoretical sampling. Theoretical sampling allows data to be collected as long as it provides new insights to the research question, and the size of a sample can be adapted to first results [5, 30, 39].

Therefore, we suggest conducting qualitative interviews [6, 11, 20, 26, 48] with open source software developers and contributors paired with content analysis of accessible written documentation in online forums and user groups etc. Qualitative interviews seek to describe and understand individual meanings in life worlds of the interviewees and to get behind subjective stories and experiences [20, 22, 26]. In this way, qualitative research can provide a more detailed and personal perspective than, for example, quantitative surveys. Qualitative interviews as well enable the researcher to adjust to the live interview situation and to ask specific follow-up questions based on the progress of the interview and the information obtained from the interviewee [6, 48]. Interviewees function as experts of their professional and social community [8, 26]. They can provide detailed insights into the structures of a research area. Therefore, interviews should either be semi-structured or narrative to ensure open questions for individual subjects, ideas, and structuring by the interviewee [6]. However, general conversation guiding and stimulating questions should be prepared, also to ensure easier coding in the following analysis and interpretation of the interview data. Semi-structured or narrative interviews are also always recorded and transcribed.

The analysis of interview data in Grounded Theory then follows three basic steps. First, identifying codes in the interview transcripts, that means, marking key aspects and tagging them with new or already generated codes. Second, combining codes and connecting them to formulate concepts. Third, comparing concepts and relating them to each other to formulate a theory. For coding interviews data analysis software, such as MAXQDA or Dedoose, can be used. These steps as well involve writing memos that may be integrated at any step while going through codes and concepts. The constant comparison of concepts includes extracting negative cases that do not fit to refine the theory [19].

Interviewees can, for example, be recruited by using existing contacts in academia as well as via online forums [26]. Regarding our research questions, it appears to be significant to, on the one hand, interview people who are newer to the open source software community and, on the other hand, people who are seniors and have contributed and developed a decent number of open source software projects. This is to broaden the potential field of qualitative data given the assumptions of ongoing changes in open source software development.

Further questions could ask about personal attitudes towards social and cultural concepts in open source communities, career factors regarding personal involvement in open source development, or personal experiences with industry and open source contributions.

4 In Closing

Open source is changing. However, the motivation and benefits that we can see from quantitative studies for people in the past are unlikely to change in the future. People will most likely continue or start to devote their time and skills to open source projects in favor of community-based learning, sharing, helping, advancing etc. or because of job prospects. This is the foundation of the open source philosophy. In contrast, what is changing is that industry is increasingly exploiting the good will of people to build products that only a small minority may potentially benefit from while at the same time industry is gaining more and more commercial profit.

Consequentially, it seems these trends are challenging the ideals and foundations of open source philosophy and hence people's mindset on joining or staying in open source projects. To find out whether open source contributors are aware of these changes, and to find out about their personal inner processes of weighing and deciding of how to contribute or being part of a community, one can't rely on quantitative surveys or data.

Rather, as proposed, qualitative methods can provide in-depth insights, detailed information, and individual perspectives on the dynamics of open source. We know about the apparent changes on the industry side, but we do not know entirely about potential changes in decision-making, attitudes, inner processes etc. on the contributors' side as they are less visible within the generally "unseen labor" [24] in open source software.

Qualitative interviews are the most useful way to collect this sort of information and data from individuals. Grounded Theory is the most sensible way of structuring, describing, analyzing, and interpreting this kind of interview data to eventually come up with a social theory that can provide general validity about changing attitudes in and towards open source software.

References

[1] Matt Asay. *Why open source developers are burning out: No respect*. Sept. 2017. URL: https://www.techrepublic.com/article/why-open-source-developers-are-burning-out-no-respect/.

[2] Guilherme Avelino, Leonardo Passos, Andre Hora, and Marco Tulio Valente. “A Novel Approach for Estimating Truck Factors”. In: *Proceedings of the 24th International Conference on Program Comprehension (ICPC)* (Apr. 2016).

[3] Guilherme Avelino, Marco Tulio Valente, and Andre Hora. *What is the Truck Factor of popular GitHub applications? A first assessment*. Technical report. PeerJ PrePrints, 2017.

[4] Hoda Baytiyeh and Jay Pfaffman. “Open source software: A community of altruists”. In: *Comput. Human Behav.* 26.6 (Nov. 2010), pages 1345–1354.

[5] Pedro F Bendassolli. “Theory building in qualitative research: Reconsidering the problem of induction”. In: *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*. Volume 14. 1. 2013.

[6] Bruce L Berg. “Methods for the social sciences”. In: *Qualitative Research Methods for the Social Sciences. Boston: Pearson Education* (2004).

[7] Ryan Bigg. *Open source work*. Nov. 2015. URL: https://ryanbigg.com/2015/11/open-source-work.

[8] Alexander Bogner, Beate Littig, and Wolfgang Menz. “Introduction: Expert interviews—An introduction to a new methodological debate”. In: *Interviewing experts*. Springer, 2009, pages 1–13.

[9] Andrea Bonaccorsi and Cristina Rossi. “Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business”. In: *Knowledge, Technology & Policy* 18.4 (Dec. 2006), pages 40–64.

[10] Pierre Bourdieu. “The forms of capital (English version)”. In: *Handbook of theory and research for the sociology of education* (1986), pages 241–258.

[11] Svend Brinkmann. “Interview”. In: *Encyclopedia of critical psychology*. Springer, 2014, pages 1008–1010.

[12] Ronald S Burt. *Structural Holes*. Harvard University Press, 1995.

[13] Ronald S Burt. “Structural holes versus network closure as social capital”. In: *Social capital*. Routledge, 2017, pages 31–56.

[14] Yuanfeng Cai and Dan Zhu. “Reputation in an open source software community: Antecedents and impacts”. In: *Decis. Support Syst.* 91 (Nov. 2016), pages 103–112.

[15] Brett Cannon. *Why I took October off from OSS volunteering*. Oct. 2016. URL: https://snarky.ca/why-i-took-october-off-from-oss-volunteering/.

[16] Kevin Carillo, Sid Huff, and Brenda Chawner. “What makes a good contributor? Understanding contributor behavior within large Free/Open Source Software projects–A socialization perspective”. In: *The Journal of Strategic Information Systems* 26.4 (2017), pages 322–359.

[17] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. "Developer Onboarding in GitHub: The Role of Prior Social Links and Language Experience". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. New York, NY, USA: ACM, 2015, pages 817–828.

[18] Kathy Charmaz. "Constructing grounded theory: A practical guide through qualitative research". In: *SagePublications Ltd, London* (2006).

[19] Kathy Charmaz. "Grounded Theory: Objectivist and Constructivist Methods". In: *Handbook of qualitative research*. Jan. 2000.

[20] Kathy Charmaz and Liska Belgrave. "Qualitative interviewing and grounded theory analysis". In: *The SAGE handbook of interview research: The complexity of the craft* 2 (2012), pages 347–365.

[21] Namjoo Choi and Joseph A Pruett. "The characteristics and motivations of library open source software developers: An empirical study". In: *Libr. Inf. Sci. Res.* 37.2 (Apr. 2015), pages 109–117.

[22] Rosalind Edwards and Janet Holland. *What is Qualitative Interviewing?* en. A&C Black, Nov. 2013.

[23] Nadia Eghbal. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*. Ford Foundation, 2016.

[24] Nadia Eghbal. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*. Ford Foundation, 2016.

[25] Mivian Ferreira, Marco Tulio Valente, and Kecia Ferreira. "A comparison of three algorithms for computing truck factors". In: *Proceedings of the 25th International Conference on Program Comprehension (ICPC)* (2017).

[26] Uwe Flick. *An introduction to qualitative research*. Sage, 2014.

[27] Nicolai J Foss, Lars Frederiksen, and Francesco Rullani. "Problem-formulation and problem-solving in self-organized communities: How modes of communication shape project behaviors in the free open-source software community". In: *Strategic Manage. J.* 37.13 (2016), pages 2589–2610.

[28] Giampaolo Garzarelli and Riccardo Fontanella. "Open source software production, spontaneous input, and organizational learning". In: *Am. J. Econ. Sociol.* 70.4 (2011), pages 928–950.

[29] *GitTrends*. http://gittrends.io/.

[30] Barney G Glaser and Judith Holton. "Remodeling Grounded Theory". In: *Historical Social Research / Historische Sozialforschung. Supplement* 19 (2007), pages 47–68.

[31] Barney G Glaser and Anselm L Strauss. *Discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.

[32] Daning Hu, J Leon Zhao, and Jiesi Cheng. "Reputation management in an open source developer social network: An empirical study on determinants of positive evaluations". In: *Decis. Support Syst.* 53.3 (June 2012), pages 526–533.

[33] Mike Krieger. *Understanding Changes in the Software & Venture Capital Industries*. Apr. 2013. URL: http://opbeat.com/blog/posts/picking-tech-for-your-startup/.

[34] Sandeep Krishnamurthy. "On the intrinsic and extrinsic motivation of free/libre/open source (FLOSS) developers". In: *Knowledge, Technology & Policy* 18.4 (Dec. 2006), pages 17–39.

[35] Karim R Lakhani and Robert G Wolf. "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects". In: (Sept. 2003).

[36] Yan Li, Chuan-Hoo Tan, and Hock-Hai Teo. "Leadership characteristics and developers' motivation in open source software development". In: *Information & Management* 49.5 (2012), pages 257–267.

[37] Patrick Mair, Eva Hofmann, Kathrin Gruber, Reinhold Hatzinger, Achim Zeileis, and Kurt Hornik. "Motivation, values, and work design as drivers of participation in the R open source project for statistical computing". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 112.48 (Dec. 2015), pages 14788–14792.

[38] Vishal Midha and Prashant Palvia. "Factors affecting the success of Open Source Software". In: *J. Syst. Softw.* 85.4 (Apr. 2012), pages 895–905.

[39] Janice M Morse. "Sampling in grounded theory". In: *The Sage handbook of grounded theory* (2010), pages 229–244.

[40] Chitu Okoli and Wonseok Oh. "Investigating recognition-based performance in an open content community: A social capital perspective". In: *Information & Management* 44.3 (Apr. 2007), pages 240–252.

[41] Shaul Oreg and Oded Nov. "Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values". In: *Comput. Human Behav.* 24.5 (Sept. 2008), pages 2055–2073.

[42] Robert D Putnam. *Bowling Alone: The Collapse and Revival of American Community*. en. Simon and Schuster, Aug. 2001.

[43] Black Duck Software, editor. *2015 Future of Open Source Survey Results*. Apr. 2015. URL: https://www.slideshare.net/blackducksoftware/2015-future-of-open-source-survey-results.

[44] Black Duck Software, editor. *78% of Companies Run on Open Source Yet Lack Formal Policies*. Apr. 2015. URL: https://www.blackducksoftware.com/about/news-events/releases/companies-lack-open-source-policies.

[45] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. "A systematic literature review on the barriers faced by newcomers to open source software projects". In: *Information and Software Technology* 59 (Mar. 2015), pages 67–85.

[46] Anselm Strauss and Juliet Corbin. *Basics of qualitative research: Procedures and techniques for developing grounded theory*. 1998.

[47] Mark Suster. *Understanding Changes in the Software & Venture Capital Industries*. June 2011. URL: https://bothsidesofthetable.com/understanding-changes-in-the-software-venture-capital-industries-b69a7e3a1ec7.

[48] Steven J Taylor, Robert Bogdan, and Marjorie DeVault. *Introduction to qualitative research methods: A guidebook and resource*. John Wiley & Sons, 2015.

[49] Jose Teixeira, Gregorio Robles, and Jesús M González-Barahona. "Lessons learned from applying social network analysis on an industrial Free/Libre/Open Source Software ecosystem". en. In: *Journal of Internet Services and Applications* 6.1 (July 2015), page 14.

[50] *The reason people burn out on open source*. https://news.ycombinator.com/item?id=8712370.

[51] Christopher Tozzi and Jonathan Zittrain. *For Fun and Profit: A History of the Free and Open Source Software Revolution*. en. MIT Press, Aug. 2017.

[52] Vinod Valloppillil. *Halloween Document 1: Open Source software—a (new?) development methodology. Microsoft internal strategy memorandum leaked to Eric S. Raymond*. 1999.

[53] Bogdan Vasilescu, Alexander Serebrenik, Mathieu Goeminne, and Tom Mens. "On the variation and specialisation of workload—A case study of the Gnome ecosystem community". en. In: *Empir. Softw. Eng.* 19.4 (Aug. 2014), pages 955–1008.

About the authors

Daniel Klug is an Assistent in the Seminar for Media Studies in Department of Art, Media, and Philosophy at the University of Basel, Switzerland. daniel.klug@unibas.ch

Heather Miller is an Assitant Clinical Professor of Computer Science at Northeastern University, in Boston, and the Executive Director of the Scala Center, an open source foundation focused on supporting the Scala programming language at EPFL in Lausanne, Switzerland. heather@ccs.neu.edu