Is There Collaboration in Open Collaboration? The Role of Producers and Corporate Users in Open Source Software Development

Sabine Brunswicker
Discovery Park District,
Purdue Polytechnic Institute, Purdue University, U.S.

Stefan Haefliger
House of Innovation, Stockholm School of Economics, Sweden
Bayes Business School, City St.George's University of London, UK

Abstract

Innovation in open collaboration projects involves producers and users, both private and corporate, each engaging in and pairing problems with solutions. While the innovation literature has focused predominantly on motivations to contribute, we face a paucity of insights into how incentives shape the producers' and users' choice to design and contribute solutions for problems and needs expressed by others. We explore, in the case of Open Source software development, if and how different actor types and task complexity can be linked to such collaborative problem-solving, where one developer solves another's problem instead of working independently. Patterns of problem-solution pairing in a large Open Source Software development project identify corporate users as ranking first in attracting solutions to their problems. We find that collaboration occurs systematically and actors' perceived incentives are shaped inside and outside the open collaboration project, namely via options to sell complementary services to corporate users. In order to pair solutions with problems of corporate users, significant effort in understanding their needs is required from innovators, extending the viability of the open collaboration model into the domain of producer innovation that regularly incurs communication and coordination costs. Our findings advance theory on open collaboration as a unique form of organizing innovation. We discuss implications and insights for management and policy.

Keywords: open collaboration, open source community, selective revealing, problem-solving, incentives

1. INTRODUCTION

Communities of production like Open Source software (OSS) communities have become an important model for designing high quality and innovative products as a public good openly accessible for everybody (West, 2003; Bonaccorsi et al., 2003; Lerner and Tirole, 2002, von Hippel, 1976, 1986, 2006). Recent innovation economics has provided evidence that communities of production can follow the model of open collaboration, an extension of the original user innovation model (Baldwin & von Hippel, 2011, von Hippel, 1976, 1986, 2006): In open collaboration communities, not only users benefit from using the innovations but also producers allocate their resources to the design efforts inside the communities because they seek direct but also indirect benefits - such as the option to sell complementary services (Henkel, 2006; Dahlander and Wallin, 2006; Teece, 1986). Open collaboration works when such benefits outweigh the design and communication costs (Baldwin & von Hippel, 2011) and the innovation literature has come to see open collaboration models working in parallel and independently of producer-led innovation models (von Hippel, 2007). Yet, we argue that our understanding of the different incentives facing various actors is incomplete because we lack empirical observations of actual collaboration - defined as the creation of design artifacts for solving each other's tasks in open collaboration.

The growing literature on OSS communities accounts for the presence of both sponsored "producer" developers as well as independent "user" developers and highlighted the differences in incentives among these two types of developers (Stewart et al., 2006; Germonprez et al., 2016). One recent insight is that open collaboration works because developers can innovate independently on their own problems by simply overlaying code in the existing code base (Howison and Crowston, 2014; Baldwin and Clark, 2006). They benefit the most if they can work on solutions to problems in line with their individual needs without much effort to coordinate with others during the design process. In fact, costly search and collaboration effort can be reduced or avoided by layering motivationally independent work and deferring a solution (Howison and Crowston, 2014). Work on design motifs shows patterns in how developers engage with the existing technology to innovate when benefits outweigh costs (Brunswicker & Mukherjee, 2023). While such studies provide important insights into how incentives shape the nature of collaboration in open collaboration, they treat communication costs and indirect benefits as affecting all developers equally. We do not know if and how incentives for collaboration differ across developers and we lack a deeper understanding when and how problems and solutions are paired if and when both producers and users are present, each guided by possibly distinct incentives. This need becomes even more pertinent, due to an additional gap in the literature: Existing work does not sufficiently distinguish between developers that work for producer firms and those working for user firms - both are treated as employees or "sponsored" developers (Stewart et al., 2006). Specifically, a contemporary OSS community is composed of at least the following three types of actors - producer developers, corporate user developers, and private users developers - who seek very particular direct and indirect benefits, e.g. the option to sell complementary services that can be exercised "outside" the community. In the case of OpenStack, producer developers include HP and Cisco (Subramanian, 2018; Sotiriadis et al., 2016), corporate user developers include Purdue University and CERN (Toor et al., 2012; Trader, 2014). and private user developers include students and individuals (e.g. Price, 2016).

And it is exactly those differences in perceived benefits that may shape their choice to forgo a focus on one's own tasks and shift attention to problems and tasks of other developers despite the higher coordination costs associated with collaboration. If there is no reason for producers and users to shift attention to problems of other actors, it remains unclear how problems are being solved in collaboration at all (Levin and Prietula, 2014; Gächter et al. 2010). Hence, we currently face a paucity of predictions that affects the relative viability of open collaboration projects (Baldwin and von Hippel, 2011). The question that remains to be answered is: *How do producers and users, corporate and individual users, solve each others' problems in open collaboration communities when they do?*

To answer this question, we extend Baldwin and von Hippel's (2011) model of open collaboration and consider all three actors - producers, corporate users, and individual users, who participate in the design processes in OSS communities. For each actor, we identify distinct direct and indirect benefits - in the sense of option values that can be exercised at a later stage, as well as perceived costs, associated with different forms of collaborative problem-solving. To identify these incentives, we draw upon the rich literature on incentives in OSS communities. Not only the problem, its scope and complexity (Howison and Crowston, 2014), but also the actor who revealed it may shape the perceived direct and indirect benefits and costs. If perceived benefits are misaligned, this may jeopardize collaboration and, in the extreme, lead to design processes without any collaboration across actors.

We apply an exploratory research design for the following reasons: while the routines of open collaboration in OSS development are well known (Raymond, 1999; Lakhani and von Hippel, 2003), the scope and patterns of collaboration are not, and we lack empirical observation and scope in collaborations. Actor types have been identified, and corporate actors are known to operate and contribute to open collaboration (Dahlander, 2007), yet we find ourselves too early in the debate to generate solid hypotheses about patterns of collaboration across actor types. Our analytical process, outlined below, remains sufficiently open to uncover general tendencies without rushing to conclusions (Edmondson and McManus, 2007) and we work with a large set of data from a core repository within

OpenStack where 298 actors published and completed 7,470 tasks or problems, each task requiring on average 300.27 lines of code to solve.

First, extending the structural predictions for problem-solution pairing emerging from the design literature (Baldwin and Clark, 2006; Brunswicker and Mukherjee, 2023, Howison and Crowston, 2014), our findings suggest that each actor's perceived incentives for collaboration are shaped not only by the scope of the problem but also the actor type who revealed the problem. This suggests that the incentives created via open collaboration, an organizational form sitting in between user innovation and collective action, and producer innovation, are constituted and mediated by actor-specific economic relationships. It calls for a more fine-grained theorizing about open collaboration.

Second, we find that users and producers create solutions for other producers and users with the likely goal to reap indirect benefits of selling complementary services from the public good. While this is an obvious finding from a producer-oriented model of innovation, it emphasizes that contributions to the public are linked to temporally delayed benefits and option values that are nurtured strategically through the "pairing" of complementary knowledge that will translate into complementary assets that can be economically exploited. As a consequence the cost-benefit calculus and incentives to collaborate change.

Third, and maybe most consequentially, we find that corporate users play a unique role in consistently ranking first in attracting solutions to their problems from both producers and private users. This pattern changes the default innovation model for open collaboration in that the perceived incentives for collaboration with corporate users outweigh the costs of understanding their needs and problems – something Baldwin and von Hippel (2011) sum up under communication costs. It appears that open collaboration encroaches on the domain of producer innovation and may remain viable despite high design and communication costs.

2. THEORY

In the following section, we first briefly review the existing literature related to open collaborative models innovation in OSS communities, with its roots in the influential work on user innovation. This stream of research - with a long tradition - helps us understand why the different actors - producers, corporate and private uses, contribute to open collaboration in the case of OSS development, guided by the assumption of generalized exchange as driving benefit for participation. All actors benefit from the public good created collectively, irrespective of whether these actors actually collaborate inside the community by solving each others' problems. Subsequently, we turn to the discussion on whether and how developers in their role as representing one of the three actors contribute to collaborative problem-solving, defined as a design effort in which one actor contributes designs in the form of source code to another actor's expressed design problem (e.g. a new feature request).

2.1. Actors in open collaborative OSS and their incentives for participation in collaborative problem-solving

The literature on open collaboration and, more specifically, Open Source software development has, since the beginning of academic interest in the phenomenon, studied the incentives to contribute to the public good (Lerner and Tirole, 2002; von Hippel and von Krogh, 2003; von Krogh et al., 2012b). The varying motivations and economic incentives to contribute to a public good include helping behavior and reciprocity (Lakhani and von Hippel, 2003) yet also the case where developers receive a salary and contribute on their employers' behalf (Roberts et al., 2006) leading to a complex patchwork of dynamics including individual motivation and corporate strategies (von Krogh et al., 2012a; Alexy et al., 2013). At the same time, the composition of actors within open collaboration has largely been assumed implicitly as consisting of paid and volunteer developers (Stewart et al., 2006; Roberts et al., 2006).

The definition of actors present usually covers users and producers, defined originally by Eric von Hippel (1982), and knowledge revealing as a strategy in innovation has first been introduced by von Hippel over forty years ago (von Hippel, 1976; 1978; 1986) when he introduced the theory of user innovation describing how and explaining why users purposively share innovations with fellow users or the public, first in specific industries and later more widely online. The main argument is that users derive benefits from their own use of the innovation but do not mind other users free riding or improving based on their work. Their use benefits outweigh the costs of losing control over their knowledge. Producer innovators are classically defined, by contrast, as innovators who derive benefit from selling the innovation (von Hippel, 1986; 2005).

Implicitly or explicitly, both literature streams on users and on producers assume that there is some form of an open collaboration project in which problems and solutions are revealed and “paired” in order to produce innovation artifacts (e.g. software packages, services, etc.). While the original user innovation theory assumes that such open collaboration projects are dominated by users, Baldwin and von Hippel (2011) distinguish open collaboration projects from user innovation projects, stating that open collaboration brings together both different producers as well as users but lacks clear arguments about the interactions between multiple competing producers and users inside an open collaboration project via collaborative problem-solving as we define it above.

2.2. Costs and benefits of problem-solution pairing in open collaboration

Understanding different actors' motivations for joining OSS communities doesn't fully explain how they actually engage with others' problems and needs, beyond just building on their own solutions (Howison and Crowston, 2014). Some participants may simply advocate open collaboration without contributing to the design process. Only limited research, notably by Baldwin and von Hippel (2011), has explored how problem-solution matching occurs within OSS projects, suggesting that these pairings are guided by participants' assessment of benefits and costs (von Hippel and von Krogh, 2016).

With respect to *costs*, Baldwin and von Hippel (2011)'s work suggests that both actors are guided by the costs of design and communication occurring inside the community. If design costs are high, the theory argues that producers are more likely to engage in open collaboration projects since the ability to share the costs of development with others creates immediate benefits (West and Gallagher, 2006). However, this does not tell us much about whether they actually interact with other producers and users inside the open collaboration project in the sense that they work on problems revealed by other producers or users or just their own. Indeed, whether this might happen may not be a question of the overall level of the design costs but the level of modularity of the technology under development (Baldwin and Clark, 2000; Simon, 1968; Parnas, 1972). If the technology under development, or the knowledge shared openly, is sufficiently modular then opportunities to work on other actors' problems become easily recognizable (Baldwin and Clark, 2006). At the same time, modularity facilitates the division of labor in the sense that actors can work on their own tasks without having to interact with others (Baldwin and von Hippel, 2011). So it remains unclear if, under conditions of high design costs and high modularity, actors actually work on each other's problems.

Communication costs significantly influence how developers evaluate cost-benefit trade-offs in open collaboration projects. While online platforms enable near-zero cost sharing of complex information and design documents, making it easier to match needs with solutions (Boudreau and Lakhani, 2015), the situation is more nuanced. Low communication costs can encourage knowledge sharing and reuse between project members, as demonstrated by how visible artifacts lead to increased reuse and recombination (Boudreau and Lakhani, 2015; Stanko, 2016). However, this argument doesn't fully address the distinction between problem-related artifacts (like feature requests) and solution artifacts (like source code), particularly when different types of actors interact. The communication costs between corporate users and OSS producers may be higher than between similar actors due to knowledge translation needs at their boundaries.

In terms of *benefits*, Baldwin and Clark (2006) emphasize the role of future value derived from the design created in OSS communities (Baldwin and Clark, 2006, p.1117). However, the future value of a design is uncertain, and thus, there is only an “option value”. In their model they do not focus on open collaborative OSS communities that involve producers and users but instead assume that all developers are individual users that benefit from the free use of successfully realized designs, allowing them to exercise the option. Modeling a game of involuntary altruism, they show that whether developers work on a task together depends on the option value. The option value is influenced by code modularity and developers' discount rates. In highly modular systems, developers can work independently while still benefiting from module complementarity, preventing free-riding behavior. This aligns with theories of generalized exchange among user innovators (Gosh 2005), where independent work contributes to a collective public good. However, when both users and producers are involved, the collaboration dynamics become more complex.

Baldwin and von Hippel (2011) describe this as a hybrid innovation model, where producers offer complementary products and services that extend the open, modular innovations from OSS communities. While producers benefit from complementary assets (Teece, 1986) created outside the open collaboration setting, such as installation and upgrade services, their interaction patterns with users vary. Corporate users typically need these complementary services, while private users ("tinkerers") often prefer to handle technical challenges independently. This suggests producers are more likely to engage with corporate users due to the potential for product-solution pairs that enable proprietary service offerings. However, there remains a gap in both theoretical and empirical research regarding how producers' external benefits relate to joint problem-solving with corporate users.

Complementary to such an economic view towards future value and indirect benefits, the literature on communities as fabrics of social exchange offers an alternative view to theorize about benefits. Some of the earliest forms of online open collaboration projects include the Free Software development projects from the 1970 and 1980 onward (see Moody, 2002). Starting with a pioneer of the Free Software movement, Richard Stallman explained his and others' motivations to collaborate over the creation of software as rooted in mutual assistance and reciprocity (Stallman, 2002). Early observers of the Free and Open Source software movement attributed a sense of community and loyalty to the efforts of jointly building software (DiBona et al., 1999; Zeitlyn, 2003; Shah, 2006). The social practice lens describes learning and socialization as the mechanisms for integrating individual developers into routines of collaboration and sharing (von Krogh et al. 2012). Problem-solving "pairing" as mutual assistance among peers has been observed throughout the history of Free and Open Source software development (Raymond, 1999; Lakhani and von Hippel, 2003) and we can expect mutual assistance to persist unless the purpose of the open collaboration is a competition or remains largely anonymous such as in some formats of broadcast search or crowdsourcing (Jeppesen and Lakhani, 2010). Thus, the collaborations we expect to see based on social exchange are with equals, probably between users rather than crossing actor types because of the perceived differences in ideology (Stewart and Gosain, 2006).

As already pointed out earlier, an important caveat and possibly important distinction relates to the complexity of the problem at hand. The complexity or scope of problem-solving required has been considered theoretically by Alexy and colleagues (2013) as relevant if and when actors expect reciprocity and direct benefits from others' contributions to their problem. Several studies show that developers of OSS are keenly aware of the complexity of their and others' problem-solution pairs when they select between design solutions (Brunswicker and Mukherjee, 2023) or between solving now or deferring to a later stage (Howison and Crowston, 2014). Highly complex problem-solution pairs may significantly increase the costs of design and communication since the scope increases the uncertainty and complexity of knowledge revealed: As a result, neither producers nor users may value the opportunity to work on others' problems since the perceived costs outweigh the perceived direct benefits (see Baldwin and Clark, 2006).

Following this theoretical sketch as a starting point, we apply an exploratory research design in a rigorous analysis of a large sample of actors and problems (labeled as "tasks" in the following) in an open collaboration project before discussing the observed patterns in light of the theoretical starting points. This research design allows us to explore an important issue for innovation management and research policy in open collaboration projects that received little attention yet ample assumptions in recent innovation studies.

3. DATA AND METHOD

3.1. Case Setting

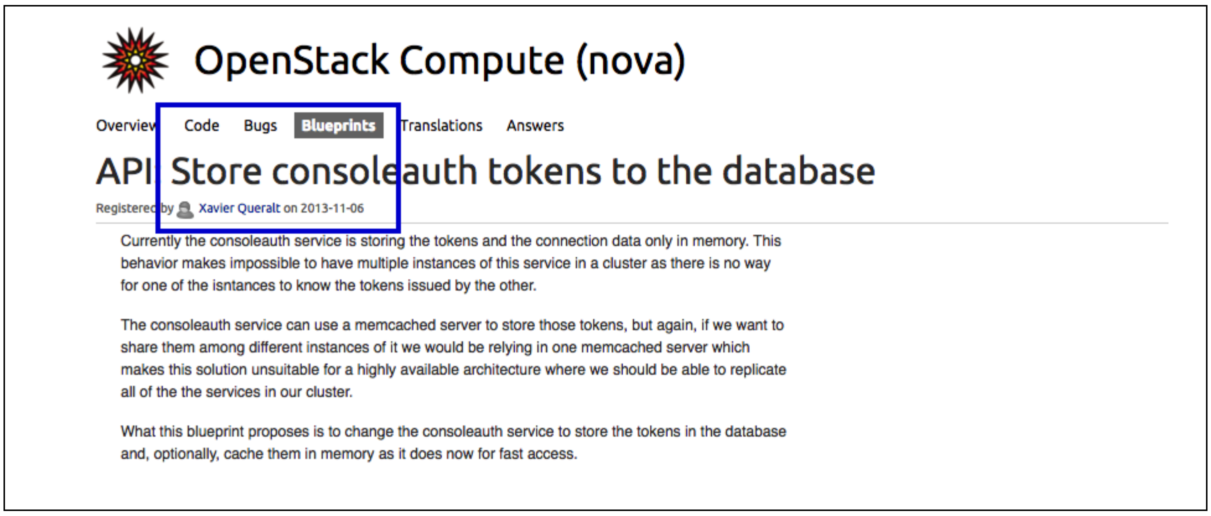
To explore how contributors to open collaboration choose tasks published by multiple actors, we considered several open collaboration projects, and eventually selected a project called NOVA within OpenStack, organized via an open and collaborative OSS community.
Our case selection was motivated by Baldwin and von Hippel (2011) who emphasize the importance of open collaborative models for innovation. Empirically, such models typically emerge in contexts where both producer innovation, single user innovation, and open collaborative innovation co-exist (Baldwin and von Hippel, 2011, p. 1412). Our motivation for the case selection was not to build theory through induction (Eisenhard 1989). Instead, our goal was to explore collaborative problem-solving in open collaboration guided by

economic theories of incentives and situated problem solving assuming human agency guided by humans' perceived costs and benefits of their intended actions (Brunswicker & Mukherjee, 2023). We chose a case that represents a *typical case* in the realms of Baldwin and von Hippel (2011)'s theory of open collaborative innovation, as such innovation models typically occur in settings involving multiple producers and users (e.g. Roberts et al., 2006; Baldwin et al., 2006; Henkel, 2006; Rullani and Haefliger, 2013). These producers and users follow their models of producer and user innovations "outside" of the open collaboration community (Herron and Quinn, 2016). In that sense, our case selection was guided by the goal to choose a case that is empirically representative.

OpenStack is a platform that "seeks to produce the ubiquitous Open Source cloud computing platform that will meet the needs of public and private clouds regardless of size, by being simple to implement and massively scalable" (OpenStack 2016). From an architectural perspective, OpenStack as a platform uses a modular software architecture, with NOVA being one of the core packages within the software stack. Organizationally - following Baldwin and Clark (2011) - NOVA as a project organized within OpenStack represents a typical empirical hybrid innovation model, in which open collaborative innovation, producer innovation, and user innovation co-exist. The open collaborative OSS community and the project team of NOVA is at the core of the hybrid innovation model. Contributors who are producers co-create the core stack, including NOVA, with other producers to ultimately offer software services to manage the entire setup, including software, hardware and networking of the stack. Examples of OpenStack producers are Red Hat, Mirantis, and Whitestack. Users of Open Stack like Target use OpenStack for its private cloud infrastructure. Some of the users may ultimately also use OpenStack to build applications that their users use. The development of the software platform is organized in different repositories that are structured in accordance with the platform's core technological components and packages (i.e. compute, image service, object storage, dashboard, identity service, networking, block storage, orchestration, telemetry, database, bare metal provisioning, multiple tenant cloud messaging, shared file system service, DNSaaS and security API). OpenStack was started by Rackspace Hosting and NASA, which released the data constituting two central "core" repositories of OpenStack: Nova, which focused on virtualization technologies, and Neutrons, which provided technologies that support networking between different devices (OpenStack 2016). Since its public launch, the project has attracted several significant investments from producers, also called OSS cloud vendors (Konrad 2015). The adoption of OpenStack by large technology-oriented corporate users, such as Yahoo, and corporate users from the public sector, such as CERN and Harvard, underlines the success. OpenStack is a project dominated by contributions from sponsored developers (OpenStack 2016), meaning developers who are contributing on behalf of their employers. The publicly available dashboard on contribution behavior reveals that both producers (HP, Cisco) and corporate users (Yahoo, CERN) contribute software code. Within Open Stack, we chose Nova, one of the oldest packages that has successfully evolved in its organizational form of an open collaboration community over time. Given its age, it represents a mature organizational setting of OSS innovation as part of a hybrid innovation model. We will next discuss the data that we sampled about Nova.

3.2. Data

Nova is designed to automate and manage resources and work with existing virtualization technologies (e.g. VMware, Xen or KVM). All three actors - producers, corporate users, and private users - contribute to Nova. We collected data about problem solving activities over a two-year time frame, from 2013 to 2015, in which the 298 actors published and completed 7,470 tasks. For our analysis, after removing empty commits as well as those commits that were not clearly registered as a commit for a particular task (a bug or a feature request), we used only 5,015 tasks. In total, the actors submitted 1,505,868 lines of code to solve the 5,015 tasks. On average, each task required 300.27 lines of code. OpenStack uses Gerrit, a web-based code collaboration and review system, to manage the distributed programming process (Gerrit 2017). OpenStack's Gerrit is closely integrated with git, a distributed version control system (git 2017). Gerrit and git allow each developer to track every single change made to a file at a certain point of time, not only using the terminal but also using a web browser. Jointly, git and Gerrit allow OpenStack developers to work with high data integrity and to maintain software quality. OpenStack's development workflow also supports the selection of development tasks through a web interface. The OpenStack website offers a web-based reporting tool, a so-called Launchpad, that allows developers to register and monitor reported tasks, both bug reports as well as new feature requests. All tasks are visible, observable, and evaluable. The task description provides access to the knowledge and the experiences of the originator of the task, who represents a certain actor, a producer, a corporate user, or a private user (see Figure 1).



API Store consoleauth tokens to the database

Registered by Xavier Queralt on 2013-11-06

Currently the consoleauth service is storing the tokens and the connection data only in memory. This behavior makes impossible to have multiple instances of this service in a cluster as there is no way for one of the isntances to know the tokens issued by the other.

The consoleauth service can use a memcached server to store those tokens, but again, if we want to share them among different instances of it we would be relying in one memcached server which makes this solution unsuitable for a highly available architecture where we should be able to replicate all of the the services in our cluster.

What this blueprint proposes is to change the consoleauth service to store the tokens in the database and, optionally, cache them in memory as it does now for fast access.

Figure 1: Visible Tasks in the Web-based Reporting Tool Launchpad (Blueprints are Features)

The reporting tool provides information about the status of the development process (open, in progress, or closed), and thus supports actors to select tasks, and commit code to them. For this analysis, we sample only tasks that have been successfully solved (closed reports). The web-based reporting tool available via the OpenStack website tracks bugs and features separately. It tracks bugs in a bug report. Bugs reports are ‘used to track known issues and defects in OpenStack software’ (OpenStack 2015b). The OpenStack Launchpad also reports a list of feature reports, so-called blueprint reports, from which developers can assign themselves to a feature development task. OpenStack maintains blueprint reports to ‘track the implementation of significantly new features in OpenStack’ (OpenStack 2015a).

All tasks and their solutions (the code commits and code comments made as well as meta data about when and by whom the commit was made) are traceable via OpenStack’s web-based development workflow (using the tool Gerrit). Thus, we were able to extract information about all the solutions (code commits) made by a certain individual developer. To aggregate this developer level information to the actor-level and associate an actor with a certain task or a certain solution, we used mandatory email information available in the OpenStack profile pages. We classified programmers as sponsored developers of producers or corporate users if their email referred to a corporate domain. We classified developers as private users if a person used an email with a private domain in their profile[1], following prior work on OSS (see e.g. Spaeth et al., 2010). The use of a private domain email instead of a corporate one (e.g. @intel) signals their interest to make contributions independently without any organizational affiliations. This allows them to affirm the (free) nature of their authorship activities e.g. in order to avoid copyright issues but also have the freedom to make contributions guided by personal goals and technical expectations without the need to align with organizational interests and priorities. This distinction, and the role of individual freedoms and rights of all software users if of great historical importance in the free and Open Source software community[2]. To differentiate between producers and corporate users,

[1] e.g. john@intel.com for corporate emails, and email messages associated with Gmail, Hotmail, students, alum, Me, Yahoo, 126, 163, QQ, or RR as private emails.

[2] For references, see the purpose of the Free Software Foundation (https://www.fsf.org/) which lies at the heart of the Free software movement and preceded the Open Source software definition. The foundation offers various tools of identity protection, registration, and historically drew a strong distinction

we used the list of producers available on the OpenStack website (OpenStack 2015b). If two firms contributed to the project before and after a merger and corporate investment, they were considered as one firm if there were no significant contributions made by the acquired company prior to the investment (e.g. Enovance was acquired by Red Hat, PistonCloud was acquired by Cisco, CloudScaling was acquired by Cisco, CloudScaling was acquired by EMC and Bluebox was acquired by IBM). More detailed information about how we mined our data can be found in the online supplementary information (section 5).

3.3. Research Strategy, Measures, and Descriptive Analysis

We implement a task-level analysis to examine actor-specific patterns of collaboration, that is the behavioral choices that the three actors make when working on problems revealed by other actors. In other words, we examine the likelihood that a certain actor develops more solutions (makes more commits) for a certain visible non-focal task compared to another non-focal one. By non-focal tasks we refer to tasks where the originator of the task and the solver of the task are different actors, that is, tasks where problem solving pairing happens. We next discuss the measures and descriptive data used in our analysis.

3.3.1. Task-level Variables and Descriptive Data

Our dependent, independent, and control variables are described below. Dependent variable: Our dependent variable describes the type of task. There are two dimensions that classify a task. The most important dimension for classifying a task is its source (who published the task). The source of the task generates three categorical dependent variables (producer, corporate user, and private users). Furthermore, we consider a second dimension to classify a task, namely its scope (bugs or features). The source and scope of the task generate a total of six categorical dependent variables. Table 2 presents details on our dependent variables.

between individuals and corporations and their interests and power distance. It is safe to say that Free and Open Source software developers are not using individual identifiers lightly or mindlessly.

Data show a relatively balanced number across the sources of tasks (producer tasks, corporate user tasks, private user tasks).

Table 2: Number of Tasks Published in Each Task Category

	Name Category	Absolute values	As percentage of total number of tasks (%)	Number of actors publishing tasks
	Tasks by source			
1	Producer tasks (features & bugs)	1973	39.34 %	23
2	Corporate user tasks (features & bugs)	1876	37.41 %	151
3	Private user tasks (features & bugs)	1166	23.25 %	200
	Tasks by source and scope			
1	Producer features (P features)	711	14.18 %	9
2	Producer bugs (P bugs)	1262	25.16 %	14
3	Corporate user features (CU features)	969	19.32 %	48
4	Corporate user bugs (CU bugs)	907	18.09 %	103
5	Private user features (PU features)	439	8.75 %	53
6	Private user bugs (PU bugs)	727	14.50 %	147

Independent variables*:* Our independent variables measure the allocation of resources of problem-solving actors through the total number of solutions developed by the different actors for a certain type of task. We use the cumulative number of commits made by an actor over a period of two years for a certain published task. Only a cumulative measure can provide insight into an actor’s tendency to work more on a certain task compared to another task, as a task requires a series number of solutions (commits). There are four independent variables: (1) solutions by non-focal actors (independent from actor type) (2) solutions by non-focal producers (3) solutions by non-focal corporate users, and (4) solutions by non-focal private users.

Control variables: We include five control variables: (1) task tenure, that is, the time that has passed since the task was published (in days), (2) participation in the task formulation, measured in the number of reviewers of a certain task, (3) task interdependence, measured by the number of dependencies between the task and other tasks in the sample and (4) solution size, measured by the average number of lines per commit. We also control for the (5) number of solutions contributed by the 'focal' actor, that is, the number of commits made by an actor for their own tasks (solved by the actor who published the task). Table 3 depicts the basic descriptors and correlation coefficients of our variables (before transformation).

Table 3: Variables, Descriptors and Correlation Coefficients (before transformation)

			Correlation								
Variable	µ	SD	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Control Variables											
(1) Task tenure	399.47	218.60		0.35	0.00	0.00	0.03	-0.13	-0.06	-0.11	0.02
(2) Participation	5.89	2.03	0.35		-0.04	-0.02	0.07	-0.02	0.00	-0.02	0.00
(3) Task interdependence	0.76	1.16	0.00	-0.04		0.04	0.16	0.07	-0.03	0.11	-0.03
(4) Solution size	90.47	3465.63	0.00	-0.02	0.04		-0.01	-0.01	0.00	0.00	0.00
(5) Solutions focal actor	4.14	7.13	0.03	0.07	0.16	-0.01		-0.25	-0.14	-0.19	-0.07
Independent Variables											
(6) Solutions by non-focal actors	2.32	5.00	-0.13	-0.02	0.07	-0.01	-0.25		0.52	0.78	0.17
(7) Solutions by non-focal CU	0.78	2.98	-0.06	0.00	-0.03	0.00	-0.14	0.52		-0.08	-0.03
(8) Solutions by non-focal P	1.40	4.17	-0.11	-0.02	0.11	0.00	-0.19	0.78	-0.08		-0.03
(9) Solutions by non-focal PU	0.15	1.08	0.02	0.00	-0.03	0.00	-0.07	0.17	-0.03	-0.03	
*All correlations higher than 0.10 are statistically significant, sample size of 5015											

For our statistical analysis, all variables have been log-transformed to correct for skewness as well as scaled using the standard z-scores (Basic descriptors can be found in the online supplement in section 2). We do not detect signs of multicollinearity in our regression analyses when using transformed and scaled items (highest VIF was 1.96<4).

3.3.2. Statistical Modeling

As mentioned earlier, we focus on the allocation of solutions to certain tasks in order to examine the variability in the creation of problem-solution pairs.. Here, our dependent variable (task type) is discrete and nominal. Thus, we use a multinomial logit (MNL) regression to probe our research question. MNL allows us to estimate the probabilities that certain actors contribute their solutions to certain discrete problem categories (McFadden 1974). MNL is one of the most commonly used regression model in social science for nominal outcomes. We use the existing library "mlogit" available for the statistical software R to carry out the multinomial logit model (Croissant 2013).

We split our analysis in two phases, gradually increasing the granularity of our analysis. In phase 1, we focus on three categorical variables: corporate user tasks, producer tasks and private user tasks. Using these three variables, we estimate three MNL models. The first model includes all the control variables, the second model includes our single independent variable (solutions from other non-focal actors), and the third model includes three categorical independent variables (solutions from non-focal corporate users, non-focal producers, non-focal private users). We examine each model's statistical significance and the improvement in the goodness of fit when sequentially adding additional variables. The MNL requires us to define a comparison group. Given the importance of private users in the history of OSS, we choose private user tasks as the comparison group.

Our research question requires us to estimate whether certain tasks are more likely than other ones, implying changing comparison groups. Thus, when using private user tasks as the only comparison group, we cannot examine a different comparison group, e.g. whether producers are more likely to work on corporate user tasks compared to producer tasks (or vice versa). Thus, in accordance with Long and Freese (2003) we calculate the binary contrasts (e.g. corporate user tasks over producer task). These effects are computed using logits according to the following formulas to compute the new mean, variance, and z-value.

$$\widehat{\beta_{new}} = \widehat{\beta_{old2}} - \widehat{\beta_{old1}}$$

$$\widehat{Var}(\widehat{\beta_{new}}) = \widehat{Var}(\widehat{\beta_{old2}}) + \widehat{Var}(\widehat{\beta_{old1}}) - 2\widehat{Cov}(\widehat{\beta_{old2}}, \widehat{\beta_{old1}})$$

$$z = \frac{\widehat{\beta_{new}}}{\sqrt{\widehat{Var}(\widehat{\beta_{new}})}}$$

In phase 1, we focused on three contrasts: (1) corporate user over private user tasks, (2) producer over private user tasks, and (3) producer over corporate user task. After that, we transformed the logged odds into exponentiated factor changes (or odds ratios) in accordance with Long and Freese (2003) to articulate the actual effect size of the coefficients. A coefficient of 1 implies that the independent variable (amount of solutions allocated to a certain task) does not have an effect on the dependent variable (task choice). A coefficient smaller than 1 (e.g. 0.9) suggests that, the odds that an actor works on that particular task (e.g. a producer task) is smaller (e.g. smaller by the factor 0.9, or 10% smaller) compared to the comparison group (e.g. a corporate user task), holding all other independent variables constant (Szumilas 2010). The opposite is true for an effect size larger than 1 (e.g. large by the factor 1.1, or 10 % larger). In all our models, we report factor changes in order to ease the interpretation of the results.

In phase 2, we move a step further and account for the source as well as the scope of the task, leading to six categorical variables as dependent variables (see Table 2). As in the prior phase, we first chose private user bug reports as the comparison group when estimating the MNL. We calculated six binary contrasts following Long and Freese (2003). Essentially, we compared not only the two different sources (e.g. whether a producer is more likely to work on a corporate user task than a producer task) but looked into the scope of the task more specifically when engaging with others' innovations (e.g. are producer more likely to work on corporate user features compared to producer features). The six contrasts are: 1) Corporate user features over private user features, 2) corporate user bugs over private user bugs, 3) producer features over private user features, 4) producer bugs over private user bugs, 5) producer features over corporate user features, and 6) producer bugs over corporate user bugs. In phase 2, we also report exponentiated coefficients to express effect sizes.

We performed a series of additional explorative regression analyses. We calculated all 30 potential binary contrasts that provided further information about problem-solution pairs (e.g. probabilities of producer feature over corporate user bug). Finally, we performed a robustness analysis at the commit level. We changed our data structure and estimated a robust regression with clustered standard errors using the ClusterSEs package in R in order to account for the potential bias because of clustering of solutions among task sources (Esarey 2017). The results are reported in the supplementary information (section 4).

4. FINDINGS

We report our results in two stages. First, we report the results for our analysis with three different task sources as choice variables: corporate user tasks, producer tasks, and private user tasks. Then, we report the results for six different task choices that consider both the source as well as the scope of the task. In both stages, we start first with an exploratory discussion using heatmaps before summarizing the regression results.

4.1. Stage 1: Problem-Solution Pairing

A visualization of the collaboration activities across the 5015 tasks provides a first descriptive insight into the actors' problem-solution pairing. Figure 2 maps the solutions from different actors onto the tasks published by others. The color coding highlights the number of solutions (commits) made by the different actors, dark colors indicating a lower number of commits and bright colors (yellow) indicating a greater number of commits allocated.

The color coding makes one cell stand out from the rest. In column 2 of the heatmap, the bright yellow color indicates that corporate user tasks (CU tasks) receive the highest number of solutions overall. They come from producers. When comparing the cells for other tasks to which producers could allocate their time and resources, we learn that private user tasks (PU tasks) rank second, while other producer tasks (P tasks) receive the lowest number of commits (less than 200 solutions). When looking at the problem-solution pairing of corporate users (solutions by non-focal corporate users, column 1), we learn that corporate users allocate more solutions to corporate user tasks (CU tasks) and private user tasks (PU tasks) than to producer tasks (P tasks).

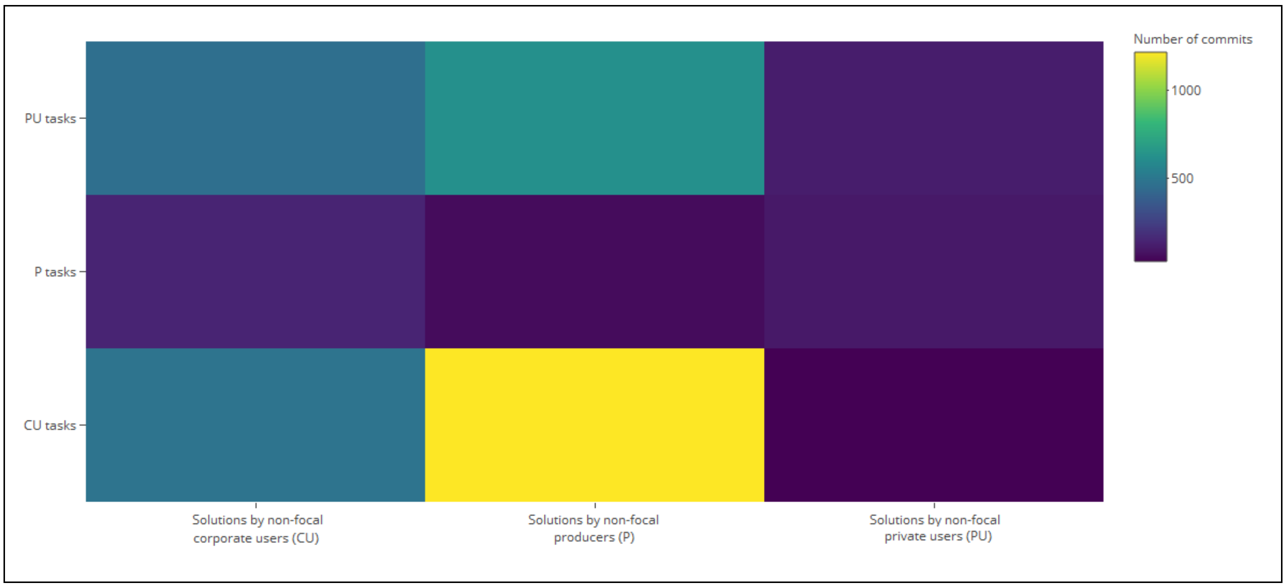


Figure 2: Heatmap with Three Task Categories (by Source) to Number of Solutions (Commits)

Finally, in column 3, we learn that private users overall make a smaller number of commits. Further, the color coding does not show pronounced patterns of collaboration.

Equipped with these exploratory results, we next present the results of our MNL regression analyses. As indicated earlier, in the first set of analyses we focus on three types of tasks (corporate user task, producer tasks, private user tasks). In our regressions we estimated binary choice variables for three behavioral preferences (or choices): (1) corporate user over private users tasks (CU over PU tasks), (2) producer over private user tasks (P over PU tasks), and (3) producer over corporate user tasks (P over CU tasks).

We estimated our models in three stages in order to gradually increase the insight into our research question. First, we estimate our control model (model 0) with the control variables only. We then estimate model 1 that contains a single independent variable, namely the number of solutions committed by other actors (meaning all actors except for source of the task). This simple model provides insight into whether there is a project-level pattern of collaboration, that is, a general tendency to work more on certain task sources than others, independent of the actual problem solver (the actor who submitted the solution). Subsequently, we estimate model 2 which includes three different independent variables. These independent variables differentiate the choices made by different actors, meaning we distinguish solutions made by corporate users, solutions made by producers, and solutions made by private users. All models were statistically significant. The goodness of fit improved as we added more variables. The AIC value is smallest for model 2 (AIC of 9248.2 for model 2 compared to 9476.4 for model 1) suggesting that it provides the best fit. Table 4 presents the results for model 1 and model 2. We report the control model 0 in the online supplement (section 3.1). In this table we present factor changes (exponentiated odds ratios) in order to express the effect sizes of the coefficient on certain behavioral patterns.

In model 0, we find statistically significant effect for almost all control variables across all the three behavioral choice sets, suggesting that task tenure, participation, task interdependence, and also the solutions from the focal actor shape problem-solving patterns. Only the coefficient solution size, the average number of lines of code associated with all the commits made for a particular task, was not or only barely statistically significant and also very small in effect size. This suggests that the size of a solution is not an important predictor of behavioral preferences for tasks (please refer to online supplement for further details on the control model 0).

In model 1, we include only a single independent variable to predict collaborations across all actors. That variable is the number of solutions contributed by non-focal actors, in other words, all commits made to a certain task that do not come from the focal actor. We find three statistically significant effects over and above the effect of the control variables. This clearly suggests variability in problem-solution pairing. We observe a strong tendency of all actors to engage more with corporate user tasks compared to other tasks. First, we find that the coefficient of producer over private user tasks is 0.29 (P over PU). If actors work on a task from another actor, they are less likely to work more on tasks published by a producer compared to tasks published by a private user. The odds are 71 % smaller. Simply speaking, any kind of actor who engages in collaboration with other actors is more likely to work on tasks published by a private user compared to a task published by a producer. Second, the coefficient for a producer task over a corporate user task is 0.23 (P over CU). The odds are 77 % smaller. In other words, actors are less likely to work on a producer task compared to a corporate user task. In sum, these findings reveal that need-solution pairing is not random. There is a project-level pattern. Actors, independent of their type, have a strong inclination to work more on corporate user and private user tasks compared to producers.

In model 2, we include three independent variables to estimate actor-specific problem-solving behaviors:1) solutions by non-focal corporate users, 2) solutions by non-focal producers, and 3) solutions by non-focal private users. Again, we estimate and report the statistical significance of the effects for three contrasts (CU tasks over PU tasks, P tasks over PU tasks, and P tasks over CU tasks). Seven effects are statistically significant at $p < 0.05$.

We find that corporate users are less likely to develop a solution for a producer task compared to a private user task or a corporate user task. The estimate is 0.50 for both task choices (P over PU, and P over CU). The odds are 50 % smaller that corporate users work on tasks published by producers compared to tasks published by corporate users or private users. In other words, corporate users work more on tasks of other users compared to producers.

We find that producers are also less likely to work on producer tasks compared to private user tasks and corporate user tasks. The estimated coefficients are 0.18 (P over PU) and 0.14 (P over CU) respectively. The odds are 82 % and 86 % lower respectively. Further, there is evidence that producers are also more likely to work on corporate user tasks compared to private user tasks. In sum, the results suggest that problem-solution pairing by producers unfolds as follows: They engage with tasks of corporate users first, followed by tasks of private users, and lastly by tasks of other producers

Table 4: MNL Results for Three Contrast Values (reported in exponentiated coefficients)

	Model 1			Model 2		
	CU over PU task	P over PU task	P over CU task	CU over PU task	P over PU task	P over CU task
Control Variables						
Intercept	1.50***	3.04***	2.03***	1.74***	1.08	0.62***
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Task tenure	1.22***	1.12**	0.92*	1.22***	1.09*	0.90**
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Participation	0.78***	0.77***	0.99	0.78***	0.77***	0.99
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Task interdependence	1.25***	1.29***	1.03	1.23***	1.33***	1.08*
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Total solution effort	0.96	0.98	1.02	0.97	0.97	1.01
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
	1.38***	1.43***	1.03	1.40***	1.37***	0.98

Solutions by focal firm	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Indep. Variables						
Solutions by other actors	1.24*** (1.00)	0.29*** (1.01)	0.23*** (1.01)			
Solutions by other non-focal corporate users				1.01 (1.00)	0.50*** (1.00)	0.50*** (1.00)
Solutions by other non-focal producers				1.27*** (1.00)	0.18*** (1.01)	0.14*** (1.01)
Solutions by other non-focal private users				1.26*** (1.00)	1.10* (1.00)	0.88*** (1.00)
Model Parameters	*Df:14*	*-LL:-4724.2*	*AIC:9476.4*	*Df:18*	*-LL:-4606.1*	*AIC: 9248.2*

Furthermore, we find that private users are less likely to work with producers compared to corporate users. The estimate is 0.88 (P over CU), suggesting that the odds are 12 % lower. Interestingly, though, private users are also more likely to work with corporate users compared to other private users. Private users also prefer producer tasks over private user tasks. These results create a more refined understanding of how private users engage in collaboration: Corporate users rank first, followed by producers, and finally other private users, their own peers.

4.2. Stage 2: Task Complexity

We next consider not only the source of the task but also the scope of the task (bug or feature) to differentiate between the complexity of knowledge and work required. Before discussing the regression results, we first present a visualization of the problem-solution pairs inside the open collaboration project to provide a descriptive overview of how solutions are paired with problems that are either path extending or path creating. Figure 3 presents a heatmap that describes at an aggregated level the distribution of solutions (commits) across six different task categories.

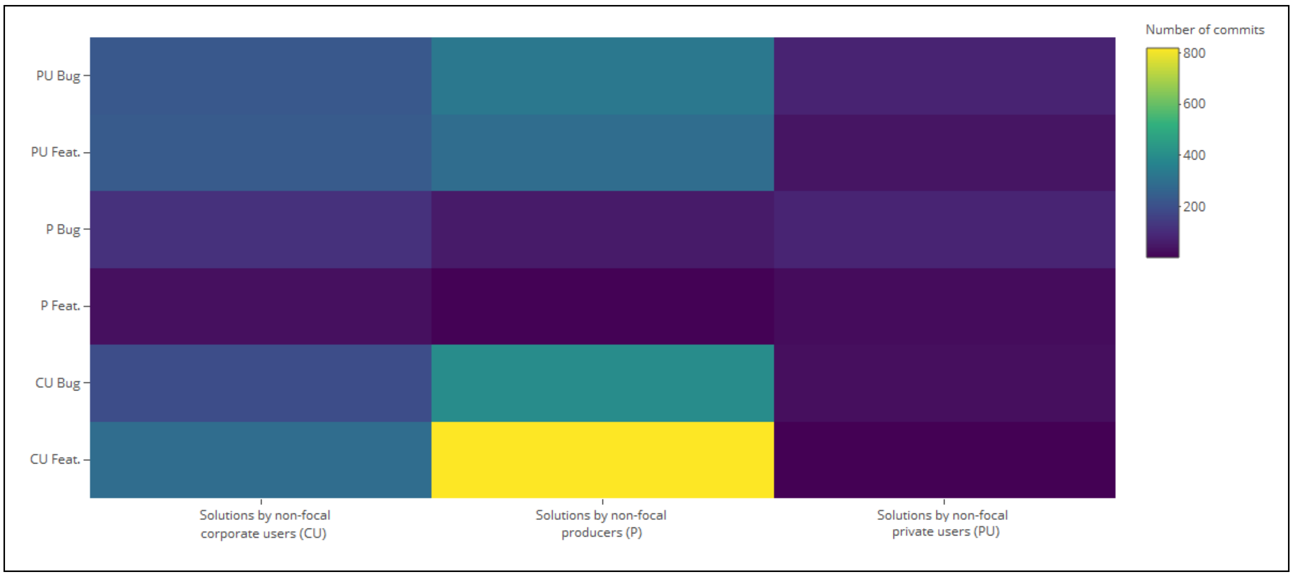


Figure 3: Heatmap with Six Task Categories (by Source and Scope) to Number of Solutions (Commits)

The heatmap gives a first indication of actor-specific problem-solution pairing. Again, one bright yellow cell stands out from the rest. Producers make more than 800 commits to corporate user features. When comparing this effort with efforts made for other tasks in column 2 we see that corporate user bugs rank second, closely followed by bugs and features published by private users. This suggests that novel features published by corporate users impact behavior of producers, potentially because path-creating problems raised by corporate users have a high potential to improve the performance of the technology. In column 1, we observe that corporate users are also focused on features published by other corporate users while features published by producers are least important. In column 3, the descriptive data do not suggest any particular patterns of problem-solution pairing in the open collaboration project.

Equipped with these descriptive observations, we now present our MNL results with six binary contrasts in order to examine path extension and path creation: (1) corporate user feature over private user features, (2) corporate user bugs over private user bugs, (3) producer features over private user features, (4) producer bugs over private user bugs, (5) producer features over corporate user features, and (6) producer bugs over corporate user bugs. We estimate three different models (model 0 with control variables only, model 1, 2 and 3 sequentially adding the three independent variables). All models were statistically significant. Further, the AIC was lowest in model 3 compared to all other models (for further details and additional contrasts please refer to the online supplement section 3.2). In Table 5 we present the results for six binary contrast values for model 2 that includes three independent variables: (1) solutions by non-focal corporate users, (2) solutions by non-focal producers, and (3) solutions by non-focal private users. We report exponentiated coefficients (factor changes). The regressions provide insight into the role of scope for actor-specific collaboration.

With respect to corporate users, we find that they are less likely to work on producer features compared to private user features. The same applies to producer bugs compared to private user bugs. The effect sizes are 0.45 (P features over PU features) and 0.90 (P bugs over PU bugs) respectively. This suggests that the odds that a corporate user works on a producer task rather than a private user task are lower (55 % compared to 10 %) if the task is about features and not about bugs. Further, we find that corporate users are less likely to work on producer features compared to corporate user features. They are also less inclined towards producer bugs compared to corporate user bugs. The coefficients are 0.44 (P features over CU features) and 0.58 (P bugs over CU bugs) respectively, suggesting that this tendency to work on corporate user tasks rather than on producer task is particularly strong for features. Taken together, these results strengthen our arguments about the importance of user tasks: Corporate users are more likely to engage with tasks revealed by other users compared to producer tasks, in particular if such tasks relate to features rather than bugs (path creation rather than path extension).

With respect to producers, we find that they are less likely to work on other producer features compared to private user features. The effect size is 0.09 (P features over PU features) suggesting that the odds that a producer contributes more solutions to another producer's feature instead of another private user feature decrease by 91 %. For bugs, we did not find a preference. The results also suggest that producers are less likely to work on producer features compared to corporate user features. The same negative tendency exists for producer bugs compared to corporate user bugs. The estimated coefficients are 0.06 (P features over CU features) and 0.18 (P bugs over CU bugs) respectively. The odds that a producer works on another producer's feature compared to a feature published by a corporate user are 94 % smaller (compared to 82 % for bugs). In other words, producers are much more likely to work on features published by corporate users and private users compared to features published by other producers. The variability of problem-solution pairs is reinforced when we take the strategies of path extension and path creation into account (scope of the task, i.e. feature versus bug). Path creation opportunities lead producers to focus even more on user tasks.

Finally, we gain more refined insight into the problem-solution pairing efforts of private users. First, we find that private users are less likely to work on producer bugs compared to corporate user bugs. The estimated effect size is 0.87 (P bugs over CU bugs), suggesting that the odds decrease by 13 %. Interestingly the effect is not statistically significant for the choice of producer features over corporate user features. This suggests that the collaboration efforts with corporate users are mainly driven by bugs rather than features.

Table 5: MNL Results with Six Contrast Values (exponentiated coefficients)

	CU over PU		P over PU		P over CU	
	CU feat. over PU feat.	CU bugs over PU bugs	P feat. over PU feat.	P bugs over PU bugs	P feat. over CU feat.	P bugs over CU bugs

Control Variables						
Intercept	2.53***	1.36***	0.67*	1.36***	0.26***	0.89*
	(1.01)	(1.00)	(1.04)	(1.00)	(1.03)	(1.00)
Task tenure	1.34***	1.15**	1.05	1.15**	0.78***	0.95
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Participation	0.73***	0.80***	0.72***	0.80***	0.98	1.02
	(1.00)	(1.00)	(1.01)	(1.00)	(1.00)	(1.00)
Task interdependence	1.19**	1.12*	1.34***	1.12*	1.12*	1.21***
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Total solution effort	0.78***	1.09*	0.80***	1.09*	1.02	1.05
	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)	(1.00)
Solutions by focal firm	1.56***	1.10	1.90***	1.10	1.22**	0.96
	(1.01)	(1.01)	(1.01)	(1.01)	(1.00)	(1.01)
Independent Variables						
Solutions by other non-focal corporate users	1.01	0.90*	0.45***	0.90*	0.44***	0.58***
	(1.00)	(1.00)	(1.01)	(1.00)	(1.01)	(1.01)
Solutions by other non-focal producers	1.33***	1.01	0.09***	1.01	0.06***	0.18***
	(1.00)	(1.01)	(1.14)	(1.01)	(1.14)	(1.02)
Solutions by other non-focal private users	1.66**	1.15**	1.51*	1.15**	0.91	0.87***
	(1.04)	(1.00)	(1.04)	(1.00)	(1.00)	(1.00)

We also find that private users are indeed more likely to work on producer features compared to private user features. We identify a slightly higher tendency to work on producer bugs compared to private user bugs. The estimated effect sizes are 1.51 (P features over PU features) and 1.15 (P bugs over PU bugs) respectively, suggesting that the odds of producer features increase by about 50 % compared to private user features, while the odds of producer bugs increase by 15 % compared to private user bugs. Finally, we observe that private users prefer corporate user features over private user features. There is also a slightly greater tendency to work on corporate user bugs compared to private user bugs. Our robustness analyses reported in the online supplement (section 4) further corroborate that problem-solution pairing we identify in the prior sections are strengthened when path creation is at stake. However, this is only true for producers and corporate users. For private users we do not find the strong dominance of corporate user features over producer features. Instead, private users engage in path extension knowledge revealed by corporate users - related to niche creation revealing strategies - that make them more attractive than producer tasks.

5. DISCUSSION

5.1. Theoretical Contributions

The emergence of open collaboration highlights the shift towards new forms of organizing the development of economically important technologies the Open Source way. In this paper, we explore how users and producers pair problems and solutions inside open collaboration and specifically an OSS community that follows Baldwin and von Hippel's (2011) model of open and collaborative innovation. Open collaboration can bring together autonomous producers and users – including corporate and private users – in the same organizational context in which they purposively engage in collaboration to jointly create a public good. The actors differ in their incentives they face when engaging in collaboration, which are shaped both inside *and* outside of the community, in the actors' context of competition and use (von Hippel, 1994; Dahlander and Magnusson, 2008).

Our key findings are the following: First, we find that problem-solution pairing does not occur only driven by the design structure of the code (Baldwin and Clark, 2006) since each actor's underlying perceived incentives are shaped not only by the scope of the problem but also the actor type who revealed the problem. Second, we find that users and producers create solutions for others' problems instead of just openly solving their own problems (Howison and Crowston, 2014) or shaping their own agenda or nurturing their own niche by developing software in public with the likely goal of selling complementary products and services that are layered “on top” of the public good as part of a greater hybrid innovation architecture (Baldwin and von Hippel, 2011; Teece, 1986). Third, we extend the innovation model underlying open collaboration by pointing to the specific role that corporate users (versus private users) appear to play: by signaling needs and feature requests corporate users communicate options to sell complementary services outside the OSS community. Corporate users’ problems attract producers who tend to pair solutions and problems with special regard to their problems. These findings provide new insights into theories of selective revealing (Alexy et al. 2011) but also extend a line of research on arguments of generalized exchange (Gosh, 2005; Baldwin and Clark, 2006). Figure 1 depicts a summary of the collaboration dynamics we observe and conceptually answers the question of “who solves whose problems” without claiming to explain the complex incentive mechanisms that cause these dynamics. We chose a conceptual figure to illustrate the dominant problem-solving patterns (which type of actor produces solutions for tasks published by a certain type of actor) and also to highlight the need to consider the perceived context of use that surrounds corporate users and the tasks (features and bugs) they publish in the community.

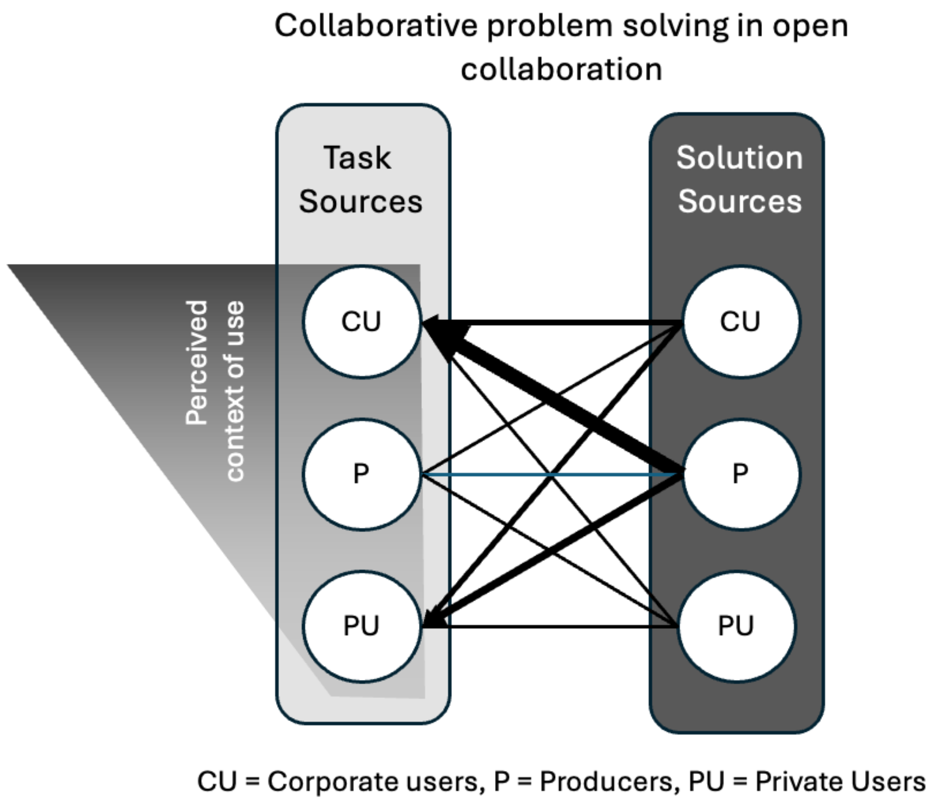

Collaborative problem solving in open collaboration
Task Sources
Solution Sources
Perceived context of use
CU
P
PU
CU
P
PU
CU = Corporate users, P = Producers, PU = Private Users

Figure 1: Collaboration Patterns in Open Collaboration

Most importantly it deepens our understanding of the inner working mechanism of open collaborative OSS communities (Baldwin and von Hippel, 2011) assuming human agency in assessing costs and benefits when making a choice about whose problems to solve. Our explorative results call for more attention in research on the economics of innovation in new organizational forms of innovation that are hybrid in nature and mix open collaborative innovation along with producer and user innovation: Such research should focus on offering new theories and explanations of how different actors and their actions are shaped by unique incentive structures created “inside” and “outside” of the OSS community that follow Baldwin and von Hippel’s (2011) open and collaborative model of innovation.

First, we report a general pattern of collaboration across actor types which departs from the assumptions shared in earlier work that many contributors work mostly on their own problems or choose their tasks based on the option value offered by the (modular) structure of the technology (Brunswicker and Mukherjee, 2023; Howison and Crowston, 2014; Baldwin and Clark, 2006). Earlier work on open collaboration tended to take the composition of actors in the community as relatively homogenous, with early work already differentiating between paid or sponsored versus volunteers (Stewart et al., 2006; Roberts et al., 2006). However, future research should evaluate to what extent differences within the user community matter, in that sponsored developers enter with their corporate interests that cast a wider net across industry and complementary as well as competitive interests. Corporate users appear most successful in setting the agenda in open collaboration projects possibly because they occupy a nexus of user knowledge and commercial relevance for other actors. This insight may come to inform the bigger dynamics of knowledge flows and help discriminate between actors of higher relevance to the sustained innovation activity in open collaboration projects (Faraj et al., 2011).

Task scope might play a more ambiguous role in relation to collaboration inside an open collaboration project. Howison and Crowston (2014) observed that greater scope and the associated complexity of the task and knowledge revealed in path-creating actions leads potential collaborators to defer engagement with knowledge revealed by others and focus on their own problems in the majority of innovation work that occurs within open collaboration projects. We see here that path creating innovations, however, can actually lead to increased need-solution pairing if, and only if, the revealing occurs from corporate users. While we cannot settle this empirical puzzle, we suggest that future research focuses more attention on the source of the knowledge revealed because of the indirect benefits to actors: in other words, incentives to reveal knowledge that originate outside the open collaboration project may influence the engagement and need-solution pairing with other actors within the project.

Second, we directly contribute to the theory of open collaboration anchored in innovation economics which assumes that both producers and users are guided by distinct direct and indirect costs and benefits when choosing to participate in open collaboration (Baldwin and Clark 2000; Baldwin and von Hippel 2011). To our knowledge this study is the first to suggest that producers and users may actually see direct benefits from contributing to others' problems, even though such contributions may potentially increase their own costs. If there are different actors in the open collaboration community, they are not just contributing their solutions following their own interest assuming complementarity between the different needs and solutions. They perceive benefits of generalized exchange because each code contribution is complementary to each other without the need to align interest. Instead, they accept higher communication costs and also transaction costs that occur when they coordinate their joint problem-solving via actionable transparency and observability (Baldwin and von Hippel, 2011; Boudreau and Lakhani, 2015). Even though they can see revealed problems, articulated in natural language as well as programmatic ones (e.g. code snippets), it is not a straightforward task to develop solutions that meets that needs. It requires coordination and knowledge collaboration. Thus, our research deepens our understanding of how open collaboration actually works in hybrid settings of innovation, where producers seek to sell complementary services "outside" of the OSS community, e.g. for deploying the software to users of the platform (Baldwin and von Hippel, 2011; Teece, 1986). Further, it also advances producer's strategy to selectively reveal knowledge (e.g. by contributing problems and solutions) to the shape other actors' agenda, may actually be recursively shaped by other actors' revealing strategy (Harhoff et al., 2003; Henkel, 2006; Alexy et al., 2013). Producers, for example, may not just follow their own interest inside the community but instead may purposively build modules in response to corporate users needs, in order to create the "foundations" (the technical features and interfaces) that are needed to build complementary proprietary solutions “layered on top” of the OSS community’s product. In other words, we corroborate and further refine the argument made by Alexy et al. (2013) that producers and users go beyond free-riding to reap indirect benefits “outside” of the project without any actual code contribution. They perceive indirect benefits from contributing to other actors’ problems because this may help them benefit from complementary assets created outside of the OSS community (see Teece, 1986). And while many problems may still be solved by producers on their own, obvious from the relatively low engagement between producers and users, our empirical study suggests that actors can also expect others to engage with problems or solutions they reveal in order to understand their needs and intentions. For example, producers may work on corporate users’ problems to learn about their needs and internal challenges. This may potentially impact and change the agenda set forth by corporate users since the problems may be solved in other ways than expected. This finding could neither be expected from the existing literature on open collaboration or OSS communities nor does it lend itself to simple explanations.

We may further speculate about the reasons for the high pairing of problems revealed by corporate users (in their attempt to spread issues or maybe even shape an agenda) with solutions from other corporate users and producers. It seems obvious that this creates a direct benefit to the corporate user - they get their problem solved. Additionally, direct and indirect benefits are available to the producers and users who engage with knowledge shared by corporate users. We note that the indirect benefits accruing to the actor seeking out the knowledge shared by corporate users may explain part of the pattern we observe: producers might serve or hope to serve the corporate user or comparable similar users in contracts outside the open collaboration project and thus engage with their knowledge. The engagement may accelerate innovation learning because they can observe corporate users’ needs when solving corporate users’ problems (Boudreau and Lakhani, 2015). In addition, it may help to build improved relationships with them that are beneficial outside of the open collaboration project to sell complementary products and services while potentially saving marketing costs. As discussed by Baldwin and von Hippel (2011), these direct and indirect benefits outweigh the costs and the risk from revealing solution knowledge related to problems that could potentially be solved in a proprietary way.

Third, this last point raises a potentially more general point in how open collaboration projects take shape and sustain within wider market and industry dynamics. The effort to understand and engage with other actors' needs and problems falls under communication costs, which tend to undermine the viability of open collaboration in favor of producer innovation (Baldwin and von Hippel, 2011). We find a first indication that open collaboration might be more widely viable than previously expected in that corporate users rank first in attracting other actors' solutions and implying their communication costs outweighed by the potential benefit of complementary (outside) market opportunities of solving corporate users' problems. Our finding that complex tasks appear even more attractive adds robustness to the insight that communication costs are overcome and attention shifted towards corporate users in the OSS community.

While from a perspective of knowledge exchange within the project (see Faraj et al., 2011) interaction dynamics appear fragile and actors' engagement uncertain, a shift in perspective to wider (potential) market and non-market relationships between corporate actors may reveal the networks and relationships that could lead to the emergence of innovative open collaboration projects in the first place, or lead to their demise over time. We refer to the indirect benefits of engaging with specific other actors, corporate users, and the possibility of facing lower direct communication costs while foregoing higher indirect communication costs to understand the needs of potential clients and customers. This mediation mechanism inside open collaboration projects manifesting in the distinct interaction profile of and with corporate users changes the default innovation model for open collaboration (Baldwin and von Hippel, 2011: p1408) and suggests incentives similar to a single producer model when in fact the technology under development is a public good. It remains an open question to what extent the attention shift towards corporate users represents an investment into a potential customer group outside and translates into sales of complementary services. Further, it also requires further research to examine how such joint problem-solving actions challenge the viability of the open and collaborative innovation model (Baldwin and von Hippel, 2011). It may fail if it is primarily guided by producers' interest to sell complementary proprietary solutions "on top", unless producers continuously revisit their actions with respect to their alignment with the OSS community's goal to create a public good and create solutions inside the community that are complementing other solutions in creating publicly valuable solutions.

Furthermore, the relative absence of peer reciprocity in our data comes as a surprise to scholars of Free and Open Source software development (von Krogh et al., 2012) as well as user innovation more generally (Franke and Shah, 2003; Lakhani and von Hippel, 2003). It may well be that peer reciprocity applies more strongly to pure-play peer communities characterized by close ties and shared experience and less to projects with heavy corporate involvement. Future research should attempt to evaluate the role that shared values play (they are present in our context in the form of the commitment to Open Source software), shared individual experience, individual passion, and corporate support or stated interests.

5.2. Limitations and Implications for Policy and Management

The analysis was performed on a dataset in open collaboration that took place over a period of two years in one of the most prominent and successful Open Source software development projects. A limitation of this study is that it focuses on a single case, Open

Stack, in which all interactions take place. Given the foremost appeal of Open Stack to corporate users and producers the role of private users, and insights gained from their behavior, remains limited here. Future work should sample multiple projects of broad scope and appeal in order to integrate multiple sectors and a higher diversity of corporate and non-corporate participants. Future studies should compare variability of problem-solution pairing across projects that all include both producers and users. The necessity for coordination of knowledge work is not well understood in distributed work, and our results suggest that more qualitative work is needed to understand how different actors build collaborative relationships and select between possible collaborators of one type. In addition, our work also raises questions as to the (limited) amount of typical selective revealing as the actor's decision of sharing knowledge in the first place. Verbal communication and parallel discussions represent limited knowledge sharing and could precede code submissions. Furthermore, we are not measuring actor-specific goals and experiences but infer them from the actor type. Even though there is evidence that producers and users have distinct goals for software development (Germonprez et al. 2016), future studies should advance our work by actually empirically observing and measuring goals and experiences.

Mainstream open collaboration outcomes (e.g. Linux, Apache, or Android), now used by billions of users, have a notable market share in the industry and have attracted large financial investment by companies and corporate donors (Black Duck Software 2015; Crowston et al. 2012; Ghosh et al. 2002). The success of open source has led scholars to seek to deepen the understanding of this new organizational model of technology development. Yet, this is not an easy task as many firms do not know how to engage effectively with communities that bring together multiple producers and users, either as producer or corporate user.

More generally, innovation policy should take note that the availability and viability of open collaboration projects depends both on available incentives inside as well as outside the project (Gächter et al., 2010; Faraj et al., 2011) and that actors who are users and share innovation of reasonable scope tend to have outsized influence over the collaboration that unfolds. Identifying users, and corporate users in particular, becomes a critical task for policy implementation: users face stronger direct benefits from collaboration due to their own use of the innovation and likely much lower indirect benefits.

An important insight is that selective revealing of knowledge by corporate users is disproportionately shaping how the technology evolves - specifically the revealing of path creating knowledge. Our findings hold insights building on recent work on public policy to support open innovation (Cano-Kollmann et al., 2017), namely that the type of firm receiving support matters. Supporting corporate users in their selective revealing of innovation may trigger stronger and different engagements in open collaboration projects than supporting producers and research has yet to differentiate between firm types along other demographic and behavioral characteristics.

Lastly, innovators have long attempted to increase the gains possible from attracting contributions to their projects from others, including volunteers, and have sponsored open collaboration projects (Spaeth et al., 2015, Stewart et al., 2006). Our finding that problem-solution pairing frequently involves corporate users might suggest that they are best placed to orchestrate open collaboration projects and are likely to receive engagement from other actors. The notion that the actor type is important when adopting a sponsorship role in an open collaboration project is novel and complements the idea that selective revealing drives the adoption of an agenda or the creation of a niche (Alexy et al., 2013) and that community-based credibility is important (Spaeth et al., 2015). The motivation (and nature of incentives driving the actor) appears to matter beyond the actor themselves and influence the variability of actors creating problem-solution pairs in open collaboration.

6. REFERENCES

Alexy, O., George, G., Salter, A.I., 2013. Cui Bono? The Selective Revealing of Knowledge and Its Implications for Innovative Activity. Academy of Management Review 38, 270–291. https://doi.org/10.5465/amr.2011.0193

Baldwin, C., Clark, K., 2000. Design Rules: the power of modularity. MIT Press.

Baldwin, C., Hienerth, C., von Hippel, E., 2006. How user innovations become commercial products: A theoretical investigation and case study. Research Policy 35, 1291–1313. https://doi.org/10.1016/j.respol.2006.04.012

Baldwin, C., von Hippel, E., 2011. Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation. Organization Science 22, 1399–1417. https://doi.org/10.1287/orsc.1100.0618

Baldwin, C.Y., Clark, K.B., 2006. The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? Management Science 52, 1116–1127. https://doi.org/10.1287/mnsc.1060.0546

Black Duck Software, 2015. 2015 Future of Open Source Survey Results.

Bonaccorsi, A., Rossi, C., 2003. Why Open Source software can succeed. Research Policy 32, 1243–1258. https://doi.org/10.1016/S0048-7333(03)00051-9

Boudreau, K.J., Lakhani, K.R., 2015. “Open” disclosure of innovations, incentives and follow-on reuse: Theory on processes of cumulative innovation and a field experiment in computational biology. Research Policy 44, 4–19. https://doi.org/10.1016/j.respol.2014.08.001

Brunswicker, S., & Mukherjee, S. (2023). The microstructure of modularity in design: A design motif view. Industrial and Corporate Change, 32(1), 234–261. https://doi.org/10.1093/icc/dtac051

Cano-Kollmann, M., Hamilton, R.D., III, Mudambi, R., 2017. Public support for innovation and the openness of firms’ innovation activities. Industrial and Corporate Change 26, 421–442. https://doi.org/10.1093/icc/dtw025

Chesbrough, H., n.d. GE’s Ecomagination Challenge: An Experiment in Open Innovation - Henry Chesbrough, 2012 [WWW Document]. URL https://journals-sagepub-com.ezproxy.lib.purdue.edu/doi/abs/10.1525/cmr.2012.54.3.140 (accessed 11.7.22).

Croissant, Y., 2013. mlogit: multinomial logit model. URL: http://CRAN.R-project.org/package=mlogit

Crowston, K., Wei, K., Howison, J., Wiggins, A., 2012. Free/Libre Open-source Software Development: What We Know and What We Do Not Know. ACM Comput. Surv. 44, 7:1-7:35. https://doi.org/10.1145/2089125.2089127

Dahlander, L., Magnusson, M., 2008. How do Firms Make Use of Open Source Communities? Long Range Planning 41, 629–649. https://doi.org/10.1016/j.lrp.2008.09.003

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

DiBona, C., Ockman, S., 1999. Open Sources: Voices from the Open Source Revolution. O'Reilly Media, Inc.

Esarey, J., 2017. Package 'clusterSEs.'

Faraj, S., Jarvenpaa, S.L., Majchrzak, A., 2011. Knowledge Collaboration in Online Communities. Organization Science 22, 1224–1239. https://doi.org/10.1287/orsc.1100.0614

Franke, N., Shah, S., 2003. How communities support innovative activities: an exploration of assistance and sharing among end-users. Research Policy 32, 157–178. https://doi.org/10.1016/S0048-7333(02)00006-9

Gächter, S., von Krogh, G., Haefliger, S., 2010. Initiating private-collective innovation: The fragility of knowledge sharing. Research Policy 39, 893–906. https://doi.org/10.1016/j.respol.2010.04.010

Germonprez, M., Kendall, J.E., Kendall, K.E., Mathiassen, L., Young, B., Warner, B., 2016. A Theory of Responsive Design: A Field Study of Corporate Engagement with Open Source Communities. Information Systems Research. https://doi.org/10.1287/isre.2016.0662

Gerrit, 2017. Gerrit Code Review. URL https://www.gerritcodereview.com/

Ghosh, R. (2006). CODE: Collaborative Ownership and the Digital Economy. MIT Press.

Ghosh, R.A., Glott, R., Krieger, B., Robles, G., 2002. Free/libre and open source software: Survey and study. Part iv:" Survey of developers". http://www. infonomics.nl/FLOSS/report/FLOSS_Final4. pdf.

Gruber, M., MacMillan, I.C., Thompson, J.D., 2013. Escaping the Prior Knowledge Corridor: What Shapes the Number and Variety of Market Opportunities Identified Before Market Entry of Technology Start-ups? Organization Science 24, 280–300. https://doi.org/10.1287/orsc.1110.0721

Haefliger, S., von Krogh, G.F., Spaeth, S., 2008. Code Reuse in Open Source Software Development. Management Science 54, 180–193. http://dx.doi.org/10.1287/mnsc.1070.0748

Harhoff, D., Henkel, J., von Hippel, E., 2003. Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations. Research Policy 32, 1753–1769. https://doi.org/10.1016/S0048-7333(03)00061-1

Henkel, J., 2006. Selective revealing in open innovation processes: The case of embedded Linux. Research Policy 35, 953–969. https://doi.org/10.1016/j.respol.2006.04.010

Herron, M. C., & Quinn, K. M. (2016). A Careful Look at Modern Case Selection Methods. Sociological Methods & Research, 45(3), 458–492. https://doi.org/10.1177/0049124114547053

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

Howison, J., Crowston, K., 2014. Collaboration Through Open Superposition: A Theory of the Open Source Way. MIS Quarterly 38, 29–50.

Jeppesen, L.B., Lakhani, K.R., 2010. Marginality and Problem-Solving Effectiveness in Broadcast Search. Organization Science 21, 1016–1033. https://doi.org/10.1287/orsc.1090.0491

Konrad, A., 2015. Intel Leads $100 Million Bet On Mirantis' OpenStack Cloud Software Efforts - Forbes.

Lakhani, K.R., von Hippel, E., 2003. How open source software works: "free" user-to-user assistance. Research Policy 32, 923–943. https://doi.org/10.1016/S0048-7333(02)00095-1

Lerner, J., Tirole, J., 2002. Some simple economics of open source. Journal of Industrial Economics 50, 197–234.

Levine, S.S., Prietula, M.J., 2014. Open Collaboration for Innovation: Principles and Performance. Organization Science 25, 1414–1433. https://doi.org/10.1287/orsc.2013.0872

Long, J.S., Freese, J., 2003. Regression models for categorical dependent variables using stata. Stata Press, College Station, TX.

Majchrzak, A., Malhotra, A., 2016. Effect of Knowledge-Sharing Trajectories on Innovative Outcomes in Temporary Online Crowds. Information Systems Research 27, 685–703. https://doi.org/10.1287/isre.2016.0669

Moody, G., 2002. Rebel Code: Linux And The Open Source Revolution, 1st edition. ed. Basic Books, Cambridge, Mass.

Murthy, R. K., & Madhok, A. (2021). Overcoming the Early-stage Conundrum of Digital Platform Ecosystem Emergence: A Problem-Solving Perspective. Journal of Management Studies, 58(7), 1899–1932. https://doi.org/10.1111/joms.12748

OpenStack, 2015a. distros » OpenStack Open Source Cloud Computing Software [WWW Document]. URL http://www.openstack.org/marketplace/distros/ (accessed 12.14.15).

OpenStack, 2015b. Wiki of OpenStack [WWW Document]. URL https://wiki.openstack.org/wiki/Main_Page (accessed 12.7.15).

OpenStack, 2016. Foundation » OpenStack Open Source Cloud Computing Software [WWW Document]. URL http://www.openstack.org/foundation/ (accessed 1.4.16).

Parnas, D.L., 1972. Information Distribution Aspects of Design Methodology, Information Processing. Carnegie-Mellon University.

Price, A. (2016, December 21). Healthcare application wins grand prize at Hackathon. Superuser. https://superuser.openinfra.org/articles/healthcare-application-wins-grand-prize-at-hackathon/

Raymond, E., 1999. The cathedral and the bazaar. Know Techn Pol 12, 23–49. https://doi.org/10.1007/s12130-999-1026-0

Shah, S.K., 2006. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. Management Science 52, 1000–1014. https://doi.org/10.1287/mnsc.1060.0553

Simon, H.A., 1968. The sciences of the artificial. MIT Press.

Sotiriadis, S., Bessis, N., Amza, C., & Buyya, R. (2016). Vertical and horizontal elasticity for dynamic virtual machine reconfiguration. IEEE Transactions on Services Computing, (99), 1-1.

Spaeth, S., Stuermer, M., von Krogh, G., 2010. Enabling knowledge creation through outsiders: towards a push model of open innovation. International Journal of Technology Management 52, 411–431. https://doi.org/10.1504/IJTM.2010.035983

Spaeth, S., von Krogh, G., Fang He, 2015. Perceived Firm Attributes and Intrinsic Motivation in Sponsored Open Source Software Projects. Information Systems Research 26, 224–237. https://doi.org/10.1287/isre.2014.0539

Stallman, R., 2002. Free Software, Free Society: Selected Essays of Richard M. Stallman. Lulu.com.

Stanko, M.A., 2016. Toward a Theory of Remixing in Online Innovation Communities. Information Systems Research 27, 773–791. https://doi.org/10.1287/isre.2016.0650

Stewart, K.J., Ammeter, A.P., Maruping, L.M., 2006. Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects. Information Systems Research 17, 126–144.

Stewart, K.J., Gosain, S., 2006. The Impact of Ideology on Effectiveness in Open Source Software Development Teams. MIS Quarterly 30, 291–314.

Subramanian, S. (2018). What's HP's cloud strategy? Part 1: HP Helion Overview. Medium. https://clouddon.ai/whats-hp-s-cloud-strategy-part-1-hp-helion-overview-68bec1d65a73

Szumilas, M., 2010. Explaining Odds Ratios. J Can Acad Child Adolesc Psychiatry 19, 227–229.

Teece, D.J., 1986. Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. Research Policy 15, 285–305. https://doi.org/10.1016/0048-7333(86)90027-2

Toor, S., Toebbicke, R., Resines, M. Z., & Holmgren, S. (2012). Investigating an Open Source Cloud Storage Infrastructure for CERN-specific Data Analysis. 2012 IEEE Seventh International Conference on Networking, Architecture, and Storage. https://doi.org/10.1109/nas.2012.14

Trader, T. (2014, November 5). CERN details OpenStack journey. HPCwire. https://www.hpcwire.com/2014/11/04/cern-details-openstack-journey/

Von Hippel, E. (1982). Appropriability of innovation benefit as a predictor of the source of innovation. Research Policy, 11(2), 95–115. https://doi.org/10.1016/0048-7333(82)90037-3

von Hippel, E. (2007). Horizontal innovation networks—By and for users. Industrial and Corporate Change, 16(2), 293–315. https://doi.org/10.1093/icc/dtm005

von Hippel, E., 1976. The dominant role of users in the scientific instrument innovation process. Research Policy 5, 212–239. https://doi.org/10.1016/0048-7333(76)90028-7

von Hippel, E., 1978. Successful Industrial Products from Customer Ideas. Journal of Marketing 42, 39–49. https://doi.org/10.2307/1250327

von Hippel, E., 1986. Lead Users: A Source of Novel Product Concepts. Management Science 32, 791–805. https://doi.org/10.1287/mnsc.32.7.791

von Hippel, E., 1994. "Sticky Information" and the Locus of Problem Solving: Implications for Innovation. Management Science 40, 429–439. https://doi.org/10.1287/mnsc.40.4.429

von Hippel, E., 2006. Democratizing Innovation. The MIT Press.

von Hippel, E., von Krogh, G., 2016. CROSSROADS—Identifying Viable "Need–Solution Pairs": Problem Solving Without Problem Formulation. Organization Science 27, 207–221. https://doi.org/10.1287/orsc.2015.1023

von Krogh, G., Haefliger, S., Spaeth, S., & Wallin, M. W. (2012a). Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. MIS Quarterly, 36(2), 649–676. https://doi.org/10.2307/41703471 von Krogh, G., Rossi-Lamastra, C., & Haefliger, S. (2012b). Phenomenon-based Research in Management and Organisation Science: When is it Rigorous and Does it Matter? Long Range Planning, 45(4), 277–298. https://doi.org/10.1016/j.lrp.2012.05.001

West, J., & Gallagher, S. (2006). Challenges of open innovation: The paradox of firm investment in open-source software. R&D Management, 36(3), 319–331. https://doi.org/10.1111/j.1467-9310.2006.00436.x

West, J., 2003. How open is open enough? Melding proprietary and open source platform strategies. Research Policy 32, 1259.

Zeitlyn, D., 2003. Gift economies in the development of open source software: anthropological reflections. Research Policy 32, 1287. https://doi.org/10.1016/S0048-7333(03)00053-2