Social Exchange and the Reciprocity Roller Coaster: Evidence from the Life and Death of Virtual Teams

Working Paper

Author(s):
Hergueux, Jérôme; Henry, Emeric; Benkler, Yochai; Algan, Yann

Publication date:
2022-02

Permanent link:
https://doi.org/10.3929/ethz-b-000533399

Originally published in:
Center for Law & Economics Working Paper Series 2022(03)

ETH zürich

Center for Law & Economics Working Paper Series

Number 03/2022

Social Exchange and the Reciprocity Roller Coaster: Evidence from the Life and Death of Virtual Teams

Jérôme Hergueux
Emeric Henry
Yochai Benkler
Yann Algan

February 2022

To cite this paper :

Hergueux, Jérôme, Emeric Henry, Yochai Benkler and Yann Algan (forthcoming). Social Exchange and the Reciprocity Roller Coaster: Evidence from the Life and Death of Virtual Teams, *Organization Science*.

https://doi.org/10.1287/orsc.2021.1515

Social Exchange and the Reciprocity Roller Coaster: Evidence from the Life and Death of Virtual Teams

Jerome Hergueux* Emeric Henry† Yochai Benkler‡ Yann Algan§

July 2021

Abstract

Organizations are riddled with cooperation problems, i.e., instances in which workers need to *voluntarily* exert effort to achieve efficient collective outcomes. To sustain high levels of cooperation, the experimental literature has demonstrated the centrality of reciprocal preferences, but has also overlooked some of its negative consequences. In this paper, we ran lab-in-the-field experiments in the context of open source software development teams to provide the first field evidence that highly reciprocating groups are not necessarily more successful in practice. Instead, the relationship between high reciprocity and performance can be more accurately described as U-shaped. Highly reciprocal teams are generally more likely to fail, and only outperform other teams conditional on survival. We use the dynamic structure of our data on field contributions to demonstrate the underlying theoretical mechanism. Reciprocal preferences work as a catalyst at the team level: they reinforce the cooperative equilibrium in good times, but also make it harder to recover from a negative signal (the project dies). Our results call into question the idea that strong reciprocity can shield organizations from cooperation breakdowns. Instead, cooperation needs to be dynamically managed through relational contracts.

Keywords: Cooperation; Reciprocity; Social Exchange; Organizational Behavior; Virtual Teams; Open Source Software.

*French National Center For Scientific Research (CNRS, BETA lab), Berkman Klein Center for Internet & Society at Harvard University, and ETH Zurich Center for Law and Economics. e-mail: `jerome.hergueux@gess.ethz.ch`

†Sciences Po, Department of Economics. e-mail: `emeric.henry@sciencespo.fr`

‡Harvard Law School and Berkman Klein Center for Internet & Society at Harvard University. e-mail: `ybenkler@law.harvard.edu`

§Sciences Po, Department of Economics. e-mail: `yann.algan@sciencespo.fr`

1 Introduction

Public goods problems are ubiquitous within organizations.[1] Because employment contracts cannot enumerate (or even predict) the myriad of cooperation opportunities that may arise within organizations, workers often need to cooperate *voluntarily* (i.e., at a cost to themselves) in order to achieve efficient collective outcomes. In most industries, successful organizations are the ones in which individuals readily cooperate by sharing relevant information and knowledge, helping each other, and collectively engaging in problem solving.[2]

Over the past decades, however, experimental research has shown that cooperative equilibria are typically unsustainable within organizations (Ledyard, 1994; Chaudhuri, 2011). The reason for this is that most individuals endorse the norm of reciprocity in social dilemma situations (Gouldner, 1960; Dufwenberg and Kirchsteiger, 2004; Sobel, 2005). Reciprocal individuals behave as "conditional cooperators": they are willing to cooperate as long as they think that others will respond in kind (see Chaudhuri (2011) for a survey). A major consequence of this preference is that cooperation is highly fragile within organizations (Fischbacher et al., 2001; Fischbacher and Gächter, 2010). Reciprocal workers withdraw their willingness to cooperate whenever they feel like their co-workers are not doing their "fair share." In repeated experiments, this triggers a self-reinforcing process that eventually leads to the breakdown of cooperation (see Ledyard (1994) for an extensive review).

To sustain cooperation at the group level, experimental research has identified two classes of mechanisms aimed at enforcing strong reciprocity: decentralized monetary punishment (Fehr and Gachter, 2000; Gächter et al., 2008), and screening through peer selection or fiat (Page et al., 2005; Cinyabuguma et al., 2005; Charness and Yang, 2014). The message of this literature is that the cooperation breakdown problem can be solved (i.e., a cooperative equilibrium reached) by mechanisms that enable strong reciprocity at the group level, such as designing employment contracts that screen non-reciprocal workers (Kosfeld and Von Siemens, 2011; Bartling et al., 2012).

In this paper, we argue that strong reciprocity does not shield organizations from the cooperation breakdown problem. The reason is quite simple: to the contrary of most laboratory ex-

[1]By "organization", we mean any group or team of individuals who work together towards some shared purpose. This could be a nation, a firm, a not-for-profit organization, or an open innovation community (e.g., a research team).

[2]Among many other examples, see Gittell (2000) and Gittell et al. (2004) for the U.S. airlines industry, Ichniowski et al. (1997) for the steel industry, Dyer and Nobeoka (2000) and Helper and Henderson (2014) for the automobile industry, and Gittell et al. (2010) for the health care sector.

periments,[3] the details of the cooperative game played by organization members are usually not common knowledge. In the field, individuals need to learn the details of the cooperative game they play as they work together. This includes the set of possible actions available to each player in various (and possibly unforeseen) contingencies, together with their associated costs and organizational benefits. Those parameters are typically heterogeneous across group members and subject to change over time.

The consequence is straightforward: individuals in the field receive noisy signals of the willingness of their peers to cooperate. Those signals must then be interpreted, notably in the light of the history of previous interactions within the organization – otherwise referred to as "norms" or "culture" (Kreps, 1990). Because reciprocity is both the key to sustained cooperation and the main mechanism through which it unravels (Fischbacher and Gächter, 2010; Chaudhuri, 2011), highly reciprocating groups may not always be more successful in practice. Over the course of the relationship, even highly reciprocating teams may receive negative signals that can lead to the unraveling of cooperation. In other words, strong reciprocal preferences allow organizations to sustain efficient cooperative equilibria, leading to above-average performance, but they also make them *more* sensitive to the cooperation breakdown problem (in the case of bad news).

This paper tests these ideas in the field context of Sourceforge, a large open innovation platform hosting virtual teams that seek to develop novel open source software (OSS) products. This environment is particularly well suited to our research purposes. OSS projects tend to be complex and uncertain, and largely rely on voluntary code contributions from team members in order to thrive. At the same time, the resulting software product is freely available for anyone to use. Team members therefore face a repeated public goods dilemma in which they receive (noisy) signals that reflect their teammates' past contributions, and have to decide how much they want to contribute to the project, if at all, in the current period.

In 2011, we contacted a sample of 2,534 open source developers registered with Sourceforge to participate in an online experiment. We collected lab data on 1,194 subjects (i.e., a 47% participation rate), and used their conditional contribution decisions in a public goods game to construct a lab measure of reciprocal preferences. In parallel, we collected field data on all the projects that our experimental subjects *and* their collaborators had ever contributed to. This yielded a final

[3] A few exceptions include Bereby-Meyer and Roth (2006), Kunreuther et al. (2009), Xiao and Kunreuther (2016), Grechenig et al. (2010) and Ambrus and Greiner (2012). In general, those experimental papers show that sustaining a cooperative equilibrium is more difficult in uncertain environments.

sample of 5,557 projects, many of which had failed, and involving 10,537 developers overall. For each project, we extracted the monthly number of code contributions (or "commits") made by each developer.

An important feature of field research is that it is only possible to recruit subjects from active organizations at the time of the experiment. In order to expand our research sample in the direction of previously failed projects and their members, we use our lab measure of reciprocity to validate a generalizable measure of reciprocity, computed on the base of the historical dynamics of project contributions between developers. We then use this field measure to expand our research sample in the direction of previously failed projects and their members. This strategy allows us to separately analyze the determinants of failure (i.e., the project died) from that of performance (measured through Sourceforge's flagship indicator).

The paper makes two main contributions to the literature on cooperation. First, we show that organizations with a higher share of reciprocal members are not necessarily more successful. Instead, the relationship between high reciprocity and success can be more accurately described as U-shaped. Highly reciprocal teams are significantly more likely to fail, and only outperform other teams conditional on survival. Second, we analyze the dynamics of field contributions in the data to identify the underlying micro-level mechanism. Reciprocal preferences work as a catalyst at the organizational level. They reinforce the cooperative equilibrium (foster increased contribution levels) when team members receive positive signals. However, they also accelerate the slowdown when they receive negative ones – leading to a lower probability of recovery after a period of inactivity (the project fails).

Our field results suggest that strategies aimed at screening workers according to their social type (through, e.g., sorting (Page et al., 2005) or contract design (Bartling et al., 2012)) are unlikely to generate a self-sustained cooperative equilibrium within organizations. Instead, future experimental and theoretical research on cooperation should focus on how organizations can dynamically sustain a cooperative equilibrium in the face of adverse shocks and imperfect monitoring. To do this, they must define the implicit norms that govern behavior among team members since those implicit norms, or "relational contracts", allow workers to make sense of the signals they receive and, ultimately, to avoid inefficient cooperation breakdowns (Chassang, 2010; Acemoglu and Jackson, 2015).

2 Related literature

2.1 The Lab-in-the-field literature

The early literature, which attempted to identify individual preferences, beliefs and institutional designs that encourage voluntary cooperation, relied on laboratory experiments (Ledyard, 1994; Roth, 1995). One of its most robust findings is that cooperation gradually decays in repeated experiments. While most individuals initially make non-zero contributions, their willingness to cooperate steadily declines with repetition (see Chaudhuri (2011) for a survey). This finding follows from the fact that many individuals exhibit reciprocal preferences: they are "conditional cooperators" (i.e., willing to cooperate as long as others do so as well), hence the positive initial contribution levels. However, because reciprocators have a preference for matching the contributions they observe or expect from others, cooperation will inevitably collapse if: (i) the group contains free-riders, who never cooperate in order to maximize their private payoffs; (ii) the group contains "weak" reciprocators, who behave as conditional cooperators, but with a "self-serving bias" (Fischbacher et al., 2001) (i.e., they contribute, but generally less than others); and (iii) some group members have reasons to believe that others will reduce their contributions in the future.

The major takeaway from this lab-based literature is that even though most individuals are not selfish, cooperation is highly fragile (Fischbacher and Gächter, 2010). In particular, reciprocal preferences are crucial to sustain cooperation within organizations, but require institutions that either discipline non-reciprocating types (through monetary and non-monetary punishment), or provide a mechanism for excluding them from the group (Fehr and Gächter, 2000; Chaudhuri, 2011; Bartling et al., 2012).

More recently, the literature has questioned the field (referred to as "external" or "ecological") validity of these lab-based results. Since reciprocal preferences are hard to identify in the wild, empirical papers increasingly rely on "lab-in-the-field" designs (see Gneezy and Imas (2017) for a detailed description). This fast-growing literature relies on validated lab paradigms to elicit psychological traits or preferences, and links these measures to outcomes of theoretical interest in the field. In other words, the spirit of lab-in-the-field experiments is to match individuals' behaviors in experimental games (the lab) with choices by the same individuals in the field and/or the performance of the organizations they work in. This approach is a powerful tool for organizational researchers interested in microfounding the processes that underlie aggregate outcomes in the field (Felin and Foss, 2005; Felin et al., 2015; Bitektine et al., 2018).

Initial lab-in-the-field studies linked behavior in the lab to individual behaviors and outcomes in field environments, without focusing on group-level outcomes. In a seminal paper in economics, Karlan (2005) obtained experimental measures of reciprocity at the individual level and shows that they predict loan repayment among participants in a microcredit program. Charness and Villeval (2009) show that senior workers are typically more cooperative than junior ones in a standard public goods game. Fehr and Leibbrandt (2011) and Leibbrandt (2012) conducted a public goods game among Brazilian shrimp catchers and sellers, and show that more cooperative subjects are less likely to engage in over-extraction, and achieve better market outcomes. Kosfeld and Rustagi (2015) show that the way traditional leaders "punish" players in their community based on how they behave in a public goods game predicts how successfully they cooperate in the field. [4]

However, there are fewer lab-in-the-field studies devoted to the link between reciprocal preferences and group-level outcomes. The first paper to study this question in a field setting was Anthony (2005), who did not adopt a lab-in-the-field approach per se, but instead relied on survey answers. The paper measures reciprocal behavior in 106 microcredit borrowing groups in the U.S. through a number of survey questions answered by randomly selected group members. It finds reciprocity to be the variable the most closely associated with low levels of loan delinquency and higher group longevity. Barr and Serneels (2009) were the first to obtain experimental measures of reciprocal preferences from a sample of workers in 20 Ghanaian manufacturing firms, which they then coupled with survey-based data on workers' individual wages and aggregate firm productivity. They found that reciprocal workers generally earn higher wages, and reported a strong firm-level relationship between reciprocating behavior and aggregate productivity. Rustagi et al. (2010) studied 49 local groups that participated in a publicly-funded forest conservation program

[4]There is a similar literature in political science: Finan and Schechter (2012) experimentally elicit reciprocal preferences in a population of community leaders in Paraguay and show that highly reciprocal village chiefs are more likely to be targeted by politicians for vote-buying purposes. Similarly, Baldassarri and Grossman (2011) and Grossman and Baldassarri (2012) demonstrate that cooperation in a repeated public goods game where a "leader" has the ability to punish group members based on past contributions predicts field cooperation among Ugandan farmers, but only when the leader is elected by the people – which corresponds to the way chiefs are appointed in this field setting. Gilligan et al. (2014) use exogenous variation in the extent to which local communities in Nepal were affected by civil war to show that stronger exposure to violence can lead to collective coping through social cohesion, as measured by subjects' behavior in a standard public goods game. In a related paper, Blair (2018) runs lab-in-the-field experiments in Liberia to show that exposure to war-time violence increases the government's ability to instruct citizens to make voluntary contributions to public goods.

in Ethiopia where they were responsible for maintaining and cultivating the forest (the "public good"). They elicited reciprocal preferences using a conditional public goods game (Fischbacher et al., 2001), and notably measured the share of "conditional" and "weakly conditional" cooperators in each group, which they then associated with an independently collected measure of success in forest commons management. They found that groups with a larger share of highly reciprocating types are generally more successful. Similarly, Carpenter and Seki (2011) use the public goods game to elicit reciprocal preferences from Japanese fishermen. Even though their sample only contains 12 fishing crews, they found that those that exhibited higher levels of reciprocity were generally more productive.

Taken together, those papers reveal a clear empirical result: groups composed of more reciprocal types, as measured by incentivized lab experiments, achieve significantly better outcomes. The evidence comes from a variety of field settings where members face social dilemmas, such as forest commons management, profit maximizing firms and financial markets. This unambiguous relationship between reciprocity and group success is surprising in light of the lab literature on social dilemma. In this literature, reciprocal preferences drive cooperation levels down whenever conditional cooperators observe (or believe) that some team members are not doing their "fair share".

We add to this literature by highlighting a potential limitation of the lab-in-the-field methodology, which may mechanically produce the strong positive relationship observed between reciprocity and organizational success. The key to lab-in-the-field designs is to recruit the experimental subjects directly from their work environment. We argue that this introduces a bias, at both the individual and organizational level. Failed organizations are much less likely to be sampled since the members are no longer present on the research site. More generally, for organizations that still exist, the propensity of members to participate in the experiment might be strongly correlated with the level of activity of the project. This sampling bias is likely to be particularly severe in field settings where an inability to sustain cooperation can more easily lead to the death of the organization (e.g., in competitive markets, innovation-driven activities or volunteer communities).

To address this concern, we propose an additional way to use lab data in the field: we use the lab data as a benchmarking tool for subjects' reciprocity motives. We then propose a generalizable field measure of reciprocity (inspired by the experimental game) and show that the lab and field measures are strongly correlated. To build our field measure, we leverage the panel structure of our data and measure developers' reactions to the past observed contribution decisions of their teammates (see Section 4 for details). Because this field measure is computed based on archival data, we can use it to expand our research sample in the direction of previously failed organizations.

2.2 Reciprocity and social exchange theory

At a theoretical level, our paper is related to the literature on social exchange theory (SET). Since the early contributions of Homans (1961), Blau (1964) and Emerson (1976), SET has been a highly influential theoretical construct in social psychology and organizational behavior. More precisely, our experimental and field measures of reciprocity are directly linked to the most paradigmatic instance of social exchange, known as "reciprocal exchange" (Cropanzano and Mitchell, 2005). In this framework, reciprocity can be seen as a social norm (Gouldner, 1960) whereby an individual receiving an unconditional benefit from another party feels bound to respond in kind.[5] Positive reciprocity dynamics therefore lead to a form of mutual commitment resulting in a virtuous, self-reinforcing cycle of cooperation.

Much of the existing literature to date has relied on laboratory experiments to test these predictions (see Cook and Emerson (1978), Montgomery (1996), Molm et al. (2000), Molm (2003), as well as Cook et al. (2013) for a review). Our results add to this literature by exploring how the dynamics of reciprocal exchange occur in an online generalized exchange system such as OSS, where peers intend to co-produce a public good based on voluntary contributions. We take advantage of the fact that OSS teams differ in the extent to which their members endorse reciprocity. This allows us to link group-level reciprocity to an objective measure of organizational success. By combining lab-in-the-field experiments with the analysis of behavioral data over time, we hope to convince organizational researchers that field research can achieve relatively high levels of internal validity, as well as ecological relevance (Schram, 2005). We argue that this is especially true in computerized environments where researchers can gather a significant amount of data on field behavior (Lazer et al., 2009), while retaining the ability to conduct controlled experiments (Hergueux and Jacquemet, 2015).

[5] The formal definition of reciprocity is strikingly similarly in economics (see Sobel (2005), Dufwenberg and Kirchsteiger (2004) and Falk and Fischbacher (2006)). Furthermore, the result that individuals differ in how strongly they endorse the norm of reciprocity, obtained in the context of the experimental literature on the decay of cooperation in repeated public goods experiments (Fischbacher et al. (2001); Fischbacher and Gächter (2010)), had previously been established in the context of lab-based tests of SET (see Eisenberger et al. (1987)).

Note that our work should be set apart from the important literature on negative reciprocity. In a recent survey of the literature on social exchange, Cropanzano et al. (2017) note that SET actually "fails to distinguish the presence of negative constructs (e.g., abuse) from the absence of positive constructs (e.g., support)." They argue that this leads to some confusion in terms of behavioral predictions: negative behavior is predicted to lead to negative reciprocal responses (i.e., negative reciprocity), whereas the *absence* of positive behavior should, instead, lead to a lack of positive response (i.e., the decay of cooperation). Our empirical results illustrate the relevance of this distinction for SET: in the face of relative inactivity, reciprocal developers adjust their own cooperation level downwards, and eventually stop contributing. Since the ability to "punish" non-reciprocating types is altogether absent from our field of study (see Section 3.1), our paper is not related to the literature on negative reciprocity. In this respect, our field setting is most closely related to lab designs where subjects can select their teammates based on observed past behavior, which has been found to have a dramatic impact on their ability to sustain very high levels of cooperation over time (see Page et al. (2005), Cinyabuguma et al. (2005) and Charness and Yang (2014)).[6]

2.3 Relational contracts

At a theoretical level, our paper also connects to a related but distinct literature on relational (or "implicit") contracts in organizational economics. Chassang (2010) studied how agents can develop a successful cooperative relationship when the details of cooperation are not common knowledge. In his model, teammates can observe agents' actions and their associated payoffs. However, there is uncertainty regarding which actions are "cooperative" since they do not always yield the intended organizational benefits. Relationships are therefore quite sensitive to adverse shocks, and may only become resilient after significant "relational knowledge" has been built among the parties involved. The concept of relational knowledge implies that a workable subset of cooperative actions could be identified and developed as a routine at the organizational level. Because this routine develops as a function of the agents' previous histories, random events occurring during the relationship can lead to unexpected cooperation breakdowns and/or have a

[6]Note that the possibility of decentralized punishment does not necessarily lead to enhanced cooperation and welfare, especially in noisy environments such as those considered in this paper (see Grechenig et al. (2010) and Ambrus and Greiner (2012)). See also the designs in which punished subjects decide to engage in welfare reducing "anti-social punishment" (Denant-Boemont et al., 2007; Nikiforakis, 2008; Nikiforakis et al., 2012; Herrmann et al., 2008).

lasting impact on the way players approach cooperation. In a related paper, Gibbons and Henderson (2012) argue that the relational contracts that sustain cooperation in the field have to solve the twin problem of credibility (i.e., the misalignment of incentives within the organization) and clarity (i.e., the uncertainty over which actions are collectively considered “cooperative” for a given agent in a given situation). Such informal cooperative agreements take time to develop and are highly path-dependent (see also Acemoglu and Jackson (2015)).[7]

Qualitative research has shown that organizations dedicate enormous resources to building and maintaining relational contracts (Mayer and Argyres, 2004; Kellogg, 2009). Unfortunately, the existing experimental literature provides very little guidance regarding the type of informal norms that are most likely to sustain an efficient cooperative equilibrium at the organizational level, with two notable exceptions. Fudenberg et al. (2012) studied a repeated public goods game where intended actions are implemented with noise (i.e., uncertain intentions). They experimentally show that “lenient” (i.e., not retaliating after the first signal of defection) and “forgiving” (i.e., returning to cooperation after retaliation) strategies are most efficient in this case. More recently, Gibbons et al. (2020) studied a repeated bilateral trade relationship where the state of the world is subject to exogenous shocks (i.e., uncertain environment). They found that, in such environments, relational contracts based on general principles perform better than those that adhere to specific rules, but that high-performing relational contracts are typically difficult to build. By empirically showing that screening workers according to their social type is unlikely to eliminate the cooperation breakdown problem, we hope to encourage more experimental research of this sort.

3 Setup and data collection

3.1 Open source software

To set the stage, we provide some background information on open source software. OSS currently mobilizes millions of loosely-connected developers from around the world who self-organize in virtual teams to develop software products (Faraj et al., 2011; Levine and Prietula, 2013). OSS is responsible for most of the basic utilities on which the Internet runs (e.g., the Apache Web server), popular programming languages (e.g., Python, R) and programming environments (e.g., Eclipse).

[7] Because of this, relational contracts are also very difficult to imitate and can represent a significant source of competitive advantage. See Gibbons and Henderson (2012) for a discussion of relational contracts at Lincoln Electric, Toyota and Merck, as well as Helper and Henderson (2014) for General Motors.

It also competes with many of its proprietary counterparts in the realm of end-user applications (e.g., Android), operating systems (e.g., Linux), and Web browsers (e.g., Firefox). At present, most businesses and public organizations rely on OSS for their daily activities (Walli et al., 2005; Ghosh, 2007; Greenstein and Nagle, 2014).

Apart from the above-mentioned projects, which are both very large and quite well-known, hundreds of thousands of smaller-scale OSS projects are hosted by online platforms such as Sourceforge, which was dominant at the time of our study, and, more recently, Github. These platforms provide developers with a set of free standard online tools for collaborative software development (e.g., a code versioning system, a bug tracker). Any developer can initiate a software project on such platforms, and the source code of each project is readily available for anyone to see and modify. Projects are therefore developed in the context of geographically distributed virtual teams that coordinate their activities in the absence of formal leadership, pre-specified design rules or markets (Benkler, 2002; Hippel and Krogh, 2003; Von Krogh and Von Hippel, 2006). Contributors typically resolve potential disagreements over future developments through discussion, and, in some rare cases, through "forking", whereby some team members decide to split off and develop their own version of the project. As a result, OSS is usually seen as a "technical meritocracy" (Scacchi, 2007) where developers typically acquire influence by contributing elegant code that "just works" (Weber, 2004; Marlow et al., 2013). Similar to fundamental research, OSS development has thus been modeled as an evolutionary learning process, driven by peer criticism and error correction (Lee and Cole, 2003).

Because developers need to invest time and effort contributing to projects that are made freely available for anyone to use, OSS has been described as a privately-produced public good (O'Mahony, 2003) where developers reveal their code in the expectation that others will reciprocate (Maurer and Scotchmer, 2006). About 50% of OSS developers are volunteers who only contribute in their free time, while the other half derives either direct or indirect revenue from their contributions (Hertel et al., 2003; Lakhani et al., 2005). In the latter case, the developer can be paid by a firm to dedicate working hours to a project that serves corporate goals (Dahlander and Magnusson, 2005). Some innovation-heavy firms (e.g., Google) also allow their employees to dedicate working hours to any project of their choosing, on the assumption that developing OSS will (i) allow them to acquire new skills, and (ii) keep them in touch with a fast-moving open innovation community.

3.2 Collecting lab data

In May 2011, 2,534 OSS developers registered with Sourceforge.net were contacted and asked to participate in an online experiment that we describe in more detail in Section 3.2.2. The experimental platform remained active for ten complete days, and 1,194 subjects – a 47% take-up rate – participated. Before describing the details of the experimental procedure, we will begin by describing how the initial sample of 2,534 developers was selected out of the large Sourceforge community that counted 221,802 projects registered in 2010.

3.2.1 Experimental sample selection

To select the initial pool to be contacted, we set up a two-tier selection procedure, first selecting projects and then selecting individuals within these projects. To select the projects, we used two stratification variables: size of project and type of license, as described below. There is great heterogeneity between Sourceforge projects in terms of the number of contributors, and previous research efforts have been somewhat biased towards a handful of large and highly successful projects (Crowston et al., 2012). To avoid this pitfall, the first stratification variable that we considered was the project size, defined as the number of contributors. Second, following Belenzon and Schankerman (2015), who argue that reciprocal developers prefer restrictive project licenses, we used the variable "license restrictiveness" as an additional stratification criterion, making it more likely that we would include diverse cooperative types in our pool.[8]

Specifically, we extracted all the Sourceforge projects that were active in 2010, defined as having either solved a bug or added a feature in 2010. This yielded a sample of 1,242 active projects. Of the 8,858 developers who contributed to those active projects, we identified those who had some development activity in 2010. We then ordered projects according to their number of active developers, and relied on Belenzon and Schankerman (2015)'s classification of the 44 existing OSS license types to label their licensing terms as highly, moderately or weakly restrictive.

Since there were only 83 projects with more than seven active contributors, we selected all of those projects, irrespective of their license terms. For all the projects with six or fewer active contributors, we chose to construct a sample containing an equal number of highly, moderately and weakly restrictive licenses. For instance, out of the 365 projects that had only one active

[8]Two main features define the restrictiveness of a project license: (i) the extent to which the code and any of its modifications can subsequently be embedded in commercial software; and (ii) whether modifications to the code have to remain open source (i.e., free to use, study, share and modify by anyone).

developer, 239 featured highly restrictive licenses, 57 featured moderately restrictive licenses and 69 featured weakly restrictive licenses. We thus retained the 57 projects with moderately restrictive licenses and then randomly selected 57 projects from the pool of projects with both highly and weakly restrictive licenses. We ended up with a sample of 322 active projects, balanced in terms of both size and license restrictiveness. Table 1 lists the number of projects selected by order of size and license type.

For the 322 projects selected, we kept all 1,019 developers who were active in 2010. In addition, we also randomly selected three non-active developers. We ended up with a sample of 2,534 Sourceforge developers eligible to participate in the experiment. Table 1 summarizes the selection procedure.

TABLE 1: CONSTRUCTING THE SAMPLE OF ELIGIBLE SUBJECTS

No. of active developers on project	License restrictiveness	Total no. of projects	No. of projects randomly selected	Active developers randomly selected	Non-active developers randomly selected
1	High	239	57	57	57
1	Moderate	57	57	57	57
1	Low	69	57	57	57
2	High	118	28	56	56
2	Moderate	28	28	56	56
2	Low	29	28	56	56
3	High	51	12	36	36
3	Moderate	12	12	36	36
3	Low	15	12	36	36
4	High	24	6	24	18
4	Moderate	13	6	24	18
4	Low	6	6	24	18
5	High	18	7	35	21
5	Moderate	9	7	35	21
5	Low	7	7	35	21
6	High	18	5	30	15
6	Moderate	5	5	30	15
6	Low	5	5	30	15
7+		83	83	962	249

3.2.2 The online experiment

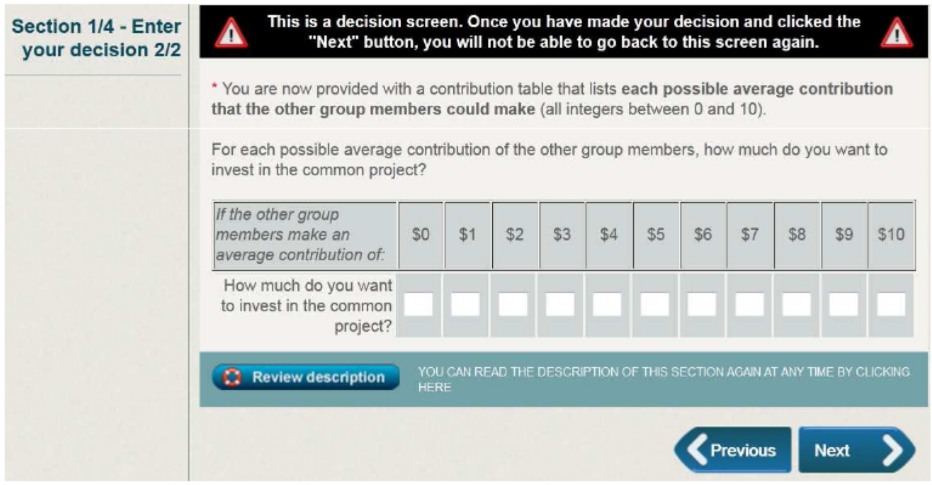
With the support of the Sourceforge platform, we collected the e-mail addresses of all 2,534 selected developers and sent them individual invitations to participate in the experiment. By clicking on a link included in the invitation message, eligible developers were able to log into the system with their Sourceforge username, which allowed us to identify them and subsequently collect their entire history of contributions to OSS. Subjects were then redirected to the welcome screen of the experimental platform.

Our design strictly follows the Internet-specific procedures detailed in Hergueux and Jacquemet (2015). (See Appendix A for further details.) The key experimental game we used to elicit reciprocal preferences is the one-shot public goods game. This game is played in groups of four players, each with an initial endowment of \$10. Group members need to decide how much to contribute to a common project. Each dollar invested in the common project produces \$1.6, which is then equally distributed among group members. Thus, a \$1 investment only yields a private return of \$0.4, but benefits all other members of the group. This design captures the social dilemma faced by open source developers in the field: contributing code to OSS can be individually costly, but is socially efficient. Specifically, for player i who makes a contribution $contrib_i$, the final private payoff is given by:

$$\pi_i = 10 - contrib_i + 0.4 \sum_{j=1}^{4} contrib_j.$$

Following the example of Fischbacher et al. (2001), we elicited two types of contribution decisions: first an unconditional contribution, and then a conditional contribution. For the unconditional contribution, each subject had to decide on his or her contribution in the game described above. For the conditional contribution, each subject determined his or her intended contribution for each possible value (0,1,2, . . . 10) of the average contribution of the three other members of the group. The conditional contributions allowed us to measure the subjects' willingness to behave reciprocally (i.e., to be conditionally cooperative). This design is incentive-compatible since, after the match with other participants has been carried out, one randomly selected decision (i.e., unconditional or conditional) is used to compute the subjects' earnings. The screen eliciting conditional contributions is presented in Figure 1.

FIGURE 1: THE DECISION SCREEN OF THE CONDITIONAL PUBLIC GOODS GAME



We ended up collecting lab data on 1,194 developers out of the pool of 2,534 eligible to participate. Right after the experiment and before payment, we asked subjects for some standard demographic information, i.e., their age, gender, education and salary range. These variables are described in Table 2.

We found that the population of OSS developers who answered our survey is generally young (32-years-old on average) and overwhelmingly male (about 3% of developers report being female). The average developer in our experiment has a 4-year college degree (BA, BS), with 17.5% of the population of developers having a lower qualification than a 2-year college degree and almost half of the population (i.e. 49%) having a Master's degree or a PhD. The average developer earns between \$2000 and \$4000 per month, with 32% of the population earning less than \$2000, and 20% earning more than \$7500. These statics are consistent with survey studies on OSS developers (David and Shapiro, 2008).

Finally, we also ask developers a few questions about their motivations for contributing to OSS. Specifically, we followed the previous literature (see David and Shapiro (2008)) and asked them to

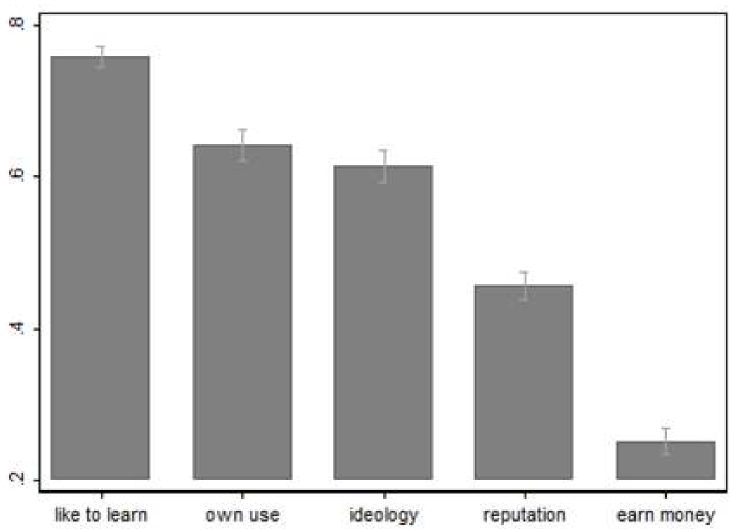
TABLE 2: SUBJECT-LEVEL DESCRIPTIVE STATISTICS

Variable	Obs	Mean	Std. Dev.	Min	Max
Age	1,192	32.20	8.42	16	72
Female	1,194	0.03	0.16	0	1
Education level	1,184	5.29	1.64	1	8
Income level	1,110	4.76	2.24	1	9

Age is measured in years.. Education level: 1 = less than high school; 2 = high school; 3 = some college; 4 = 2-year college degree; 5 = 4-year college degree (BA, BS); 6 = Master's degree; 7 = professional degree (MD, JD); 8 = doctoral degree. Income level (monthly): 1 = 0 USD; 2 = less than 1000 USD; 3 = between 1000 and 2000 USD; 4 = between 2000 and 3000 USD; 5 = between 3000 and 4000 USD; 6 = between 4000 and 5000 USD; 7 = between 5000 and 7500 USD; 8 = between 7500 and 10000 USD; 9 = more than 10000 USD.

state their level of agreement with the following reasons for contributing to OSS, on a scale ranging from 0 ("strongly disagree") to 10 ("strongly agree"): (i) to learn and develop new skills; (ii) to solve a problem that could not be solved by proprietary software; (iii) because I think software should not be a proprietary product; (iv) to build myself a reputation on the OSS developer scene; and (v) to make money. The means of those self-declared motivations are illustrated in Figure 2, together with their 95% confidence interval.

FIGURE 2: DEVELOPERS' REPORTED MOTIVES FOR CONTRIBUTING TO OSS



3.3 Field data

In addition to the lab data, we collected panel data documenting team members' monthly code contributions to individual projects during the period March 2005 – February 2013. We obtained this data from the Sourceforge Research Data Archive (SRDA),[9] a project hosted at the University of Notre Dame that collected monthly data dumps from Sourceforge so as to make them available to the research community. Our panel data ends in February 2013, at which time Sourceforge put an end to its data sharing agreement.

We collected this data for all team members belonging to (i) projects to which our starting set of 1,194 experimental subjects contributed, and (ii) all the other projects their teammates (9,343 co-developers) worked on without their participation. We obtained a final sample of 5,557 OSS projects involving 10,537 developers. By the end of our time period, about half of those projects had failed and died (i.e., we observed no activity in those projects in the final 12 months of our time period).

For each developer, we collected monthly data on the number of code contributions (i.e., "code commits") over those 96 consecutive months. A commit is a set of changes to the source code of a project that makes logical sense (i.e., implements a new feature or solves a bug). In addition, we extracted the creation date of each project and took advantage of the fact that OSS development teams often document the characteristics of their projects on the Sourceforge platform to collect additional project-level information. This includes license restrictiveness, popularity of the programming languages used, working languages used and target user population. (See Table 3 below.)

Finally, we needed to address the challenge of reliably measuring the level of success of OSS projects. Given that OSS projects are made freely available for anyone to use, standard measures of popularity (e.g., sales) cannot be used as a proxy for success. In addition, since projects largely rely on voluntary contributions for their development, measures of input (e.g., code contributions) should be seen as an indicator of success in their own right. As a result, success needs to be defined at the project level as a function of both user popularity (i.e., "use") and community input (Grewal et al., 2006; Crowston and Scozzi, 2002; Crowston et al., 2004; Van Antwerp and Madey, 2010).

We achieved this goal by extracting Sourceforge's own ranking measure: the monthly "activity percentile" of each project, which combines the above dimensions to compute an exogenous,

[9]See `http://Archive, www3.nd.edu/ oss/Data/data.html` and Van Antwerp and Madey (2008).

dynamic measure of success. The activity percentile is automatically calculated by Sourceforge and is prominently displayed on each project summary page. As Van Antwerp and Madey (2010) put it, "projects with a high activity percentile are popular projects since the measure is based on downloads, site views, development activity, and administrator activity."

Specifically, the measure aggregates (i) the size of the project user base, (ii) the intensity of contributors' development activity, and (iii) the use of project-related communication channels (see Appendix D for further details):

$$Activity\ Percentile = \frac{1}{3}\ User\ Traffic + \frac{1}{3}\ Development\ Activity + \frac{1}{3}\ Project\ Communication.$$

We end this section by summarizing our project-level variables in Table 3. The average activity percentile is relatively high in our project sample. This results from the fact that (i) our design tends to focus on a minority of collaboratively authored software projects within Sourceforge, and that (ii) the platform regularly purges its data from zombie projects. The average project in our data was about 5-years-old by the end of our study. The oldest project in our data is 43-years-old (i.e., older than Sourceforge itself), and was probably imported to Sourceforge from another hosting site. We see that projects generally tend to adopt relatively restrictive licenses (mean score of 2.4 out of 3). About a fourth of software projects are targeted at end users as opposed to other developers or system administrators, and 30% of teams use English as their working language. Finally, we counted the log number of teams in our dataset that reported using any available programming language as a measure of its overall popularity in the community of developers. We then built a measure of the popularity of the programming languages used by each project by averaging those popularity scores at the project level.

4 Measuring reciprocity

In this section, we describe the construction of our main explanatory variables, i.e., our lab and field measures of developers' reciprocal preferences. As explained above, lab-in-the-field experiments typically adopt the following methodology. At the time of the experiment, a sample of existing organizations is selected. Volunteers from these organizations participate in experimental games (the lab part) and the preferences elicited in those games are then related to measures of

TABLE 3: PROJECT-LEVEL DESCRIPTIVE STATISTICS

Variable	Obs	Mean	Std. Dev.	Min	Max
Success score ("activity percentile")	5,336	78.137	17.873	19.803	99.991
Project age (years)	5,558	5.36	2.74	0	43.10
License restrictiveness	4,871	2.35	0.75	1	3
Software aimed at end user	5,557	0.24	0.36	0	1
Team works in English	5,557	0.30	0.40	0	1
Mean popularity of programming languages	4,777	2.74	0.45	0	3.24

Project age is measured in years. License restrictiveness ranges from 1 (low restrictiveness) to 3 (high restrictiveness), as in Belenzon and Schankerman (2015). Software projects are considered to be aimed at end users when the team marks them as such (as opposed to being aimed at other developers or system administrators). We consider that a team works in English if English is listed as one of its working languages. Finally, to compute the mean popularity of the programming languages used at the project level, we take the log number of teams in our dataset that report the use of any given programming language as a measure of overall programming language popularity. We then compute the average of these popularity scores at the project level (since many teams report using several programming languages at the same time).

organizational performance (the field part). For organizations that are relatively stable, i.e., less likely to die as a result of bad performance (e.g., a government agency), this is a powerful tool.

However, in the context of OSS as in many others (e.g., private firms operating in competitive markets or volunteer communities), organizations are much more volatile: some might grow, while others may quickly fail and die. Thus, when the experiment is run, only the participants in projects that have not yet died are available to be sampled. More generally, the activity level of a project and, consequently, the effective presence of its members on the research site may impact the likelihood that they participate in the experiment. This would then generate a sampling bias where successful organizations are over-represented compared to those that fail.

To overcome this problem, we propose another way to use our lab data in the field. The lab data serves to validate an analogous field measure of reciprocity based on archival data (or "activity traces"), which we can use in order to capture team members and projects that had already failed by the time of our study. We begin this section by describing our lab measure of reciprocity. We then discuss the construction of our related field measure, before showing how both variables are correlated.

4.1 Experimental measure of reciprocity

As described in Section 3.2.2, our subjects in the public goods game reported both unconditional and conditional contribution decisions. As in Fischbacher et al. (2001)'s seminal paper, we used the conditional contribution decisions to compute a measure of reciprocal preferences at the developer level. We defined this measure as the correlation between the player's conditional contribution decisions and the corresponding average contribution of the three other members of his or her group (from 0 to $10, as illustrated in Figure 1). This variable is distributed with a mean of 0.73 and a standard deviation of 0.45. Note that it can only capture positive reciprocity, not negative reciprocity, since there was no opportunity in the standard public good games we used to pay a personal cost to decrease others' earnings. (The worst one can do to hurt others is to contribute 0, which is also the strategy that maximizes private payoffs.)

While directly inspired by Fischbacher et al. (2001), the experimental measure of reciprocity that we defined was computed as a simple correlation that captures the subject's willingness to be conditionally cooperative in the public goods game (i.e., behave reciprocally). This measure has the benefit of simplicity. It also had a direct analog in our field setting, as we explain in the next section. By comparison, Fischbacher et al. (2001) classify their student subjects into three exclusive categories based on a visual examination of their conditional contribution patterns: (i) free-riders, who never contribute regardless of the contributions of others (this would imply a correlation of zero in our setting); (ii) conditional cooperators, who match the contributions of the other members of their group (this would imply a correlation of 1 in our setting); and (iii) conditional cooperators with a "self-serving bias" or "weak conditional cooperators" (Rustagi et al., 2010), who contribute to the public good, but generally less than others (this would imply a positive correlation of less than 1 in our setting).[10]

4.2 Field measure of reciprocity

The next step in our study was to propose a field measure of reciprocity, inspired by the lab measure, that uses the observed patterns of contributions. The general idea was to measure the cor-

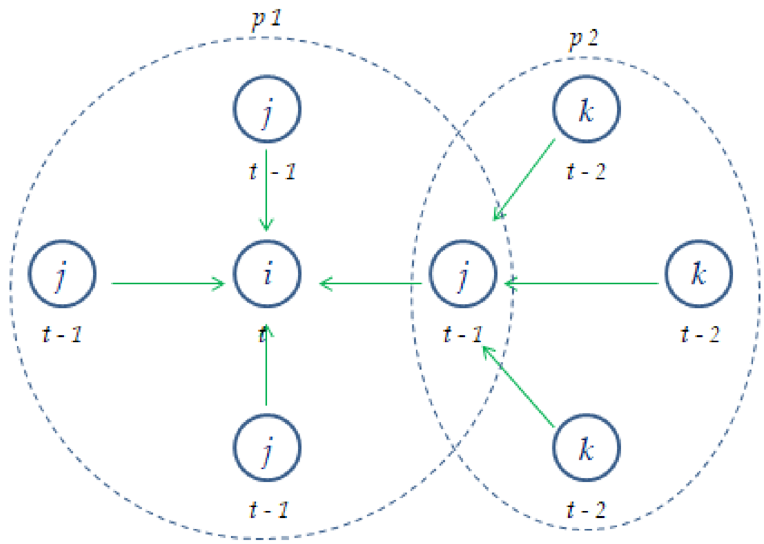
[10] Some differences between our pool of OSS developers and the populations of students typically used in lab experiments are noteworthy. Only 4% of our subjects could be classified as free-riders (compared to 20-30% in student populations), 48% were perfect reciprocators, and 41% were weak reciprocators. Finally, 7% of our subjects unconditionally contributed *all* of their endowment to the public good (an altruistic pattern of contributions that is typically not observed among students).

relation between a participant's contributions in any given month with the sum of contributions made by his or her team members in the previous month. This measure might however be biased since both the contributions of a given developer and those of his or her team members in the previous period might be affected by common external factors. Suppose, for instance, that a productivity shock affects the project, such as one participant making a breakthrough that facilitates contributions by all the others. Such a shock, unobservable to us, could, in theory, simultaneously have affected the contribution level of a developer and those of his or her team members in the previous period. We might thus incorrectly conclude, based on a positive correlation, that the individual was behaving reciprocally.

We thus proposed and built a field measure of reciprocity that corrected for this concern. For each developer, we computed the correlation between his or her contributions at the project $\times$ month level and the *predicted* contributions of his or her fellow team members in the previous month. This predicted measure used the variation in the sum of contributions made by their own collaborators on the other projects that they pursued independently. By "independently", we mean that we required that the developer under consideration did not contribute to those other projects and, thus, never directly interacted with the collaborators of his or her team members. We provide the formal description of the measure in Appendix C.

Figure 3 provides a graphical illustration of this strategy. For each developer i in our sample, we measure i's reactions in t to the monthly variation in contributions of his/her team members j in $t-1$, as predicted by the exogenous variation in contributions of their own team members k in $t-2$ on the projects that they pursue without developer i's involvement.

FIGURE 3: BUILDING A MEASURE OF RECIPROCITY BASED ON FIELD "ACTIVITY TRACES"



When computing the measure this way, we found that 3,700 out of the 8,250 developers for whom we can compute a field measure of reciprocal preferences in our final sample have a measure of reciprocity that is negative (i.e., they tend to decrease their level of contribution to a given project in the next period whenever their collaborators increase their own in the current period). This could possibly be an example of "altruistic" developers who care about providing as much public good as possible. Consequently, when their collaborators increase their contribution levels in a given team, they switch to contributing to other, relatively less well-developed open source projects. Such preferences cannot be captured by our experimental design where subjects are faced with a single common project. Alternatively, this pattern could be consistent with a scenario of substitutable inputs: if two programmers are perfect substitutes, and one developer makes a contribution, this contribution can no longer be made by the other developer. Either way, in order to increase the conceptual link between our field and lab measures where participants cannot contribute negatively to the public good, we defined our field measure of reciprocity as the maximum of zero and of the correlation calculated above. This variable is distributed with a mean of 0.24, a standard deviation of 0.62, and a min and max value of 0 and 11.05, respectively.

Table 4 explores the correlation between our lab and field measures of reciprocity. We can see from column (1) that both variables are strongly correlated: when the lab measure increases from 0 to 1, the field measure increases by an average of 0.5. Interestingly, the age, gender and education level of the developers (column (2)), as well as their self-reported motives for contributing (column

(3)), do not appear to be significantly related to their reciprocity preferences.

TABLE 4: CORRELATION BETWEEN LAB AND FIELD MEASURES

	(1) Reciprocity field	(2) Reciprocity field	(3) Reciprocity field
Reciprocity lab	0.50**	0.50**	0.49*
	(0.22)	(0.24)	(0.25)
Age		0.00	0.00
		(0.01)	(0.01)
Female		0.72	1.23
		(0.77)	(1.14)
Education level		-0.00	-0.02
		(0.06)	(0.07)
Motive: Ideology			0.25
			(0.22)
Motive: Like to learn			0.24
			(0.34)
Motive: Own use			0.07
			(0.20)
Motive: Establish reputation			0.33
			(0.31)
Motive: Pay			-0.03
			(0.19)
Regression type: cross-sectional, developer level			
R-squared	0.01	0.03	0.06
N. of obs	231.00	230.00	210.00

OLS estimates with robust standard errors in parentheses. Column (1) includes no control, column (2) includes developers' self reported demographics, and column (3) adds their reported motives for contributing to OSS.

5 Reciprocity and the success or failure of organizations

In the last part of the paper, we show that correcting for sampling bias generates new findings about the role of reciprocity in the success of organizations. We present these results in Section 5.1, and test the underlying theoretical mechanism in Section 5.2.

5.1 Reciprocity and failure

We set the stage in Table 5, where we link our experimental measure of reciprocity at the team level to average project-level success in our "lab-in-the-field" sample. In most cases, we only recruit one developer from each project through our experiment, so that a single lab measure serves as a proxy for the average reciprocity level of the team. For 11% of the projects (i.e., 131 out of 1,143), we capture more than one developer per team and therefore average their lab reciprocity scores.

In all of the following analyses, we standardize the activity percentile variable. As a result, the coefficients reported represent the effect of reciprocity in terms of standard deviations of the success score in the underlying population of projects. When we apply the lab-in-the-field method directly (i.e., we do not correct for sampling bias), we obtain the same result as that found in the previous literature. In column (1), we see that moving from no reciprocity to full reciprocity at the team level (i.e., the correlation between subjects' contributions in the experiment and the observed contributions of their team members shifts from 0 to 1) is associated with a significant 0.12 standard deviation increase in the success score.

In column (2), we add controls for project level characteristics. Our coefficient of interest remains relatively unchanged (it slightly increases to a value of 0.13). Among the control variables, using English as a working language is the one most significantly associated with success (i.e., 0.19 standard deviation increase in the success score). In column (3), we further add some controls for subjects' demographic characteristics and self-reported motives for contributing to OSS. This results in an increase in the statistical significance of our point-estimate, which reaches a value of 0.16. In terms of magnitude, this represents 89% of the effect on success associated with the team mastering English as a working language (a significant advantage to attract contributions in a globalized virtual work environment). Finally, we also see from column (3) that teams that include more women and are relatively more motivated by their reputation generally achieve significantly higher success scores.

TABLE 5: SUCCESS AND SURVIVAL: LAB-IN-THE-FIELD SAMPLE

	(1)	(2)	(3)
	Success score	Success score	Success score
Mean reciprocity lab	0.12**	0.13**	0.16***
	(0.06)	(0.06)	(0.06)
Project age		0.00*	0.00*
		(0.00)	(0.00)
License restrictiveness		-0.01	-0.00
		(0.03)	(0.04)
Software aimed at end user		0.12*	0.15**
		(0.06)	(0.07)
Team works in English		0.19***	0.18***
		(0.06)	(0.06)
Project uses popular programming language		-0.09*	-0.04
		(0.05)	(0.05)
Mean age			0.01*
			(0.00)
Mean female			0.38***
			(0.12)
Mean education level			0.01
			(0.02)
Mean motive: Ideology			-0.03
			(0.08)
Mean motive: Like to learn			-0.11
			(0.13)
Mean motive: Own use			0.03
			(0.09)
Mean motive: Establish reputation			0.22**
			(0.09)
Mean motive: Pay			-0.00
			(0.05)
Regression type: cross-sectional, project level			
R-squared	0.00	0.26	0.25
N. of obs	1143.00	1013.00	933.00

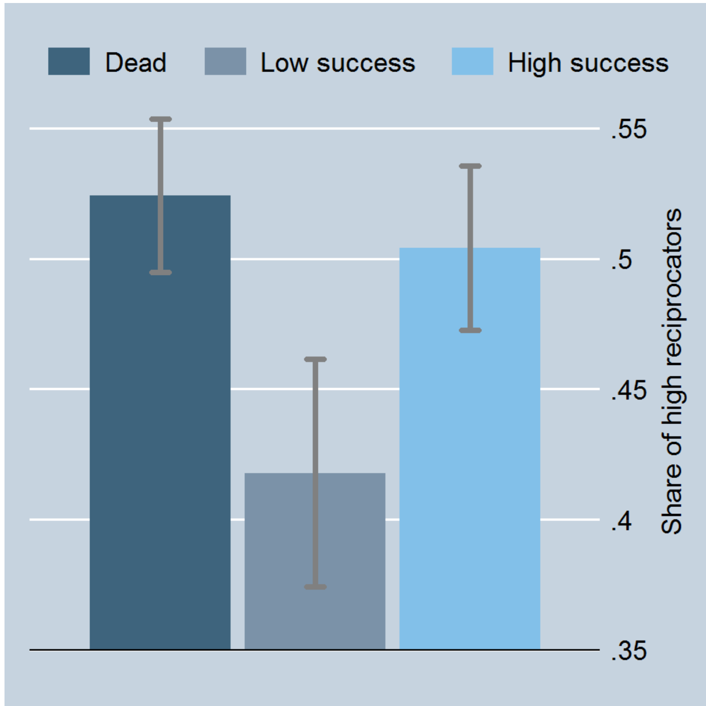
OLS estimates with robust standard errors in parentheses. Column (1) includes no control, column (2) includes project-level characteristics, and column (3) adds developers' self-reported demographics and motives for contributing to OSS (those scores are averaged if there is more than one subject per project).

The rest of the paper shows that we obtain far more nuanced results if we use our field measure of reciprocity on our expanded sample of organizations and their members. We begin by providing a graphical representation of our main result in Figure 4. We divide projects into three categories: (i) dead projects, defined as those that did not receive any contribution in the last 12 months of our time period; (ii) active but low-success projects, which have an average activity percentile that is lower than the median in the sample; and (iii) active and high-success projects, which have an average activity percentile greater than the median in the sample.[11] For these three different types of projects we plot the proportion of high reciprocators, defined as those that have an above-median field reciprocity measure.

Consistent with previous lab-in-the-field evidence, Figure 4 shows that low-success projects have an average of 40% of high reciprocators in their team, while high-success projects have 50% of high reciprocators – a 25% increase in proportion. Strikingly, however, dead projects do not significantly differ from highly successful ones in terms of the share of high reciprocators in their teams: 52% on average. Post-hoc power analyses suggest that our study is properly powered to detect the empirical differences that we seek to uncover. Setting the probability of a Type 1 error at α=5%, the estimated power of a two-sided t-test of the difference in the proportion of high reciprocators between high and low success teams is of 91,7%. Similarly, the estimated power of a test of the difference in proportion between dead and low success teams is of 98.5%.

[11] For the purpose of this figure, we only include projects with at least two developers, and for which we can compute a field measure of reciprocity for at least a third of the team members. Our subsequent regression analysis releases these constraints to establish the robustness of this result.

FIGURE 4: SUCCESS AND FAILURE OF PROJECTS WITH THE SHARE OF CONDITIONAL COOPERATORS



This graphical analysis is confirmed in a regression framework. In Table 6, we examine the relationship between the proportion of high reciprocators in a given OSS team and project-level performance using our field measure of reciprocity. All regressions control for project-level characteristics (i.e., age, license restrictiveness, target user population, English as a working language, popularity of the programming languages used), and rely on robust standard errors for inference. The dependent variable in column (1) is a binary variable indicating whether the project is dead or not (i.e., did not receive any contribution in the last 12 months of our time period). Moving from an organization with no high reciprocators to one composed only of highly reciprocal members is associated with a 12% increase in the probability of project failure and death. Consistent with our results from Table 5, mastering English as a working language is the control variable most strongly associated with project failure. Specifically, teams that do not list English as one of their working languages are, on average, 8% more likely to fail. As a point of comparison, this represents 67% of the effect associated with working in a fully reciprocal team.

We next examine how the proportion of high reciprocators influences team success. To do so, we use the activity percentile variable. In column (2), we see that groups with a high share of highly reciprocal members achieve better success scores. Moving from a team with no high reciprocators to one composed only of this type is associated with a 0.17 standard deviation increase in the activity percentile. This relationship becomes stronger in column (3), where we only retain projects that did not fail and remained active over the period.

In columns (4) to (6), we reproduce columns (1) to (3), but attempt to increase the precision of our estimates by excluding projects in which we cannot compute our field measure of reciprocity for at least one third of the team members. With this restriction, the link between team level reciprocity and the probability of failure becomes stronger (column (4)), while the link with project success becomes statistically insignificant (column (5)). However, we recover this significant positive relationship when we restrict our sample of projects to those that did not fail over the period (column (6)).

TABLE 6: SUCCESS AND SURVIVAL: EXPANDED SAMPLE

	(1)	(2)	(3)	(4)	(5)	(6)
	Project dead	Success score	Success score	Project dead	Success score	Success score
Share of high reciprocators	0.12***	0.17**	0.19**	0.15***	0.11	0.21**
	(0.04)	(0.08)	(0.09)	(0.05)	(0.08)	(0.10)
Project age	-0.00***	0.00***	0.00*	-0.00***	0.00***	0.00*
	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)
License restrictiveness	0.03*	-0.07*	-0.02	0.05**	-0.06	-0.01
	(0.02)	(0.03)	(0.04)	(0.02)	(0.04)	(0.04)
Software aimed at end user	0.05	0.06	0.09	0.05	0.07	0.09
	(0.04)	(0.06)	(0.06)	(0.04)	(0.07)	(0.07)
Team works in English	-0.08**	0.26***	0.20***	-0.08**	0.25***	0.18***
	(0.04)	(0.06)	(0.06)	(0.04)	(0.06)	(0.06)
Project uses popular programming language	-0.02	-0.02	-0.06	-0.02	-0.05	-0.06
	(0.04)	(0.05)	(0.05)	(0.04)	(0.05)	(0.05)
Regression type: cross-sectional, project level						
R-squared	0.02	0.06	0.07	0.03	0.05	0.07
N. of obs	1142.00	1142.00	551.00	1049.00	1049.00	510.00

OLS estimates with robust standard errors in parentheses. The dependent variable in columns (1) and (4) takes the value 1 if the project died during our time period. The dependent variable in columns (2)-(3) and (5)-(6) is the project-level standardized success score. Columns (1) and (2) use our full sample of projects, while column (3) only retains projects that survive during our entire time period. Columns (4) and (5) exclude all projects from the full sample for which we cannot compute the field measure of reciprocity for at least one third of the team members (to increase the precision of our estimates), whereas column (6) restricts this sample to projects that survive during our entire time period.

Our results at this point show that sampling bias may have prevented the extant lab-in-the-field literature from replicating lab-based results that imply that reciprocal preferences can also lead to increased failure rates at the team level. In Appendix E, we consider a number of robustness checks on these results. In particular, our results are robust to various ways of defining project death and field reciprocity. We can also exclude the interpretation according to which dead projects would actually be the most successful ones – having reached a mature stage where contributions are no longer needed – by showing that (i) projects are more likely to fail at an early development stage, and (ii) mature projects are actually the ones that are most actively developed (see Appendix E). One last concern could be particularly worrisome, so we address it in the main text. Namely, our field measure of reciprocity could capture the effect of individual-level characteristics other than reciprocal preferences, such as developers' cognitive abilities.

To demonstrate that omitted variable bias is unlikely to influence our field results, Table 7 reproduces the first three columns of Table 6. However, it considers the sample of projects that only have one contributor, i.e., where reciprocity preferences should not normally determine their success or failure. When we do this, we fail to find any strong statistical link between each developer's level of reciprocity and either the success of his or her project, or the probability that it eventually fails and dies. This suggests that our results are indeed motivated by the reciprocal dynamics that we seek to capture at the team level.

TABLE 7: PROJECTS WITH A SINGLE CONTRIBUTOR

	(1)	(2)	(3)
	Project dead	Success score	Success score
High reciprocator	0.01	0.10*	-0.06
	(0.02)	(0.05)	(0.08)
Project age	0.00***	-0.00***	-0.00**
	(0.00)	(0.00)	(0.00)
License restrictiveness	0.01	-0.06	-0.13**
	(0.01)	(0.04)	(0.06)
Software aimed at end user	0.01	0.28***	0.26**
	(0.03)	(0.08)	(0.13)
Developer works in English	-0.11***	0.35***	0.36***
	(0.03)	(0.08)	(0.14)
Project uses popular programming language	0.02	0.02	0.05
	(0.02)	(0.06)	(0.12)
Regression type: cross-sectional, project level			
R-squared	0.04	0.04	0.08
N. of obs	1433.00	1433.00	265.00

OLS estimates with robust standard errors in parentheses. This table replicates columns (1)-(3) of Table 6 using the sample of single-authored projects. The dependent variable in column (1) takes the value 1 if the project died during our time period. The dependent variable in columns (2)-(3) is the project level standardized success score. Columns (1) and (2) use our full sample of projects, while column (3) only retains projects that survive during our entire time period.

5.2 Mechanism: the reciprocity rollercoaster

In the lab, reciprocal preferences typically work as a catalyst, which equally amplifies positive and negative contribution dynamics at the group level. As a result, the presence of many highly reciprocal members in a team leads to the decay of cooperation whenever some observe that others do not match their contribution level in any given period (Fischbacher and Gächter, 2010). At the same time, strong reciprocity makes it possible to sustain high levels of cooperation when everybody contributes at a relatively high rate (Page et al., 2005).

Our paper is the first to provide a field test of this micro-level mechanism. We hypothesize that highly reciprocal individuals will react positively to increased contributions from team members, but will also decrease their contributions at a higher rate if they believe that others do not exert sufficient effort. An organization with a high proportion of strong reciprocators is therefore likely to perform at above-average levels when contributions are already relatively high. On the other hand, the same organization is predicted to have a harder time recovering from a period of inactivity, resulting in a significant increase in the probability of death. Note that for cooperation to collapse within this framework, it is sufficient for team members to *believe* that others are not contributing. This is especially important in field contexts where actual contributions (i.e., contributions that are observable to other team members) often imperfectly reflect individuals' effort levels (i.e., their willingness to cooperate). Over the course of many months or years, some periods of inactivity in our virtual team setting will doubtlessly result from idiosyncratic shocks at the individual or project level. Such shocks could then affect beliefs about individual effort levels, and ultimately determine the dynamics of cooperation within the team.

In Table 8, we take advantage of the panel structure of our data on individual team contributions to analyze the dynamics of cooperation within projects at a monthly frequency, over the eight-year period covered by our study. To do this, we run developer-level panel regressions where our unit of observation is at the developer x project x month level.

We first analyze whether the share of highly reciprocal developers at the team level impacts the probability that a project recovers from a period of inactivity (column (1)). In this regression, we define inactivity as a period of three consecutive months without any team contribution, and include an interaction term indicating whether the developer is of the high reciprocity type. Our dependent variable is a dummy indicating whether the developer made any contribution to the project in the following month. The regression controls for all available project-level characteristics and reports robust standard errors clustered at the project level. We see that facing a period of inactivity increases the probability that the developer will make no contribution in the subsequent month by 13% when he or she is not highly reciprocal. Consistent with our hypothesis, this probability increases by an additional 10% in the case of a highly reciprocal developer. Conversely, when contributions have been made to the project in previous periods, highly reciprocal developers have a 10% lower probability of making no contribution in the following period.

These results are confirmed in column (2), where we take the total number of code contributions (or "commits") made by a developer to a project in any given month as an alternative dependent variable. The specification is the same as in column (1), except that we now include developer level fixed effects in order to properly account for unobservable characteristics at the individual level. We obtain similar results. Facing a period of inactivity decreases the average number of contributions made in the current period by 1.25 for non-reciprocators. In the case of high reciprocity types, this number decreases to -3.54, a threefold increase in magnitude.[12]

Finally, column (3) relies on the same econometric specification as column (2), but focuses on the benefits of reciprocal preferences at the team level, i.e., the fact that they typically reinforce cooperative dynamics. For each project, we define a period of "high activity" as a period in which the number of contributions made in the three previous months was higher than the median number of monthly contributions over the history of the project. We find that non-highly reciprocal developers make an average of 2.3 more contributions following a period of high activity, while highly reciprocal ones make an average of 5.5 more contributions – a 150% increase in contribution levels. Taken together, these results provide evidence for the micro-level mechanism that we posit behind our aggregate results: highly reciprocal organizations are both more highly represented among top-performers (at least conditional on survival), but are also more likely to experience cooperation breakdowns and fail.

[12]In order to compute the effect size on highly reciprocal types, the baseline coefficient is added to the interaction term: $-1.25 - 2.28 = -3.54$.

TABLE 8: MICRO-LEVEL MECHANISM

	(1) No contribution	(2) Total no. of commits	(3) Total no. of commits
No contribution over last three periods	0.13***	-1.25**	
	(0.01)	(0.61)	
interaction with high reciprocity type	0.10***	-2.28***	
	(0.01)	(0.79)	
Above-median contributions over last three periods			2.31**
			(1.03)
interaction with high reciprocity type			3.16**
			(1.33)
High reciprocity type	-0.10***	x	x
	(0.01)		
Project age	0.00***	-0.00**	-0.00**
	(0.00)	(0.00)	(0.00)
License restrictiveness	0.00	0.08	0.08
	(0.00)	(0.10)	(0.10)
Software aimed at end user	-0.01	0.07	0.06
	(0.00)	(0.18)	(0.18)
Team works in English	-0.01	0.44*	0.50*
	(0.00)	(0.25)	(0.27)
Project uses popular programming language	0.00	-0.31*	-0.34*
	(0.00)	(0.17)	(0.19)
Regression type: panel, developer × project × month level			
Developer fixed effect	NO	YES	YES
R-squared	0.12	0.01	0.02
N. of obs	2.1e+05	2.1e+05	2.1e+05

OLS estimates with robust standard errors clustered at the developer level in parentheses. The dependent variable in column (1) is a dummy variable indicating whether no contribution is made by the developer in a given project × month. We regress it on a variable that captures whether no contributions were made by team members in the previous 3 months, interacted with the social type of the developer (highly reciprocal or not). The dependent variable in columns (2)-(3) is the number of code contributions (or "commits") made by the developer in a given project × month. We regress it on a variable that captures whether team members made above median contributions to the project in the previous 3 months, interacted with the social type of the developer (highly reciprocal or not). Note that columns (2) and (3) include developer-level fixed effects (this is why the coefficient on the variable "high reciprocity type" is dropped from those regressions).

6 Conclusion

This paper provides the first field evidence that strong reciprocity is a double-edged sword for organizations. We study the context of open source software development (OSS) – where team members need to cooperate voluntarily towards the provision of a public good – and link lab and field measures of reciprocity to project-level archival data to demonstrate that highly reciprocal teams are not necessarily more successful. They outperform other teams conditional on survival, but are also more likely to experience cooperation breakdowns. Moreover, we use the detailed panel structure of our data to pinpoint the micro-level mechanism behind these aggregate results. Reciprocal preferences work as a catalyst at the team level: they reinforce the cooperative equilibrium as long as team members receive positive signals, leading to top-notch performance, but they also increase the probability of a cooperation breakdown in the face of negative news.

At a methodological level, our paper adds to the fast-growing "lab-in-the-field" literature. This methodology is a powerful tool for organizational research. The experimenter relies on validated lab paradigms to elicit psychological traits or preferences that are difficult to identify "in the wild", and links them to outcomes of theoretical interest in the field. In this process, he or she can maintain a close connection with laboratory-based research, thus alleviating the tension between internal and external validity in experimental research (Gneezy and Imas, 2017). At the same time, the lab-in-the-field method is most useful to disentangle competing micro-level mechanisms that could potentially drive aggregate empirical regularities, therefore contributing to the micro-foundation movement in organizational behavior (Felin and Foss, 2005; Felin et al., 2015; Bitektine et al., 2018).

Several previous studies have used this methodology to test the link between reciprocity and group-level performance (Barr and Serneels, 2009; Rustagi et al., 2010; Carpenter and Seki, 2011). Surprisingly, all of them report an unambiguous positive relationship between reciprocal preferences and organizational success. This result represents a relative disconnect with the lab literature, where reciprocal preferences trigger the decay of cooperation whenever highly reciprocal types observe that others' efforts are not commensurate with their own. We hypothesize that this disconnect may result from the fact that the lab-in-the-field method requires researchers to run their experiments directly within the natural environment of interest. We argue that this can introduce a significant sampling bias: previously failed organizations are much less likely to be included in the design since former members may no longer be present on the research site. Thus, because they rely on a design that tends to ignore failed projects and their members, previous studies may not capture the dark side of reciprocal dynamics at the organizational level. To address this problem, we propose an additional way to use lab data in the field, i.e., to validate an analogous field measure based on archival data (or field "activity traces") to expand our research sample, notably in the direction of previously failed organizations.

Our field results suggest that mechanisms that screen workers according to their social type are unlikely to generate a self-sustained cooperative equilibrium. In an uncertain world in which organizations are subject to adverse shocks and imperfect monitoring, implicit norms define the nature of the cooperation game that agents play on a daily basis, and "relational contracts" guide the interpretation of the (noisy) signals received by individual members (Chassang, 2010; Gibbons and Henderson, 2012; Acemoglu and Jackson, 2015). Like all others, highly reciprocating teams are subject to "unforeseen contingencies" (Kreps, 1990). Depending on the members' previous shared history (or "relational knowledge"), such adverse shocks may or may not trigger the unraveling of cooperation within the organization. Future experimental research on cooperation should seek to explore how relational contracts can be defined in a way that minimizes the probability of inefficient cooperation breakdowns among reciprocally-minded workers (Fudenberg et al., 2012; Gibbons et al., 2020).

Acknowledgements

This research was funded by the consolidator grant agreement 647870 from the European Research Council, European Union's Horizon 2020 Research and Innovation Programme. We gratefully acknowledge logistical support from the Sciences Po médialab. We are indebted to Sourceforge.net for endorsing our study, and to the Sourceforge Research Data Archive (SRDA) at the University of Notre Dame for granting us privileged access to their data. We are grateful to Anne l'Hôte, Romain Guillebert and Matt van Antwerp for their outstanding research assistance. Final thanks go to the open source software developers who took part in this study. The authors have no conflicts of interest to disclose.

References

Acemoglu, D. and M. O. Jackson (2015). History, expectations, and leadership in the evolution of social norms. *The Review of Economic Studies 82*(2), 423–456.

Ambrus, A. and B. Greiner (2012). Imperfect public monitoring with costly punishment: An experimental study. *American Economic Review 102*(7), 3317–32.

Anthony, D. (2005). Cooperation in microcredit borrowing groups: Identity, sanctions, and reciprocity in the production of collective goods. *American Sociological Review 70*(3), 496–515.

Baldassarri, D. and G. Grossman (2011). Centralized sanctioning and legitimate authority promote cooperation in humans. *Proceedings of the National Academy of Sciences 108*(27), 11023–11027.

Barr, A. and P. Serneels (2009). Reciprocity in the workplace. *Experimental Economics 12*(1), 99–112.

Bartling, B., E. Fehr, and K. M. Schmidt (2012). Screening, competition, and job design: Economic origins of good jobs. *The American Economic Review 102*(2), 834–864.

Belenzon, S. and M. Schankerman (2015). Motivation and sorting of human capital in open innovation. *Strategic Management Journal 36*(6), 795–820.

Benkler, Y. (2002). Coase's penguin, or, linux and" the nature of the firm". *Yale Law Journal*, 369–446.

Bereby-Meyer, Y. and A. E. Roth (2006). The speed of learning in noisy games: Partial reinforcement and the sustainability of cooperation. *American Economic Review 96*(4), 1029–1042.

Bitektine, A., J. Lucas, and O. Schilke (2018). Institutions under a microscope: experimental methods in institutional theory. In *Unconventional methodology in organization and management research*, pp. 147–167. Oxford University Press.

Blair, R. (2018). Legitimacy after violence: Evidence from two lab-in-the-field experiments in liberia. *Available at SSRN 2326671*.

Blau, P. M. (1964). *Exchange and power in social life*. Transaction Publishers.

Carpenter, J. and E. Seki (2011). Do social preferences increase productivity? field experimental evidence from fishermen in toyama bay. *Economic Inquiry 49*(2), 612–630.

Charness, G. and M.-C. Villeval (2009). Cooperation and competition in intergenerational experiments in the field and the laboratory. *American Economic Review 99*(3), 956–78.

Charness, G. and C.-L. Yang (2014). Starting small toward voluntary formation of efficient large groups in public goods provision. *Journal of Economic Behavior & Organization 102*, 119–132.

Chassang, S. (2010). Building routines: Learning, cooperation, and the dynamics of incomplete relational contracts. *American Economic Review 100*(1), 448–65.

Chaudhuri, A. (2011). Sustaining cooperation in laboratory public goods experiments: a selective survey of the literature. *Experimental Economics 14*(1), 47–83.

Cinyabuguma, M., T. Page, and L. Putterman (2005). Cooperation under the threat of expulsion in a public goods experiment. *Journal of public Economics 89*(8), 1421–1435.

Cook, K. S., C. Cheshire, E. R. Rice, and S. Nakagawa (2013). Social exchange theory. In *Handbook of social psychology*, pp. 61–88. Springer.

Cook, K. S. and R. M. Emerson (1978). Power, equity and commitment in exchange networks. *American sociological review*, 721–739.

Cropanzano, R., E. L. Anthony, S. R. Daniels, and A. V. Hall (2017). Social exchange theory: A critical review with theoretical remedies. *Academy of Management Annals 11*(1), 479–516.

Cropanzano, R. and M. S. Mitchell (2005). Social exchange theory: An interdisciplinary review. *Journal of management 31*(6), 874–900.

Crowston, K., H. Annabi, J. Howison, and C. Masango (2004). Towards a portfolio of floss project success measures. In *Workshop on Open Source Software Engineering, 26th International Conference on Software Engineering, Edinburgh*. IET.

Crowston, K. and B. Scozzi (2002). Open source software projects as virtual organisations: competency rallying for software development. *IEE Proceedings-Software 149*(1), 3–17.

Crowston, K., K. Wei, J. Howison, and A. Wiggins (2012). Free/libre open-source software development: What we know and what we do not know. *ACM Computing Surveys (CSUR) 44*(2), 7.

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

David, P. A. and J. S. Shapiro (2008). Community-based production of open-source software: What do we know about the developers who participate? *Information Economics and Policy 20*(4), 364–398.

Denant-Boemont, L., D. Masclet, and C. N. Noussair (2007). Punishment, counterpunishment and sanction enforcement in a social dilemma experiment. *Economic theory 33*(1), 145–167.

Dufwenberg, M. and G. Kirchsteiger (2004). A theory of sequential reciprocity. *Games and economic behavior 47*(2), 268–298.

Dyer, J. H. and K. Nobeoka (2000). Creating and managing a high-performance knowledge-sharing network: the toyota case. *Strategic management journal 21*(3), 345–367.

Eisenberger, R., N. Cotterell, and J. Marvel (1987). Reciprocation ideology. *Journal of personality and social psychology 53*(4), 743.

Emerson, R. M. (1976). Social exchange theory. *Annual review of sociology 2*(1), 335–362.

Falk, A. and U. Fischbacher (2006). A theory of reciprocity. *Games and economic behavior 54*(2), 293–315.

Fallucchi, F., R. A. Luccasen III, and T. Turocy (2017). Behavioural types in public goods games: A re-analysis by hierarchical clustering. *SSRN working paper*.

Faraj, S., S. L. Jarvenpaa, and A. Majchrzak (2011). Knowledge collaboration in online communities. *Organization science 22*(5), 1224–1239.

Fehr, E. and S. Gachter (2000). Cooperation and punishment in public goods experiments. *American Economic Review 90*(4), 980–994.

Fehr, E. and S. Gächter (2000). Fairness and retaliation: The economics of reciprocity. *The journal of economic perspectives 14*(3), 159–181.

Fehr, E. and A. Leibbrandt (2011). A field study on cooperativeness and impatience in the tragedy of the commons. *Journal of Public Economics 95*(9-10), 1144–1155.

Felin, T. and N. J. Foss (2005). Strategic organization: A field in search of micro-foundations.

Felin, T., N. J. Foss, and R. E. Ployhart (2015). The microfoundations movement in strategy and organization theory. *The Academy of Management Annals 9*(1), 575–632.

Finan, F. and L. Schechter (2012). Vote-buying and reciprocity. *Econometrica 80*(2), 863–881.

Fischbacher, U. and S. Gächter (2010). Social preferences, beliefs, and the dynamics of free riding in public goods experiments. *The American economic review 100*(1), 541–556.

Fischbacher, U., S. Gächter, and E. Fehr (2001). Are people conditionally cooperative? evidence from a public goods experiment. *Economics letters 71*(3), 397–404.

Fudenberg, D., D. G. Rand, and A. Dreber (2012). Slow to anger and fast to forgive: Cooperation in an uncertain world. *American Economic Review 102*(2), 720–49.

Gächter, S., E. Renner, and M. Sefton (2008). The long-run benefits of punishment. *Science 322*(5907), 1510–1510.

Ghosh, R. A. (2007). Economic impact of open source software on innovation and the competitiveness of the information and communication technologies (ict) sector in the eu.

Gibbons, R. and R. Henderson (2012). Relational contracts and organizational capabilities. *Organization science 23*(5), 1350–1364.

Gibbons, R. S., M. Grieder, H. Herz, and C. Zehnder (2020). Building an equilibrium: Rules versus principles in relational contracts. *CESifo Working Paper No. 7871*.

Gilligan, M. J., B. J. Pasquale, and C. Samii (2014). Civil war and social cohesion: Lab-in-the-field evidence from nepal. *American Journal of Political Science 58*(3), 604–619.

Gittell, J. H. (2000). Organizing work to support relational co-ordination. *International Journal of Human Resource Management 11*(3), 517–539.

Gittell, J. H., R. Seidner, and J. Wimbush (2010). A relational model of how high-performance work systems work. *Organization science 21*(2), 490–506.

Gittell, J. H., A. Von Nordenflycht, and T. A. Kochan (2004). Mutual gains or zero sum? labor relations and firm performance in the airline industry. *ILR Review 57*(2), 163–180.

Gneezy, U. and A. Imas (2017). Lab in the field: Measuring preferences in the wild. In *Handbook of economic field experiments*, Volume 1, pp. 439–464. Elsevier.

Gouldner, A. W. (1960). The norm of reciprocity: A preliminary statement. *American sociological review*, 161–178.

Grechenig, K., A. Nicklisch, and C. Thöni (2010). Punishment despite reasonable doubt—a public goods experiment with sanctions under uncertainty. *Journal of Empirical Legal Studies 7*(4), 847–867.

Greenstein, S. and F. Nagle (2014). Digital dark matter and the economic contribution of apache. *Research Policy 43*(4), 623–631.

Grewal, R., G. L. Lilien, and G. Mallapragada (2006). Location, location, location: How network embeddedness affects project success in open source systems. *Management science 52*(7), 1043–1056.

Grossman, G. and D. Baldassarri (2012). The impact of elections on cooperation: Evidence from a lab-in-the-field experiment in uganda. *American journal of political science 56*(4), 964–985.

Helper, S. and R. Henderson (2014). Management practices, relational contracts, and the decline of general motors. *Journal of Economic Perspectives 28*(1), 49–72.

Hergueux, J. and N. Jacquemet (2015). Social preferences in the online laboratory: a randomized experiment. *Experimental Economics 18*(2), 251–283.

Herrmann, B., C. Thöni, and S. Gächter (2008). Antisocial punishment across societies. *Science 319*(5868), 1362–1367.

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

Hippel, E. v. and G. v. Krogh (2003). Open source software and the "private-collective" innovation model: Issues for organization science. *Organization science 14*(2), 209–223.

Homans, G. C. (1961). Human behavior: Its elementary forms.

Ichniowski, C., K. Shaw, and G. Prennushi (1997). The effects of human resource management practices on productivity: A study of steel finishing lines. *The American Economic Review 87*(3), 291–313.

Karlan, D. S. (2005). Using experimental economics to measure social capital and predict financial decisions. *American Economic Review 95*(5), 1688–1699.

Kellogg, K. C. (2009). Operating room: Relational spaces and microinstitutional change in surgery. *American journal of sociology 115*(3), 657–711.

Kosfeld, M. and D. Rustagi (2015). Leader punishment and cooperation in groups: Experimental field evidence from commons management in ethiopia. *American Economic Review 105*(2), 747–83.

Kosfeld, M. and F. A. Von Siemens (2011). Competition, cooperation, and corporate culture. *The RAND Journal of Economics 42*(1), 23–43.

Kreps, D. M. (1990). Corporate culture and economic theory. *Perspectives on positive political economy 90*(109-110), 8.

Kunreuther, H., G. Silvasi, E. Bradlow, D. Small, et al. (2009). Bayesian analysis of deterministic and stochastic prisoner's dilemma games. *Judgment and Decision Making 4*(5), 363.

Lakhani, K. R., R. G. Wolf, et al. (2005). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. *Perspectives on free and open source software 1*, 3–22.

Lazer, D., A. Pentland, L. Adamic, S. Aral, A.-L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, et al. (2009). Social science. computational social science. *Science (New York, NY) 323*(5915), 721–723.

Ledyard, J. O. (1994). Public goods: A survey of experimental research.

Lee, G. K. and R. E. Cole (2003). From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development. *Organization science 14*(6), 633–649.

Leibbrandt, A. (2012). Are social preferences related to market performance? *Experimental Economics 15*(4), 589–603.

Levine, S. S. and M. J. Prietula (2013). Open collaboration for innovation: Principles and performance. *Organization Science 25*(5), 1414–1433.

Marlow, J., L. Dabbish, and J. Herbsleb (2013). Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 117–128. ACM.

Maurer, S. M. and S. Scotchmer (2006). Open source software: the new intellectual property paradigm. Technical report, National Bureau of Economic Research.

Mayer, K. J. and N. S. Argyres (2004). Learning to contract: Evidence from the personal computer industry. *Organization science 15*(4), 394–410.

Molm, L. D. (2003). Theoretical comparisons of forms of exchange. *Sociological Theory 21*(1), 1–17.

Molm, L. D., N. Takahashi, and G. Peterson (2000). Risk and trust in social exchange: An experimental test of a classical proposition. *American Journal of Sociology 105*(5), 1396–1427.

Montgomery, J. D. (1996). The structure of social exchange networks: a game-theoretic reformulation of blau's model. *Sociological Methodology*, 193–225.

Nikiforakis, N. (2008). Punishment and counter-punishment in public good games: Can we really govern ourselves? *Journal of Public Economics 92*(1-2), 91–112.

Nikiforakis, N., C. N. Noussair, and T. Wilkening (2012). Normative conflict and feuds: The limits of self-enforcement. *Journal of Public Economics 96*(9-10), 797–807.

O'Mahony, S. (2003). Guarding the commons: how community managed software projects protect their work. *Research policy 32*(7), 1179–1198.

Page, T., L. Putterman, and B. Unel (2005). Voluntary association in public goods experiments: Reciprocity, mimicry and efficiency. *The Economic Journal 115*(506), 1032–1053.

Roth, A. E. (1995). Introduction to experimental economics. *The handbook of experimental economics 1*, 3–109.

Rustagi, D., S. Engel, and M. Kosfeld (2010). Conditional cooperation and costly monitoring explain success in forest commons management. *Science 330*(6006), 961–965.

Scacchi, W. (2007). Free/open source software development: recent research results and emerging opportunities. In *The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers*, pp. 459–468. ACM.

Schram, A. (2005). Artificiality: The tension between internal and external validity in economic experiments. *Journal of Economic Methodology 12*(2), 225–237.

Sobel, J. (2005). Interdependent preferences and reciprocity. *Journal of economic literature 43*(2), 392–436.

Van Antwerp, M. and G. Madey (2008, September). Advances in the sourceforge research data archive (srda). In *Fourth International Conference on Open Source Systems, IFIP 2.13 (WoPDaSD 2008)*, Milan, Italy.

Van Antwerp, M. and G. Madey (2010). The importance of social network structure in the open source software developer community. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pp. 1–10. IEEE.

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

Walli, S., D. Gynn, and B. Von Rotz (2005). The growth of open source software in organizations. *Technical report*.

Weber, S. (2004). *The success of open source*. Harvard University Press.

Xiao, E. and H. Kunreuther (2016). Punishment and cooperation in stochastic social dilemmas. *Journal of Conflict Resolution 60*(4), 670–693.

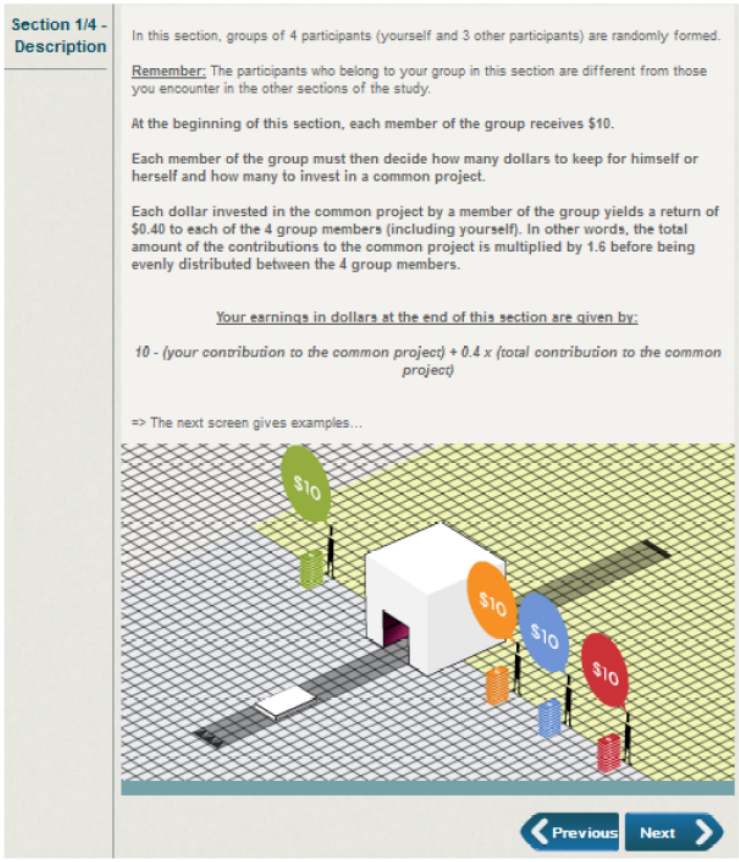
Appendix

A Detailed experimental procedures

Our online experiment relied on a fully self-contained interface designed specifically to increase the reliability of the experimental data collected over the Internet (Hergueux and Jacquemet, 2015). The welcome page of the decision interface provided subjects with general information about the experiment, including the number of sections, expected completion time, and how earnings are computed. In order to minimize potential demand effects and in-group biases, we were very careful not to present the study as being OSS-oriented. We made it very clear on the introductory screen that subjects would interact with a diverse pool of Internet users. Final earnings were computed by randomly matching our subjects with individuals from a pool made up of OSS developers and students.

One important methodological aspect of the online implementation of the experiment is to guarantee a quick and thorough understanding of the instructions when no interaction with the experimenter is possible. We strengthened the internal validity of our online experiment through three distinctive features of the interface. First, we included novel flash animations illustrating the written experimental instructions at the bottom of the instruction screen (see Figure A1). Second, the instruction screen was followed by a screen providing some examples of decisions, along with a detailed calculation of the resulting payoffs for each player. These examples were supplemented on the subsequent screen by an earnings calculator. On this interactive page, subjects were allowed to test any scenario they wanted to consider. Finally, the system provided quick access to the instruction material at any moment during decision-making.

FIGURE A1: THE INSTRUCTION SCREEN OF THE PUBLIC GOODS GAME



At the end of the experiment, subjects' final payoffs were added to their $10 participation fee. Payments were made via an automated PayPal transfer.[13] It is important to stress that OSS developers can be very hostile to monetary rewards. In order to ensure that the experiment was equally incentive-compatible for all subjects, we allowed them to donate their final earnings to the International Committee of the Red Cross upon completion of the experiment. This possibility was made clear on the welcome screen of the decision interface.

[13]Such a payment procedure guarantees a fungibility similar to that of cash transfers in lab experiments since money transferred via PayPal can be readily used for online purchases or easily transferred to one's personal bank account at no cost. We only required a valid e-mail address to process the payment. To strengthen the credibility of the payment procedure, we asked subjects to enter the e-mail address that was (or would be) associated with their PayPal account right after the introductory screen of the decision interface.

B Pairwise correlation of the control variables

Table 4 in the main text discusses the correlation between our explanatory variable of theoretical interest (our reciprocity measure) and subjects' background characteristics, i.e., their demographics and self-reported motivations for contributing to open source software projects. The main conclusion from this Table is twofold. First, our lab and field measures of reciprocity are strongly correlated across developers, indicating that our field measure of reciprocity is conceptually valid. Second, our developer-level control variables are not significantly correlated with subjects' reciprocal preferences, alleviating the concern that omitted variable bias might drive our results (see also the discussion about Table 7 in the main text).

For the sake of completeness, this appendix reports the full pairwise correlation structure among our subject-level and project-level control variables in Tables A1 and A2, respectively.

TABLE A1: PAIRWISE CORRELATION OF THE SUBJECT-LEVEL CONTROL VARIABLES

Variables	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
(1) Age	1.000								
(2) Female	0.022 0.467	1.000							
(3) Education level	0.318*** 0.000	0.107*** 0.000	1.000						
(4) Income level	0.425*** 0.000	-0.018 0.577	0.272*** 0.000	1.000					
(5) Motive: Like to learn	-0.175*** 0.000	0.033 0.306	-0.139** 0.000	-0.175*** 0.000	1.000				
(6) Motive: Own use	0.030 0.342	-0.009 0.781	0.037 0.245	0.012 0.727	0.065** 0.041	1.000			
(7) Motive: Ideology	-0.036 0.256	0.032 0.312	-0.029 0.361	-0.105*** 0.001	0.129*** 0.000	0.255*** 0.000	1.000		
(8) Motive: Reputation	-0.110*** 0.001	-0.052 0.101	-0.097*** 0.002	-0.063* 0.056	0.255*** 0.000	0.183*** 0.000	0.222*** 0.000	1.000	
(9) Motive: Pay	0.077** 0.016	0.003 0.927	0.023 0.481	-0.043 0.194	0.074** 0.019	0.111*** 0.001	0.073** 0.021	0.288*** 0.000	1.000

***, ** and * denote statistical significance at the 0.01, 0.05 and 0.1 level, respectively.

TABLE A2: PAIRWISE CORRELATION OF THE PROJECT-LEVEL CONTROL VARIABLES

Variables	(1)	(2)	(3)	(4)	(5)
(1) Project age	1.000				
(2) License restrictiveness	-0.044**	1.000			
	0.010				
(3) Software aimed at end user	0.346***	0.190***	1.000		
	0.000	0.000			
(4) Team works in English	0.410***	0.053***	0.398***	1.000	
	0.000	0.002	0.000		
(5) Project uses popular progr. language	0.133***	0.027	0.003	-0.055***	1.000
	0.000	0.131	0.846	0.002	

***, ** and * denote statistical significance at the 0.01, 0.05 and 0.1 level, respectively.

C Constructing the field measure of reciprocity

This appendix formally describes the construction of our measure of reciprocity in the field. Similar to our experimental design, the idea is to measure the contributions of an individual made in reaction to variations in the aggregate contributions of his or her team members in the previous period. The main issue is that those contributions could all be driven by common factors at the project level. This could be a technological shock or breakthrough, which would lead us to wrongly conclude that members of a project behave reciprocally. In the spirit of an instrumental variables approach, we therefore measure the variation in the contribution level of each developer i within our sample of 10,537 developers at the project $\times$ month level as a function of the lagged contributions of his or her team members, which we predict using the (exogenous) variation in the contributions of their own team members on the projects that they pursue independently (i.e., where developer i does not participate).

Specifically, consider developer i working on a set of projects $\mathcal{P}_i$ at time t, for whom we want to measure reciprocity based on field data. For a given project $p \in \mathcal{P}_i$, we denote y_{ipt} the contributions of that individual. An OLS specification of the relationship between the contributions of individual i at time t and the contributions of the other members of group p at time $t-1$ can be expressed as:

$$y_{ipt} = \beta_0 + \beta_1 \sum_{j \neq i} y_{jpt-1} + \beta_i + \beta_p + \gamma_{ipt}$$

It would be natural to define β_1 as a measure of reciprocal behavior in the field.

However, in practice, because some projects might experience common productivity shocks, it is likely that:

$$corr\left(\sum_{j \neq i} y_{jpt-1}, \gamma_{ipt}\right) \neq 0$$

We therefore construct a source of exogenous variation for y_{jpt-1}. To do this, we predict y_{jpt-1} based on the variation in the contributions of developer j's team members to the other projects to which he or she contributes at time $t-2$, excluding those where i participates (i.e., $p' \in \mathcal{P}_j/p, p' \notin \mathcal{P}_i$):

$$y_{jpt-1} = \alpha_0 + \alpha_1 \sum_{p' \in \mathcal{P}_j/p, p' \notin \mathcal{P}_i} \sum_{k \in p'} y_{kp't-2} + \alpha_j + \alpha_p + \varepsilon_{jpt-1} \qquad (1)$$

Columns (1) and (2) of Table A3 present the results of both the OLS and IV regressions. The contributions of j's team members to his or her other projects at time $t-2$ is a strong instrument for $y_{jp,(t-1)}$ (the F-statistic reaches a value of 92). On average, if team members contribute more to projects $p' \in \mathcal{P}_j/p, p' \notin \mathcal{P}_i$, developer j will, in turn, contribute significantly more to project p.

TABLE A3: ESTIMATING FIELD RECIPROCITY

	(1)	(2)
	ln(commits per month)$_p$	ln(commits per month)$_p$
Lagged contributions of team members in projects p'	0.09***	0.08***
	(0.01)	(0.00)
R-squared	0.04	0.04
N. of obs	9.9e+05	9.9e+05

Robust standard errors in parentheses
* p<0.1, ** p<0.05, *** p<0.01

For each developer, our measure of field reciprocity is therefore computed as the correlation between his or her contributions y_{ipt} and the predicted value $\hat{y}_{jpt-1}$ of the contributions of his or her team members in the previous month, using $y_{kp't-2}$ as the explanatory variable, as specified in Equation 1. If this correlation is negative, our reciprocity measure is constrained to take the value 0.

D The Activity Percentile metric

This appendix provides the formal definition of the Activity Percentile metric, which is used in the main text as our measure of project-level success (see Section 3.3). The measure is defined as follows:

$$Activity\ Percentile = \frac{1}{3}\ User\ Traffic + \frac{1}{3}\ Development\ Activity + \frac{1}{3}\ Project\ Communication,$$

with:

- $UserTraffic = \frac{ln(1+total\ downloads)}{ln(1+max[all\ projects])} + \frac{ln(1+total\ logo\ hits)}{ln(1+max[all\ projects])} + \frac{ln(1+total\ website\ hits)}{ln(1+max[all\ projects])}$,
- $DevelopmentActivity = \frac{ln(1+total\ commits)}{ln(1+max[all\ projects])} + \frac{100-nb\ days\ since\ last\ file\ release}{100} + \frac{100-nb\ days\ since\ last\ project\ admin\ login}{100}$,
- $ProjectCommunication = \frac{ln(1+total\ bug\ tracker\ submissions)}{ln(1+max[all\ projects])} + \frac{ln(1+total\ mailing\ list\ posts)}{ln(1+max[all\ projects])} + \frac{ln(1+total\ project\ forum\ posts)}{ln(1+max[all\ projects])}$.

While it could have been interesting to break down this indicator for the purpose of our analysis, Sourceforge neither separately stores nor reports the disaggregated components of its Activity Percentile metric.

E Robustness of regression results

E.1 Measure of death of a project

A potential concern with our measure of death at the project level could be that it actually captures highly successful projects that have reached a mature stage where contributions are no longer needed. Two main arguments challenge this view. First, projects tend to die relatively more frequently at an early development stage, as can be seen from Figure A2. Second, as can be seen from

Figure A3, the average number of commits that goes into each project tends to increase with its development stage, so that projects are actually most actively developed when they reach more advanced development stages.

FIGURE A2: DISTRIBUTION OF STAGE COMPARING DEAD AND ACTIVE PROJECTS

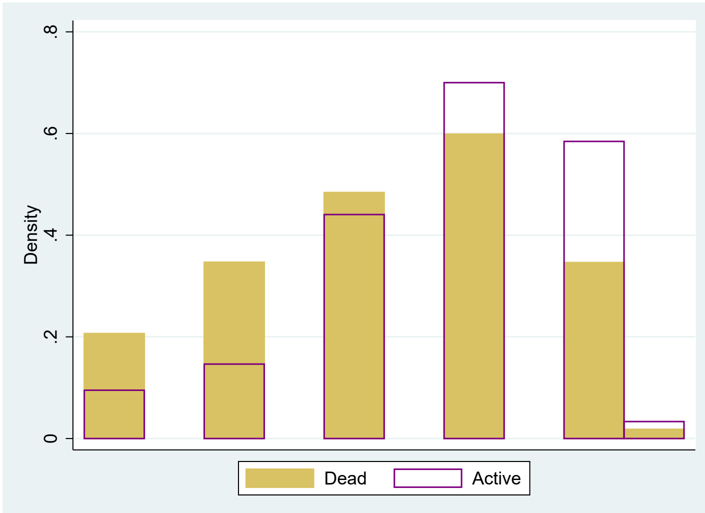
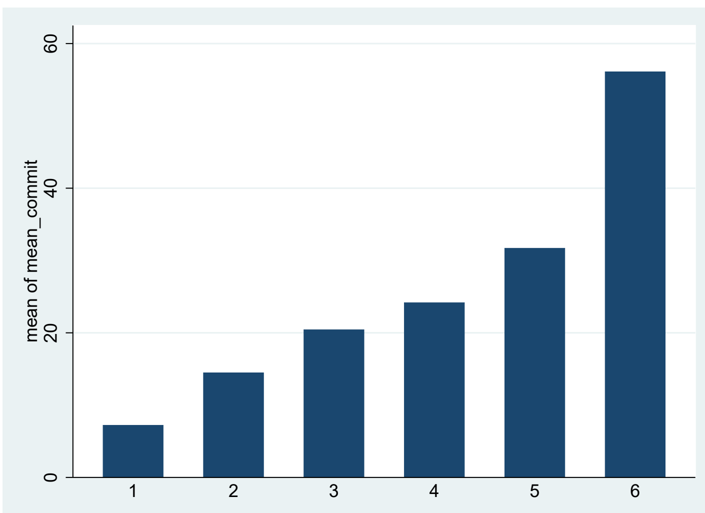


FIGURE A3: MEAN NUMBER OF COMMITS PER DEVELOPMENT STAGE



Finally, our main results are robust to restricting our sample to projects in early development stages. In Table A4, column (1) excludes projects that finish in production and mature stages, while column (2) further excludes projects that finish in beta version.

TABLE A4: RESTRICTING THE SAMPLE TO PROJECTS IN EARLY PHASE

	(1)	(2)
	Project dead	Project dead
Share of high reciprocators	0.10*	0.16***
	(0.06)	(0.05)
R-squared	0.11	0.11
N. of obs	530.00	894.00

Robust standard errors in parentheses.
All regressions include project-level controls (see Table 6 in main text).
* p<0.1, ** p<0.05, *** p<0.01

Moreover, we adopted a particular definition of dead projects: projects that received no contributions in the last 12 months of our sample. There are no official administrative data that record the death of a project, so we chose 12 months since it appeared to be reasonable and allowed us to split our sample equally between dead projects (52%) and active projects (48%). We nevertheless examine the robustness of our results to changes in the definition, varying the number of months without contributions at the end of the project. In column (1), we consider a period of 6 months, and in column (2), a period of 18 months. In all of the cases, an increase in the proportion of reciprocators in the group significantly increases the probability of project death.

TABLE A5: DEFINITION OF DEAD PROJECT

	(1)	(2)
	Project dead (6)	Project dead (18)
Share of high reciprocators	0.16***	0.11**
	(0.05)	(0.05)
R-squared	0.10	0.10
N. of obs	927.00	927.00

Robust standard errors in parentheses.
All regressions include project-level controls (see Table 6 in main text).
* p<0.1, ** p<0.05, *** p<0.01

One concern with our definition of dead projects is that projects could have migrated to a different platform. This is especially a concern with the competing platform Github, which increased in prominence during our sample period. (The competitor launched by Google, called Google Code, never managed to challenge Sourceforge.) To address this concern, we collected information on the Github projects and, in the main specifications of Table 6, removed all projects that migrated to Github. In Table A6, we do not exclude these projects and show that the results are very similar, indicating that this was not, in fact, a major concern for our estimation.

TABLE A6: RESTRICTING THE SAMPLE TO PROJECTS THAT DID NOT MIGRATE TO GITHUB

	(1)	(2)	(3)
	Success score	Project dead	Success score
Share of high reciprocators	0.05	0.17***	0.16*
	(0.08)	(0.05)	(0.10)
R-squared	0.22	0.11	0.13
N. of obs	927.00	927.00	435.00

Robust standard errors in parentheses.
All regressions include project-level controls (see Table 6 in main text).
* p<0.1, ** p<0.05, *** p<0.01

E.2 Definition of high reciprocity types

In Table 6 of the paper, we classify developers into two categories – high reciprocity and low reciprocity – according to the median value of our field measure of reciprocity. This approach is consistent with the lab-in-the-field literature on reciprocity, which typically distinguishes between "weak" and "strong" reciprocators (Fallucchi et al., 2017). Our results are relatively robust to variations in the way we define high reciprocity types. First, in Table A7, we define a high reciprocity type as a developer that has a field measure of reciprocity in the top quartile (as opposed to above median). Second, instead of assigning discrete types to developers, Table A8 reports the impact of our field measure of reciprocity when averaged over all developers in the project.

TABLE A7: HIGH RECIPROCITY – THE 75 PERCENTILE

	(1)	(2)	(3)
	Success score	Project dead	Success score
Share of high reciprocators (top quartile)	-0.01	0.16***	0.08
	(0.10)	(0.06)	(0.11)
R-squared	0.22	0.11	0.13
N. of obs	927.00	927.00	435.00

Robust standard errors in parentheses.
All regressions include project-level controls (see Table 6 in main text).
* p<0.1, ** p<0.05, *** p<0.01

TABLE A8: SUCCESS AND SURVIVAL WITH AVERAGE TEAM RECIPROCITY

	(1)	(2)	(3)
	Success score	Project dead	Success score
Average reciprocity in project	-0.13***	0.07**	-0.04
	(0.05)	(0.03)	(0.04)
R-squared	0.22	0.11	0.13
N. of obs	927.00	927.00	435.00

Robust standard errors in parentheses.
All regressions include project-level controls (see Table 6 in main text).
* p<0.1, ** p<0.05, *** p<0.01

E.3 Proportion of team members with a field measure of reciprocity

In the main specification of Table 6, we imposed the constraint that at least one third of the group members should have a measure of field reciprocity to guarantee that the average share of high reciprocators was not measured with excessive noise. In Table A9, we remove this constraint, whereas in Table A10 we impose a more stringent one, i.e., that we can compute a field measure of reciprocity for at least half of the team members. The result on project death is preserved. The result on the effect of the share of high reciprocators on the success of the project when restricting projects to those still active (column (3) in the tables) is no longer significant, although the magnitude and the size remain the same. For the case of no constraint, the standard deviation increases because the measure is less precise. For the case of the stronger constraint, the problem is that the number of observations is lower.

TABLE A9: NO CONSTRAINT ON PROPORTION

	(1)	(2)	(3)
	Success score	Project dead	Success score
Share of high reciprocators	0.11	0.14***	0.15
	(0.08)	(0.05)	(0.09)
R-squared	0.22	0.11	0.13
N. of obs	1011.00	1011.00	473.00

Robust standard errors in parentheses.
All regressions include project-level controls (see Table 6 in main text).
* p<0.1, ** p<0.05, *** p<0.01

TABLE A10: AT LEAST HALF OF PROJECT MEMBERS WITH MEASURE OF RECIPROCITY

	(1)	(2)	(3)
	Success score	Project dead	Success score
Share of high reciprocators	-0.01	0.23***	0.17
	(0.09)	(0.05)	(0.11)
R-squared	0.20	0.12	0.13
N. of obs	806.00	806.00	392.00

Robust standard errors in parentheses.

All regressions include project-level controls (see Table 6 in main text).

* p<0.1, ** p<0.05, *** p<0.01

E.4 Robustness of Table 8

Table 8 addresses the question of whether a high share of reciprocators serves as an amplification of variations, making it both more likely to fail in bad times and more likely to succeed in good times. To define good and bad times, we used the past history of the project, and in the main specification of Table 8, we used the three-month lag. We explore robustness by considering a two-month lag in Table A11 and a four-month lag in Table A12.

TABLE A11: TWO-PERIOD LAG

	(1)	(2)	(3)
	No contribution	Total no. of commits	Total no. of commits
No contribution over last three periods	0.14***	-1.52**	
	(0.01)	(0.72)	
interaction with high reciprocity type	0.10***	-2.59***	
	(0.01)	(0.90)	
High reciprocity type	-0.11***	x	x
	(0.01)		
Above median contributions over last three periods			2.27**
			(0.96)
interaction with high reciprocity type			3.07**
			(1.19)
R-squared	0.12	0.02	0.02
N. of obs	2.1e+05	2.1e+05	2.1e+05

Robust standard errors clustered at the developer level in parentheses.
All regressions include project-level controls (see Table 8 in main text).
Columns (2) and (3) include developer fixed effects.
* p<0.1, ** p<0.05, *** p<0.01

TABLE A12: FOUR-PERIOD LAG

	(1) No contribution	(2) Total no. of commits	(3) Total no. of commits
No contribution over last three periods	0.12***	-1.13**	
	(0.01)	(0.56)	
interaction with high reciprocity type	0.09***	-2.16***	
	(0.01)	(0.73)	
High reciprocity type	-0.10***	x	x
	(0.01)		
Above median contributions over last three periods			2.32**
			(1.10)
interaction with high reciprocity type			3.53**
			(1.43)
R-squared	0.11	0.01	0.02
N. of obs	2.1e+05	2.1e+05	2.1e+05

Robust standard errors clustered at the developer level in parentheses.
All regressions include project-level controls (see Table 8 in main text).
Columns (2) and (3) include developer fixed effects.
* p<0.1, ** p<0.05, *** p<0.01