Cathedrals Opening Bazaars: Firms' Open-Source Projects and Community Participation*

Xiaofeng Gong

This version: November 6, 2026

Abstract

This paper studies how participation in firms' open-source projects relates to the technologies the firms patent. I link Compustat firms to their USPTO patents and GitHub accounts and use a large language model to place each project in the Cooperative Patent Classification, the scheme patent examiners use. I document that most of these projects lie in technologies in which the firm was recently granted patents. I then compare projects first active in 2015–2022, conditional on firm-by-year and technology-by-year effects and on controls for each project's early activity, text and language. I find that projects in the firm's patented technologies have more new participants and more contributions from them. The difference in new participants holds for people with no earlier activity on the firm's GitHub accounts and without the five largest firms. Contributions from everyone are also higher, and among projects with a contribution there is no clear difference in new participants' share. According to my estimates, projects in the firm's patented technologies have 29.5 percent more new participants in the 24 months after their first active quarter, 45.5 percent more contributions from them, and 37.6 percent more contributions from everyone.

1 Introduction

A firm can patent inventions and publish open-source code in similar technologies, and the open-source code sometimes attracts people to contribute. When firms publish code in the technologies they patent, who participates, and how much do new participants contribute to the firms' projects? Firm-level totals cannot show whether participation is higher in the projects that overlap a firm's patented technologies, so I make the comparison repository by repository, with code and patents in one classification. Repositories in the technologies where their firm was recently granted patents have 29.5 percent more new participants and 45.5 percent more contributions from them in the 24 months after their first active quarter. These are associations, conditional on effects for each firm and year and for each technology and year and on controls, and the estimates leave open why.

*The image of the cathedral and the bazaar is from Raymond (2001), where the two are styles of building software. Casting a firm that patents as the cathedral is this paper's adaptation.

Raymond (2001) contrasts two ways of building software. In the cathedral, a small group works in isolation and releases rarely. In the bazaar, code is released early and often, and many people take part. A large firm that files patents plays the cathedral, and the open-source code it publishes, which anyone can read, use and build on, enters the bazaar. Projects run by companies account for a large share of public open-source code (Spinellis et al., 2020).

Firms choose what to reveal and what to keep secret. Patenting can reveal information that raises a competitor's profits (Horstmann et al., 1985), and firms also reveal some of the code they develop and keep the rest proprietary (Henkel, 2006). Firms in every industry protect knowledge by means other than patents (Cohen et al., 2000), and what a firm shares and what it protects is a question in the economics of appropriability (Teece, 1986).

As in the bazaar, people new to the firms' projects take part in them. The software-engineering literature studies how newcomers join projects (Pinto et al., 2018; Casalnuovo et al., 2015). I describe participants by their earlier activity on GitHub, and no actor is matched to an employer. The repositories studied are the 102,987 repositories of the firms defined below, first active in 2015–2022 (section 3.4). Of these repositories, 53.5 percent have at least one *new participant*, a person who first contributes to the repository in its first two years, and 41.1 percent have one with no earlier activity at the firm on GitHub (section 3.3). GitHub's public record shows the work people leave on a repository, while its users stay unseen, and the new participants are the users whose first work on it falls in those two years (section 3.3). The perspectives the paper draws on point both ways. One line of work expects less participation in projects tied to patented inventions, and others expect more under conditions they state (section 2).

Two gaps in the data kept a firm's patents and its open-source projects apart. Open-source code carries no patent class, so setting a firm's projects beside the technologies of its patents needs a classification of the code. Compustat and the patent records list no GitHub accounts, so each study builds its own link (Cao et al., 2026). Studies of corporate open-source code therefore relate how much code a firm opens to its patents, across firms (Fosfuri et al., 2008; Cao et al., 2026), or follow contributions around a dated event (Chen et al., 2022; Haese and Peukert, 2026). This paper adds a comparison of each repository with its own firm's patents. The data built here close both gaps. Crosswalks link firms to their patents and to their GitHub accounts, and a large language model (LLM) labels the repositories with the main groups, the detailed patent classes, that examiners assign to patents. Each labeled repository can then be set against its firm's own patents in one classification, repository by repository.

I link Compustat records to their USPTO patents and their GitHub repositories, and I label each repository with main groups of the Cooperative Patent Classification (CPC), the detailed classes that patent examiners assign (section 3). A person participates in a repository with a contribution, which is an issue, a pull request, a review or a comment, in GH Archive's record of GitHub's public events (section 3.3). I call a repository *inside* when at least half of the weight of its CPC labels lies in main groups where its firm was granted patents in the five grant years up to and including the year of its first activity, so that grants later in that year count, and I call those main groups

the firm's *patented main groups*. Inside records shared technology classes, not whether a patent covers the repository's code. Of the 102,987 repositories studied, 82,046 (79.7%) are inside, so the comparison is with the minority of the firm's repositories that lie outside its patented main groups. The main estimates use the labeled repositories of the firms with at least five patents and five labeled repositories in a five-year window ending in any year of 2019–2024, and one column takes every firm linked to a repository and a patent (section 3.4).

The paper makes three contributions. The first is a method that places open-source code in the patent classification, repository by repository. An LLM reads each repository's text and assigns it main groups in two stages, first subclasses, the level of the classification above the main group, and then the main groups under them (section 3.2). I check the method against two references built independently of it. The labels share at least one main group with those of the European Patent Office's (EPO) text categorizer, a tool that assigns patent classes to text, for 76.4% of the repositories both classify, against 38.5% when the label sets are dealt out at random among repositories of the same CPC section, the classification's broadest level. Run on patents, the same two stages with another LLM, DeepSeek-V4-Flash, give labels that share a main group with the examiners' classification for 89.1% of them, against 20.8% when the model's label sets are dealt out at random among patents of the same CPC section (Appendix B.1). Ganglmair and Kann (2025) place research disclosures in patent classes, and the May 2026 draft of Won (2026) infers a CPC profile for a released project. Here each repository of the firms in the data carries labels of its own (Appendix B.5).

The second contribution is the data. They link Compustat records to their patents and to their open-source code, and unlike the links that earlier studies built for their own use, each link records its supporting evidence and the year in which the firm owns the account. I group each firm with its subsidiaries under the top firm in its ownership tree that has a Compustat record, which holds both the patents and the GitHub organizations, and a repository belongs to the firm whose GitHub organizations hold it in the year of its first activity (section 3.1). The patent crosswalk covers 99.7% of the 2015–2021 patents that DISCERN (Arora et al., 2021) links to firms headquartered in the United States, and 96.3% of the 2015–2024 patents that the extended data of Kogan et al. (2017) link to CRSP firms. Both are among the crosswalk's inputs, so that agreement is partly built in. The Global Corporate Patent Dataset (GCPD) of Bena et al. (2017) and Barkemeyer et al. (2026) never entered the crosswalk, and where both link a 2015–2022 patent to a Compustat firm, GCPD names the same firm for 92.8% of the patents. The account crosswalk names the same firm as the commit-email data of Spinellis et al. (2020) for 93.9% of the accounts both hold. Among patenting firms with no accepted account, a search of GitHub by name finds an account with a matching name for 39.4% of them, which bounds from above the share of those firms whose accounts the crosswalk misses, provided the name search finds every such account (Appendix A.4).

The links also follow the firms' projects into US patent documents, whose references cite 0.65% of these firms' projects, against 0.179% of the projects of other firms in Compustat with the same mix of creation year, stars and language (Appendix D.1). This fact shows that the links reach

patent documents, and it enters no estimate of participation.

The third contribution is the result. Conditional on firm-by-year and main-group-by-year effects and controls for the repository's early activity, text and language, inside repositories have 29.5 percent more new participants in their first two years than other repositories (95 percent interval 14.6 to 46.4), and I call this difference the *inside difference*. Unlike the studies across firms cited above, the data here compare repositories by whether they lie in their own firm's patented main groups. Inside repositories also have 45.5 percent more contributions from new participants, counting their issues, pull requests, reviews and comments, and 37.6 percent more contributions from everyone, which is more activity overall, and among repositories with a contribution there is no clear difference in new participants' share of contributions (95 percent interval −0.52 to 2.55 points). For new participants with no earlier activity at the firm on GitHub, the inside difference is 27.6 percent. The paper's economic contribution is to document how participation aligns with firms' patented technologies, and project importance, audience and corporate support remain possible explanations of this conditional association. With firm-by-year and main-group-by-year effects, the main estimate compares a firm's repositories inside its patented main groups with its other repositories, and within a main group and year, one firm patents the main group and another does not, so the comparison is also partly between repositories of different firms (section 4.2).

Section 2 states what published perspectives predict for the inside difference. Section 3 builds the comparison from the linked data, and section 4 reports it, its main estimate first.

2 Literature review

The question is whether firms' open-source code in the technologies they patent draws more or fewer new participants, and the published perspectives point both ways. None of the perspectives below names patent classes or repositories, so each prediction for code in the firm's patented main groups states how the perspective maps to repositories.

2.1 Perspectives that predict less participation

Lin and Rai (2024) and the work they summarize predict less participation in code tied to patents. They suggest that patenting software could deter open communities from contributing, and they summarize earlier work as finding that projects tied to patented inventions attract less interest from users (Bar-Gill and Parchomovsky, 2003; Wen et al., 2013). They expect less participation at firms that put more weight on patenting. Wen et al. (2013) predict that user interest and developer activity fall in projects related to the technology that an enforcement action concerns, although this paper observes no enforcement. Under its mapping to repositories (Appendix Table 22), this line predicts fewer new participants in inside repositories.

2.2 Appropriability, products and visible contributions

Theory gives several reasons for firms to release open-source code. Firms may take their profit in a proprietary segment complementary to the released program (Lerner and Tirole, 2005), and some firms profit from products and services complementary to open-source products (Athey and Ellison, 2014). A sponsor may open a platform to drive its adoption and to harness outside developers (Economides and Katsamakas, 2006; Parker and Van Alstyne, 2018). Users in open-source communities typically reveal their innovations freely (von Hippel and von Krogh, 2003), and firms that contribute gain more from their use of open-source code than peers that do not (Nagle, 2018). Developers make open-source code available to everyone on equal terms, which Harhoff et al. (2003) call free revealing.

The first perspective rests on appropriability. Teece (1986) ties an innovator's ability to capture the profits from an innovation to its appropriability regime, and in his view commercializing an innovation needs its know-how to be used together with other capabilities or assets. Later work carries this to the release of code. Alexy et al. (2013) define selective revealing as the release of knowledge that the firm could have kept proprietary, and Henkel (2006), citing Teece, names complementary assets as a means of appropriation that works whatever the license of the released code. Fosfuri et al. (2008) argue that a firm's stock of software patents can be such a complementary asset to its open-source products. Under its mapping to repositories (Appendix Table 22), the firm's patents mark where its code is better supported, and this perspective predicts more participation in inside repositories. These sources explain what firms release, not who takes part in it, and Fosfuri et al. (2008) note that a patenting firm's tight control of a project might discourage independent developers. On SourceForge, Belenzon and Schankerman (2015) study who takes part in open-source projects from observed membership and contributions, and they find that developers sort on a project's license, size, intended audience and corporate sponsorship.

The second perspective concerns code tied to products. Upstream knowledge can be public while the downstream products built on it are protected (Gambardella and Hall, 2006), and commercial firms support open-source projects (Athey and Ellison, 2014). Appendix Table 22 gives this perspective's mapping to repositories.

The third perspective concerns visible contributions. Theories of participation explain why developers contribute, for delayed rewards that signal their talent (Lerner and Tirole, 2002) or out of reciprocal altruism (Athey and Ellison, 2014). In Lerner and Tirole (2002), the signal is stronger the more visible a performance is to the relevant audience and the more attributable it is to one person, and in open source, outsiders can see what each person contributed and whether it worked. On GitHub, an issue or a pull request carries work that others can see and judge, since a pull request makes a change visible for review (Kalliamvakou et al., 2015), actions on the platform are public, and others read a person's actions there as signals of competence (Dabbish et al., 2012; Pinto et al., 2018). A star carries no such work. It is tied to an account, but developers give stars mainly to show appreciation or to bookmark a project (Borges and Valente, 2018), and studies count them as popularity, apart from contributions (Pinto et al., 2018). Under its mapping to repositories

(Appendix Table 22), this perspective predicts a larger inside difference on issues and pull requests than on stars.

2.3 Related studies

Empirical studies relate firms' open-source code to their patents across firms. Firms with larger stocks of software patents release more open-source products (Fosfuri et al., 2008). Lin and Rai (2024) find that software firms' open-source activity rose after the Supreme Court's 2014 *Alice* decision narrowed the scope of software patents, mainly among firms with a large stock of software patents beforehand. Other studies measure firms' use of and investment in open-source code (Gortmaker, 2025), place firms in a patent-technology space (Habibi, 2025), relate firms' patent profiles to the level of their GitHub activity (Cao et al., 2026), and value open-source projects from market reactions (Emery et al., 2025).

The closest papers on firms' code and patents relate the two in other ways. Appendix D.1 sets Petralia (2025) and Petralia and Boschma (2026), on patents that mention a project, beside this paper's counts of projects cited in patent documents. For the participation result, Lin and Rai (2024) relate a firm's open-source activity to the scope of software patents across firms and industries. Here repositories are compared by their overlap with their own firm's grants, and the comparison is partly across firms. Wen et al. (2016) relate start-ups' entry with open-source products to IBM's pledge of patents, market by market and year, through a concordance from patent classes to software markets. Here the tie is drawn repository by repository within the firm, with no dated event. Chen et al. (2022) and Haese and Peukert (2026) follow contributions by a firm's own developers and by outsiders around a dated event, an acquisition or Microsoft's move to Chromium, and assign developers to firms by membership lists or email domains. Here no event is used and no actor is matched to an employer.

3 The data and the Cooperative Patent Classification

I link Compustat firms to their open-source code on GitHub and to their USPTO patents, and an LLM assigns patent classes to the code. Which repositories are compared, and how is each placed against its firm's patents?

3.1 Firms and their repositories

I start from the firms in Compustat and combine four sources (Appendix A.1) at the level of the firm defined below. Compustat supplies the list of firms and each firm's industry. A *Compustat record* is one firm's entry in Compustat, which may be a subsidiary's. The *Compustat record firm* is the firm with a Compustat record at the top of the ownership tree, together with its subsidiaries (Figure 1). It holds both the patents and the GitHub organizations of the firm and its subsidiaries. A repository belongs to the Compustat record firm whose organizations hold it in the year of its first

recorded activity, and the repository's participants are the people who contribute to it (section 3.3). From here on, *the firm* means the Compustat record firm.

The *linked firms* are the firms that the account crosswalk, which links Compustat records to GitHub organizations, links to at least one organization (Appendix A.4). The firms studied are the linked firms with at least five patents and at least five labeled open-source projects in a five-year window ending in any year of 2019–2024 (section 3.4). I call them the open-source-software-active firms, or *OSS-active firms*.

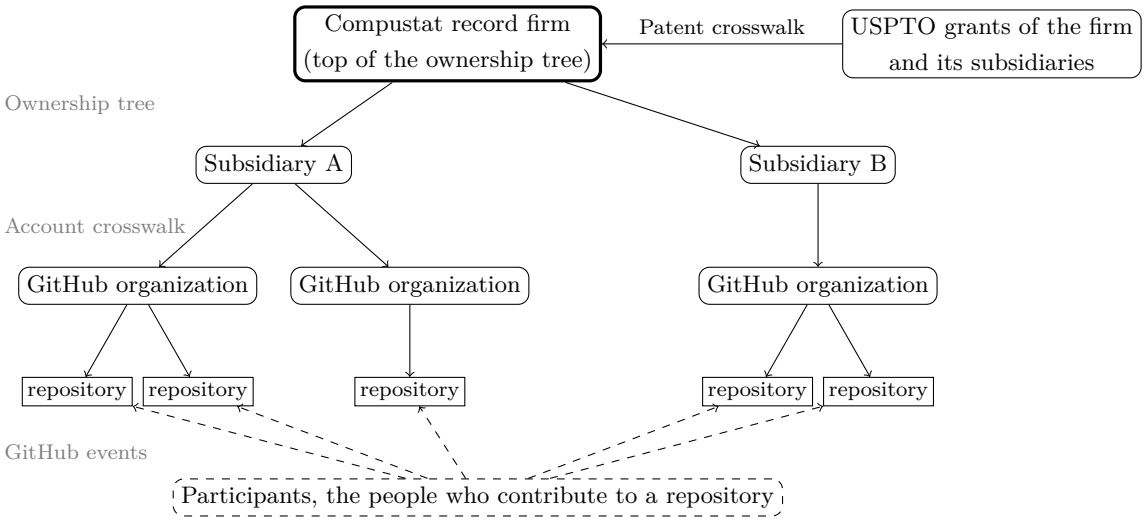


Figure 1: From a Compustat record firm to its repositories and their participants

Notes: The Compustat record firm is the firm with a Compustat record at the top of the ownership tree. The patents of the firm and its subsidiaries are assigned to it through the patent crosswalk, and the account crosswalk links each GitHub organization to the Compustat record that runs it (section 3.1). A repository belongs to the Compustat record firm whose organizations hold it in the year of its first recorded activity. Its participants are the people who contribute to it (section 3.3).

GitHub was the largest host of source code (Conti et al., 2025) and by far the most popular platform for open-source development (Lin and Rai, 2024). An account on GitHub belongs to a user or to an organization, and as in Conti et al. (2025), the accounts linked here are mostly organization accounts (Appendix A.4). A repository holds the code and history of a project under an account, though a repository is not necessarily a project (Kalliamvakou et al., 2016). I use repository and project interchangeably. The events come from GH Archive, a public record of GitHub's public timeline (Conti et al., 2025), and a project's activity is its dated events from 2015 onward.[1] Events are extracted only for the GitHub organizations that the crosswalk links to firms, and I call an event in that extract a *held* event. The account crosswalk links the firms to their organizations, as Cao et al. (2026) link Compustat firms to theirs. Appendix A.2 lists what the GitHub data leave out.

Neither Compustat nor the patent records list a firm's GitHub accounts, so I build the link

[1]The events include pushes, issues, pull requests, releases, stars and forks of the project. The archive's yearly files do not correspond to calendar years. They lack most of January 2018, 2019 and 2020 and include January 2021 twice, and I use corrected event dates throughout.

from firms to their open-source code. The account crosswalk also dates its links. *Dated ownership* is a record, for each year, of the firm that owns a project's account, and so it follows repositories through acquisitions and the ends of firms' records. A second crosswalk, built separately, links Compustat records to PatentsView assignee identifiers. An LLM also takes part in both crosswalks. Appendix A.4 describes the search for the accounts, checks both crosswalks against published ones and follows the counts along the chain.

3.2 Classifying repositories by patent class

The patents are the firms' utility grants of 2015–2024 from PatentsView, the United States Patent and Trademark Office's (USPTO) public data, each dated by its grant year. The data hold grants from 2015 on. Every timing rule below uses grant years, and no version on application years is estimated. The Cooperative Patent Classification (CPC) runs from sections through subclasses and main groups to subgroups. A patent's inventive CPC codes are those marked as inventive information, as distinct from additional information, and each patent's weight is split equally across the main groups of its inventive codes. A main group is one of the firm's patented main groups for a repository when any of the firm's grants in the five grant years up to the year of the repository's first activity carries weight in it, however small (section 4.1). Because the data hold grants from 2015 on, the patented main groups of the earliest repositories rest on fewer than five grant years.

Open-source code carries no patent classification, so I assign patent classes to compare it with a firm's patents. A prompted LLM can assign main groups to any project without word lists or labeled examples. The model is GLM-5.3-Flash, which I call *the labeler*, and it assigns the repository's text to classes of the classification used for patents. It reads the project's name, description, topics, programming language and README, with long READMEs truncated. It labels in two stages, first CPC subclasses and then the main groups under them (Figure 2), and Appendix B.4 gives each stage. Two other LLMs serve only as checks. DeepSeek-V4-Flash labels patents, so that its labels can be compared with the examiners' (Appendix B.6), and it also labels a sample of repositories (Appendix C.3). An anonymous model served under the name `space-bunny-alpha`, which I call *the second labeler*, labels a sample of the repositories (Appendices B.1 and C.3).

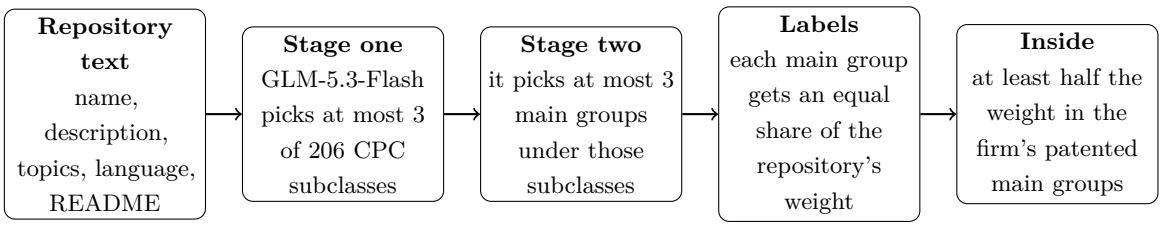


Figure 2: How a repository is labeled

Notes: The model reads the text of the 209,352 projects linked to firms, as it stood in 2026, at temperature zero. Either stage may return no applicable class, which leaves the repository unlabeled, and 188,899 of the projects receive at least one main group. The firm's patented main groups are those of its grants in the five years up to the repository's first-activity year (section 4.1).

A repository is inside when at least half of its label weight lies in its firm's patented main groups. Each main group the model returns carries an equal share of the project's weight, so the ranking of the labels leaves their weight unchanged. Of the 209,352 projects linked to firms, 188,899 receive at least one main group. The model reads the repositories' text as it stood in 2026, after every outcome window. I define inside at the level of the main group because the firms' open-source code is concentrated in a few classes (Appendix A.7). At the level of the CPC subclass, G06F alone holds 72.56% of the firms' open-source code, so most repositories of a firm that patents in G06F would share a class with its patents whatever they do. At the main group, the same code spreads over 676 main groups and takes 11 of them to reach 80 percent.

One repository, the running example, shows the steps (Appendices A.9 and B.8). The repository `facebookresearch/fastMRI` belongs to Meta Platforms, one of the OSS-active firms, and its first activity falls in the fourth quarter of 2018. The labeler read its name, description, topics, programming language and README, and returned three main groups from the subclasses G06N, G06T and G16H, each with a third of the repository's weight. They are G06N 20/00 (machine learning), G06T 5/00 (image enhancement or restoration) and G16H 30/00 (handling of medical images). Meta was granted patents in G06N 20/00 and G06T 5/00 in 2015–2018 and none in G16H 30/00, so two thirds of the repository's weight lies in Meta's patented main groups, and the repository is inside.

I check the labels against two references built independently of them (Appendix Table 13). The first reference is the European Patent Office's text categorizer, a tool that assigns patent classes to text, run on the same repositories. The second is the examiners' classification of patents. For it, DeepSeek-V4-Flash applies the same two stages to the titles and abstracts of patents, and its labels are set against the classes the examiners assigned, as Lee and Hsiang (2020) score a patent classifier. A set of labels agrees with a reference when the two share at least one main group. The checks measure agreement with other classifications rather than accuracy, and Appendix B.1 reports them in full and shows how far a label error can move a repository across the inside threshold.

3.3 New participants and the outcomes

Participation is measured from GH Archive's record of GitHub's public event stream, held for the organizations the crosswalk links to firms. The held record has one row per organization, repository, event type, calendar quarter and actor, with the number of such events. The record holds fifteen event types, which fall into three families (Appendix Table 8). Attention is starring and forking. Contribution is opening, closing and commenting on issues, opening, closing, reviewing and commenting on pull requests, and commenting on commits. Write and administration covers pushes and the other actions that need write access to the repository.

Participation in a repository is a contribution-family event on it by an actor who is not a bot, and the repository's *participants* are the people who contribute to it in this way. Wright et al. (2023) measure participation by new commits, but these records hold commits only inside push events, which fall in the write family. I do not count pushes and the other write events as participation,

because they need write access to the repository, which the firm grants, whereas contribution-family events are open to anyone with a GitHub account. Write events enter two of the outcomes below only, whether new participants keep working on a repository and whether the repository is still updated, and no estimate counts pushes as participation.

The held record shows the work people leave on a repository and some signs of attention, its stars and forks, while reading, downloading and running its code leave no trace in it. Studies of user innovation describe a project's contributors as the part of its users who go on to post or change code, beside users who take the code and contribute nothing (von Hippel, 2005), readers of a help forum who never post (Lakhani and von Hippel, 2003) and the silent subscribers of a project's mailing lists (von Krogh et al., 2003). In these terms, a repository's participants are the users who leave contribution-family work on it. A screen of actors' names and behavior flags bots, which are dropped from every count (Appendix A.1).

A *new participant* on a repository has no held event of any type through its first-activity quarter, and makes a first contribution-family event on it in the following eight quarters. A repository's clock starts at its *first-activity quarter* q_0, the first calendar quarter with a held event by an actor who is not a bot, which the text also calls the repository's first activity. The year of q_0 is the repository's *first-activity year* y_0. The *window* is the 24 months after q_0, quarters $q_0 + 1$ to $q_0 + 8$.

I fix the window's length so that every repository is counted over the same span of its life, whatever its year of first activity, and a repository first active early does not gather more participants only because it is observed longer. Two years is the longest span the records cover in full for every repository first active through 2022, since they end in 2024. On the repositories studied that were first active in 2015–2020, whose four years after q_0 are all held, the window holds 53.0 percent of the people who first contribute to the repository within those four years, just over half, and between 49.8 and 57.0 percent in each year of first activity. Medappa et al. (2025) likewise judge a GitHub project's survival over the two years from its start, a horizon they base on earlier evidence that many open-source projects stop within their first two years. Pinto et al. (2018) count the newcomers to company projects in the six months after the projects become public. No study among those cited here counts new participants over this window, so its start at first activity, its length and its quarterly grain are this paper's own choice.

The first-activity quarter is the first held event under a linked organization, not the repository's creation. The regressions carry indicators for a repository created in the four quarters up to q_0 and for one with no creation record (section 3.4), and Appendix A.4 counts each case. Actors with a held event of any type on the repository in q_0, stars and forks included, are its *founding-quarter actors*. I do not count them as new, because quarterly data cannot order events within q_0 and so cannot separate them from the people who published the repository. Inside repositories have more of them on average (Table 3), and the regressions control for their number.

A new participant's *arrival* is their first held event of any type on the repository, which can come before their first contribution, and I count arrivals as person–repository pairs. The software-engineering literature calls such people newcomers (Pinto et al., 2018; Casalnuovo et al., 2015).

New participants follow the bazaar of Raymond (1998), in which a project's users become its co-developers. Raymond describes the early Linux community as "a great babbling bazaar of differing agendas and approaches," and he writes of a project's users that, "properly cultivated, they can become co-developers." In the layers that Crowston and Howison (2005) describe around a project's core developers, co-developers submit patches and active users report bugs, while passive users "do not speak on the project's lists or forums." The contribution family records the work of co-developers and active users, and the passive users are those the held record leaves unseen.

Several studies of open-source projects, firms' projects among them, count new contributors as part of the community around a project and describe its continued activity as depending on them. Pinto et al. (2018) write that "going open source means creating, maintaining, and fostering a community around technology," for example by attracting new developers, and they count the new contributors to company projects after the code becomes public. Haese and Peukert (2026) study "the open source community behind" a firm-led project through its external contributors, among them newcomers with no history of contributing to it. Steinmacher et al. (2015) write that open collaboration communities "leverage and depend on new contributors," and Zhou and Mockus (2015) that projects "need ongoing contributions from new and existing participants." Valiev et al. (2018) count a project as sustainable from a community perspective if it attracts and retains newcomers, and in their survival models of Python packages, packages with more contributors are less likely to become dormant. Yamashita et al. (2016) likewise measure a GitHub project's attraction of developers by the share of its contributors in a period who did not contribute before, and Nagaraj and Piezunka (2024) a volunteer community's recruitment by its first-time contributors in each quarter. New participants count these arrivals, and long-term contribution follows whether they stay.

Two questions about a new participant's held events before arriving split the new participants further. The first is whether the person was *active at the firm*, with a held event of any type on the firm's organizations at any time before the first held event on the repository. *Participants new to the firm* are the new participants who were not active at the firm. The second is whether the person was *active at another linked firm*, with a held event on another linked firm's organizations in the four quarters before arriving, and one outcome counts the participants new to the firm who were active at another linked firm. The two questions give four groups, (a) to (d) in Table 1. A person active at both the firm and another linked firm, group (b), counts among the new participants but not among those new to the firm. On the reference set, the labeled repositories with a single firm first active in 2015–2022 that the main estimates use (section 3.4), group (b) holds 251,114 of the 1,385,308 new participants' arrivals. The groups are defined by held events alone. No actor is matched to an employer, so a participant new to the firm is new to its organizations on GitHub, not known to come from outside the firm. Appendix A.5 gives the rules for bots and for actors who may be the firm's staff.

Table 1: New participants by whether they were active at the firm (at any time before) and at another linked firm (in the four quarters before)

	Active at another linked firm in the four quarters before arriving		
Active at the firm before arriving	No	Yes	All
Yes	(a) 270,345	(b) 251,114	521,459
No, the participants new to the firm	(d) 621,168	(c) 242,681	863,849
All new participants, (a) to (d)	891,513	493,795	1,385,308
Founding-quarter actors, never new			2,712,361

Notes: The numbers are person–repository pairs on the reference set, over each repository's 24-month window. A person is active at the firm with a held event of any type on the firm's GitHub organizations at any time before the first held event on the repository, and active at another linked firm with a held event on another linked firm's organizations in the four quarters before it. The four cells do not overlap and together hold every new participant. Cell (a) is active at the firm only, (b) at both, (c) at another linked firm only, and (d) at neither. The outcomes count cells (a) to (d), the participants new to the firm in (c) and (d), and those of them active at another linked firm in (c). The founding-quarter actors, active in q_0, fall in no cell. No actor is matched to an employer, so the people in (a) and (b) are not known to be the firm's staff, and those in (c) and (d) are not known to come from outside the firm. In distinct people, the new participants are 853,648, and (a) to (d) hold 155,481, 130,829, 185,433 and 566,961, which sum to more than the total because a person new on several repositories can fall in different cells on each. The founding-quarter actors are 881,757 distinct people.

Outcome variables. The outcomes are counted in the window, except long-term contribution, which follows each arrival past it. *Contributions* are the contribution-family events of new participants. These are issues, issue comments, pull requests, pull-request reviews and review comments, and commit comments, the ways in which people without write access report problems and submit code for consideration (Medappa et al., 2025). *Stars* are watch events by actors whose first held event of any type on the repository falls in the window, whom I call *new users*. A watch event is the starring of a repository (Conti et al., 2025), and developers star repositories mainly to show appreciation and to bookmark them (Borges and Valente, 2018). Stars are a common measure of a project's popularity (Zhou et al., 2020). Those studies count all of a repository's stars, while this count keeps only the actors new to the repository in the window.

Long-term contribution is the share of new participants arriving within 12 months who have a contribution- or write-family event on the repository in quarters 4 to 7 after arriving, on repositories first active in 2015–2021. Unlike the contributions count, it also counts write-family events. The name follows the term "long-term contributor" in Casalnuovo et al. (2015) and Zhang et al. (2021), and Haese and Peukert (2026) likewise follow which contributors continue and how many newcomers arrive. Zhou and Mockus (2015) define a long-term contributor as one who stays with a project for at least three years and stays productive. The window of quarters 4 to 7 after arriving is shorter than three years, and it is this paper's own. An arrival in quarter $q_0 + 4$ is followed to quarter $q_0 + 11$, so long-term contribution runs past the 24-month window, and only repositories first active through 2021 enter, so that every arrival is followed in full. Its unit is the new participant's arrival,

651,778 arrivals on the reference set, with standard errors clustered by firm.

A repository is *maintained* if it has a push or a release in its second year, quarters $q_0 + 5$ to $q_0 + 8$. Medappa et al. (2025) count a project as inactive when no code changes are made within two years of its start, and the maintained indicator is close to the opposite of that rule. Coelho et al. (2018) likewise count a project with a recent release as under maintenance. The maintained indicator is estimated, like long-term contribution, by a linear probability model, and it has its own row in Table 4.

3.4 The regression dataset

The regression dataset holds repositories, each assigned to its Compustat record firm in the year of its first held activity (section 3.1). Among repositories with a held event that is not a bot's, 111,009 repositories first active in 2015–2022 each have a firm, among 695 firms. That range leaves at least eight quarters of activity before the data end in 2024. A repository has a *single firm* when only one OSS-active firm's organizations hold it in its first-activity year (Appendix A.3). The base sample, the *reference set*, is the 102,987 labeled repositories with a single firm (Table 2). Columns 1, 2, 4 and 6 of Table 4 draw their rows from the reference set, column 3 widens the firms, and Table 4 gives each column's count. Appendix A traces every step from the event records to each estimation sample.

The firms are the OSS-active firms of section 3.1, those with a five-year window ending in any year of 2019–2024 that holds at least five patents and at least five labeled open-source projects. On dated ownership, 713 firms meet that condition, and these are the OSS-active firms. Appendix A.3 reconciles this count with the firms that the crosswalk's current links give. With the crosswalk's links extended to grants of 2011–2014, a window can end in any year of 2016–2024, and 761 firms meet that condition. The estimates here keep the windows ending in 2019–2024.

For the grants of 2011–2014, the crosswalk flags the owning entity as unknown for 19.2% of the links, against 9.4% for 2015–2024. On the repositories linked to the firms in 2011–2014, the labels cover 84%, 89%, 84% and 90% of the events in 2011, 2012, 2013 and 2014. For the events of 2012–2014 that carry no account id, the bot screen reads the account's login. Event counts can be compared across 2014 and 2015 within two limits, a step in the events per grouped record, and events with no account id on logins that the later screen does not cover.

Column 3 of Table 4 adds every firm the crosswalks link to a repository and a patent, with 1,110 firms in its sample. Inside is defined from the firm's patents in the five grant years up to the repository's start, so each repository needs such patents. On the wider set, 32,451 repositories of 1,454 firms have no such patents and enter no comparison. This group is defined repository by repository, so its firms are not only firms without patents. Some of them are granted patents in other years, and some also own repositories that enter the comparison. The main estimates keep the OSS-active firms. The linked data were built and checked on them, and with at least five patents and five labeled repositories in the window, inside is well defined for each firm.

Participation in the regression dataset is large in total (Table 2, Panel B). Over the 24-month

windows of the 111,009 repositories first active in 2015–2022, an actor's first held event on a repository occurs in 9,826,995 actor–repository pairs. Of these, 1,402,368 are new participants' arrivals, by 861,804 distinct people, and the rest are actors with no contribution-family event on the repository in the window. On the reference set, new participants arrive 1,385,308 times, by 853,648 distinct people. Of the new participants' arrivals on the reference set, 863,849, by 710,484 people, are new to the firm, and 242,681, by 185,433 people, are new to the firm and active at another linked firm. The outcomes are described repository by repository, and Table 3 summarizes them.

Table 2: The regression dataset

	Repositories	Firms
Panel A. Repositories and firms		
With a held event, owned by an OSS-active firm at first activity	141,224	
first active in 2015–2022	111,009	695
labeled, with a single firm (the reference set)	102,987	693
of which inside	82,046	
Columns 1 and 2 of Table 4	102,980	686
Column 3, every linked firm with a patent	107,018	1,110
Column 4 of Table 4, its estimate for new participants	96,854	626
Panel B. Participation		
	Events or pairs	Distinct people
On the repositories first active in 2015–2022		
Actors' first held events on a repository in its window	9,826,995	
New participants' arrivals	1,402,368	861,804
On the reference set		
Held events of every type, 2015–2024	114,936,122	5,994,624
Contribution-family events, 2015–2024	62,496,484	2,002,193
Push events, 2015–2024	18,625,845	
New participants' arrivals	1,385,308	853,648

Notes: Panel A counts repositories with a held event that is not a bot's, each assigned to the OSS-active firm that holds it in the first-activity year. Columns 1 and 2 drop firms with a single repository, and column 3 draws its repositories from every firm linked to a repository and a patent. Column 4 also needs the main group of the repository's top-ranked label and drops the cells that its effects leave with one repository, which leaves 98,795 repositories of 667 firms for the two shares of Table 4. Each Poisson estimate further drops the firms or cells whose outcome is zero throughout, and column 4's row gives the count for new participants after that drop (Appendix A). Panel B counts held events by actors who are not bots, the founding-quarter actors included, over all held quarters (section 3.3). The window is a repository's 24 months after its first-activity quarter, and an arrival is one person's first held event on one repository. The records hold pushes, not the commits inside them.

Most repositories lie inside their firm's patented main groups. Of the 102,987 labeled repositories with a single firm, 82,046 (79.7%) lie inside their firm's patented main groups, as defined just before Table 3. Section 4 therefore compares the inside repositories with the minority that lie outside, conditional on firm-by-year and main-group-by-year effects and on controls for the repository's early

activity, text and language, defined below.

Many repositories have no new participant, and these zeros stay in the comparison. Within two years of first activity, 53.5 percent of these repositories have at least one new participant, and 41.1 percent have one with no earlier held event on the firm's GitHub organizations (section 3.3). The rest have none. The estimator of section 4.1, Poisson pseudo-maximum likelihood, keeps these repositories in the comparison rather than dropping them or adding a constant before taking logarithms. It still drops the firms, or the groups of repositories that share a firm and year or a main group and year, that hold a single repository or in which no repository has a new participant (Table 2).

Controls. The regressions control for the repository's activity at its start, its text and its language. The *founding controls* are measured at or before the first-activity quarter, except the length of the text, which comes from the text the labeler read, as it stood in 2026 (Appendix A.4). They are the logarithms of one plus the events in q_0 of each family (attention, contribution, and write and administration), of the number of founding-quarter actors, and of the length of the text the labeler read. The controls also include an indicator that the repository was created in the four quarters up to q_0, indicators for repositories with no creation or text record, and language effects for the 20 most common languages, the others grouped.

Table 3 summarizes the outcomes and the founding controls over all, inside and other repositories. A repository is inside when at least half of its label weight, split equally over its labels, lies in main groups in which its firm was granted patents in the five grant years up to its first-activity year. The running example, fastMRI, is inside, since two of its three labels lie in Meta's patented main groups (section 3.2). Inside repositories have more new participants on average than the others, before any comparison within the firm, 14.36 against 9.89.

Table 3: Summary statistics

		All		Inside		Other	
	N	Mean	SD	Mean	SD	Mean	SD
Panel A. Outcomes, in the 24 months after the first-activity quarter							
All new participants	102,987	13.45	147.94	14.36	160.59	9.89	81.13
new to the firm, no earlier activity there	102,987	8.39	114.35	8.69	122.96	7.21	71.16
of those, active at another linked firm in the prior year	102,987	2.36	30.4	2.24	30.44	2.81	30.25
Contributions by new participants	102,987	110.24	1098.11	119.82	1187.03	72.75	638.81
Stars by new users	102,987	68.76	716.36	71.95	760.6	56.28	506.95
Maintained in the second year (0/1)	102,987	0.422	0.494	0.43	0.495	0.394	0.489
Long-term contribution, per arrival (0/1)	651,778	0.115	0.319	0.116	0.32	0.11	0.313
Panel B. Inside and the founding controls							
Inside the patented main groups (0/1)	102,987	0.797	0.402	—	—	—	—
Attention events in q_0	102,987	25.19	286.02	26.6	304.82	19.67	195.56
Contribution events in q_0	102,987	19.41	126.63	20.68	135.17	14.41	85.12
Write events in q_0	102,987	18.98	75.61	19.27	80.37	17.82	52.95
Founding-quarter actors	102,987	26.34	275.55	27.75	293.85	20.8	187.29
Text length (characters)	102,987	2951.38	2218.62	3029.2	2219.18	2646.49	2,190
Repositories			102,987		82,046		20,941

Notes: The rows are the reference set's repositories (Table 2), except the long-term-contribution row, which has one row per new participant who arrives within 12 months on a repository first active in 2015–2021. Section 3.3 defines each outcome, and section 4.1 defines inside, so the table shows only inside's overall mean and standard deviation. The prior year is the four quarters before arriving. The founding controls enter the regressions as the logarithm of one plus each. Panel B does not show the indicator for a repository created near its first activity, the indicators for no creation or text record, or the language effects, which also enter as controls. Appendix Table 11 gives each variable's median, upper percentiles, maximum and total.

4 Participation where code and patents overlap

The firms are the OSS-active firms of section 3.1, and 693 of them hold the 102,987 repositories of the reference set (Table 2), Meta Platforms, which holds the running example, among them. By SIC division, 318 of these firms are in services and hold 67.4% of the repositories, and 258 are in manufacturing and hold 19.4%. The five largest by repositories, Microsoft, IBM, Amazon, Alphabet and Cisco, hold 33.7% of the repositories.

The literature of section 2.2 gives reasons a firm may release open-source code. A firm may take its profit in products, services or assets complementary to the released code (Lerner and Tirole, 2005; Athey and Ellison, 2014; Henkel, 2006), or open a platform to drive its adoption and draw in outside developers (Economides and Katsamakas, 2006; Parker and Van Alstyne, 2018). Outside developers may in turn contribute where others can see their work (Lerner and Tirole, 2002). These are the literature's reasons, and the estimates below do not show why any firm releases code.

This section uses the data to ask whether repositories inside their firm's patented main groups

have more new participants over their first two years than other repositories, conditional on firm-by-year and main-group-by-year effects and the founding controls. The firm-by-year effects hold fixed what differs between firms in a year, such as size, reputation and how the firm runs its open-source program, and the main-group-by-year effects what differs between main groups in a year.

In the main estimate, a firm's repositories in the technologies it patents (inside its patented main groups) have more new participants, and more contributions from them, than other repositories, and the comparison is partly across firms. These repositories also have 29.3 percent more contributions from others, and the proportional differences for new participants and others are not separated from each other (section 4.2). These are conditional associations.

4.1 Estimation strategy

The comparison sets a firm's repositories inside its patented main groups against its other repositories, and the outcomes count, over each repository's first two years, its new participants, their contributions and the stars of new users (section 3.3). I estimate the difference between inside and other repositories by Poisson pseudo-maximum likelihood (PPML). The reasons come from the outcomes, which are counts with a long tail and many zeros, and from the users the records leave unseen.

The counts are uneven across repositories. On the reference set, the median repository has 1 new participant and the 90th percentile 18, while the mean is 13.45 and the largest repository has 23,766. Among all the repositories first active in 2015–2022, the largest one percent hold 50.7 percent of the new participants, and the largest ten percent hold 85.4 percent.

Zeros are common in these counts (section 3.4), and a model in which inside multiplies the expected count fits both the long tail and the zeros. Under an exponential mean, the expected count of inside repositories is e^{β} times that of other repositories for a coefficient β on inside, conditional on the effects and controls. This ratio is the same for small and large repositories, and the exponential keeps the expected count positive. The main estimate keeps the largest repositories, and the exponential mean gives them the weight of their counts. Without the five largest firms, its 29.5 percent for new participants and 45.5 percent for contributions are 28.1 and 40.1 percent, and without the top one percent of repositories by founding-quarter activity they are 19.7 and 40.8 percent. The five firms therefore do not carry the estimates, while the estimate for new participants falls by about a third without the repositories with the most founding activity. The intervals overlap, and no test compares the runs. A linear model of the count would give a difference in levels, and a model of the logarithm cannot take the zeros without an added constant.

PPML estimates such a model with the zeros kept, and two of its properties matter here. PPML is consistent when the conditional mean is right, whether or not the counts are Poisson, and conditioning on each firm-and-year total removes the firm-by-year effect from the likelihood, as conditioning on each firm's total removes a firm effect (Santos Silva and Tenreyro, 2006; Wooldridge, 1999). Standard errors are clustered by firm, which allows any dependence among one firm's repositories, and Appendix C.1 gives the derivation.

A repository's users are unseen (section 3.3), and they enter its expected count as a factor

of a product. The expected number of new participants over the window is the number of the repository's users during the window times the share of them who leave contribution-family work on it. In logarithms the product is a sum, so if each factor is an exponential function of inside status, the effects and the controls, the product is one too, and β is the sum of the two factors' log ratios. The firm-by-year and main-group-by-year effects absorb the part of the users common to a firm and year or to a main group and year, since conditioning on each group's total removes a multiplicative effect (Wooldridge, 1999). Variation in the users within those cells, where it moves with inside status, stays in β. The users and the share enter the expected count as a product, so I keep the exponential mean for this reason as well as for the zeros and the uneven counts.

I therefore estimate the model by PPML, with firm-by-year and main-group-by-year effects and the founding controls. For each count outcome y_r of section 3.3 and a repository r of firm p, first active in year y_0, whose top-ranked label lies in main group $g(r)$, its *primary main group*, the model of the main estimate is

$$E\left[y_r \mid \text{Inside}(r), p, g(r), y_0, X_r\right] = \exp\left(\beta\,\text{Inside}(r) + \alpha_{py_0} + \lambda_{g(r)y_0} + X_r'\delta\right). \tag{1}$$

The left-hand side is the expected count of the outcome for repository r, given whether it lies inside, its firm p, the Compustat record at the top of its ownership tree (section 3.1), its primary main group $g(r)$, its first-activity year y_0 and its founding controls X_r. The model describes the mean of the count given these terms, and Inside(r) is among them because β compares the mean of an inside repository with that of a repository outside that shares its firm-by-year level, its main-group-by-year level and its controls. The firm's patented main groups are the main groups in which the firm was granted a patent in the five grant years up to y_0, and Inside(r) indicates that at least half of the repository's label weight, with equal weights on its labels, lies in those main groups. Here α_{py_0} is a firm-by-year effect, $\lambda_{g(r)y_0}$ a main-group-by-year effect, the year being that of first activity, and X_r the founding controls of section 3.4 with the language effects. PPML estimates β on the counts as they are, and only the estimate is transformed afterwards, to $100(e^{\beta} - 1)$, the percent difference in the expected count between inside and other repositories, with its standard error on the same scale. The two shares, maintenance and long-term contribution, come from linear probability models with the same effects as the counts, and the differences they give are in percentage points.

4.2 Baseline results

If the appropriability perspective of section 2.2 holds, as Appendix Table 22 maps it to repositories, inside repositories are expected to have more new participants than the firm's other repositories. If the line of section 2.1 holds, so that code tied to patents draws less participation, inside repositories are expected to have fewer. The other perspectives of section 2.2 predict how the inside difference varies across repositories and participants, which section 4.3 reads. An estimate that matches a prediction is consistent with its perspective and does not establish it.

In the main estimate, column 4 of Table 4, inside repositories have 29.5 percent more new

participants than other repositories, conditional on firm-by-year and main-group-by-year effects and the founding controls (95 percent interval 14.6 to 46.4) and 45.5 percent more contributions from them. Averaged over inside repositories, the difference from the specified comparison is 3.34 new participants in the 24 months (95 percent interval 1.86 to 4.64). This average comes from a skewed count, since 43.7% of inside repositories have no new participant, and the difference is 1.68 without the top one percent of repositories by founding-quarter activity. For participants new to the firm the difference is 27.6 percent, and it is 21.9 percent for participants new to the firm active at another linked firm in the four quarters before arriving.

The six columns of Table 4 differ in what they compare, and I call a column's effects and controls its *form*. Columns 1 to 3 compare a firm's inside and other repositories with firm and year effects, column 2 with the founding controls and column 3 on the wider set of firms. Column 4 has firm-by-year, main-group-by-year and programming-language effects with the founding controls, so the repositories compared need not share both the firm and the main group, and its comparison is partly across firms (Appendix C.2). In its estimation sample, 930 of the 1,181 main-group-by-year cells hold both inside and other repositories, with a median of 9 firms a cell, and 921 of them have two or more firms. Column 5 keeps column 4's form and draws inside at the CPC subgroup, and column 6 drops the main-group-by-year effects, so it compares repositories of one firm first active in one year. I call an estimate *significant* when its p-value, adjusted by Holm's method for the outcomes tested together (Holm, 1979) (Table 4's notes), is below five percent, and two or three stars mark such an estimate.

Table 4: Participation in a firm's repositories inside and outside its patented main groups

	(1) Firm and year effects	(2) With founding controls	(3) Every linked firm	**(4) Main-group-by-year effects**	(5) Inside at the subgroup	(6) No main-group effects
				Main estimate		
Panel A. New participants within 24 months, inside against other repositories						
All new participants (%)	77.3***	34.3***	79.9***	**29.5*** **	27.8***	36.8***
	(25.7)	(11.1)	(22.4)	(8.1)	(7.0)	(9.7)
new to the firm, no earlier activity there (%)	76.6***	30.5**	79.8***	27.6***	35.8***	33.3***
	(27.9)	(12.4)	(26.2)	(9.9)	(9.0)	(11.7)
of those, active at another linked firm in the prior year (%)	69.4***	26.2**	73.0***	21.9***	30.8***	25.5**
	(28.0)	(12.7)	(27.0)	(8.0)	(9.0)	(10.9)
Panel B. Contributions within 24 months, inside against other repositories						
Contributions by new participants (%)	84.5***	28.1**	108.3***	**45.5*** **	21.6*	41.4***
	(27.6)	(12.8)	(26.6)	(12.9)	(11.0)	(13.8)
All contributions (%)	81.2***	19.5**	99.9***	**37.6*** **	15.4	34.7***
	(22.9)	(8.7)	(24.0)	(10.2)	(8.4)	(9.3)
Contributions by others (%)	77.5***	11.4	91.6***	29.3***	9.2	27.7***
	(21.9)	(8.9)	(24.6)	(11.5)	(8.0)	(9.3)
Panel C. Stars, maintenance and long-term contribution						
Stars by new users (%)	44.0*	19.5*	42.6*	7.1	11.4*	16.0
	(28.8)	(12.3)	(29.5)	(11.3)	(7.3)	(11.6)
Maintained in the second year (points)	6.4***	3.8***	5.6***	2.4***	—	3.8***
	(1.2)	(1.1)	(1.0)	(0.7)		(0.8)
Long-term contribution (points)	0.5	0.1	0.8	0.2	—	0.1
	(0.9)	(0.7)	(0.7)	(0.4)		(0.6)
Firm and year effects	Yes	Yes	Yes			
Firm-by-year effects				Yes	Yes	Yes
Main-group-by-year effects				Yes	Yes	
Founding controls		Yes		Yes	Yes	Yes
Language effects		Yes		Yes	Yes	Yes
CPC level of inside	main group	main group	main group	main group	subgroup	main group
Repositories	102,980	102,980	107,018	96,854	96,854	97,951
Firms	686	686	1,110	626	626	630

Notes: Column 4 is the main estimate, and its key entries are in bold. Each entry in Panels A and B and the first row of Panel C is $100(e^{\beta}-1)$, the percent difference in the expected count between inside and other repositories, where β is the Poisson pseudo-maximum-likelihood coefficient on the inside indicator in equation (1). The maintained and long-term-contribution rows are differences in percentage points from linear probability models with the same effects. Standard errors clustered by firm, on the scale of each entry, are in parentheses. The founding controls are those of section 3.4, and the prior year is the four quarters before arriving. All contributions are every actor's contribution-family events in the window, bots aside, and contributions by others are those less the new participants'. Repository and firm counts are those of the estimate for new participants, after Poisson drops the firms or cells whose outcome is zero on every repository in columns 4 to 6, and before that drop in columns 1 to 3. Column 4's two shares use 98,795 repositories of 667 firms, and column 6's maintained share uses 99,455 repositories of 671 firms. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$, with each p-value adjusted by Holm's method (Holm, 1979) within its column. The three families are the counts of Panel A with contributions by new participants and stars, all contributions and contributions by others with all events (not shown), and the two shares of Panel C with the shares of repositories that have a new participant, one new to the firm and one of those active at another linked firm (not shown).

The point estimate for new participants is larger with coarser technology effects, and it is positive in all three forms, though no test compares them. It is 36.8 percent with no technology effects (column 6), 34.8 percent with effects at the subclass, one level above the main group, by cohort, the year of first activity (Appendix Table 17), and 29.5 percent with main-group-by-year effects (column 4), and each passes Holm's adjustment. For contributions by new participants the same

three are 41.4, 49.6 and 45.5 percent, which are not ordered in the same way.

With firm and year effects alone the differences are larger, and the founding controls attenuate them. In column 1, with a firm effect and a year effect, inside repositories have 77.3 percent more new participants than the firm's other repositories (95 percent interval 33.5 to 135.6). The founding controls bring this to 34.3 percent, and on the scale of β the coefficient falls by 49 percent. On the wider set of firms, the difference is 79.9 percent. The founding controls can absorb part of the inside difference itself (Appendix C.2).

Inside repositories have more contributions overall, not only more from new participants. In column 4, all contributions are 37.6 percent higher, and contributions by others, everyone other than the new participants, are 29.3 percent higher. In one stacked model, built as in section 4.4, the inside coefficient for others less that for new participants is -0.1179 (standard error 0.0959, p-value 0.2192), and in no column are the two groups separated from each other.

New participants make 49.6 percent of all contributions on inside repositories and 49.7 percent on the others, shares of the totals, in which large repositories count for more. Repository by repository, among those with a contribution, the new participants' share of a repository's contributions is 1.02 points higher inside in column 4's form (95 percent interval -0.52 to 2.55, adjusted p-value 0.2496) and 1.66 points higher in column 6's (interval 0.13 to 3.19, adjusted p-value 0.1308), so neither is separated from zero after Holm's adjustment, though column 6's interval excludes zero before it. That share exists only for repositories with a contribution, and inside repositories are 2.53 points more likely to have one in column 4's form, so the two sides differ slightly in which repositories are compared, and I set no bound in advance for how close the two shares must be. Inside repositories have more contributions from both groups, and the estimates do not establish that new participants account for a larger share.

Three qualifications go with column 4. First, a linear regression with column 4's effects can put negative weights on some repositories, and in that regression for new participants they sum to -0.206, against -0.025 with column 6's, a diagnostic of the linear regression and not of the Poisson coefficient (Appendix C.2). Second, if a repository's new participants are the users who leave contribution-family work on it, inside repositories may have more users or a larger share of users who leave such work, and the data cannot separate the two (Appendix C.2). Third, a grant can follow a repository's first activity, and the labeler read the text as it stood in 2026 (Appendices A.7 and B.7).

In column 4, inside repositories are 2.4 points more likely to be maintained in their second year. Stars by new users differ by 7.1 percent and long-term contribution by 0.2 points, and neither is separated from zero after Holm's adjustment. Appendix C.5 discusses these outcomes and the test of the perspective of visible contributions. The positive inside difference itself runs against the mapped prediction of less participation in code tied to patents (section 2.1).

4.3 Contrasts by perspective

The line of work of section 2.1 predicts the sign of the inside difference, and the three perspectives of section 2.2, with a fourth, predict where it is larger. Each reaches the data through a mapping that I state, and Appendix Table 22 sets the mappings out.

The fourth perspective, of related experience, predicts where the difference is larger but not its sign. In Athey and Ellison (2014), programmers who have used open-source code publish their improvements, and the mapped prediction concerns new participants whose earlier activity lies in the same or a related main group. In their conclusion, Athey and Ellison (2014) remark that raising a project's quality can have perverse effects, so this perspective fixes no sign for the inside difference itself.

For less participation in code tied to patents (Lin and Rai, 2024; Wen et al., 2013), β, the coefficient on inside, is predicted negative. For appropriability (Teece, 1986), β and the interaction of inside with the firm's patent weight in the repository's primary main group are both predicted positive. For code tied to products (Gambardella and Hall, 2006; Athey and Ellison, 2014), the interaction of inside with membership in the two classes of a classifier of repositories that hold such code, described below, is predicted positive. For visible contributions (Lerner and Tirole, 2002), the inside difference on issues and on pull requests minus that on stars is predicted positive. For related experience (Athey and Ellison, 2014), the inside difference for new participants with related experience minus that for the others is predicted positive.

Each prediction of where the difference is larger is read from one contrast in column 4's form, and Appendix C.5 sets out the visible-contribution contrasts.

The estimates compare proportional differences, and a larger gain in participation can come with a smaller proportional difference (Wooldridge, 2023; Chen and Roth, 2024), so a prediction of where the difference is larger needs a condition on the proportional difference that none of the perspectives states. This qualification was written after the results were seen.

None of the five contrasts for the last four perspectives above passes its declared test, Holm's adjustment at the five-percent level, so these tests give no statistically clear support for their mapped predictions (Appendix Table 23). Clear support here means that a contrast passes Holm's adjustment at the five-percent level, the level at which I call an estimate significant (section 4.2). As in Galasso and Schankerman (2015), each prediction is read from one contrast, estimated separately in column 4's form.

I report every contrast with its interval for two reasons. First, the mappings, the contrasts and the outcomes adjusted together by Holm's method were specified before the results, and reporting all of them keeps the reading from resting on the subset of contrasts that happen to pass (Anderson, 2008). Second, each contrast tests one perspective's mapped prediction, and its interval shows which sizes of the predicted contrast these data leave open, so a contrast that misses the test still informs its perspective.

The repository-level mapping of the appropriability perspective (Teece, 1986) predicts a larger inside difference where the firm holds more patent weight in the repository's primary main group.

The interaction of inside with the weight is 6.5 percent for new participants (95 percent interval −7.3 to 22.4), and no outcome's interaction is significant. The interaction compares how the count changes with the weight among inside repositories with how it changes among outside ones. The weight is zero for 19,982 of the 20,250 outside repositories with a primary main group, so the outside side of the comparison rests on the 268 with positive weight, and the interval reaches 22.4 percent.

The mapping of the perspective of code tied to products (Athey and Ellison, 2014; Gambardella and Hall, 2006) predicts a larger inside difference in code tied to the firm's products. A rule-based classifier sorts each repository by its own text into six classes, and code tied to products joins two of them, (i) software development kits, interfaces and libraries, and (ii) drivers, integrations and plugins. The interaction of inside with membership in those two classes is −10.1 percent (interval −28.1 to 12.4), not significant, with a point estimate opposite the mapped prediction and an upper limit of 12.4 percent.

The mapping of the perspective of visible contributions (Lerner and Tirole, 2002) predicts a larger inside difference on issues and pull requests than on stars, which gives two contrasts. With inside as in Table 4, the inside difference on the new participants' issues exceeds that on stars by 31.6 percent (interval 1.5 to 70.7), and on their pull requests by 36.3 percent (interval −1.3 to 88.2). The pointwise interval for issues excludes zero and the one for pull requests includes it, and each contrast misses the declared test (Holm's adjustment over issues, pull requests and commit comments), with adjusted p-values of 0.1152 and 0.1193.

The mapping of the perspective of related experience (Athey and Ellison, 2014) predicts a larger inside difference for new participants whose earlier held events include a repository in the same or a related main group. The contrast between their inside difference and that of the other new participants is −7.5 percent (interval −17.0 to 3.1), not significant, again with a point estimate opposite the mapped prediction and an upper limit of 3.1 percent.

Of the five contrasts, only the issue contrast has a pointwise interval that excludes zero. Here the mapped predictions describe patterns in the data and say nothing of what a firm intends.

4.4 Heterogeneity across new participants

The splits by participants' histories and by what they do next divide the inside difference among groups of arrivals, so, taken together, they describe who arrives and what the arrivals then do, not what any group prefers.

By their whole held history before arriving, new participants fall into three groups. People with a history at the firm have a held event of any type on the firm's organizations before the quarter of their first held event on the repository. People with a history only at another linked firm have no such event and at least one on another linked firm's organizations. People with a history at no linked firm have no held event on any linked firm's organizations, the firm's included, though they may still have been active on personal accounts or on organizations outside the linked firms. The history runs from 2015 to the quarter of arrival. Table 1 uses the same history for the firm but only the four quarters before arriving for another linked firm, so its cells differ from these groups. For

each group, the outcome is the repository's count of new participants in that group, estimated in column 4's form. Chen et al. (2022) likewise estimate contributions to a firm's open-source projects separately for its own developers and for outsiders, a split approximated here by held history, with no actor matched to an employer, and Haese and Peukert (2026) set contributors with an earlier history on one project against the others.

Over the full history from 2015, the inside difference is 31.7 percent for people with a history at no linked firm, which is significant, 32.8 percent for people with a history at the firm, and 23.7 percent for people with a history only at another linked firm, both also significant. The history is shorter for repositories first active early, so a check looks back the same 16 quarters before every arrival, on the repositories first active in 2019–2022, as studies measure a history over a window of the same length for everyone (Deryugina and Molitor, 2020) or hold the first years of the data back as history (Cunningham and Kapacinskaite, 2026). There, the inside difference for people with a history at no linked firm is 5.2 percent, and not significant. The inside difference for this group, narrower than the participants new to the firm, is therefore fragile, significant on the whole history and not on the 16 quarters. The two estimates differ both in how far the history looks back and in the repositories they cover, so they do not show which of the two changes accounts for the difference between them. No group's inside difference is distinguishable from another's. The differences between groups come from one model in which the groups' counts are stacked, every effect and control is interacted with the group, and standard errors are clustered by firm, as in Albert and Caggese (2021) and Batistich and Bond (2023). Its group coefficients equal those of the separate models. A history counts held events on the linked firms' organizations only, not all of GitHub.

New participants also differ in what they do in the four quarters after arriving. This split covers the new participants who arrive in the repository's first year, quarters $q_0 + 1$ to $q_0 + 4$, so that every person is followed for four full quarters within the window. The arrival quarter itself is left out, since the order of events within a quarter is not held. Two questions, both about contribution-family events in those four quarters, place each person. The first is whether the person contributed to another repository of the firm's organizations, and the second whether the person contributed on another linked firm's organizations. People thus go on to neither, to another linked firm only, to the firm's other repositories only, or to both, and as with the history, each group's count is estimated in column 4's form.

The inside difference appears in every group. Among people who go on to neither, it is 23.4 percent, which is significant. For people who go on to the firm's other repositories only, it is 30.8 percent, also significant. In column 4, no contrast between two groups is significant, and the wide intervals leave open whether the groups differ. The groups are defined by observed events, and no actor is matched to an employer, so they describe who arrives. Studies of open-source projects likewise class newcomers by what they do later (Zhou and Mockus, 2015) and divide contributions by where they go (Chen et al., 2022).

4.5 Robustness

Inside rests on a share of label weight, a level of the classification and the firm's own patents, each of which could be drawn another way. A repository's label weight is split equally over the main groups the labeler returns for it (section 3.2), and the main estimate counts the repository as inside when at least half of that weight lies in the firm's patented main groups.

Table 5 repeats column 4 of Table 4 with inside drawn in other ways. Panel A moves the cut on the share of label weight, as Kerr (2010) moves the cut that defines a breakthrough patent and Bandiera et al. (2021) the cut on a share that defines their indicator. The inside difference in new participants and in contributions is positive under every definition of inside, and Appendix C.2 reports each in the forms of Table 4's columns 1 and 2. When every label must fall in the firm's patented main groups, the inside difference in new participants is 16.4 percent. This is about half the estimate of Table 4. It passes Holm's adjustment here, narrowly, with an adjusted p-value of 0.0413. In a check that drops GitHub's create and member events from the records, it does not pass, at 15.6 percent on new participants and 18.0 percent on contributions, with adjusted p-values of 0.0663 and 0.0763. In the running example, fastMRI has two of its three labels in Meta's patented main groups, so it falls outside when all of the weight must lie there (Appendix A.9).

When the share of the repository's label weight in those main groups replaces the inside indicator, the inside difference is 43.5 percent for a move of the share from zero to one (95 percent interval 21.0 to 70.1), which is significant. Bandiera et al. (2021) likewise report the estimate on a continuous share beside the indicator built by cutting it, and here the share's estimate is read for its sign.

The cuts of Panel A are not doses, because the share divides by the number of a repository's labels. Two labels in the firm's patented main groups give a share of one on a repository with two labels and of two thirds on a repository with three. Each cut also changes both sides of the comparison, since raising it moves repositories from inside to the other repositories. Raising the cut from two thirds to all of the weight moves 1,552 reference repositories from inside to the other repositories, 1,397 of them in column 4's sample. Each of them has 3 labels, of which all but one lie in the firm's patented main groups. Their raw mean of new participants is 19.67, against 14.94 for the repositories with all of their label weight in those main groups. No other range of the share has a higher raw mean. The fall in Panel A from 30.9 percent at two thirds to 16.4 percent at all of the weight comes from how the cut sorts repositories between the two sides, since nothing else changes between the two estimates.

In a direct test of the step from a share of two thirds to a share of one, no fall is shown, which is not the same as showing none. In column 4's form, repositories with a share of one have 32.5 percent more new participants than those with two thirds within three-label repositories (95 percent interval −15.1 to 106.8), and −1.2 percent across all repositories with label-count effects (interval −26.4 to 32.6). For contributions by new participants the two are 11.6 percent (interval −36.4 to 95.7) and −11.1 percent (interval −40.4 to 32.7). Holm's adjustment across these four estimates, a family of their own, leaves none separated from zero, and the smallest adjusted p-value is 0.8613. The intervals are wide, so they leave open a fall as well as a large rise. Panel A's fall is a different

comparison, since each cut moves repositories between the two sides.

Measured by the number of labels in the firm's patented main groups, no fall is shown, and no steady rise is shown either. In one model in column 4's form with label-count effects, one, two and three such labels go with 29.2, 34.3 and 43.2 percent more new participants than none, and 54.8, 88.8 and 74.6 percent more contributions. Of the four steps from one count to the next, two for each outcome, only the step from one label to two on contributions is separated from zero after Holm's adjustment (adjusted p-value 0.0336), and the point estimate for contributions is lower at three labels than at two.

The inside difference stays positive when inside is drawn without the share. With inside read from the first label alone, or from any label with one row per label, and with main-group-by-year and label-count effects, inside repositories have 22.0 and 26.6 percent more new participants and 49.1 and 45.2 percent more contributions, each with an adjusted p-value below one percent (Appendix C.2 and Table 16). With a control for repository size, measured after the outcome window, those two estimates for new participants are no longer separated from zero after Holm's adjustment, and the two for contributions still are (Appendix Table 16).

Panel B redraws inside one level coarser and one level finer in the classification, as Thompson and Fox-Kean (2005) redraw control patents at a finer class and Bloom et al. (2013) rebuild their measure with finer classes, which they caution are not necessarily better. At the CPC subclass, 92,365 of the 102,987 reference repositories are inside, so the comparison rests on the few left outside. There the inside difference misses Holm's adjustment in the forms of Table 4's columns 1, 2 and 4 (Appendix C.2). The subclass estimate for new participants is not smaller, 43.3 percent, but its standard error is 25.0, against 8.1 at the main group, where the estimate is 29.5 percent, and 7.0 at the subgroup, where it is 27.8 percent. Of the three levels, the standard error is small only at the main group and the subgroup, and drawing inside at those levels needs labels that name main groups (section 3.2).

GitHub's archived flag, which marks a repository its owner has made read-only, is read from 2026 records. It marks 34.9 percent of column 1's inside repositories and 22.6 percent of the others (Appendix A.4). The flag describes 2026 rather than the outcome window, and no estimate drops archived repositories.

By arrival quarter, the column-4 inside difference in new participants is positive in every quarter, with larger point estimates in later quarters (Appendix Table 19). It is 12.6 percent in the first quarter, not significant, and 41.1 percent in the eighth, which is significant. In column 1, the first quarter is significant, at 44.1 percent. The format follows Abebe et al. (2021), who estimate an effect for each period since a start date. No test across quarters was declared, and pointwise intervals support only pointwise readings (Freyaldenhoven et al., 2019), so the larger later estimates are a description. The clock starts at the first observed activity rather than a date on which outsiders could first see the repository, so the profile describes participation by the repository's age.

Forks, repositories copied from another repository, are 5.4 percent of column 1's inside repositories and 10.4 percent of the others, and dropping them leaves every significant estimate significant. Since

a fork can operate apart from its original and is not the same unit (Kalliamvakou et al., 2016), studies of GitHub drop forks from their samples (Abou El-Komboz and Goldbeck, 2025; Medappa et al., 2025).

On the repositories that the second labeler also labels, the column-1 estimate is close under both sets of labels, and the column-4 estimate is imprecise under both (Appendix C.3). The comparison of the two sets of labels is read as agreement between labelers that shows how the estimate moves with the labeler, not as accuracy.

Table 5: The inside difference in column 4's form under other definitions of inside

	Repositories	Firms	Inside repositories	New participants	Contributions
Panel A. The share of label weight in the firm's patented main groups					
Any of the label weight (> 0)	96,854	626	83,001	32.0^{***}	67.2^{***}
				(10.2)	(15.0)
At least half ($\geq 1/2$)	96,854	626	82,046	29.5^{***}	45.5^{***}
				(8.1)	(12.9)
At least two thirds ($\geq 2/3$)	96,854	626	72,041	30.9^{***}	55.5^{***}
				(6.8)	(15.7)
All of the label weight ($= 1$)	96,854	626	70,489	16.4^{**}	19.9^{**}
				(6.7)	(8.4)
Panel B. The level of the CPC, at half the label weight					
Subclass	96,854	626	92,365	43.3	53.9
				(25.0)	(29.5)
Main group	96,854	626	82,046	29.5^{***}	45.5^{***}
				(8.1)	(12.9)
Subgroup	96,854	626	73,184	27.8^{***}	21.6^{*}
				(7.0)	(11.0)

Notes: Each row repeats column 4 of Table 4, with firm-by-year effects, main-group-by-year effects, language effects and the founding controls, and changes only how inside is drawn. Each estimate is $100(e^{\beta} - 1)$, with its standard error clustered by firm in parentheses. The repository and firm counts, the same in every row, are those of each row's estimate for new participants. The row at half the label weight in Panel A and the main-group row of Panel B are column 4 of Table 4, and the subgroup row is its column 5. The inside repositories are counted among the 102,987 reference repositories. Panel B keeps half the label weight and compares labels and patents at another level of the CPC. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$, with each p-value adjusted by Holm's method for the five count outcomes of the row's estimate.

5 Conclusion

The paper makes three contributions, a method, data and one result. The method places open-source code in the patent classification, repository by repository, with an LLM's labels in CPC main groups (section 3.2). The labels were checked against two references built independently of them. They share at least one main group with those of the European Patent Office's text categorizer for 76.4%

of the repositories both classify, against 38.5% when the label sets are dealt out at random among repositories of the same CPC section. Run on patents with another LLM, the same method gives labels that share a main group with the examiners' classification for 89.1% of them, against 20.8% when the model's label sets are dealt out at random among patents of the same CPC section. The method allows a firm's code and its patents to be set side by side in one classification.

The data link Compustat firms to their patents and to their GitHub accounts, each link with its evidence and the year in which the firm owns the account (section 3.1). The Global Corporate Patent Dataset, which never entered the crosswalk, names the same firm for 92.8% of the 2015–2022 patents both link to a Compustat firm, and the commit-email data of Spinellis et al. (2020) name the same firm for 93.9% of the accounts both hold. The data also show that US patent documents cite the OSS-active firms' projects more often than comparable projects of other listed firms, and that citations are rare in both groups (Appendix D.1).

Conditional on firm-by-year and main-group-by-year effects and the founding controls, inside repositories have 29.5 percent more new participants than other repositories (95 percent interval 14.6 to 46.4) and 45.5 percent more contributions from new participants (95 percent interval 22.3 to 73.0), alongside 37.6 percent more contributions from everyone. For new participants with no earlier activity at the firm on GitHub, the inside difference is 27.6 percent. Without the five largest firms, the estimates for new participants and contributions are 28.1 and 40.1 percent. With inside read from the first label alone, or from any label with one row per label, and with main-group-by-year and label-count effects, they are 22.0 and 26.6 percent for new participants and 49.1 and 45.2 percent for contributions. Each of these differences has an adjusted p-value below one percent. The association is partly across firms (section 4.2). A repository's overlap with its firm's own patents differs from the enforcement and the change in patent scope that Lin and Rai (2024) discuss, so this result and theirs stand side by side.

To my knowledge, this paper is the first to compare participation across a firm's open-source projects by whether each lies in the firm's own patented technologies, and much remains to be done with that comparison. Why firms publish code where they patent, and what lies behind the difference between inside and other repositories, are questions for further work. A match of actors to employers would show who employs the people who come, and repository text dated to each repository's first activity would allow labels from that date. The crosswalk now extends to grants of 2011–2014 (section 3.4), so the comparison can reach earlier years. The same classification can take in the firms' products, and a model of the firm's and the contributors' choices can build on these facts. Because the labels place a firm's open-source code and its patents in one classification, the data also allow a comparison of the two portfolios within a firm. A distance between them can be read against a reference built from the firm's own projects and patents, and that comparison is the subject of separate work.

Use of AI tools

I used LLMs for three data-construction tasks and report the checks or bounds for each. To find firms' GitHub accounts, I used LLMs from several providers, run through command-line agents that search the web. I accepted every account the search reported as a verified official page. A recall check bounds the share of firms whose accounts the search missed (Appendix A.4). In the patent crosswalk, LLMs from several providers, run the same way, assisted the identification of the firm for 6.91% of all firm–patent links (Gong, 2026). I required evidence for the exact identifiers in each link (Appendix A.4). I used GLM-5.3-Flash to label the open-source projects. In Appendix B.1, I check the labels against the EPO's text categorizer and, by running DeepSeek-V4-Flash on patents, against the examiners' labels. The AI assistants Claude Code and Codex also helped with coding, editing and drafting. I reviewed all text and results and take responsibility for them.

References

Abebe, G., Caria, A. S., Fafchamps, M., Falco, P., Franklin, S., and Quinn, S. (2021). Anonymity or distance? Job search and labour market exclusion in a growing African city. *Review of Economic Studies*, 88(3):1279–1310.

Abou El-Komboz, L. and Goldbeck, M. (2025). Career concerns as a public good: The role of signaling for open source software development. *Labour Economics*, 97:102800.

Albert, C. and Caggese, A. (2021). Cyclical fluctuations, financial shocks, and the entry of fast-growing entrepreneurial startups. *Review of Financial Studies*, 34(5):2508–2548.

Alexy, O., George, G., and Salter, A. J. (2013). Cui bono? The selective revealing of knowledge and its implications for innovative activity. *Academy of Management Review*, 38(2):270–291.

Anderson, M. L. (2008). Multiple inference and gender differences in the effects of early intervention: A reevaluation of the Abecedarian, Perry Preschool, and Early Training Projects. *Journal of the American Statistical Association*, 103(484):1481–1495.

Arora, A., Belenzon, S., and Sheer, L. (2021). Matching patents to Compustat firms, 1980–2015: Dynamic reassignment, name changes, and ownership structures. *Research Policy*, 50(5):104217.

Athey, S. and Ellison, G. (2014). Dynamics of open source movements. *Journal of Economics & Management Strategy*, 23(2):294–316.

Bandiera, O., Best, M. C., Khan, A. Q., and Prat, A. (2021). The allocation of authority in organizations: A field experiment with bureaucrats. *Quarterly Journal of Economics*, 136(4):2195–2242.

Bar-Gill, O. and Parchomovsky, G. (2003). The value of giving away secrets. *Virginia Law Review*, 89(8):1857–1895.

Barkemeyer, D., Bena, J., and Li, X. (2026). Technology life cycles: Investment, growth, and firm value. Working paper 6350538, SSRN.

Batistich, M. K. and Bond, T. N. (2023). Stalled racial progress and Japanese trade in the 1970s and 1980s. *Review of Economic Studies*, 90(6):2792–2821.

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

Bena, J., Ferreira, M. A., Matos, P., and Pires, P. (2017). Are foreign investors locusts? The long-term effects of foreign institutional ownership. *Journal of Financial Economics*, 126(1):122–146.

Bloom, N., Schankerman, M., and Van Reenen, J. (2013). Identifying technology spillovers and product market rivalry. *Econometrica*, 81(4):1347–1393.

Borges, H. and Valente, M. T. (2018). What's in a GitHub star? Understanding repository starring practices in a social coding platform. *Journal of Systems and Software*, 146:112–129.

Cao, Y., Mkrtchyan, A., Tucker, J. W., and Wan, C. (2026). Corporate disclosure of innovation on GitHub. SSRN Working Paper 5932514, Donald G. Costello College of Business at George Mason University. First draft June 2025; this draft March 2026.

Casalnuovo, C., Vasilescu, B., Devanbu, P., and Filkov, V. (2015). Developer onboarding in GitHub: the role of prior social links and language experience. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*, pages 817–828. ACM.

Chen, J. and Roth, J. (2024). Logs with zeros? some problems and solutions. *Quarterly Journal of Economics*, 139(2):891–936.

Chen, W., Jin, F., and Xue, L. (2022). Flourish or perish? The impact of technological acquisitions on contributions to open-source software. *Information Systems Research*, 33(3):867–886.

Coelho, J., Valente, M. T., Silva, L. L., and Shihab, E. (2018). Identifying unmaintained projects in GitHub. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–10. ACM.

Cohen, W. M., Nelson, R. R., and Walsh, J. P. (2000). Protecting their intellectual assets: Appropriability conditions and why U.S. manufacturing firms patent (or not). Working Paper 7552, National Bureau of Economic Research.

Conti, A., Peukert, C., and Roche, M. P. (2025). Beefing IT up for your investor? Engagement with open source communities, innovation, and startup funding: Evidence from GitHub. *Organization Science*, 36(4):1551–1573.

Crowston, K. and Howison, J. (2005). The social structure of free and open source software development. *First Monday*, 10(2).

Cunningham, C. and Kapacinskaite, A. (2026). Keeping invention confidential. *Management Science*, 72(2):1658–1679.

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

Deryugina, T. and Molitor, D. (2020). Does when you die depend on where you live? Evidence from Hurricane Katrina. *American Economic Review*, 110(11):3602–3633.

Economides, N. and Katsamakas, E. (2006). Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Science*, 52(7):1057–1071.

Emery, L. P., Lim, C., and Ye, S. (2025). The private value of open-source innovation. SSRN Working Paper 5054880, revised November 2025.

Fosfuri, A., Giarratana, M. S., and Luzzi, A. (2008). The penguin has entered the building: The commercialization of open source software products. *Organization Science*, 19(2):292–305.

Freyaldenhoven, S., Hansen, C., and Shapiro, J. M. (2019). Pre-event trends in the panel event-study design. *American Economic Review*, 109(9):3307–3338.

Galasso, A. and Schankerman, M. (2015). Patents and cumulative innovation: Causal evidence from the courts. *Quarterly Journal of Economics*, 130(1):317–369.

Gambardella, A. and Hall, B. H. (2006). Proprietary versus public domain licensing of software and research products. *Research Policy*, 35(6):875–892.

Ganglmair, B. and Kann, A. (2025). Looking for innovation beyond the patent system: Evidence from research disclosures. Slides presented at the DRUID 2025 Conference, June 27, 2025.

Gong, X. (2026). Firms, patents and open source code: Two crosswalks. Working paper.

Gortmaker, J. (2025). Open source software policy in industry equilibrium. Working paper, January 19, 2025; jeffgortmaker.com.

Habibi, M. (2025). Open sourcing GPTs: Economics of open sourcing advanced AI models. arXiv:2501.11581.

Haese, J. and Peukert, C. (2026). Open at the core: Moving from proprietary technology to building a product on open source software. *Management Science*, 72(5):4173–4199.

Harhoff, D., Henkel, J., and von Hippel, E. (2003). Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations. *Research Policy*, 32(10):1753–1769.

Henkel, J. (2006). Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy*, 35(7):953–969.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Horstmann, I., MacDonald, G. M., and Slivinski, A. (1985). Patents as information transfer mechanisms: To patent or (maybe) not to patent. *Journal of Political Economy*, 93(5):837–858.

Kalliamvakou, E., Damian, D., Blincoe, K., Singer, L., and German, D. M. (2015). Open source-style collaborative development practices in commercial projects using GitHub. In *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 574–585.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., and Damian, D. (2016). An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering*, 21(5):2035–2071.

Kerr, W. R. (2010). Breakthrough inventions and migrating clusters of innovation. *Journal of Urban Economics*, 67(1):46–60.

Kogan, L., Papanikolaou, D., Seru, A., and Stoffman, N. (2017). Technological innovation, resource allocation, and growth. *Quarterly Journal of Economics*, 132(2):665–712.

Lakhani, K. R. and von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy*, 32(6):923–943.

Lee, J.-S. and Hsiang, J. (2020). Patent classification by fine-tuning BERT language model. *World Patent Information*, 61:101965.

Lerner, J. and Tirole, J. (2002). Some simple economics of open source. *Journal of Industrial Economics*, 50(2):197–234.

Lerner, J. and Tirole, J. (2005). The economics of technology sharing: Open source and beyond. *Journal of Economic Perspectives*, 19(2):99–120.

Lin, Y.-K. and Rai, A. (2024). The scope of software patent protection in the digital age: Evidence from Alice. *Information Systems Research*, 35(2):657–672.

Medappa, P. K., Srivastava, S. C., and Favaron, S. D. (2025). Write access provisioning and organizational ownership in open source software projects: Exploring the impact on project novelty and survival. *Research Policy*, 54(8):105284.

Nagaraj, A. and Piezunka, H. (2024). The divergent effect of competition on platforms: Deterring recruits, motivating converts. *Strategy Science*, 9(3):277–296.

Nagle, F. (2018). Learning by contributing: Gaining competitive advantage through contribution to crowdsourced public goods. *Organization Science*, 29(4):569–587.

Parker, G. and Van Alstyne, M. (2018). Innovation, openness, and platform control. *Management Science*, 64(7):3015–3032.

Petralia, S. (2025). Open source software as digital platforms to innovate. *Journal of Economic Behavior and Organization*, 237:107109.

Petralia, S. and Boschma, R. (2026). Actionable disclosure in patents: How open code reduces spatial barriers to knowledge diffusion. Papers in Evolutionary Economic Geography 26.11, Utrecht University. SSRN Working Paper 7090442, version of 9 July 2026.

Pinto, G., Steinmacher, I., Dias, L. F., and Gerosa, M. (2018). On the challenges of open-sourcing proprietary software projects. *Empirical Software Engineering*, 23(6):3221–3247.

Raymond, E. S. (1998). The cathedral and the bazaar. *First Monday*, 3(3).

Raymond, E. S. (2001). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates, Sebastopol, CA, revised edition. The essays are also at http://www.catb.org/~esr/writings/cathedral-bazaar/.

Santos Silva, J. M. C. and Tenreyro, S. (2006). The log of gravity. *Review of Economics and Statistics*, 88(4):641–658.

Spinellis, D., Kotti, Z., Kravvaritis, K., Theodorou, G., and Louridas, P. (2020). A dataset of enterprise-driven open source software. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 533–537.

Steinmacher, I., Conte, T. U., Gerosa, M. A., and Redmiles, D. F. (2015). Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '15)*, pages 1379–1392.

Teece, D. J. (1986). Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy*, 15(6):285–305.

Thompson, P. and Fox-Kean, M. (2005). Patent citations and the geography of knowledge spillovers: A reassessment. *American Economic Review*, 95(1):450–460.

Valiev, M., Vasilescu, B., and Herbsleb, J. (2018). Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem. In *Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '18)*, pages 644–655.

von Hippel, E. (2005). *Democratizing Innovation*. The MIT Press, Cambridge, MA.

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

von Krogh, G., Spaeth, S., and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, 32(7):1217–1241.

Wen, W., Ceccagnoli, M., and Forman, C. (2016). Opening up intellectual property strategy: Implications for open source software entry by start-up firms. *Management Science*, 62(9):2668–2691.

Wen, W., Forman, C., and Graham, S. J. H. (2013). The impact of intellectual property rights enforcement on open source software project success. *Information Systems Research*, 24(4):1131–1146.

Won, P. Y. (2026). Commoditize your complement: Strategic open-source releases and the cannibalization of rival firms. Conference draft, May 2026. https://www.paris-december.eu/sites/default/files//papers/2026/7456_sutari4697_2026_complete.pdf.

Wooldridge, J. M. (1999). Distribution-free estimation of some nonlinear panel data models. *Journal of Econometrics*, 90(1):77–97.

Wooldridge, J. M. (2023). Simple approaches to nonlinear difference-in-differences with panel data. *Econometrics Journal*, 26(3):C31–C66.

Wright, N. L., Nagle, F., and Greenstein, S. (2023). Open source software and global entrepreneurship. *Research Policy*, 52(9):104846.

Yamashita, K., Kamei, Y., McIntosh, S., Hassan, A. E., and Ubayashi, N. (2016). Magnet or sticky? measuring project characteristics from the perspective of developer attraction and retention. *Journal of Information Processing*, 24(2):339–348.

Zhang, Y., Zhou, M., Mockus, A., and Jin, Z. (2021). Companies' participation in OSS development—an empirical study of OpenStack. *IEEE Transactions on Software Engineering*, 47(10):2242–2259.

Zhou, M. and Mockus, A. (2015). Who will stay in the FLOSS community? Modeling participant's initial behavior. *IEEE Transactions on Software Engineering*, 41(1):82–99.

Zhou, S., Vasilescu, B., and Kästner, C. (2020). How has forking changed in the last 20 years? A study of hard forks on GitHub. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE 2020)*, pages 445–456. ACM.