The Faster the Better? Innovation Speed and User Interest in Open Source Software

Dong, John Qi; Wu, Weifang; Zhang, Yixin Sarah

Published in:
Information & Management

DOI:
10.1016/j.im.2018.11.002

Publication date:
2019

Information & Management 56 (2019) 669–680

Contents lists available at ScienceDirect

Information & Management

journal homepage: www.elsevier.com/locate/im

The faster the better? Innovation speed and user interest in open source software

John Qi Dong[a,*], Weifang Wu[b], Yixin (Sarah) Zhang[c]

[a] Faculty of Economics and Business, University of Groningen, 9747 AE Groningen, The Netherlands
[b] Department of Digitalization, Copenhagen Business School, DK-2000 Frederiksberg, Denmark
[c] Swedish Center for Digital Innovation, Department of Applied Information Technology, University of Gothenburg, 41756 Gothenburg, Sweden

ARTICLE INFO

Keywords:
Innovation speed
Signals
Software development
Open source software
Digital innovation
Open innovation
Crowdsourcing

ABSTRACT

It is often believed that for open source software (OSS) projects the faster the release, the better for attracting user interest in the software. Whether this is true, however, is still open to question. There is considerable information asymmetry between OSS projects and potential users as project quality is unobservable to users. We suggest that innovation speed of OSS project can signal the unobservable project quality and attract users' interest in downloading and using the software. We contextualize innovation speed of OSS projects as initial release speed and update speed and examine their impacts on user interest. Drawing on the signaling theory, we propose a signaling effect through which a higher initial release speed or update speed increases user interest, while the effect diminishes as initial release or update speed increases. Using a large-scale panel data set from 7442 OSS projects on SourceForge between 2007 and 2010, our results corroborate the inverted U-shaped relationships between initial release speed and user downloads and between update speed and user downloads.

1. Introduction

Open source software (OSS) has gained great popularity in recent years [1,64]. OSS projects are hosted on the online platforms and rely on voluntary participation of crowd labor [2]. The online workforce contributes to the code base of OSS, making it essentially a result of crowdsourcing of the software innovation projects (e.g., [3,4]). Well-known OSS projects include Apache, Linux, MySQL, R, Perl, and Open Office, which have attracted numerous users [5]. For example, the accumulative downloads of Open Office grew over 63 times within three and half years[1]. What factors may influence user interest in an OSS and make it popular is, therefore, an important question [6–10].

For potential users, the adoption decision of an OSS involves the information asymmetry regarding project quality, in terms of developer quality and code quality [11]. To deal with such information asymmetry, users need to rely on observable cues. Prior OSS research has noted that certain project factors may serve as signals to reduce information asymmetry. For example, Stewart et al. [8] suggested that organizational sponsorship to OSS projects could serve as a cue when evaluating an OSS. Sen et al. [12] proposed that OSS license choice could indicate the utility of OSS to potential users. Similarly, Wen et al. [10] suggested that intellectual property rights (IPR) enforcement provides a proximate signal about the risks to developers. Recently, Ho and Rai [11] found that quality controls of firm-participating OSS projects can signal project quality to developers and increase their continuous participation. In this study, we broaden this stream of research by examining the impacts of timing factors, which has been understudied in the literature, on user interest in an OSS. We argue that innovation speed can signal unobservable project quality to potential users and influence their decisions to download and use an OSS.

The innovation literature suggests that innovation speed substantially impacts the success of new product development projects [65,66]. Relevant to innovation speed, release management has been suggested as one of the most important project management issues in OSS development [13,14]. How does innovation speed of OSS projects influence user interest in OSS projects? This is an important question because in the OSS context, it is often reckoned that the faster the better. For example, a general guideline for OSS projects on SourceForge suggests: 'Release early, release often' is the key motto in open source development[2]. We challenge this wisdom by unveiling the

* Corresponding author at: Department of Innovation Management and Strategy, Faculty of Economics and Business, University of Groningen, 9747 AE Groningen, The Netherlands.
E-mail address: john.dong@rug.nl (J.Q. Dong).

[1] Available on http://www.openoffice.org/stats/downloads.html.
[2] Available on http://sourceforge.net/apps/trac/sourceforge/wiki/Project%20control.

https://doi.org/10.1016/j.im.2018.11.002
Received 17 March 2017; Received in revised form 27 October 2018; Accepted 7 November 2018
Available online 11 November 2018
0378-7206/

complex signaling effects of innovation speed of OSS projects on user interest by taking the information symmetry between OSS projects and potential users into consideration.

Specifically, we contextualize the concept of innovation speed and conceptualize initial release speed and update speed in the OSS context, where *initial release speed* refers to how fast the first prototype is released in an OSS project, and *update speed* refers to how often updates and patches are released in an OSS project. Drawing on the signaling theory [67,68], we argue that there is notable information asymmetry between OSS projects and users, while fast initial release and frequent updates of an OSS have an overlooked side effect to signal unobservable project quality to potential users. On the one hand, a high initial release speed or update speed indicates strong capabilities of developers and high quality of software, encouraging potential users to download and use an OSS. On the other hand, however, a very high speed elicits users' concerns regarding unobservable developer and code quality, reducing their interest in the OSS. Using a large-scale panel data set from 7442 OSS projects on SourceForge between 2007 and 2010, we find empirical evidence corroborating the inverted U-shaped relationships between initial release speed and user downloads, and between update speed and user downloads.

This study has several contributions to the OSS literature. Drawing on the signaling perspective, we recognize the information asymmetry between OSS projects and potential users and theorize how innovation speed may signal unobservable project quality to potential users. Furthermore, we theorize and examine innovation speed in the OSS context, a critical timing aspect of OSS projects that has not been systematically studied. Specifically, we conceptualize two dimensions of innovation speed for OSS projects (i.e., initial release speed and update speed) and postulate the inverted U-shaped relationships between initial release speed and user downloads, and between update speed and user downloads. Our empirical evidence enriches the OSS literature by providing robust and generalizable findings corroborating the hypothesized inverted U-shaped relationships.

This paper is organized as follows. Section 2 provides a review of the relevant OSS literature. In Section 3, we first conceptualize innovation speed in the OSS context and characterize the context to illustrate why the signaling theory can be used. We then develop theoretical arguments for the signaling theory articulating the impacts of innovation speed on user interest in OSS projects. Empirical methodology is described in Section 4, and results are reported in Section 5. In Section 6, we conclude the paper by discussing the theoretical and practical implications, as well as the limitations and directions for future research.

2. Literature Review

Diverse topics have been investigated in the OSS literature [5,15], such as developers' motivations to contribute to OSS projects [2,16], developers' interaction and diversity [17], license choice [8,12,18], code reuse [19], project team learning [20,21], firm sponsorship [16], firms create and capture value from OSS [1,22], and competition between open source and proprietary software [23]. Since OSS innovation process and user interest are explicitly relevant to our study, the literature review focuses on these two areas.

2.1. OSS innovation process

OSS projects usually require long term contribution from a group of members [24]. Moon and Sproull [25] conducted a case study of Linux and described how the Linux project started from Linus Torvalds's improvements of Minix and evolved into a large-scale collaboration. Gallivan [26] performed a secondary analysis of published cases of OSS studies and highlighted the role of trust among the different participants of OSS projects (core developers, peripheral developers, and posters to mailing list). In addition to case studies, there are also studies based on archival data. Lee and Cole [27] analyzed archival data of Linux project and noted that developers naturally sort themselves into a two-tier structure, a small group of core developers and a large group of periphery developers. The core developers include a project leader and hundreds of maintainers, and the periphery developers are organized into the development team and the bug reporting team. Their roles merge in the process of performing tasks. Using archival data from SourceForge, Daniel et al. [6] recognized the diversity of OSS teams, and examined the effects of team diversity, in terms of separation, variety, and disparity, on responses to forum posts, artifact (e.g., bug report and feature request) closure, and total number of downloads. Au et al. [20] examined team learning effects, and found as the number of bugs resolved increased, the average bug resolution time decreased. Team size and developers experiences also influence bug resolution time. Recognizing the complexity in OSS teams, Singh and Tan [17] built non-cooperative game-theoretic model to study the network formation in an OSS team and found there may exist several stable structures that are inefficient and there may not always exist a stable structure that is efficient. To examine developers' learning, Singh et al. (2011) built hidden Markov model to analyze archival data, and found that there were different learning patterns, developers of low or high statues may learn from their own experiences or from peers. Though these studies provide insights about OSS innovation process, understanding about the dynamics in OSS projects from a timing perspective is relatively limited.

2.2. User interest in OSS

In the OSS literature, a significant portion of studies are about developer's motivation to contribute to OSS projects (e.g., [2,11,16,28–30]). Fewer studies took the users' perspective to examine user interest in downloading and using OSS. Among such studies, Setia et al. [7] noted the role of peripheral developers (who are not formal members of the core development team and contribute much less code to the projects) and showed that the existence of peripheral developers has a positive effect on the popularization of OSS (measured by the number of downloads). Stewart et al. [8] examined the impacts of license restrictiveness and organizational sponsorship on user interest and noted projects that are sponsored by nonmarket organizations and that employ nonrestrictive licenses attracted more subscribers. Wen et al. [10] investigated the effects of intellectual property right (IPR) enforcement on open source software project success, and their analysis showed that the lawsuits of SCO v. IBM and FireStar/DataTern v. Red Hat lead to decline in user downloads. Daniel et al. [6] founded that developers' diversities influence developer engagement and project market success (operationalized by the number of downloads). Sutanto et al. [9] discussed the relationship between properties of the user support network and OSS popularity (in terms of the number of downloads or the number of active users — users who are active in contributing to the forum). These studies all consider user interest as an important indicator of OSS project success, however, as we noted, the studies either did not explicitly discuss how users assess the quality of the OSS project before downloading the software (e.g., [6]) or only alluded to the challenges users face when evaluating the quality of the OSS project [8,10]. As we will discuss in the theory and hypotheses section, users are facing difficulties when assessing project quality due to information asymmetry, and innovation speed of OSS projects is likely to be a signal influencing user interest in OSS.

3. Theory and hypotheses

3.1. Innovation speed in the OSS context

Innovation speed has been widely studied in the research on new product development (e.g., [65,69,70,71]). Overall, three perspectives have been used to conceptualize innovation speed. Kessler and

Table 1
Different Perspectives of Conceptualization Innovation Speed.

Perspective	Conceptualization	Reference
Initial release perspective	The length of time elapsed from the beginning of applied research (if there were any) by the innovator on a new product or process to the date of the new product's or process's first commercial introduction.	Mansfield [75]
	The time elapsed between start of the development process and market introduction.	Clark and Fujimoto [74]
	Total project time from the beginning of idea generation to the end of market launch.	Ali et al. [73]
Schedule tracking perspective	The degree to which a project met an assigned schedule.	Keller [76]
	The degree to which a project was ahead, on or behind schedule.	McDonough [77]
Upgrading perspective	The timing of introducing the upgrades of a new release.	Padmanabhan et al. [72]
	The degree to which marginal products are regularly weeded out.	Tambe et al. [71]

Chakrabarti [65] reviewed innovation speed research and categorized two different perspectives: the initial release perspective and the schedule tracking perspective (Table 2, p. 1151). Some other studies took another perspective — the upgrading perspective (e.g., [71,72]), as some products may have long lifetime and need to be periodically upgraded. Table 1 summarizes the definitions and key references for each perspective on innovation speed.

As for the initial release perspective, scholars have been focusing on innovation speed based on how long it takes for an idea to be transformed into a marketable entity (e.g., [65,73–75]). In this perspective, innovation speed is defined as "the time elapsed between initial development, including the conception and definition of an innovation and ultimate commercialization, which is the introduction of a new product into market place" ([65], p. 1144). From this perspective, we define *initial release speed* in the OSS context as how soon an OSS project releases its first version of software since project launch.

From the schedule tracking perspective, some scholars often compare the actual progress of a project with its planned schedule (e.g., [76,77]). This perspective is, however, not suitable for OSS projects, because typical schedules usually do not exist in OSS projects. The schedule of OSS development is not written anywhere, and not made visible [37]. Thus, we do not theorize innovation speed in the OSS context from the schedule tracking perspective.

As for the upgrading perspective, innovation speed has been defined as the timing strategy of introducing the upgrades of a product [72], or the degree to which marginal products are weeded out [71]. Because OSS, or software in general, needs to be updated continuously during its lifecycle [79], the upgrading perspective is particularly relevant to OSS projects. An important attribute of software innovation is that it is continuously updated during the lifetimes [78]. After an initial release of an OSS, users usually identify and report vulnerabilities or malfunctions. Patches are created and released by developers to fix these problems [79,80]. Meanwhile, users may require new features, so upgrades are developed to address their requests. Modification requests, bug fixes, and new enhancements are continuously released in OSS projects [38,81]. Therefore, initial release is only a start of the innovation journey; after releasing the first version, OSS developers need to persistently work on the project to update the software. Complementary to initial release, updates reflect the innovation speed during OSS project lifecycle. From the updating perspective, we define *update speed* of OSS projects as how frequent an OSS project releases new version or new patch of software.

Beyond the OSS context, time relevant factors have been considered as important measures of performance in the broader software development literature [39]. Software development research has discussed delays in software projects, and the losses due to the delays were frequently observed in practice [40,41]. Some studies examined the antecedents of software development speed, such as how geographic distance and cross-site communication among development team members may influence development speed [42]. Others investigated software development process management, in which time is considered as a measure of project input or constraint, or a measure of outcome or success (e.g., [40,43]). For example, Boehm [44] noted the emphasis on time-to-market and how this emphasis influenced changes in software development processes, as "a major shift away from the sequential waterfall model to models emphasizing concurrent engineering…" (p. 18). To the best of our knowledge, however, none of these studies have paid attention to how timely release and updating frequency may influence user interest in software and instead, timely release itself was considered as part of software project quality. In this study, we specifically examine the role of innovation speed of OSS project as a signal in influencing user interest in OSS.

3.2. Innovation speed of OSS projects as a signal

Downloading, installing, and learning software involve time and effort and are even subject to risks due to malfunctions. Such costs make users less likely to try every OSS, but rather urge them to select one software that has good quality and reliable support. Different from proprietary software products, OSS software products usually involve more uncertainties that influence product quality and survival [7]. Proprietary software products are introduced to the market after considerable testing, and companies have dedicated developer teams to provide support and maintenance [45]. More importantly, potential users of proprietary software have access to a vast amount of information released from advertising campaigns by software companies, in conjunction with the reputation of software companies, considerably helping them to assess the quality of proprietary software. [7]. Software companies heavily invest in various commercial and marketing initiatives, in order to reduce potential users' information asymmetry about their software products and promote user interest. Taking Microsoft — one of the largest development software companies — as an example, its annual expenditure on marketing and sales have raised up to more than 17 billion U.S. dollars in 2018[3]. Similarly, Electronic Arts (EA) — one of the largest games software companies — spent 641 million U.S. dollars on marketing and sales in 2018[4].

Prior studies have either explicitly or implicitly noted that it is challenging for users to assess OSS project quality [8,11,82]. For example, Stewart et al. [8] noted that "organizational sponsorship may imply the availability of technical support, upgrades, and other resources that may be needed over the long term by consumers of software products" (p. 131). Sen et al. [12] proposed that OSS license choice "can signal the overall utility of the software to potential users" (p. 209). Wen et al. [10] suggested that "the implications of IPR enforcement through litigation actions rather than the existence of formal IPR like patents or copyrights because the former provides a more proximate signal about the risks to users and developers" (p. 1132). Ho and Rai [11] noted developers may take quality control procedures (i.e., accreditation, code reuse) as signals, and influence their decisions to continue participate in OSS projects, and proposed that "this information is relatively limited, and therefore developers rely on signals

[3] Available on https://www.statista.com/statistics/506534/microsoft-sales-marketing-expenditure.

[4] Available on https://www.statista.com/statistics/672141/electronic-arts-marketing-and-sales-spending.

associated with quality controls for a more comprehensive assessment of project quality to inform their continued participation intentions" (p. 604).

Although OSS projects open the code of their software, it is very labor-intensive and skill-demanding to inspect the code to assess project quality [11]. For example, OSS projects with less than 100,000 lines of code had a defect density of 0.4 and OSS projects with more than 100,000 lines of code had a defect density of 0.6 [46]. To find out a defect in the code, an OSS developer needs to engage in an inspection of on average 1700–2500 lines of code. While some OSS developers are users, the majority users never participate in OSS development [47]. It is difficult for them without sophisticated knowledge and skills to assess project quality by checking the open code. Software development is a knowledge intensive process and project quality depends on all aspects of software development [48]. Even if a user is a software developer, he or she may not be capable of assessing all aspects. Depending on their roles, developers only read and write a small portion of code in the OSS project with a modular architecture. Thus, the available code may not be thoroughly assessed and automated inspection tools are limited in their ability to inspect defects [49]. Different from proprietary software customers who are informed by companies through various commercial and marketing initiatives, OSS users have limited knowledge about the quality of OSS projects. Potential users may rely on observable signals to distinguish different projects and make their download decisions. Although the signaling theory often assumes that signalers deliberately send signals to receivers, it is possible that signalers signal receivers without intent [50]. However, it does not affect how receivers respond to signals [51].

The signaling theory is relatively new to OSS research, in which a couple of prior studies briefly mentioned some OSS project characteristics can serve as signals (e.g., [8,10,12]). To the best of our knowledge, Ho and Rai [11] is the only study systematically using the signaling theory to investigate continued participation intentions of developers in firm-participating OSS projects. Their recent work provides useful insight into how OSS project features — OSS project leaders' quality controls (accreditation and code acceptance), though in a different context in which firms are involved, produce a side effect by signaling unobservable input and output quality of an OSS project — developer quality and code quality [11]. Developer quality is critical to OSS projects to produce successful and popular software, as 80% of OSS projects failed during development [2]. Code quality determines the usefulness of an OSS, and therefore users' decision to download and use the software [52]. Although there is a paucity of OSS studies drawing on the signaling theory, it has been widely used across different contexts (see Table 2 for a summary).

Our study focuses on a different research context consisting of OSS projects without firm participation, in which accreditation and code acceptance are not formally established. For such projects based on crowd only, we identify initial release speed and update speed of OSS projects that can signal unobservable developer quality and code quality. Our central argument is that signaling associated with innovation speed is helpful to users in assessing developer quality and code quality of an OSS project when making their decisions to download and use a software, while OSS project leaders may not be aware that their decisions regarding innovation speed act as signals to inform unobservable project quality to users. Next, we detail the rationale about how initial release speed and update speed impact user interest in OSS through signaling developer quality and code quality.

3.3. Innovation speed of OSS projects and user interest

Information asymmetry exists between OSS projects and potential users. This is especially difficult for the initial release of OSS, because they cannot draw inferences about developer quality and code quality from other users' downloads or feedback in this stage. We propose that initial release speed can serve as a signal of developer quality and code quality when users consider whether to download and use the first version of an OSS. Achieving a high initial release speed is costly, making it a trustworthy signal to users [55]. Software development requires considerable programming efforts and skills. Quick initial release can be interpreted as a positive signal indicating strong capabilities of developers in an OSS project in the short run. Additionally, more capable developers are likely to deliver high quality code in a short cycle time. In other new product development contexts, fast release is often associated with project success [65,66,74]. Thus, a higher initial release speed of an OSS project increases users' confidence of software quality and encourages their downloads.

On the other hand, however, OSS development is usually a time-consuming and iterative process, as prototype is tested, modified, and tested again [38,53]. Developers' participation determines the sustainability of an OSS project [16,30]. When the initial release is too fast, an OSS is often developed by a small group of core developers in a closed environment without outside audience or participant [54]. It indicates that the software has benefited very little from developing, testing, feature adding, bug reporting and fixing by a vast of crowd in a distributed environment, lowering the quality of code. Thus, very fast release also elicits users' concern about the capability of developers as it suggests the degree to which the project can attract more developers and form a strong project team is highly uncertain. Also, very fast release may also indicate potential flaws in an OSS. Thus, users may interpret a very high initial release speed as a negative signal of developer quality and code quality. Overall, users are likely to interpret initial release speed within the middle range in the most positive way. Hence, we suggest that the positive signaling effect of initial release speed on user downloads will diminish when initial release speed increases, leading to an inverted U-shaped relationship between initial release speed and user downloads.

H1. Initial release speed of an OSS project has an inverted U-shaped relationship with its user downloads.

Compared to the initial release, the information asymmetry is less serious in the later stage of OSS projects, as many users have downloaded and used the initial version of software. However, it is still difficult for the users, especially for those who haven't used the software, to predict whether developers are capable to continuously offer reliable support in an OSS project. While users' downloads and feedback also provide information about unobserved project quality, update speed provides unique, additional information compared with initial release speed about the capabilities and dedications of developers to an OSS project in the long term. Again, achieving a high update speed is costly, making it a trustworthy signal to users [55]. Users can take update speed as a signal, because frequent updates imply the developers' capabilities in an OSS project to respond quickly to users' problems and requests and the quality of software. More capable developers continue to respond promptly to the reports of bugs and requests for new features, leading to better code quality. In other new product development contexts, it has been suggested that responding quickly to requests for changes and keeping the clients informed of project progress are critical for success [83]. A high update speed of an OSS project can signal that developers have sufficient capabilities to address user needs and increase users' confidence of code quality, thereby encouraging their downloads.

On the other hand, however, too fast updates may lead potential users to doubt the quality of developers, making them worried about the developers' capabilities to develop functional software. OSS development requires significant long-term investment in maintenance as a result of its long-term, interactive nature [56]. Frequent patches may also suggest vulnerabilities in the earlier versions of software, inviting potential users' concerns about code quality of the software. Overall, users are likely to interpret an update speed within the middle range in the most positive way. Hence, we caution that the positive signaling effect of update speed on user downloads will diminish when update

Table 2
A Summary of Prior Studies Using the Signaling Theory.

Reference	Signals	Context	Findings
Boulding and Kirmani [31]	Product warranties (e.g., warry length, and scope)	Consumers	The study examines consumers' perceptions of warranties through the lens of signaling theory. Warranty's quality in terms of length and scope can signal product quality. Experiment results show that consumers perceive a product with a better warranty is of higher quality than a product with a poorer warranty, when firms face penalty for not-fulfilling the warranty.
Albinger and Freeman [32]	Firm's corporate social performance (CSP)	Job seekers	Firms' CSP may serve as signals of working conditions, organizational values and norms to job candidates for job seeking population. For job seekers with more choices, there exists a strong positive relationship between organizations' CSP and organization their attractiveness.
Wang et al. [33]	Online retailers' website cues	Consumers	Small online retailers may use website cues as signals. Experiment results show that security disclosures and awards from neutral sources increase consumers' trust, and seals of approval and privacy disclosures increases consumer's likelihood to disclose information.
Cohen and Dean [34]	Top management team (TMT) legitimacy	Investors	In the context of initial public offering (IPO), TMT legitimacy can serve as signal to investors. Investors perceive legitimate TMT involvement as a signal of the economic potential of the firm, and TMT legitimacy reduces the likelihood of IPO underpricing.
Mavlanova et al. [35]	E-commerce website contents and features (e.g., contact information, privacy policy, their party seals, live chat, and consumer feedback)	Consumers	E-commerce pharmacies can signal their qualities through website contents and features. The signals are classified according to cost of signaling and difficult of verification. Analysis of archival data show that low-quality pharmacies use fewer signals than high-quality pharmacies in the pre-contractual phase. Low-quality pharmacies seem to avoid easily verifiable signals, and high-quality pharmacies do not refrain from displaying easily verifiable signals. Low-quality pharmacies also avoid high-cost signals.
Ahlers et al. [36]	Entrepreneurs retain equity and providing detailed information about risks.	Investors	In the context of equity crowdfunding, entrepreneurs can signal the quality of a project by retaining equity and providing more detailed information about risks. When entrepreneurs retain equity and provide more information about risks, likelihood of funding success increases.
Ho and Rai [11]	OSS accreditation and code acceptance	OSS developers	OSS accreditation can be perceived as a signal of input quality and code acceptance can be perceived as a signal of output quality of OSS projects for OSS developers. Results suggest accreditation and code use positively influence developers' continued participation intention in OSS projects.

speed increases. Summarizing the above rationale, we hypothesize an inverted U-shaped relationship between update speed and user downloads.

H2. Update speed of an OSS project has an inverted U-shaped relationship with its user downloads.

4. Methodology

4.1. Data

We collected a large-scale panel data set from SourceForge (sourceforge.net). SourceForge was selected because it is one of the largest online OSS communities[5]. It provides a standard technology toolset for OSS projects, including code repositories, download statistics, defect reporting, and other project management tools. SourceForge organizes projects under different categories. We selected all the projects from the largest category — "development" projects. To construct a project-month panel, we gathered the data over a 36-month period from the launch date of each project. As we collected data in January 2013, to ensure enough time spam we sampled all development projects that were launched between 2007 and 2010.

Among the registered products on SourceForge, there are some projects which provide no software for users to download. As including such projects with no release and thus no downloads can bias our results, we only focused on the projects with at least one release of software. Among a total of 20,212 development projects launched on SourceForge between 2007 and 2010, there are 7521 projects with releases. Moreover, there are projects with downloads before the "first" release date, these downloads indicated that old version files might have been removed by the project team. To avoid any contamination of our data, such projects were dropped from our sample. Our final sample consists of 7442 projects, launched between 2007 and 2010.

4.2. Measures

4.2.1. User downloads

Scholars have proposed various indicators of OSS project success, such as the number of downloads, the number of individuals who have subscribed to the project's mailing list, release of new features, traffic on the website, writes to codes, etc. [8,17,84,85]. Basically, these indicators fall into two categories — developer activity and user interest [8][6]. As we are interested in how users make download decisions under conditions of information asymmetry, we focus on user interest. Among the indicators of user interest, user downloads are widely used in prior OSS studies [8,85,86]. Though it cannot perfectly capture software use, it can be considered a reasonable proxy[7]. Therefore, monthly numbers of downloads were collected for each project to measure user downloads. To reduce the skewness, we take the natural logarithm of the

[5] At the time of data collection in January 2013, SourceForge provided free hosting for more than 324,000 projects and more than 3.4 million members. It attracted more than 46 million users and more than 4,000,000 downloads per day.

[6] Grewal et al. [86] similarly framed these two categories as technical success and commercial success of OSS projects. Daniel et al. [6] framed them as community engagement and market success.

[7] "The number of downloads is a market-based measure of popularity, which should relate to product use, particularly when software is distributed through a single channel as in the case of SourceForge.net (e.g., [84]). When a software product is freely available, researchers have used downloads as a surrogate for 'sales' (e.g., Chandrashekaran et al. 1999)" ([86], p. 1047).

number of downloads. To avoid reverse causality, we lag all variables to user downloads for one period, namely one month[8].

4.2.2. Innovation speed

Two aspects of innovation speed were measured according to their definitions. Initial release speed is defined as how soon an OSS project releases its first version of software since project launch. We measure initial release speed by the reciprocal of the number of days from project launch on SourceForge[9] to the initial release. Update speed is defined as how frequent an OSS project releases new version or new patch of software. We use the number of updates in each month after the initial release as the measure of update speed. Because significant updates matter to the users, four criteria were enforced to identify the updates of a project: 1) not only updates in the root directory but also updates in the second level and the third level directory of the file folders are identified; 2) multiple updates on the same day are counted only once; 3) small changes (e.g., a change of file of which the type is text, pdf, jpg, etc.) are excluded; and 4) the updates are downloadable files.

4.3. Control variables

While our study focuses on examining the impacts of initial release speed and update speed on user downloads, other attributes may also contribute to user interest in an OSS project. Hence, we control for a set of factors that might affect user downloads based on prior OSS literature. We use monthly data for time-variant control variables, otherwise we use time-invariant data for each project. We take the natural logarithm of continuous variables to reduce the skewness.

4.3.1. User downloads in previous period

Technology adoption decisions are considerably influenced by prior adopters [87,88]. We dynamically control user downloads in previous period that provide potential users an important cue of project quality. Dynamic control for the dependent variable over time also helps to address endogeneity as the effects of unobservable omitted variables on user downloads are controlled.

4.3.2. Developer writes

Prior studies have suggested that developers' efforts are related to the success of OSS projects [57,86,89,90]. We control the number of writes made by developers into codes or files on SourceForge as a proxy of their efforts to a project.

4.3.3. Length of project description

Clarity of goals and targets is related to project success [65]. We follow Hahn et al.'s [82] measure of project information availability and control the number of words in project description as a proxy of goal clarity.

4.3.4. Number of categories

Audience type can influence user interest [8,21]. A project is possible for different types of use, and thus might be under different categories. The number of categories indicates how many different categories a project belongs to at the same time. Because the projects in different categories usually have different target users, we use the number of categories as a proxy for the variety of user type.

4.3.5. Number of leaders

Project leaders are usually the key developers of OSS projects [91], which are listed on the project webpage on SourceForge. We control the number of project leaders who are key developers in a project.

4.3.6. Multiple participations of leaders

Multiple participations of project leaders may indicate their experience and effort devoted to a project, which is controlled by the average number of projects that they have participated in.

4.3.7. Initial release size

File size indicates the richness of features and functions in the software, therefore it may affect user downloads. We control the size of first version, as measured by kilobytes of the software.

4.3.8. Code reuse

There is a "fork" phenomenon in OSS projects, in which developers reuse codes from other projects. We follow the approach of Nyman and Mikkonen [92] and compile a list of keywords, including "clone", "fork", and "repackage", based on the terms identified in prior OSS studies (e.g., [19,93–95]). We also included additional keywords, including "reuse", "library", "libraries", "integrate", "integration", "redefine", and "overlap". We search these keywords in the description of each project and identify the projects with code reuse. If so, we code it as 1 and 0 otherwise.

4.3.9. License dummies

Researchers have suggested that license choice impacts OSS diffusion, because license restrictiveness may affect the benefits and costs from using the software [8,10,58,96]. We follow prior literature to code and control for three types of licenses as highly restrictive, restrictive, or unrestrictive [59,60,85]. First, strong copyleft indicates whether a project has the GNU General Public License (GPL), which requires the subsequent derivative programs based on the original must also be licensed similarly. Second, weak copyleft indicates whether a project has the GNU Lesser General Public License (LGPL), which requires that the subsequent programs based on the original must also be licensed similarly, but the modified software can be released under a different license under certain conditions. Finally, no copyleft indicates whether a project has the Berkeley Software Distribution (BSD), in which developers are not obligated to inherit the original license when they redistribute any derivative work.

4.3.10. Time dummies

Quarterly time dummies based on project registration date were included. These dummy variables can control the temporal differences.

Table 3 provides the description for each variable. Descriptive statistics of all variables are reported in Table 4, and correlations of these variables are reported in Table 5. There are statistically significant and positive correlations between user downloads and the two innovation speed variables. Besides, user downloads in continuous periods are highly correlated.

4.4. Analysis strategy

Our research design leads to a nested structure as each OSS project has multiple observations at different points of time. There are two levels of random variations — the user downloads within OSS projects over time and the user downloads between projects. We therefore applied a mixed effects hierarchical linear model to allow different intercepts for each OSS project. This model can control for level-specific

[8] In a robustness check, we also tried different time lags and found that our results were consistent.

[9] It is likely that some OSS projects may have developed the first version of software before project launch on SourceForge. However, we argue that this should not be the case for the majority as most projects launch on SorceForge for obtaining useful input from other voluntary developers to complete the software [2]. Even if this is the case for most of OSS projects, we should not be able to observe any significant effect of initial release speed on user interest. Thus, our measure of initial release speed could only be more conservative if more projects completed the first version of software before project launch on SourceForge. If we are still able to observe any significant effect of initial release speed on user interest, this concern actually makes our evidence stronger.

Table 3
Description of Variables.

Variable	Description
User downloads	The logged number of downloads in the corresponding time period
Recommendation ratio	The number of positive ratings divided by the total number of ratings
Initial release speed	The reciprocal of the number of days from project launch to the initial release
Update speed	The number of updates per month
Developer writes	The logged number of times developers write the project code/file in the corresponding time period
Length of description	The logged number of words in project description
Number of categories	The logged number of categories the project belonged to (e.g., browsers, chat, and email)
Number of leaders	The logged number of leaders of the project
Multiple participations of leaders	The logged average number of projects that project leaders participate in
Initial release size	The logged kilobytes of the first version released by the project
Code reuse	Dummy variable indicates whether the focal project reuse codes from other projects
Strong copyleft	Dummy variable indicates whether the project has a very restrictive license GLP
Weak copyleft	Dummy variable indicates whether the project has a less restrictive license LGLP
No copyleft	Dummy variable indicates whether the project has no restrictive license BSD

unobserved heterogeneity by including random or fixed effects terms [61]. In addition, it allows for unbalanced panel data and does not require observation independence [61]. The parameters were estimated by mixed effects maximum likelihood regression analysis. For hypotheses testing, we assessed *z* statistics and calculated *p*-values for the regression coefficients. We compared the different models by using goodness-of-fit statistics, including log likelihood, Akaike information criterion (AIC) and Bayesian information criterion (BIC). We also computed pairwise χ^2 statistics to examine significance of the improving model fit.

5. Results

5.1. Hypotheses testing

To test our hypotheses, we stepwise amended a baseline model with control variables only and random intercepts included. We then added initial release speed and its squared term to the control model. Next, we added initial release speed and its squared term to the control model. Finally, we estimated a full model with both initial release speed and update speed, as well as their quadratic terms. To control for reverse causality, we lagged all variables to user downloads for one month.

The regression results are reported in Table 6. Adding initial release speed and its squared term significantly improved the model fit, comparing to that of the control model. Adding update speed and its squared term also significantly improved the model fit, comparing to that of the control model. The full model shows the best model fit, comparing to that of the control model. The improvement of model fit can also be seen from the decrease of AIC or BIC.

Initial release speed had a statistically significant and positive effect on user downloads while the squared term of initial release speed had a statistically significant and negative effect on user downloads. These results indicate an inverted U-shaped relationship between initial release speed and user downloads. Thus, H1 was supported. Update speed had a statistically significant and positive effect on user downloads while its squared term is statistically significant and negative, suggesting an inverted U-shaped relationship between update speed and user downloads. Thus, H2 was also supported.

Besides, developer writes, length of description, number of categories, number of leaders, initial release size positively affected user downloads. Consistent with previous findings in the literature [8,85], restrictive licenses (i.e., strong and weak copyleft) negatively impacted user downloads, as it might limit developer activity and thus lower user interest.

To visualize the inverted U-shape of these relationships, we plotted the relationships in the feasible ranges of initial release speed and update speed. Fig. 1 shows the inverted U-shaped relationship between initial release speed and user downloads, while Fig. 2 shows the inverted U-shaped relationship between update speed and user downloads.

Table 4
Descriptive Statistics.

	Mean	SD	Min	Max
User downloads	2.394	1.704	0	14.540
Initial release speed	0.307	0.338	0.001	1
Update speed	0.113	0.510	0	27
User downloads in previous period	2.417	1.693	0	13.679
Developer writes	0.269	1.087	0	11.725
Length of description	3.226	0.563	0	5.170
Number of categories	1.023	0.332	0.693	1.946
Number of leaders	0.773	0.225	0.693	2.833
Multiple participations of leaders	1.274	0.716	0	4.615
Initial release size	5.658	2.432	0.004	14.451
Code reuse	0.007	0.082	0	1
Strong copyleft	0.380	0.485	0	1
Weak copyleft	0.130	0.336	0	1
No copyleft	0.090	0.286	0	1

5.2. Robustness checks

We conducted several robustness checks. First, we examined the sensitivity of our results to different time lag specifications[10]. In hypotheses testing, we lagged all variables to user downloads for one month. In this robustness check, we tried 3-month, 6-month, 9-month, and 12-month time lags. Table 7 reports the results. We found that the curvilinear effects of initial release speed and update speed on user downloads still existed, even after 12 months.

Second, we examined the generalizability of our findings based on another category on SourceForge — “games” projects, which is also a big category but quite different from development projects. While development projects focus on providing functionality and utility, games projects may focus more on entertainment and fun and therefore may attract a lot of tryouts from individual users[11]. We used the same sampling procedure and ran the analysis again with 88,977 project-month observations from 2634 games projects in 36 months since the launch date of each project. The results in Table 8 show basically consistent results supporting H1 and H2.

Third, since a number of project-month observations in our panel data have no downloads, we used a Tobit model with left-censoring to test H1 and H2 again. In the Tobit model, 33,776 observations were left censored in the analysis. The results reported in Table 8 are qualitatively similar to the main results, providing support for H1 and H2.

[10] We thank the anonymous associate editor and reviewer for suggesting this point.

[11] We thank one anonymous reviewer who suggested this point.

Table 5
Correlations.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
(1) User downloads													
(2) Initial release speed	**−0.071**												
(3) Update speed	**0.206**	**−0.017**											
(4) User downloads in previous period	**0.890**	**−0.071**	**0.216**										
(5) Developer writes	**0.193**	**−0.097**	**0.254**	**0.195**									
(6) Length of description	**0.137**	**−0.018**	**0.028**	**0.136**	**0.027**								
(7) Number of categories	**0.142**	**−0.071**	**0.033**	**0.141**	**0.066**	**0.153**							
(8) Number of leaders	**0.164**	**−0.137**	**0.048**	**0.162**	**0.146**	**0.028**	**0.095**						
(9) Multiple participations of leaders	**−0.060**	**0.027**	**0.019**	**−0.064**	**0.004**	**−0.031**	**−0.007**	**−0.033**					
(10) Initial release size	**0.192**	**−0.156**	**0.042**	**0.192**	**0.101**	**0.079**	**0.125**	**0.178**	**−0.025**				
(11) Code reuse	**0.028**	−0.000	0.002	**0.028**	**0.005**	**0.030**	**0.008**	**0.019**	−0.002	**0.014**			
(12) Strong copyleft	**−0.036**	**-0.032**	**−0.036**	**−0.033**	**−0.085**	**−0.012**	0.004	**−0.059**	**−0.10**	**−0.021**	0.003		
(13) Weak copyleft	**−0.008**	**−0.040**	**−0.014**	**−0.008**	**−0.012**	**0.006**	**0.015**	**−0.016**	**−0.053**	**0.010**	**0.010**	**−0.191**	
(14) No copyleft	**0.007**	**-0.029**	**0.012**	**0.007**	**0.025**	**−0.005**	**−0.007**	**0.041**	**0.008**	**−0.007**	**0.009**	**−0.206**	**−0.075**

Note: Correlations in bold are significant at $p < 0.05$.

Table 6
Mixed Effects Hierarchical Linear Model Results.

	(1)	(2)	(3)	(4)
Intercept	0.089	0.046	0.093	0.050
	(-0.066)	(-0.068)	(-0.067)	(-0.068)
User downloads in previous period	0.522***	0.521***	0.515***	0.515***
	(-0.002)	(-0.002)	(-0.002)	(-0.002)
Developer writes	0.031***	0.031***	0.020***	0.020***
	(-0.002)	(-0.002)	(-0.002)	(-0.002)
Length of description	0.148***	0.146***	0.149***	0.147***
	(-0.014)	(-0.014)	(-0.014)	(-0.014)
Number of categories	0.233***	0.235***	0.234***	0.236***
	(-0.025)	(-0.025)	(-0.025)	(-0.025)
Number of leaders	0.378***	0.388***	0.382***	0.392***
	(-0.036)	(-0.036)	(-0.036)	(-0.036)
Multiple participations of leaders	−0.057***	−0.057***	−0.059***	−0.059***
	(-0.011)	(-0.011)	(-0.011)	(-0.011)
Initial release size	0.048***	0.050***	0.049***	0.050***
	(-0.003)	(-0.003)	(-0.003)	(-0.003)
Code reuse	0.198*	0.195*	0.200*	0.197*
	(-0.098)	(-0.098)	(-0.099)	(-0.099)
Strong copyleft	−0.091***	−0.094***	−0.091***	−0.094***
	(-0.018)	(-0.018)	(-0.018)	(-0.018)
Weak copyleft	−0.078**	−0.077**	−0.078**	−0.077**
	(-0.025)	(-0.025)	(-0.025)	(-0.025)
No copyleft	−0.027	−0.027	−0.028	−0.028
	(-0.029)	(-0.029)	(-0.029)	(-0.029)
Time dummies	Yes	Yes	Yes	Yes
Initial release speed		0.327***		0.323***
		(-0.091)		(-0.091)
Initial release speed squared		−0.316***		−0.311***
		(-0.089)		(-0.089)
Update speed			0.099***	0.099***
			(-0.004)	(-0.004)
Update speed squared			−0.005***	−0.005***
			(0.0005)	(0.0005)
Intercept variance	0.462***	0.461***	0.467***	0.467***
	(-0.009)	(-0.009)	(-0.009)	(-0.009)
Residual variance	0.475***	0.475***	0.474***	0.474***
	(-0.001)	(-0.001)	(-0.001)	(-0.001)
Log likelihood	−276603	−276596	−276290	−276283
χ^2		13.200**	626.650***	639.300***
AIC	553264	553255	552641	552632
BIC	553566	553578	552965	552977

Note: n of projects = 7442; n of obs. = 251,807. * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$. Standard errors are in parentheses. Dependent variable is user downloads with one-month time lag.

Finally, we used recommendation ratio as an alternative measure for user interest. User ratings have been widely used to evaluate user satisfaction about products or services (e.g., [97,98]). We calculated recommendation ratio as the number of positive ratings that an OSS project received from users on SourceForge divided by the total number of ratings. Because we can only observe user ratings and recommendation ratio at the time of data collection in January 2013, we conducted a cross-sectional analysis by regressing recommendation ratio on initial release speed, update speed, and their squared terms. User downloads in previous period is omitted due to cross-sectional design. As monthly data for a 36-month period is not required here, we include all the 21,105 development projects with releases in the regression. The results reported in Table 8 again support both H1 and H2. Thus, we conclude that our results are robust.

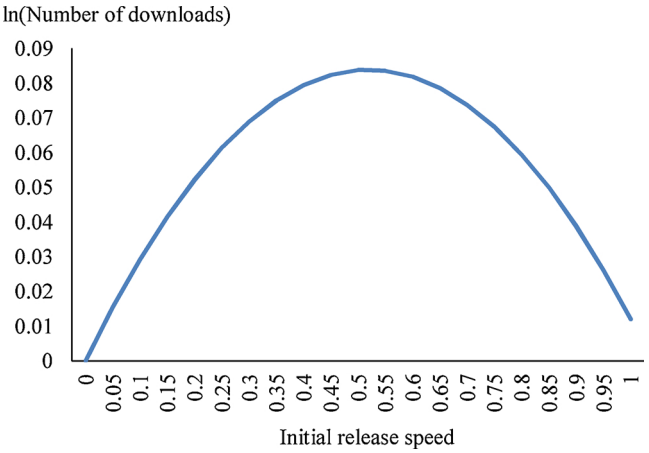

Fig. 1. The Relationship between Initial Release Speed and User Downloads.

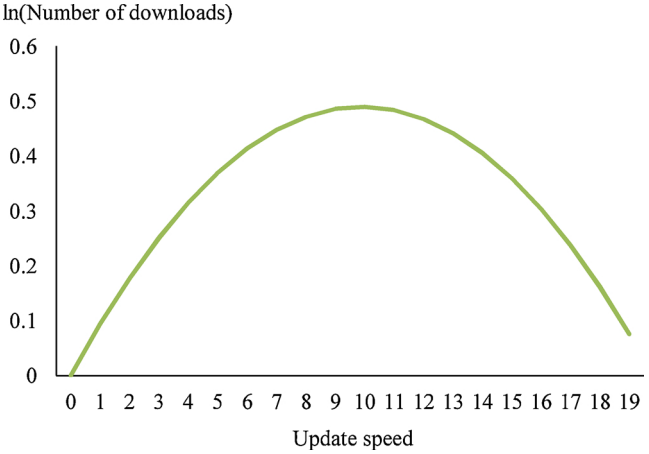

Fig. 2. The Relationship between Update Speed and User Downloads.

Table 7
Sensitivity Analysis of Different Time Lags.

	(1) 3-month	(2) 6-month	(3) 9-month	(4) 12-month
Intercept	0.032	0.062 (0.113)	0.088	0.051
	(-0.097)		(-0.134)	(-0.143)
User downloads in previous period	0.329***	0.221***	0.083***	0.036***
	(-0.002)	(-0.002)	(-0.002)	(-0.002)
Developer writes	0.021***	0.024***	0.027***	0.026***
	(-0.002)	(-0.002)	(-0.002)	(-0.002)
Length of description	0.203***	0.232***	0.268***	0.280***
	(-0.020)	(-0.024)	(-0.028)	(-0.030)
Number of categories	0.336***	0.390***	0.458***	0.487***
	(-0.035)	(-0.041)	(-0.049)	(-0.052)
Number of leaders	0.573***	0.664***	0.791***	0.866***
	(-0.051)	(-0.060)	(-0.071)	(-0.076)
Multiple participations of leaders	−0.075***	−0.085***	−0.098***	−0.093***
	(-0.016)	(-0.019)	(-0.022)	(-0.024)
Initial release size	0.070***	0.081***	0.095***	0.010***
	(-0.005)	(-0.006)	(-0.007)	(-0.007)
Code reuse	0.273	0.316	0.374	0.385
	(-0.14)	(-0.163)	(-0.192)	(-0.203)
Strong copyleft	−0.144***	−0.173***	−0.207***	−0.221***
	(-0.026)	(-0.030)	(-0.036)	(-0.038)
Weak copyleft	−0.115**	−0.135**	−0.163***	−0.178***
	(-0.036)	(-0.042)	(-0.049)	(-0.052)
No copyleft	−0.045	−0.058	−0.066	−0.075
	(-0.041)	(-0.048)	(-0.057)	(-0.060)
Time dummies	Yes	Yes	Yes	Yes
Initial release speed	0.427***	0.475**	0.566**	0.595**
	(-0.129)	(-0.151)	(-0.178)	(-0.189)
Initial release speed squared	−0.420***	−0.469**	−0.567**	−0.601**
	(-0.126)	(-0.147)	(-0.174)	(-0.185)
Update speed	0.037***	0.044***	0.077***	0.084***
	(-0.004)	(-0.005)	(-0.005)	(-0.005)
Update speed squared	−0.002***	−0.002***	−0.004***	−0.004***
	(-0.001)	(-0.001)	(-0.001)	(-0.001)
Intercept variance	0.947**	1.290***	1.798***	2.019***
	(-0.017)	(-0.023)	(-0.031)	(-0.035)
Residual variance	0.536***	0.548***	0.549***	0.532***
	(-0.002)	(-0.002)	(-0.002)	(-0.002)
Log likelihood	−277321	−257239	−233377	−205890
AIC	554707	514543	466820	411846
BIC	555050	514883	467155	412177
n of projects	7422	7404	7378	7340
n of obs.	236,929	216,083	193,899	171,802

Notes: * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$. Standard errors are in parentheses. Dependent variable is user downloads with different time lags.

Table 8
Robustness Checks for Different Projects, Alternative Models and Measures.

	(1) Games projects	(2) Tobit model	(3) Recommendation ratio
Intercept	−0.048	−0.080	−0.012
	(-0.106)	(-0.071)	(0.064)
User downloads in previous period	0.574***	0.561***	0.019***
	(-0.003)	(-0.002)	(0.001)
Developer writes	0.029***	0.021***	0.026***
	(-0.003)	(-0.002)	(0.005)
Length of description	0.114***	0.161***	0.078***
	(-0.020)	(-0.015)	(0.009)
Number of categories	0.186***	0.247***	0.144***
	(-0.045)	(-0.026)	(0.013)
Number of leaders	0.370***	0.402***	−0.029***
	(-0.061)	(-0.037)	(0.004)
Multiple participations of leaders	0.068**	−0.156***	0.012***
	(-0.022)	(-0.012)	(0.001)
Initial release size	0.052***	0.049***	0.098*
	(-0.005)	(-0.004)	(0.040)
Code reuse	0.078	0.212*	−0.062***
	(-0.062)	(-0.102)	(0.007)
Strong copyleft	−0.108***	−0.072***	−0.043***
	(-0.028)	(-0.019)	(0.009)
Weak copyleft	−0.205***	−0.061*	−0.017
	(-0.058)	(-0.026)	(0.011)
No copyleft	−0.120*	−0.030	0.019***
	(-0.058)	(-0.030)	(0.001)
Time dummies	Yes	Yes	Yes
Initial release speed	0.179	0.315***	0.096**
	(-0.134)	(-0.095)	(0.033)
Initial release speed squared	−0.229 $^{+}$	−0.302**	−0.091**
	(-0.128)	(-0.093)	(0.034)
Update speed	0.153***	0.101***	0.468***
	(-0.007)	(-0.005)	(0.041)
Update speed squared	−0.007***	−0.00530***	−0.064***
	(-0.001)	(-0.001)	(0.014)
Intercept variance	0.362***	0.501***	
	(-0.012)	(-0.010)	
Residual variance	0.479***	0.595***	
	(-0.002)	(-0.002)	
Log likelihood	−97813	−295741	
AIC	195691	591548	
BIC	196001	591893	
Adj. R^2			0.108
n of projects	2634	7442	
n of obs.	88,977	251,807	21,105

Notes: $^{+}$ $p < 0.1$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$. Standard errors are in parentheses. Dependent variable is user downloads with one-month time lag or recommendation ratio at the time of data collection.

6. Discussion and conclusion

6.1. Theoretical implications

This study has three major theoretical implications for OSS research. First, the signaling perspective emphasizes the importance of conveying information about project quality to potential users, which is relatively neglected in the OSS literature. Previous studies mainly discussed factors that may influence developers' motivation to participate in OSS projects (e.g., [2,16,62,63]) or factors that may influence user interest in downloading and using OSS (e.g., [7,8,10]), but little attention was paid to how the information about unobservable projects quality in terms of developer quality and code quality can be communicated to potential users. While Ho and Rai [11] recently drew on the signaling theory to study how accreditation and code acceptance in firm-participating OSS projects can signal project quality to *developers* and motivate their continued participation intentions, our study differs from their work by explaining how innovation speed of OSS projects signals project quality to *users* and attract their interest in OSS. We note that users have limited knowledge about OSS projects, leading to a high degree of information asymmetry between OSS projects and users. In this context, OSS projects, and their project leaders in particular, may signal developer quality and code quality to potential users through innovation speed in release management.

Second, we introduce innovation speed, a timing concept from the innovation literature, to OSS research. To the best of our knowledge, ours is the first study taking a timing perspective to theorize the impacts of OSS project characteristics. Prior OSS studies seldom discussed timing factors and mostly focused on other factors that influence user interest, such as developers' motivations (e.g., developer values, and prior experience) and project characteristics (e.g., license choice, and project category [6,8]). We identify and theorize two constructs for innovation speed in the OSS context — initial release speed and update speed. *Initial release speed* refers to how soon an OSS project releases its first version of software since project launch, and *update speed* refers to how frequent an OSS project releases new version or new patch of software. Our results indicate that the timing of initial release and subsequent updates can significantly impact user interest in OSS.

Third, we theorize and empirically examine the effects of initial release speed and update speed on user downloads in the OSS context. Interestingly, both initial release speed and update speed could be beneficial and detrimental depending on their levels. On the one hand, fast initial release can indicate high developer quality and code quality.

On the other hand, as initial release speed is too high, users may doubt the possibility of accumulating sufficient developers in project team and the quality of software that is produced in a rush. Similarly, frequent updates may be interpreted as a favorable signal indicating developers' capabilities to provide high-quality software and continuous support to users on the one hand, and may also be interpreted as a warning signal suggesting the quality of software is not satisfying on the other. We theorize the contradictory signaling effects of initial release speed and update speed on user downloads and empirically demonstrate the evidence that corroborates our theoretical arguments. Overall, our results suggest that innovation speed has both usefulness and downsides to signal developer quality and code quality of OSS projects.

6.2. Practical implications

Our findings shed light on the importance of release management for OSS projects. Both initial release speed and update speed are under the control of OSS project teams and project leaders can strategically manage the timing of first and subsequent releases of software. Our results indicate that initial endeavor to deliver first version of software and continuous effort to update the software are both important to attract user interest. For the marginal effect in our sample of OSS projects, an increase of initial release speed for one standard deviation can lead to a 45% increase of downloads relative to the average number of downloads of our sampled OSS projects. At the same time, one extra update per month also leads to a 46% increase of downloads relative to the average number of downloads of our sampled OSS projects. These findings offer actionable guidance for OSS project leaders to strategically plan for the timing of releases.

In fact, the OSS community has long recognized that innovation speed is critical, and it however holds a general guidance of "the faster the better". For example, SourceForge claims: "The Open Source community lives by 'Release Early, Release Often'"[12]. Our findings challenge this wisdom and show that it only captures part of the story. A very high speed of release may sometimes do more harm than good and have a detrimental effect to user interest in OSS. When the initial release is introduced in the project launch day or one day after, or more than 10 updates are released per month, the number of user downloads could decline in our sampled OSS projects. Our study reminds OSS project leaders to be very cautious in release management, as we do observe that some projects released the first version too soon on the project launch date and the maximum number of updates per month is as high as 27 in some sampled projects.

6.3. Limitations and future research

Our study has some limitations. First, we collect archival data from OSS projects, which are objective but provide limited information about what developers and users perceive and behave. Future study may use other methodologies, such as survey or experiment, to test how users' perceived innovation speed of OSS projects may influence their interest in OSS. Second, although we use time lag to avoid reverse causality and dynamically control the dependent variable and other control variables in the analysis, we cannot fully test the causality. Future research may look for instrumental variables or conduct controlled experiments to better examine the causal relationship between innovation speed of OSS projects and user interest. Last but not least, although we use a large-scale sample in this study, we only include development projects, and games projects in a robustness check, on SourceForge. Future study may collect data from other types of projects or from other OSS communities (e.g., CodePlex, GitHub, and Google Code) to examine the generalizability of our findings.

[12] Available on http://sourceforge.net/apps/trac/sourceforge/wiki/Get%20started%20with%20your%20new%20project.

Acknowledgements

We thank Sirkka Jarvenpaa, James Jiang, Gary Klein, Arun Rai, Kar Yan Tam, Sean Xu, the editor-in-chief Patrick Chau, the associate editor, and three anonymous reviewers for their comments and guidance. The authors are listed alphabetically and contribute equally. Any opinions, findings, and suggestions expressed in this paper are those of the authors, which do not necessarily reflect the views of SourceForge.

References

[1] M. Germonprez, J.E. Kendall, K.E. Kendall, L. Mathiassen, B. Young, B. Warner, A theory of responsive design: a field study of corporate engagement with open source communities, Inf. Syst. Res. 28 (1) (2017) 64–83.
[2] Y. Fang, D. Neufeld, Understanding sustained participation in open source software projects, J. Manag. Inf. Syst. 25 (4) (2009) 9–50.
[3] J.Q. Dong, W. Wu, Business value of social media technologies: evidence from online user innovation communities, J. Strateg. Inf. Syst. 24 (2) (2015) 113–127.
[4] T. Ogink, J.Q. Dong, Stimulating innovation by user feedback on social media: the case of an online user innovation community, Technol. Forecast. Soc. Change (2018) In Press.
[5] K. Crowston, K. Wei, J. Howison, A. Wiggins, Free/Libre open-source software development, ACM Comput. Surv. 44 (2) (2012) 1–35.
[6] S. Daniel, R. Agarwal, K.J. Stewart, The effects of diversity in global, distributed collectives: a study of open source project success, Inf. Syst. Res. 24 (2) (2013) 312–333.
[7] P. Setia, B. Rajagopalan, V. Sambamurthy, R. Calantone, How peripheral developers contribute to open-source software development, Inf. Syst. Res. 23 (1) (2012) 144–163.
[8] K.J. Stewart, A.P. Ammeter, L.M. Maruping, Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects, Inf. Syst. Res. 17 (2) (2006) 126–144.
[9] J. Sutanto, A. Kankanhalli, B.C.Y. Tan, Uncovering the relationship between OSS user support networks and OSS popularity, Decis. Support Syst. 64 (1) (2014) 142–151.
[10] W. Wen, C. Forman, S.J.H. Graham, The impact of intellectual property rights enforcement on open source software project success, Inf. Syst. Res. 24 (4) (2013) 1131–1146.
[11] S.Y. Ho, A. Rai, Continued voluntary participation intention in firm-participating open source software projects, Inf. Syst. Res. 28 (3) (2017) 603–625.
[12] R. Sen, C. Subramaniam, M.L. Nelson, Determinants of the choice of open source software license, J. Manag. Inf. Syst. 25 (3) (2008) 207–240.
[13] K. Crowston, J. Howison, H. Annabi, Information systems success in free and open source software development: theory and measures, Softw. Process. Improv. Pract. 11 (2) (2006) 123–148.
[14] J. Hahn, C. Zhang, An exploratory study of open source projects from a project management perspective, Working Paper, Purdue University, West Lafayette, IN, 2005.
[15] A. Aksulu, M. Wade, A comprehensive review and synthesis of open source research, J. Assoc. Inf. Syst. 11 (11) (2010) 576–656.
[16] S. Spaeth, G. von Krogh, F. He, Perceived firm attributes and intrinsic motivation in sponsored open source software projects, Inf. Syst. Res. 26 (1) (2015) 224–237.
[17] P.V. Singh, Y. Tan, Developer heterogeneity and formation of communication networks in open source software projects, J. Manag. Inf. Syst. 27 (3) (2010) 179–210.
[18] R. Sen, C. Subramaniam, M.L. Nelson, Open source software licenses: Strong-copyleft, non-copyleft, or somewhere in between? Decis. Support Syst. 52 (1) (2011) 199–206.
[19] S. Haefliger, G. von Krogh, S. Spaeth, Code reuse in open source software, Manage. Sci. 54 (1) (2008) 180–193.
[20] Y.A. Au, D. Carpenter, X. Chen, J.G. Clark, Virtual organizational learning in open source software development projects, Inf. Manag. 46 (1) (2009) 9–15.
[21] P.V. Singh, Y. Tan, N. Youn, A hidden markov model of developer learning dynamics in open source software projects, Inf. Syst. Res. 22 (4) (2010) 790–807.
[22] L. Morgan, J. Feller, P. Finnegan, Exploring value networks: theorising the creation and capture of value with open source software, Eur. J. Inf. Syst. 22 (5) (2013) 569–588.
[23] R. Sen, A strategic analysis of competition between open source and proprietary software, J. Manag. Inf. Syst. 24 (1) (2007) 233–257.
[24] C. Herstatt, D. Ehls, Open Source Innovation: The Phenomenon, Participant's Behaviour, Business Implications, Routledge, New York, NY, 2018.
[25] J.Y. Moon, L.S. Sproull, Essence of distributed work: the case of the Linux kernel, in: P.J. Hinds, S. Kiesler (Eds.), Distributed Work, MIT Press, Cambridge, MA, 2002, pp. 381–404.
[26] M.J. Gallivan, Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies, Inf. Syst. J. 11 (4) (2001) 277–304.
[27] G.K. Lee, R.E. Cole, From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development, Organ. Sci. 14 (6) (2003) 633–649.
[28] Y. Li, C.-H. Tan, H.-H. Teo, Leadership characteristics and developers' motivation in open source software development, Inf. Manag. 49 (5) (2012) 257–267.

[29] C.-G. Wu, J.H. Gerlach, C.E. Young, An empirical analysis of open source software developers' motivations and continuance intentions, Inf. Manag. 44 (3) (2007) 253–262.
[30] B. Xu, D.R. Jones, B. Shao, Volunteers' involvement in online community based software development, Inf. Manag. 46 (3) (2009) 151–158.
[31] W. Boulding, A. Kirmani, A consumer-side experimental examination of signaling theory: Do consumers perceive warranties as signals of quality? J. Consum. Res. 20 (1) (1993) 111–123.
[32] H.S. Albinger, S.J. Freeman, Corporate social performance and attractiveness as an employer to different job seeking populations, J. Bus. Ethics 28 (3) (2000) 243–253.
[33] S. Wang, S.E. Beatty, W. Foxx, Signaling the trustworthiness of small online retailers, J. Interact. Mark. 18 (1) (2004) 53–69.
[34] B.D. Cohen, T.J. Dean, Information asymmetry and investor valuation of IPOs: top management team legitimacy as a capital market signal, Strateg. Manage. J. 26 (7) (2005) 683–690.
[35] T. Mavlanova, R. Benbunan-Fich, M. Koufaris, Signaling theory and information asymmetry in online commerce, Inf. Manag. 49 (5) (2012) 240–247.
[36] G.K.C. Ahlers, D. Cumming, C. Günther, D. Schweizer, Signaling in equity crowdfunding, Entrep. Theory Pract. 39 (4) (2015) 955–980.
[37] P. Vixie, Software engineering, in: M. Stone, S. Ockman, C. Dibona (Eds.), Open Sources: Voices from the Open Source Revolution, O'Reilly & Associates, Sebastopol, CA, 1999, pp. 80–88.
[38] M.-W. Wu, Y.-D. Lin, Open source software development: an overview, Computer 34 (6) (2001) 33–38.
[39] J.D. Blackburn, G.D. Scudder, L.N. Van Wassenhove, Improving speed and productivity of software development: a global survey of software developers, IEEE Trans. Softw. Eng. 22 (12) (1996) 875–885.
[40] J.J. Jiang, G. Klein, H.G. Hwang, J. Huang, S.Y. Hung, An exploration of the relationship between software development process maturity and project performance, Inf. Manag. 41 (3) (2004) 279–288.
[41] K.R. Linberg, Software developer perceptions about software project failure: a case study, J. Syst. Softw. 49 (2-3) (1999) 177–192.
[42] J.D. Herbsleb, A. Mockus, An empirical study of speed and communication in globally distributed software development, IEEE Trans. Softw. Eng. 29 (6) (2003) 481–494.
[43] J.H. Iversen, L. Mathiassen, P.A. Nielsen, Managing risk in software process improvement: an action research approach, Mis Q. 28 (3) (2004) 395–433.
[44] B. Boehm, A view of 20th and 21st century software engineering, Proceedings of International Conference on Software Engineering (2006) 12–29.
[45] A. Boulanger, Open-source versus proprietary software: Is one more reliable and secure than the other? IBM Syst. J. 44 (2) (2005) 239–248.
[46] Synopsys. 2015. Open Source Report 2014. Available on http://go.coverity.com/rs/157-LQW-289/images/2014-Coverity-Scan-Report.pdf.
[47] M. Saini, K. Kaur, A review of open source software development life cycle models, Int. J. Softw. Eng. Appl. 8 (3) (2014) 417–434.
[48] M.A. Weiss, Data Structures and Algorithm Analysis in Java, Addison-Wesley Longman Publishing., Boston, MA, 1998.
[49] J. Zheng, N. Nagappan, J.P. Hudepohl, M.A. Vouk, On the value of static analysis for fault detection in software, IEEE Trans. Softw. Eng. 32 (4) (2006) 1–14.
[50] M. Spence, Signaling in retrospect and the informational structure of markets, Am. Econ. Rev. 92 (3) (2002) 434–459.
[51] B.L. Connelly, S.T. Certo, R.D. Ireland, C.R. Reutzel, Signaling theory: a review and assessment, J. Manage. 37 (1) (2011) 39–67.
[52] I. Stamelos, L. Angelis, A. Oikonomou, G.L. Bleris, Code quality analysis in open source software development, Inf. Syst. J. 12 (1) (2002) 43–60.
[53] A. Mockus, R. Fielding, J.D. Herbsleb, Two case studies of open source software development: apache and Mozilla, Acm Trans. Softw. Eng. Methodol. 11 (3) (2000) 309–346.
[54] A. Capiluppi, M. Michlmayr, From the cathedral to the bazaar: an empirical study of the lifecycle of volunteer community projects, in: J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti (Eds.), Open Source Development, Adoption and Innovation, Springer, Boston, MA, 2007, pp. 31–44.
[55] R.B. Bird, E.A. Smith, Signaling theory, strategic interaction, and symbolic capital, Curr. Anthropol. 46 (2) (2005) 221–248.
[56] V. Midha, A. Bhattacherjee, Governance practices and software maintenance: a study of open source projects, Decis. Support Syst. 54 (1) (2012) 23–32.
[57] J.A. Roberts, I. Hann, S.A. Slaughter, Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the apache projects, Manage. Sci. 52 (7) (2006) 984–999.
[58] S. Daniel, K. Stewart, Open source project success: resource access, flow, and integration, J. Strateg. Inf. Syst. 25 (3) (2016) 159–176.
[59] J. Lerner, J. Tirole, The scope of open source licensing, J. Law Econ. Organ. 21 (1) (2005) 20–56.
[60] C. Fershtman, N. Gandal, Open source software: motivation and restrictive licensing, Int. Econ. Econ. Policy 4 (2) (2007) 209–225.
[61] W.H. Greene, Econometric Analysis. Upper Saddle River, Prentice Hall, NJ, 2003.
[62] I.-H. Hann, J.A. Roberts, S.A. Slaughter, All are not equal: an examination of the economic returns to different forms of participation in open source software communities, Inf. Syst. Res. 24 (3) (2013) 520–538.
[63] G.V. Krogh, S. Haefliger, S. Spaeth, M.W. Wallin, Carrots and rainbows: motivation and social practice in open source software development, Mis Q. 36 (2) (2012) 649–676.
[64] G. von Krogh, E. von Hippel, The promise of research on open source software, Manage. Sci. 52 (7) (2006) 975–983.
[65] E.H. Kessler, A.K. Chakrabarti, Innovation speed: a conceptual model of context, antecedents, and outcomes, Acad. Manage. Rev. 21 (4) (1996) 1143–1191.
[66] E.H. Kessler, P.E. Bierly, Is faster really better? An empirical test of the implications of innovation speed, IEEE Trans. Eng. Manage. 49 (1) (2002) 2–12.
[67] M. Spence, Job market signaling, Q. J. Econ. 87 (3) (1973) 355–374.
[68] M. Spence, Signaling in retrospect and the informational structure of markets, Am. Econ. Rev. 92 (3) (2002) 434–459.
[69] S.L. Brown, K.M. Eisenhardt, Product development: past research, present findings, and future directions, Acad. Manage. Rev. 20 (2) (1995) 343–378.
[70] P. Carbonell, A.I. Rodriguez, The impact of market characteristics and innovation speed on perceptions of positional advantage and new product performance, Int. J. Res. Mark. 23 (1) (2006) 1–12.
[71] P. Tambe, L.M. Hitt, E. Brynjolfsson, The extroverted firm: how external information practices affect innovation and productivity, Manage. Sci. 58 (5) (2012) 843–859.
[72] V. Padmanabhan, S. Rajiv, K. Srinivasan, New products, upgrades, and new releases: a rationale for sequential product introduction, J. Mark. Res. 34 (4) (1997) 456–472.
[73] A. Ali, R.J. Krapfel, D. LaBahn, Product innovativeness and entry strategy: impact on cycle time and break-even time, J. Prod. Innovation Manage. 12 (1) (1995) 54–69.
[74] K.B. Clark, T. Fujimoto, Product Development Performance: Strategy, Organization, and Management in the World Auto Industry, Harvard Business Press, Cambridge, MA, 1991.
[75] E. Mansfield, The speed and cost of industrial innovation in japan and the united states: external vs. internal technology, Manage. Sci. 34 (10) (1988) 1157–1168.
[76] R.T. Keller, Technology-information processing fit and the performance of R&D project groups: a test of contingency theory, Acad. Manage. J. 37 (1) (1994) 167–179.
[77] E.F. McDonough, Faster new product development: investigating the effects of technology and characteristics of the project leader and team, J. Prod. Innovation Manage. 10 (3) (1993) 241–250.
[78] C.D. Rosso, Continuous evolution through software architecture evaluation: a case study, J. Software Maint. Evol.: Res. Pract. 18 (5) (2006) 351–383.
[79] H. Cavusoglu, H. Cavusoglu, J. Zhang, Security patch management: share the burden or share the damage? Manage. Sci. 54 (4) (2008) 657–670.
[80] O. Temzikan, L.K. Ram, P. Sungjune, S. Chandrasekar, Patch release behaviours in response to vulnerabilities: an empircal analysis, J. Manage. Inf. Syst. 28 (4) (2012) 308–338.
[81] J.O. Gilliam, Improving the Open Source Software Model with UML Case Tools, (2001) Available on http://www.unixuser.org/lg/issue67/gilliam.html.
[82] J. Hahn, J.Y. Moon, C. Zhang, Emergence of new project teams from open source software developer networks: impact of prior collaboration ties, Inf. Syst. Res. 19 (3) (2008) 369–391.
[83] G. Winch, A. Usmani, A. Edkins, Towards total project quality: a gap analysis approach, Constr. Manage. Econ. 16 (2) (1998) 193–207.
[84] K. Crowston, H. Annabi, J. Howison, Defining open source software project success, Proceedings of International Conference on Information Systems, Association for Information Systems, Atlanta, GA, 2003, pp. 1–14.
[85] C. Subramaniam, R. Sen, M.L. Nelson, C. Subramaniama, R. Senb, M.L. Nelsonc, Determinants of open source software project success: a longitudinal study, Decis. Support Syst. 46 (2) (2009) 576–585.
[86] R. Grewal, G.L. Lilien, G. Mallapragada, Location, location, location: how network embeddedness affects project success in open source systems, Manage. Sci. 52 (7) (2006) 1043–1056.
[87] G. Peng, D. Dey, A dynamic view of the impact of network structure on technology adoption: the case of OSS development, Inf. Syst. Res. 24 (4) (2013) 1087–1099.
[88] E. Rogers, Diffusion of Innovations, Free Press, New York, NY, 2003.
[89] S.K. Shah, Motivation, governance, and the viability of hybrid forms in open source software development, Manage. Sci. 52 (7) (2006) 1000–1014.
[90] K.J. Stewart, S. Gosain, The impact of ideology on effectiveness in open source software developent teams, MIS Q. 30 (2) (2006) 291–314.
[91] E. von Hippel, G. von Krogh, Open source software and the "private-collective" innovation model: issues for organization science, Organ. Sci. 14 (2) (2003) 209–223.
[92] L. Nyman, T. Mikkonen, To fork or not to fork: fork motivations in SourceForge Projects, Int. J. Open Sour. Softw. Process. 3 (3) (2011) 1–9.
[93] A. Majchrzak, L.P. Cooper, O.E. Neece, Knowledge reuse for innovation, Manage. Sci. 50 (2) (2004) 174–188.
[94] M. Sojer, J. Henkel, Code reuse in open source software development: quantitative evidence, drivers, and impediments, J. Assoc. Inf. Syst. 11 (12) (2010) 868–901.
[95] G. von Krogh, S. Spaeth, S. Haefliger, Knowledge reuse in open source software: an exploratory study of 15 open source projects, Proceedings of Hawaii International Conference on System Sciences, Washington, DC: IEEE Computer Society, 2005, p. 198b.
[96] W. Scacchi, Free and open source development practices in the game community, IEEE Softw. 21 (1) (2004) 59–66.
[97] S. Ba, P.A. Pavlou, Evidence of the effect of trust building technology in electronic markets: price premiums and buyer behavior, MIS Q. 26 (3) (2002) 243–268.
[98] F. Zhu, X. Zhang, Impact of online consumer reviews on sales: the moderating role of product and consumer characteristics, J. Mark. Res. 74 (2) (2010) 133–148.

John Qi Dong is an Associate Professor of Strategy and Organization at Faculty of Economics and Business, University of Groningen in the Netherlands. He holds a PhD degree in information systems from Hong Kong University of Science and Technology. He also received a master degree and a bachelor degree, both in management, from Renmin University of China. His research interests include big data and analytics, collaborative innovation, digital entrepreneurship, digital innovation, and organizational learning. His

work has been published or forthcoming in *MIS Quarterly*, *Journal of Management*, *Journal of Product Innovation Management*, *Long Range Planning*, *Journal of Strategic Information Systems*, *Information and Management*, *Technological Forecasting and Social Change*, *and Journal of Business Research*, among others. His research was awarded for the best paper award runner-up at Academy of Management Annual Meeting and was nominated for the best paper award at Strategic Management Society Special Conference. He was the winner of the outstanding junior researcher award from Faculty of Economics and Business at University of Groningen. He currently serves as the associate editor for *Information and Management* and the editorial board member for *Journal of Strategic Information Systems*. He was the associate editor for *Decision Support Systems* special issue on omnichannel business. He is also the founding chair of big data and analytics track at Wuhan International Conference on E-Business.

Weifang Wu is an Assistant Professor at Department of Digitalization, Copenhagen Business School in Denmark. She holds a PhD degree in information systems from Hong Kong University of Science and Technology. She also received a master degree and a bachelor degree, both in information systems, from Harbin Institute of Technology. Her research interests include big data and analytics, digital innovation, online advertising, and social media. Her work has been published or forthcoming in *Journal of Strategic Information Systems*, and *Information and Management*, among others.

Yixin (Sarah) Zhang is a Senior Lecturer at Department of Applied Information Technology, University of Gothenburg in Sweden. She is also a member of Swedish Center for Digital Innovation. She holds a PhD degree in information systems from Hong Kong University of Science and Technology. She also received bachelor degrees (first class honors) in information systems and software engineering from University of Hong Kong. Her research interests include digital activism, green consumption, human computer interaction, and social media. Her work is forthcoming in *Information and Management* and has been published in other outlets. She was the associate editor for International Conference on Information Systems.