Advances in Complex Systems
Vol. 17, Nos. 7 & 8 (2014) 1550006 (25 pages)

DOI: 10.1142/S021952591550006X

World Scientific
www.worldscientific.com

COMMUNICATION IN INNOVATION COMMUNITIES: AN ANALYSIS OF 100 OPEN SOURCE SOFTWARE PROJECTS

MARKUS M. GEIPEL*, KERSTIN PRESS† and FRANK SCHWEITZER‡

Chair of Systems Design, ETH Zurich, Weinbergstrasse 58, 8092 Zurich, Switzerland
**markus.geipel@alumni.ethz.ch*
†*kerstin.press@access.uzh.ch*
‡*fschweitzer@ethz.ch*

Received 11 March 2013
Revised 16 March 2015
Accepted 18 March 2015
Published 19 May 2015

We develop a model of innovation communities which allows us to address in a systematic way the influence of users and developers as well as communication between and within these groups. Based on this model, we derive a formal approach to quantify communication flows, community activity and community turnover. These measures are calculated using the data of 100 open source software projects. Our empirical analysis shows that: (i) Users play indeed a predominant role in communication, which points towards the vivid role of an active user community; (ii) communication is highly concentrated, which points towards the importance of active individuals and (iii) community turnover exhibits only little correlation with community segregation, which may allow to benefit from high turnover rates while keeping negative effects small. We argue that insight from this extensive analysis not only complements existing case studies, it also provides a reference frame to put these singular results into perspective when aiming at generalizations.

Keywords: User innovation; innovation community; open source software.

1. Introduction

With increasing technological complexity and dynamics, the need for fast-paced and cost-efficient innovation grew continuously. As a response, the classical entrepreneur and firm-centered models of innovation promoted by Schumpeter [27, 28] have been supplemented with novel, open forms of innovation (see [1] among others). One major trend is the growing recognition of users: As innovators they at times succeed where firm research departments fail. The insight that many innovations — for

‡Corresponding author.

example in petrol engineering [8] — originated from users, led to the concept of "user innovation" advocated by Von Hippel [35]. While it was originally associated with user-inventors operating in a rather isolated fashion [16, 22, 29], more recent work started to emphasize the role of communities [11, 37, 38]. Especially in open source software this "private-collective" nature of the innovations is evident [40].

Fusing user innovation with a community process generates new complexity. Innovation is accompanied by interaction and the community is actively involved in developing the product: Governance structures emerge, users assume roles and collectively take decisions [5, 10, 20, 23]. This leads to a differentiation in the community. In the simplest case, it is divided into users who also act as developers and classical users. While the *user-developers* can directly implement their ideas, *classical users* need to communicate to user-developers in order to add to the project. Therefore, communication within and between both groups is an integral part of open source innovation communities.

So far, there is limited insight on this communication. Several case studies employ communication analysis as an auxiliary construct to back related theories.

For instance von Hippel and Lakhani [39] analyzed communication in the Apache field support system to answer the question: "Why would information providers voluntarily help information seekers for free?" They found that providers of help reaped direct learning benefits and that forum activity among participants was highly heterogeneous with a Gini coefficient of 0.68.[a]

Another paper by von Krogh *et al.* [41] investigated the strategies and processes by which new people join an existing community of software developers in the context of the Freenet project. Communication analysis played an important role in identifying stages of joining and joining approaches. It was found that the mean number of messages per discussion thread was 6.5, and that forum contributions by users and developers were balanced. Moreover, forum activity was once more very heterogeneous with the Gini coefficient of authorship being 0.89.

Finally, Spaeth *et al.* [32] examined source code and newsgroup data collected on the Eclipse Development Platform to investigate the contribution of private and commercial development parties. They showed that the "outsiders" invested as much communication effort in the project as did the founding firm, IBM.

These case-by-case measurements of single projects currently lack a reference frame. We cannot tell whether a Gini coefficient of 0.68 is high or not or judge whether balanced contributions of users and developers in a forum are an outstanding example of innovation communities. Moreover, community dynamics are likely to influence communication dynamics. For instance, what turnover of community participants is "normal" for an open source software project and how does it affect innovation? To address these issues, we present a comparative study of

[a]The Gini coefficient measures the degree of inequality and ranges between 0 and 1. If all contributions stemmed from only one person in the group the Gini coefficient would be 1. Conversely if contributions were evenly distributed, it would be 0. See also Sec. 3.3.

communication in innovation communities. It provides a reference for future case studies and may assist in answering questions such as "research should investigate if projects differ much in terms of turnover among various contributors, and as well what factors impact on turnover." Asked for instance in [41, p. 1236].

To achieve this aim, Sec. 2 develops a stylized model of communication and innovation in open source software communities based on the existing literature. A quantitative analysis of this model in several open source software projects allows us to determine the general characteristics of communication in these innovation communities. The specific measures for this analysis are defined in Sec. 3 alongside a discussion of the data. Section 4 presents and discusses the specific results with a list of the measures for each project being available in the appendix. Finally, Sec. 5.3 discusses the implications of the results and highlights possible avenues for future research.

2. A Model of Innovation Communities

User innovation has come a long way since von Hippel [35] coined the term. While user innovators were once mainly professional users, improving their tools and relying on the manufacturer to take up their ideas, today they are diverse in background, organized in communities and — in the case of information products — entirely independent of a manufacturer [37].

In this section, an abstract model is presented to describe the interplay between the innovating community and the innovation subject. First, the two building blocks of an innovation community are discussed: The user innovator in Sec. 2.1 and the user community in Sec. 2.2. In Sec. 2.3, the pieces are put together and present a concise model of innovation communities. Finally, this model is used to identify the relevant research questions concerning communication in such communities.

2.1. *User innovation*

User innovation denotes the fact that important innovations often stem from users of the product rather than the manufacturing firm. The phenomenon of users as sources of innovation has first been noted by Enos [8] in petroleum engineering. Since then cases of user innovation have been reported in more and more sectors. An extensive analysis and definition of the field can be found in the work of von Hippel [35].

Figure 1 represents this situation. There are two entities: First, the user ⓤ. The lower case ⓤ indicates that he or she acts as an individual. Second, the artifact Ⓐ which is subject to singular modification (⇢). The usage relationship between the user ⓤ and the artifact Ⓐ is symbolized by a *solid* arrow to express the continuity of it. To give an example: The wind-surfing pioneer Larry Stanley ⓤ attached footstraps (⇢) to his surf board Ⓐ to enable jumping. The idea for this innovation occurred to him naturally, as he is a passionate user of his surf board (←) (see [29] for details on the case). The usage relationship is of central interest, as it is

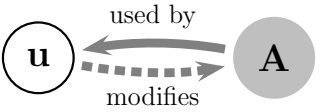

Fig. 1. Abstraction of the classical user innovation process. A single user, symbolized by an lower case ⓤ continuously uses an artifact Ⓐ (←). The knowledge gained through usage, enables ⓤ to improve Ⓐ via a single independent modification (- - →). To keep the diagram simple, the original manufacturer of Ⓐ is not shown in this figure.

connected to the acquisition of tacit knowledge about the used product, or "sticky information" as von Hippel [36] puts it. The existence of this knowledge explains why in some cases users successfully invent in fields where research departments fail: As the one applying a product, tool or technology, the user accumulates knowledge hardly accessible for producers. With the locus of knowledge, shifts the locus of innovation.

User innovation has been identified in many sectors: Ogawa [22] finds evidence with convenience stores while user innovation in sports equipment is described, among others, by Shah [29] and Luthje *et al.* [18]. Lee [16] adds machine tools to the list and Morrison *et al.* [21] study user innovation in the online public access catalog (OPAC) library system. Finally, personal computing is yet another field rich in user-innovators [4].

2.2. *User communities*

Another building block of innovation communities is the user community. In the "classical" case, ideas and knowledge are exchanged in this community. This fosters the diffusion of innovation. In most cases the user community is tied together by the shared interest in an artifact.

Figure 2 delineates this situation. The two entities involved are the community of users Ⓤ and the artifact of interest Ⓐ. We use an upper case Ⓤ to express multitude in contrast to the single user in Fig. 1. The community sustains a communication process (↶). The conversations are nourished by the continued usage (←) of the artifact.

An example of such a user community is the Homebrew Computer Club [19] which existed at Stanford University during the early days of home computing.

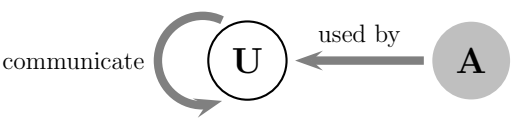

Fig. 2. Abstraction of the community process. A community of users, symbolized by an upper case Ⓤ), discusses (communication arrow) aspects of an artifact Ⓐ which is connected to them by a relationship of usage. To keep the diagram simple, the original manufacturer of Ⓐ is not shown in this figure.

Members Ⓤ such as Lee Felsenstein, John Draper or Steve Jobs joined meetings to discuss (↶) circuit layouts and programming issues (Ⓐ). An online equivalent hosting discussions on nearly every imaginable topic is the Usenet [30]. These classical user communities have attracted the interest of many scholars. Contributions range from prescriptive approaches concerning design and management [13, 24, 44] to more analytical ones [2, 15, 31].

A notable fact is that often major innovations are not developed in the user community directly. Instead, members interested in commercializing their innovations split and thus extend the community by forming a subgroup dedicated to business: Steve Jobs, Steve Wozniak and Ronald Wayne invented the *Apple I* outside the club and stopped attending the meetings afterwards.

2.3. *Innovation communities*

Several scholars have emphasized the growing importance of communities for user innovation [11, 37] due to the benefits of specialization. Especially open source software development incorporated this combination in the notion of "private collective innovation" [40]; a fusion of user innovation and user communities. Innovation communities in open source projects often start from an act of user innovation fueled by a need, not satisfied by existing products. This first step follows the schema in Fig. 1. Examples are Linux [34], Sendmail and Emacs. The subsequent development of the artifact marks a shift from lone innovators towards innovation communities by attaching a user community to the new artifact. For instance, Linus Torvalds introduced Linux to the world via a Usenet posting. Subsequently, a community formed and took part in the development process.

Due to the continuous development process of the software sustained by a multitude of individuals, the community is confronted with the following question: Who should have access to the artifact? While a general read access does not pose a problem, a general write access is highly problematic. Modifications of the artifact need to be coordinated and decisions about extensions and architectural changes need to be taken, to keep the software working and evolving. Different governance forms have emerged in open source software communities to facilitate these decision processes [10, 23]. The Apache Foundation — responsible for the Apache Web Server — established a hierarchical system based on meritocratic principles [9]. In Linux, decisions are reached through discussions with a number of 'lead' developers including Linus Torvalds himself [33].

Along with governance structure comes a diversification of member roles. Not all members have the same rights. The write access is generally very restricted and community members have to prove themselves before gaining the privilege to modify the common code repository [26]. In general, we observe a division of the community in privileged members with write access and a large number of classical users with read access only [6]. For simplicity's sake, let us refer to the former as *developers* and to the latter as *users*, while not denying that the developers also are user of the

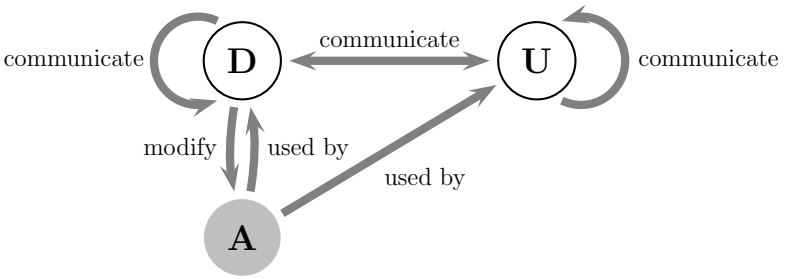

Fig. 3. Abstraction of an innovation community. A community of developers Ⓓ modify an artifact Ⓐ. A community of users Ⓤ is attached to this process by communication with the developers and a usage relationship with the artifact.

artifact. Users and developers are connected by communication. Communication allows developers to integrate users in the innovation process. Ideas or concrete contributions from users are taken up by developers acting as gatekeeper to the artifact. Moreover, communication with developers is a first step for users towards attaining the status of developers [41].

The resulting innovation scheme is depicted in Fig. 3. We see two subcommunities: Users Ⓤ and developers Ⓓ. Both use the artifact Ⓐ but only developers modify it. All relationships between Ⓤ, Ⓓ and Ⓐ are continuous. Two major differences exist between this scheme and the ones in Figs. 1 and 2: First, the contribution of users to the innovation process is indirect. They first need to communicate their contribution to a developer, who then modifies the artifact: Ⓤ→Ⓓ→Ⓐ. This starkly contrasts with the direct nonrecurring modification in Fig. 1 and entails the second difference: Communication differentiates in communication among users, communication among developers and communication connecting both sub communities: Ⓓ↔Ⓤ. There are several open questions concerning this communication.

2.4. *Aspects of communication*

Based on the previous model existing studies (see Sec. 1) we identify three important aspects of communication in innovation communities.

(i) Communication flows between Ⓓ and Ⓤ: This stream of communication serves two functions. First, it ensures that user ideas and suggestions stand a chance to be incorporated in the artifact. Second, it provides users wanting to become developers in the future with an entry point to the community [41]. Strong communication between users and developers is thus desirable for the dynamics and the viability of the project. We proxy the strength of this communication by investigating the share of user-developer communication and by studying how user-developer interaction within communication threads affect communication (proxied by discussion length and reply rates). Long discussions generally indicate more complex communication as opposed to a simple question–answer interaction. High reply rates generally indicate high involvement and reactivity of the community. Similar measures were

calculated for the Freenet project by von Krogh *et al.* [41] to shed light on joining, and specialization dynamics in the developer community.

(ii) Distribution of communication activity among Ⓓ and Ⓤ: As mentioned earlier [39, 41], open source forum communication tends to be highly concentrated with a few individuals. This provides valuable insight into the community's structure and viability: A strong concentration of the communication activity on few authors indicates a more "centralized" community, that has a higher dependency on individuals but that may well-facilitate decision making. A low concentration on the contrary would mean that the community is mainly sustained by a crowd of more or less equally active members.

(iii) Community turnover in Ⓓ and Ⓤ: In the context of open source software, turnover plays an ambivalent role: On the one hand, turnover exposes the community to new ideas and knowledge. On the other hand, turnover is inimical to integration of community members [3], which should be visible in communication patterns. Turnover is thus an important aspect of innovation communities and open source projects in particular. Moreover, a comparative analysis of turnover is a research gap yet to be filled [41].

3. A Quantitative View on Communication

In this section, the stage is set for a quantitative comparative analysis of communication in innovation communities. First, data acquisition and filtering are described. Next, in three separate subsections, empirical measures to quantify the different aspect of communication presented in Sec. 2.4 are introduced.

3.1. *Data sources and filtering*

The study is based on data from the open source incubator site SourceForge,[b] the largest Internet platform for open source projects worldwide. Two data sources were tapped. Forum communication allows us to analyze the communication of users and developers and derive various community metrics. Version control system logs[c] include every change to the software and thus indicate who was involved in the development.

As Howison and Crowston [14] and Rainer and Gale [25] pointed out, the quality of data from SourceForge is compromised by an overwhelming number of inactive, stillborn or small one-man projects. Thus, we define quality criteria to select projects with reliable data. The first criterion pertains to the version control system logs: All projects with less than one year of development activity and less than

[b]http://www.SF.net.
[c]In our case, the CVS logs. CVS stands for Concurrent Version System and is the most widely used tool to facilitate distributed work on software. For more information see http://www.nongnu.org/cvs/.

1000 change events are discarded as they are unlikely to have gathered a viable innovation community.

Before filtering out projects based on forum activity, we need to know which types of forums we are interested in. Two forum types are distinguished: Help forums and open discussion forums. In the latter, general development issues are debated. They are of major interest as this is the place where user involvement in the innovation process happens. We thus concentrate on them and exclude all help forums. This procedure is in line with the analysis conducted by von Krogh [41].

To ensure sufficient data quality, the projects are required to have a one year history of forum activity and at least 700 messages in nonhelp forums. After filtering forum and CVS data, our sample contained 100 projects meeting both criteria.[d] These projects contain an average of 3530 messages and an average of 16,901 change events. The average observation period for communication and development activity in the projects is nearly five years.

On the forum messages of the selected projects yet another filter is applied: All messages of users posting anonymously under the name "nobody" are excluded from the analysis. Scanning the data showed that such authors often post spam. Besides this, they are hardly integrated into the community. In fact, in certain communities such as the Azureus project, anonymous authors are ignored by other authors.

3.2. *Quantifying communication flows* (M, m, ρ)

To analyze communication flows as sketched in Fig. 3, two pieces of information about the messages are needed: First, whether the message initializes a thread or answers to another message. Second, whether the author is a user or a developer. While the first information is easily acquired, the classification of the author deserves a brief discussion.

To begin with, each project is treated separately. To identify the developers belonging to the project we revert to two sources of information: The SourceForge project description and the CVS logs. Every author a belongs to the group of developers ($a \in \textcircled{D}$) if one of the following two criteria is fulfilled: (i) Author a is listed as a developer on the SourceForge website. This rule only takes into account current developers. To avoid misclassification of messages by former developers, a second criterion is added: (ii) Any author a who appears in the CVS change logs of the projects is counted as a developer. We emphasize that only authors with commit privileges are considered as developers, authors who are part of the developer team but without commit privileges are classified as users.

Misclassifications occur when authors change status from user to developer or vice versa. Yet, these misclassifications are minor. Developers who "retire" from a project either leave the community or remain developers with reduced contribution.

[d]The data has been collected between January and March 2008. The CVS data was directly extracted from the CVS logs. The forum data was crawled from the SourceForge website. The SourceForge names of the selected projects can be found in Appendix A.

GraphDuplex	semiosys	2008-09-30 16:58
RE: GraphDuplex	e-flat	2008-10-26 22:29
RE: GraphDuplex	semiosys	2008-10-27 14:29
RE: GraphDuplex	duelli	2008-10-28 13:03
RE: GraphDuplex	e-flat	2008-10-28 16:33

Fig. 4. Thread in the forum of the Jung project as seen on the SourceForge website. In this case *e-flat* is a developer, the other participants are users.

Either case does not interfere with the analysis. In the case of users becoming developers, messages posted during the transition will be misclassified. The result is a slight,[e] yet unavoidable bias towards developers in the attribution of authorship.

Having classified the authors, communication flows can be calculated. To explain this procedure, let us consider an example: Figure 4 shows a thread in the open discussion forum of the Jung project. In this case *e-flat* is a Jung developer, the other participants are users.

Both users and developers may initialize new discussions. In the example the discussion is initialized by the user *semiosys*. The set of such initial postings is labelled M_*, using M_U and M_D to distinguish initial posts from users and developers. Besides the initial posts, communication splits into four different flows: User to user, ($M_{U\to U}$); user to developer, ($M_{U\to D}$); developer to user, ($M_{D\to U}$); and finally developer to developer, ($M_{D\to D}$). Equation (1) expresses the communication flows mathematically while Fig. 5 gives a graphical representation.

$$M = M_* \dot\cup M_{*\to *} = (M_U \dot\cup M_D) \dot\cup (M_{U\to U} \dot\cup M_{D\to U} \dot\cup M_{U\to D} \dot\cup M_{D\to D}). \tag{1}$$

The answer of *e-flat* (developer) to *semiosys* (user) in Fig. 4 is thus counted in $M_{D\to U}$, that of *duelli* (user) to *semiosys* (user) contributes to $M_{U\to U}$ and so on. In total, the discussion is started by a user ($M_U = 1, M_D = 0$), *e-flat* (developer) replied twice to users ($M_{D\to U} = 2$), no user replied to the developer ($M_{U\to D} = 0$) and two users (*semiosys*, *duelli*) to user *semiosys*, i.e., $M_{U\to U} = 2$.

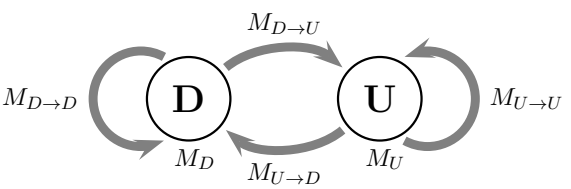

Fig. 5. A more detailed view of the message flows within an open source software community consisting of users Ⓤ and developers Ⓓ. There are four flows generated by answer messages ($M_{U\to U}, M_{D\to U}, M_{U\to D}, M_{D\to D}$) and two constituting initial posts (M_U, M_D).

[e]An estimate of less than 2% seems reasonable.

When comparing projects with different numbers of users and developers, absolute values for the message flows are impractical. We therefore use relative message flows and refer to them as m. In the case of M_U, m_U is calculated as follows:

$$m_U = \frac{|M_U|}{|M|}. \tag{2}$$

Similarly, we can calculate m for the other communication flows. For instance, $m_{U\to D}$ denotes the share of user replies to developers in total forum communication whereas $m_{D\to U}$ that of developers replying to users. For the discussion in Fig. 4, this means $M = 5$, $m_{D\to U} = 2/5 = 40\%$ and $m_{U\to D} = m_{U\to U} = 1/5 = 20\%$. Adding the share of the initial user post ($m_U = 1/5 = 20\%$) completes the description of this communication excerpt.

One caveat has to be added to the calculation of communication flows: The just presented measures only catch the publicly observable communication and do not comprise communication via email, phone conferences, chat rooms and so forth. This may in particular concern the measure $m_{D\to D}$ as developers are likely use a variety of communication channels to communicate amongst themselves. We argue, however, that this missing communication is not a thread to the validity of the presented analysis as the focus of this paper is the entire community with a special focus on interaction between users and developers. These aspects are well covered by the public forum communication.

Besides communication flows, average thread length and reply rates are important characteristics of forum communication. The reply rate ρ_U for user postings is calculated as follows:

$$\rho_U = \frac{|M_{*\to U}|}{|M_{U\to *} \dot{\cup} M_U|}. \tag{3}$$

Here, $*$ stands for either U or D, i.e., $|M_{*\to U}|$ considers *all* messages received by users. ρ_D is calculated along the same lines. In the example of Fig. 4 ρ_D equals 0 as no one answered to the developer messages, and ρ_U equals 4/3 as the 3 user messages received 4 answers.

Next, the length of threads, ρ^*, is measured:

$$\rho^* = \frac{|M|}{|M_U| + |M_D|}. \tag{4}$$

To dig deeper we also differentiate between the number of replies a user-started thread accumulates on an average (ρ^*_U) and the number of replies developer-started threads accumulate (ρ^*_D). Let $R \subseteq M^2$ be the reply relation between the messages in a forum such that if message b is an answer to message a then $(a, b) \in R$. Let further R^* be the transitive closure of R. For instance the last message from e-flat (Fig. 4) is not a direct answer to semiosys' first posting. It would thus be in R^*, but not in R. The average length of user initiated threads is then given by

$$\rho^*_U = \frac{|\{(a, b) \in R^* : a \in M_U\}|}{|M_U|}. \tag{5}$$

ρ_D^* is calculated along the same lines. If the example in Fig. 4 had been the only thread in the forum, ρ_U^* would have been 4.

3.3. *Quantifying communication activity* $(\mathbf{L}(x), g)$

To analyze concentration of communication, the Lorenz curve [17] and the Gini coefficient [12] are applied. First, we define a set of authors $a \in A$. M_a is used to refer to the messages authored by a while M_A refers to the messages authored by all authors A. Next a contribution c_a is defined as the share of messages a wrote:

$$c_a = \frac{|M_a|}{|M_A|}. \tag{6}$$

The contributions are ordered in ascending order:

$$k < l \Rightarrow c_k \leq c_l. \tag{7}$$

Finally, the Lorenz curve $\mathrm{L}(x)$ with $x \in [0, 1]$ is calculated by cumulating the first x percent of the contributions:

$$\mathrm{L}(x) = \sum_{k=0}^{\lfloor x*|A| \rfloor} c_k. \tag{8}$$

$\mathrm{L}(x)$ is interpreted as follows: The x percent least active authors wrote $\mathrm{L}(x)$ percent of the messages. If every author writes an equal number of messages, $\mathrm{L}(x)$ takes the form $\mathrm{L}(x) = x$; a straight diagonal line. The area between this line and $\mathrm{L}(x)$ serves as a measure for concentration. Normalized to the interval $[0, 1]$ it is called the Gini coefficient:

$$g = 1 - 2 \int_0^1 \mathrm{L}(x) dx. \tag{9}$$

Let us use $\mathrm{L}(x)$ and g to measure concentration of communication both among users and developers.

3.4. *Quantifying community turnover* (τ)

Users and developers come and go. To estimate this turnover, the concept of active users (U_t) and active developers (D_t) has to be established. U_t is read as: Users active at time t. A user is active if he or she posted at least one message in the "recent past". In the same vein, a developer is active if he or she committed at least one change to the code in the "recent past". Six months are a reasonable choice for this "recent past": If a user or developer was not active for more than half a year, he or she can hardly be called active. Moreover, periods shorter than a few months would count sporadic contributors erroneously as turnover. Apart from that the time window is the same for each project thus any plausible choice will do for a comparative analysis.

Based on U_t and D_t the turnover of the community can be calculated. It is commonly measured as the percentage of people leaving a group in a certain period of time. Accordingly we calculate the turnover in the user community (τ_U) as the average percentage of users leaving the set of active users U_t per period Δt:

$$\tau_U = \frac{1}{T} \sum_{t=0}^{T} \frac{|U_{t-\Delta t} \textcircled{U}_t|}{|U_t|}. \tag{10}$$

Let $t = 0$ be the start of the recorded project activity and $t = T$ be the end. Δt was chosen to be six months. So, $\tau_U = x$ reads as x percent of the users leave the community on average every six months. The turnover in the developer community (τ_D) is measured based on the sets of D_t following the same scheme.

4. Empirical Results

In this section, the aforementioned measures are put to work. The three sections run parallel to the three different aspects of communication sketched in Sec. 2.4. First, findings on of the communication flows are presented. Second, the activity of the authors is analyzed. The third part of this section is dedicated to the relationship between community turnover and communication.

4.1. *Communication flows between Ⓓ and Ⓤ*

The results on the communication flows and discussion length answer our first research question and set the stage for further analysis. Figure 6 shows the communication flow scheme of Fig. 5 with the empirical measurements. The numbers give the average relative flow volumes in the 100 projects. The dominant flows are the one within the user community, $m_{U \to U}$ with 32.6%, and the flow from developers to users, $m_{D \to U}$ with 24%. It can also be seen that 22.9% of the messages are written by users to start a new discussion thread. Users thus start 86.1% of all threads. All in all messages by users make up 62.7% of the total forum communication. This is interesting, given the fact, that only open discussion forums are considered and *not* help forums where user involvement is necessarily high. From the flows it can also be deduced that users and developers mix quite well: Users receive 42.39% of the replies to their postings from developers, and developers receive 42.61% of the replies to their postings from users.

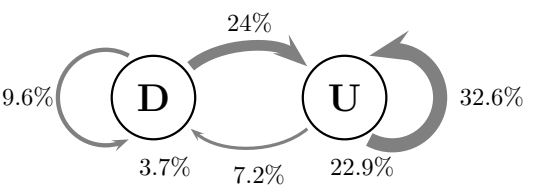

Fig. 6. Circles represent users and developers, arrows reply messages. Messages starting a new thread are written below the circles. All measures are shares.

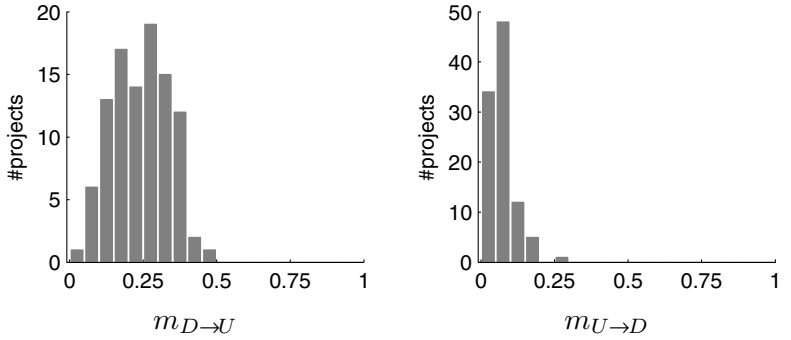

Fig. 7. Distribution of the shares of communication between users and developers in forum communication for 100 open source software projects.

Let us now take a closer look at the message flows connecting users and developers ($m_{D\to U}$ and $m_{U\to D}$). Figure 7 shows their size distributions. The flow $m_{D\to U}$ exhibits a broad distribution: For 90% of the projects $m_{D\to U}$ lies between 10% and 40%. In comparison, the flow $m_{U\to D}$ exhibits a narrow distribution with over 90% of the projects between 0 and 0.15. This means that developers differ from users who exhibit a more uniform communication behavior.

Concerning the reply rates, we observe an interesting fact: On the one hand, user postings receive more replies on average than developer postings: $\bar{\rho}_U = 0.94$ versus $\bar{\rho}_D = 0.41$. On the other hand, threads started by developers spawn longer discussions on average: $\bar{\rho}^*_D = 5.49$ versus $\bar{\rho}^*_U = 3.96$. At first sight this is counter-intuitive. A plausible explanation is that developer threads attract users who then fuel the discussion. A correlation analysis was conducted to test this. The results are depicted in Table 1.

The average thread length of developer-started discussions (ρ^*_D) has a weak negative correlation with m_U, meaning that prolific initial posting by users lowers average discussion length. Contrariwise, discussion length benefits from a strong flow from users to developers ($m_{U\to D}$), and interestingly from communication between developers ($m_{D\to D}$). This indicates that developer-started discussions may benefit from user involvement as pointed out in the previous paragraph. A quite similar pattern can be observed for the number of replies to user-started threads, ρ^*_U: Again, there is a negative correlation with initial postings by users, m_U. This seems to be plausible given the calculation of of the reply rate ρ (see Eq. (3)). Nevertheless a likewise effect is not seen for the share of initial posting by developers, m_D. As it

Table 1. Correlations of communication and discussion length.

	m_U	m_D	$m_{U\to U}$	$m_{D\to D}$	$m_{D\to U}$	$m_{U\to D}$
ρ^*_D	-0.25^*	-0.12	$+0.21^*$	-0.03	-0.18	$+0.52^{**}$
ρ^*_U	-0.82^{**}	$+0.20$	-0.09	$+0.32^{**}$	$+0.19$	$+0.39^{**}$

Note: Statistical significance: $^{**}p < 0.01$, $^{*}p < 0.05$.

was the case for the number of replies to developer-started threads (ρ_D^*), the of replies to user-started threads (ρ_U^*) seems to benefit from discussion in the other subcommunity as well as from users replying to developers. This result is in line with the average reply rates discussed earlier: It also indicates strong involvement of users in developer discussions and vice versa.

4.2. *Distribution of communication activity in Ⓓ and Ⓤ*

Going beyond the aggregated message flows of users and developers, we focus on the authors of the messages and present the degree of concentration of communication activity represented by Lorenz curves ($\mathrm{L}(x)$) and the corresponding Gini coefficients (g).

Figure 8 shows the Lorenz curves for each subcommunity in the Azureus project. Only 20% of the developers are responsible for over 95% of the developer activity in the forums. Also among users, the communication effort is distributed in an unequal way: 20% of the users write over 75% of the messages. A straightforward measurement of this concentration is the Gini coefficient g (Eq. (9)). For Azureus $g_D = 0.88$ and $g_U = 0.71$. As $\mathrm{L}(x)$ cannot be shown for all projects, only the distribution of the corresponding Gini coefficients g are shown in Fig. 9. The Gini coefficient is high, both for user and developer communities. The averages are $\bar{g}_D = 0.76$ and $\bar{g}_U = 0.68$, respectively. To sum things up: Inequalities in communication are strong, both among users and developers. A small number of individuals is responsible for a large part of the communication.

4.3. *Community turnover in Ⓓ and Ⓤ*

The results of community turnover are presented in two steps: First the turnover rates are presented, followed by the results on the correlation between turnover and communication flows.

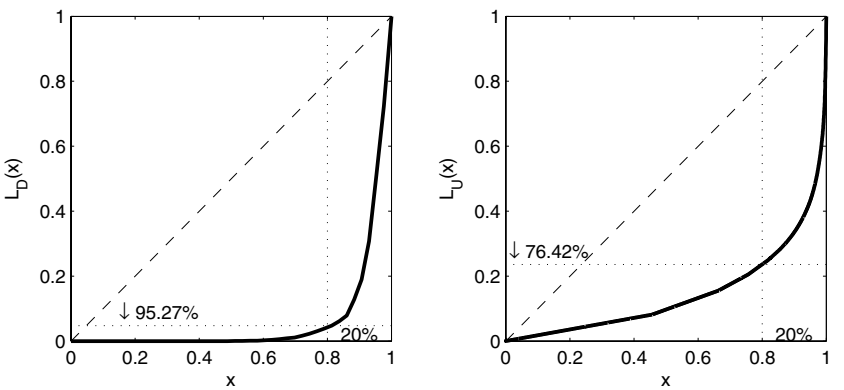

Fig. 8. The Lorenz curves for developer communication (left) and user communication (right) in the Azureus project.

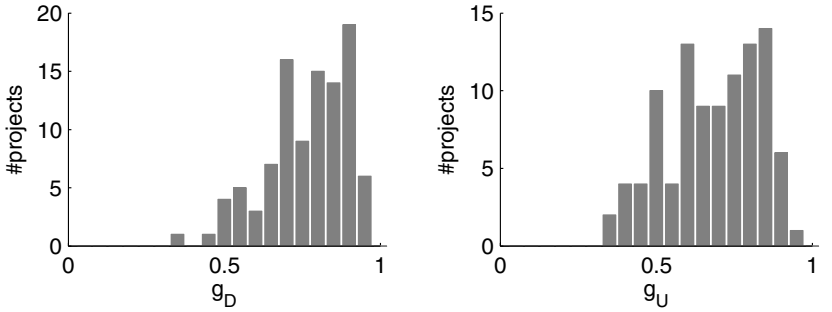

Fig. 9. The distribution of the Gini coefficient g in the 100 projects. Left: Among developers. Right: Among users.

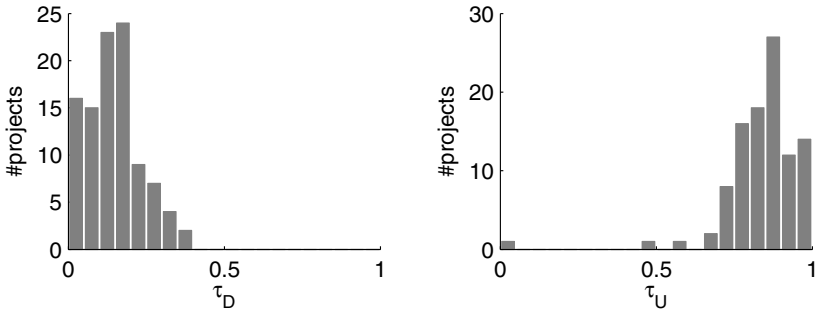

Fig. 10. Turnover in the 100 projects. Left: $\bar{\tau}_D$, turnover among developers. Right: $\bar{\tau}_U$, turnover among users.

In the developer community, on an average, about 15% ($\bar{\tau}_D = 0.15$) of the members leave every six months. For the user community this value ($\bar{\tau}_U$) is significantly higher: 84% of the users leave every six months. Even though τ_D is small compared to τ_U, it still is considerable.

Of course, particular projects deviate from these averages. Figure 10 shows the distribution of τ_D and τ_U for all 100 projects. For the user turnover τ_U, there is one outlier at $\tau_U = 0$. This is not an artifact: 13 users were active in the forum of the project Tigermud for nearly 2 years, until development was stopped on 19 November 2005.

The remainder of this section is dedicated to correlations between turnover and the other measurements. First of all, turnover among users τ_U and turnover among developers τ_D are not correlated with each other. τ_U and τ_D show several correlations with communication flows and reply rates, though.

In Table 2, the communication flows are correlated with the turnover rates: It can be seen that high turnover among users or developers goes along with segregation between users and developers, meaning that both subcommunities tend to interact less. In the case of turnover in the user community (τ_U), we see a significant

Table 2. Correlations of turnover rates and communication flows.

	m_U	m_D	$m_{U\to U}$	$m_{D\to D}$	$m_{D\to U}$	$m_{U\to D}$
τ_U	$+0.38^{**}$	-0.11	$+0.08$	-0.20	-0.01	-0.32^{**}
τ_D	-0.06	$+0.38^{**}$	-0.20^{*}	$+0.34^{**}$	-0.31^{**}	-0.09

Note: Statistical significance: $^{**}p < 0.01$, $^{*}p < 0.05$.

Table 3. Correlations of reply rates and turnover for 100 open source software projects.

	ρ_D^*	ρ_U^*	ρ_D	ρ_U
τ_U	-0.16	-0.50^{**}	-0.31^{**}	-0.20^{*}
τ_D	-0.15	-0.15	$+0.12$	-0.22^{*}

Note: Statistical significance: $^{**}p < 0.01$, $^{*}p < 0.05$.

positive correlation with the share of messages posted by users (m_U). New users thus increase the share of initial postings. Furthermore, τ_U is negatively correlated with the share of messages from users to developers ($m_{U\to D}$). For the turnover in the developer community (τ_D) the relation with segregation is even more conspicuous: Developers start more threads and reply more often to each other (positive correlation with m_D and $m_{D\to D}$) while replies to users decrease (negative correlation with $m_{D\to U}$).

Besides these segregation effects, high turnover is also connected to shorter discussions as evidenced by the negative correlations with the different reply rates in Table 3.

Turnover among users (τ_U) is associated with shorter discussions started by users (ρ_U^*) and lower reply rates both for user and developer postings (ρ_U, ρ_D). Then again, turnover among users (τ_U) only correlates weakly negative with the reply rate to user postings, ρ_U. Furthermore, the correlation does not imply causation. Based on the data we are not able to decide whether segregation causes higher turnover or vice versa. We only know that both go hand in hand.

5. Discussion

The conclusions are presented in two parts: First, implications of the particular results are deduced. Second, we show how the results serve as a reference frame for case studies.

5.1. *Implications of the results*

(i) Communication flows between Ⓓ and Ⓤ: In a nutshell, users play a predominant role in communication: 62.7% of the messages are posted by them. In initializing discussions, users tend to be active while developers are more reactive. Furthermore, long threads are generated by the interaction between users and developers which

is generally strong. This shows empirically that a vibrant user community is an integral part of open source projects.

(ii) Distribution of communication activity in Ⓓ and Ⓤ: From the average behavior of community members we should not infer that open source projects are shaped by a crowd of more or less equal agents. Quite the contrary is true: Communication concentrates, meaning that communication is shaped by individuals rather than crowds. Open source software is not a self-organizing accumulation of equally active community members. The strong concentration both in user and developer communication thus supports anecdotal evidence of individuals taking lead roles in a project (see Refs. 26, 42 and 43). Furthermore, our results for 100 projects match with the concentration value measured by von Krogh *et al.* [41] in the Freenet Project. The finding that communication activity concentrates has the following corollary: It is rather important to acquire several high impact users and developers. The mere number of contributors (user as well as developers) is secondary.

(iii) Community turnover in Ⓓ and Ⓤ: The analysis shows that turnover and segregation are associated. This finding adds to an ongoing controversy on turnover: Raymond [26], for example, sees turnover as a chance to acquire novel ideas from new members. De Marco and Lister [7], and Brooks [3] on the other side argue decidedly against high turnover in developer teams as it undermines identification with the group, nullifies positive effects of specialization and causes a higher organizational overhead. The positive correlation of turnover with segregation seems to back the latter opinion. Yet, the correlations are only small to medium in strength. Furthermore, developer turnover hardly affects reply rates. This means that there is the possibility to benefit from high turnover and still keep negative side effects small. Projects in which this is the case would be a worthwhile target for further case studies.

5.2. *Towards a frame of reference*

Apart from the specific findings just discussed, we can draw an important conclusion with respect to research on open source: A lack of strong correlations and the strong concentration of communication activity on a small number of individuals entail that no case study is likely to yield results valid for open source projects in general. At the same time, it means that case studies are valuable and necessary to fully understand the plethora of existing projects as no general rule will be able to describe them. Therefore, we argue that large scale quantitative analysis has to go hand in hand with case study-based research. The following paragraphs exemplify this synergy.

In Sec. 2.4, we showed that concentration of communication, the share of user communication among total communication, turnover etc. have been measured by several scholars in the context of case studies. The problem is that these isolated numbers are missing a reference frame. The work presented in this paper makes the first step to establish such a reference.

Let us take for example the concentrations of communication measured by von Krogh *et al.* [41], and von Hippel and Lakhani [39] for the Apache field support system and the Freenet project: The respective Gini coefficients were 0.68 and 0.89. We can now augment these pieces of information by comparing them to our sample. Of the 100 analyzed projects 16% exhibit a lower concentration than 0.68, and only 8% exhibit a higher concentration than 0.89. This implies that the Apache field support system and the Freenet project are situated on the two opposite ends of the spectrum. This new information paves the way for new research investigations: For instance, given now that the Apache field support system and the Freenet forums differ significantly in concentration of communication activity, it might be insightful to investigate the causes and ramifications of this fact.

To better support such an investigation, it might prove useful to extend the reference frame presented in this paper. For instance, different measures were averaged over time. A closer look at the change of these measures over time might provide further insights. It would allow us to control for project age, and in combination with additional information such as change effort per software module, for example, measure the maturity of a project. Furthermore, in this paper we concentrated on forum communication, taking our cues from previous publications in this field [39, 41]. Nevertheless, not all communication in open source projects is captured by these forums. Other communication channels include mailing lists, bug reports and wikis. These different communication channels might be used by users and developers in different ways. Thus, incorporating these alternative channels into a reference frame certainly broadens its applicability and scope.

5.3. *Concluding remarks*

In this paper, we shed light on the communication between users and developers in 100 open source projects. To structure the analysis we developed an abstract model of innovation communities based on the existing literature. In terms of what to measure we took our cues from existing case studies: We selected communication and community aspect which either were idenfied as important by these case studies and measured, or just proposed to be measured in future research.

The results of our analysis contribute to the current state of research in two ways: First, the measurements of important communication and community characteristics across 100 projects allowed us to extract general patterns which could have not been found in a case study. Examples are the high concentration of communication activity on only a few authors across virtually all projects, or the relationship between turnover and segregation between users and developers.

The second contribution consists of the establishment of a reference frame for existing and future case studies. The tables printed in Appendix A allow researcher to put the characteristics of their project of interest in context. We hope that especially this reference frame sets the stage for research to come.

Acknowledgments

The authors would like to thank the Swiss National Fund for financially supporting our research on open source software projects (Grant No. CR12I1_125298).

Appendix A. Detailed Results

The tables on the following pages list the raw results of our analysis for each project, as well as aggregate information such as mean, standard deviation etc., on each measured aspect. The following abbreviations are used in the tables:

name : SourceForge short name of the project.
changes : Number of events recorded in the change logs.
size : Size of the project as number of files.
msg : Number of messages posted in the forums.
age : Age of the project in days.
$m_U \cdots m_{UD}$: Message flows as described in Sec. 3.2.
ρ^* : Average thread length in the forums as described in Sec. 3.2.
g_U, g_D, g : Gini coefficients of the concentration of communication on authors. See Sec. 3.3.
τ_U, τ_D : Turnover rates in the user and the developer community. See Sec. 3.4.
domain : Application domain of the software as listed on SourceForge.

Table A.1.

Name	Changes	Size	msg	age	m_U	m_D	m_{UU}	m_{DD}	m_{DU}	m_{UD}	ρ^*	g_U	g_D	g	τ_U	τ_D	Domain
ajaxtags	3091	576	1900	929	39.1	0.6	41.8	0.3	16.1	2.1	2.52	0.82	0.71	0.84	96.5	32.1	Object Oriented
architecturware	51,698	12,968	2181	1525	20.4	4.6	28.7	10.6	28.2	7.6	4.01	0.82	0.88	0.82	72.9	8.8	Code Generators
arianne	57,867	5622	1177	2856	22.0	1.3	48.4	3.1	18.5	6.6	4.30	0.79	0.92	0.80	93.5	17.0	Internet
awstats	8171	756	19,504	2611	43.6	0.0	47.9	0.1	7.4	1.1	2.29	0.83	0.70	0.84	88.1	3.2	CGI Tools/Libraries
azureus	52,328	5834	24,194	1577	22.2	0.8	52.0	1.6	17.0	6.3	4.34	0.71	0.88	0.76	83.6	13.4	Internet
bo2k	1819	459	1132	2482	36.2	0.9	42.0	0.9	17.8	2.2	2.70	0.67	0.82	0.72	96.2	16.7	Networking
bochs	22,715	837	1214	2578	37.7	0.7	40.6	1.7	17.7	1.6	2.60	0.39	0.83	0.49	91.7	16.4	Emulators
bzflag	58,147	5199	1233	2580	25.6	4.4	27.8	8.9	27.0	6.2	3.33	0.49	0.86	0.62	90.9	12.9	Simulation
cmusphinx	39,114	254	3857	2834	29.2	1.1	37.0	2.8	25.8	4.1	3.30	0.60	0.81	0.69	81.5	14.1	Speech
dev-cpp	8271	761	69,003	1815	18.9	0.2	66.0	0.9	11.7	2.3	5.24	0.95	0.84	0.95	83.8	9.4	Software Development
dirac	4721	978	894	1329	19.8	3.9	27.9	3.8	35.8	8.8	4.22	0.70	0.83	0.78	82.8	16.6	Video
directshownet	3042	869	5508	909	18.3	0.2	34.8	1.1	36.7	8.9	5.39	0.63	0.66	0.76	76.6	0.0	Still Capture
drm	6656	330	806	1755	13.2	2.2	40.8	1.9	26.3	15.6	6.50	0.67	0.82	0.74	79.3	7.8	Sound/Audio
drpython	3434	227	1331	629	5.1	29.1	5.3	39.6	11.7	9.2	2.92	0.61	0.68	0.88	89.8	27.8	Education
dws	4015	1557	1067	1353	32.1	1.7	35.0	2.2	24.6	4.3	2.96	0.82	0.77	0.82	83.0	25.1	Compilers
easyeclipse	47,605	21,011	706	993	27.8	0.3	39.5	0.9	28.1	3.5	3.57	0.84	0.91	0.87	100.0	13.5	Software Development
ehcache	2843	376	941	1143	38.6	1.3	30.7	0.5	24.6	4.4	2.51	0.41	0.82	0.56	88.3	0.0	Other/Nonlisted Topic
emma	2031	554	2836	657	24.5	2.1	27.5	8.4	32.7	4.8	3.76	0.69	0.72	0.81	88.1	12.5	Quality Assurance
eumex	1972	173	1058	1790	12.3	1.5	31.5	4.9	45.2	4.6	7.25	0.74	0.61	0.84	70.5	10.0	Telephony
exponent	16,912	3136	2236	720	17.8	4.7	31.8	13.3	20.9	11.5	4.45	0.57	0.57	0.69	86.1	18.5	Site Management
fluxbox	8509	380	1112	2100	39.0	0.1	51.9	0.0	8.5	0.5	2.56	0.85	0.73	0.85	94.3	16.2	Window Managers
freeimage	4696	858	2802	2674	26.1	2.3	28.4	6.8	30.4	6.1	3.52	0.51	0.68	0.69	86.3	14.3	Viewers
freemercator	2013	558	959	409	21.5	7.2	29.6	10.4	20.1	11.2	3.49	0.77	0.56	0.80	85.0	33.3	Point-Of-Sale
freemind	12,227	1041	6922	2687	39.2	1.8	29.7	5.5	11.3	12.6	2.44	0.74	0.65	0.78	83.6	12.3	Graphics
fwbuilder	49,755	2273	4353	2738	27.2	0.6	27.5	0.2	37.7	6.7	3.60	0.51	0.74	0.70	88.5	4.9	Firewalls
g3d-cpp	18,001	4264	7807	1682	7.2	23.3	8.7	42.0	14.3	4.6	3.29	0.76	0.84	0.91	78.9	15.6	Games/Entertainment
ganttproject	11,622	1911	1610	1737	28.3	3.5	22.9	4.3	31.4	9.6	3.14	0.39	0.91	0.61	93.3	21.0	Scheduling
gift	4998	178	1353	1803	24.6	0.9	57.0	0.6	11.5	5.4	3.92	0.84	0.83	0.85	97.7	13.7	File Sharing
gimp-print	17,955	366	763	3701	35.5	0.1	29.1	1.1	29.5	4.7	2.81	0.34	0.94	0.52	97.1	13.3	Printing
gltron	6587	645	1784	1869	19.1	1.1	54.5	0.7	15.6	9.1	4.96	0.89	0.91	0.89	76.8	11.8	Games/Entertainment
gpac	11,977	1980	1233	1590	28.5	1.0	26.6	1.6	33.8	8.5	3.40	0.65	0.78	0.76	77.7	9.4	Sound/Audio
gxsm	12,836	2140	1043	3463	7.2	12.1	13.5	41.3	20.7	5.2	5.19	0.57	0.89	0.81	89.1	12.9	Hardware Drivers
hibernate	60,885	6914	16,580	1553	24.9	1.3	29.8	3.0	33.4	7.6	3.82	0.61	0.96	0.75	90.9	13.5	Database
hsqldb	14,929	2722	2133	2406	20.0	4.2	19.9	17.8	29.8	8.3	4.13	0.62	0.88	0.78	92.7	12.8	Database Engines/Servers

Table A.1. (*Continued*)

Name	Changes	Size	msg	age	m_U	m_D	m_{UU}	m_{DD}	m_{DU}	m_{UD}	ρ^*	g_U	g_D	g	τ_U	τ_D	Domain
ibatisdb	3225	673	4942	870	25.5	1.3	43.8	2.0	21.9	5.4	3.73	0.83	0.49	0.86	90.0	0.0	Software Development
ibs	4274	1257	2056	638	27.4	0.5	38.9	1.1	27.5	4.6	3.59	0.83	0.72	0.86	76.5	12.5	Systems Administration
jep	4030	426	1115	2387	28.2	0.4	28.2	0.9	38.7	3.5	3.50	0.86	0.61	0.89	91.3	6.3	Mathematics
jgraph	6409	1697	865	1906	28.5	5.1	21.7	11.0	28.2	5.4	2.97	0.62	0.88	0.74	96.8	23.3	Visualization
jhotdraw	6867	1256	743	1657	18.2	8.3	11.8	25.7	27.5	8.5	3.77	0.54	0.82	0.74	95.1	15.7	Vector-Based
jpf	2858	669	1871	2338	22.9	1.1	34.8	0.9	32.9	7.4	4.16	0.71	0.73	0.80	76.8	0.0	Frameworks
jtds	4877	212	1691	2222	12.6	6.6	15.7	33.2	23.5	8.3	5.20	0.50	0.78	0.79	87.4	27.4	Database
jung	17,542	1691	5409	1677	23.8	0.7	31.5	1.9	37.0	5.1	4.09	0.85	0.76	0.90	83.5	7.1	Visualization
kinterbasdb	1770	166	809	2212	24.3	1.0	27.7	4.8	34.6	7.5	3.95	0.69	0.71	0.79	75.3	2.8	Front-Ends
kopete	3860	840	2351	1640	39.7	0.1	53.5	0.0	6.3	0.3	2.51	0.89	0.92	0.89	98.2	29.5	ICQ
lush	6236	1622	782	2335	30.8	6.0	27.9	4.9	26.6	3.8	2.72	0.58	0.82	0.70	79.0	10.8	Artificial Intelligence
mantisbt	22,708	925	2933	2527	41.4	0.9	36.5	1.3	16.1	3.8	2.36	0.81	0.96	0.84	90.0	18.8	Software Development
mfcti	7445	13	1948	398	15.9	0.2	70.7	0.1	10.4	2.7	6.20	0.69	0.50	0.72	83.9	0.0	Real Time Strategy
mingw	35,478	16,659	3198	2895	30.3	0.6	40.9	2.2	21.9	4.1	3.24	0.48	0.90	0.59	89.9	15.1	Build Tools
mjbworld	5727	1913	1307	1392	18.5	0.5	36.7	1.4	37.8	5.1	5.27	0.79	0.67	0.85	70.8	8.3	Three-dimensional Modeling
movix	4738	1327	1459	1033	21.4	2.7	26.9	9.0	30.9	9.0	4.14	0.46	0.91	0.66	94.0	19.6	Software Distribution
nasm	3122	312	3406	1961	27.3	0.1	45.2	1.0	24.4	1.9	3.64	0.90	0.90	0.91	89.2	26.6	Compilers
nco	13,836	320	1470	3361	8.4	13.7	7.1	45.0	17.3	8.4	4.52	0.80	0.70	0.89	90.0	6.2	Mathematics
nst	19,266	2152	793	1575	22.8	0.8	28.6	0.9	40.0	6.9	4.24	0.46	0.65	0.67	83.3	0.0	Communications
nunit	14,298	1187	1342	2607	28.2	0.7	34.7	0.6	27.8	8.0	3.46	0.46	0.86	0.60	86.5	10.3	Software Development
obiblio	5397	368	779	2099	26.2	1.5	39.5	1.9	22.3	8.5	3.61	0.78	0.69	0.81	71.3	10.2	Library Metadata
open-beos	60,284	23,337	1027	1305	27.7	0.7	66.5	0.1	3.6	1.5	3.53	0.50	0.80	0.49	84.4	10.3	Desktop Environment
opencyc	4574	675	2072	1986	21.8	5.5	37.5	7.6	19.9	7.6	3.66	0.61	0.71	0.71	81.4	21.3	Artificial Intelligence
openemr	12,773	3390	3254	1904	9.6	8.1	14.9	35.6	23.0	8.8	5.64	0.68	0.67	0.83	74.2	17.6	Office/Business
openknights	5362	1331	1406	1110	2.8	15.0	5.6	54.5	10.5	11.6	5.60	0.64	0.73	0.79	84.6	24.0	Role-Playing
openqrm	16,039	15,596	1052	784	16.2	2.0	32.9	5.8	38.8	4.4	5.51	0.87	0.91	0.85	90.0	25.0	Clustering
openxava	161,717	13,484	715	1093	20.3	3.1	24.1	2.7	42.8	7.1	4.28	0.65	0.93	0.78	77.8	22.9	Frameworks
owl	37,363	16,536	4086	2715	21.9	2.2	25.9	4.7	38.1	7.1	4.14	0.51	0.83	0.72	87.9	4.0	File Sharing
parchive	1043	397	2249	2142	39.4	1.8	33.7	3.0	15.8	6.4	2.43	0.89	0.80	0.89	83.6	34.6	Usenet News
phpchrystal	8638	1849	1774	817	16.7	5.9	22.9	17.9	32.2	4.4	4.42	0.61	0.75	0.76	82.8	13.1	Games/Entertainment
phpgedview	36,422	2410	4747	1857	12.7	5.6	26.7	17.7	26.1	11.2	5.46	0.66	0.72	0.77	72.0	16.2	Genealogy
phpopenchat	7633	957	894	2005	44.9	0.1	48.0	0.0	6.3	0.8	2.22	0.37	0.81	0.40	97.5	19.4	Chat
phpscheduleit	1412	266	1183	826	25.6	1.5	24.7	3.7	35.2	9.3	3.69	0.65	0.88	0.78	89.9	0.0	Resource Booking
pocketinsanity	1616	430	1437	3732	25.1	0.9	50.8	0.8	10.7	11.8	3.85	0.91	0.61	0.90	79.3	8.6	Emulators
popfile	8966	898	25,519	1934	11.6	3.5	32.0	16.8	19.7	16.4	6.60	0.81	0.55	0.88	77.9	10.6	Filters

Table A.1. (*Continued*)

Name	Changes	Size	msg	age	m_U	m_D	m_{UU}	m_{DD}	m_{DU}	m_{UD}	ρ^*	g_U	g_D	g	τ_U	τ_D	Domain
ptypes	4167	211	881	1835	26.0	1.2	34.5	1.5	30.2	6.6	3.67	0.74	0.72	0.80	74.1	20.0	Software Development
pydev	15,121	1944	3294	1579	25.3	0.2	29.5	0.5	38.3	6.1	3.91	0.61	0.85	0.76	91.8	18.1	Software Development
qtstalker	12,651	559	1783	2388	12.4	5.1	20.6	14.1	30.2	17.5	5.71	0.73	0.63	0.83	77.1	2.6	Investment
quantum	8594	1124	1059	2664	22.9	1.4	21.0	4.8	40.2	9.7	4.12	0.40	0.72	0.65	90.0	14.5	Front-Ends
quartz	5327	412	7070	1492	30.4	1.2	35.0	0.9	28.8	3.8	3.17	0.84	0.90	0.88	87.2	16.5	Software Development
reactivision	2549	637	1029	773	19.3	0.8	53.0	0.8	19.5	6.6	4.97	0.75	0.67	0.79	60.0	0.0	Human Machine Interfaces
rimps	1756	84	789	2825	26.4	4.3	33.1	9.9	18.8	7.6	3.26	0.79	0.36	0.77	95.6	27.8	WWW/HTTP
rssowl	42,353	1385	1188	1582	23.2	1.8	31.2	2.9	31.7	9.3	4.01	0.76	0.83	0.84	93.3	0.0	WWW/HTTP
secureideas	2804	248	1236	1395	22.3	0.1	43.2	0.4	30.5	3.5	4.46	0.44	0.91	0.59	89.3	16.4	Systems Administration
seq	10,440	943	2989	2668	15.9	3.9	42.0	7.6	16.6	14.1	5.06	0.60	0.76	0.69	83.0	19.5	Multi-User Dungeons (MUD)
sim-icq	28,132	1352	2584	1287	37.1	0.3	40.2	0.5	18.0	3.9	2.67	0.52	0.90	0.60	89.1	16.5	AOL Instant Messenger
smartwin	16,517	2527	956	1413	15.9	6.8	17.7	23.3	30.1	6.2	4.41	0.73	0.82	0.82	79.8	9.8	Frameworks
snap	1872	751	4392	1580	5.2	28.2	7.5	45.8	5.4	7.9	3.00	0.71	0.54	0.55	100.0	37.5	Software Development
sqlunit	3586	333	1116	1195	12.5	7.6	16.9	25.1	29.8	8.2	4.98	0.58	0.82	0.80	83.2	10.0	Front-Ends
synergy2	2592	467	1978	1780	23.3	0.4	47.3	0.8	13.8	14.6	4.23	0.83	0.50	0.85	87.9	7.1	Desktop Environment
tenebrae	2701	704	4438	852	16.2	0.6	64.1	2.1	11.7	5.4	5.97	0.74	0.45	0.76	87.0	18.1	First Person Shooters
tigermud	2903	160	1909	520	2.9	17.3	4.3	58.4	10.5	6.6	4.95	0.50	0.73	0.84	0.0	34.3	Multi-User Dungeons (MUD)
tikiwiki	95,082	8992	775	1850	20.6	10.7	18.2	18.6	21.6	10.3	3.19	0.52	0.96	0.64	98.3	24.5	Site Management
tutos	25,292	1552	862	2732	44.0	0.9	40.8	0.4	12.2	1.7	2.23	0.74	0.91	0.75	85.0	13.6	Dynamic Content
tyrant	7757	247	2494	2508	14.6	5.8	31.1	22.1	18.3	8.1	4.90	0.84	0.85	0.83	72.2	39.8	Turn-Based Strategy
ufraw	2144	113	1846	1031	17.1	1.6	39.3	2.3	24.5	15.3	5.37	0.81	0.70	0.84	65.4	7.3	Editors
unh-iscsi	1345	175	928	1317	25.1	0.9	40.7	1.2	26.5	5.6	3.85	0.59	0.48	0.68	84.3	18.3	Hardware Drivers
webcalendar	14,866	463	5021	2775	26.5	2.7	26.9	9.5	22.5	11.9	3.42	0.69	0.56	0.79	88.8	2.9	Resource Booking
whichbot	5187	1342	1878	1692	8.8	3.3	39.1	4.7	15.6	28.4	8.27	0.85	0.72	0.85	46.9	3.3	Real Time Strategy
wowcastparty	5280	778	5218	869	15.8	1.4	46.7	4.6	14.7	16.8	5.81	0.91	0.71	0.91	77.6	27.2	Role-Playing
xmm	3409	1735	1009	1536	18.6	6.4	16.6	26.7	27.9	3.9	3.99	0.84	0.81	0.86	100.0	21.4	Video
xoops	92,464	26,520	1635	2152	4.0	9.9	5.8	58.6	9.6	12.2	7.20	0.64	0.85	0.72	85.2	24.0	Dynamic Content
xrms	14,974	5647	2419	1586	16.5	6.1	24.1	14.8	28.0	10.5	4.42	0.57	0.82	0.74	79.0	17.5	CRM
xui	10,720	1301	1154	1121	11.1	5.5	16.3	33.8	30.2	3.1	6.01	0.60	0.88	0.82	85.7	5.6	Desktop Environment
yale	47,967	3326	2252	2063	26.1	0.6	29.7	0.4	39.5	3.7	3.75	0.78	0.95	0.84	65.3	16.7	Information Analysis
zentrack	8833	1191	993	2469	31.1	3.5	30.6	3.2	27.0	4.5	2.89	0.78	0.90	0.83	97.5	12.7	Quality Assurance

Table A.2.

Name	Changes	Size	msg	age	m_U	m_D	m_{UU}	m_{DD}	m_{DU}	m_{UD}	ρ^*	g_U	g_D	g	τ_U	τ_D	Domain
Mean	16,906.32	2731.71	3530.91	1809.46	22.88	3.70	32.62	9.64	24.00	7.16	4.08	0.68	0.77	0.77	0.84	0.15	—
Median	7695.00	934.00	1622.50	1767.50	22.89	1.53	31.35	3.01	24.60	6.61	3.88	0.69	0.81	0.79	0.86	0.14	—
Standard deviation	24,097.42	5020.48	7815.12	752.15	9.65	5.34	13.89	14.12	9.75	4.33	1.21	0.15	0.13	0.11	0.13	0.09	—
Kurtosis	16.08	11.91	51.66	2.58	2.76	12.54	3.20	5.99	2.18	7.77	3.76	2.19	3.10	4.00	22.31	2.97	—
Skewness	3.16	3.04	6.50	0.26	0.15	2.91	0.26	1.94	−0.05	1.54	0.87	−0.35	−0.77	−1.06	−3.37	0.45	—

References

[1] Allen, R. C., Collective invention, *J. Econ. Behav. Organ.* **4** (1983) 1–24.

[2] Bagozzi, R. P. and Dholakia, U. M., Open source software user communities: A study of participation in Linux user groups, *Manage. Sci.* **52** (2006) 1099–1115.

[3] Brooks, F. P., *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edn.* (Addison-Wesley Professional, 1995).

[4] Caminer, D., Aris, J., Hermon, P. and Land, F., *User-Driven Innovation: The World's First Business Computer* (McGraw-Hill, NY, 1996).

[5] Crowston, K., Li, Q., Wei, K., Eseryel, U. Y. and Howison, J., Self-organization of teams for free/libre open source software development, *Inf. Softw. Technol.* **49** (2007) 564–575.

[6] Crowston, K., Wei, K., Li, Q. and Howison, J., Core and periphery in free/libre and open source software team communications, in *HICSS '06: Proc. 39th Annual Hawaii Int. Conf. System Sciences* (IEEE Computer Society, Washington DC, 2006).

[7] DeMarco, T. and Lister, T. R., *Peopleware* (Dorset House Publishing, NY, 1987).

[8] Enos, J. L., *Petroleum Progress and Profits: A History of Process Innovation* (MIT Press, Massachusetts, 1962).

[9] Fielding, R., Shared leadership in the Apache project, *Commun. ACM* **42** (1999) 42–43.

[10] Franck, E. and Jungwirth, C., Reconciling rent-seekers and donators–the governance structure of open source, *J. Manage. Gov.* **7** (2003) 401–421.

[11] Franke, N. and Shah, S., How communities support innovative activities: An exploration of assistance and sharing among end-users, *Res. Policy* **32** (2003) 157–178.

[12] Gini, C., Measurement of inequality of incomes, *Econ. J.* **31** (1921) 124–126.

[13] Godwin, M., Nine principles for making virtual communities work, *Wired* **2** (1994) 72–73.

[14] Howison, J. and Crowston, K., The perils and pitfalls of mining sourceforge, in *Proc. Workshop on Mining Software Repositories at the Int. Conf. on Software Enginneering ICSE* (Syracuse University, 2004).

[15] Johnson, C., A survey of current research on online communities of practice, *Internet High. Educ.* **4** (2001) 45–60.

[16] Lee, K. R., The role of user firms in the innovation of machine tools: The Japanese case, *Res. Policy* **25** (1996) 491–507.

[17] Lorenz, M., Methods of measuring the concentration of wealth, *Publ. Amer. Stat. Assoc.* **9** (1905) 209–219.

[18] Luthje, C., Herstatt, C. and von Hippel, E., User-innovators and 'local' information: The case of mountain biking, *Res. Policy* **34** (2005) 951–965.

[19] Meyer, P. B., Episodes of collective invention, Working papers 368, US Bureau of Labor Statistics, Washington DC (2003).

[20] Mockus, A., Fielding, R. T. and Herbsleb, J. D., Two case studies of open source software development: Apache and Mozilla, *ACM Trans. Softw. Eng. Methodol.* **11** (2002) 309–346.

[21] Morrison, P. D., Roberts, J. H. and von Hippel, E., Determinants of user innovation and innovation sharing in a local market, *Manage. Sci.* **46** (2000) 1513–1527.

[22] Ogawa, S., Does sticky information affect the locus of innovation? Evidence from the Japanese convenience-store industry, *Res. Policy* **26** (1998) 777–790.

[23] O'Mahony, S. and Ferraro, F., The emergence of governance in an open source community, *Acad. Manage. J.* **50** (2007) 1079–1106.

[24] Preece, J., Sociability and usability in online communities: Determining and measuring success, *Behav. Inf. Technol.* **20** (2001) 347–356.

[25] Rainer, A. and Gale, S., Evaluating the quality and quantity of data on open source software projects, in *First Int. Conf. Open Source Software*, Genova, Italy (University of Genova, 2005).
[26] Raymond, E. S., *The Cathedral and the Bazaar*: *Musings on Linux and Open Source by an Accidental Revolutionary* (O'Reilly, Sebastopos, 1999).
[27] Schumpeter, J., *Theorie der wirtschaftlichen Entwicklung* (Springer, 1912).
[28] Schumpeter, J. A., *Capitalism, Socialism and Democracy* (Harper, New York, 1942).
[29] Shah, S., Sources and patterns of innovation in a consumer products field: Innovations in sporting equipment, Working Paper #4105, MIT Sloan School of Management (2000).
[30] Smith, M., Invisible crowds in cyberspace: Mapping the social structure of the Usenet, in *Communities in Cyberspace* (Routledge, New York, 1999), pp. 195–219.
[31] Smith, M. and Kollock, P., *Communities in Cyberspace* (Routledge, 1999).
[32] Spaeth, S., von Krogh, G. and Stuermer, M., Enabling knowledge creation through outsiders: Towards a push model of open innovation, *Int. J. Technol. Manage.* **52** (2010) 411–431.
[33] Torvalds, L., The linux edge, *Commun. ACM* **42** (1999) 38–39.
[34] Torvalds, L. and Diamond, D., *Just for Fun: The Story of an Accidental Revolutionary*, 2nd edn. (Collins Business, 2002).
[35] von Hippel, E., *The Sources of Innovation* (Oxford University Press, 1988).
[36] von Hippel, E., Sticky information and the locus of problem solving: Implications for innovation, *Manage. Sci.* **40** (1994) 429–439.
[37] von Hippel, E., Innovation by user communities: Learning from open source software, *Sloan Manage. Rev.* **42** (2001) 82–86.
[38] von Hippel, E., *Democratizing Innovation* (The MIT Press, Cambridge, 2005).
[39] von Hippel, E. and Lakhani, K., How open source software works: 'Free' user-to-user assistance? *Res. Policy* **32** (2003) 923–943.
[40] von Hippel, E. and von Krogh, G., Open source software and the "private-collective" innovation model: Issues for organization science, *Organ. Sci.* **14** (2003) 209–223.
[41] von Krogh, G., Spaeth, S. and Lakhani, K. R., Community, joining, and specialization in open source software innovation: A case study, *Res. Policy* **32** (2003) 1217–1241.
[42] West, J. and O'Mahony, S., Contrasting community building in sponsored and community founded open source projects, in *HICSS '05*: *Proc. 38th Annual Hawaii Int. Conf. System Sciences* (*HICSS'05*) (IEEE Computer Society, Washington DC, 2005), p. 196c.
[43] Wheeler, D., Why open source software/free software (OSS/FS)? Look at the numbers! (2007), http://www.dwheeler.com/oss_fs_why.html.
[44] Williams, R. and Cothrel, J., Four smart ways to run online communities, *Sloan Manage. Rev.* **41** (2000) 81–92.