Measuring the Effect of Social Communications on Individual Working Rhythms: A Case Study of Open Source Software

Qi Xuan*†, Mohammad Gharehyazie*, Premkumar T Devanbu *, Vladimir Filkov*
*Department of Computer Science, University of California, Davis
Davis, California 95616–8562
†Department of Automation, Zhejiang University of Technology
Hangzhou 310023, China
Email: qxuan@ucdavis.edu

***Abstract*—This paper proposes novel quantitative methods to measure the effects of social communications on individual working rhythms by analyzing the communication and code committing records in tens of Open Source Software (OSS) projects. Our methods are based on complex network and time-series analysis. We define the notion of a working rhythm as the average time spent on a commit task and we study the correlation between working rhythm and communication frequency. We build communication networks for code developers, and find that the developers with higher social status, represented by the nodes with larger number of outgoing or incoming links, always have faster working rhythms and thus contribute more per unit time to the projects. We also study the dependency between work (committing) and talk (communication) activities, in particular the effect of their interleaving. We introduce multi-activity time-series and quantitative measures based on activity latencies to evaluate this dependency. Comparison of simulated time-series with the real ones suggests that when work and talk activities are in proximity they may accelerate each other in OSS systems. These findings suggest that frequent communication before and after committing activities is essential for effective software development in distributed systems.**

***Index Terms*—social network; time-series; committing rhythm; open source software; work and talk;**

I. Introduction

Recently, much attention has been paid to social networks [1]–[3], where nodes represent different individuals and links between pairs of nodes mean the corresponding individuals are friends or directly communicate with each other. Generally, there are two reasons that can explain this great interest in networks. First, can conveniently model the topological structure of complex systems, providing a series of structural characteristics [4], [5] that can help differentiate the roles of individuals working on these systems. For example, Albert *et al*. [6] and Holme *et al*. [7] found that the communication efficiencies of networks are more likely to depend on the nodes with larger degrees or betweenness centralities, *i.e*., the network performance decreases quickly once these nodes are removed. Second, it is widely believed that network structure, or more specifically, communications themselves can influence the individual actions, and thus some cooperative behaviors such as synchronization [8]–[10] naturally emerge.

However, such influence is often difficult to measure, and subject to conflicting viewpoints. Consider the increasing interest of developers to contribute to open source software (OSS) projects [11]–[13]. Developers communicate through emails and commit to different files in real time. Since most work on OSS is voluntary, the success of these OSS projects is mainly determined by the committing activities of developers [14], which leads to an important question [15] in this area: whether and how do the communication activities influence the committing activities? For this question, different people may have different answers before such influence can be quantified.

It is often argued that social communication activities have some negative effects on the efficiencies of working activities [16], since both of these activities may compete for the time resources of individuals [17]–[19]. As a result, people always prefer to find ways to reduce "communication overheads". Baldwin and Clark [20] argued that the communication and coordination in large systems can be significantly reduced by adopting proper design rules. On the other hand, in our society, productivity and communication often go hand-in-hand. In OSS projects, it can be argued that once a user finds a bug [21], [22], he/she would want to report them as soon as possible in order to gain some sense of achievement, while once the developers received some (possibly negative) evaluations of their work, they may respond by updating the software right away, in order to preserve their reputation among the social circle of developers. In such a situation, since task-relevant information flows through communication activities, it can be argued that communication activities accelerate committing activities in the OSS projects to some extent.

Then, which is it? Do the communication activities impede committing activities or do they accelerate them? Although a recent study [23] on OSS projects indicates that there are strong correlations between the number of messages sent by an individual and the number of code changes he/she made, this result still cannot answer the above question. Clearly, both of these activities are positively correlated with the time interval over which they take place; however, this result doesn't necessarily imply that one accelerates the other.

978-0-7695-5015-2/12 $26.00 © 2012 IEEE
DOI 10.1109/SocialInformatics.2012.17

In order to answer this question more definitely, here we first provide the definition of working rhythm, or committing rhythm more specifically, for developers. Then, we propose two methods to measure the effects of communication activities on committing rhythms. In particular, the main contributions of this paper include the following two parts:

1) *Macroscopic view*. We build communication networks for code developers from ten years of email records in 31 OSS projects, and utilize the local structural properties in complex network theory, such as outgoing degree and incoming degree, to quantify the social status of developers. We find that the developers with higher social status, represented by the nodes with larger number of outgoing or incoming links, always have faster working rhythms and thus contribute more per unit time to the projects.
2) *Microscopic view*. We introduce multi-activity time-series and propose the definitions of *evaluation* and *response* latencies between the successive committing and incoming communication activities in order to quantitatively measure the dependency between work and talk activities. We introduce a mechanism to generate independent simulated time-series of incoming communication activities which have precisely the same statistical properties as the real ones. By comparing measurements on the simulated time-series with those on the real ones, we find that the committing and communication activities may significantly accelerate each other in OSS systems.

We study how work and talk activities interact; our findings suggest that frequent, interleaving communication around committing activities is essential for effective software development in a distributed setting; but our findings may have broader implications beyond OSS. Many real-world systems can be described by complex networks and individual actions can be modeled using time-series, the methods proposed here can also be used to quantify certain relationships in other areas.

The rest of the paper is organized as follows. In Section II, the communication and committing data obtained from OSS projects are briefly introduced, where communication networks and committing networks are constructed and some basic properties are provided. In Section III, the methodologies, including the definition of committing rhythm and the network and time-series based methods are introduced. In Section IV, the main results are obtained based on the proposed definition and methods. The paper is finally concluded in Section V.

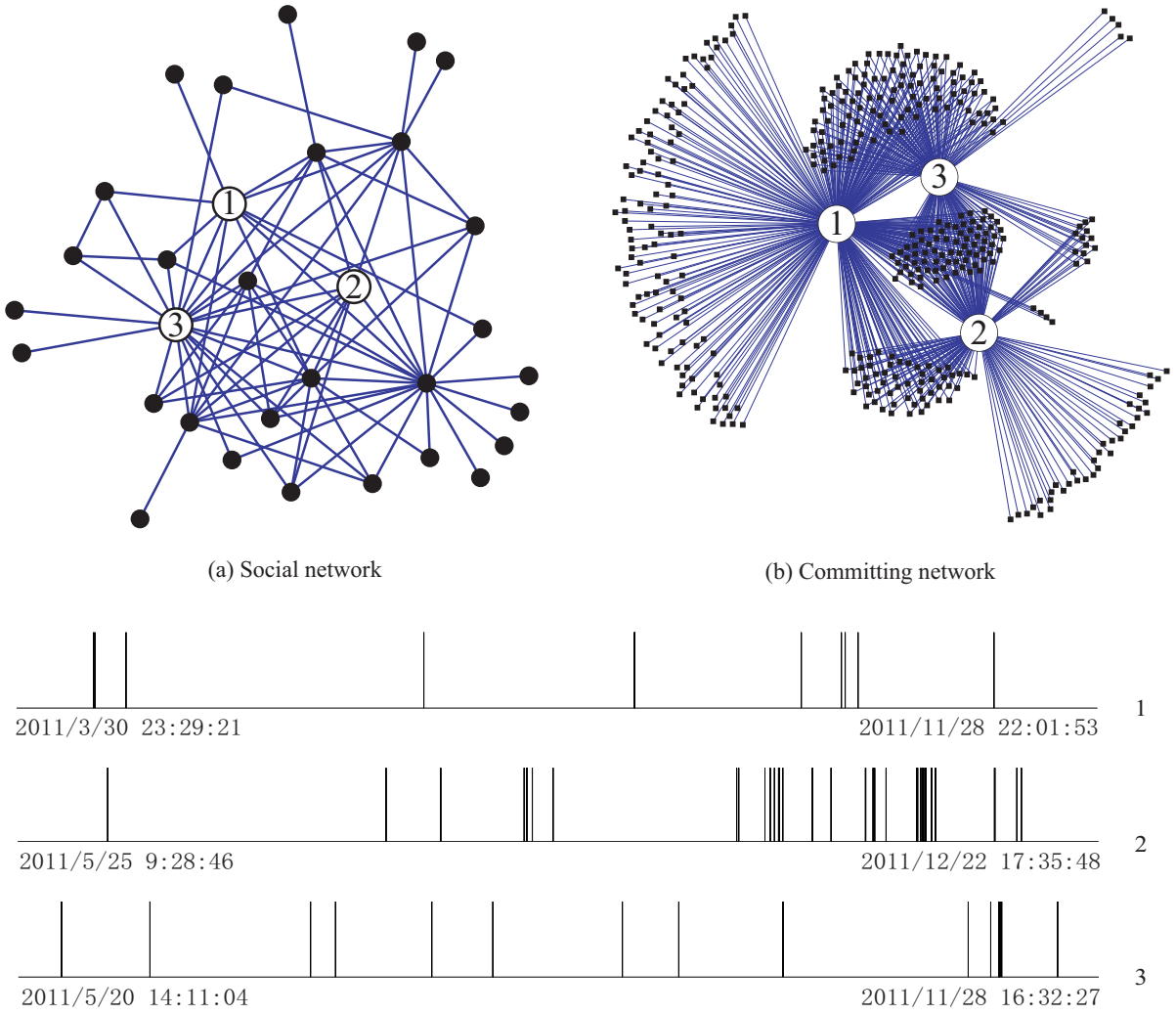
II. Data Description

A total of 31 OSS projects were obtained from the *Apache Software Foundation* on March 24th, 2012. For each project, a communication social network is constructed from online developer mailing lists [24]. These mailing lists are used for communication and coordination among the normal users and developers, where each email has an ID, a sender ID, and a reference ID with date time and body. Here, the reference ID is the ID of the email that this email is in response to. In such a network, the nodes are the people sending messages on the list, if a person P_1 replies to a message from another person P_2, then there is a directed link from the node representing P_1 to that representing P_2. The social networks constructed by this method are directed networks, and each node v_i in a network will have an incoming degree k_i^{in} and an outgoing degree k_i^{out} representing the numbers of directed links from and to this node, respectively. The average outgoing degree of the network is denoted by $\langle k_{out} \rangle$, and the average incoming degree of the network has the exactly same value. Meanwhile, the committing activities of developers on different files in each project are gathered from the corresponding Git repository and can also be considered as a network, where a node represents a developer or a file, and a link between a developer and a file represents that the developer has contributed to the file (*i.e.*, add some codes in this file). The committing networks constructed by this method are bipartite networks, i.e., links only exist from developers to files. A developer v_i^D has a degree d_i^F representing the number of files s/he has committed. Then the average degree $\langle d_F \rangle$ in a project denotes the mean number of files committed by a developer. Note that, in these projects, users can have multiple aliases; these were resolved using a semi-automatic approach devised by Bird *et al.* [23].

Several basic properties, including name of the project, number of users N_T (including the developers), number of developers N_D, number of files N_F, average outgoing de-

TABLE I
Some basic properties of the 31 OSS projects.

Project	N_T	N_D	N_F	$\langle k_{out} \rangle$	$\langle d_F \rangle$
Accumulo	75	5	1622	3.5	833.2
Mahout	552	16	5123	4.4	698.9
Lucene	2148	41	6674	4.2	414.2
Nutch	862	16	3072	3.4	424.3
Derby	1128	35	6563	5.6	660.1
Ode	377	18	11006	3.8	1013.4
Openejb	179	38	43960	5.8	2374.0
Log4php	88	9	1409	2.1	242.0
Wicket	540	24	48045	5.5	3907.3
Log4j	540	19	5519	2.3	472.7
Bookkeeper	32	3	407	3.0	245.0
Xerces2_j	922	33	3732	1.7	347.4
Hive	321	18	7333	3.4	887.2
Axis2_java	3758	76	129978	2.8	3034.1
Hadoop_hdfs	264	25	1153	2.4	171.2
Camel	844	31	36965	3.2	1713.1
Avro	284	12	3021	3.0	387.6
Abdera	196	13	3193	2.7	352.4
Cassandra	397	13	17125	3.7	1534.7
Activemq	2053	29	16788	2.2	946.3
Cxf	443	45	37867	4.2	1726.2
Log4net	297	7	1060	1.4	320.9
Ant	1406	45	11620	3.2	666.5
Empire_db	8	5	2341	0.9	807.6
Axis2_c	608	24	10262	4.4	805.3
Cayenne	170	20	31489	3.8	2629.9
Log4cxx	92	6	2966	1.7	730.0
Harmony	709	25	14898	9.0	636.8
Pluto	265	23	5971	3.1	483.0
Solr	839	19	8534	3.8	655.8
Ivy	68	9	3513	2.3	523.2



(a) Social network

(b) Committing network

(c) Time series of committing activities

Fig. 1. The topological structure of (a) social network and (b) committing network of the project called *bookkeeper*, where the three marked unfilled nodes are the developers who contributed to the project and the filled nodes represent the other users and files in the corresponding networks. (c) The time-series of the committing activities of the three corresponding developers, with the time of their first and last committing activities provided, where the horizontal axis denotes time (in one second) and each vertical line corresponds to an activity of adding codes.

gree $\langle k_{out} \rangle$ in the corresponding communication network, and average committing files $\langle d_F \rangle$, of the 31 OSS projects are presented in TABLE I. By considering all the projects together, there are totally 20465 users, 702 developers and 483,209 files. In particular, the social network structure and the committing network structure of the project called *bookkeeper* are shown in Fig. 1 (a) and (b), respectively, where the three developers are represented by the unfilled nodes and marked by 1, 2, and 3, respectively, while the other users and files are represented by the filled nodes, in the corresponding networks. Meanwhile, the time-series of committing activities of the three corresponding developers are visualized in Fig. 1 (c), where there is a vertical line if the developer added codes at that time.

III. Methodology

A. Definition of committing rhythm

Suppose there are totally $M(M \geq 2)$ activities for an individual at different consecutive time $t_1, t_2, \ldots, t_M$ satisfying $t_1 < t_2 < \ldots < t_M$, there will be $M-1$ inter-activity time buckets, denoted by

$$\Delta t_i = t_{i+1} - t_i, i = 1, 2, \ldots, M-1. \quad (1)$$

Since many human activities are Poisson processes [25] where independent events occur at a constant rate λ, the inter-activity time between two consecutive activities of an individual follows an exponential distribution as follows:

$$P(\Delta t) = \lambda e^{-\lambda \Delta t}. \quad (2)$$

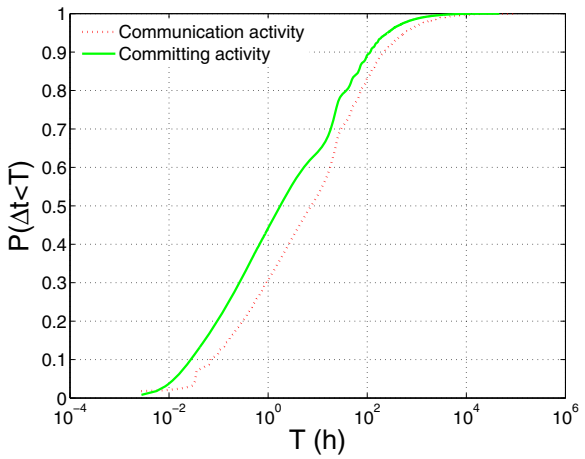


Fig. 2. The cumulative inter-activity time distributions of the communication activities and the committing activities in 31 OSS projects.

Committing activity is no exception either, as shown in Fig. 2, where the near linear functions in semilog coordinates indicate that both inter-activity time distributions of communication and committing activities follow exponential distributions.

Therefore, the average inter-activity time of an individual, calculated by

$$\langle \Delta t \rangle = \frac{\sum_{i=1}^{M-1} \Delta t_i}{M-1} = \frac{t_M - t_1}{M-1}, \tag{3}$$

is close to the inverse of the only parameter λ of the distribution, and thus is a reasonable way to measure its committing rhythm, *i.e.*, smaller average inter-activity time means faster rhythm.

B. Network based method

Generally, communications between individuals can be described by social networks, as shown in Fig. 1 (a), while in the area of social network analysis, the status of an individual is more likely to be characterized by its local structural properties [4], such as degree [5], rather than the number of communication records themselves. In particular, the social status of a developer here is characterized by the incoming and outgoing degree of the corresponding node in the network [26], since in OSS projects, it could be considered that a developer with higher incoming degree is widely trusted in the local society while one with higher outgoing degree has a better sense of responsibility for the project. In most cases, these two local structural properties are correlated with each other to a certain extent, *i.e.*, an individual with higher responsibility for the project is always widely trusted by others in the corresponding local society.

In order to study such social responsibility on committing rhythms of developers, it is intuitive to divide the developers into different groups by their incoming or outgoing degrees. Since there are only 702 developers, here the developers are only divided into two groups in order to make sure that the final results have statistical meaning. In particular, there are two cases as follows:

- *Case I*: The developers are divided into two groups according to their incoming degrees, *i.e.*, the developers with incoming degree $k_{in} \leq 50$ and the developers with incoming degree $k_{in} > 50$.
- *Case II*: The developers are divided into two groups according to their outgoing degrees, *i.e.*, the developers with outgoing degree $k_{out} \leq 50$ and the developers with outgoing degree $k_{out} > 50$.

Note that here the degree threshold value to divide the developers into two groups is set to 50, which is just because, in such a case, there are similar number of activities in the two groups, although, in fact, the results almost keep the same for different degree threshold values. Then, all the inter-activity time buckets of the developers in the same group are put together and is compared with those of the developers in the other group. If committing activities are driven by social responsibility, it is reasonable to find that, statistically, the developers with larger incoming or outgoing degrees have faster committing rhythms. Otherwise, if communication and committing activities compete the time resources of developers, the opposite phenomenon may be observed.

C. Time-series based method

At a fine-grained level, the interaction between communication and committing activities can be directly measured by comparing their time-series. In fact, there is an evaluation-response mechanism in many OSS projects. That is, once a developer $\mathcal{D}$ submits a section of codes at time t_1, denoted by action A_1, s/he may receive several evaluations (such as code reviews or problem reports) from other users (including other developers) at time $t_2, t_3, \ldots, t_k$ via email. $\mathcal{D}$ may respond to these evaluations by adding new section of codes at time t_{k+1}, denoted by action A_2. Suppose that actions A_1 and A_2 are successive, *i.e.*, $\mathcal{D}$ does not add any code in the time interval (t_1, t_2), and we have that $t_1 < t_2 \leq t_3 \leq, \ldots, \leq t_k < t_{k+1}$, the evaluation latency and the response latency then are defined by

$$\tau_E = t_2 - t_1, \tag{4}$$

$$\tau_R = t_{k+1} - t_k, \tag{5}$$

respectively, as shown in Fig. 3.

Using the actual incoming communication activities of each individual, but assuming that there is no co-ordination between communication and committing activities, it is possible to generate a series of simulated incoming communication activities for each individual independently. In particular, for each developer v_i, the precise method to generate such a simulated time-series of incoming communication activities is as follows:

1) Suppose there are totally $U_i (U_i \geq 2)$ incoming communication activities for v_i at different times, denoted by $t_1, t_2, \ldots, t_{U_i}$, respectively, as shown in Fig. 4 (a). Then, $U_i - 1$ ordered time intervals, denoted by

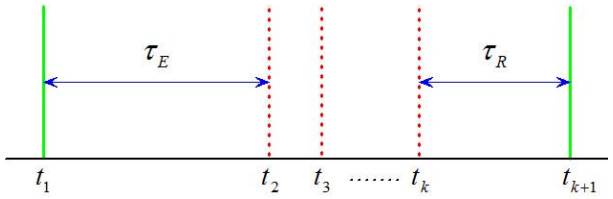


Fig. 3. The definitions of the evaluation latency τ_E and the response latency τ_R for a developer. The horizontal axis denotes time. Each solid vertical line corresponds to a commiting activity of adding codes at time t_1 and t_k, while each dotted vertical line corresponds its incoming communication activities, *i.e.*, he/she received emails at time $t_2, t_3, \ldots, t_k$ satisfying $t_1 < t_2 \leq t_3 \leq , \ldots, \leq t_k < t_{k+1}$. Then the evaluation latency τ_E and the response latency τ_R are defined by Eqs. (4) and (5), respectively.

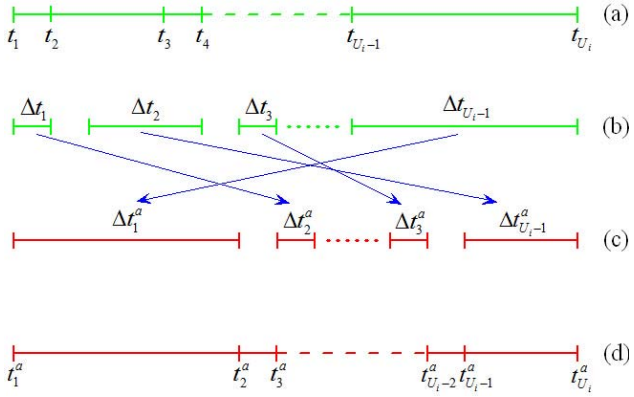


Fig. 4. The steps to generate a simulated time-series of incoming communication activities for each developer.

$\Delta t_1, \Delta t_2, \ldots, \Delta t_{U_i-1}$, respectively, can be obtained by $\Delta t_i = t_{i+1} - t_i$, as shown in Fig. 4 (b).

2) Randomly rearrange the $U_i - 1$ time intervals and get a new sequence of time intervals, denoted by $\Delta t_1^a, \Delta t_2^a, \ldots, \Delta t_{U_i-1}^a$, respectively, as shown in Fig. 4 (c). This essentially generates random orderings of "idling periods" for the developer, but ensures that his "idling" periods are exactly the same as actually observed.
3) Weld these new ordered time intervals one by one, as shown in Fig. 4 (d), and then get a new time-series $t_1^a, t_2^a, \ldots, t_{U_i}^a$, satisfying that

$$\begin{cases} t_i^a = t_i, i = 1, \\ t_i^a = t_{i-1}^a + \Delta t_{i-1}^a, i \geq 2. \end{cases} \quad (6)$$

Note that the simulated time-series of incoming communication activities generated by this mechanism preserves similar statistical properties as the real one, *viz.,* the same distribution of inter-activity interval time.

By replacing the real time-series of communication activities by this simulated time-series and comparing with the real time-series of committing activities, we can get two series of simulated evaluation and response latencies which are still calculated by Eqs. (4) and (5), respectively. Generally, for the real communication and committing activities, there are several possible relationships between them, which are listed as follows and can be characterized by comparing the real and simulated evaluation and response latencies.

1) *They are independent from each other.* If this is the case, the distributions of real evaluation and response latencies will be statistically indistinguishable from the simulated ones.
2) *They delay each other*. In reality, both communication and committing activities do take time, *i.e.*, they compete for the time resources of developers. In addition, developers may spend time responding to bug reports, questions, challenges *etc* in the emails; this might delay committing activities to a certain extent. If this is the case, the actual evaluation and response latencies will be statistically longer than the simulated ones.
3) *They accelerate each other*. As discussed earlier, the desire to enhance and or maintain reputations may incentivize users to respond more quickly to tasks that relate to received email correspondence. Bug finders may accelerate bug reports to gain recognition. Likewise, programmers may be hastening to respond to bug reports, or design/coding critiques, in order to maintain their peer-reputation. If this is the case, statistically, the real evaluation and response latencies will be relatively shorter than the simulated ones.

Certainly, there are also other cases where only one kind of activities influence the other. For example, only incoming communication activities accelerate committing activities. In this case, it can be expected that, statistically, the real response latencies will be relatively shorter than while the real evaluation latencies will be close to the simulated ones. And other cases can be confirmed by the similar manner.

IV. Results

A. Higher social status indicate faster committing rhythms

Here, we use network-based measures. As one can see, the human activity rhythms can be statistically analyzed using the cumulative inter-activity time distribution, which allows us to compare the committing rhythms of developers belonging to different groups. The cumulative inter-activity time distributions of the committing activities for different groups of developers characterized by their incoming degrees (*Case I*) and outgoing degrees (*Case II*) are shown in Fig. 5 (a) and (b), respectively. By comparison, there are more short inter-activity time intervals for the developers with higher social status, *i.e.*, larger incoming or outgoing degrees, which indicates that, statistically, the developers with higher social status may have faster committing rhythms. More interestingly, it is found that the difference of cumulative inter-activity time distribution between the two distinct groups of developers is mainly introduced when $\Delta t > 1$ (h) while there seems no difference between them when $\Delta t \leq 1$ (h), which suggests that social status can only influence the long-term, but have little effects on the short-term committing rhythms. This is reasonable because the short-term committing rhythm is more

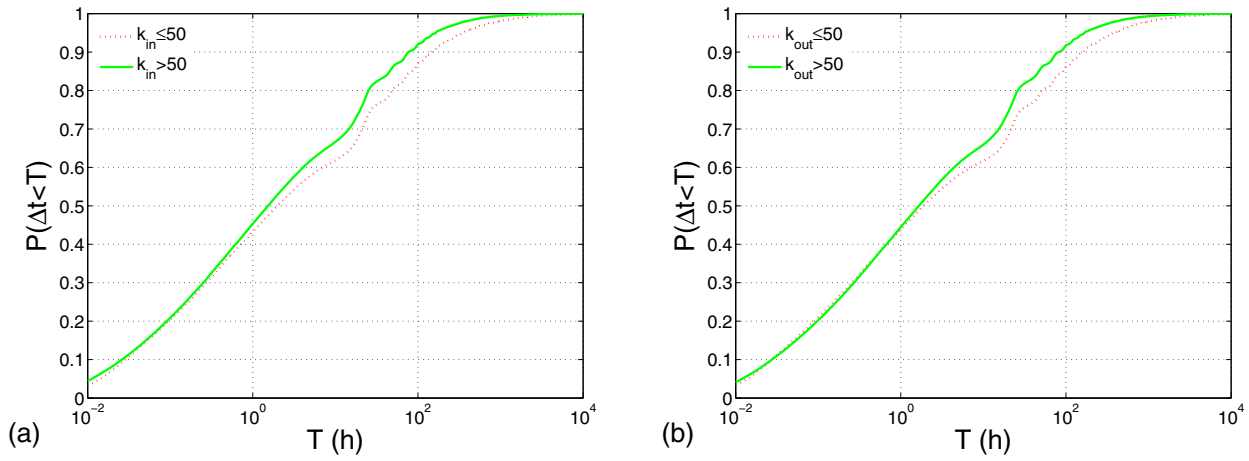


Fig. 5. The cumulative inter-activity time distributions of the committing activities for different groups of developers characterized by their (a) incoming degrees and (b) outgoing degrees. As one can see, here the difference of cumulative inter-activity time distribution between two distinct groups of developers is mainly introduced when $\Delta t > 1$ (h).

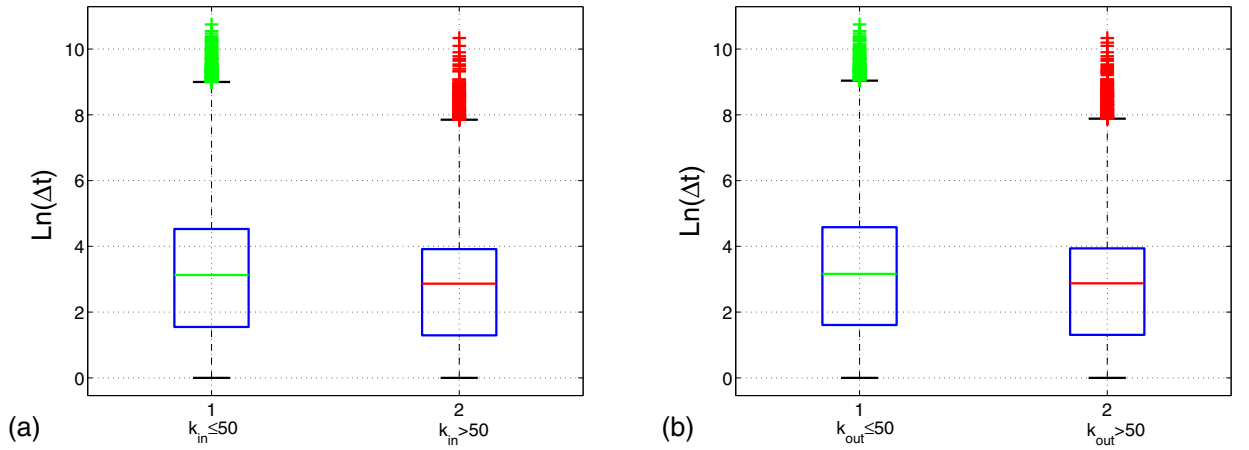


Fig. 6. The corresponding box-and-whisker diagrams of the inter-activity time for different groups of developers characterized by their (a) incoming degrees and (b) outgoing degrees. Here, only the inter-activity time with length longer than one hour is considered, and the y-axes are logarithmically transformed in order to present the difference of the committing activities between different groups of developers more clearly.

a reflection of the developer's programming habit, *i.e.*, saving codes in real time when he/she commits files, which is for sure seldom influenced by social factors. Just considering the inter-activity time with length longer than one hour, the differences of committing rhythms between the developers belonging to different groups can be presented by the box-and-whisker diagrams more clearly, as shown in Fig. 6 (a) and (b), respectively.

In order to provide more credible results, simple T-tests are implemented for both cases and the statistics including the average inter-activity time length, T-value, and significance are recorded in TABLE II, where k means k_{in} for *Case I* and k_{out} for *Case II*. Generally, the average inter-activity time length of the developers with larger incoming (or outgoing) degrees is much smaller than that of the developers with smaller incoming (or outgoing) degrees, and the differences in both cases are quite significant, with the relatively large T-values 20.4 and 22.3, respectively. More specifically, denote by Δt_L^{in} (or Δt_L^{out}) and Δt_S^{in} (or Δt_S^{out}) the average inter-activity time lengths of the developers with incoming (or outgoing) degree larger and smaller than 50, respectively. Then, on average, the differences of committing rhythm between the different groups of developers in two cases can be qualified by

$$\Delta t_S^{in} - \Delta t_L^{in} = 57.5(h), \quad (7)$$

$$\Delta t_S^{out} - \Delta t_L^{out} = 62.6(h), \quad (8)$$

respectively. These differences will be further enlarged if only the inter-activity time buckets with length longer than one hour are considered. For comparison, extra T-tests are also implemented for the two cases in this situation, and the

TABLE II
T-TESTS FOR THE DIFFERENCES OF INTER-ACTIVITY TIME LENGTHS BETWEEN DIFFERENT GROUPS OF DEVELOPERS IN *Case I* AND *Case II*.

T-test	$k \leq 50$	$k > 50$	T-value	Significance
Case I	103.0 (h)	45.5 (h)	20.4	$p < 10^{-6}$
Case II	111.2 (h)	48.5 (h)	22.3	$p < 10^{-6}$

TABLE III
T-TESTS FOR THE DIFFERENCES OF INTER-ACTIVITY TIME LENGTH LONGER THAN ONE HOUR BETWEEN DIFFERENT GROUPS OF DEVELOPERS IN *Case I* AND *Case II*.

T-test	$k \leq 50$	$k > 50$	T-value	Significance
Case I	181.4 (h)	83.0 (h)	20.5	$p < 10^{-6}$
Case II	198.0 (h)	87.2 (h)	22.0	$p < 10^{-6}$

statistics are recorded in TABLE III, where one can see that the gaps are almost doubled when only considering long-term rhythms.

Moreover, From Eqs. (7) and (8), one can see that, by comparison, the social status characterized by outgoing degrees have slightly more remarkable effects than those characterized by incoming degrees on committing rhythms of developers, although these two characters are strongly correlated with each other. This phenomenon suggests that the developments of these OSS projects may be more likely determined by the strong responsibility of the developers.

B. Communication and committing activities accelerate each other

Here, we use a time-series based method. The box-and-whisker diagrams for real and simulated evaluation latencies and response latencies of all the developers are shown in Fig. 7 (a) and (b), respectively, where one can see that both the real evaluation and response latencies are shorter than simulated ones. Here, the y-axes are log-transformed in order to present the differences of evaluation and response latencies between real and simulated cases more clearly. This phenomenon seems to suggest that these two kinds of activities may accelerate each other in reality. However, this result can be claimed only when the differences between the real evaluation and response latencies and the simulated ones are statistically significant.

The statistics of the T-tests are shown in TABLE IV. It is found that the average real response and evaluation latencies equal to 58.5 (h) and 57.2 (h), which are much shorter than the average simulated evaluation and response latencies that equal to 89.9 (h) and 97.0, respectively. The T-tests show that the differences are significant with relatively large T-values 8.21 and 9.92. According to this result and that obtained by the network based method, it is reasonable to say that, statistically, the communication activities can accelerate committing activities in reality. Note that, recent studies on human activities suggest that real distributions of inter-activity time may have relatively heavy tails [27]–[29], *i.e.*, there may be activities separated by long periods of inactivity, which may result in more extremely long evaluation or response latencies in the real situation than in the simulated situation, as shown in Fig. 7 (a). However, since the numbers of these extremely long latencies are very small, they will hardly influence the results presented here.

TABLE IV
T-TESTS FOR THE DIFFERENCES BETWEEN SIMULATED AND REAL EVALUATION AND RESPONSE LATENCIES.

T-test	simulated	Real	T-value	Significance
Evaluation	89.9 (h)	58.5 (h)	8.21	$p < 10^{-6}$
Response	97.0 (h)	57.2 (h)	9.92	$p < 10^{-6}$

V. CONCLUSION AND DISCUSSION

In this paper, the network and time-series based methods are proposed to quantify the influence of social communications on working rhythms by analyzing the communication and committing data of 31 OSS projects in about 10 years, where some new definitions, such as evaluation and response latencies, and a mechanism to generate simulated communication time-series are introduced. Based on these methods, it is found that the developers with higher social status always have relatively shorter average inter-activity time and the average real evaluation and response latencies are also shorter than the average simulated ones, which suggests that social communications may accelerate committing rhythms of developers in reality. Since positive correlation was revealed between committing rhythms and the length of added codes, it can also be considered that social communications have positive effects on the direct contributions of developers. These findings can help researchers better understand the evolution mechanism of OSS systems, and then further help to design more efficient software engineering groups.

In the future, this work can be further expanded in the following several ways. First, the network and time-series methods can be used together in order to reveal whether the individuals response at different rhythms for the evaluations from others of different social status. This issue is important because it involves the efficiency and fairness and thus may determine the success of focused systems to a certain extent. Second, other microscopic multi-activities patterns need to be revealed, because more patterns will definitely provide more information about the interaction between communication and committing activities. Finally, based on these metrics, the co-evolution between different activities can be modeled.

ACKNOWLEDGMENT

The authors would like to thank all the members in our research group in the Department of Computer Science, University of California in Davis, for the valuable discussion about the ideas and technical details presented in this paper. All authors gratefully acknowledge support from the Air Force Office of Scientific Research, award FA955-11-1-0246. QX acknowledges support from the National Natural Science Foundation of China (Grant No. 61004097, 61273212).

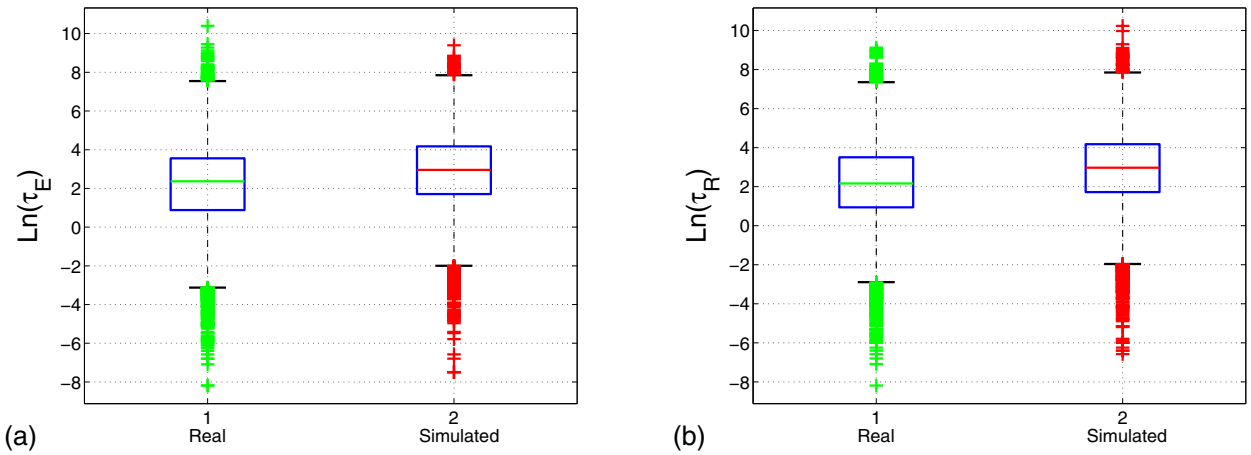


Fig. 7. The box-and-whisker diagrams for real and simulated (a) evaluation latencies and (b) response latencies. Here, the y-axes are logarithmically transformed in order to present the differences of latencies between real and simulated cases more clearly.

REFERENCES

[1] J. Scott, "Social network analysis", *Sociology*, vol. 22, no. 1, pp. 109-127, 1988.

[2] M. E. J. Newman, S. Forrest, and J. Balthrop, "Email networks and the spread of computer viruses", *Physical Review E*, vol. 66, no. 3, pp. 035101(R), 2002.

[3] Q. Xuan, F. Du, and T.-J. Wu, "Empirical analysis of Internet telephone network: From user ID to phone", *Chaos*, vol. 19, no. 2, pp. 023101, 2009.

[4] L. D. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas, "Characterization of complex networks: A survey of measurements", *Advances in Physics*, vol. 56, no. 1, pp. 167-242, 2007.

[5] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: Structure and dynamics", *Physics Reports*, vol. 424, no. 4-5, pp. 175-308, 2006.

[6] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks", *Nature*, vol. 406, no. 6794, pp. 378-382, 2000.

[7] P. Holme and B. J. Kim, "Attack vulnerability of complex networks", *Physical Review E*, vol. 65, no. 5, pp. 056109, 2002.

[8] R. E. Mirollo and S. H. Strogatz, "Synchronization of Pulse-Coupled Biological Oscillators", *SIAM Journal on Applied Mathematics*, vol. 50, no. 6, pp. 1645-1662, 1990.

[9] A. Arenas, A. D.-Guilera, J. Kurths, Y. Moreno, and C. Zhoug, "Synchronization in complex networks", *Physics Reports*, vol. 469, no. 3, pp.93-153, 2008.

[10] W. Yu, G. Chen, and J. Lü, "On pinning synchronization of complex dynamical networks", *Automatica*, vol. 45, no. 2, pp. 429-435, 2009.

[11] C. P. Ayala, D. S. Cruzes, O. Hauge, and R. Conradi, "Five facts on the adoption of open source software", *IEEE Software*, vol. 28, no. 2, pp. 95-99, 2011.

[12] S. K. Sowe, I. Stamelos, and L. Angelis, "Understanding knowledge sharing activities in free/open source software projects: An empirical study", *Journal of Systems and Software*, vol. 81, no. 3, pp. 431-446, 2008.

[13] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla", *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309-346, 2002.

[14] R. Sen, S. S. Singh, and S. Borle, "Open source software success: Measures and analysis", *Decision Support Systems*, vol. 52, no. 2, pp. 364-372, 2012.

[15] D. S. Pattison, C. A. Bird, and P. T. Devanbu, "Talk and work: A preliminary report", *Proceedings of the 2008 International Working Conference on Mining Software Repositories*, pp. 113-116, Leipzig, Germany, 2008.

[16] R. S. Mano and G. S. Mesch, "E-mail characteristics, workperformance and distress", *Computers in Human Behavior*, vol. 26, no. 1, pp. 61-69, 2010.

[17] D. W. Arnesen and W. L. Weis, "Developing an effective company policy for employee Internet and email use", *Journal of Organizational Culture, Communications and Conflict*, vol. 11, no. 2, pp. 53-67, 2007.

[18] D. Zelikovich, "The negative effect of e-mails at work", *Review of International Comparative Management*, vol. 12, no. 3, pp. 575-585, 2011.

[19] L. M.-Carter and T. W. Jackson, "Effects of e-mail addiction and interruptions on employees", *Journal of Systems and Information Technology*, vol. 14, no. 1, pp. 82-94, 2012.

[20] C. Baldwin and K. Clark, *Design Rules*. Cambridge, MA: MIT Press, 2000.

[21] T. Gyimóthy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction", *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 897-910, 2005.

[22] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, "Have Things Changed Now?-An empirical study of bug characteristics in modern open source software", *Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability*, pp. 25-33, San Jose, USA, 2006.

[23] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks", *Proceedings of the 2006 International Working Conference on Mining Software Repositories*, pp. 137-143, Shanghai, China, 2006.

[24] http://mail-archives.apache.org/mod_mbox/

[25] F. A. Haight, *Handbook of the Poisson Distribution*. New York: Wiley, 1967.

[26] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Open borders? Immigration in open source projects", *Proceedings of the Fourth International Workshop on Mining Software Repositories*, pp. 1-6, Washington, DC, USA, 2007.

[27] A. L. Barabási, "The origin of bursts and heavy tails in human dynamics", *Nature*, vol. 435, no. 7039, pp. 207-211, 2005.

[28] R. D. Malmgren, D. B. Stouffer, A. E. Motter, and L. A. N. Amaral, "A Poissonian explanation for heavy tails in e-mail communication", *Proceedings of the National Academy of Sciences U. S. A*, vol. 105, no. 47, pp. 18153-18158, 2008.

[29] Y. Wu, C. Zhou, J. Xiao, J. Kurths, and H. J. Schellnhuber, "Evidence for a bimodal distribution in human communication", *Proceedings of the National Academy of Sciences U. S. A*, vol. 107, no. 44, pp. 18803-18808, 2010.