StackOverflow and GitHub: Associations Between Software Development and Crowdsourced Knowledge

Conference Paper · September 2013

DOI: 10.1109/SocialCom.2013.35

Bogdan Vasilescu
Carnegie Mellon University
50 PUBLICATIONS **842** CITATIONS
SEE PROFILE

Vladimir Filkov
University of California, Davis
84 PUBLICATIONS **2,204** CITATIONS
SEE PROFILE

StackOverflow and GitHub: Associations Between Software Development and Crowdsourced Knowledge

Bogdan Vasilescu
Dept. of Math and Computer Science,
Eindhoven University of Technology,
The Netherlands
b.n.vasilescu@tue.nl

Vladimir Filkov
Computer Science Dept.,
UC Davis,
USA
filkov@cs.ucdavis.edu

Alexander Serebrenik
Dept. of Math and Computer Science,
Eindhoven University of Technology,
The Netherlands
a.serebrenik@tue.nl

***Abstract*—StackOverflow is a popular on-line programming question and answer community providing its participants with rapid access to knowledge and expertise of their peers, especially benefitting coders. Despite the popularity of StackOverflow, its role in the work cycle of open-source developers is yet to be understood: on the one hand, participation in it has the potential to increase the knowledge of individual developers thus improving and speeding up the development process. On the other hand, participation in StackOverflow may interrupt the regular working rhythm of the developer, hence also possibly slow down the development process.**

In this paper we investigate the interplay between StackOverflow activities and the development process, reflected by code changes committed to the largest social coding repository, GitHub. Our study shows that active GitHub committers ask fewer questions and provide more answers than others. Moreover, we observe that active StackOverflow askers distribute their work in a less uniform way than developers that do not ask questions. Finally, we show that despite the interruptions incurred, the StackOverflow activity rate correlates with the code changing activity in GitHub.

I. Introduction

Developers create and maintain software by standing on the shoulders of others [1]: they reuse components and libraries, and go *foraging* on the Web for information that will help them in their tasks [2]. For help with their code, developers often turn to programming question and answer (Q&A) communities, most visible of which is StackOverflow[1] (SO) [3]. To engage its participants to contribute more, StackOverflow employs gamification [4]: questions and answers are voted upon by members of the community; the number of votes is reflected in a person's *reputation* and *badges*; in turn, these can be seen as a measure of one's expertise by potential recruiters [5] and are known to motivate users to contribute more [4]. By *asking questions* on StackOverflow, developers can seek help and advice from their peers, e.g., about their own code snippets or about undocumented technology features [6]. By *answering questions* posed by others, developers can share their knowledge and expertise, help and educate others, or compete in the "game" to achieve higher reputation.

The analogy of StackOverflow as an effective educational institution asserts itself then. The extended effect of education, beyond the immediate edification, is to accelerate or catalyse societal advances. Does StackOverflow have the same effect on software development communities? The connection between developer productivity and their using of StackOverflow is not well-understood. On the one hand, StackOverflow is known to provide good technical solutions [6] and to provide them fast [3], to the extent that closer integration between Q&A websites and modern IDEs is now advocated [7], [8]. On the other hand, as an exponent of social media, using StackOverflow may lead to interruptions impairing the developers' performance [1], especially when gamification is factored in.

In this paper we investigate the interplay between asking and answering questions on StackOverflow and committing changes to open-source GitHub[2] repositories. GitHub is arguably the largest social coding site [9], hosting more than three million software projects maintained by over one million registered developers. The two platforms overlap in a knowledge-sharing ecosystem: GitHub developers can ask for help on StackOverflow to solve their own technical challenges; similarly, they can engage in StackOverflow to satisfy a demand for knowledge of others, perhaps less experienced than themselves. By identifying GitHub users active on StackOverflow and studying their activities on both platforms, we can study if a connection exists between their participation in StackOverflow and their productivity on GitHub. That is,

Goal: Is participation in StackOverflow related to productivity of GitHub developers?

Here, following Adams et al. [10], we look at only one, but representative, facet of developer productivity: the number of commits made by developers in a given time period. Clearly, commits can be of different length and quality, and thus their number is insufficient to quantify the total contribution of a developer to a project, or even to quantify their energy expenditure while doing so. However, it is a reasonable representative, or sample, of the overall activities a developer undertakes while working on a project.

[1]http://stackoverflow.com

[2]http://github.com

As complex relationships are best understood when looked at from different angles and at a range of resolutions, we examine the relationship between StackOverflow and GitHub at three different levels. At the *macro-level*, we look at the time-aggregate (overall) activities over the two platforms (questions, answers and commits) of developers active on both. At this level we would like to identify differences between GitHub contributors in terms of their involvement in StackOverflow, and understand whether some groups of developers benefit more from participation in SO than others. Indeed, GitHub users are a mix of novice and professional programmers [11]. While it is known that foraging is common for novices and experts alike [2], their *diets* are different [12], with potentially different impact on their performance. Similarly, different roles can be identified among StackOverflow participants based on the quantity and quality of their questions, answers, and comments [13]. However, for both platforms, such roles are identified using only information about the activities of contributors *within* each platform. We would like to understand how such groups of developers relate *across* platforms, and to which extent activity (expertise) on one platform can be used as a proxy for activity (expertise) on the other. For example, such relations become immediately important when evaluating the reliability of social signals for career advancement [5].

RQ1: (Macro level) How are the overall activity levels of developers related across the two platforms? E.g., do active GitHub committers ask more or fewer questions on SO? Are more active answerers also committing more?

At the *intermediate level*, we attempt to capture the distribution of work units over time vis-a-vis people's participation on StackOverflow. We are interested in understanding how developers distribute their time over commits, given their amount of Q&A activity. Do different groups of developers exhibit different working rhythms (e.g., do active SO answerers, presumably more experienced, work differently than less active ones)? Bursts of intense commit activity followed by long periods of inactivity would suggest more focused attention at any given time, while a more egalitarian distribution of inter-commit time intervals would suggest a more steady but less focused working rhythm, where attention at any given time is divided between the two platforms. Working rhythms of developers are known to influence the quality of the software [14]. Specifically, we answer the following research question:

RQ2: (Intermediate level) Are the commit (work) patterns on GitHub of developers more active on SO different than the commit patterns of those developers less active on StackOverflow?

Lastly, at the *micro-level*, we associate GitHub commits and StackOverflow questions and answers over time. We wish to understand whether activities in the two platforms show signs of coordination, i.e., whether the rate of asking or answering questions on SO is related to the rate of commit activities in GitHub. Specifically, we ask:

RQ3: (Micro level) Is there a functional interaction, or coordination between commit and question/answer activities? I.e., when commits are close to Q&A in time, are they more frequent? How about vice-versa?

The remainder of this paper is organised as follows. After reviewing the related work in Section II, in Section III we discuss how the data has been obtained and prepared. We distinguish between the macroscopic (e.g., are heavy GitHub committers also heavy StackOverflow users?), intermediate (e.g., how do StackOverflow activities affect the working rhythm of the developers?) and microscopic (e.g., are StackOverflow activities occurring in lockstep with GitHub commits?) views and discuss them in Sections IV, V and VI, respectively. Finally, we summarise our contribution and sketch directions for further research in Section VII.

II. Related work

The abundance of information to which developers are exposed via social media is changing the way they collaborate, communicate, and learn [1], [5], [11], thus ultimately impacting the way they write software. Specifically, StackOverflow is known to cover numerous software engineering topics and attract numerous software developers. Popularity of StackOverflow among software developers has lead to increased interest from the research community as well [15]. However, the productivity implications of StackOverflow remain unclear.

On the one hand, it can be argued that participating in StackOverflow leads to *interruptions that could impair a developer's performance* [1]. Indeed, Bacchelli et al. [7] and Cordeiro et al. [8] argue that the current lack of integration between Q&A websites and modern IDEs forces developers to interrupt their flow and change context every time they need to deal with them, thus delaying their activity. Xuan et al. [16] argue that social communication activities (such as asking or answering StackOverflow questions) may delay programming activities, since both of these activities compete for the time resources of developers. Indeed, it is well known that "a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it" [17].

On the other hand, it can be argued that participating in StackOverflow *speeds up development activities* as quick solutions to technical challenges are provided, thus saving the developers precious time. Mamykina et al. [3] show that most StackOverflow questions are answered in a median time of 11 minutes. Parnin et al. [6] argue that StackOverflow is a better source of API documentation, while Brandt et al. [2] propound that by relying on information and source code fragments from the Web, developers more effectively distribute their cognition, allowing them to devote more energy to higher-level tasks.

Moreover, both StackOverflow participation and software development in public GitHub repositories can be seen as social activities. StackOverflow evaluates the participant's contribution in terms of reputation points and badges, that allow participants to access new features and gain more control on others' postings. In this way, StackOverflow encourages the participants to ask "good" questions and to give "good" answers [5]. Similarly, heavy GitHub users are aware of being watched by their peers, and this awareness influences how they behave and construct their actions, for example, by making changes less frequently [11].

III. Data preparation: StackOverflow and GitHub

To study the interplay between communication and code commit activities we integrate information extracted from two sources: StackOverflow and GitHub. In this section we discuss how the data has been obtained and merged. All data used in this study as well as the tools developed are available for replication on http://www.win.tue.nl/mdse/stackoverflow/.

A. Extraction

All public data in StackOverflow, including the list of members and the history of their activity, can be downloaded in XML format as part of the Stack Exchange data dump[3]. Data dumps are released every three months under the Creative Commons license. Here we explore the one released in August 2012, containing information about 1,295,622 registered users since July 2008 until August 2012.

The GitHub data comes from GHTorrent [18], a service that gathers event streams and data from GitHub and provides that data back to the community in the form of incremental MongoDB data dumps[4]. The GitHub dataset contains information about 397,348 users and 10,323,714 commits, most from the July 2011 to April 2012 period.

B. Preprocessing

Git commits contain information about both the *author* (the person who originally changed the code) and the *committer* (the persons who last applied the change), each with their own timestamp. The two are not necessarily one and the same person (e.g., they can differ when someone *rebases*[5] or *cherry picks*[6] a commit). In this paper we consider only the commits which record the same person as both author and committer (97.8% of the commits in our dataset), and record the date at which a commit was *authored* (rather than *committed*).

In addition, git allows commit metadata, including the authorship date, to be overwritten. For instance, we conjecture that commits with the 1969-12-31 or 2050-07-18 timestamps underwent such a history rewriting process. Therefore, we restrict our study to the period July 2011 to April 2012 (depicted in Figure 1) which contains the bulk of the commits in GHTorrent (approximately 99%).

Finally, the GHTorrent authors acknowledged that bugs in their extraction process led to some duplicate commits being recorded. We ignore duplicate commits, i.e., commits authored by the same person and having the same timestamp.

[3]http://www.clearbits.net/torrents/2076-aug-2012

[4]Accessible via https://github.com/gousiosg/github-mirror

[5]http://www.kernel.org/pub/software/scm/git/docs/git-rebase.html

[6]http://www.kernel.org/pub/software/scm/git/docs/git-cherry-pick.html

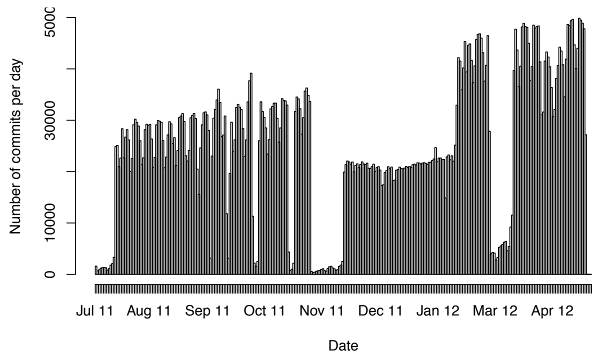


Fig. 1: Number of commits per day in the GitHub dataset between July 2011 and April 2012.

TABLE I: Sizes of the original and intersection datasets.

Dataset	Number of users (in intersection; active)
GitHub	397,348 (23.6%; 11.8%)
StackOverflow	1,295,622 (7.2%; 3.6%)
Intersection	93,771
Intersection (active)	46,967

C. Intersecting the two datasets

A key step in our process is merging the GitHub and StackOverflow datasets, i.e., identifying those contributors which are active on both platforms. Merging aliases used by the same person in different software repositories is a well-known problem [19]–[21]. For example, Bird et al. [19] try to match full names or email addresses shared by different aliases, and use heuristics to "guess" email prefixes based on combinations of name parts (e.g., *jsmith* and *John Smith*). Kouters et al. [21] use Latent Semantic Analysis (LSA), a popular information retrieval technique, and report better results in presence of very noisy data. However, all existing approaches are known to produce false positives and false negatives [20].

To limit the number of false positives[7], we follow a more conservative approach and make use of email addresses. In the GitHub dataset email addresses are present. In the StackOverflow dataset email addresses are obscured, but their MD5 hashes are available. Therefore, we decide to merge (i.e., link) a GitHub and a StackOverflow user if the computed MD5 hash of the former's email address is identical to the MD5 email hash of the latter. Table I presents basic statistics about the two datasets, before and after intersecting. More advanced approaches to identity merging, e.g., that also take into account names or email prefixes [25], are considered as future work.

As a result of this process, approximately one quarter of the GitHub users (23.6%, or 93,771) are linked to StackOverflow. However, it is possible that not all users in the GitHub & StackOverflow intersection have authored at least one commit on GitHub between July 2011 and April 2012 (see the discussion above). Similarly, it is possible that not

[7]The accuracy of the identity merging algorithm cannot be estimated in the absence of an "oracle" (i.e., the absolute truth) against which to compare the results. Such an oracle does not exist for the two datasets.

all users in the GitHub & StackOverflow intersection have actively participated in StackOverflow by asking or answering during the same period. Therefore, we further require users to have been active on both platforms, hence we filter out those users that neither authored any commits, nor asked or answered any question between July 2011 and April 2012. This further reduces the size of the intersection dataset to 46,967 users (or 11.8% of the GitHub dataset).

IV. Macroscopic view

To study how GitHub committing reflects StackOverflow activities we start by taking the macroscopic view and studying distributions of the number of events of each type (C for commit, Q for question, A for answer). For each ordered pair of event types (e.g., (C, Q)), with the data sorted along one of the dimensions (e.g., C), we split the other dimension (e.g., Q) into multiple groups and compare the resulting distributions. We performed experiments with our groups being quartiles and deciles but report only the results obtained for quartiles, since splitting into deciles yielded similar results.

This "split-and-compare" approach was chosen over the traditional statistical approaches of comparing correlation, like the correlation coefficient and regression modeling, because the latter are only capable of detecting monotonic relations (e.g., "high number of commits corresponds to high number of questions and low number of commits corresponds to low number of questions", or "high number of commits corresponds to low number of answers and vice versa"). The "split-and-compare" approach, when used with an appropriate statistical testing procedure, as we do below, can *also* detect non-monotonic relationships (e.g., "both the low and the high number of commits correspond to high number of questions, while if the number of commits is neither too high nor too low, the number of commits is low").

A. Comparing multiple distributions

Traditionally, comparison of multiple groups follows a two-step approach: first, a global null hypothesis is tested, and then multiple comparisons are used to test sub-hypotheses pertaining to each pair of groups. The first step is commonly carried out by means of ANOVA or its non-parametric counterpart, the Kruskal-Wallis one-way analysis of variance by ranks. The second step uses the t-test or the rank-based Wilcoxon-Mann-Whitney test, with Bonferroni correction. Unfortunately, the global test null hypothesis may be rejected while none of the sub-hypotheses are rejected, or vice versa [22]. Moreover, simulation studies suggest that the Wilcoxon-Mann-Whitney test is not robust to unequal population variances, especially in the case of unequal sample sizes [23]. Therefore, one-step approaches are preferred: these should produce confidence intervals which always lead to the same test decisions as the multiple comparisons.

To this end, we employ the recently-proposed multiple contrast test procedure $\widetilde{\mathbf{T}}$ [24] using the traditional 5% family-wise error rate. $\widetilde{\mathbf{T}}$ is robust against unequal population variances and is applicable to different types of contrasts, including comparisons of all pairs of distributions, the so called *Tukey-type contrasts*. For Tukey-type contrasts, we summarise the results of $\widetilde{\mathbf{T}}$ by means of $\widetilde{\mathbf{T}}$-graphs [25]. In such a directed acyclic graph, nodes correspond to the different groups being compared, and edges to the results of the pairwise comparisons. There is an edge from A to B if A tends to have higher values for a given metric than B (i.e., for the comparison A–B, $\widetilde{\mathbf{T}}$ reports $p < 0.05$). Since $\widetilde{\mathbf{T}}$ respects transitivity, in a $\widetilde{\mathbf{T}}$-graph we omit direct edges between A and B if there is a path from A to B passing through at least one other node. Consider the example $\widetilde{\mathbf{T}}$-graph from Figure 2, summarising the results of the $\widetilde{\mathbf{T}}$ procedure applied to four groups of values A, B, C and D: D tends to have higher values than both B and C, but lower than A; A tends to have higher values than all other groups (D directly, B and C transitively).

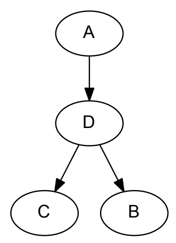


Fig. 2: Example $\widetilde{\mathbf{T}}$-graph.

B. Results

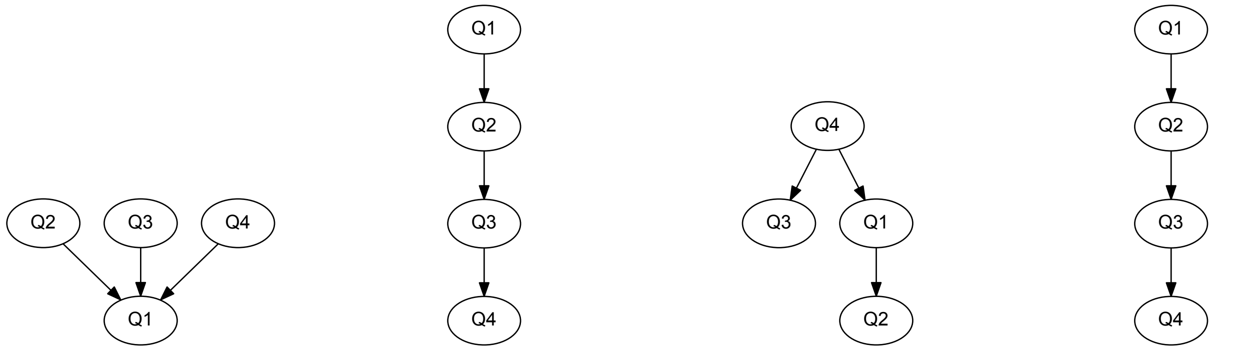
Are heavy committers also heavy question askers?: With the data sorted along the C dimension in decreasing order, we split the Q dimension into quartiles and compare the resulting groups pairwise. The results (Figure 3a) reveal that the most active 25% of the committers (Q1) ask fewer questions on StackOverflow than any of the other quartiles, but Q2, Q3 and Q4 are virtually indistinguishable from each other in terms of their Q activity. This suggests that active GitHub committers are experienced developers that do not need much technical advice: they perform numerous commits without asking much for help on StackOverflow.

> Active GitHub committers ask fewer questions on StackOverflow than others.

Are heavy committers also heavy answer givers?: With the data sorted along the C dimension in decreasing order, we split the A dimension into quartiles and compare the resulting groups pairwise. The results (Figure 3b) reveal a perfect ordering: more active committers provide more answers (e.g., Q2 developers answer more questions than any of Q3 or Q4, but fewer than Q1). This suggests that GitHub activity can be seen as a proxy for one's willingness to answer technical questions on StackOverflow, or one's level of expertise. When further put into the context of gamification, this finding suggests that top users on StackOverflow are "superstars" rather than "slackers" [26]: they don't just compete for reputation and badges, but are actually active software developers.

> More active GitHub committers provide more answers on StackOverflow.

Are heavy question askers also heavy committers?: With the data sorted along the Q dimension in decreasing order, we split the C dimension into quartiles and compare the resulting groups pairwise. The results (Figure 3c) reveal a non-monotonic relation between Q and C that could not have been revealed by traditional correlation techniques. One the one hand, the least active askers (the Q4 users with the fewest questions asked) author more commits than any of the others.



(a) Q1 committers ask fewer questions on SO than any of the others.

(b) More active committers answer more questions on SO.

(c) Q4 askers commit more than any of the others. Q1 askers commit more than Q2 ones.

(d) More active answerers author more commits on GitHub.

Fig. 3: Macroscopic view of activity levels of users active on both GitHub and StackOverflow [July 2011–April 2012].

This observation is in line with its complement above: the most active committers ask the least. On the other hand, Q1 askers (most active) commit more than Q2 ones, suggesting an active learning process in which seeking answers to technical challenges is accompanied by experimenting with the proposed solutions, and committing the changes to GitHub. However, this conjecture will have to be further investigated.

The least active askers author more commits than others. However, more active askers are not indistinguishable in terms of their commit activity: the most active askers commit more than the second most active ones.

Are heavy answer givers also heavy committers?: With the data sorted along the A dimension in decreasing order, we split the C dimension into quartiles and compare the resulting groups pairwise. The results (Figure 3d) reveal another perfect ordering: more active answerers commit more. This observation is in line with its complement above: more active committers provide more answers. This suggests that answering questions on StackOverflow can be seen as a proxy for one's commit activity.

More active StackOverflow answerers make more commits on GitHub.

Summary: We find a *direct relationship between GitHub commit activity and StackOverflow question answering activity*: the more active a committer, the more answers she gives; similarly, the more active an answerer, the more commits she authors. In contrast, we find an *inverse relationship between GitHub commit activity and StackOverflow question asking activity*: active GitHub committers ask fewer questions than others; less active question askers produce more commits. Overall, these findings suggest that an activity-based ranking of StackOverflow contributors reflects one extracted from their open-source contributions to GitHub, increasing the confidence in the reliability of SO-based social signals (e.g., heavy SO answerers tend to be also very active GitHub committers).

V. Intermediate view

In the macroscopic view we have ignored when commits, questions and answer occur, restricting our attention solely to the number of events. Next we refine the approach and include information about the time intervals separating subsequent events. Following Xuan et al. [16], we define a *working rhythm* of an individual in a given activity (committing, asking/answering questions) as determined by a series of interactivity times: $\Delta t_i = t_{i+1} - t_i$, where t_i is a timestamp of the i'th activity instance (commit, question, answer). Specifically, in this section we focus on committing rhythms.

A. Methodology

We are interested in understanding how developers distribute their time over commits, i.e., whether or not they are following a steady working rhythm. To evaluate the committing rhythm of a developer, we calculate the Gini index over the lengths of her inter-commit time intervals. The Gini index is a popular econometric measure designed to study inequality of income or wealth distributions; it is often being used to aggregate software metrics, e.g., [27], [28]. The Gini index values range over $[0; 1 - \frac{1}{n}]$, where n is the number of values being aggregated: Gini index equal to zero would indicate an egalitarian distribution of developers' time over commits, i.e., developers following a steady working rhythm; Gini index close to the maximum would correspond to one big inter-commit time interval and numerous small inter-commit time intervals. Since the number of inter-commit intervals significantly varies from one developer to another, we normalise the Gini index values by dividing them with $1 - \frac{1}{n}$.

Similarly to Section IV, we split the individuals into quartiles depending on the number of questions asked or answers given on StackOverflow. Then, we compare the normalised Gini values computed for the time series of GitHub inter-commit intervals, for the individuals associated with each quartile. This helps us understand whether different groups of developers exhibit different working rhythms. For example, active StackOverflow answerers (shown above to be also active committers), presumably more experienced, might work differently than less active ones. To compare multiple distributions (each quartile generates a distribution of Gini index values), we follow the methodology described in Section IV-A.

B. Results

We have first used the number of questions as a basis for grouping the developers into quartiles. However, the median equals 0, i.e., half of the developers did not ask any questions, and we can no longer distinguish between $Q1$ and $Q2$. Hence, we compare three distributions of GitHub inter-commit time intervals: normalized Gini index values of the individuals that do not ask questions $Q12$, that ask few questions $Q3$, and that ask the most questions $Q4$. Using the $\widetilde{\mathbf{T}}$ procedure we conclude that the normalized Gini index values are higher for active askers than for developers that do not ask questions ($p = 0.013$), i.e., active askers distribute their effort in a less egalitarian way than developers that do not ask questions. In other words, developers who ask many questions on StackOverflow commit changes to GitHub in bursts of intense activity followed by longer periods of inactivity, i.e., they focus their attention at any given time. Specialization (or focus) of developers has also been noted previously in the context of activity types (e.g., coding versus translating) or files touched as part of a shared project [25], [29].

On the other hand, no differences can be observed between the normalised Gini index values for individuals grouped into quartiles based on the number of answers given on StackOverflow. Therefore, *asking* questions on StackOverflow influences how developers distribute their time over commits on GitHub, while *answering* questions does not seem to have the same effect. This observation is in line with our previous conjecture on developers learning from StackOverflow and committing their experiences to GitHub, as well as the literature on the impact of social media on software development [1], [11].

Active StackOverflow askers distribute their work in a less egalitarian way (i.e., focus their attention more) than developers that do not ask questions.

VI. Microscopic view

So far we have ignored the ordering between commits, questions and answers. The microscopic view takes this temporal aspect into account by considering committing, asking and answering as time-series. To study the interaction between activities we follow the approach proposed by Xuan et al. [16].

A. Interaction between the activities

Consider the timeline of GitHub and StackOverflow activities of a particular developer (Figure 4a). Let $\mathcal{A}$ and $\mathcal{B}$ be two activities we would like compare (e.g., C and Q). For every event $t_i^{\mathcal{A}}$ of $\mathcal{A}$ we measure the *evaluation latency*[8] $\epsilon_i^{\mathcal{B}}$ as the difference between the earliest event of $\mathcal{B}$ following $t_i^{\mathcal{A}}$ and $t_i^{\mathcal{A}}$, and the *response latency* $\rho_i^{\mathcal{B}}$ as the difference between $t_{i+1}^{\mathcal{A}}$ and the latest event of $\mathcal{B}$ preceding $t_{i+1}^{\mathcal{A}}$ (Figure 4b). The sequences $\epsilon^{\mathcal{B}}$ and $\rho^{\mathcal{B}}$ characterise the relationship between $\mathcal{A}$ and $\mathcal{B}$. Next, to study whether the sequence of $\mathcal{B}$ events for this particular

[8]For the sake of readability we use a lightly different notation than in the original paper [16].

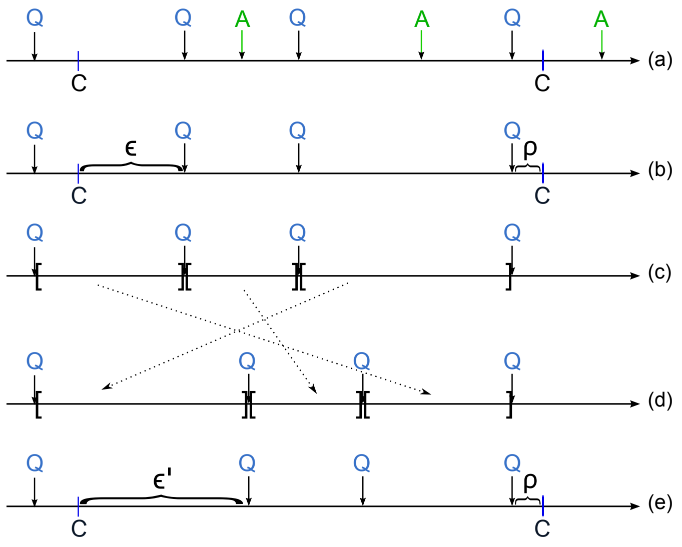


Fig. 4: The steps to generate a simulated time-series of StackOverflow activities.

developer could have occurred by chance, we create m random permutations of $\mathcal{B}$ events[9] ($\mathcal{B}_1, \ldots, \mathcal{B}_m$). Reshuffling is done in such a way that the durations of the "idling periods" between two consecutive events of activity $\mathcal{B}$ are preserved, but the order of the "idling periods" is randomised (Figures 4c,d). Let $\epsilon^{\mathcal{B}_1}, \ldots, \epsilon^{\mathcal{B}_m}$ and $\rho^{\mathcal{B}_1}, \ldots, \rho^{\mathcal{B}_m}$ be series of evaluation and response latencies corresponding to $\mathcal{B}_1, \ldots, \mathcal{B}_m$ (Figure 4e).

Finally, we aggregate all the sequences $\epsilon^{\mathcal{B}}$ for the different developers into $E^{\mathcal{B}}$, all $\rho^{\mathcal{B}}$ into $P^{\mathcal{B}}$, etc. Then, we compare $E^{\mathcal{B}}$ with each one of $E^{\mathcal{B}_1}, \ldots, E^{\mathcal{B}_m}$ and $P^{\mathcal{B}}$ with each of $P^{\mathcal{B}_1}, \ldots, P^{\mathcal{B}_m}$. However, as opposed to Section IV-A, we are no longer interested in performing all pairwise comparisons between different groups, but in comparing one of the distributions ("control") against multiple alternatives ("treatments"). This kind of comparisons is known as a *Dunnett-type contrast*, and it is also supported by $\widetilde{\mathbf{T}}$. Hence, we apply $\widetilde{\mathbf{T}}$ for Dunnett type contrasts with $E^{\mathcal{B}}$ and $P^{\mathcal{B}}$ as control groups, and the traditional 5% family-wise error rate:

- If $\mathcal{A}$ and $\mathcal{B}$ are independent from each other, $E^{\mathcal{B}}$ and $P^{\mathcal{B}}$ will be statistically indistinguishable from their simulated counterparts.
- If $\mathcal{A}$ delays $\mathcal{B}$, $E^{\mathcal{B}}$ will be statistically longer than the simulated evaluation latencies. Similarly, if $\mathcal{B}$ delays $\mathcal{A}$, $P^{\mathcal{B}}$ will be statistically longer than the simulated response latencies.
- If $\mathcal{A}$ accelerates $\mathcal{B}$, $E^{\mathcal{B}}$ will be statistically shorter than the simulated evaluation latencies. Similarly, if $\mathcal{B}$ accelerates $\mathcal{A}$, $P^{\mathcal{B}}$ will be statistically shorter than the simulated response latencies.

To address the potential inconsistencies between the $\widetilde{\mathbf{T}}$ results for the m randomisations, we apply the following schema. We say that two activities *do not influence* each other (denoted "none") if at most one of the simulations resulted in a statistically significant comparison. Otherwise, we speak of

[9]In our experiments we chose $m = 10$.

TABLE II: Mutual influence of StackOverflow activities and GitHub committing, for different committers (from least active $Q1$, to most active $Q4$).

Q	Influence of			
	asking on committing	committing on asking	answering on committing	committing on answering
$Q1$	none	none	none	none
$Q2$	none	inconclusive	inconclusive	inconclusive
$Q3$	accelerates	accelerates	accelerates	accelerates
$Q4$	accelerates	accelerates	accelerates	accelerates

acceleration (*delay*) if at least 80% of the simulations have been found to indicate acceleration (delay). In all other cases we say that the influence is *inconclusive*.

Since we focus on the impact of StackOverflow activities (Q, A) on GitHub committing (C) and vice versa, we always choose GitHub committing as one of the activities and vary a different StackOverflow activity as the other one.

B. Results

To investigate whether StackOverflow activities impact only specific groups of committers (e.g., those very active), we split the committers into quartiles based on their total *number of commits* (as in the previous sections). Results of our investigation are summarised in Table II. First of all, we observe that the results are consistent. Moreover, for the most active half of the committers ($Q3$ and $Q4$), the real latencies consistently tend to be lower than the simulated ones. This suggests that for these developers, committing and asking questions accelerate each other, as well as committing and answering questions accelerate each other.

For active committers, asking questions on StackOverflow catalyses committing on GitHub. Similarly, for active committers, answering questions on StackOverflow catalyses committing on GitHub.

Similar differences in influence of StackOverflow activities on GitHub committing between more and less active developers can be observed after grouping by *length of involvement* in GitHub (the catalysis is more visible for individuals who have been involved in GitHub for sufficiently long time), or *number of questions* asked on SO (the catalysis is more visible for active askers).

Finally, when grouping is done according to the *number of answers* given on SO, slightly different results are obtained (Table III). By answering questions, there is a benefit only for the most active answerers ($Q4$): answering questions on StackOverflow and committing changes to GitHub accelerate each other. In contrast, by asking questions, acceleration is visible for both the active answerers ($Q3$ and $Q4$) as well as for the developers that do not answer any questions at all (exclusive askers or exclusive knowledge seekers; $Q1$): asking questions on StackOverflow and committing changes to GitHub accelerate each other.

TABLE III: Mutual influence of StackOverflow activities and GitHub committing, for different answerers (from least active $Q1$, to most active $Q4$). Individuals in $Q1$ do not give answers.

Q	Influence of			
	asking on committing	committing on asking	answering on committing	committing on answering
$Q1$	accelerates	accelerates	n/a	n/a
$Q2$	none	none	none	none
$Q3$	accelerates	accelerates	none	none
$Q4$	accelerates	accelerates	accelerates	accelerates

For the most active answerers as well as for developers that do not answer any questions at all, their StackOverflow activities accelerate their GitHub committing.

VII. Conclusions and future work

In this paper we studied the relationship between question & answer activities carried out by individuals on StackOverflow and their contributions to GitHub repositories. Our findings are based on the data for 46,967 users active both on StackOverflow and GitHub.

First, we focussed on differences in StackOverflow involvement of the GitHub developers (RQ1). We observed that individuals who tend to ask few questions tend to have a high number of commits, and individuals with a high number of commits tend to ask few questions. Moreover, individuals that tend to answer many questions tend to have a high number of commits, and individuals that have a high number of commits tend to answer many questions. This suggests that highly productive (in terms of GitHub commits) individuals tend to take the role of a "teacher" more actively involved in providing answers rather than asking questions.

Next, we studied whether the working rhythm of the GitHub contributors is related to their StackOverflow activities (RQ2). We observed that individuals that tend to answer many questions distribute their work in a less uniform way than developers that do not ask questions at all. No differences were observed between the work distributions for individuals grouped based on the number of answers given.

Finally, we showed that despite interruptions incurred, for active GitHub developers StackOverflow activities are positively associated with the social coding in GitHub (RQ3). Similar observations hold for active askers as well as individuals who have been involved in GitHub for sufficiently long time. Finally, StackOverflow activities accelerate GitHub committing also for the most active answerers as well as for developers that do not answer any questions at all.

To deepen our understanding of the impact StackOverflow has on GitHub we intend to expand the research presented as follows. First of all, we would like to refine the classification of activities: we plan to distinguish between questions and answers pertaining to different subjects (as expressed by StackOverflow tags) and commits pertaining to different projects. Then, using information retrieval techniques we intend to classify questions and answers as being related, or not, to a given commit. For instance, we expect to observe a closer relation between commits and the topics of questions asked compared to the relation between commits and the topics of answers given, as answers are more likely to pertain to the general knowledge of the individual. As a continuation of our work on the impact of the number of questions on the working rhythms, we intend to study to what extent can the inequality in the inter-commit time intervals' distribution be explained by different aspects of the GitHub projects and their developers (including StackOverflow activities of the latter ones). To measure the explanation we intend to employ the Theil index [30], [31]. Moreover, we plan to investigate the impact of the committing rhythm of the individual on her activity rhythm on StackOverflow: are questions being asked or answered when no committing is done, or rather interleaved with commits? We also would like to augment our study of the intermediate view by applying further models of inter-event time distribution [32], [33] to our data. Finally, to obtain additional insights in the combined StackOverflow & GitHub activities we would like to apply process mining techniques originally developed for information systems [34] and successfully applied to traditional software repositories such as version control systems, mail archives and bug trackers [35].

Acknowledgments

We are grateful to Dr. Frank Konietschke for providing us with the R implementation of the $\widetilde{\mathbf{T}}$ procedure. Bogdan Vasilescu has been supported by project 600.065.120.10N235 of the Dutch Science Foundation (NWO). Vladimir Filkov gratefully acknowledges support from the US Air Force Office of Scientific Research, award FA955-11-1-0246.

References

[1] M.-A. D. Storey, C. Treude, A. van Deursen, and L.-T. Cheng, "The impact of social media on software engineering practices and tools," in *FoSER*. ACM, 2010, pp. 359–364.

[2] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: interleaving web foraging, learning, and writing code," in *CHI*. ACM, 2009, pp. 1589–1598.

[3] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest Q&A site in the west," in *CHI*. ACM, 2011, pp. 2857–2866.

[4] S. Deterding, "Gamification: designing for motivation," *Interactions*, vol. 19, no. 4, pp. 14–17, 2012.

[5] A. Capiluppi, A. Serebrenik, and L. Singer, "Assessing technical candidates on the social web," *IEEE Software*, vol. 30, no. 1, pp. 45–51, 2013.

[6] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow," Georgia Institute of Technology, Tech. Rep., 2012.

[7] A. Bacchelli, L. Ponzanelli, and M. Lanza, "Harnessing Stack Overflow for the IDE," in *RSSE*. IEEE, 2012, pp. 26–30.

[8] J. Cordeiro, B. Antunes, and P. Gomes, "Context-based search to overcome learning barriers in software development," in *RAISE*, 2012, pp. 47–51.

[9] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang, "Network structure of social coding in GitHub," in *CSMR*. IEEE, 2013, pp. 323–326.

[10] P. J. Adams, A. Capiluppi, and C. Boldyreff, "Coordination and productivity issues in free software: The role of Brooks' law," in *ICSM*. IEEE, 2009, pp. 319–328.

[11] L. A. Dabbish, H. C. Stuart, J. Tsay, and J. D. Herbsleb, "Social coding in GitHub: transparency and collaboration in an open software repository," in *CSCW*. ACM, 2012, pp. 1277–1286.

[12] B. Evans and S. Card, "Augmented information assimilation: social and algorithmic web aids for the information long tail," in *CHI*. ACM, 2008, pp. 989–998.

[13] D. Posnett, E. Warburg, P. Devanbu, and V. Filkov, "Mining Stack Exchange: Expertise is evident from earliest interactions," in *Social Informatics*. ASE, 2012.

[14] J. Eyolfson, L. Tan, and P. Lam, "Correlations between bugginess and time-based commit characteristics," *Empirical Software Engineering*, pp. 1–31, 2013.

[15] B. Vasilescu, "Academic papers using Stack Overflow data," http://meta.stackoverflow.com/q/134495/185480, 2012.

[16] Q. Xuan, M. Gharehyazie, P. T. Devanbu, and V. Filkov, "Measuring the effect of social communications on individual working rhythms: A case study of open source software," in *Social Informatics*. ASE, 2012.

[17] H. A. Simon, "Designing organizations for an information rich world," in *Computers, communications, and the public interest*. Johns Hopkins Press, 1971, pp. 37–72.

[18] G. Gousios and D. Spinellis, "GHTorrent: Github's data from a firehose," in *MSR*. IEEE, 2012, pp. 12–21.

[19] C. Bird, A. Gourley, P. T. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *MSR*. ACM, 2006, pp. 137–143.

[20] M. Goeminne and T. Mens, "A comparison of identity merge algorithms for software repositories," *Science of Computer Programming*, vol. 78, no. 8, pp. 971–986, 2011.

[21] E. Kouters, B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand, "Who's who in Gnome: Using LSA to merge software repository identities," in *ICSM*. IEEE, 2012, pp. 592–595.

[22] K. R. Gabriel, "Simultaneous test procedures—some theory of multiple comparisons," *ANN MATH STAT*, vol. 40, no. 1, pp. 224–250, 1969.

[23] E. Brunner and U. Munzel, "The Nonparametric Behrens-Fisher Problem: Asymptotic Theory and a Small-Sample Approximation," *Biometrical Journal*, vol. 42, no. 1, pp. 17–25, 2000.

[24] F. Konietschke, L. A. Hothorn, and E. Brunner, "Rank-based multiple test procedures and simultaneous confidence intervals," *Electronic Journal of Statistics*, vol. 6, pp. 738–759, 2012.

[25] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload—a case study of the Gnome ecosystem community," *Empirical Software Engineering*, pp. 1–54, 2013.

[26] StackOverflow-community, "Top users on stackoverflow: slackers or superstars?" http://meta.stackoverflow.com/q/12468, 2009.

[27] R. Vasa, M. Lumpe, P. Branch, and O. M. Nierstrasz, "Comparative analysis of evolving software systems using the Gini coefficient," in *ICSM*. IEEE, 2009, pp. 179–188.

[28] B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand, "You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics," in *ICSM*. IEEE, 2011, pp. 313–322.

[29] D. Posnett, R. D'Souza, P. Devanbu, and V. Filkov, "Dual ecological measures of focus in software development," in *ICSE*. IEEE, 2013, pp. 452–461.

[30] A. Serebrenik and M. G. J. van den Brand, "Theil index for aggregation of software metrics values," in *ICSM*. IEEE, 2010, pp. 1–9.

[31] F. A. Cowell and S. P. Jenkins, "How much inequality can we explain? a methodology and an application to the United States," *Economic Journal*, vol. 105, no. 429, pp. 421–430, March 1995.

[32] A.-L. Barabási, "The origin of bursts and heavy tails in human dynamics," *Nature*, vol. 435, pp. 207–211, 2005.

[33] Y. Wu, C. Zhou, J. Xiao, J. Kurths, and H. J. Schellnhuber, "Evidence for a bimodal distribution in human communication," *PNAS*, vol. 107, no. 44, pp. 18 803–18 808, 2010.

[34] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," *Fundam. Inform.*, vol. 94, no. 3-4, pp. 387–412, 2009.

[35] W. Poncin, A. Serebrenik, and M. G. J. van den Brand, "Process mining software repositories," in *CSMR*. IEEE, 2011, pp. 5–14.