GitHub Developers Use Rockstars to Overcome Overflow of News

Michael J. Lee
University of Washington
Mary Gates Hall, Box 352840
Seattle, WA 98195-2840, USA
mjslee@uw.edu

Jungpil Hahn
National University of Singapore
21 Lower Kent Ridge Road,
Singapore, 119077
jungpil@nus.edu.sg

Bruce Ferwerda
Yonsei University
134 Shinchon-dong
Seodaemun-gu, Seoul, S.Korea
bruce@yonsei.ac.kr

Jae Yun Moon
Korea University
145 Anam-ro, Seongbuk-gu,
Seoul, South Korea
jymoon@korea.ac.kr

Junghong Choi
Yonsei University
134 Shinchon-dong
Seodaemun-gu, Seoul, S.Korea
junghong@yonsei.ac.kr

Jinwoo Kim
Yonsei University
134 Shinchon-dong
Seodaemun-gu, Seoul, S.Korea
jinwoo@yonsei.ac.kr

CHI 2013 Extended Abstracts, April 27–May 2, 2013, Paris, France.
ACM 978-1-4503-1952-2/13/04.

Abstract
Keeping track of a constantly updating stream of news items on social networking enabled software development sites may be difficult. We analyzed the actions of 544 GitHub.com developers working across 5,657 projects to examine how the network of developers and projects influence where developers choose to contribute. Our analyses revealed the existence of a group of extremely well connected developers, or *rockstars*. We found that these rockstars': 1) actions have a greater influence on their followers compared to regular developers, 2) type of action affect their followers differently, 3) influence on followers may depend on a project's age, 4) increased activity on a project increases activity by followers, and 5) followers use as guides to projects to work on. We discuss the implications of these findings to the design of software development environments.

Author Keywords
Open source; social computing; social coding; GitHub.

ACM Classification Keywords
H.5.3. [Group and Organization Interfaces]: Computer-Supported Cooperative Work.

General Terms
Human factors; design.

Introduction

Recently, many websites have started implementing social network service (SNS) features, allowing their users to easily share information and stay aware of connected people's activities. Adding SNS features to websites have been shown to increase activity and collaboration among users [14]. One domain where SNS features are starting to appear is in software development, which raises important questions about how these features could affect developers' work, their decisions to contribute to others' projects, and how their social network influences these actions.

The importance of social factors in traditional software development has been stressed in many prior works. For example, software development has been conceptualized as a highly collaborative activity where developers work on various, fragmented pieces of code to make larger software applications [1]. DeMarco found that developers spend up to 70% of their time working with others [5] and similarly, Perry et al. reported that over half of developers' time was spent interacting with coworkers [12] to increase awareness [10] and create code [9].

Current online repositories (e.g., Sourceforge.net and Google Code) enable individual developers and teams to collaborate on projects but lack features to make social ties and keep up with others' updates. One site that has successfully integrated SNS features is GitHub (www.github.com), which allows developers to "follow" other developers, or to "watch" others' projects. Once these ties are established, any activity by these connections will appear in the developers' news feeds.

Using in-depth interviews, Dabbish et al. [4] found that GitHub developers make rich inferences about others and others' projects based on information they receive from these SNS features. However, although the use of SNS features may increase social awareness and transparency, as developers add more social connections, they also increase the number of activity notifications in their news feeds. Once the quantity of these notifications become excessive, it may cause stress [13], dilute the usefulness of the information [6], and be a source of distraction [11] from working on projects. These prior findings raise additional questions about how increased social awareness and transparency of others' work affect developers and their participation in others' software projects [4,13].

To investigate the effects of social awareness and transparency on developers' contributions to others' projects, an exploratory analysis was conducted (explained further in our methods section). We compiled a dataset of 544 randomly selected developers over three months from GitHub. Basic descriptive statistics indicated that GitHub developers using social features received a large number of activity notifications in their news feeds (more than half of the developers in our sample received at least 254 news feed items per day on average). Despite the potential for information overload, these developers appeared to be productive members of the site, not only working on their own projects but also participating in others' projects. In fact, our data showed that those developers who had more SNS connections – and therefore received more frequent updates to their news feed – were actually more likely to contribute to socially connected projects than those with fewer SNS connections. This suggests that developers were able to

somehow cope with the exorbitant influx of activity information from their social connections.

A possible strategy these developers used to cope with the large amount of activity information was to follow and learn from the actions of other developers. This tactic is related to social learning theory, which argues that people learn from observing role models in day-to-day life and that they will begin to act like the people they observe [2,3], and social navigation [7], which states that users navigate through information space guided by the activities of others. Dabbish et al. [4] labels these role models as "rockstars" in their qualitative study of GitHub developers. Rockstars are developers with thousands of followers and deemed to have some special skill or knowledge. Other developers can learn from these rockstars by tracking how they are coding, where they are giving their attention, and how they are solving problems [4].

Based on these ideas, we decided to focus on the following research question: how do extremely well connected individuals influence others' activities on an SNS-enabled software development site? From this question, we established five propositions to investigate further: 1) do rockstars' really have a larger influence on their followers compared to regular developers? 2) do certain actions by rockstars affect their followers differently? 3) does a project's age affect a rockstars' influence on followers? 4) does the amount of activity by a rockstar on a project influence followers? And 5) do followers use rockstars as guides to projects?

Method

Detailed activity and social connectivity data was collected over a three-month period for a random sample of 544 developers. A total of 38,681 actions across 5,657 projects were observed. Our sample controlled for those using the site as a code dump, and included only developers that performed at least one action, and was following or watching at least one other person or project (owned by someone else). A custom script using GitHub's API logged all users' actions every three minutes into a database. Actions included: *who* (developer), *what* (action), *where* (project), and *when* (timestamp). The actions included *commit comment, create, delete, download, follow, fork, gist, gollum, issue comment, issues, public, pull request review comment, pull request, push,* and *watch* events (which are further explained in [8]). The following metadata was also collected: when a project first began, who developers were connected to (who they were following, and who were following them), and what projects they were connected to (i.e., watching).

One particularly interesting group of developers collaborating often with others were "rockstars" [4]. Ordering developers by their number of *followers* and number of *actions performed* revealed four rockstars (followers: $\bar{x}$=2035.5, SD=1850, actions: $\bar{x}$=100.5, SD=60.8), that were significantly different from other developers (followers: $\bar{x}$=40.4, SD=69.7, actions: $\bar{x}$=32.1, SD=87.8). Combined, the four rockstars were connected to 183 projects (91 owned by the rockstars, and 92 owned by others), and 2,139 unique followers.

Findings

Using a data visualization tool for initial insights and statistical analyses for subsequent empirical exploration, we observed a number of empirical regularities, which we summarize as propositions.

Rockstars Do Strongly Influence their Followers
Results of a linear regression indicated that more of the rockstars' followers performed subsequent actions ($\bar{x}$=83.2, *SD*=27.9) on the same projects than the followers of regular developers ($\bar{x}$=12.7, *SD*=8.5), meaning that rockstars have a bigger influence on their followers. Being a rockstar was found to be a significant predictor (β=96.8, p<.001) accounting for 44.8% of the variance (R^2=.448, F(1, 394)=319.92, p<.001) explaining the number of followers performing a subsequent action.

Rockstars' Action Type Affects Project Attractiveness
Next, we wanted to investigate if certain types of actions by rockstars affected how many of their followers performed subsequent actions on the same projects. To test this, we focused on projects that rockstars could contribute to, without having any strong ties. We identified 92 projects that were not owned or associated with rockstars.

For the analysis, we divided the actions performed on these projects into two groups: 1) *watch* and *fork*, and 2) *open issue*, *pull request*, and *comment*. This distinction was made because the actions of the first group trigger a news feed event, but do not necessarily indicate a contribution to the project. In contrast, the actions in the second group trigger a news feed event and constitute a contribution or addition to the project. Additionally, projects where rockstars were the last contributor to a project were excluded (n=11) since there was no subsequent follower activity to analyze.

The results of the linear regression indicated that the action type of rockstars reliably explained 11.6% of the variance (R^2=.116, F(1,79)=10.32, p<.01) predicting the subsequent actions of their followers (β=.04, p<.01). So, the type of action a rockstar performs will attract a different amount of participation from their followers on the same project. When a rockstar performs a *watch* or *fork* action, followers are performing fewer actions in the same project compared to when they are performing an *open issue*, *pull request*, or *comment* action.

Project Age Affects Project Attractiveness
Next, we wanted to observe whether the age of a project affected the influence of a rockstar on their followers. For this analysis we used the 92 projects that rockstars did not own and were not associated to. These projects' absolute age ranged from 4 to 40 months ($\bar{x}$=12.7, *SD*=8.5).

Results of a linear regression indicated that the age of projects explained 10.8% of the variance (R^2=.108, F(1, 90)=10.85, p<.01) of the subsequent actions by rockstars' followers on the same projects (β=-.59, p<.01). Furthermore, this was a negative relationship, meaning that rockstars' actions on older projects were correlated with fewer subsequent actions by followers.

Rockstars' Intensity Affects Project Attractiveness
We wanted to test whether high activity on a project by rockstars affected their followers' contributions to the same project. To investigate this, we examined the actions of rockstars' on the projects they owned (n=91) and the number of followers doing a subsequent action on the same projects. We chose to focus on projects owned by rockstars since we could control for contributions by others (e.g., actions such as *pull requests* require the intervention of an owner, which could potential disrupt or halt further actions).

Results of a linear regression indicated that when rockstars perform more actions on their owned projects, it attracts more followers to perform actions on the same project. The number of actions rockstars made in their projects explained 70.8% of the variance (R^2=.708, F(1, 89)=215.76, p<.001) of the subsequent actions made by their followers on the same projects (β=.44, p<.001). This indicates that followers are attracted to perform actions on projects as rockstars increase their activity on those projects.

Rockstars' Actions Guide Other Developers to Projects
Finally, we wanted to see whether followers were using the rockstars as a guide for finding projects to work on. We focused on projects rockstars did not own since we were interested to see if followers were guided to projects where rockstars did not have strong ties.

Using a data visualization tool as a guide, two types of behaviors emerged: 1) after a rockstar performs any action on a project, a follower also contributes to the project, but by doing an unrelated action (e.g., a rockstar performs a *comment* action, and the follower performs a subsequent *pull request* action), and 2) after a rockstar opens a new issue or comments on an existing issue reported about a project, a follower also comments on the same issue thread.

Similar to social learning theory and social navigation, where people follow the actions of others [2,7], a follower's action was preceded by a rockstar's action in the same project in both of the cases above. To investigate the size of this effect, we analyzed the 92 projects that rockstars contributed to, but did not own. A total of 307 followers performed subsequent actions following a rockstar's action on the same project. However, only two cases among these were found where followers contributed to the same issue thread. The remaining 305 followers performed actions on the same project, but on different issues.

Discussion

Our findings show that rockstars generally have a larger influence on their followers than ordinary developers have on their followers. Based on our data, developers using SNS-features are inundated with updates in their news feeds, potentially making it difficult to filter out interesting or relevant projects. Our findings show that developers using SNS features may utilize strategies involving rockstars to guide, filter out, and identify projects to work on.

One of our results found that certain actions by rockstars affect their followers differently. This may be explained by considering that a *watch* or *fork* action might be interpreted by a follower as the initiation of interest in a project, whereas other contributions might indicate more serious and substantive interest in the project. Another finding was that a project's age affects rockstars' influence on followers. This might be attributed to the maturity of a project, where older, stable projects may have fewer changes to implement.

Given our results, there are several HCI-related implications to consider about the use of SNS features on sites like GitHub to improve the user experience and help developers find projects to work on. Currently, Githib's news feed displays a mixture of information even though some news items may be more relevant or useful to the developer. Since we found that the activity intensity of rockstars on their own projects had a significant effect on attracting others' contributions, it

may be advantageous to showcase these or create a recommendation system so more developers can be exposed to these projects. Exploring new ways to recommend interesting projects may also lead to a stronger sense of community and more collaboration.

In addition, GitHub currently provides a tool that allows users to search for text within projects, users, and code. However, it is unclear how useful this search tool will be for finding relevant or interesting projects to work on. Given our results showing that developers' actions are highly correlated with those of rockstars' actions, providing search tools including SNS-related functions (e.g., ranking results by contributors' social connections, sorting by the total number of actions on a project, or filtering by types of actions) may help developers identify interesting projects.

Finally, our results focused on a subset of developers who used social features, and collaborated with other developers. We hope this work can be extended by examining other types of developers, including those who do not use SNS features, to create new and improved collaborative online coding environments.

References

[1] Ahmadi, N., Jazayeri, M., Lelli, F., & Nesic, S. A Survey on Social Software Engineering. *Int'l Social Software Engineering and Applications* (2008), 1–12.

[2] Bandura, A. *Social Learning Theory*. New York: General Learning Press (1977).

[3] Burke, M., Marlow, C., & Lento, T. Feed Me: Motivating Newcomer Contribution in Social Network Sites. *ACM CHI* (2009), 945-954.

[4] Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. Social coding in GitHub: Transparency and collaboration in an open software repository. *ACM CSCW* (2012), 1277-1286.

[5] DeMarco, T. & Lister, T. *Peopleware: Productive projects and teams*. New York: Dorset House Publishing Co., Inc., 1987.

[6] De Souza, C.R.B., Redmiles, D., & Dourish, P. 'Breaking the Code', Moving between Private and PublicWork in Collaborative Software Development. *ACM SIGGROUP* (2003), 105-114.

[7] Dourish, P. & Chalmers, M. Running Out of Space: Models of Information Navigation. *HCI* (1994).

[8] GitHub. https://help.github.com/categories/63/articles. Retrieved August 20, 2012.

[9] Ko, A.J., DeLine, R., & Venolia, G. Information Needs in Collocated Software Development Teams. *ICSE* (2007), 344-353.

[10] Lee, M.J., Ko, A.J. Representations of user feedback in an agile, collocated software team. *IEEE CHASE* (2012), 76-82.

[11] McFarlane, D. & Latorella, K. The scope and importance of human interruption in human-computer interaction design. *HCI* (2002), 1-61.

[12] Perry, D.E., Staudenmayer, N.A., & Votta, L.G. People, Organizations and Process Improvement. *IEEE Software* (1994), 36-45.

[13] Stuart, H.C., Dabbish, L., Kiesler, S., Kinnaird, P., & Kang, R. Social transparency in networked information exchange: a theoretical framework. *ACM CSCW* (2012), 451–460.

[14] Zhao, D. & Rosson, M.B. How and why people Twitter: The role that micro-blogging plays in informal communication at work. *ACM GROUP* (2009), 243-252.