Revisited: Testing Culture on a Social Coding Site

Raphael Pham[*], Leif Singer[†], Olga Liskin[*], Fernando Figueira Filho[‡], Kurt Schneider[*]

[*] Software Engineering Group
Leibniz Universität Hannover, Germany
{firstname.lastname}@inf.uni-hannover.de

[†] CHISEL Group
University of Victoria, Canada
lsinger@uvic.ca

[‡] Cooperative Interaction Lab
Universidade Federal do Rio Grande do Norte, Brazil
fernando@dimap.ufrn.br

Abstract: Testing is an important part of software development. However, creating a common understanding of a project's testing culture is a demanding task. Without it, the project's quality may degrade. We conducted a Grounded Theory study to understand how testing culture is communicated and disseminated in projects on GitHub. We investigated how the transparency of interactions on the site influences the testing behavior of developers. We found several strategies that software developers and managers can use to positively influence the testing behavior in their projects. We report on the challenges and risks caused by this and suggest guidelines for promoting a sustainable testing culture in software development projects.

1 Social Transparency and Testing Culture on GitHub

Social coding sites provide a high degree of social transparency ([SDK+12]). Members are able to easily find out who they are interacting with, whom everyone else is interacting with, and who has interacted with which artifacts. This transparency influences the behavior of software developers [DSTH12]). The social coding site GitHub.com acts as a version control repository with a Web interface, as well as a social network site. Users (*contributors*) browse projects, clone a public repository of interest and make changes to it. Then, they offer these changes back (*making a pull request*) to the *project owner*, who decides whether or not to accept them.

In our study, we explored the prevalent testing behavior on GitHub and the impact of social transparency on it (see [PSL+13]). GitHub's **contribution process** is straightforward: a project owner receives a pull request, manually inspects it, runs a test suite and merges it. However, different factors influence this process. Contributions from unknown developers were checked more thoroughly (*trust*). Small changes (*size*) were accepted without tests while new features (*type*) triggered a demand for automated tests. In our study, several **challenges** for GitHub users became apparent. Project owners felt a need for automated tests simply for reasons of *scale* (too many contributions were flowing in). The *constant flux* of different contributors and the shortness of engagement made it difficult to effectively communicate requirements for automated tests. Different **coping strategies**

emerged: Project owners *lowered the barriers* for contributors to provide tests by using well-known testing frameworks, providing easy *access to learning resources* or *actively supporting* users in writing tests. Contributors reacted to obvious signals for automated testing. They were more inclined to provide tests in their contributions, if they saw automated tests already present in a project. Moreover, contributors heavily relied on existing tests as examples for their own test cases. The **impact of social transparency** on testing behavior was manifold: Some projects used their testing practices as advertisement for high quality development. Effective communication of testing guidelines removed uncertainties in contributors about how to participate correctly. Also, contributors to well-tested projects reported to feel more confident as problems would quickly become visible.

2 Conclusion and Outlook

Project owners on a social coding site interact with contributors with varying values regarding testing. Our study reports on the influences of GitHub's high degree of social transparency, low barriers, and high degrees of integration and centralization on testing practices. On GitHub, developers browse for projects of interest, contribute swiftly and gradually get more involved. Other users quickly contribute without further involvement. This creates large peripheries of contributors for popular projects. In an ongoing initiative [PSS13], we are exploring how to direct this peripheral potential towards automated testing by using crowdsourcing mechanisms. This way, projects could make their needs for automated tests more visible to peripheral users. Understanding the impact of social transparency on testing behavior is a key factor in designing a suitable crowdsourcing platform for software testing. Lastly, our findings can help developers to gain insights into issues that contributors may face and strategies for handling them.

References

[DSTH12] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proc. of the ACM 2012 Conf. on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[PSL+13] Raphael Pham, Leif Singer, Olga Liskin, Fernando Figueira Filho, and Kurt Schneider. Creating a Shared Understanding of Testing Culture on a Social Coding Site. In *Proceedings of the 35th International Conference on Software Engineering (ICSE 2013), pages 112 - 121, San Francisco, USA*, 2013.

[PSS13] Raphael Pham, Leif Singer, and Kurt Schneider. Building Test Suites in Social Coding Sites by Leveraging Drive-By Commits. In *Proceedings of the 35th International Conference on Software Engineering (ICSE 2013, NIER Track), pages 1202 - 1212, San Francisco, USA*, 2013.

[SDK+12] H. Colleen Stuart, Laura Dabbish, Sara Kiesler, Peter Kinnaird, and Ruogu Kang. Social transparency in networked information exchange: a theoretical framework. In *Proc. of the ACM 2012 Conf. on Computer Supported Cooperative Work*, CSCW '12, pages 451–460, New York, NY, USA, 2012. ACM.