CS 6704: CTTC 1

Sourabh Shetty

Almost There: A Study on Quasi-Contributors in Open Source Software Projects

I agree with most of the author's assertions in the paper. I have personally experienced such barriers when it comes to contributing to OSS projects too and so can speak from experience. Nonacceptance of a pull-request is indeed disheartening and demotivating, even if the comments from the integrators are polite. A fair bit of the tech community do tend to not be too kind towards new contributors, which contributes to the cycle of people not feeling welcome.

I do disagree with the choice of selecting just 21 of the most popular projects on GitHub. If we're researching the apprehension of these quasi-contributors, we have to consider that many may not feel comfortable contributing to very popular projects at all, and that they might start out with less popular projects instead. Restricting the study to just 21 projects would leave out a large number of such potential contributors. Studying their decision to select less popular projects could lead to greater insights into the barrier potential contributors perceive.

I think one potential solution could be to provide a mechanism to more easily suggest changes to the contributor if the changes are not major. For example if the only reason for nonacceptance is some issue with coding styles, perhaps GitHub could have a feature to allow the integrator to suggest changes without requiring much more effort on the part of the contributor. This could potentially help people stay "in" if they don't feel like their contribution is not welcome.

DeFlaker: Automatically Detecting Flaky Tests

I like the idea behind DeFlaker. Existing tools tend to have extensive coverage including all the code that was not changed, so reducing the coverage to just the changed code is quite inspired and seems intuitive. It is a great idea to be able to filter out the tests whose coverage do not include the changed code.

That being said, I think I would like to see the tool in practice more before I can fully champion it. I don't believe the paper made a compelling case as to how suitable the tool would be for all cases. It seems like the authors have more stated the benefits rather than demonstrably proven them. That is not to say that the results are misleading, more that they could be explained more clearly in the paper.

The authors do seem aware of some of the threats to the validity of the paper, and so I do think that with a bit more work done on it, this can turn out to be an indispensable tool when it comes to testing software after changes.