Testing and Code Review Practices in Research Software Development
(the slides are available under “Presentation Materials” in the above URL)
Date: September 9, 2020
Presented by: Nasir Eisty (California Polytechnic State University)

Q. Do you think the lower emphasis on quality with research software is mostly due to ignorance (i.e., most researchers in the scientific community are not well-trained in software development or computer science in general), or something else?

A. Not only ignorance but also there are challenges to employ testing in research software. Both training and motivation are the key factors here. This depends on culture somewhat. On the positive side, most agree that testing is necessary. Additional recognition is starting to make an impact as well.

Q. How do we ensure software quality without intervening with research and frequent changes to the software, especially if researchers are developing the software?

A. Two processes are possible: Test, then develop a stable software and test afterward. If testing is not feasible, code review would be a nice quality activity to try on. It certainly doesn’t intervent the research but helps produce correct results with confidence.

Q.Did you differentiate in any way between automated testing and manual testing?

A. There are many ways to make testing as automated as possible, but every project needs its own specialized method to get to this point. The line is blurry because determining whether results are correct sometimes requires human intervention. Did not differentiate in the survey, but future surveys will go in-depth on different types of testing. Overall, for a large project, manual testing might be very difficult to perform after every change made to the codebase.

Q. What do you classify as commercial testing methods? (Does it include CI/CD? Fuzzing?) Did you provide a definition to participants?

A. Methods: unit/system/integrations/system/acceptance vs. techniques: fuzzing, boundary values, assertions, etc. -- each category was clearly defined in the survey. Fifteen+ techniques were listed as examples to disambiguate these. This matters because even unit testing has multiple interpretations.

Q. Any thoughts on practical approaches to motivate a culture that emphasizes the importance of testing in a research team?

A. Arranging tutorial/workshop sessions could help motivate the culture by showing the importance of testing. To motivate the developers to spend time on testing is

incentives/rewards.

(Ad from PSIP team) <- 👍 (1)

Q. In certain classes of research software (the ones involving stochastic methods, e.g. Monte Carlo) it is difficult to design a regression test using conventional "commercial" testing software. E.g., re-run of the program may produce different results. Could you please comment on this?

A. Statistical tests would play out the output to give a sense of pass/fail. There are different techniques for non-deterministic testing. As far as I know, there is not a tool available for statistical testing of software but it is doable. Please take a look at this article - https://link.springer.com/chapter/10.1007/978-3-030-50436-6_33

Q. What did “Commercial/IT methods” refer to?

A. Unit testing / integration/system/acceptance testing - common practices

Q. Could you provide a definitive reference for all the different testing methods and techniques? Or share the 15 definitions you used in your survey?

A. https://bssw.io/events/testing-research-software-survey You find many definitions here.

Q. Was there any study of survey respondent bias? It seems likely that people who do testing would be more likely to respond, and the results for research software in the wild might be worse than presented here.

A. This is addressed in the Journal paper to be published! The talk includes poll numbers on how familiar respondents were.

Q. How does team size affect code review practices?

A. Large teams have the ability to specialize (and tend to do more reviews), but small teams usually put more roles on every member. ~100 developer teams do lots of code review, and view it as a must.

Q. It is surprising that majorities of the participants get 75% of code reviewed. This is more than some commercial production environments. Are survey results from large research organizations? What’s the minimum team size for participants? Thanks

A. We spread the survey to a wide range of participants. I think the people who took the survey have experience of code review on their project. Our demographic analysis shows that participants came from a wide range of team size but most came from small teams.

Q. What is meant by Code Review? E.g. a merge request that has to be merged by a developer can be a high or low bar?

A. Code review is a systematic process to review each other's code before merging it into the repository.

Q. There are a lot of tools for CI/CD, but the setup cost is still very high for a research code with limited/untrained developers. How do we lower the bar to entry of these tools?
A. I'm not sure how to lower the setup cost.
Q. Are there training resources for the research team to adopt testing?
A. I'm developing a tutorial. Please keep an eye on IDEAS/BSSw emails.

Q. A resource that may be of interest to Nasir and others on the call: The US Research Software Engineer Association https://us-rse.org, also Society of Research Software Engineering https://society-rse.org/
A. Appreciate it.