Title of Work: Automatically Translating Bug Reports into Test Cases for Mobile Apps

Conference:
ISSTA 2018 Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. Pages 141-152
https://conf.researchr.org/home/issta-2018
The ACM SIGSOFT International Symposium on Software Testing and Analysis is the leading research symposium on software testing and analysis, bringing together academics, industrial researchers, and practitioners to exchange new ideas, problems, and experience on how to analyze and test software systems.

Rationale to ensure venue quality:
ISSTA is considered the one of, if not the, highest ranking conferences on software testing. This reflects on the rankings given by ERA and QUALIS ratings. It also has an H-5 ranking of 25, placing it as the 12th highest Software Engineering conference according to https://aminer.org/ranks/conf.

Problem Statement:
Reading and analyzing bug reports is usually a very high time consuming task, especially when the reports come from end-users or non-technical users that will generally describe the issue in an unorganized manner. The researchers propose a way of reducing this effort by implementing a technique that automatically generates test cases from bug reports making it easier to developers and analyst analyze and reproduce the bugs.

Paper Synopsis:

On mobile devices, a good amount of all the reported bugs of an application come from user generated reports. Most of the apps allow for users to generate these bugs reports explaining what the issue is and how to reproduce it. The generated reports will then reach the developers and analysis who will have the task of analyzing the bug report to properly identify the reported issue and to be able to reproduce it. After the bug has been analyzed it will then be fixed.

The process of analyzing the bug reports alone can take a lot of time. This is because of the different ways users report the issues (some might include steps to reproduce, or sa simple paragraph explaining the problem) and the different ways of explaining the issue (the words they use and the way of explaining the bug can vary depending on the person).

On the paper, the researchers propose a new technique (that was implemented in a tool) to help reduce the amount of effort spent on the bug report analysis task. A very high level description of the new approach, called YAKUSU, is that it first reads the bug reports and based on relevant keywords used on it, then maps it to a set of different elements on the application UI and finally produces a test case that reproduces the bug.

For the study, the researchers executed the approach on 62 applications (one bug report for each app). The set was selected by searching on GitHub database for issues using specific keywords such as “android”, “crash”, “reproduce” and “version”. On the paper, the authors detail the reasoning behind each of the keywords. The search returned a set of 2709 issues out of which, the authors randomly selected 100 of them. They then tried to build the exact version on which the fault was reported, from the 100 apps, they were able to build 91 of them. From those 91 apps, they ended up with the previously mentioned list of 62 application because these were the only ones were they could manually reproduce the reported issues.

The study focused on two areas, how effective the approach is and how efficient it is on doing so. The results regarding how efficient the approach is, according to the authors, are very satisfactory. The tool was able to generate test cases for 59.7% of the reported bugs. The authors made sure that these test cases were not false positives, by manually verifying them. From the rest of the bugs that the tool was not able to produce a test case, the researchers placed them on different categories explaining the reasons why the tool was not able to do so.

To determine how efficient the tool is, the authors timed the execution of the tool through all of its phases. On average, the tool read the report, mapped to the UI elements and generated a test case in 5 minutes. The longest time it took the tool to be finished was around 30 minutes. The researches explain this is due to the nature of the application that the tool was executed against.

Future Work:

To see how the tool can really benefit the developers in real-life scenarios.

To integrate the tool with others to help it increase the amount of test cases generated. For example, this approach could be used alongside the approach described by Chaparro et al [1] where they aim at detecting missing information in bug descriptions and propose a way of letting the reporters know when information is missing.

[1] Oscar Chaparro, Jing Lu, Fiorella Zampetti, Laura Moreno, Massimiliano Di Penta, Andrian Marcus, Gabriele Bavota, and Vincent Ng. 2017. Detecting Missing Information in Bug Descriptions. In Proceedings of ESEC/FSE’17, Paderborn, Germany, September 04-08, 2017, 12 pages. https://doi.org/10.1145/3106237.3106285