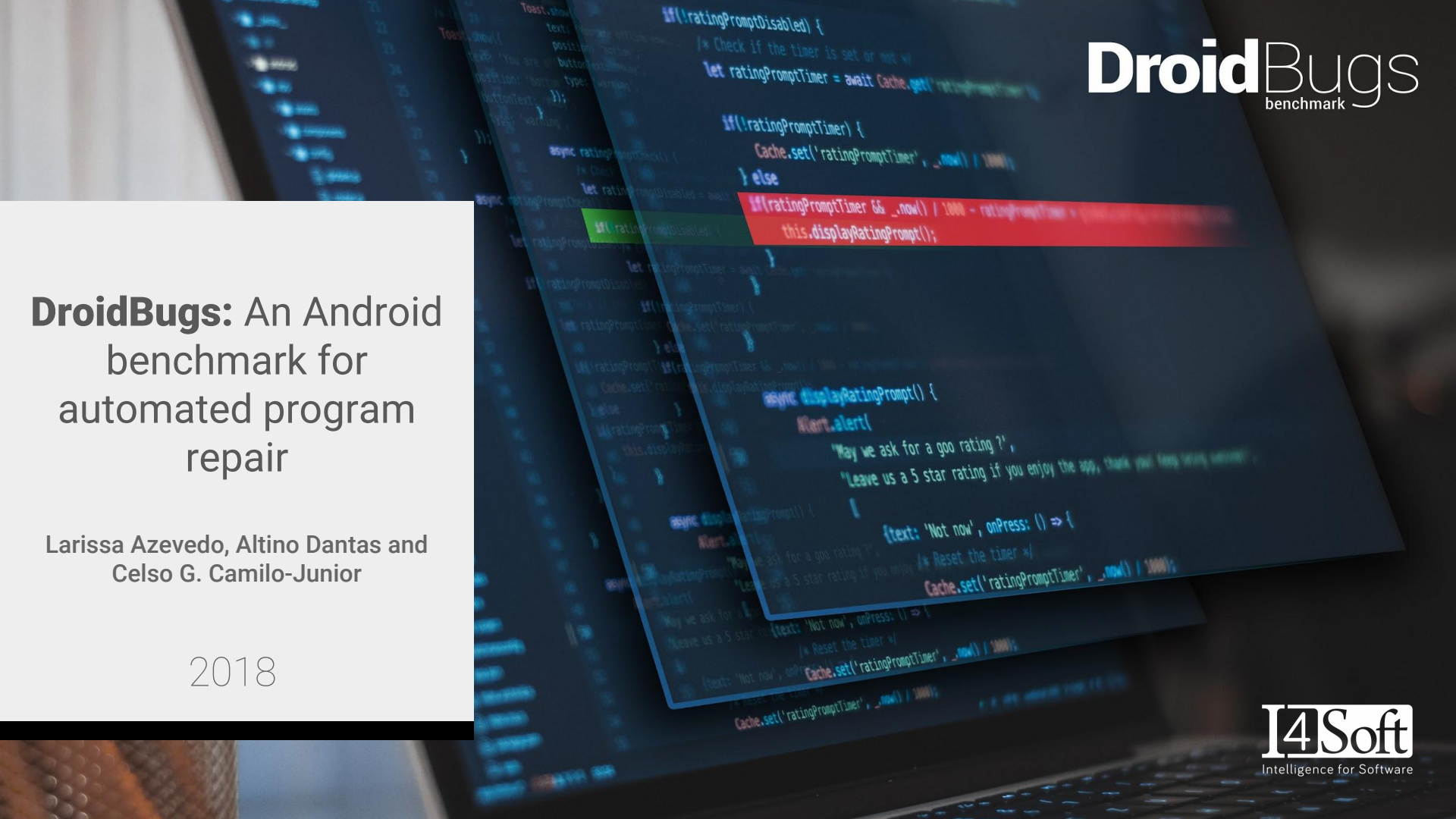


DroidBugs: An Android benchmark for automated program repair

Larissa Azevedo, Altino Dantas and Celso G. Camilo-Junior

2018

I4Soft
Intelligence for Software

Agenda

- Introduction
- Methodology
- The Benchmark
- Experiments
- Discussions
- Conclusions

Introduction

Context

- Automated Program Repair (APR) has gained importance;
- There is a number of Benchmarks for evaluating APR techniques;
- No Benchmark to APR methods applied to mobile applications.

Objective

Introduce the DroidBugs, a Benchmark with **real and reproducible bugs**, collected from open source mobile projects, allowing to evaluate and improve **APR techniques** developed for repairing Android applications.

Introduction | Contributions

- **Introduce** and **provide** the first public Benchmark for APR in the context of Android development;
- Apply a recent APR tool, **Astor4Android**, and report results on **localizing** and **repairing** the bugs in the Benchmark;
- Point out **important challenges** to produce a relevant Benchmark for mobile APR based on open source projects.

Methodology

- Selecting projects/Apps
  - F-Droid, GitHub, popularity;
  - 360 -> 50 (JUnit usage only);
- Collecting projects’ versions
  - Mining by looking for “bug” and “fix” words in commit messages;
  - Collected current and previous version;
- Running test cases
  - Test suite from the “fixed version”;
  - At least one negative test case becomes positive;

Benchmark DroidBugs

- Contains only projects compatible with Astor4Android;
- compound of 13 bugs from 5 projects
  - Wikipedia;
  - Kore;
  - Poet-Assistant;
  - Habit;
  - K9.

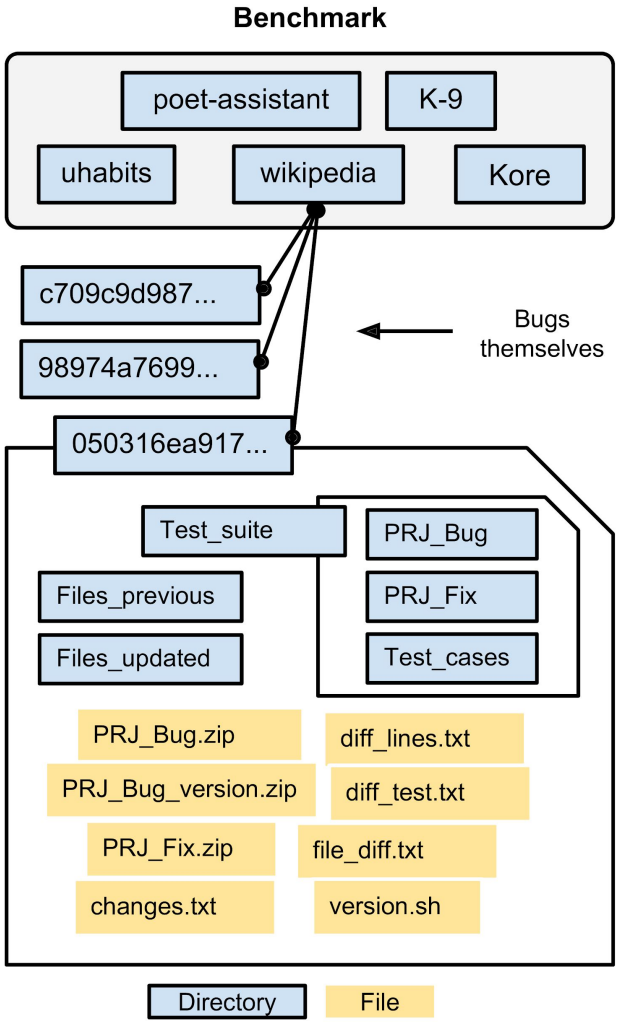
DroidBugs | Apps details

TABLE I. PROJECTS SELECTED TO COMPOUND THE BENCHMARK

Project	Category	Downloads	LOC	Test cases
Wikipedia Android	Books and References	10,000,000	197,569	446
K-9 Mail	Communication	5,000,000	208,785	1351
Kore	Play and edit videos	1,000,000	401,950	131
Loop Habit Tracker	Productivity	1,000,000	103,890	286
Poet Assistant	Books and References	100,000	82,962	128

DroidBugs

Repository's structure



DroidBugs | stats

TABLE II. BUGS, TEST CASES AND VERSIONS OF EACH PROJECT IN DROIDBUGS. "I" AND "L" REPRESENT INSTRUMENTATION AND LOCAL TEST CASE, RESPECTIVELY

Project	Versions	Total of Bugs	Tests cases (I/L)
Wikipedia Android	389	6	0/6
K-9 Mail	87	3	2/1
Kore	4	2	2/0
Poet Assistant	12	1	1/0
Loop Habit Tracker	7	1	1/0

Experiments

- Astor4Android was run for each Bug in DroidBugs;
  - Two FL techniques, Ochiai and Tarantula, were used;
  - JGenProg, JKali and JMutRepai were executed;
  - None of the three algorithms produced fixies.

TABLE III. RESULTS OF FAULT LOCALIZATION WITH OCHIAI AND TARANTULA OVER ALL BUGGY VERSIONS

Formula	acc@1	acc@3	acc@5	wef	bugs not found
Ochai	4	4	5	677.53	3
Tarantula	4	4	5	839.61	5

Discussions

- Filters limited the number of candidate applications
  - Open source, versioning control, test suite etc.
- Missing evolution of the test suite;
- Compilation errors in buggy version due to additions of classes or methods in the fix version;

Conclusions

- DroidBugs was introduced as the first Benchmark for APR in the context of mobile development;
- Difficulties were found on mining repositories and identifying actual bugs;
- Experiment demonstrated how relevant is a Benchmark for this context.

Questions?

Thanks!