How Developers Debug Software
The DbgBench Dataset

www.bit.do/dbgbench

Marcel Böhme, Ezekiel O. Soremekun, Sudipta Chattopadhyay, Emamurho Ughergughe, and Andreas Zeller

Experiment

Main Objective

- to develop a **dataset** that allows **to evaluate** novel debugging **techniques** w.r.t. **humans.**
- How do **developers explain** the bug?
  - Which **fault locations** do experts point to?
  - Do developers **agree** on a single explanation?
- How do **developers patch** the bug?
  - Do **patch** and **fault locations overlap**?
  - How many human-generated **patches** are **plausible but incorrect**?

Demography

27 real bugs
with simplified bug report and test cases

12 software professionals
11 developers + 1 researcher

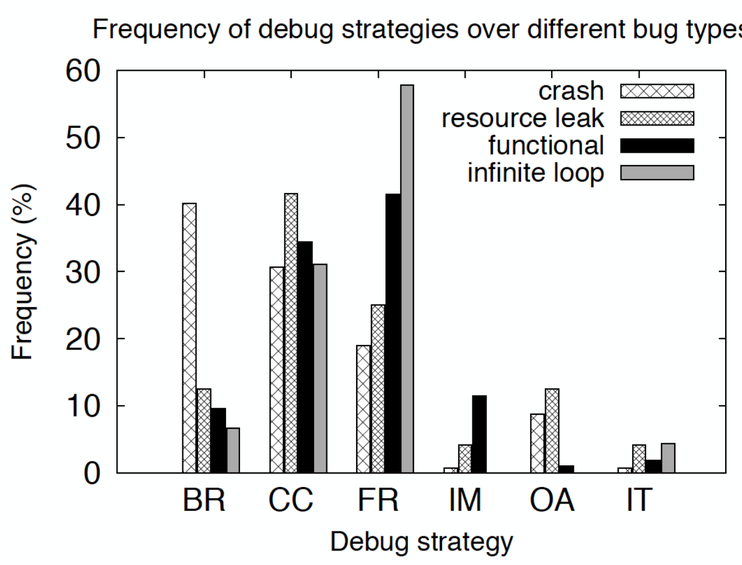
07 plus years experience

06 countries
Russia, India, Slovenia, Spain, Canada, Ukraine

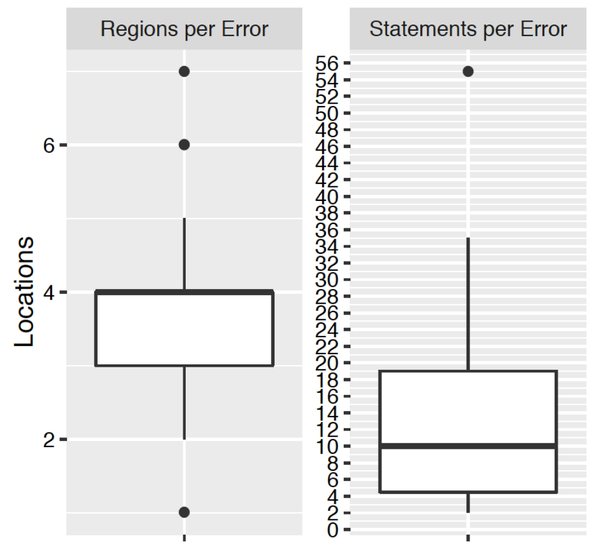
02 Open Source Software
GNU Grep, GNU Find, each with ~17KLOC

29 working days spent debugging these bugs
About 27 hours per developer

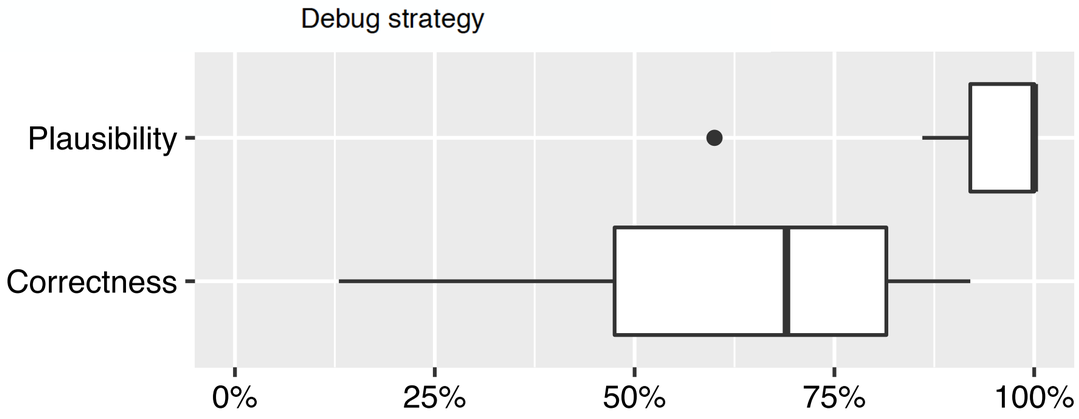
Evaluating debugging aids is difficult and time consuming.

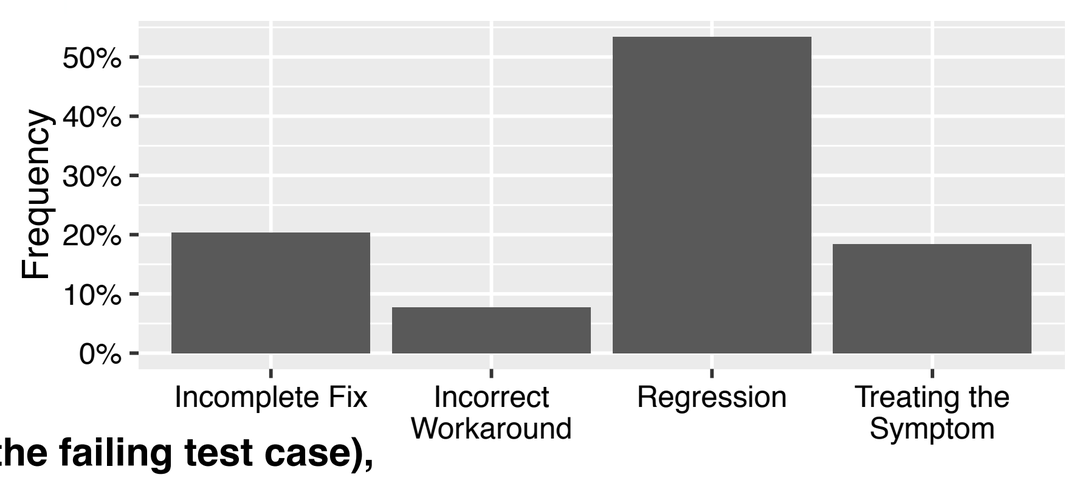


Debugging Strategies
BR - Backward Reasoning
CC - Code Comprehension
FR - Forward Reasoning
IM - Input Manipulation
OA - Offline Analysis
IT - Intuition



For a **single bug**, developers identify **4-20 faulty stmts** in **3-4 distinct regions** distributed across several functions.





While for most bugs *all* submitted patches are *plausible* (i.e., pass the failing test case), for most bugs *30%* of patches are *incorrect* (i.e., fail the code review)!

DbgBench allows for effective evaluation of debugging aids.

Find "-mtime [+-n]" is broken (behaves as "-mtime n")

```
Lets say we created 1 file each day in the last 3 days:
$ mkdir tmp
$ touch tmp/a -t $(date --date="yesterday" +"%y%m%d%H%M")
$ touch tmp/b -t $(date --date="2 days ago" +"%y%m%d%H%M")
$ touch tmp/c -t $(date --date="3 days ago" +"%y%m%d%H%M")

Running a search for files younger than 2 days, we expect
$ ./find tmp -mtime -2
tmp
tmp/a

However, with the current grep-version, I get
$ ./find tmp -mtime -2
tmp/b

Results are the same if I replace -n with +n, or just n.
```

(a) Bug Report and Test Case

If `find` is set to print files that are strictly younger than n days (-mtime -n), it will instead print files that are exactly n days old. The function get_comp_type actually increments the argument pointer timearg (`parser.c:3175`). So, when the function is called the first time (`parser.c:3109`), timearg still points to '-'. However, when it is called the second time (`parser.c:3038`), timearg already points to 'n' such that it is incorrectly classified as COMP_EQ (`parser.c:3178`; exactly n days).

(b) Bug diagnosis and Fault Locations

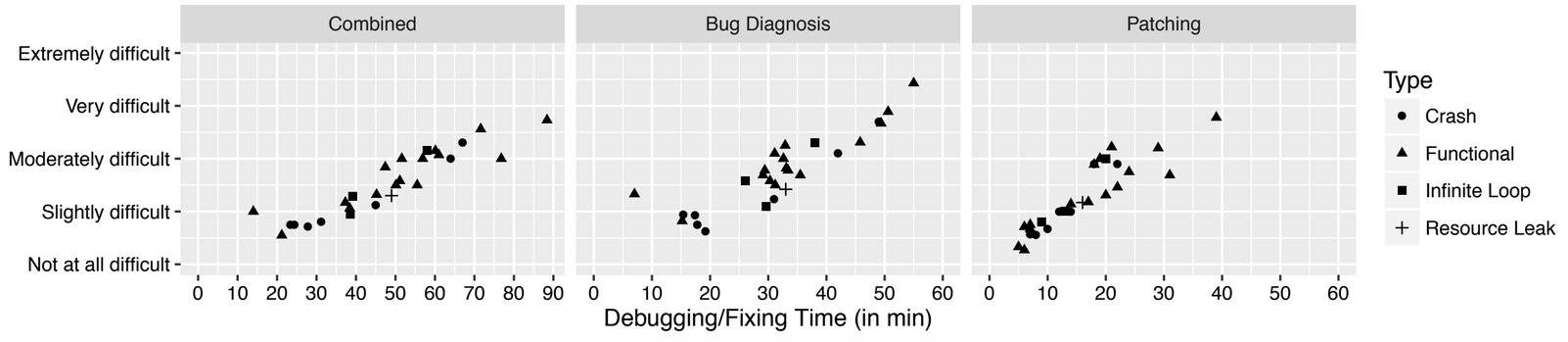
Example Correct Patches

- Copy timearg and restore after first call to get_comp_type.
- Pass a copy of timearg into first call of get_comp_type.
- Pass a copy of timearg into call of get_relative_timestamp.
- Decrement timearg after the first call to get_comp_type.

Example an Incorrect Patch

- Restore timearg only if classified as COMP_LT (*Incomplete Fix* because it does not solve the problem for -mtime +n).

(c) Examples of (in-)correct Patches



DBGBENCH helps to evaluate:

- automated **fault localisation,**
- automated **bug diagnosis,** and
- automated **repair** techniques.

DBGBENCH helps to compare:

- how much **faster** a developer is **diagnosing** and **repairing** a bug using a **novel debugging aid**.

CISPA Center for IT-Security, Privacy and Accountability
SPONSORED BY THE
Federal Ministry of Education and Research
European Research Council
Established by the European Commission
NUS National University of Singapore
UNIVERSITÄT DES SAARLANDES
SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
Established in collaboration with MIT