verifyter

Measuring the Gain of Automatic Debug

Daniel Hansson

CEO

Agenda

- Definition of Automatic Debug
- Tools and Methodologies
- How we measured the gain of Automatic Debug
- Random Tests
- Results

Definition: Regression Failure

- Something that used to work, but has stopped working.
- The failure is caught by test failures, which needs manual debug.
- A lot of engineering hours and project time is spent on regression failures

Test: rx_test_45
Build: build_23
Error: Failuire: Expected RX data 0x4545, but got 0x5555

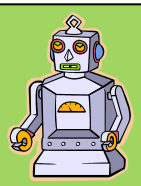
Test: rx_test_51
Build: build_13
Error: Failuire: Expected RX data 0x4545, but got 0x5555

Test: rx_test_54,
Build: build_13
Error: Failuire: No data

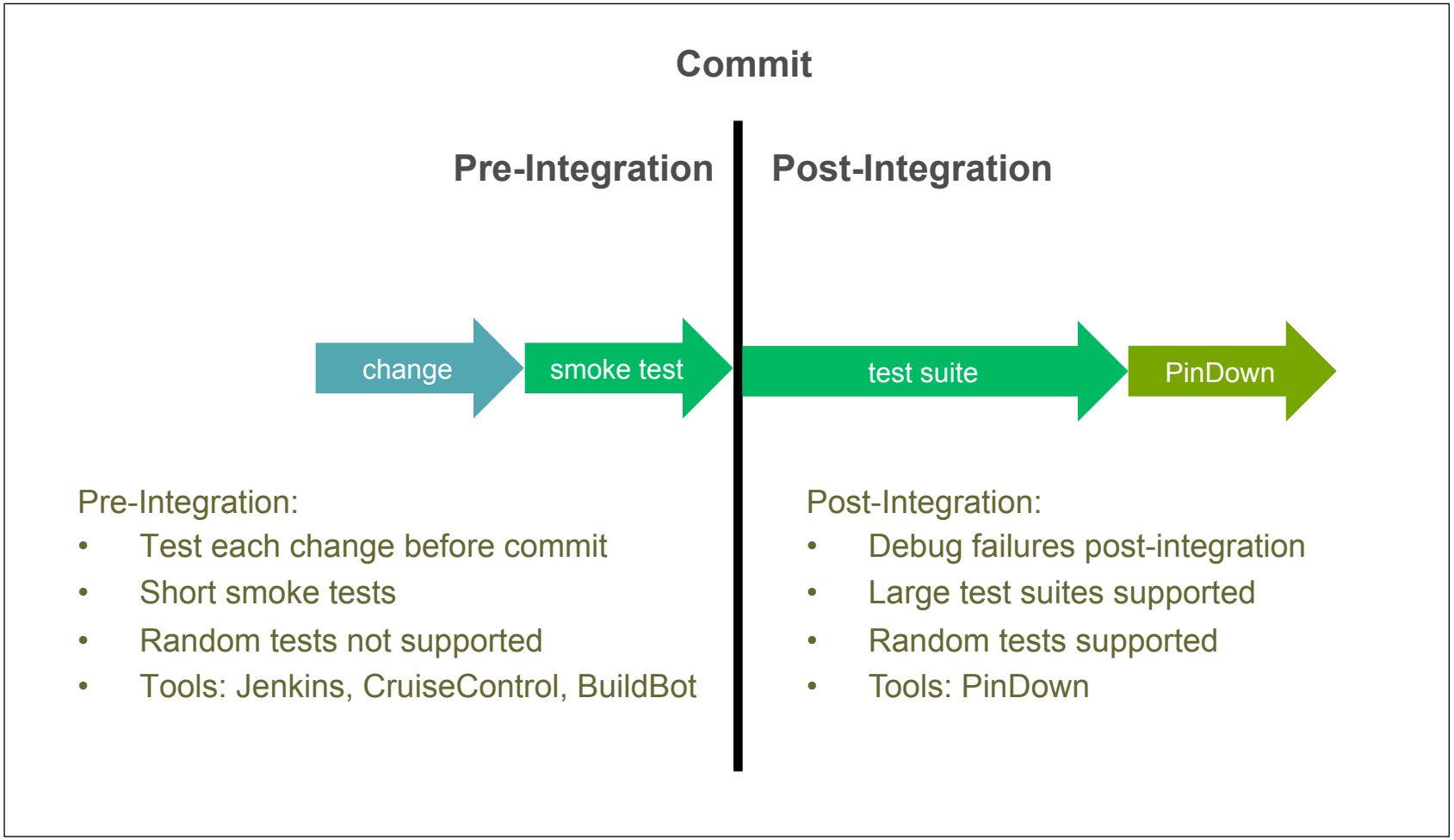
Definition: Automatic Debug of Regression Failures

- Automatically finding the bad commit in the revision control
- Directly assign and report the bug to the responsible committer
- No engineering hours spent, except for bug fixing
- Active 24/7 – bugs are pushed back faster

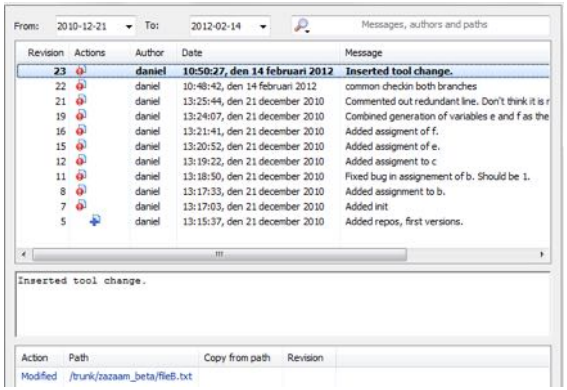
Bug No: 1 (new bug)
Test: rx_test_45 (in total 3 failures)
Build: build_23
Error: Failuire: Expected RX data 0x4545, but got 0x5555
Committer: dhansson
Commit Message:
268243. Changed default value of rx_mode_reg
Committed Files:
//dwarc/Hardware/ASICproject/main/dev/productFast/hardware/rx_main.v#4



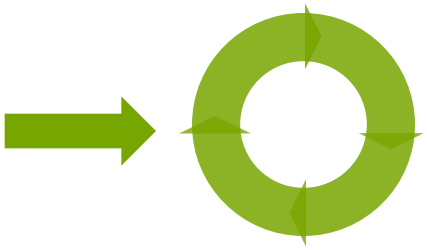
Tools And Methodologies



PinDown™



Version Control System

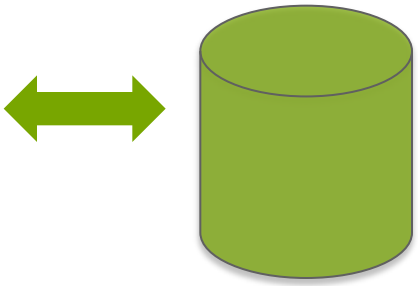


PinDown Job

Test Executor
(LSF Farm)

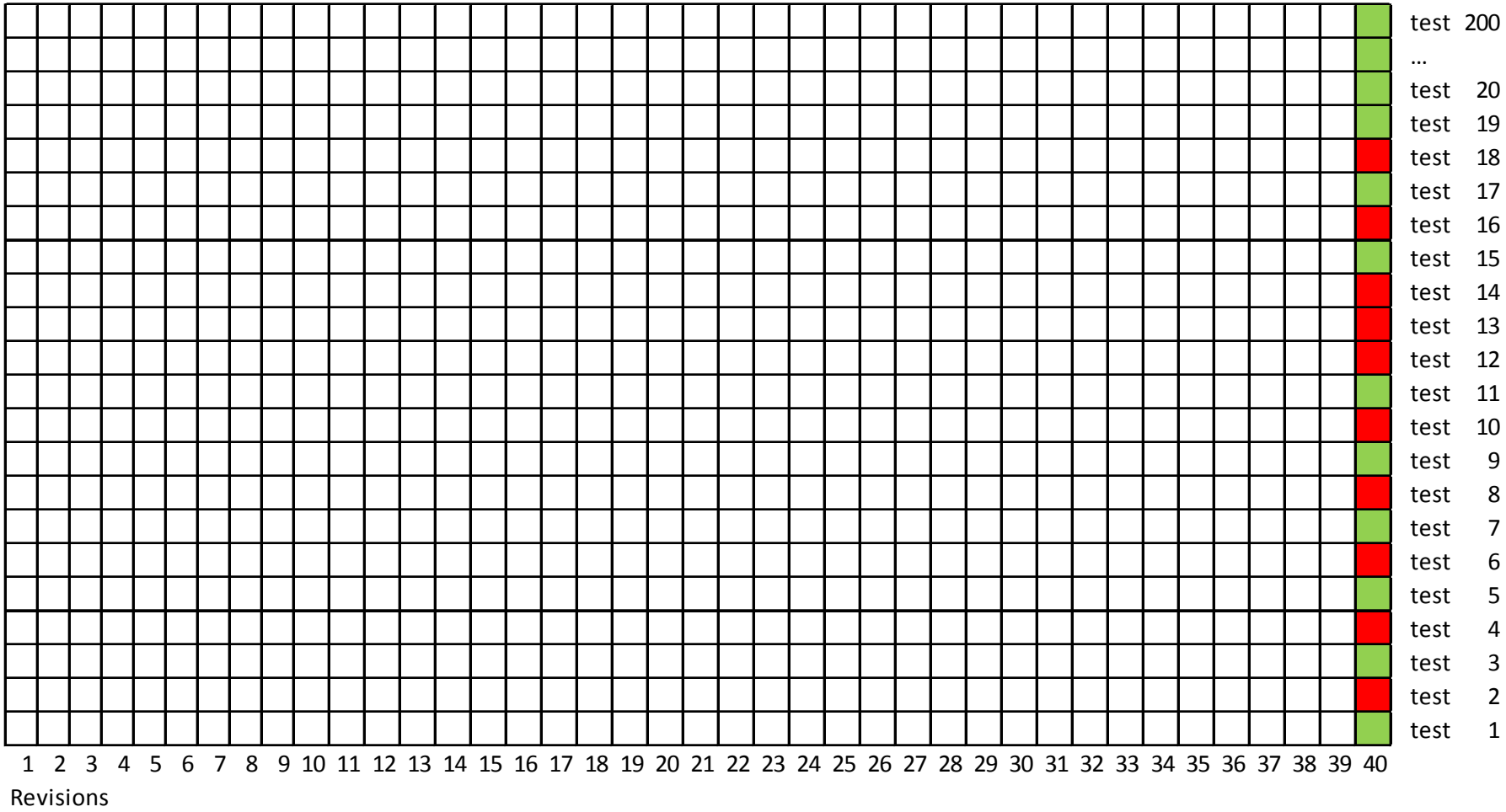
Start Job

PinDown TestHub



PinDown Results Database

Automatic Debug

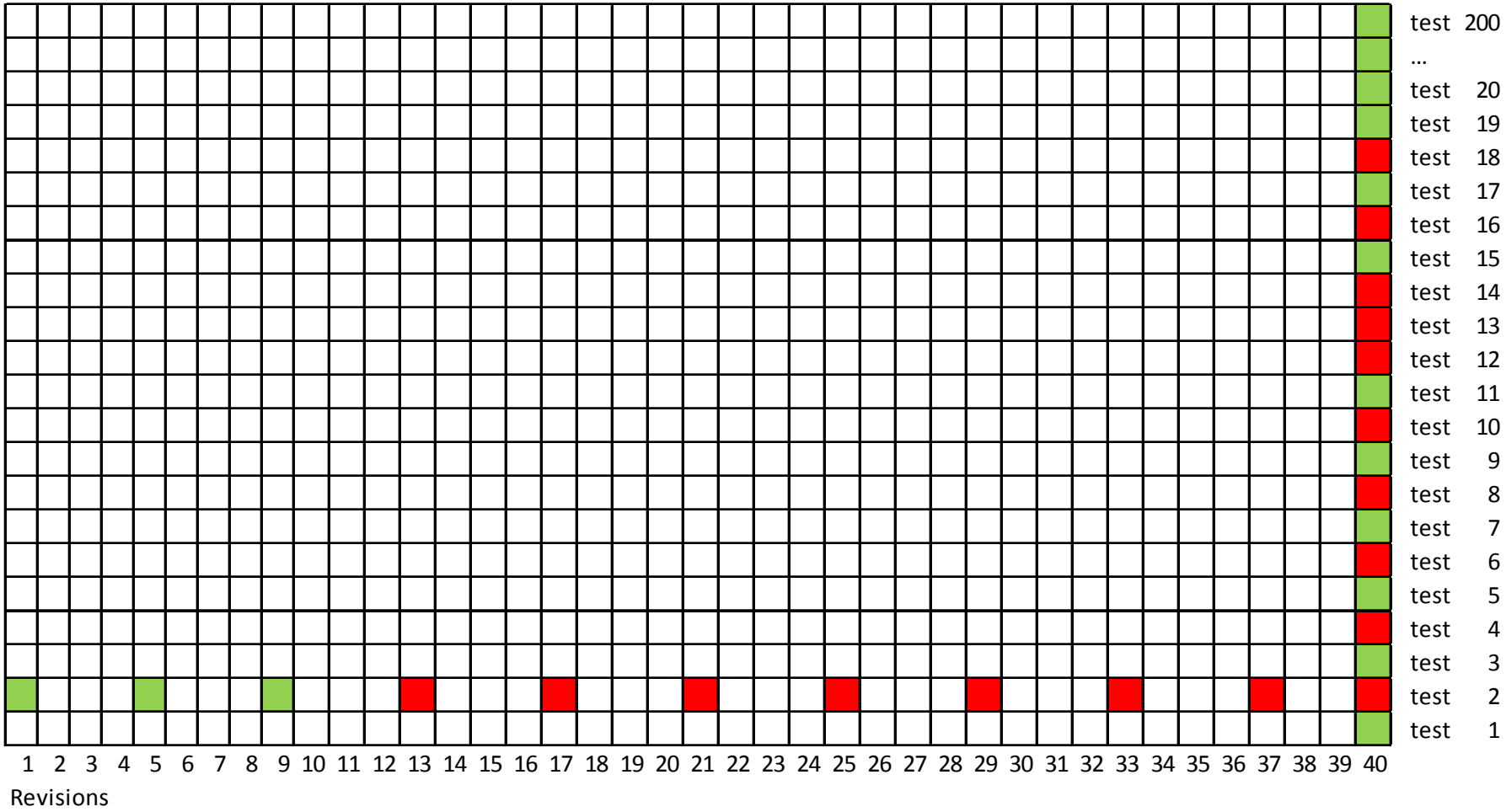


Starting point: 200 tests run, 10 failed.

Each test takes 1 hour. Max allowed debug jobs on the farm: 10.

Patent Pending

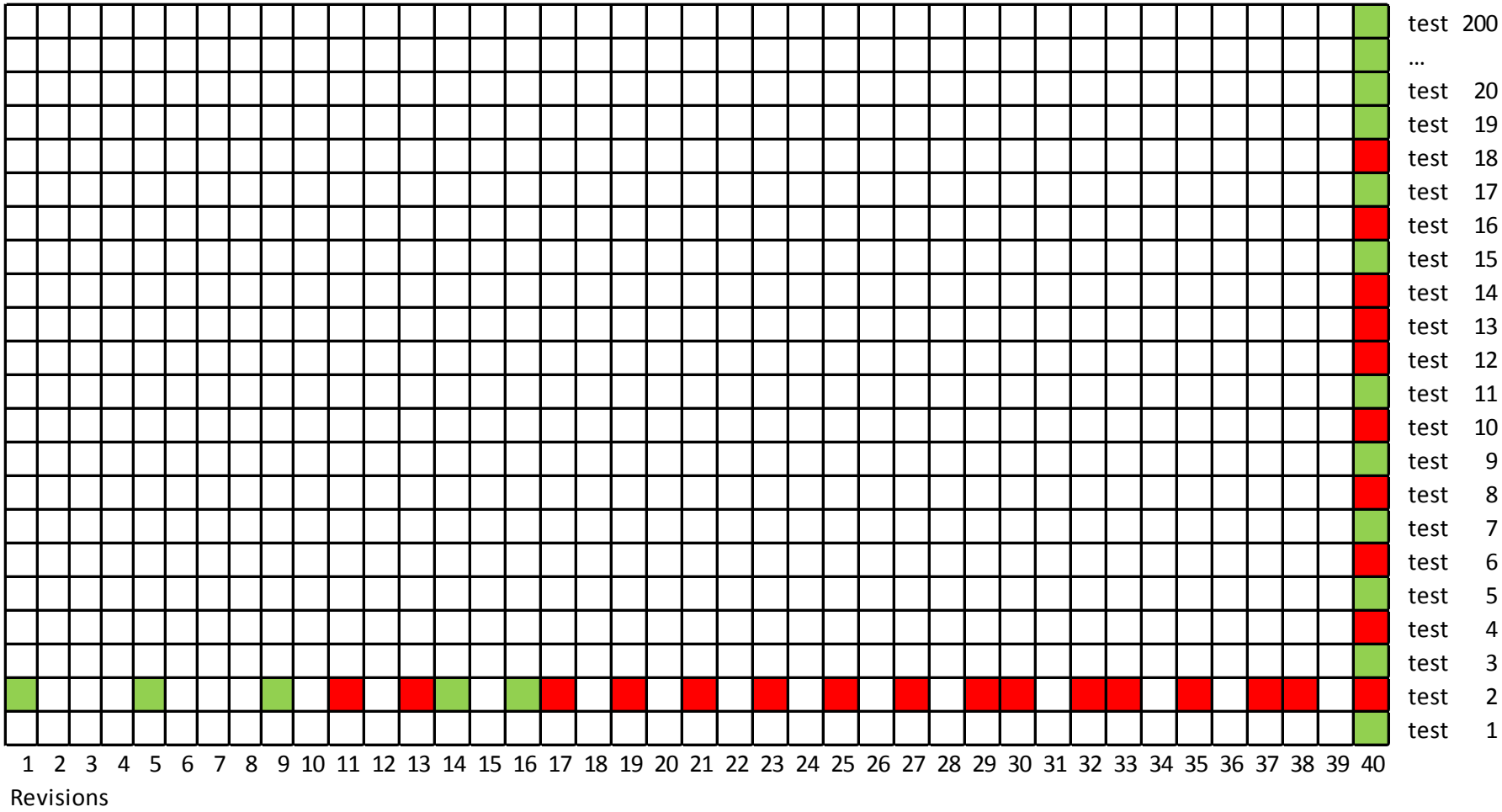
Automatic Debug



Select the pilot test, i.e. the fastest test. Test it on 10 revisions in parallel.
Found a pass-fail transition (rev 9 and 13), but there may be more transitions,

Patent Pending

Automatic Debug

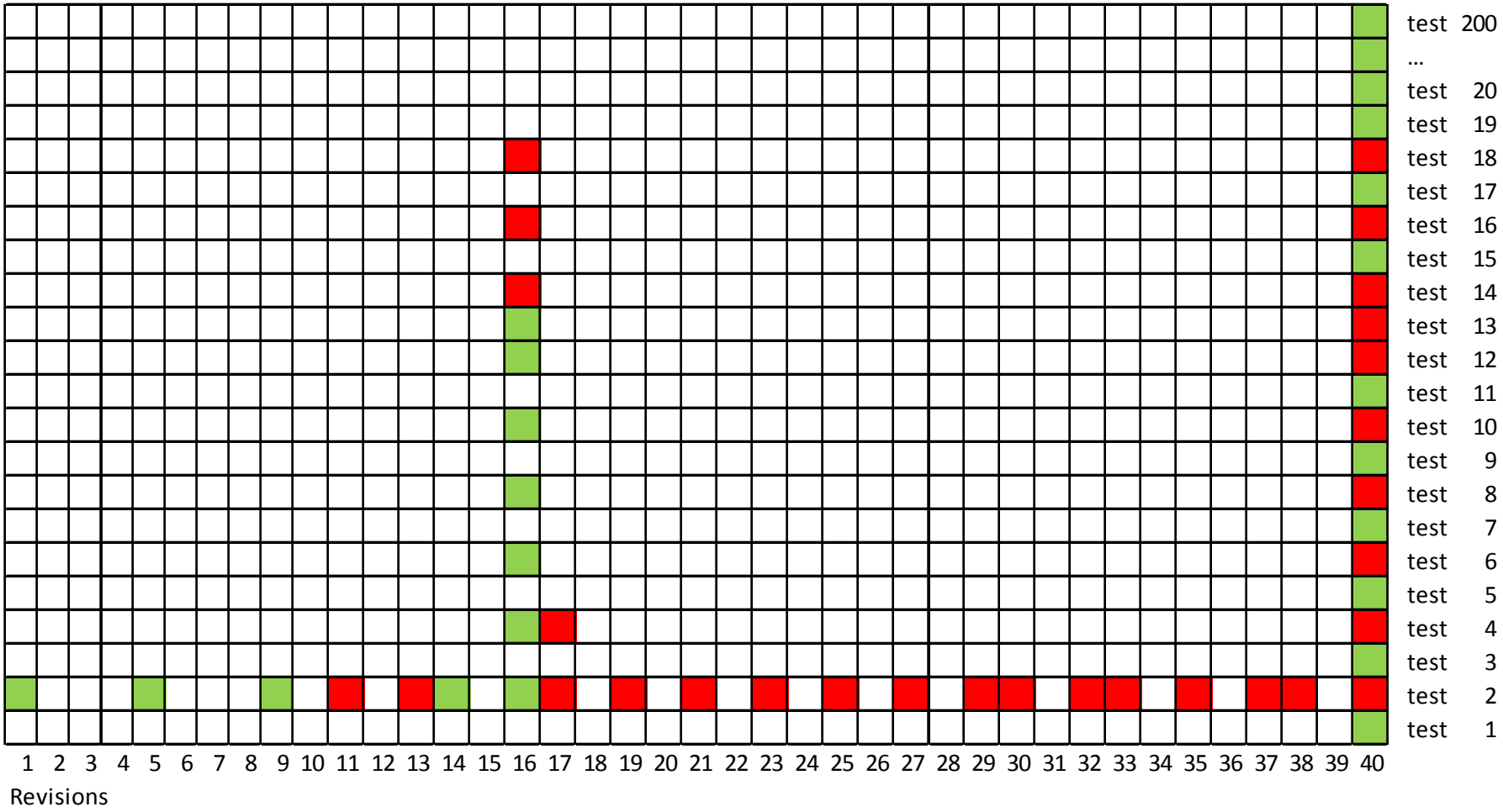


Test another 10 revisions to get high probability in debug analysis.

Found first faulty rev (17). Bug report emailed to committer after 2 hours debug time.

Patent Pending

Automatic Debug

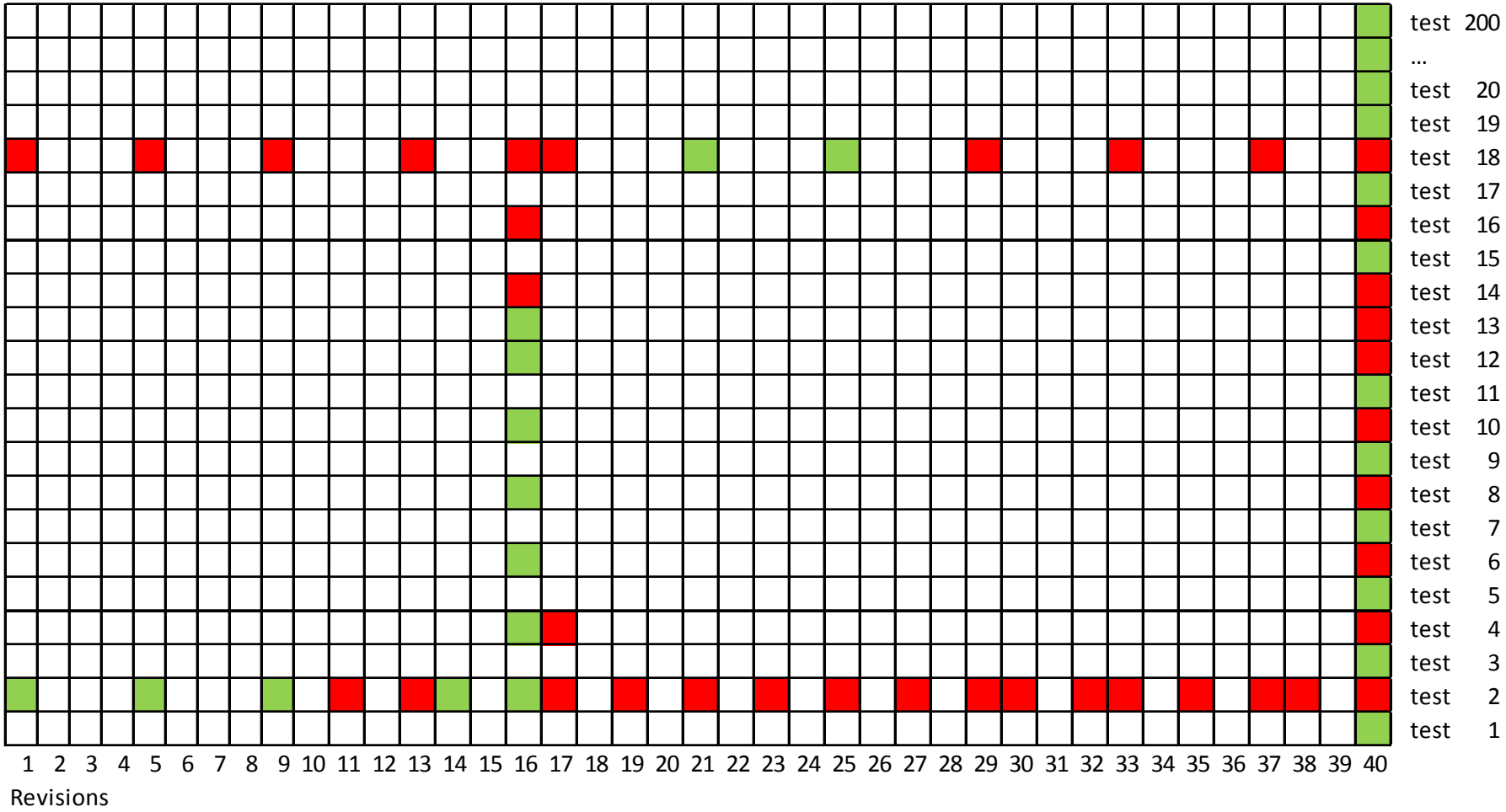


Check if the other failing tests have the same tipping point. Start with the pass.

The result: 3 tests don't. They are due to a second bug.

Patent Pending

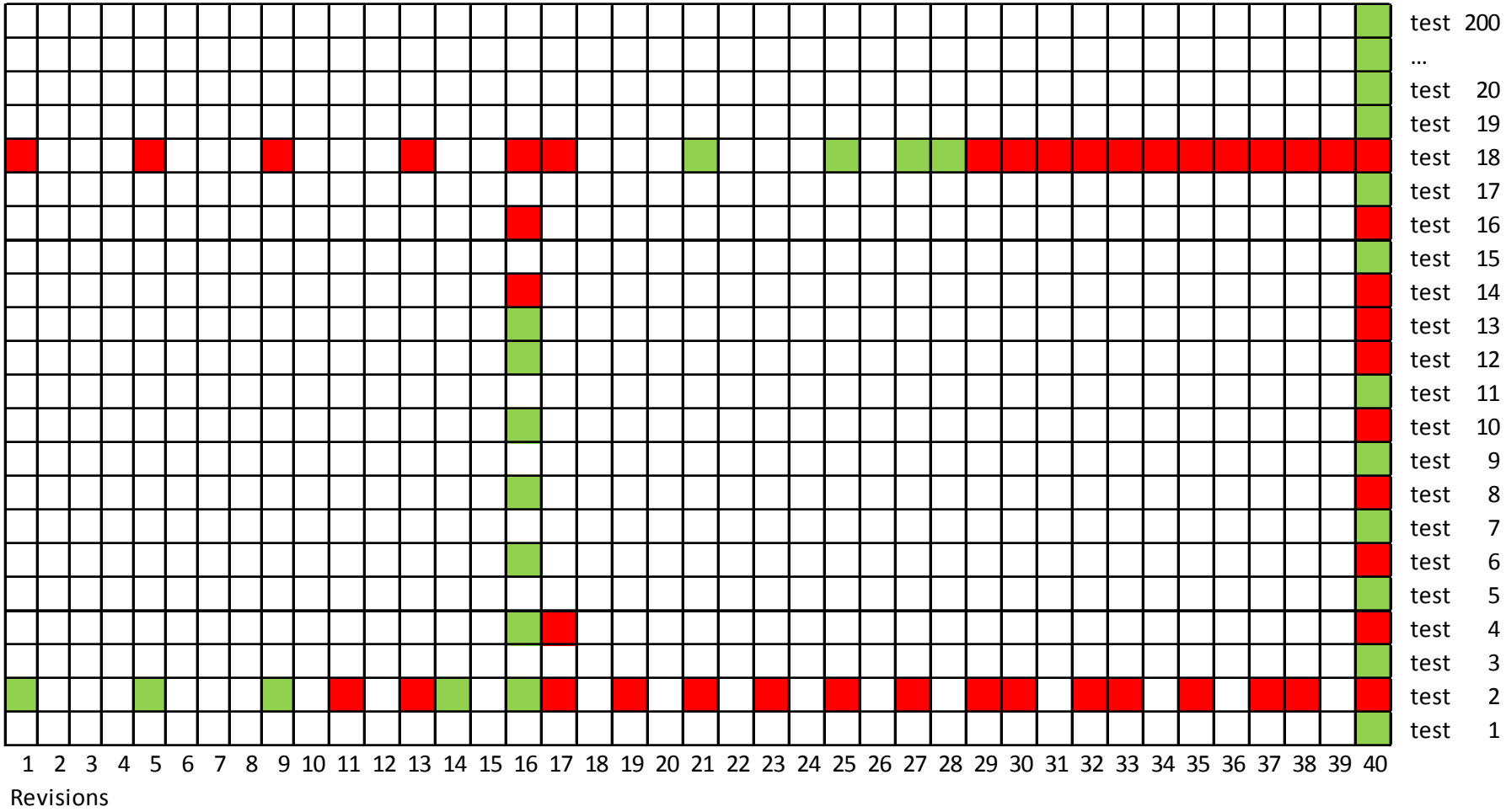
Automatic Debug



Pick the pilot test for the second bug and run on 10 revisions

Patent Pending

Automatic Debug

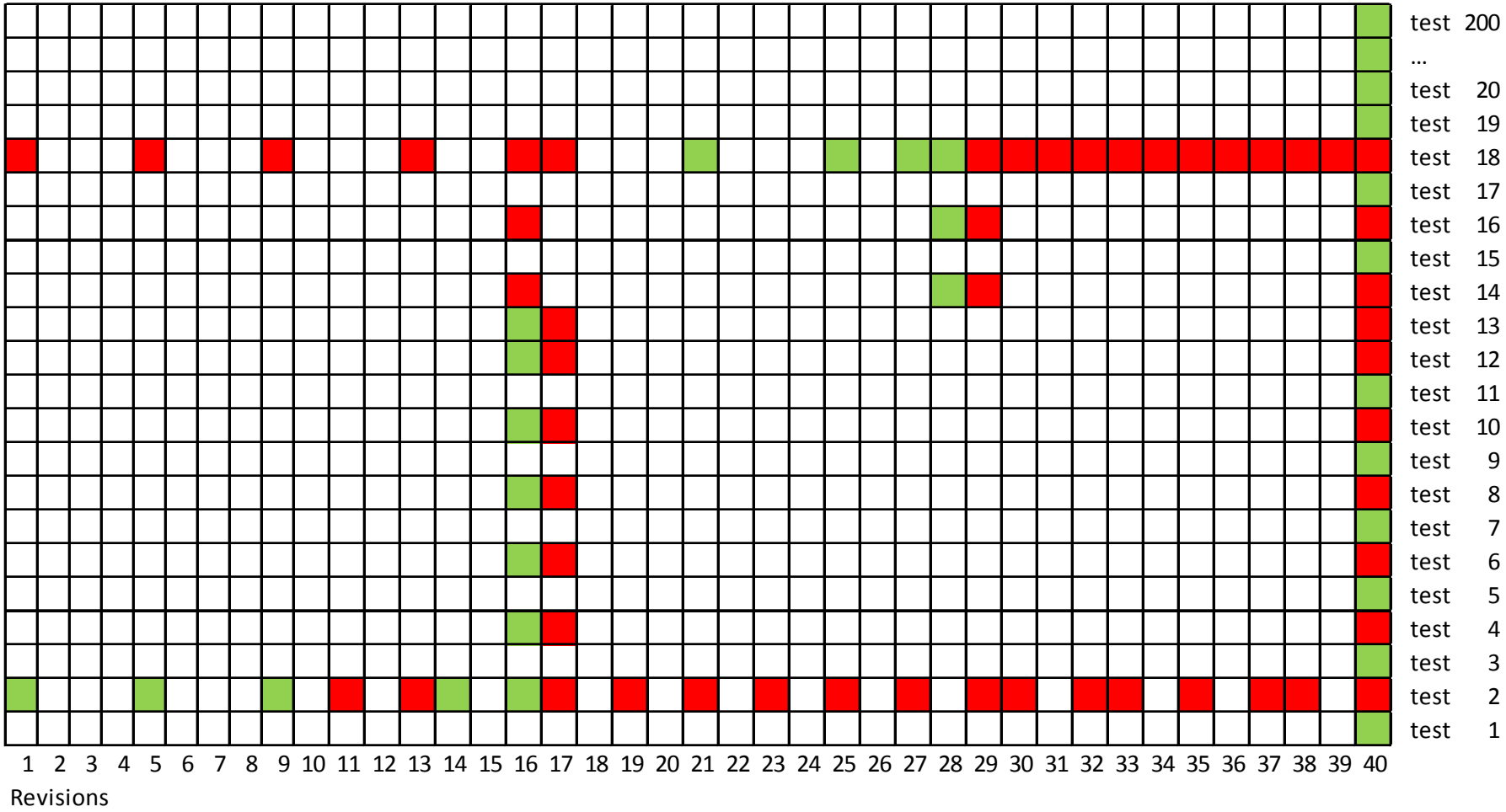


After the second run of the pilot test for bug no 2 we find the tipping point

Bug no 2 emailed to committer after 5 hours

Patent Pending

Automatic Debug



Checking if all remaining tests match the tipping points of the two reported bugs

They do. Debug run completes after 6 hours.

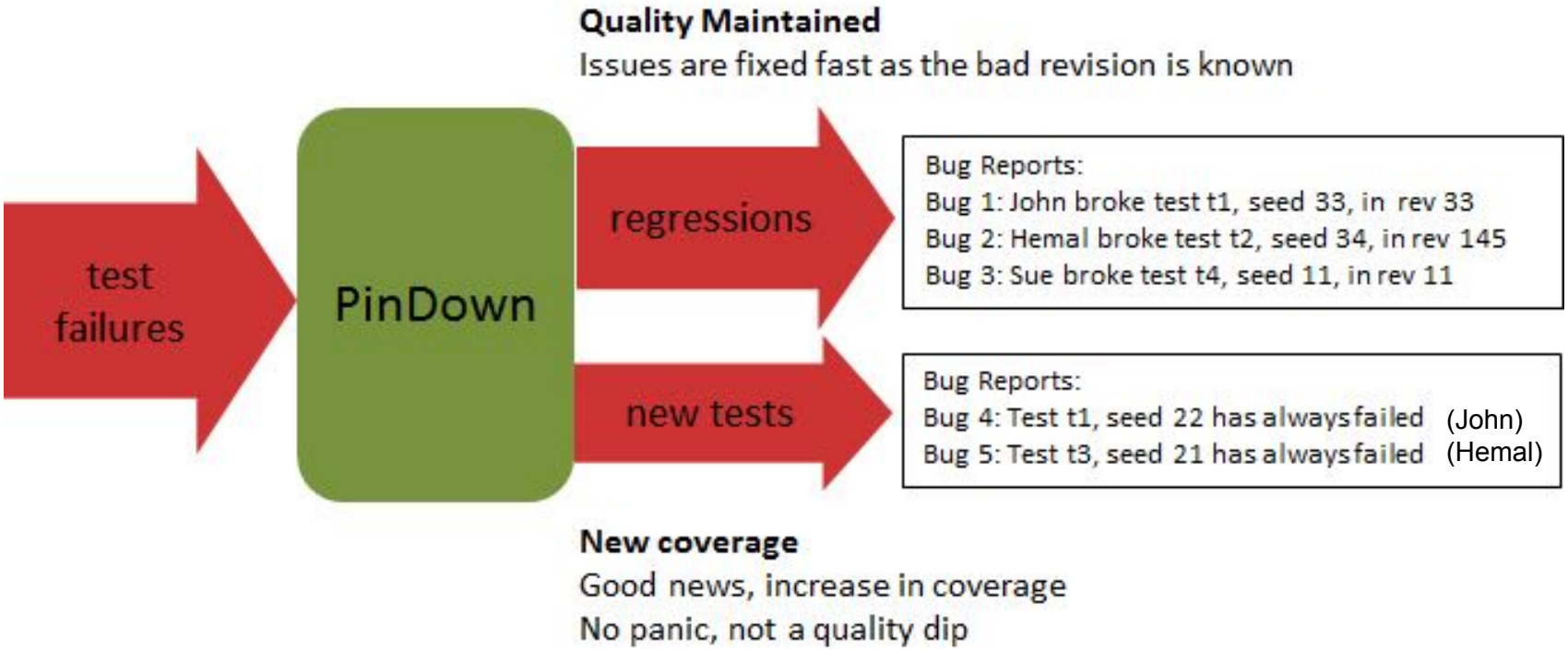
Patent Pending

Random Test Setup

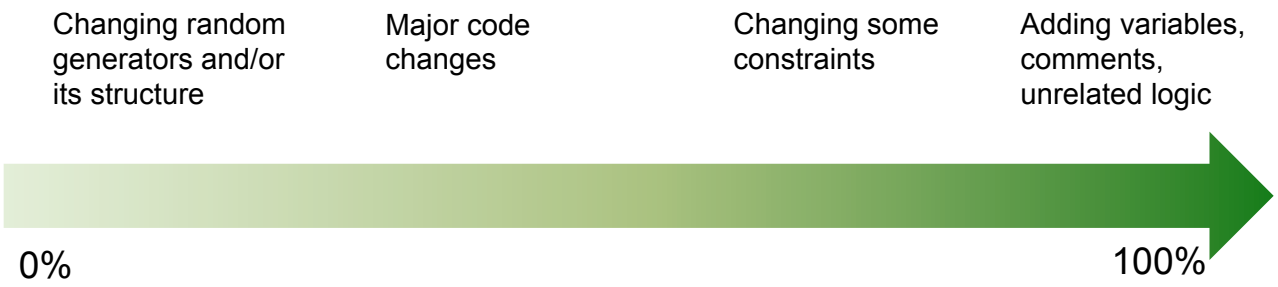
Regression Testing with Random tests cannot separate a regression from new coverage

- Inefficient debugging of regressions
- Unable to separate bad news (regressions) from good news (new coverage)

PinDown solves this:



Random Stability



Random stability is depends on the type of testbench change

- PinDown may sometimes not be able to reproduce the issue
  - Less of a problem when frequent testing

- PinDown allows a setup choice: Include/Not include Testbench

Measuring how fast bugs were fixed

- Real commercial ASIC project:
  - Based on random testing
  - We analyzed 39 regression bugs found during 3 months of the project
- PinDown was setup to track the following in the revision control system:

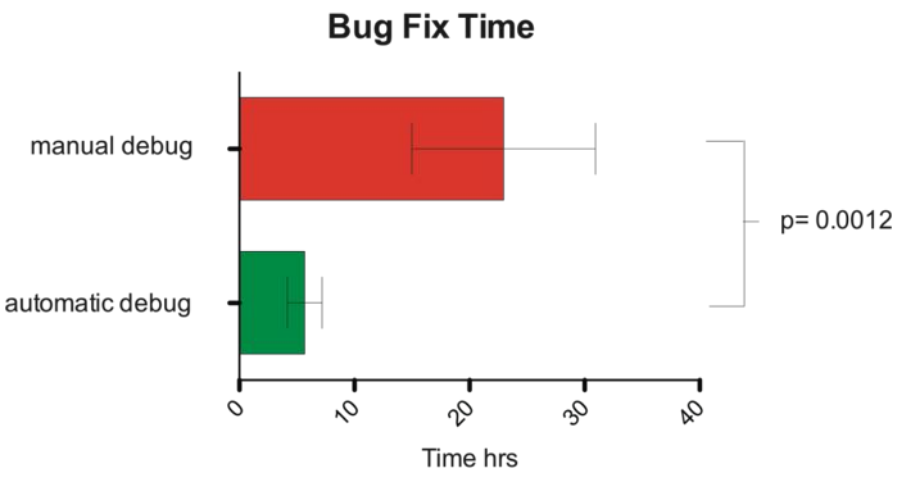
Tracked:
- Device under test (DUT)
- Testbench
- Scripts and tools for build and test

Not Tracked:
- Some internally developed tools and scripts (they were under revision control, but not tracked by PinDown)

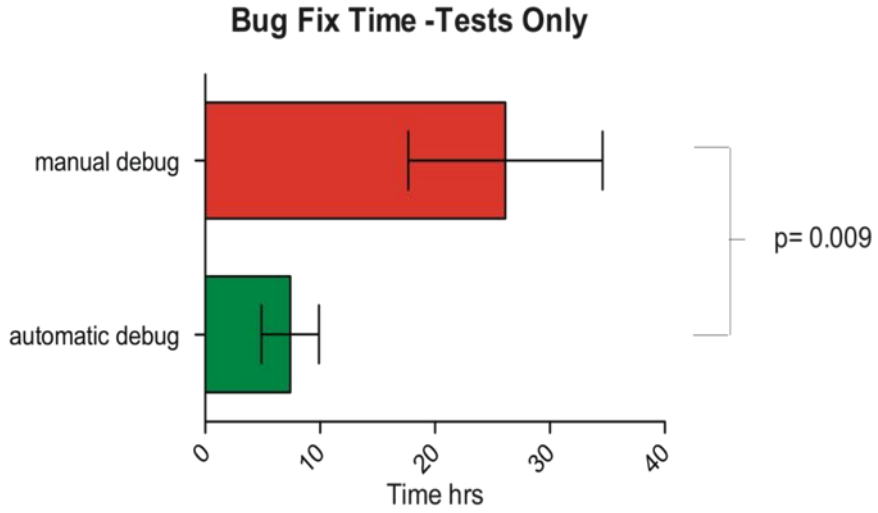
- We measure the time from when the report is emailed out until the fix is committed:

- In both cases PinDown is emailing out the report, but automatic debug is only enabled for the areas that PinDown is setup to track
- No difference in complexity between tracked and non-tracked areas.
  - Complex bugs were fixed by simply removing the commit.

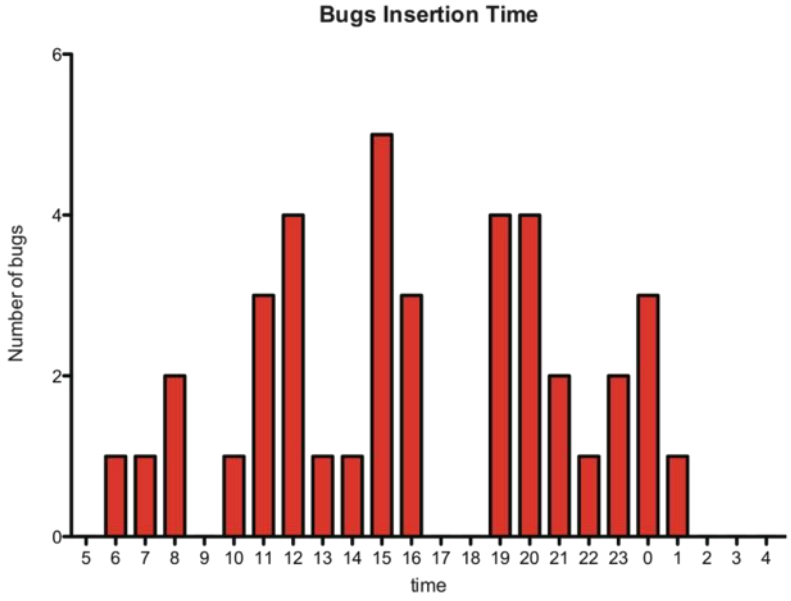
Results: 400% faster bug fixing



(5,7h for automatic debug, 23h for manual debug)



(7,4h for automatic debug, 26,1h for manual debug)



Average insertion time for bugs: 4 pm

5x less discussions
(0,5 emails automatic debug, 2,6 emails manual debug)

verifyter

Measuring the Gain of Automatic Debug Summary

✓ Automatic Debug speed up bug fixing by 400% in this ASIC project

How was that achieved?

✓ Precision and clear responsibility
- The bug report shows who did what and when
- This in turn lead to 5x less discussion prior to bug fixing

✓ Automatic Debug is active 24/7
- Bugs occur late in the day, leaving little time for same-day manual debug

verifyter

For more info:

info@verifyter.com

www.verifyter.com