Automating Qt GUI Tests
10 Pitfalls And How To Avoid Them

Qt Developer Days 2012

by Reginald Stadlbauer

About me

- Name: Reginald Stadlbauer
- Company: froglogic GmbH, vendor of Squish for Qt and Squish Coco
- Position: co-founder and CEO
- Worked as Software Engineer at Trolltech and the KDE project

Overview

- Types of Testing
- Why Automate?
- Pitfalls 1-10
- Demo based on “Rohde & Schwarz PowerViewerPlus”

Types of Testing

- Unit Testing
- Performance Testing
- ...
- Functional GUI Testing
  - Black/Gray Box Testing
  - Assume user's point of view
  - Automate to spot regressions
  - Combinable with profiling, coverage and other analysis and monitoring tools

Why Automate?

- Faster
  - Get results quicker
  - Run more tests in the same time
- Trivial to replay in different configurations
- Reliable, reproducible and repeatable
- Relieve testers from monotonous tasks

But...

- Automating GUI tests is not trivial
- Following best practices is vital for the success of automated GUI tests

1. Rely on capture and replay

- Produces massive test scripts
- Not readable
- Not maintainable
- No code re-use possible
- Brittle against changes in the UI

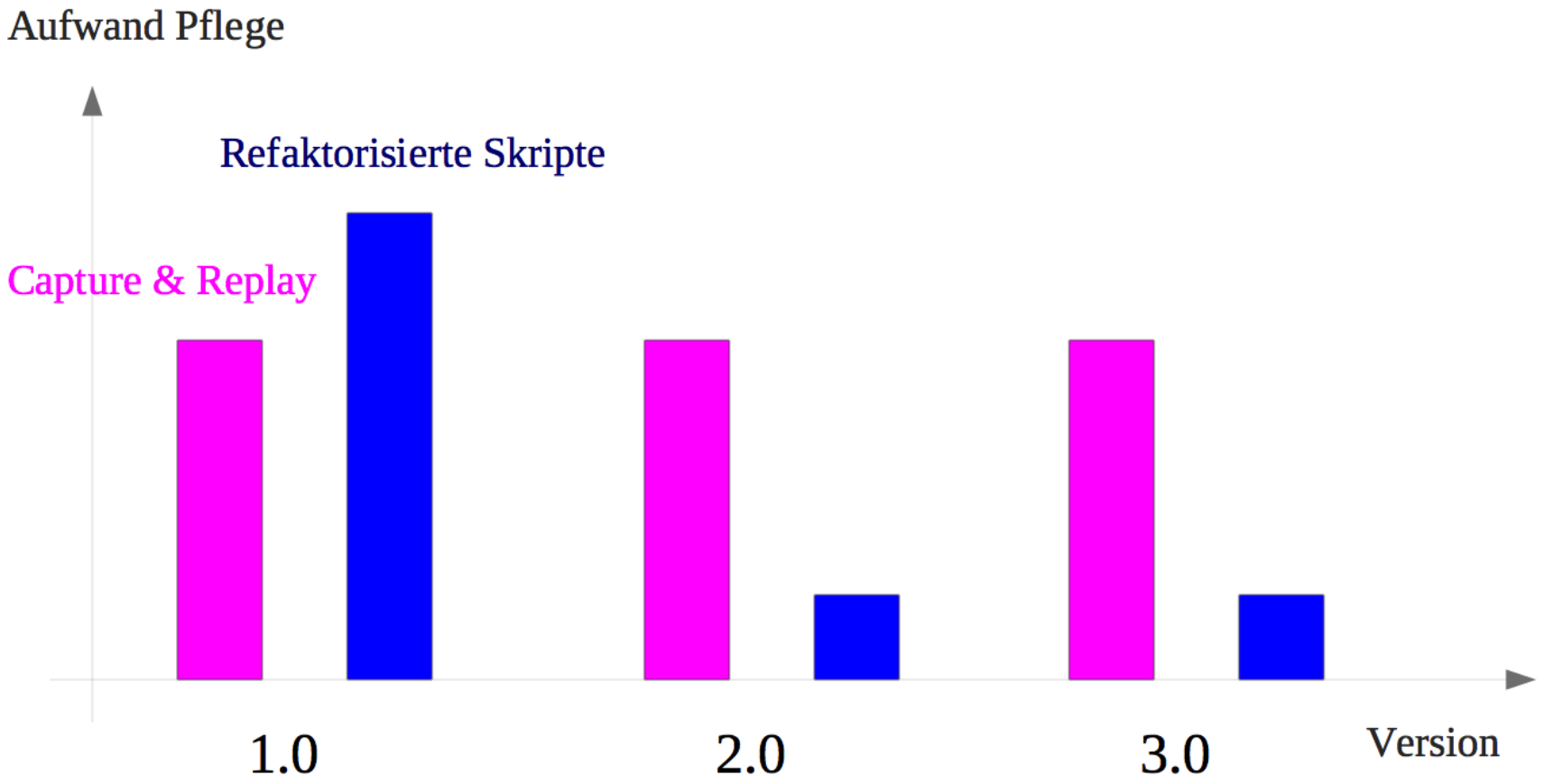
- Solution: Scripting & Refactoring

2. Use primitive macro language

- Limited to small set of features
- No way to “break out”
- No way to utilize 3rd party libraries (database access, etc.)
- No way to deal with dynamic tests

- Solution: Use scripting solution for test automation

Scripted Approach vs. Capture & Replay



3. Rely on screen coordinates

- Addresses screen positions and not UI controls
- Breaks with UI layout changes
- Depends on GUI style and platform
- Scripts hard to understand

- Solution: Address objects based on properties

4. Rely on screen captures / OCR

- No knowledge of GUI controls
- Too much heuristics
- Depends on irrelevant data (colors, fonts, etc.)
- Many incorrect fails / errors

- Solution: Identify on and compare object properties

5. Rely on “Windows” or “Accessibility” test tools

- Only “knows” standard Windows controls
- Cannot drill into Qt / QML / Quick / WebKit controls
- Object identification based on limited amount of properties
- Not cross-platform

- Solution: Use a tool which understands Qt controls

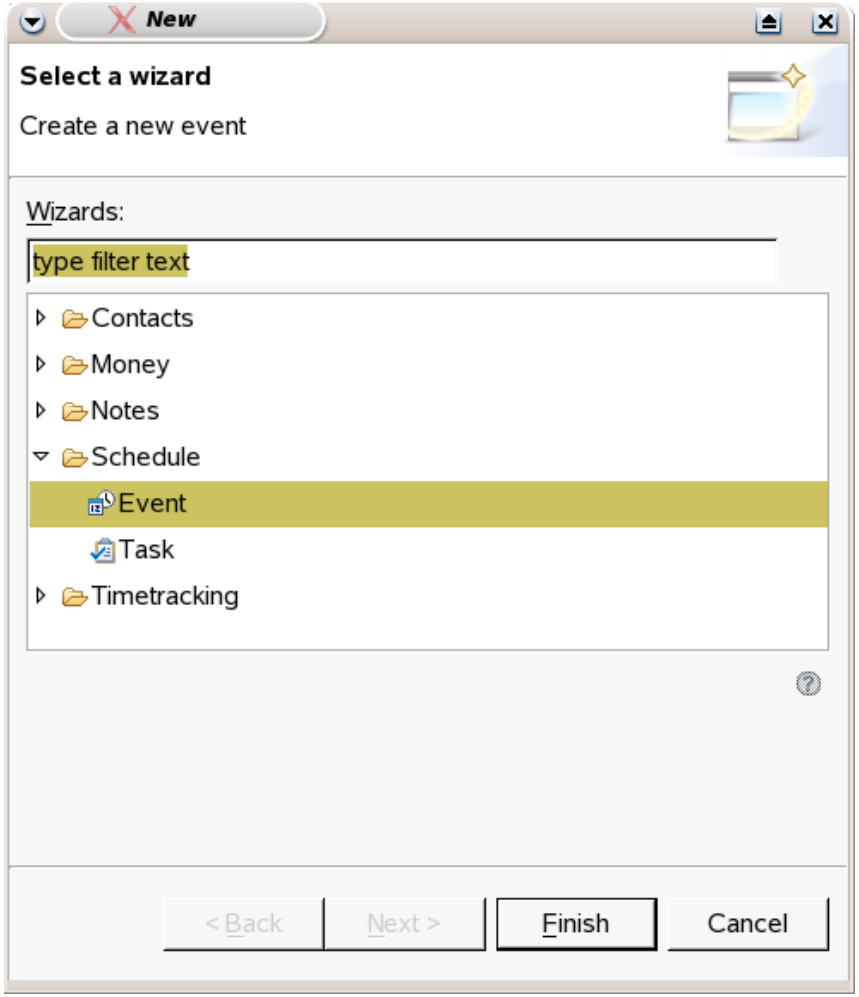
Example: Widget Recognition Options

```
Very BAD:
   MouseClick(132, 367)

BAD:
   MouseClick('Tree', 30, 136)

BAD:
   MouseClick(
     FindObjByImg('item-image.png'))

GOOD:
   ClickItem('Tree', 'Event')
```



6. Tests embedded in application

- Tempting to test API rather than GUI
- Application crash or freeze not handled well
- Can only test one application per test case
- Not suitable for remote testing (embedded devices, mobile)
- Modifies application

- Solution: Run test in a separate process

7. Rely on unique Qt objecName

- Burden for developers
- Not realistically doable if testing is introduced later
- Need uniqueness checking

- Solution: Use multi-property naming

8. Rely on AUT's object hierarchy

- Long and unreadable names
- Relies on application internal “helper widgets”
- Small layout changes breaks naming

- Solution: Use multi-property naming

9. Create tests “on the side”

- Development resources are already restricted
- There is always “one more important dev task”
- Easy to delay “until tomorrow”

- Solution: Dedicated resource for testing

10. Setup automation “when ready”

- Nobody runs the tests and sees the fails/errors
- Tests will become unmaintained and not work anymore
- Tests will be forgotten

- Solution: First task: set up automation, then start creating tests

11. There are more...

- Thinking there are only 10 pitfalls :-)

Squish for Qt Demo on Rohde & Schwarz PVP

froglogic

- Discuss Record & Replay
- Verifications
- Object naming
- Refactoring & Scripting
- Screenshot Verifications
- Keyword driven testing

- Q & A