Testing of embedded and mobile Qt and QML Applications

Qt Developer Days 2013

by Harri Porten

About me

- Name: *Harri Porten*
- Company: *froglogic GmbH*
- Position: *co-founder and CTO*
- Qt usage: *since 1997 (KDE project)*
- Qt development: *Software Engineer at Trolltech*

Overview

- Types of Testing
- Why Automate?
- Challenges on embedded and mobile platforms
- Live demo

Types of Testing

froglogic

- 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
- Typical reason for test effort failures: wrong test approach

Platform Challenge

froglogic

Qt runs on:

- Windows (various versions)
- Linux (desktop and embedded)
- Mac OS X
- Android
- Boot to Qt
- iOS
- QNX
- VxWorks
- Nucleus
- ....

Toolkit Challenge

Those may play a role:

- QWidgets
- QML elements
- Native controls
- Web!

Most challenging: combinations of the above.

Platform Solution 1/2

Biggest help from....

Qt itself

Platform Solution 2/2

Additional help through:

- Resolution independence
- Synchronization methods
- UI abstractions
- Reusable functions/objects
- Mock objects

Virtualization

froglogic

Target hardware

- the real thing
- limited number
- harder to automate

Virtual systems

- VMware, Virtual Box, qemu
- emulator vs. simulator
- easy replication, resets and automation
- Simulation of hardware features, limitations and events.

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

Script Languages

Beware of “vendor scripts” or “macros”!

Open and powerful choices exist:

- JavaScript
- Python
- Perl
- Ruby
- Tcl
- ...

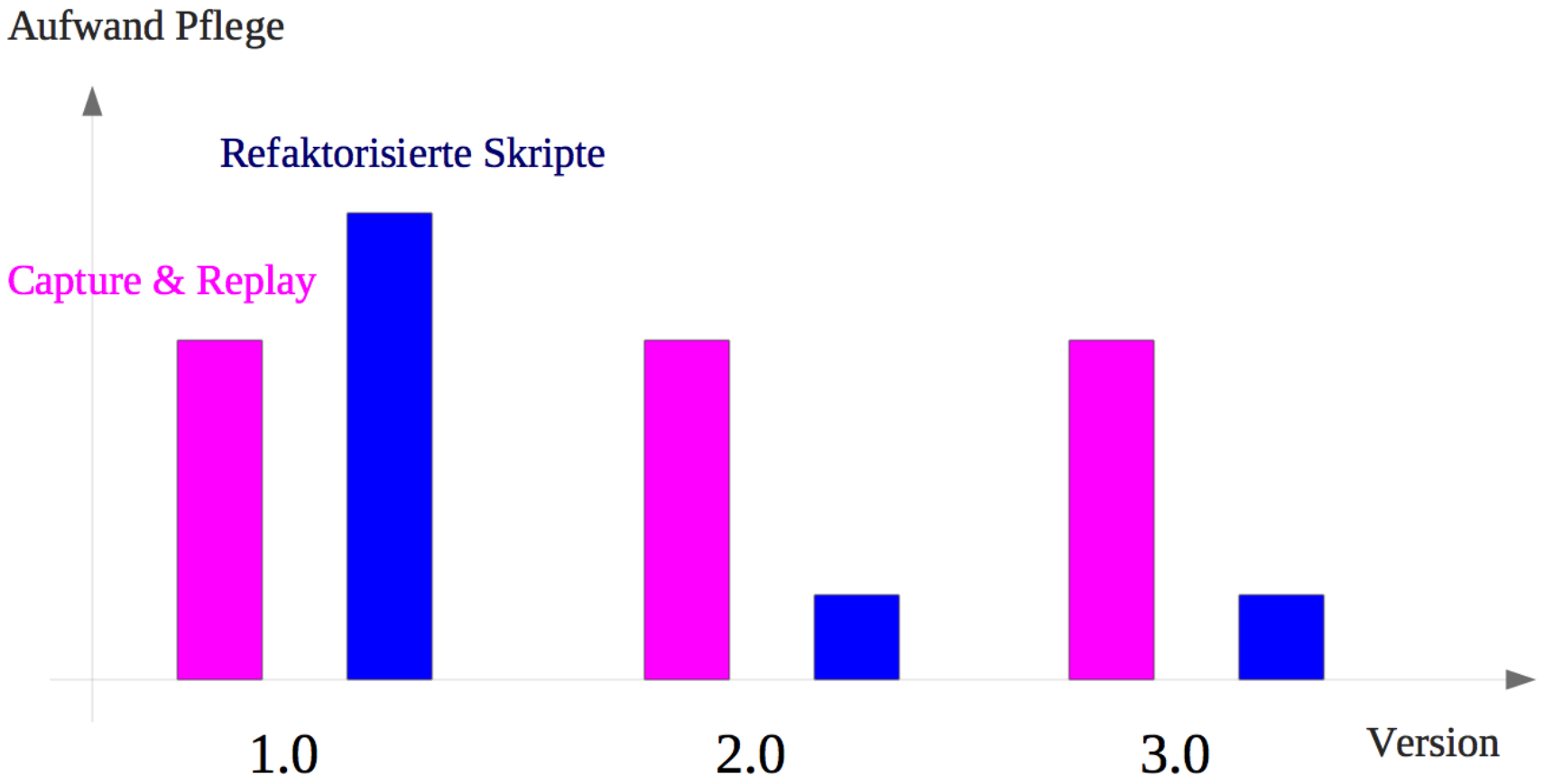
Factorization

```
function main() {
    launchApplication(“clean”);
    loadData(“sample.dat”);
    changeParameter(“ParameterA”, 10);
    runCalculation();
    dumpData(“out.txt”);
    compareData(“out.txt”, “expected.txt”);
}
```

GUI Objects

```
login = LoginScreen()
login.tryLogin(“myuser”, “wrongpassword”)
test.compare(login.success, False)
test.compare(login.message, “Wrong password”)
login.tryLogin(“myuser”, “realpassword”)
test.compare(login.success, True)
```

Scripted Approach vs. Capture & Replay



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

Reliance on screen captures

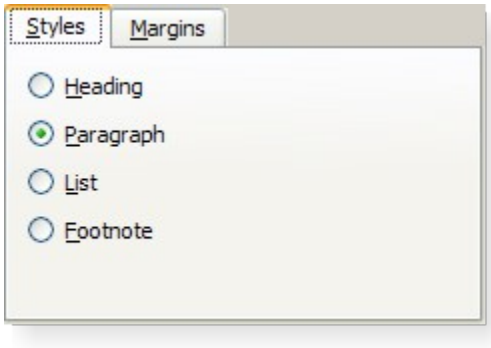
- No knowledge of GUI controls
- Too much heuristics
- Depends on irrelevant data (colors, fonts, etc.)
- Many incorrect fails / errors

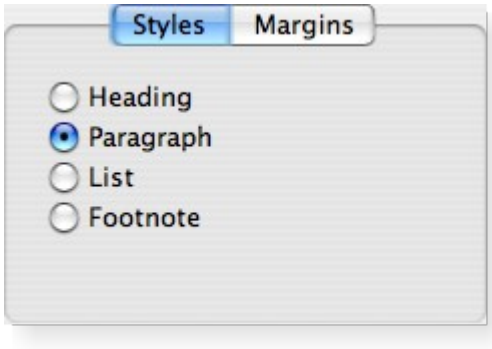
- Solution: Identify on and compare object properties

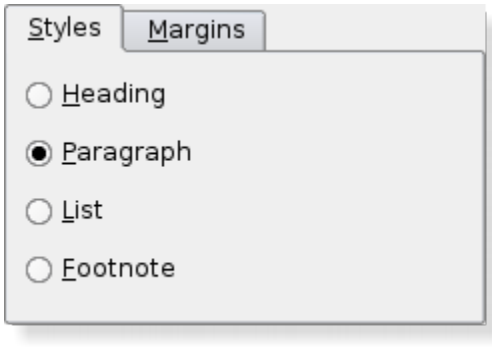
UI Styles

froglogic

Tab Control

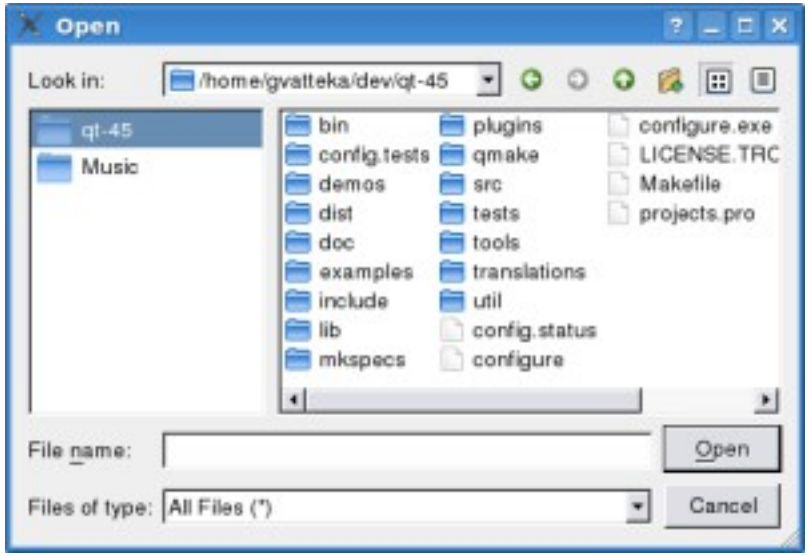


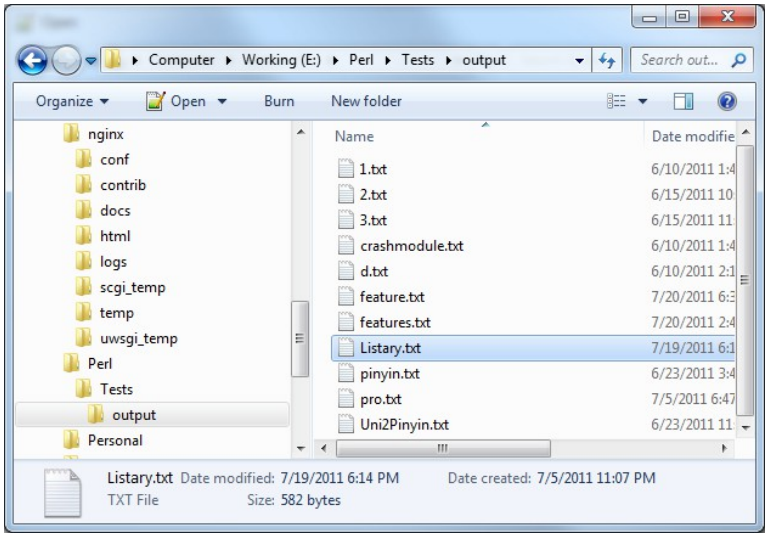




UI Styles

File Selectors

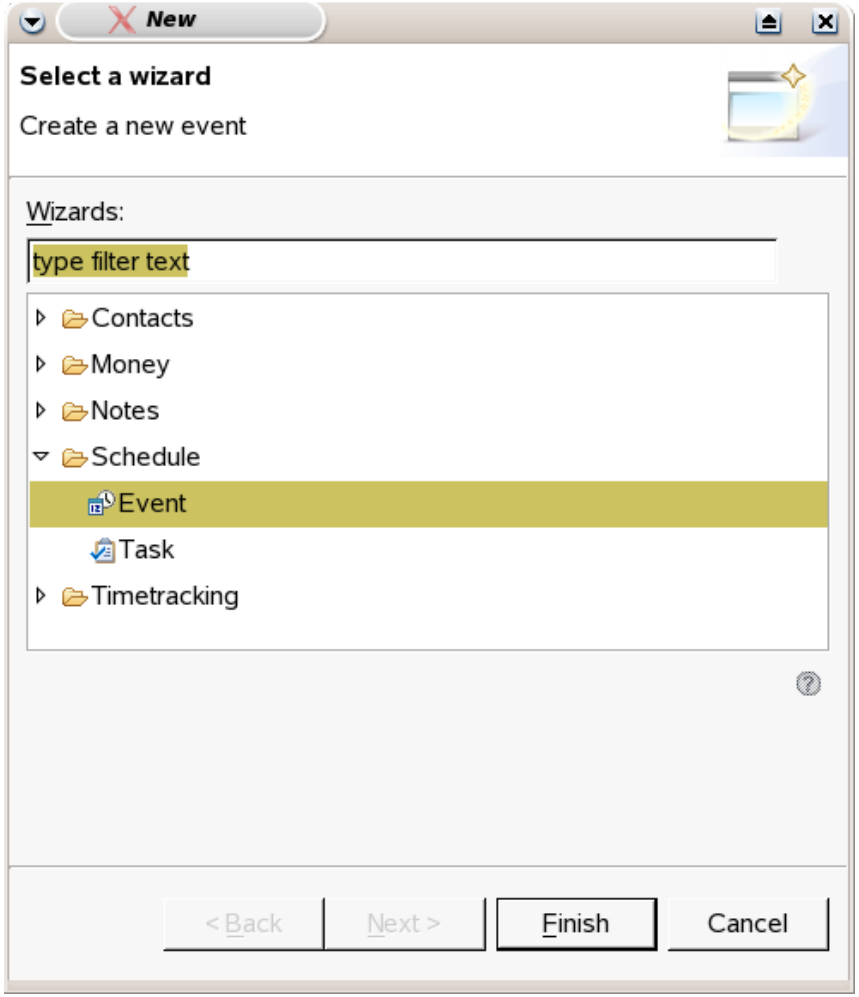




And mobile and embedded..???

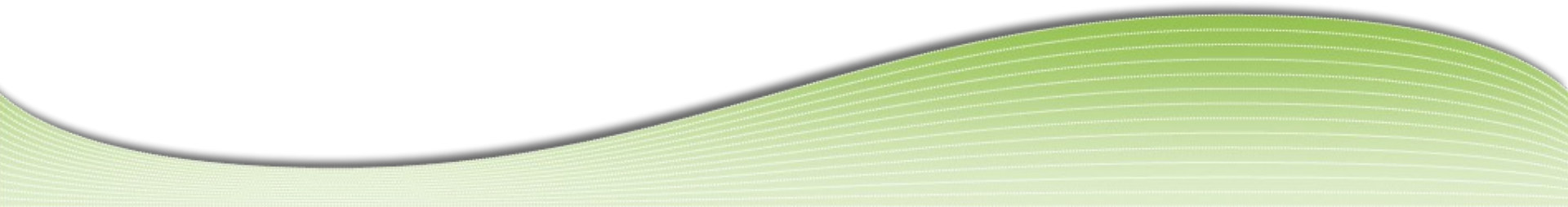
Example: Widget Recognition Options

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

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

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

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



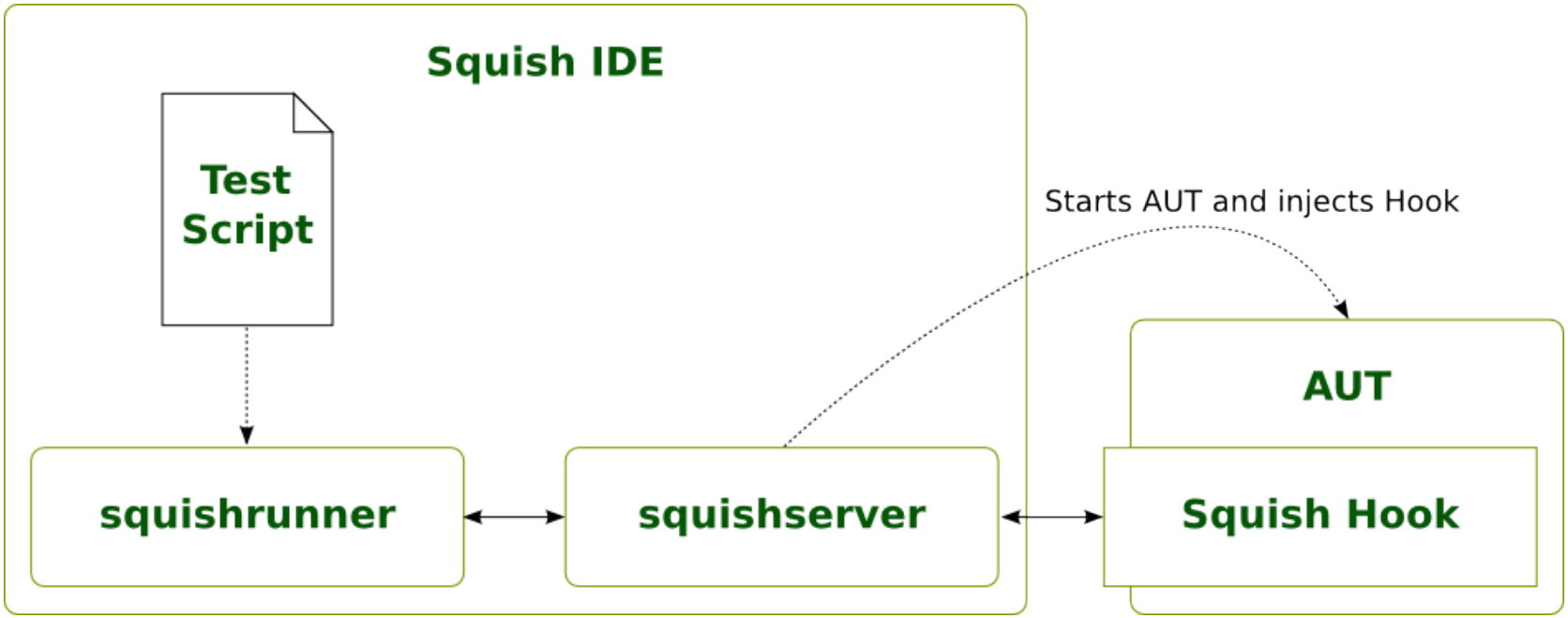
Help from Developers

- QObject::setObjectName()
- QML “id” property



Architecture

Location vs. Remote



Demo

froglogic

Live