
# Write Once, Test Everywhere
# The Challenge of Cross Platform GUI Test Automation

**Gregor Schmid**
**Quality First Software GmbH**
gregor.schmid@qfs.de
Tel: +49 8171 919870


# Overview

- Quality First Software GmbH
- Cross Platform Development
- Java GUI Technologies: Web, AWT, Swing, SWT
- GUI Testing in General
- GUI Test Automation, its ROI and Cross-Platform Aspects
- Available Automation Tools
- Specifics of Swing Test Automation
- Specifics of SWT Automation
- Results
- Questions...


# Quality First Software GmbH

- Established 2001
- Primary product: ***qftestJUI*** – The Java GUI Testtool
- Employees: 5
- Based near Munich
- Committed to quality
- Focus on Java and test automation
- Over 200 customers worldwide in all kind of business categories

# References


hp
invent


bea


AGFA
Agfa


ESO


PHILIPS


tarent


UFA


CURSOR
SOFTWARE AG


océ


TROPOS
networks


softface


Zend
The php Company

# Wanted: Swiss Distributor


# Cross Platform GUI Development

- Windows is still the predominant target platform.
- Various non-Java GUI toolkits available, tcl/tk, gtk, qt, wxWindows...
- Java drastically simplifies cross platform development.
- Java IDEs are themselves available on multiple platforms.
- „Write once, run everywhere“ implies „Write once, test everywhere“.
- Programmer's paradise becomes tester's hell...


# Java GUI Technologies: Web

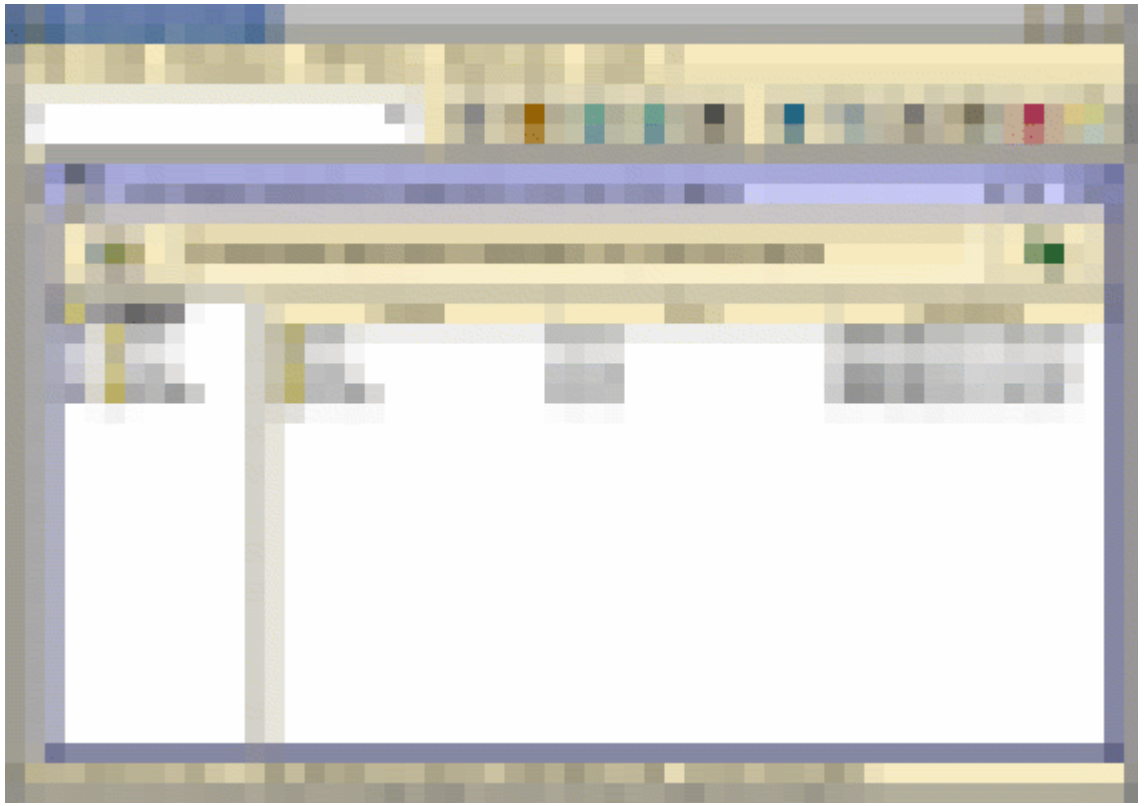
- Server side Java, client side HTML and Javascript.
- Very portable
- No deployment effort.
- Limited functionality (thin client).
- Browser compatibility issues.


# Java GUI Technologies: AWT (Abstract Widget Toolkit)

- Very limited set of components.
- Terrible look and feel (Bild?).
- Heavyweight, waste of resources.
- Neither true native nor common cross-platform look and feel.

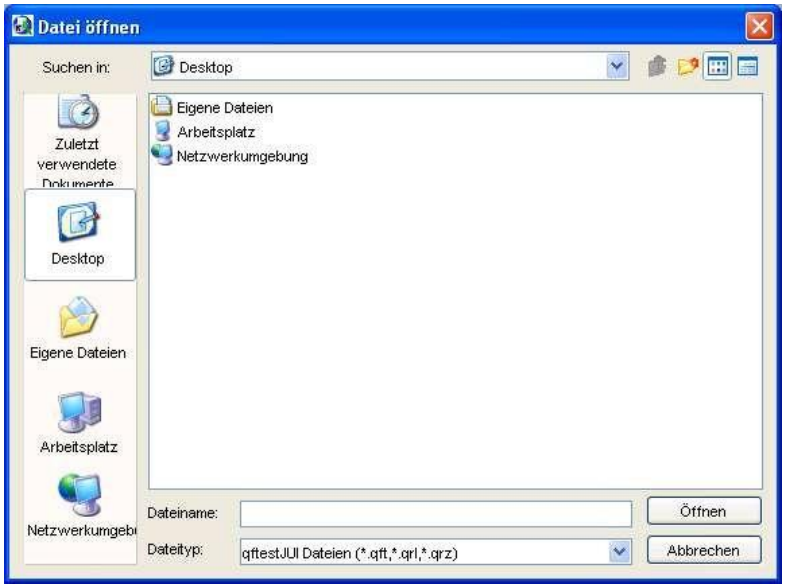
# Java GUI Technologies: Swing

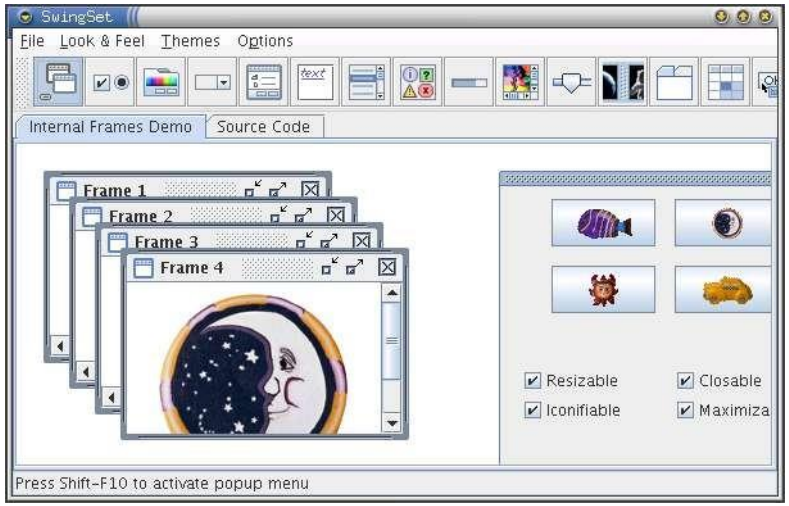
- Built on top of a thin layer of AWT.



# Java GUI Technologies: Swing

- Built on top of a thin layer of AWT.
- Various look and feel variants. Current versions are very close to native look.


# Java GUI Technologies: Swing

- Built on top of a thin layer of AWT.
- Various look and feel variants. Current versions are very close to native look.
- Older versions were slow and bloated, performance of current versions is very good except for start-up.
- Rich set of components and features, flexible architecture, highly extendable, mature.


# Java GUI Technologies: SWT (Standard Widget Toolkit)

- Implemented as a thin layer on top of native GUI toolkits.
- Modelled very closely after Win32 API, not consequently object oriented.
- Initially, only few systems were supported to varying degrees. Today widely applicable, Windows, Unix and Mac OS X versions quite mature.
- Feature set not as complete as Swing, harder to extend.
- Enormous momentum due to Eclipse and the Rich Client Platform.


Swing Best cross-platform toolkit in terms of supported platforms, coherent behaviour and extensibility.

SWT Best platform integration and highest performance.


# GUI Testing in General

- Unit tests are very important, but test isolated subsystems, typically at class level.
- Integration tests that test subsystems in combination are difficult to set up.
- Neither are a substitute for system tests.
- GUI Tests don't test ***the GUI***, but the system as a whole ***through the GUI***.
- GUI Tests operate from the point of view of the end user.
- In a cross-platform situation, complete system tests should be run on all target platforms.

# GUI Test Automation

- Manual GUI testing is time consuming and tedious.
  ⇒Automation has a high potential for savings.

## Who should automate tests?

# ROI for GUI Test Automation

| | Manual | Automation | Influencing Factors |
|---|---|---|---|
| **Preparations** | Test planning<br>Provisions for testing environment | | |
| **Determining Test-cases** | Analysis of business cases | | |
| **Test development** | Preparation of instruc-tions for testers ☺☺ | Development of test-cases with test tool | Complexity or ease of use of the tool<br>Possibilites for reuse |
| **Documentation** | Test-plans correlate with test instructions | Generated from test-cases | |
| **Test Management** | Maintenance of docu-ments | Management of test-suites, scripts and data | Format of test-suites, scripts and data |
| **Test Execution** | Slow, high costs for mul-tiple testers and associ-ated hardware | Automatic, fast, optimal use of available hardware ☺☺ | Quality and reliability of test execution engine |
| **Management of Results** | Manual entries for test-results | Automatic report generation ☺ | Quality of reports |
| **Maintenance of Tests** | Changes to test-instructions only if use-cases change fundamentally ☺☺ | Adaptation to changes in the GUI | Quality of component recognition, adaptability to GUI changes, support for modularization |

# Influence of Cross-Platform Aspects on the ROI for GUI Test Automation

| | Manual | Automation without cross-platform support | Automation with cross-platform support |
|---|---|---|---|
| **Preparations** | | | |
| **Determining Test-cases** | | | |
| **Test development** | Adaptation of instructions for platform-dependent test-cases | Development of test-cases for each platform | Adaptation of test-cases that are platform-dependent, provision of platform-dependent data |
| **Documentation** | | Different formats for doc-umentation | |
| **Test Management** | Maintenance of documents | Separate management of test-suites and data for each tool | Management of platform-dependent aspects in a single code-base |
| **Test Execution** | Multiplied by number of platforms | Multiplied by number of platforms | Multiplied by number of platforms |
| **Management of Results** | | Different report formats of separate tools | |
| **Maintenance of Tests** | | Adaptation to changes in the GUI for each tool | Adaptation to changes in the GUI required only once |


# Benefits of Cross-Platform Test Automation

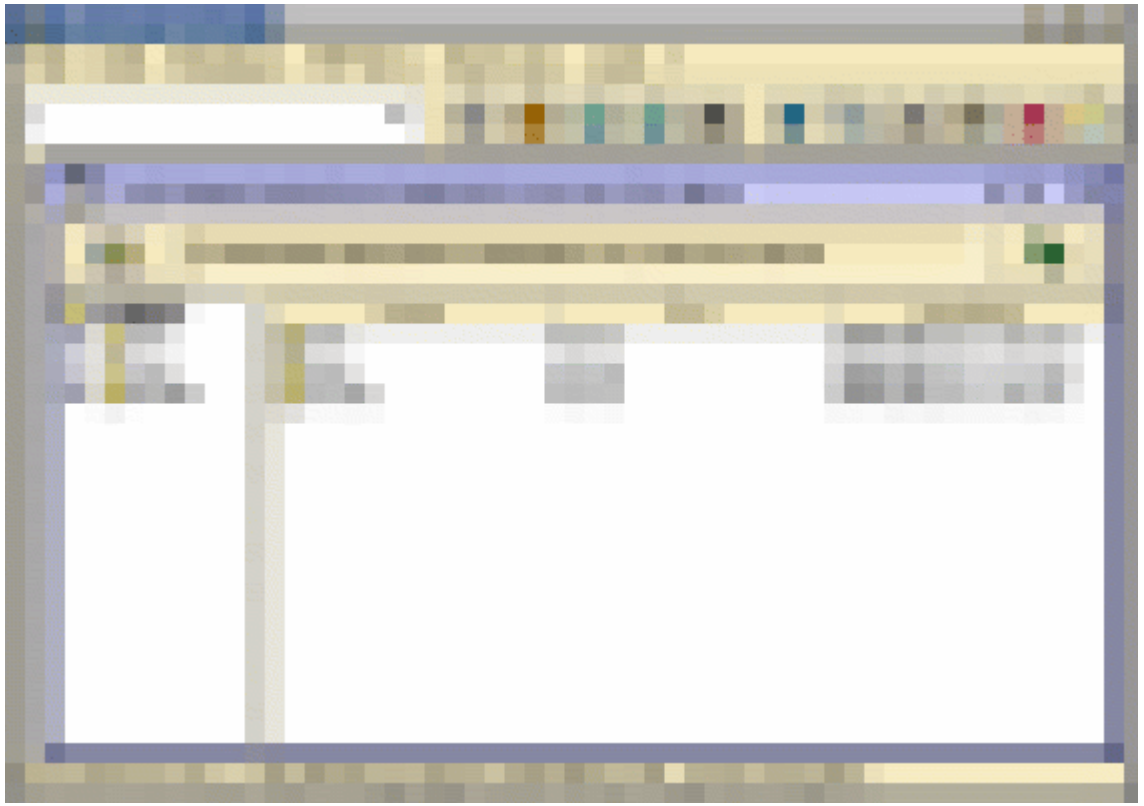
- Reduced costs for tools, only one tool is required.
- Reduced overhead, only one tool has to be learned.
- Drastically reduced effort for test development.
- Tests are easier to maintain in a single test codebase.
- After changes to the application that break tests, only one set of tests needs to be updated.
- No tendency to favour one platform.
- Increased potential for saving compared to manual testing.

# Available Automation Tools

- Web:
  - Until recently all tools were based on Internet Explorer. Now ***AdventNet QEngine*** and ***Selenium*** also support Mozilla and Firefox on Unix.
- AWT/Swing:
  - ***qftestJUI*** is the only true cross platform tool.
  - Windows based test tools like ***WinRunner*** (now QuickTest Professional), ***Rational Robot*** (now XDE) , ***Silktest*** have Java plugins that can drive the SUT on non-windows systems.
- SWT:
  - We're working on it. Windows based tools can be used, but to limited extent.

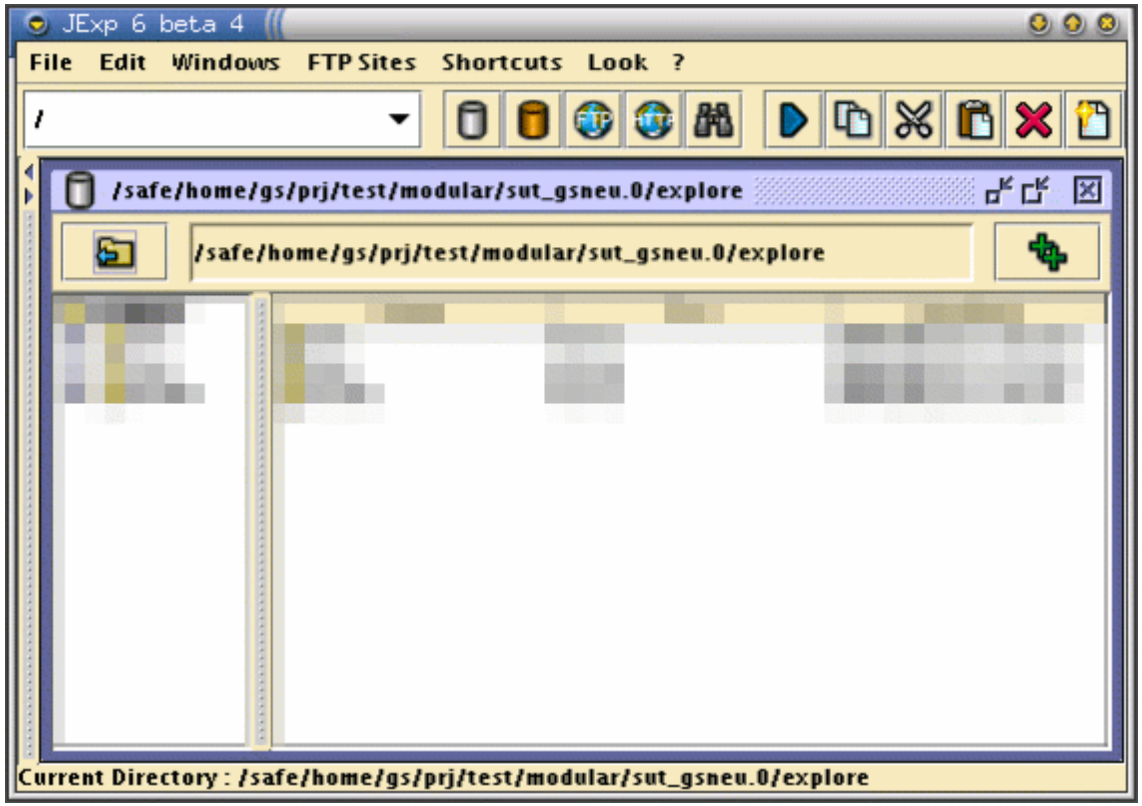
# Specifics of Swing Test Automation

- Component structure is invisible from the Operating System.



# Specifics of Swing Test Automation

- Sub-items of complex components are just „rendered data“.



# Specifics of Swing Test Automation

- Subtle differences in Look&Feel implementations:
  - Different classes -> Abstraction to Look&Feel independent base class
  - Different component layout, e.g. JComboBox
    Windows: 
    
    Linux: 
    
  - Different timing dependent on Look&Feel, e.g. MultiClickThreshold in GTK.


# Specifics of Swing Test Automation

- Benefits for testing:
  - Java reflection makes internals of the application accessible which improves component recognition.
  - Very high level of control thanks to Java Event Queue.
  - Testing independent of „hard“ events at Operating System level which are easily interfered with.


# Specifics of SWT Test Automation

- On each platform, only the absolute minimum of interfaces required for the programmer are implemented. No support for testing or accessibility.
- Widget and Event Loop implementations are different on each platform.
- No common layer of abstraction between the native toolkit and the public API.
- Test engines for each system have to be implemented very close to the native toolkit. This is possible only by extending SWT itself.

# Results

- GUI Test Automation has high potential for savings, provided that tool support is adequate.
- Cross-platform test requirements increase both potential gains and requirements on automation tools.
- Many test tools are available for web client testing, but practically all support Internet Explorer only.
- For Java and Swing, excellent test tools are available, though qftestJUI is the only true cross-platform tool.
- For Java and SWT the situation is difficult. Several vendors are working on SWT automation, though QFS is probably alone in its cross-platform approach.


# Thank you for your attention!

## Questions?