Write Once, Test Everywhere
Wann lohnt sich die Automatisierung von Java GUI Tests?

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

Überblick

- Quality First Software GmbH
- GUI Entwicklung mit Java
- Java GUI Technologien: Web, AWT, Swing, SWT
- GUI Tests im Allgemeinen
- ROI von GUI Testautomatisierung
- Verfügbare Automatisierungstools
- Besonderheiten von Swing und SWT
- Ergebnisse
- Fragen jederzeit!

Quality First Software GmbH

- Gegründet 2001
- Hauptprodukt: ***qftestJUI*** – Das Java GUI Testtool
- Mitarbeiter: 5
- Sitz nahe München
- Qualität steht im Vordergrund
- Fokus auf Java und Testautomatisierung
- Mehr als 200 Kunden weltweit in allen Wirtschaftszweigen

Referenzen


hp
invent


bea


AGFA
Agfa


ESO


PHILIPS


tarent


UFA


CURSOR
SOFTWARE AG


océ


TROPOS
networks


softface


Zend
The php Company

GUI Entwicklung mit Java

- Windows nach wie vor die wichtigste Zielplattform
- Plattformunabhängigkeit dennoch oft ein Thema
- Java vereinfacht die plattformübergreifende Entwicklung drastisch
- Java IDEs sind auf diversen Plattformen verfügbar
- „Write once, run everywhere“ bedeutet auch „Write once, test everywhere“
- Das Paradies für Entwickler wird zur Hölle für Tester...

Java GUI Technologien: Web

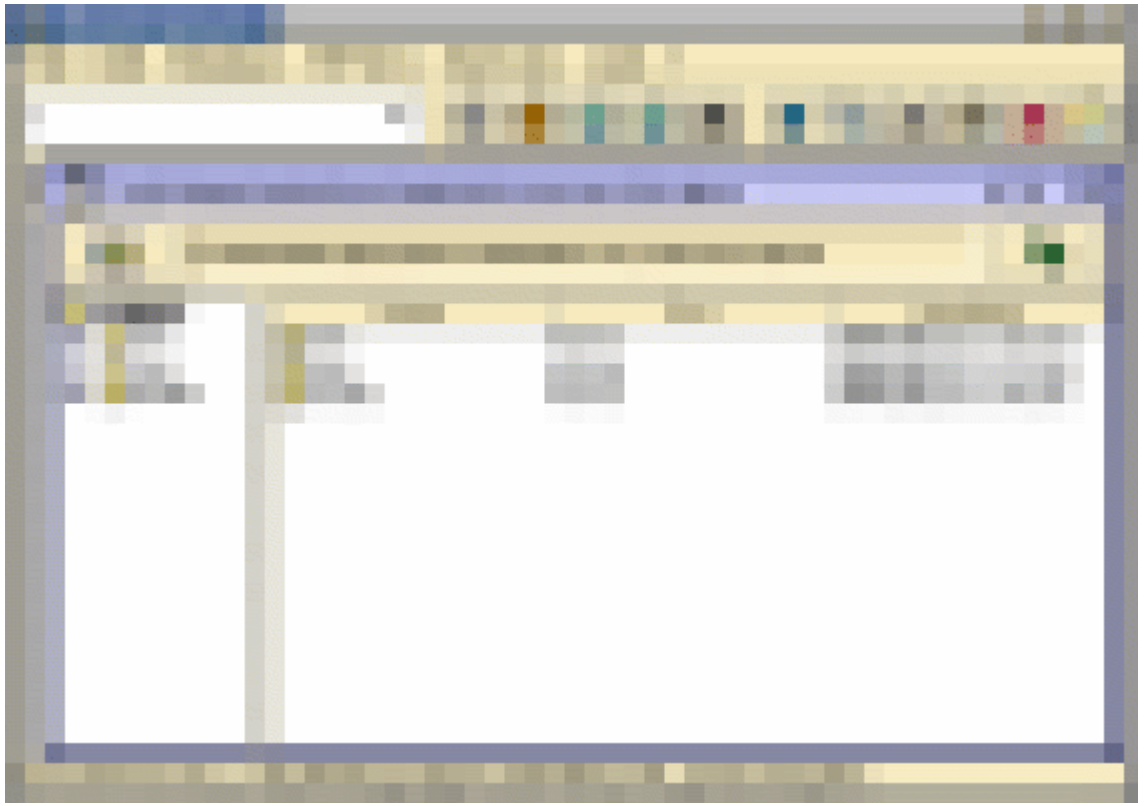
- Java auf dem Server, Clients mit HTML und Javascript
- Sehr portabel
- Kein Aufwand auf der Client Seite bei Installation und Deployment
- Eingeschränkte Funktionalität (thin Client) aber interessante Möglichkeiten mit „rich thin Clients“
- Abhängigkeit von Browsern und Probleme mit der Kompatibilität

Java GUI Technologien: AWT (Abstract Widget Toolkit)

- Ursprüngliche, veraltete GUI Technologie von Java
- Stark eingeschränkte Auswahl von Komponenten
- Schwergewichtig, hoher Verbrauch an Ressourcen
- Weder echter “native” noch gemeinsamer, plattformübergreifender Look&Feel

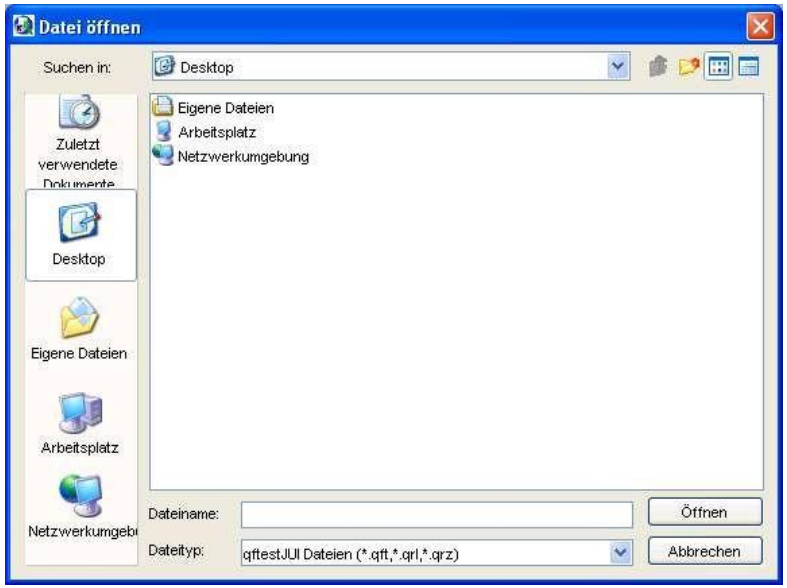
Java GUI Technologien: Swing

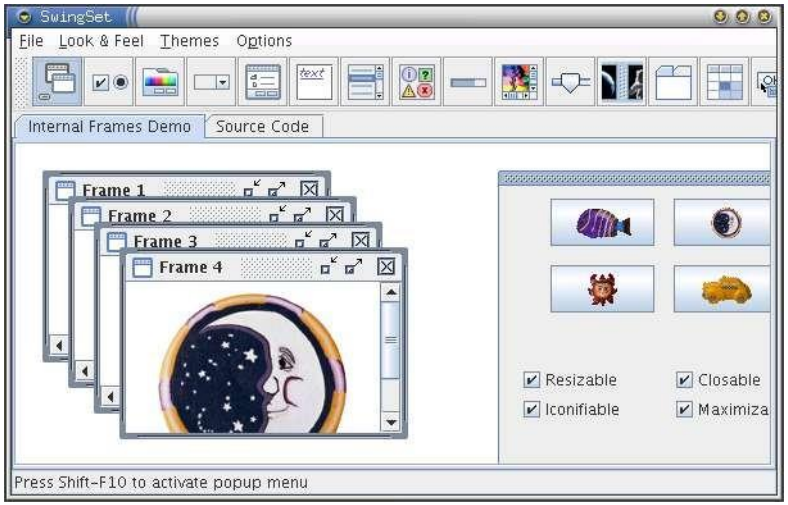
- Setzt auf einer dünnen AWT Schicht auf



Java GUI Technologien: Swing

- Setzt auf einer dünnen AWT Schicht auf
- Verschiedene Look&Feel Varianten verfügbar, aktuell sehr nahe an „native“ Look&Feel





Java GUI Technologien: Swing

- Setzt auf einer dünnen AWT Schicht auf
- Verschiedene Look&Feel Varianten verfügbar, aktuell sehr nahe an „native“ Look&Feel
- Ältere Versionen waren träge und hatten sehr hohen Ressourcenverbrauch
- Aktuelle Versionen sind performant, mit Ausnahme der Initialisierung
- Reiche Auswahl an Komponenten mit vielen Features
- Flexible, durchgängige Architektur, sehr gut erweiterbar, ausgereift

Java GUI Technologien: SWT (Standard Widget Toolkit)

- Als dünne Schicht auf dem jeweiligen “native” GUI Toolkit implementiert
- Sehr stark am Win32 API ausgerichtet, nicht konsequent objektorientiert
- Zu Beginn wurden nur wenige Betriebssysteme und diese unterschiedlich gut unterstützt
- Inzwischen auf Windows, Unix und Mac OS X ausgereift
- Funktionsumfang nicht so vollständig wie Swing, schwieriger zu erweitern
- Gewaltiger Schub durch Eclipse und die Rich Client Plattform

Java GUI Technologien: Swing vs. SWT

Swing

In Bezug auf Vollständigkeit, Zahl der unterstützten Plattformen, und Erweiterbarkeit das beste plattformübergreifende Toolkit.

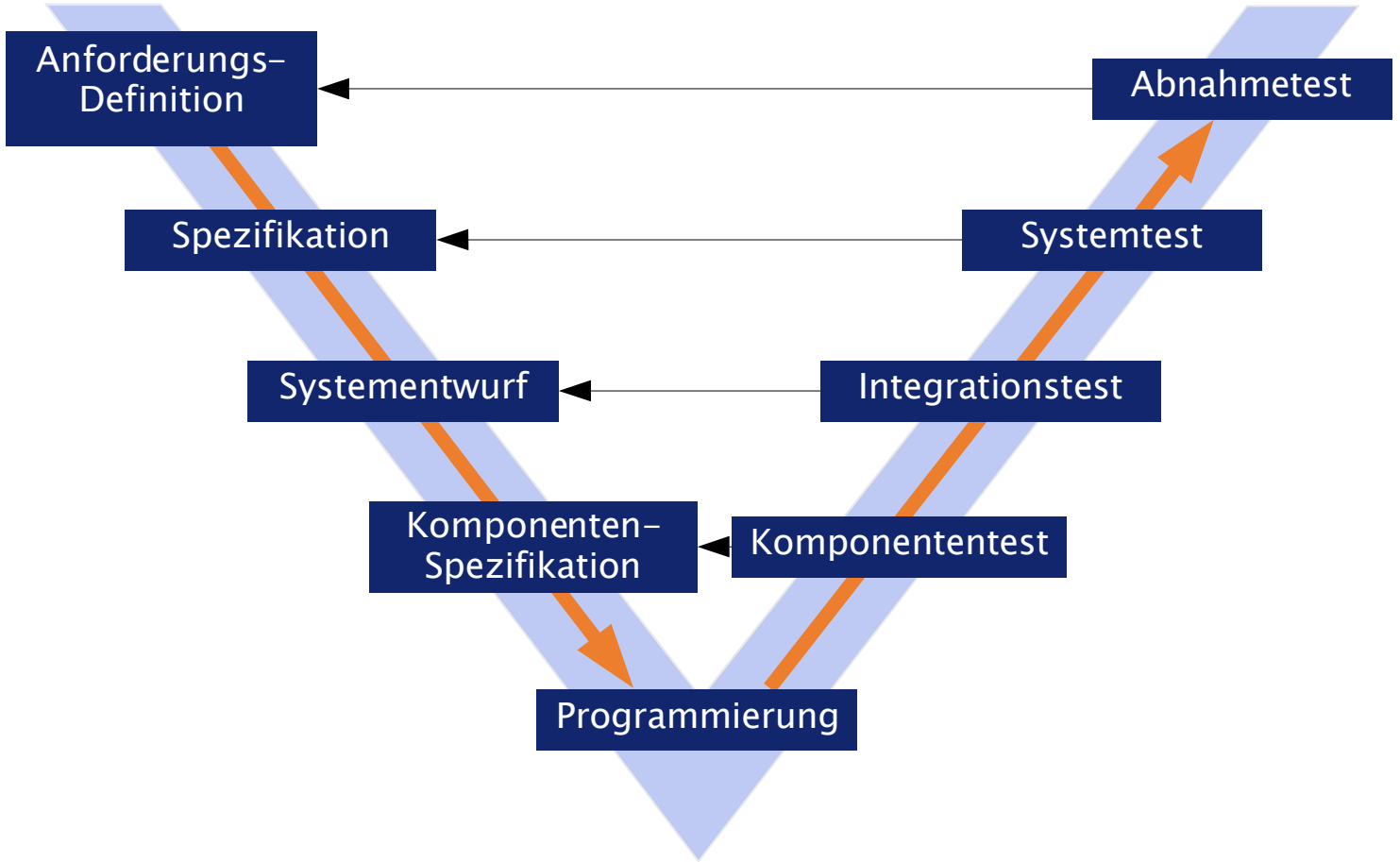
SWT

Beste Plattform Integration und Performance.

Warum Testen?

Warum Testen?

Testen: Das V-Modell



GUI Tests im Allgemeinen

- Unit Tests testen isolierte Subsysteme, typischer Weise auf Klassenebene.
- Integrationstests testen das Zusammenspiel von Subsystemen. Sie sind sehr schwer aufzusetzen.
- Beide sind kein Ersatz für Systemtests.
- GUI Tests testen nicht ***das GUI***, sondern das System als Ganzes ***über das GUI***.
- GUI Tests werden aus Sicht des Endanwenders an einem „lebenden“ System ausgeführt.
- Für plattformübergreifende Anwendungen sollten komplette System Tests auf allen Zielplattformen durchgeführt werden.

Weitere Anwendungen für GUI Tests

- Last- oder Performance Tests können in kleinerem Rahmen auch über das GUI durchgeführt werden.
- Mit System-Überwachungs-Tests (Business Process Monitoring) über das GUI lässt sich kontinuierlich sicherstellen, dass ein System im laufenden Betrieb aus Anwendersicht funktioniert.

GUI Testautomatisierung

- Manuelle GUI Tests sind zeitraubend und monoton
  ⇒Automatisierung hat ein hohes Einsparpotential

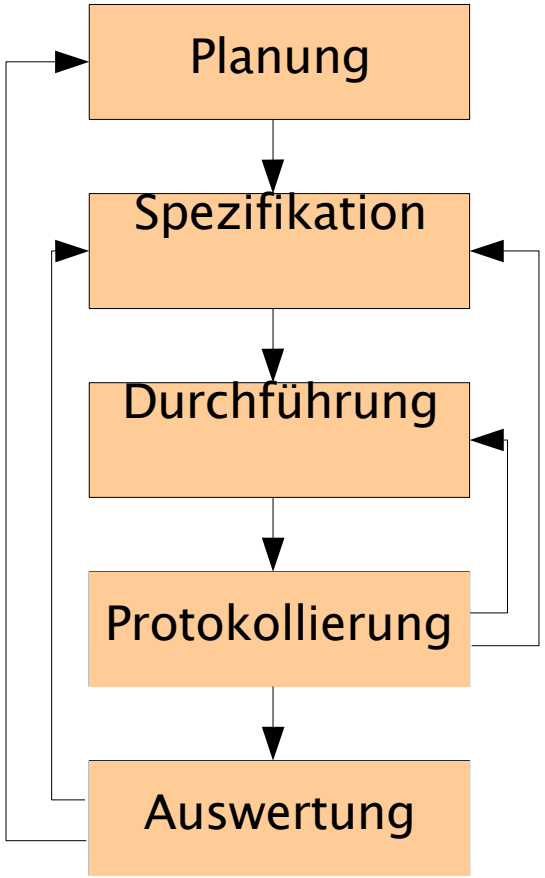
Kurze Demo:
Capture/Replay und darüber hinaus

GUI Testautomatisierung

- Manuelle GUI Tests sind zeitraubend und monoton
  ⇒Automatisierung hat ein hohes Einsparpotential

Wann lohnt sich die Automatisierung von GUI Tests?

Phasen im Testprozess



- Testplanung
- Spezifikation der Testfälle
- Testfall Entwicklung
- Testfall Dokumentation
- Testfall Verwaltung
- Testdurchführung
- Verwaltung der Ergebnisse
- Pflege der Testfälle

Nach British Standard 7925-2

Phasen mit geringem Einfluss auf ROI

Manuell	Automatisch	Einflussfaktoren

Testplanung	Planen der Tests Bereitstellen der Testumgebung

Phasen mit geringem Einfluss auf ROI

	Manuell	Automatisch	Einflussfaktoren
Testplanung	Planen der Tests Bereitstellen der Testumgebung		
Spezifikation der Testfälle	Analyse und Beschreibung der fachlichen Testfälle		

Phasen mit geringem Einfluss auf ROI

	Manuell	Automatisch	Einflussfaktoren
Testplanung	Planen der Tests Bereitstellen der Testumgebung		
Spezifikation der Testfälle	Analyse und Beschreibung der fachlichen Testfälle		
Testfall Dokumentation	Testplan korreliert mit Testanweisungen	Aus Testfällen generierbar	

Phasen mit geringem Einfluss auf ROI

	Manuell	Automatisch	Einflussfaktoren
Testplanung	Planen der Tests Bereitstellen der Testumgebung		
Spezifikation der Testfälle	Analyse und Beschreibung der fachlichen Testfälle		
Testfall Dokumentation	Testplan korreliert mit Testanweisungen	Aus Testfällen generierbar	
Testfall Verwaltung	Verwaltung der Dokumente	Verwaltung von Testsuiten, Skripten und Daten	Format von Testsuiten, Skripten und Daten

Phasen mit geringem Einfluss auf ROI

	Manuell	Automatisch	Einflussfaktoren
Testplanung	Planen der Tests Bereitstellen der Testumgebung		
Spezifikation der Testfälle	Analyse und Beschreibung der fachlichen Testfälle		
Testfall Dokumentation	Testplan korreliert mit Testanweisungen	Aus Testfällen generierbar	
Testfall Verwaltung	Verwaltung der Dokumente	Verwaltung von Testsuiten, Skripten und Daten	Format von Testsuiten, Skripten und Daten
Verwaltung der Ergebnisse	Manuelles Eintragen der Ergebnisse	Automatische Report Generierung	Qualität der Reports

Phasen mit starkem Einfluss auf ROI

	Manuell	Automatisch	Einflussfaktoren
Testfall Entwicklung	Erstellen der Anweisungen für die Tester	Implementierung der Testfälle mit dem Testtool	Komplexität, Bedienbarkeit des Tools, Möglichkeiten zur Wiederverwendung

Phasen mit starkem Einfluss auf ROI

	Manuell	Automatisch	Einflussfaktoren
Testfall Entwicklung	Erstellen der Anweisungen für die Tester	Implementierung der Testfälle mit dem Testtool	Komplexität, Bedienbarkeit des Tools, Möglichkeiten zur Wiederverwendung
Test-durchführung	Langsam, hohe Kosten für Personal und Hardware	Automatisch, schnell, optimale Ausnutzung der Hardware	Zuverlässigkeit des Testtools bei der Testdurchführung

Phasen mit starkem Einfluss auf ROI

	Manuell	Automatisch	Einflussfaktoren
Testfall Entwicklung	Erstellen der Anweisungen für die Tester	Implementierung der Testfälle mit dem Testtool	Komplexität, Bedienbarkeit des Tools, Möglichkeiten zur Wiederverwendung
Test-durchführung	Langsam, hohe Kosten für Personal und Hardware	Automatisch, schnell, optimale Ausnutzung der Hardware	Zuverlässigkeit des Testtools bei der Testdurchführung
Pflege der Testfälle	Anpassung der Anweisungen nur nach fundamentalen Änderungen	Anpassung der Testfälle an die Veränderungen im GUI	Qualität der Wiedererkennung, Anpassungsfähigkeit an verändertes GUI, Modularisierung

Einfluss von Cross-Plattform Aspekten auf ROI

	Manuell	Automatisch ohne Cross-Plattform	Automatisch mit Cross-Plattform
Testfall Entwicklung	Anpassen der Anweisungen an plattformabhängige Abläufe	Implementierung der Testfälle mit jedem Testtool separat	Anpassen nur der plattformabhängigen Testfälle

Einfluss von Cross-Plattform Aspekten auf ROI

	Manuell	Automatisch ohne Cross-Plattform	Automatisch mit Cross-Plattform
Testfall Entwicklung	Anpassen der Anweisungen an plattformabhängige Abläufe	Implementierung der Testfälle mit jedem Testtool separat	Anpassen nur der plattformabhängigen Testfälle
Testfall Dokumentation		Unterschiedliche Formate für Dokumentation	

Einfluss von Cross-Plattform Aspekten auf ROI

	Manuell	Automatisch ohne Cross-Plattform	Automatisch mit Cross-Plattform
Testfall Entwicklung	Anpassen der Anweisungen an plattformabhängige Abläufe	Implementierung der Testfälle mit jedem Testtool separat	Anpassen nur der plattformabhängigen Testfälle
Testfall Dokumentation		Unterschiedliche Formate für Dokumentation	
Testfall Verwaltung		Separate Verwaltung der Tests für jedes Tool	Gemeinsame Verwaltung aller Tests

Einfluss von Cross-Plattform Aspekten auf ROI

	Manuell	Automatisch ohne Cross-Plattform	Automatisch mit Cross-Plattform
Test-durchführung	Multipliziert mit Anzahl der Plattformen	Multipliziert mit Anzahl der Plattformen	

Einfluss von Cross-Plattform Aspekten auf ROI

	Manuell	Automatisch ohne Cross-Plattform	Automatisch mit Cross-Plattform
Test-durchführung	Multipliziert mit Anzahl der Plattformen	Multipliziert mit Anzahl der Plattformen	
Verwaltung der Ergebnisse		Unterschiedliche Formate für Reports	

Einfluss von Cross-Plattform Aspekten auf ROI

	Manuell	Automatisch ohne Cross-Plattform	Automatisch mit Cross-Plattform
Test-durchführung	Multipliziert mit Anzahl der Plattformen	Multipliziert mit Anzahl der Plattformen	
Verwaltung der Ergebnisse		Unterschiedliche Formate für Reports	
Pflege der Testfälle		Anpassung der Testfälle an die Veränderungen im GUI für jedes Tool	Anpassung der Testfälle an die Veränderungen im GUI nur einmal

Vorteile von Cross-Plattform Testautomatisierung

- Reduzierte Kosten für Tools, nur ein Tool wird benötigt
- Geringerer Aufwand für Einarbeitung
- Drastisch reduzierter Aufwand für Testentwicklung
- Tests sind einfacher zu verwalten
- Nach größeren Änderungen am GUI der Anwendung muss nur ein Satz von Tests nachgepflegt werden
- Keine Tendenz, auf einer Plattform bevorzugt zu testen
- Höheres Einsparpotential gegenüber manuellem Testen

Weitere Einsatzmöglichkeiten: Lasttests über das Client GUI

- Alternative zu Protokoll basierten Lasttests
- Höherer Ressourcenverbrauch, daher nur mit begrenzter Zahl von Clients möglich
- Einfache Erstellung komplexer Tests
- Hoher Grad an Wiederverwendbarkeit von funktionalen Tests
- Es werden End-To-End Zeiten ermittelt, im Gegensatz zu Antwortzeiten des Servers

Weitere Einsatzmöglichkeiten: Systemüberwachung über das GUI

- Alternative zum Prüfen der Server auf „Lebenszeichen“.
- Prüft aus Sicht des Endanwenders Verfügbarkeit, Antwortzeitverhalten und Korrektheit in einem.
- Hoher Grad an Wiederverwendbarkeit von funktionalen Tests.

Entscheidend für den ROI: Wiederverwendung

- Der Grad an Wiederverwendung auf allen Ebenen ist das zentrale Kriterium:
  - Wiederverwendung innerhalb der Tests
  - Häufigkeit der Regressionstests
  - Stabilität der Tests bei Systemveränderung
  - Einsatz auf mehreren Plattformen
  - Wiederverwendung der funktionalen Tests, z.B. für Lasttests oder zur Systemüberwachung

⇒ Wie gut unterstützt das jeweilige Tool diese Arten von Wiederverwendung?

Weitere Vorteile von Automatisierung

- Tests sind schneller und können häufiger komplett durchgeführt werden
  ⇒ kürzere Entwicklungszyklen und frühere Fehlererkennung
- Höhere Zuverlässigkeit (menschlicher Faktor)
- Reproduzierbare Ergebnisse

Σ schnellerer Markteintritt bei höherer Softwarequalität

Verfügbare Automatisierungstools

Web

- Diverse kommerzielle Capture/Replay Tools in allen Preiskategorien, ebenso diverse Open Source Tools, allerdings Skript basiert ohne Aufnahmemöglichkeit

AWT/Swing

- Open Source: Abbot, JFCUnit, Marathon, entwicklerlastig, ohne oder mit stark eingeschränkten Aufnahmemöglichkeiten
- Windows basierte Testtools wie ***WinRunner*** (jetzt QuickTest Professional), ***Rational Robot*** (jetzt XDE) , ***Silktest*** bieten inzwischen Java Plugins für Swing
- ***qftestJUI*** ist auf Swing spezialisiert

SWT

- Abbot (nur harte Events mit kritischem Timing), Windows basierte Tools mit Einschränkungen
- ***qftestJUI*** demnächst

Verfügbare Cross-Plattform Automatisierungstools

Web

- Bis vor Kurzem nur Internet Explorer. Inzwischen unterstützen zumindest ***AdventNet QEngine*** und ***Selenium*** auch Mozilla und Firefox unter Unix
- Programmierte Tests mit Open Source Tools sind plattformunabhängig

AWT/Swing

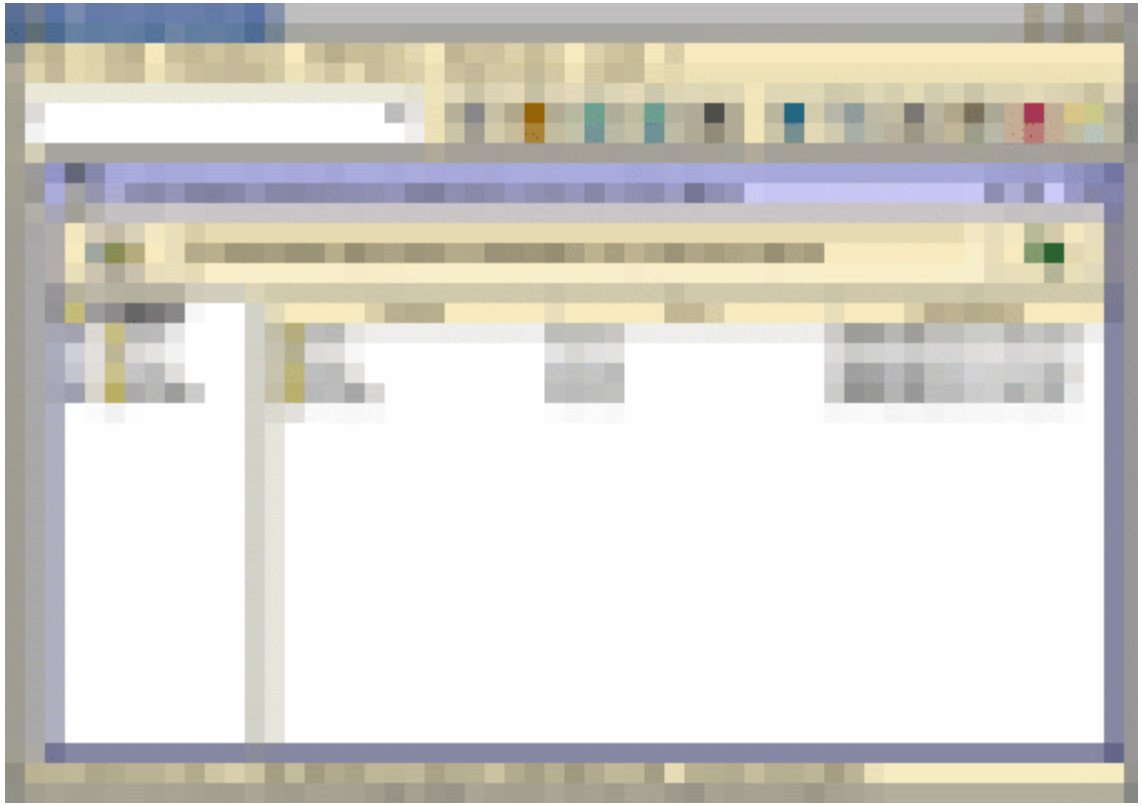
- ***qftestJUI*** ist das einzige echte Cross-Plattform Capture/Replay Tool
- Java Plugins der Windows basierten Testtools können das SUT auch auf nicht-Windows Systemen treiben

SWT

- ***qftestJUI*** wird zunächst Windows und Linux/Gtk unterstützen
- Abbot sollte für SWT ebenfalls plattformunabhängig sein

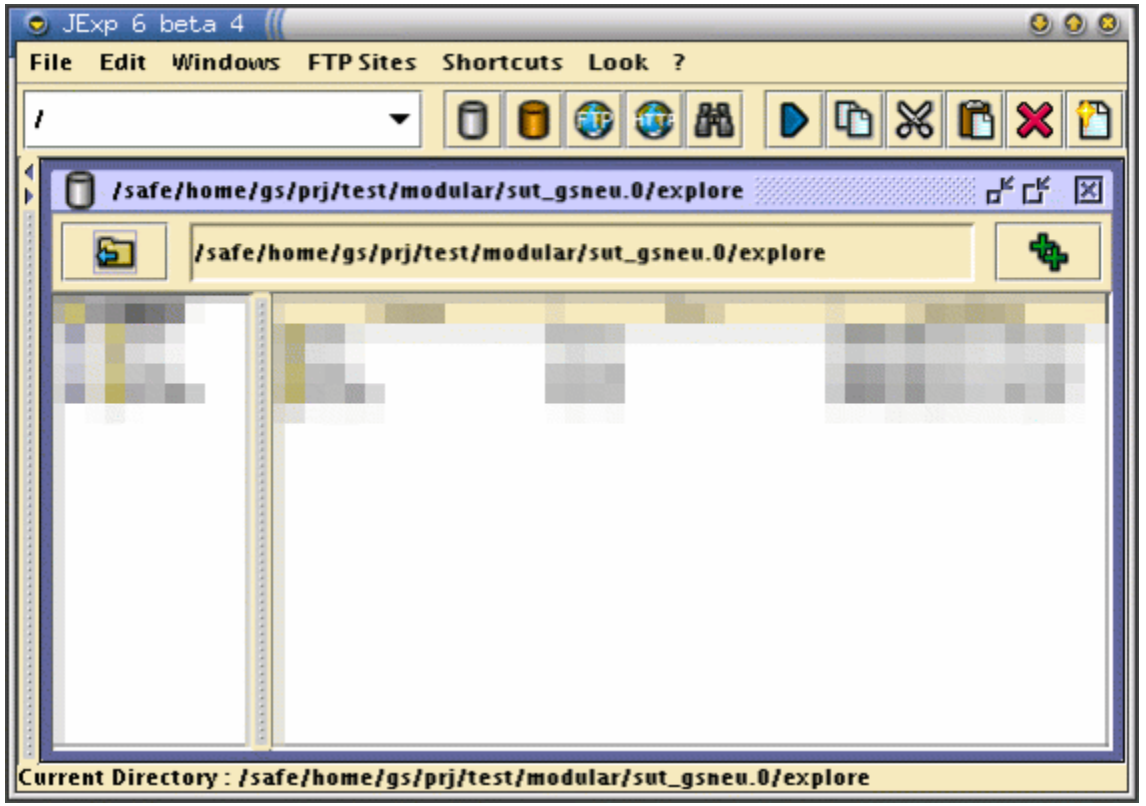
Besonderheiten der Testautomatisierung für Swing

- Die Komponentenstruktur ist für das Betriebssystem unsichtbar



Besonderheiten der Testautomatisierung für Swing

- Unterelemente komplexer Komponenten sind nur „gemalte Daten“



Besonderheiten der Testautomatisierung für Swing

- Subtile Unterschiede in Look & Feel Implementierungen:
  - Unterschiedliche Klassen -> Abstraktion auf vom Look&Feel unabhängige Basisklassen
  - Unterschiede im Layout der Komponenten, z.B. Windows:  Linux: 
  - Unterschiede im Timing abhängig vom Look&Feel, z.B. MultiClickThreshold in GTK

Besonderheiten der Testautomatisierung für Swing

- Vorteile von Swing für Testautomatisierung:
  - Java Reflection erlaubt Zugriff auf Interna der Anwendung was die Wiedererkennung von Komponenten deutlich verbessert
  - Sehr präzise Kontrolle über die Anwendung dank eigener Java Event Queue
  - Tests sind unabhängig von “harten” Events auf Ebene des Betriebssystems
  - Filterung von harten Events für gleichzeitiges Testen mehrerer Clients

Besonderheiten der Testautomatisierung für SWT

- Auf jeder Plattform wurde nur das absolute Minimum implementiert, um das SWT API für Entwickler bereitzustellen
- Implementationen von Widgets und Event Loop sind auf jeder Plattform anders
- Keine gemeinsame Abstraktionsschicht zwischen dem „native“ Toolkit und dem SWT API
- Test Engines müssen für jede Plattform sehr nahe am “native” Toolkit implementiert werden. Dazu muss SWT selbst erweitert werden.

Ergebnisse

- Für Anwendungen mit komplexem GUI sind Systemtests über das GUI unverzichtbar.
- GUI Testautomatisierung hat ein hohes Einsparpotential, wenn die Toolunterstützung angemessen ist.
- Entscheidend für den ROI von GUI Testautomatisierung ist der Grad an Wiederverwendung auf allen Ebenen.
- Die Palette an verfügbaren Testtools ist sehr breit. Bei der Auswahl der Testtools sollte der langfristige Nutzen genau bedacht werden, insbesondere die Unterstützung der Wiederverwendung.

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?