Wann lohnt sich GUI-Testautomatisierung?

... und was Entwickler dafür tun können.

Martin Moser

Quality First Software GmbH
qfs@qfs.de
Tel: +49 8171 919870

Überblick

- **Hintergrund**
- Motivation
- ROI der GUI-Testautomatisierung
- Vorteile sichern

Quality First Software GmbH

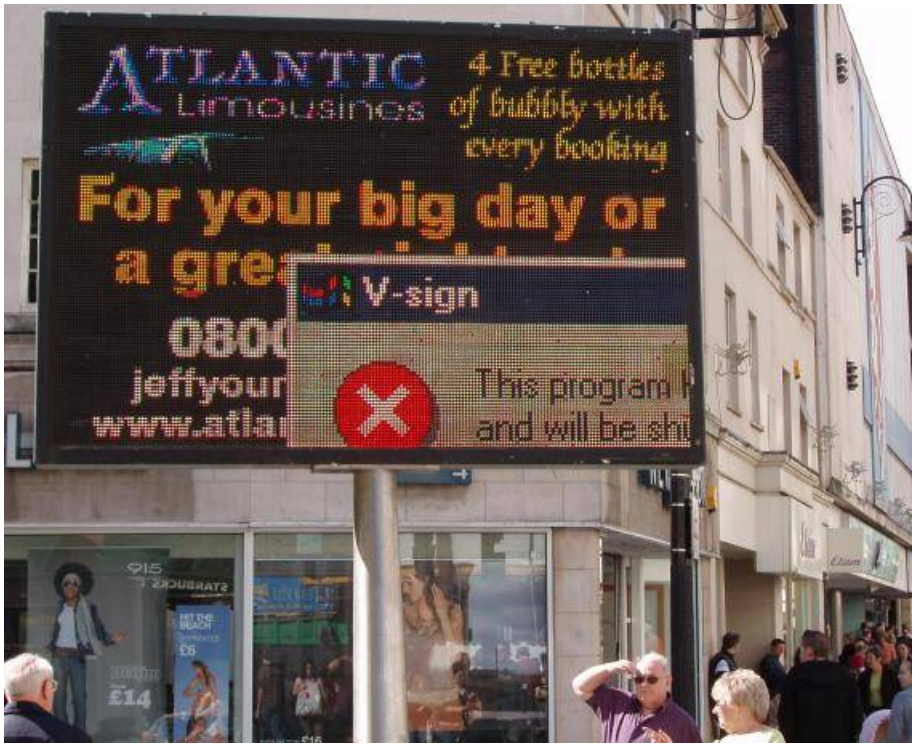
- Gegründet 2001
- Hauptprodukt: **QF-Test** – Das Java GUI Testtool
- Mitarbeiter: 7
- Sitz nahe München
- Qualität steht im Vordergrund
- Fokus auf Java und Testautomatisierung
- Mehr als 400 Kunden weltweit in allen Wirtschaftszweigen

Überblick

- Hintergrund
- **Motivation**
- ROI der GUI-Testautomatisierung
- Vorteile sichern

Warum Testen?

Warum Testen?



Überblick

- Hintergrund
- Motivation
- **ROI der GUI-Testautomatisierung**
- Vorteile sichern

Definition GUI-Tests

- **Unit-Tests**
  - sehr wichtig, aber isolierte Subsysteme
  - auf Klassenebene

- **Integrationstests**
  - testen das Zusammenspiel von Subsystemen
  - schwierig aufzusetzen

- **GUI-Tests**
  - testen nicht das GUI, sondern das System *als Ganzes über das GUI* → ein häufiges Missverständnis
  - werden aus Sicht des Endanwenders an einem „lebenden“ System ausgeführt

Return On Investment (ROI)

© Imbus AG, www.imbus.de

Phasen des Testprozesses

Phasen mit wenig Einfluss auf den ROI

	Manuell	Automatisiert	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 Reportgenerierung	Qualität der Reports

Phasen mit hohem Einfluss auf den ROI

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

Phasen mit hohem Einfluss auf den ROI

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

Phasen mit hohem Einfluss auf den ROI

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

Phasen mit hohem Einfluss auf den ROI

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

Plattformübergreifende Testautomatisierung

	Manuell	Automatisiert	Plattform-übergreifend
Testfall Entwicklung	Erstellen der Anweisungen für die Tester	Implementierung der Testfälle mit dem Testtool	Anpassung der Testfälle, die plattformabhängig sind. Bereitstellung plattformspezifischer Testdaten
Test Ausführung	Langsam, hohe Kosten für Personal und Hardware	Automatisch, schnell, optimale Ausnutzung der Hardware	Abdeckung mehrerer Plattformen
Pflege der Testfälle	Anpassung der Anweisungen nur nach fundamentalen Änderungen	Anpassung der Testfälle an die Veränderungen im GUI	Änderungen im GUI nur einmal nachziehen

Plattformübergreifende Testautomatisierung

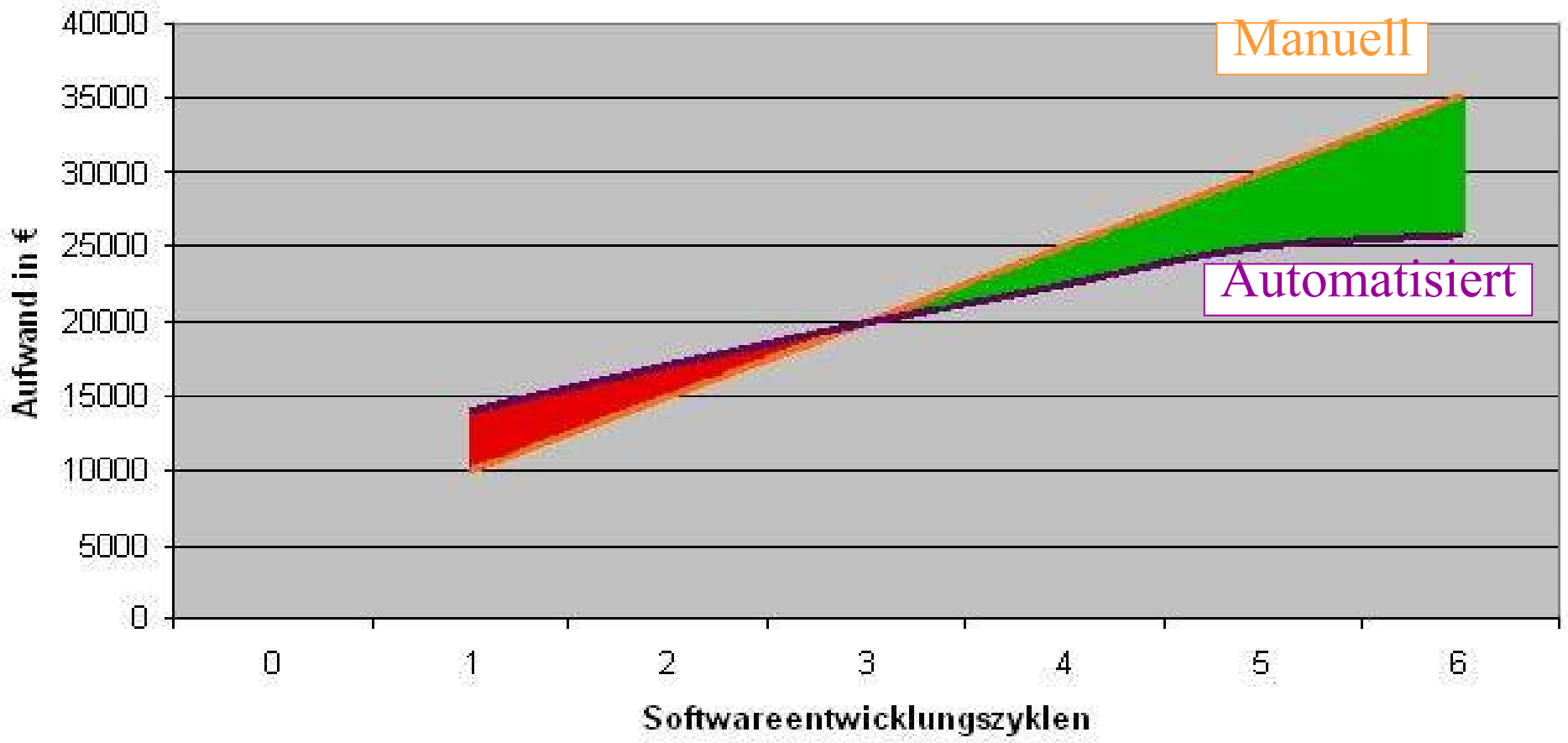
	Manuell	Automatisiert	Plattform-übergreifend
Testfall Entwicklung	Erstellen der Anweisungen für die Tester	Implementierung der Testfälle mit dem Testtool	Anpassung der Testfälle, die plattformabhängig sind. Bereitstellung plattformspezifischer Testdaten
Test Ausführung	Langsam, hohe Kosten für Personal und Hardware	Automatisch, schnell, optimale Ausnutzung der Hardware	Abdeckung mehrerer Plattformen
Pflege der Testfälle	Anpassung der Anweisungen nur nach fundamentalen Änderungen	Anpassung der Testfälle an die Veränderungen im GUI	Änderungen im GUI nur einmal nachziehen

Entscheidend für den ROI

Wiederverwendbarkeit

- Wiederverwendbarkeit innerhalb mehrerer Tests durch Modularisierung
- Häufigkeit der Regressionstests
- Stabilität der Tests bei Systemveränderung
- Einsatz auf mehreren Plattformen
- Tests verschiedener Produktversionen, -linien
- Wiederverwendung der funktionalen Tests, z.B. für Lasttests oder zur Systemüberwachung

Return On Investment (ROI)



© Imbus AG, www.imbus.de

Weitere Vorteile von Automatisierung

- Ermöglicht Regressiontests → häufigere und schnellere Testausführung → kürzere Entwicklungszyklen
- Höhere Zuverlässigkeit (menschlicher Faktor)
- Reproduzierbare Ergebnisse
- Unbeaufsichtigte Testausführung ohne Benutzerinteraktion
- Motivation für manuelle Tester: Konzentration auf schwierige Testszenarien anstatt langweiliger Routinetests

schnellerer Markteintritt
höhere Produktqualität
höhere Zuverlässigkeit

Überblick

- Hintergrund
- Motivation
- ROI der GUI-Testautomatisierung
- **Vorteile sichern**

Erfolgreicher GUI-Testprozess

- Kommunikation zwischen Testern und Entwicklern
- Erstellung wiederverwendbarer Testprozeduren
- Erstellung spezifischer Testbibliotheken für Softwarekomponenten
- Entwicklungszyklus der Testautomatisierung parallel oder ähnlich dem Softwareentwicklungszyklus

Auswahl des Testtools

- Stabiles und verlässliches Capture/Replay?
- Erkennung aller Arten von GUI-Elementen (auch komplexe Elemente wie Bäume oder Tabellen)?
- Modularisierung von Tests möglich?
- Parametrisierung von Testprozeduren möglich?
- Trennung von Testablauf- und Testeingabedaten möglich?
- Plattformübergreifend? Werden alle testrelevanten Plattformen unterstützt?
- Integrationsmechanismus mit existierenden Test-Execution/Management-Tools?
- Lokalisierungs(L10N)-Tests möglich?

... und was können Entwickler dafür tun?

- Swing Oberflächen
- Eclipse/SWT Oberflächen
- Web Oberflächen

- Größter Einfluss bei Komponentenerkennung!
- Sonst kaum Einfluss

Komponentenerkennung – Wahl des Bezeichners

- Eindeutig
- Sprechend
- Langfristig stabil

- können u.a. auch verwendet werden für
  - Accessibilty Interfaces/Screen Reader
  - Hilfefunktionen

- **Achtung bei dynamischer Generierung !!**

... und was können Entwickler dafür tun?

- Swing Komponentenerkennung
  - Component.setName() spielt bei fast allen eine wichtige Rolle
  - Bereitstellung von toolspezifischen Erweiterungen
  - JLabel.setLabelFor(Component) besser als nichts

... und was können Entwickler dafür tun?

- SWT/Eclipse Komponentenerkennung
  - sehr unterschiedliche Ansätze bei Erkennung der Komponenten
  - setData(„name“) bei einigen Tools
  - diverse Settermethoden bei Views, Workbenches bei RCP Anwendungen
  - bei meisten wird Label herangezogen
  - Bereitstellung/Anpassung von toolspezifischen Erweiterungen

... und was können Entwickler dafür tun?

- Web
  - Id besonders wichtig, sollte also im Dokument eindeutig sein (wird von HTML nicht erzwungen)
  - Bei Cascading Stylesheets (CSS) besser über class referenzieren als über Id:

    HTML: `<SPAN class="red">…</SPAN>`

    CSS: `span.red { color: red }`

    an Stelle von

    HTML: `<SPAN id="red">…</SPAN>`

    CSS: `span#red { color: red }`
  - Schwierig bei AJAX und anderen Toolkits die Id dynamisch generieren
  - Name bei FORM und INPUT

Verfügbare Automatisierungstools

Windows

- QuickTest Professional (Mercury/HP, aka WinRunner), XDE Functional Tester (IBM Rational, aka Robot), Silktest (Borland), TestPartner, QARun (Compuware), Squish (Froglogic) etc.

Unix

- XRunner (Mercury/HP), XDE Functional Tester, Silktest, Squish (Froglogic for QT and XView).

Web

- Diverse kommerzielle Capture/Replay Tools in allen Preiskategorien, ebenso diverse Open Source Tools
- QF-Test (Quality First Software GmbH) ab Ende 2007

Java/Swing

- Open Source: Abbot, JFCUnit, Marathon – sehr entwicklerlastig
- Windows-basierte Testtools bieten inzwischen Java Plugins für Swing.
- QF-Test (Quality First Software GmbH)

Java/SWT

- Open-Source: Abbot
- Open-Source: TPTP für Eclipse Plugins.
- Windows-basierte Testtools bieten inzwischen Java Plugins für SWT.
- QF-Test (Quality First Software GmbH)

Wann lohnt sich GUI-Testautomatisierung?

- Etablierung des Prozesses
- Modularisierung und Wiederverwendbarkeit
- Einsatz des geeigneten Tools
- Unterstützung von Entwicklern

Wann lohnt sich GUI-Testautomatisierung?

Diskussion/Fragen

Martin Moser

Quality First Software GmbH
qfs@qfs.de
Tel: +49 8171 919870