

Testest Du schon?

Verfahren und Tools zum Testen von Software

Martin Kompf
Dezember 2010

JAVA USER GROUP DARMSTADT

Testing Software

Ziel des Softwaretests ist es, Fehler aufzudecken. Nachzuweisen, dass keine Fehler vorhanden sind, ist nicht Ziel des Softwaretests.

Quelle: Wikipedia

intro

Inhalt

Akzeptanztest Analyse Architektur **Arquillian** Automatisiert **Black-Box**
Cloud Coverage Feedback Fehler Grenzwertanalyse Hudson

Integrationstest JUnit Lasttest Mock Mockito Persistenz Prüfen

Qualität Reporting Review Risiko **Selenium** ShrinkWrap Software

Sonar Strategie Stresstest Systemtest **Testen** Testfall User-Story

White-Box

Über mich

- Homepage:
 - www.kompf.de
 - Artikel, Buch und Tutorials zu
 - Java
 - Java EE
 - C++
- Beruf:
 - Entwickler, Architekt und Tester bei der PROSTEP AG in Darmstadt
 - www.prostep.com

basics

Klassifizierung von Softwaretests

- Prüftechnik
 - Statisch: Review, ...
 - Dynamisch: Daten-, Funktionsorientiert, ...
- Testkriterium
 - Funktional: Komponente, Schnittstelle, User-Story, ...
 - Nicht-Funktional: Performance, Sicherheit, ...
- **Informationsstand**
 - Black-, White-, and Grey-Box

Black-Box-Test

- Überprüfung der Übereinstimmung eines Softwaresystems mit seiner Spezifikation
- Ableitung der Testfälle aus formaler oder informaler Spezifikation
 - Dies ist oftmals nicht möglich! (Unvollständige Spezifikation, Schleichende Änderungen während Implementierung)
- Hoher organisatorischer Aufwand
 - Eigenes Testteam
 - Zusätzliche Infrastruktur

Black-Box-Test

Gibt keine Antwort auf die Frage: Welche Programmteile sind getestet?

White-Box-Test

- Testfälle werden aus dem Programm selber abgeleitet
 - Unter Umständen ist es schwierig, sämtliche Ausführungspfade zu erkennen, aber formelle Methoden und Tools können hier helfen (Grenzwertanalyse, Klassifikationsbaum, Cobertura)
- Geringerer organisatorischer Aufwand
 - Tests werden durch Entwickler des zu testenden Systems geschrieben
- Gute Toolunterstützung (*Unit*) und Automatisierbarkeit (Luntbuild, Hudson, ...)

White-Box-Test

Gibt keine Antwort auf die Frage: Sind alle Teile der Aufgabenstellung realisiert?

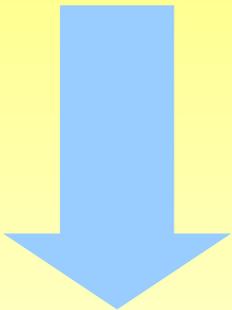
White- oder Black-Box?

- Eine sinnvolle Testreihe sollte Black-Box-Tests und White-Box-Tests kombinieren!
- Aber immer noch Nachteile:
 - Hoher organisatorischer Aufwand
 - Schwierigkeit der Testplanerstellung aus informaler Spezifikation
 - Unvollständige Spezifikation
 - Schleichende Änderungen während Implementierung

Testen in allen Entwicklungsphasen!



Analyst & Architekt

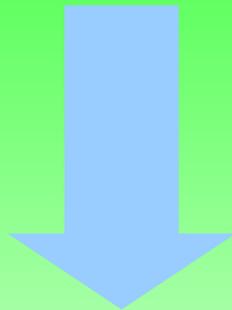


Requirements- und Designtest

- Reviews
- Statische Analyse



Developer

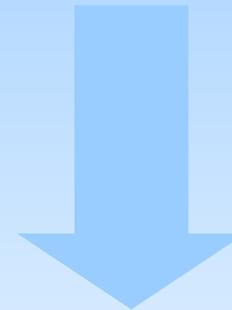


Unit- und Integrationstest

- Dynamische Tests
- White Box
- Kurze Feedbackzyklen



Tester



System- und Akzeptanztests

- Dynamische Tests
- Black Box

Grey-Box-Test

- Eine Methode aus dem Extreme Programming
 - **Testgetriebene „Test-First“ Entwicklung**
- Gemeinsamkeiten mit
 - White-Box-Test
 - Programmierung durch Entwickler des zu testenden Systems
 - Black-Box-Test
 - Unkenntnis über die Interna des zu testenden Systems, weil der Grey-Box-Test vor dem zu testenden System geschrieben wird (Test-First-Programmierung)

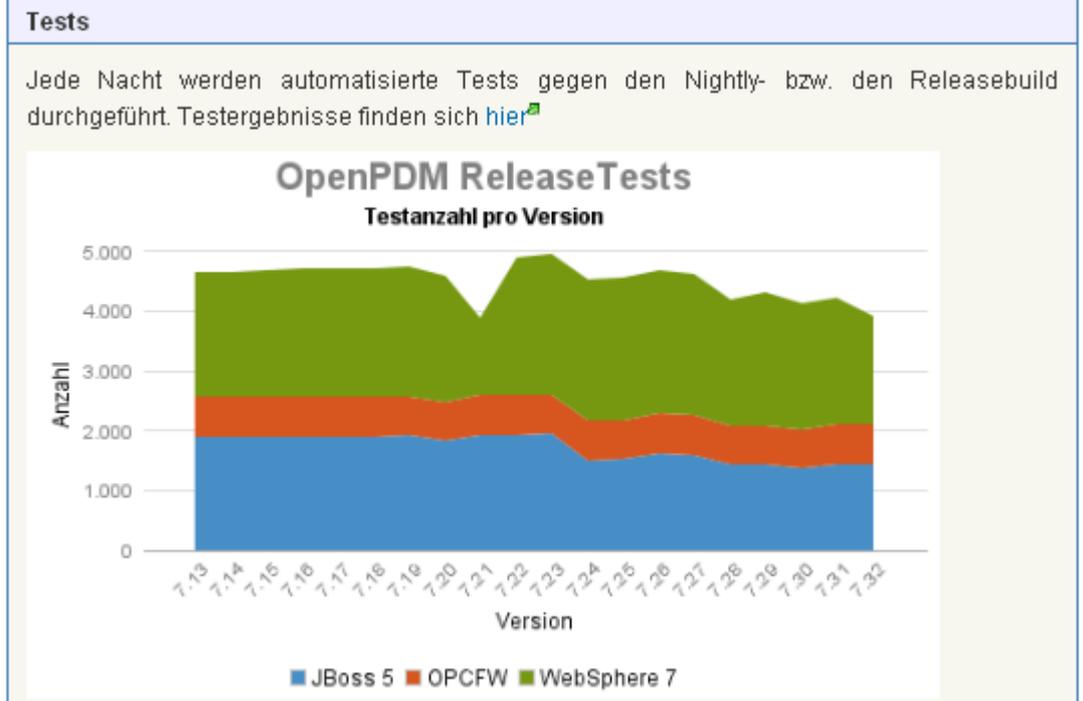
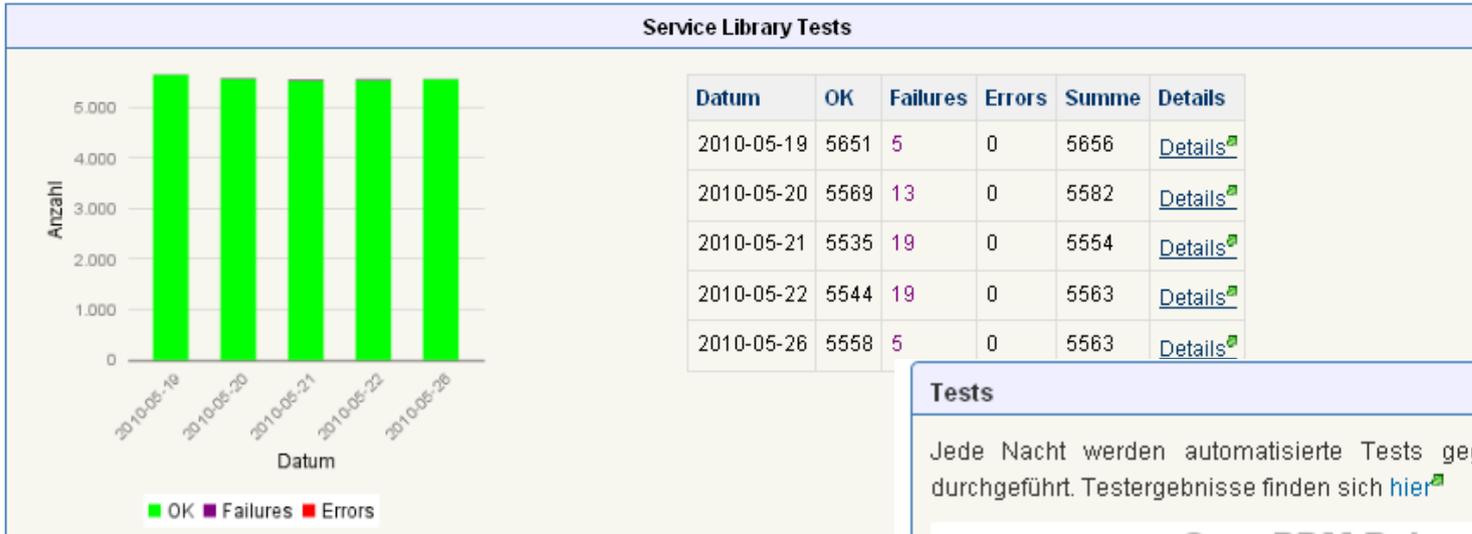
Grey-Box-Test

- sollte nicht unbedacht als vollwertiger Ersatz für Black-Box-Tests gesehen werden.
- Die stützenden Säulen agiler Prozesse müssen unbedingt vorhanden sein!
 - Peer-Review, Paarprogrammierung
 - Akzeptanztests sind nach wie vor erforderlich

Kommunikation!

- Aktive Kommunikation des Teststatus nach außen
 - Testabdeckung
 - Teststatus
 - Lesbar für alle Stakeholder!
 - Nutzung vorhandener Tools (Intranet, Wiki)
 - Automatische Erstellung nach jedem Testlauf

Kommunikation – Beispiel (1)



Kommunikation – Beispiel (3)

Hudson

Hudson » [PLMLC_1_GUI_TESTS](#) search ? log in

[ENABLE AUTO REFRESH](#)

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

Build History [\(trend\)](#)

- #16 [Nov 25, 2010 5:31:38 AM](#)
- #15 [Nov 24, 2010 1:36:02 PM](#)
- #14 [Nov 24, 2010 5:31:24 AM](#)
- #13 [Nov 23, 2010 3:55:46 PM](#)

Project PLMLC_1_GUI_TESTS

GUI Tests (Selenium) of the admin UI

[Workspace](#)

[Last Successful Artifacts](#)

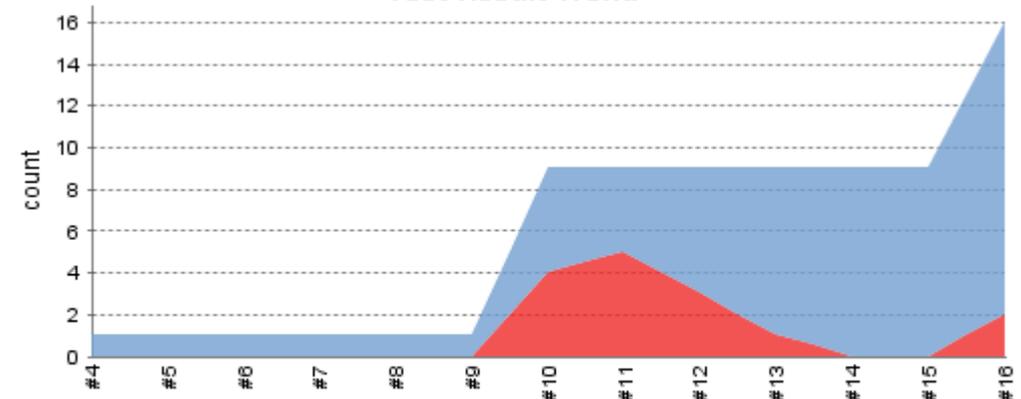
- [version.properties](#) 

[Recent Changes](#)

[Latest Test Result](#) (2 failures / +2)

Test Result Trend

[edit description](#)



Build	Total Count	Failures
#4	1	0
#5	1	0
#6	1	0
#7	1	0
#8	1	0
#9	1	0
#10	9	4
#11	9	5
#12	9	1
#13	9	0
#14	9	0
#15	9	0
#16	9	0
#17	15	1
#18	16	2

[\(just show failures\)](#) [enlarge](#)

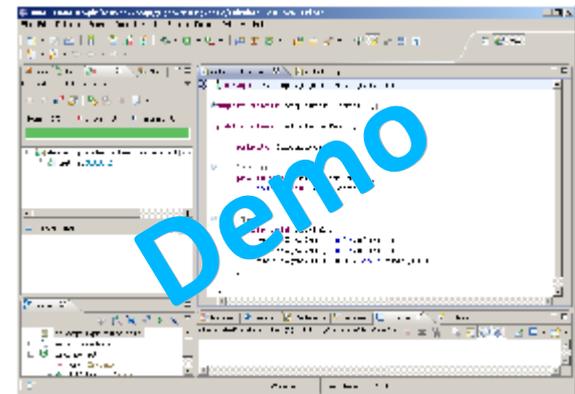
Martin Kompf 2010

17

junit

JUnit

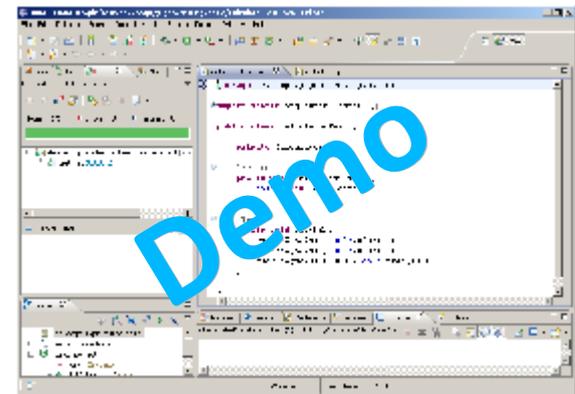
- Testen == Programmieren : Macht Spaß!



integrate

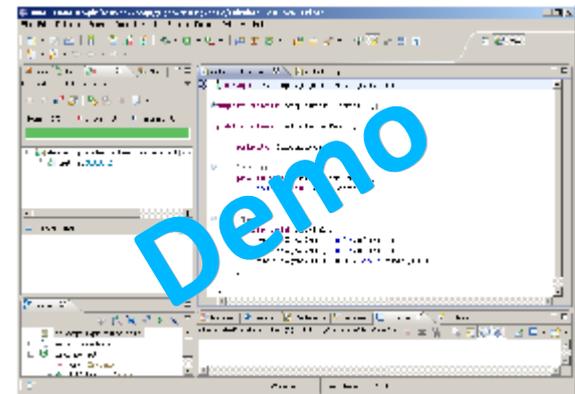
Integrationstests

- Erfordern oft Deployment
 - **ShrinkWarp** „Microdeployment“
- Erfordern oft besondere Ablaufumgebung
 - CDI, Java EE
 - **Arquillian** aktiviert embedded oder remote Container



Integrationstests

- Was ist, wenn Teile der Implementierung noch unvollständig sind?
- Verwendung von **Mock** Objekten!
 - Zum Beispiel **Mockito**



Exkurs: Java EE

- Java Enterprise Edition
 - Industriestandard, aktuell ist Version 6
- Definiert aktive Komponenten
 - z.B. Enterprise Java Beans (EJB)
 - Verhalten wird über Annotationen gesteuert:
 - Transaktionen, Nebenläufigkeit, Sicherheit, Lebenszyklus
 - „Verdrahtung“ erfolgt aktiv zur Laufzeit durch JEE-Container
- Applikationsserver JBoss, WebSphere, Glassfish

Exkurs: CDI

- Java Contexts and Dependency Injection (JSR-299)
- Standard für eine Brücke zwischen Web-Schicht (JSF) und Transaktionaler Ebene (EJB)
 - Beans
 - Dependency Injection
 - Scopes and Contexts
- Referenzimplementierung **Weld**
- Bestandteil von Java EE 6

webgui

Testautomatisierung

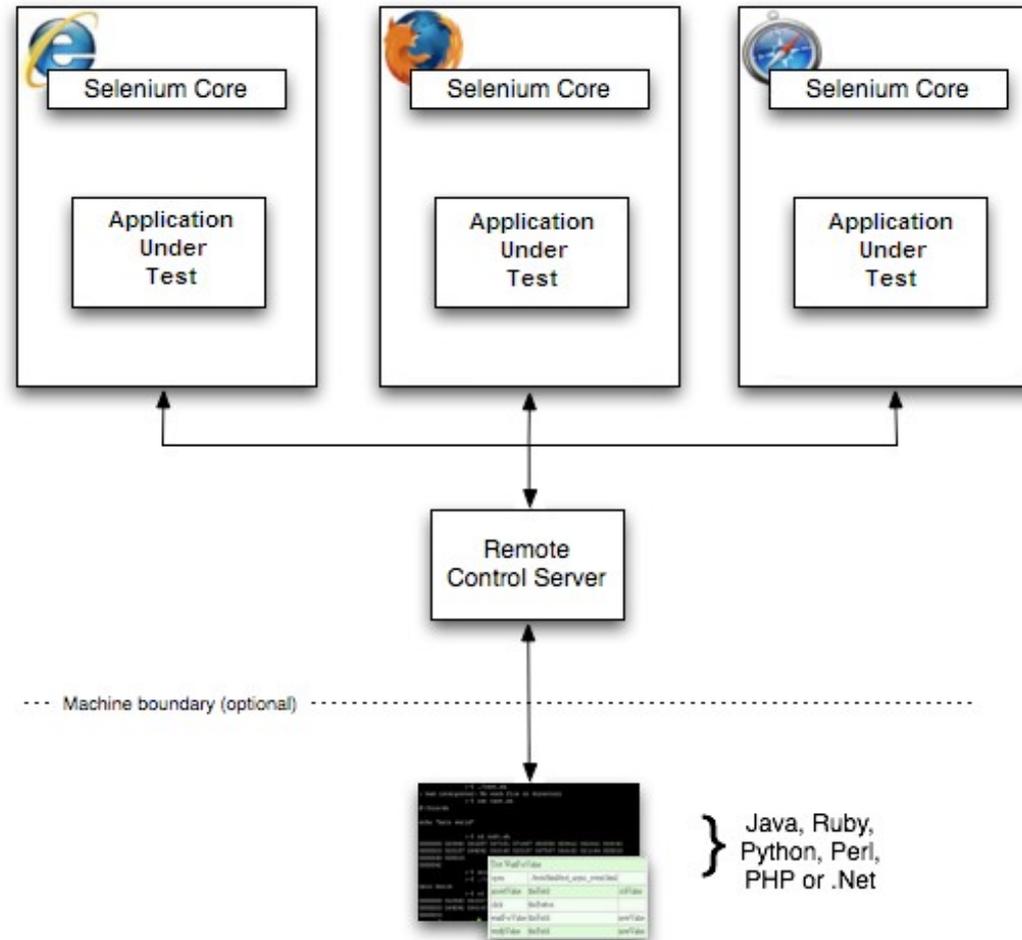
- White-Box-Tests sind in der Regel schon automatisiert
- Black-Box-Tests oftmals noch nicht
 - Grund: Hoher Aufwand!
- Automatisierung der Black-Box-Tests sinnvoll, wenn
 - die Tests regelmäßig wiederholt werden
 - das Verhalten der Anwendung sich nicht bei jedem Release ändert

WebGUI Test – Selenium (1)

- **Selenium**
 - Läuft im Browser
 - Capture und Replay durch Firefox-Plugin
 - Export der Testfälle in eine Programmiersprache: Java, C#, Python, ...
 - Testausführung mittels Selenium RC: Benutzt IE, Firefox, Opera, ...
 - Reporting mittels JUnit
 - Apache 2 Lizenz
- Entwickelt bei ThoughtWorks und Google
 - zum Testen von Gmail und Google Docs

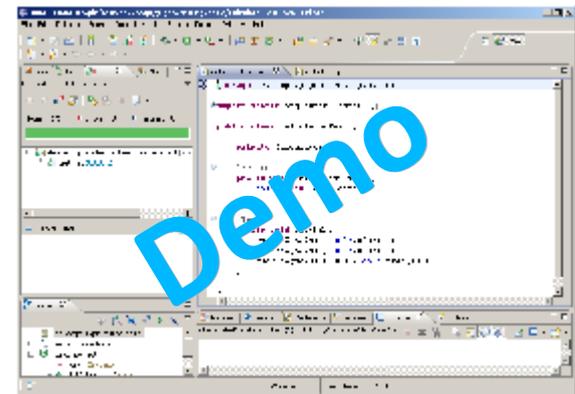
WebGUI Test – Selenium (2)

Windows, Linux, or Mac (as appropriate)...



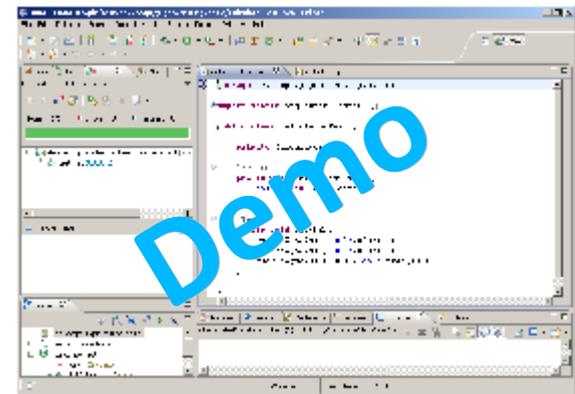
Selenium

- Demo: Click and Verify Links
 - **Assert** versus **Verify**
 - **ClickAndWait** versus **Click**
 - Capture and Replay
 - Export and Test



Selenium + JSF: Real Life

- Links, die keine sind
- Das „Dynamische ID Problem“
 - IDs werden dynamisch während der Erzeugung der Seite auf dem Server generiert, Selenium bevorzugt aber gerade diese IDs!
- **XPath** statt ID
 - FireBug und FireXPath helfen
- Erweiterung **UI Element**
- **Statische IDs** vergeben!



result

Tipps

- Klein anfangen
 - Smoke Tests
 - Reviews
- Kommunizieren
 - Ja, wir testen!
- Test-aware Development
 - Statische IDs in Webanwendungen vergeben
 - Sichtbarkeit von Injection-Points bedenken

Fragen und Antworten

the end