Orientation in Objects

10 Jahre Java Qualitätssicherung

Beim Erscheinen von Java genügten noch sechs simple konstruktive Eigenschaften um die Java Plattform und Sprache als Qualitätsangebot zu positionieren. Heute erstrecken sich die Qualitätsmanagementanstrengungen in der Java, XML und Open Source Welt von konstruktiven Verfahren wie syntaxvalidierenden Entwicklungsumgebungen bis hin zu kontinuierlichen automatisierten Analysen von Codes,Funktionen und Architekturen. Wie kam es zu dieser Entwicklung und hat sich die Qualität der erstellten Software in dieser Zeit wirklich erhöht

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Gliederung

Orientation in Objects

- **Grundlagen des Qualitätsmanagements**
- 1998-2008 Impressionen aus dem OIO-Qualitätsmanagement
- Herausforderung Java 2.0 Qualtitätsmanagement
- Ausblick

Magisches Viereck (Sneed)

Orientation in Objects

Fertigstellungszeitpunkt **Qualität**

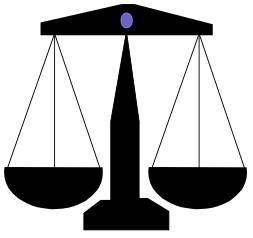
- Entwicklungs-Projekt

Personeller Aufwand **Funktionaler Umfang**

Qualitätskosten ein Minimierungsproblem

Orientation in Objects

- Softwarekosten = Herstellungskosten + Qualitätskosten
- Qualitätskosten = Qualitätssicherungskosten + Fehlerkosten



Qualität

Orientation in Objects

- transzendent
- produktbezogen
- benutzerbezogen
- prozessbezogen
- Kosten/Nutzen -bezogen

Qualität

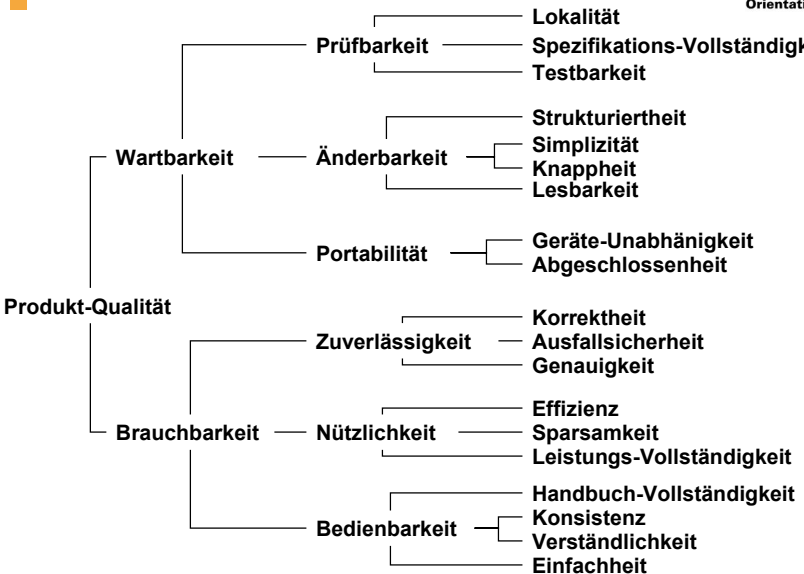
Orientation in Objects

- „Als Qualität eines Gegenstandes bezeichnen wir die Gesamtheit seiner charakteristischen Eigenschaften.“
- „Softwarequalität ist die Gesamtheit der Merkmale und Merkmalswerte eines Softwareproduktes, die sich auf dessen Eignuing beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen“(ISO 9126)

Qualitätsmodelle

- Qualität wird durch Merkmale beschrieben - (engl. Factor)
  - meist benutzerbezogen
- Merkmale können in Teilmerkmale strukturiert werden - (Criteria)
  - eher produktbezogen
- mess- oder bewertbarkeit durch Indikatoren
- Qualitätsmaße sind quantitative Skala und Meß-Methode
- bekannte FCM-Modelle(produktbezogen)
  - ISO 9126
  - FURPS
  - McCall
  - Boehm
  - DGQ

FCM-Modelle und Qualität



Strategien im Qualitätsmanagement

Orientation in Objects

- Bottom-up
  - einfach in die Entwicklung zu integrieren (test first?)
  - Unterstützung durch Unit-Frameworks
  - Anlehnung an andere Ingenieurwissenschaften

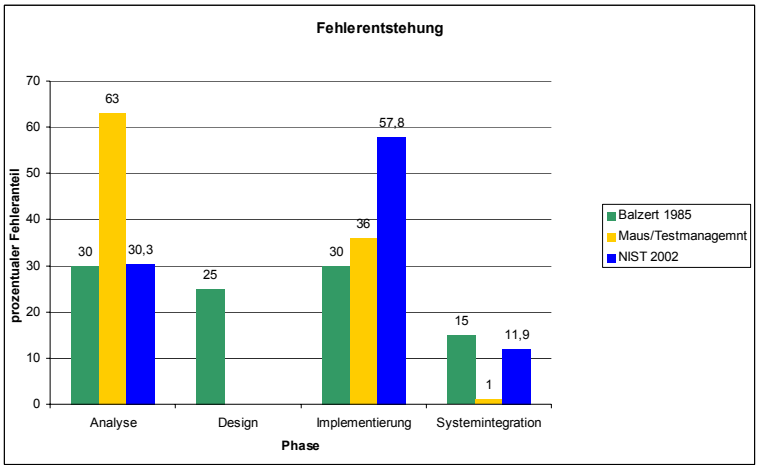
- Top-down Ansätze
  - in Form von C/R-Ansätzen
  - ineffiziente Bsp. bekannt
  - in anderen Ingenieurwissenschaften teilweise sehr spektakulär und teuer

Kaufmännischer Hintergrund

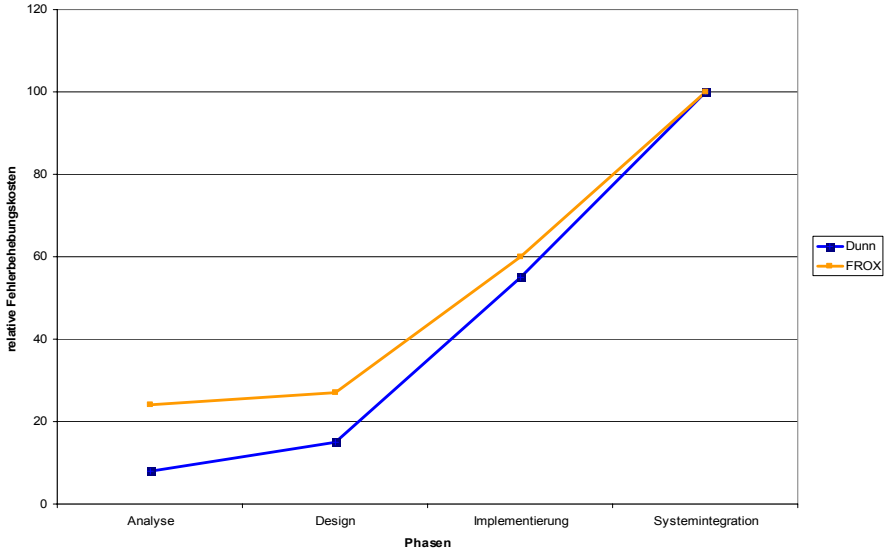
Orientation in Objects

- Aufwand für QM sehr ist hoch
- Ist dieser Aufwand gerechtfertigt, bzw. bezahlbar?
- Risiko durch
  - Personenschaden
  - Hohe Kosten
  - Imageverlust
- „Sei mindestens genauso gut wie die Konkurrenz“

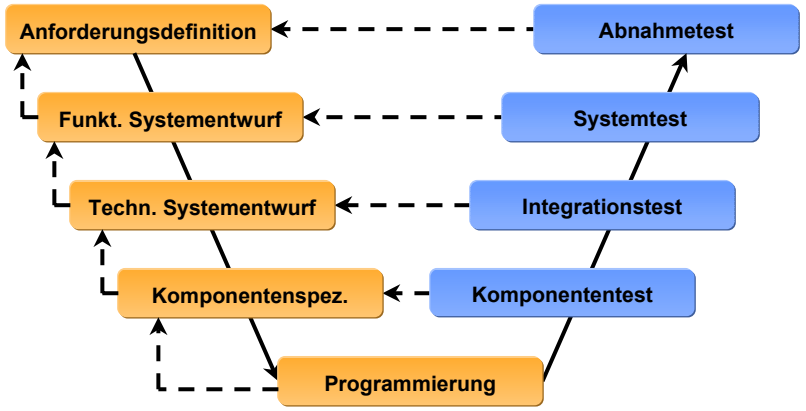
Fehlerentstehung



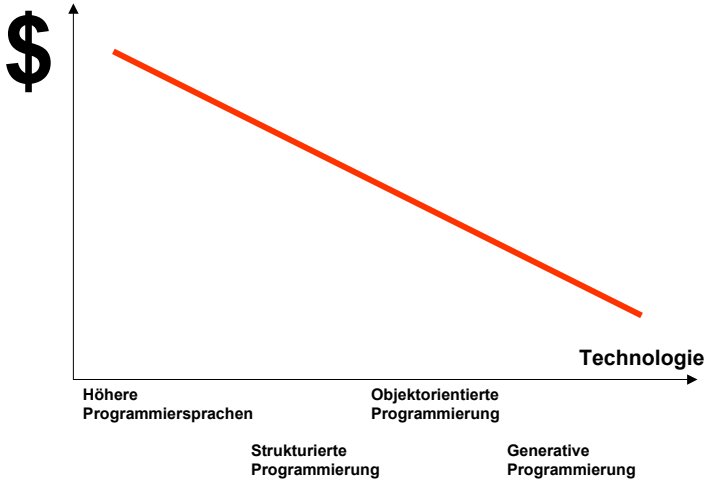
Ökonomische Gründe zum Werkzeugeinsatz II



Allgemeines V-Modell

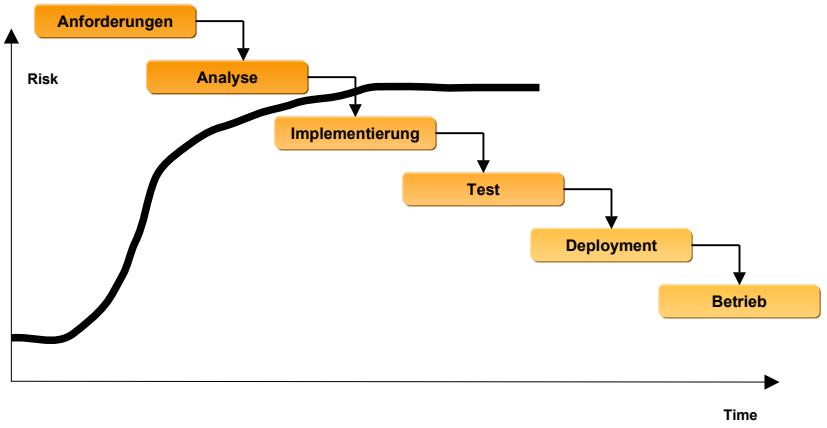


„Die agile These“ - Preis für Änderungen



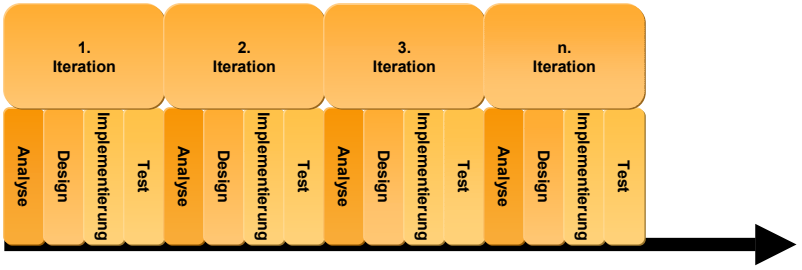
Wasserfallmodell

Orientation in Objects



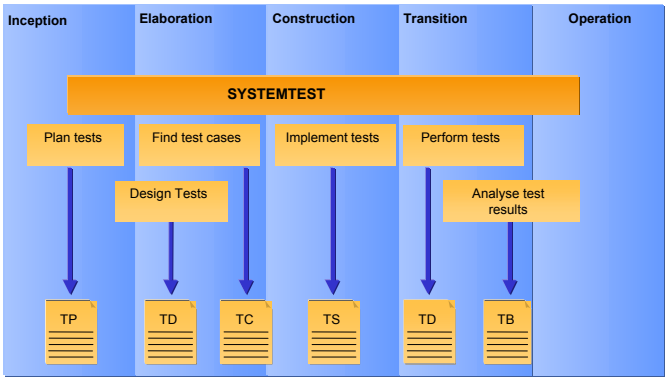
Inkrementell und Iterativ

Orientation in Objects



Classical System Test

Orientation in Objects



TP - Test plan

TD - Test design specification

TC - Test case specification

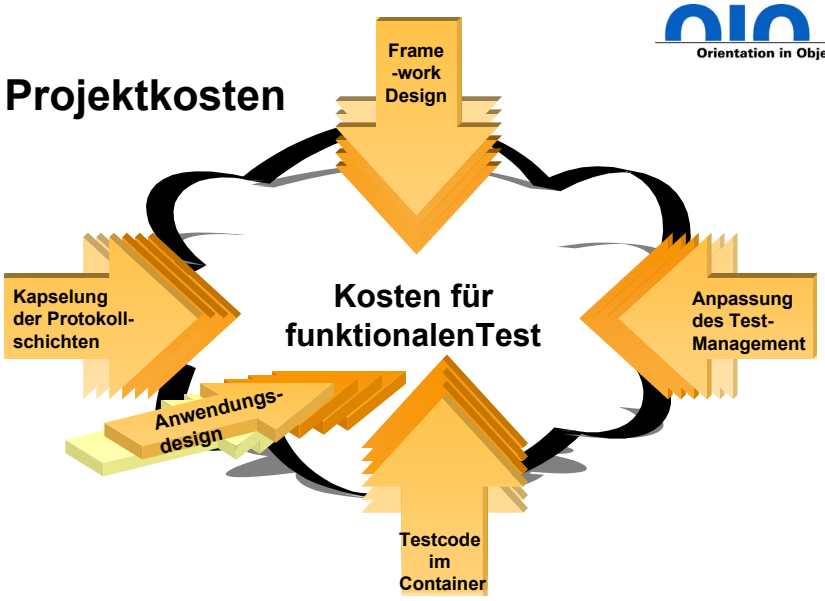
TS - Test scripts

TE - Test execution protocol

TR - Test report

Projektkosten

Orientation in Objects



Gliederung

Orientation in Objects

- Grundlagen des Qualitätsmanagements
- **1998-2008 Impressionen aus dem OIO-Qualitätsmanagement**
- Herausforderung Java 2.0 Qualtitätsmanagement
- Ausblick

Gliederung

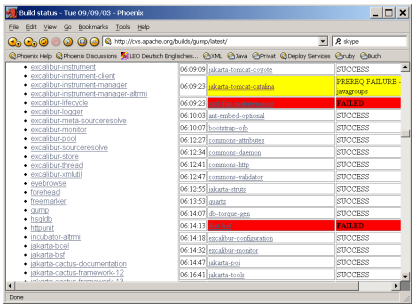
Orientation in Objects

- Grundlagen des Qualitätsmanagements
- **1998-2008 Impressionen aus dem OIO-Qualitätsmanagement**
- Herausforderung Java 2.0 Qualtitätsmanagement
- Ausblick

- **1998-2001**
- 2001-2003
- 2003-2005
- 2005-2006
- 2007-2008

QM-Trend: Open Source für kommerzielle Projekte

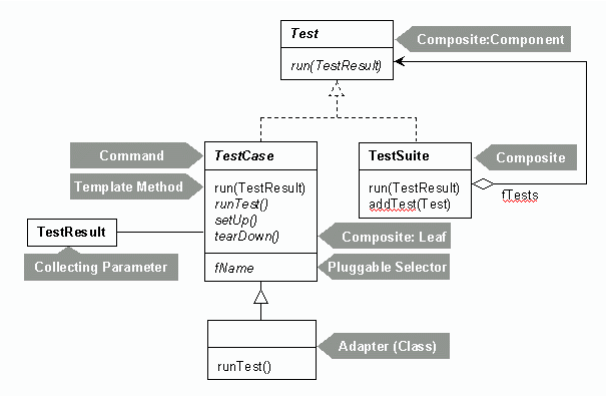
- Versionsverwaltung
- Global Ownership of Code
- Dokumentation in Mailarchiv und Code
- Bugtracking
- Continuous Integration
- Wiki



JUnit das Open Source QM-Projekt

- Entwicklung ursprünglich Gamma/Beck (Atlantikflug?)
- www.junit.org
- viele Designinformationen im Download
  - java api-doc
  - cookbook
  - cookstour
  - faq
  - testinfected

Frameworkaufbau



Quelle: „A Cooks Tour“

weitere Trends

- Erste Continuous Integration
  - Cruisecontrol
- plattformunabhängiges Buildmanagement
  - Apache ANT
- Konfigurationsmanagement
  - SCM mit CVS
  - Einsatz java Doclet-API

Gliederung

- Grundlagen des Qualitätsmanagements
- **1998-2008 Impressionen aus dem OIO-Qualitätsmanagement**
- Herausforderung Java 2.0 Qualtitätsmanagement
- Ausblick

- 1998-2001
- **2001-2003**
- 2003-2005
- 2005-2006
- 2007-2008

QM Trend: Test Driven Development

- Enterprise Extensions für Junit
  - z.B. Cactus, HttpUnit, HTMLUnit, jWebUnit, JUnitEE
- Anwendungsgebietsspezifische Erweiterung
  - DBUnit
  - JPerfUnit
- Assertions in der Plattform (jdk 1.4)
- Endoskopisches Testen
  - z. B. Mockcreator

Kostentreiber - Weyuckers Axiome

Orientation in Objects

- Antiextensionalität
- Antidecomposition
- Anticomposition

- daher Testsuite der Oberklasse reicht nicht für Unterklasse
- Aufbau einer parallelen Testklassenhierarchie

Ergebnis: JUnit Effizienz-Checklist

Orientation in Objects

- Typische Verwendung
- Grenzwerte
- Äquivalenzklassen
- Fehlerfälle/Exceptions
- DBC-Ideen
- Einfache Objektinteraktionen
- komplexe Objektinteraktionen entlang des Integrationspfades
- Typische Fehlerfälle
- gemäß Ergebniskategorien
- Zufallstestfälle

Gliederung

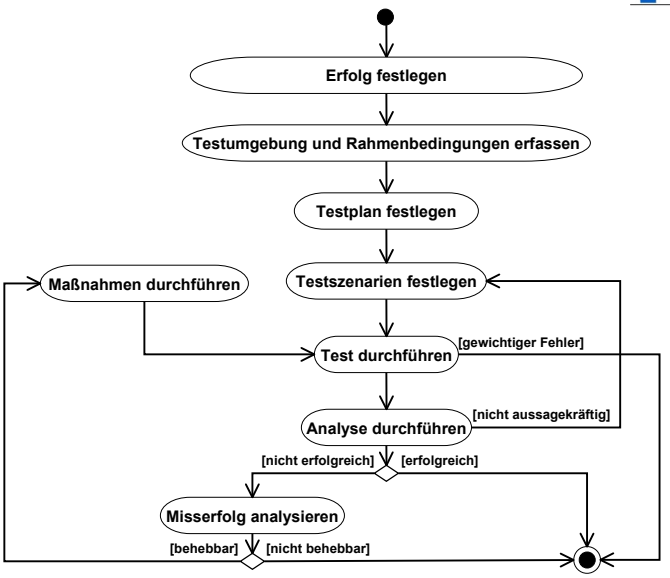
- Grundlagen des Qualitätsmanagements
- **1998-2008 Impressionen aus dem OIO-Qualitätsmanagement**
- Herausforderung Java 2.0 Qualtitätsmanagement
- Ausblick

- 1998-2001
- 2001-2003
- **2003-2005**
- 2005-2006
- 2007-2008

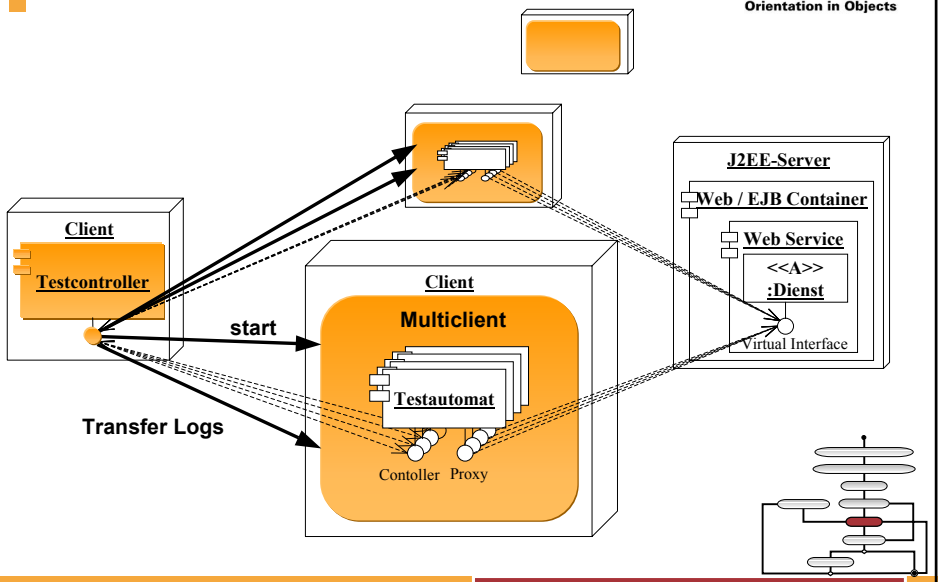
QM Trend - „become Enterprise ready“

- nichtfunktionale Qualität
  - Last- und Performanztests implementieren
  - mit Open Source Treibern wie jmeter und the Grinder
  - Analysen und Monitioring über JMX und VM Erweiterungen
- Design for Testability
  - POJO Development für Tests
  - Lightweight Container führt z.B. zu Spring
-

OIO:Testplanung der nichtfunktionalen QM



OIO: Performanz Tests mit Grinder durchführen



weitere QM Trends

Orientation in Objects

- Model Driven Architecture
  - generative Softwareentwicklung soll QM-Kosten durch konstruktive Maßnahmen senken Buildmanagement
  - muss man generierten Code testen ?
- Verbreitung von werkzeugunterstütztem Issue Tracking
  - z.B. Bugzilla
- starke Integration von QM-Werkzeugen
  - IDE als Plattform
  - z.B. Eclipse und CVS, JUnit, Lightweightcontainer..

Gliederung

Orientation in Objects

- Grundlagen des Qualitätsmanagements
- **1998-2008 Impressionen aus dem OIO-Qualitätsmanagement**
- Herausforderung Java 2.0 Qualtitätsmanagement
- Ausblick

- 1998-2001
- 2001-2003
- 2003-2005
- **2005-2006**
- 2007-2008

QM Trend: Skalierung und Modularisierung

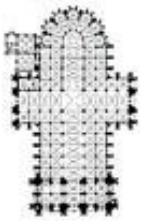
Orientation in Objects

- Lightweight Container wie z.B. Spring senken Test-Kosten
  - Developertests mit geringeren Integrationskosten
  - Herausforderung ist die Skalierbarkeit der Architektur in der nichtfunktionalen analytischen QM
- Verteiltes Buildmanagement
  - z.B. durch Maven, Ivy
  - Herausforderung:
    - **Modulorientierte und integrierte QM**
    - **Integration mit Konfigurationsmanagment**
- verbessertes Konfigurationsmanagment
  - z.B. SVN

Problem: Versionskonflikte

Orientation in Objects

Lange oder kurze Cyclen?



3.3.17 4.1.28

1.1 RC 2

2.1 0.9 Alpha

3.2.1

4.0 Beta

Eric Raymond: Release Early, Release Often

Versions-Dschungel

Orientation in Objects

- Viele Komponenten in der Laufzeit
  - Oft gibt es keine Vergleichsinstallationen
  - Lösung: Disziplin - sind neue Versionen notwendig?
  - Bundles: z.B. jboss-4.2.3_tomcat-5.5
- Viele Komponenten zur Compilezeit
  - Repositories verteilter Buildmanagementsysteme

Funktioniert Apache 2.0.45 mit
Tomcat 6.0.12 mit JK 2?
Gibt es dafür eine Anleitung?

weitere Trends

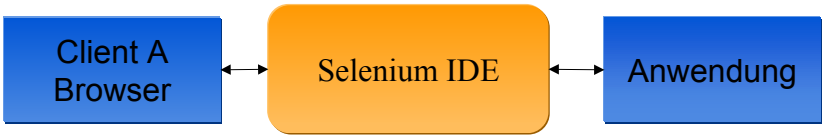
Orientation in Objects

- WEB 2.0
  - AJAX-Hype
  - Web-Mashup
  - Usergruppen akzeptieren ständiges Beta-Status
  - Herausforderung: Testwerkzeuge
    - **z.B. Canoo Webtest, Selenium**
- Analyse Werkzeuge
  - Codecoverage
    - **z.B. Cobertura**
  - Code Style
    - **z.B: Checkstyle**
  - Integration in CI
    - **z.B. Maven Report Plugins**

OIO: Selenium Remote Control

Orientation in Objects

„Selenium is a test tool for web applications. Selenium tests run directly in a browser, just as real users do. And they run in Internet Explorer, Mozilla and Firefox on Windows, Linux, and Macintosh. No other test tool covers such a wide array of platforms.“



Gliederung

Orientation in Objects

- Grundlagen des Qualitätsmanagements
- **1998-2008 Impressionen aus dem OIO-Qualitätsmanagement**
- Herausforderung Java 2.0 Qualtitätsmanagement
- Ausblick

- 1998-2001
- 2001-2003
- 2003-2005
- 2005-2006
- **2007-2008**

QM Trend: Next Generation Tools

Orientation in Objects

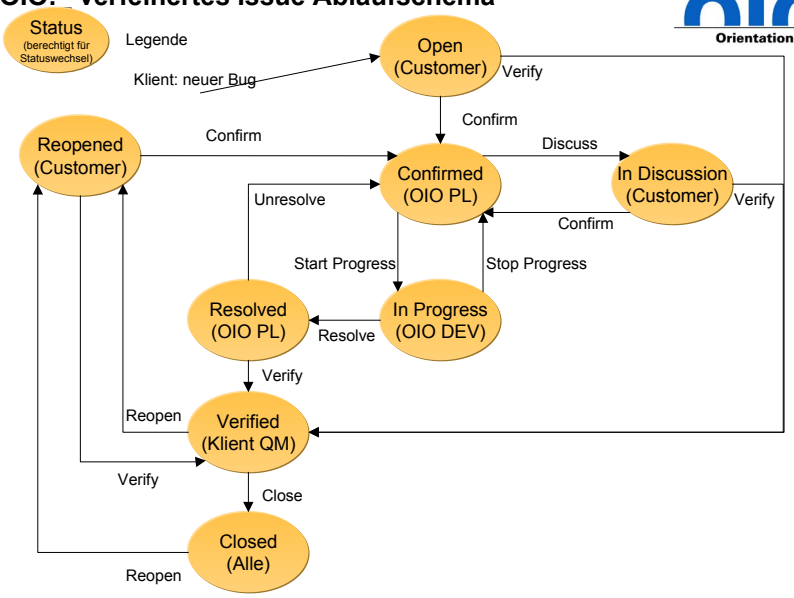
- Continuous Integration
  - z.B: Hudson
- verteiltes Buildmanagement
  - z.B. Maven2
  - verbesserte Moduldefinitionen OSGi
    - **Bundle Repository**
- Konfigurationsmanagment
  - z.B. atomare SVN-Commits
- Analysewerkzeuge
  - z.B.Clover Probability Test Reordering
- Integration in IDE
  - Issuetracker z.B. Mylin

weitere Trends

Orientation in Objects

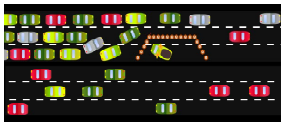
- Reife Prozesse
  - Integration Issuetracker/SCM/Projketplanungswerkzeuge/Wiki
  - Automatisiertes Architekturmanagement
    - **z.B: Hello2morrow**
- Hochintegrierte Softwareprodukte
  - Herausforderung: wie managt man die Qualität einer Software aus mehr als 100 Artefakten(„jar-Archiven“)
  - Wie
- Agile Entwicklung
  - Feature getrieben
  - kurze Releasezyklen
  - hohes Bugfixingaufkommen

OIO: verfeinertes Issue Ablaufschema

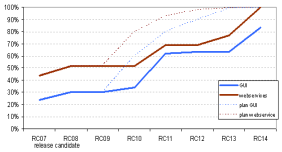


OIO: Performance Tactics

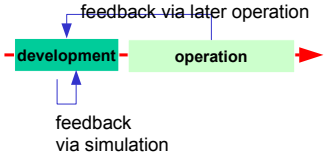
1.) Engpässe, deduktive Fehlersuche
einfache Tools für DB und memory



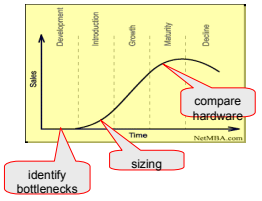
2.) Kennzahlen
agile Antwort auf „Performance afterthought“ Antipattern



3.) Simulation
Stabilität wird Prio 1 in Betriebsinfrastruktur



4.) benchmark
Skalierung an die Systemgrenzen



Gliederung

- Grundlagen des Qualitätsmanagements
- 1998-2008 Impressionen aus dem OIO-Qualitätsmanagement
- **Herausforderung Java 2.0 Qualtitätsmanagement**
- Ausblick

Java 2.0

- Dynamic Languages = Produktivität ?
  - Objektorientiert
  - dynamisch typisiert
  - einsatzspezifische Frameoworks
  - hohe Produktivität vs. Flexibilität
- Polyglot Programming
  - JVM Basis
    - **z.B. JRuby**
  - direkte Kopplung mit Java Bibliotheken und Objekten
    - **nichtfunktionale Qualitätsaspekte von Java nutzbar**
  - JSR 223
- Realtime Java
  - JSR 1 auf Solaris frei verfügbar
  - Scala ?

Polyglot Programming vs. Babylonische Zustände

Orientation in Objects

- Riesige Auswahl an Sprachen
  - z.B. Groovy,JRuby,Jython, Scala ..
  - JSR 223 Javascript, Python, Ruby ..
- DSL
  - dynamische Sprachen erlaubt Laufzeit DSLs
  - MDD macht DSLs vom Reißbrett möglich
  - populäre Spezialansätze wie z.B. GWT, JavaFX
- Technology Mashups
  - z.B. Rich UI mit AJAX, GWT und Spring/JPA Backend

Gliederung

Orientation in Objects

- Grundlagen des Qualitätsmanagements
- Qualitätsmanagement in der Java und XML Welt
- Java 2.0 Qualtitätsmanagement
- **Ausblick**

2010 ?

Orientation in Objects

- werden Mobiltelefone den PC als dominantes Verbindungsgerät ins Internet ablösen (Forrester 2006)
- weltweiter Medien und Entertainment Markt erreicht $1.83 Billionen US$ (PWC 2006)
- China will be the largest market for PCs (Forrester 2005)
- Digital Music Sales account for 40% of Recording Industry (iSuppli 2006)
- 12% of the global handset population will be TV capable with over 100 Million new Subscribers (iSuppli 2006)
- Content comes from ISVs and SPs, not operators' portals
- Nearly all applications “phone home” for updates

Java als Plattform

Orientation in Objects

- Java ist überall
- ca. 4.5 Milliarden Geräte
  - davon 1.5 Milliarden Handys
  - 800 Millionen Desktops
  - 6 Millionen Set-Top Boxen
  - 180 Operator Organisationen
- 5 Millionen Entwickler

Auswirkungen

Orientation in Objects

- Webentwicklung
  - kontinuierliche Entwicklungs- und Releasemodelle
  - inkrementelle 7-14tägige Releasezyklen
- Releasetestaufwände müssen minimiert werden
  - weitere Steigerung der QM-Aktivitäten in der Entwicklung
  - immer höhere Integrationsstufen der Developertests
  - entwicklungsbegleitende, kontinuierliche nichtfunktionale Optimierung
- Release Tests fokussieren auf
  - echte Integrationsszenarien
    - **produktorientiert nicht technologische definiert**
  - Usability

Qualitätsmanagement

Orientation in Objects

- „Qualitätsmanagement(QM) umfaßt alle Tätigkeiten der Gesamtführungsaufgabe, welche die Qualitätspolitik, ziele und Verantwortungen festlegt, sowie diese durch Mittel wie Qualitätsplanung, -sicherung und -verbesserung im Rahmen des QM-Systems verwirklichen“(ISO 8402).
- „Qualitätssicherung umfaßt alle geplanten und systematischen Tätigkeiten, die innerhalb des QM-systems verwirklicht sind,.., um angemessenes Vertrauen zu schaffen, daß eine Einheit die Qualitätsforderung erfüllen wird.“
  - konstruktive Maßnahmen
  - analytische Maßnahmen

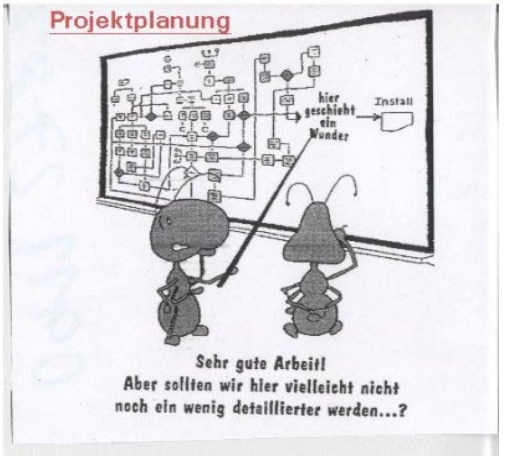
Qualitätsmanagement als Strategie – wozu eigentlich?

Orientation in Objects

Wenn das System komplex wird, kommen detaillierte Pläne zuweilen an ihre Grenzen

Dann brauchen Sie eine Strategie

Eine Strategie ist ein längerfristig **ausgerichtetes** planvolles Anstreben einer vorteilhaften Lage oder eines Ziels.
http://de.wikipedia.org/wiki/Strategie





Orientation in Objects

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0