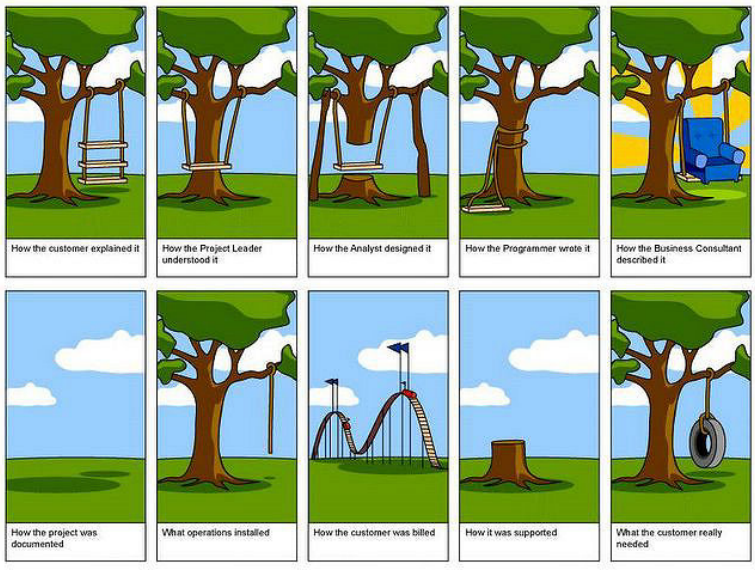
Erfolgreiche Qualitätssicherung extern entwickelter Web-Anwendungen mit modellgetriebener Testentwicklung

OOP 2007, München, 23. Jan `07, 14:00 – 14:45 Uhr

Dipl.-Inf. Peter Schnell, ALD Automotive

Dipl.-Ing. Carsten Sensler, blueCarat AG

Motivation



Motivation

Selbstverständlich verfolgen wir einen iterativen Entwicklungsprozess…

Risikomanagement ist Projektmanagement für Erwachsene
[Tom de Marco]

Wieso ? Was ist mit den Anforderungen ?
Ach so, ja, testen muss man das natürlich auch…

Peter Schnell

- Seit August 2005 **ALD Autoleasing D GmbH** in Hamburg
  Abteilungs- und Projektleiter für IT Car Financing
- 1994 bis 2005 **Gothaer Versicherungen / IDG mbH** in Göttingen
  Projektleitung und Systemanalyse
- 1988 bis 1994 Studium der Informatik an der **TU Clausthal**

- **Schwerpunkte / Erfahrungen:**
  Projektleitung mittlerer und größerer Software-Projekte im eBusiness- und MDA-Umfeld, sowie Requirements-Engineering

- **Kontakt:** Peter.Schnell@aldautomotive.com

ALD Automotive

- ALD Automotive ist eines der bedeutendsten Unternehmen im herstellerunabhängigen Full-Service-Leasing und Fuhrparkmanagement in Deutschland.
- Kennzahlen (Stand: 31.12.05) * :

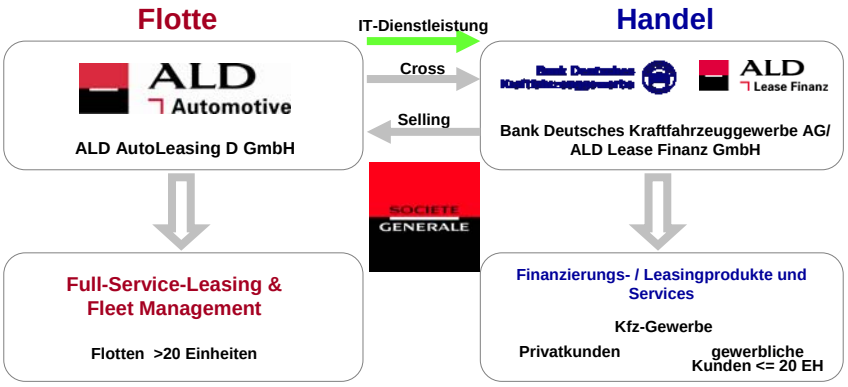
Umsatzerlöse:	403 Mio. EURO
Bestand:	93.950 Fahrzeuge
Mitarbeiter:	375
Gesellschafter:	Société Générale, Paris 100 %

- Die ALD Automotive gehört zum Konzernverbund der Société Générale, die mit 103.000 Mitarbeitern zu den größten Banken in der Eurozone zählt.
  Die ALD Automotive ist dabei in den strategischen Geschäftsbereich DSFS – Department of Specialised Financial Services – eingegliedert.

SOCIETE GENERALE

* Zahlen inkl. CPM und BP, exkl. ALD Lease Finanz, exkl. BDK

BDK – Bank Deutsches Kraftfahrzeuggewerbe



- Die BDK ist der **Auftraggeber** des im Rahmen diesen Vortrages vorgestellten Projektbeispiels

Carsten Sensler

- Seit Oktober 2005: Angestellter bei der blueCarat AG
- Studium der **Elektrotechnik** mit Studienrichtung **Technische Informatik und Internet Engineering** an der Fachhochschule Münster
  - ✓ Abschluss im Oktober 2005: Diplom-Ingenieur
- **Diplomsemester**: Oktober 2004 – April 2005 bei der IDG mbH in Göttingen
  - ✓ Diplombetreuer: Dipl.-Inf. Peter Schnell, Freimut Hennies
  - ✓ Diplomarbeit: „Generierung von Testskripten für automatische Regressionstests mit Werkzeugen und Methoden der generativen Softwareentwicklung“
- **Praxissemester**: März 2004 - Oktober 2004 bei der IDG mbH in Göttingen
- Gründer und Entwickler von oAW-Test
- Verschiedene Veröffentlichungen und Konferenzauftritte
  - ✓ OBJEKTspektrum, Javamagazin, OOP, JAX
- Kontakt:
  - ✓ Carsten.Sensler@bluecarat.de
  - ✓ Carsten.Sensler@mdtd.org

blueCarat AG

- IT-Beratungshaus ohne Hersteller- oder Produktbindung
- gegründet 2001
- ca. 50 Angestellte
- Branchenübergreifender Einsatz u.a. bei
  - ✓ Versicherungen,
  - ✓ Banken,
  - ✓ im Handel
  - ✓ und der Telekommunikation
- Standorte in Köln, Stuttgart, Frankfurt und Hamburg
- Kontakt: info@bluecarat.de, www.bluecarat.de

blueCarat Software & Consulting

Agenda

- Vorstellung
- Das Projektbeispiel: „ Online-Car-Configurator“
- Capture-/ Replay-Testverfahren
- Model-Driven Test-Development mit oAW-Test
- Einführungsstrategie von oAW-Test
- Erfolgsfaktoren und Vorteile von MDTD/ oAW-Test
- Zusammenfassung/ Ausblick
- „Live“ Demonstration

Abgrenzung

Was können wir im Rahmen dieses Vortrages leider nicht leisten ?

Leider…

- keine Einführung in die Modellgetriebene Softwareentwicklung (MDSD)
- keine Einführung in das Testen
- keine Einführung in das openArchitectureWare Generator Framework
- keine ausführliche Einführung in oAW-Test

Projektbeispiel der ALD/BDK – „Online Car-Configurator“

Aufgabenstellung:

- Unter http://www.carconfigurator.ald.de befindet sich als Web-Anwendung der **Online-Car-Configurator** zum Abschluss vom PKW-Leasingverträgen der ALD Lease Finanz GmbH und deren Kooperationspartner. Die bestehende Anwendung soll durch eine modernere Anwendung ersetzt werden.
- Es wurde ein **externer Dienstleister** ausgewählt, der die Anwendung als sog. **Werkleistung** zu einem Festpreis entwickelt und in Lizenz zur Verfügung stellt.

  Folge: Der Sourcecode der Anwendung steht nicht zur Verfügung
- Es wurde ein **iterativer, inkrementeller Entwicklungsprozess** vereinbart, der **regelmäßige Releases** vorsieht, um den Projektfortschritt zu beurteilen

  Folge: Regelmäßige, wiederkehrende Testaufgaben (Regressionen!)
- Die Anforderungsdokumente an die neue Software beschreiben u.a. auch **Nicht-Funktionale-Anforderungen** (Lastverhalten, Maskenwechselzeiten etc.)

  Folge: Regelmäßige Lasttests sind notwendig, um das Lastverhalten beurteilen zu können.

Zu lösende Probleme im Projektbeispiel

- Kein Zugriff auf den Sourcecode der Anwendung
  - ➔ White-Box Tests, wie z.B. JUnit-Tests sind nicht möglich
  - ➔ Änderungen in der Anwendung sind nur den Releasenotes o.ä. zu entnehmen, technische Verfahren, Vergleiche sind nicht möglich
- Regelmäßige, neue Releases
  - ➔ regelmäßige Regressionstests, die bestehende und neue Funktionalität sicher testen
    - ➔ im Laufe der Entwicklung ständig steigender Testaufwand
    - ➔ automatische Tests werden notwendig
    - ➔ reine Oberflächentests, da kein Sourcecode vorhanden
- Überprüfung der Nicht-funktionalen Anforderungen
  - ➔ manuell nicht oder nur mit sehr großem Aufwand leistbar
    - ➔ automatische Tests nur durch Oberflächentests realisierbar

Lösungen der Probleme im Projektbeispiel

- Problem: **Mangelnde Testressourcen und steigender Testaufwand**
  - ✓ Es war eine Test-Automatisierung dringend erforderlich
- Problem: **Hohe Komplexität der Anwendung (Funktionsumfang!)**
  - ✓ Manuell war eine umfassende Testabdeckung unmöglich
- Problem: **Oberflächentests**
  - ✓ Es war notwendig das Capture-/ Replay-Testverfahren zu nutzen, da kein Sourcecode und keine technische Dokumentation verfügbar ist

Aber:

- Das herkömmliche Capture-/ Replay-Testverfahren bringt auch Probleme mit sich.

…die Wartungsfalle droht…

ALD Automotive
OOP 2007 Software meets Business

Das Capture-/ Replay-Testverfahren

Capture -/Replay*-Testverfahren

- to capture - einfangen
- to replay – wiedergeben

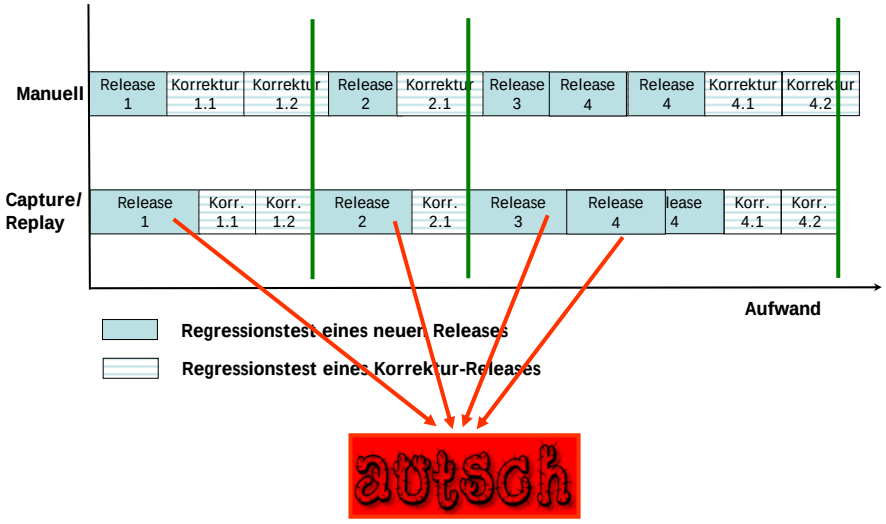
- Nur auf dem Presentation-Layer möglich
- Verfahren (theoretisch)
  - ✓ Aufzeichnung von Benutzerinteraktionen
  - ✓ Abspielen mit Vergleich auf Änderungen
- Tools (Auswahl)
  - ✓ Mercury Interactive Winrunner/ Quicktest Pro (kommerziell)
  - ✓ Rational Robot (kommerziell)
  - ✓ Bad Boy (kommerziell/ Open Source)
  - ✓ Apache JMeter (Open Source)

*CR

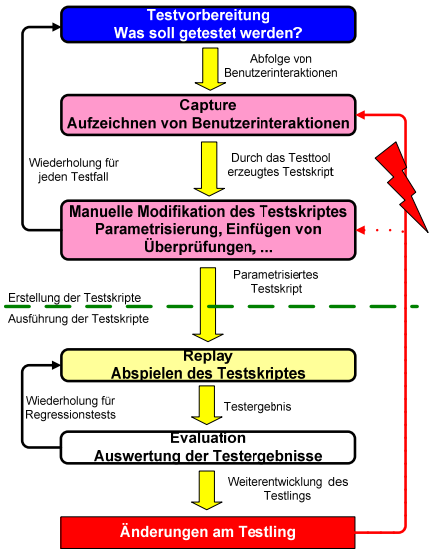
Probleme des Capture-/ Replay-Testverfahrens (1/2)

- Wartungsaufwand ist höher als zur Verfügung stehende Testzeit
- Regressionstests liegen auf dem kritischen Pfad
- Capture-Replay nur für GUI-zentrierte Tests möglich
- Erhöhter Erstellungsaufwand (ca. Faktor 2) im Vergleich zum manuellen Testen
- Erhöhter Wartungsaufwand im Vergleich zum manuellen Testen
  - ✓ Modifikationen am Testling erzwingen eine teilweise Neuerstellung der Testskripte.
  - ✓ Insbesondere bei
    - – Änderungen an der Benutzeroberfläche des Testlings
    - – Technische Änderungen des Testlings
      - Session handling
      - Login Mechanismus
- Toolgebunden, da kein Standardformat für Testskripte existiert

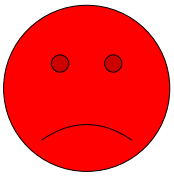
Die Wartungsfalle



Probleme des Capture-/ Replay-Testverfahrens (2/2)



- **Gefahr:** Wartungsfalle der Testskripte bei Änderungen am Testling
- Testen ist „**teuer**“ - insbesondere bei Regressionstests!



Model-Driven Test-Development mit oAW-Test bei der ALD Automotive

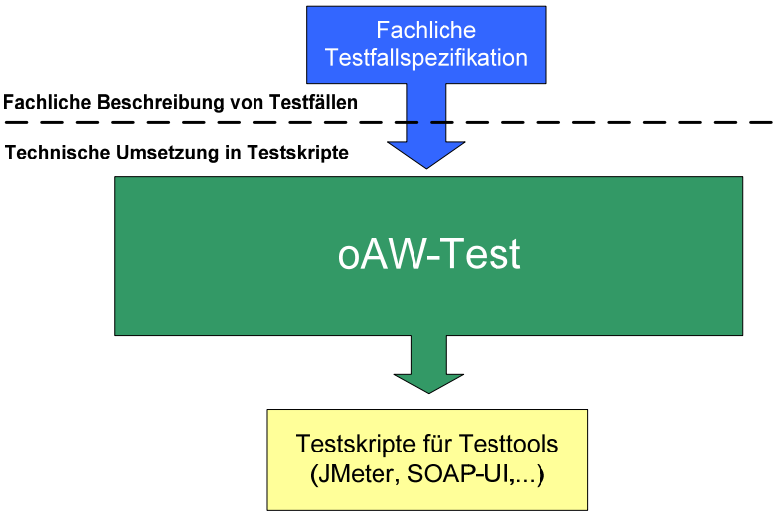
Was ist modellgetriebene Testentwicklung?

- Modellgetriebene Testentwicklung (Model-Driven Test-Development, MDTD) überträgt die Instrumente (Modelle und Transformationen) und Ziele (Standardisierung und Durchsetzung von Architektur, Erhöhung von Qualität und Effizienz usw.) von MDSD auf den Bereich der Testautomation
- Weiteres Ziel in Analogie zur „Generativen Architektur“ der MDSD: Bereitstellung einer übergreifenden, alle wesentlichen Aspekte im Bereich Testautomation abdeckende „Generativen Testarchitektur“ für ein oder mehrere Projekte
- Konkrete Syntax: UML, texttuelle DSLs, …

- Derzeitige wesentliche Inhalte
  - ✓ Referenzmodell
  - ✓ oAW-Test

- Alternative Ansätze im Bereich modellbasierter Testautomation vorhanden, aber hier nicht weiter betrachtet

Konzeption MDTD



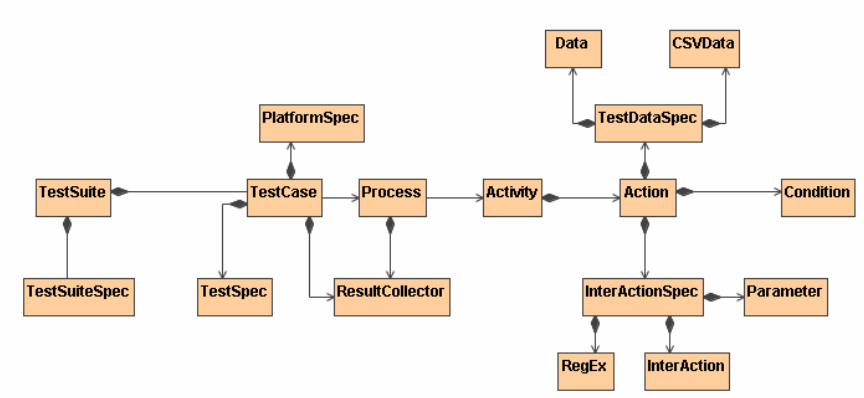
Wesentliche Ziele der Konzeption von MDTD

- Fachliche Beschreibung von Testfällen soweit möglich und sinnvoll unabhängig von
  - ✓ Teststufen (GUI-Tests, Service-Tests, …)
  - ✓ Basistechnologie (HTML, RCP, Web-Services, …)
  - ✓ Testtool (JMeter, SOAP-UI, …)
- Fachliche Beschreibung von Testfällen mit deutlich reduzierter (Tool-) Komplexität gegenüber Entwicklungsvorhaben eines Testskriptes; erstellbar durch z.B.
  - ✓ ein Testteam
  - ✓ den Fachbereich (mit geeigneter Toolunterstützung)
  - ✓ Entwickler
- **Anwendungsübergreifende**, **toolübergreifende** Wiederverwendung der fachlichen Testfallspezifikationen
- **Investitionssicherung der Aufwende für Testautomation**
  - ✓ **„Raus aus der Wartungsfalle“**

Die domänenspezifische Sprache von oAW-Test

- DSL (Domain Specific Language)
  - ✓ Dient dem Zweck, Schlüsselaspekte einer Domäne (begrenztes Wissensgebiet/ Fachgebiet) in der Sprachwelt des Domänenexperten formal auszudrücken (zu modellieren
- Bestandteile einer DSL sind (nach [STVÖ05])
  - ✓ das **Metamodell**, welches die abstrakte Syntax und die statische Semantik beschreibt,
  - ✓ die **konkrete Syntax**, welche die Notation festlegt, mit der die Modelle niedergeschrieben werden, und
  - ✓ die **Semantik**, welche die Bedeutung der Modellelemente definiert.

Das Metamodell der DSL



(vereinfachte Darstellung)

Die konkrete Syntax der DSL

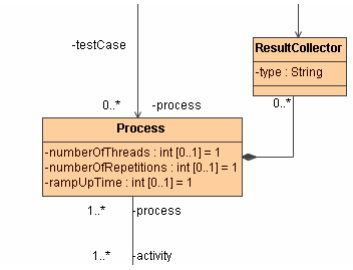
- XML
  - ✓ Leicht lesbar
  - ✓ Leicht modellierbar
  - ✓ Leicht erweiterungsfähig
  - ✓ Geringer Lernaufwand
- Variablen Konzept:
  - ✓ Variable: ${Name}
  - ✓ Wert: NAME
- Modularer Aufbau der Testfallspezifikationen (Modelle)
  - ✓ <Tag fileName = „Verzeichnis::Datei.xml“/>
  - ✓ Es können Tags inkludiert werden:
  - ✓ Es gibt keine Beschränkung der Inkludierungstiefe

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT TestSuite (TestSuiteSpec, TestCase+)>
<!ELEMENT TestSuiteSpec EMPTY>
<!ELEMENT TestCase (PlatformSpec, TestSpec,
          ResultCollector+, Process+)>
<!ELEMENT PlatformSpec EMPTY>
<!ELEMENT TestSpec EMPTY>
<!ELEMENT ResultCollector EMPTY>
<!ELEMENT Process (Activity+, ResultCollector+)>
<!ELEMENT Activity (Action+)>
<!ELEMENT Action (InterActionSpec, TestDataSpec,
          Condition+)>
<!ELEMENT TestDataSpec (Data+,, CSVData+)>
<!ELEMENT Data EMPTY>
<!ELEMENT CSVData EMPTY>
<!ELEMENT InterActionSpec (Interaction, Parameter+,
          RegEx+)>
<!ELEMENT Interaction EMPTY>
<!ELEMENT Parameter EMPTY>
<!ELEMENT RegEx EMPTY>
<!ELEMENT Condition EMPTY>
```

DTD zur DSL (vereinfacht)

```
<Process name="zweiter Thread" fileName="Process::GoogleProcess.xml"/>
```


Semantik der Modellelemente (Ausszug)




```
<Process
    numberOfThreads=" "
    numberOfRepetitions=" "
    rampUpTime=" ">
    <ResultCollector
        typ=" "/>
</Process>
```

Process:

- Wiederverwendbar
- numberOfThreads
  - ✓ Anzahl der nebenläufigen Threads (Lasttest)
- numberOfRepetitions
  - ✓ Anzahl der Wiederholungen
- rampUpTime
  - ✓ Startzeitspanne der Threads

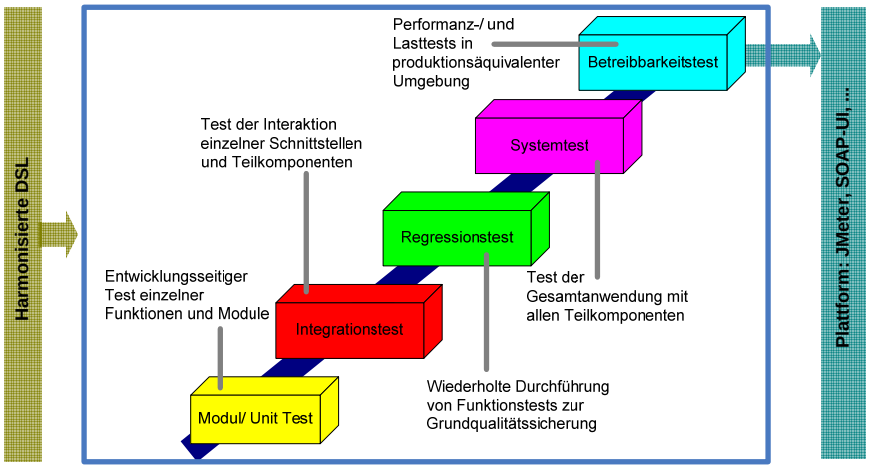
ResultCollector

- Ergebnisprotokoll-Komponente
- typ:
  - ✓ unterschiedliche Ausprägungen

oAW-Test: Allgemeines

- Open Source Projekt
  - ✓ 3 Entwickler
  - ✓ Start: Juni 2006
- Idee:
  - ✓ Diplomarbeit „Generierung von Testskripten für automatische Regressionstests mit Werkzeugen und Methoden der generativen Softwareentwicklung“
- Anschließend:
  - ✓ Erweiterungen und Umsetzung weiterer Abstraktionen und Verallgemeinerungen
    - ➔ oAW-Test

Konzeptioneller Überblick oAW-Test

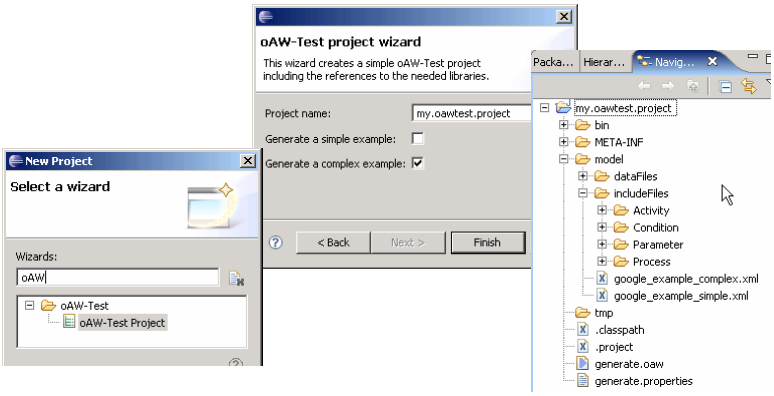


oAW-Test: Technik

- Vorraussetzung:
  - ✓ Eclipse 3.2
  - ✓ openArchitectureWare 4.1
- Testwerkzeug-Unterstützung
  - ✓ JMeter 2.1.1
  - ✓ SOAP-UI 1.6 (alpha Stadium)
- Eclipse Plugins
  - ✓ oAW-Test Eclipse Update-Site: http://www.mdtd.de/oawtest/updatesite
  - ✓ oAW-Test Core
  - ✓ oAW-Test Hilfe
  - ✓ Projekt-Wizard
  - ✓ Transformator: JMeter-Testskripte -> oAW-Test Modelle

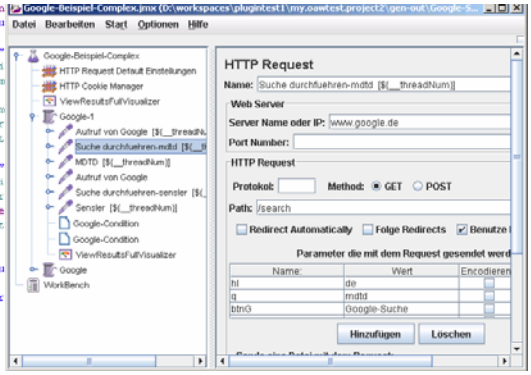
oAW-Test: Projekt-Wizard

- Erleichtert den Einstieg in oAW-Test
- Erstellt ein Projekt mit allen Abhängigkeiten und optional oAW-Test Beispielmodelle



Beispiel: oAW-Test Modell

```
<Action name="Suche durchfuehren-mdtd  [${__threadNum}]">
    <InterActionSpec>
        <Parameter value="de" technicalName="hl" />
        <Parameter value="${SUCHE}" technicalName="q" />
        <Parameter value="Google-Suche" technicalName="btnG" />
        <Parameter value="" technicalName="meta" />
        <InterAction method="GET" url="/search" server="www.google.de" />
    </InterActionSpec>
    <TestDataSpec> <Data parameter="SUCHE" value="mdtd" /> </TestDataSpec>
</Action>
```



Einführungsstrategie von oAW-Test bei der ALD Automotive

Einführungsstrategie - Allgemein

- Ermittlung des voraussichtlichen manuellen Testaufwandes
  - ➔ Funktionsumfang
  - ➔ Zeitstrecke
  - ➔ Anzahl der geplanten (Zwischen-)Releases
- Ermittlung der vorhandenen Ressourcen für Testvorbereitung und Durchführung
- Entscheidung, ob Last- und Performanztests notwendig sind
  - ➔ Wenn notwendig – wer führt diese durch – wie werden sie dokumentiert ?
- Abschätzung der daraus resultierenden Risiken für das Projekt

ALD Automotive
OOP 2007 Software meets Business

Folgende Strategie hat sich bewährt:

- Workshop mit Entscheidungsträgern - Problemsensibilisierung - Entscheidung
- Schulung Entwickler / (IT-)Testmitarbeiter (Keyplayer) anhand eines Beispiels
- Direkter Einstieg mit oAW-Test in die reale, zu testende Anwendung dabei
  - ✓ Coaching der Entwickler / (IT-)Tester durch erfahrene Mitarbeiter (intern/extern)
- Support gewährleisten:
  - ✓ Vor Ort
  - ✓ per Mail
  - ✓ durch Forum
- Coaching weiterer Entwickler / (IT-) Tester durch eingearbeitete Keyplayer

Einführung von oAW-Test bei der ALD/BDK (1/2)

Folgende Situation im Frühjahr/Sommer 2006:

- Wöchentliche Releases
- Relativ hohe Fehlerrate – wiederkehrende Fehler
- Großer Funktionsumfang
- Anfangs noch keine Stabilität in der Web-Oberfläche
- Lasttests waren nicht manuell durchführbar (300 parallele User etc.)
- Projektrisiken waren unbedingt zu begrenzen

- Herkömmliche Capture-Replay-Tests (z.B. JMeter) waren nur eingeschränkt nutzbar aufgrund häufig notwendiger Änderungen
- Personelle Ressourcen für ausführlichen, wöchentlichen, fachlichen Regressionstest waren nicht ausreichend vorhanden
- Lasttests mussten toolgestützt durchgeführt werden
- Erfahrungen mit CR-Tests waren nur eingeschränkt in der IT vorhanden
- Projektbudget war einzuhalten

Einführung von oAW-Test bei der ALD/BDK 2 (2)

Folge:

- Darstellung des Problems im Projekt-Lenkungsausschuss
- Vorstellung von Lösungsmöglichkeiten (Einsatz oAW-Test)
- Vorstellung von oAW-Test in einem Einführungs-/ Vorstellungs-workshop für Entwickler und Entscheider
- Verweis auf bereits gemachte, gute Erfahrungen
  ➔ Gothaer Versicherungen
  ➔ Unternehmen der Telekommunikationsbranche
- Entscheidung im Lenkungsausschuss für das Vorgehen
- Verpflichtung externen Dienstleisters (Carsten Sensler) zur Unterstützung und zum Coaching / Support
- Einführung direkt im realen Projekt, Schulung der Keyplayer als Multiplikatoren
- Zeitstrecke: ca. 1 Monat / Aufwand ca. 15 PT extern , 20 PT intern

Erfolgsfaktoren und Vorteile von MDTD/ oAW-Test

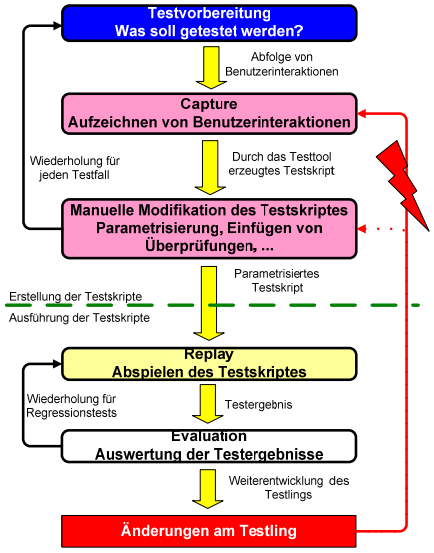
Erfolgsfaktoren - Allgemein

Wann ist es sinnvoll oAW-Test einzusetzen ?

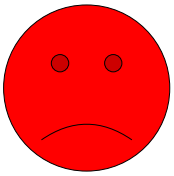
- **Mindestvoraussetzung:** Es liegt eine **Web-Anwendung** vor (noch!)
- Eine **komplexe Anwendung** bedeutet komplexe Tests, die den Initialaufwand rechtfertigen
- Keine Bedingung ist ein **modellgetriebenes Vorgehen** im Projekt
- Keine Bedingung ist ein **iteratives Vorgehen** im Projekt
  ➔ es wird aber sehr **gut durch oAW-Test unterstützt**
- Es handelt sich um **langlebige Anwendung**, die entsprechende Aufwende rechtfertigt

- **Bereitschaft** des Projektteams und des Management ist vorhanden, diesen Prozess zu unterstützen und selbst zu gehen
- Der notwendige **Initialaufwand** wird investiert

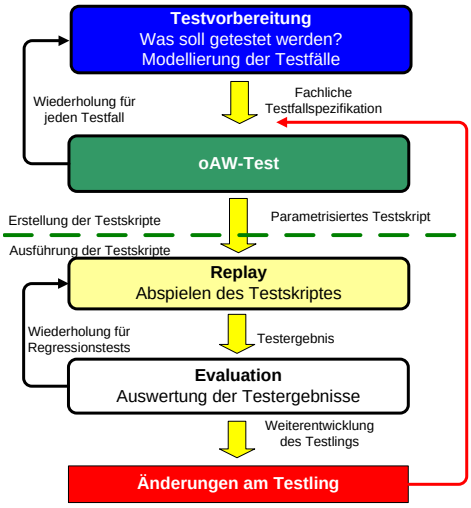
Zur Erinnerung: Probleme des Capture-/ Replay-Testverfahrens



- **Gefahr:** Wartungsfalle der Testskripte bei Änderungen am Testling
- Testen ist „**teuer**“ - insbesondere bei Regressionstests!

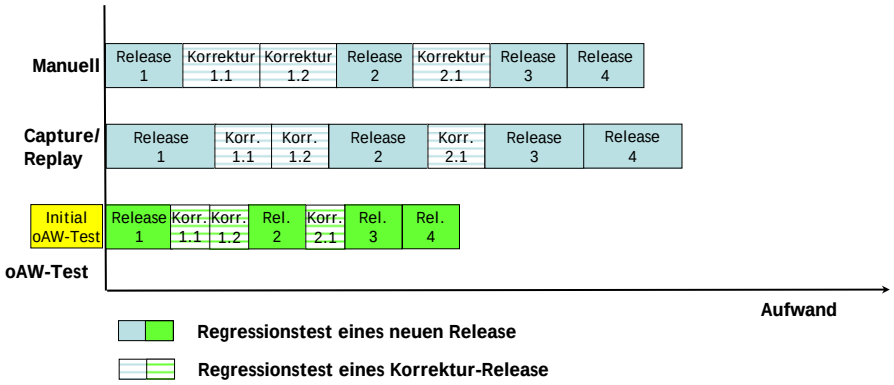


Vorgehensmodell für den Einsatz von oAW-Test



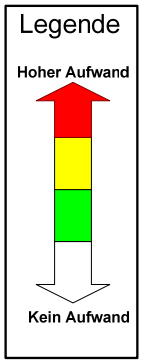
- Minderung der Wartungsfalle
- „Leichtes“ Austauschen der
  - ✓ Testart
  - ✓ Technik des Testlings
  - ✓ Testtool
- Investitionssicherung
- Aufwandsreduktion insbesondere in der Wartung

Minderung der Wartungsfalle durch den Einsatz von oAW-Test (qualitativ)



Aufwende CR-basierter und modellgetriebener Testentwicklung am Beispiel von JMeter

Änderungen am Testling	Konventionelles CR Vorgehen	Modellgetriebenes Vorgehen
Layout	Keine Änderungen	Keine Änderungen
Umbennen eines Formfeldes	Manuelle Änderungen in allen relevanten Skripten	zentrale Änderung
Hinzufügen eines Formfeldes	Manuelle Änderungen in allen relevanten Skripten	zentrale Änderung
Maskennavigation bei gleichbleibenden Events	Erneutes Aufzeichnen aller relevanten Testskripte	Änderung
Maskennavigation durch neue Transitionen	Erneutes Aufzeichnen aller relevanten Testskripte	Änderung
Neue Maske und neue Transitionen	Erneutes Aufzeichnen aller Testkripte	Änderung
Technische Umstellung der Anwendung	Erneutes Aufzeichnen aller Testkripte	zentrale Änderung



Erfolgsfaktoren – „Online Car-Configurator“ der ALD

Was waren bei der ALD/BDK die Erfolgsfaktoren im Projekt ?

- Es wurde rechtzeitig analysiert: Eine hohe Komplexität wurde identifiziert, die einen hohen, nicht mehr manuell tragbaren Testaufwand nach sich zog
- Es gab die Bereitschaft der Entscheider, „Neues zu wagen“ und die Inititalaufwende von 40 PT zu investieren, um das Risiko dadurch zu minimieren: aktives Risikomanagement !
- Es gab die Bereitschaft der Entwickler aktiv in ein neues Verfahren einzusteigen, um auch Software zu testen, die sie nicht und nur teilweise selbst entwickelt haben
  ➔ Das Verfahren ist zu komplex für eine alleinige Verantwortung im Fachbereich!
- Das Projekt profitierte von den Erfahrungen derjenigen, die bereits erfolgreich mit oAW-Test Projekte im Bereich Versicherungen und Telekommunikation durchgeführt haben
- Der Ansatz direkt im aktuellen Projekt („Learning by doing“) das Verfahren einzusetzen und die Keyplayer (hier: erfahrene Entwickler) zu schulen und zu coachen hat sich als richtig erwiesen.

ALD Automotive
OOP 2007 Software meets Business

Zusammenfassung

- Schon ab mittlerer Komplexität lohnt sich nach unserer Erfahrung der Einsatz von oAW-Test bei Web-Applikationen
  ➔ Initialaufwende sind dann sinnvoll investiert
- Der Ansatz durch Coaching und „Learning by doing“ im aktuellen Projekt (Motivation!) hat sich bewährt
- Der Einsatz von oAW-Test ist nur dann sinnvoll, wenn die Bereitschaft und die Akzeptanz bei Entscheidern und Testern wirklich vorhanden ist
- Im aktuellen Projekt hat sich der Einsatz gelohnt:
  - ✓ Die Softwarequalität ist spürbar besser geworden
  - ✓ Der Fachbereich konzentrierte sich auf die fachlich kritischen Punkte
  - ✓ Dem Dienstleister konnte mess- und reproduzierbar ein Weg zur Performanceverbesserung (bis zu Faktor 10!) aufgezeigt werden

Auch oAW-Test kann und will menschliche Tests nicht ersetzen!

Ausblick

- oAW-Test
  - ✓ DSL Editor für Eclipse
    - – Textuelle Repräsentation der oAW-Test Modelle inkl. Auflösen der „Includes“
    - – Grafische Repräsentation der oAW-Test Modelle
  - ✓ Erweiterung der Unterstützung von SOAP-UI und der Webservice Komponenten im Apache JMeter zum Testen von Web-Services
  - ✓ Unterstützung für QF-Test 2.0 (Juni 2007)

- ALD Automotive
  - ✓ 2007: Einsatz von oAW-Test in weiteren Projekten (Anwendungen)
  - ✓ In weiteren IT-Abteilungen wird oAW-Test bereits mit zwei anderen Anwendungen erfolgreich eingesetzt

  - ✓ Die ALD Automotive unterstützt aktiv das Open Source Projekt „oAW-Test“
    - – z.B. umfassende Dokumentation rund um oAW-Test

Danke!

Live Demo

Wir werden folgende Schritte „live“ demonstrieren:

- Zu testender Online-Car-Configurator wird aufgerufen
- Das oAW-Test Modell für einen Testfall wird vorgestellt
- Ein zugehöriges JMeter Testskript wird generiert
- Das JMeter Testskript wird gegen die Anwendung laufen gelassen
- Die Ergebnisse des Testlaufes werden präsentiert.

Links und Literatur

- http://www.mdtd.de : Plattform für Modellgetriebene Testentwicklung (im Aufbau)
- http://www.mdtd.de/oawtest/updatesite : oAW-Test Eclipse Update-Site
- http://www.mdtd.de/pdf/doc_01_installation.pdf : Installationsanleitung
- [STVÖ05]: T. Stahl, M. Völter, Modellgetriebene Softwareentwicklung, d.punkt Verlag, 2005, http://www.mdsd-buch.de
- C. Sensler, M. Kunz, P. Schnell, Testautomatisierung mit modellgetriebener Testskriptentwicklung, OBJEKTspektrum 3-2006 http://www.sigs.de/publications/os/2006/03/kunz_sensler_OS_03_06.pdf
- **upcoming**: Javamagazin Ausgabe 03/07 und 04/07: Modellgetriebene Testentwicklung, Konzeption und praktische Umsetzung

Fragen und Diskussionen