# Raus aus der Wartungsfalle - so machen Regressionstests Spaß

## Modellgetriebene Testentwicklung (MDTD)

Dipl.-Ing. Carsten Sensler, blueCarat AG

Freimut Hennies, IDG mbH



# Agenda

- Vorstellung
- Abgrenzung
- Grundlagen
- Die Wartungsfalle
- Raus aus der Wartungsfalle – Modellgetriebene Testentwicklung
  - Die domänenspezifische Sprache
  - Modelle und Transformationen
  - Nutzen der modellgetriebenen Testentwicklung
- Technische Umsetzung
- Der Domäneneditor „oAW-Xplor“
- Fazit
  - Modellgetriebene Testentwicklung Vorgehensmodell
  - Vorteile von Modellgetriebener Testentwicklung
  - Erfahrung und Ausblick der IDG
- Demonstration oAW-Test Version 0.2
- Ausblick

# Vorstellung 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
- Verschiedene Veröffentlichungen und Konferenzauftritte
  - OBJEKTspektrum, OOP, JAX
- Kontakt: Carsten.Sensler@bluecarat.de




# Vorstellung 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 und Frankfurt
- Kontakt: info@bluecarat.de, www.bluecarat.de

# Vorstellung Freimut Hennies

- Seit 1991 bei der Gothaer / IDG Softwarearchitekt
  - Schwerpunkte:
    - MDSD
    - MDTD
    - J2EE
    - Webfrontend
- 1987-91 Anwendungsentwicklung bei der BASF AG
- Studium des Lehramtes für Mathematik und Sachunterricht an der Universität Göttingen
- Konferenzauftritte
  - JAX, Stuttgarter SoftwareForum, openMDA
- Kontakt: Freimut_Hennies@idg.de




# Vorstellung IDG mbH

- 100%-Tochter der Gothaer Versicherungen
- gegründet 1994
- Firmensitz in Köln
- ca. 600 Mitarbeiter
- übernimmt sämtliche IT-Aufgaben für den Gothaer Konzern:
  - Betrieb des Rechenzentrums
  - Betrieb Server, Netze etc.
  - Anwendungsentwicklung
  - IT-Projektmanagement / IT-Beratung
  - Steuerung externer Firmen und Mitarbeiter

# Warum Testen?





# Abgrenzung

- Keine Einführung in die Modellgetriebene Softwareentwicklung (MDSD)
- Keine Einführung in das Testen
- Keine Einführung in das openArchitectureWare Generator Framework
- Keine Differenzierung verschiedener Ausprägungen von Modelltransformationen

# Grundlagen



# Warum testen?

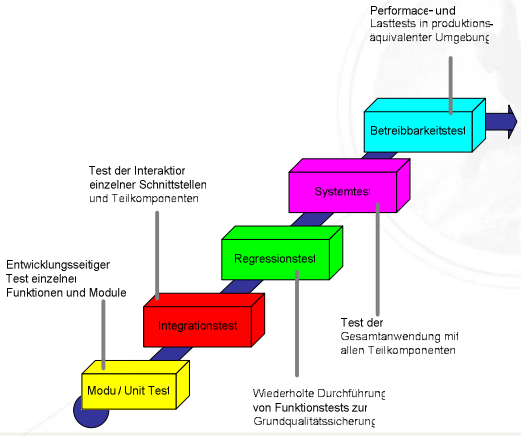
- Nachweis der Fehlerfreiheit
- Nachweis, dass eine Erfüllung der Anforderungen an das Softwaresystem vorliegt
- Nachweis, dass das Softwaresystem erwartete Reaktionen auf ein Ereignis liefert

# Der Regressionstest

- Von lat. Regression = Rückschritt
- Wiederholung aller oder einer Teilmenge aller Testfälle, um Nebenwirkungen von Modifikationen in bereits getesteten Teilen der Software aufzuspüren
- Modifikationen entstehen z. B. aufgrund von
  - Pflege und Wartung der Anwendung
  - Bugfixing
  - Weiterentwicklung



# Einordnung in den Testzyklus



# Die Wartungsfalle



# Ausgangssituation bei der IDG

- Das 1. Projekt: Postkorbanwendung
  - Projektlaufzeit ca. 2 Jahre
  - MDSD
  - J2EE, Webfrontend
  - Ca. 25 Masken
  - Beliebige Transitionen zwischen den Masken
  - Jede Woche 1 Releasebuild mit Regressionstest
  - Maximal 1 Tag Zeit für den Regressionstest
  - Testskripte im CR-Verfahren für den JMeter erstellt

# Capture -/Replay*-Verfahren

- 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







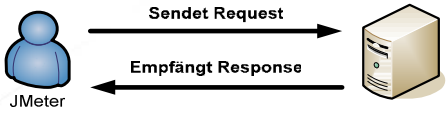
# Der Apache JMeter - Allgemeines

- Aktuelle Version 2.1.1
- Open Source im Apache Jakarta Projekt
- Javabasiertes Capture-/ Replay-Testtool
  - Aufzeichnen von Testskripten mit HTTPProxy/ Recording Controller
- Ursprünglich für Lasttests konzipiert
- Drei Ausführungsmodi
  - GUI-Mode
  - Headless-Mode
  - Server-Mode (Non-GUI)
- Textbasierte Speicherung der Testskripte (in XML)

# Funktionsweise


```
POST http://localhost:8080/SampleApp_1/j_security_check
Query data:
j_username=peter&j_password=peter

Cookie Data:
JSESSIONID=B8EE642FA1801E4FFD377390E65BA13D
```

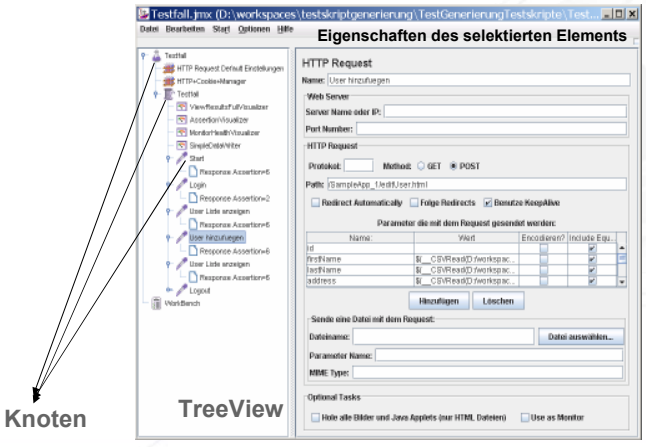









# GUI-Mode



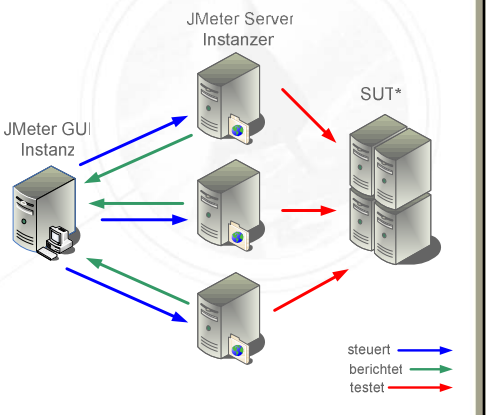
# Headless-Mode

- Steuerbar über
  - Kommandozeile
  - Ant Task (siehe Links)
- Ausführung der Testskripte, entweder
  - ein spezifisches Testskript
  - alle Testskripte in einem spezifischen Verzeichnis
- Ant Task eignet sich insbesondere, um den JMeter in Nightly Builds bzw. in einem Continuous Integration Prozeß einzubinden

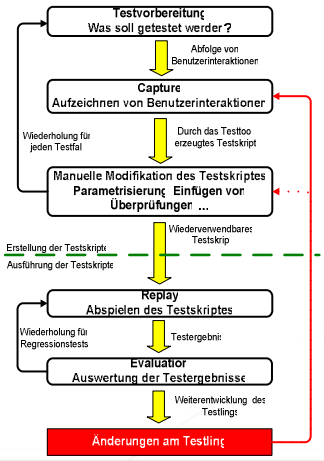



# Server-Mode (Non-GUI)

- JMeter Serverinstanzen agieren als Agenten (Stichwort: verteiltes Testen)
- Steuerung erfolgt über eine JMeter Instanz im GUI-Mode
- Vornehmlich für Lasttests von Interesse
- Kommunikation der Instanzen erfolgt per RMI



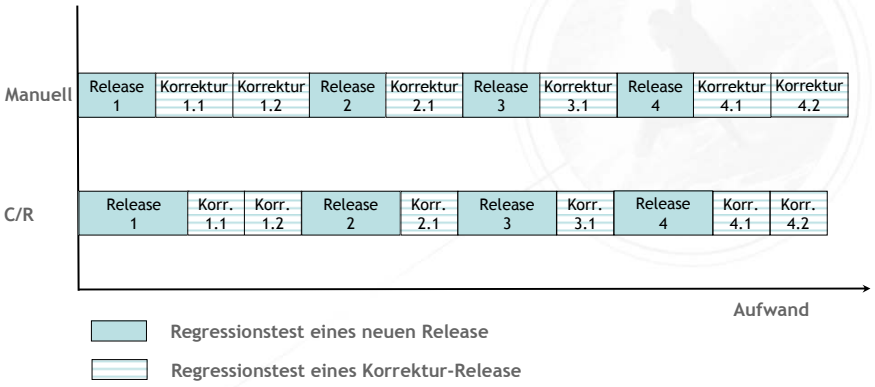
# Capture-/replay-basiertes Vorgehen



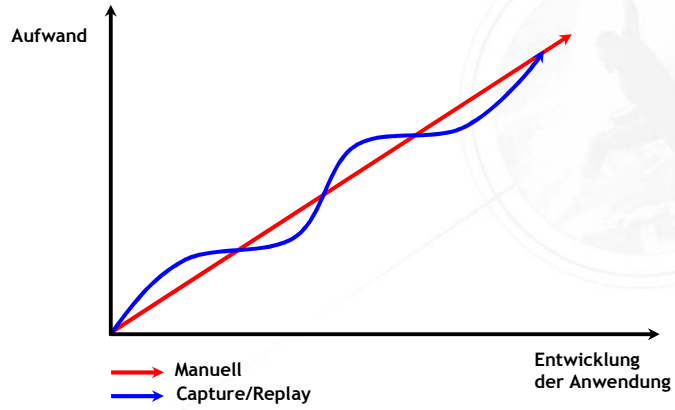

- **Gefahr:** Wartungsfalle der Testskripte bei Änderungen am Testling
- Testen ist „**teuer**“!



# Die Wartungsfalle



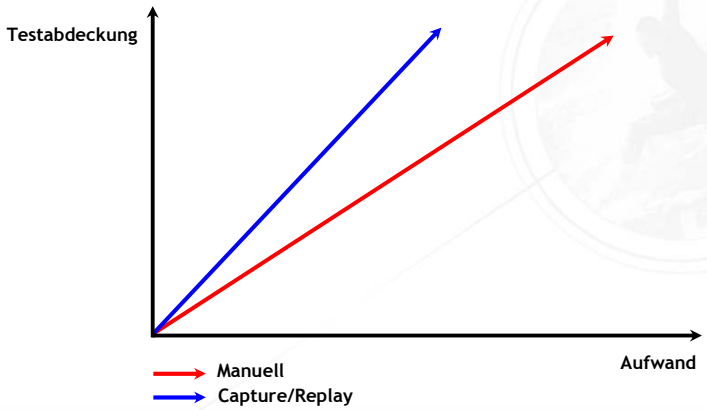
# Qualitativer Aufwandsvergleich







# Testabdeckung im Vergleich



# Probleme der Capture-Replaytechnik

- Wartungsaufwand ist höher als zur Verfügung stehende Testzeit
- Regressionstests liegen auf dem kritischen Pfad
- Erhöhter Erstellungsaufwand 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
- Toolgebunden, da kein Standardformat für Testskripte existiert

- Technische Grenzen
  - Capture-Replay nur für GUI-zentrierte Tests möglich
  - Übliche Schwierigkeiten der Capture-/ Replaytechnik
    - Layout-Überprüfungen (Anordnung von Elementen, …)
    - Layoutnahe Überprüfungen (Sortierreihenfolge, …)




# Raus aus der Wartungsfalle

Modellgetriebene Testentwicklung (MDTD)
im IDG Projektalltag

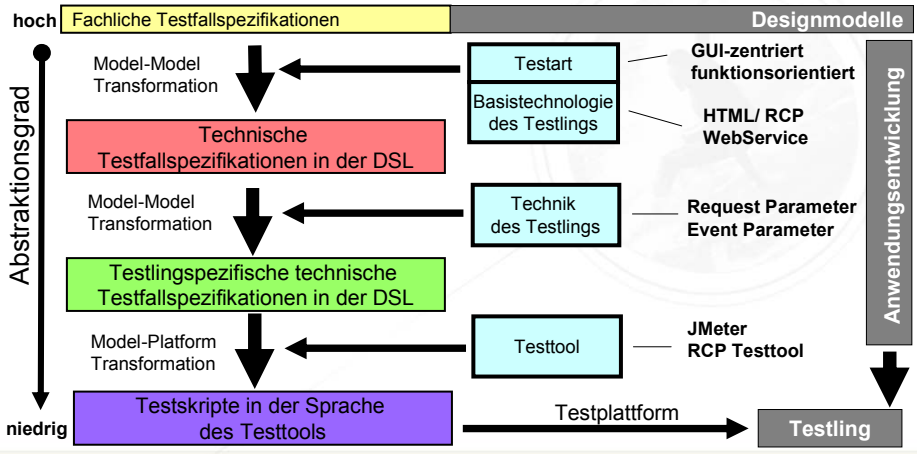
# Was ist MDTD?

- Model Driven Test Development (MDTD) bezeichnet in Anlehnung an MDSD eine modellgetriebene Entwicklung von automatisierten Tests.

- Bestandteile:
  - Domänenspezifische Sprache
  - Verschiedene Transformationen
  - Domäneneditor (geplant)




# Abstraktionen und Kaskadierung I



# Abstraktionen und Kaskadierung II

- Anzahl der Abstraktionsebenen (Kaskadierungen) (Modelle) beeinflusst die Anzahl der Transformationen.

Kaskadierter MDA-Ansatz:

…eine Abfolge von Transformationen, wobei jede Transformation als Ausgangspunkt das Ergebnis der vorhergehenden Transformation hat.





# Die domänenspezifische Sprache

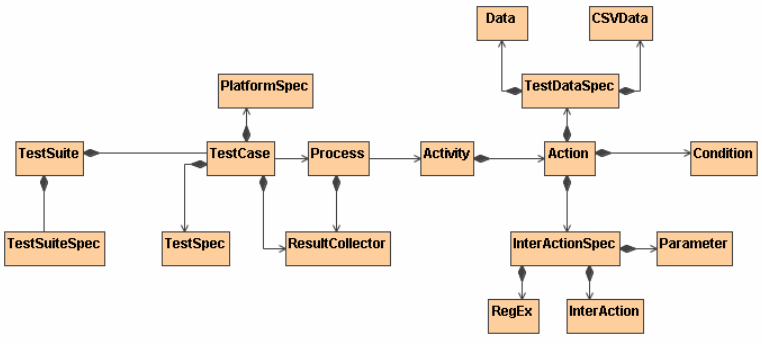
… für die Domäne Test

# Die DSL - Allgemeines

- Domäne:
  - Begrenztes Interessen- oder Wissensgebiet
  - Beispiel: Persistenz, GUI-Layout, Versicherungssparten (Leben, Kfz), Test- und Softwareentwicklung
- DSL (Domain Specific Language)
  - Dient dem Zweck, Schlüsselaspekte einer Domäne in der Sprachwelt des Domänenexperten formal auszudrücken (zu modellieren)
- Bestandteile einer domänenspezifischen Sprache (DSL) sind
  - 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.



# Metamodell der DSL




(vereinfachte Darstellung)

# Konkrete Syntax der DSL

- XML
  - Leicht lesbar
  - Leicht modellierbar
  - Leicht erweiterungsfähig
  - Verschachtelung von Artefakten möglich
    - XInclude
    - tagName**Spec**

```
<?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>
```

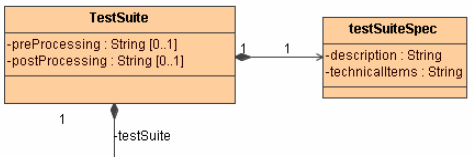
DTD zur DSL (vereinfacht)





# Semantik der Modellelemente der DSL

# TestSuite



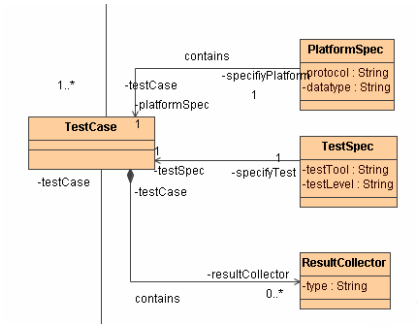

```
<TestSuite
    testSuiteSpec=" "
    preProcessing=" "
    prostProcessing= " " >
    ...
</TestSuite>
```

- Symbol:
- Abfolge von Testfällen
- Wiederverwendbar

- preProcessing
  - Bearbeitungsschritt **vor** der Testdurchführung
- postProcessing
  - Bearbeitungsschritt **nach** der Testdurchführung



# TestCase



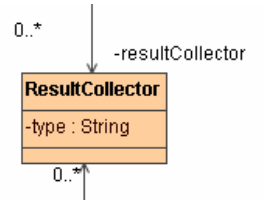

```
<TestCase
    platformSpec=" "
    testSpec=" ">
    <ResultCollector
        typ=" "/>
</TestCase>
```

- Symbol:
- Je TestCase wird ein Testskript erzeugt

- Spezifikation von
  - Basistechnologie des Testlings (platformSpec)
  - Testart (testSpec)

- Wiederverwendbar

# ResultCollector

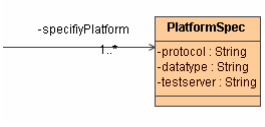



- Symbol:
- Testprotokollkomponente
- type
  - Spezifikation der Protokollfunktion
- Beispiel:
  - type="ViewResults"

```
<ResultCollector type=" "/>
```




# PlatformSpec

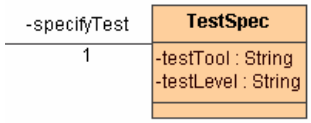



- Symbol:
- Spezifikation der Basistechnologie des Testlings
- Testserver
- Wiederverwendbar

```
<PlatformSpec protocol=" " dataType=" "
              testserver=" "/>
```

- Beispiel:
  - protocol= "HTTP"
  - dataType= "HTML"

# TestSpec

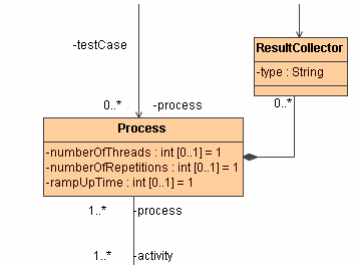



```
<TestSpec testLevel=" " testTool=" "/>
```

- Symbol:
- Spezifikation von
  - Teststufe
  - Testtool
- Beispiel
  - testLevel=“ GUI“
  - testTool=“JMeter“



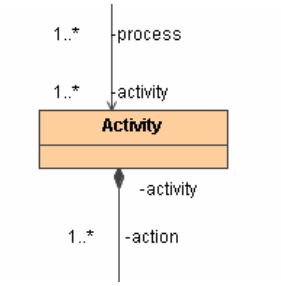
# Process




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

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

# Activity



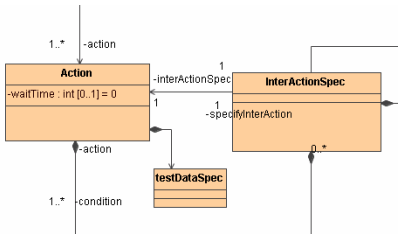

- Symbol:
- Aggregation von Aktionen
- Wiederverwendbar

- fileName
  - Verweis auf eine externe Aktivität

```
<Activity fileName="*.xml">
    ...
</Activity>
```



# Action




- Symbol:
- Elementarer Testschritt
- Wiederverwendbar

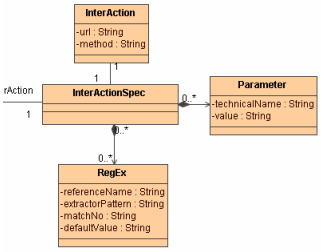
- waitTime: Zeitspanne bis zur nächsten Aktion

- Verweis auf
  - Testdaten (testDataSpec)
  - Technik des Testlings (interActionSpec)

```
<Action
    waitTime=" "
    interactionSpec=" .xml"
    testDataSpec=" .xml">

</Action>
```

# InterActionSpec

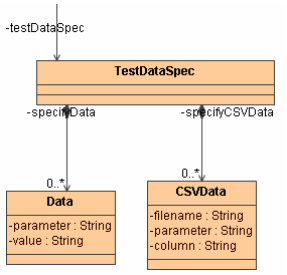



```
<Interaction
    url=" "
    method=" "
/>
<Parameter technicalName=" " value=" "/>
<RegEx
    referenceName=" "
    extractorPattern=" "
    matchtNo=" "
    defaultValue=" "
/>
</InteractionSpec>
```

- Symbol:
- Technik des Testlings
- Anwendungsspezifisch
- Werte können über symbolische Namen referenziert werden
- Definition von regulären Ausdrücken



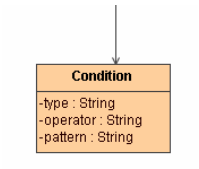

# TestDataSpec




```
<TestDataSpec>
    <Data parameter=" " value=" "/>
    <CSVData parameter=" " filename=" " column=" "/>
</TestDataSpec>
```

- Symbol:
- Data
  - Konkrete Testdaten
    - parameter
    - value
- CSVData
  - Testdaten aus einer CSV Datei
    - parameter
    - filename
    - column

# Condition




- Symbol:
- Überprüfung der Ergebnisse einer Aktion

```
<Condition
      type=" "
      operator=" "
      pattern=" ">
 </Condition>
```

- type:
  - Art der Überprüfung
- operator
- pattern
  - Prüfmuster
  - Kann mit symbolischen
  - Namen belegt werden



# Beispiel: Expandierte testlingspezifische technische Testfallspezifikation

```
<TestSuite name="Buchbestellung" preProcessing="nothing"
    postProcessing= "runCMD='xslt'">
    <TestSuiteSpec description="BeispielSuite" technicalItem=""/>
    <TestCase name="Suche Buch nach Autor">
        <PlatformSpec protocol="HTTP" dataType="HTML" testServer="192.168.1.100"/>
        <TestSpec testLevel="GUI" testTool="JMeter"/>
        <ResultCollector type="ViewResultsTree"/>
            <Process name="AutorenSuche" numberOfThreads="2" numberOfRepetitions="2" rampUpTime="50">
                <Activity name="start" fileName="buchbestellung_start.xml"/>
                <Activity name="sucheBuch">
                    <Action name="sucheNachAutor" waitTime="0">
                        <InterActionSpec>
                            <Interaction url="/buchSuche.faces" method="POST"/>
                            <Parameter technicalName="verlag" value="${VERLAG}"/>
                            <Parameter technicalName="autor" value="${AUTOR}"/>
                            <Parameter technicalName="button" value="submit"/>
                        </InterActionSpec>
                        <TestDataSpec>
                            <Data parameter="VERLAG" value="dPunkt"/>
                            <CSVData parameter="AUTOR" filename="testdata.csv" column="0"/>
                        </TestDataSpec>
                        <Condition name="Maske-Suche" type="text" operator="contains"
                                   pattern="${AUTOR}"/>
                        <Condition name="Maske-Suche" type="text" operator="contains not"
                                   pattern="Fehler"/>
                    </Action>
                </Activity>
                <Activity name="logoff" fileName="buchbestellung_stop.xml"/>
            </Process>
    </TestCase>
</TestSuite>
```

```
tesdata.csv
--------------------
Völter
Stahl
--------------------
```

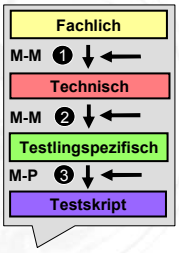
# Modelle und Transformationen

## Top-Down

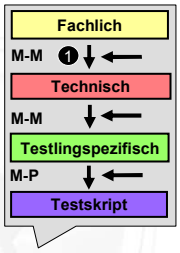


# Die Transformationen

- Je nach Ausgestaltung max. 3 Transformationen
- Jede Transformation produziert als Ergebnis den Input für die folgende Transformation.
- Eine Transformation konkretisiert (technisch und/oder anwendungsspezifisch) das Ergebnis der vorhergehenden Transformation.



# 1. Transformation



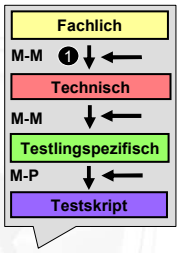

- Typ: Model – Model
- Sprache: (noch unklar)
- IN Modell:
  - Fachliche Testfallspezifikation und Designmodelle
  - Parameter:
    - Testart, Basistechnologie
- OUT Modell:
  - Technische Testfallspezifikation

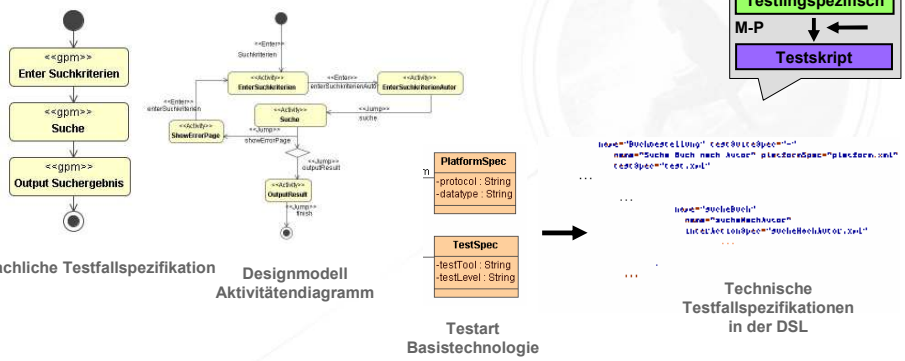






# Beispiel: 1. Transformation



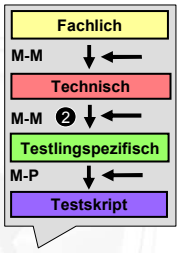

Fachliche Testfallspezifikation

Designmodell Aktivitätendiagramm

Testart Basistechnologie

Technische Testfallspezifikationen in der DSL

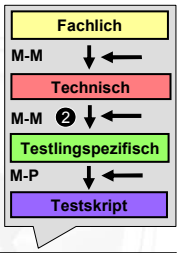
# 2. Transformation




- Typ: Model – Model
- Sprache: Java (JDom)
- IN Modell:
  - Technische Testfallspezifikation
  - Parameter:
    - Technik des Testlings (anwendungsspezifisch)
    - Testdaten
- OUT Modell:
  - Testlingspezifische technische Testfallspezifikation




# Beispiel: 2. Transformation




```
<Action name="sucheNachAutor"
    waitTime="0"
    interactionSpec="sucheNachAutor.xml"
    testDataSpec="suchNachAutorTestData.xml">
</Action>
```

Technische Testfallspezifikation

```
in sucheNachAutor.xml
<InterActionSpec>
   <Interaction url="/buchSuche.faces" method="POST"/>
   <Parameter technicalName="verlag" value="${VERLAG}"/>
   <Parameter technicalName="autor" value="${AUTOR}"/>
   <Parameter technicalName="button" value="submit"/>
</InterActionSpec>
```

Technik des Testlings

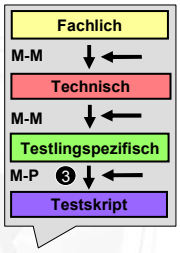
```
in suchNachAutorTestData.xml
<TestDataSpec>
   <Data parameter="VERLAG" value="dPunkt"/>
   <CSVData parameter="AUTOR" filename="testdata.csv" column="0"/>
</TestDataSpec>
```

Testdaten

```
<Action name="sucheNachAutor" waitTime="0">
   <InterActionSpec>
      <Interaction url="/buchSuche.faces" method="POST"/>
      <Parameter technicalName="verlag" value="${VERLAG}"/>
      <Parameter technicalName="autor" value="${AUTOR}"/>
      <Parameter technicalName="button" value="submit"/>
   </InterActionSpec>
   <TestDataSpec>
      <Data parameter="VERLAG" value="dPunkt"/>
      <CSVData parameter="AUTOR" filename="testdata.csv"
               column="0"/>
   </TestDataSpec>
   <Condition name="Maske-Suche" type="text"
              operator="contains"
              pattern="${AUTOR}"/>
   <Condition name="Maske-Suche" type="text"
              operator="contains not"
              pattern="Fehler"/>
</Action>
```

Testlingspezifische technische Testfallspezifikation

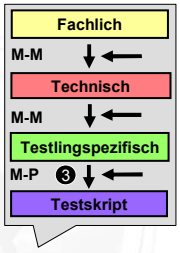
# 3. Transformation




- Typ: Model – Platform
- Sprache: XPand2
- IN Modell:
  - Testlingspezifische technische Testfallspezifikation
  - Parameter:
    - Testtool
- OUT Modell:
  - Testskript für ein spezifisches Testtool




# Beispiel: 3. Transformation

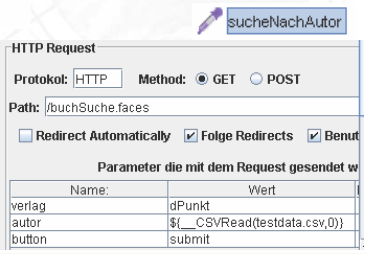



```
<Action name="sucheNachAutor" waitTime="0">
    <InterActionSpec>
        <Interaction url="/buchSuche.faces" method="POST"/>
        <Parameter technicalName="verlag" value="${VERLAG}"/>
        <Parameter technicalName="autor" value="${AUTOR}"/>
        <Parameter technicalName="button" value="submit"/>
    </InterActionSpec>
    <TestDataSpec>
        <Data parameter="VERLAG" value="dPunkt"/>
        <CSVData parameter="AUTOR" filename="testdata.csv" column="0"/>
    </TestDataSpec>
</Action>
```

Testlingspezifische technische Testfallspezifikation

```
tesdata.csv
--------------------
Völter
Stahl
--------------------
```

Testskript im JMeter



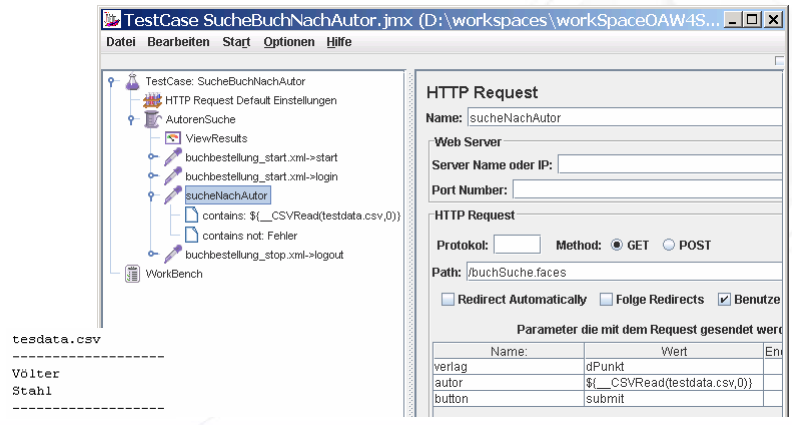
# Beispiel „Buchbestellung“ (1/2)

```
<TestSuite name="Buchbestellung" preProcessing="nothing"
    postProcessing= "runCMD='xslt'">
    <TestSuiteSpec description="BeispielSuite" technicalItem=""/>
    <TestCase name="Suche Buch nach Autor">
        <PlatformSpec protocol="HTTP" dataType="HTML" testServer="192.168.1.100"/>
        <TestSpec testLevel="GUI" testTool="JMeter"/>
        <ResultCollector type="ViewResultsTree"/>
            <Process name="AutorenSuche" numberOfThreads="2" numberOfRepetitions="2" rampUpTime="50">
                <Activity name="start" fileName="buchbestellung_start.xml"/>
                <Activity name="sucheBuch">
                    <Action name="sucheNachAutor" waitTime="0">
                        <InterActionSpec>
                            <Interaction url="/buchSuche.faces" method="POST"/>
                            <Parameter technicalName="verlag" value="${VERLAG}"/>
                            <Parameter technicalName="autor" value="${AUTOR}"/>
                            <Parameter technicalName="button" value="submit"/>
                        </InterActionSpec>
                        <TestDataSpec>
                            <Data parameter="VERLAG" value="dPunkt"/>
                            <CSVData parameter="AUTOR" filename="testdata.csv" column="0"/>
                        </TestDataSpec>
                        <Condition name="Maske-Suche" type="text" operator="contains"
                                pattern="${AUTOR}"/>
                        <Condition name="Maske-Suche" type="text" operator="contains not"
                                pattern="Fehler"/>
                    </Action>
                </Activity>
                <Activity name="logoff" fileName="buchbestellung_stop.xml"/>
            </Process>
    </TestCase>
</TestSuite>
```

```
tesdata.csv
--------------------
Völter
Stahl
--------------------
```



# Beispiel „Buchbestellung“ (2/2)




```
tesdata.csv
--------------------
Völter
Stahl
--------------------
```

# Nutzen in der Testentwicklung

- **Anwendungsübergreifende** Verwendbarkeit wesentlicher Inhalte der Transformationen
- **Testtoolübergreifende** Verwendbarkeit wesentlicher Inhalte der Transformationen
- Verwendung wesentlicher Inhalte der Transformationen und der DLS über mehrere **Basistechnologien**
- übergreifende Verwendung einer domänenzentrierten Modellierung über **Anwendungsentwicklung** und **Testentwicklung** hinweg
- Verwendung **fachlicher Testfallspezifikationen** über **Teststufen**, **Basistechnologien** und **Anwendungen** hinweg

- Kurz: maximales Maß an Wiederverwendbarkeit





# Technische Umsetzung

Bottom-Up

# openArchitectureWare (1/2)

- Ursprünglich von der b+m Informatik AG in Kiel entwickelt
- seit 2003 Open Source
- Aktuelle Version 4.0
- komplett in Java implementiert
- Leicht individuell anpassbar durch
  - Plugin Mechanismus (Version 3)
  - Workflowkonzept (Version 4)

- Ein MDA-Generatorframework, um MDA konforme Generatoren erstellen zu können.
- Definition von geschützten Bereichen für die Fachlogik möglich (Protected Regions)



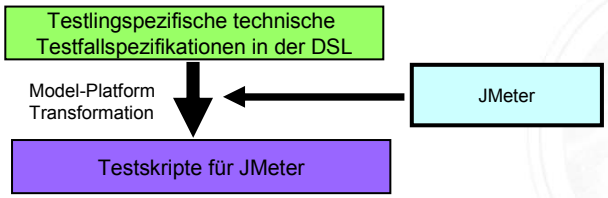


# openArchitectureWare (2/2)

- Das Generatorframework ist unabhängig vom UML-Modellierungstool
  - XMI Import Schnittstelle (XMI-Instanziator)
  - UML-Mapping Adapter + Tooladapter
  - Diverse Nachbearbeitungsmechanismen

- Templatesprache Xpand(2) zur Sourcecode-Expansion
- Wombat als Model-Model Transfomationssprache (V. 4)
- Extend
- Check
- Das Generatorframework ist nicht auf XMI als Eingabeformat fixiert
  - EMF-Modelle
  - XML-Modelle
  - Textuelle-Modelle
  - ...

# Version 0.1

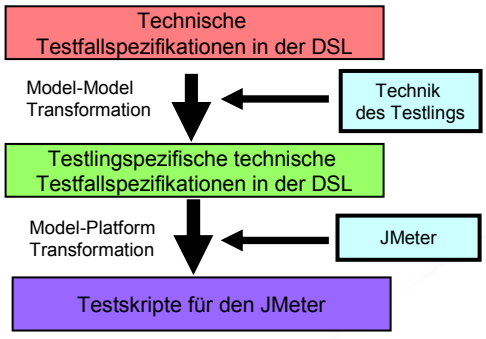




- Diplomarbeit
  - „Generierung von Testskripten für automatische Regressionstests mit Werkzeugen und Methoden der generativen Softwareentwicklung“
- Neukonzeption der DSL
- Realisierung für genau
  - eine Testart (GUI-zentriert)
  - eine Basistechnologie (HTML)
  - eine Anwendung (Gothaer Anwendung)
  - ein Testtool (JMeter)
- Technik
  - Testool
    - JMeter 2.1.1
  - MDA Generatorframework
    - openArchitectureWare 3

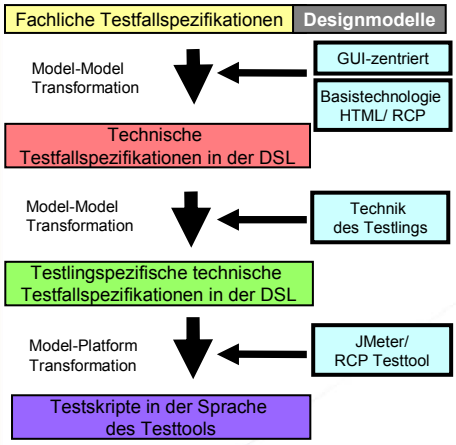



# Version 0.2




- Aktuell in der Entwicklung
- Realisierung für
  - eine Testart (GUI-zentriert)
  - eine Basistechnologie (HTML)
  - beliebige Anwendungen
  - ein Testtool (JMeter)
- Technik:
  - Testtool
    - JMeter 2.1.1
  - MDA Generatorframework
    - openArchitectureWare 4

# Version 1.0




- Geplant für Q4 2006/ Q1 2007
- Realisierung für
  - eine Testart (GUI-zentriert)
  - zwei Basistechnologien (HTML, RCP)
  - beliebige Anwendungen
  - zwei Testtools (JMeter, RCP Testtool)
- Technik:
  - Testtool
    - JMeter 2.1.1
    - RCP Testtool
  - MDA Generatorframework
    - openArchitectureWare 4



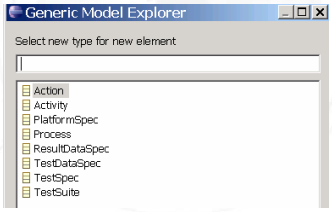


# Der Domäneneditor oAW-Xplor

Vielen Dank an Marcus Greuel.

# Domäneneditoren

- Direkte Einbettung des Typsystems der Zielplattform in einen Editor
  - z. B. Java Editor
- Effizientere Modellierung mit der DSL
- Steigerung der Akzeptanz der DSL
- Stärkere Validierungsmöglichkeiten



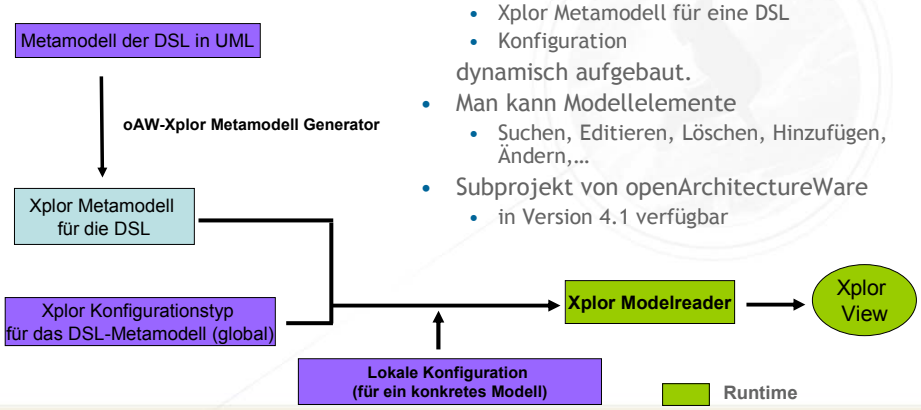




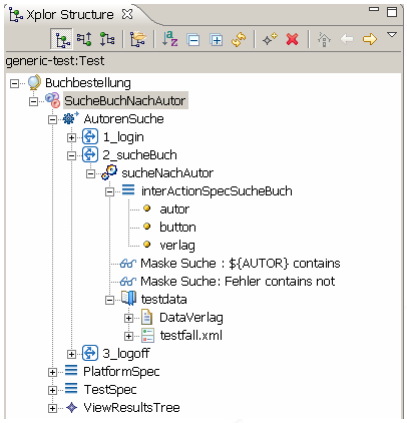
# oAW-Xplor Allgemeines

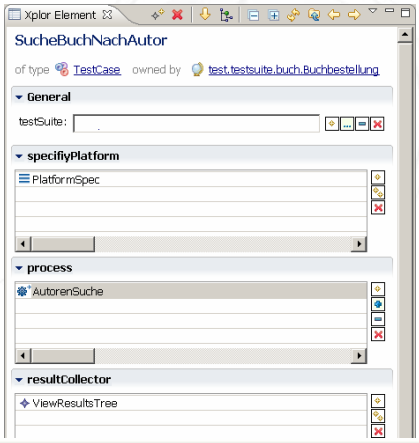
- Der Xplor-View wird zur Runtime aus
  - Xplor Metamodell für eine DSL
  - Konfiguration

  dynamisch aufgebaut.
- Man kann Modellelemente
  - Suchen, Editieren, Löschen, Hinzufügen, Ändern,…
- Subprojekt von openArchitectureWare
  - in Version 4.1 verfügbar



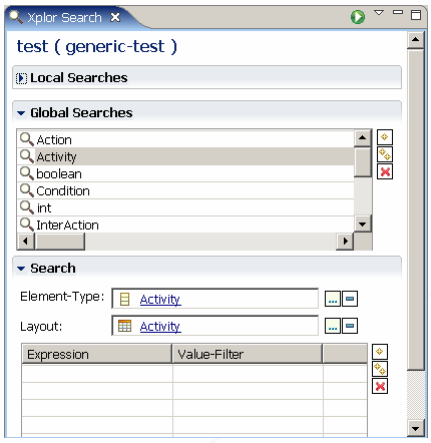
# oAW-Xplor: Structure-/Element-View

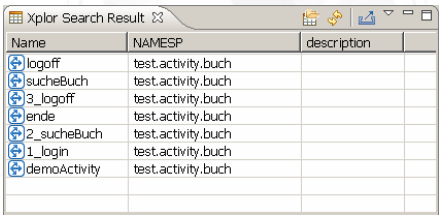


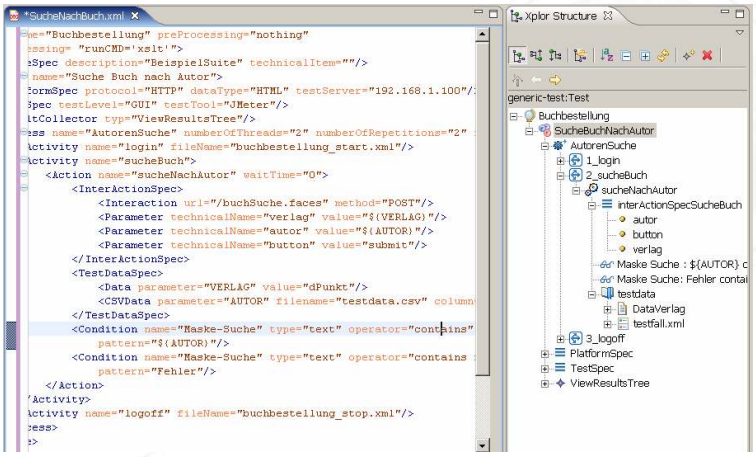



# oAW-Xplor: Search- / Searchresult-View



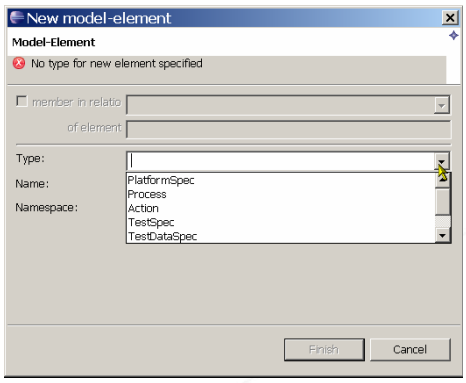



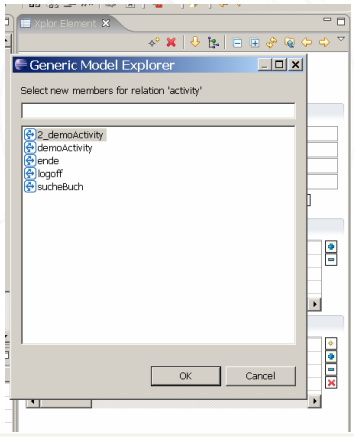
# oAW-Xplor: XML-Struktur vs. Modell-Struktur






# oAW-Xplor: New Element/ Add Element

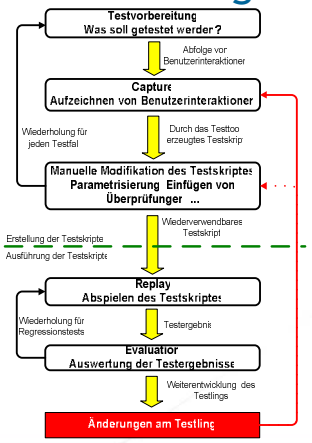


# Vorteile der Modellgetriebenen Testentwicklung

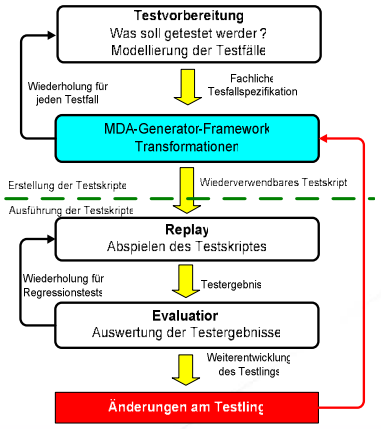


# Zur Erinnerung: CR-basiertes Vorgehen in der Testentwicklung




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

# Vorgehensmodell für den Einsatz von modellgetriebener Testentwicklung

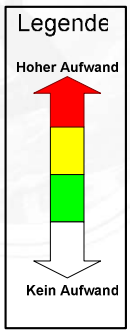



- Minderung der Wartungsfalle
- „Leichtes“ Austauschen der
  - Testart
  - Technik des Testlings
  - Testling
- Wiederverwendung

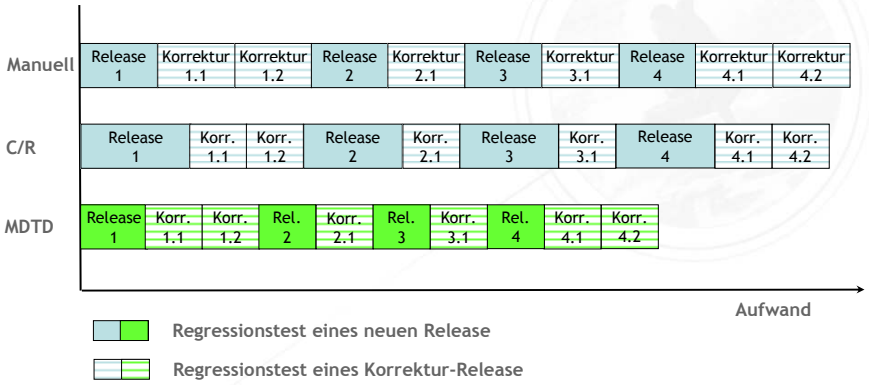



# Aufwende CR-basierter und modellgetriebener Testentwicklung (Version 0.1) am Beispiel von JMeter

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



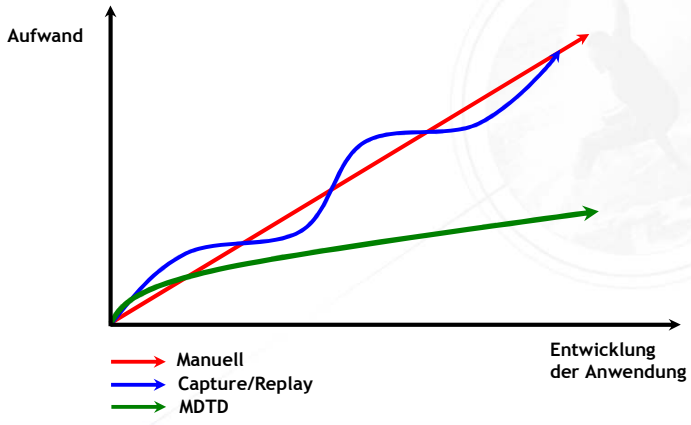
# Minderung der Wartungsfalle durch den Einsatz von MDTD



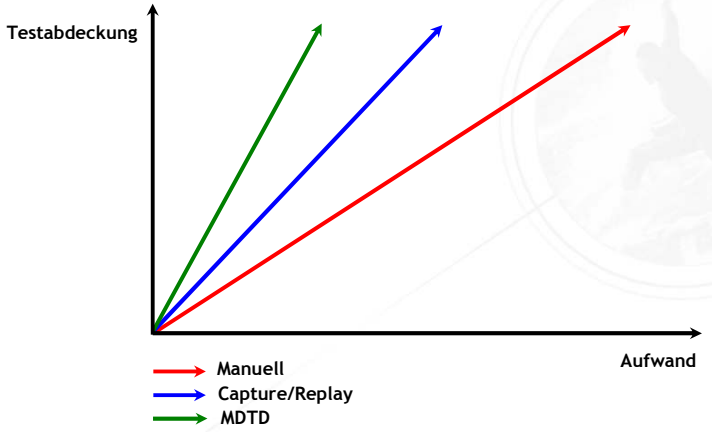




# Qualitativer Aufwandsvergleich



# Testabdeckung im Vergleich







# Erfahrungen und Ausblick der IDG

- Postkorbanwendung:
  - Umfangreiche technische Umstellung
    - 2 Tage Erweiterung der Transformation
  - Größere Maskenänderungen
    - 1-2 Stunden Änderung an der Transformation
- Abteilung „Testmanagement“ fördert die Einführung der modellgetriebenen Testentwicklung in weiteren Projekten
- Mehrere Projekte erkennen die Wartungsfalle und wollen daher die Modellgetriebene Testentwicklung nutzen
- 2007: Quasi-Standard für Webprojekte innerhalb der IDG

## Demonstration oAW-Test Version 0.2

Vielen Dank an Stefan Sensler.



## oAW-Test Version 0.2

- Realisierung für
  - eine Testart (GUI-zentriert)
  - eine Basistechnologie (HTML)
  - beliebige Anwendungen
  - ein Testtool (JMeter)
- Ab 1.6.2006 online verfügbar (weitere Infos unter www.mdtd.de)

# Ausblick

- Open-Source Projekt (Start: 01.04.2006)
  - http://www.mdtd.de
- oAW-Test Version 0.2 ab 1.6.2006 online
- Unterstützung weiterer Testarten und Zielplattformen
- Dokumentation
- Weiterentwicklung des Domänen-Editors oAW-Xplor




# Links und Literatur

- http://www.mdtd.de : Plattform für Modellgetriebene Testentwicklung (im Aufbau)
- http://www.eclipse.org/gmt/oaw : openArchitectureWare 4
- http://jakarta.apache.org/jmeter/ : Apache JMeter
- http://www.programmerplanet.org/ant-jmeter/ : Ant Task für JMeter
- http://www.w3c.org/TR/xinclude : XInclude

- 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

# Fragen und Diskussion