Modellgetriebene Testentwicklung (MDTD)

MDA in der Qualitätssicherung

Steinfurt, 19. April 2006

Dipl.-Ing. Carsten Sensler, blueCarat AG

blueCarat Software & Consulting

Freimut Hennies, IDG mbH

IDG

Warum Testen?

Vortrag

Modellgetriebene Textentwicklung (MDTD) - MDA in der Qualitätssicherung

Carsten Sensler von der blueCarat AG und Freimut Hennies von der IDG Köln geben einen Überblick über den Stand der Technik.

Datum:	Mittwoch, 19. April 2006
Kategorien:	Elektrotechnik und Informatik
Ort:	Raum D 144 der Abteilung Steinfurt-Burgsteinfurt der Fachhochschule Münster
Beginn:	17:15 bis 18:15 Uhr

Kurzbeschreibung:
Der Vortrag setzt die Reihe "Kolloquium im Fachbereich Elektrotecl

Agenda

- Vorstellung
- Abgrenzung
- Motivation
- Modellgetriebene Testentwicklung
- Die domänenspezifische Sprache
- Modelle und Transformationen
- Technische Umsetzung
- Xplor
- Die Wartungsfalle
- Ausblick

Vorstellung Carsten Sensler

- Seit Oktober 2005: Angestellter bei 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 Informationsverarbeitung und Dienstleistungen GmbH in Göttingen
  - Diplombetreuer: Dipl.-Inf. Peter Schnell, Freimut Hennies
- Praxissemester: März 2004 - Oktober 2004 bei der Informationsverarbeitung und Dienstleistungen GmbH in Göttingen
  - Einsatz in der Qualitätssicherung und Testkoordination
- Verschiedene Veröffentlichungen und Konferenzauftritte
  - OBJEKTSpektrum, OOP, JAX
- Kontakt: Carsten.Sensler@bluecarat.de

Vorstellung Freimut Hennies

- Studium: 1980–84 Lehramt Mathematik und Sachunterricht an der Universität Göttingen
- Job:
  - 1987-91 Anwendungsentwicklung bei der BASF AG
  - seit 1991 bei der Gothaer / IDG Softwarearchitekt
- Schwerpunkte:
  - MDSD
  - MDTD
  - J2EE
  - Webfrontend
- Konferenzauftitte
  - JAX, openMDA, Stuttgarter Softwareforum
- Kontakt: Freimut_Hennies@idg.de

Vorstellung: IDG mbH

- 100%-Tochter der Gothaer Versicherungen
- Firmensitz 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

Abgrenzung

- Keine Einführung in die Modellgetriebene Softwareentwicklung (MDSD)

- Keine Einführung in das Testen
- Keine Einführung in das openArchitectureWare Generator Framework

Motivation

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

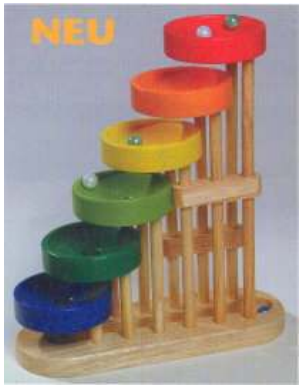
- Kurz: maximales Maß an Wiederverwendbarkeit

Modellgetriebene Testentwicklung

Was ist MDTD?

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

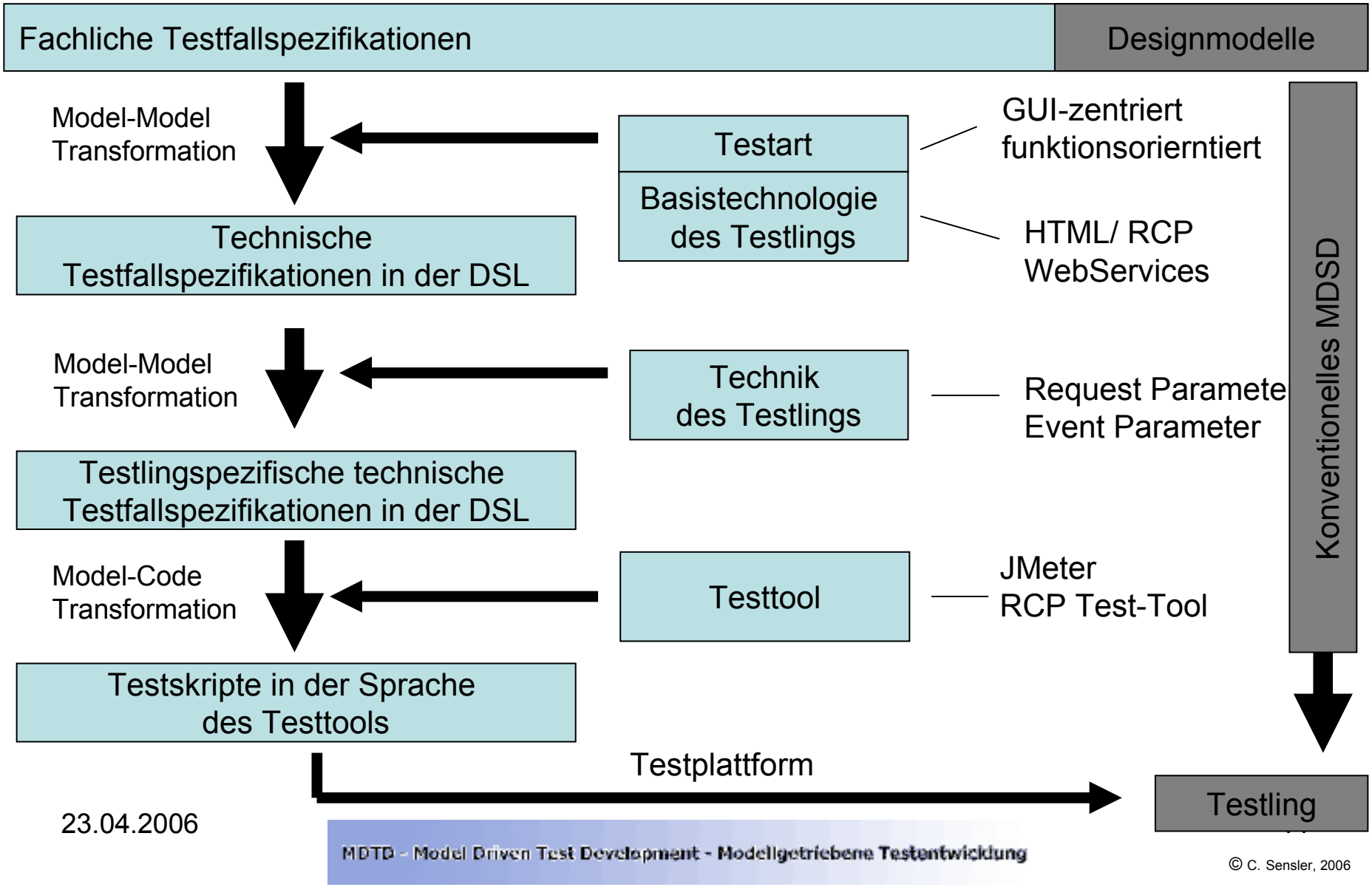
- Anzahl der Kaskadierungen (Abstraktionsebenen) (a) (Modelle) beeinflusst die Anzahl der Transformationen (a-1)



Kaskadierter MDA-Ansatz:

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

Abstraktionen und Kaskadierungen



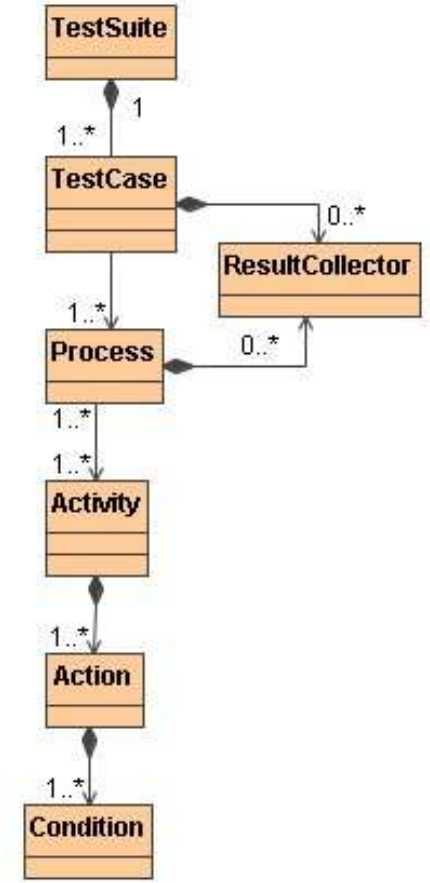
Die domänenspezifische Sprache (DSL)

für die Domäne **Test**

Die DSL - Allgemeines

- Domäne: Fachlicher Bereich
  - Beispiele
    - Versicherung, Logistik
    - Test, Anwendungsentwicklung
- Domänenspezifische Sprache (DSL)
  - Auf eine Domäne zugeschnittene Sprache
- XML
  - Leicht lesbar
  - Leicht modellierbar
  - Modular
  - Leicht erweiterungsfähig
  - Verschachtelung von Artefakten möglich
    - XInclude
    - http://www.w3.org/TR/xinclude
    - tagName**Spec**

DSL –Überblick

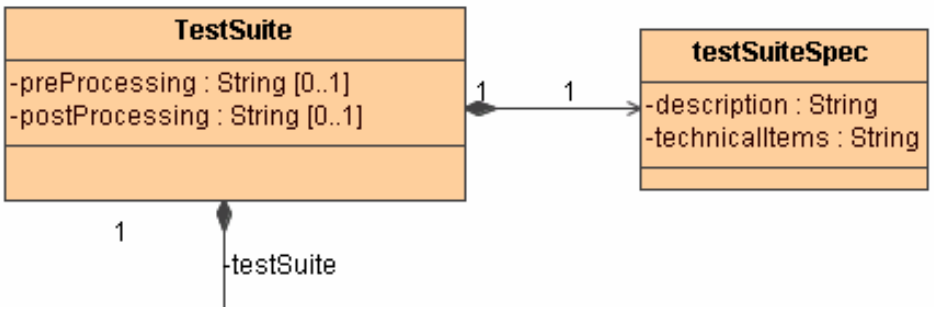


Metamodell der DSL

```
<TestSuite>
    <TestCase>
        <ResultCollector/>
        <Process>
            <ResultCollector/>
            <Activity>
                <Action>
                    <Condition/>
                </Action>
            </Activity>
        </Process>
    </TestCase>
</TestSuite>
```

DSL

TestSuite

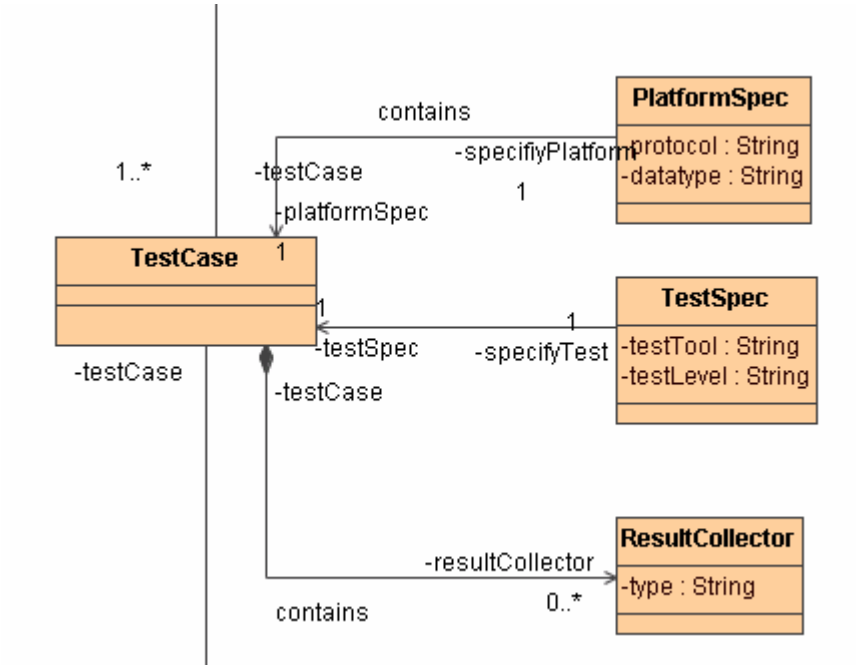



```
<TestSuite
    testSuiteSpec=" "
    preProcessing=" "
    prostProcessing= " " >
    ...
</TestSuite>
```

- Symbol: 
- Abfolge von Testfällen
- Wiederverwendbar

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

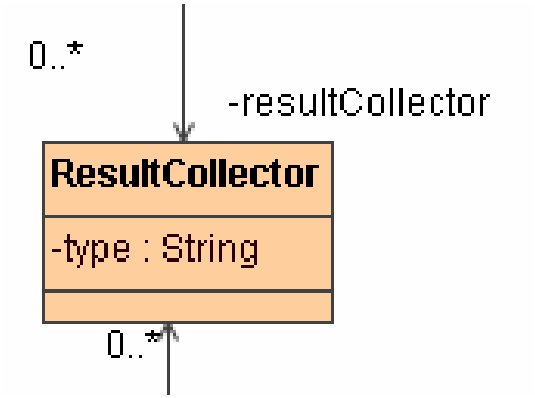
TestCase




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

- Symbol:
- Je TestCase wird ein Testskript erzeugt
- Spezifikation von
  - Basistechnologie des Testlings (platformSpec)
  - Testart (testSpec)
- Wiederverwendbar

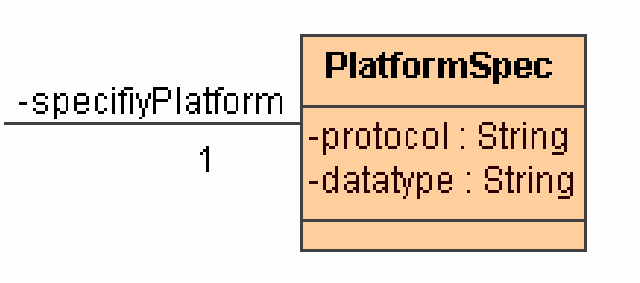
ResultCollector



- Symbol: 
- Testprotokollkomponente
- type
  - Spezifikation der Protokollfunktion

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

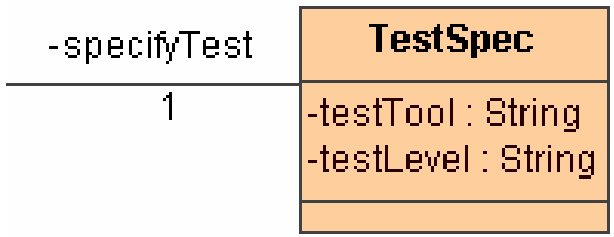
PlatformSpec



- Symbol:
- Spezifikation der Basistechnologie des Testlings
- Wiederverwendbar
- Beispiel:
  - protocol= “HTTP“
  - dataType= “HTML“

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

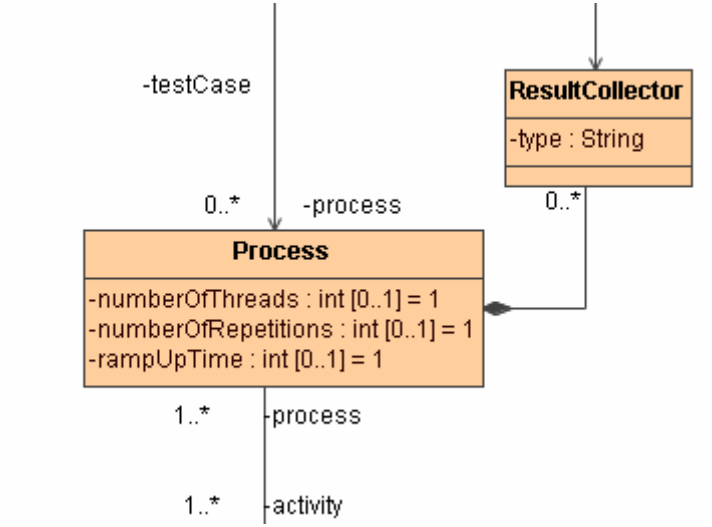
TestSpec



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

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

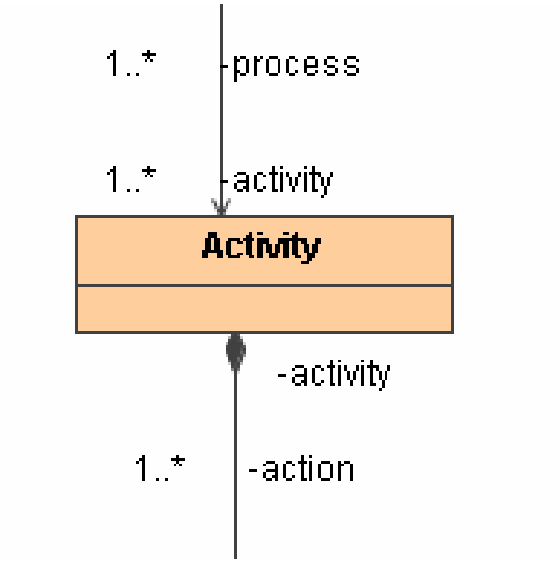
Process




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

- Symbol:
- Abgeschlossene Einheit eines Softwaresystems
- Wiederverwendbar
- numberOfThreads
  - Anzahl der nebenläufigen Threads (Lasttest)
- numberOfRepitions
  - Anzahl der Wiederholungen
- rampUpTime
  - Startzeitspanne der Threads

Activity

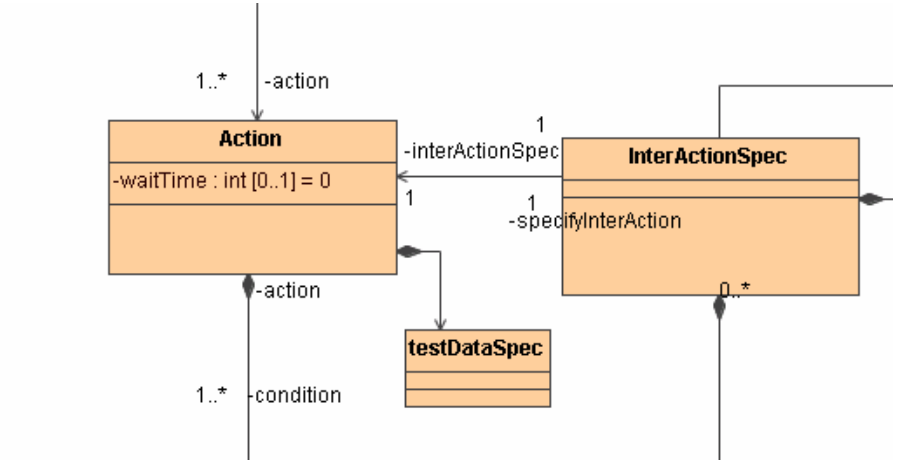


- Symbol:
- Aggregation von Aktionen
- Wiederverwendbar

- fileName
  - Verweis auf eine externe Aktivität

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

Action



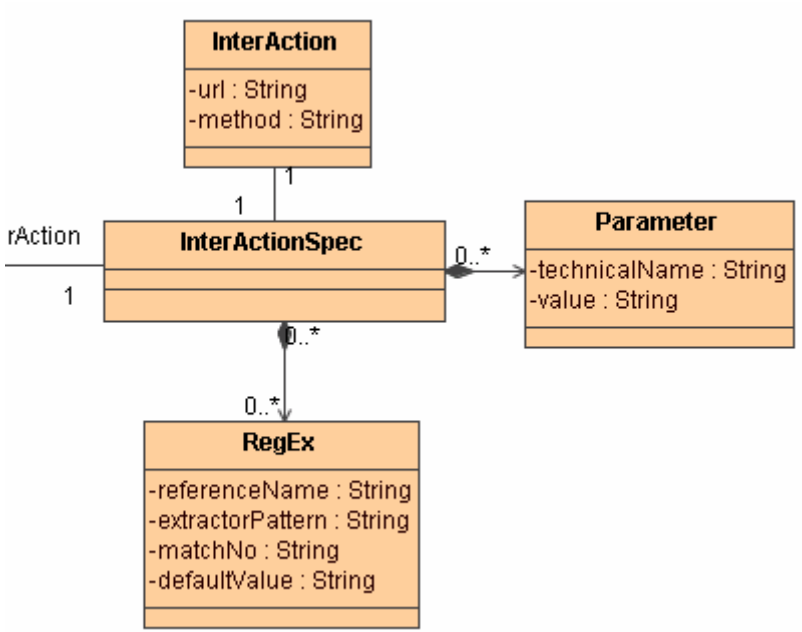

```
<Action
     waitTime=" "
     interactionSpec=" .xml"
     testDataSpec=" .xml">

</Action>
```

- Symbol:
- Elementarer Testschritt
- Wiederverwendbar

- waitTime: s.g. Zeitspanne bis zur nächsten Aktion

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

InterActionSpec

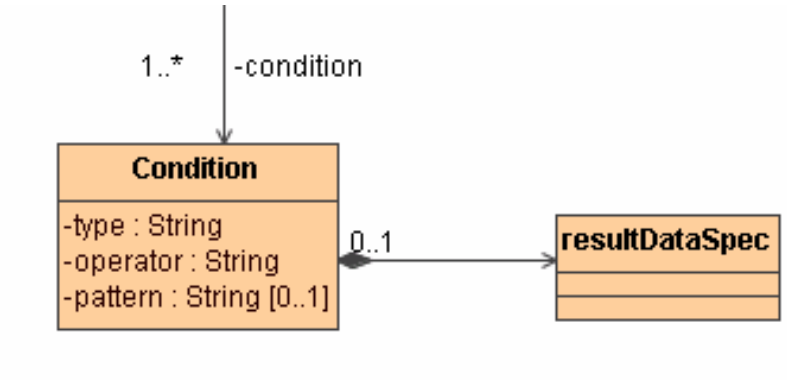


- Symbol:
- Technik des Testlings
- Anwendungsspezifisch

- Definition von regulären Ausdrücken

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

Condition




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

- Symbol:
- Überprüfung der Ergebnisse einer Aktion

- type:
  - Art der Überprüfung
- Operator
- Pattern
  - Prüfmuster

Beispiel: Expandierte testlingspezifische technische Testfallspezifikation

```
<TestSuite name="Buchbestellung" preProcessing="nothing"
    prostProcessing= "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 typ="ViewResultsTree"/>
            <Process name="Autor=Hagmann" numberOfThreads="1" 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="autor" value="Hagmann"/>
                            <Parameter technicalName="button" value="submit"/>
                        </InterActionSpec>
                        <Condition name="Maske-Suche" type="text" operator="contains"
                                   pattern="Grundlagen der Elektrotechnik"/>
                        <Condition name="Maske-Suche" type="text" operator="contains not"
                                   pattern="Fehler"/>
                    </Action>
                </Activity>
                <Activity name="logoff" fileName="buchbestellung_stop.xml"/>
            </Process>
    </TestCase>
</TestSuite>
```

Transformationen und Modelle

Top-Down

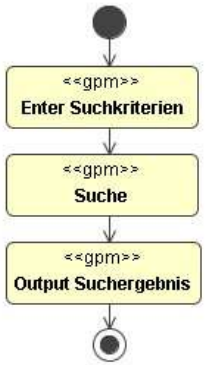
Die Transformationen

- Je nach Ausgestaltung max. 3 Transformationen

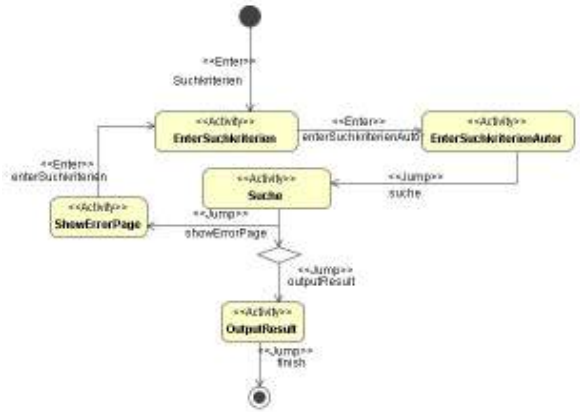
- Jede Transformation produziert als Output den Input für die folgende Transformation

1. Transformation

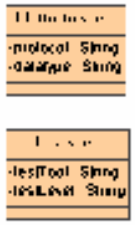
- Typ: Model – Model
- Sprache: Wombat
- IN Modell: Testfallspezifikation und Designmodelle
  - Parameter: Testart, Basistechnologie
- OUT Modell: Technische Testfallspezifikation



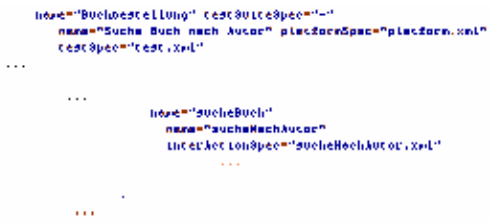
Testfallspezifikation



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)
- OUT Modell: Testlingspezifische technische Testfallspezifikation

```
<Action name="sucheNachAutor"
        waitTime="0"
        interActionSpec="sucheNachAutor.xml">
 </Action>
```

Technische Testfallspezifikation

```
in sucheNachAutor.xml:
<InterActionSpec>
    <Interaction url="/buchSuche.faces" method="POST"    />
    <Parameter technicalName="autor" value="Hagmann"/>
    <Parameter technicalName="button" value="submit"/>
</InterActionSpec>
```

Technik des Testlings

```
<Action name="sucheNachAutor" waitTime="0">
    <InterActionSpec>
        <Interaction url="/buchSuche.faces" method="POST"    />
        <Parameter technicalName="autor" value="Hagmann"/>
        <Parameter technicalName="button" value="submit"/>
    </InterActionSpec>
</Action>
```

Testlingspezifische technische Testfallspezifikation

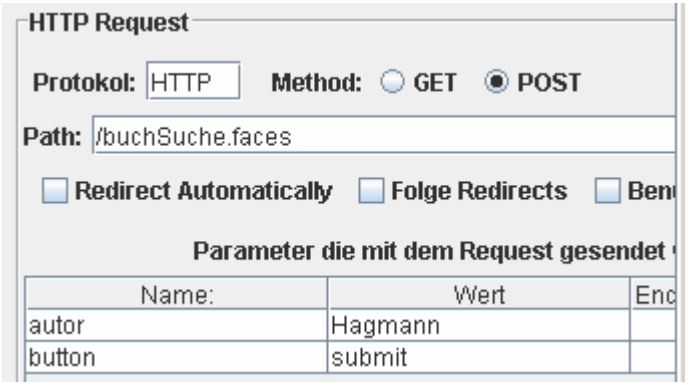
3. Transformation

- Typ: Model – Code
- Sprache: XPand2 (oAW4 Templatesprache)
- IN Modell: Testlingspezifische technische Testfallspezifikation
  - Parameter: Testtool
- OUT : Testskript für ein spezifisches Testtool

```
<Action name="sucheNachAutor" waitTime="0">
    <InterActionSpec>
        <Interaction url="/buchSuche.faces" method="POST"    />
        <Parameter technicalName="autor" value="Hagmann"/>
        <Parameter technicalName="button" value="submit"/>
    </InterActionSpec>
</Action>
```

Testlingspezifische technische Testfallspezifikation

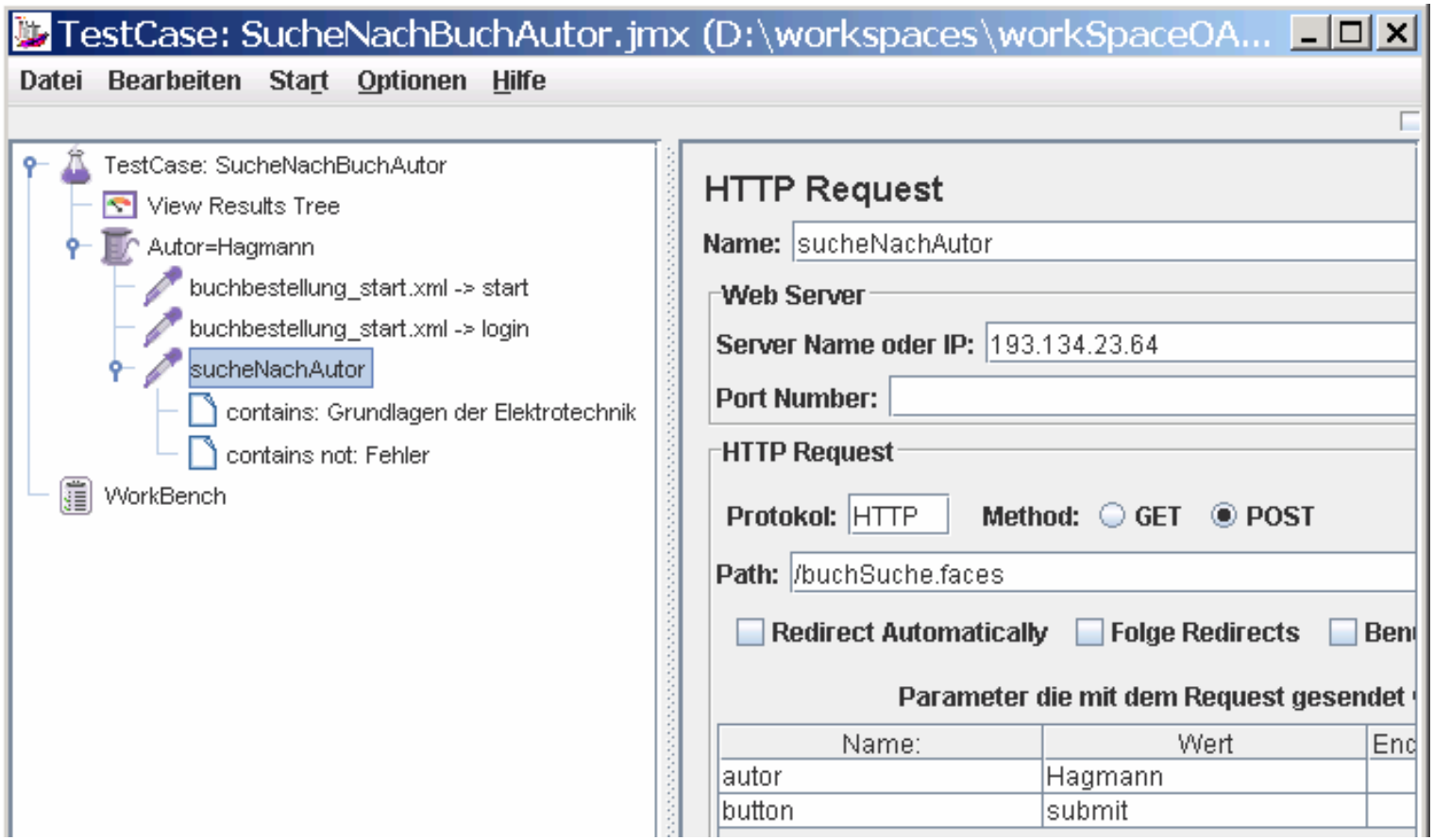
Testskript im JMeter



Beispiel „Buchbestellung“ (1/2)

```
<TestSuite name="Buchbestellung" preProcessing="nothing"
    prostProcessing= "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 typ="ViewResultsTree"/>
            <Process name="Autor=Hagmann" numberOfThreads="1" 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="autor" value="Hagmann"/>
                            <Parameter technicalName="button" value="submit"/>
                        </InterActionSpec>
                        <Condition name="Maske-Suche" type="text" operator="contains"
                                   pattern="Grundlagen der Elektrotechnik"/>
                        <Condition name="Maske-Suche" type="text" operator="contains not"
                                   pattern="Fehler"/>
                    </Action>
                </Activity>
                <Activity name="logoff" fileName="buchbestellung_stop.xml"/>
            </Process>
    </TestCase>
</TestSuite>
```

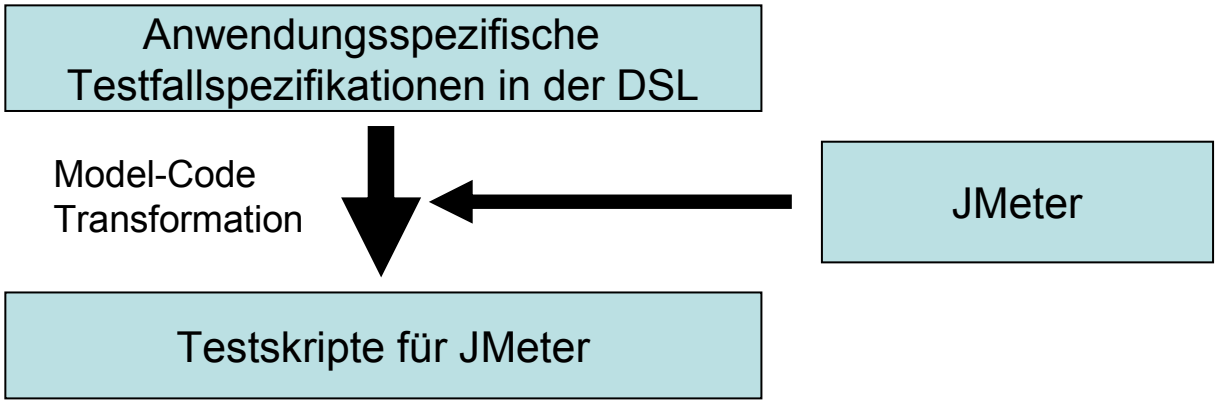
Beispiel „Buchbestellung“ (2/2)



Technische Umsetzung

Bottom-Up

Version 0.1

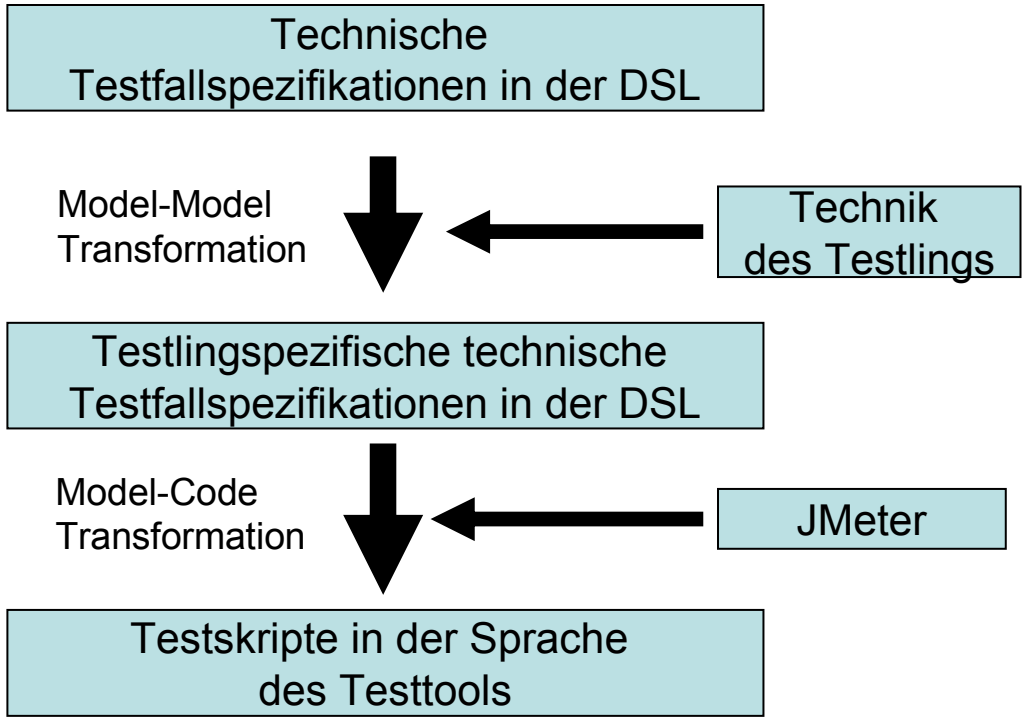


Diplomarbeit

- „Generierung von Testskripten für automatische Regressionstests mit Werkzeugen und Methoden der generativen Softwareentwicklung“

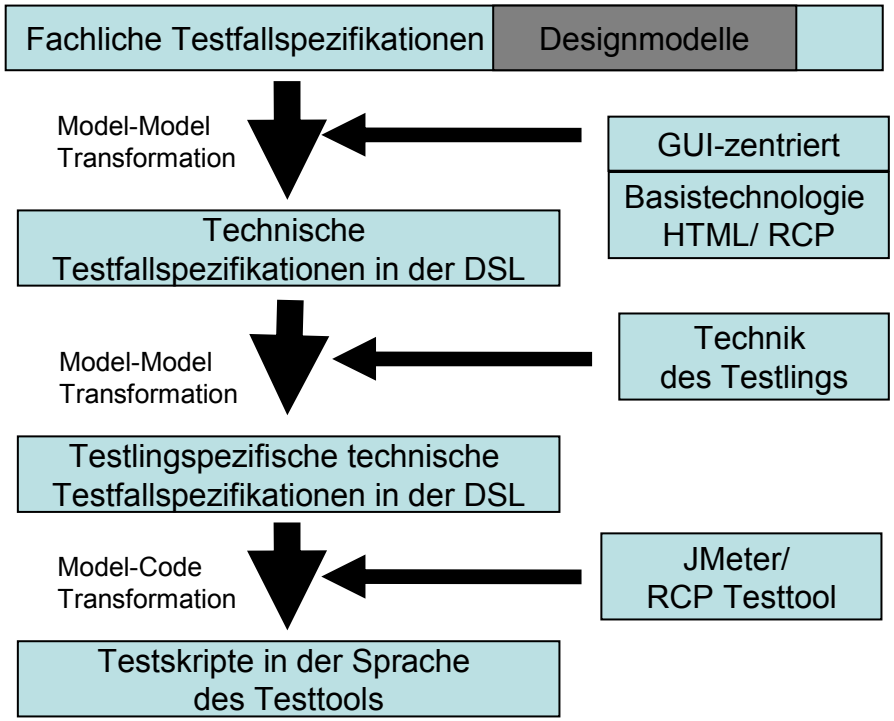
- Neukonzeption der DSL
- Realisierung für genau
  - eine Testart (GUI-zentriert)
  - eine Basistechnologie (HTML)
  - eine Anwendung (Gothaer Anwendung)
  - ein Testtool (JMeter)

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



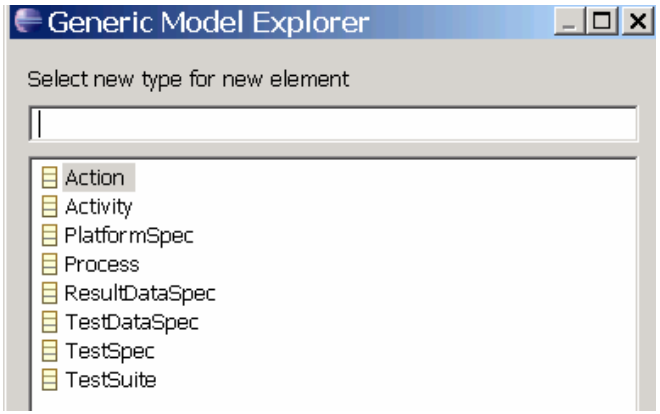
- Geplant für Q4 2006/ Q1 2007
- Realisierung für
  - eine Testart (GUI-zentriert)
  - zwei Basistechnologien (HTML, RCP)
  - beliebige Anwendungen
  - zwei Testtools (JMeter, RCP Testtool)
- Technik:
  - Testtool
    - JMeter 2.1.1
    - RCP Testtool
  - MDA Generatorframework
    - openArchitectureWare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

Der Domäneneditor Xplor

Vielen Dank an Marcus Greuel!

Domäneneditoren

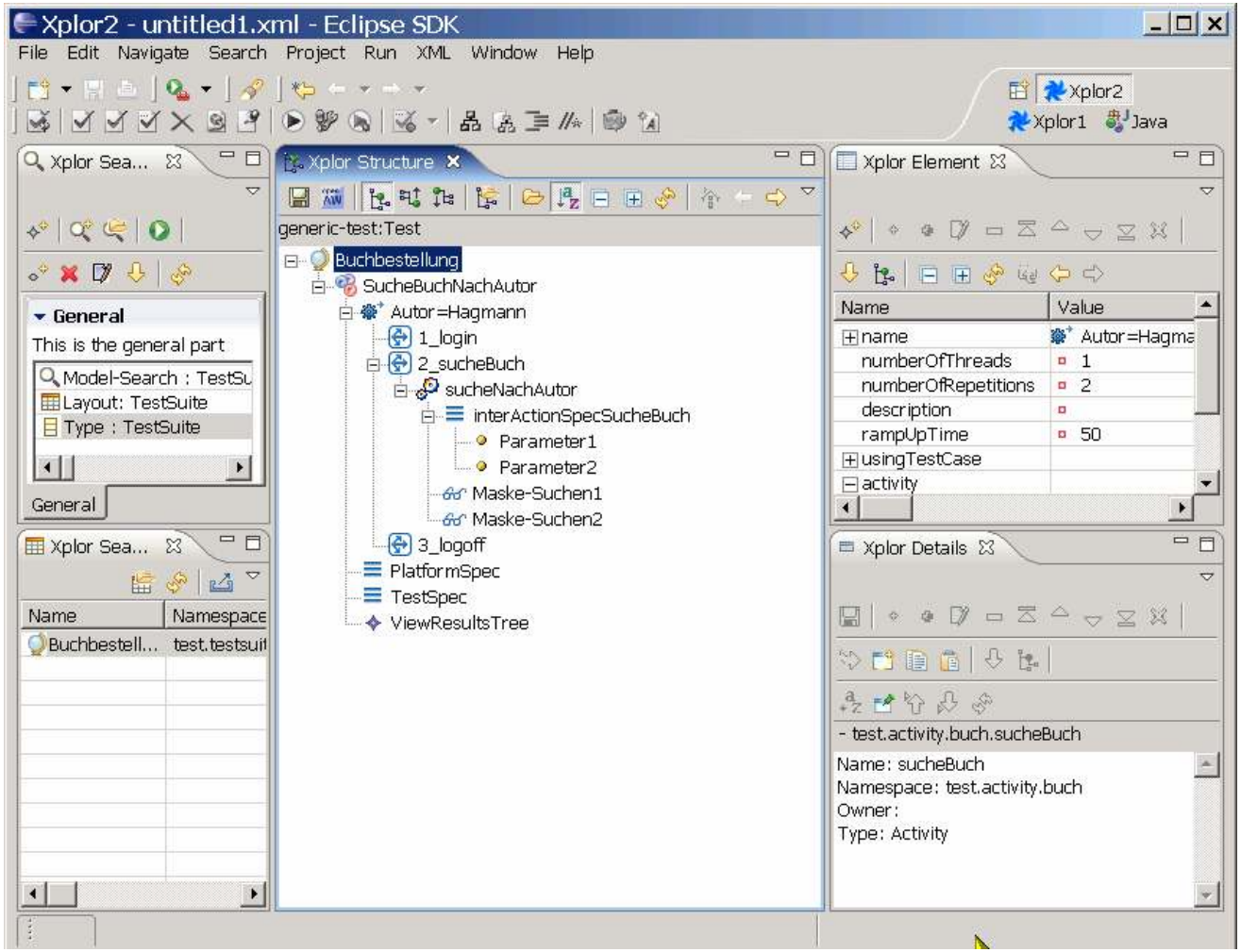
- Direkte Einbettung des Typsystems der Zielplattform in einen Editor
  - z. B. Java Editor
  - UML Werkzeug



- Effizientere Modellierung mit der DSL

- Besser Validierungsmöglichkeiten
- Steigerung der Akzeptanz der DSL

Xplor Demo



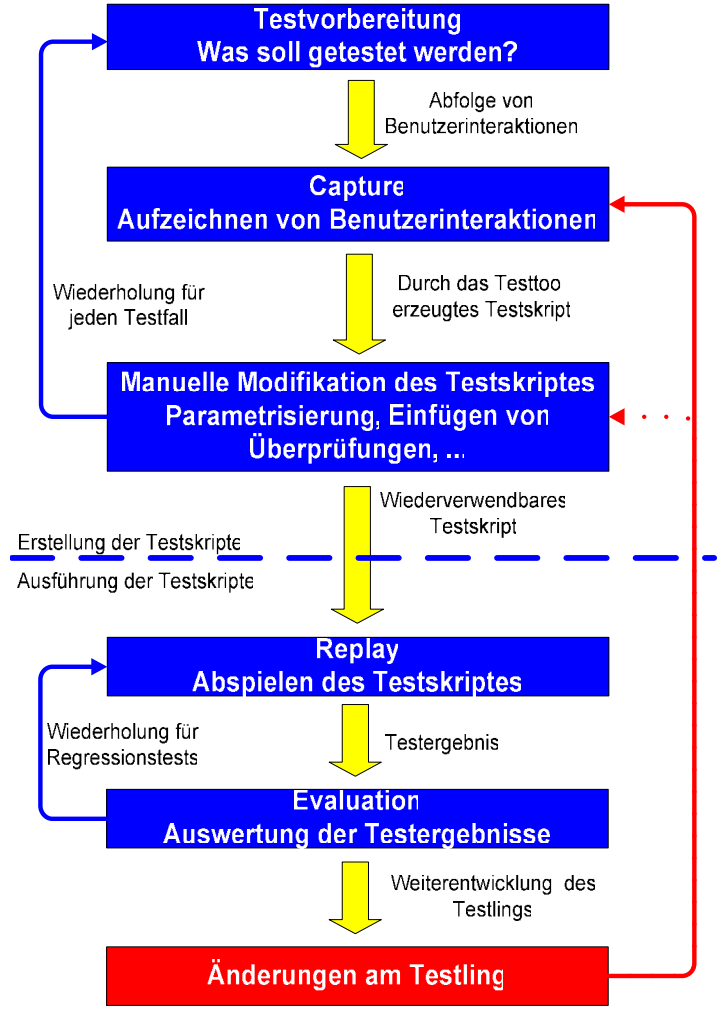
Die Wartungsfalle

MDTD im IDG-Projektalltag

IST-Zustand

- Das 1. Projekt: Postkorbanwendung
  - J2EE, Webfrontend
  - Ca. 25 Masken, beliebige Navigation
  - Jede Woche 1 Releasebuild mit Regressionstests
  - Max. 1 Tag Zeit für Regressionstest
  - Testscripte im CR-Verfahren erstellt (JMeter)

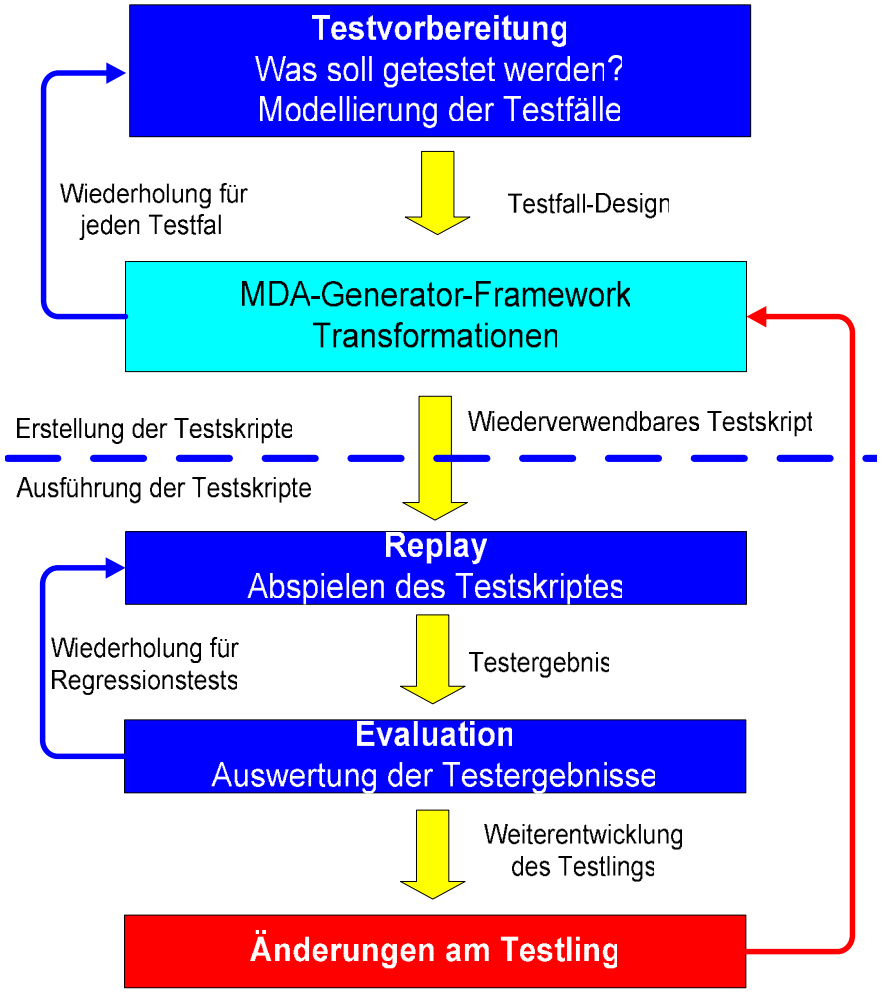
CR*-basiertes Vorgehen



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

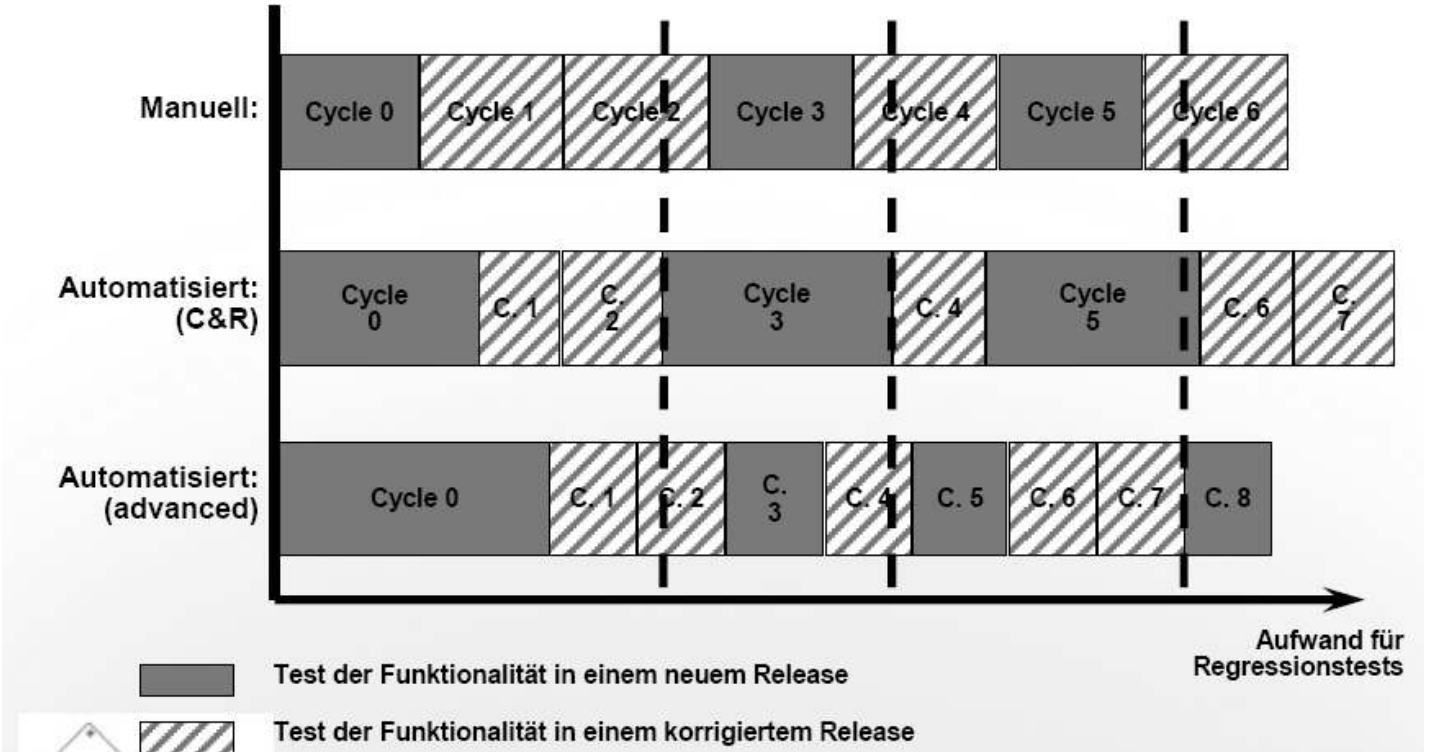
*Capture/ Replay

Modellbasiertes Vorgehen



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

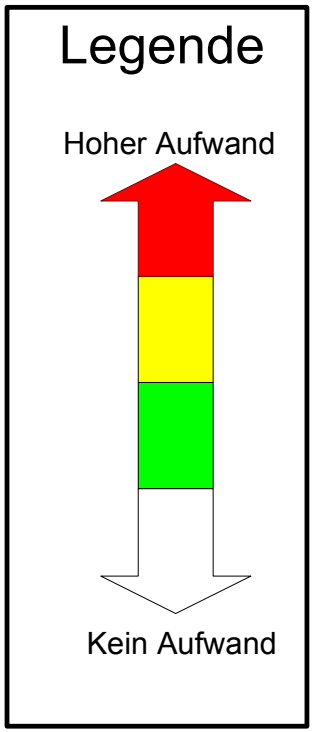
Die Wartungsfalle



Quelle: imbus Vortrag OOP 2006

Aufwende konventioneller und modellgetriebener Testentwicklung am Beispiel von JMeter

Änderungen an der Anwendung	Konventionelles CR Vorgehen	Modellbasierte Generierung
Layout	Keine Änderungen	Keine Änderungen
Umbennen eines Formfeldes	Manuelle Änderungen in allen relevanten Skripten	zentrale Änderung (Templates)
Hinzufügen eines Formfeldes	Erneutes Aufzeichnen aller relevanten Testskripte	zentrale Änderung (Templates)
Maskennavigation bei gleichbleibenden Events	Erneutes Aufzeichnen aller relevanten Testskripte	Änderungen im Testfalldesign
Maskennavigation durch neue Transitionen	Erneutes Aufzeichnen aller relevanten Testskripte	Änderung (Templates/ Testfalldesign)
Neue Maske und neue Transitionen	Erneutes Aufzeichnen aller Testkripte	Änderung (Templates/ Testfalldesign)
Technische Umstellung der Anwendung	Erneutes Aufzeichnen aller Testkripte	zentrale Änderung (Templates)



Erfahrungen und Ausblick

- Postkorbanwendung:
  - Umfangreiche technische Umstellung
    - 2 Tage Erweiterung der Scriptgenerierung
  - Maskenänderungen
    - 1-2 Stunden (Templateänderungen)
- Testabteilung übernimmt Verfahren
- Weitere Projekte stehen kurz vor Einführung des Verfahrens
- Quasi-Standard für Webprojekte innerhalb der IDG

Ausblick

- Wird MDTD die Zukunft der Testentwicklung ?
  - Produktionsstrassen im Automobilbau
- Open-Source Projekt (Start: 01.04.2006)
  - http://www.mdtd.de
- Dokumentation
- Weiterentwicklung des Domänen-Editors Xplor als RCP Anwendung
- Unterstützung weiterer Testarten und Zielplattformen
- Kopplung von MDSD und MDTD

Fragen und Diskussion