

# Bugkilla

funktionale Akzeptanztests  
für agile J2EE Entwicklung

**Orientation in Objects GmbH**  
Weinheimer Str. 68  
68309 Mannheim  
<http://www.oio.de>  
info@oio.de

**Dirk M. Sohn**  
<sohn@oio.de>

**Christian Dedek**  
<dedek@oio.de>

1

## Agenda

- **Motivation**
- Projektentscheidung
- Fachliche Idee
- Demo 1
- Grobarchitektur
- Integrationsideen
- Vollständiger Funktionaler Test
- Demo 2
- Fazit

2

## XP - the central principles

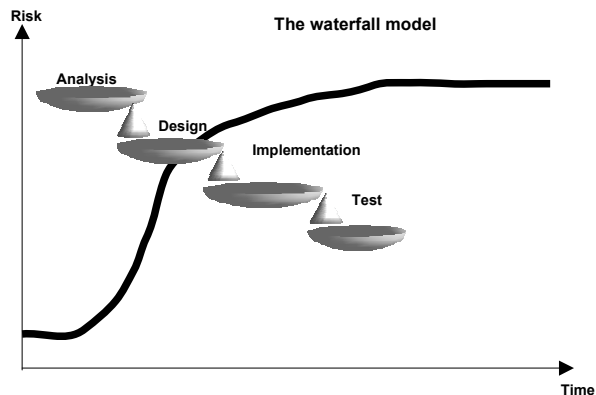
- Fast and detailed feedback
  - test driven design with acceptance tests **20%**
  - Customer On-site **30%**
  - pair programming **100%**
- Common understanding
  - planning game **100%**
  - simple design **100%**
  - System metaphor **80%**
  - collective code ownership **100%**
  - coding conventions **80%**
- Continuous process:
  - continuous integration **100%**
  - refactoring **100%**
  - frequent and small releases **100%**
- Developer Welfare
  - a week has 40 hours **95%**

3

## Waterfall model of system development

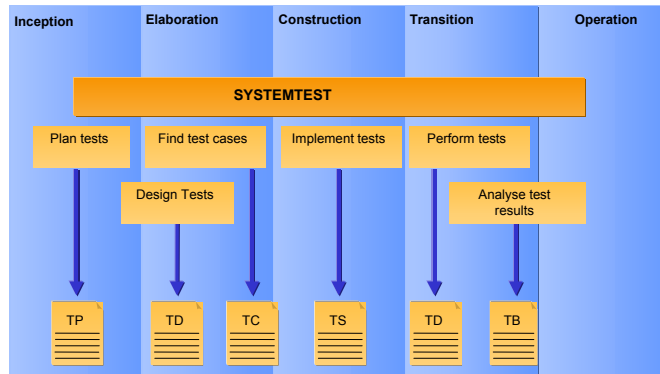
four phases  
each separated  
risk cumulated

(Tom Gilb:  
„if you do not  
actively attack  
the risks in your project,  
they will actively  
attack you.“)



4

# Classical System Test

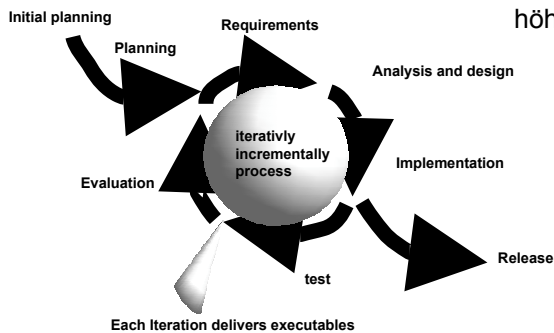


TP - Test plan  
 TD - Test design specification  
 TC - Test case specification  
 TS - Test scripts  
 TE - Test execution protocol  
 TR - Test report

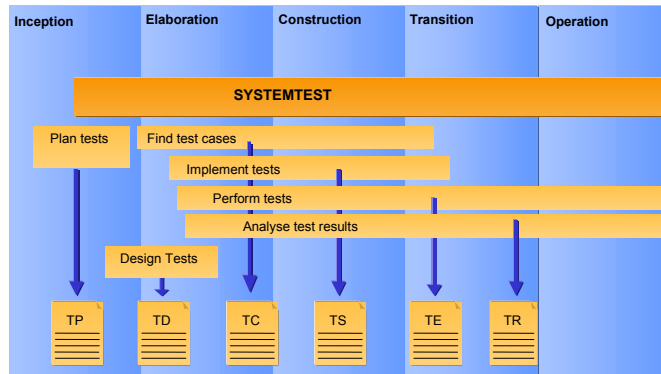
# Iterative Development

Softwareentwicklung ist  
 weder Massenproduktion  
 noch Ingenieurskunst

Risikominimierung durch  
 viele Iterationen führt zu  
 höheren Testaufwänden.

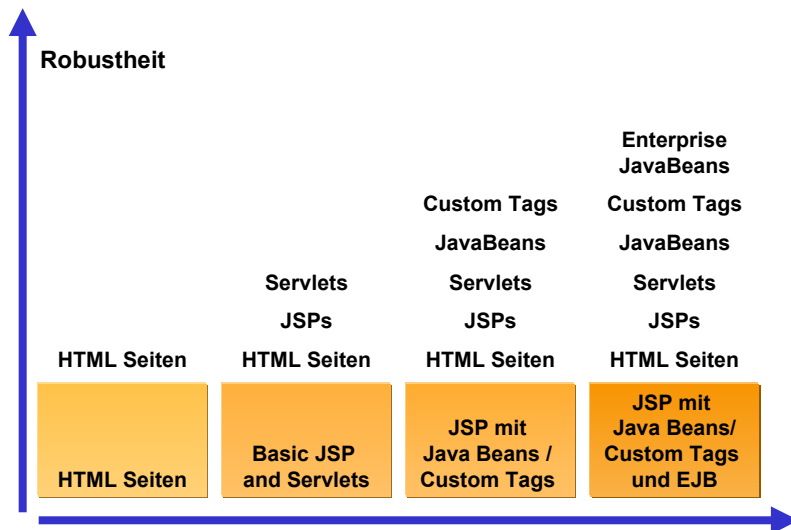


# System Test - Bugkilla style



TP - Test plan  
 TD - Test design specification  
 TC - Test case specification  
 TS - Test scripts  
 TE - Test execution protocol  
 TR - Test report

# Exkurs: Application Design Spektrum



Quelle: DEA1

- Im Server
  - Komponententests
    - **StrutsTestCase**
  - Integrationstests
    - **Cactus**
- Ausserhalb des Servers
  - Systemtest
    - **httpunit**
  - Akzeptanztests
    - **htmlunit**

- Motivation
- **Projektentscheidung**
- Fachliche Idee
- Demo 1
- Grobarchitektur
- Integrationsideen
- Vollständiger Funktionaler Test
- Demo 2
- Fazit

- Schlüssel zur Kostensenkung ist Automatisierung.
- Testfallspezifikation enthält Ein- und Ausgaben des Systems innerhalb des Testfalls
- Hierzu gehört für einen vollständigen funktionalen Test:
  - Direkte Eingaben des Anwenders durch Interaktion mit Browser
  - Direkte Ausgabe als HTTP-Response-Streams (Dokumente)
  - Anfangszustände der Geschäftsobjekte sind indirekte Eingaben
  - Endzustände der Geschäftsobjekte sind indirekte Ausgaben

- Aufgaben für Automatisierung:
  - Abbildung der direkten Ein- und Ausgaben des Users auf ein Aufzeichnungsformat (leicht bei Browser-Clients)
  - Übersetzung der Zustandsübergänge der fachlichen Geschäftsobjekte auf Zustandsübergänge von technischen Objekten.
- Lösung für die direkten Ein- und Ausgaben
  - Aufzeichnung der HTTP-Kommunikation. Eine Interaktion löst eventuell mehrere Requests aus, die jeweils von einer Response beantwortet werden.
  - Alternativen: Schaltflächenaufzeichnung, TCP-Pakete ohne Sitzung
- Lösung für die Zustandübergänge
  - ToDo

- Auswertung der direkten Ausgaben (Response-Dokumente )
  - Vergleich der Antworten der Testfallspezifikation mit denen der Testdurchführung - einfach nur HTML-Daten auswerten.
- Auswertung der Zustandsübergänge im Container
  - Übersetzung auf technische Objekte bei Einsatz von Frameworks deutlich präzisierbar.
  - Mechanismus zur Abfrage der Zustände fachlich relevanter Objekte
  - Kontrollflußsteuerung der Zustandsabfrage im Container nötig

- Motivation
- Projektentscheidung
- **Fachliche Idee**
- Demo 1
- Grobarchitektur
- Integrationsideen
- Vollständiger Funktionaler Test
- Demo 2
- Fazit

# Fachliche Idee von Bugkilla



Orientation in Objects

6.) Bewertung und Analyse

1.) Evolutionärer Prototyp

2.) Testfallspezifikation

5.) Ausführen der Testsprozeduren

3.) Testskripte aufzeichnen

4.) Testfälle aus Testskripten erstellen

```
<void property="requestBean">
  <object
    class="de.oio.bugkilla.HttpRequest">
    ...
    <void property="method">
      <string>GET</string>
    </void>
    <void property="name">
      <string>RB-Tue Aug 13 20:09:55 CEST
      2002</string>
    </void>
    <void property="protocol">
      <string>HTTP/1.1</string>
    </void>
  </object>
</void>
```



15

Bugkilla Project

© 2003 Orientation in Objects GmbH

# OIO Tooling philosophy



Orientation in Objects

1. Define clearly what is a tools job.
2. There is always a tool doing the job right. A project team has to find out, how to use it right.
3. If no tool doing the job right could be found by the team, the extension of a tool found during the evaluation is the solution.
4. Integration of tools doing their jobs right is the key to lasting success.

16

Bugkilla Project

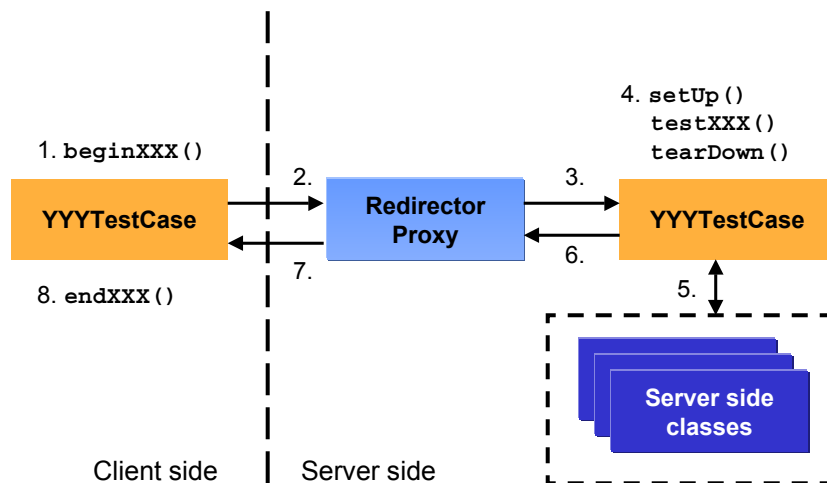
© 2003 Orientation in Objects GmbH

## Erweiterung eines der gefundenen Tools?

- Gefundene Kandidaten
  - Canoo (open source)
  - httputil (open source)
  - Exacum (kommerziell)

17

## Warum nicht Cactus ?



18

- Kommerzielle oder freie Lizenz, Open oder closed source?
  - Budget für Entwicklung und Marketing
  - Dienstleister und Produktfirma
  - Strategische Ausrichtung zum Support Center für OS-Server
  - also Open Source mit freier Lizenz

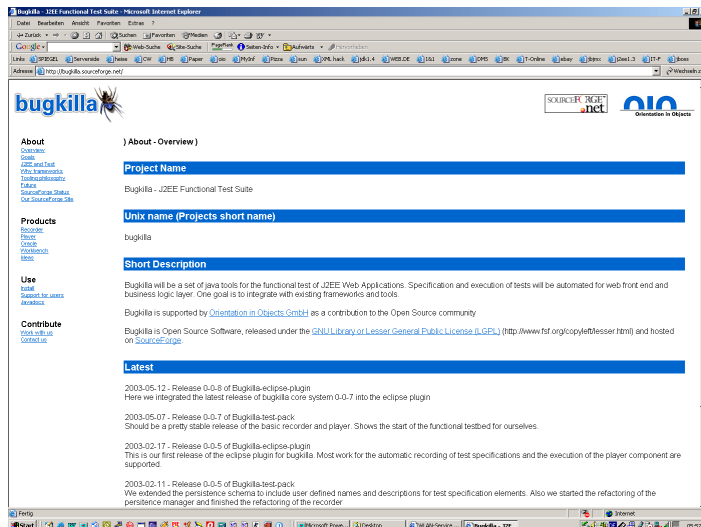
- Idee
  - *„Every good work of software starts by scratching a developer's personal itch.“* - Eric Raymond
- 1. Version
  - *„It's fairly clear that one cannot code from the ground up in bazaar style. One can test, debug and improve in bazaar style, but it would be very hard to originate a project in bazaar mode.“* - Eric Raymond

## Der Start der Basars

- Allgemeiner Funktionsumfang - muß für viele interessant sein
- Am besten stark modularisiert - bessere Arbeitsteilung
- Motivation
  - „There's one lesson that's really obvious: You cannot motivate the best people with money. Money is just the way to keep score. The best people in any field are motivated by passion. This becomes more true the higher the skill level gets.“ - Eric Raymond
- Infrastruktur
  - „The real lesson to be learned from open source communities are the techniques of networked collaboration that they've pioneered.“ - Tim O'Reilly

21

## Der Basar der Basare



**bugkilla - J2EE Functional Test Suite**

**About - Overview**

**Project Name**  
Bugkilla - J2EE Functional Test Suite

**Unix name (Projects short name)**  
bugkilla

**Short Description**  
Bugkilla will be a set of java tools for the functional test of J2EE Web-Applications. Specification and execution of tests will be automated for web front end and business logic layer. One goal is to integrate with existing frameworks and tools.  
Bugkilla is supported by [Orientation in Objects GmbH](#) as a contribution to the Open Source community.

**Latest**

2003-05-12 - Release 0.0.8 of Bugkilla-eclipse-plugin  
Here we integrated the latest release of bugkilla core system 0.0.7 into the eclipse plugin

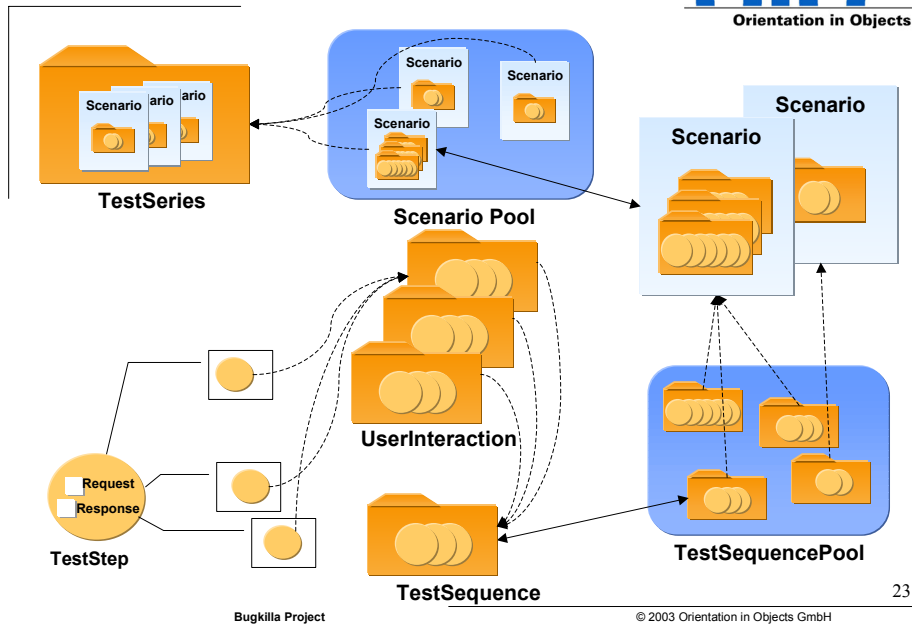
2003-05-07 - Release 0.0.7 of Bugkilla-test-pack  
Should be a pretty stable release of the basic recorder and player. Shows the start of the functional testbed for ourselves.

2003-02-17 - Release 0.0.5 of Bugkilla-eclipse-plugin  
This is our first release of the eclipse plugin for Bugkilla. Most work for the automatic recording of test specifications and the execution of the player component are supported

2003-02-11 - Release 0.0.5 of Bugkilla-test-pack  
We extended the persistence schema to include user defined names and descriptions for test specification elements. Also we started the refactoring of the persistence manager and finished the refactoring of the recorder

22

# Objektmodell



# Glossar

- **TestStep**
  - einzelner Testschritt, bestehend aus einem Request- und einem Response-Objekt
- **TestSequence**
  - ordnet TestSteps zu einem zusammengehörigen Ablauf
  - beinhaltet Informationen zum Environment (Aufzeichnungsdatum, Servername)
- **TestSequencePool**
  - beinhaltet sämtliche aufgezeichnete Sequenzen in Form von TestSequence-Files (zusammenhanglos)
- **Scenario**
  - ordnet mehrerer Sequenzen zu einem fertigen Testablauf, läuft in einer HTTP-Session ab, ist fachlich eindeutig zu beschreiben

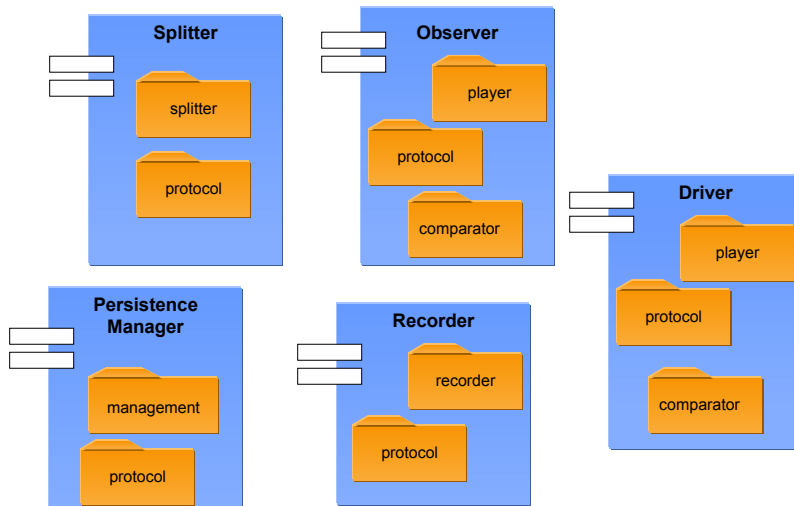
- SzenarioPool
  - beinhaltet sämtliche Szenarien in Form von Szenarien-Files (zusammenhanglos)
- TestSeries
  - Anordnung mehrerer Szenarien zum einer Testserie. Diese Szenarien haben untereinander keine Abhängigkeiten und werden der Reihe nach abgespielt.(eventuell bestehen Abhängigkeiten aber dann muss ein reset-des Servers erfolgen )
- Testspezifikation
  - Abstraktion von Teststeps, Testsequenzen, Szenarien, Testserien

- Motivation
- Projektentscheidung
- Fachliche Idee
- **Demo 1**
- Grobarchitektur
- Integrationsideen
- Vollständiger Funktionaler Test
- Demo 2
- Fazit

# Agenda

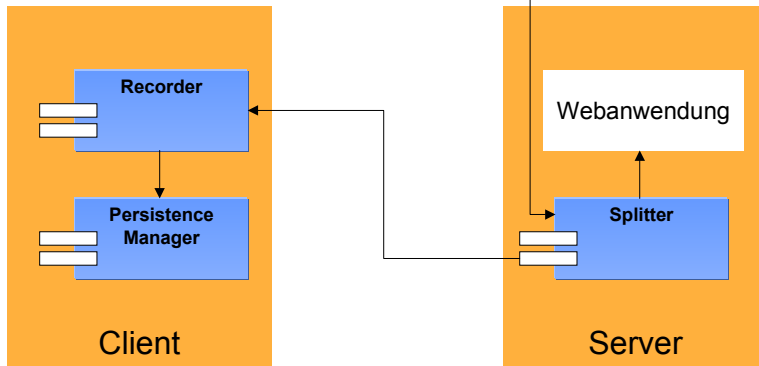
- Motivation
- Projektentscheidung
- Fachliche Idee
- Demo 1
- Grobarchitektur
- Integrationsideen
- Vollständiger Funktionaler Test
- Demo 2
- Fazit

# Komponenten

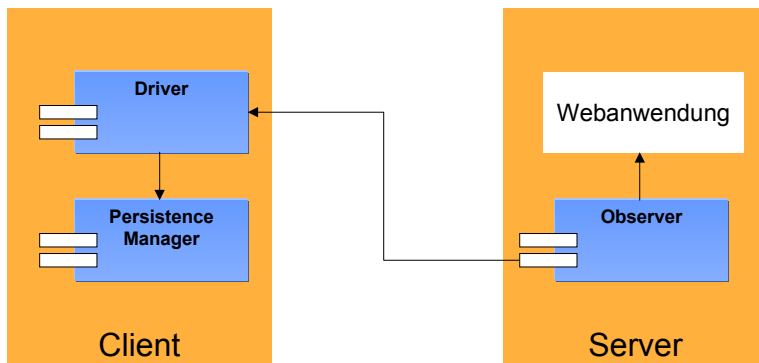


# Konfiguration: Recorder

Kategorie Personen Login	
NAME	Pauch
VORNAME	Markus
STRASSE	Vierdenk 34
ORT	60515 Frankfurt
TEL	(069) 18646
MAIL	pauch@recorder.com
abschicken	



# Konfiguration: Player

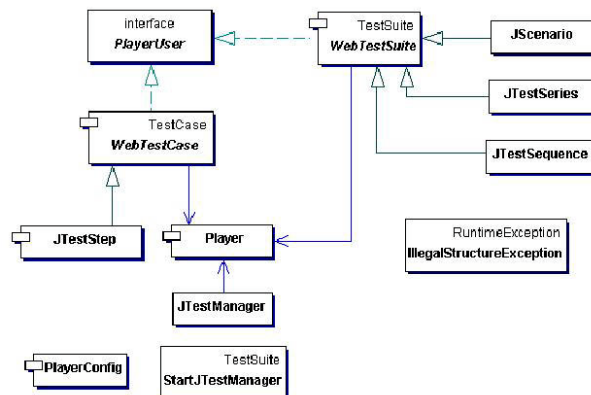


# Agenda

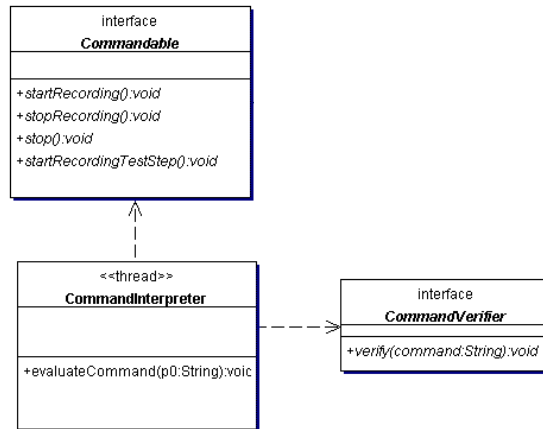
- Motivation
- Projektentscheidung
- Fachliche Idee
- Demo 1
- Grobarchitektur
- **Integrationsideen**
- Vollständiger Funktionaler Test
- Demo 2
- Fazit

# Testdriver

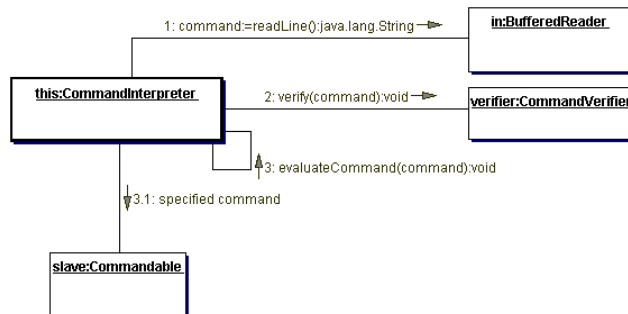
- JUnit Framework (HttpUnit - Schritt I)

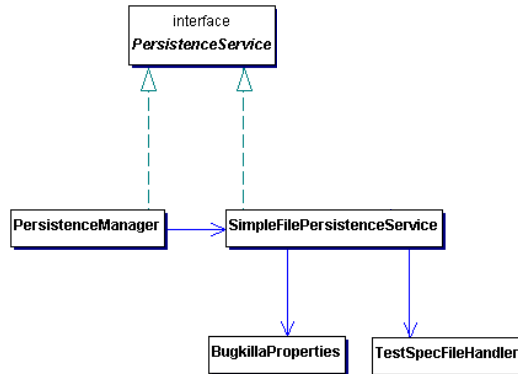


# Commandable



# Commadable - collaboration

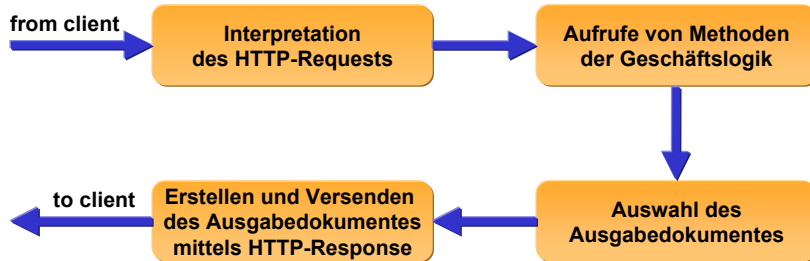




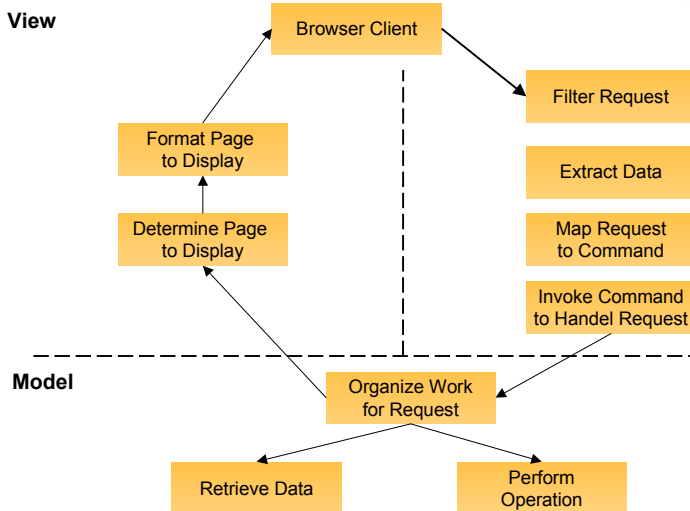
## Agenda

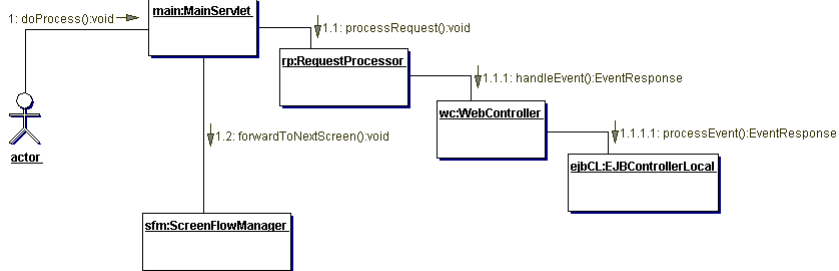
- Motivation
- Projektentscheidung
- Fachliche Idee
- Demo 1
- Grobarchitektur
- Integrationsideen
- **Vollständiger Funktionaler Test**
- Demo 2
- Fazit

# Web tier service cycle



# WAF - MVC





39

## AspectJ

- Instrumentierung - ein orthogonaler Aspekt
- Vollständige Trennung der Anwendung von der Instrumentierung (generativer Ansatz)
- Formulierung einer Strategie
- Definition von Eingriffspunkten
- Integration in die Anwendung („Einweben“)
- Veränderung des Entwicklungsprozesses
  - Konfigurationsmanagement
  - weiteres Paradigma notwendig

40

## Agenda

- Motivation
- Projektentscheidung
- Fachliche Idee
- Demo 1
- Grobarchitektur
- Integrationsideen
- Vollständiger Funktionaler Test
- **Demo 2**
- Fazit

## Agenda

- Motivation
- Projektentscheidung
- Fachliche Idee
- Demo 1
- Grobarchitektur
- Integrationsideen
- Vollständiger Funktionaler Test
- Demo 2
- **Fazit**

## Stand der Dinge

- Recorderkonfiguration
  - integriert und „lauffähig“ seit mehreren Releases
- Playerkonfiguration - Schritt I
  - HttpUnit vollständig integriert
- Playerkonfiguration - Schritt II
  - als Architekturprototyp im CVS Head
- IDE Integration
  - Eclipse Plugin
- Qualitätssicherung
  - automatisierter Komponententest
  - Selbsttest
  - automatisierte Code Coverage Messung
  - Cruise Control

43

## Stand der Dinge II

- Unhandliche Recorderkonfiguration
- manuelle Erstellung von Testorakeln
- keine Unterstützung von EJB-Instrumentierung
- Objektmodell noch nicht vollständig IEEE konform
  
- Wenig Tools für RAD von J2EE Webanwendungen
- keine build-in-Tests in J2EE

44

- Q 3
  - Release mit Strutsinstrumentierung
- Q 4
  - Weber für weitere Webframeworkinstrumentierung
  - Instrumentierung der EJB-Schicht

- Proxy-Recorder-Konfiguration
- Benutzung von Aspekten in Recorderkonfiguration
- Automatisierung der Spezifikation von Zustandsübergängen
- UML Testprofile

Vielen Dank für Ihre Aufmerksamkeit

**Dirk M. Sohn**  
<sohn@oio.de>

**Christian Dedek**  
<dedek@oio.de>

**Orientation in Objects GmbH**  
Weinheimer Str. 68  
68309 Mannheim  
<http://www.oio.de>  
info@oio.de

<http://bugkilla.sourceforge.net/>