Orientation in Objects

Test-driven Development aus dem Open Source Regal

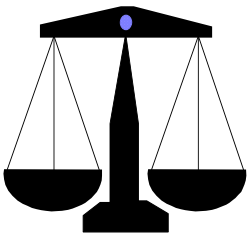
Christian Dedek (dedek@oio.de)
Sabine Winkler (winkler@oio.de)

Orientation in Objects GmbH
Weinheimer Str. 68
68309 Mannheim
www.oio.de

Qualitätskosten ein Minimierungsproblem

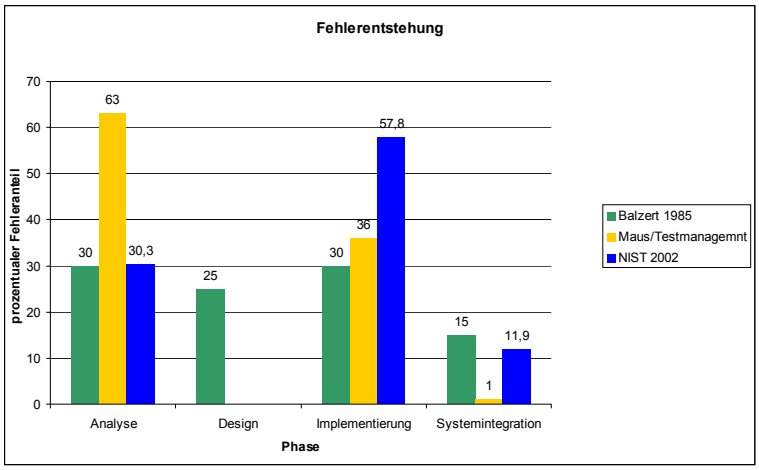
Orientation in Objects

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



Fehlerentstehung

Orientation in Objects



Der Hund und Testen ?

Orientation in Objects

- Der Hund kackt im Jahr etwa das 3,5-fache seines eigenen Körpergewichts.

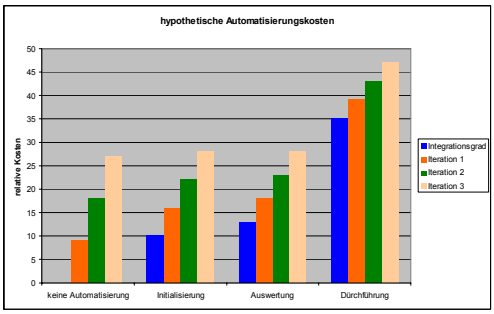
- Die Aussage ist frei erfunden.
- Möglicherweise liegt sie gar nicht so schlecht.
- Einen Nachweis habe ich nicht gegeben.
- Die meisten Bücher über das Testen sind voll von Äußerungen solcher Qualität.

Ökonomische Gründe zum Werkzeugeinsatz I

Orientation in Objects

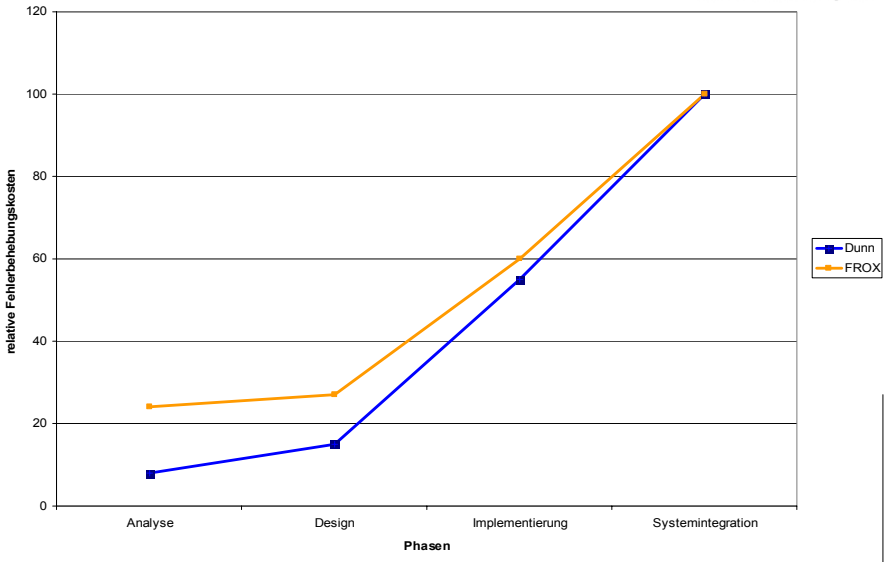
- *„Bereits im Anwendungsentwicklungsprojekt lassen sich durch TPA bis zu 10 - 15% der Gesamttestkosten einsparen“ [SQS]*
- *„Bei Vorgehensmodellen mit laufend wiederkehrenden Testausführungen amortisiert .. nach 4 bis 12 Retests“ [SQS]*

- Kostenreduktion durch Engineering - „Testware“
  - Wiederverwendung
  - Kapselung
  - Modularisierung
  - Integration



Ökonomische Gründe zum Werkzeugeinsatz II

Orientation in Objects



Die Vorgeschichte eines OS-Werkzeugs

- 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

OIO Tooling philosophy

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.

XP - the central principles

Orientation in Objects

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

Strategien im Qualitätsmanagement

Orientation in Objects

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

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

„Regal“-Kategorien

Orientation in Objects

- Funktionaler Test
  - Komponententest
  - integrierter Test
  - Systemtest
- Nicht-funktionaler Test

Lebenszyklus eines Unit Tests

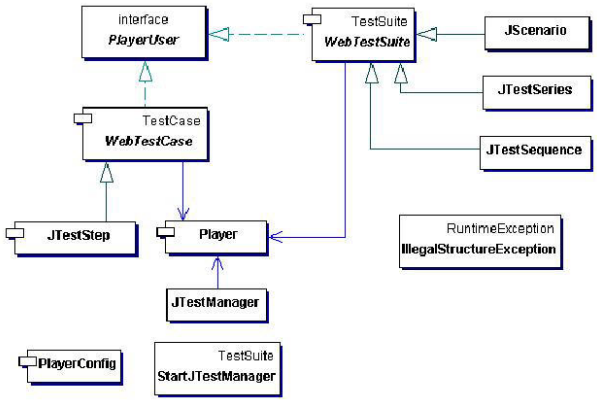
Orientation in Objects

- Erzeugung eines oder mehrer Objekte
- Objekte in definierten Anfangszustand bringen
- Ausführung einer Folge von Methoden, deren Wirkung getestet werden soll
- Endzustände, bzw. Wirkung auf Umwelt wird überprüft

Testdriver

Orientation in Objects

- JUnit Framework (HttpUnit - Schritt I)



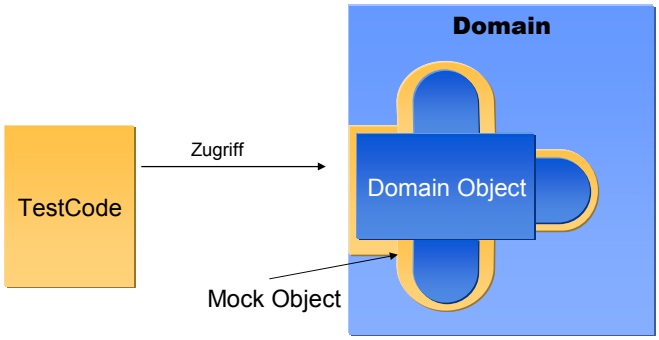
Junit Decorator

Orientation in Objects

- JUnitPerf
  - TimedTest Sicherstellung des Maximalen Zeit des Zeitverbrauchs eines Testfalls
  - LoadTest Simulation mit Threads

Mockobjects

Orientation in Objects

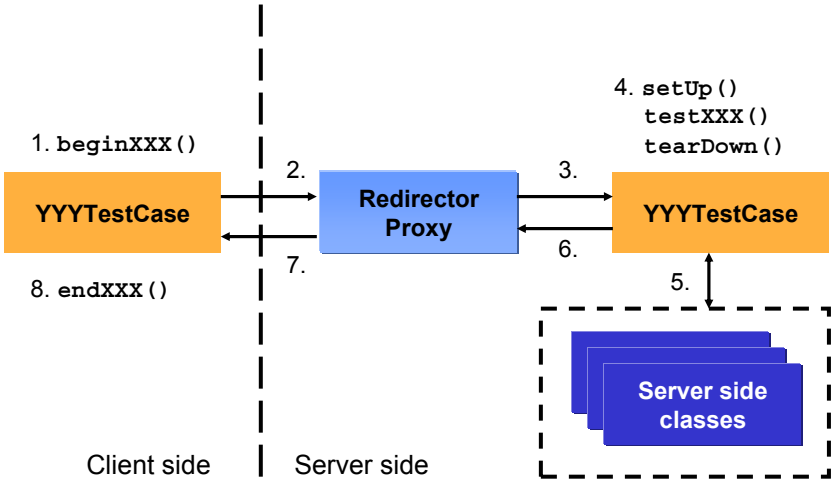


Mock Objects

Orientation in Objects

- früheres Testen von integrierten Objekten
- Abtrennung Komponente Container in Testumgebung möglich
- Anwendungsdesign verändert sich(BIT vs. lose Kopplung)
- Einsparungen durch Codegeneration von Mock-Klassen aus schnittstellen - Funktionalität der Mock-klassen sind Testkosten
- Kopplung zwischen Mocks vermeiden, funktionsreiche Mocks umstritten
- Toolkits: MockCreator, Mockmaker,MockObjects

Cactus

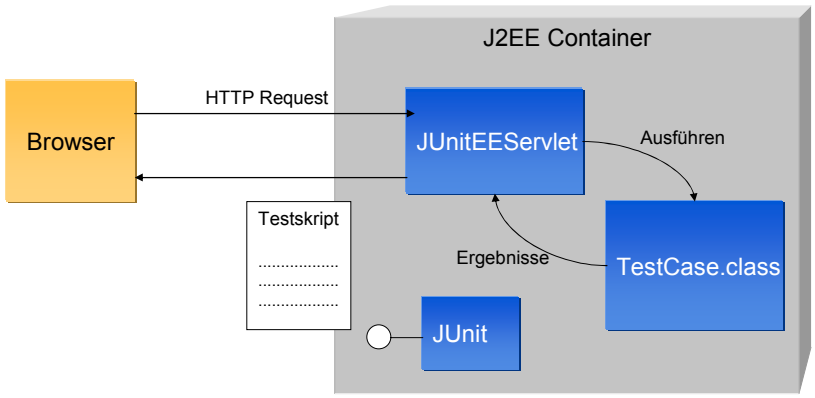


Cactus

- Test von Java Komponenten in der VM des Web Containers
  - z.B. Servlets, JSPs, Filter, EJBs (über local interfaces), Tags
- Integrations- und Systemtests
- Spezieller Ant Support

JUnitEE

Orientation in Objects

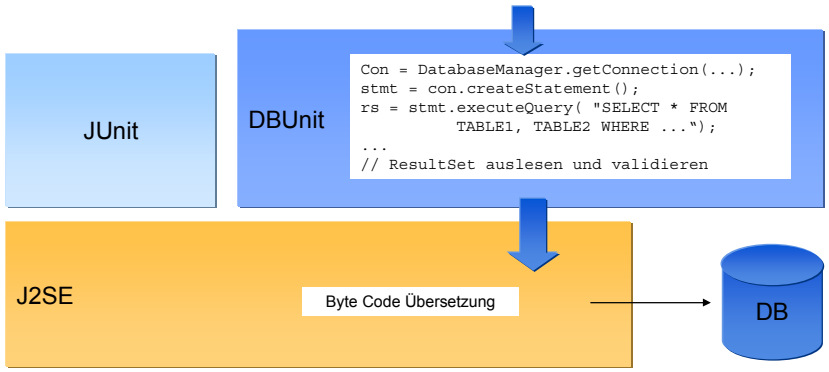


DBUnit

Orientation in Objects

DBUnit Testskript:

```
ITable actual = getConnection().createQueryTable("RESULT_NAME",
                    "SELECT * FROM TABLE1, TABLE2 WHERE ...");
Assertion.assertEquals(expected, new
CompositeTable(expected.getTableMetaData(), actual));
```



DBUnit

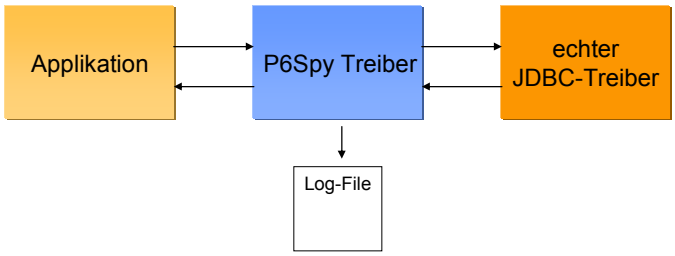
Orientation in Objects

- Ablage von Testdaten in XML
  - Ausgangsdaten des Systems
  - Erwartungswerte des Systems
- einfache Abfragen und Updates der relationalen Datenbestände
- Antintegration
- Problem:
  - Handling von Nullwerten
  - Portabilität der XMl-Datenbestände
  - Speicher und Zeitverbrauch bei großen Testdatenbeständen

P6Spy

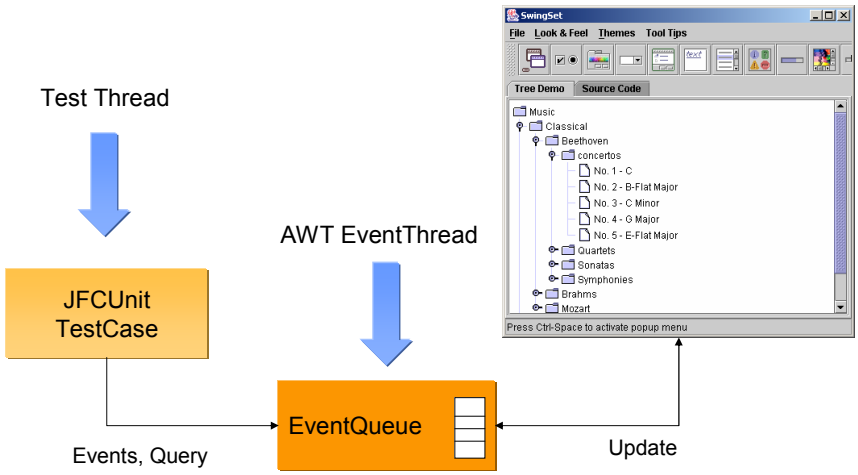
Orientation in Objects

- Betrachtung der Performance-kritischen Bestandteile
  - Logging sämtlicher JDBC Operationen mit P6Log Module
  - Zeitabhängiges Filtern von JDBC Statements mit P6Outage Module



JFCUnit

Orientation in Objects



JFCUnit

Orientation in Objects

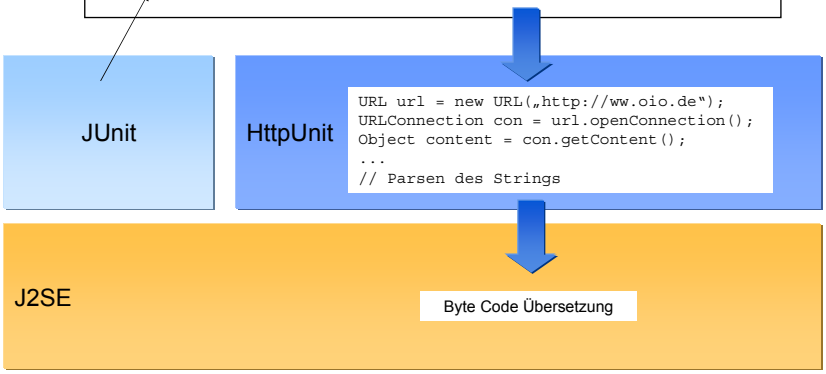
- API zum Suchen von JFC-Komponenten
- API zum Auslösen von Events
- besonderer Support für Single-Thread Architektur
  - asynchrone Eventbearbeitung nicht unterstützt
- Problem:
  - komplexe GUI-Komponenten(JTree,JTable)
  - modale Dialoge

HttpUnit

Orientation in Objects

HttpUnit Testskript:

```
WebConversation wc = new WebConversation();
WebResponse wr =
wc.getResponse( "http://www.oio.de" );
wr.getLinkWith(„Testen“).click();
assertEquals(resp.getTitle(), “OIO");
```



HttpUnit

Orientation in Objects

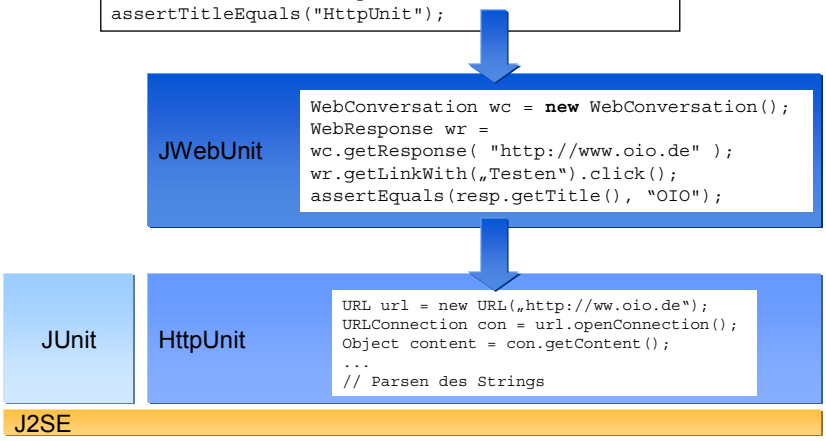
- JUnit Aufsatz zur Erstellung von Testskripten die
  - Steuerung des HTTP Protokolls
  - Behandlung von Sessions, Cookies, Frames
  - Bereitstellung von Textinhalten, XML DOM Dokumenten zur Bearbeitung via JUnit
- Systemtest von Webanwendung

JWebUnit

Orientation in Objects

JWebUnit Testskript:

```
getTestContext().setBaseUrl("http://www.oio.de");
beginAt("/");
clickLinkWithText("HttpUnit");
assertTitleEquals("HttpUnit");
```



JWebUnit

Orientation in Objects

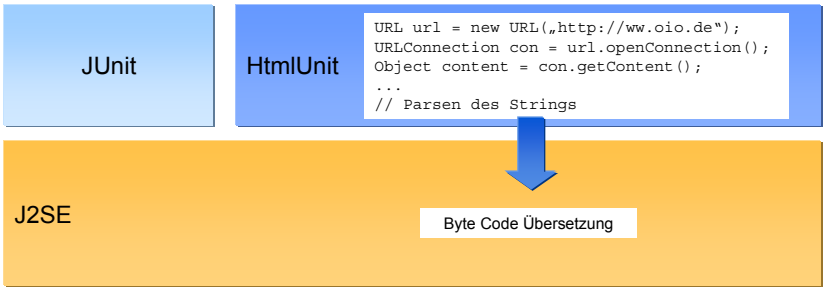
- Aufsatz auf HttpUnit, Rhino, Nekohtml
- Navigations API
  - Unterstützung für Formulare, Links, Frames und PopUp inklusive Navigation innerhalb dieser
- Assertions API
  - Validierung von Texten, Textbestandteile des HTML Dokuments, Zugriff auf Tabelleninhalte und HTML Elementen
  - Einbindung von Erwartungswerten über Properties ermöglicht
- Systemtest von Webanwendung

HTMLUnit

Orientation in Objects

HtmlUnit Testskript:

```
WebClient webClient = new WebClient();
URL url = new URL("http://www.oio.de");
HtmlPage pageOIO = (HtmlPage)webClient.getPage(url);
HtmlForm form = page1.getFormByName("search");
HtmlSubmitInput button =
      (HtmlSubmitInput)form.getInputByName("submitbutton");
HtmlTextInput textField =
      (HtmlTextInput)form.getInputByName("s      chString");
textField.setValueAttribute("xxxxx");
HtmlPage oioSearchPage = (HtmlPage)button.c      ();
```



HTMLUnit

Orientation in Objects

- Parallele Entwicklung zu HttpUnit
- Fokus: HTML Dokumente
  - Java Wrapper für eine Vielzahl von HTML Elementen
    - **z.B. HtmlBoby, HtmlButton, HtmlHeader1, ...**
- Testausführung über JUnit
- Systemtest von Webanwendung

TagUnit

Orientation in Objects

- „Taglibs“ als Komponenten testbar
  - Überprüfung zwischen Deklaration (TLD) und Umsetzung (Taglib-Klassen)
  - Testumgebung zur Ausführung im Container - TagUnit Tags
  - Beispiel:

```
<tagunit:assertEquals name="Simple test of generated content">
        <tagunit:expectedResult>...</tagunit:expectedResult>
        <tagunit:actualResult>...</tagunit:actualResult>
</tagunit:assertEquals>
```


JMeter

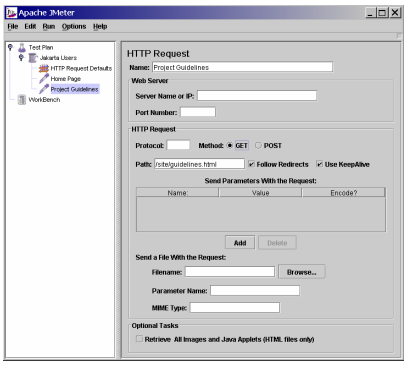
Orientation in Objects

- Support für Proxy, Authorization, Website Spider
- Testplanung Remote und zeitgesteuert durchzuführen
- Ant Treiber Support
- Verteilung der Testprozesse über RMI
- Auswertung: Antwortzeit und Durchsatz

JMeter

Orientation in Objects

- Testpläne basiert auf eigener Sprache
- Erstellung wird grafische unterstützt
- Capture & Replay für HTTP
- Swing und JDBC Tests
- Threadgroup
- Controller
  - Logic Controller
  - Sampler



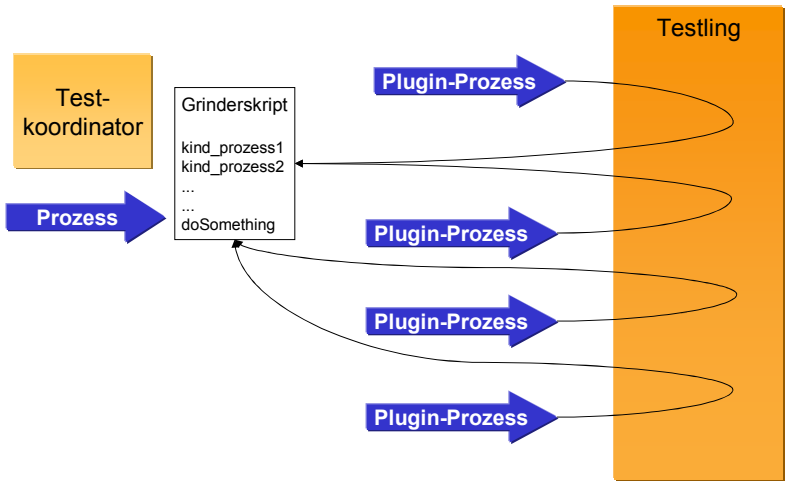
The Grinder

Orientation in Objects

- Grinder Testskripte steuern Aktivitäten in verschiedenen Prozessen über mehrere Rechner verteilt gezielt auf die Anwendung
  - Nicht-funktionale Systemtests
- *Grinder 2*: Plugin Mechanismus
  - Plugins für HTTP(S), RMI, Plain Java
- *Grinder 3*: basierend auf Jython
  - Ausführung über Java
- Problem: Auswertungsunterstützung
  - nur Transaktionszahl bereitgestellt

The Grinder

Orientation in Objects



TestMaker

Orientation in Objects

- Grinder3 ähnliches Konzept
- C&R Funktionalität z.B. für Http
- kein multi-client support
- Monitoring nur auf Response ausgerichtet
- Problem: Auswertung
- usw. Loadsim auf Muffin Proxy

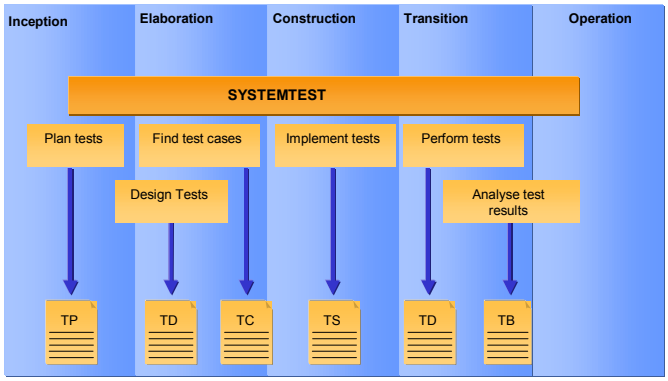
Weitere Kategorien

Orientation in Objects

- Integration
  - Ant, JUnit, Eclipse, ...
- Aktivität des Projekts
  - Mailing list, CVS, ...
  - Dokumentation
- Relevanz
  - Wiederverwendung in anderen Produkten ?
- Lizenz
- ...

Werkzeugunterstützung ?

Orientation in Objects



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

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

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