GEBIT Solutions

Technologie beherrschen ▶ Prozesse gestalten ▶ Lösungen schaffen ▶▶

Akzeptanztesten mit Integrity und FitNesse
Ein Vergleich

Dehla Sokenou

GEBIT Solutions

TAV35, Ingolstadt

Motivation

- Akzeptanztest als letzte Phase im Softwareentwicklungsprozess
  - Idealerweise durch den Kunden durchgeführt
  - Tests idealerweise für die (Weiter-) Entwicklung zur Verfügung gestellt
- Automatisierte Testausführung wichtig, aber wie?
- Werkzeugunterstützung durch Testwerkzeuge, aber welches?
  - Scriptbasiert (FitNesse, Integrity, JBehave, Cucumber, …)
  - Capture & Replay (Abbot, Selenium, QF-Test, TestComplete, …)

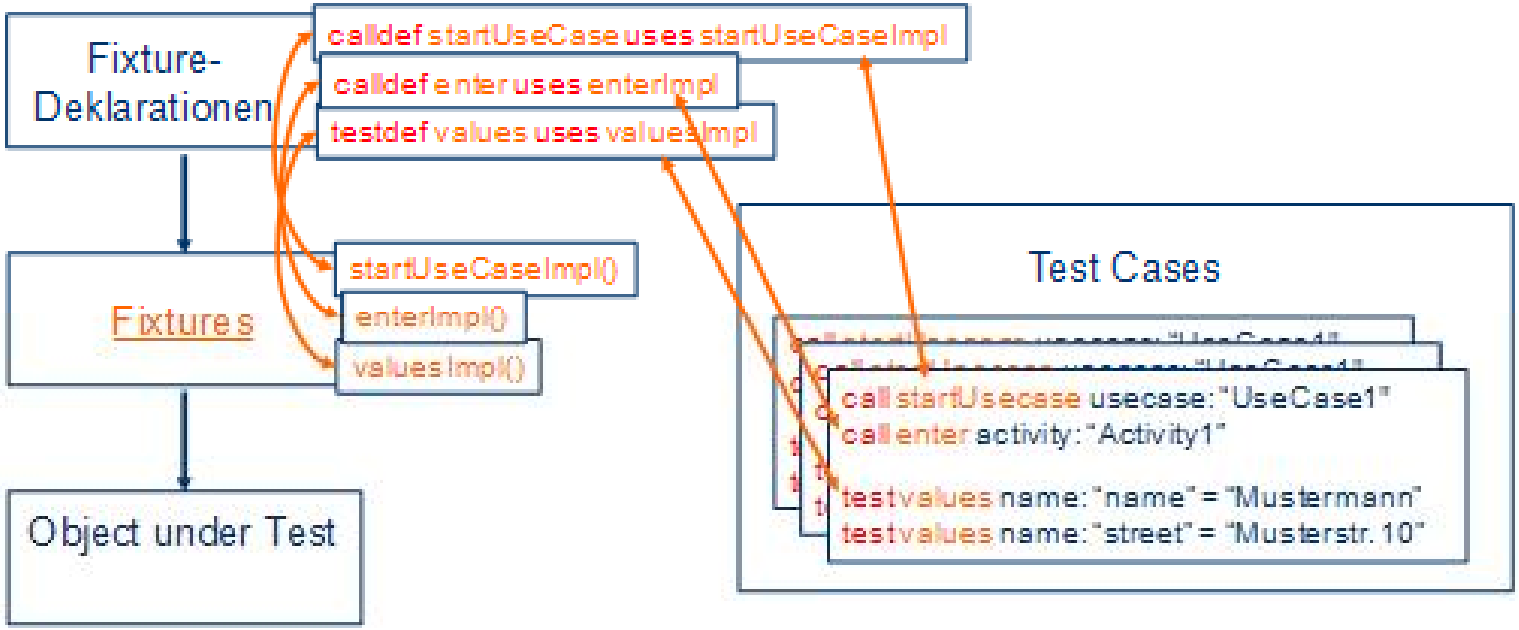
Integrity (1)

- Acceptance-Test-Framework
- Intern bei GEBIT entwickelt
- Seit 2 Jahren im produktiven Einsatz
  - Anbindung an eigenes modellbasiertes Entwicklungsframework
- Inzwischen Open Source unter EPL
  - Download unter http://www.integrity-tf.org
- Fokus auf
  - IDE-Support: Code Completion, Formatter, References…
  - Lesbarkeit von Testfällen und Ergebnissen
  - Flexibilität

Integrity (2)

- Eclipse-Plugin
- XText-basierte DSL
- Definition von
  1. Fixture-Klassen (Java, ggf. mehr als tatsächlich verfügbar)
  2. Fixture-Deklarationen (Integrity-Dateien, tatsächlich nutzbare Fixtures)
  3. Testsuiten (Integrity-Dateien)
- Unterstützt
  - Tests und Calls
  - Testtabellen
  - Variablen
  - Links, Referenzen, Abhängigkeiten
- Testausführungsumgebung inklusive Debugger
  - Eigene Testauswertung möglich

Integrity (3)



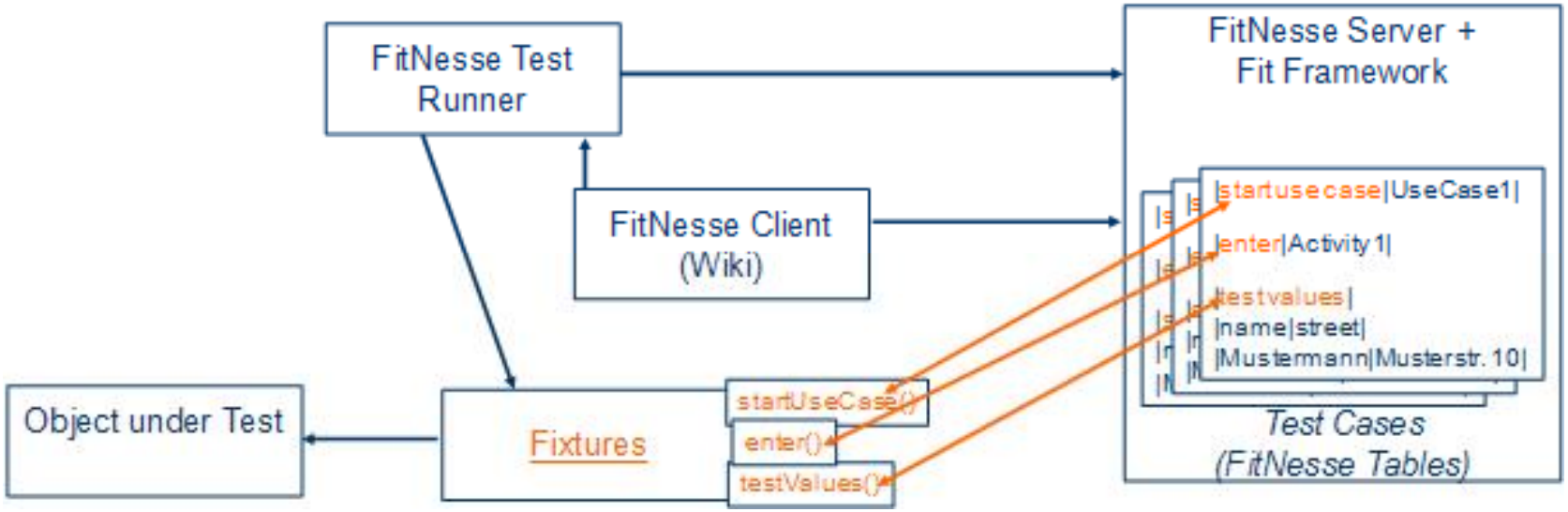
FitNesse (1)

- Open-Source-Framework
  - Download unter http://www.fitnesse.org
- Ursprünglich basierend auf dem FIT Test-Framework
  - Inzwischen auch mit anderer Testausführung auf Basis Slim verfügbar
- Weit verbreitet
  - Viele Erweiterungen verfügbar

FitNesse (2)

- Webserver / Wiki
- DSL in Wiki-Syntax
- Definition von
  1. Fixture-Klassen (Java oder andere unterstützte Programmiersprache)
  2. Testsuiten (Wiki oder anderes unterstütztes Format, bspw. Excel)
  - ► Testsuiten automatisch anhand Namen auf Fixture-Klassen gemappt
- Unterstützt
  - Testtabellen
  - Variablen (global, lokal)
  - Links, Referenzen
  - Inline-Testdokumentation
- Testausführungsumgebung
  - Eigene Testauswertung möglich

FitNesse (3)



Erfahrungen im Projektalltag

- Beide Testframeworks im produktiven Einsatz bei GEBIT
  - FitNesse seit 2005
  - Integrity seit 2011
- Zum Teil werden beide Testframeworks in den selben Projekten eingesetzt
  - Unterschiedliche Anwendungsbereiche
  - Historische Gründe: Integrity erst reif für den Einsatz, als Projekt schon lief
  - Wahlweiser Einsatz durch Bindung an gleiches modellbasiertes Entwicklungsframework TREND möglich
- Viele GEBIT-spezifische Erweiterungen für FitNesse, die z.T. auch in Integrity verfügbar sind

▶ Gute Vergleichbarkeit

Dokumentation

Integrity

- Code gut dokumentiert
- Handbuch verfügbar, enthält aber noch nicht alle Funktionen
  - Weitentwicklung des Frameworks schneller als Dokumentation

FitNesse

- Code schlecht bis gar nicht dokumentiert
- Handbuch sehr ausführlich, aber nicht sehr übersichtlich
- Handbuch direkt im Wiki verfügbar
  - Auch eigene Dokumentation kann hier eingebunden werden

Erweiterbarkeit

Integrity

- Testbefehle
  - Implementierte Fixtures und deklarative Fixture-Bindings
- Testumgebung
  - Eigene Testauswertung

FitNesse

- Testbefehle
  - Implementierte Fixtures (FIT = interne Datenstruktur)
- Testumgebung
  - Eigene Testauswertung
  - Eigene Wiki-Widgets, z.B. für relative Datumseingaben
  - Eigene Responder, z.B. zur einfachen Eingabe der Testbefehle

▶ Erweiterbarkeit für Testbefehle notwendig, da nur einfache Testaufgaben out-of-the-Box verfügbar

Testdefinition (1)

Integrity

- Eigener XText-basierter Editor unter Eclipse
  - Syntax-Highlighting, Auto-Complete
  - Referenzen finden
  - Durch Eclipse Anbindung an Versionskontrollsystem
  - Annotations für bessere Lesbarkeit der Testergebnisse

FitNesse

- Tests bevorzugt definiert im Wiki
  - Keine zusätzliche Client-Software nötig
  - Erweiterbarer Edit-Responder unterstützt Testeingabe durch Templates
  - Andere Wege möglich
  - Kein Syntax-Highlighting, kein Auto-Complete
- FitNesse-Server kann lokal oder remote genutzt werden
  - Mischbetrieb durch Referenzen möglich

Testdefinition (2)

Integrity

- Unterscheidung zwischen
  - Aufrufen (Calls)
  - Tests
  - Testsuiten
- Testsuite-Hierarchien
- Testfall-Templates
- Referenzen auf andere Testfälle / Testsuiten

FitNesse

- Unterscheidung zwischen
  - Testfällen
  - Testsuiten
  - Mischformen
- Testsuite-Hierarchien
- Testfall-Templates
- Referenzen auf andere Testfälle / Testsuiten
  1. Durch Referenzierung direkt im Testfall
  2. Durch Einbindung als eigener Testfall in einer Suite (auch mehrfach)

test use case	Maintain Flight List
run	

Initially neither the browse, nor reset buttons should be enabled

check	is enabled	browse.airline	false
check	is enabled	reset.airline	false

Go to new record mode

press	new

Now we should be able to browse for airlines

check	is enabled	browse.airline	true
check	is enabled	reset.airline	false

Check for correctly initialized reference data editors

test list values	departureLocation
Hamburg-Hamburg-Airport	
Düsseldorf-Rheinflughafen	
Berlin-Tegel	
Berlin-Tempelhof	

define variable	listsize1	browser list size

Create the 1st flight object

set values					
flightNumber	airline.code	departureTime	price	departureLocation	destinationLocation
LX123	LH	12:00:00	1234.00	Berlin-Tegel	Hamburg-Hamburg-Airport

Associate with the airline

press	browse.airline

Now we should be able to reset

check	is enabled	browse.airline	false
check	is enabled	reset.airline	true

Save the data

press	save

FitNesse - Testeditor

Test2CreateFlights

Help text:

Tags:

- Insert Fixture Table - | TREND FitNesse Binding Documentation

Spreadsheet to FitNesse | FitNesse to Spreadsheet | Format | Insert Template | wrap

```
!|test use case|Maintain Flight List|
|run|

''' Initially neither the browse, nor reset buttons should be enabled '''
|check|is enabled|browse.airline|false|
|check|is enabled|reset.airline|false|

''' Go to new record mode'''
|press|new|

''' Now we should be able to browse for airlines '''
|check|is enabled|browse.airline|true|
|check|is enabled|reset.airline|false|

''' Check for correctly initialized reference data editors '''
|test list values|departureLocation|
|Hamburg-Hamburg-Airport|
|Düsseldorf-Rheinflughafen|
|Berlin-Tegel|
|Berlin-Tempelhof|

|define variable|listsize1|browser list size|

''' Create the 1st flight object'''
```

Save | Cancel

Integrity – Testeditor

```
import trend.gui.*
import trend.workflow.*
import trend.math.*


packagedef de.gebit.traveldemo_bookflight with

    suitedef rootSuite with

        variable listsize1
        variable listsize2

        call launch
        call runUsecase usecase: "Maintain Flight List" startFromMenu: true

        call transition name: "Refresh"

        call getBrowserListSize -> listsize1

        call transition name: "New"

        call set name: "flightNumber" value: "AB123"
        call selectItem name: "airline" value: "Air Berlin"
        call set name: "departureTime" value: "12:00:00 "
        call set name: "price" value: "1234.00"
        call selectItem name: "departureLocation" value: "Berlin-Tegel Airport"
        call selectItem name: "destinationLocation" value: "Hamburg-Hamburg Airport"

        call transition name: "Save"

        call getBrowserListSize -> listsize2

        test inc value: listsize1 = listsize2

        tabletest get
            | name          | =         |
            | "flightNumber" | "LX123"   |
            | "price"        | "1234.00" |

    suiteend

packageend
```

Testausführung

Integrity

- Testausführung in Eclipse mit oder ohne Testcontrol
- Testergebnisse als Beschreibungen der Tests inklusive Testdaten
  - Lesbarkeit abhängig von den annotierten Fixtures
- Direkter Sprung in den Testfall möglich
- Testabbruch jederzeit möglich

FitNesse

- Testausführung aus dem Browser oder per Testrunner
- Testergebnisse entsprechen 1:1 Testfällen
  - Inklusive aller Kommentare
  - Testergebnis wird farbig markiert, ggf. angereichert um Kommentare und Ausnahmen
  - Zusammenfassung über alle Tests
  - Als Testreport verwendbar
- Testabbruch jederzeit möglich

FitNesse - Testergebnisse

test use case	Maintain Flight List
run	

Initially neither the browse, nor reset buttons should be enabled

check	is enabled	browse.airline	false
check	is enabled	reset.airline	false

Go to new record mode

press	new

Now we should be able to browse for airlines

check	is enabled	browse.airline	true
check	is enabled	reset.airline	false

Check for correctly initialized reference data editors

test list values	departureLocation
Hamburg-Hamburg-Airport	
Düsseldorf-Rheinflughafen	
Berlin-Tegel	
Berlin-Tempelhof	

define variable	listsize1	browser list size

Create the 1st flight object

set values					
flightNumber	airline.code	departureTime	price	departureLocation	destinationLocation
LX123	LH	12:00:00	1234.00	Berlin-Tegel	Hamburg-Hamburg-Airport

Associate with the airline

press	browse.airline

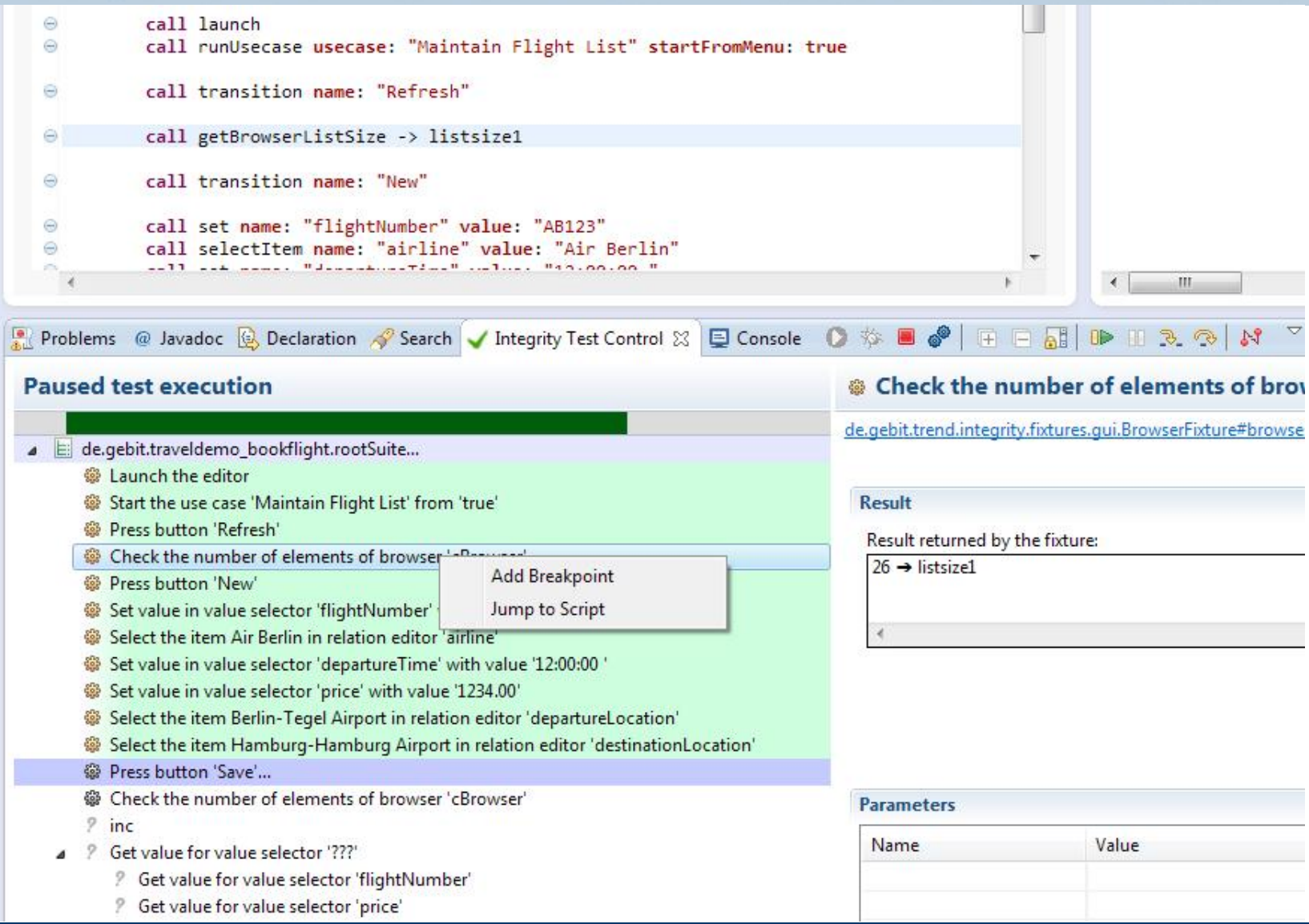
Now we should be able to reset

check	is enabled	browse.airline	false
check	is enabled	reset.airline	true

Save the data

press	save

Integrity - Testcontrol



Debugging

Integrity

- Testcontrol enthält Debugger
  - Breakpoints im Testfall setzen
  - Breakpoints im Code setzen
- Da in Eclipse integriert, einheitlicher Zugriff auf Code und Tests, einheitliche Behandlung der Breakpoints

FitNesse

- Entweder per Testrunner
  - Breakpoints im Code setzen
- Oder Eigenentwicklung
  - Breakpoints per Testbefehl im Testfall setzen
- Remote-Debugging wird unterstützt

Integrationstests zwischen verschiedenen Systemen

Integrity

- Von vornherein so konzipiert, dass mehrere Anwendungen parallet getestet werden können

FitNesse

- Prinzipiell nur Test einer Anwendung zu einer Zeit
- Durch Erweiterungen möglich

Plattform- und Sprachunabhängigkeit

Integrity

- Plattformunabhängigkeit durch Java-Implementierung gegeben
- Bisher nur Java
- Erste .net-Experimente
- Anbindung an Selenium für Webapplikationstests

FitNesse

- Plattformunabhängigkeit durch Java-Implementierung gegeben
- Weitere Sprachen als Java durch andere Testrunner möglich
  - Unterstützung für Ruby, PHP, Python, Smalltalk, … vorhanden
- Anbindung an Selenium für Webapplikationstests

Lizenz

Integrity

- Eclipse Public License (EPL)
- Weitergabe von Modifikationen unter EPL
- Erweiterungen können andere Lizenz besitzen
  - Fixtures müssen nicht ebenfalls EPL sein

FitNesse

- GNU General Public License (GPL)
- Copy-Left
  - Weitergabe von Erweiterungen / Ergänzungen nur, wenn diese wieder unter GPL veröffentlicht
  - Gilt somit auch für Fixtures
- Sehr genaue Analyse und Sorgfalt bei Implementierung notwendig, wenn Weitergabe geplant

Akzeptanz von Entwicklerseite

Integrity

- Durch Eclipse-Integration Vorteile wie
  - Syntax-Highlighting
  - Auto-Complete
  - Suchfunktionen
  - Sprung an fehlerhafte Stelle
  - Gewohnte Umgebung
  - …

► Gute Akzeptanz bei Entwicklern

FitNesse

- Elementare Funktionen wie Syntax-Highlighting fehlen
- Schlechte Dokumentation der Sourcen verhindert Implementierung neuer Fixtures

► Schlechte Akzeptanz bei Entwicklern

Akzeptanz von Kundenseite

Integrity

- Installation zusätzlicher Software beim Kunden notwendig
  - Eclipse + Integrity-Plugin
- Testsprache muss gelernt werden
  - Mehrsprachigkeit und kundenspezifische Wünsche durch Fixture-Binding umsetzbar

▶ Bisher wenig Erfahrung mit Tests von Kundenseite

FitNesse

- Installation zusätzlicher Software beim Kunden nicht notwendig
  - Nur Webbrowser notwendig
  - FitNesse-Server kann auch vom Entwicklungshaus zur Verfügung gestellt werden
- Testsprache muss gelernt werden

▶ Mittlere Akzeptanz von Kundenseite

Zusammenfassung

- Integrity gute Alternative zu FitNesse
- Beide Frameworks als Open-Source verfügbar
- Einsatz abhängig vom Kontext
  - FitNesse bei
    - Sprachunabhängigkeit
    - Zentraler Erfassung der Testfälle oder
    - Einfacher Verteilung der Testumgebung an die Tester
  - Integrity bei
    - Java-Anwendungen
    - Integration in vorhandene Entwicklungsumgebung
    - Integrationstest zwischen verschiedenen Systemen