Specification of CBB Testing for Fractal

5th Fractal Workshop at ECOOP’2006

D. Deveaux - P. Collet

UBS/Lab. Valoria/ALCC team - UNSA/Lab. I3S/OCL team

daniel.deveaux@univ-ubs.fr philippe.collet@unice.fr

Ce travail est réalisé dans le cadre du projet SafeCode soutenu par l’ANR (ARA SSIA 2006-2009)
Goal

Background

Specification of the Framework
◆ Principles
◆ Implementation in Fractal

Future Work
For several years, Contract Based Testing (CBT) is considered as one of the best testing techniques for OO-software [Binder96].

For components, contracts are at the center of many studies and several CBT proposals have be made recently:
- methodological approaches [Gross05],
- practical frameworks [Valentini.ea05]

Such a framework has not been proposed for the Fractal component model.
Background

Design by contract

STclass

UBS/Valoria - IRISA/Triskell

Specification and Test-First approach

Contract Based Built-in Testing for Java

Test result management

STclass-4.0 for Java

www.stclass.org

B. Meyer - UBS/Valoria - IRISA/Triskell - 135/OCL

Specification of CBB Testing for Fractal

5th Fractal Workshop at ECOOP'2006 - p. 4/14
Background

Design by contract

STclass

UBS/Valoria - IRISA/Triskell

Specification and Test-First approach

Contract Based Built-in Testing for Java

Test result management

STclass-4.0 for Java
www.stclass.org

Fractal

FT-R&D - INRIA
Hierarchical Component Model
fractal.objectweb.org

2000

2002

2004

2006
Background

- Agenda
  - Goal: a Testing Framework for Fractal
  - Background
    - STclass contributions
    - ConFract: Mastering Complexity
    - Types of Contracts
    - CBBT Framework Principles
    - Built-in Test: a Dynamic TestBed
    - Contract Based Testing
    - TestUnit, TestCase, TestSuite
    - Populate the TestBed
    - Actual and Future Works

---

### Design by contract

<table>
<thead>
<tr>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>STclass</td>
</tr>
<tr>
<td>2002</td>
<td>Specification and Test-First approach</td>
</tr>
<tr>
<td>2004</td>
<td>Test result management</td>
</tr>
<tr>
<td>2006</td>
<td>ConFract</td>
</tr>
</tbody>
</table>

**STclass**

- UBS/Valoria - IRISA/Triskel
- STclass-4.0 for Java
- www.stclass.org

**ConFract**

- I3S/OCL - FT-R&D
- Contract framework for Fractal
- Kind of contracts
- Responsibilities

---

**Fractal**

- FT-R&D - INRIA
- Hierarchical Component Model
- fractal.objectweb.org

---

Specification of CBB Testing for Fractal

5th Fractal Workshop at ECOOP’2006 - p. 4/14
Design by contract

- **STclass**: UBS/Valoria - IRISA/Triskell
  - **Specification and Test-First approach**
  - **Contract Based Built-in Testing for Java**
  - **Test result management**
  - **STclass-4.0 for Java**
    - www.stclass.org

- **ConFract**: I3S/OCL - FT-R&D
  - **Contract framework for Fractal**
  - **Kind of contracts**
  - **Responsibilities**

- **Fractal**: FT-R&D - INRIA
  - **Hierarchical Component Model**
    - fractal.objectweb.org

**CBBT-Fractal**: UBS/Valoria - I3S/OCL
- **Contract Based Built-in Testing for Fractal**
STclass contributions

- **STclass** (2000-05 D.Deveaux, Y.Le Traon, JM. Jézéquel) supports a Specification and Test first approach,
**STclass contributions**

- **STclass** (2000-05 D. Deveaux, Y. Le Traon, JM. Jézéquel) supports a *Specification and Test first* approach,

- **Contract Based Built-in testing** at class level with contracts and tests inheritance for the Java language,
  - Postconditions and invariant make good and salient oracles.
  - Testing code consists only in simple method calls (scenario description)
  - Preconditions limit the scope of the test.
**STclass contributions**

- **STclass** (2000-05 D.Deveaux, Y.Le Traon, JM. Jézéquel) supports a *Specification and Test first* approach,

- *Contract Based Built-in testing* at class level with contracts and tests inheritance for the Java language,
  - Postconditions and invariant make good and salient oracles.
  - Testing code consists only in simple method calls (scenario description)
  - Preconditions limit the scope of the test.

- Why CBB-Testing rather than JUnit-Testing?
  - *separation of concerns*: **functional specification** in contracts, **dynamic specification** in senarii;
  - *better documentation*;
  - *modeling approach* rather than coding approach.
ConFract: Mastering Complexity

- **ConFract** (P. Collet, R. Rousseau) is a contracting system for the Fractal component platform
**ConFract: Mastering Complexity**

- **ConFract** (P. Collet, R. Rousseau) is a contracting system for the Fractal component platform

- **Usually, contracts are either**
  - *implicit* between software artefacts or,
  - some *interpretation of a specification* (responsibility, blame),
**ConFract: Mastering Complexity**

- **ConFract** (P. Collet, R. Rousseau) is a contracting system for the Fractal component platform.

- Usually, contracts are either
  - *implicit* between software artefacts or,
  - some *interpretation of a specification* (responsibility, blame),

- **Confract: adapting programming by contract to Fractal**
  - Contracts are first class objects
  - Types of contracts (see next slide)
  - Responsibilities
    - components are participants in contracts
    - Each participant has a well-defined responsibility, guarantor and beneficiary
Types of Contracts
Types of Contracts

Library Contract

DbC functionnal contract
Scope: inner library classes
Handled by STclass

Server interface
Client interface

<Car>

<SpeedCtrl>
csp
CarSpeed

<SensorCtrl>
att
Attributes

Prompt ppt

<SpeedCtrl>
csp

<Car>

<Car>

sns
Sensor

att
Attributes
Types of Contracts

Object contract
Scope: the 2 interfaces
Clauses: local to interfaces
Types of Contracts
Types of Contracts

- Library Contract
- Interface Contract
- External Composition Contract
- Internal Composition Contract

Server interface
Client interface

Internal behavior => assembly
Scope: composite internal int.
Clauses: f(internal interfaces, internal components)

- Types of Contracts
- CBBT Framework Principles
- Built-in Test: a Dynamic TestBed
- Contract Based Testing
- TestUnit, TestCase, TestSuite
- Populate the TestBed
- Actual and Future Works
CBBT Framework Principles

- Writing tests only relies on basic concepts:
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.
**CBBT Framework Principles**

- **Writing tests only relies on basic concepts:**
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.

- **Testing is Contract-Based:** see next slides
CBBT Framework Principles

- **Writing tests only relies on basic concepts:**
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.

- **Testing is Contract-Based:** see next slides

- **Tests are built-in:**
  - each component contains its own testing information,
  - a test controller generates a test bed that surrounds the CUT.
CBBT Framework Principles

- **Writing tests only relies on basic concepts:**
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.

- **Testing is Contract-Based:** see next slides

- **Tests are built-in:**
  - each component contains its own testing information,
  - a test controller generates a test bed that surrounds the CUT.

- **The Framework is Pure Fractal:** adaptation to different contracts models or different implementations
Built-in Test: a Dynamic TestBed

One Component Level

TestBed

Client1  C1R1

Provider1  Pr1F1

Provider2  Pr2F1

Provider3  Pr3F1

Client2  C2R1

ComponentUnderTest

 CUT CUT

CutF1  CutF2

CutF3  CutF4

CutF5  CutF6

CBBT Framework Principles

Types of Contracts

STclass contributions

Goal: a Testing Framework for Fractal

Background

Agenda
Contract Based Testing

ConFract responsibility model is reused and different kinds of test can be provided:

- Isolated testing or in situ testing,

- Black-box unit testing (TestUnit, TestCase, TestSuite)

- Gray-box testing

- Admission testing for the providers

- Test Reports
TestUnit: a scenario; cannot be executed out of a TestCase but can participate to several TestCases
TestUnit, TestCase, TestSuite

- **TestUnit**: a scenario; cannot be executed out of a TestCase but can participate to several TestCases

- **TestCase**: environment of TestUnits
  1. parameters for testing stubs
  2. setup action
  3. list of TestUnits
  4. teardown action
**TestUnit, TestCase, TestSuite**

- **TestUnit**: a scenario; cannot be executed out of a TestCase but can participate to several TestCases

- **TestCase**: environment of TestUnits
  1. parameters for testing stubs
  2. setup action
  3. list of TestUnits
  4. teardown action

- **TestSuite**: ordered list of TestCases, TestSuites or TestUnits to be activated (Composite pattern)
Populate the TestBed
Populate the TestBed

One Component Level

TestBed

ComponentUnderTest

Driver1

Driver2

Driver3

Client1

Client2

Provider1

Provider2

Provider3
Populate the TestBed

One Component Level

TestBed

Provider1
Provider2
Provider3

Client1
Client2

C1R1
C2R1

Driver1
Driver2
Driver3

T0
T0
T1

CutF1
CutF1
CutF2

ComponentUnderTest

CutR1
CutR2
CutR3

CTA

Stub1
CutR1

CTA

Stub2
CutR2

CTA

Stub3
CutR3

CTA

CTT

TC

Populate the TestBed

---

One Component Level

TestBed

ComponentUnderTest

CTC_CIT

CutF1
CutF2

Driver1
T0
CutF1

Driver2
T0
CutF2

Driver3
T1
CutF3

Provider1
Pr1F1

Provider2
Pr2F1

Provider3
Pr3F1

Tracker1

Actual and Future Works

---

Populate the TestBed

---

Agenda
● Goal: a Testing Framework for Fractal
● Background
● STclass contributions
● ConFract: Mastering Complexity
● Types of Contracts
● CBBT Framework Principles
● Built-in Test: a Dynamic TestBed
● Contract Based Testing
● TestUnit, TestCase, TestSuite
Populate the TestBed
Populate the TestBed

One Component Level

TestBed

TestRunner1

AdminDriver

Provider1

Pr1

Provider2

Pr2

Provider3

Pr3

Client1

C1R1

Client2

C2R1

Driver1

CutF1

Driver2

CutF1, CutF2

Driver3

CutF2

ComponentUnderTest

CutF1

CutF2

CutF3

CTA

Stub1

CutX1

CTA

Stub2

CutX2

CTA

Stub3

CutX3

CTA

Pr1-proxy

Pr2-proxy

Pr3-proxy

CTC CTT

Specification of CBB Testing for Fractal
Populate the TestBed
Actual and Future Works

- Tests are defined in a declarative way;
- based only on Fractal components and a test controller;
- low dependency to ConFract: adaptation to other environments;
- not limited to unit-testing: support for admission, integration and regression test;
- possible adaptation to control hierarchical testing.

A prototype of this framework is under design and construction.
CBBT-Fractal is like a "life-jacket" for Fractal Components

Questions ? ...