JUnit 4 and Java™ EE 5 Better Testing by Design

Kent Beck
Alberto Savoia

Agitar Software Inc.
www.agitar.com

TS-1580

Simplifying Developer Testing

JUnit 4 further simplifies testing for developers.

See what's new in JUnit 4 and hear what we've learned about developer testing.

JUnit 4 and Java™ Platform, Enterprise Edition 5: Better Testing by Design

- The Developer Testing Revolution
- JUnit 4
- The Growing JUnit Ecosystem
- Lessons Learned

JUnit 4 and Java EE 5: Better Testing by Design

The Developer Testing Revolution

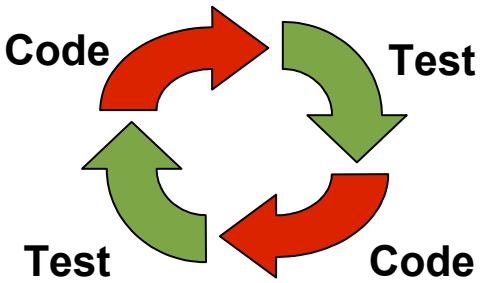
JUnit 4

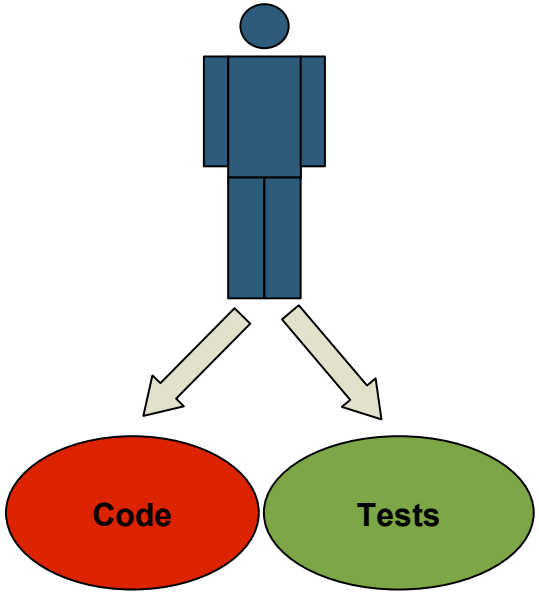
The Growing JUnit Ecosystem

Lessons Learned

What Is Developer Testing?

Developers ***creating*** and ***executing***
fast and ***flexible*** tests ***while*** developing their code





Testing Is Finally Getting Some Respect From Developers

- Agile software development and XP are gaining in popularity
  - Start-ups as well as large organizations
  - ISVs as well as IT
- Developer/unit testing is a core practice of Agile methodologies

Out of the Dark Ages—and Into the Age of Paradox

- The Dark Ages (…just a few years ago)
  - "Developers writing tests? You've gotta be kidding. Testing is for the QA folks."
- The Age of the Developer Testing Paradox (today)
  - Theoretical consensus:
    - "Developer testing is a great idea. Everyone should do it."
  - Practical reality:
    - Developer testing is still only practiced by a minority
    - Testing is a low-status activity

The Future of Developer Testing

- Scenario 1
  - Back to the dark ages
- Scenario 2
  - Developer testing is here to stay, but practiced by a small minority of organizations and developers
- Scenario 3
  - Developer testing is practiced by a majority of software development organizations

Cautious Optimism

- Many indications that developer testing is here to stay and to become a common practice
  - Web searches*:
    - “extreme programming”: ~2M results
    - “unit testing” OR “developer testing”: ~1.8M results
    - “junit + java”: ~1.5M results
  - All major IDEs have JUnit support
  - Number of books and articles on developer/unit testing
  - Unit testing a popular topic at developer conferences
  - Growing ecosystem of open source and commercial testing tools aimed at developers

*Source: results of Google™ searches February 2006

Securing the Future of Developer Testing

- Key factors
  - Belief that developer testing is the right thing to do
  - Assumption that developers want to do the right thing
- Must make it easy/easier for developers to do the right thing
- It's easier to do the right thing if you have the right tools for doing it:
  - JUnit 4
  - The JUnit ecosystem

JUnit 4 and Java5™: Better Testing by Design

The Developer Testing Revolution

JUnit 4

The Growing JUnit Ecosystem

Lessons Learned

Goals of JUnit

1. Approachable
   - Clean
   - Simple
   - Easy-to-use
   - Minimalist
2. Isolated tests
3. Fast
4. Flexible

What's New?

- No required superclass
- Fixtures identified by @Before/@After
- Tests identified by @Test
- Test for exceptions with expected=XXX
- Forward and backward compatibility

Free to Use Any Superclass for Tests

Old

```
public class Example extends TestCase
{
}
```

New

```
public class Example {
}
```

Opens up new possibilities for organizing test code through inheritance

Fixture Methods Identified by @Before

Old

```
List empty;
public void setUp() {
    empty= new ArrayList();
}
```

New

```
List empty;
@Before public void allocate() {
    empty= new ArrayList();
}
```

Now possible to have multiple fixture methods

Test Methods Identified by @Test

Old

```
public void testSize() {
    assertEquals(0, empty.size());
    empty.add(new Object());
    assertEquals(1, empty.size());
}
```

New

```
@Test public void size() {
    assertEquals(0, empty.size());
    empty.add(new Object());
    assertEquals(1, empty.size());
}
```

Avoids typographical errors and reads better

Test for Exceptions With Expected

Old

```
public void testOutOfBounds() {
    try {
        empty.get(0);
        fail();
    } catch (IndexOutOfBoundsException e) {
    }
}
```

New

```
@Test(expected=IndexOutOfBoundsException.class)
public void outOfBounds() {
    empty.get(0);
}
```

Simplifies testing for exceptions

Access to Assertions Through Static Import

```
import static org.junit.Assert.assertEquals;
public class Example {
   …
   @Test public void size() {
    assertEquals(0, empty.size());
    empty.add(new Object());
    assertEquals(1, empty.size());
   }
}
```

Special purpose assertions are easy to integrate

Forward and Backward Compatible

```
public class Example {
   …
   public static junit.framework.Test suite() {
     return new JUnit4TestAdapter(Example.class);
   }
}
```

Preserves investment in tests and runners

JUnit 4 and Java5™: Better Testing by Design

The Developer Testing Revolution

JUnit 4

The Growing JUnit Ecosystem

Lessons Learned

The Growing JUnit Ecosystem

- Open source and commercial support for JUnit
  - IDEs
  - Test generators
  - Dashboards/coverage analysis
  - Continuous testing
  - Continuous integration
  - Dedicated websites and online communities
  - Books, articles, user-groups
  - …
- JUnit influence beyond Java™ technology
  - Imitation is the sincerest form of flattery
    - A myriad of *Unit frameworks

DEMO

- New and cool JUnit-based open source tools

JUnit 4 and Java EE 5: Better Testing by Design

The Developer Testing Revolution

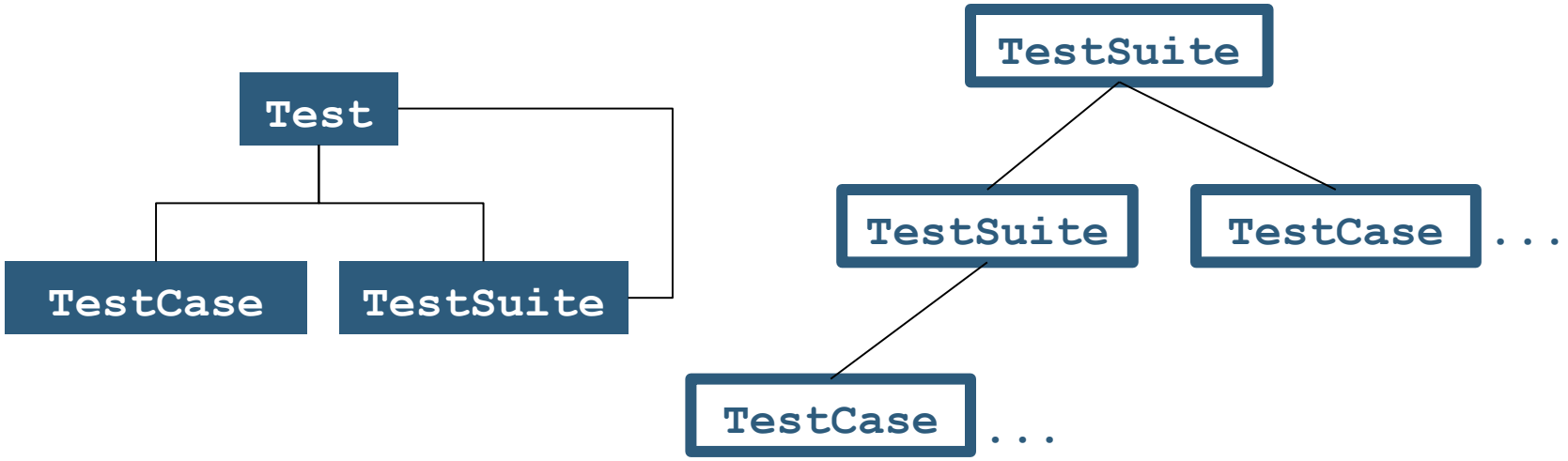
JUnit 4

The Growing JUnit 4 Ecosystem

Lessons Learned

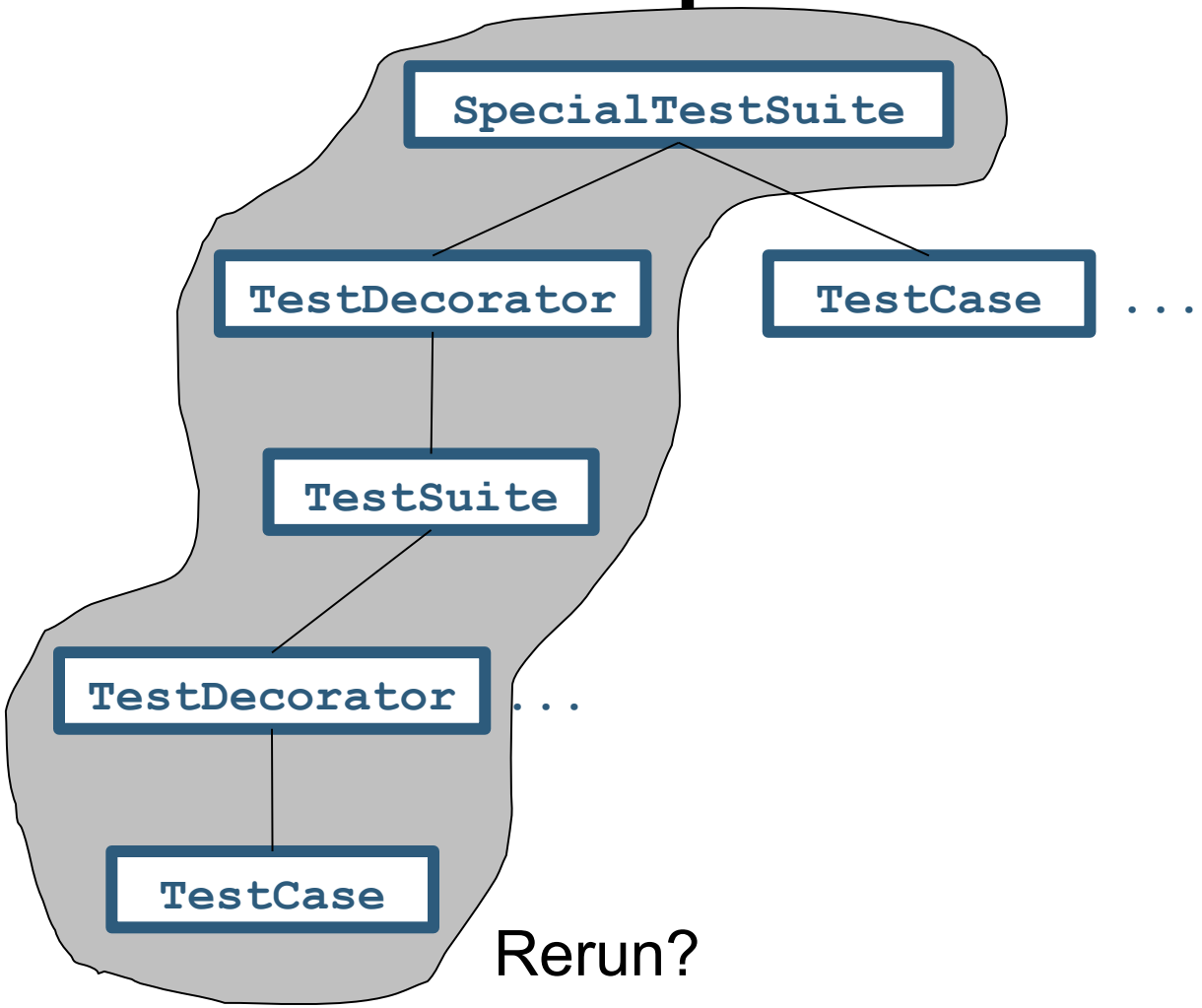
Composite Carries Hidden Costs

Good Idea on Paper, Serious Consequences In the Field



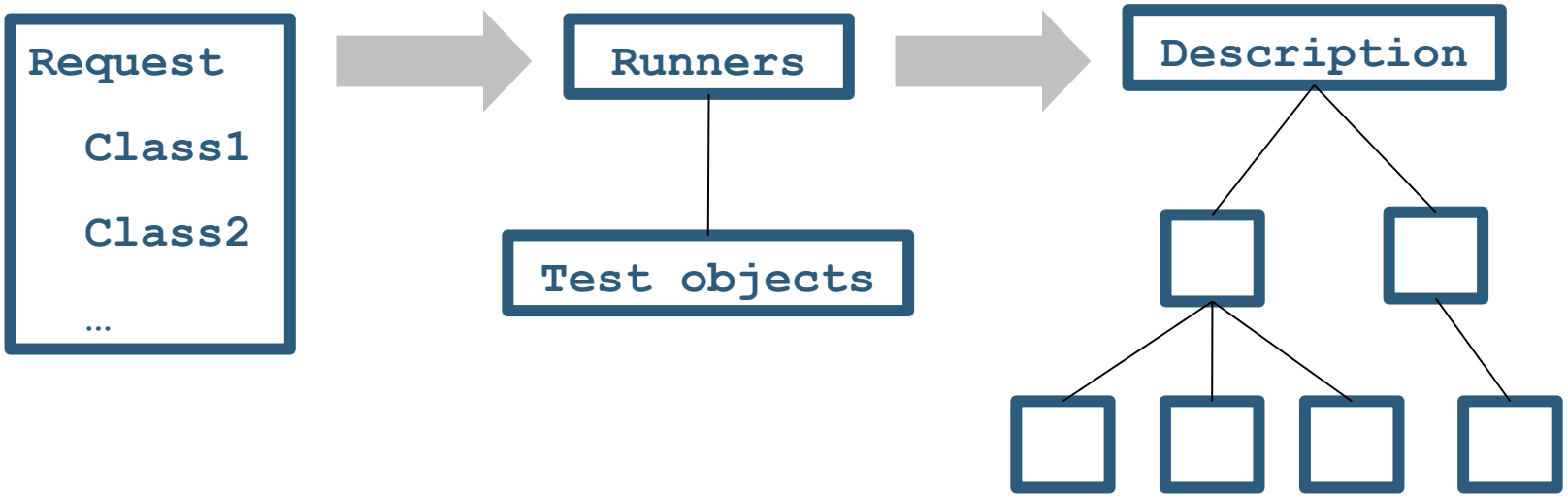
So far, so good…

Real Uses of Composite



Rerun?

Alternative: Flat Arrays, Dumb Trees



Source: org.junit.runners.*

Nothing Extends Like Success

If You Build it (and People Use it), They Will Extend it

- Reveal only what you want to support
- Everything has a purpose, even if you don't know what it is
- Sharp swords cut both ways
- Platform leadership comes with responsibility

Sometimes Good Ideas Are Not Enough

- Everyone agreed that early testing was good
- Tools were available
- Writing a new tool wasn't difficult
  - Many people had the skill to write JUnit

Tools Can Facilitate Change

- JUnit acted as the seed crystal for developer testing
  - Written by developers
  - Used familiar technology and metaphors
- Simplified entry to developers
- Now:
  - Developers test at the touch of a button
  - IDEs support testing as a first-class activity
  - Dashboards keep you apprised of the overall state of project tests
  - Add-ons amplify the value of tests

Cultural Change Happens

At the Scale of Decades

- Shift to developer testing harmonizes with social trend toward accountability in business
- Basic ideas had been around for decades
- Evolution of programming is not just driven by technology
  - Iteration length
  - Deployment frequency
  - Business models
  - Importance of relationships

Summary

- The developer testing revolution is under way
- The growing JUnit ecosystem is evidence of progress
- JUnit 4 is designed to make developer testing even easier
- Try it and give us your feedback so we can continue to improve on it

Thank you Erich Gamma, David Saff, Mike Clark (FAQ), Erik Meade (webmaster), and our reviewers

For More Information

- JUnit
  - junit.org—the starting point for exploring the JUnit ecosystem
  - JUnit Yahoo! group
- Continuous Testing
  - pag.csail.mit.edu/continuoustesting
- Other community websites/blogs of interest
  - testdriven.com
  - developertesting.com
  - Threeriversinstitute.org

Q&A

JUnit 4 and Java™ EE 5 Better Testing by Design

Kent Beck

Alberto Savoia

Agitar Software Inc.
www.agitar.com

TS-1580