Develop Software with Confidence

Agitar @ Java Forum Stuttgart
Artur Hildebrandt, Solutions Consultant, July 6th 2006
Our Mission

Profoundly improve the quality of software and the economics, transparency, and agility of software development.

By making developer testing effective for the enterprise.
Software Development Is Still Immature . . .

- $100+ Billion wasted annually on software bugs
- Only 29% of IT projects succeed
- Projects take 84% longer than originally scheduled
- 50% of all software projects are total failures

The Traditional Approach Is Flawed

<table>
<thead>
<tr>
<th>Development</th>
<th>Quality Assurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Construction</td>
<td>Debug &amp; Rework</td>
</tr>
<tr>
<td></td>
<td>Integration &amp; System Testing</td>
</tr>
</tbody>
</table>

Source: "The Software Development Paradox", Alberto Savoia, 2004
There Is a Better Approach

**Test Bugs Out**

- Give QA the main responsibility for quality
- Test late
- Use unverified components
- Deliver, and fix defects later

**Build Quality In**

- Hold everyone responsible for quality
- Test early and often
- Use verified components
- Stop the assembly line

**Typical software development process**

**Mature world-class manufacturing process**
Developer Testing Is the Solution

Also known as “unit testing” or “programmer testing”

- Each code unit is accompanied by unit tests that validate and document its correct behavior
- The developer creates these unit tests while coding
- Each unit test is self-sufficient and self-evaluating so it can be run and checked automatically
- The unit tests are run very frequently
- Developer testing may also include functional (scenario) testing done by developers, often using JUnit
Find and Fix Defects Sooner

Continuous Feedback

Development with Developer Testing

Integration & System Testing by QA

The Economics Are Compelling

Percentage of Bugs

- Coding: 85%
- Unit Test
- Function Test
- System Test
- After Release

Cost to repair defects:
- $25
- $100
- $250
- $1000
- $16,000

% Defects introduced in this phase
% Defects found in this phase
$ Cost to repair defect in this phase

Source: Applied Software Measurement, Capers Jones, 1996
Why Is Developer Testing Not Yet Pervasive?

**Poor Division of Labor**
- Developers deliver code, not knowing if it works
- QA verifies, integrates, and does system tests, but cannot fix problems
- Detection and repair is deferred until long after the bug is created

**Too Much Manual Work**
- A combinatorial problem, so full coverage is hard for unit tests or debugger
- Often 2x – 4x more test code than code to test
- Most time spent in tedious setup and data creation
- Tests get stale
- Tough to be thorough and meet the schedule

**Uncertain Status and Goals**
- Are you done, and have you done enough?
- What has been tested and what has not?
- Have you focused enough on the riskiest or most complex code?

Manual approaches (usually JUnit-based) tend to stall.
**Agitator®: Unprecedented Automation**

1. **Analyze**
   - Code
   - Software Agitation

2. **Summarize**
   - Observations of Code Behavior, and Coverage Data

3. **Review**
   - Create Tests
   - If observation describes desired behavior, click to promote it to assertion

4. **Fix Bug**
   - If observation reveals a bug, fix it!

**Developer**

**Validate intended code behavior**

- Discover invariants (or specify and validate them)
- Achieve high data and state coverage without manual setup
- Test without needing application server, database, etc.
Testing with JUnit

Environment and Testdata Setup
100% Manual

Assert Statement
100% Manual

Teardown
100% Manual

100% manual work

Testing with Agitar

80% Automated
Environment and Testdata Setup

Manual Testdata via Factories and Mocks

Assert Statement
100% Manual

Promote Observation/Additional Assertion

100% Automated
Teardown

0% Manual

10-30% manual work

100% Automated
100% Manual

100% Automated
100% Manual
Improve the Developer’s Daily Work

CREATE
High-quality new code

FIX
Defective code quickly

CHANGE
Existing code without fear

Goal: code that works

Typical Development

Developed with Agitator

40% Coding
20% Testing
40% Rework

10% Coding
60% Testing
30% Rework

© Agitar Software, 2006
Real Visibility into Unit-Level Quality

Project Summary

<table>
<thead>
<tr>
<th>Test Results</th>
<th>Current</th>
<th>Changes²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assertion Failures</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Outcome Failures</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rule Errors</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Targets</th>
<th>Current</th>
<th>Changes²</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>90.1 %</td>
<td>-0.6</td>
<td>95 %</td>
</tr>
<tr>
<td>Test Points</td>
<td>5504</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>Classes with Test Points</td>
<td>100.0 %</td>
<td>0.0</td>
<td>95 %</td>
</tr>
<tr>
<td>Methods with Test Points</td>
<td>88.6 %</td>
<td>0.0</td>
<td>80 %</td>
</tr>
<tr>
<td>Outcomes with Test Points</td>
<td>83.2 %</td>
<td>0.0</td>
<td>75 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overrides</th>
<th>Current</th>
<th>Changes²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bugs</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Suppressions</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Coverage</td>
<td>27</td>
<td>0</td>
</tr>
</tbody>
</table>

Test Failures

Coverage %
### Metrics for High-Performance Teams

**Project dashboard**

<table>
<thead>
<tr>
<th>Test Targets</th>
<th>Current</th>
<th>Changes*</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>89.8%</td>
<td>0.0</td>
<td>95 %</td>
</tr>
<tr>
<td>Test Points</td>
<td>5273</td>
<td>5</td>
<td>5000</td>
</tr>
<tr>
<td>Classes with Test Points</td>
<td>100.0 %</td>
<td>0.0</td>
<td>95 %</td>
</tr>
<tr>
<td>Methods with Test Points</td>
<td>75.5 %</td>
<td>-0.1</td>
<td>80 %</td>
</tr>
<tr>
<td>Outcomes with Test Points</td>
<td>76.2 %</td>
<td>-0.0</td>
<td>75 %</td>
</tr>
</tbody>
</table>

### High code coverage

### Every class has assertions

### High % of methods and outcomes have assertions

### 1:1 ratio of assertions to lines of code

**Project Code Metrics**

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>Changes*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Classes</td>
<td>131</td>
<td>0</td>
</tr>
<tr>
<td>Test and Harness Classes</td>
<td>189</td>
<td>0</td>
</tr>
<tr>
<td>Project Methods</td>
<td>1355</td>
<td>1</td>
</tr>
<tr>
<td>Test Methods</td>
<td>416</td>
<td>0</td>
</tr>
<tr>
<td>Project Executable Lines</td>
<td>5279</td>
<td>1</td>
</tr>
<tr>
<td>Test Executable Lines</td>
<td>7070</td>
<td>0</td>
</tr>
</tbody>
</table>
Demo Placeholder
Add Agitator to Your JUnit Testing

- 95% of our customers start with JUnit
- JUnit is not required to use Agitator
- Agitator and Dashboard understand and leverage existing JUnit
The Leader in Developer Testing

Unprecedented Recognition

Top-Tier Investors

SEQUOIA CAPITAL®
NEA® New Enterprise Associates®
GLOBESPAN CAPITAL PARTNERS

Gartner

SDTimes
2005 100

JOLT
15th annual product excellence award

CTO 25 AWARDS

FORRESTER
Helping Business Thrive
On Technology Change

InfoWorld
5th annual excellence awards

© Agitar Software, 2006