Pushing the Boundaries of Testing and Continuous Integration

Fabrizio Cannizzo
Raghav Ramesh
Robbie Clutton
Who are we?

http://flickr.com/photos/kerrybuckley/1356791754/in/pool-web21c

http://flickr.com/photos/zoonabar/159442457/
Agile to the Core!

XP/Scrum
- TDD
- Pair programming
- Continuous Build/Integration
  - Unit/Acceptance Test
  - Reporting
Feature completeness ≠ completed product

http://flickr.com/photos/mattimattila/2188826561/
What's Missing?

Robustness

Performance

Scalability

Failover
We needed to know the thing would stand up

http://flickr.com/photos/alan-light/2435511982/
But we didn't want to wait for formal QA testing
We had to last three spooky nights

http://flickr.com/photos/luchilu/677786684/
After four weekends, we were done

Bugs
Memory Leaks
Poor Environment
Log Extractors

http://flickr.com/photos/22243977@N06/2457787000/
What next?
Create scenarios
Define expectations
Automate
Continuously test
So... how does it **really** work?
Technologies

Cruise Control
Java Spring Framework
Hand crafted Java code
JMX Console
UltraMonkey
SIPp
JainSIP
Mockphones
Cruise control is the engine that continuously runs typical usage scenarios.

Ensure the entire scenario played out as expected.
Why are scenarios relevant?

Robustness

Performance

Scalability

Failover
We **automated** phone behaviour using SIPp...

Different **configurations** are implemented using different spring application contexts and network setups.

**Scenario runners** are java applications.

**Scenarios** are POJOs loaded from spring application context.
<table>
<thead>
<tr>
<th>Project</th>
<th>Status (since)</th>
<th>Last failure</th>
<th>Last successful</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>callFlow-robustness</td>
<td>building (12:00)</td>
<td></td>
<td>09:31</td>
<td>callFlow-robustness-build.185</td>
</tr>
<tr>
<td>RobustnessDatabase</td>
<td>waiting (08:31)</td>
<td></td>
<td>17:29</td>
<td>RobustnessDatabase-build.334</td>
</tr>
<tr>
<td>RobustnessPerformance</td>
<td>waiting (08:31)</td>
<td></td>
<td>17:33</td>
<td>RobustnessPerformance-build.242</td>
</tr>
<tr>
<td>RobustnessPerformanceTerracotta</td>
<td>waiting (10:48)</td>
<td></td>
<td>10:48</td>
<td>RobustnessPerformanceTerracotta-build.232</td>
</tr>
<tr>
<td>SipStoneTerracotta</td>
<td>waiting (11:03)</td>
<td></td>
<td>10:48</td>
<td>SipStoneTerracotta-build.29</td>
</tr>
<tr>
<td>SipStoneDatabase</td>
<td>waiting (11:20)</td>
<td></td>
<td>11:04</td>
<td>SipStoneDatabase-build.55</td>
</tr>
<tr>
<td>SipStoneMemory</td>
<td>waiting (11:36)</td>
<td></td>
<td>11:20</td>
<td>SipStoneMemory-build.45</td>
</tr>
<tr>
<td>SipStoneDatabaseHA</td>
<td>waiting (11:52)</td>
<td></td>
<td>11:36</td>
<td>SipStoneDatabaseHA-build.17</td>
</tr>
<tr>
<td>RobustnessMemory</td>
<td>waiting (11:56)</td>
<td></td>
<td>11:52</td>
<td>RobustnessMemory-build.381</td>
</tr>
<tr>
<td>RobustnessDatabaseMultistack</td>
<td>waiting (12:00)</td>
<td></td>
<td>11:56</td>
<td>RobustnessDatabaseMultistack-build.185</td>
</tr>
</tbody>
</table>
BUILD COMPLETE - RobustnessPerformance-build.242

Date of build: 04/27/2008 17:33:28
Time to build: 0 minutes 30 seconds
Last changed: 03/14/2008 09:57:49
Last log entry:

Build Artifacts

Errors/Warnings: (27/20)

log4j WARN No appenders could be found for logger (org.springframework.context.support.ClassPathXmlApplicationContext).

log4j WARN Please initialize the log4j system properly.

2008-04-27 17:33:38 INFO [PerformanceBatchTest::main()] - Loading application context

Java version: 1.6.0_13, vendor: Sun Microsystems Inc.


Jain Sip Jars: jain-sip-api-1.2.jar, jain-sip-r-1.2.jar

Jain Sip Thread Count: 200, Simple Sip Stack Thread Count: 200, Database Pool Thread Count: 20

Number of scenarios to be run: 30, number of application threads: 16

2008-04-27 17:33:41 INFO [PerformanceBatchTest::logSystemInformation()] - OS Name: Linux, version: 2.6.5-7.202.7-smp

2008-04-27 17:33:38 INFO [PerformanceBatchTest::main()] - Running tests with 10 concurrent threads

2008-04-27 17:33:38 INFO [PerformanceBatchTest::main()] - Running tests with 10 concurrent threads

2008-04-27 17:33:42 INFO [BatchTestScenarioBase::start()] - Starting scenario createCallTerminateCallScenario:61e71398f8aa18756684f75c4ed054b90, createCallTerminateCallScenario:61e71398f8aa18756684f75c4ed054b90

2008-04-27 17:33:42 INFO [BatchTestScenarioBase::start()] - Starting scenario createCallTerminateCallScenario:487134addc777c36c156884e5e301040, createCallTerminateCallScenario:487134addc777c36c156884e5e301040

2008-04-27 17:33:42 INFO [BatchTestScenarioBase::start()] - Starting scenario createCallTerminateCallScenario:17515b48e084d0232fe163cd814eac5, createCallTerminateCallScenario:17515b48e084d0232fe163cd814eac5

2008-04-27 17:33:42 INFO [BatchTestScenarioBase::start()] - Starting scenario createCallTerminateCallScenario:7370eaa309e6c01ec65cd25c73d00f0a, createCallTerminateCallScenario:7370eaa309e6c01ec65cd25c73d00f0a

2008-04-27 17:33:42 INFO [BatchTestScenarioBase::start()] - Starting scenario createCallTerminateCallScenario:606b27d032192826e1f4f22ed1702, createCallTerminateCallScenario:606b27d032192826e1f4f22ed1702

2008-04-27 17:33:42 INFO [BatchTestScenarioBase::start()] - Starting scenario createCallTerminateCallScenario:3370eaa309e6c01ec65cd25c73d00f0a, createCallTerminateCallScenario:3370eaa309e6c01ec65cd25c73d00f0a

2008-04-27 17:33:42 INFO [BatchTestScenarioBase::start()] - Starting scenario createCallTerminateCallScenario:206b27d032192826e1f4f22ed1702, createCallTerminateCallScenario:206b27d032192826e1f4f22ed1702
Scenario createCallTerminatesCallScenario:52ce01f401018245b6304f1d688d87a73 succeeded
Scenario twoCallsSharingCalLegScenario:98b6c00b0f02010b362622c8547f6db succeeded
Scenario callAnswerTimeoutScenario:7410f009379f994a670cb4767397bd succeeded
Scenario createCallTerminatesCallScenario:749a899e03fa95f155e0e030e529a1 succeeded
Scenario badAddressScenario:3538ad4e6ee57e06700195056e00189f79fd succeeded
Scenario maxCalDurationScenario:068396c0c277e853713a994d057ad5c succeeded
Scenario maxCalDurationScenario:3efb59d737b89303417c8305a5216f6 succeeded
Scenario callAnswerTimeoutScenario:1a1dc05e883c813d53c8d31318ad4a69 succeeded
Scenario badAddressScenario:12330a107d4a230e2a17299ea0537a07d successfully
Scenario badAddressScenario:4a9c807f1033763752e540e0be2e6f3 succeeded
Scenario maxCalDurationScenario:58a9eb7b864853860138c87165904123 succeeded
Scenario maxCalDurationScenario:e1b757c5f4eb06417caca5ac2b35c8 succeeded
Scenario createCallTerminatesCallScenario:59cc12744c189ce707a2689ac3b238 succeeded
Scenario twoCallsSharingCalLegScenario:17685a66c6d930e034e02e096956c succeeded
500 tests run
Average delay between scenarios: 1202.330000ms
announcementCallScenario: 14/14 (100.0%)
badAddressScenario: 91/91 (100.0%)
callAnswerTimeoutScenario: 0/0 (0.0%)
createCallTerminatesCallScenario: 104/104 (100.0%)
firstCallLegFailureScenario: 0/0 (0.0%)
maxCalDurationScenario: 119/119 (100.0%)
secondCallLegFailureScenario: 0/0 (0.0%)
twoCallsSharingCalLegScenario: 77/77 (100.0%)
Overall success rate: 100.0%
ReplaceList invocations at dialog level: 5140, at call level: 2660, at conference level: 0, combined: 7800
Concurrent Exceptions at dialog level: 12, at call level: 19, at conference level: 0, combined: 31

Unit Tests: 0
No Tests Run
This project doesn't have any tests
TestRR

- Generic Framework
  - Robustness
  - Performance

Who else is talking about this?

**James Bull** - Pragmatic Performance Testing - [http://is.gd/SUv](http://is.gd/SUv)

**Dr. Dobbs** - Continuous Integration and Performance Testing - [http://is.gd/12Ds](http://is.gd/12Ds)

**Manfred Lange** - Introducing Automated Performance Testing - [http://is.gd/SUj](http://is.gd/SUj)

**Paul King & Dalton Cranston** – Open Source Performance Testing Tools For The Web
What we have learned that you could take away...

Automate *early* and *often*

**unit** and **acceptance** tests are an essential first step

maintaining these builds can have severe impact on **velocity**

the **customer** must buy in: formulate user stories to drive the creation of tests
... and also...

This drives creation of robust code earlier

Leverage existing frameworks for performance testing

Build up testing strategies incrementally

These tests are resource hungry, use an appropriate environment
Q & A

Fabrizio Cannizzo:
- http://smartrics.blogspot.com

Raghav Ramesh:
- http://ragstorooks.wordpress.com

Robbie Clutton:
- http://blog.iclutton.com

Aloha SIP A/S: https://trac.osmosoft.com/Aloha
Web21C SDK: http://web21c.bt.com