Performance and Test-Driven Development: Are they compatible?

Alan Gordon & David Evans
Test Management Summit 2009
## Performance Test-Driven Development
### The Premise

<table>
<thead>
<tr>
<th>‘David’</th>
<th>‘Alan’</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔ Does agile, test-driven development</td>
<td>✔ Does performance testing</td>
</tr>
<tr>
<td>✔ Never been a performance tester</td>
<td>✔ Self-confessed agile novice</td>
</tr>
<tr>
<td>✔ But knows it’s important...</td>
<td>✔ Wants to start performance testing as early as possible on a project</td>
</tr>
</tbody>
</table>

- They seem to have different views on how things should be done
- They want your help to resolve this
- They want to know your experience and your views
- They want to know what you think of our ideas
Agile Development basics

- Avoid ‘Big Design Up Front’
- Develop iteratively, based on the Customer’s current priority
- Deliver valuable, working software at the end of every iteration
- Evolve functionality through User Stories

User Story basics

- Stories represent tangible functionality for a user or other stakeholder
  - “As a <role type>, I want <some function>, so that <some value>”
- Stories are like ‘pieces of cake’
  - Vertical slices of usable functionality, not layers of architecture
Performance Test-Driven Development
Recap: Test-Driven Development

- Test-Driven Development basics
  - Use testable acceptance criteria to elaborate each Story
  - Specify and automate the tests before the Story is implemented
  - The Story is 'done' when all the acceptance tests pass
  - Re-test all completed Stories every iteration, to prevent regression

- TDD preferences
  - Isolate the components under test
  - Favour fast tests for fast feedback
  - Use code coverage analysis to evaluate your test suite
Recap: Performance Testing

- Performance Testing – the “Classic Model”:
  - Test in a production-like environment
  - Test a mature application
  - Test the whole integrated system
  - Use a dedicated performance specialist team
  - Use specialist tools
  - Test end-to-end from the user perspective
  - Pass / fail status may be subjective
  - Tests are long & hard – therefore expensive
    - Schedule testing carefully to minimise cost
Performance Test-Driven Development

Initial Issues...

- Iterative development
  - Testing in the iteration typically does not include performance testing
  - Performance is non-linear – can’t guarantee how close to complete we are

- System under test
  - ‘The whole’ is an evolving thing

- Knowledge
  - The Customer has the final say on priority and acceptance criteria, but is probably no expert on performance

- Performance and other non-functional requirements
  - Can be cumbersome to express as User Stories

- Environments
  - Early test environments are unlikely to be production-like and may be unstable
## Performance Test-Driven Development

### How do the basics measure up?

<table>
<thead>
<tr>
<th>Test-Driven Development basics</th>
<th>Performance Testing basics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use testable acceptance criteria to elaborate each Story</td>
<td>✓</td>
</tr>
<tr>
<td>Specify and automate the tests before the Story is implemented</td>
<td>?</td>
</tr>
<tr>
<td>The Story is 'done' when all the acceptance tests pass</td>
<td>✓</td>
</tr>
<tr>
<td>Re-test all completed Stories every iteration, to prevent regression</td>
<td>?</td>
</tr>
<tr>
<td>Isolate the components under test</td>
<td>✓</td>
</tr>
<tr>
<td>Favour fast tests for fast feedback</td>
<td>✗</td>
</tr>
<tr>
<td>Use code coverage analysis to evaluate your test suite</td>
<td>✗</td>
</tr>
<tr>
<td>Test in a production-like environment</td>
<td>✗</td>
</tr>
<tr>
<td>Test a mature application</td>
<td>✗</td>
</tr>
<tr>
<td>Test the whole integrated system</td>
<td>✗</td>
</tr>
<tr>
<td>Use a dedicated Performance specialist team</td>
<td>✗</td>
</tr>
<tr>
<td>Use specialist tools</td>
<td>?</td>
</tr>
<tr>
<td>Test end-to-end from the user perspective</td>
<td>?</td>
</tr>
<tr>
<td>Pass / fail status may be subjective</td>
<td>?</td>
</tr>
<tr>
<td>Test long &amp; hard</td>
<td>?</td>
</tr>
</tbody>
</table>

---

**Our conclusion:** The “classic” way of performance testing will have to adapt
Performance Test-Driven Development
Possible Models

- Iterative development, waterfall test
  - Build & System Test
  - Build & System Test
  - Build & System Test
  - Performance Test
  Pros: Efficient, good quality performance testing
  Cons: Least agile, less opportunity for improving performance

- In-iteration performance testing
  - Build, System & Performance Test
  - Build, System & Performance Test
  - Build, System & Performance Test
  Pros: Fast feedback, most agility
  Cons: Resource and time intensive

- Parallel non-functional testing team
  - Build & System Test
  - Build & System Test
  - Build & System Test
  - Performance Test
  - Performance Test
  - Performance Test
  Pros: Reasonably fast feedback
  Cons: Still not very agile
 Performance Test-Driven Development

Discussion...

- What are your experiences?
  - Have you encountered these issues?
  - Have you encountered others?

- What are your views on the models?
  - What are the pros and cons?
  - What other models have you tried?

- Who should be doing the performance testing?
  - Testers embedded in agile team?
  - Separate team dedicated to project?
  - Centralised performance test service?

- How should the Customer prioritise performance issues in the backlog?
A little bit of everything when pragmatic and suited to context

- Run whatever tests we can immediately after each build – likely to be unit or component level, e.g. multi-thread tests, queries, simple volume tests
- Performance tester embedded in team to ensure involvement
- Separate performance team building a more “heavyweight” test suite
- Given time to run soak tests, multiple test scenarios on larger environment
Performance Test-Driven Development
Towards an updated approach

Classic Performance Testing

- Test in a production-like environment
- Test a mature application
- Test the whole integrated system
- Use a dedicated Performance specialist team
- Use specialist tools
- Test end-to-end from the user perspective
- Pass / fail status may be subjective
- Test long & hard

Our Approach

- Test in the best environment you have
- Test from the first iteration
- Test individual system components or modules first, then whole system later
- Whole team appreciates performance testing, no single points of knowledge
- Use any tool or range of tools
- Test internals of system as well as E2E
- Always have defined success criteria
- Parallel non-functional test stream?
Performance Test-Driven Development

Some Thoughts...

- Avoid increasing ‘undone work’ every iteration
- Remove all faults as soon as possible after they are injected
- Educate the Customer about non-functional requirements
- Include performance requirements on Stories where relevant
- Describe overall NFR’s as Constraints
- Define quantifiable goals for non-functional qualities
- Gain as much useful information as quickly and cheaply as possible
- Refactor performance tests regularly

Discussion:

- What is your opinion now?
- Will you do anything different next time?
Performance Test-Driven Development
Some Good References on the topic

- Scott Ambler
  - http://www.ddj.com/architect/210601918
- Tom & Kai Gilb
  - http://gilb.com/Requirements
- Scott Barber
- Jamie Dobson
  - http://www.jamiedobson.co.uk/?q=node/21
- Mike Cohn
Thanks for your attention

Alan.Gordon@sqs-uk.com
David.Evans@sqs-uk.com

7 Moorgate | London, EC2R 6AF, United Kingdom
Tel.: +44 (0) 20 7448 4620 | Fax: +44 (0) 20 7448 4651
E-Mail: info@sqs-uk.com
Internet: www.sqs-uk.com | www.sqs-group.com