Agile Performance Testing

Cesario Ramos  
Independent Consultant  
AgiliX Agile Development Consulting
Overview

• Why Agile performance testing?
• Nature of performance testing
• Agile performance testing
Why Agile Performance Testing?
We experience

Primary problems after project release are NOT system crashes or incorrect system responses

Primary problems are
1. Responsiveness degradation
2. Inability to handle required throughput

Source: Weyuker E.
Webshop for phones

• Worked good during testing

• In production system failed after 15 minutes
  – Authentication server load was not taken into account 😞
We know
We see

- Hand-offs
- Defects
- Delay
- Relearning
- Inventory
- Task Switching
We use
We expect?

- To decrease overall lead-time
  - Throughput accounting v.s. Cost accounting
- To have less problems in production
  - Decrease risk, increase reliability
- To achieve overall better quality
  - Usability, Stability, Performance
Still we see mostly
Pre-production validation only!
WHY?
WHY?

• Performance test team is expensive, overworked
  – We can only test the complete system
  – Performance testing against a moving target
  – What happens to my baselines during sprints?
  – “First make it work, then make it fast”

• Deployment team is expensive, overworked
  – Deploying to test environment is very difficult and time consuming
  – The fewer deployments the better
WHY?

• “premature optimization is waste!”
• Separate teams
  – Developers, Testers, Deployers, Analysts
• Costs are high for frequent deployment
• Production like environments are expensive
• We need specialists for performance
• ...

© AgiliX Agile Development Consulting
What could we do?
Performance tests executed throughout the project
Performance activities, refined and executed in collaboration.
What’s in it for you?

• Project manager
  – Improved quality.
  – Decreased risk.

• Developer
  – Will have effective discussions with requirement engineers and performance testers.

• Tester
  – Not waste any time on trivial problems.
  – Help the team earlier and more efficiently.
Nature of performance testing
Grey box testing

- You must understand how it works in order to write relevant test scripts.
- You must understand the architecture.
Beware of

Integration points
Chain reactions
Blocked threads
Unbalanced capacities

No more simple record playback

Source: Nygard M.
Build stub components

• Simulate SLA's
  • Test against the stubs

• Create failure modes
  – Slow network responses
  – Huge data responses
Correlation problem

Per server
- Thread pools
- Connection pools
- Request Response times
- CPU usage
- Memory usage
- JVM behavior
- Code hotspots
Tuning

Who dominates the CPU?
- Operating system? (OS system load on CPU)
- JVM? (object lifecycle, garbage collection)
- Application? (algorithms)
- None (blocked threads most of the time)

There’s always one most prominent performance problem

All performance problems express themselves in the hardware

Source: Kirk Pepperdine
Agile performance testing
• Feature driven planning is needed
  – Build a release plan where you performance test every few sprints, when couple of related features are done!

• Need for establishing Baselines
  – Deployed and production
  – Useful when architecture stabilizes
• Invest in automatic
  – Deployment
  – Smoke tests
  – Integration testing
Exploratory performance testing

• Local performance tests
  – Memory leaks
  – Memory usage patterns
  – Threading issues

• Code Reviews
  • Checks on use of boundedness, use of timeouts
  • Use of caching, cache configuration
  • Best practices on algorithms, synchronization, designs, value objects, functional programming, modularization, ...
• Deployed performance tests
  – Workload issues
    • #clients, request frequency, arrival rate, duration
  – Configuration
    • Thread pool sizes, Connection pool sizes, Cache sizes, JVM configuration, queue sizes, ...
  – Application
    • Replication, remote methods calls, database interactions, asynchronous messages, security service, ...
When a performance problem is found

1. Write a failing test for it
2. Make it work local
3. Make it work on the deployed environments
Summarizing

Performance testing is agile and iterative by nature!

Grey box testing
Avoid re-runs
Automatic deployment
Need to correlate various measurement results
Establish baselines
Agile Performance Testing

Cesario Ramos
Independent Consultant
AgiliX Agile Development Consulting

www.agilix.nl
www.codecentric.nl
cesario@agilix.nl