

UCL



Tools for Testing Software Architectures

Wolfgang Emmerich
 Professor of Distributed Computing
 University College London
<http://sse.cs.ucl.ac.uk>

UCL

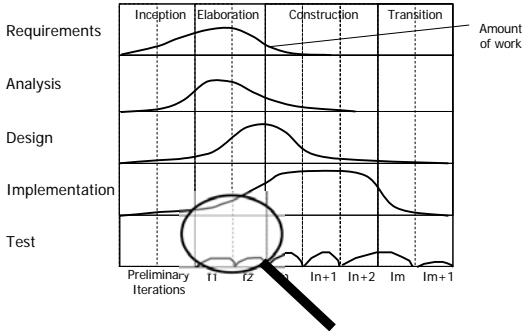
Learning Objectives

- To discuss tools to validate software architectures
- Review ways in which implementations of software architecture can be tested against different non-functional requirements
- Gain some practical experience with some such tools (Apache's Jmeter)

2

UCL

Context



3

Elaboration Stage

- test design products by analysis, simulation, walkthroughs and inspection
- generate user acceptance test cases from use cases
- generate test cases that validate non-functional requirements
- implement test architecture that automates tests
- execute non-functional tests to validate lifecycle architecture

4

Aim of validating software architectures

- Ascertain that all relevant non-functional requirements are being met by the life cycle architecture
- Can be done analytically
 - Modelling the software architecture using an appropriate formalism (e.g. process algebra or queuing network)
 - Subjecting the models to analysis (e.g. reachability analysis or solution of queuing networks)
- Problem with analytical approaches: models often do not correspond to the developed architecture
- Alternative: testing the executable code that implements the architecture.

5

Overview of architecture test strategy

- Ease of deployment - Measure time to deploy
 - Openness - Integration test interfaces
 - Legacy Integration - Integration test interfaces
 - Usability - Conduct usability experiment
 - Security - Get expert hackers to break in
 - Availability
 - Latency
 - Throughput
 - Scalability
- } - Conduct performance tests

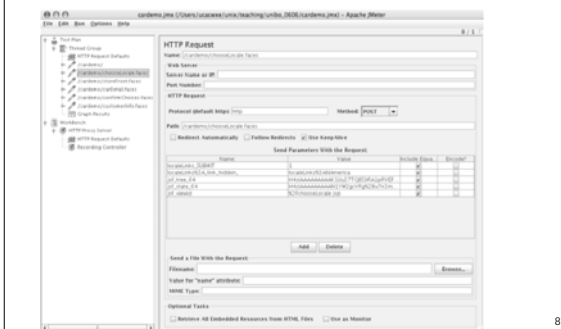
6

Performance tests - latency under no load

- Work out how the system performs in the best case scenario
- Identify use cases and user stories that are sensitive to latency
- Translate them into performance test cases that exercise the non-loaded system and measure the latency.
- Use performance testing tools, e.g.
 - Mercury LoadRunner or
 - Apache Jmeter (open source)
 to perform the tests

7

Using JMeter to test online car dealership



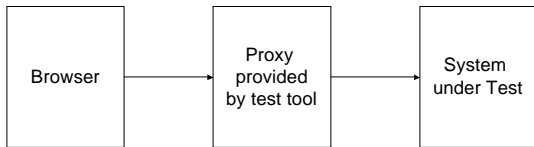
8

Capture and Replay

- Defining test cases at protocol level cumbersome
- When possible use capture and replay capabilities offered by testing tools, e.g. in
 - Mercury WinRunner
 - Apache Jmeter
- Capture interactions with the system at protocol level by inserting proxies between the browser and the system under test
- Use the captured interaction as baseline for test cases

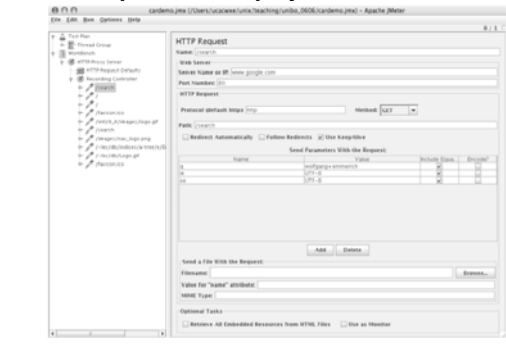
9

Architecture for Capture and Replay



10

JMeter capture and replay demo



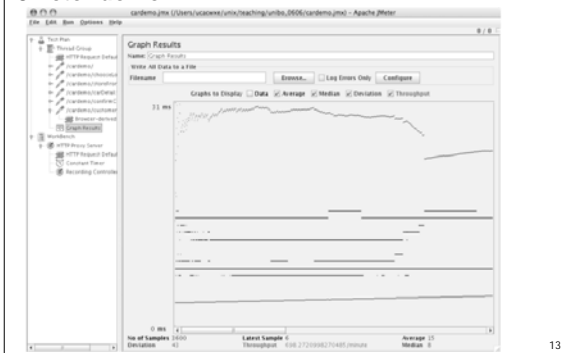
11

Testing latency under no load

- Sample system under test sufficiently often to have a statistically significant sample set of test execution
- Observe minimal, average and maximum latency
- Observe how latency behaves over an extended period of time. Latency can be adversely affected by:
 - Garbage collection in Java containers
 - Memory leaks (and yes they also exist in Java and C#!)
 - Increases in data sets / file sizes
- Latency observed represents the best case scenario for the latency real users will experience

12

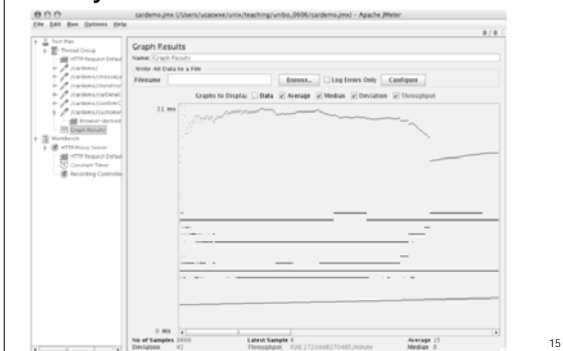
JMeter demo



Testing latency under load

- Determine how the maximum number of concurrent users
- Determine typical user behaviour (e.g. by measuring the time users are idle between requests in a capture/replay session)
- Define a number of test cases that gradually increase numbers of users up to the max. required
- Measure the latency of the system under test in these different test cases

Latency under load with JMeter



Testing throughput

- You can use the same test cases used for determining latency under load also for throughput.
- Expect the throughput to have a maximum when latency begins to increase as load increases.
- Ideally you want to ascertain that the throughput is constant no matter what the load.
- In practice throughput deteriorates under high loads (due to e.g. memory paging).

16

Testing availability

- Leave a load test running for an extended period of time (e.g. day or even a week) to see whether the system performance degrades during that period
- Then leave a load test running for a shorter period and selectively switch off hosts in your distributed system
- Observe how the system reacts
- Expect to see latency and throughput degradation due to redistribution of load


17

Testing scalability

- Add hardware resources to your distributed system
- Verify that you do not need to change the software of your distributed system (apart from deployment configurations)
- Repeat the throughput tests
- Ascertain that the throughput is now higher
- Throughput increase should be proportional to the hardware resources you have added.

18

UCL



Key Points

- Architecture tests complement architecture analysis
- Performance testing requires an executable baseline architecture
- Built in RUP/USDP during Elaboration
- Performance tests can assess whether architectures meet requirements on
 - Latency
 - Throughput
 - Availability
 - Scalability

UCL

References

- Jmeter. <http://jakarta.apache.org/jmeter>
- Eclipse Test and Performance Tool Platform. <http://www.eclipse.org/tppt/>

20
