

Testing Strategies

Brett G. Palmer

Chief Architect,
In2M Corporation

Outline

- Why does testing fails?
- Misconceptions of testing
- Testing Definition Review
- Approaches to Testing
- Available Testing Tools
- Q & A

Why Testing Fails?

- Building products is a higher priority
- Testing is perceived as a cost center
- Testing takes too long or is too difficult
- Tests become outdated
- Ineffective Testing Practices
 - We did unit tests but the system failed anyway

Testing Misconceptions for Programmers

- Testing isn't programming
- Testers aren't as respected as software developers
- Testing isn't fun
- Testing takes too long
- Testing is done by the test group

Testing Realities for Programmers

- More freedom to design, architect, and experiment in testing
- Having a testing perspective makes you a better programmer
- Automated testing is a lot of fun
- Automation and testing strategies improves testing time
- All developers should be testers and they should share their test code with other organizations

Testing Misconceptions from Management

- I can't quantify the value of testing
- Testing slows down product releases
- Programmers should get it right the first time.
Do we have the right people?

Testing Realities for Management

- Testing reduces the overall cost of software development
- Testing reduces risk; the sooner you test the better able you are to manage risk
- Your programmers will be better able to add features if they are confident they can make changes
- Software defects are a reality. Testing can reduce the number of bugs found in the future.

Testing Definitions Reviewed

- Unit Test: contract to test a single unit of code
- Integration Test: combines software modules or units into an effective test groups
- System Test: combines components into a complete system to verify a specified requirement. This could also be called functional testing or application testing

Testing Definitions Continued

- Load Testing: Determines if system can support a specified load (e.g. 500 concurrent users)
- Stress Testing: Load testing over an extended period of time
- Capacity Testing: maximum load a system can sustain

Testing Definitions Continued

- Mock Objects: an object used to simulate the real world object. (e.g. HttpSession object or an Bluetooth scanner)
- White Box (or glass box) Testing: testing is done with a knowledge of the code. Programmer knows internal implementation and verified tests results.
- Black Box Testing: also functional testing, tester does not know internal implementation and verifies results returned

Testing Approaches

- Priority Base Testing
- Unit Testing versus Integration Testing
- System Testing
- What are the metrics for testing? How many unit, integration, and system tests?
- Kent Beck (author of Junit) recommends developers spend 25-50% of their time developing tests

Unit Testing

- Bottom up approach to testing
- Benefits: facilitates change/refactoring, simplifies integration testing, good method for documenting code
- Limitations: won't catch integration, performance, or other system related defects

Unit Testing Tools

- JUnit by Erich Gamma and Kent Beck (www.junit.org) is the standard
- Kent Beck, recommends 1 unit test per class
- There are lots of extensions for other languages
- Good example of effective use of software patterns

JUnit Overview

- Junit uses a Composite pattern
- Tests implement a TestCase class
- These tests can be combined into suites of tests
- Tests are then executed using TestRunner
 - Text mode or UI mode
- Entire suite of tests can be executed quickly and efficiently

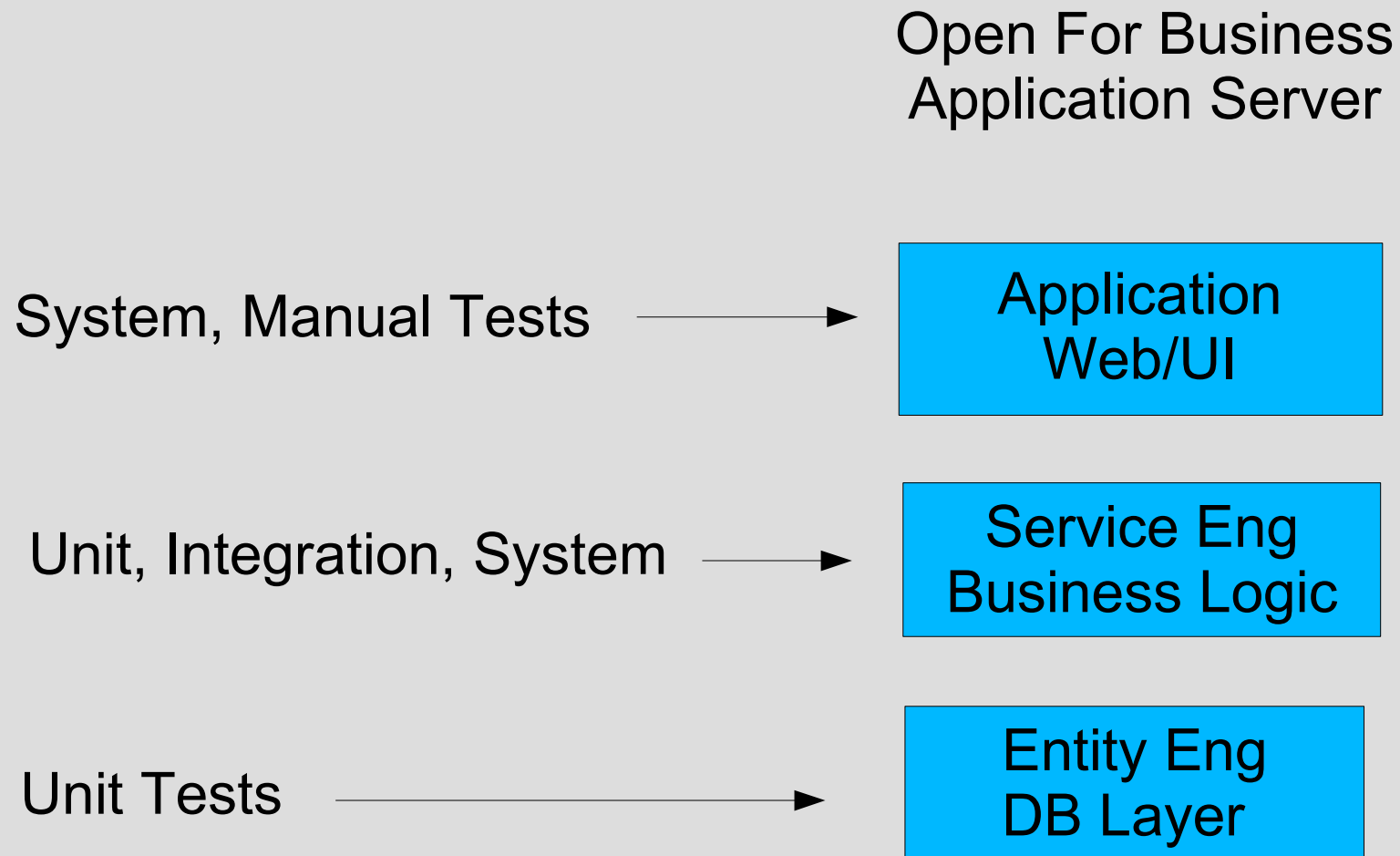
Integration Testing

- Combines unit tests into effective test group
- In distributed systems the difference between a unit test and an integration test is gray
 - E.g. A login module may include several units of code
- Distributed systems require better test planning
- Challenges: data and code dependencies

System Testing

- Combine tests to verify functionality of a complete system
- Problems:
 - data dependencies
 - Setup times become problematic
- Solutions:
 - Try to leverage existing tests and utilities where possible
 - Create setup scripts to reduce setup times
 - Leverage database utilities

Testing Strategy Example



Introduction to Grinder 3.0

- Java load-testing framework
(<http://grinder.sourceforge.net>)
- Originally for the Professional Java 2 Enterprise Edition with BEA WebLogic Server by Paco Gómez and Peter Zadrozny.
- Philip Aston took ownership of the code and reworked it to create The Grinder 2 and 3.

Grinder Capabilities

- System Testing
- Load Testing
- Capacity Testing
- Stress Testing

Grinder 3.0 Features

- Newest features is Jython support for test scripts
 - Jython is a Java port of Python.
 - Scripting allows for faster test creation and overall execution time
 - Jython can easily call any Java object
- Tests are organized by project via directories
- Excellent tool for development and testing organizations.
- Distributed Test tool: Test can be executed on multiple machines from a single console

Grinder Setup

- Tests can be organized by project via directories
 - For example:
 - website_project
 - db_project
 - Project_1
 - project_2
- Grinder.properties file sets parameters for a given tests
 - grinder.processes: number of worker agents
 - grinder.threads: number threads each worker should start
 - grinder.runs: number of iterations each thread runs

Grinder Setup Continued

- You can use Grinder to test different levels of your system
- The data can then be used to drive other tests
- Leverage existing Unit tests

Grinder Console

- Controls tests run on multiple machines
- Allows you to start, stop, and suspend tests from a single workstation
- Collects data from each test
- Simple reporting within console
- Allows for export into CSV format

Grinder Console

The Grinder Console

File Action Distribute Help

Sample interval: 1000 ms

Ignore 0 samples

Collect samples forever

Waiting for samples

0.00 TPS

Total

- ms (mean)
- 0.00 TPS (mean)
- 0.00 TPS (peak)
- 0 tests
- 0 errors

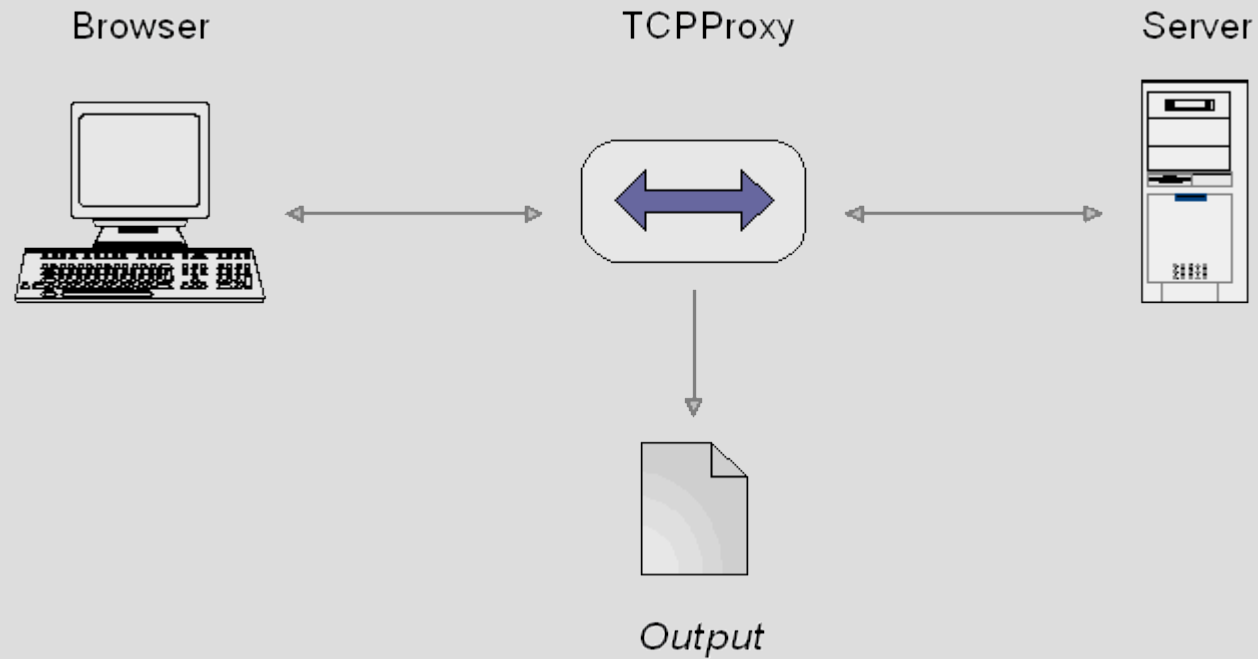
Graphs Results Processes Script

GG3

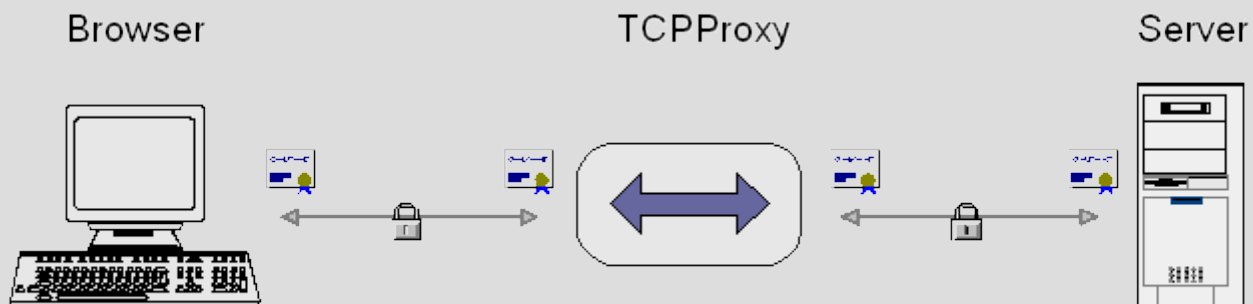
Grinder TCP Proxy

- Capture TCP/HTTP requests to a server and stubs out test code
 - Code can then be edited to create effective system test
- Supports SSL and HTTPS
- Support port forwarding

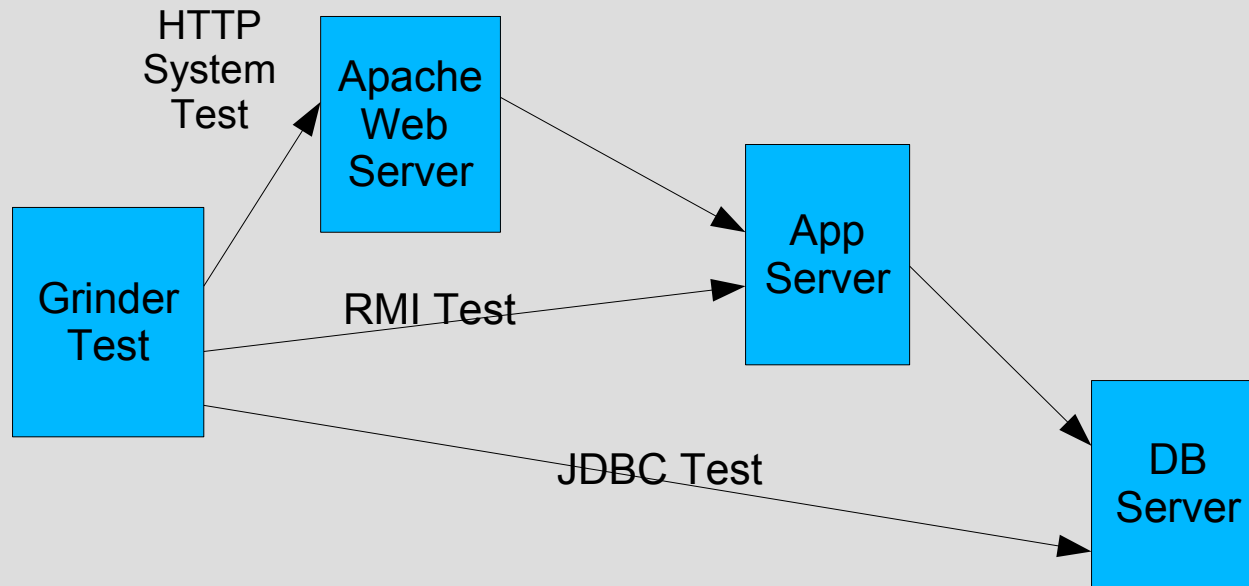
Grinder TCP Proxy



Grinder TCP Proxy with HTTPS



Sample Grinder Test Scenario



How to create a testing culture?

- Maintain a library of test suites and run them often
- Keep your tests simple and organized. If your developers can't easily use the test they won't be used
- Create test policies (e.g. All bugs must have a unit test to verify fix.)
- Use graphical reports to communicate test results
 - System tests are effective showing performance problems, etc.
 - Unit tests showing number of bugs fixed and found over time

Conclusion

- Testing is very worth while and will save your organization time and money.
- It is important to educate your organization on these savings through effective reports
- Practicing proper testing strategies will make you a better developer, architect, and manager

Q & A