

Mark Rotteveel (Pluton IT)

- Mark Rotteveel
- Born in 1979
- Hobbies: reading, working with Firebird and more
- In IT since 1999
- Working as a tester since 2006
- Joined Pluton in 2008
- Loadtesting CRM systems at T-Systems / T-Mobile NL



- Introduction testing
- Why : reasons for testing
 - Verifying performance requirements
 - Resource planning
- What : targets for testing
 - Designing tests
 - Metrics to use
- How : methods of testing
 - Tools
 - Recording / writing tests
 - Execution
 - Results

- Finding errors
 - Verifying software matches requirements
 - Validating software meets user needs
 - Other errors
- Mitigating (business) risks
- Minimizing cost

- Verifying performance requirements
- Checking behavior under load
- Resource planning
- Representative load (eg normal, peak, max transaction rate with 'normal' resource use)

- Finding limits of software (or hardware)
- Behavior beyond expected load or with limited resources
 - Bugs under high load / constrained resources
 - Error handling / graceful degradation or failure
- Exaggerated load (eg 2x peak, until application breaks)

Why : Reasons for testing

- Requirements state objectives for the application (and the test)
 - Response time
 - Load to handle
 - Etc
- Test to verify application meets those requirements!

Target metrics (example)

- Response time
- Transactions per second
- Concurrent users

Source

- List of non-functional requirements
- Usage model
 - Operational profile
 - Load profile

- Scenario : login
 - Process 5 logins per minute
 - 500 ms response time target

- Resource planning: knowing when to upgrade your hardware or add new servers
- Test to check or enhance your planning for upgrades,
- or to make a contingency plan for unexpected increase of traffic

What

- Current usage:
required resources
- Expected growth
- Determine triggers /
warnings on
resource usage

Why

- Plan ahead
- Contingency plan for
unexpected growth
- Resource impact of
new version

What : Targets of testing

- Select requirements to test
- Select (or write) use cases
- Determine user count or load target
- Amount of 'thinktime'

- Order tests by business risk or importance
- Most important tests first!
- Test individual and combined
- Aggregate test cases
- Realistic load for combined scenario

- Login
- Target : 5 logins per minute
- Determine response time (should be < 500 msec)
- Access main page
- Enter login information
- Click login

Metrics are measurements or performance indicators used to monitor the test, and determine success or failure of the test.

- Select metrics for the requirements
- Too much is better than too little
- But, more measurements influence test (eg additional load)

- Resource planning: memory, CPU, bandwidth
- User experience: response time, latency, throughput
- Correctness: % failed tests

How : Methods for testing

- Tools : open source or proprietary
- Recording / writing actual test
- Preparation
- Execution
- Reporting on results

- Open source tools: Jmeter, The Grinder (and more)
- Proprietary tools: HP LoadRunner, SilkPerformer (and more)
- Select the tool that matches your budget and needs

Open source

- Cheap
- Extendable (not always easy)

Proprietary

- Support, documentation
- More functionality

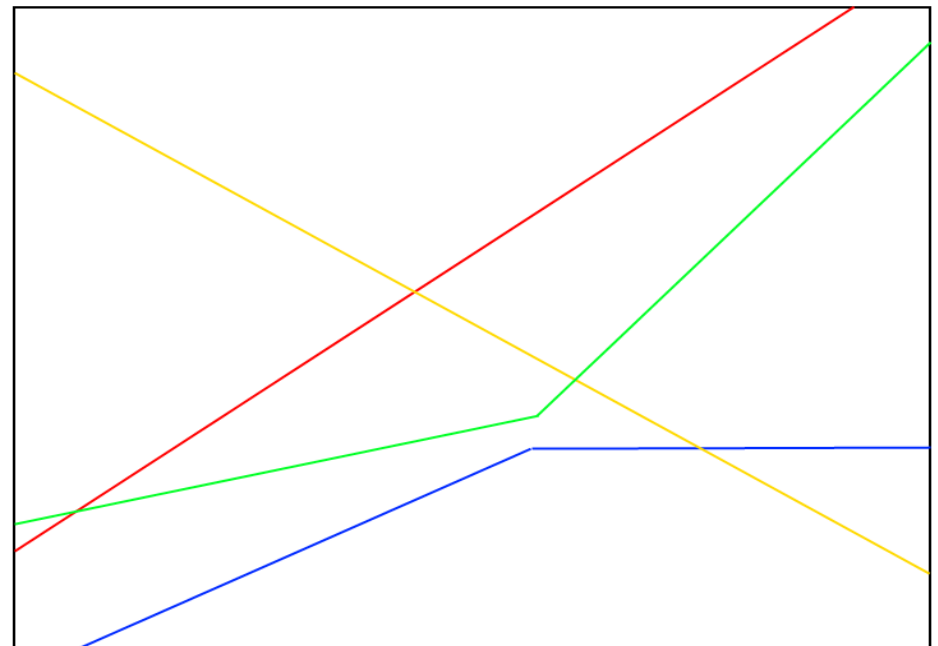
- Translate test design to executable test
- Record and playback vs 'handwritten'
- Parametrize user data
 - Scenario 'consumables' (eg usernames, order information)
 - Dynamic session data (eg cookies)
- Add assertions for expected results
- Verify playback (both single and multiple user)

- Simple example: recording login into WebGUI

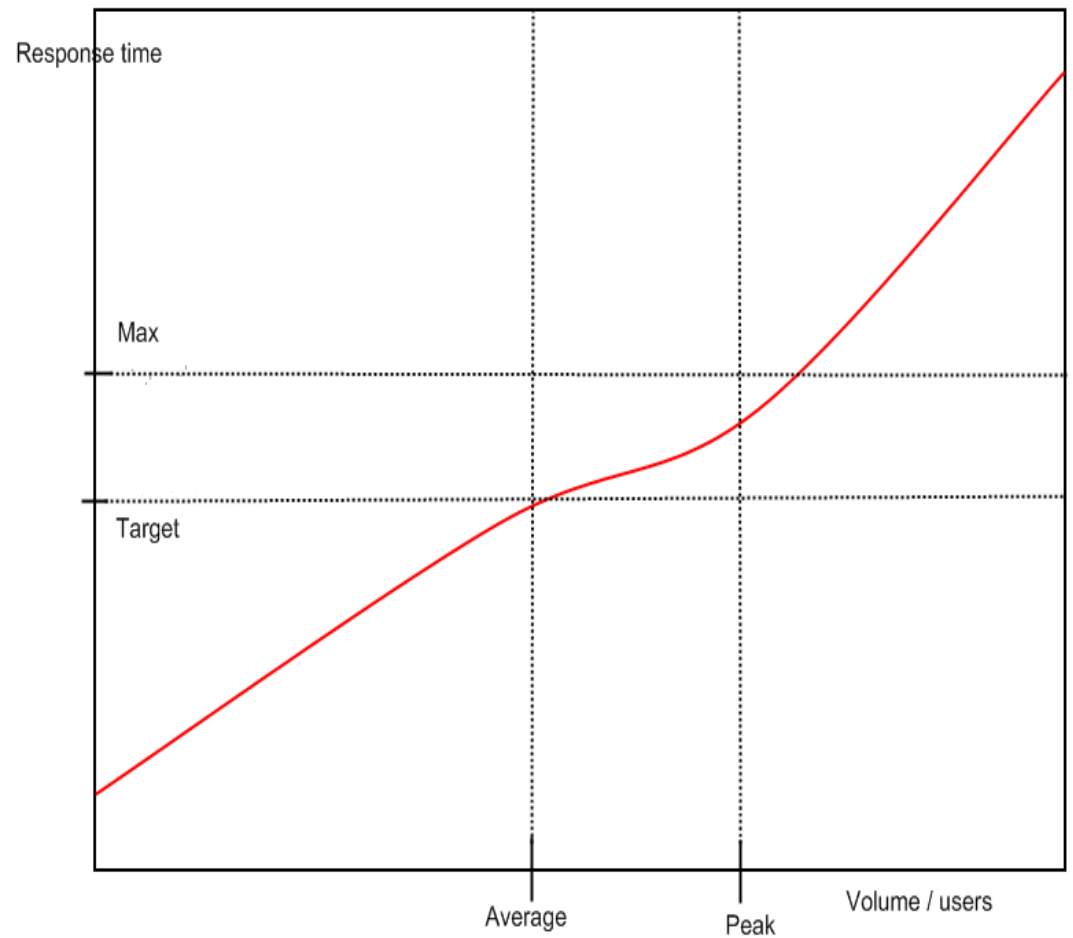
- Configure scenario (number of users, 'thinktime', ramp up, duration)
- Combine scenarios for 'realistic' load
- Setup monitoring of target application(s) / server(s)
- Collect data for scenario 'consumables'

- Start the test
- Keep an eye on metrics
- Manually check the application during test
- Verify that loadtest server(s) is not overloaded

- Correlate metrics : dependencies and bottlenecks
- Draw conclusions



- Verify requirements ('did we reach our targets')



- Communicate results and findings to development, management
- Record lessons learned for future reference
- Preserve testware for future use

- Using one loadserver: bottleneck is the testserver
- Flooding the target machine
- Not parametrizing data (caching effects, errors due to duplicate or incorrect data)

- Loadtest software is not a browser (usually no or limited javascript support!)
- Not checking for expected data (or absence of errors) in response of server
- Test environment not enough production-like (impossible to predict behavior in production)

- Manage session cookies (eg use HTTP Cookie Manager in JMeter)
- Parameter separator of WebGUI (';') not supported by all software; only standard '&' (minor problems recording, WebGUI does support '&' in playback)
- JMeter bug: recording proxy sometimes incorrectly encodes GET requests (stop and start proxy before recording next step)

Questions?