QM06

Don‘t Be The Bottleneck

Results-Driven Testing

John Straathof
john.straathof@zyntax.com
Director / Performance Analyst
Zyntax Consulting
RSDC Silver Sponsor

IBM Rational Software Development Conference 2008

WHERE TEAMS ARE R-HEROES

IBM

Rational. software

Myths and Facts

The world is flat. We are the center of the universe.

If the world *was* flat, could it actually *be* the center of the universe?

Testing is a *‘tickbox’* before an application goes into production.

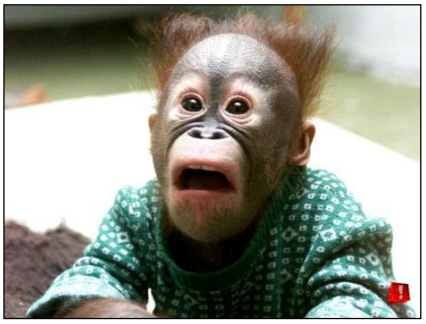
I have:

Test plan? √

Execution plan? √

Success criteria? √

Expect issues and *plan* for them?



Well, what if there *weren't* any issues?

Known cases when no issues were found.[1]

- History from 1989-1993

None

- History from 1994-1998

None (there was a fable, later proved untrue)

- History from 1999-2003

None

- History from 2004-2008 [2]

Holding out hope

[1] These cases can be found on Wikipedia ®

[2] through 22 Sep 2008

Process of Results-driven Testing

- Find issues early
  - Until first issue is found, **you (testing) *are* the bottleneck**
  - Find an issue? **Issue Resolver becomes bottleneck**
  - Testing carries on
- Accept application quality is what it is, now.

Key Points

- Issue resolution is *not* a problem, it's an improvement
- Some issues sequential
- Issues found early allow more sequential improvement

Typical performance test

Test Scripts (Test Cases) -> Results

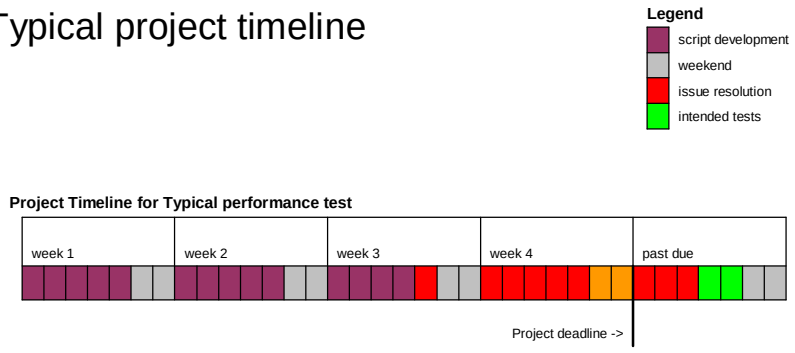
Example Planning

1. Project deadline at 20 days
2. Develop 6 scripts (estimate 12 days)
3. Execute workloads of 25, 50, 100, 500 users (max 8 days)
4. Perform trend analysis
5. Learn what you can

Where?

- Resolving unplanned issues (10 days)

Typical project timeline



Results-driven performance test

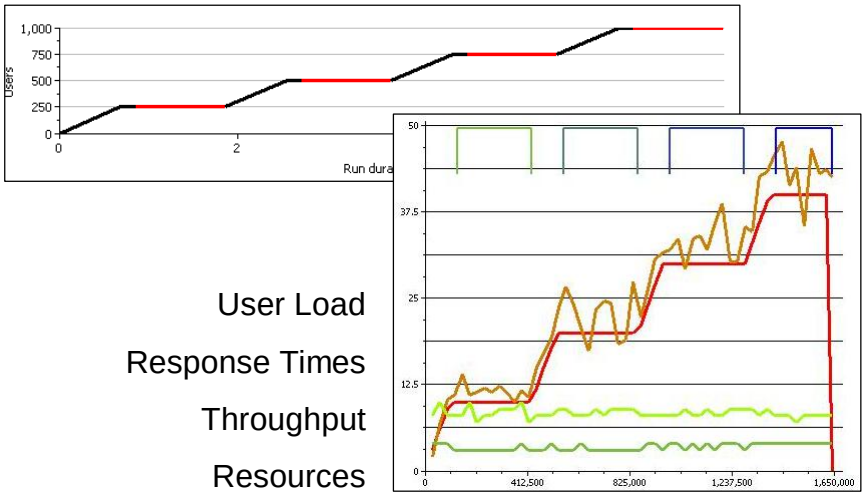
Typical	Test Scripts → Results
Results-driven	Results → Test Scripts

1. Decide which risks we *need* to cover
2. Risks determine Trends
3. Trends determine Tests
4. Tests need test scripts

How many test scripts?

To start, what *are* your risks?

Risk if Scalable



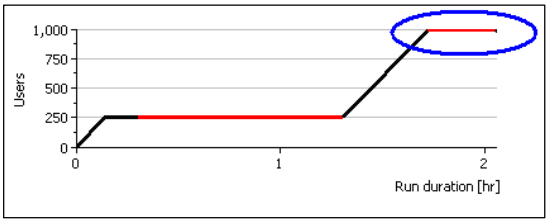
User Load

Response Times

Throughput

Resources

Risk from Peak Load



NEWS FLASH

The London Stock Exchange's new order-driven trading system was not to blame for a computer crash that led to a delay to the start of trading

9 September 2008 ?

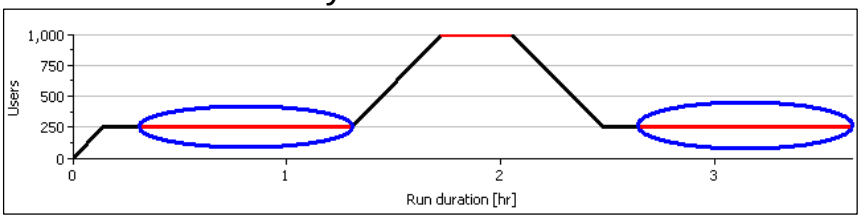
8 November, 2005 ?

10 July 2000 ?

5 November, 1997?

Peak Load.. A serious risk.

Risk of no Recovery



Compare *before* and *after* peak

- Response Times
- Resource Usage

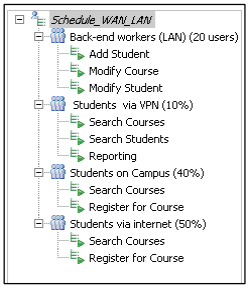
Example of sequential issue; must first handle Peak Load

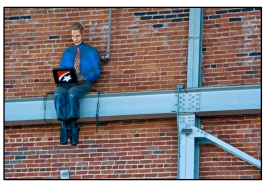
Risking failure through Contention

Many contention issues can be identified simply

with just *1 use case* and just *2 users*

Risk: Supporting User Mobility





Covering Other Risks - Examples

- Batch Processes (with and without)
- Database Volumes (now, in 1, 2, ..., 5 years)
- Database Type (DB2, MySQL, Oracle)
- Operating System (Windows, AIX, Linux, ...)
- Hardware Configuration (pSeries, Blades, zSeries, ...)
- Stress Test
- Regression
- ...

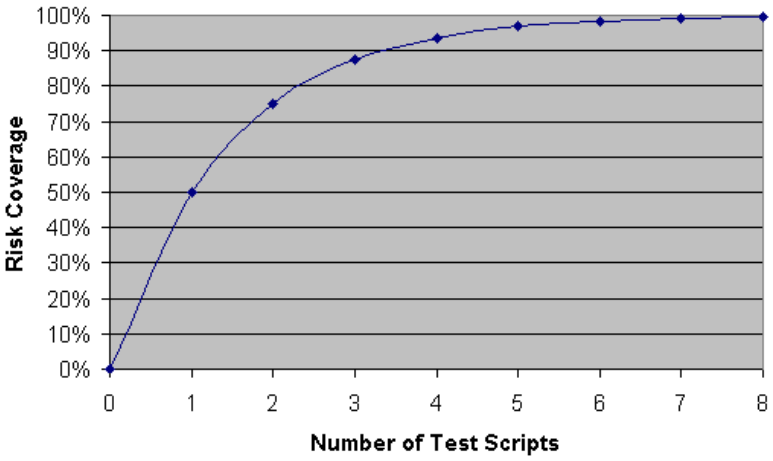
Results-driven performance test (review)

Decide

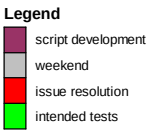
1. Which risks
2. Which trends
3. Which tests
4. Minimum number of test scripts

More test scripts -> more risk coverage?

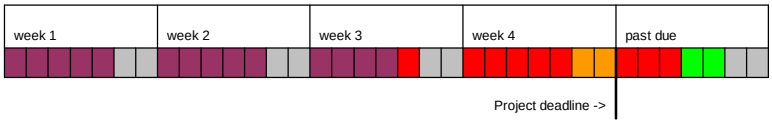
Risk Coverage vs Number of Test Scripts



Typical vs. Results-driven



Project Timeline for Typical performance test



Results-driven Process Summary

- Decide what you need, what the project risks are
- Work back to trend analysis
- Minimise test scripting (it‘s a cost)

- Accept there *will* be issues
  - Plan for them
  - See them as Improvements
  - Make every execution *useful*

- When is the Performance Test Complete?
  - When you‘re the bottleneck, time to ask

Conclusion

- Focus on Results
- Find issues early
- Improve quality

- And the basic rule

Don‘t Be The Bottleneck

If not for sensible reasons, it hurts... just ask *this* bottleneck

Don‘t Be The Bottleneck

Results-driven testing

Thank You