CMG 2008

Agile Performance Testing

Alexander Podelko

apodelko@yahoo.com

Agenda

- Existing Approaches
- Testing Early
- Workload Generation
- Tuning and Troubleshooting
- Building a Model
- Making Performance Testing Agile

Performance Testing

- An important project step in many corporations
- A project itself
  - We can apply software development methodologies to this project
- Waterfall approach in most cases
- Agile approach

Waterfall Approach

- Flowing steadily downwards through the phases
  - Get the system ready
  - Develop scripts requested
  - Run scripts in the requested combinations
  - Compare with the provided requirements
  - If requirements missed, throw back to development

Main Problems

- The system must be ready
  - Late changes are expensive
  - If we want earlier, should be more agile / explorative
- Test scripts are also software
  - Record/playback may give a false impression that script creation is easy
  - You still need to correlate, parameterize, debug, and verify

Main Problems

- Running all scripts together makes it very difficult to tune and troubleshoot
  - Shot-in-the-dark anti-pattern
  - Tuning and troubleshooting are iterative
- Minimal information about the system behavior
  - You cannot build any kind of model

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- ***Individuals and interactions over processes and tools***
- ***Working software over comprehensive documentation***
- ***Customer collaboration over contract negotiation***
- ***Responding to change over following a plan***

That is, while there is value in the items on the right, we value the items on the left more.

Agenda

- Existing Approaches
- Testing Early
- Workload Generation
- Tuning and Troubleshooting
- Building a Model
- Making Performance Testing Agile

Testing Early

- Nobody argues against
  - Software Performance Engineering has long been advocated
- Rarely happens in practice
  - Dr. Gunther in "Guerrilla Capacity Planning" shows that underlying general resistance from management is a set of unspoken assumptions

Some Unspoken Assumptions

- Schedule is the main measure of success
- Product production is more important than product performance
- We build product first and then tune performance
- Hardware is not expensive, we can just add more hardware if necessary
- There are plenty of commercial tools that can do it

Guerrilla Approach

- Dr. Gunther suggests Guerrilla Approach
  - Tactical planning
  - Opportunistic intervention
  - Rapid Analysis
  - Method vs. Madness
- Do whatever you could with early performance engineering
  - Don't wait until everything gets in place

Unit Performance Testing

- Any part of the system
- Not a standard practice
- Do not wait the system is assembled
- Test cases are simpler, fewer variables
- Many systems are monolithic
  - Test-Driven Development may be an answer
- Third-party components

Mentality Change

- Late record/playback performance testing -> Early Performance Engineering
- System-level requirements -> Component-level requirements
- Record/playback approach -> Programming to generate load/create stubs
- "Black Box" -> "Grey Box"

Single-User Performance

- If performance for one user isn't good, it won't be any better for multiple users
- Single-user testing is conducted throughout the development life cycle
  - Gathering performance data can be extremely helpful
  - Can provide a good indication of what business functions need to be investigated further

Early Involvement

- Any early involvement would be beneficial
  - Even if only asking a few key questions
  - Don't wait until everything gets in place
  - A few guerrilla-style actions can save a lot of time and resources later
- Unfortunately, you often get involved in the project at a later stage
  - Next sections are still fully applicable

Agenda

- Existing Approaches
- Testing Early
- Workload Generation
- Tuning and Troubleshooting
- Building a Model
- Making Performance Testing Agile

Don't Underestimate Workload Generation

Only the Paranoid Survive

- The title of Andy Grove's book should be performance engineering's motto

Be a Performance Test Architect

- Gather all requirements and project them onto the system architecture
  - Business requirements are not the "Holy Scripture"
  - Iterative process, evaluate and validate
  - Scrutinize workload
  - Project requirements onto the system architecture

    What components are involved

Be a Performance Test Architect

- Make sure that the system is properly configured and results are meaningful
  - Difference between environments
  - Data used
  - User access
- Make the test environment as close to the production as possible
- Performance testing isn't an exact science

Scripting Process

- Record/playback is easy for static content and plain HTML
  - Not many such systems anymore
  - No guarantee that it would be easy
- Software Development Project
  - Correlate and parameterize
  - Validate
    - Lack of error messages is not the proof
    - Test different input data

Not so Simple: Example I

- **Back-end calculation (Financial consolidation)**
  - **Long time, shows progress bar**
  - **Polling back-end**
  - **Explicit loop is needed to work properly**

Recorded Script

```
web_custom_request("XMLDataGrid.asp_7","URL={URL}/
    Data/XMLDataGrid.asp?Action=EXECUTE&TaskID=1024
    &RowStart=1&ColStart=2&RowEnd=1&ColEnd=2&SelTy
    pe=0&Format=JavaScript", LAST);
web_custom_request("XMLDataGrid.asp_8","URL={URL}/
    Data/XMLDataGrid.asp?Action=GETCONSOLSTATUS",
    LAST);
web_custom_request("XMLDataGrid.asp_9","URL={URL}/
    Data/XMLDataGrid.asp?Action=GETCONSOLSTATUS",
    LAST);
web_custom_request("XMLDataGrid.asp_9","URL={URL}/
    Data/XMLDataGrid.asp?Action=GETCONSOLSTATUS",
    LAST);
```

Working Script

```
web_custom_request("XMLDataGrid.asp_7","URL={URL}/
   Data/XMLDataGrid.asp?Action=EXECUTE&TaskID=1024
   &RowStart=1&ColStart=2&RowEnd=1&ColEnd=2&SelTy
   pe=0&Format=JavaScript", LAST);
do {
   sleep(3000);
   web_reg_find("Text=1","SaveCount=abc_count",LAST);
   web_custom_request("XMLDataGrid.asp_8","URL={UR
   L}/Data/XMLDataGrid.asp?Action=GETCONSOLSTATU
   S",  LAST);
   } while (strcmp(lr_eval_string("{abc_count}"),"1")==0);
```

Not so Simple: Example II

- Web form to enter and save data
- Drop-down box with department name
  - Form Point-of-View (POV)
- Both department name and ID should be parameterized
  - ID should be queried from the repository
- No errors if ID is not parameterized
- Real validation is checking that data are saved

Working Script

```
web_submit_data("WebFormGenerated.asp","Action=http://hfmtest.us.schp.com/HFM/data/WebFormGenerated.asp?FormName=Tax+QFP",
ITEMDATA,
"Name=SubmitType", "Value=1", ENDITEM,
"Name=FormPOV", "Value=TaxQFP", ENDITEM,
"Name=FormPOV", "Value=2007", ENDITEM,
"Name=FormPOV", "Value=Periodic", ENDITEM,
"Name=FormPOV", "Value={department_name}", ENDITEM,
"Name=MODVAL_19.2007.50331648.1.{department_id}.14.409.2130706432.4.1.90.0.345", "Value=<1.7e+2>;;", ENDITEM, LAST);
```

Agenda

- Existing Approaches
- Testing Early
- Workload Generation
- Tuning and Troubleshooting
- Building a Model
- Making Performance Testing Agile

Performance Testing

- Often is not separated from:
  - Tuning

    System should be properly tuned
  - Troubleshooting / Diagnostics

    Problems should be diagnosed further to the point when it is clear how to handle them
  - Capacity Planning
- "Pure" performance testing is rare
  - Regression testing ?

Tuning and Troubleshooting

- Both are iterative processes
  - Make a change
  - Run the test
  - Analyze results
  - Repeat if necessary
- You apply the same synthetic workload
  - Easy to quantify the impact of the change

Issues

- Requires close collaboration of all stakeholders
- Impossible to predict how many test iterations would be necessary
  - Shorter / simpler tests may be needed
  - Complex tests may make problems less evident
- Usually there are no indications at the beginning of the project how much time and resources would be required

Process

- Asynchronous process doesn't work well
  - Usually used in functional testing
  - Log the defect into a tracking system, then it would be prioritized and assigned

1. Problem often blocks further testing
2. Full setup is usually needed to reproduce
3. Debugging is a collaborative process

What is Needed?

- **Synchronized work of performance engineers, developers, and other stakeholders must occur to fix problems and complete performance testing**

Agenda

- Existing Approaches
- Testing Early
- Workload Generation
- Tuning and Troubleshooting
- **Building a Model**
- Making Performance Testing Agile

Building a Model

- Significantly increases the value of performance testing
  - One more way to validate tests and help to identify problems
  - Answers questions about sizing the system
- Doesn't need to be a formal model
  - May be just simple observations as to how much resources are needed by each component for the specific workload

Modeling

- Often is associated with queuing theory
  - Great mechanism, but not required in simple cases
- Most good performance engineers have a model in their mind
  - May be not documented / formalized
  - But they see when the system behaves in an unusual way

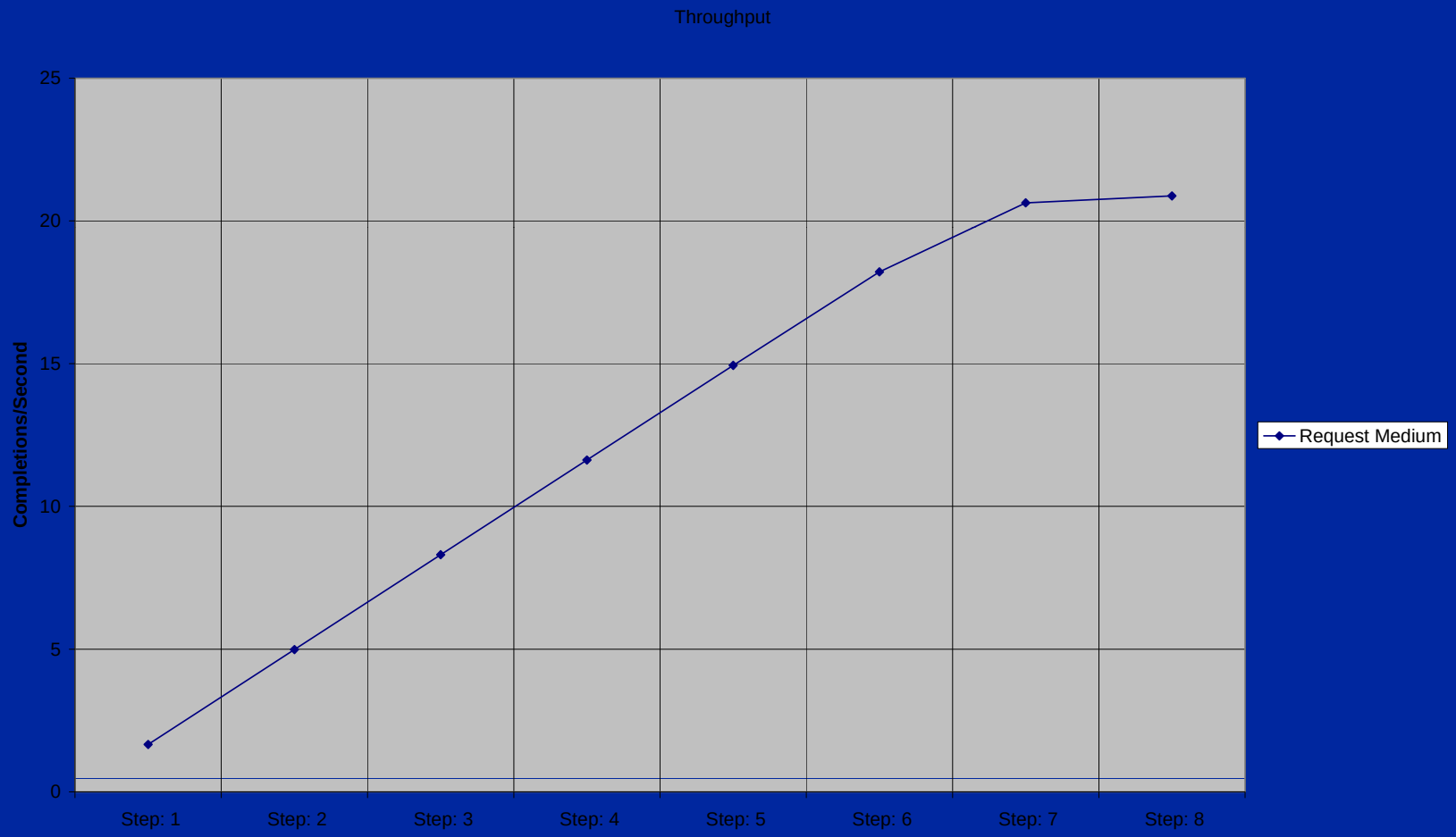
Sometimes Linear Model Works

- Linear model may often work for multi-processor machines
  - Considering the working range of CPU utilization
    - Most IT shops don't want more than 70-80%
  - Throughput and CPU utilization increase proportionally
  - Response times increase insignificantly

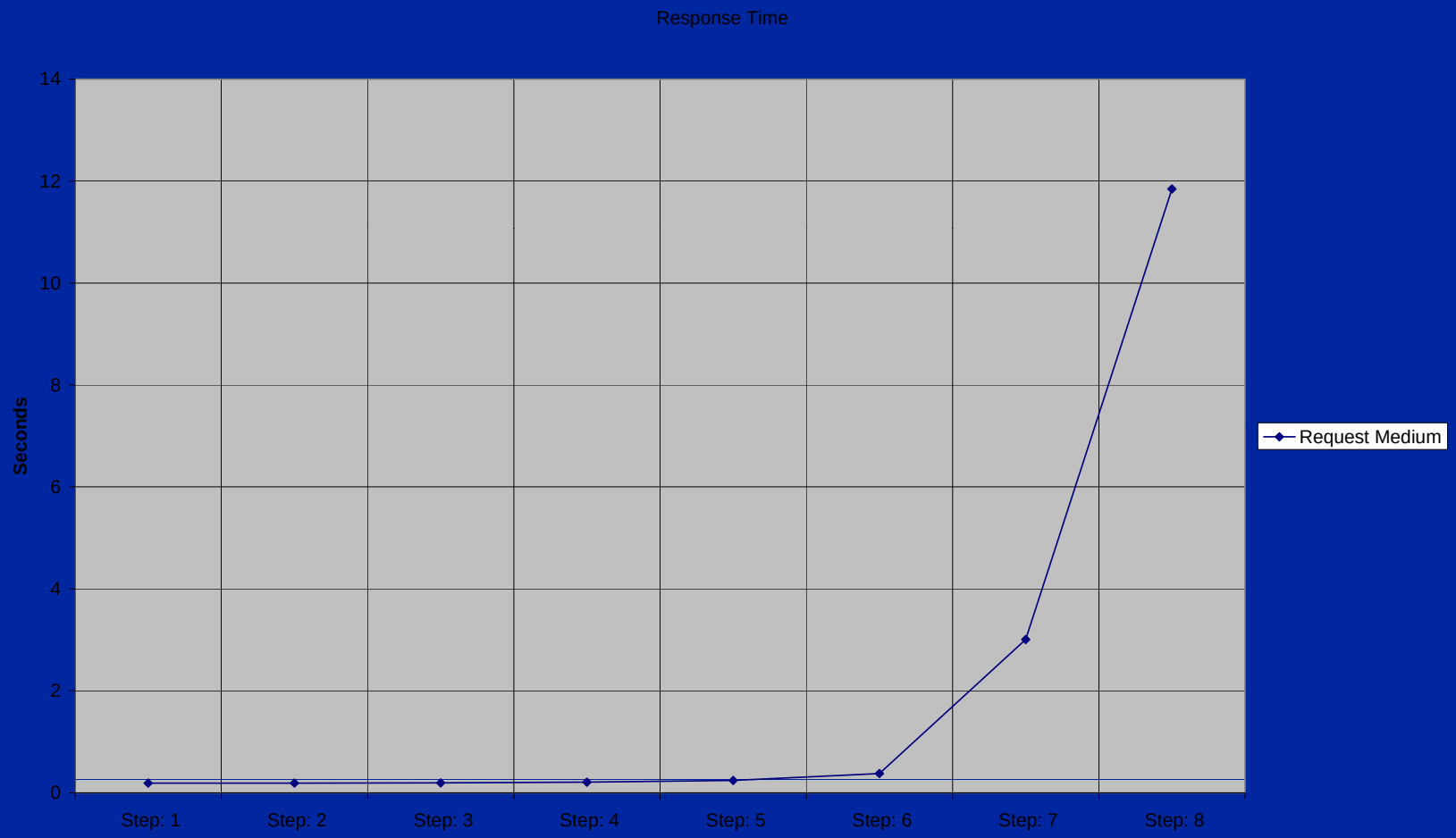
Example

- Simple queuing model was built using TeamQuest
- Specific workload executed on a four-way server
- Eight different load levels
  - Step 1 represents 100-user workload, each next step adds 200 users, step 8 represents 1,500 users
- Linear model would be good through step 6
  - With CPU utilization 87%

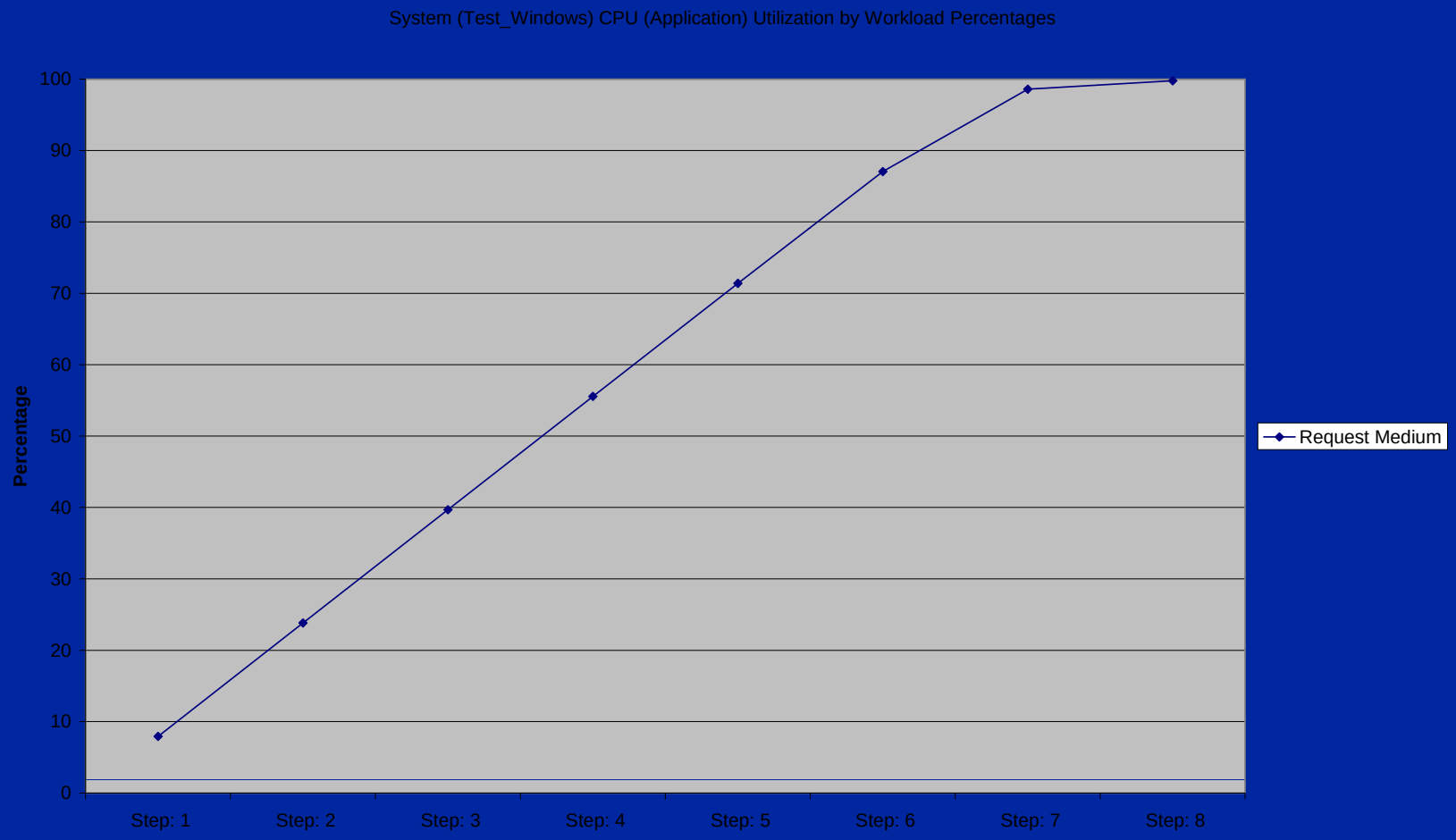
Throughput


Throughput
Completions/Second
25
20
15
10
5
0
Step: 1
Step: 2
Step: 3
Step: 4
Step: 5
Step: 6
Step: 7
Step: 8
Request Medium

Response Time



CPU Utilization



How to Approach?

- **Run independent tests for each business function**
  - **To see how much resources each function requires**
  - **Load shouldn't be too light**

    To get steady, not distorted by noise resource utilization
  - **Load shouldn't be too heavy**

    To avoid non-linear effects

Agenda

- Existing Approaches
- Testing Early
- Workload Generation
- Tuning and Troubleshooting
- Building a Model
- Making Performance Testing Agile

Agile

- 'Agile software development refers to a group of software development methodologies that promotes development iterations, open collaboration, and process adaptability through the life-cycle of the project'
- Fully applicable to performance testing
- Performance testing is agile by its nature

Agile Performance Testing

- ***Here*** **means finding new opportunities inside existing processes using collaboration, iterative and adaptive processes**
- **Performance testing in agile development is a completely different topic**
- **Most good performance engineers already use similar approaches**
  - **Although waterfall-like plan is usually presented to management**

Agile Approach

- Get involved early if possible
- Have a plan, but it needs to be very adaptable
- Expect iterative process involving tuning and troubleshooting
  - Each test provides a lot of information
  - If you find a bottleneck, you need to fix it before running further tests

Agile Approach

- As soon as you get a working script
  - Run it for one, a few, and many users
  - Sort out errors
  - Note resource utilization (build a "model")
- Even with a single script you can find many problems and, at least partially, tune the system

Agile Approach

- Make sure that every script works before running a real-life mixed scenario
  - Verify that the system behaves as expected
  - Pay attention to every issue / deviation
- Work in close cooperation with developers, administrators, and other experts

Summary

- **Small extra efforts, making the process more agile, increase efficiency significantly – and usually pay off multi-fold**
  - **Get involved early**
  - **Be paranoid with workload generation / system setup**
  - **Expect multiple tuning and troubleshooting iterations**
  - **Build a "model"**

Questions?

Alexander Podelko

apodelko@yahoo.com

Links and references may be found in the paper and at www.alexanderpodelko.com