coding {the} architecture

Load Testing for Developers

Join the conversation

#prognet11

simon.brown@codingthearchitecture.com

@simonbrown on Twitter

Hands-on Software Development

Training, Coaching & Consulting

Coding the Architecture

http://www.codingthearchitecture.com

Does your architecture “work”?

Satisfies the architectural drivers

Foundations for the code

Platform for solving the business problem

Architectural drivers?

- ☐ Performance ✓
- ☐ Scalability ✓
- ☐ Availability ✓
- ☐ Security ✓
- ☐ Disaster Recovery
- ☐ Accessibility
- ☐ Monitoring ✓
- ☐ Management
- ☐ Audit ✓
- ☐ …

Runtime

- ☐ Flexibility
- ☐ Extensibility
- ☐ Maintainability
- ☐ Interoperability ✓
- ☐ Legal
- ☐ Regulatory
- ☐ Compliance ✓
- ☐ i18n
- ☐ L10n
- ☐ …

Non-runtime

Performance is about how

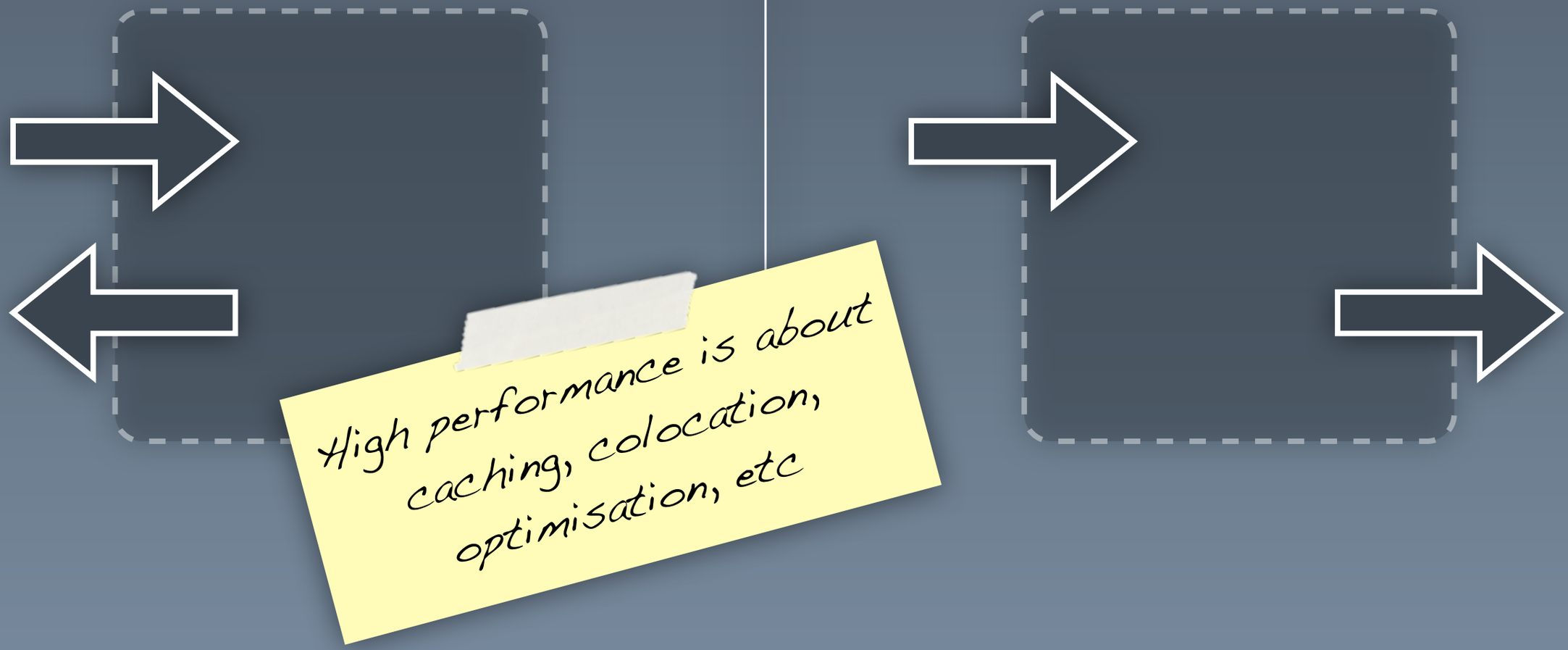
fast something is

Applicable to web apps,
UIs, messaging systems,
SOAs, etc...

Ever been told that
"the system is slow"?

Response time

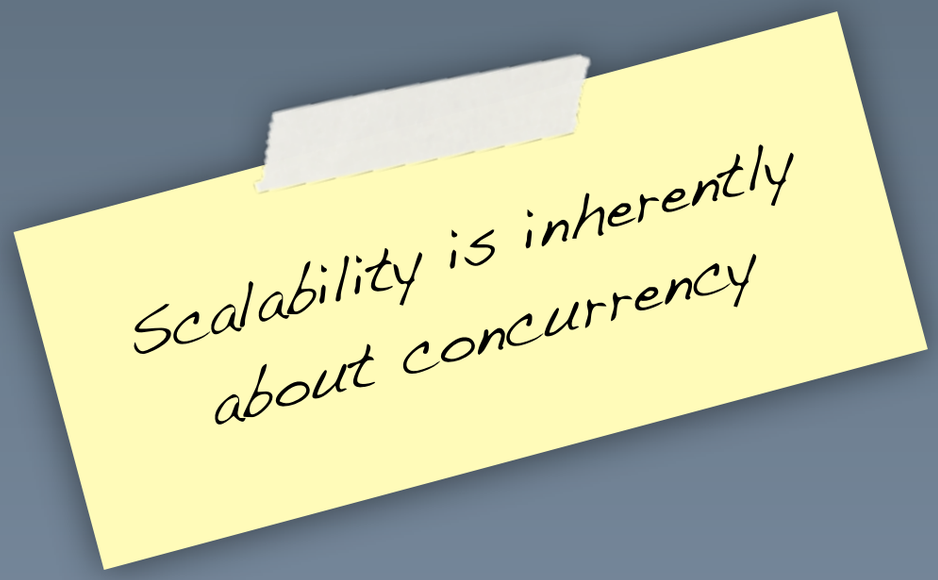
Latency and throughput



Scalability is about

doing more

(more requests, more data,
more users, more messages, ...)



Vertical

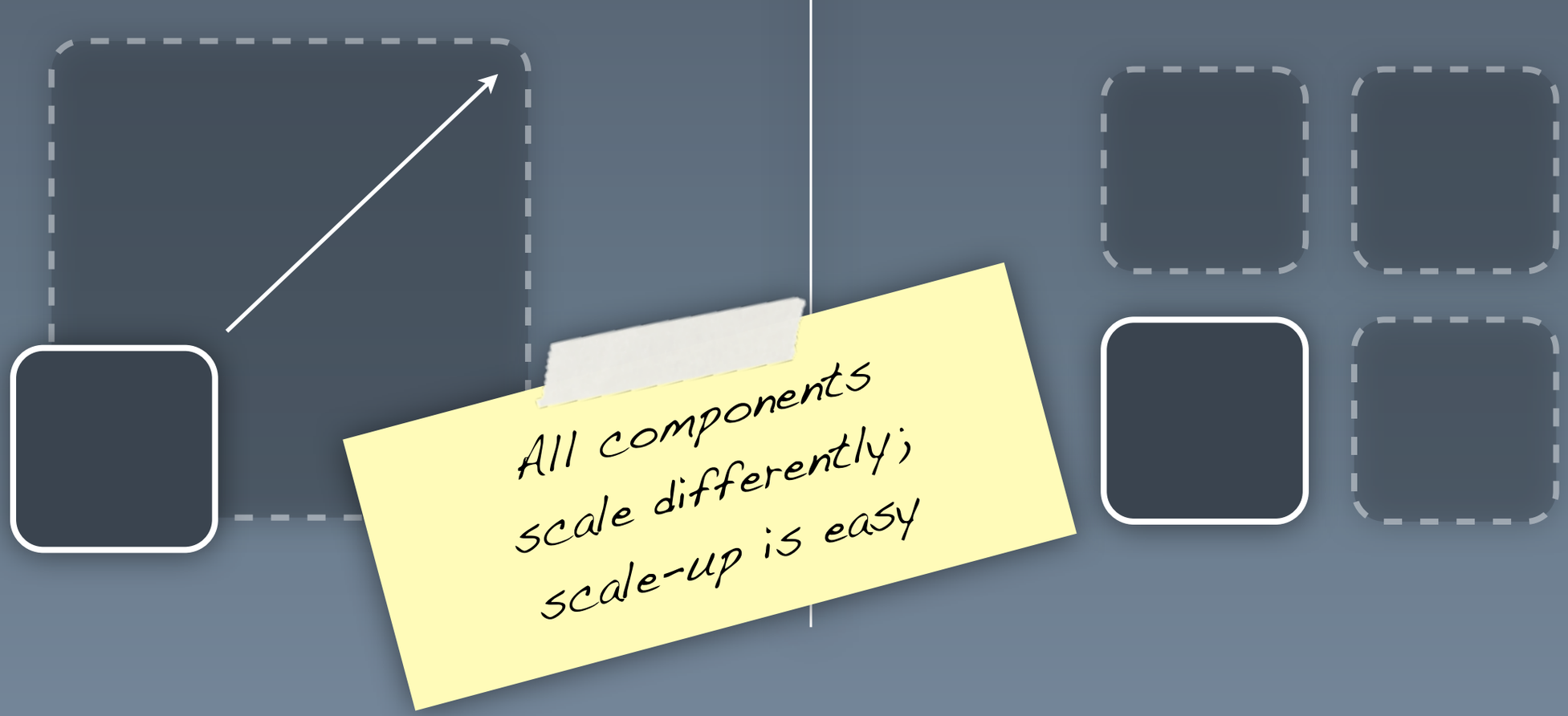
scalability

(scale-up)

Horizontal

scalability

(scale-out)



What do you want to know?

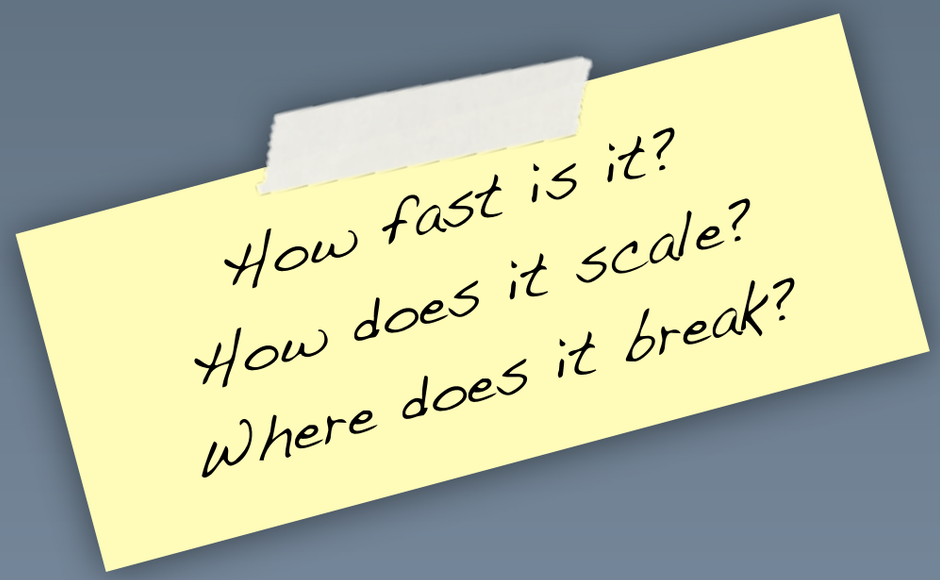
(and why?)

You should have a

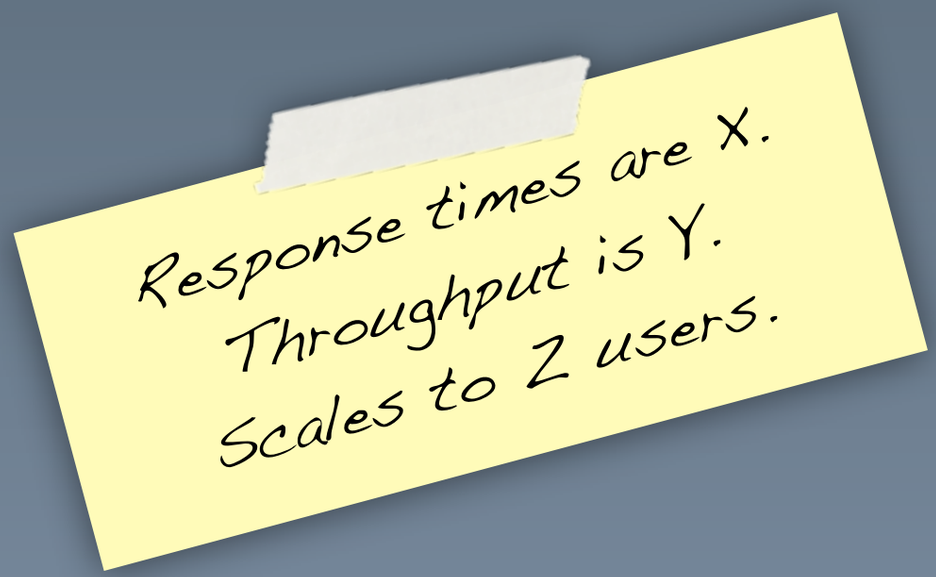
reason

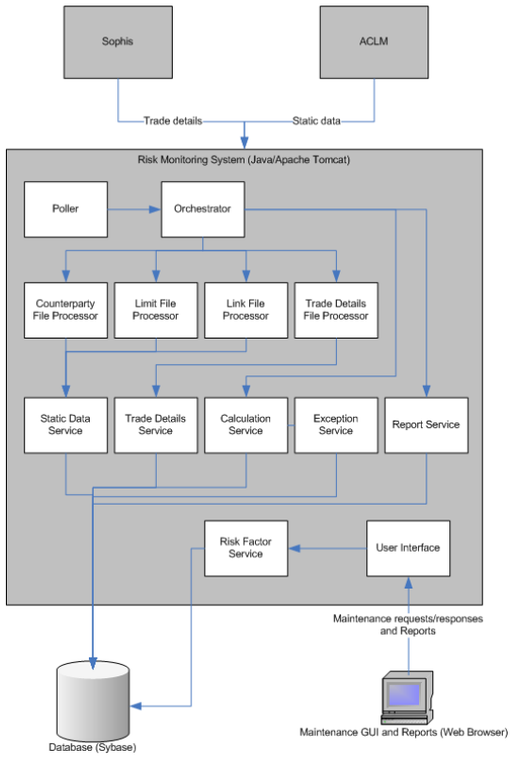
for load testing

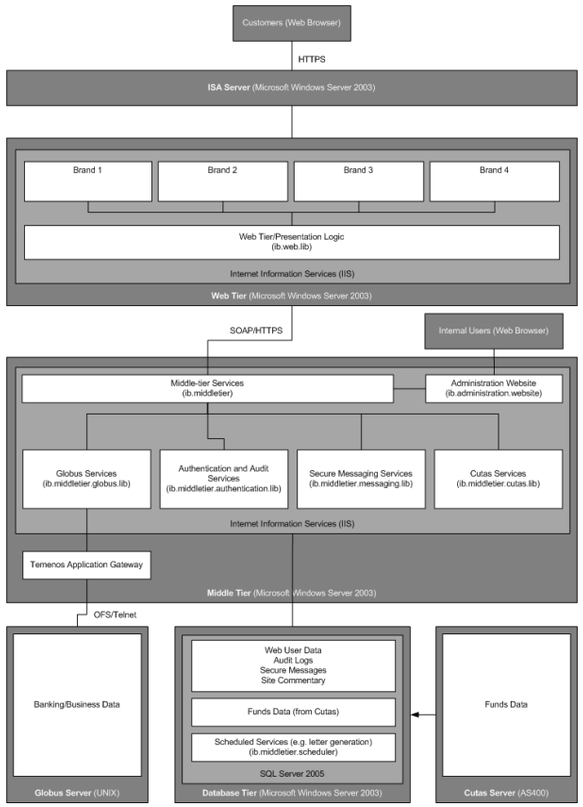
What do you want to learn about your system?

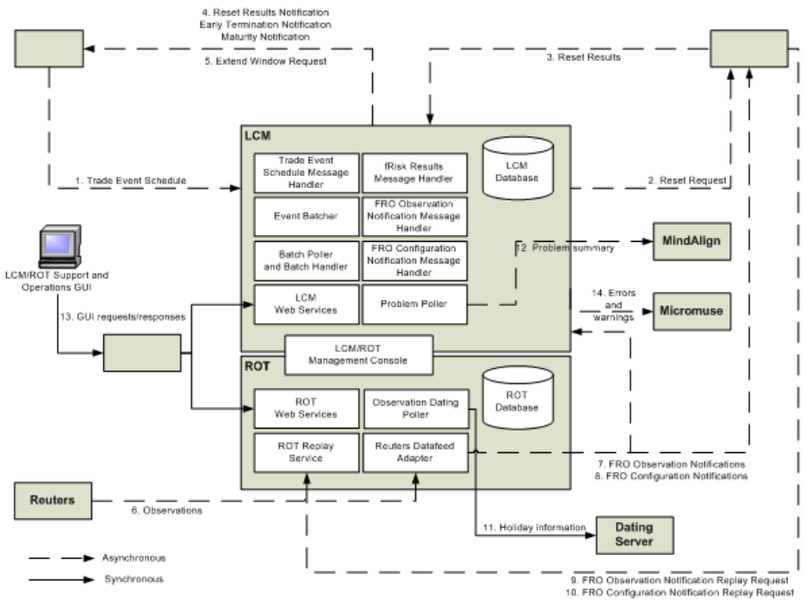


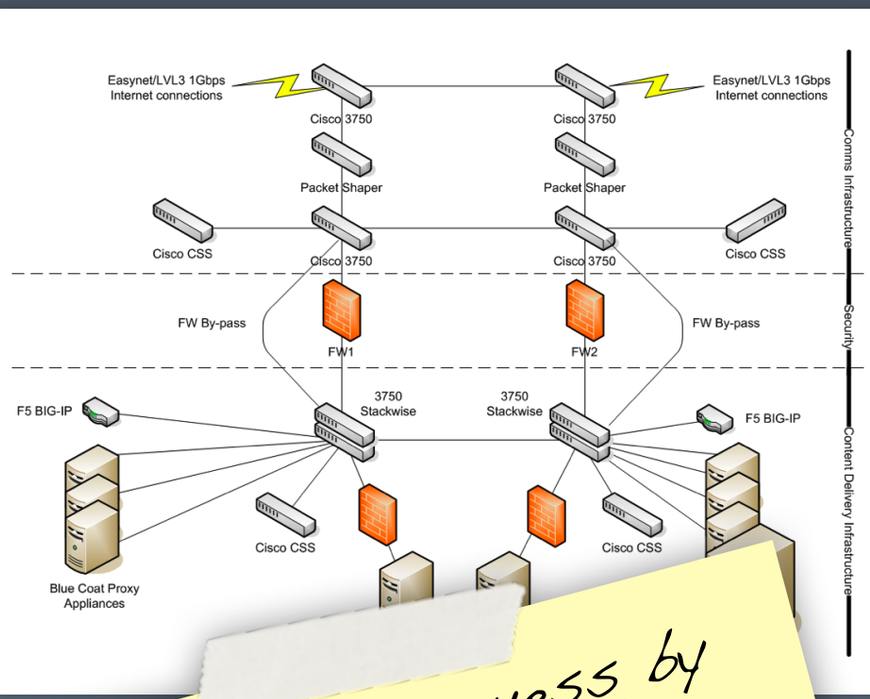
What do you want to prove about your system?

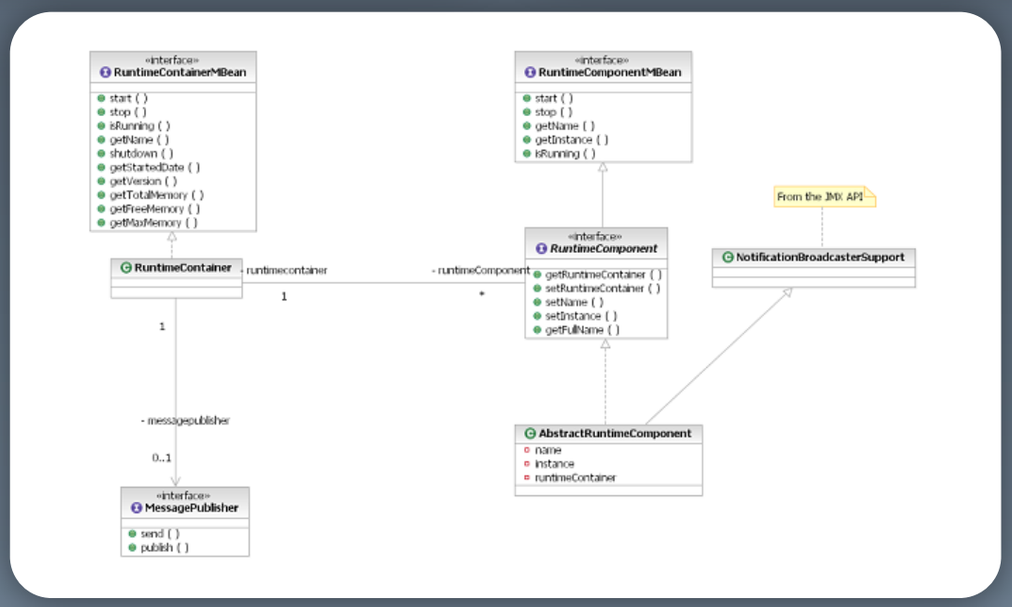












Will these software systems

perform and scale acceptably?

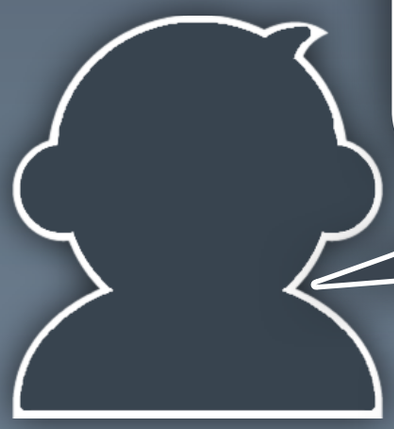
You can only guess by looking at diagrams or source code

Software development is not a relay sport

This new system will work because we have a

spreadsheet

showing performance figures from a past project



A software “architect”

What is load testing?

Testing provides

confidence

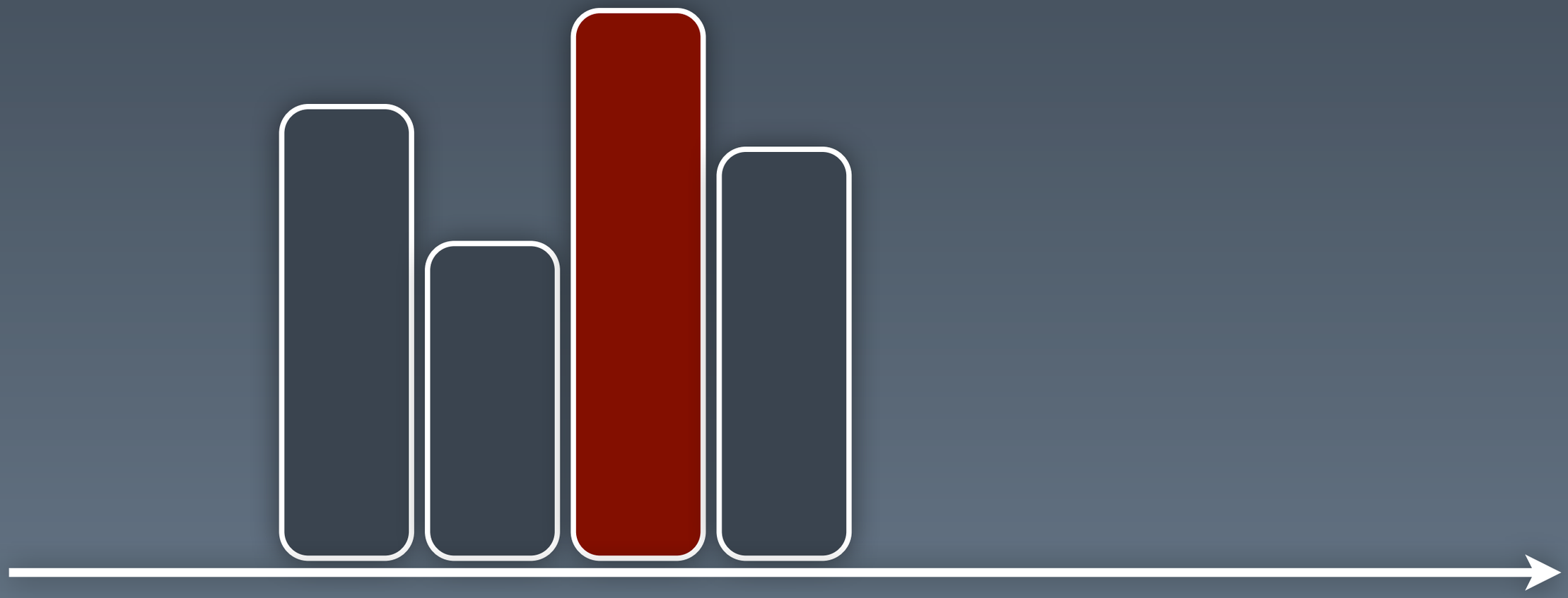
Load testing is one way to

evaluate

or # validate

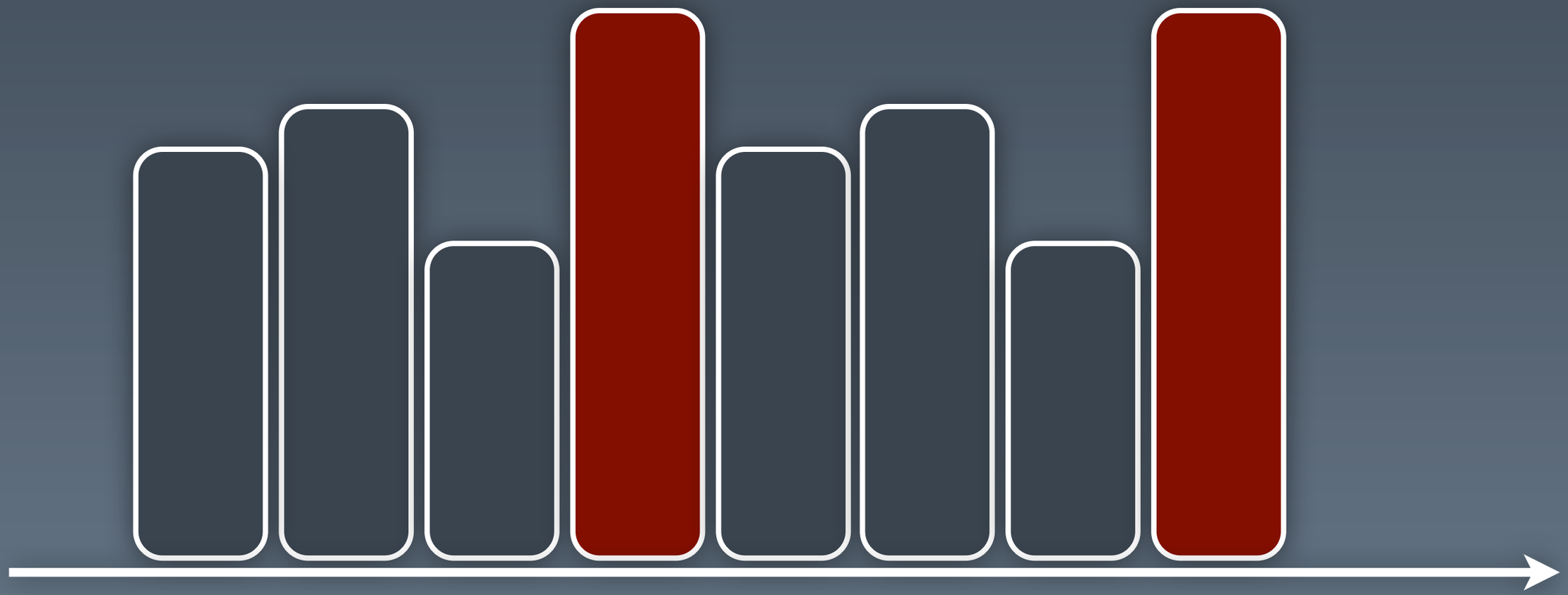
your architecture

Caveat: if performance and scalability are important characteristics

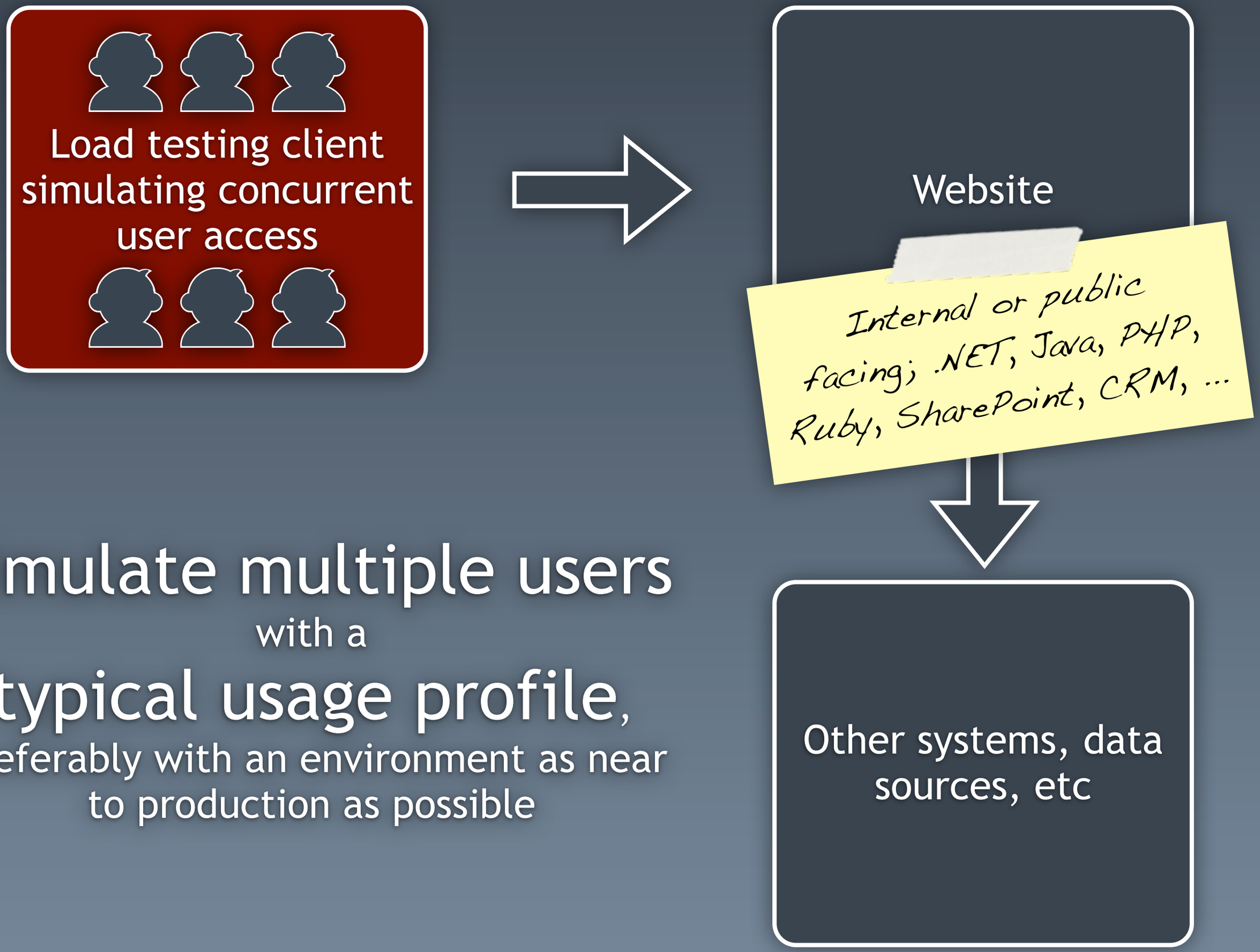


Load testing is asserting how the architecture performs under load with a view to monitoring the response times for key transactions

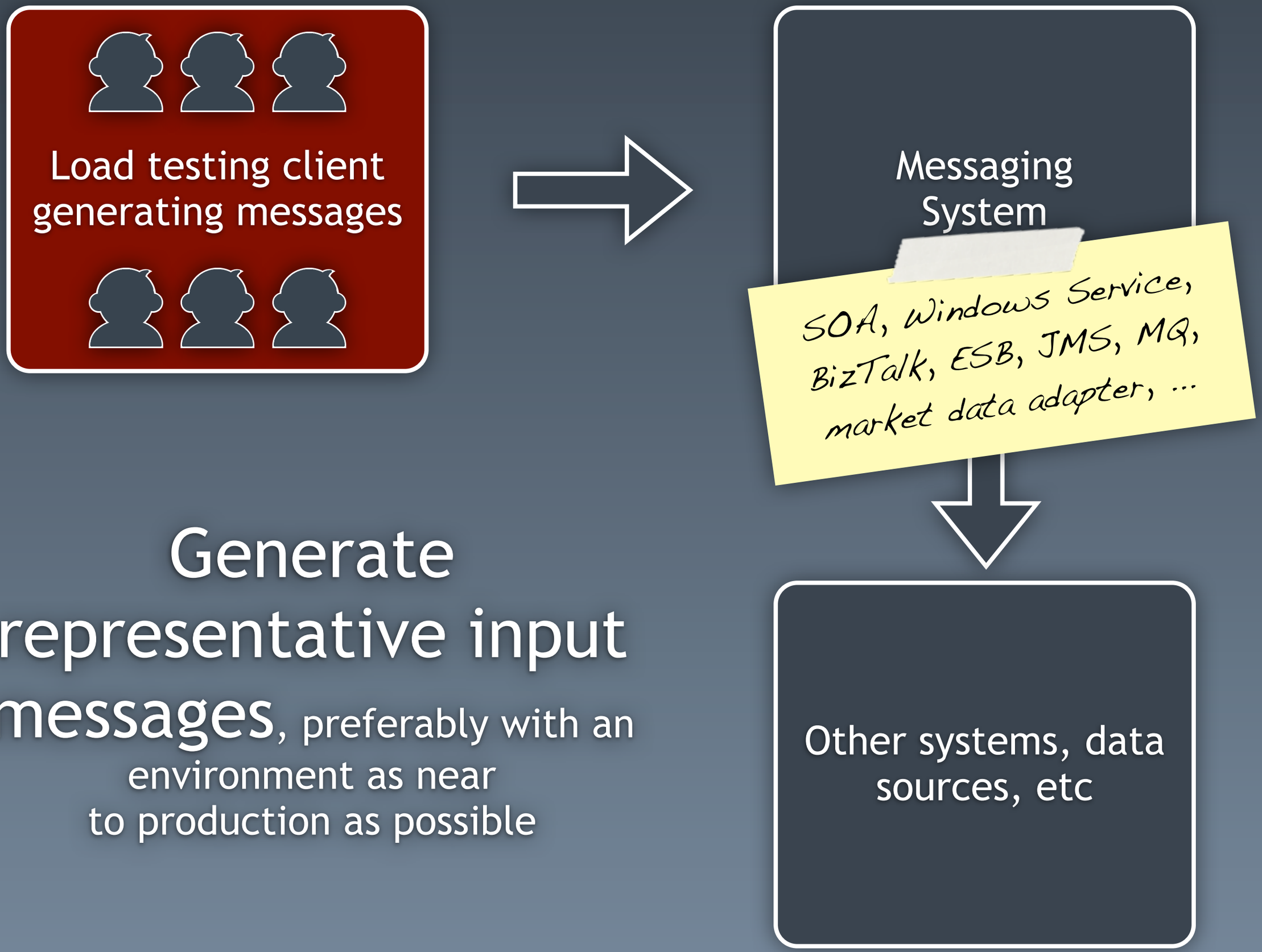
Stress testing is asserting what the upper bounds are for the scalability of the architecture, understanding how it reacts when stressed



Soak testing is asserting that the performance of the architecture remains stable over longer periods of time



Simulate multiple users
with a
typical usage profile,
preferably with an environment as near to production as possible



Generate representative input messages, preferably with an environment as near to production as possible

Step 1/7

Understand what you want to learn/prove

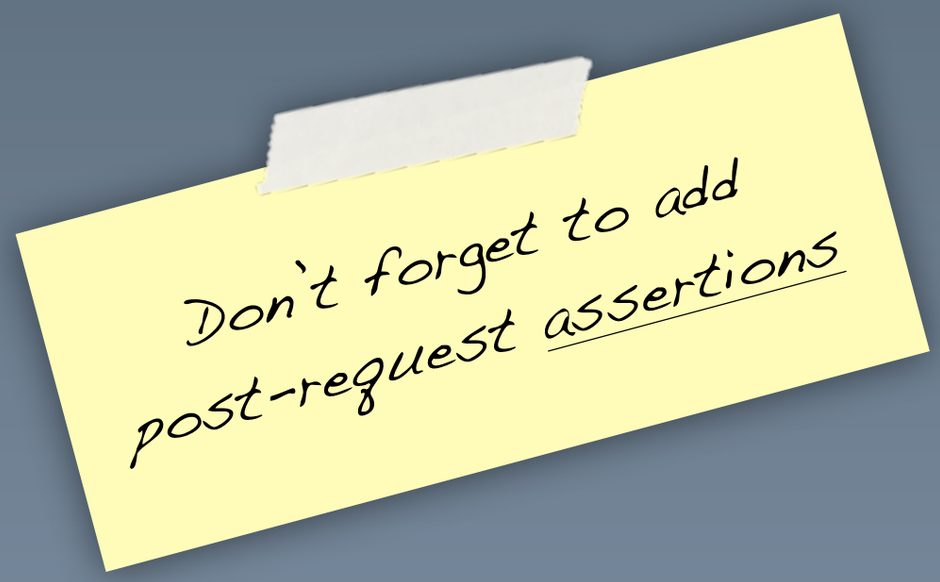
Performance and scalability characteristics, typical usage profiles, etc.

Use log files, audit logs or analytics to understand usage patterns

Step 2/7

Define the test script

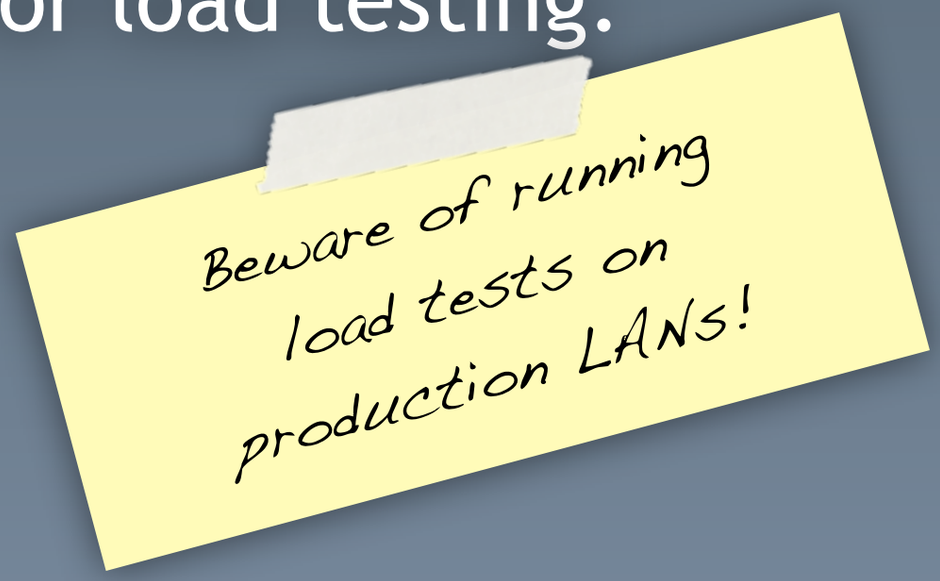
Determine the actions to simulate from the test script and implement with a load testing tool.



Step 3/7

Schedule and configure environment

Book testing slots to minimise disruption and configure/clear data for load testing.



Step 4/7

Determine metrics to record and monitor

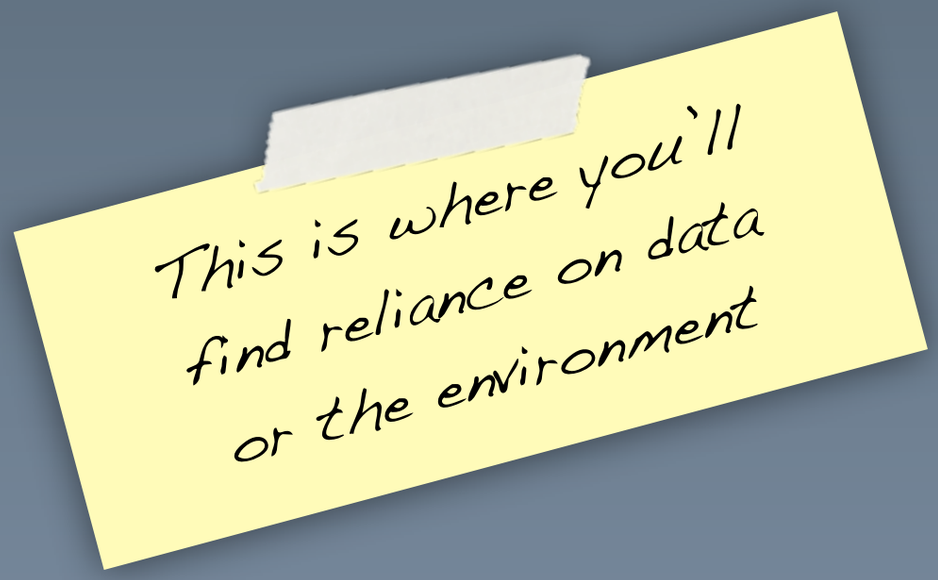
Ensure that the test script captures the appropriate statistics and determine the system characteristics to monitor (e.g. CPU, RAM, IO, etc).

Don't forget to monitor the load testing client(s)

Step 5/7

Execute pre-tests

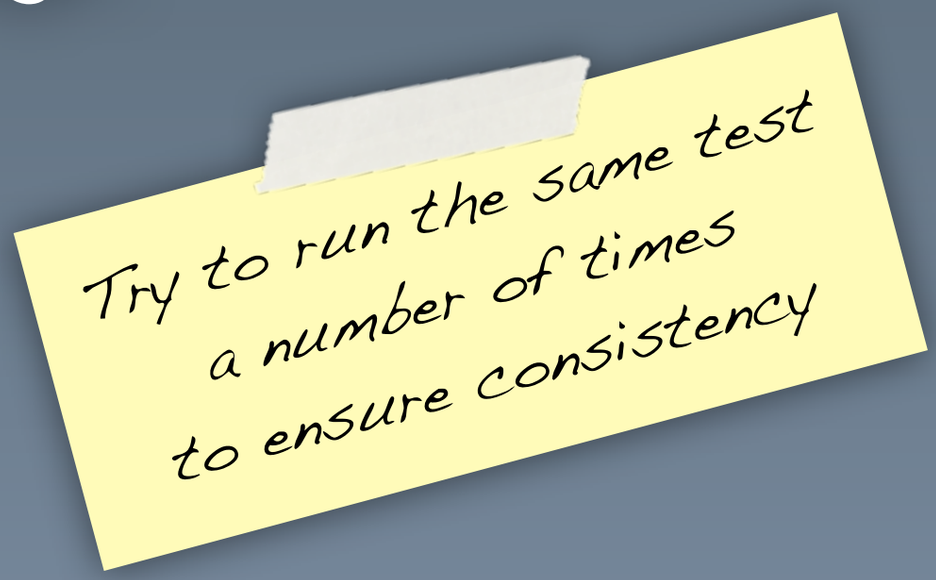
Test the test scripts and monitor,
refining if necessary.



Step 6/7

Execute tests

Clear down the environment, warm it up, execute tests at varying levels of concurrent usage.



Step 7/7

Analyse results

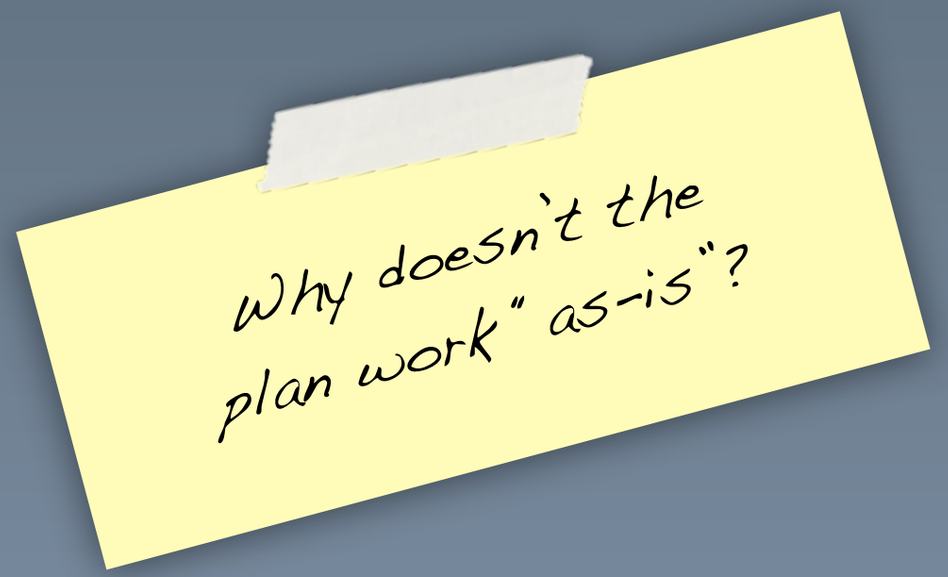
Calculate useful statistics (average, maximum and 95th percentile response times), draw graphs and make conclusions.

Let’s load test!

Create a basic load test script using Apache JMeter

Make the plan work for a

single user



Add some

response assertions

Make the plan work for

five users

Parameterise

the account number used when requesting the view account page

Run the plan in JMeter's

headless mode

Parameterise

the number of users, so that it can be specified on the command line

Parameterise

the hostname and port for all samplers

Make the plan more

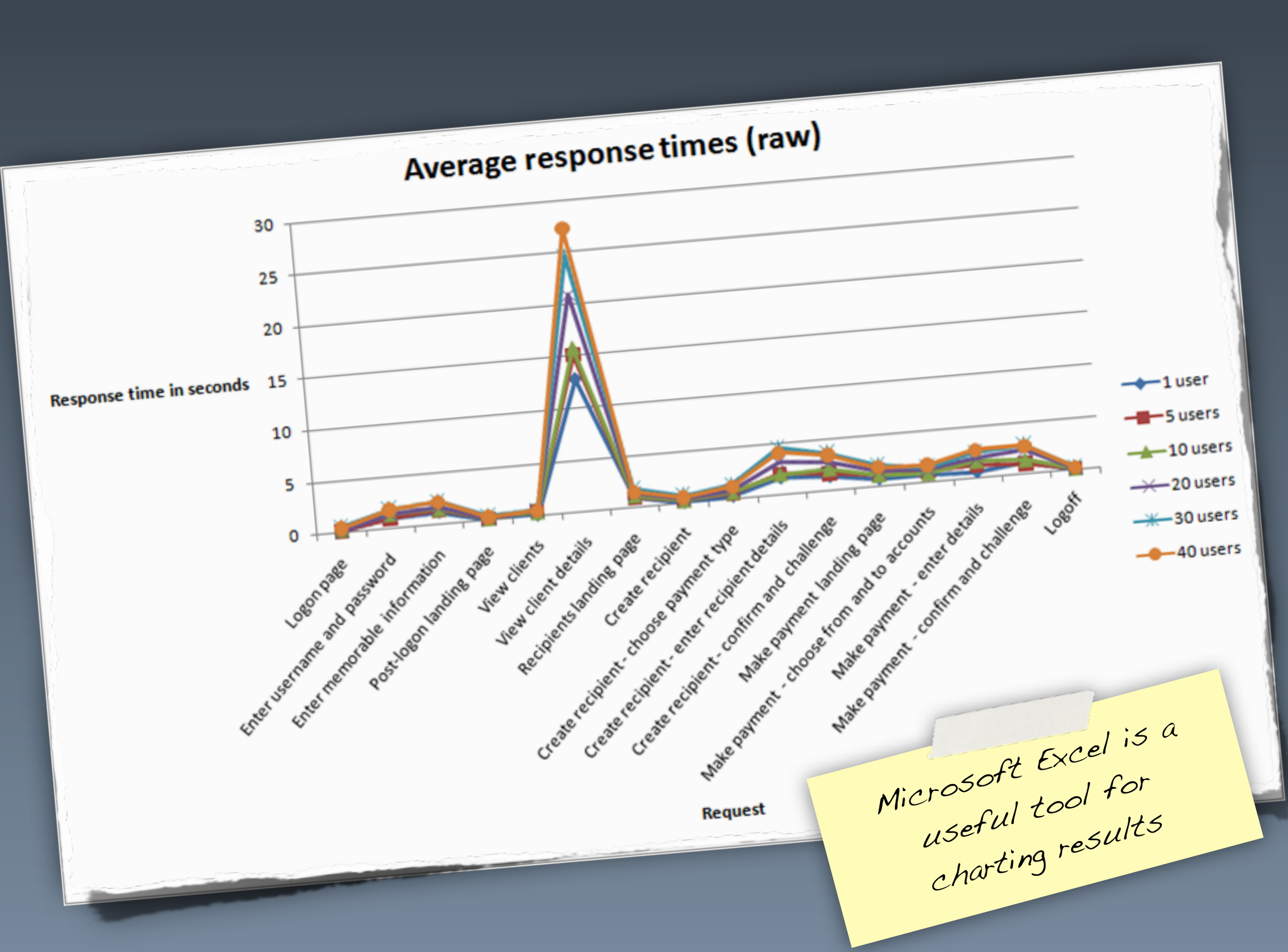
realistic

Execute some tests and

capture the results

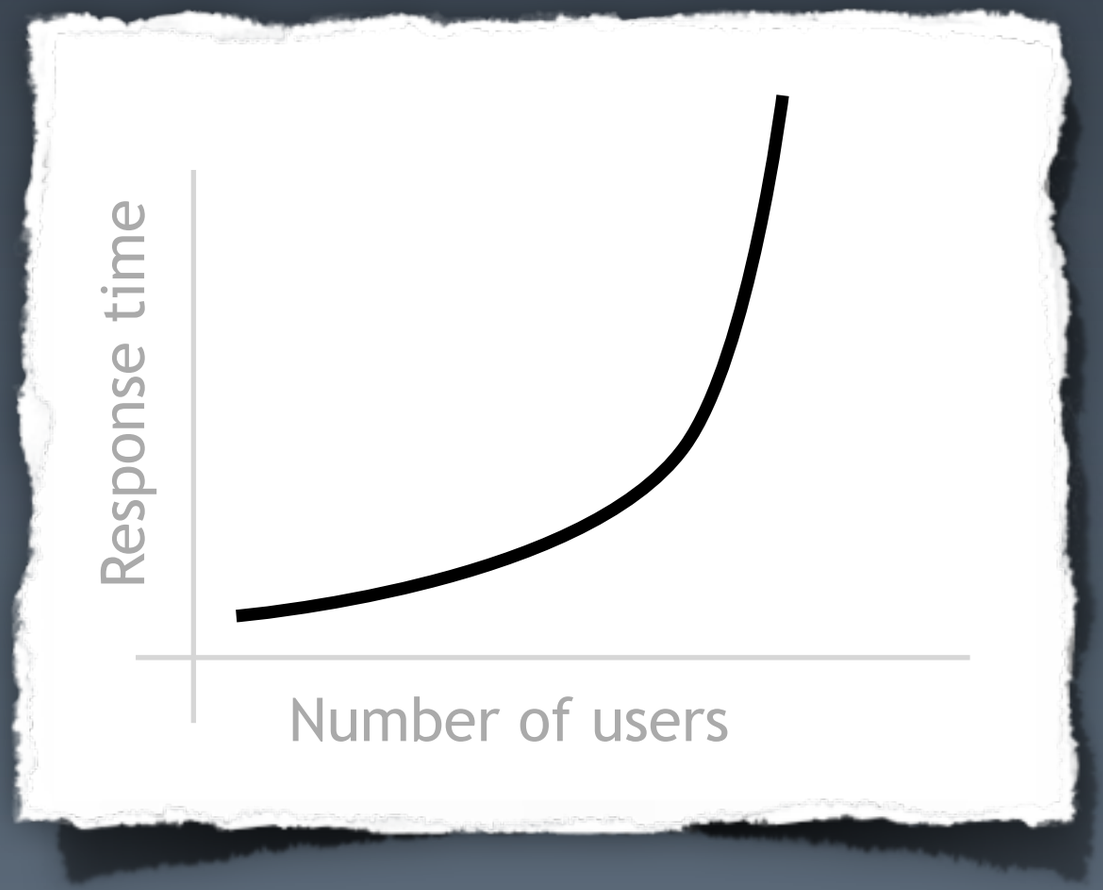
Analysing the results

Post-process the raw results



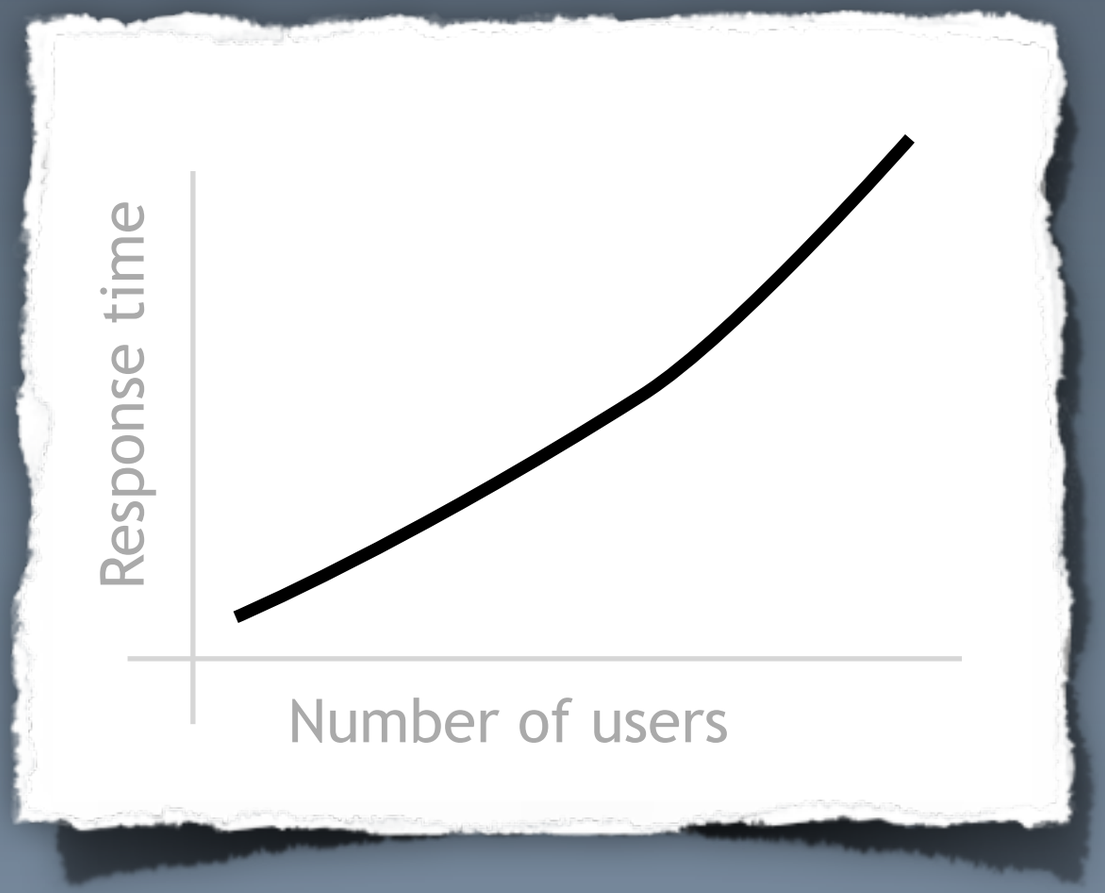
Microsoft Excel is a useful tool for charting results

Draw this sort of graph if you need to understand scalability



This shows **poor** scalability because response times have started to increase exponentially

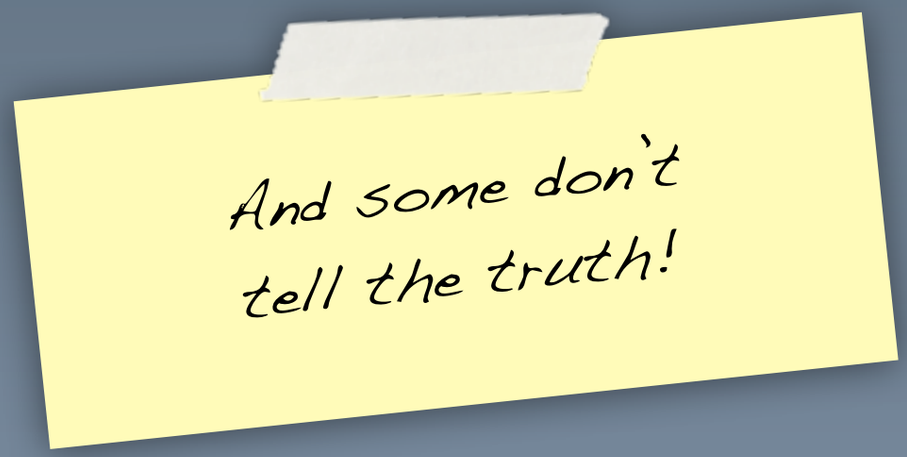
This shows **good** scalability because response times increase in a predictable way



Load testing in the real world?

Few

software teams do any sort of load testing



Even a little testing can increase

confidence,

particularly if done

early

How much

architecture do you need to do?

This much?

This much?

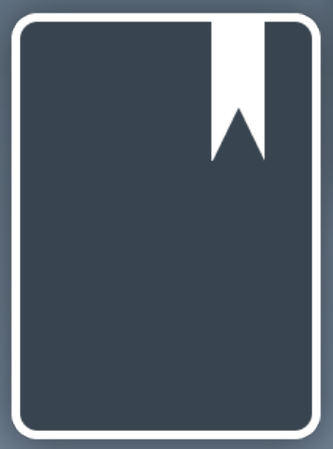
You need to define

“just enough”

architecture

Just enough to give you clarity and vision?

Base your architecture on requirements, travel light and prove your architecture with concrete experiments.



Scott Ambler
http://www.agilemodeling.com/essays/agileArchitecture.htm

Agile Architecture

http://www.agilemodeling.com/essays/agileArchitecture.htm

Concrete experiments

Requirements

A system

Foundations

A throwaway prototype/ proof of concept or production code

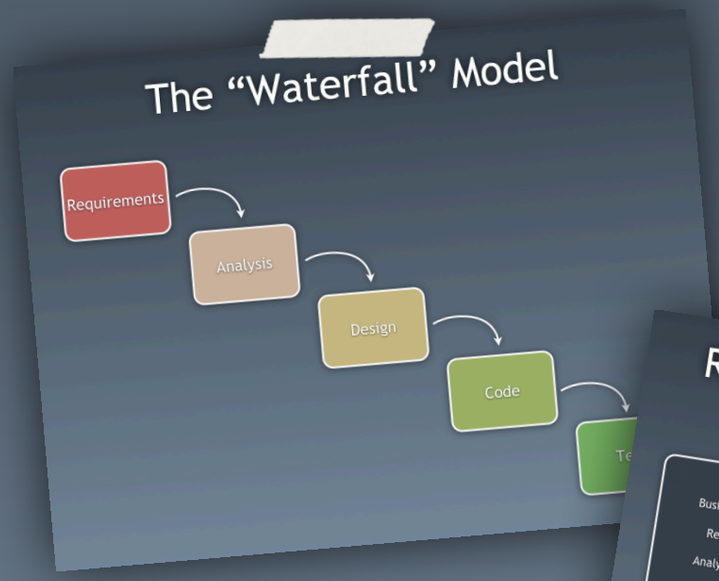
What is architecturally **significant**?

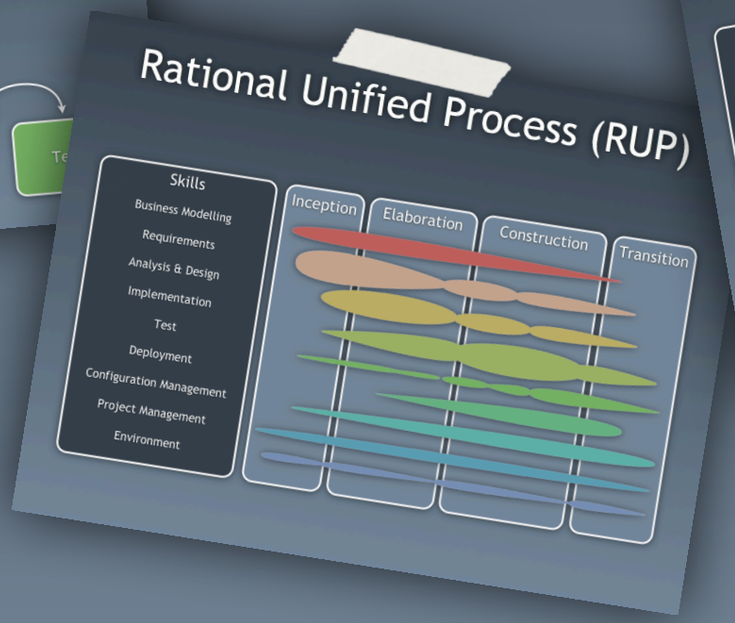
Costly to change
(can you refactor it in an afternoon?)

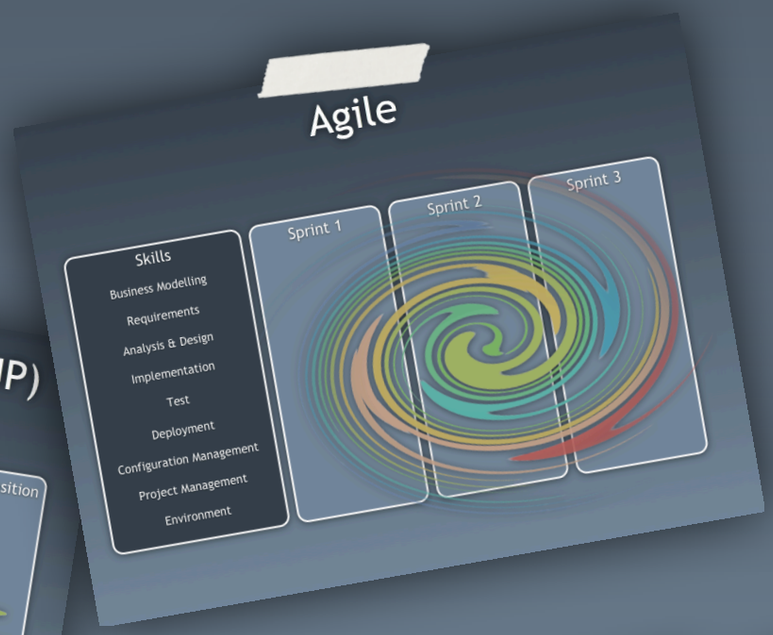
Complex and risky

New

Test early







Who defines your software development methodology?

Diagnosing problems

What happens if you find

performance problems?

Look at your monitoring data

and log files,

run profilers,

etc…

Maybe the environment needs tuning

[Memory | Connection pools | Worker thread pools | Database indexes & hints | etc]

The **big** picture and

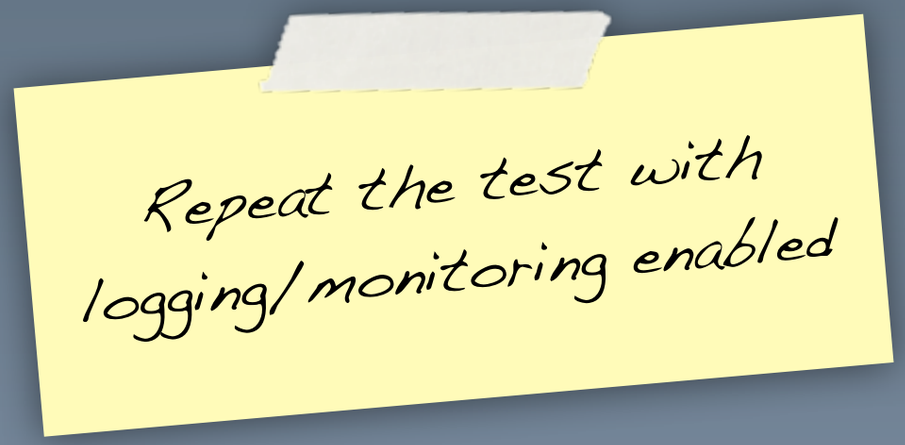
the **detail**

are equally as important

My application has

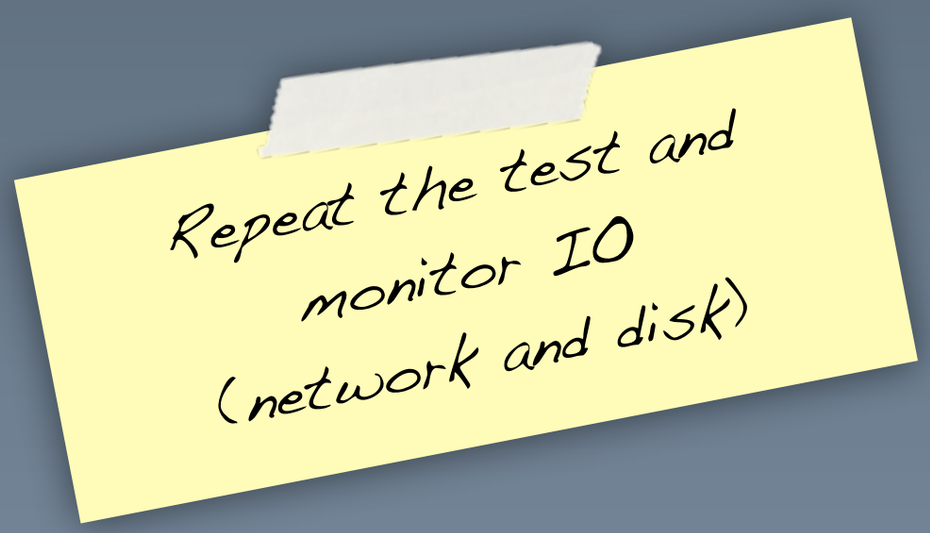
intermittent

poor performance



My application is slow, but it's

not using any CPU



My application is

slow

Find the bottleneck by profiling your application

.NET Gotchas?

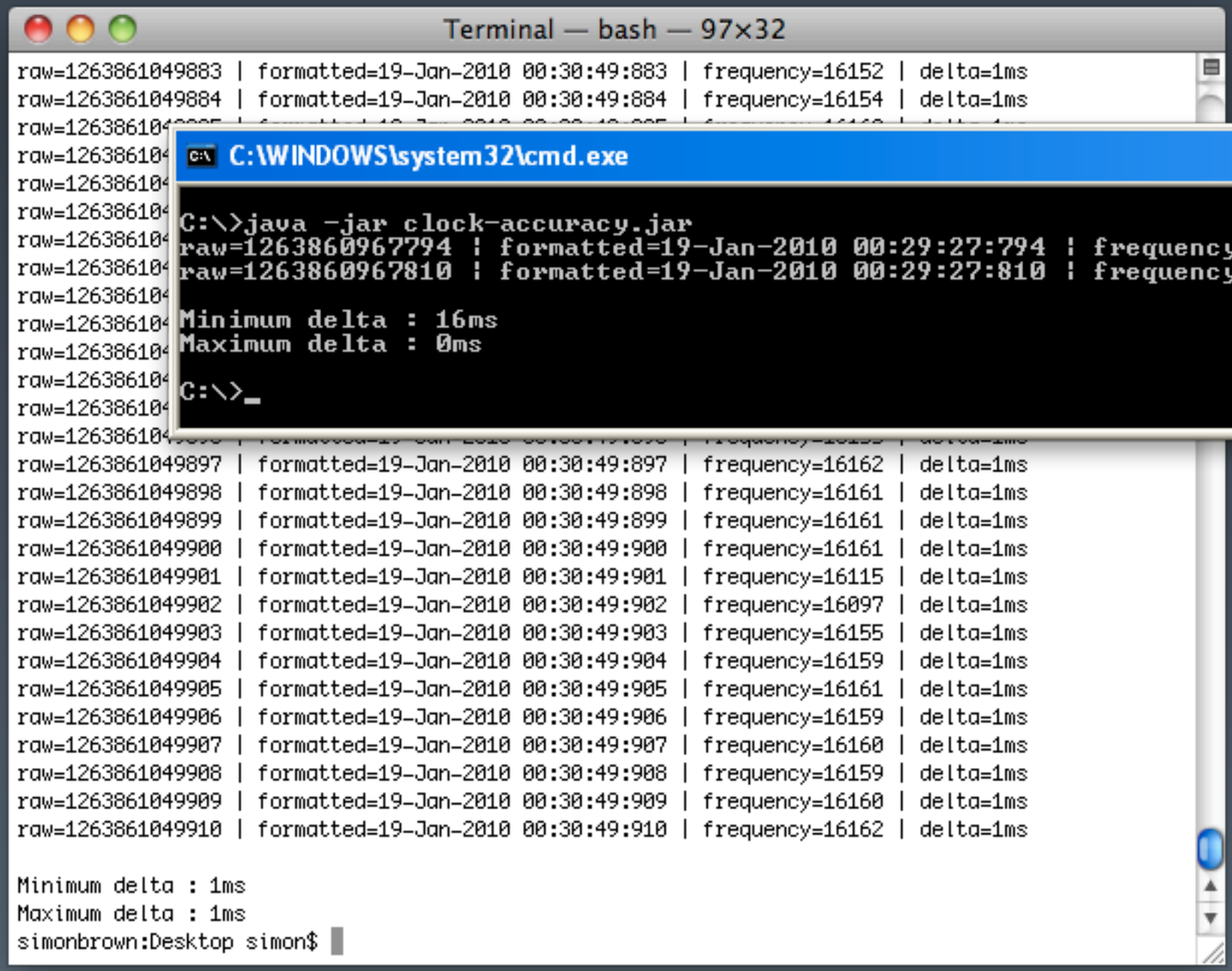
Not really, but watch out for

ViewState

(and EventValidation)

I've load tested AJAX, WIF and SharePoint sites without any major problems

Other Gotchas?



16 milliseconds on Windows

Clock granularity

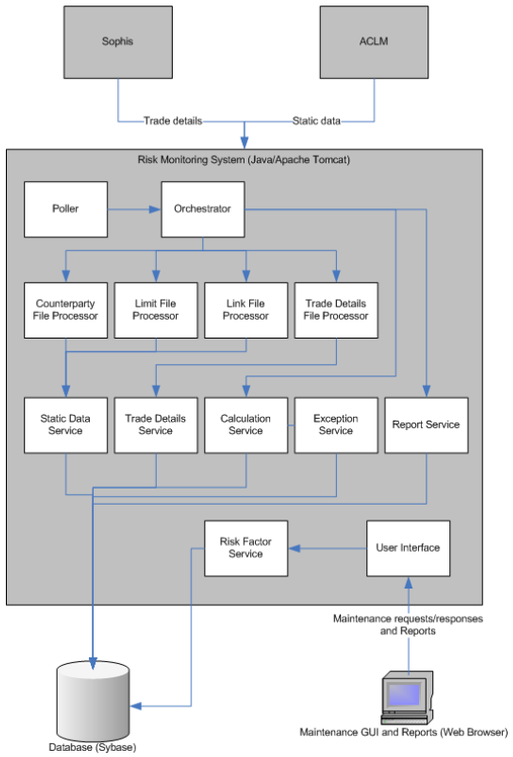
can vary by platform

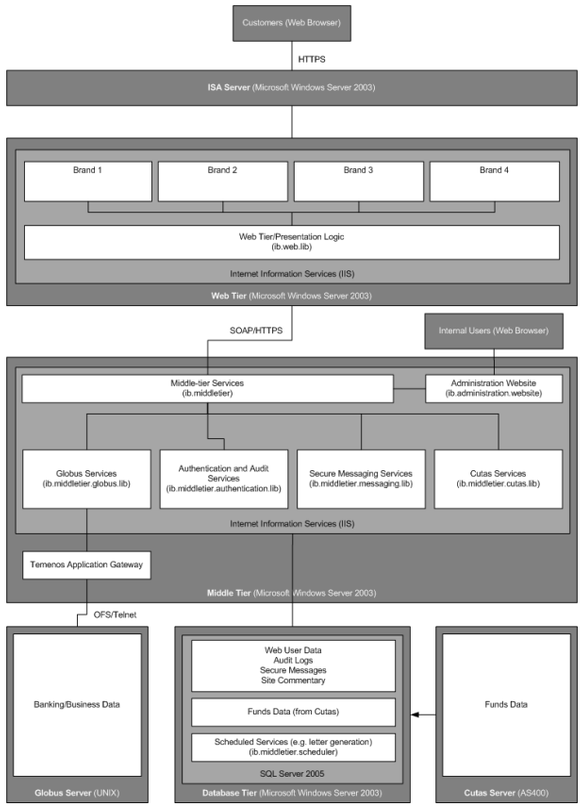
What's the

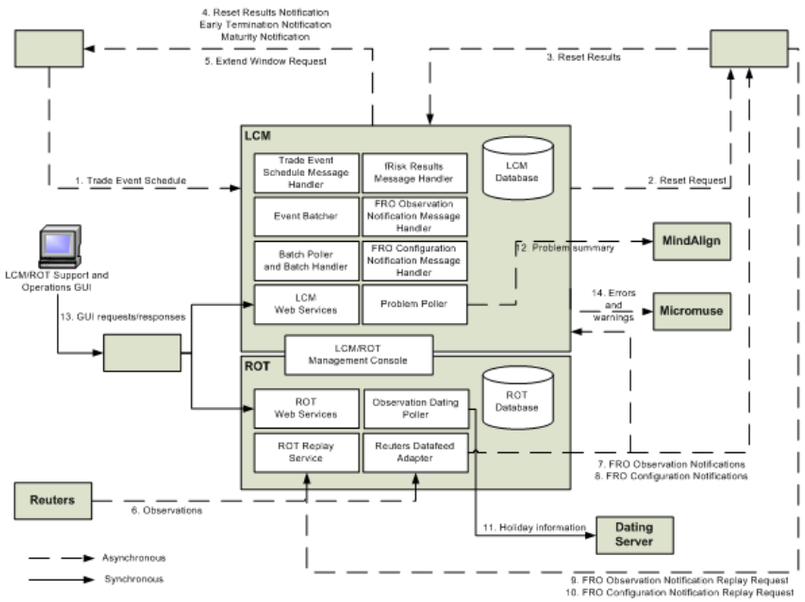
time?

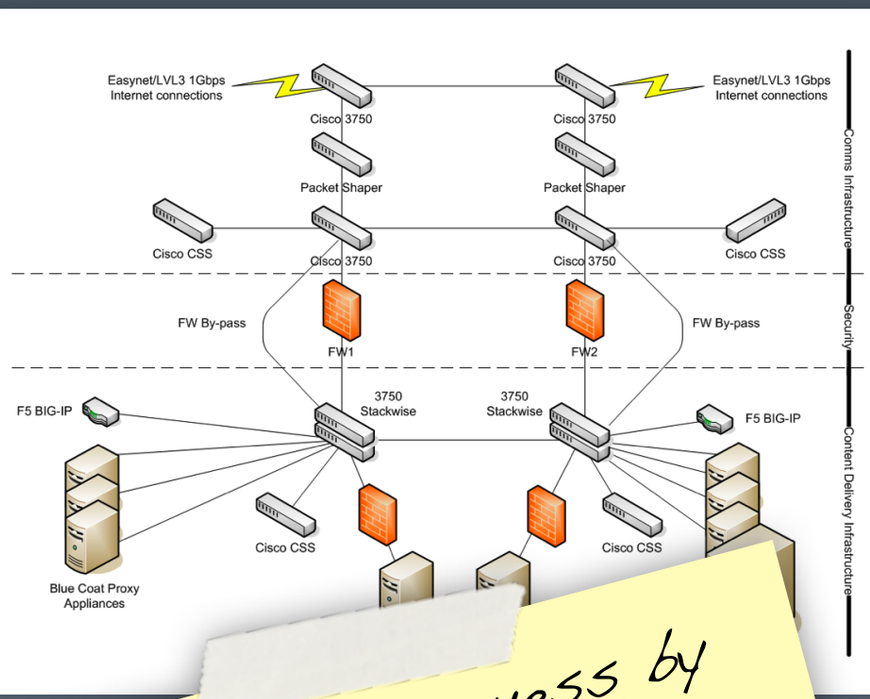
(are you sure?)

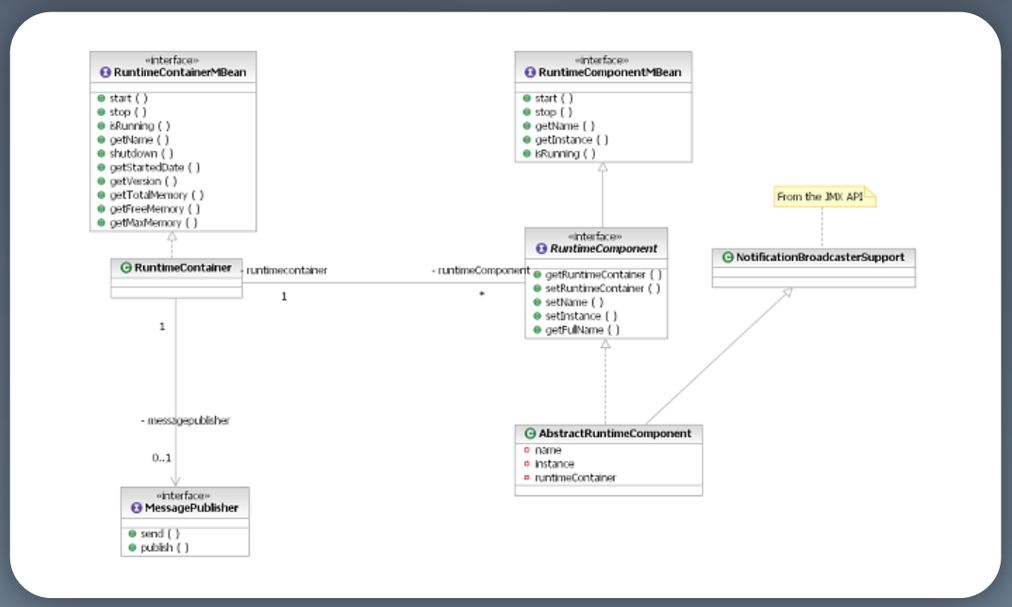
Let’s wrap up











Will these software systems

perform and scale acceptably?

You can only guess by looking at diagrams or source code

You need to define

“just enough”

architecture

Identify whether performance and/or scalability are important

Load testing

should be part of

the software developer role

Anybody striving for cross-discipline agile teams?

Don't leave load testing until the end!

Thanks!

simon.brown@codingthearchitecture.com

@simonbrown on Twitter