

coding {the} architecture

Load Testing for Developers

Join the conversation
#prognnet11

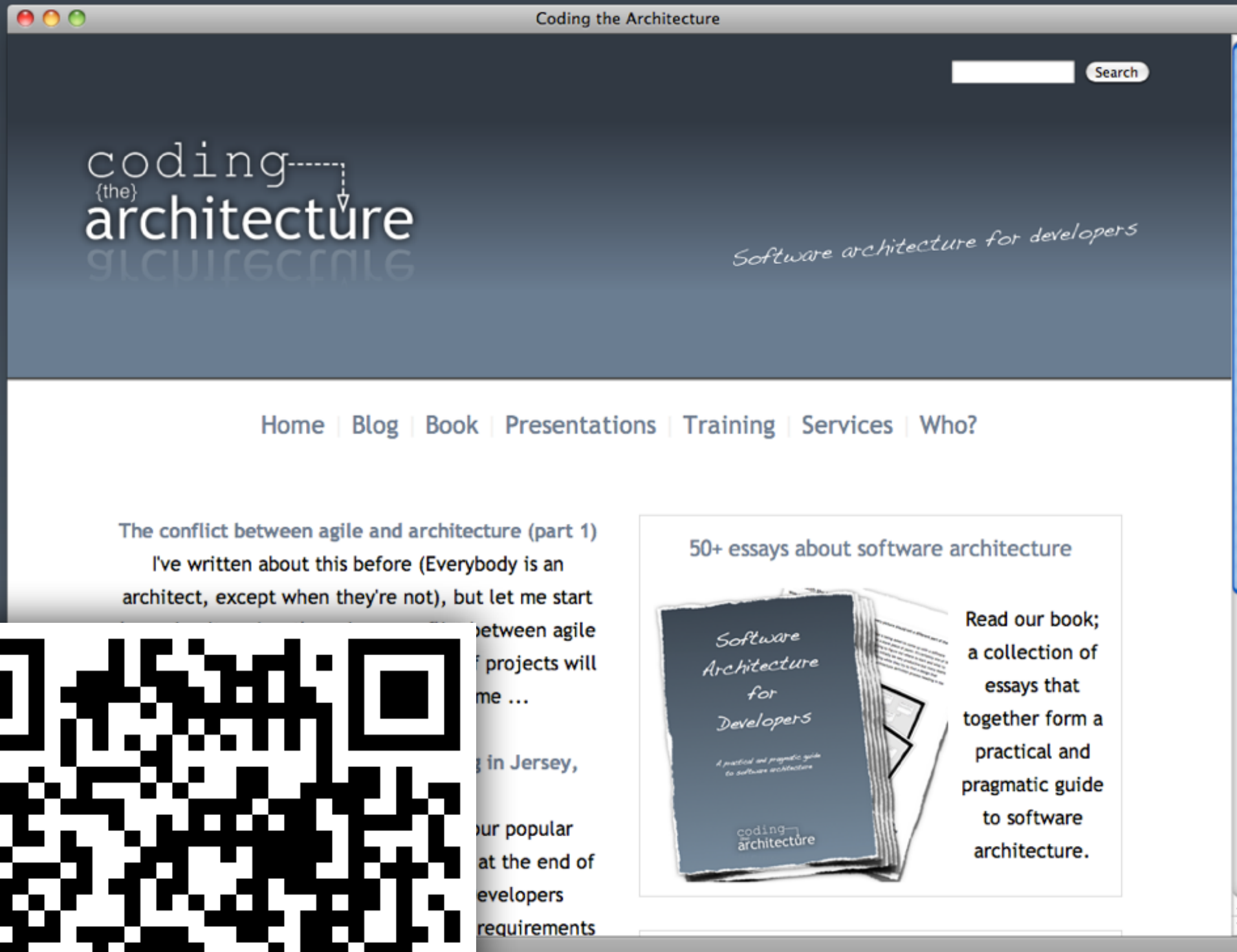


simon.brown@codingthearchitecture.com

@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...

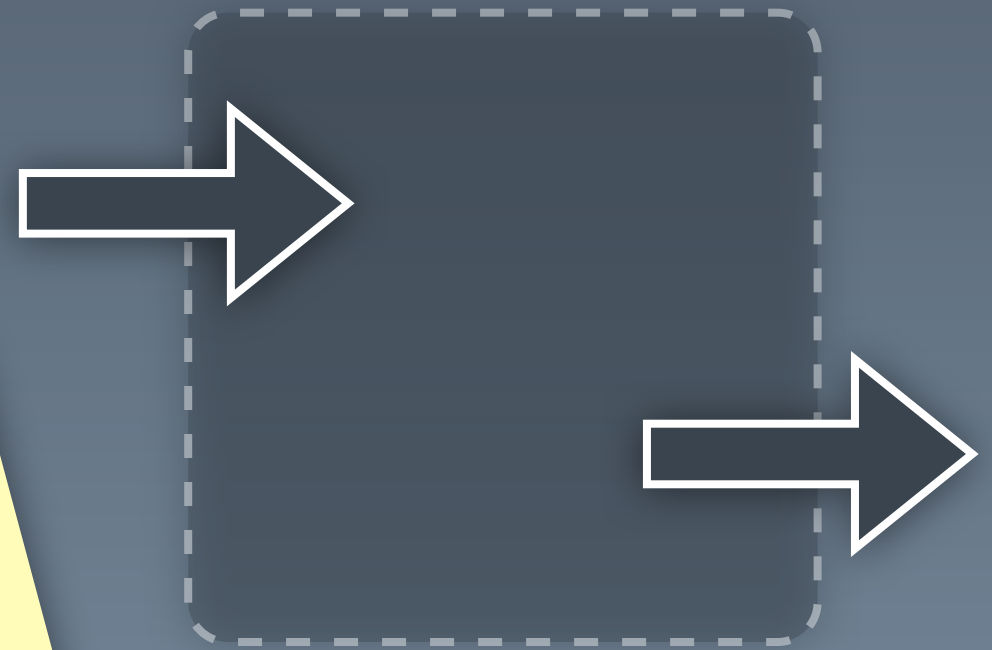
Ever been told that
"the system is slow"?

Response time

Latency^{and} throughput

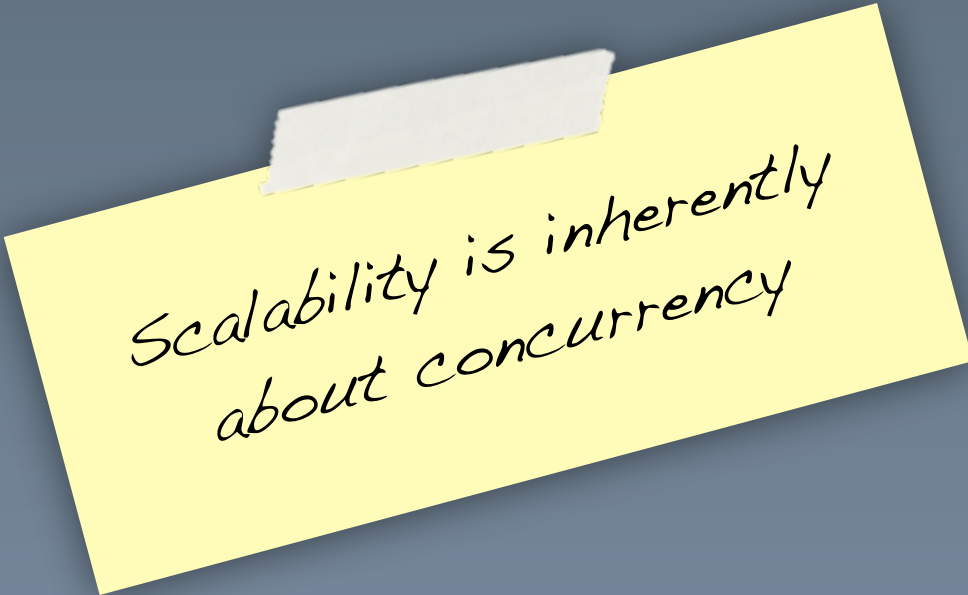


*High performance is about
caching, colocation,
optimisation, etc*



Scalability is about doing more

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

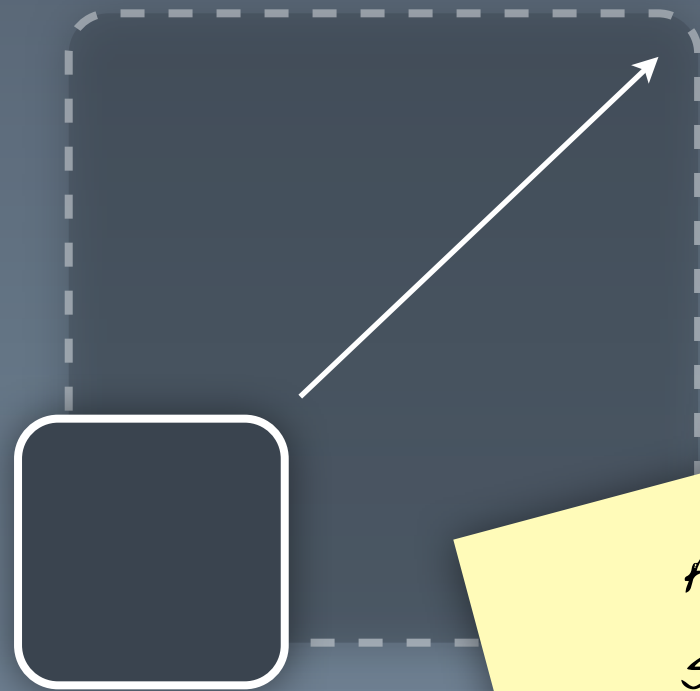


*Scalability is inherently
about concurrency*

Vertical

scalability

(scale-up)

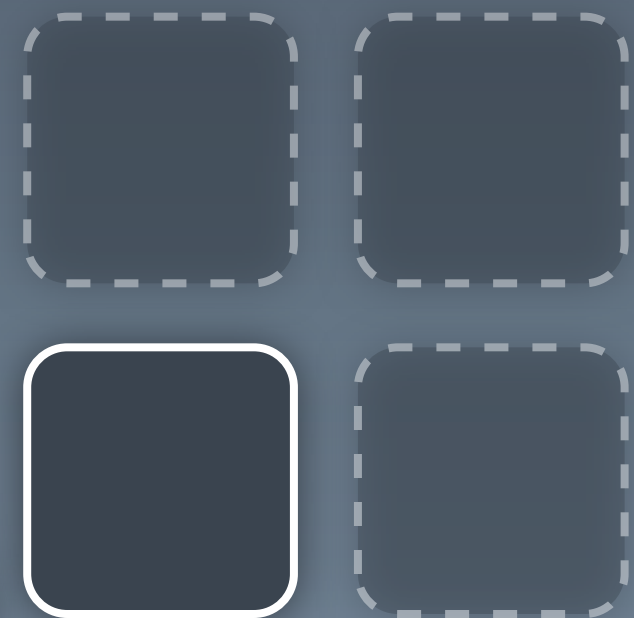


*All components
scale differently;
scale-up is easy*

Horizontal

scalability

(scale-out)

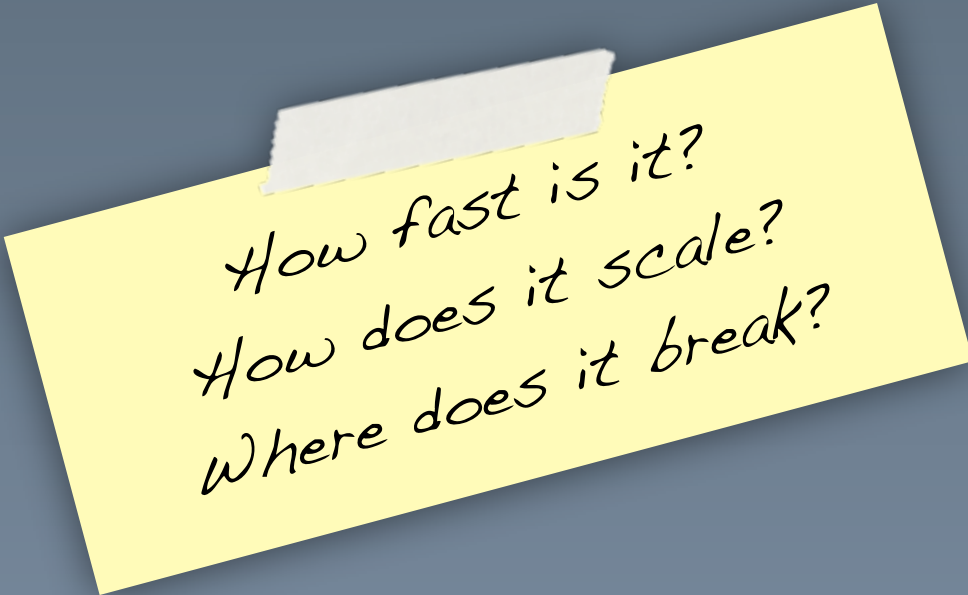


What do you
want to know?

(and why?)

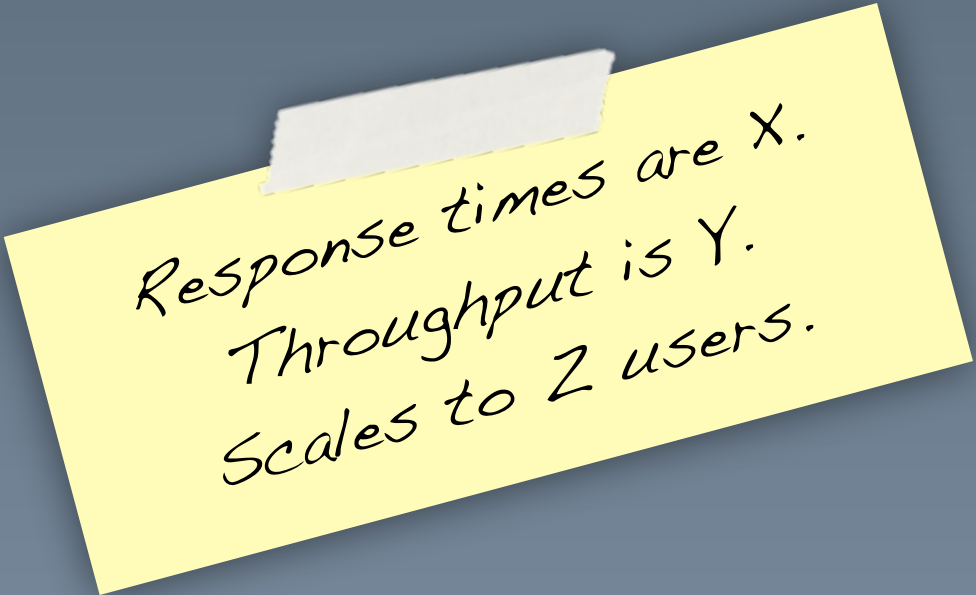
You should have a
reason
for load testing

What do you want to
learn
about your system?

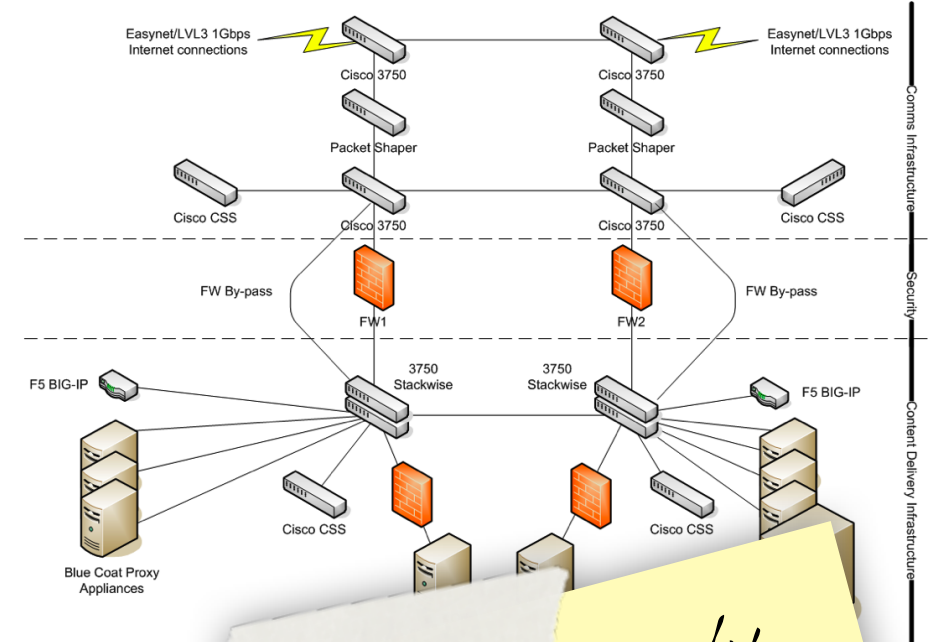
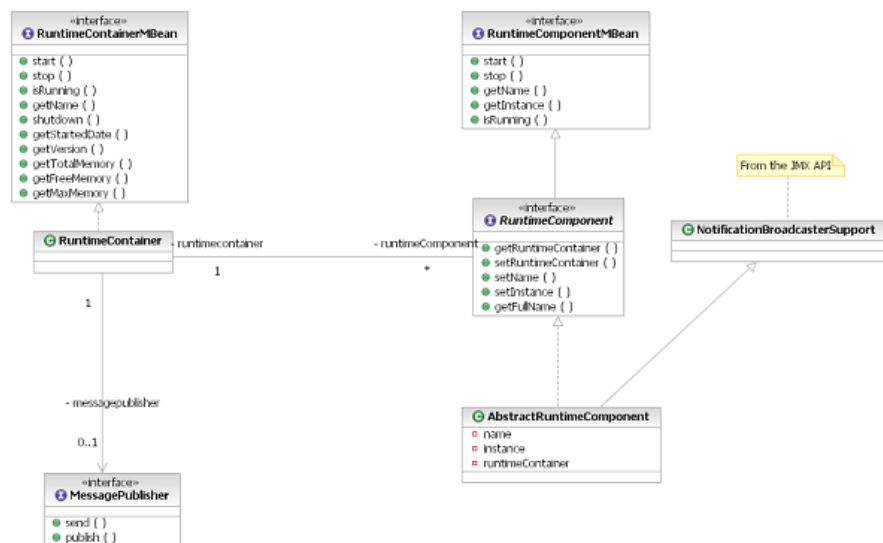
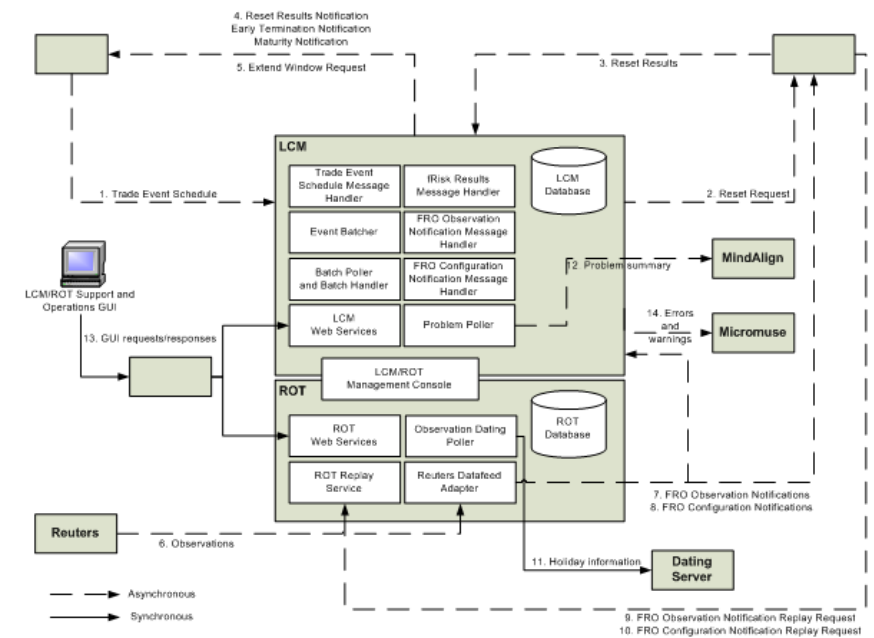
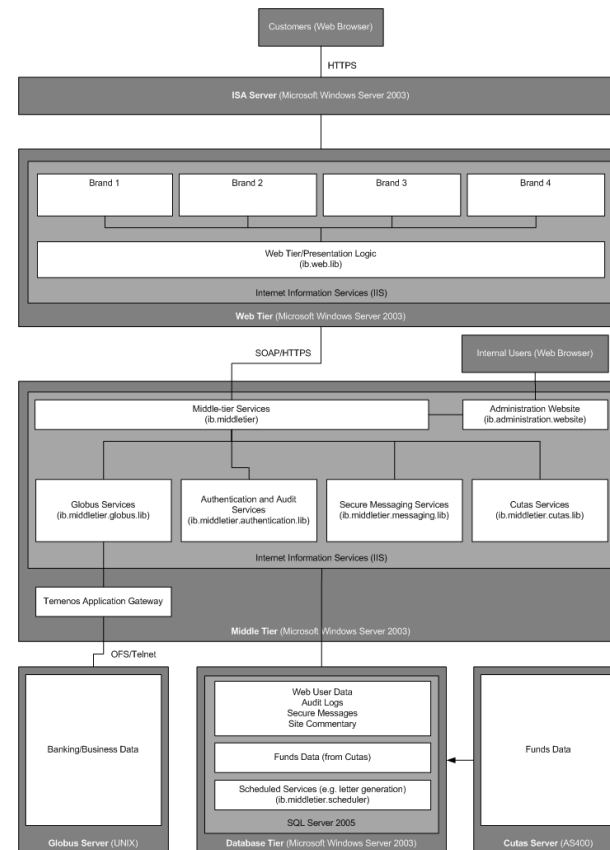
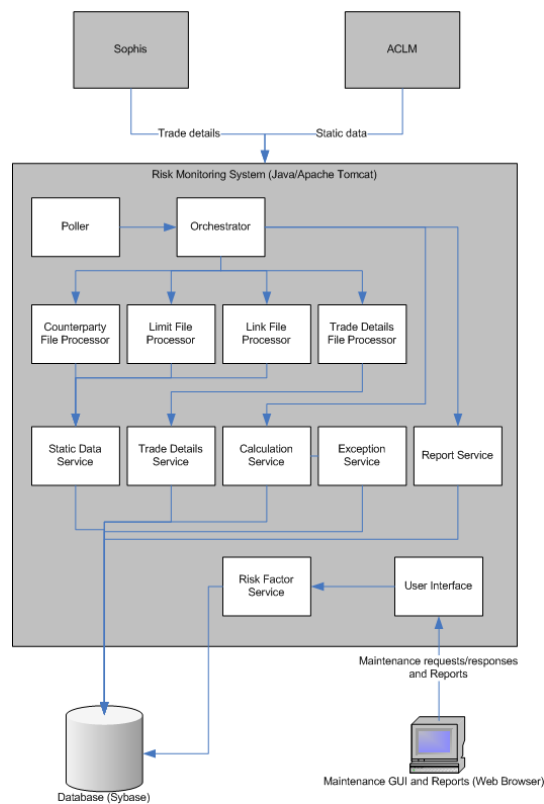


How fast is it?
How does it scale?
Where does it break?

What do you want to
prove
about your system?



Response times are X.
Throughput is Y.
Scales to Z users.



You can only guess by looking at diagrams or source code

Will these software systems perform and scale acceptably?

Software development is not a
relay sport



*This new system will work
because we have a
spreadsheet
showing performance figures
from a past project*



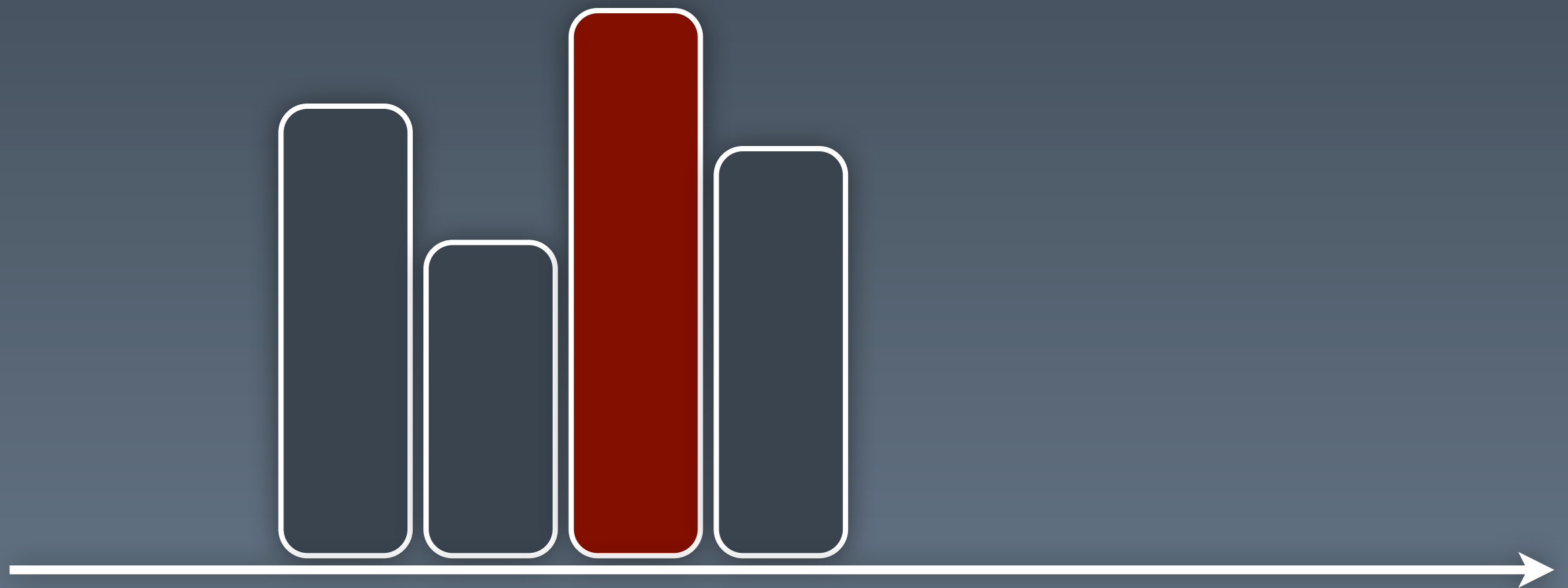
A software “architect”

What is
load testing?

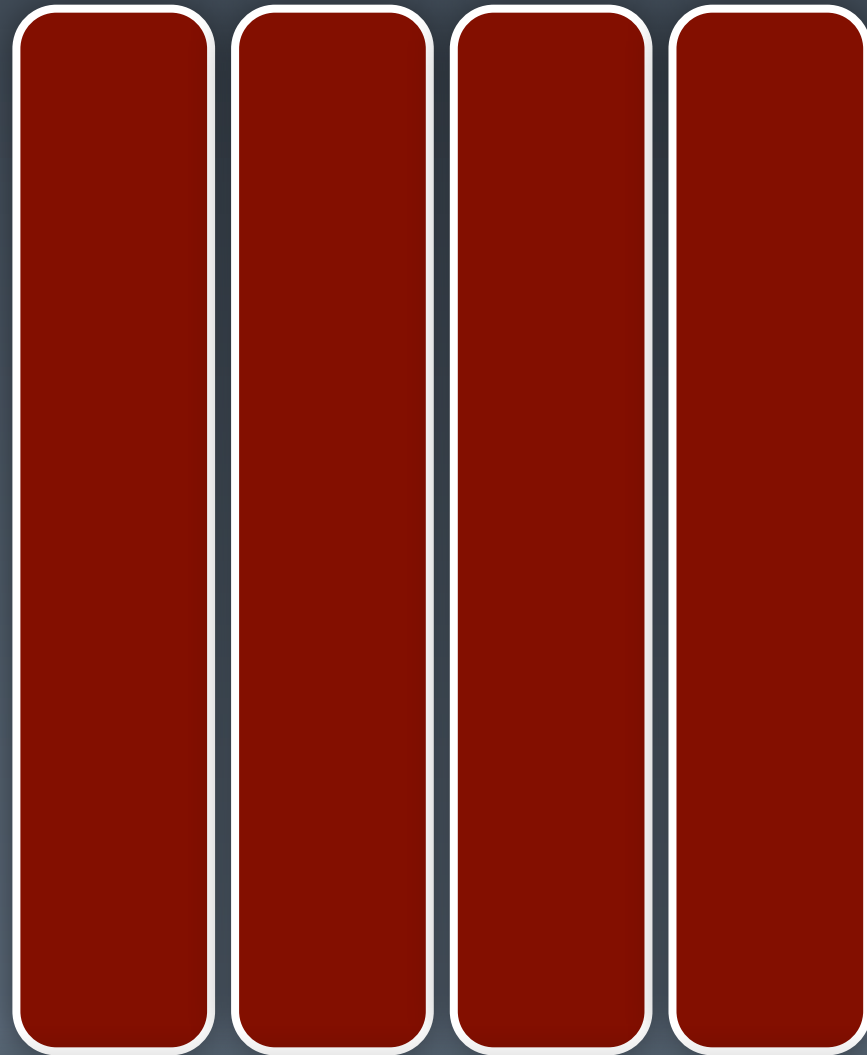
Testing provides
confidence

Load testing is one way to
evaluate
or **validate**
your architecture

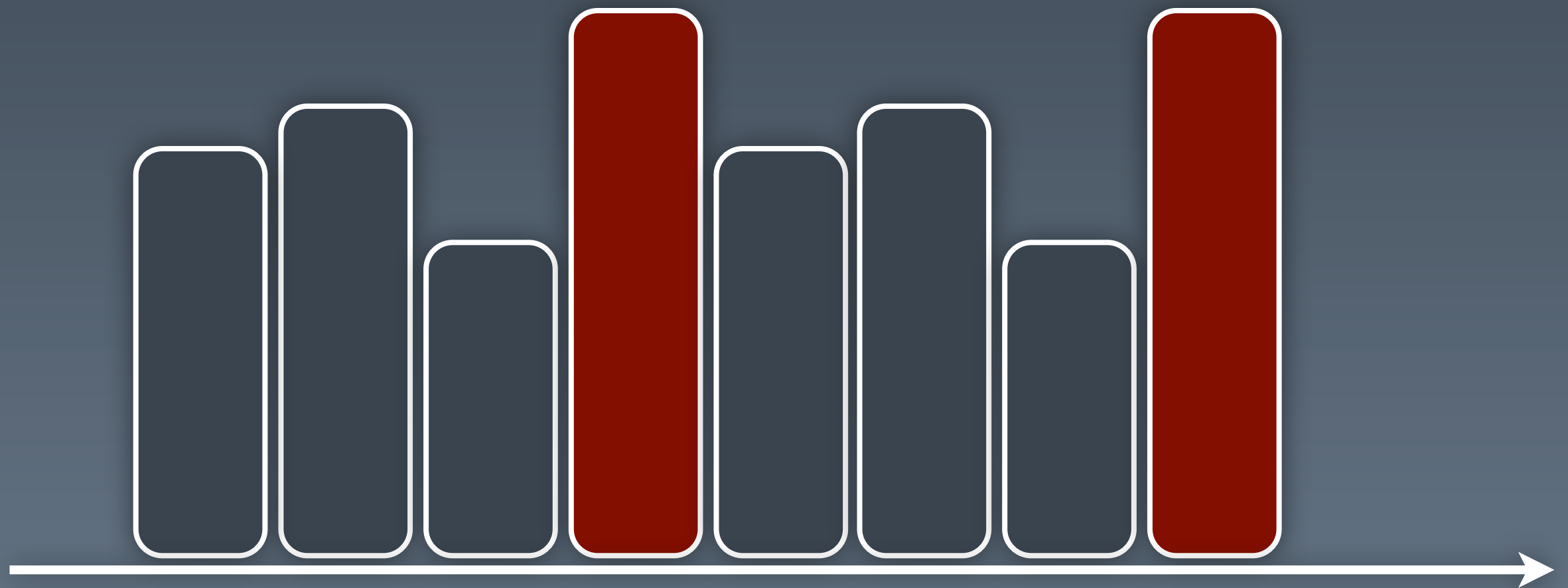
Caveat: if performance and scalability are important characteristics



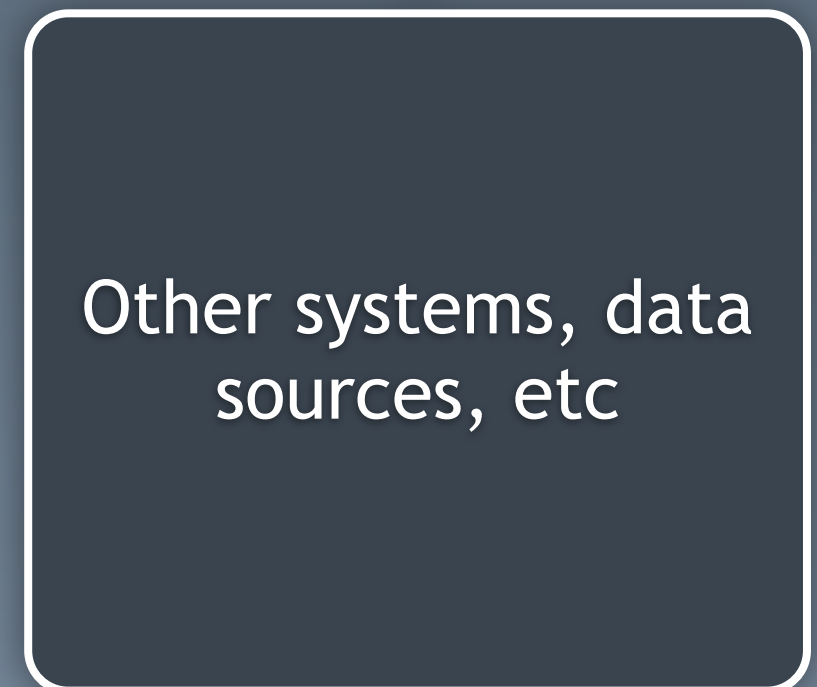
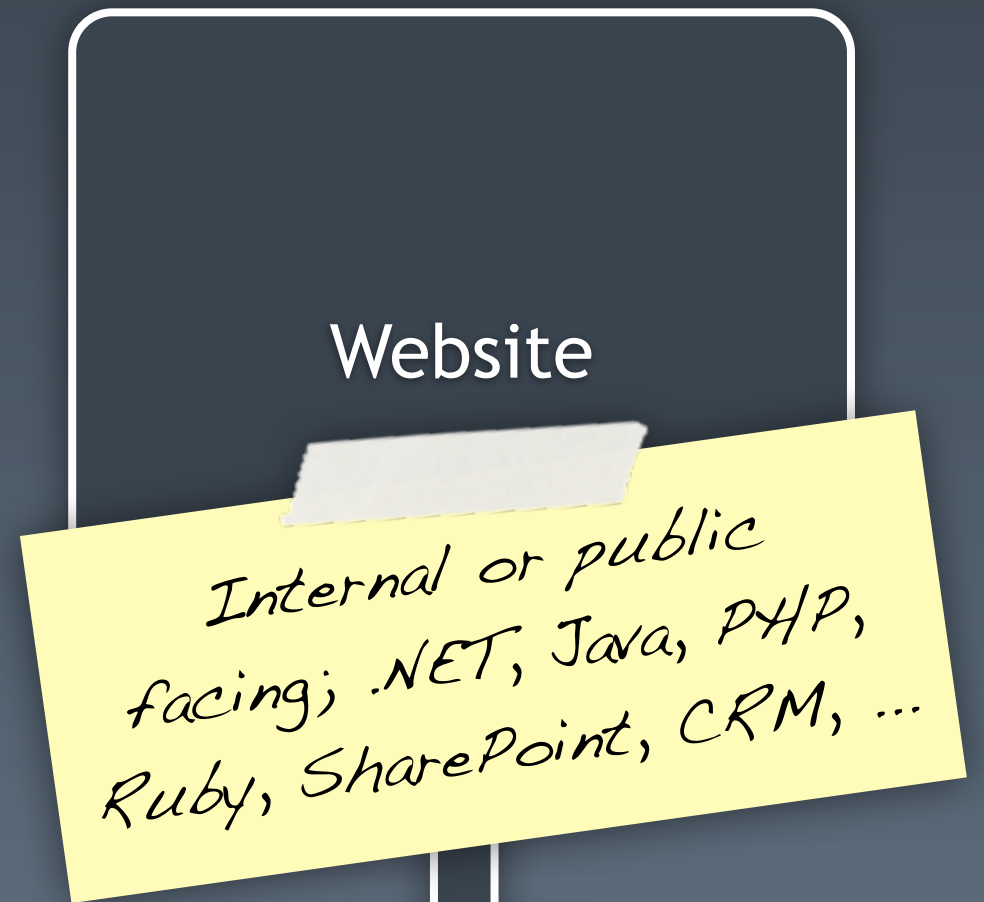
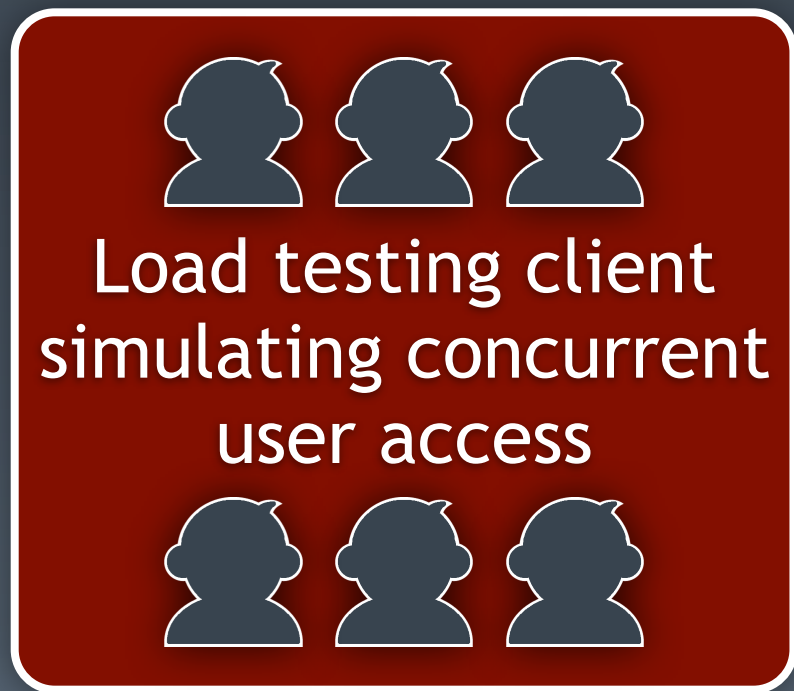
Load testing is asserting how the architecture performs under load with a view to monitoring the response times for key transactions



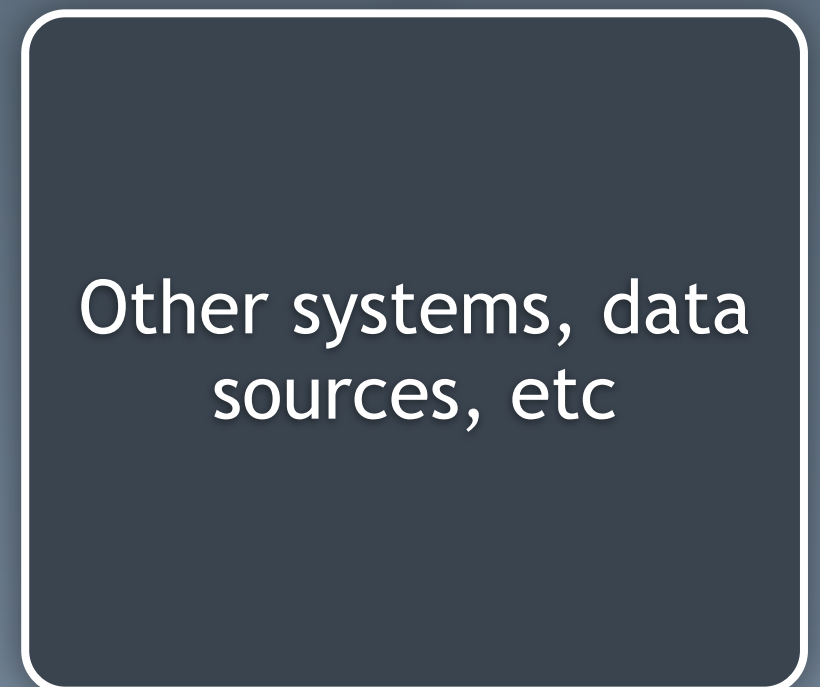
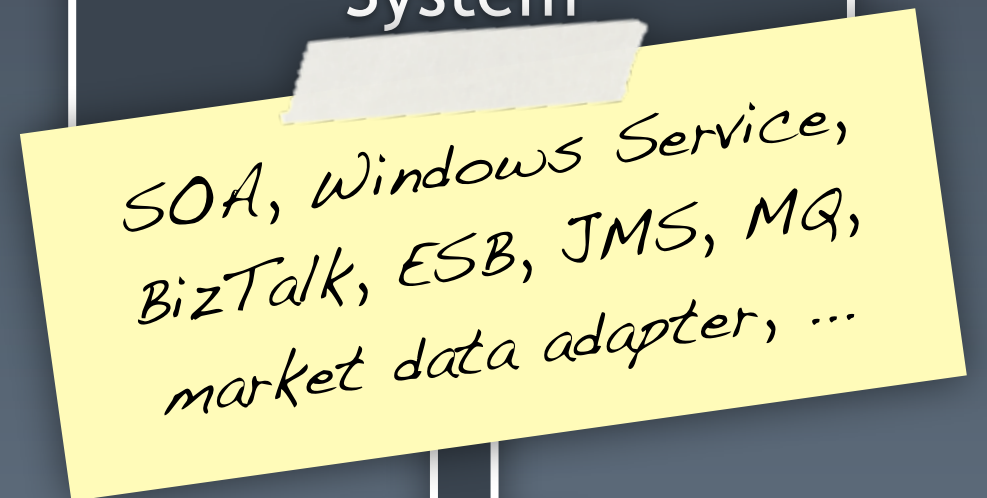
Stress testing is asserting what the upper bounds are for the scalability of the architecture, understanding how it reacts when stressed



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.

Don't forget to add
post-request assertions

Step 3/7

Schedule and configure environment

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

Beware of running
load tests on
production LANs!

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.

This is where you'll
find reliance on data
or the environment

Step 6/7

Execute tests

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

Try to run the same test
a number of times
to ensure consistency

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



Why doesn't the
plan work "as-is"?

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



*What's wrong
with our plan?*

Execute some tests and
capture the results



Analysing
the results

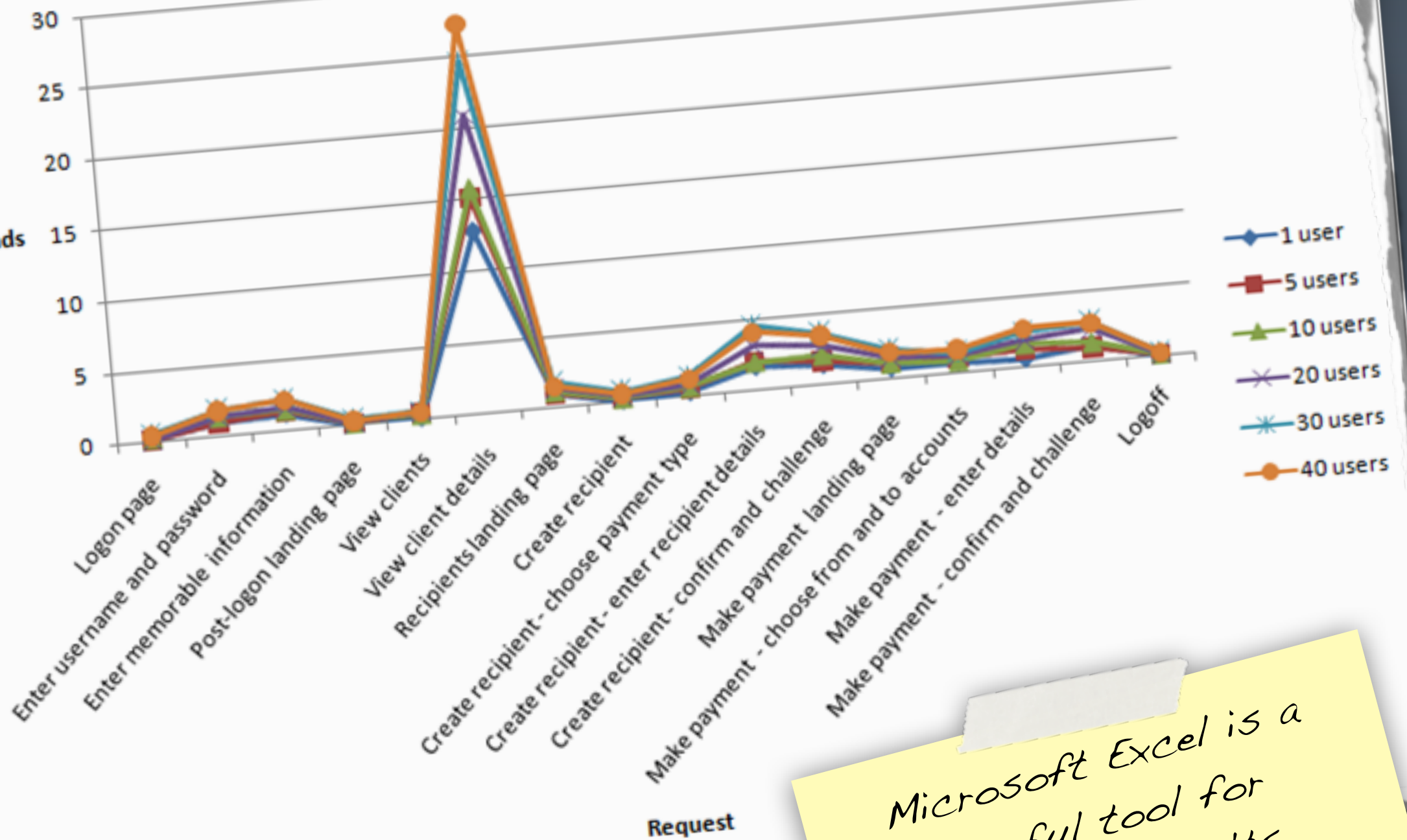
timeStamp, elapsed, label, responseCode, threadName, success, URL

1242921099104,77,01. Logon page,200,Business user 1-1,true,<http://internetbankingdev/user/Logon.aspx>
1242921099318,684,02. Enter username and password,200,Business user 1-1,true,<http://internetbankingdev/user/Logon.aspx>
1242921100101,1057,03. Enter memorable information,302,Business user 1-1,true,<http://internetbankingdev/user/Logon.aspx>
1242921101170,10,04. Post-logon landing page,200,Business user 1-1,true,<http://internetbankingdev/user/landing-page.aspx>
1242921101191,42,05. View clients,200,Business user 1-2,true,<http://internetbankingdev/view/relationship.aspx>
1242921104394,13,01. Logon page,200,Business user 1-2,true,<http://internetbankingdev/user/Logon.aspx>
1242921104420,788,02. Enter username and password,200,Business user 1-2,true,<http://internetbankingdev/user/Logon.aspx>
1242921105232,872,03. Enter memorable information,302,Business user 1-2,true,<http://internetbankingdev/user/landing-page.aspx>
1242921106120,9,04. Post-logon landing page,200,Business user 1-2,true,<http://internetbankingdev/view/relationship.aspx>
1242921106141,715,05. View clients,200,Business user 1-3,true,<http://internetbankingdev/user/Logon.aspx>
1242921109600,166,01. Logon page,200,Business user 1-3,true,<http://internetbankingdev/user/Logon.aspx>
1242921109775,582,02. Enter username and password,200,Business user 1-3,true,<http://internetbankingdev/user/landing-page.aspx>
1242921110377,835,03. Enter memorable information,302,Business user 1-3,true,<http://internetbankingdev/view/account-list.aspx>
1242921111221,9,04. Post-logon landing page,200,Business user 1-1,true,<http://internetbankingdev/view/relationship.aspx>
1242921101254,10312,06. View client details,200,Business user 1-3,true,<http://internetbankingdev/recipients/Default.aspx>
1242921111239,925,05. View clients,200,Business user 1-1,true,<http://internetbankingdev/recipients/Create.aspx>
1242921111584,1612,07. Recipients landing page,200,Business user 1-1,true,<http://internetbankingdev/recipients/Create.aspx>
1242921113208,51,08. Create recipient,200,Business user 1-1,true,<http://internetbankingdev/recipients/Create.aspx>
1242921113274,268,09. Create recipient - choose payment type,200,Business user 1-1,true,<http://internetbankingdev/user/Logon.aspx>
1242921113558,1613,10. Create recipient - enter recipient details,200,Business user 1-4,true,<http://internetbankingdev/user/Logon.aspx>
1242921114709,480,01. Logon page,200,Business user 1-4,true,<http://internetbankingdev/user/Logon.aspx>
1242921115197,626,02. Enter username and password,200,Business user 1-4,true,<http://internetbankingdev/user/Logon.aspx>
1242921115857,957,03. Enter memorable information,302,Business user 1-4,true,<http://internetbankingdev/user/landing-page.aspx>
1242921116823,11,04. Post-logon landing page,200,Business user 1-4,true,<http://internetbankingdev/view/relationship.aspx>
1242921116842,37,05. View clients,200,Business user 1-4,true,<http://internetbankingdev/view/relationship.aspx>

Post-process the raw results

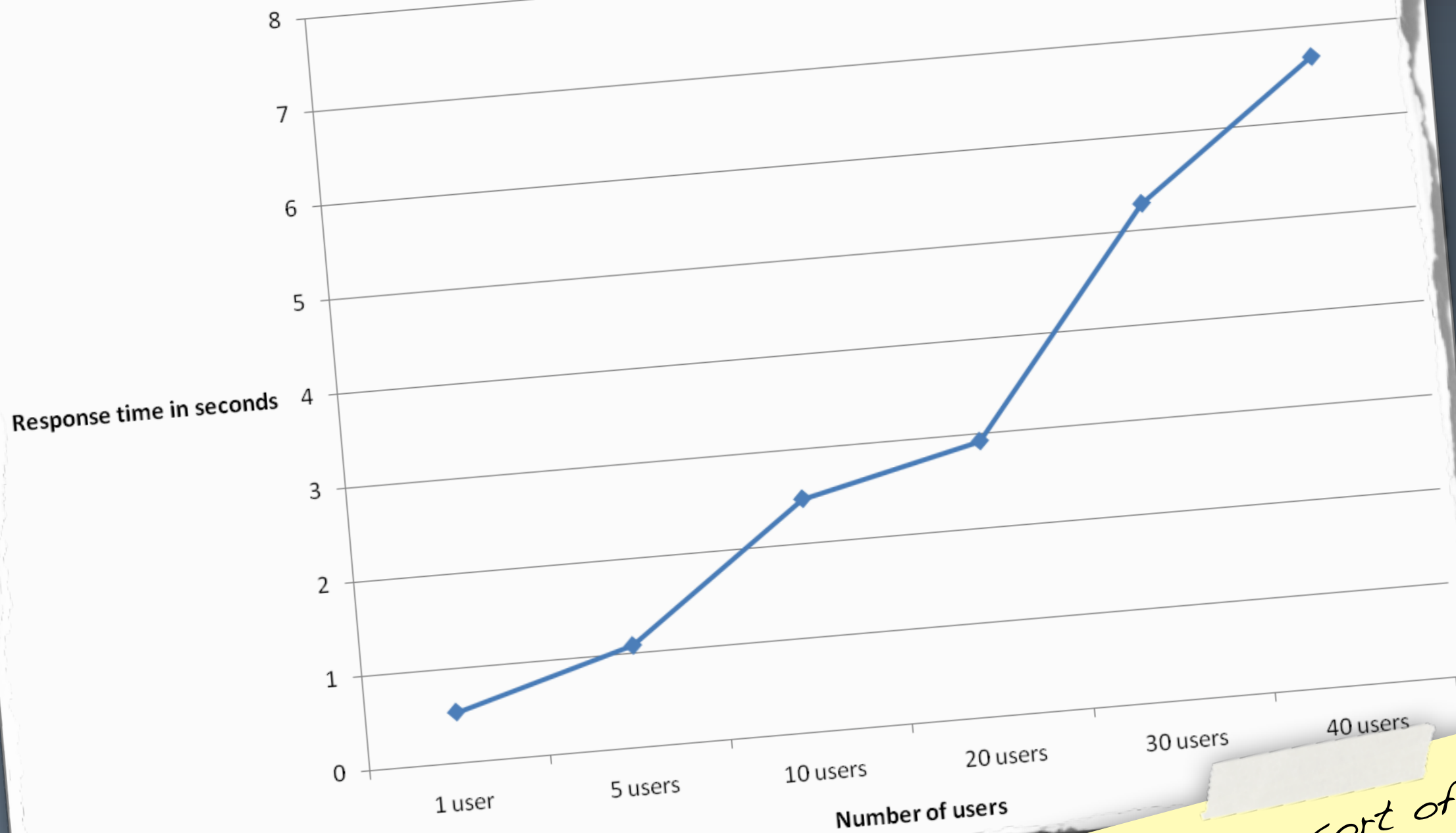
Average response times (raw)

Response time in seconds

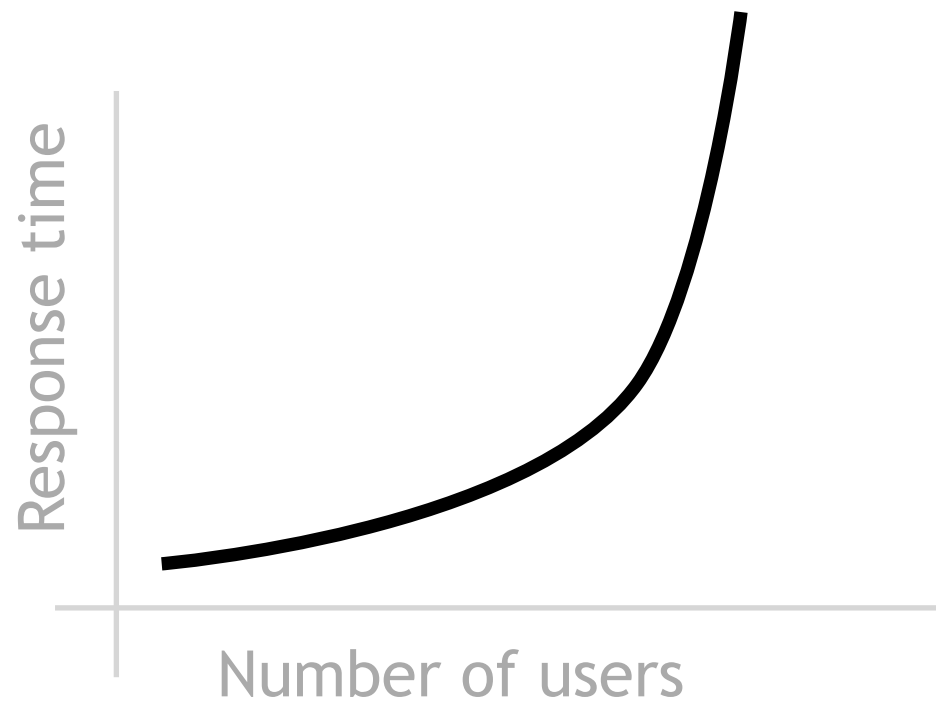


Microsoft Excel is a useful tool for charting results

Scalability: Make Payment - enter details

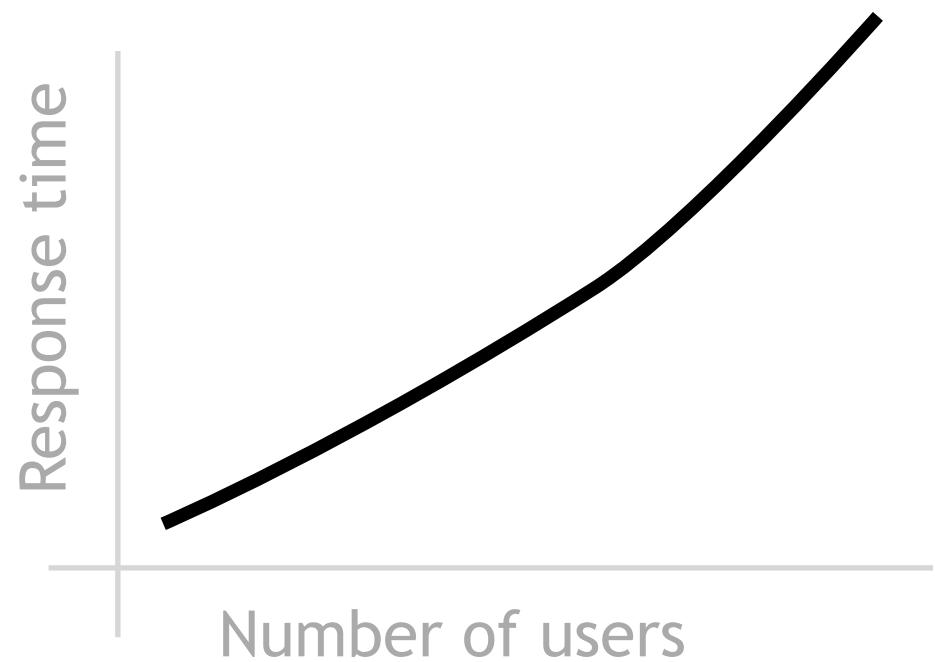


Draw this sort of graph
if you need to
understand scalability



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

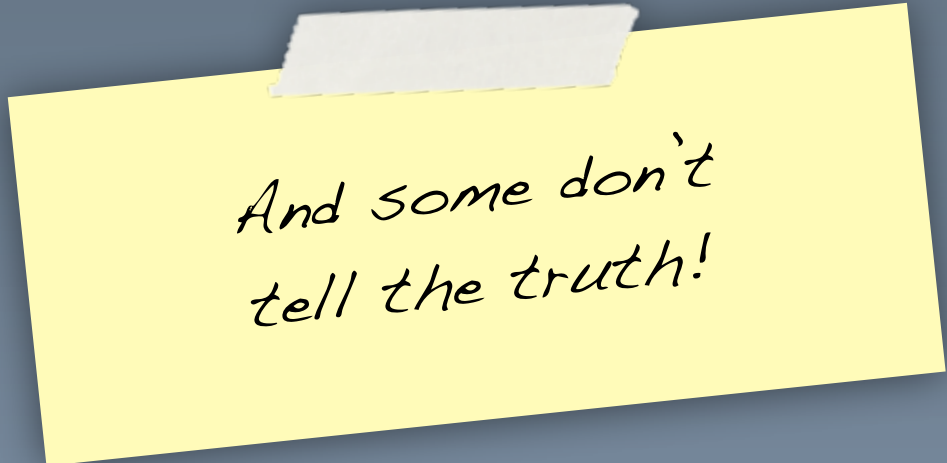
This shows good scalability because response times increase in a predictable way



Load testing in
the real world?

Few

software teams do any
sort of load testing



*And some don't
tell the truth!*

Even a little testing can increase

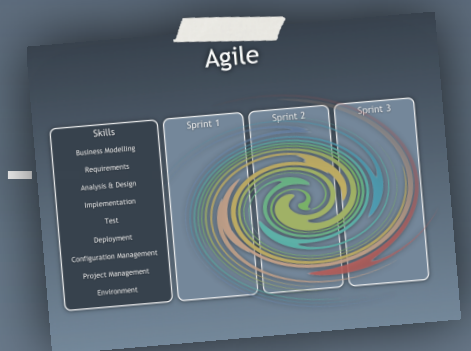
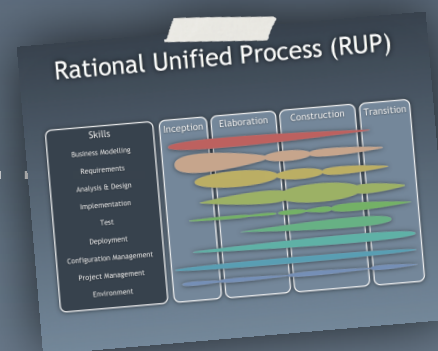
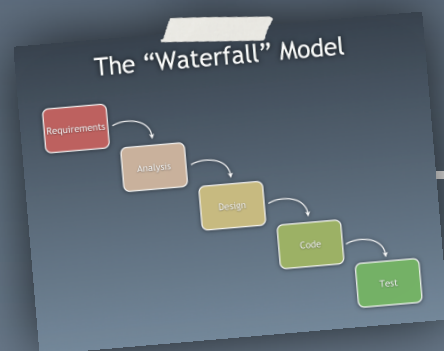
confidence,

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](http://www.agilemodeling.com/essays/agileArchitecture.htm)

Concrete experiments

Requirements

A system

Foundations

*A throwaway prototype/
proof of concept or
production code*

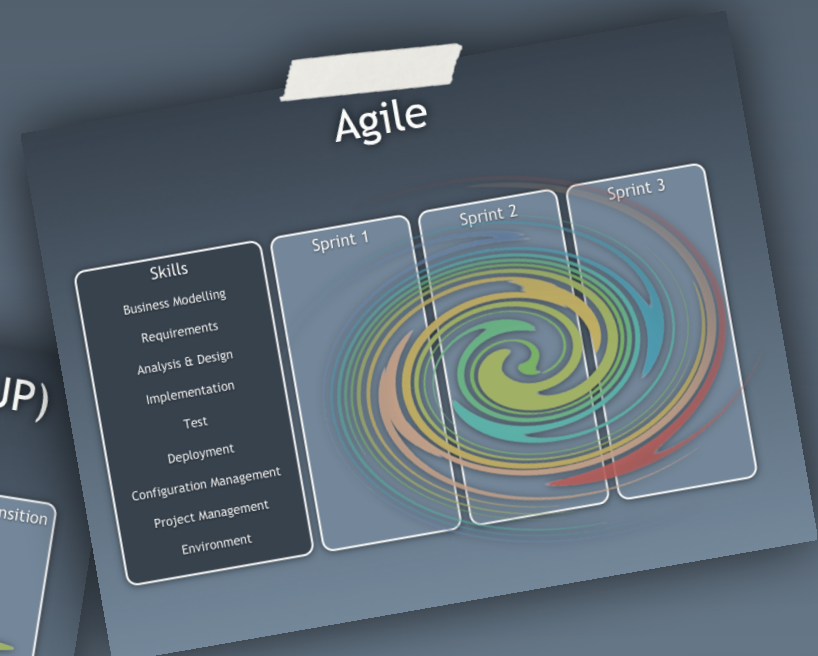
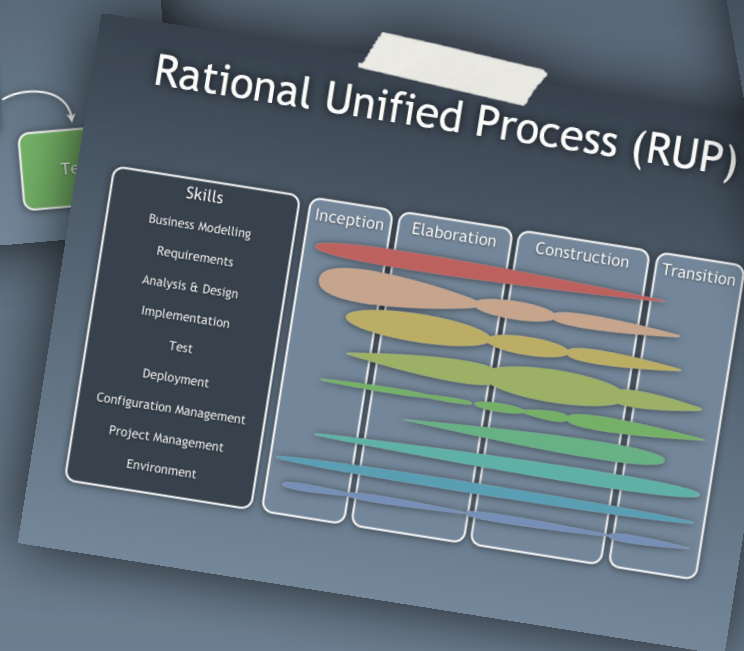
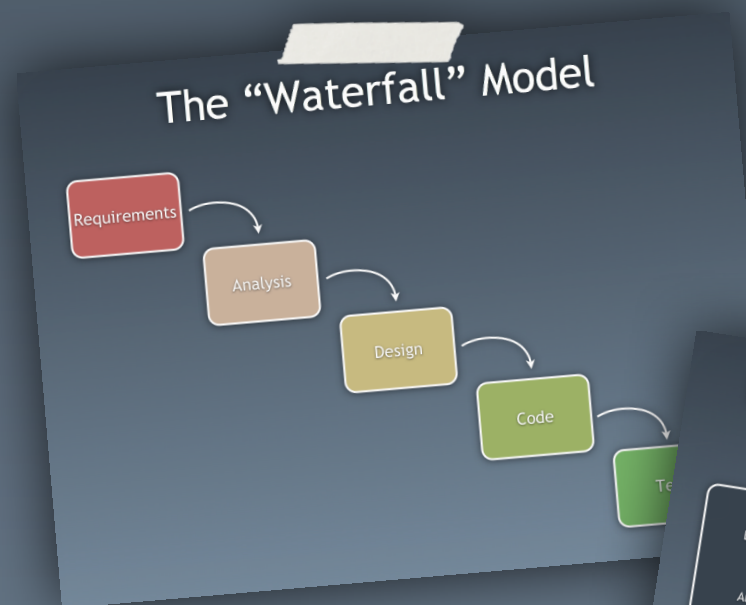
What is architecturally significant?

*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?

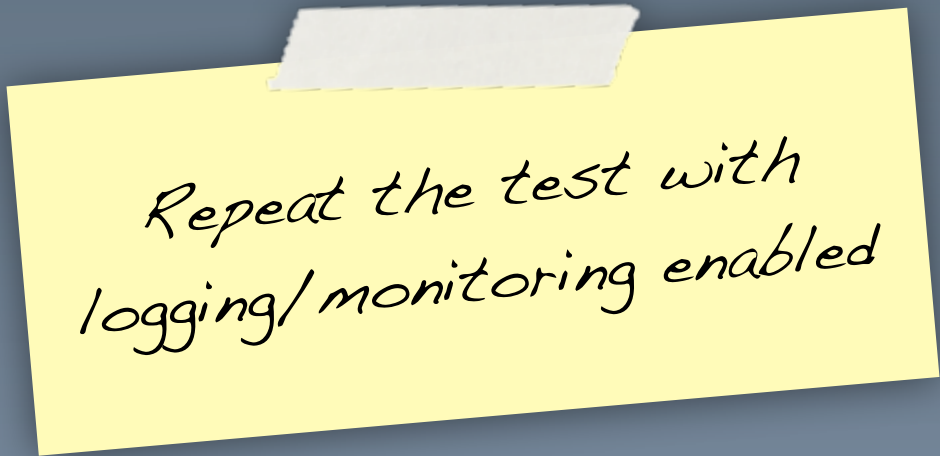
Look at your monitoring data
and log files,
run profilers,
etc...

Maybe the environment
needs **tuning**

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

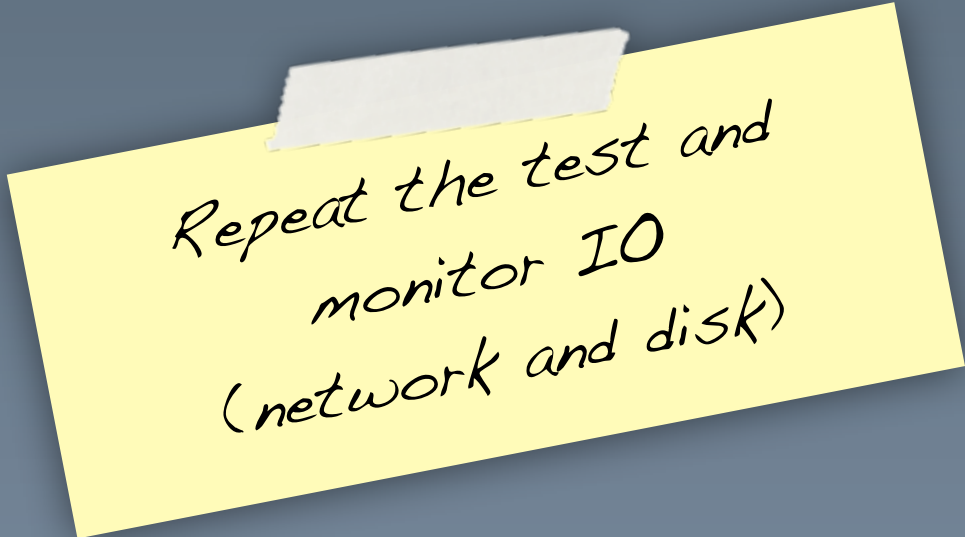
The **big** picture and
the **detail**
are equally as important

My application has
intermittent
poor performance



*Repeat the test with
logging/monitoring enabled*

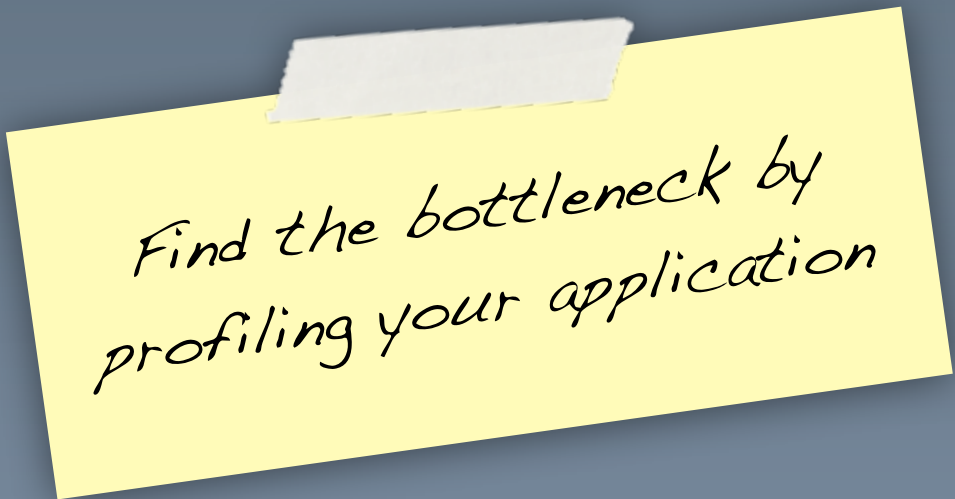
My application is slow, but it's
not using any CPU



*Repeat the test and
monitor IO
(network and disk)*

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?

The screenshot shows a terminal window with two overlapping windows. The background window displays Linux clock accuracy data, and the foreground window displays Windows clock accuracy data. A callout box points to the Windows data.

```
Terminal — bash — 97x32
raw=1263861049883 | formatted=19-Jan-2010 00:30:49:883 | frequency=16152 | delta=1ms
raw=1263861049884 | formatted=19-Jan-2010 00:30:49:884 | frequency=16154 | delta=1ms
raw=1263861049885 | formatted=19-Jan-2010 00:30:49:885 | frequency=16156 | delta=1ms
raw=1263861049886 | formatted=19-Jan-2010 00:30:49:886 | frequency=16158 | delta=1ms
raw=1263861049887 | formatted=19-Jan-2010 00:30:49:887 | frequency=16160 | delta=1ms
raw=1263861049888 | formatted=19-Jan-2010 00:30:49:888 | frequency=16162 | delta=1ms
raw=1263861049889 | formatted=19-Jan-2010 00:30:49:889 | frequency=16164 | delta=1ms
raw=1263861049890 | formatted=19-Jan-2010 00:30:49:890 | frequency=16166 | delta=1ms
raw=1263861049891 | formatted=19-Jan-2010 00:30:49:891 | frequency=16168 | delta=1ms
raw=1263861049892 | formatted=19-Jan-2010 00:30:49:892 | frequency=16170 | delta=1ms
raw=1263861049893 | formatted=19-Jan-2010 00:30:49:893 | frequency=16172 | delta=1ms
raw=1263861049894 | formatted=19-Jan-2010 00:30:49:894 | frequency=16174 | delta=1ms
raw=1263861049895 | formatted=19-Jan-2010 00:30:49:895 | frequency=16176 | delta=1ms
raw=1263861049896 | formatted=19-Jan-2010 00:30:49:896 | frequency=16178 | delta=1ms
raw=1263861049897 | formatted=19-Jan-2010 00:30:49:897 | frequency=16180 | delta=1ms
raw=1263861049898 | formatted=19-Jan-2010 00:30:49:898 | frequency=16182 | delta=1ms
raw=1263861049899 | formatted=19-Jan-2010 00:30:49:899 | frequency=16184 | delta=1ms
raw=1263861049900 | formatted=19-Jan-2010 00:30:49:900 | frequency=16186 | delta=1ms
raw=1263861049901 | formatted=19-Jan-2010 00:30:49:901 | frequency=16188 | delta=1ms
raw=1263861049902 | formatted=19-Jan-2010 00:30:49:902 | frequency=16190 | delta=1ms
raw=1263861049903 | formatted=19-Jan-2010 00:30:49:903 | frequency=16192 | delta=1ms
raw=1263861049904 | formatted=19-Jan-2010 00:30:49:904 | frequency=16194 | delta=1ms
raw=1263861049905 | formatted=19-Jan-2010 00:30:49:905 | frequency=16196 | delta=1ms
raw=1263861049906 | formatted=19-Jan-2010 00:30:49:906 | frequency=16198 | delta=1ms
raw=1263861049907 | formatted=19-Jan-2010 00:30:49:907 | frequency=16200 | delta=1ms
raw=1263861049908 | formatted=19-Jan-2010 00:30:49:908 | frequency=16202 | delta=1ms
raw=1263861049909 | formatted=19-Jan-2010 00:30:49:909 | frequency=16204 | delta=1ms
raw=1263861049910 | formatted=19-Jan-2010 00:30:49:910 | frequency=16206 | delta=1ms

Minimum delta : 1ms
Maximum delta : 1ms
simonbrown:Desktop simon$
```

```
C:\WINDOWS\system32\cmd.exe
C:\>java -jar clock-accuracy.jar
raw=1263860967794 | formatted=19-Jan-2010 00:29:27:794 | frequency=13994 | delta=0ms
raw=1263860967810 | formatted=19-Jan-2010 00:29:27:810 | frequency=753345 | delta=16ms

Minimum delta : 16ms
Maximum delta : 0ms
C:\>_
```

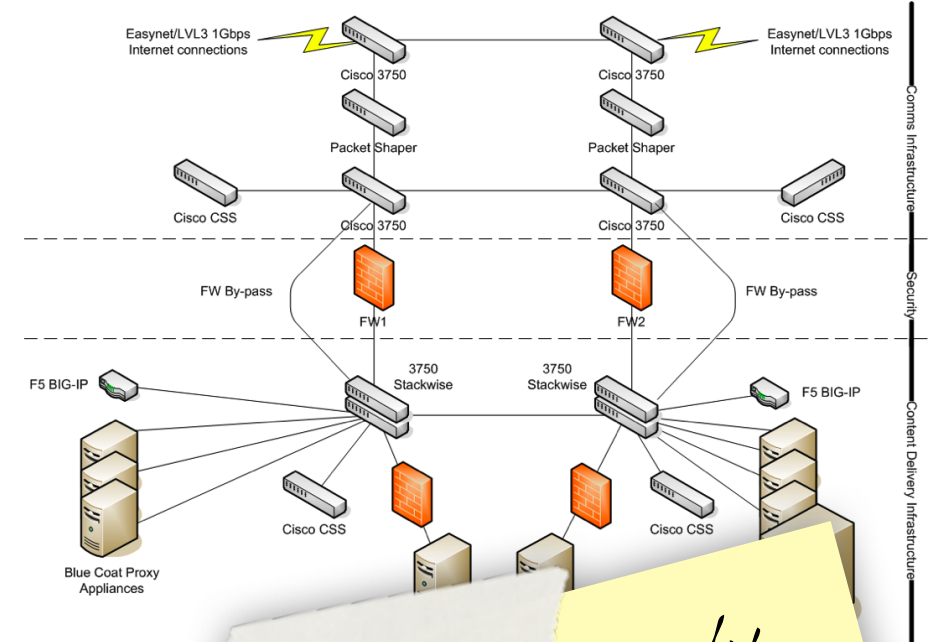
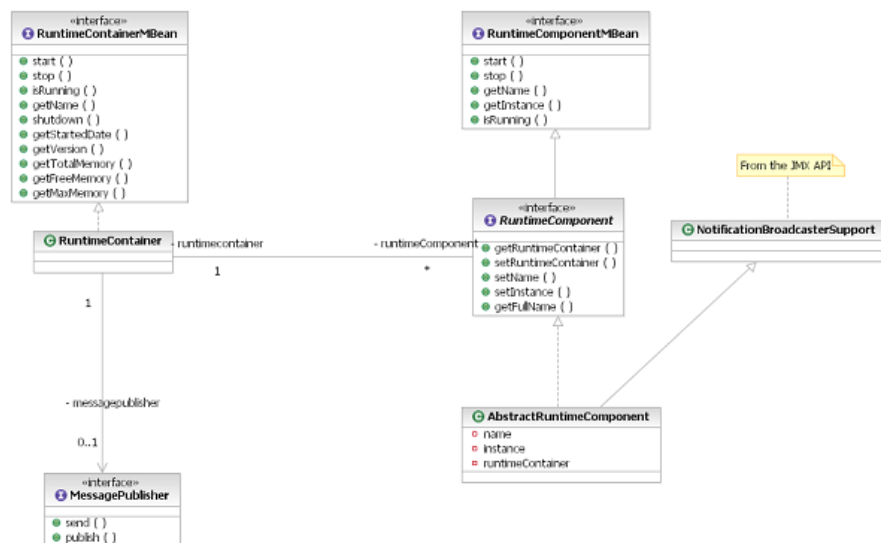
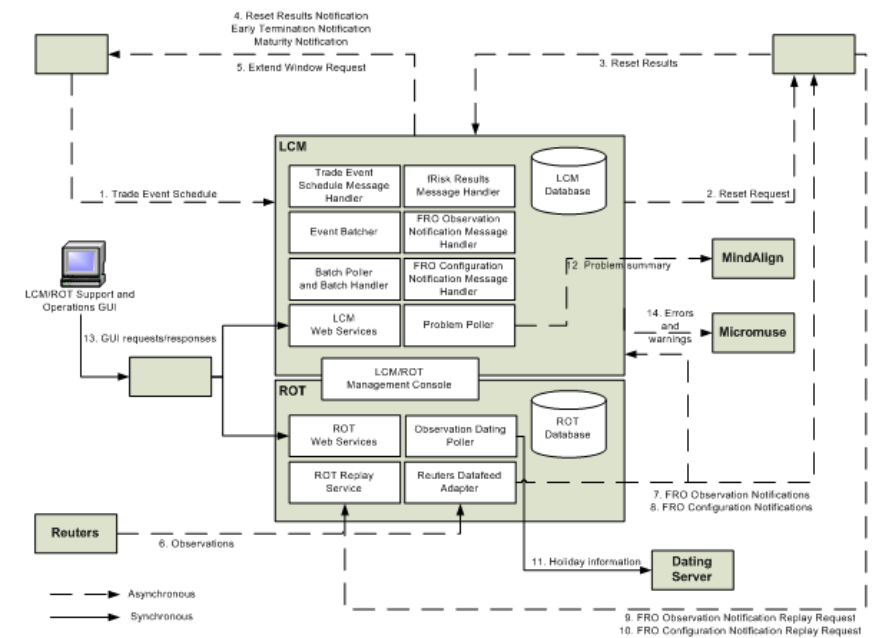
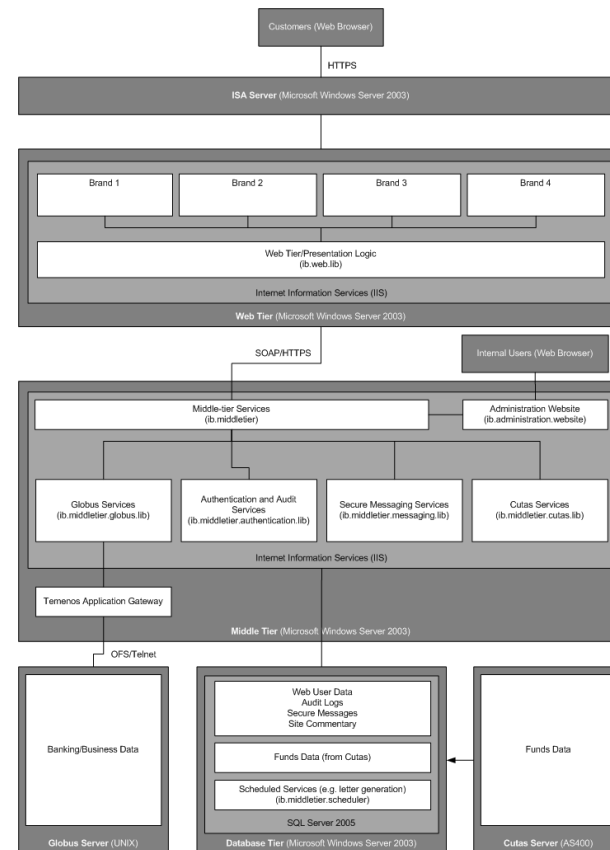
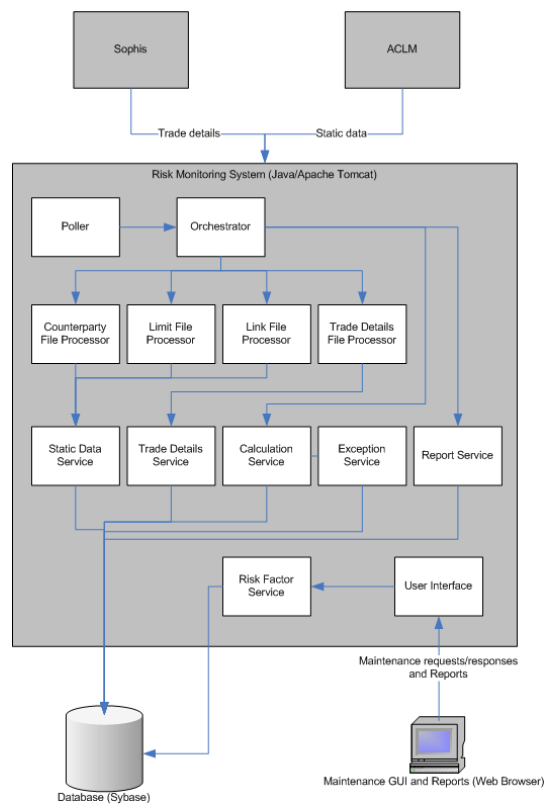
16 milliseconds on Windows

Clock granularity
can vary by platform

What's the
time?

(are you sure?)

Let's wrap up



You can only guess by looking at diagrams or source code

Will these software systems perform and scale acceptably?

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

