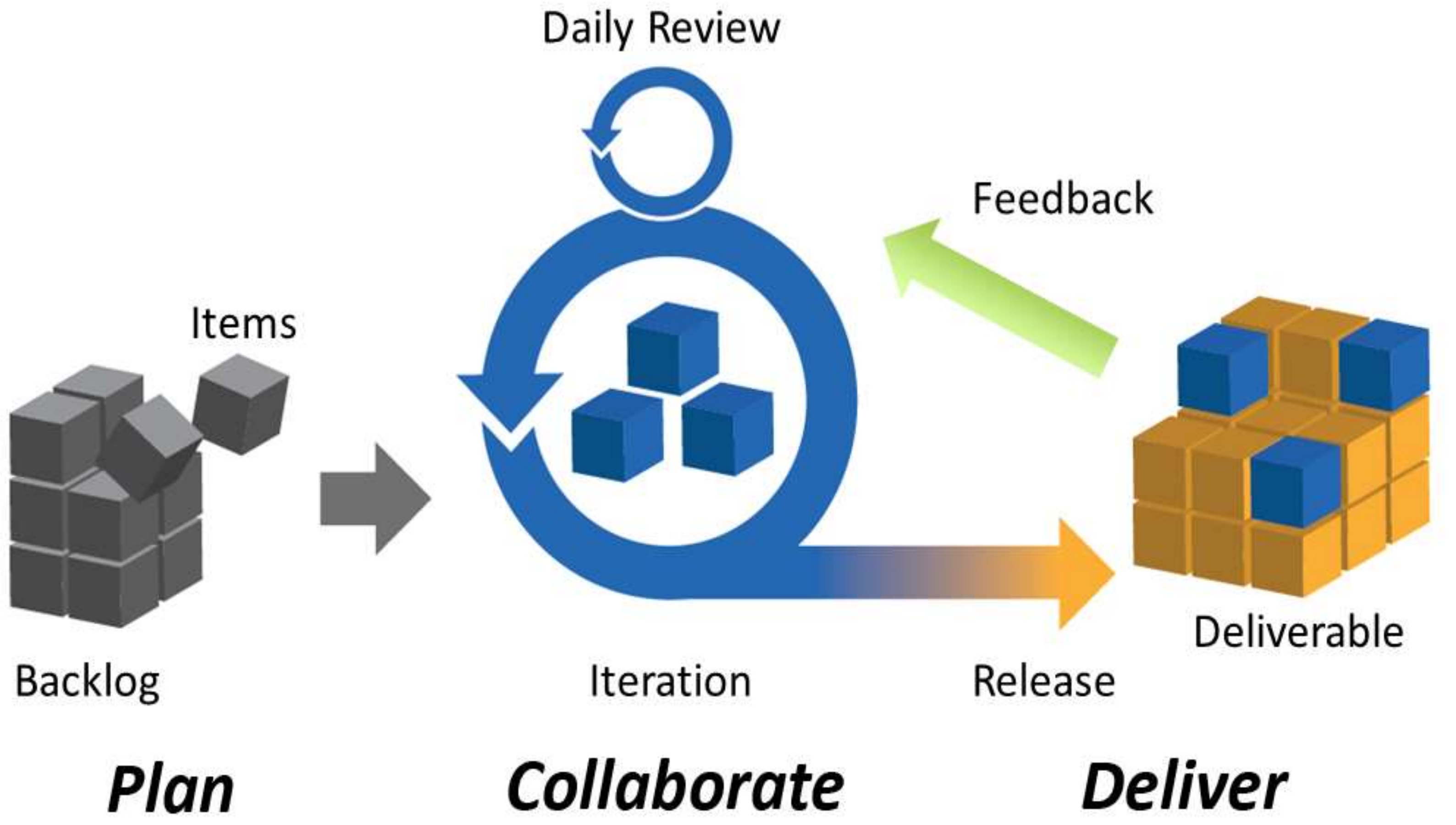
August 11 - 13, 2014
New York, NY, USA
Association for Software Testing
ast
CAST
"The Art and Science of Testing"

Early Performance Testing: News You Can Use

* Eric Proegler, @ericproegler, testingthoughts.com/ericproegler
* 12 years in performance, 19 in software
* WOPR Organizer (Workshop on Performance and Reliability, performance-workshop.org)
* This subject was discussed at WOPR22.

  WOPR22 was held May 21-23, 2014, in Malmö, Sweden, on the topic of "Early Performance Testing." Participants in the workshop included: Fredrik Fristedt, Andy Hohenner, Paul Holland, Martin Hynie, Emil Johansson, Maria Kedemo, John Meza, Eric Proegler, Bob Sklar, Paul Stapleton, Andy Still, Neil Taitt, and Mais Tawfik Ashkar.
* Questions about Peer Workshops? Get at me.



Agile Project Management: Iteration

Testing Iterations?

Testing Iterations?

- No Process-teering/Scrumbaggery here
- Testing in Agile is about providing quick feedback. How can we make performance and scalability feedback happen faster – or at all, with incomplete/unfinished systems, on non-production hardware?

* What are the risks we are testing for?
* What is “Realistic”?
* Testing Techniques in Iterative Projects
* Reporting Early Performance Results
* Performance Testing Incomplete Systems

Ideas

Performance Risks: Scalability

Scalability: Expensive operations mean that systems won't scale well

- Ops Problem? Solve with hardware?
- Tall Stacks -> Wirth's Law: "Software is getting slower more rapidly than hardware gets faster"
- Subject to use patterns and user models

What Does a Problem Look Like?

- Longer response times is a clue
- "High" CPU/Memory/Storage/Network Utilization

Capacity: System can’t support the expected load structurally/as engineered

What Does a Problem Look Like?

- Response time very sensitive to load
- Growing Queues
- Hard or Soft Resource Limitations
  - High CPU Limitation
  - Increasing I/O Latency
  - Run out of database threads

Performance Risks: Capacity

Concurrency: Operations that contend and collide (Race conditions, database locks, contention points)

What Does a Problem Look Like?

- Infrequent functional issues that seem to only occur under load (Heisenbugs?)
- Process crashes
- Not easily reproducible

Performance Risks: Concurrency

Performance Risks: Reliability

Reliability: Degradation over time, system becomes slower, less predictable, or eventually fails.

What Does a Problem Look Like?

- Memory or Object Leaks
- More frequent Garbage Collection
- Decaying response times

* What are the risks we are testing for?
* What is “Realistic”?
* Testing Techniques in Iterative Projects
* Reporting Early Performance Results
* Performance Testing Incomplete Systems

Ideas

Will the completed, deployed system support:

(*a*, *b*…) users
performing (*e*, *f*…) activities
at (*j*, *k*…) rates
on *mn*… configuration
under *rs*… external conditions,
meeting *x*, *y*… response time goals ?

(Simulation Test)

Illusion of Realism: Experiment

Illusion of Realism: Models

"All Models are wrong. Some are useful."

- Guesses at activities, and frequencies
- Organic loads and arrival rates – limitations imposed by load testing tools
- Session abandonment, other human behaviors
- Simulating every activity in the system
- Data densities (row counts, cardinality)
- Warmed caching
- Loads evolving over time

Illusion of Realism: Environment

"The environment is identical."

* Shared resources: Virtualization, SANs, Databases, Networks, Authentication, etc
* Execution environment versions and patching
* Software and hardware component changes, versions and patching
* Variable network conditions, especially last-mile
* Background processing and other activities against overlapping systems and resources

* What are the risks we are testing for?
* What is “Realistic”?
* Testing Techniques in Iterative Projects
* Reporting Early Performance Results
* Performance Testing Incomplete Systems

Ideas

* Simulation Testing occurs at the end of the project, before go live. If we find problems then, they are either bad enough to delay the whole project…

  …or they are “Deferred to Phase 2”

* Simulation Tests are Expensive to Create, Maintain, and Analyze

Simulation Tests

Simulation Tests

How does "Realistic" translate to agile?

* Maybe the answer is more control?
* Horizontal Scalability makes assumptions – let's use them
* Test subsets: single servers, single components, cheaply and repeatedly
* Calibration tests

Cheap and Fast Performance Tests

Build easily repeatable, reliable rapid tests:

* Login (measure session overhead/footprint)
* Simple workflows, avoid data caching effects
* Control variables
* Be ready to repeat/troubleshoot when you find anomalies

Who needs “Real”? Let’s find problems.
Examples:

- Burst loads – Ready, Set, GO!!!
- 10 threads, 10 iterations each
- 1 thread, 100 iterations
- 1 or 10 threads, running for hours/days

Cheap and Fast Performance Tests

Why must we have a "comparable" environment?

- Leverage horizontal scalability – one server is enough
- Who says we have to have systems distributed the same way? Why can't my test database be on this system for calibration purposes?
- Isolation is more important than "Real"

Cheap and Fast Performance Tests

Cheap and Fast Performance Tests

* Check Your Instruments!

* Calibrate, and Recalibrate as necessary between environments, builds, day/times…any time you want to be sure you are comparing two variables accurately

Now that the cost of a test is low, let's run lots

* Run five tests and average the results
* Run tests every day
* Run tests in Continuous Integration?

Cheap and Fast Performance Tests

* It’s being done, though it’s not easy to automate all of the moving parts (code deployment, system under test including data state, consumable data, etc)
* SOASTA and others can show you where they are using it
* Anyone here doing this?

Performance Testing in CI

* Stopwatch?
* Screen Captures? Videos?
* Waterfall Charts? Fiddler?
* Use Background Load and take measurements?

Sapient Techniques

* Use/extend automation – Add timers, log key transactions, build response time records
* Watch for trends, be ready to drill down. Automation extends your senses, but doesn’t replace them

Extending Automation

* What are the risks we are testing for?
* What is “Realistic”?
* Testing Techniques in Iterative Projects
* Reporting Early Performance Results
* Performance Testing Incomplete Systems

Ideas

Results Lead to Action

* Data + Analysis = Information
* Your value as a tester is the information you provide – and the action(s) it inspires
* Be willing to sound an alarm – or approach a key engineer: “Can I show you something?”

Communicate Outwards

Start your Campaign: Make results visible

* Emails
* Intranet page
* Whiteboard
* Status reports
* Gossip: “How is performance these days?”

Recruit Consumers! Become an important project status indicator. Help people who can do something understand and care

Date	Build	Min	Mean	Max	90%
2/1	9.3.0.29	14.1	14.37	14.7	14.6
3/1	***9.3.0.47***	***37.03***	***38.46***	***39.56***	***39.18***
8/2	9.5.0.34	16.02	16.61	17	16.83
9/9	10.0.0.15	17.02	17.81	18.08	18.02
10/12	10.1.0.3	16.86	17.59	18.03	18
11/30	10.1.0.38	18.05	18.57	18.89	18.81
1/4	10.1.0.67	18.87	19.28	19.48	19.46
2/2	10.1.0.82	18.35	18.96	19.31	19.24

Calibration Results for Web Login, Search, View. Burst 10 threads iterating 10 times

Reporting Example

* What are the risks we are testing for?
* What is "Realistic"?
* Testing Techniques in Iterative Projects
* Reporting Early Performance Results
* Performance Testing Incomplete Systems

Ideas

- APIs – abstraction points for developers, less brittle than interfaces
- Mocking/Stubs/Auto-Responders – built in abstraction points to simulate other components that might not be there yet. Will help Devs test, too.

Testing Components – Missing Pieces

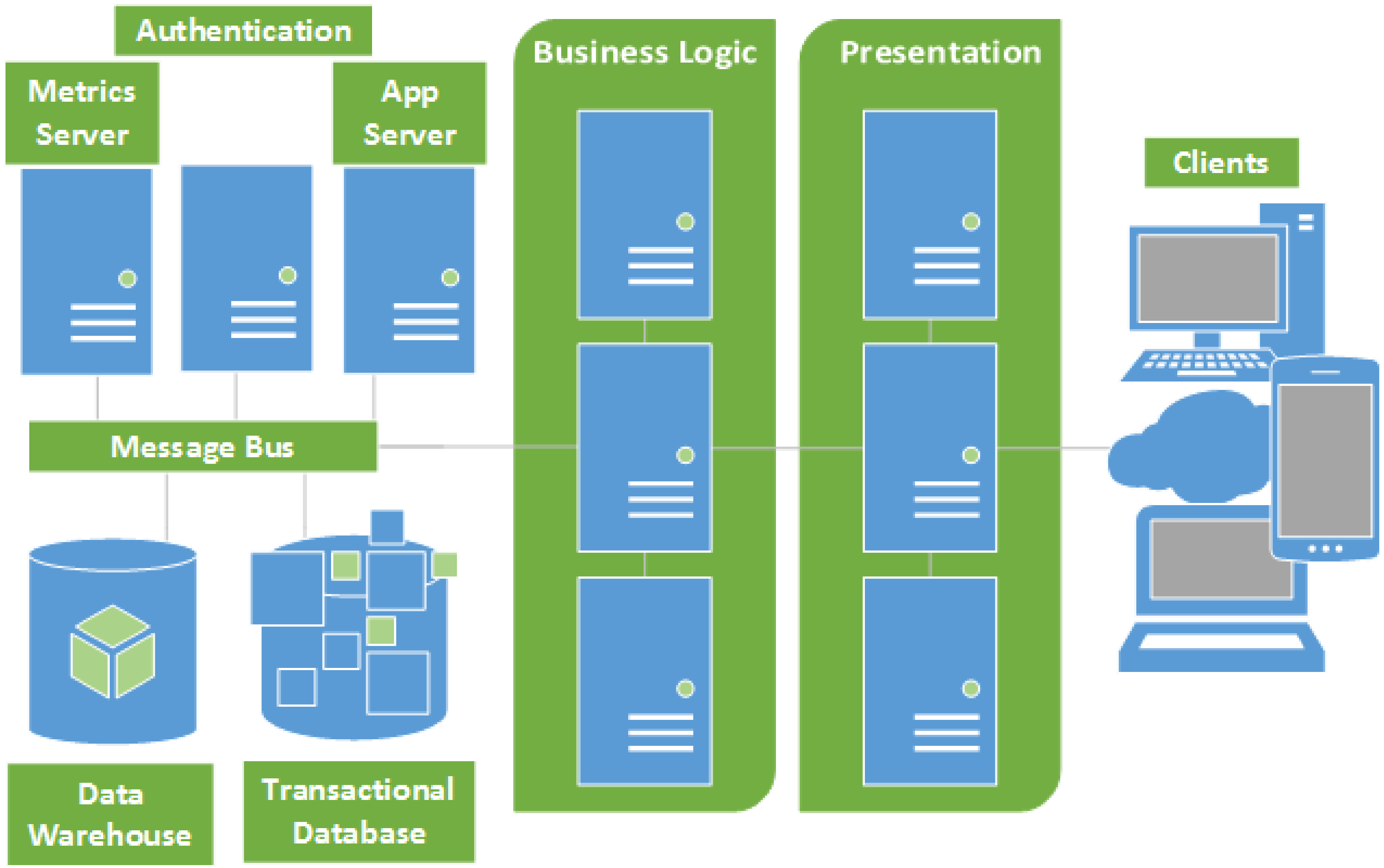
* Find and re-purpose code for harnesses. How do devs check their work?
* Mocking/Stubs/Auto-Responders – built in abstraction points to simulate other components that might not be there yet. Will help Devs test, too.

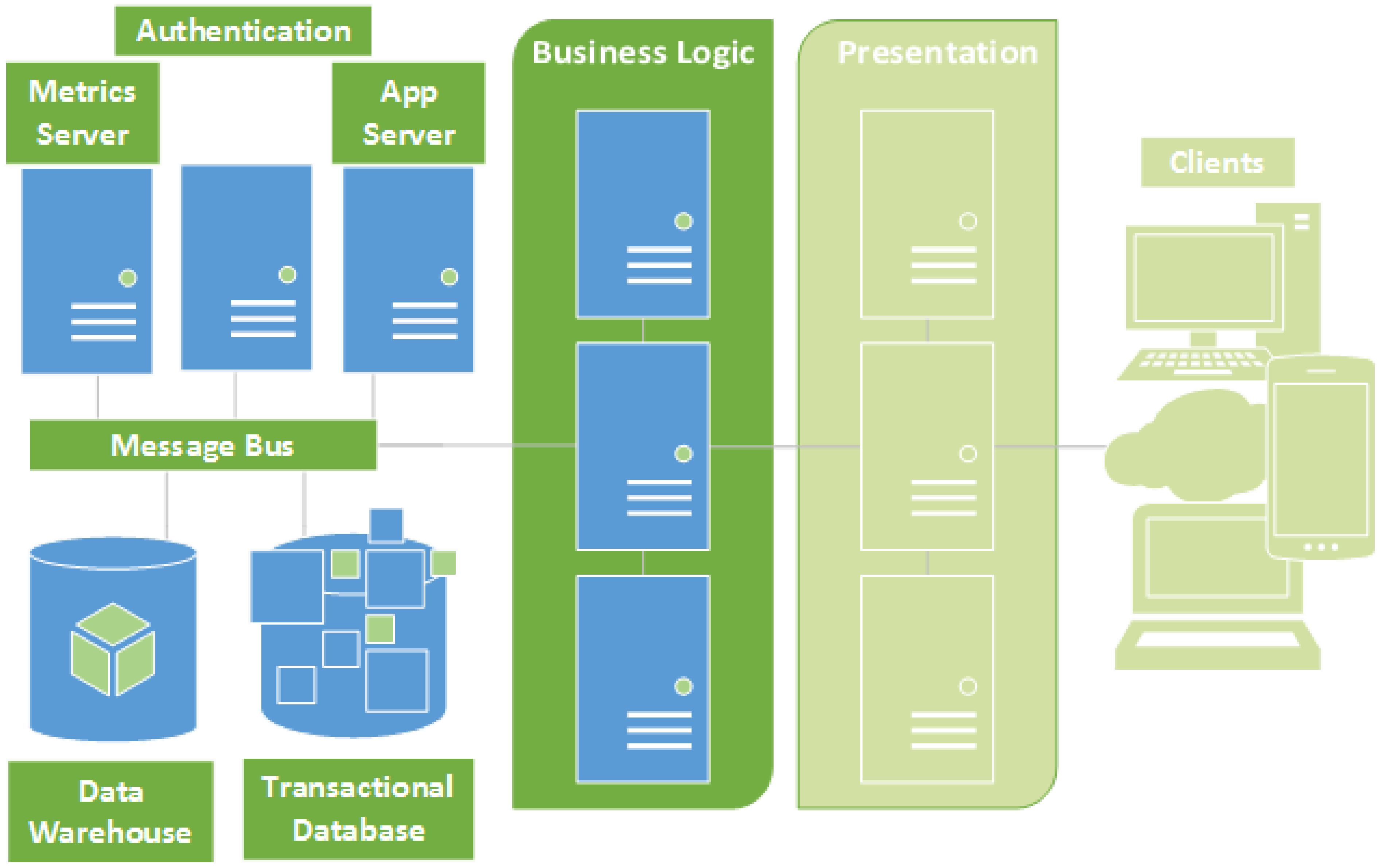
Testing Components – Individual Pieces

Testing Components

Test what is there:

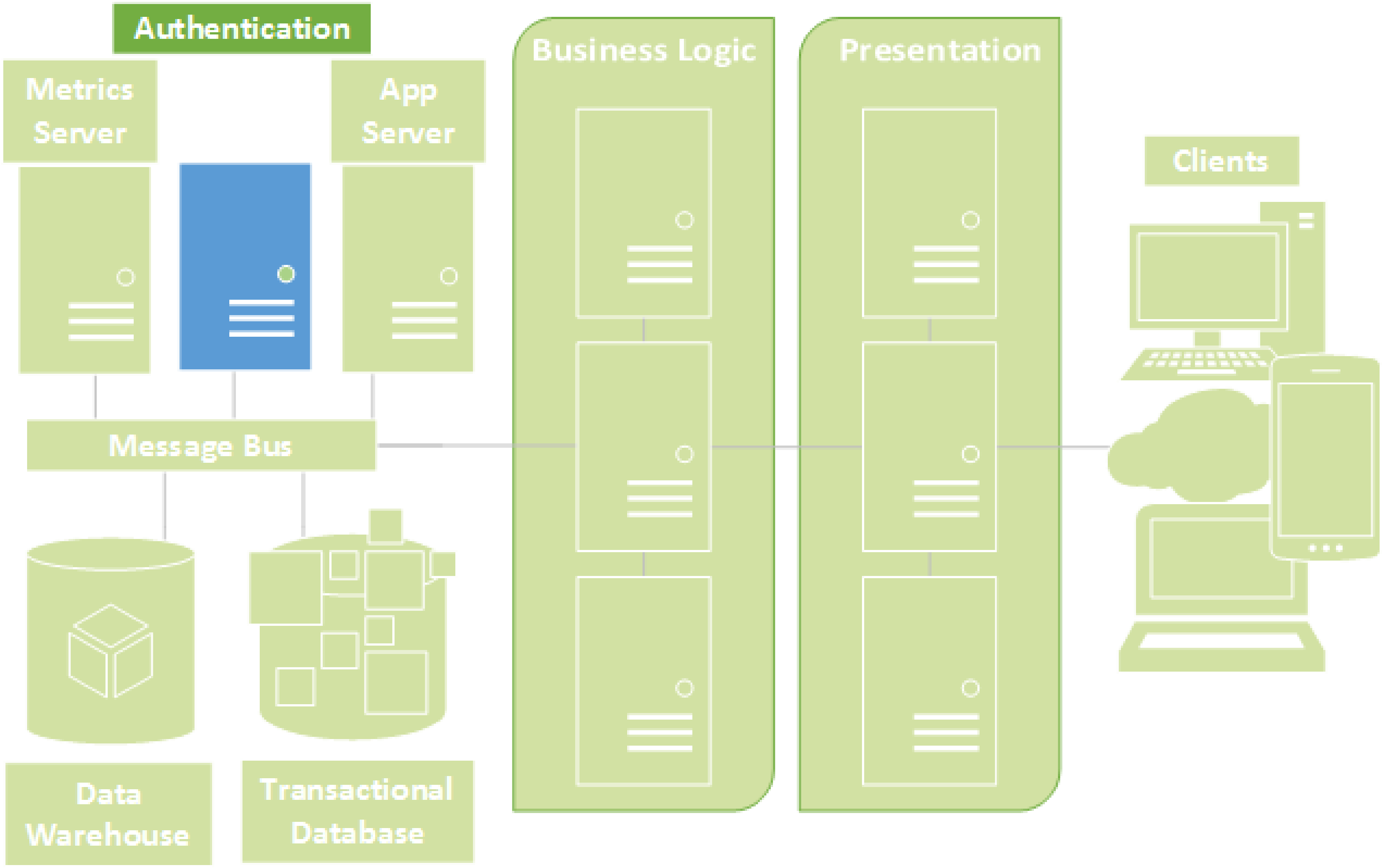
- Session models: login/logoff are expensive, and matter for everyone
- Search Functions
- Think about user activities like road systems – where are the highways?
- What will be meaningful to calibrate?


Authentication
Metrics Server
App Server
Business Logic
Presentation
Clients
Message Bus
Data Warehouse
Transactional Database


Authentication
Metrics Server
App Server
Business Logic
Presentation
Clients
Message Bus
Data Warehouse
Transactional Database

Incomplete Systems:Business Logic

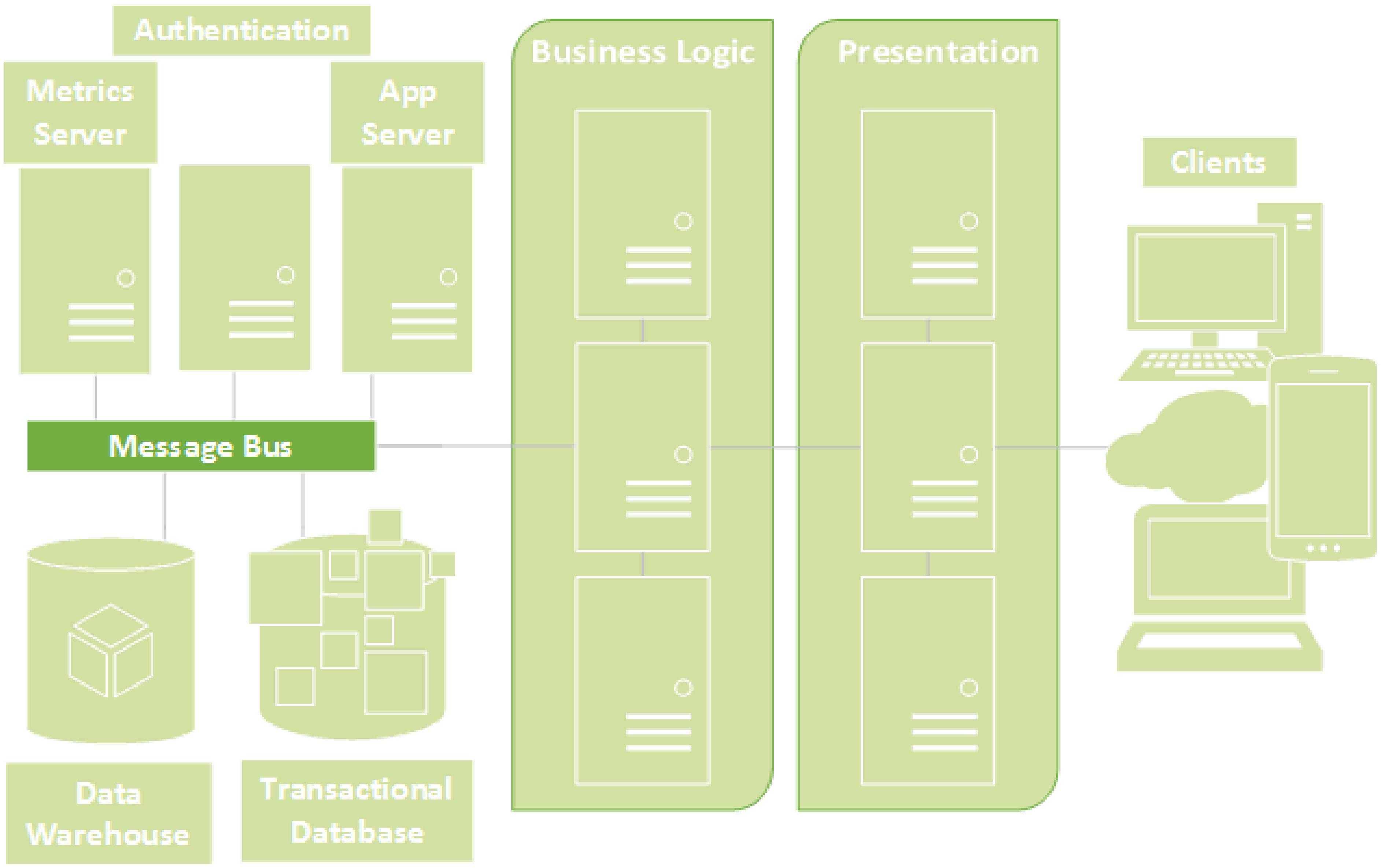
- Frequently the scaling bottleneck in this architecture pattern. Web services are big buckets of functionality usually running a lot of interpreted code
- Abstract presentation layer, borrow code from it, test the rest of the system down
- Script load tests that address web services directly


Authentication
Metrics Server
App Server
Business Logic
Presentation
Clients
Message Bus
Data Warehouse
Transactional Database

Incomplete Systems: Authentication

Create Test that just creates sessions

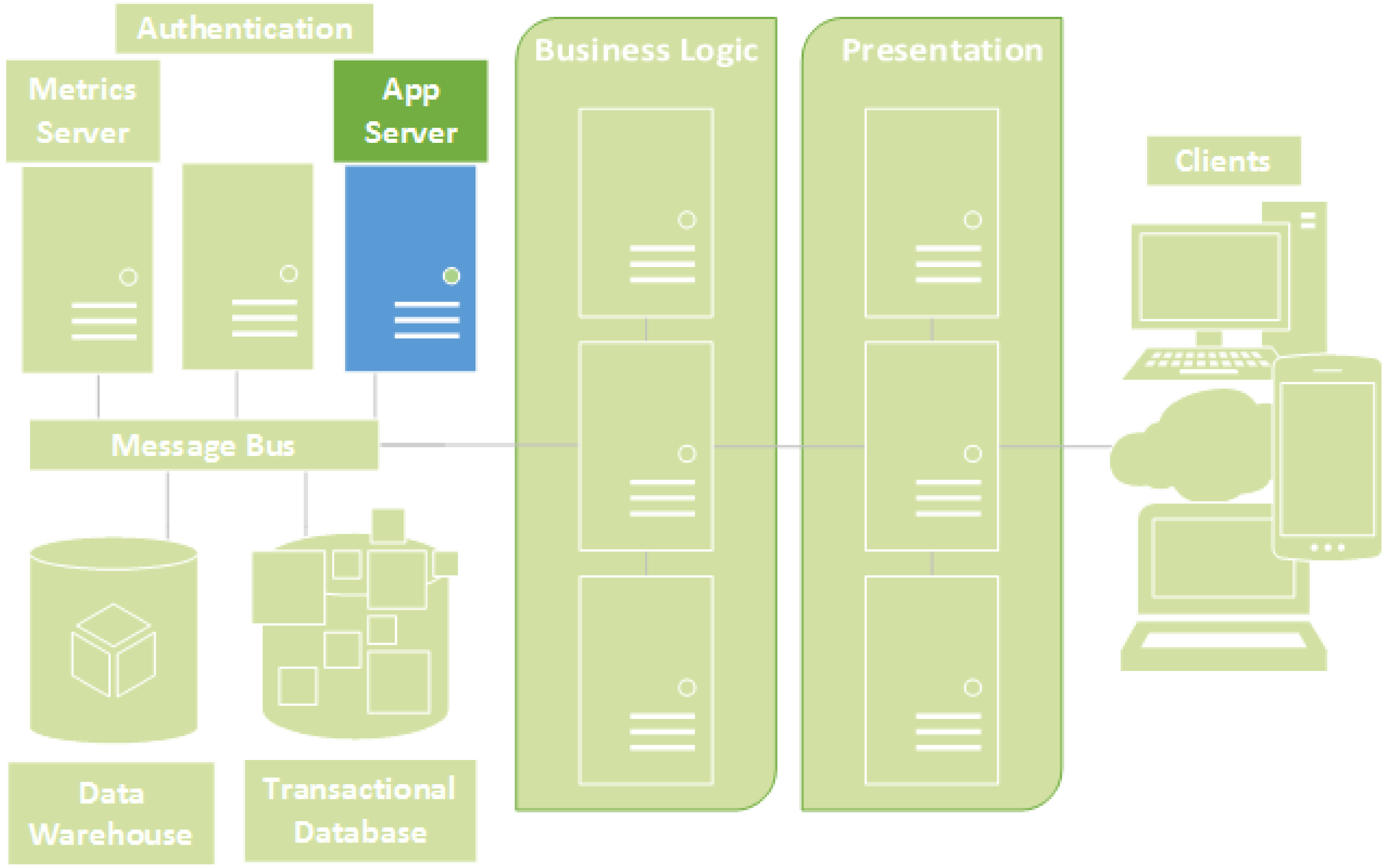
- Response time to create a session?
- How reliably can I create sessions?
- How many sessions can I create?
- What arrival rate can I support?


Authentication
Metrics Server
App Server
Business Logic
Presentation
Clients
Message Bus
Data Warehouse
Transactional Database

Find or Make a Test Harness. How do Developers Test?

* Response time to push a message?
* Response time to pull a message?
* Is this sensitive to load?
* At what rate can I create messages? With one publisher? With multiple publishers?
* At what rate can I consume messages? What if I add message servers/dispatchers?

Incomplete Systems: Message Bus


Authentication
Metrics Server
App Server
Business Logic
Presentation
Clients
Message Bus
Data Warehouse
Transactional Database

Incomplete Systems: App Server

Identify inputs – polling file? Job creation?

- Processing time for asynchronous processes – per item, startup time
- Throughputs
- How many processes can I queue up? What happens if I send a lot of synchronous stuff all at once?

Some Conclusions?

- What are the risks you are testing for?
- Unpack “Realistic”: what does it mean, and what (who) is it for?
- Consider the value of performance tests that are cheap and easily repeatable
- Broadcast Results
- Test What You Have
- Where are the highways?

* Tweet #CAST2014. You’re on Twitter, right?
* Join AST
* Come to #CAST2015
* Profit!!!

Thanks