

Testing Strategies for Monoliths and Microservices



Grok

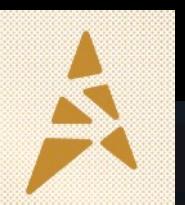
@DavidDLucas

L_{S_E}

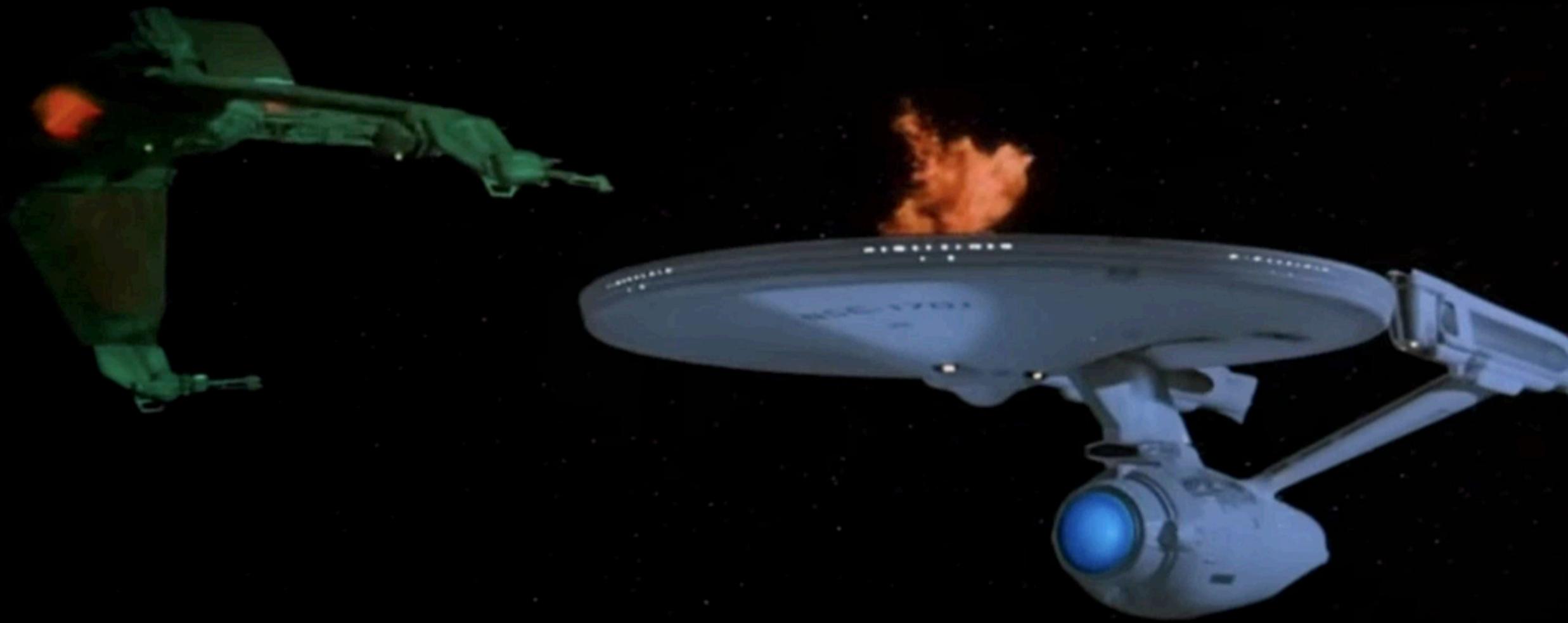


Kobayashi Maru

- How can I talk about testing at Stir Trek without talking about the greatest test scenario to ever hit the theater screen ?
- Captain Kirk vs No Win Scenario



Star Trek Kobayashi Maru



Pause (k)

Forged By Light

<https://tinyurl.com/st25-kobay>

L
S
E



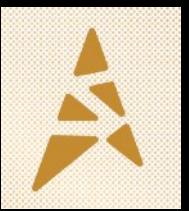
Lessons Learned From Kirk

- Testing Wrecks Havoc
- Testing should not be a no win situation
- Testing requires thinking outside the box
- Reprogram the simulation
(use Mocks and Stubs for the results you want)
- Happy Path (expected results)
- UnHappy Path (creative ingenuity for edges)
- No one likes to lose: create quality tests



Agenda

- Review Monoliths / Microservices
 - Common Patterns
 - Introduction to Testing
 - Unit Testing
 - Integration Testing
 - End-to-End
 - Other Topics
-
- parting gifts



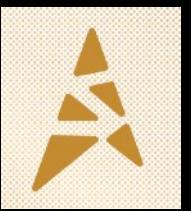
What are Monoliths

(mono=one lith=stone)



Uluru or Ayers Rock, Australia
largest monolith in the world
1,142 foot high and 5.8 miles circumference

L_S_E



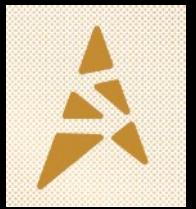
What about Polyliths?

(poly=many lith=stone)

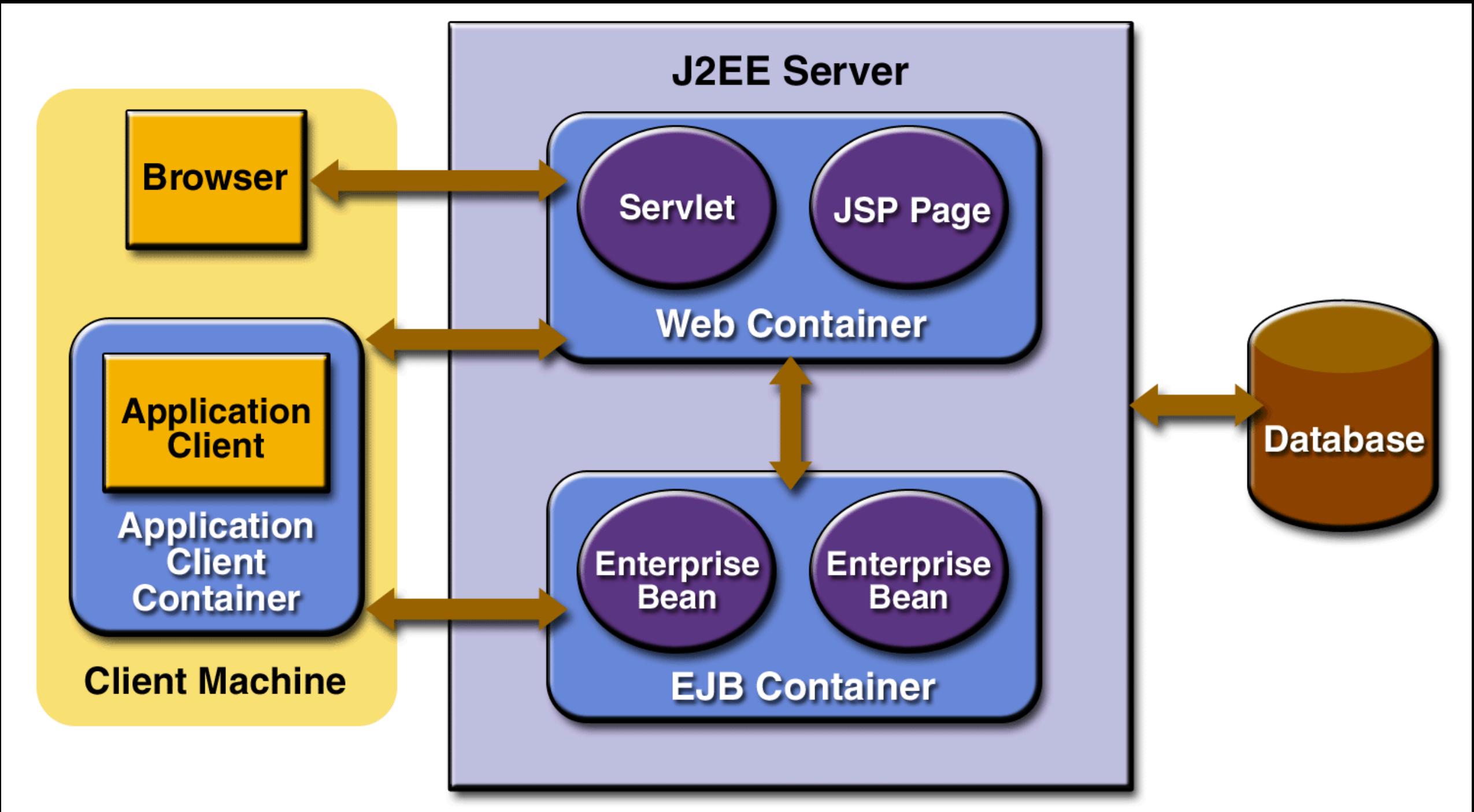


Stonehenge, prehistoric monument, Wiltshire, England,
outer ring of stones, each around 13 feet high by 7 feet
wide, weighing 25 tons

L
S
E

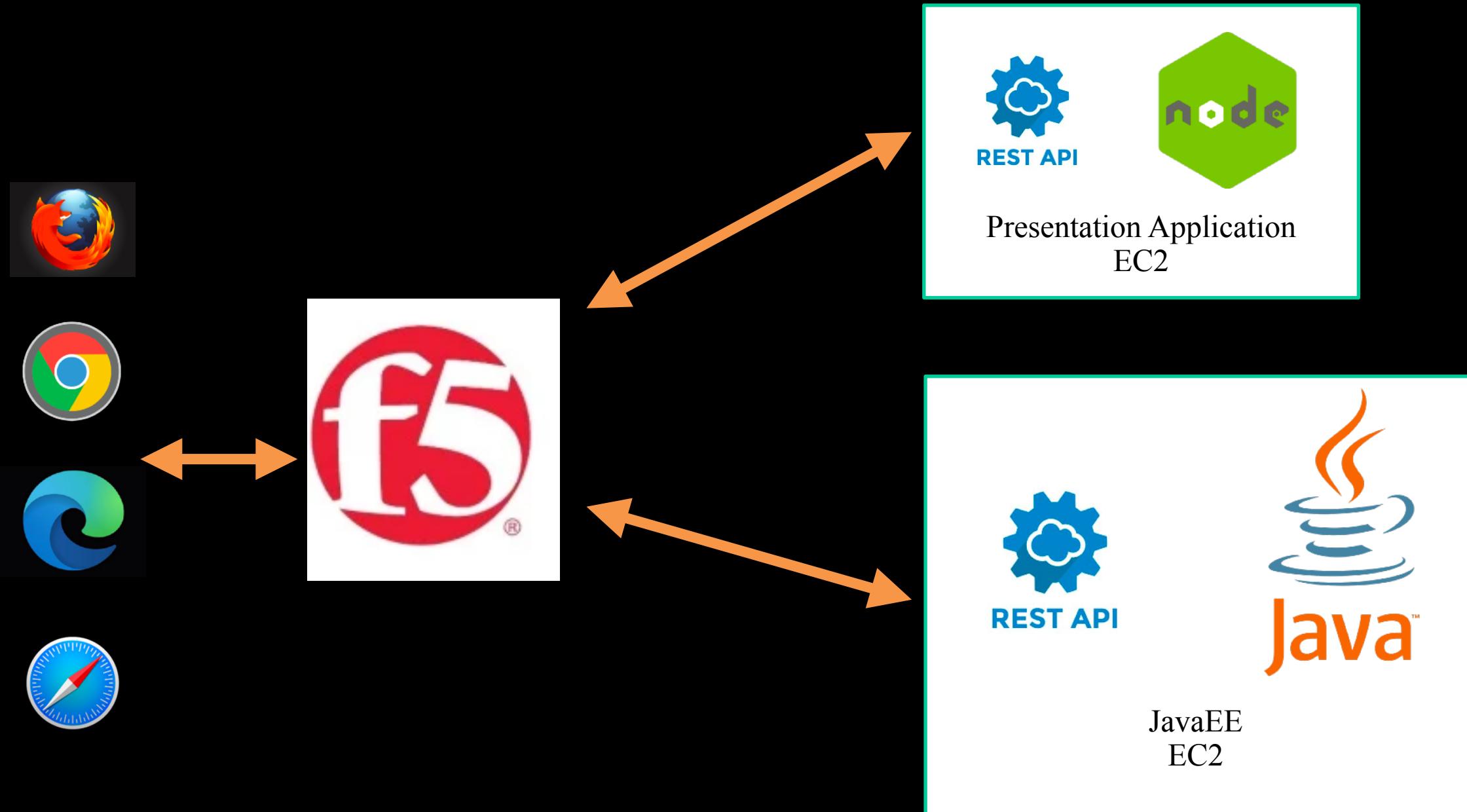


Monoliths (Software)



L
S
E

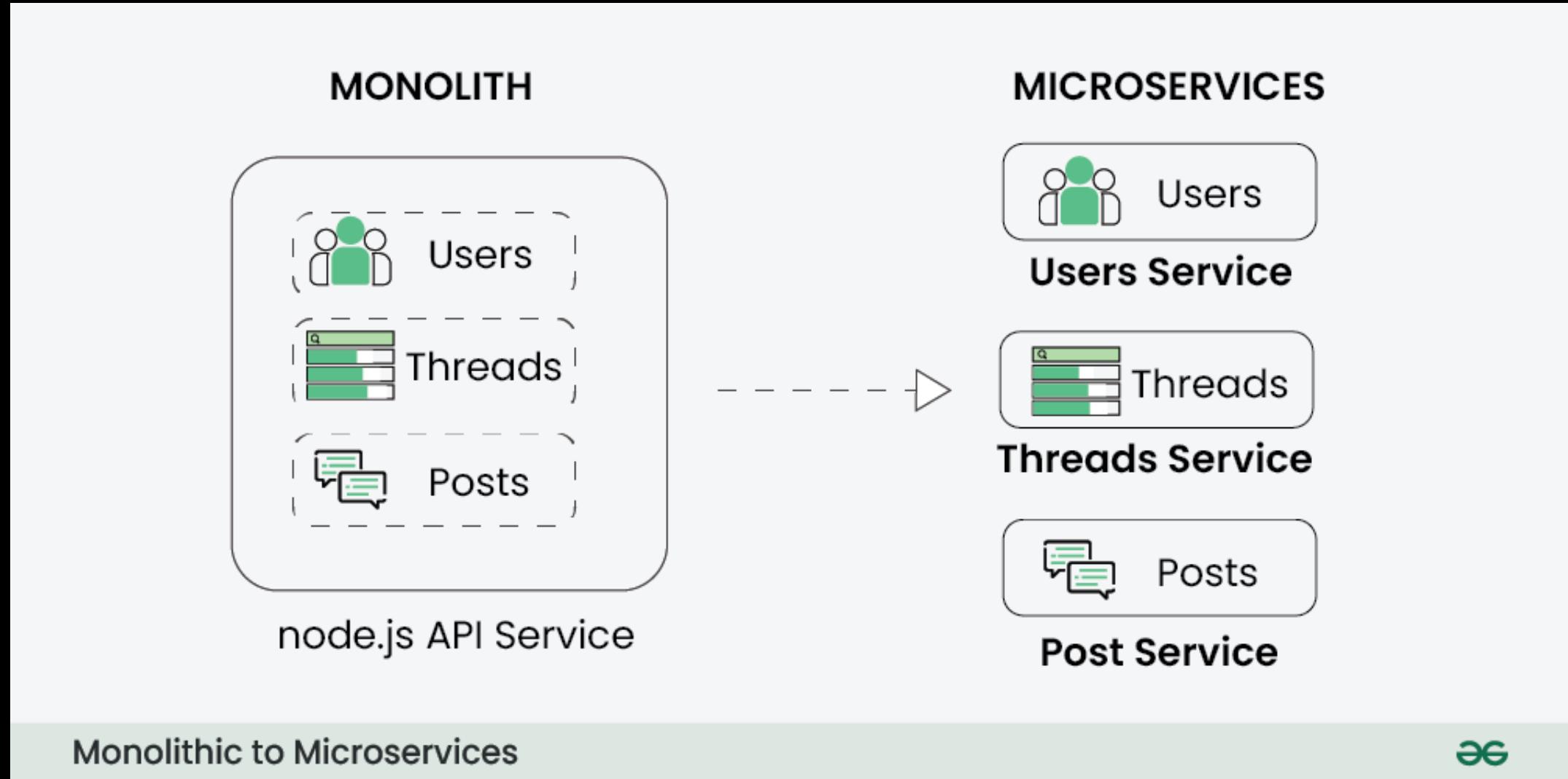
Monolith Flow



L
S
E



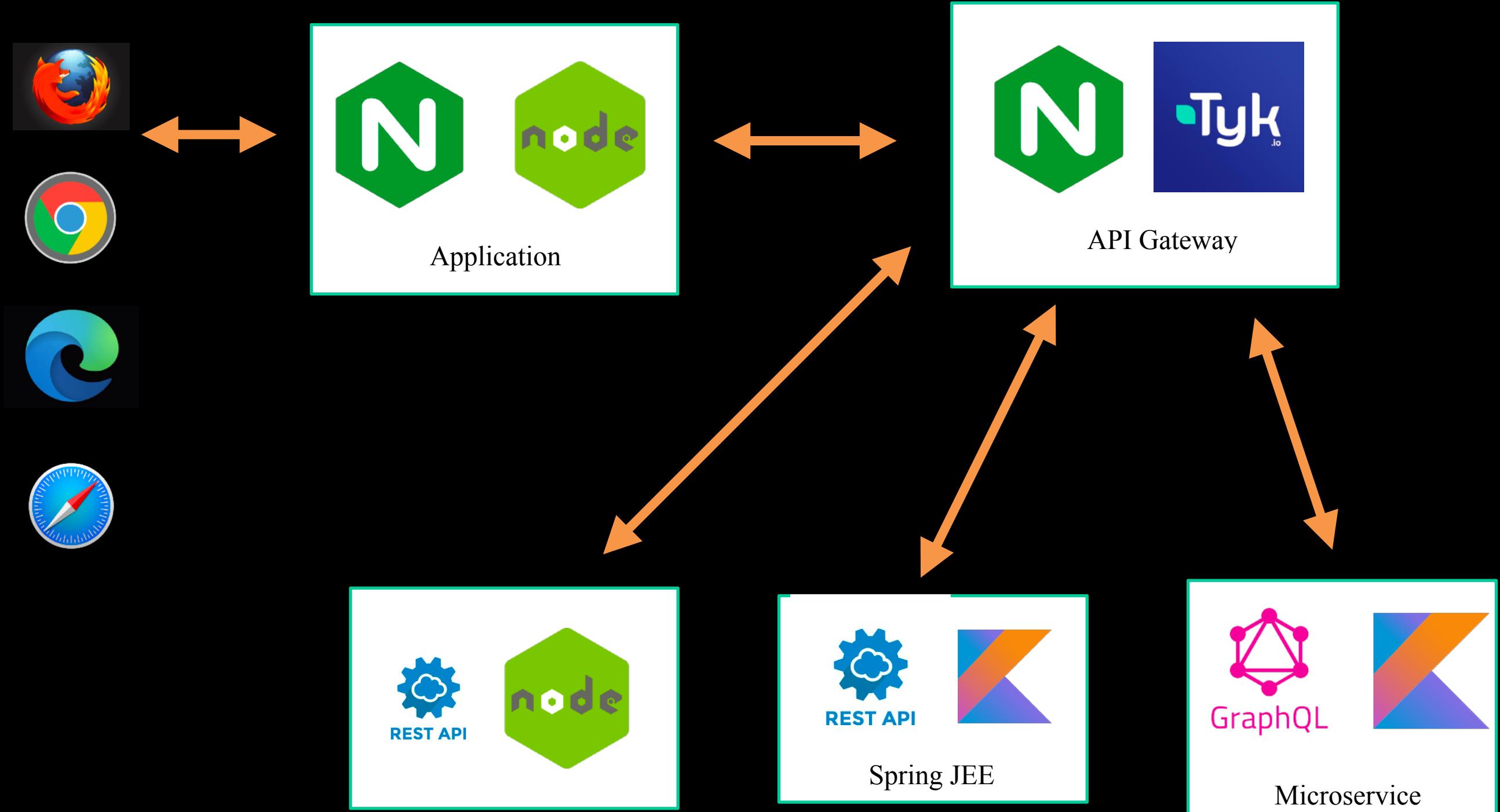
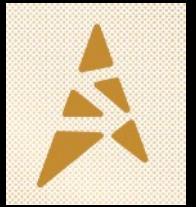
Example Blog Breakup



Monolith to Microservice
<https://www.geeksforgeeks.org/microservices/>

L
S
E

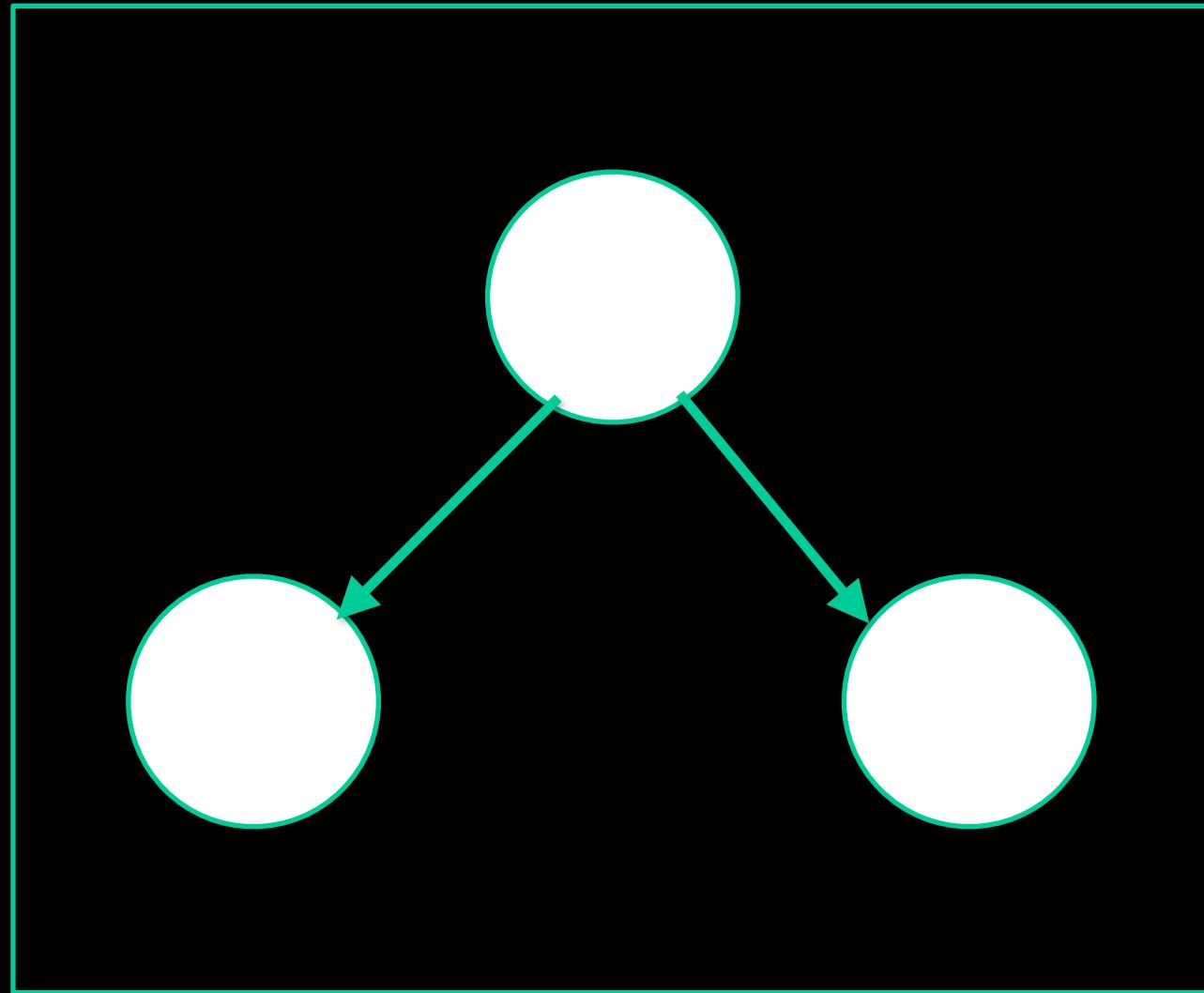
Microservice Flow



L
S
E



Patterns

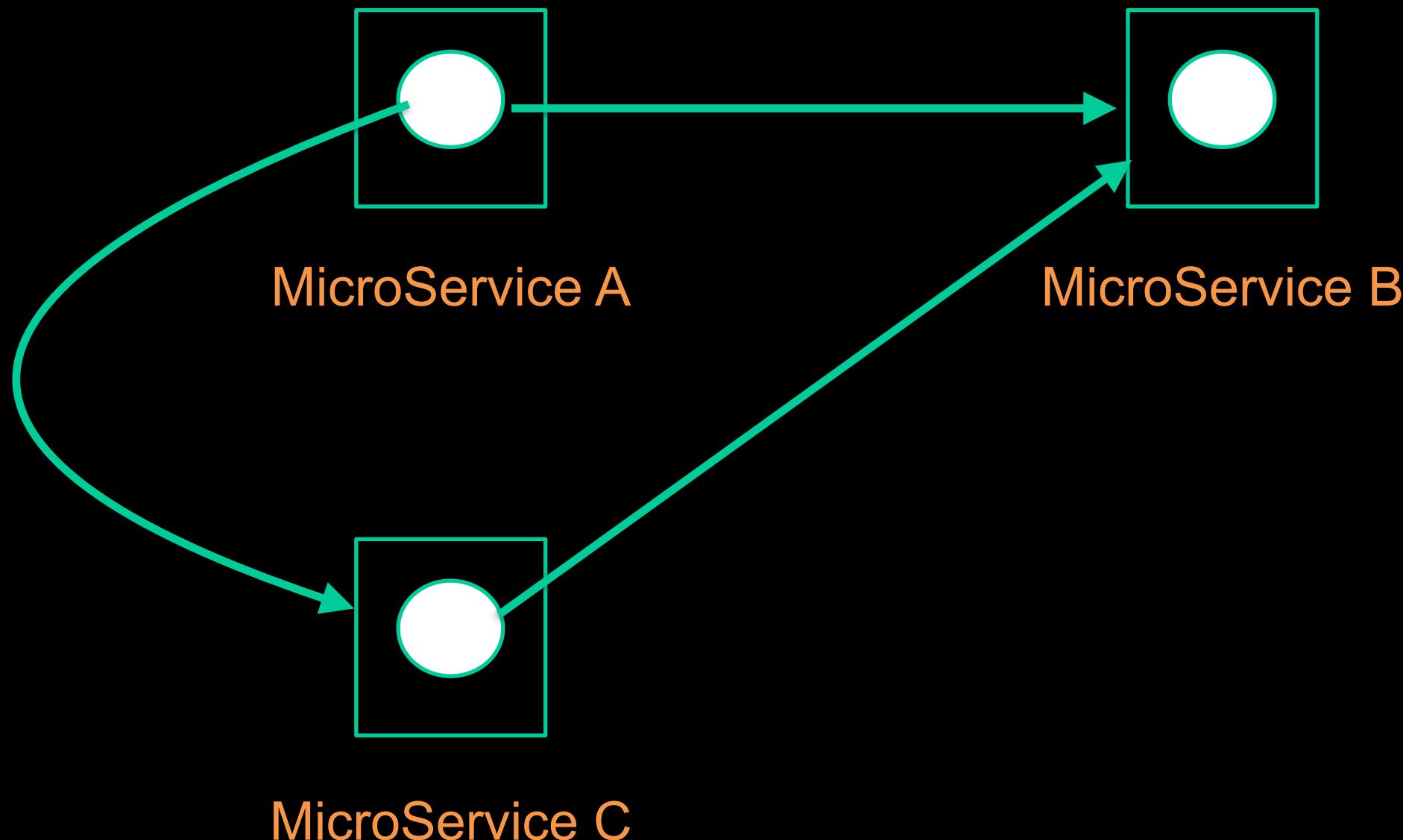


Monolith - Intra Process Communication
(internal obj messages)

L_S_E

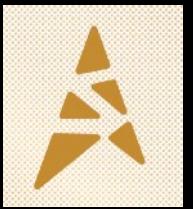


Patterns



Microservice - Inter Process Communication (IPC)

L
S
E



Mono / Micro Differences

- Mono less distributed (direct call)
- Micro more distributed (slower)
- Mono simple release cycle
- Micro complex release cycle
- Mono more resources per deployment
- Micro less resources per deployment



Patterns

Mono / Micro Services: Commonalities

- Service Registry, Locator (lookup)
- Circuit Breaker
- Saga Pattern
- Strangler
- Adapter, Bridge, Facade, Proxy and API Composite Pattern

L
S
E



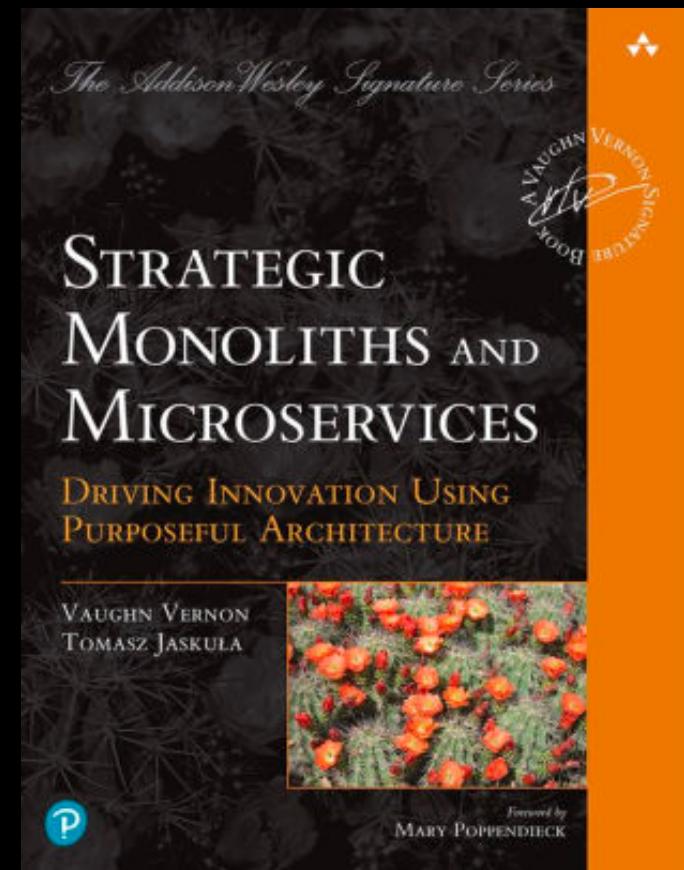
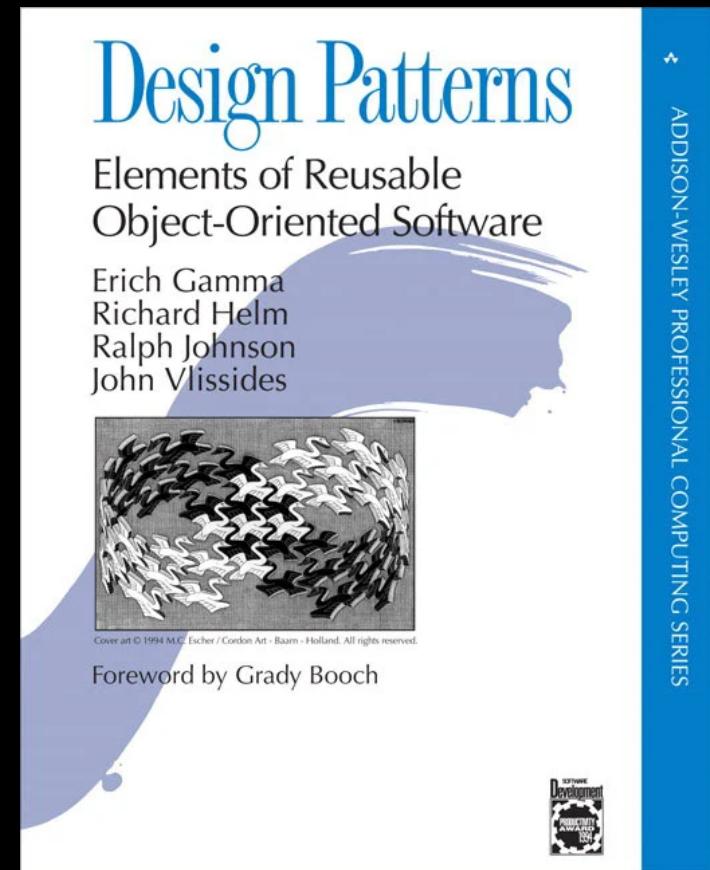
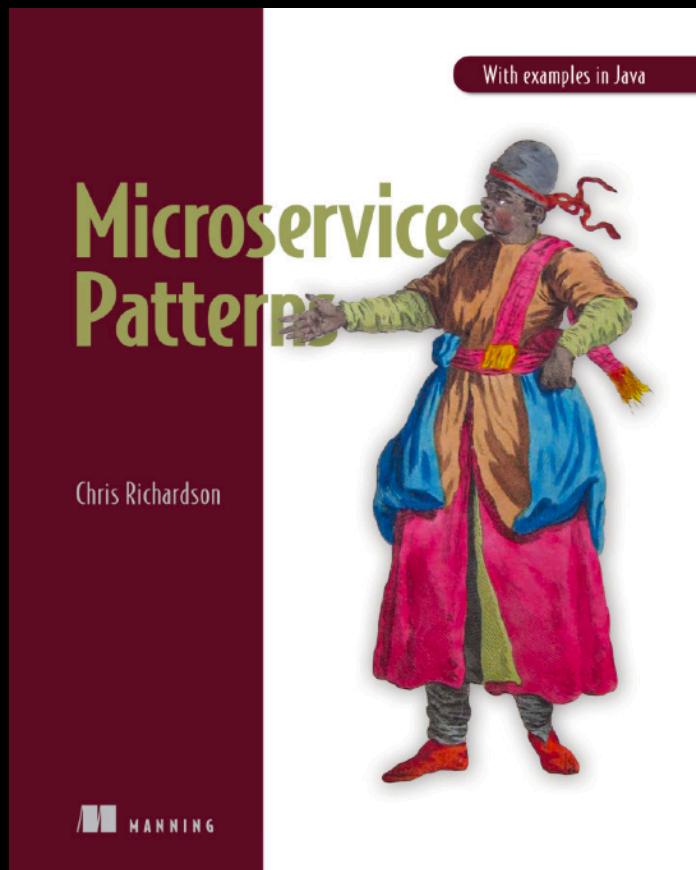
Patterns

Mono / Micro Services: Commonalities

- Controller
- Service
- Model Objects
- Adapter
- Producers / Consumers
- Repository / DAO
- Domain Drive Design (CQRS)

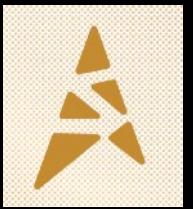


Patterns and Strategies?

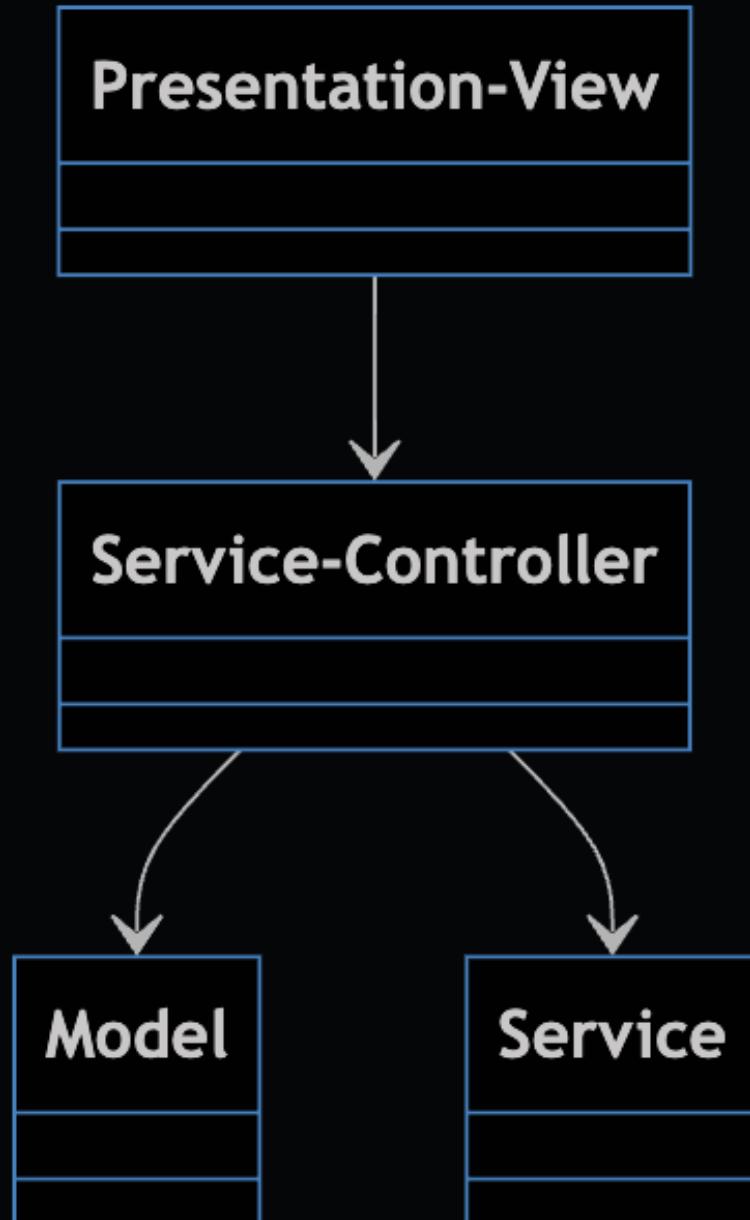


- <https://refactoring.guru>
- <https://www.geeksforgeeks.org/microservices>
- <https://www.enterpriseintegrationpatterns.com>
- <https://dzone.com/refcardz/design-patterns>
- https://en.wikipedia.org/wiki/Software_design_pattern
- <https://www.digitalocean.com/community/tutorials/gangs-of-four-gof-design-patterns>

L
S
E

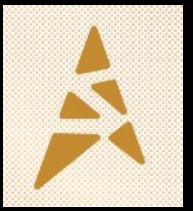


Patterns

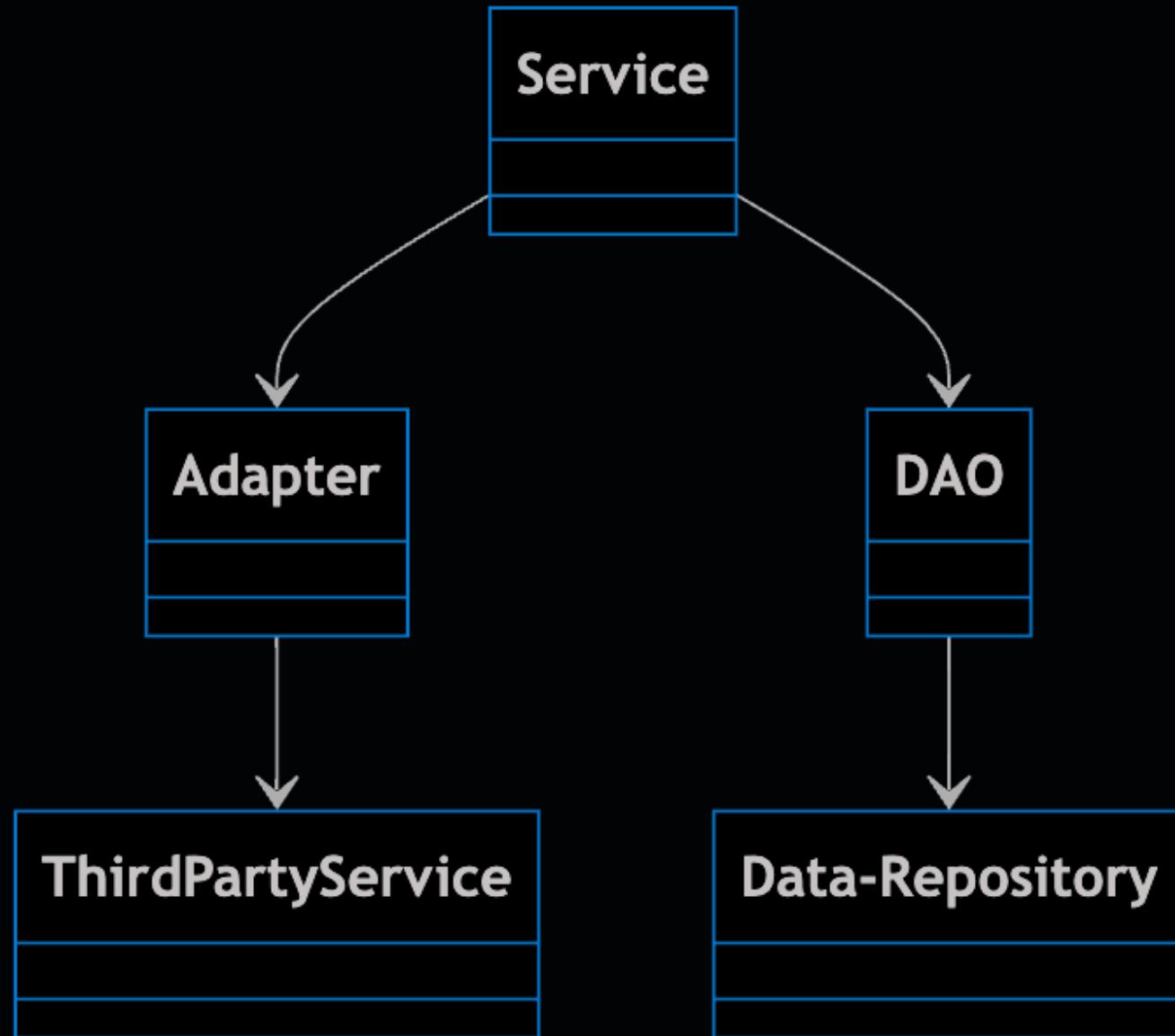


Controller-Service-Model

L
S
E

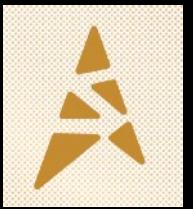


Patterns

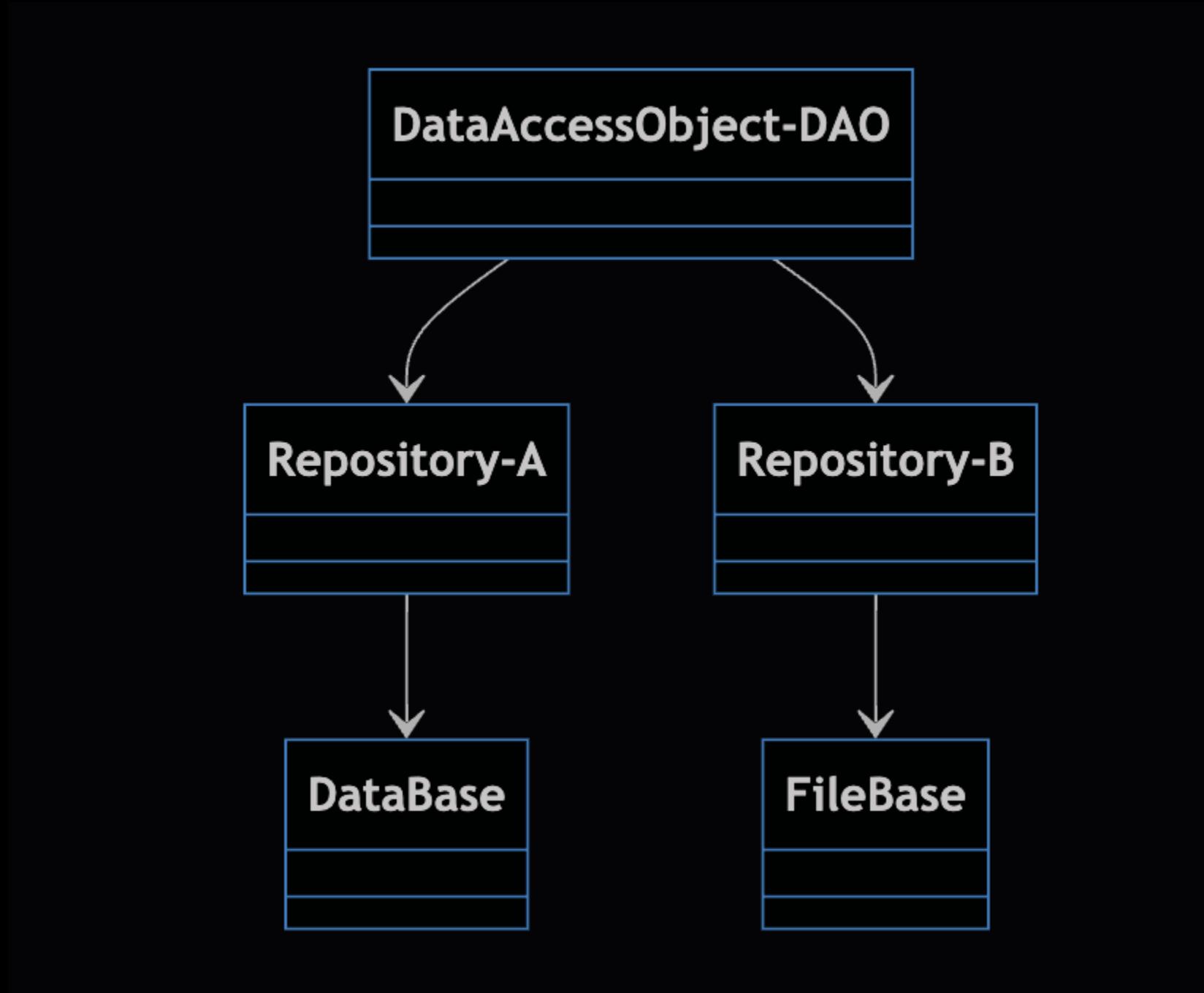


Service-Adapter-DAO

L
S
E

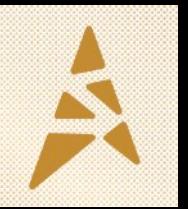


Patterns



Data Access

L
S
E



Pattern Summary

- decouple complexity, dependency, and layer against change
- help structure the implementation which supports better testing strategies
- Refactor to use patterns regardless of service type
- Implementing patterns will help with
 - function under test
 - class under test
 - isolating distributed interfaces



Introduction to Testing

Why do we test ?

L
S
E



Introduction to Testing



<https://english.stackexchange.com/questions/488178/what-does-it-mean-writing-a-minivan>

Be careful about incentives ...

L
S
E



Introduction to Testing

Why do we test ?

- 1st impressions, catch before customer
- catastrophic failure to business or life or mission
- software meets requirements
(business, security, performance)
- fast feedback on correctness, sooner in process is cheaper
- release verifiable units of delivery
- continuous delivery - release reliable, repeatable software



Introduction to Testing

Notable Testing Quotes

"If debugging is the process of removing software bugs,
then programming must be the process of putting them in."

— Edsger Dijkstra

"If you don't have time for testing,
you don't have time to be confident in your product."

— Lisa Crispin



Introduction to Testing

Notable Testing Quotes

"The bitterness of poor quality remains long after the sweetness of meeting the schedule has been forgotten."

— Karl Wiegers

"deliver better software faster to production"

— Dave Farley

L
S
E



Introduction to Testing

Notable Testing Quotes

“In a dark place we find ourselves,
and a little more knowledge lights our way”

— Yoda

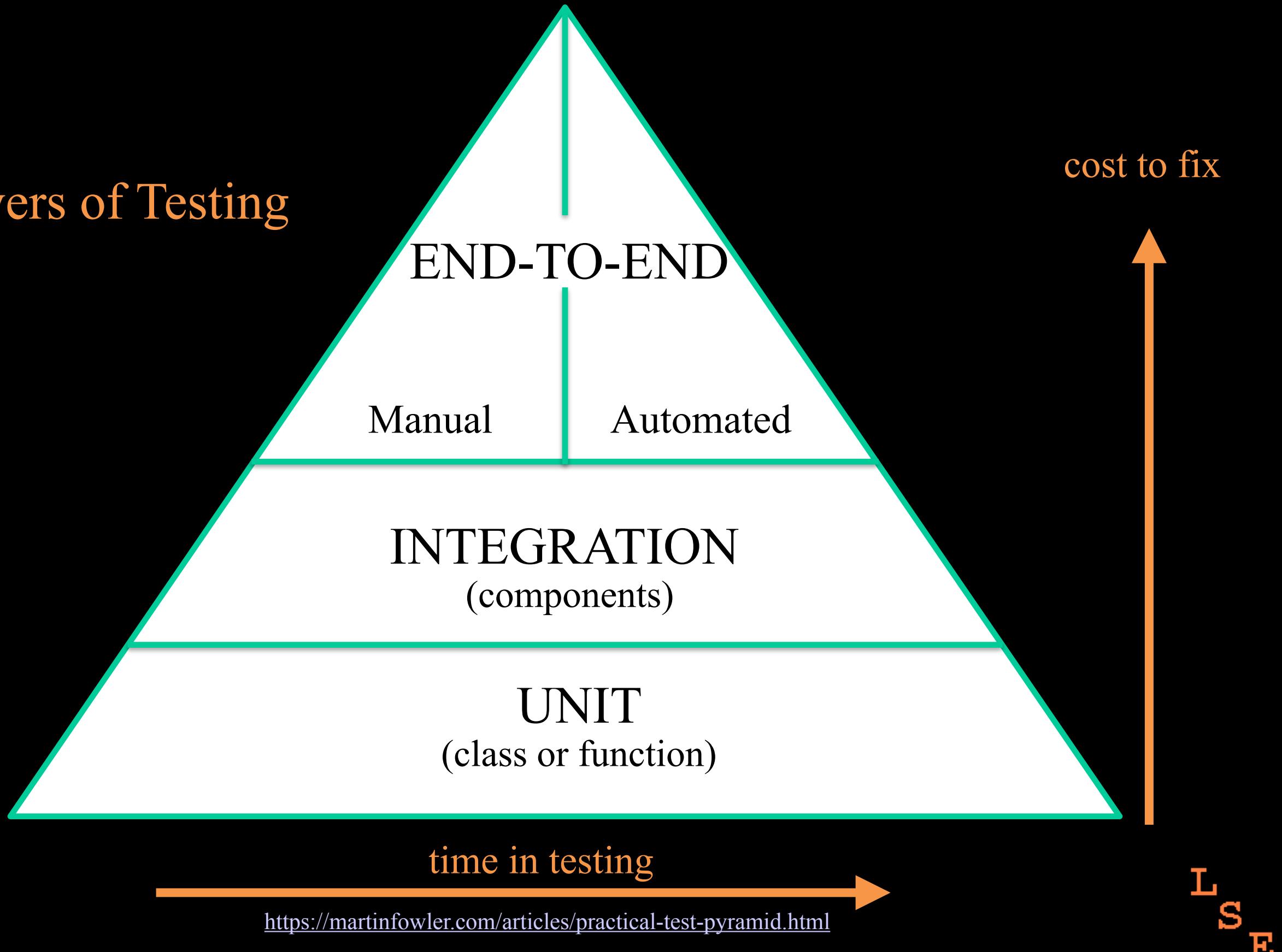
(Star Wars e3 - Revenge of the Sith, [original script](#))

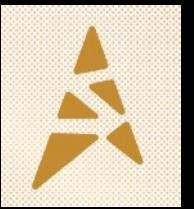
Testing sheds light on the unknown



Introduction to Testing

Layers of Testing





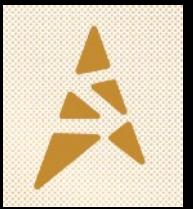
Introduction to Testing

Layers of Testing

- Units are cheaper, less resources to run
- Integration tests should be around component interfaces and boundaries
- let unit tests drive the creation of interfaces and abstractions
- simplest solutions first
- tests and automation requires maintenance, it is a cost, verify quality using code coverage and mutation testing

<https://martinfowler.com/articles/practical-test-pyramid.html>

L
S
E



Testing Terms

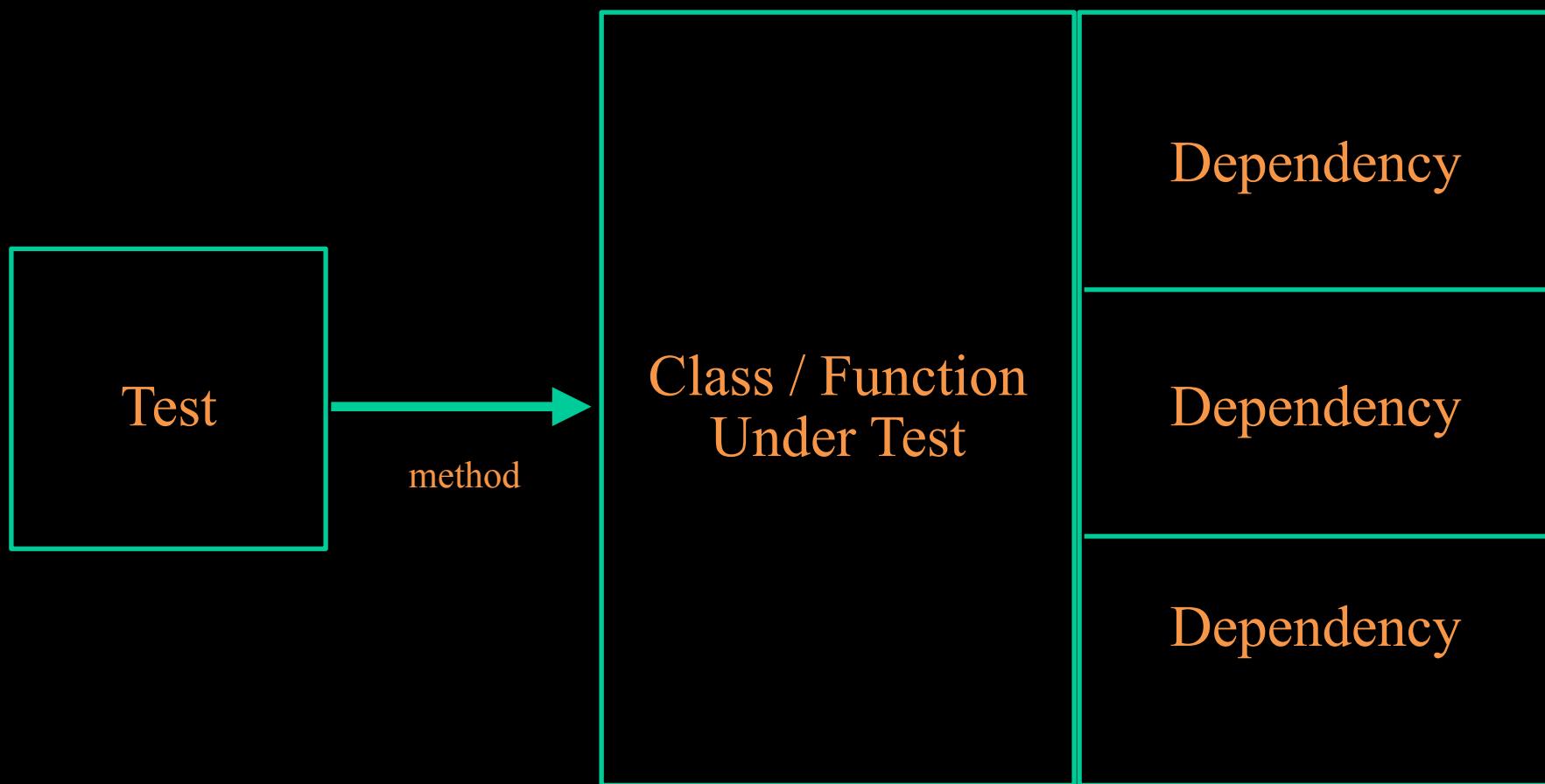
Test Doubles

(Martin Fowler: <https://martinfowler.com/articles/mocksArentStubs.html>)

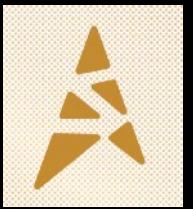
- Dummy objects: passed around but never called
- Fake objects: working non-prod implementations for testing
- Stubs: canned answers
- Spies: stubs that record information passed down to real implementation
- Mocks: objects pre-programmed with responses based on specification



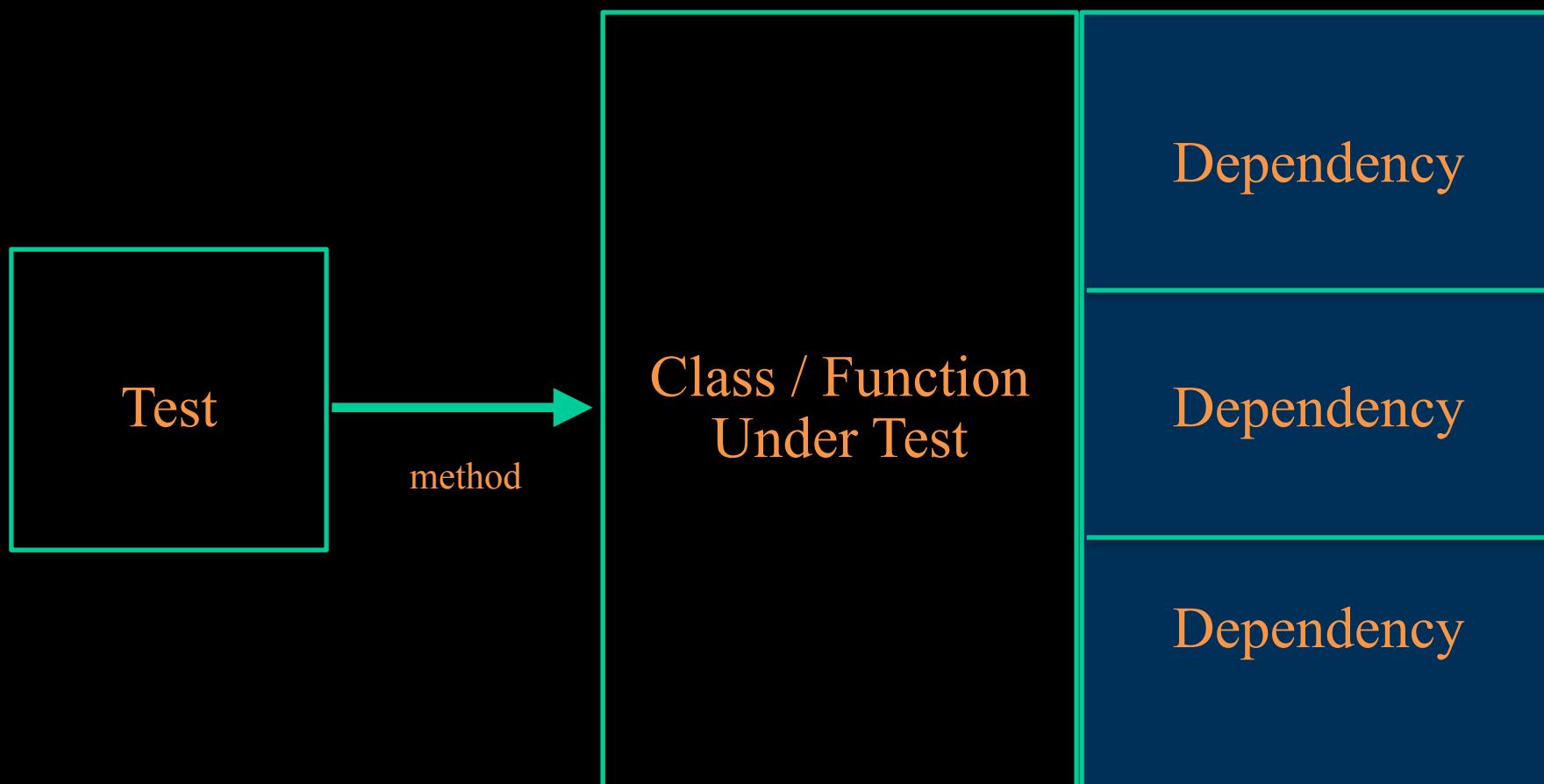
Unit Testing



L
S
E

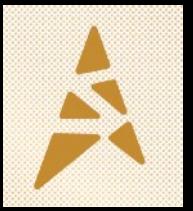


Unit Testing



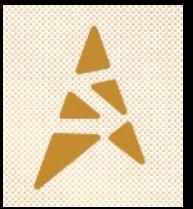
Mock The Dependencies

L
S
E



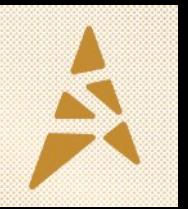
Unit Testing

- Function Under Test (FUT)
 - immutable, same input, same output
 - include edge cases, min, max, errors
 - avoid shared information, context per call
 - single responsibility principle

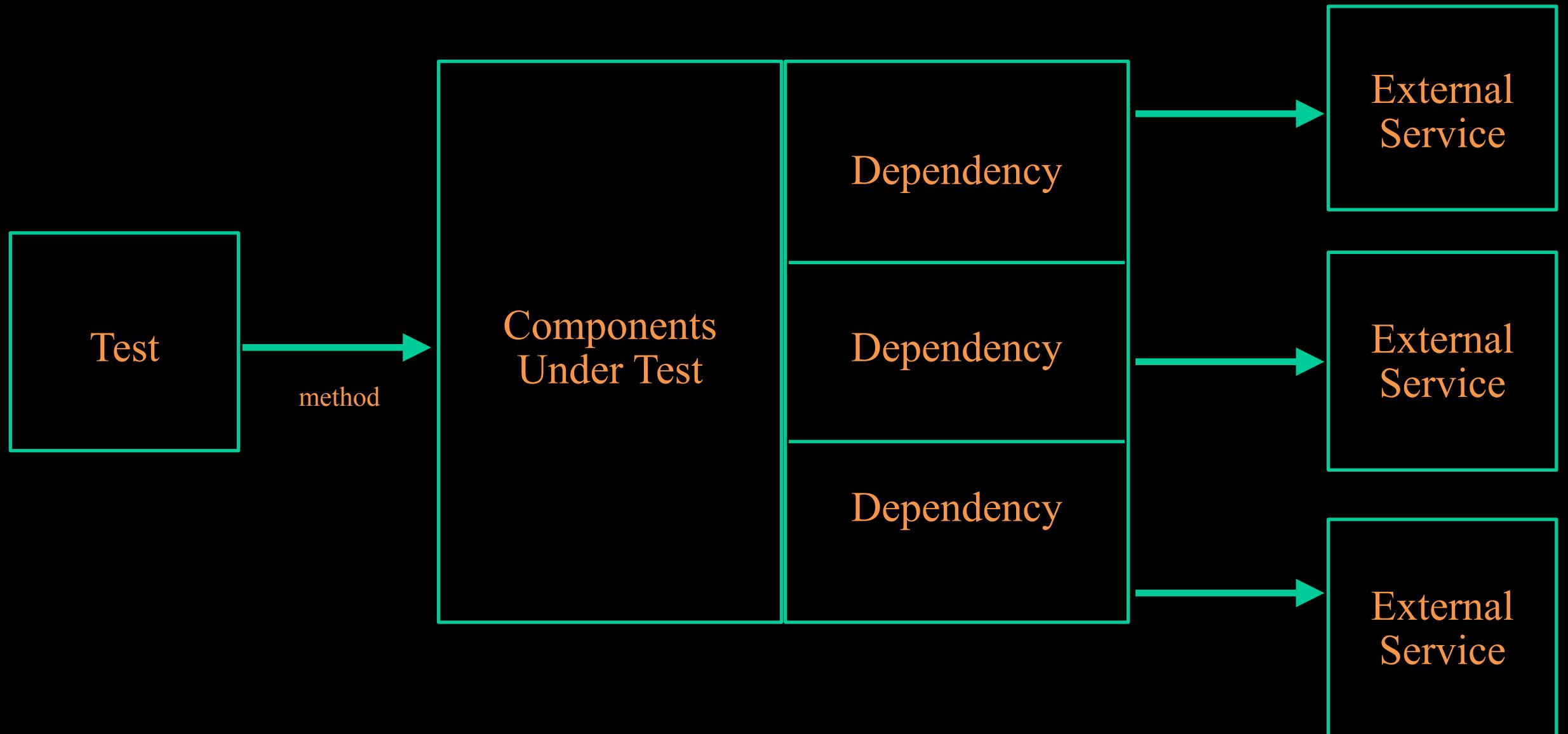


Unit Testing

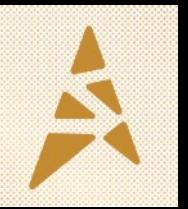
- Class Under Test (cUT)
 - test each method (class should minimize public methods, test public first)
 - class should use dependency injection
 - minimize and mock dependencies
 - include edge cases, min, max, errors
 - single responsibility principle



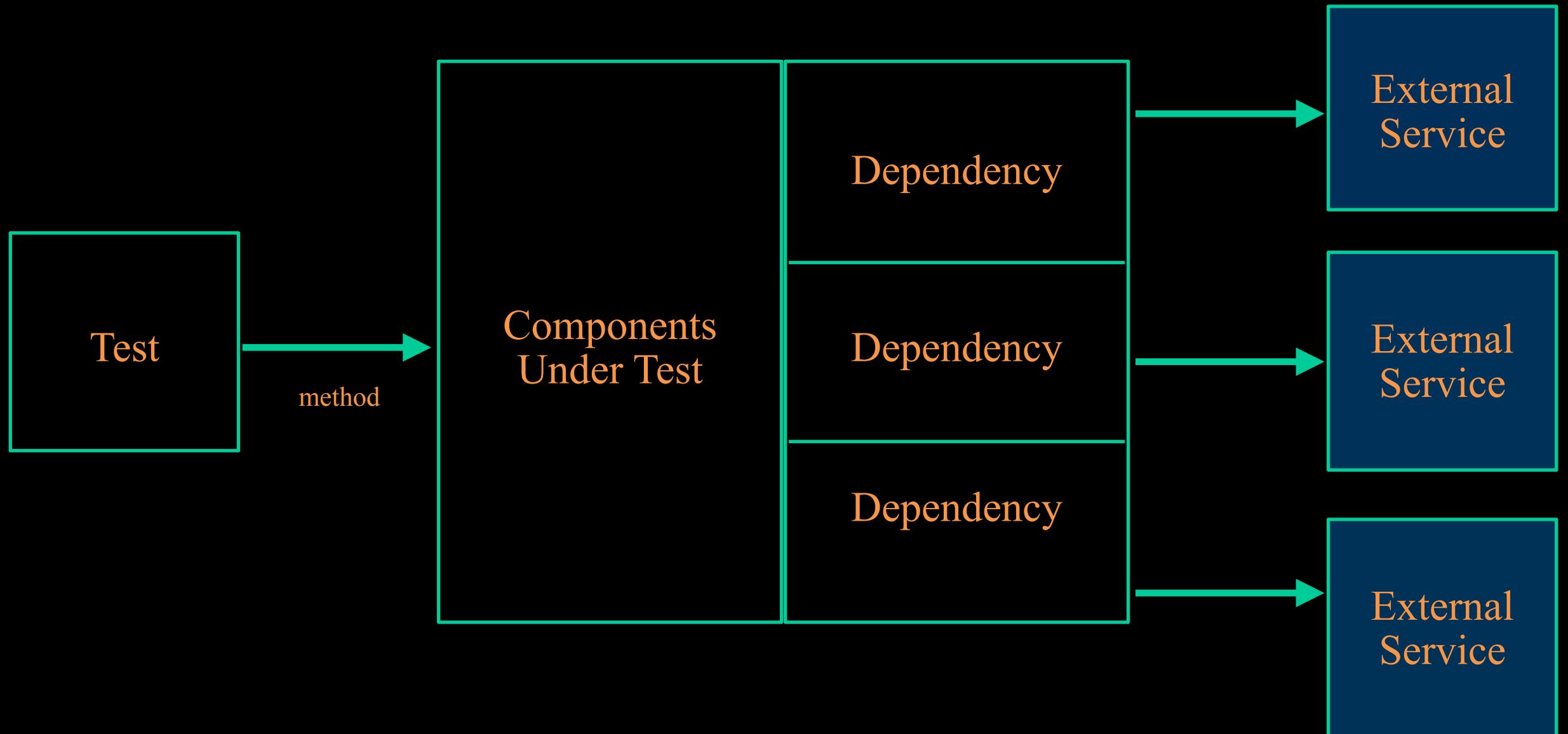
Integration Testing



L_{S_E}



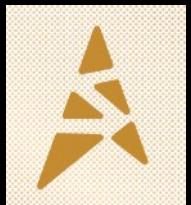
Integration Testing



Mock or Simulate or Use live System

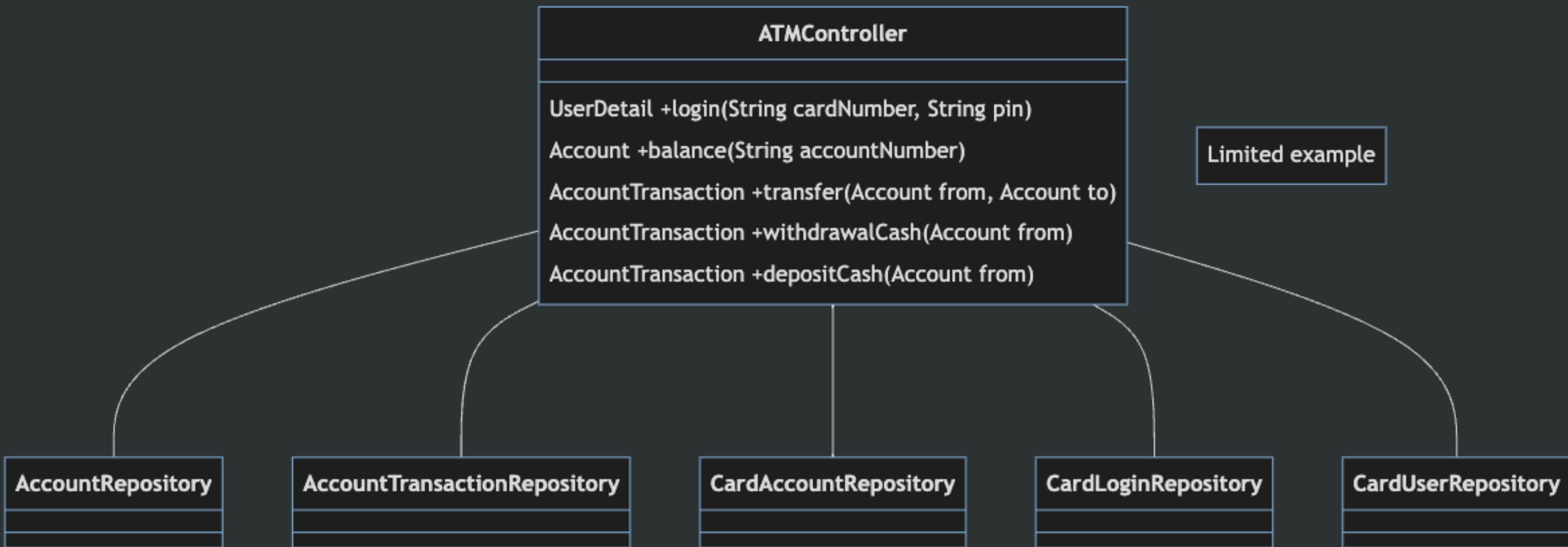
verify components and dependencies
execute as expected

L_SE

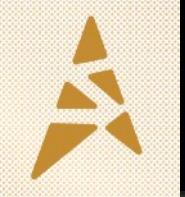


Demo Summary

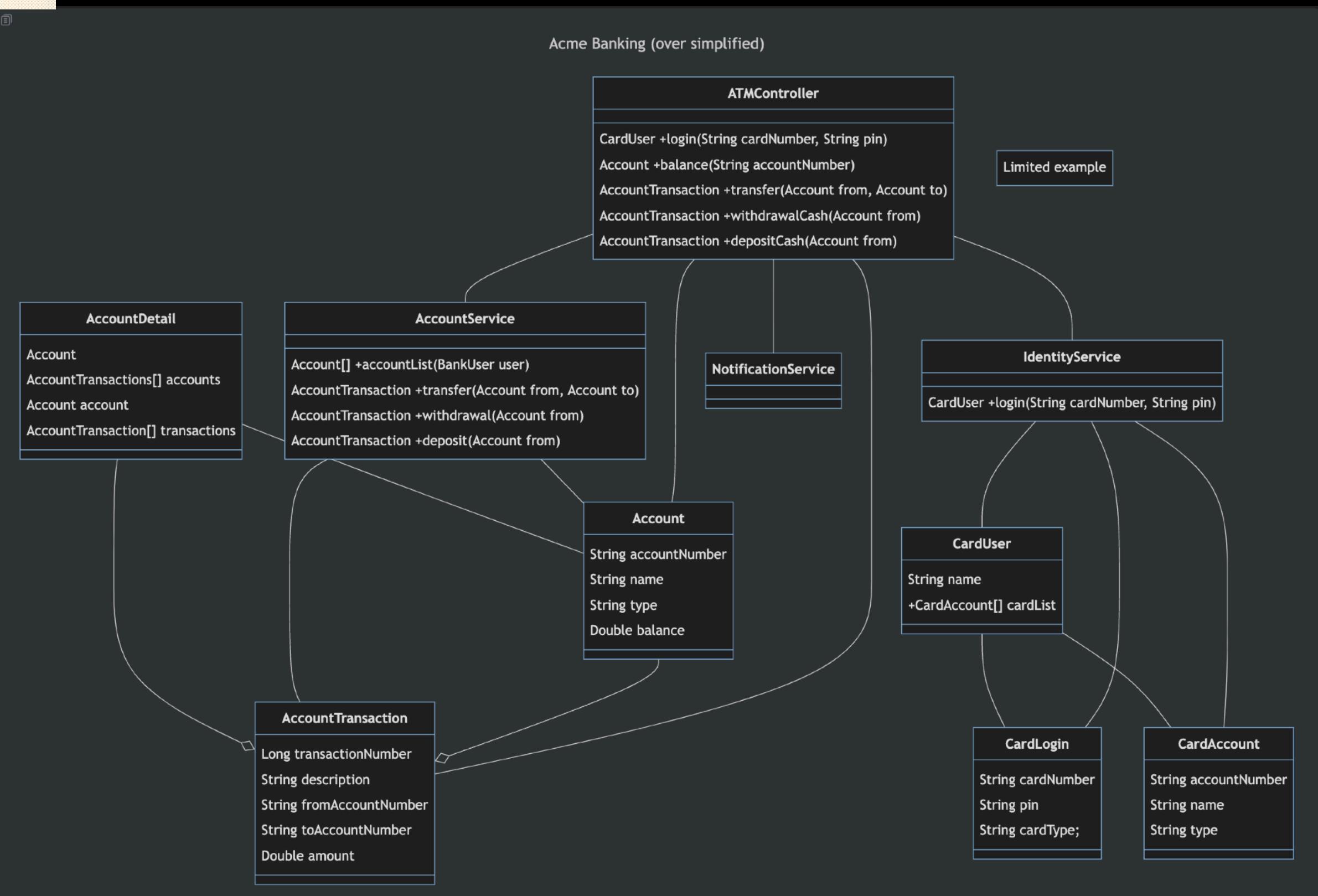
Acme Banking (over simplified)



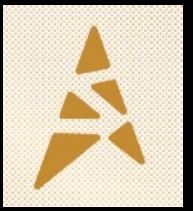
L
S
E



DEMO Refactored



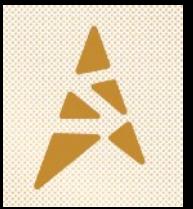
L
S
E



Examples

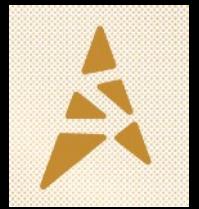
DEMO

L_{S_E}



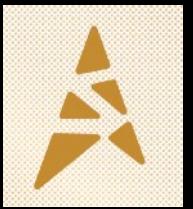
Unit Testing Tips

- Use Test Factories, Parameterized and Streams to go through different test inputs with expected outputs
- Use JSON serialization to test getter / setter properly working
- Test Enums for size and conversions (upper and lower)



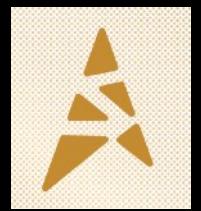
Unit Testing Tips

- Use Fake Data Generators
(avoid real production data, protect privacy)
- Avoid mocking static functions
- Make sure tests are thread safe
- Make sure frameworks are thread safe
- Avoid using `SpringBootTest`
(only integration)



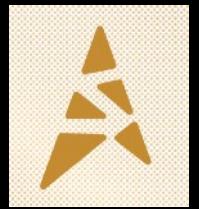
Unit Testing Tips

- Setup Mocks in `@BeforeEach` step
- Clear / Reset mocks in `@AfterEach` step
- Make Tests Order Independent
- Use Date Strategies for predictable results
- Use `ignoringFields` to avoid data you cannot control during test (eg sequence number, transaction number)
- Use JUnitPerf or Java MicrobenchHmarking (JMH) for isolated performance testing



Integration Testing Tips

- Components Under Test (IT)
 - verify group of components work together as expected
 - verify interfaces and protocols (client communications)
 - SQL verification (CRUD)
 - Message serialization (in to and out of broker)
 - Spying third party libraries for inputs and outputs
 - Proxy (mock / record-playback) third party APIs
 - Simulated third party services (S3, SFTP, SMTP, etc)
 - Contract Testing (verify interface has not changed)



End-to-End Testing Tips

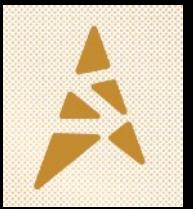
- Verification of front to back (UI or API)
- QA Manual UAT
- Selenium for HTTP UI testing
- Insomnia / Postman for API testing
- Security (Burp: <https://portswigger.net/burp/communitydownload>)



Other Topics

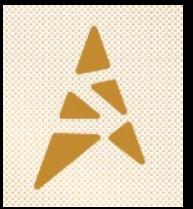
- Performance
- Load
- Chaos
- CI/CD

L
S
E



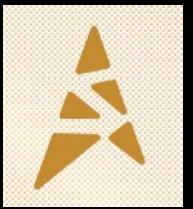
Other Topics

- Performance (speed of min and max users)
 - Scaling clients for concurrency just as important as services
 - Verify no rate limiting by IP address, use agents on different IPs
 - VPNs can slow down simulated users
 - Run with one node, then scale up and compare
 - Database and Message connection pools can impact I/O performance



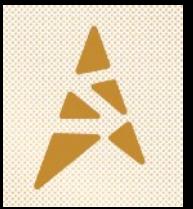
Other Topics

- Load (success of execution beyond normal)
 - Sustained Over Time (80% users)
 - Stress (breaking the system beyond 100% max users)
 - Identify bottlenecks, tune and repeat
 - Consider scaling horizontally over vertically
 - Customize load balancing based on metrics
(<https://learnk8s.io/autoscaling-apps-kubernetes>)



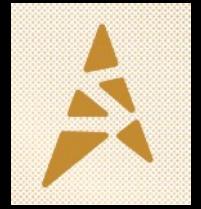
Other Topics

- Chaos
 - Netflix started with Chaos Monkey
(<https://github.com/Netflix/chaosmonkey>)
 - Random death of services
 - Goal is to maintain successful transactions
 - Identify failures and tune
 - Consider Circuit-Breakers
(retry and default answers on failure)

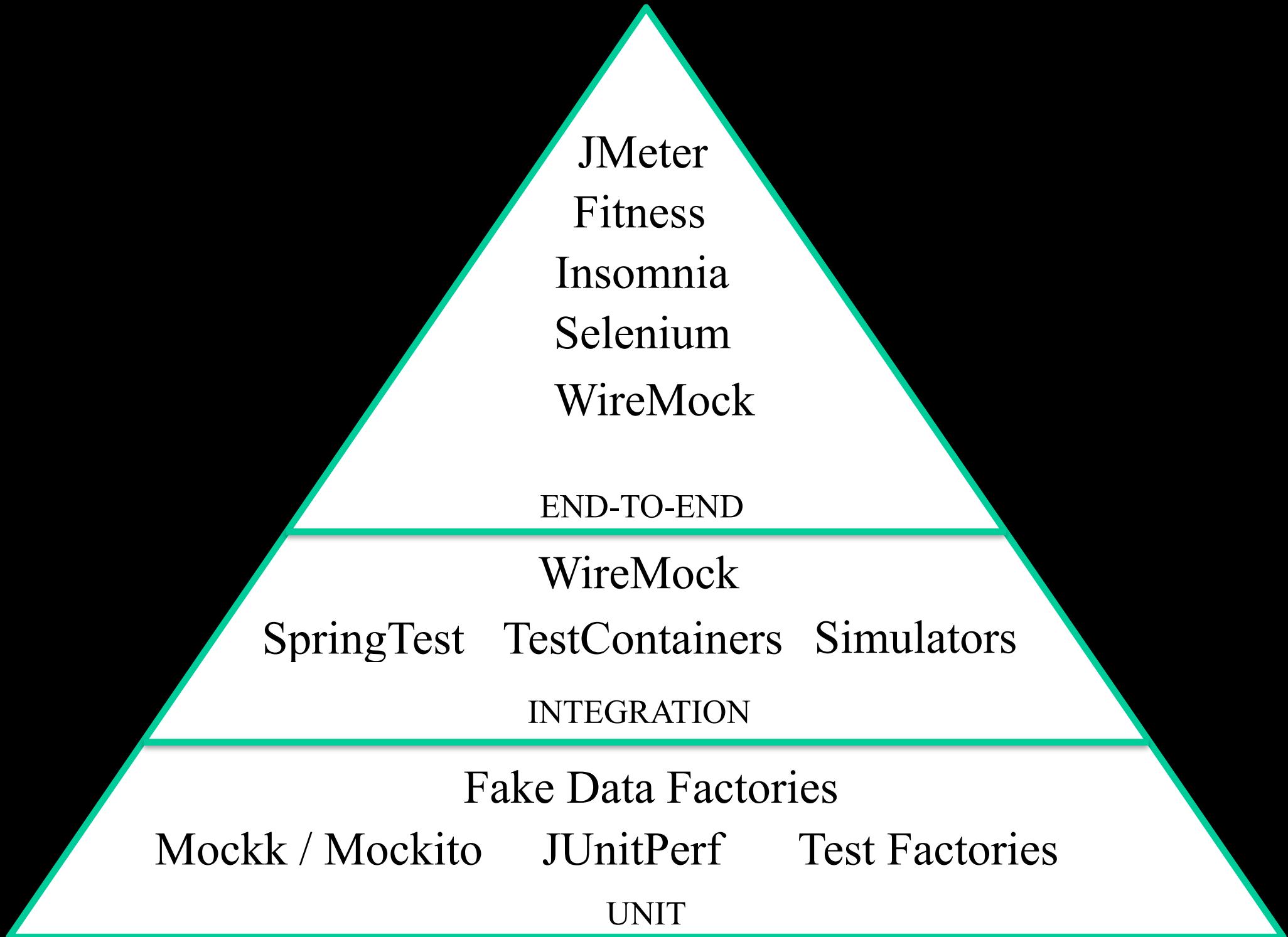


Other Topics

- CI/CD
 - MR/PR should at a minimum lint, compile, and unit test
 - Integration Test before branch/tag promotions
 - End-to-End Test after Integrations and before promotions
 - Performance / Load against deployment environment (like prod)



Tools -n- Layers



L
S
E

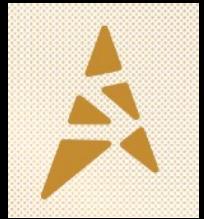


Resources

- <https://microservices.io>
- <https://refactoring.guru>
- <https://martinfowler.com/articles/practical-test-pyramid.html>
- <https://www.geeksforgeeks.org/microservices>
- <https://www.enterpriseintegrationpatterns.com>
- <https://dzone.com/refcardz/design-patterns>
- <https://martinfowler.com/articles/mocksArentStubs.html>



L
S
E



Questions ?

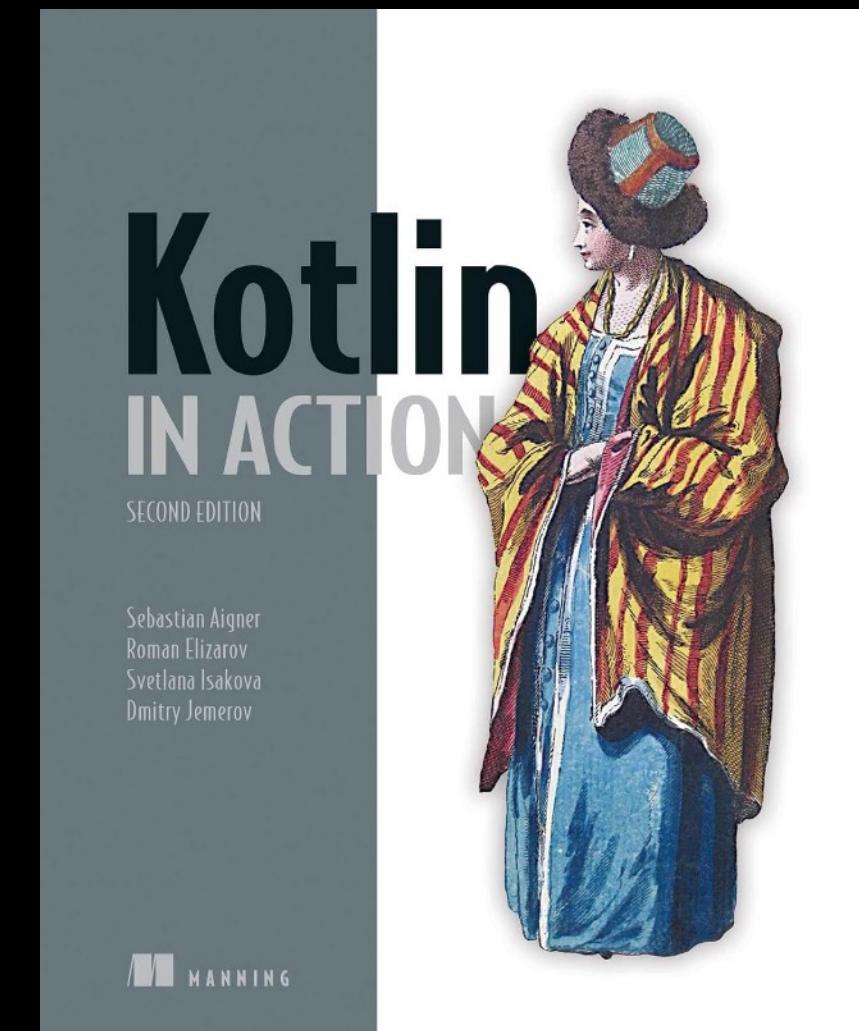
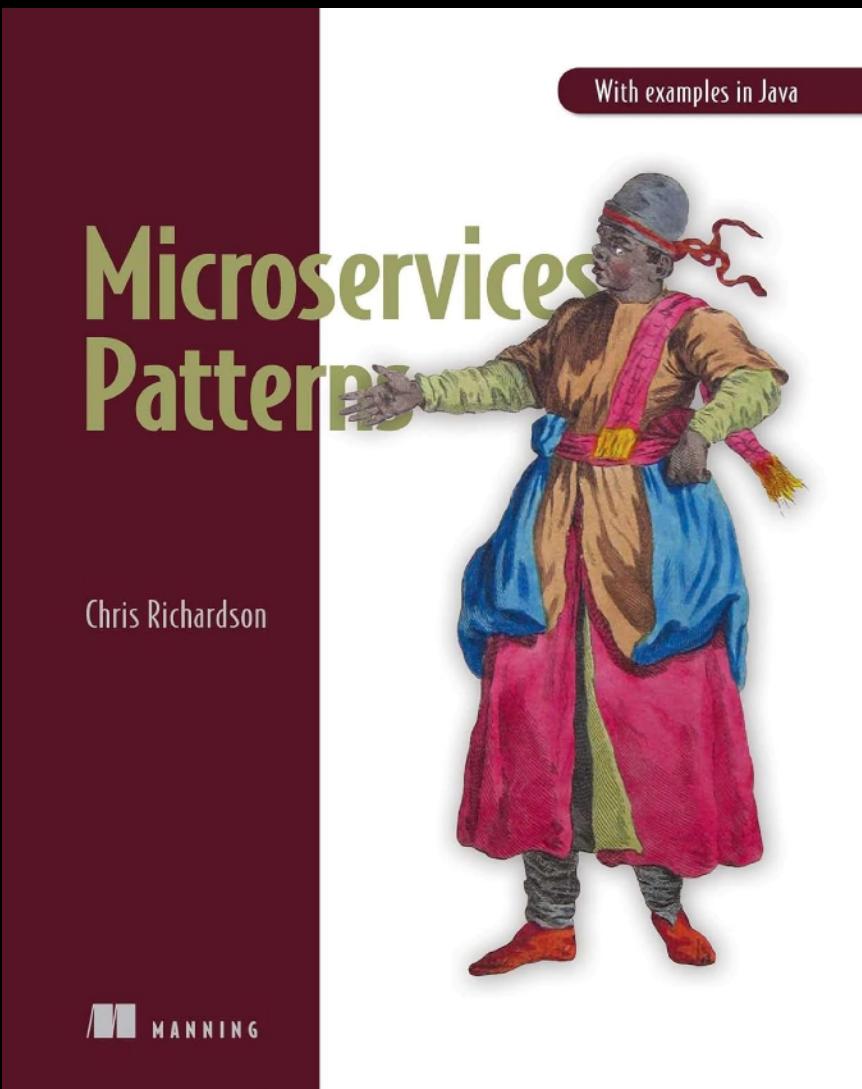
<https://github.com/lseinc/stirtrek25-backend-testing-strategies>



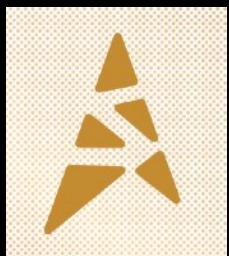
L
S
E



Gift Giveaway



L
S
E



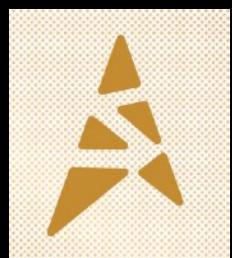
Thank You !

<https://github.com/lseinc/stirtrek25-backend-testing-strategies>



David Lucas
Lucas Software Engineering, Inc.
www.lse.com
dllucas@lse.com
[@DavidDLucas](https://twitter.com/DavidDLucas)

L_S_E



<https://github.com/lseinc/stirtrek25-backend-testing-strategies> L_S_E