Testing Strategies for Monoliths and Microservices

Presenter: David Lucas

Who am I ?

- Over 25 years in software industry
- Continuous Learner
- Java since 1998, Kotlin since 2017
- Senior Software Engineer developing Enterprise Kotlin Microservices

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

L S E

Goals

- Monoliths / Microservices (Patterns)
- Introduction to Testing
- Unit Testing
- Integration Testing
- End-to-End
- Advanced Topics

- parting gifts

Monoliths and Microservices

So how does all this relate to:

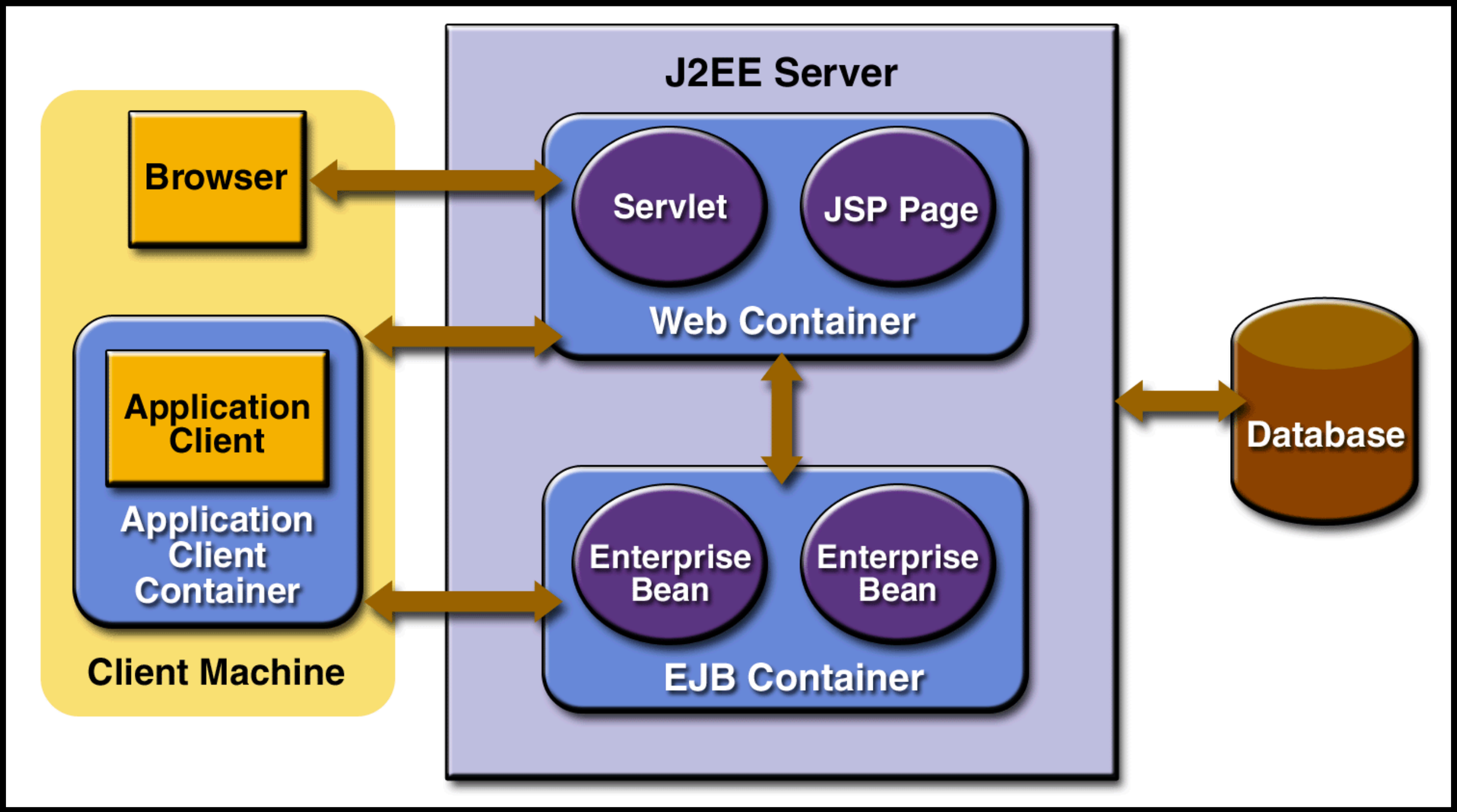
- Monoliths ?
- Microservices ?

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

Monoliths


J2EE Server
Browser
Servlet
JSP Page
Web Container
Application Client
Application Client Container
Database
Enterprise Bean
Enterprise Bean
EJB Container
Client Machine

Challenges with Monoliths

- One instance of processing power
- Centralized Processing Logic and Data
- Tight Coupling of Dependencies
- Early Application Servers
  weblogic / websphere / jboss / tomcat
- Modern JEE Services
  spring / ktor / quarkus / micronaut

What about Polyliths?

(poly=many lith=stone)

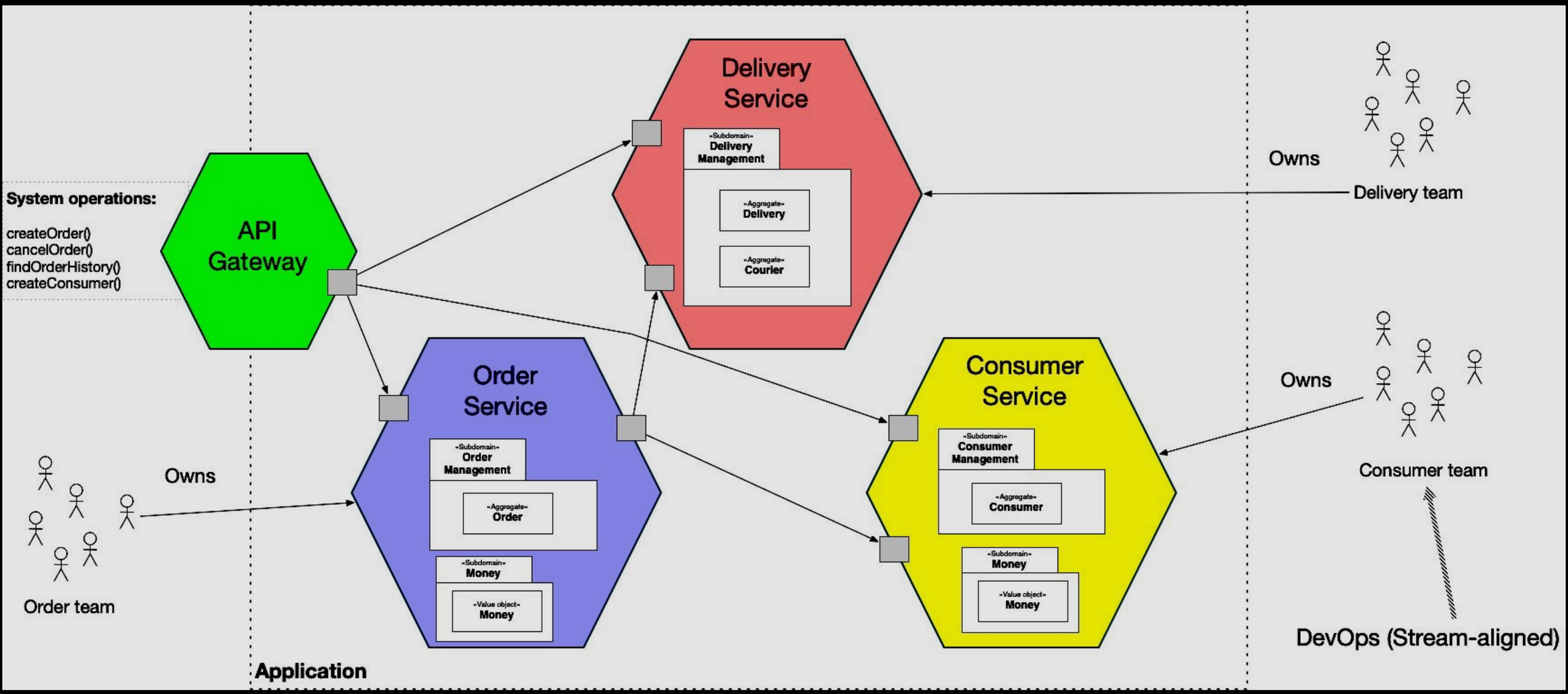
Stonehenge, prehistoric monument, Wiltshire, England, outer ring of stones, each around 13 feet high by 7 feet wide, weighing 25 tons

Microservices Components (poly-services)

- Decentralized Logic and Data
- Loosely Coupled
- Fewer support layers
- Each Instance can have specific resource (CPU / Memory / DataBase) scaling
- Easier Async communications
- Separate Release Life Cycle

Microservices

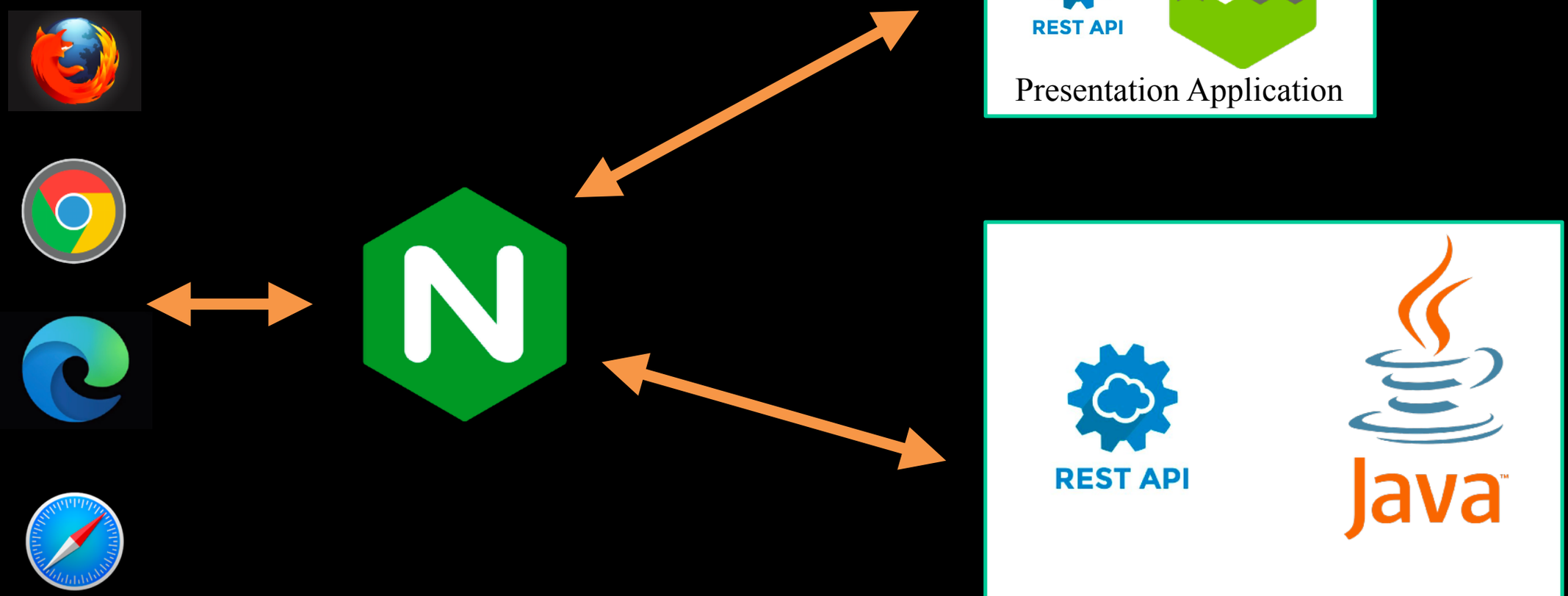
(distributed application components and teams)



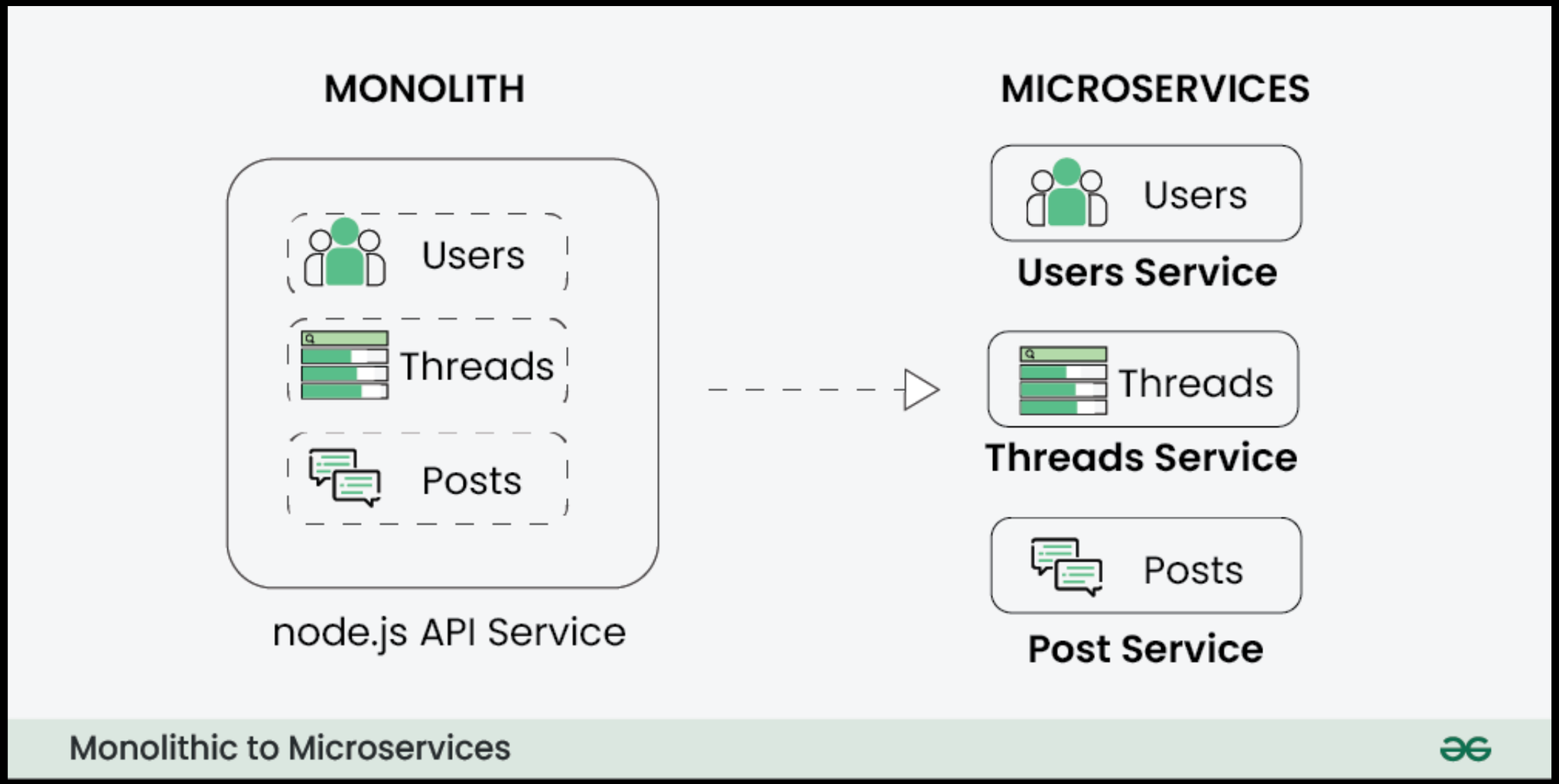
https://microservices.io/patterns/microservices.html

LSE

Monolith Flow

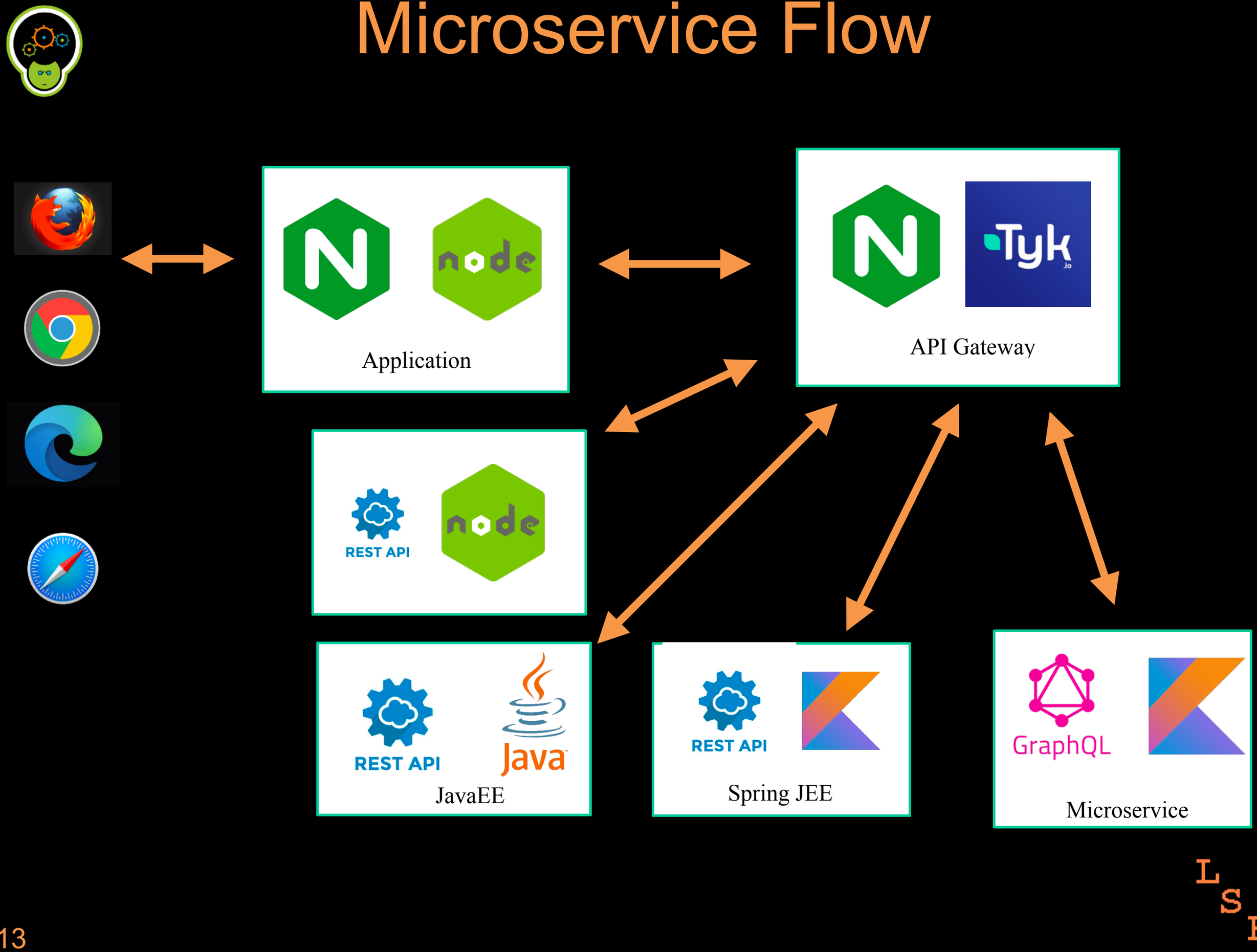


The Breakup

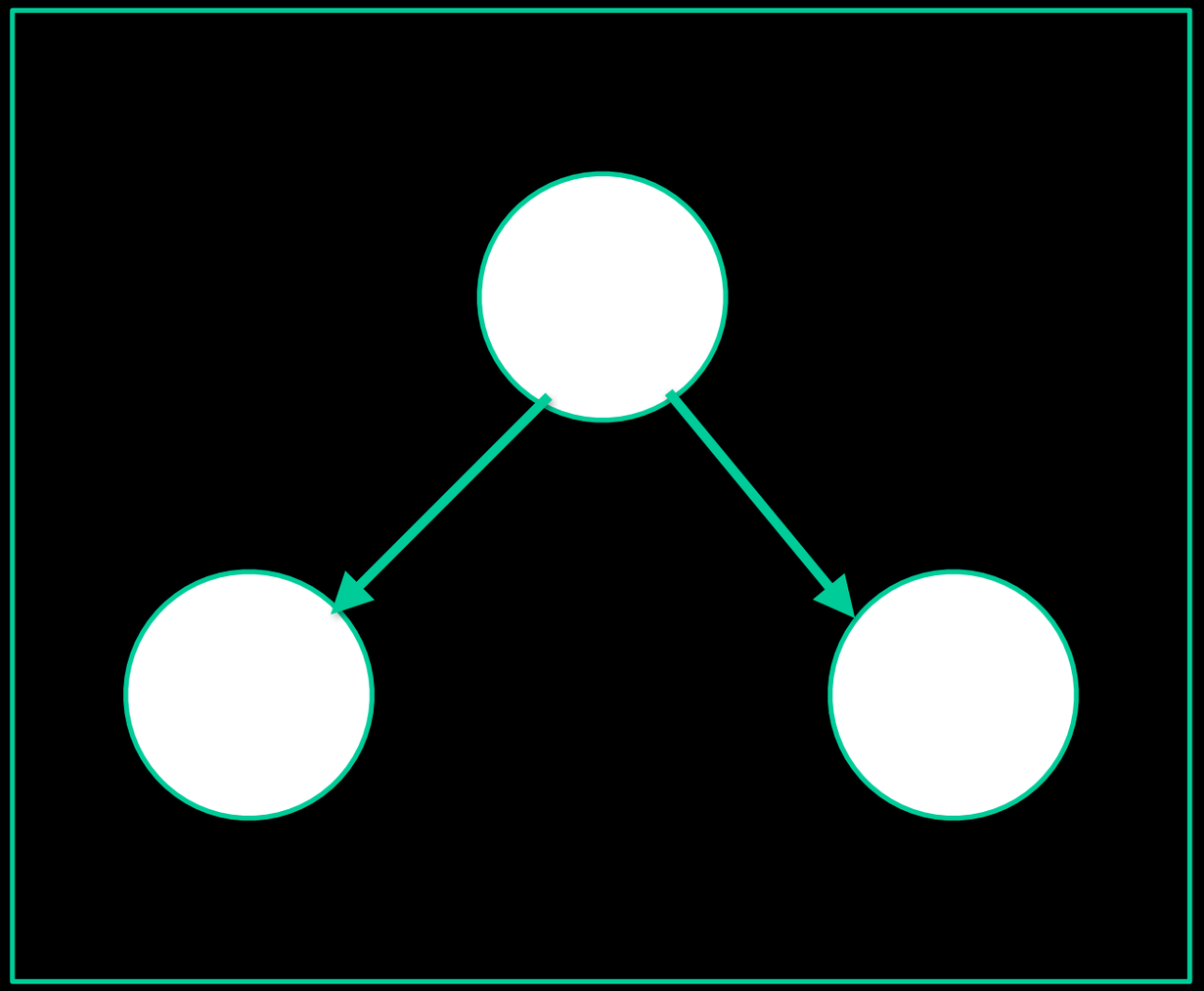


Monolith to Microservice
https://www.geeksforgeeks.org/microservices/

Microservice Flow

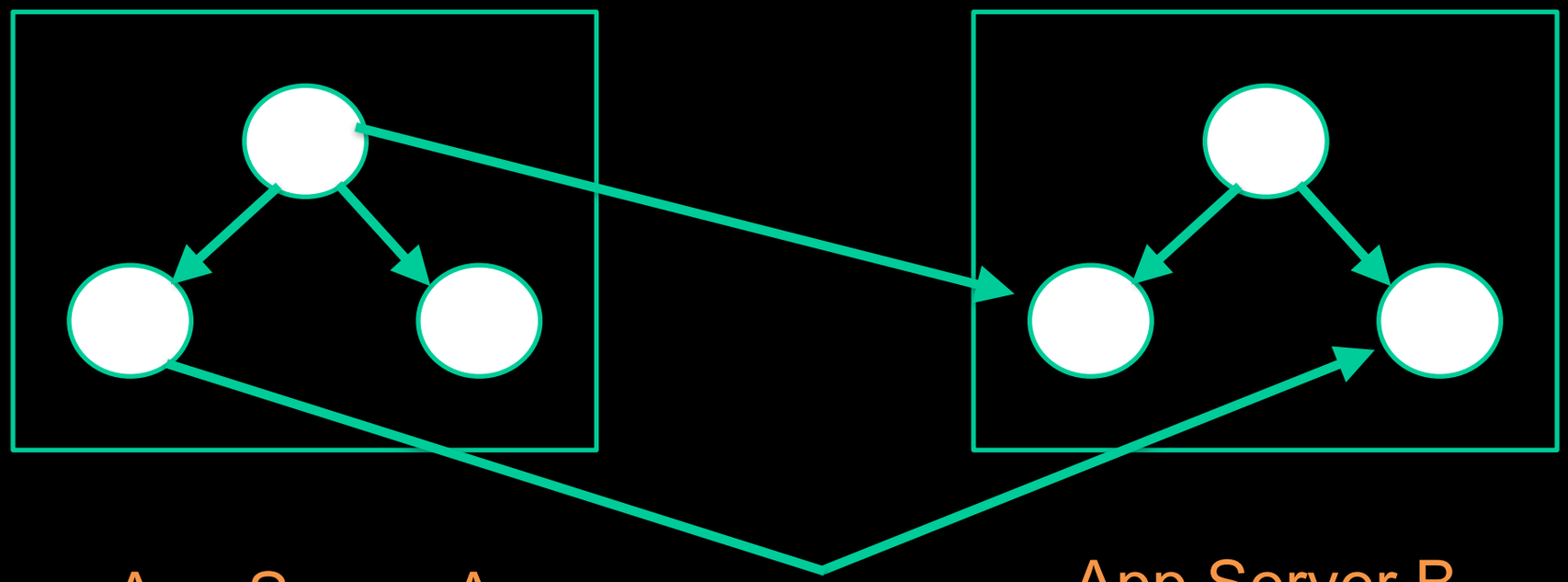


Patterns



Monolith - Intra Process Communication
(internal obj messages)

Patterns

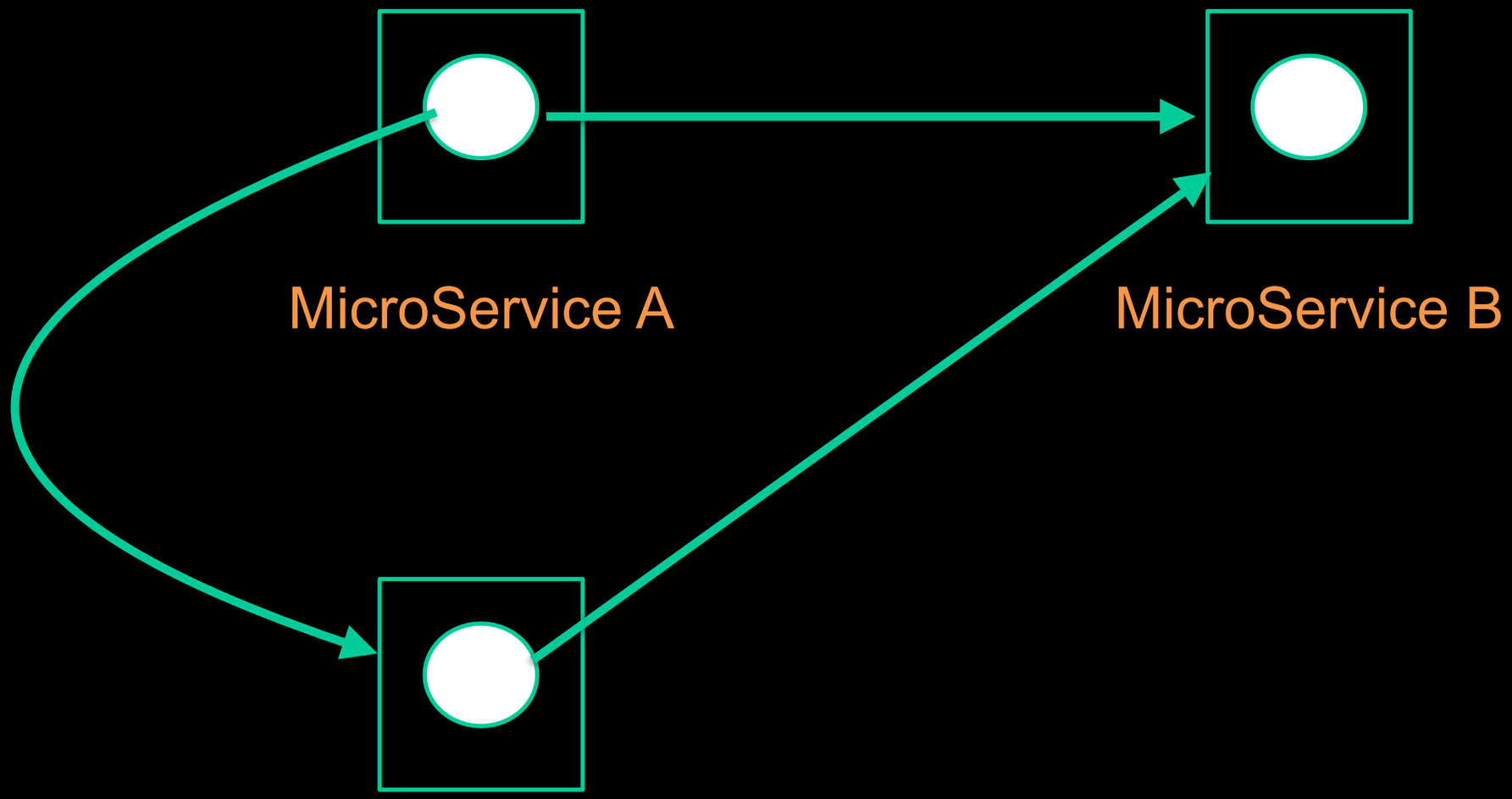


App Server A

App Server B

Monolith - Inter Process Communication (IPC)

Patterns



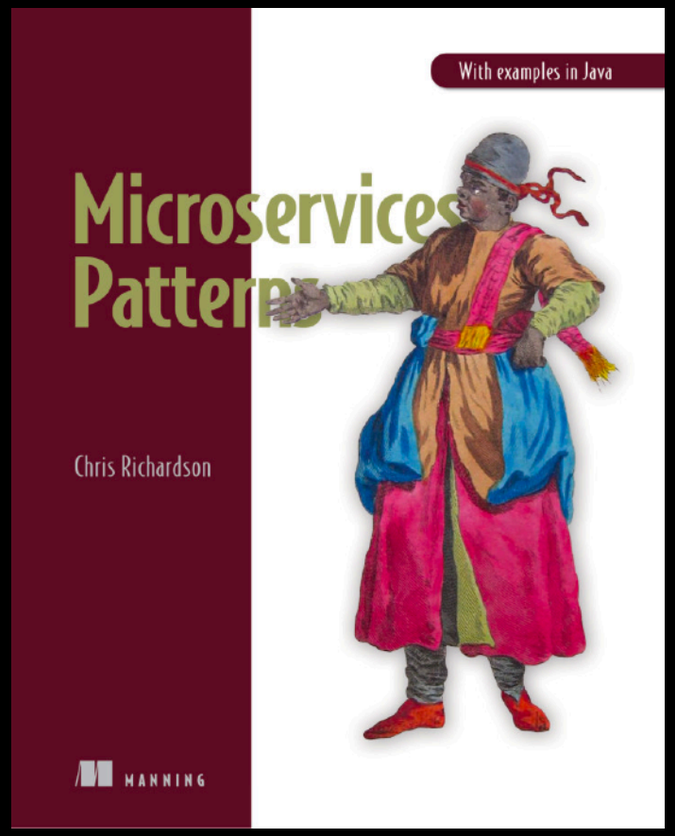
Microservice - Inter Process Communication (IPC)

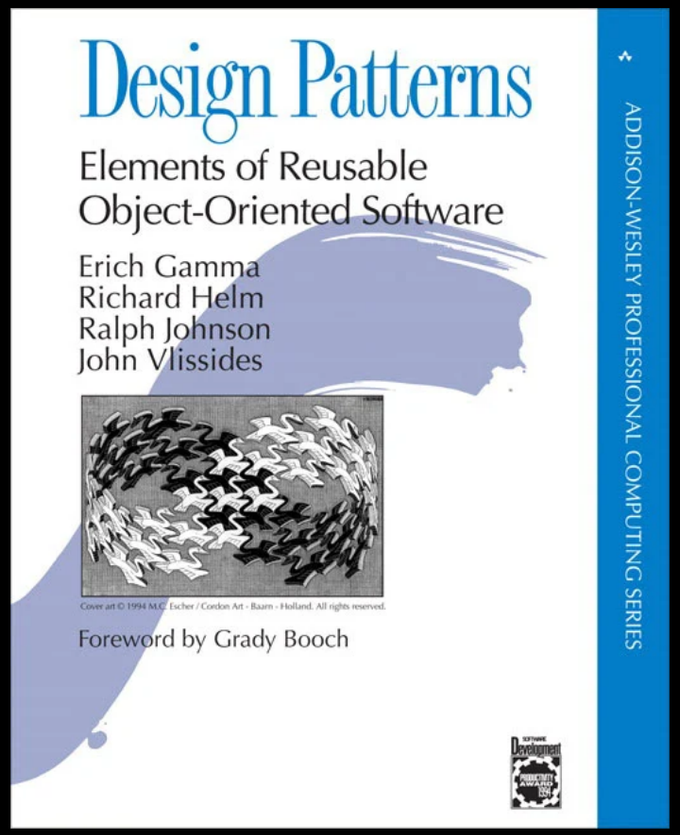
Patterns

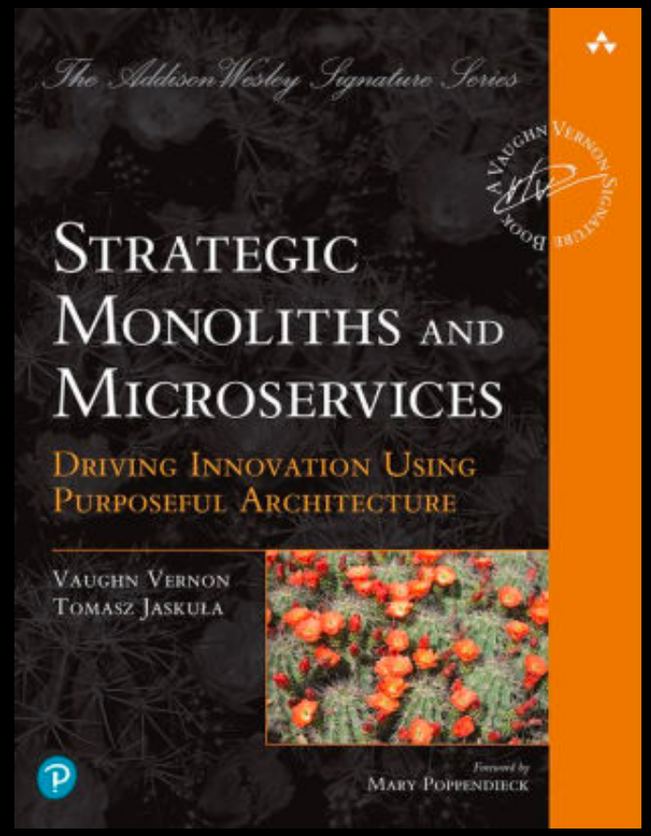
Mono / Micro Services: Differences

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

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

LSE

Patterns

Mono / Micro Services: Commonalities

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

Patterns

Mono / Micro Services: Commonalities

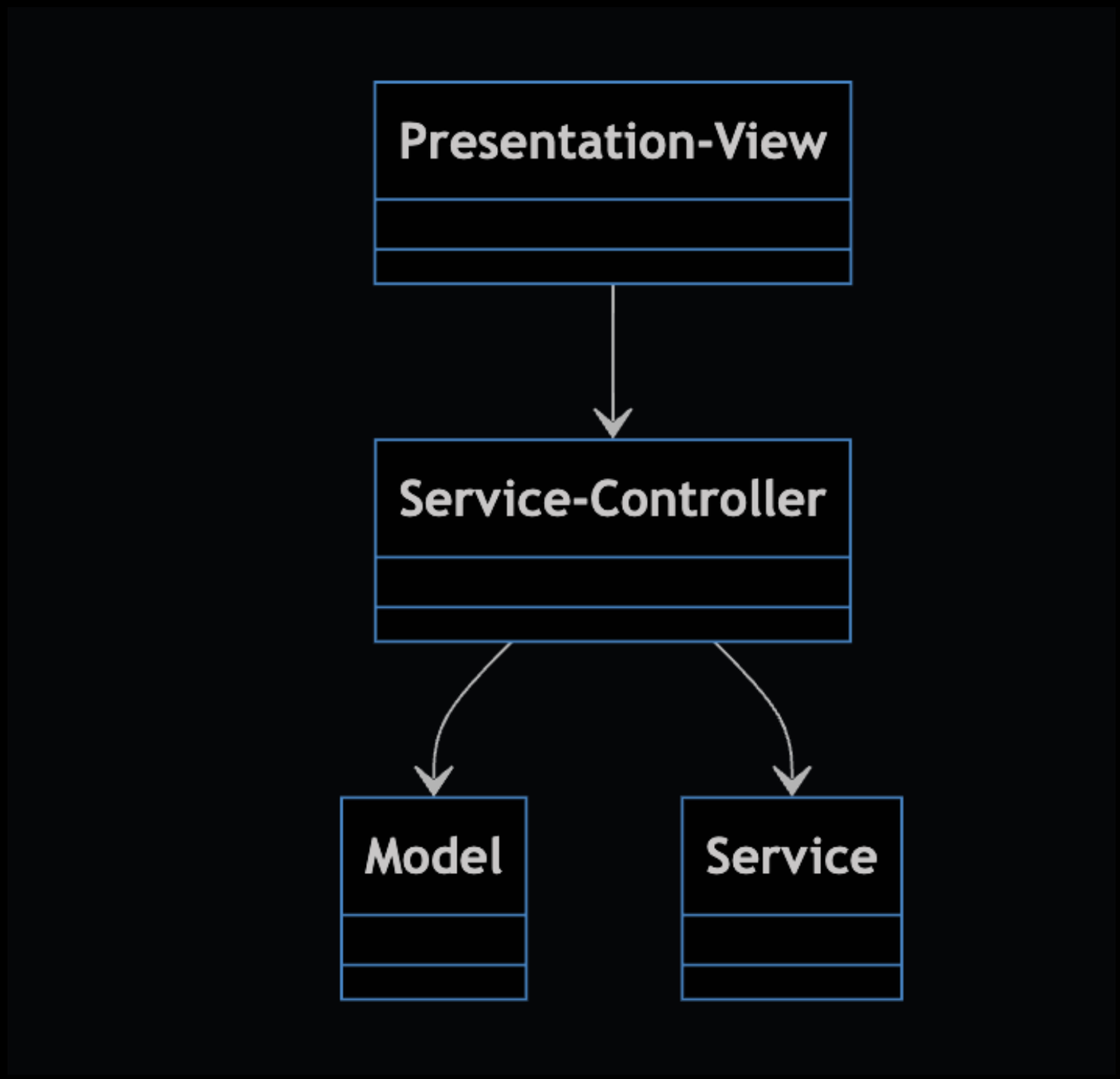
- Factory, Dependency Injection
- Singleton (single instance of interface) (does it work in threaded? distributed env?)
- Command / Strategy (authorization of data based on role)
- Decorator (extend without modifying)
- Observer Pattern (notification of change)

Patterns

Mono / Micro Services: Commonalities

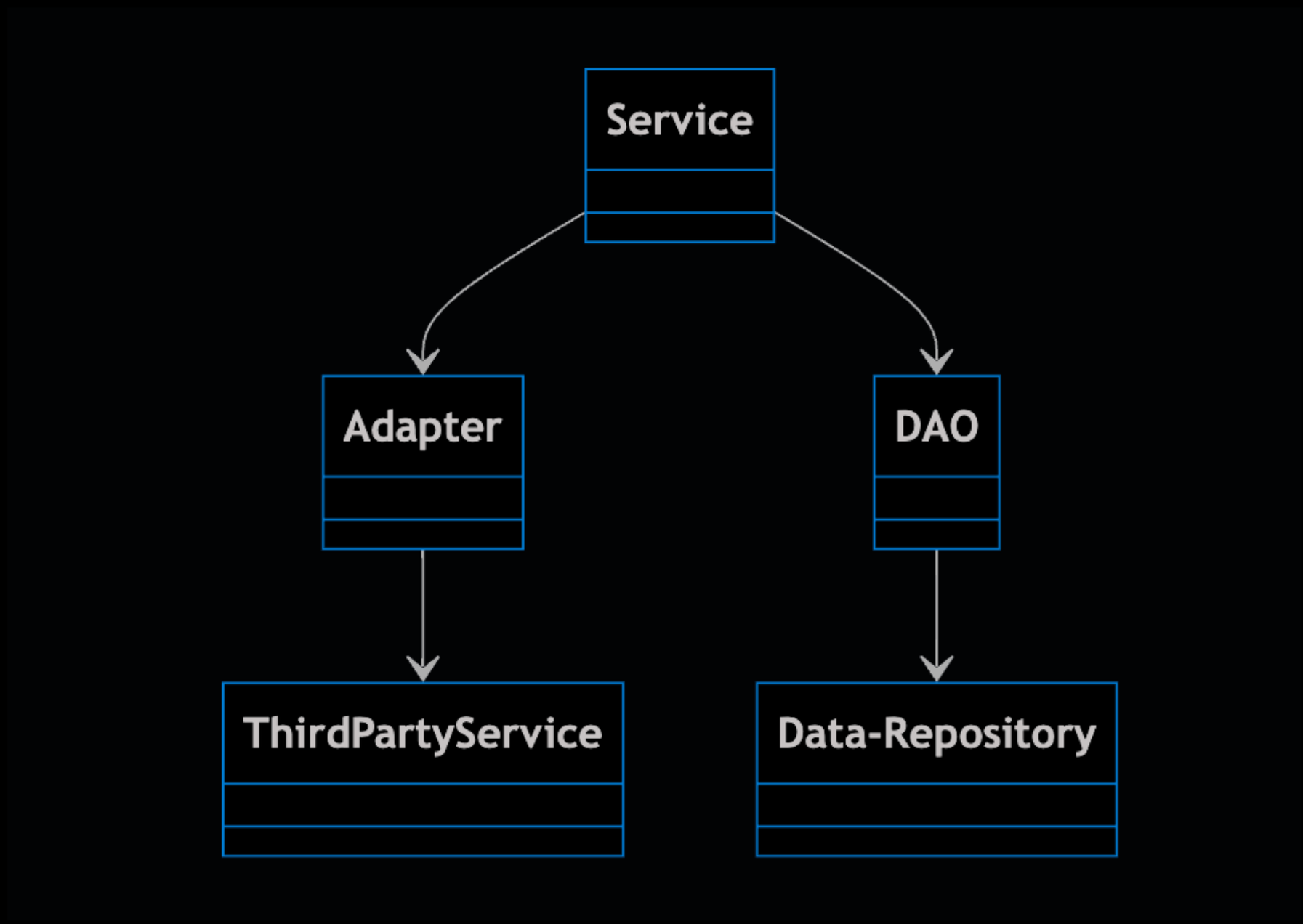
- Controller
- Service
- Model Objects
- Adapter
- Producers / Consumers
- Repository / DAO
- Domain Drive Design (CQRS)

Patterns



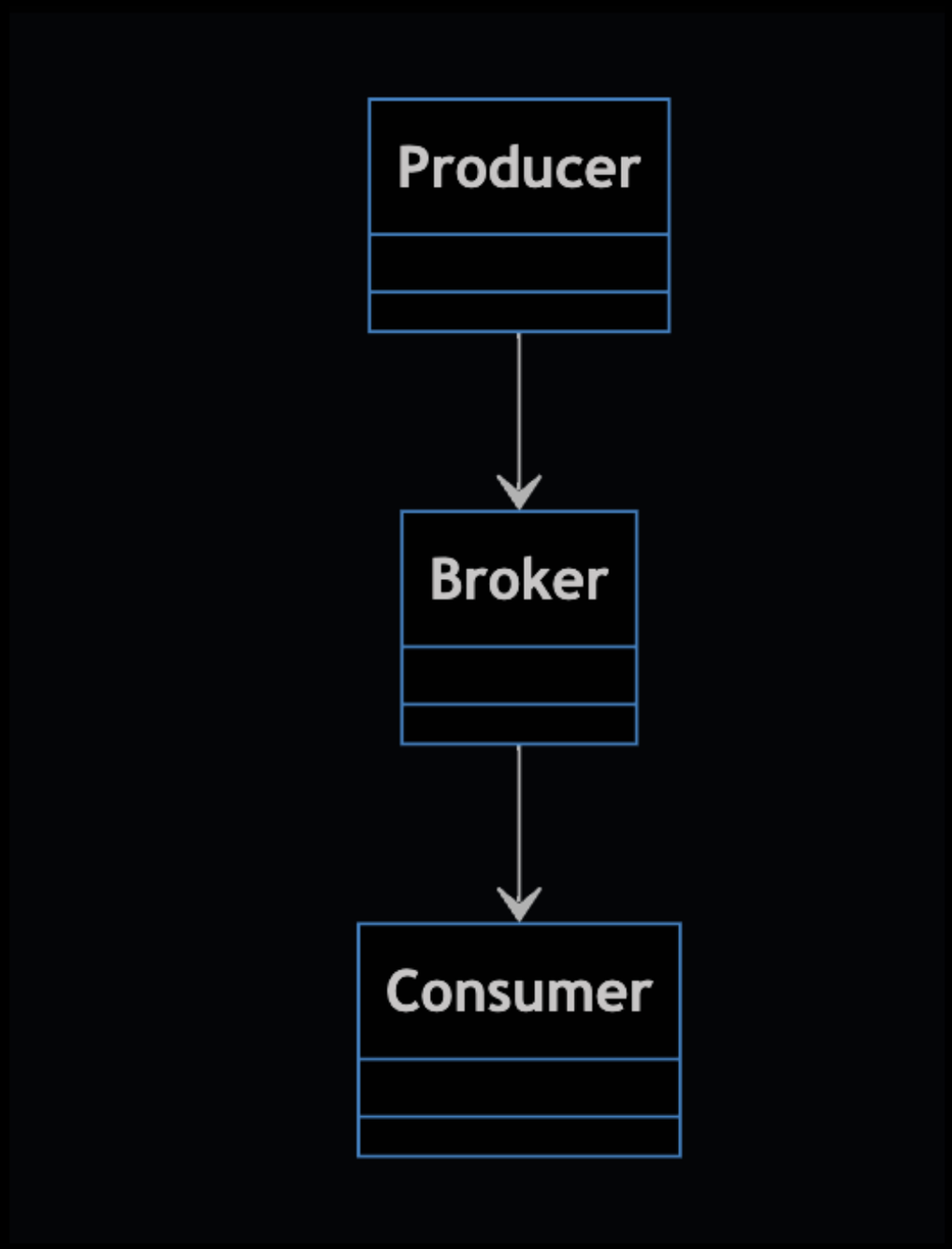
Controller-Service-Model

Patterns



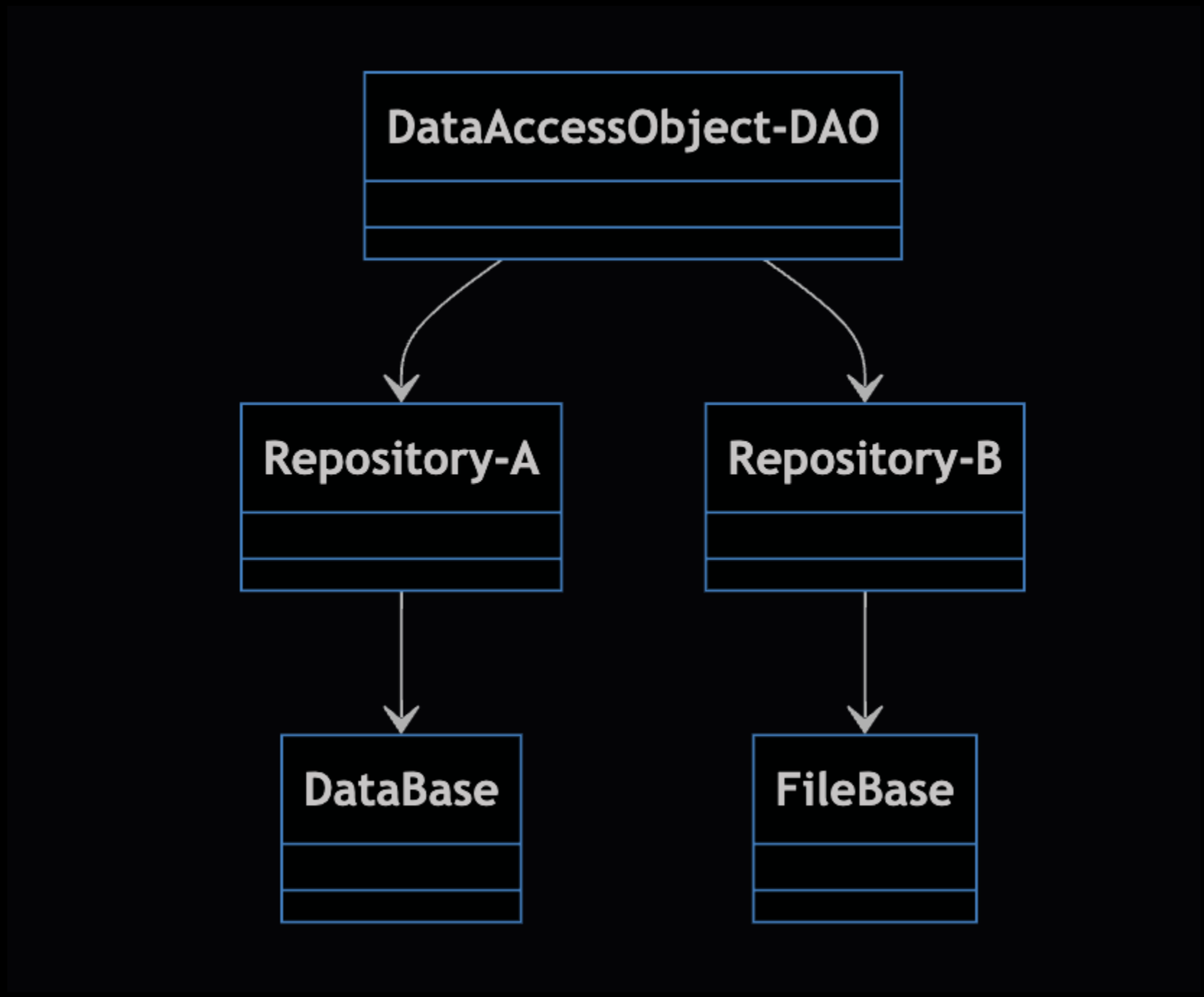
Service-Adapter-DAO

Patterns



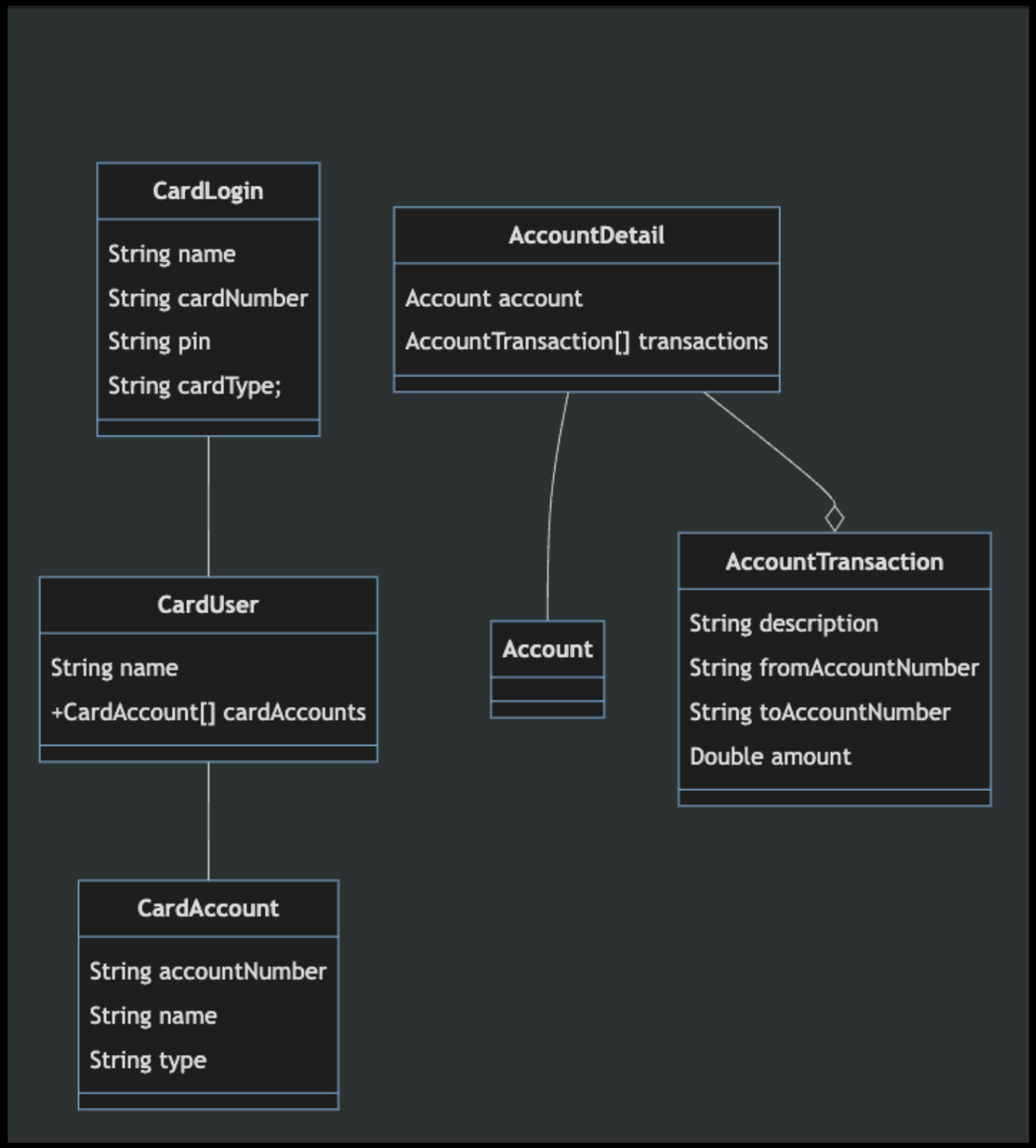
Producer-Broker-Consumer

Patterns

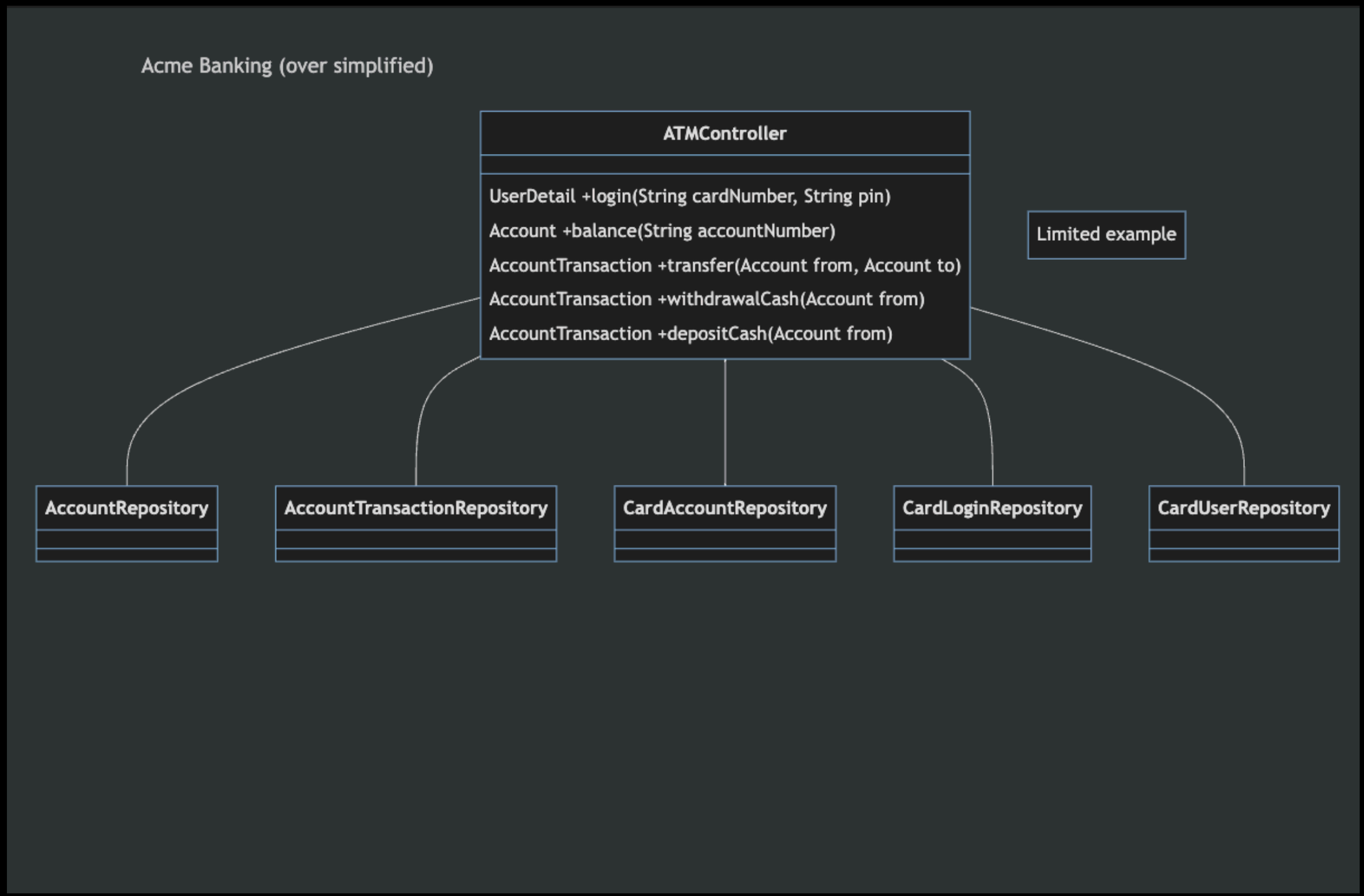


Producer-Broker-Consumer

Pattern Summary



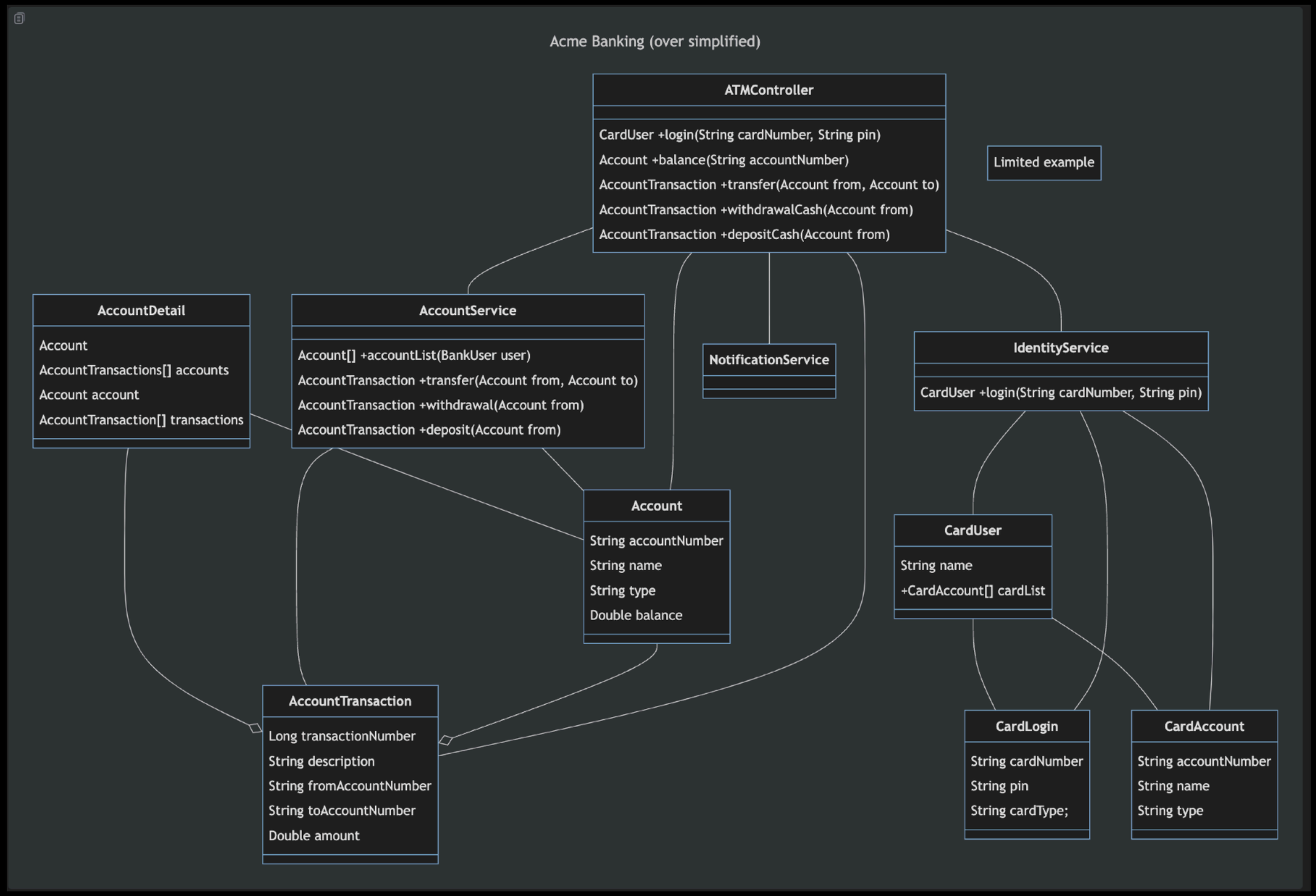
Pattern Summary



Pattern Summary

DEMO (messy vs cleaner)

Pattern Summary



Pattern Summary

- Patterns are used to decouple complexity, dependency, and layer against change

- Regardless of Monolith or Microservice, patterns help in structure and 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

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

Introduction to Testing

Notable Testing Quotes

"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

Introduction to Testing

Notable Testing Quotes

"deliver better software faster to production"

— Dave Farley

Introduction to Testing

Notable Testing Quotes

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

— Yoda

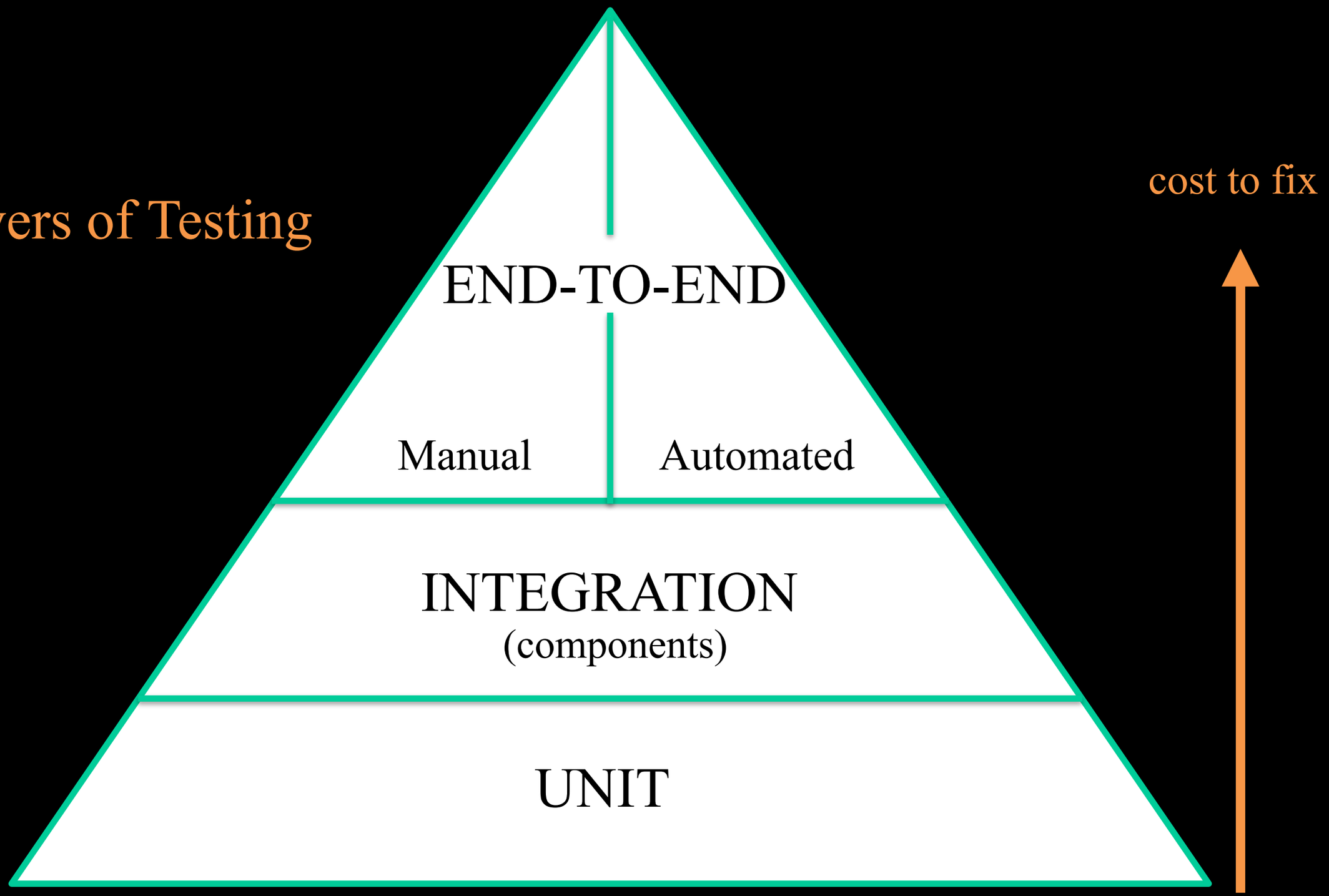
(Star Wars e3 - Revenge of the Sith, original script)

Testing sheds light on the unknown

LSE

Introduction to Testing

Layers of Testing



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

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

Introduction to Testing

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

Be careful about incentives …

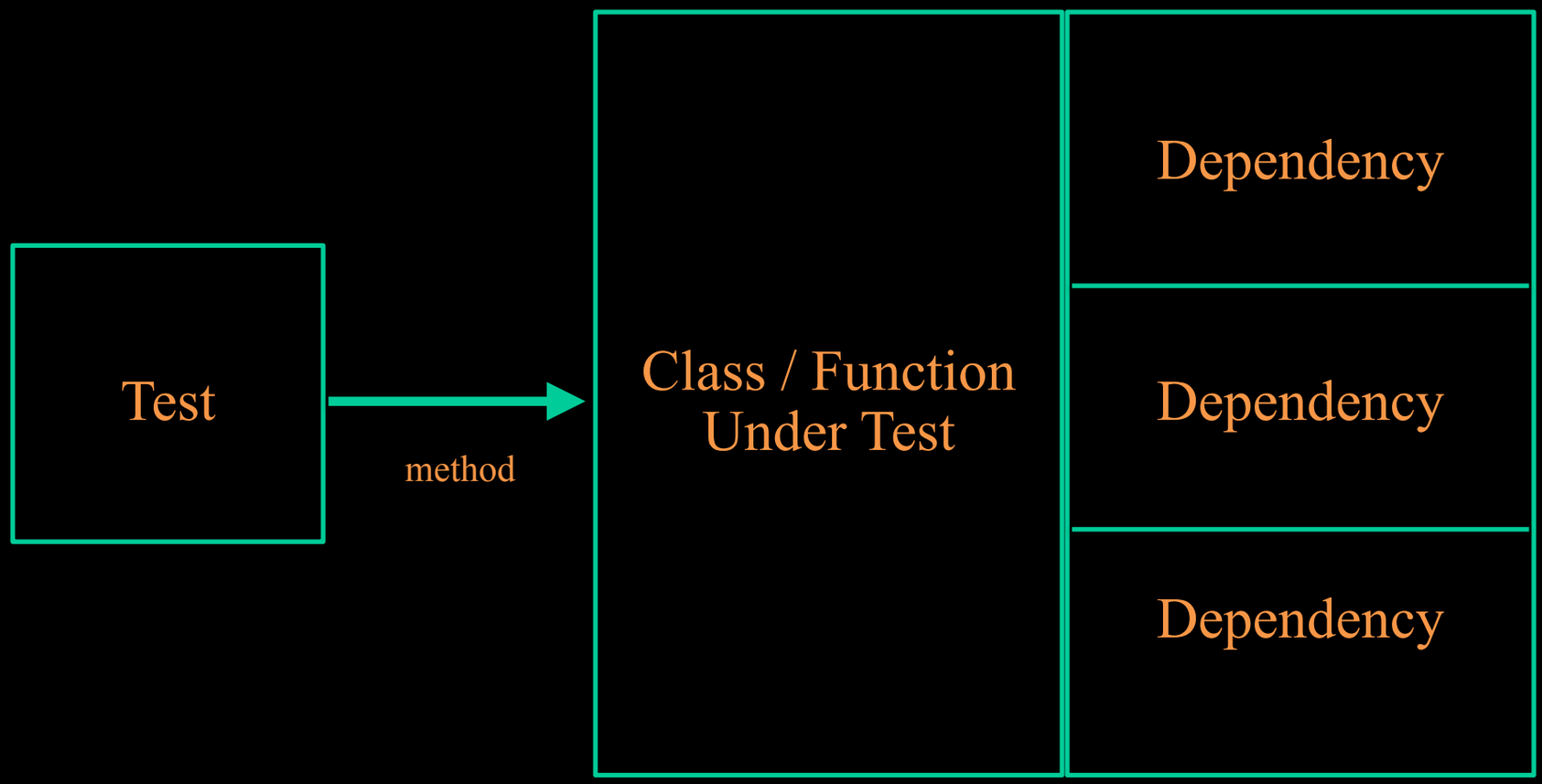
Testing Terms

Test Doubles

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

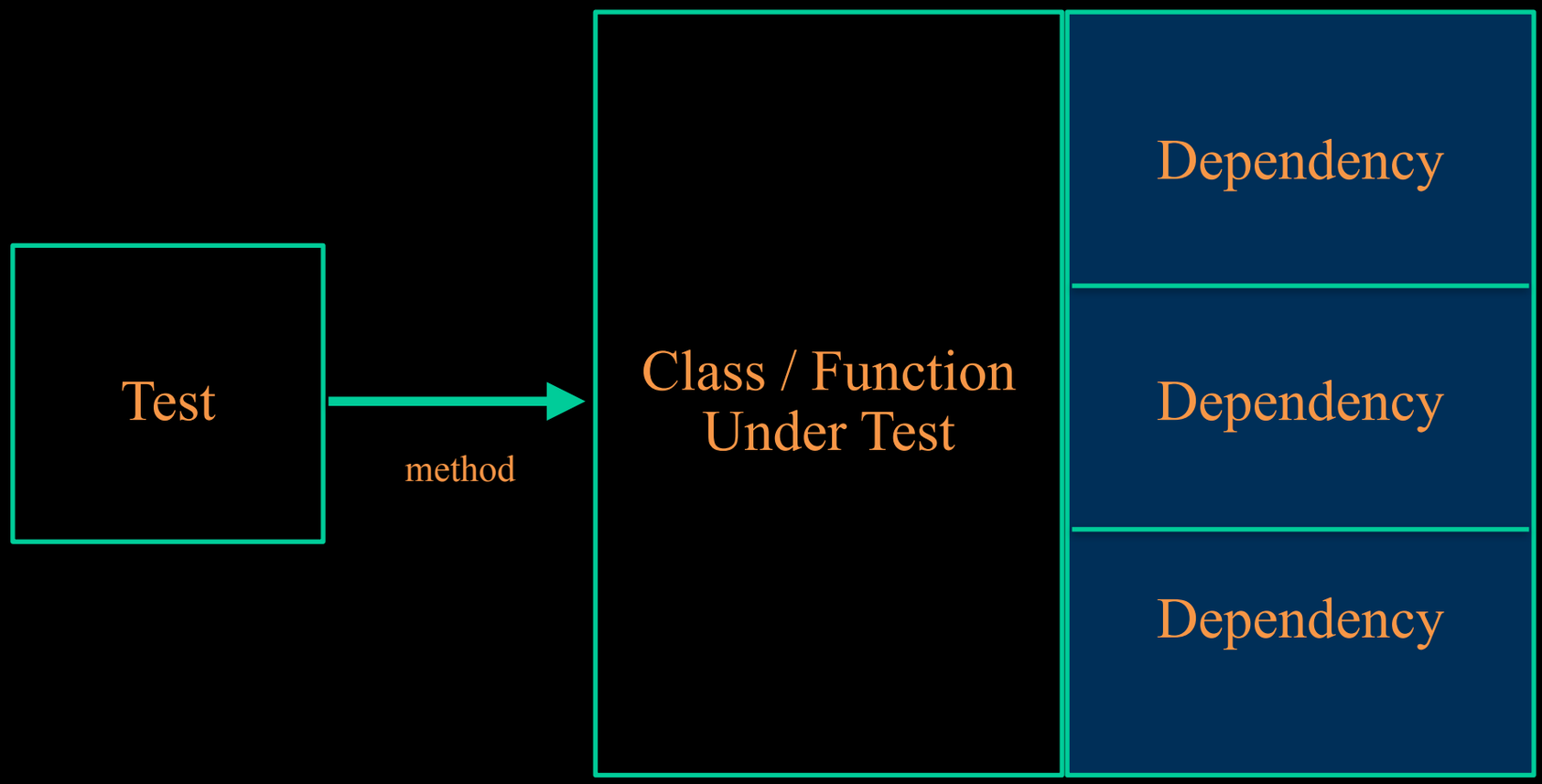
- Dummy objects: passed around but never called
- Fake objects: working non-prod implementations for testing
- Stubs: canned answers
- Spies: stubs that record information passed down to real implementation
- Mocks: objects pre-programmed with responses based on specification

Unit Testing



LSE

Unit Testing



Mock The Dependencies

LSE

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

Unit Testing

DEMO

LSE

Unit Testing Tips

- Use Test Factories 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)

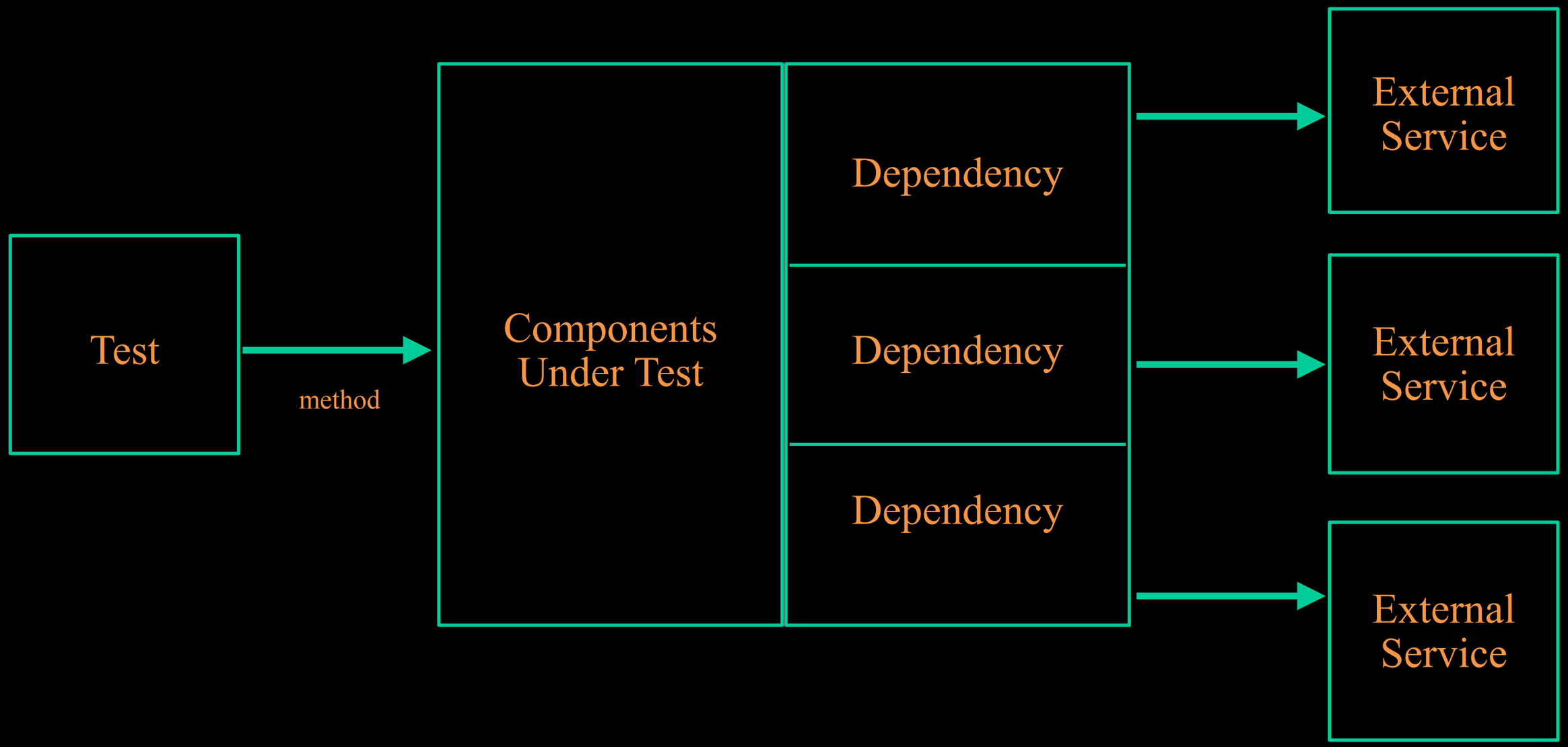
Unit Testing Tips

- Use Fake Data Generators
- Avoid mocking static functions
- Make sure tests are thread safe
- Make sure frameworks are thread safe
- Avoid using SpringBootTest (integration)

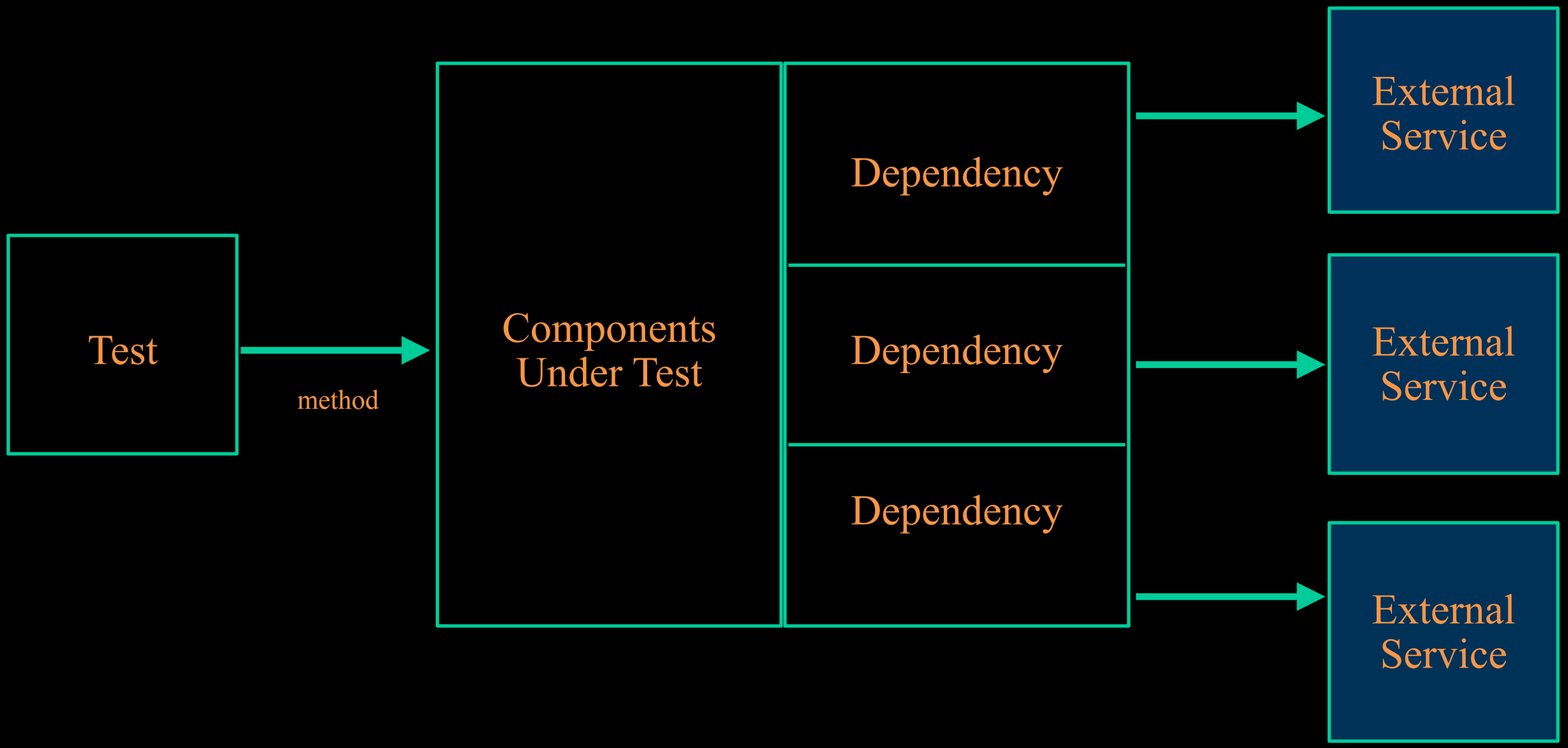
Unit Testing Tips

- 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 MicrobencHmarking (JMH) for isolated performance testing

Integration Testing



Integration Testing



Mock or Simulate or Use live System

verify components and dependencies execute as expected

LSE

Integration Testing Tips

DEMO

LSE

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

DEMO

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)

LSE

Advanced Topics

- Performance
- Load
- Chaos
- CI/CD

Advanced 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

Advanced 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)

Advanced 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)

Advanced 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)

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

LSE

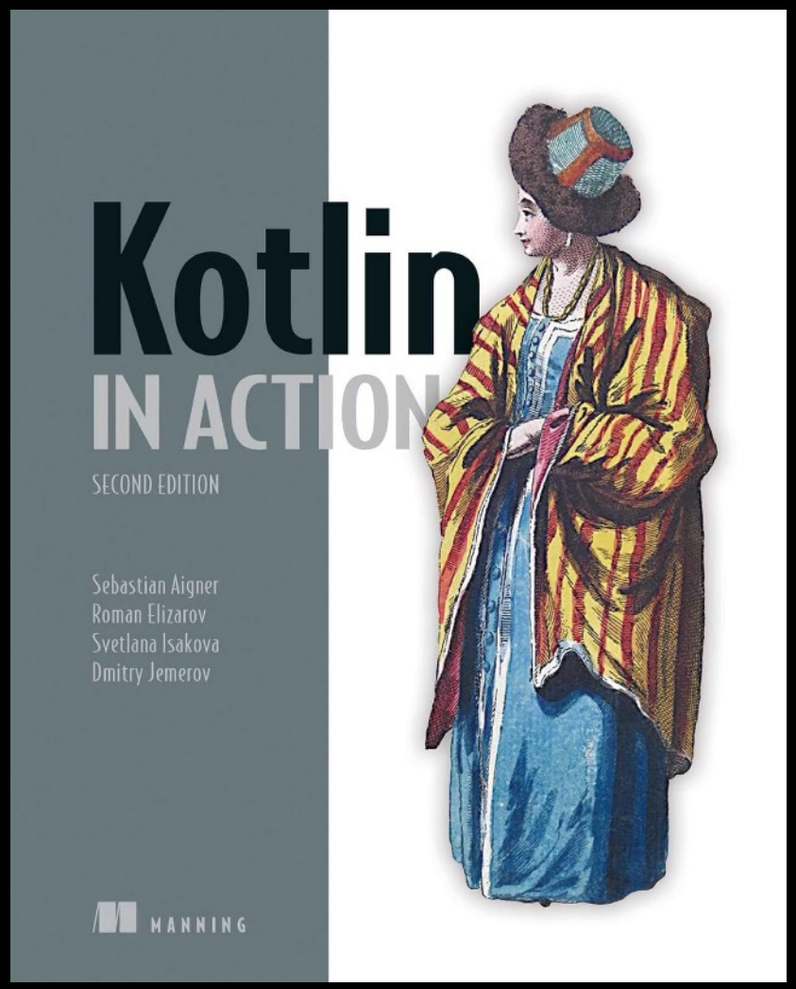
Questions ?

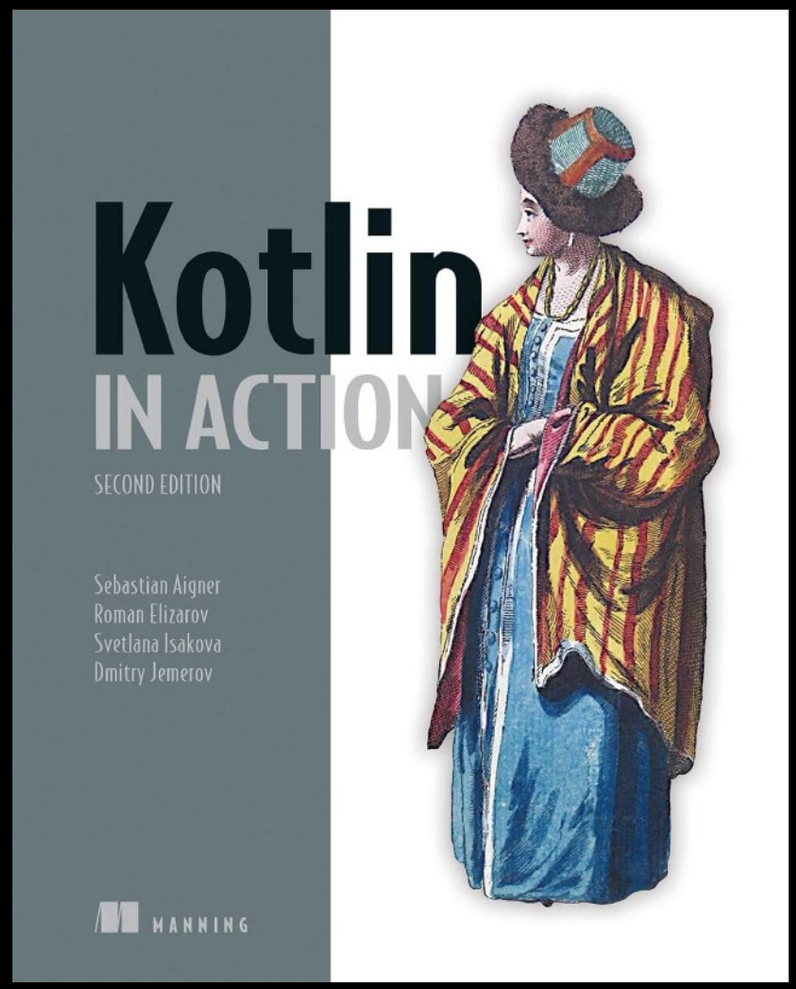
https://github.com/lseinc/cm23-break-monoliths-with-graphql.git

Don’t forget to fill
out the survey!

L
S
E

Gift Giveaway





Don't forget to fill out the survey!

LSE

Thank You ！

https://github.com/lseinc/cm25-backend-testing-strategies.git

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

L S E

https://github.com/lseinc/cm25-backend-testing-strategies.git

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

L S E