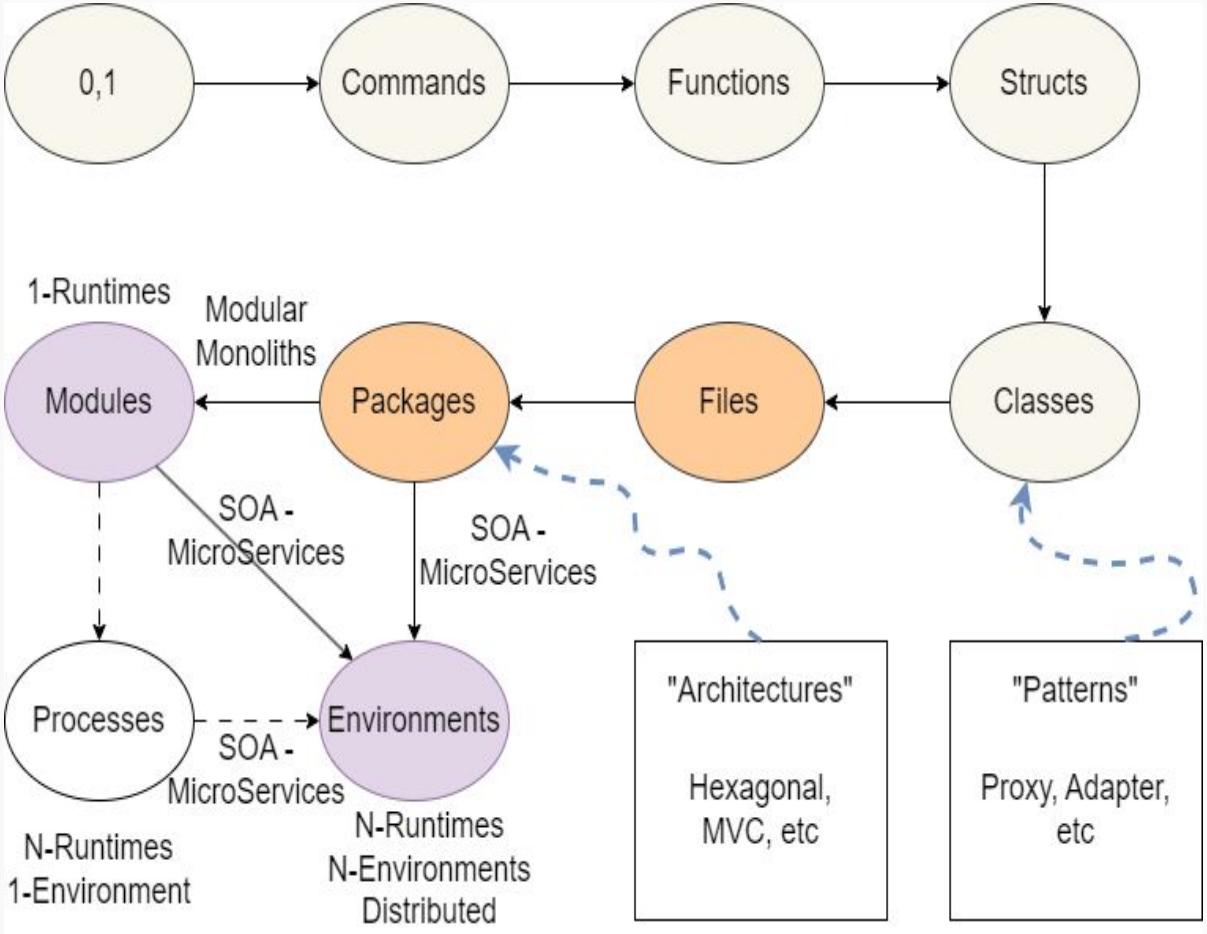
Developing Distributed Systems with Modular Monoliths and Microservices

Student: Michail Tsechelidis

Advisor: Apostolos Ampatzoglou

HELLENIC REPUBLIC
UNIVERSITY OF MACEDONIA

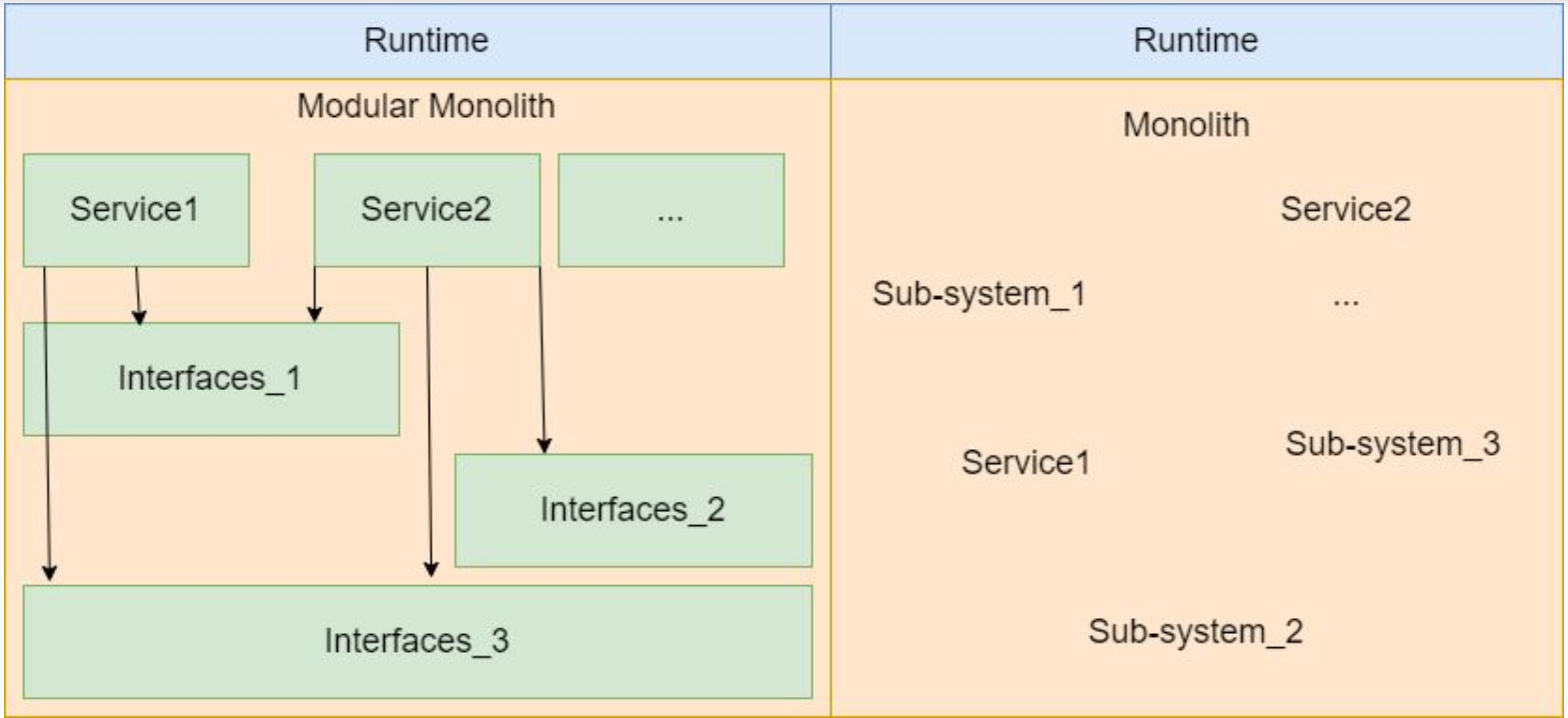
Grouping Information



Types of Requirements

- Software requirements
- Hardware requirements
- Development requirements
- Business requirements

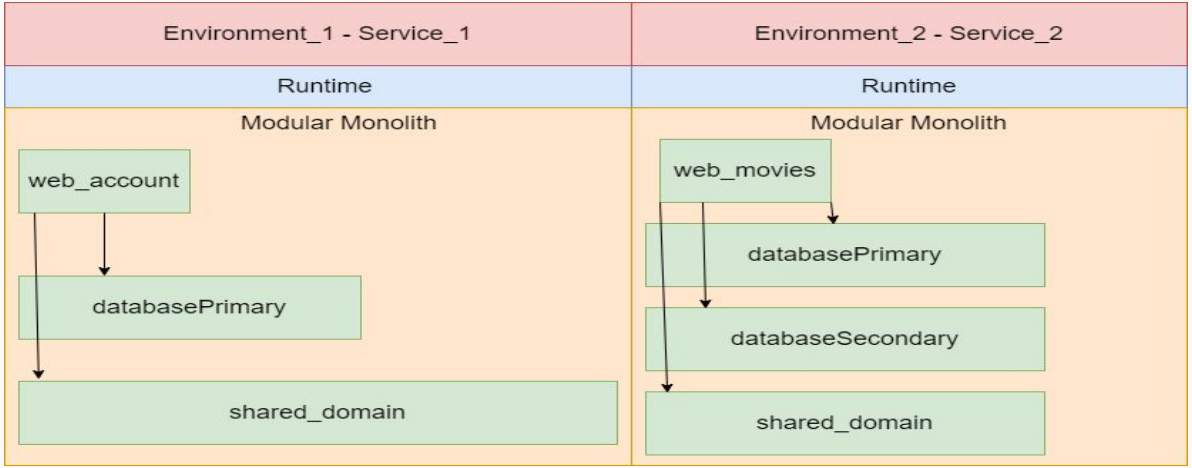
Modular Monolith?



Why?

- Resolve Development Requirements
- Control Granularity with ease
- Aim for Simpler Infrastructures (ex. people, networks, pipelines, ….)

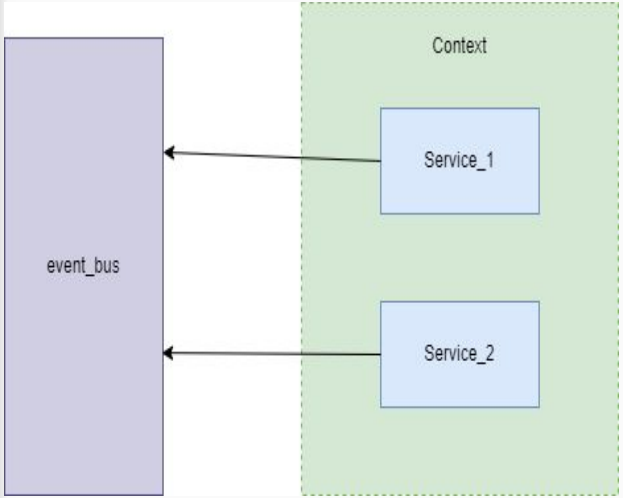
Controlling Granularity

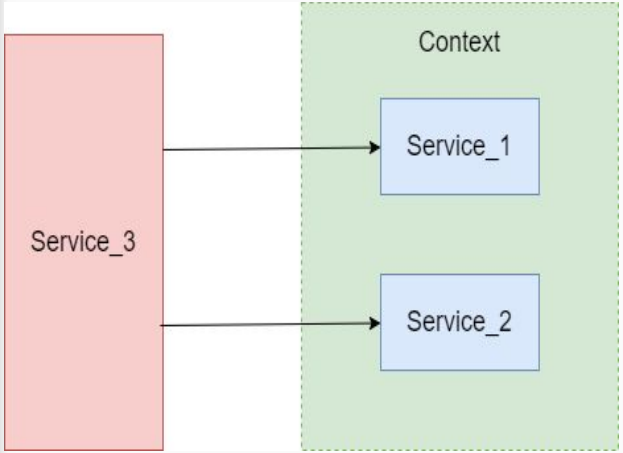


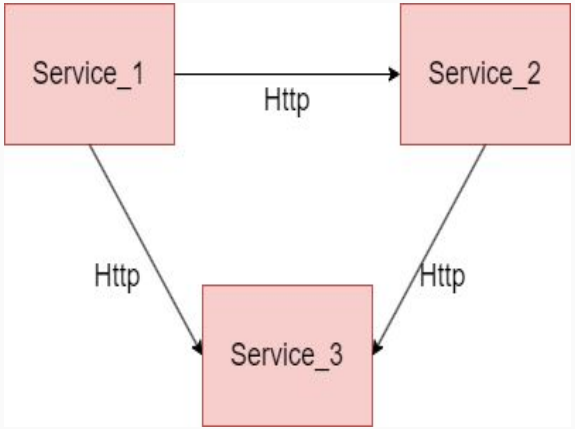

```xml
<dependencies>

    <dependency>
        <groupId>tsechelidisMichail</groupId>
        <artifactId>web_account</artifactId>
        <version>${version}</version>
        <scope>runtime</scope>
    </dependency>

</dependencies>
```

```xml
<dependencies>

    <dependency>
        <groupId>tsechelidisMichail</groupId>
        <artifactId>web_movie</artifactId>
        <version>${version}</version>
        <scope>runtime</scope>
    </dependency>

</dependencies>
```

Service Oriented Architectures







Microservices?

- Software requirements
- Hardware requirements
- Horizontal scaling?
  (Business requirement)
- Development requirements

No definition ⟹ Dubious properties

- What is small?
- Loose coupling ≠ **Independence**

Scaling

- M5 (4 CPUs, 16G memory)
- Max = 20000 Requests/second
- CPU usage = 99%
- 56% UI - 0,84% Logic

- Aim for SLA = 99,999%
- Deployment Strategies
- Scale at Lower CPU usage?
- Vertical or Horizontal?

Cloud Patterns

- Enhance the system
- Applied to any SOA

Examples:

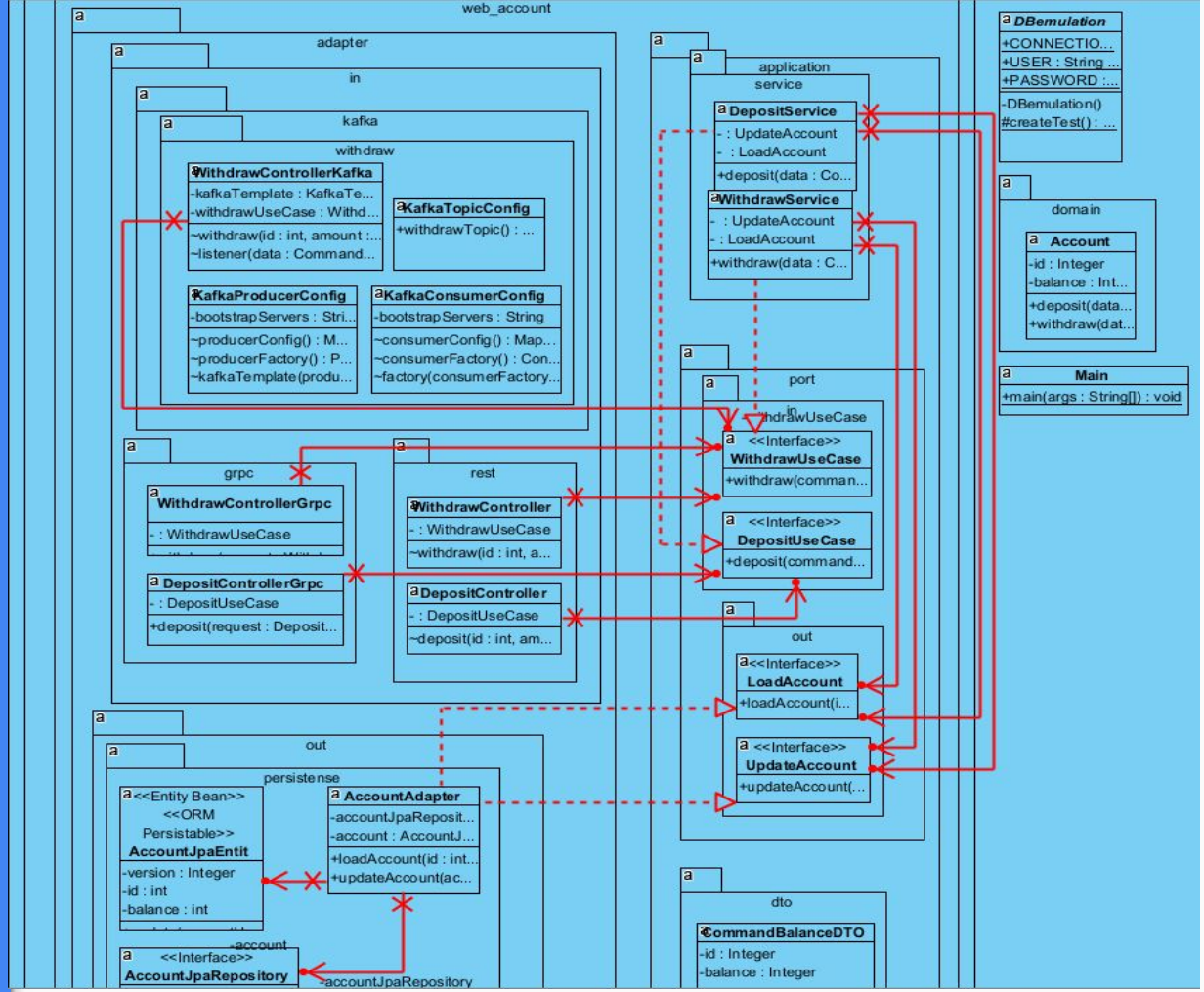
- LoadBalancer (Scalability)
- CQRS (Performance)
- Sidecar (Extensibility)
- Retry (Reliability)
- Canary release (Deployment)

Hexagonal Architecture

- Simple starting point

- Easy identification & extraction of internal functionalities

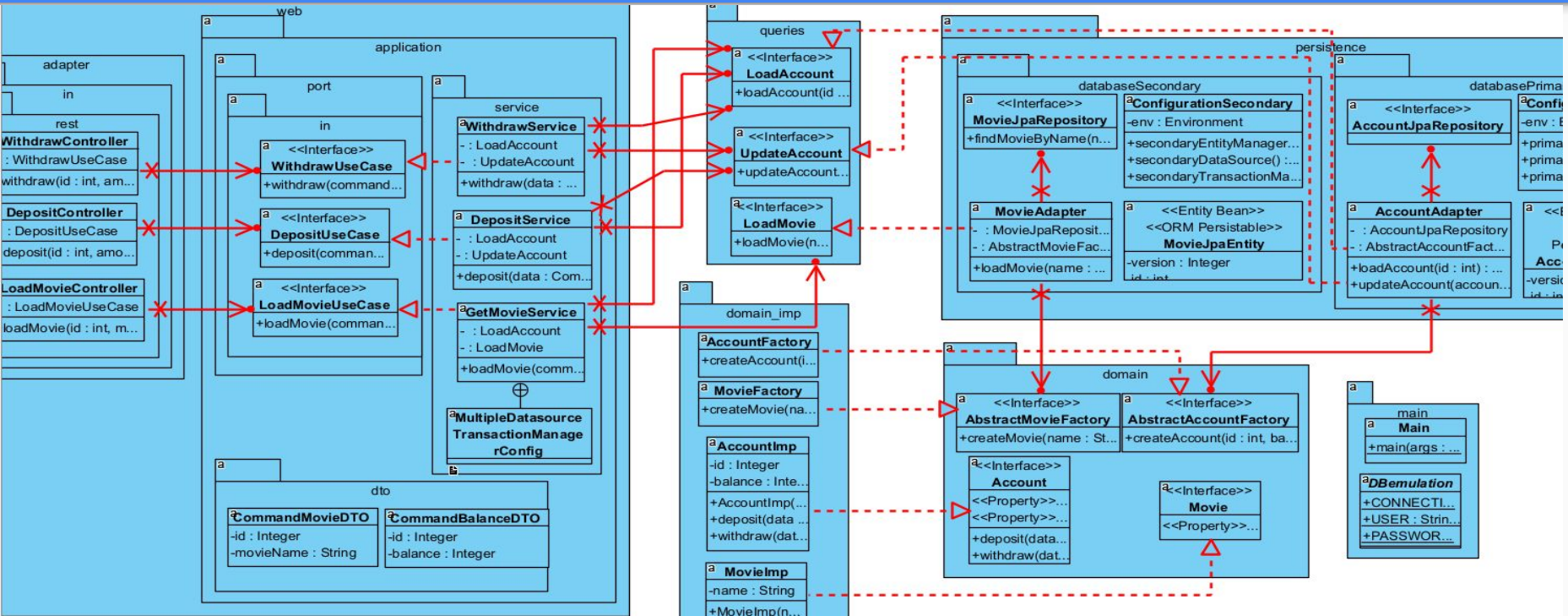
A Monolith with Dependency Injection



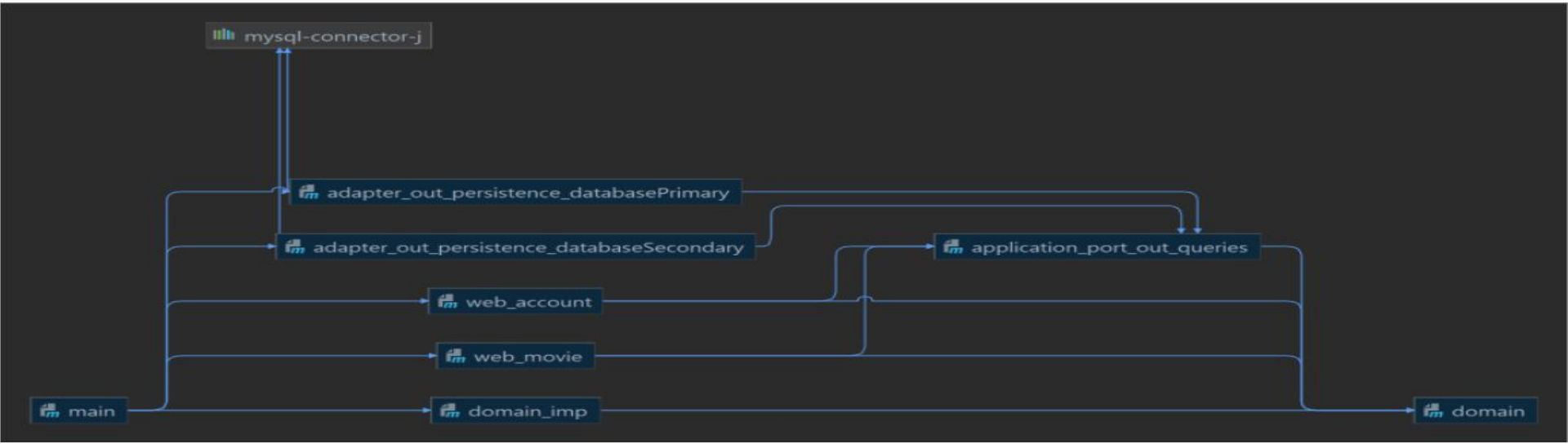
Patterns

- Dependency Injection
- Separated Interfaces
- Layer Supertype
- Abstract Factory

The Monolith Modularized



Reversed Dependency diagram



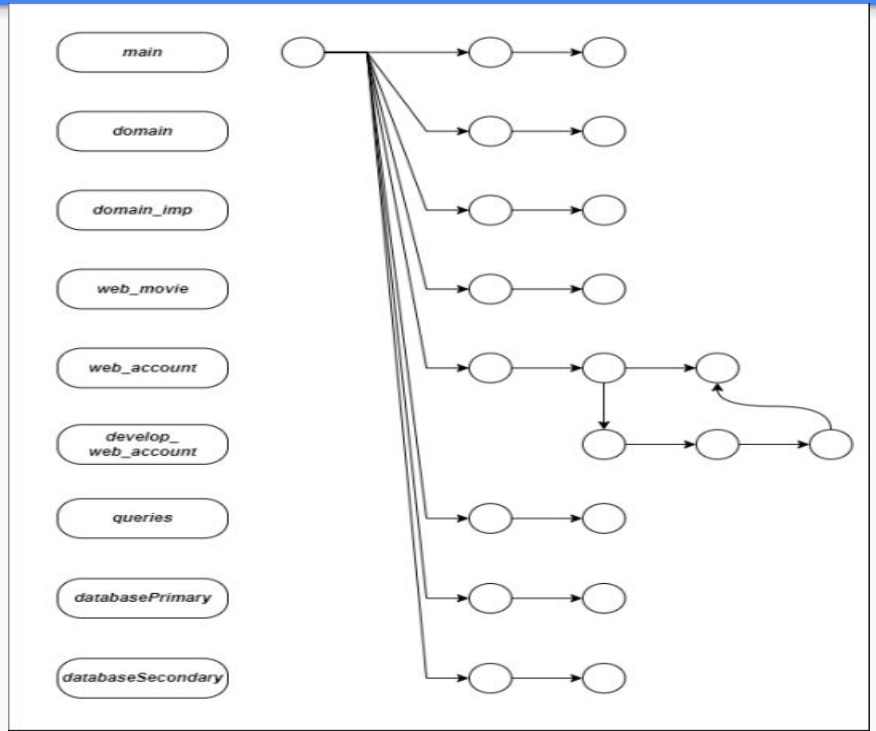
Modular Compiles

Initial state

After change

Larger systems have more Development Requirements

- Git Branches
- Pipelines (ex. Github Actions)
- Artifact Repository (ex. Github Packages)



Case Studies

Amazon:

- 90% reduced costs
- Same domain

Shopify:

- 1.27 million RPS
- SLA = 99.999%

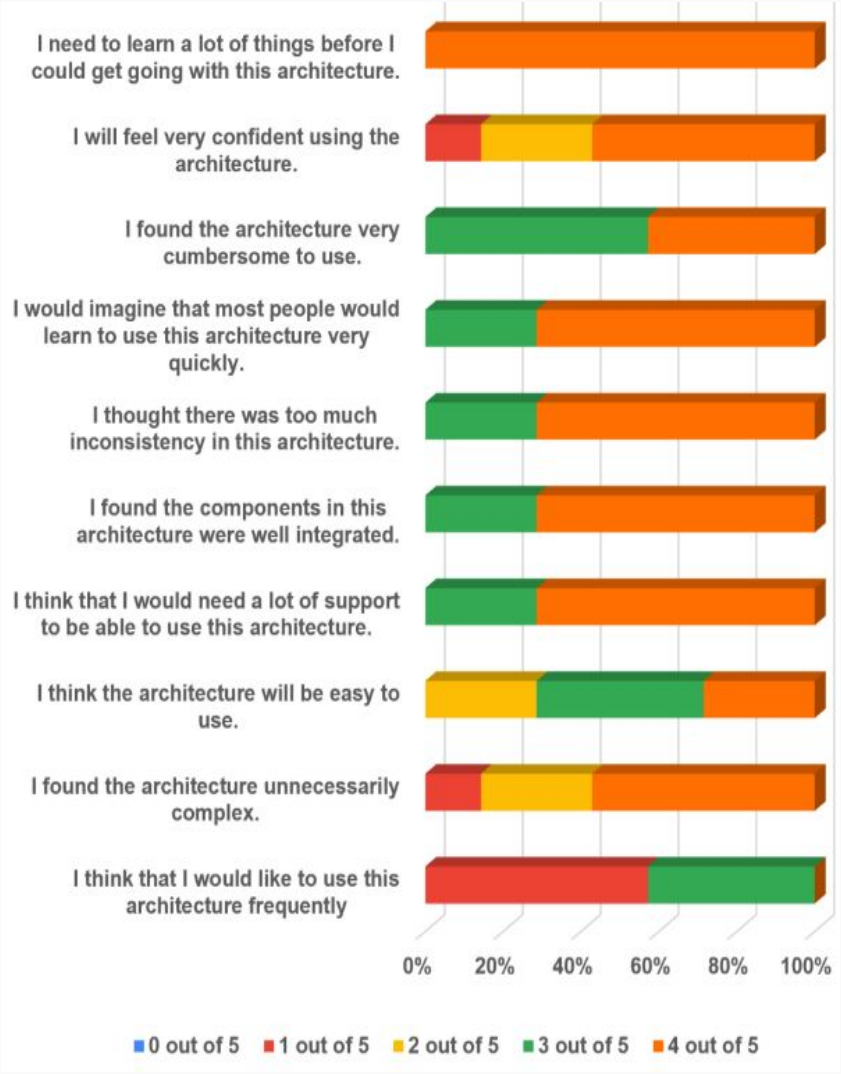
Simpler Infrastructures

⇒

- Less costs
- Easy Management

- Team Topologies
  - Independent teams
- Agile Development
  - Asynchronous centralized dependencies
- Continuous Integration and Delivery
  - Reusable pipelines - Monorepo
- Networks in the Cloud
  - Latency - Less hops
- Resource savings - Green Computing
  - 25% of 16.000 cloud servers **no** useful work

Validation


I need to learn a lot of things before I could get going with this architecture.
I will feel very confident using the architecture.
I found the architecture very cumbersome to use.
I would imagine that most people would learn to use this architecture very quickly.
I thought there was too much inconsistency in this architecture.
I found the components in this architecture were well integrated.
I think that I would need a lot of support to be able to use this architecture.
I think the architecture will be easy to use.
I found the architecture unnecessarily complex.
I think that I would like to use this architecture frequently
0%
20%
40%
60%
80%
100%
0 out of 5
1 out of 5
2 out of 5
3 out of 5
4 out of 5

Thank you!

Questions?