SOFTWARE ARCHITECTURE

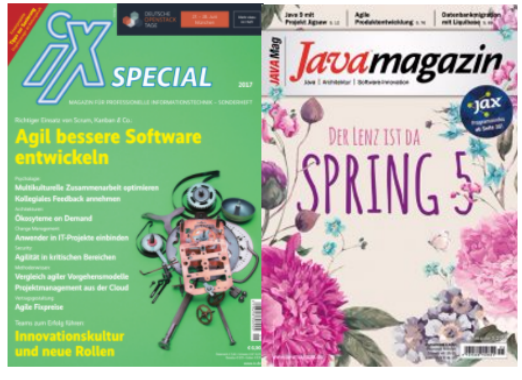
Andreas Falk

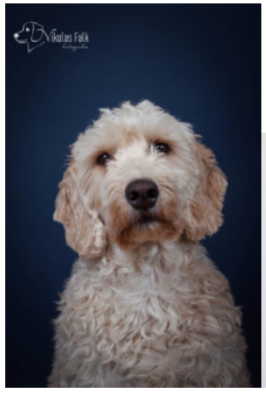
10.11.2017

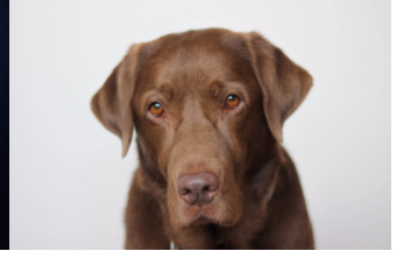
Presentation and Workshop:
https://github.com/nt-ca-aqe/ws-2017-11-10-architecture

ANDREAS FALK

andreas.falk@novatec-gmbh.de







AGENDA

- Software Craftsmanship
- Software Architecture
- Architectural Patterns
- Design Patterns & Principles
- Lunch Break
- Labs: Layered Architecture with Spring
- Q&A
- Feedback

REQUIREMENTS FOR WORKSHOP

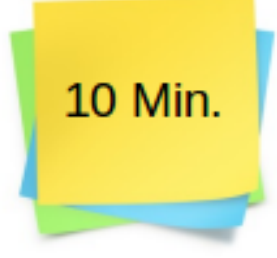
- Java 8
- Maven
- IntelliJ or Eclipse IDE (STS)

PARTICIPANTS INTRODUCTION

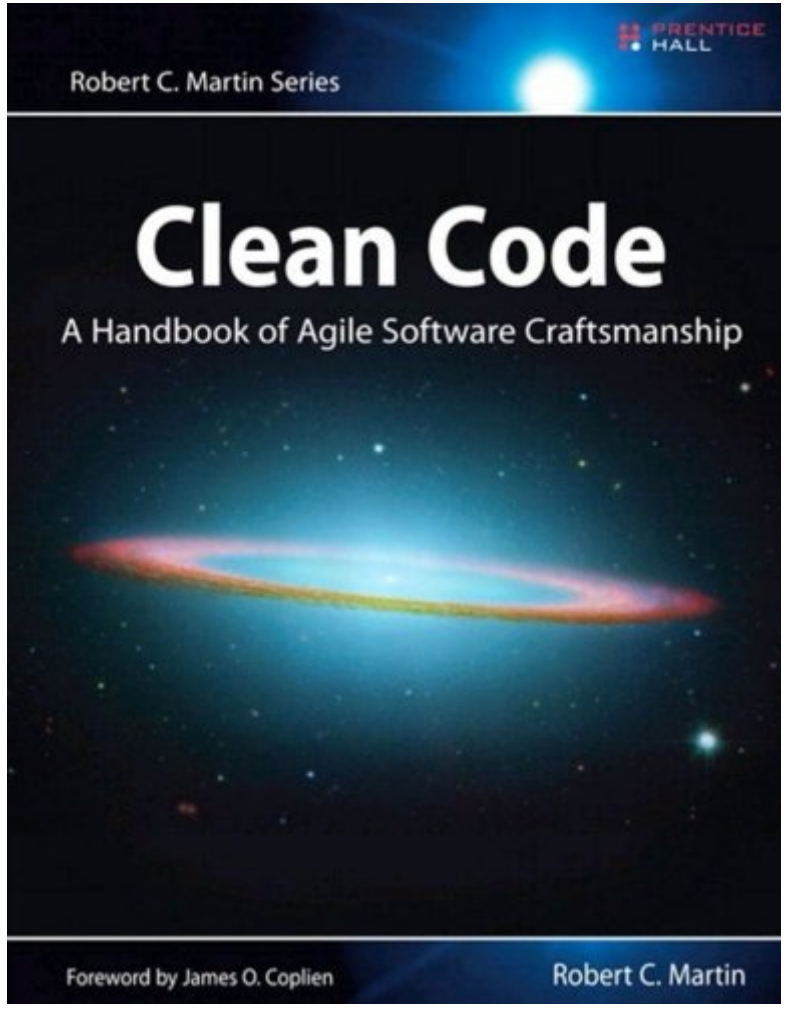
What is your name and your project role?

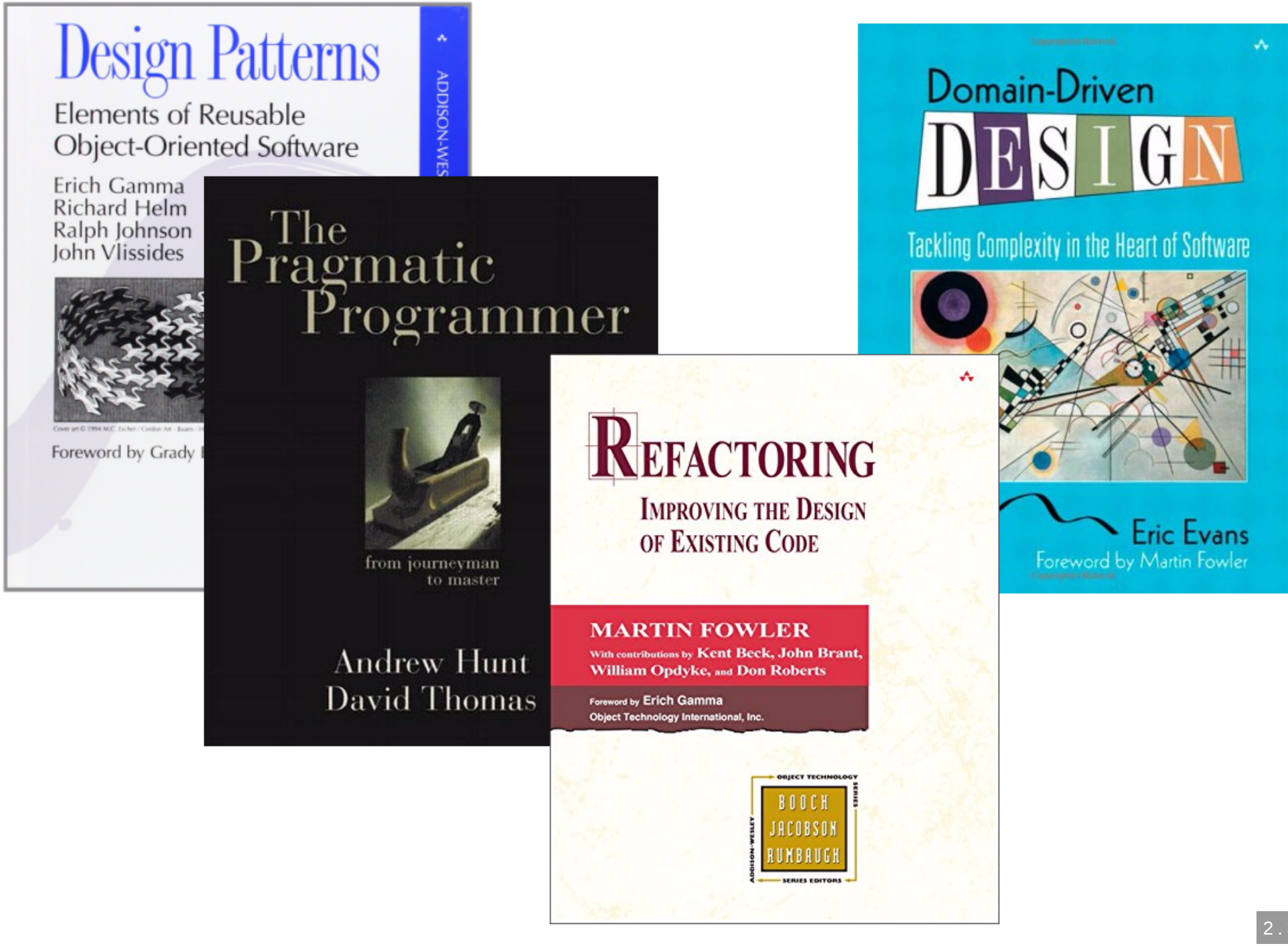
What do you want to learn about *Software Architecture*?

Do you have specific project-related questions?



SOFTWARE CRAFTSMANSHIP


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady
The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Domain-Driven DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler
REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International, Inc.
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
SERIES EDITORS
ADDISON-WESLEY

"WE NEED TO BE PROFESSIONALS"

"WE WILL NOT SHIP SHIT"

BOB MARTIN

Demanding Professionalism in Software Development

"NOT JUST CODE MONKEYS"

MARTIN FOWLER

OOP 2014 Keynote

AGILE MANIFESTO

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Craftsmanship over Execution *)

http://agilemanifesto.org

*) https://www.infoq.com/news/2008/08/manifesto-fifth-craftsmanship

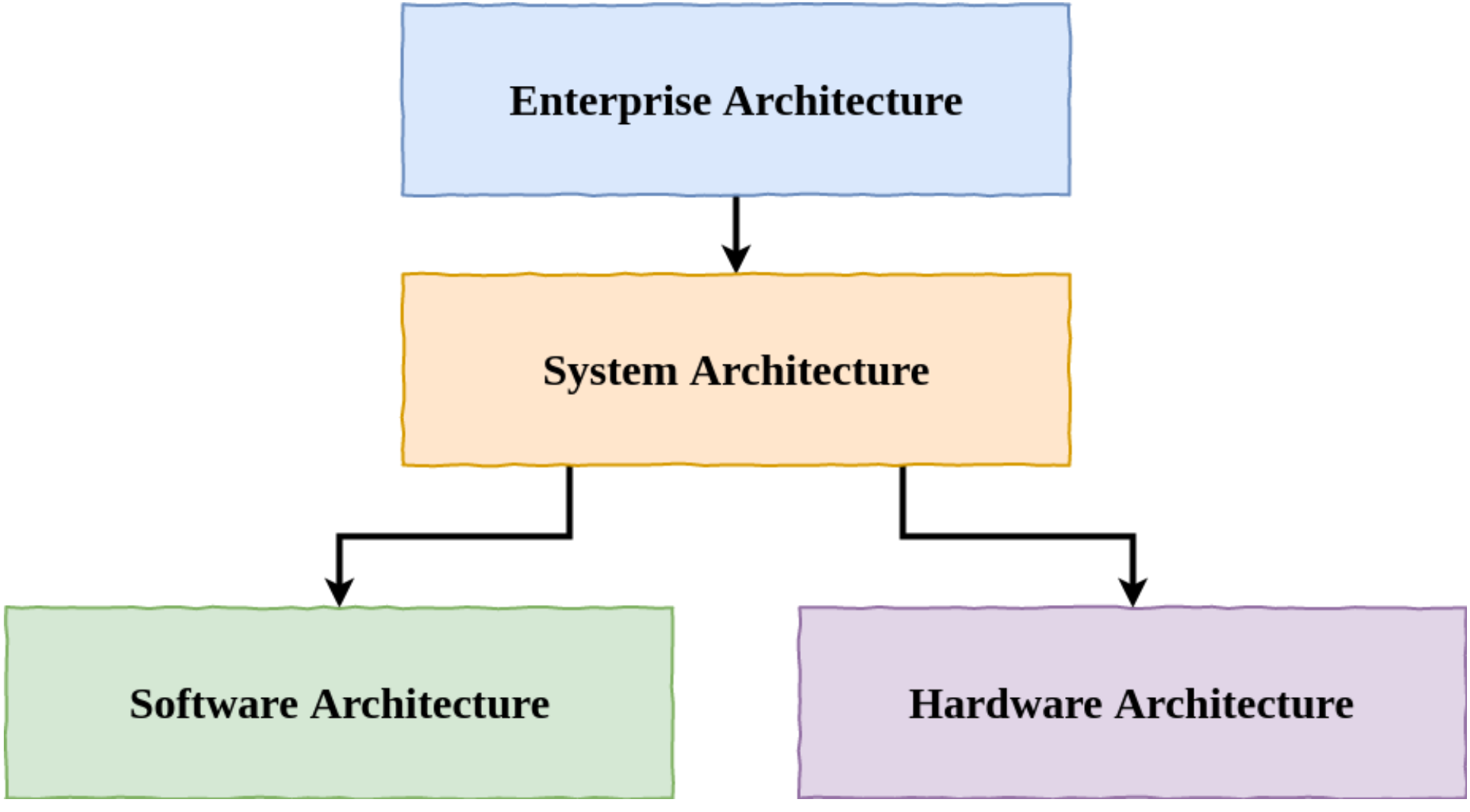

Clean Code
Clean Architecture
Object Oriented Design Patterns
Object Oriented Design Principles
Software Architecture
Domain Driven design
Refactoring
Pair Programming
Code Review
Coding Dojos
Test Driven Design (TDD)

ENFORCING CLEAN CODE

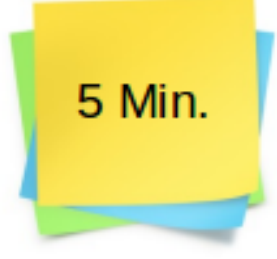
- Pair Programming
- Code Reviews
- Static Code Analysis (e.g. PMD, FindBugs)
- Coding Dojos

ARCHITECTURE

ARCHITECTURE LAYERS



WHAT IS SOFTWARE ARCHITECTURE?



“Software architecture refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures...
...Software architecture is about making fundamental structural choices which are costly to change once implemented.”

Wikipedia

“Architecture is about making some important decisions that we'll have some long lasting effects on your system.
If you ignore architecture and rush into software development by trying to drive software development solely with the functional visible part of it, you are running very rapidly into technical debt.”

Philippe Kruchten (2010)

GOAL OF ARCHITECTURE

"The goal of software architecture is to minimize the human resources required to build and maintain the required system."

Bob Martin (2017)

ARCHITECTURAL CONCERNS

RESILIENCE

SECURITY

DEPLOYABILITY

TESTABILITY

RESILIENCE

The ability of an app to recover from certain types of failure and yet remain functional from the customer perspective.

- Mean Time Between Failures (MTBF)
- Mean Time to Repair/Recover (MTTR)

SECURITY

EU GENERAL DATA PROTECTION REGULATION (GDPR)

"...should adopt internal policies and implement measures which meet in particular the principles of data protection by design and data protection by default"

http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679 (78)

A3: SENSITIVE DATA EXPOSURE

OWASP Top 10 2013	±	OWASP Top 10 2017
A1 – Injection	→	A1:2017 – Injection
A2 – Broken Authentication and Session Management	→	A2:2017 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	↘	A3:2013 – Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017 – XML External Entity (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017 – Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017 – Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017 – Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017 – Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017 – Insufficient Logging & Monitoring [NEW, Comm.]

OWASP Top 10 2017 RC 2 - https://github.com/OWASP/Top10

DEPLOYABILITY

“Releasing to production should be a boring stupid process”

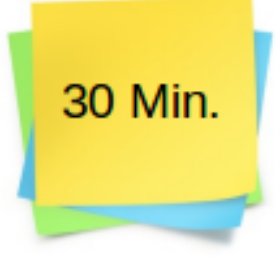
Jez Humble (Author of book "Continuous Delivery")

TESTABILITY

Test automation is key for Continuous Delivery!

Clean Code Architectural Patterns S.O.L.I.D. Design Patterns Domain Driven Design Decisions Design Thinking Consultancy Software Communication Riskmanagement K.I.S.S. Separation Of Concerns

WHAT ABOUT YOUR ARCHITECTURAL CONCERNS?



ARCHITECTURAL PATTERNS

“An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context.
Architectural patterns are similar to software design patterns but have a broader scope.”

Wikipedia

“Architecture patterns help define the basic characteristics and behavior of an application.”

Software Architecture Patterns (O'Reilly, 2015)

CLIENT/SERVER

FAT/THIN CLIENT - FAT SERVER

CORBA

ENTERPRISE APPLICATION INTEGRATION

“A collection of technologies and services which form a middleware or "middleware framework" to enable integration of systems and applications across an enterprise.”

Wikipedia

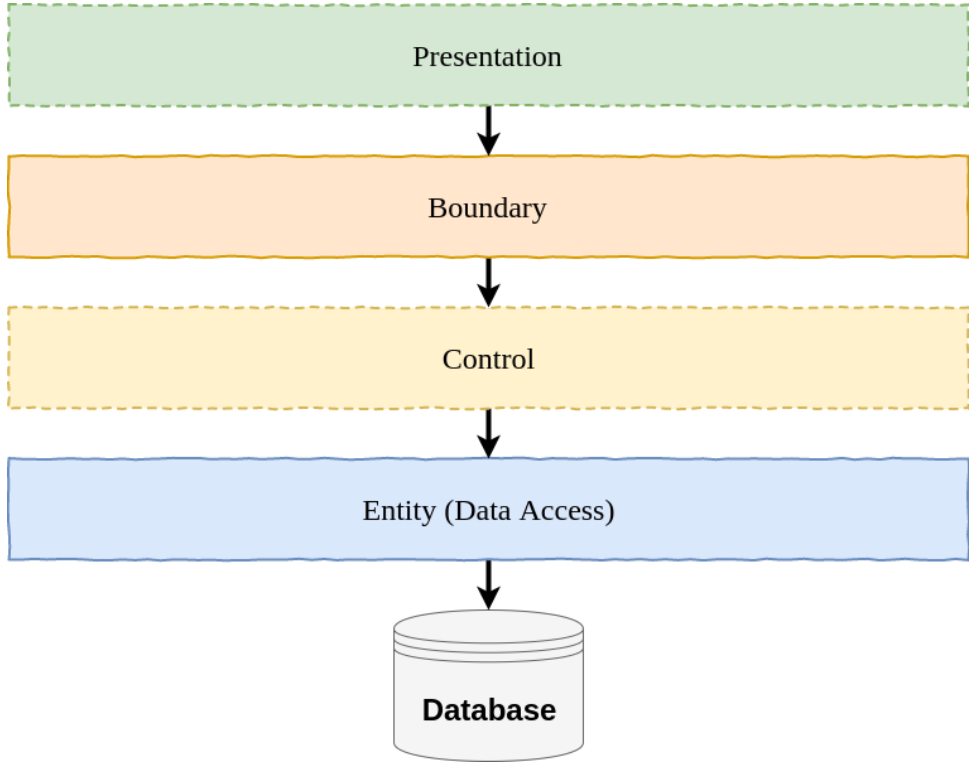
SERVICE ORIENTED ARCHITECTURE (SOA)

ENTERPRISE SERVICE BUS (ESB)

“WATER-FALL FAT MICROSERVICES”

LAYERED ARCHITECTURE

“ENTITY-CONTROL-BOUNDARY-PATTERN”



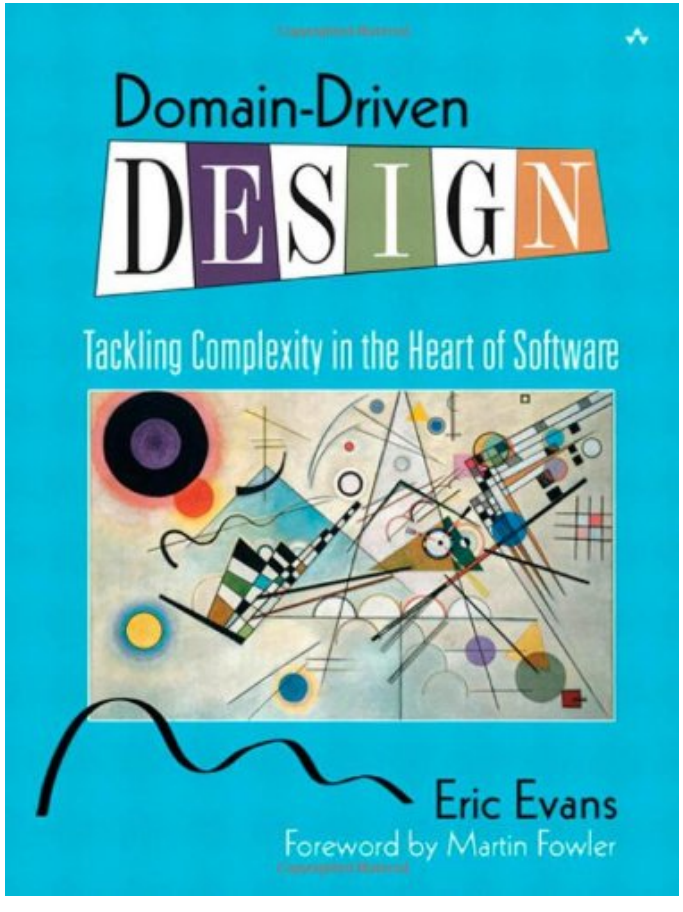
Structuring Java EE 7 Apps (Adam Bien)

MODEL VIEW CONTROLLER (MVC)

“Pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts.”

Wikipedia

DOMAIN DRIVEN DESIGN (DDD)



DDD @ https://martinfowler.com

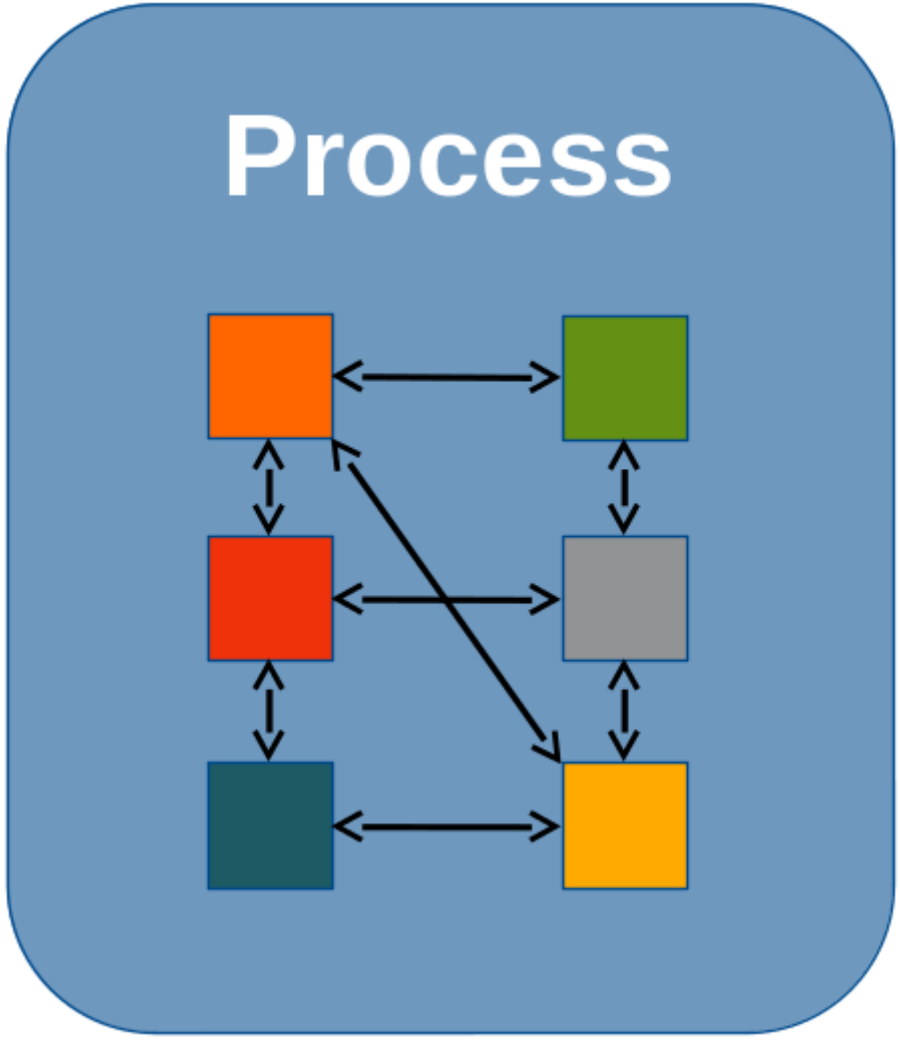
MICROSERVICES

“Agile SOA”

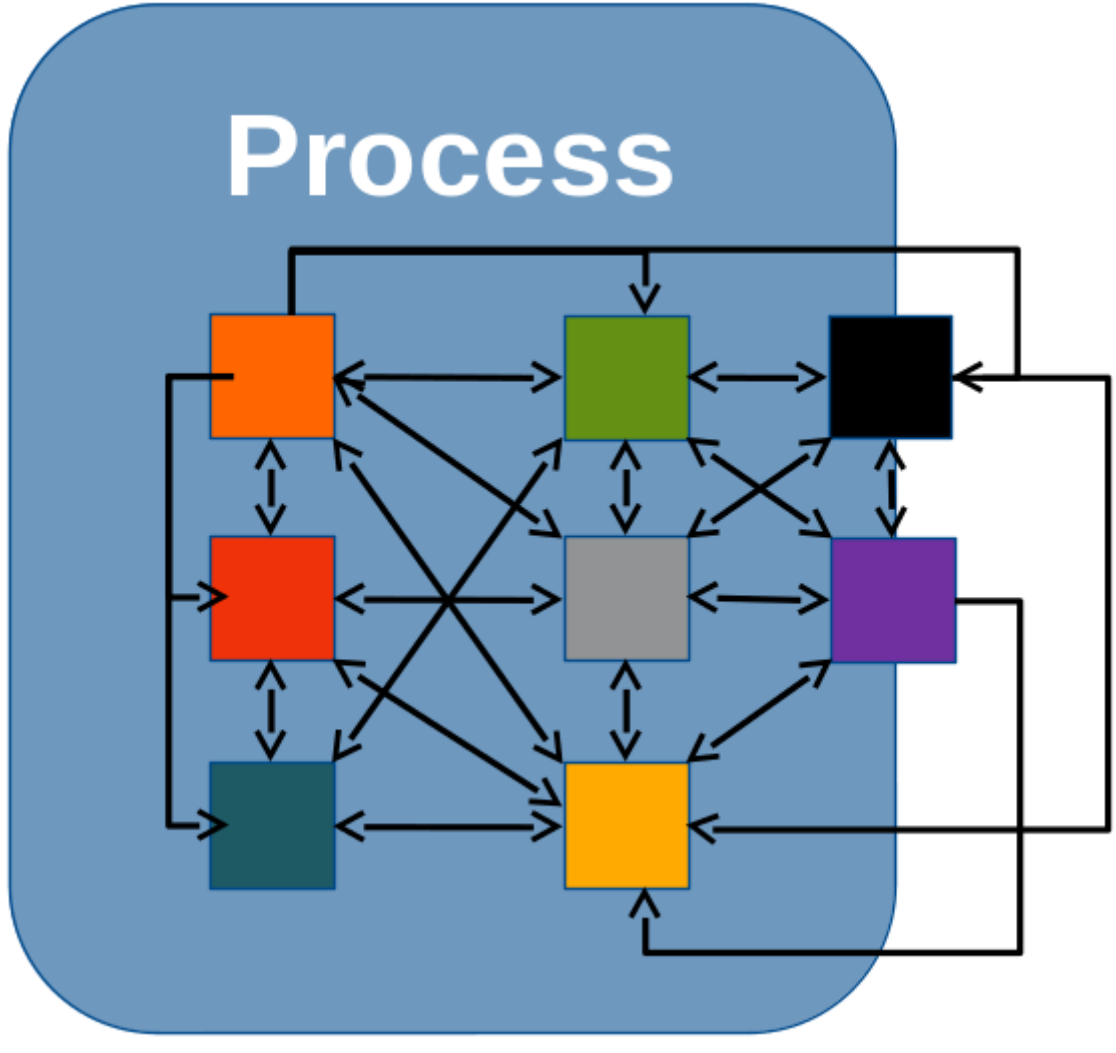
http://isa-principles.org

Video: Beyond Microservices (W-JAX 2017, Eberhard Wolff)

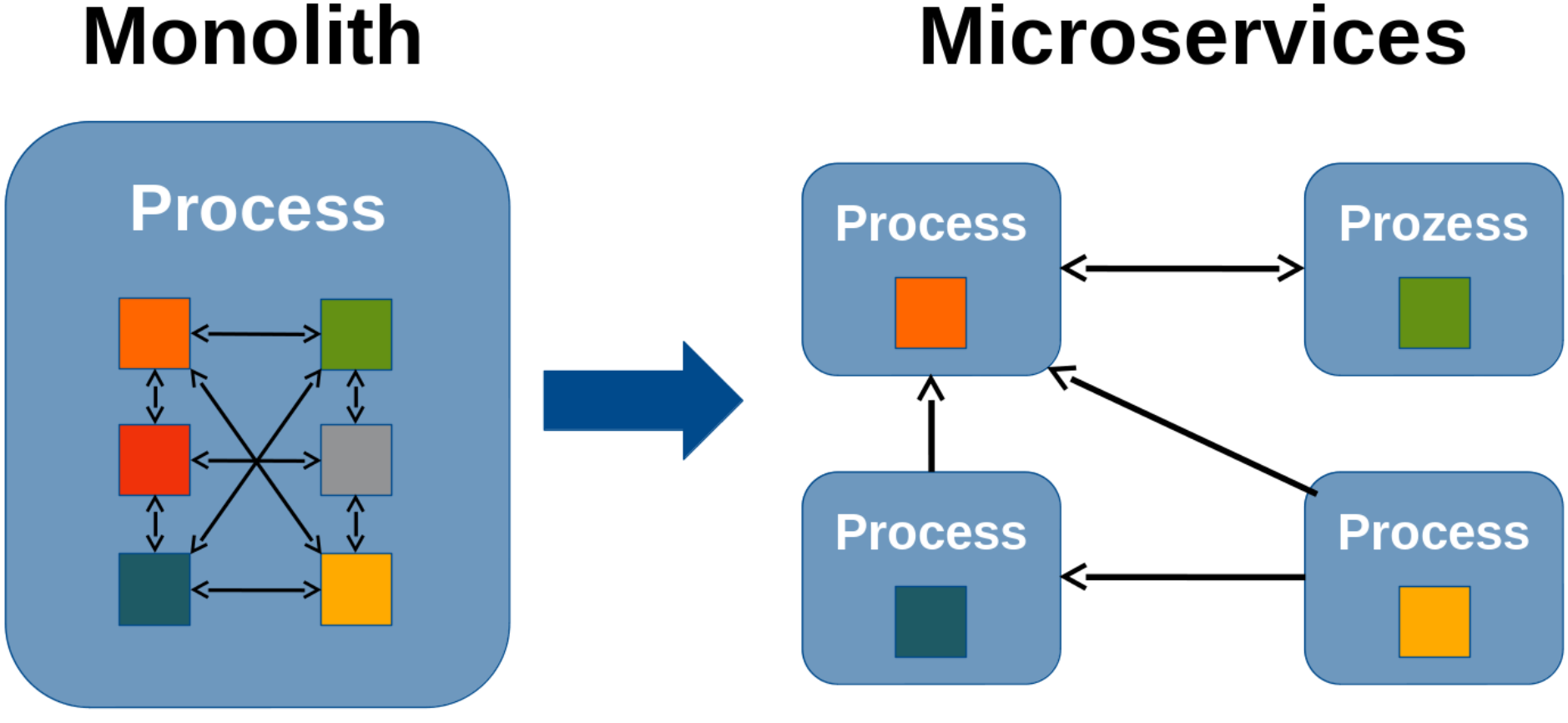
THE WELL DEFINED MONOLITH



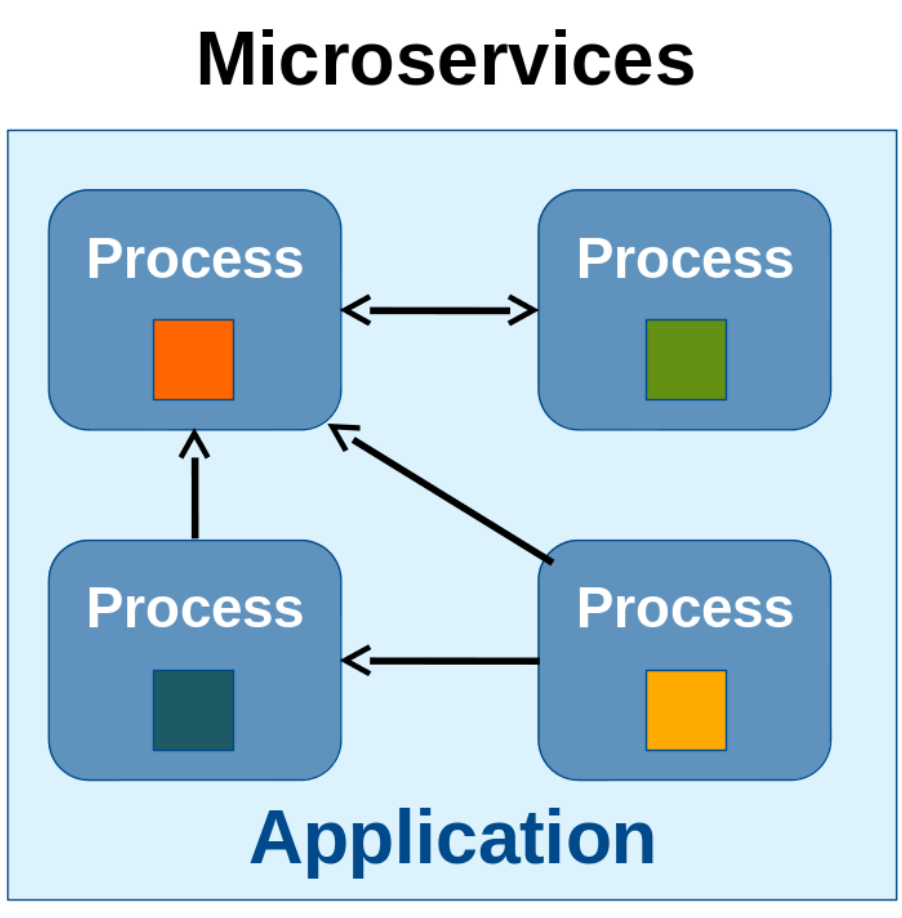
“BIG BALL OF MUD” MONOLITH



FROM MONOLITH TO MICROSERVICES



AUTONOMIC MICROSERVICES




Autonomic Services

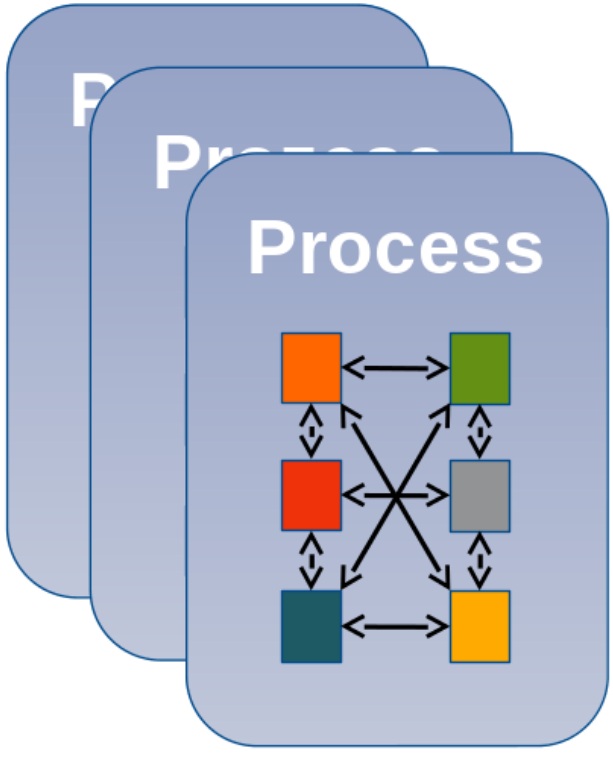

Light-weight Communication

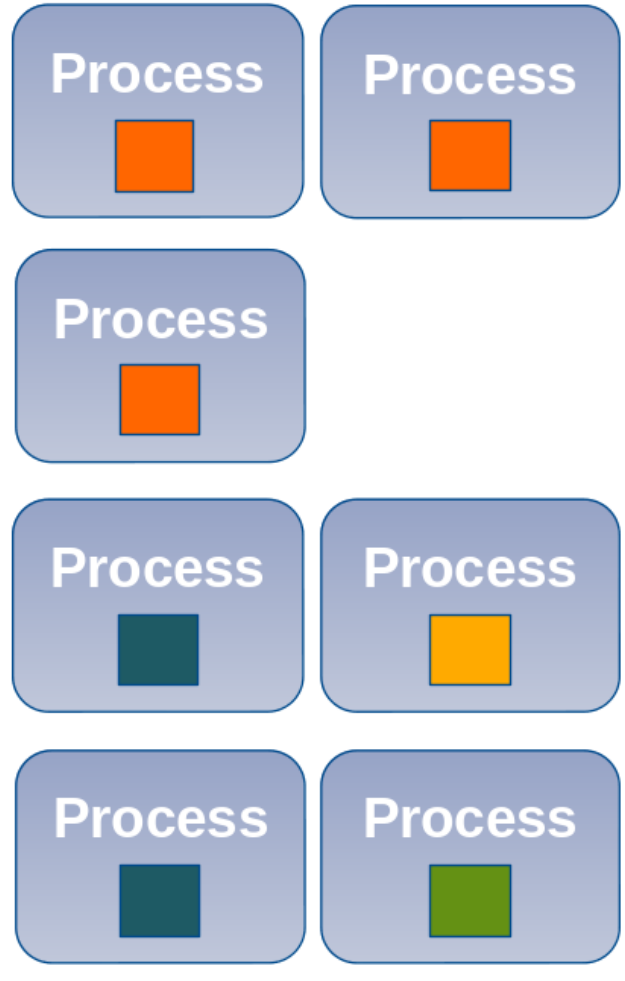

Java


Python

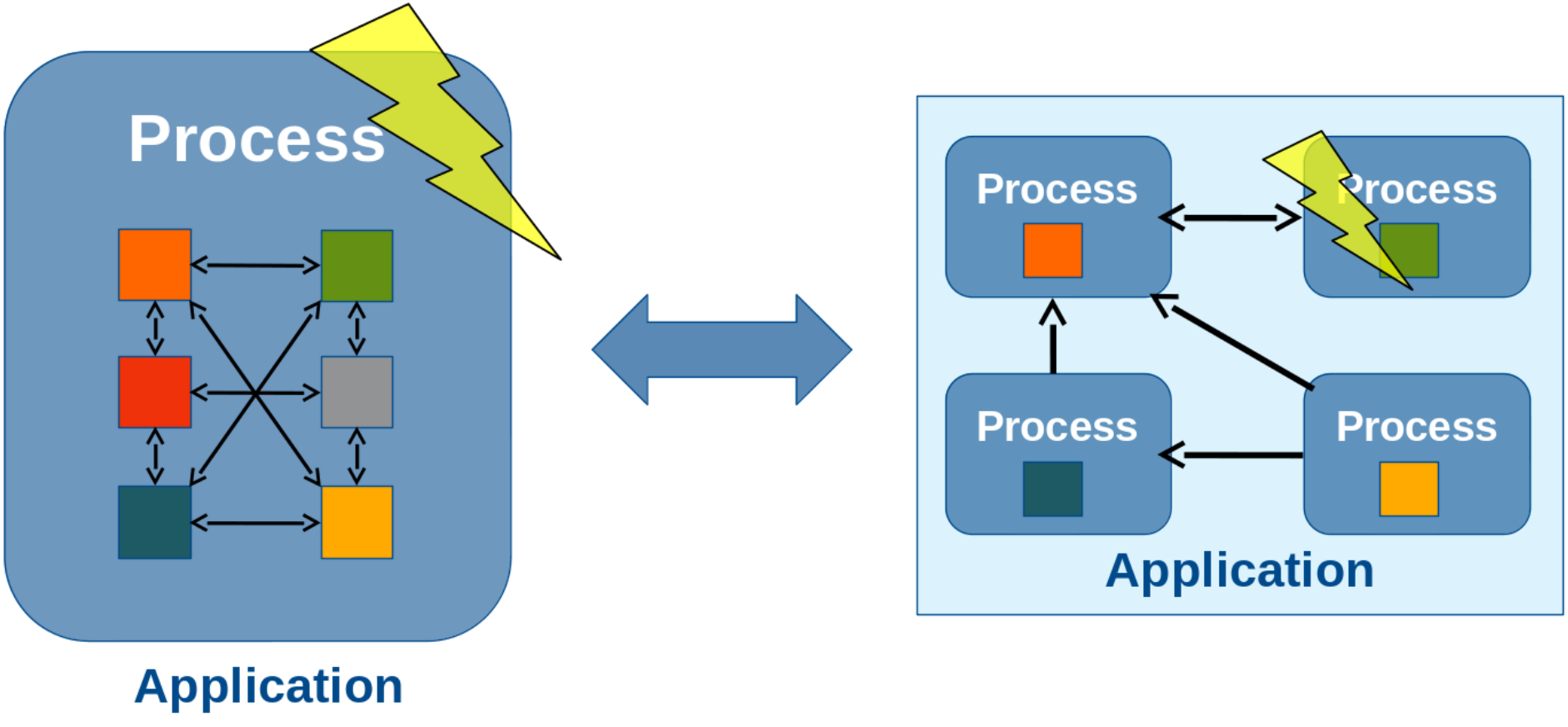

C# (.Net)

Scaling

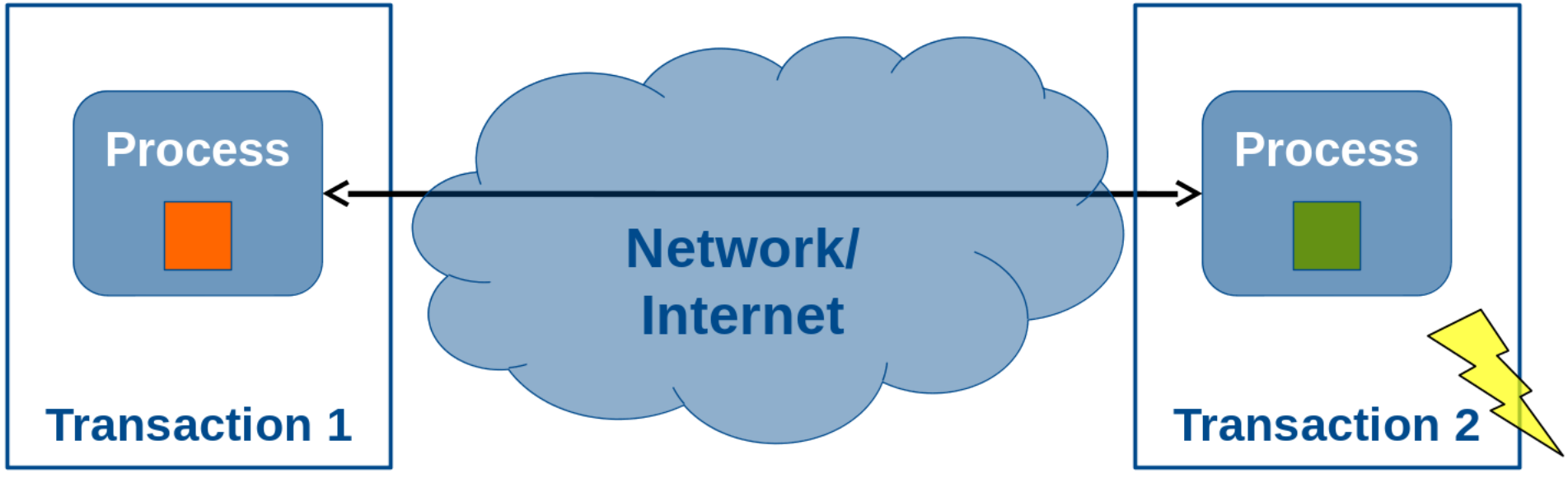

Process


Process
Process
Process
Process
Process
Process
Process
Process

Resilience

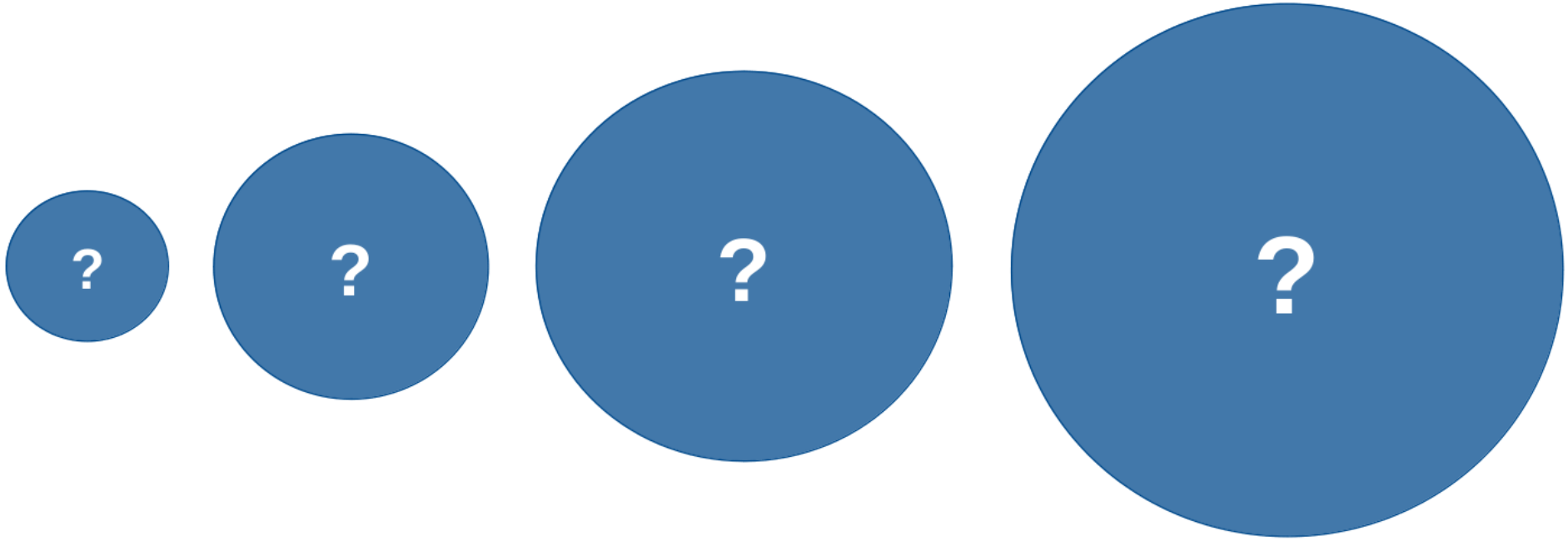

Process
Application
Process
Process
Process
Process
Application

Eventual Consistency



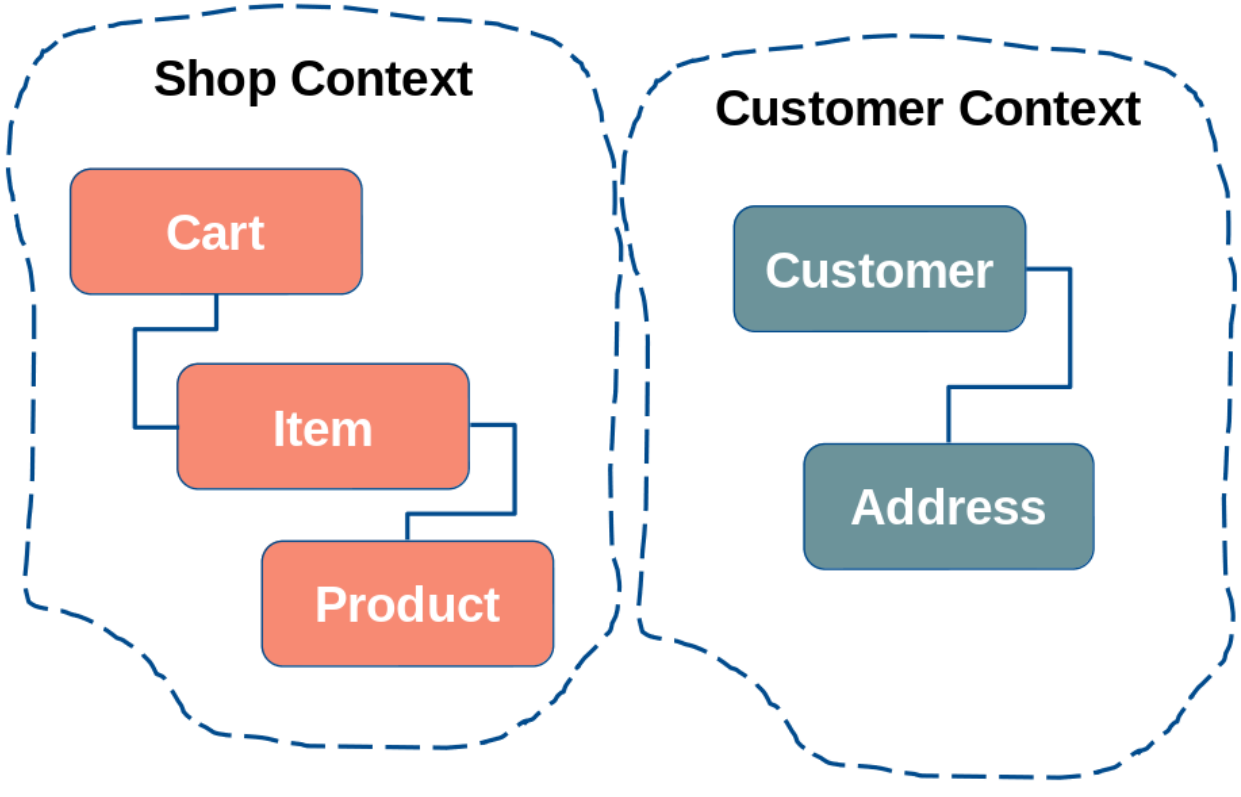
CI PIPELINES FOR MICROSERVICES

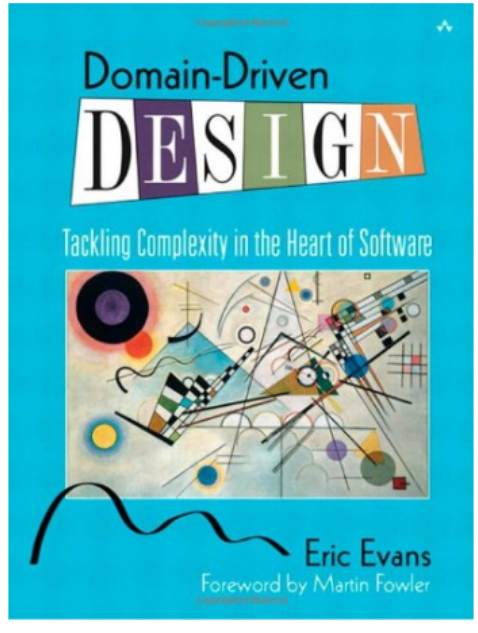
SIZE OF MICROSERVICES (I)



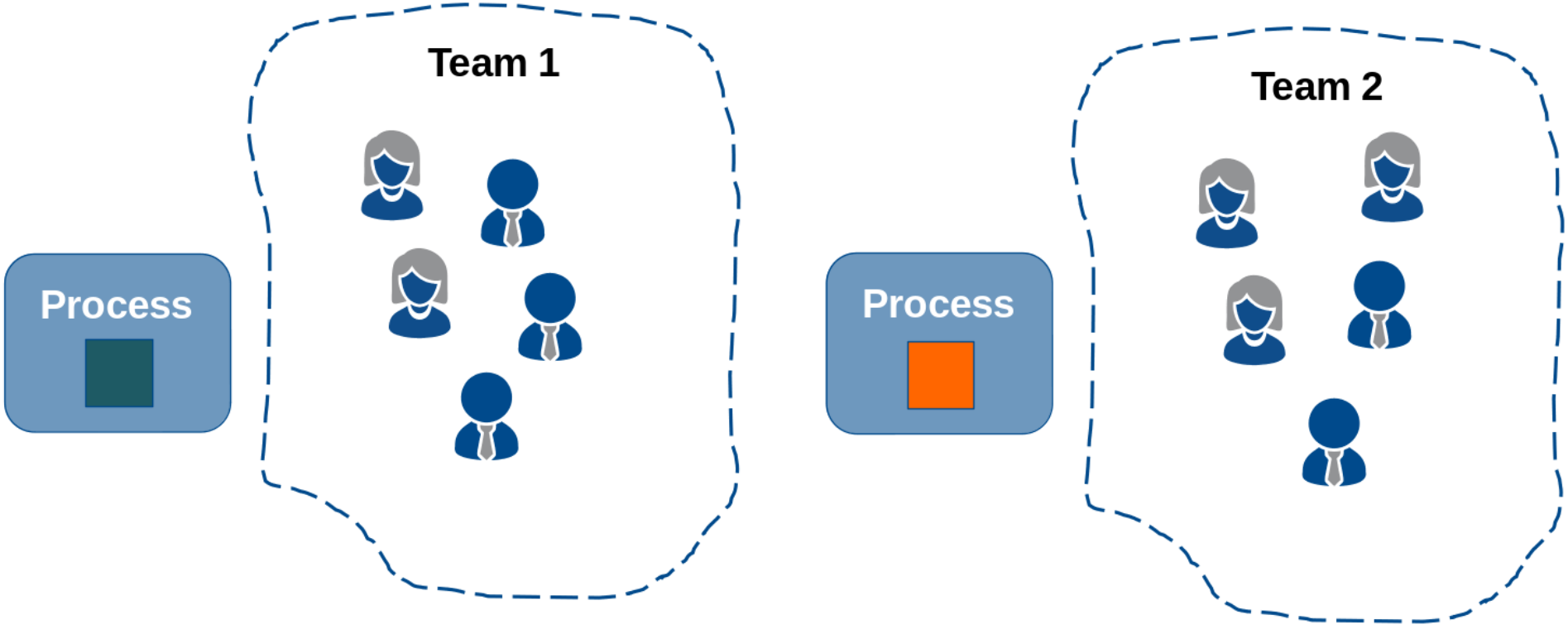
SIZE OF MICROSERVICES (II)

Bounded Contexts

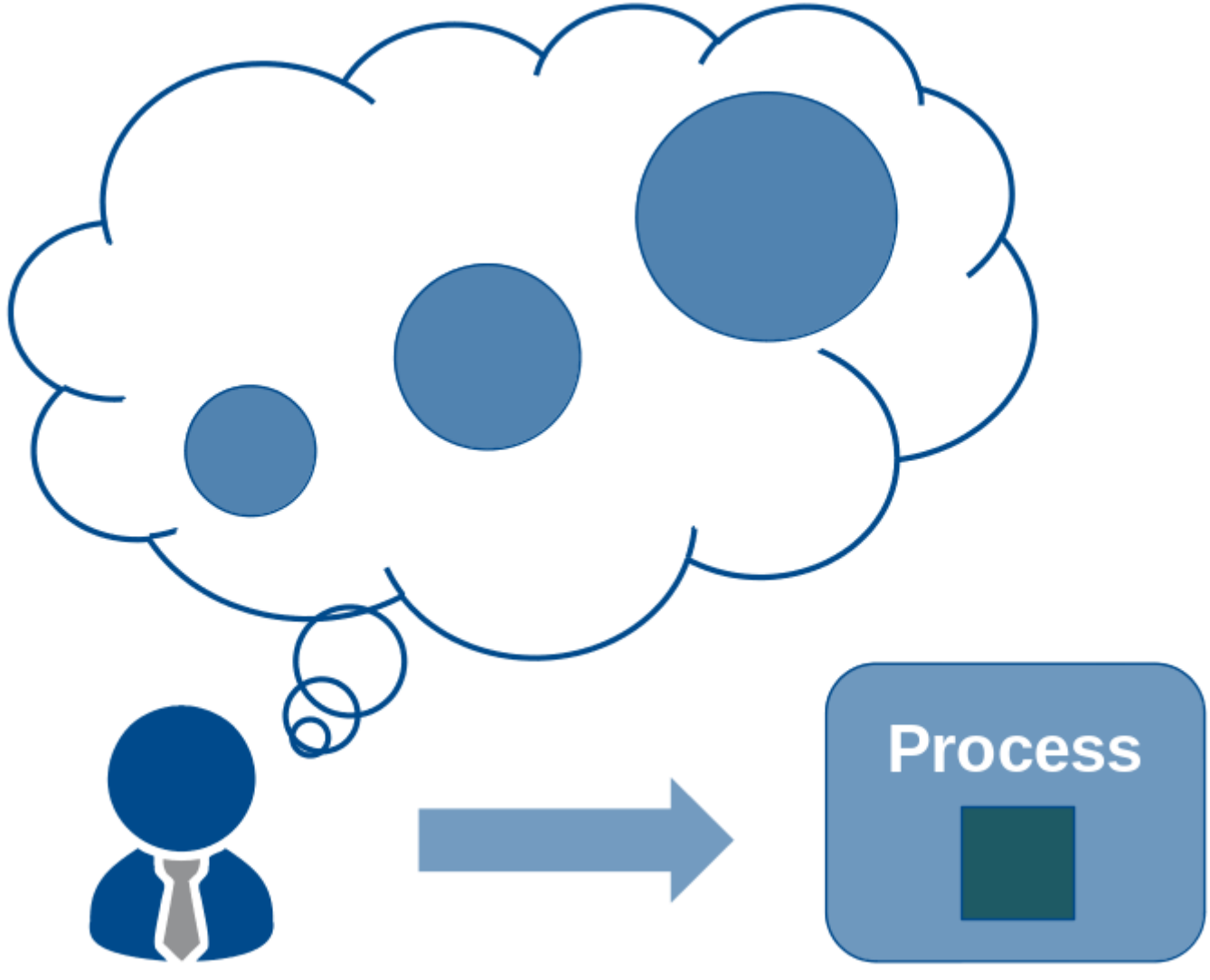




SIZE OF MICROSERVICES (III)



SIZE OF MICROSERVICES (IV)



SERVERLESS (FAAS)

“Applications where some amount of server-side logic is still written by the application developer but unlike traditional architectures is run in stateless compute containers that are event-triggered, ephemeral (may only last for one invocation), and fully managed by a 3rd party.”

Martin Fowler

SERVERLESS PROVIDERS

AWS Lambda

Azure Functions

Google Cloud Functions

CLOUD NATIVE (I)

“Cloud-native is an approach to building and running applications that fully exploits the advantages of the cloud computing delivery model.”

pivotal.io

CLOUD NATIVE (II)

THE TWELVE FACTORY APP

https://12factor.net

CLOUD NATIVE (III)

ON PREMISE

INFRASTRUCTURE AS A SERVICE (IAAS)

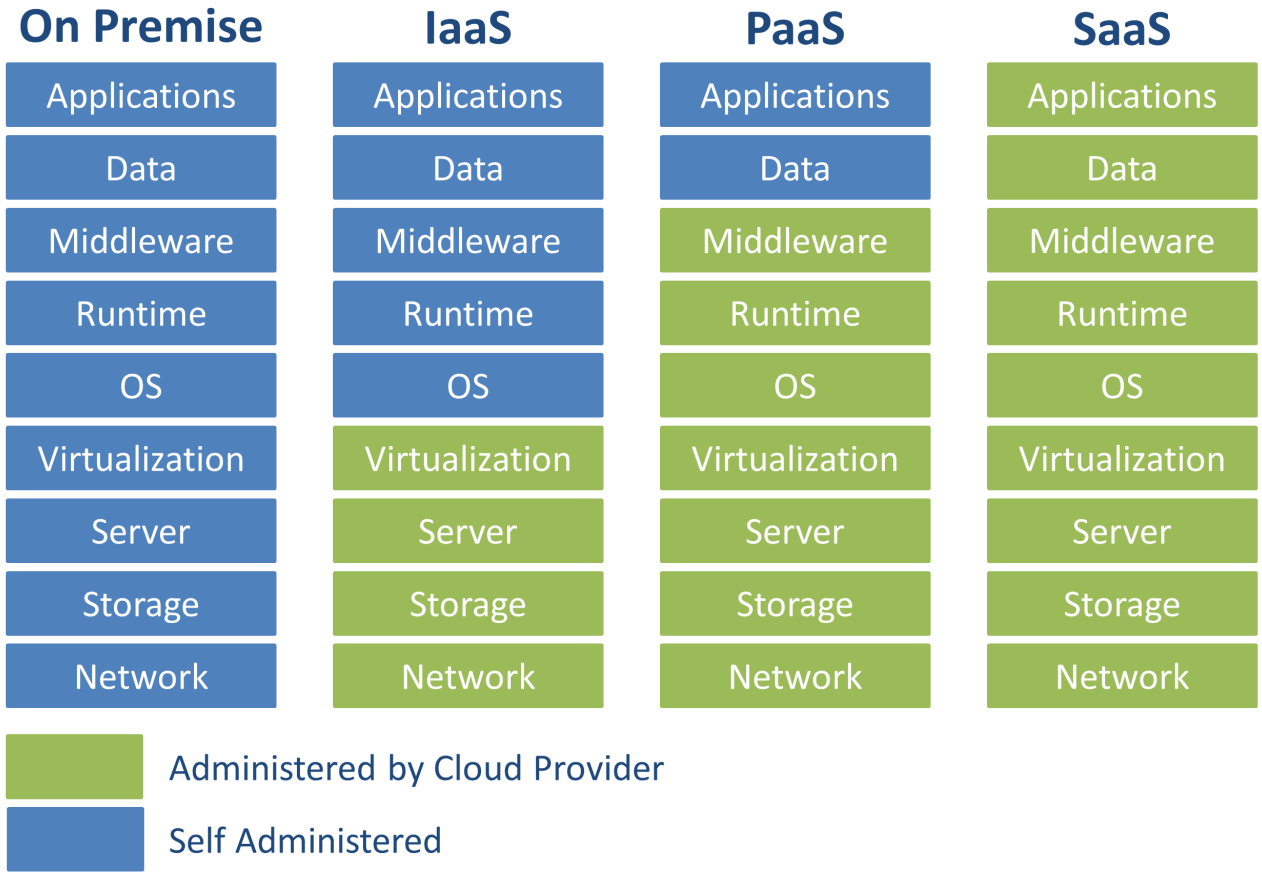
PLATFORM AS A SERVICE (PAAS)

SOFTWARE AS A SERVICE (SAAS)

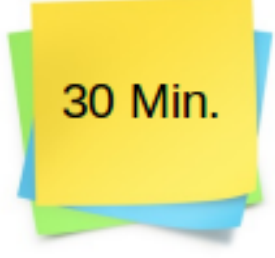
FUNCTION AS A SERVICE (FAAS)

CLOUD NATIVE (IV)

Cloud Service Models



WHICH ARCHITECTURAL PATTERNS DO YOU USE?



DESIGN PATTERNS & PRINCIPLES

SYMPTOMS OF ROTTING DESIGN

RIGIDITY

FRAGILITY

IMMOBILITY

VISCOSITY

RIGIDITY

Every change causes a cascade of subsequent changes in dependent modules.

FRAGILITY

Fragility is the tendency of the software to break in many places every time it is changed. Often the breakage occurs in areas that have no conceptual relationship with the area that was changed.

IMMOBILITY

Inability to reuse software from other projects or from parts of the same project.

VISCOSITY

When faced with a change, engineers usually find more than one way to make the change. Some of the ways preserve the design, others do not (i.e. they are hacks.)

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

“A general reusable solution to a commonly occurring problem within a given context in software design.”

Wikipedia

DESIGN PATTERNS CATALOGUE

CREATIONAL PATTERNS

STRUCTURAL PATTERNS

BEHAVIORAL PATTERNS

CONCURRENCY PATTERNS

CREATIONAL PATTERNS

FACTORY

BUILDER

SINGLETON / PROTOTYPE

DEPENDENCY INJECTION

...

STRUCTURAL PATTERNS

ADAPTER

BRIDGE

DECORATOR

PROXY

...

BEHAVIORAL PATTERNS

COMMAND

OBSERVER (PUBLISH/SUBSCRIBE)

ITERATOR

...

CONCURRENCY PATTERNS

ITERATOR

LOCK

REACTOR

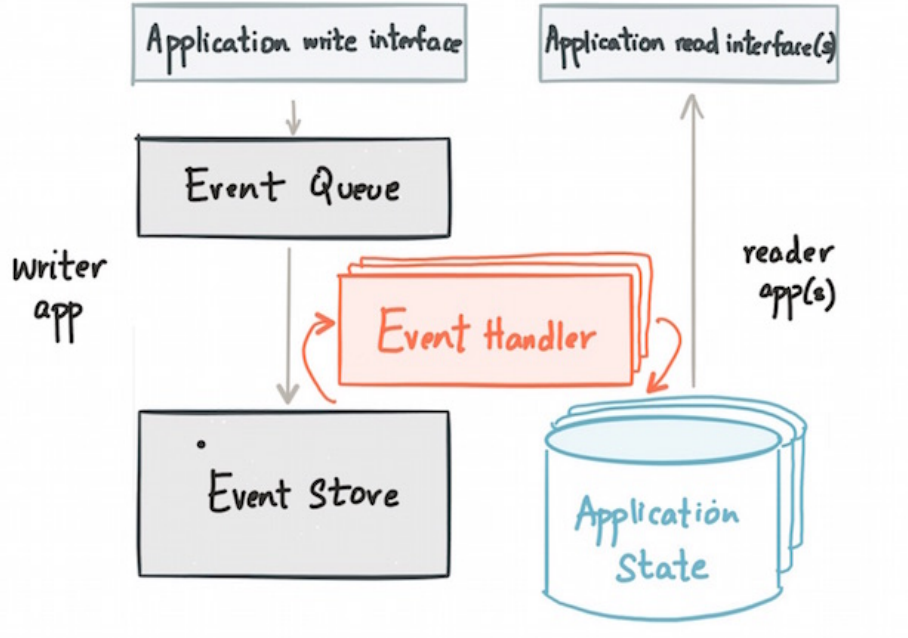
THREAD POOL

...

OTHER PATTERNS

EVENT SOURCING &

COMMAND QUERY RESPONSIBILITY SEGREGATION (CQRS)



https://www.confluent.io/blog/event-sourcing-cqrs-stream-processing-apache-kafka-whats-connection

REACTIVE SYSTEMS

“Systems that are Responsive, Resilient, Elastic and Message Driven. We call these Reactive Systems.”

The Reactive Manifesto

REACTIVE SYSTEMS IMPLEMENTATIONS

akka

Reactor

RxJava

DESIGN PRINCIPLES

SOLID

Single Responsibility Principle

Open/Closed Principle

Liskov's Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

SINGLE RESPONSIBILITY PRINCIPLE

A class should have only one reason to change

No “God Classes”

OPEN-CLOSED PRINCIPLE

You should be able to extend a classes behavior, without modifying it

LISKOV SUBSTITUTION PRINCIPLE

Derived classes must be substitutable for their base classes

INTERFACE SEGREGATION PRINCIPLE

Make fine grained interfaces that are client specific

DEPENDENCY INVERSION PRINCIPLE

Depend on abstractions, not on concretions

KEEP IT SIMPLE, STUPID (KISS)

“Simplicity should be a key goal in design and unnecessary complexity should be avoided.”

Wikipedia

YOU AREN'T GONNA NEED IT (YAGNI)

“A mantra from Extreme Programming that's often used generally in agile software teams. It's a statement that some capability we presume our software needs in the future should not be built now.”

Martin Fowler

ARCHITECTURE LAB

TARGET?

JAVA EE

EE4J

SPRING

JAVA EE

Java EE 8 is final...

...but NO commercial AppServer available!

...and still NO standards for modern security, NoSQL, Cloud

ECLIPSE ENTERPRISE FOR JAVA (EE4J)

“A nimble, flexible and open process for evolving EE4J standard APIs, implementations of those APIs, and technology compatibility kits.”

Baseline is Java EE 8

We will see what the future brings...

https://projects.eclipse.org/projects/ee4j

SPRING PLATFORM

Proven Cloud Native Java (Kotlin) Platform

Great support for NoSQL, BigData, Reactive

Modern Security (OAuth2, OpenID Connect)

Compatible with Java EE 7 & 8

=> WE WILL USE SPRING

SPRING PLATFORM

https://spring.io

SPRING PRINCIPLES

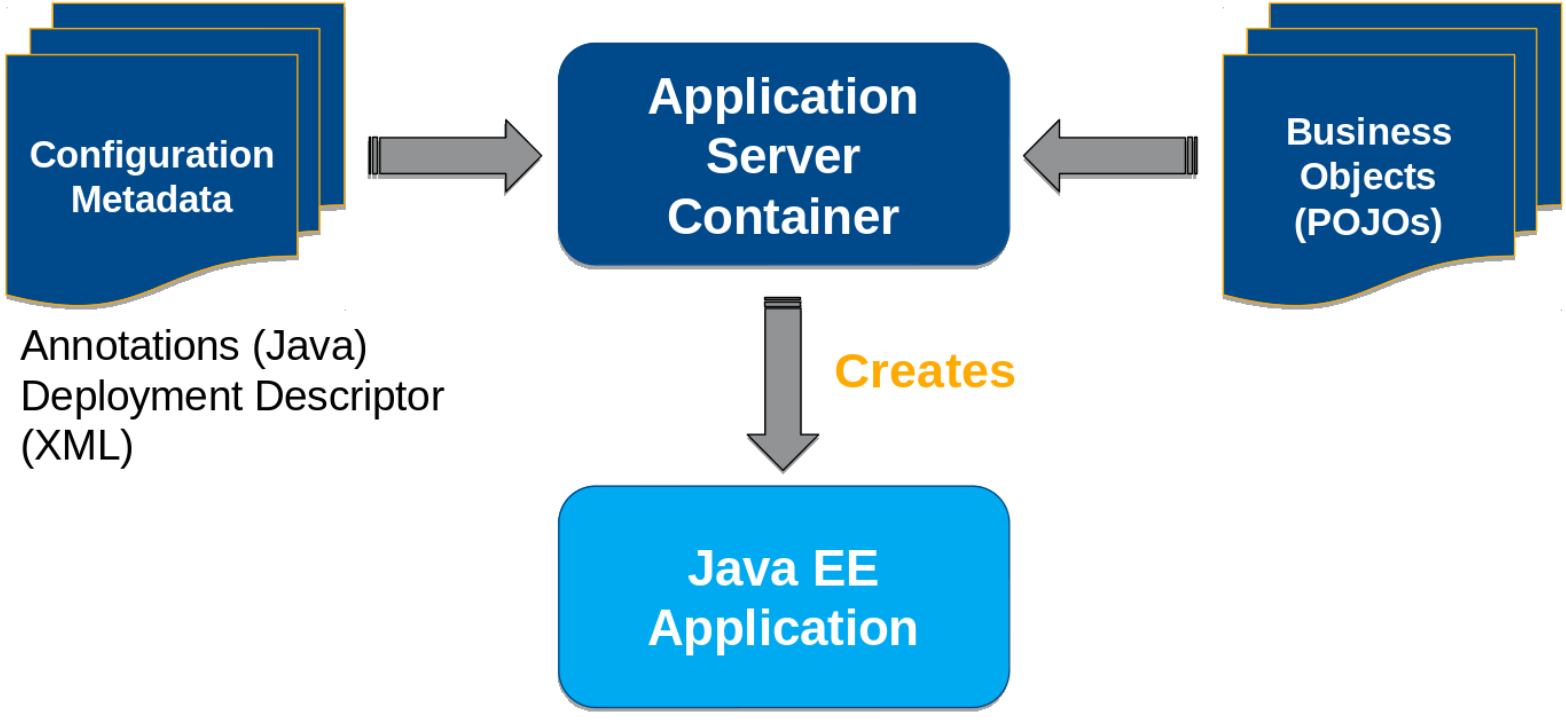
LIGHTWEIGHT DEVELOPMENT USING POJOS

DEPENDENCY INJECTION (INVERSION OF CONTROL)

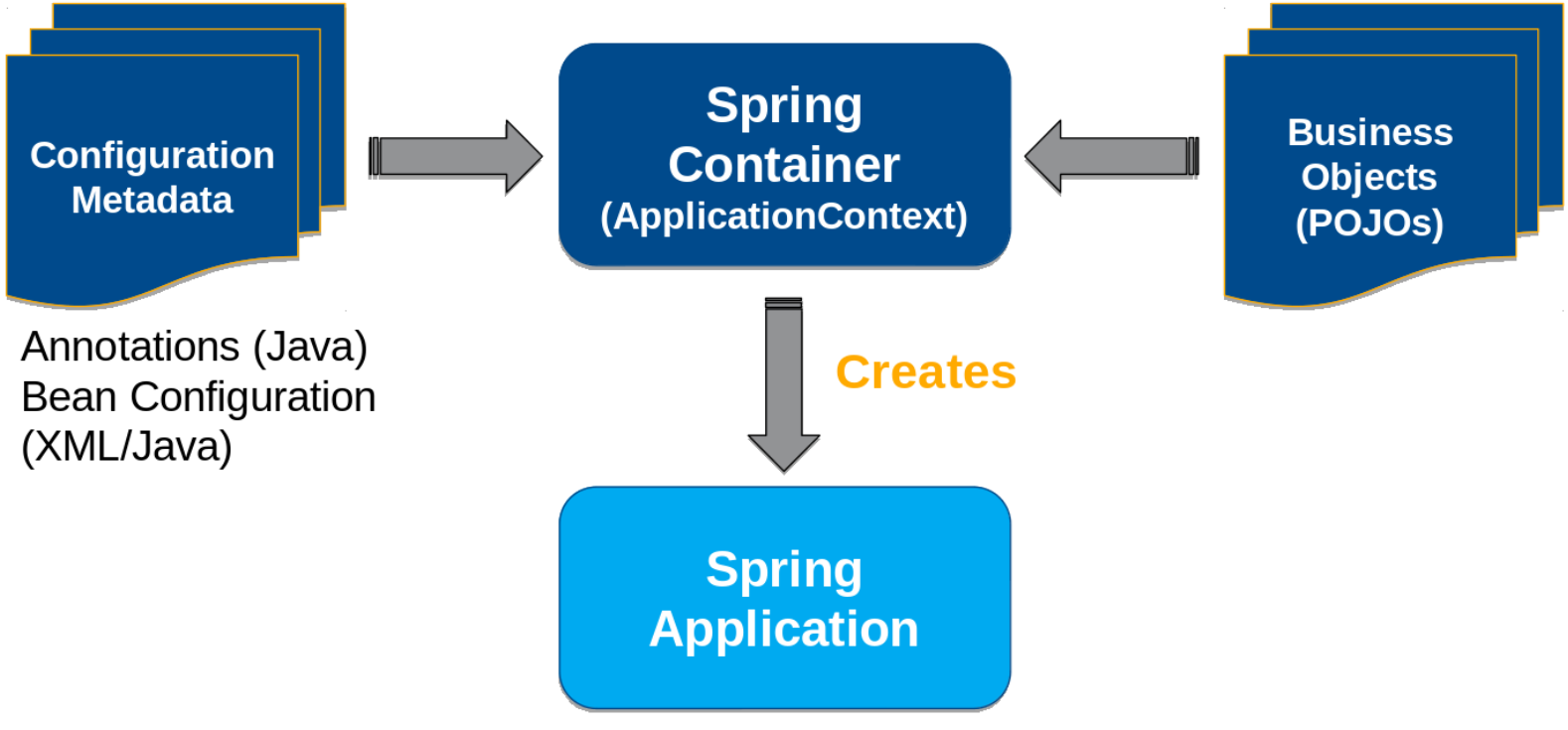
REDUCTION OF BOILERPLATE CODE

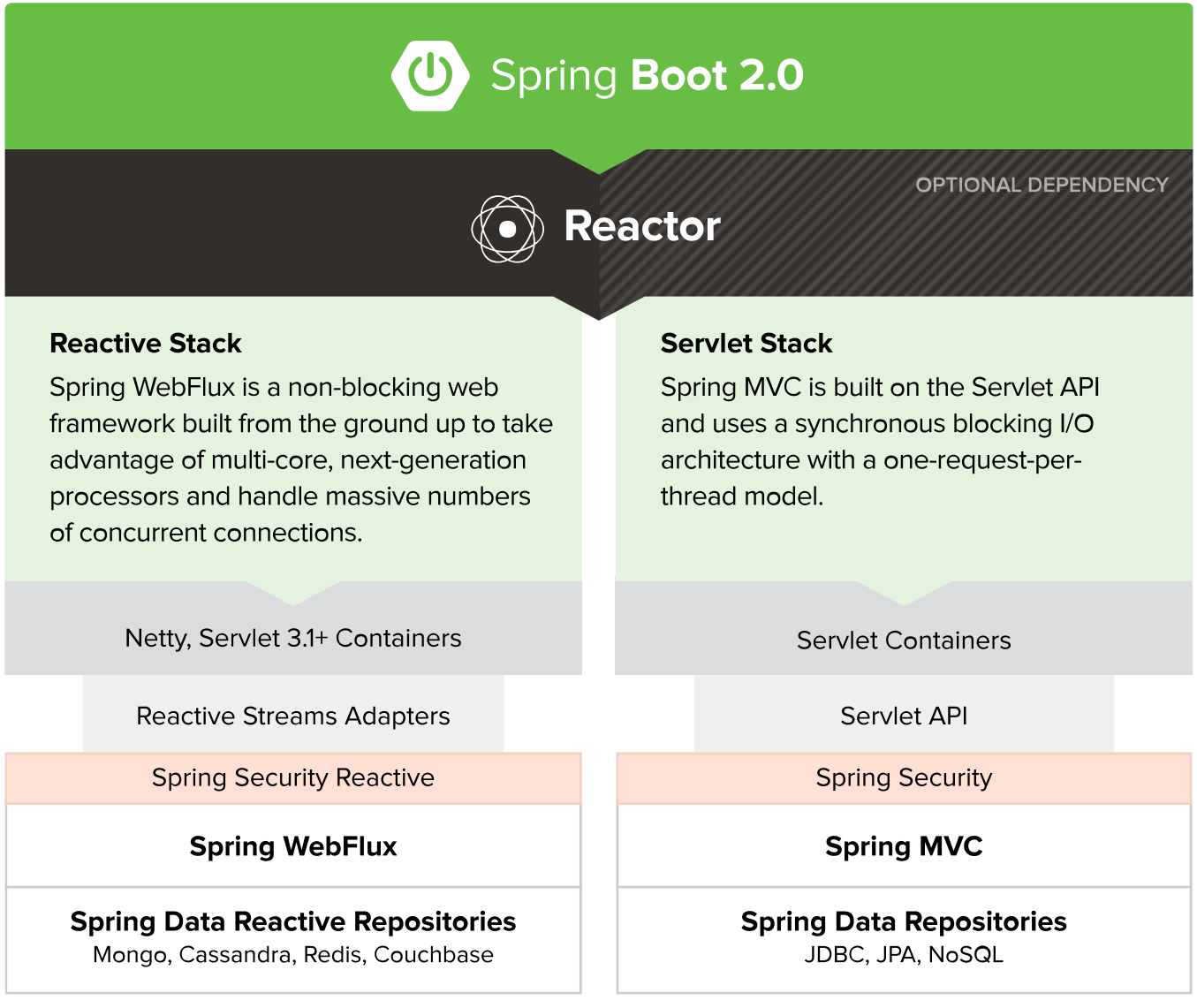
SIMPLIFY DEVELOPMENT

Java EE Application Server Container

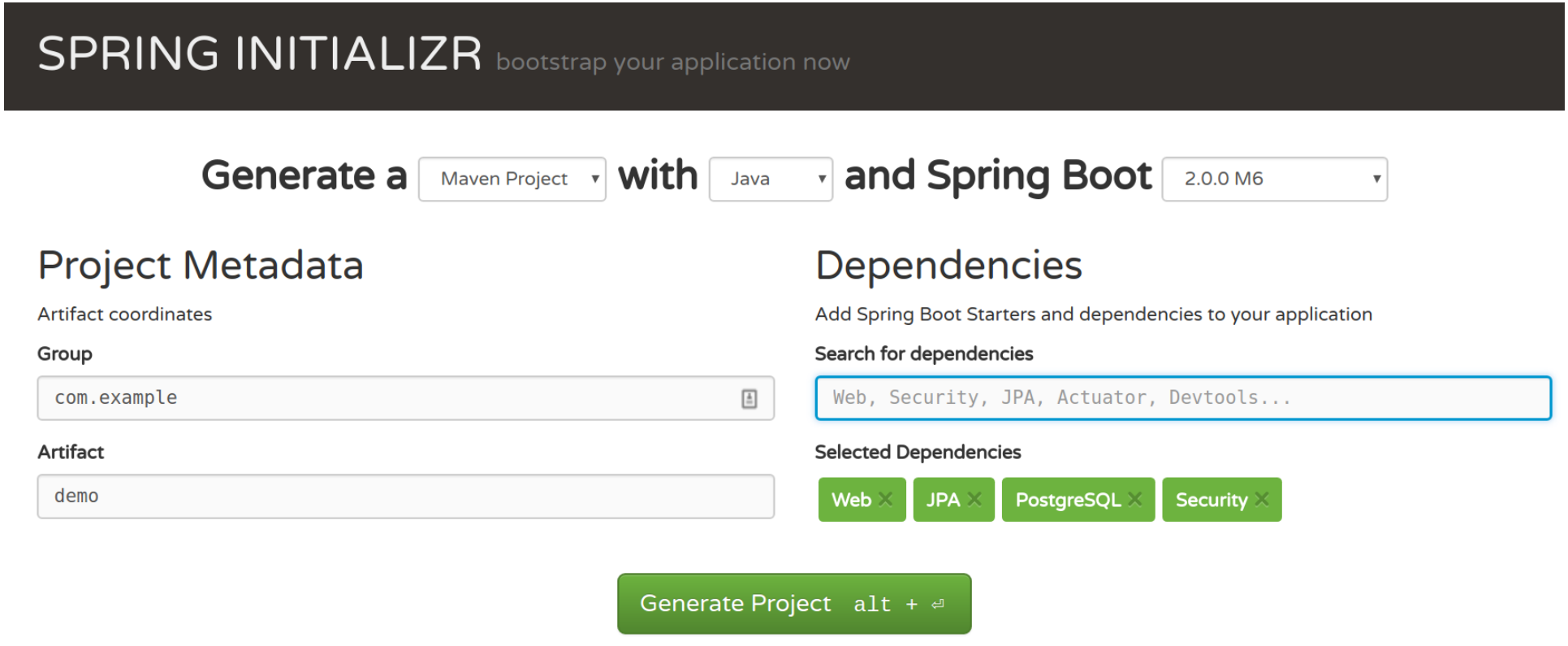


Spring Core Container





https://spring.io



https://start.spring.io

```
<parent>
    <groupid>org.springframework.boot</groupid>
    <artifactid>spring-boot-starter-parent</artifactid>
    <version>2.0.0.M6</version>
</parent>
<dependencies>
    <dependency>
        <groupid>org.springframework.boot</groupid>
        <artifactid>spring-boot-starter-web</artifactid>
    </dependency>
    ...
</dependencies>
```


1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1

Q&A

http://www.novatec-gmbh.de
http://blog.novatec-gmbh.de

andreas.falk@novatec-gmbh.de

@NT_AQE, @andifalk

REFERENCES

- Presentation and Workshop: https://github.com/nt-ca-aqe/ws-2017-11-10-architecture
- Spring Cloud (https://cloud.spring.io)
- Spring Boot (https://projects.spring.io/spring-boot)