SOFTCON 2023

PHILIPPINE SOFTWARE INDUSTRY CONFERENCE

Modular Monolithic in Practice: Explore Implementations in Java

TRISTAN MAHINAY

DECODING AI

TRISTAN MAHINAY

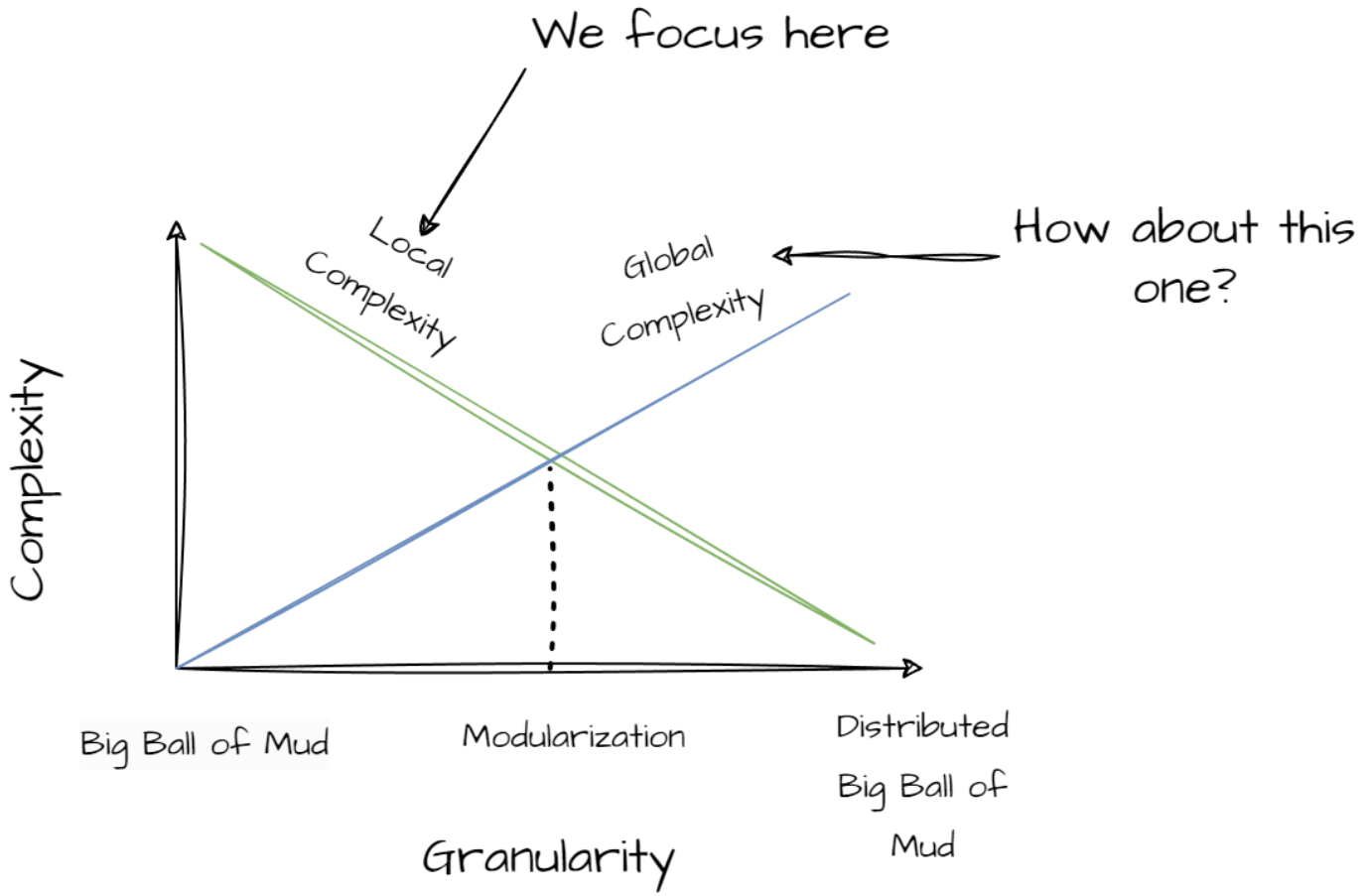
Senior Technical Specialist, JUG Leader, Java Capability Leader, Open Source Contributor

rjtmahinay

me@rjtmahinay.com

INTRODUCTION

COMPLEXITY

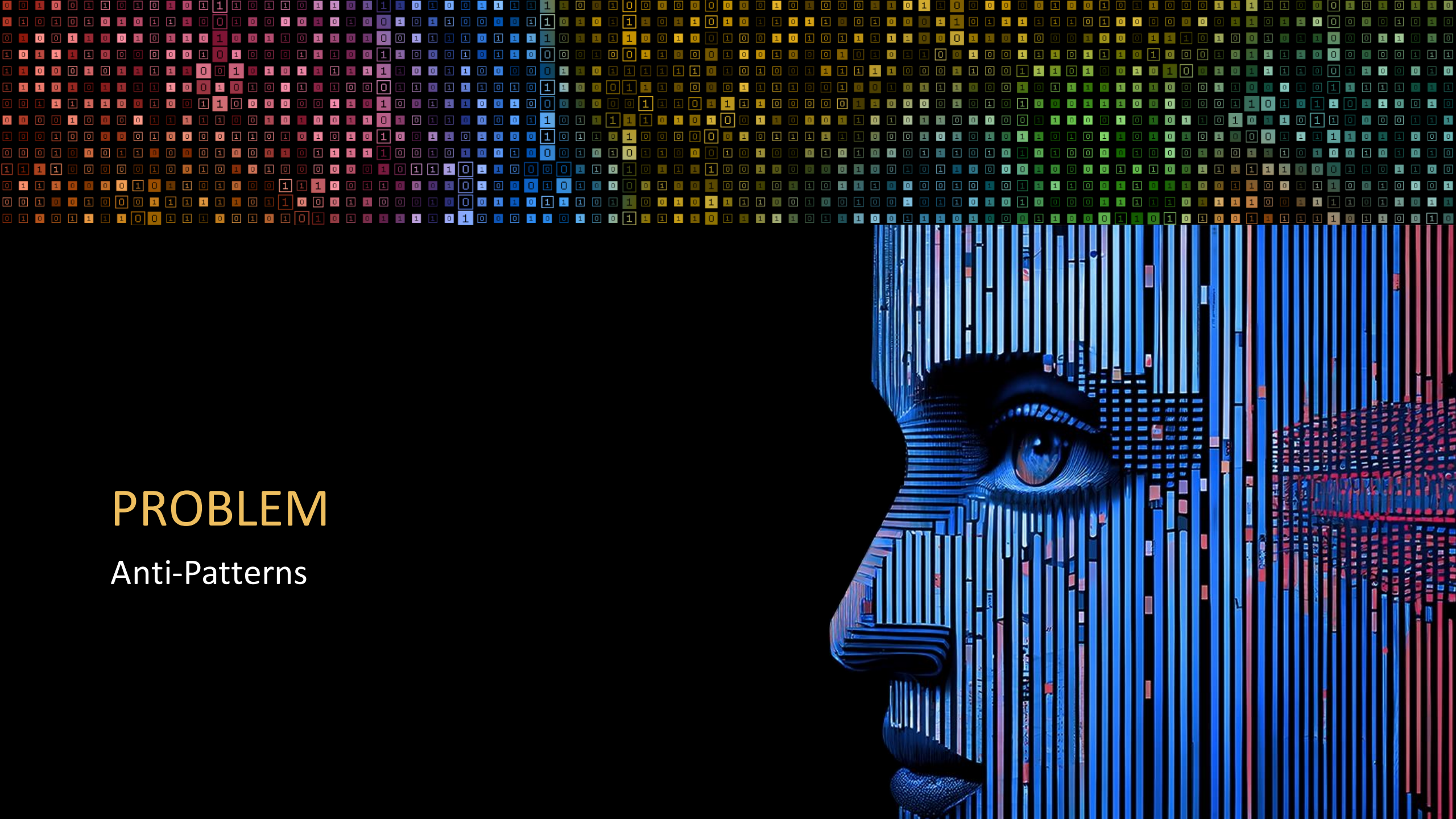

We focus here
Local Complexity
Global Complexity
How about this one?
Complexity
Big Ball of Mud
Modularization
Distributed Big Ball of Mud
Granularity

Over-optimization of Local Complexity → Big Ball of Mud (BBoM)

Over-optimization of Global Complexity → Distributed Big Ball of Mud (DBBoM)

PROBLEM

Anti-Patterns

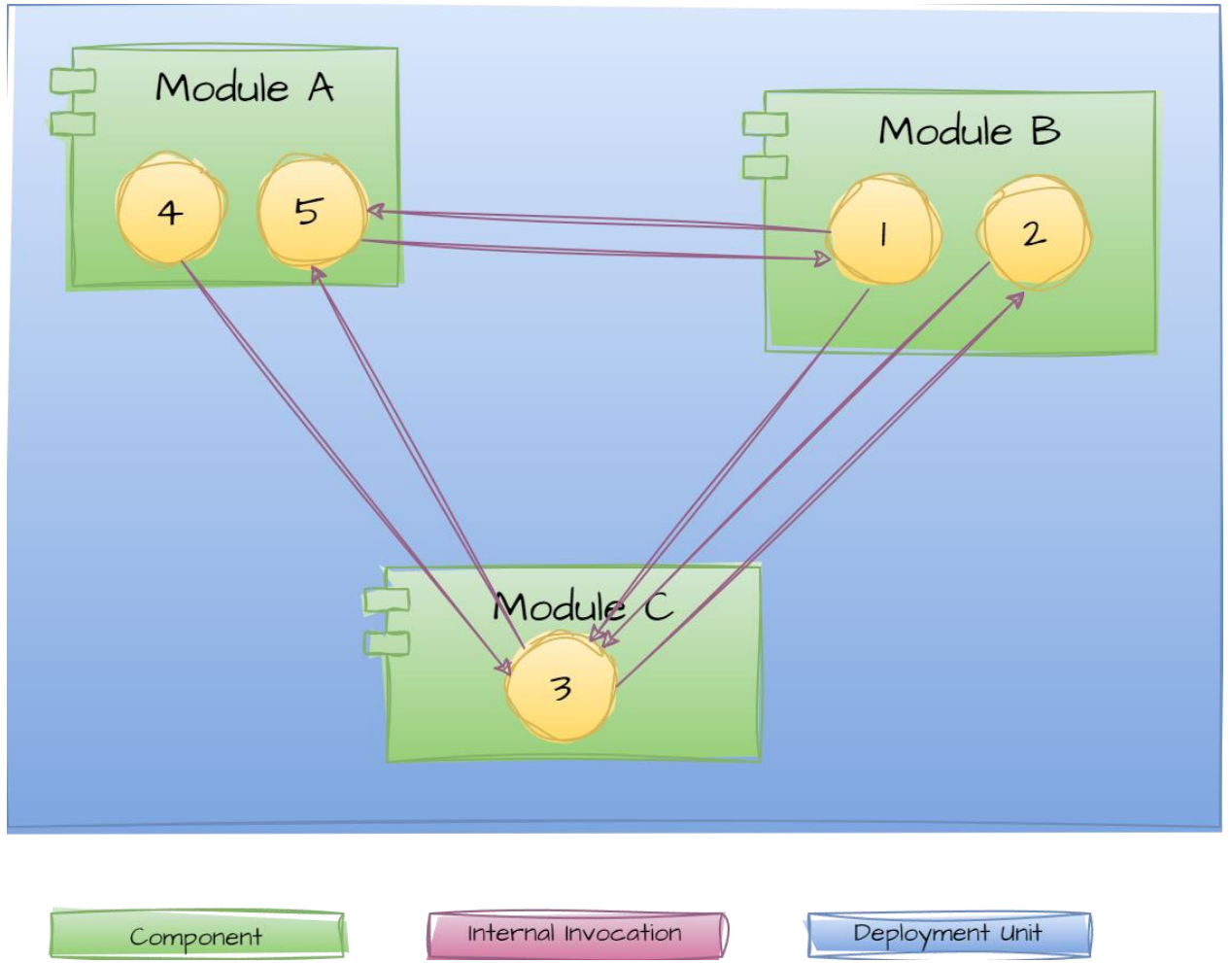


ANTI-PATTERNS

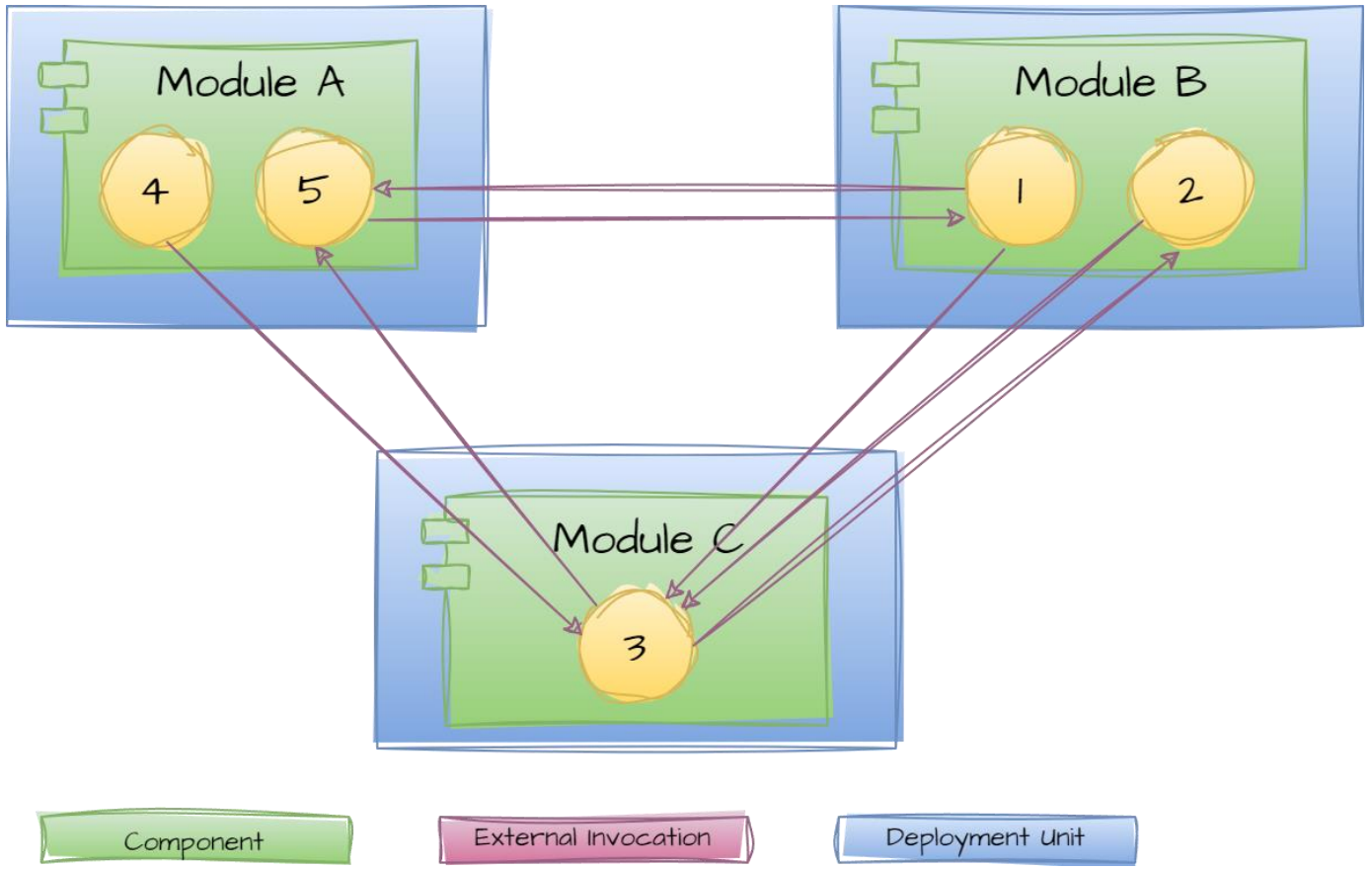
- Failed Monolithic
- High Coupling, Low Cohesion (internal calls)
- High Cost of Change (Connascence)

- Failed Microservice aka *Distributed Monolithic*
- High Coupling, Low Cohesion (remote calls)
- High Cost of Change (Connascence)

BIG BALL OF MUD



DISTRIBUTED BIG BALL OF MUD



Felix Müller
@fmueller_bln

Follow

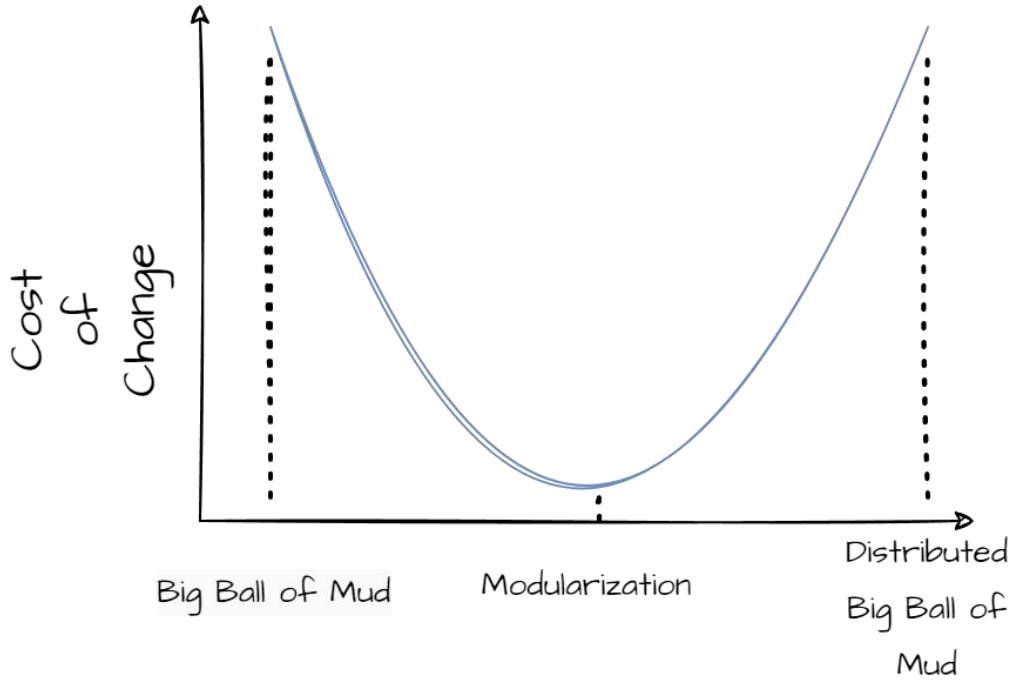
It is hard to build a well-modularised monolith. So, we decided to create many bad-modularised microservices.

11:01 PM · 16 Oct 21

41 Reposts **1** Quote **218** Likes **5** Bookmarks

COST OF CHANGE

CONNASCENCE – Components need to change to maintain overall correctness of the application. Components have Acyclic Dependencies.



What’re the cause of those two extreme cases?

 *Low Cohesion*

 *High Coupling*

 *High Connascence*

WHAT ARE WE TRYING TO SOLVE?

- ✓Functional Cohesion
- ✓A system that is fit in our current environment
- ✓Given the constraint we have, what are our priorities?
  - ✓*Architectural Characteristics*

ARCHITECTURAL THINKING == TRADE-OFFS

Never shoot for the best architecture, but rather the least worst architecture.

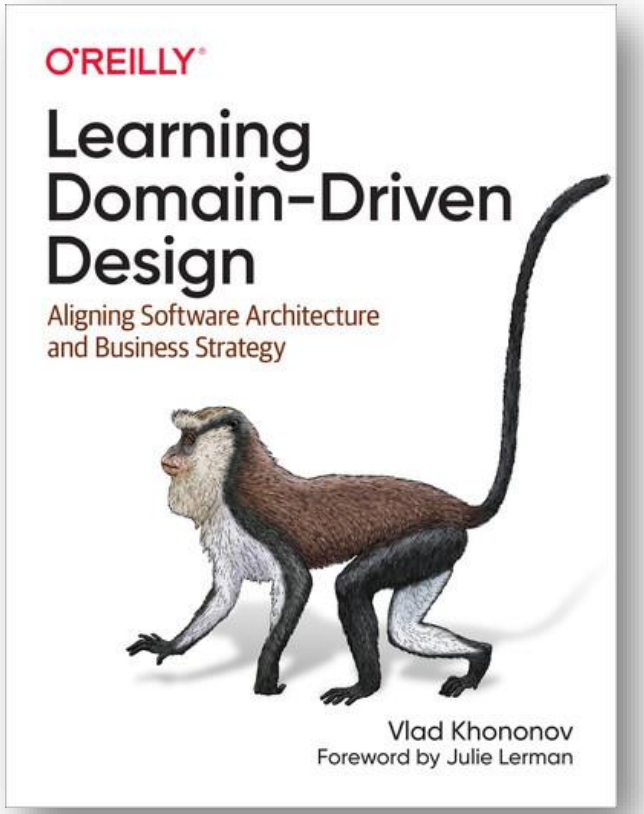
- Mark Richards & Neal Ford

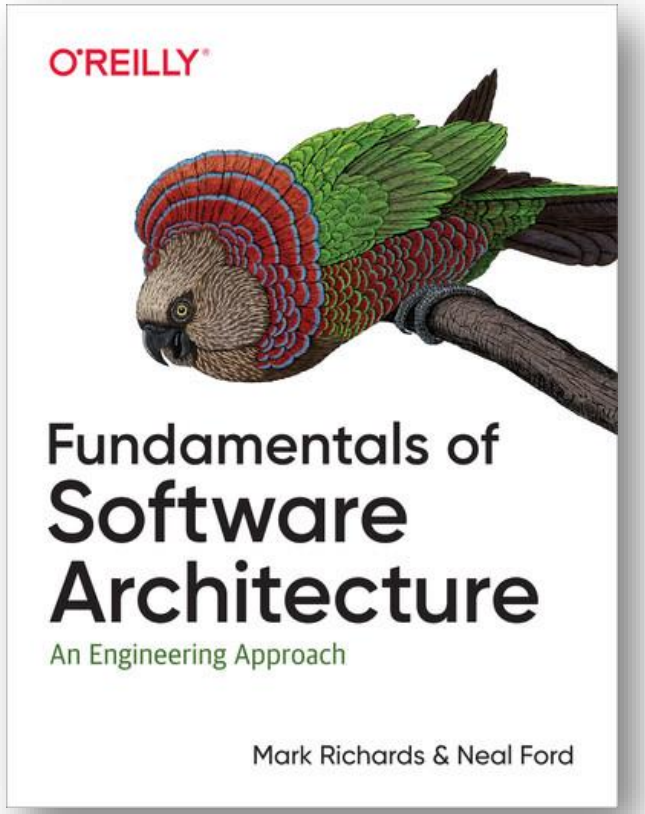
There's no right or wrong answer, only trade-offs.

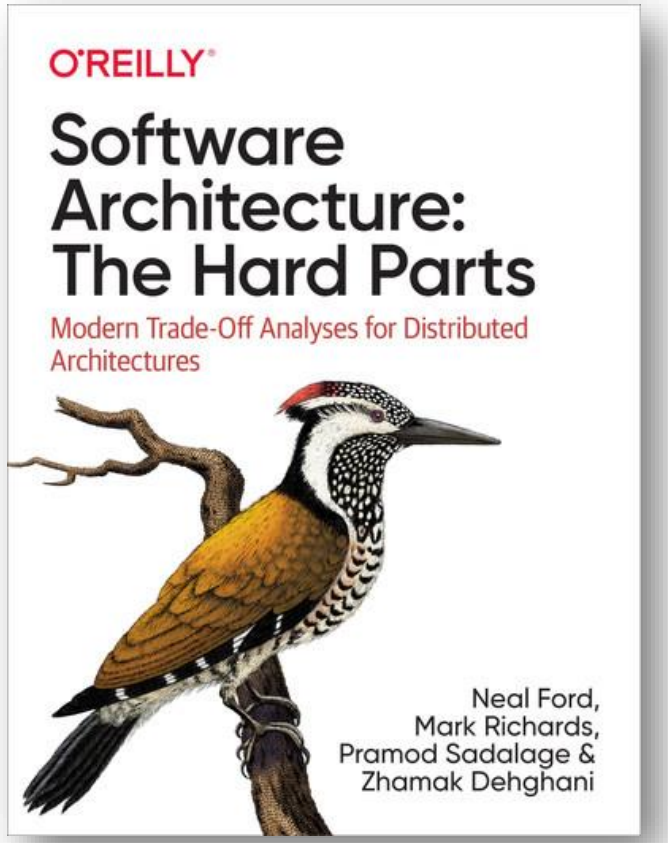
- Mark Richards & Neal Ford

There is no sense in talking about the solution before we agree on the problem, and no sense talking about the implementation steps before we agree on the solution.

- Efrat Goldratt-Ashlag


O'REILLY
Learning Domain-Driven Design
Aligning Software Architecture and Business Strategy
Vlad Khononov
Foreword by Julie Lerman

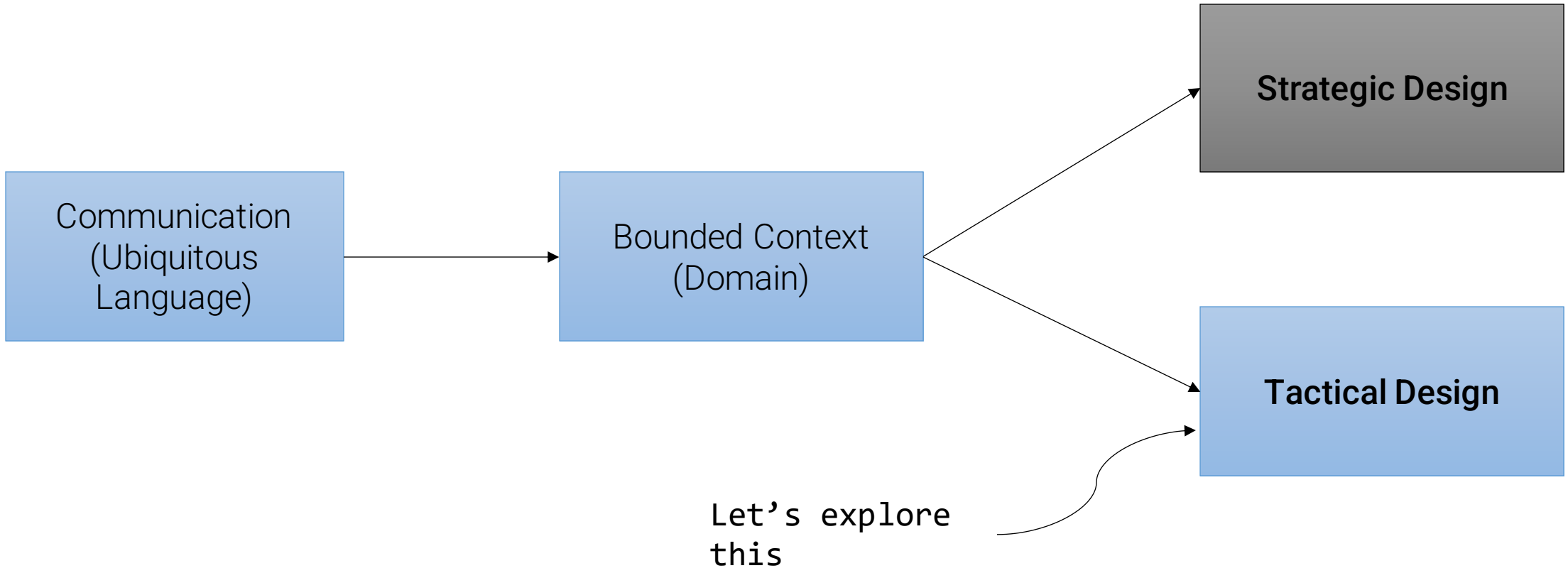

O'REILLY
Fundamentals of Software Architecture
An Engineering Approach
Mark Richards & Neal Ford


O'REILLY
Software Architecture: The Hard Parts
Modern Trade-Off Analyses for Distributed Architectures
Neal Ford,
Mark Richards,
Pramod Sadalage &
Zhamak Dehghani

SOLUTION

Apply a pragmatic approach

DOMAIN-DRIVEN DESIGN

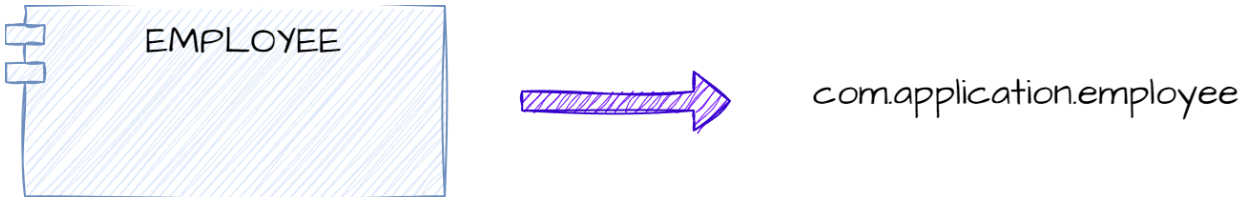


MODULARITY

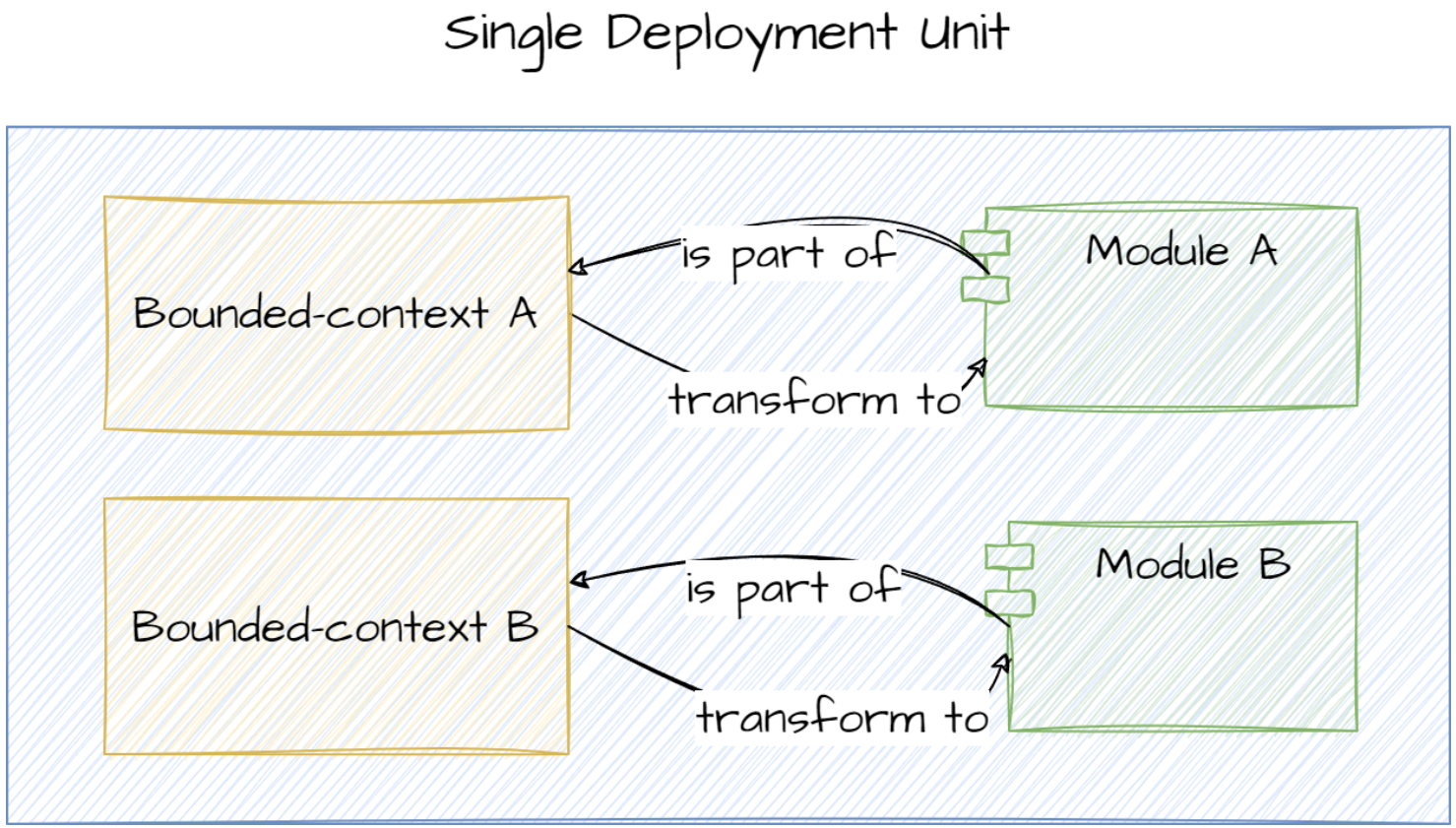
What is a modular application?

WHAT IS A MODULE?

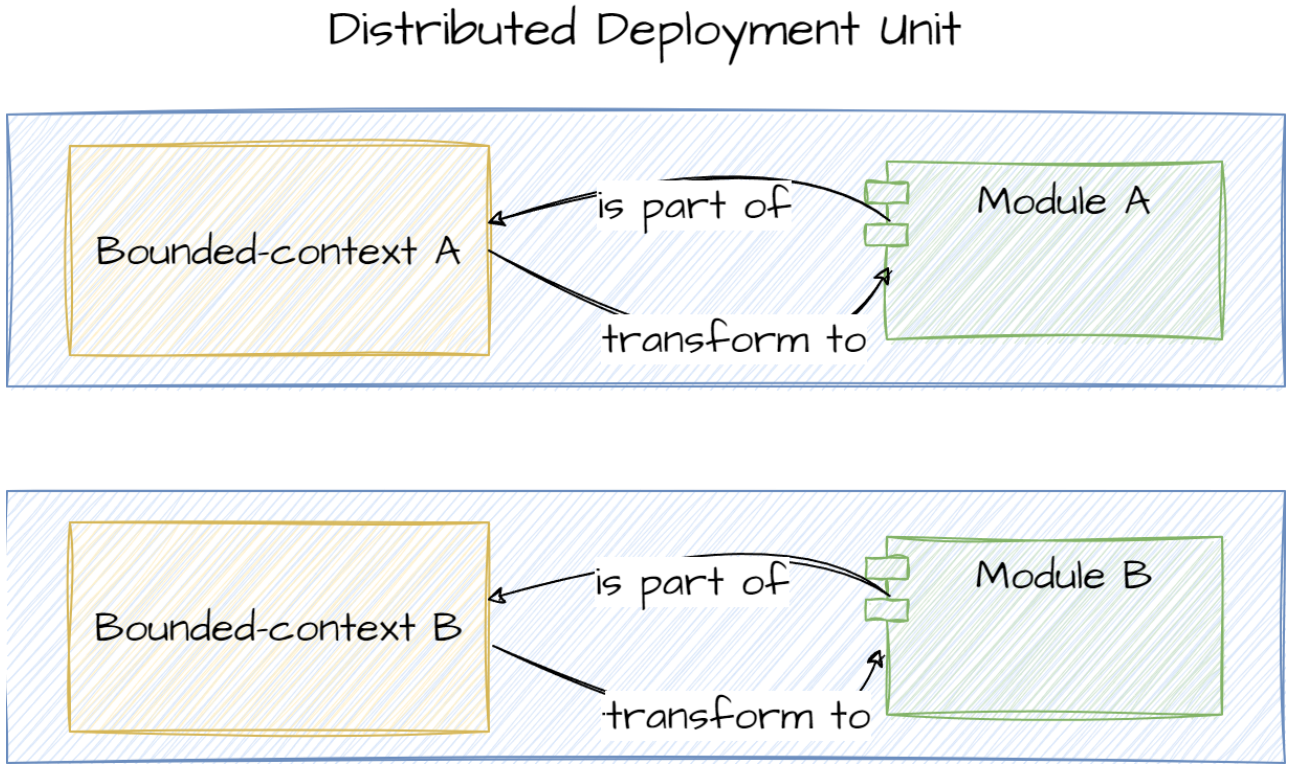
Logical grouping of your code
that is based on a domain.



RELATIONSHIP OF BOUNDED CONTEXT AND MODULES



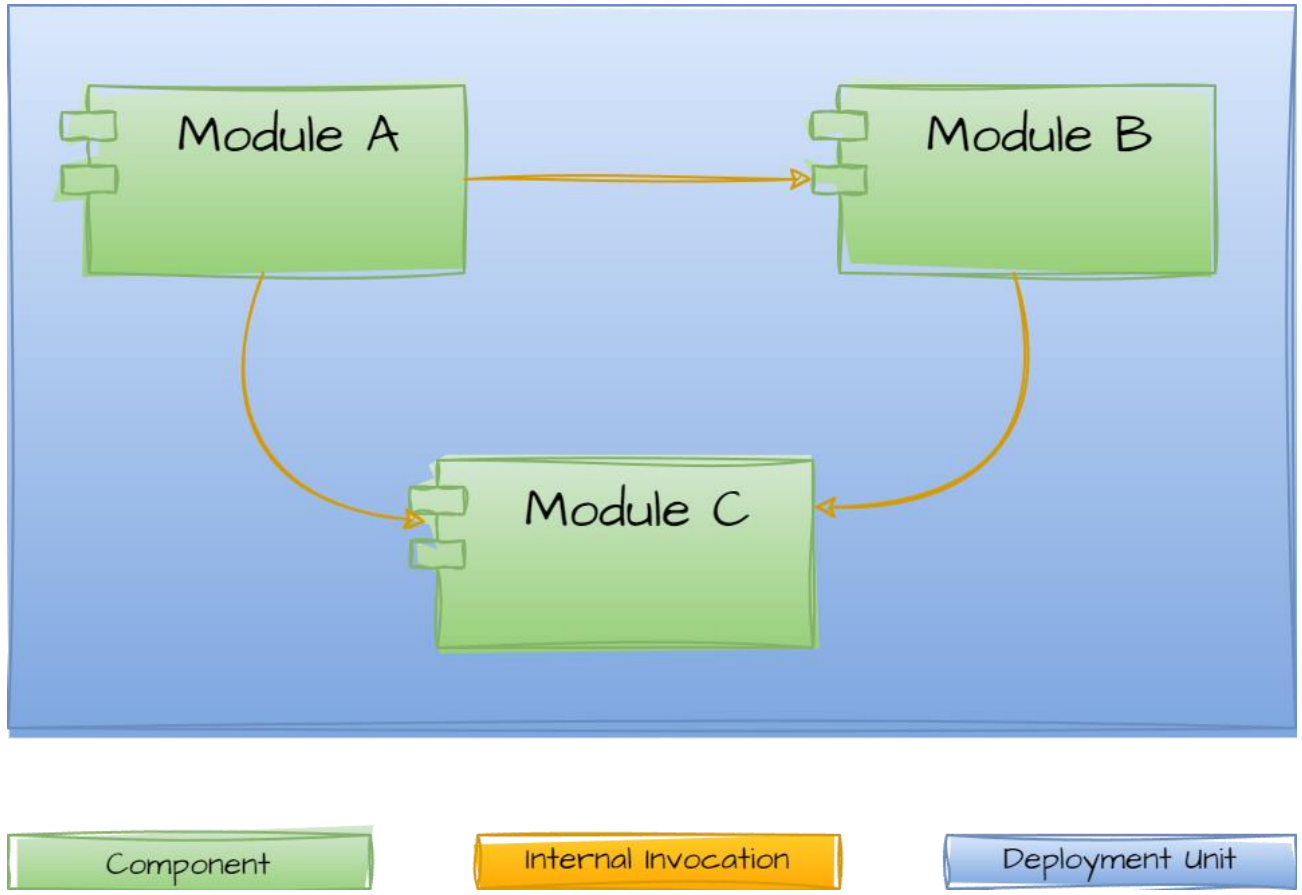
RELATIONSHIP OF BOUNDED CONTEXT AND MODULES



MONOLITH

Components are packaged in a single deployment unit.

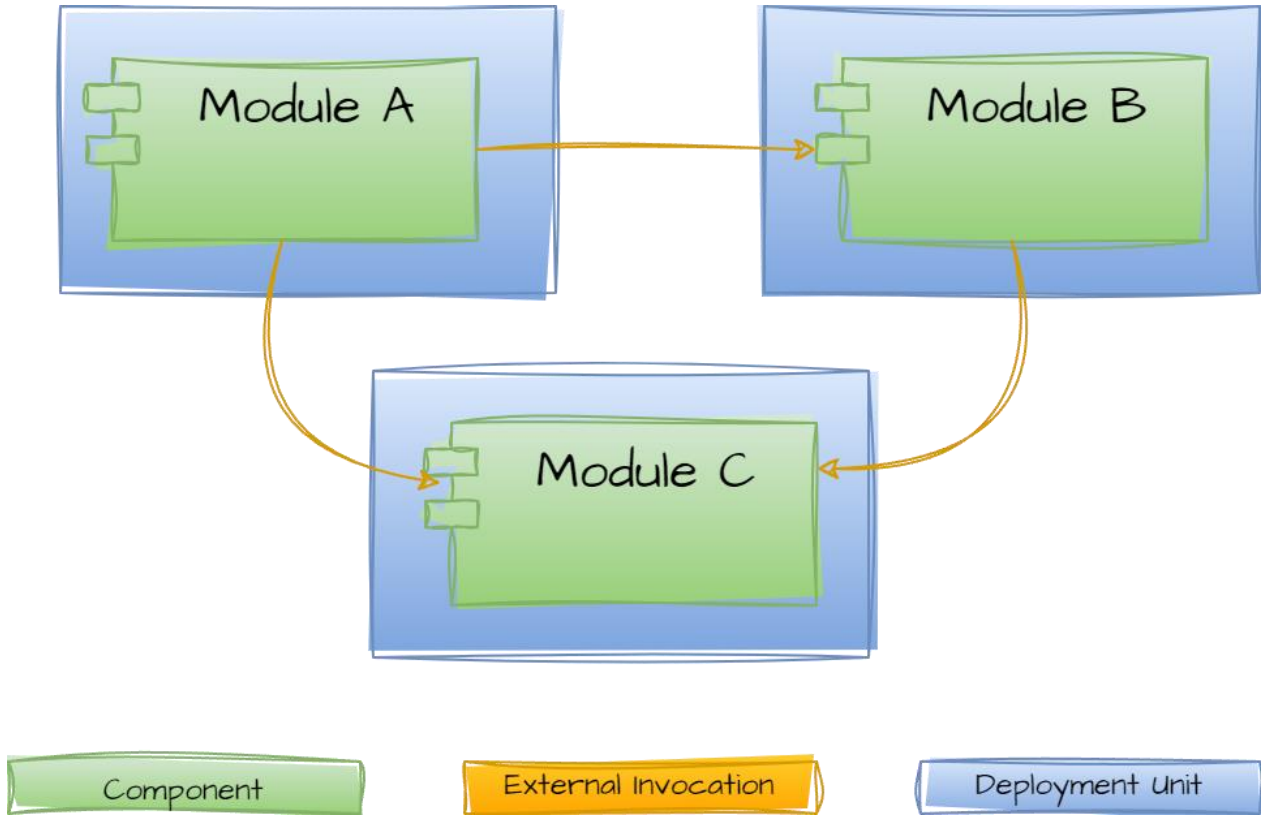
Components are composed of services that are inter-connected of each other.



MICROSERVICE

A set of components interacting remotely in its own process.

Components are composed of services that are independent of each other.



MONOLITHIC VS MICROSERVICE

- ✓One artifact
- ✓Easy to refactor
- ✓Method Calls
- ✓Single Deployment
- ✓One Database
- ✓Transactions
- ✓Scale as a single unit
- ✓Owned by a single team
- ✓One Platform

- ✓Separate artifacts
- ✓Difficult to refactor
- ✓Network Calls
- ✓Multiple Deployment
- ✓Many Databases
- ✓Eventually Consistent
- ✓Scale separately
- ✓Owned by multiple teams
- ✓Different Platforms

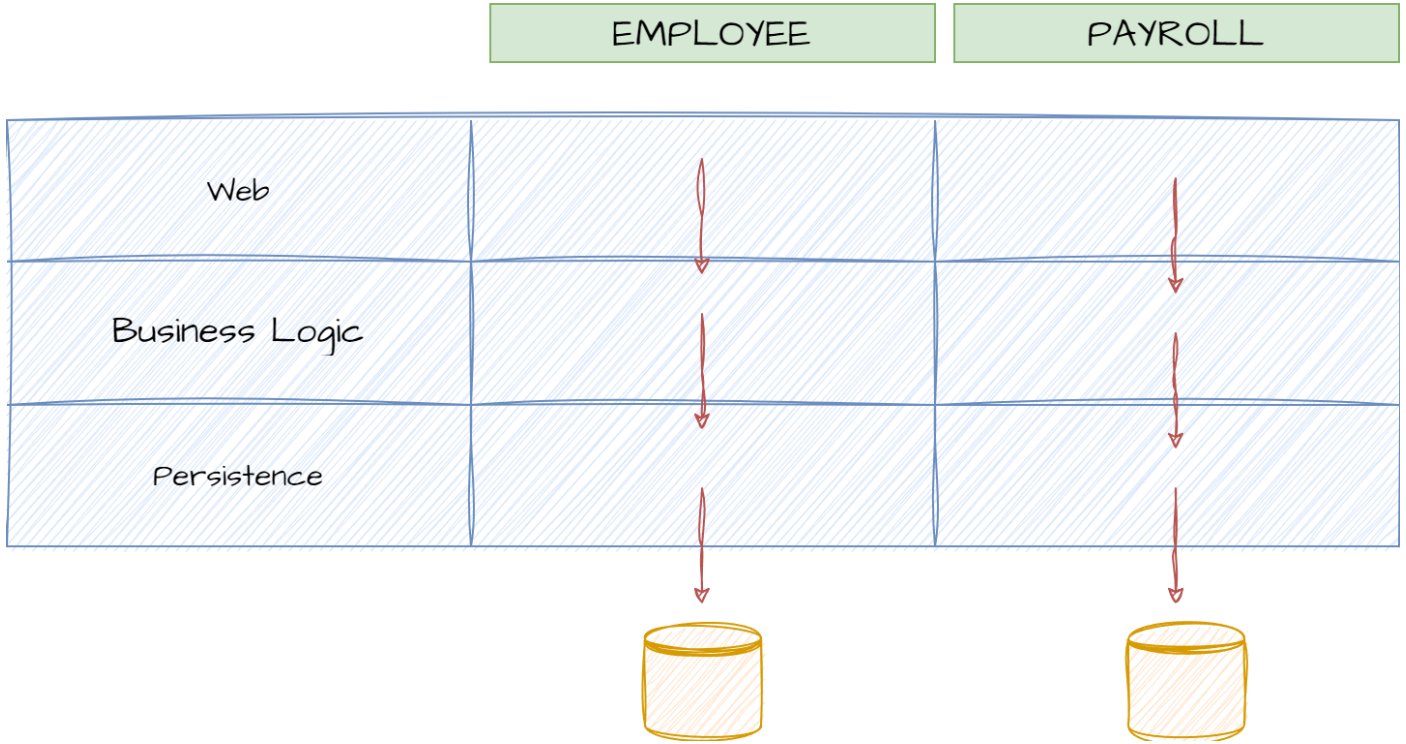
MONOLITHIC VS MICROSERVICE – WHEN TO USE?

- ✓Simple Applications
- ✓Small Teams
- ✓Good for MVPs
- ✓Good for Phase 1 Projects

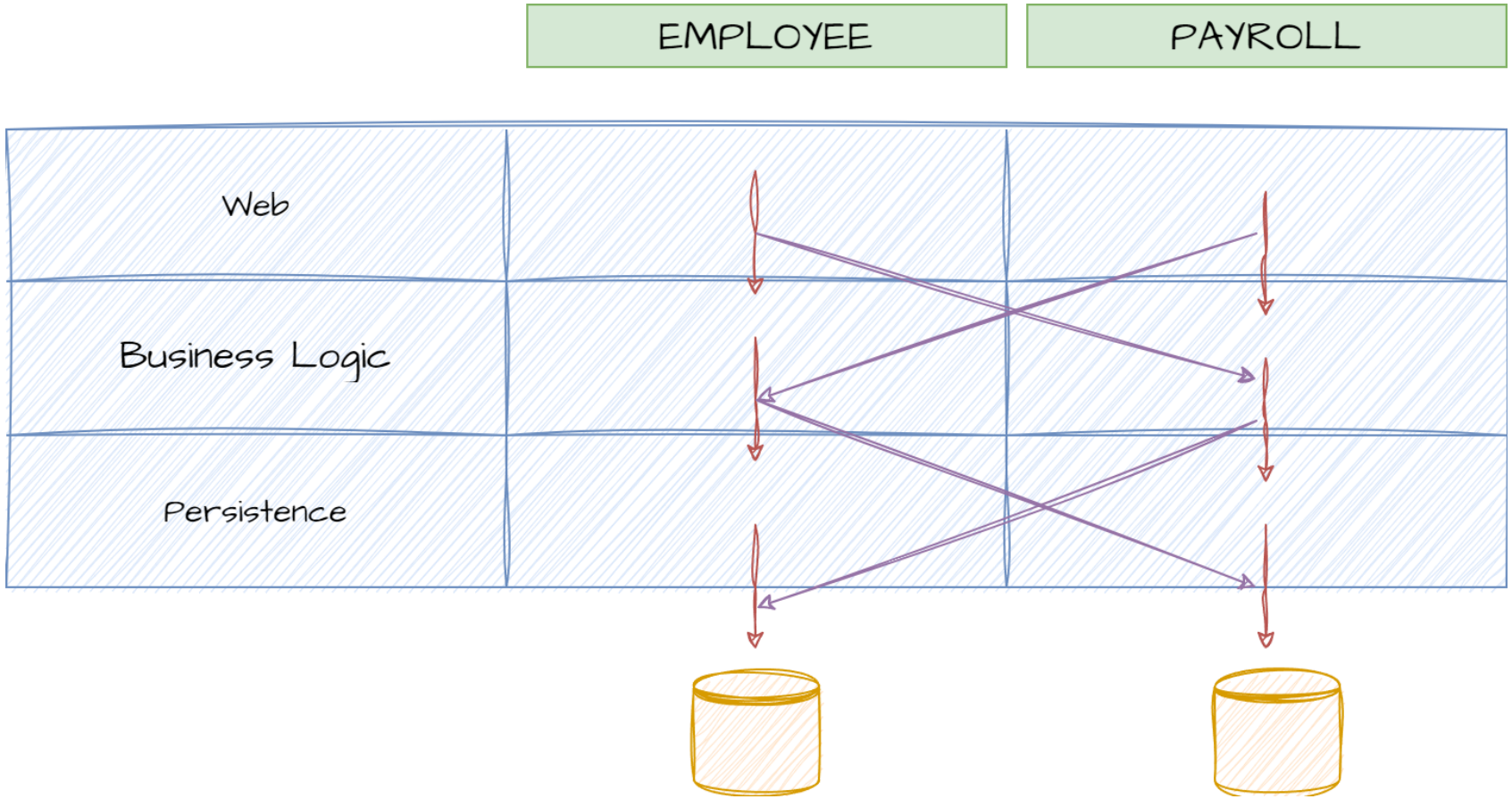
- ✓Complex Applications
- ✓Distributed Teams
- ✓Large Organizations
- ✓DevOps Culture

LAYERED ARCHITECTURE

The usual way to create an application from scratch.
Follows the *technical partitioning/structure* of components.



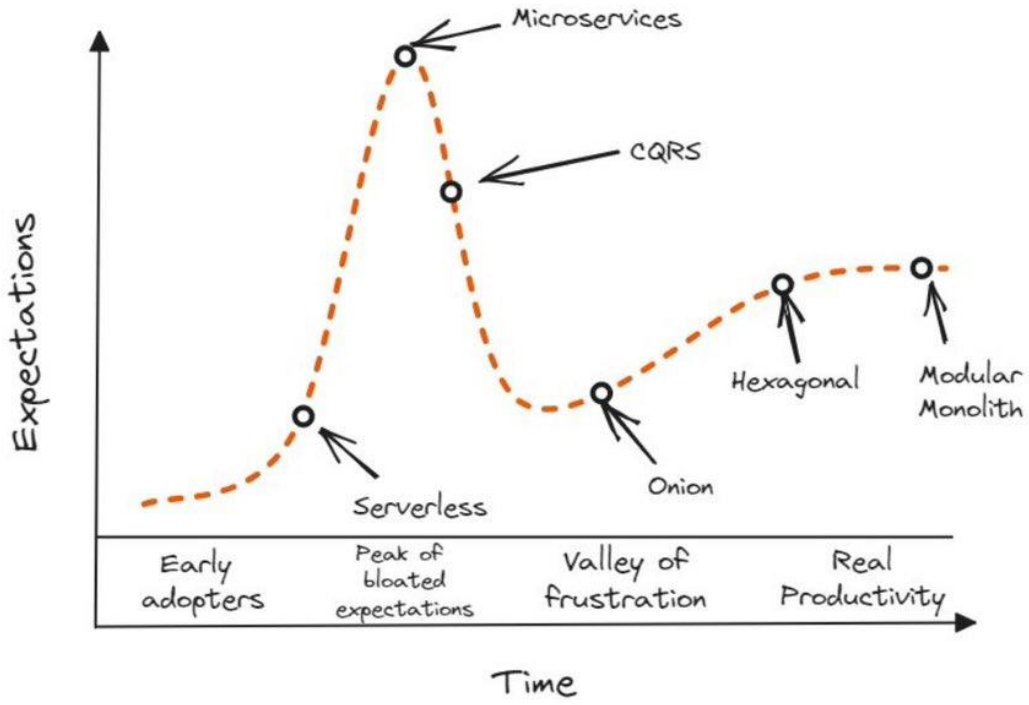
A BIG BALL OF MUD



MODULAR MONOLITHIC

A stepping-stone to Microservice

Software Architecture Hype Cycle



TechWorld WithMilan
simplifying complex topics

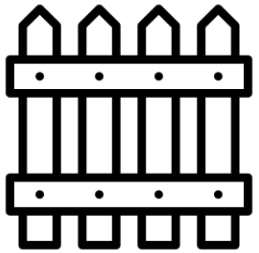
Why should we avoid creating microservice first?

“ You shouldn't start a new project with microservices, even if you're sure your application will be big enough to make it worthwhile. “

- Martin Fowler

YAGNI
(You Ain’t Gonna Need It)

- ✓ Build Simple Structure First
- ✓ Avoid Pre-mature Optimization



DOMAIN BOUNDARIES

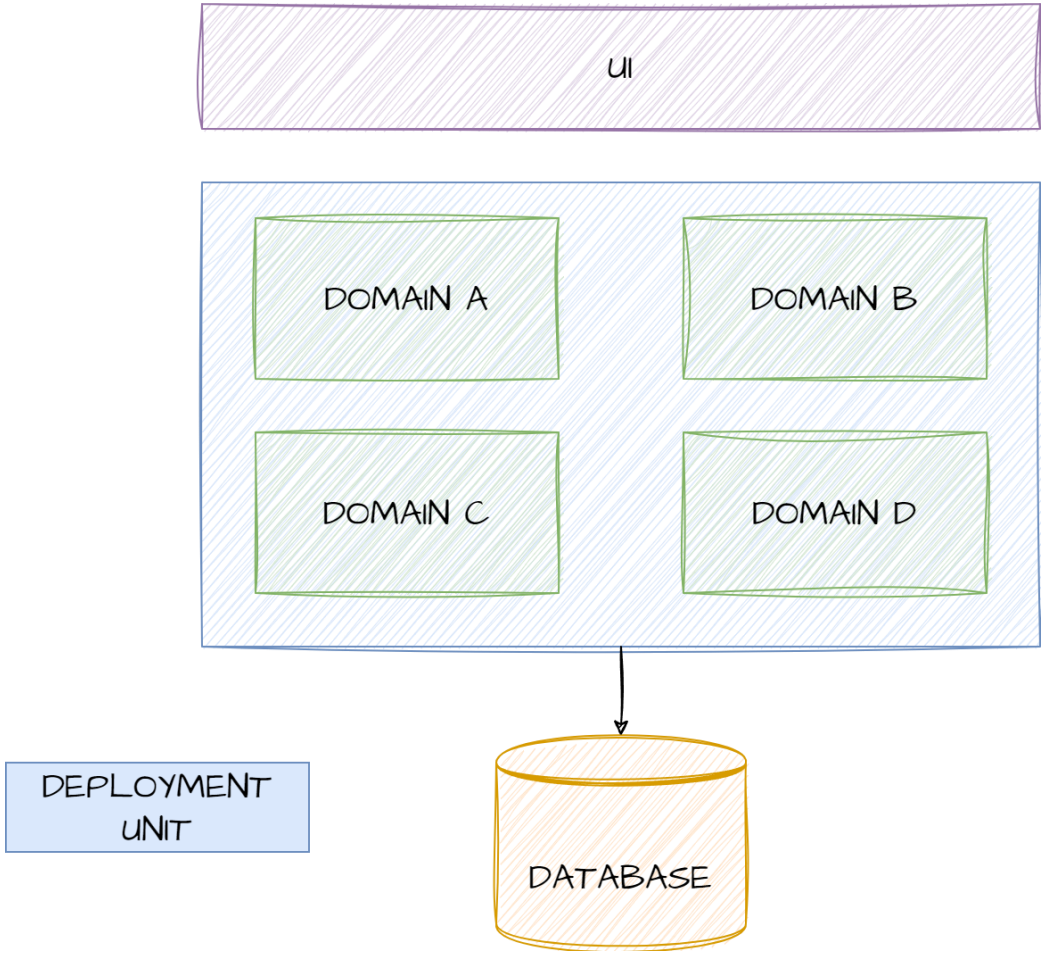
- ✓ Define your Bounded Context
- ✓ Discover your Domains

MODULAR MONOLITH

A monolithic architecture that represent its functionality based on its domain area.

These domain areas is composed of *modules*.

MODULAR MONOLITHIC ARCHITECTURE



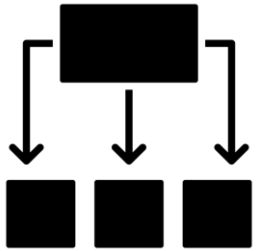
DOMAIN-BASED STRUCTURE

Grouping of your codes and directories as close as possible to the defined domain.

DOMAIN-BASED BOUNDARY

With the help of bounded context, we defined the purpose of the code structure in a functional way.

THE VALUE OF MODULAR MONOLITHIC



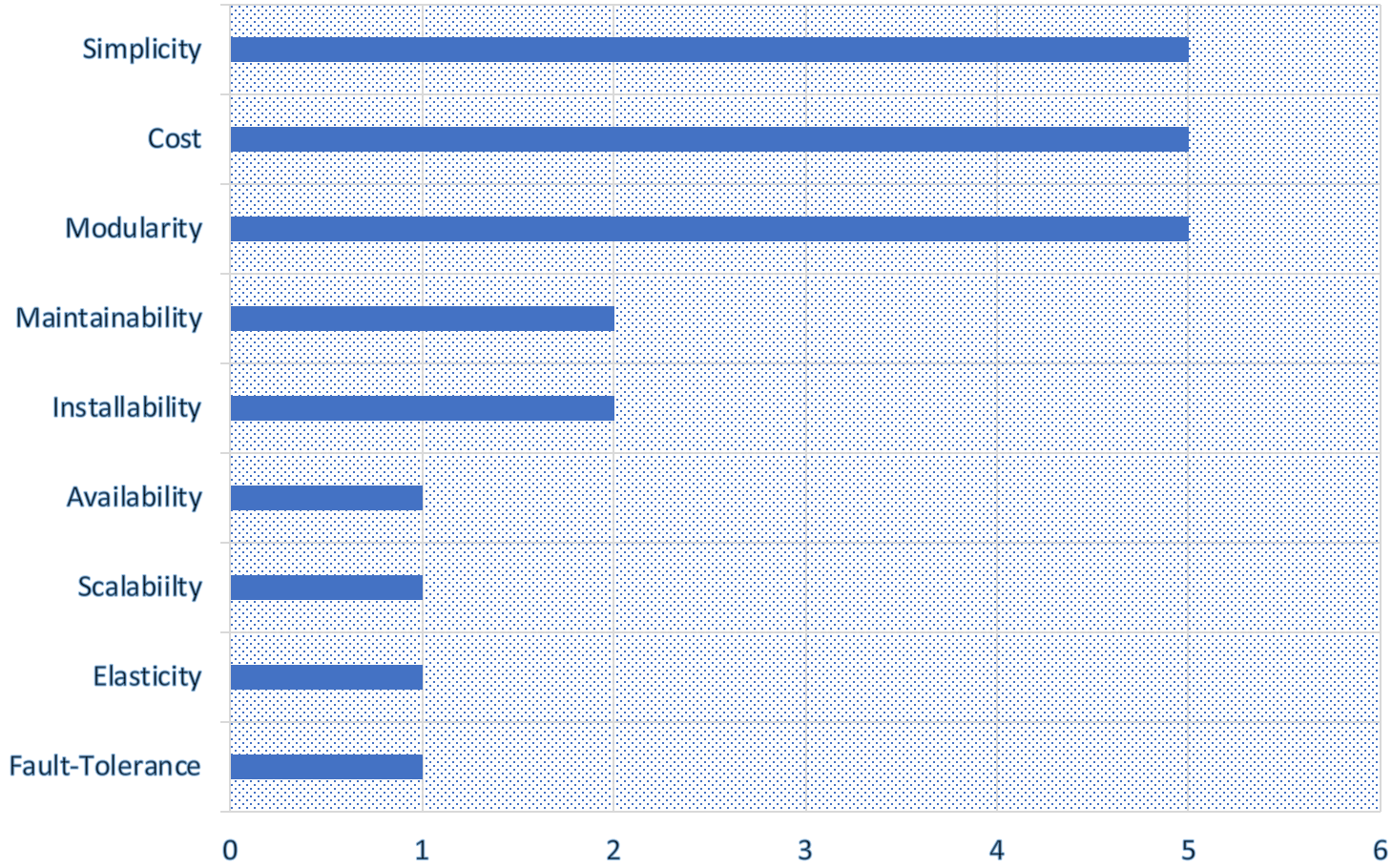
Functional Decomposition

Simple Architecture

Time Constraint

Budget Constraint

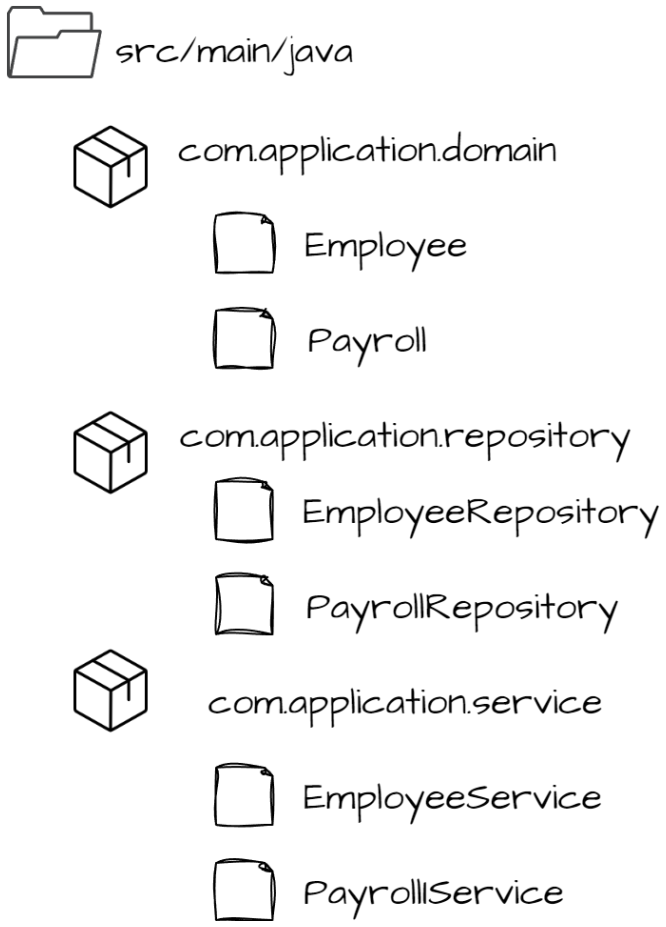
MODULAR MONOLITHIC – ARCHITECTURAL CHARACTERISTICS



PRACTICAL APPROACHES

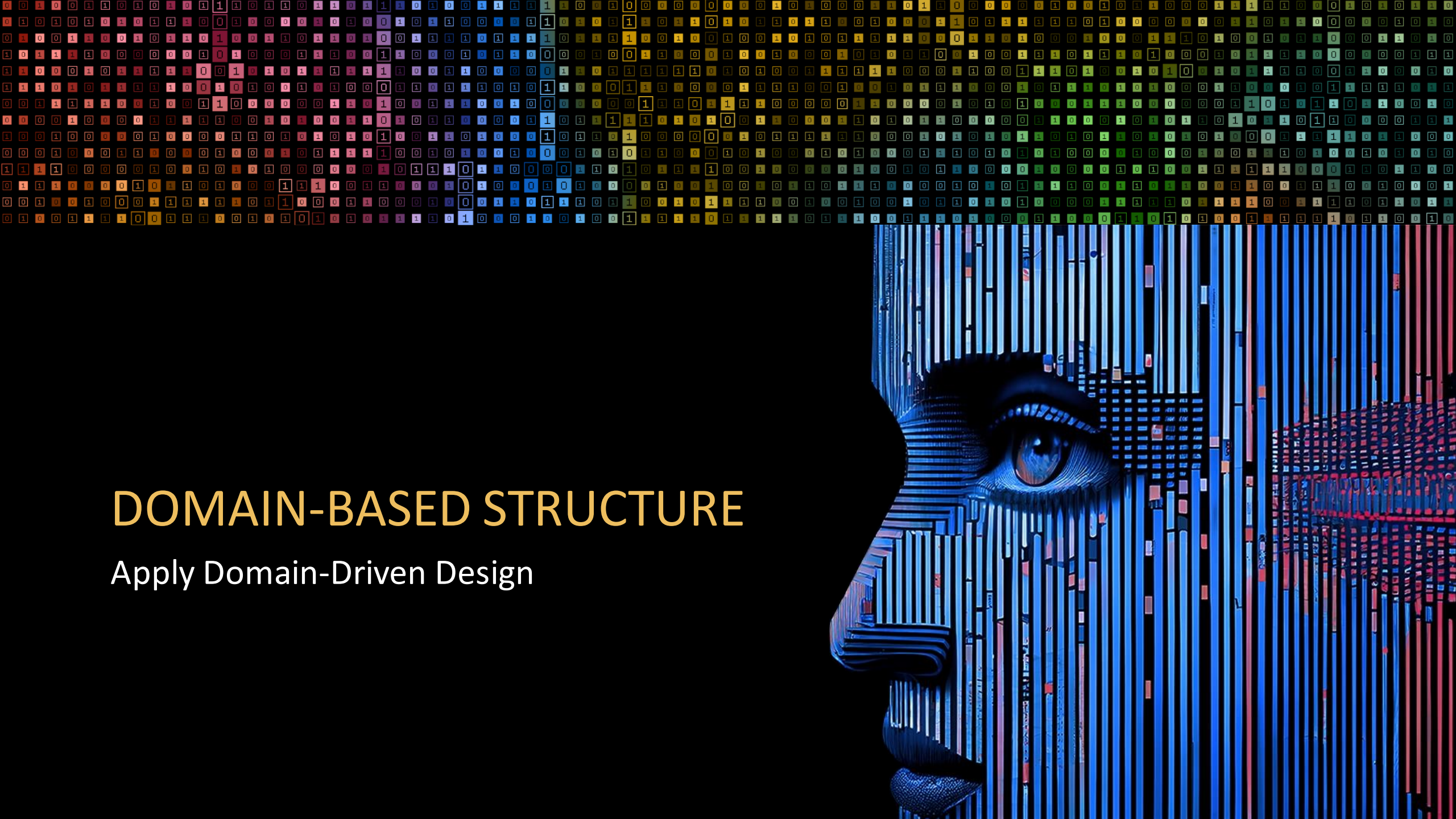
Ways to Create a Modular Monolithic

TECHNICAL PARTITIONING



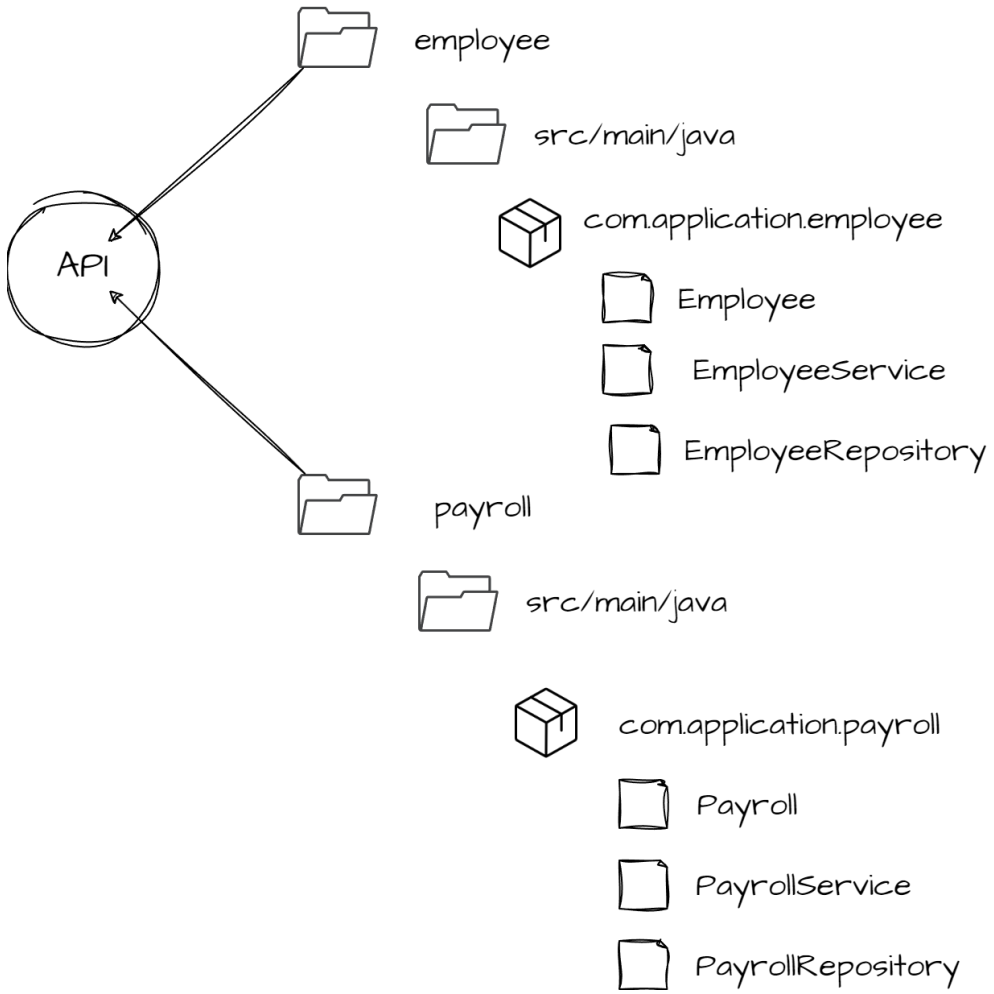
DOMAIN-BASED STRUCTURE

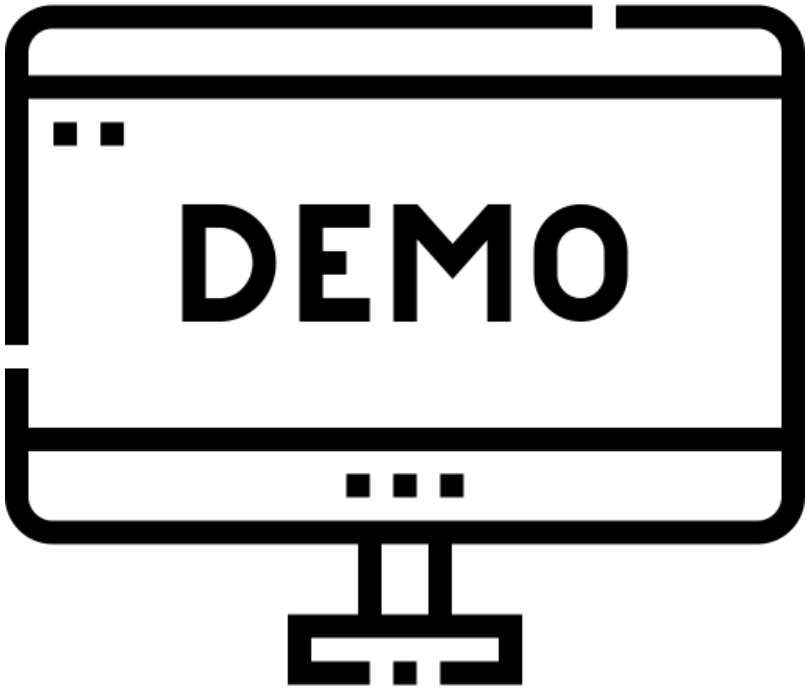
Apply Domain-Driven Design



GRADLE MULTI-MODULE

Expose each module as an API. Minimize visibility of implementations.



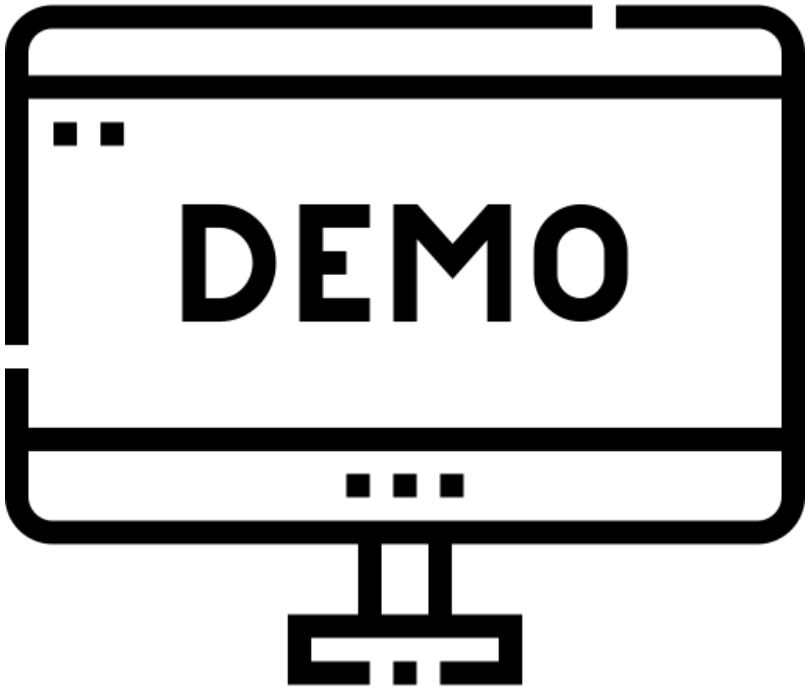

DEMO

HOW TO VERIFY MODULAR STRUCTURE?

ARCHUNIT

ArchUnit is a free, simple and extensible library for checking the architecture of your Java code using any plain Java unit test framework. That is, ArchUnit can check dependencies between packages and classes, layers and slices, check for cyclic dependencies and more. It does so by analyzing given Java bytecode, importing all classes into a Java code structure. You can find examples for the current release at ArchUnit Examples and the sources on GitHub.

Source: ArchUnit Website

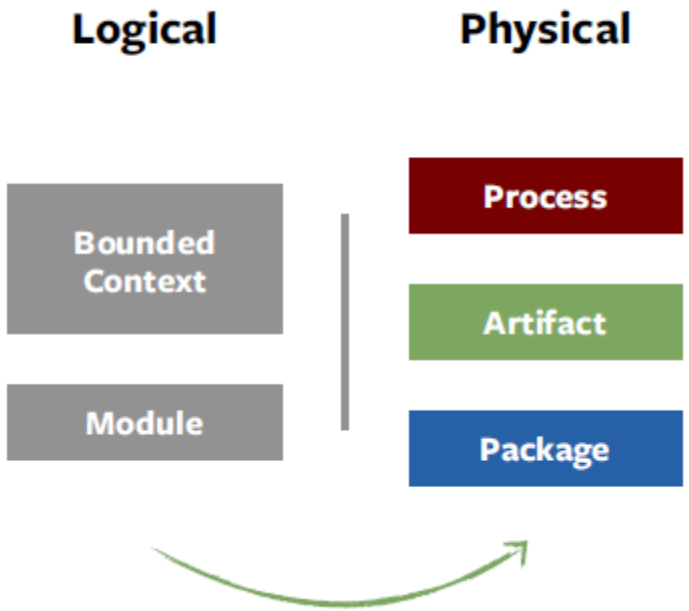

DEMO

SPRING MODULITH

Spring Modulith allows developers to build well-structured Spring Boot applications and guides developers in finding and working with application modules driven by the domain. It supports the verification of such modular arrangements, integration testing individual modules, observing the application's behavior on the module level and creating documentation snippets based on the arrangement created.

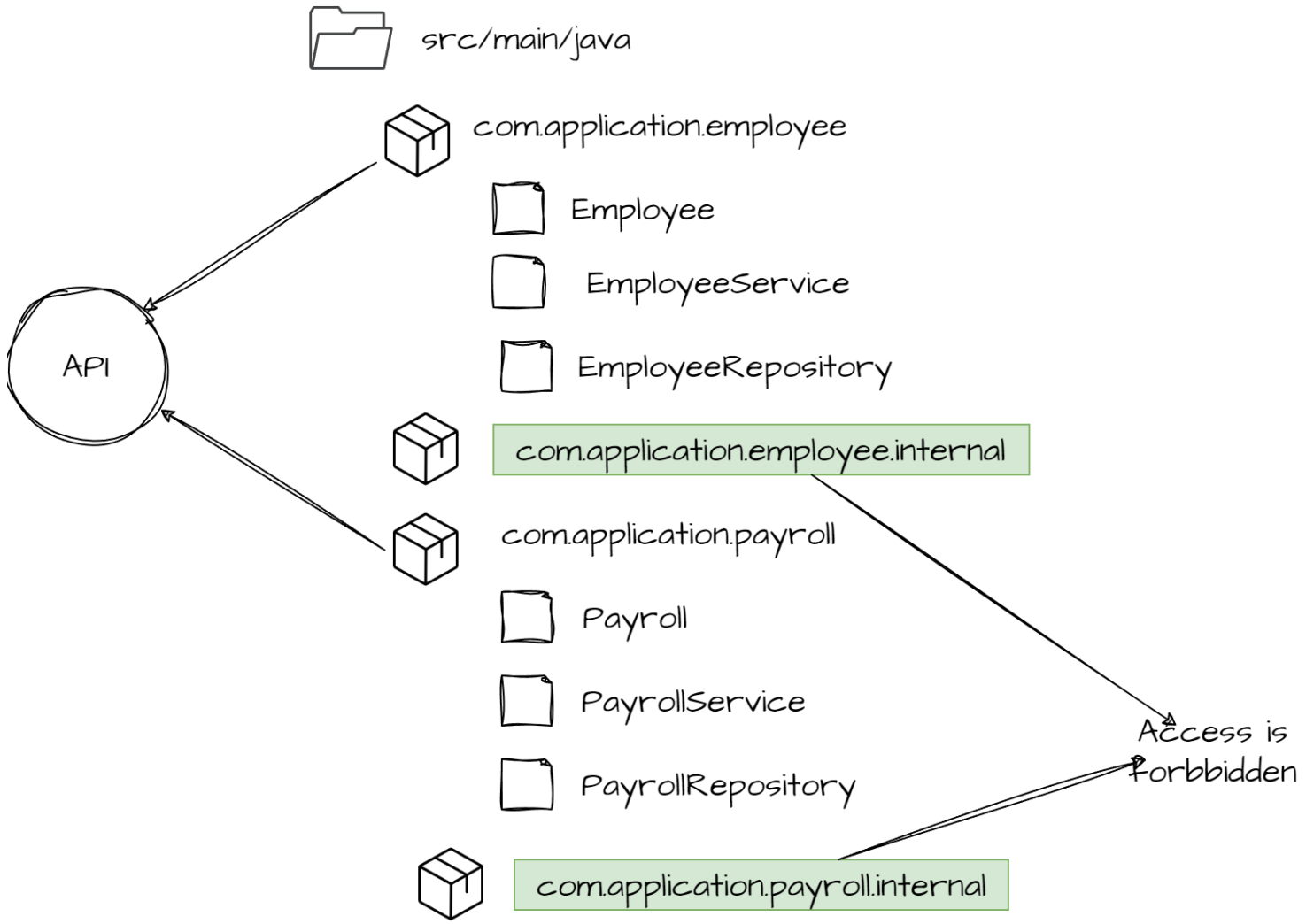
- Source: Spring Modulith Documentation

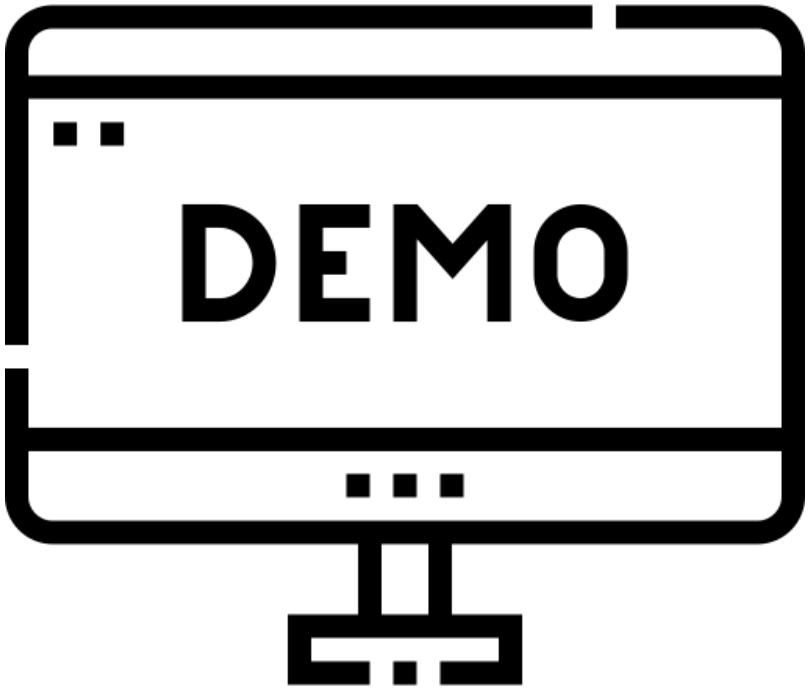
HOW SPRING MODULITH WORKS?



Spring Modulith Deep Dive by Oliver Drotbohm

API PACKAGE CONVENTION




DEMO

WILL YOU CONSIDER MODULAR MONOLITHS IN YOUR PROJECTS?

SUMMARY

SUMMARY

- The problem without modularity
- Modularization
- Domain-driven Approach
- Monolith and Microservice
- Modular Monolithic Benefits
- Explore Implementations of Modular Monolithic Design

REFERENCES

Learning Domain-Driven Design
Vlad Khononov – Book

Fundamentals of Software Architecture
Mark Richards and Neal Ford – Book

Software Architecture: The Hard Parts
Mark Richards, Neal Ford, Pramod Sadalge and Zhamak Dehghani – Book

Majestic Modular Monoliths
Axel Fontaine – Youtube

Modular Monoliths Architecture
March Richards – Youtube

Spring Modulith – A Deep Dive (Workshop)
Oliver Drotbohm – Youtube

Modular Monolith with DDD
Kamil Grzybek – GitHub

Q & A

rjtmahinay

me@rjtmahinay.com