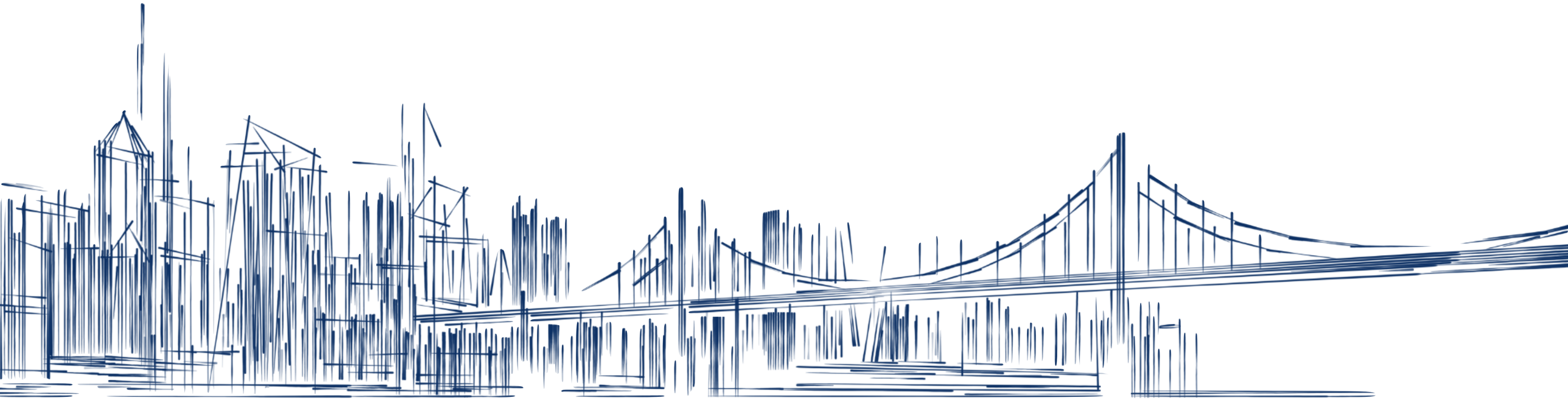


BOTTOM-UP ARCHITECTURE

BRIDGING THE ARCHITECTURE CODE GAP

Oliver Drotbohm odrotbohm oliver.drotbohm@broadcom.com

github.com/odrotbohm

odrotbohm

Type / to search

Overview | Repositories 123 | Projects 1 | Packages | Stars 77

Oliver Drotbohm

odrotbohm · he/him

Frameworks & Architecture Engineering @ VMware, OpenSource enthusiast, all things Spring, Java, data, DDD, REST, software architecture, drums & music.

Edit profile

3.4k followers · **32** following

- VMware
- Dresden, Germany
- 17:49 (UTC +01:00)
- info@odrotbohm.de
- www.odrotbohm.de
- @odrotbohm
- @odrotbohm@chaos.social
- odrotbohm
- in/odrotbohm

Pinned

Customize your pins

xmolecules/jmolecules Public
Libraries to help developers express architectural abstractions in Java code
Java ☆ 979 ⑂ 85

xmolecules/jmolecules-integrations Public
Technology integration for jMolecules
Java ☆ 48 ⑂ 11

lectures Public
Lecture scripts and slides I use during the Software Engineering course at TU Dresden
Java ☆ 68 ⑂ 22

spring-restbucks Public
Implementation of the sample from REST in Practice based on Spring projects
Java ☆ 1.2k ⑂ 404

spring-playground Public
A collection of tiny helpers for building Spring applications
Java ☆ 96 ⑂ 10

spring-projects/spring-modulith Public
Modular applications with Spring Boot
Java ☆ 531 ⑂ 67

Single sign-on to see contributions within the **pivotal** organization.

1,559 contributions in the last year

Contribution settings

Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov

Mon
Wed
Fri

Learn how we count contributions

Less More

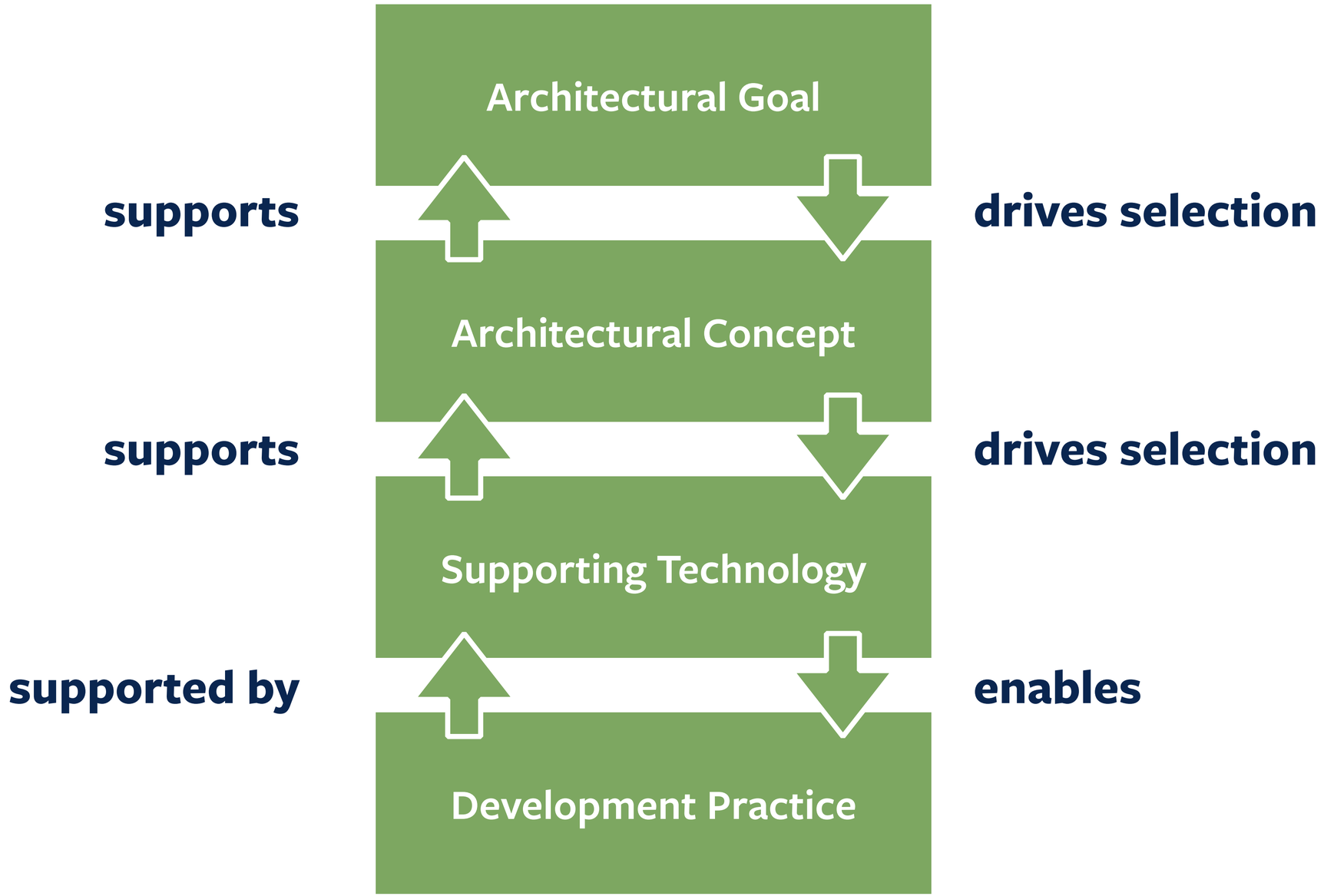
2023
2022
2021
2020
2019
2018
2017
2016
2015

@spring-projects | @st-tu-dresden | @xmolecules | More

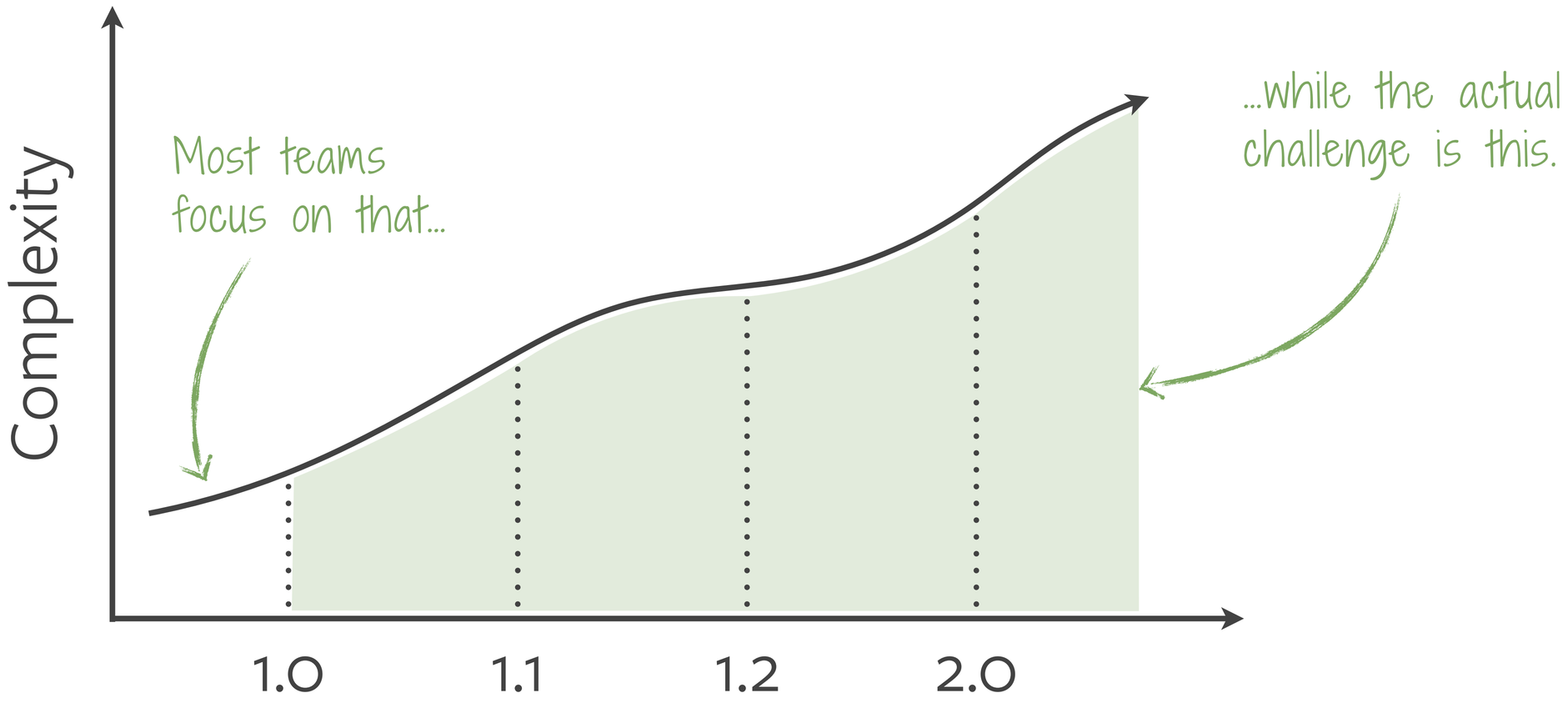
Activity overview

Contributed to

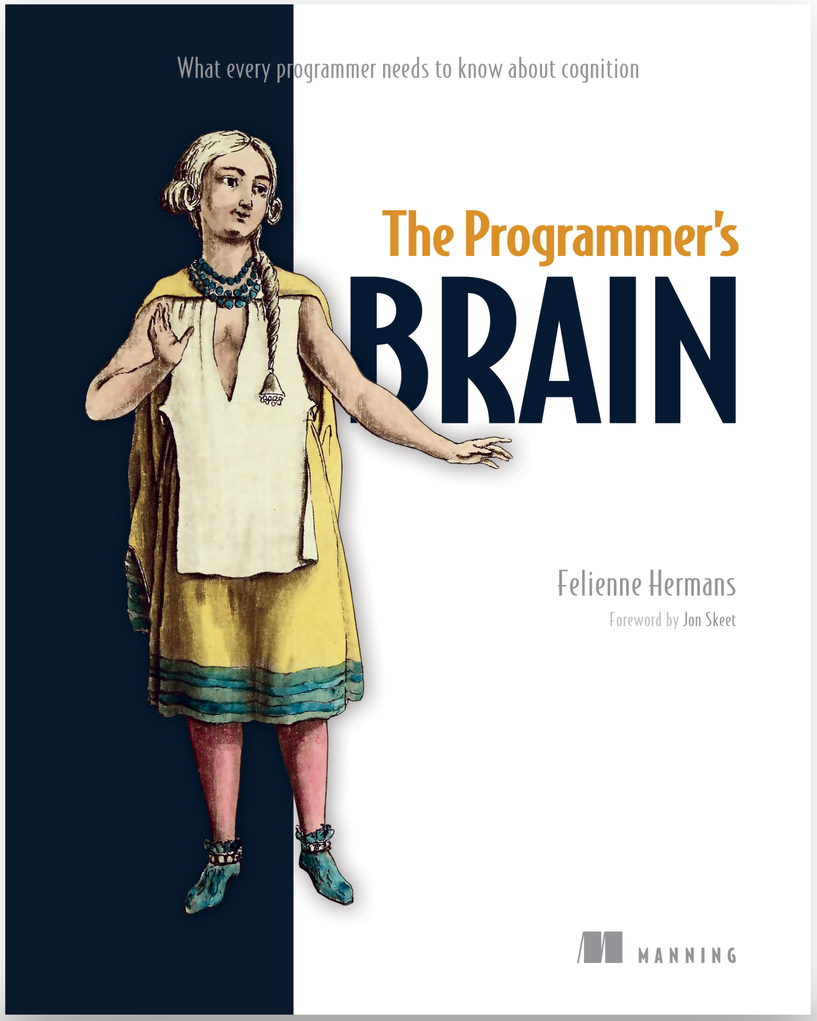
1%
Code review


Architectural Goal
supports
drives selection
Architectural Concept
supports
drives selection
Supporting Technology
supported by
enables
Development Practice

We want to build evolvable systems.


Complexity
Most teams
focus on that...
...while the actual
challenge is this.
1.0
1.1
1.2
2.0

Understandability


What every programmer needs to know about cognition
The Programmer's
BRAIN
Felienne Hermans
Foreword by Jon Skeet
MANNING


WPS WORKPLACE SOLUTIONS
dpunkt.verlag
Carola Lilienthal
Sustainable Software Architecture
Analyze and Reduce Technical Debt

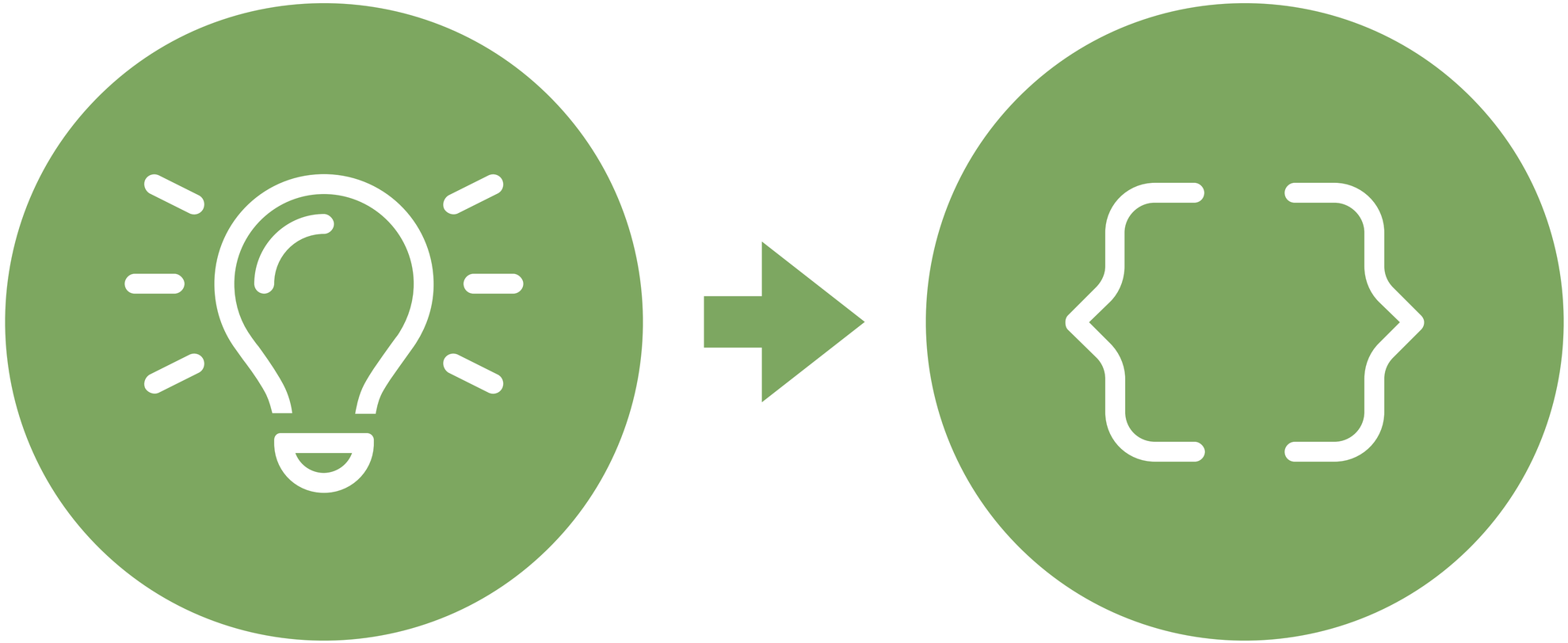
Chunking

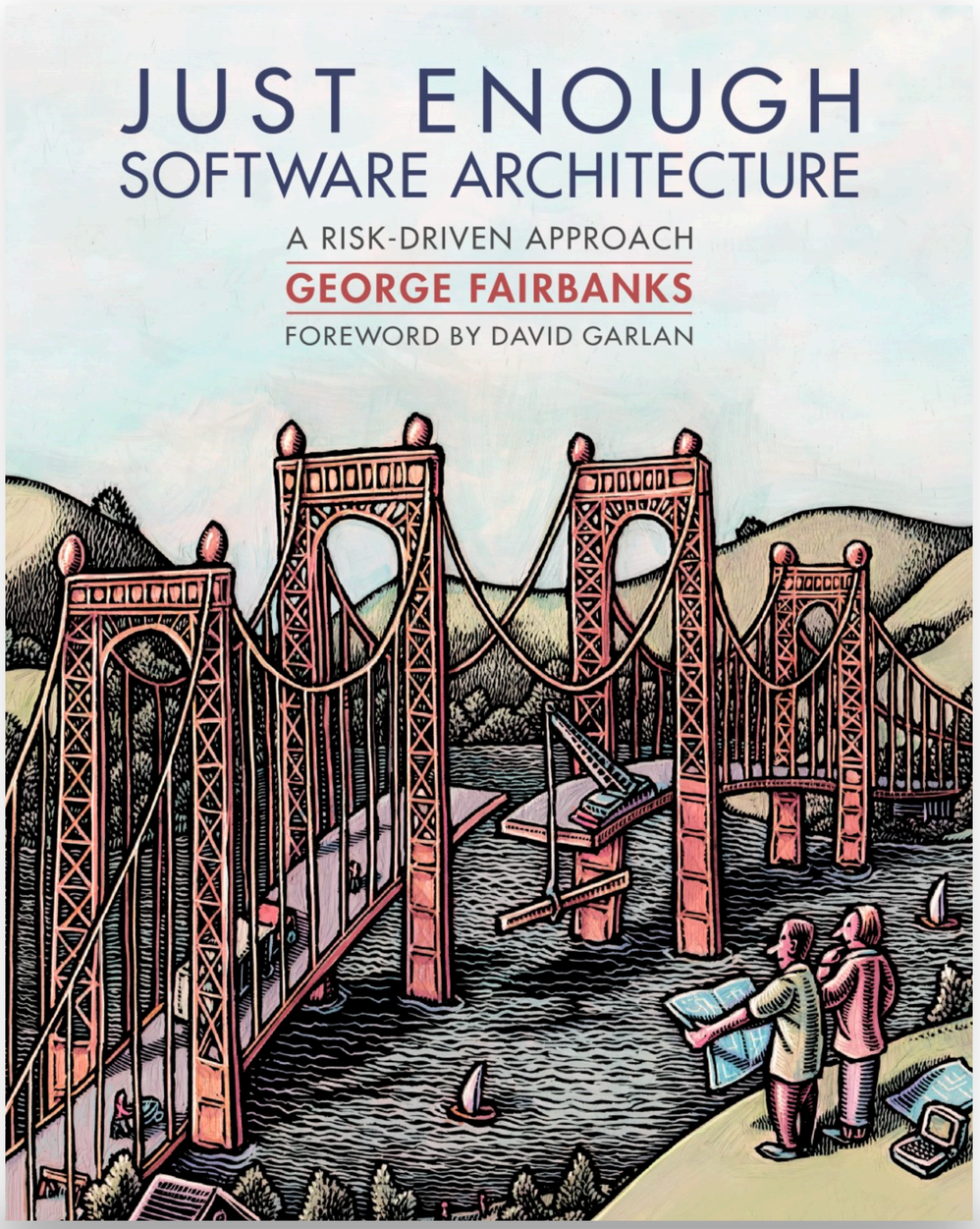
Hierarchization

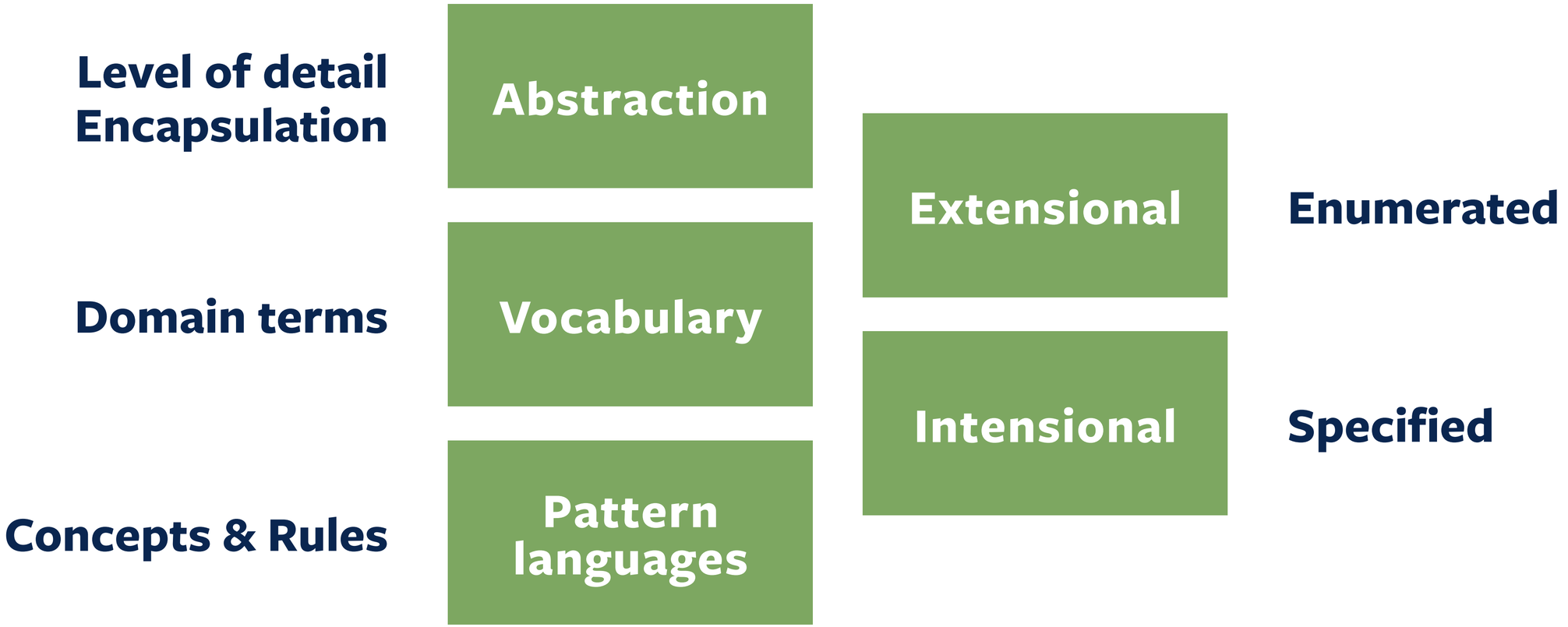
Pattern languages

"Architecture is a property of a system, not a description of its intended design."

— Stefan Tilkov in "Good Enough Architecture"




JUST ENOUGH
SOFTWARE ARCHITECTURE
A RISK-DRIVEN APPROACH
GEORGE FAIRBANKS
FOREWORD BY DAVID GARLAN


Level of detail
Encapsulation
Abstraction
Domain terms
Vocabulary
Concepts & Rules
Pattern languages
Extensional
Intensional
Enumerated
Specified

Architecturally-Evident Code?

Extensional

Components / Modules

Invoicing,
Shipment

Deployables / Build modules / Packages

Domain language

EmailAddress,
ZipCode

Classes, methods, fields

Intensional

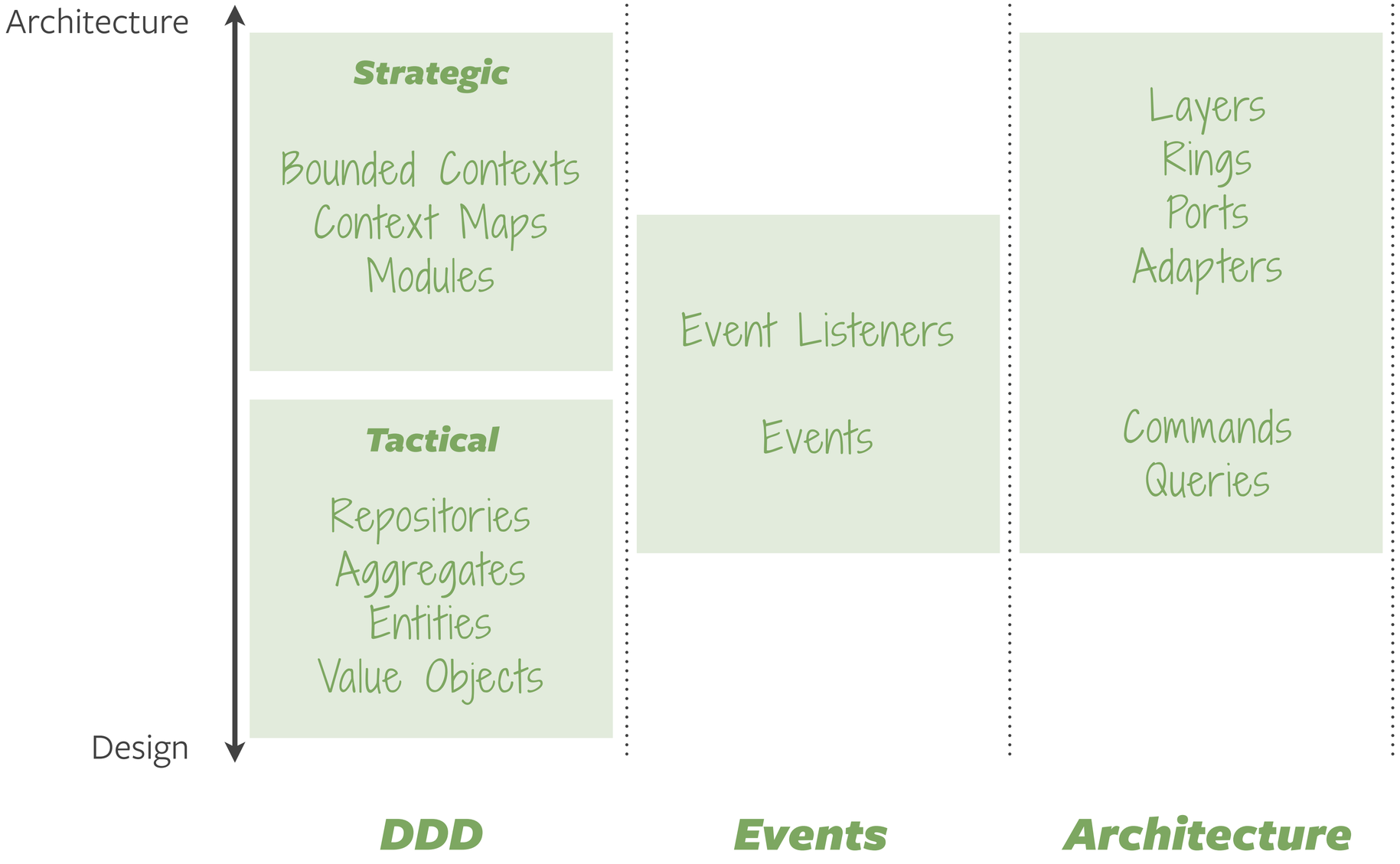
Concepts & Rules

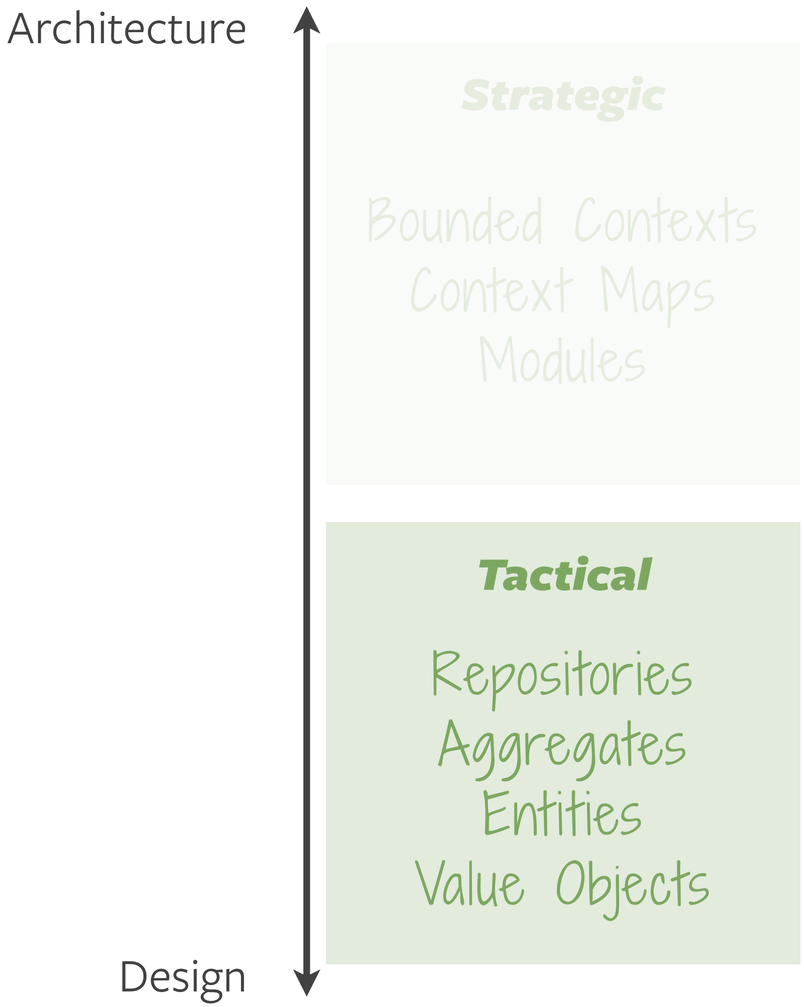
ValueObject,
Entity,
Aggregate

Layers,
Rings

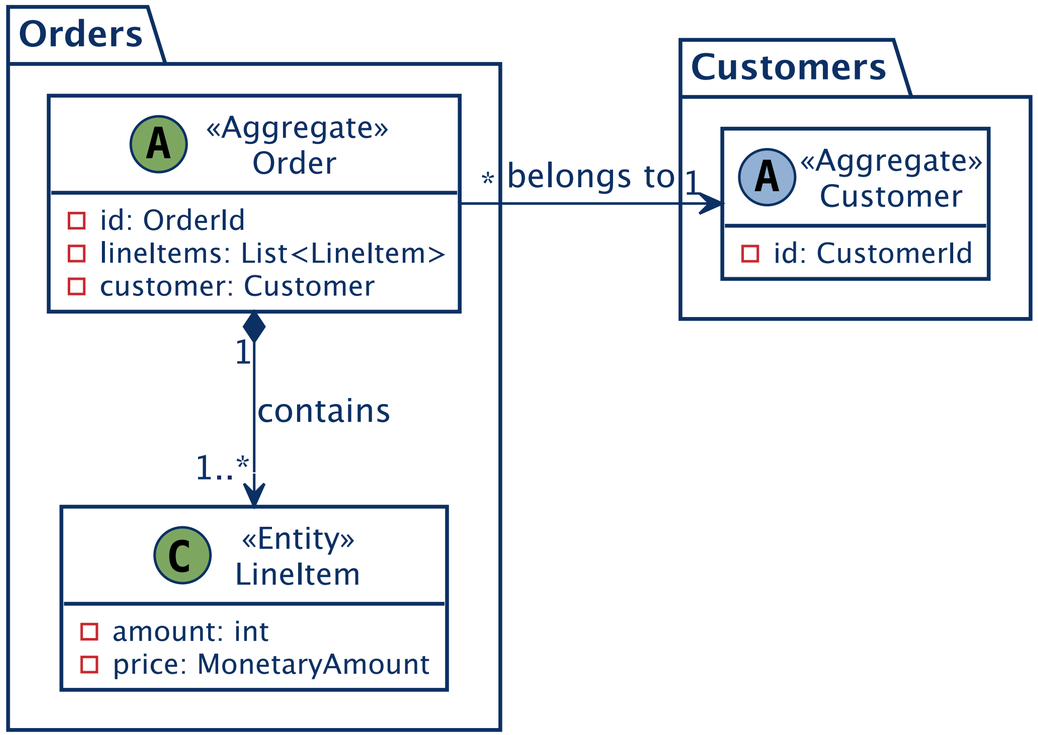
Naming conventions

What else? 🤔

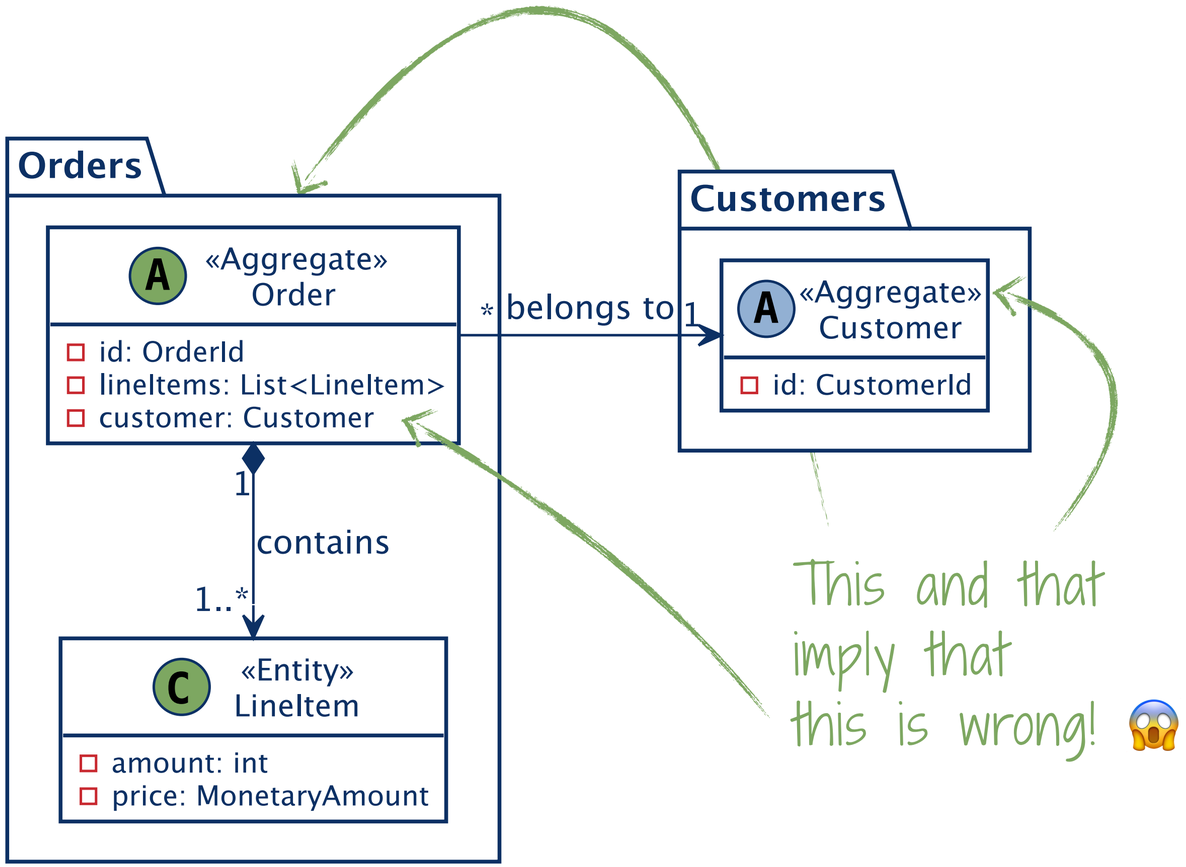

Architecture
Strategic
Bounded Contexts
Context Maps
Modules
Tactical
Repositories
Aggregates
Entities
Value Objects
Design
Event Listeners
Events
Layers
Rings
Ports
Adapters
Commands
Queries
DDD
Events
Architecture

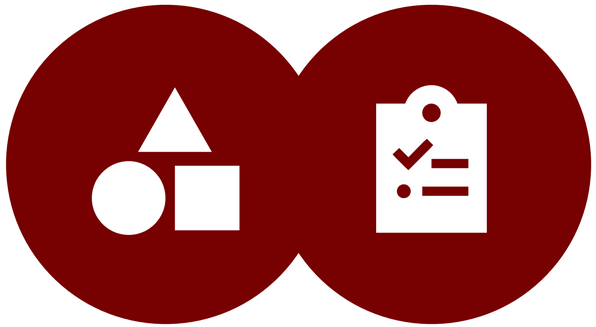

Architecture
Strategic
Bounded Contexts
Context Maps
Modules
Tactical
Repositories
Aggregates
Entities
Value Objects
Design

DDD



A simple Aggregate arrangement



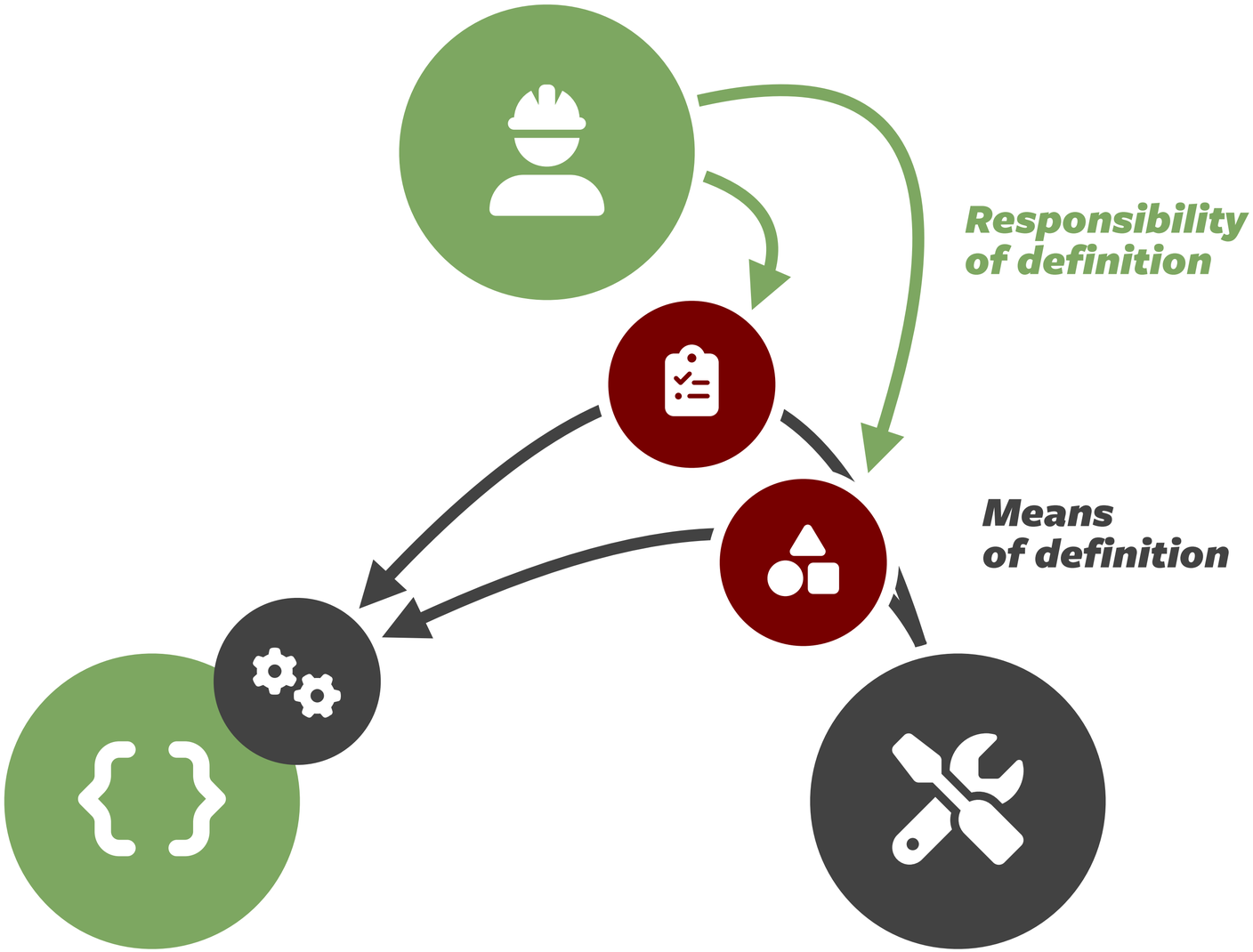


A simple Aggregate arrangement

Code

Architecture

Technology


Responsibility
of definition
Means
of definition

Your Software. Your Structures. Your Rules.

Establishing an Aggregate... in jQAssistant

```
MATCH
  (repo:Java:Type)
    -[:IMPLEMENTS_GENERIC]→ (superType)
    -[:OF_RAW_TYPE]→ (:Java:Type { fqn: "o.s.d.r.Repository"}),
  (superType)
    -[:HAS_ACTUAL_TYPE_ARGUMENT { index: 0 }]→ ()
    -[:OF_RAW_TYPE]→ (aggregateType)
SET
  aggregateType:Aggregate
RETURN
  repo, aggregateType
```

Reference to tech stack 😕

Establishes the concept

```
MATCH
  (aggregate:Aggregate)
    -[:DECLARES]→ (f:Field)
    -[:OF_TYPE]→ (fieldType:Aggregate)
WHERE
  aggregate <> fieldType
RETURN
  aggregate, fieldType
```

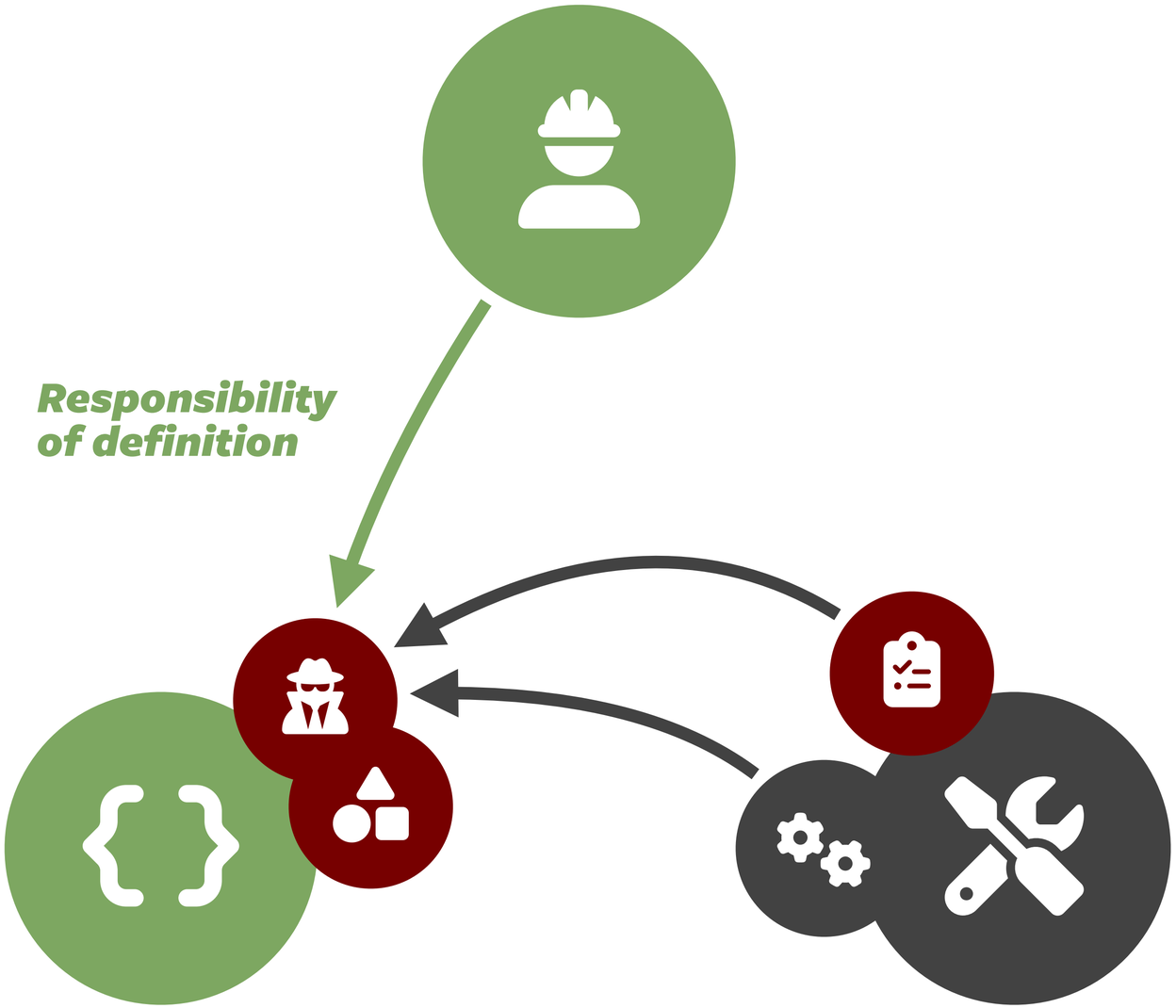
Establishes the rule

Establishing an Aggregate... in ArchUnit

```java
@AnalyzeClasses(packagesOf = Application.class)
public class ArchitectureTest {

  @ArchTest
  void verifyAggregates(JavaClasses types) {

    var aggregates = new AggregatesExtractor();
    var aggregateTypes = aggregates.doTransform(types);

    all(aggregates)
      .should(notReferToOtherAggregates(aggregateTypes))
      .check(types);
  }
}
```

Establishes the concept

Establishes the rule

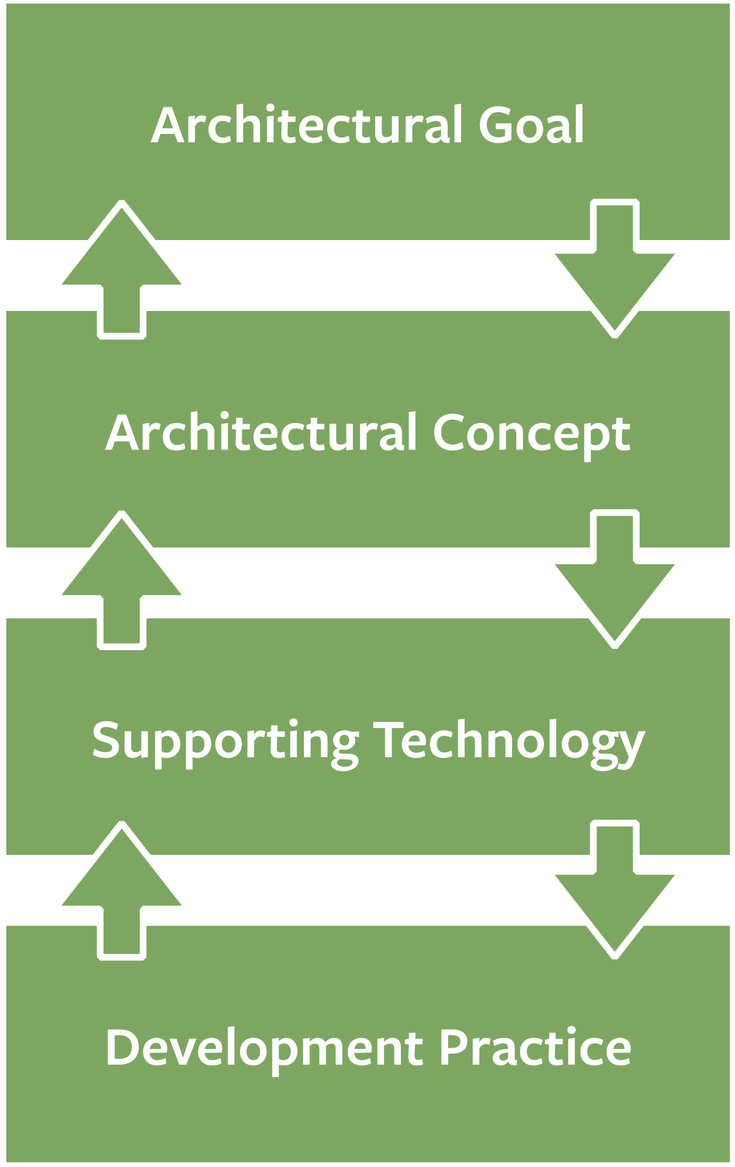

Responsibility
of definition

xMolecules

php

.NET

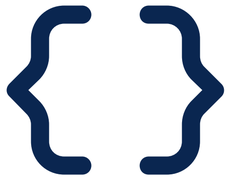
jMolecules



Evolvability

DDD Building Blocks

jMolecules

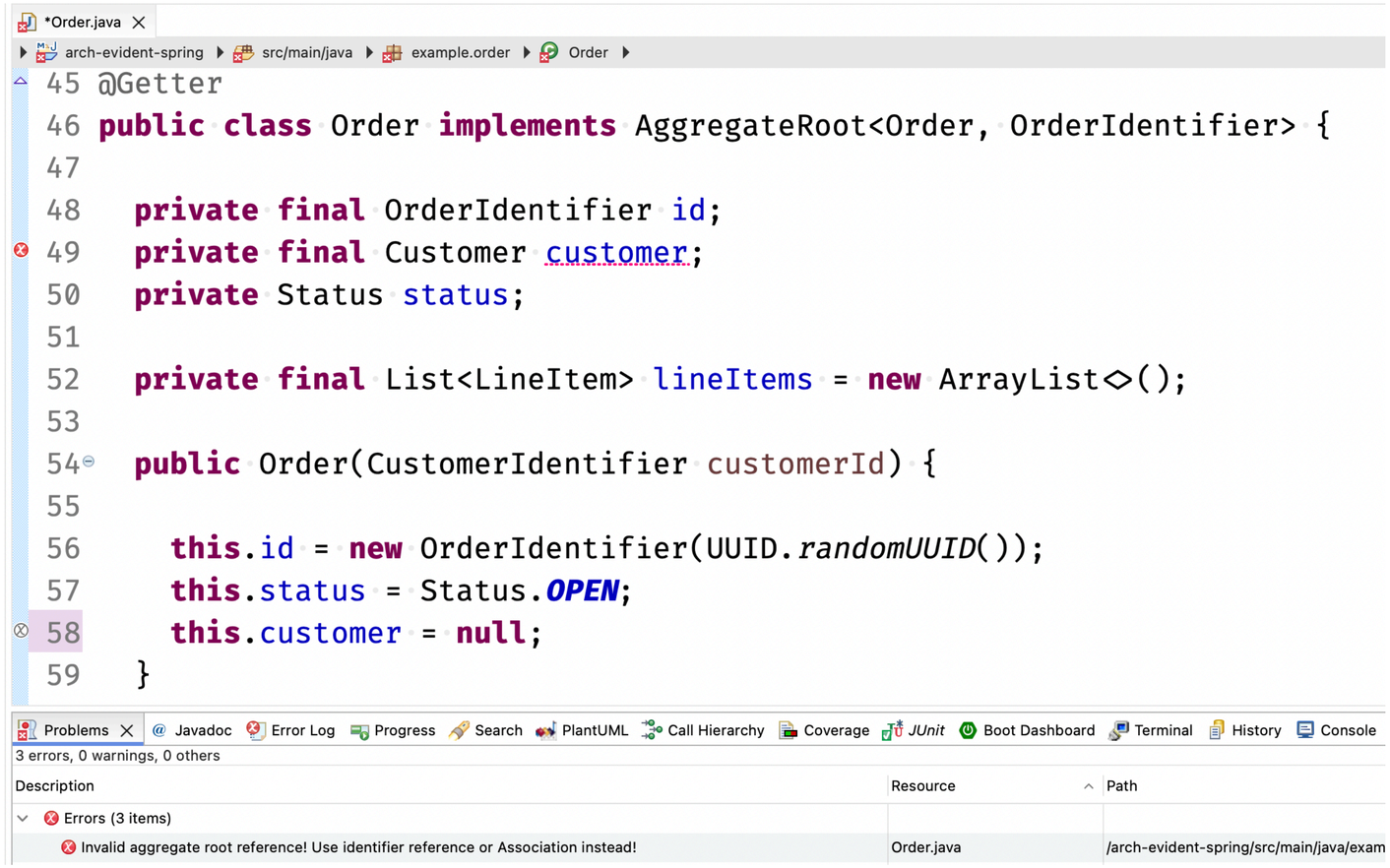


Explicit concepts

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements o.j.d.t.AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements o.j.d.t.Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

Verification



☝️ **Invalid aggregate root reference!**
Use identifier or Association instead!

Eliminate boilerplate


```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customerId;
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private Association<Customer, CustomerId> customer;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customer = Association.forId(customerId);
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

```java
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private Association<Customer, CustomerId> customer;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customer = Association.forId(customerId);
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Identifier {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

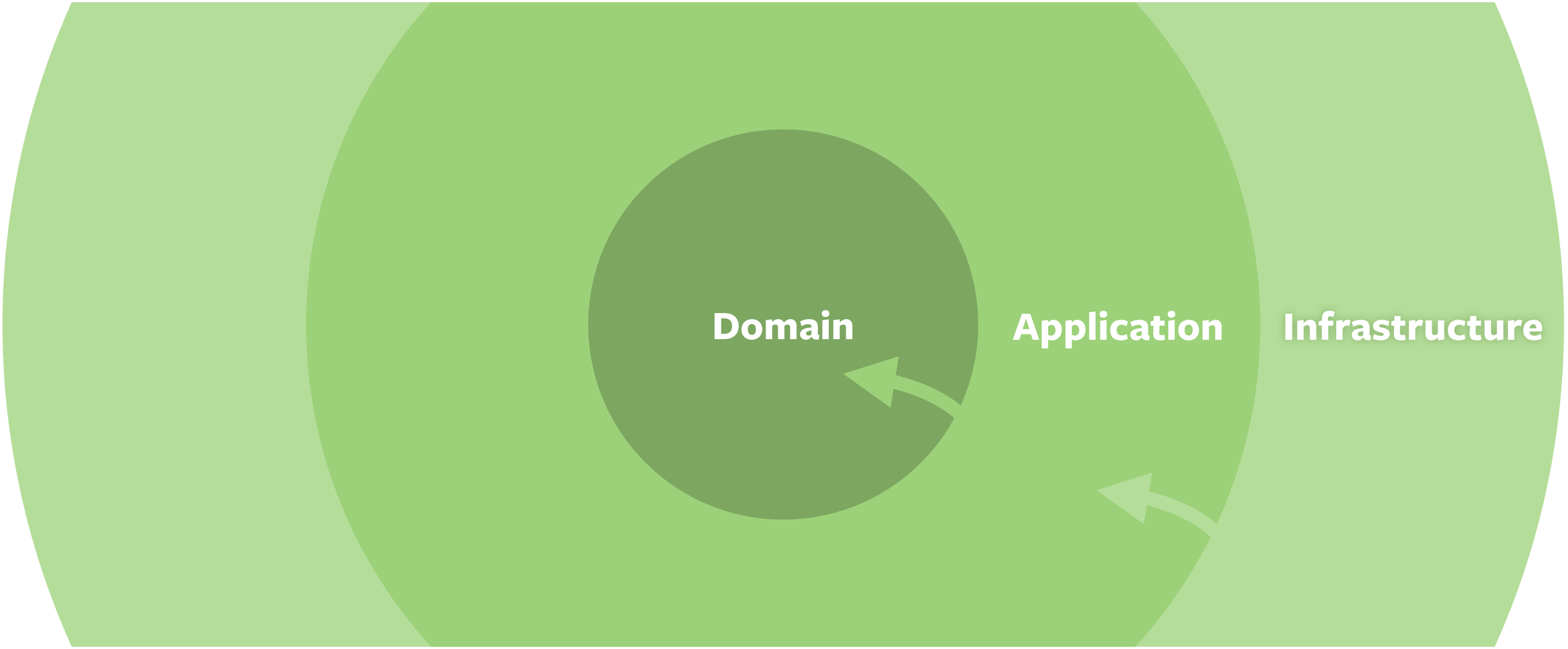
Meanwhile in your IDE…

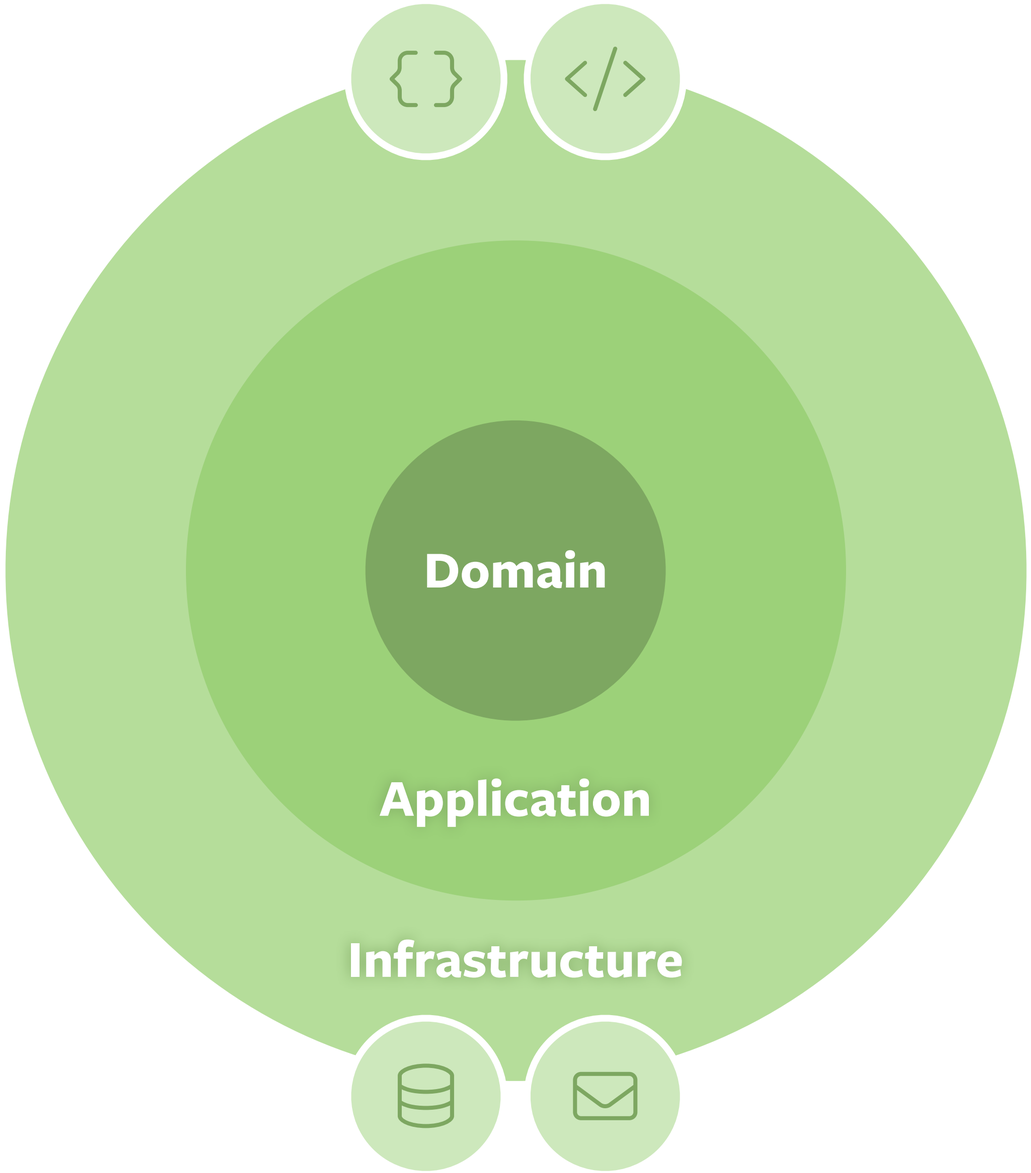
```
[INFO] ◻— example.order.Order
[INFO] ├— JPA - Adding @j.p.Entity.
[INFO] ├— JPA - Adding default constructor.
[INFO] ├— JPA - Adding nullability verification using new callback methods.
[INFO] ├— JPA - Defaulting id mapping to @j.p.EmbeddedId().
[INFO] ├— JPA - Defaulting lineItems mapping to @j.p.JoinColumn(…).
[INFO] ├— JPA - Defaulting lineItems mapping to @j.p.OneToMany(…).
[INFO] ├— Spring Data JPA - Implementing o.s.d.d.Persistable<e.o.Order$OrderIdentifier>.
[INFO] └— Spring JPA - customer - Adding @j.p.Convert(converter=…).
```

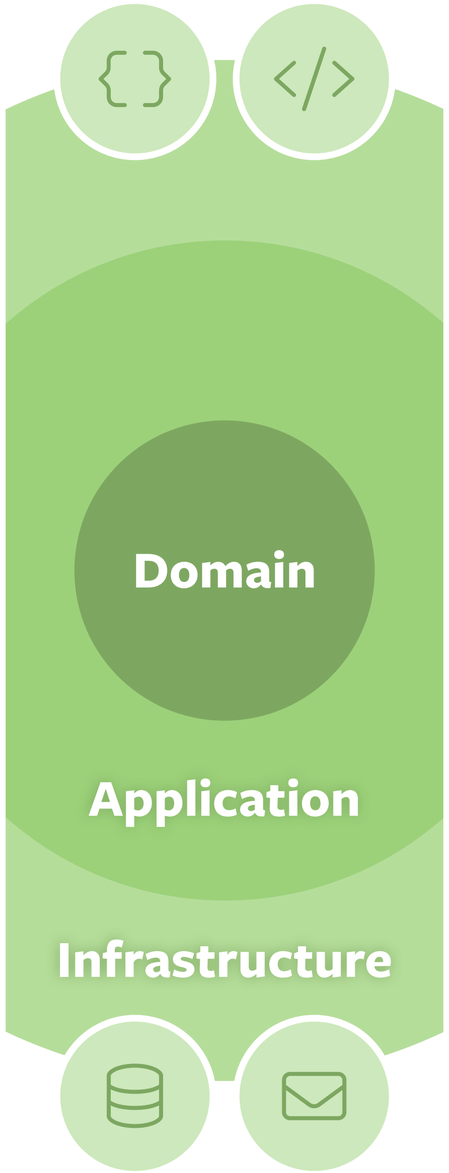
```
@Entity
@NoArgsConstructor(force = true)
@EqualsAndHashCode(of = "id")
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order {

  private final @EmbeddedId OrderId id;

  @OneToMany(cascade = CascadeType.ALL)
  private List<LineItem> lineItems;
  private CustomerId customerId;

  public Order(Customer customer) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customerId = customer.getId();
  }

  @Value
  @RequiredArgsConstructor(staticName = "of")
  @NoArgsConstructor(force = true)
  public static class OrderId implements Serializable {
    private static final long serialVersionUID = …;
    private final UUID orderId;
  }
}
```

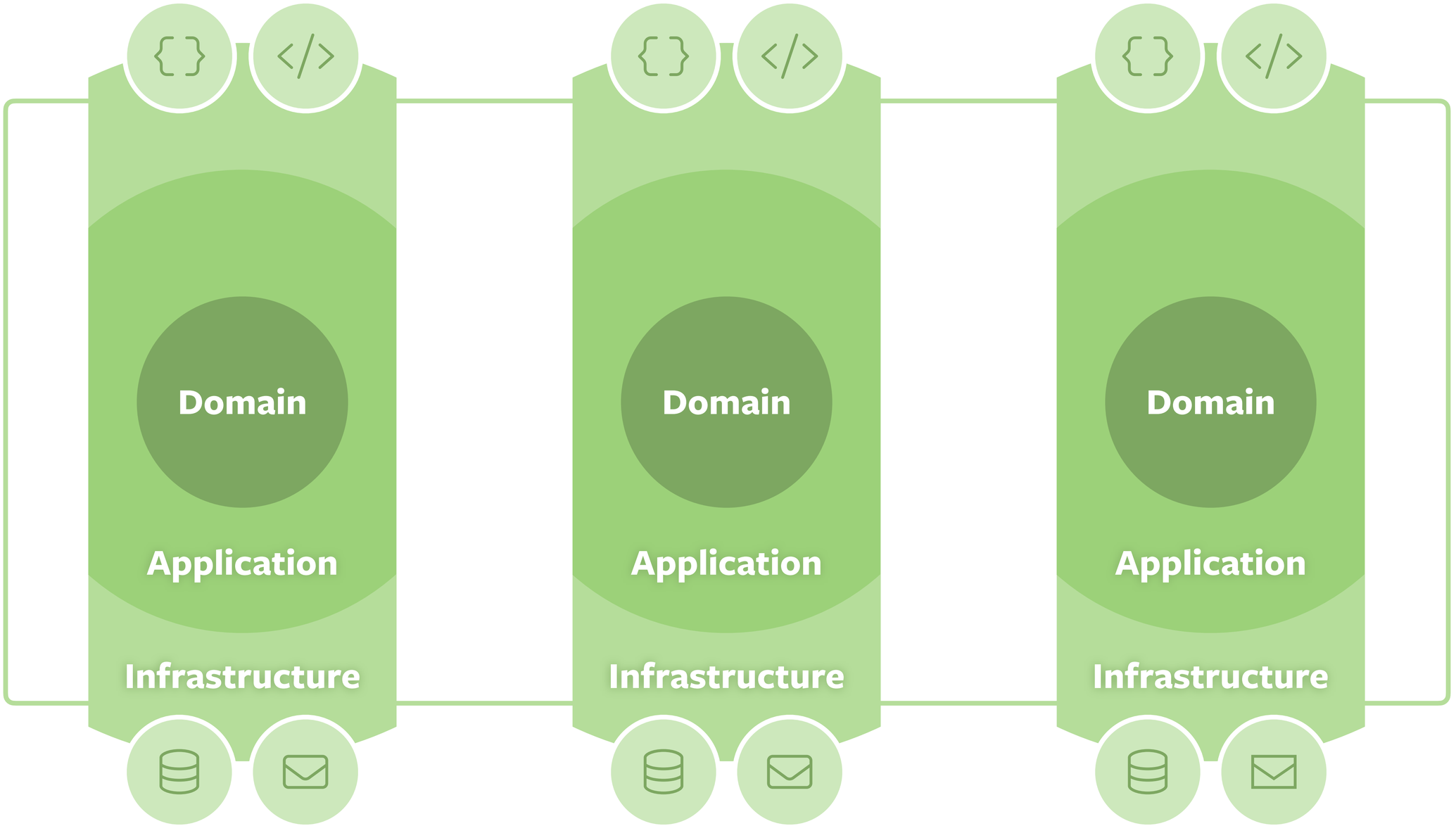
```
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  private final OrderId id;
  private List<LineItem> lineItems;
  private Association<Customer, CustomerId> customer;

  public Order(CustomerId customerId) {
    this.id = OrderId.of(UUID.randomUUID());
    this.customer = Association.forId(customerId);
  }

  @Value(staticConstructor = "of")
  public static class OrderId implements Identifier {
    private final UUID orderId;
  }
}
```

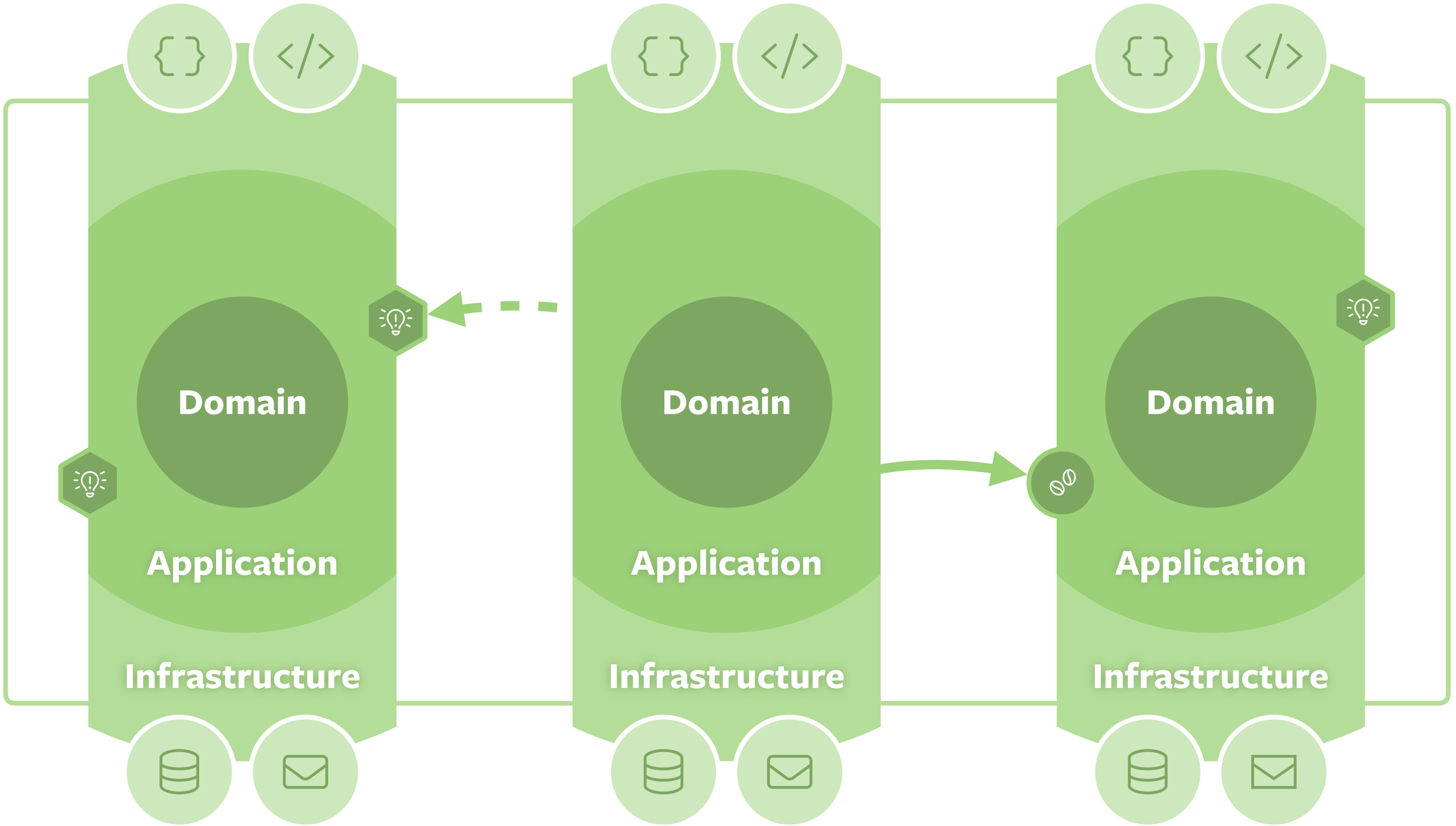
Decomposition

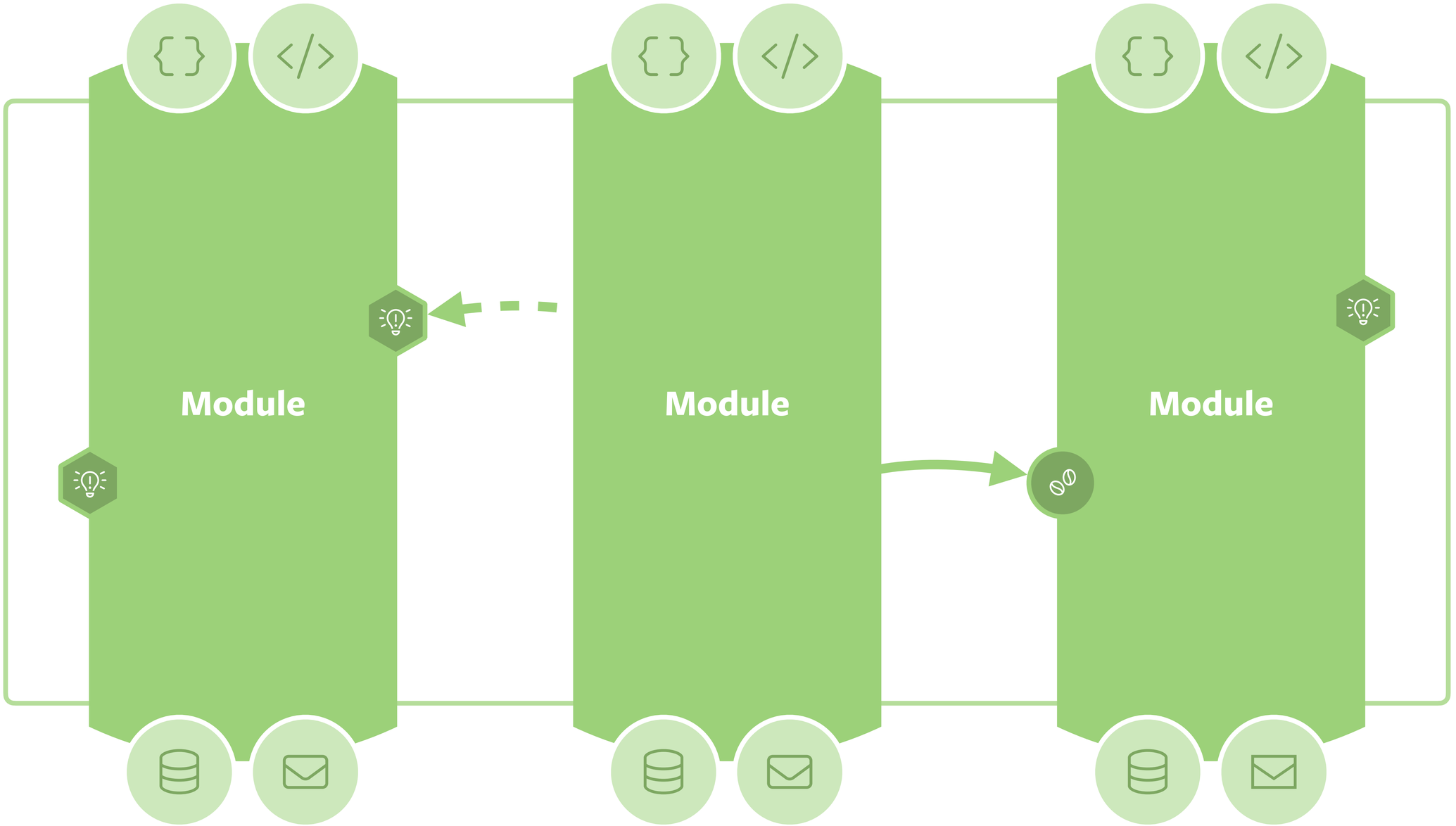

Domain
Application
Infrastructure

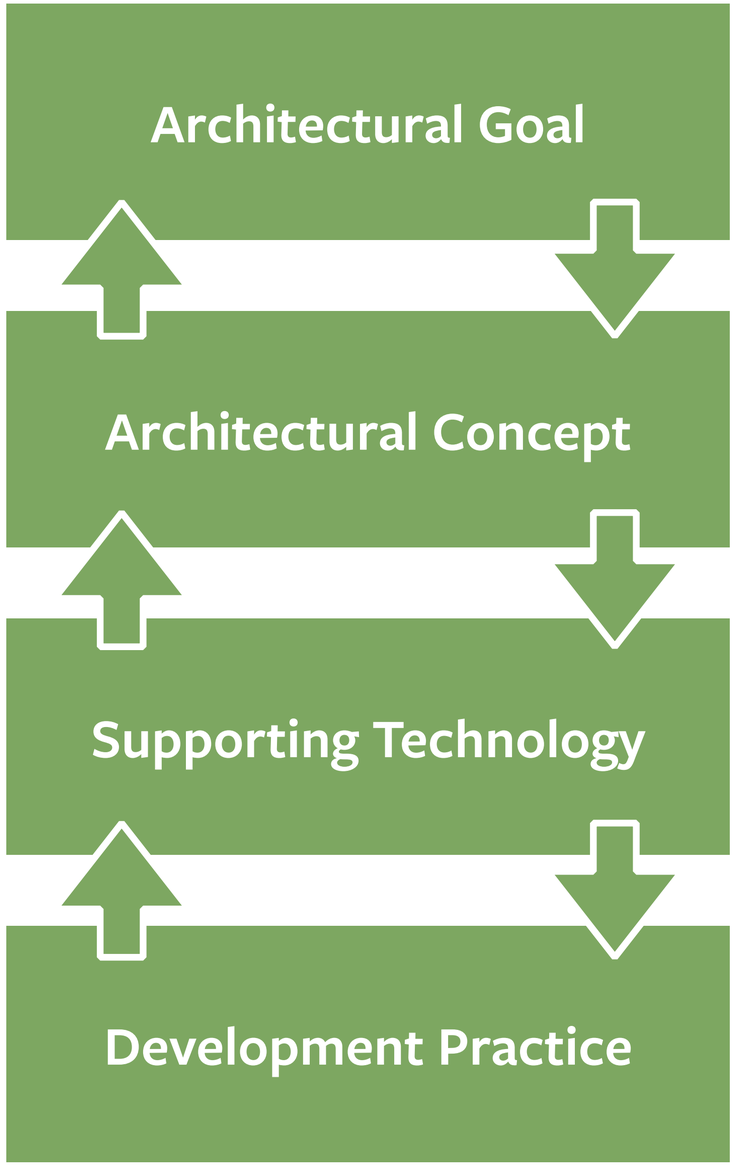

Domain
Application
Infrastructure


Domain
Application
Infrastructure


Domain
Application
Infrastructure
Domain
Application
Infrastructure
Domain
Application
Infrastructure


Domain
Application
Infrastructure
Domain
Application
Infrastructure
Domain
Application
Infrastructure


Module
Module
Module



Evolvability

Modules

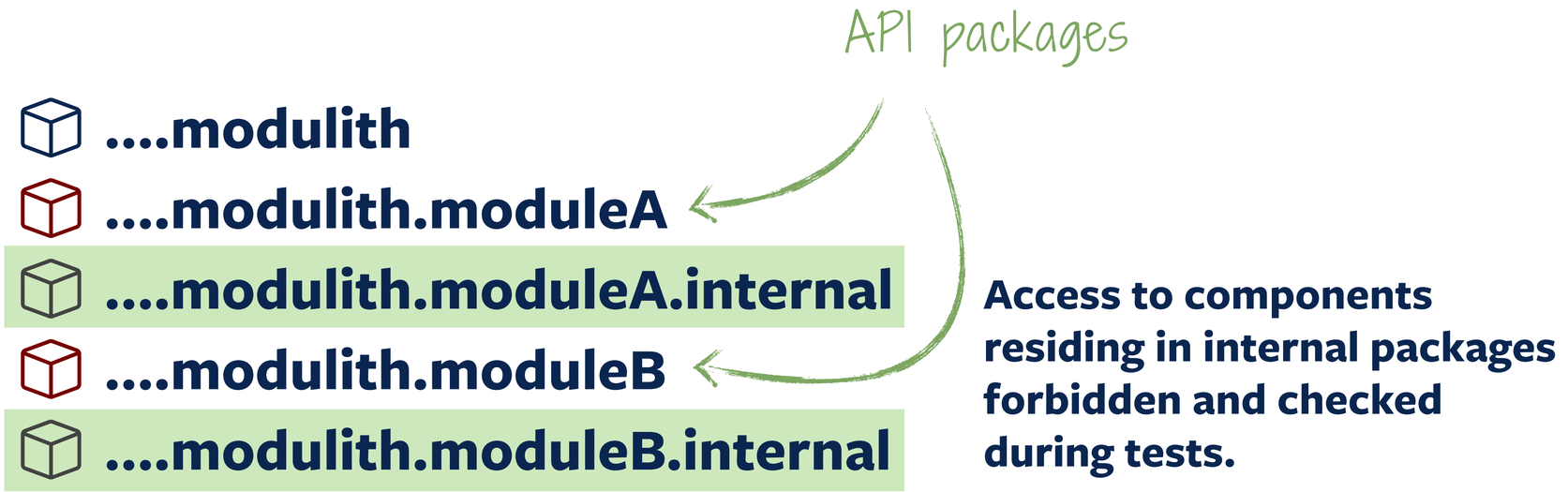
Spring Modulith

Spring Modulith

```
package com.acme.modulith

@SpringBootApplication
class MyApplication { … }
```

Standard Spring Boot Application

Package Conventions



```
package com.acme.modulith

@SpringBootApplication
class MyApplication { … }
```

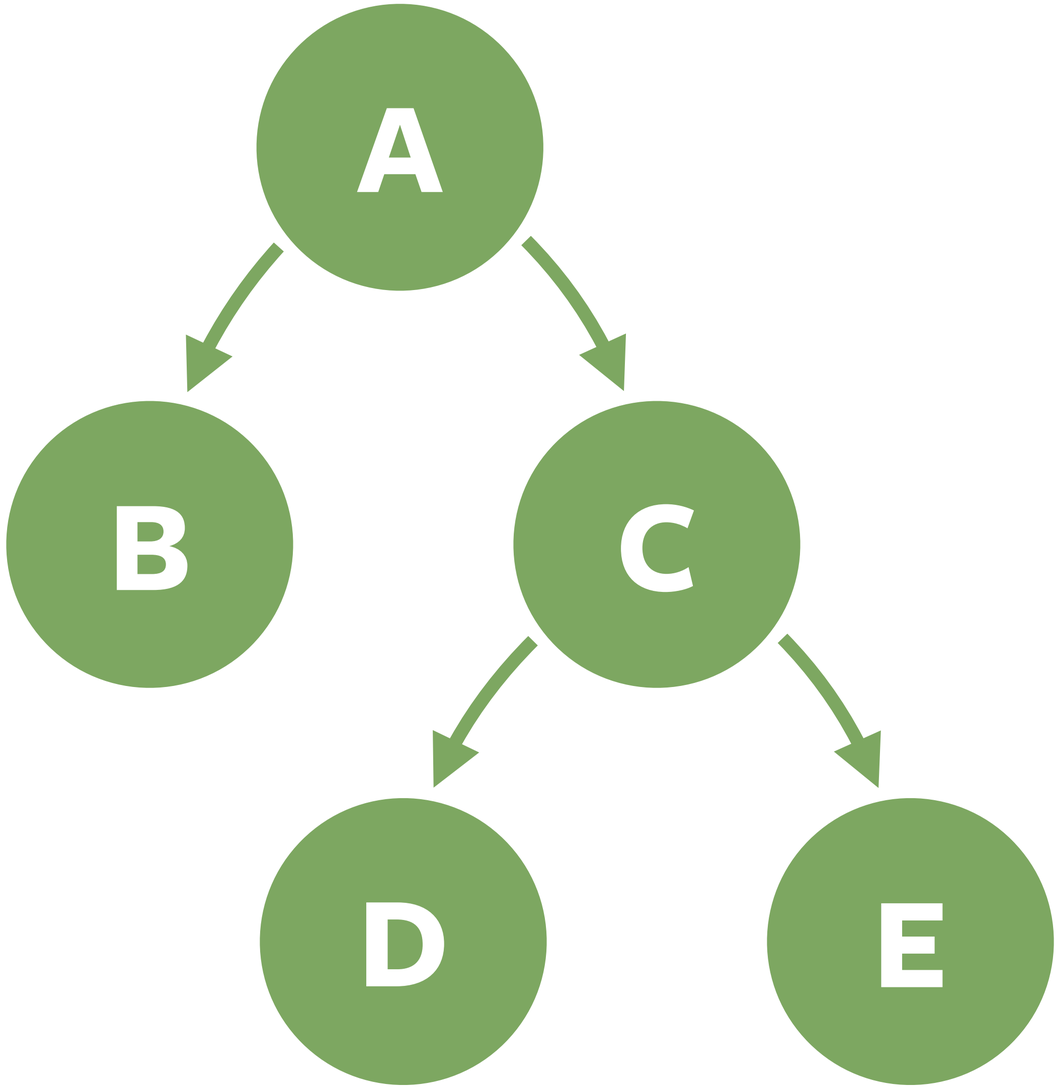
Standard Spring Boot Application

```
var modules =
  ApplicationModules.of(MyApplication.class);
modules.verify(…);
```

Verifies rules for MyApplication

	Module A	Module B	Module C
Web			
Business logic			
Data access			

	Module A	Module B	Module C
Web			
Business logic			
Data access			



Unit of...

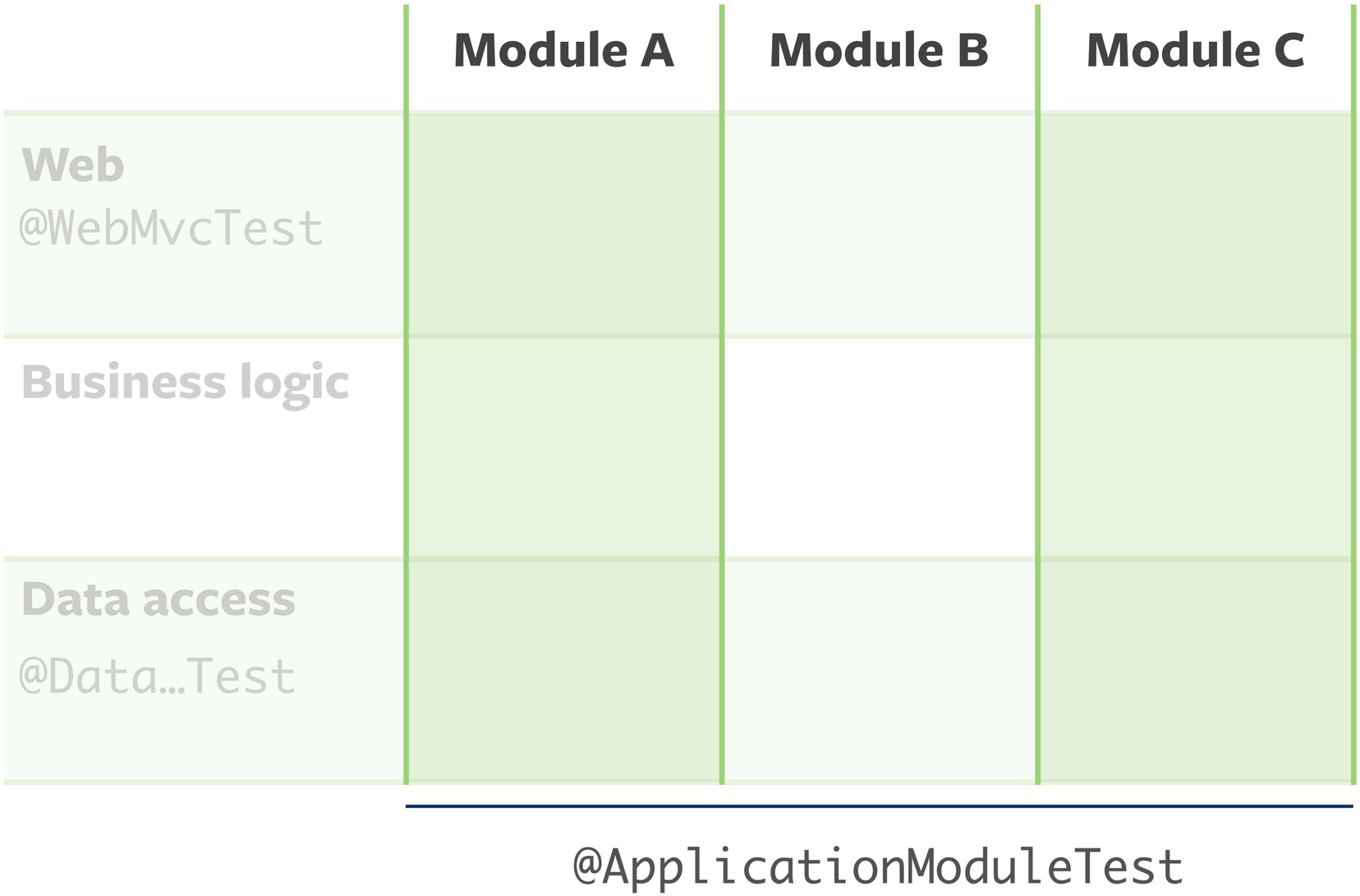
- Understanding
- Consistency
- Testing
- Documentation
- Observation

Testing

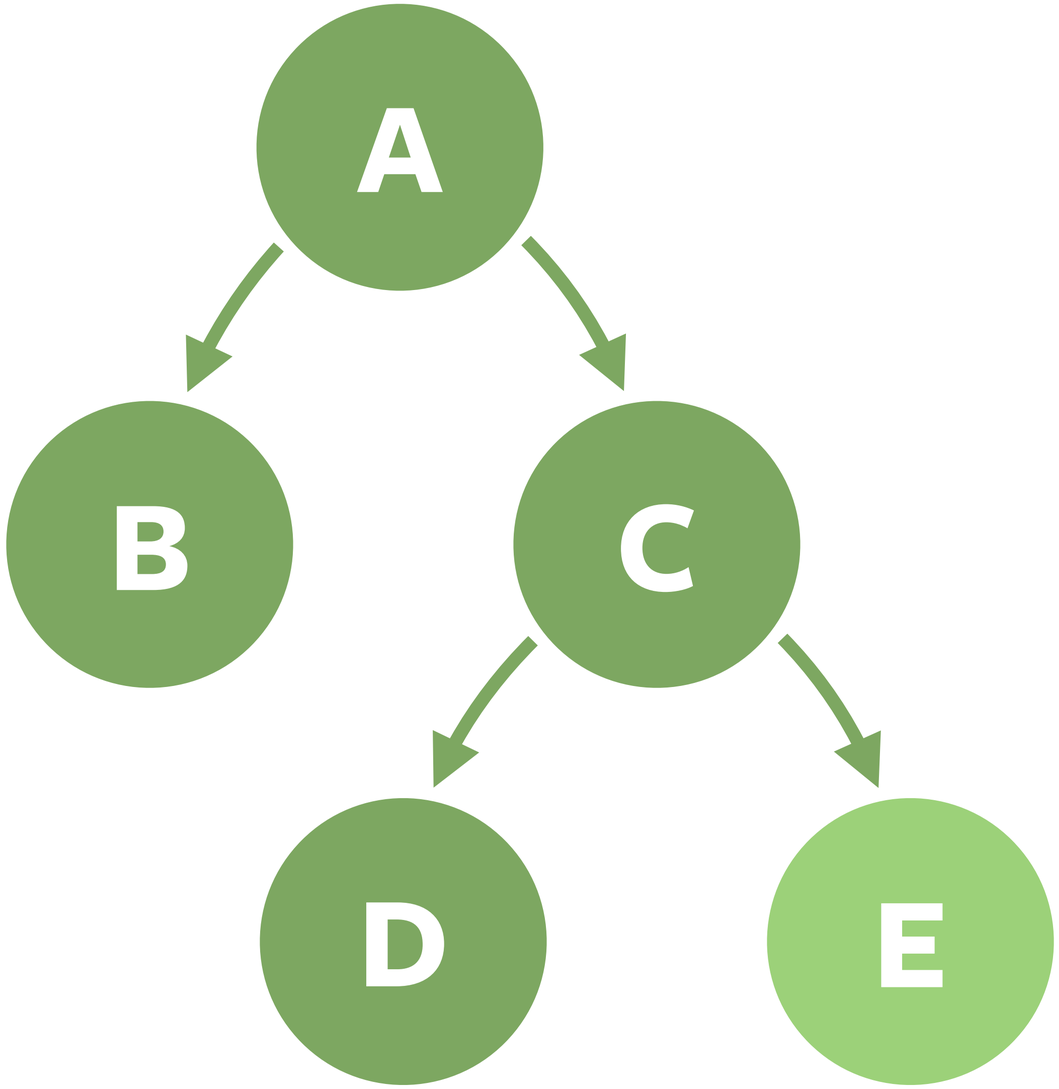
	Module A	Module B	Module C
Web			
Business logic			
Data access			

	Module A	Module B	Module C
Web `@WebMvcTest`			
Business logic			
Data access `@Data…Test`			

	Module A	Module B	Module C
Web `@WebMvcTest`			
Business logic			
Data access `@Data…Test`			

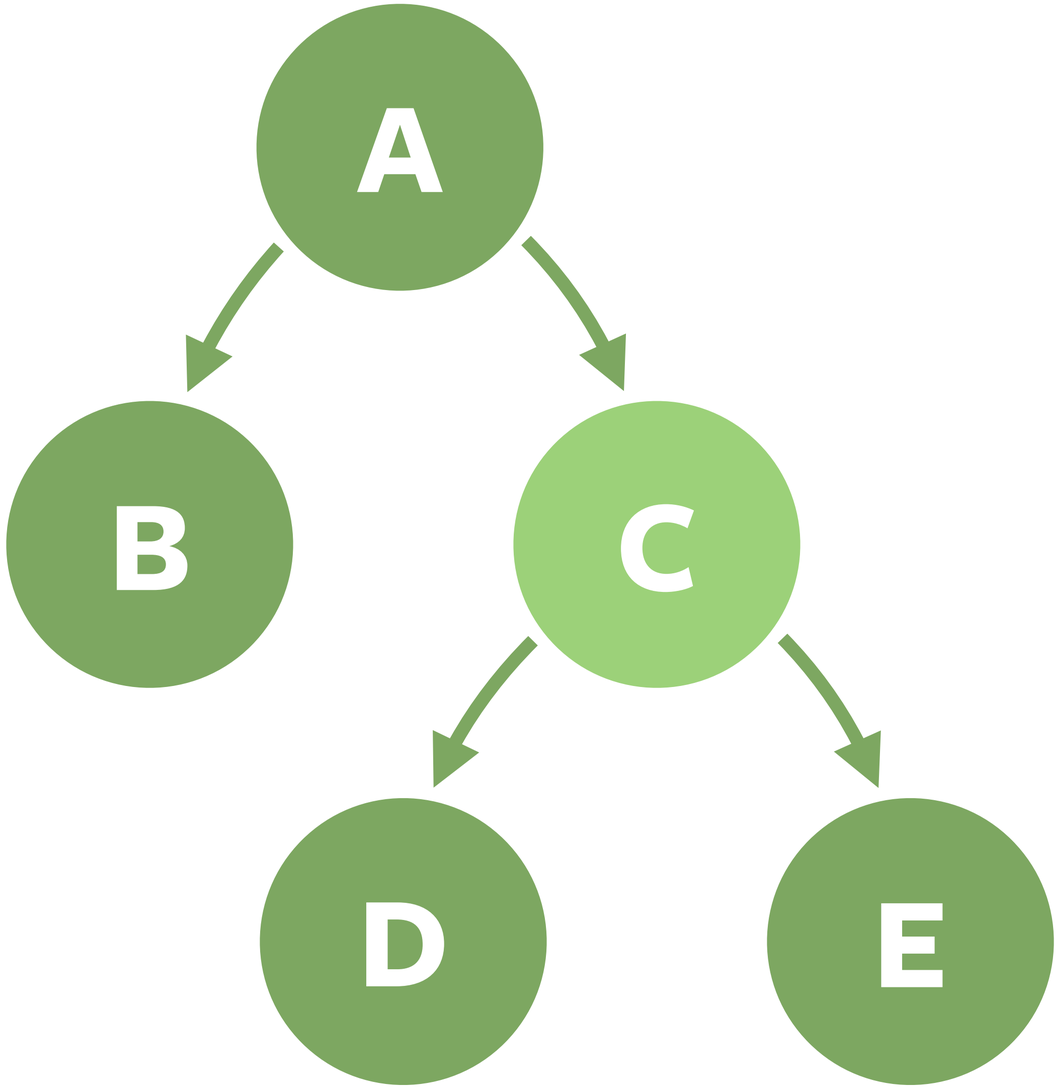

Module A
Module B
Module C
Web
@WebMvcTest
Business logic
Data access
@Data…Test
@ApplicationModuleTest

Scope of Individual Tests



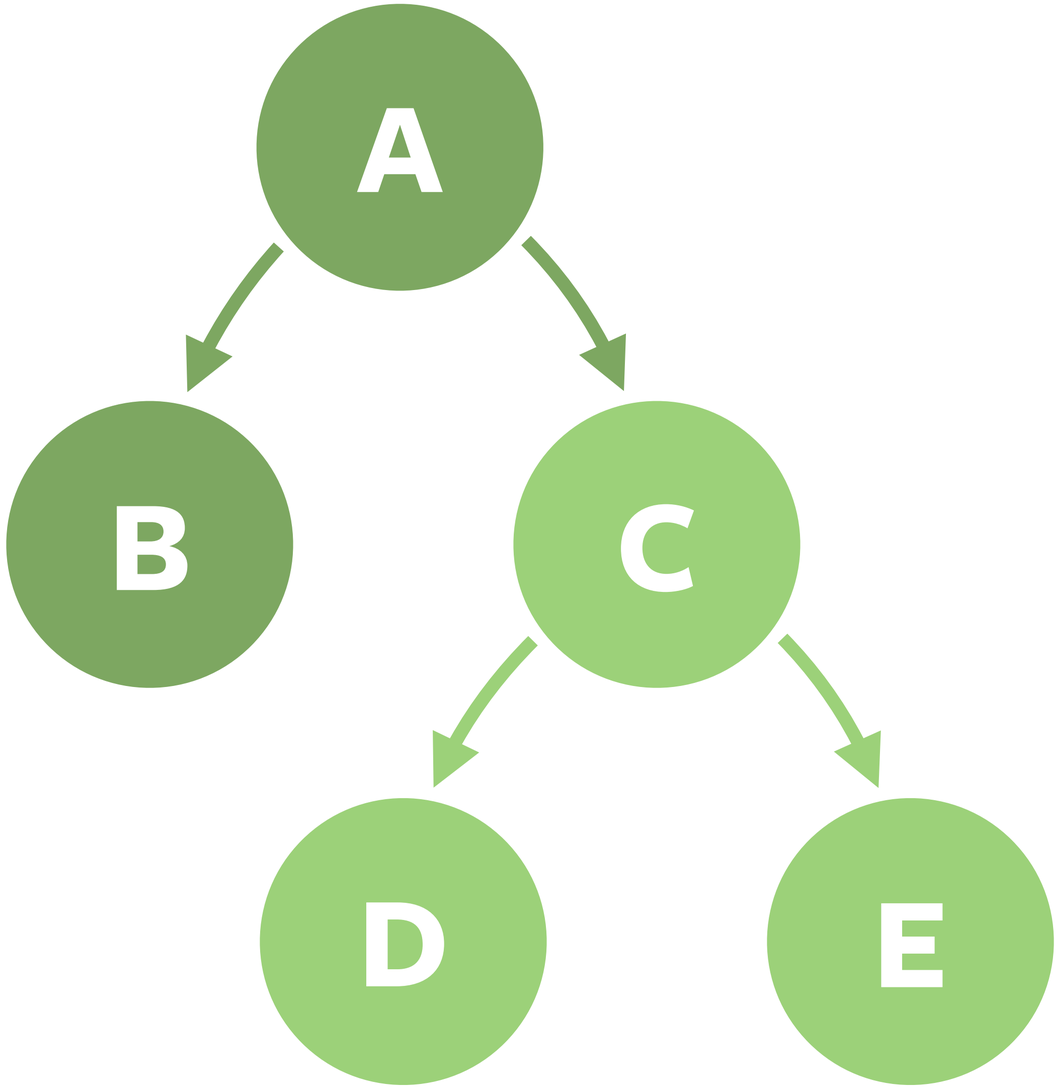
Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation



Unit of...

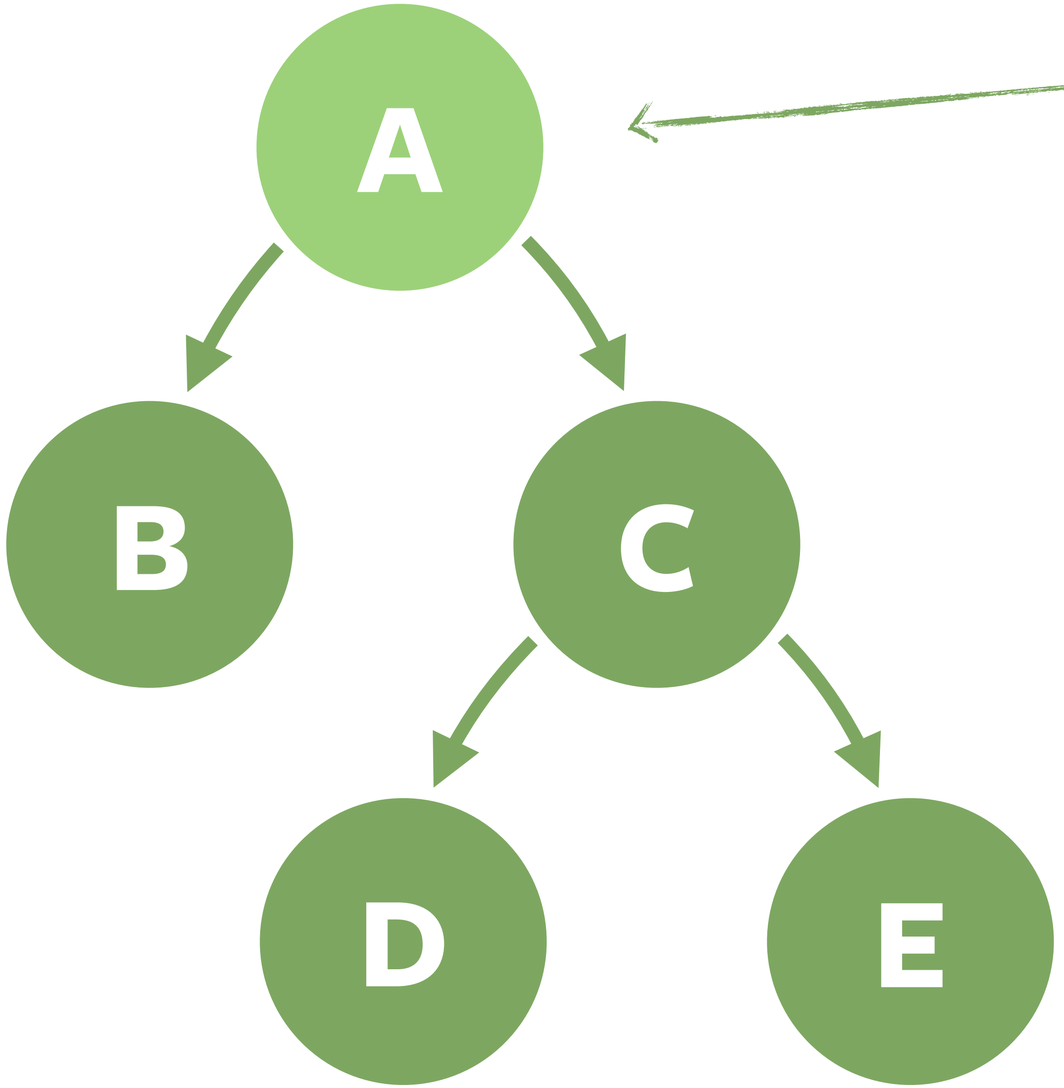
- Understanding
- Consistency
- Testing
- Documentation
- Observation



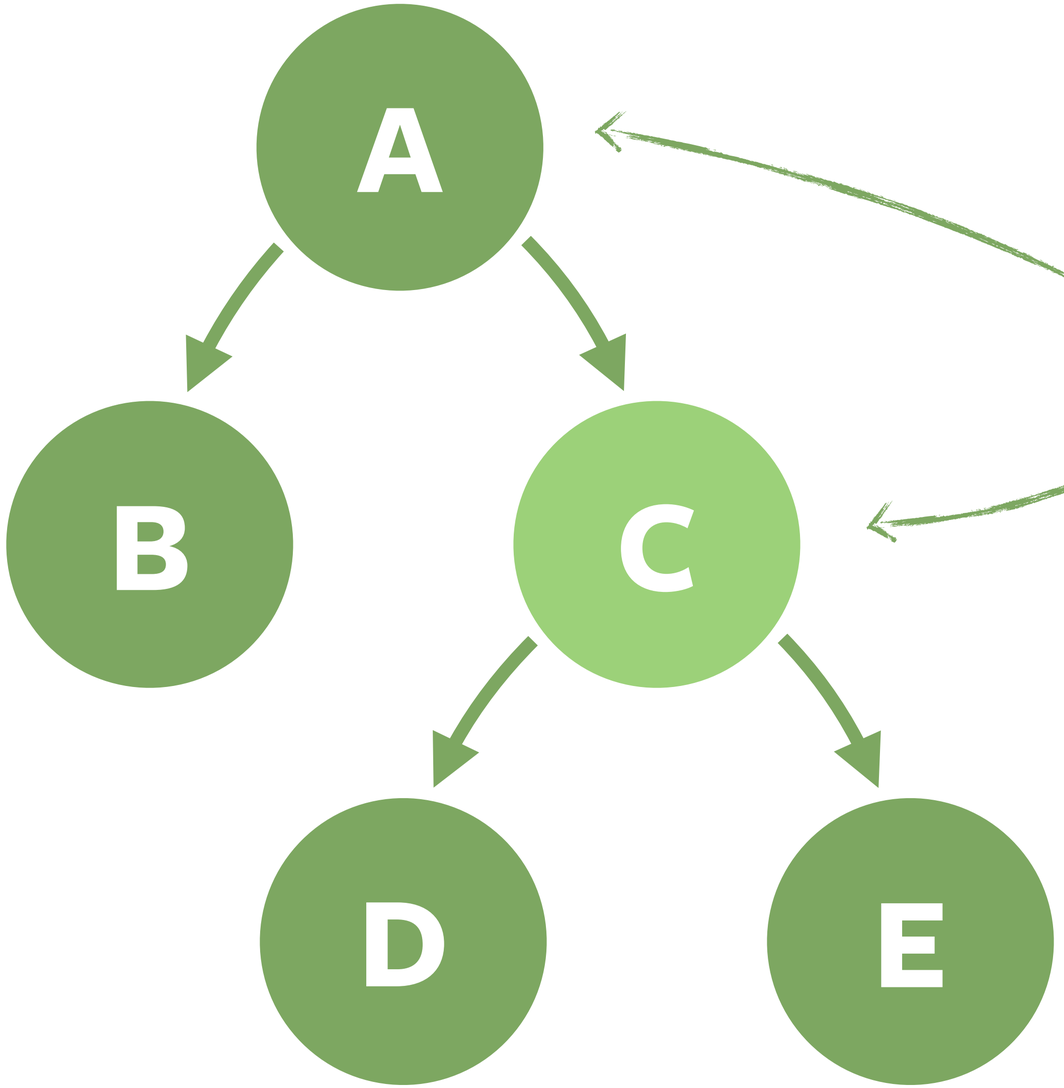
Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation

Scope of Test Execution



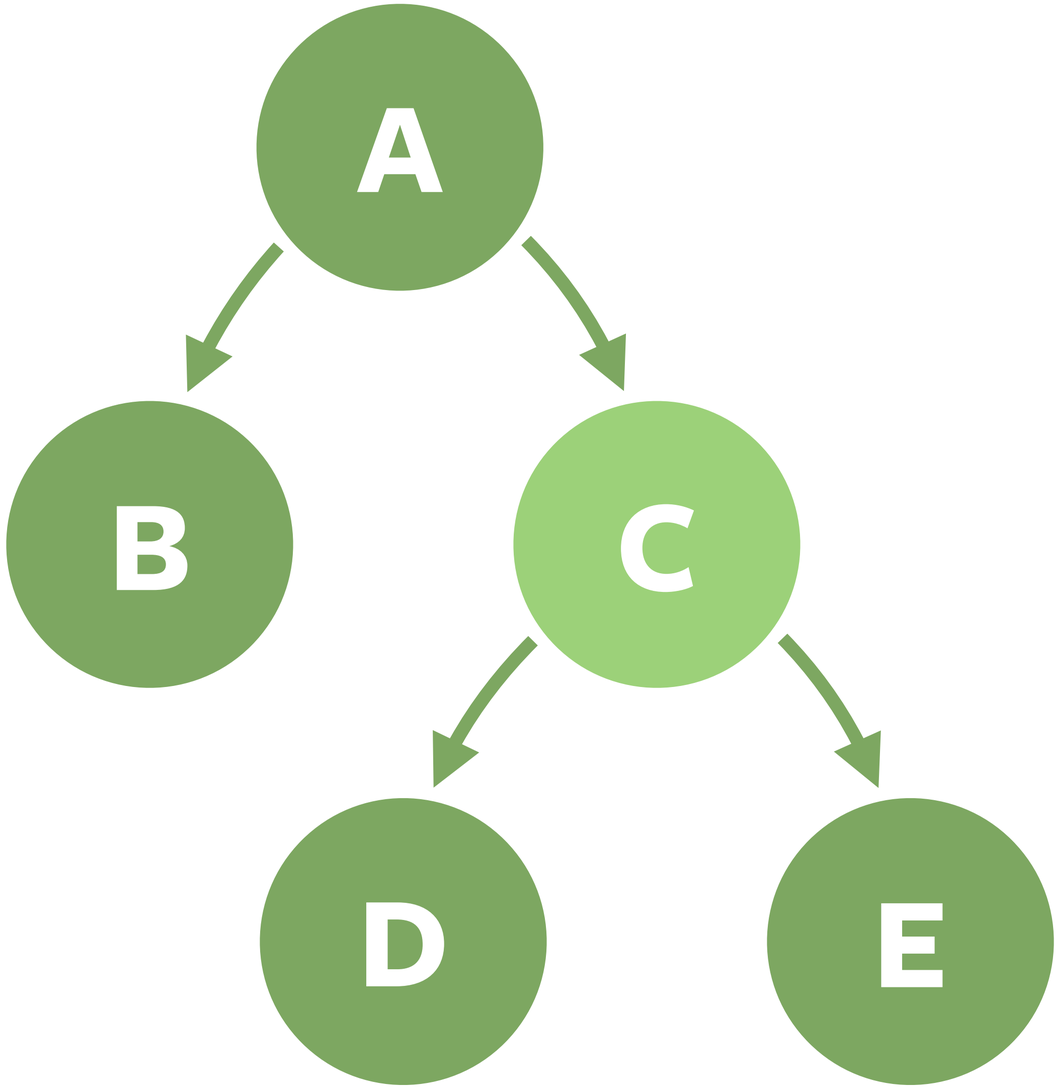
Change detected here.
We only need to test A!



Change detected in C.
We need to test C and A!

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
15:29:09.748 I ----- main : Using default file modification detector (uncommitted and unpushed changes).
15:29:10.054 I ----- main : ☕ example.inventory.Inventory
15:29:10.866 I ----- main : ⏸️  Test residing in module order not affected by changes!
[INFO] Running example.order.OrderIntegrationTests
[WARNING] Tests run: 4, Failures: 0, Errors: 0, Skipped: 4, Time elapsed: 0.002 s -- in example.order.OrderIntegrationTests
15:29:10.871 I ----- main : ⏸️  Test residing in module order not affected by changes!
[INFO] Running example.order.EventPublicationRegistryTests
[WARNING] Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed: 0 s -- in example.order.EventPublicationRegistryTests
15:29:10.873 I ----- main : ▶️  Always executing tests in root modules.
[INFO] Running example.ApplicationTests
…
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.344 s -- in example.ApplicationTests
15:29:13.235 I ----- main : ▶️  Changes detected in module inventory, executing test.
[INFO] Running example.inventory.InventoryIntegrationTests
…
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.267 s -- in example.inventory.InventoryIntegrationTests
15:29:13.504 I ----- main : ▶️  Always executing tests in root modules.
[INFO] Running example.ModularityTests
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.084 s -- in example.ModularityTests
…
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.458 s
[INFO] Finished at: 2024-09-08T15:29:13+02:00
[INFO] ------------------------------------------------------------------------
```

Documentation



Unit of...

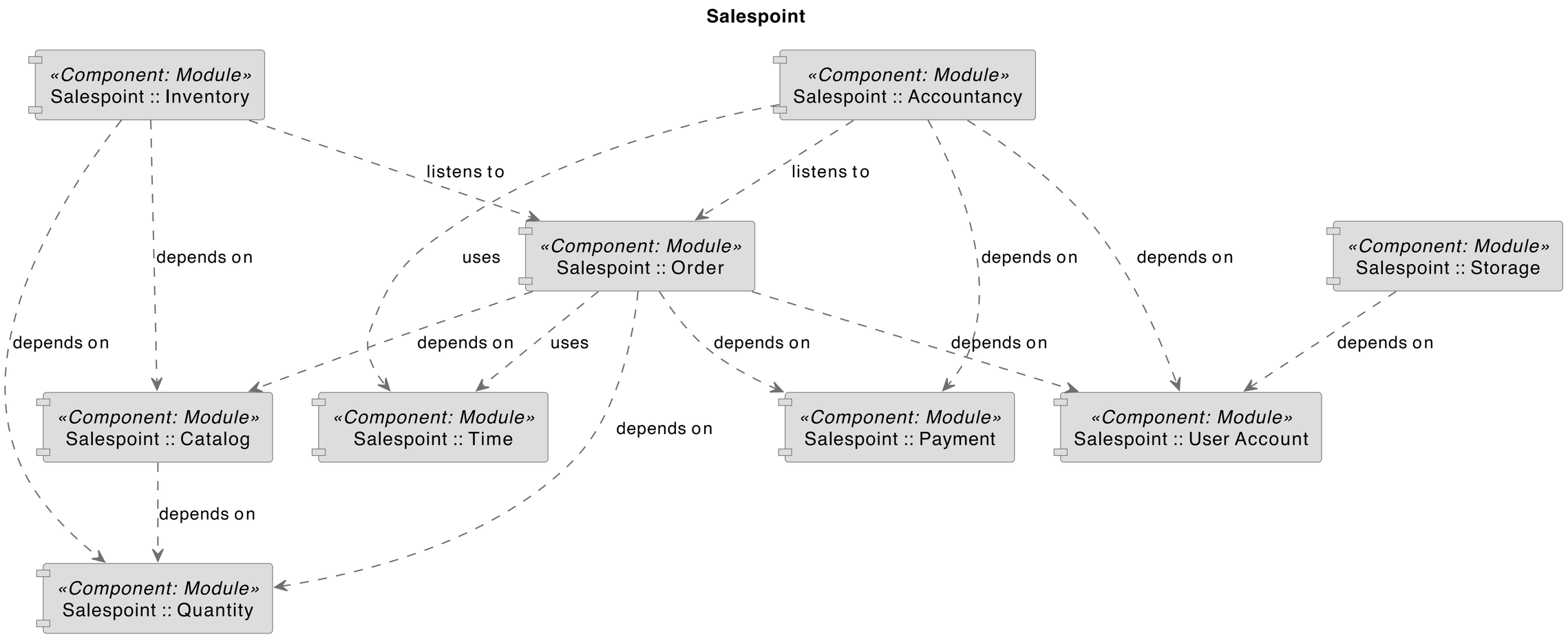
- Understanding
- Consistency
- Testing
- Documentation
- Observation

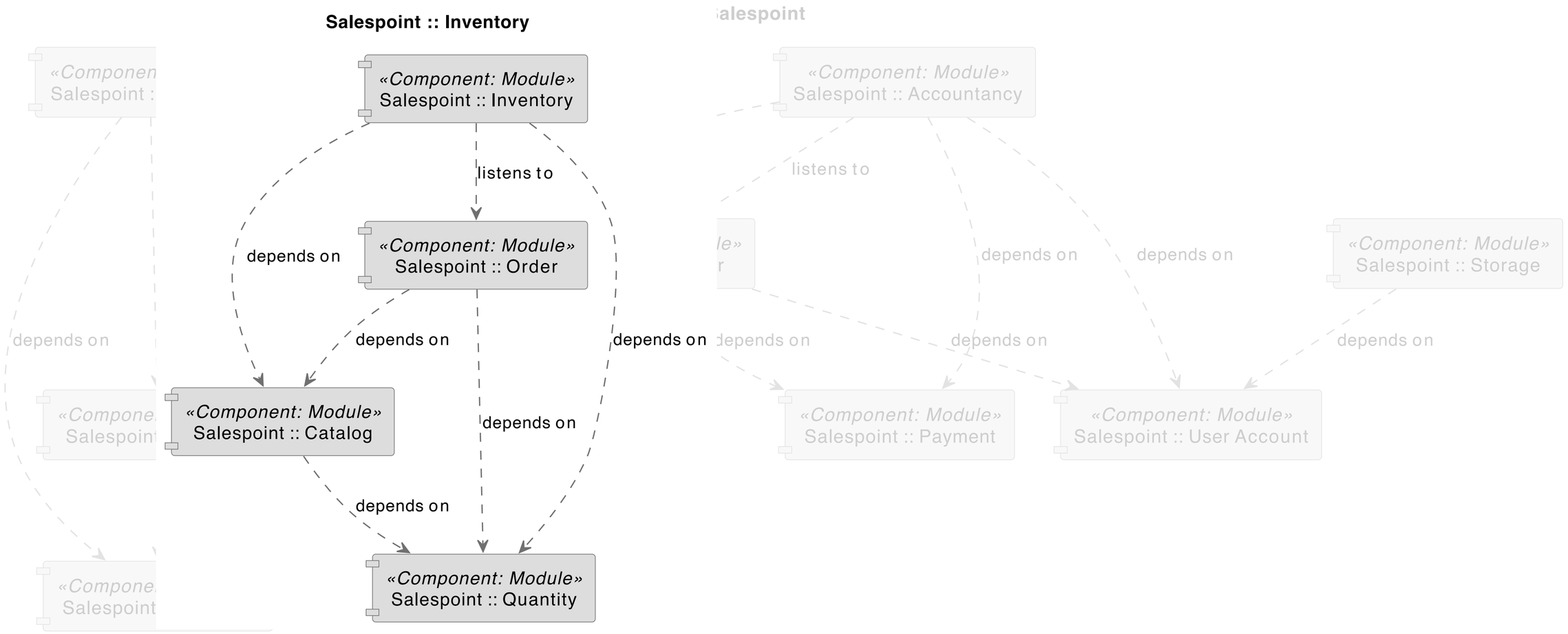
Provided Interface

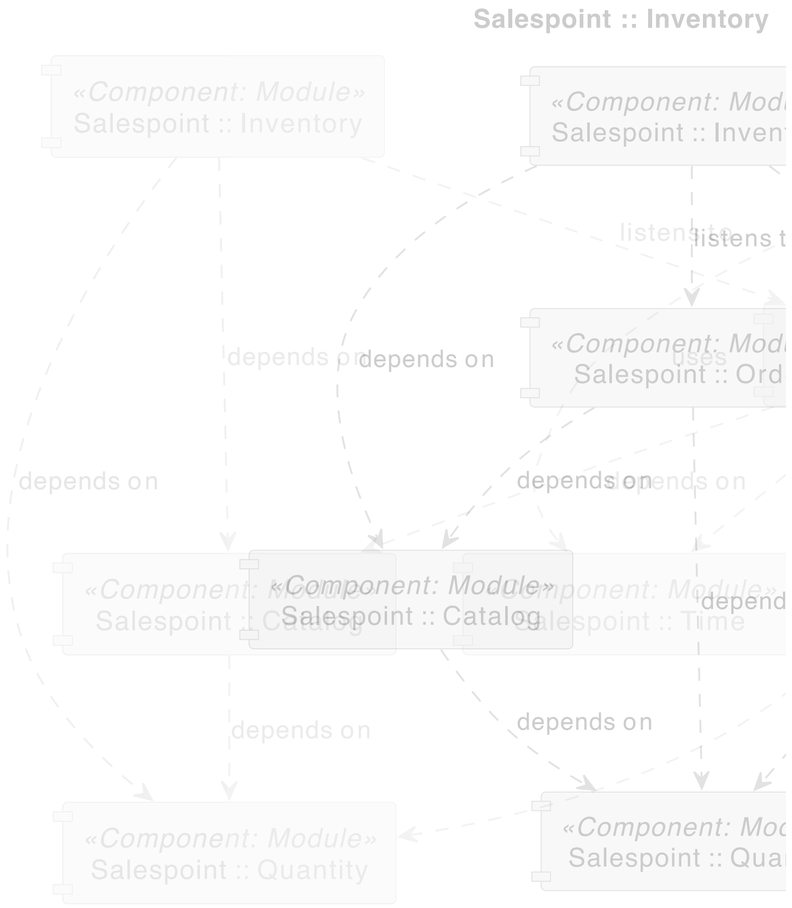
- **Exposed Service API**
  Spring Beans available for DI
- **Exposed Aggregates**
  Primary elements of the domain and constraints
- **Published Events**
  Events the component emits

Required Interface

- **Consumed Service API**
  External dependencies of Spring beans
- **Configuration**
  Spring Boot configuration properties
- **Consumed Events**
  Events that the component reacts to


Salespoint
«Component: Module»
Salespoint :: Inventory
«Component: Module»
Salespoint :: Accountancy
«Component: Module»
Salespoint :: Order
«Component: Module»
Salespoint :: Storage
«Component: Module»
Salespoint :: Catalog
«Component: Module»
Salespoint :: Time
«Component: Module»
Salespoint :: Payment
«Component: Module»
Salespoint :: User Account
«Component: Module»
Salespoint :: Quantity
listens to
listens to
depends on
uses
depends on
depends on
depends on
depends on
depends on
uses
depends on
depends on
depends on
depends on
depends on


Salespoint :: Inventory
«Component: Module»
Salespoint :: Inventory
listens to
depends on
«Component: Module»
Salespoint :: Order
depends on
depends on
depends on
depends on
«Component: Module»
Salespoint :: Catalog
depends on
depends on
«Component: Module»
Salespoint :: Quantity
alespoint
«Component: Module»
Salespoint :: Accountancy
listens to
depends on
depends on
depends on
depends on
depends on
«Component: Module»
Salespoint :: Storage
«Component: Module»
Salespoint :: Payment
«Component: Module»
Salespoint :: User Account



Base package	`org.salespointframework.useraccount`
Spring components	*Services* • `o.s.u.UserAccountManagement` (via `o.s.u.PersistentUserAccountManagement`) *Others* • `o.s.u.AuthenticationManagement` (via `o.s.u.SpringSecurityAuthenticationManagement`)
Aggregate roots	• `o.s.u.UserAccount`
Value types	• `o.s.u.EncryptedPassword` • `o.s.u.UnencryptedPassword` • `o.s.u.Role`
Published events	• `o.s.u.UserAccountCreated` created by: ◦ `o.s.u.UserAccount.onCreate()`
Properties	• `salespoint.authentication.login-via-email` — `java.lang.Boolean`, default `false`. Enables the login procedure to use the email address to lookup a user instead of their username. Defaults to `false`.

Summary

Summary

- Find means to represent architectural and design concepts in your codebase.
- Align software engineering practices with architectural abstractions.
- Favor architecturally aware technologies over allegedly agnostic ones.

Summary

- Find means to represent architectural and design concepts in your codebase.
- Align software engineering practices with architectural abstractions.
- Favor architecturally aware technologies over allegedly agnostic ones.
- Understand how technology choices affect the overall architecture of the system.

Thank you!
Questions?

Oliver Drotbohm odrotbohm oliver.drotbohm@broadcom.com

Links

- **xMolecules**
  https://xmolecules.org
- **jMolecules**
  https://jmolecules.org
- **jMolecules Examples**
  https://github.com/xmolecules/jmolecules-examples
- **Gitter – Join the community!**
  https://gitter.im/xmolecules/xmolecules

Resources

- **Software Architecture for Developers**
  Simon Brown – Books
- **Just Enough Software Architecture**
  George Fairbanks – Book
- **Architecture, Design, Implementation**
  Ammon H. Eden, Rick Kazman – Paper
- **Sustainable Software Architecture**
  Carola Lilienthal – Book
- **The Programmer's Brain**
  Felienne Hermans – Book