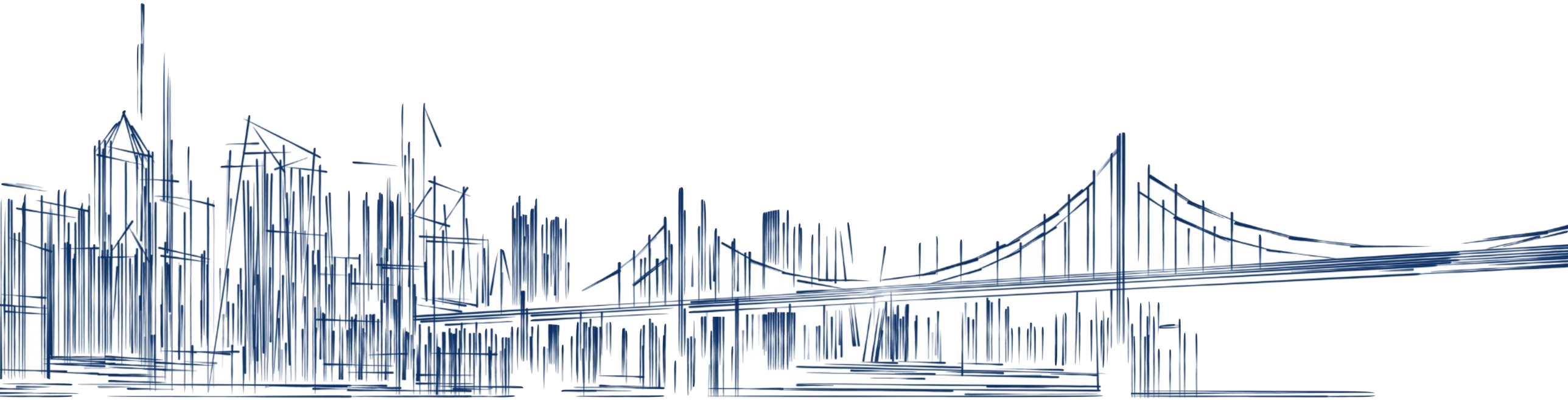


BOTTOM-UP ARCHITECTURE

BRIDGING THE ARCHITECTURE CODE GAP

Oliver Drotbohm odrotbohm oliver.drotbohm@broadcom.com

github.com/odrotbohm

odrotbohm

Type / to search

Overview | Repositories 126 | Projects | Packages | Stars 78

Oliver Drotbohm

odrotbohm · he/him

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

Edit profile

3.7k followers · **32** following

- VMware by Broadcom, Inc.
- Dresden, Germany
- 11:55 (UTC +02:00)
- info@odrotbohm.de
- www.odrotbohm.de
- @odrotbohm
- @odrotbohm@chaos.social
- odrotbohm
- in/odrotbohm

Pinned

Customize your pins

spring-projects/spring-modulith Public
Modular applications with Spring Boot
Java ☆ 797 137

xmolecules/jmolecules Public
Libraries to help developers express architectural abstractions in Java code
Java ☆ 1.2k 103

xmolecules/jmolecules-integrations Public
Technology integration for jMolecules
Java ☆ 85 23

spring-restbucks Public
Implementation of the sample from REST in Practice based on Spring projects
Java ☆ 1.2k 414

spring-playground Public
A collection of tiny helpers for building Spring applications
Java ☆ 100 11

lectures Public
Lecture scripts and slides I use during the Software Engineering course at TU Dresden
Java ☆ 72 25

1,982 contributions in the last year

Contribution settings

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

Mon
Wed
Fri

Learn how we count contributions

Less More

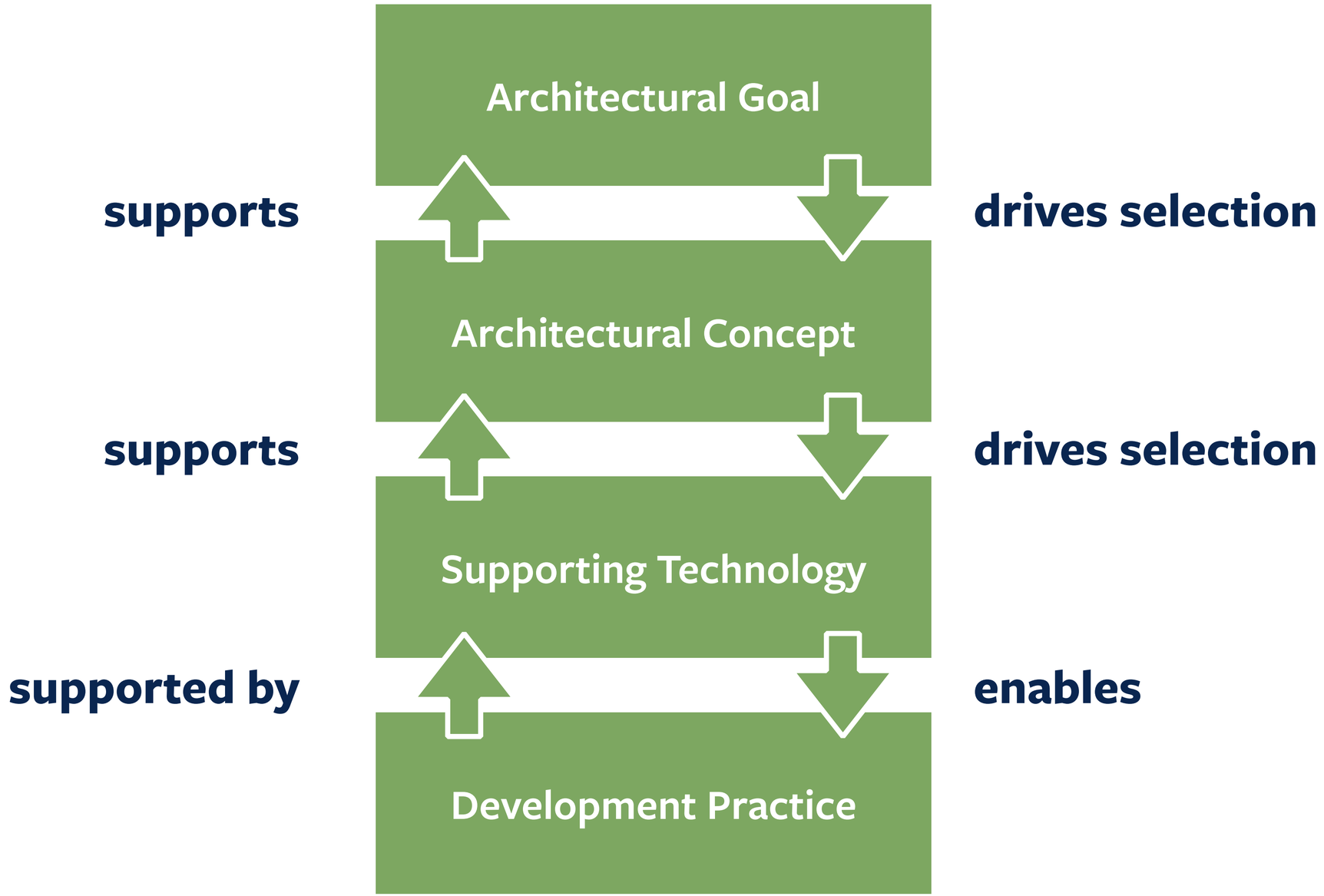
2024
2023
2022
2021
2020
2019
2018
2017

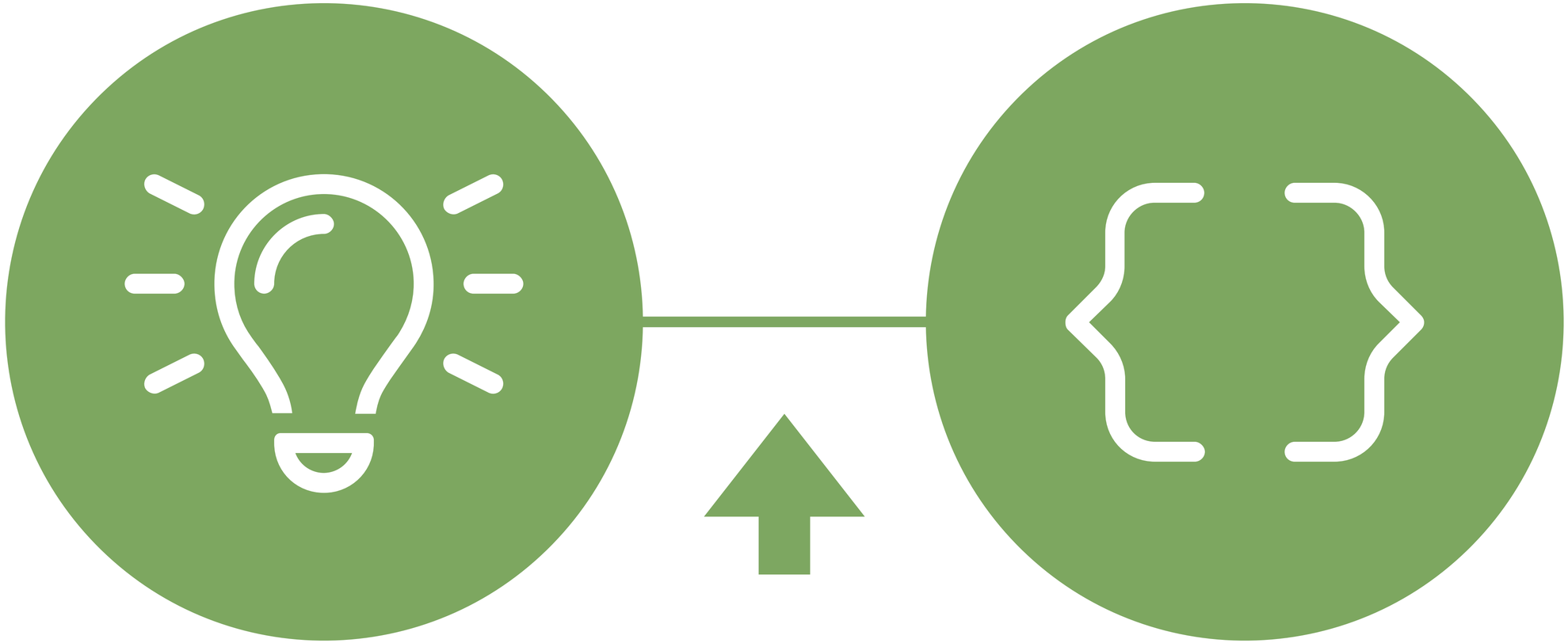
@spring-projects | @st-tu-dresden-prakt... | @xmolecules | More

Activity overview

Contributed to spring-projects/spring-modulith,

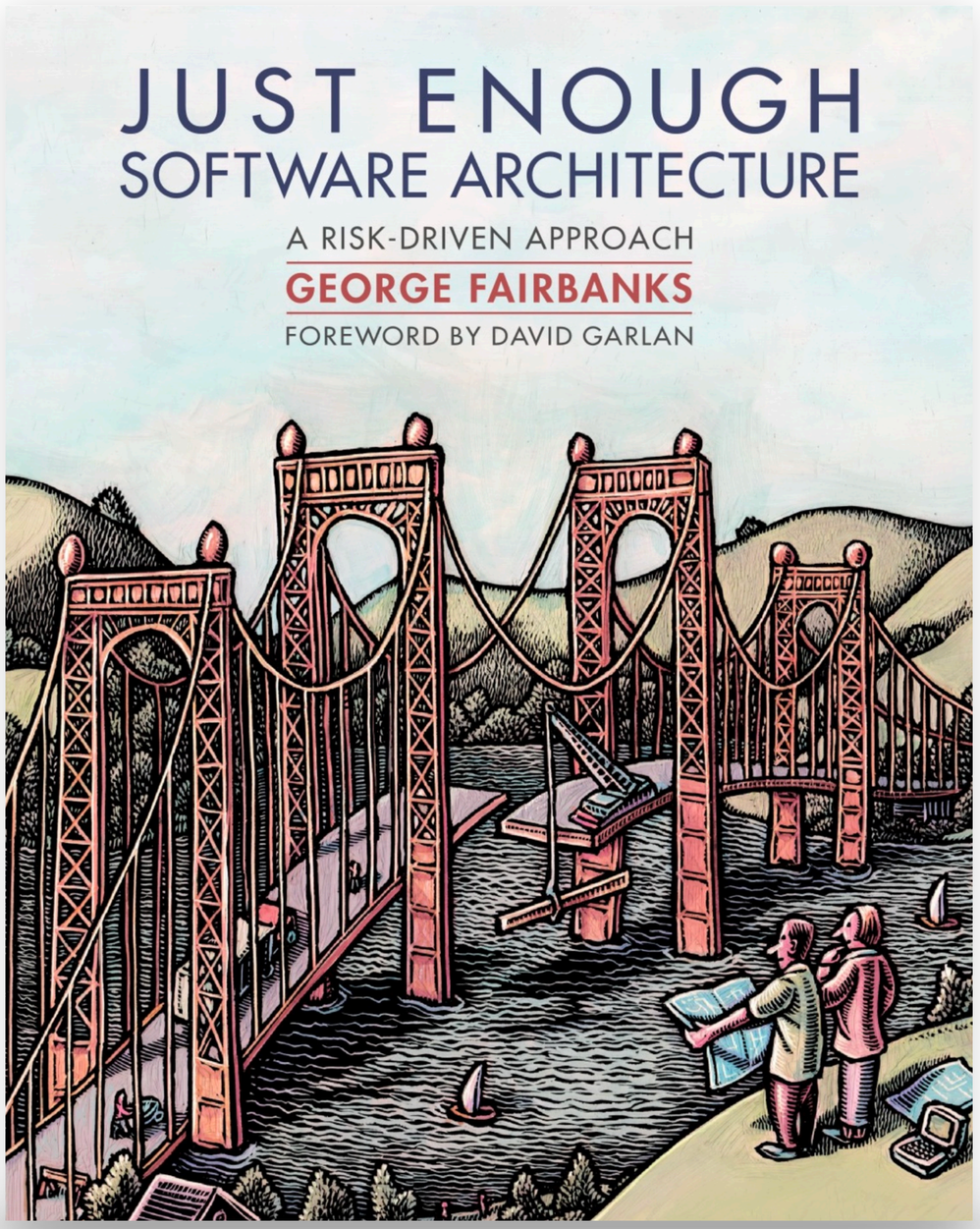
Code review

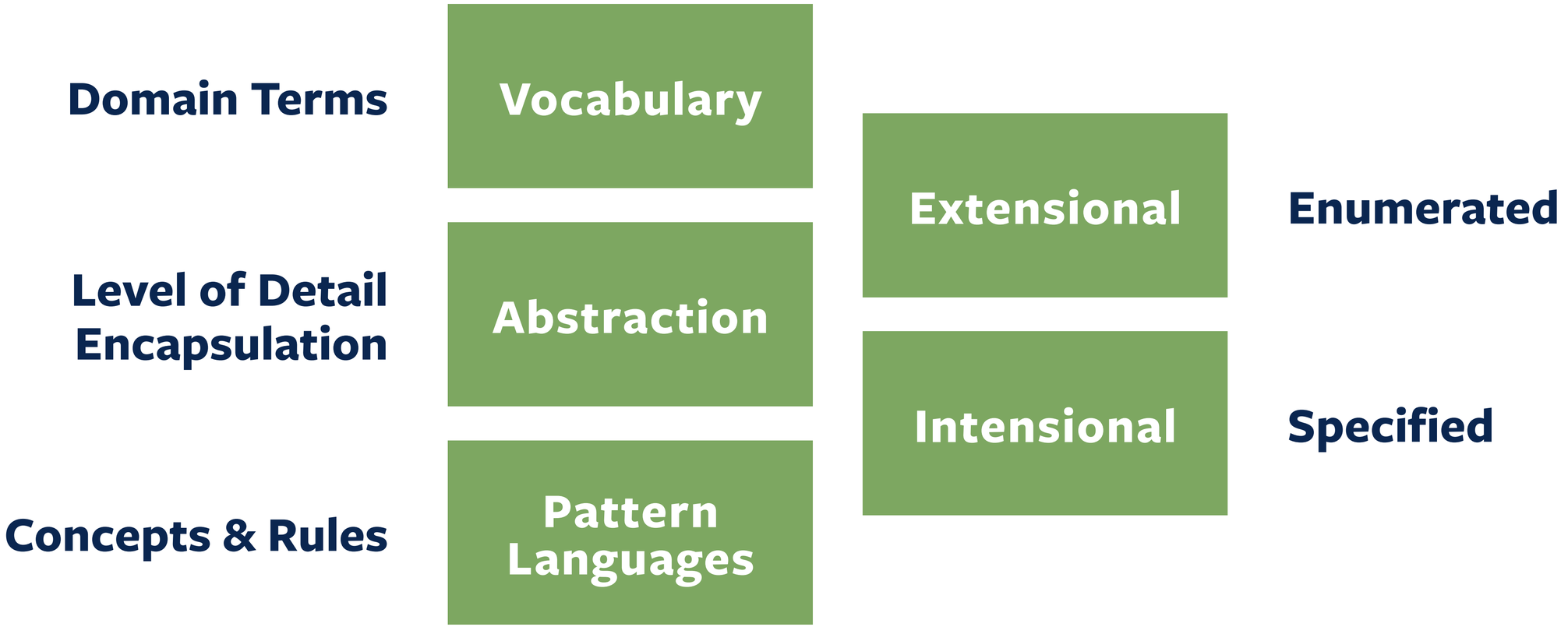

Architectural Goal
supports
drives selection
Architectural Concept
supports
drives selection
Supporting Technology
supported by
enables
Development Practice



"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


Domain Terms
Level of Detail
Encapsulation
Concepts & Rules
Vocabulary
Abstraction
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

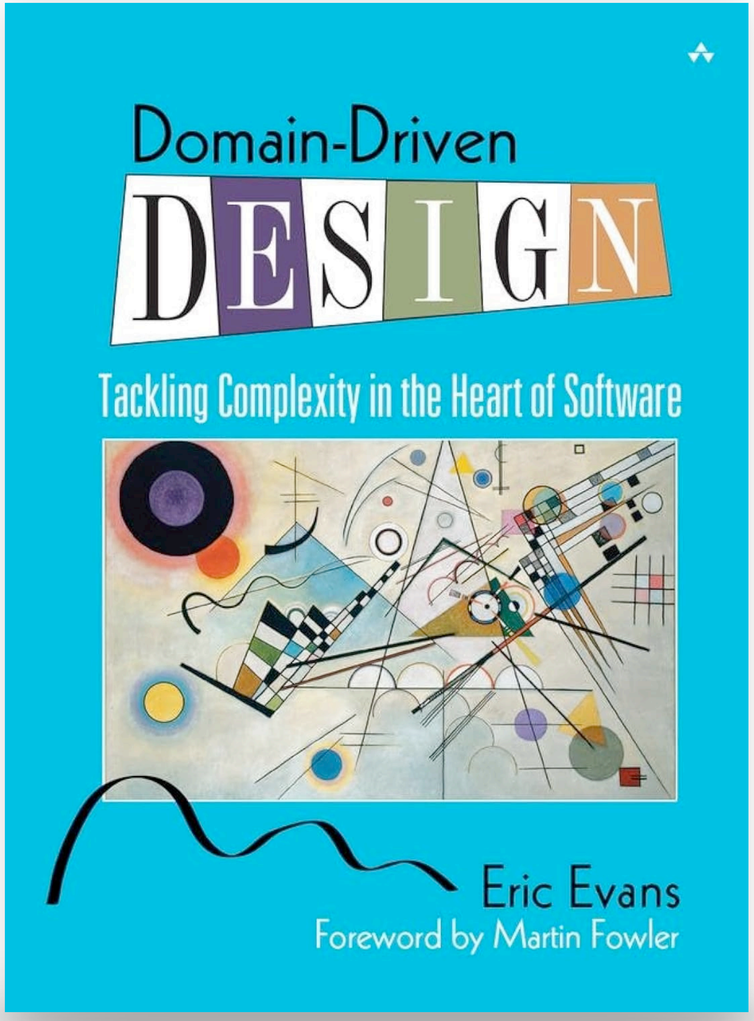
What else? 🤔

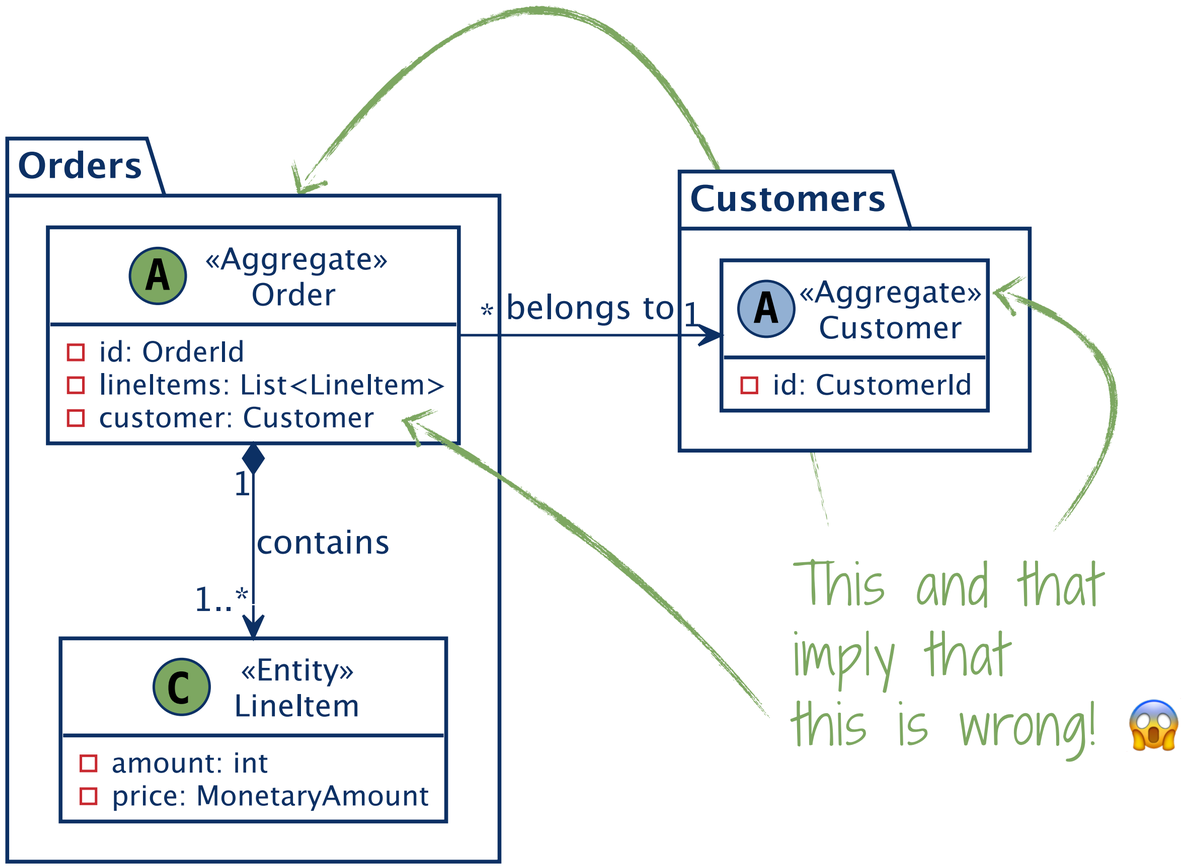
Chunking

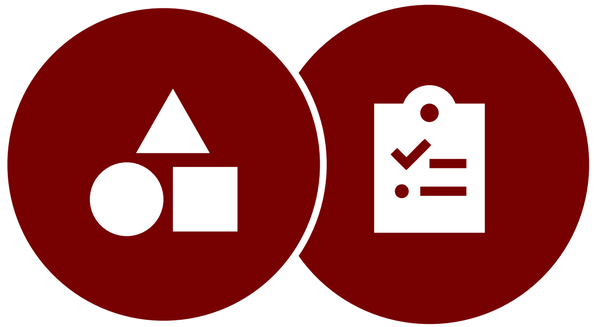
Hierarchization

Pattern languages


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


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler



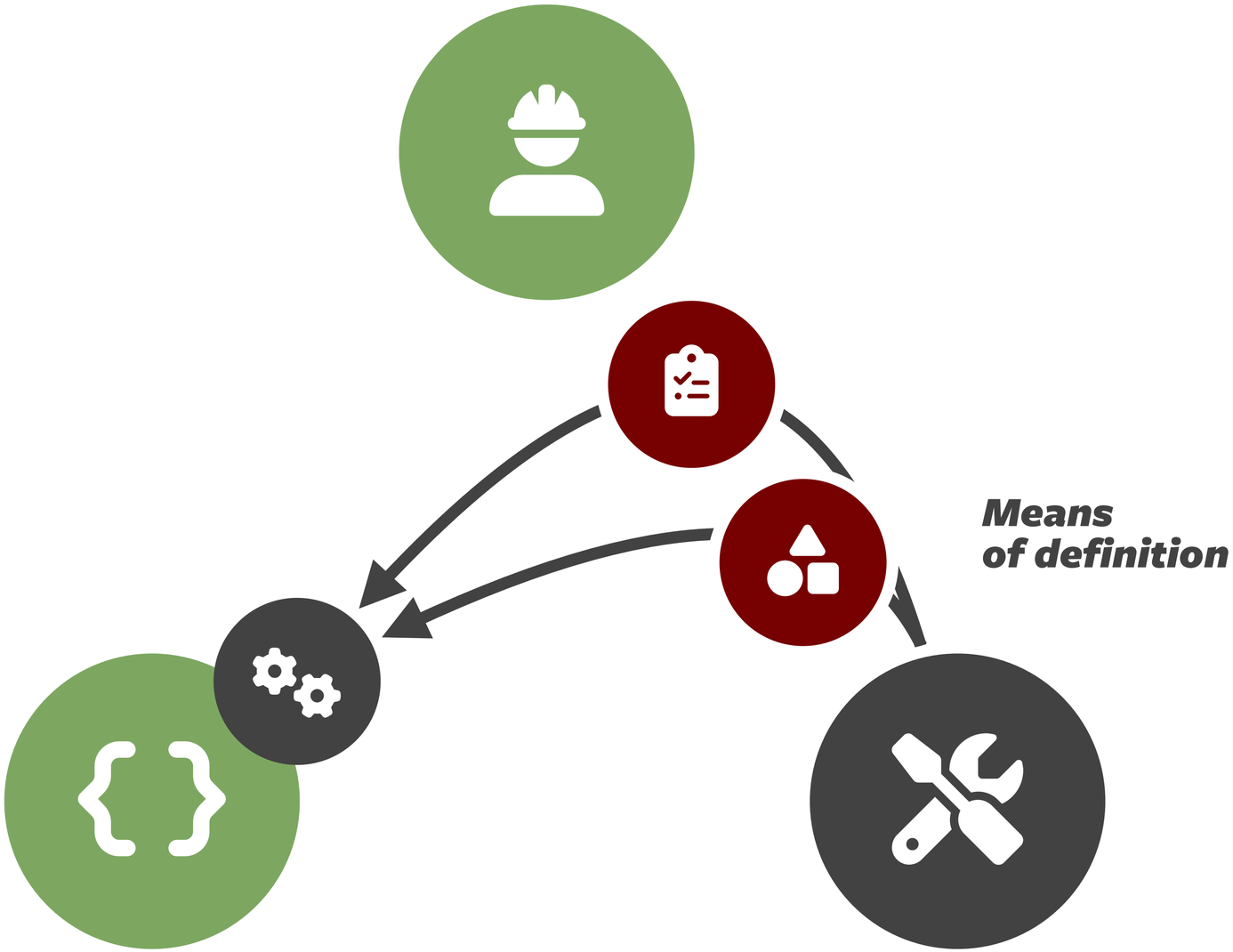


A simple Aggregate arrangement

Code

Architecture

Technology


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
```

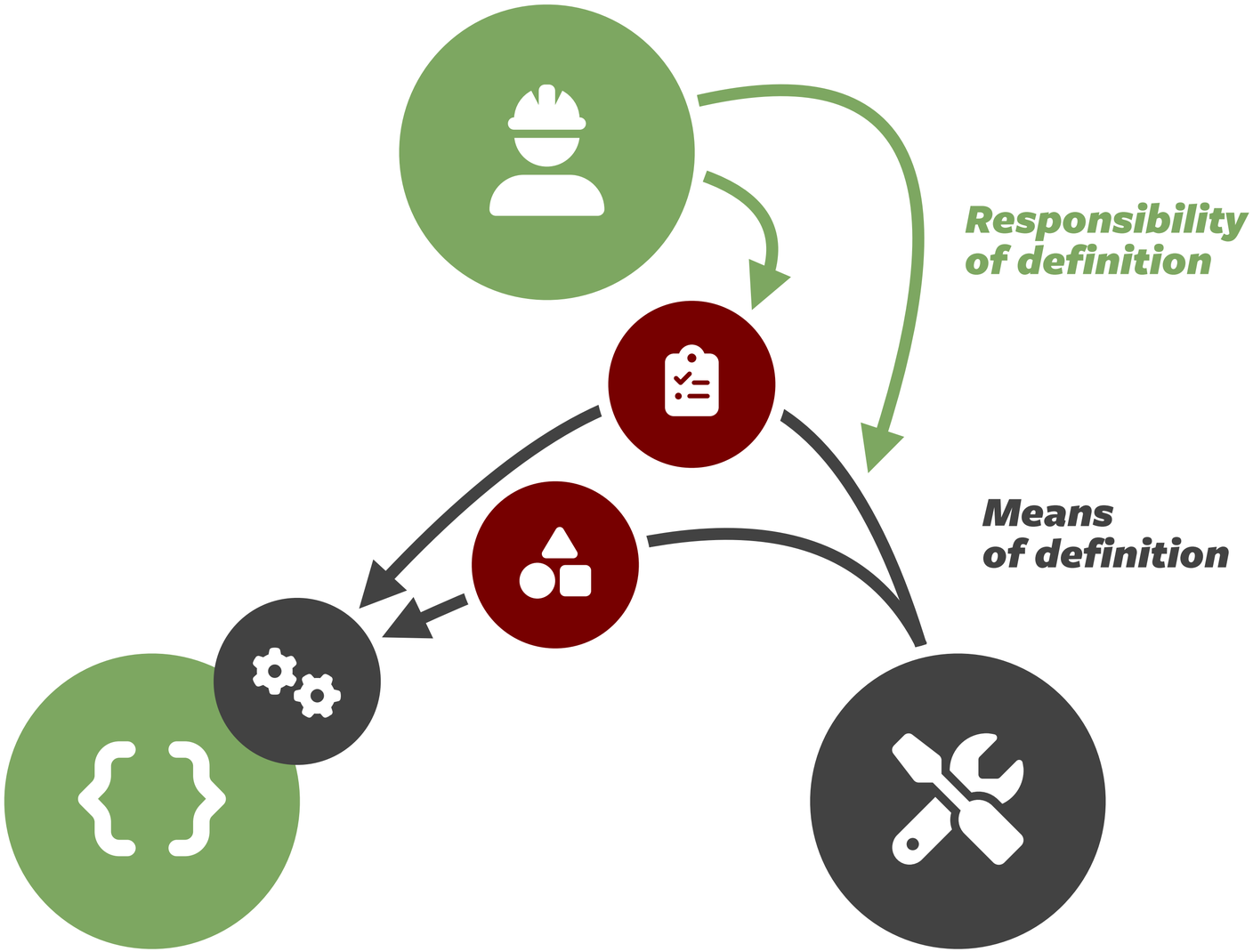
Establishes the rule

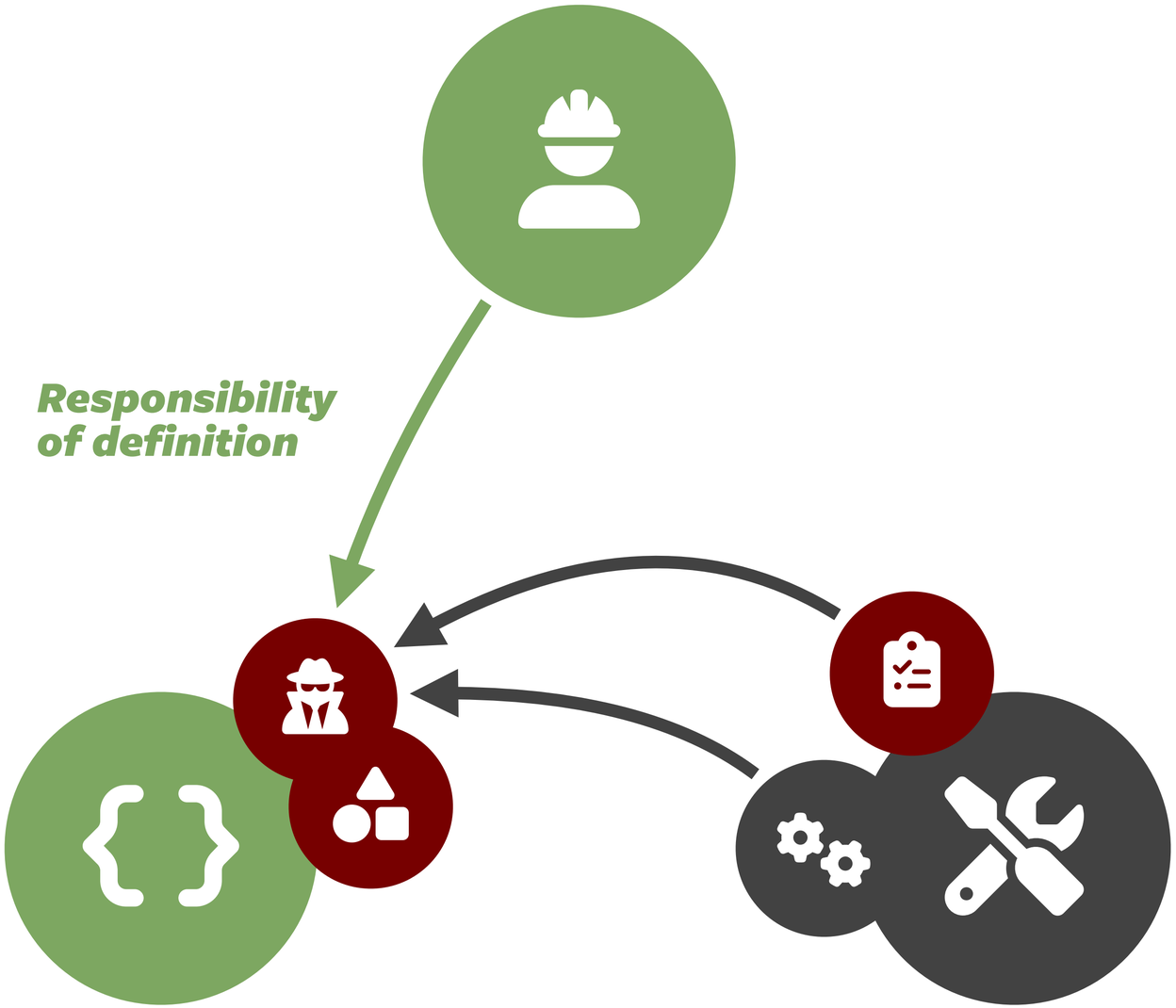
Establishing an Aggregate... in ArchUnit

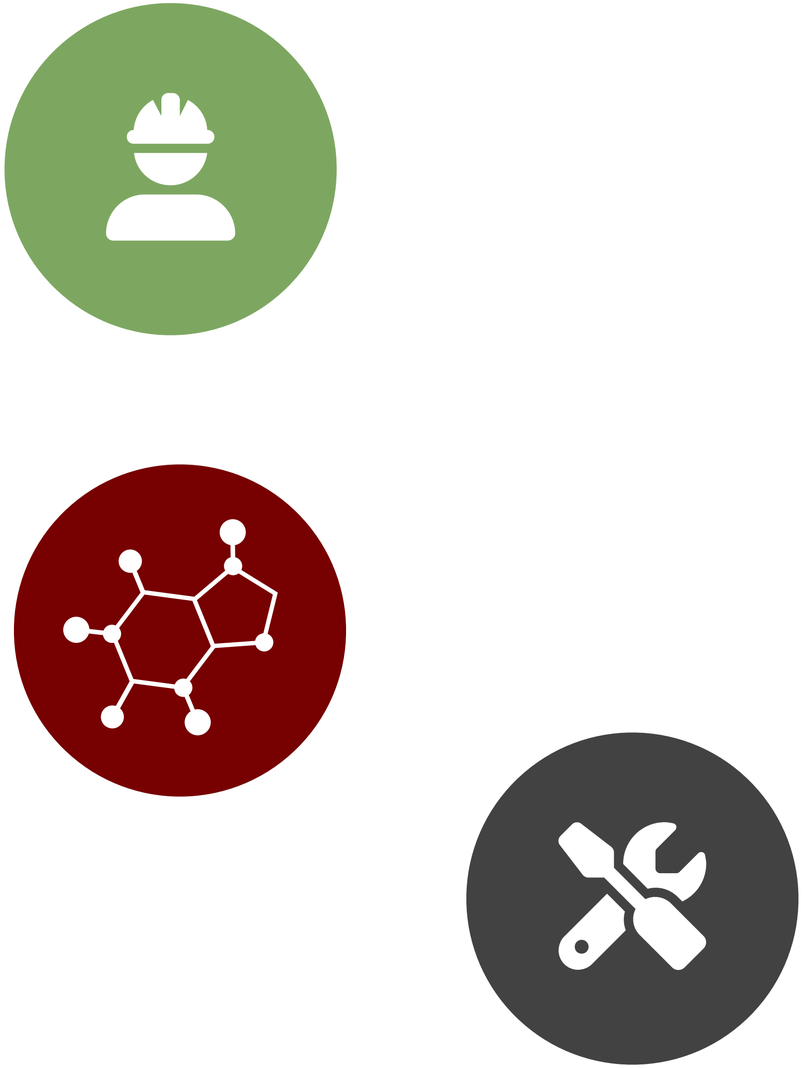
```java
@AnalyzeClasses(packagesOf = Application.class)
public class ArchitectureTest {

  @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
Means
of definition


Responsibility
of definition

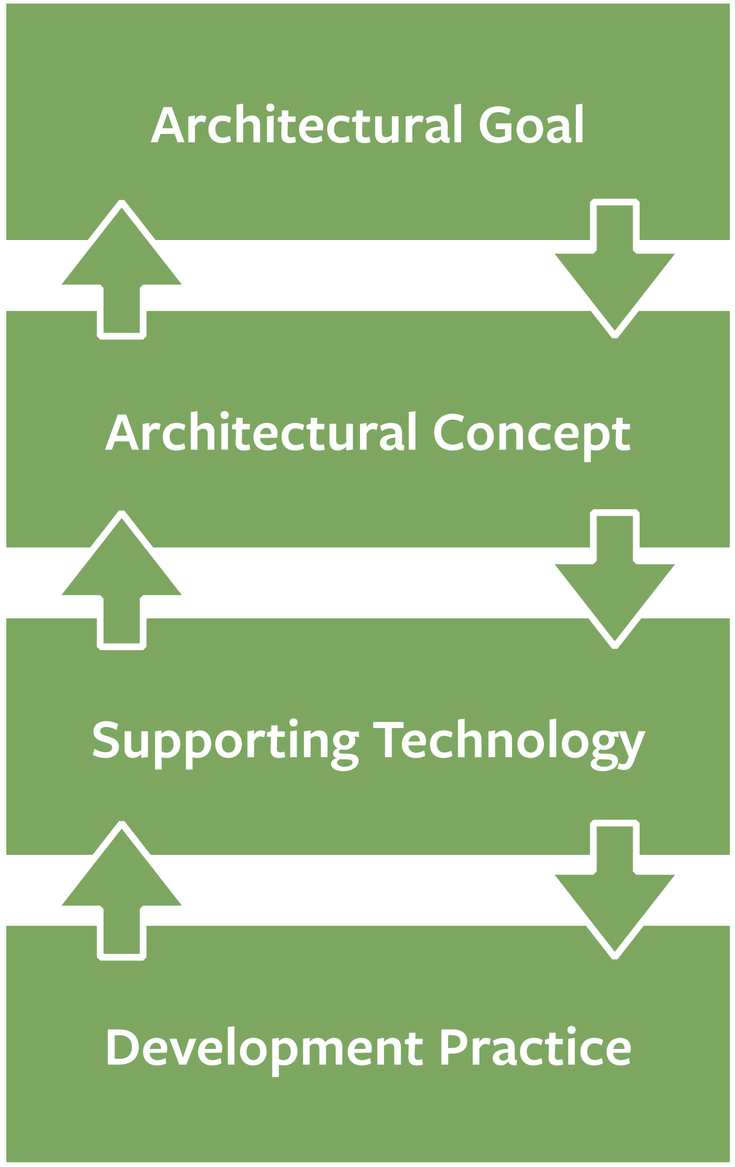


xMolecules

php

.NET

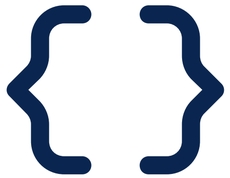
jMolecules



Understandability

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;
  }
}
```

What is the role of this class in the overall arrangement?

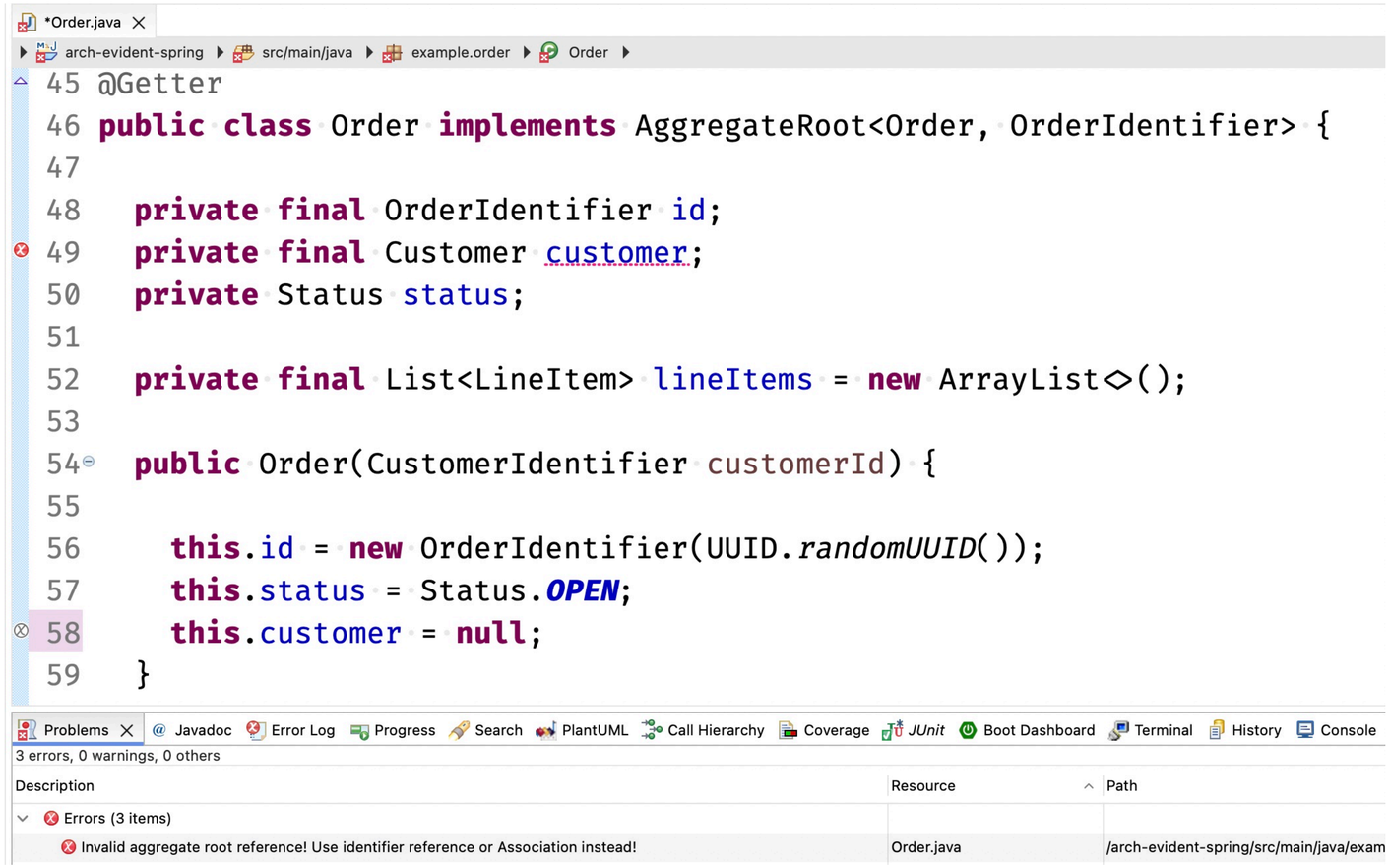
Conceptual Density

$$\frac{1}{N}$$

N = Number of source code elements needed to determine the architectural role of a piece of code

```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;
  }
}
```

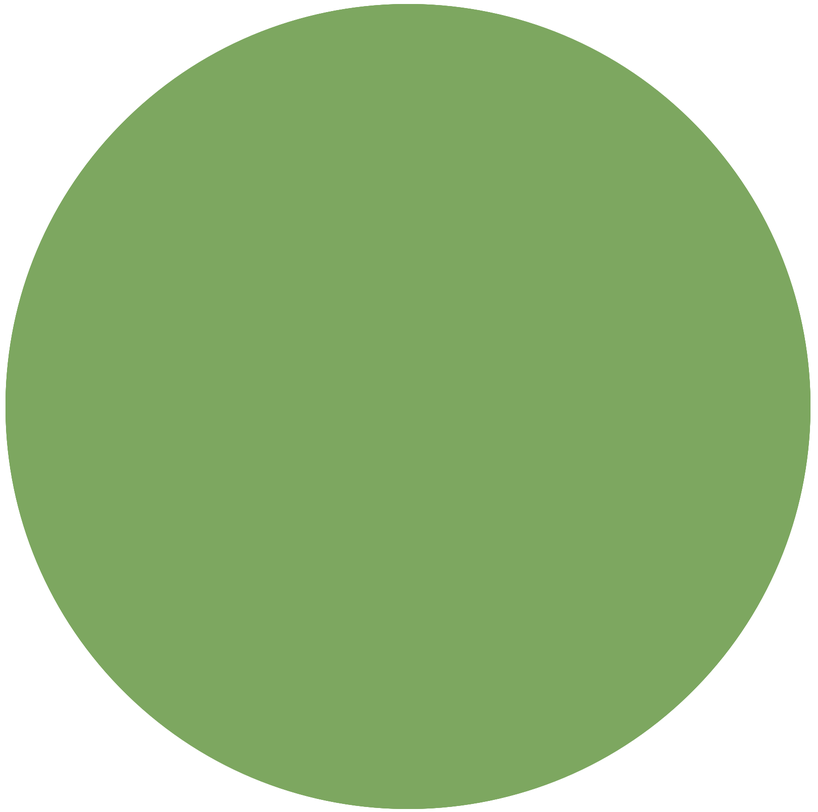
Meanwhile in your IDE…

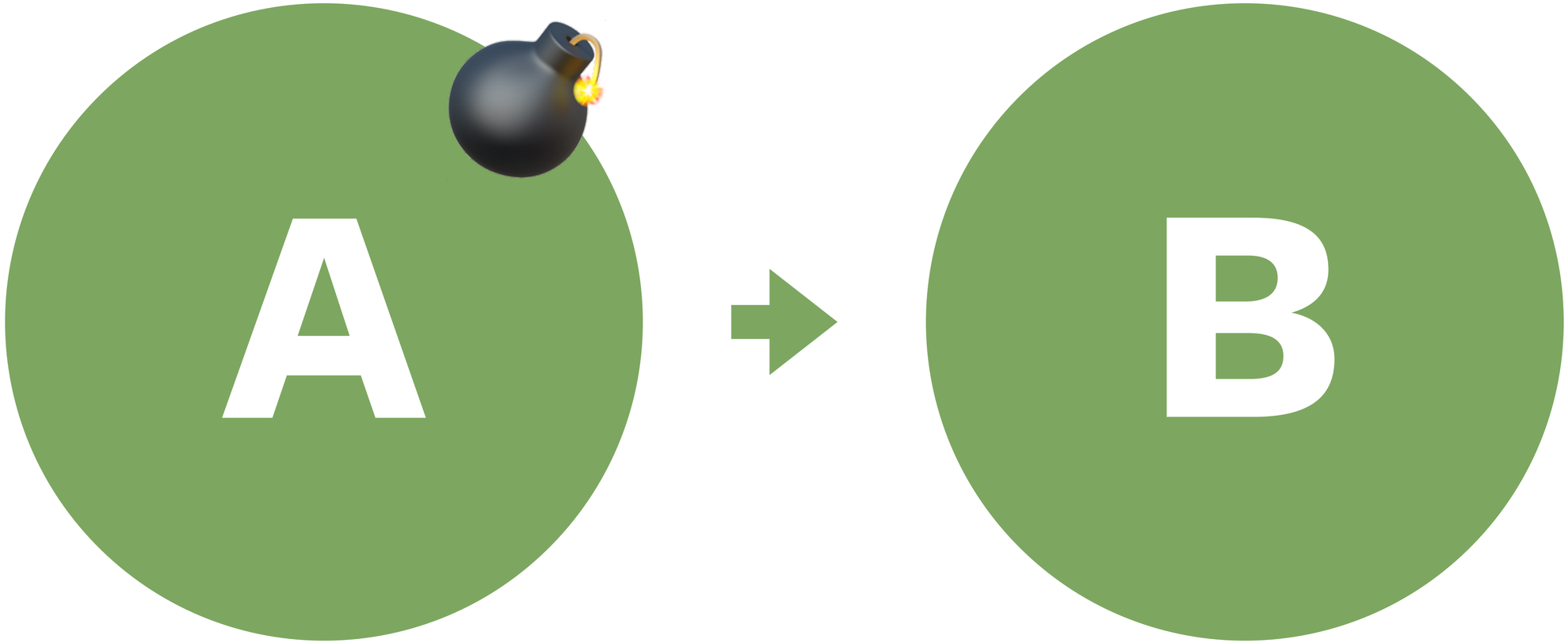
```
[INFO] ◻— example.order.Order
[INFO] ├— JPA - Adding @j.p.Entity.
[INFO] ├— JPA - Adding default constructor.
[INFO] ├— JPA - Adding nullability verification using new callback methods.
[INFO] ├— JPA - Defaulting id mapping to @j.p.EmbeddedId().
[INFO] ├— JPA - Defaulting lineItems mapping to @j.p.JoinColumn(…).
[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=…).
```

```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(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;
  }
}
```

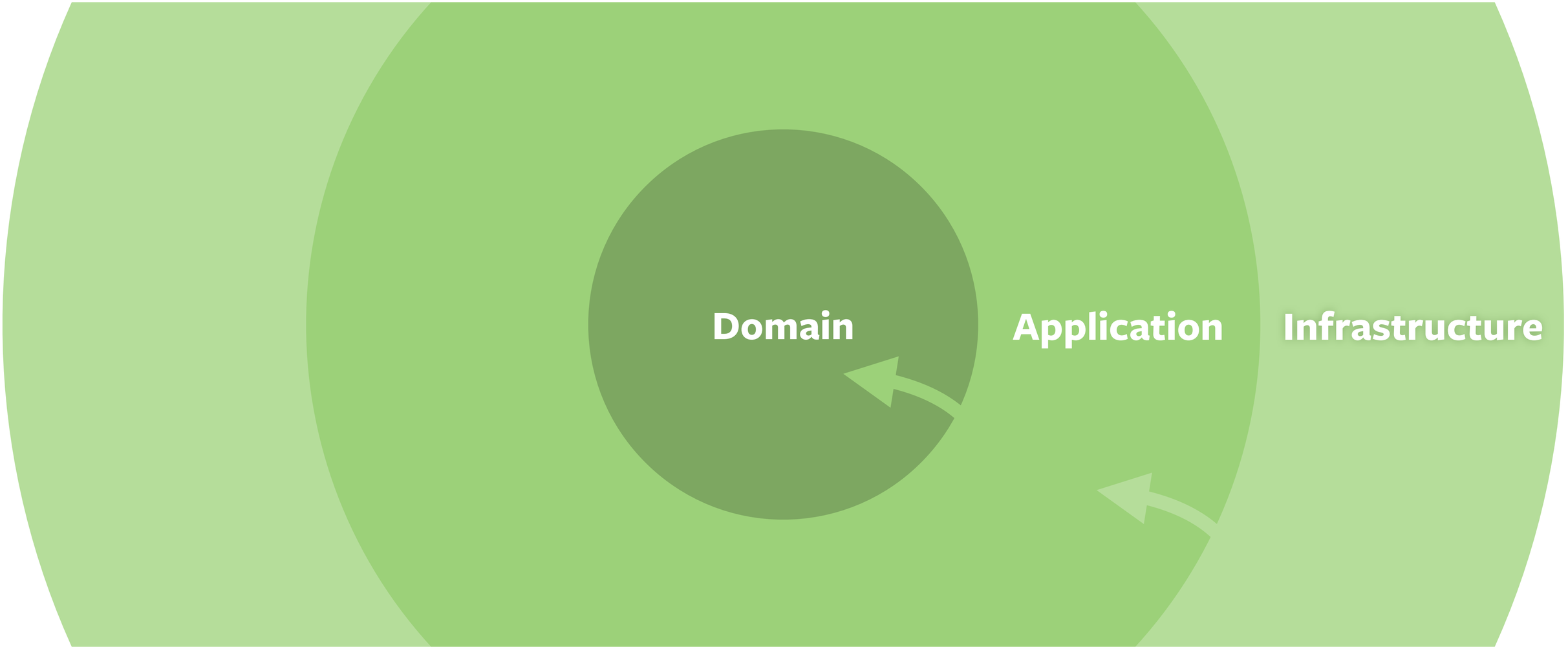
```java
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

  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);
  }

  record OrderId(UUID orderId) implements Identifier {}
}
```

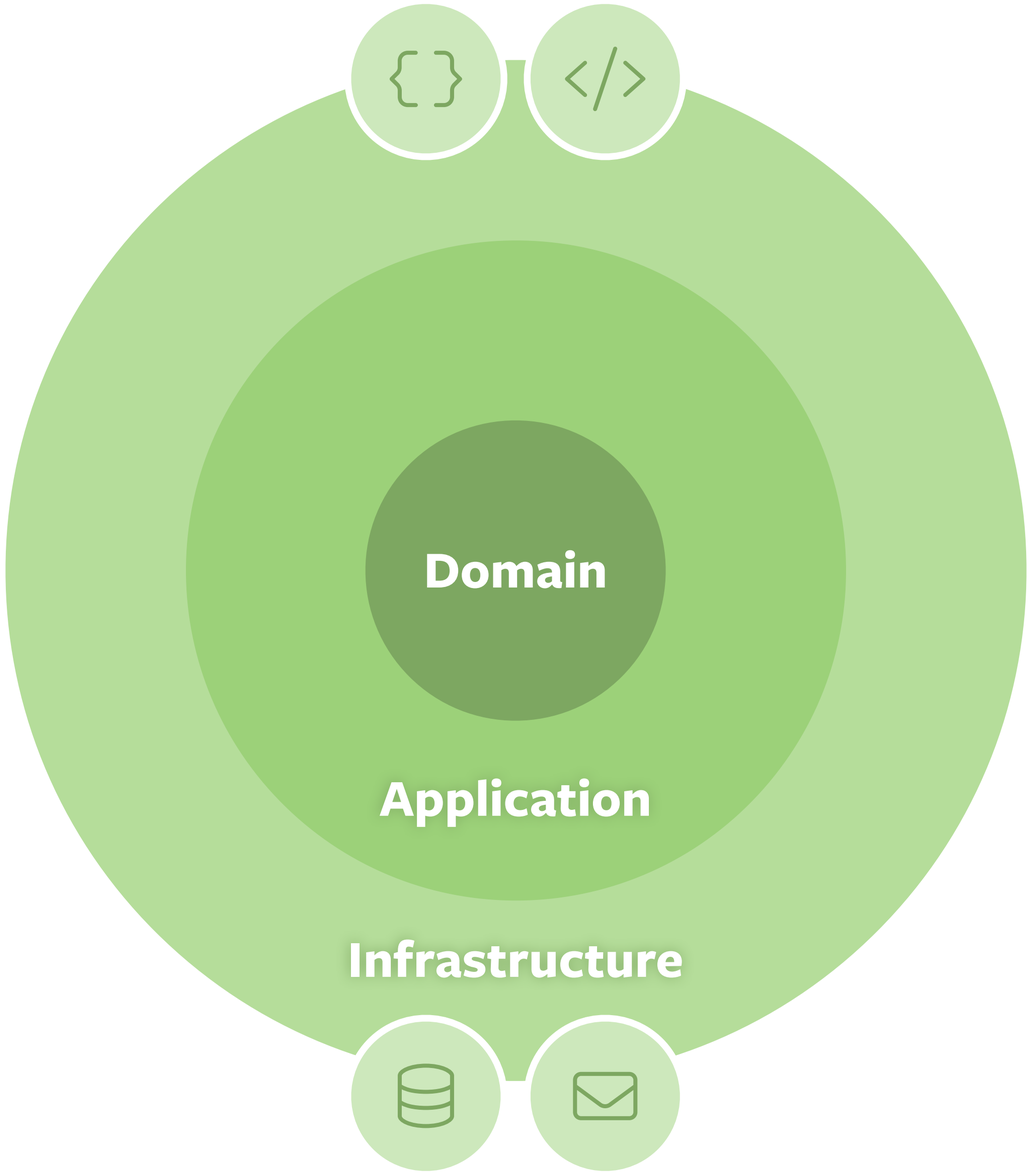
Decomposition

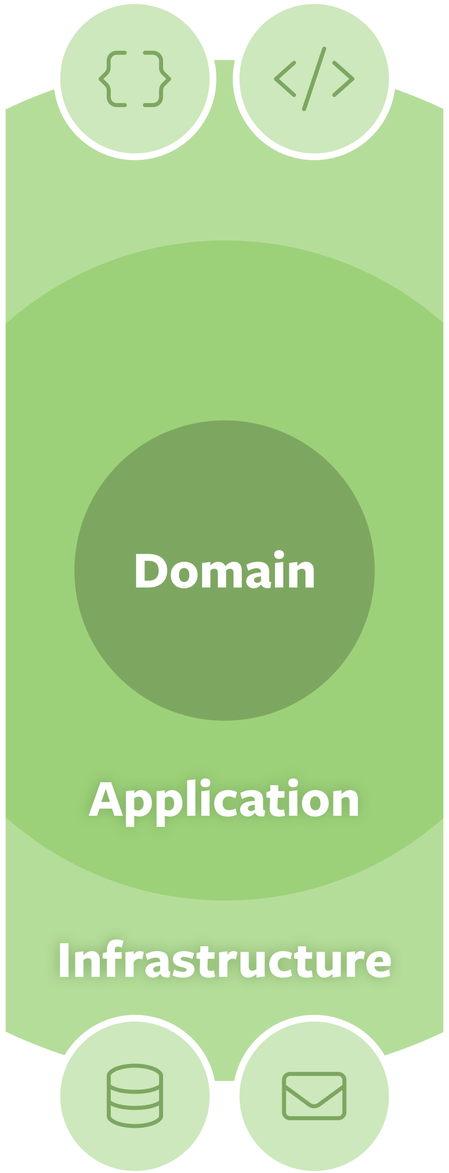


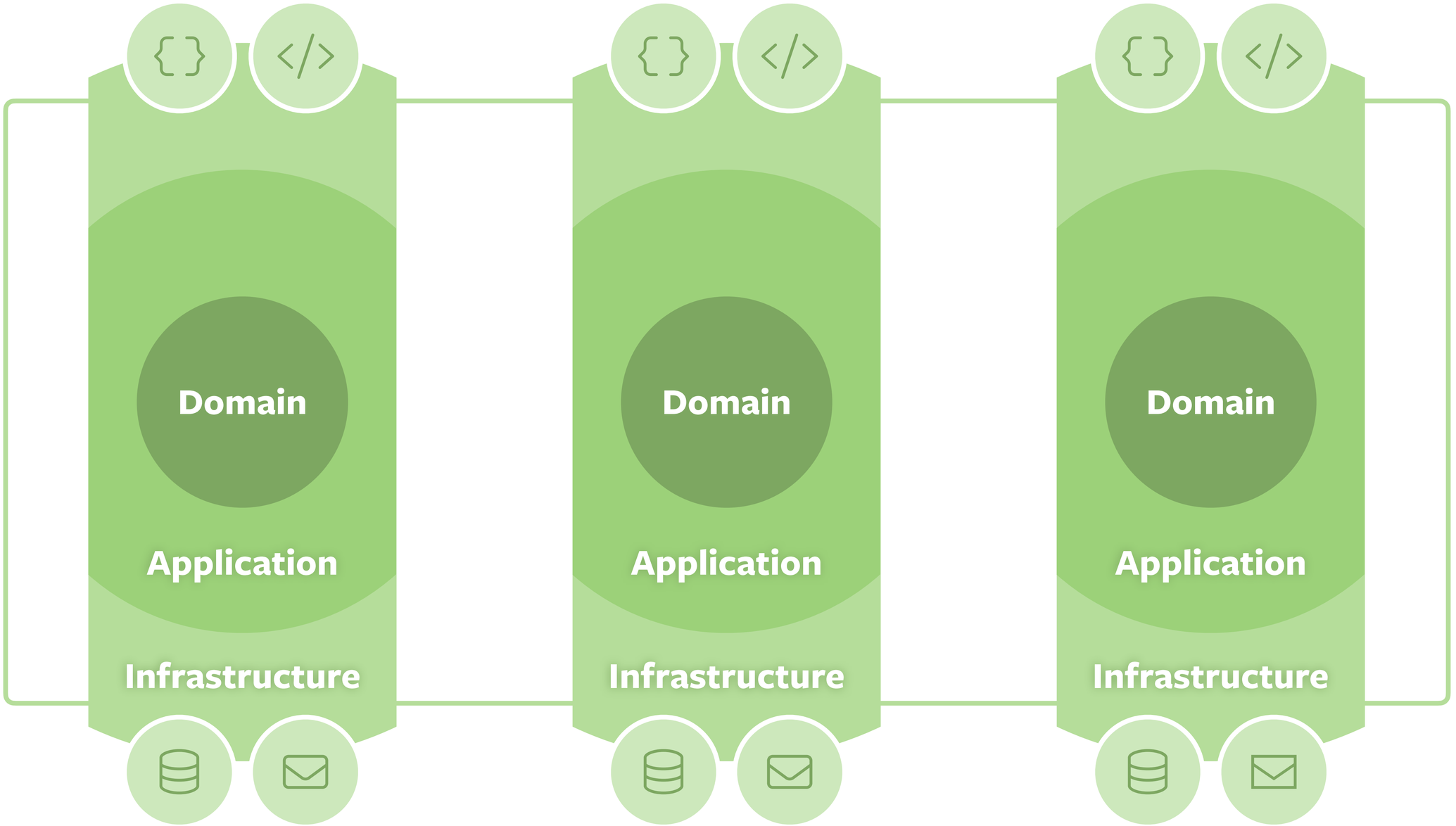


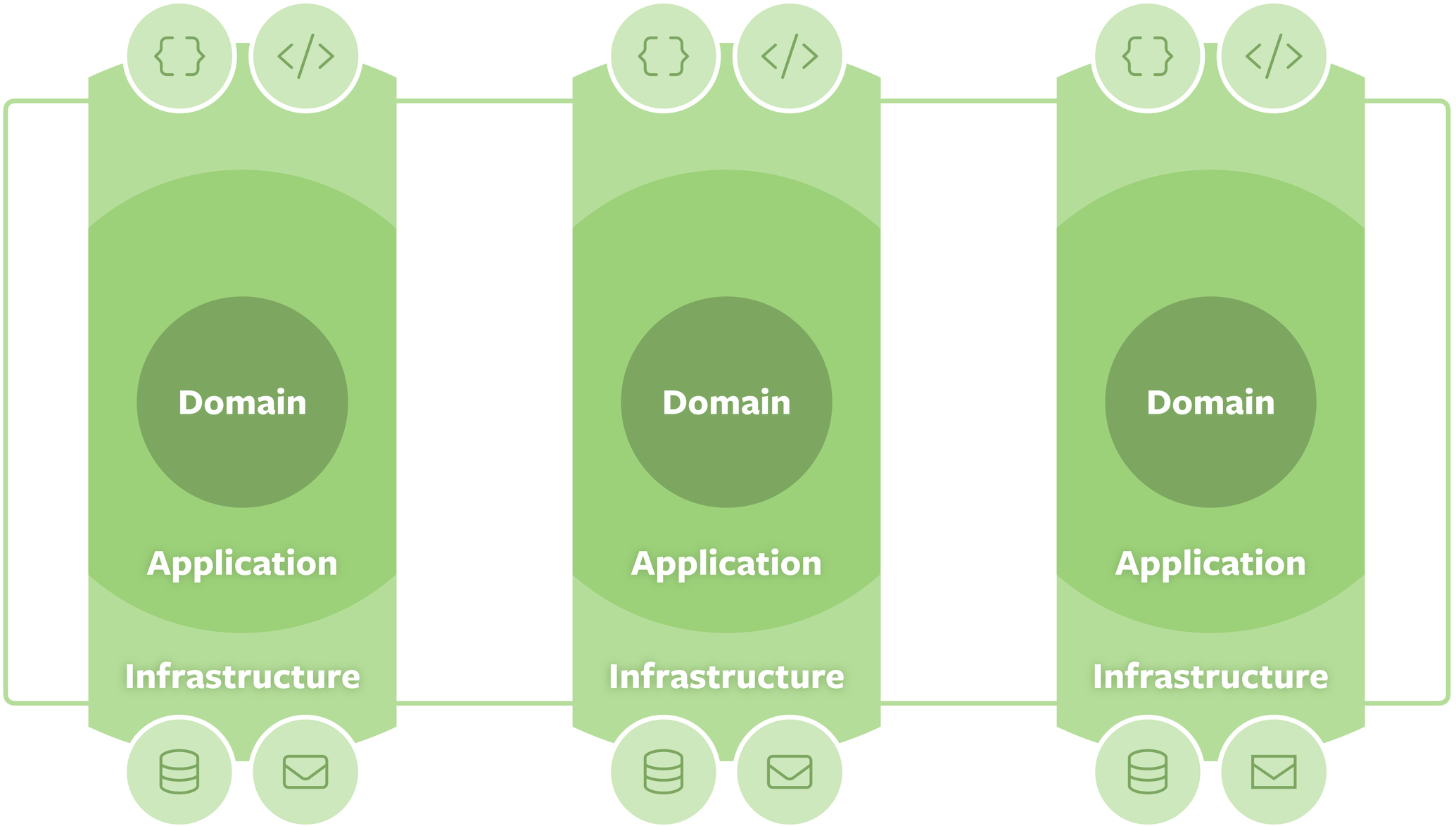
Risk of Change
Scope of Change?

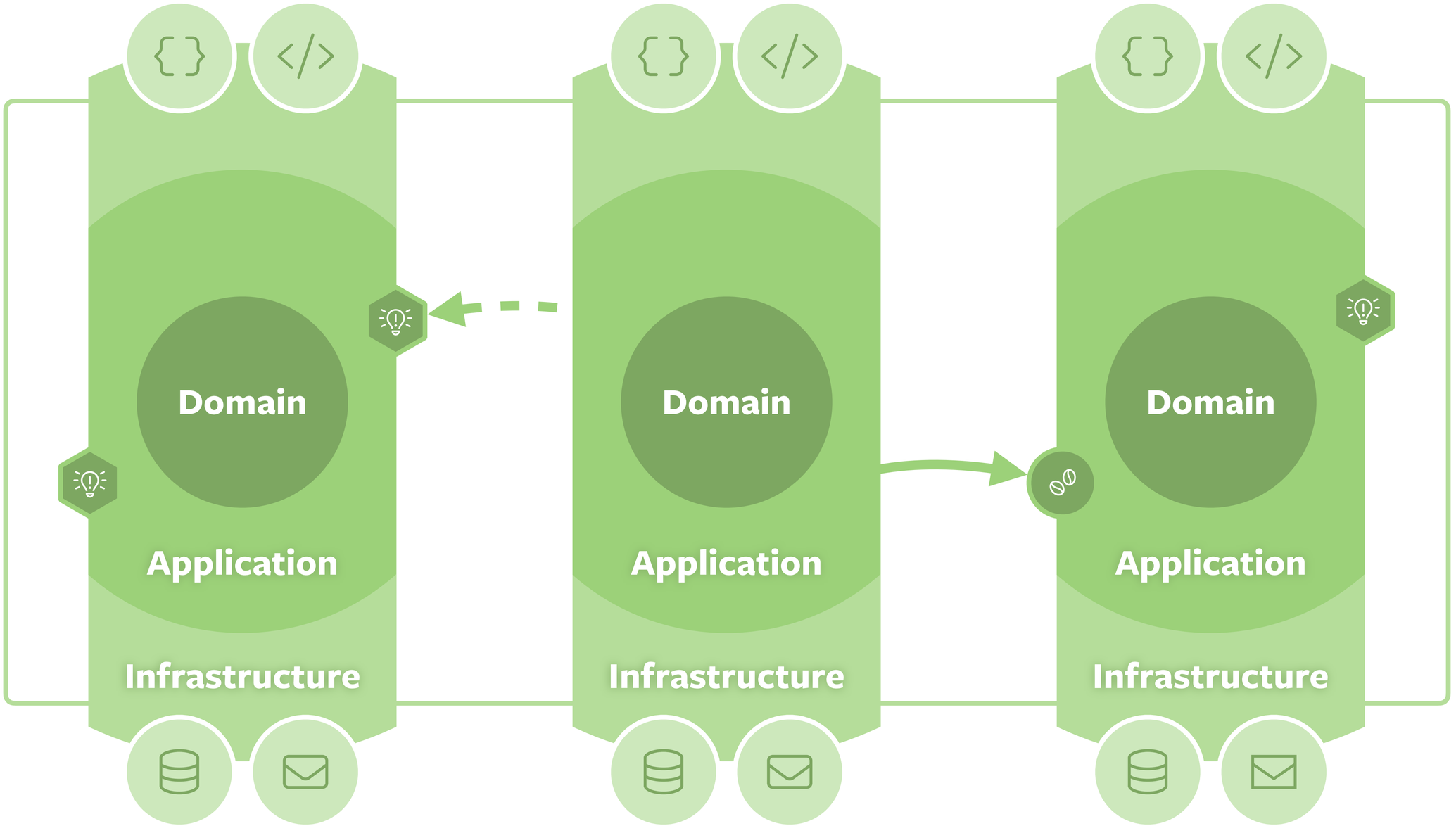

Domain
Application
Infrastructure

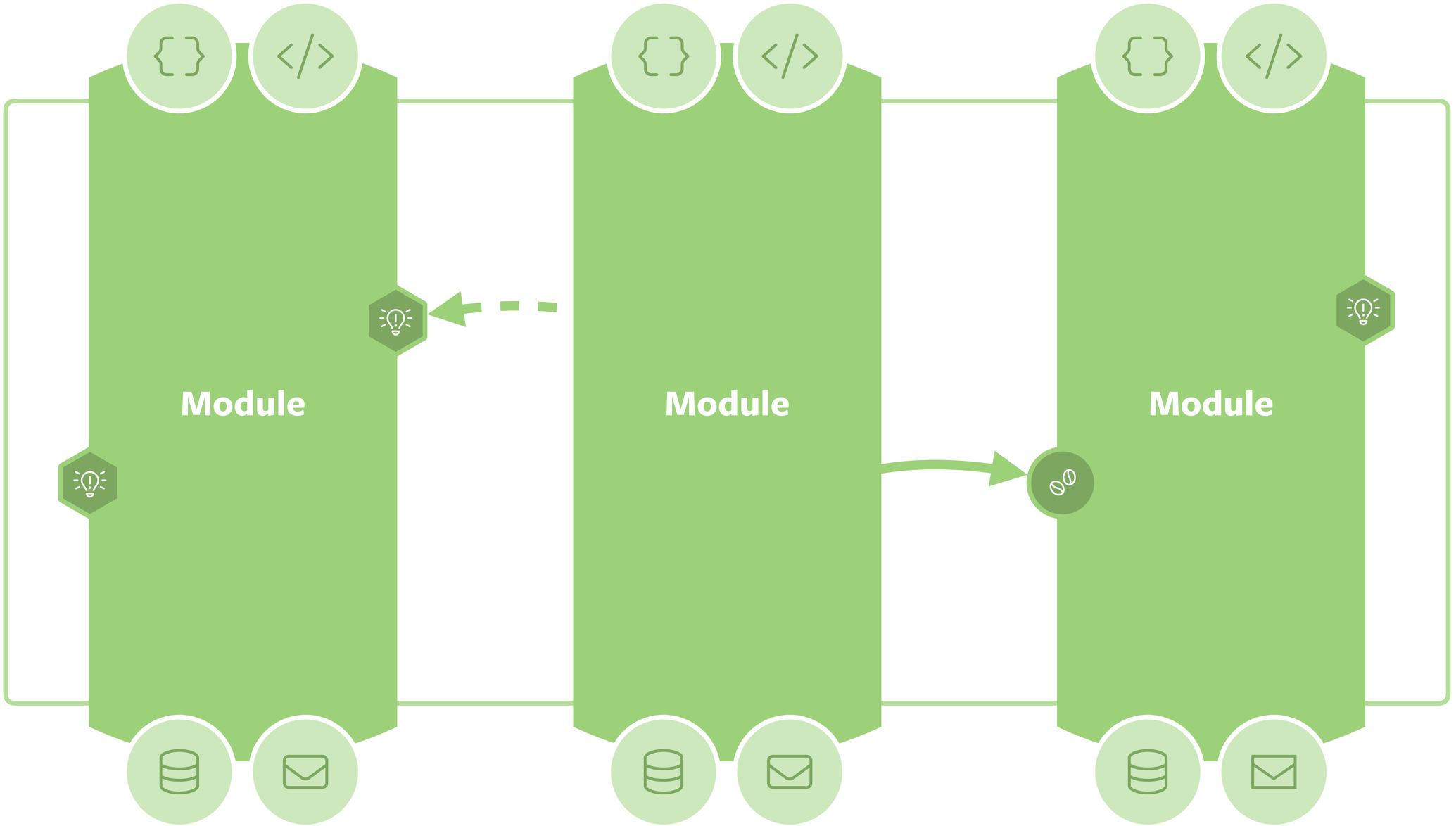

Domain
Application
Infrastructure

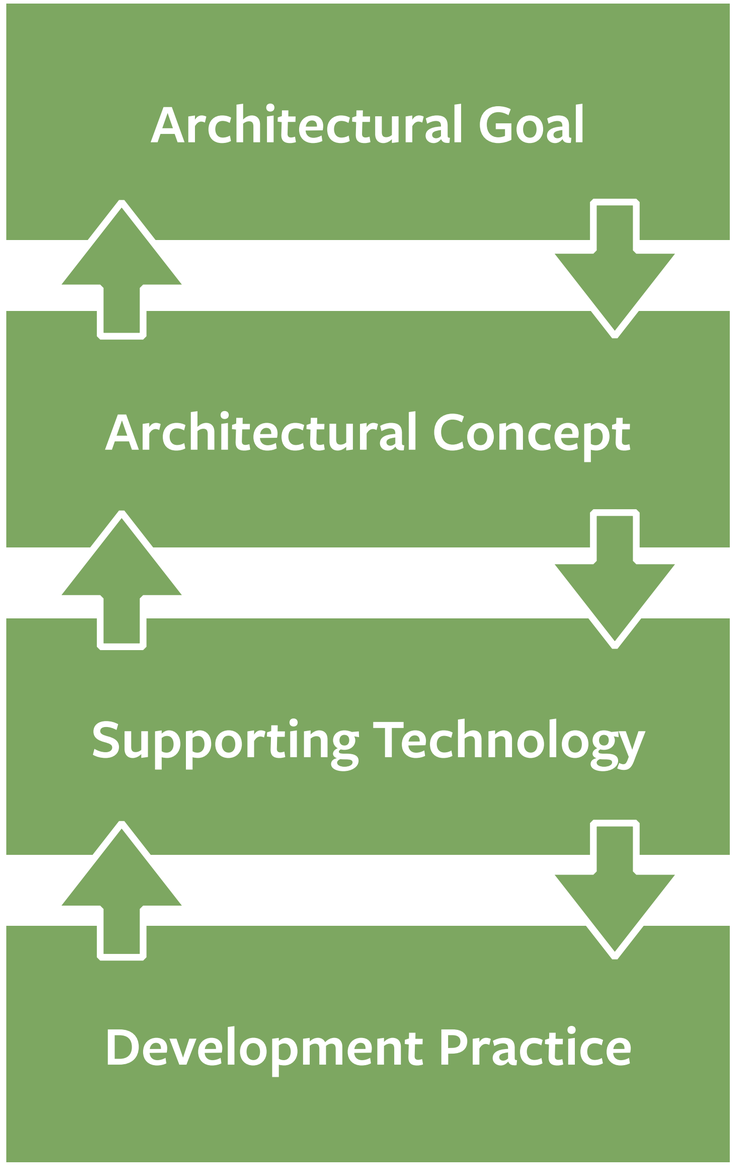

Domain
Application
Infrastructure


Domain
Application
Infrastructure
Domain
Application
Infrastructure
Domain
Application
Infrastructure


Domain
Application
Infrastructure
Domain
Application
Infrastructure
Domain
Application
Infrastructure


Domain
Application
Infrastructure
Domain
Application
Infrastructure
Domain
Application
Infrastructure


Module
Module
Module



Evolvability

Modules

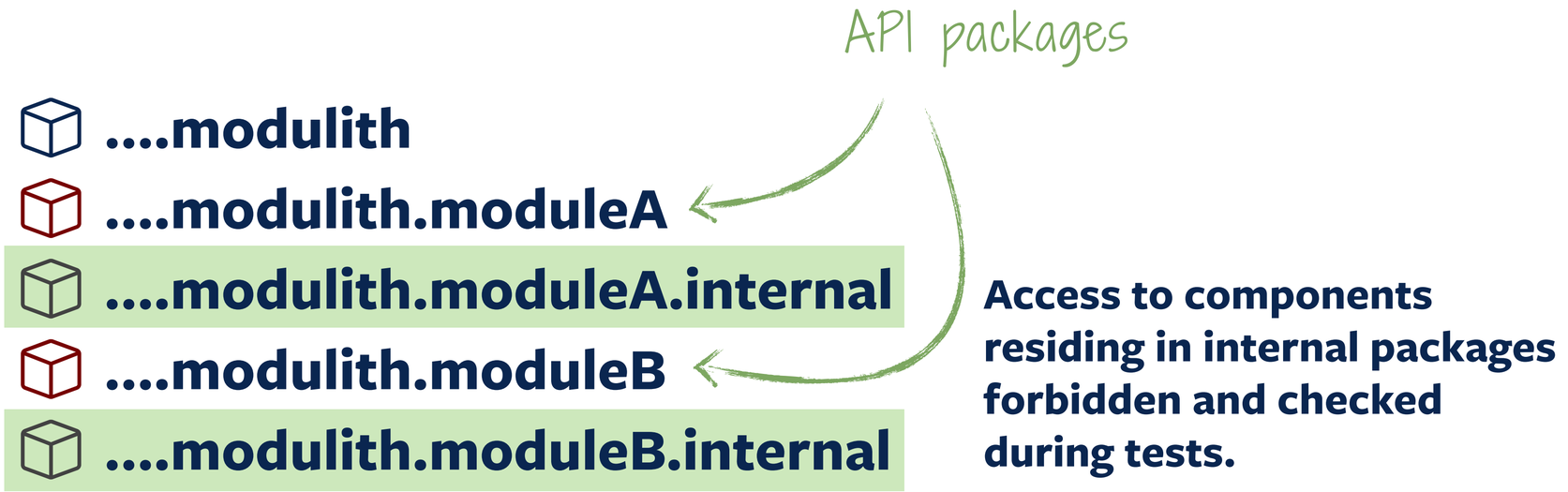
Spring Modulith

Spring Modulith

```
package com.acme.modulith

@SpringBootApplication
class MyApplication { … }
```

Standard Spring Boot Application

Package Conventions



Verification

```
package com.acme.modulith

@SpringBootApplication
class MyApplication { … }
```

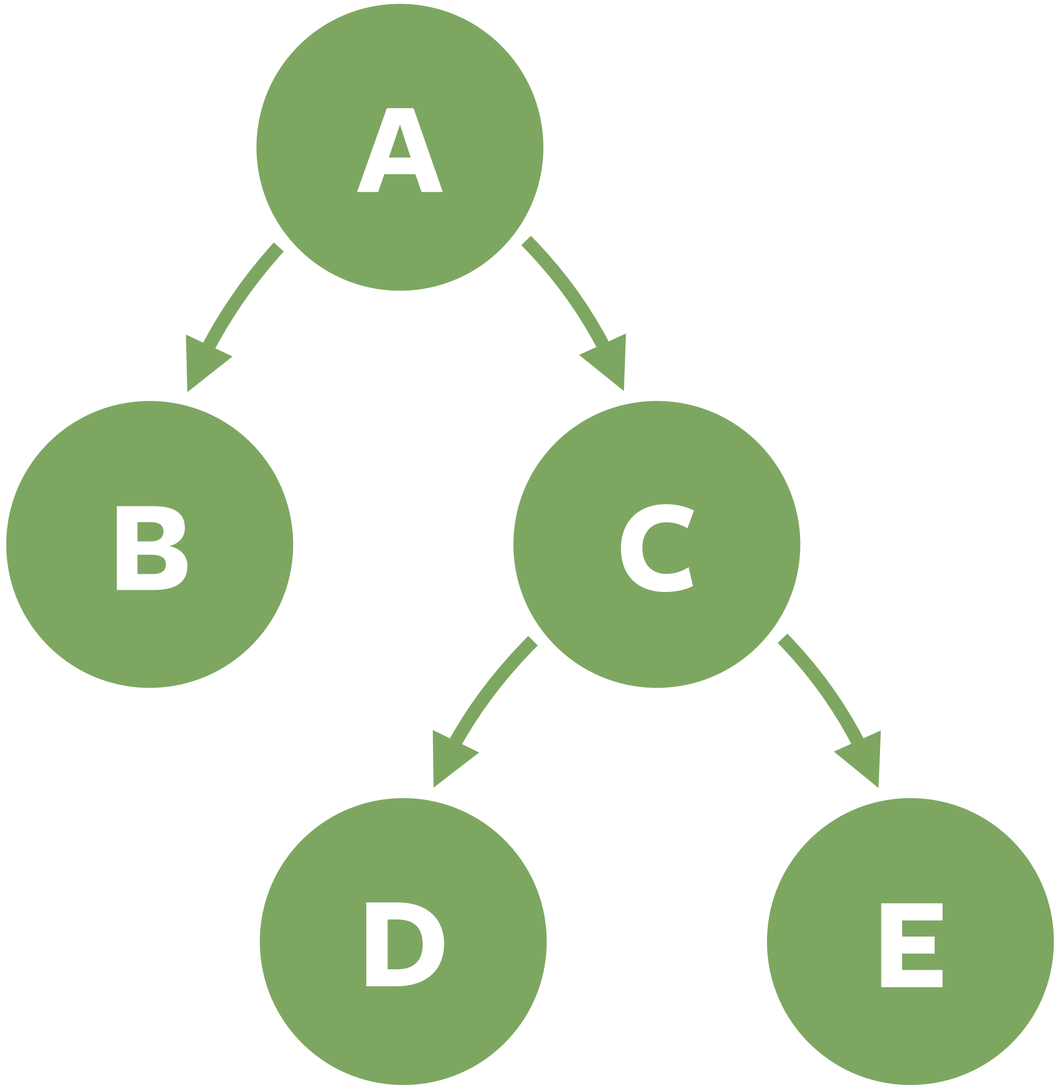
Standard Spring Boot Application

```
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			

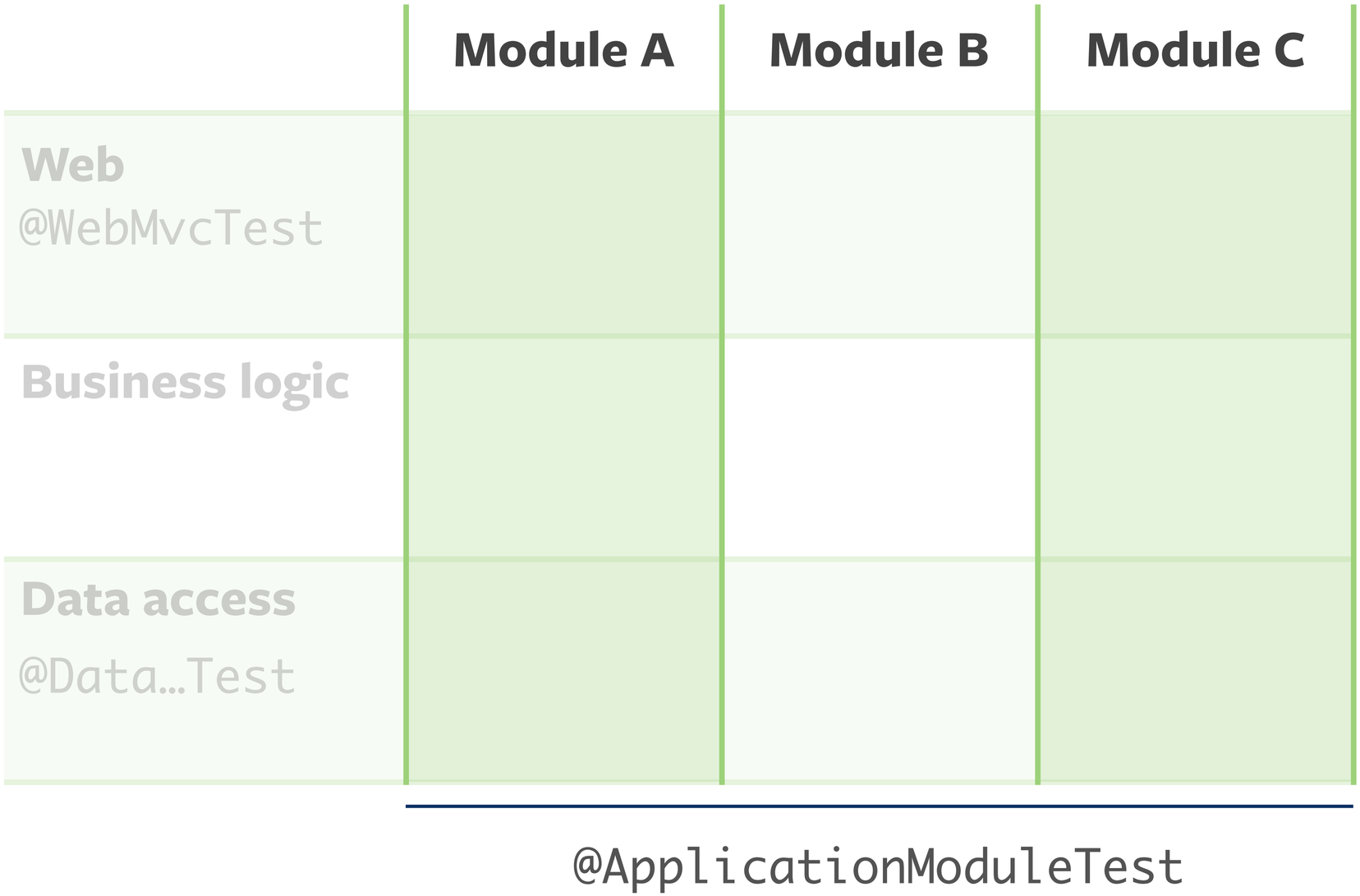


Unit of...

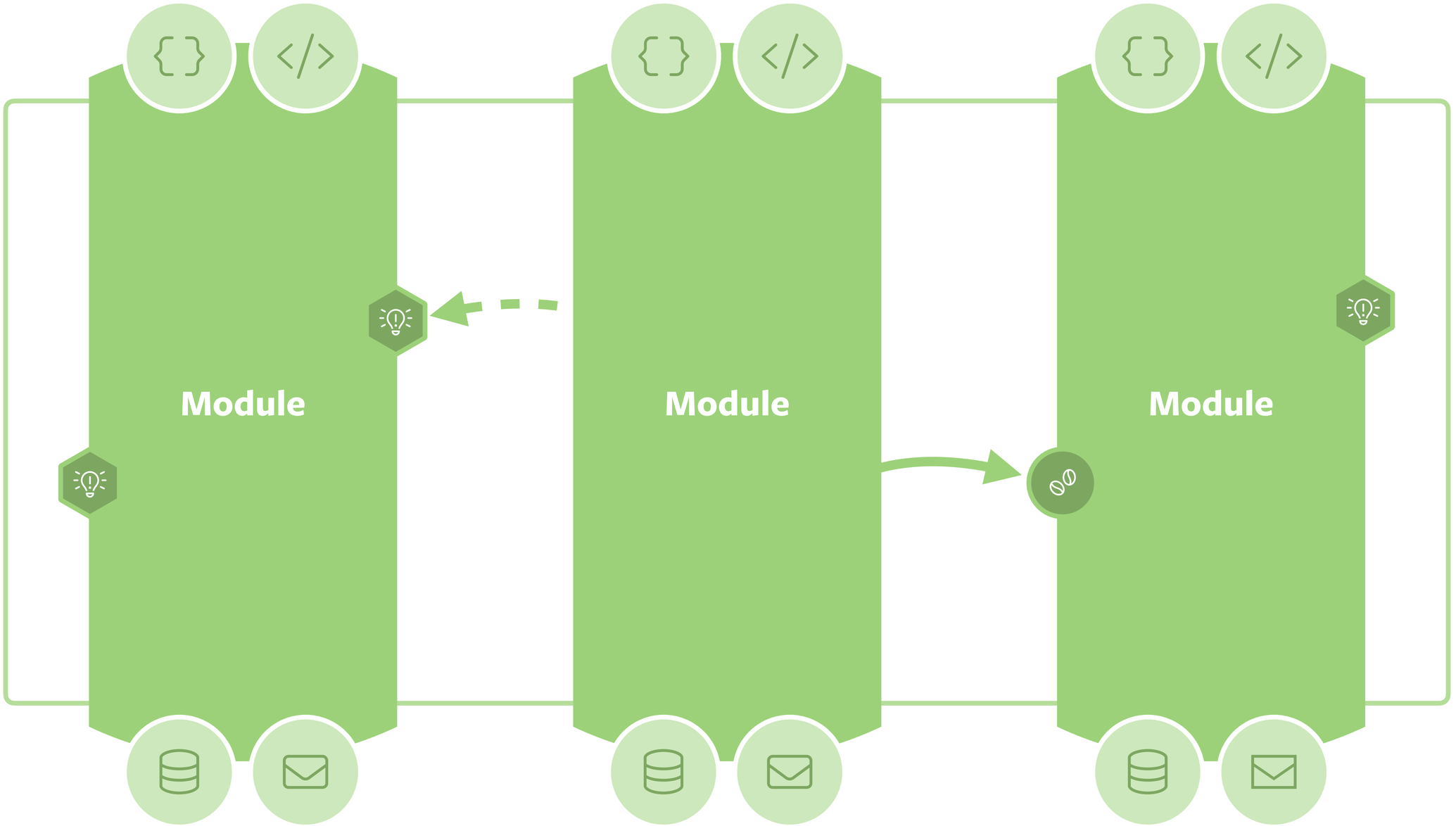
- Understanding
- Consistency
- Testing
- Documentation
- Observation

Testing

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

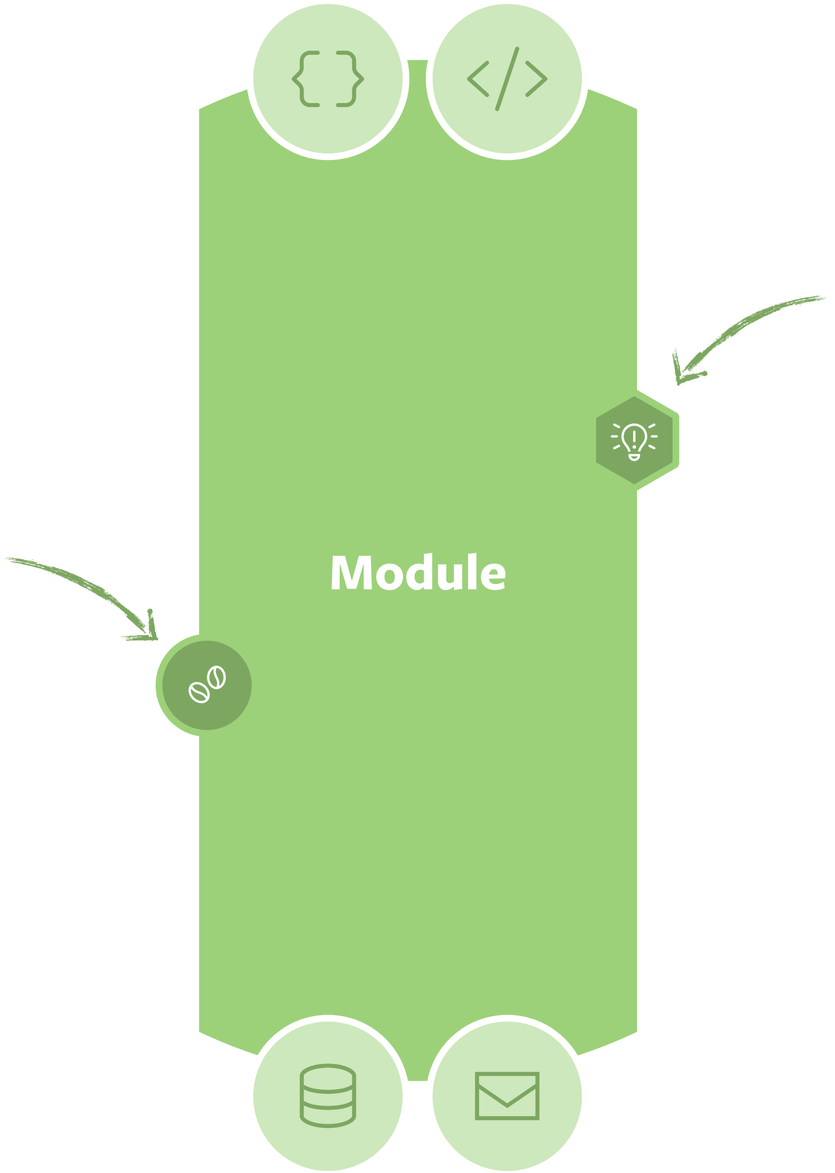

Module A
Module B
Module C
Web
@WebMvcTest
Business logic
Data access
@Data…Test
@ApplicationModuleTest

Subcutaneous Tests


Module
Module
Module

Stimulate the module by...

...invoking a business operation
...publishing an event listened to



```java
@ApplicationModuleTest
@RequiredArgsConstructor
class OrderIntegrationTests {

  private final OrderManagement orders;

  @Test
  void completionCausesEventPublished(Scenario scenario) {

    var order = new Order(new CustomerIdentifier(UUID.randomUUID()));

    scenario.stimulate(() -> orders.complete(order))

        .andWaitForEventOfType(OrderCompleted.class)
        .matchingMappedValue(OrderCompleted::id, order.getId())
        .toArrive();
  }
}
```

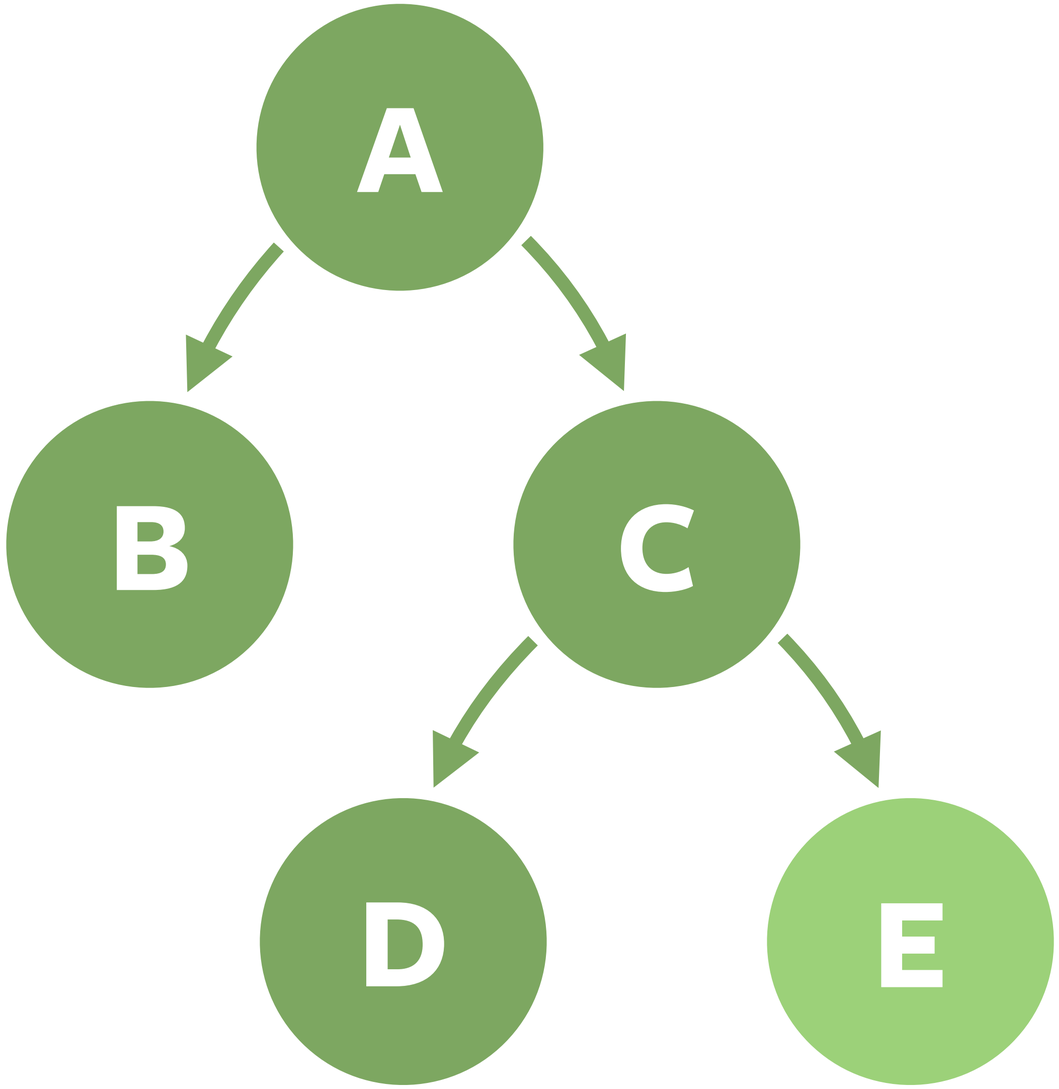
Test scoped to the module

Public API primary interaction target

Given
When
Then

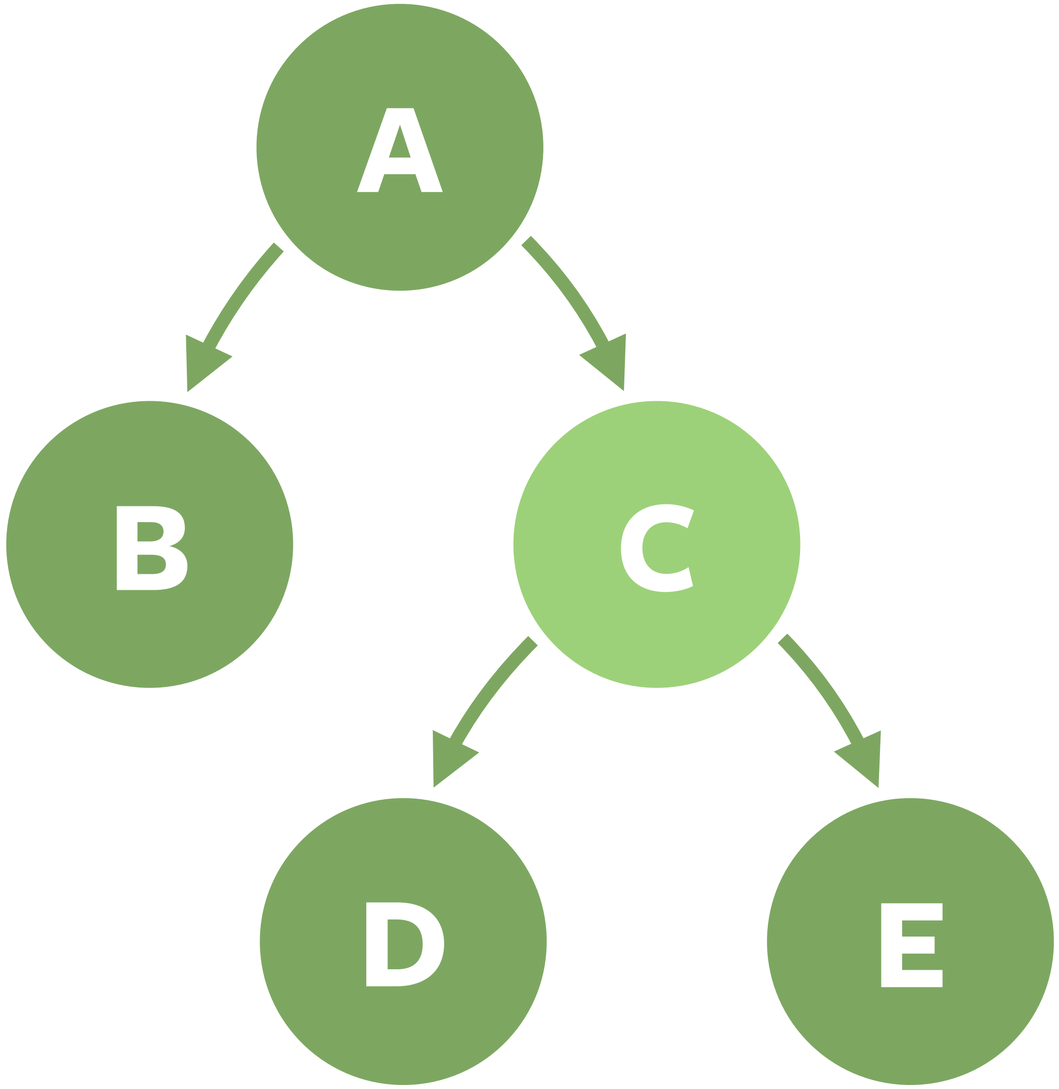
Stimulate module

Scope of Individual Tests



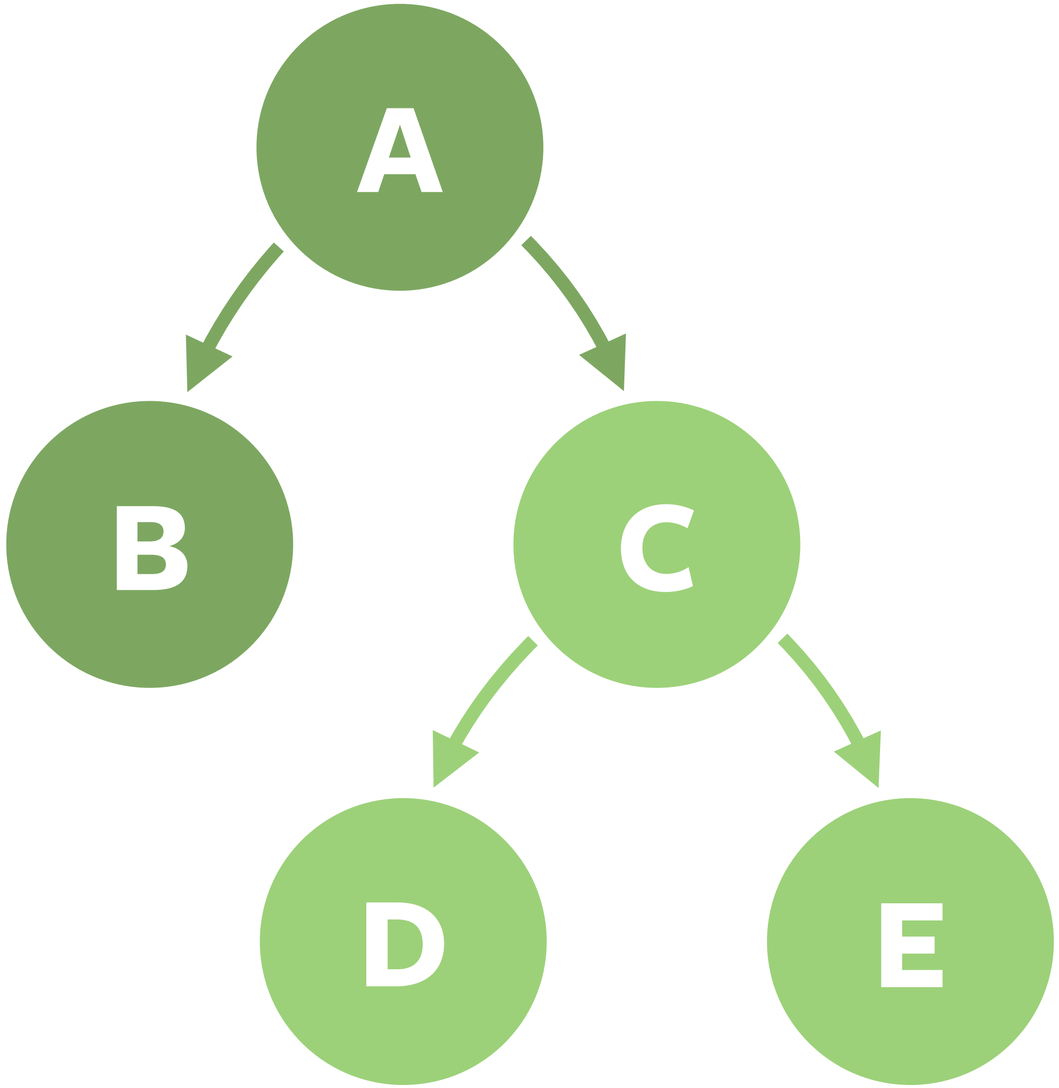
Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation



Unit of...

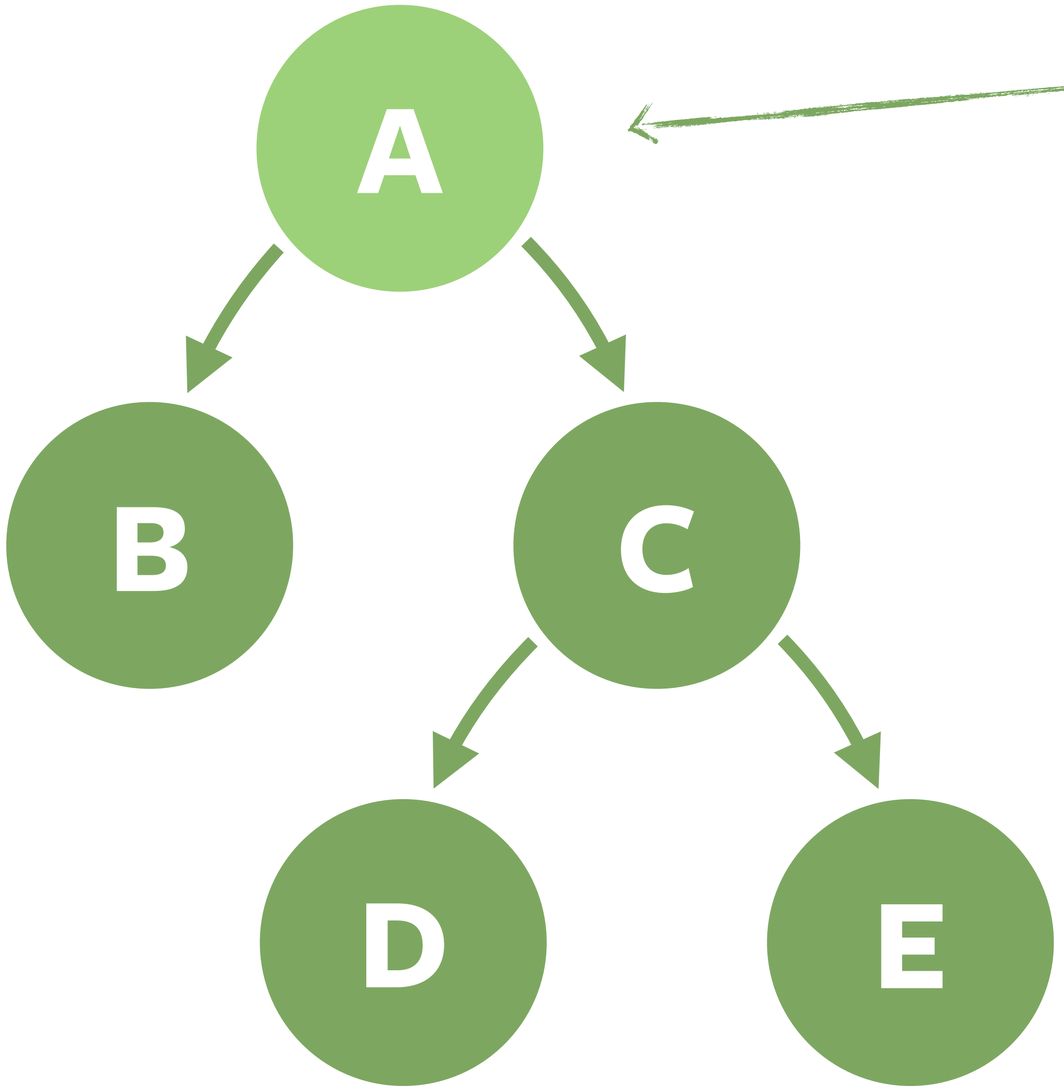
- Understanding
- Consistency
- Testing
- Documentation
- Observation



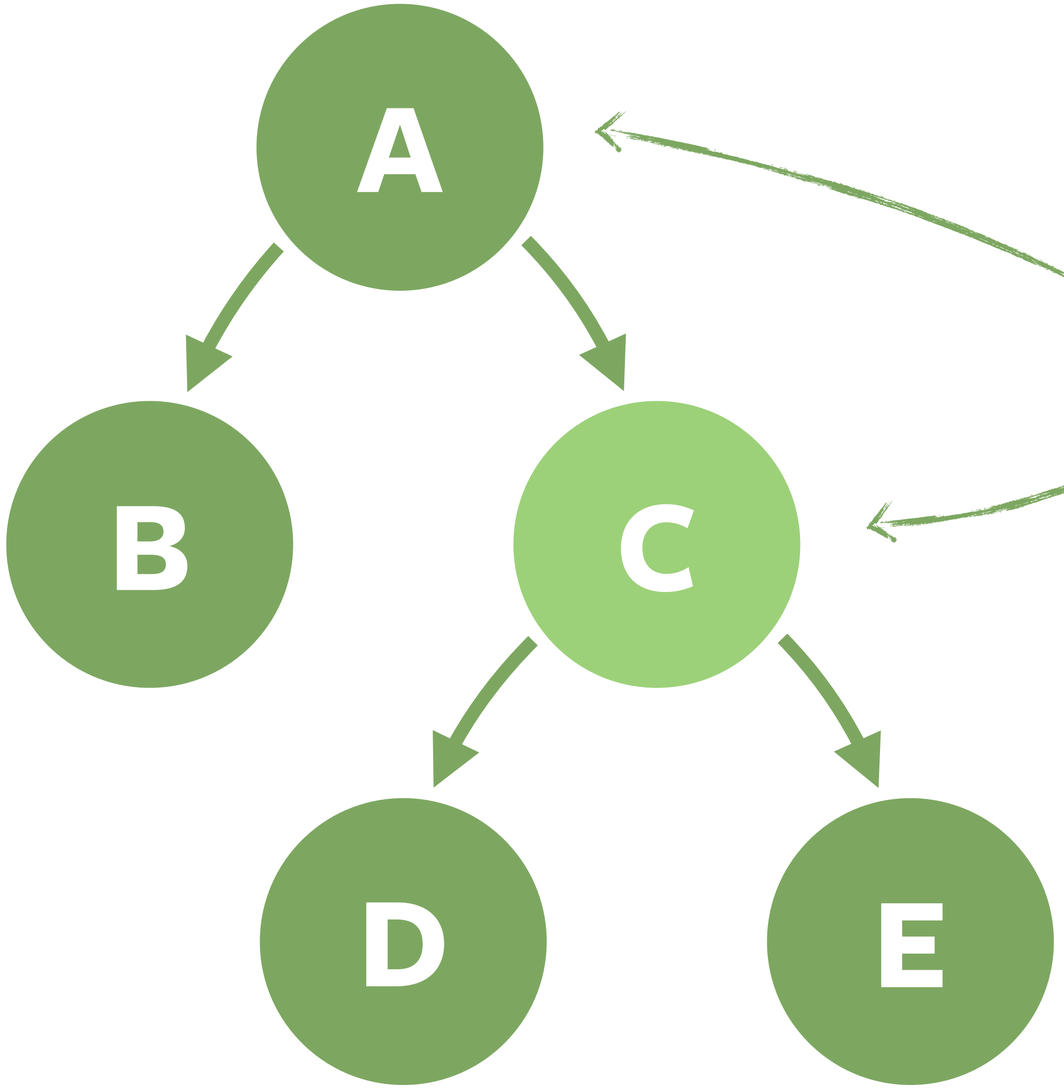
Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation

Scope of Test Execution



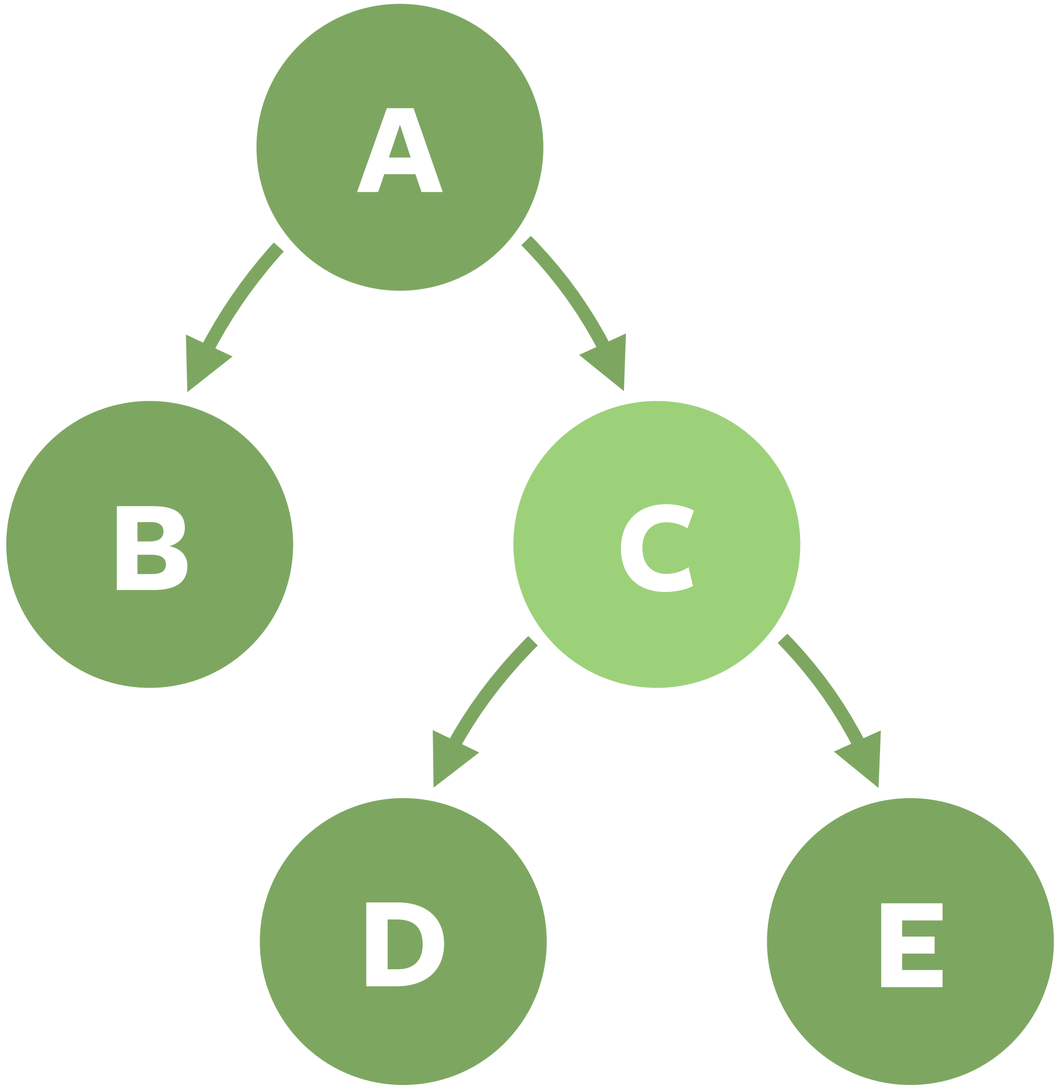
Change detected here.
We only need to test A!



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

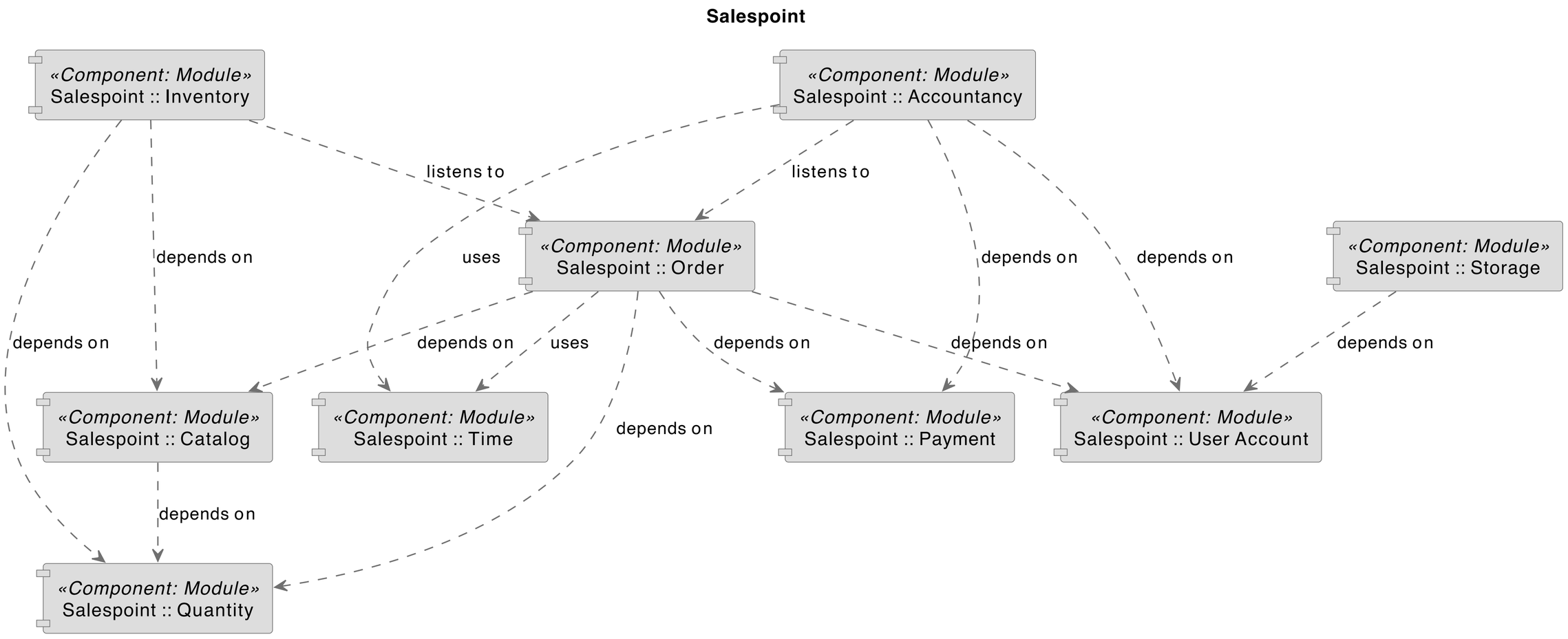
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
15:29:09.748 I ----- main : Using default file modification detector (uncommitted and unpushed changes).
15:29:10.054 I ----- main : ☕ example.inventory.Inventory
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] ------------------------------------------------------------------------
```

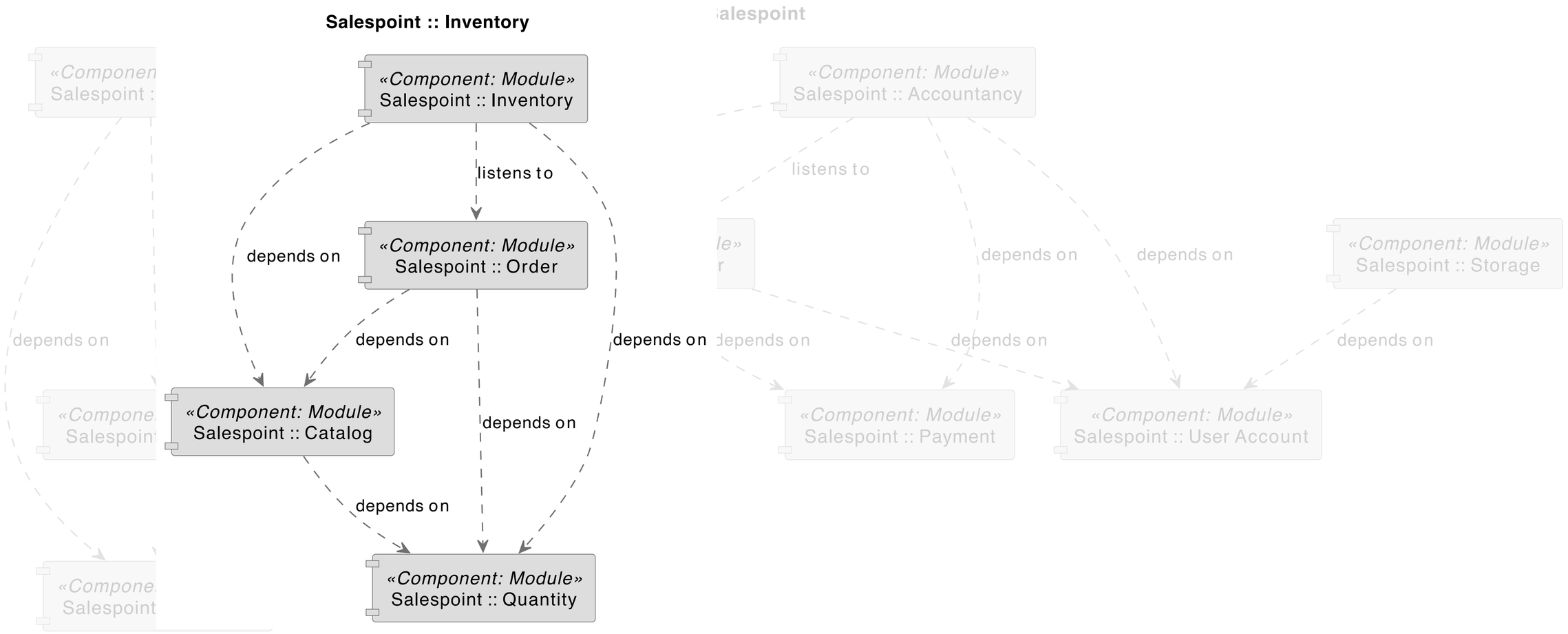
Documentation

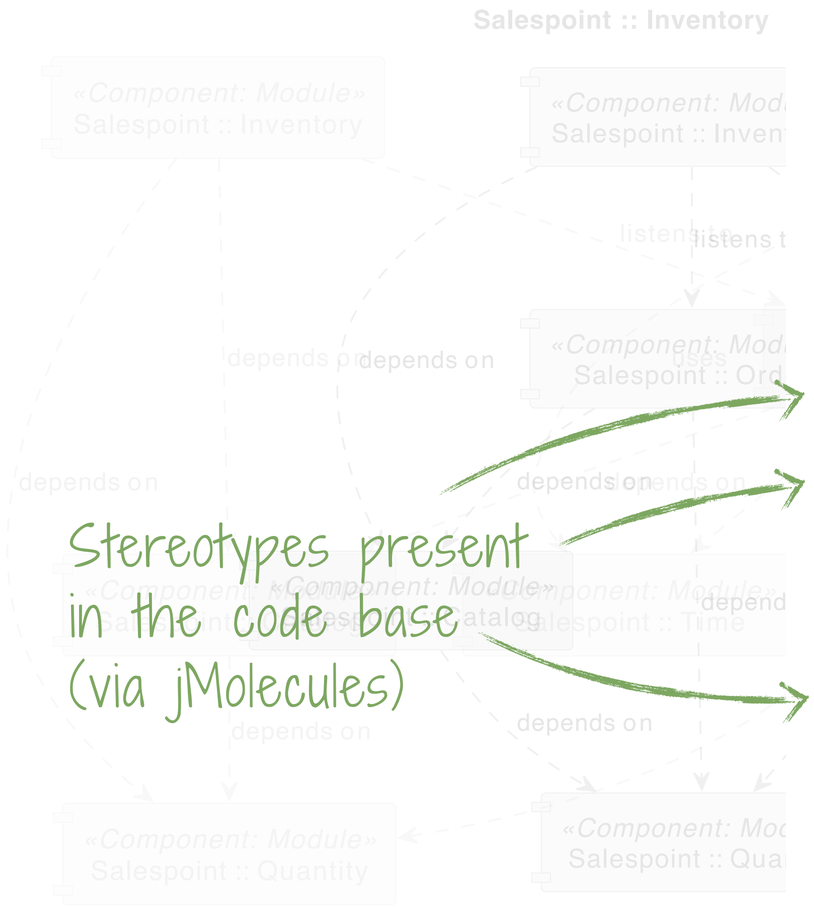


Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation


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

- To strengthen the relationship between architecture and code, find means to represent architectural and design concepts in your codebase.
- Align software engineering practices with architectural abstractions.
- Understand how technology choices affect the overall architecture of the system.
- The scope of change drives decomposition.

Thank you!
Questions?

Oliver Drotbohm odrotbohm oliver.drotbohm@broadcom.com

Resources

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

Links

- **xMolecules**

  Project website

- **jMolecules**

  Project website

- **jMolecules Examples**

  Example project @ GitHub

- **Spring Modulith**

  Project website

- **Archictecturally Evident Spring**

  Example project @ GitHub