

BOTTOM-UP ARCHITECTURE

BRIDGING THE ARCHITECTURE CODE GAP

Oliver Drotbohm

   odrotbohm

 oliver.drotbohm@broadcom.com



Oliver Drotbohm odrotbohm · he/him

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

Edit profile

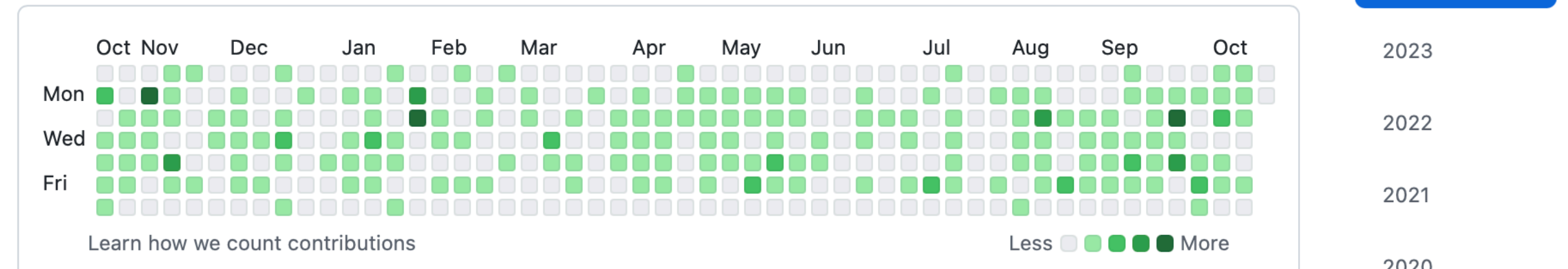
3.7k followers · 32 following

- 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

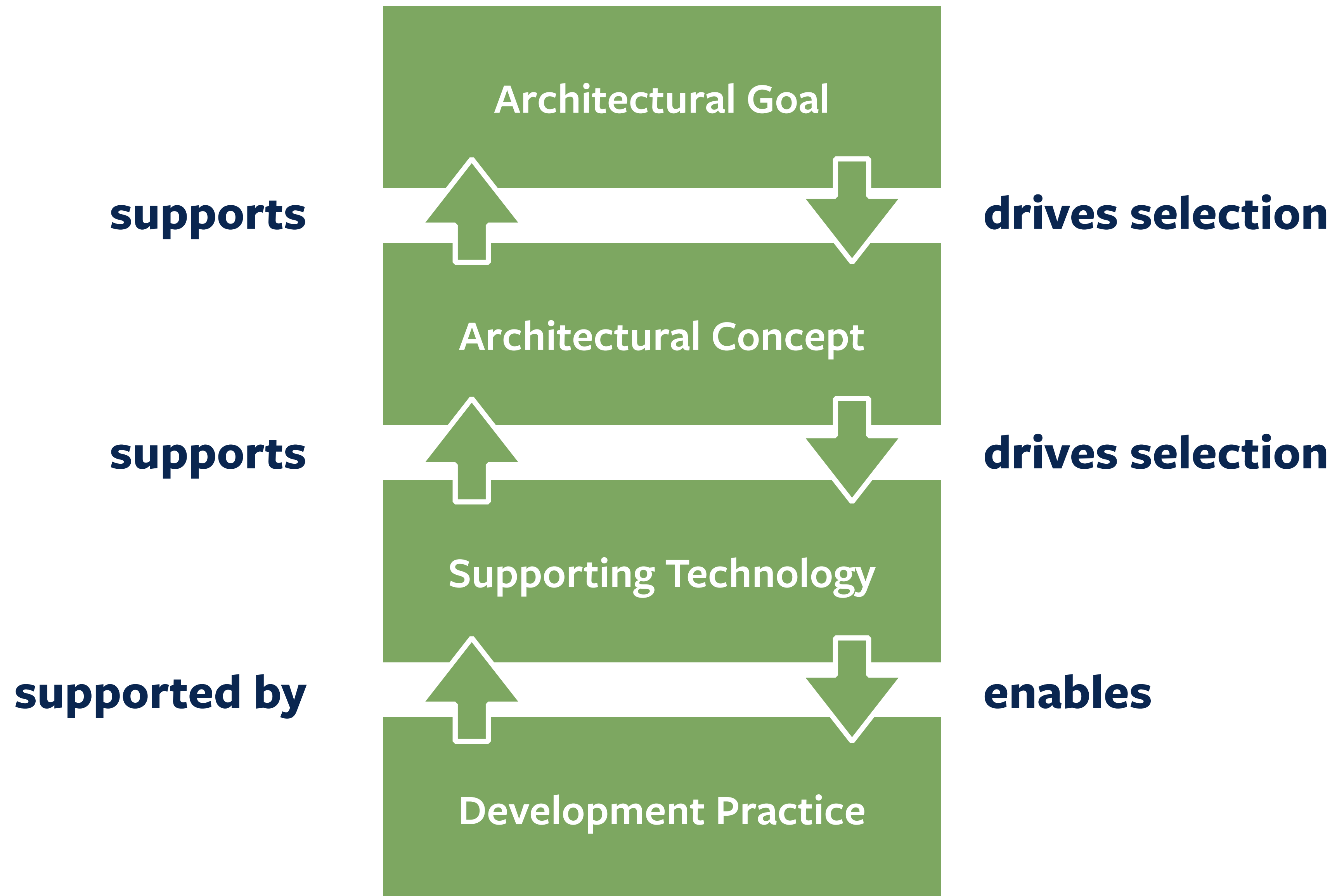
Grid of pinned repositories: spring-projects/spring-modulith, xmolecules/jmolecules, xmolecules/jmolecules-integrations, spring-restbucks, spring-playground, lectures

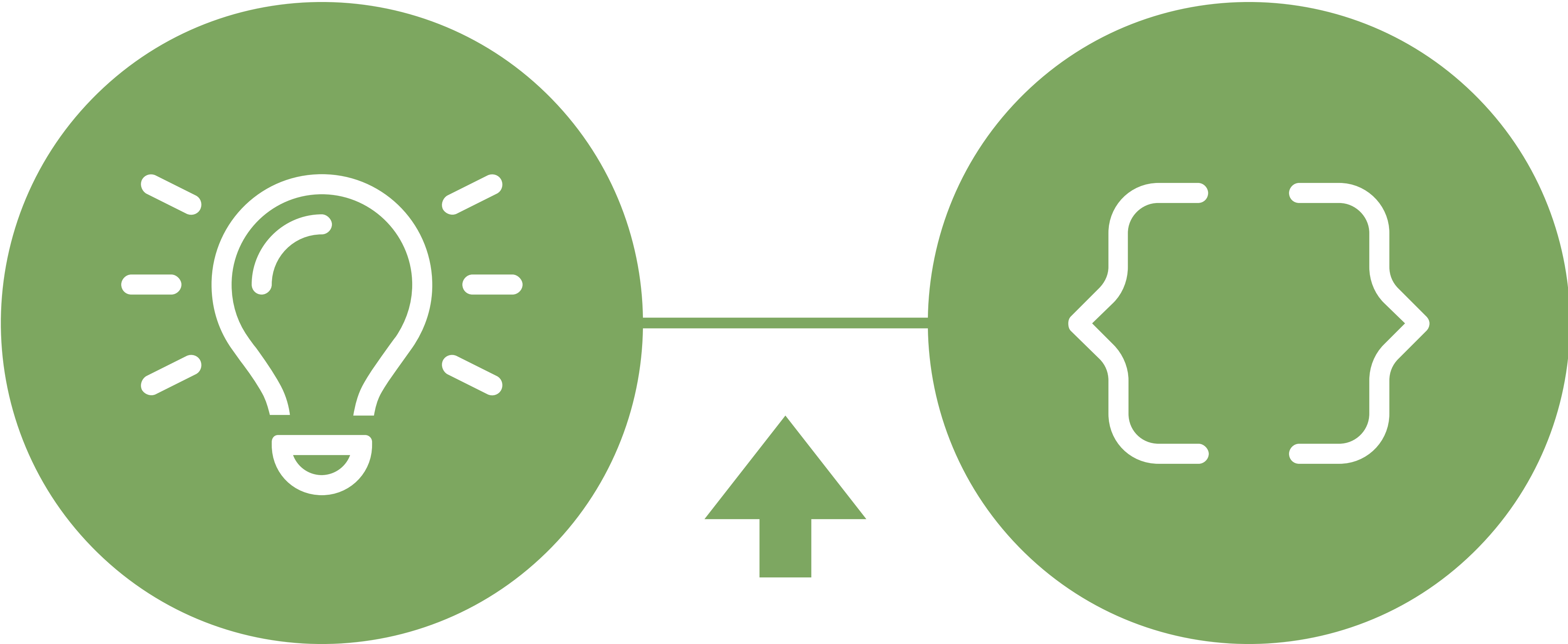
1,982 contributions in the last year Contribution settings 2024



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

Activity overview Code review Contributed to spring-projects/spring-modulith, xmolecules/jmolecules-integrations







“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

Vocabulary

Extensional

Enumerated

**Level of Detail
Encapsulation**

Abstraction

Intensional

Specified

Concepts & Rules

**Pattern
Languages**

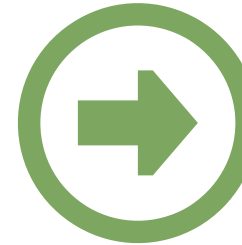
***Architecturally-
Evident Code?***



Extensional

Components / Modules

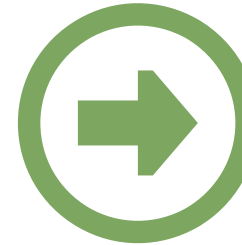
Invoicing,
Shipment



Deployables / Build modules / Packages

Domain language

EmailAddress,
ZipCode



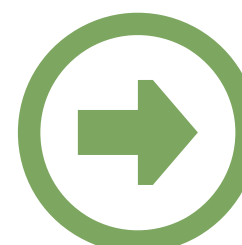
Classes, methods, fields

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

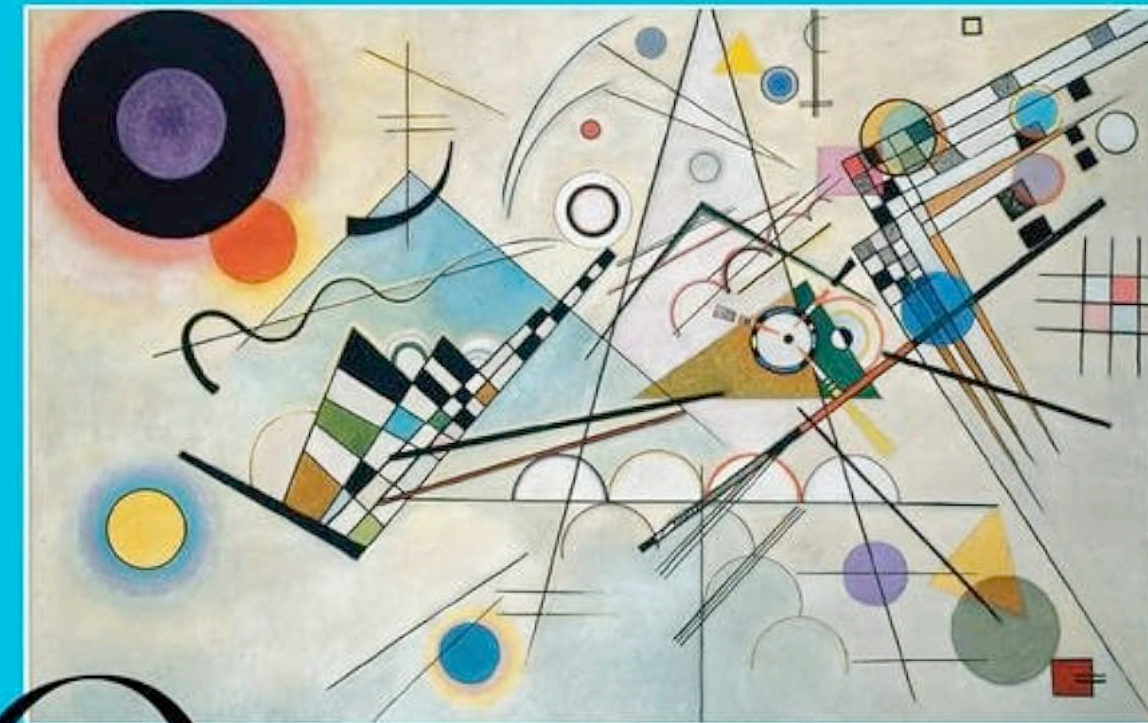
Analyze and Reduce Technical Debt



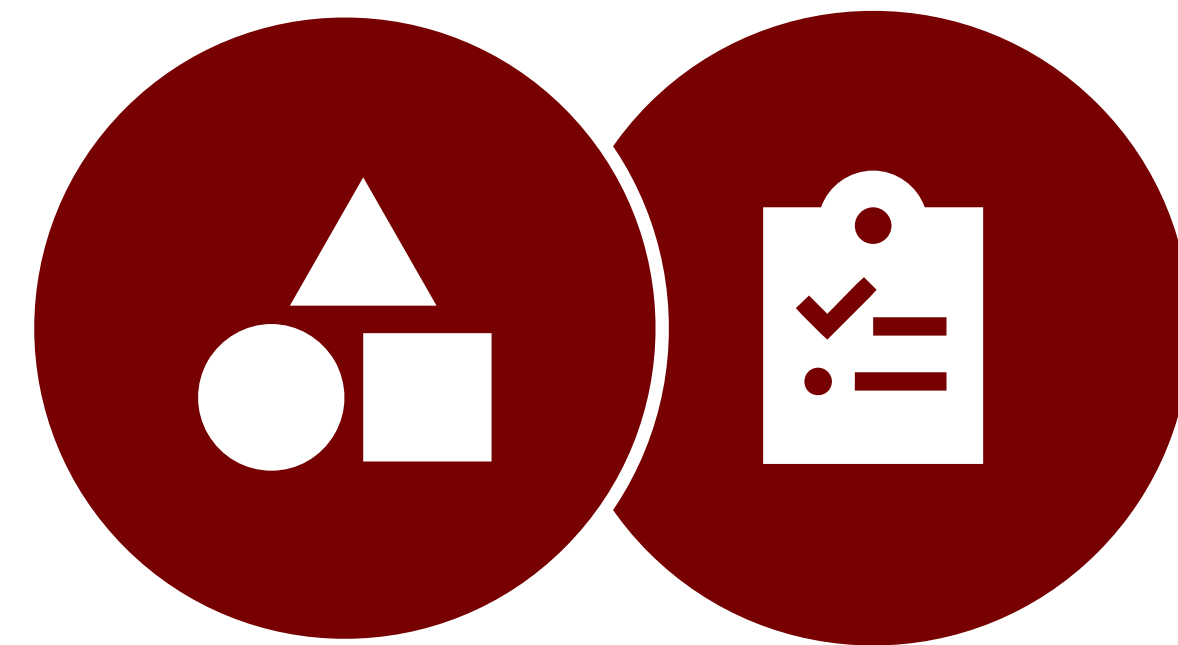
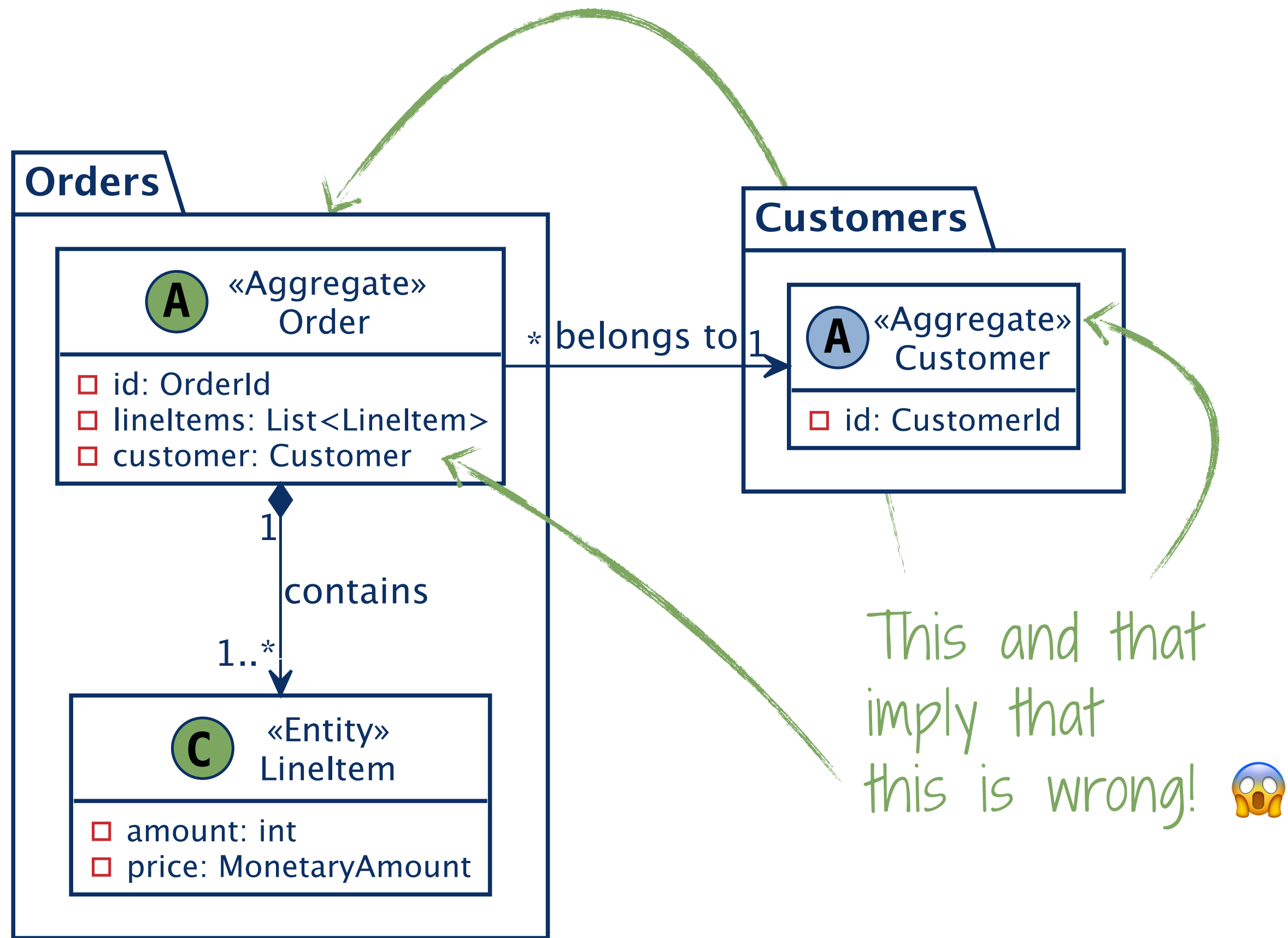
Domain-Driven

DESIGN

Tackling Complexity in the Heart of Software



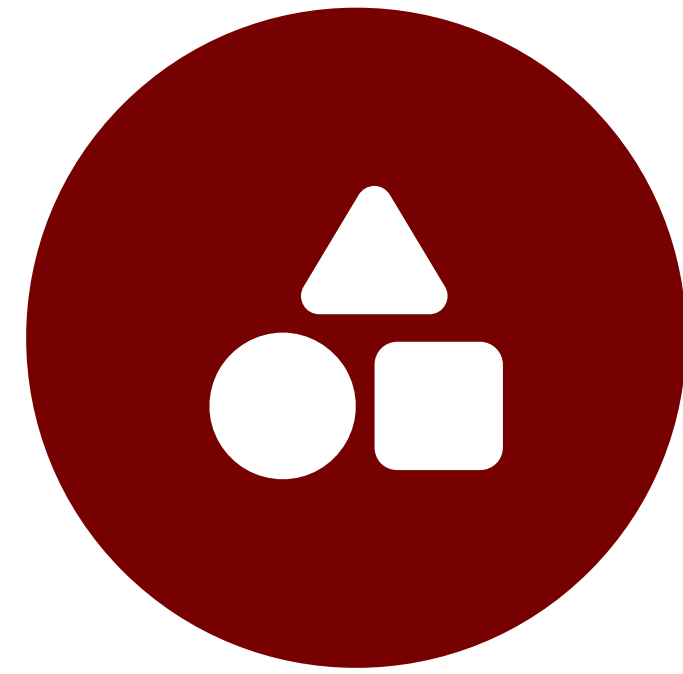
Eric Evans
Foreword by Martin Fowler



A simple Aggregate arrangement



User Code



Concepts



Rules



Frameworks

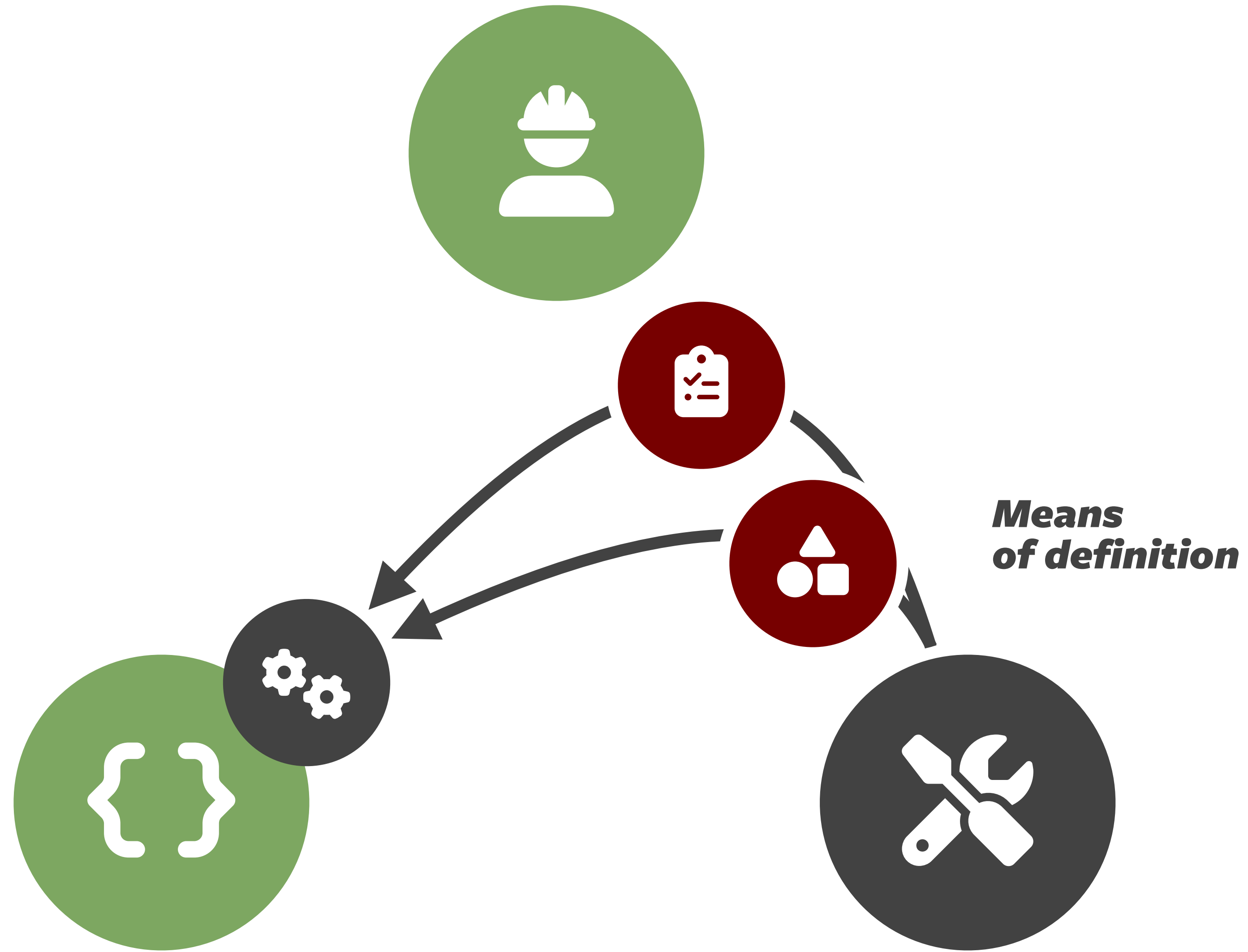


Tools

Code

Architecture

Technology



The logo for ArchUnit features a stylized blue semi-circle on the left, composed of three concentric arcs. To its right, the word "ArchUnit" is written in a blue, sans-serif font. "Arch" is in a darker blue, and "Unit" is in a lighter blue.

ArchUnit

The logo for jQAssistant features the word "jQAssistant" in a bold, black, sans-serif font. A green checkmark is positioned over the "Q" and "A" characters. The "j" is in a lighter green color.

jQAssistant

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
```

Establishes the concept

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

Establishes the rule

Reference to
tech stack 😞



Establishing an Aggregate... in ArchUnit

```
@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))
```

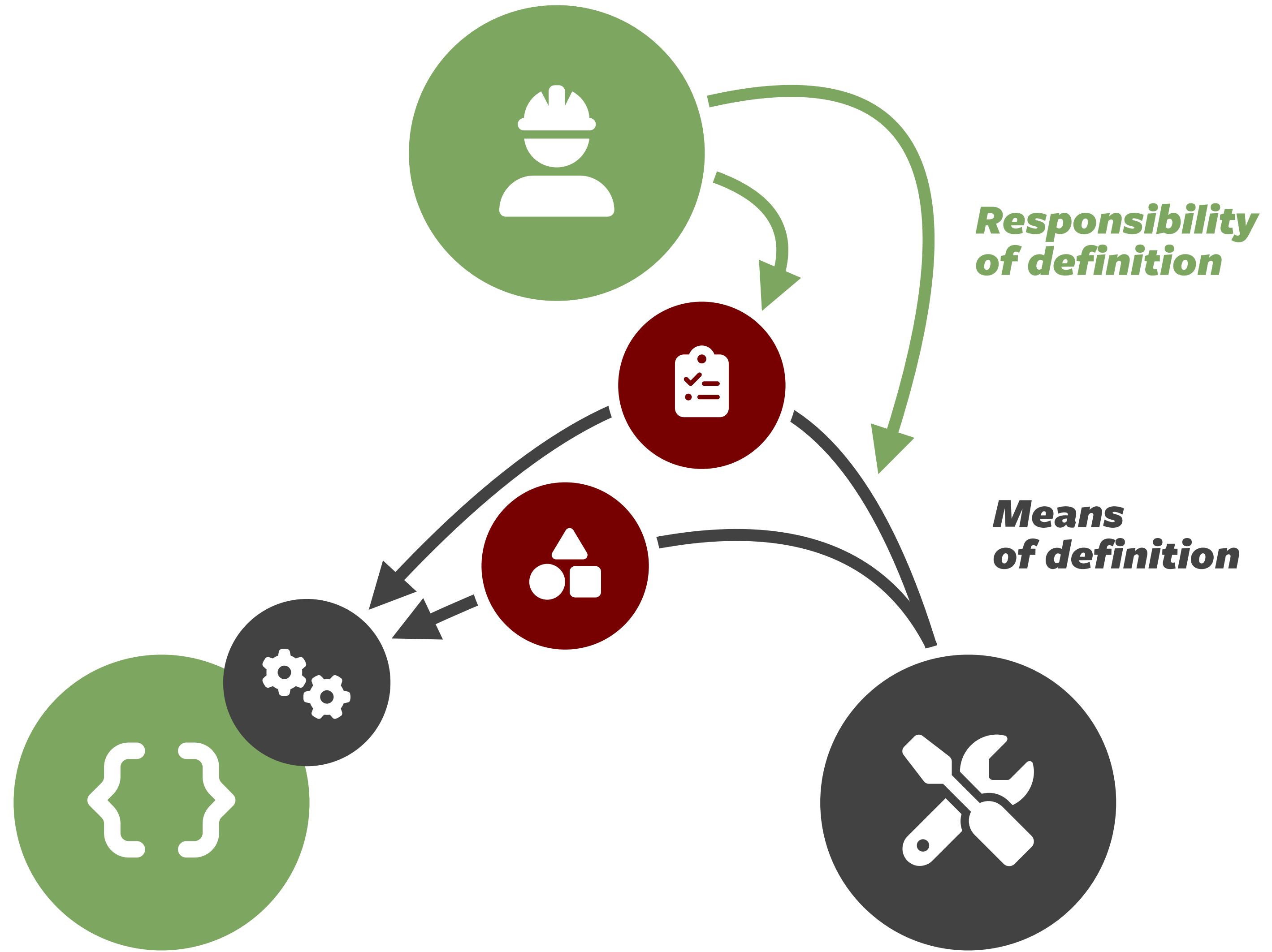
```
            .check(types);
```

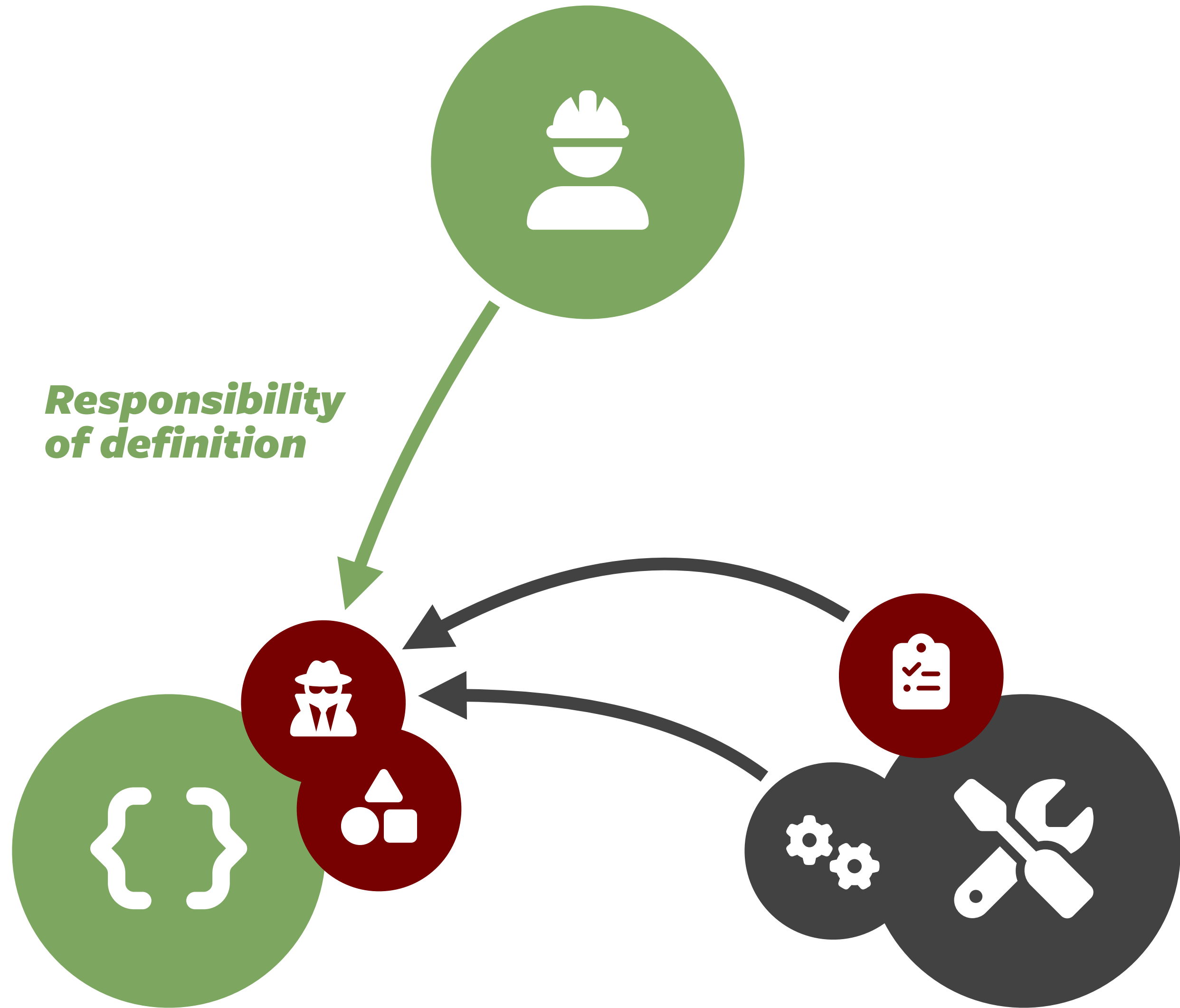
```
    }
```

```
}
```

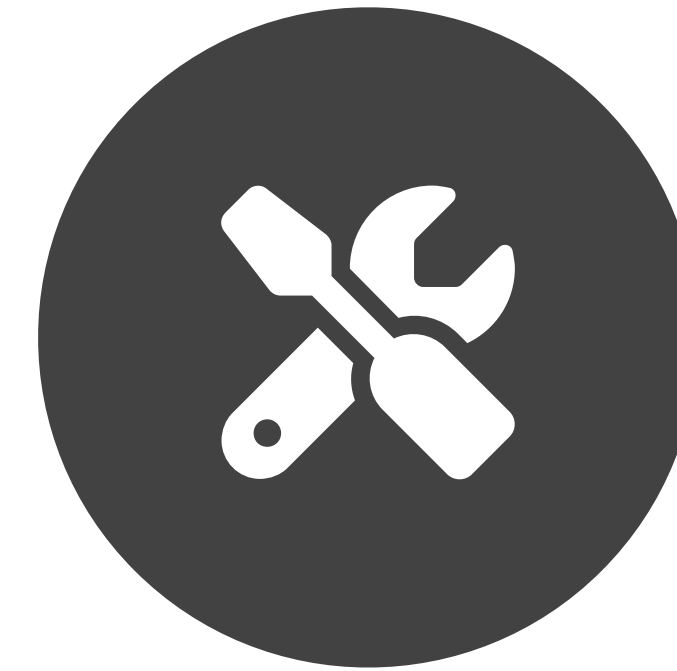
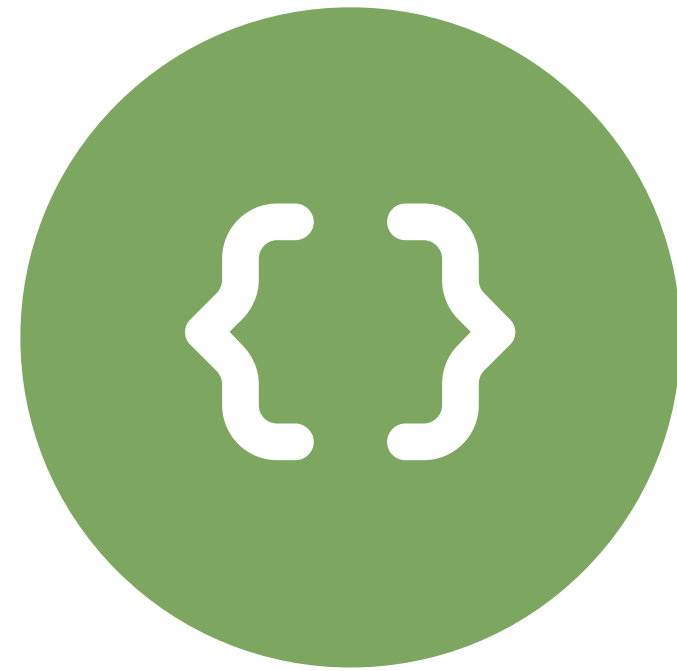
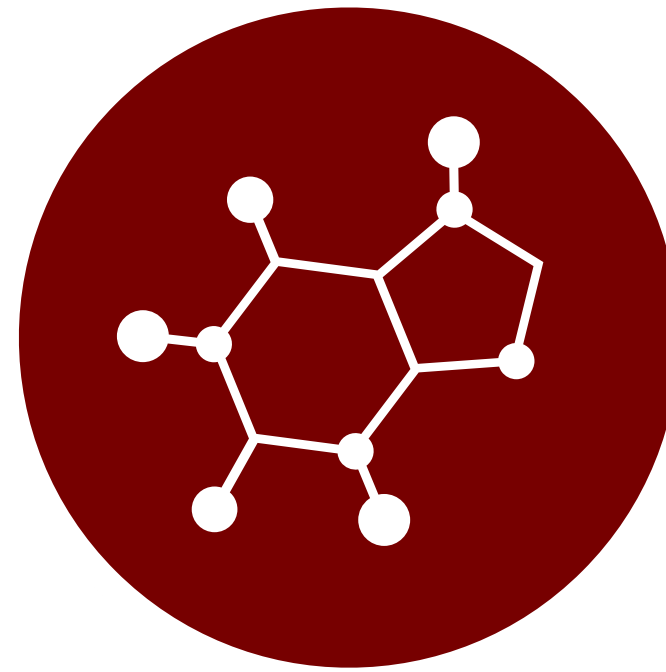
Establishes
the concept

Establishes
the rule





Responsibility of definition





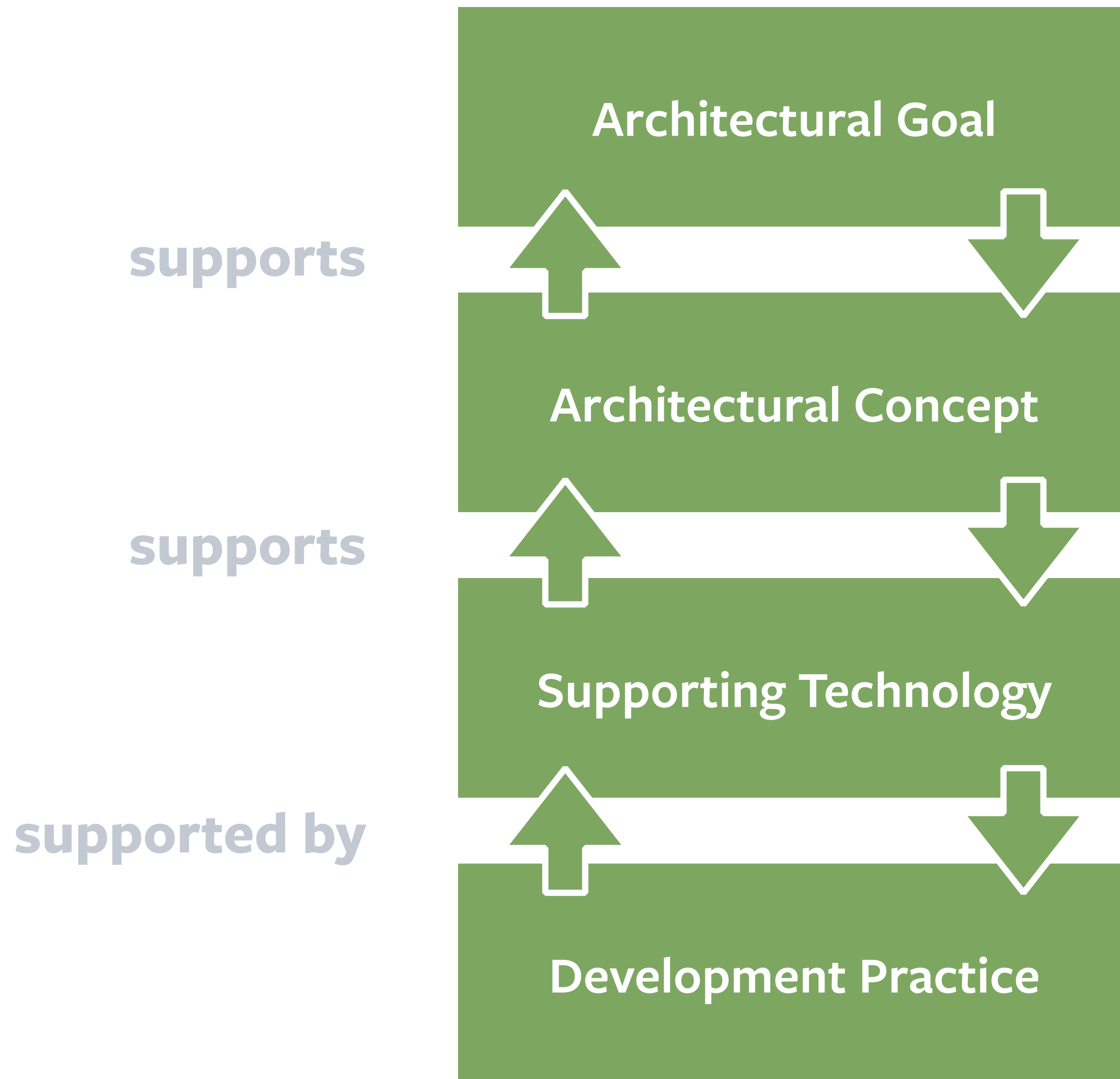
xMolecules



php







Understandability

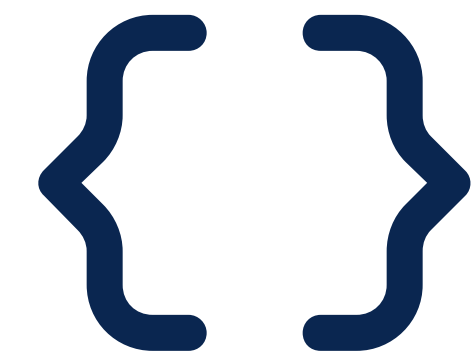
drives selection

DDD Building Blocks

drives selection

jMolecules

enables



Explicit concepts


```

@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

```
@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

```
*Order.java X
arch-evident-spring > src/main/java > example.order > Order >
45 @Getter
46 public class Order implements AggregateRoot<Order, OrderIdentifier> {
47
48     private final OrderIdentifier id;
49     private final Customer customer;
50     private Status status;
51
52     private final List<LineItem> lineItems = new ArrayList<>();
53
54     public Order(CustomerIdentifier customerId) {
55
56         this.id = new OrderIdentifier(UUID.randomUUID());
57         this.status = Status.OPEN;
58         this.customer = null;
59     }
```

Problems X Javadoc Error Log Progress Search PlantUML Call Hierarchy Coverage JUnit Boot Dashboard Terminal History Console

3 errors, 0 warnings, 0 others

| Description | Resource | Path |
|--|------------|---|
| Invalid aggregate root reference! Use identifier reference or Association instead! | Order.java | /arch-evident-spring/src/main/java/exam |



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



Eliminate Boilerplate

Model characteristics
expressed implicitly
or through
technical means

```
@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;
    }
}
```

JPA-induced
boilerplate

```

@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;
    }
}

```



```
@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.OneToOne(...).
[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;
    }
}

```

```

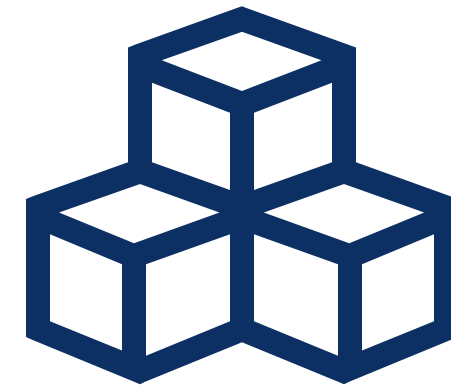
@Table(name = "SAMPLE_ORDER")
@Getter
public class Order implements AggregateRoot<Order, OrderId> {

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

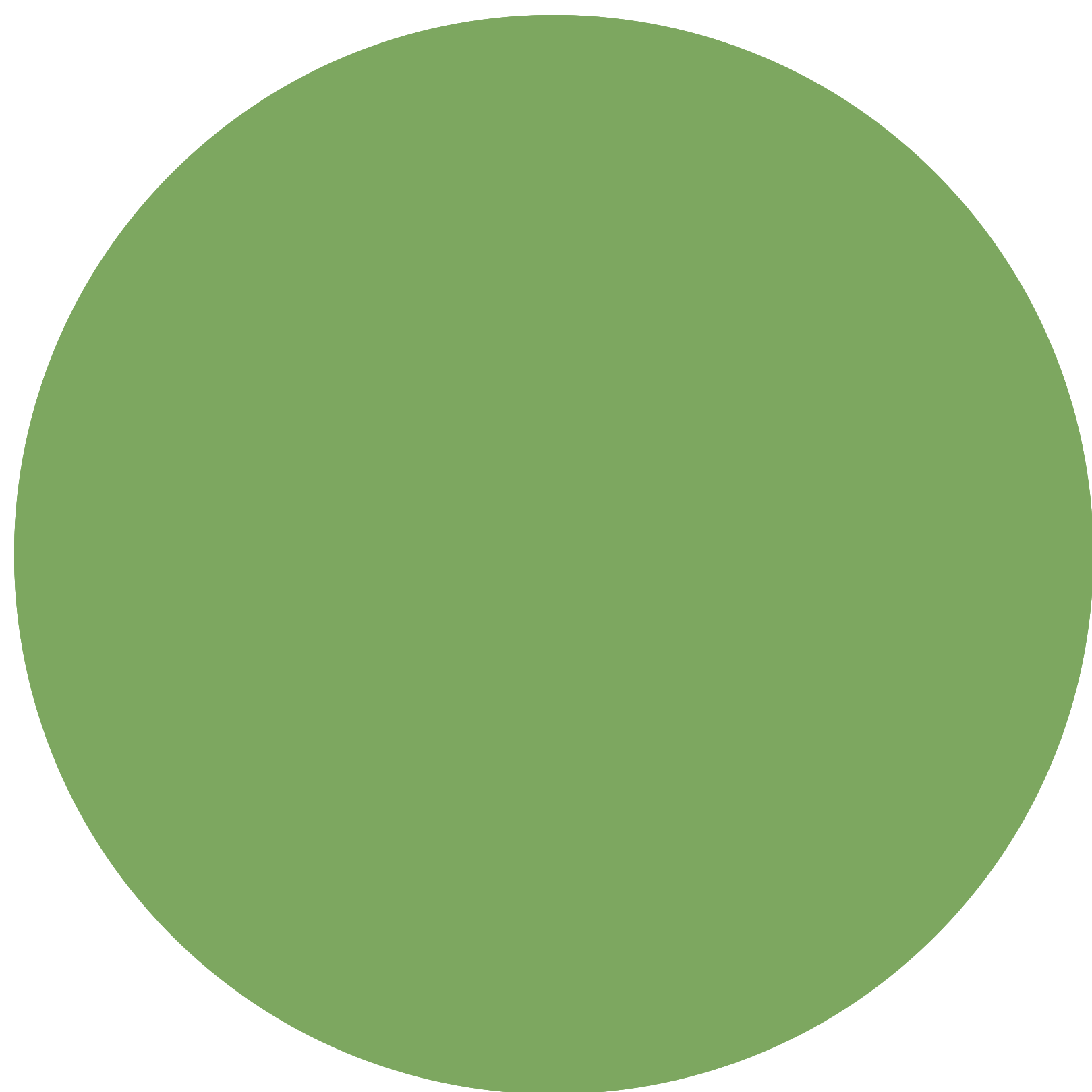
    public Order(CustomerId customerId) {
        this.id = OrderId.of(UUID.randomUUID());
        this.customer = Association.forId(customerId);
    }

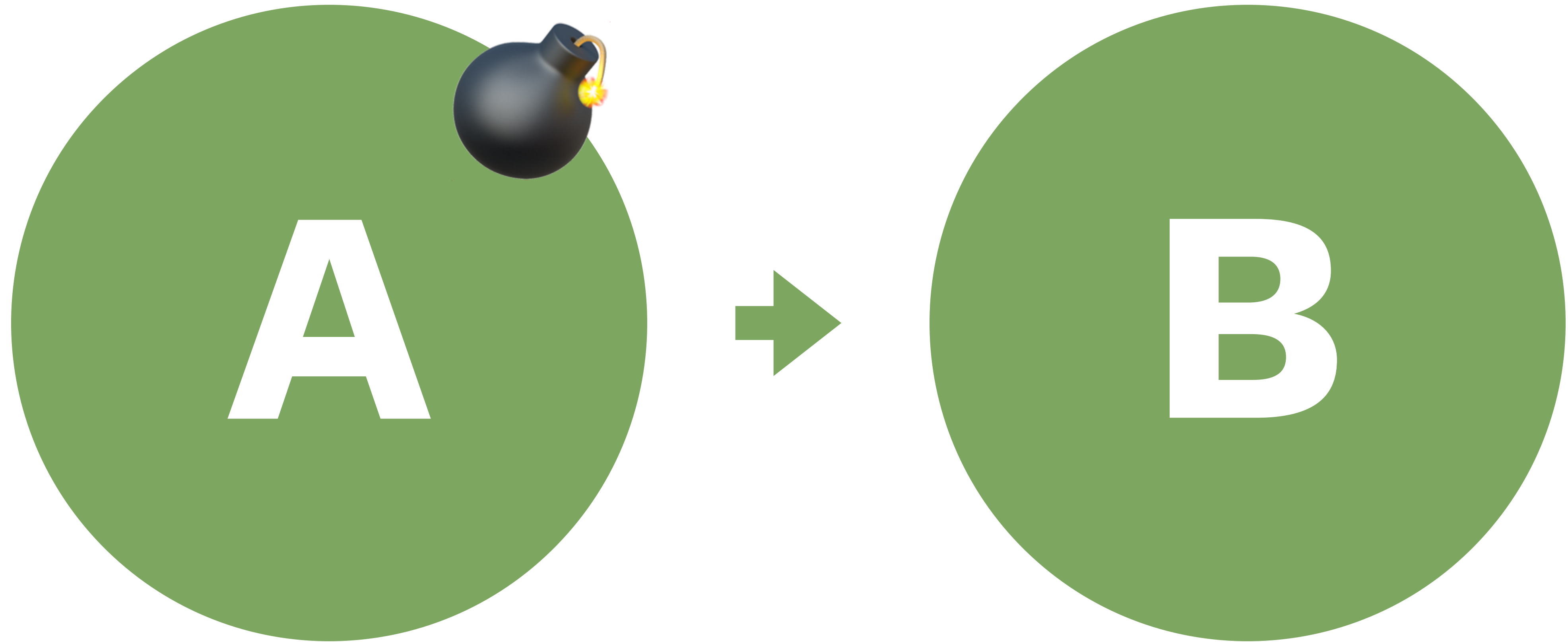
    record OrderId(UUID orderId) implements Identifier {}
}

```

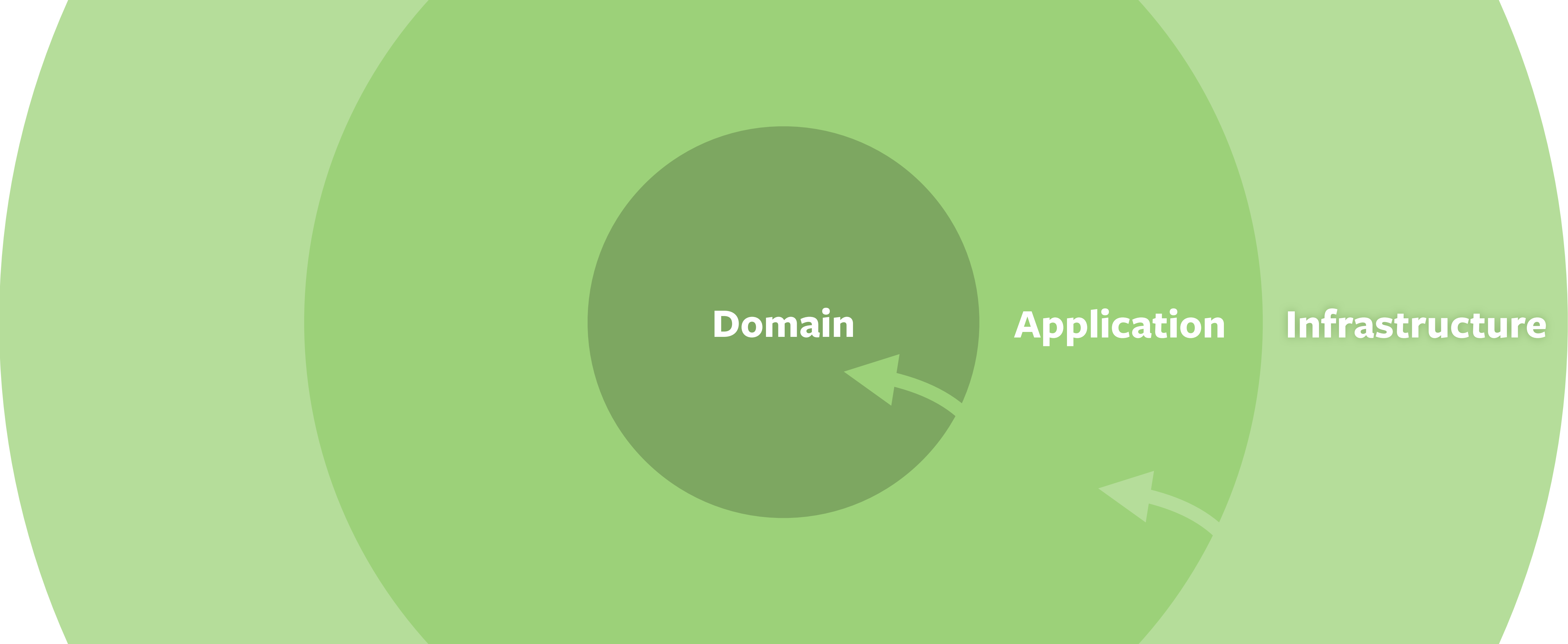


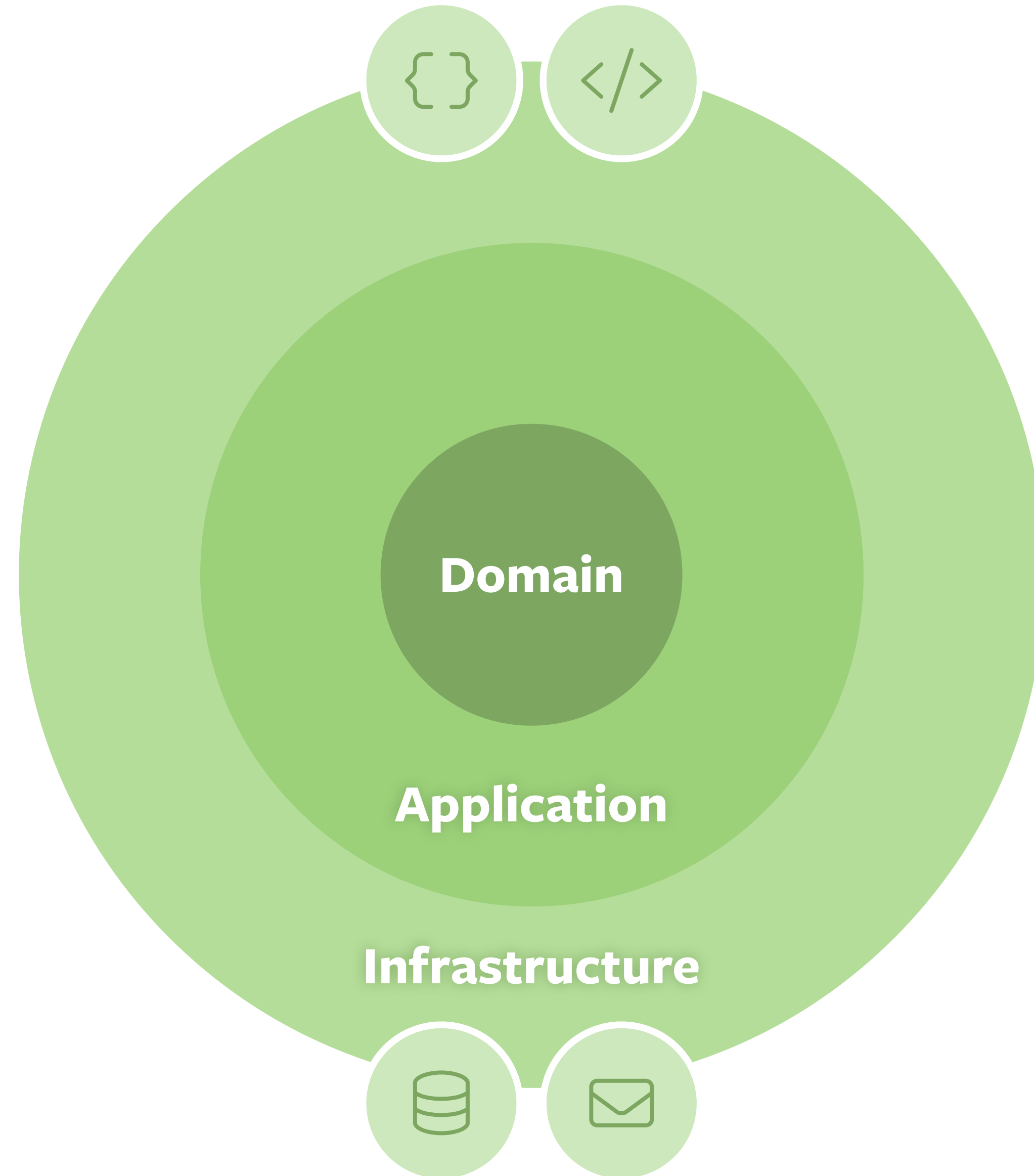
Decomposition

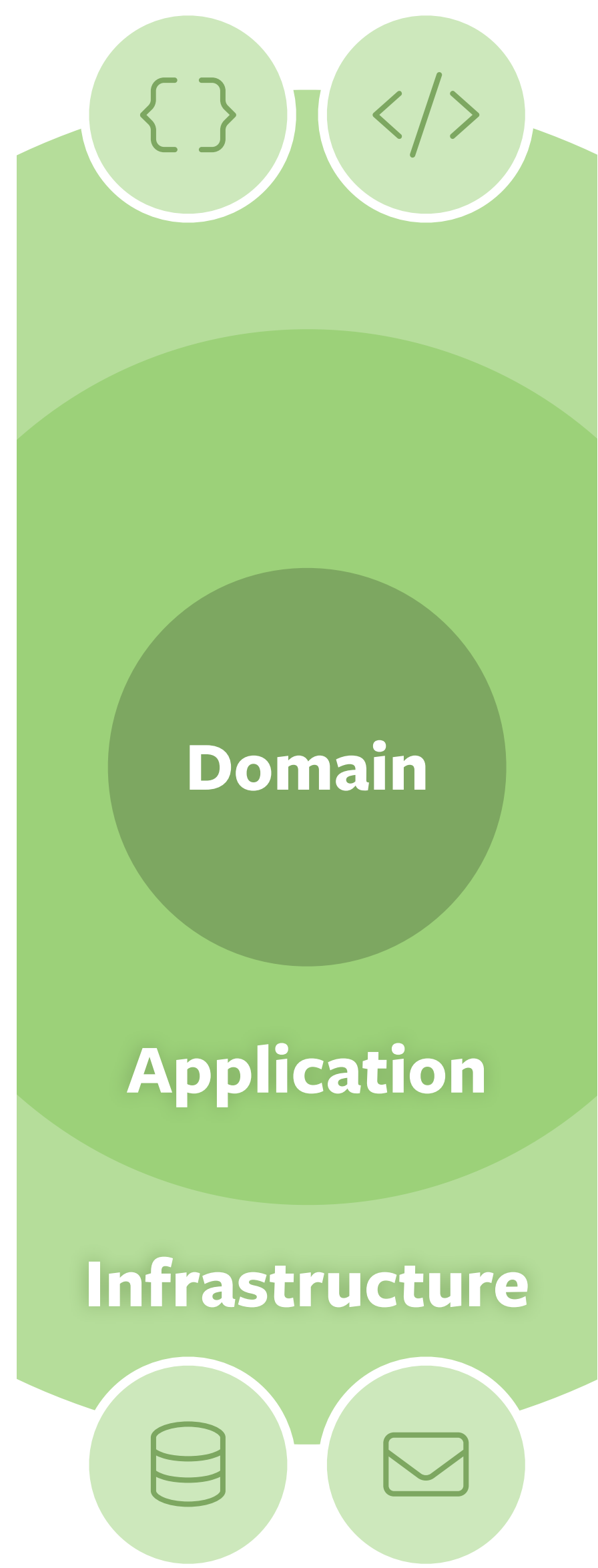


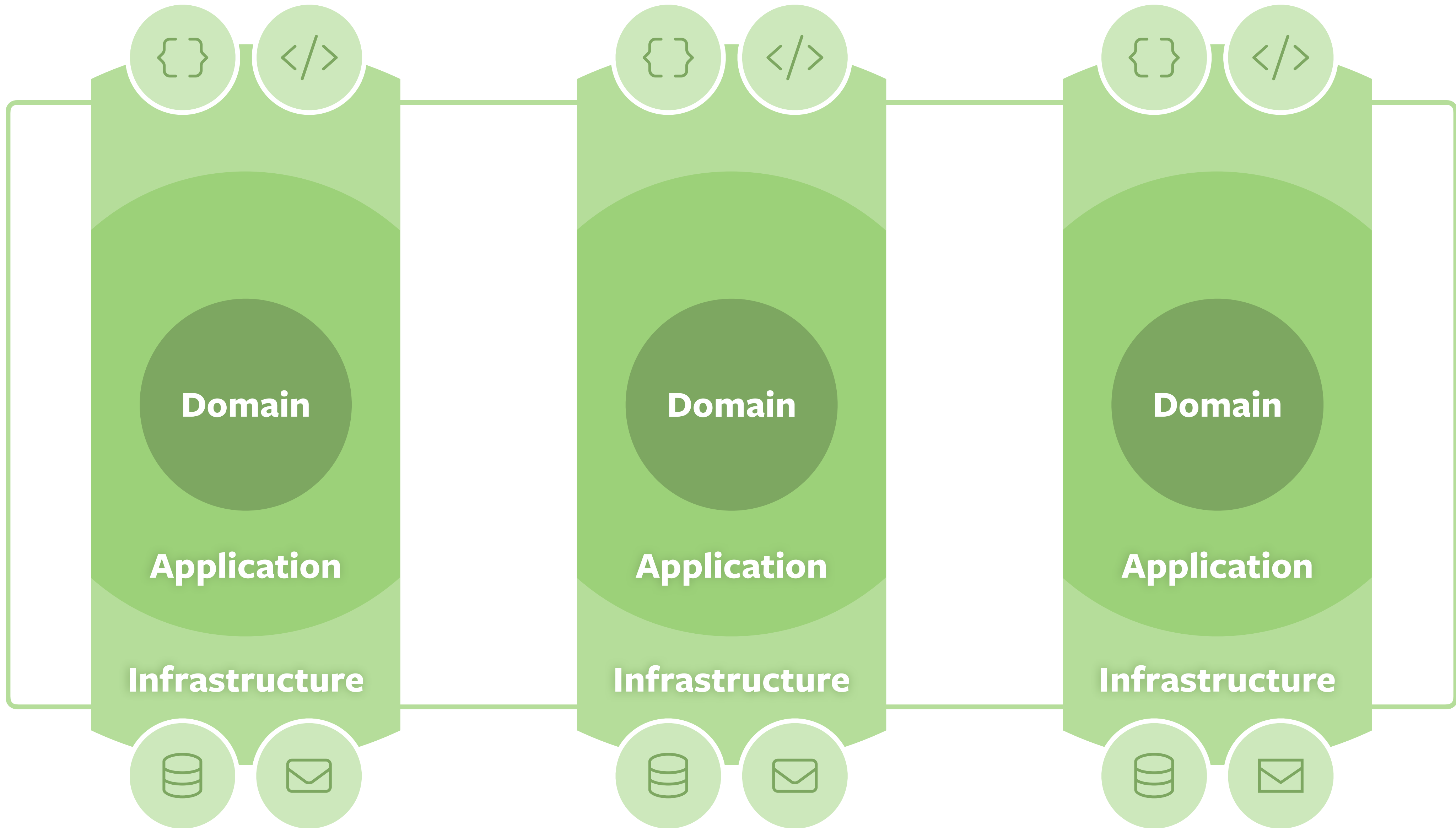


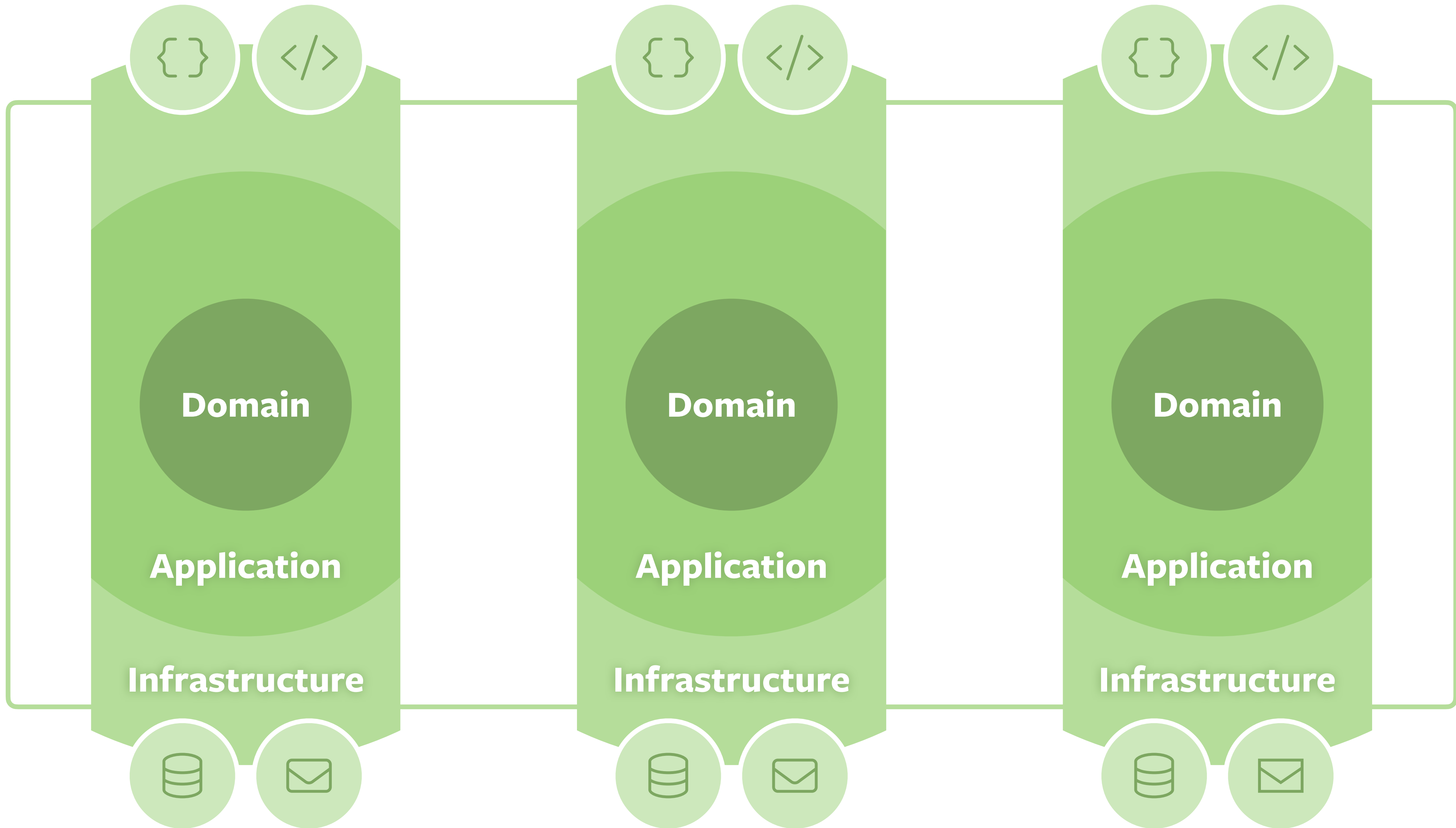
Risk of Change
Scope of Change?

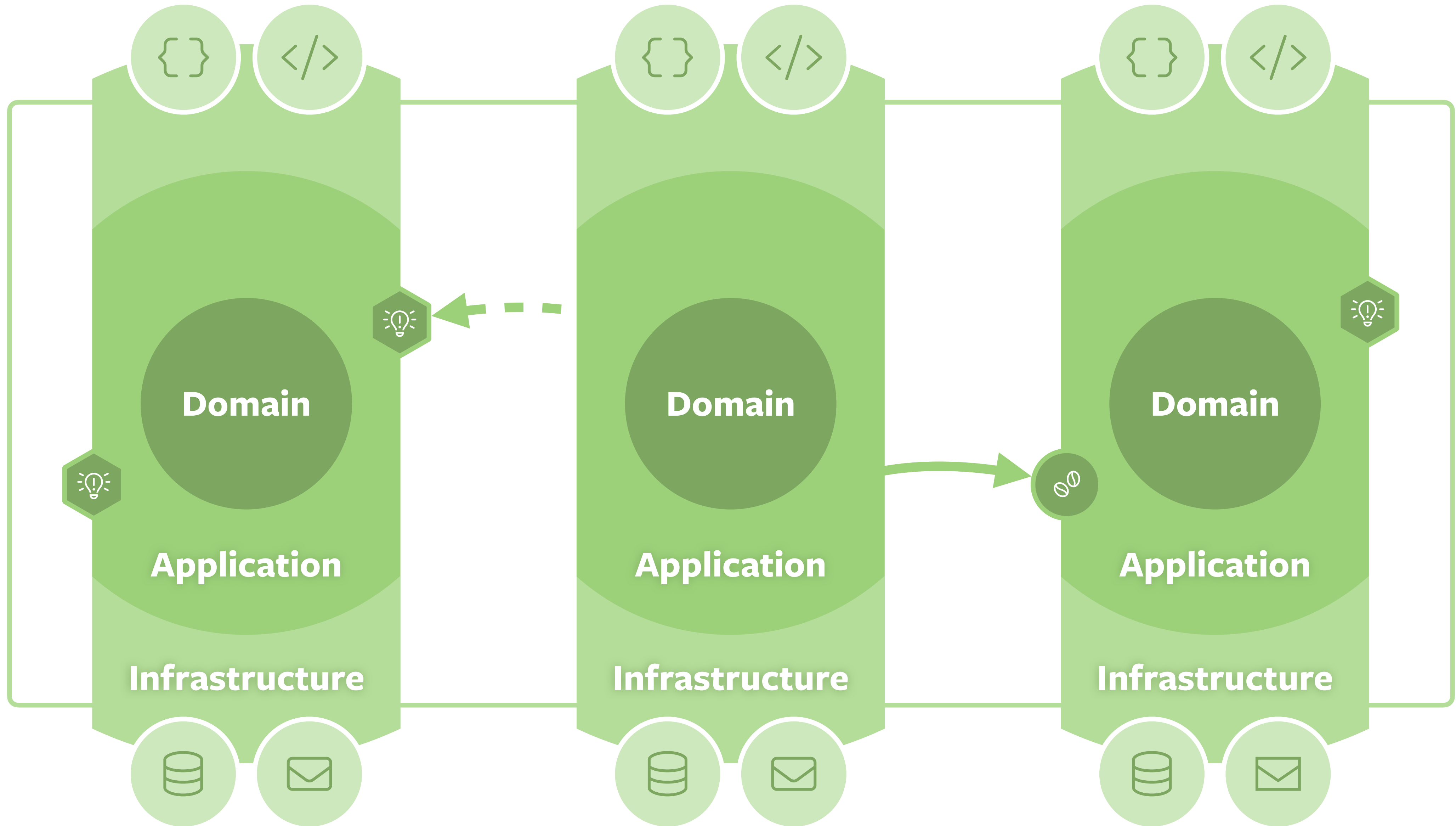


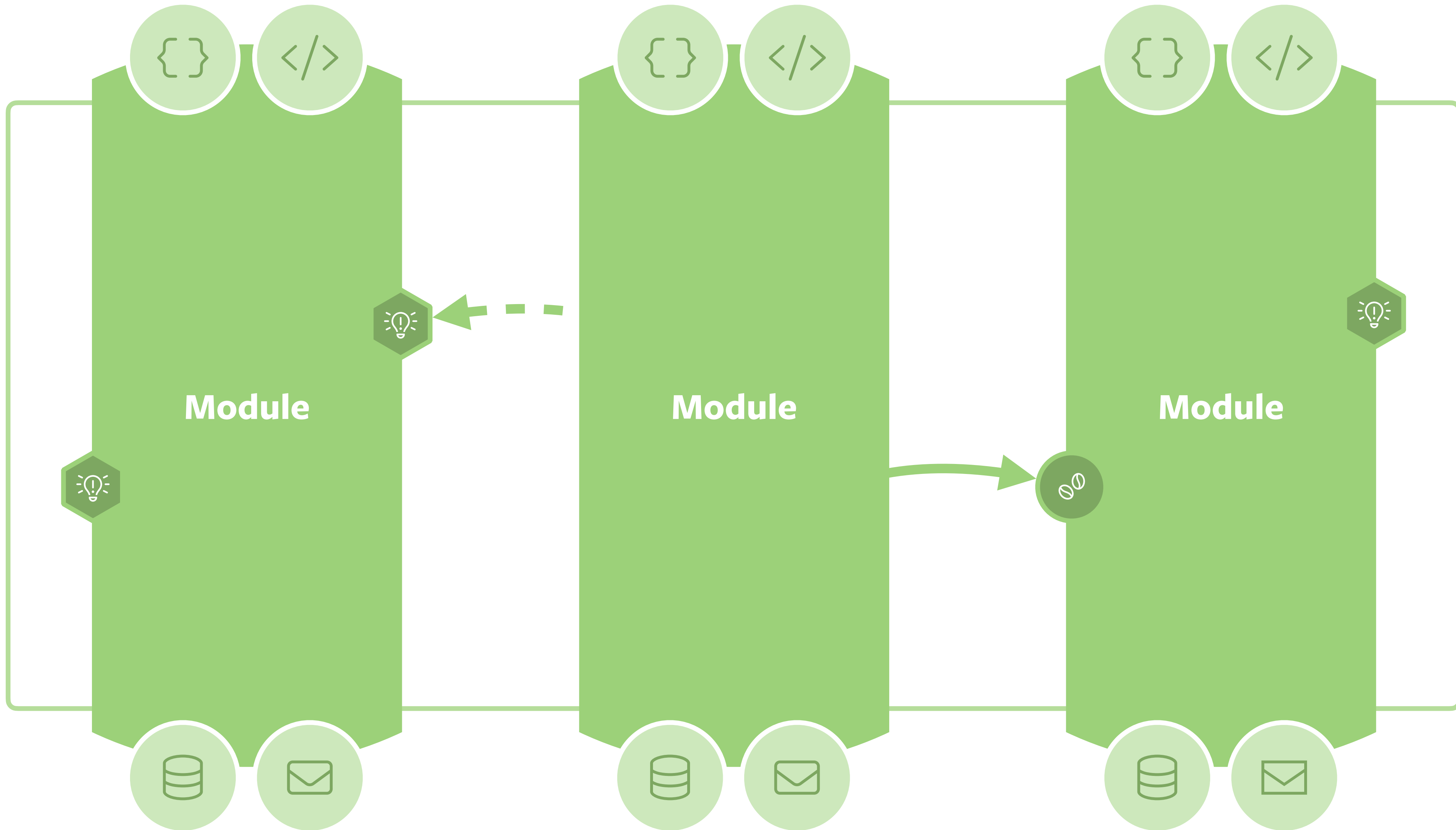


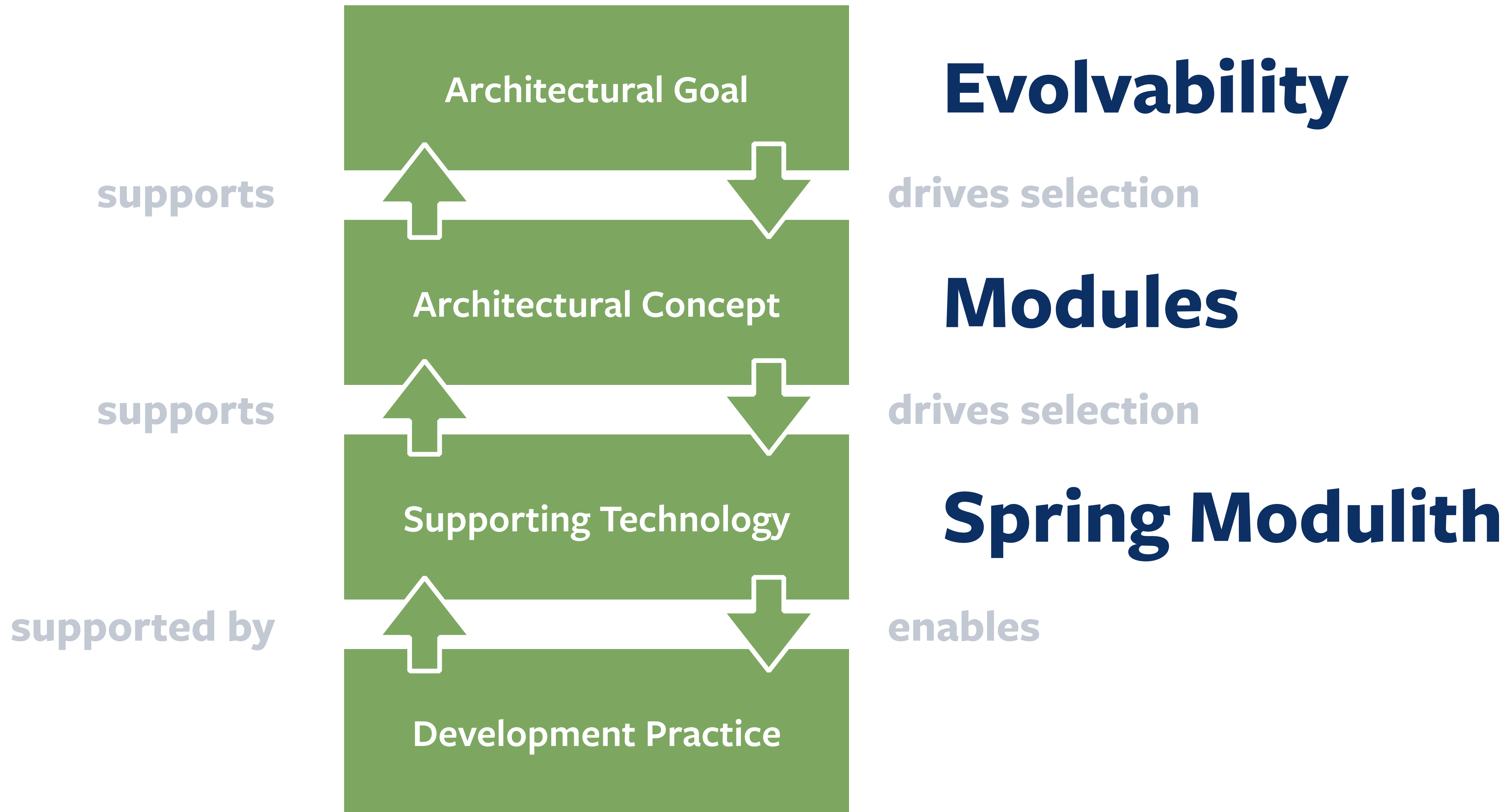














Spring Modulith

```
package com.acme.modulith
```

```
@SpringBootApplication
```

```
class MyApplication { ... }
```

Standard Spring Boot Application

Package Conventions

 **...modulith**

 **...modulith.moduleA**

 **...modulith.moduleA.internal**

 **...modulith.moduleB**

 **...modulith.moduleB.internal**

API packages

Access to components
residing in internal packages
forbidden and checked
during tests.



Verification

```
package com.acme.modulith
```

```
@SpringBootApplication
```

```
class MyApplication { ... }
```

Standard Spring Boot Application

```
var modules =
```

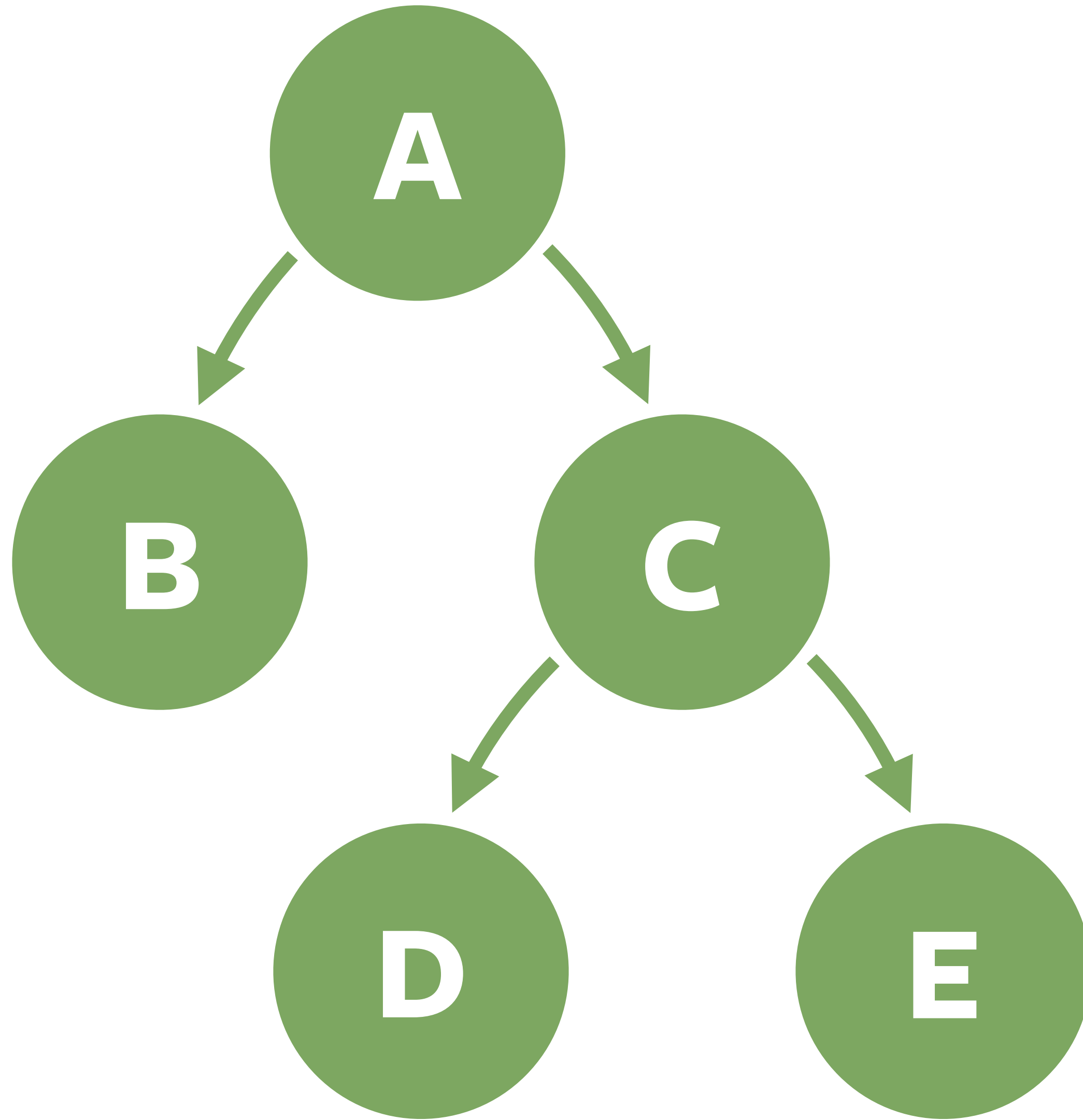
```
    ApplicationModules.of(MyApplication.class);
```

```
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

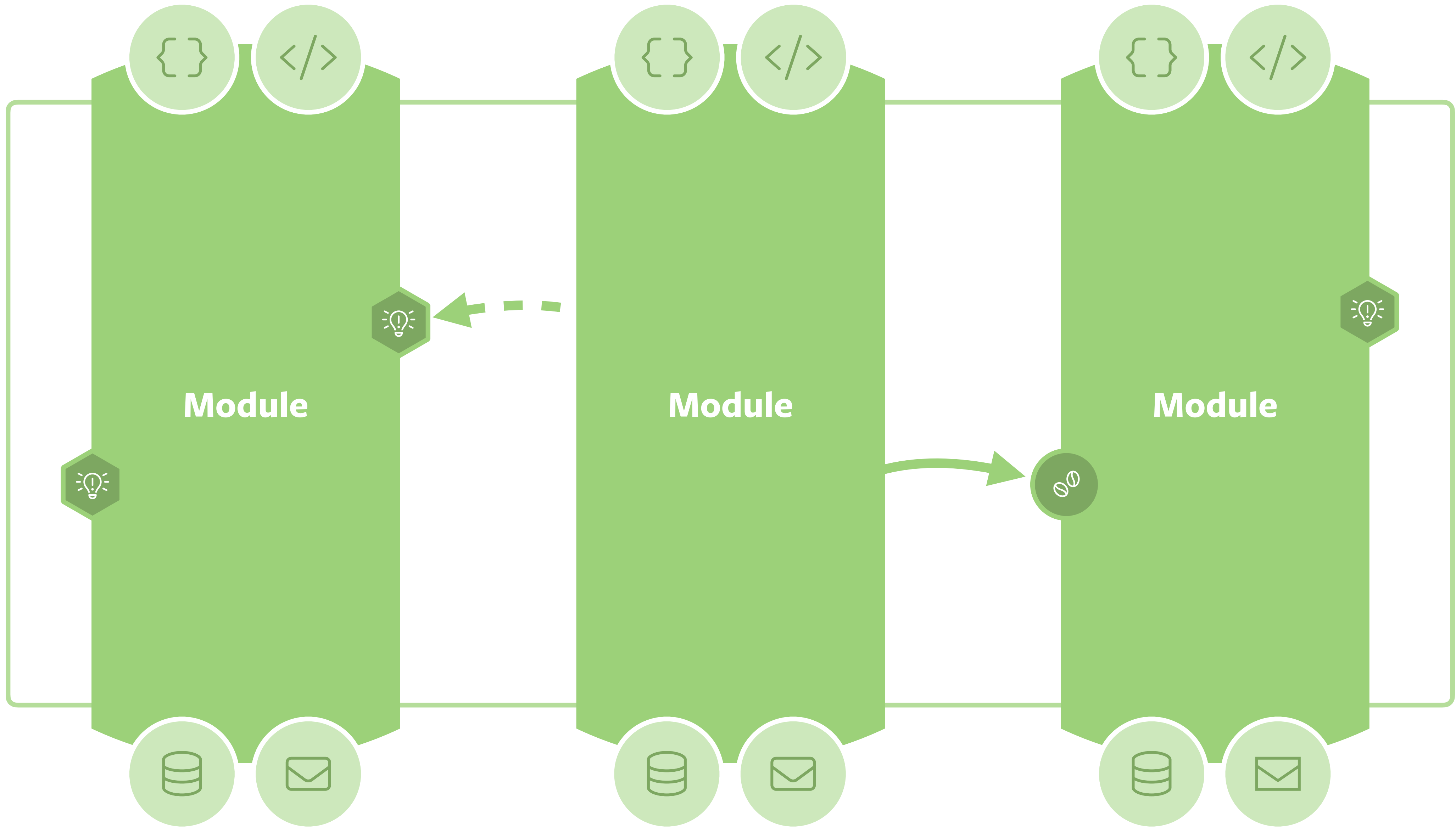
| | Module A | Module B | Module C |
|-----------------------------------|-----------------|-----------------|-----------------|
| Web @WebMvcTest | | | |
| Business logic | | | |
| Data access @Data..Test | | | |

| | Module A | Module B | Module C |
|-----------------------------------|-----------------|-----------------|-----------------|
| Web @WebMvcTest | | | |
| Business logic | | | |
| Data access @Data..Test | | | |

@ApplicationModuleTest

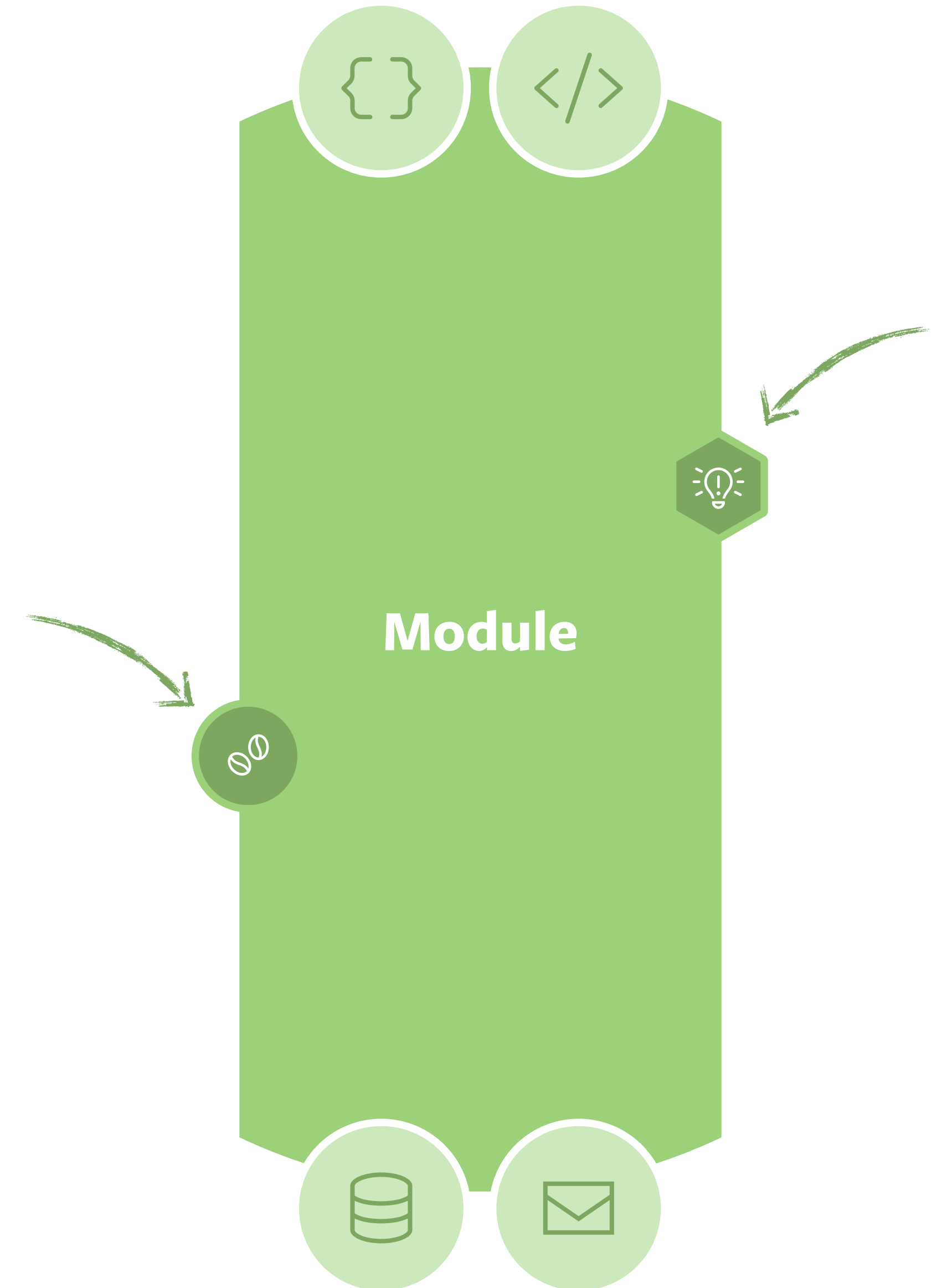


Subcutaneous Tests



Stimulate the module by...

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



```
@ApplicationModuleTest
```

← Test scoped to the module

```
@RequiredArgsConstructor
```

```
class OrderIntegrationTests {
```

```
    private final OrderManagement orders;
```

← Public API primary interaction target

```
    @Test
```

```
    void completionCausesEventPublished(Scenario scenario) {
```

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

Given
When

```
        scenario.stimulate(() → orders.complete(order))
```

← Stimulate module

Then

```
        .andWaitForEventOfType(OrderCompleted.class)
```

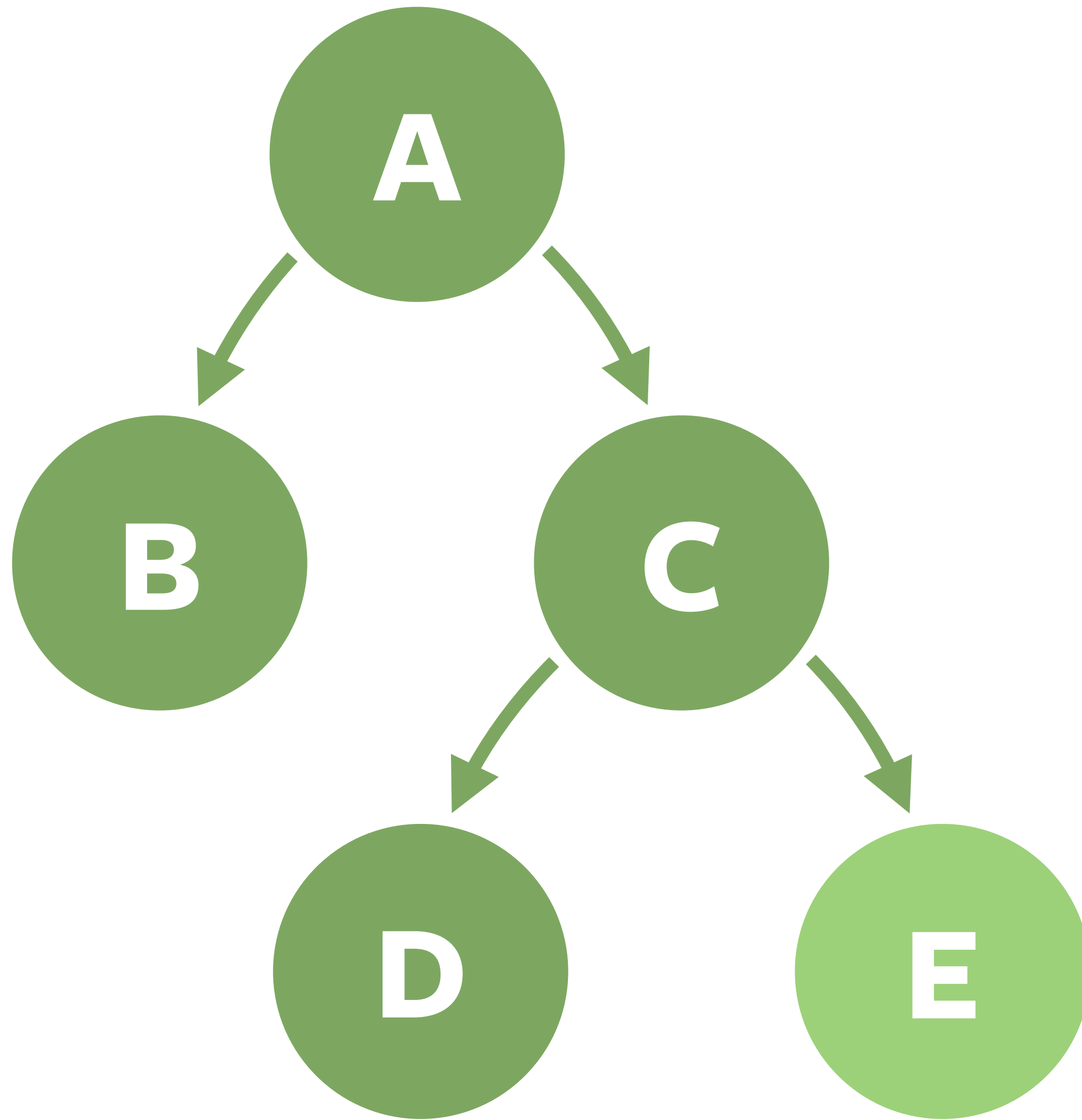
```
        .matchingMappedValue(OrderCompleted::id, order.getId())
```

```
        .toArrive();
```

```
    }
```

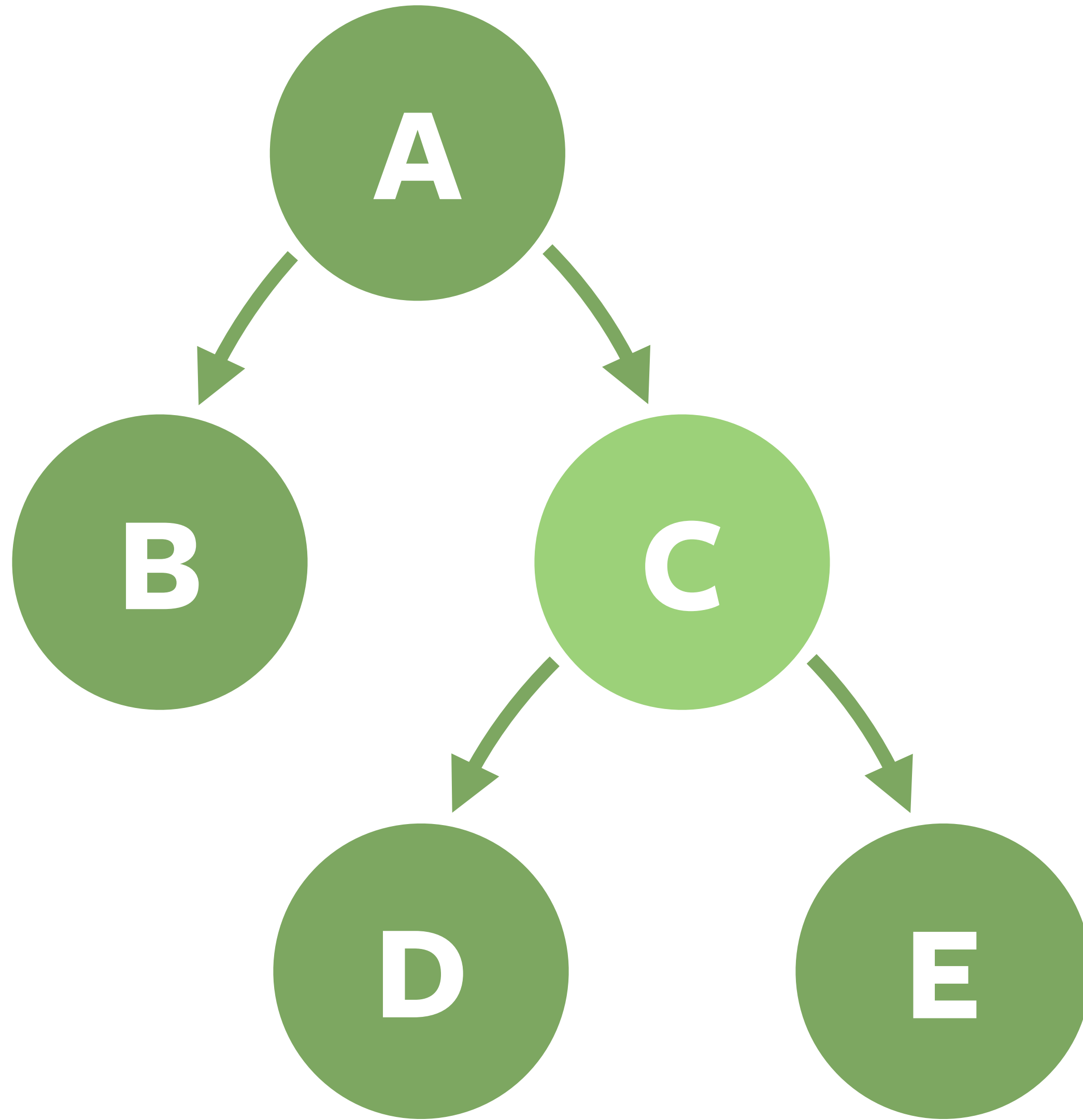
```
}
```

Scope of Individual Tests



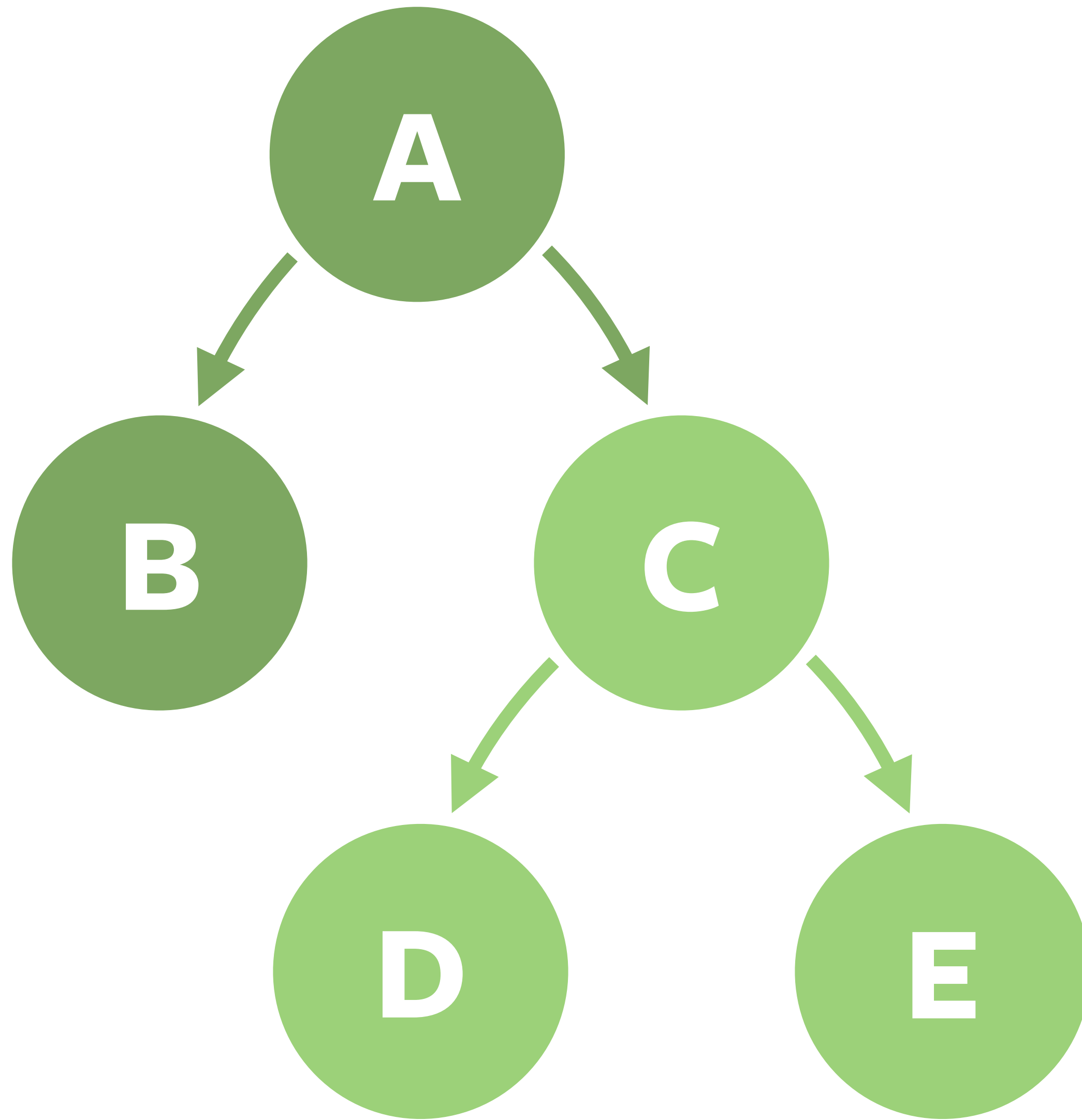
Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation



Unit of...

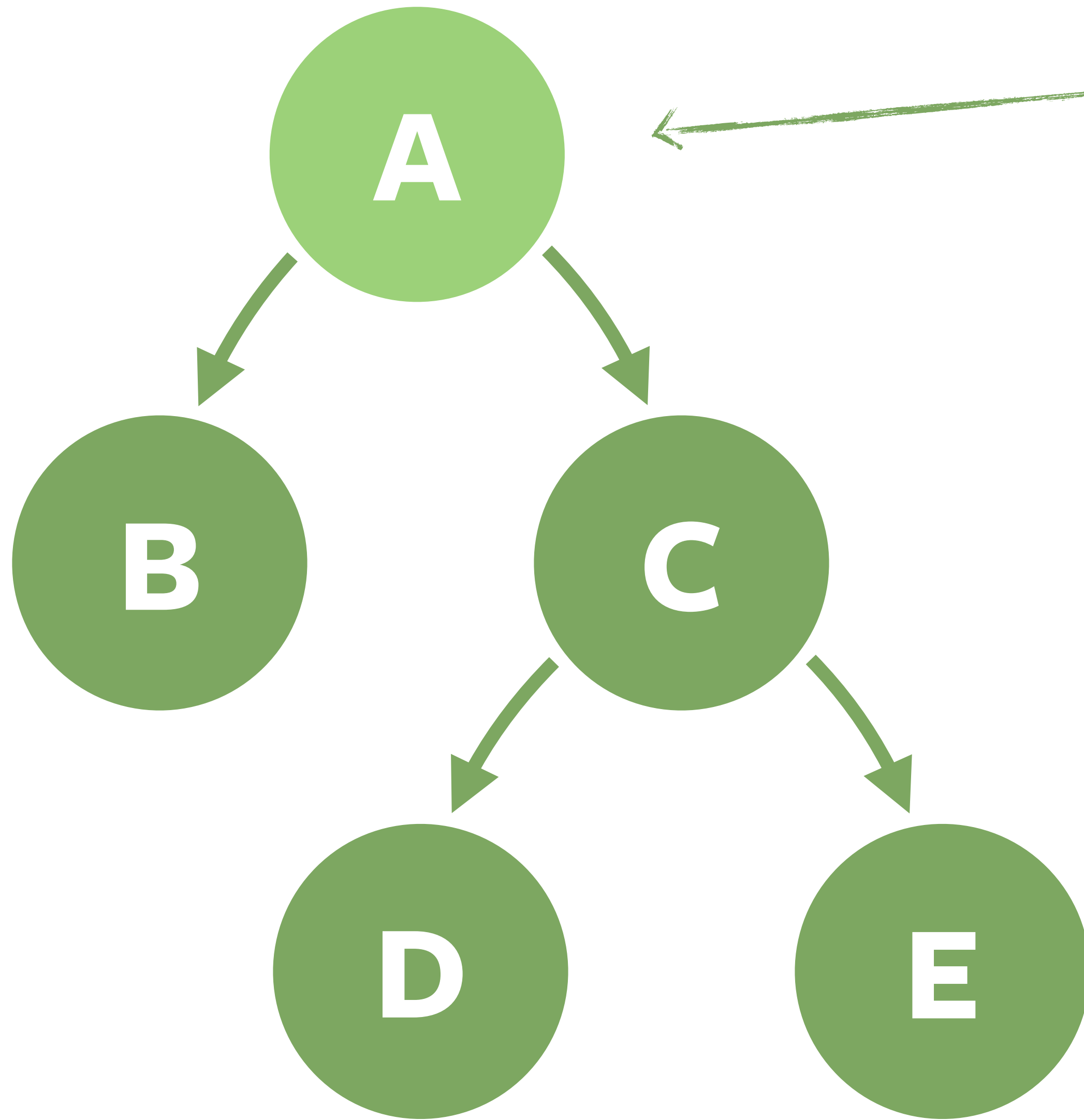
- Understanding
- Consistency
- Testing
- Documentation
- Observation



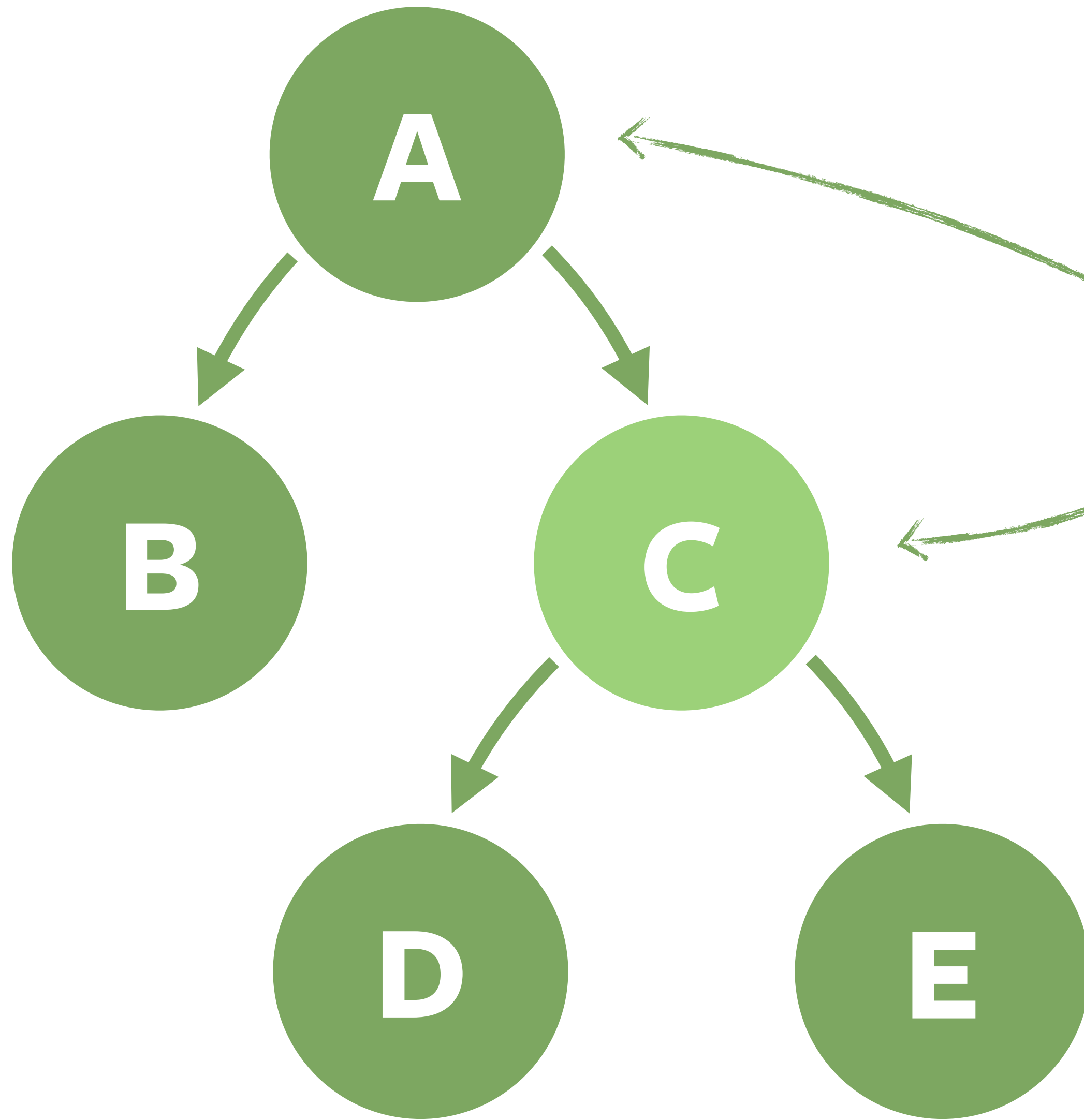
Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation

Scope of Test Execution



Change detected here.
We only need to test A!

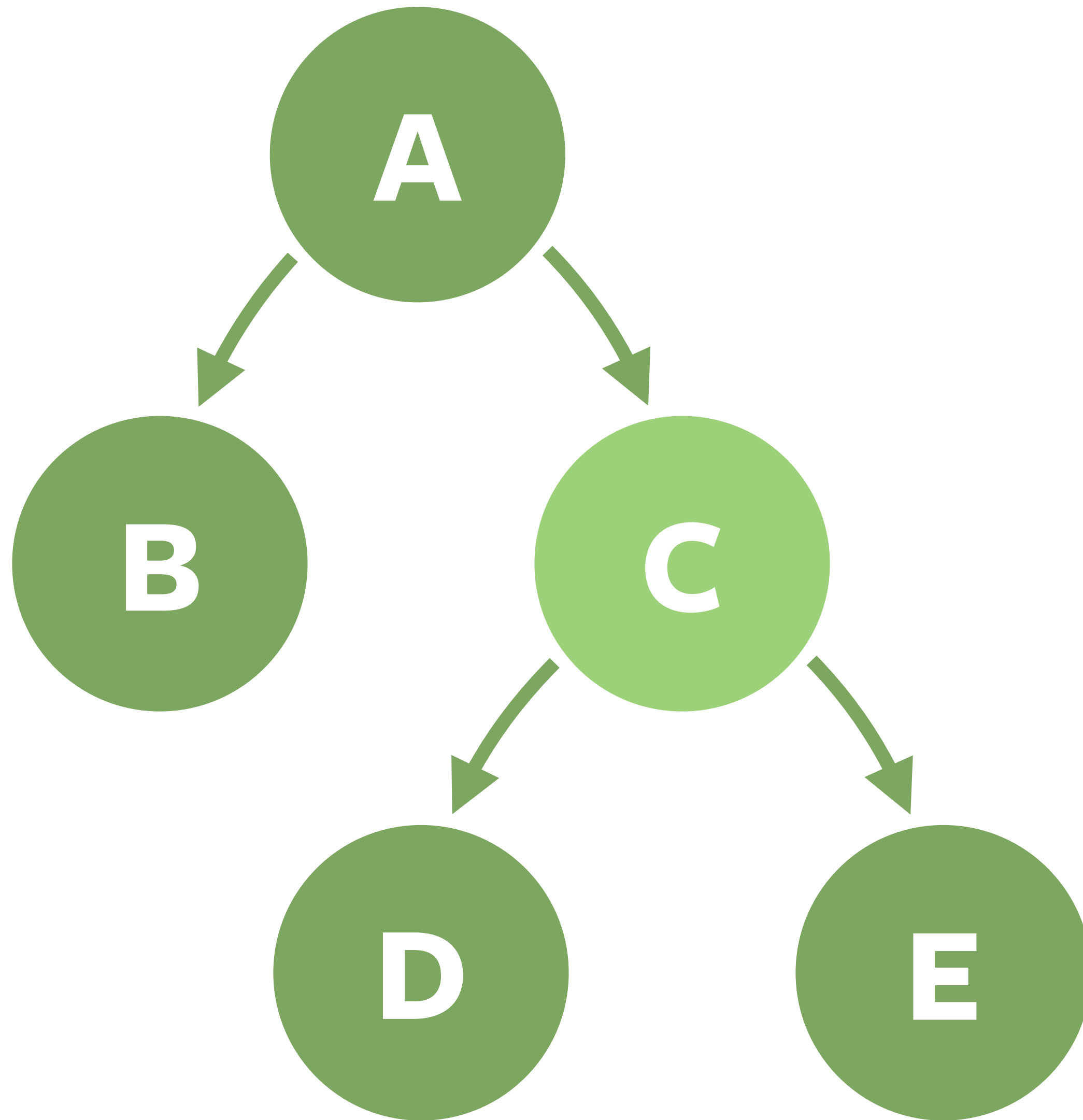


Change detected in C.
We need to test C and A!

```
[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] ·-----
```



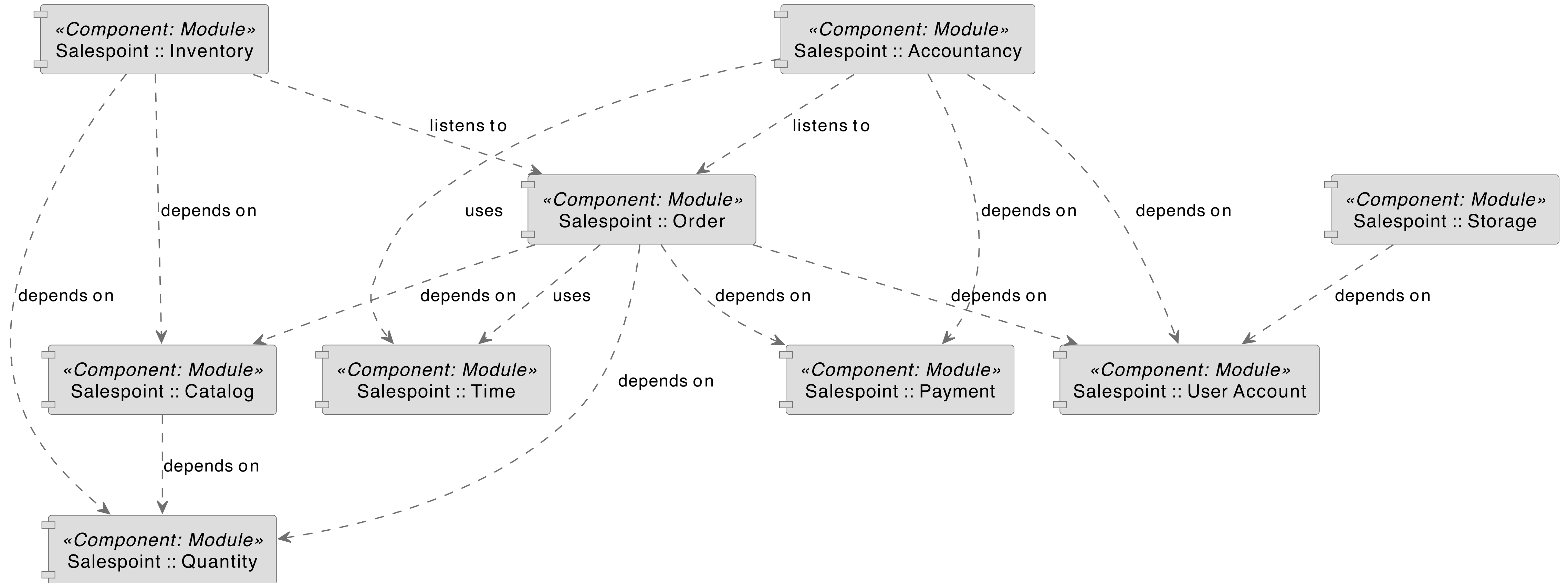
Documentation



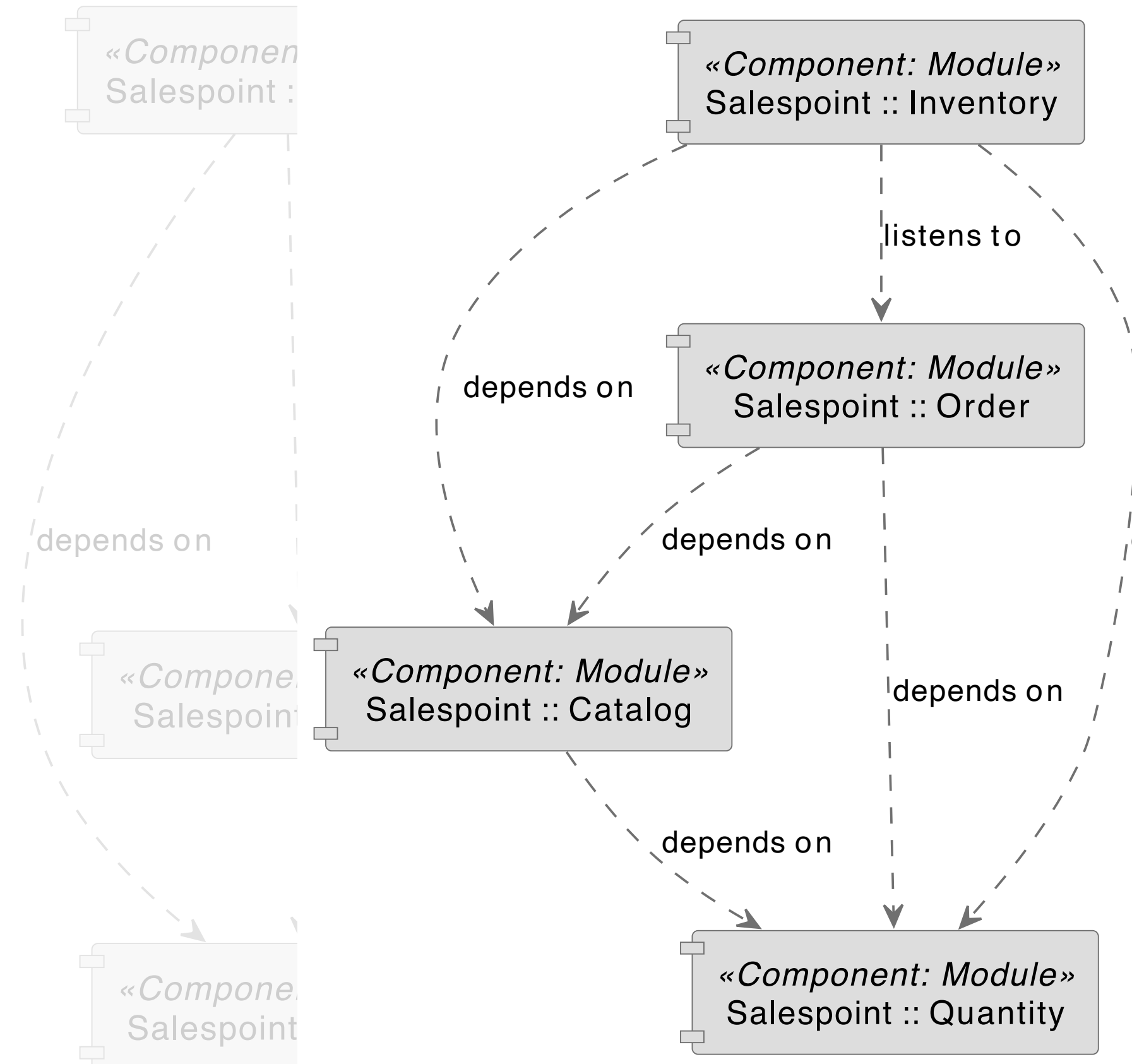
Unit of...

- Understanding
- Consistency
- Testing
- Documentation
- Observation

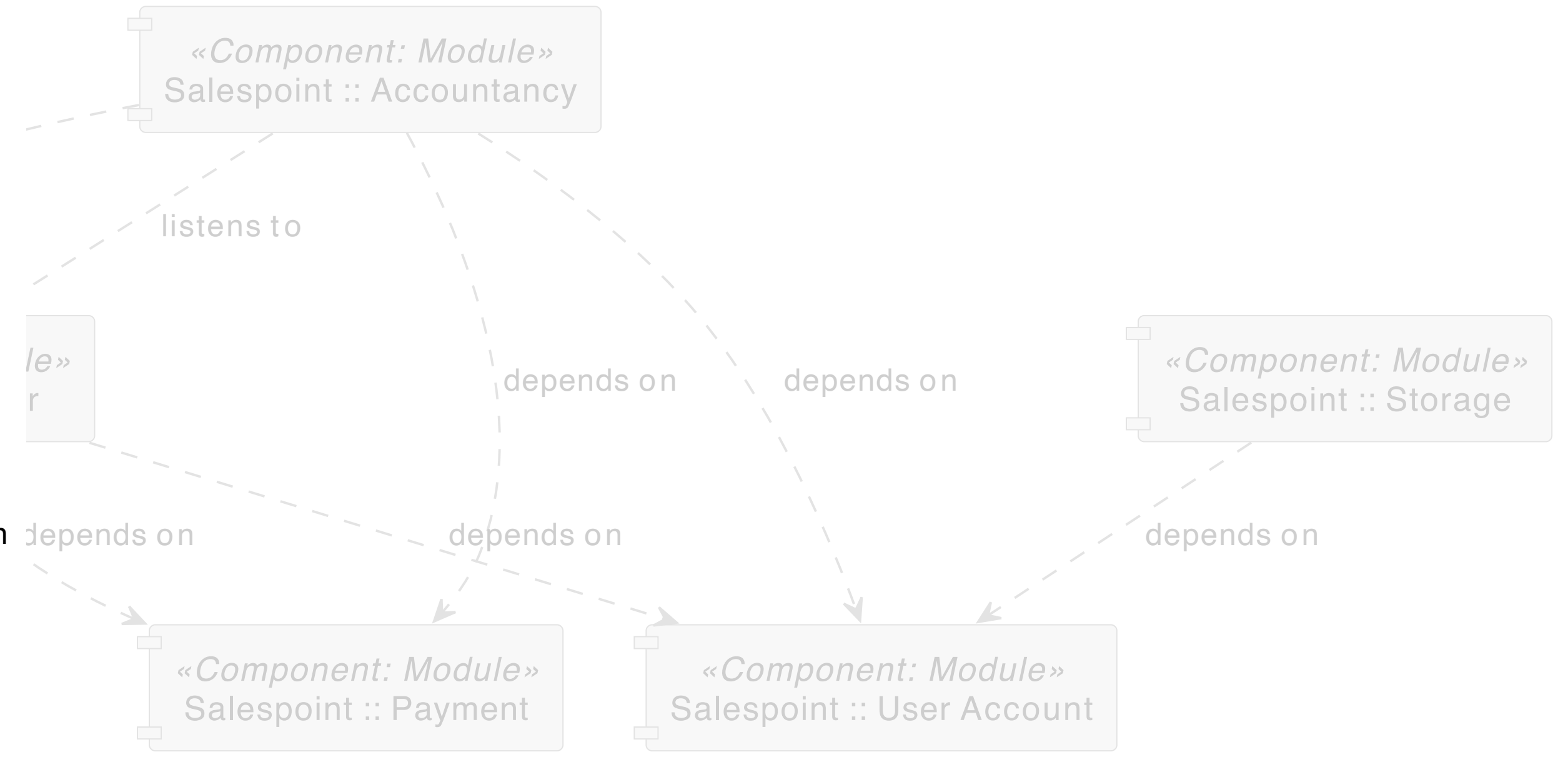
Salespoint

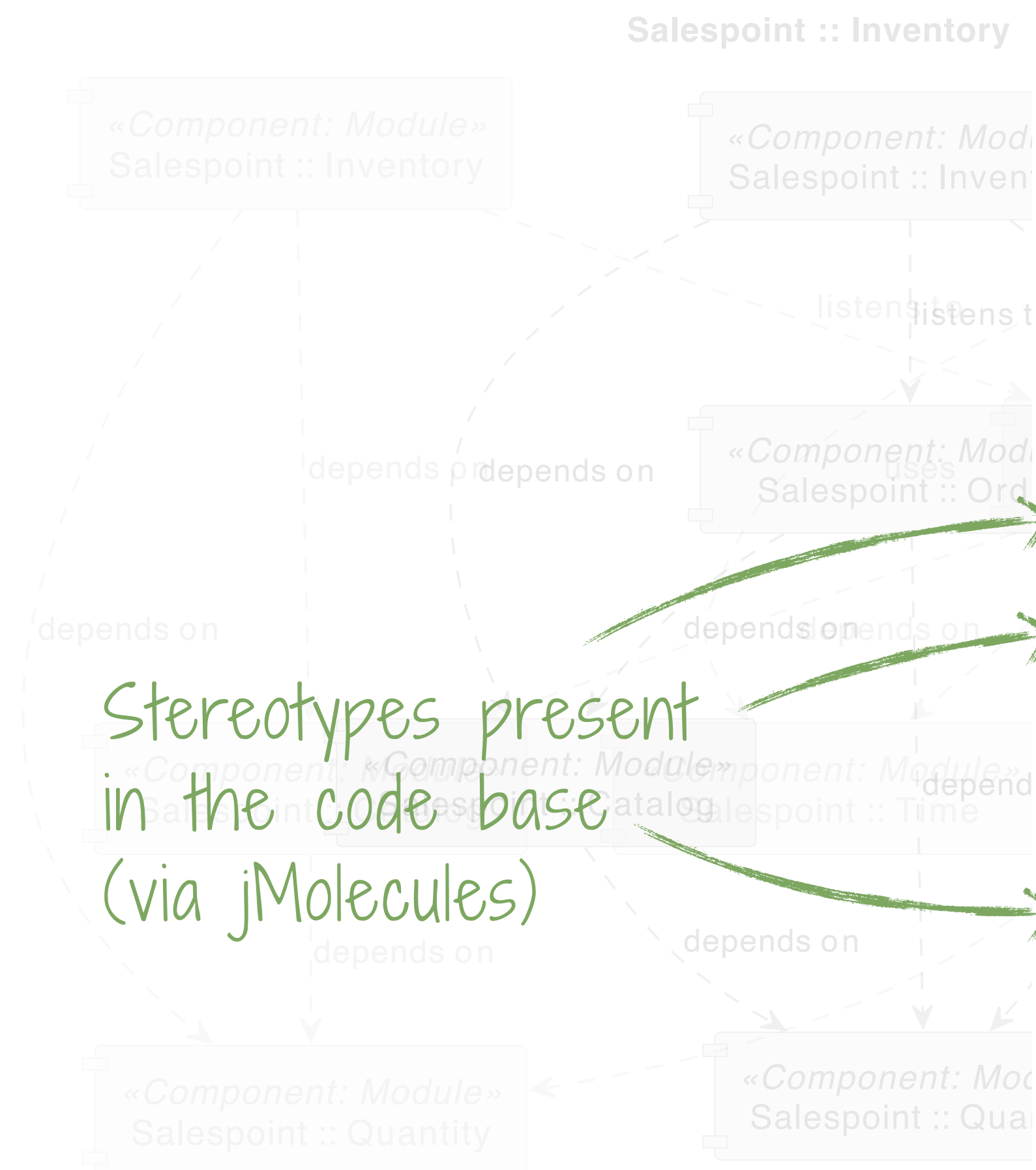


Salespoint :: Inventory



Salespoint





| | |
|--------------------------|---|
| Base package | <code>org.salespointframework.useraccount</code> |
| Spring components | <p><i>Services</i></p> <ul style="list-style-type: none"> <code>o.s.u.UserAccountManagement</code> (via <code>o.s.u.PersistentUserAccountManagement</code>) <p><i>Others</i></p> <ul style="list-style-type: none"> <code>o.s.u.AuthenticationManagement</code> (via <code>o.s.u.SpringSecurityAuthenticationManagement</code>) |
| Aggregate roots | <ul style="list-style-type: none"> <code>o.s.u.UserAccount</code> |
| Value types | <ul style="list-style-type: none"> <code>o.s.u.EncryptedPassword</code> <code>o.s.u.UnencryptedPassword</code> <code>o.s.u.Role</code> |
| Published events | <ul style="list-style-type: none"> <code>o.s.u.UserAccountCreated</code> created by: <ul style="list-style-type: none"> <code>o.s.u.UserAccount.onCreate()</code> |
| Properties | <ul style="list-style-type: none"> <code>salespoint.authentication.login-via-email</code> — <code>java.lang.Boolean</code>, default <code>false</code>. Enables the login procedure to use the email address to lookup a user instead of their username. Defaults to <code>false</code>. |



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](#)
- **Architecturally Evident Spring**
[Example project @ GitHub](#)