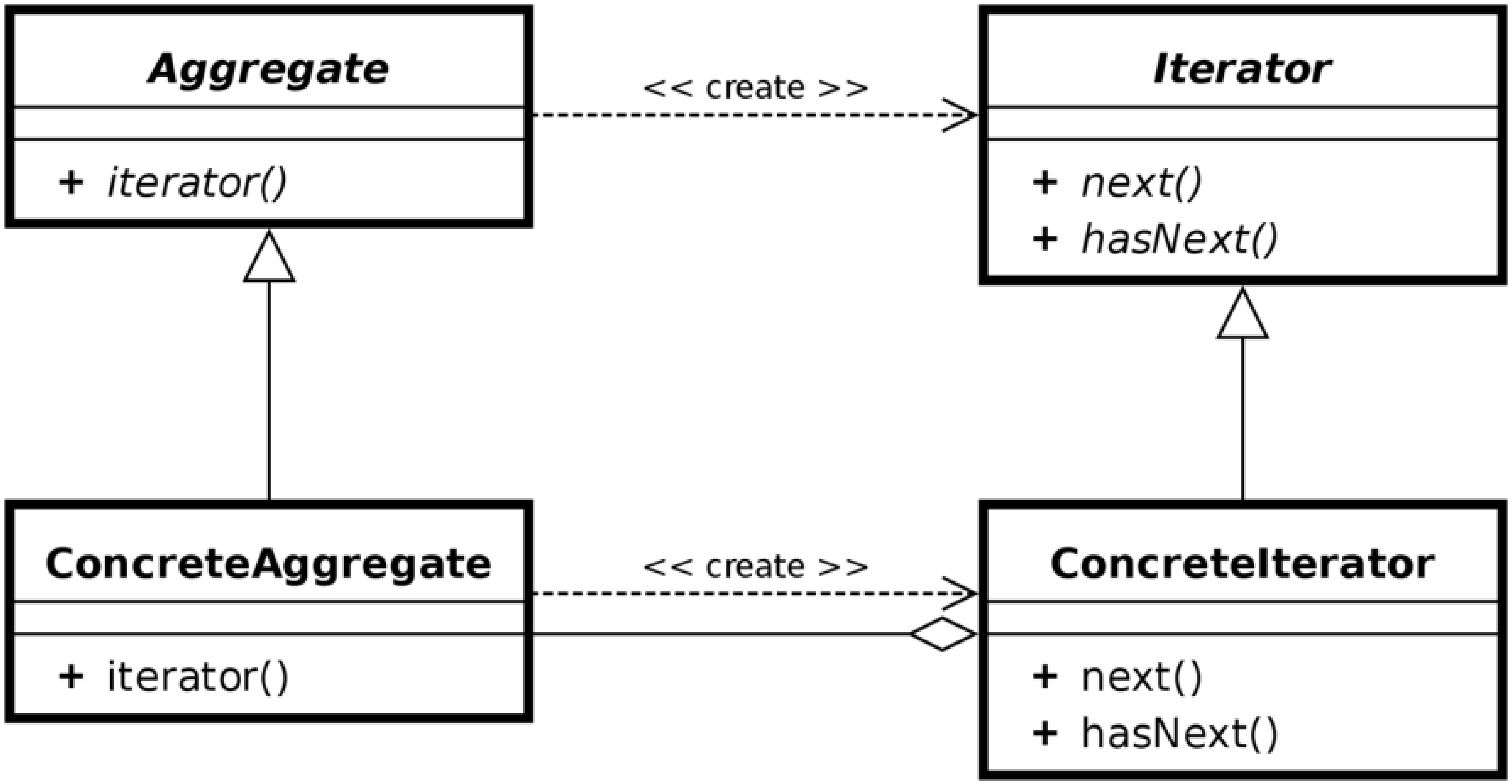
Software System Architectures (NSWI130)
Domain-Driven Architectural Pattern

Martin Nečaský
Faculty of Mathematics and Physics
Charles University in Prague

Design Pattern

Recommended coding practice which is a well-known solution to well-known problems caused by a recurring common situation.

Design Pattern



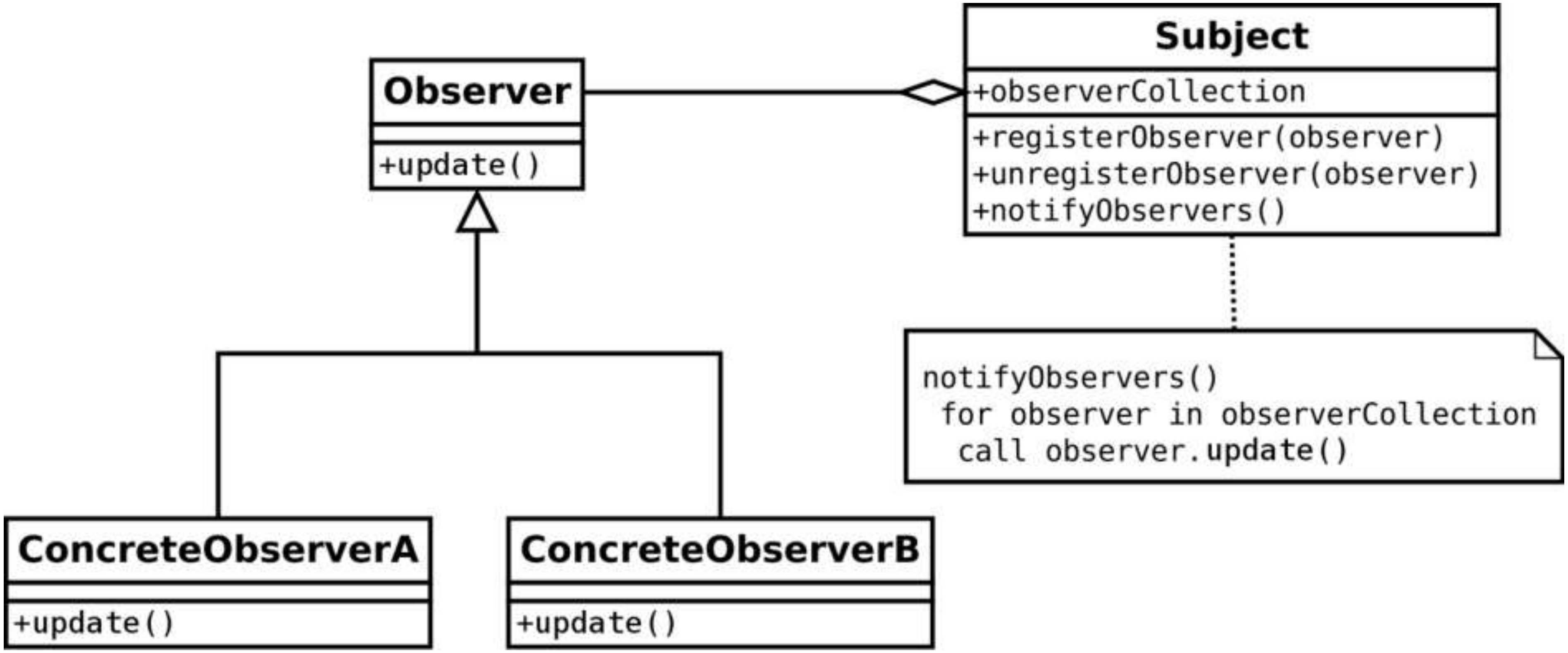
Source:
https://en.wikipedia.org/wiki/Iterator_pattern#/media/File:Iterator_UML_class_diagram.svg

Architectural Pattern

Recommended architectural practice which is a well-known solution to well-known architectural problems caused by a recurring common situation.

Design pattern	Architectural pattern
Fine grained	Coarse grained
Focused on programming problem	Focused on quality attributes
Introduces particular source code structure inside one or more architectural modules	Introduces modules or components/connectors to architectural design

Design vs Architectural Patterns



Source: https://en.wikipedia.org/wiki/Observer_pattern#/media/File:Observer_w_update.svg

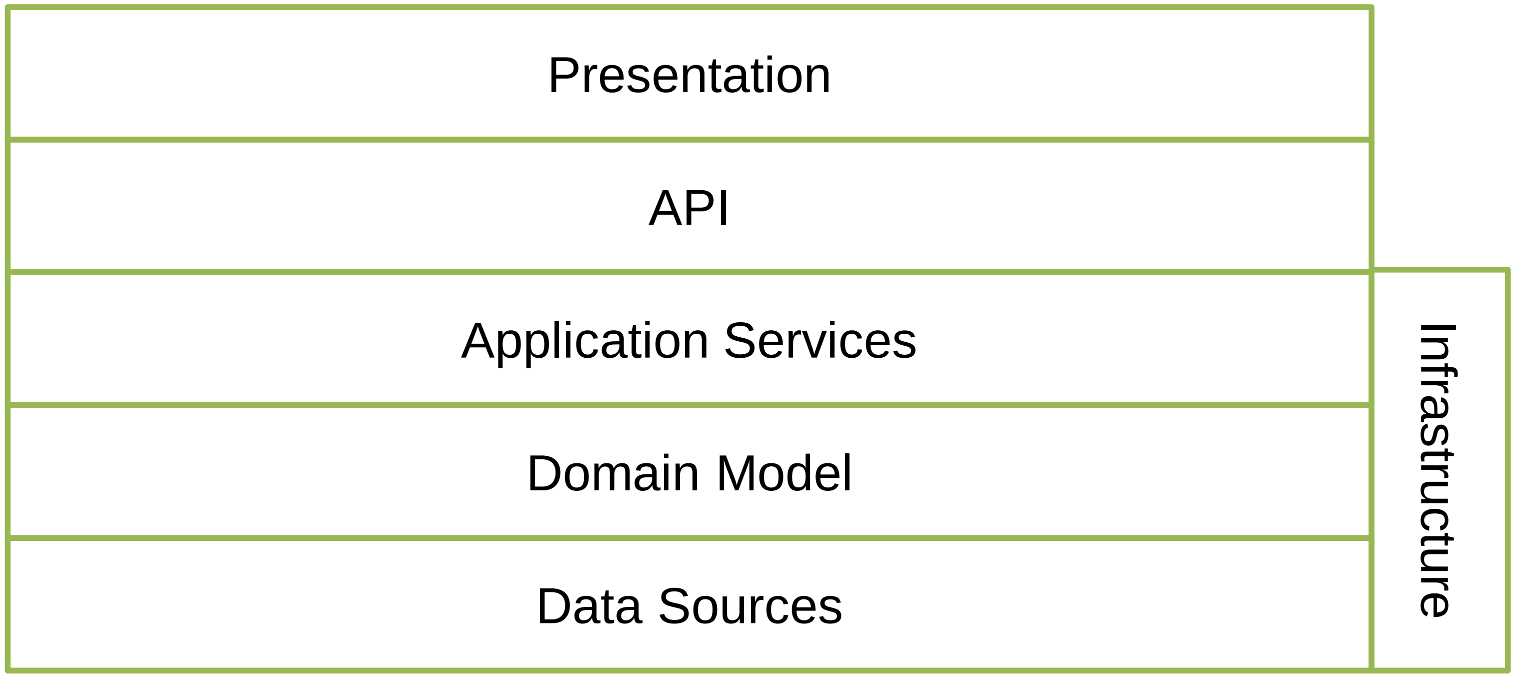
Examples of Patterns

- load balancer
- router
- broker
- application programming interface (API)

Layer Pattern

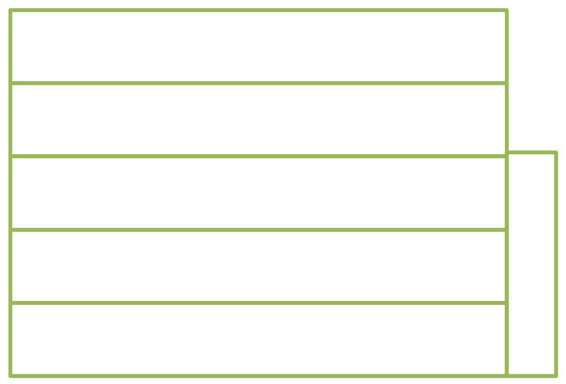
- situation - complex system where we need to develop and evolve its portions independently
- problem - dependencies between parts
- solution - layers

Layer Pattern



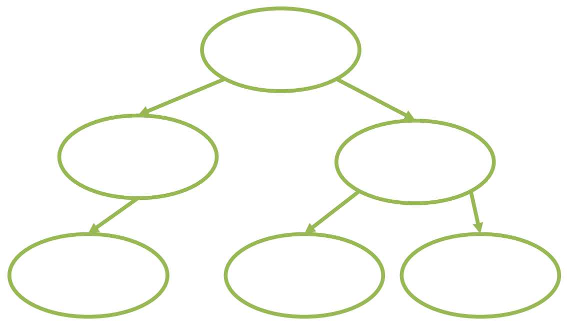
Layer Pattern

Layered Modules



Monolithic Component

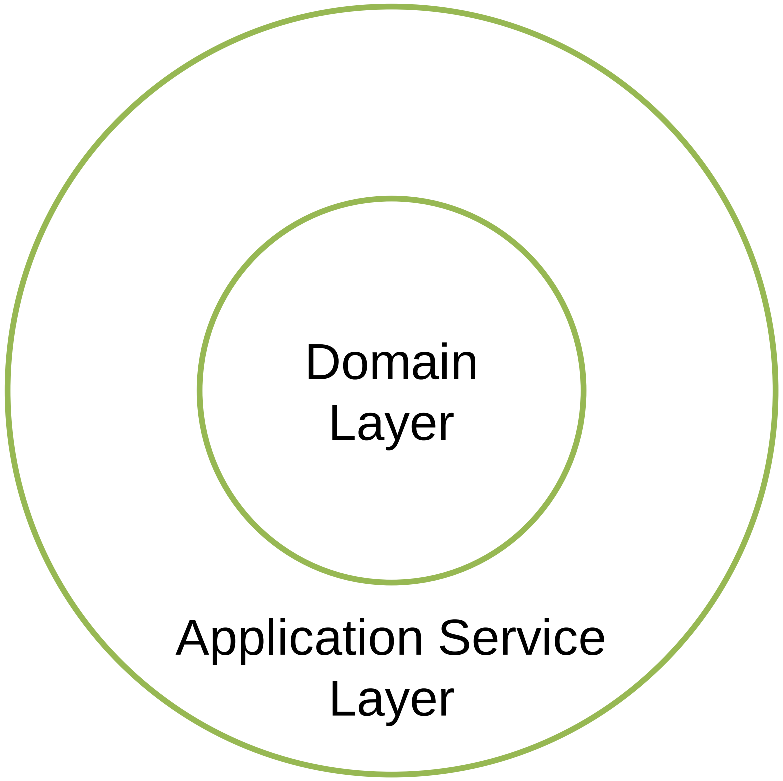
Distributed Network Components

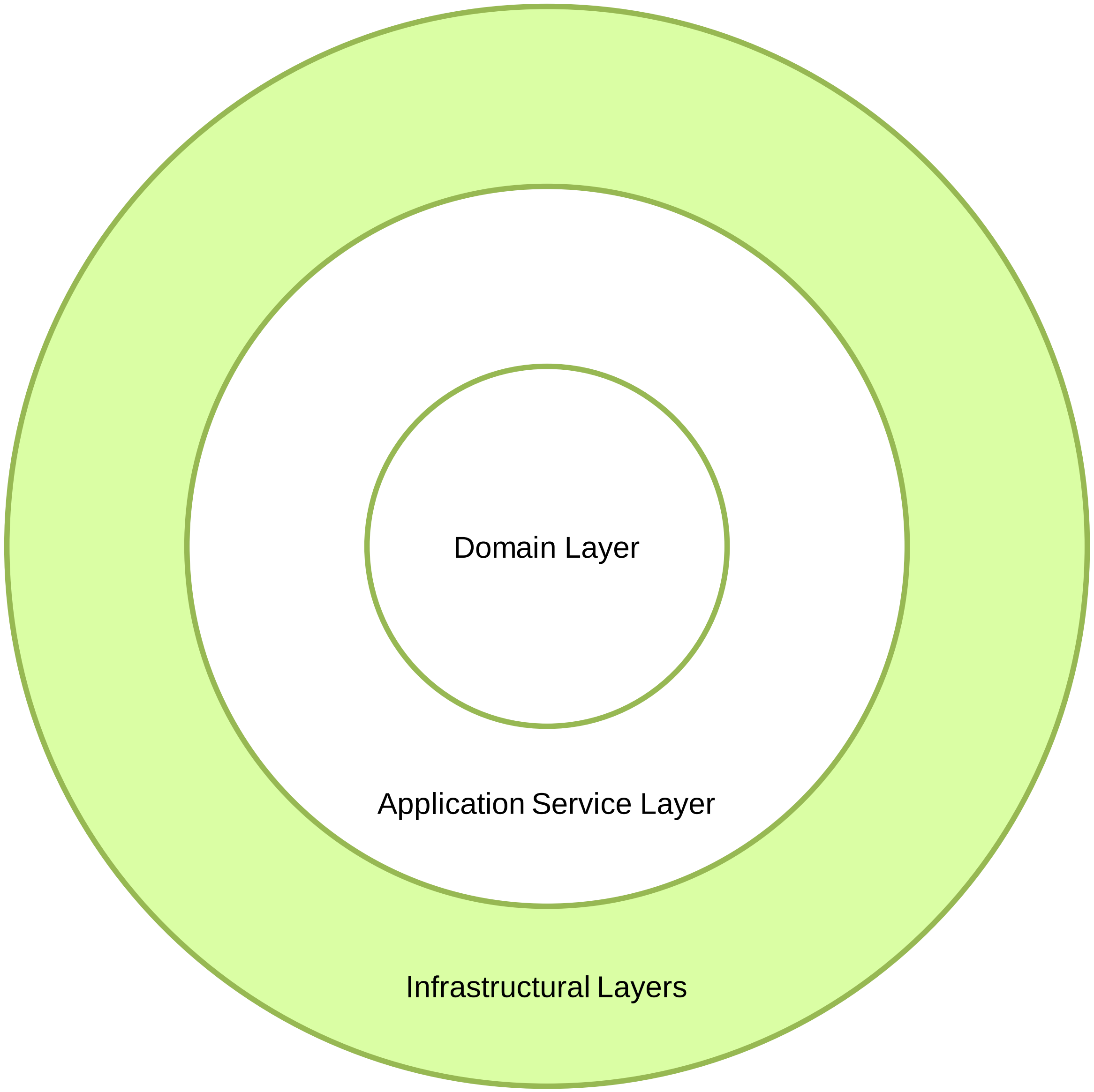


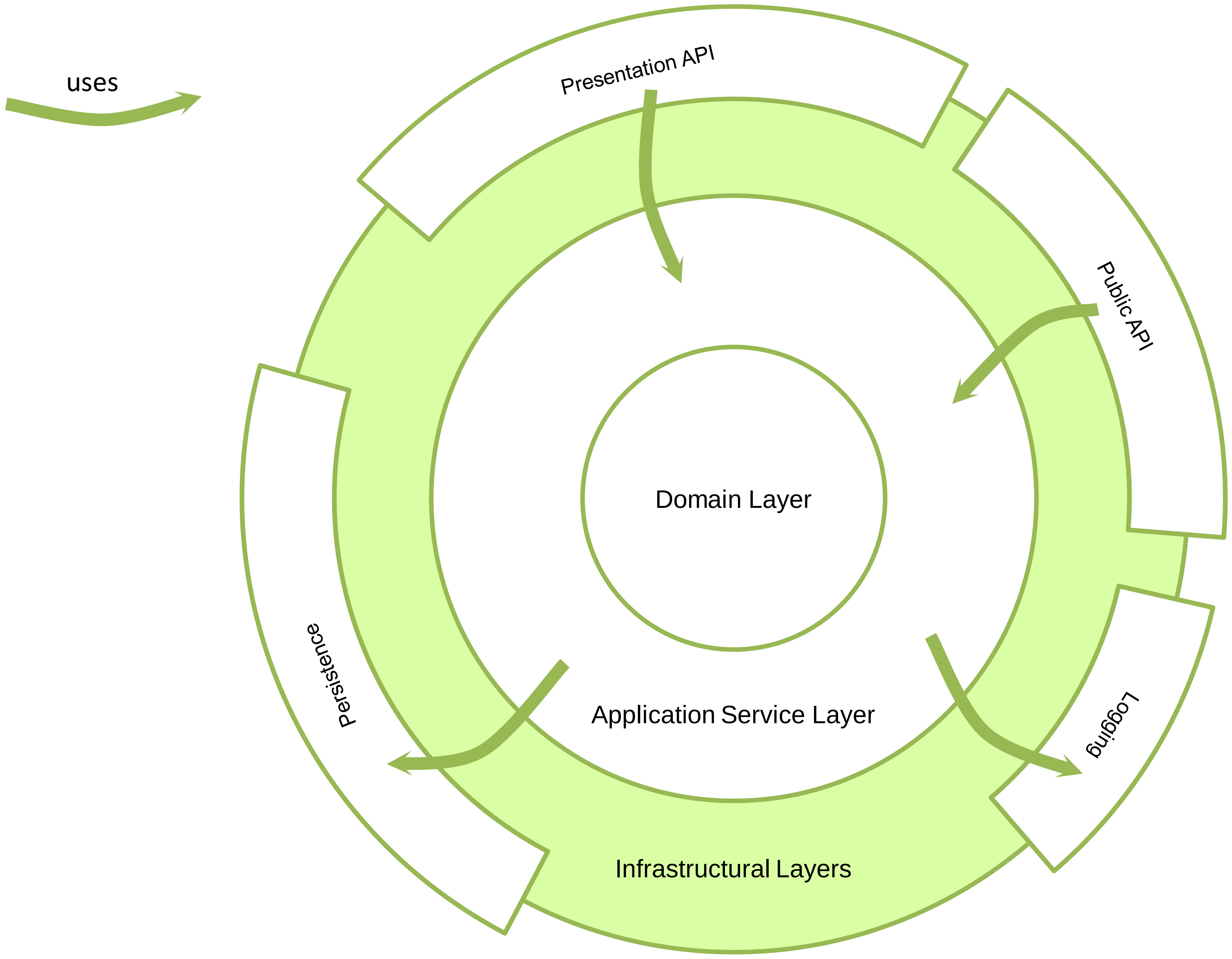
Domain-driven architecture

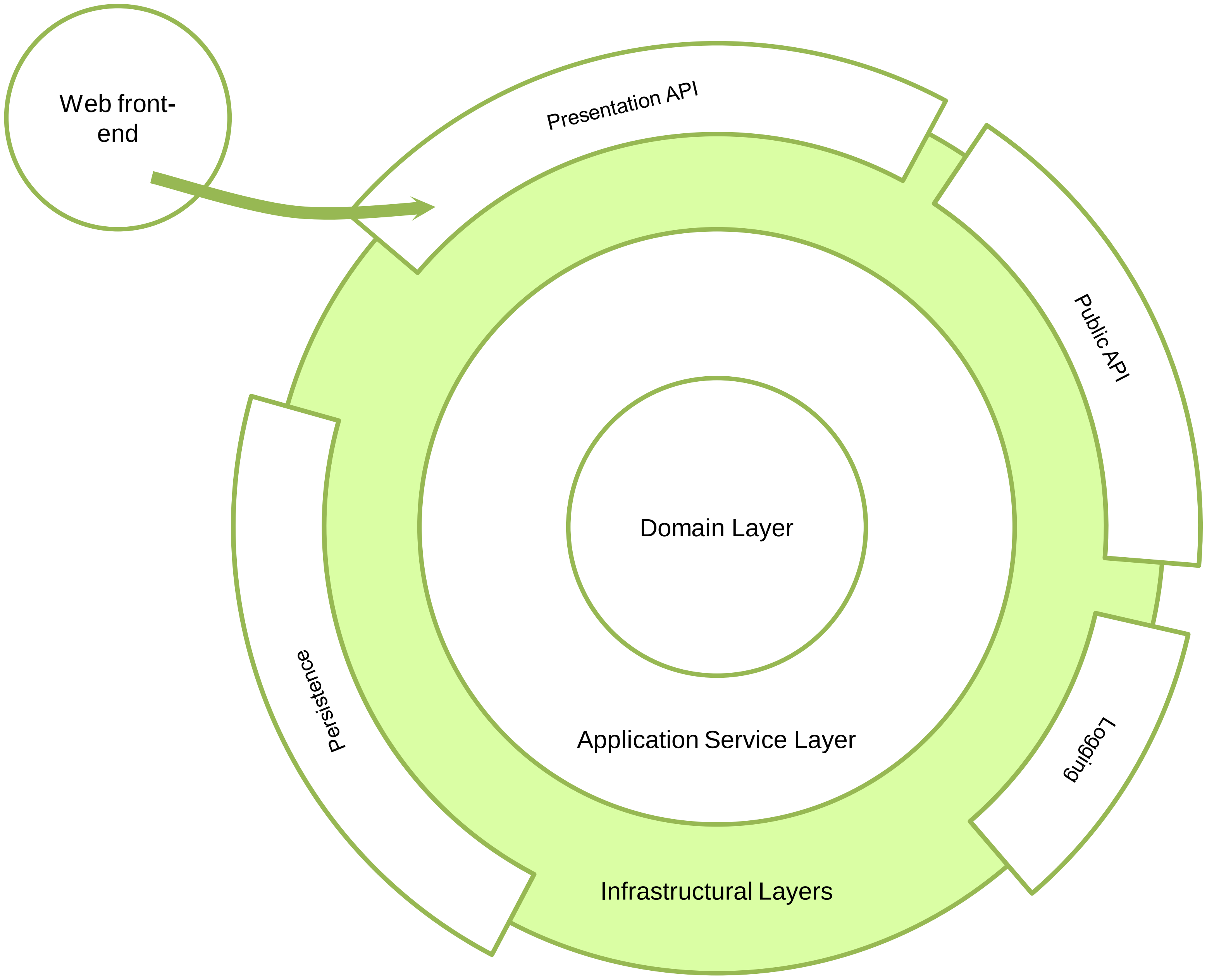
- based on separation of technical complexities from the complexities of the problem domain
- helps to change portion of the system without undesired effects
- assumption – presentation, persistence and domain logic usually change independently

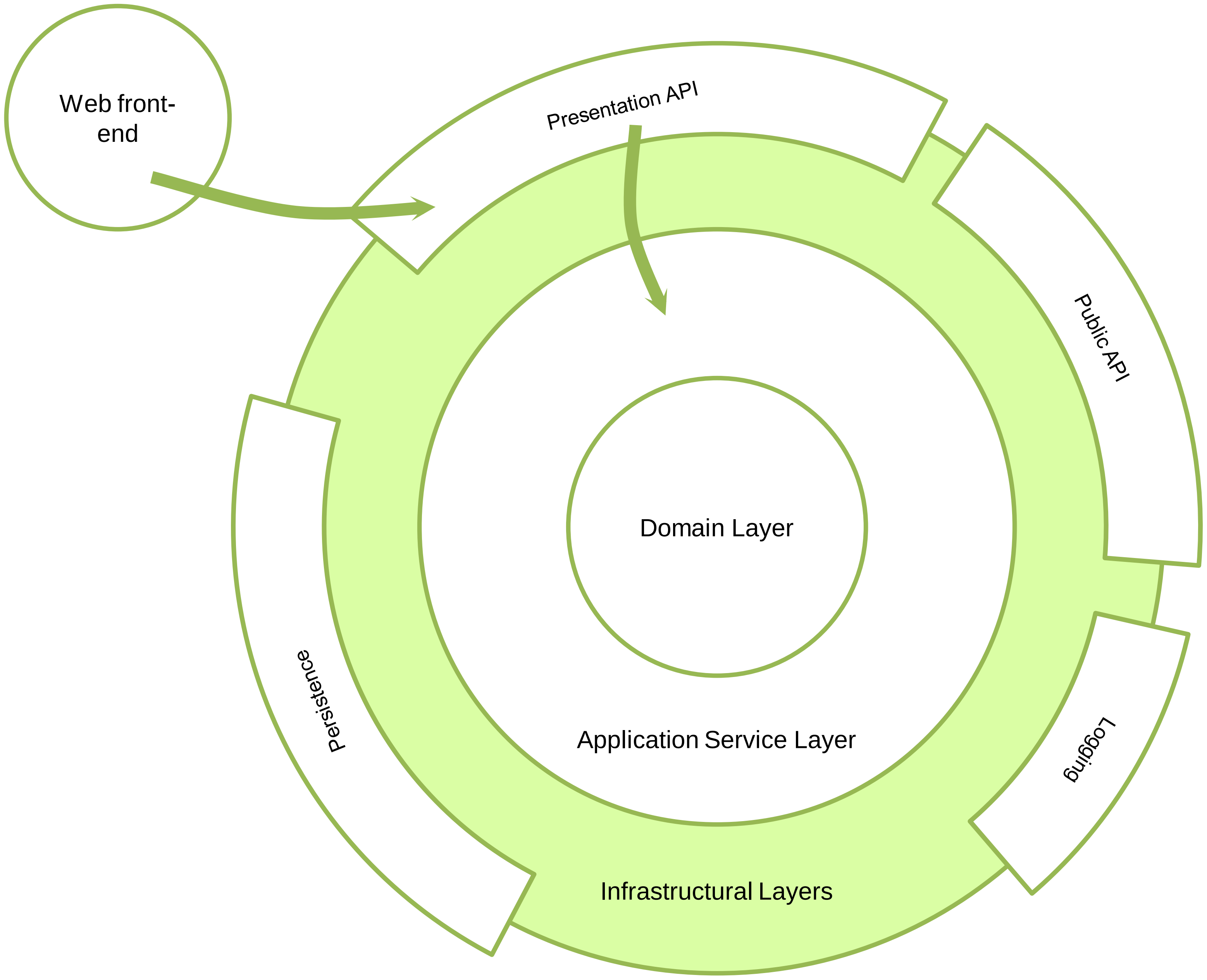
Domain Layer

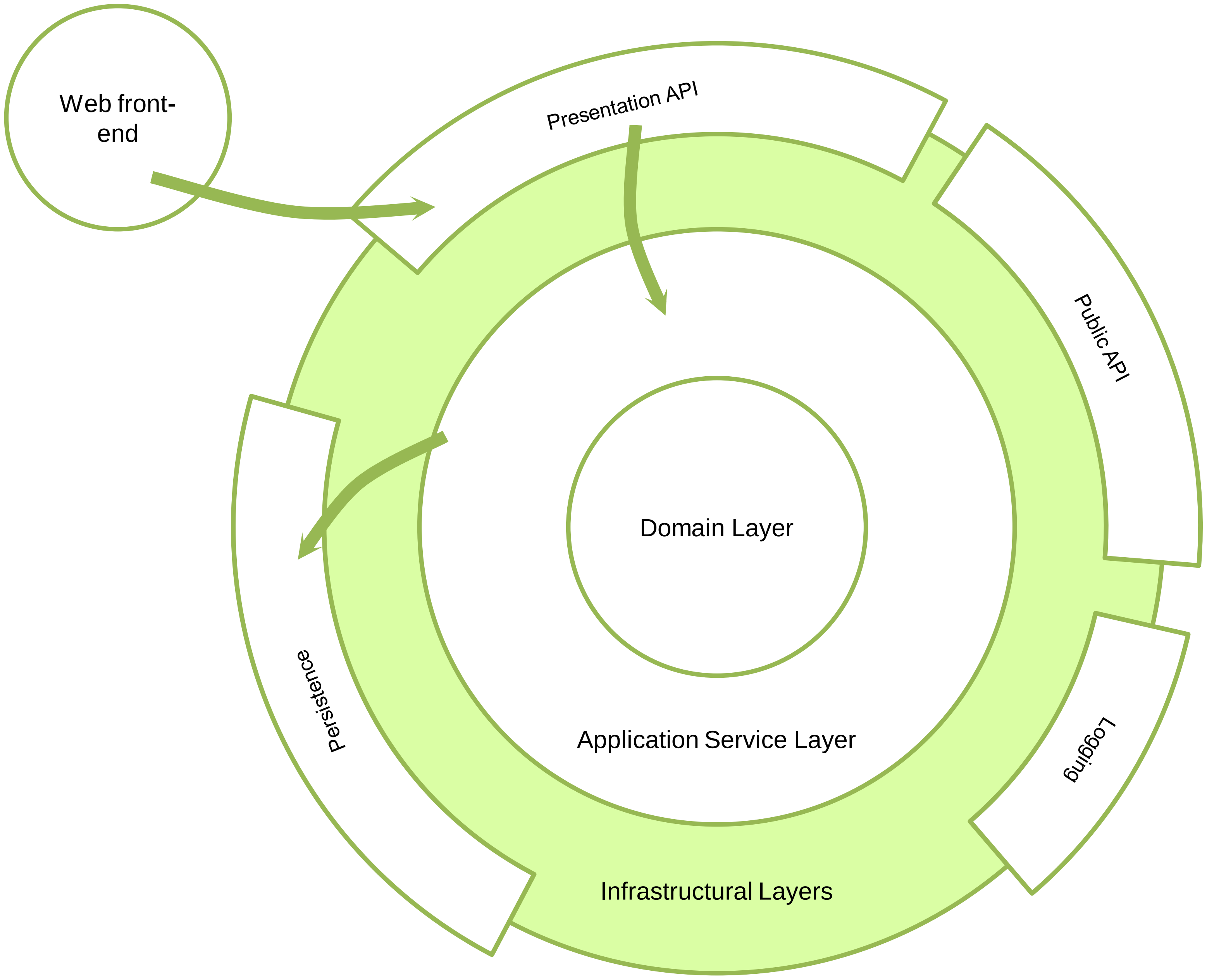

Domain Layer
Application Service Layer

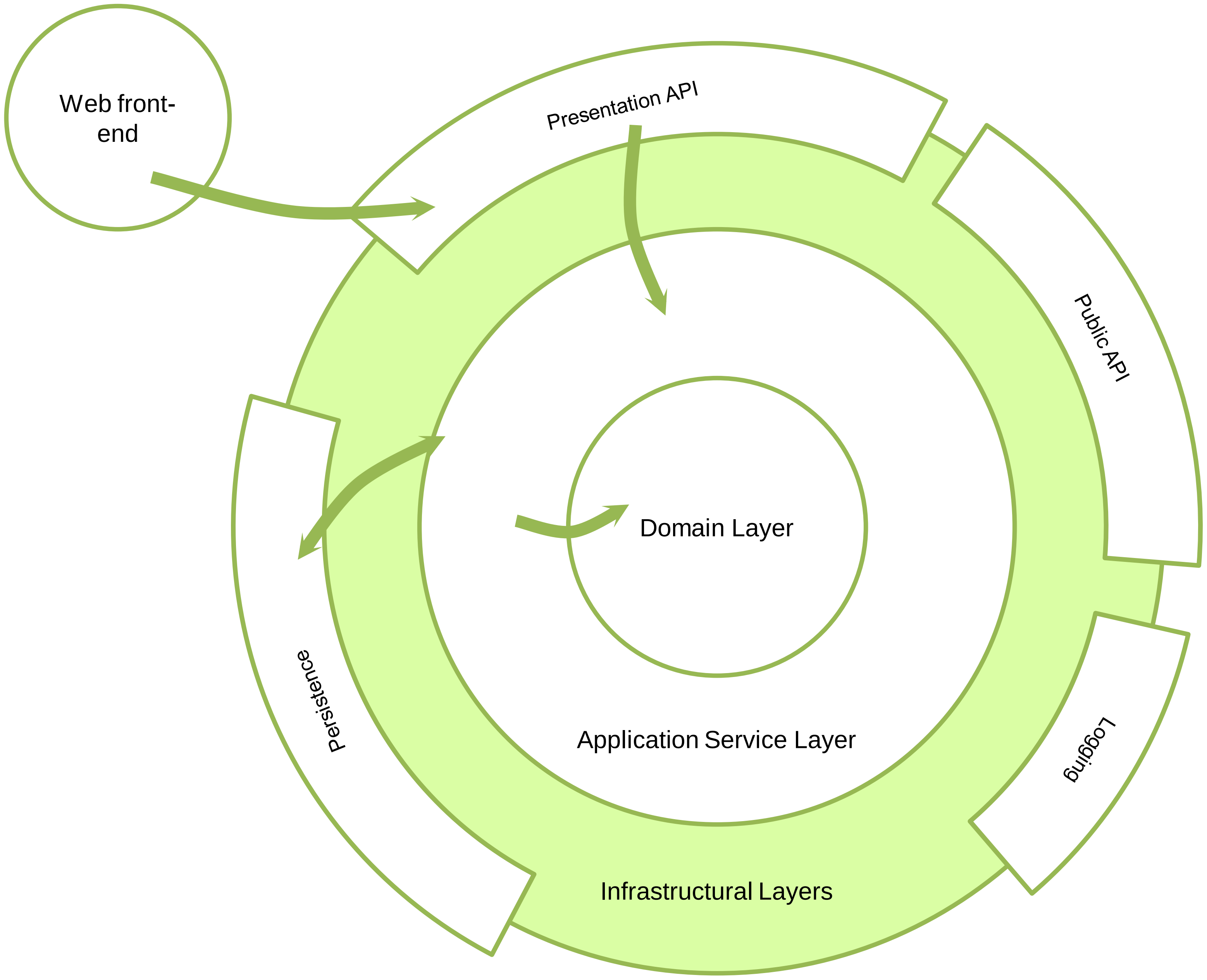

Domain Layer
Application Service Layer
Infrastructural Layers

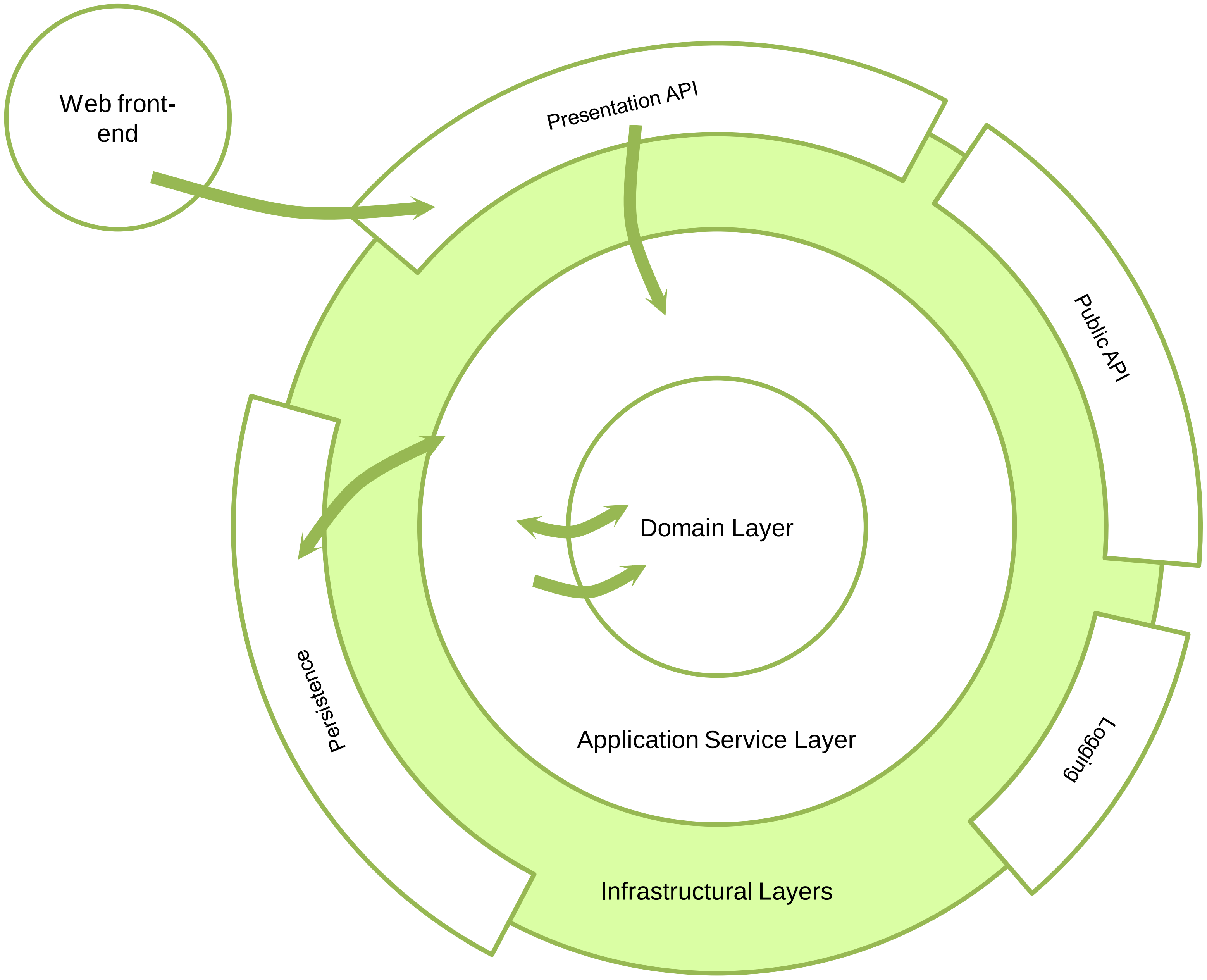

uses
Presentation API
Public API
Domain Layer
Persistence
Application Service Layer
Logging
Infrastructural Layers

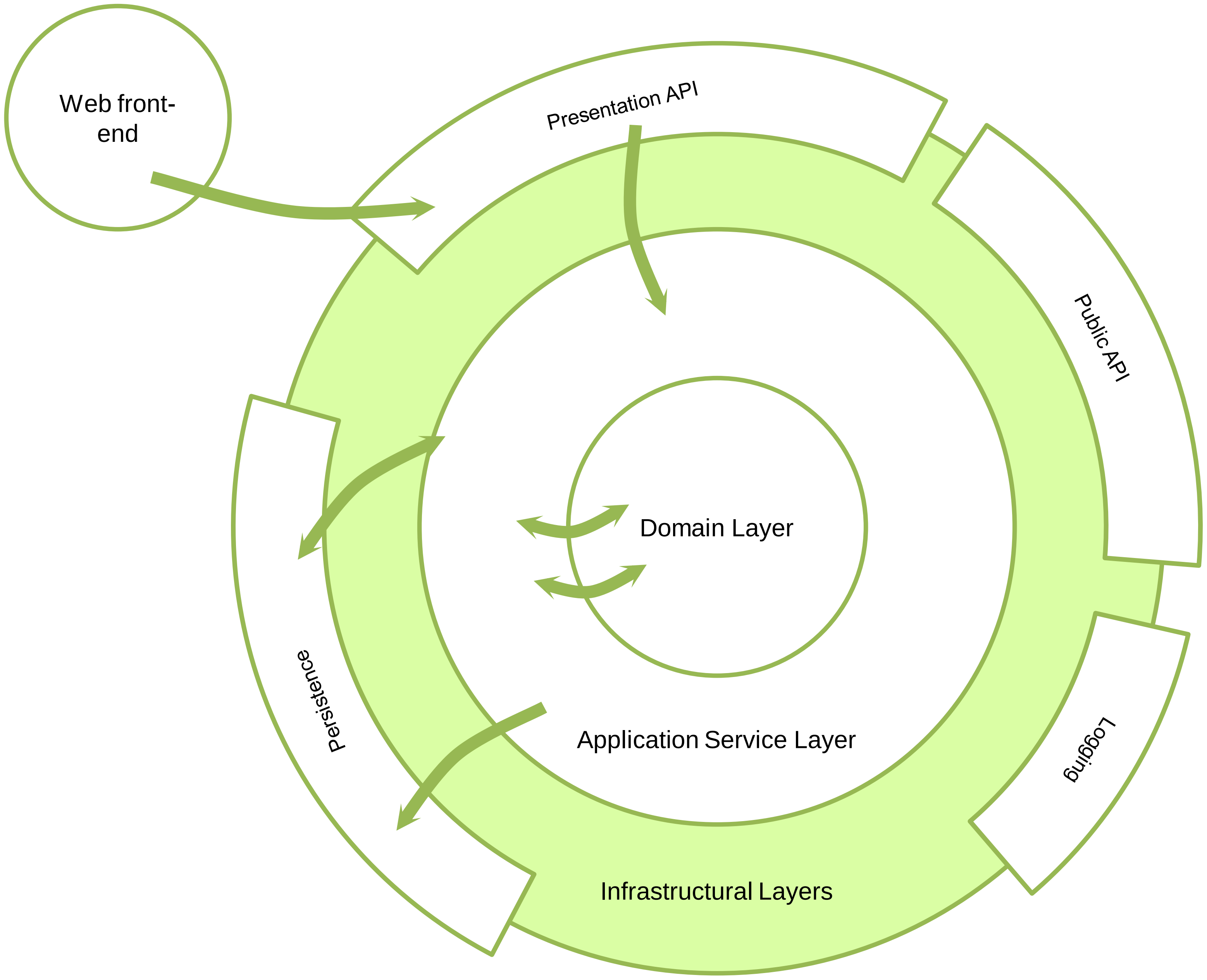

Web front-end
Presentation API
Public API
Domain Layer
Application Service Layer
Persistence
Logging
Infrastructural Layers

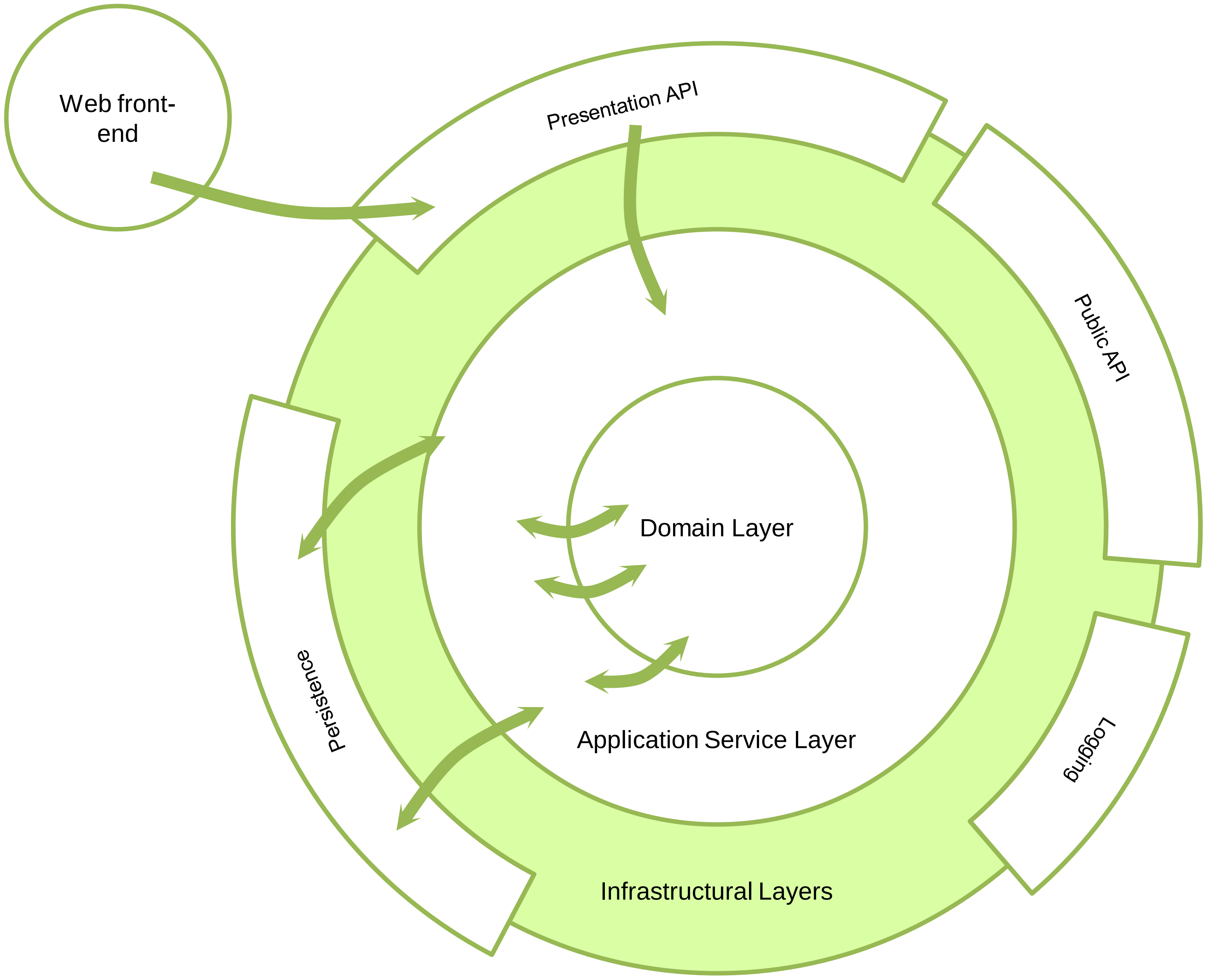

Web front-end
Presentation API
Public API
Domain Layer
Application Service Layer
Persistence
Logging
Infrastructural Layers

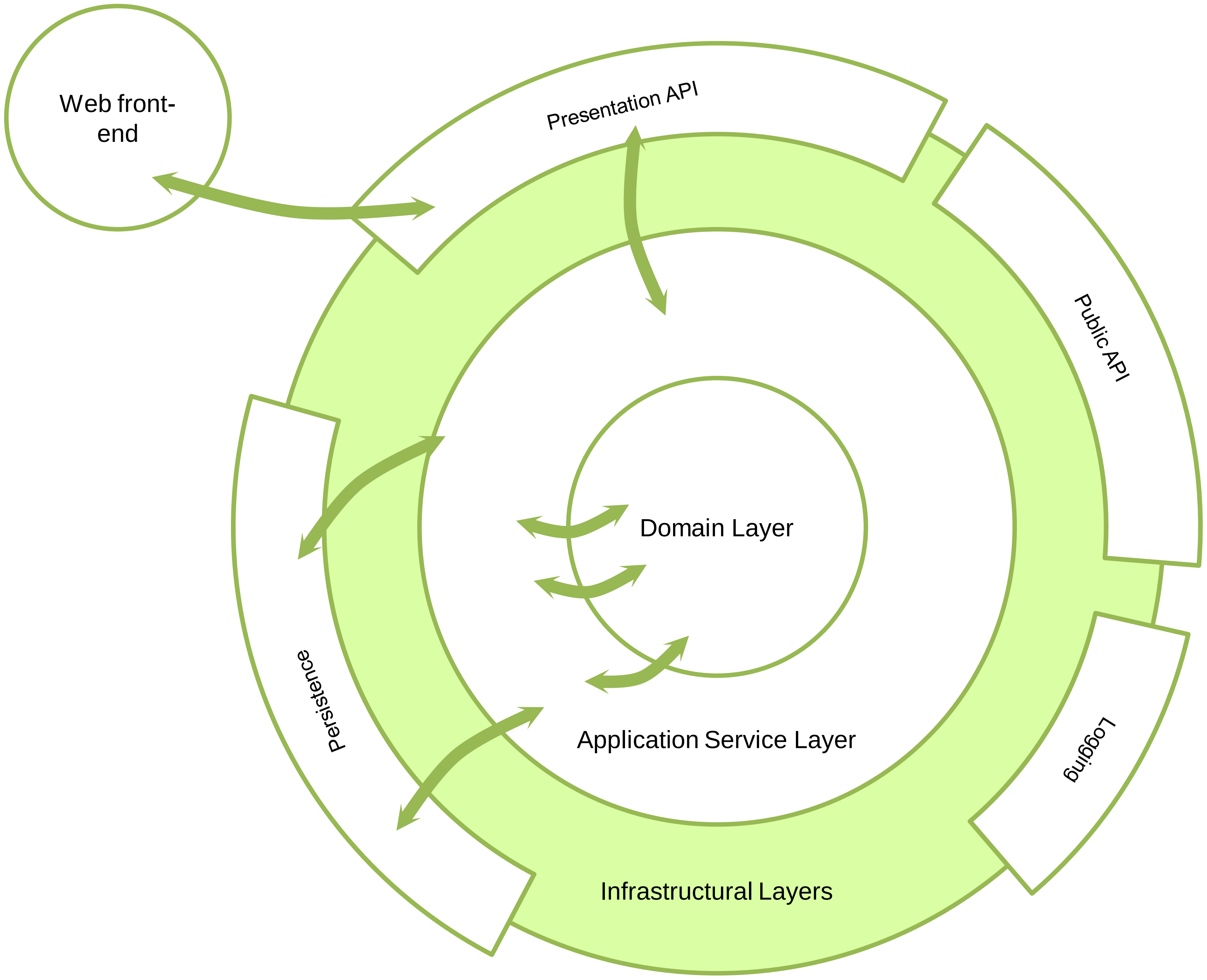

Web front-end
Presentation API
Public API
Persistence
Domain Layer
Application Service Layer
Logging
Infrastructural Layers

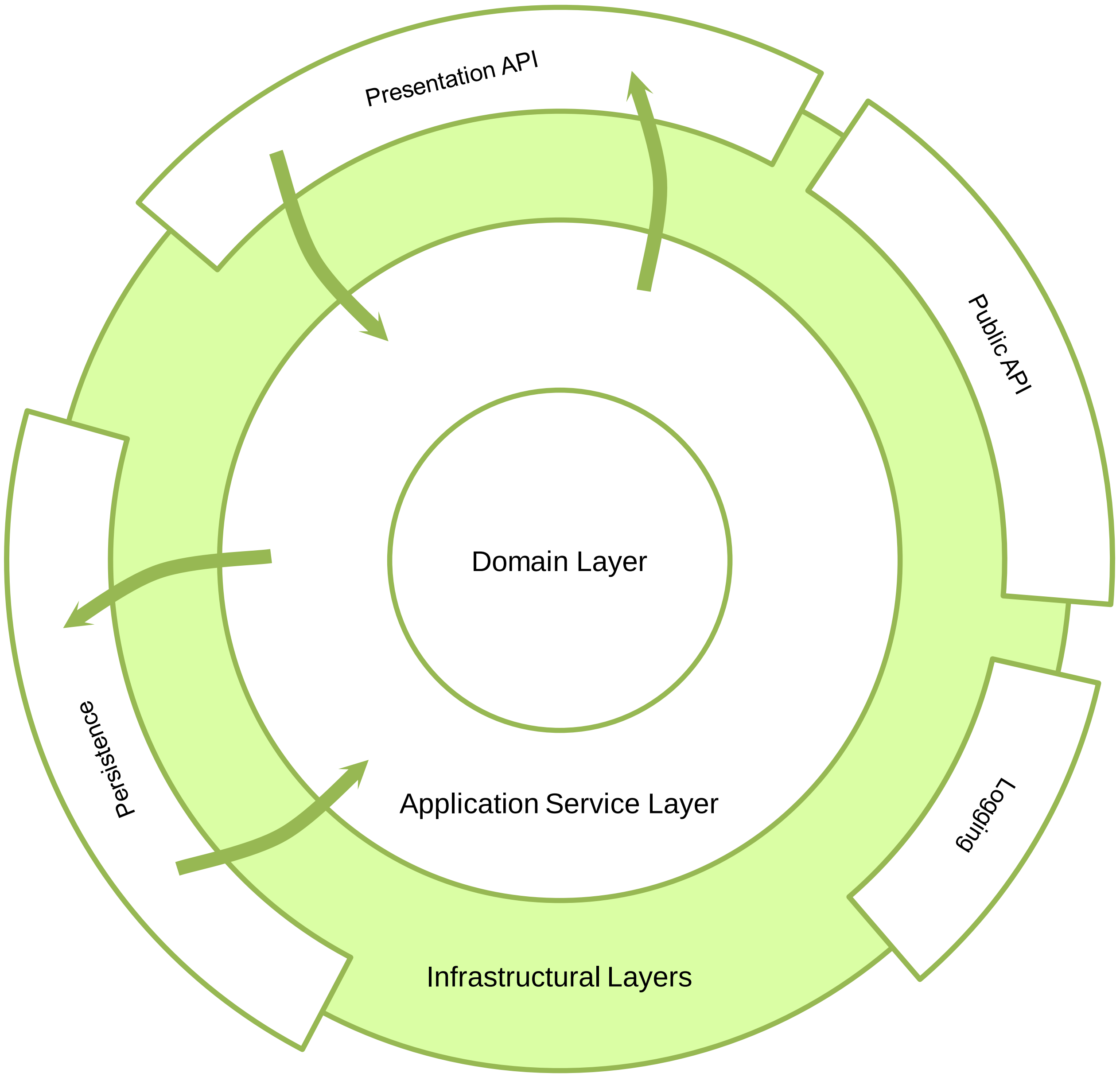

Web front-end
Presentation API
Public API
Domain Layer
Application Service Layer
Persistence
Logging
Infrastructural Layers

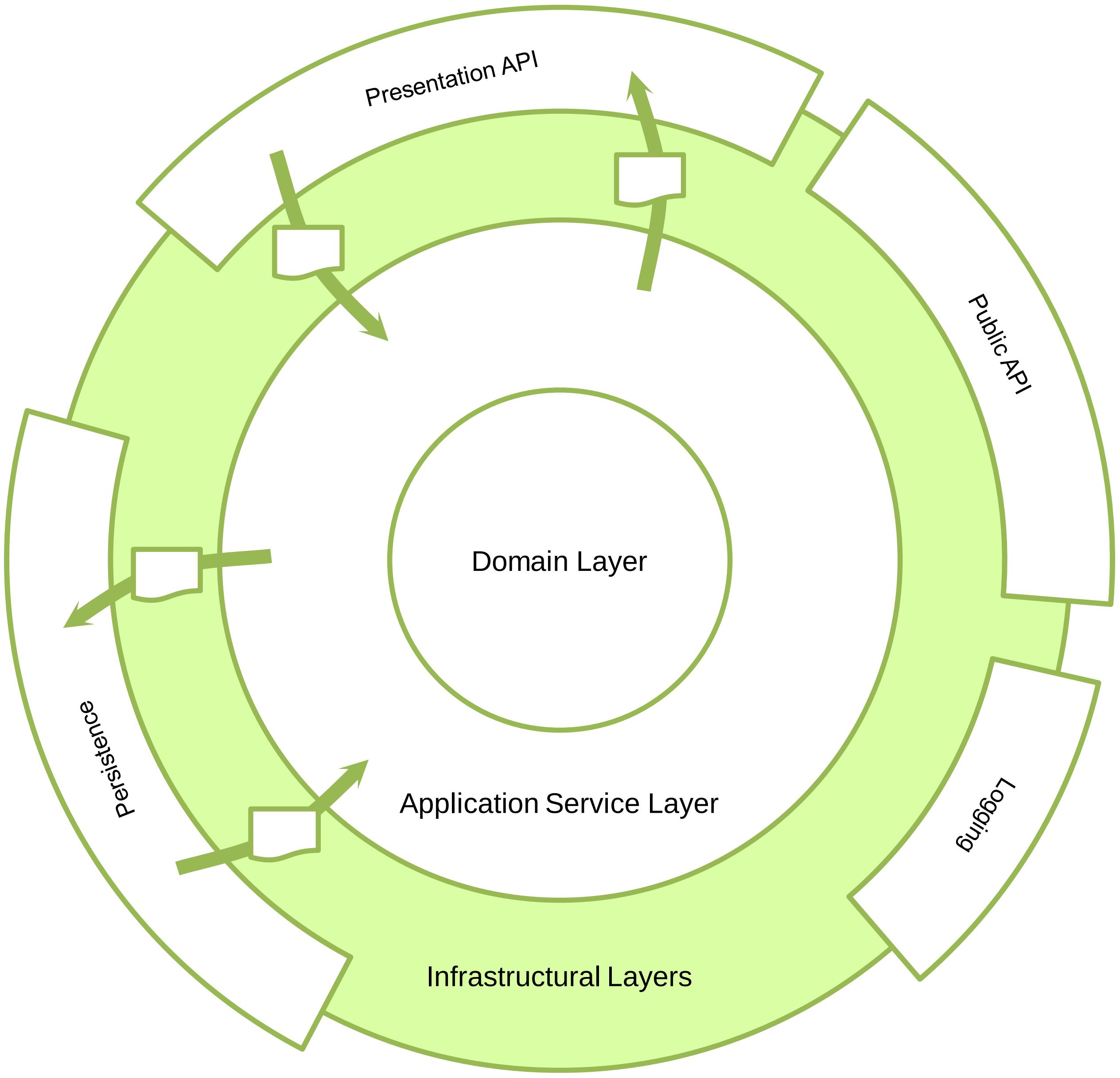

Web front-end
Presentation API
Public API
Domain Layer
Persistence
Application Service Layer
Logging
Infrastructural Layers


Web front-end
Presentation API
Public API
Domain Layer
Persistence
Application Service Layer
Logging
Infrastructural Layers


Web front-end
Presentation API
Public API
Domain Layer
Persistence
Application Service Layer
Logging
Infrastructural Layers


Web front-end
Presentation API
Public API
Domain Layer
Application Service Layer
Persistence
Logging
Infrastructural Layers


Presentation API
Public API
Domain Layer
Persistence
Application Service Layer
Logging
Infrastructural Layers


Presentation API
Public API
Domain Layer
Persistence
Application Service Layer
Logging
Infrastructural Layers

Domain-driven pattern can be applied in any software system.

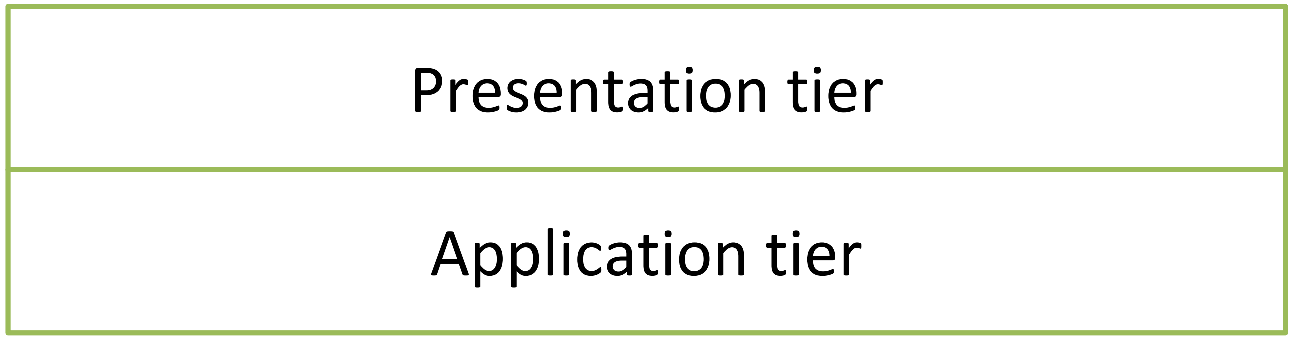
But it cannot be recommended for all cases.

Three-tier Pattern

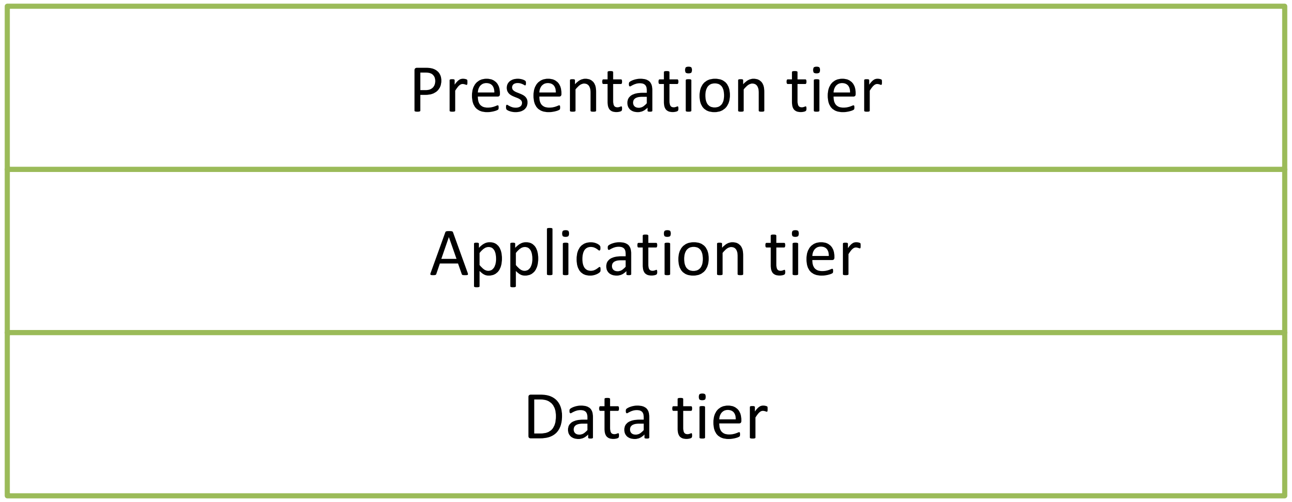
Three-tier Pattern

Presentation tier

Three-tier Pattern



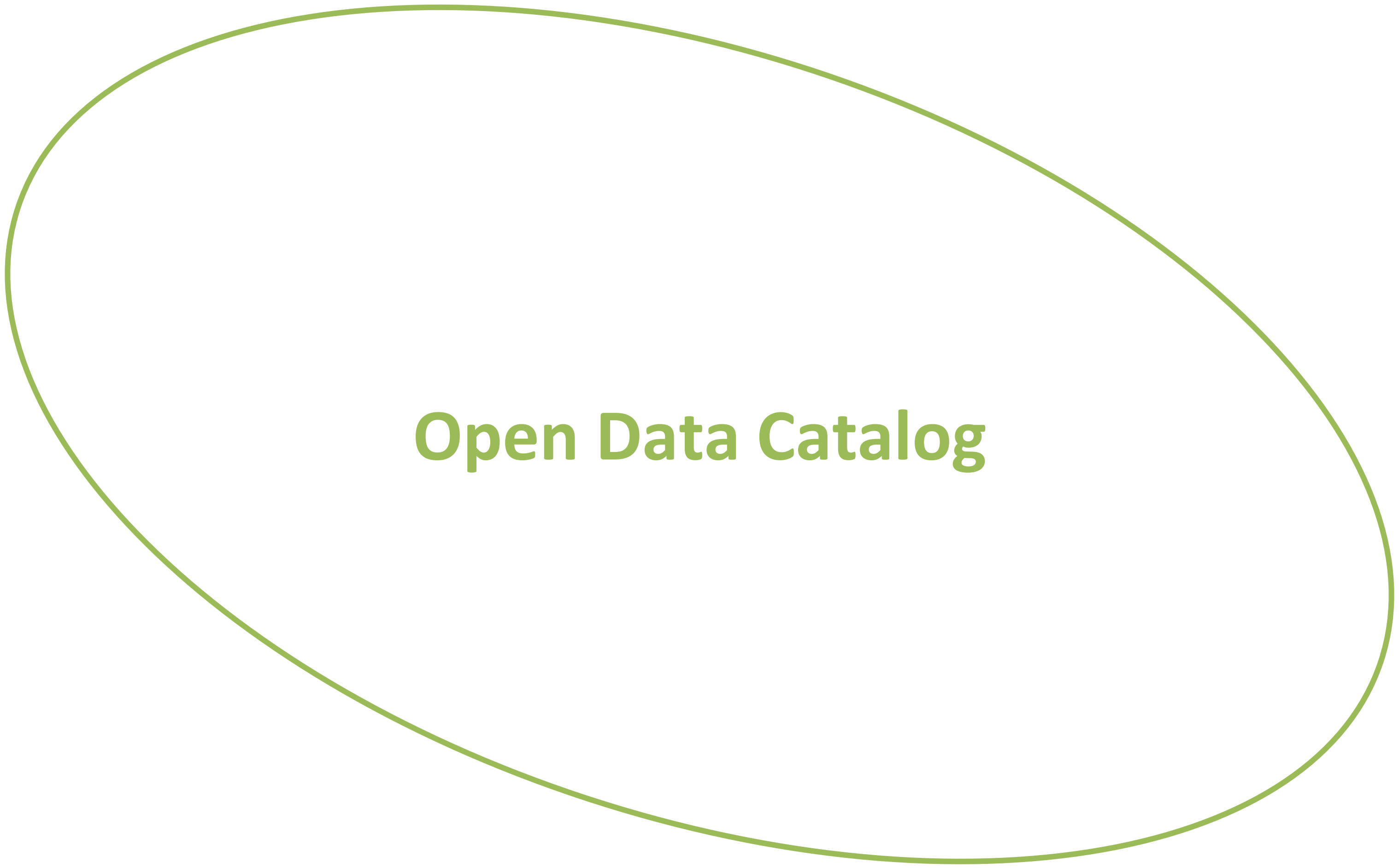
Three-tier Pattern



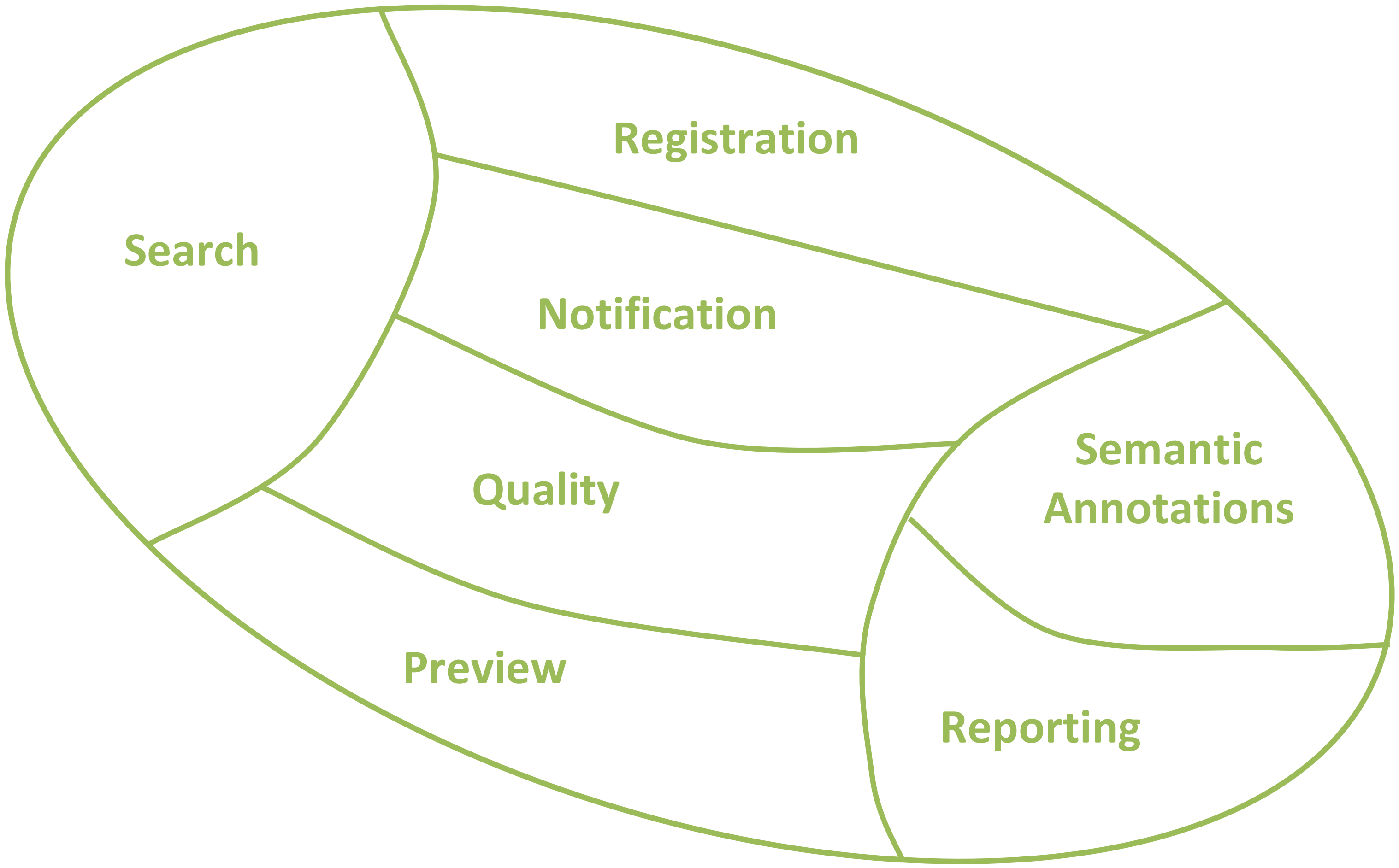
Three-tier Pattern

- Benefits - simplicity
  - separation of teams, platforms, servers
  - faster development
  - improved scalability, availability, security, modifiability and integrability
- Drawbacks
  - concentration on technical aspects of software lifecycle

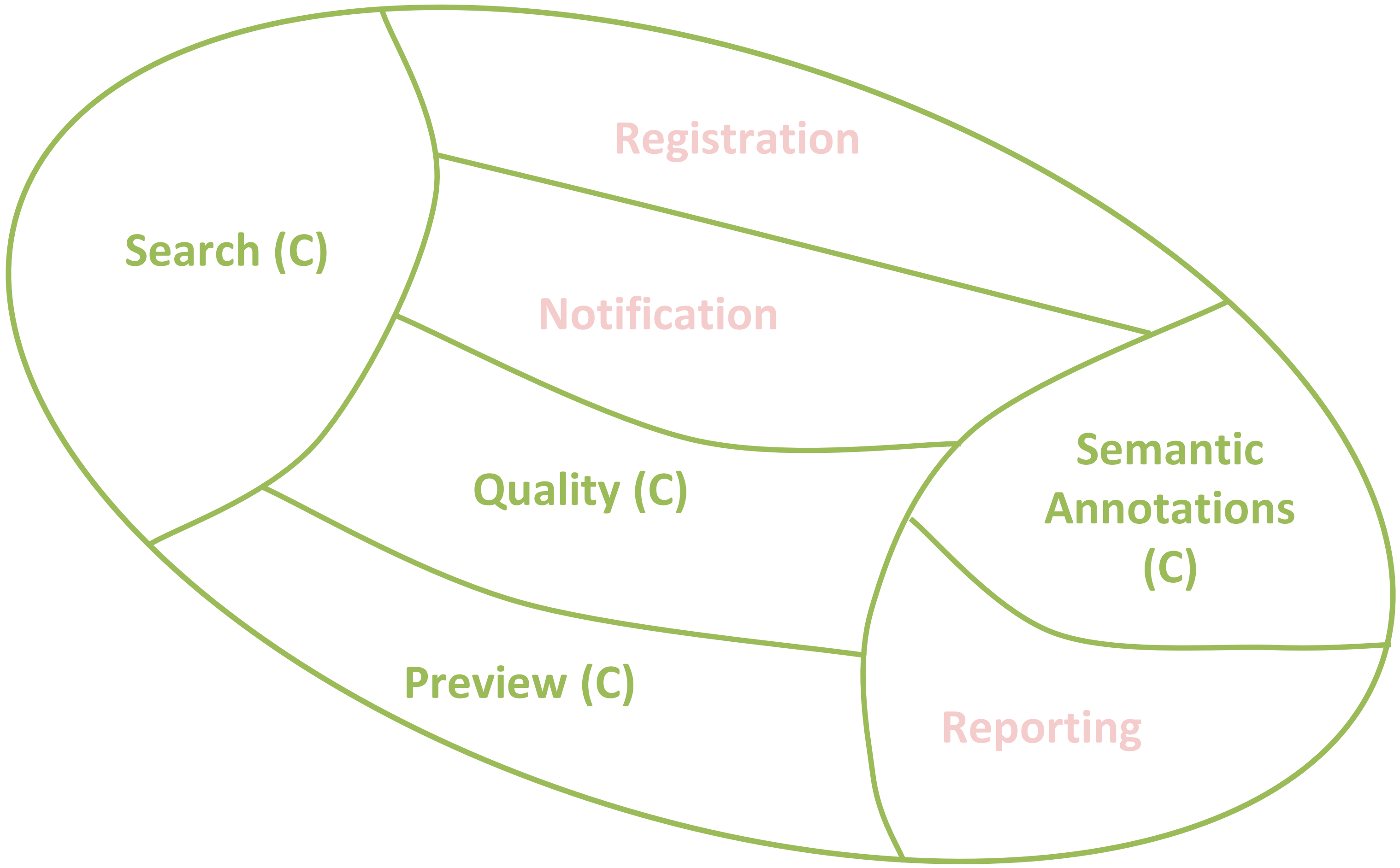
Problem Domain Overview



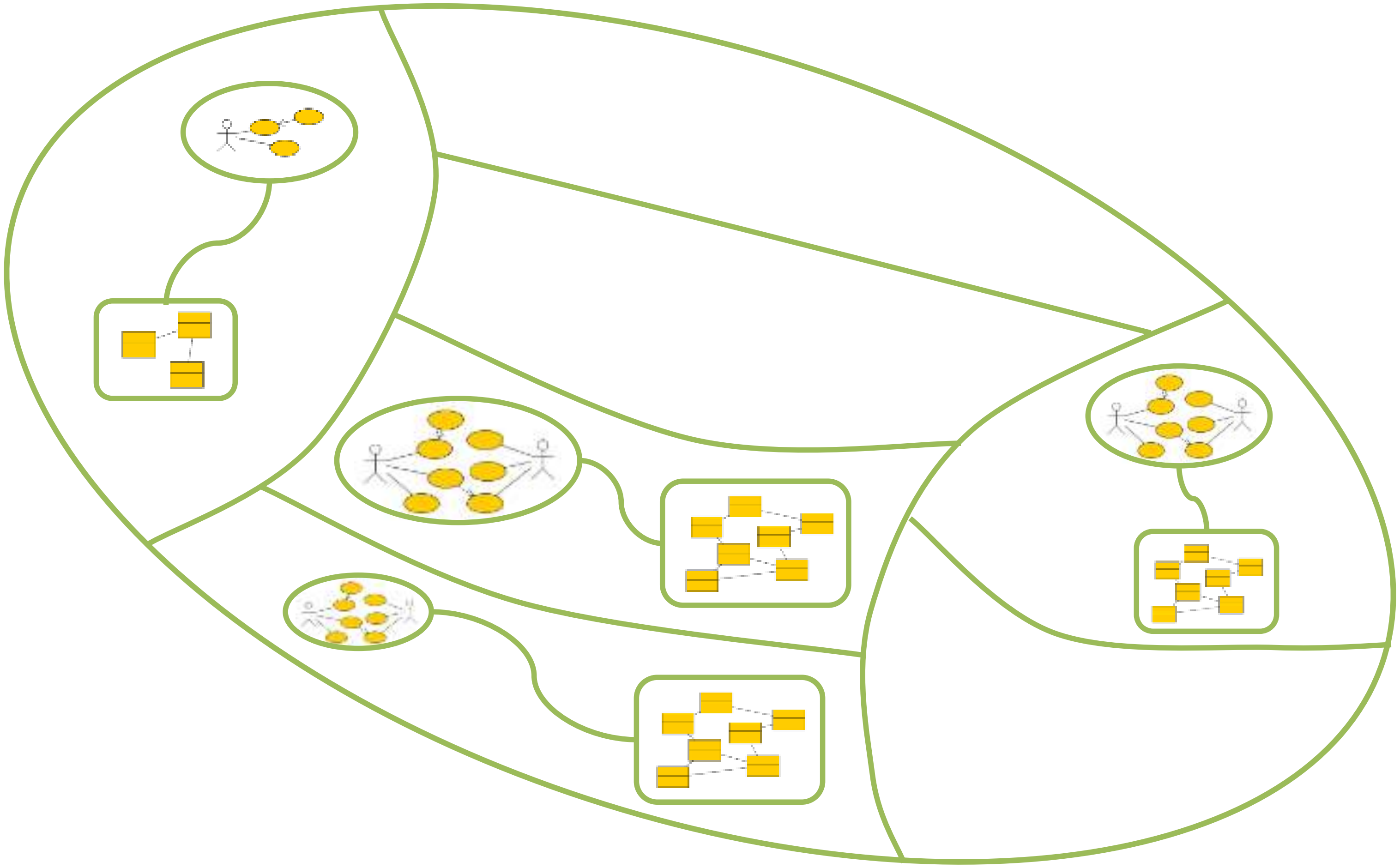
Subdomains


Registration
Search
Notification
Quality
Semantic Annotations
Preview
Reporting

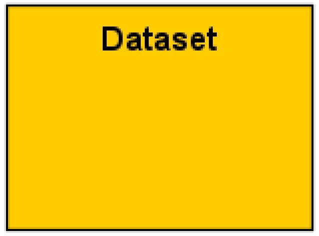
Core Domains

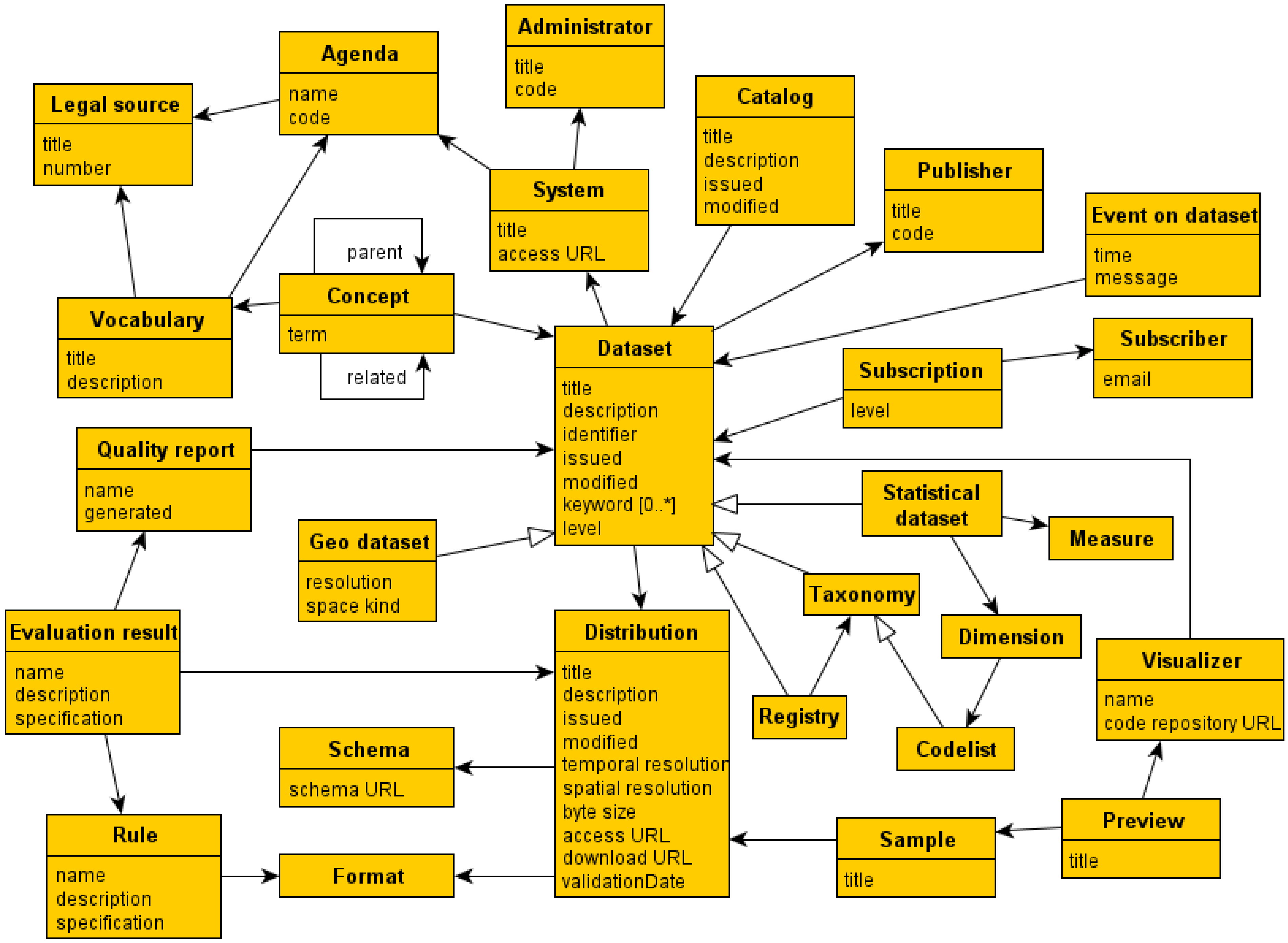


Modeling Problem Domain

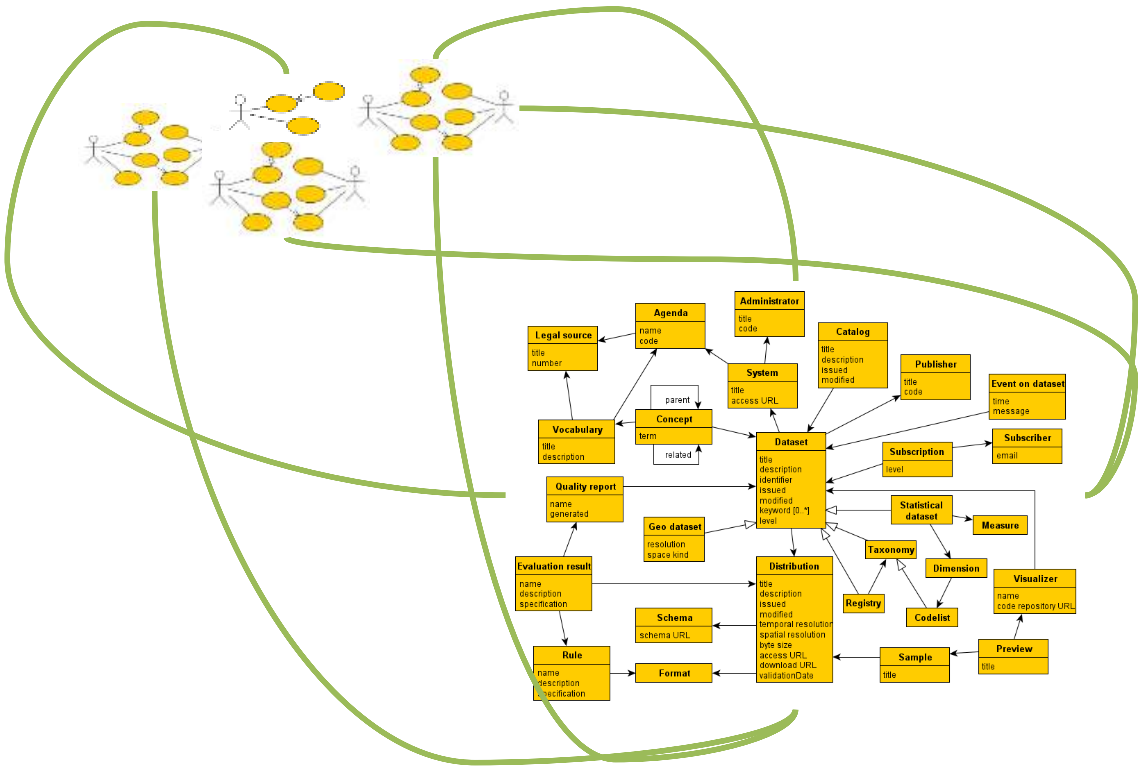


Modeling Problem Domain

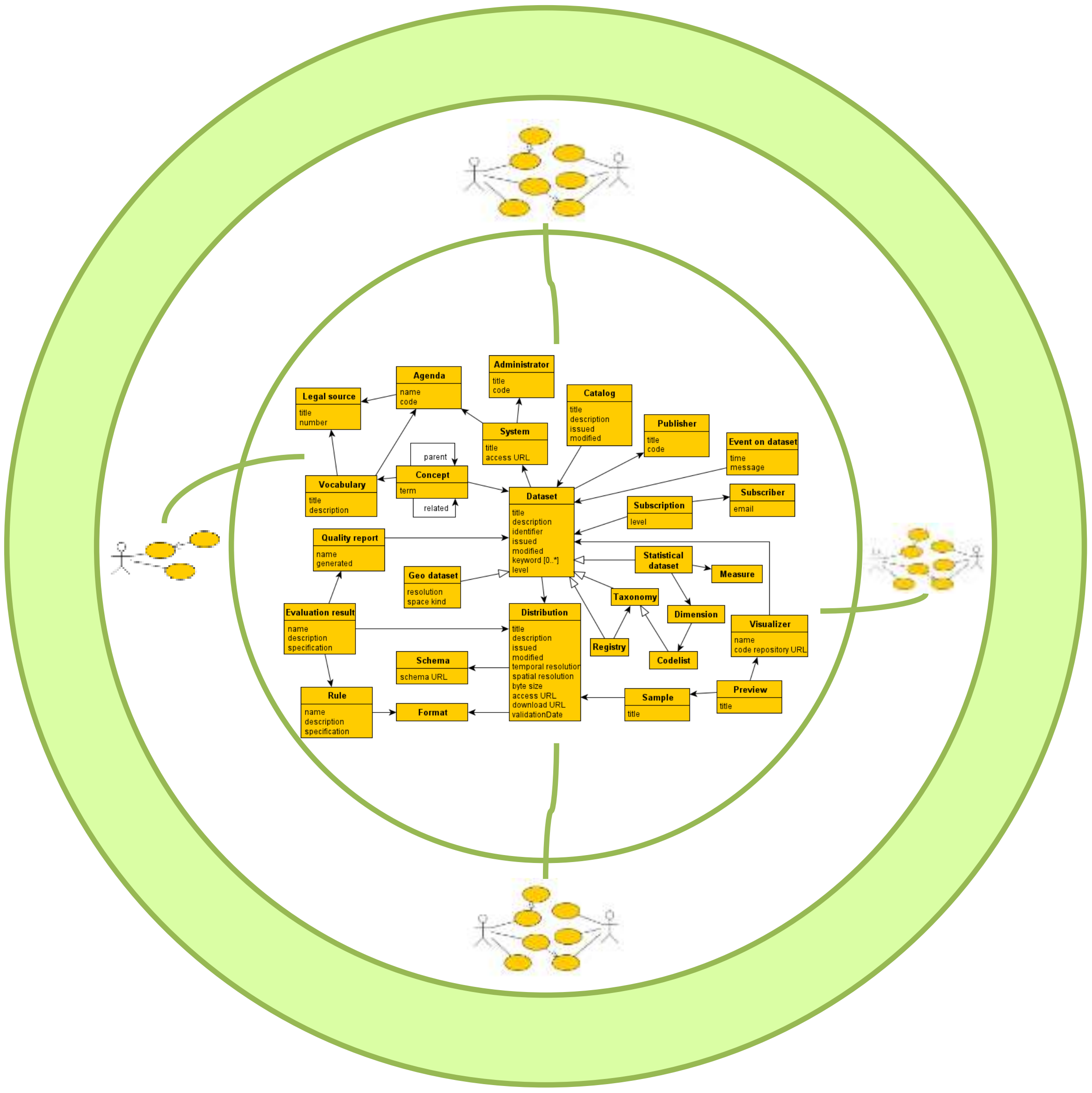



Administrator
title
code
Agenda
name
code
Legal source
title
number
Catalog
title
description
issued
modified
System
title
access URL
Publisher
title
code
Event on dataset
time
message
parent
Concept
term
related
Vocabulary
title
description
Dataset
title
description
identifier
issued
modified
keyword [0..*]
level
Subscriber
email
Subscription
level
Quality report
name
generated
Statistical dataset
Measure
Geo dataset
resolution
space kind
Taxonomy
Dimension
Evaluation result
name
description
specification
Distribution
title
description
issued
modified
temporal resolution
spatial resolution
byte size
access URL
download URL
validationDate
Visualizer
name
code repository URL
Registry
Codelist
Schema
schema URL
Rule
name
description
specification
Format
Sample
title
Preview
title

Presentation tier

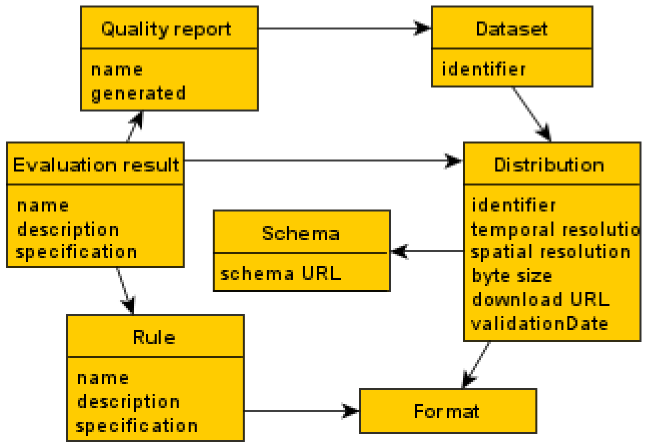


Data tier

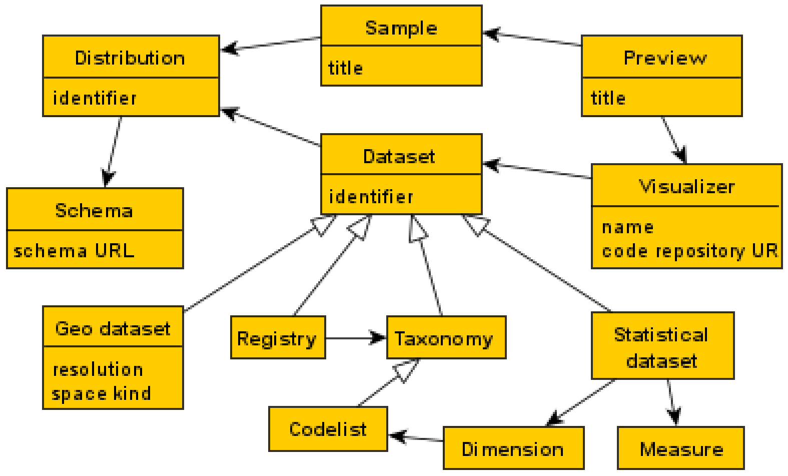


Administrator
title
code
Agenda
name
code
Legal source
title
number
Catalog
title
description
issued
modified
System
title
access URL
Publisher
title
code
Event on dataset
time
message
parent
Concept
term
related
Vocabulary
title
description
Dataset
title
description
identifier
issued
modified
keyword [0..*]
level
Subscription
level
Subscriber
email
Quality report
name
generated
Statistical dataset
Measure
Geo dataset
resolution
space kind
Taxonomy
Dimension
Evaluation result
name
description
specification
Distribution
title
description
issued
modified
temporal resolution
spatial resolution
byte size
access URL
download URL
validationDate
Registry
Visualizer
name
code repository URL
Codelist
Schema
schema URL
Rule
name
description
specification
Format
Sample
title
Preview
title

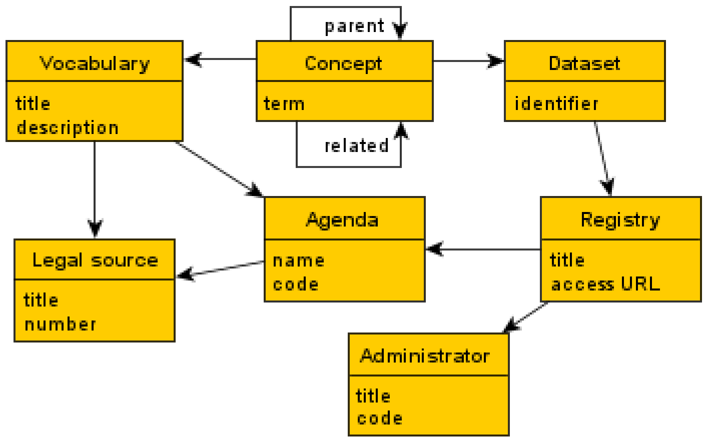
Quality Subdomain (C)

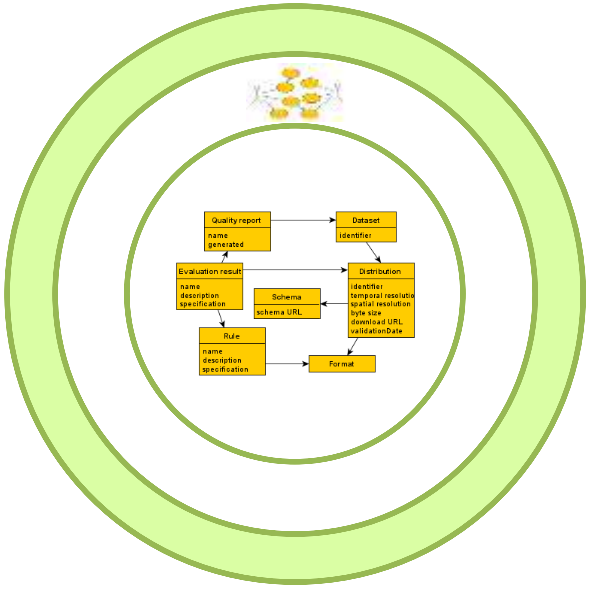


Preview Subdomain (C)

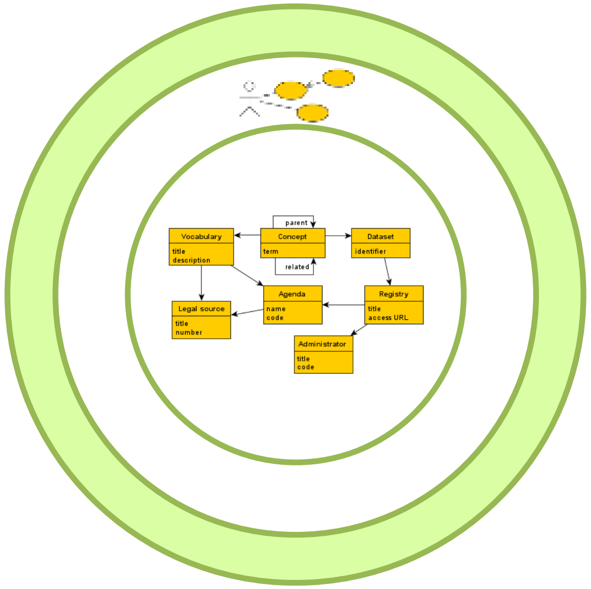


Semantic Annotations Subdomain (C)

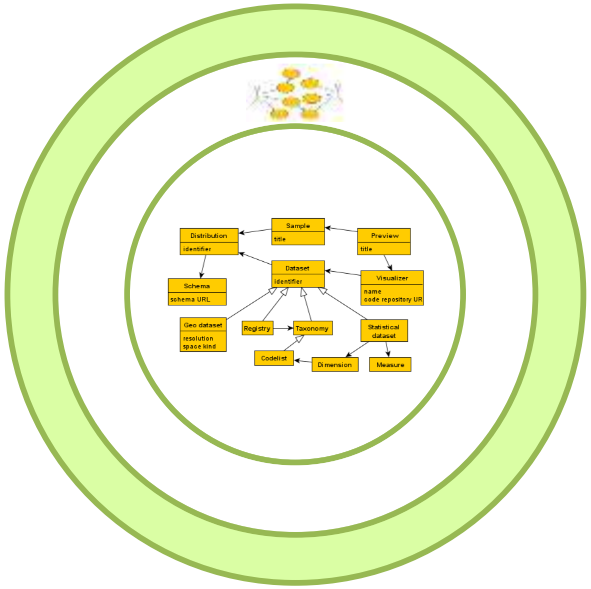




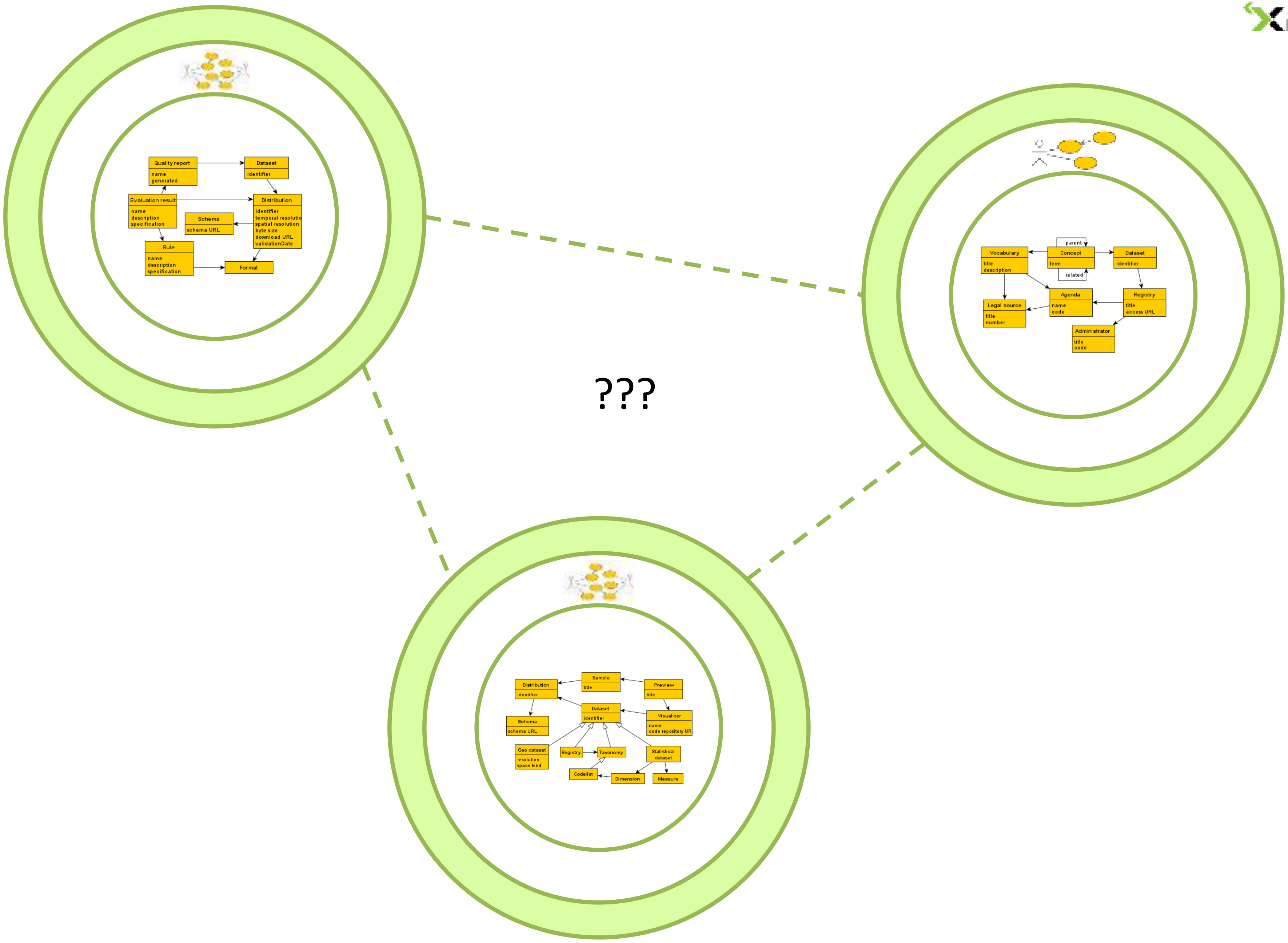
Quality report
name
generated
Dataset
identifier
Evaluation result
name
description
specification
Distribution
identifier
temporal resolutio
spatial resolution
byte size
download URL
validationDate
Schema
schema URL
Rule
name
description
specification
Format



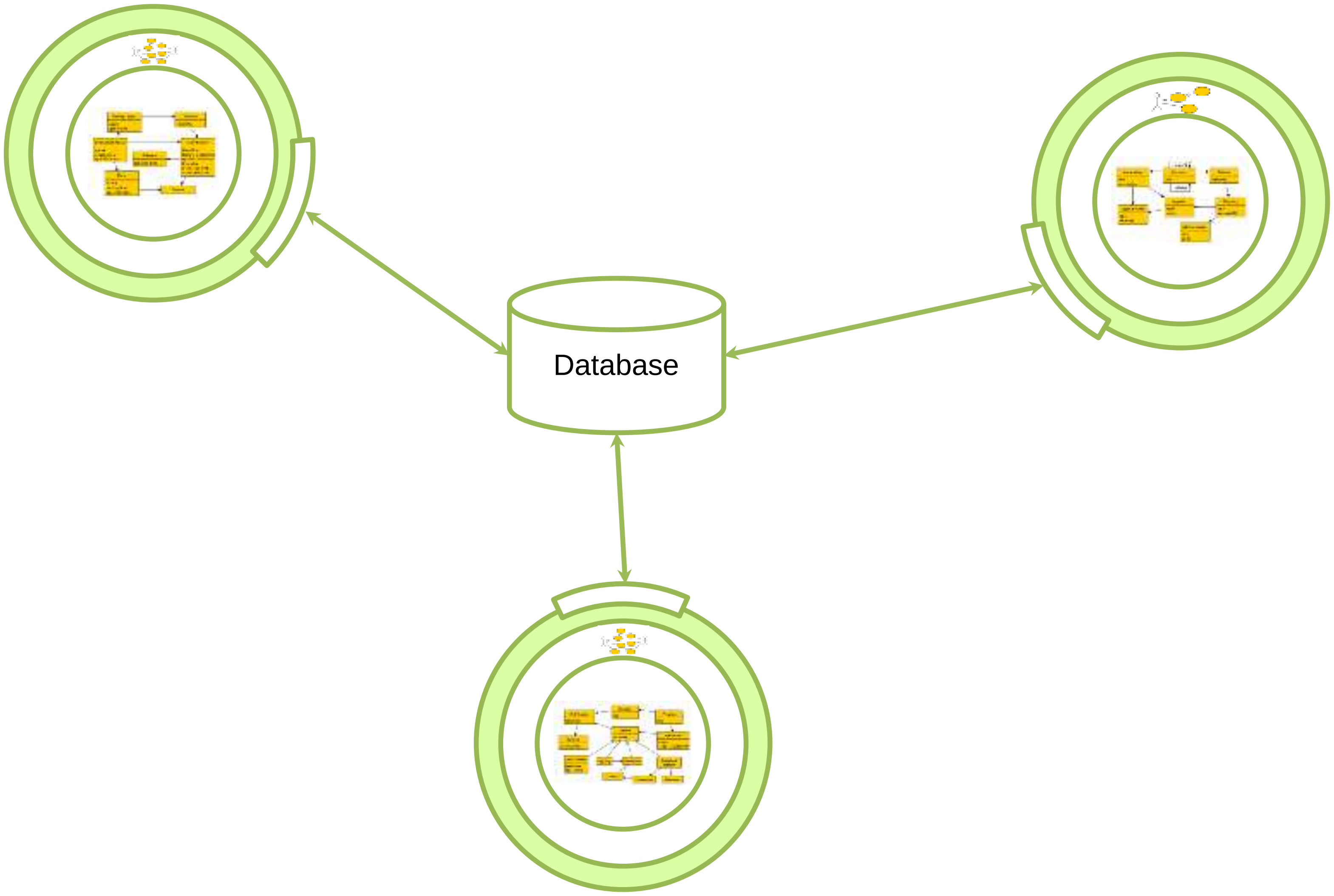
parent
Vocabulary
title
description
Concept
term
related
Dataset
identifier
Agenda
name
code
Registry
title
access URL
Legal source
title
number
Administrator
title
code

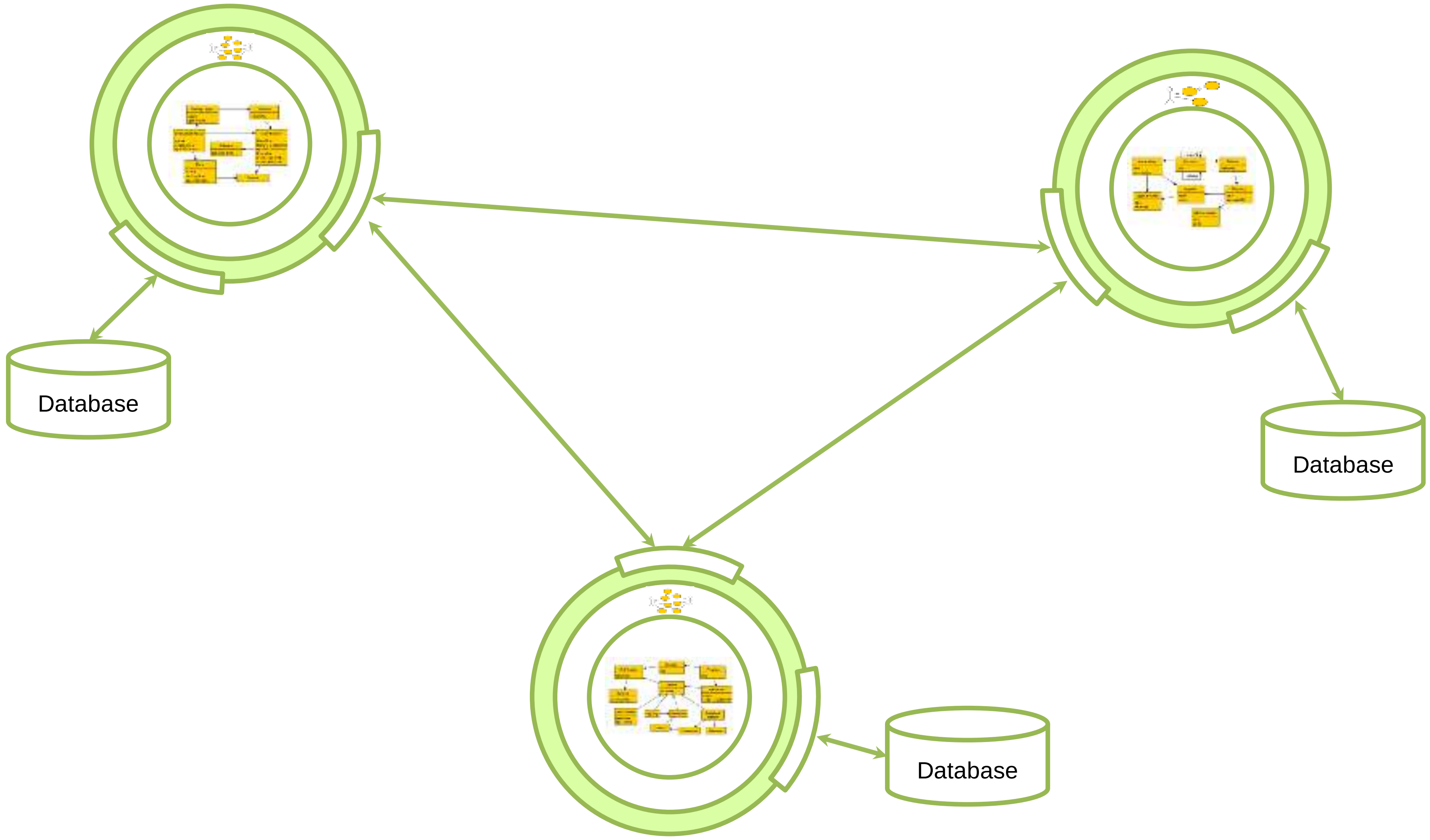


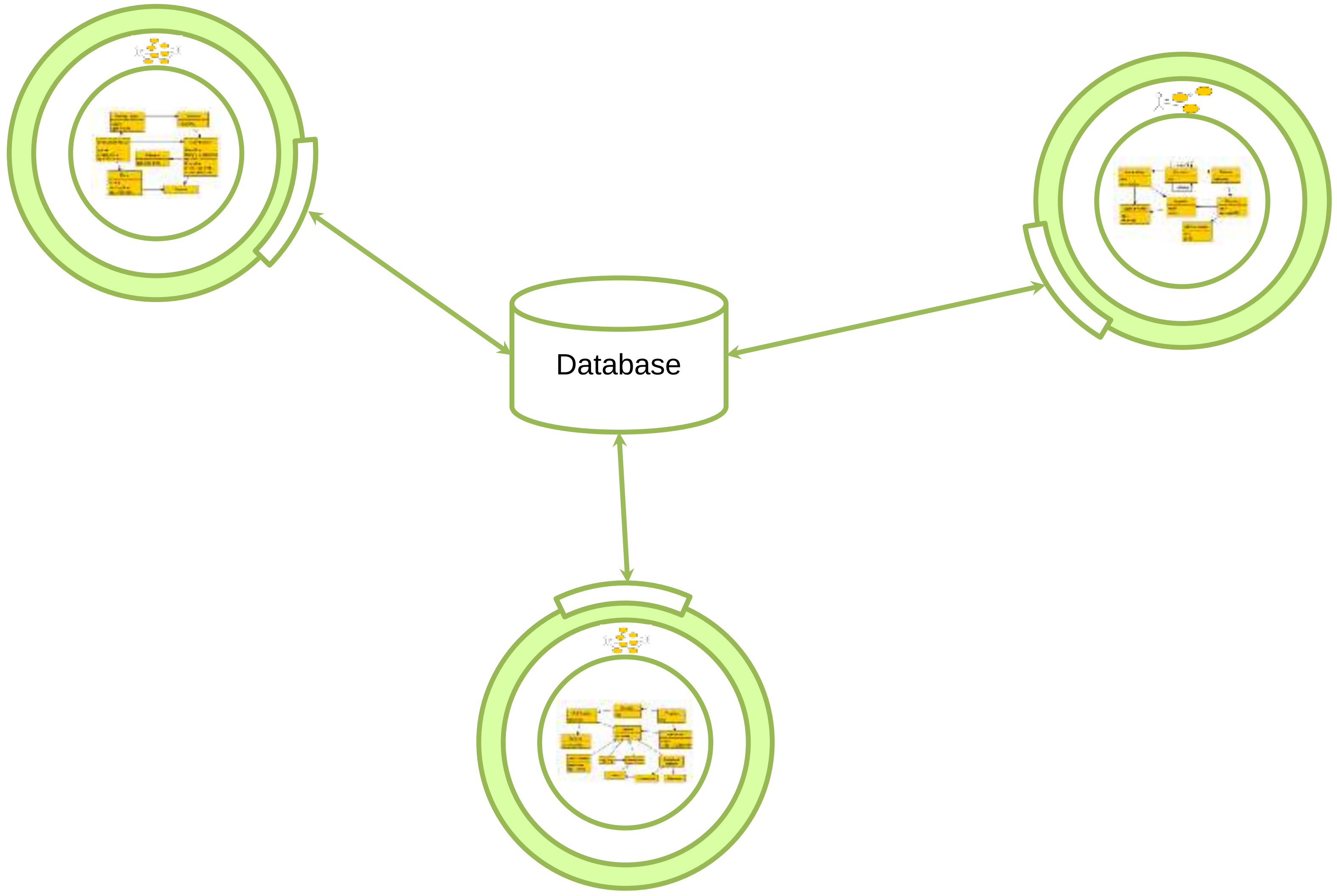
Distribution
identifier
Sample
title
Preview
title
Dataset
identifier
Visualizer
name
code repository UR
Schema
schema URL
Geo dataset
resolution
space kind
Registry
Taxonomy
Statistical dataset
Codelist
Dimension
Measure

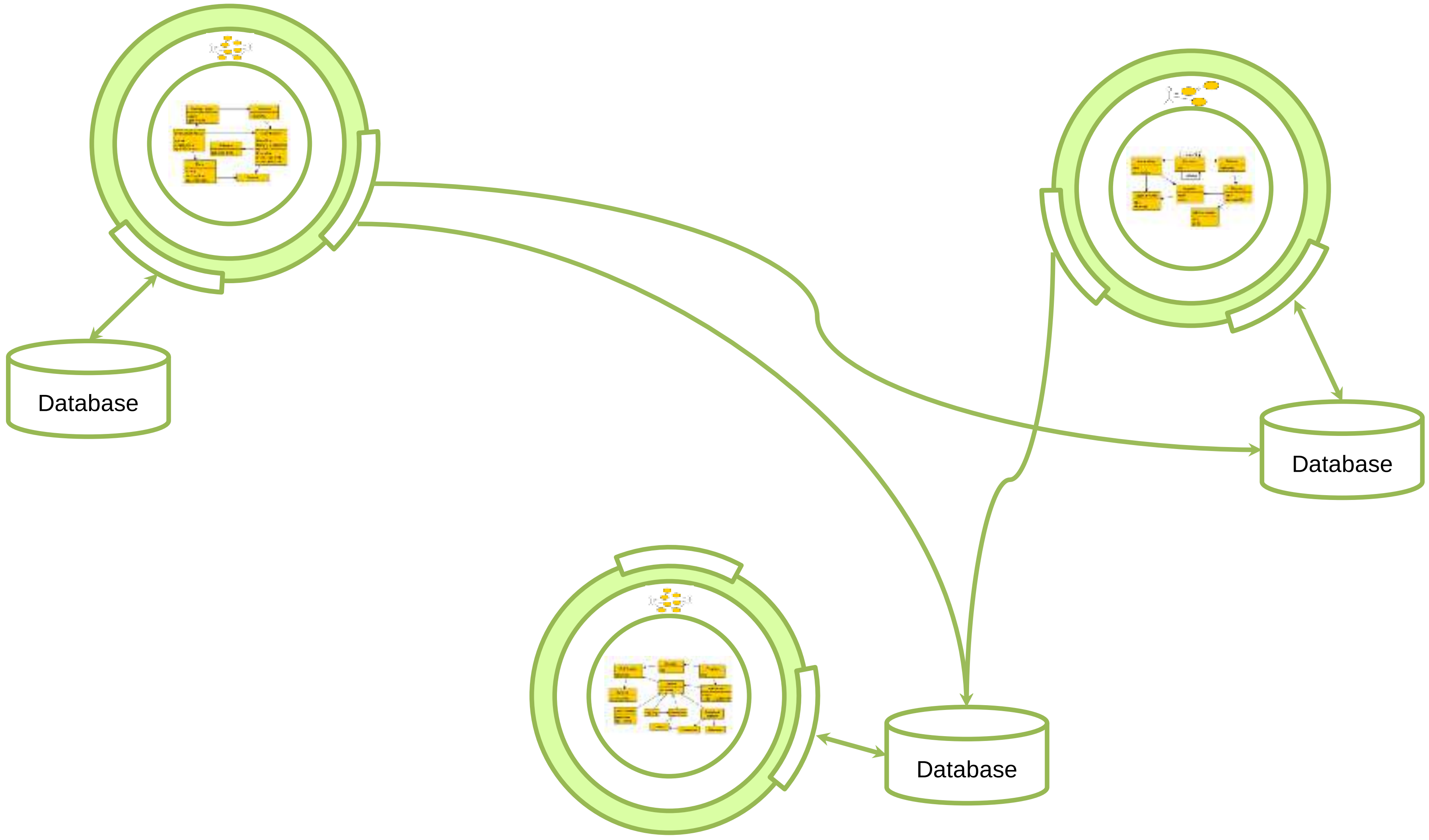

Quality report
name
generated
Dataset
identifier
Evaluation result
name
description
specification
Distribution
identifier
temporal resolutio
spatial resolution
byte size
download URL
validationDate
Schema
schema URL
Rule
name
description
specification
Format
parent
Vocabulary
title
description
Concept
term
related
Dataset
identifier
Agenda
name
code
Registry
title
access URL
Legal source
title
number
Administrator
title
code
Distribution
identifier
Sample
title
Preview
title
Dataset
identifier
Visualizer
name
code repository UR
Schema
schema URL
Geo dataset
resolution
space kind
Registry
Taxonomy
Statistical
dataset
Codelist
Dimension
Measure

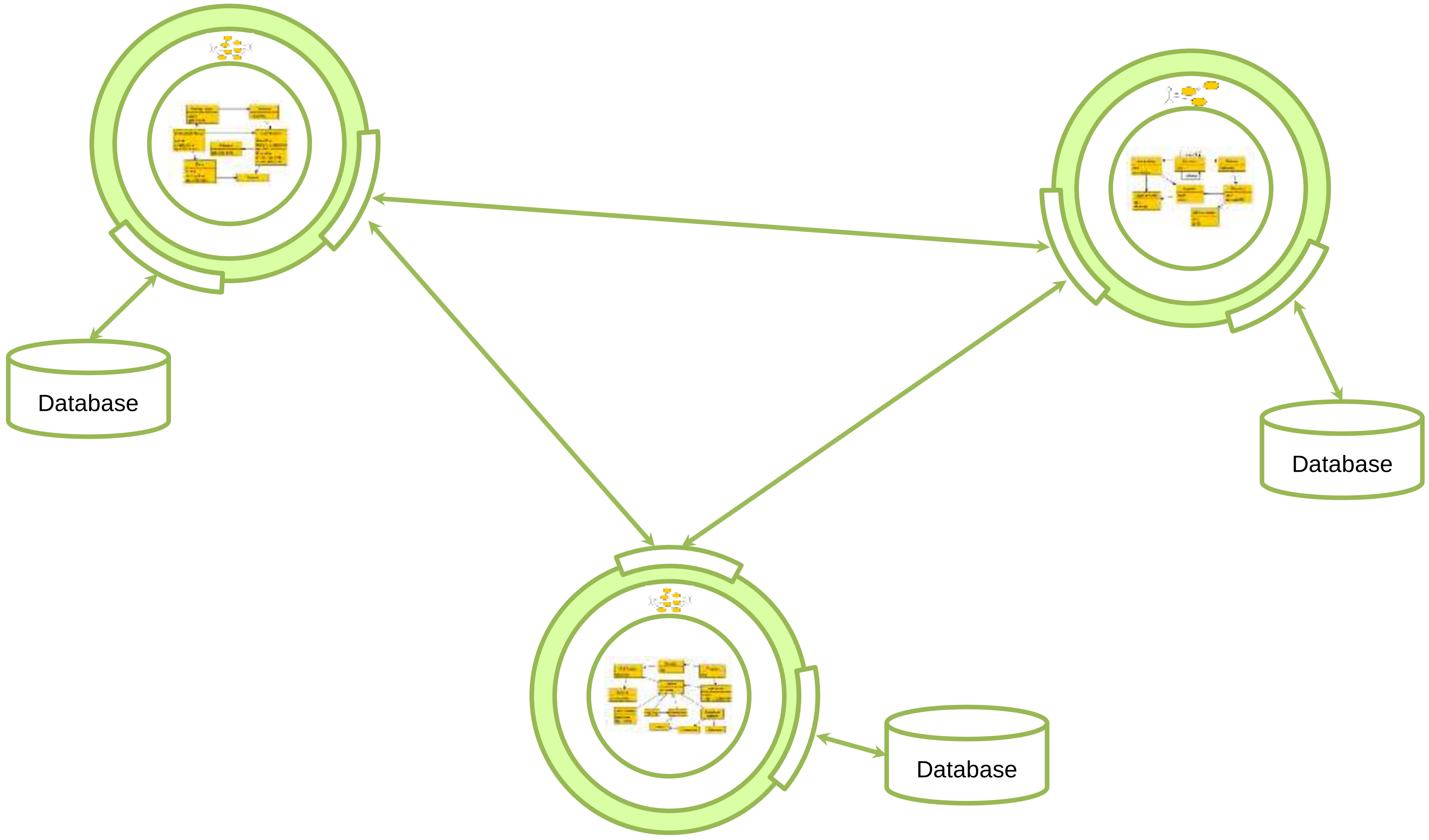
???


Database


Database
Database
Database


Database


Database
Database
Database


Database
Database
Database

Database integration

⍰

Integration through defined contracts

⍰

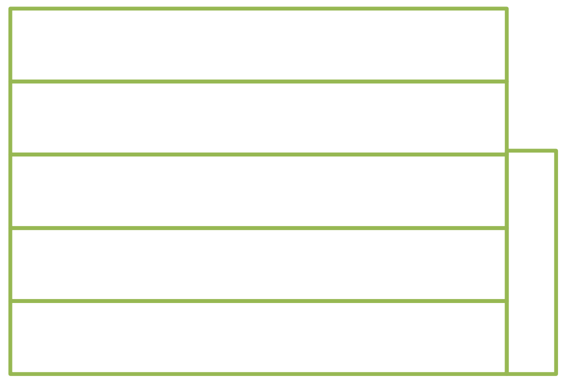
Incorporating anticorruption layers and open host services

⍰

Event-driven pattern

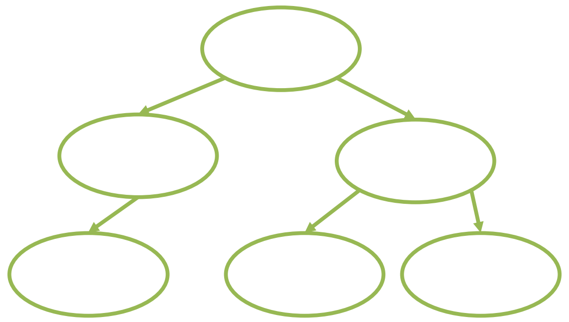
Layer Pattern

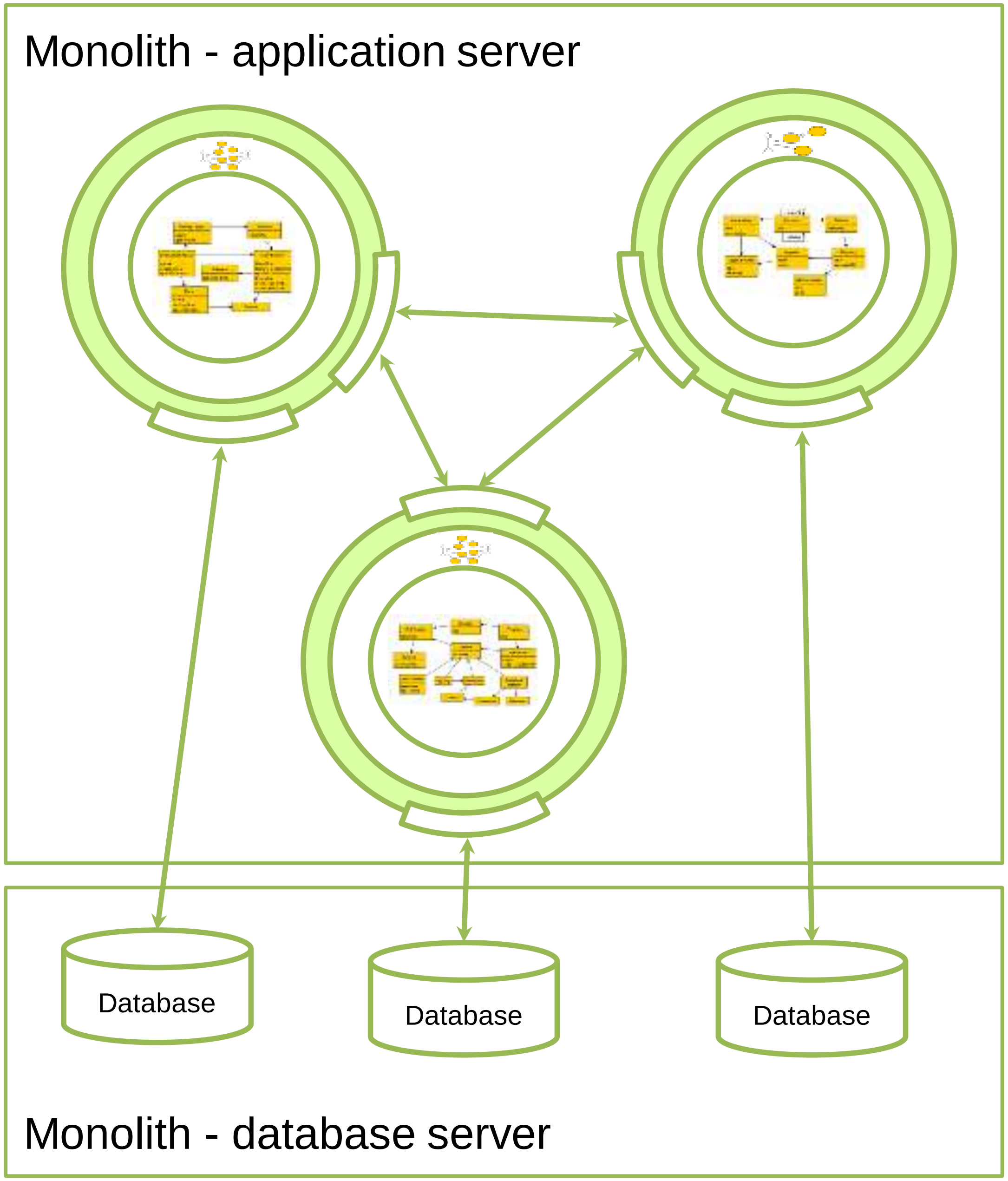
Layered Modules

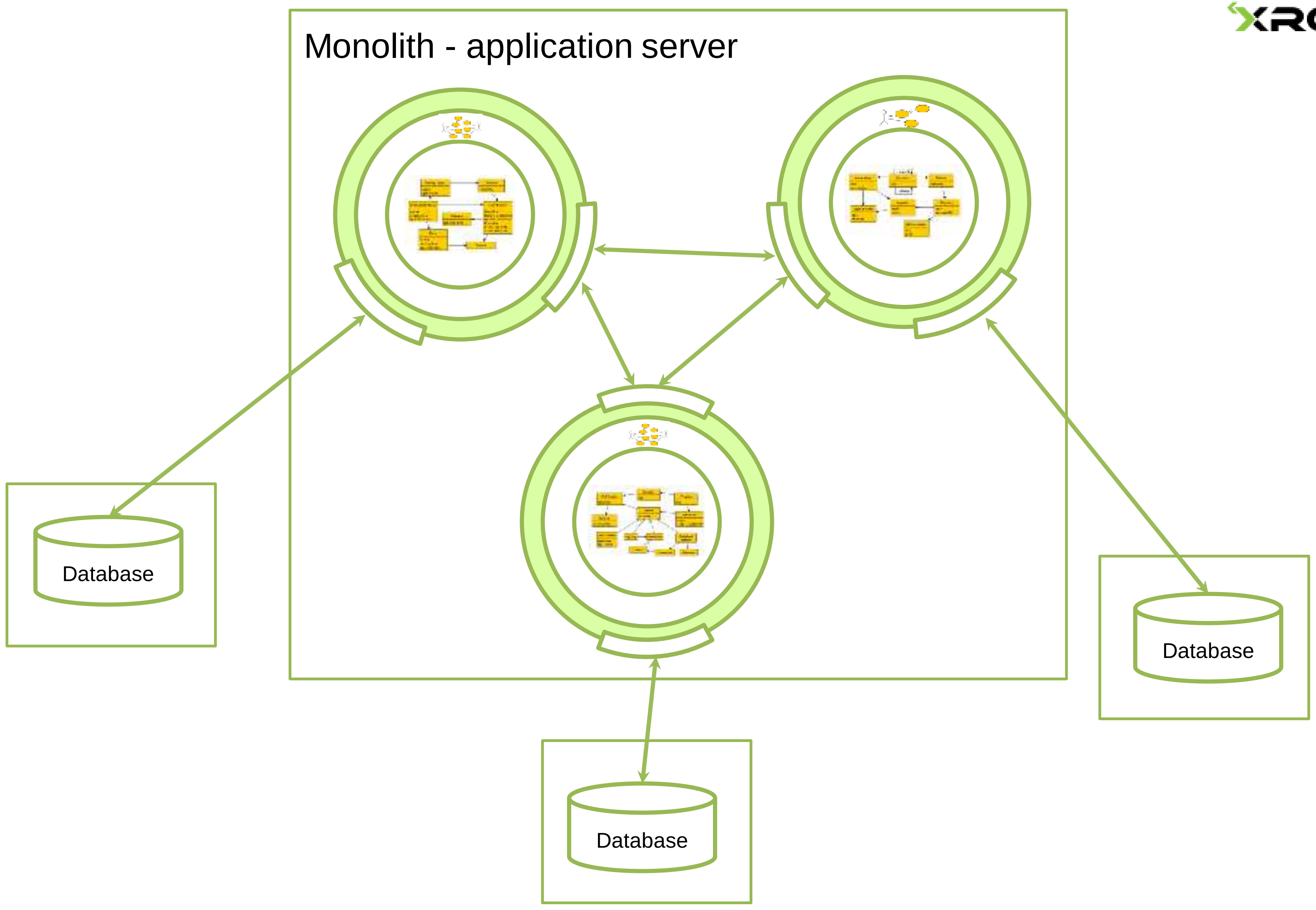


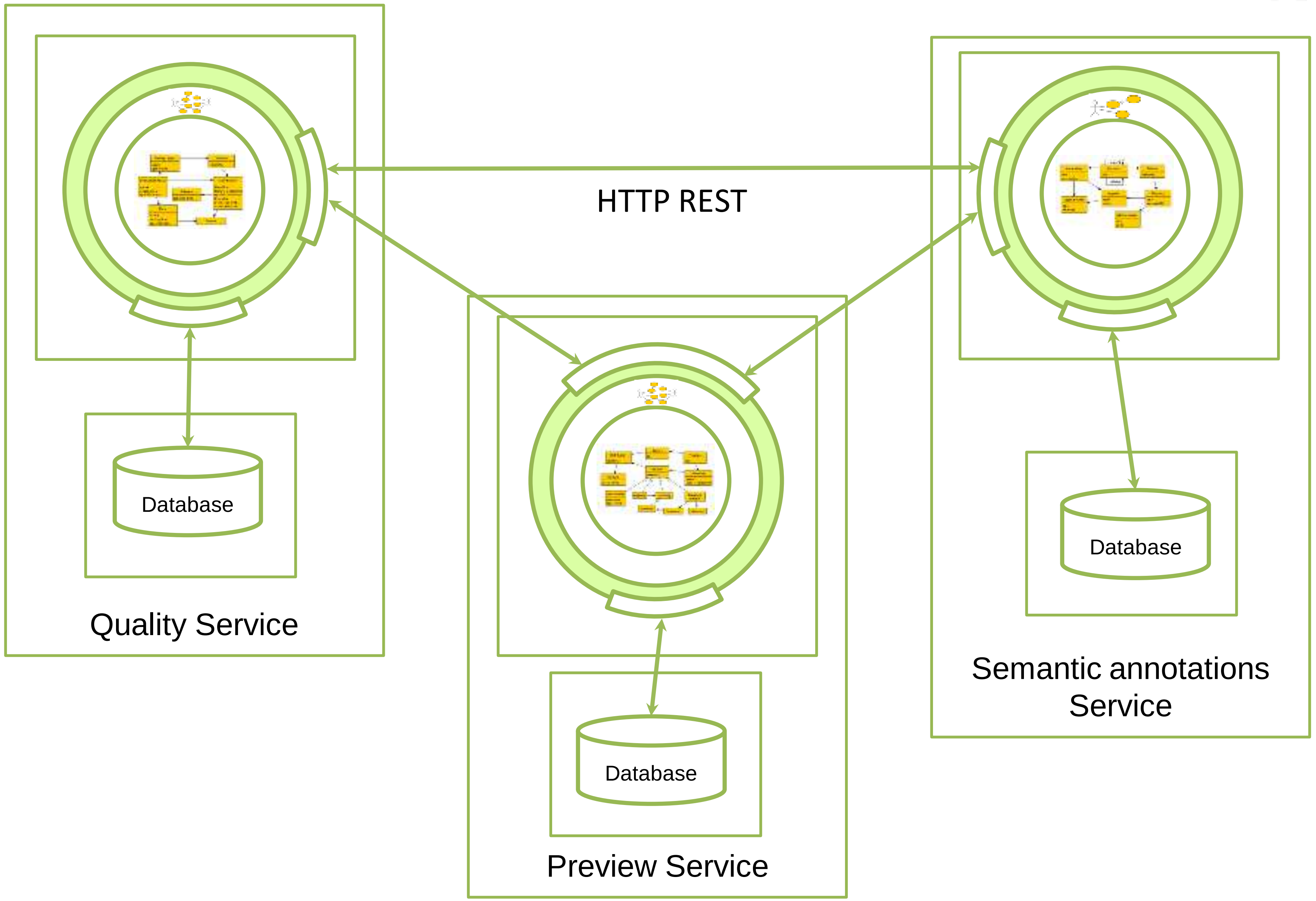
Monolithic Component

Distributed Network Components




Monolith - application server
Database
Database
Database
Monolith - database server


Monolith - application server
Database
Database
Database

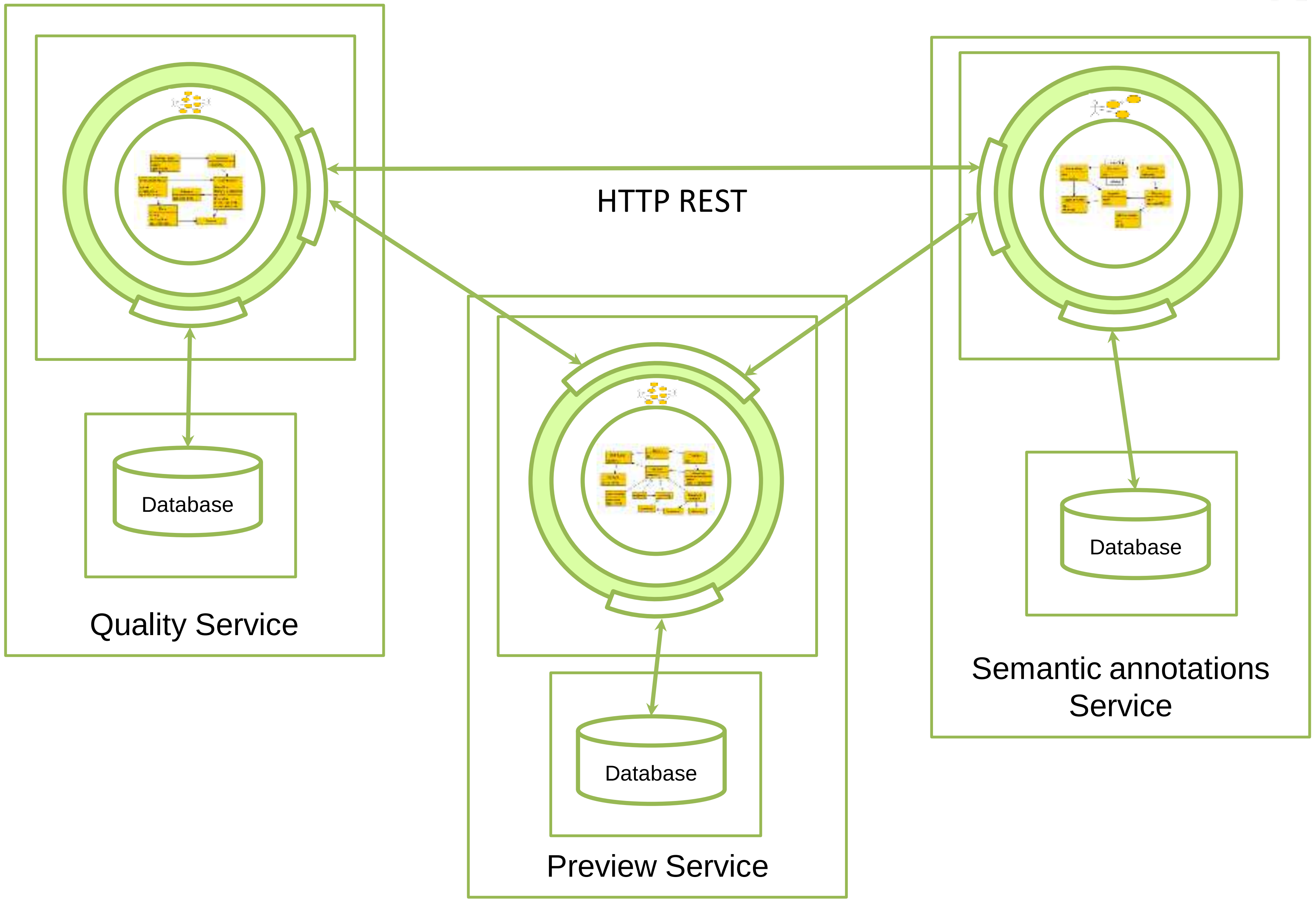

HTTP REST
Database
Quality Service
Database
Preview Service
Database
Semantic annotations Service

Service-oriented architecture (SOA)

*“**Service-oriented architecture (SOA)** is an architectural pattern for building software systems by reusing and composing functional components called **services**. A service is independent of its surrounding environment (formed by other services and service clients) and provides a certain functionality which is required by the business.”*

8 key SOA principles

- standardization
- loose-coupling
- abstraction
- reusability
- autonomy
- statelessness
- discoverability
- composability


HTTP REST
Database
Quality Service
Database
Preview Service
Database
Semantic annotations Service

Microservices pattern

- domain-driven architecture
- strictly autonomous runtime services
- small
- sagas for data consistency