

Fundamentals of aRcHiTEcTuRE StYLeS & pAtTERNs

ThoughtWorks®

NEAL FORD

Director / Software Architect / Meme Wrangler



@neal4d

<http://nealford.com>

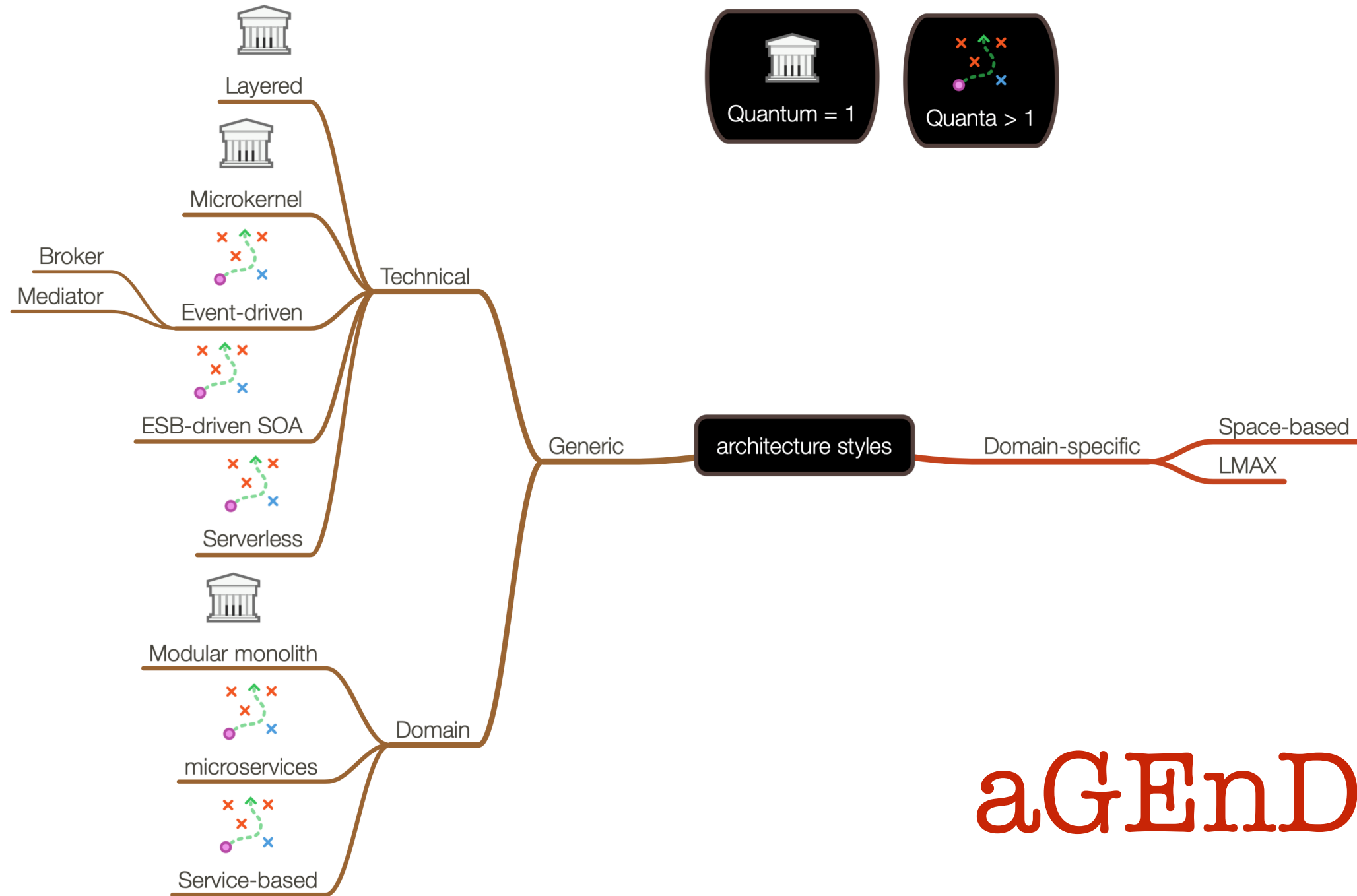
O'REILLY®



**Fundamentals of
Software
Architecture**

An Engineering Approach

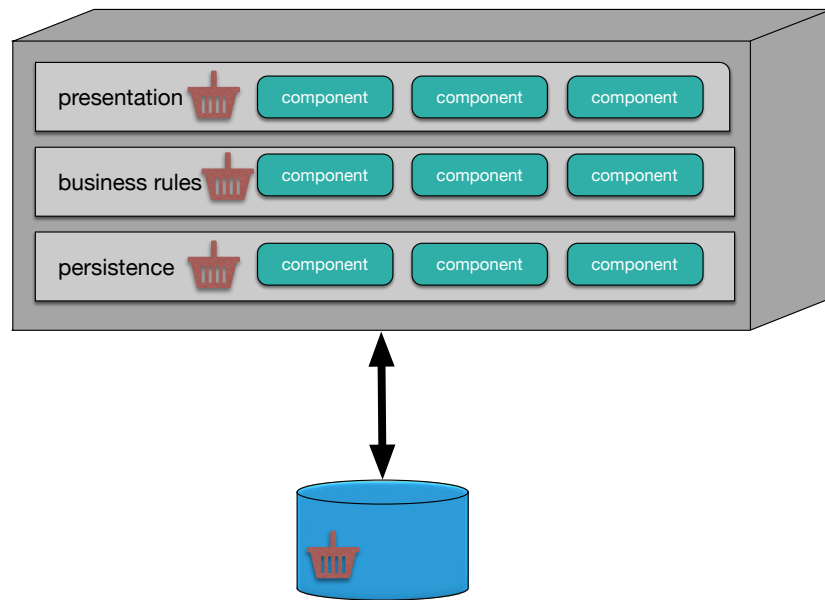
Mark Richards & Neal Ford



aGEnDA

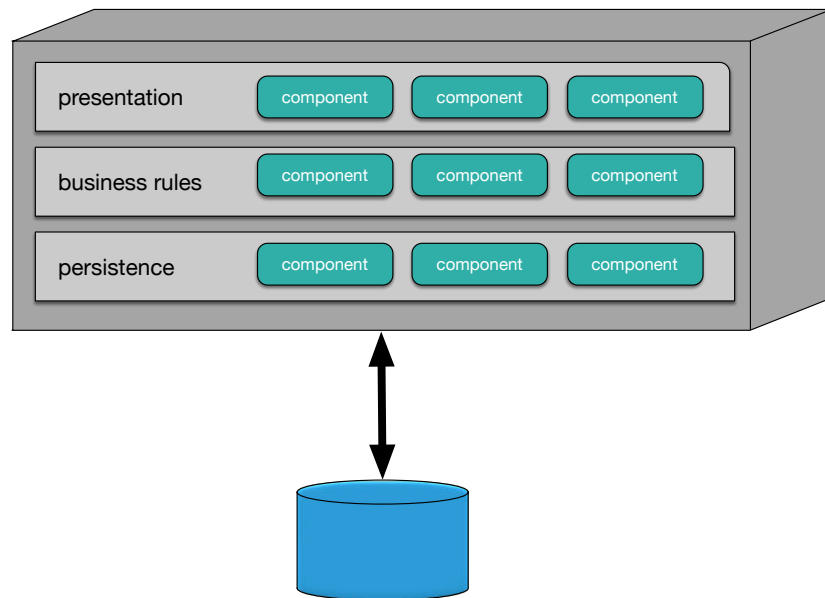
Top Level Partitioning

technical partitioning

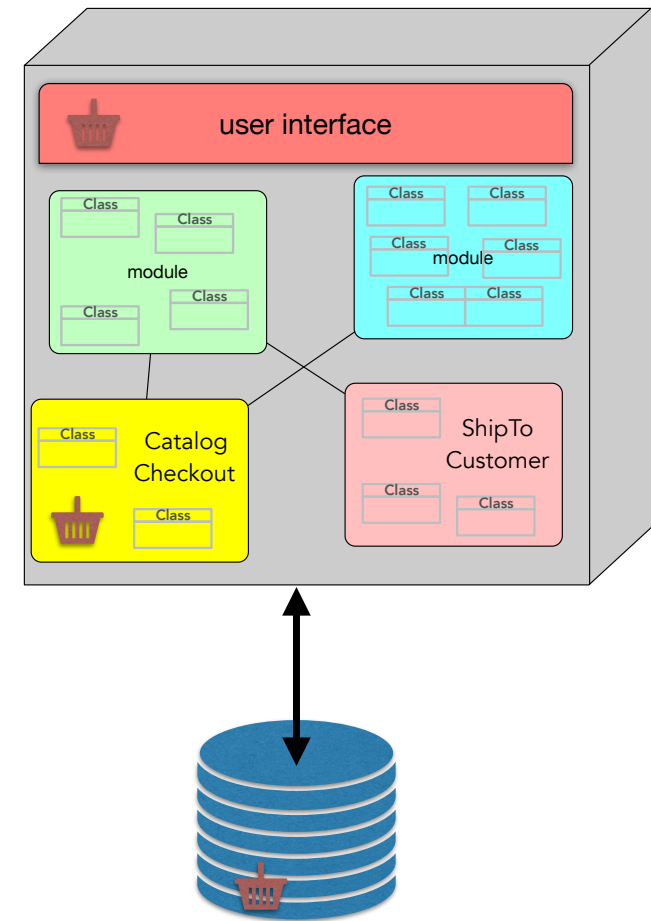


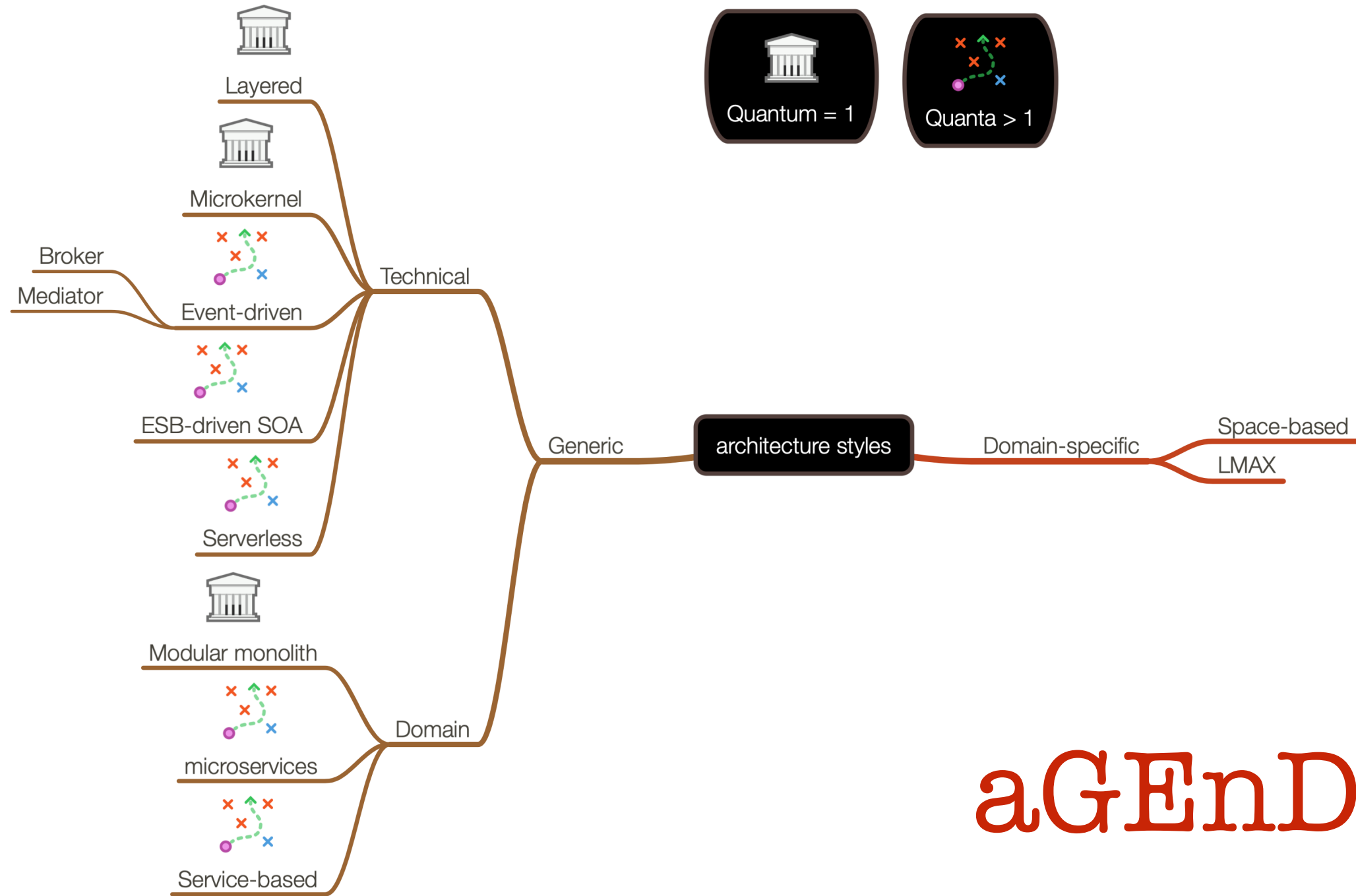
Top Level Partitioning

technical partitioning



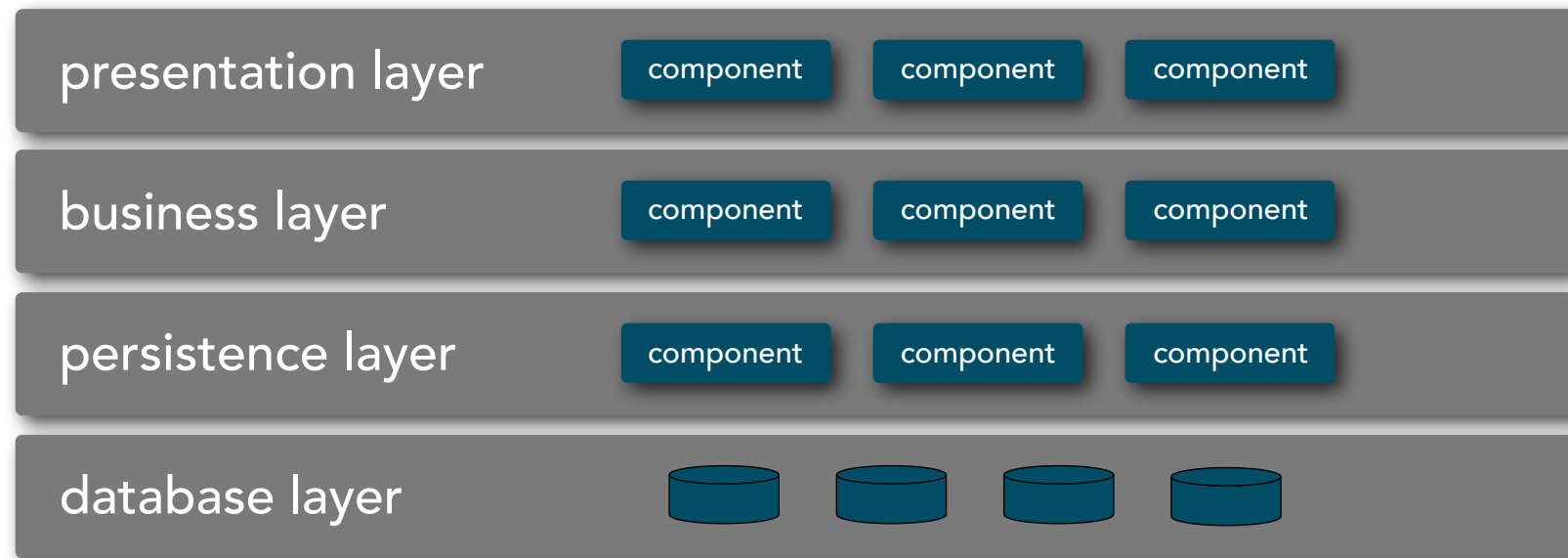
domain partitioning



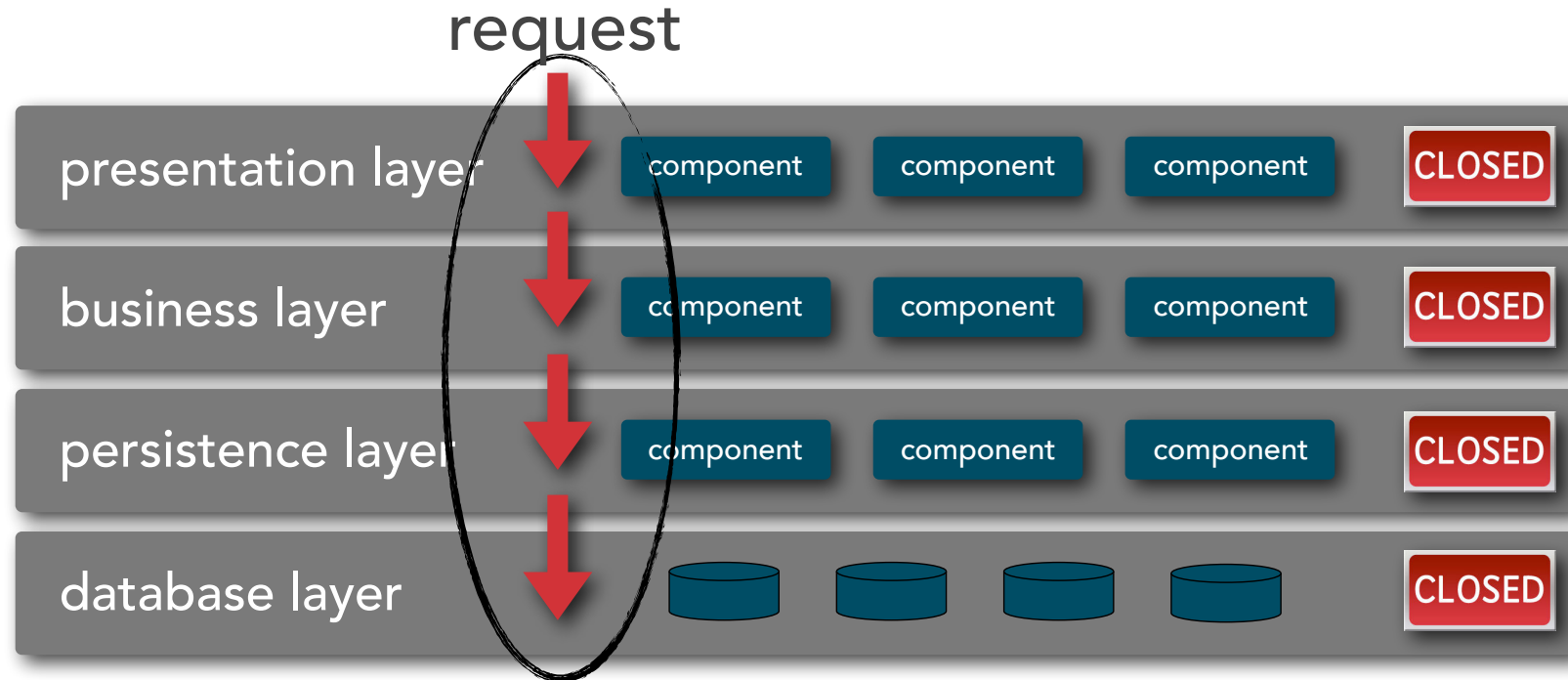


aGEnDA

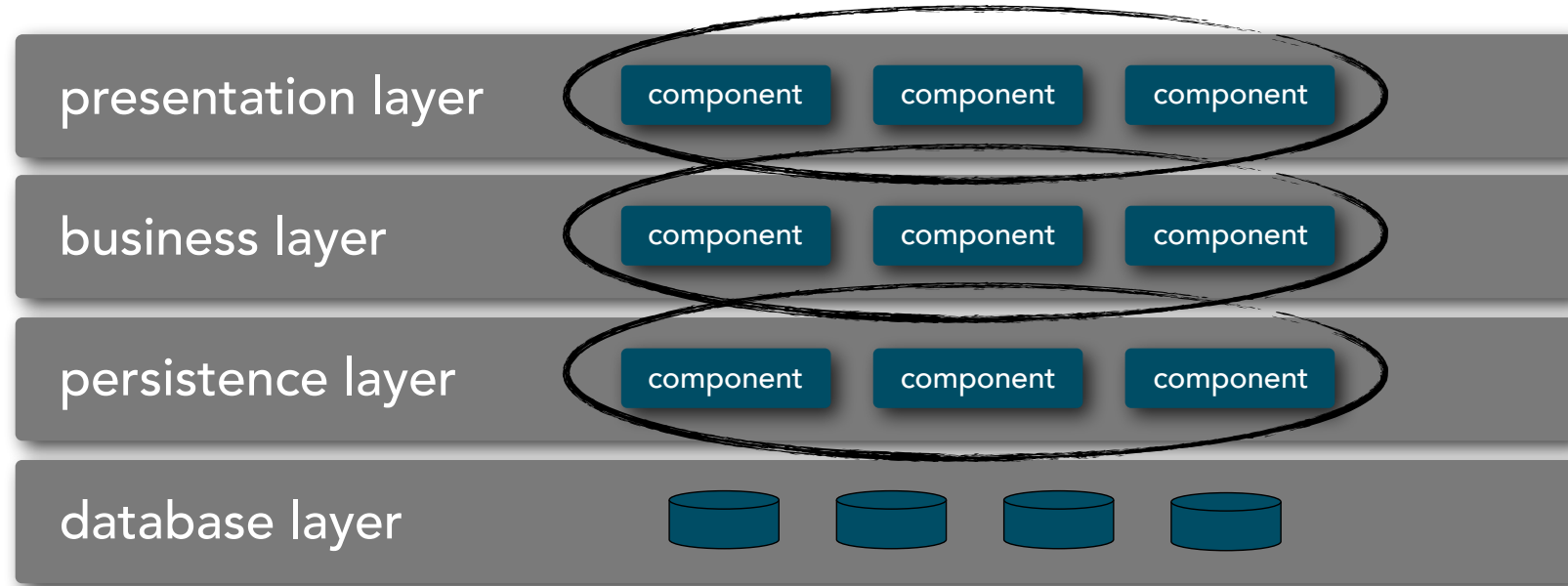
Layered Architecture



Layered Architecture



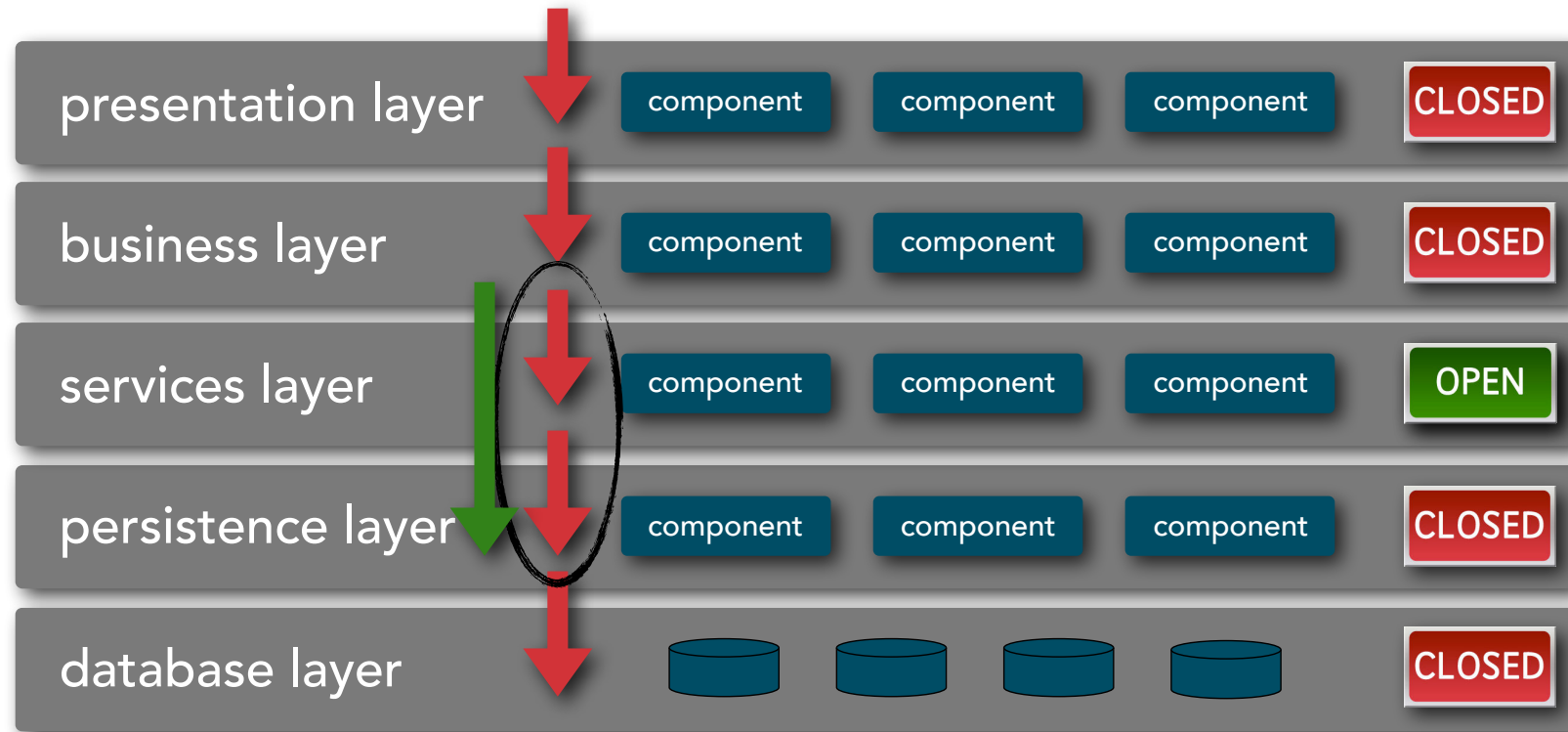
Layered Architecture



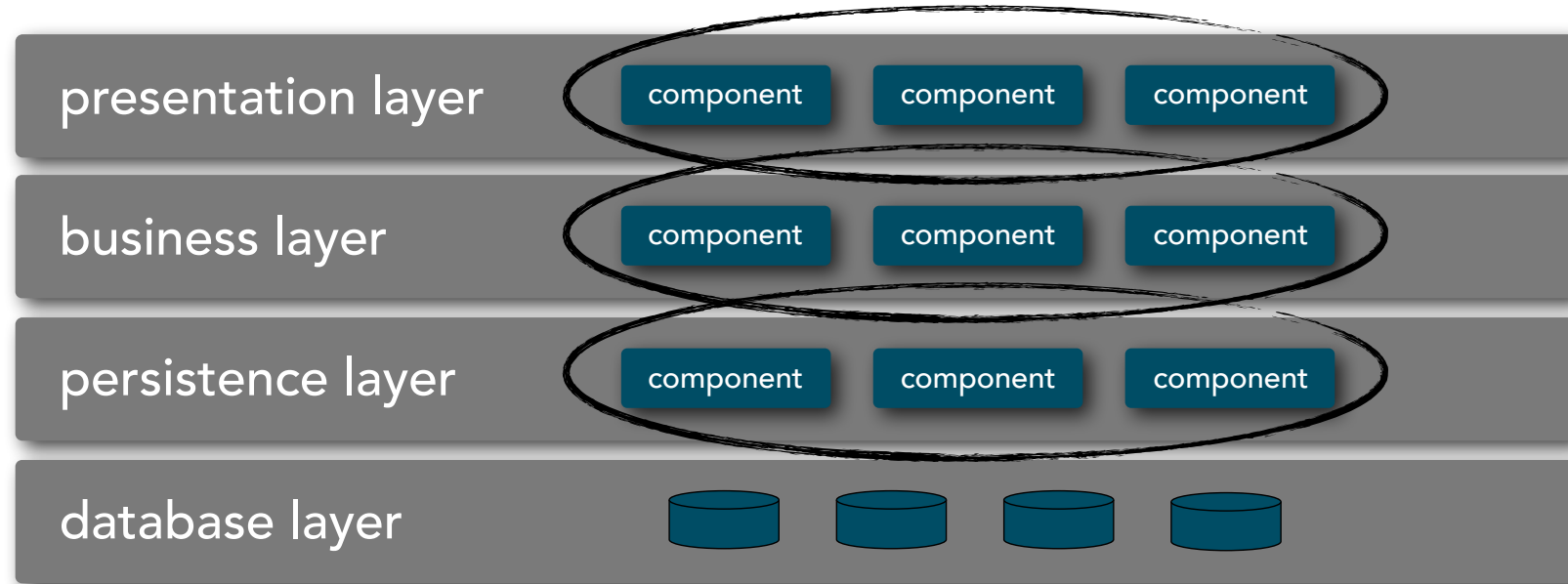
separation of concerns

Layered Architecture

hybrids and variants

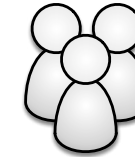


Layered Architecture

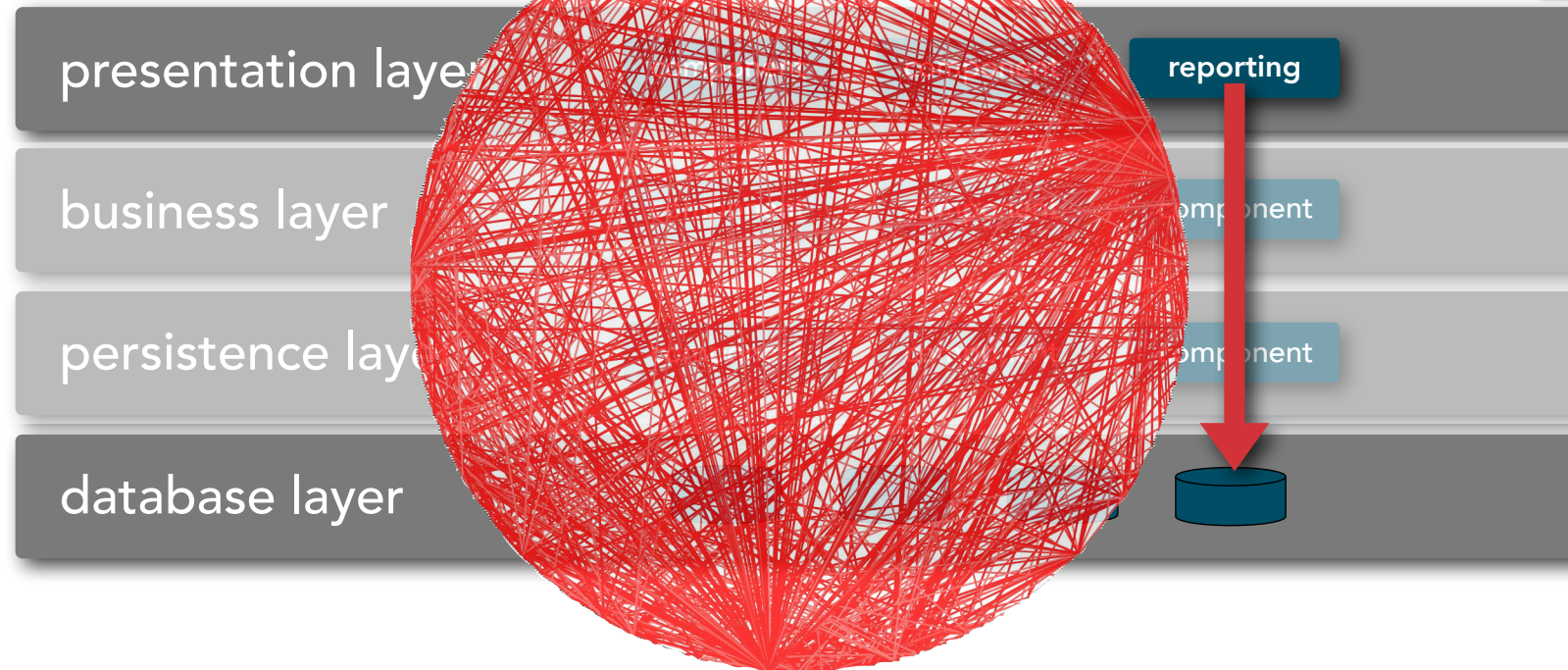


separation of concerns

Pattern Governance



We need direct access to the database for performance reasons.



```
public class LayerDependencyRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
    }

    @Test
    public void services_should_not_access_controllers() {
        noClasses().that().resideInAPackage("..service..")
            .should().accessClassesThat().resideInAPackage("..controller..").check(classes);
    }

    @Test
    public void persistence_should_not_access_services() {
        noClasses().that().resideInAPackage("..persistence..")
            .should().accessClassesThat().resideInAPackage("..service..").check(classes);
    }

    @Test
    public void services_should_only_be_accessed_by_controllers_or_other_services() {
        classes().that().resideInAPackage("..service..")
            .should().onlyBeAccessed().byAnyPackage("..controller..", "..service..").check(classes);
    }
}
```

layer dependency

NetArchTest

<https://github.com/BenMorris/NetArchTest/>

```
// Controllers should not directly reference repositories
var result = Types.InCurrentDomain()
    .That()
    .ResideInNamespace("NetArchTest.SampleLibrary.Presentation")
    .ShouldNot()
    .HaveDependencyOn("NetArchTest.SampleLibrary.Data")
    .GetResult().IsSuccessful;
```

📶 LIVE ONLINE TRAINING

Automating Architectural Governance using Fitness Functions

Agile Engineering in Architecture



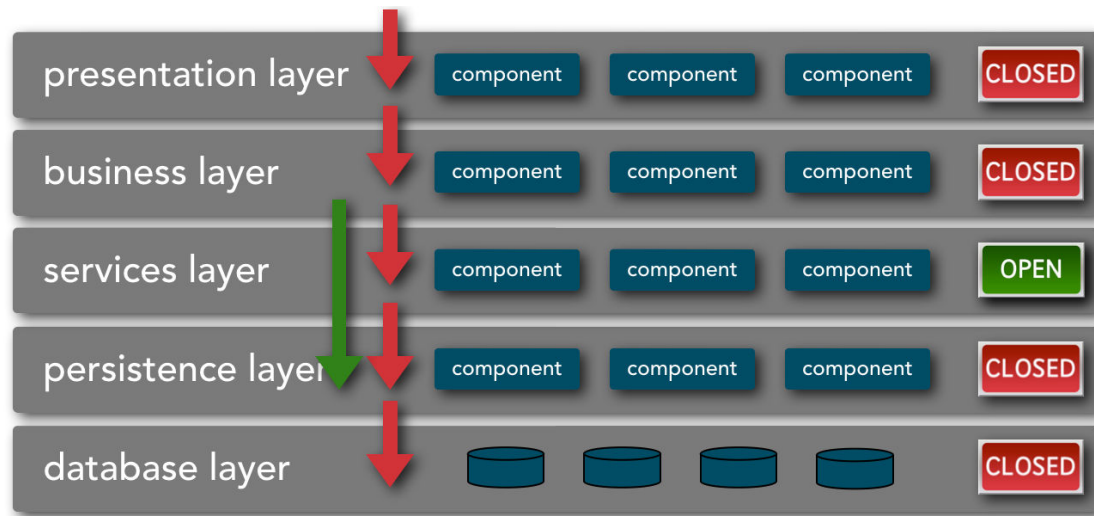
NEAL FORD



Next class: 12 March 2020 09:00-12:00 ET (06:00-09:00 PT)

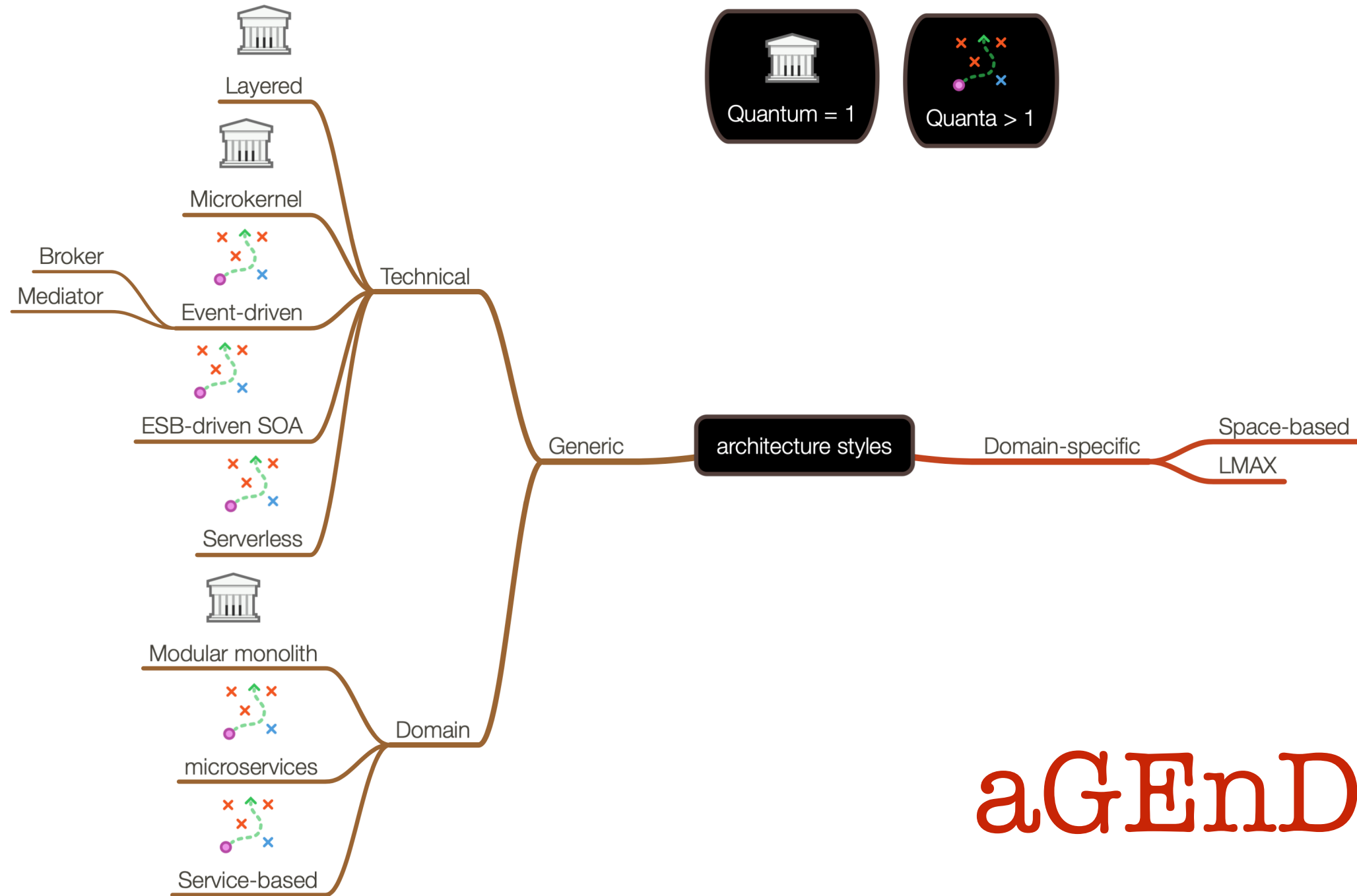
Layered Architecture

drivers



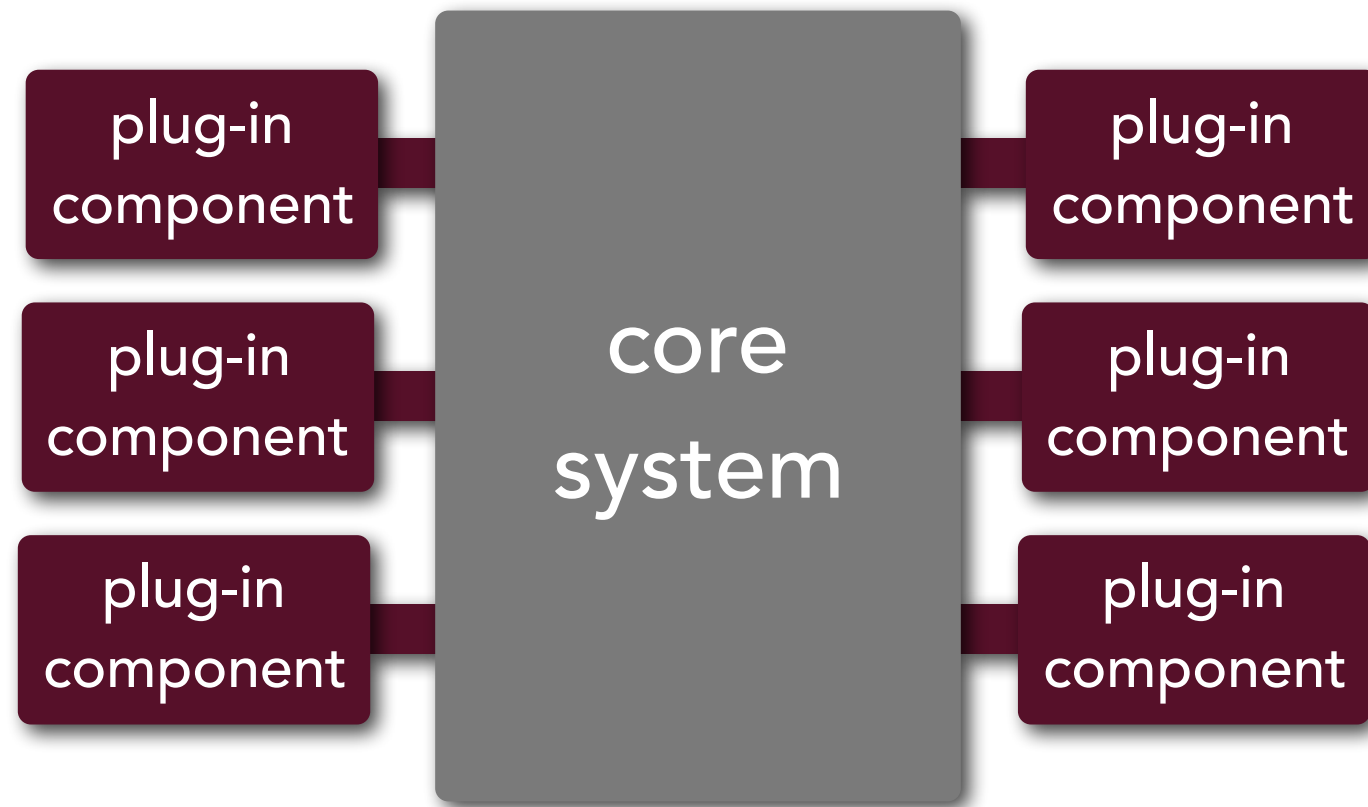
✓ simplicity

✓ cost



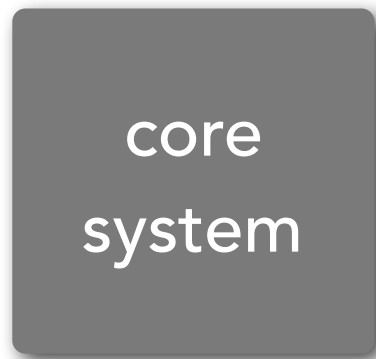
Microkernel Architecture

(a.k.a. plug-in architecture pattern)

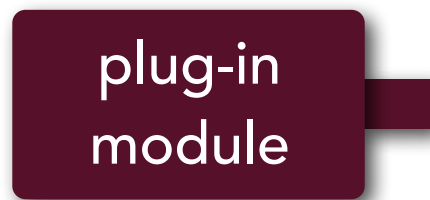


Microkernel Architecture

architectural components

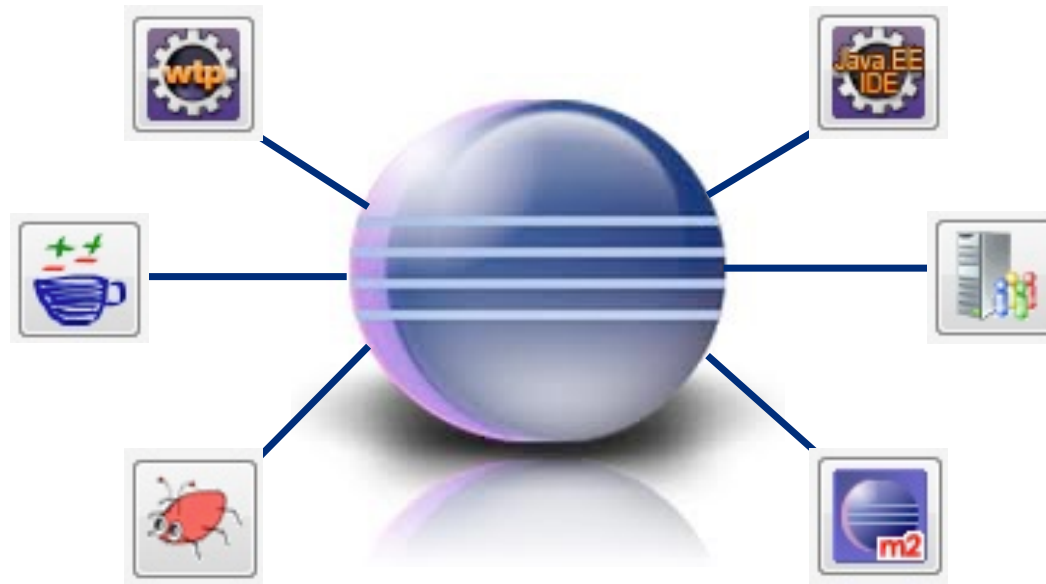


minimal functionality to run system
general business rules and logic
no custom processing



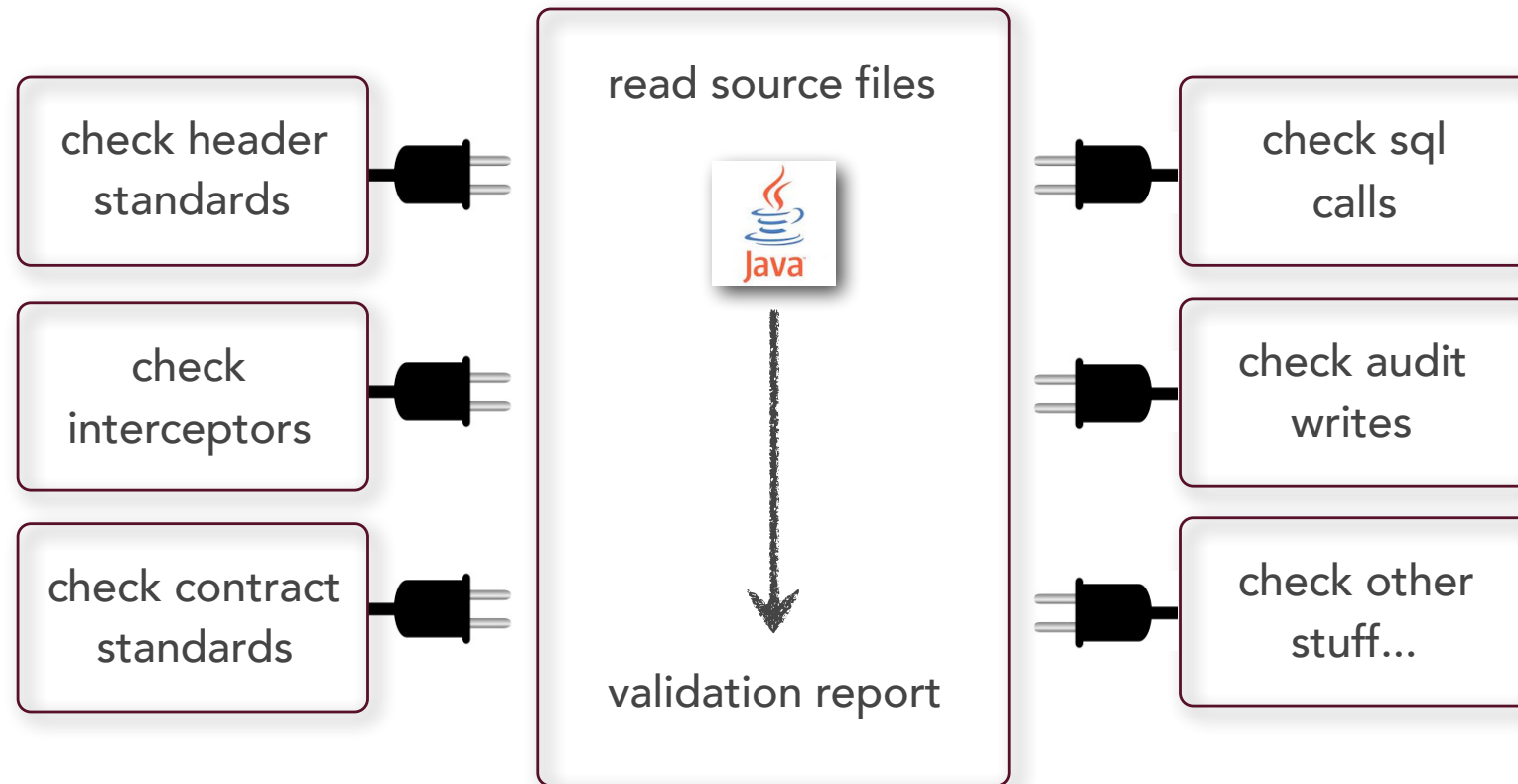
standalone independent module
specific additional rules or logic

Microkernel Architecture



Microkernel Architecture

source validation tool



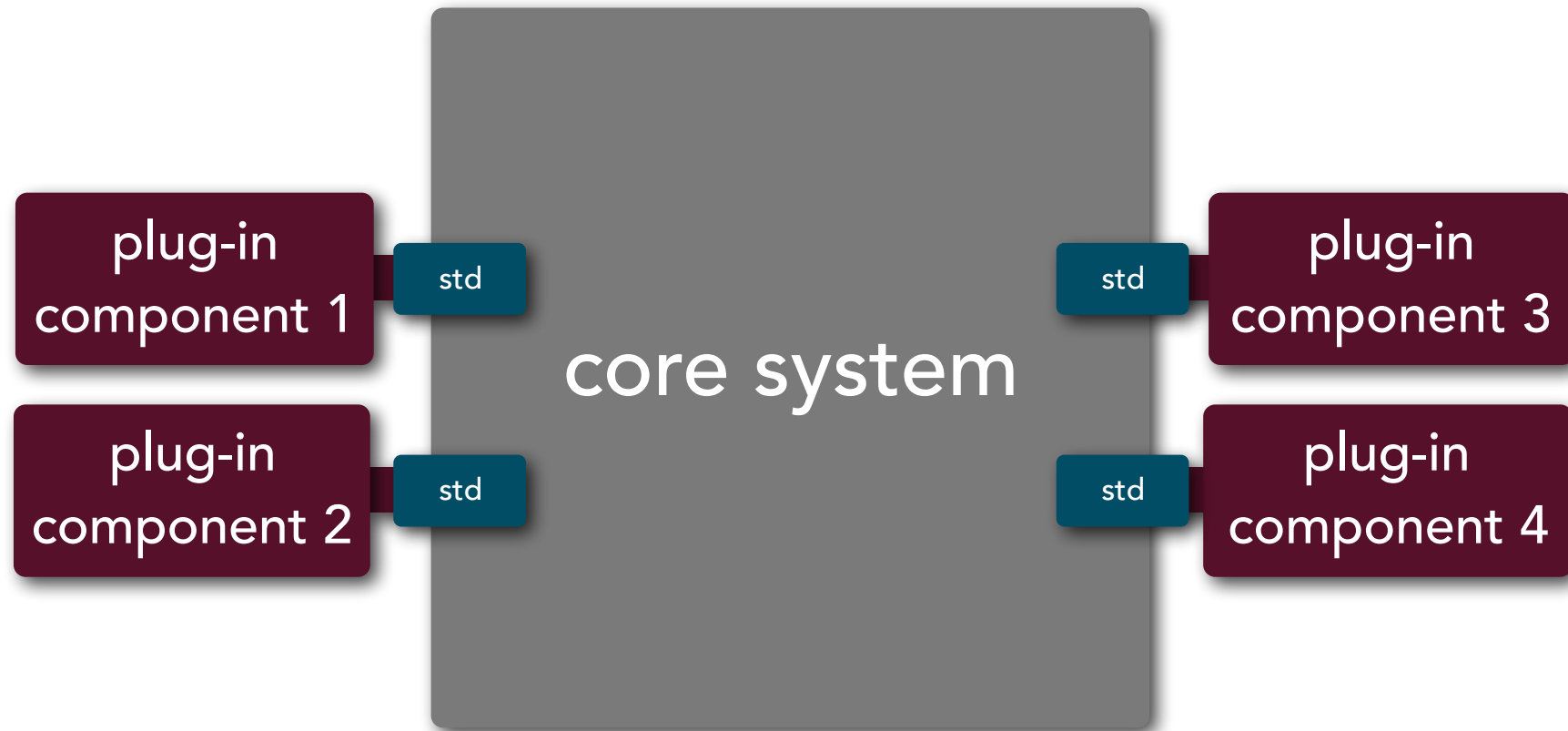
Microkernel Architecture

claims processing



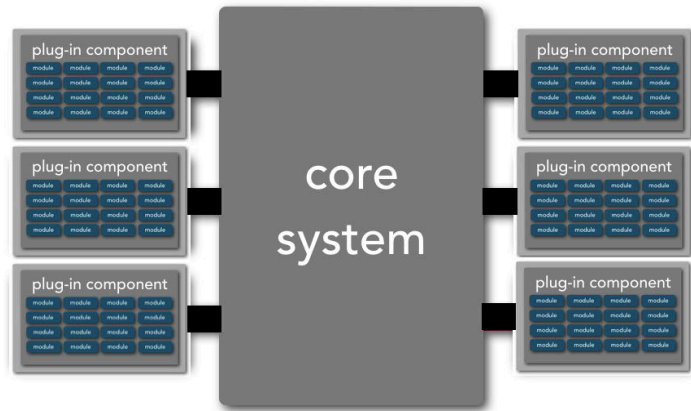
Microkernel Architecture

plug-in contracts



Microkernel Architecture

drivers



✓ modularity

✓ simplicity

✓ cost

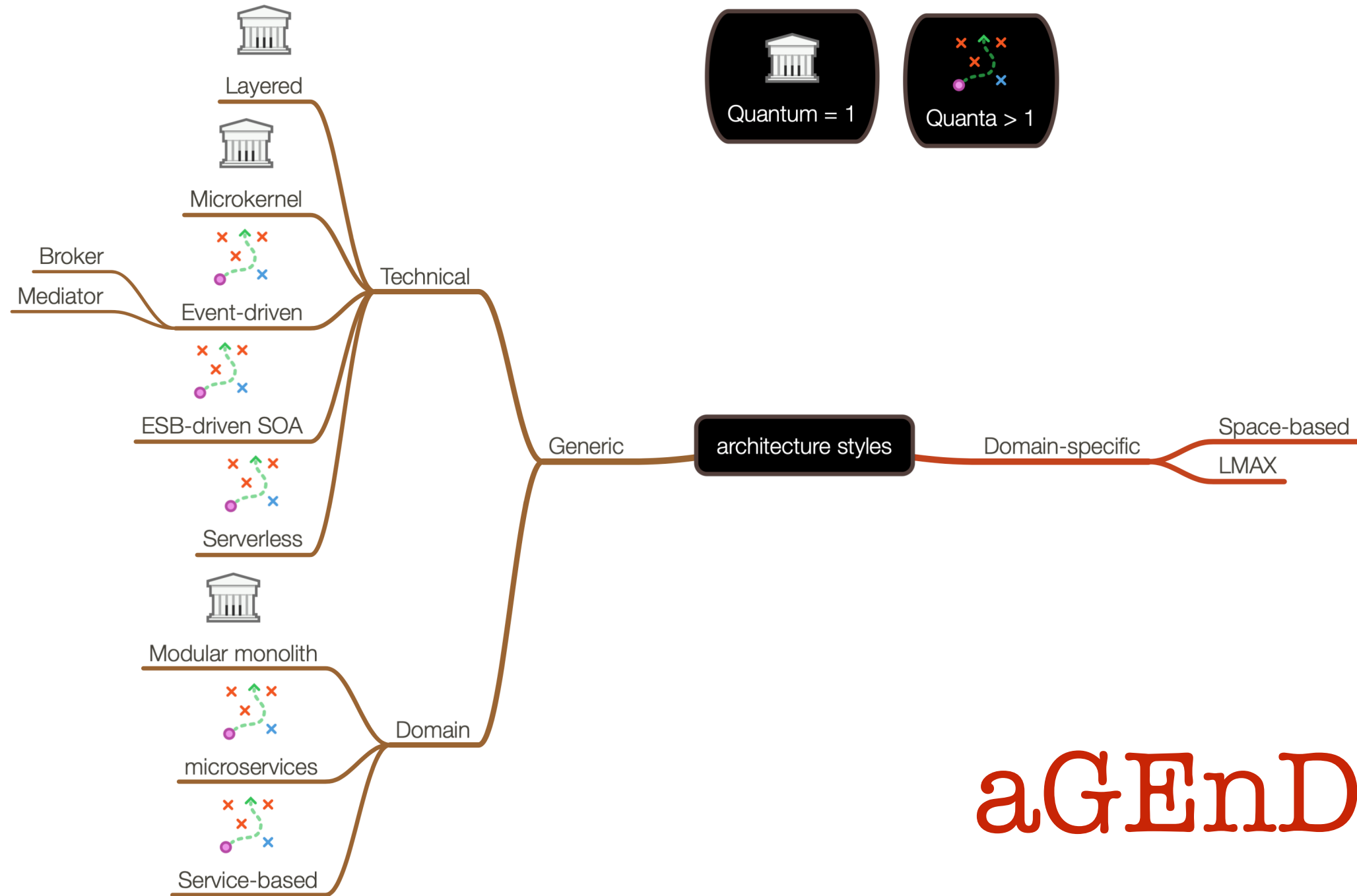
✓ adaptability

✓ evolvability

✓ agility

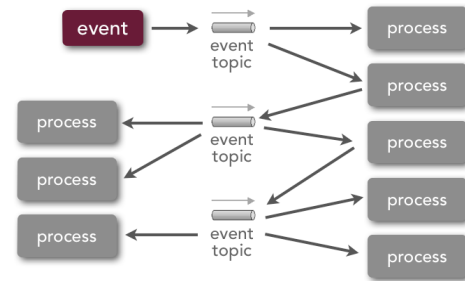
✓ testability

✓ deployability

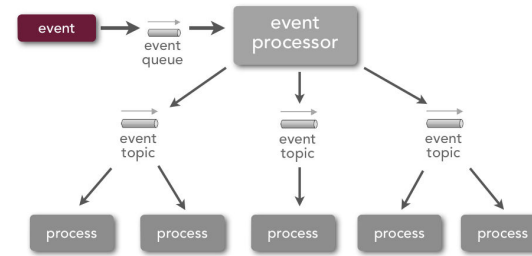


aGEnDA

Event-driven Architecture

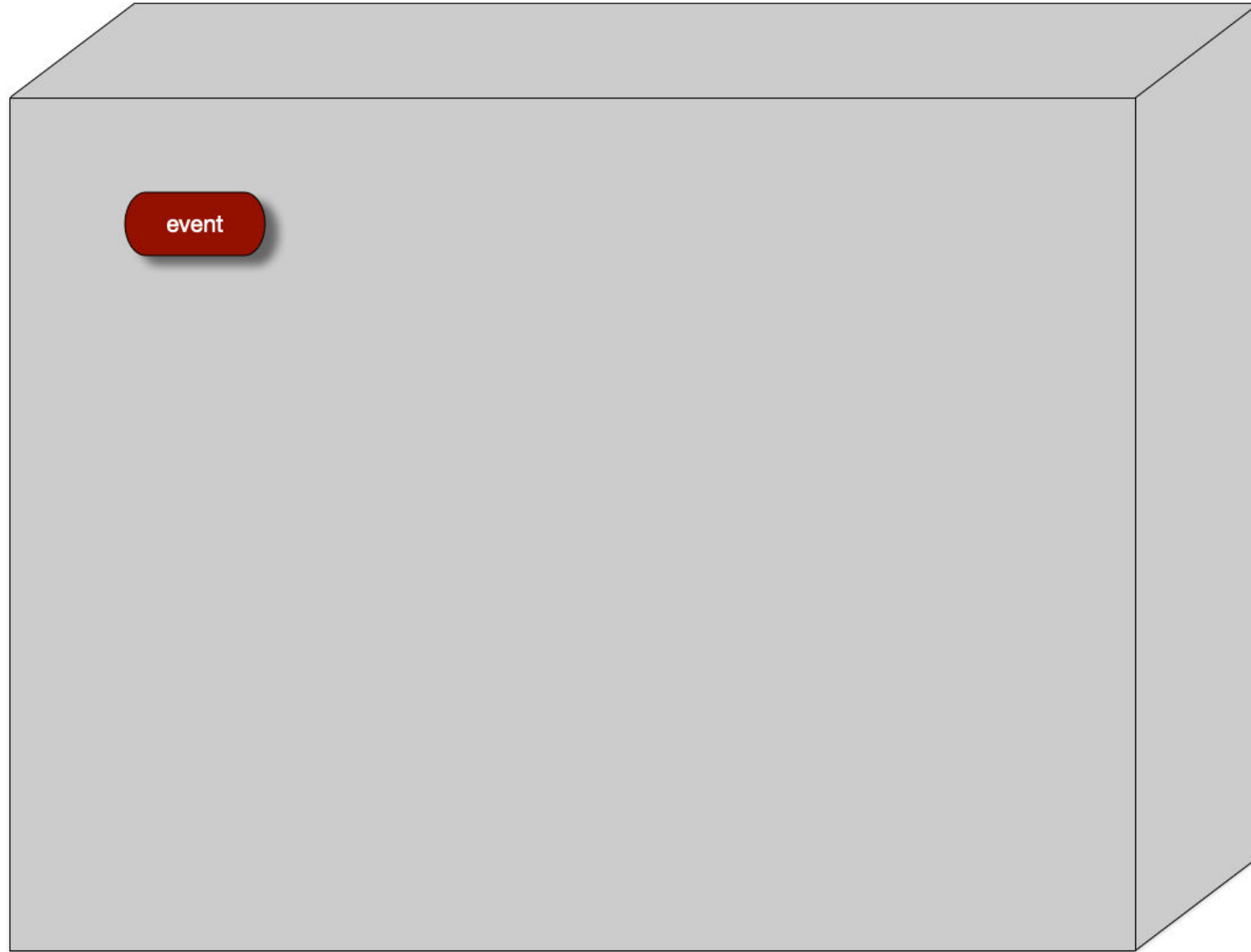


broker topology

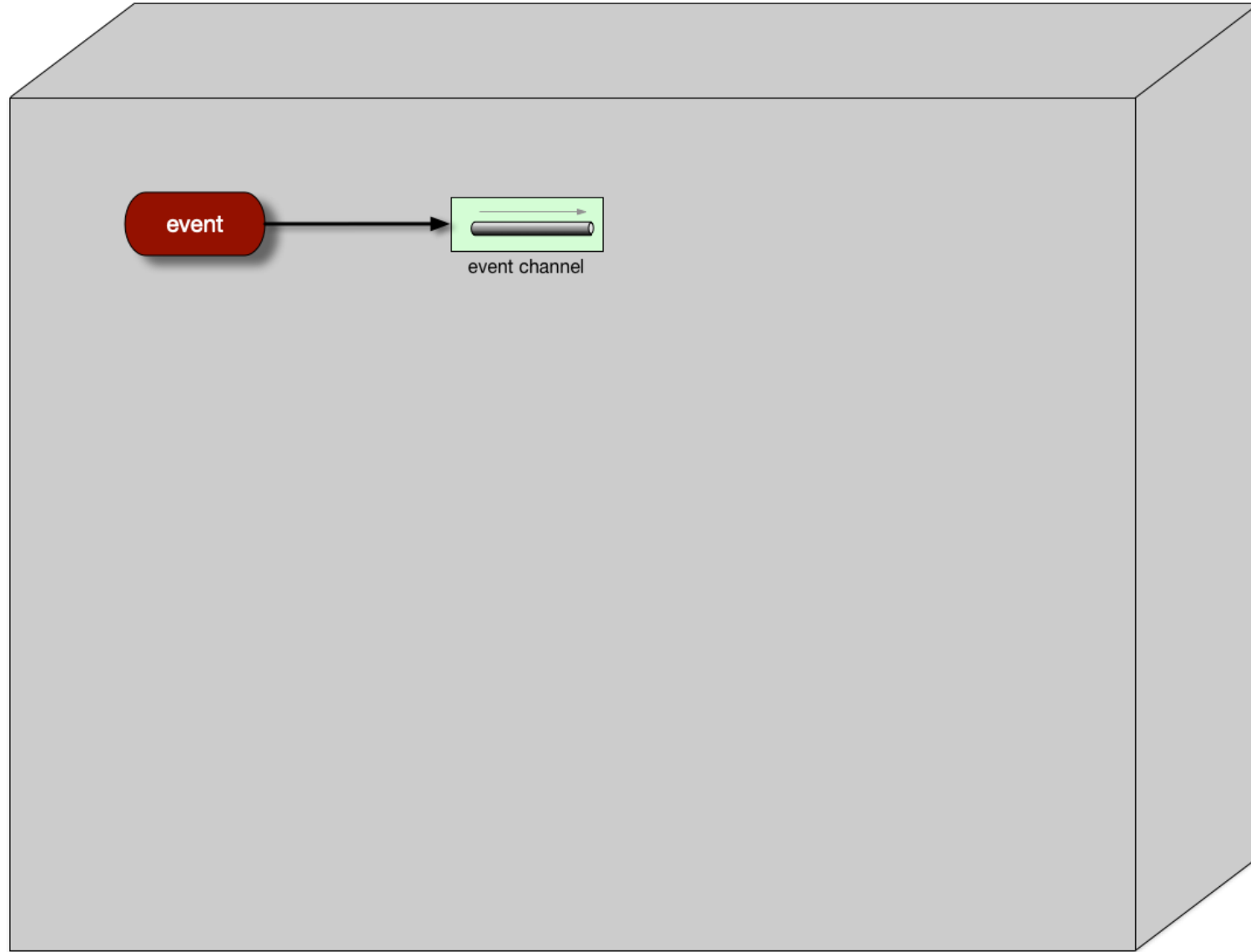


mediator topology

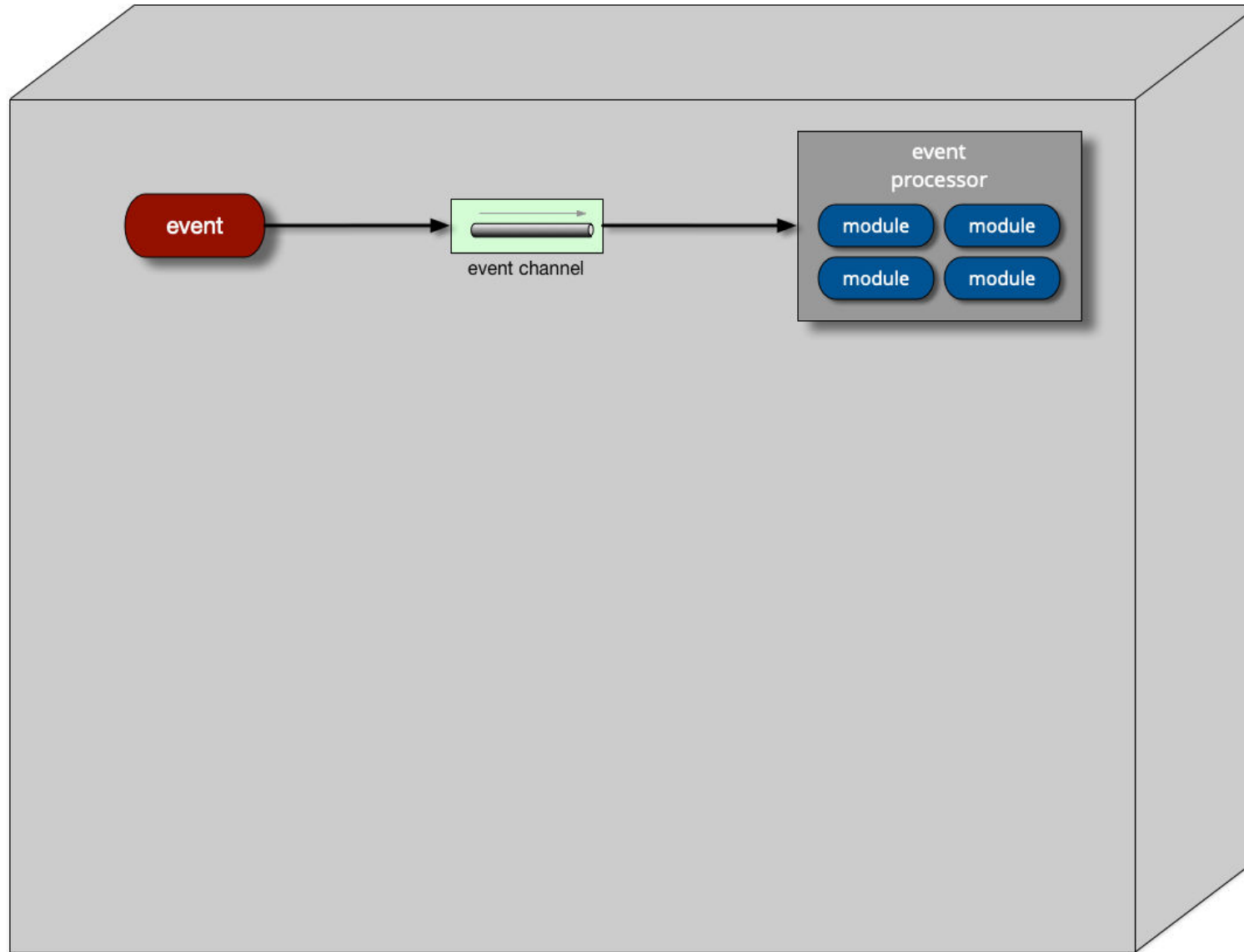
broker base architecture



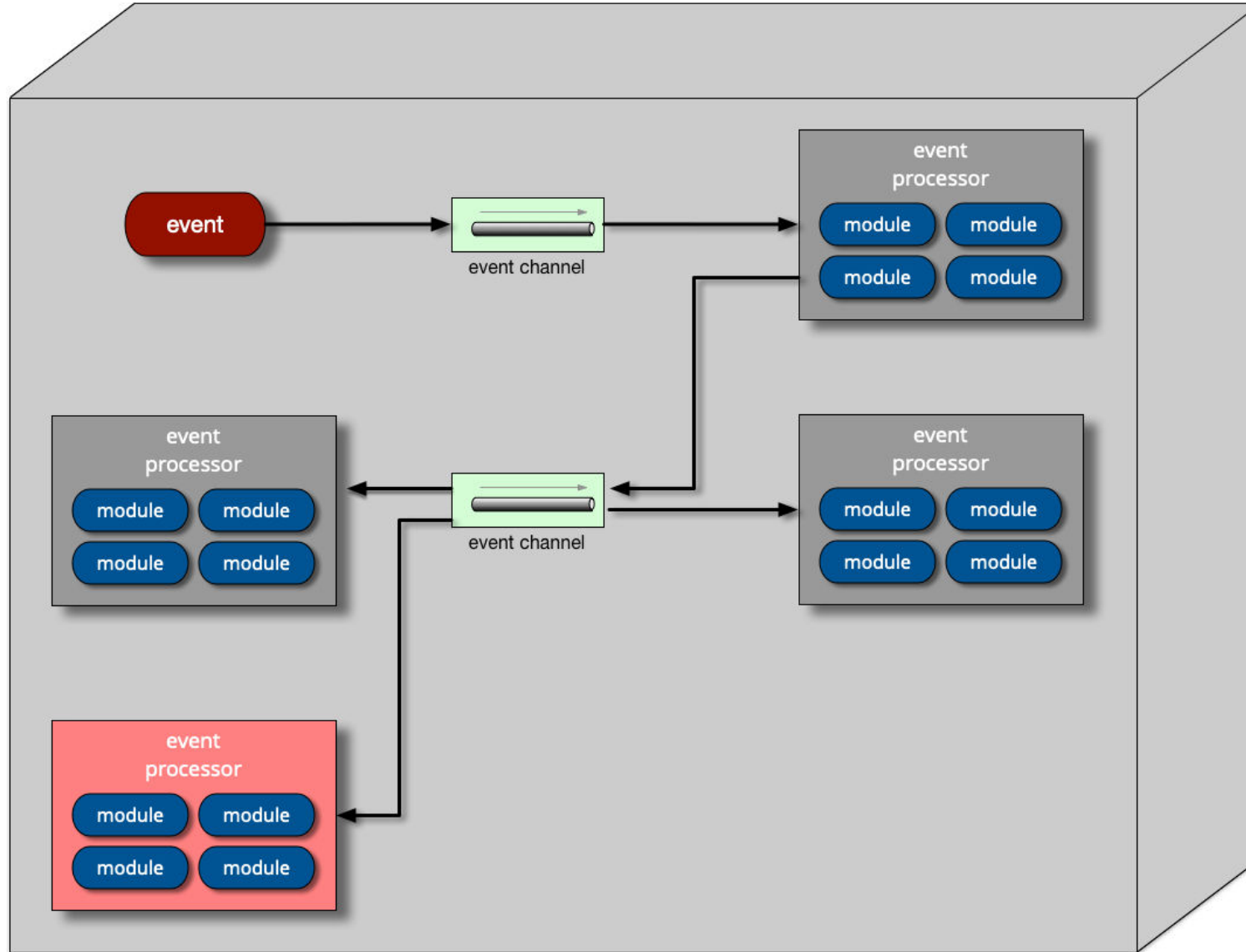
broker base architecture



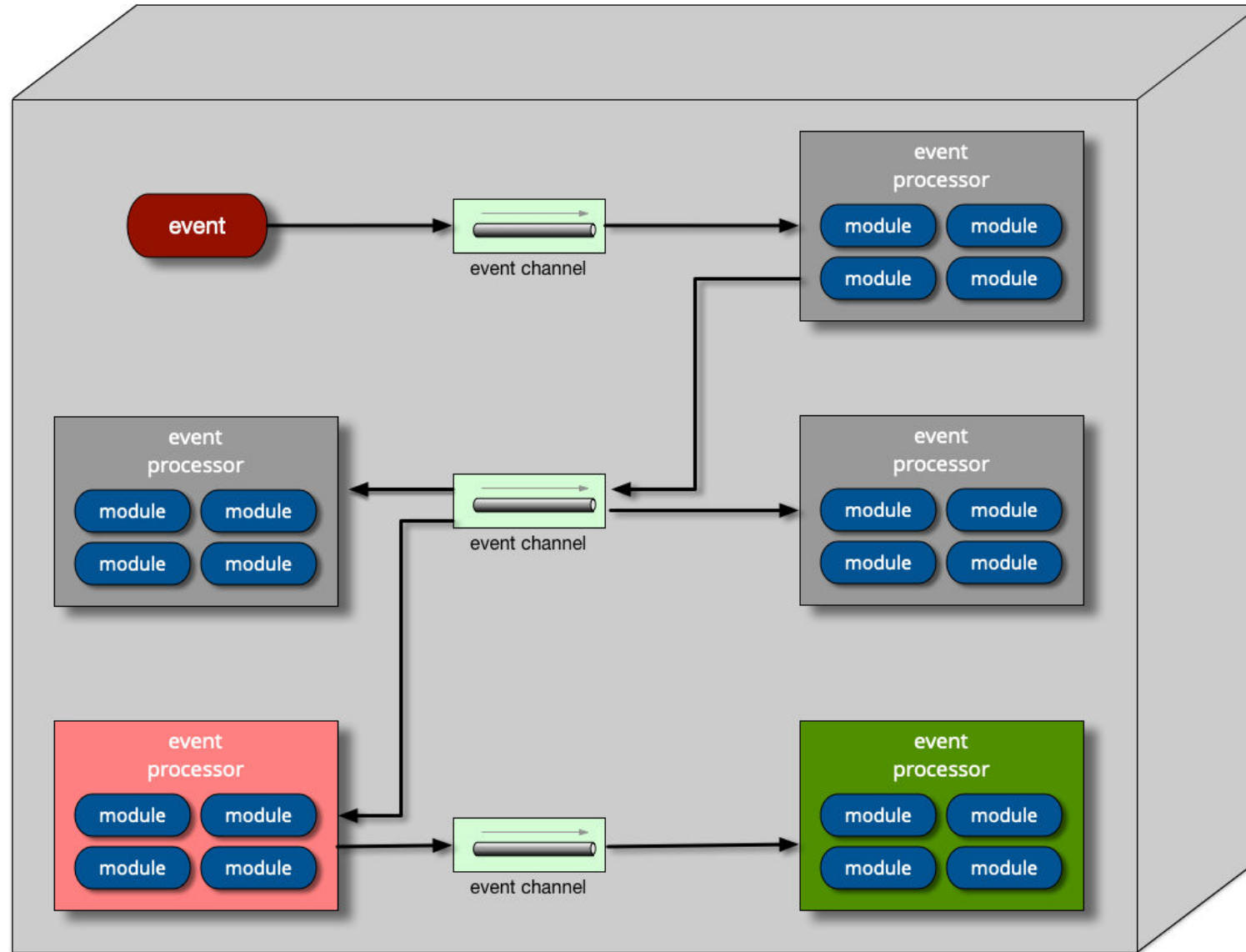
broker base architecture



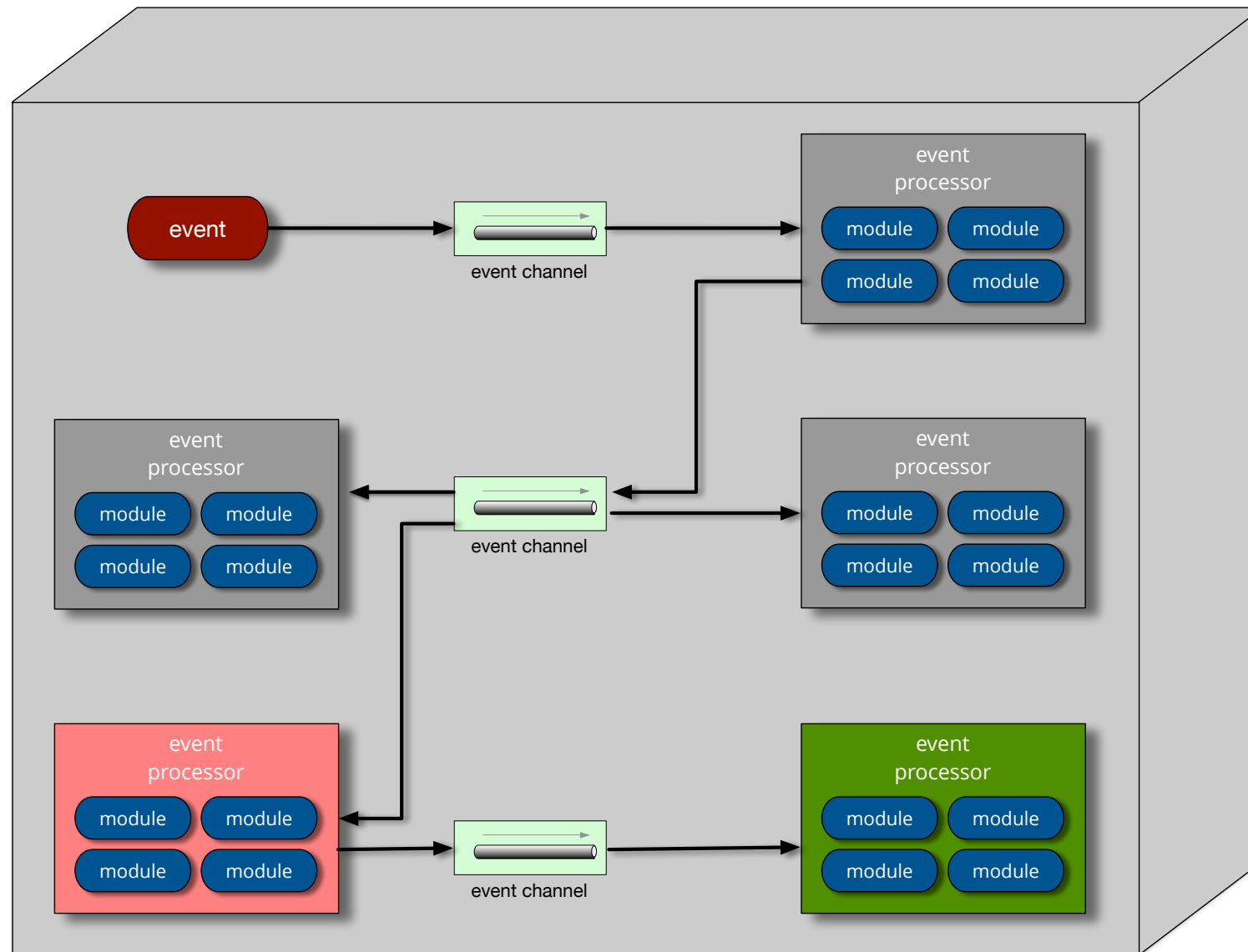
broker base architecture



broker base architecture

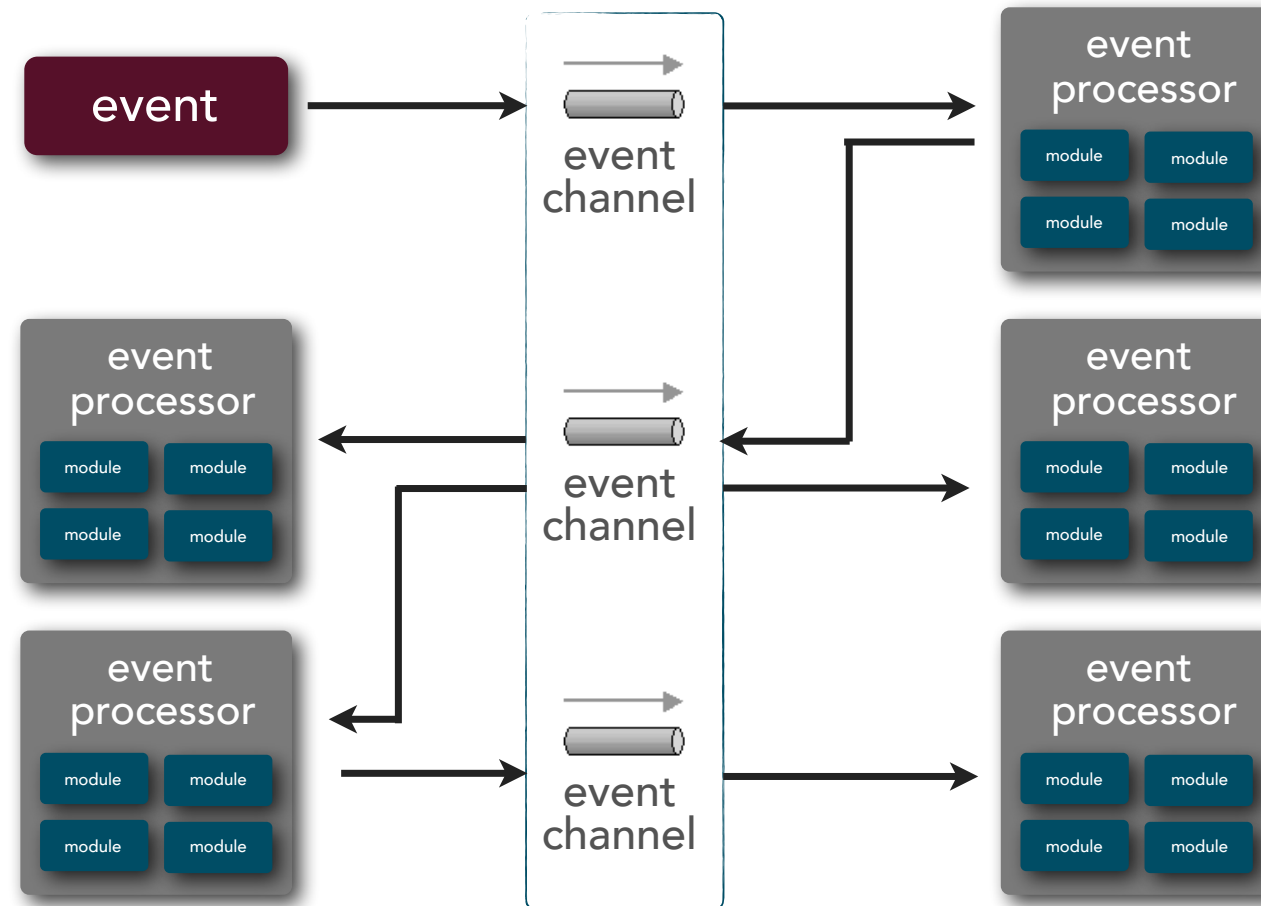


broker base architecture

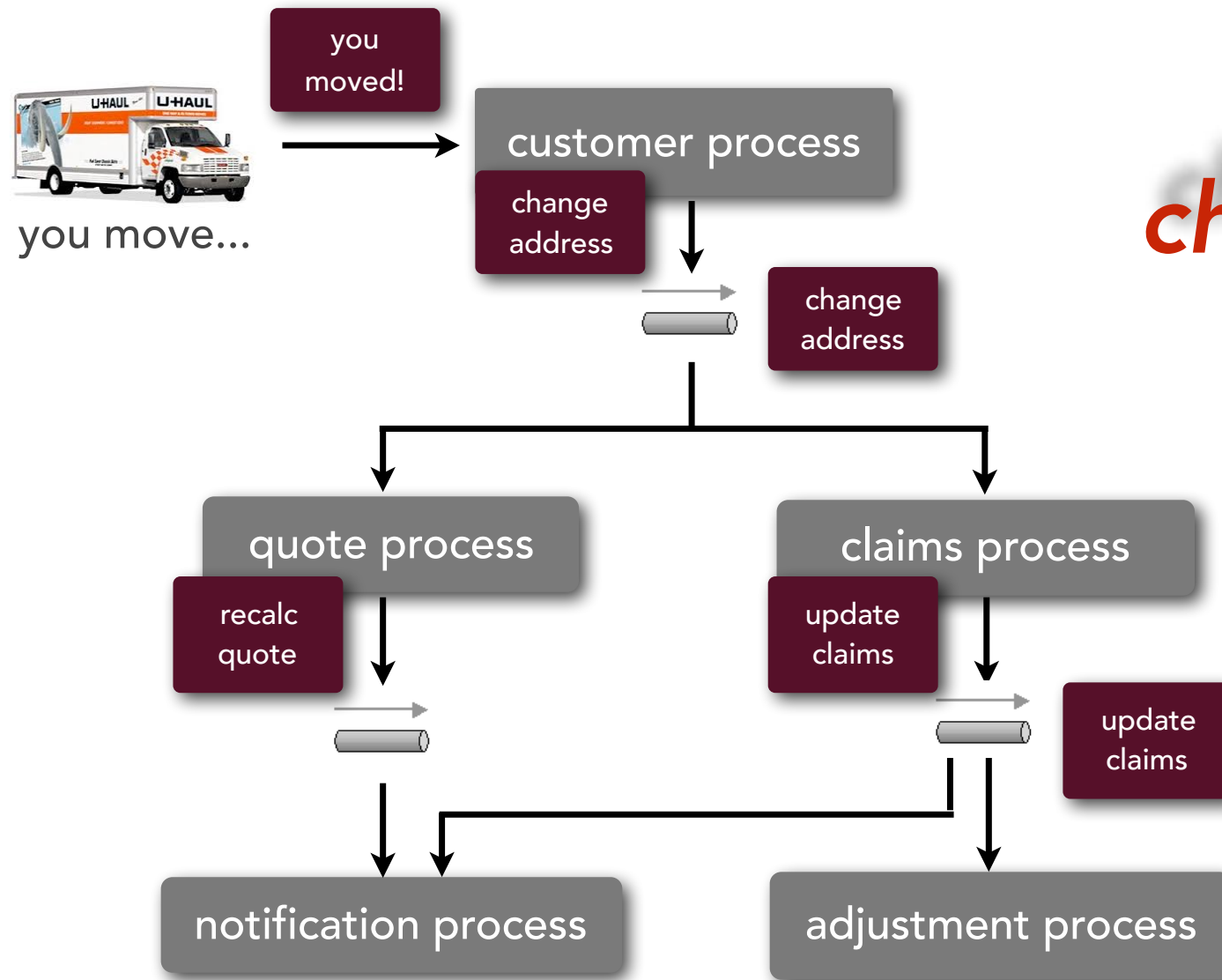


Event-driven Architecture

broker topology



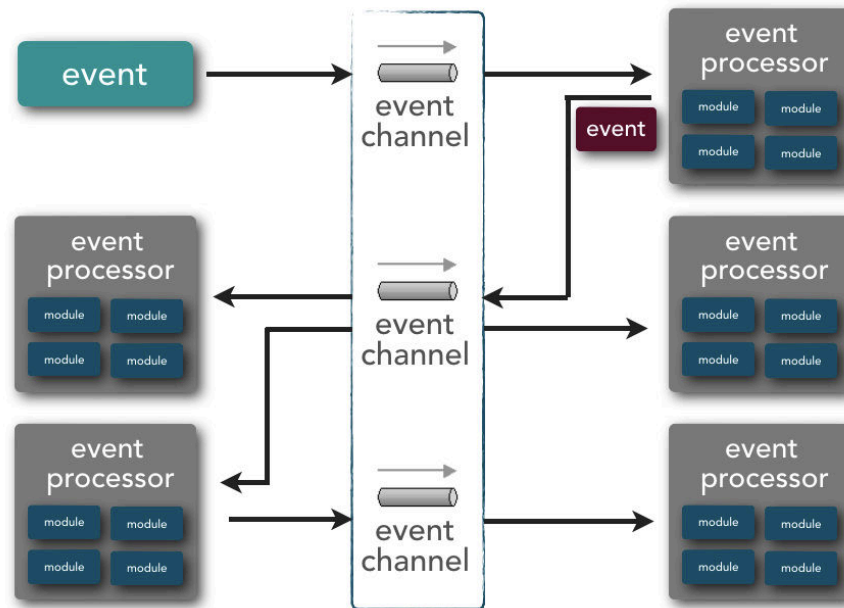
Event-driven Architecture



choreography

Event-driven Architecture

drivers

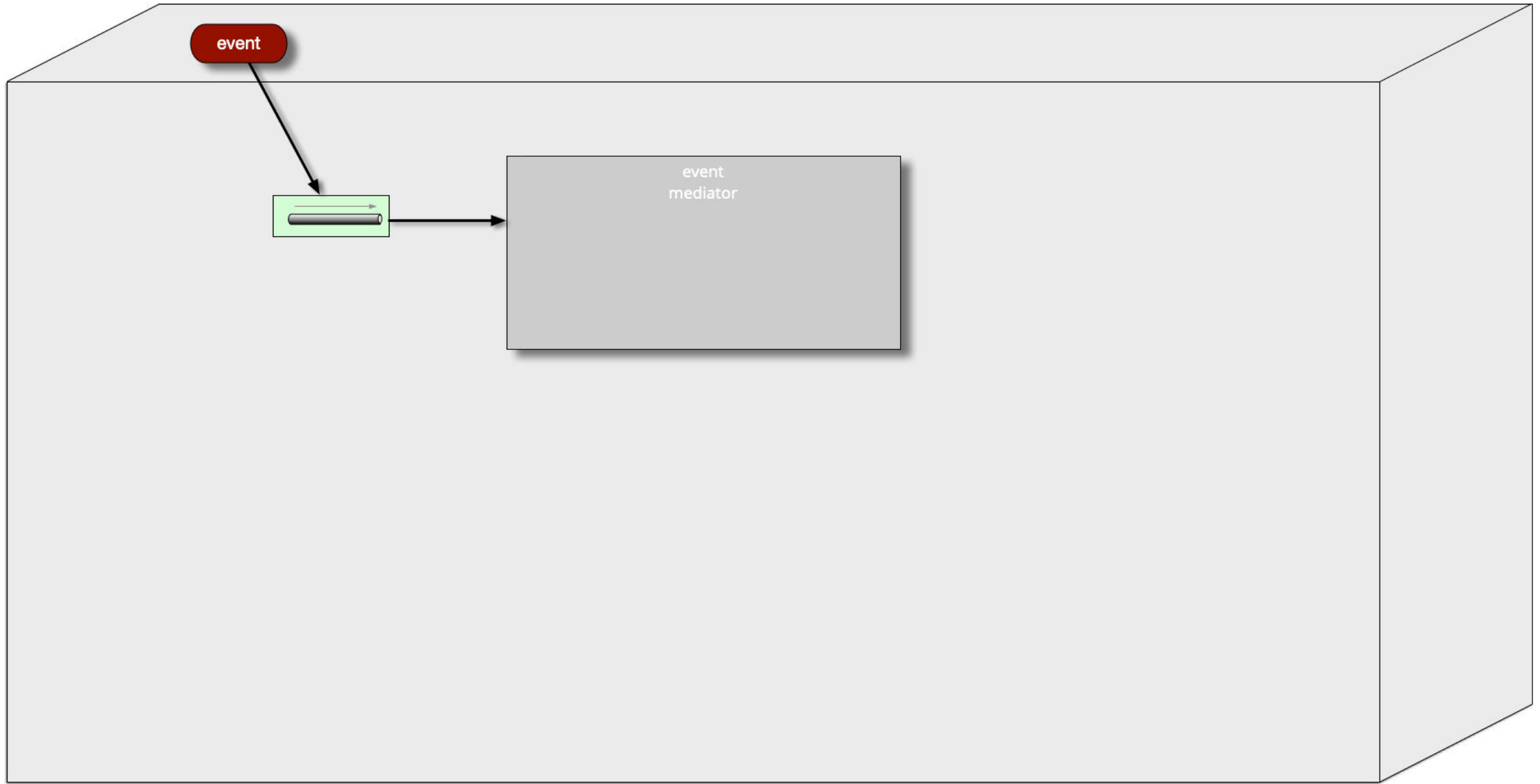


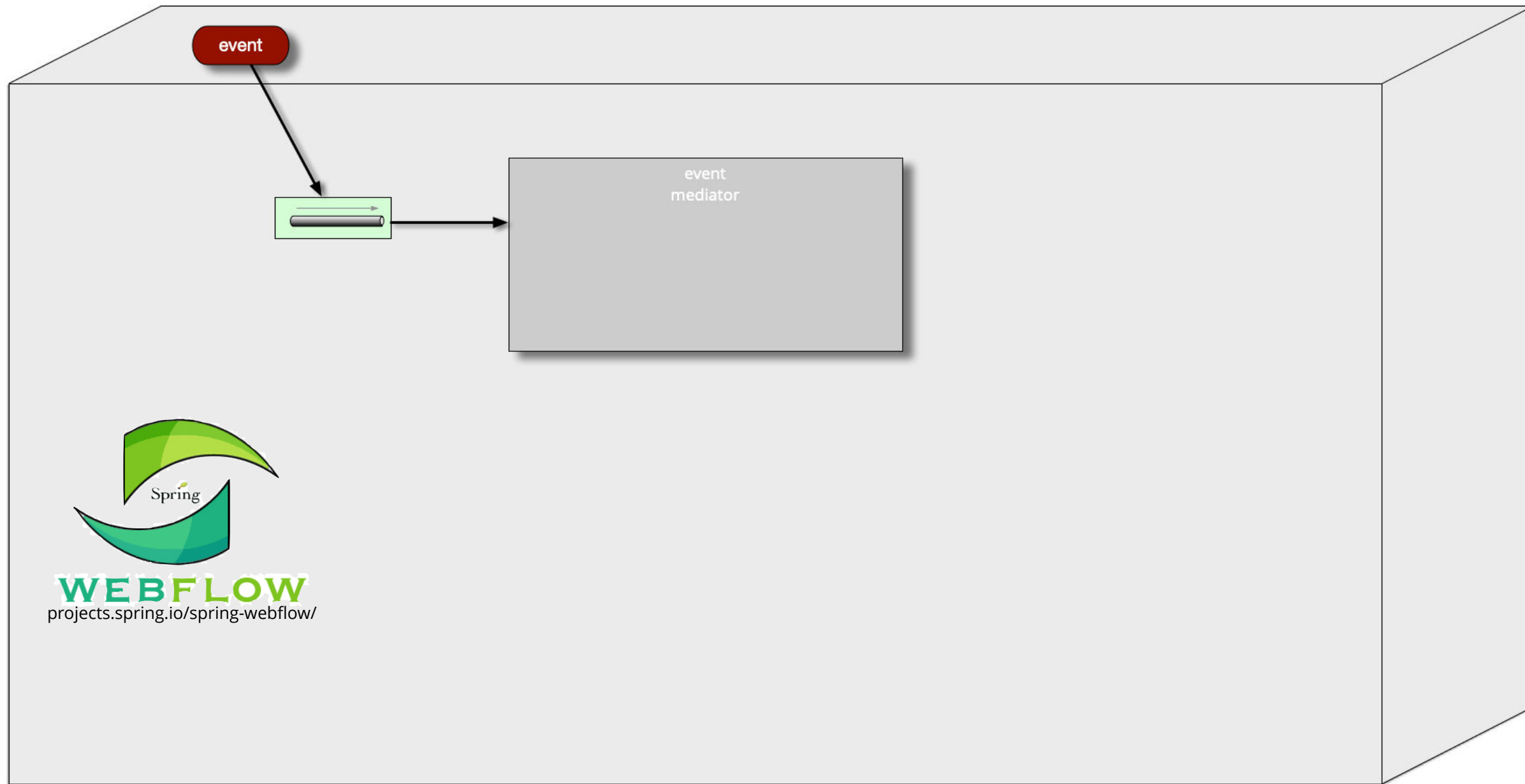
- ✓ modularity
- ✓ agility
- ✓ fault-tolerance
- ✓ scalability
- ✓ performance
- ✓ elasticity
- ✓ adaptability
- ✓ evolvability

mediator base architecture

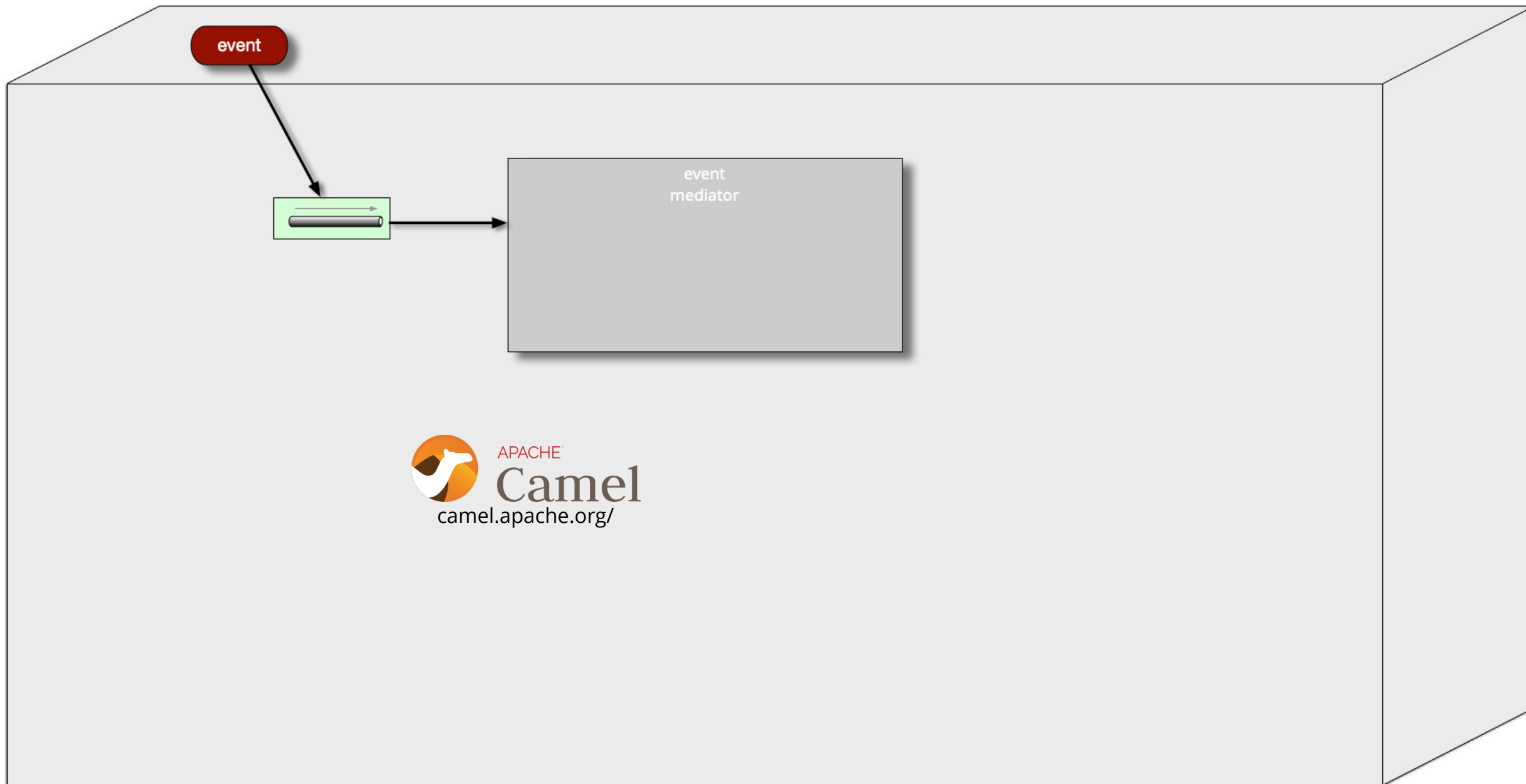


mediator base architecture

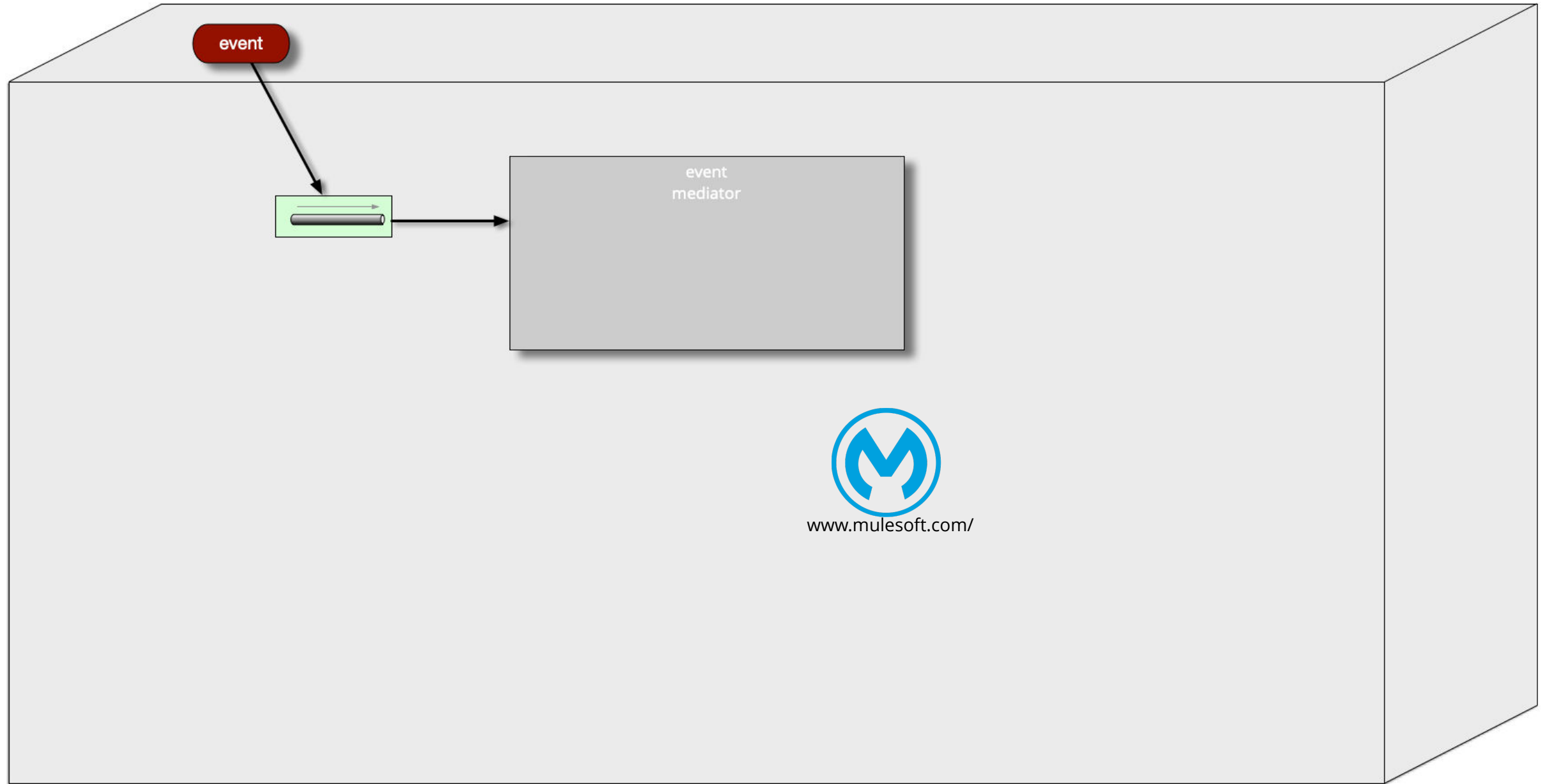




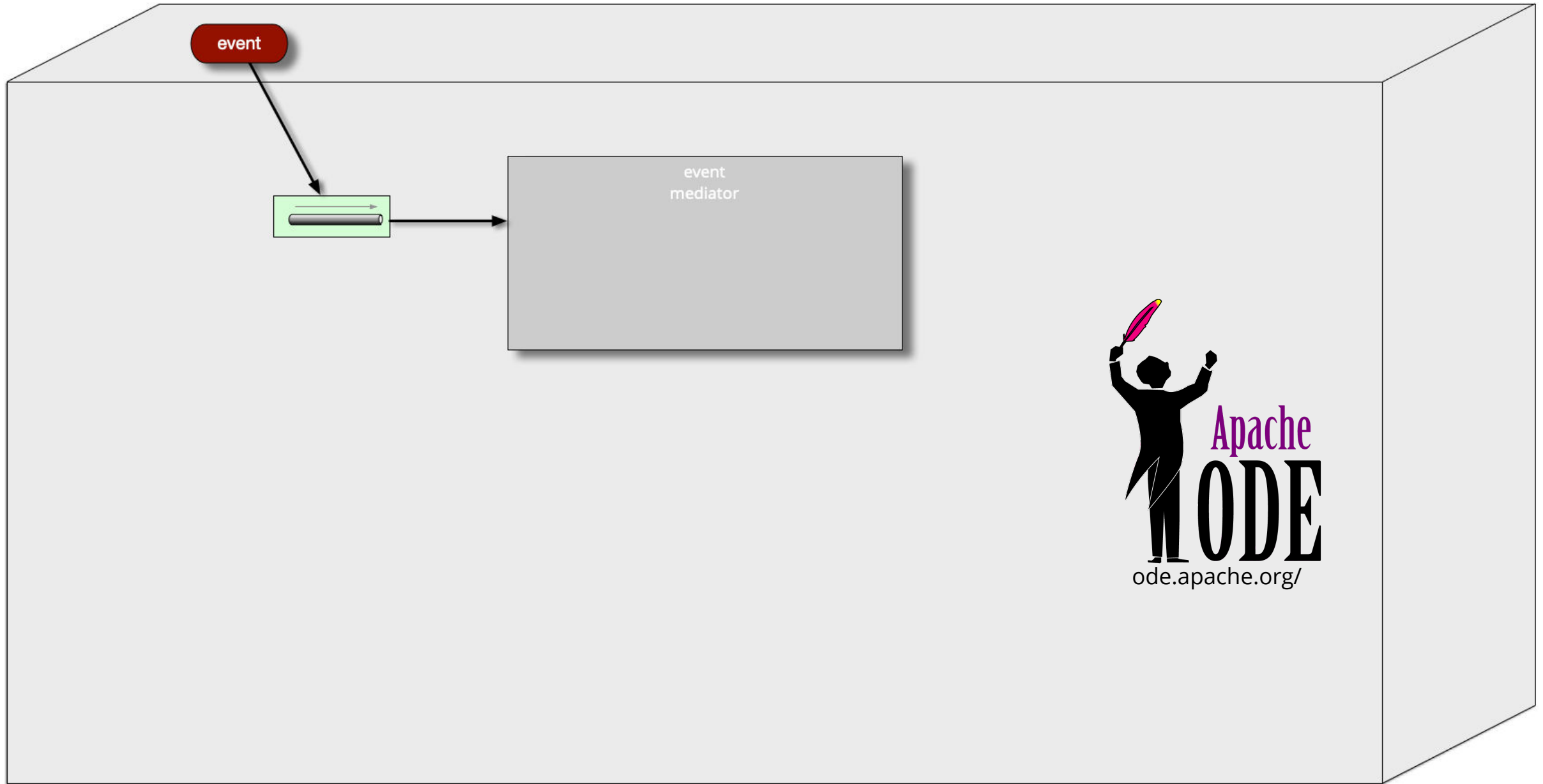
mediator base architecture



mediator base architecture



mediator base architecture





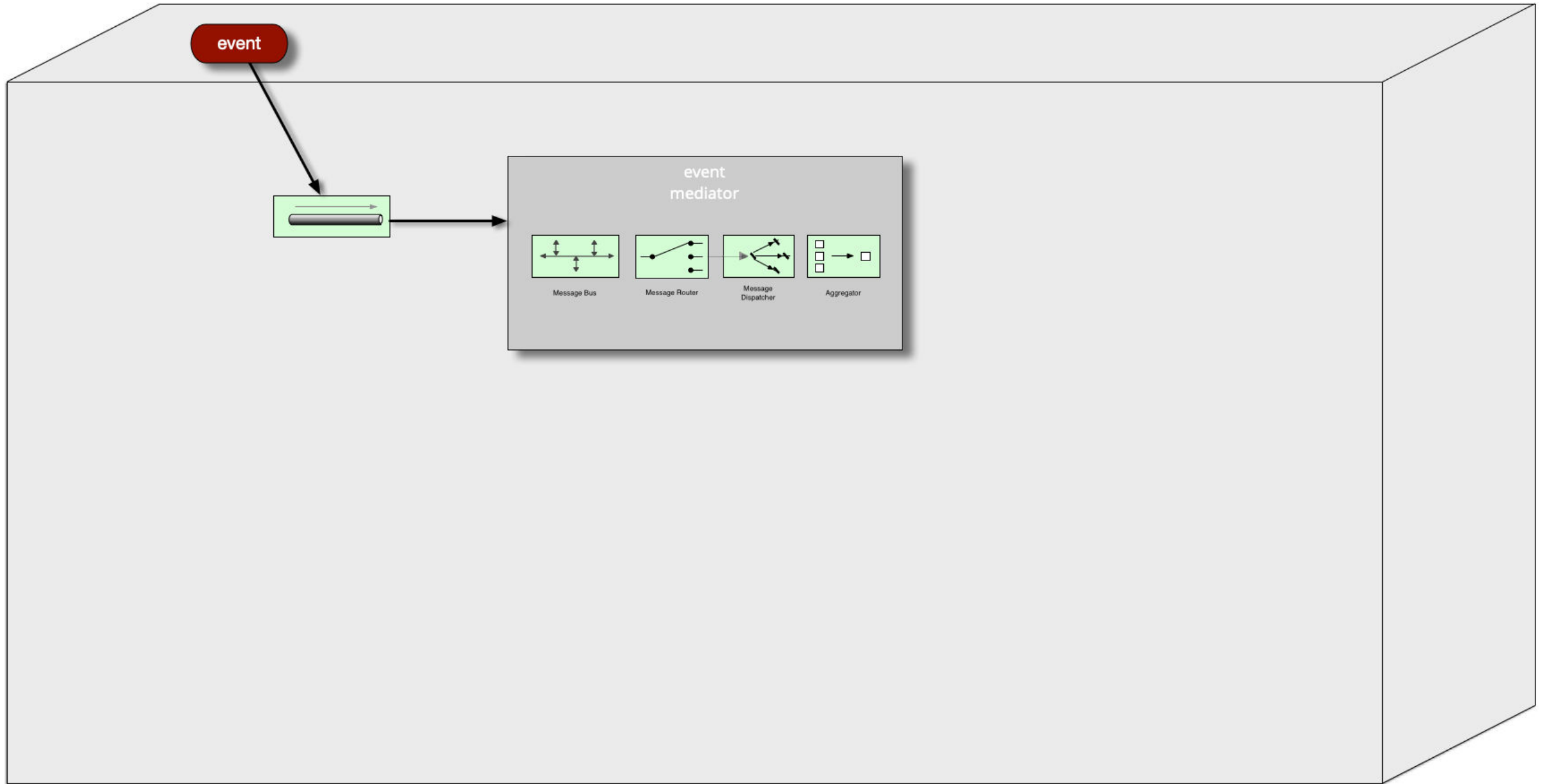




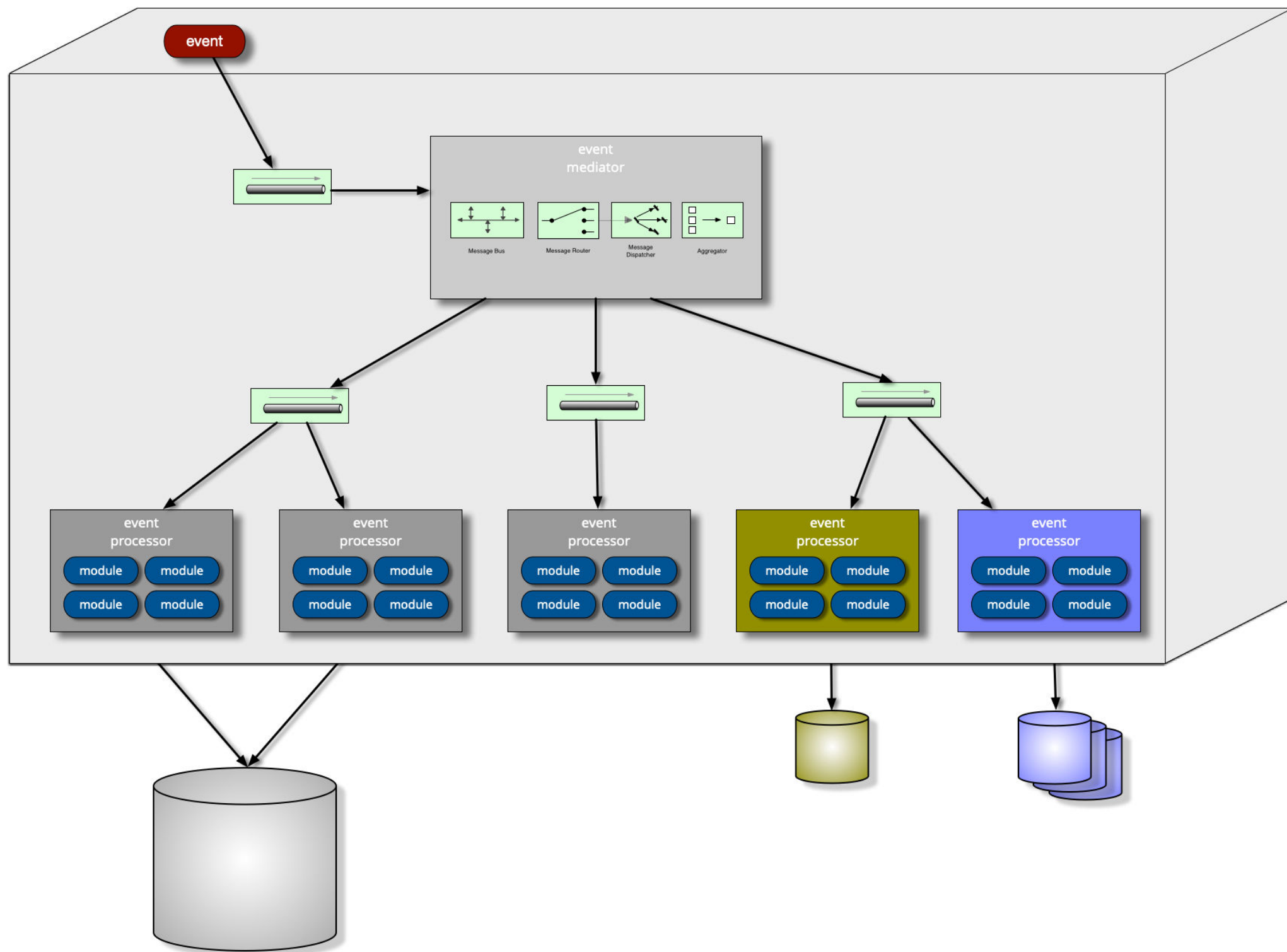




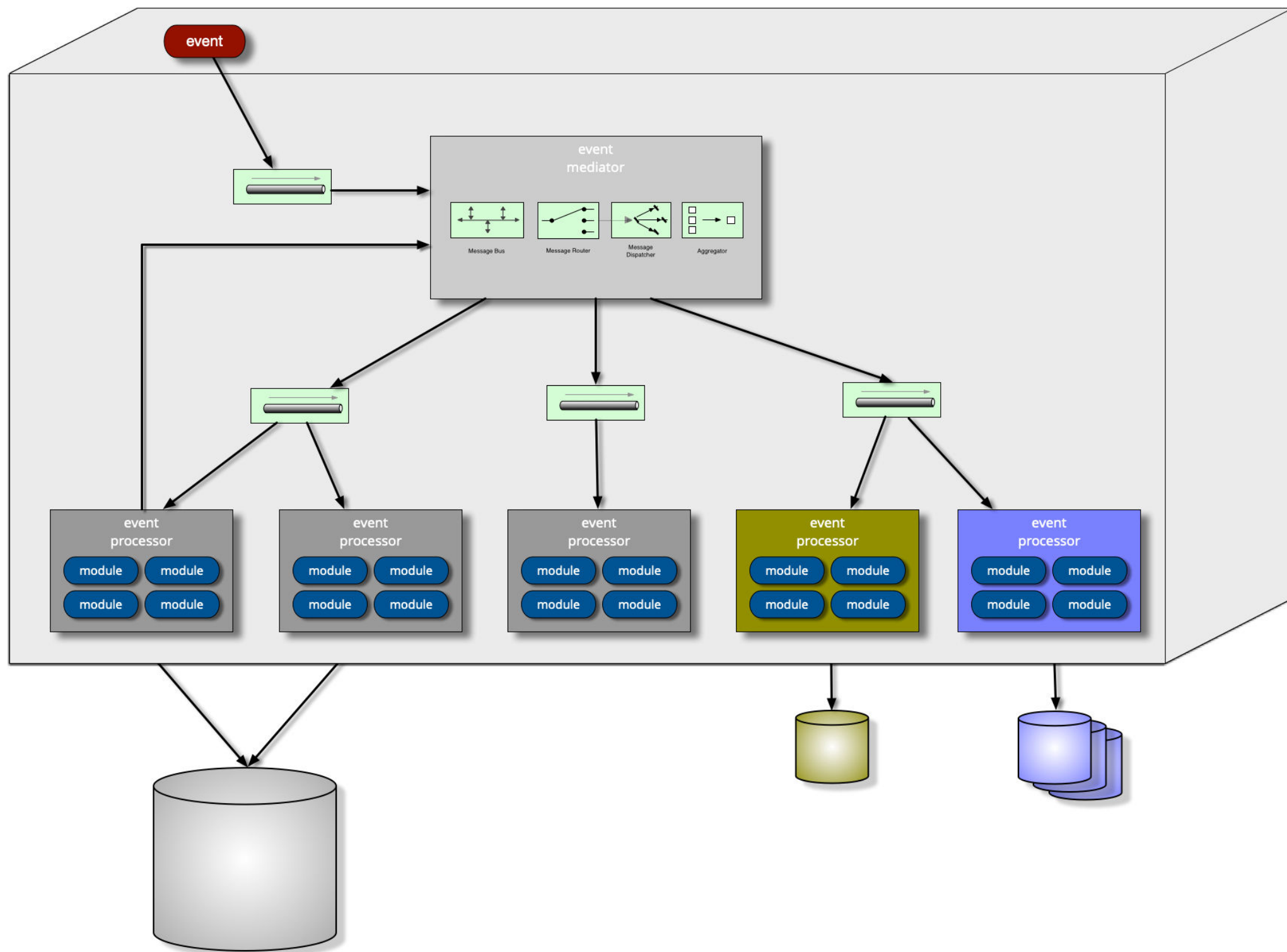
mediator base architecture



mediator base architecture



mediator base architecture

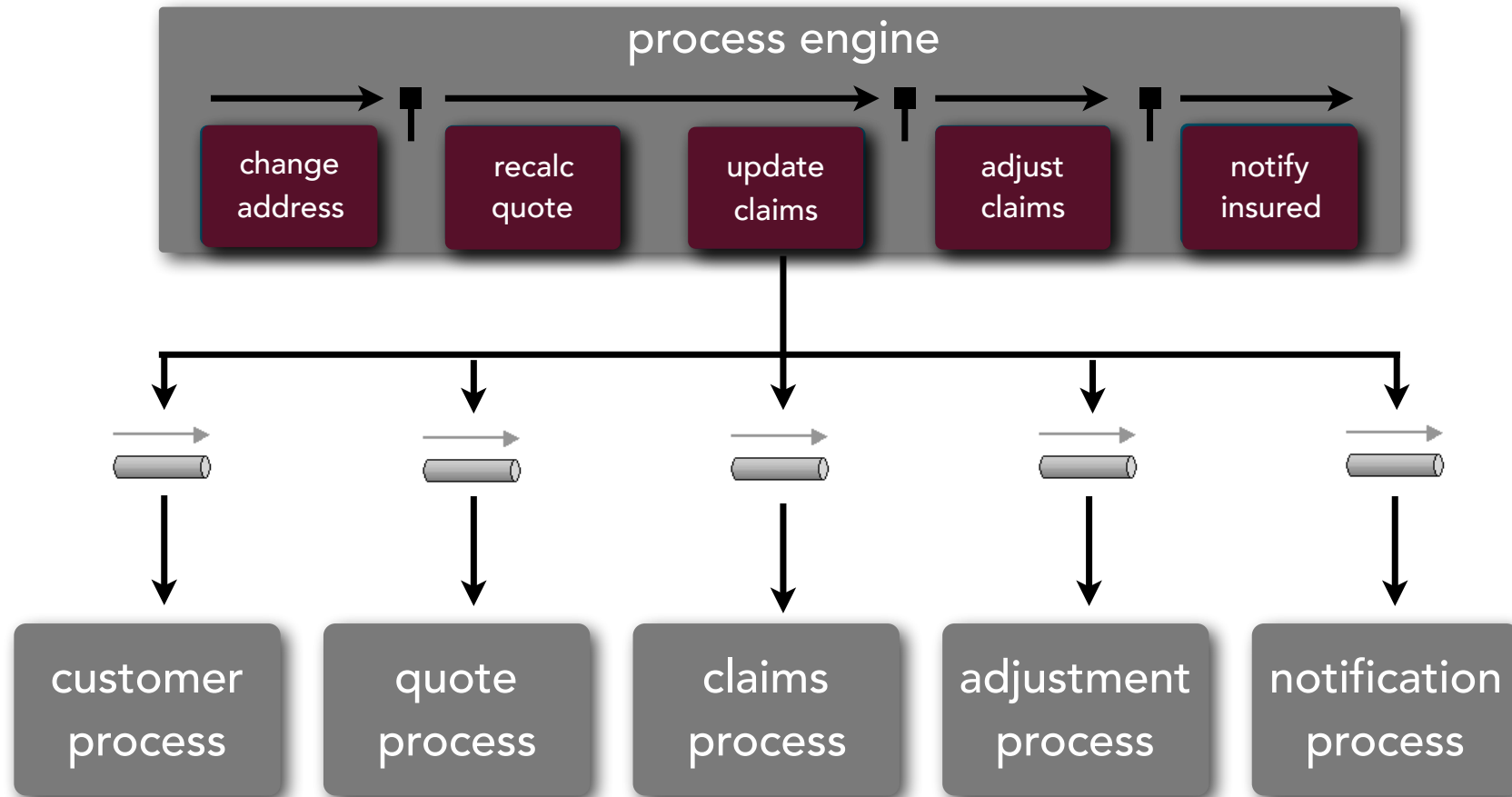


mediator message flow

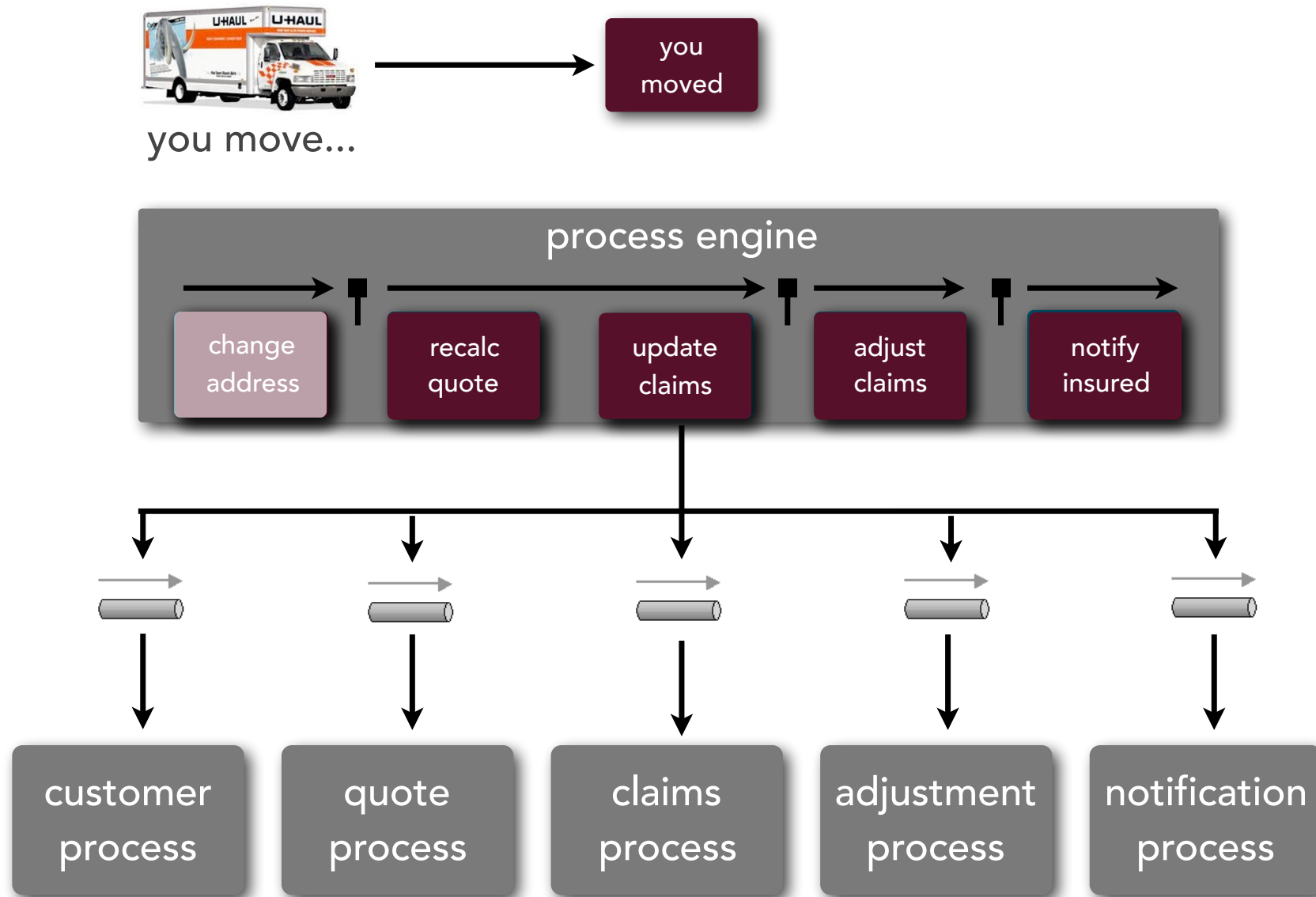


you move...

you
moved



mediator message flow

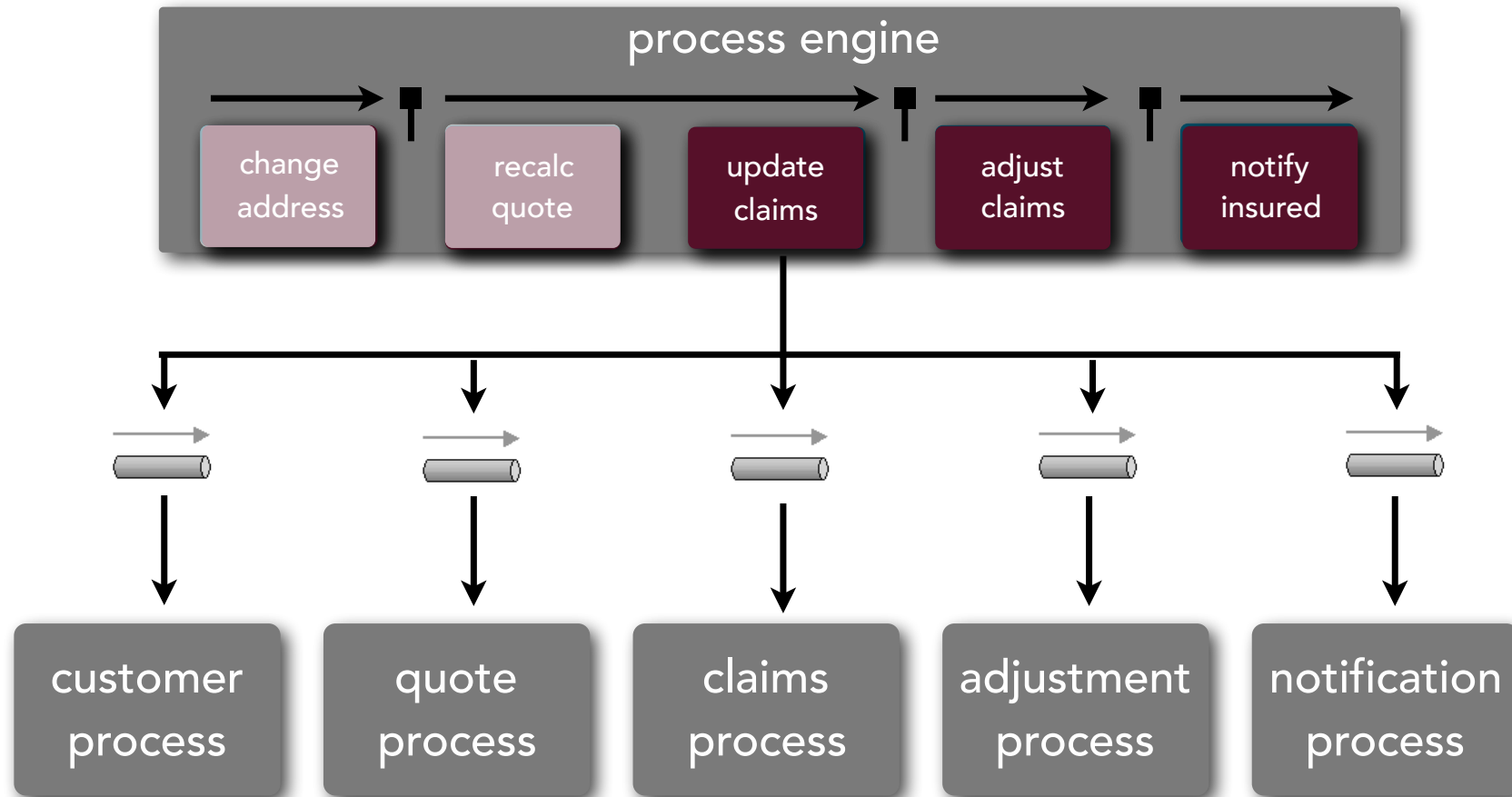


mediator message flow



you move...

you
moved

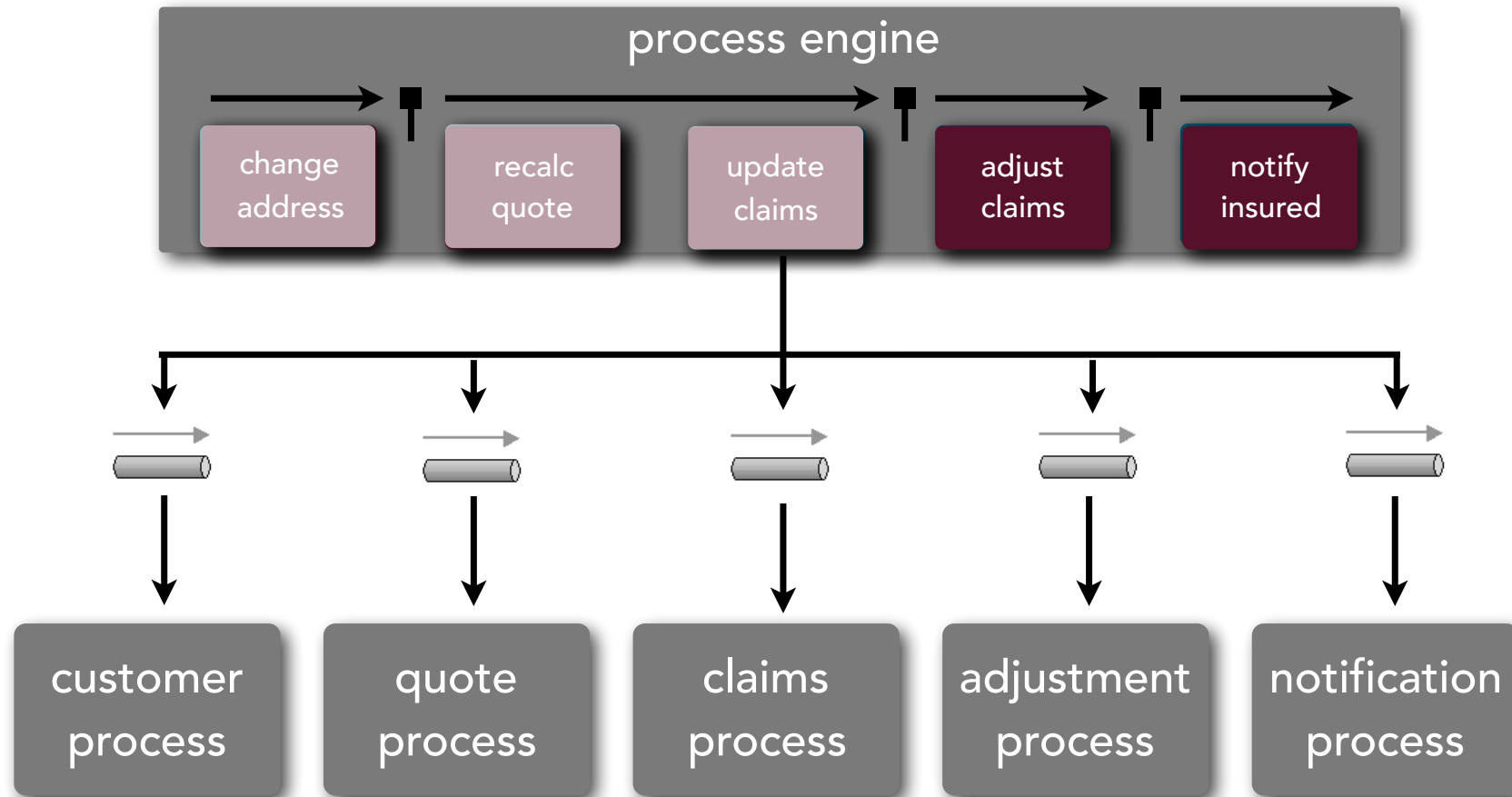


mediator message flow



you move...

you
moved

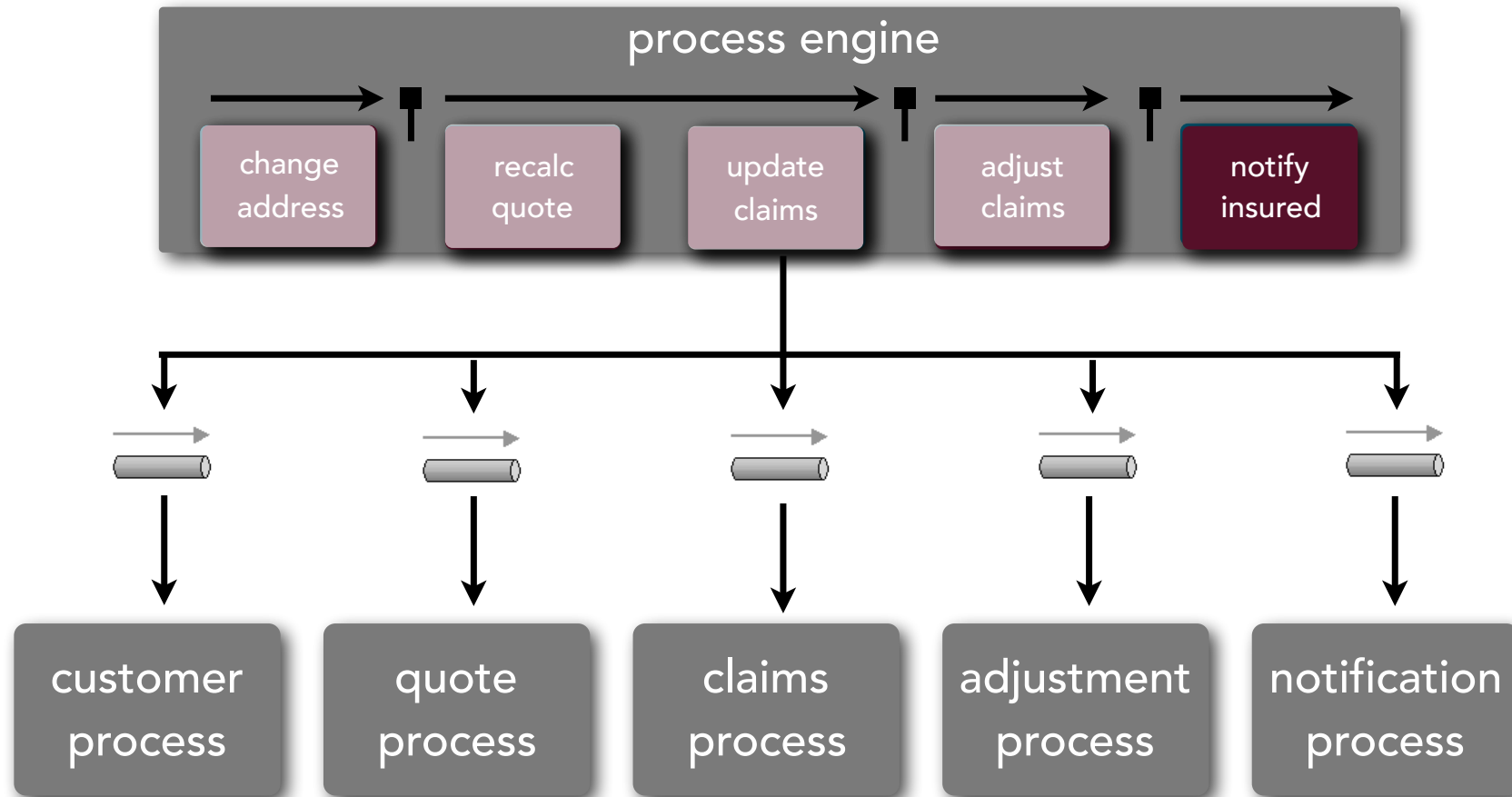


mediator message flow



you move...

you
moved

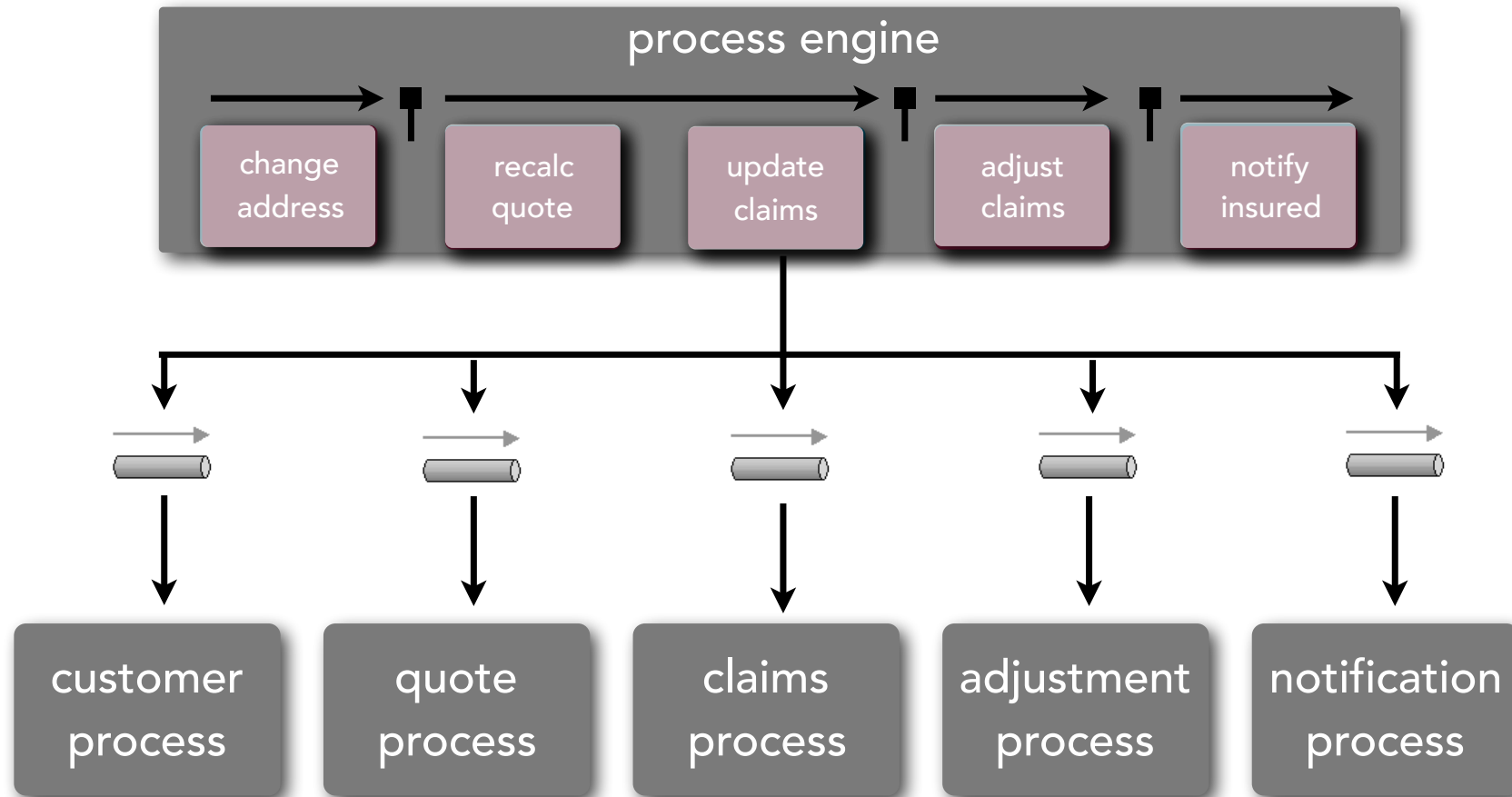


mediator message flow



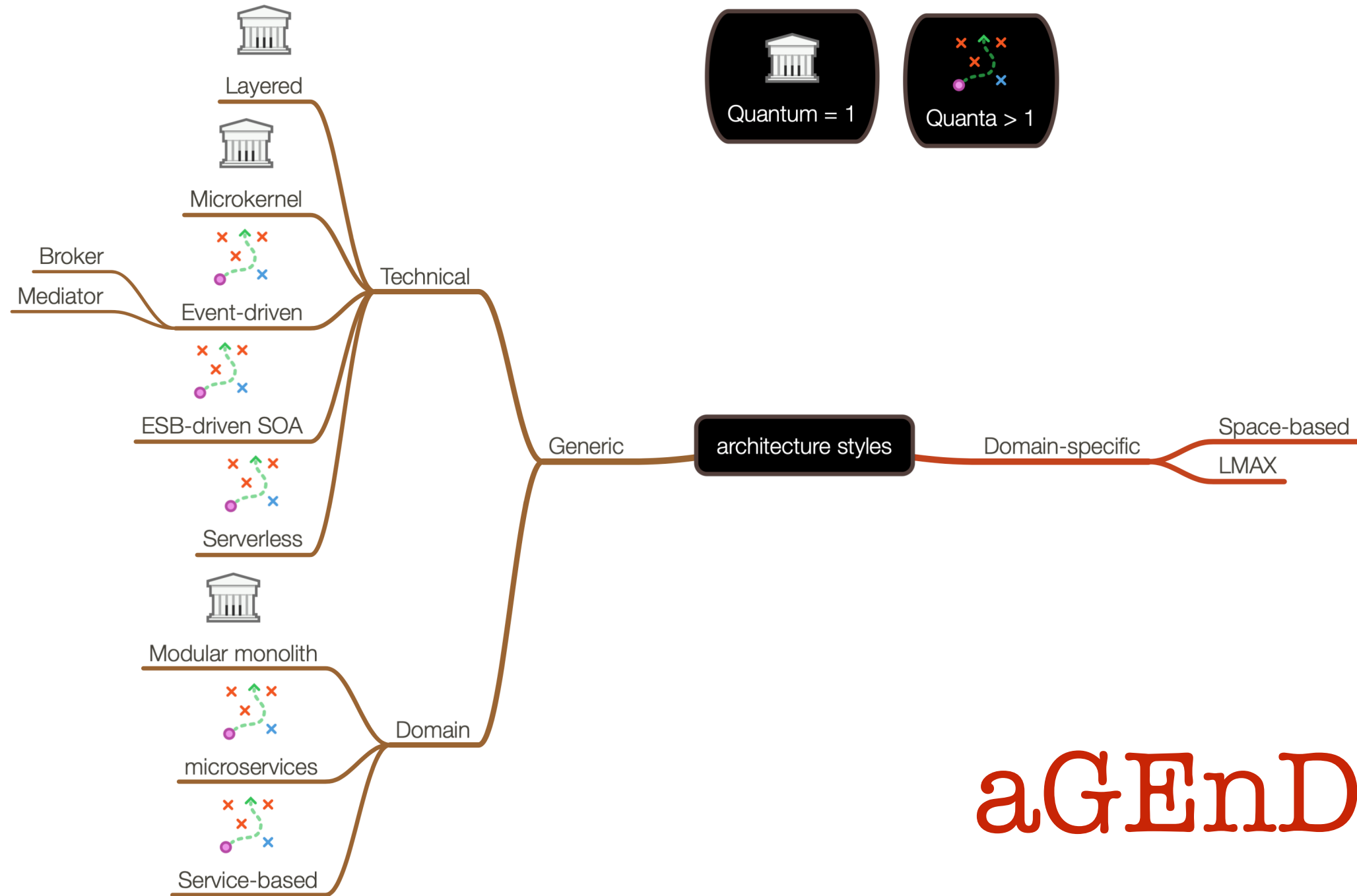
you move...

you
moved



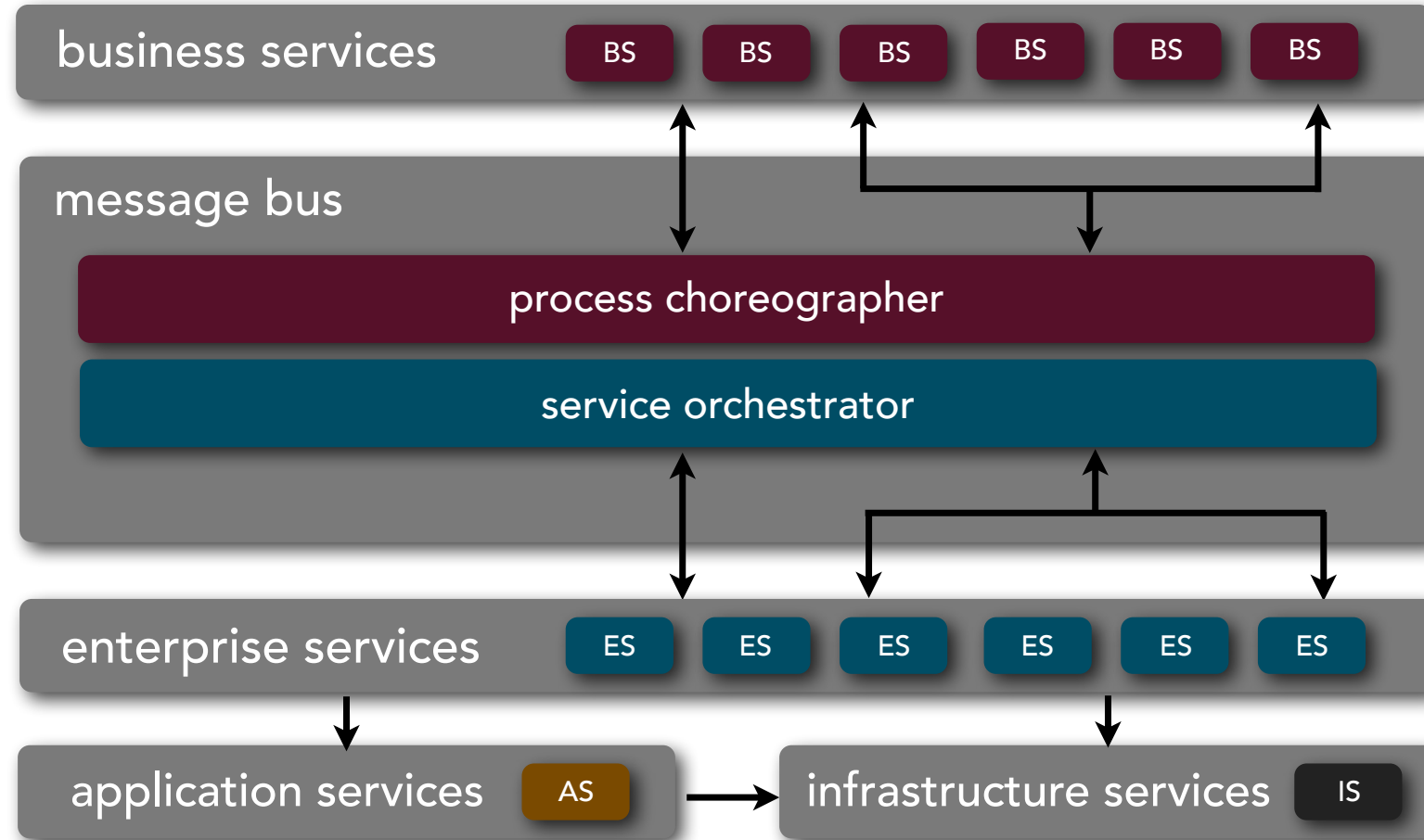
Question:

Which is more performant and more scalable—*broker* or *mediator*?

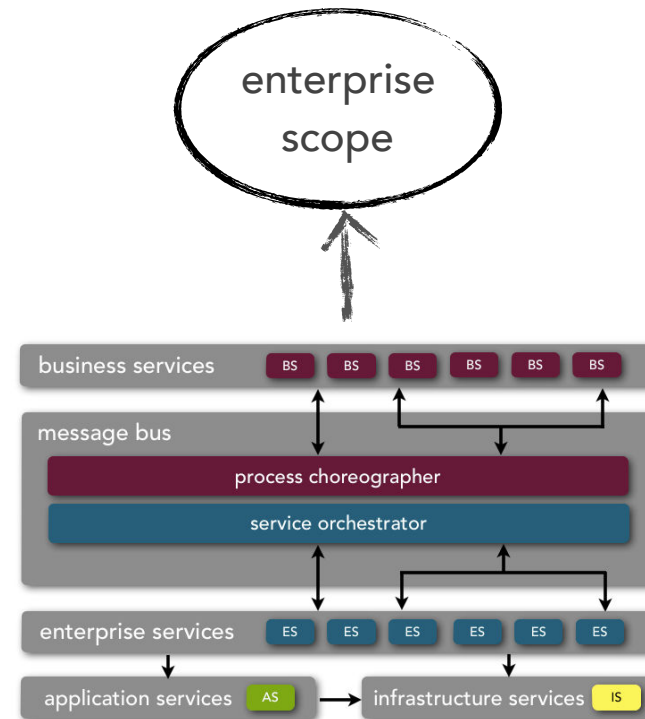


aGEnDA

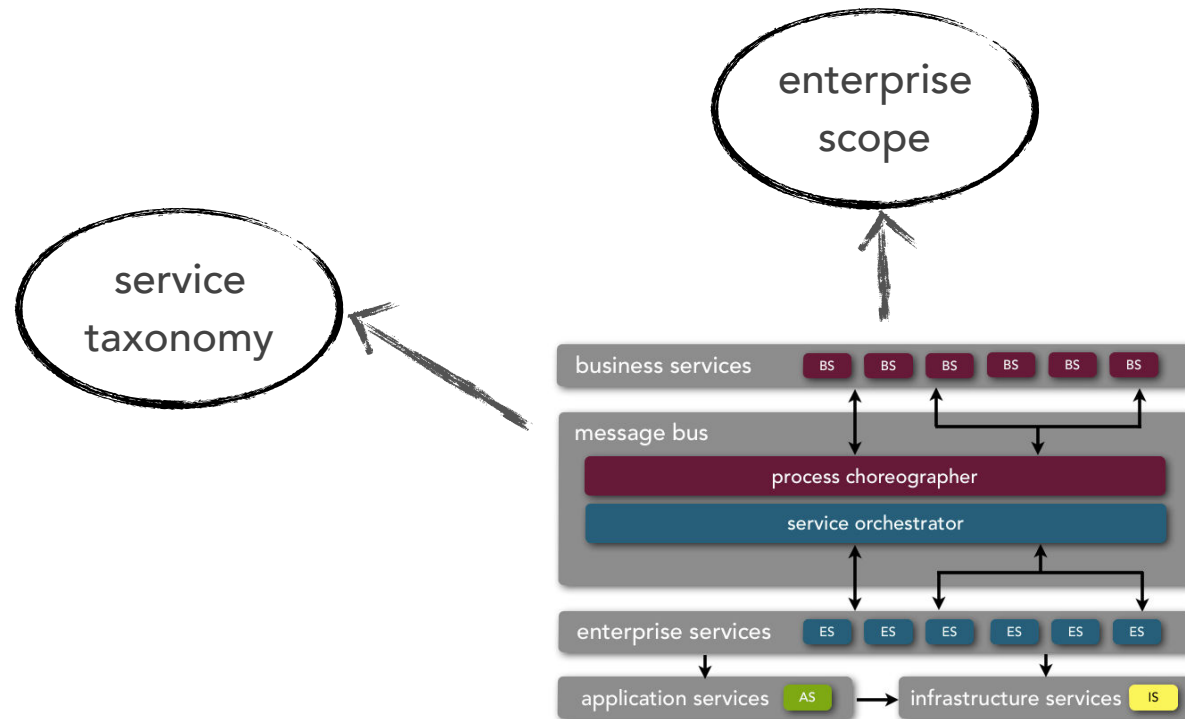
ESB-driven SOA



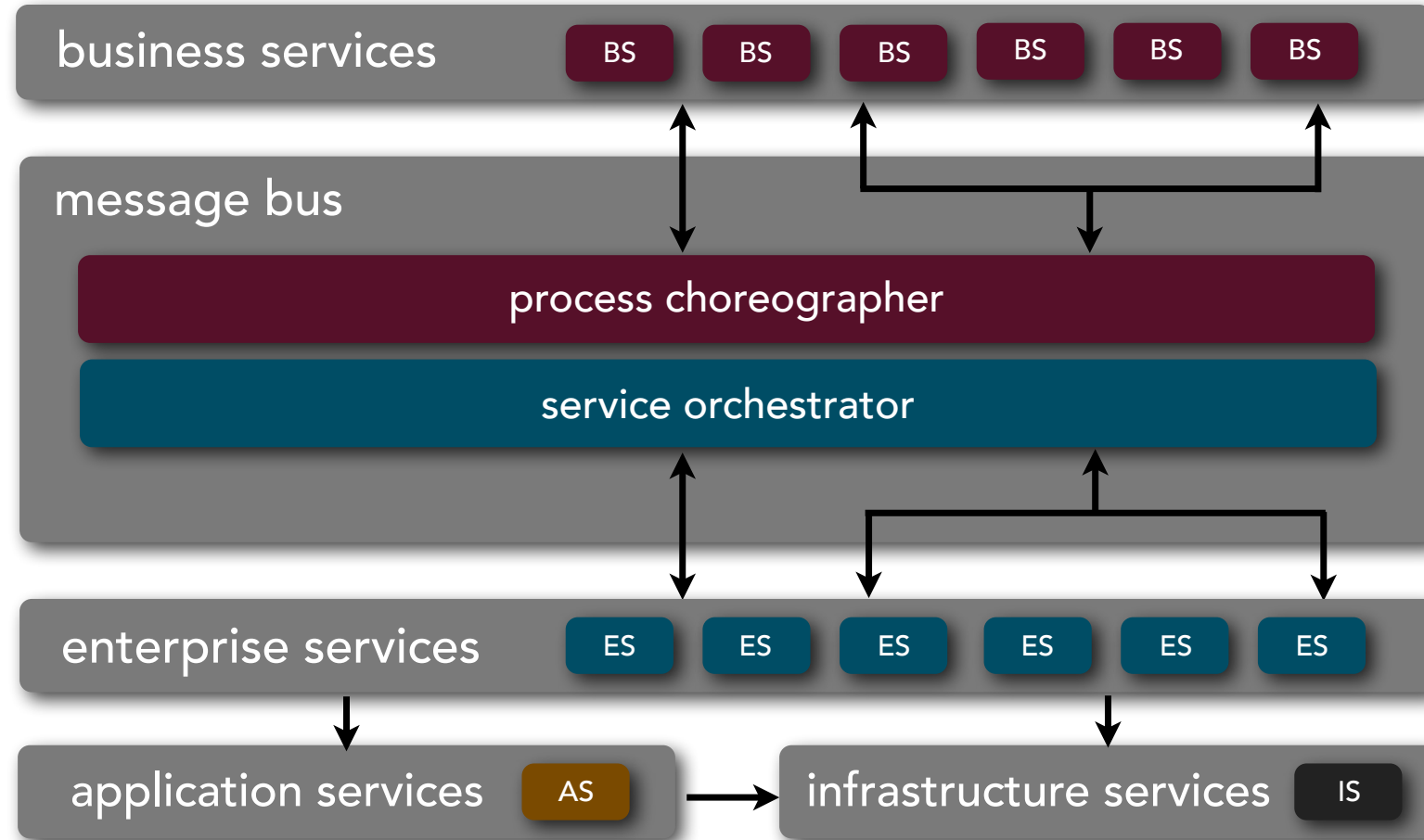
ESB-driven SOA



ESB-driven SOA

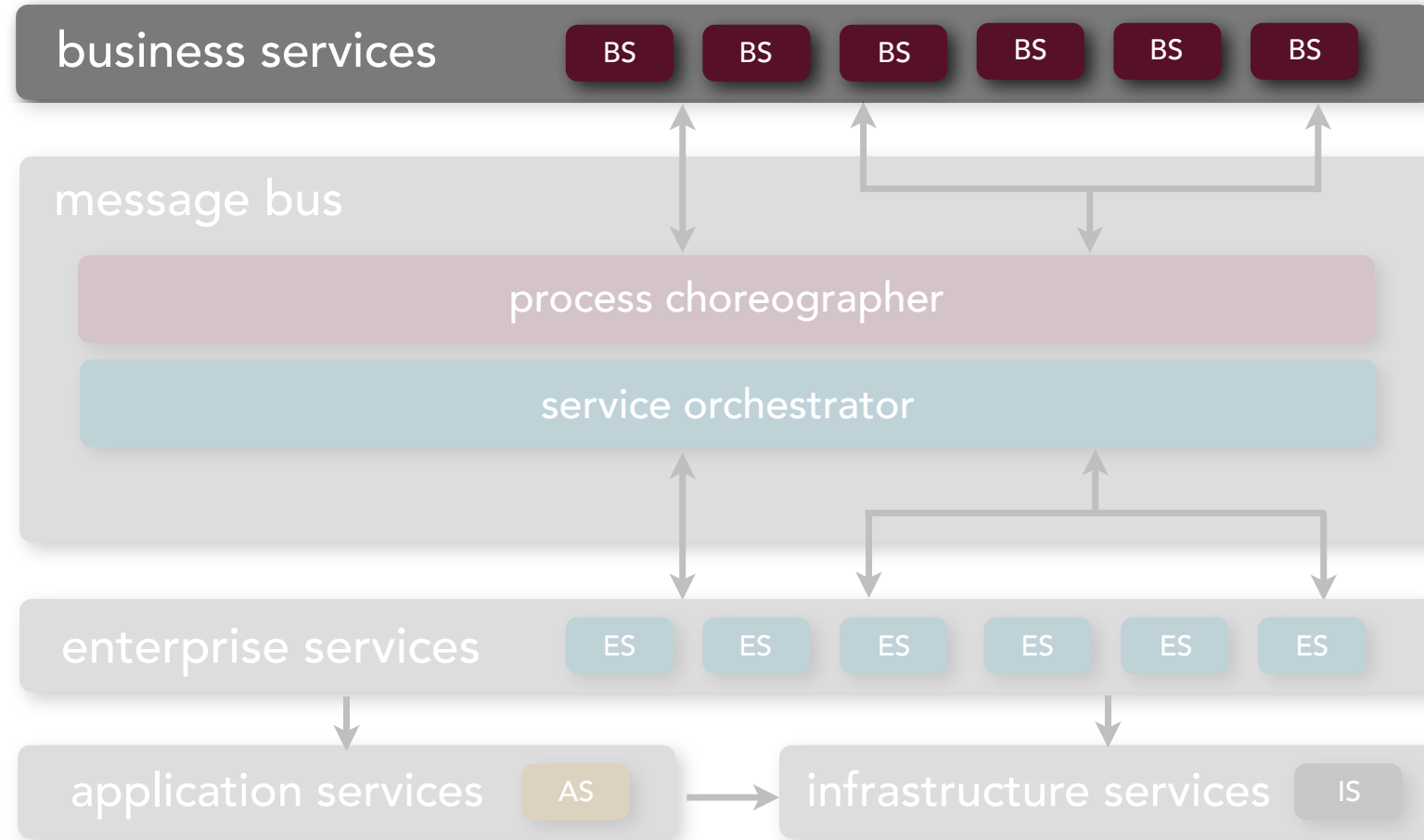


ESB-driven SOA



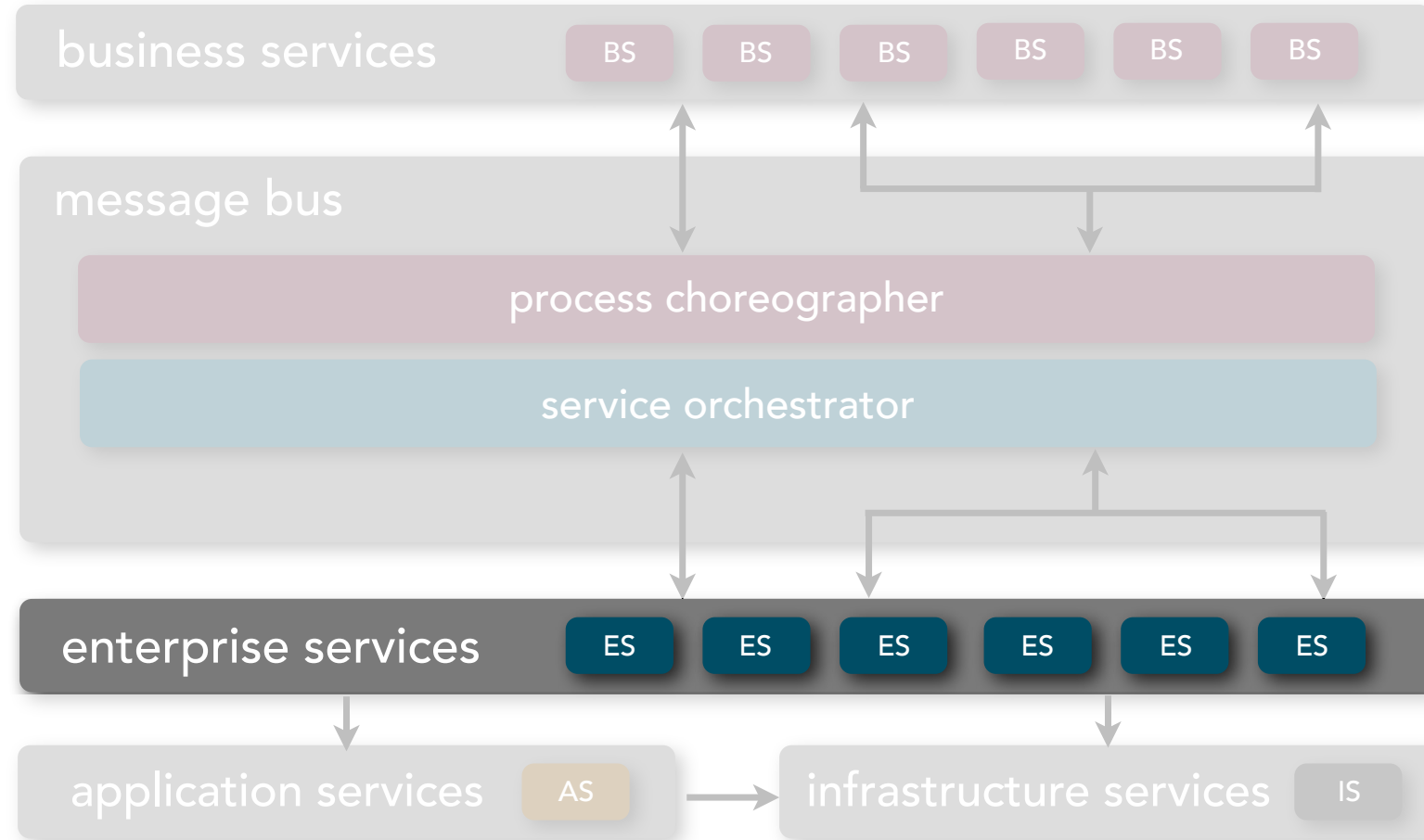
ESB-driven SOA

business services



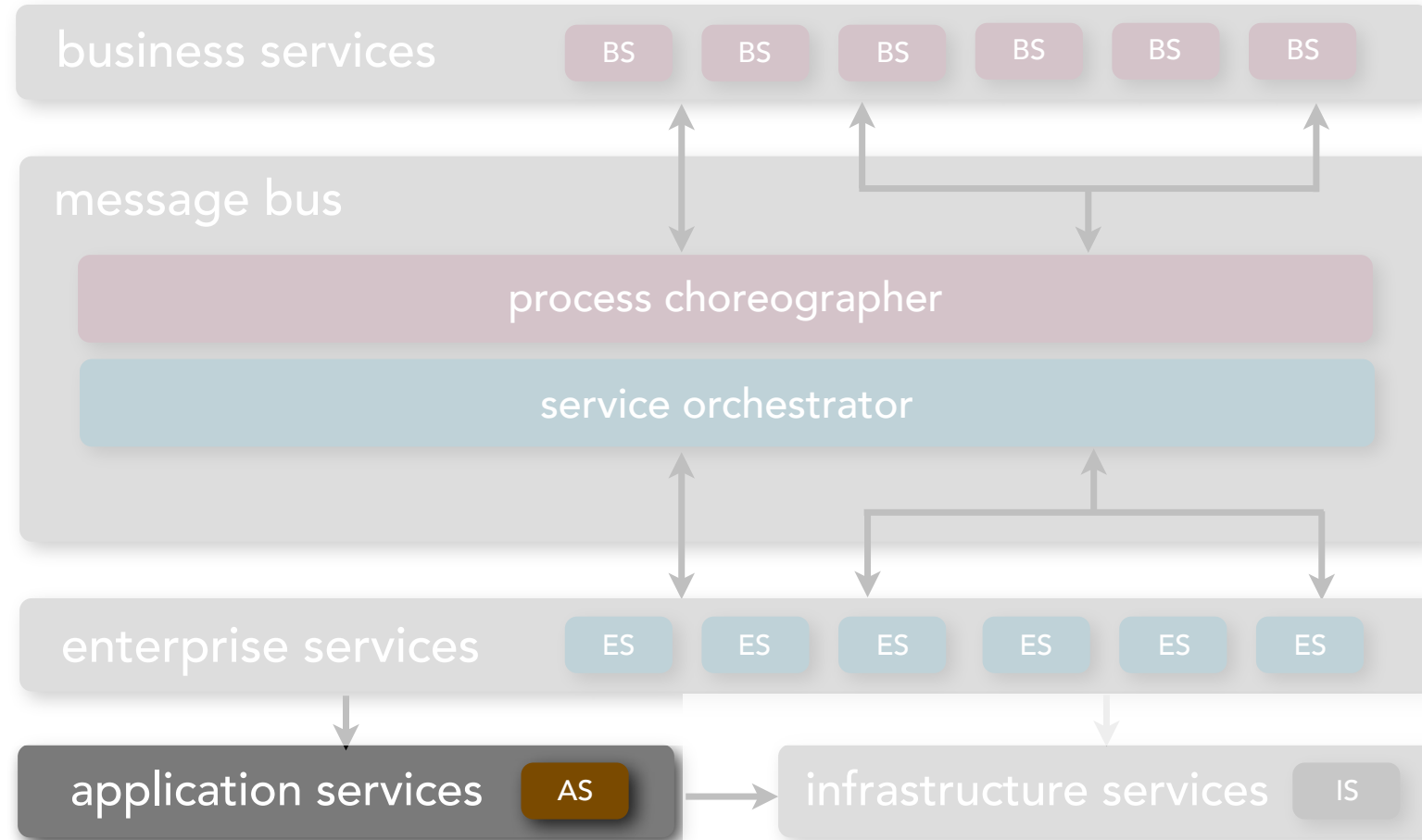
ESB-driven SOA

enterprise services



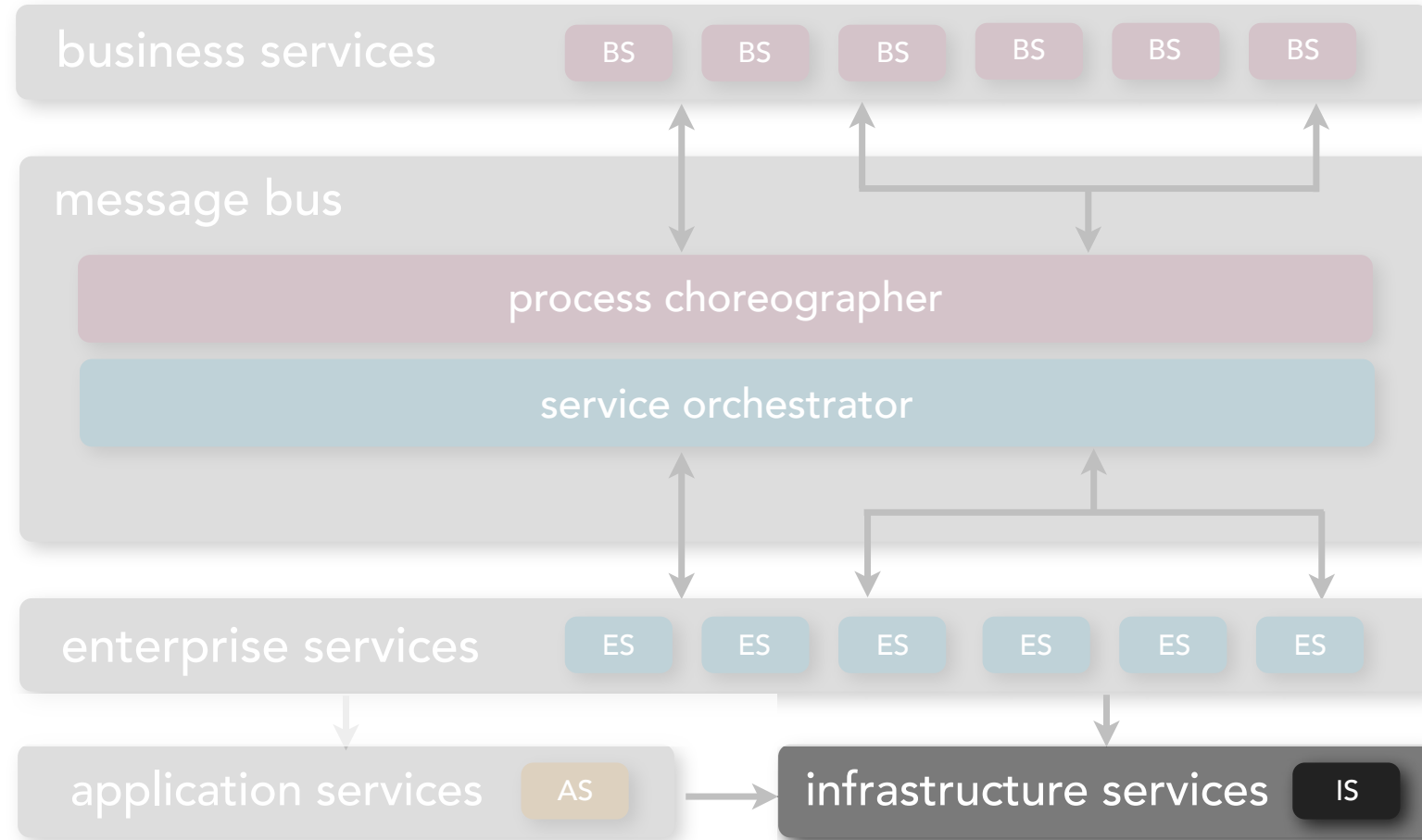
ESB-driven SOA

application services

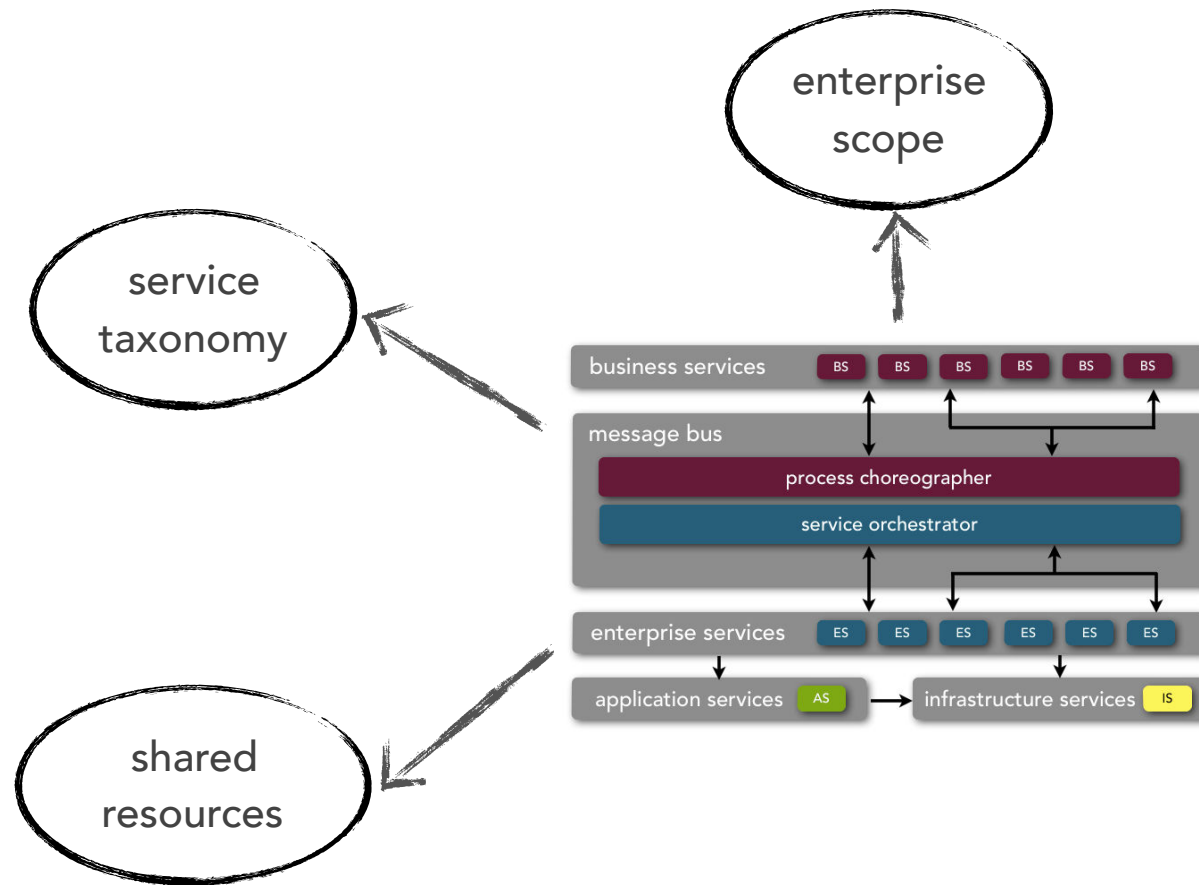


ESB-driven SOA

infrastructure services

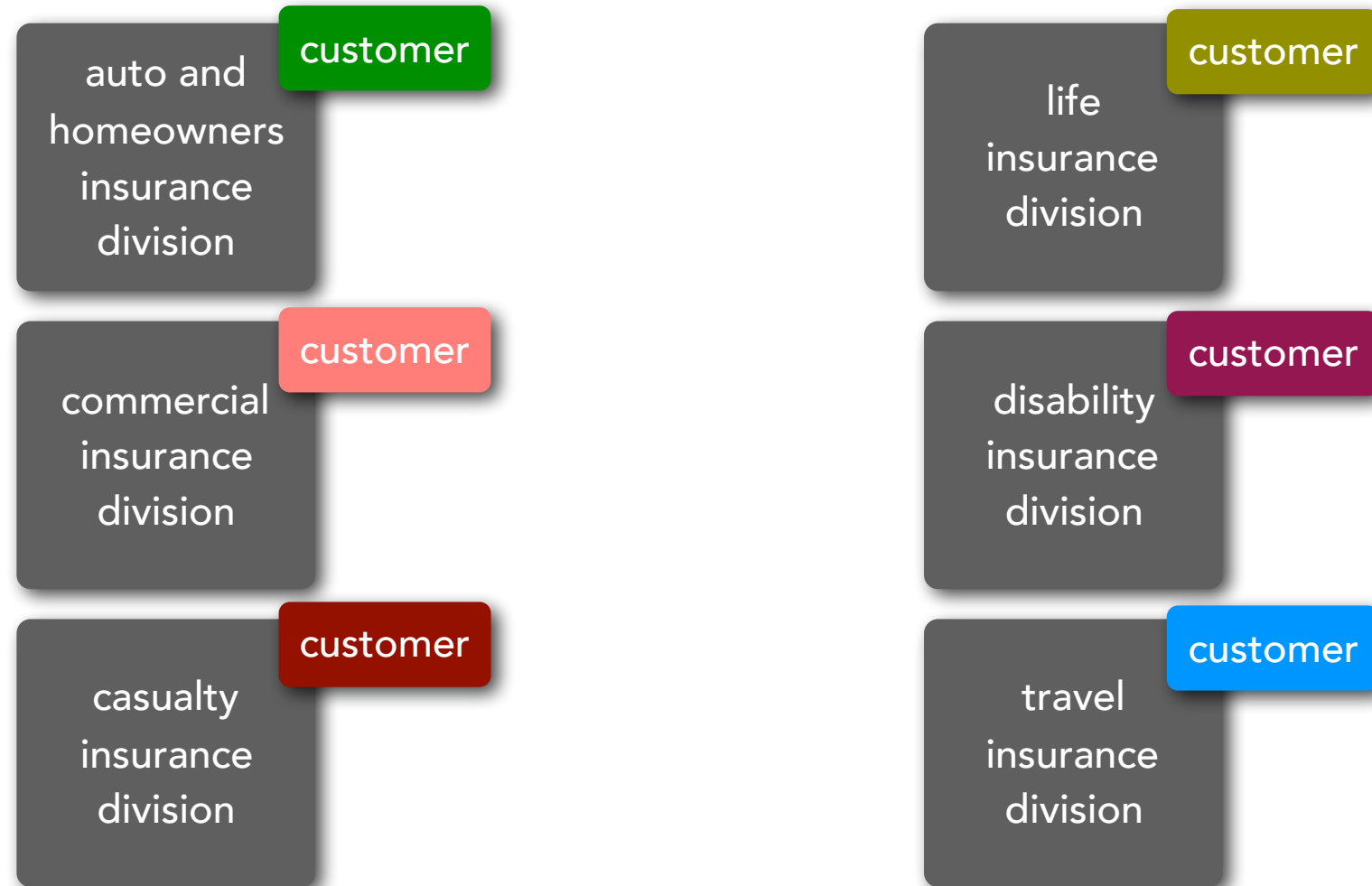


ESB-driven SOA



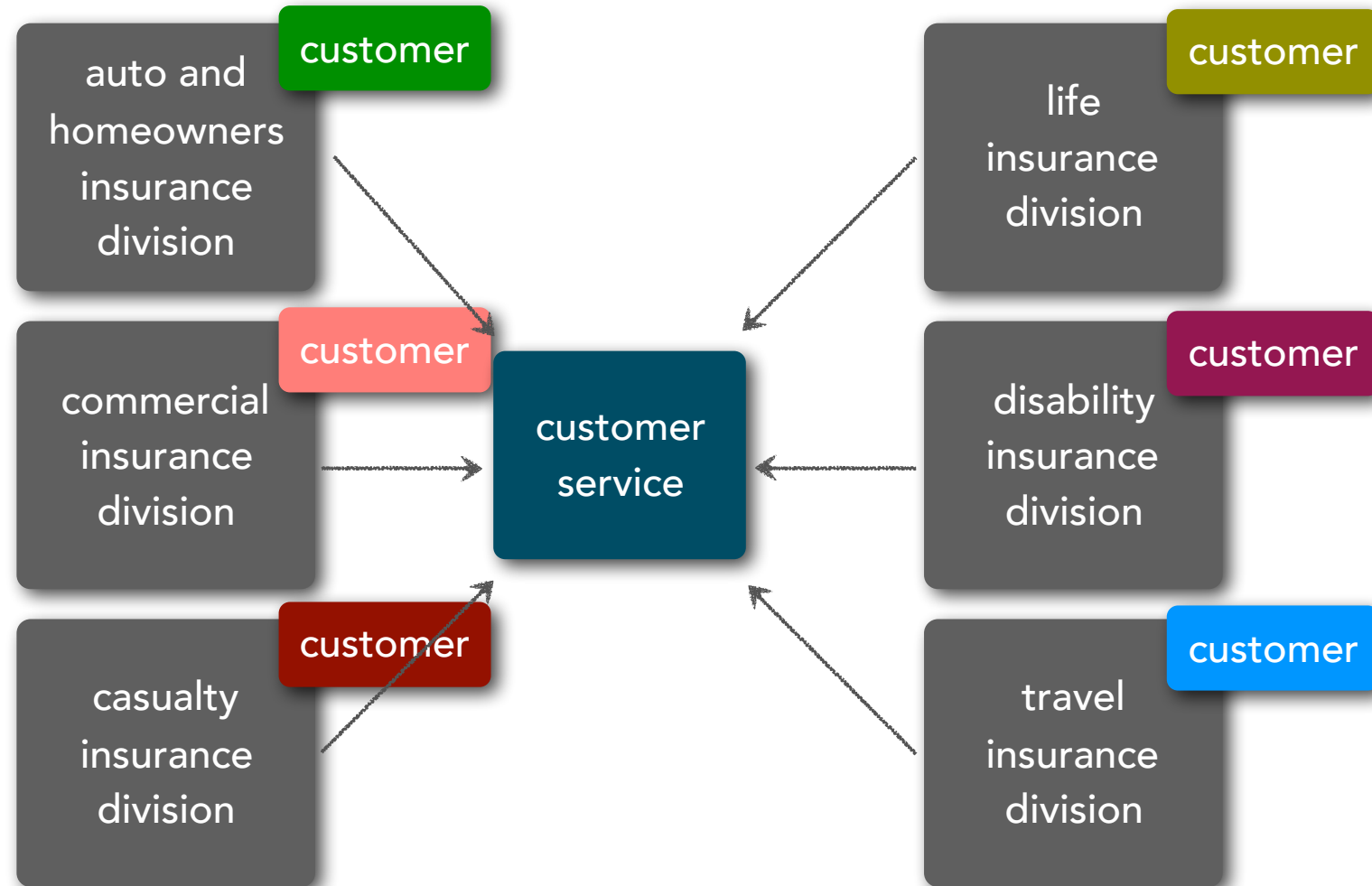
ESB-driven SOA

shared resources



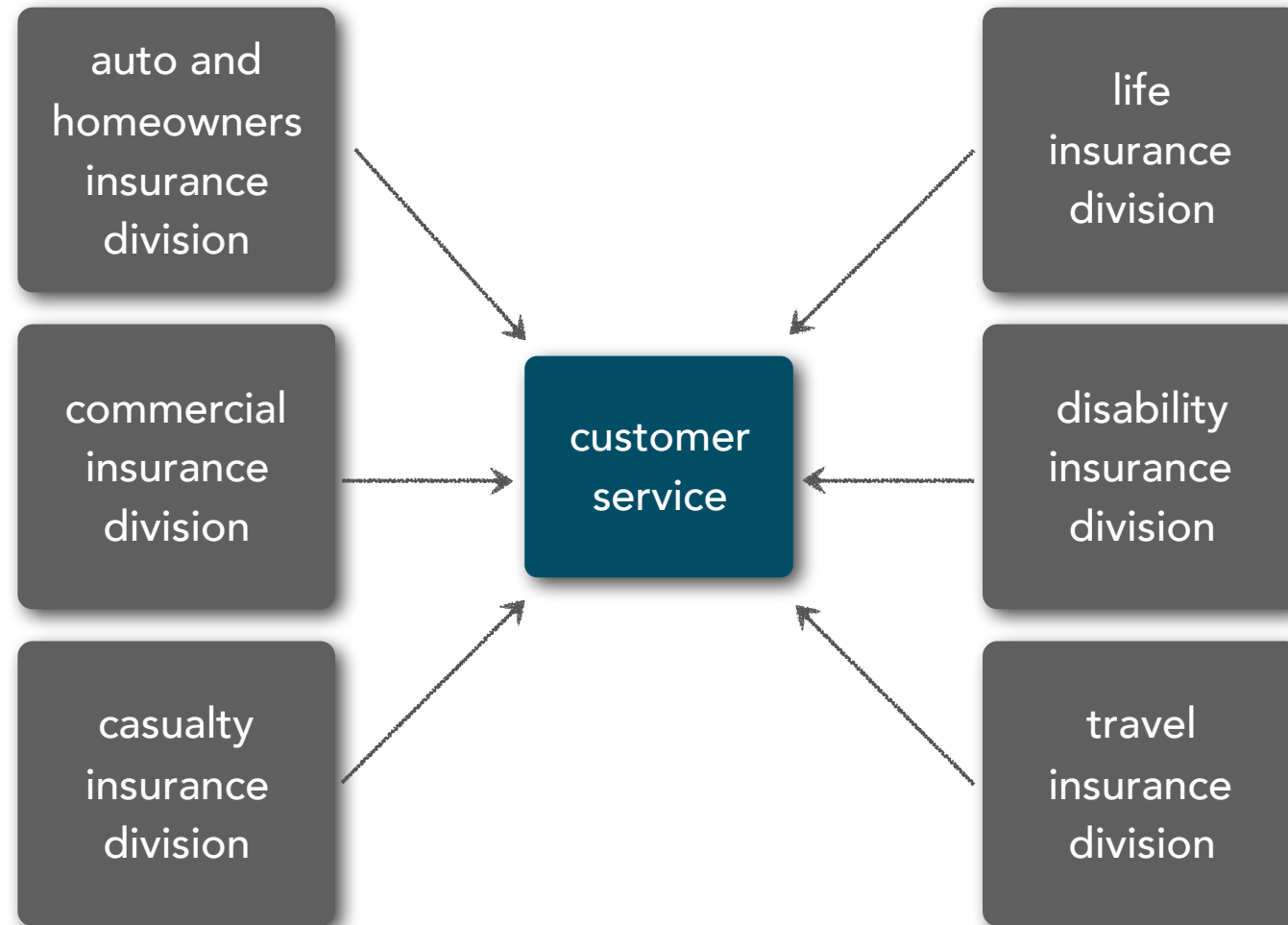
ESB-driven SOA

shared resources

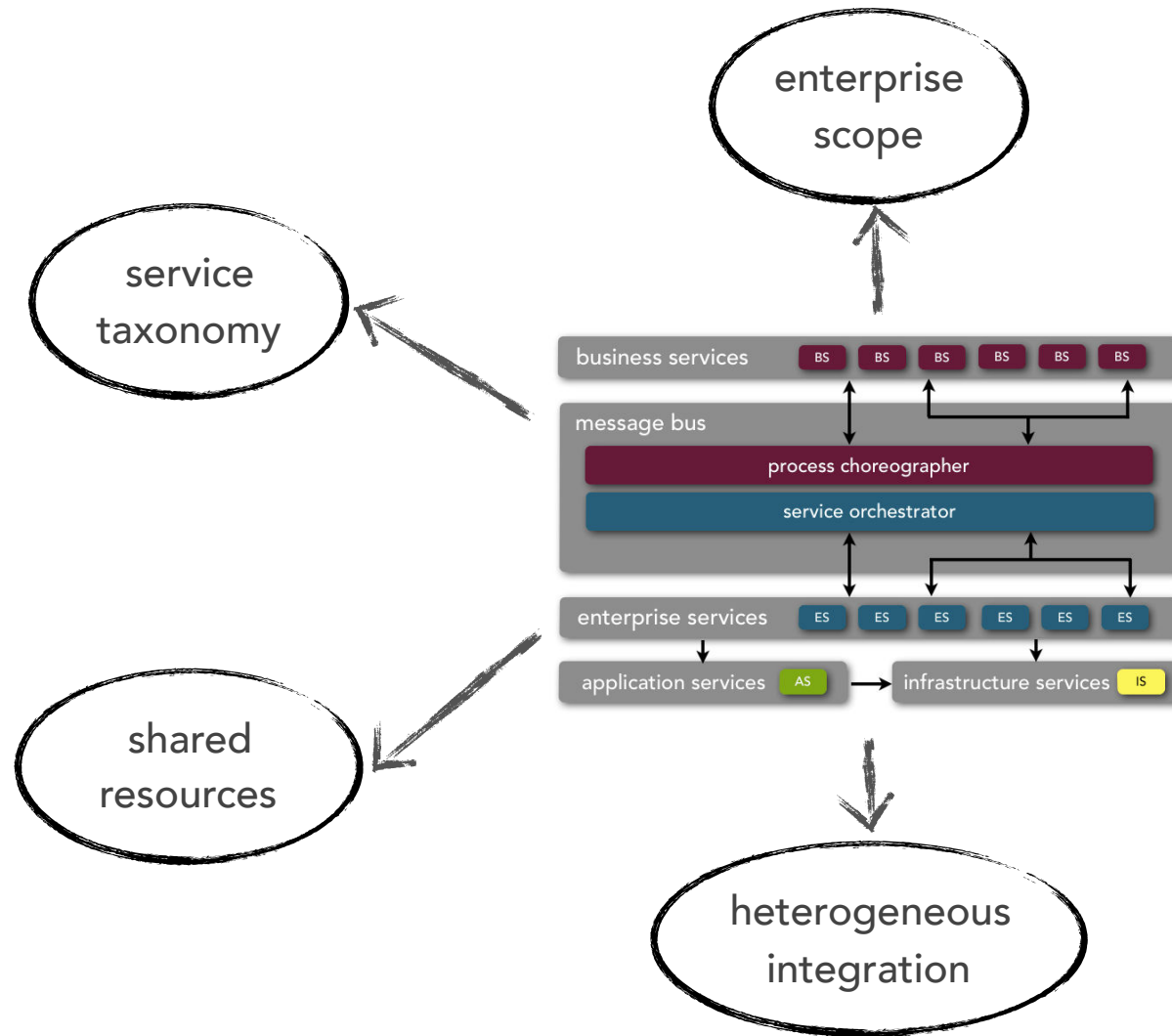


ESB-driven SOA

shared resources

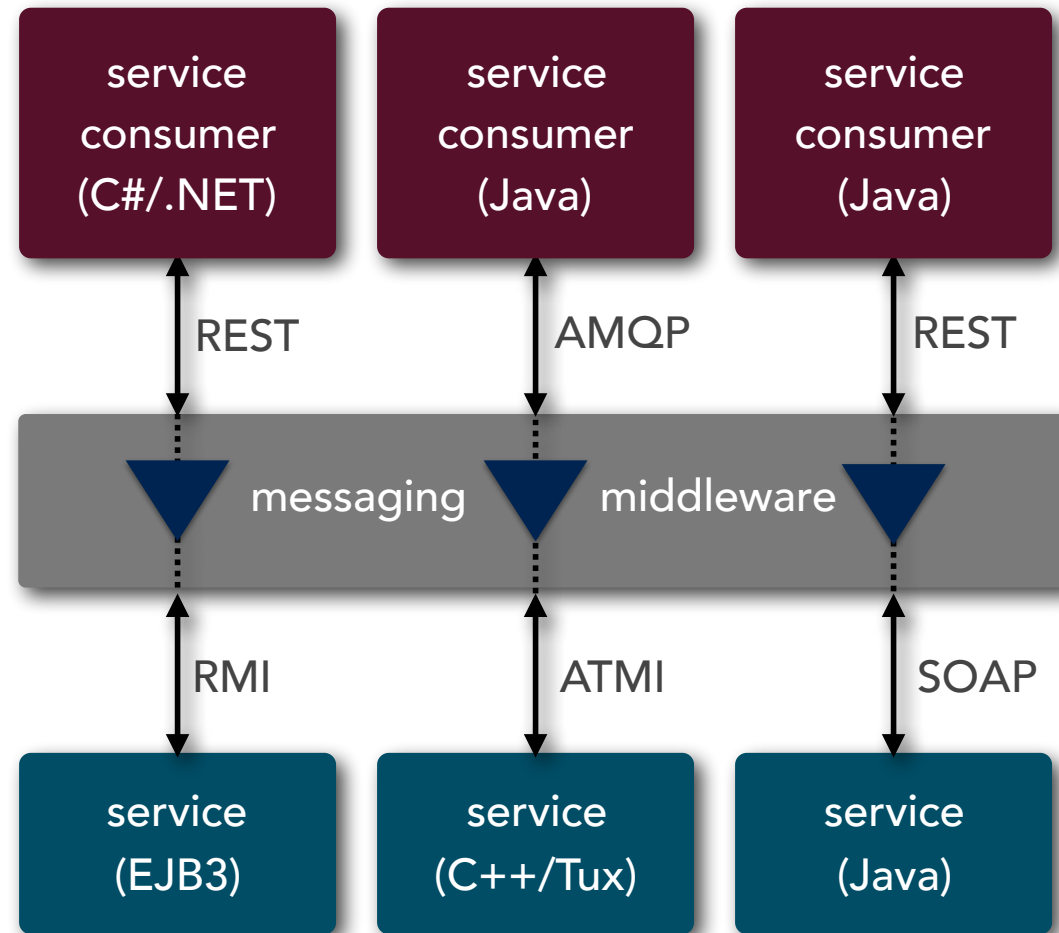


ESB-driven SOA

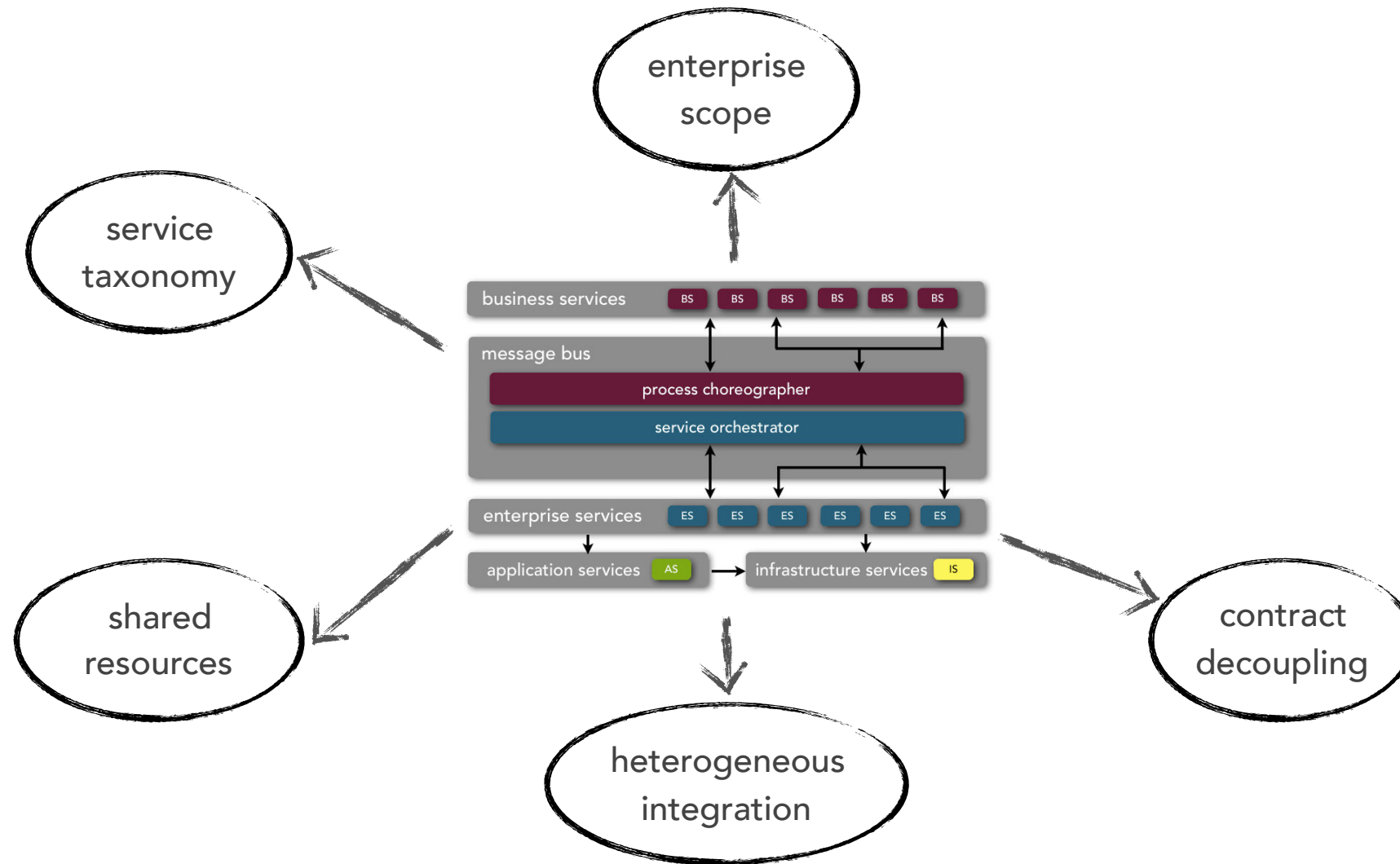


ESB-driven SOA

protocol-agnostic heterogeneous interoperability

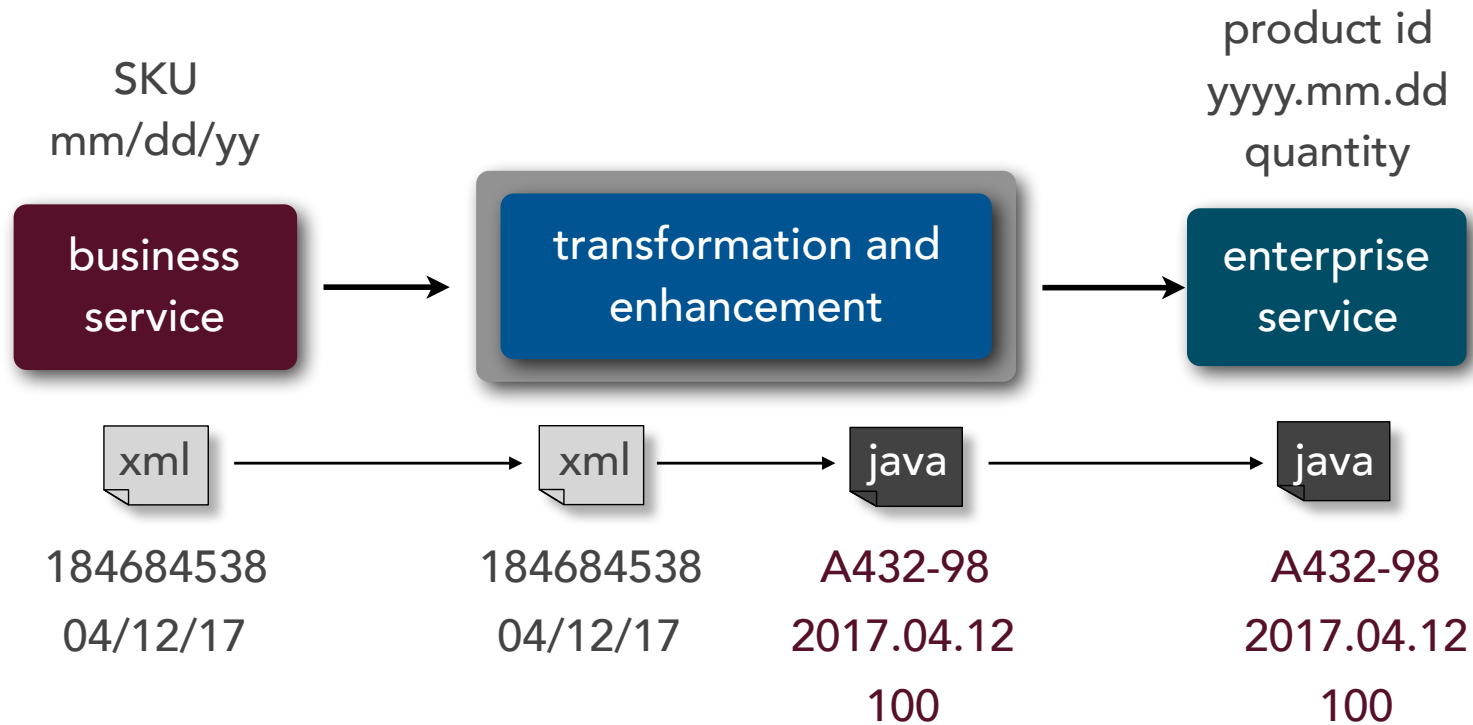


ESB-driven SOA

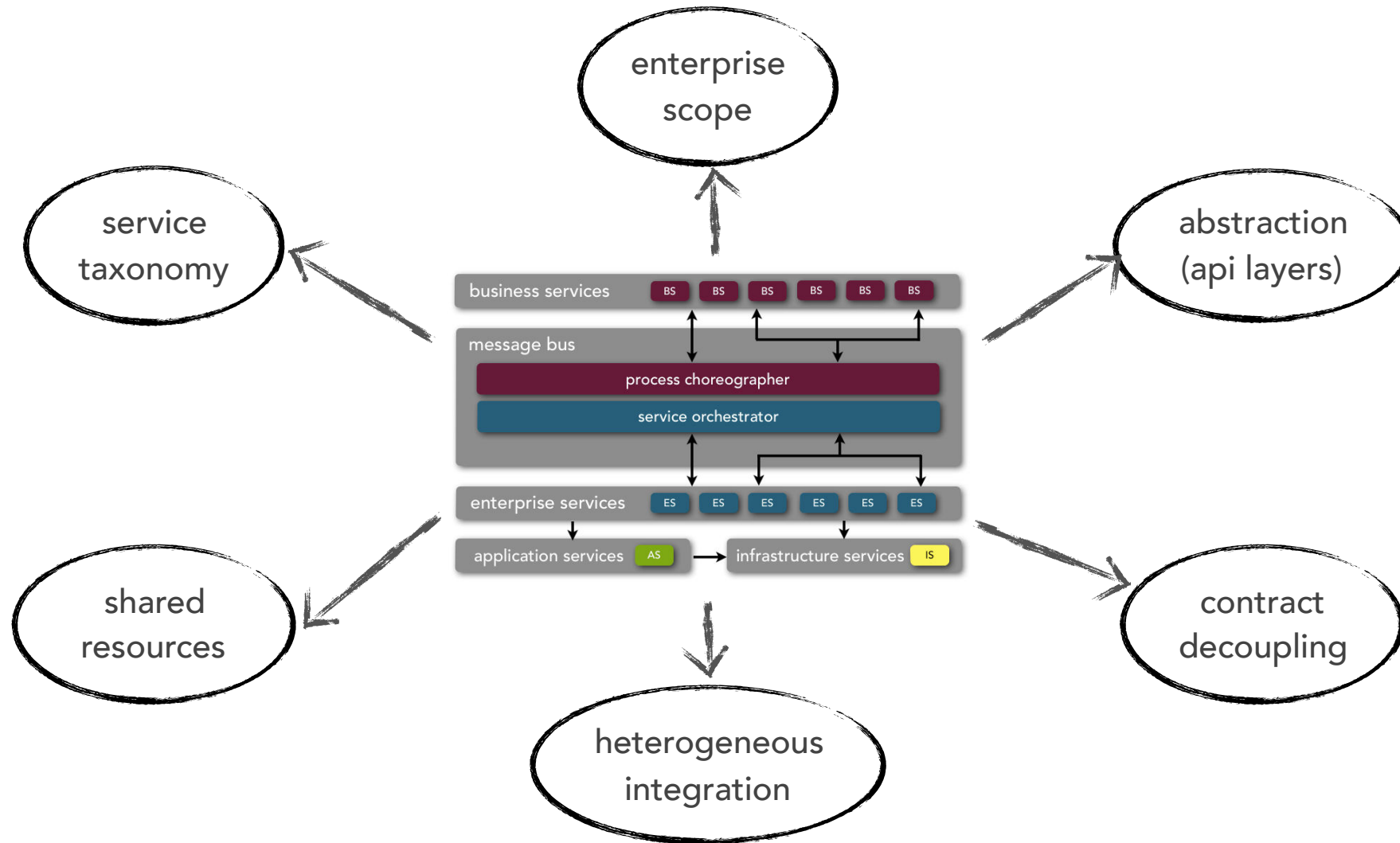


ESB-driven SOA

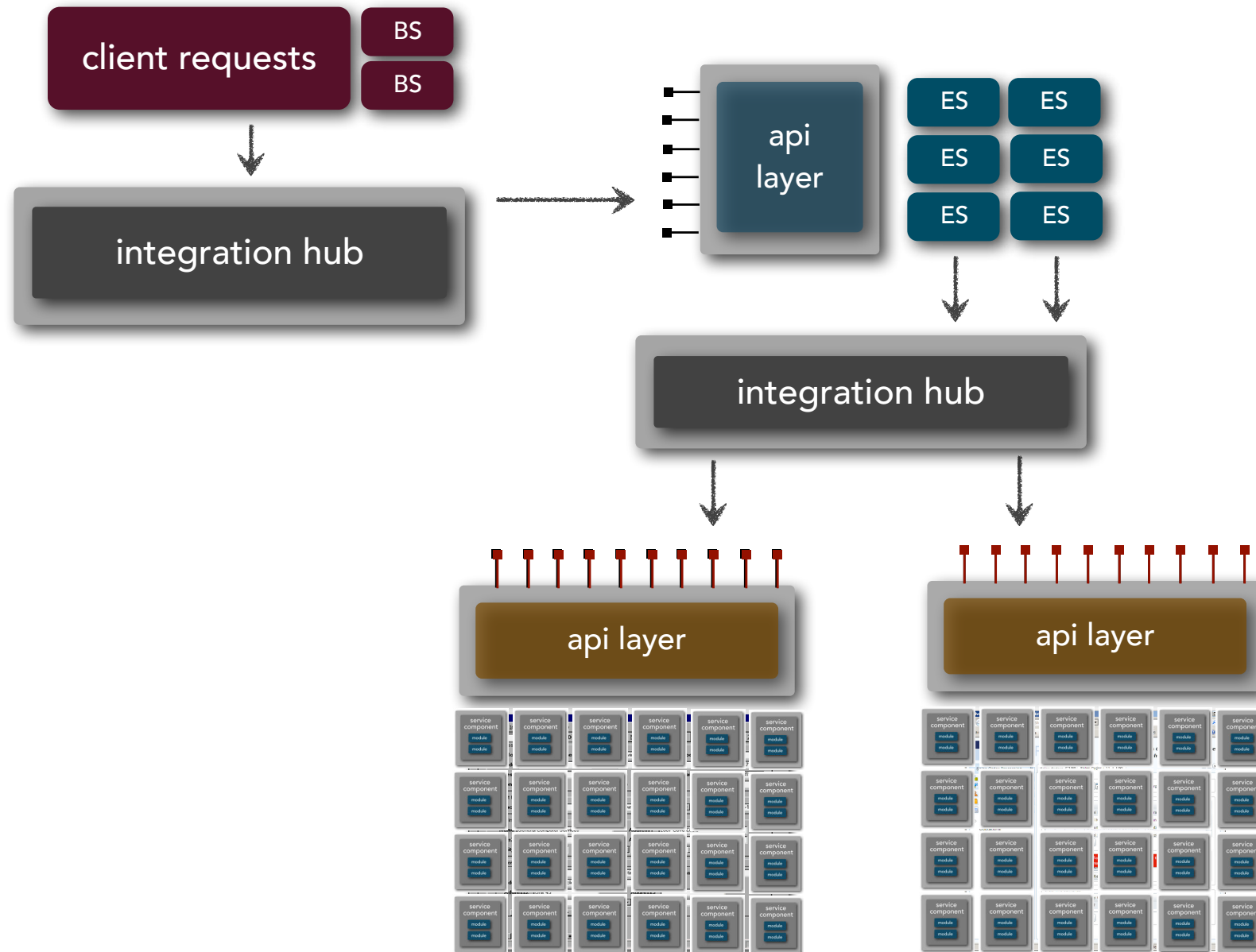
contract decoupling



service-oriented architecture

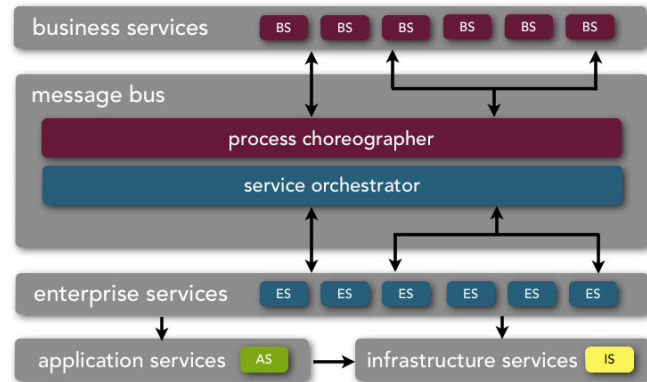


ESB-driven SOA

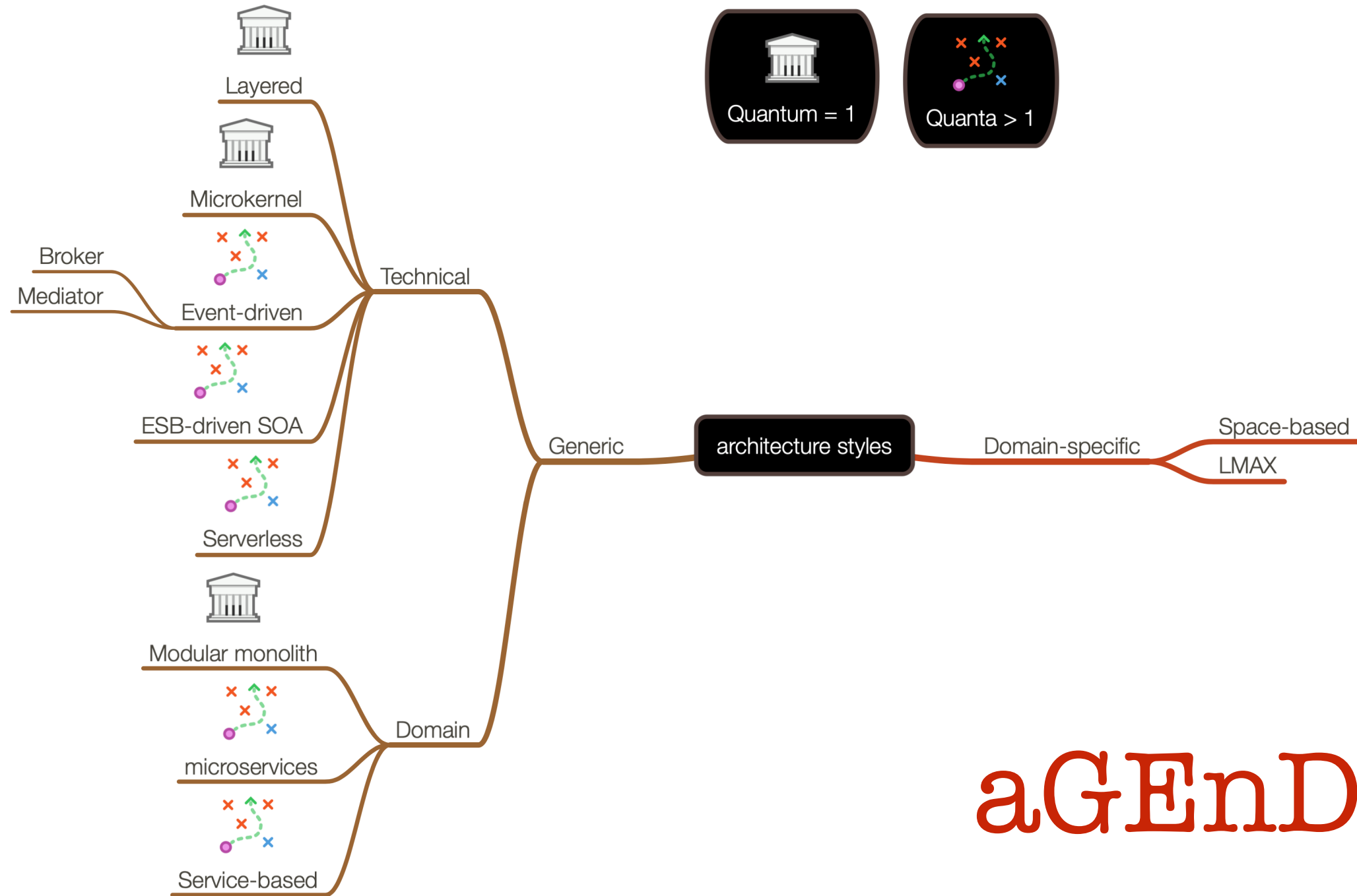


ESB-driven SOA

drivers

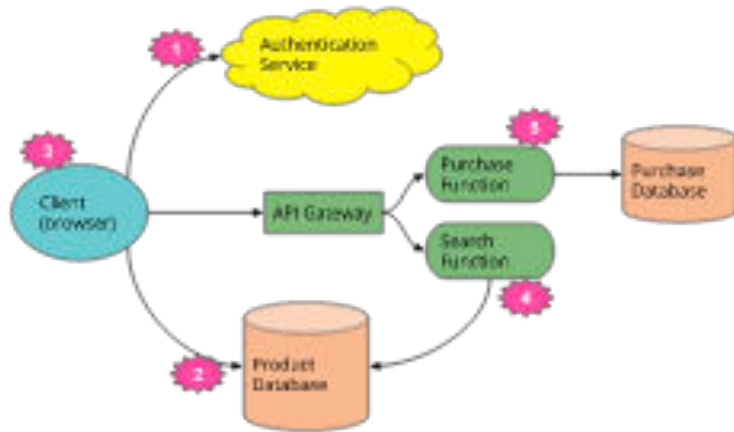


- ✓ integration
- ✓ integration
- ✓ orchestration



aGEnDA

Serverless Architectures

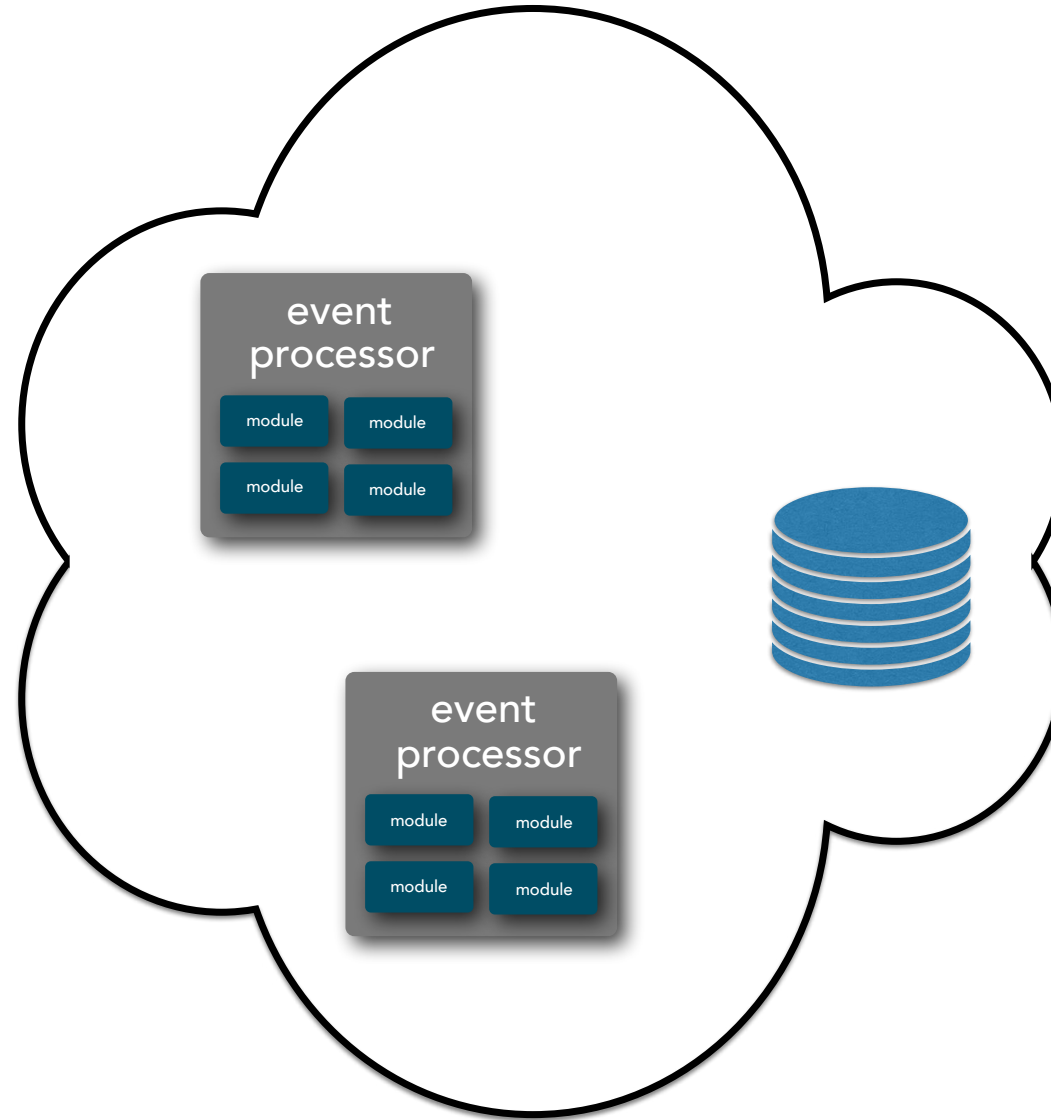


BaaS



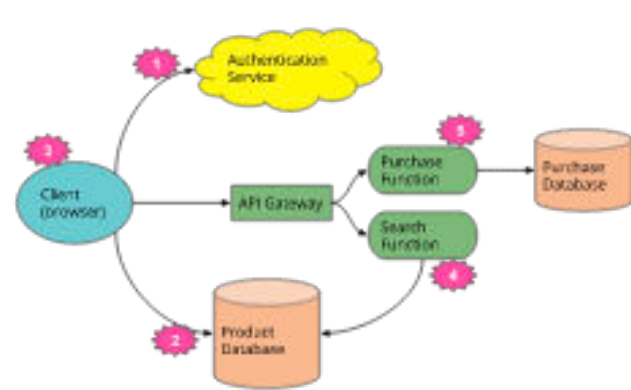
FaaS

Serverless



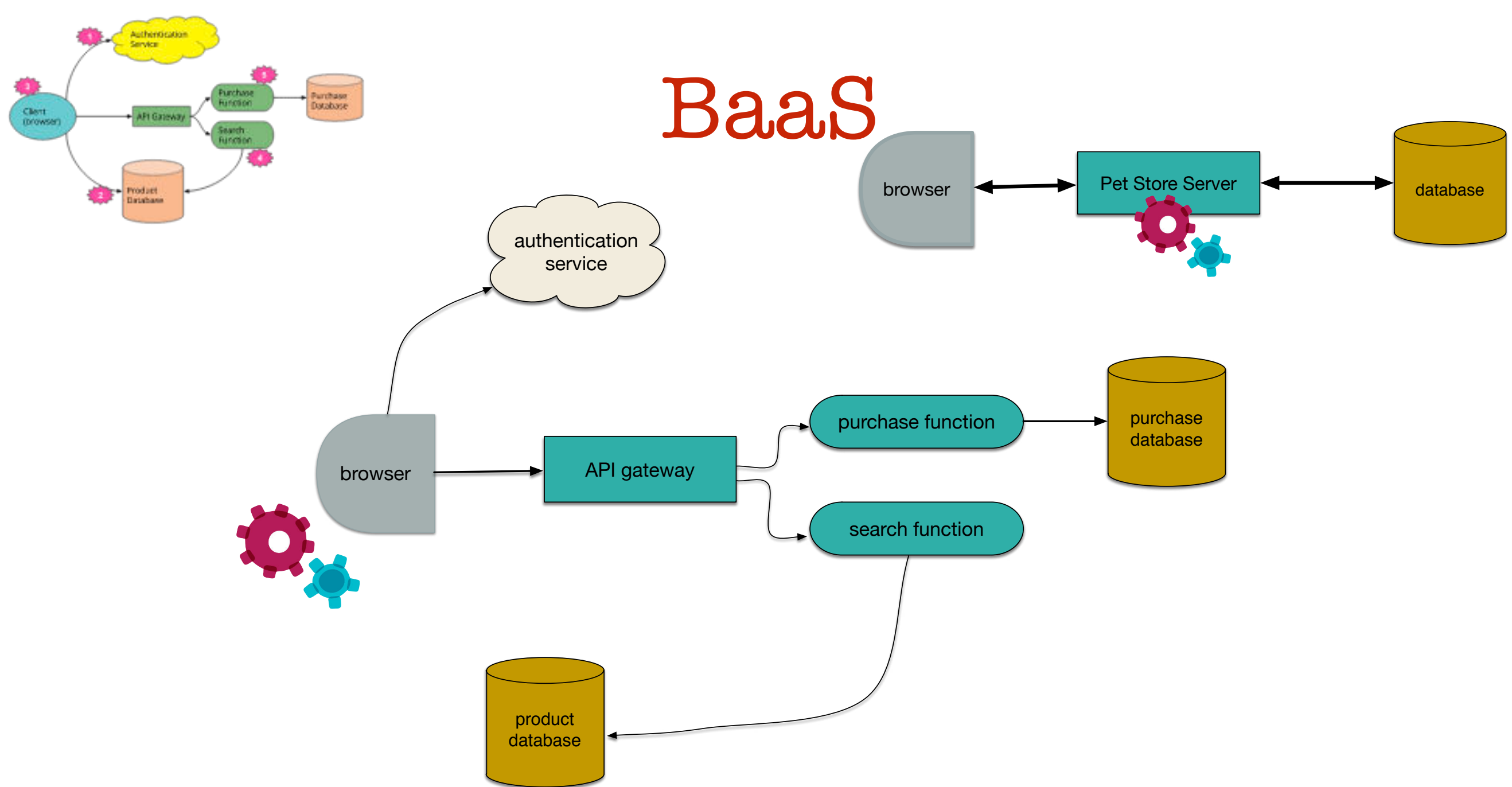
BaaS

(Backend as a Service)

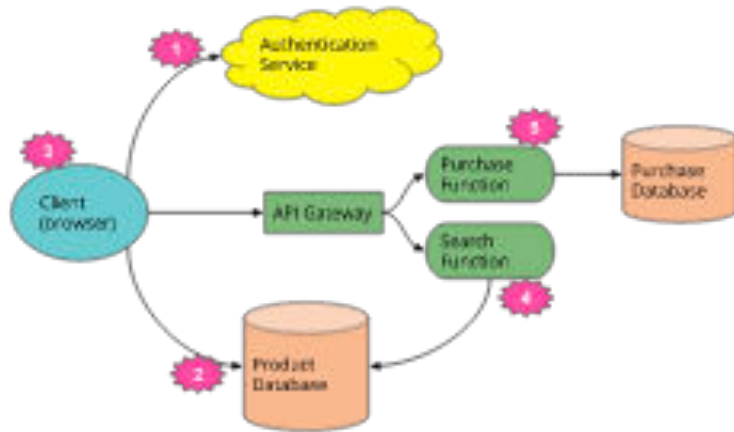


traditional client/server architecture

BaaS



Serverless Architectures



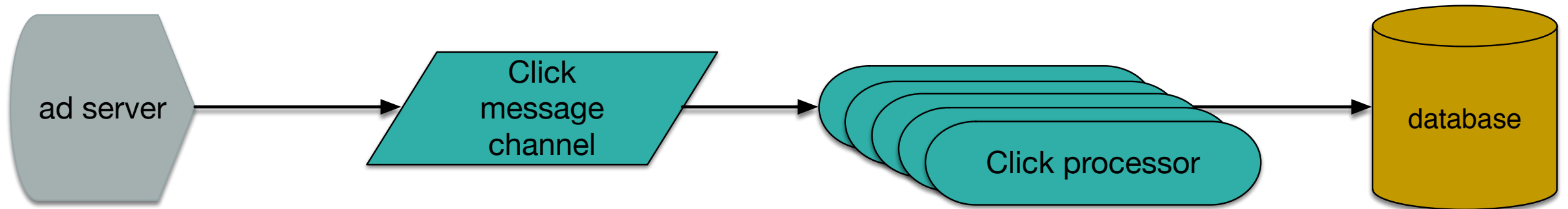
BaaS



FaaS

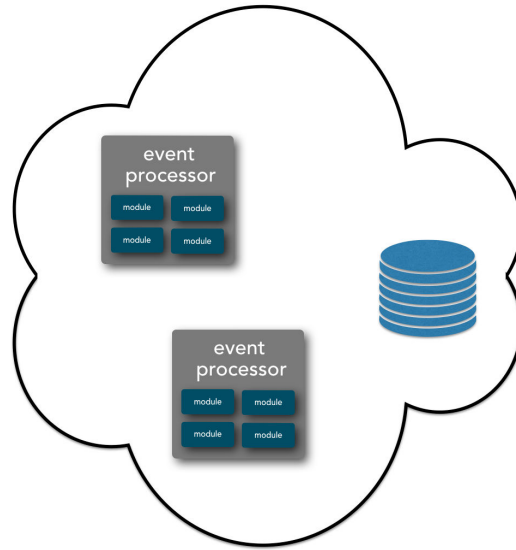
FaaS

(Functions as a Service)



Serverless

drivers



- ✓ agility
- ✓ lack of (apparent) infrastructure
- ✓ discrete functionality

Last 10% Trap

"Users always want 100% of what they want (& are never satisfied with less)."



80%

10% 10%



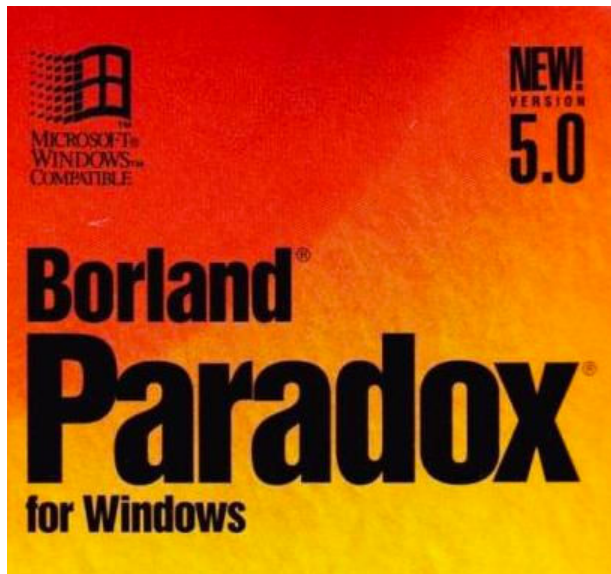
what the user wants

What happened to the 4GLs?

dBASE™

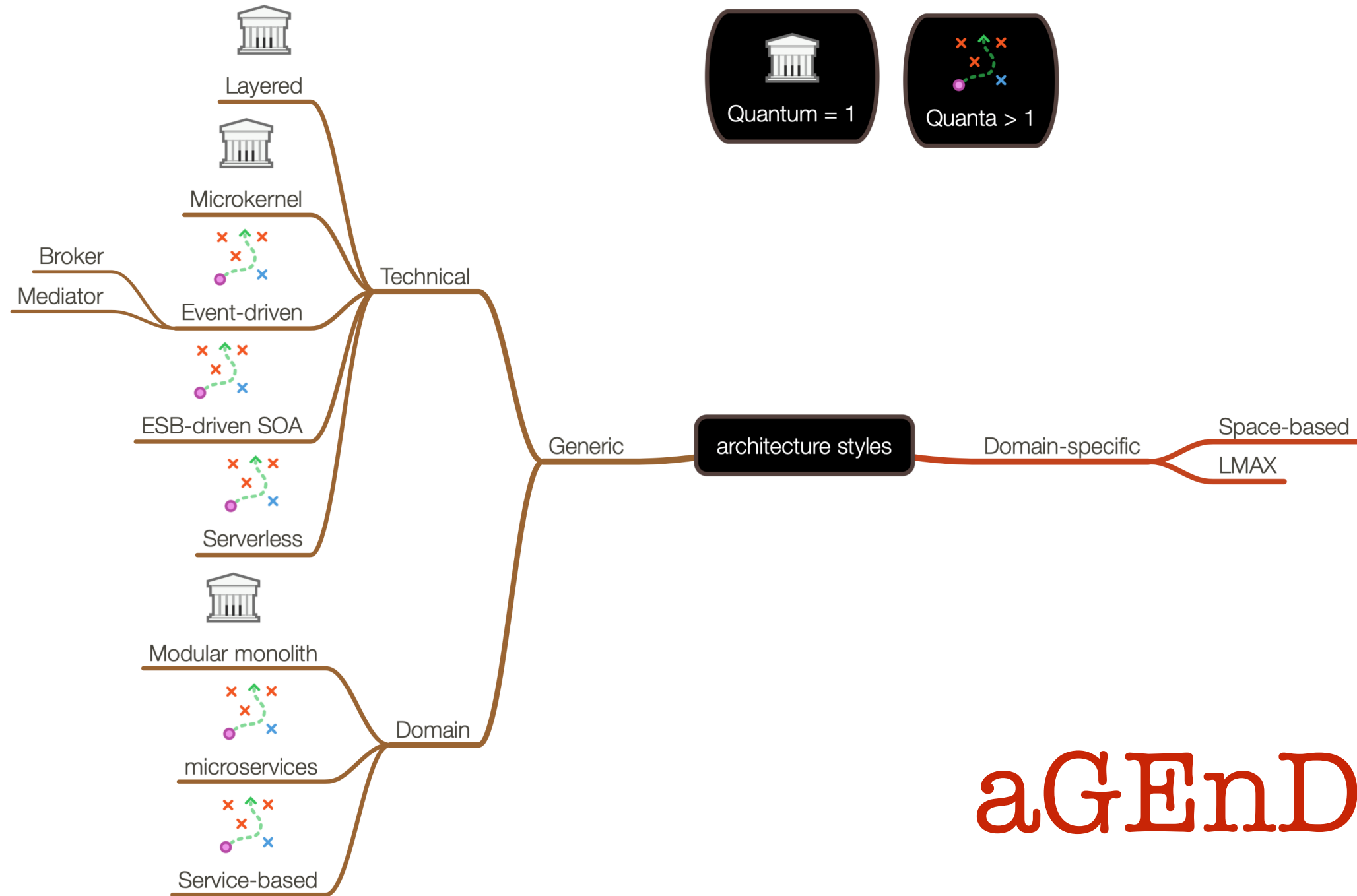


Microsoft®
Visual FoxPro®



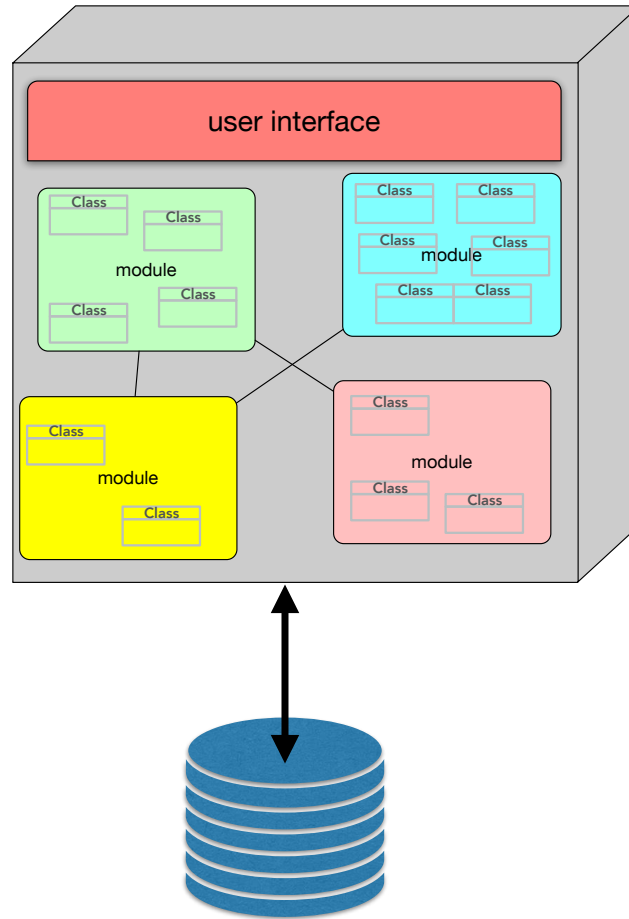
DSL





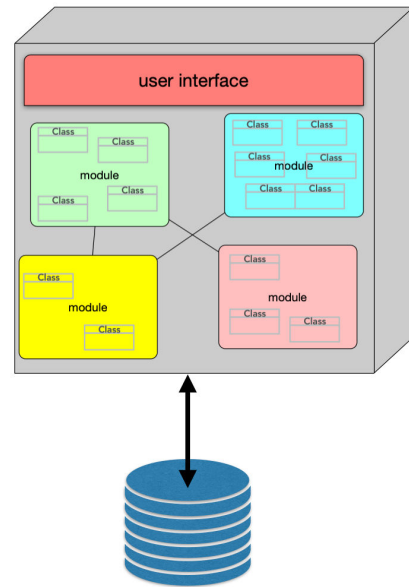
aGEnDA

Modular Monolith

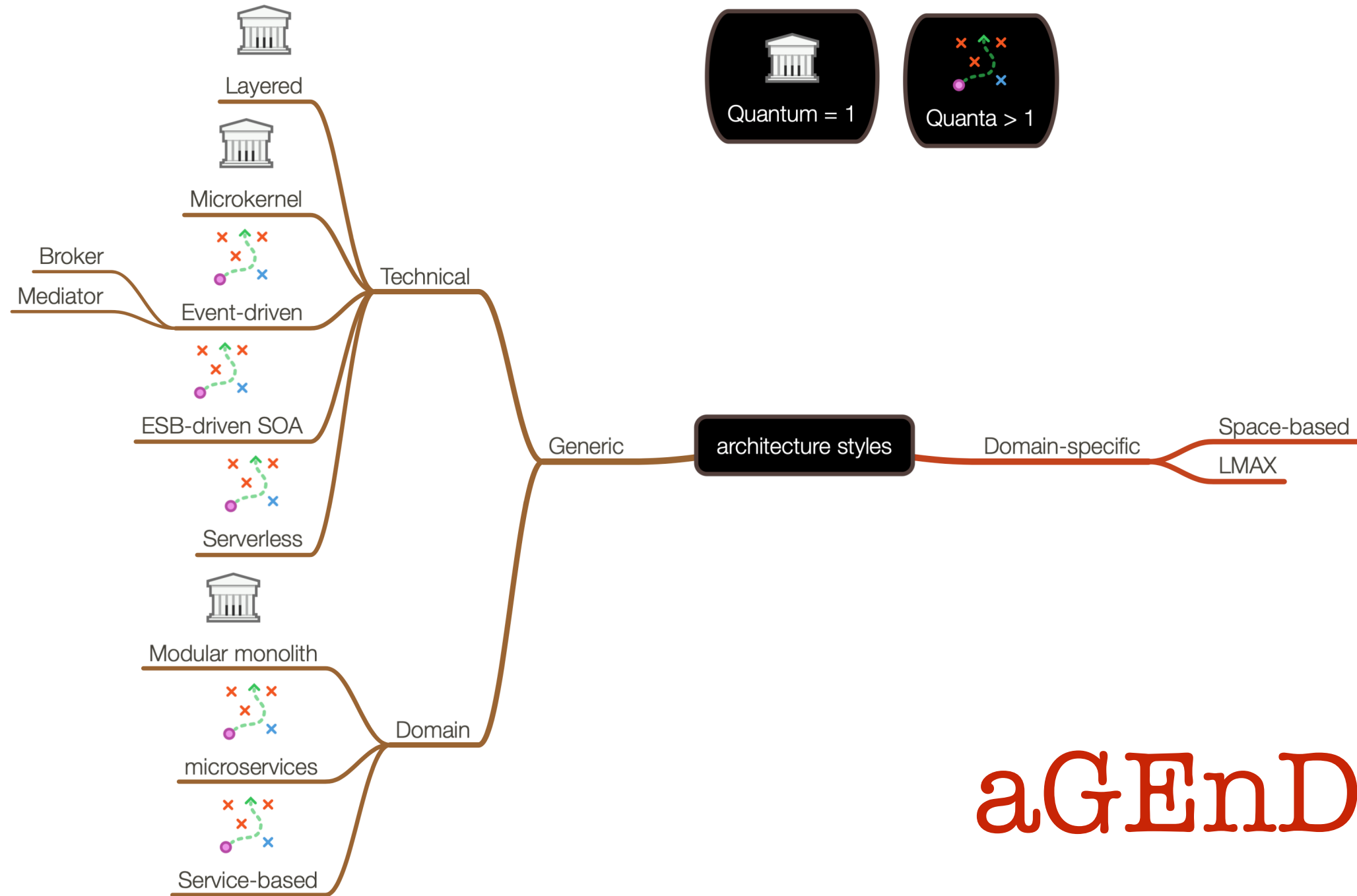


Modular Monolith

drivers

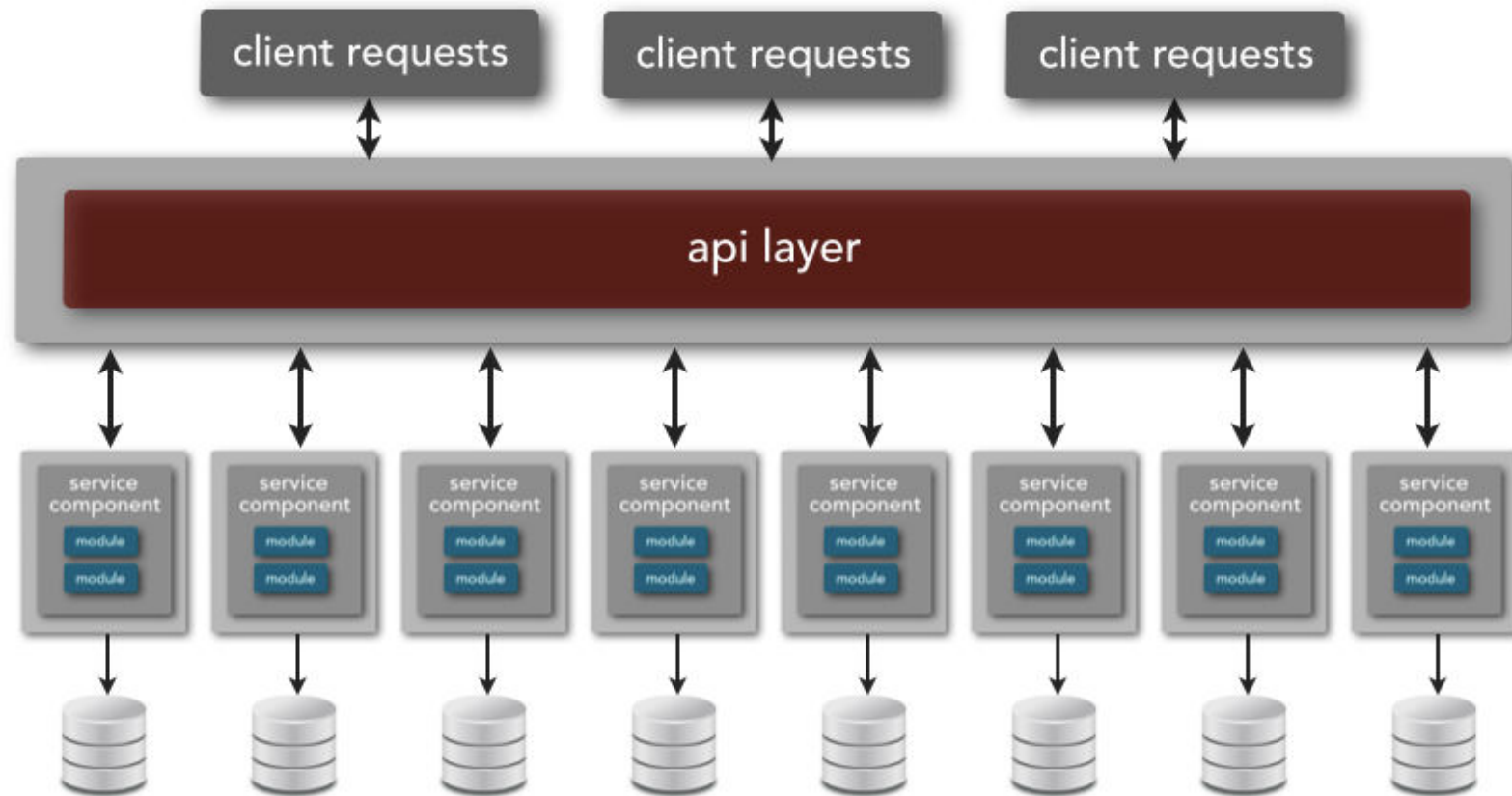


- ✓ agility
- ✓ inverse Conway Maneuver
- ✓ modularity
- ✓ deployability
- ✓ restructurability

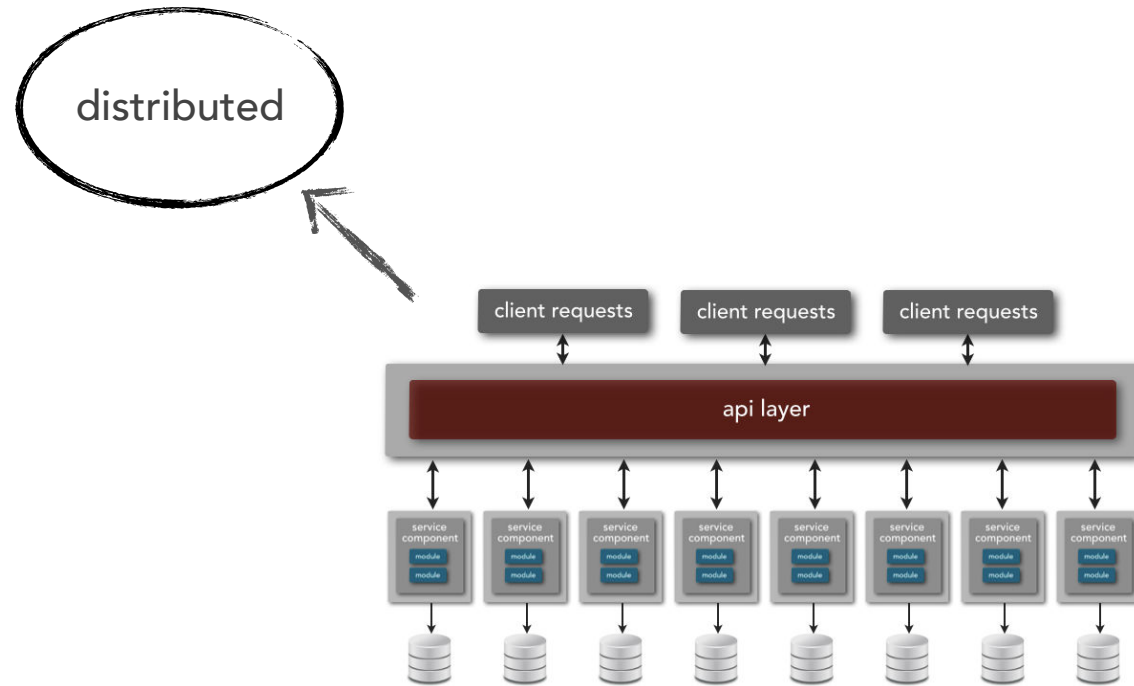


aGEnDA

Microservices Architecture

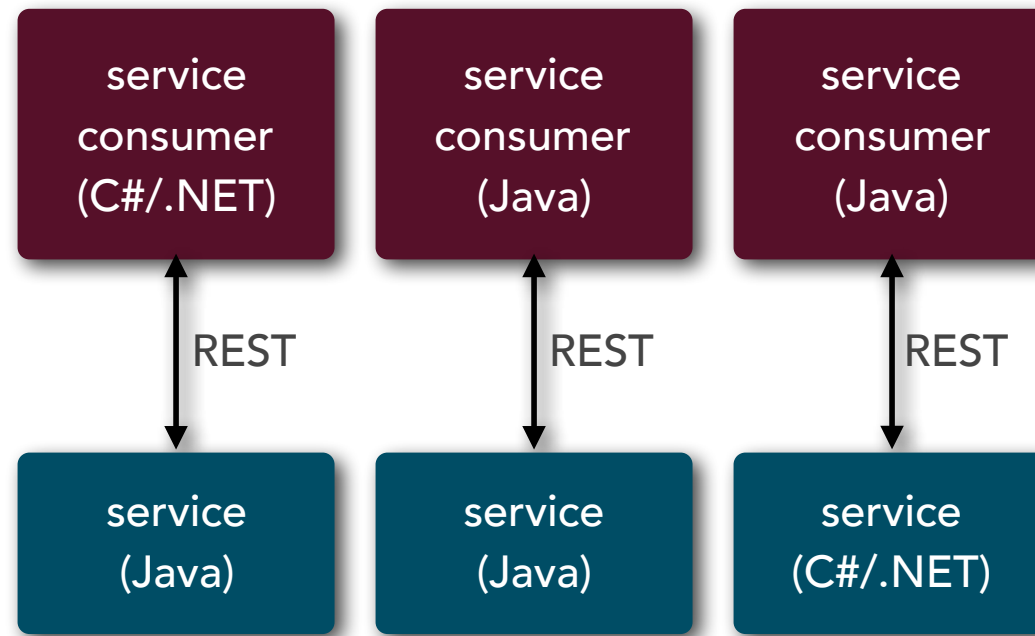


Microservices Architecture

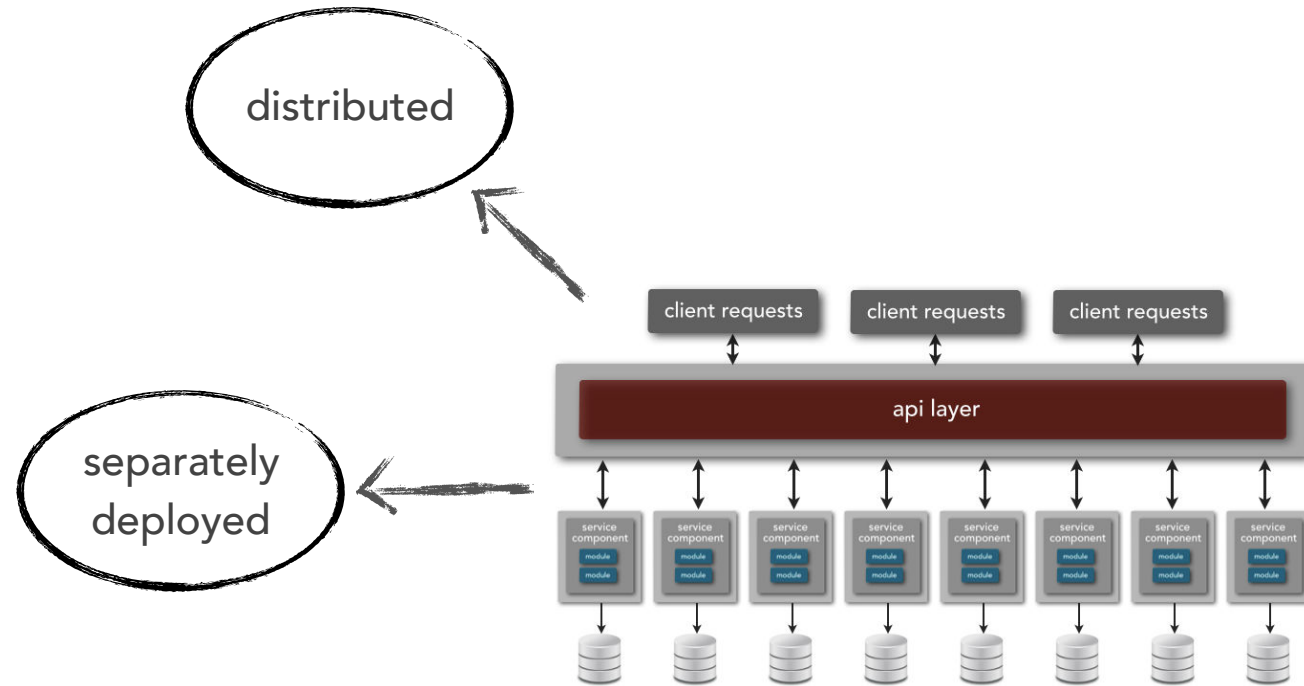


Microservices Architecture

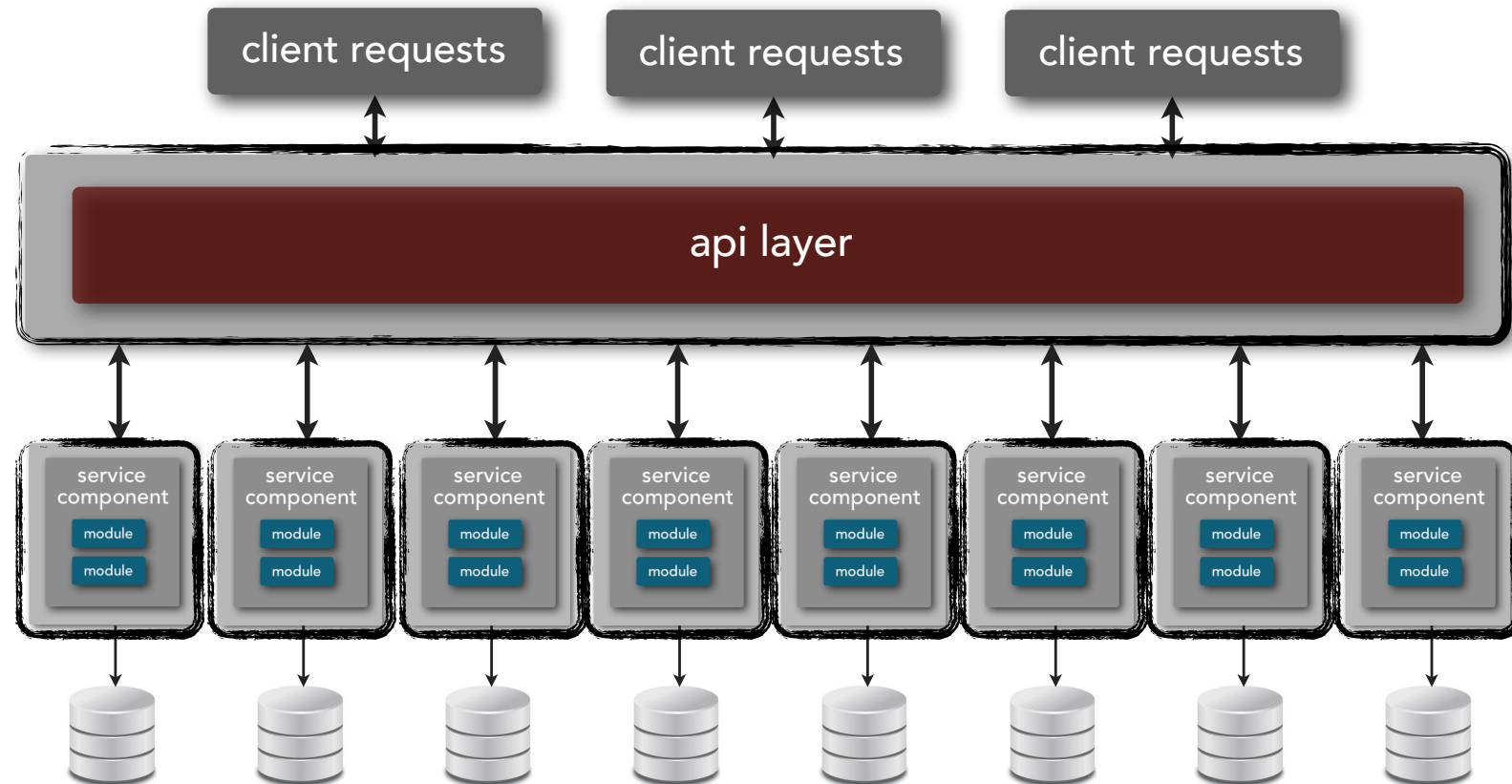
protocol-aware heterogeneous interoperability



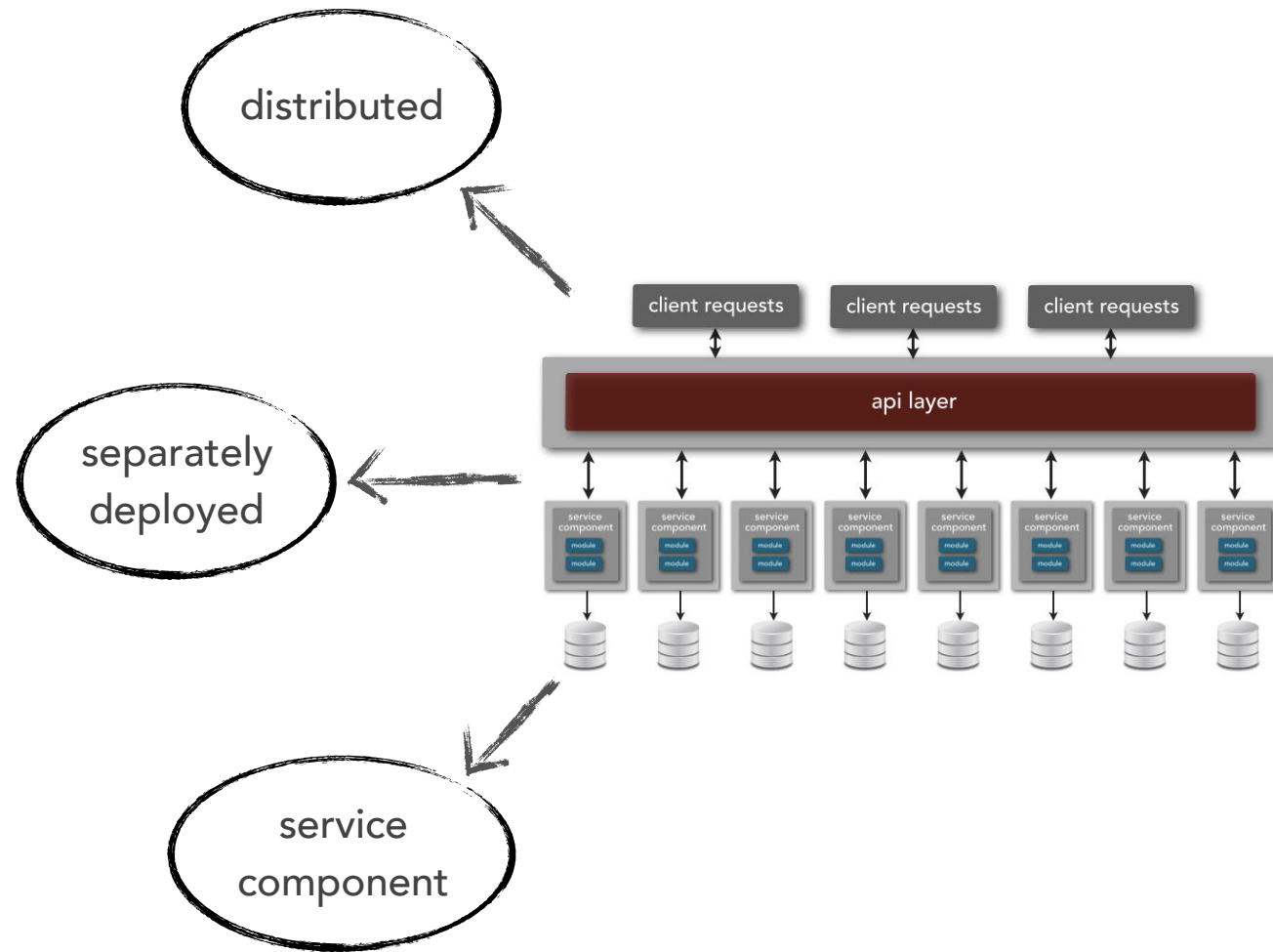
Microservices Architecture



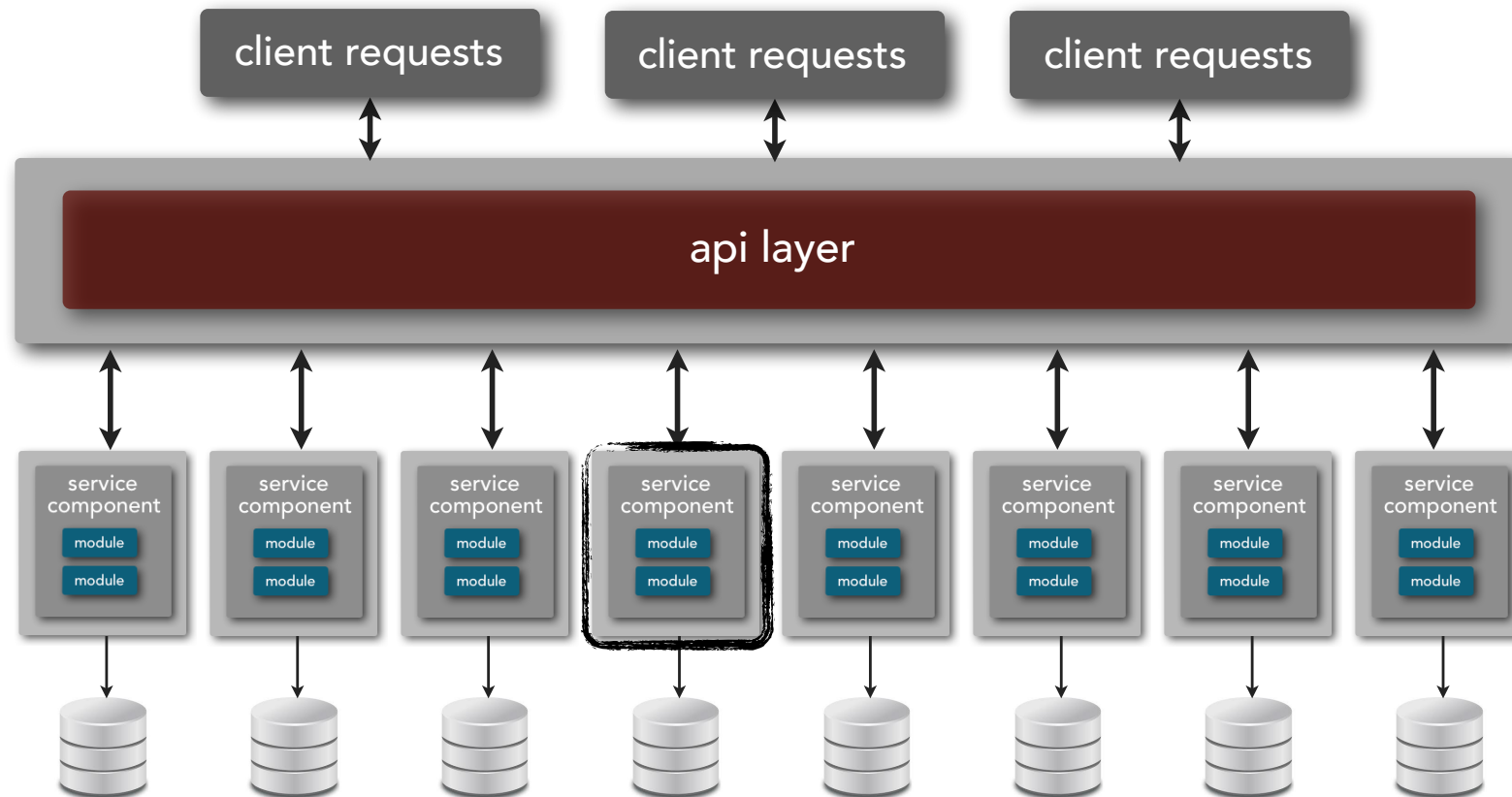
Microservices Architecture



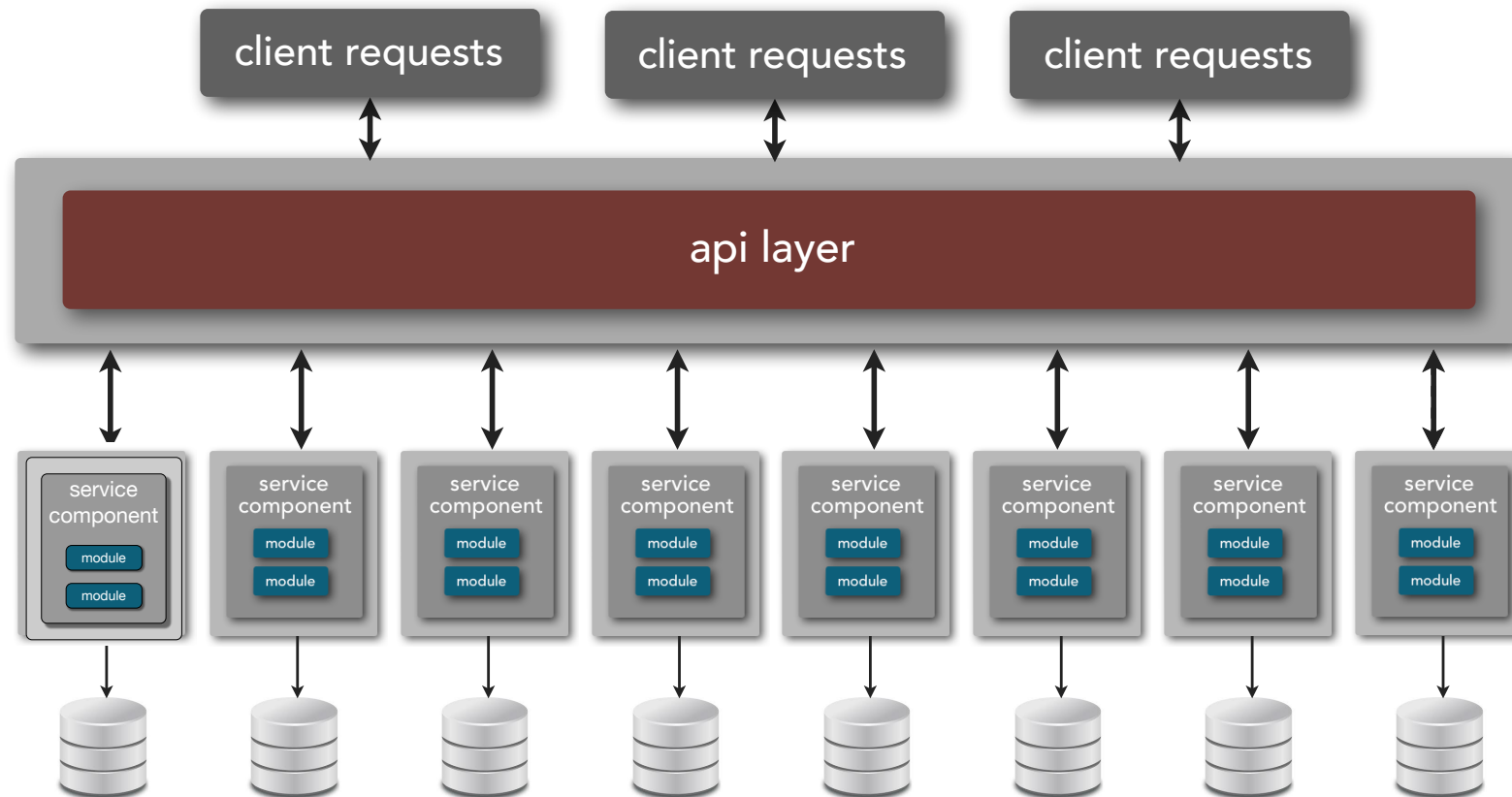
Microservices Architecture



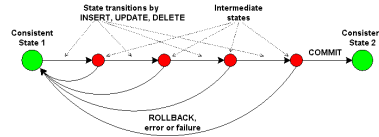
Microservices Architecture



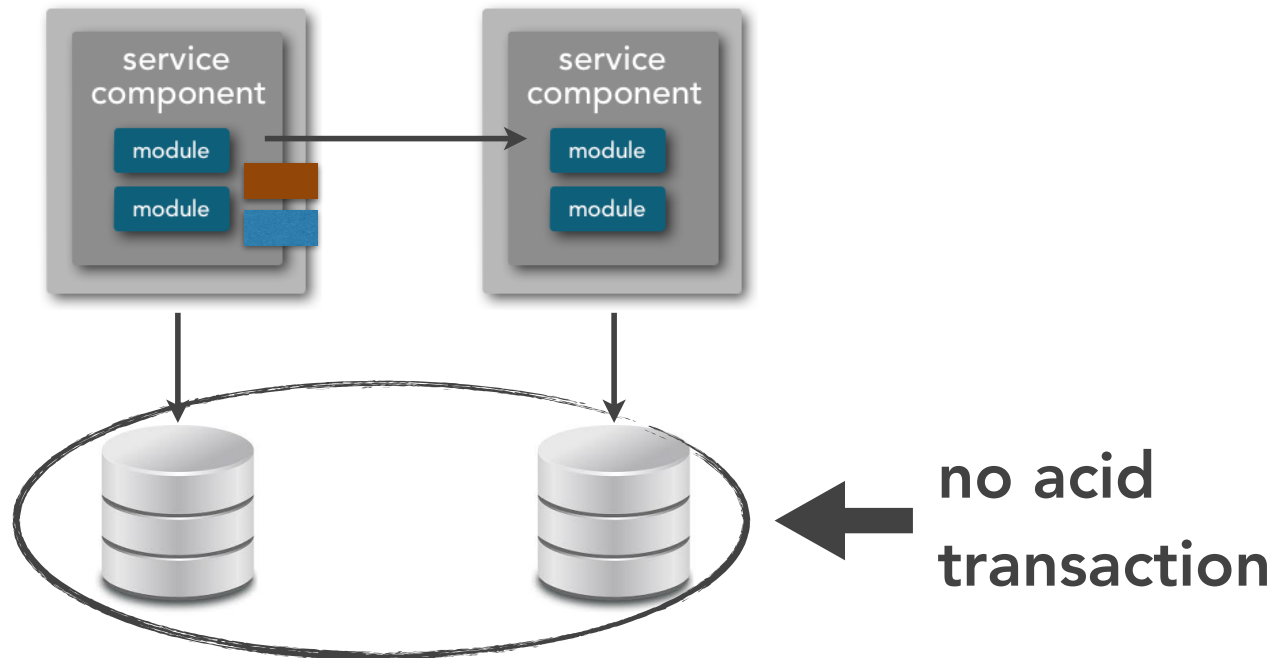
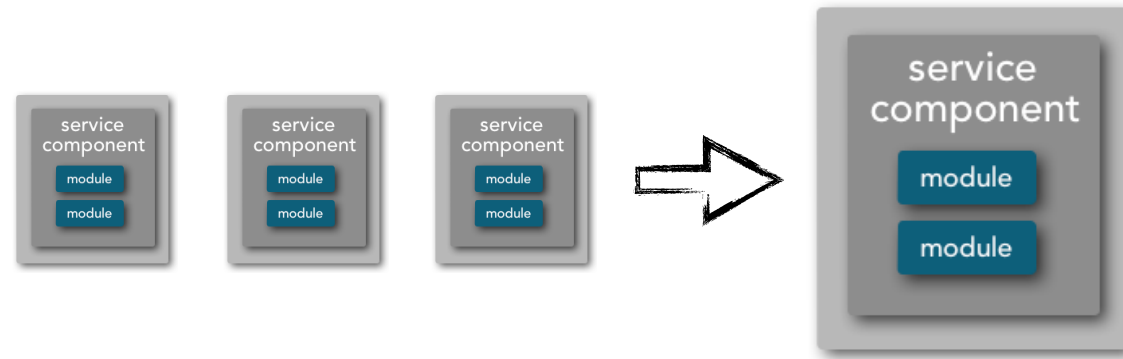
Microservices Architecture



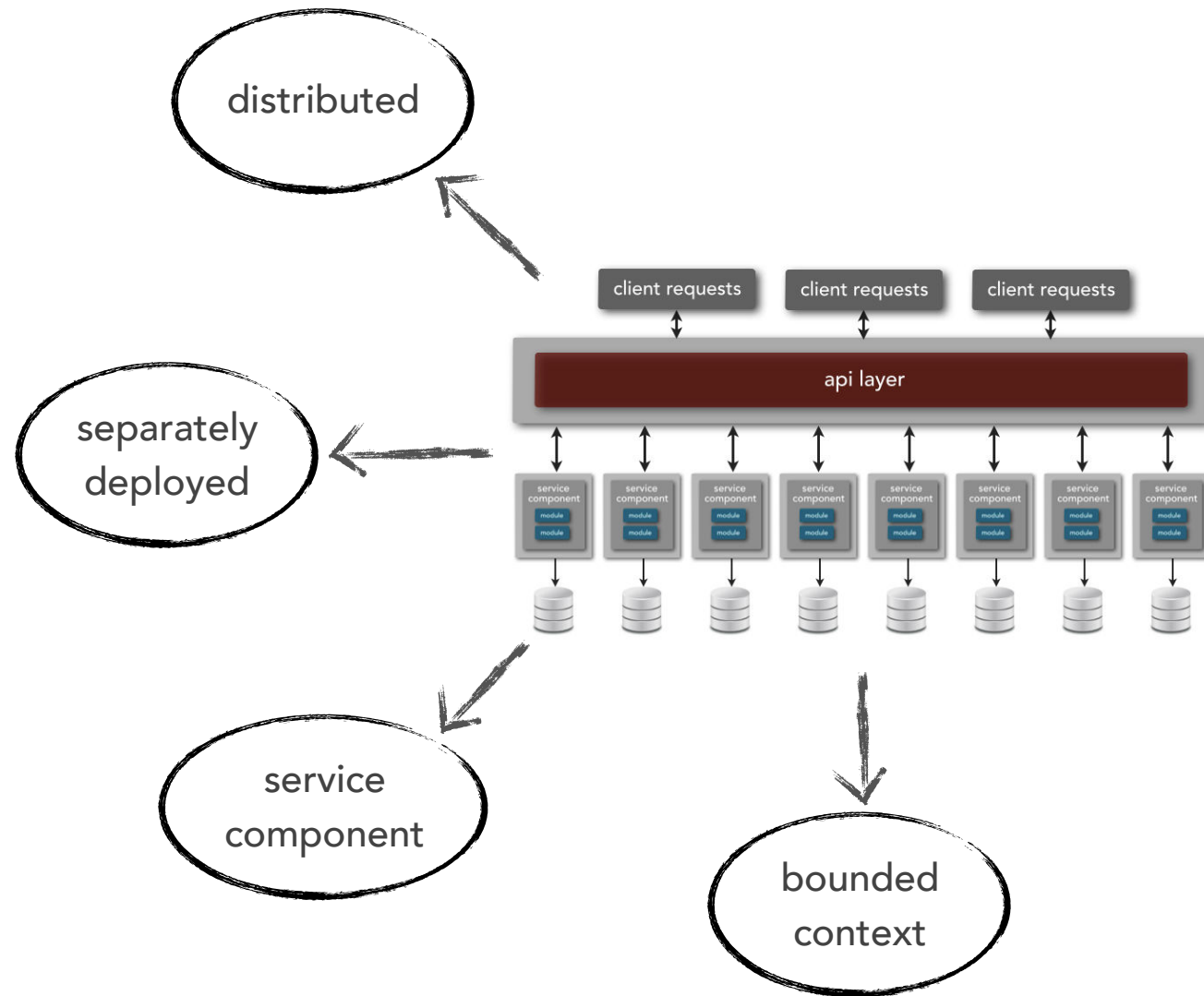
Microservices Architecture



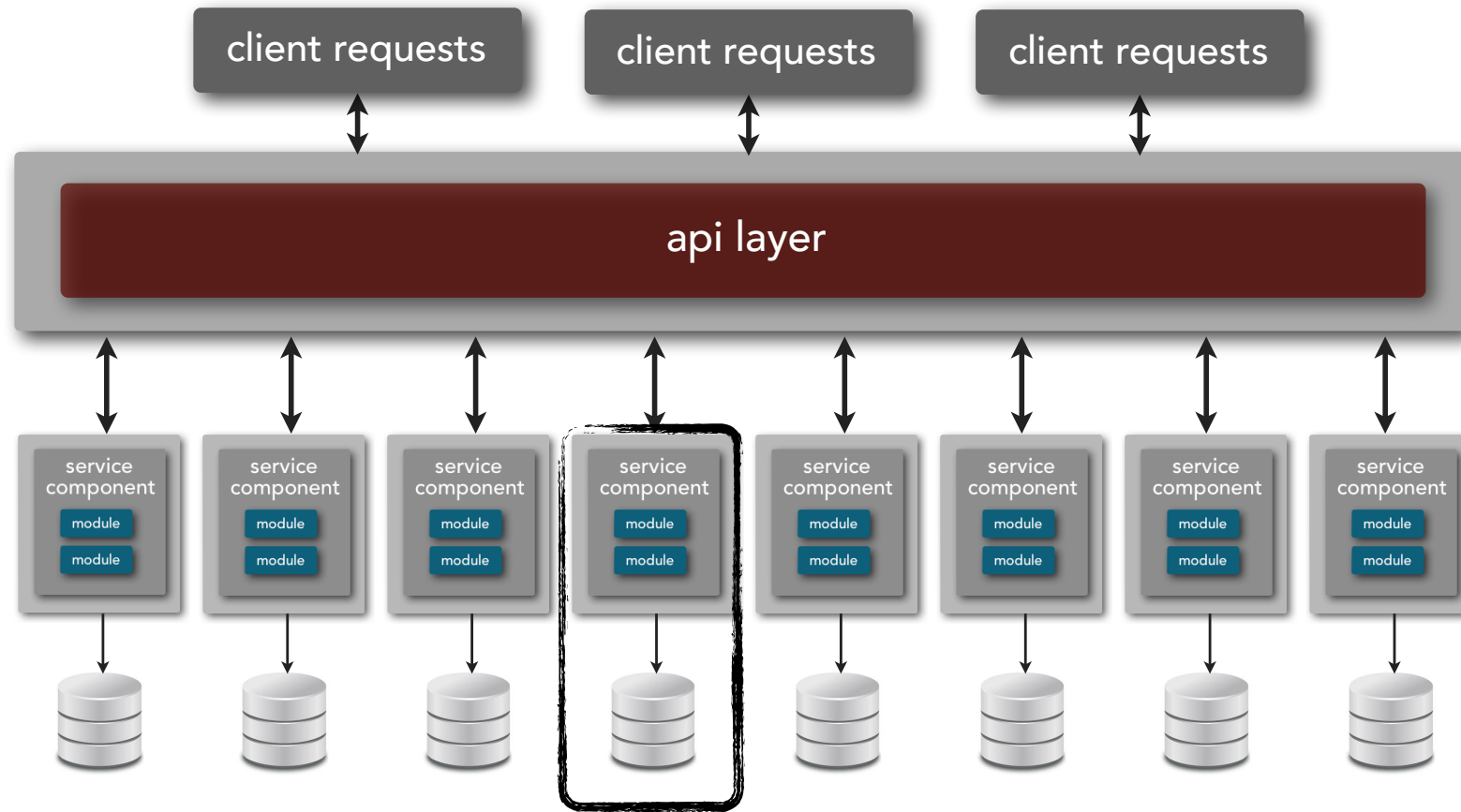
transactions



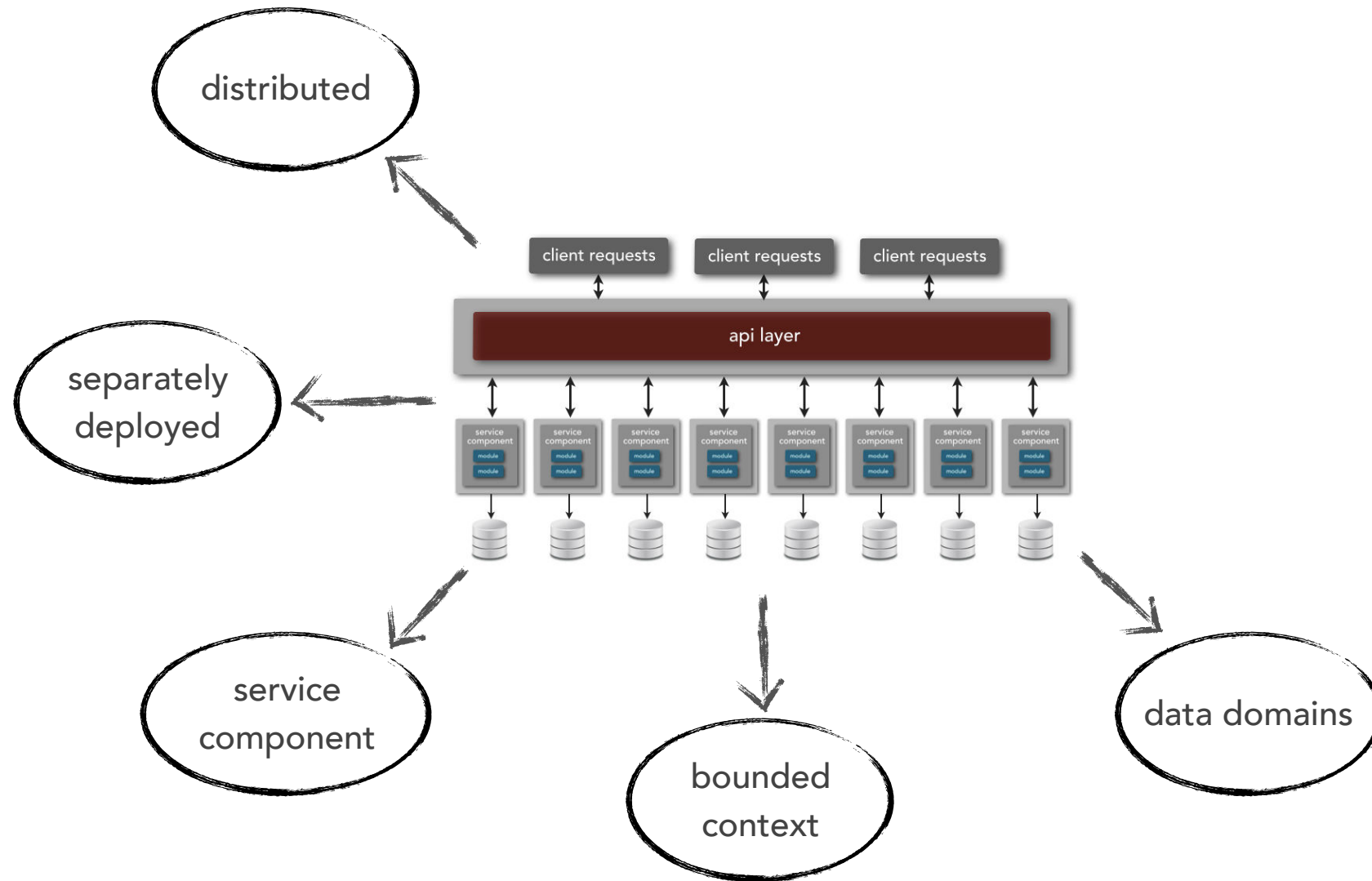
Microservices Architecture



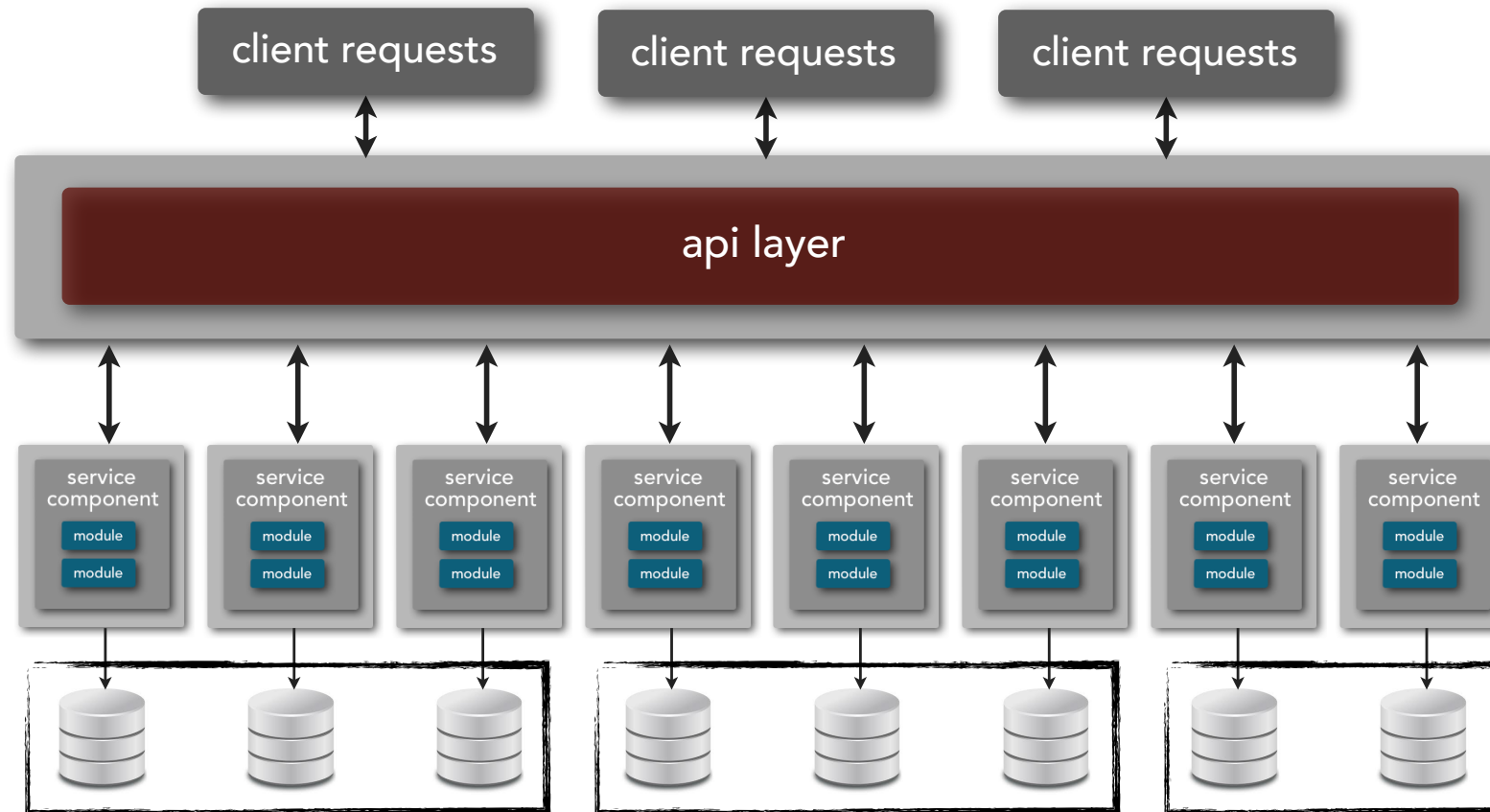
Microservices Architecture



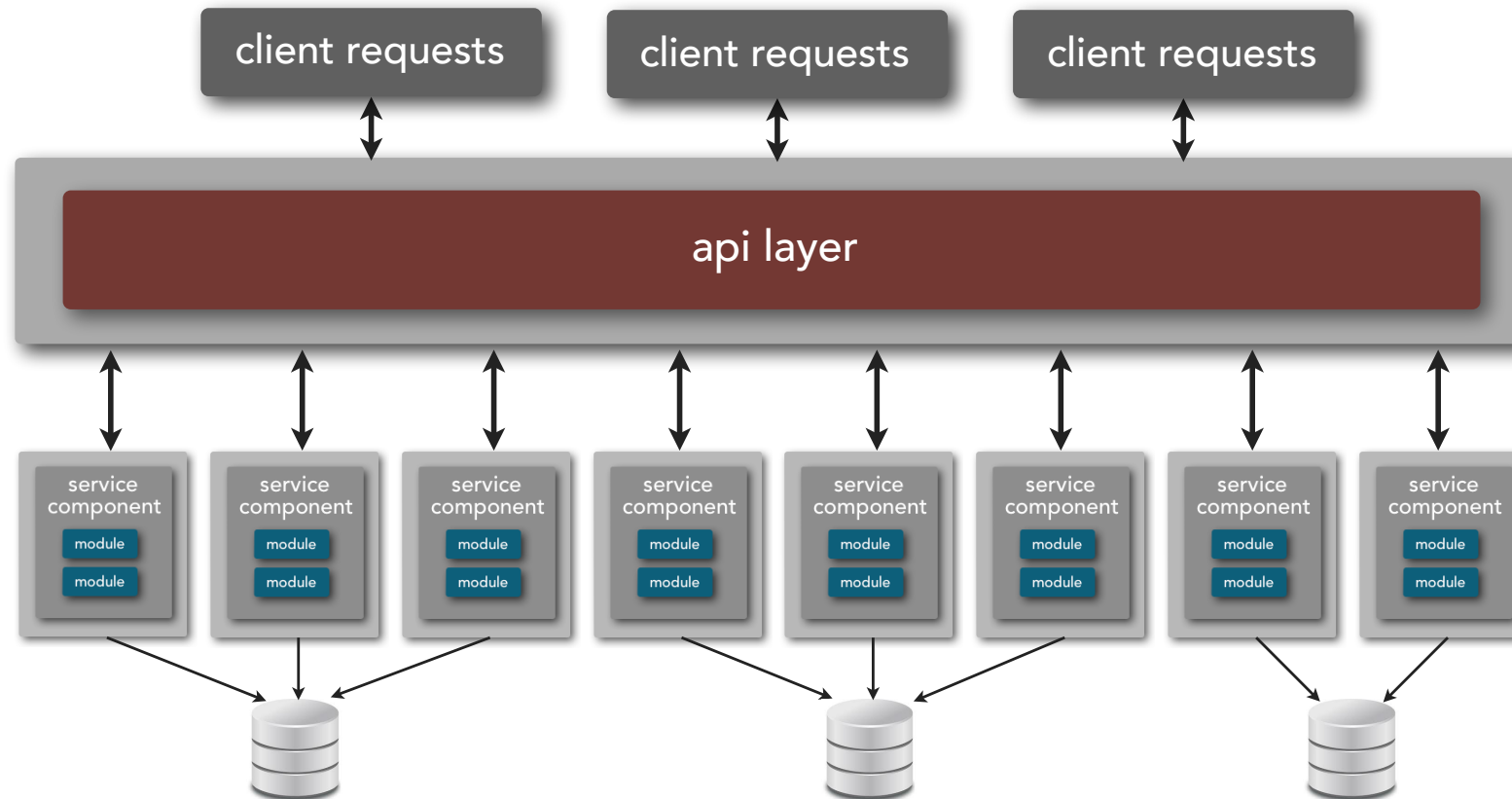
microservices architecture



Microservices Architecture

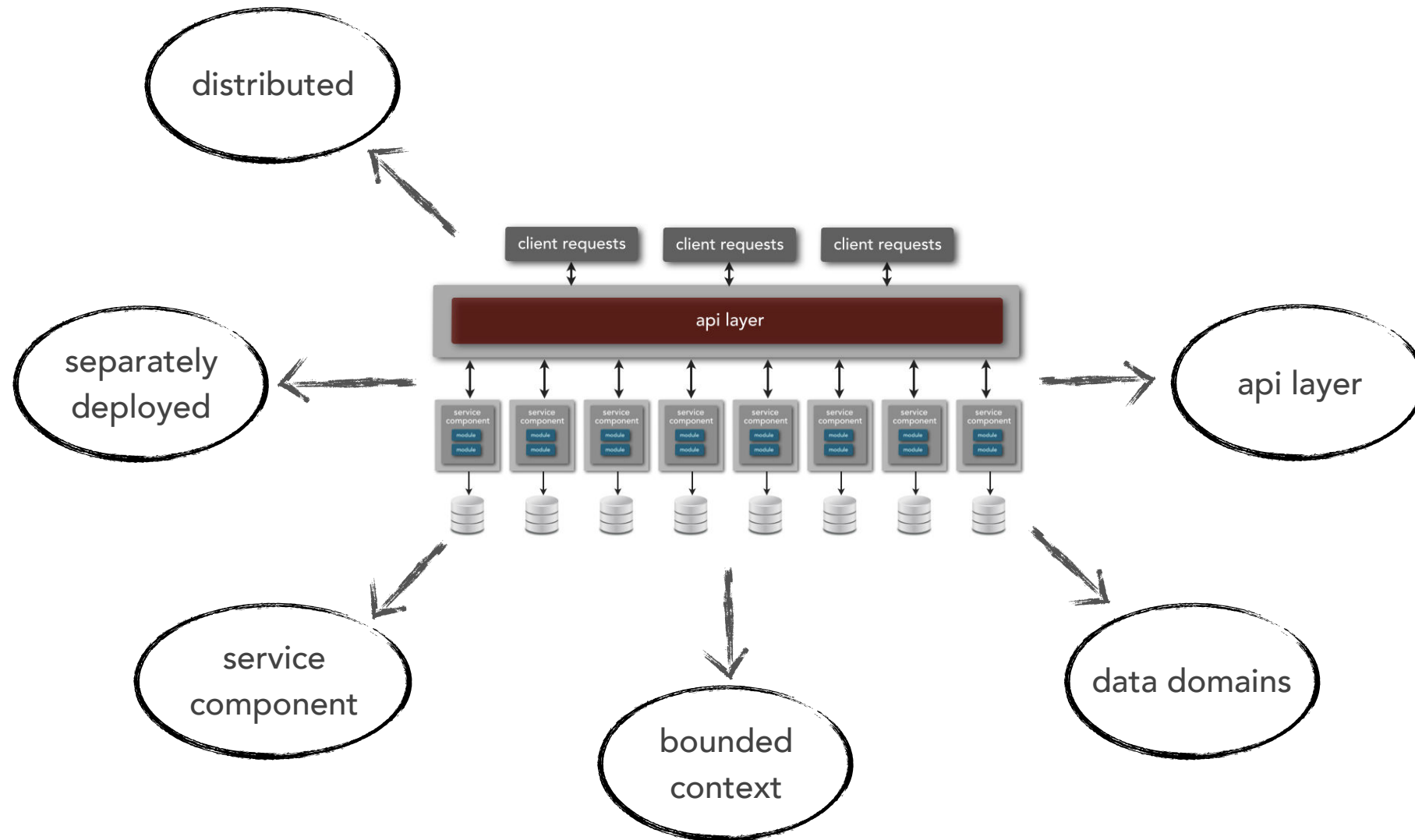


Microservices Architecture

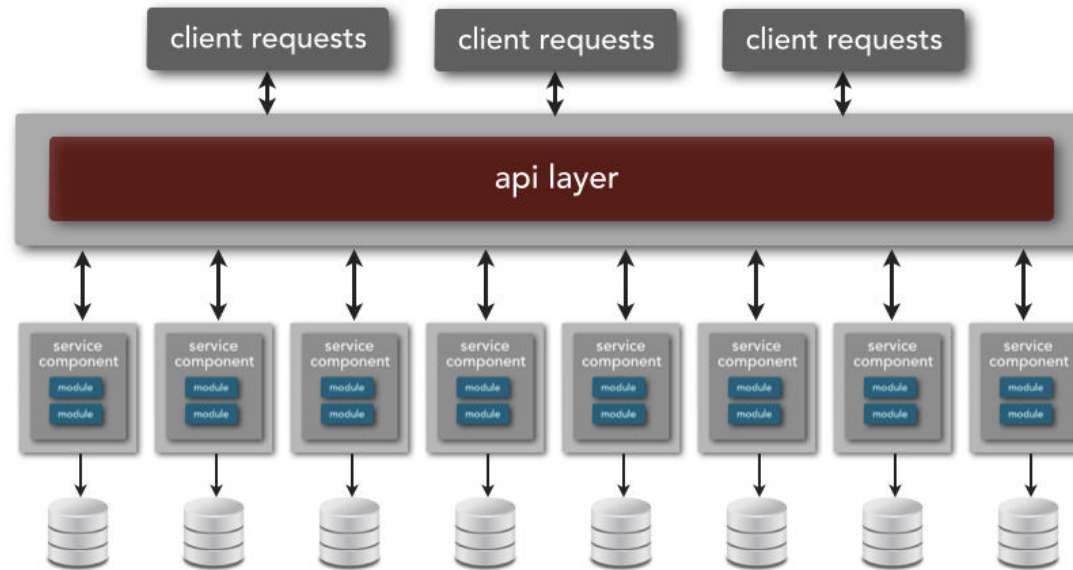


Hybrid

Microservices Architecture

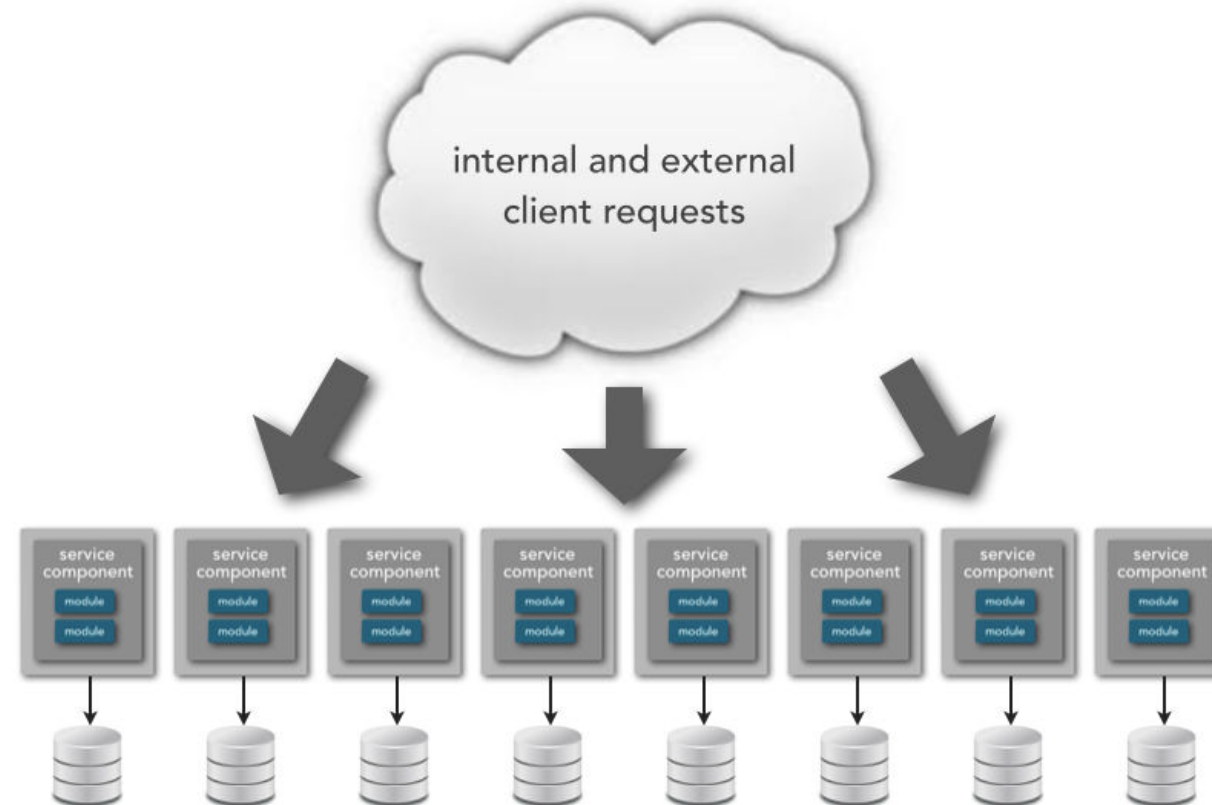


API Layer

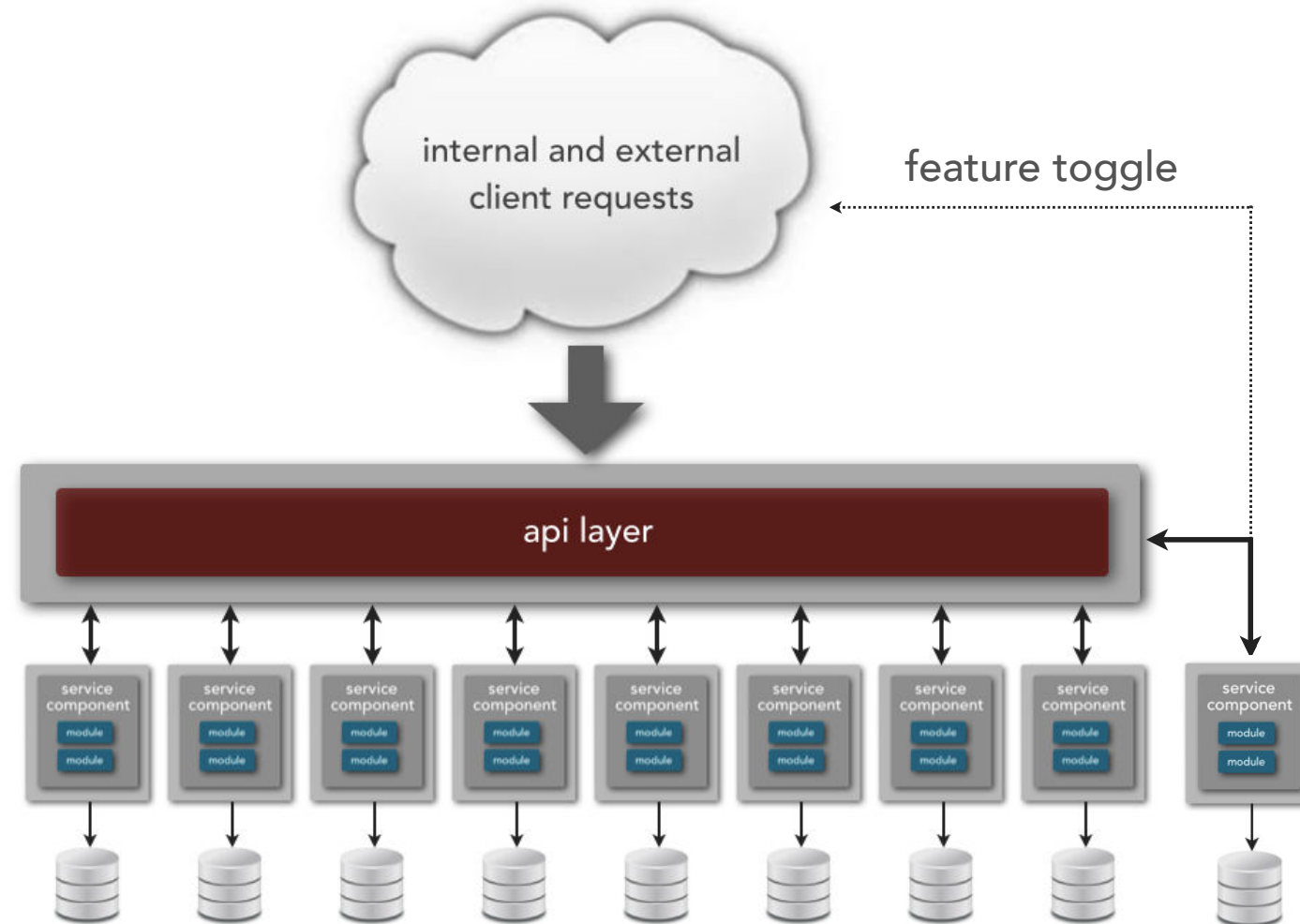


hides the actual endpoint of the service, exposing only those services available for public consumption

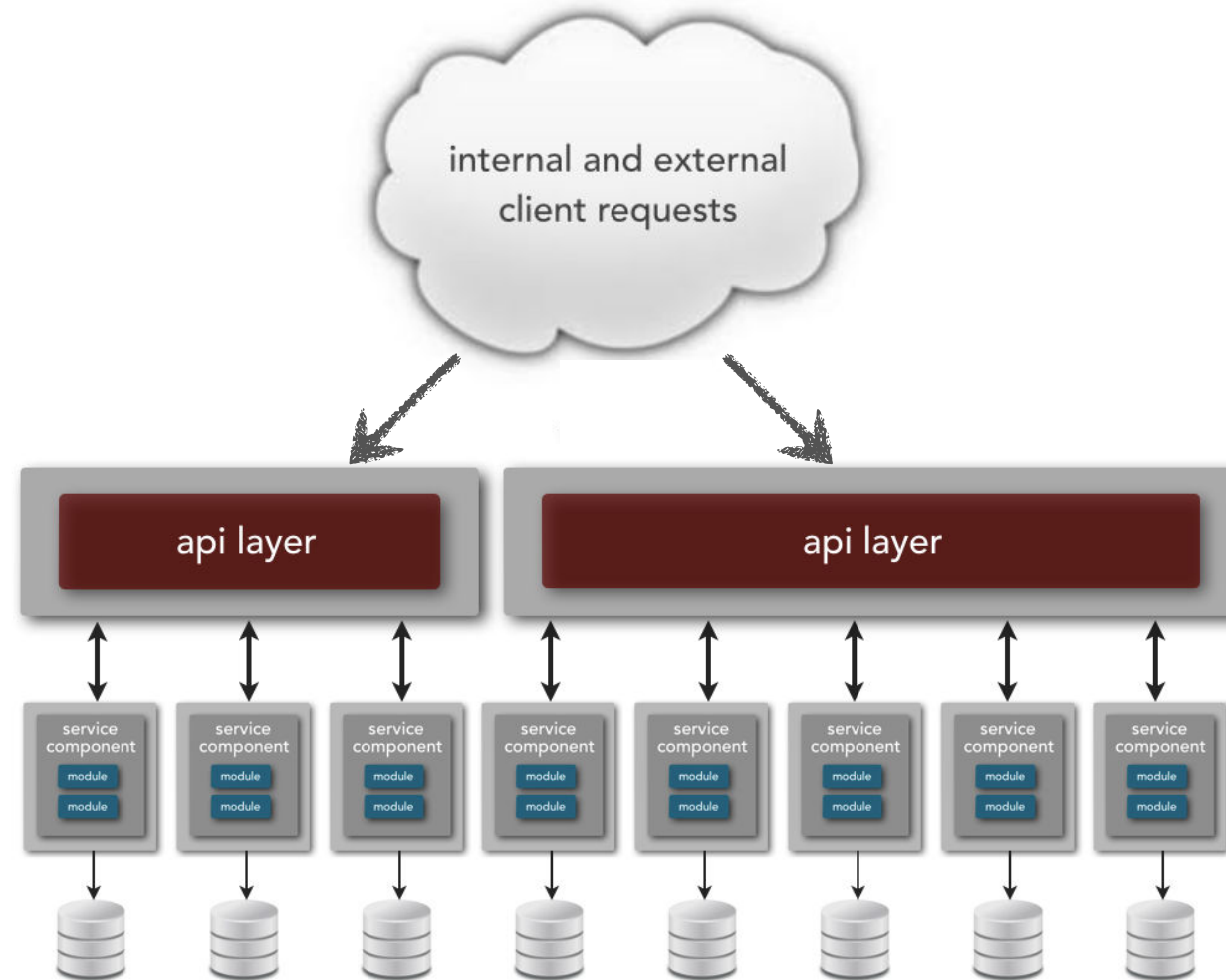
API Layer



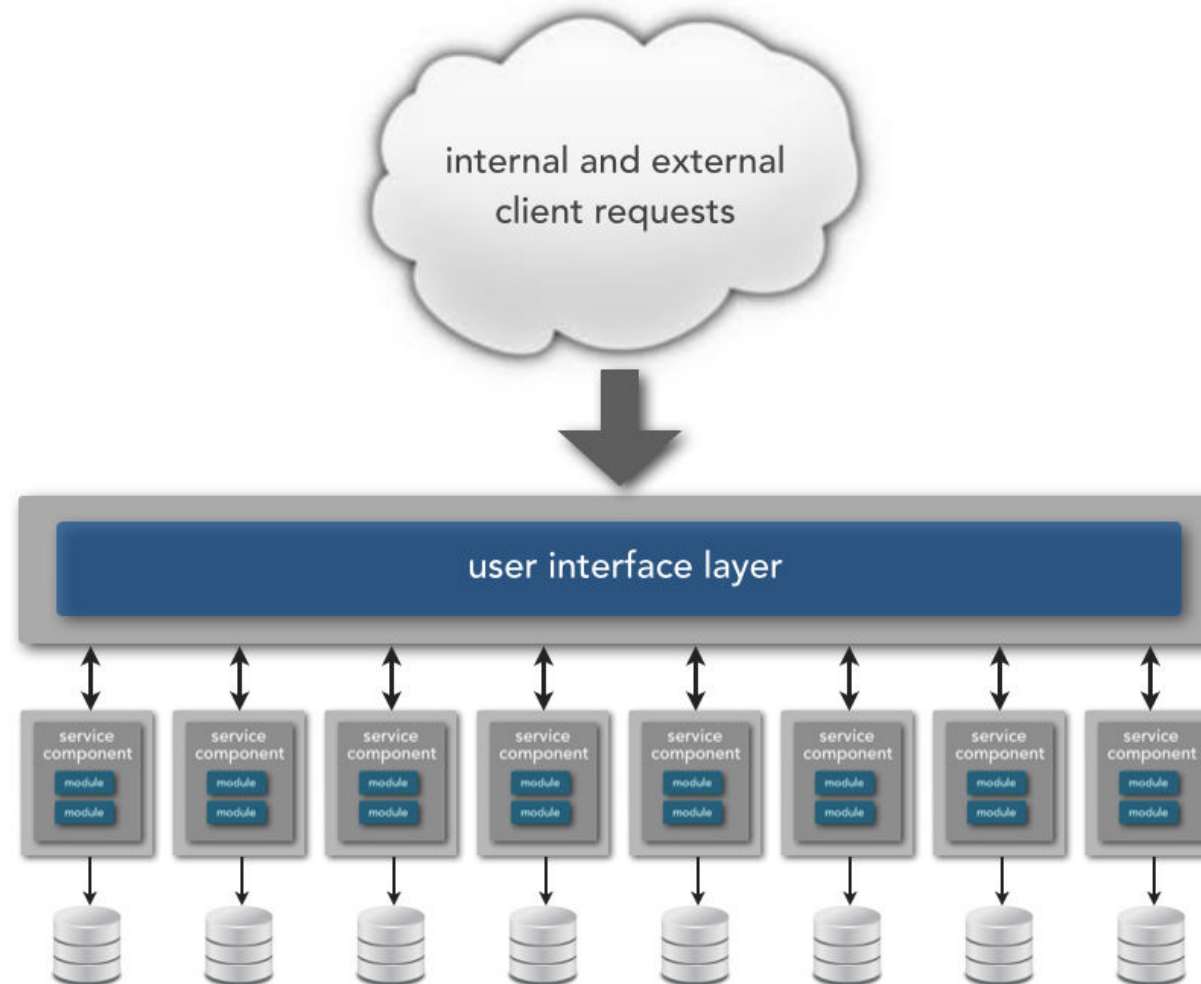
API Layer



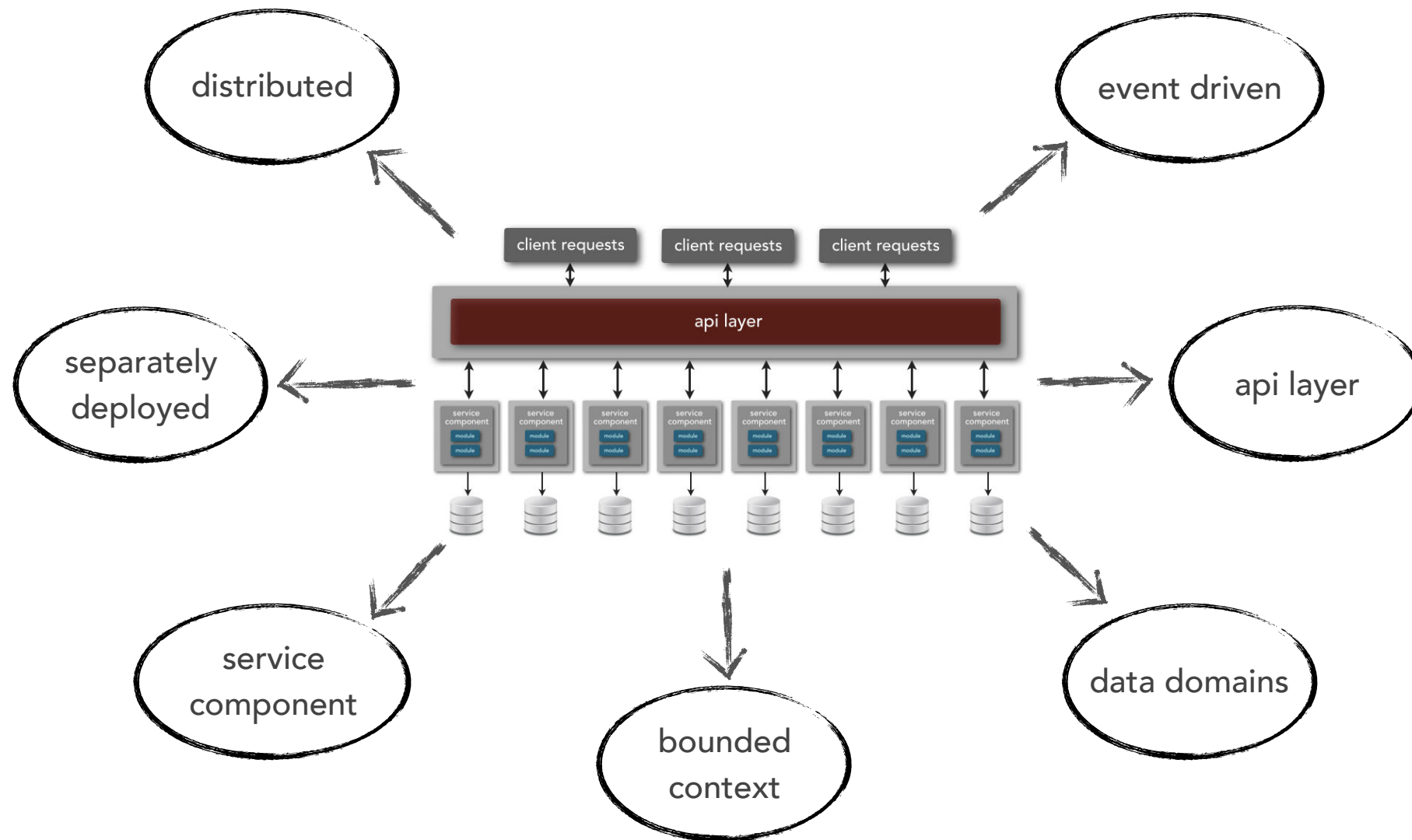
API Layer



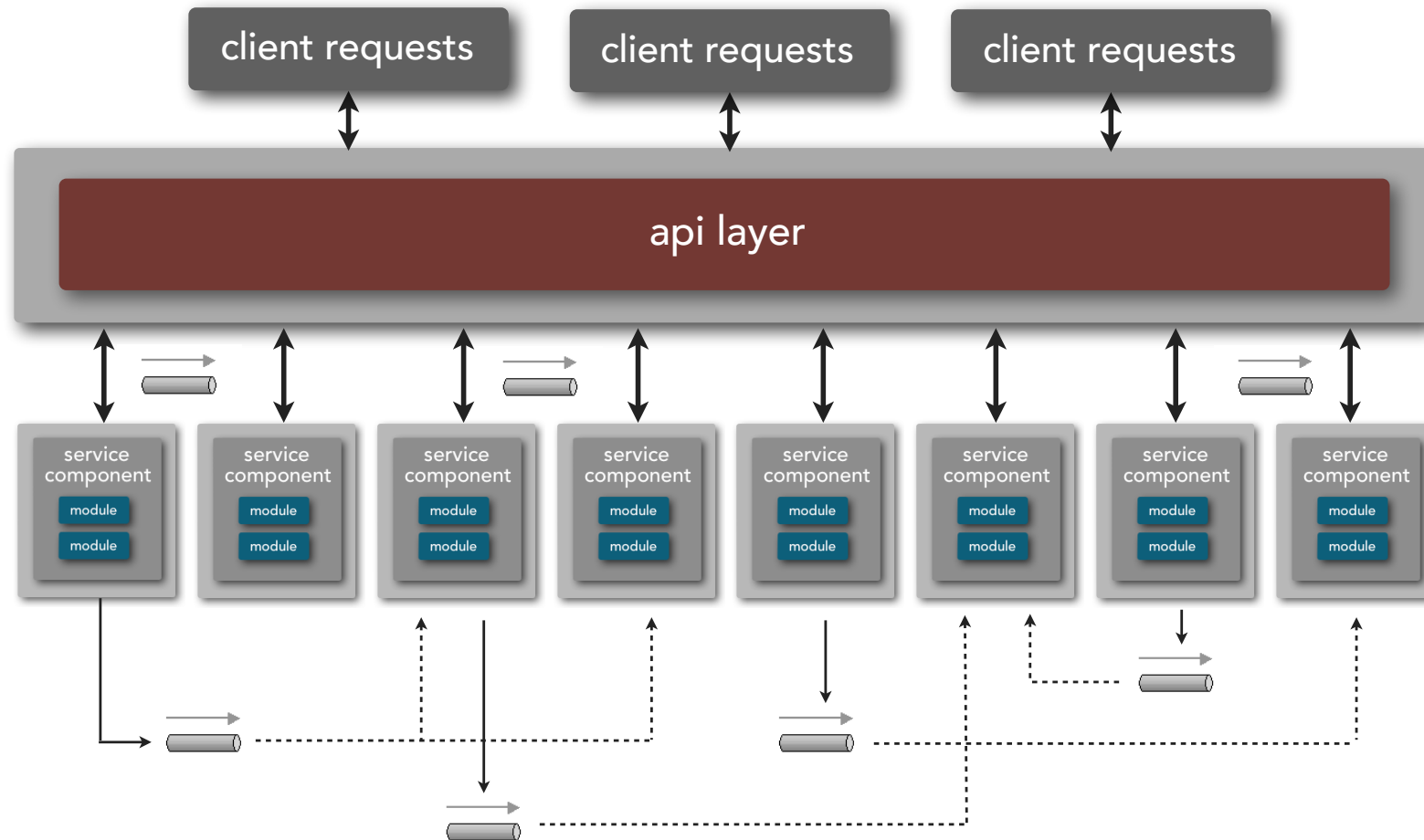
API Layer



microservices architecture

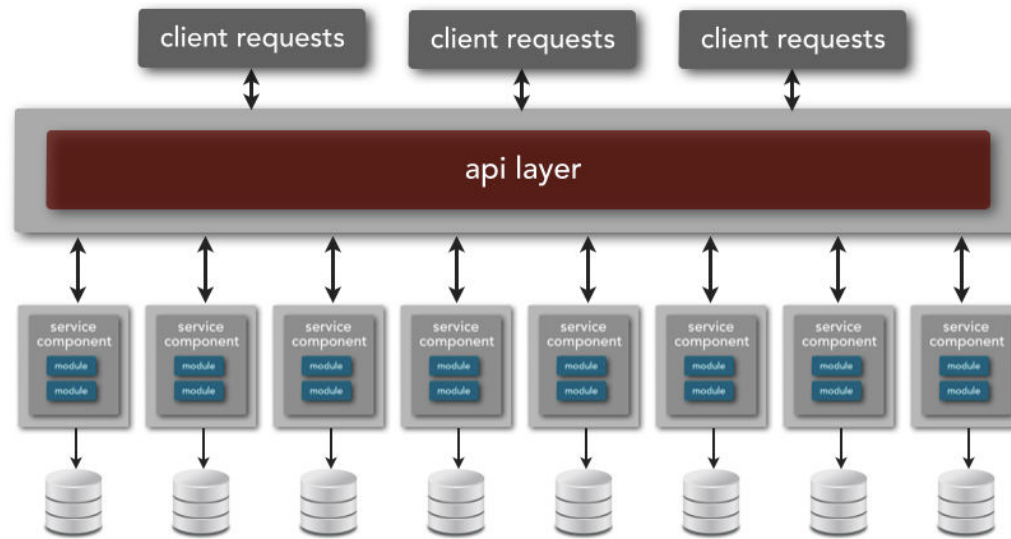


Microservices Architecture

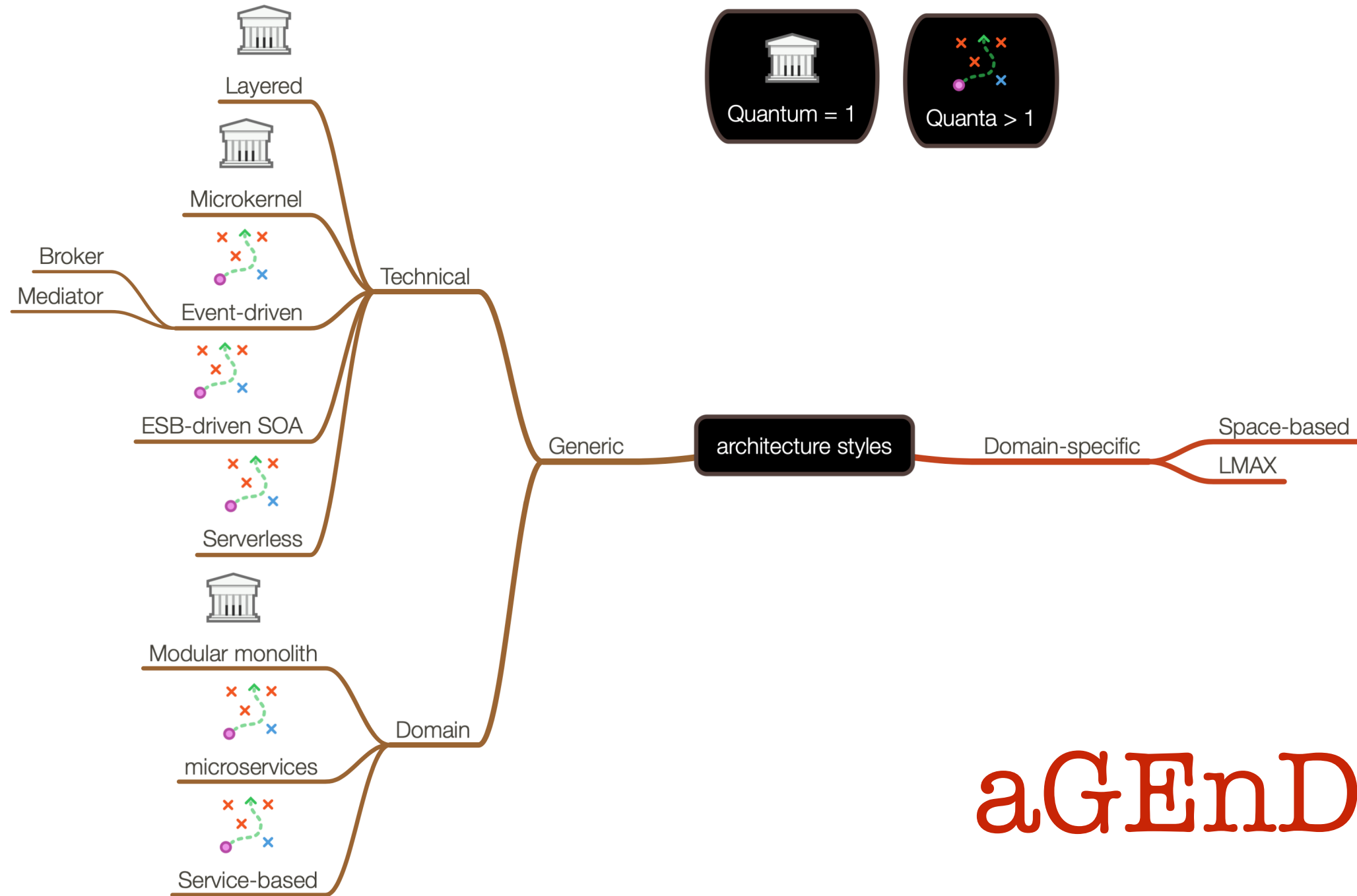


Microservices Architecture

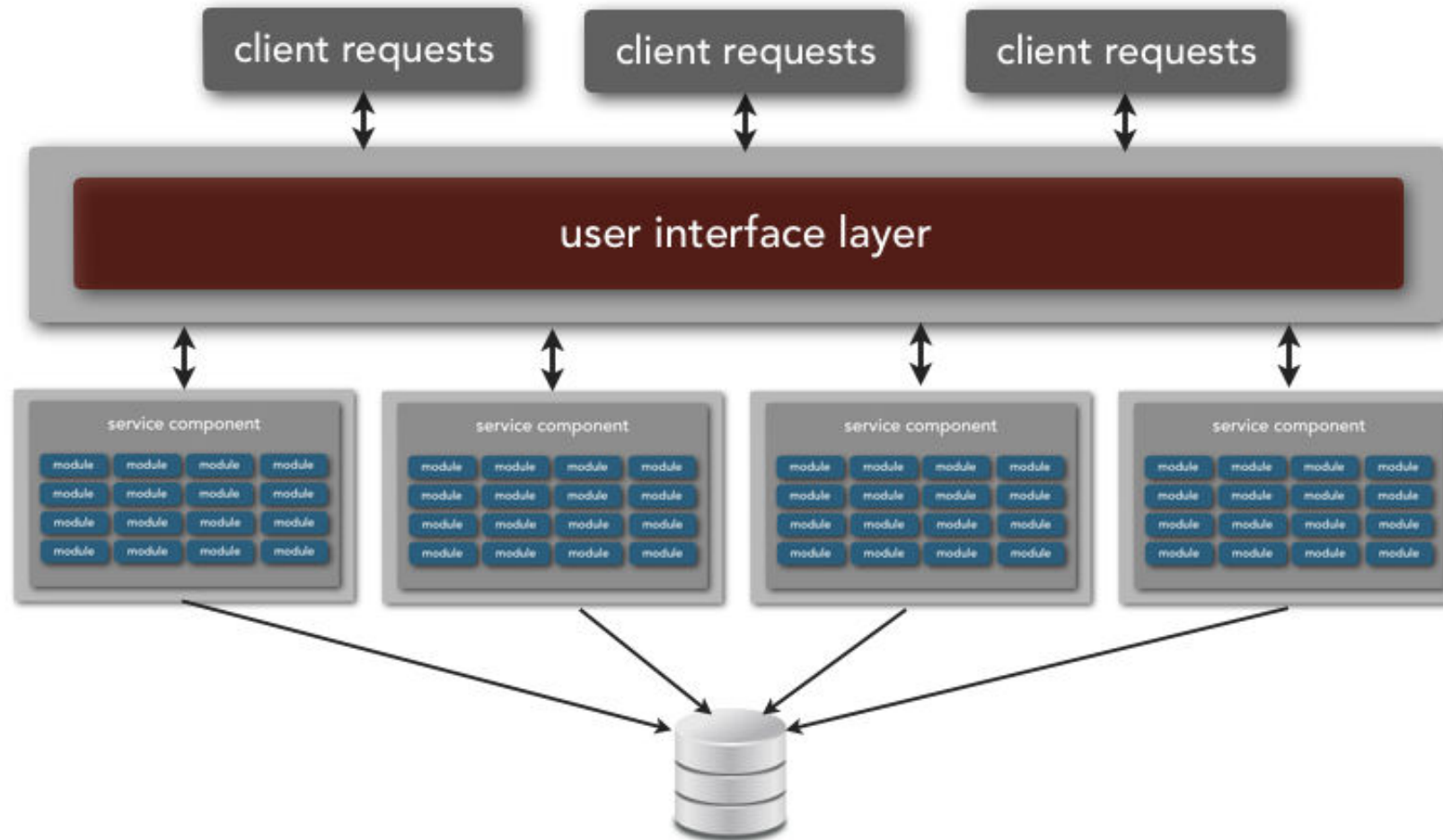
drivers



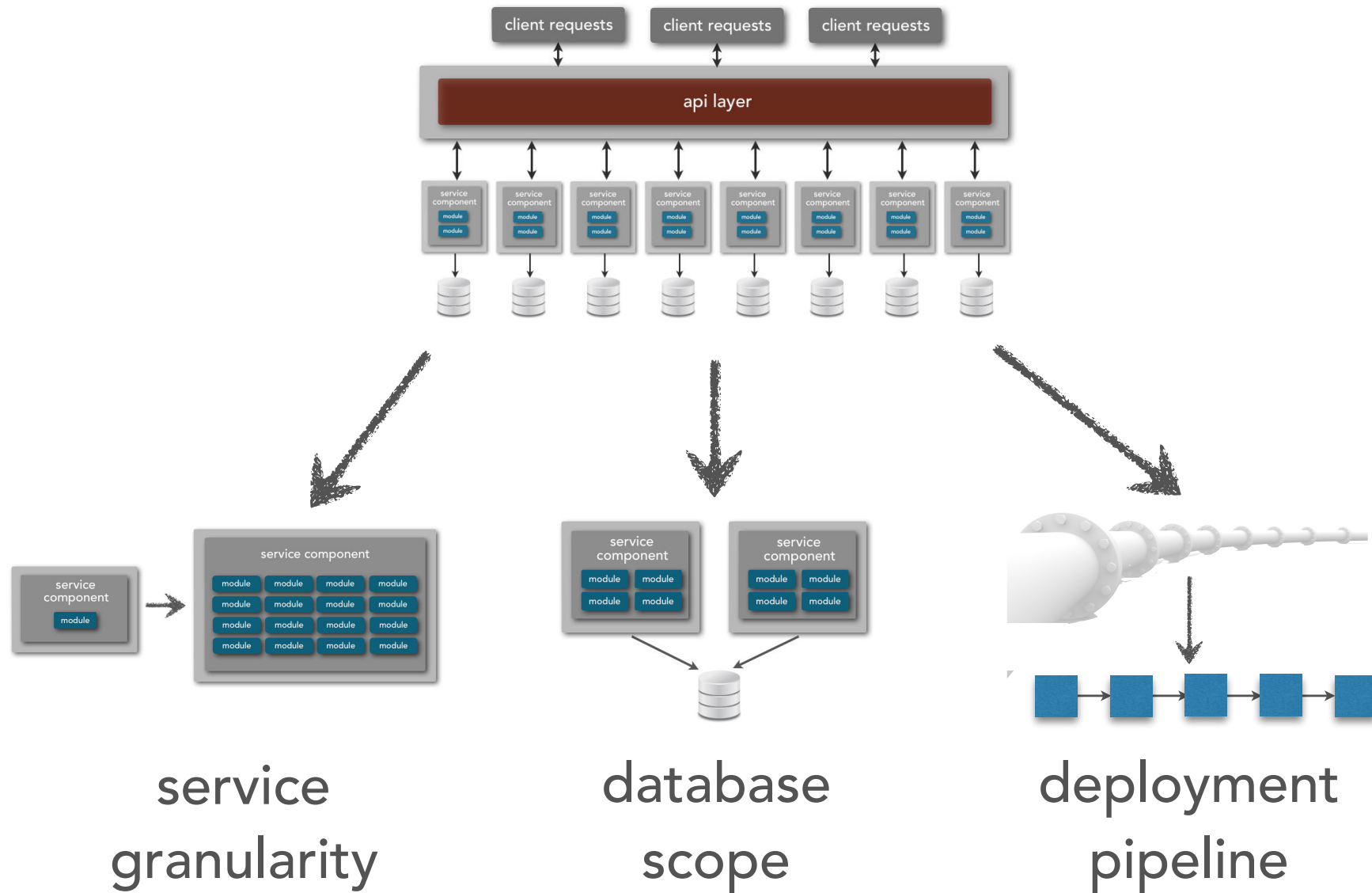
- ✓ modularity
- ✓ agility
- ✓ fault-tolerance
- ✓ scalability
- ✓ testability
- ✓ deployability
- ✓ evolvability



Service-based Architecture

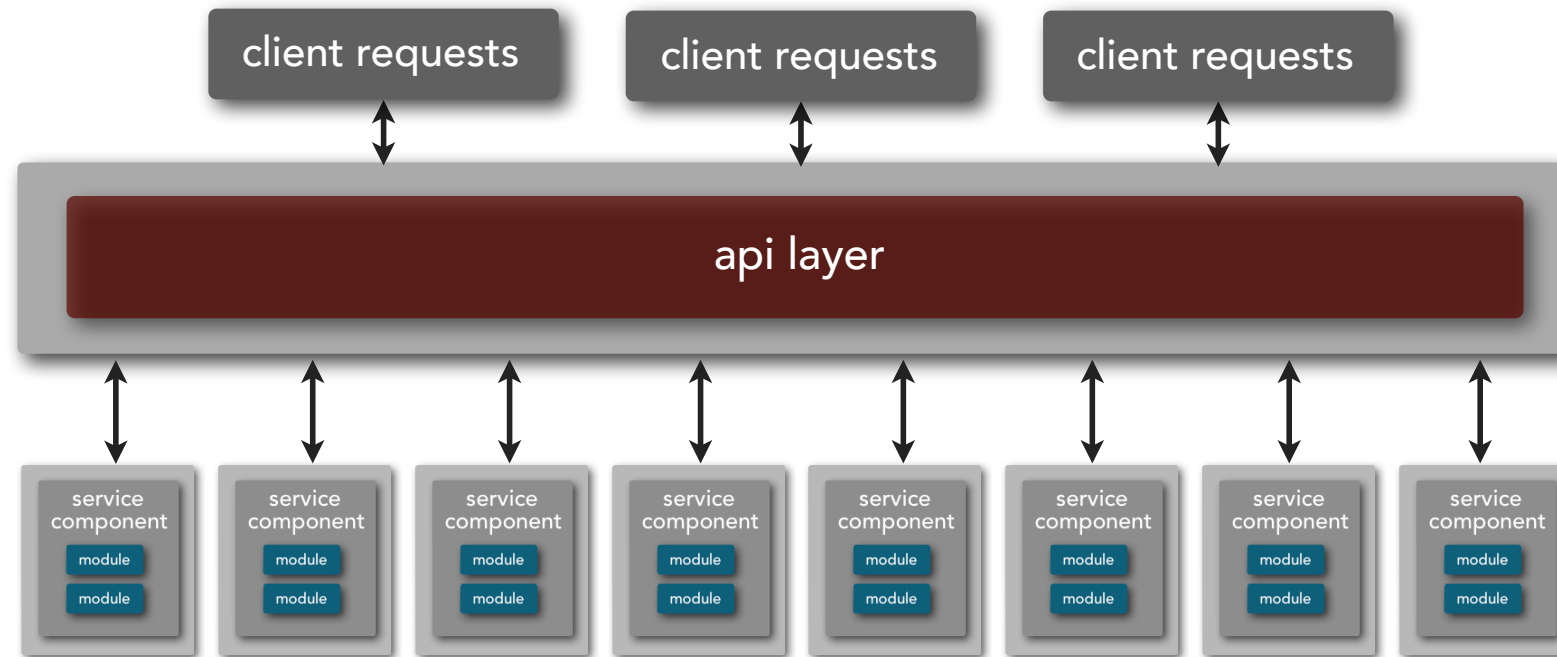


Service-based Architecture



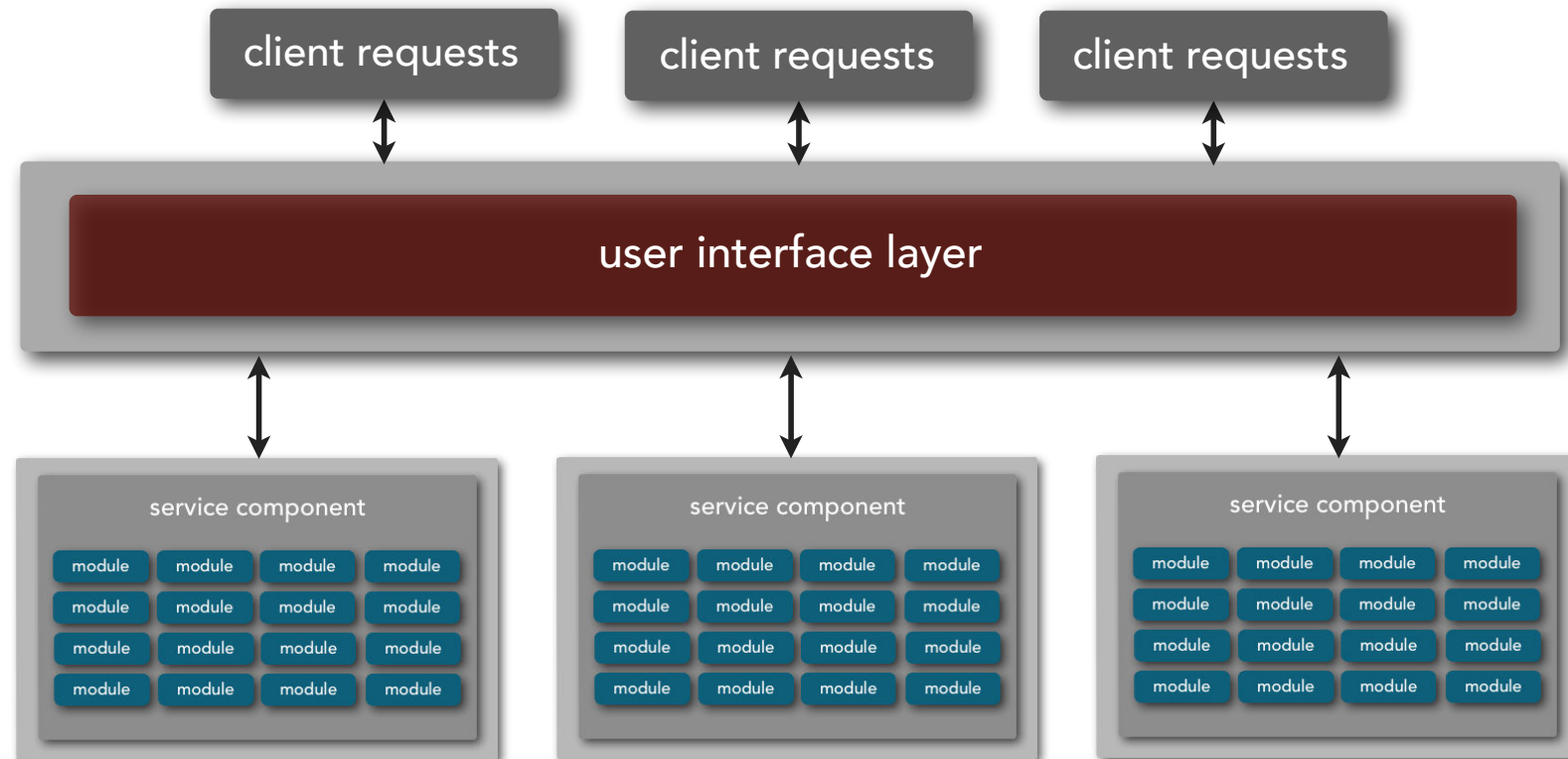
Service-based Architecture

service granularity



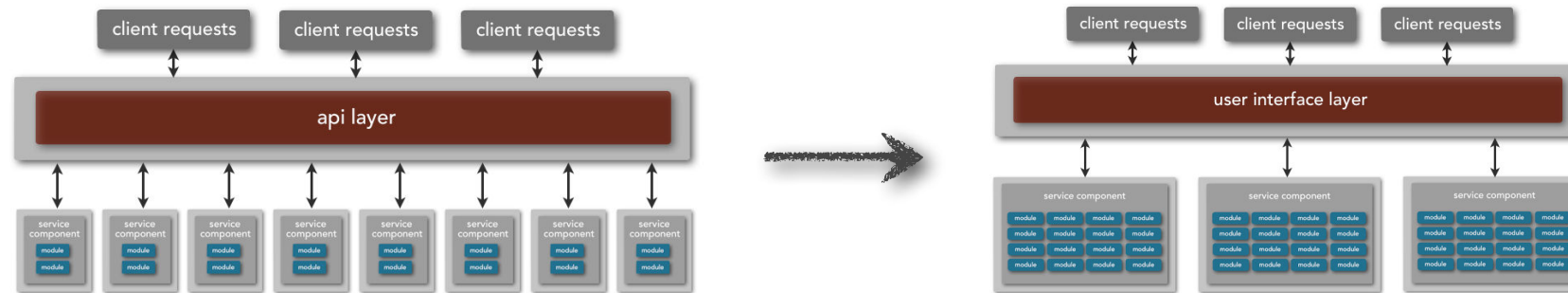
Service-based Architecture

service granularity



Service-based Architecture

service granularity



tradeoffs



performance



robustness



domain scope



service development



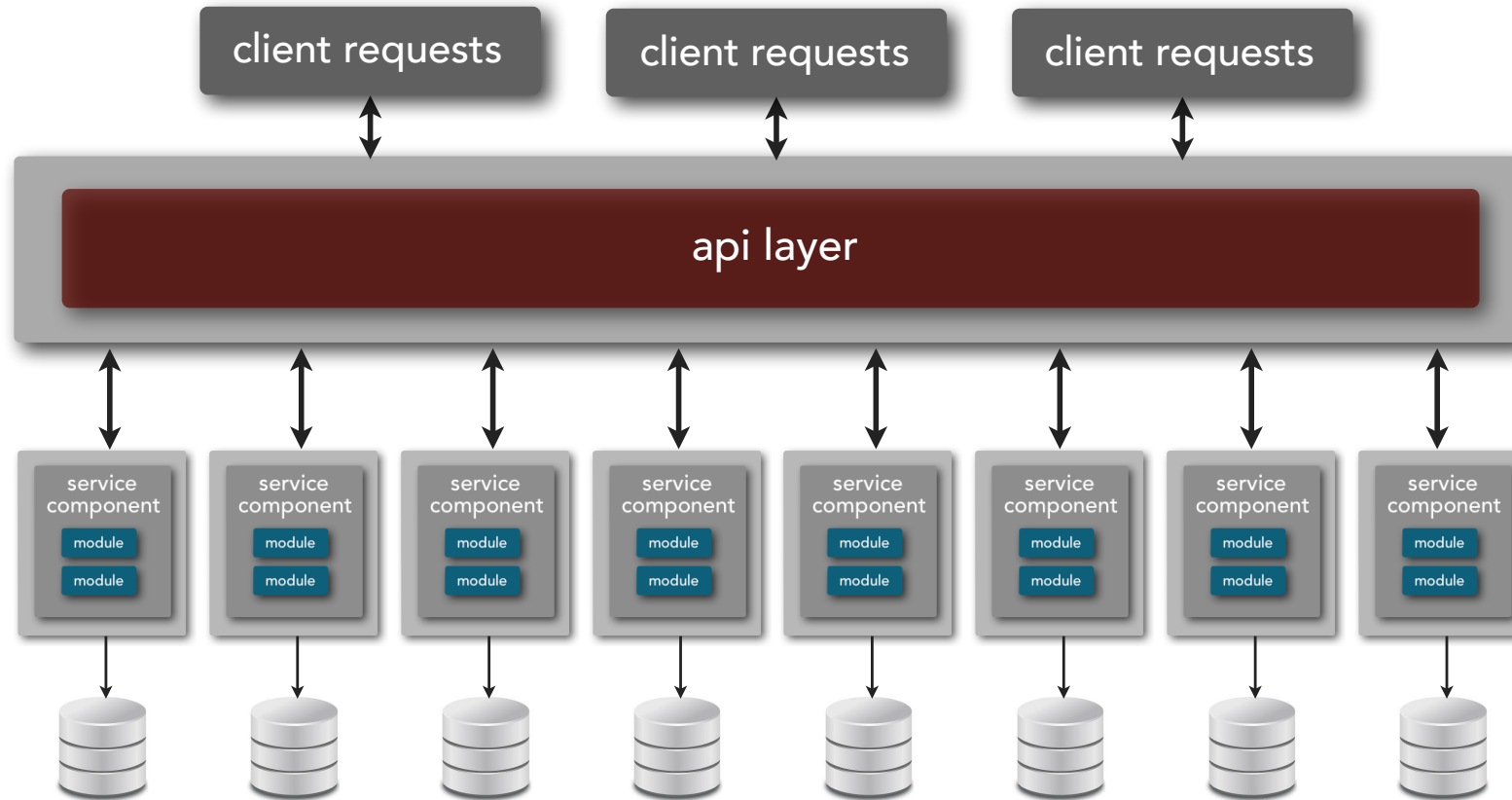
service testing



deployment pipeline

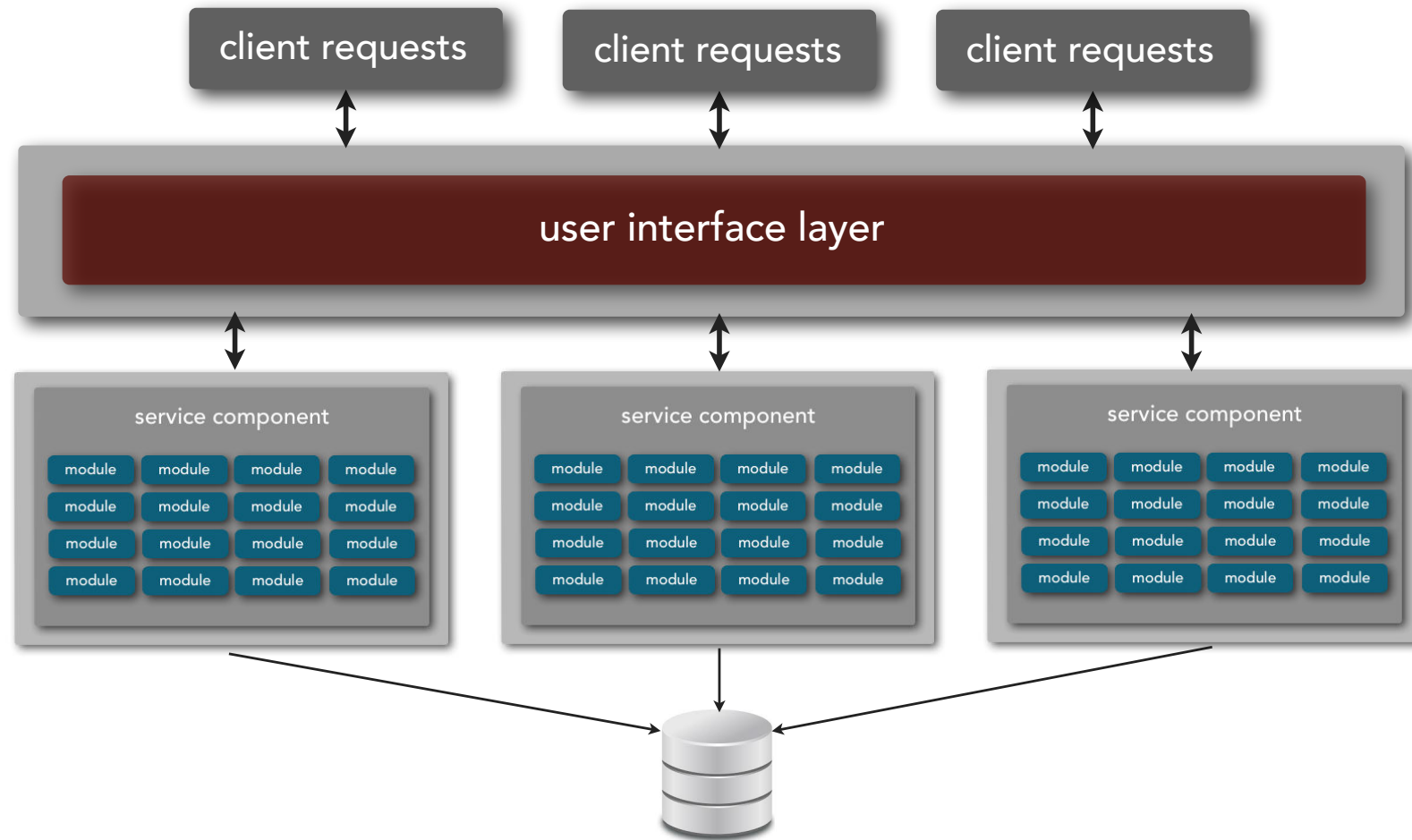
Service-based Architecture

database scope



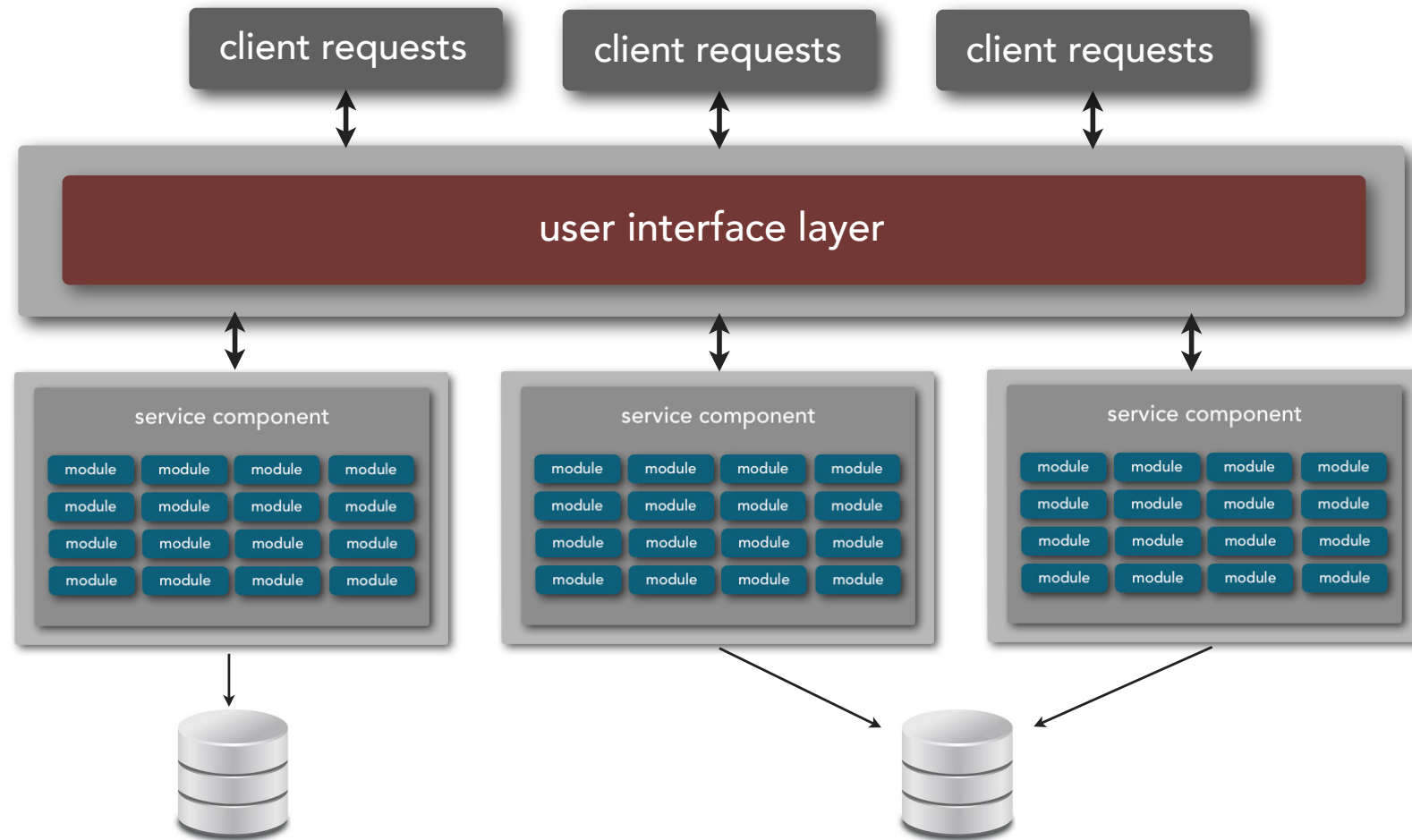
Service-based Architecture

database scope



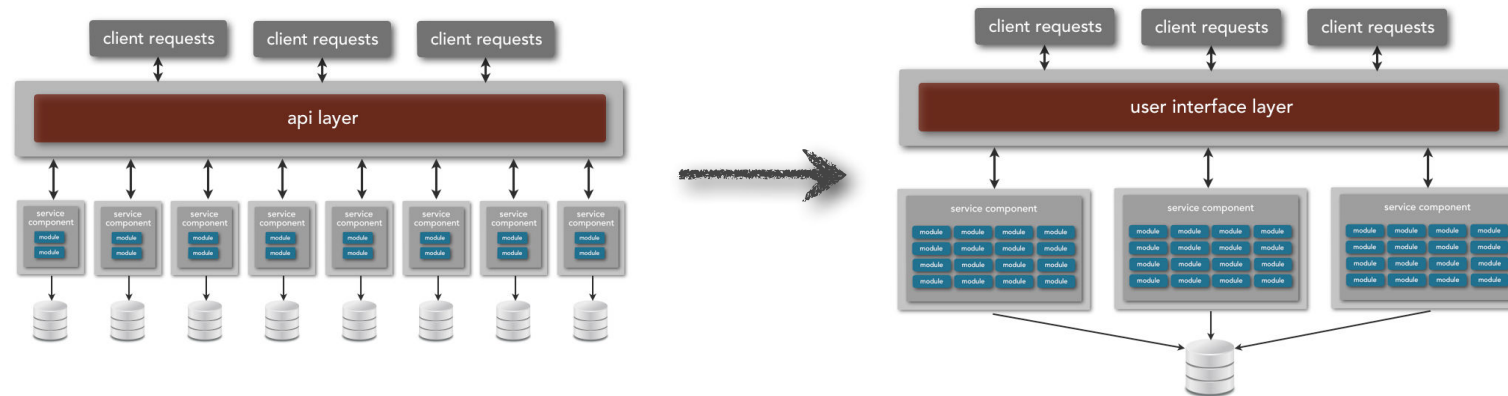
Service-based Architecture

database scope



Service-based Architecture

database scope



tradeoffs



performance



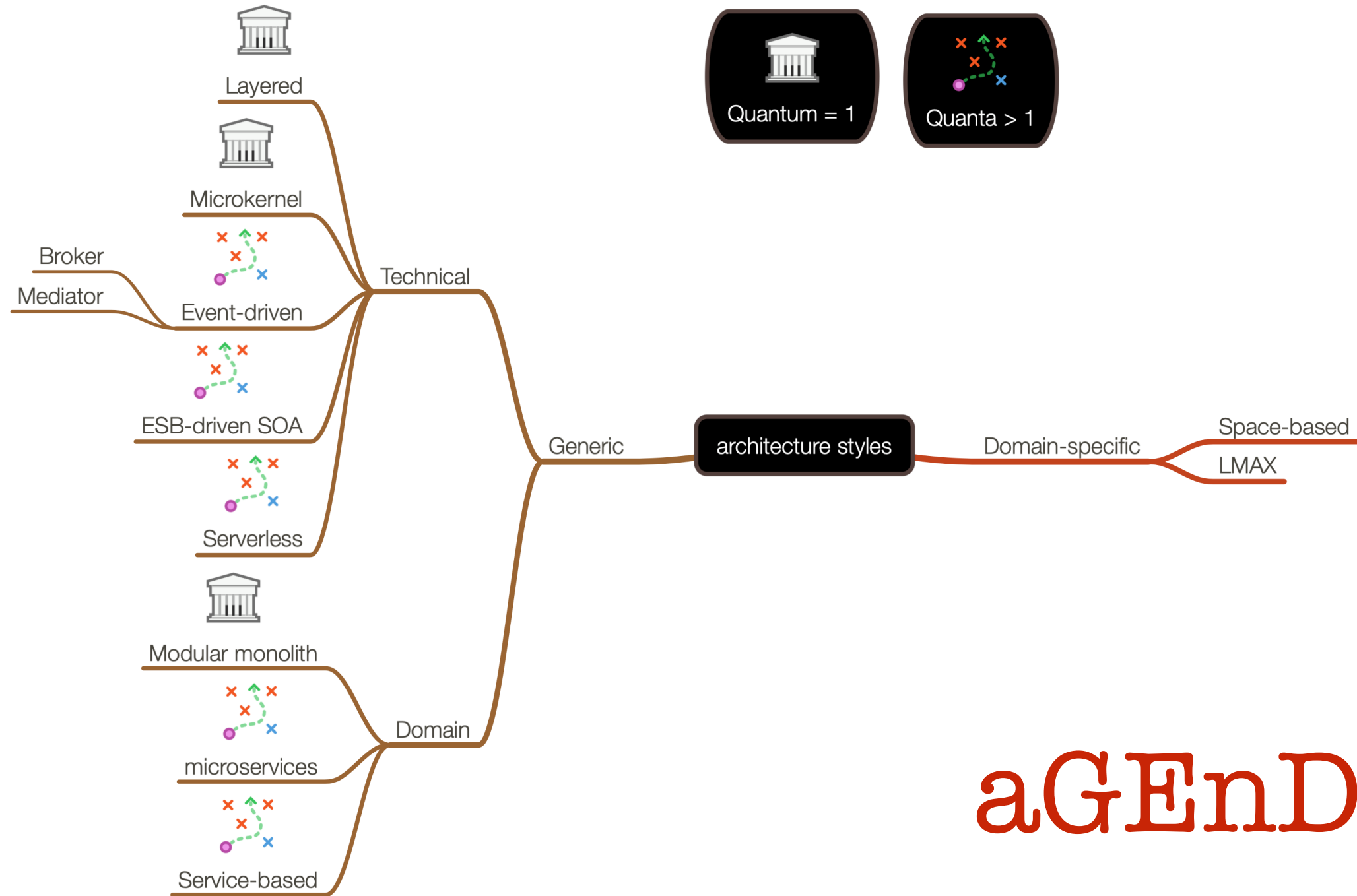
feasibility



service coupling



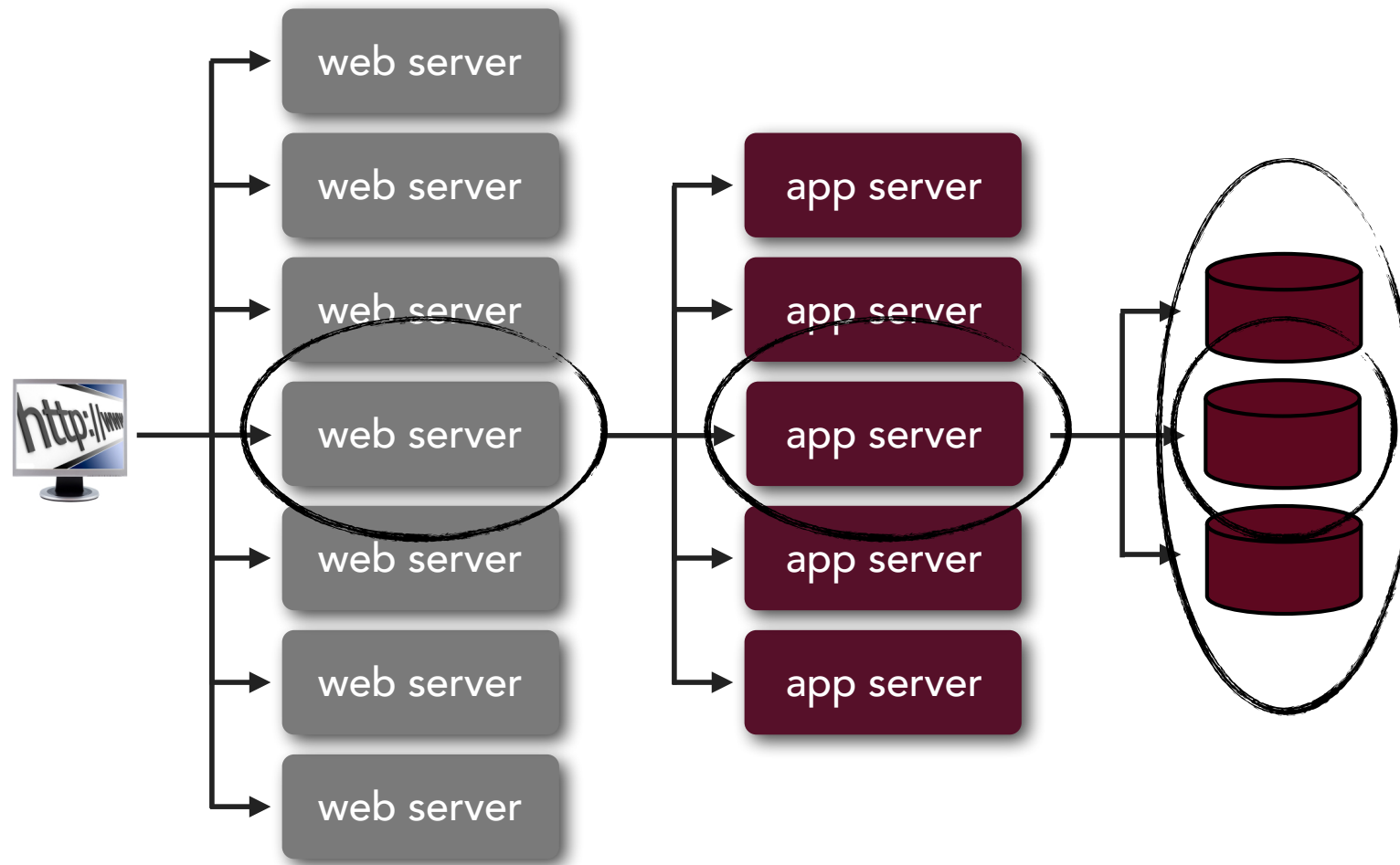
schema changes



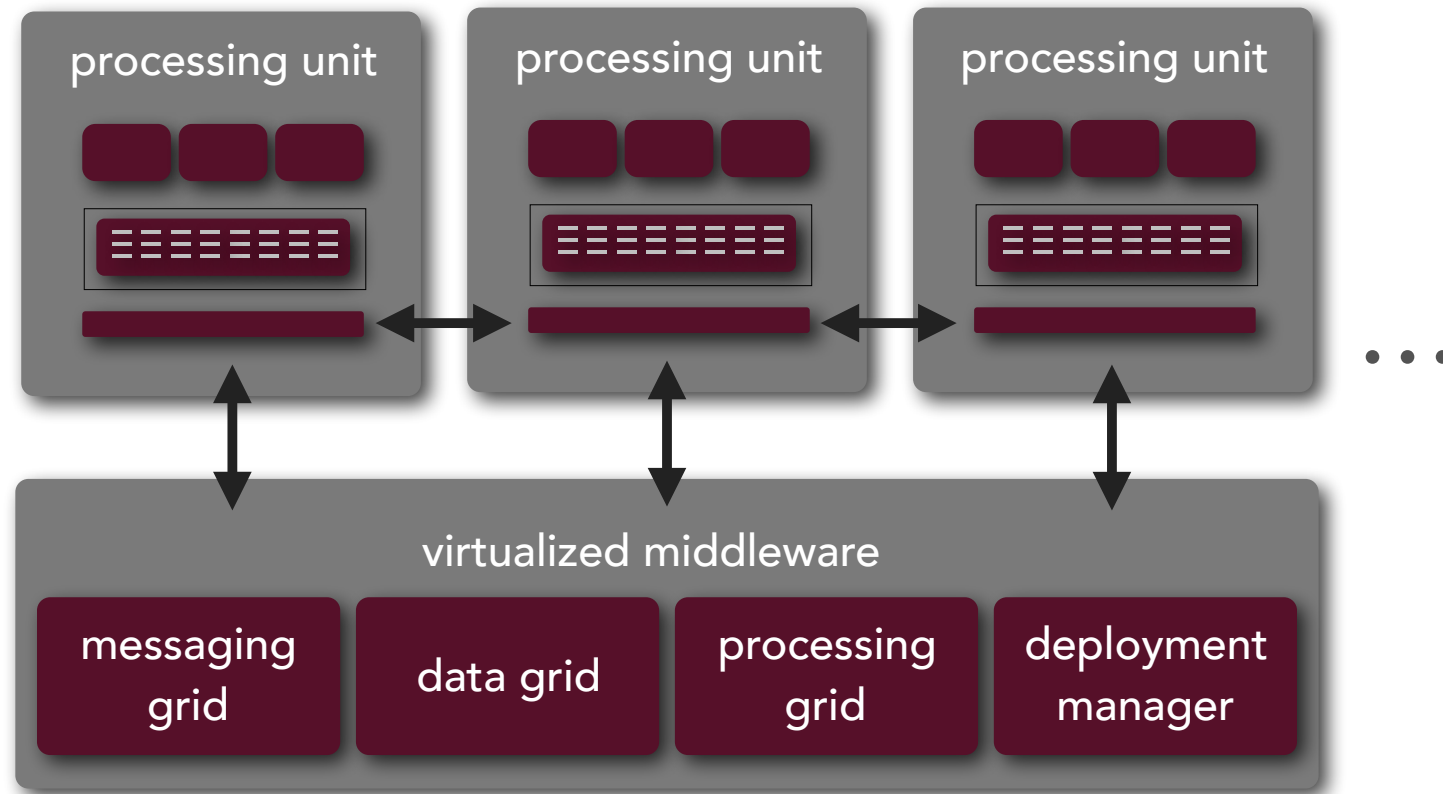
aGEnDA

Space-based Architecture

let's talk about scalability for a moment...

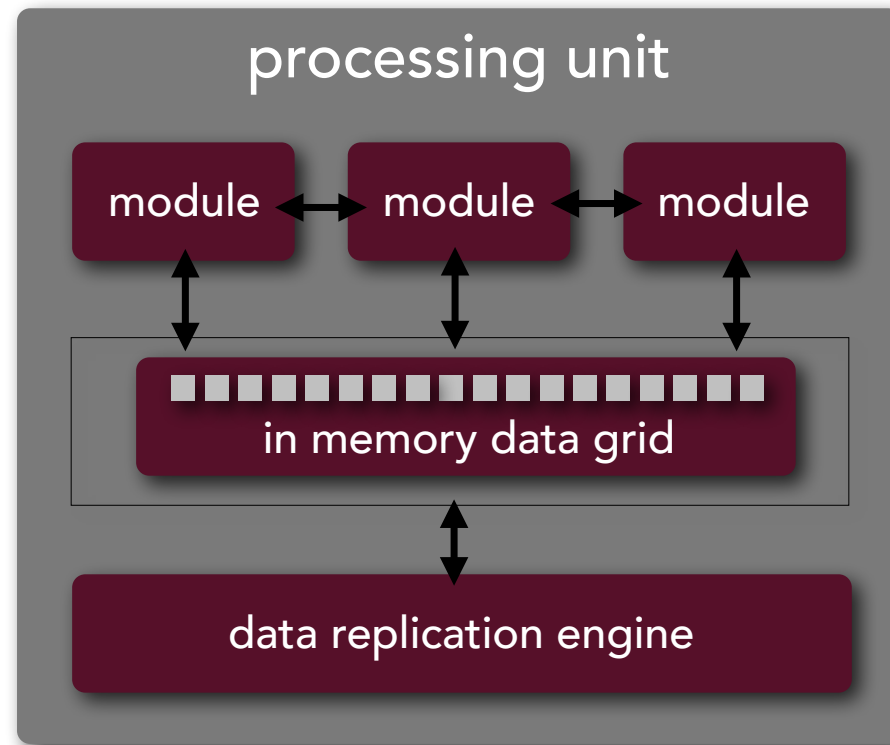


Space-based Architecture



Space-based Architecture

processing unit



Space-based Architecture

middleware

messaging
grid

data grid

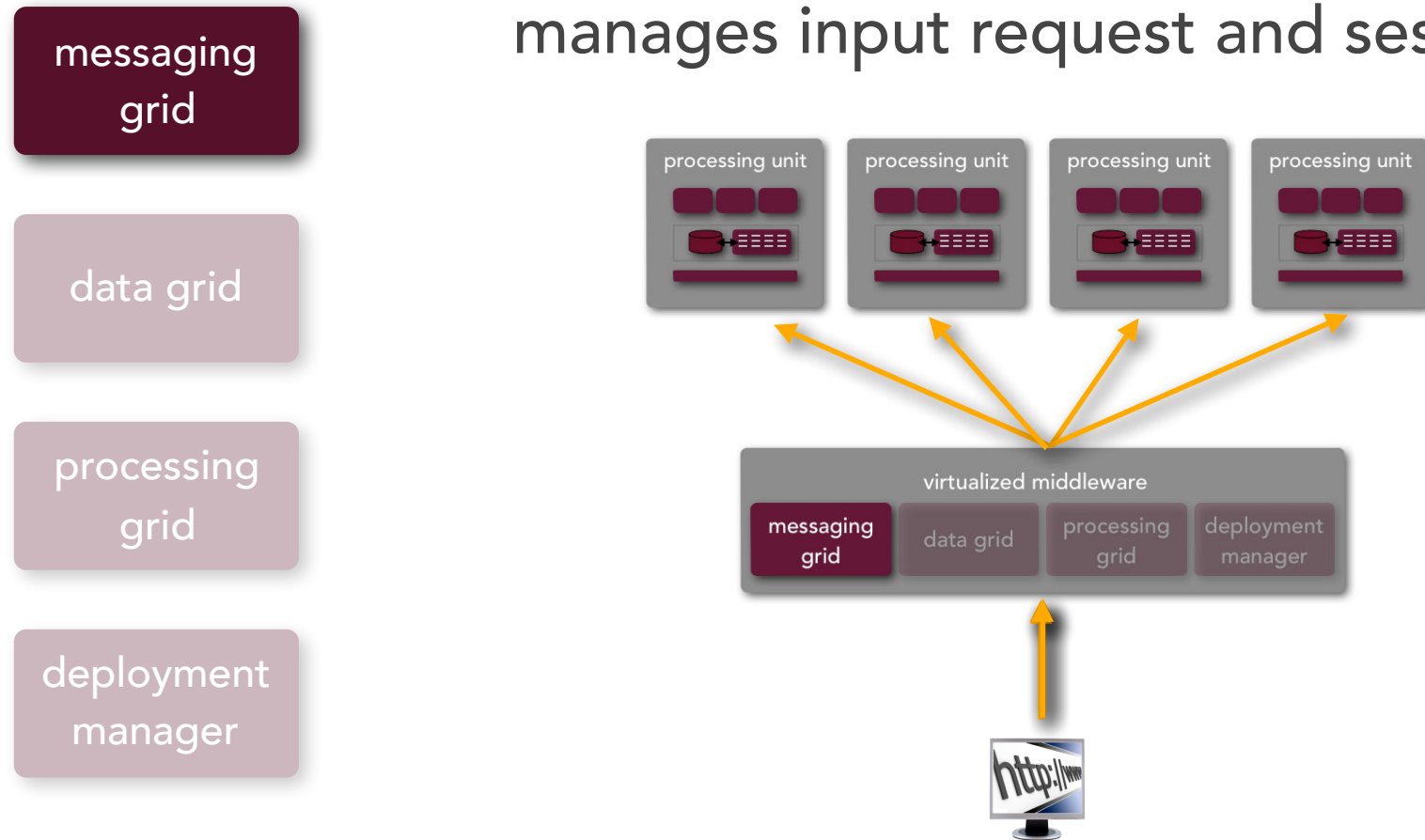
processing
grid

deployment
manager

Space-based Architecture

middleware

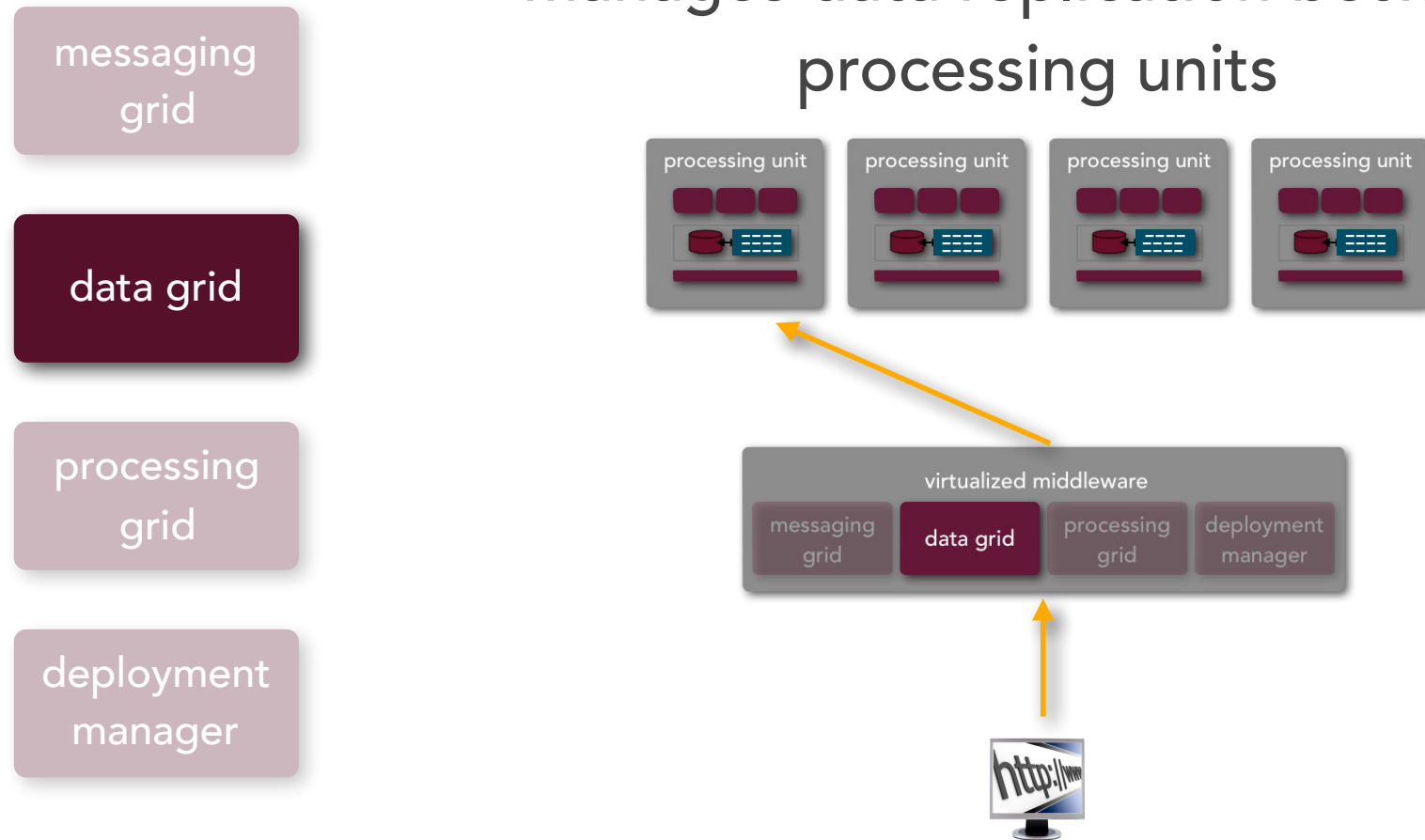
manages input request and session



Space-based Architecture

middleware

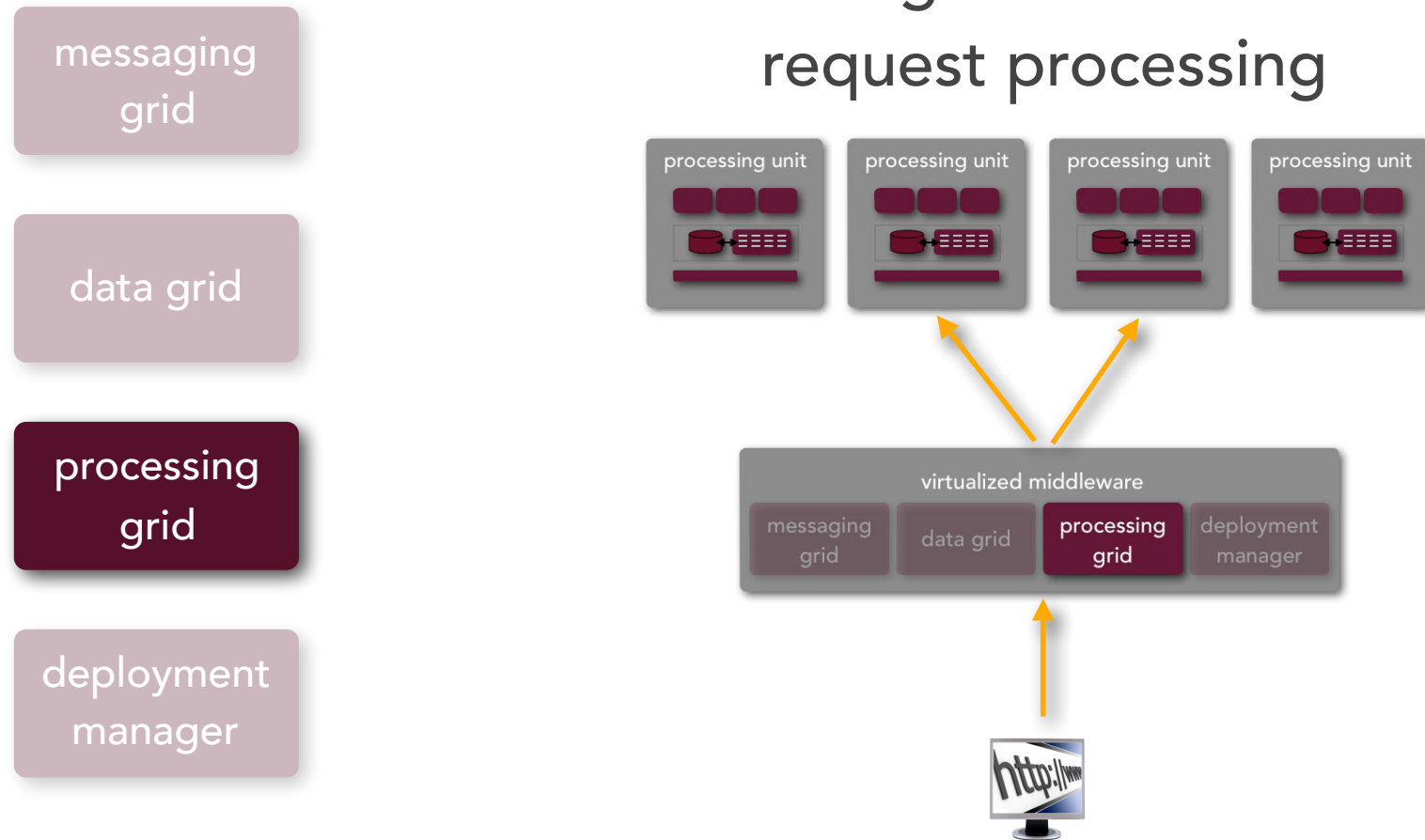
manages data replication between
processing units



Space-based Architecture

middleware

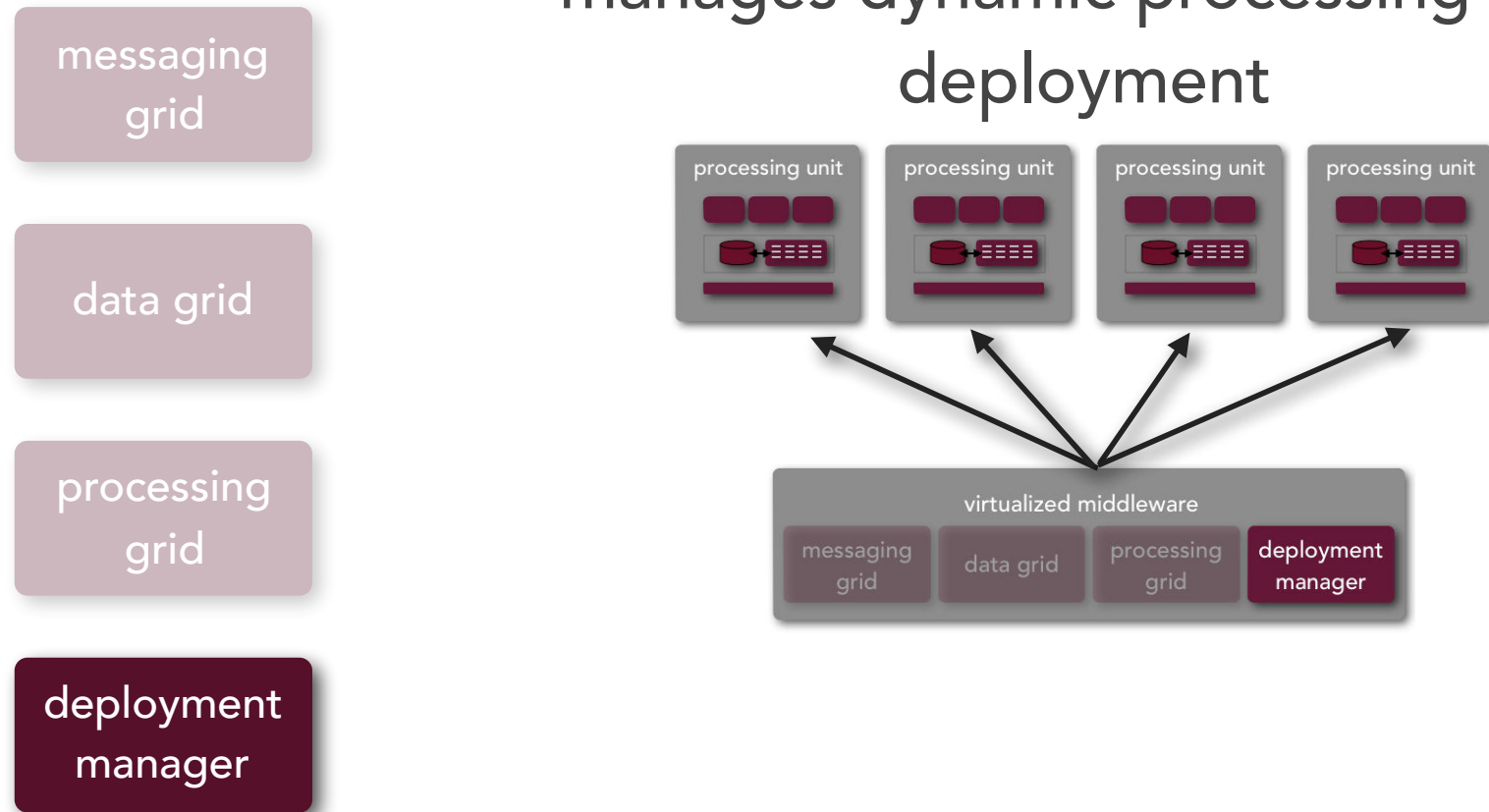
manages distributed
request processing



Space-based Architecture

middleware

manages dynamic processing unit
deployment



Space-based Architecture

product implementations

javaspaces

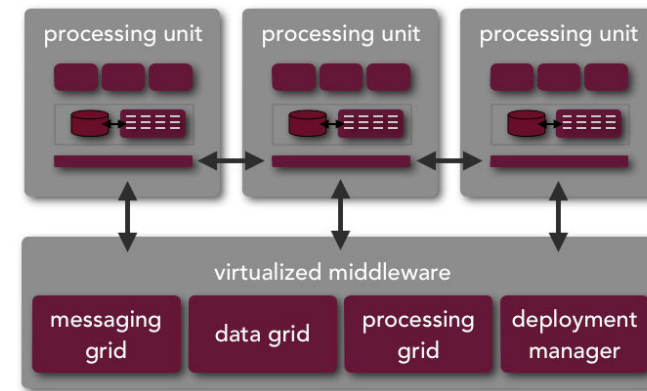
gigaspaces

ibm object grid

gemfire

ncache

oracle coherence



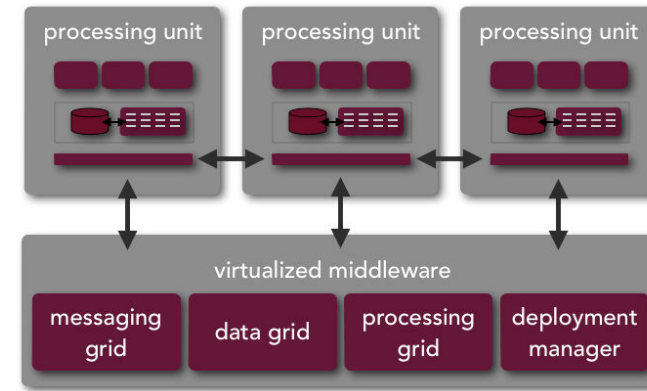
Space-based Architecture

it's all about variable scalability...

good for applications that have variable load or inconsistent peak times

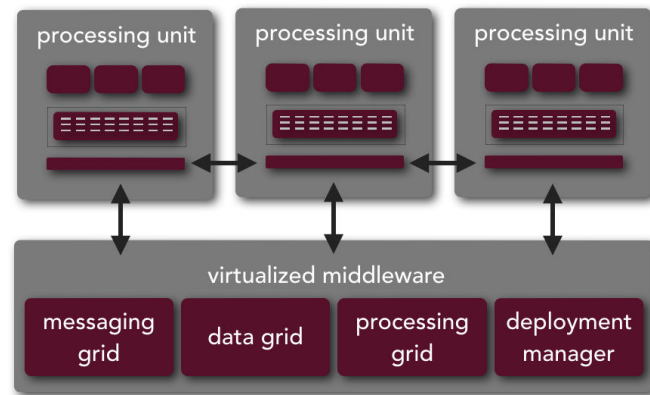
not a good fit for traditional large-scale relational database systems

relatively complex and expensive pattern to implement

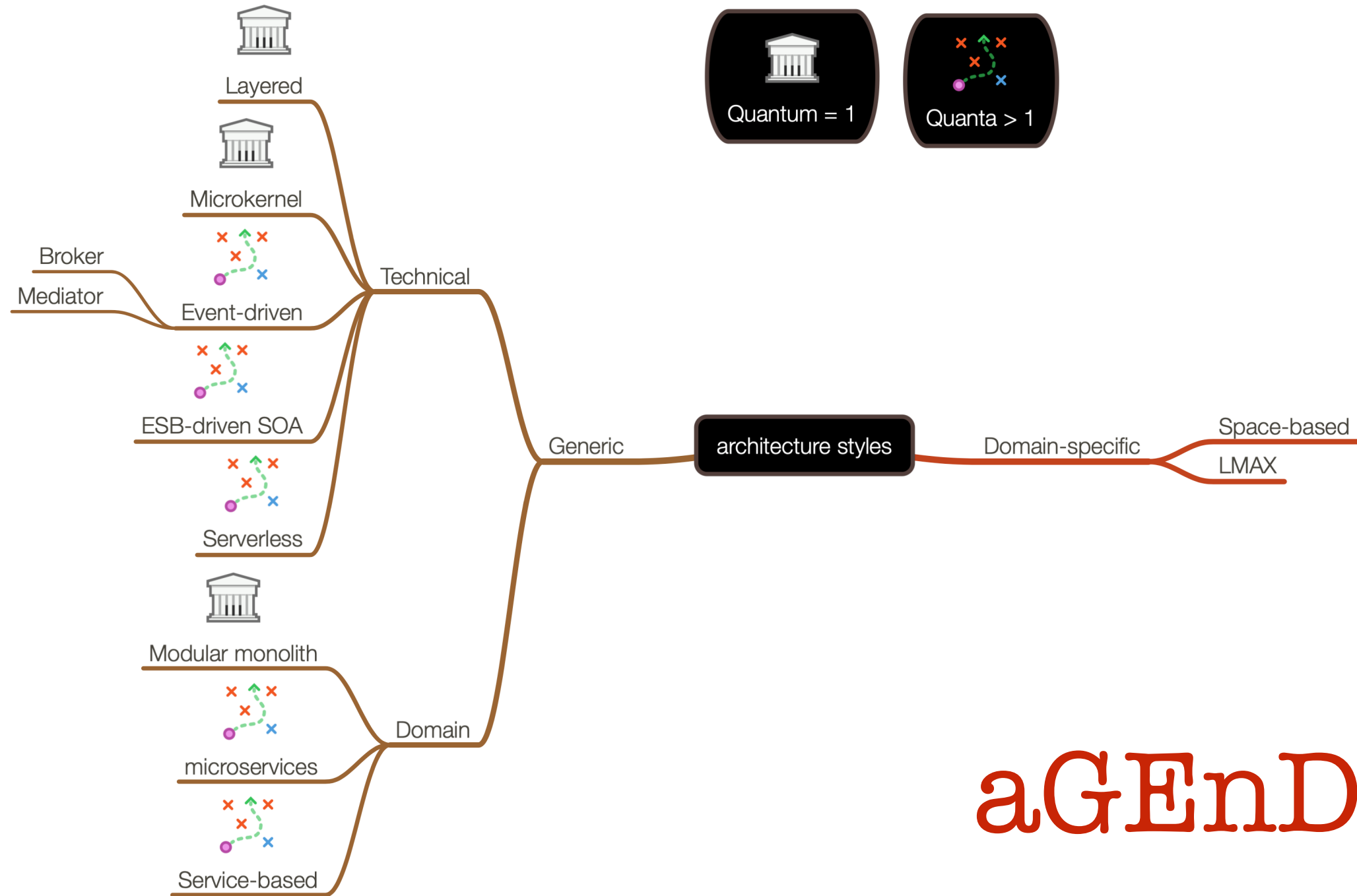


Space-based Architecture

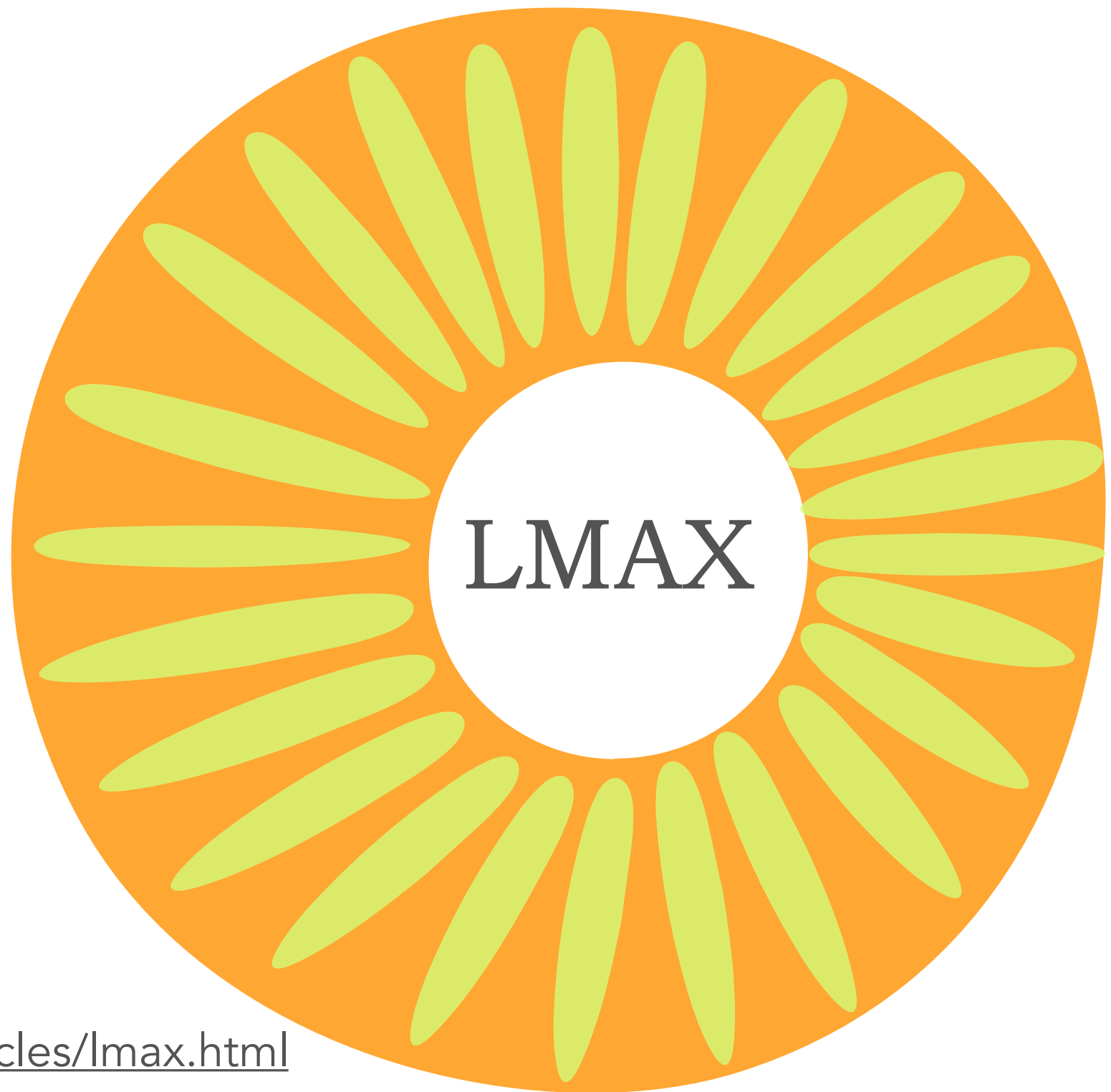
drivers



- ✓ scalability
- ✓ elasticity
- ✓ performance

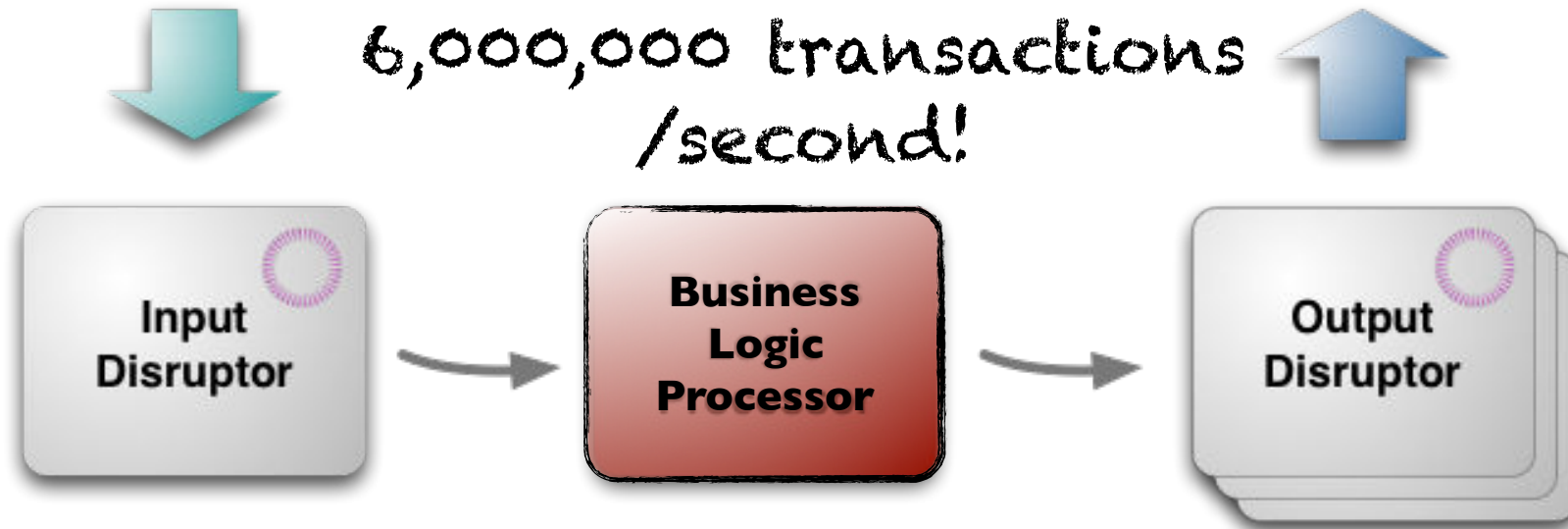


aGEnDA



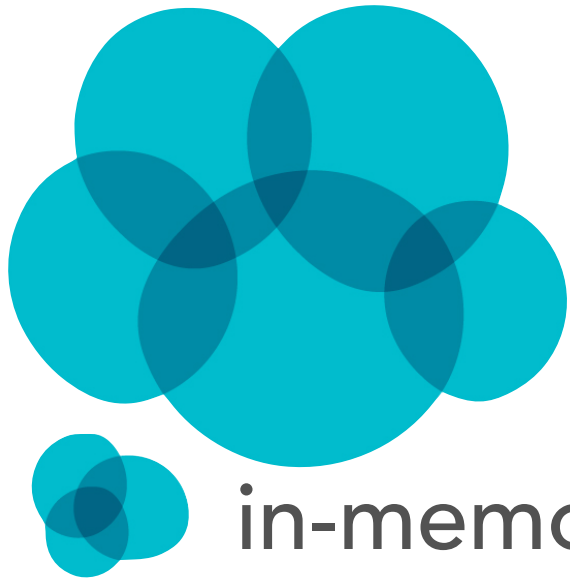
<http://martinfowler.com/articles/lmax.html>

Overall Structure



**Business
Logic
Processor**

Business Logic Processor



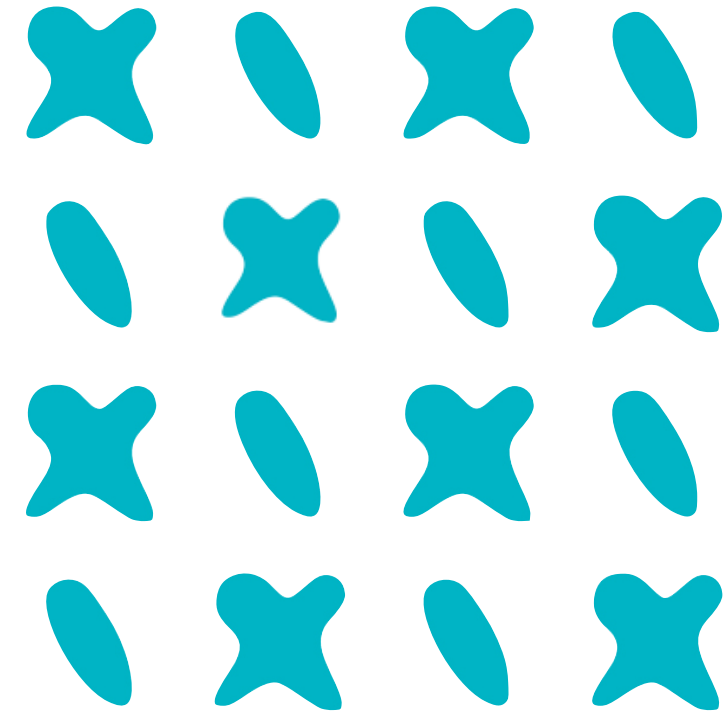
in-memory



snapshots



(full restart—JVM + snapshots— < 1 min)



multiple instances running

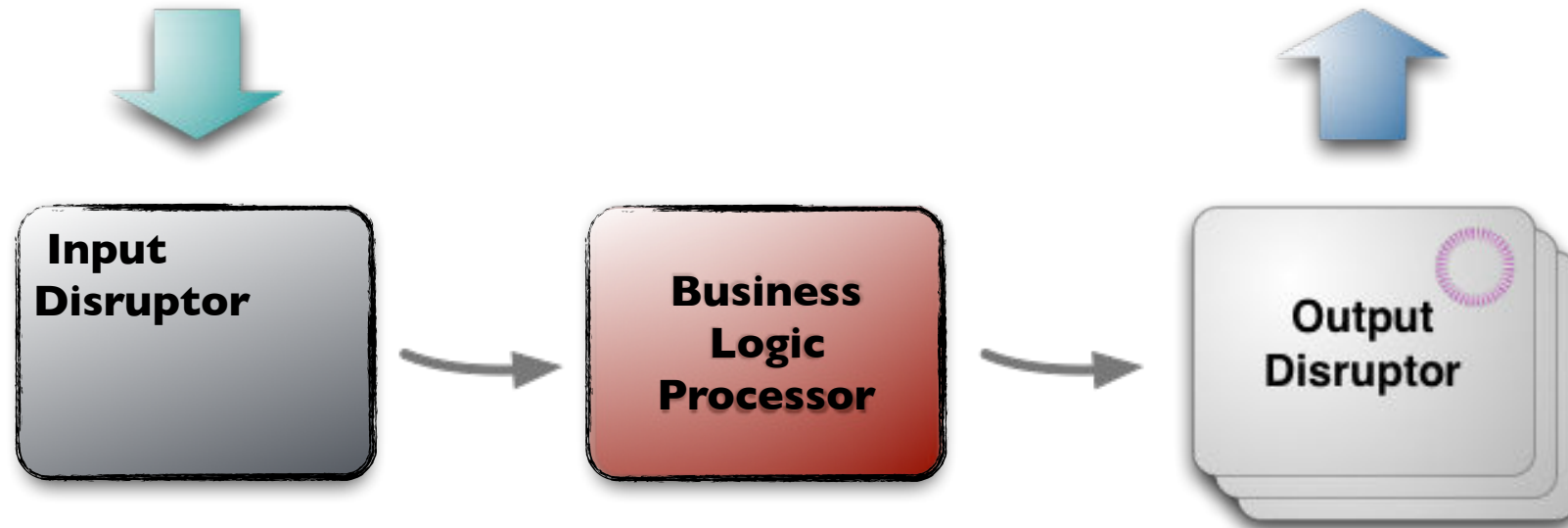
each event processed by multiple processors

but only one result used

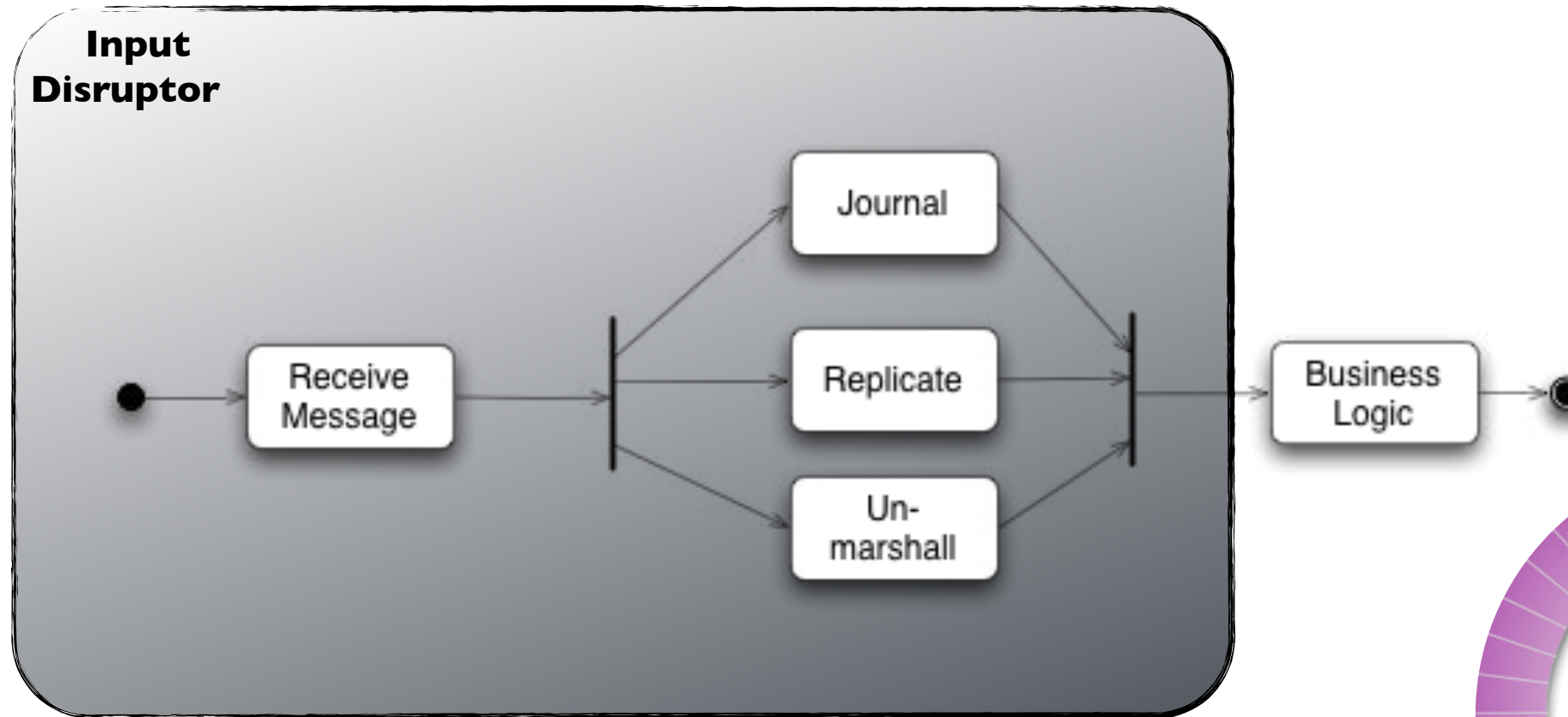


JAVA™

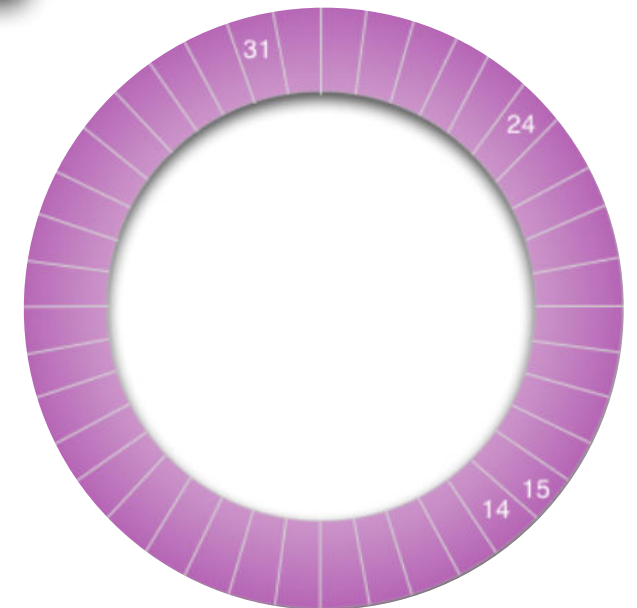
Overall Structure



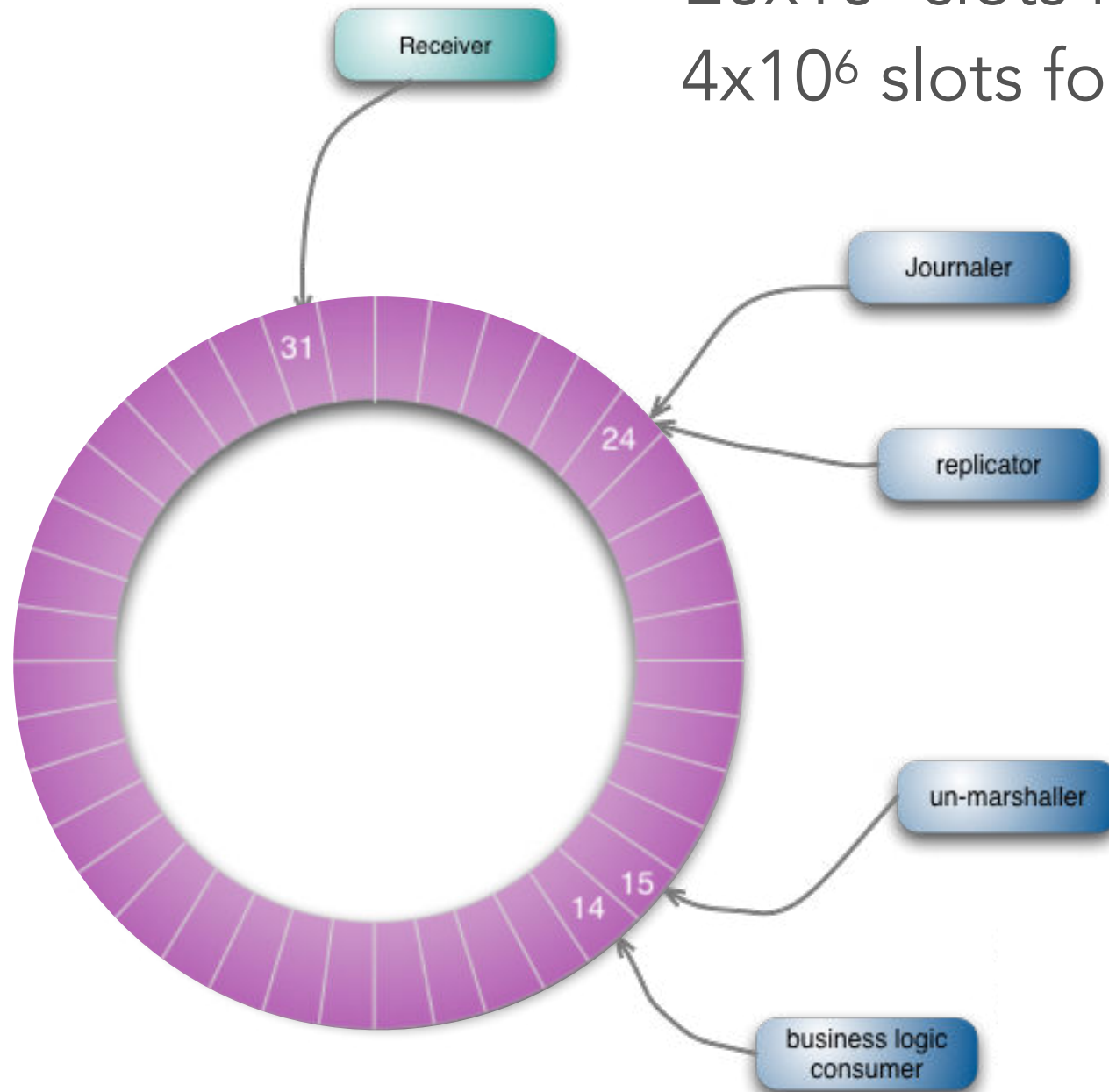
Input Disruptor



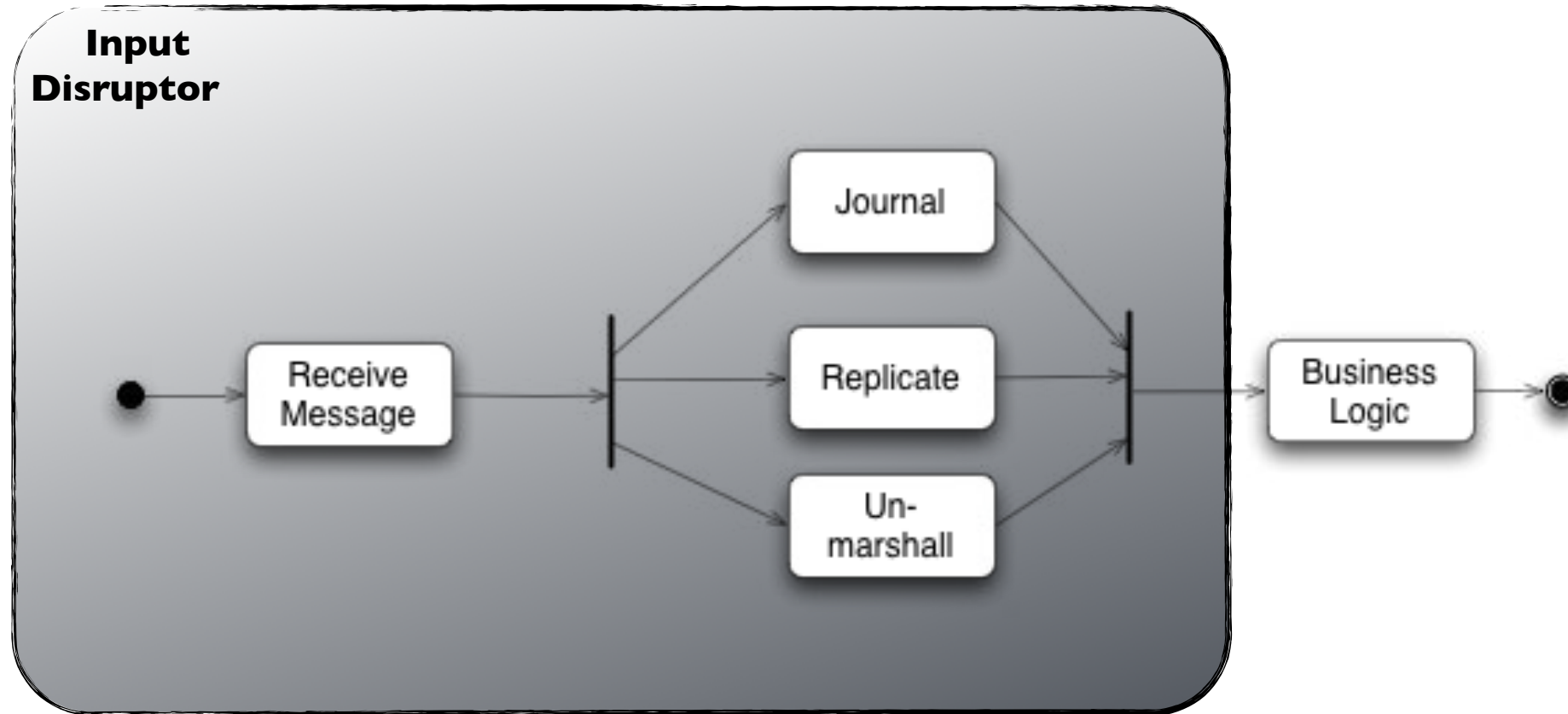
custom concurrency component



20x10⁶ slots for input buffer
4x10⁶ slots for output buffer

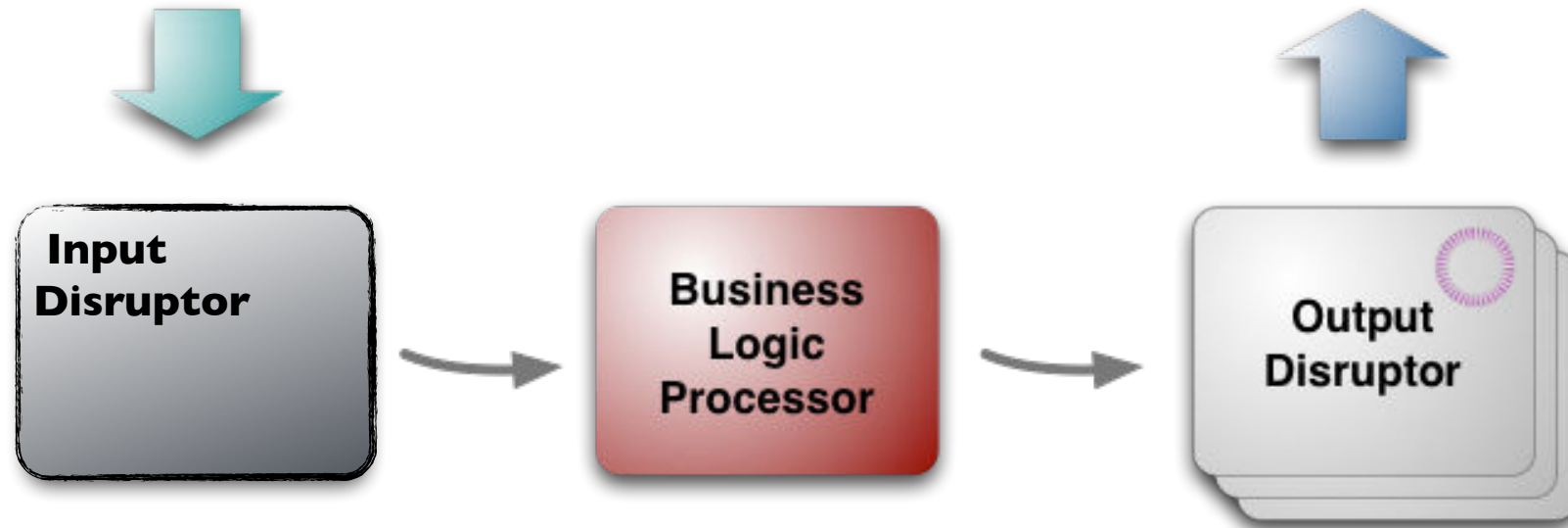


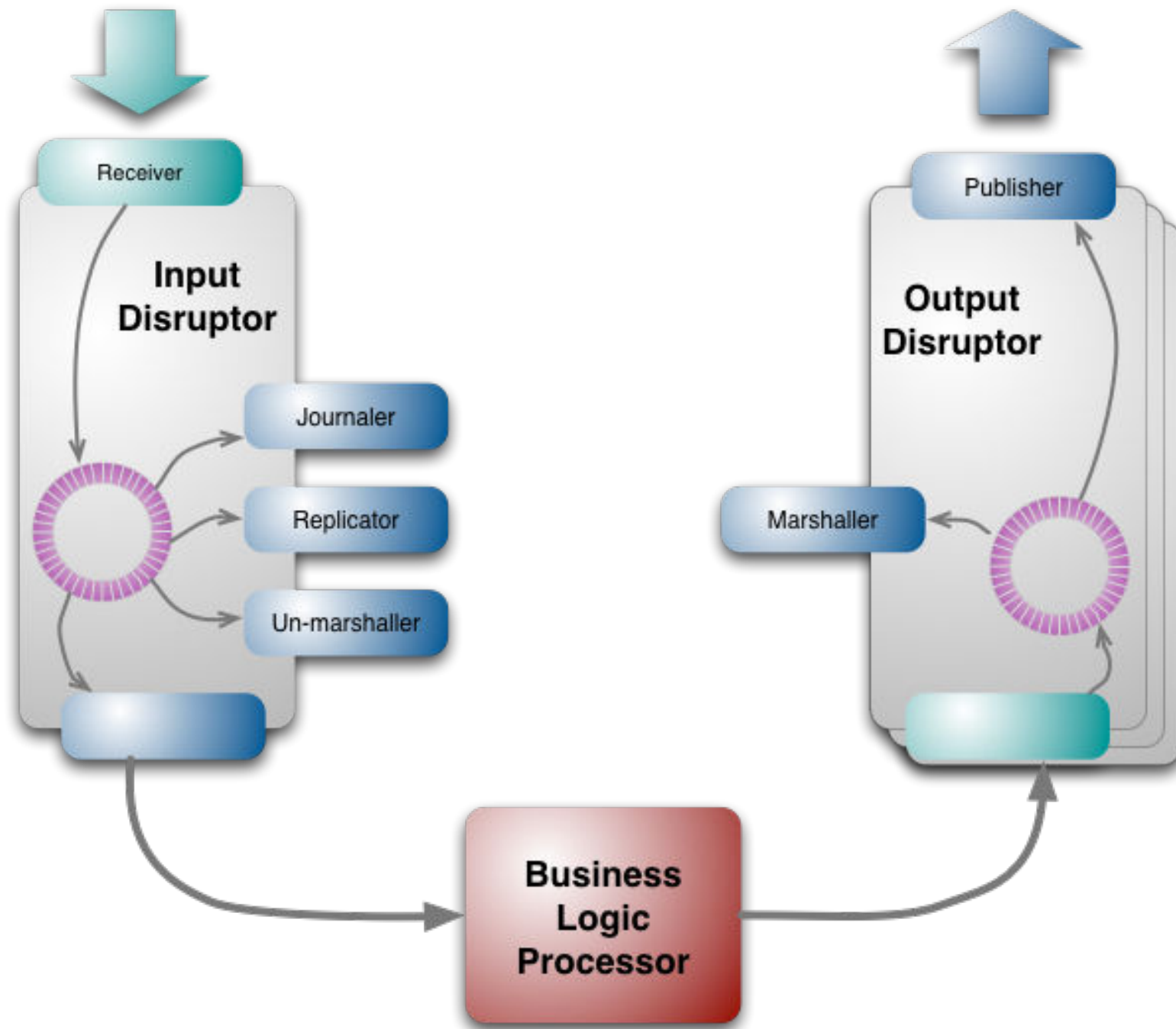
Input Disruptor

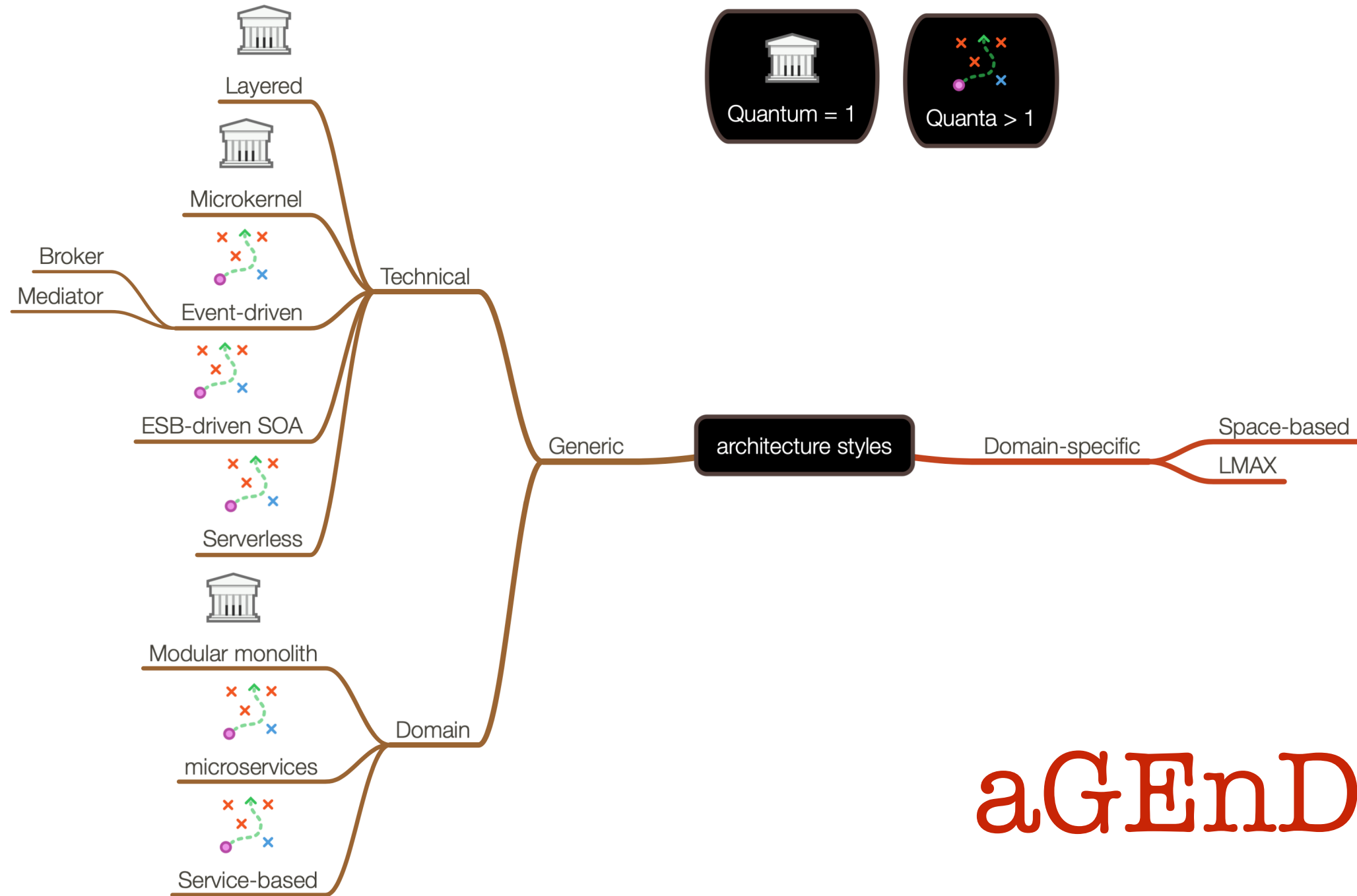


custom concurrency component

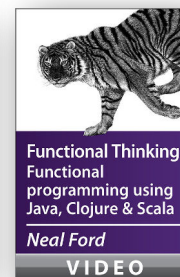
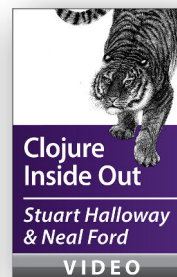
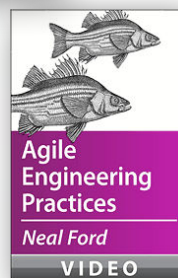
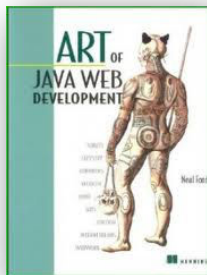
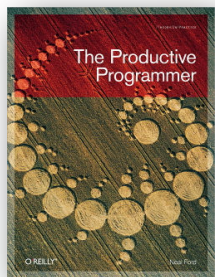
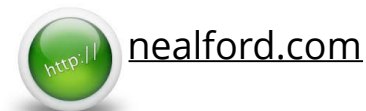
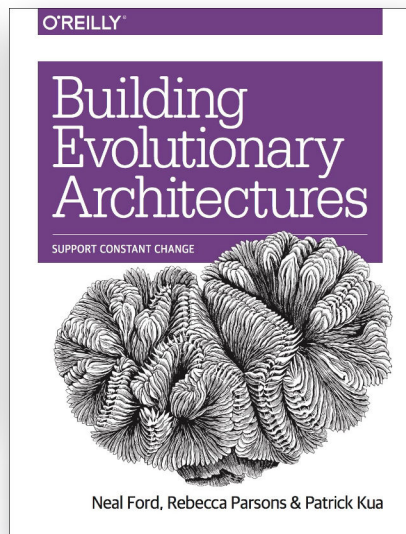
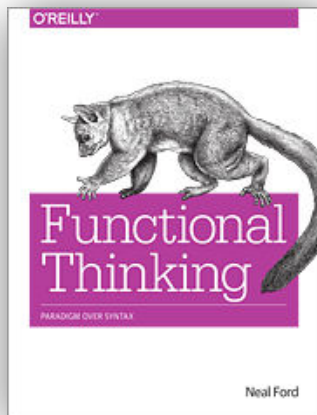
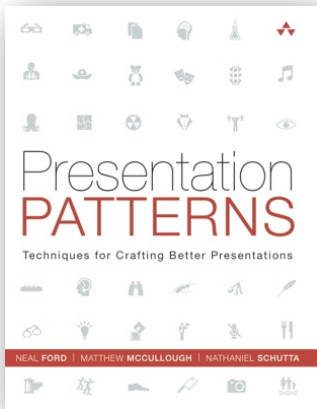
Overall Structure



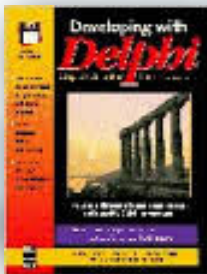
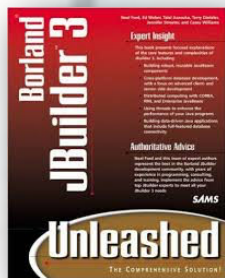




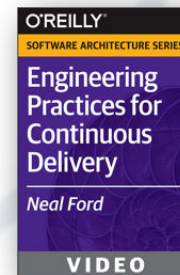
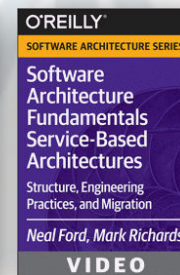
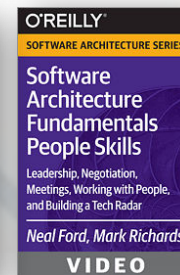
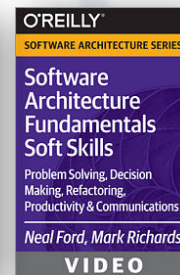
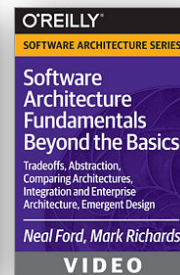
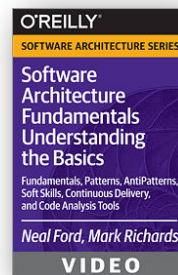
aGEnDA



nealford.com/videos



www.oreilly.com/software-architecture-video-training-series.html



nealford.com/books