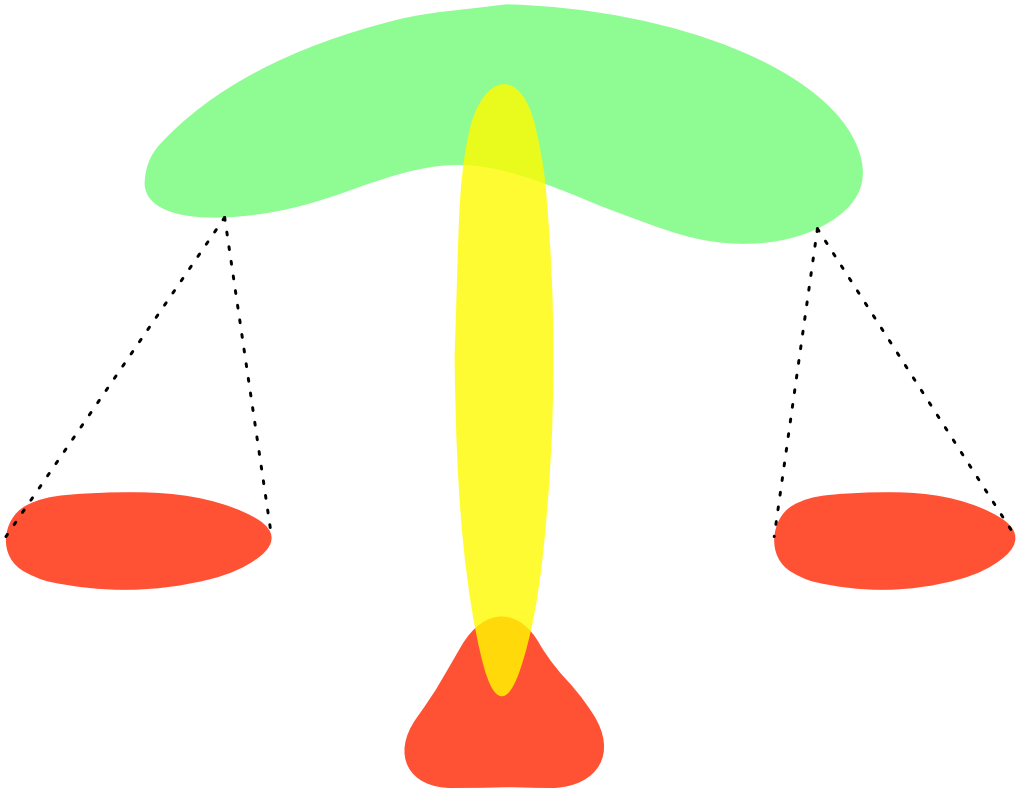
Comparing Service-based Architectures



ThoughtWorks®

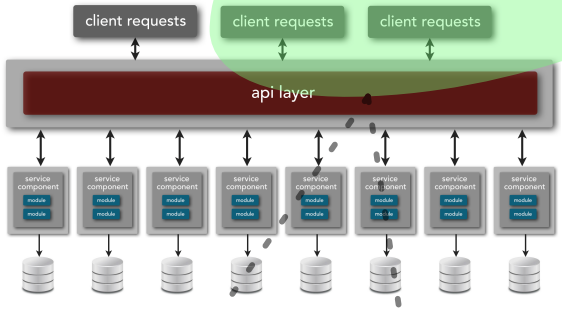
NEAL FORD

Director / Software Architect / Meme Wrangler

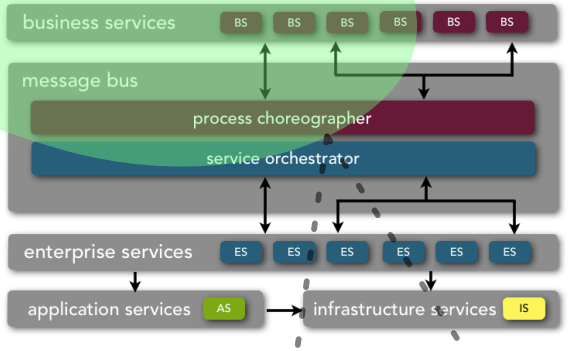
@neal4d

nealford.com

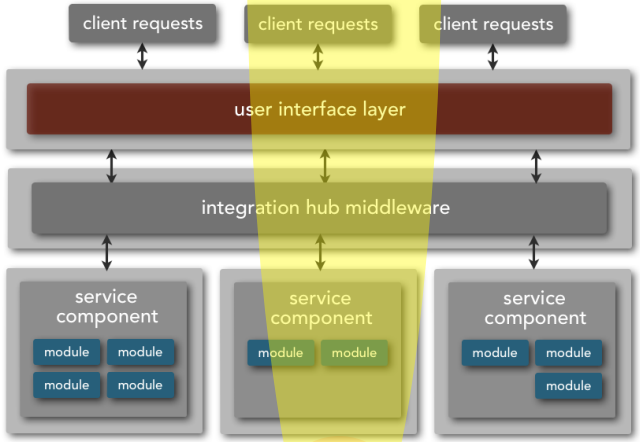
Agenda



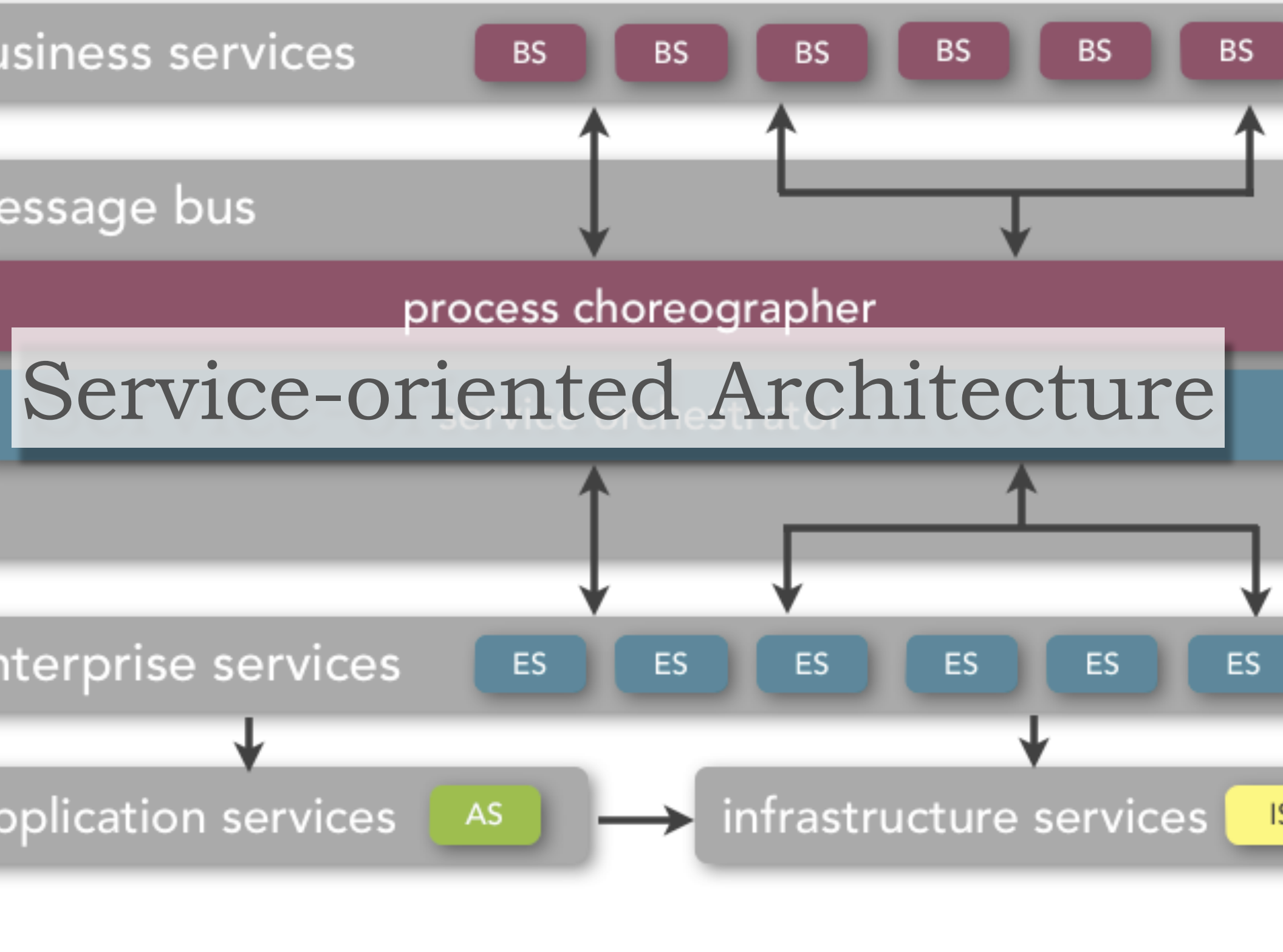
Micro



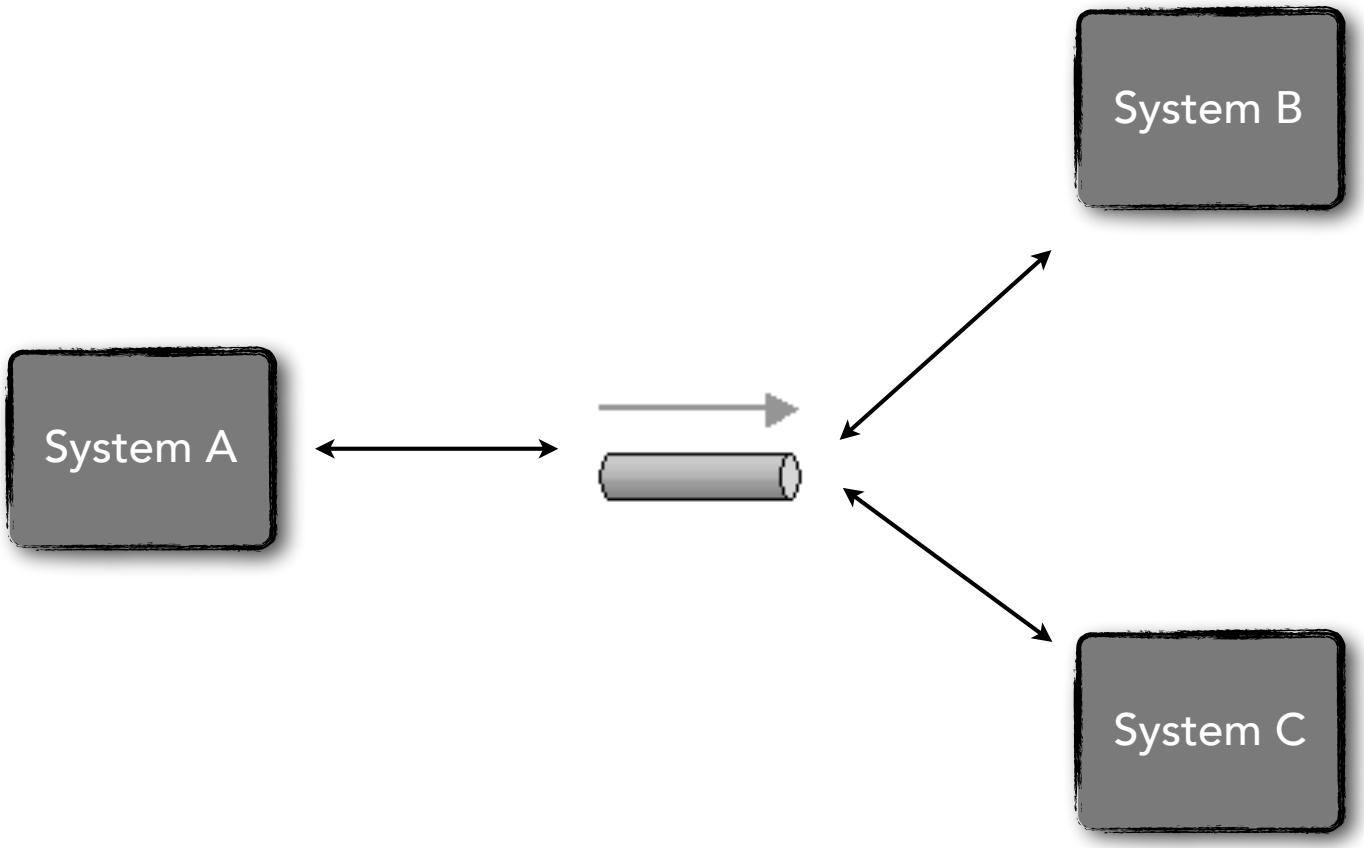
Service-oriented



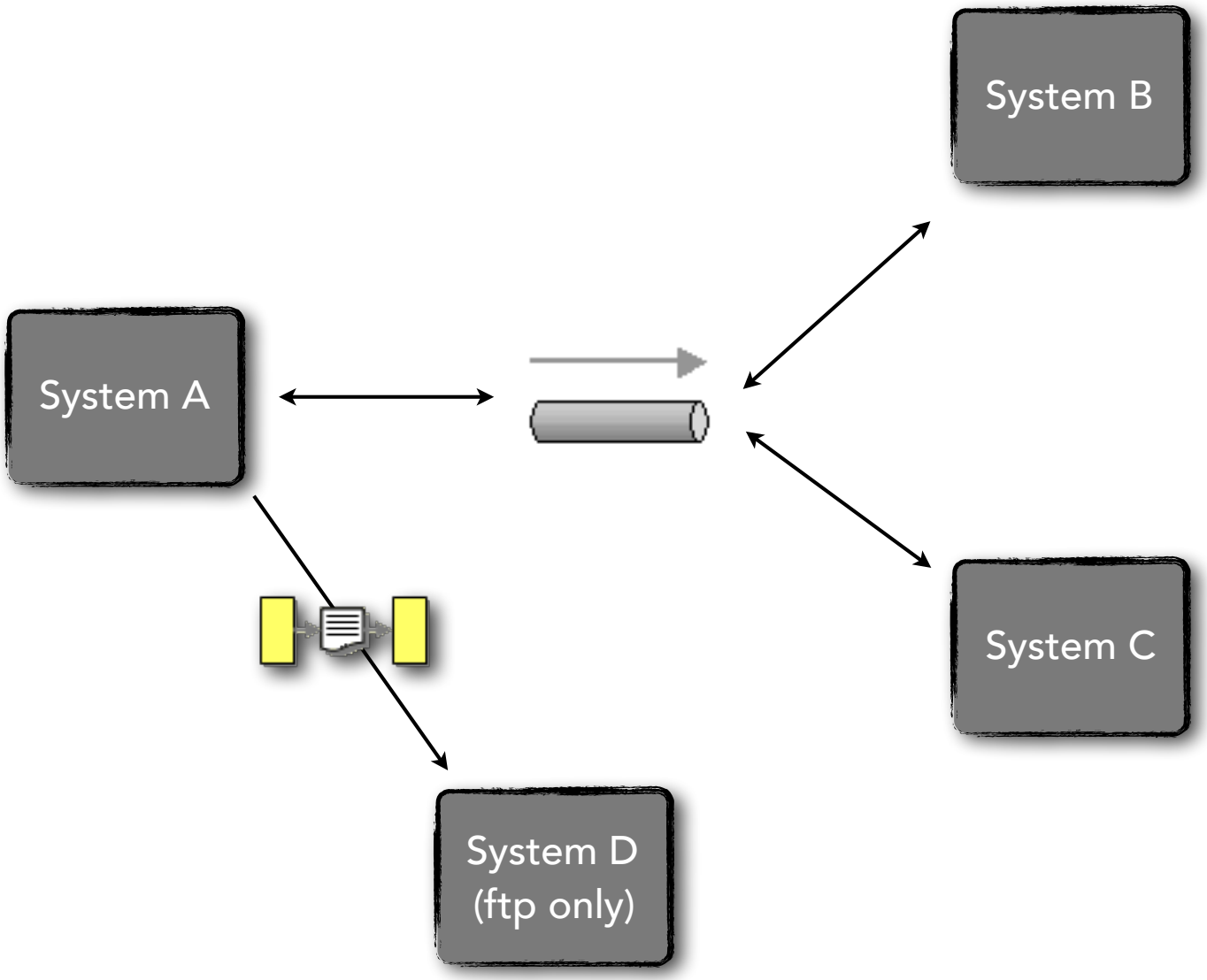
Service-based


usiness services
BS
BS
BS
BS
BS
BS
essage bus
process choreographer
Service-oriented Architecture
nterprise services
ES
ES
ES
ES
ES
ES
pplication services
AS
infrastructure services

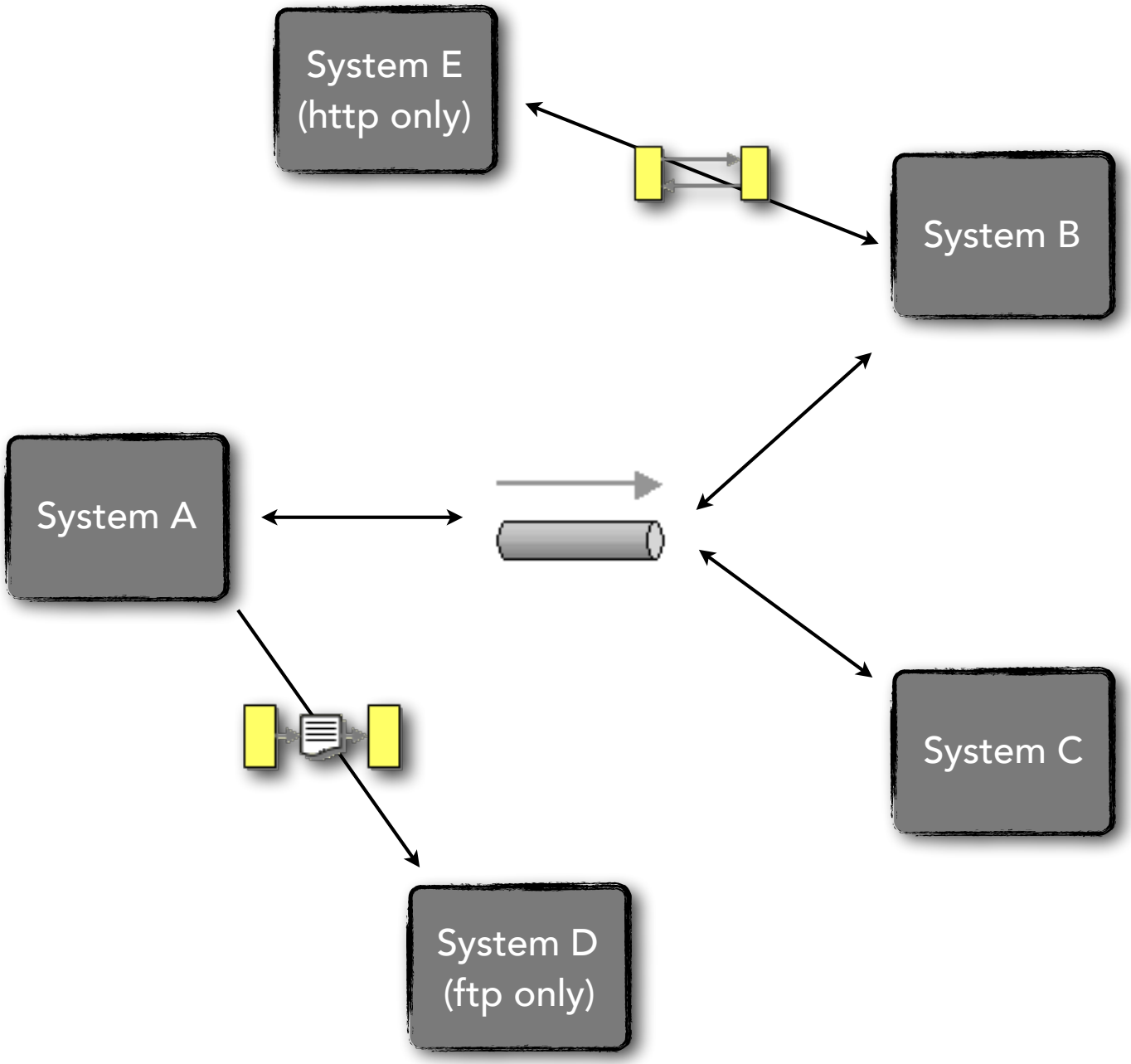
Origins: Hubs



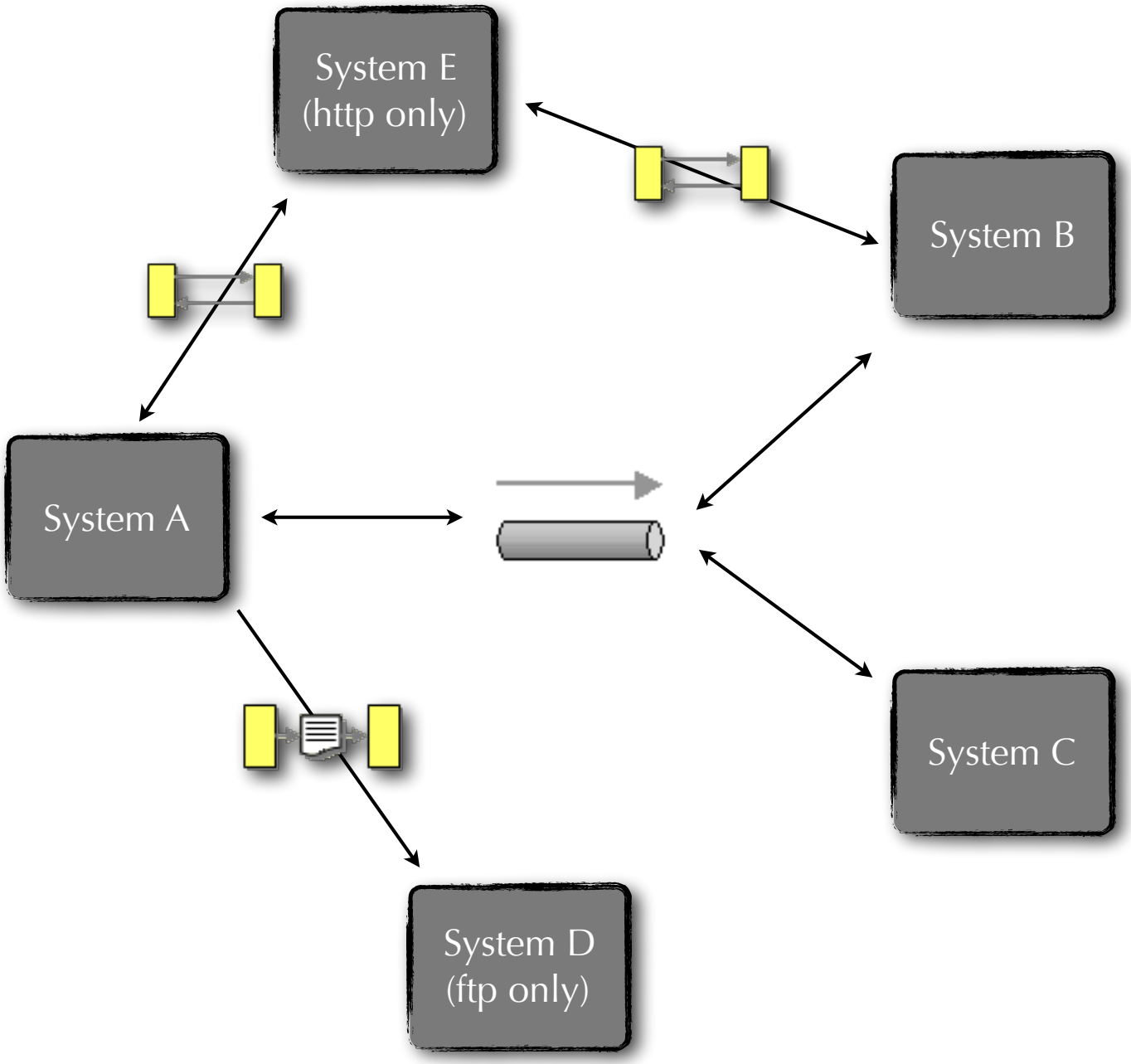
Origins: Hubs



Origins: Hubs

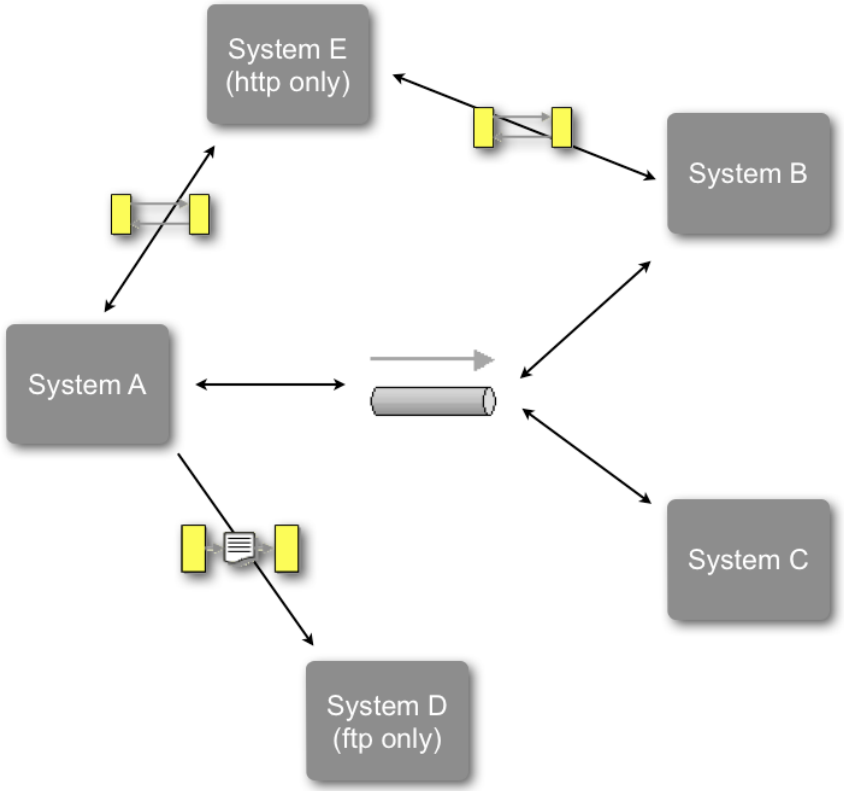


Origins: Hubs

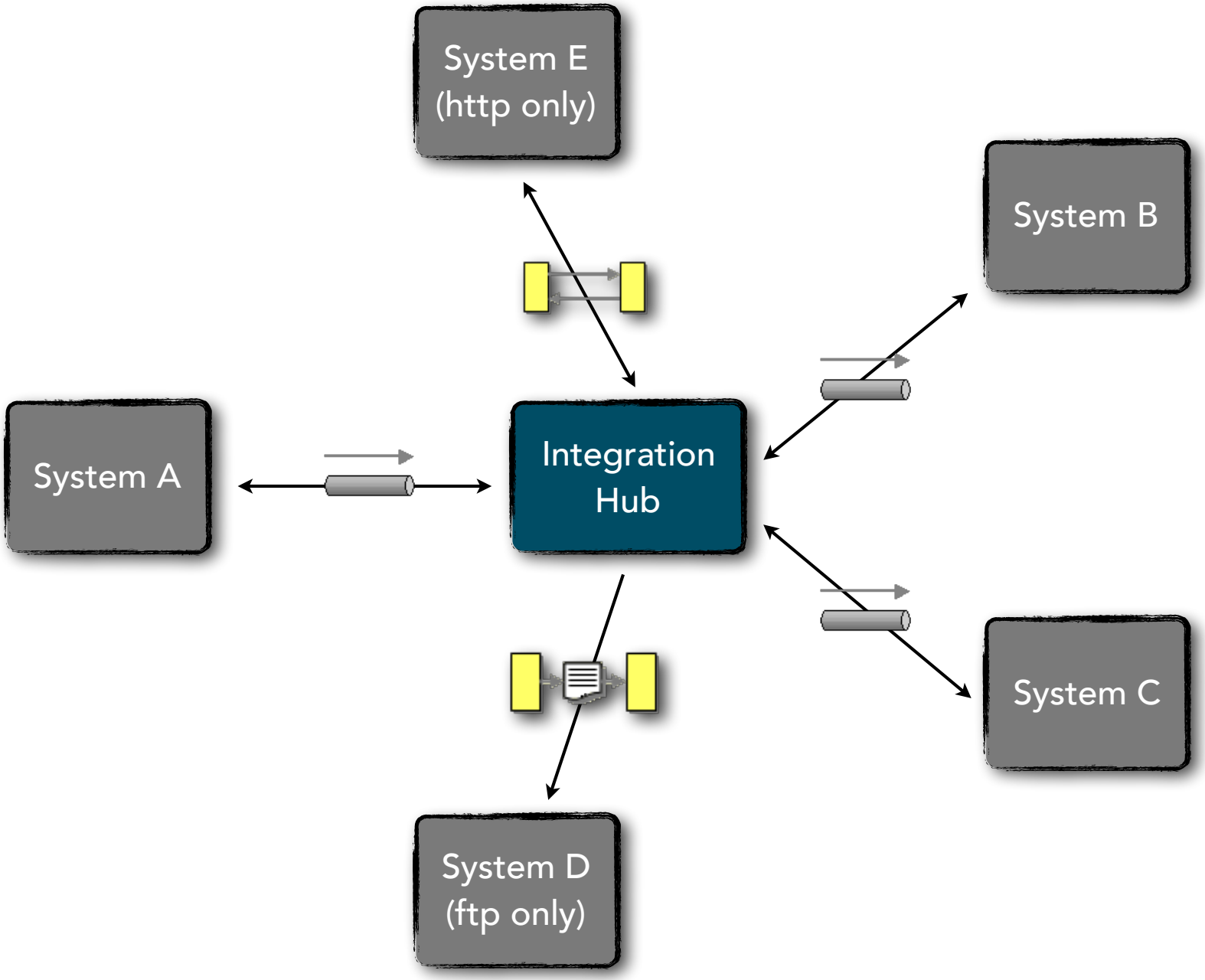


Origins: Hubs

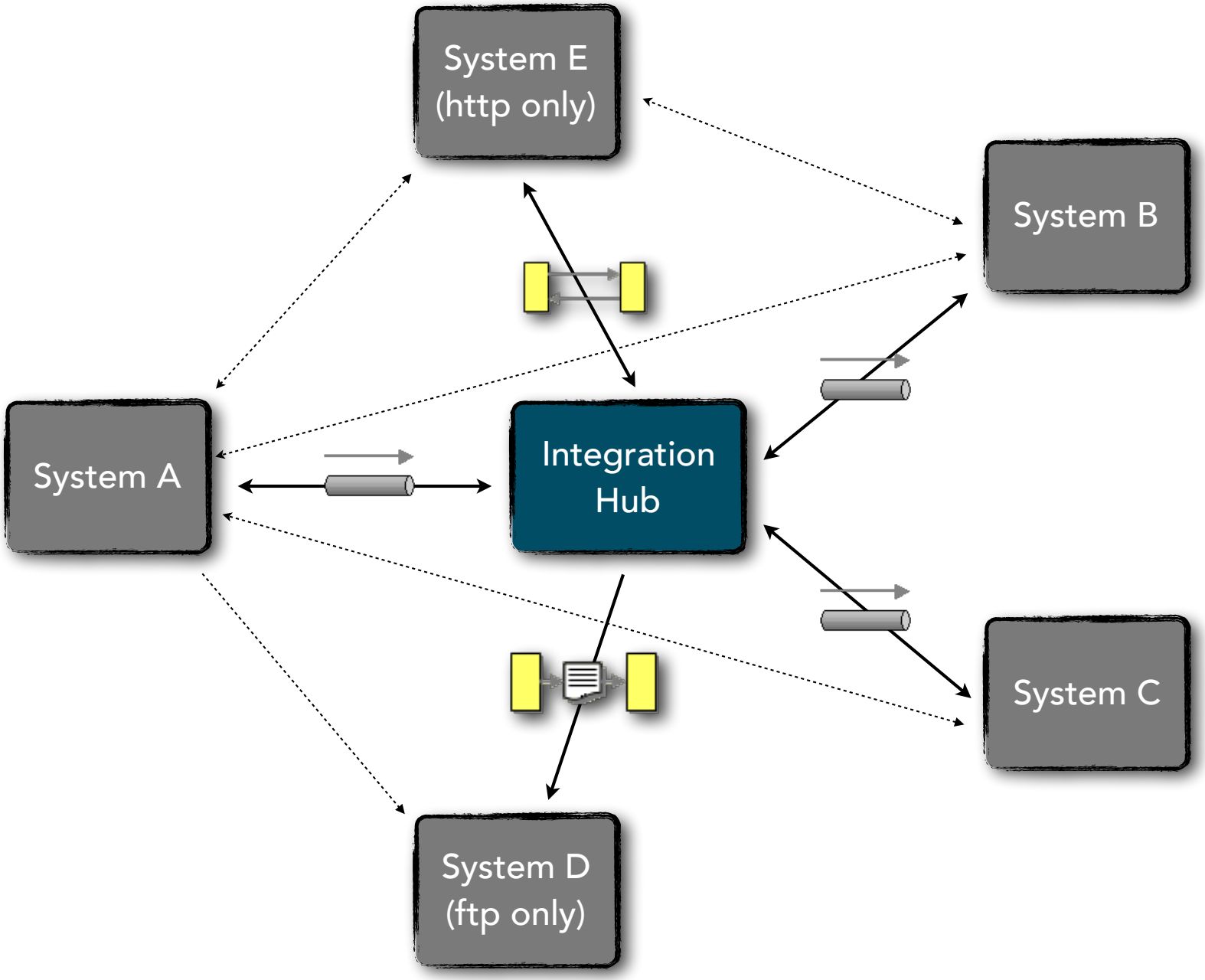
this is starting to look like a complex, highly dependent spaghetti topology...



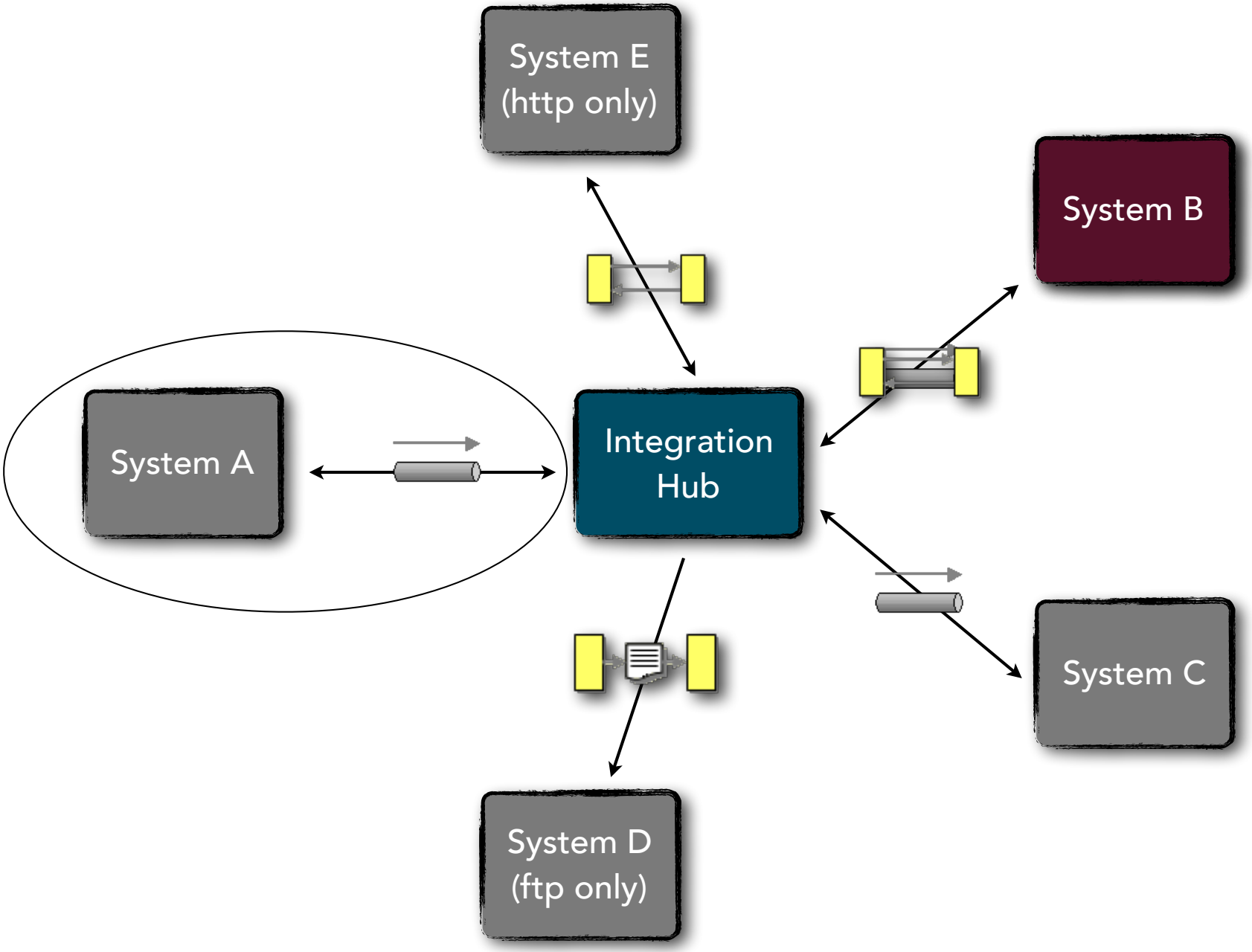
Origins: Hubs



Origins: Hubs

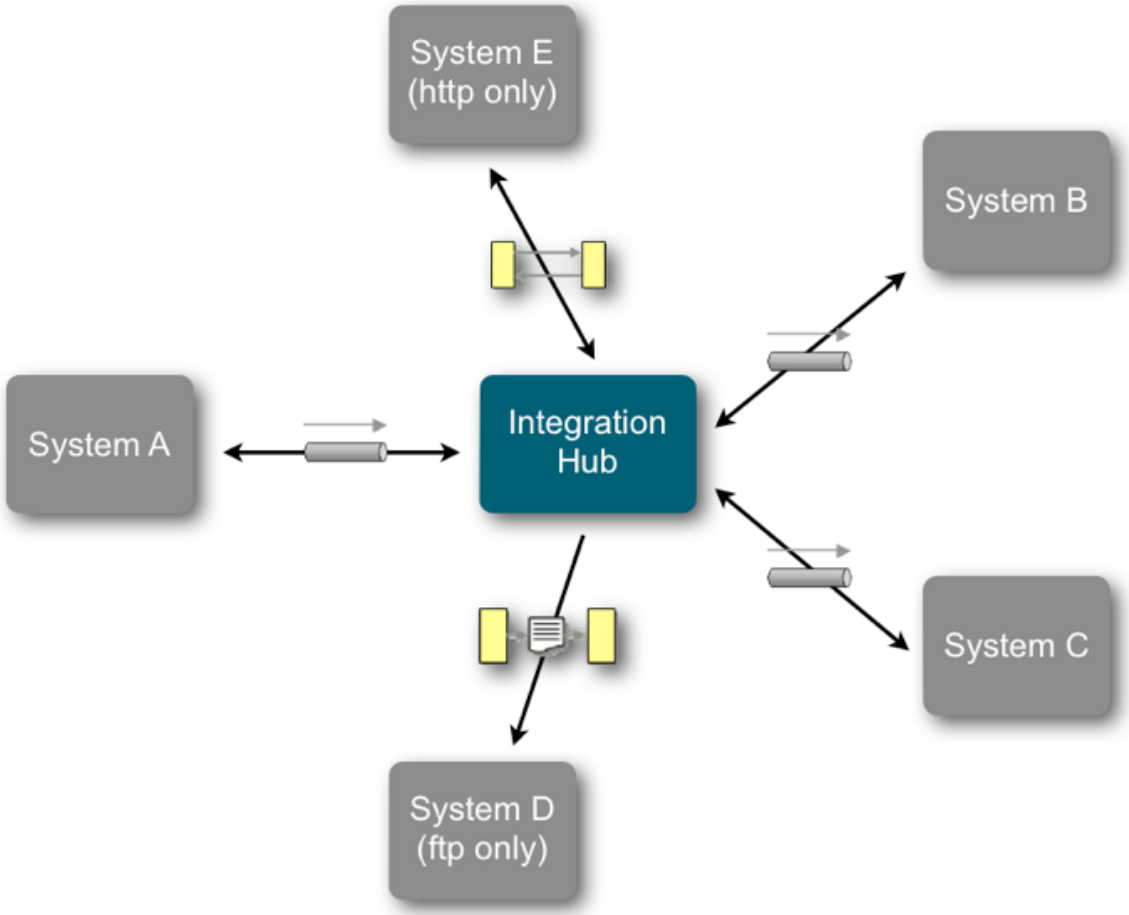


Origins: Hubs

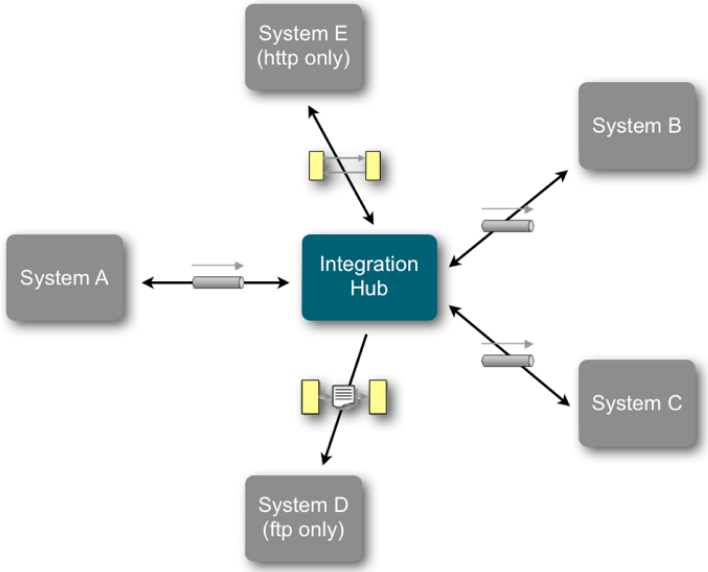


Origins: Hubs

looks great, but what about single point of failure and performance bottleneck considerations?



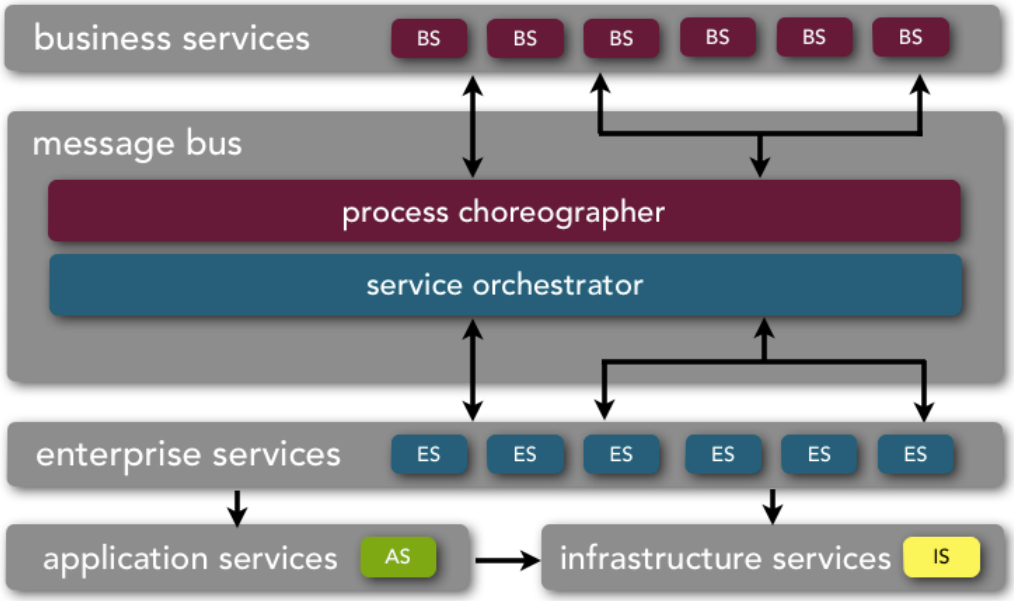
orchestration



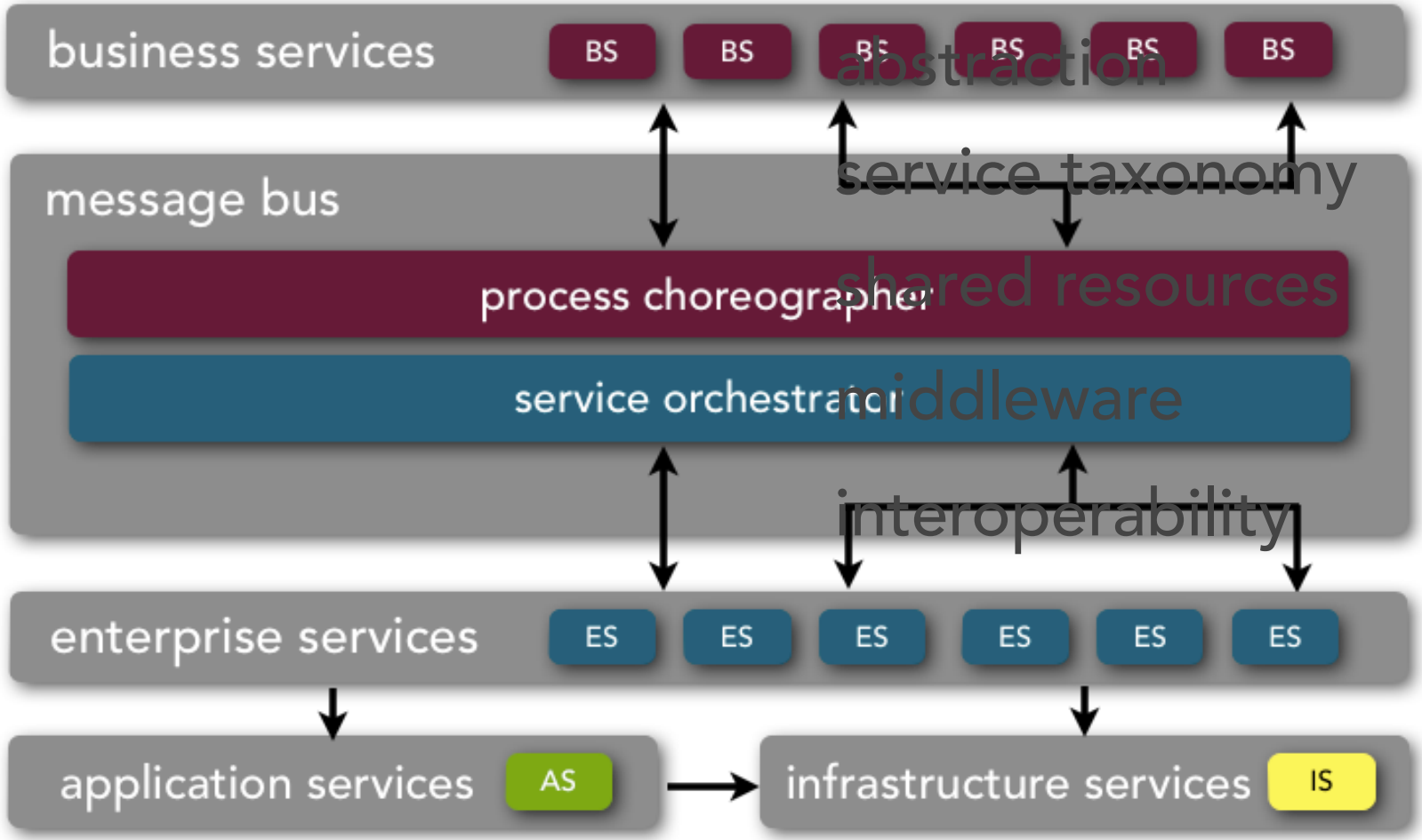
hub

intelligent hub

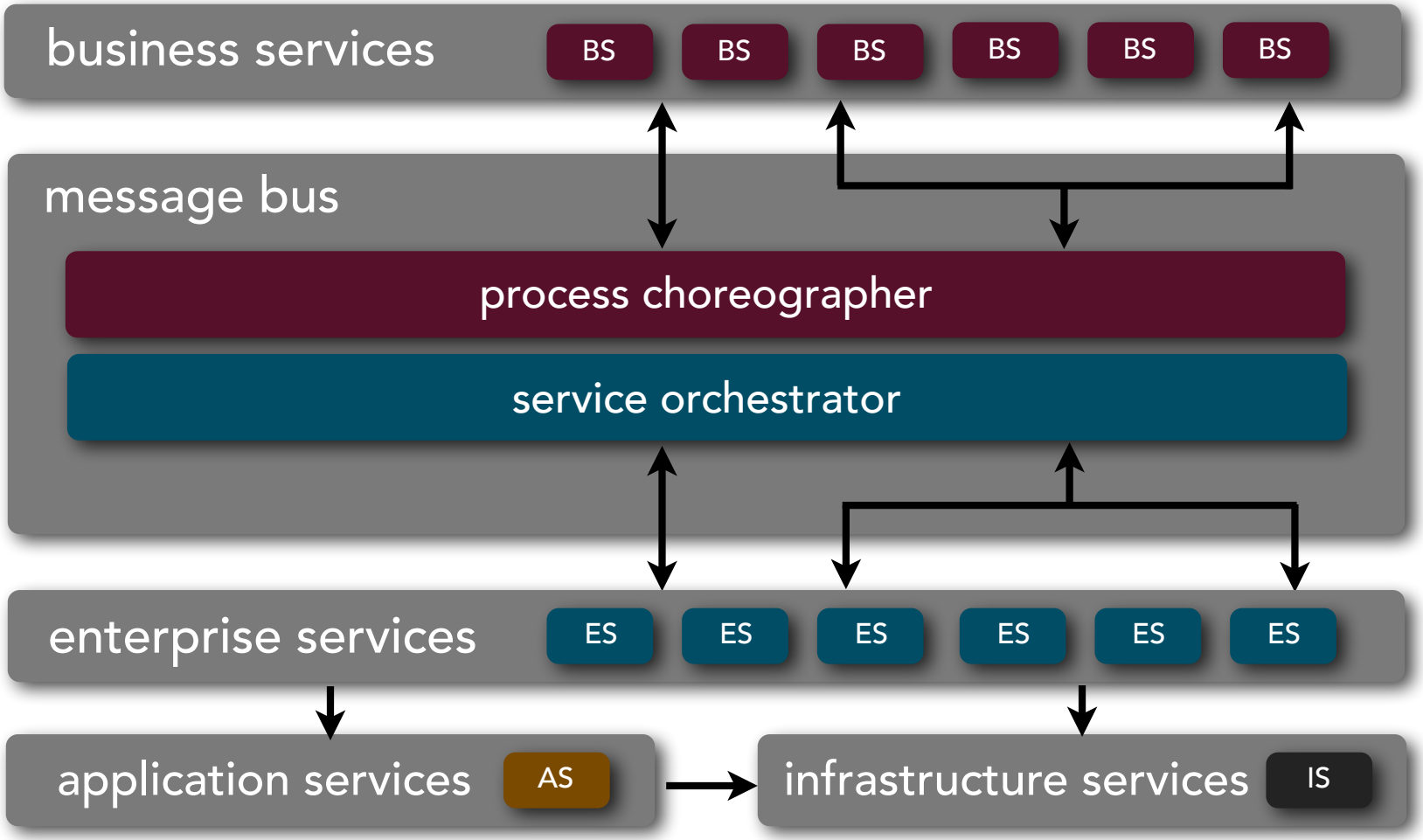
service oriented architecture /
enterprise service bus pattern



service-oriented architecture



service-oriented architecture



service-oriented architecture

abstract enterprise-level coarse-grained services owned and defined by business users

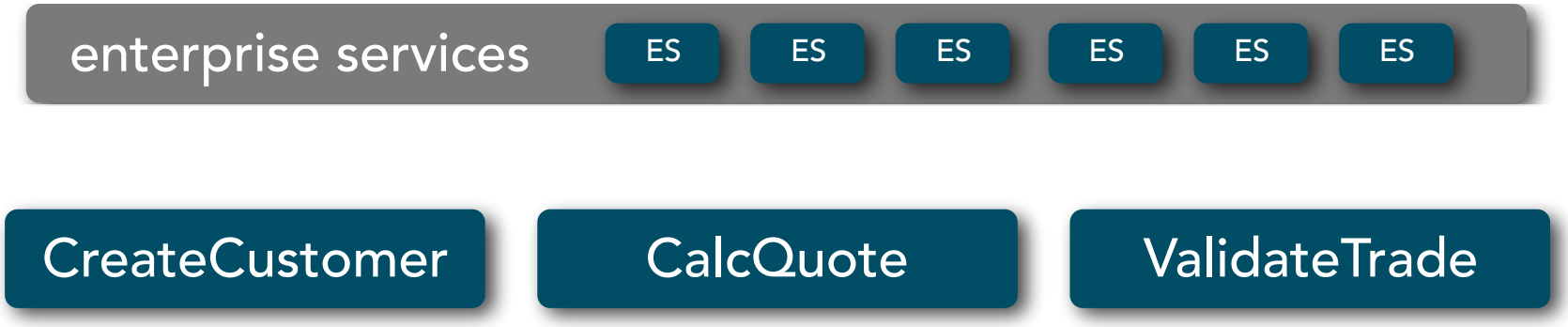
no implementation - only name, input, and output data represented as wsdl, bpel, xml, etc.

ExecuteTrade | PlaceOrder | ProcessClaim

service-oriented architecture

concrete enterprise-level coarse-grained services
owned by shared services teams

custom or vendor implementations that are one-to-one
or one-to-many relationship with business services



service-oriented architecture

concrete application-level fine-grained services
owned by application teams

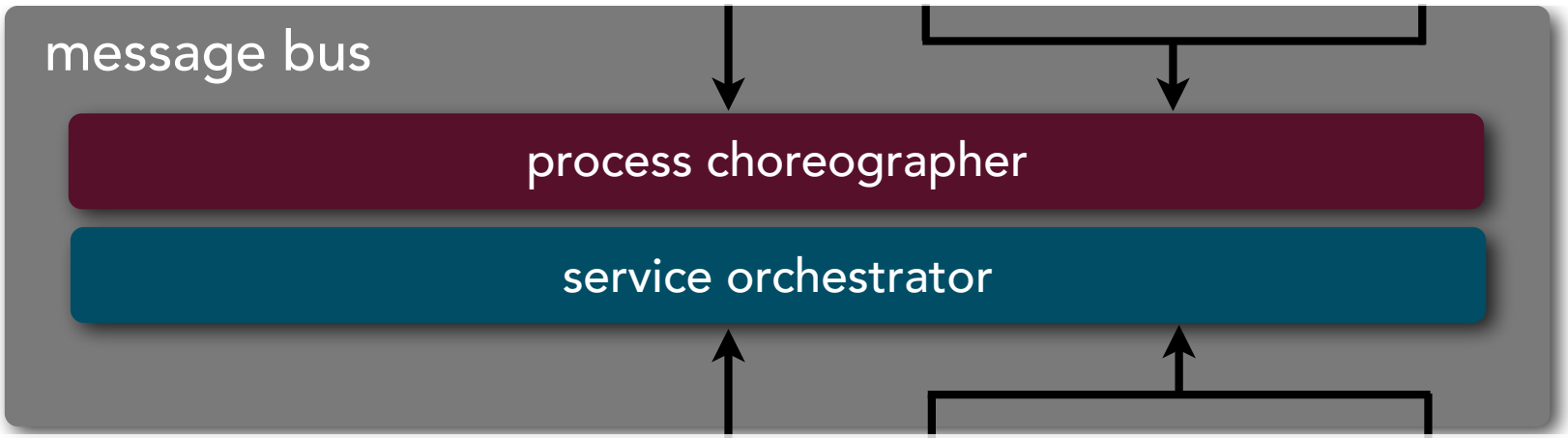
bound to a specific application context

service-oriented architecture

concrete enterprise-level fine-grained services owned by infrastructure or shared services teams

implements non-business functionality to support both enterprise and business services

service-oriented architecture



mediation and routing

process choreography

service orchestration

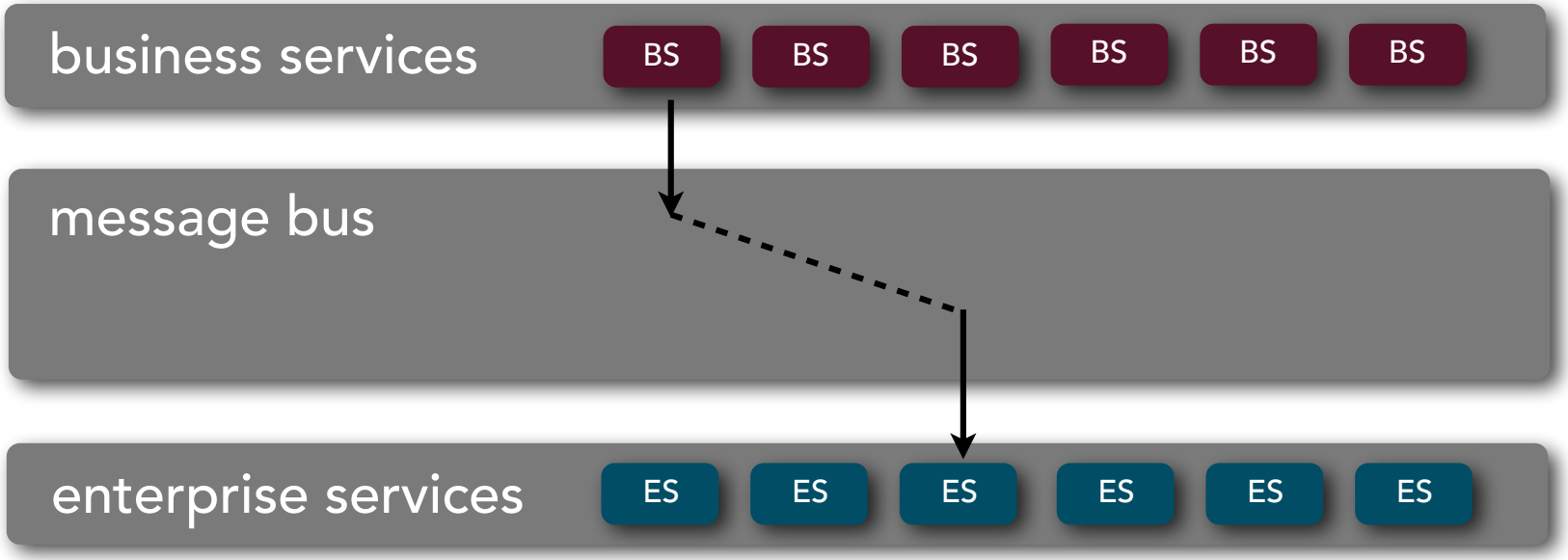
message enhancement

message transformation

protocol transformation

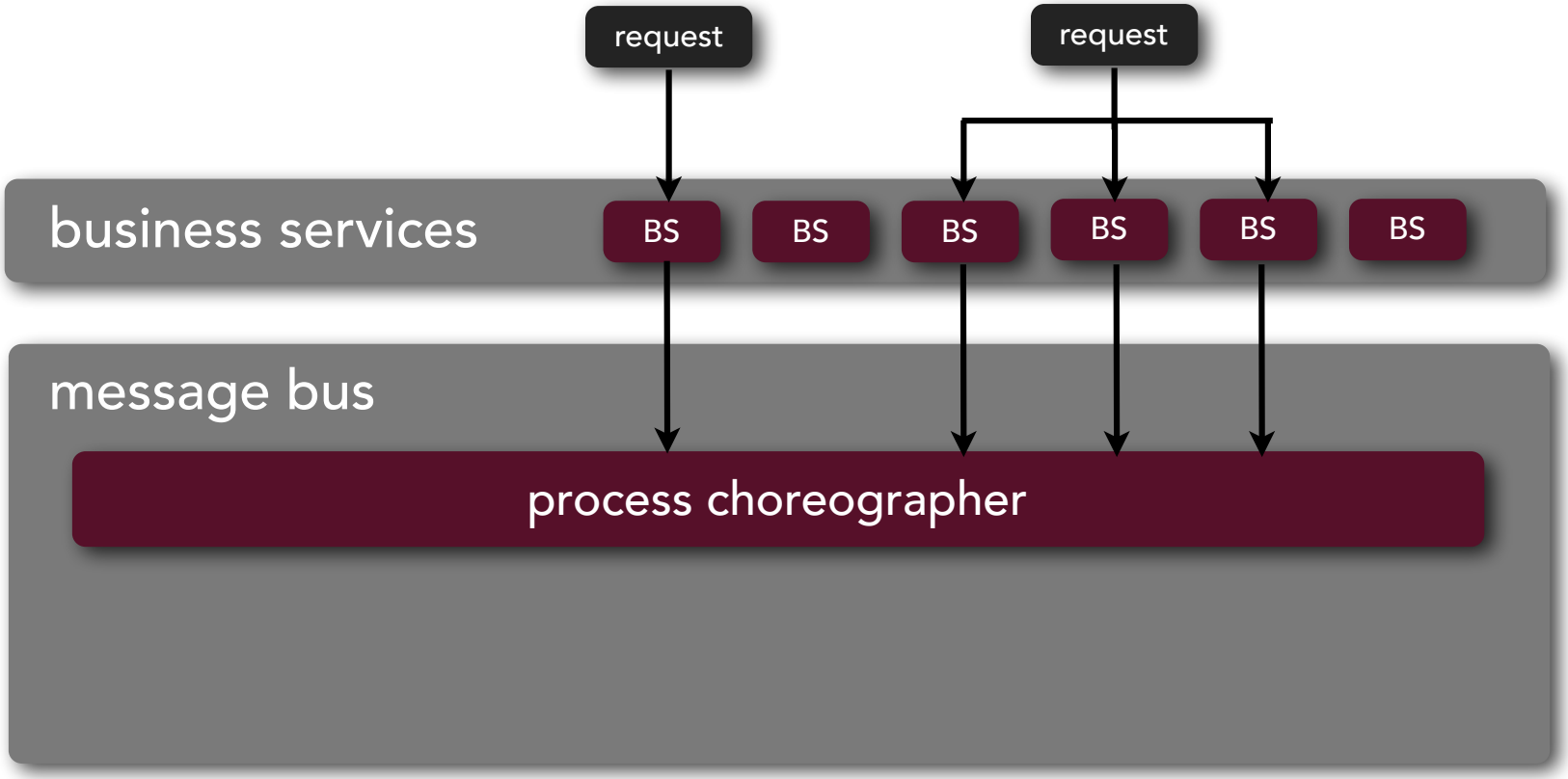
service-oriented architecture

mediation and routing



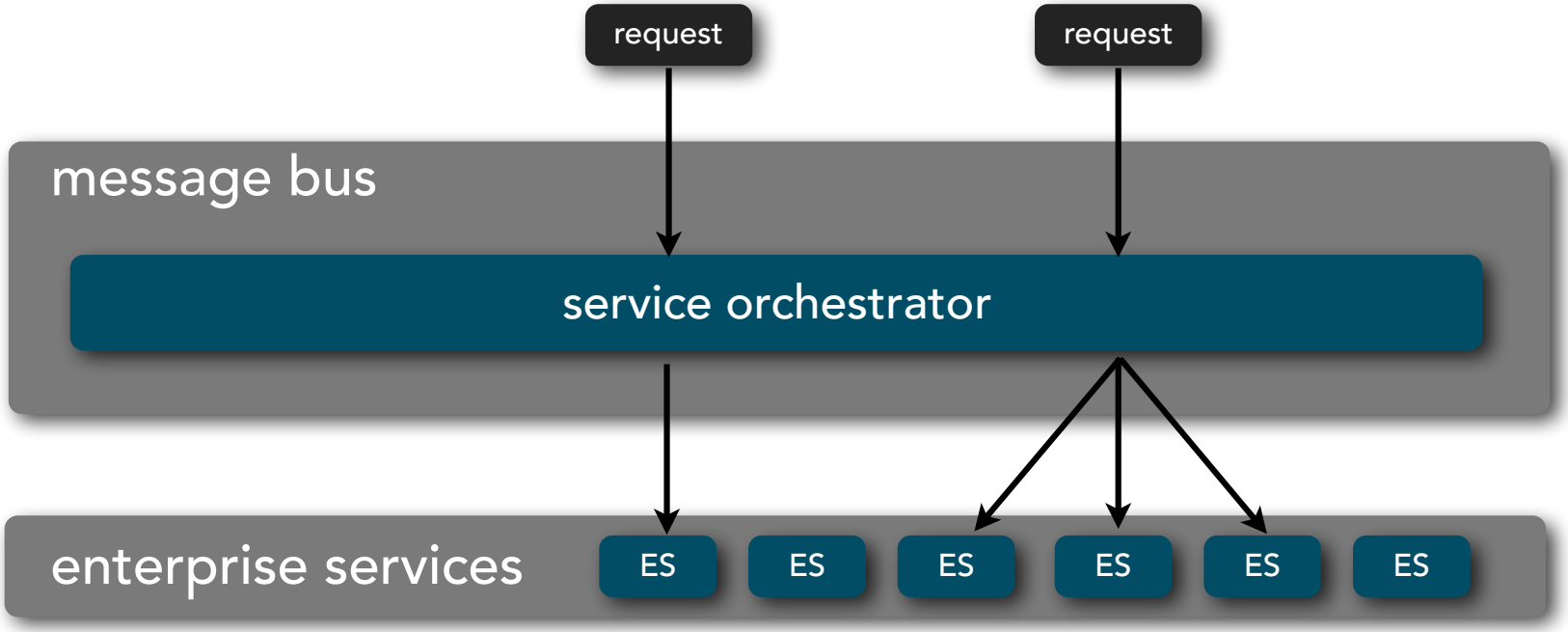
service-oriented architecture

process choreography



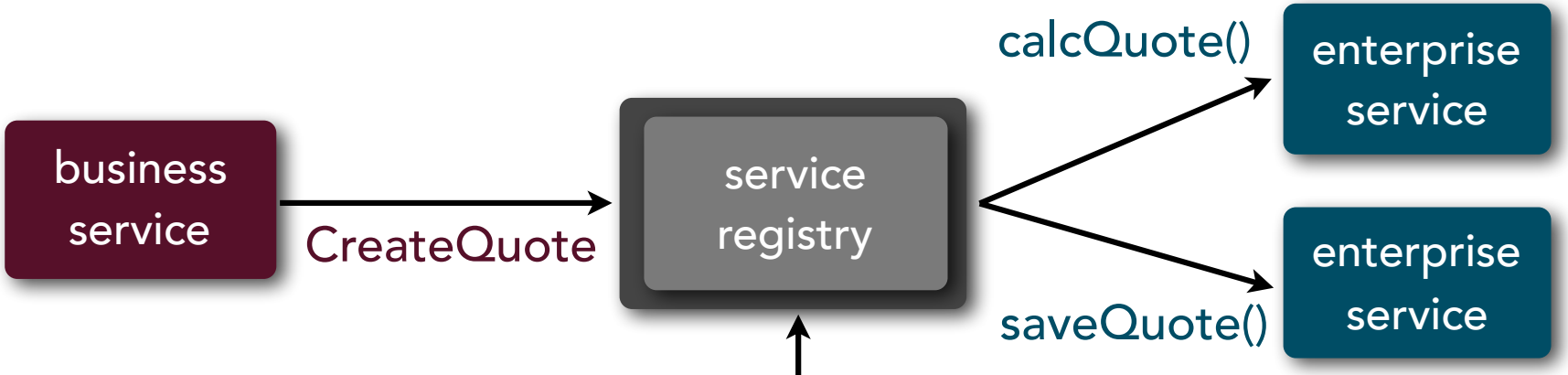
service-oriented architecture

service orchestration

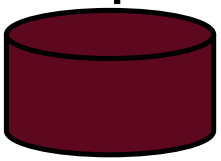


service-oriented architecture

service registry



location, name, and implementation transparency

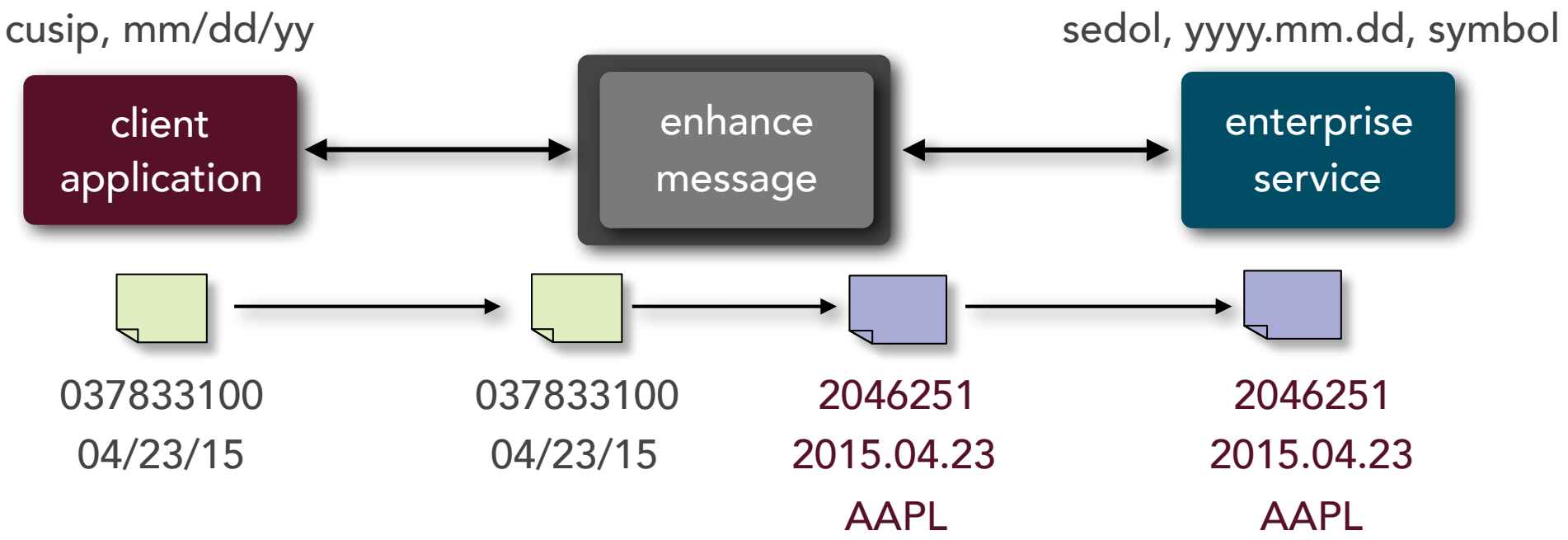


service registry

```
CreateQuote
  calcQuote
    https://grid1/CalcQuote
  saveQuote
    rmi://atlas:1299/saveQuote
```

service-oriented architecture

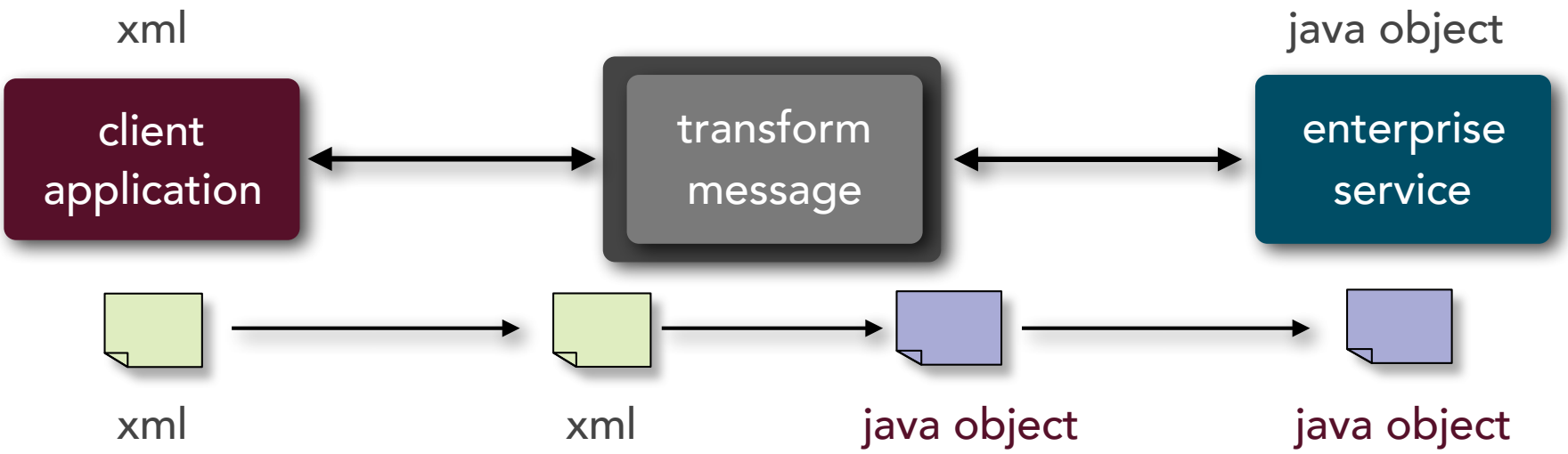
message enhancement



contract decoupling

service-oriented architecture

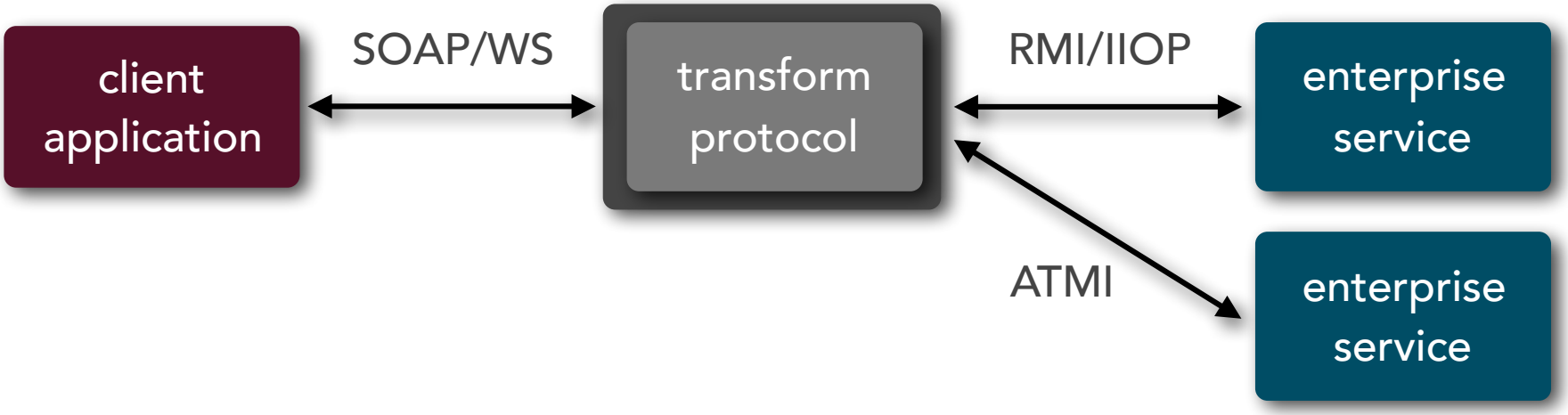
message transformation



contract decoupling

service-oriented architecture

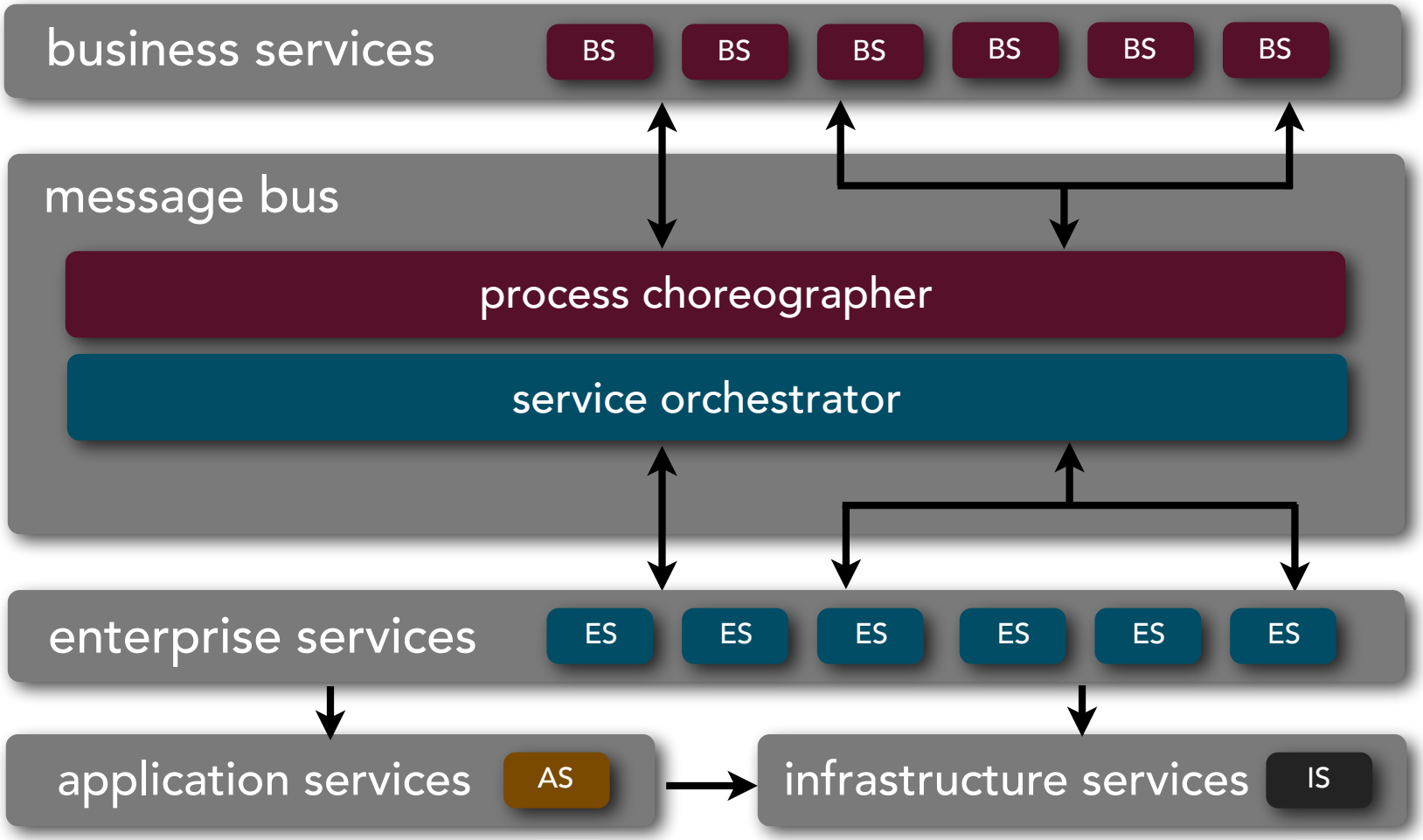
protocol transformation



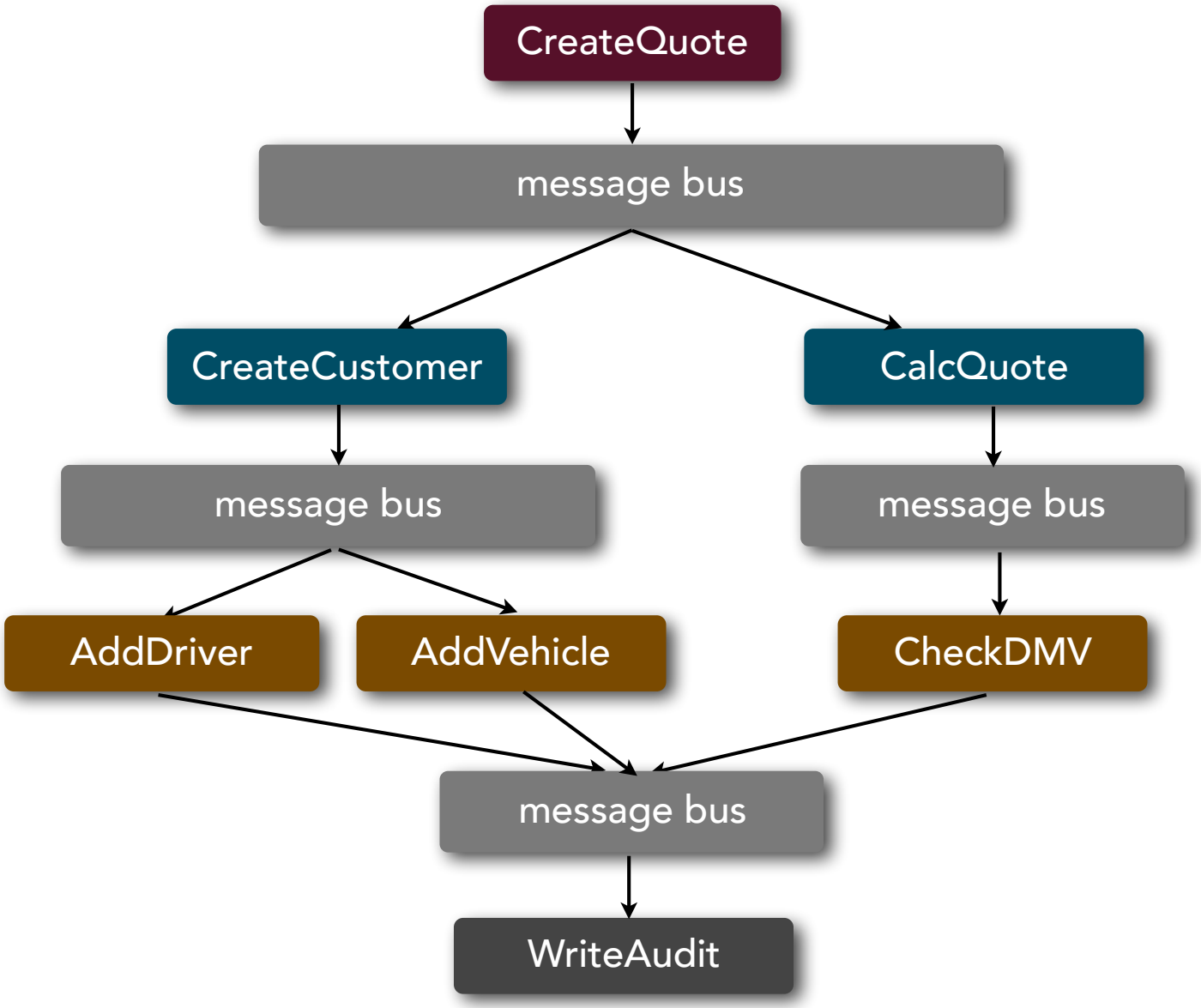
access decoupling

implementation transparency

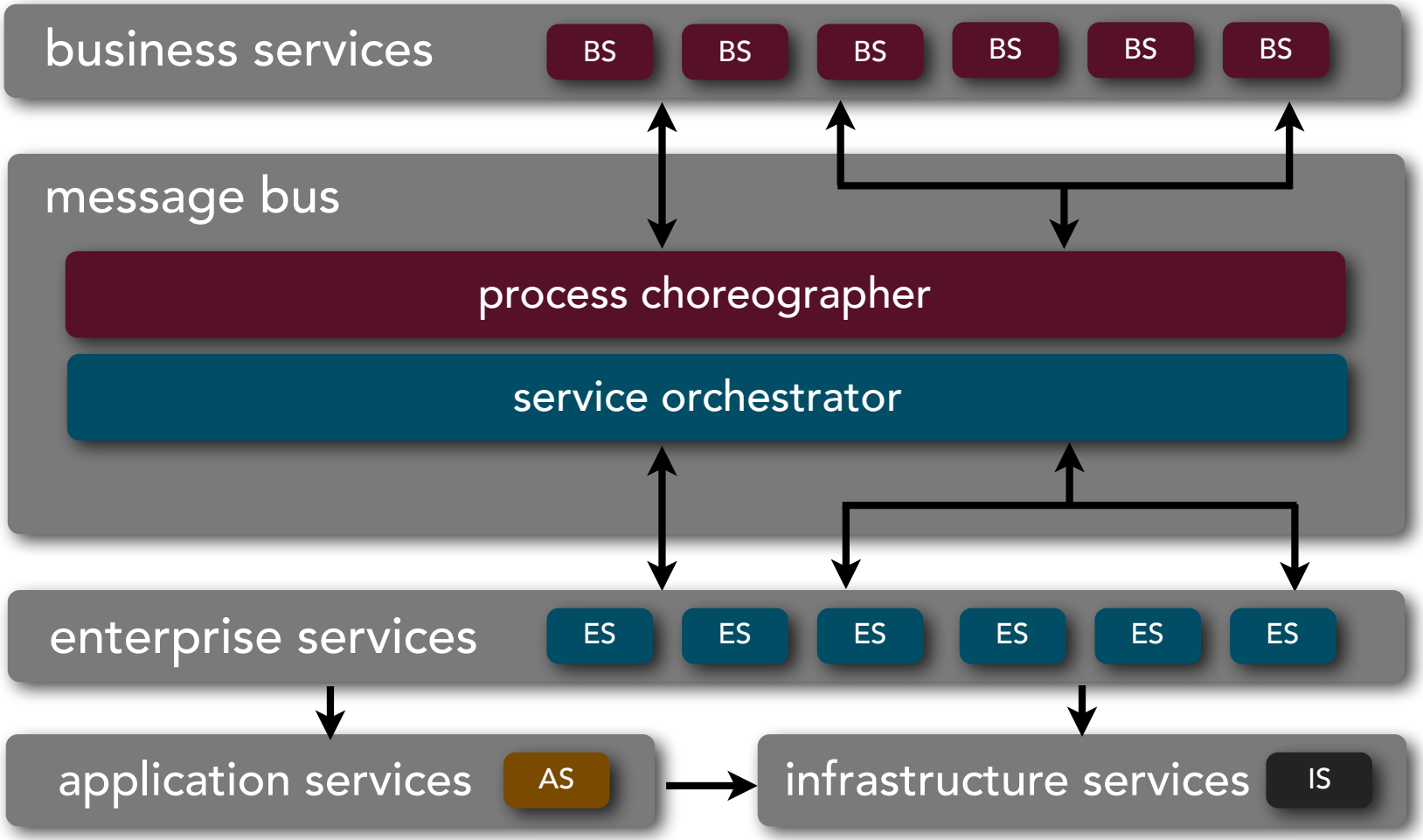
service-oriented architecture



service-oriented architecture

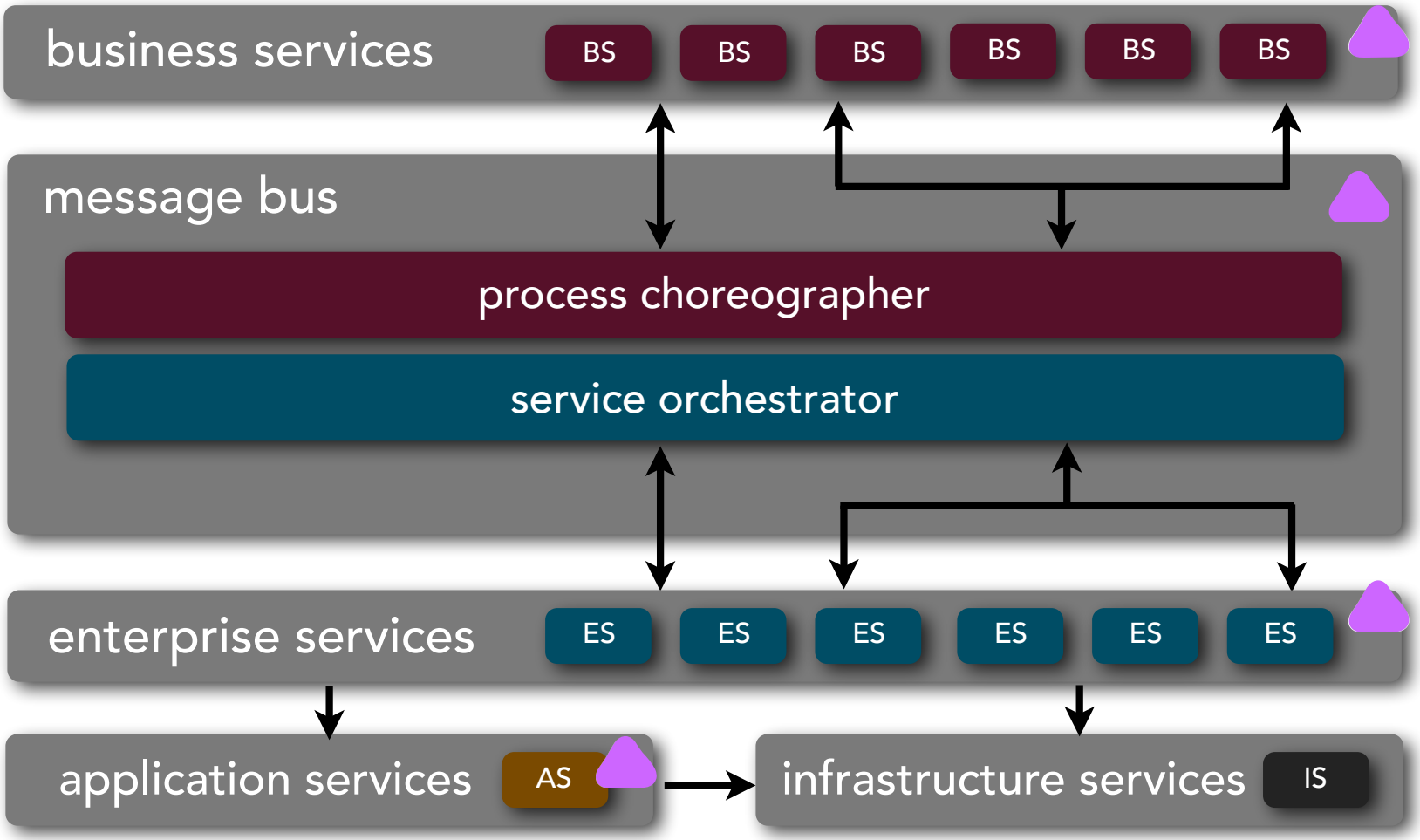


service-oriented architecture

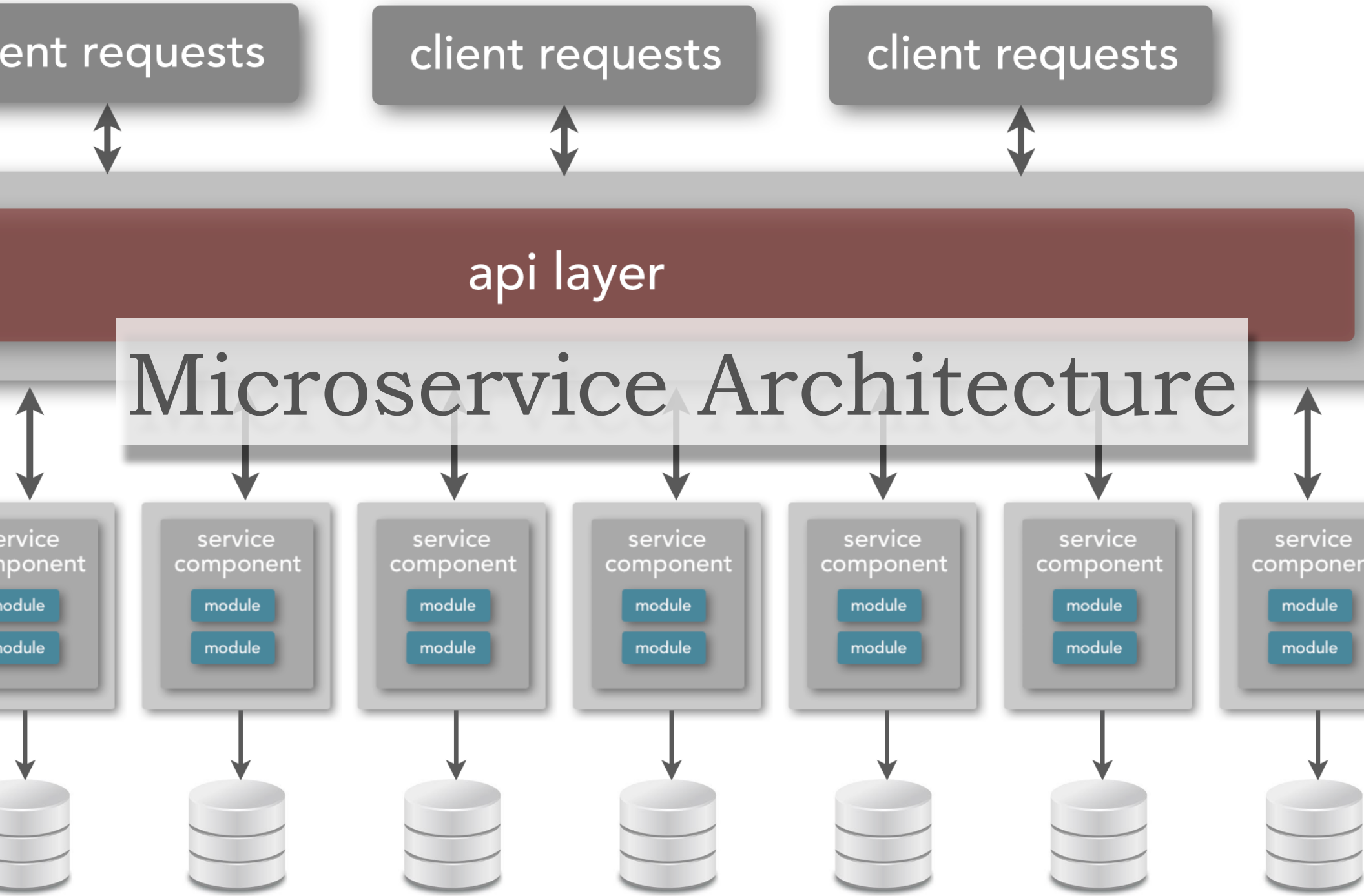


- maximize reuse
- maximize canonicality

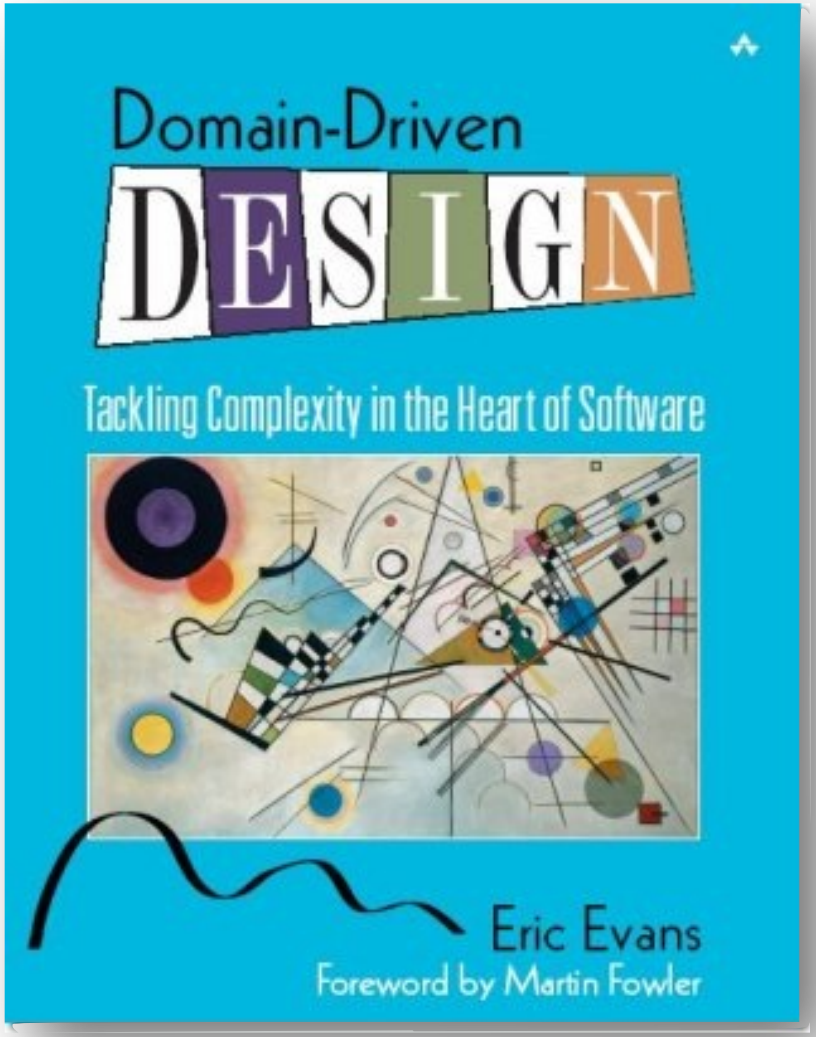
service-oriented architecture

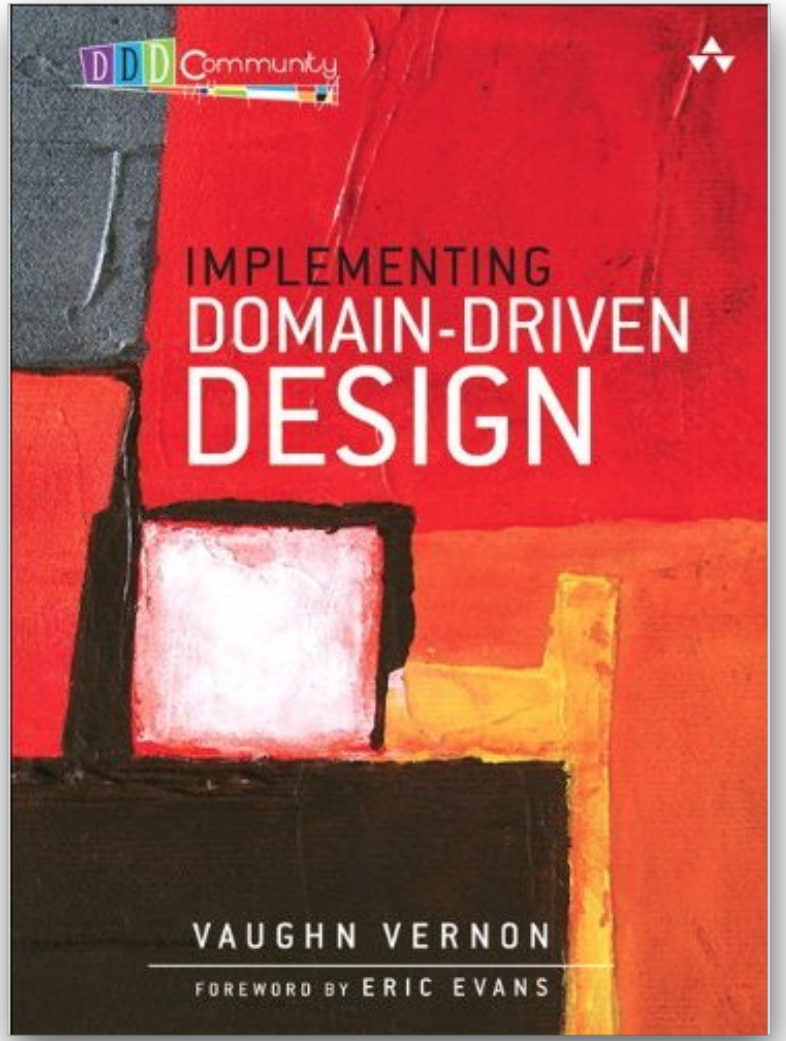


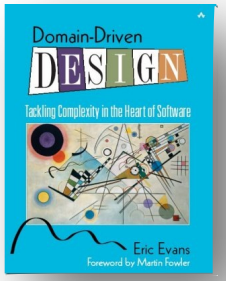
- incremental change
- operationally coupled


ent requests
client requests
client requests
api layer
Microservice Architecture
service component
module
module

Domain Driven Design

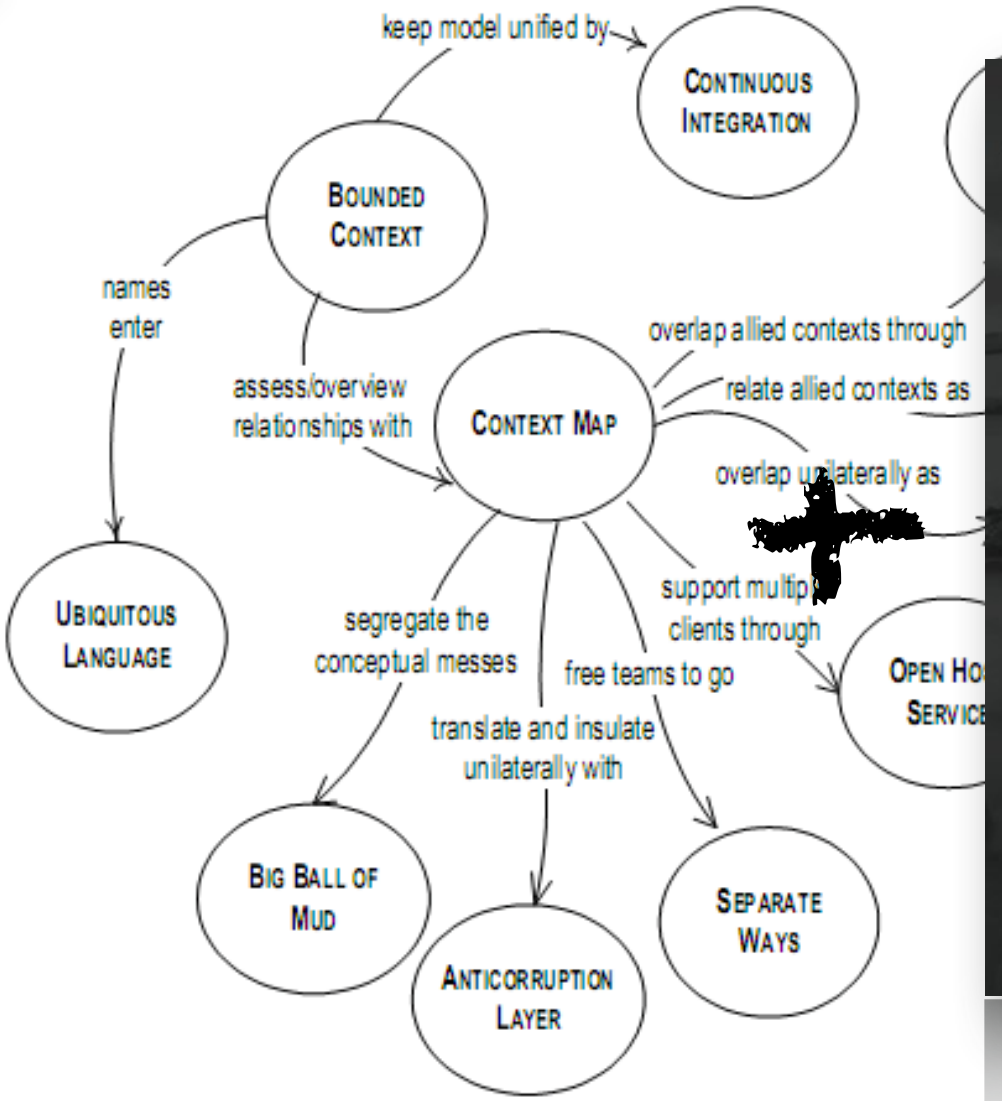


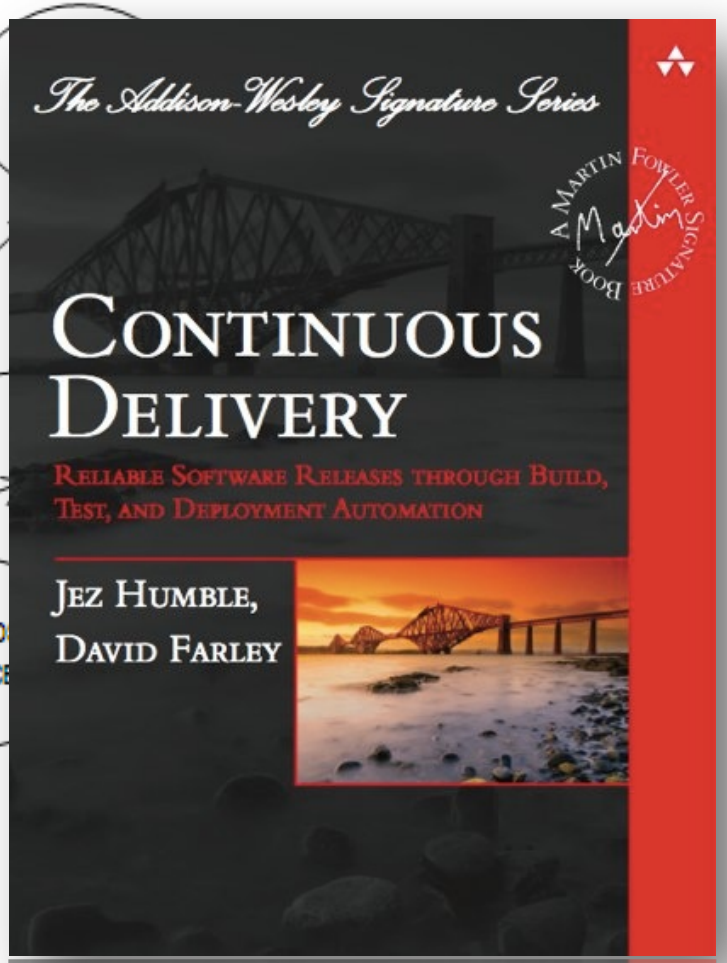




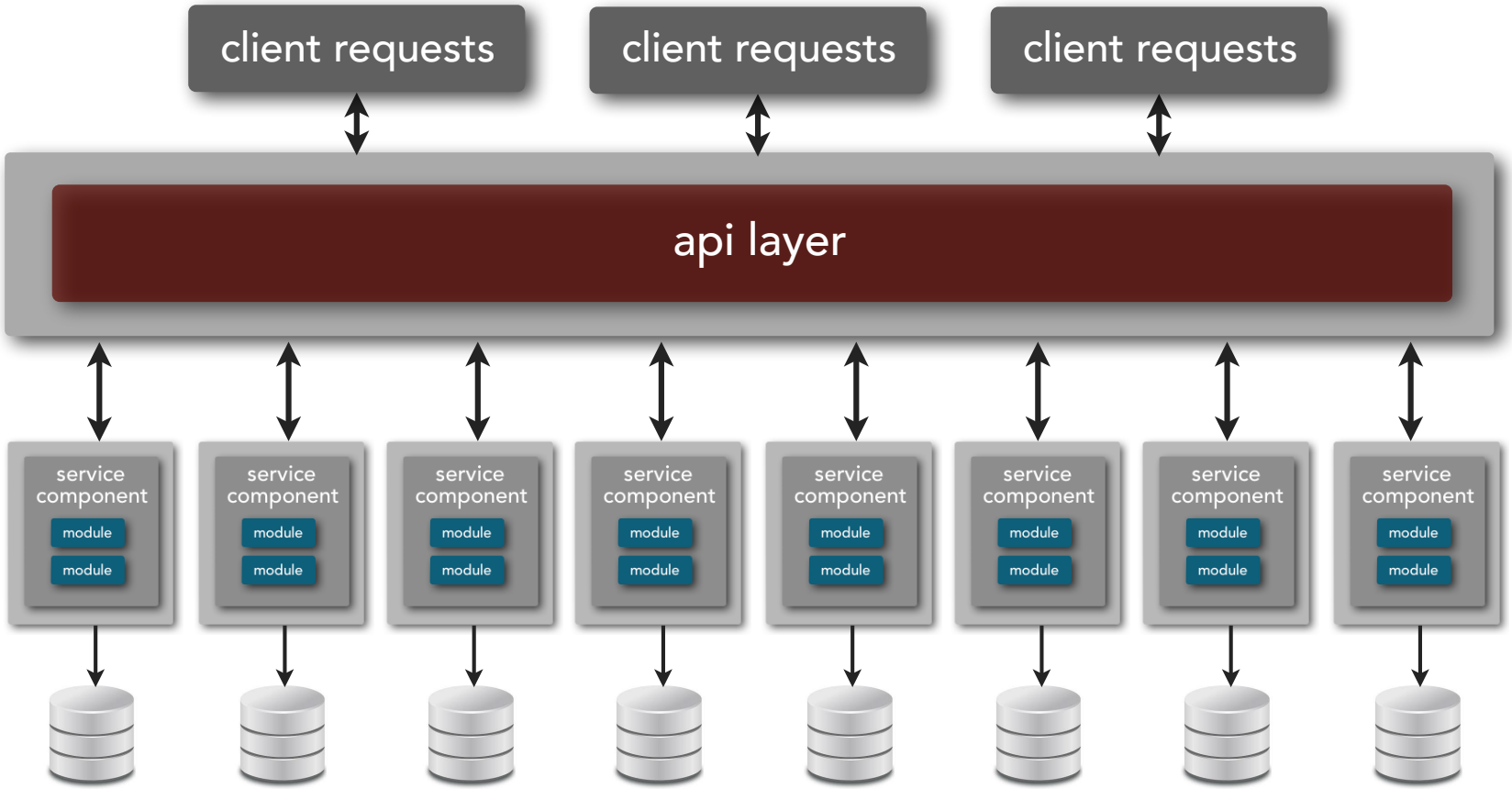
Bounded Context

Maintaining Model Integrity



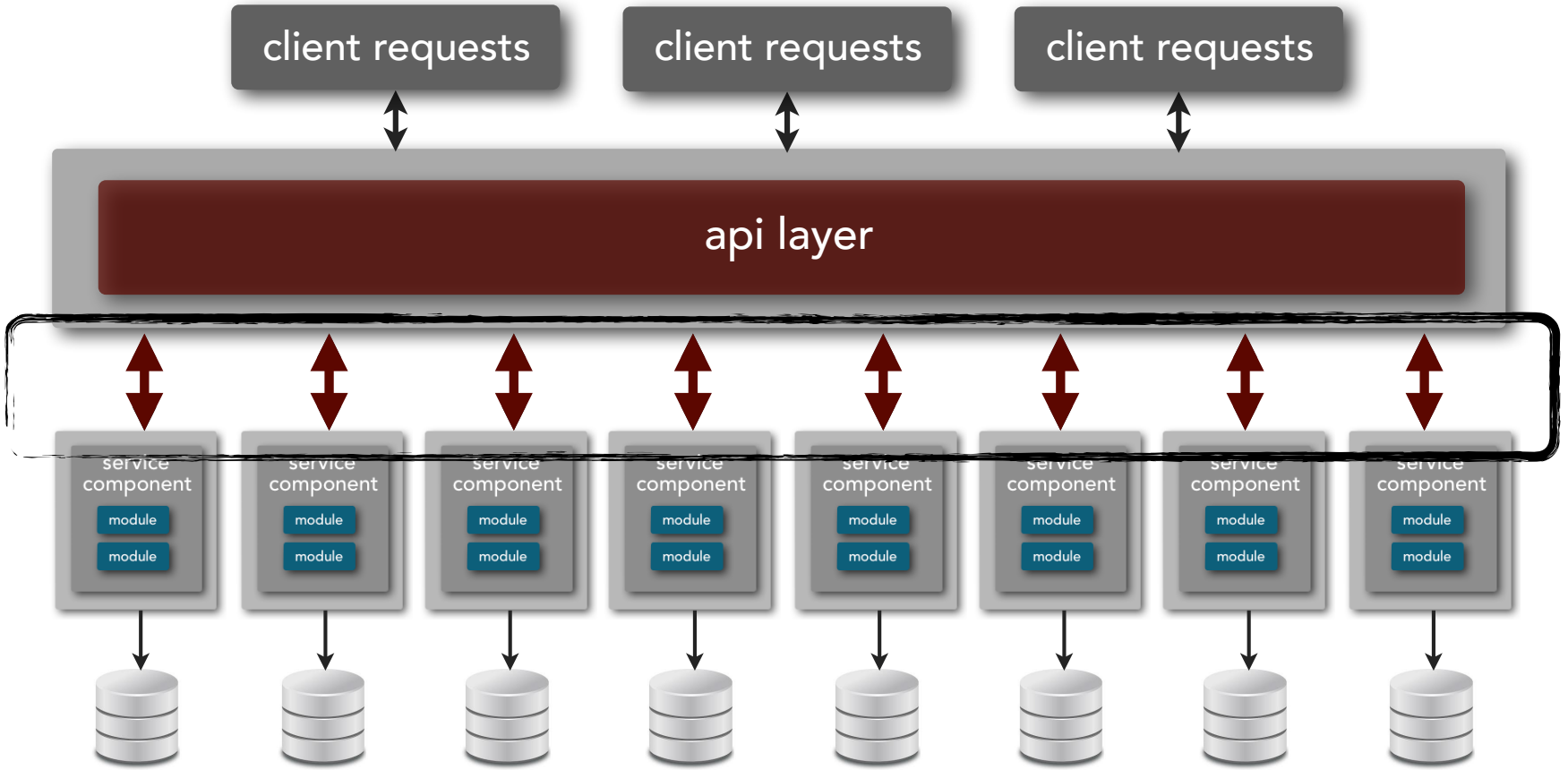


microservices architecture



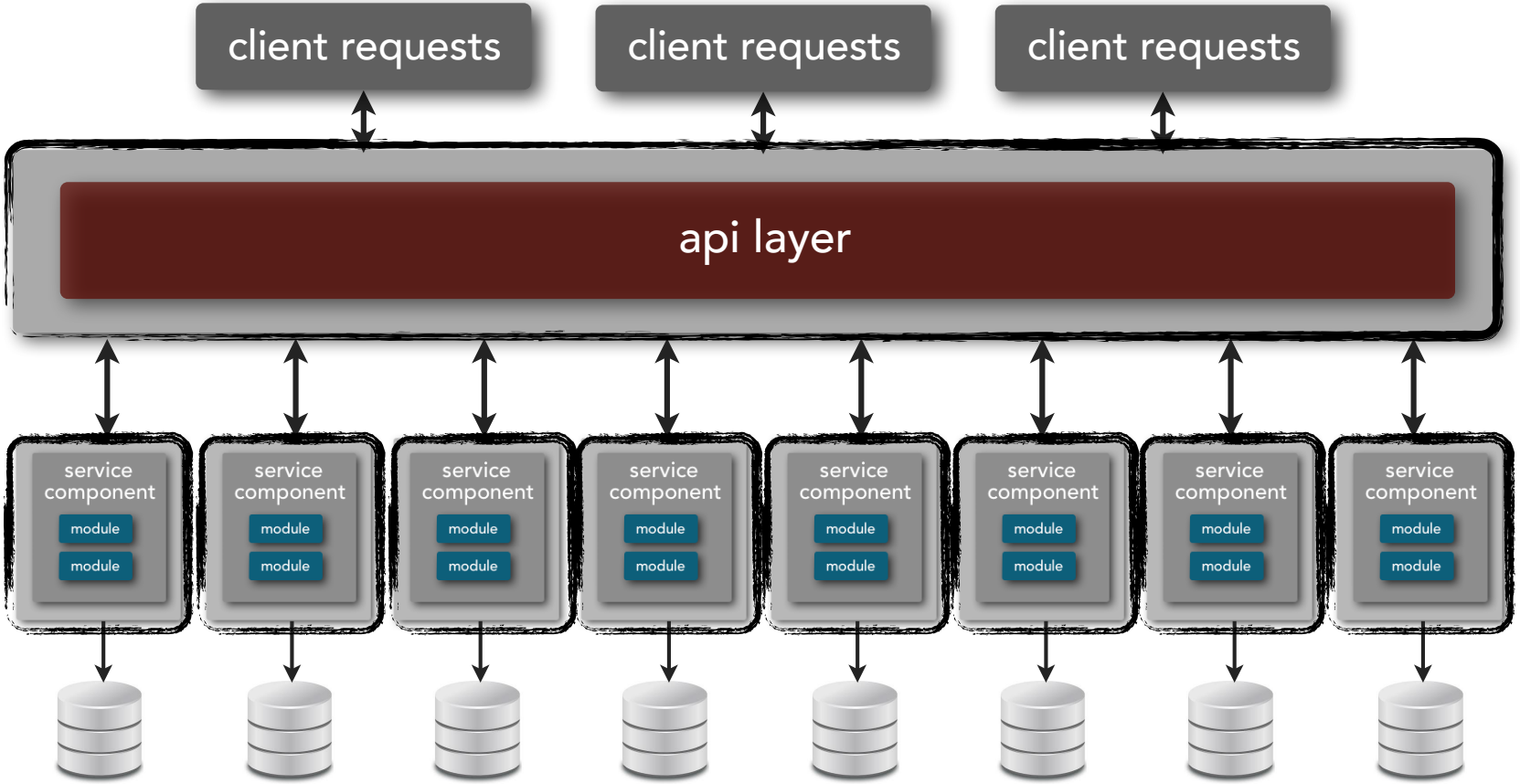
microservices architecture

distributed architecture



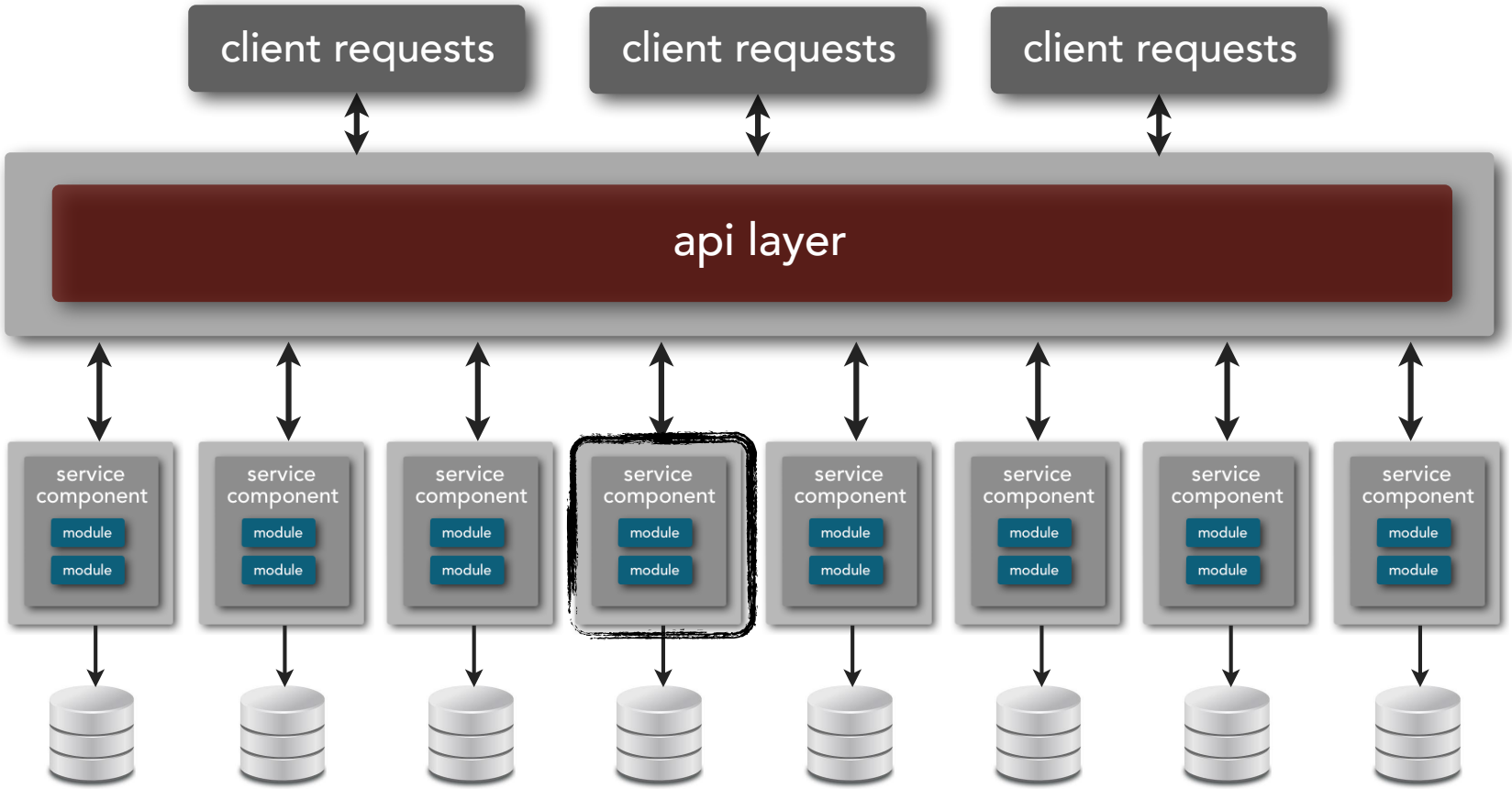
microservices architecture

separately deployed components



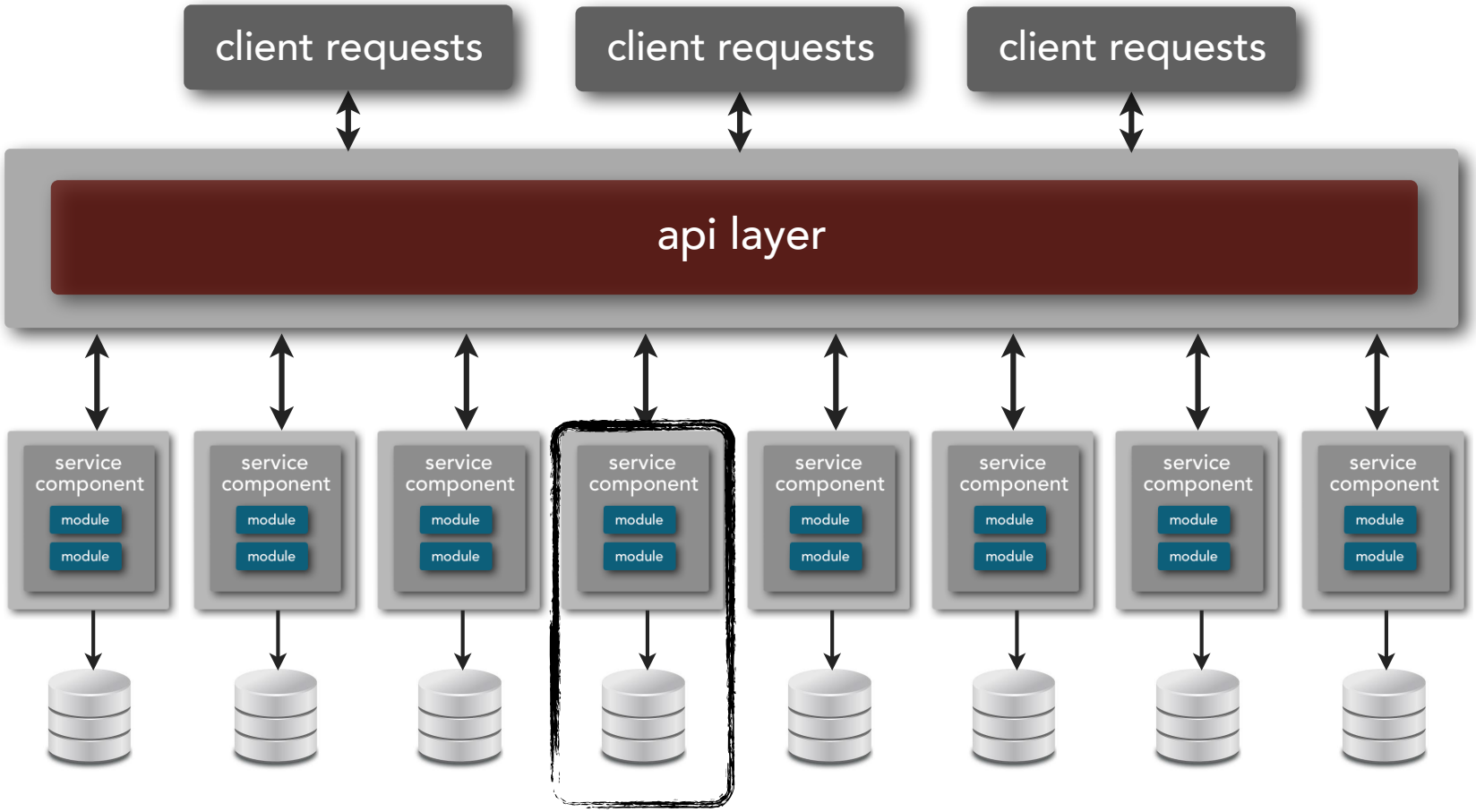
microservices architecture

service component

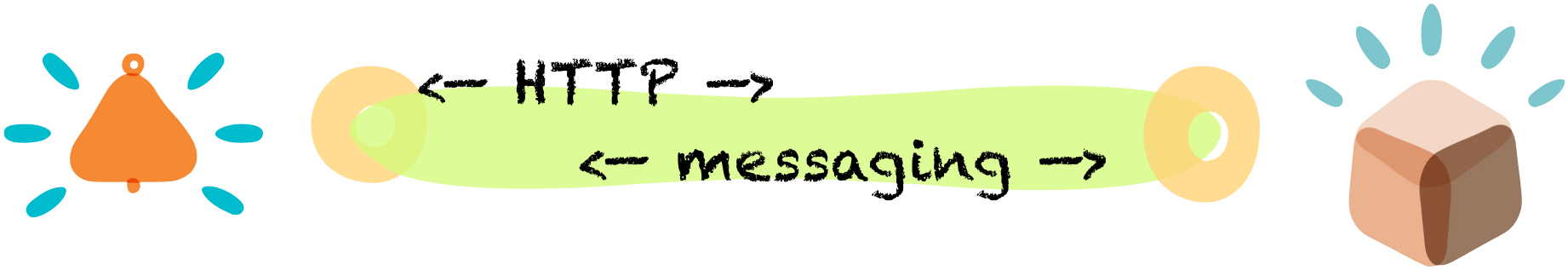


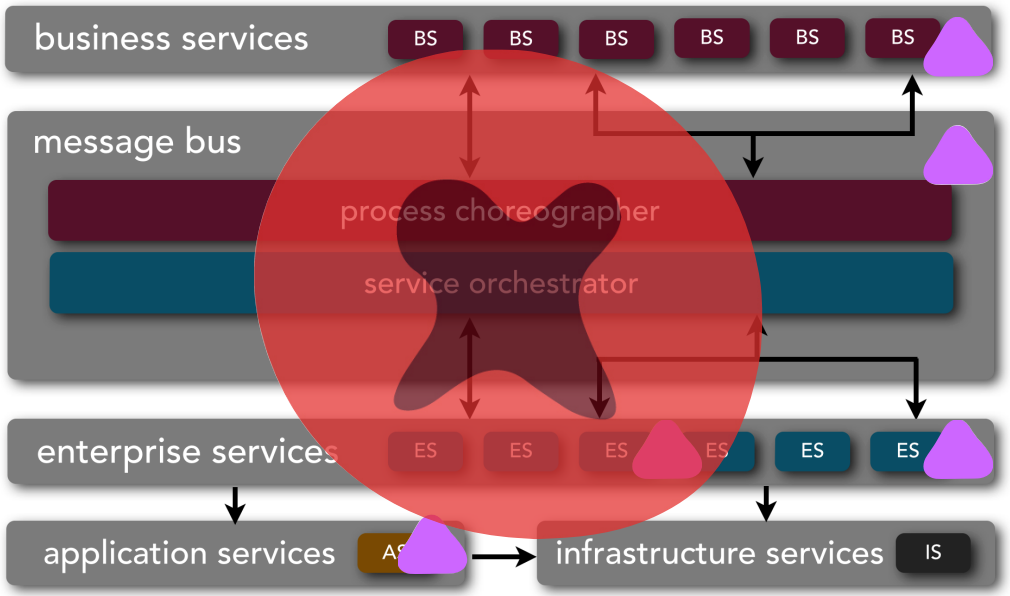
microservices architecture

bounded context

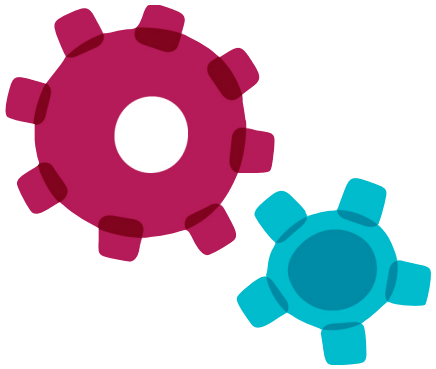


Smart Endpoints, Dumb Pipes





Standardize on integration, not platform



embrace polyglot solutions where sensible

too few languages/platforms

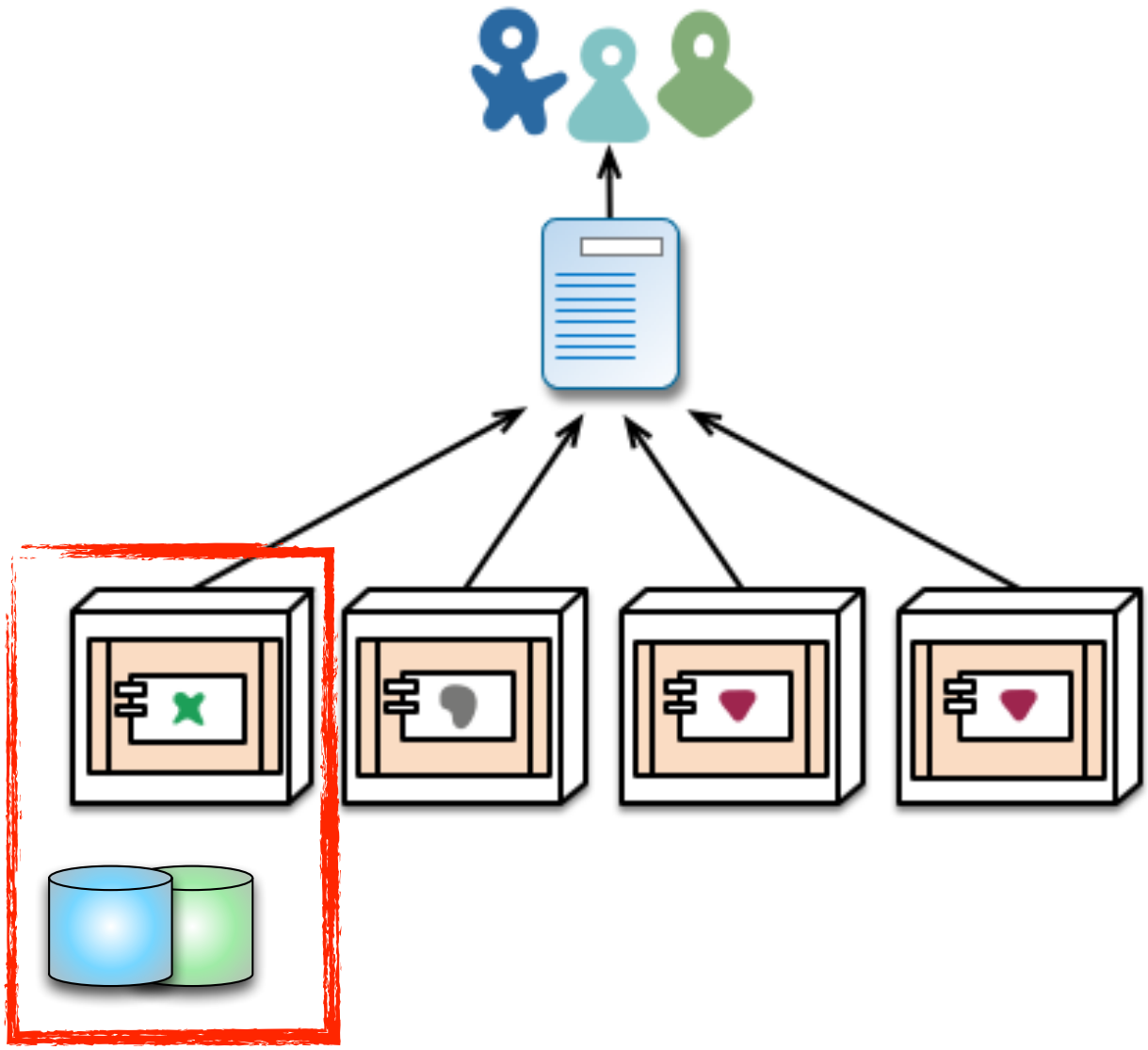
too many languages/platforms

Have one, two or maybe three ways of integrating, not 20.

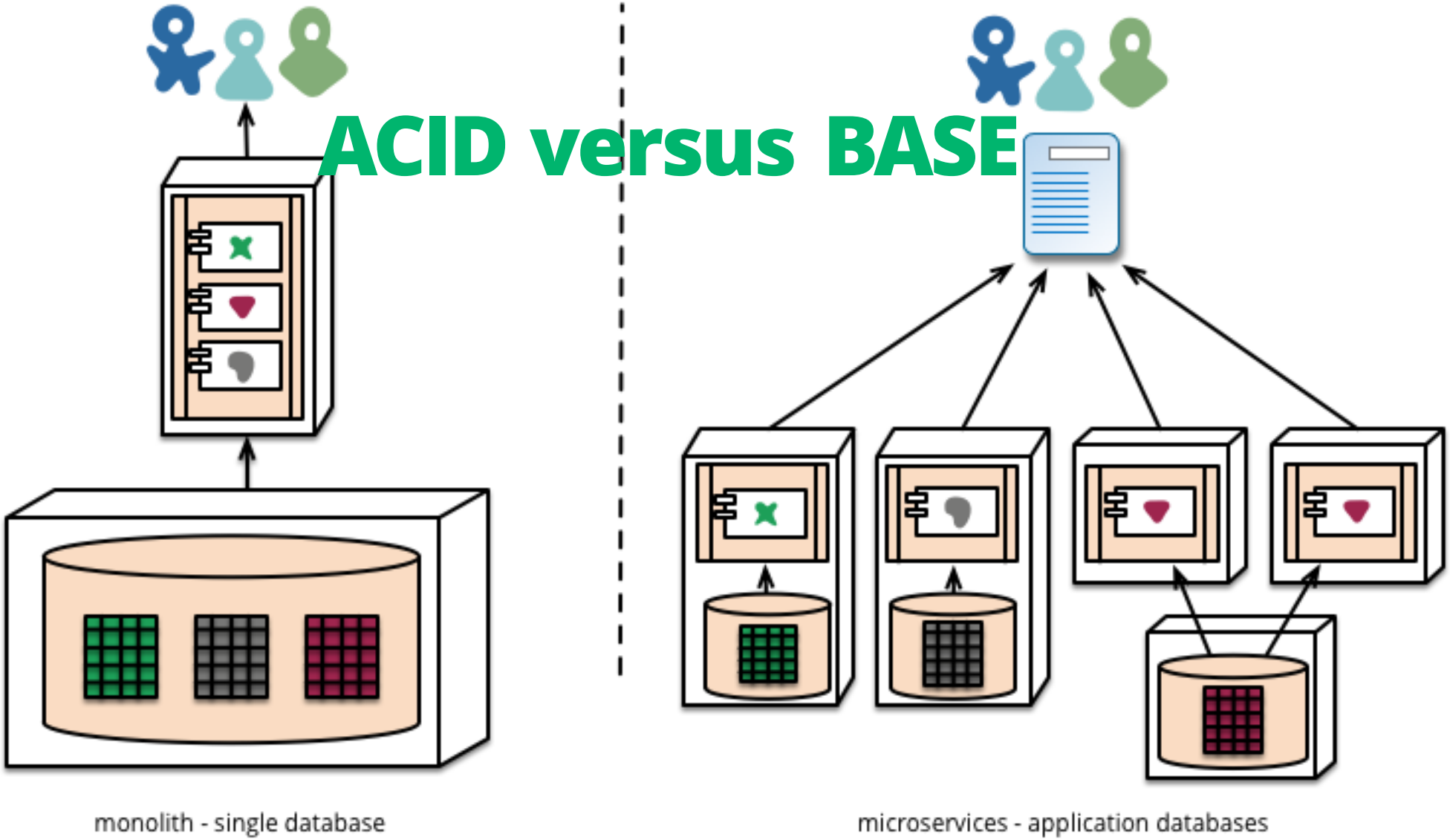
Standardize in the gaps between services - be flexible about what happens inside the boxes

Pick some sensible conventions, and stick with them.

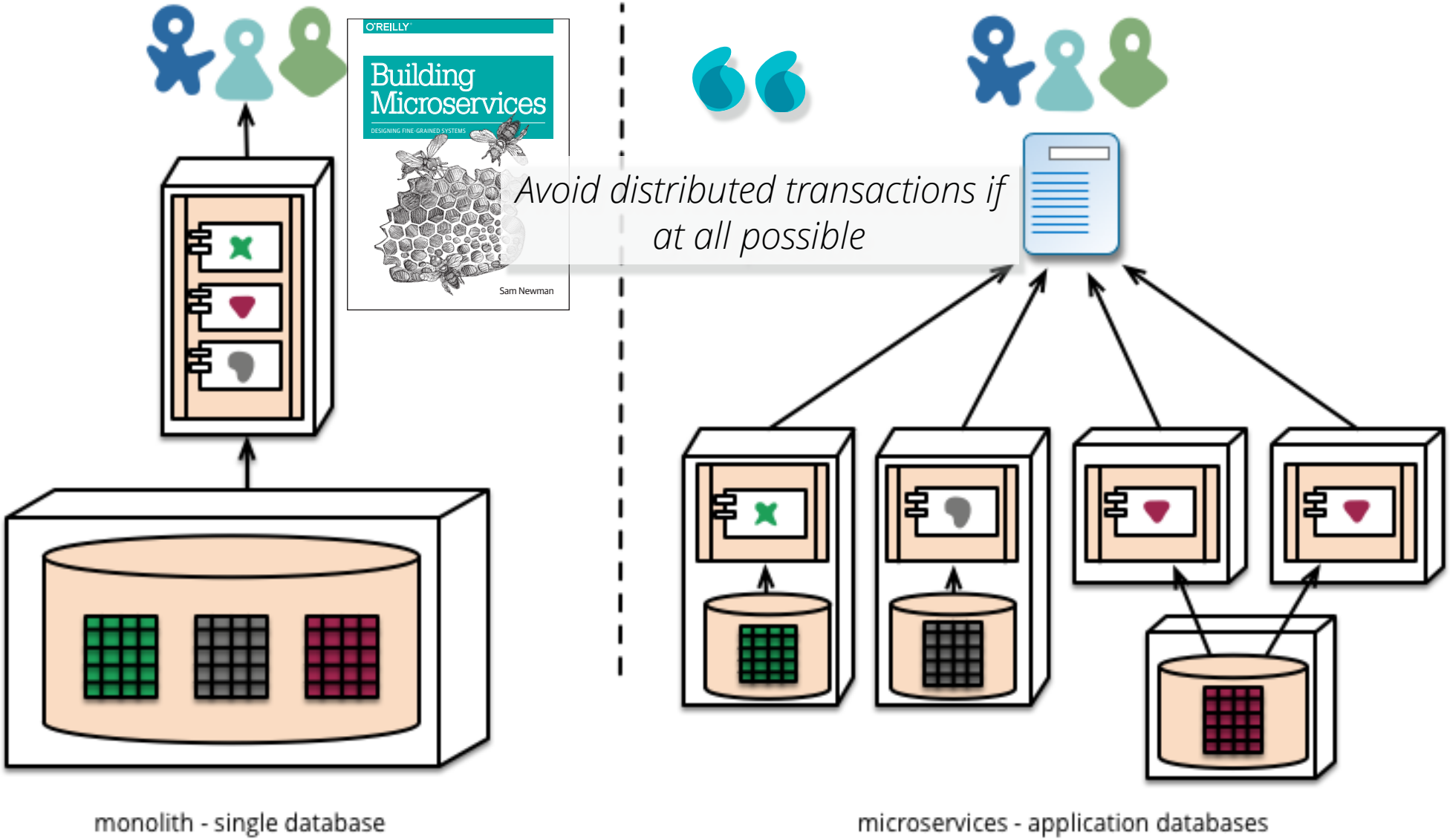
Decentralized Governance



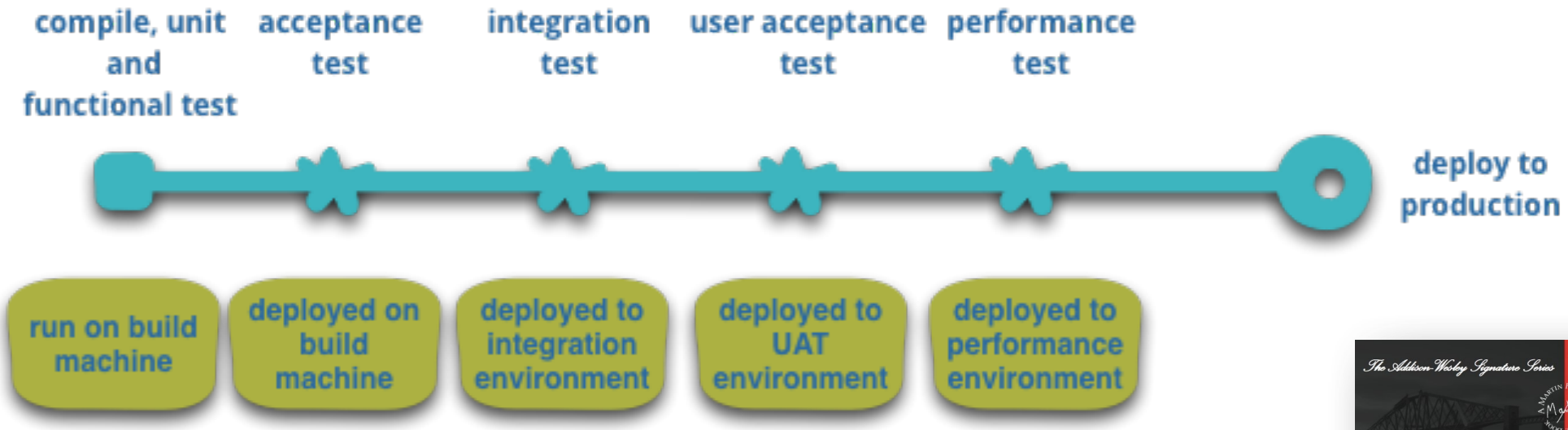
Decentralized Data Management

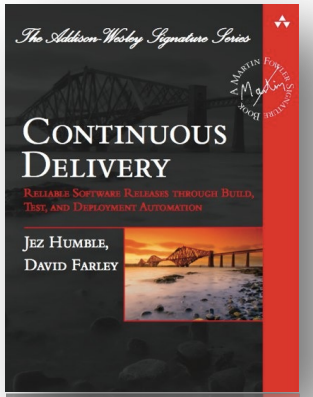


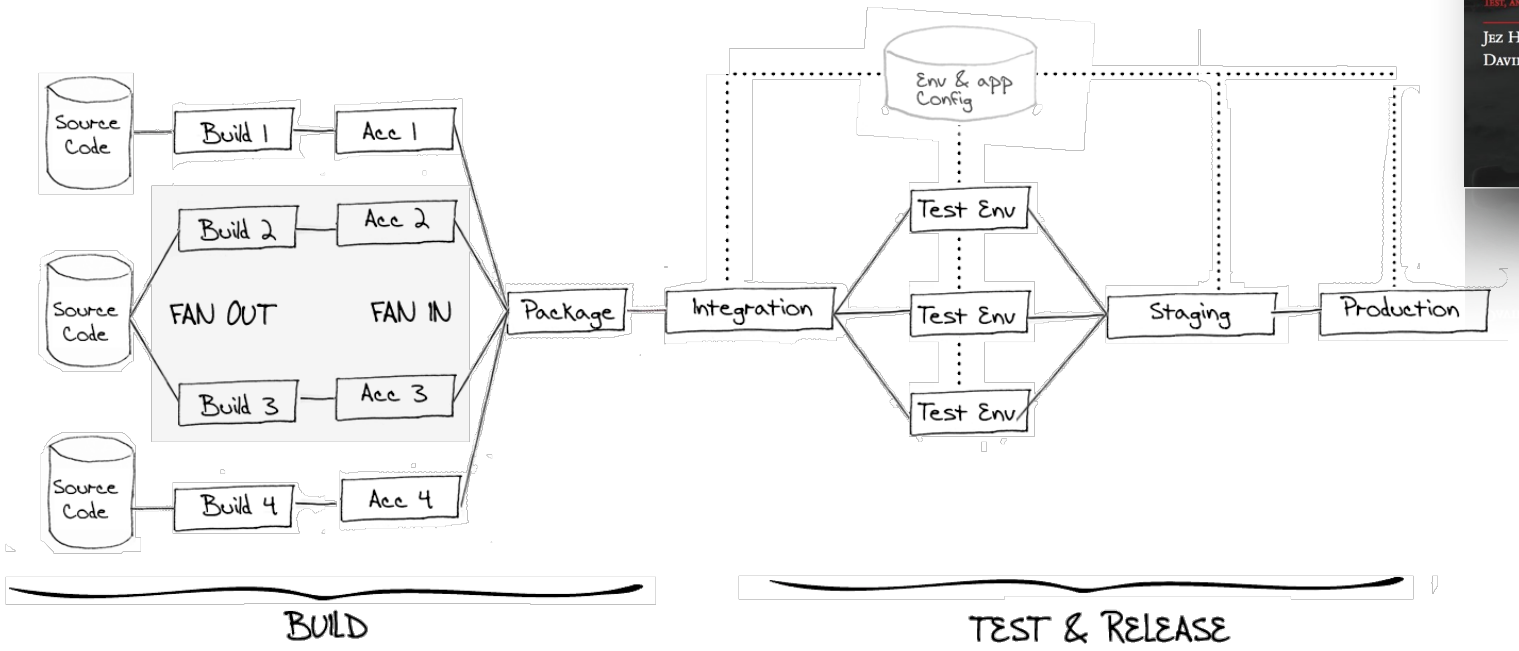
Decentralized Data Management



Infrastructure Automation

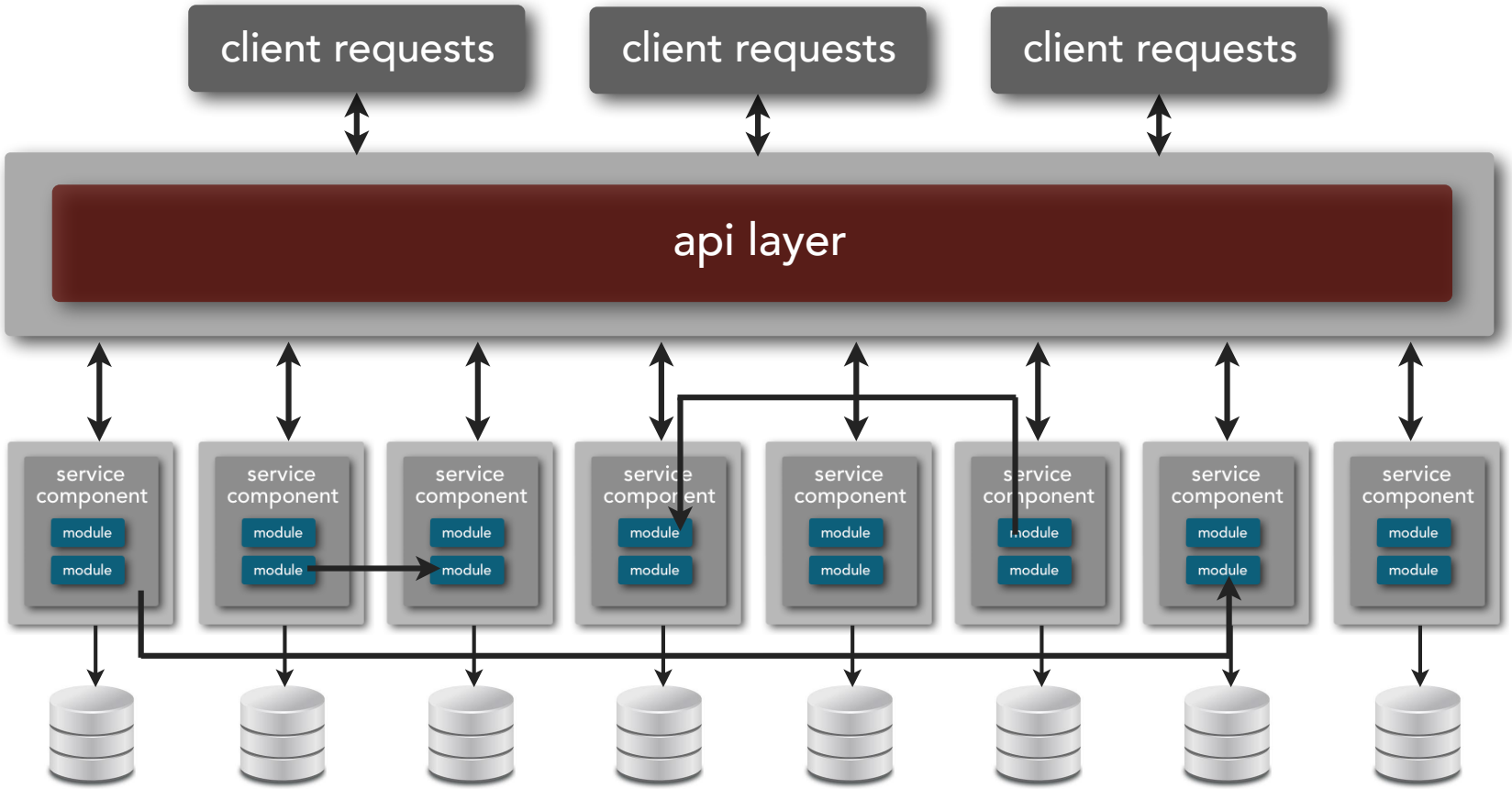






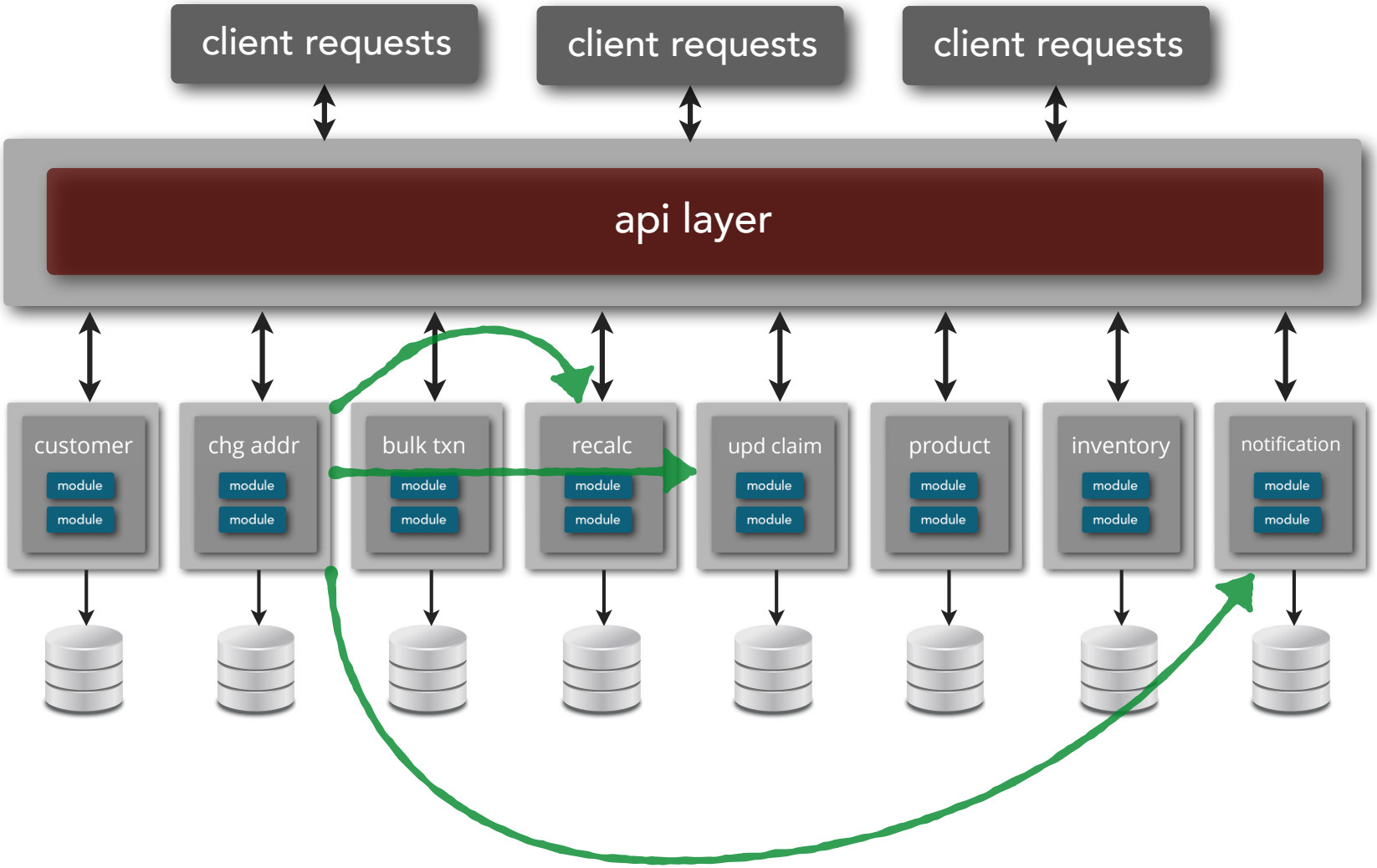
microservices architecture

service orchestration



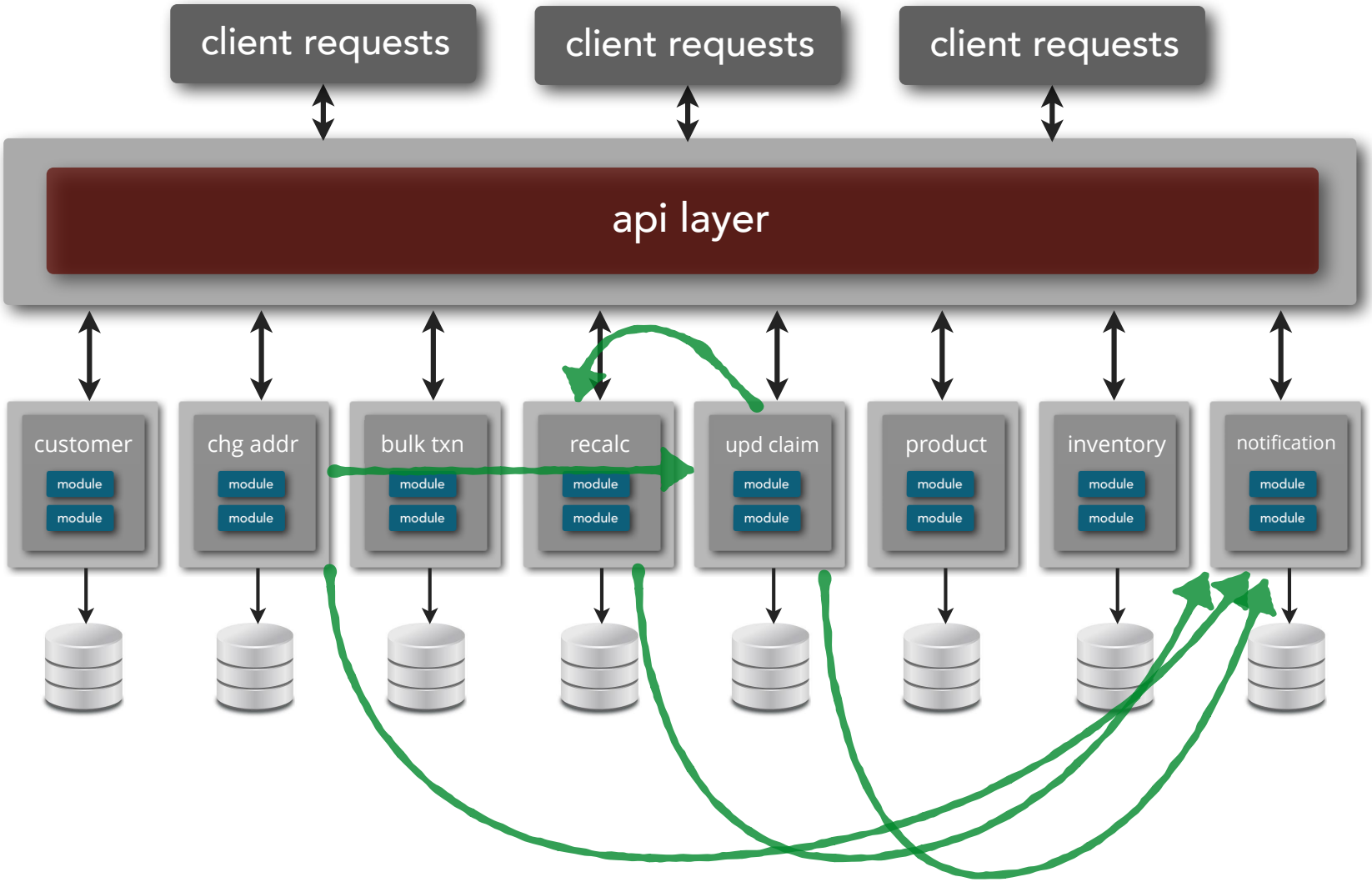
microservices architecture

orchestration

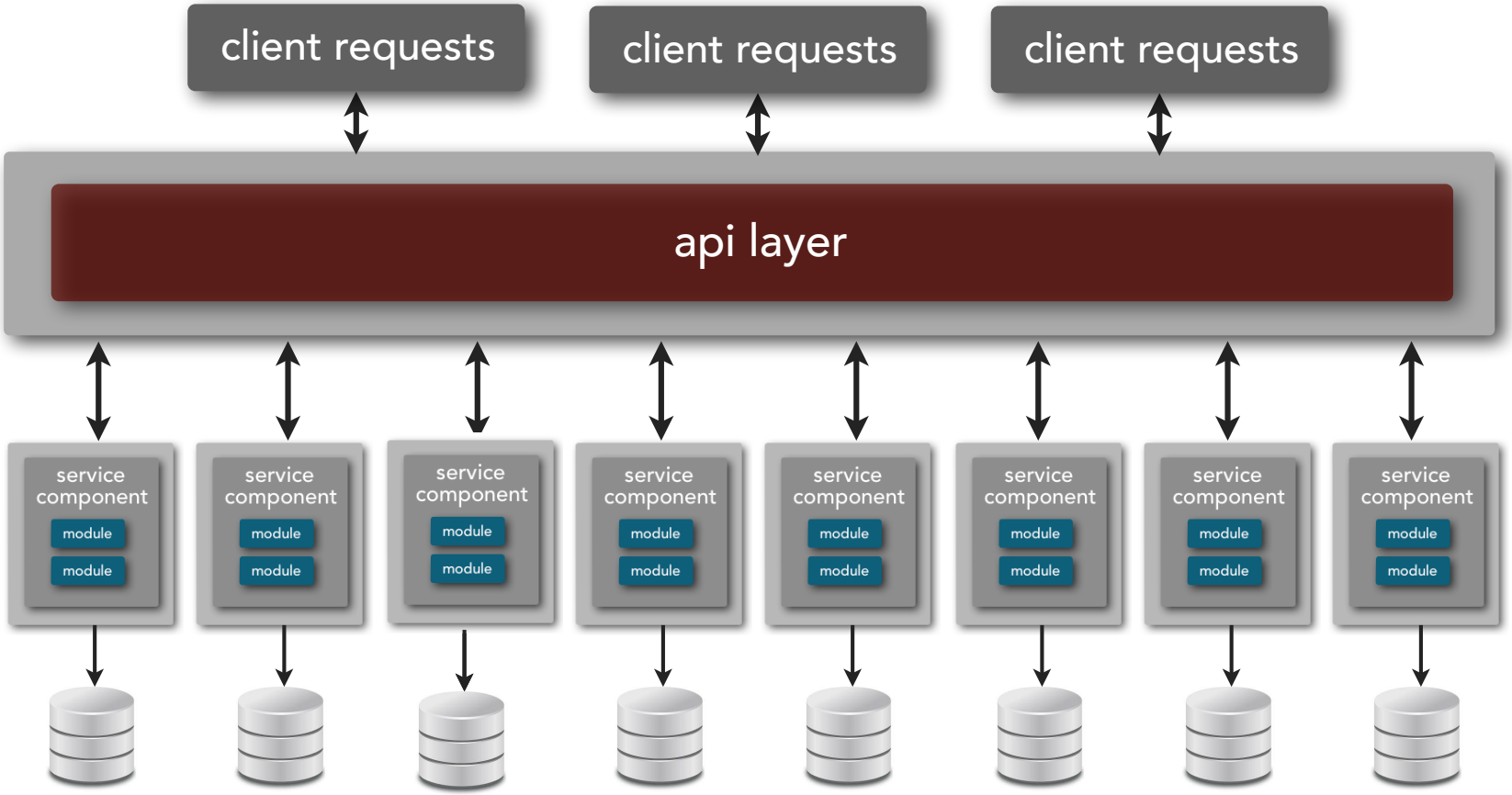


microservices architecture

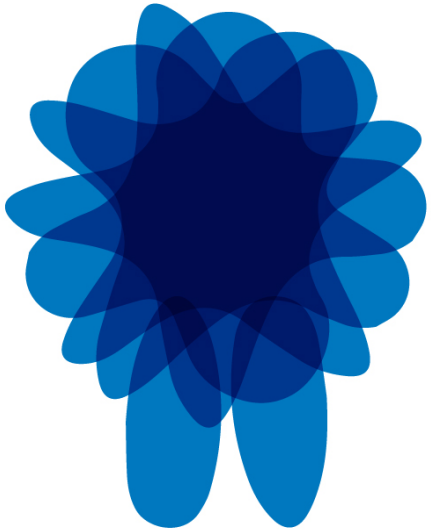
choreography



microservices architecture



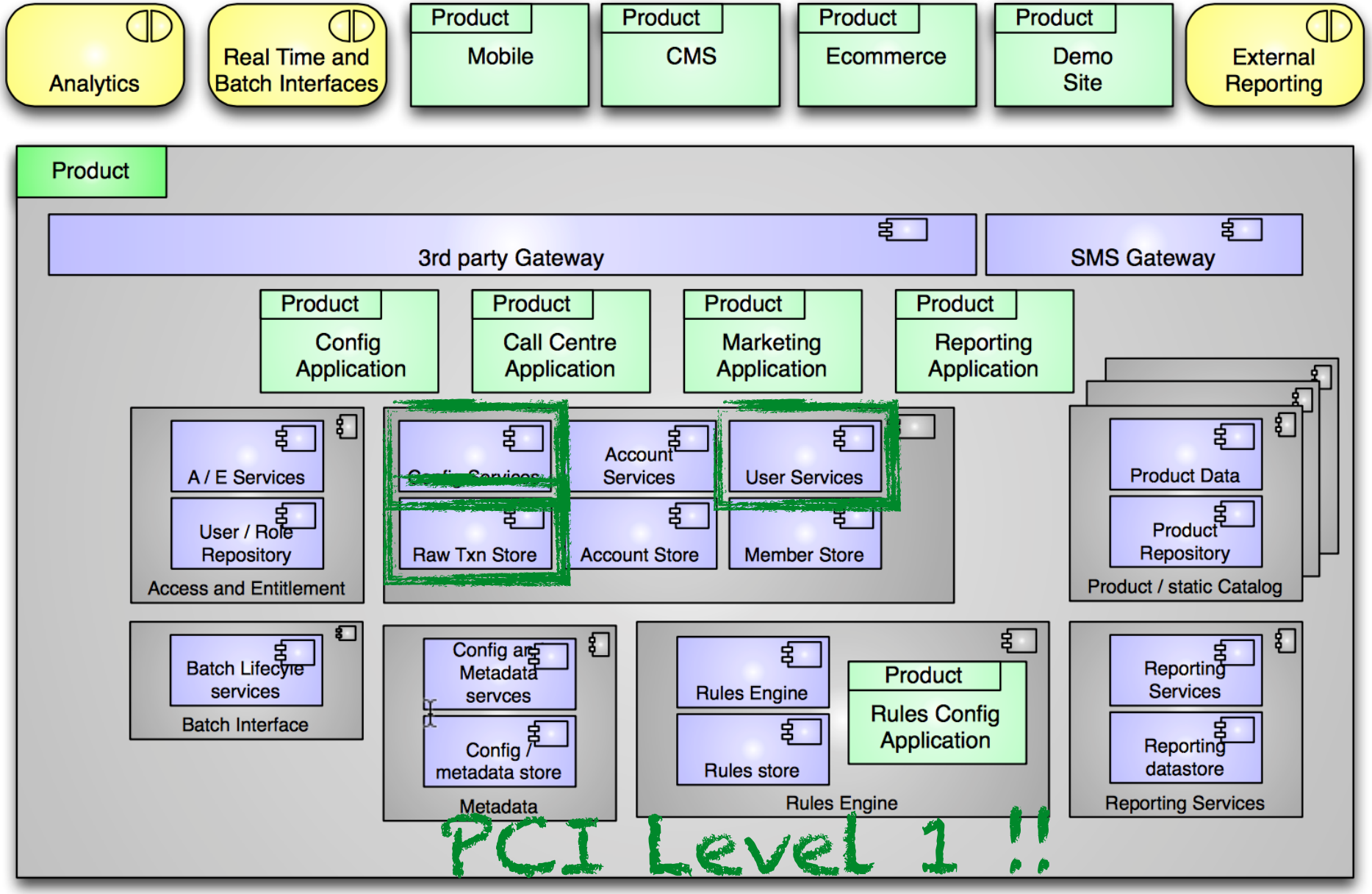
maximize easy evolution



support

Microservice is the first architectural style developed post-Continuous Delivery.

Microservice Implementation



http://2012.33degree.org/pdf/JamesLewisMicroServices.pdf

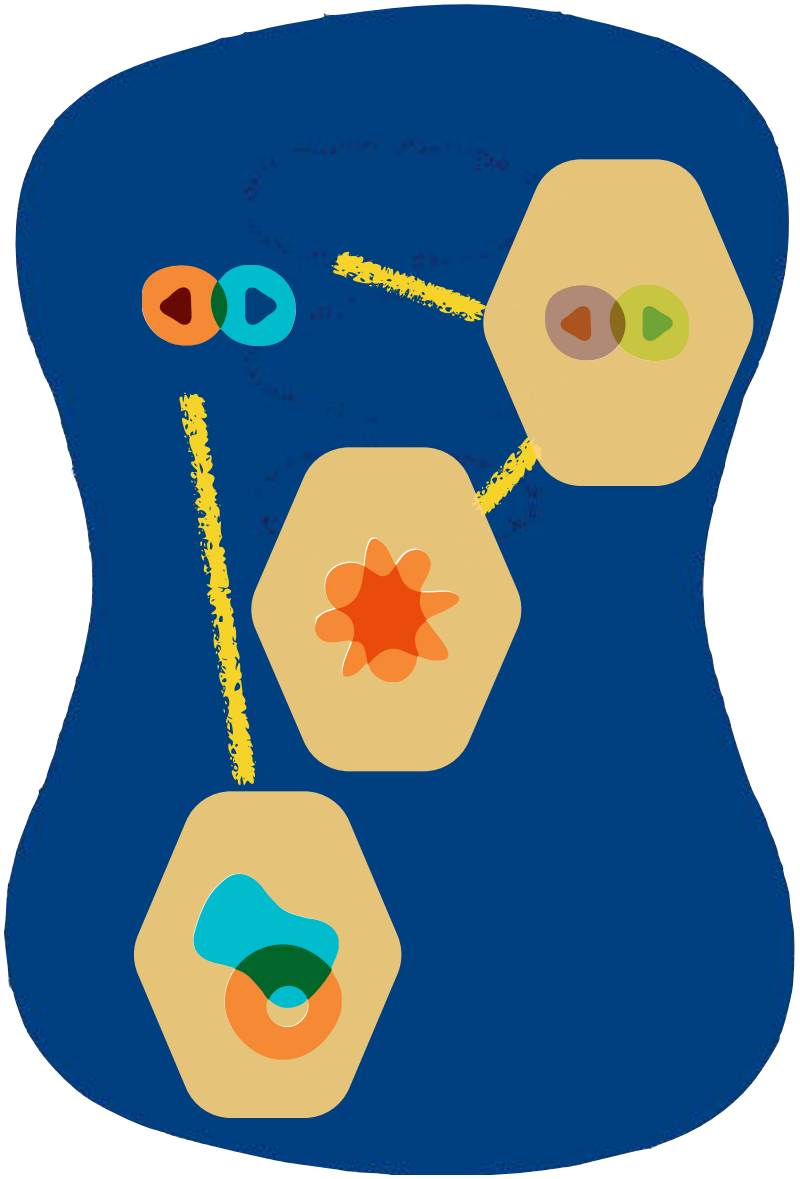
http://www.infoq.com/presentations/Micro-Services

Evolutionary Architecture

Components are *deployed*.

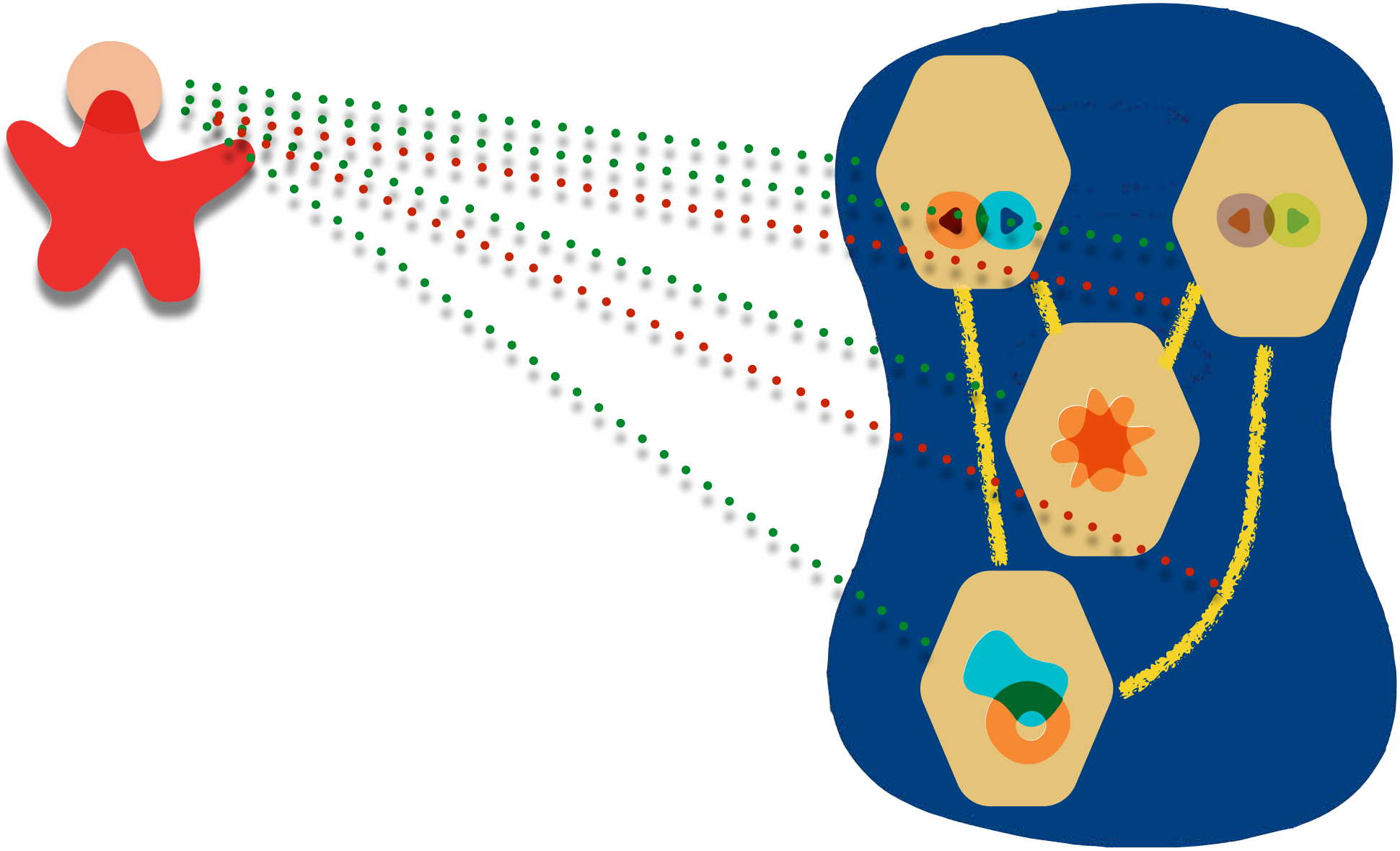
Features are *released*.

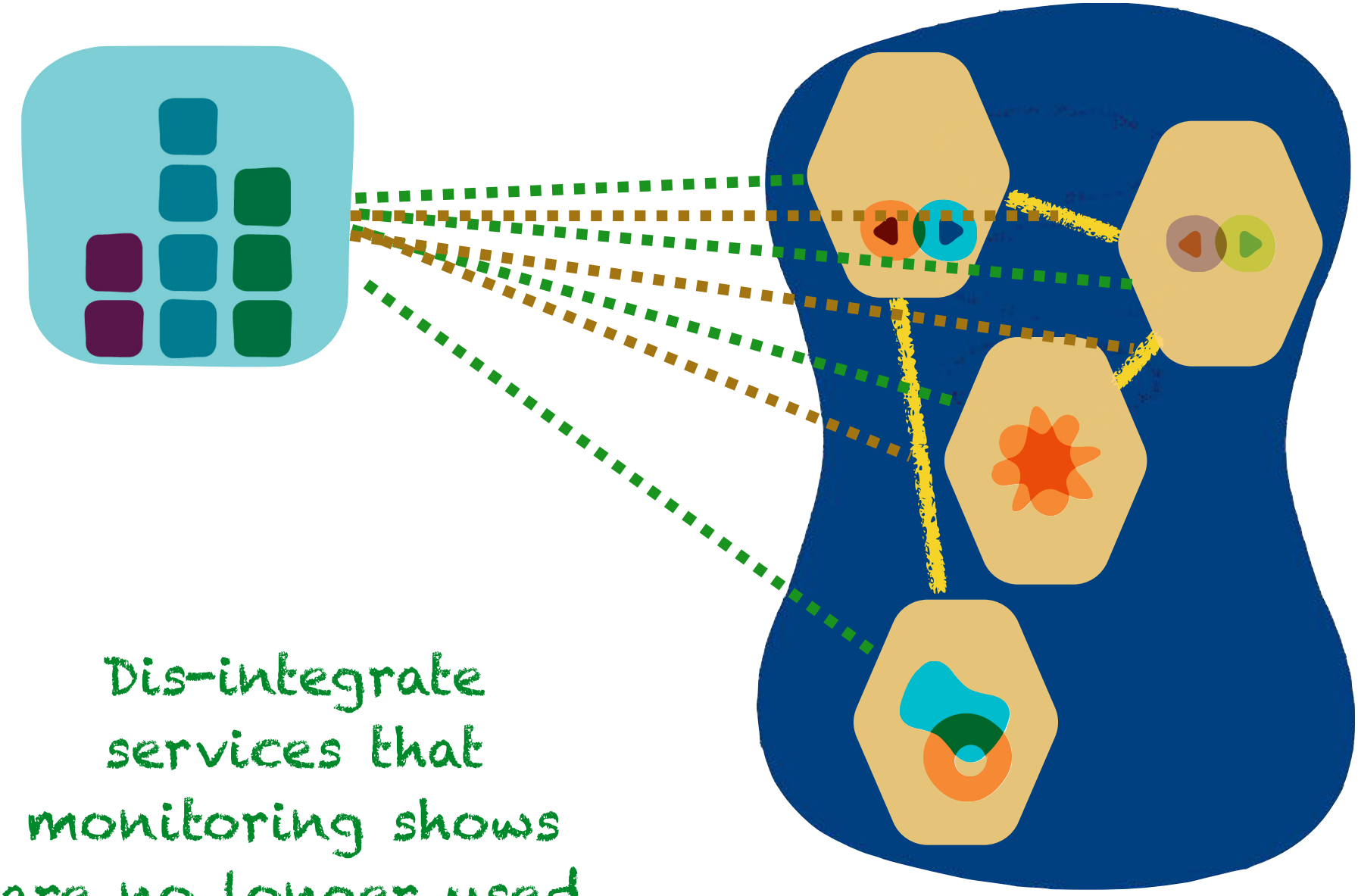
Applications consist of *routing*.



production

Evolutionary Architecture

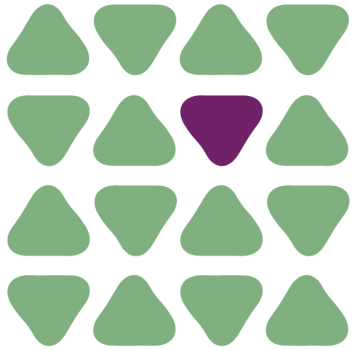


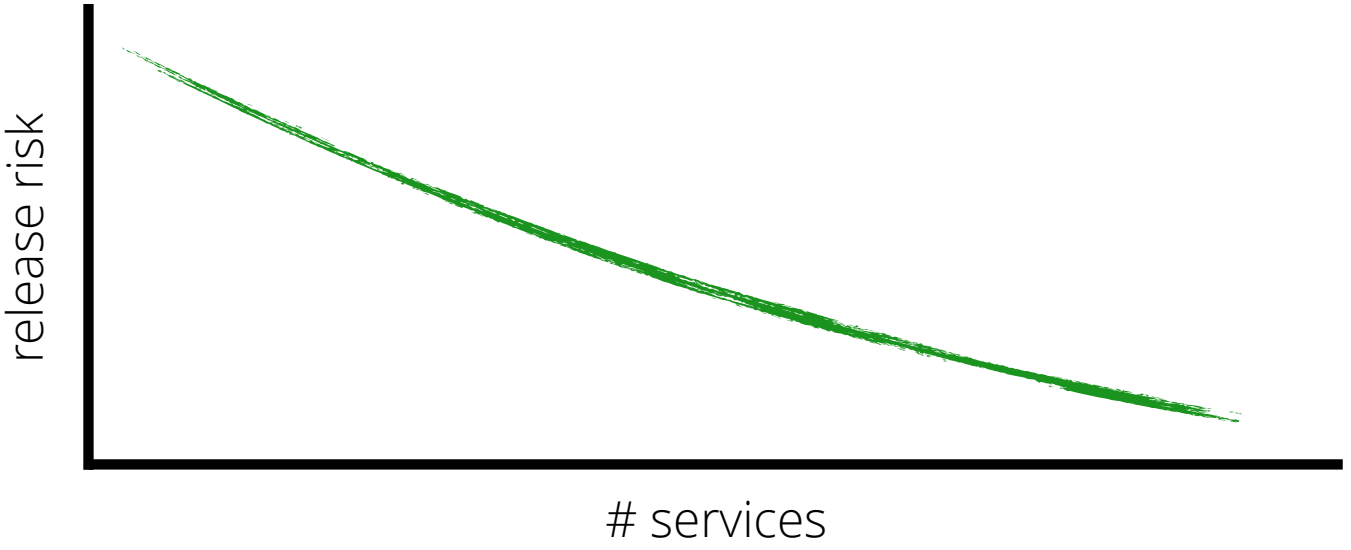


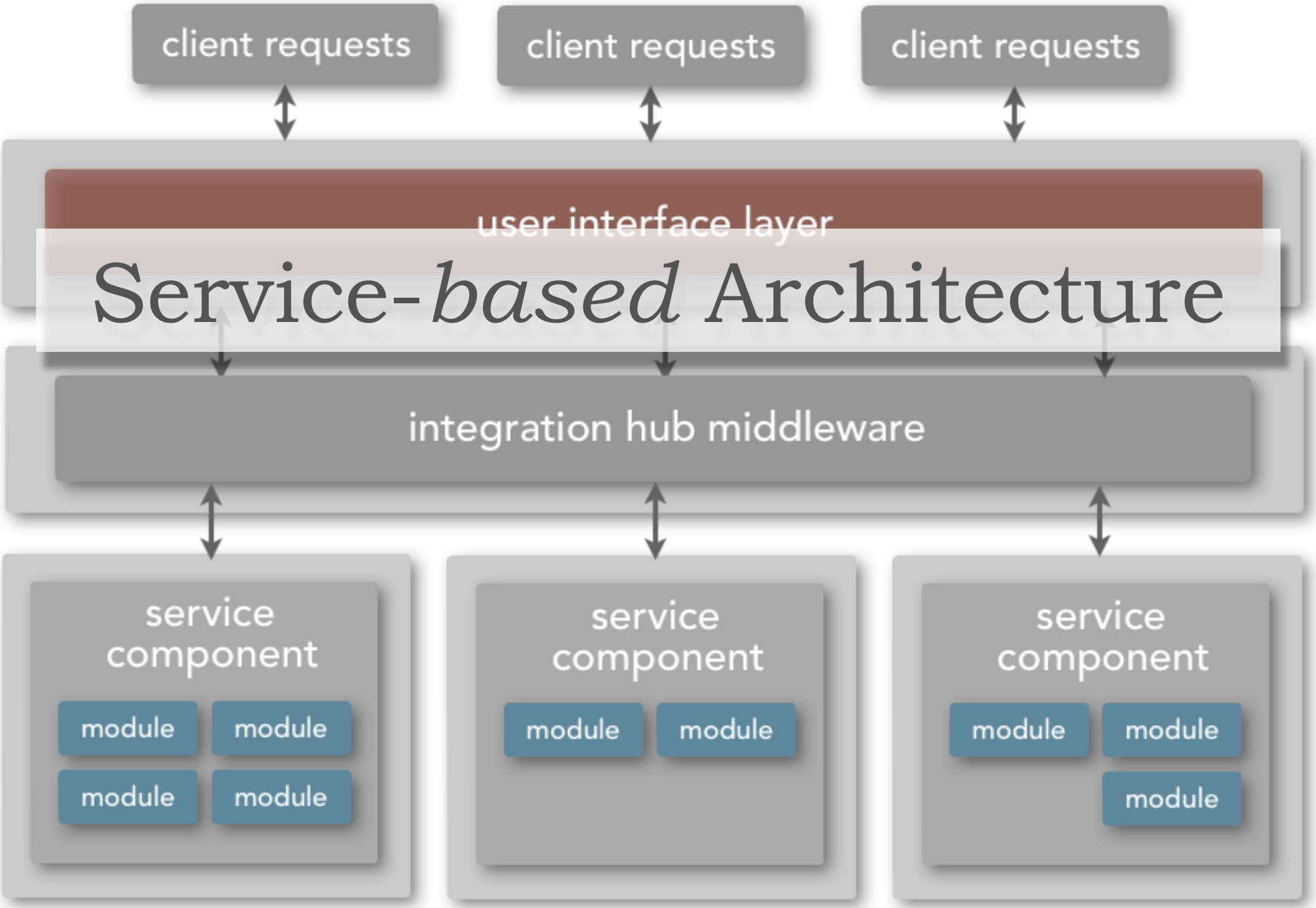
Dis-integrate services that monitoring shows are no longer used

production

How Big?



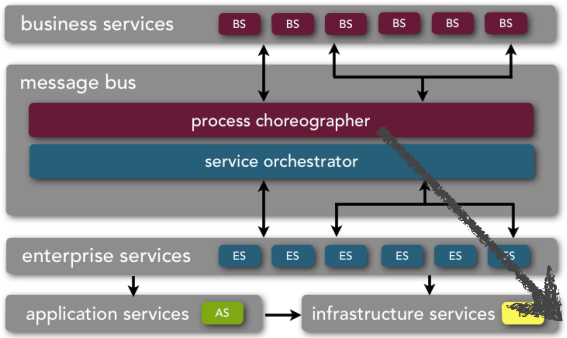


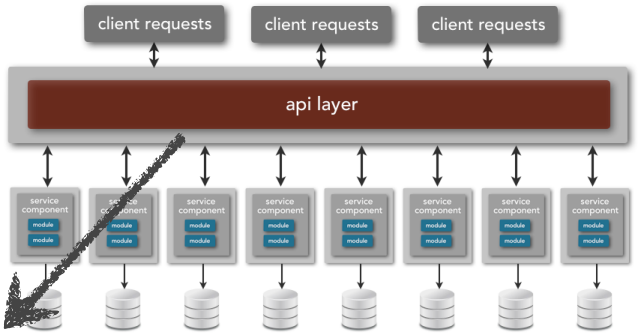


Service-*based* Architecture

service-based architecture

is there a middle ground?



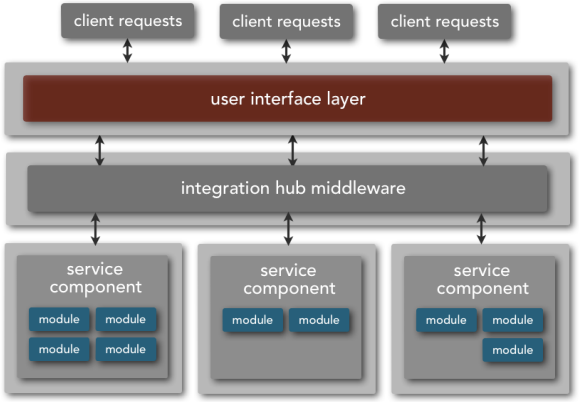


service-orie

architectu

croservices

rchitecture



service-based architecture

service-based architecture

prag·mat·ic

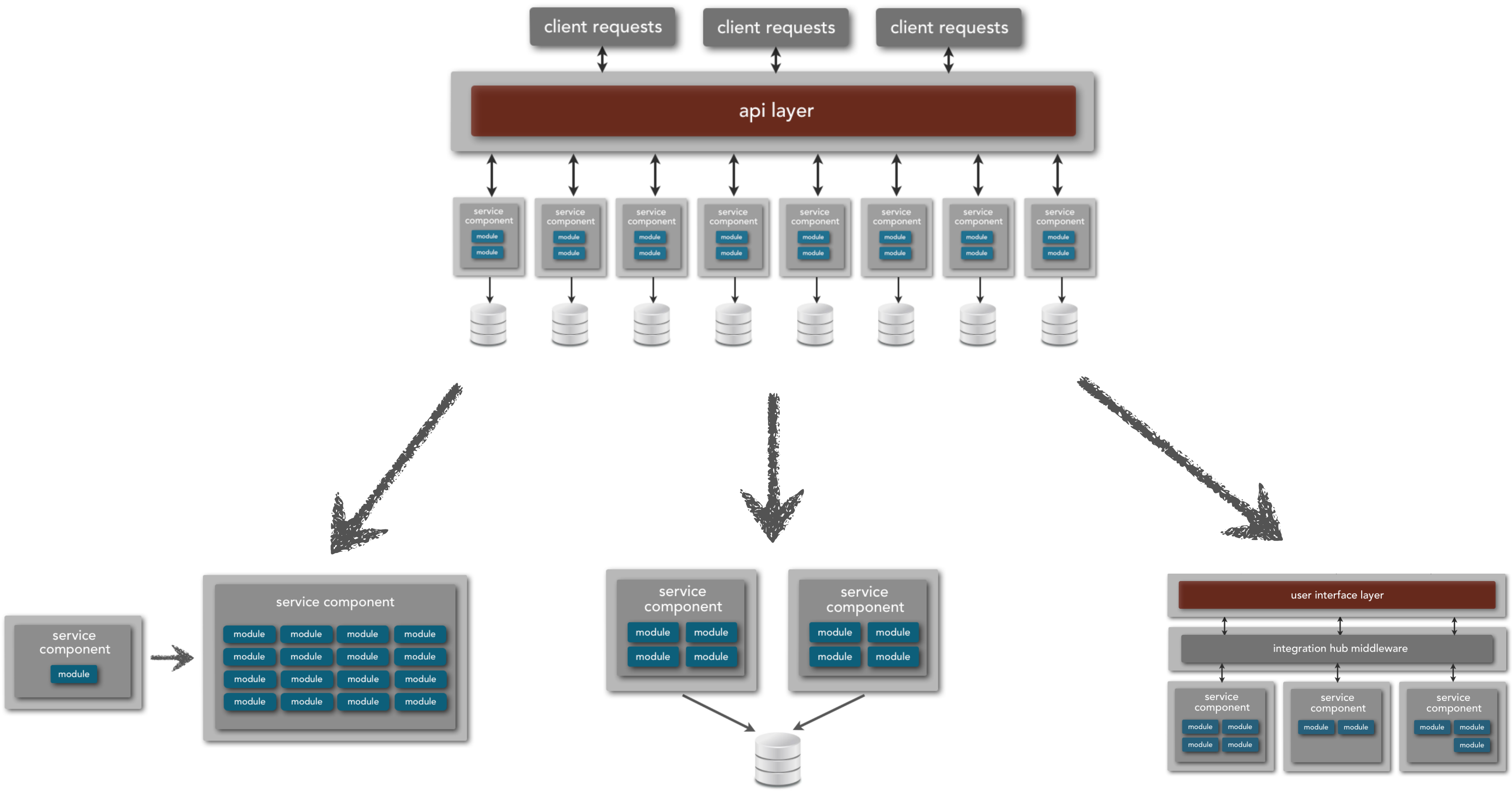
prag'madik/

adjective

adjective: **pragmatic**

1. dealing with things sensibly and realistically in a way that is based on practical rather than theoretical considerations.

service-based architecture



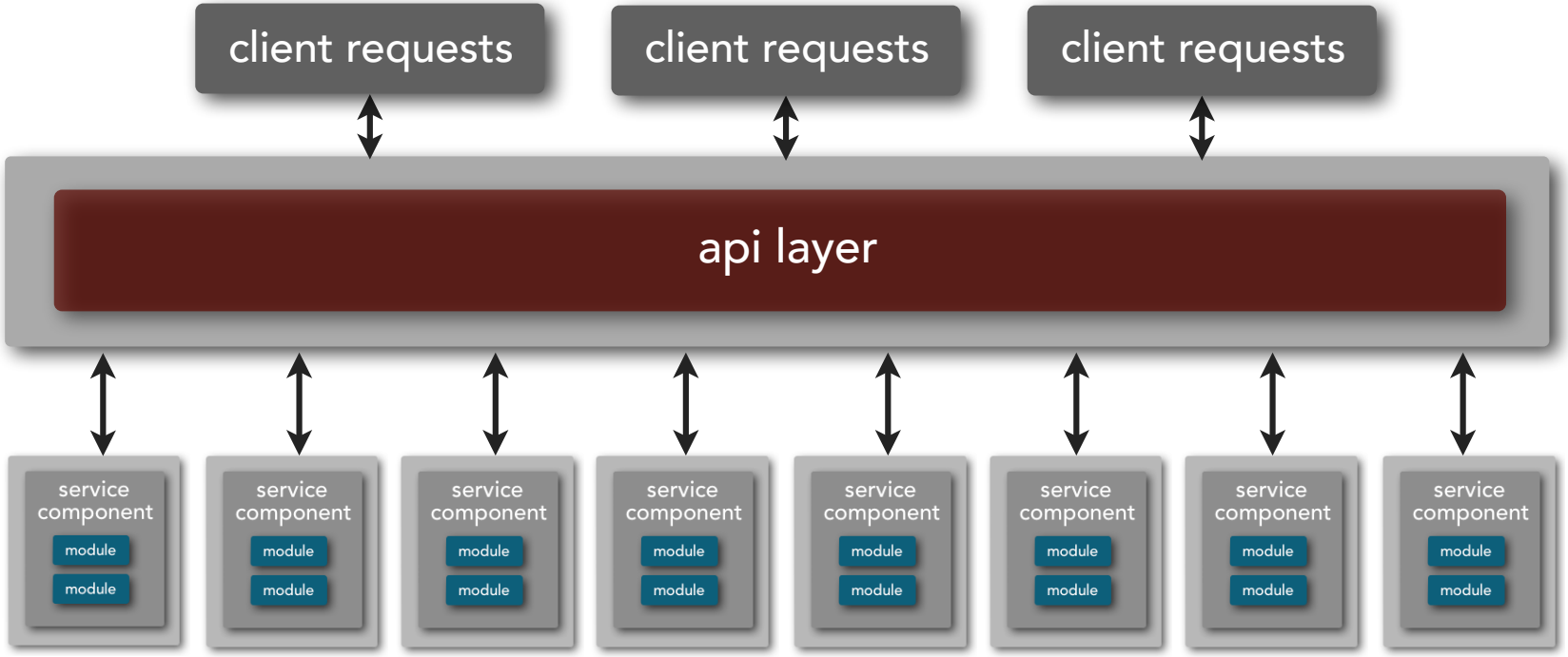
service granularity

database scope

integration hub

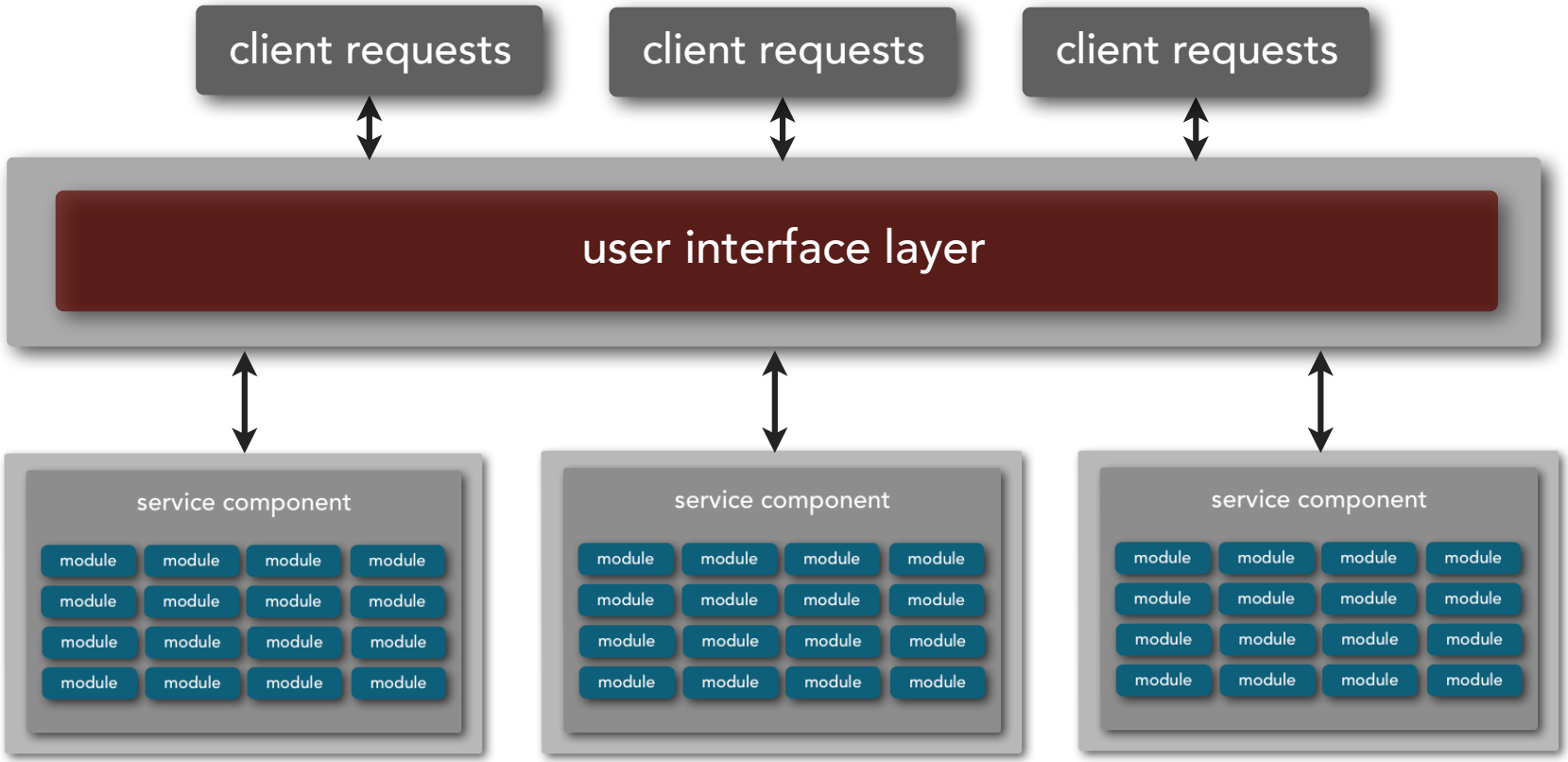
service-based architecture

service granularity



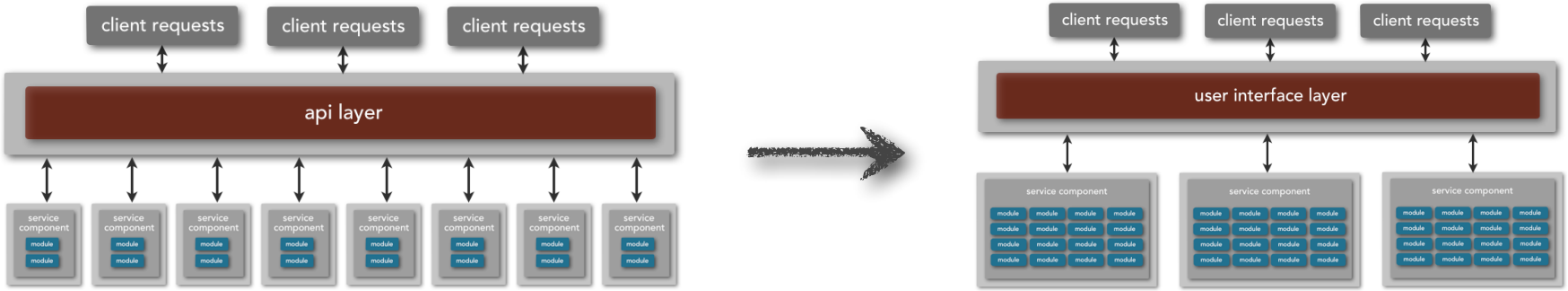
service-based architecture

service granularity



service-based architecture

service granularity



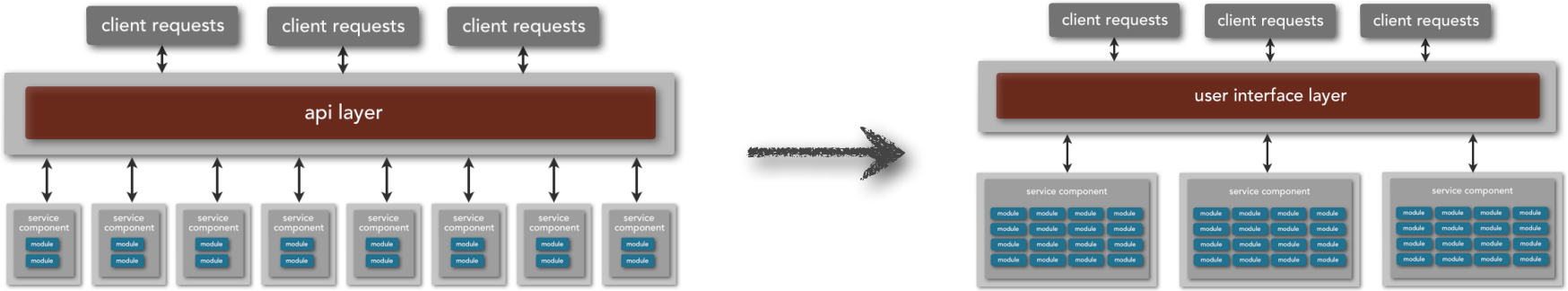
single-purpose micro-service to "portion of the application"

macro-service

- macro-services resolves orchestration and transactional issues
- allows for complex business processing within a service context

service-based architecture

service granularity



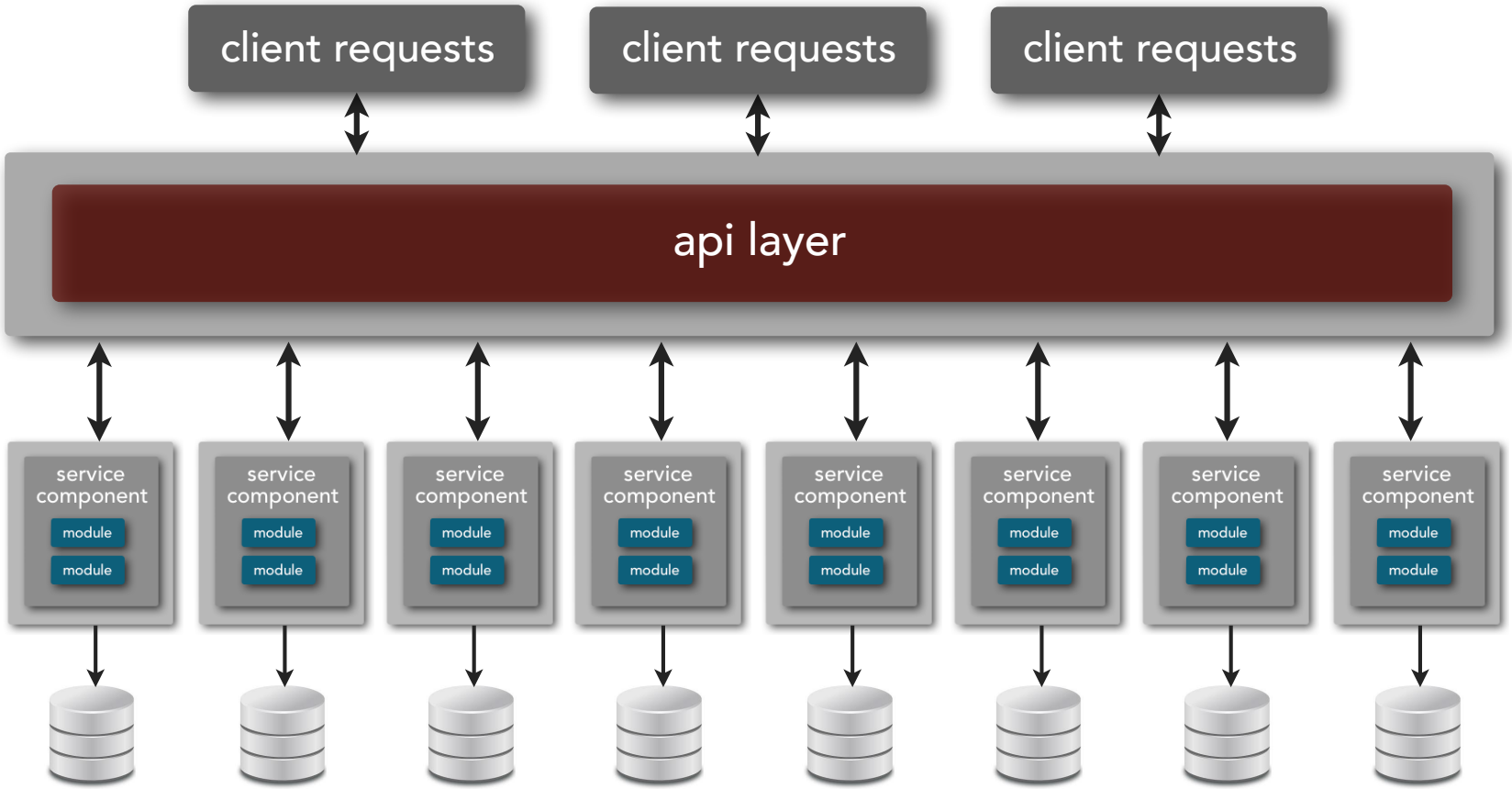
single-purpose micro-service to "portion of the application"

macro-service

- services become harder to develop and test
- deployment pipeline requires more planning
- change control becomes more difficult

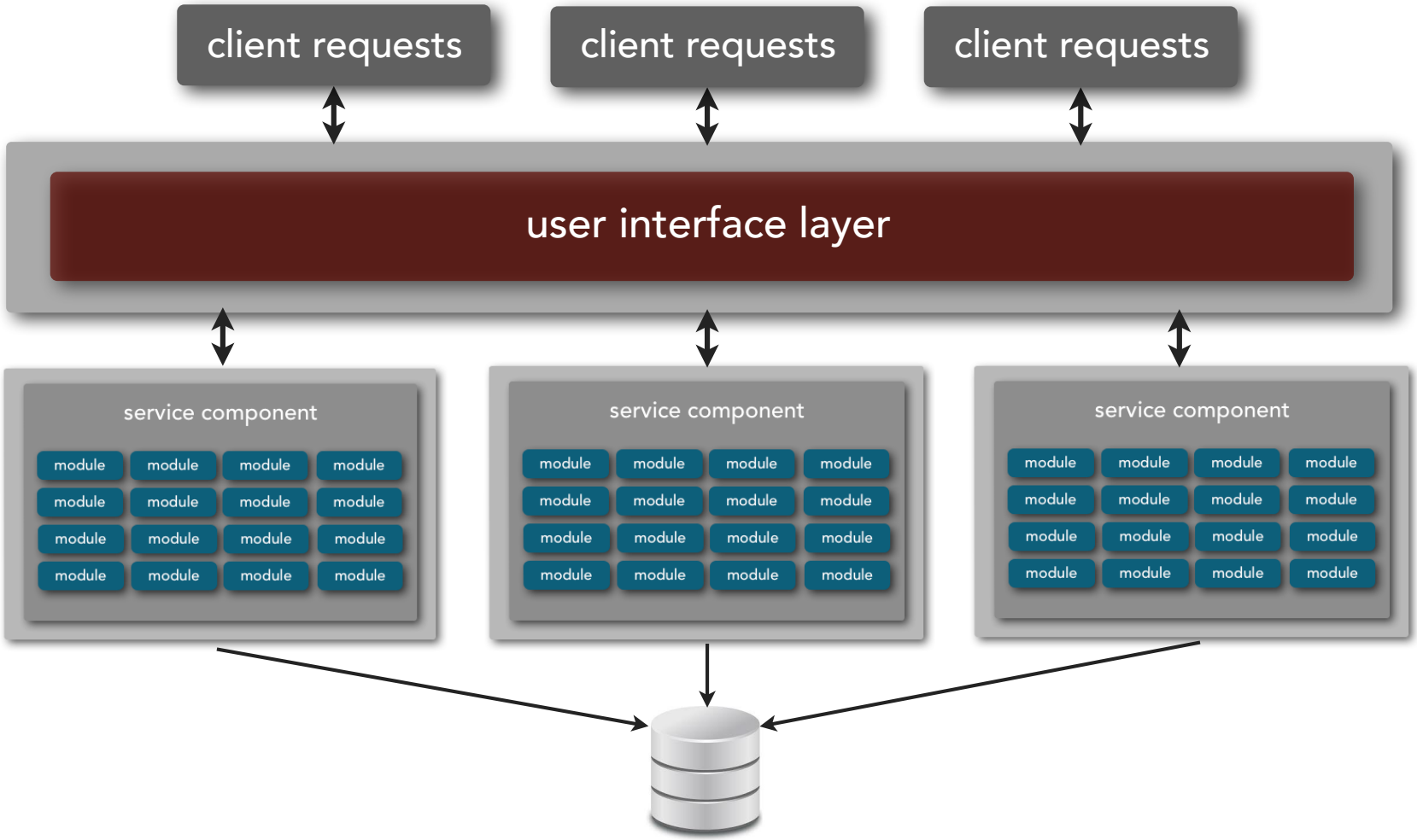
service-based architecture

database scope



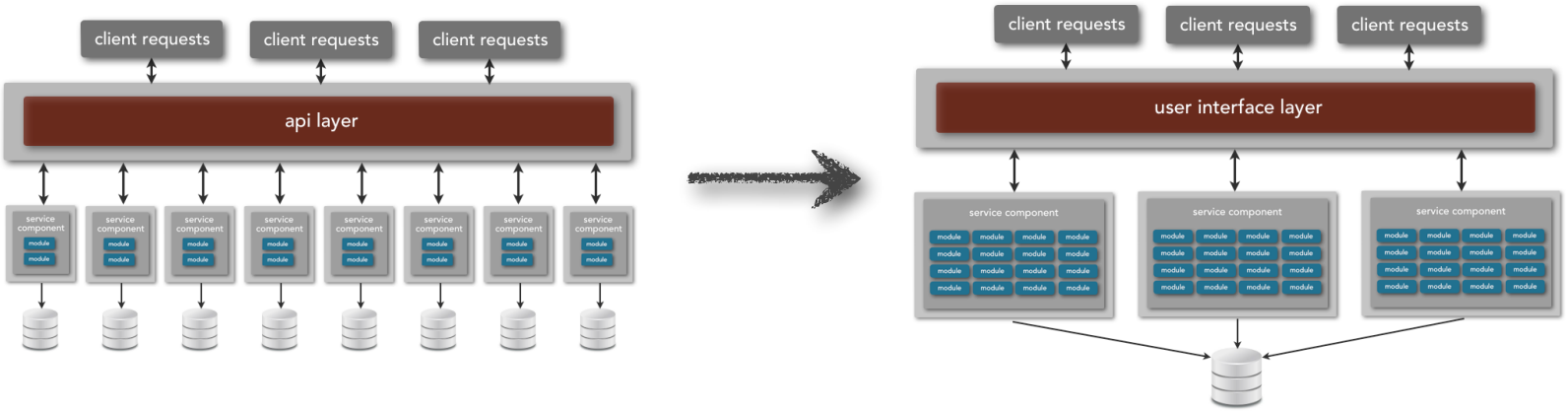
service-based architecture

database scope



service-based architecture

database scope

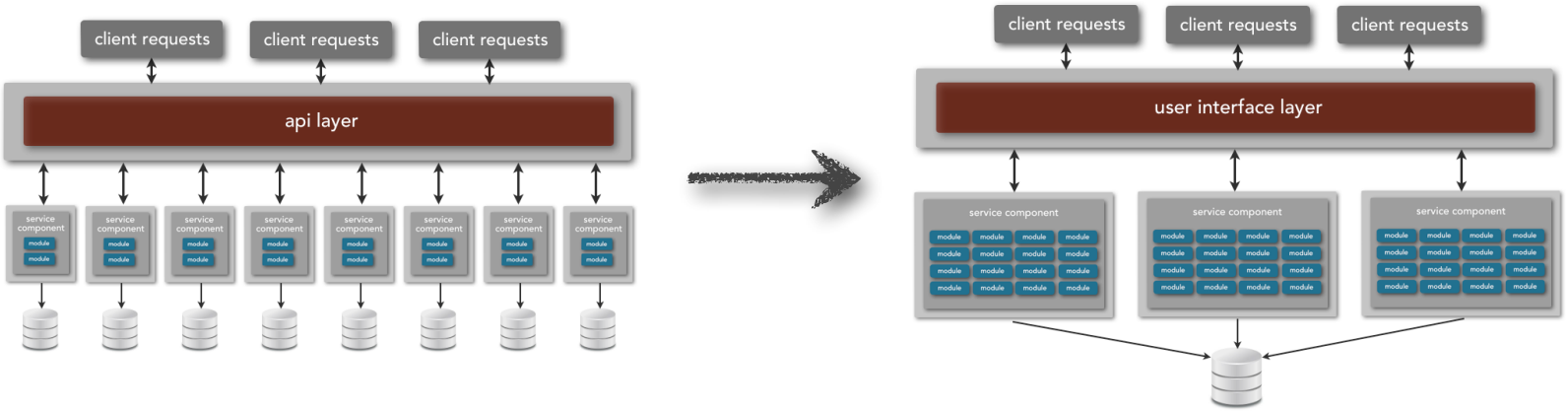


single-purpose service-based database to globally shared application database

- reduces service orchestration and contract dependencies
- improves performance due to fewer remote calls
- refactoring entire database may not be feasible or possible

service-based architecture

database scope

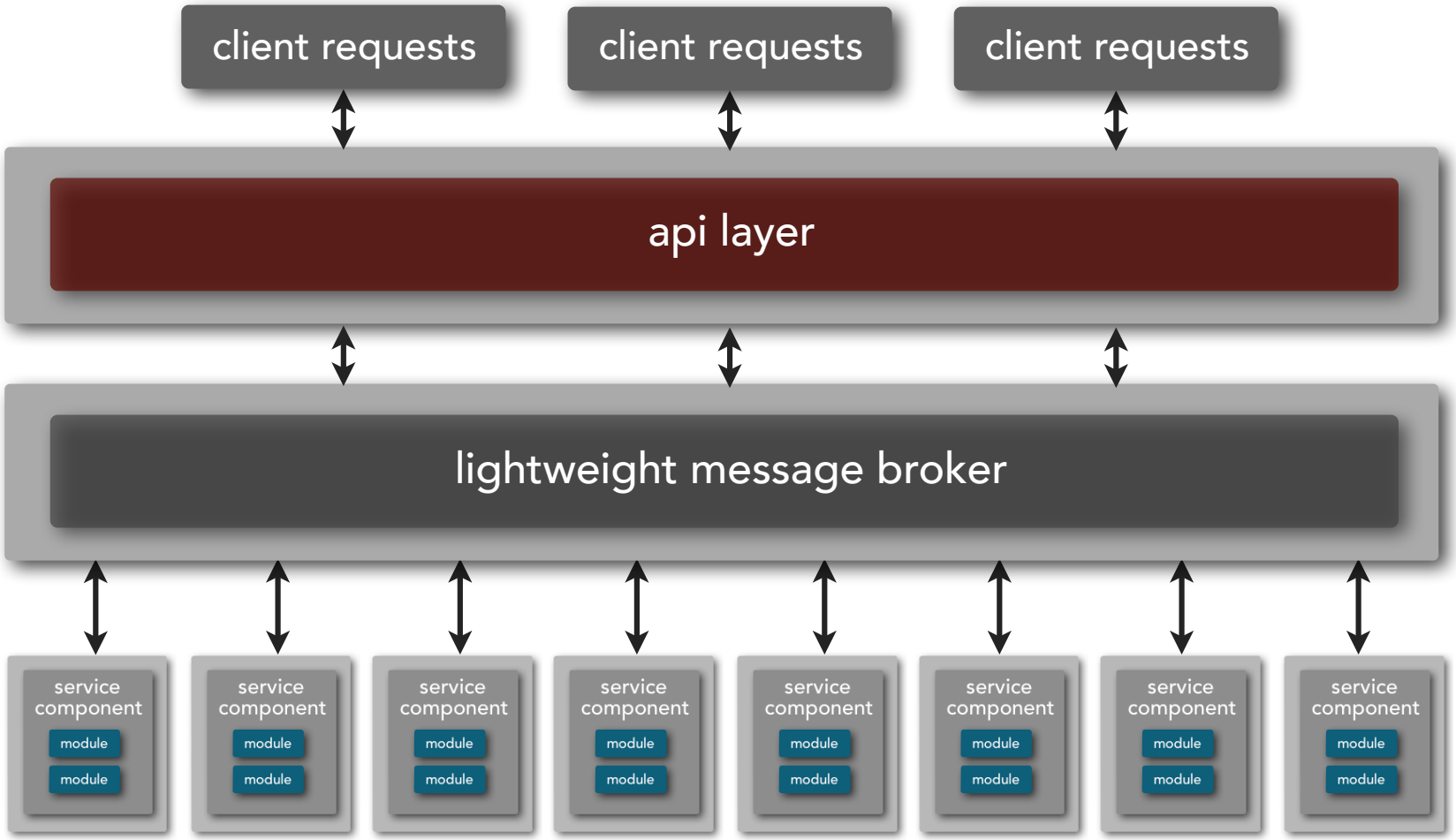


single-purpose service-based database to globally shared application database

- looser bounded context of services
- tighter service coupling based on schema
- schema changes become expensive and difficult

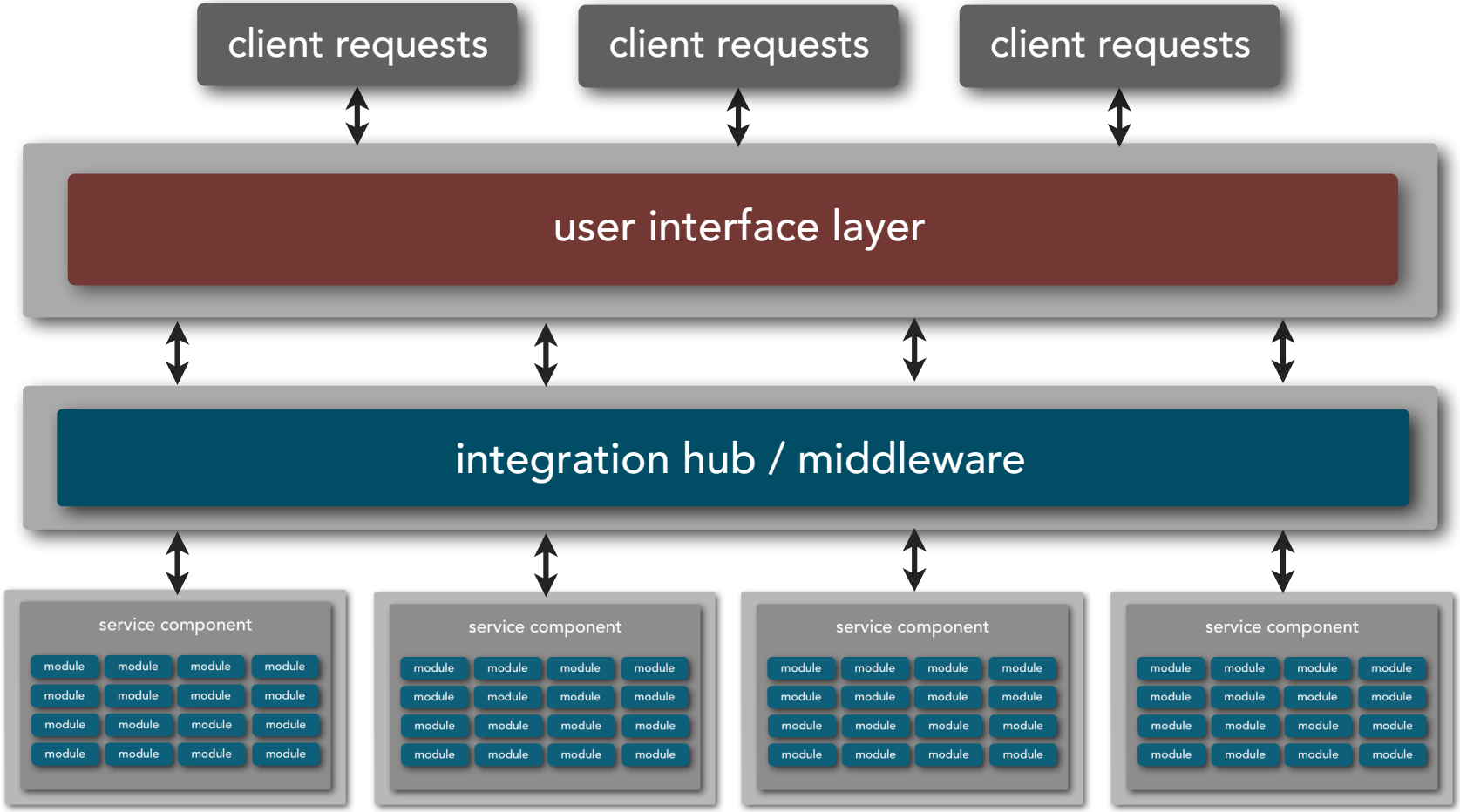
service-based architecture

integration hub



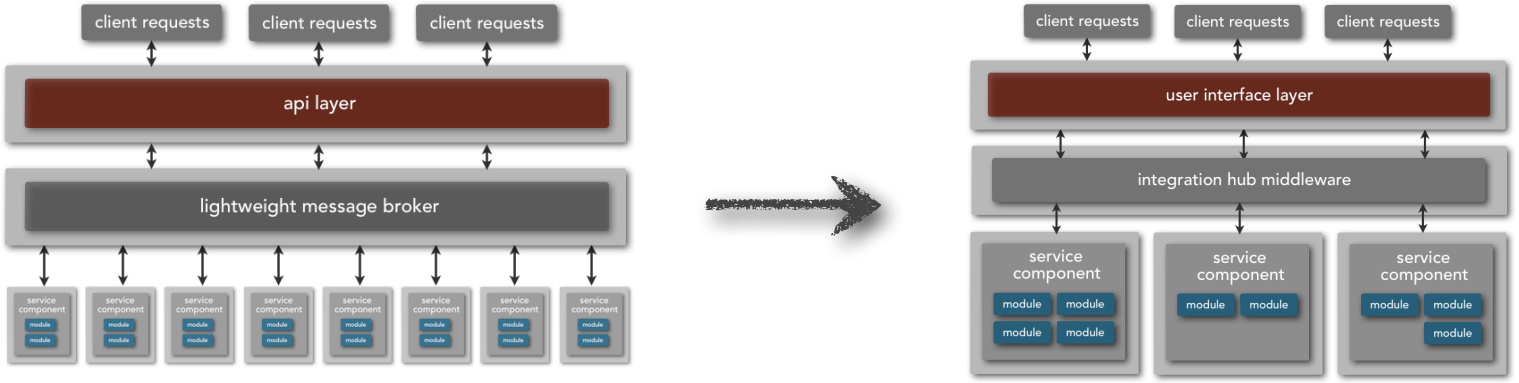
service-based architecture

integration hub



service-based architecture

integration hub

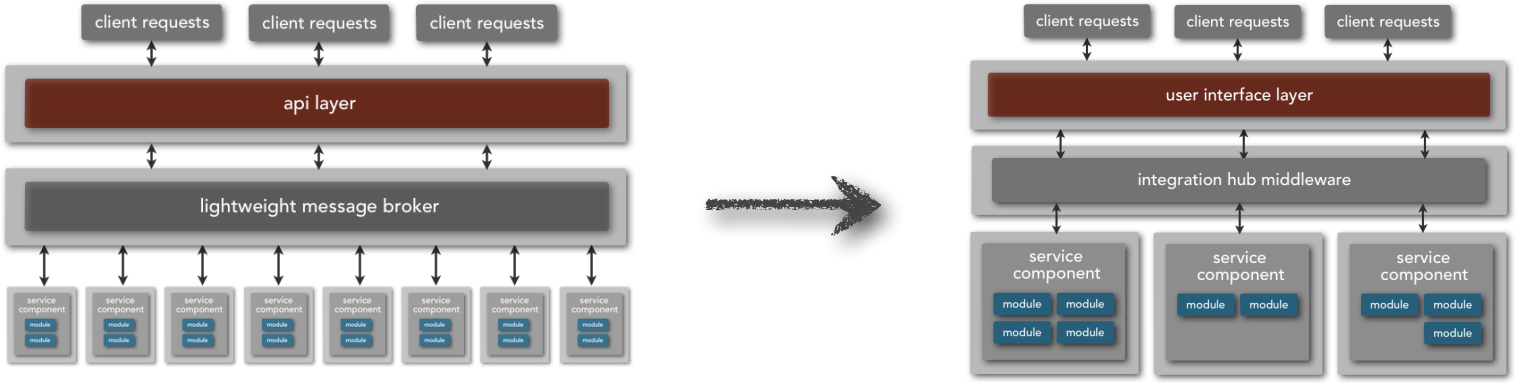


lightweight message broker to heavier integration hub

- allows for transformation of contract differences
- allows for non-transactional orchestration of services
- allows for protocol-agnostic heterogeneous interoperability
- allows for common processing logic across all services

service-based architecture

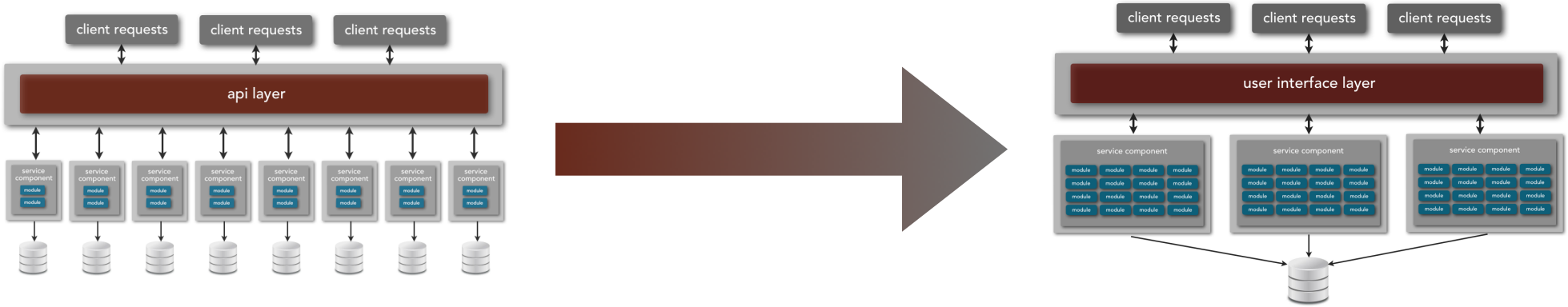
integration hub



lightweight message broker to heavier integration hub

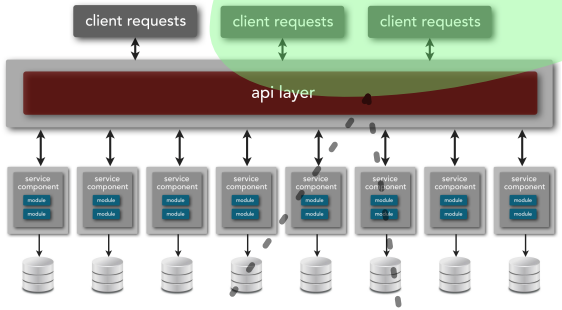
- decrease in overall performance
- added complexity and cost
- increased need for governance
- deployment pipeline requires much more planning
- services become harder to develop and test

what people really mean when they say “microservice”:

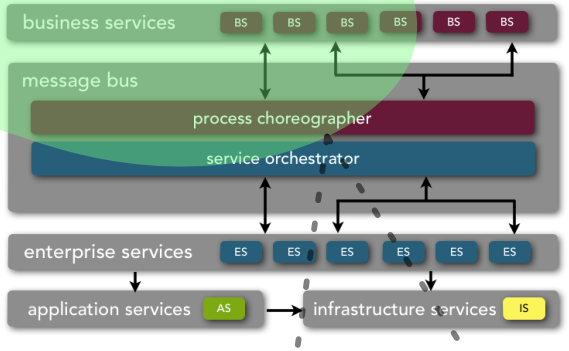


A service based architecture with domain emphasis but without the strictness of a “pure” microservice architecture.

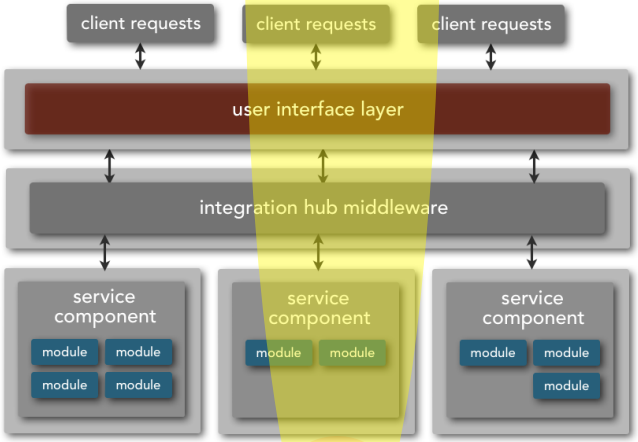
Comparing:



Micro



Service-oriented

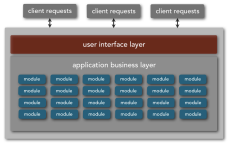


Service-based

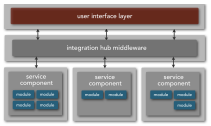
characteristics differences

overall agility

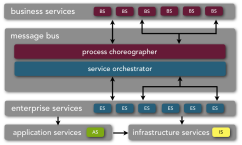
ability to respond quickly to constant change in both business and technology



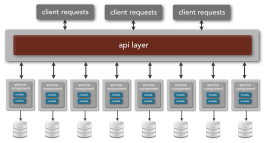
monolithic architecture



service-based architecture



service-oriented architecture

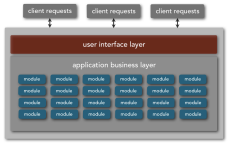


microservices architecture

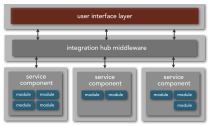
characteristics differences

ease of deployment

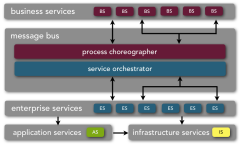
promotes an effective and fast deployment pipeline; features are quick and easy to deploy



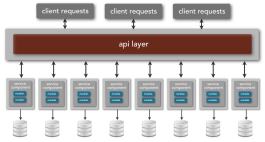
monolithic architecture



service-based architecture



service-oriented architecture

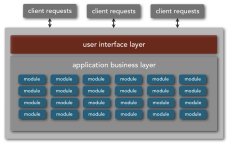


microservices architecture

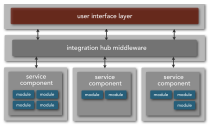
characteristics differences

ease of testing

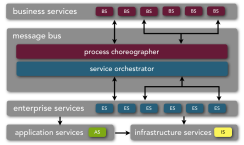
ease at which features can be tested and verified; confidence level in completeness of testing



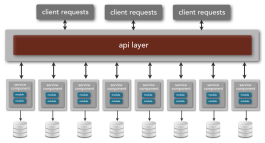
monolithic architecture



service-based architecture



service-oriented architecture

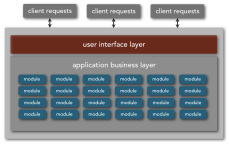


microservices architecture

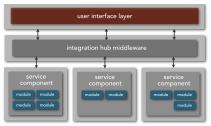
characteristics differences

overall performance

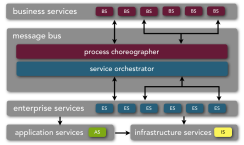
which patterns relatively promote better performing applications?



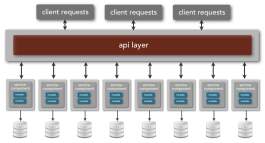
monolithic architecture



service-based architecture



service-oriented architecture

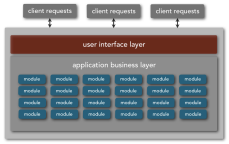


microservices architecture

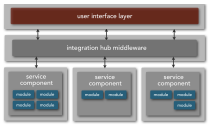
characteristics differences

overall scalability

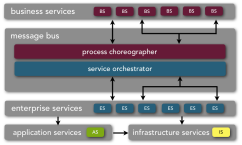
how well does the architecture pattern lend itself to highly scalable applications?



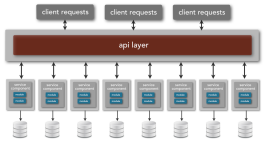
monolithic architecture



service-based architecture



service-oriented architecture

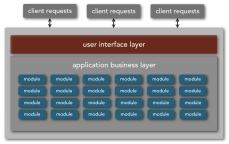


microservices architecture

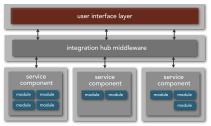
characteristics differences

overall simplicity

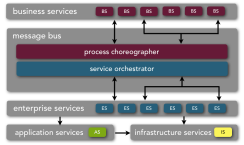
level of complexity in applications implemented using the architecture pattern



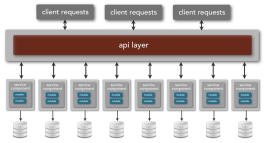
monolithic architecture



service-based architecture



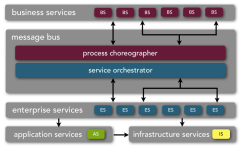
service-oriented architecture



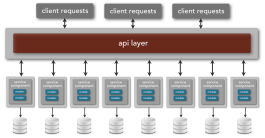
microservices architecture

characteristics differences

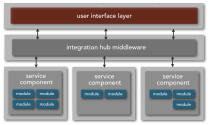
ratings relative to each pattern



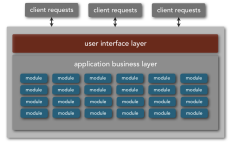
service-oriented architecture



microservices architecture



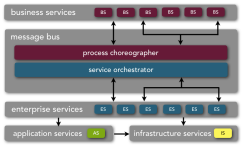
service-based architecture



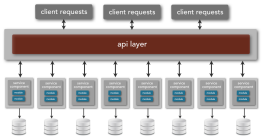
monolithic architecture

overall agility	L	H	M	L
deployment	L	H	M	L
testability	L	H	M	M
performance	L	M	M	H
scalability	M	H	M	L
overall simplicity	L	M	M	H

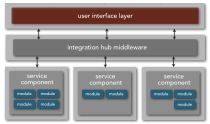
service differences



service-oriented architecture



microservices architecture



service-based architecture

P service categorization

P service granularity

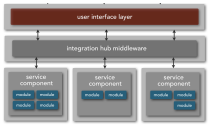
P service orchestration

P service numbers

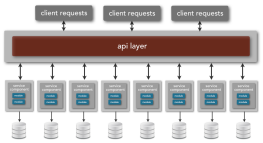
service differences

service categorization

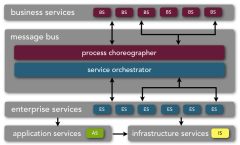
are services categorized or classified within the architecture pattern?



service-based architecture



microservices architecture



service-oriented architecture

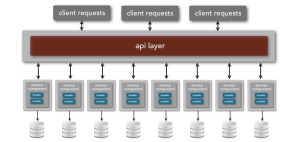
no service categorization

rigid service categorization

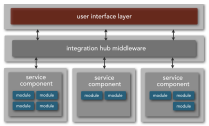
service differences

service granularity

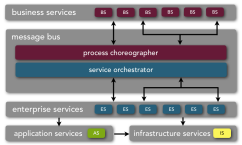
what is the typical granularity of services within this pattern?



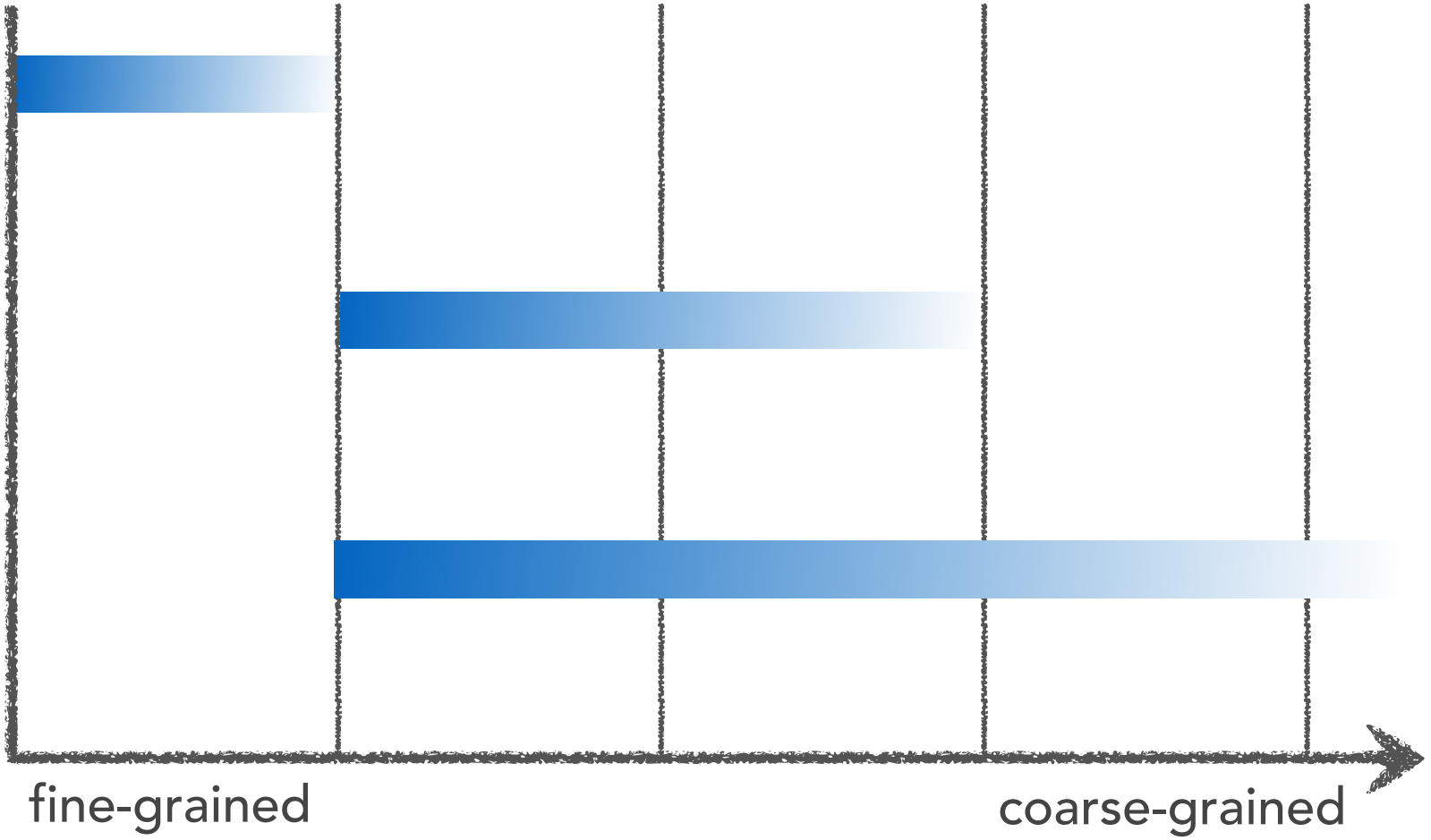
microservices architecture



service-based architecture



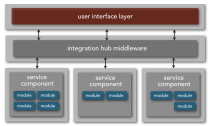
service-oriented architecture



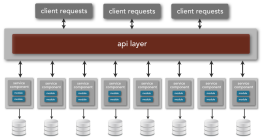
service differences

service orchestration

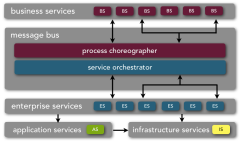
what level of service orchestration typically exists between services with this pattern?



service-based architecture



microservices architecture



service-oriented architecture

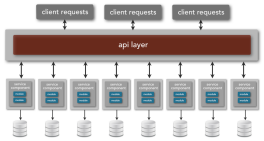
minimal service orchestration

maximum service orchestration

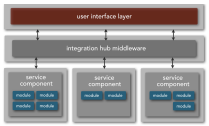
service differences

service numbers

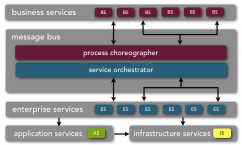
what is the typical upper limit of the number of services found?



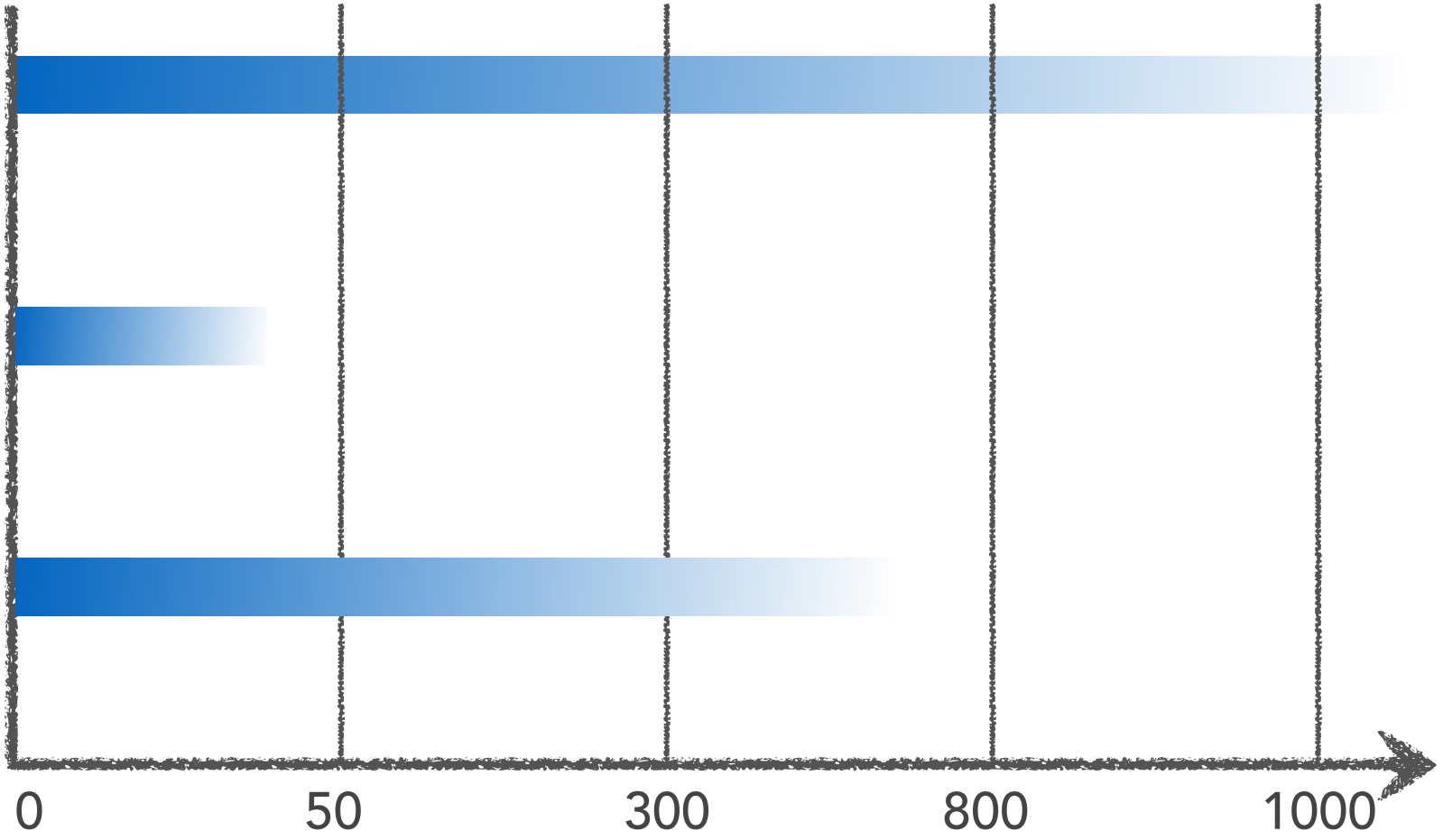
microservices architecture



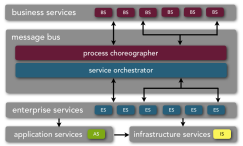
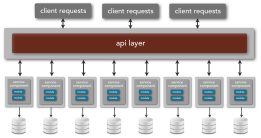
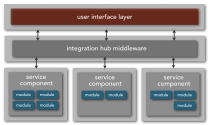
service-based architecture



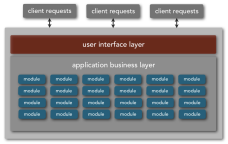
service-oriented architecture

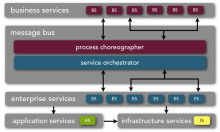


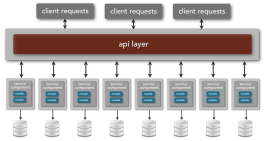
service differences

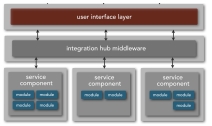
	 service-oriented architecture	 microservices architecture	 service-based architecture
service categorization	yes	no	no
service granularity	small to very large	very small	small to medium
service orchestration	yes	minimal	generally no
service numbers	hundreds	dozens	thousands

Which ?!?

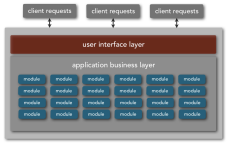

client requests
client requests
client requests
user interface layer
application business layer
module


business services
message bus
process choreographer
service orchestrator
enterprise services
application services
infrastructure services


client requests
client requests
client requests
api layer


user interface layer
integration hub middleware
service component
module

Which ?!?



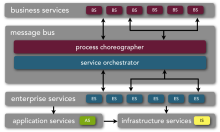
Monolith

default

easy to understand/build

doesn’t scale well in any dimension

Which ?!?



Service-oriented

service taxonomy

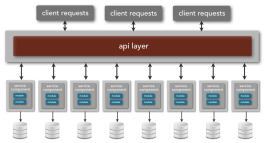
high degree of (potential) reuse

highly compatible in integration-heavy environments

operationally complex

Which ?!?

Microservices

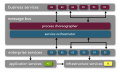


incremental change

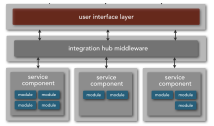
highly evolvable

complex interactions

Which ?!?



service-based

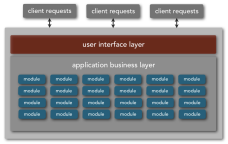


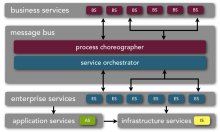
best target for migration

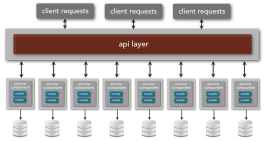
good compromise

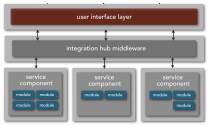
domain-centric without going *μ*service

Which ?!?





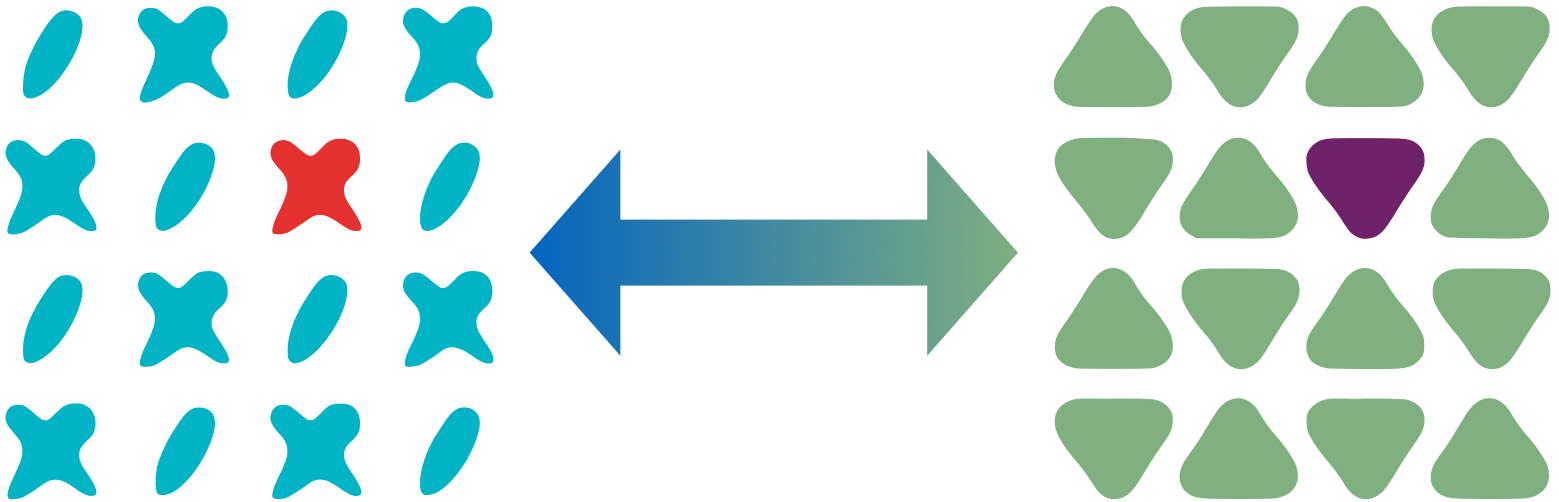


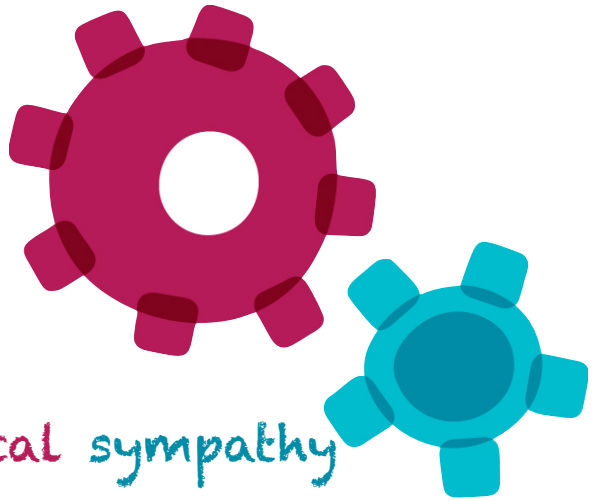


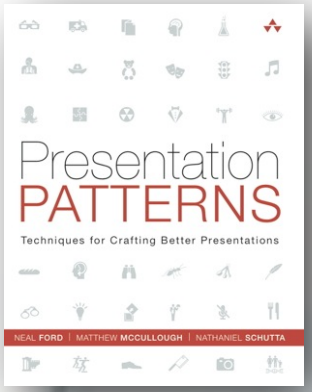
it

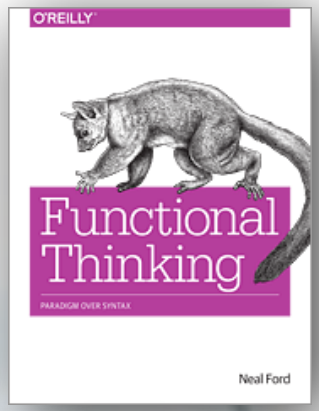
depends

domain/architecture isomorphism









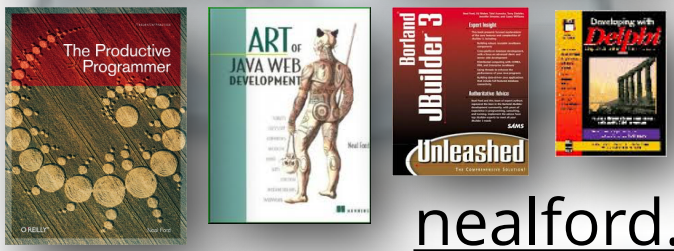
nealford.com

@neal4d

ThoughtWorks®

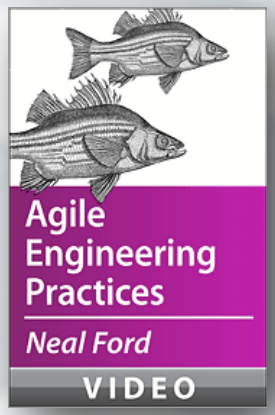
NEAL FORD

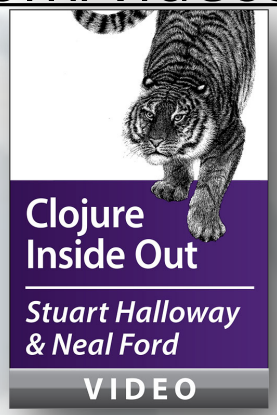
Director / Software Architect / Meme Wrangler

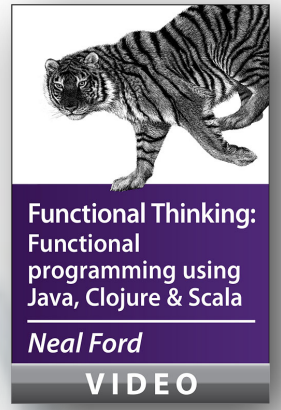


nealford.com/books

nealford.com/videos







O'REILLY®

SOFTWARE ARCHITECTURE SERIES

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