

cOmPaRiNg SeRViCE-bASeD aRchiTecTureS



ThoughtWorks®

NEAL FORD

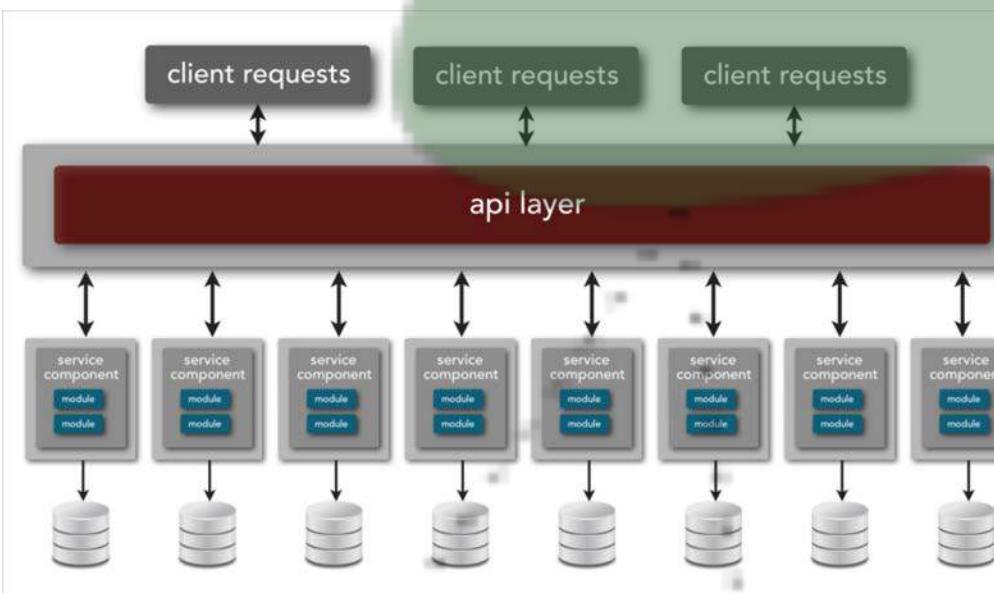
Director / Software Architect / Meme Wrangler



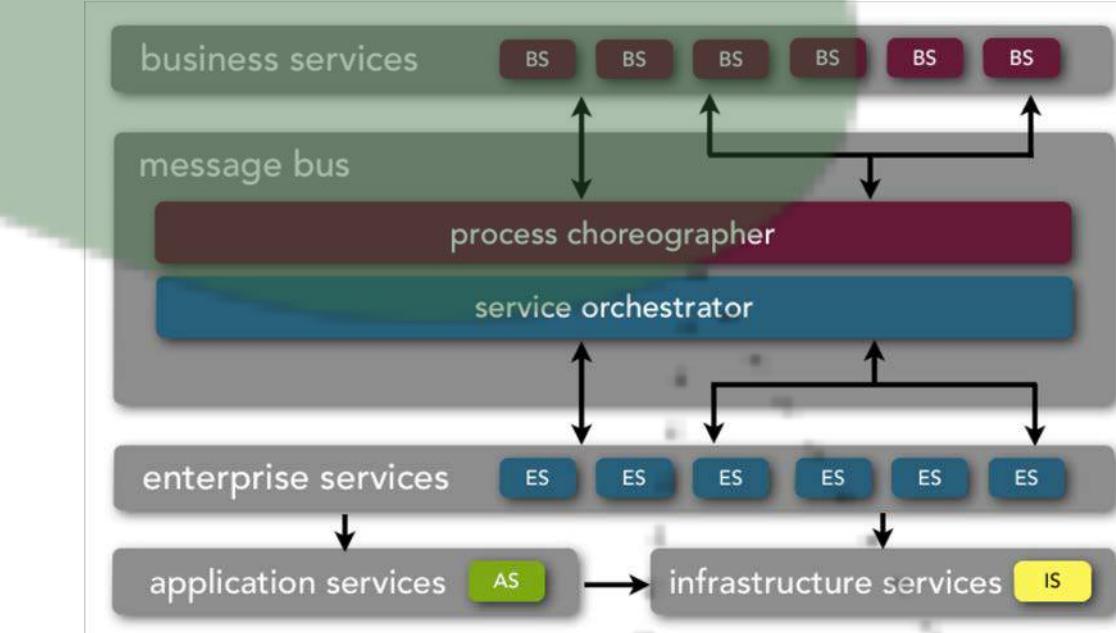
@neal4d

nealford.com

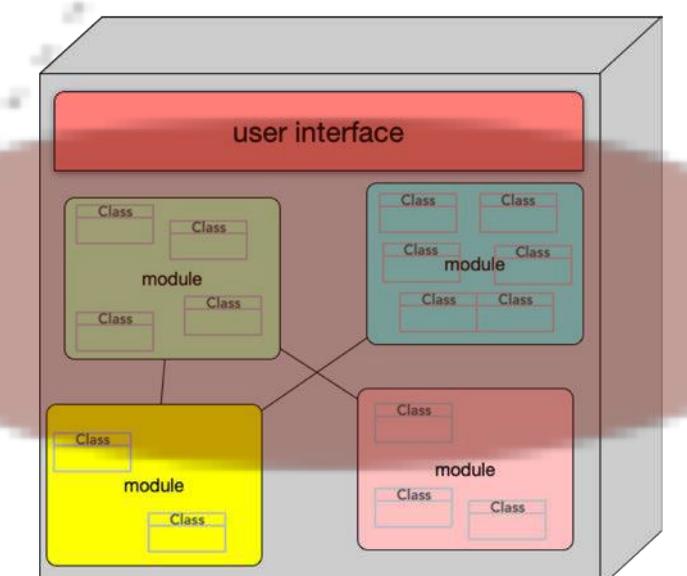
agenda



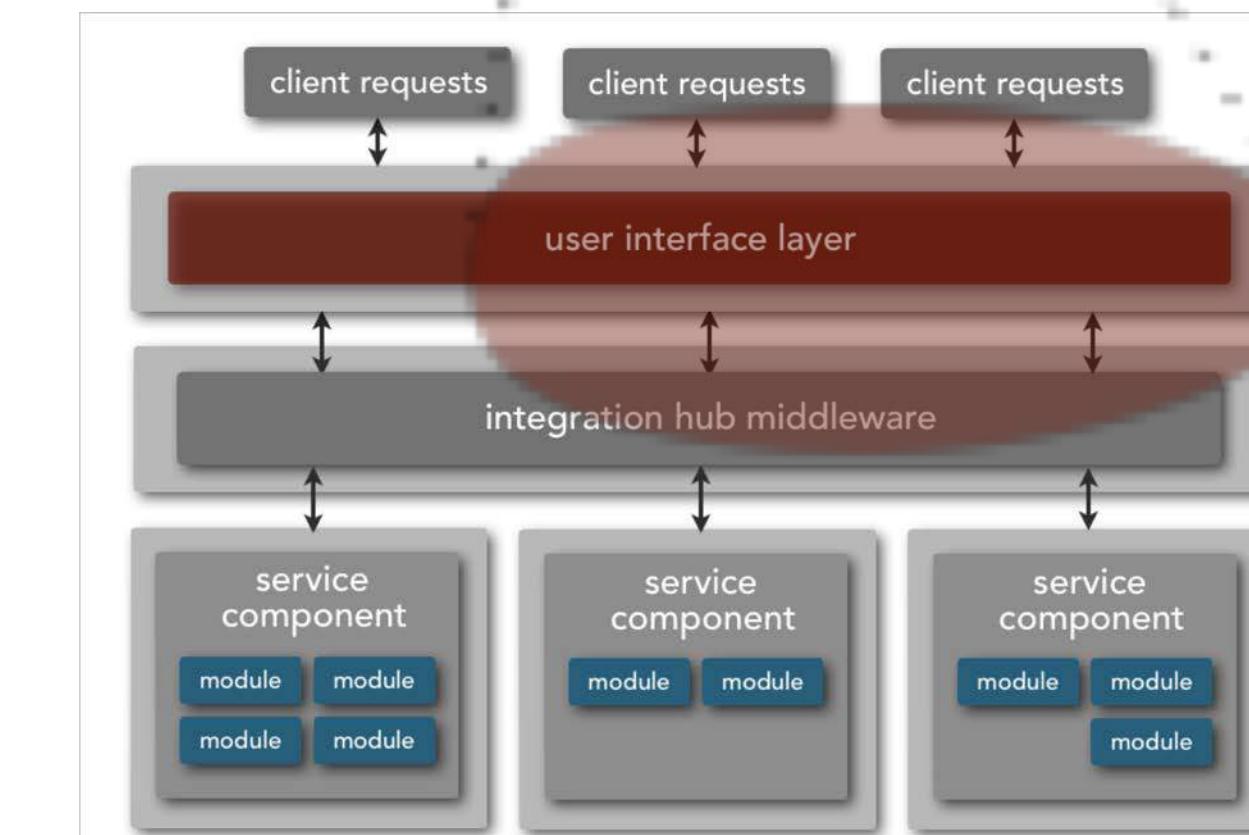
Micro



Service-oriented

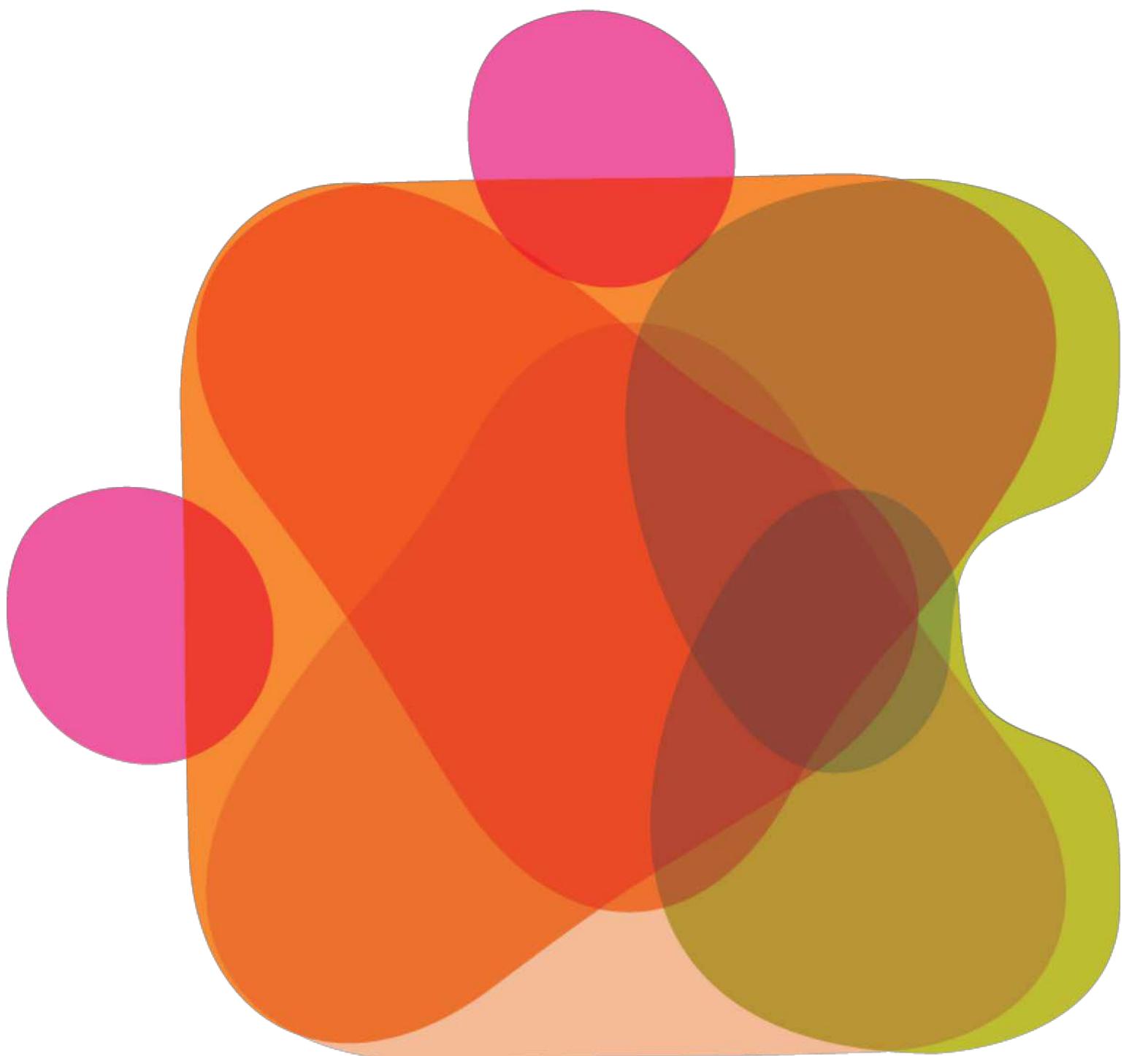


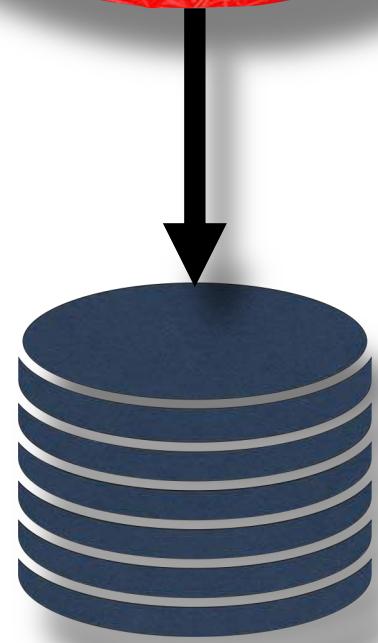
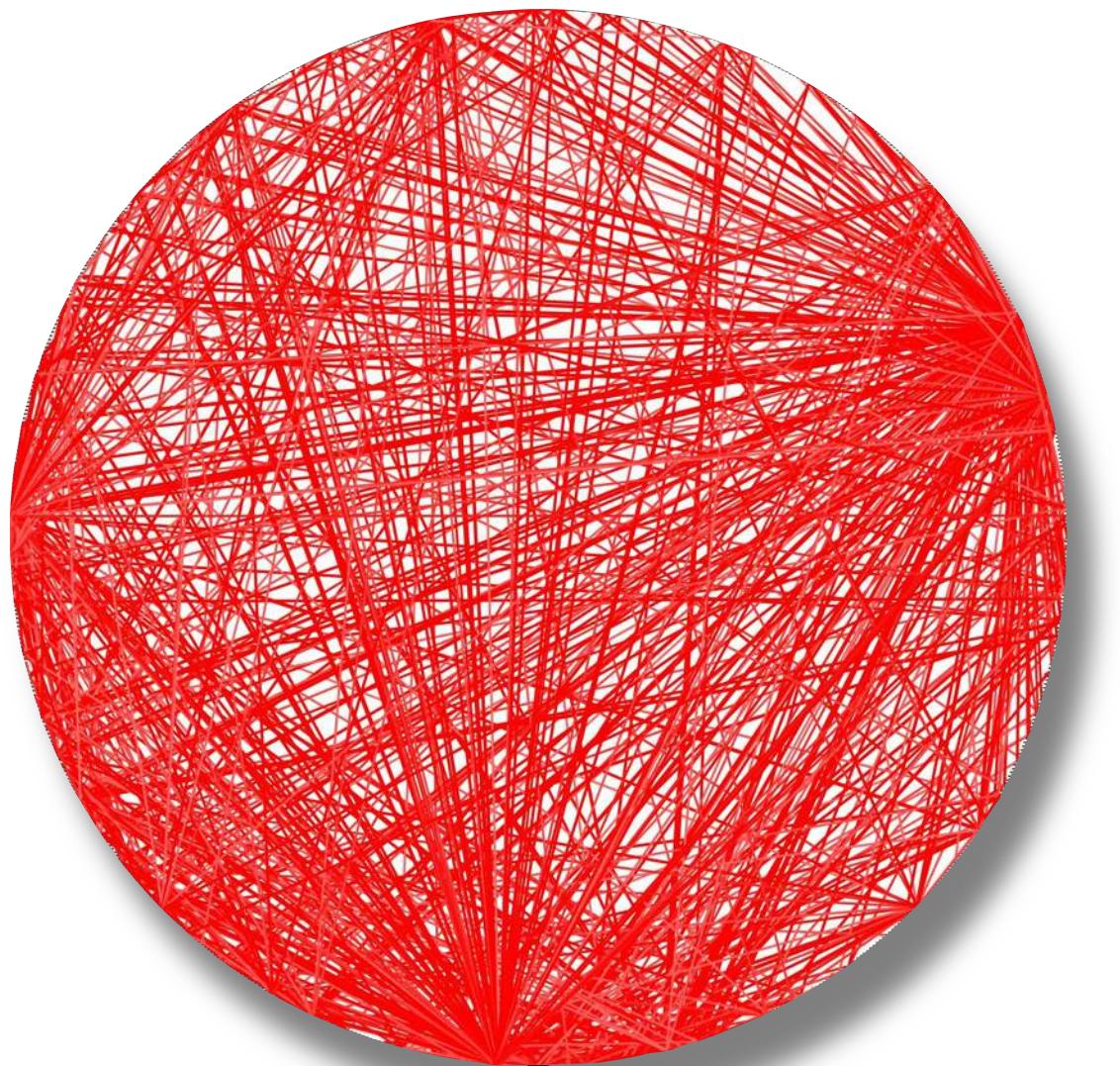
Monolith



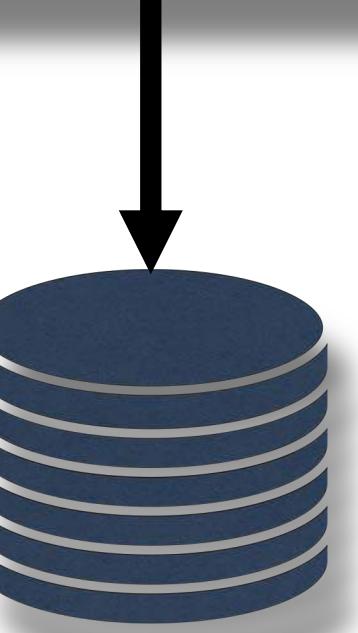
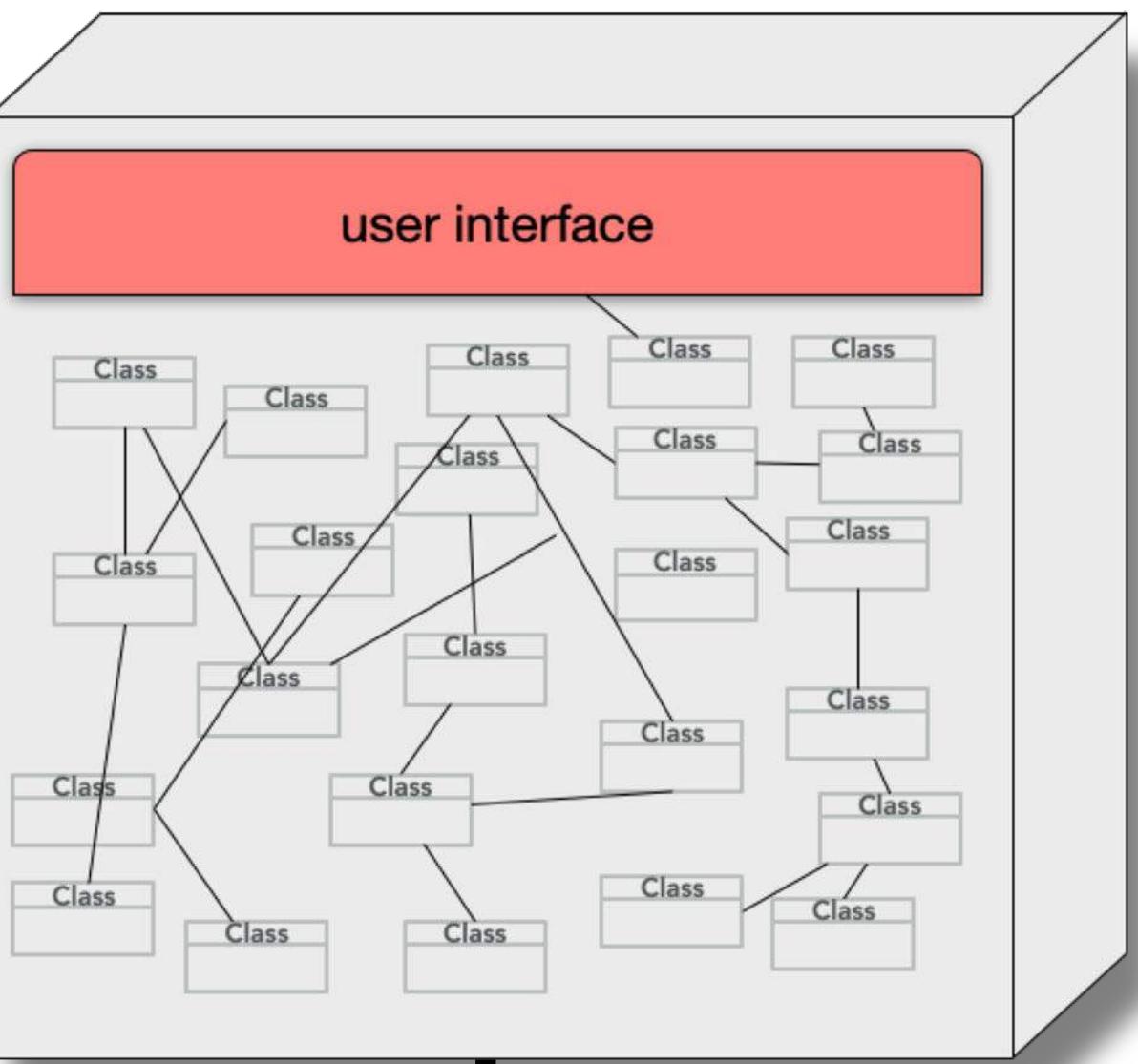
Service-based

Monoliths





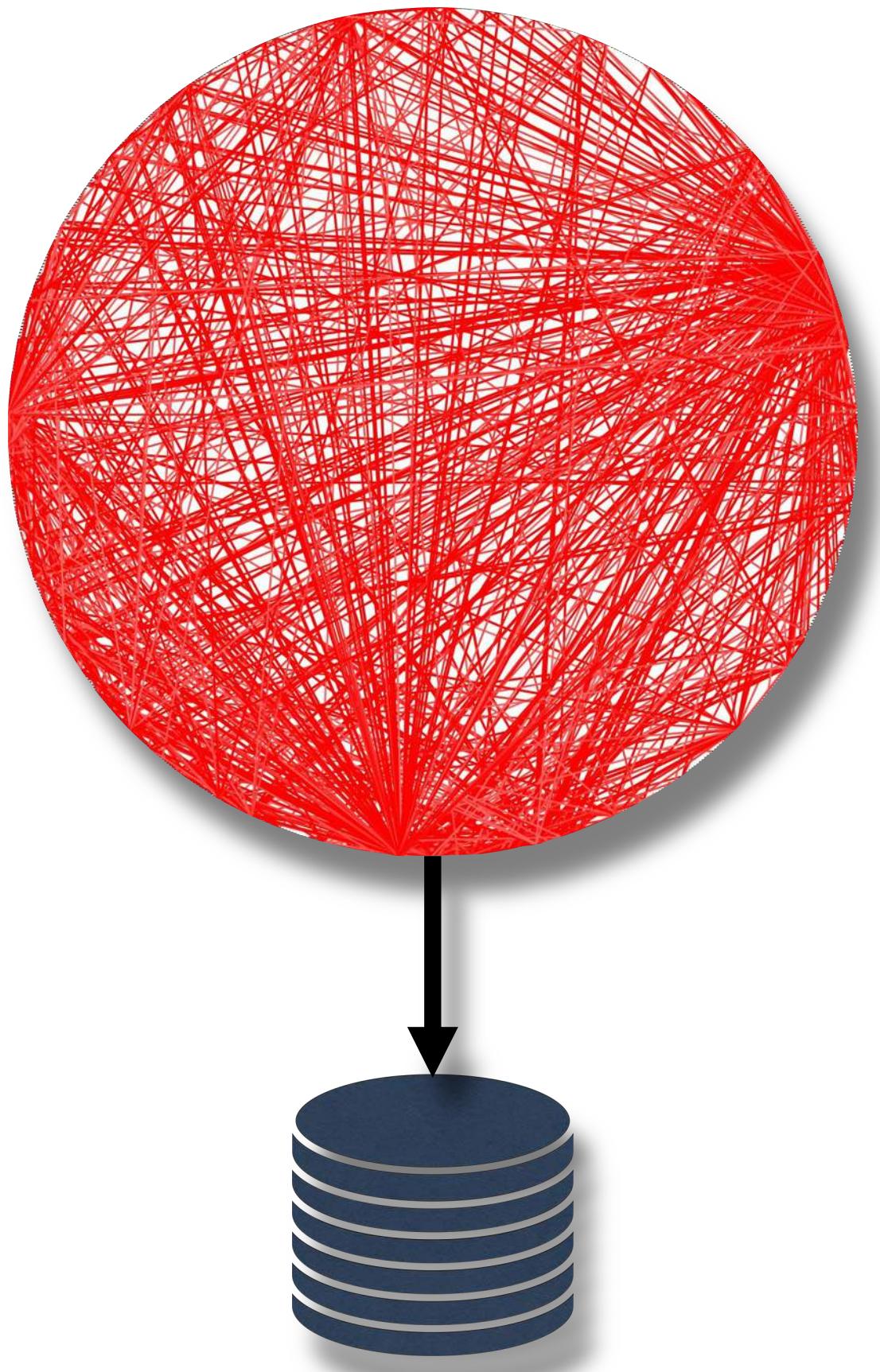
monoliths



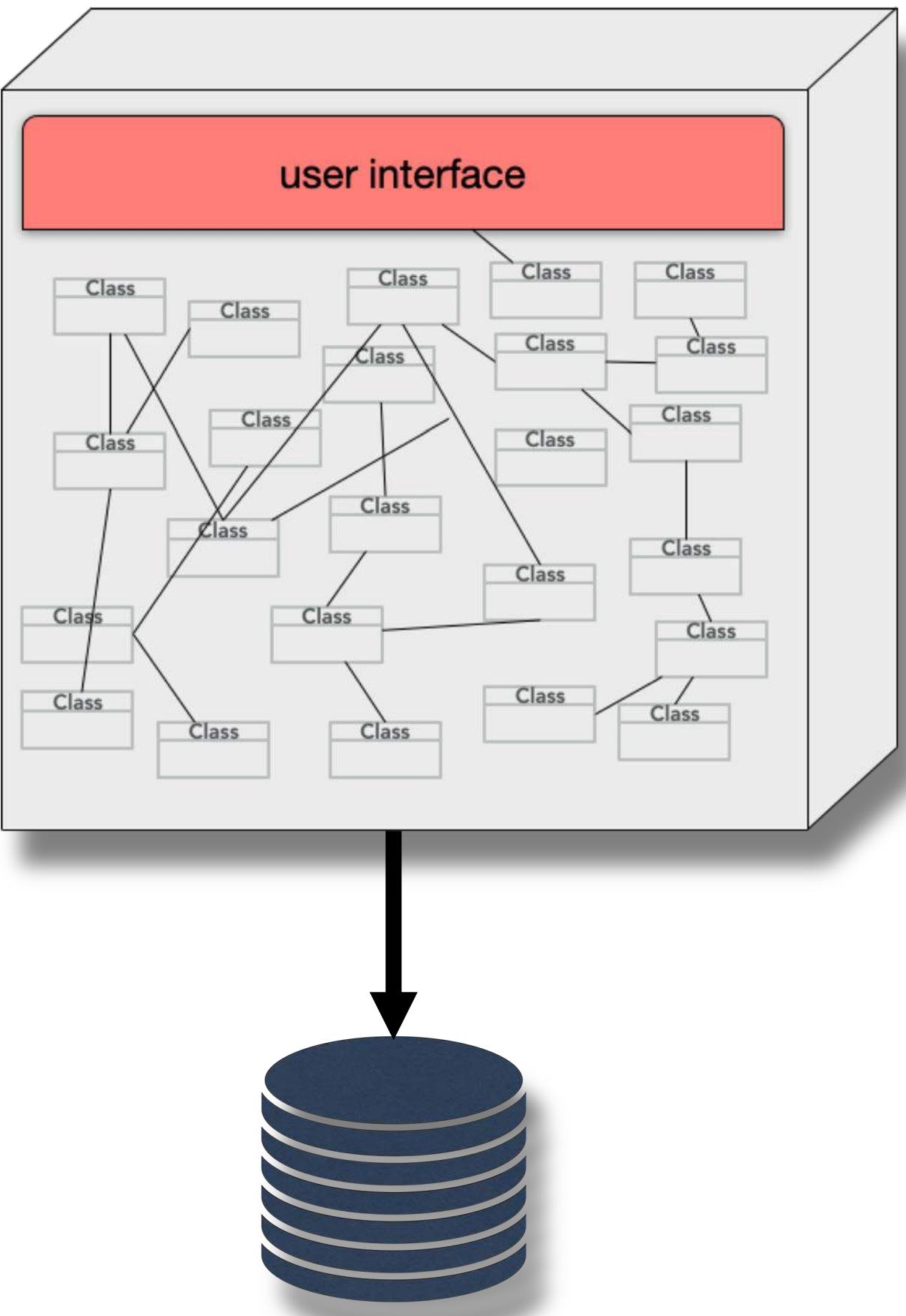
“Big Ball of Mud”

https://en.wikipedia.org/wiki/Big_ball_of_mud

unstructured
monolith



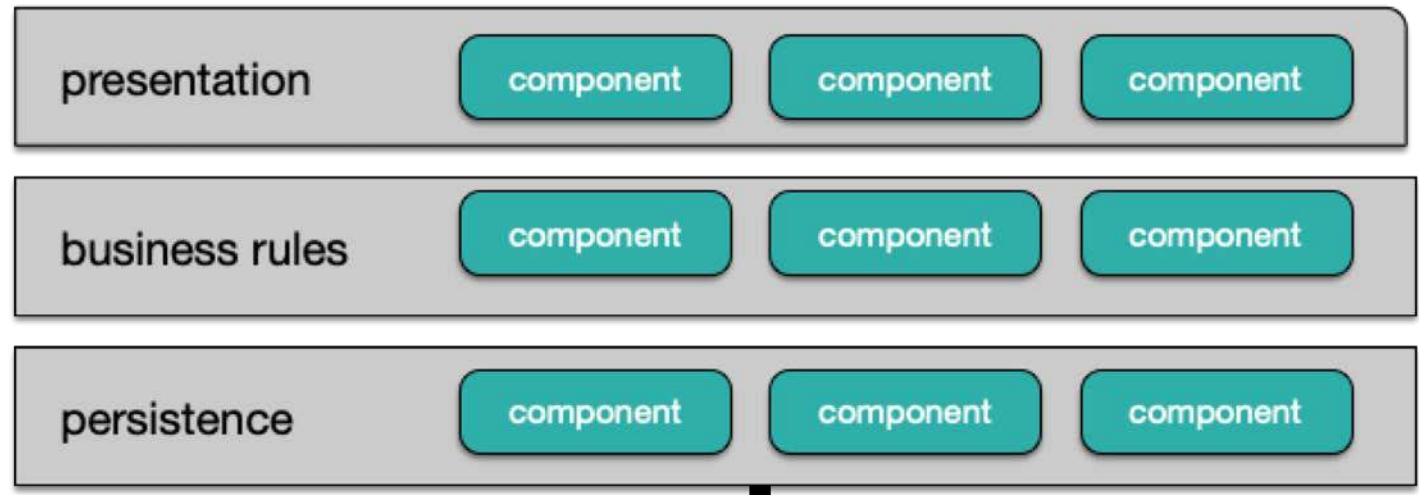
monoliths



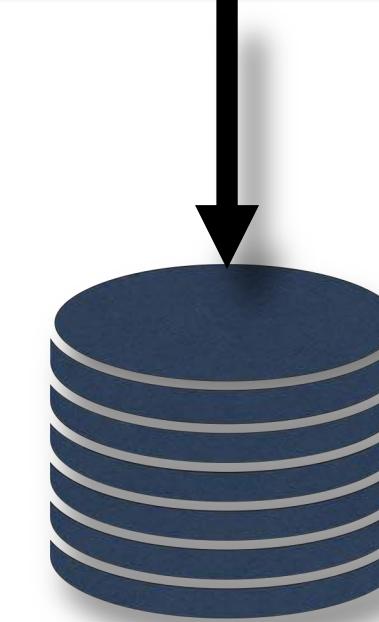
“Big Ball of Mud”

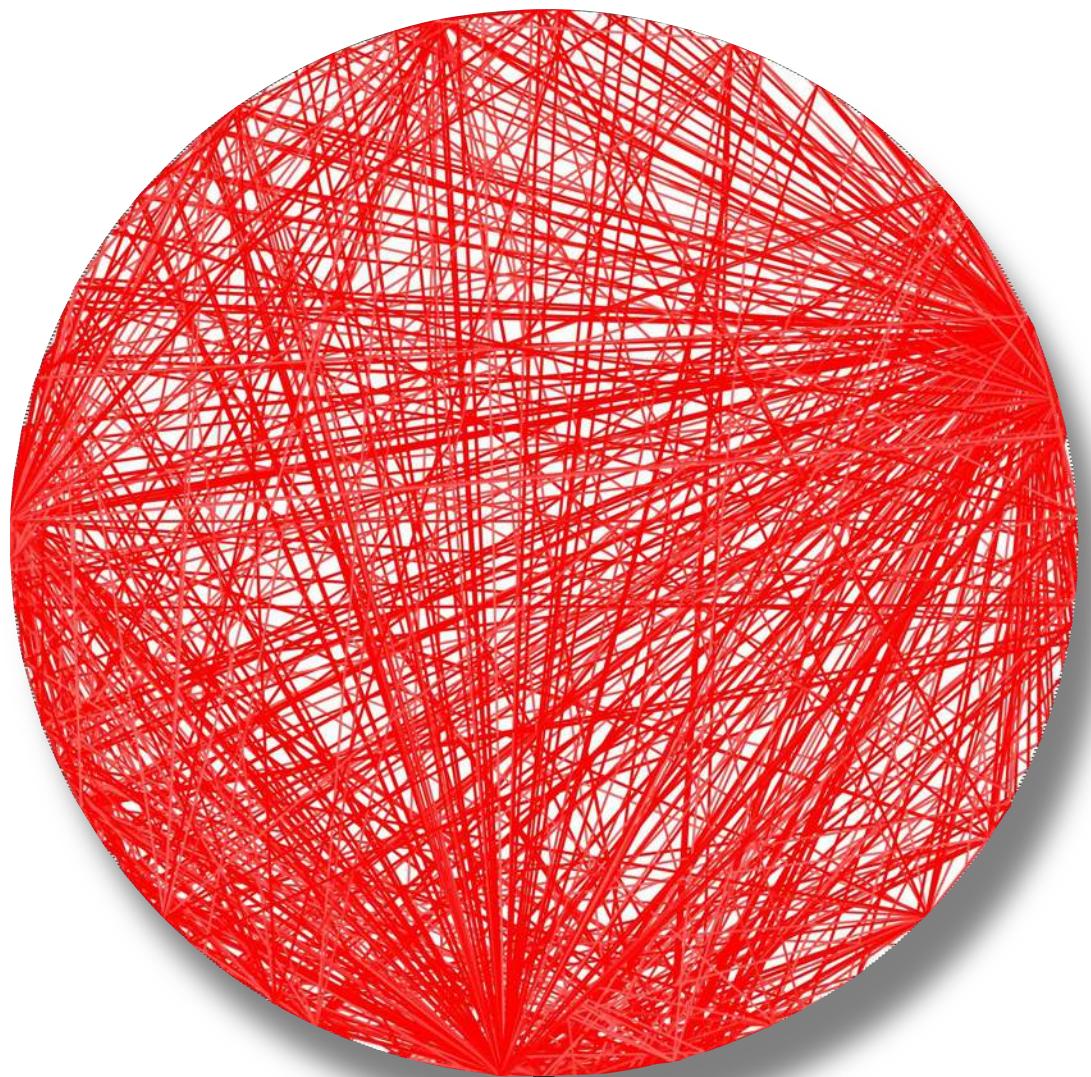
https://en.wikipedia.org/wiki/Big_ball_of_mud

unstructured
monolith

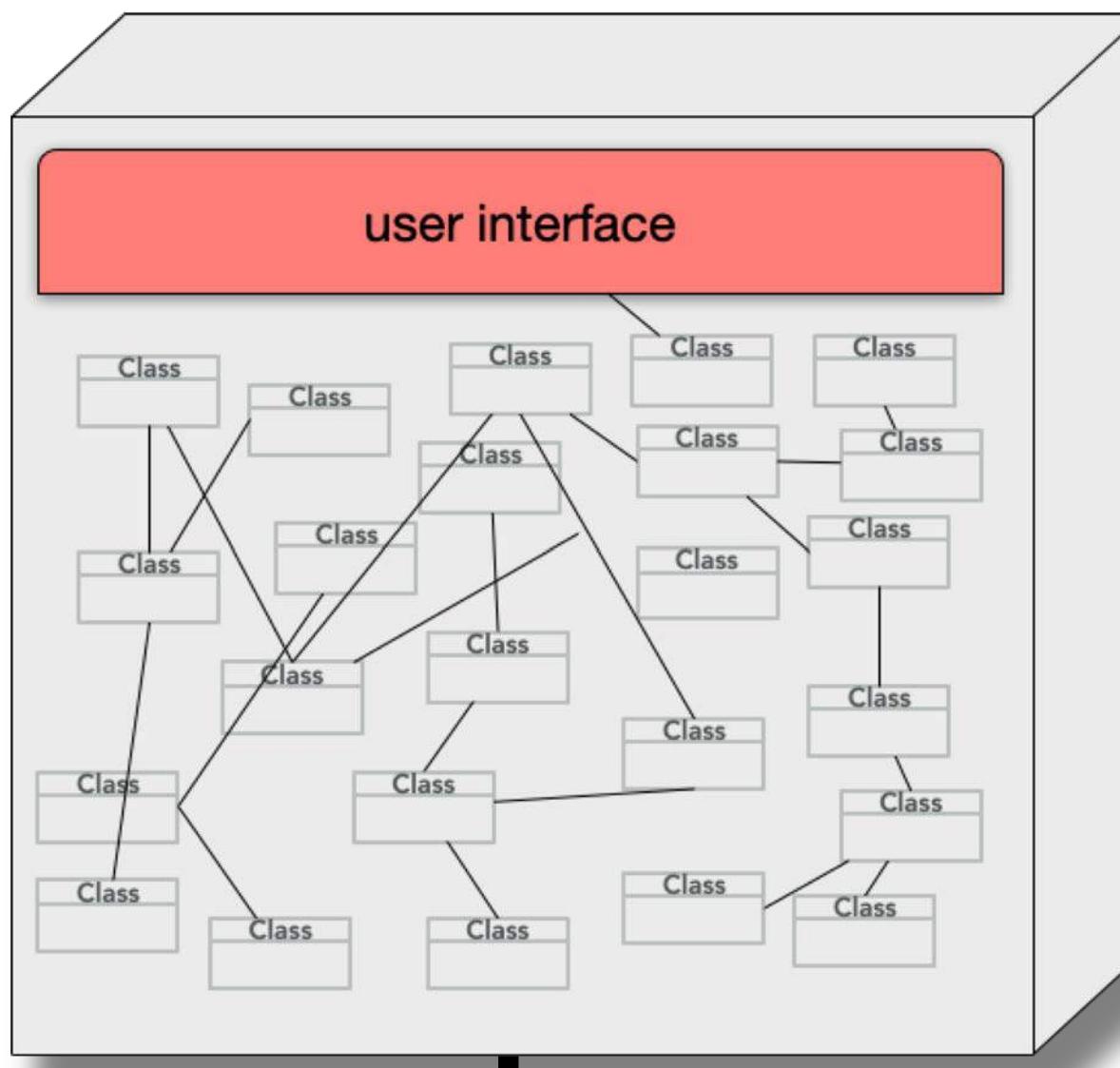


layered
architecture



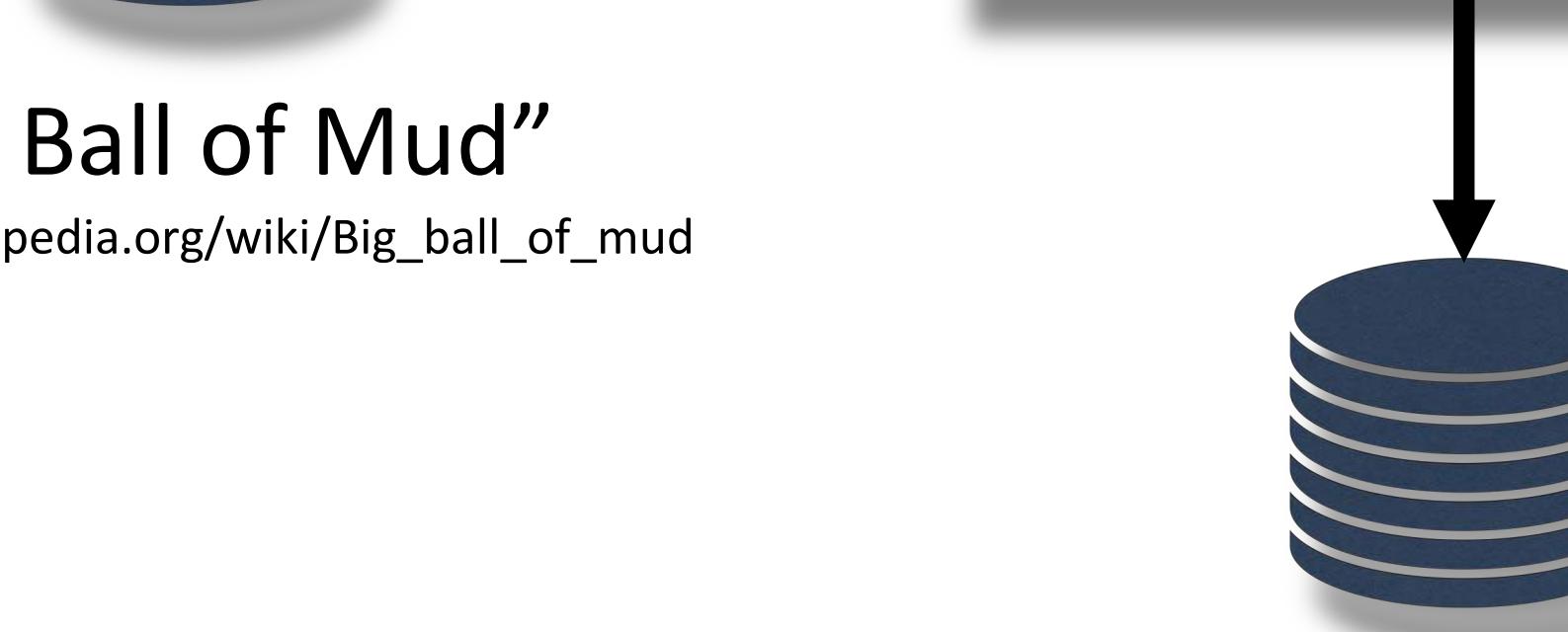


monoliths

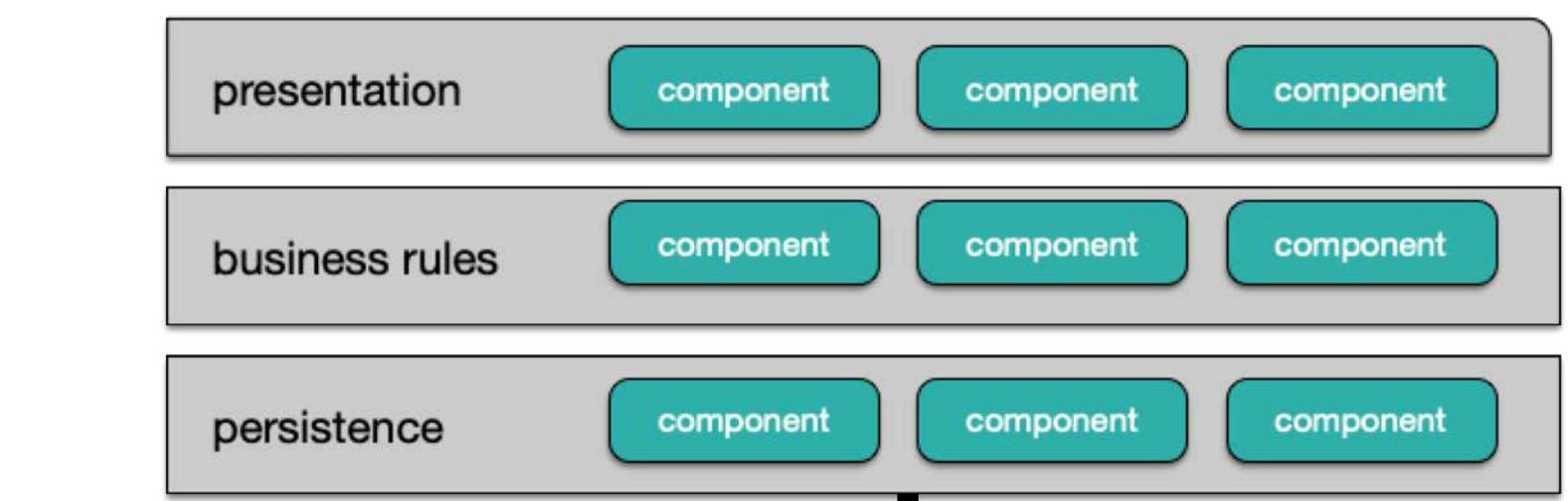


unstructured
monolith

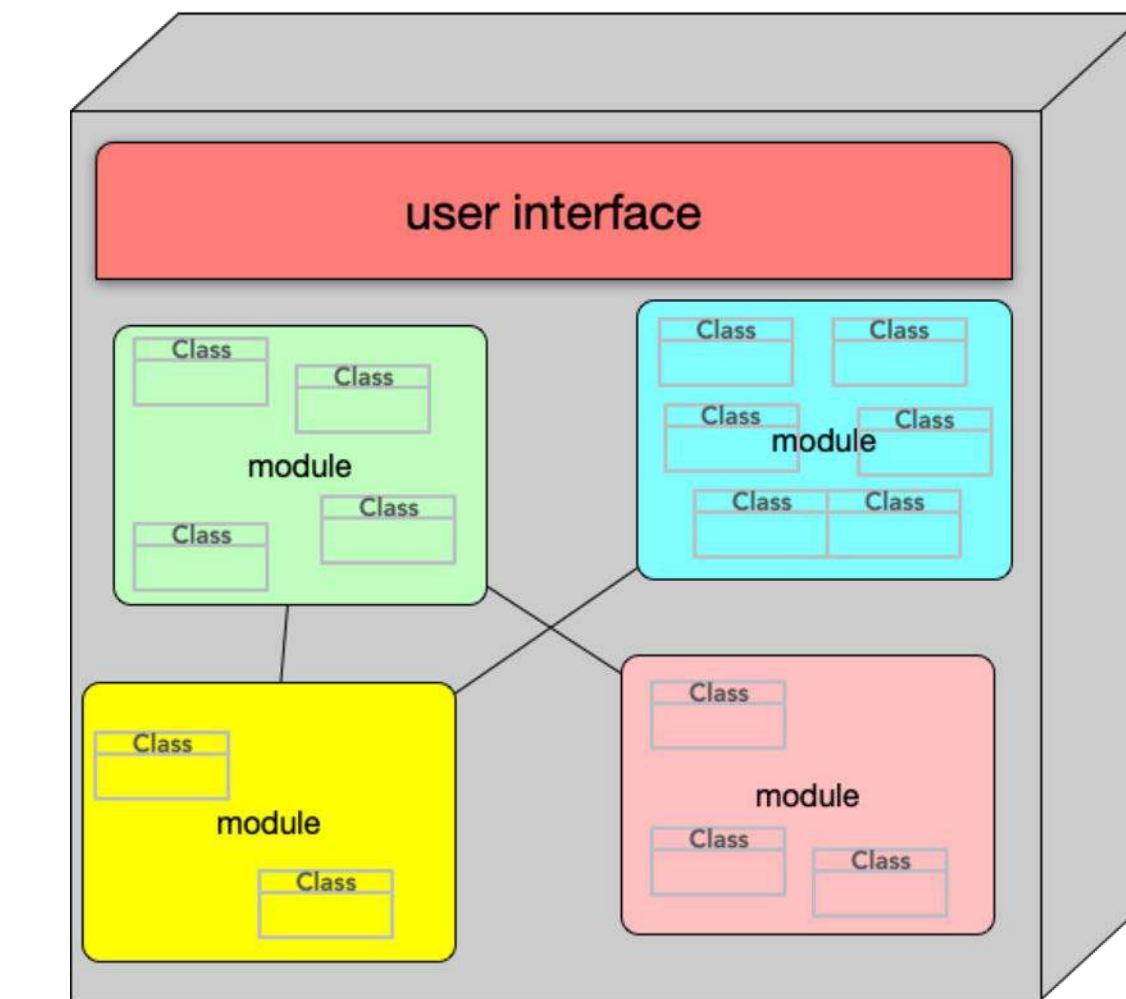
“Big Ball of Mud”
https://en.wikipedia.org/wiki/Big_ball_of_mud



unstructured
monolith

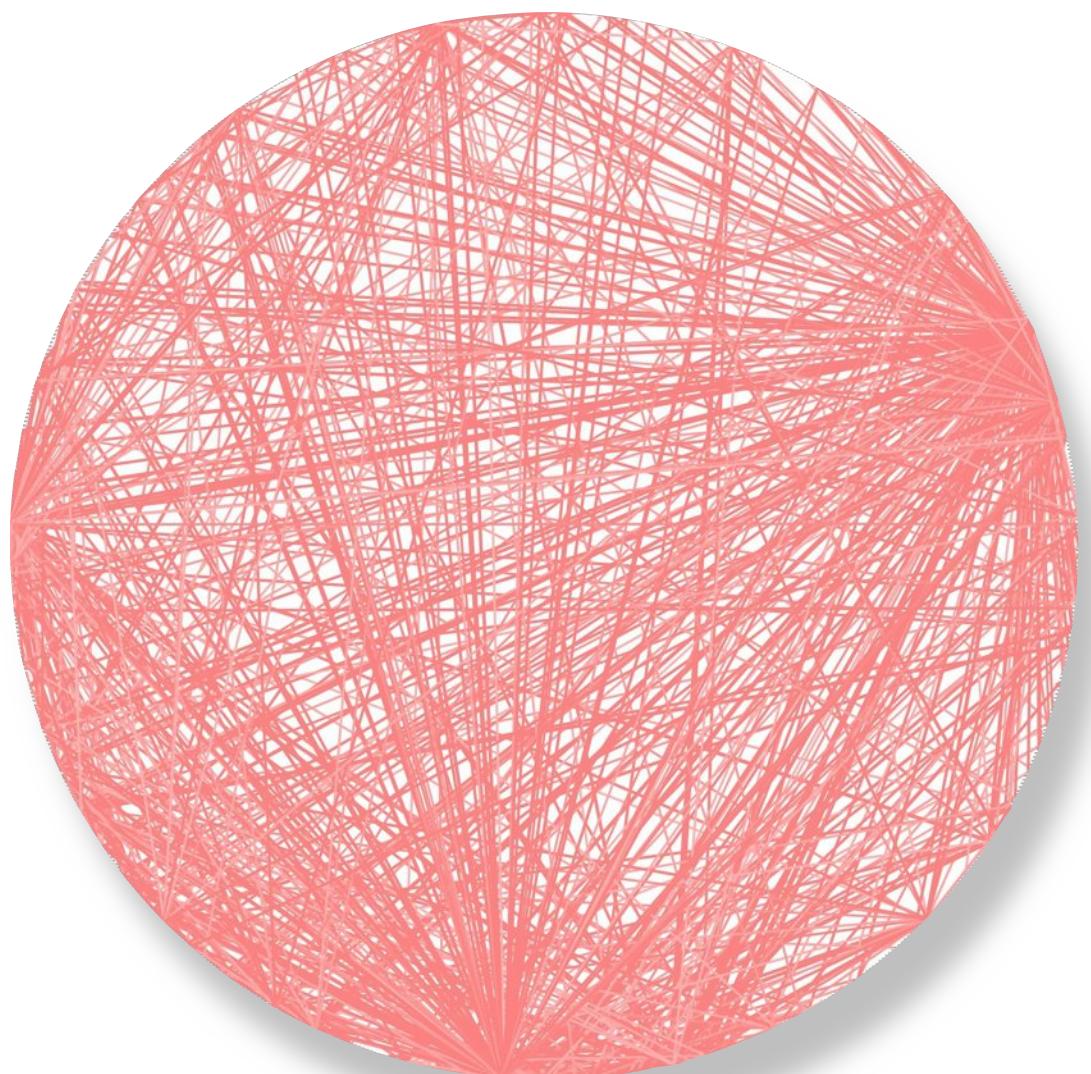


layered
architecture



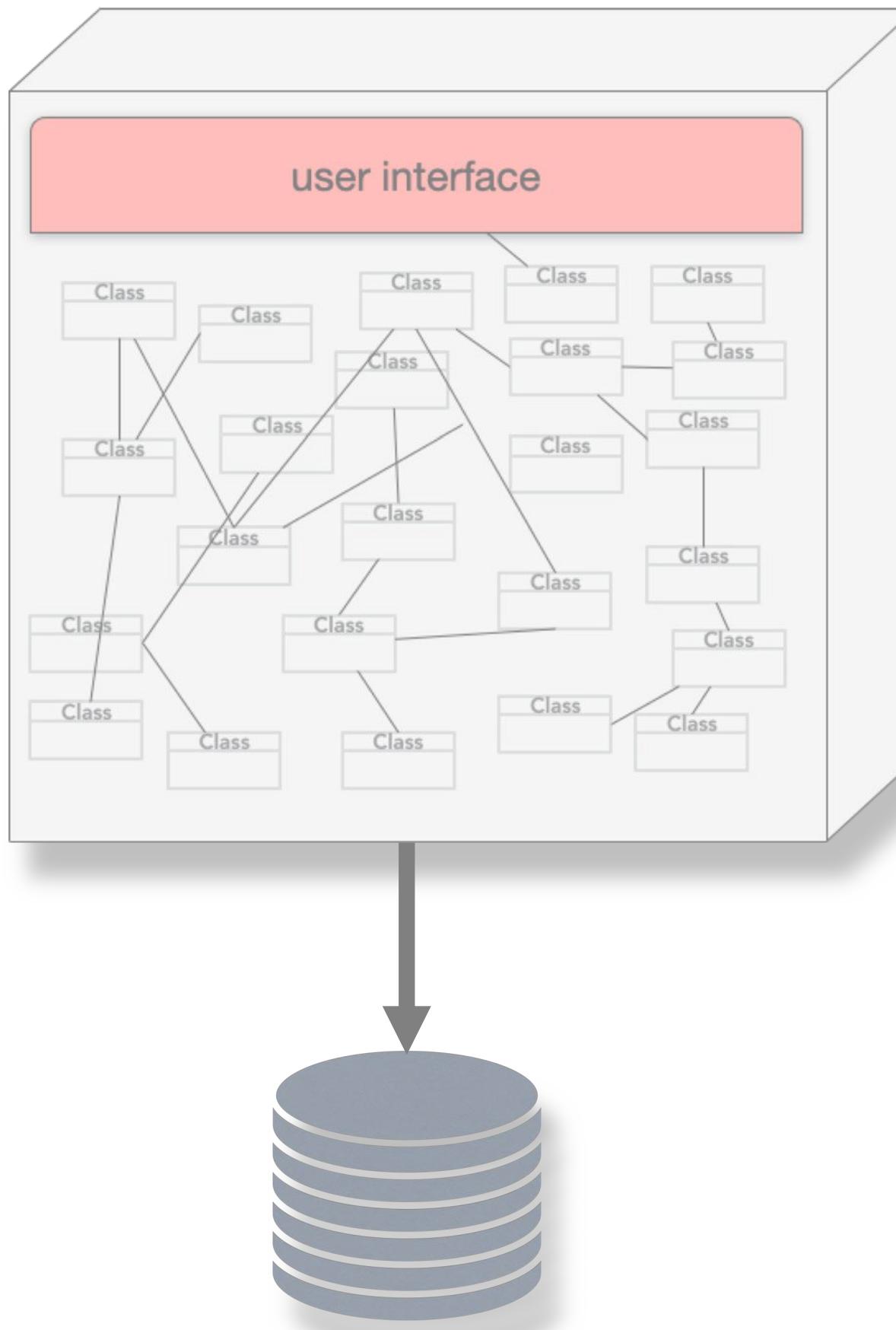
<http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths>

modular monolith



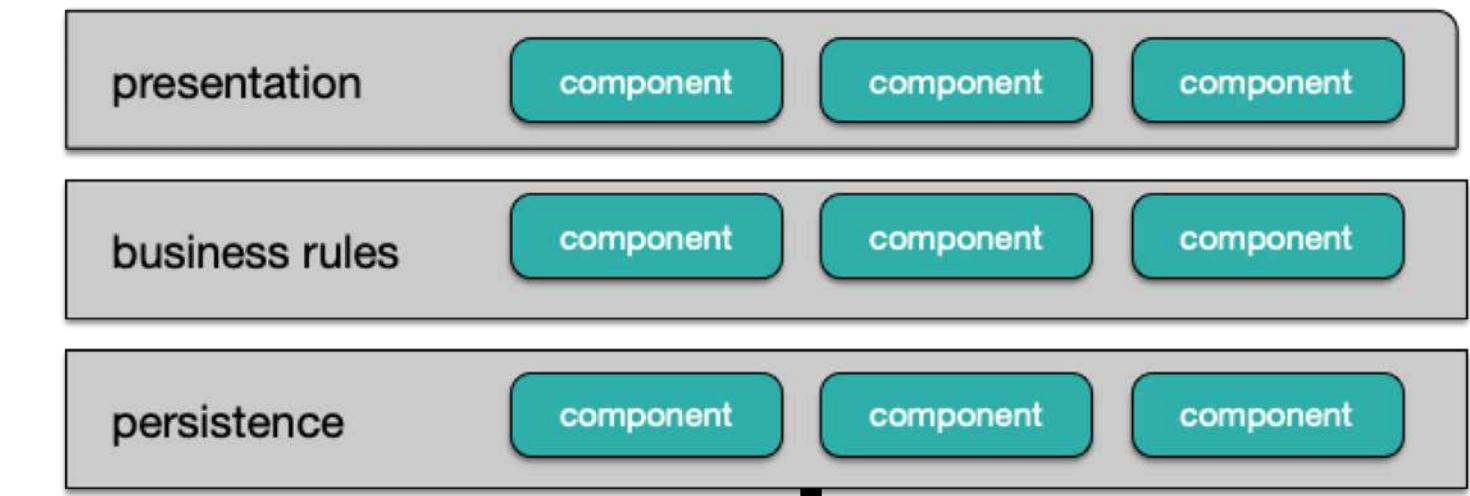
“Big Ball of Mud”

https://en.wikipedia.org/wiki/Big_ball_of_mud

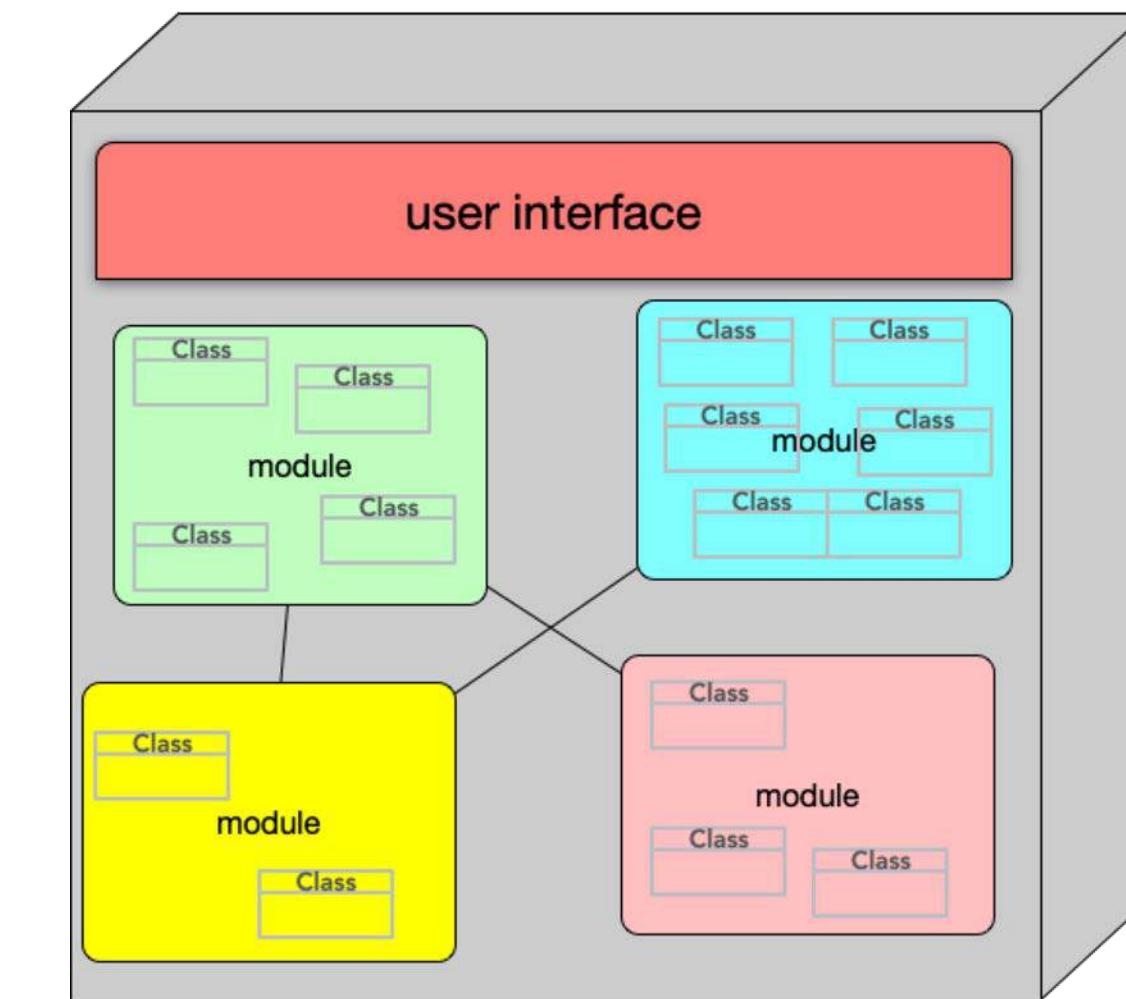


unstructured
monolith

monoliths



layered
architecture

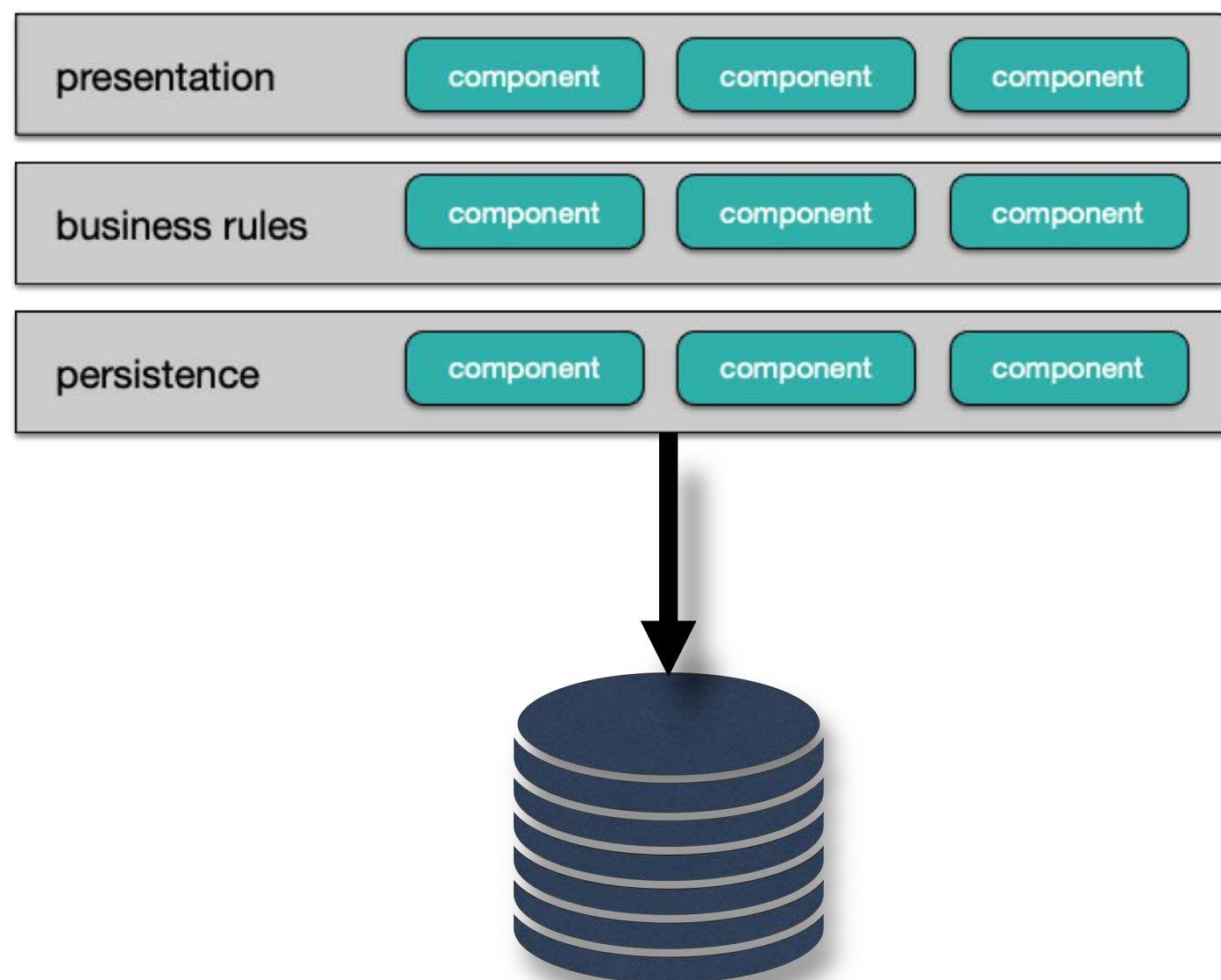


modular monolith

<http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths>

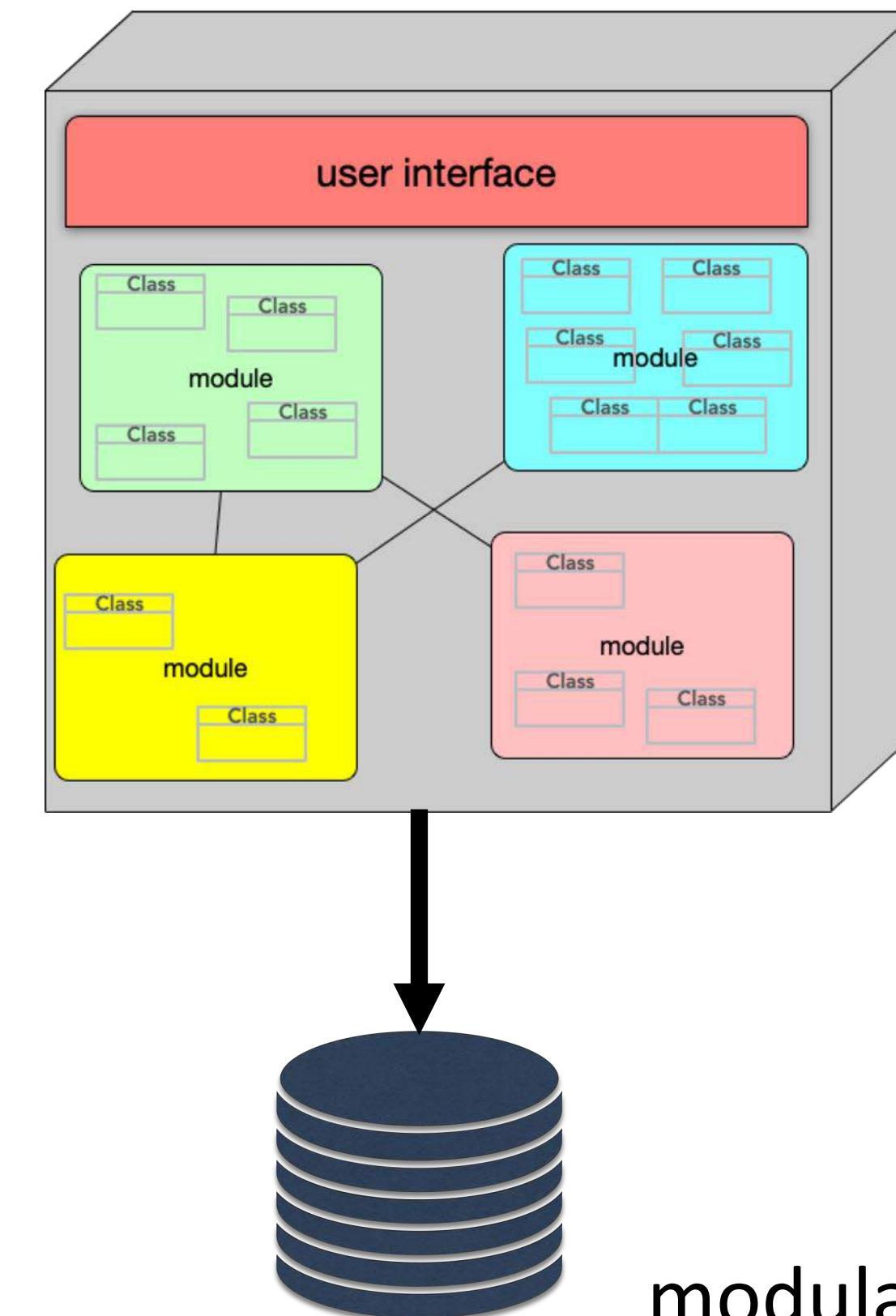
monoliths

technical partitioning



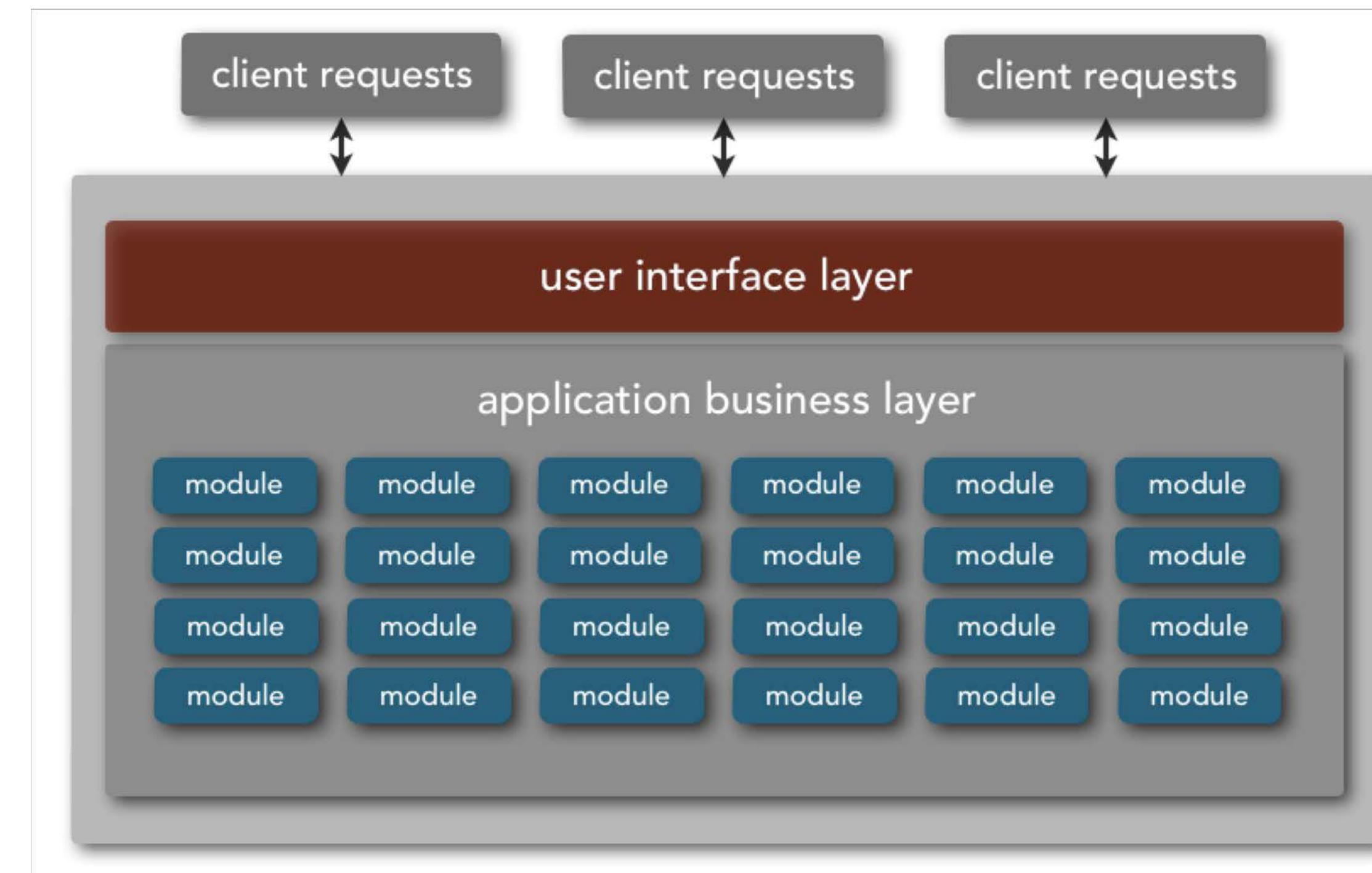
layered
architecture

domain partitioning



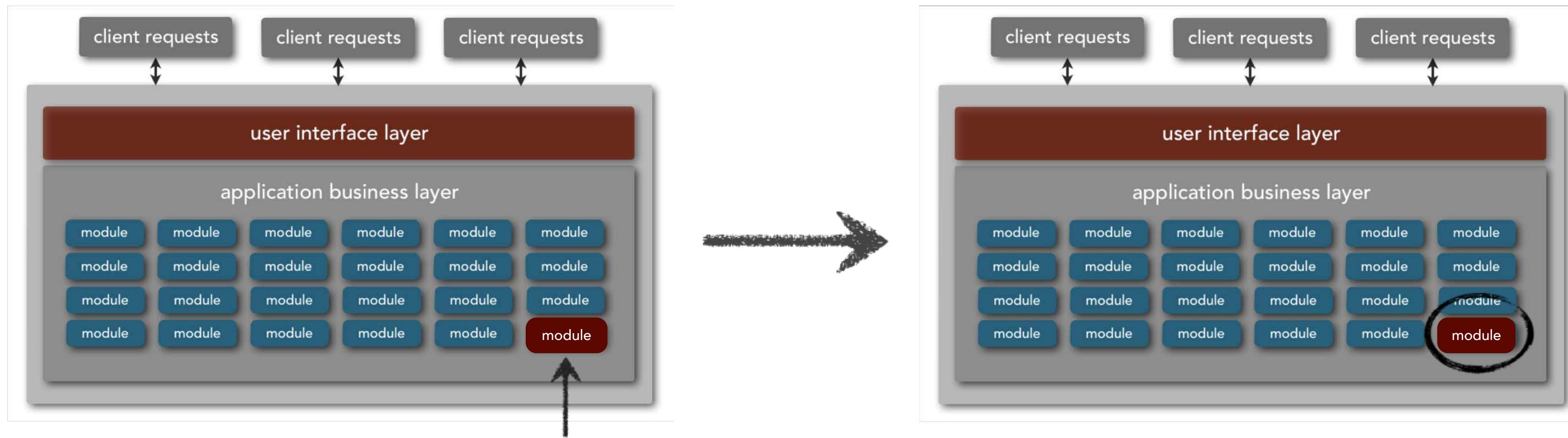
modular monolith

monolithic application issues



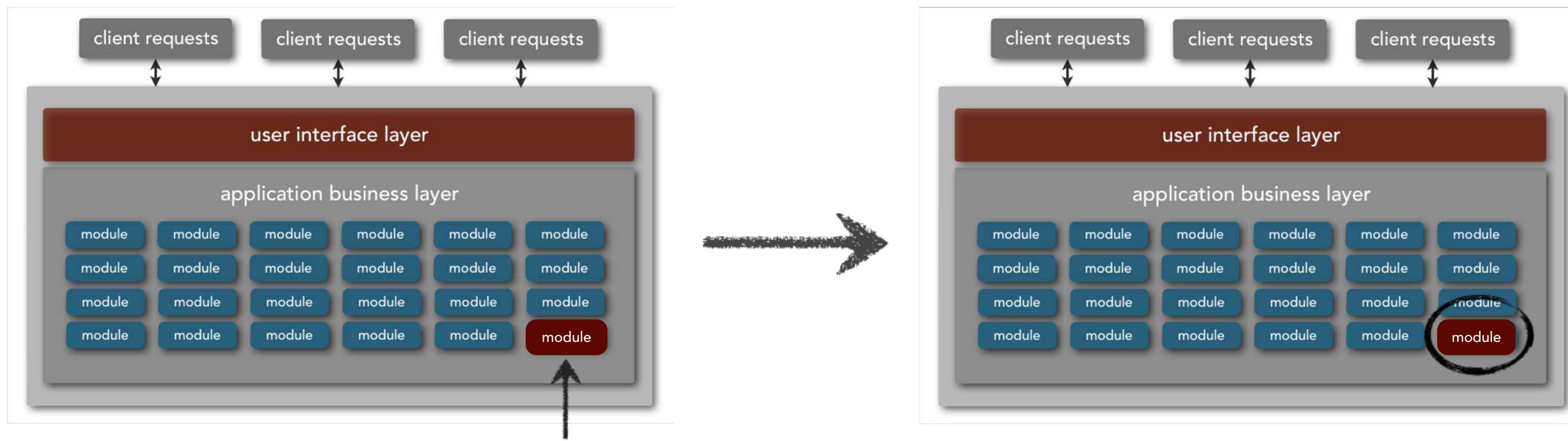
monolithic application issues

deployment



monolithic application issues

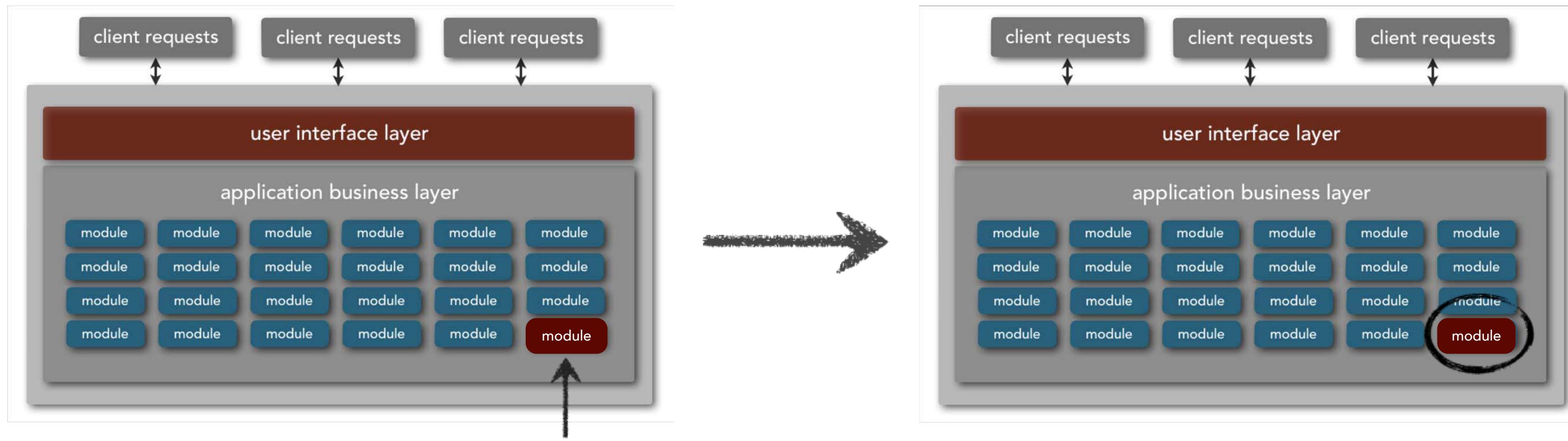
deployment



entire application must be deployed for a small change

monolithic application issues

deployment

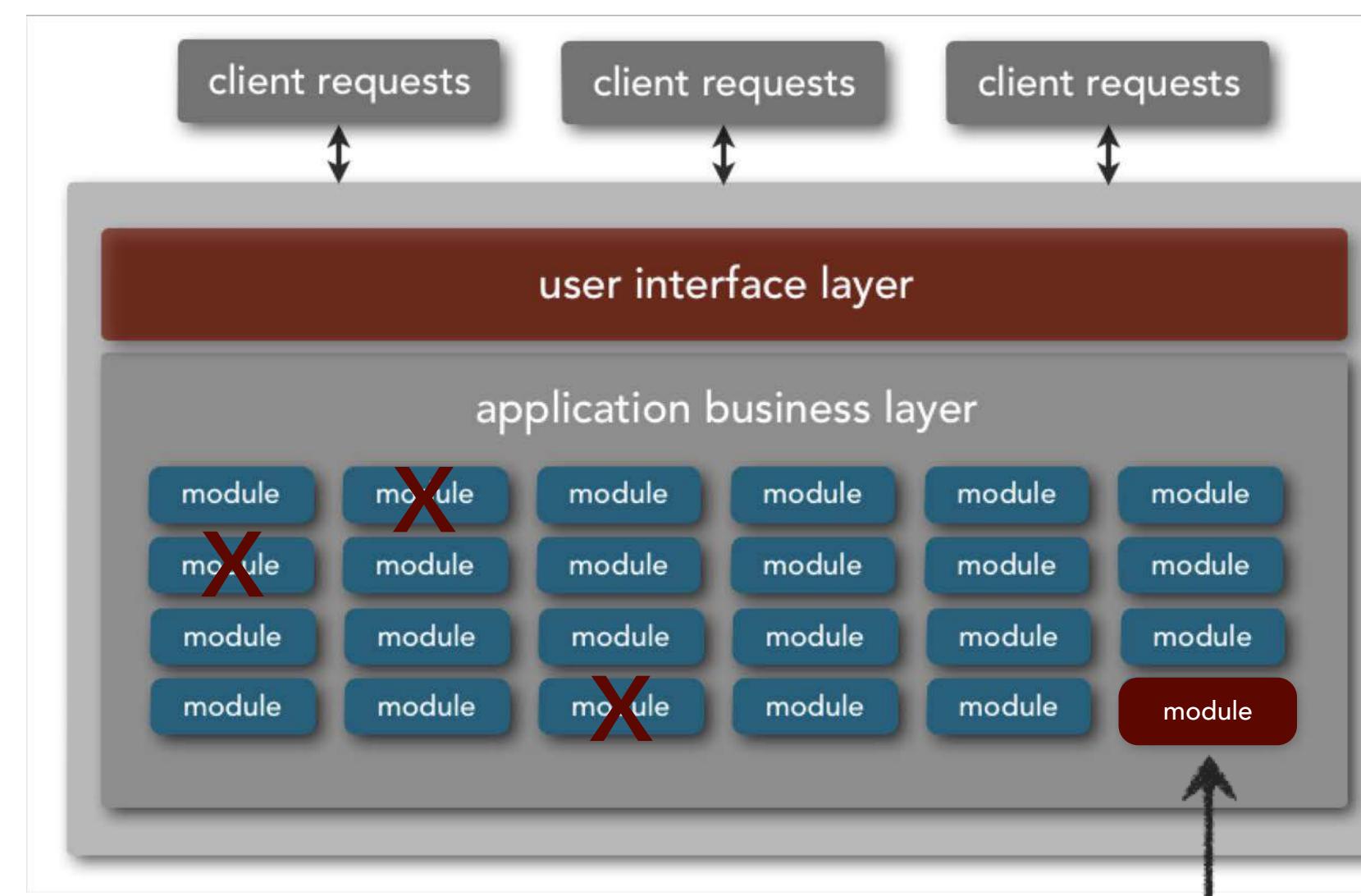


entire application must be deployed for a small change

scheduling and coordination challenges can impact deployment

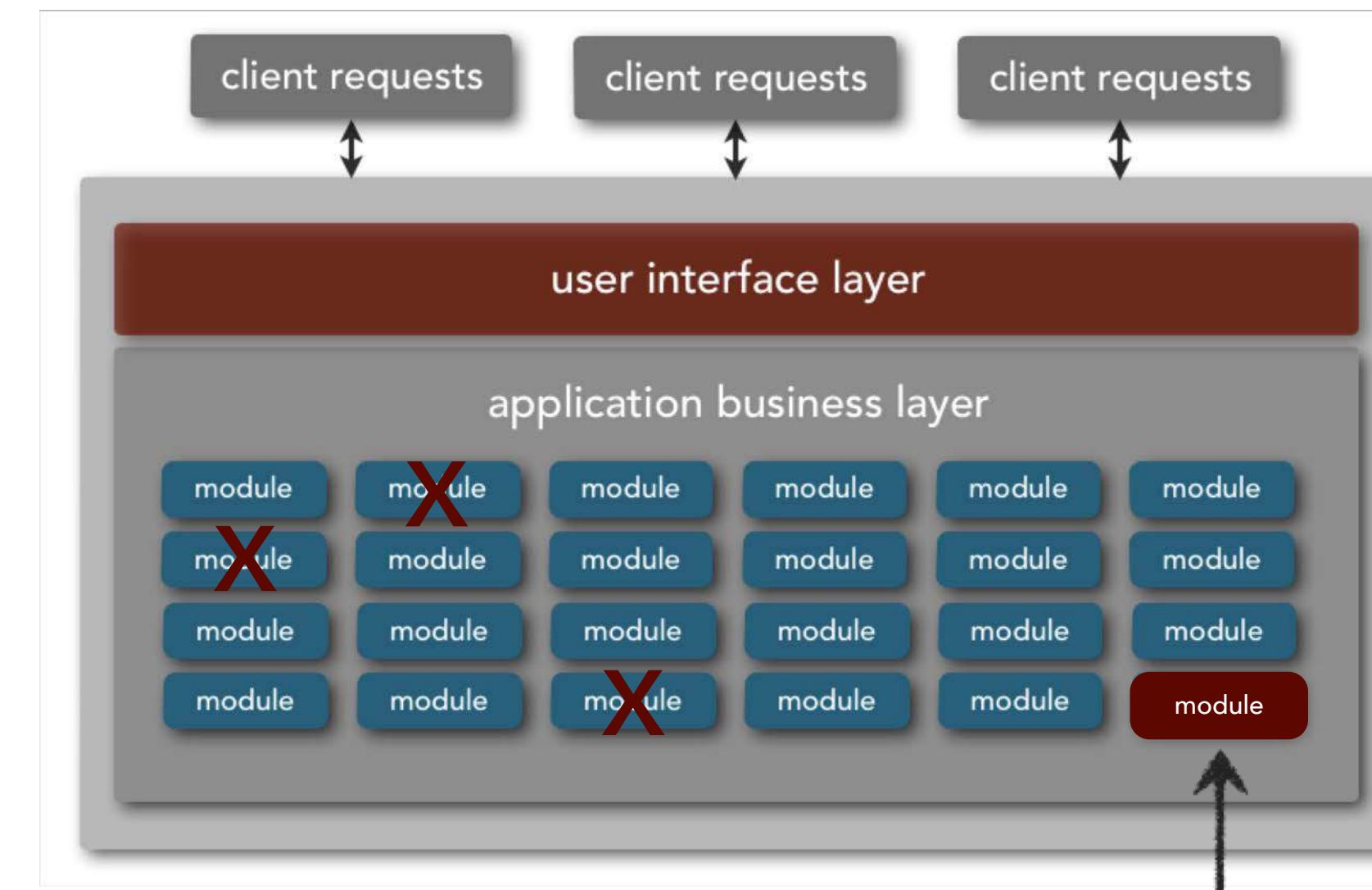
monolithic application issues

reliability and robustness



monolithic application issues

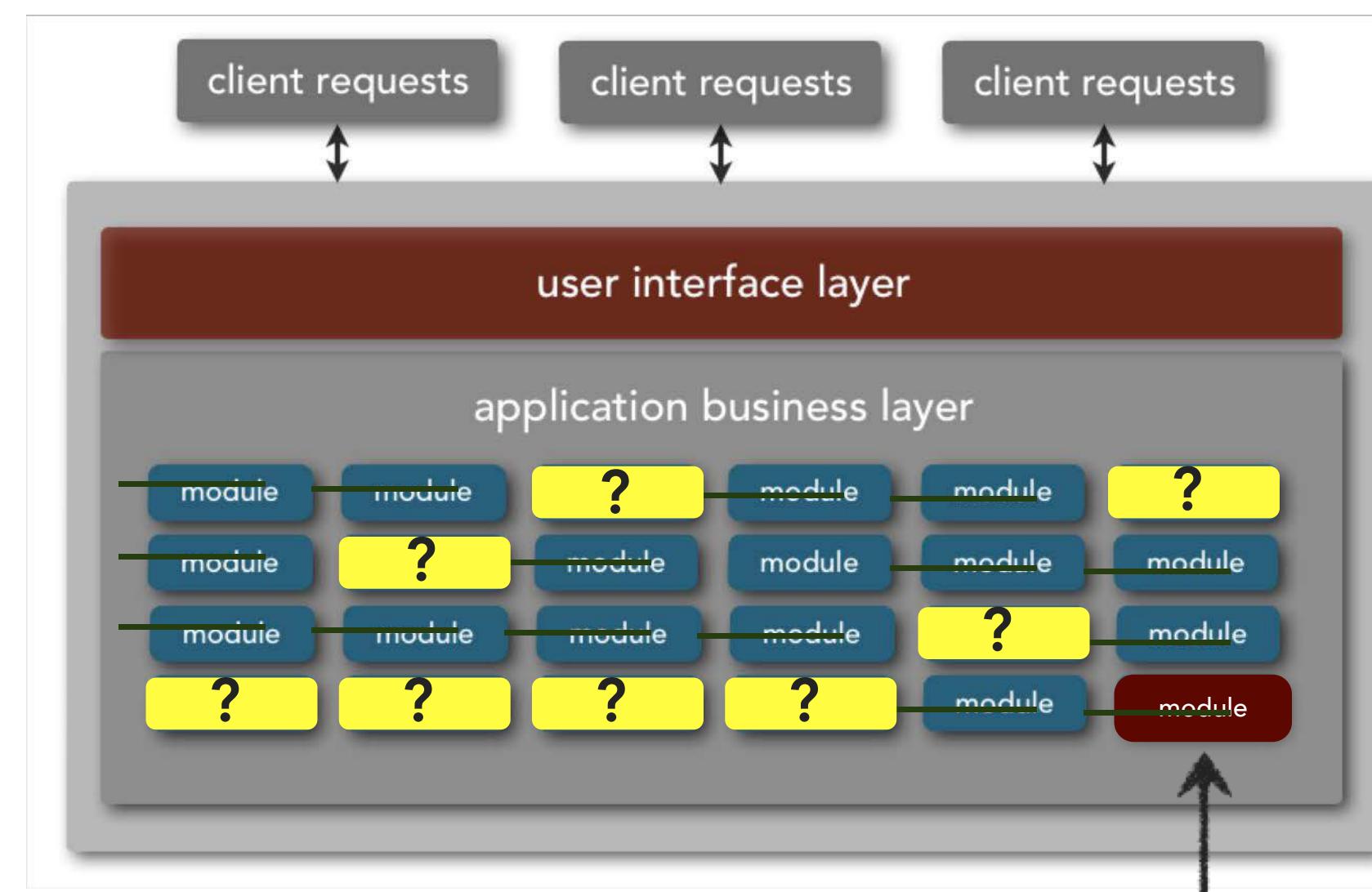
reliability and robustness



due to tight coupling, changes in one area of the application adversely affects other areas

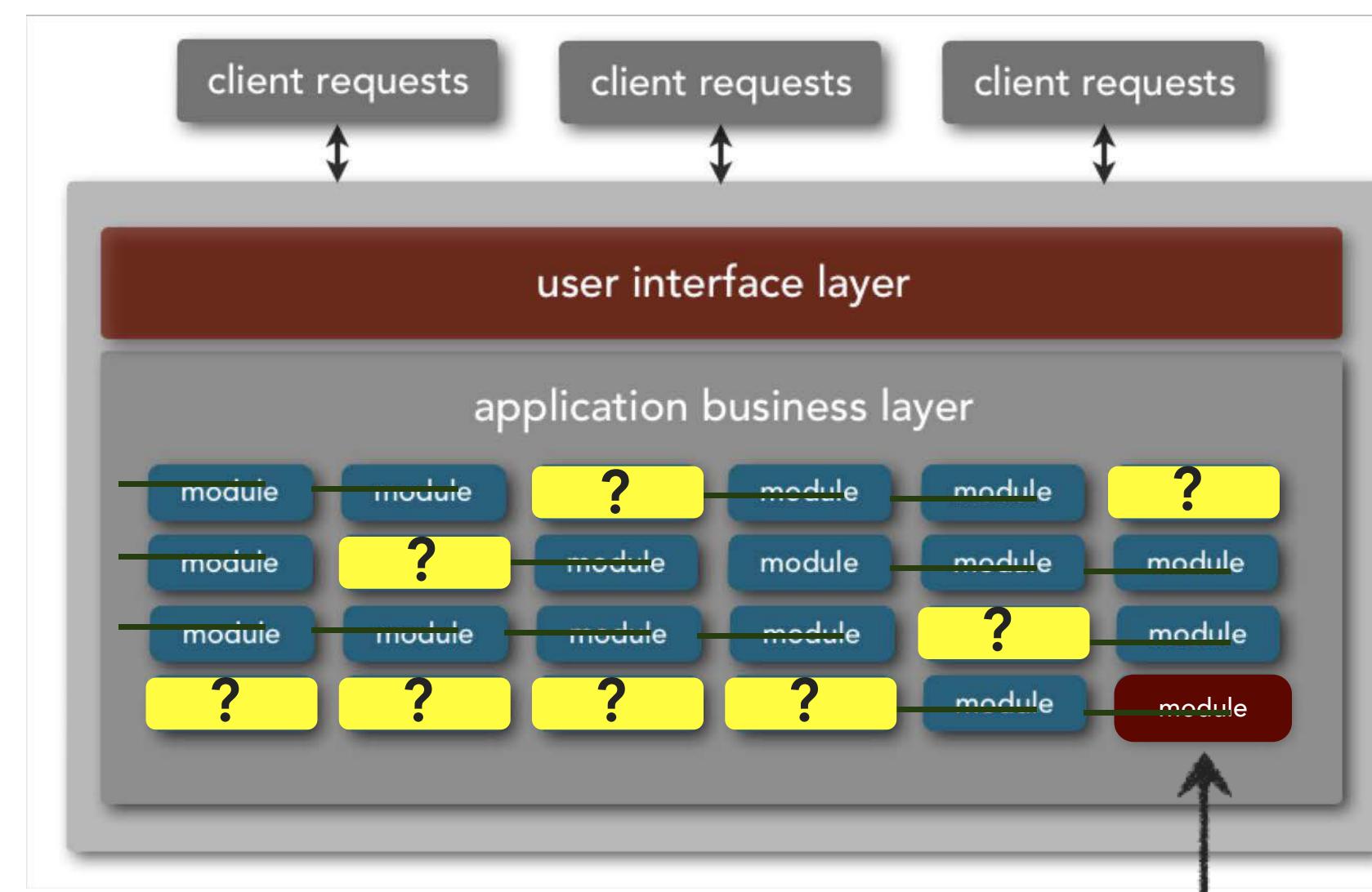
monolithic application issues

testability



monolithic application issues

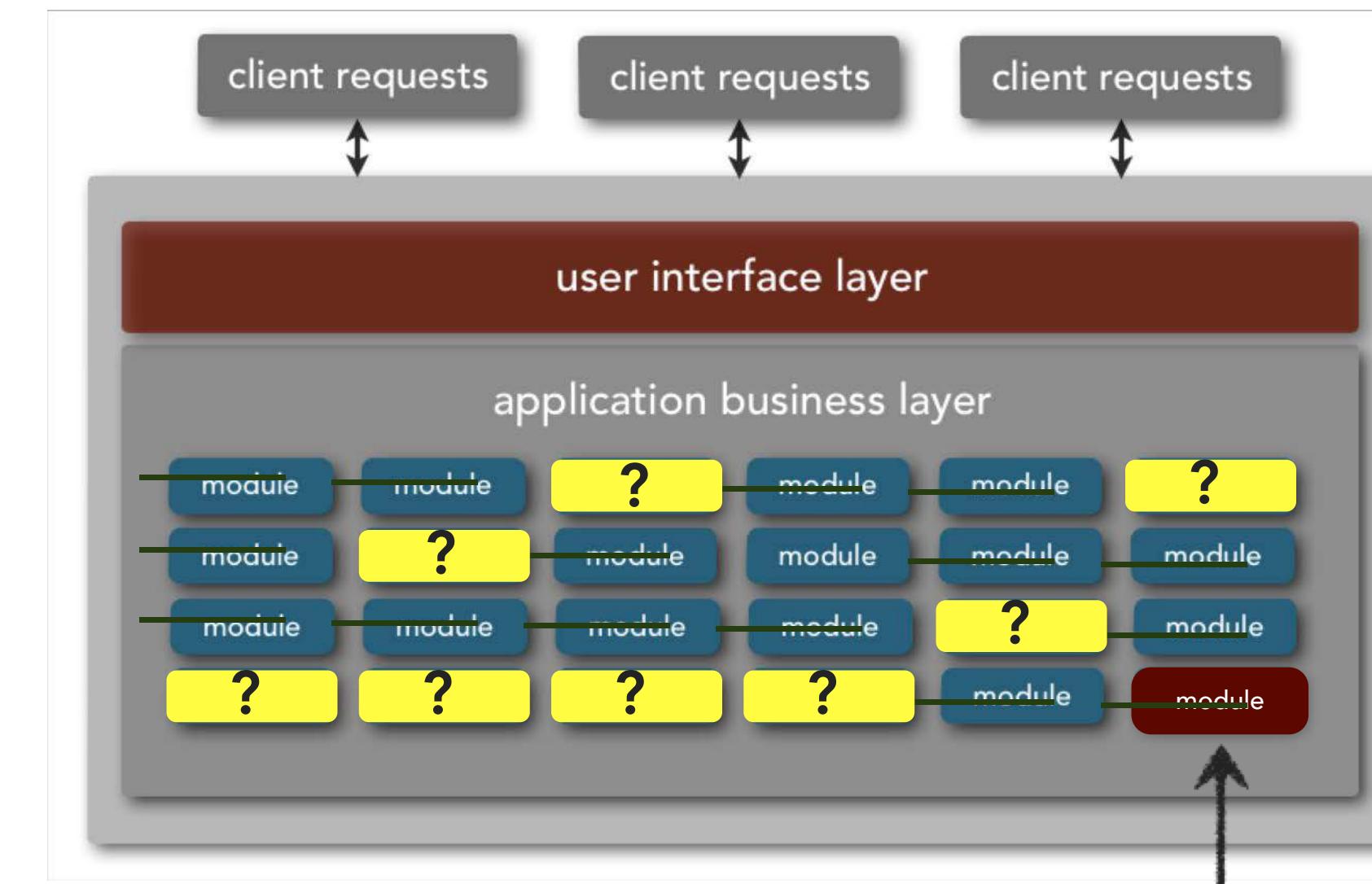
testability



poor test coverage for entire application

monolithic application issues

testability

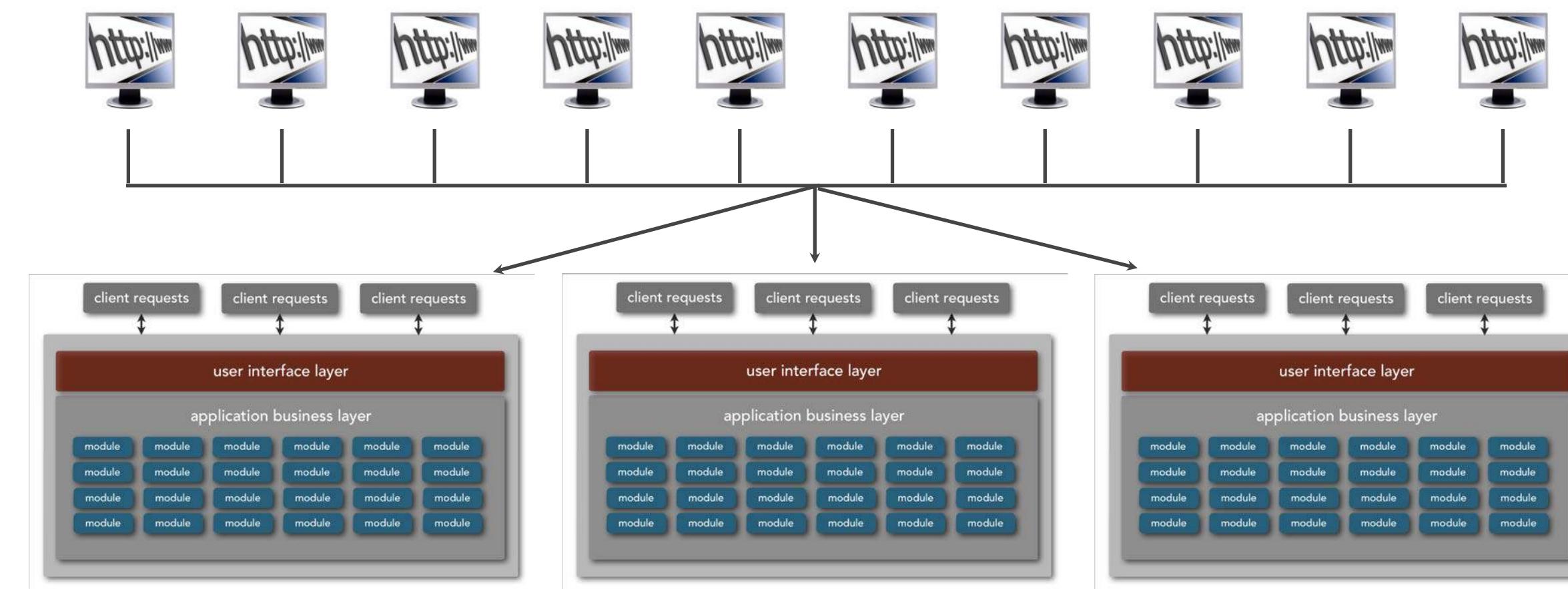


poor test coverage for entire application

low confidence level that nothing else is impacted by the change

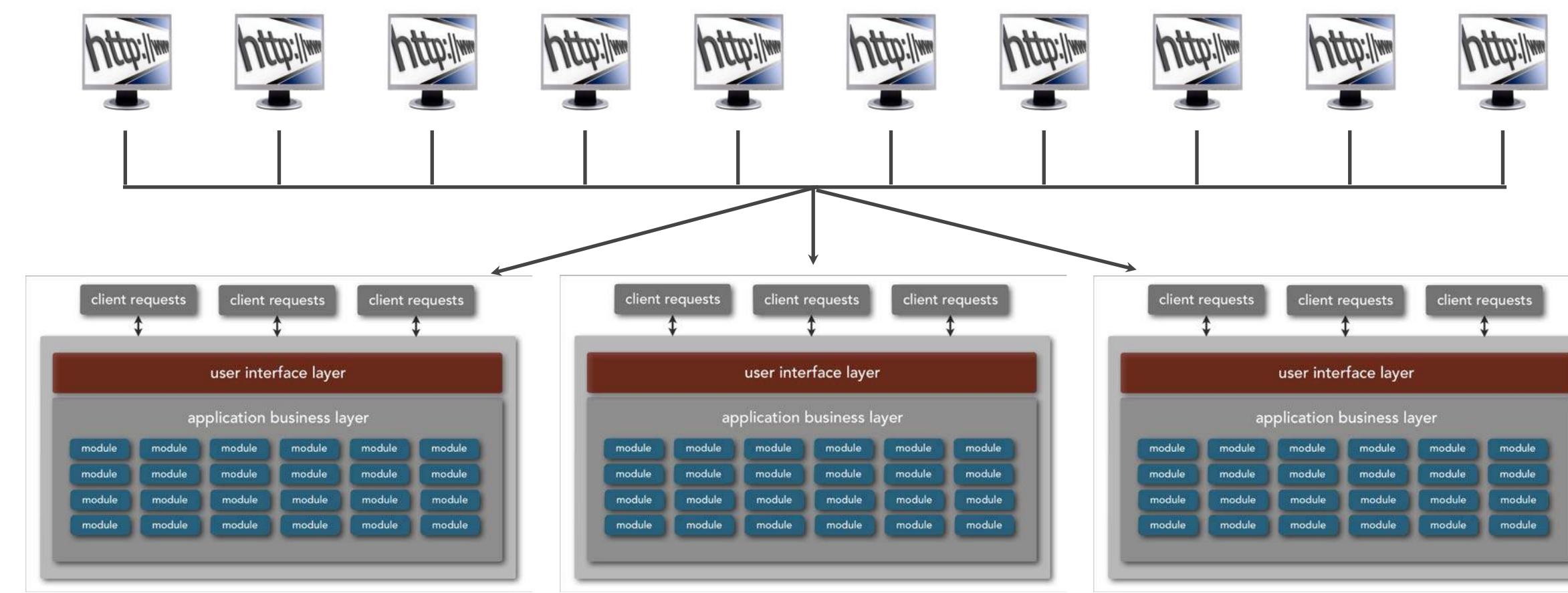
monolithic application issues

scalability



monolithic application issues

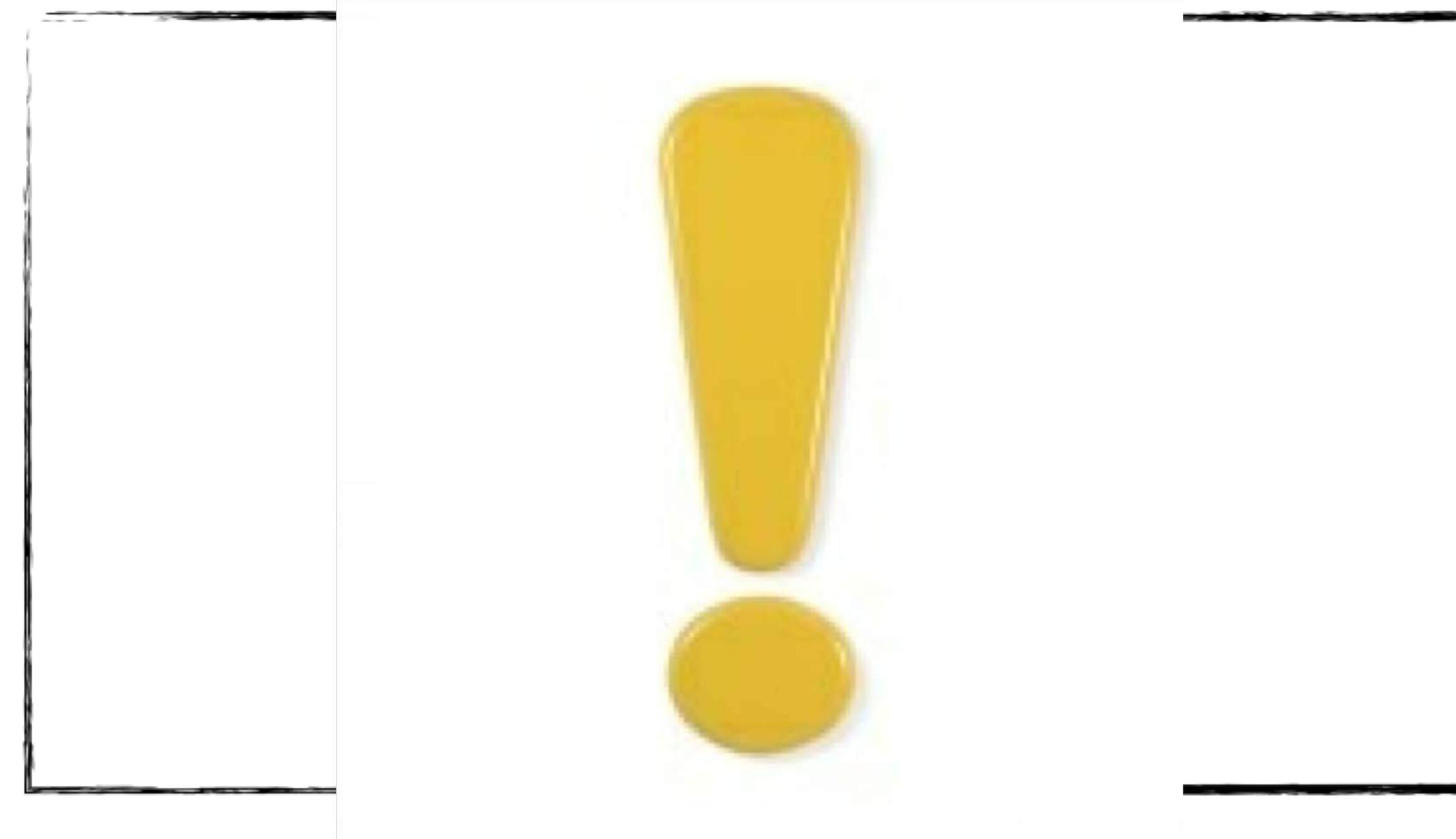
scalability



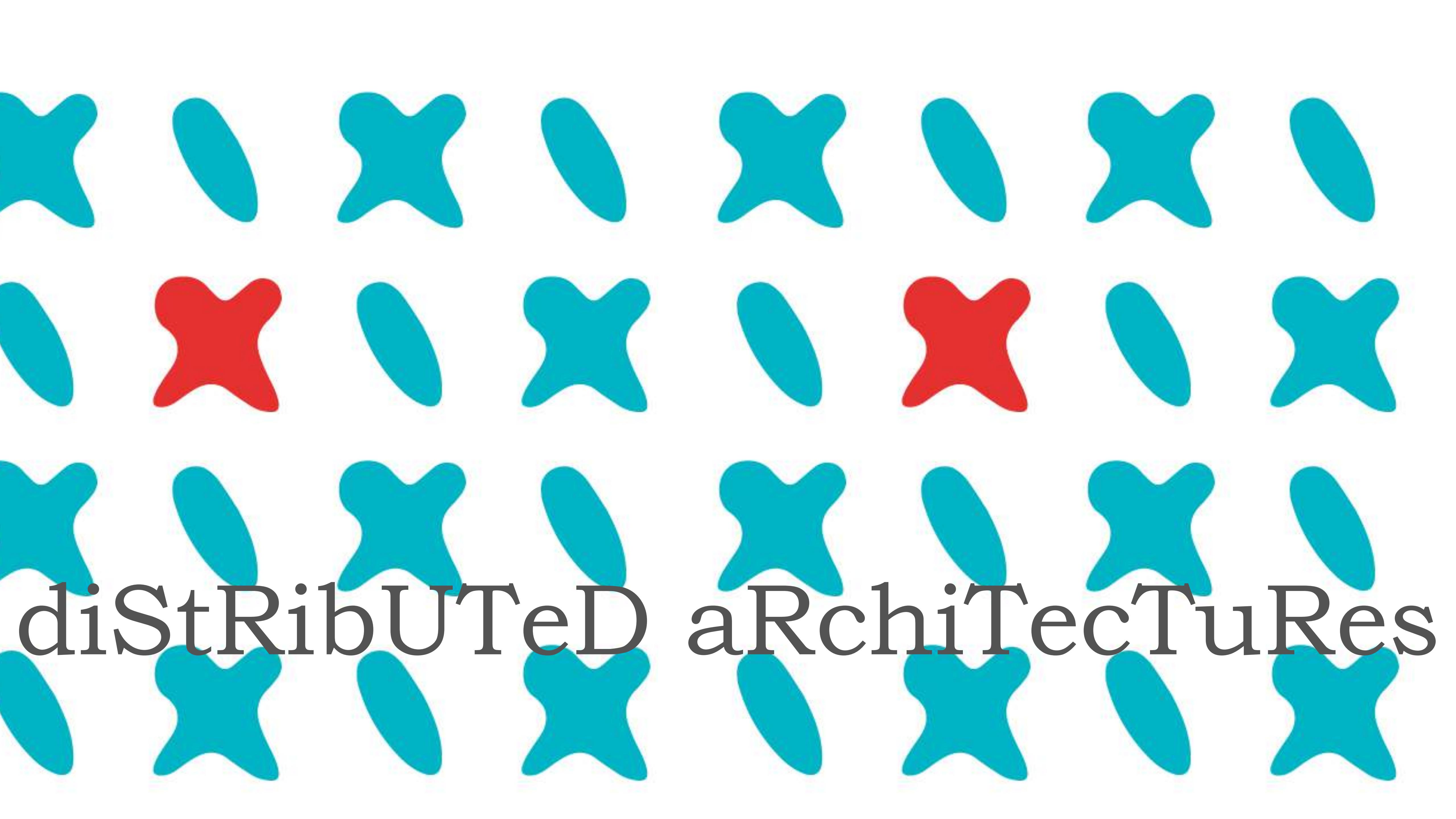
scaling monolithic applications requires you to scale the entire application, which can be both difficult and costly

monolithic application issues

application size

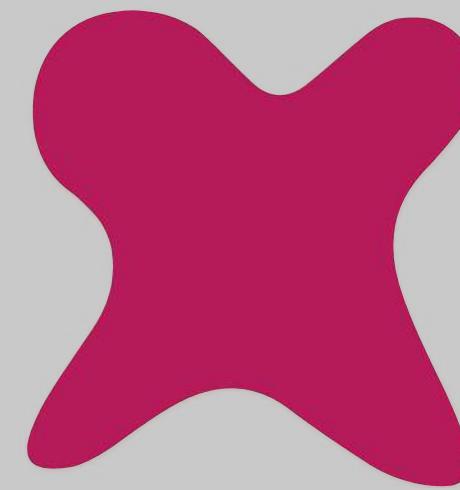


monolithic applications can grow to quickly and consume all available resources

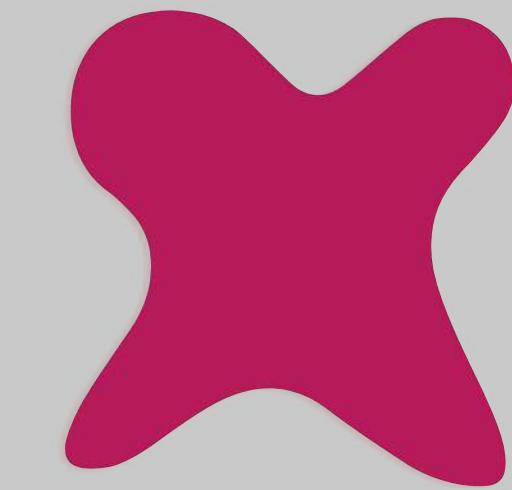


diStRibUTeD aRchiTectuReS

diStRibUTeD aRchiTecTuReS

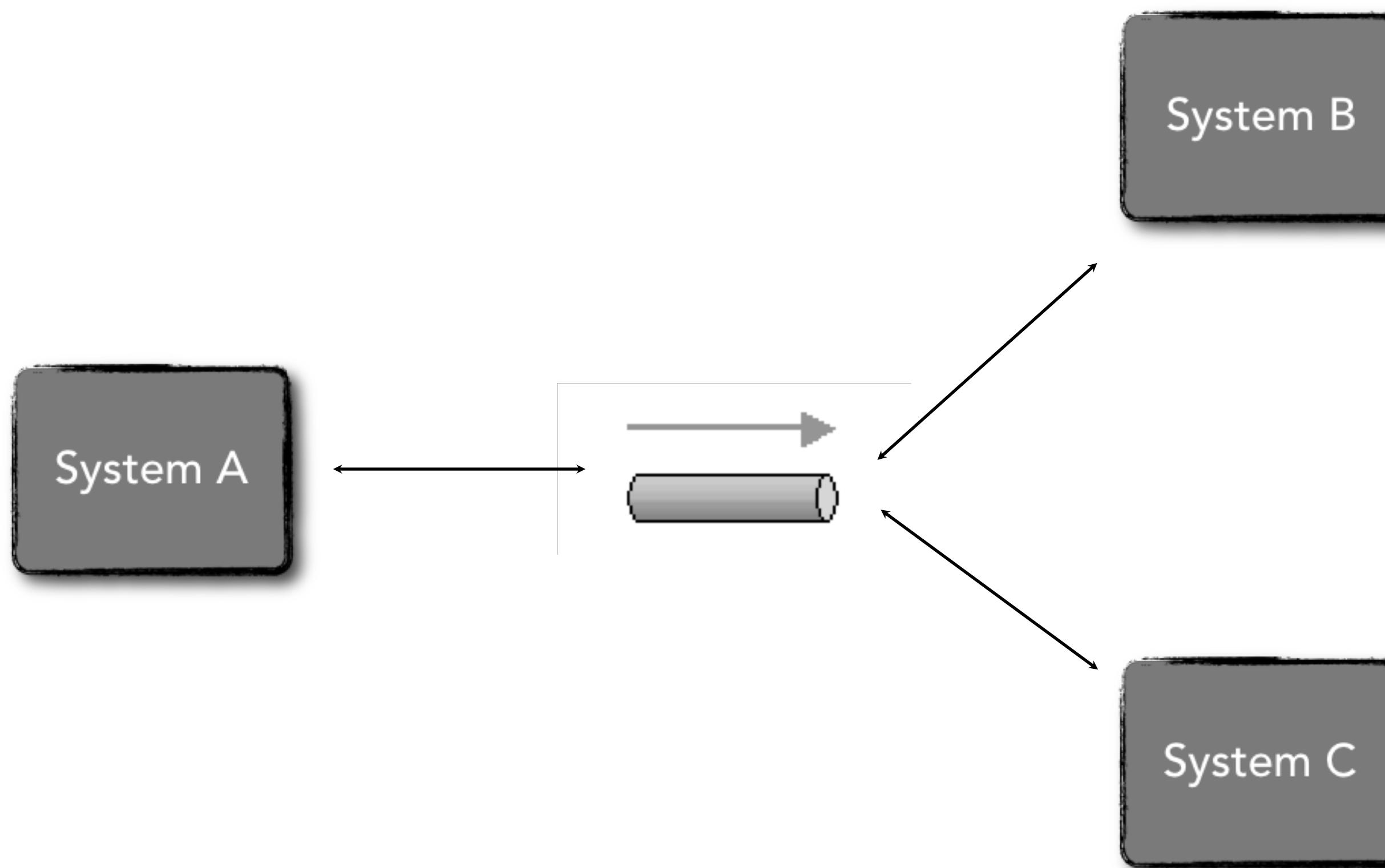


integration hubs

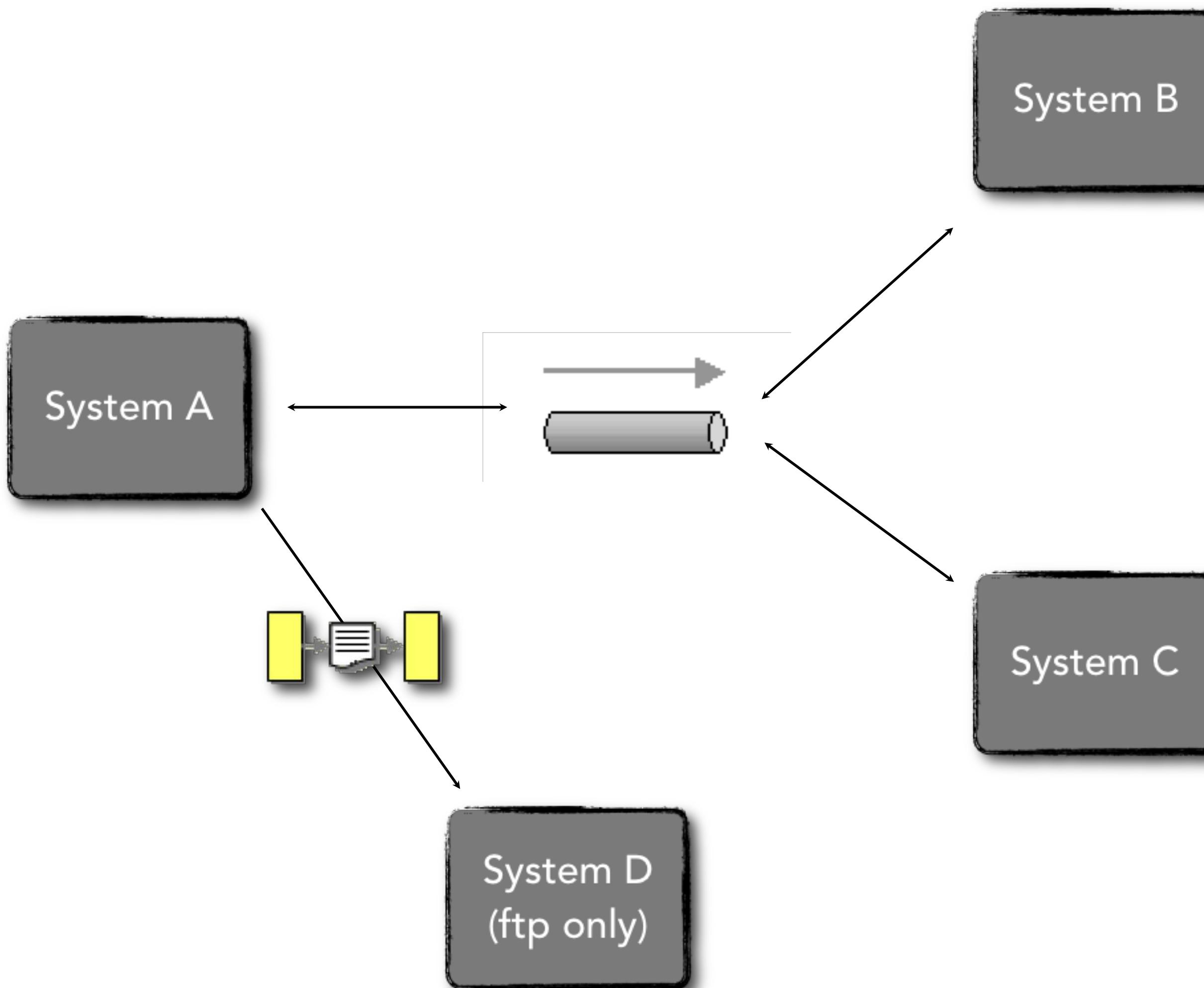


mediators

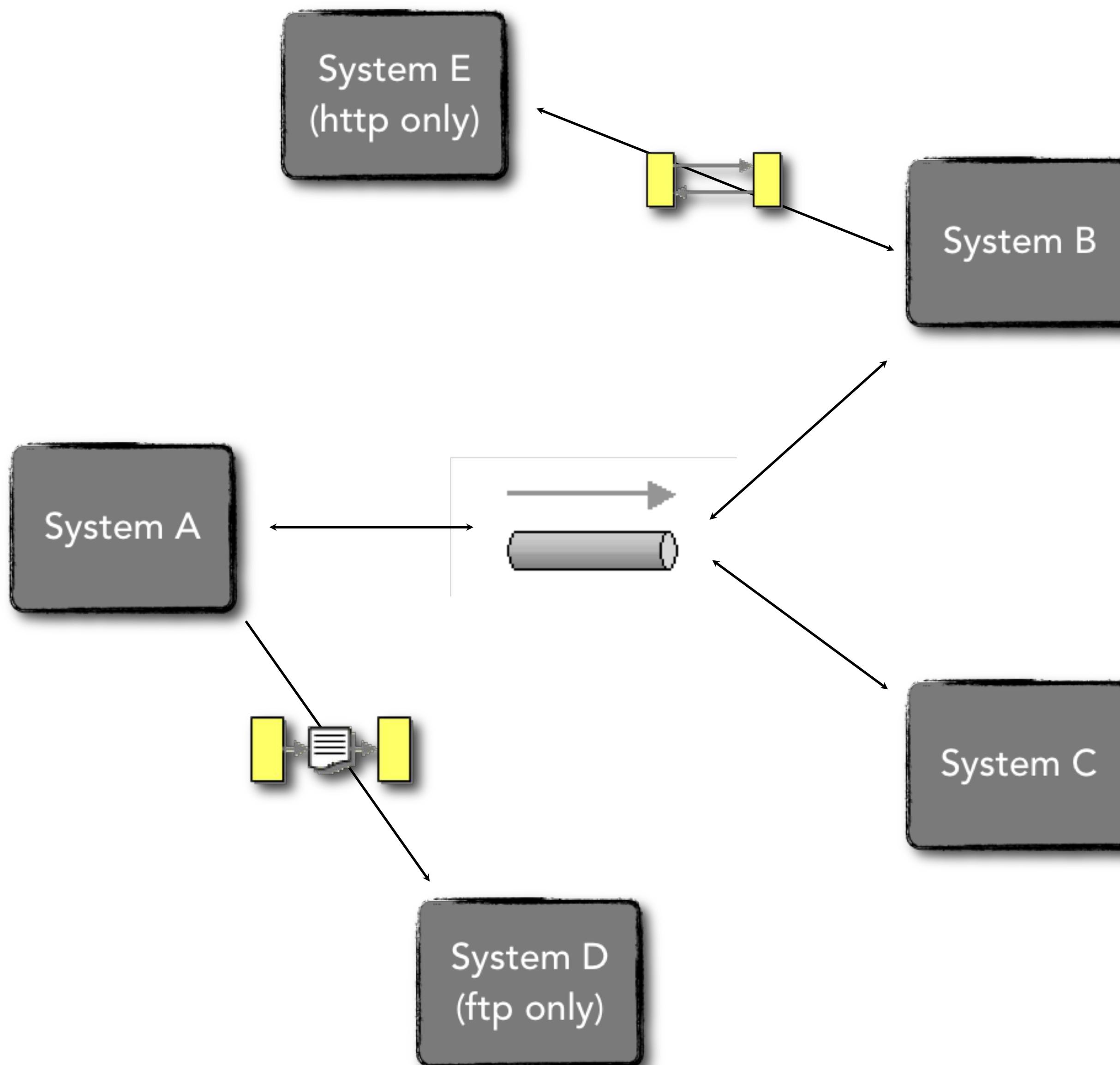
origins: hubs



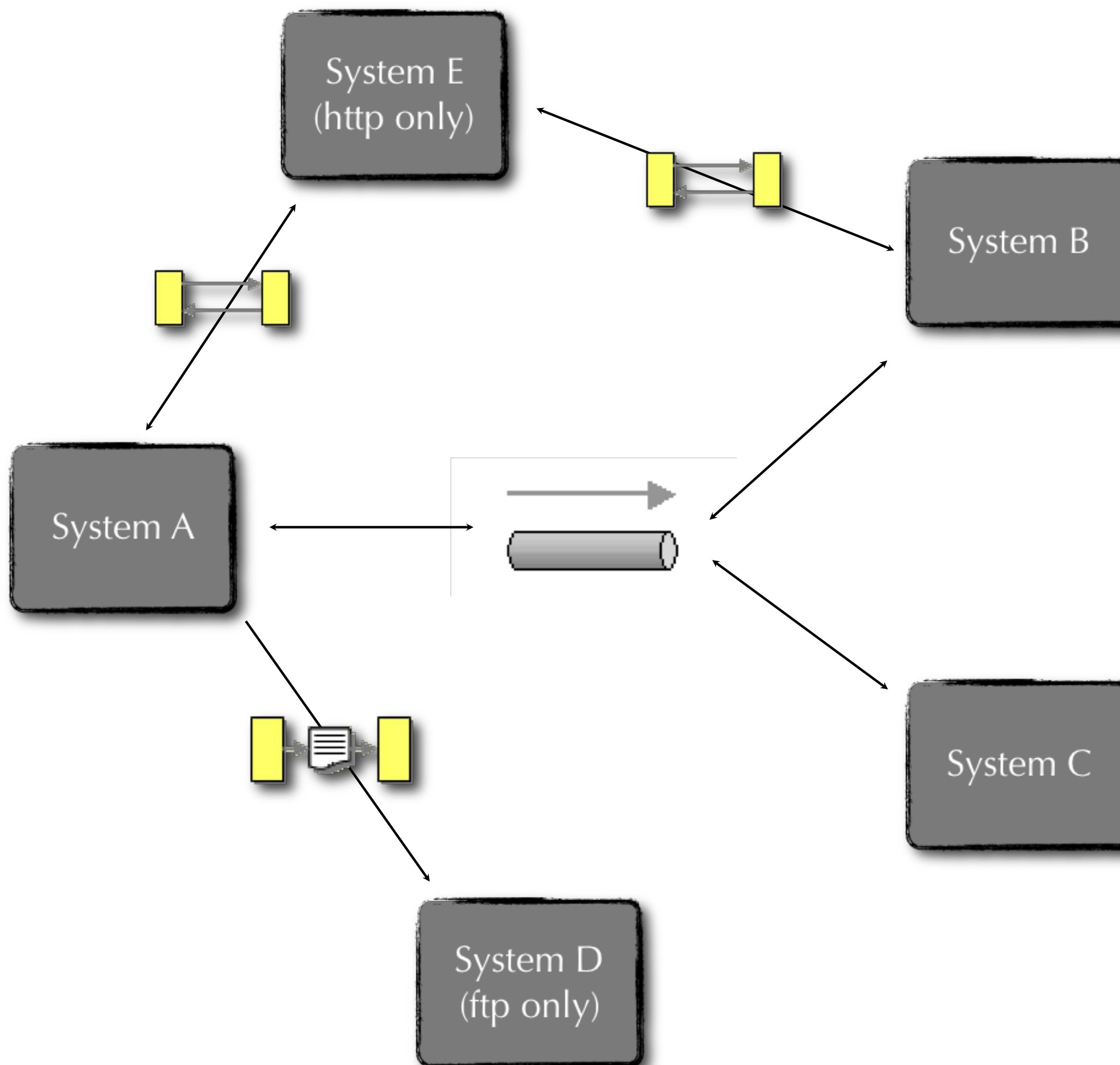
origins: hubs



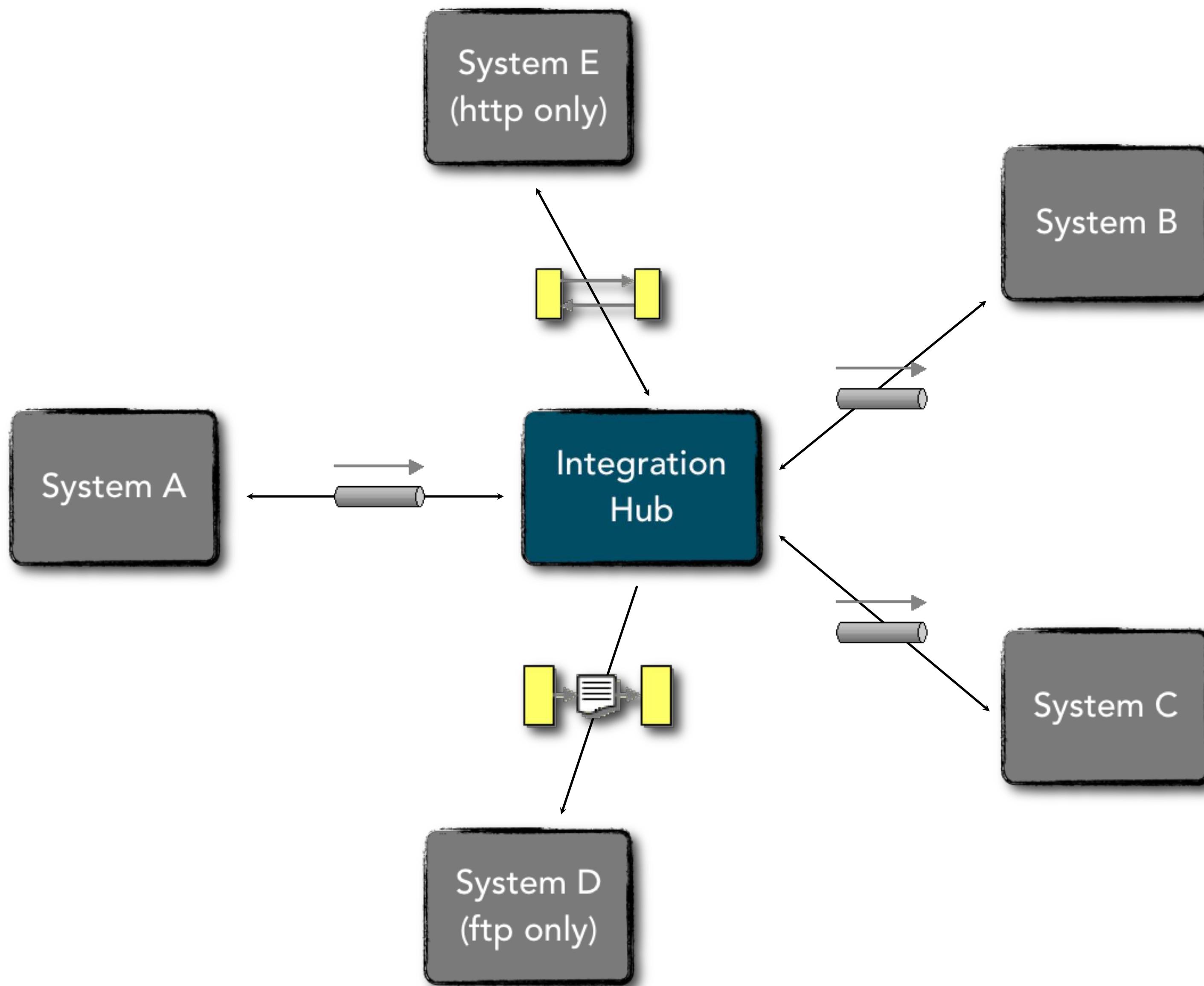
origins: hubs



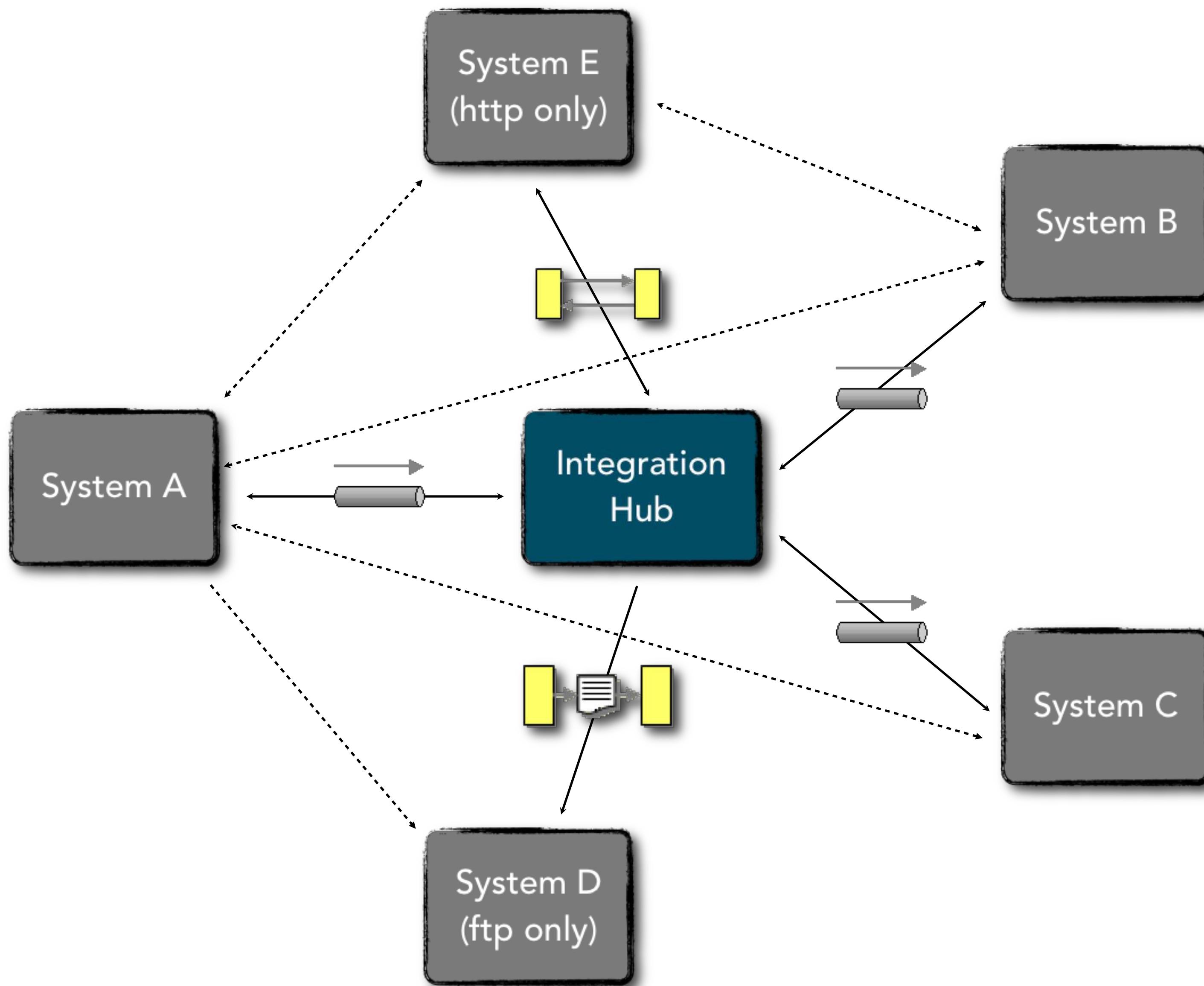
origins: hubs



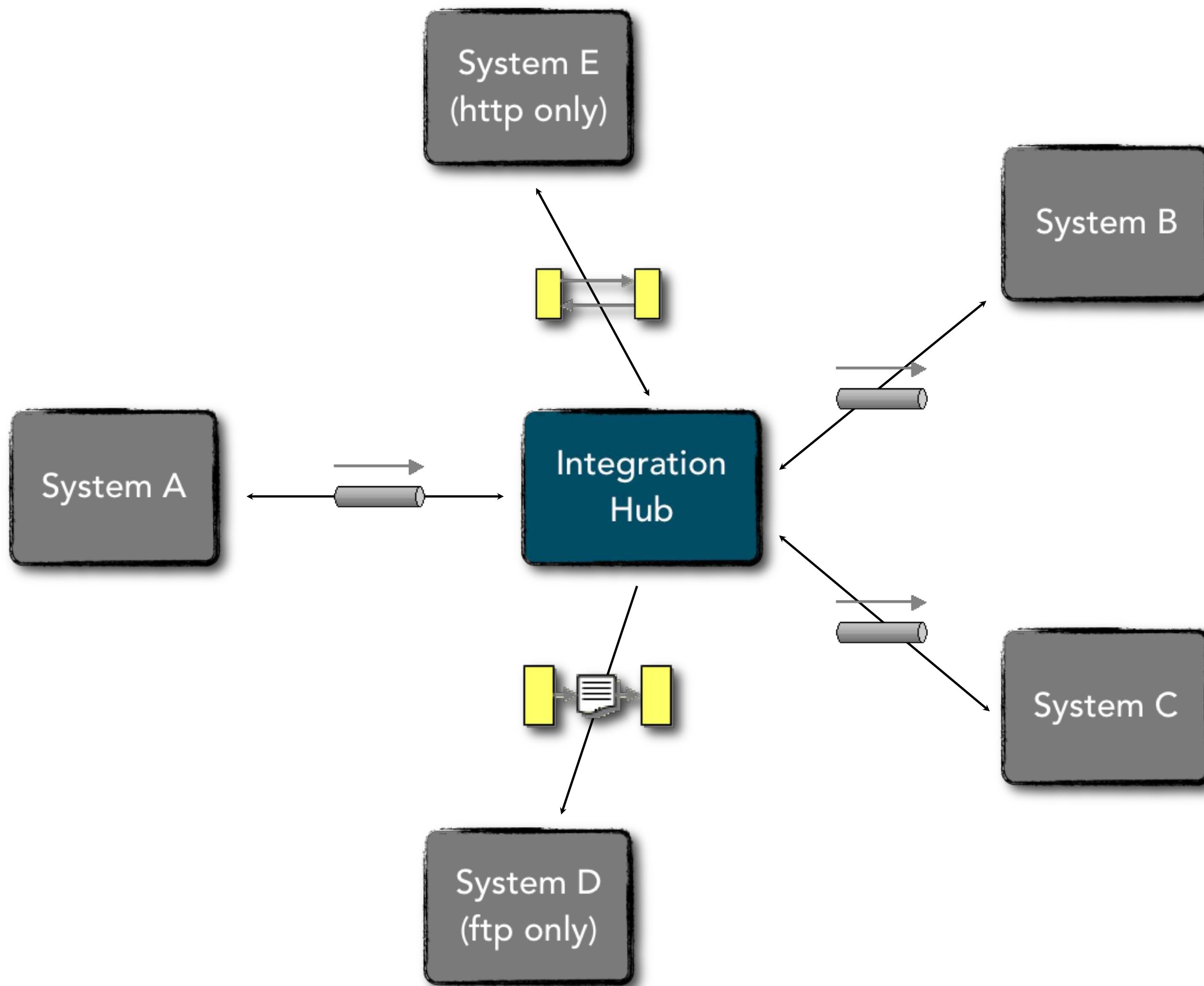
origins: hubs



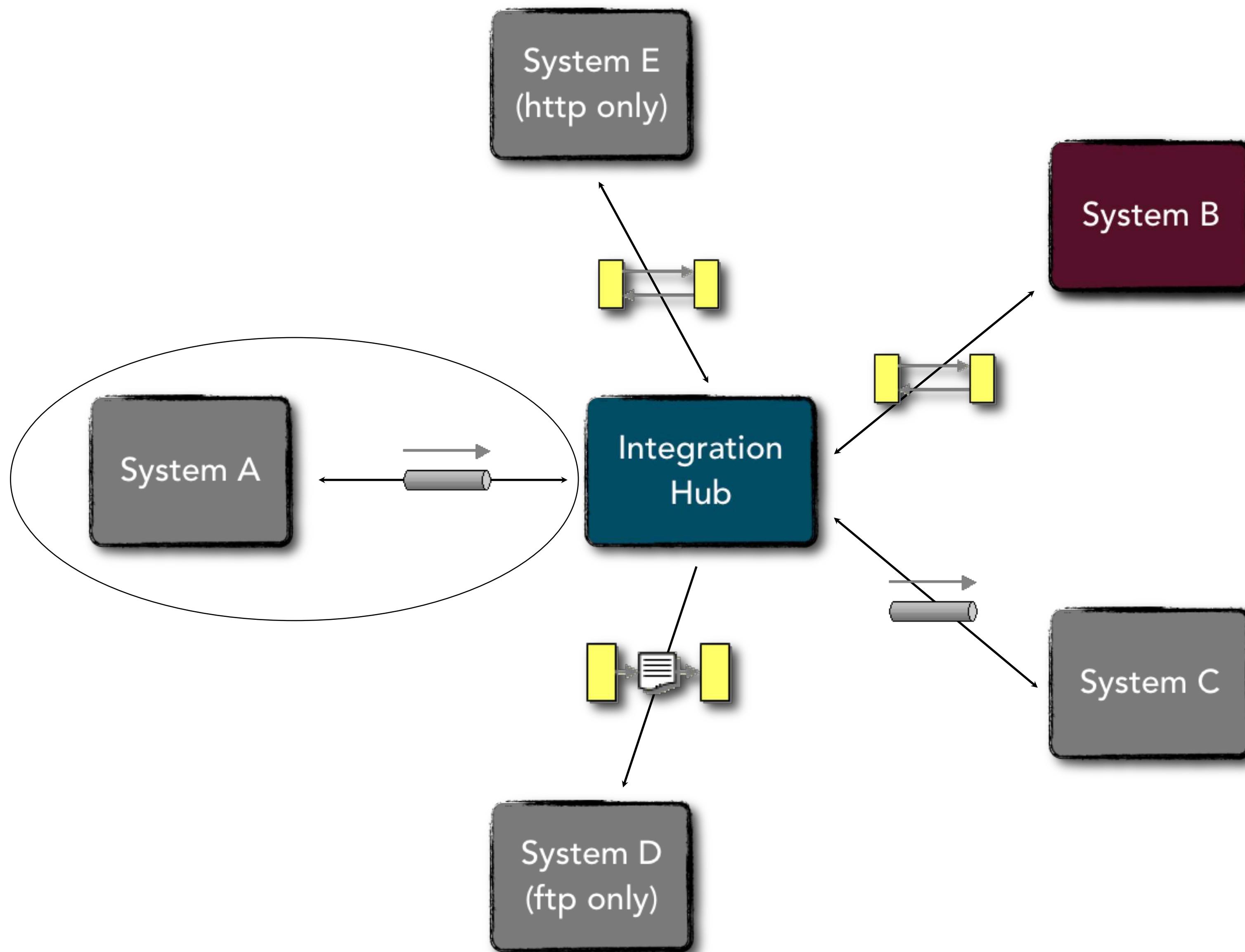
origins: hubs



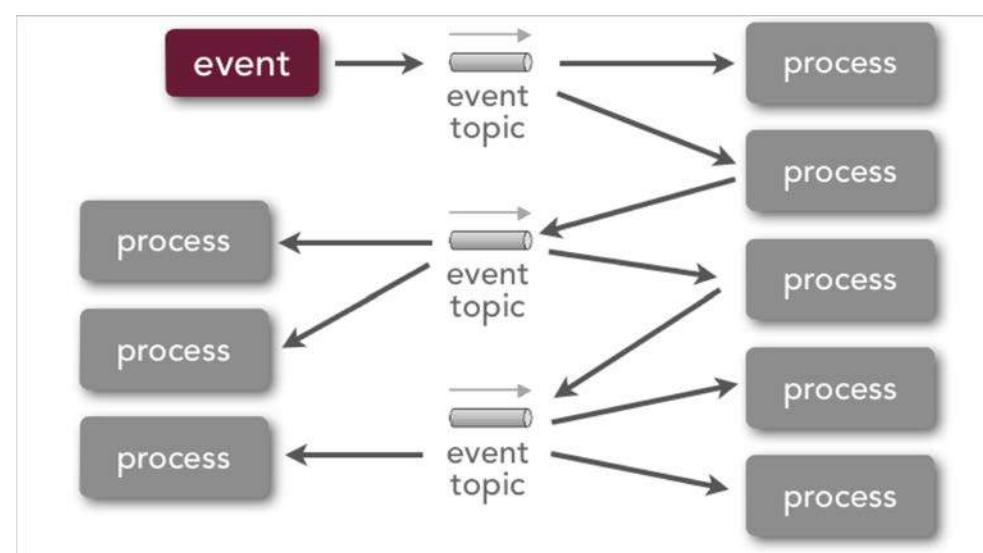
origins: hubs



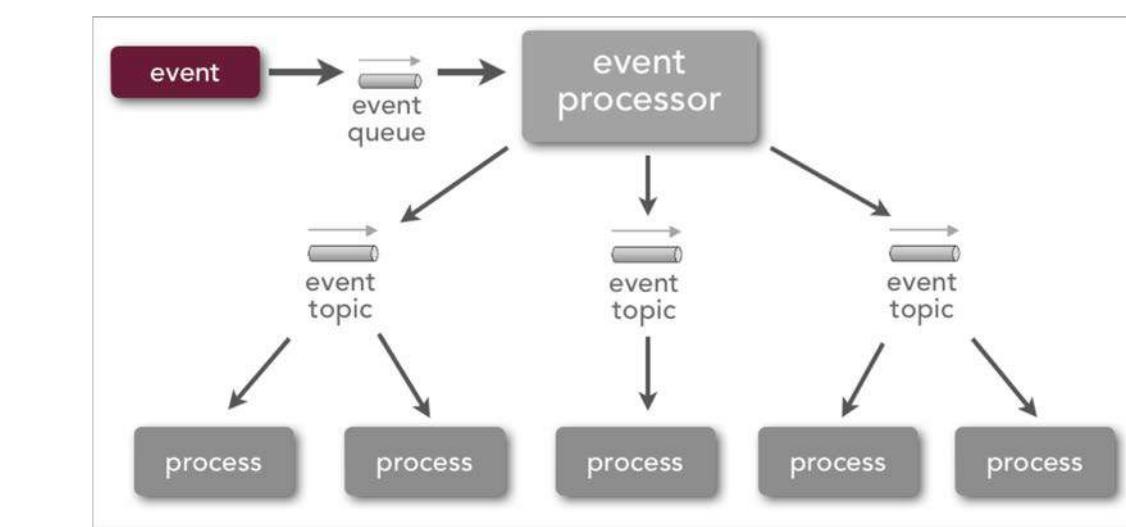
origins: hubs



event-driven architectures

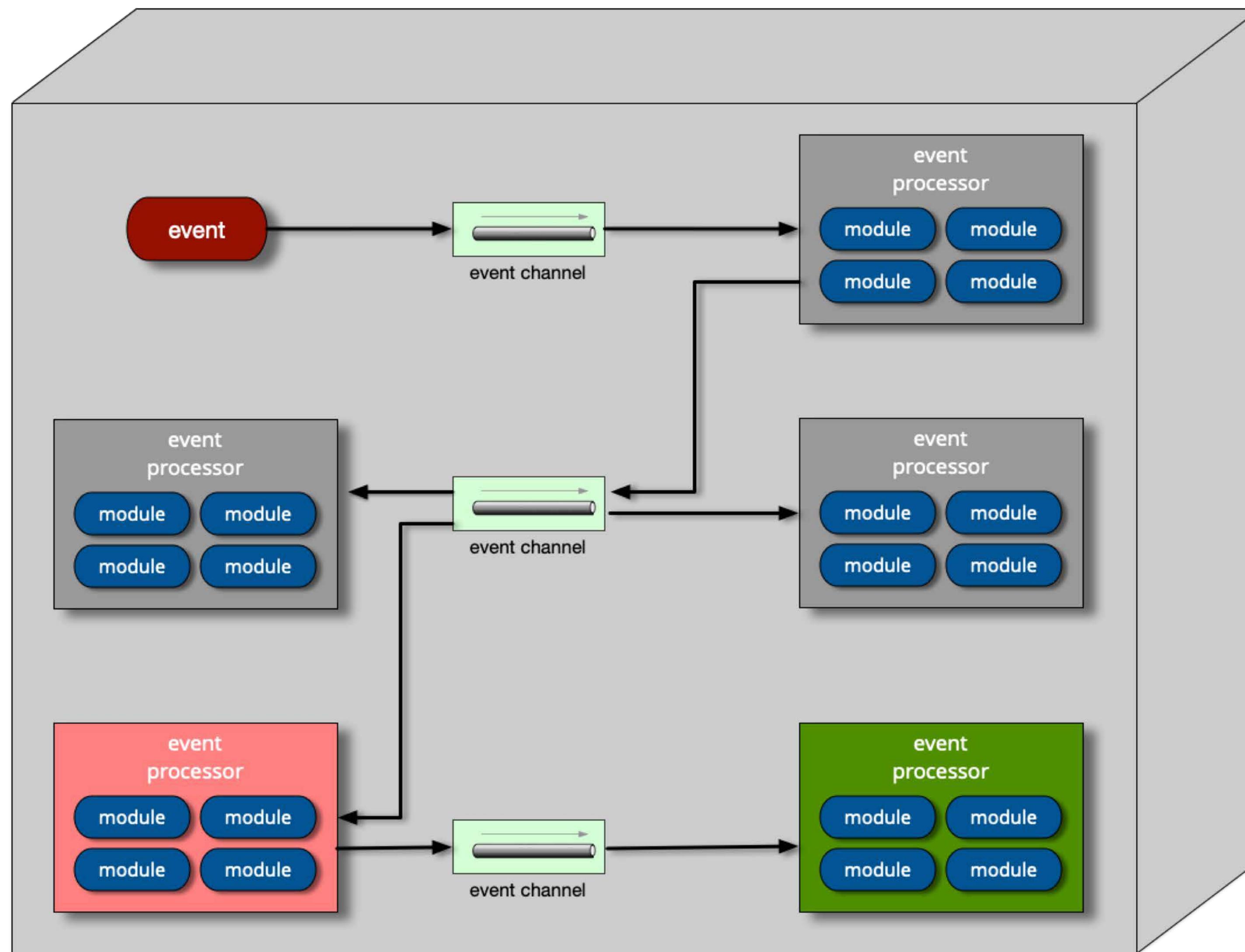


broker topology

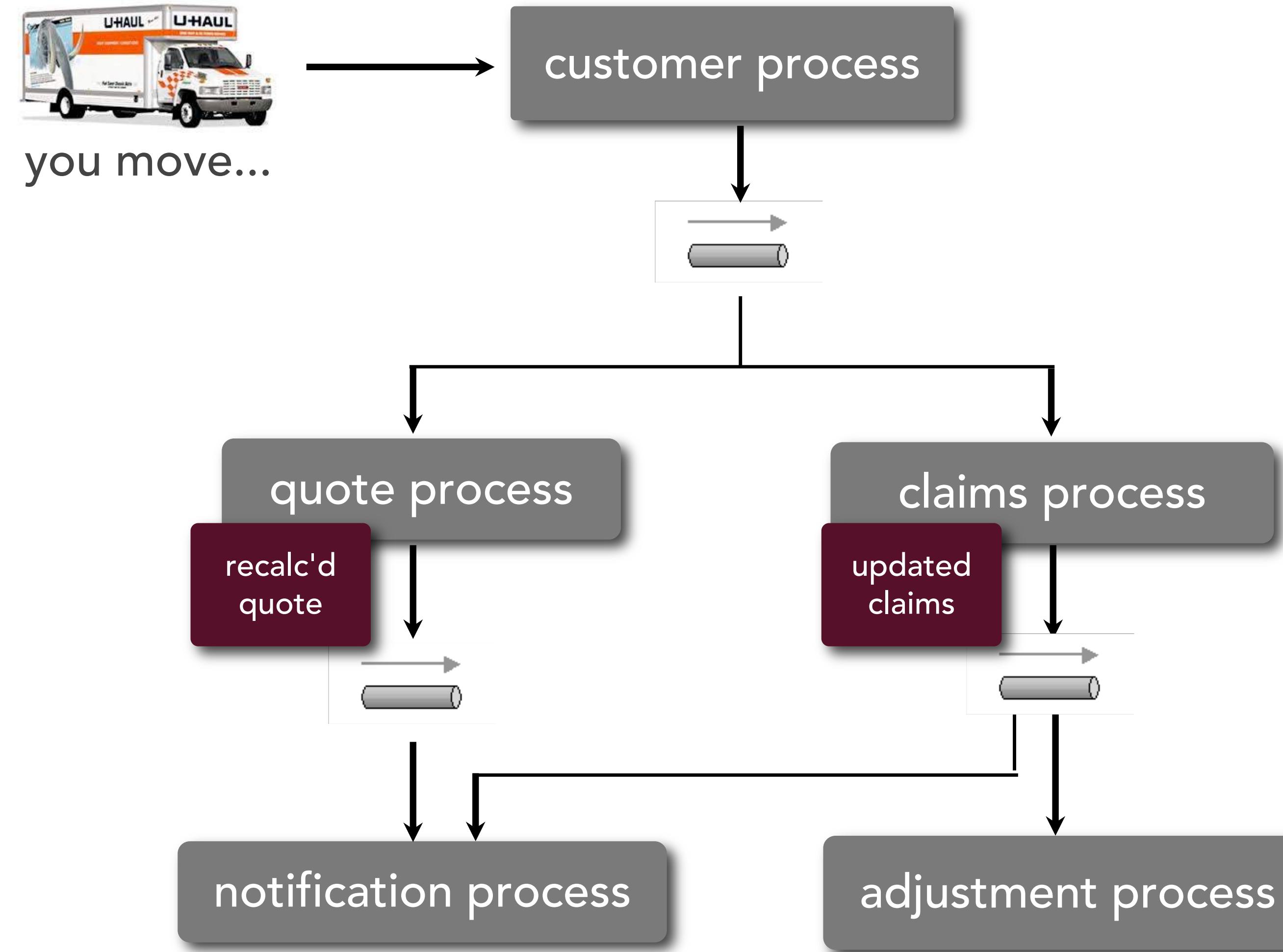


mediator topology

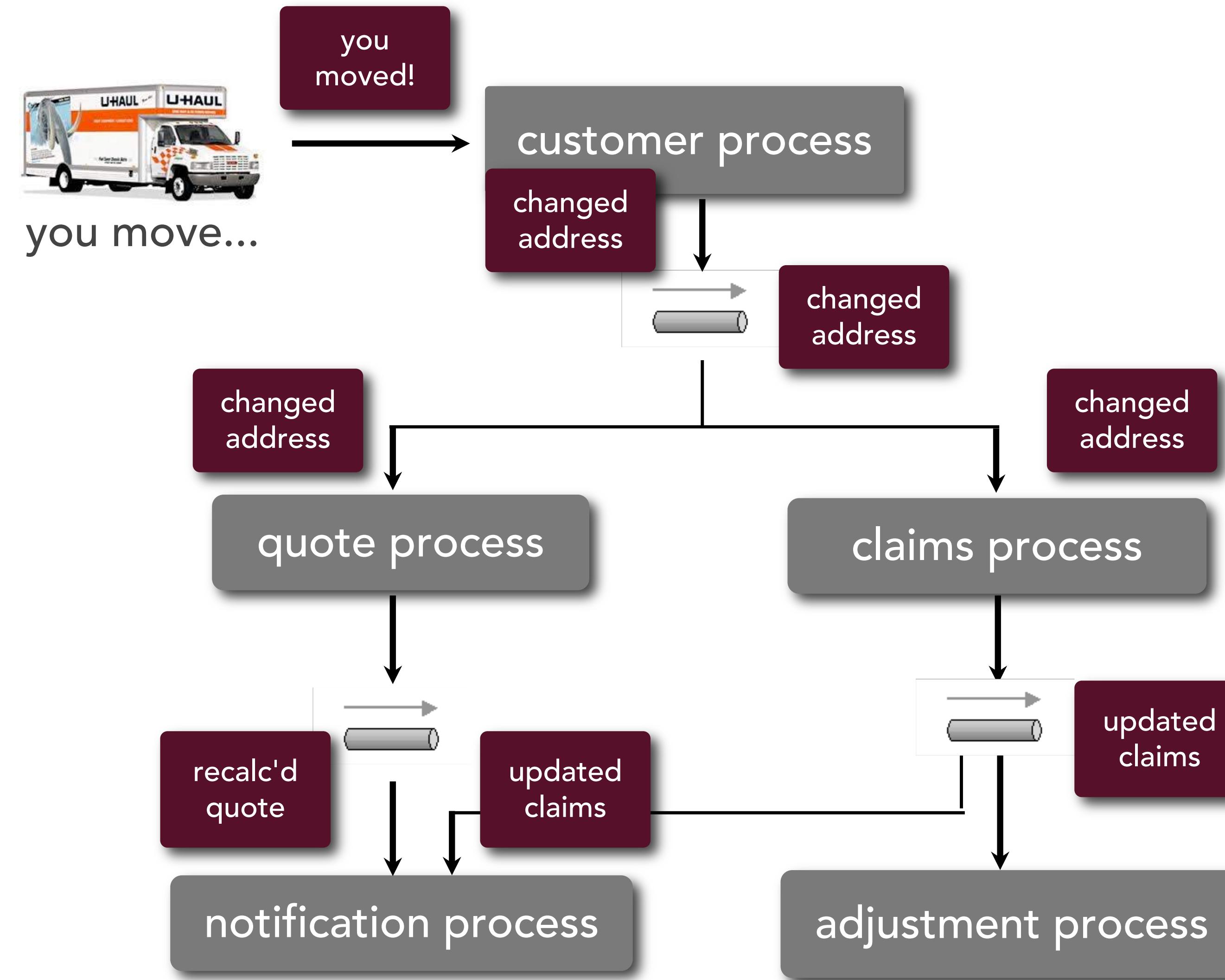
broker base architecture



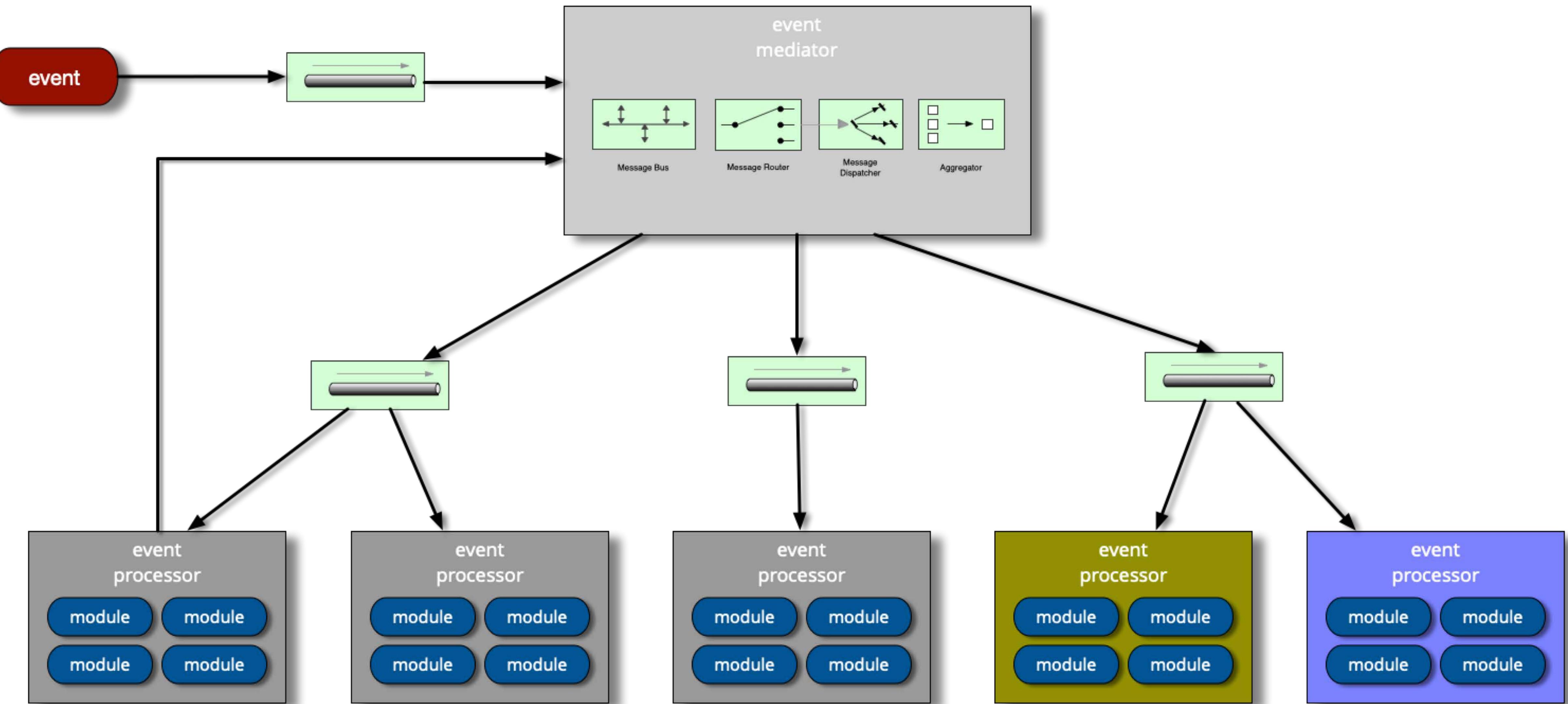
broker message passing



broker message passing



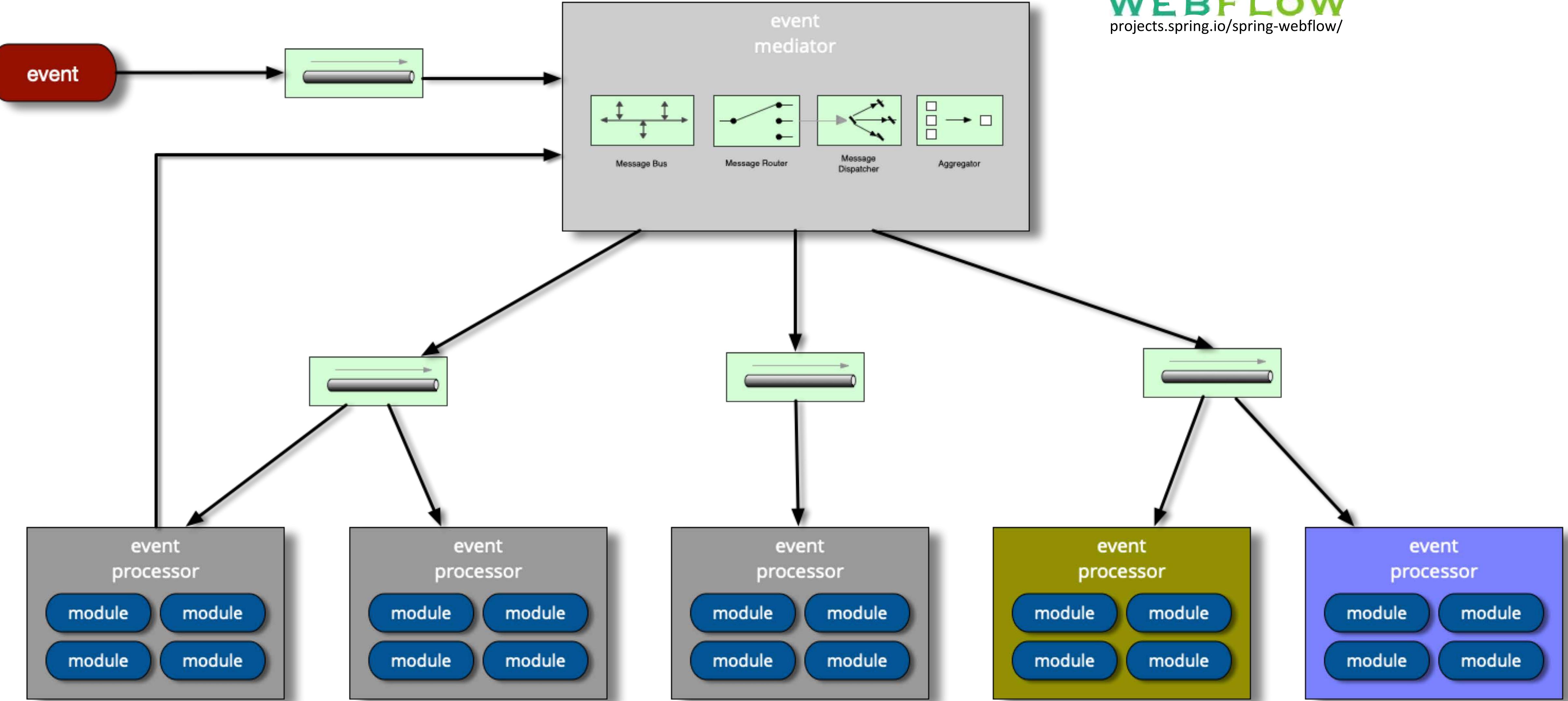
mediator EDA



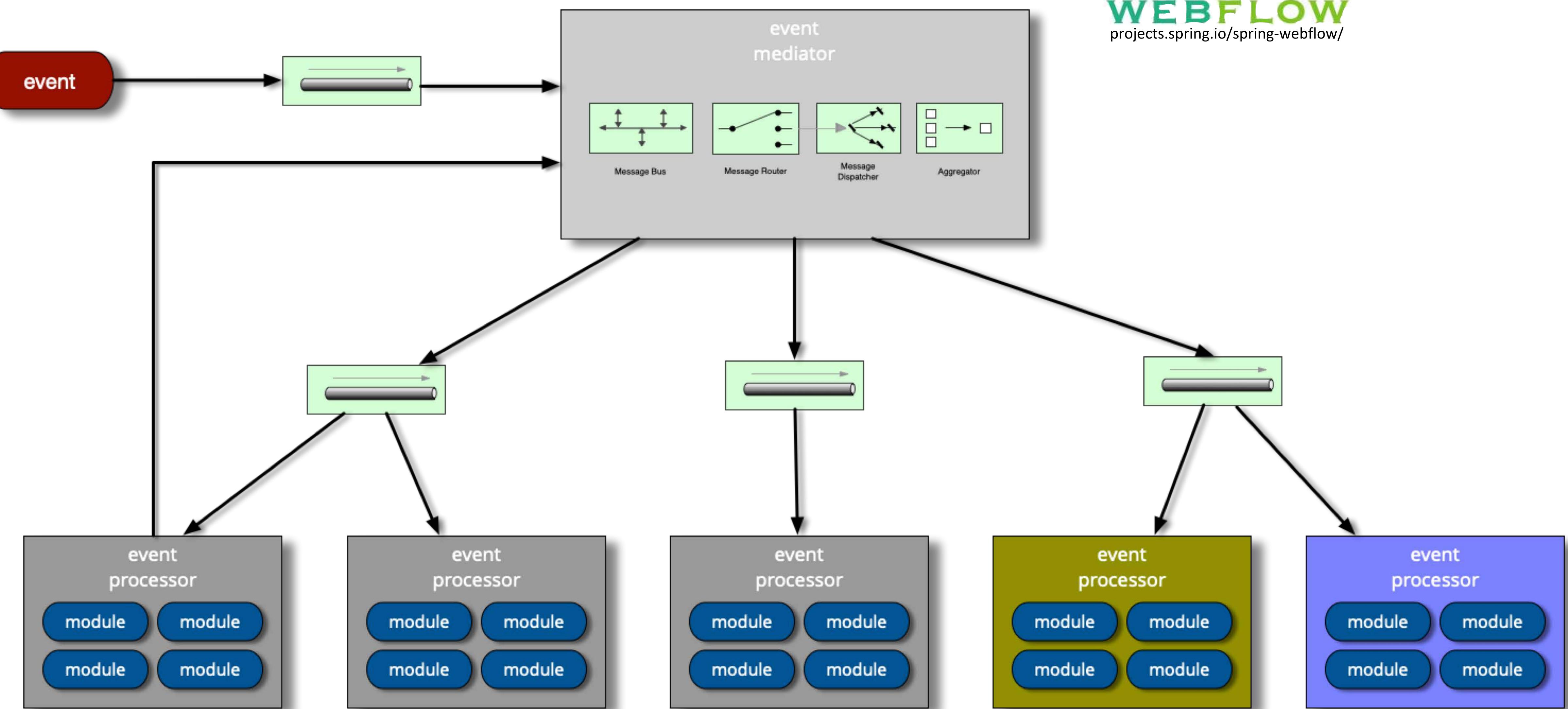
mediator EDA



WEBFLOW
projects.spring.io/spring-webflow/

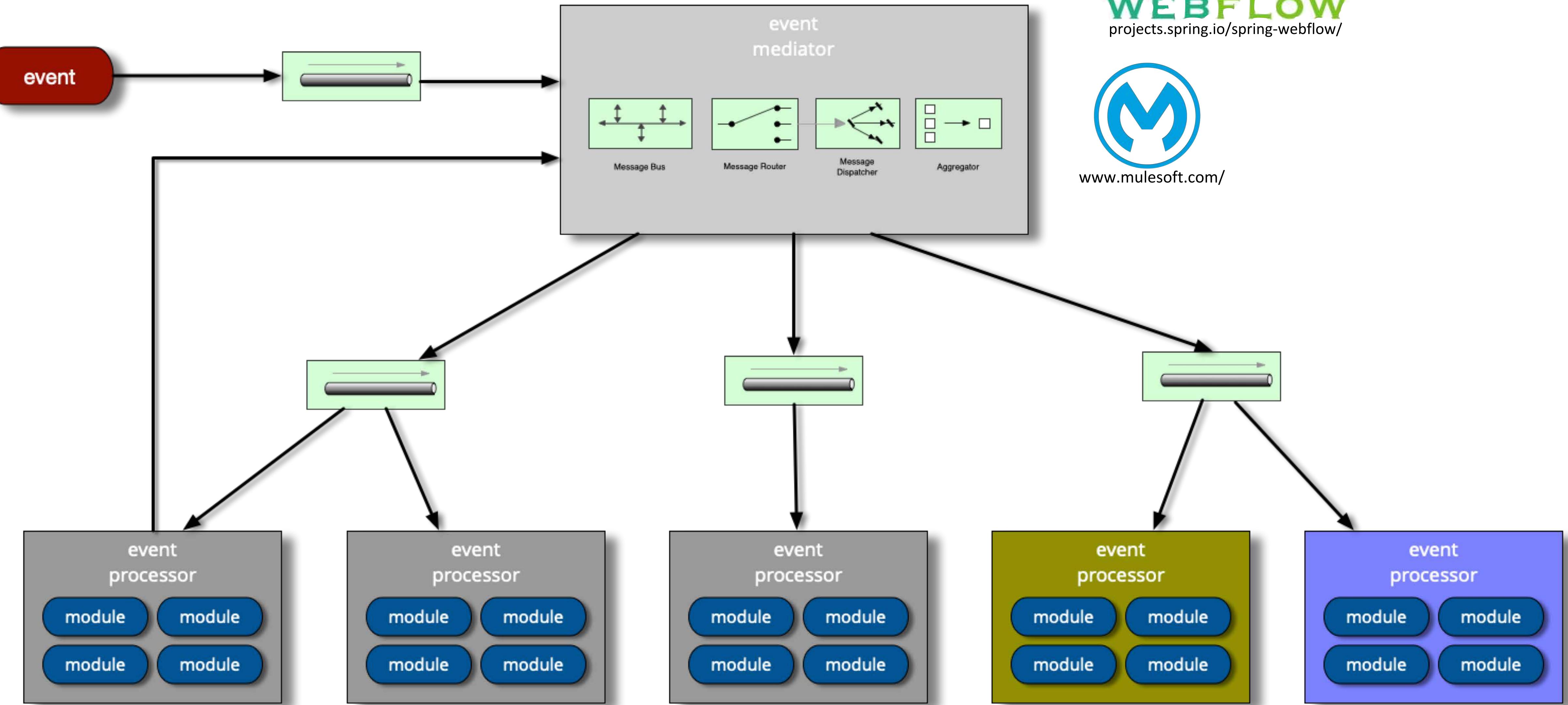


mediator EDA



WEBFLOW
projects.spring.io/spring-webflow/

mediator EDA

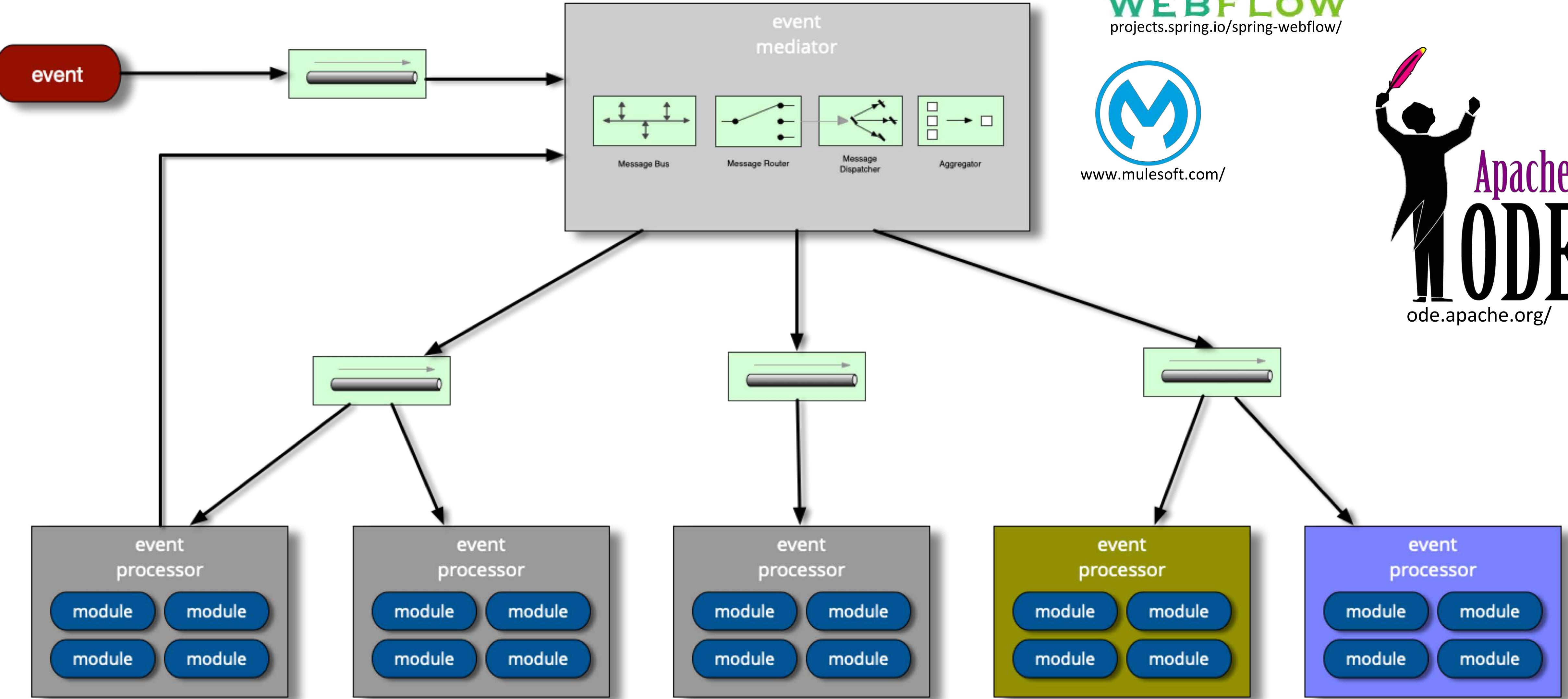


WEBFLOW
projects.spring.io/spring-webflow/

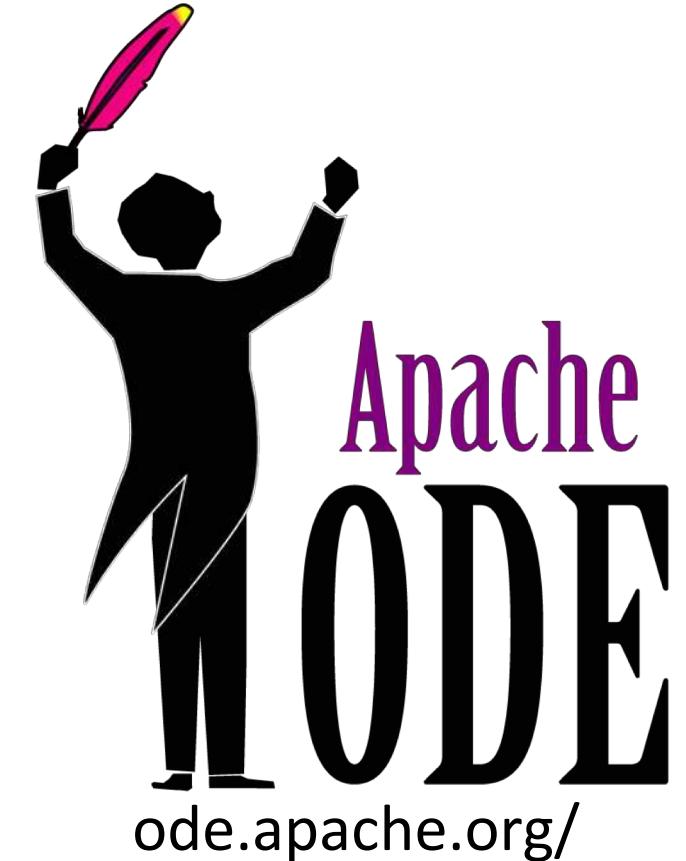


www.mulesoft.com/

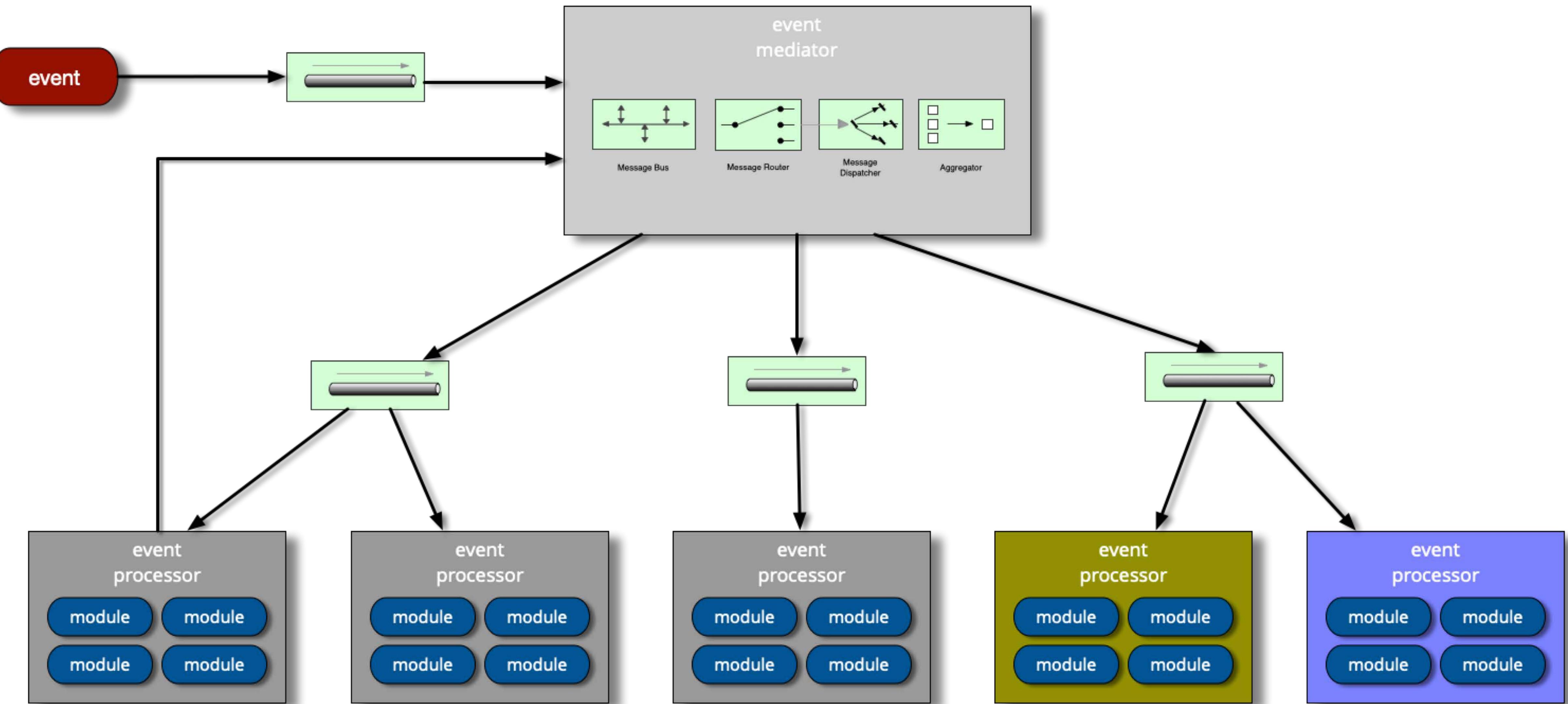
mediator EDA



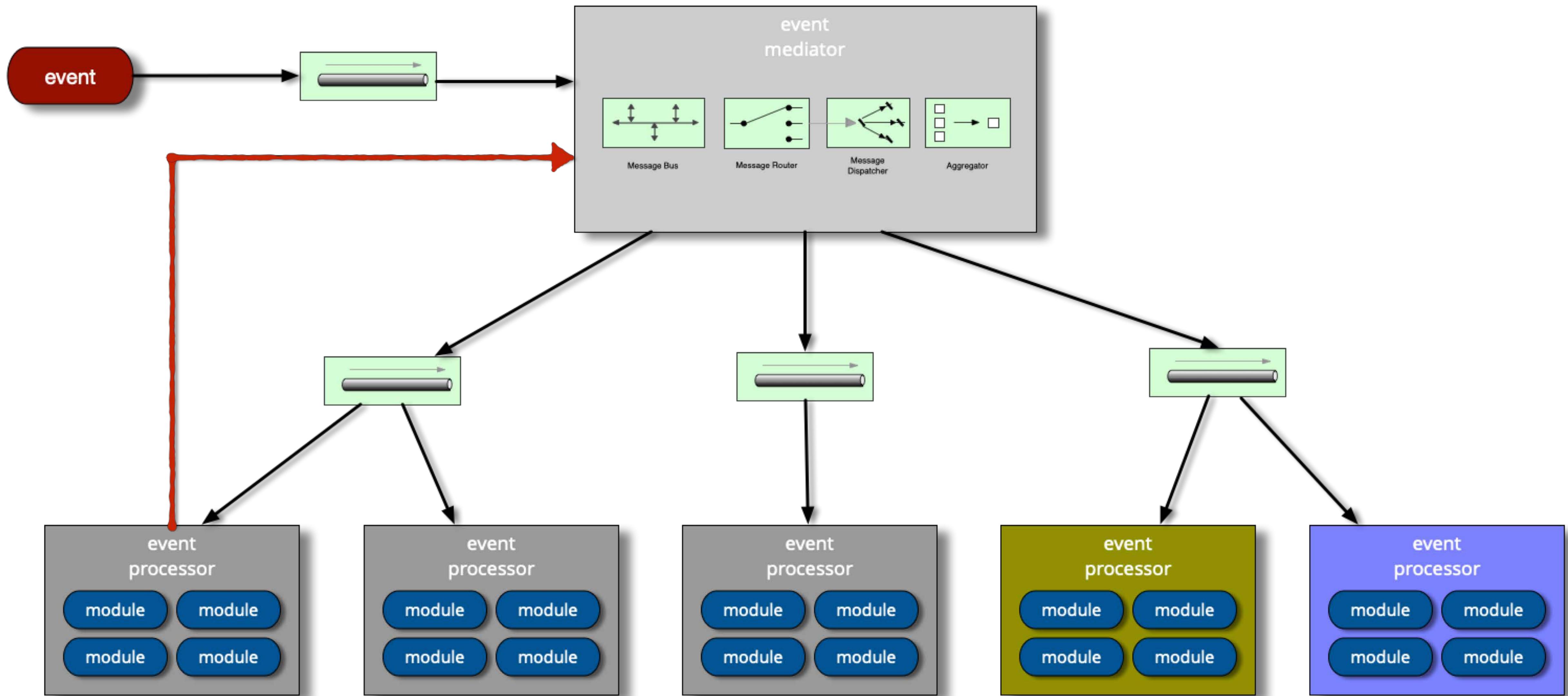
WEBFLOW
projects.spring.io/spring-webflow/



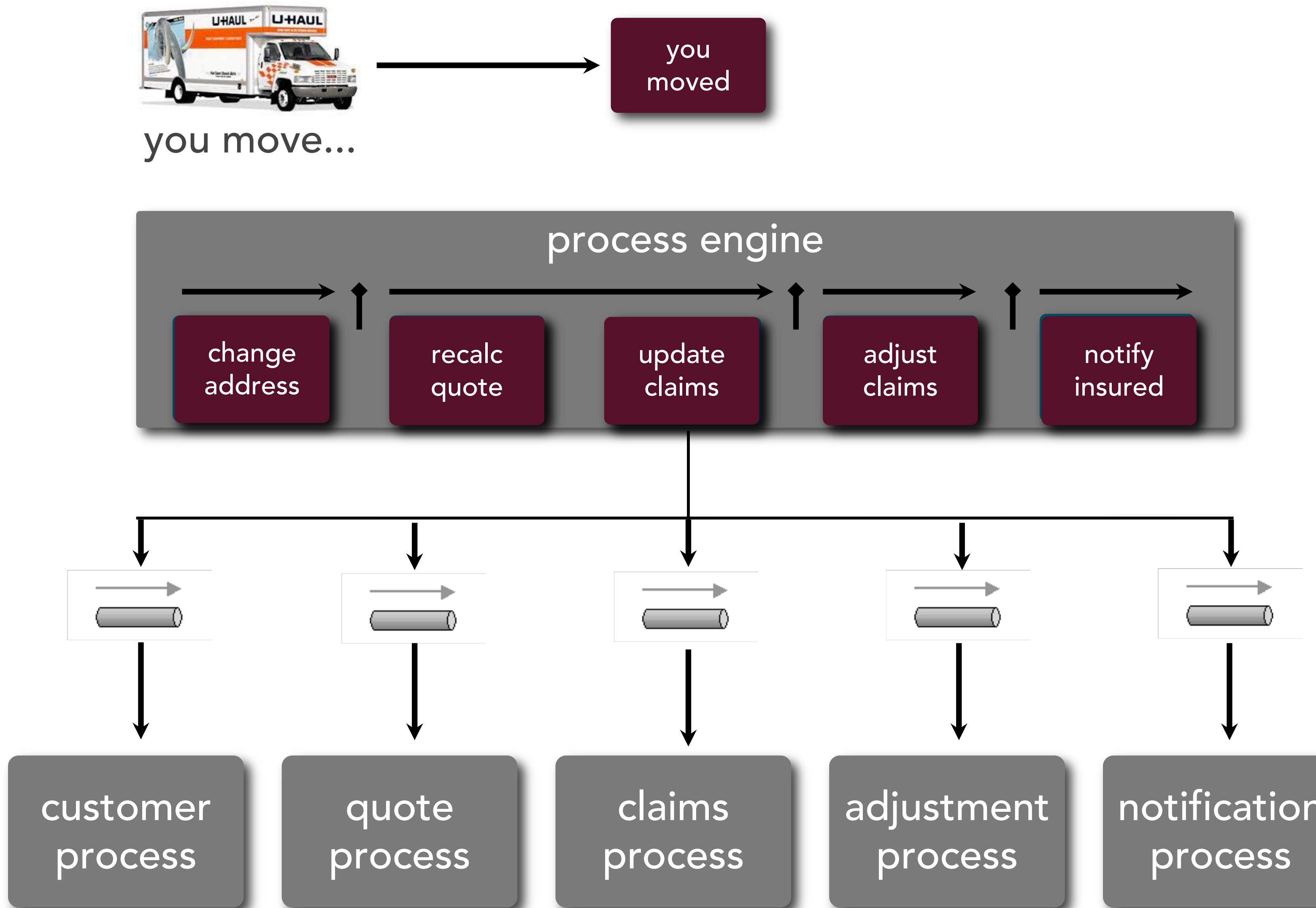
mediator EDA



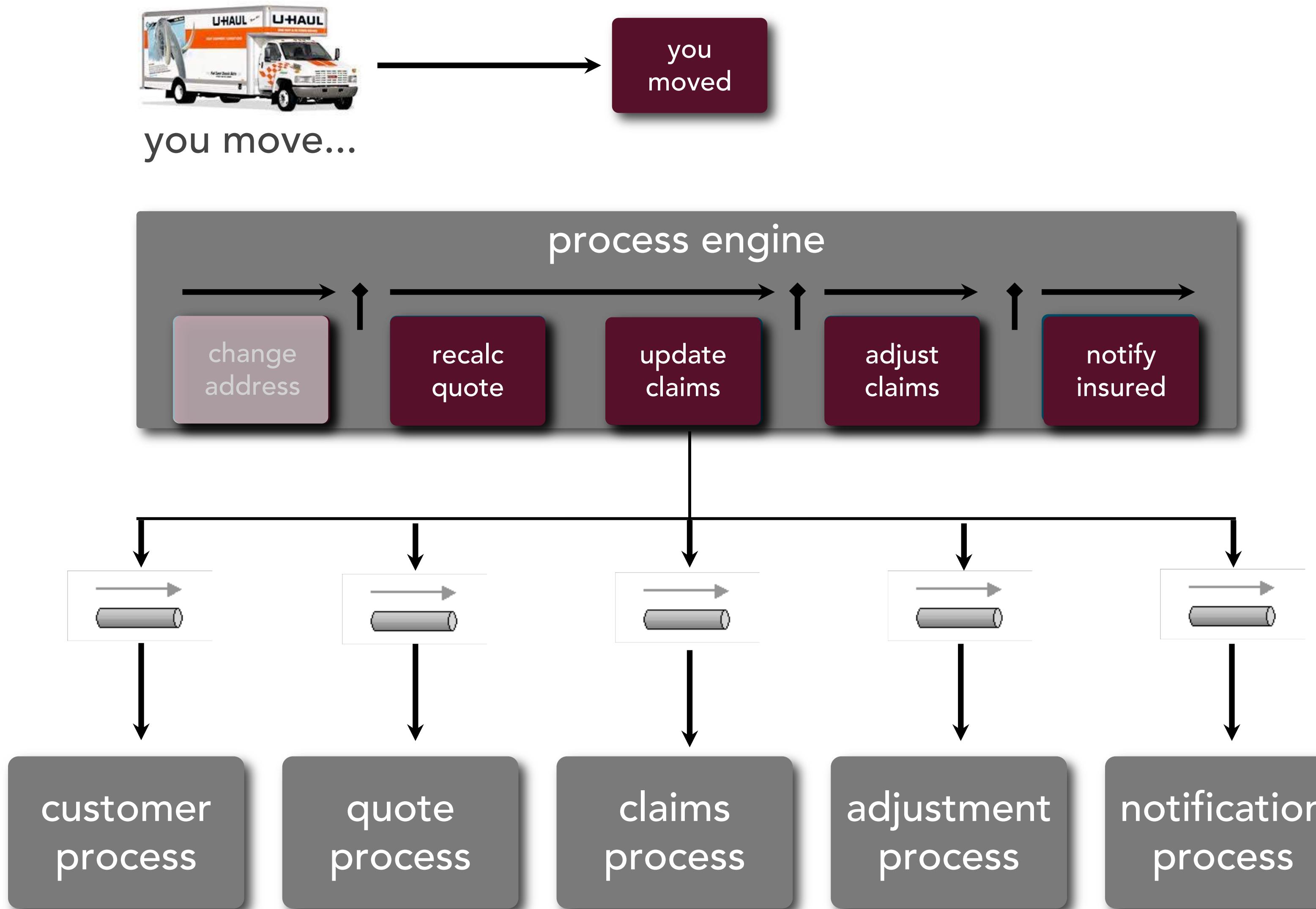
mediator EDA



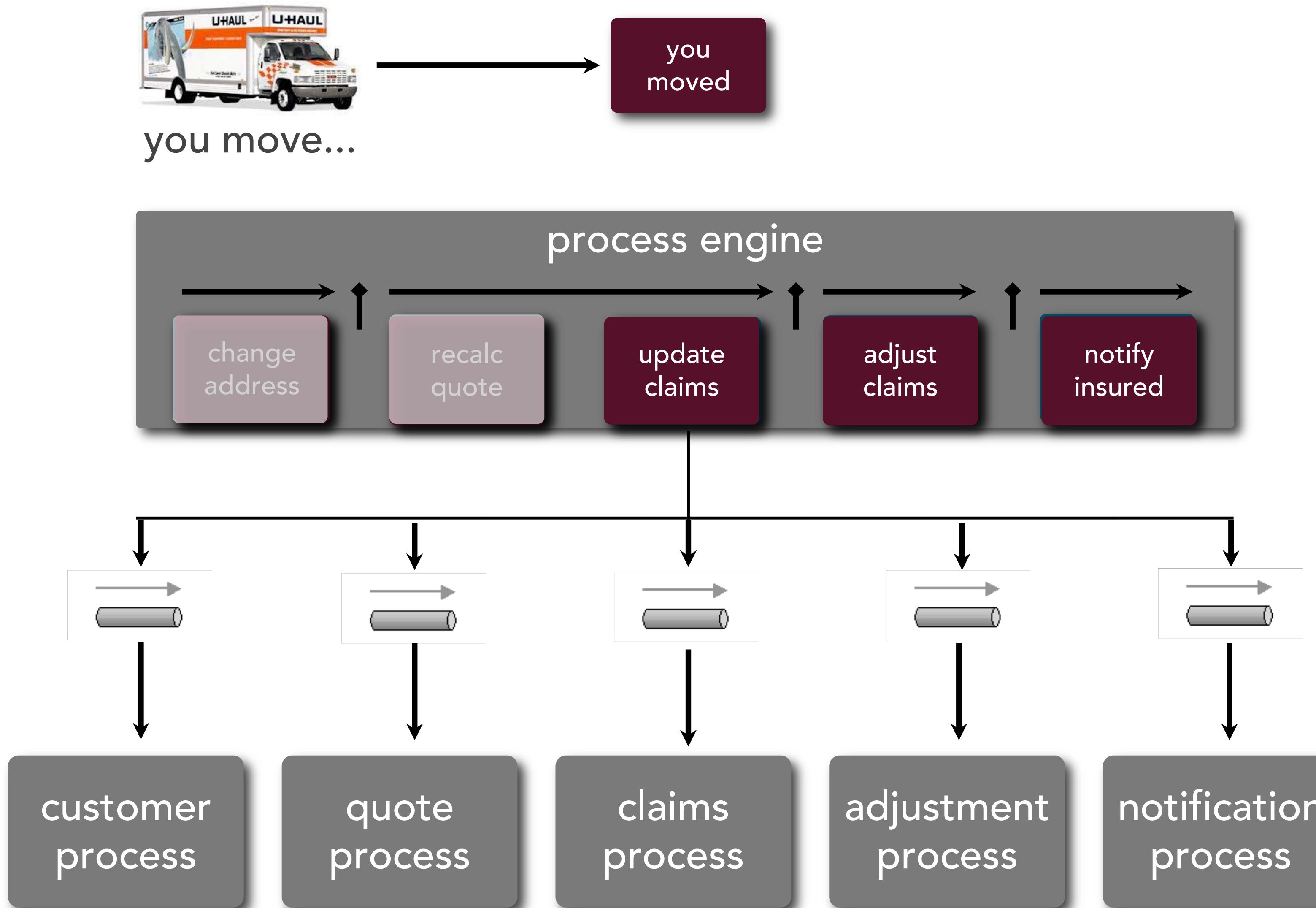
mediator message flow



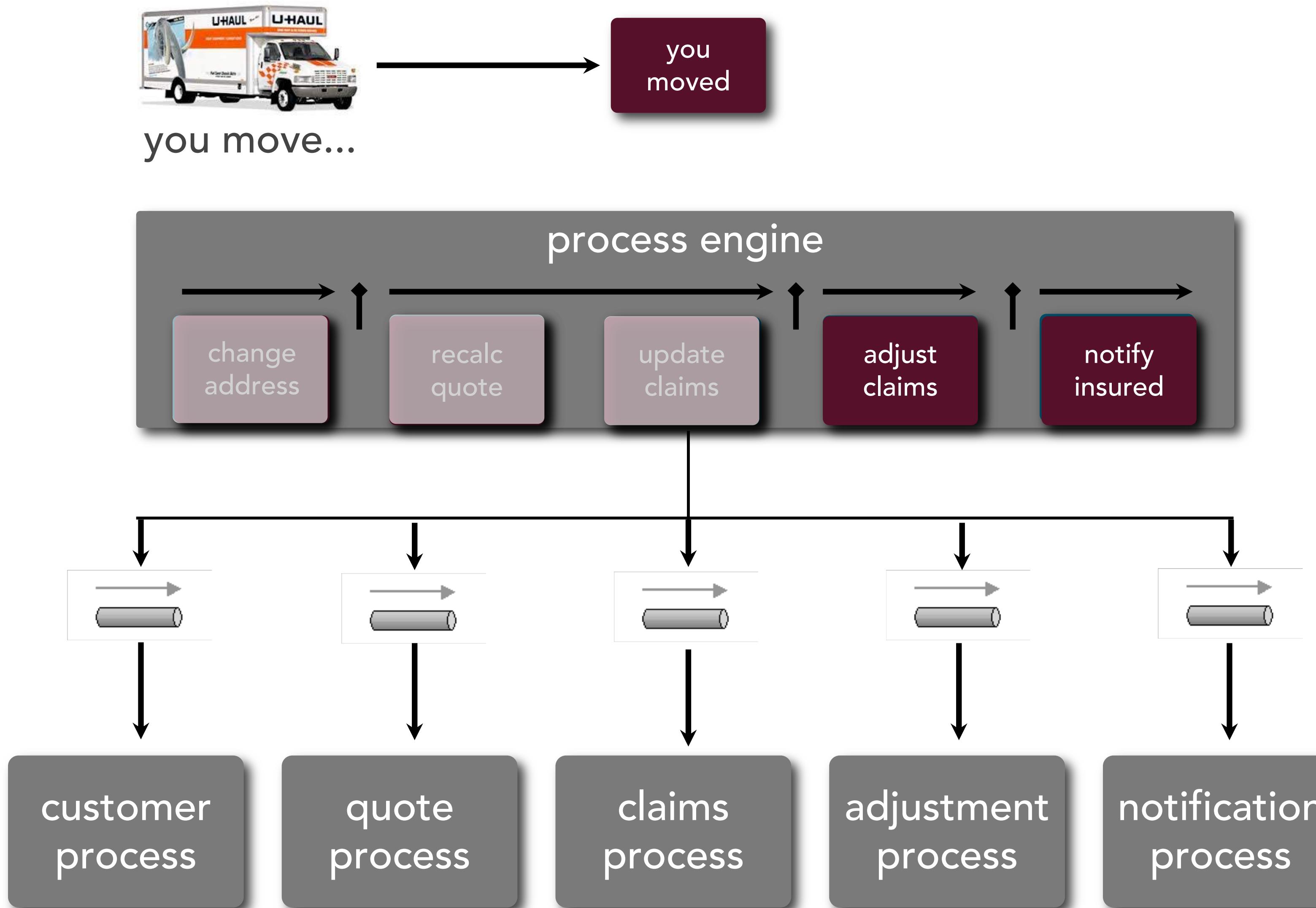
mediator message flow



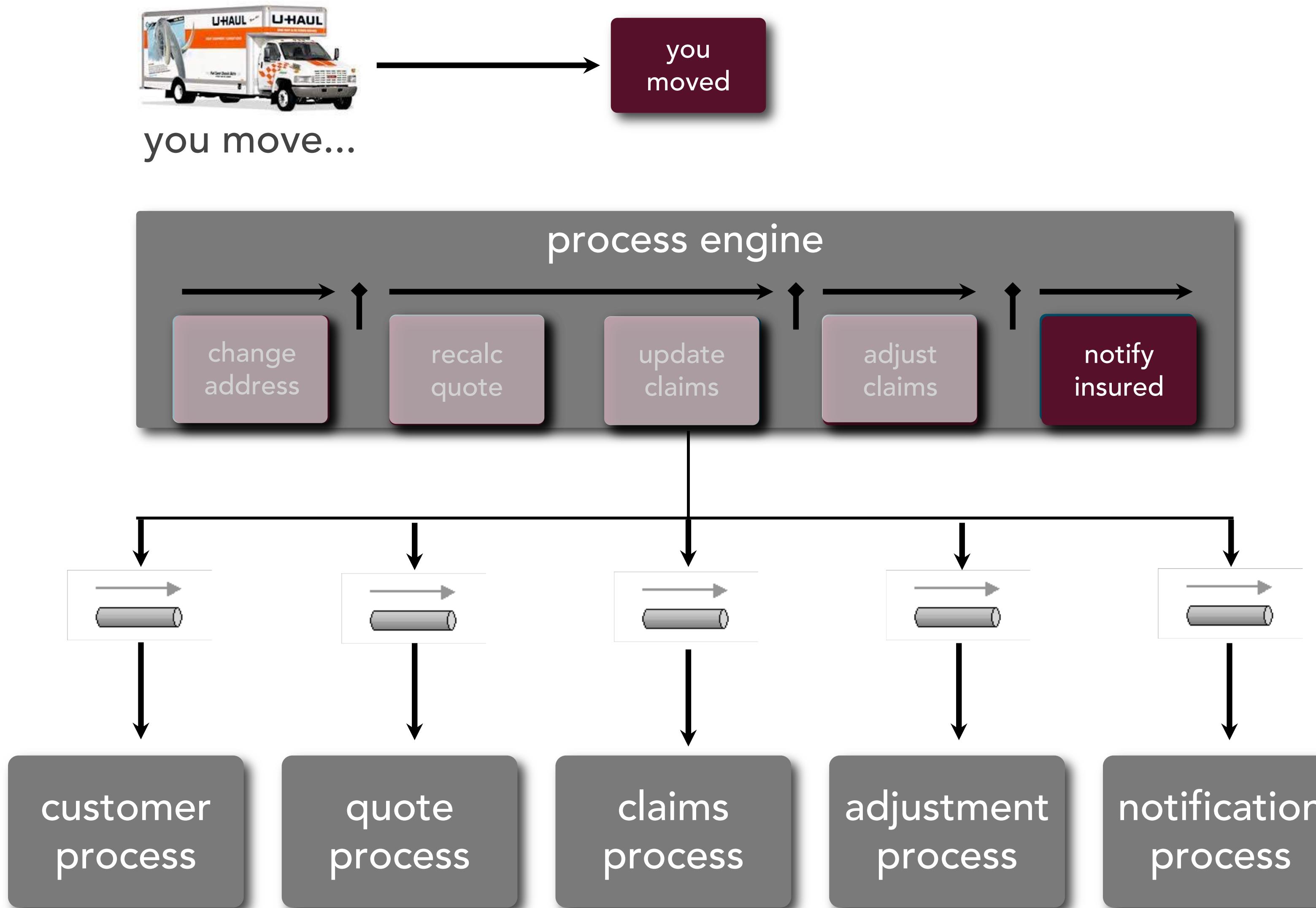
mediator message flow



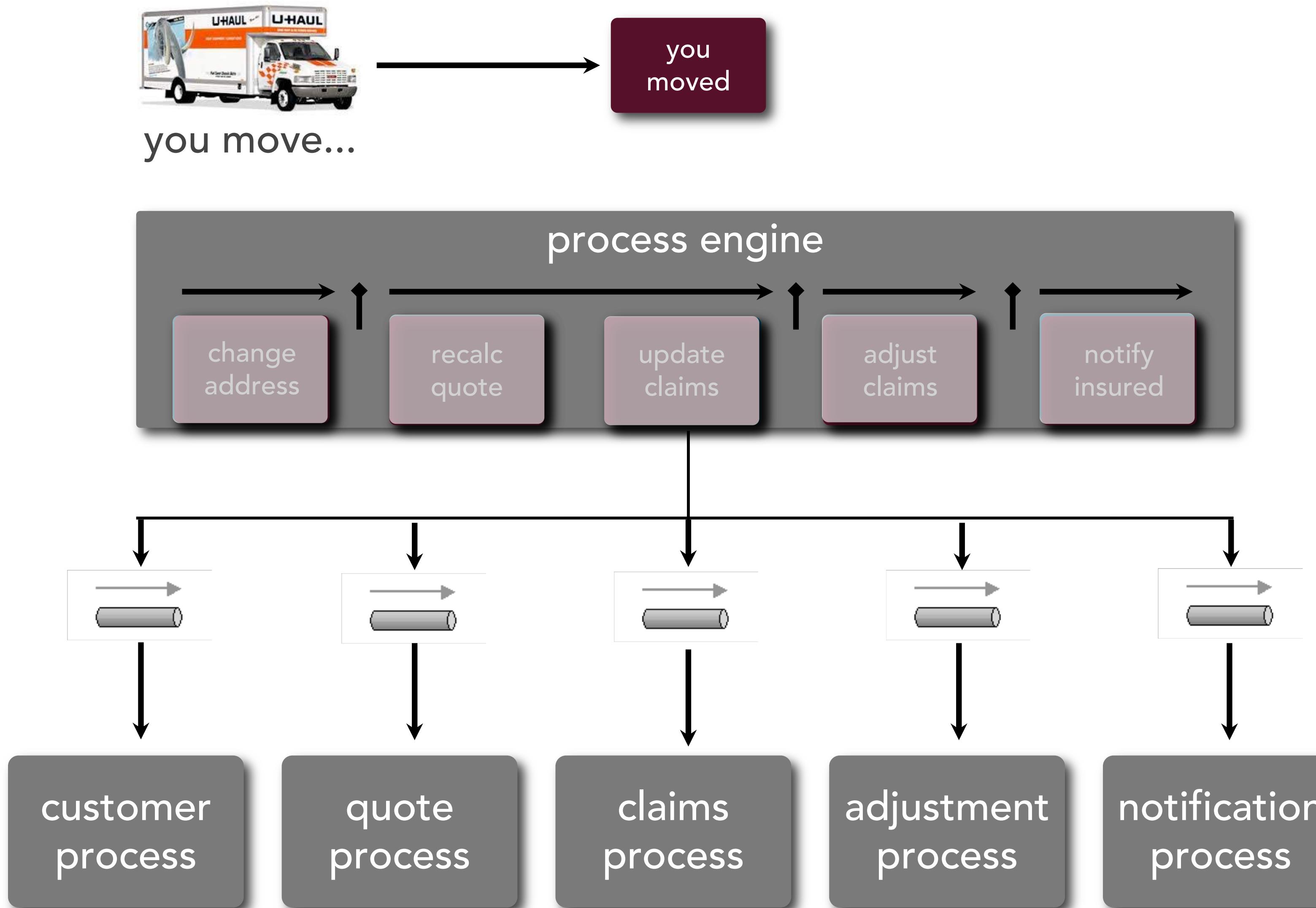
mediator message flow



mediator message flow



mediator message flow



business services

BS

BS

BS

BS

BS

BS

message bus

process choreographer

sERviCe-oRieNtEd ArcHiTeCtuRe

enterprise services

ES

ES

ES

ES

ES

ES

application services

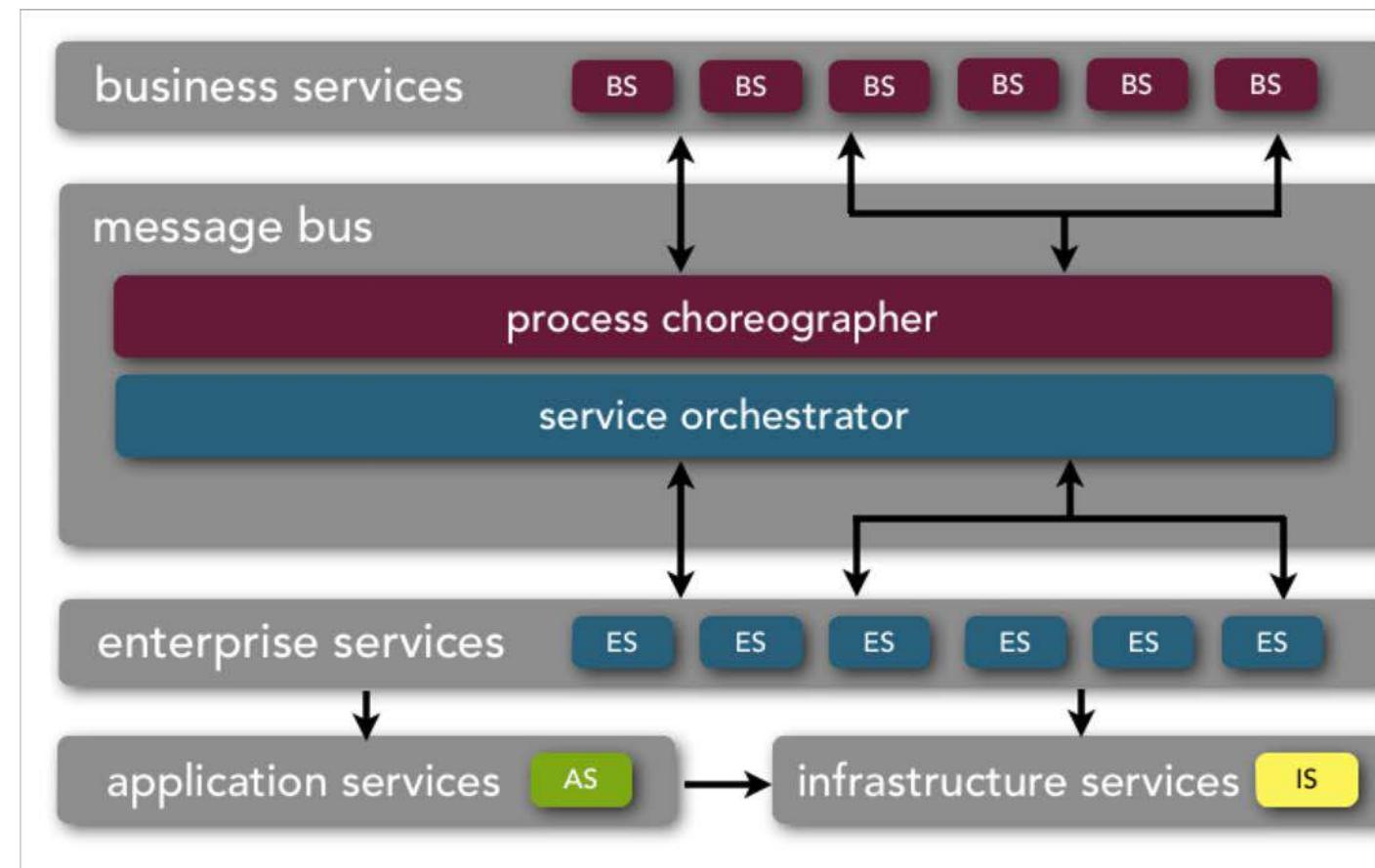
AS

infrastructure services

IS

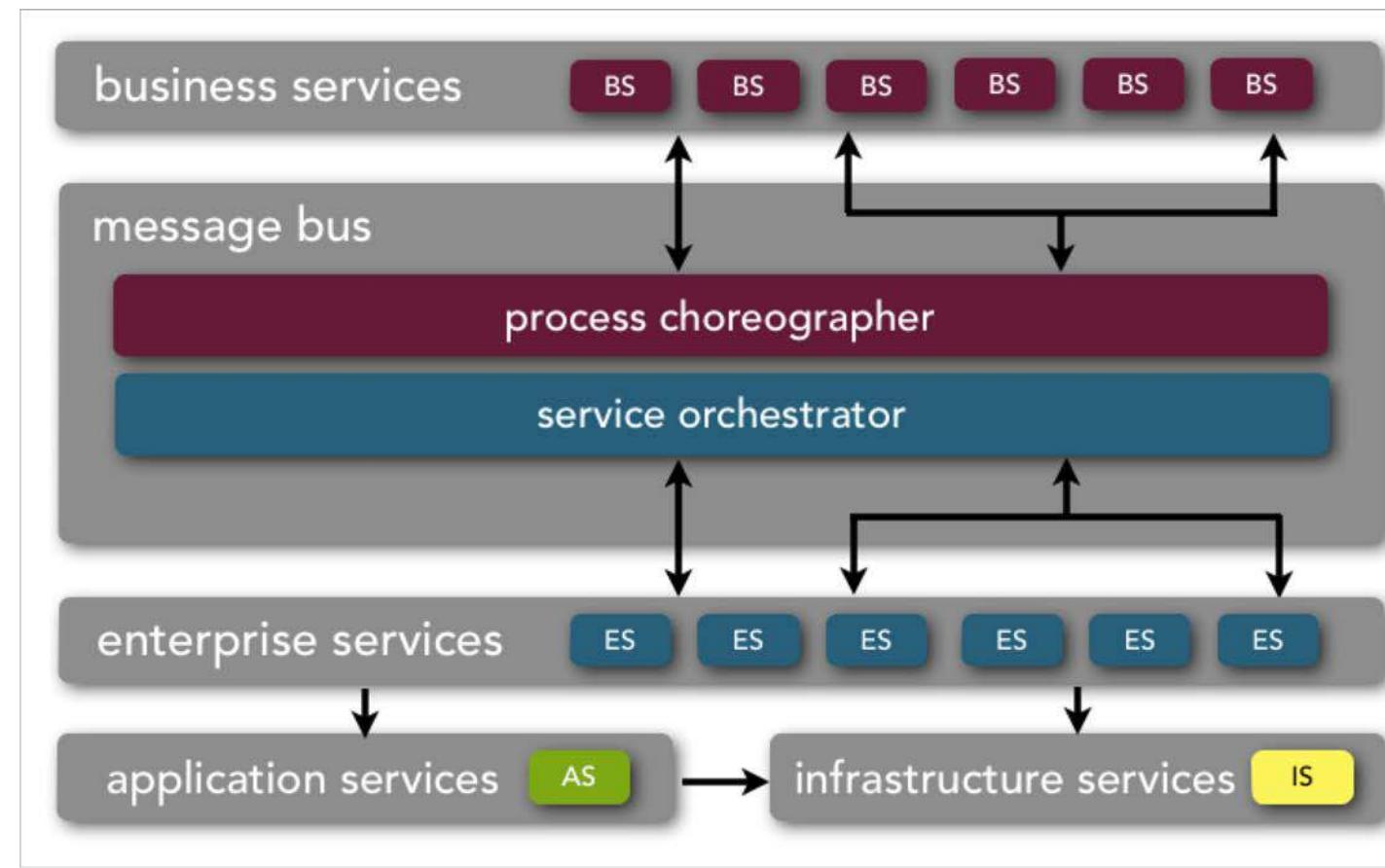


service-oriented architecture



abstraction

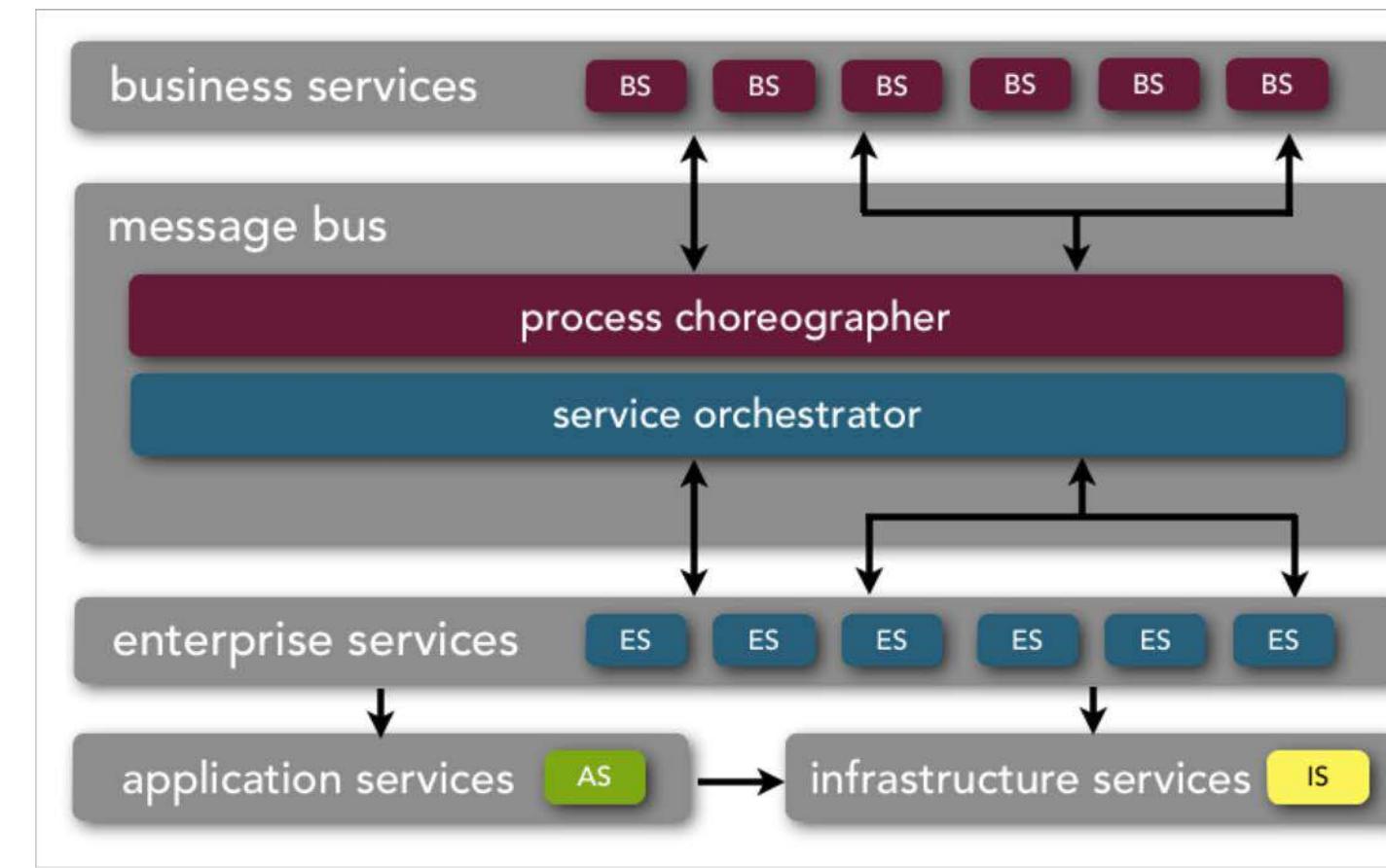
service-oriented architecture



abstraction

service taxonomy

service-oriented architecture

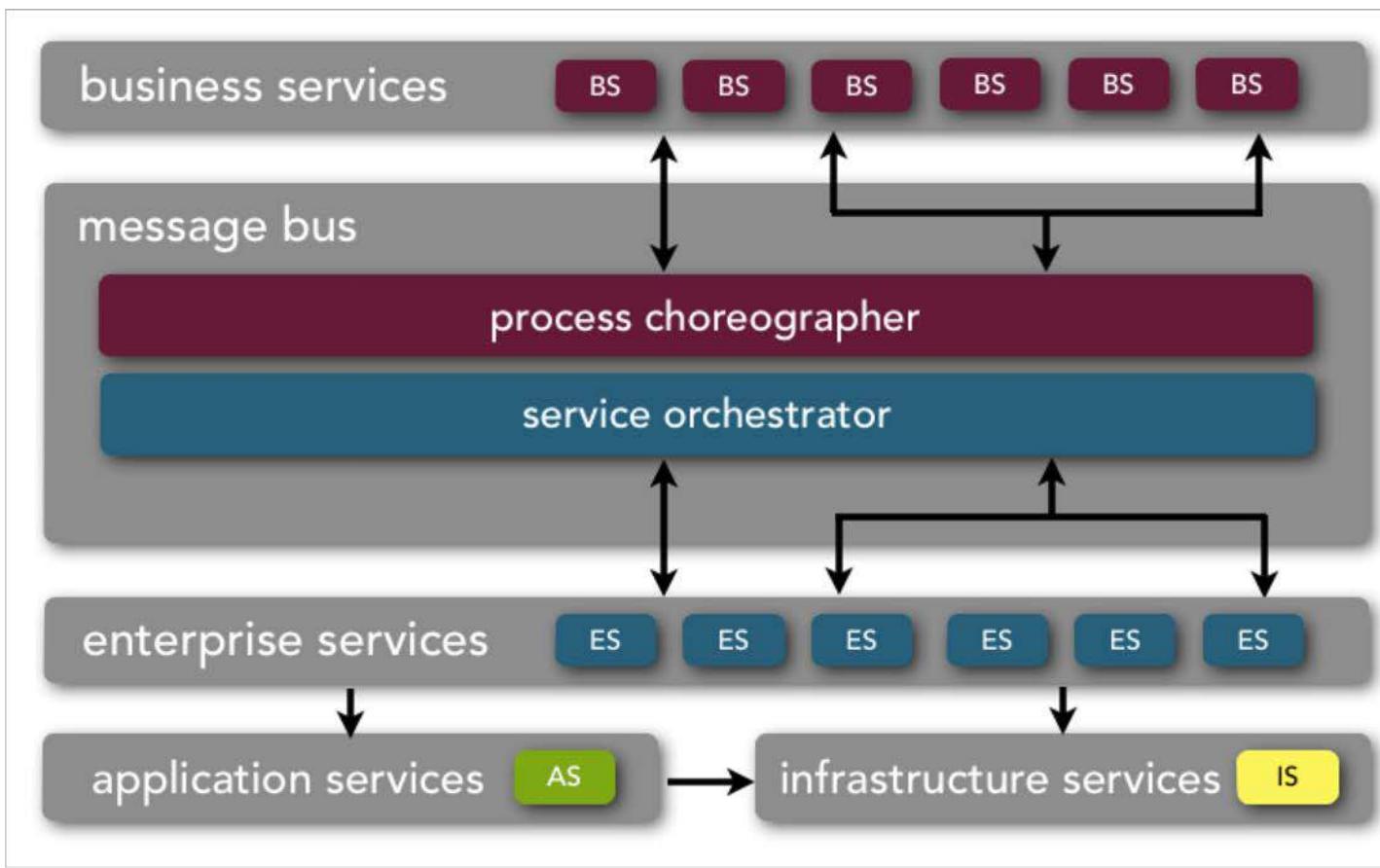


abstraction

service taxonomy

shared resources

service-oriented architecture



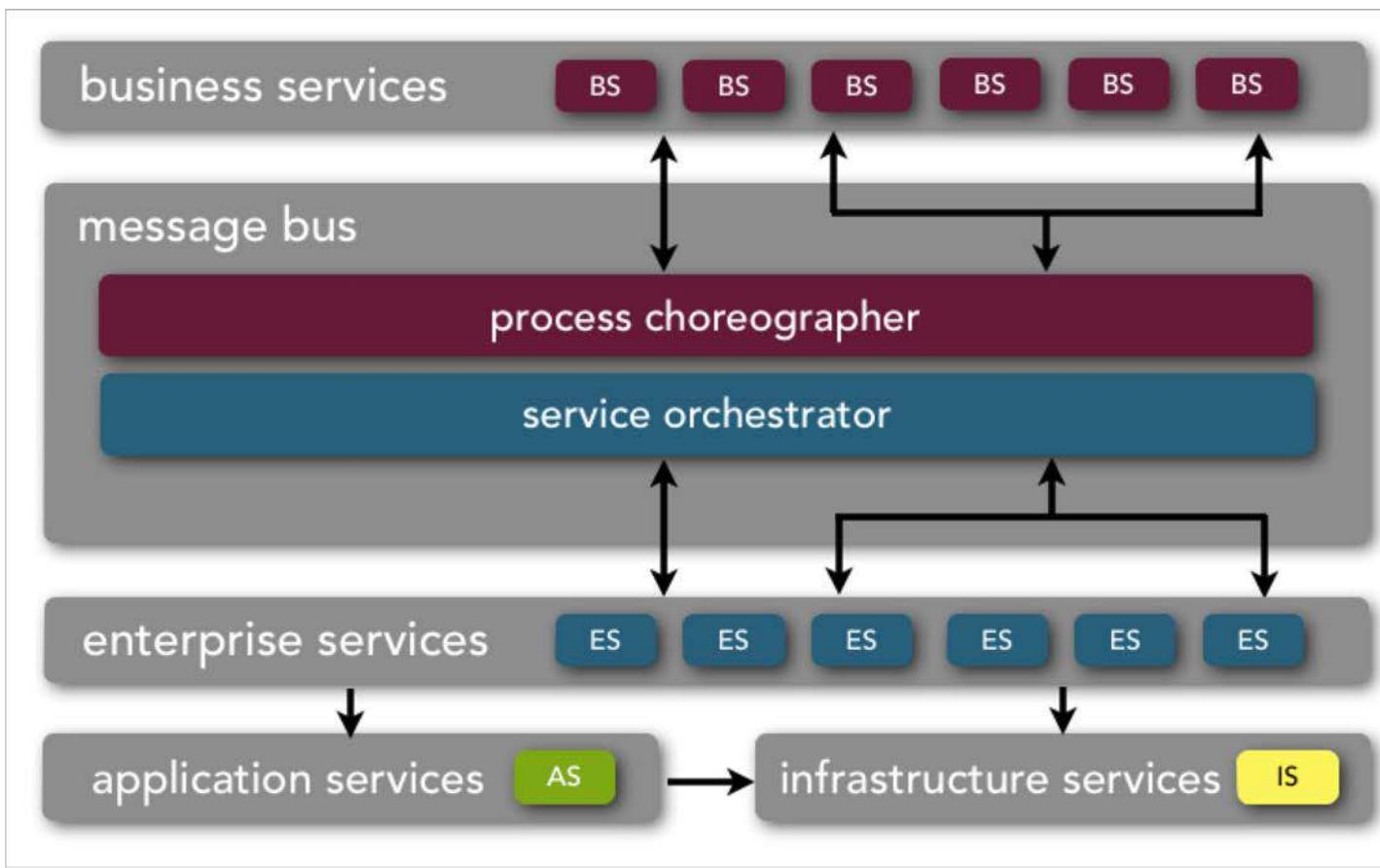
abstraction

service taxonomy

shared resources

middleware

service-oriented architecture



abstraction

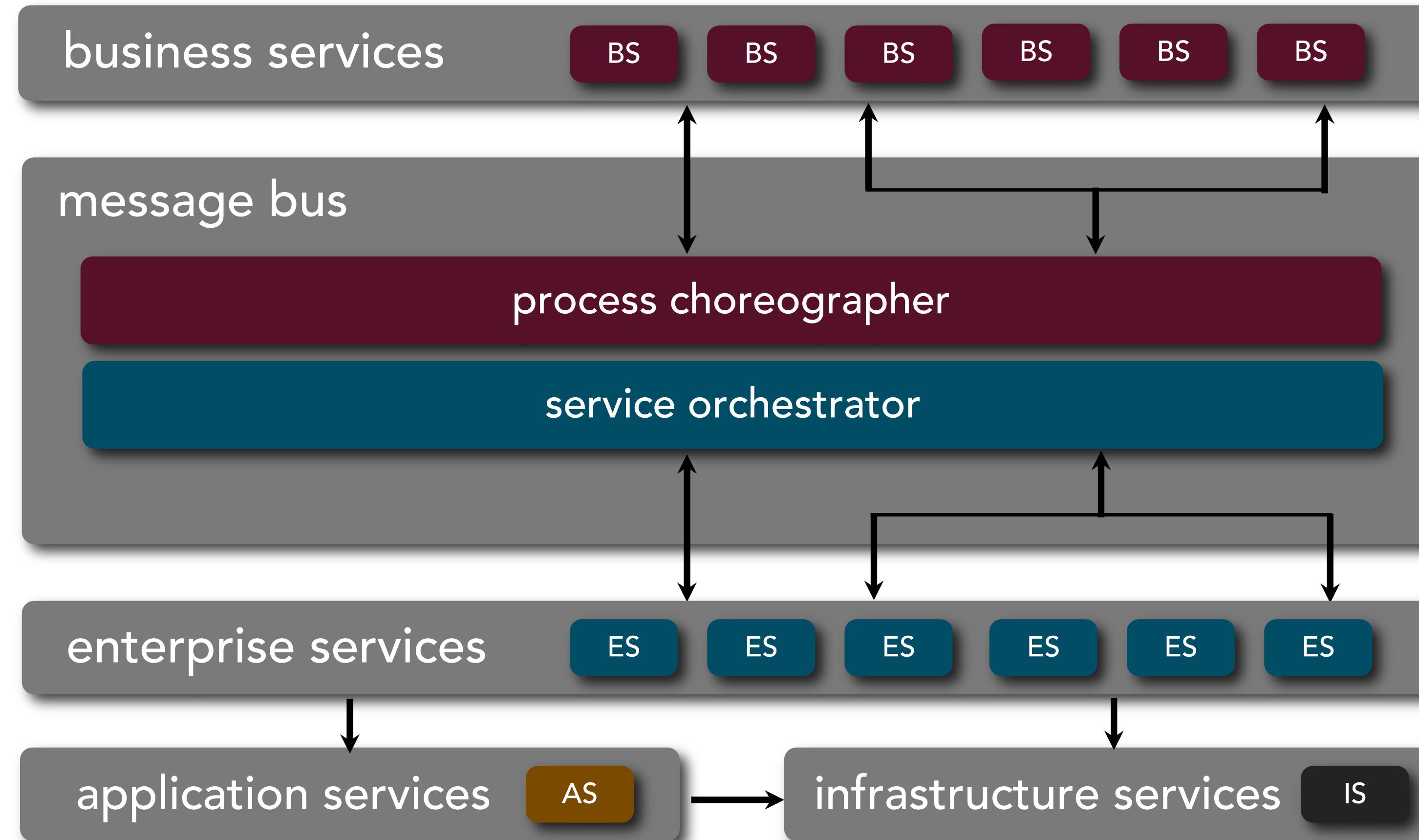
service taxonomy

shared resources

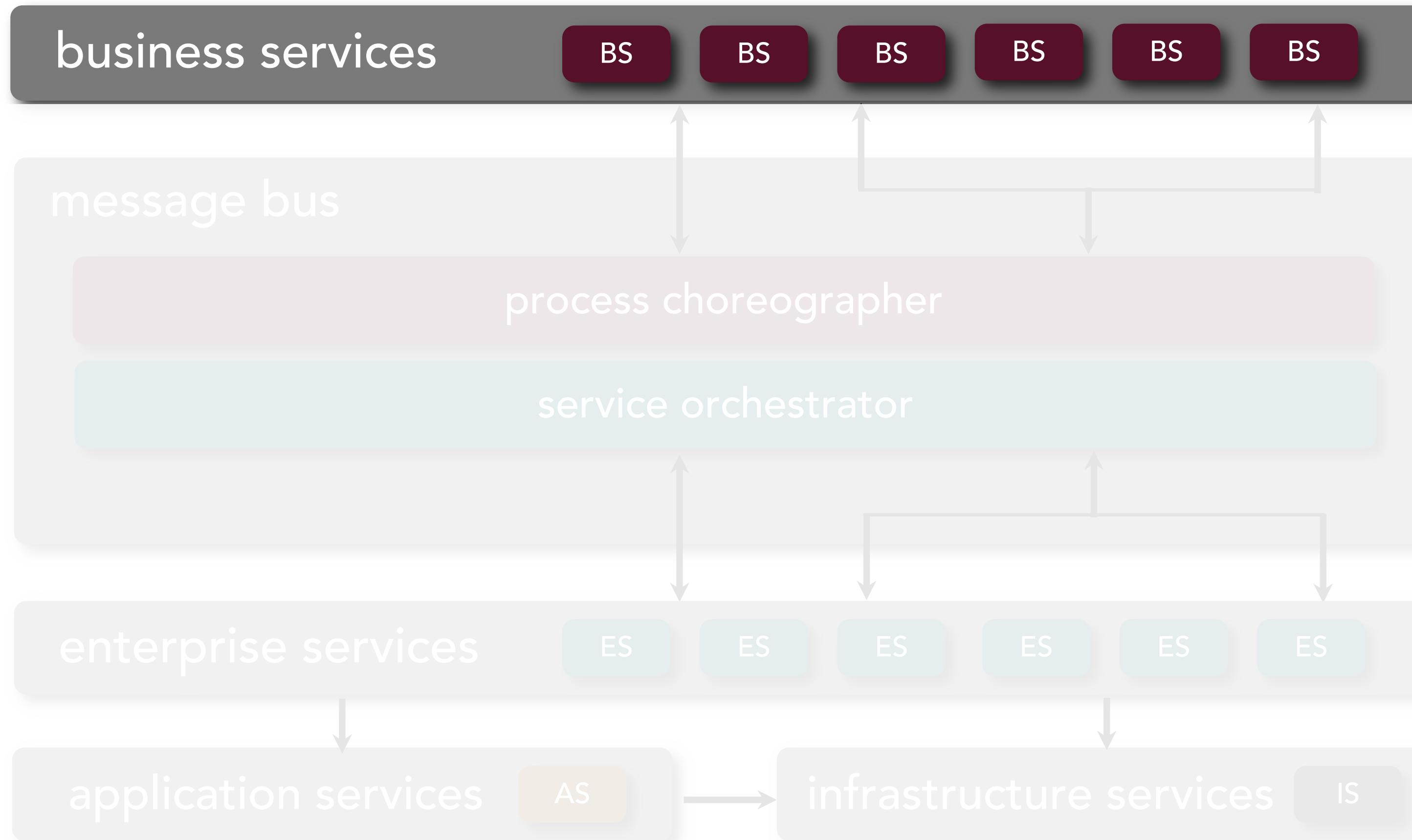
middleware

interoperability

service-oriented architecture



service-oriented architecture



service-oriented architecture

business services

BS

BS

BS

BS

BS

BS

coarse-grained

service-oriented architecture

business services

BS

BS

BS

BS

BS

BS

abstract

coarse-grained

service-oriented architecture

business services

BS

BS

BS

BS

BS

BS

abstract

enterprise-level

coarse-grained

service-oriented architecture

business services

BS

BS

BS

BS

BS

BS

abstract

enterprise-level

coarse-grained

owned and defined by business users

data represented as WSDL, BPEL, XML, etc.

service-oriented architecture

business services

BS

BS

BS

BS

BS

BS

abstract

enterprise-level

coarse-grained

owned and defined by business users

data represented as WSDL, BPEL, XML, etc.

no implementation - only name, input, and output

ExecuteTrade

PlaceOrder

ProcessClaim

service-oriented architecture

business services

BS

BS

BS

BS

BS

BS

abstract

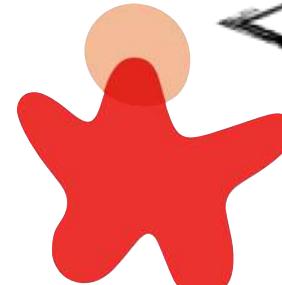
enterprise-level

coarse-grained

owned and defined by business users

data represented as WSDL, BPEL, XML, etc.

no implementation - only name, input, and output



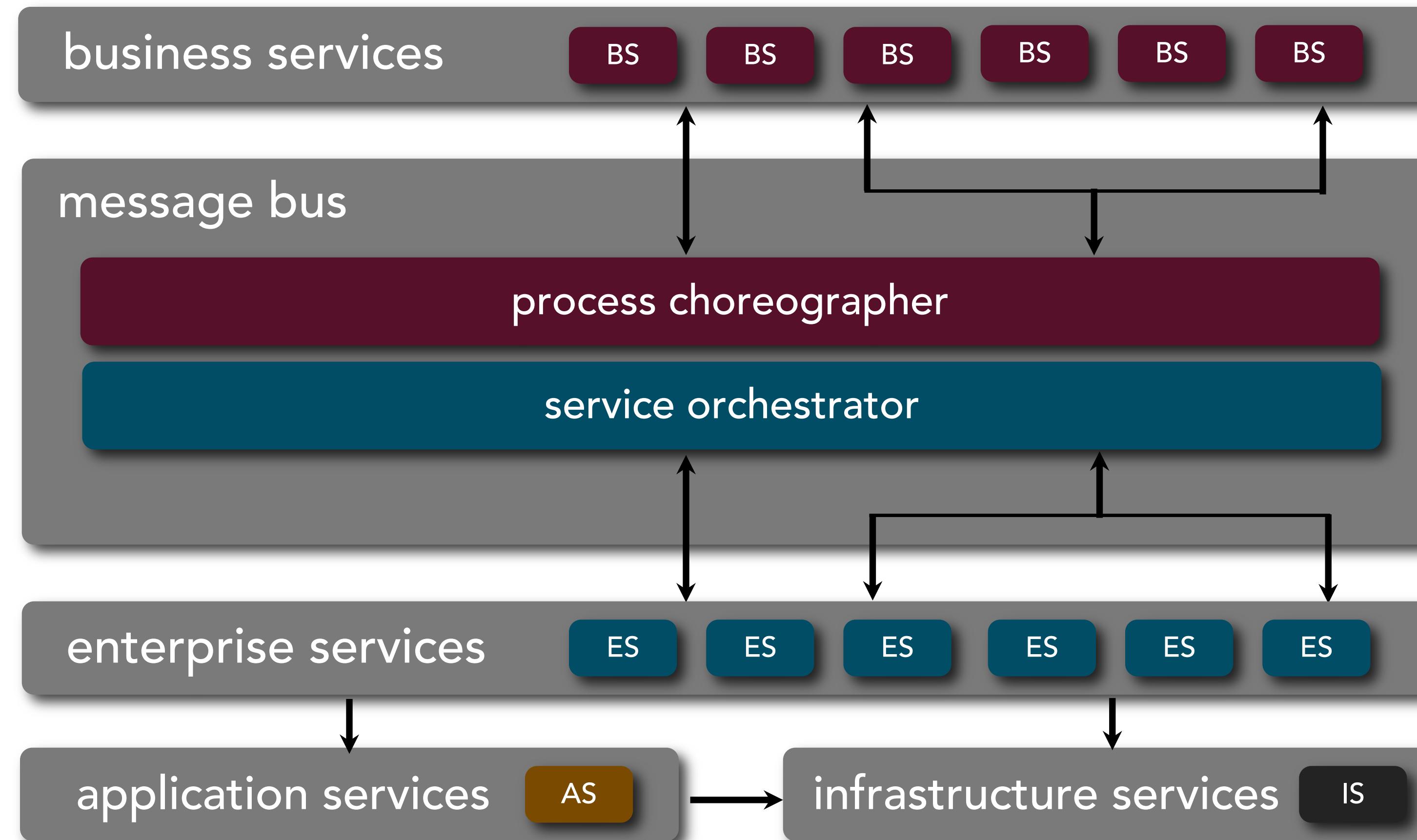
Are we in the business of...?

ExecuteTrade

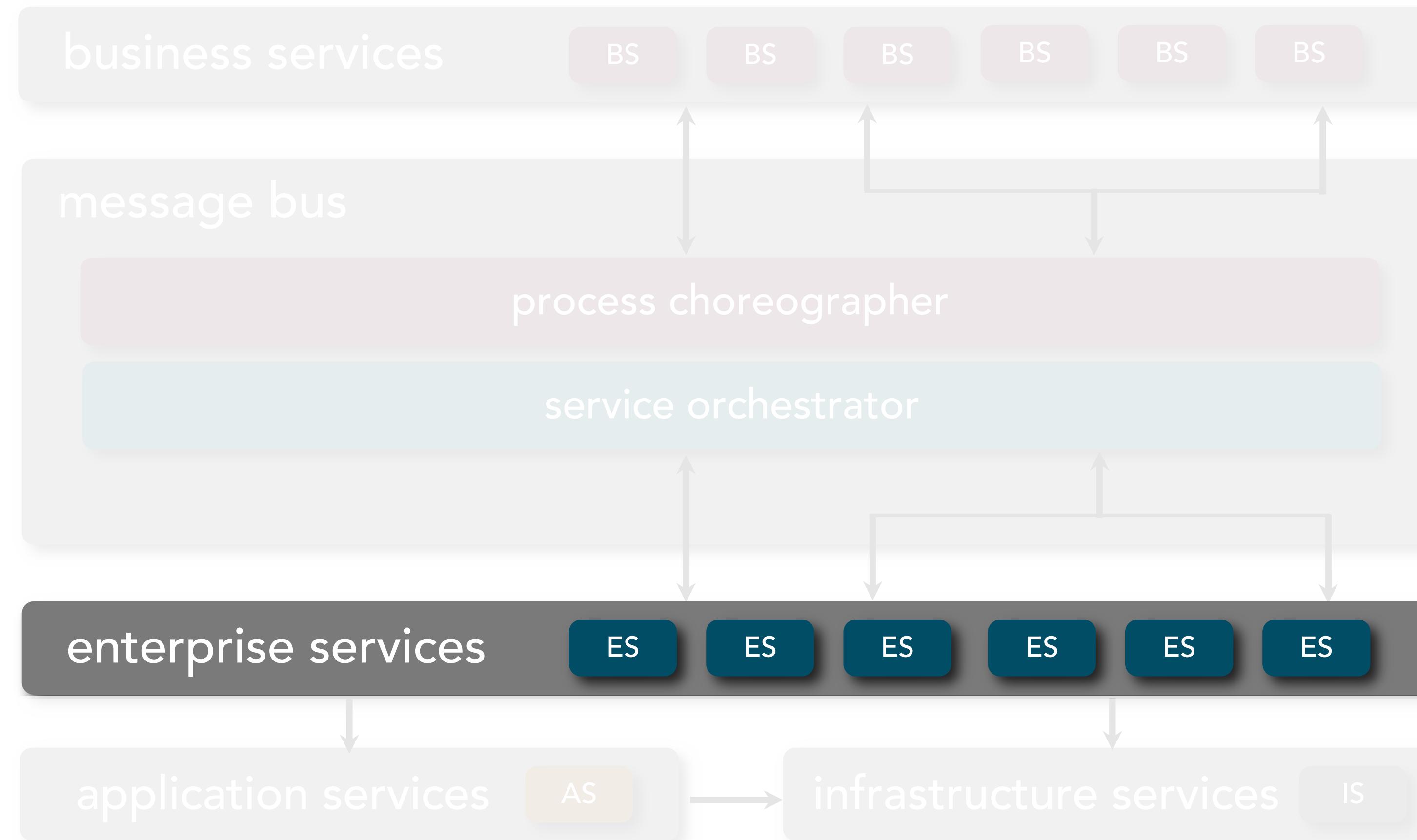
PlaceOrder

ProcessClaim

service-oriented architecture



service-oriented architecture



service-oriented architecture

owned by shared services teams

enterprise services

ES

ES

ES

ES

ES

ES

service-oriented architecture

owned by shared services teams

coarse-grained

enterprise services

ES

ES

ES

ES

ES

ES

service-oriented architecture

owned by shared services teams

enterprise-level

coarse-grained

enterprise services

ES

ES

ES

ES

ES

ES

service-oriented architecture

owned by shared services teams

concrete

enterprise-level

coarse-grained

enterprise services

ES

ES

ES

ES

ES

ES

service-oriented architecture

owned by shared services teams

concrete

enterprise-level

coarse-grained

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

enterprise services

ES

ES

ES

ES

ES

ES

service-oriented architecture

owned by shared services teams

concrete

enterprise-level

coarse-grained

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

enterprise services

ES

ES

ES

ES

ES

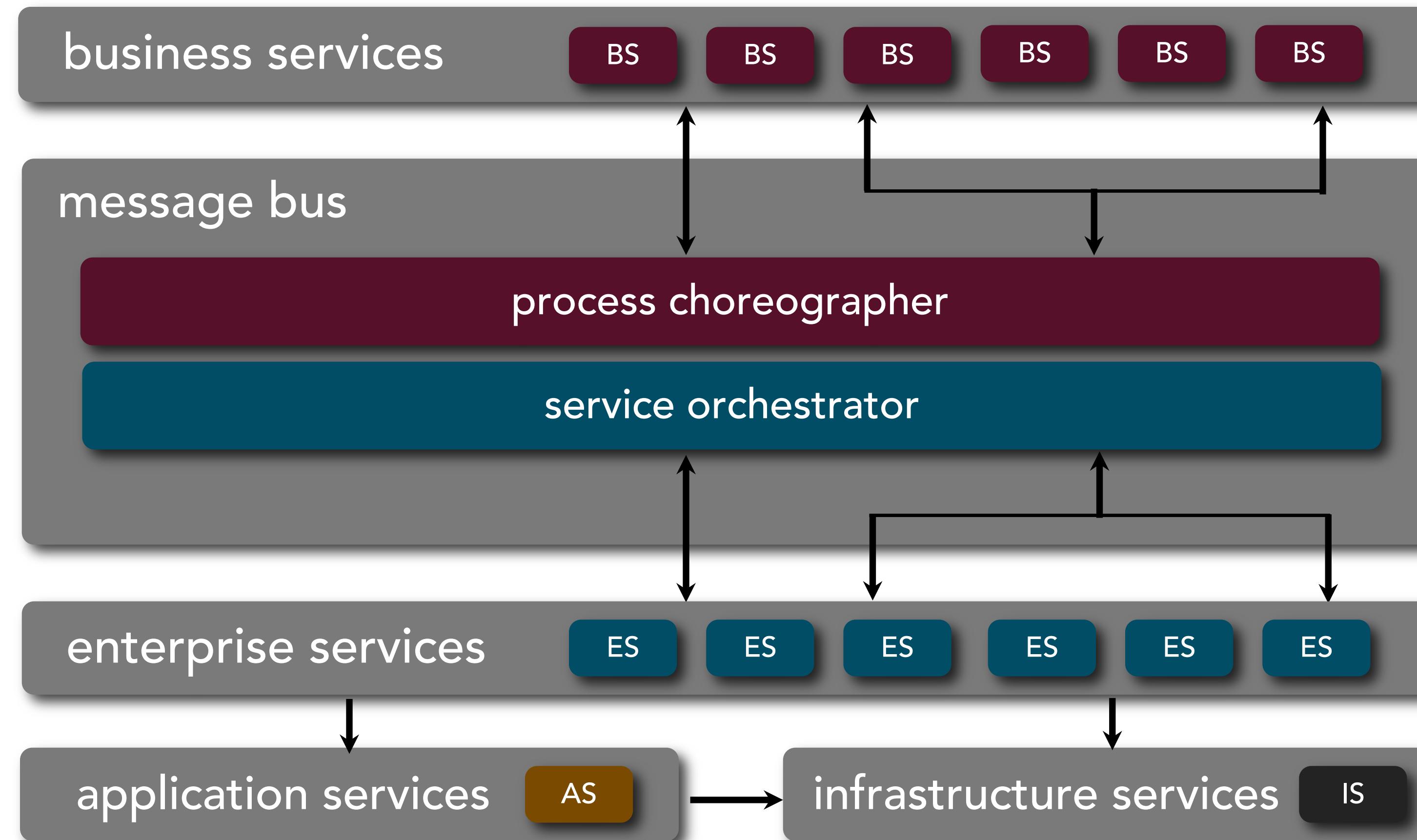
ES

CreateCustomer

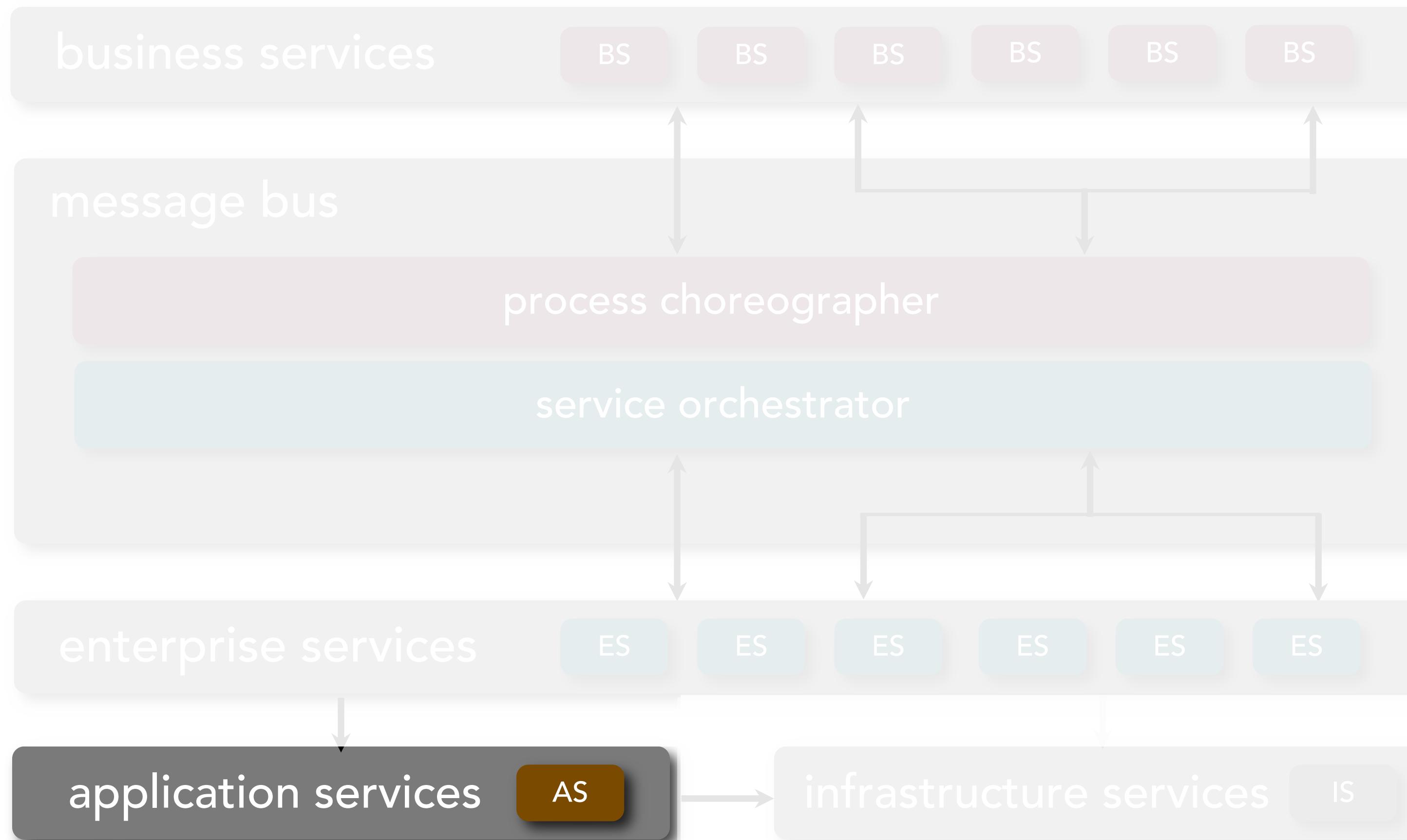
CalcQuote

ValidateTrade

service-oriented architecture



service-oriented architecture



service-oriented architecture

owned by application teams

application services AS

service-oriented architecture

owned by application teams

fine-grained

application services AS

service-oriented architecture

owned by application teams

concrete

fine-grained

application services AS

service-oriented architecture

owned by application teams

concrete

application-level

fine-grained

application services AS

service-oriented architecture

owned by application teams

concrete

application-level

fine-grained

bound to a specific application context

application services AS

service-oriented architecture

owned by application teams

concrete

application-level

fine-grained

bound to a specific application context

AddDriver

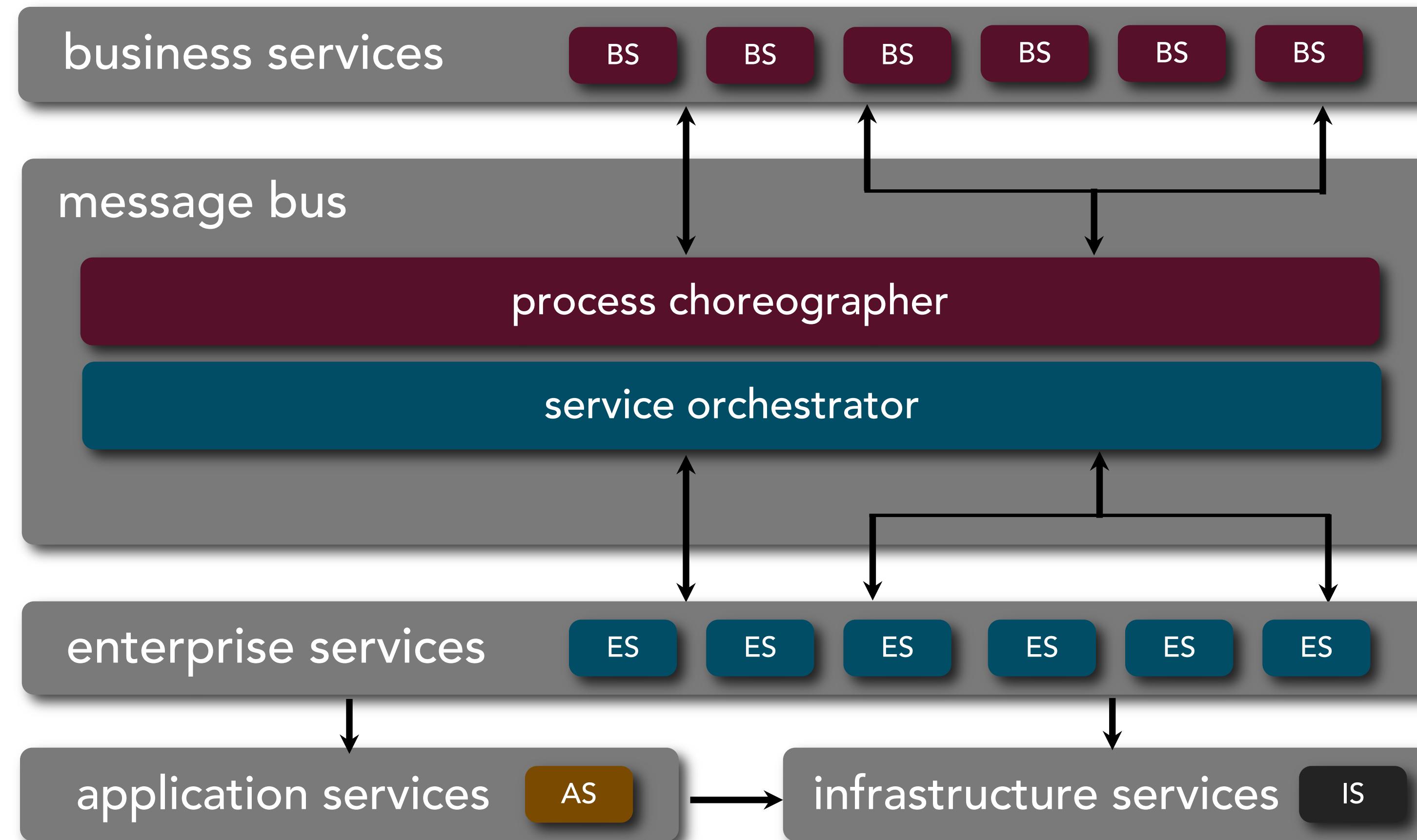
UpdateAddress

CalcSalesTax

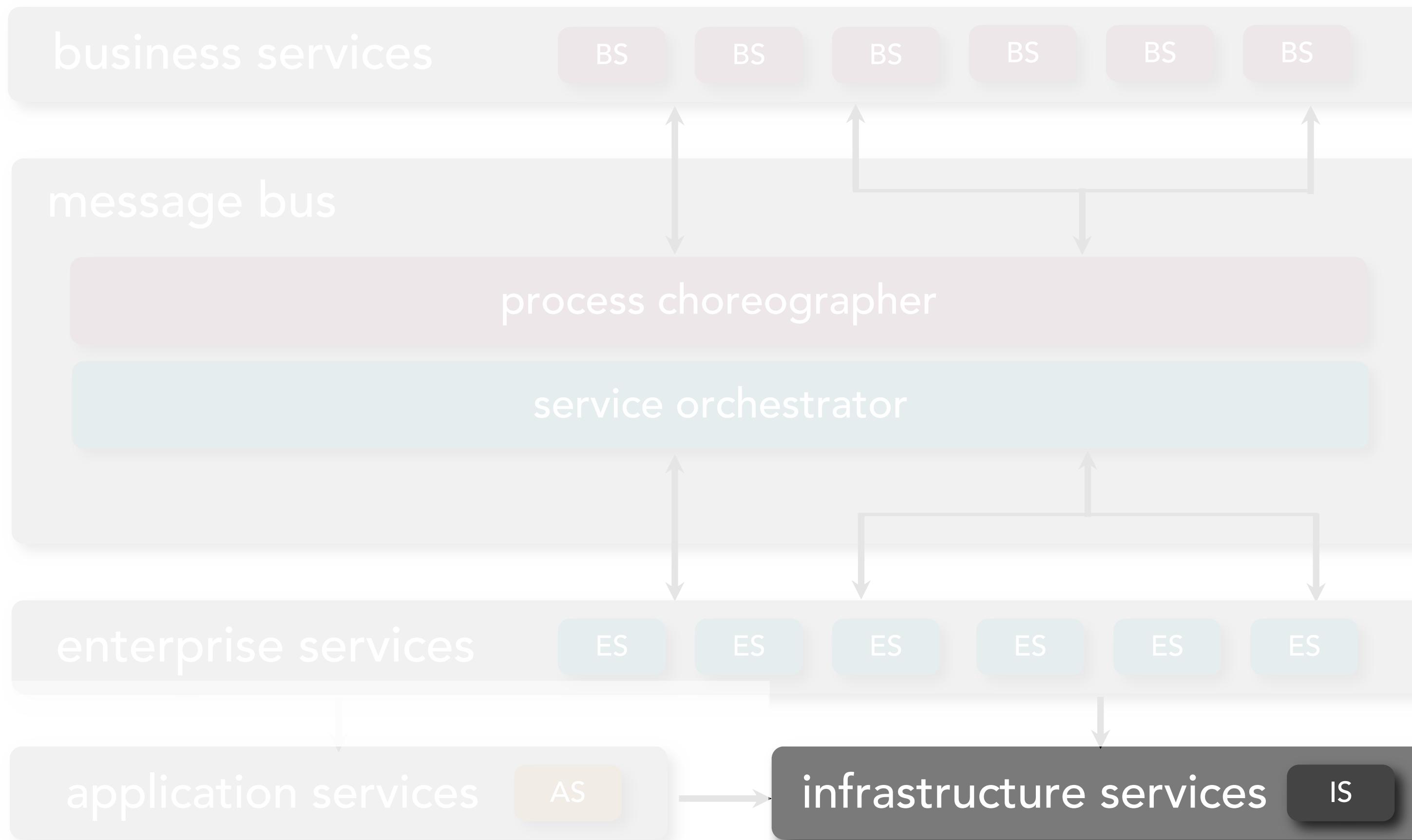
application services

AS

service-oriented architecture



service-oriented architecture



service-oriented architecture

owned by infrastructure or shared services teams

infrastructure services IS

service-oriented architecture

owned by infrastructure or shared services teams

fine-grained

infrastructure services IS

service-oriented architecture

owned by infrastructure or shared services teams

enterprise-level

fine-grained

infrastructure services IS

service-oriented architecture

owned by infrastructure or shared services teams

concrete

enterprise-level

fine-grained

infrastructure services IS

service-oriented architecture

owned by infrastructure or shared services teams

concrete

enterprise-level

fine-grained

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

infrastructure services IS

service-oriented architecture

owned by infrastructure or shared services teams

concrete

enterprise-level

fine-grained

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

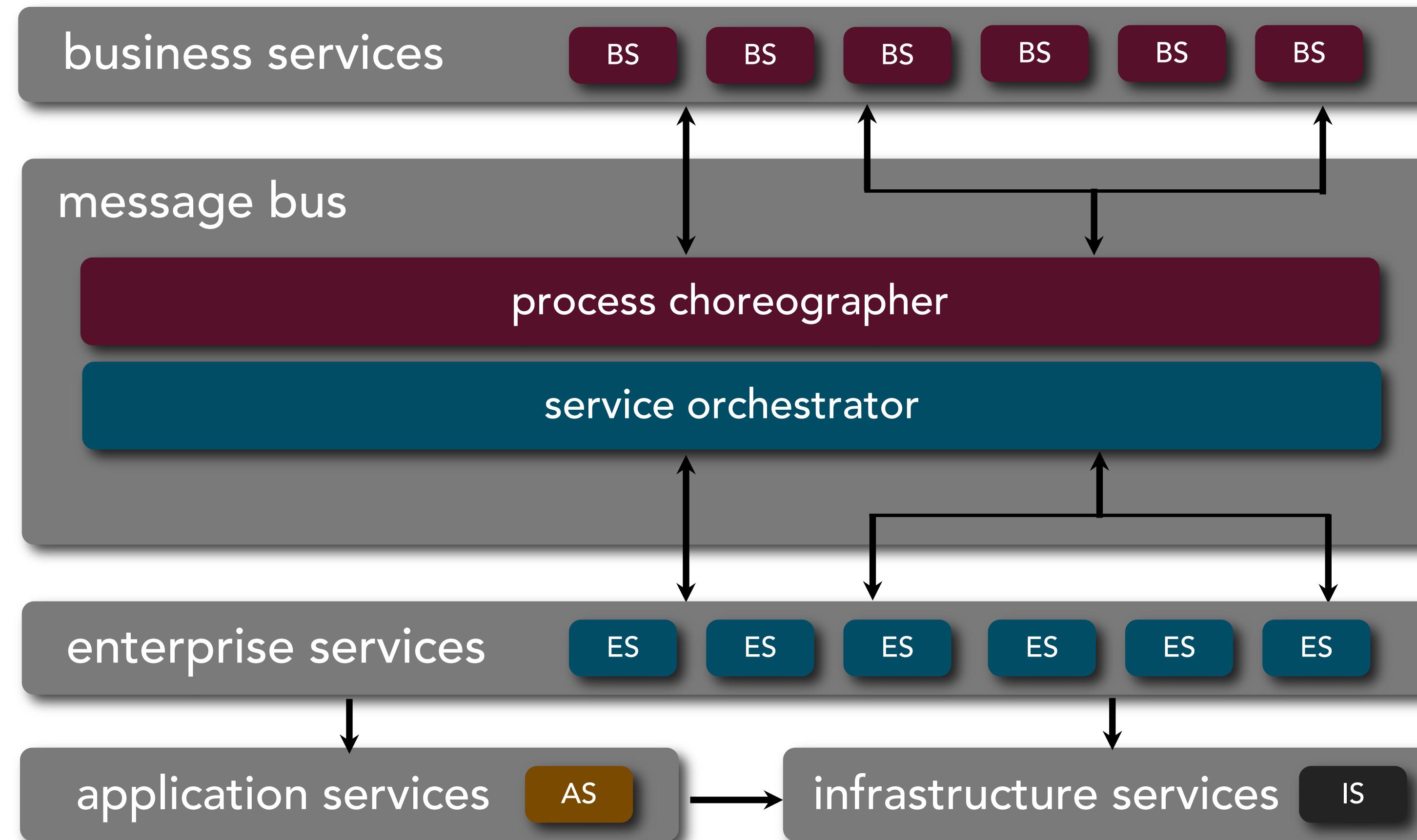
WriteAudit

CheckUserAccess

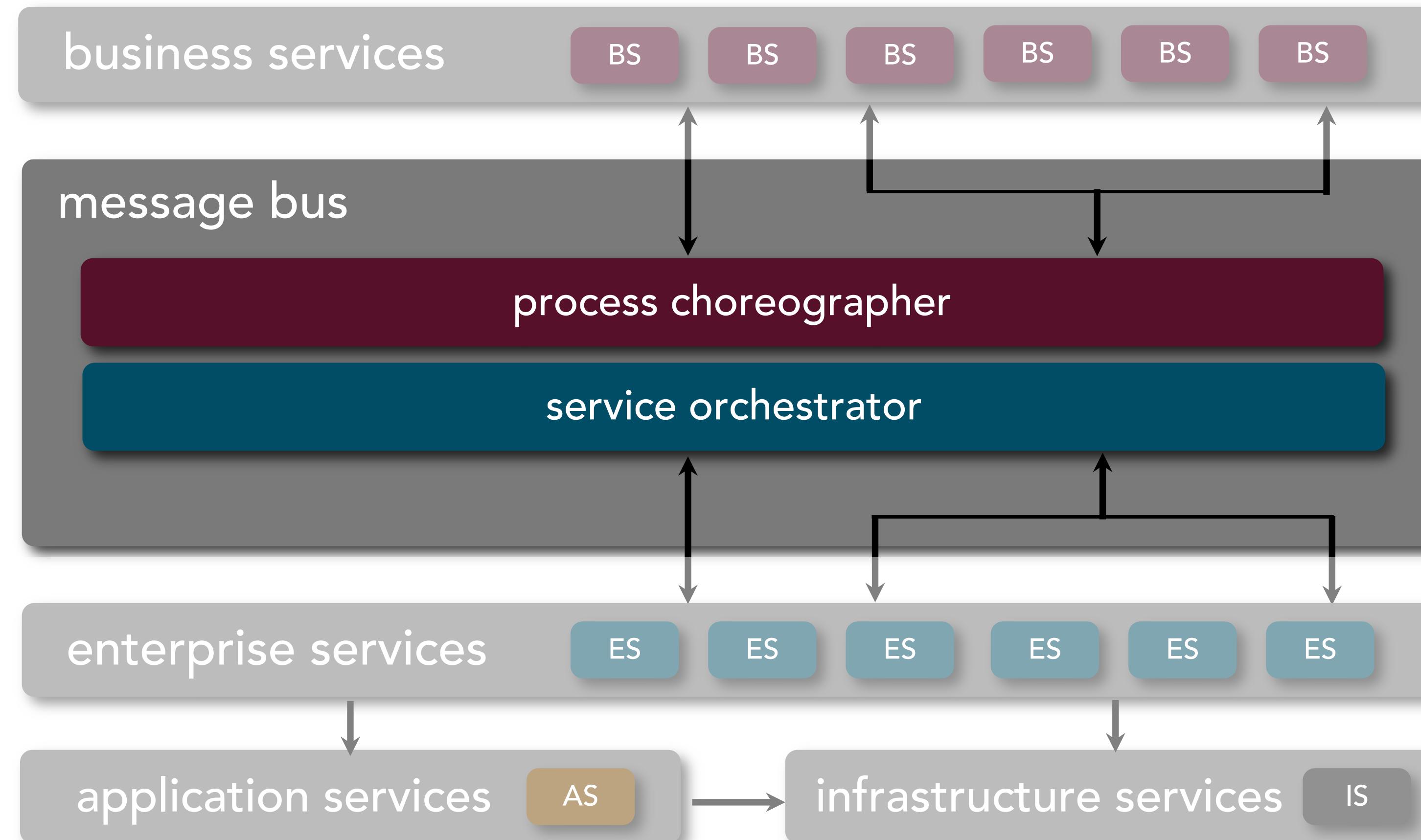
LogError

infrastructure services IS

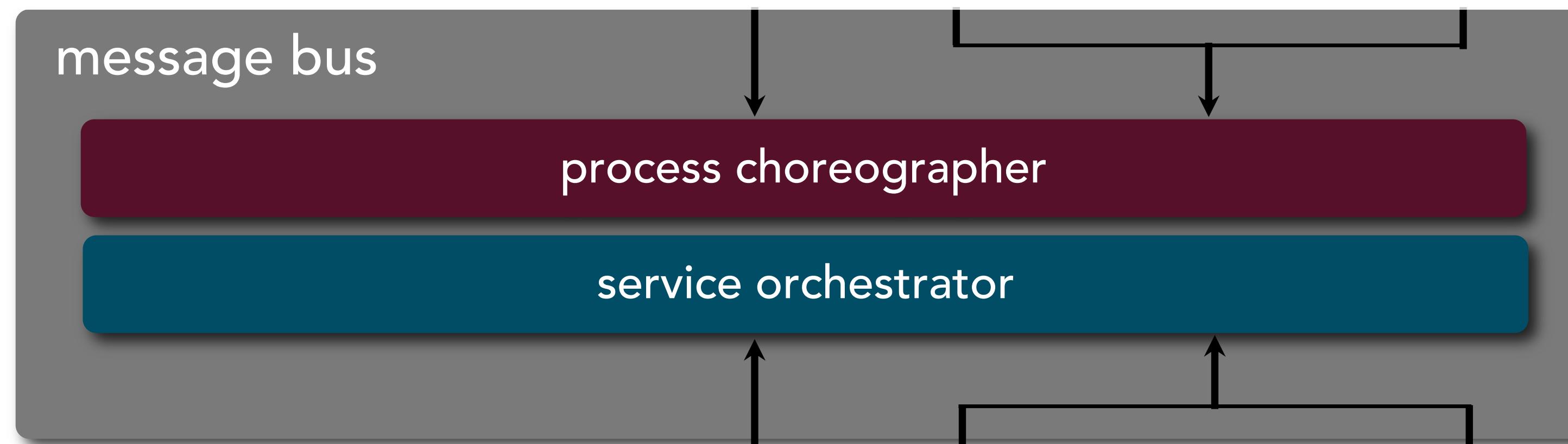
service-oriented architecture



service-oriented architecture



service-oriented architecture



mediation and routing

process choreography

service orchestration

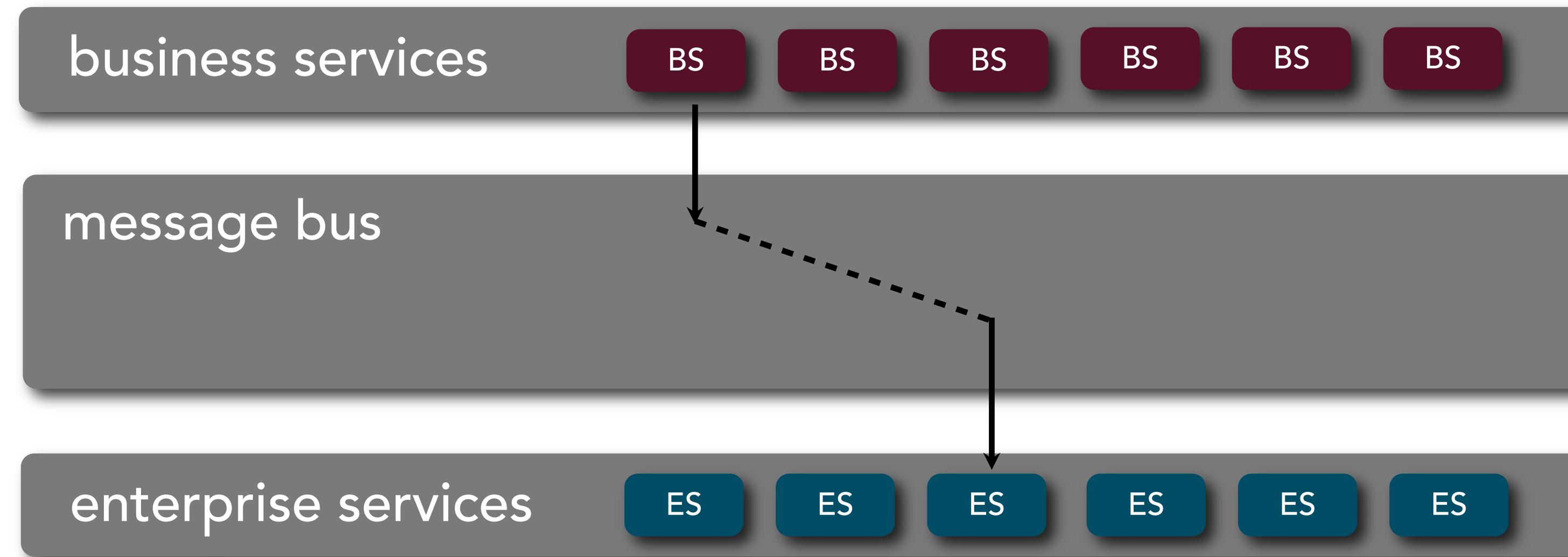
message enhancement

message transformation

protocol transformation

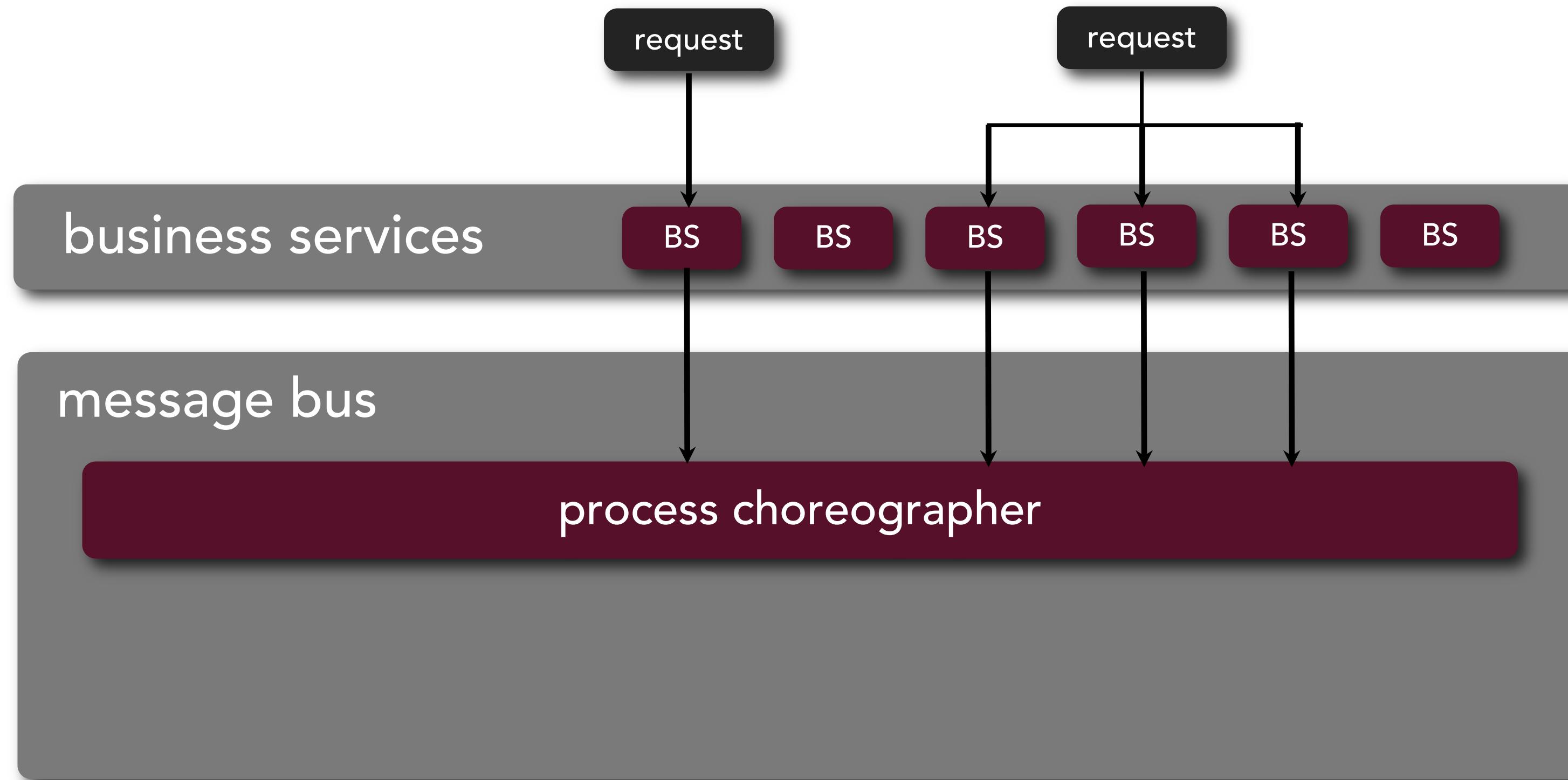
service-oriented architecture

mediation and routing



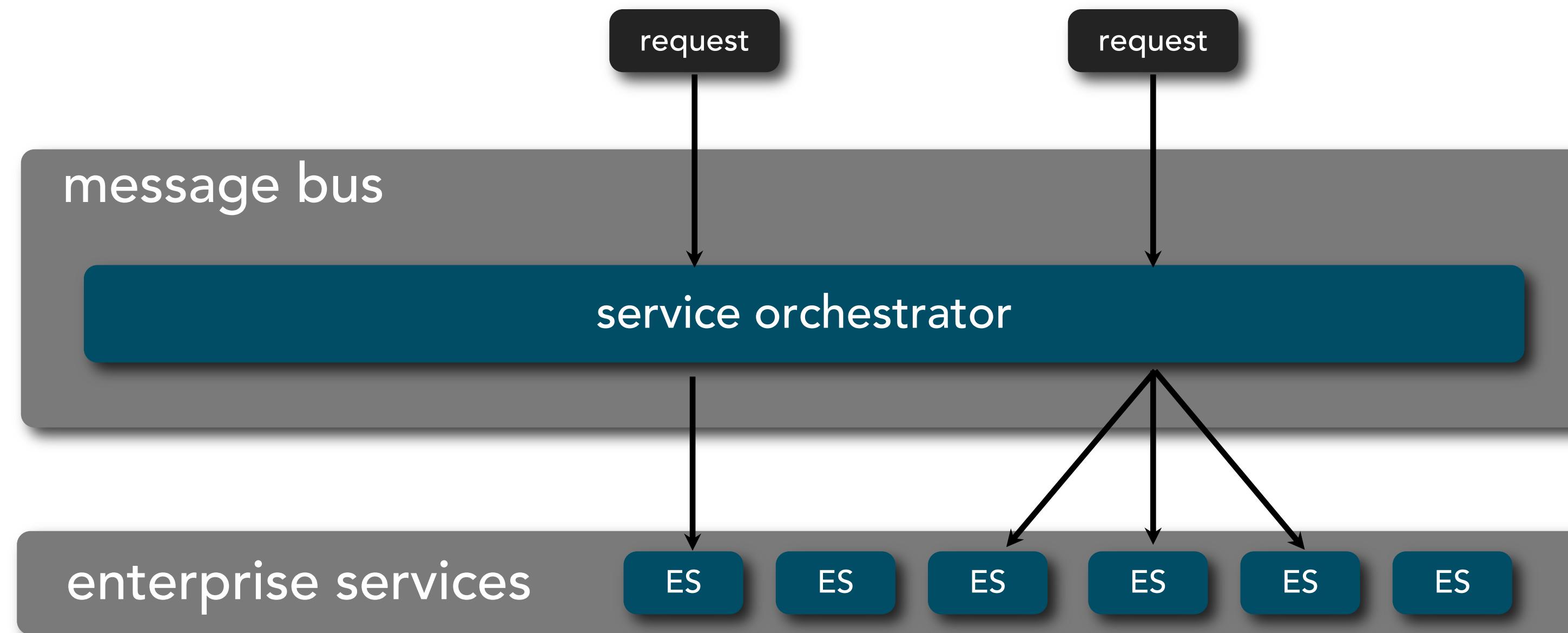
service-oriented architecture

process choreography



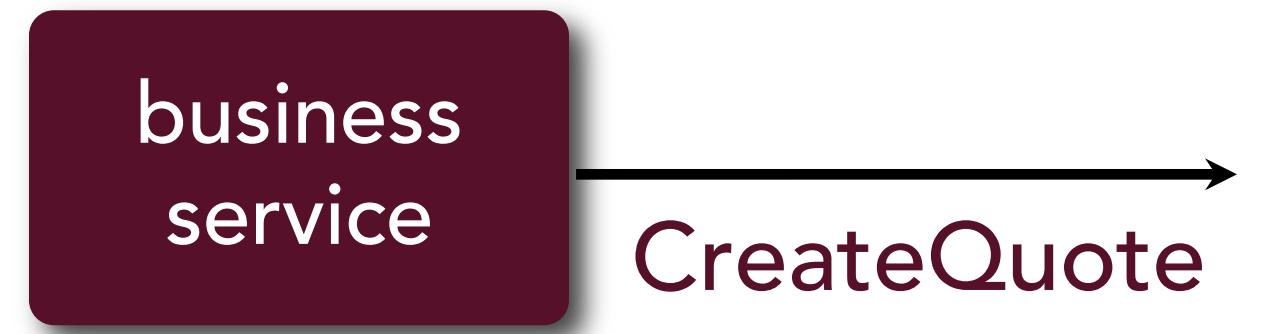
service-oriented architecture

service orchestration



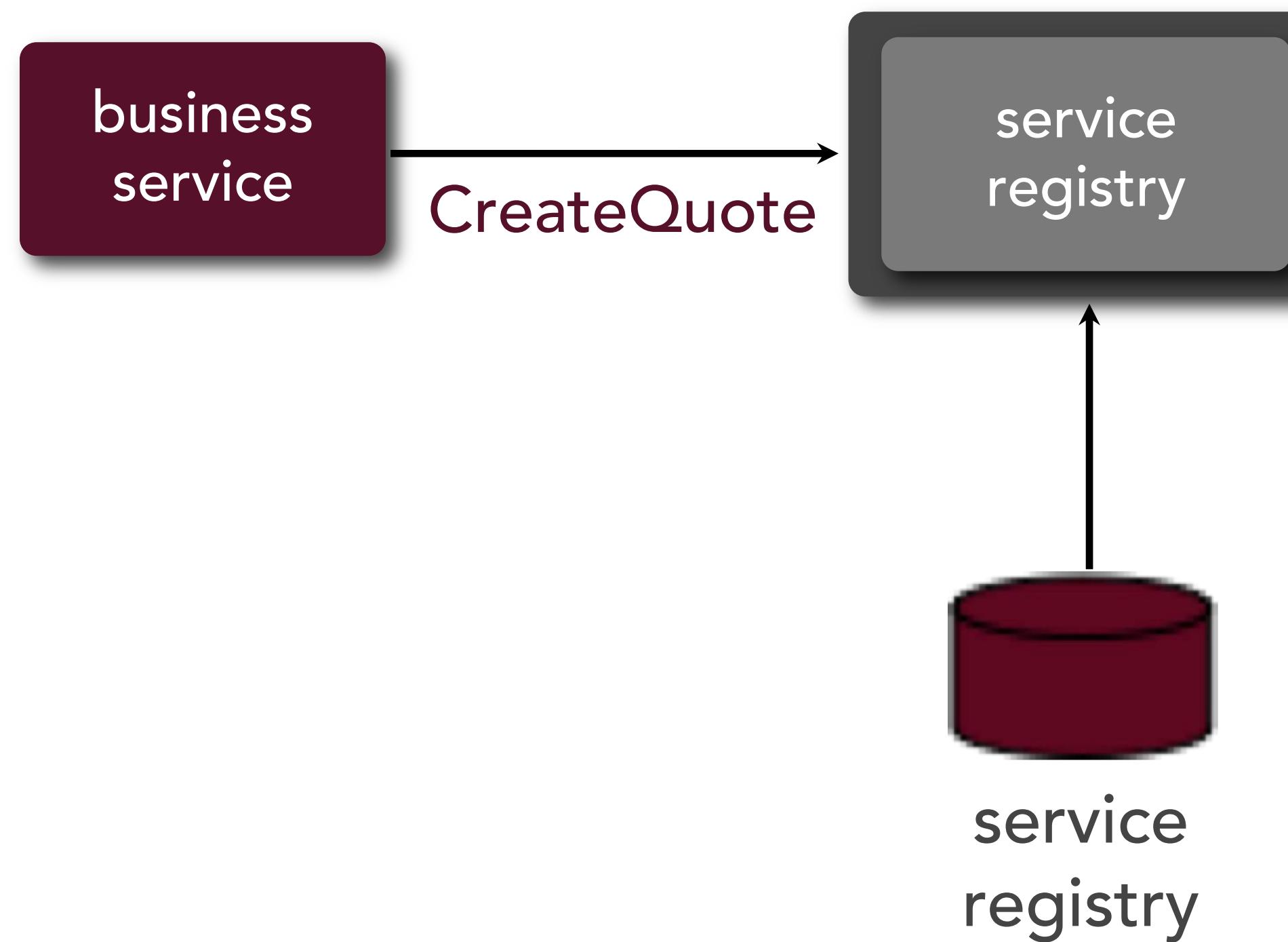
service-oriented architecture

service registry



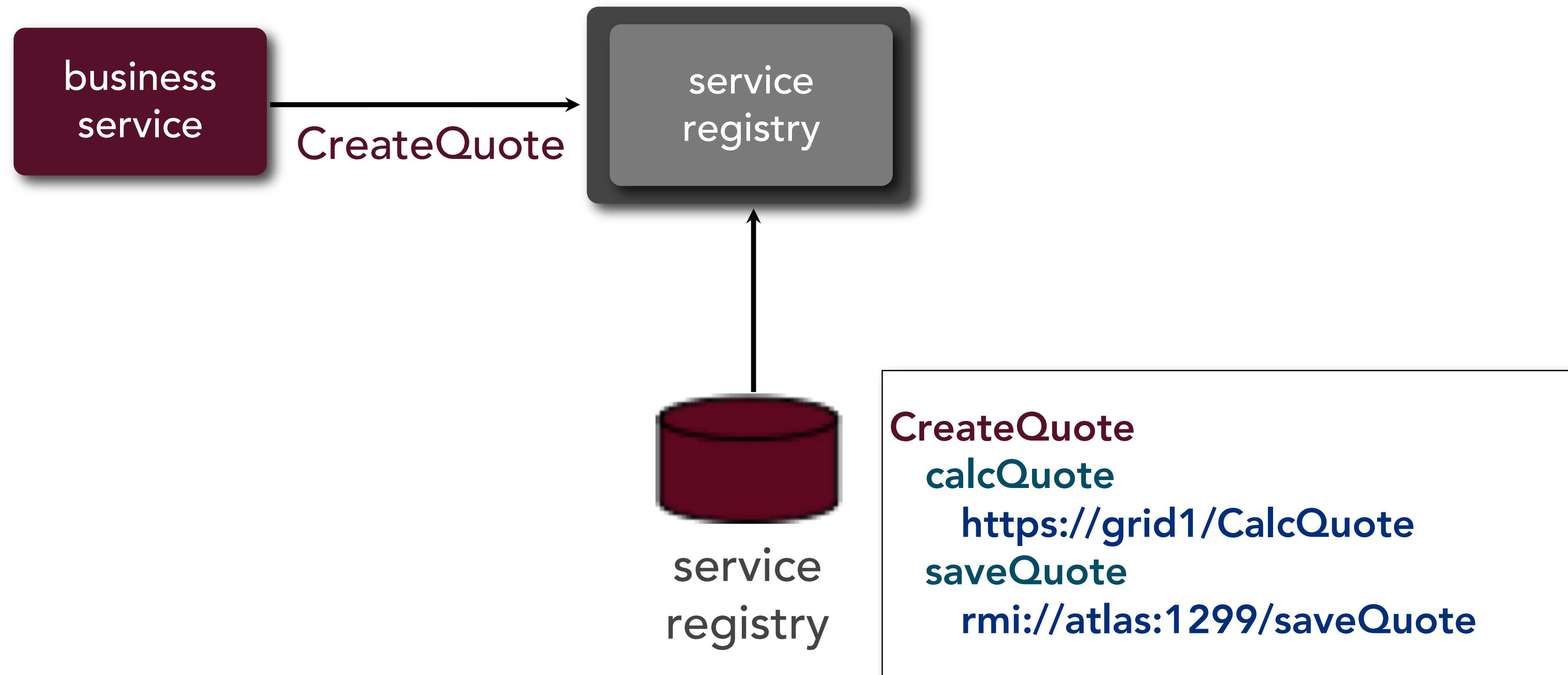
service-oriented architecture

service registry



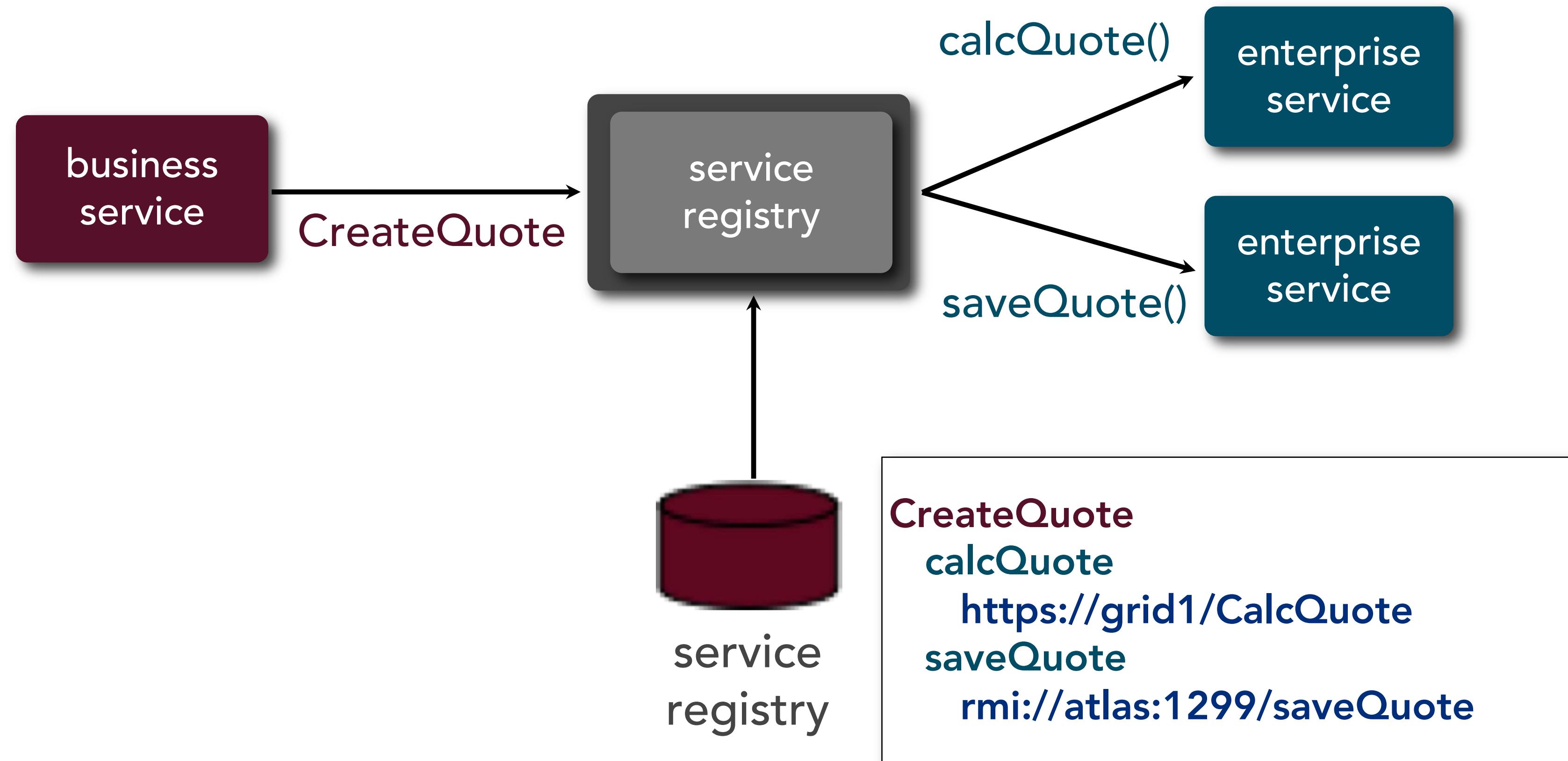
service-oriented architecture

service registry



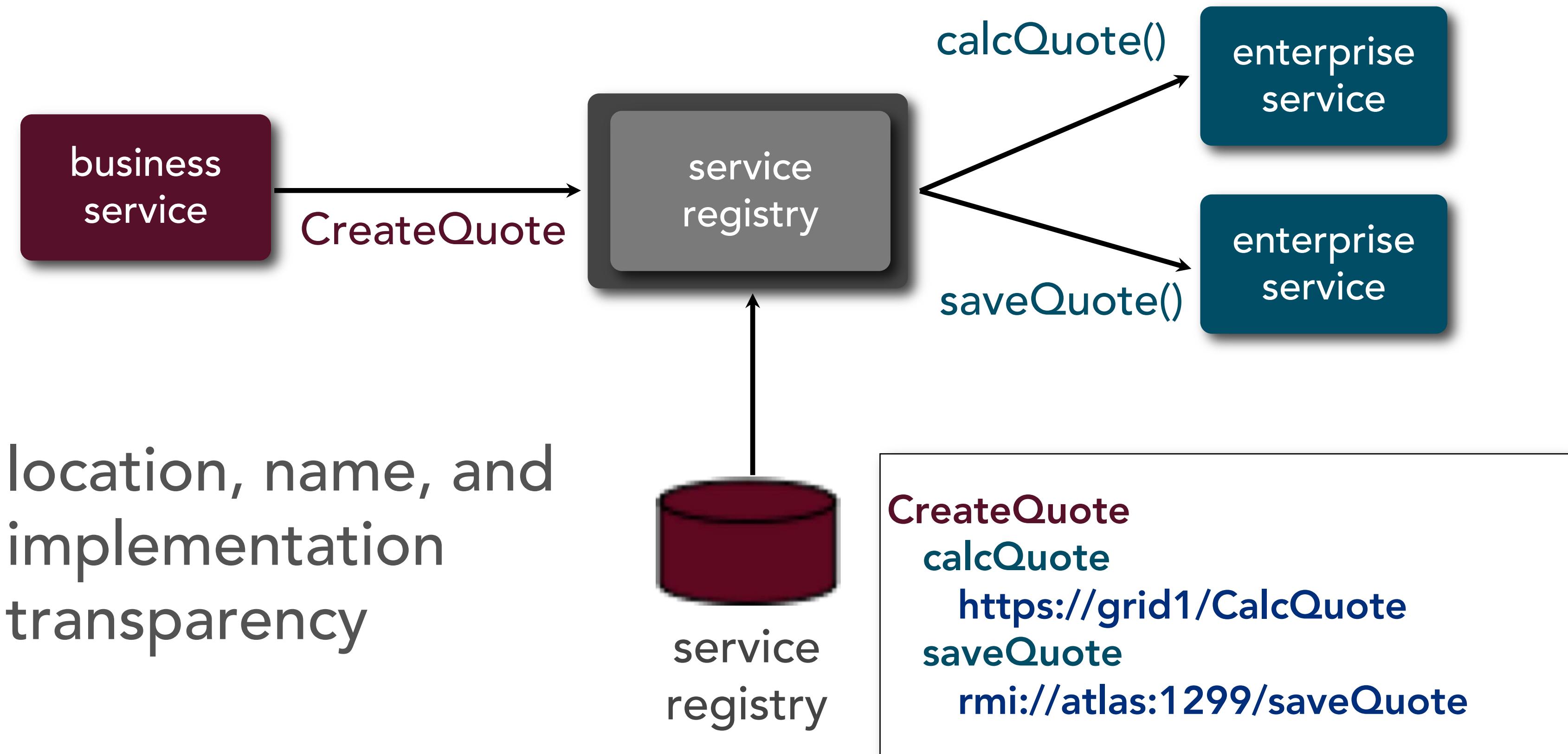
service-oriented architecture

service registry



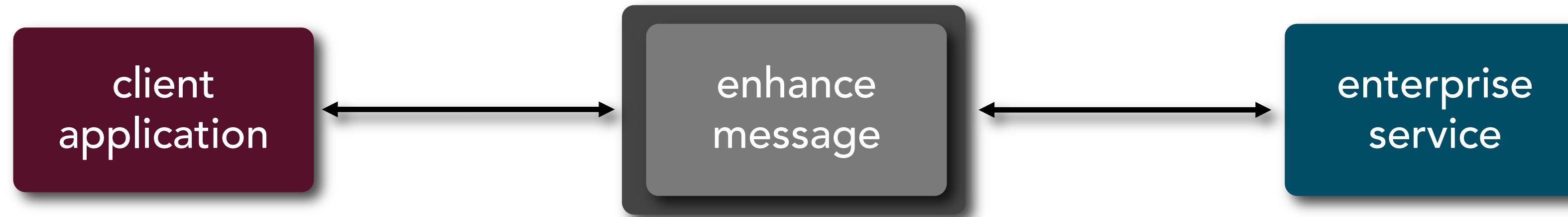
service-oriented architecture

service registry



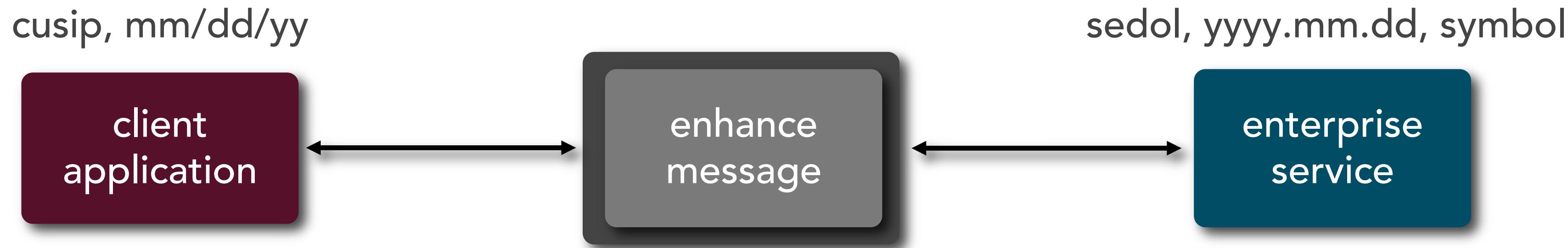
service-oriented architecture

message enhancement



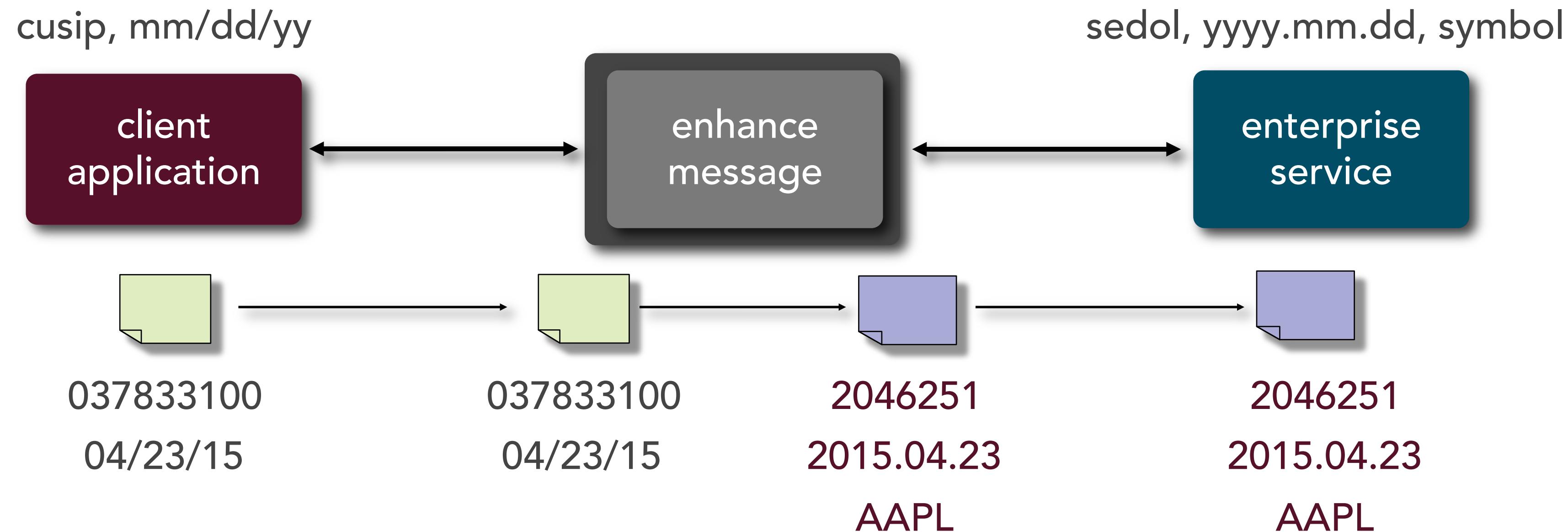
service-oriented architecture

message enhancement



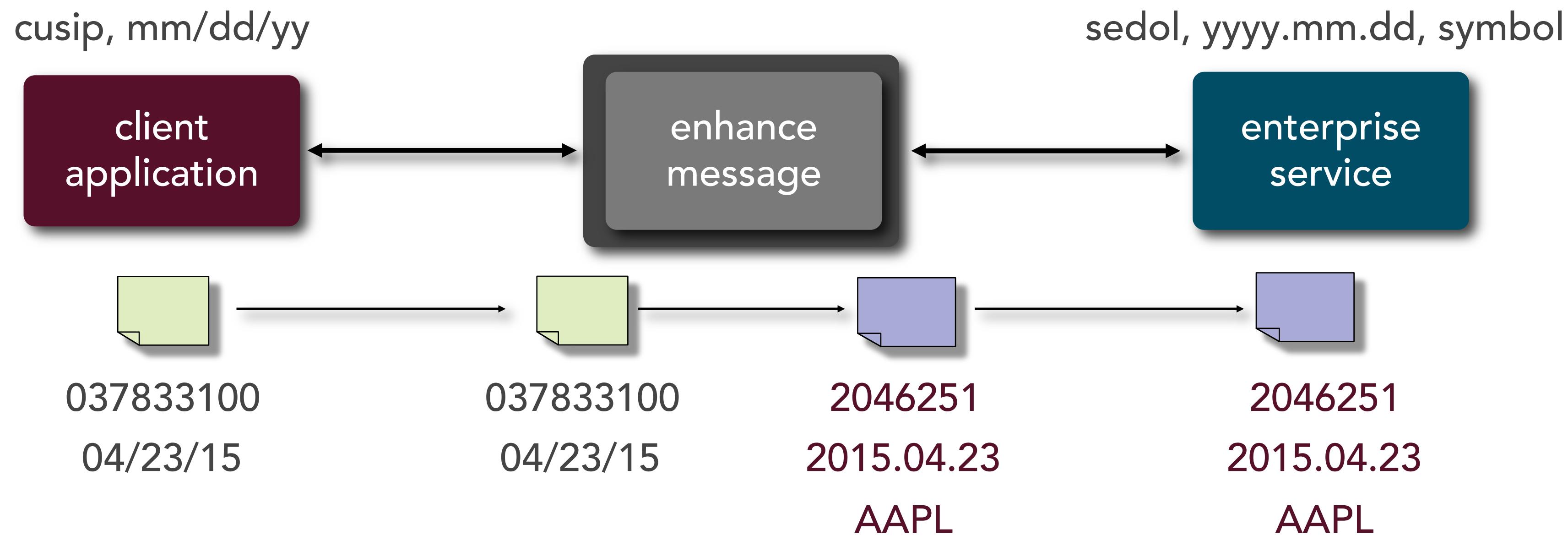
service-oriented architecture

message enhancement



service-oriented architecture

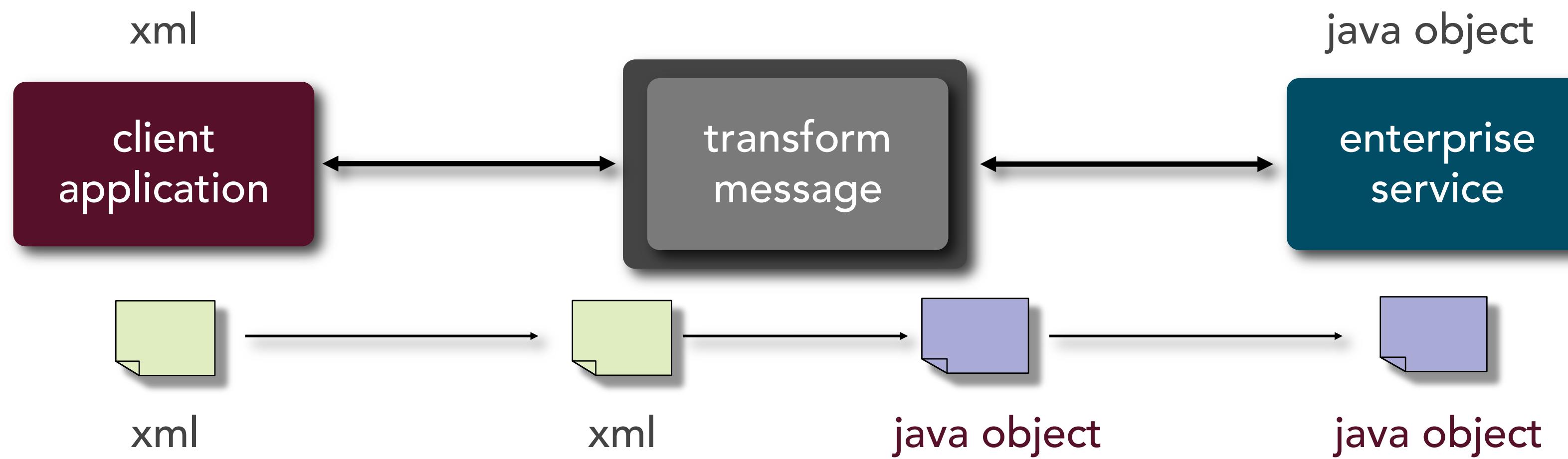
message enhancement



contract decoupling

service-oriented architecture

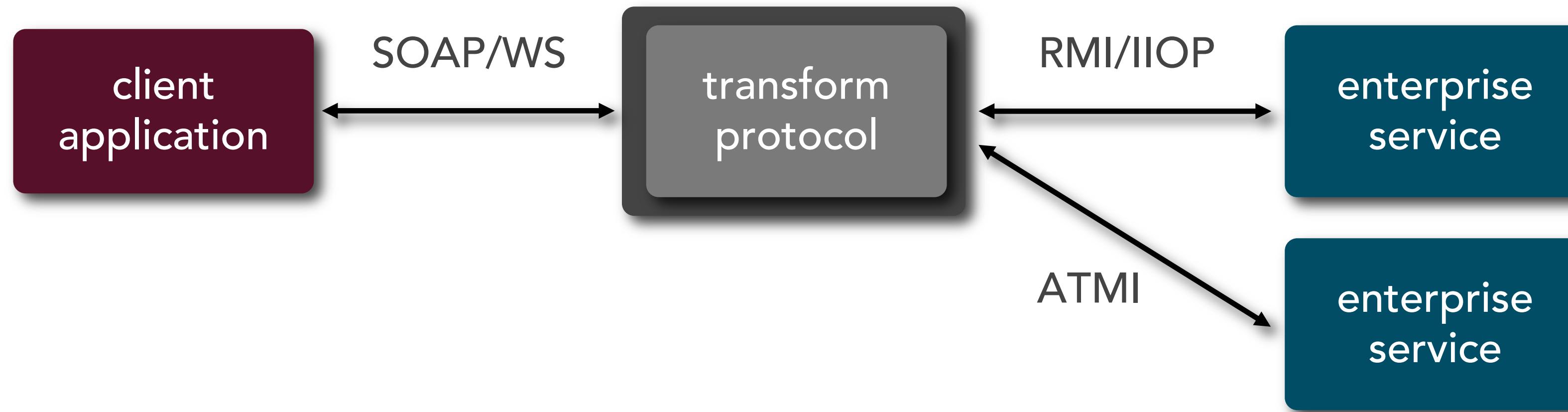
message transformation



contract decoupling

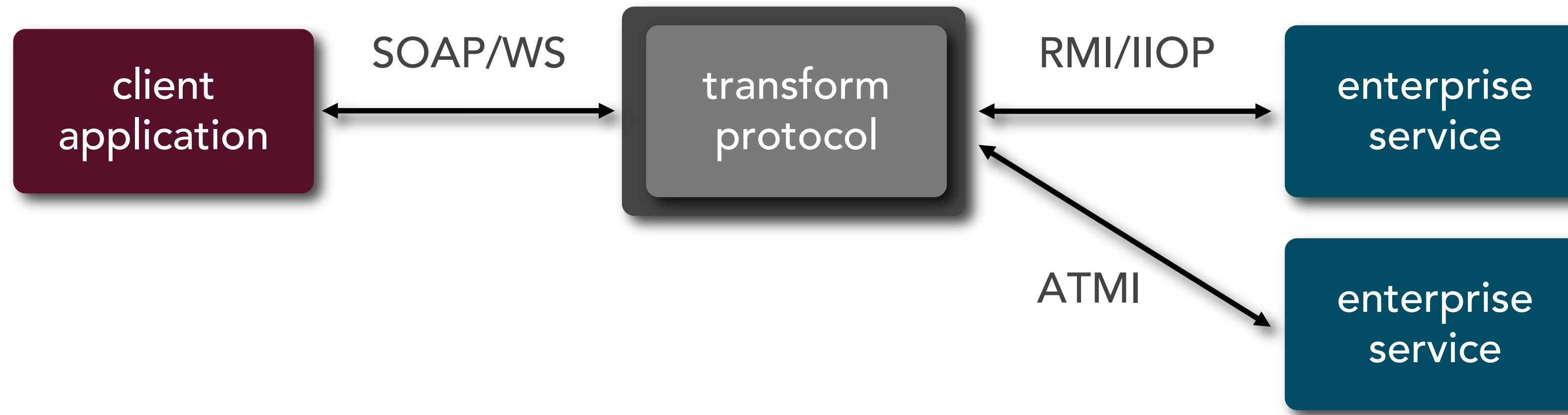
service-oriented architecture

protocol transformation



service-oriented architecture

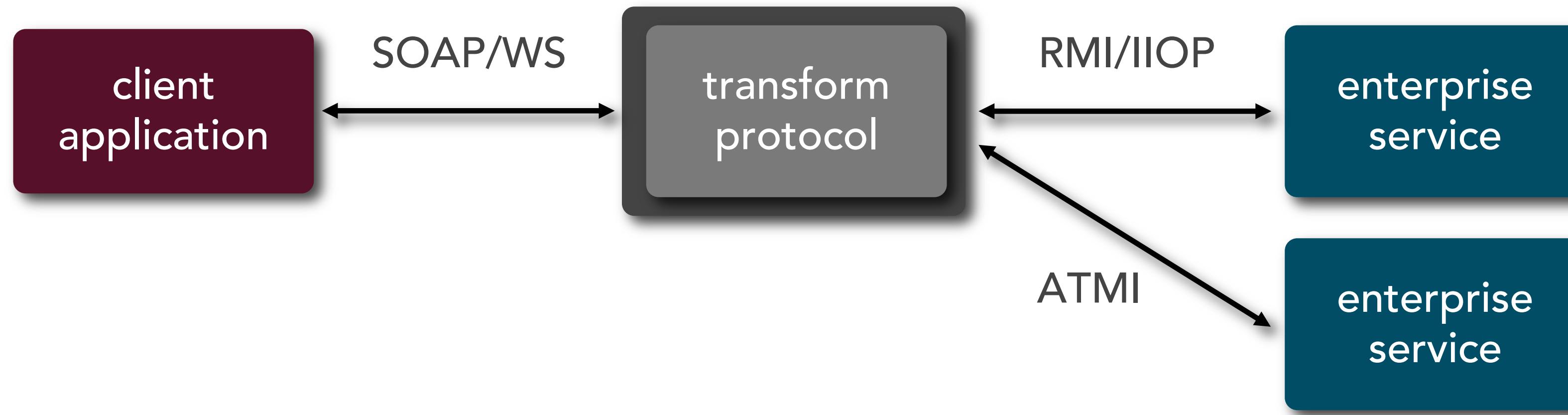
protocol transformation



access decoupling

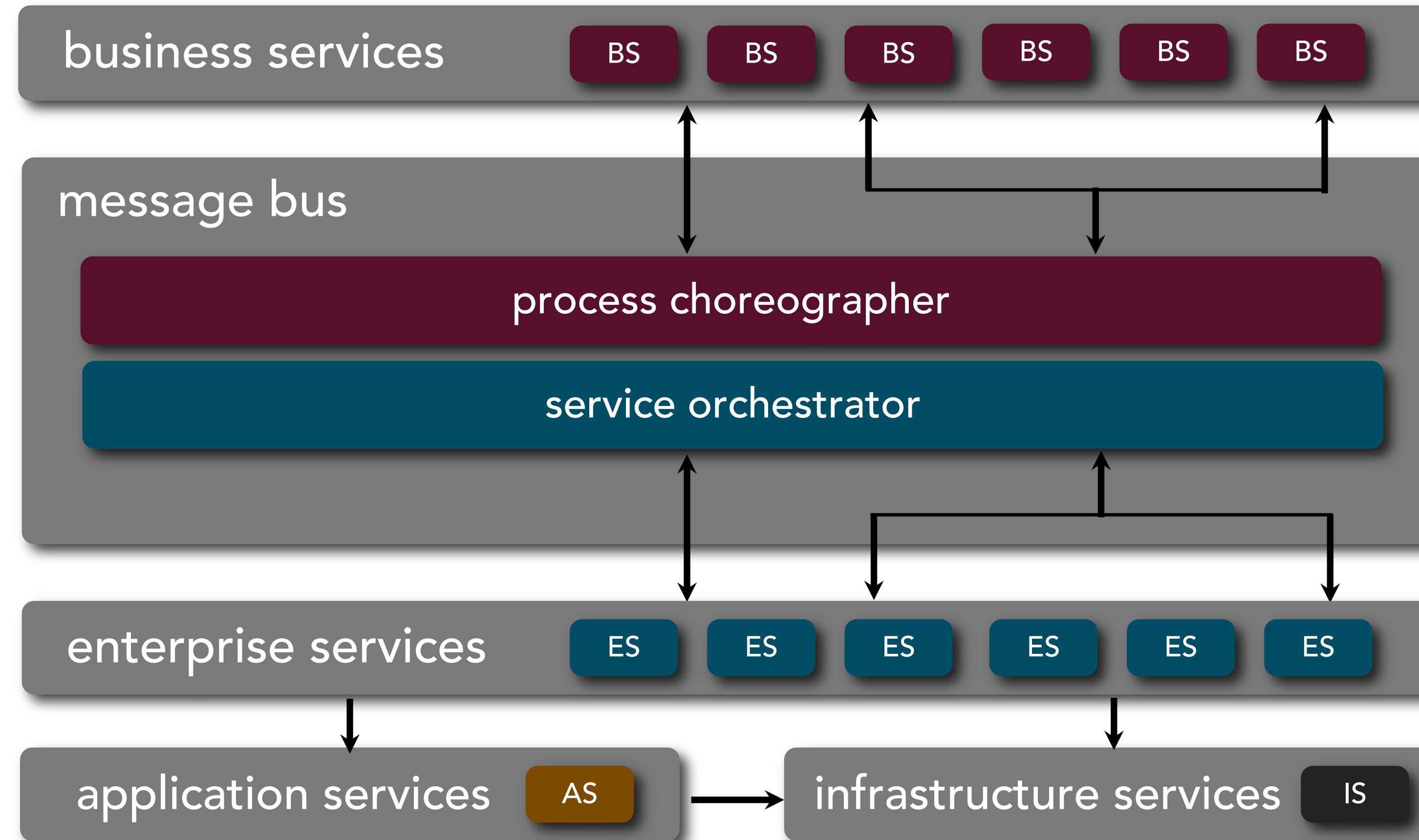
service-oriented architecture

protocol transformation

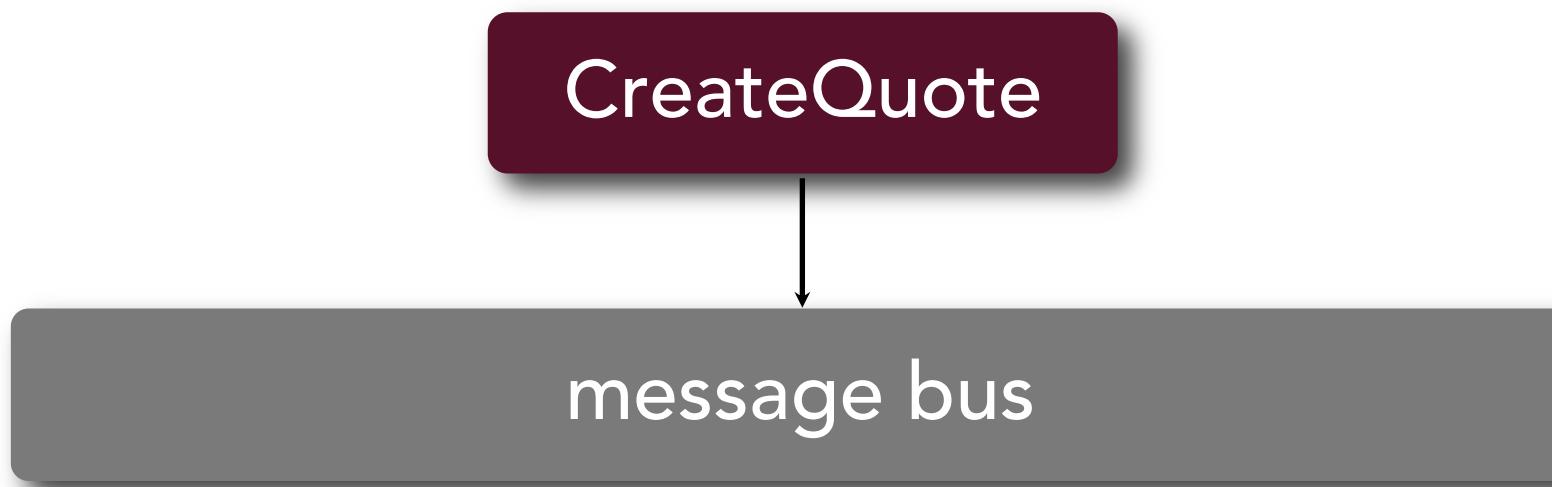


access decoupling
implementation transparency

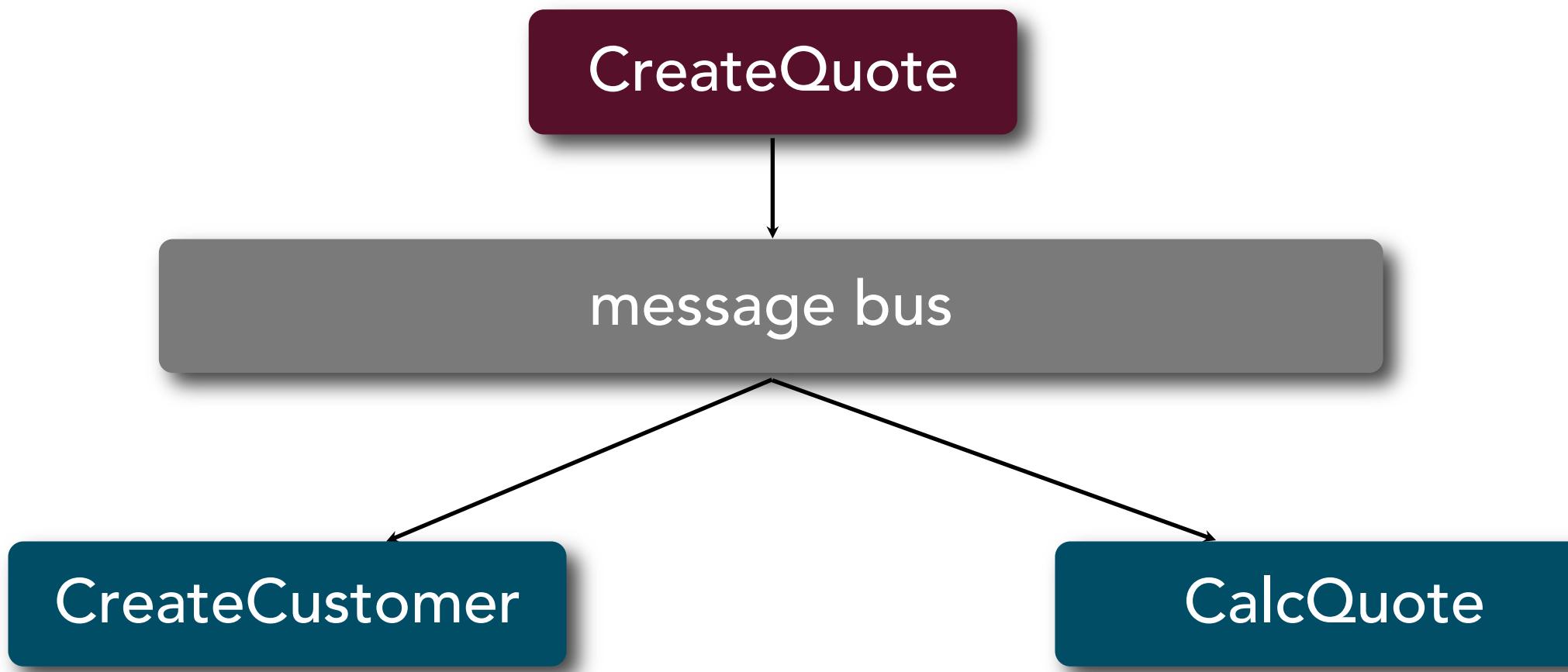
service-oriented architecture



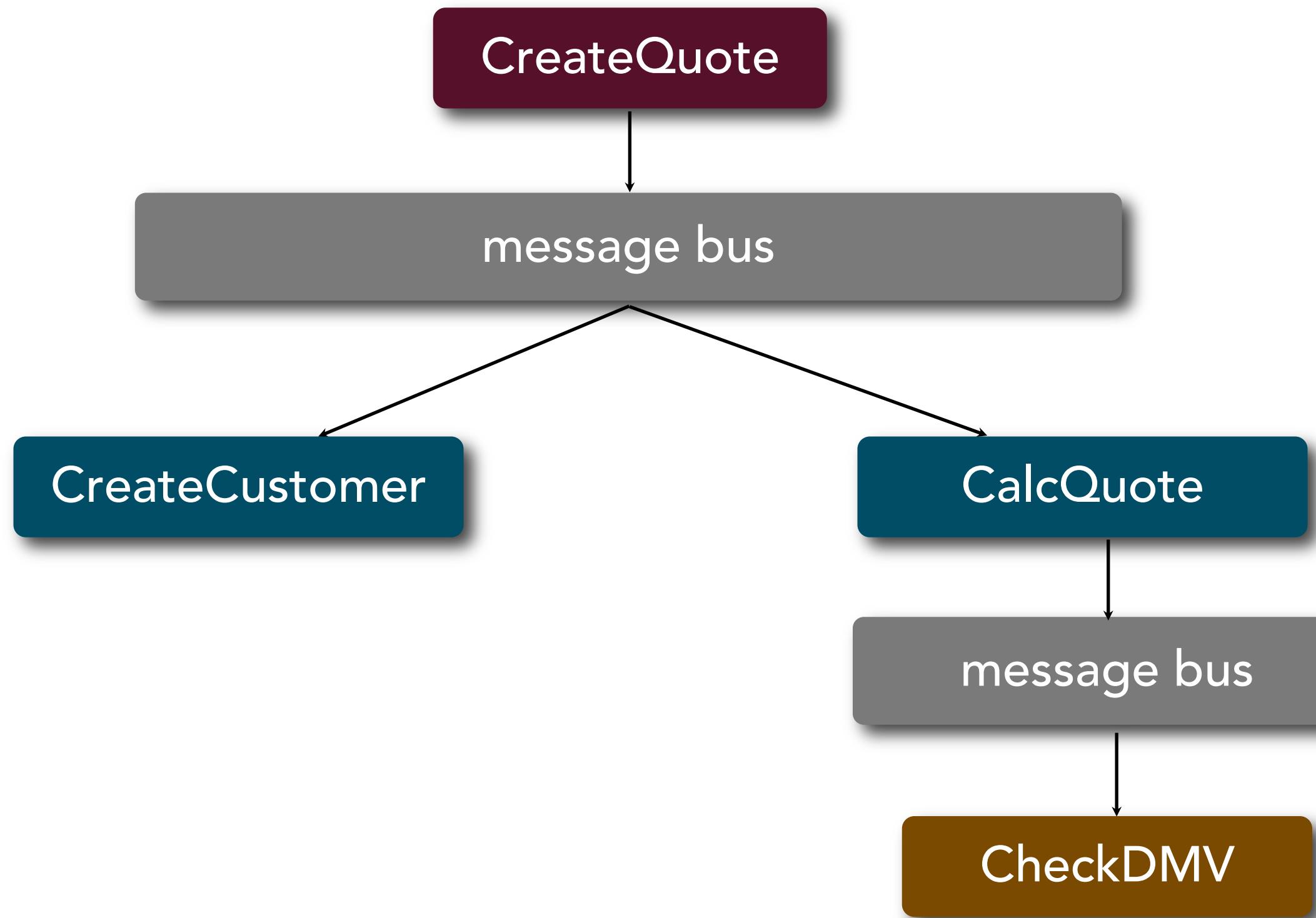
service-oriented architecture



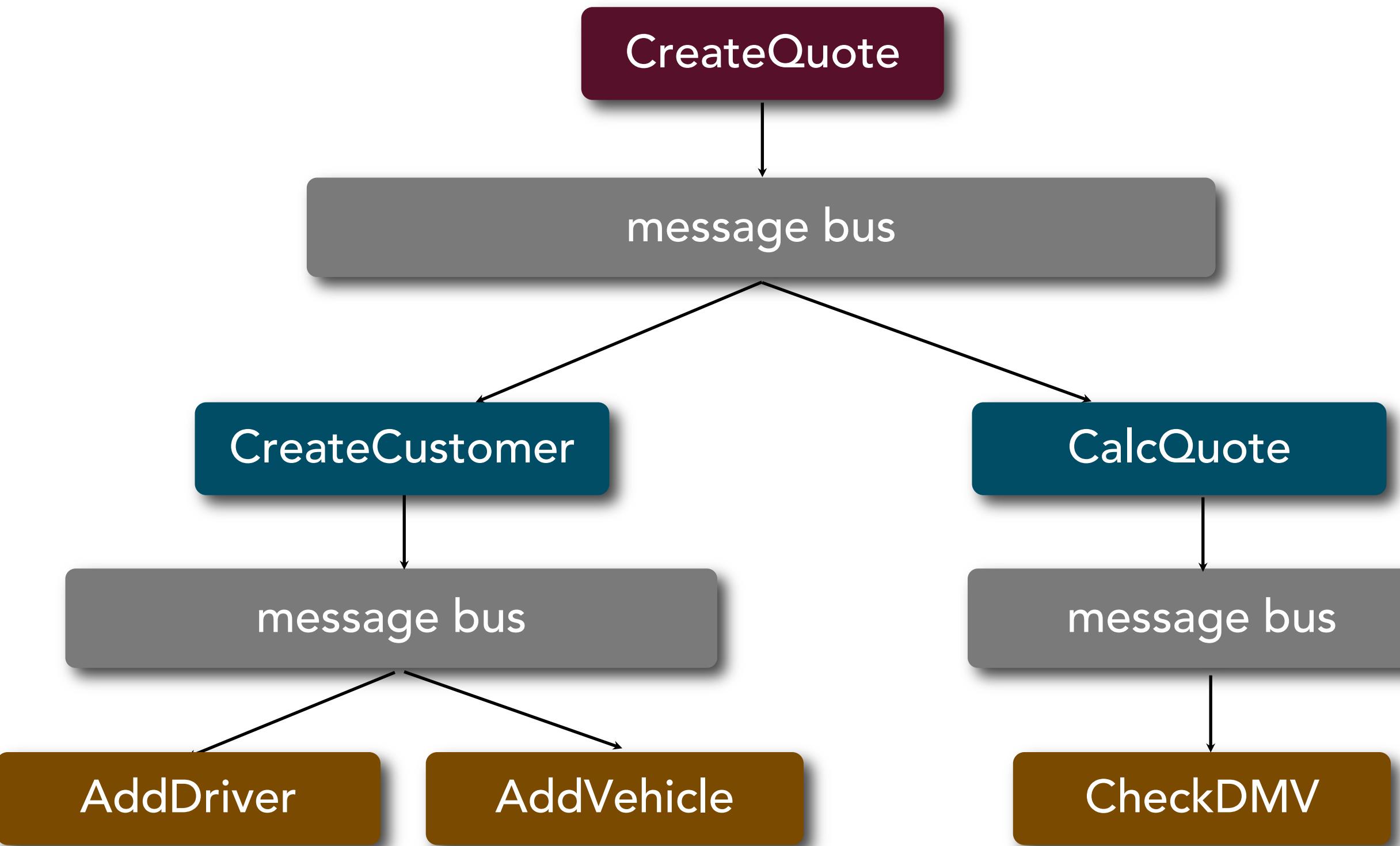
service-oriented architecture



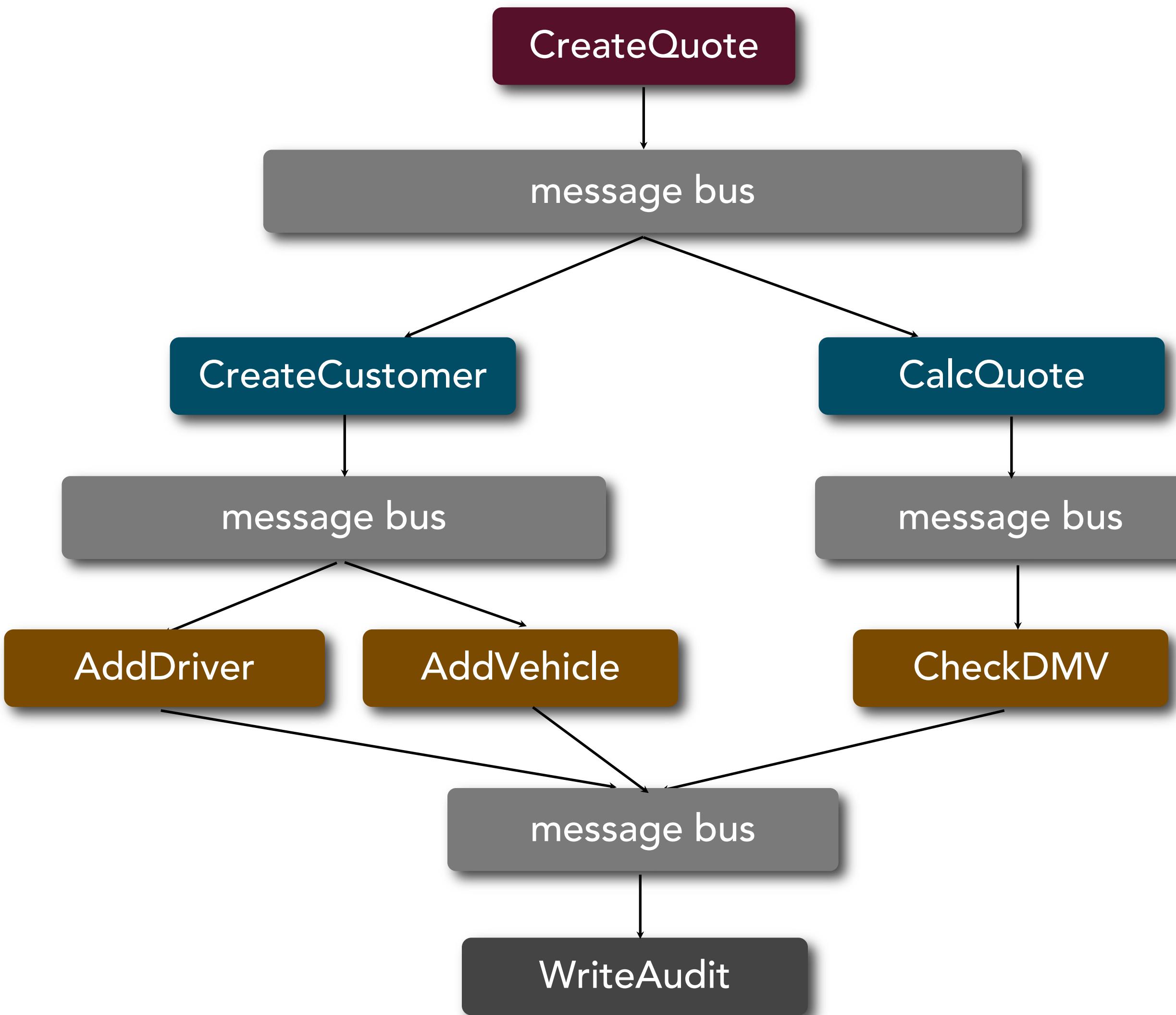
service-oriented architecture



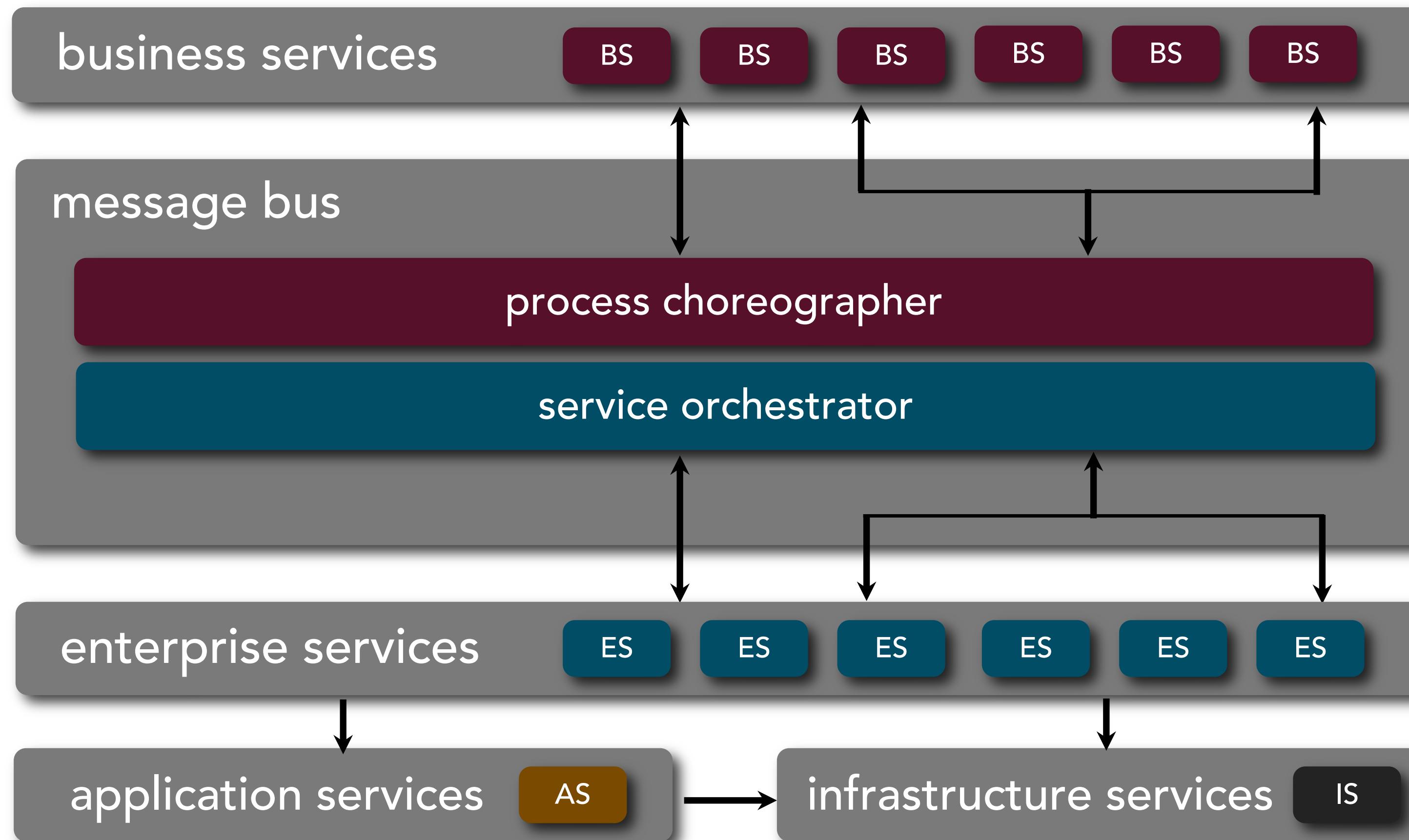
service-oriented architecture



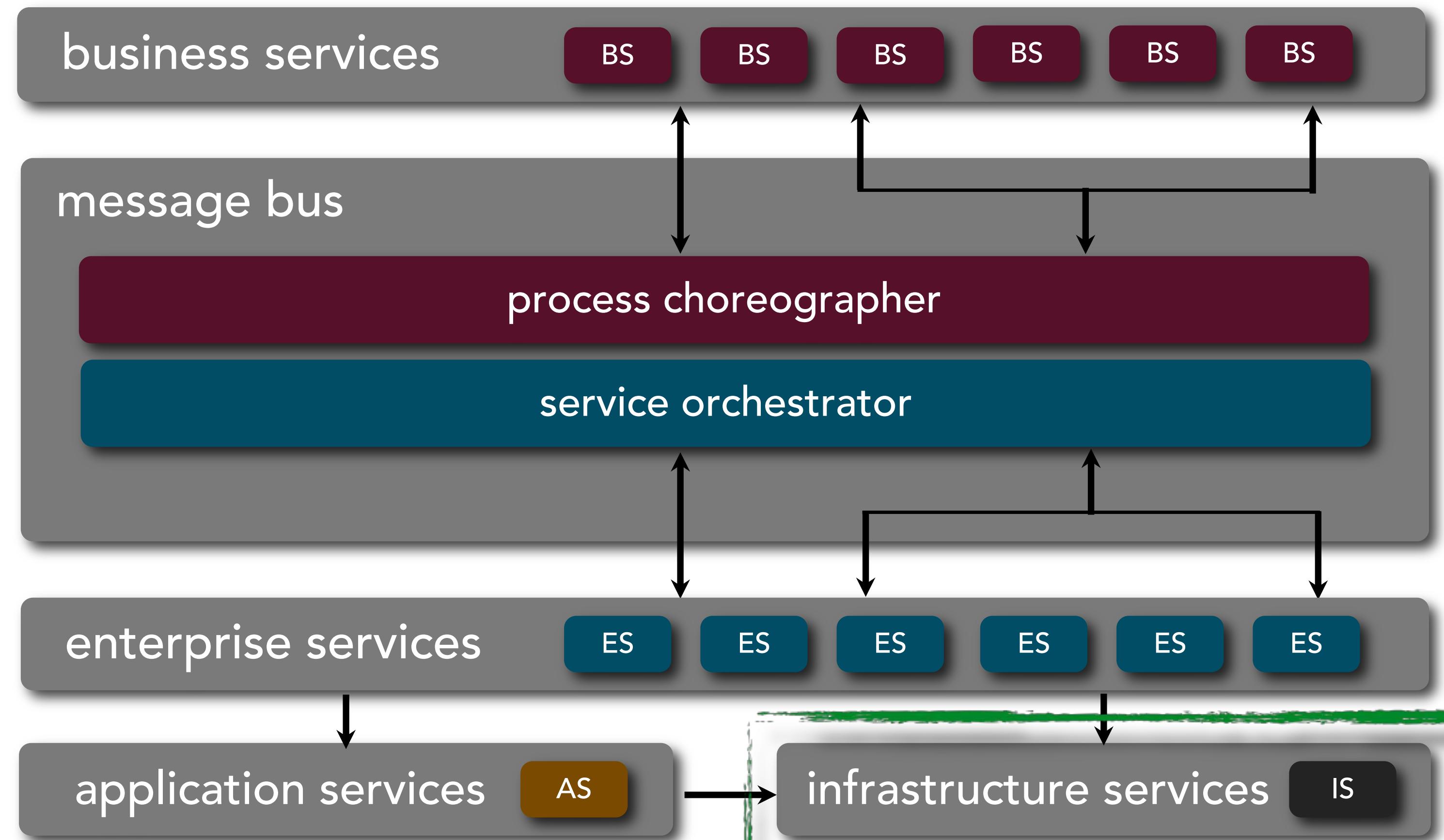
service-oriented architecture



service-oriented architecture

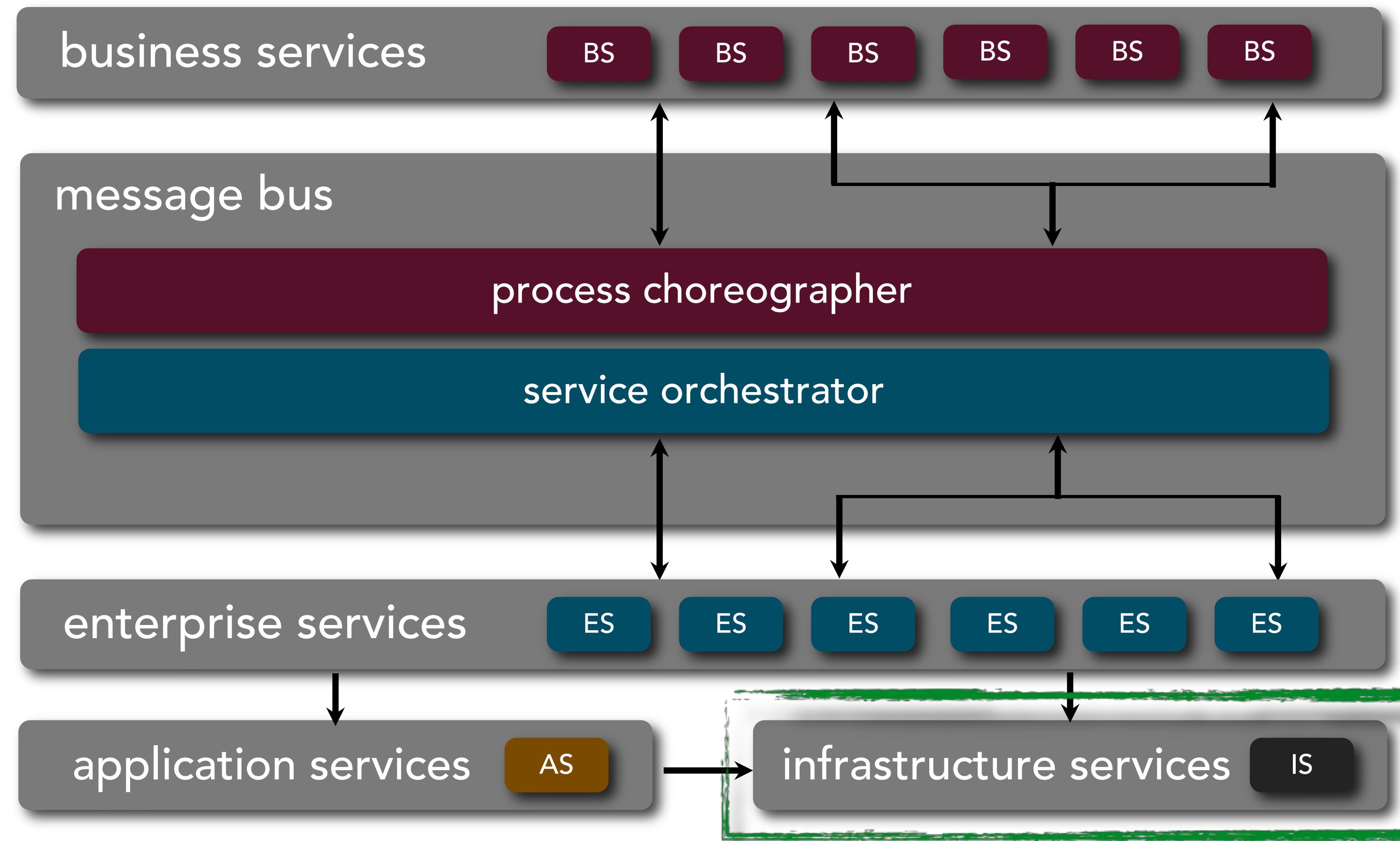


service-oriented architecture



maximize canonicity

service-oriented architecture



maximize canonicity



maximize reuse

auto and
homeowners
insurance
division

commercial
insurance
division

casualty
insurance
division

life
insurance
division

disability
insurance
division

travel
insurance
division

auto and
homeowners
insurance
division

customer

commercial
insurance
division

customer

casualty
insurance
division

customer

life
insurance
division

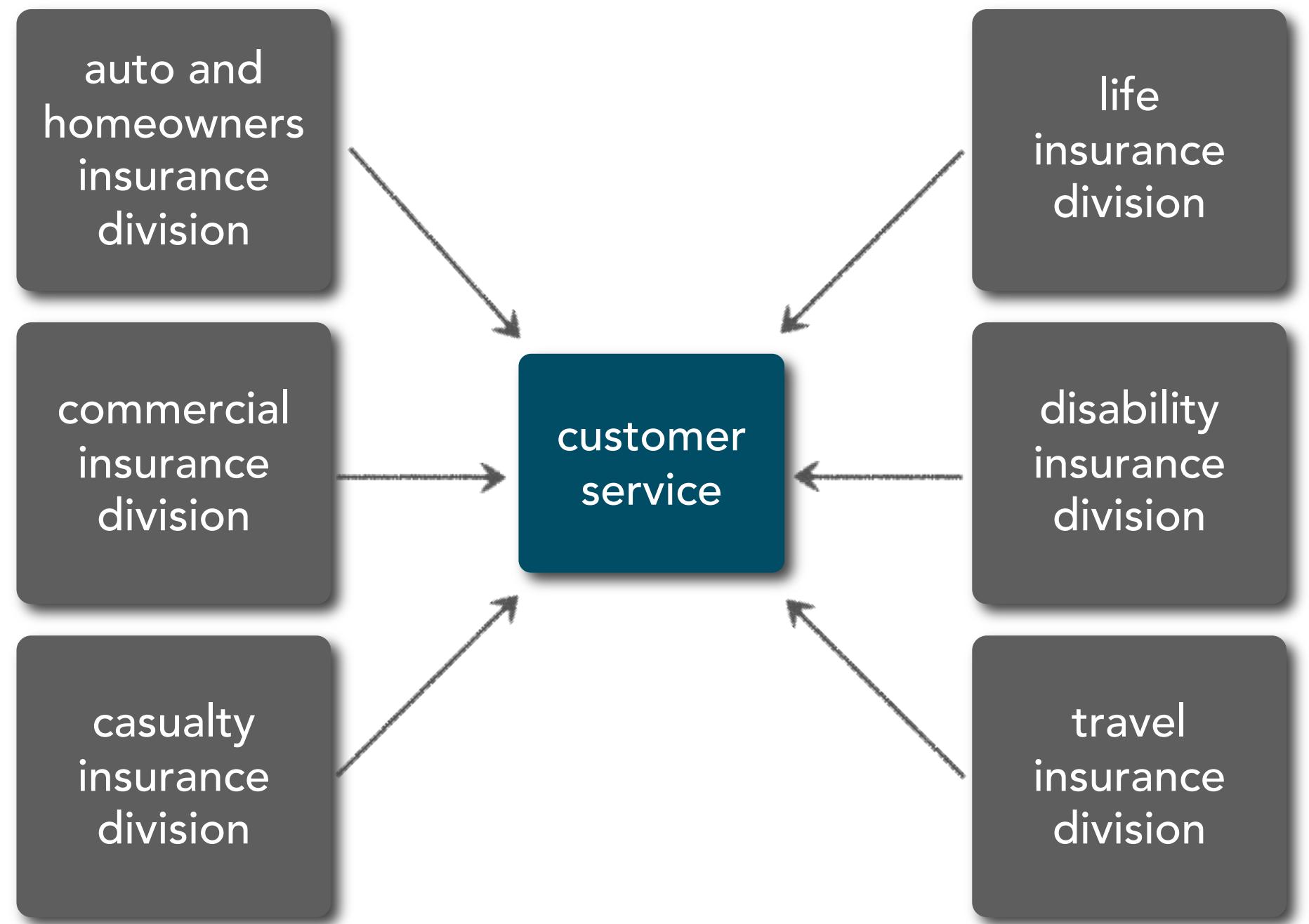
customer

disability
insurance
division

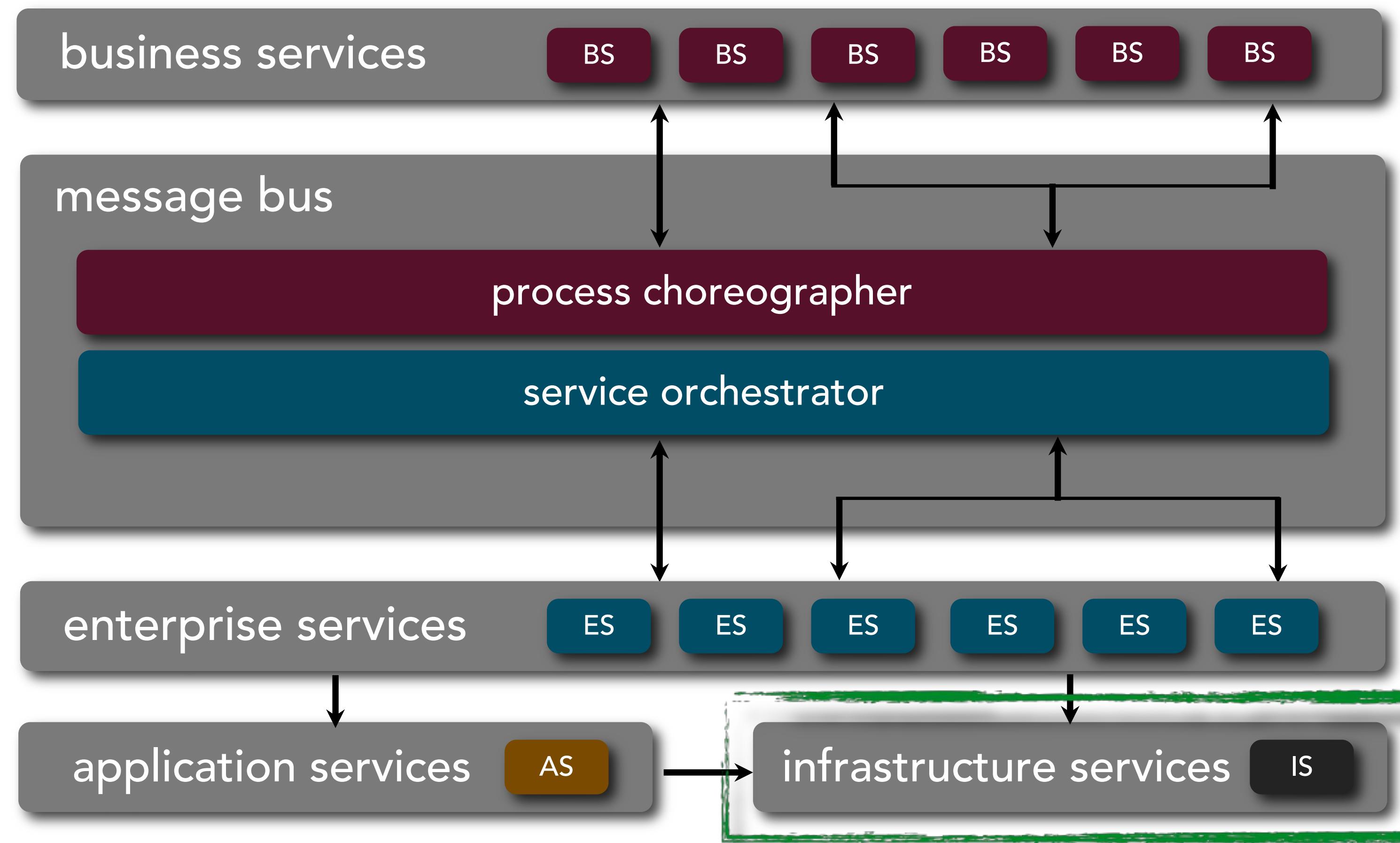
customer

travel
insurance
division

customer



service-oriented architecture

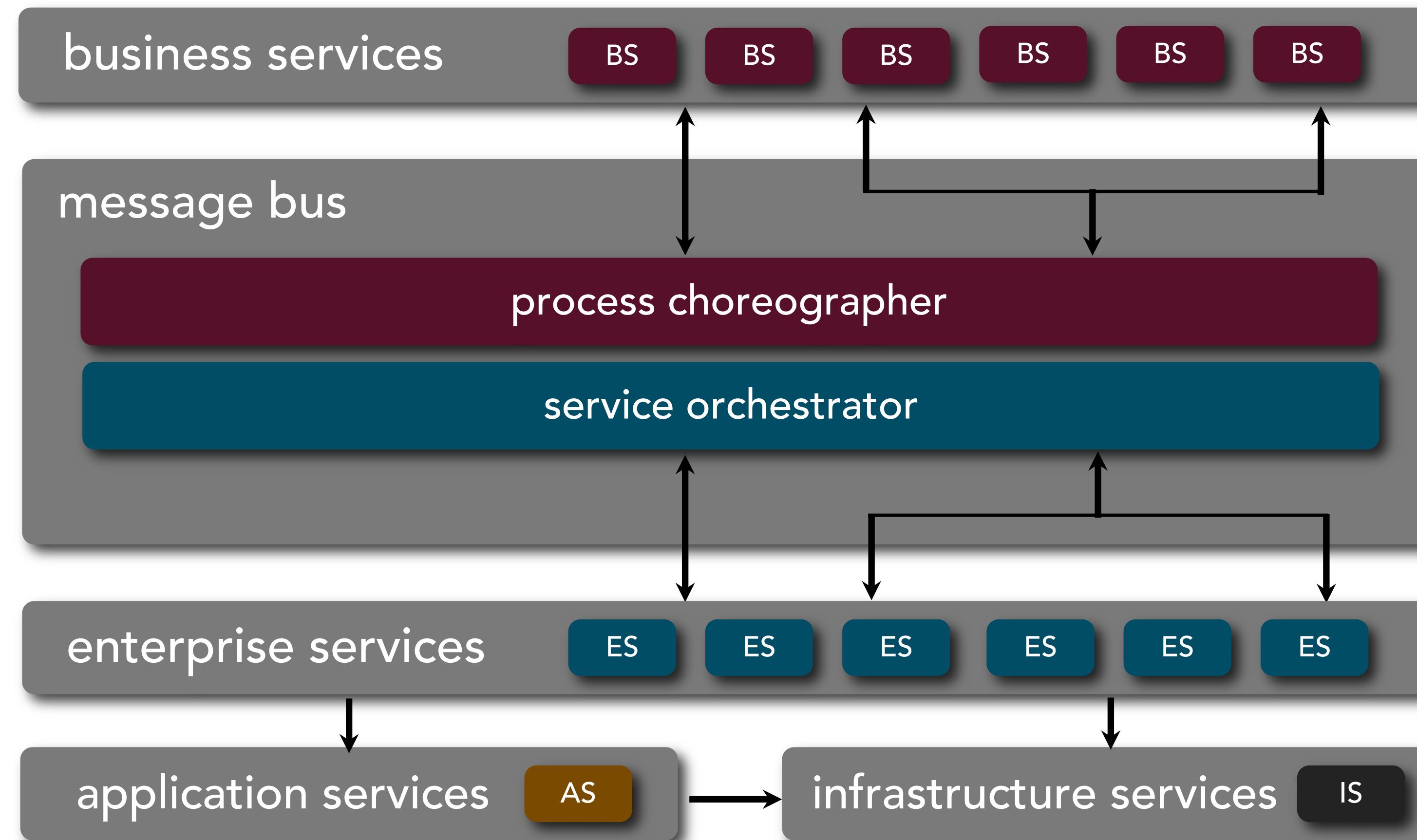


maximize canonicity

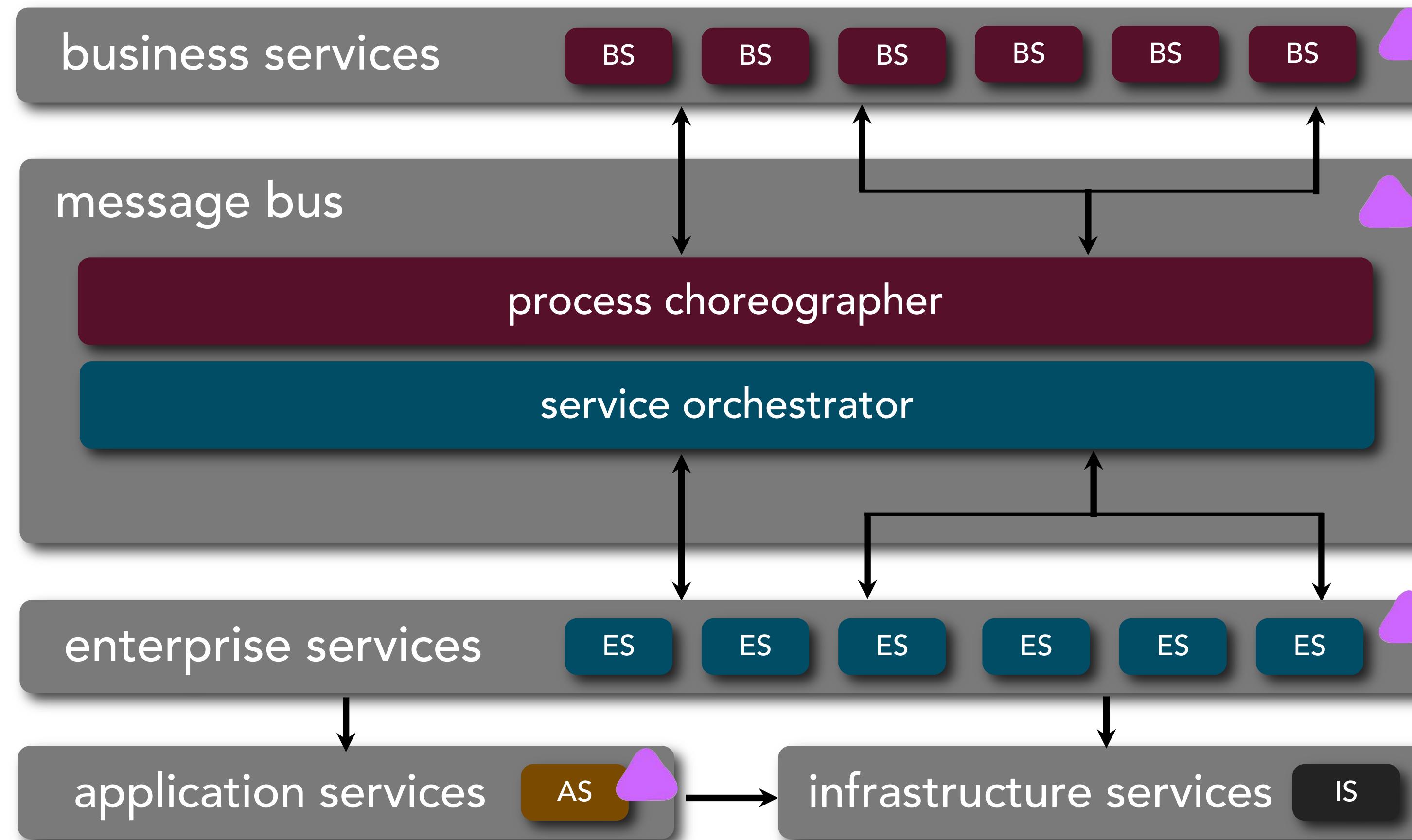


maximize reuse

service-oriented architecture



service-oriented architecture

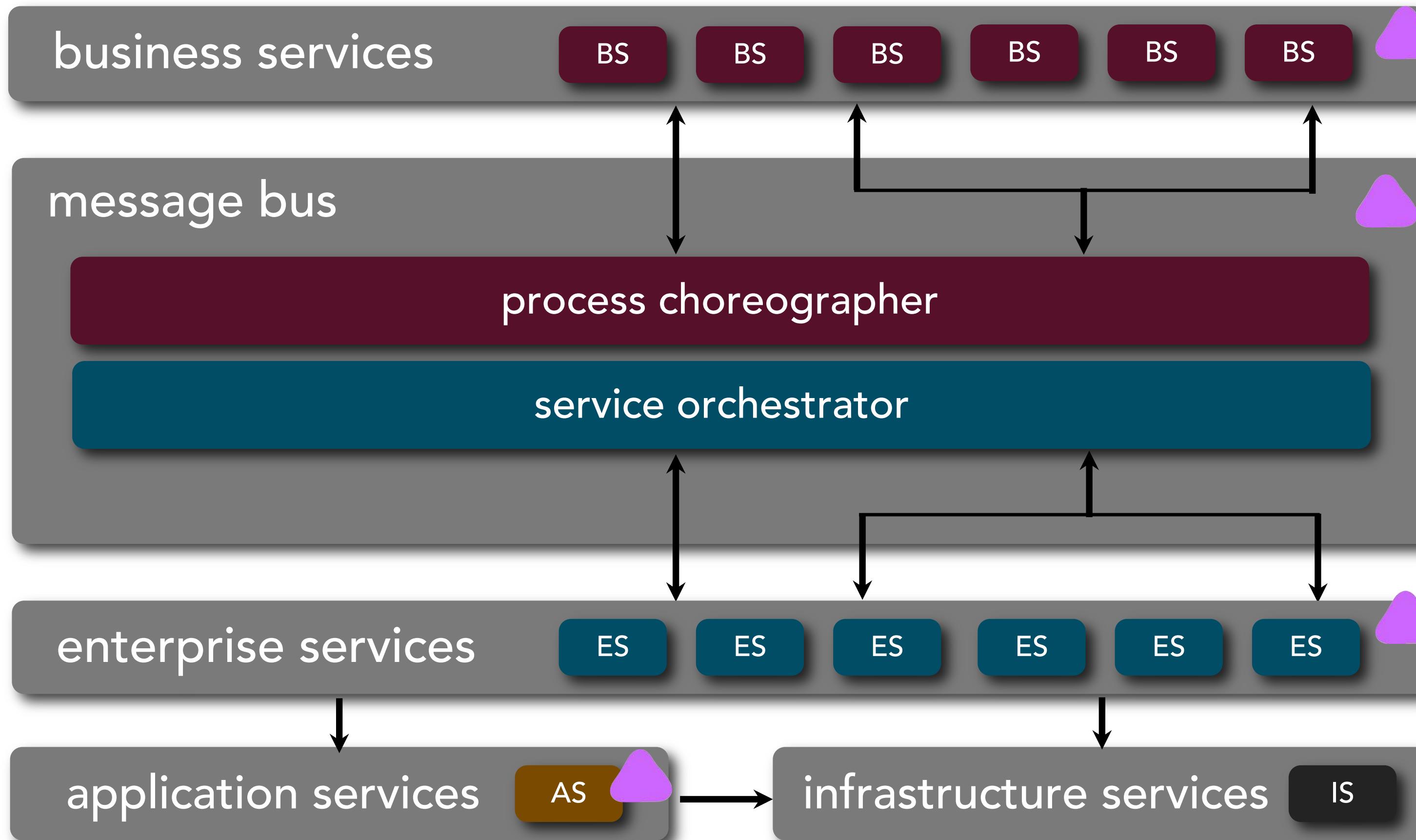


operationally coupled



incremental change

service-oriented architecture



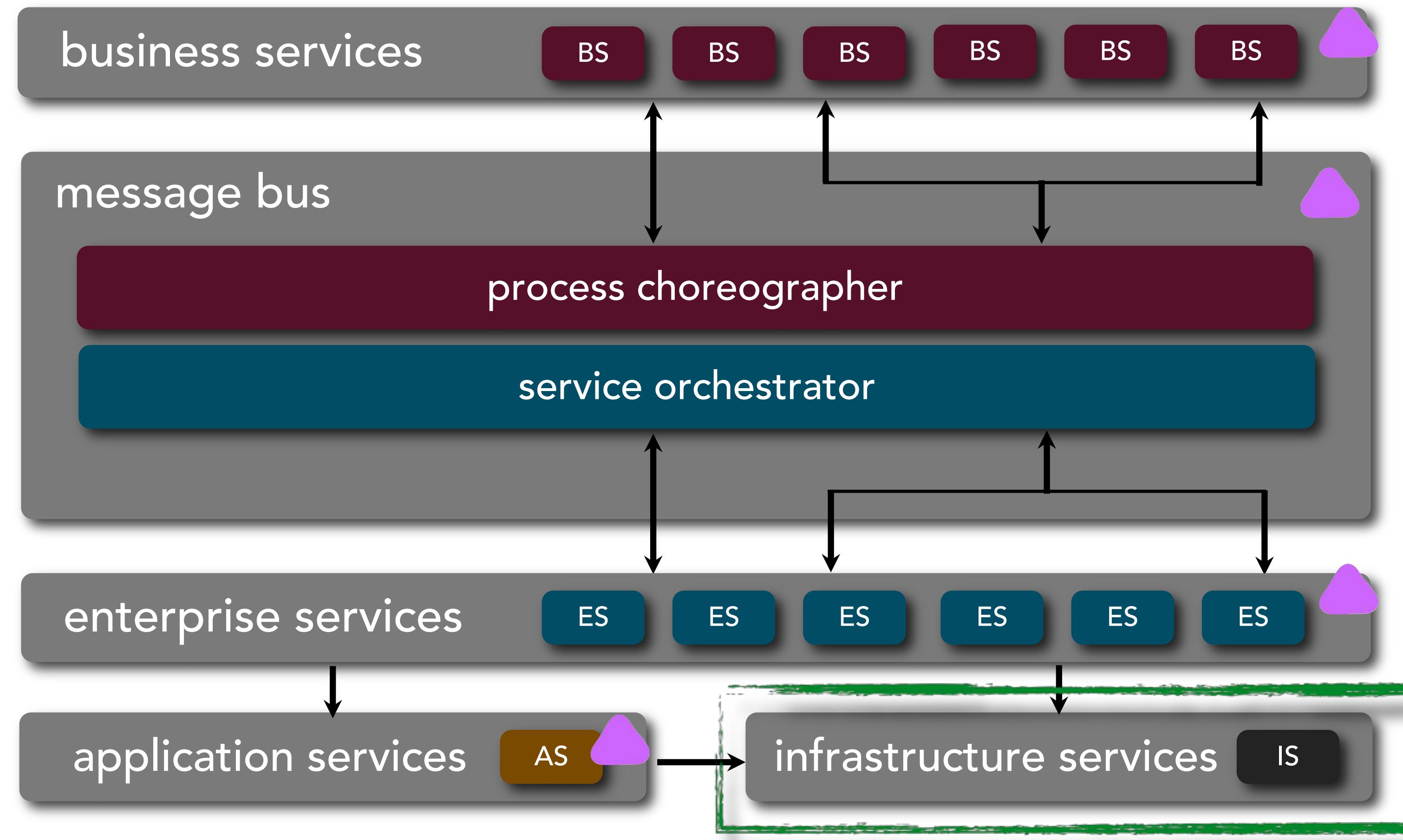
operationally coupled



incremental change

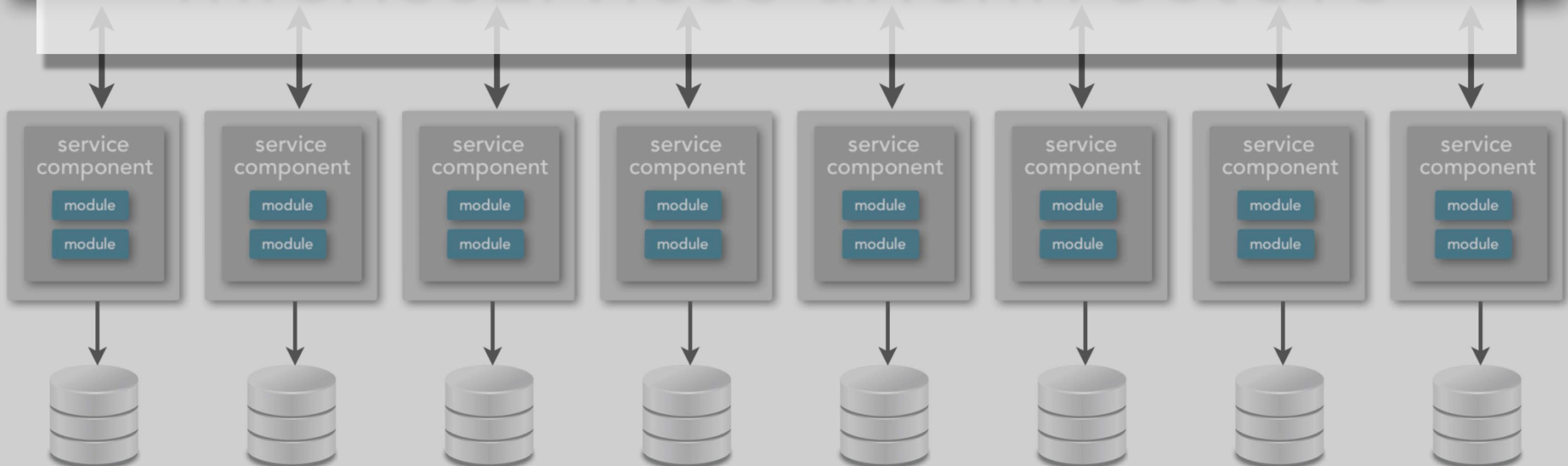
technical
partitioning

service-oriented architecture



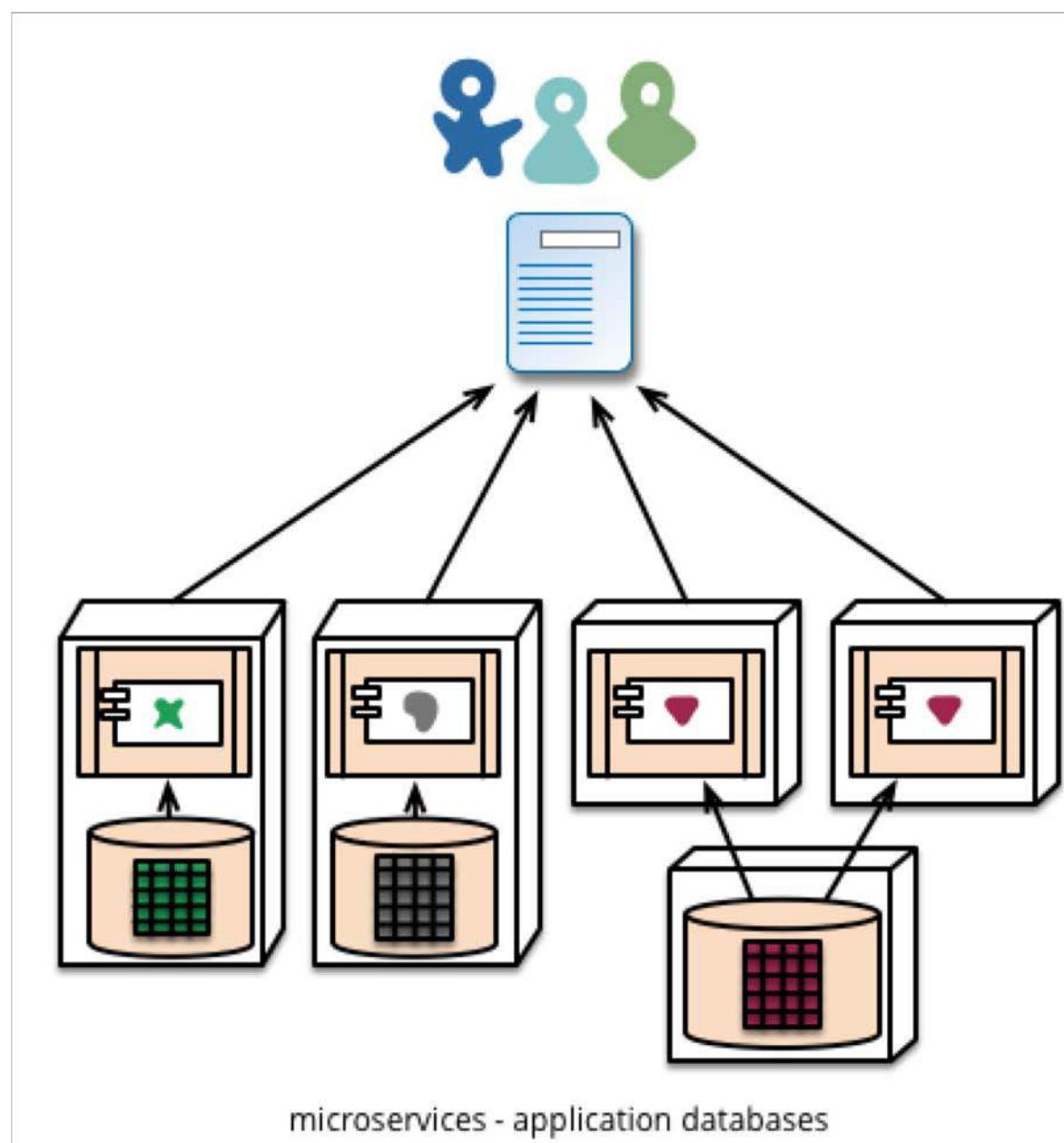
miCRoSrvlceS aRChiTeCtUre

api layer



microservices

<https://martinfowler.com/articles/microservices.html>



microservices - application databases

MARTINFOWLER.COM

Contents

Microservices

a definition of this new architectural term

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014

 **James Lewis**
James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of systems using microservices and has been an active participant in the growing community for a couple of years.

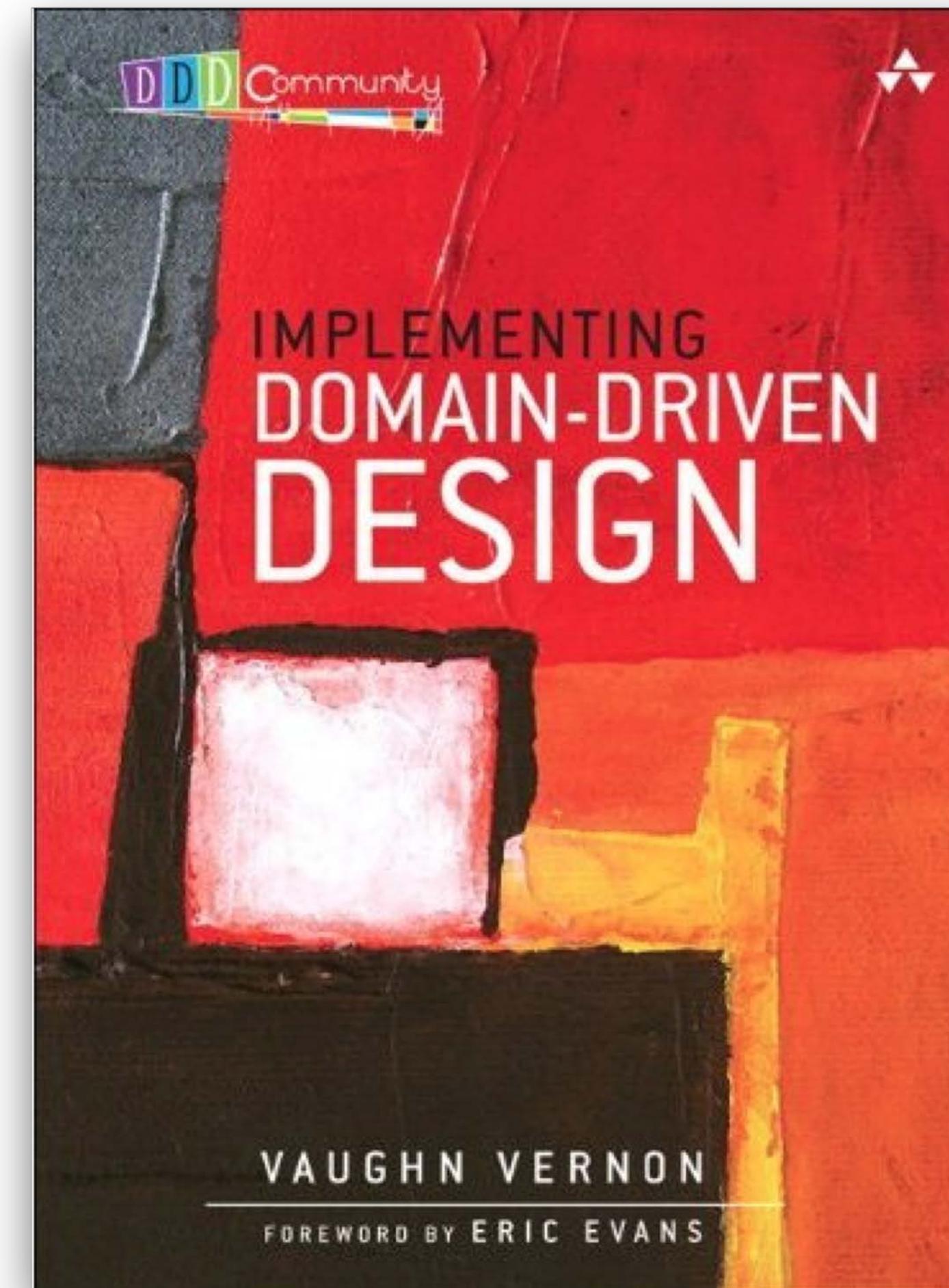
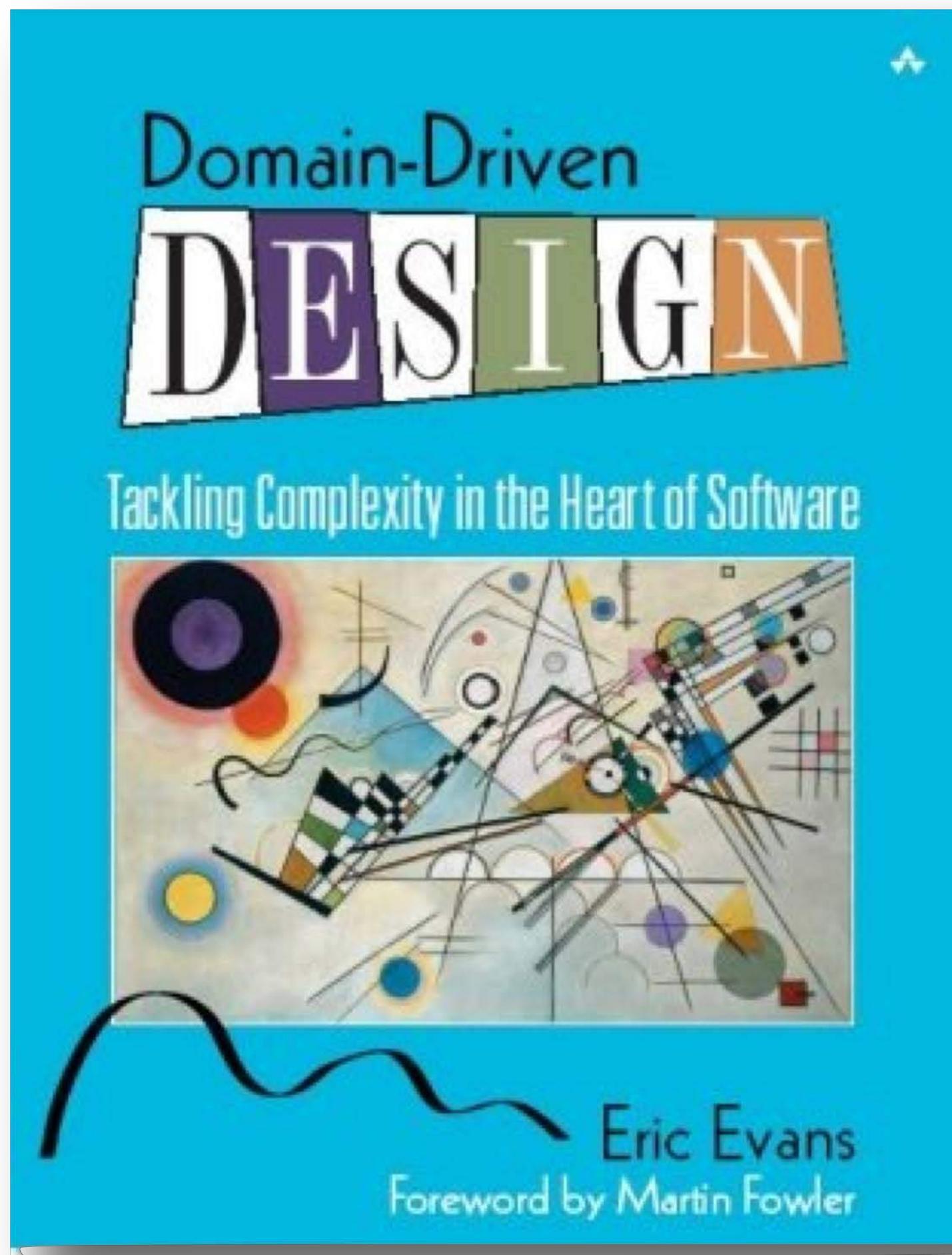
 **Martin Fowler**
Martin Fowler is an author, speaker, and general loud-mouth on software development. He's long been puzzled by the problem of how to componentize software systems, having heard more vague claims than he's happy with. He hopes that microservices will live up to the early promise its advocates have found.

Translations: [Japanese](#) · [Russian](#) · [Korean](#) · [Portuguese](#) · [Simplified Chinese](#) · [Simplified Chinese](#) · [Persian](#)

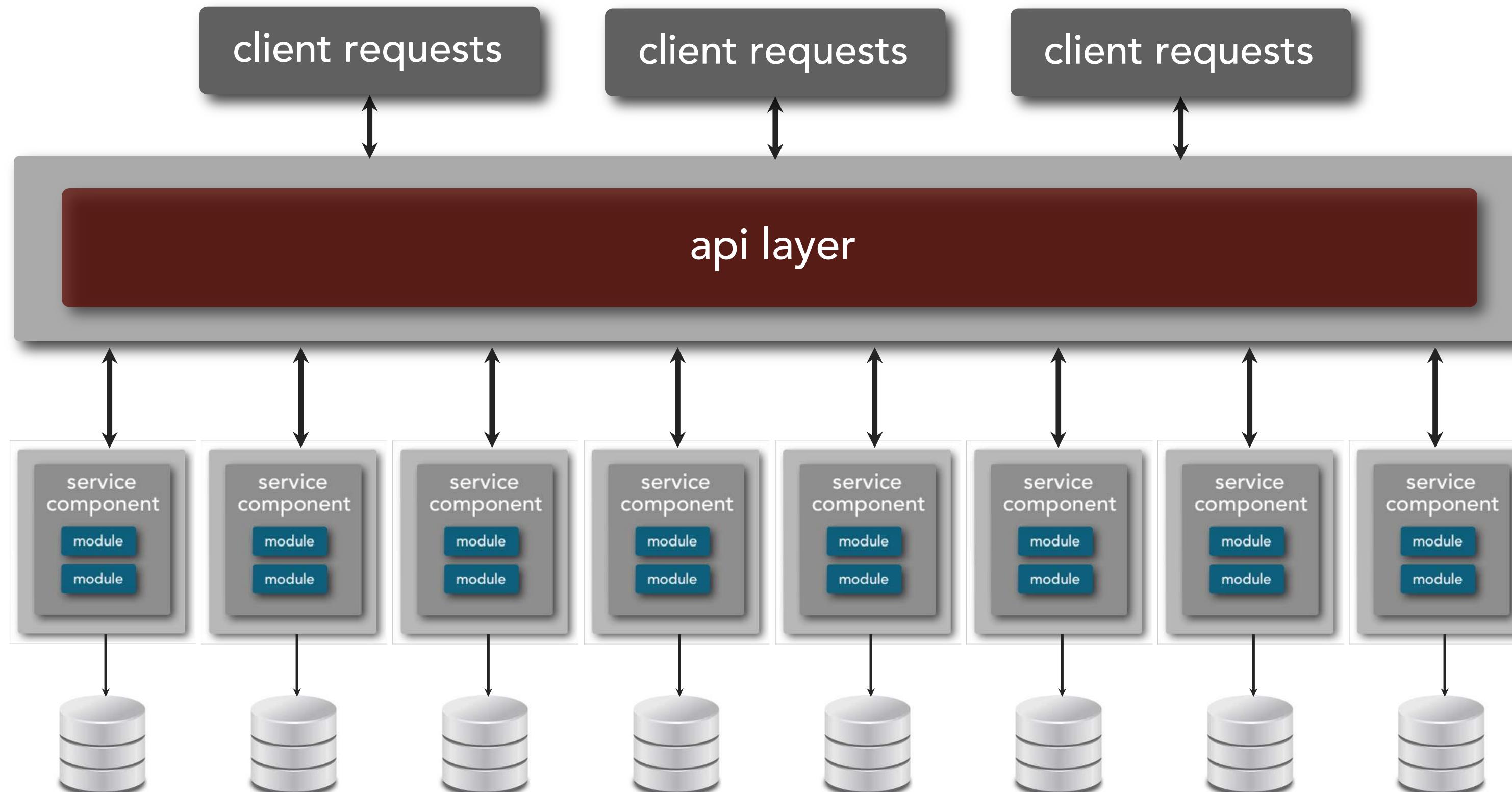
Find **similar articles** to this by looking at these tags: [popular](#) · [application architecture](#) · [web services](#) · [microservices](#)

"Microservices" - yet another new term on the crowded streets of software architecture. Although our natural inclination is to pass such things by with a contemptuous glance, this bit of terminology describes a style of software systems that we are finding more and more appealing. We've seen many projects use this style in the last few years, and results so far have been positive, so much so that

domain driven design

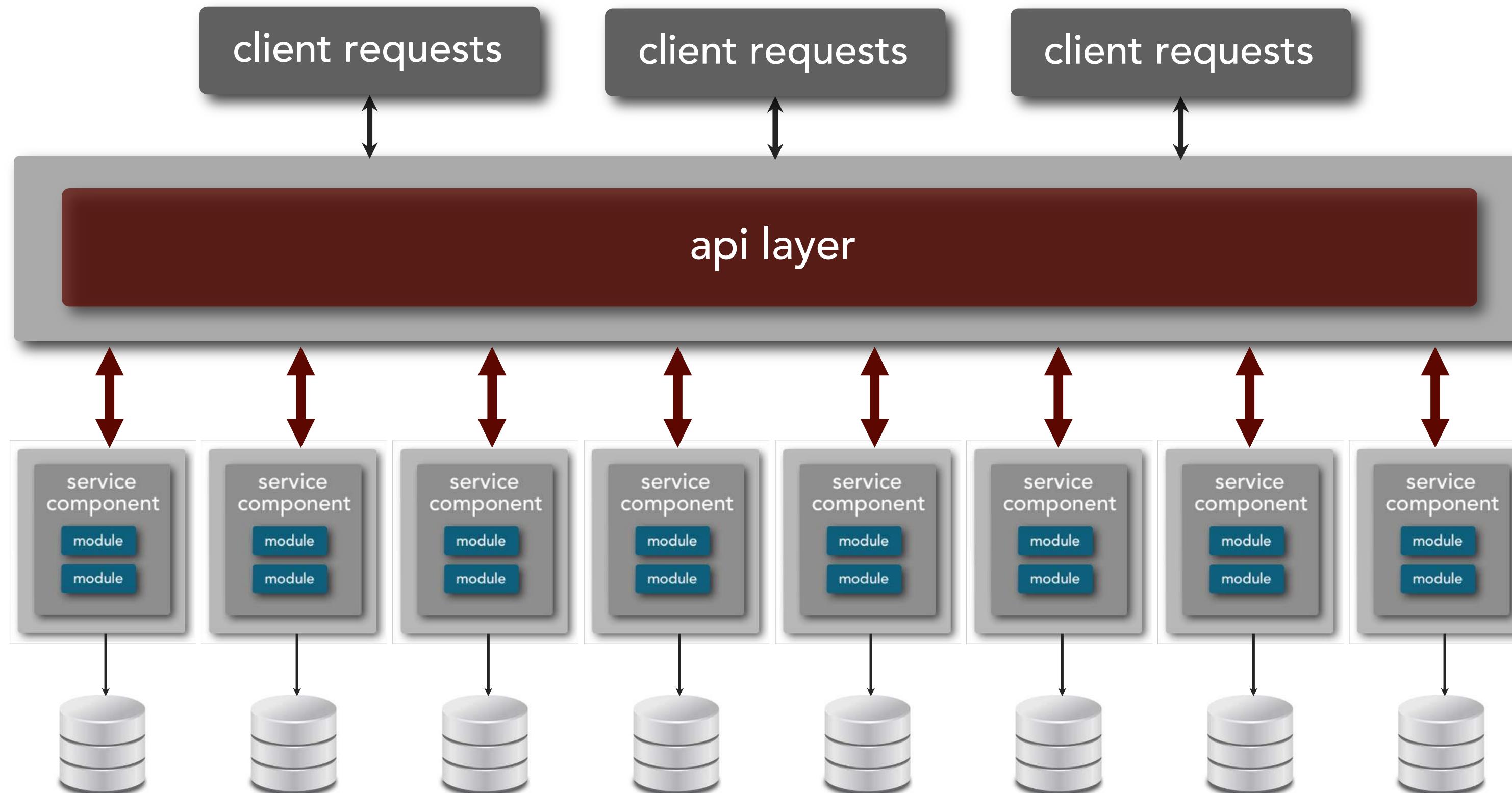


microservices architecture



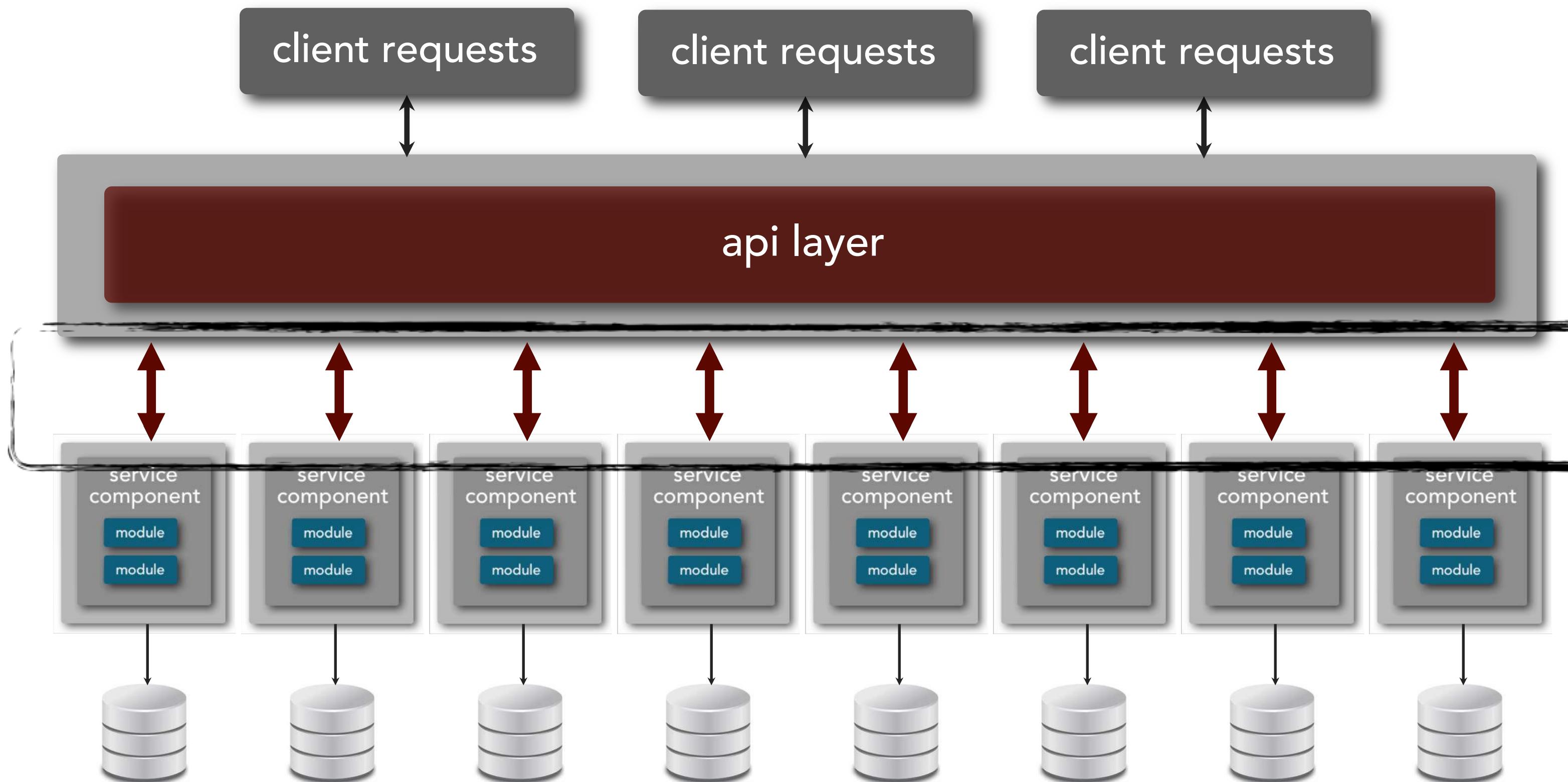
microservices architecture

distributed architecture



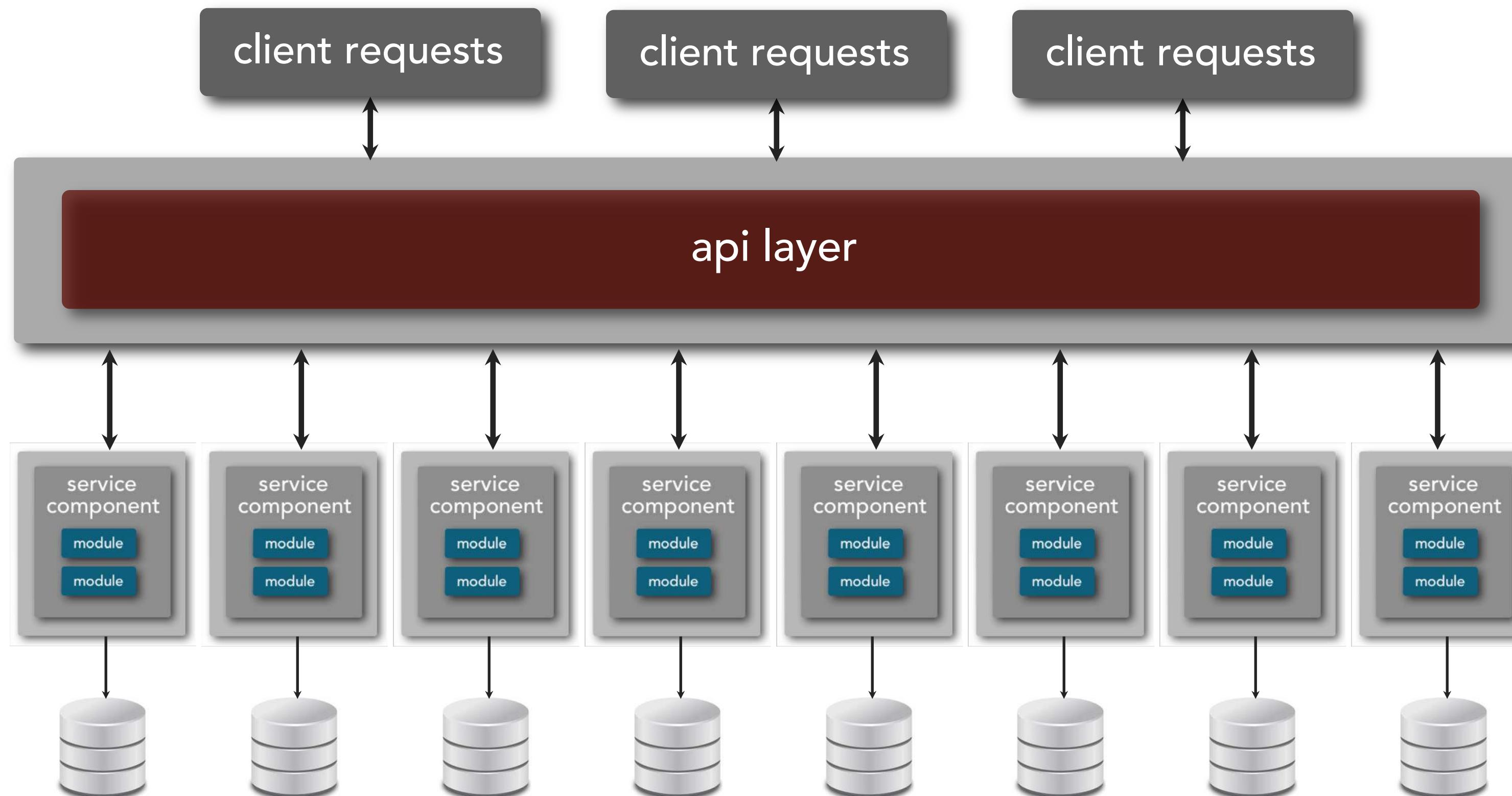
microservices architecture

distributed architecture



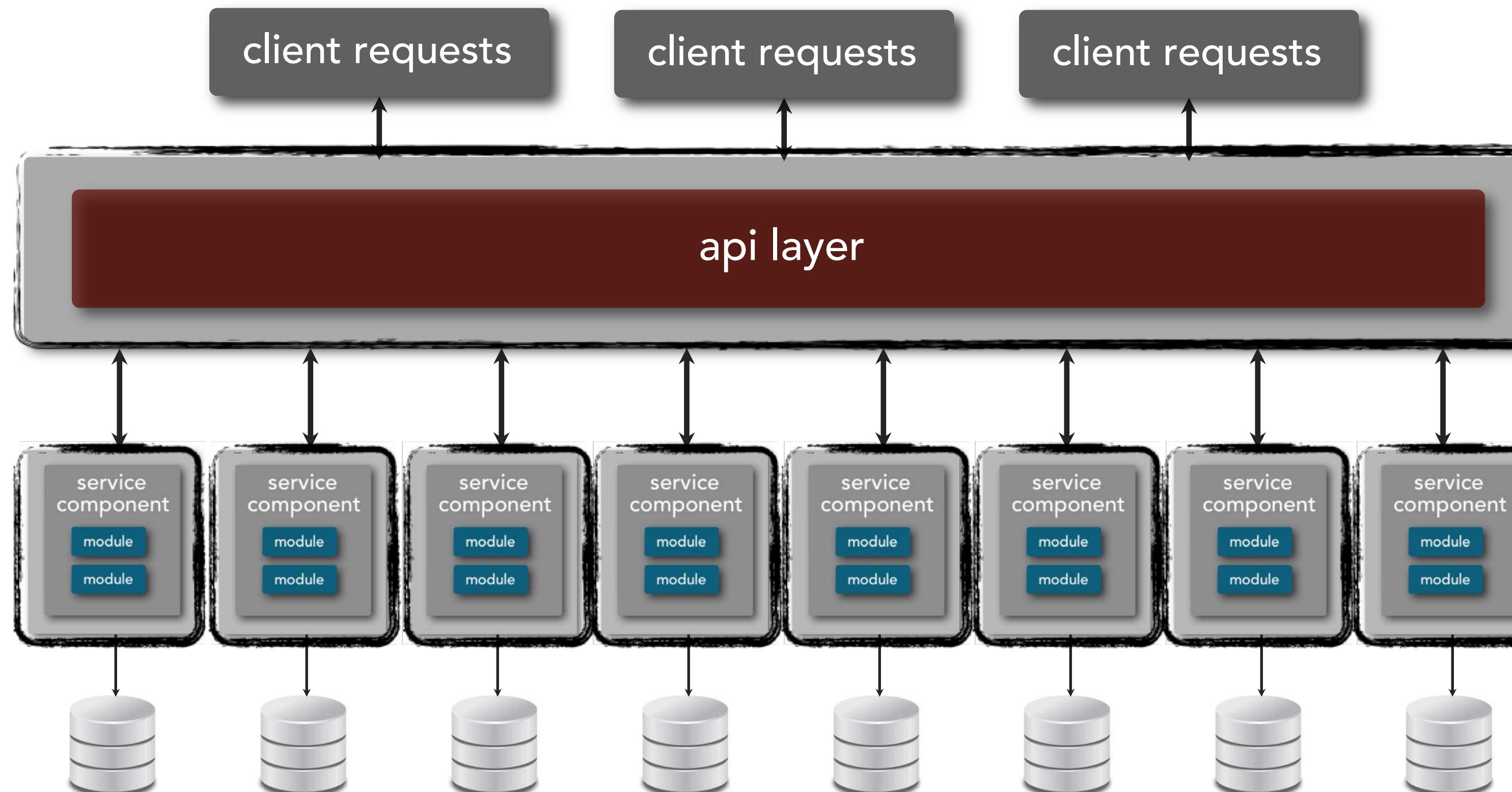
microservices architecture

separately deployed components



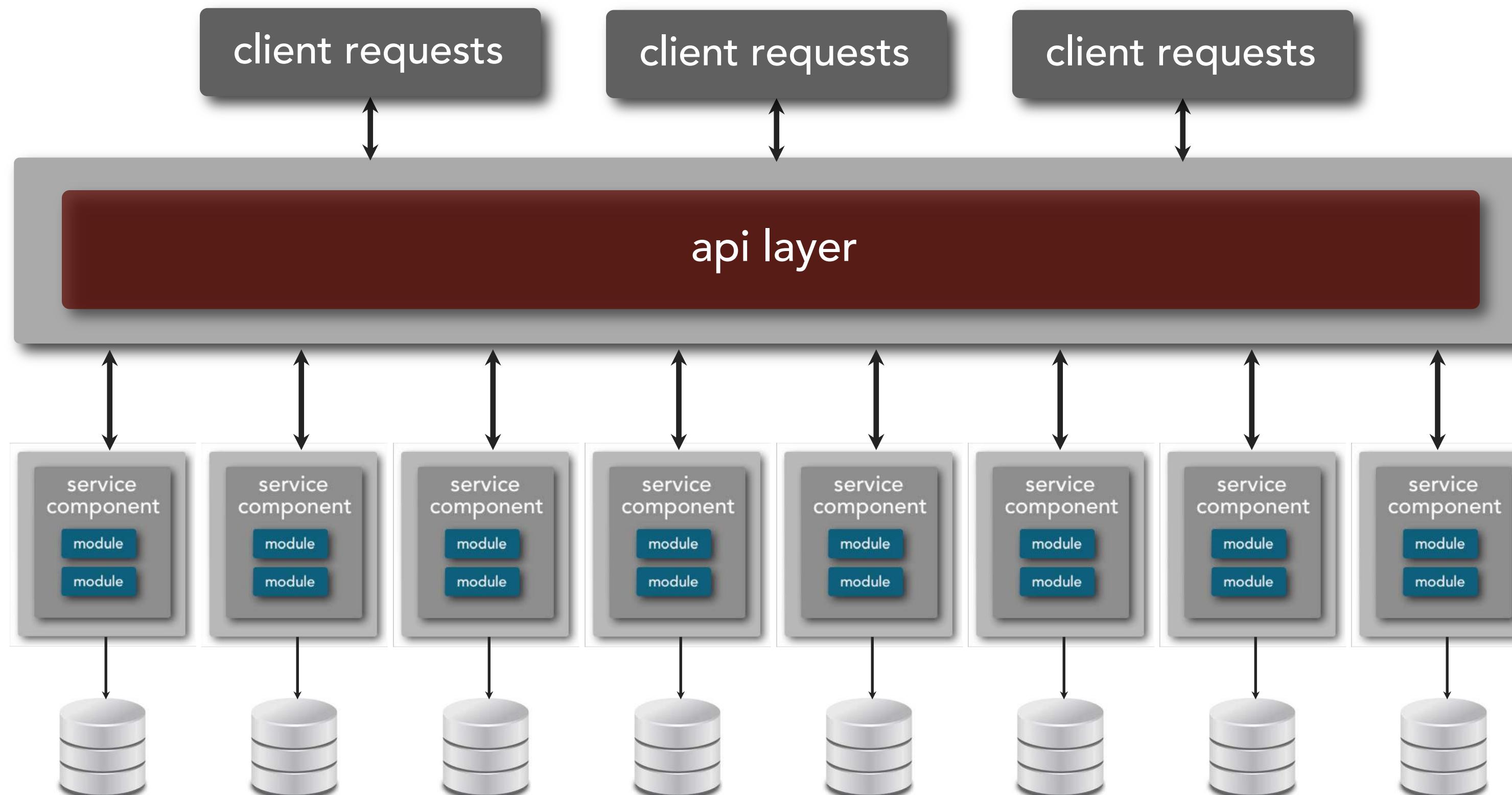
microservices architecture

separately deployed components



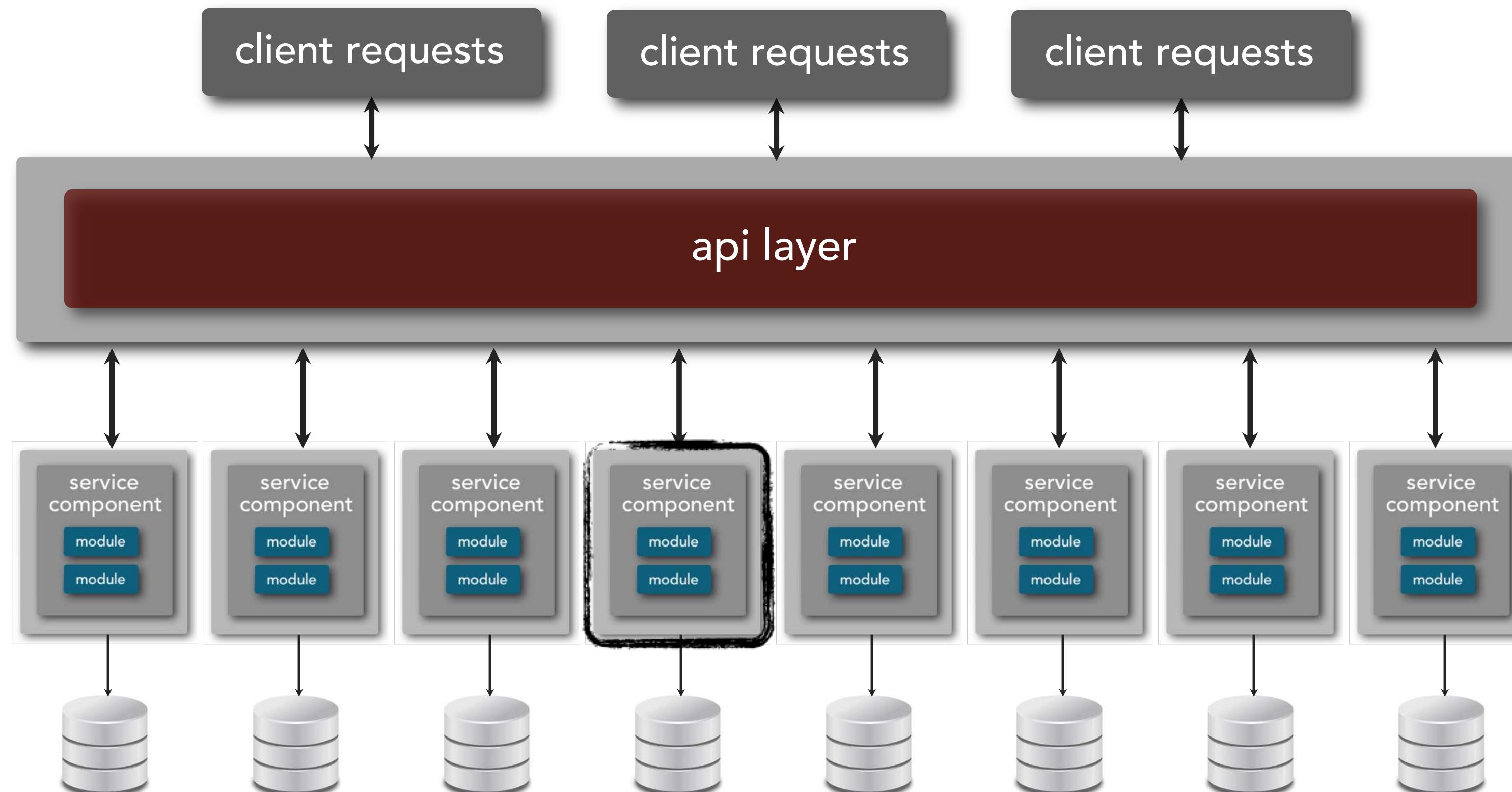
microservices architecture

service component



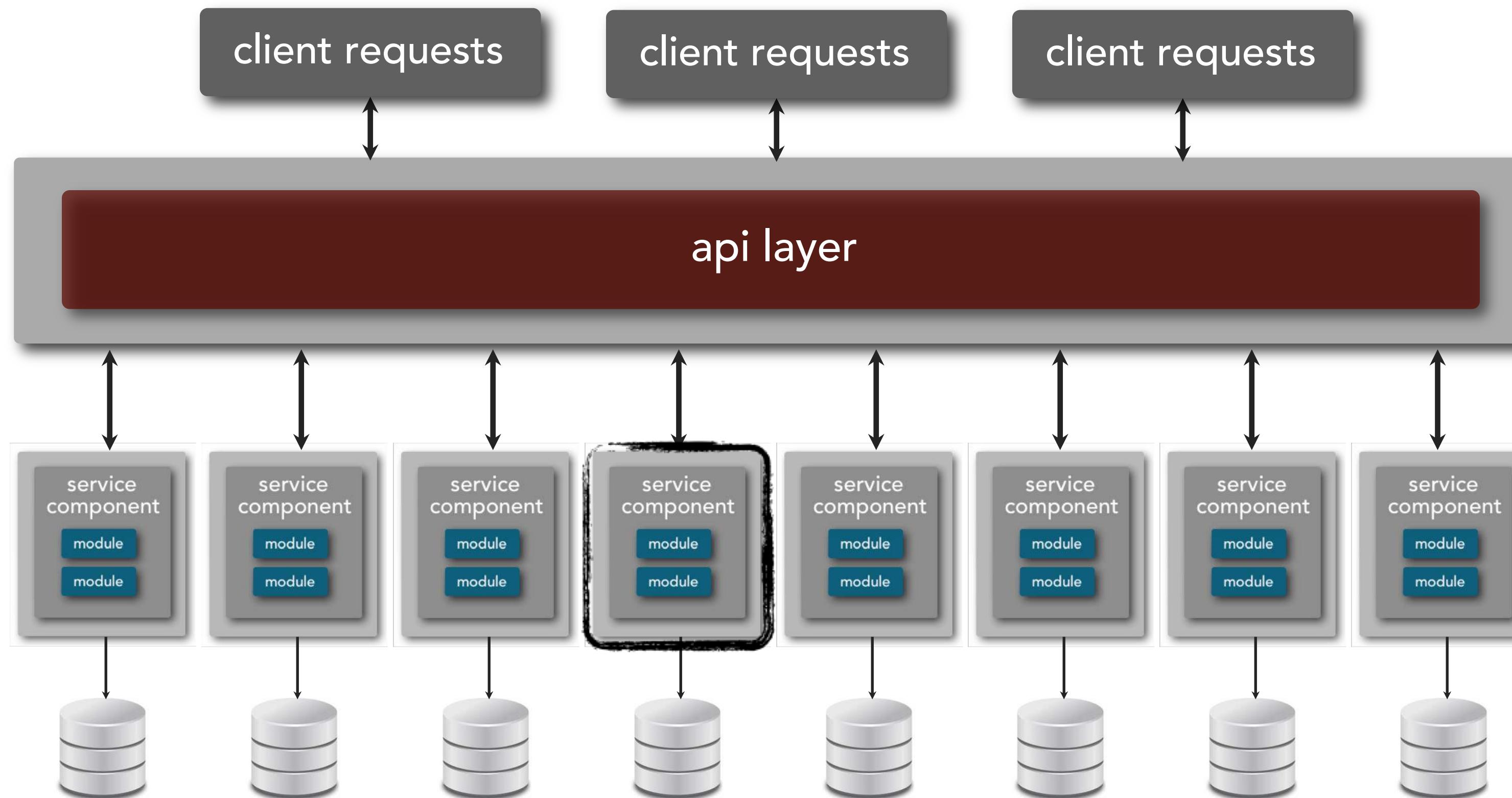
microservices architecture

service component



microservices architecture

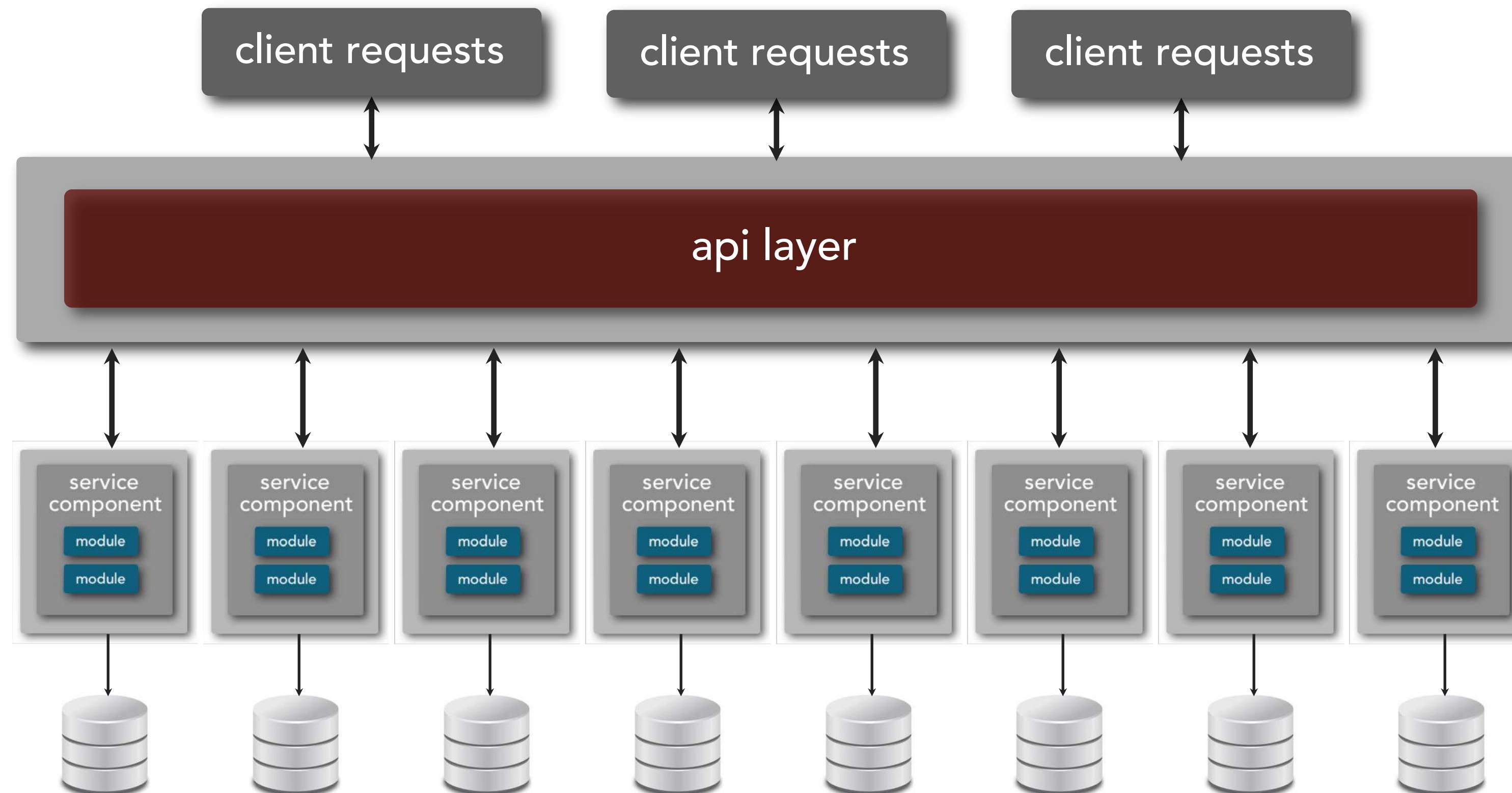
service component



microservices is a label, not a *description*.

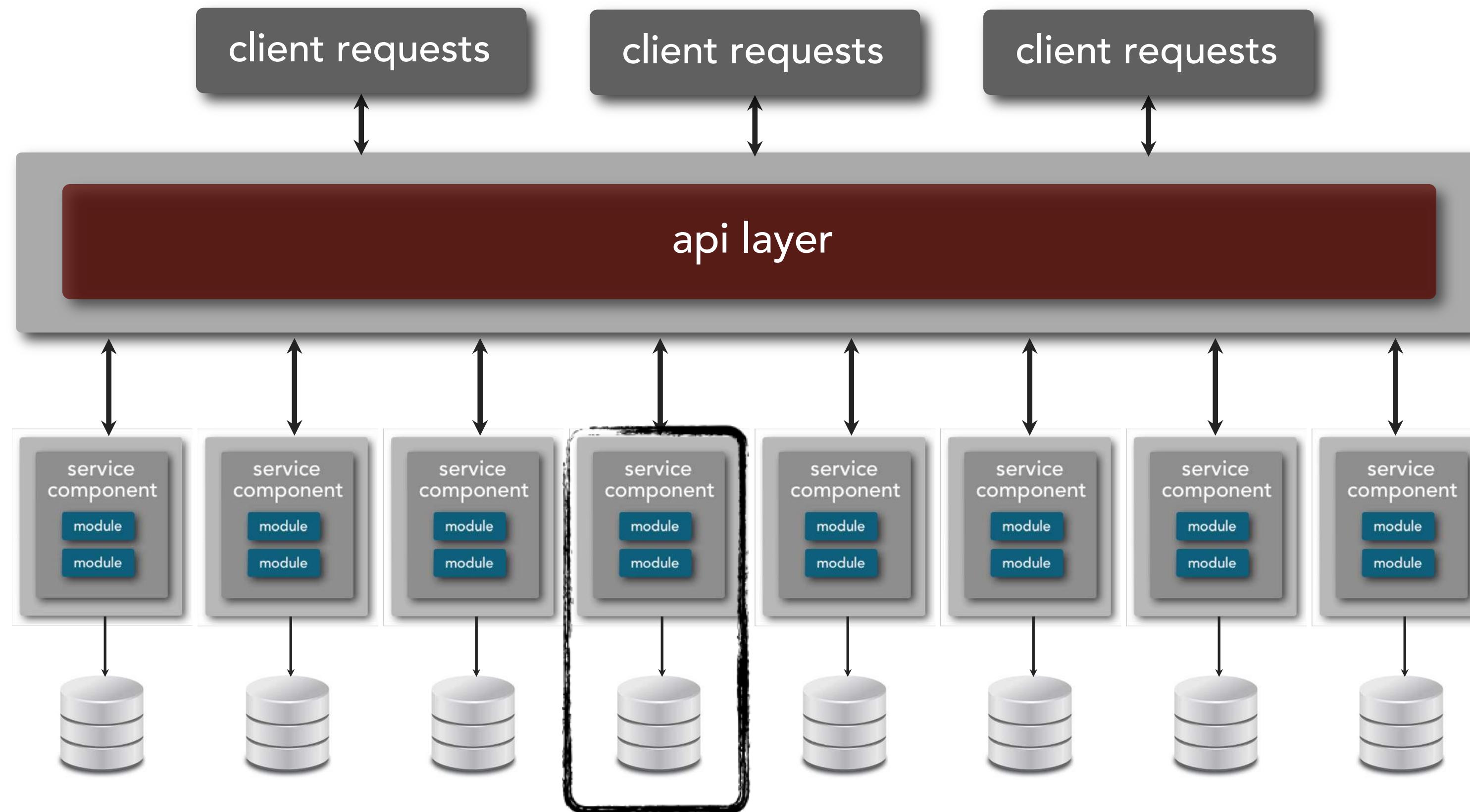
microservices architecture

bounded context



microservices architecture

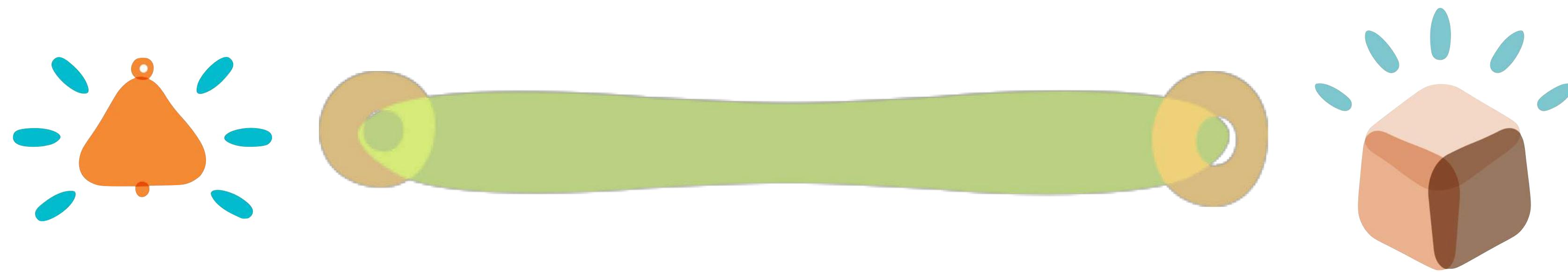
bounded context



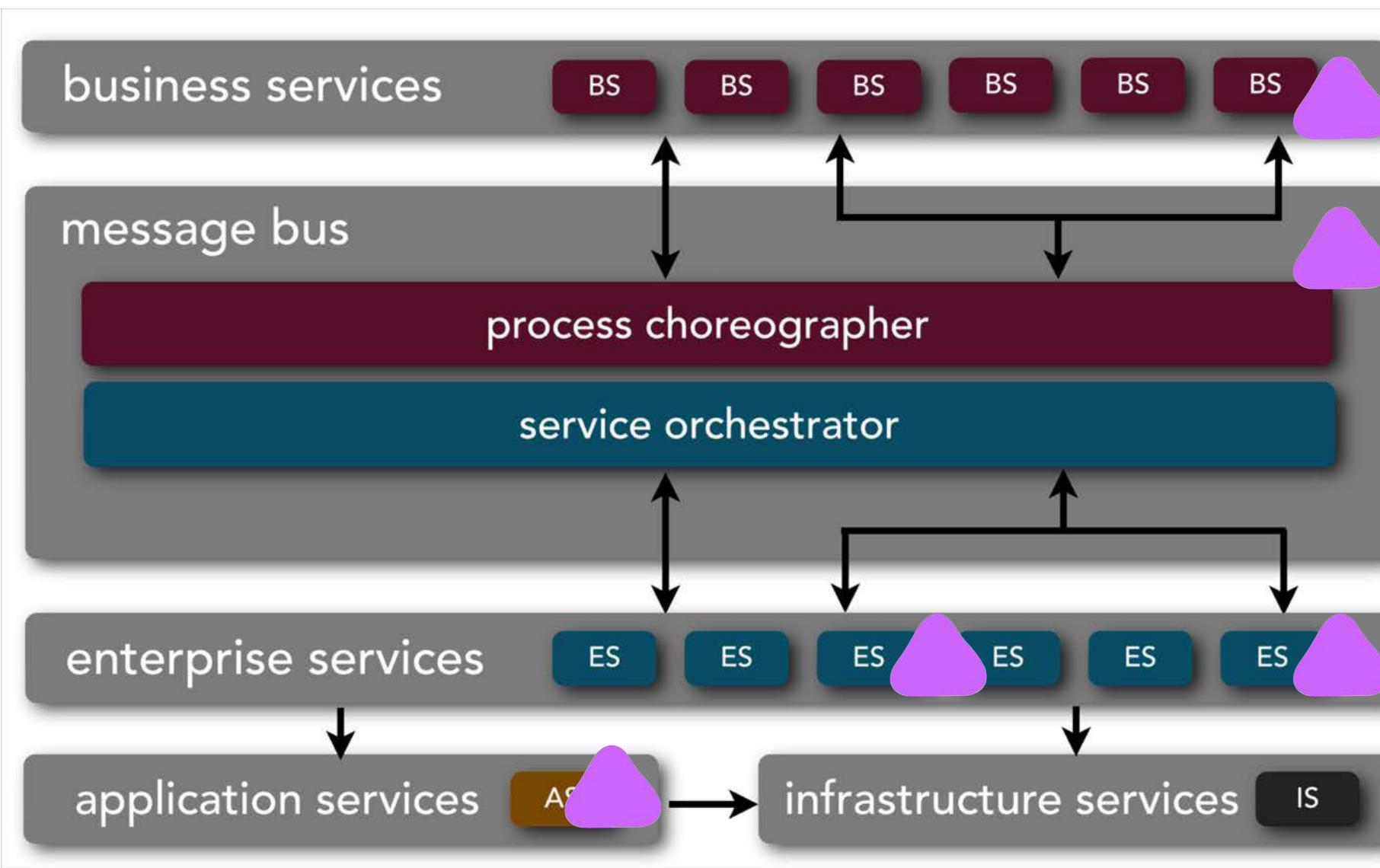
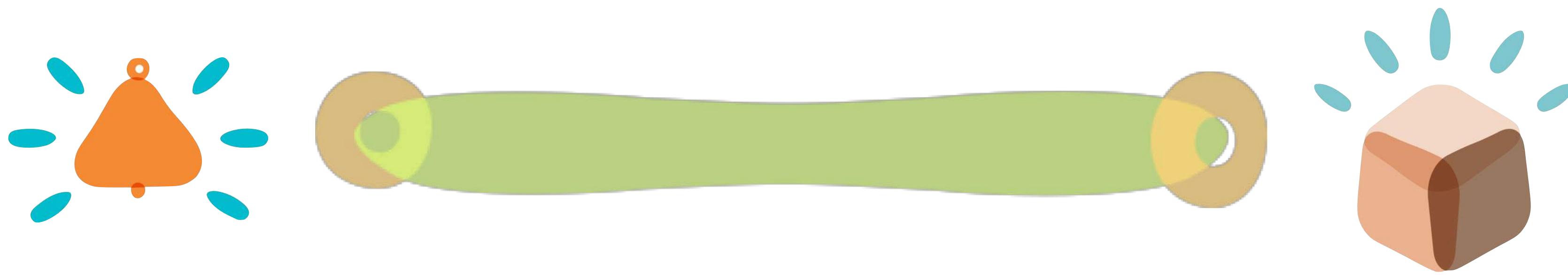
bounded context ≠ entity



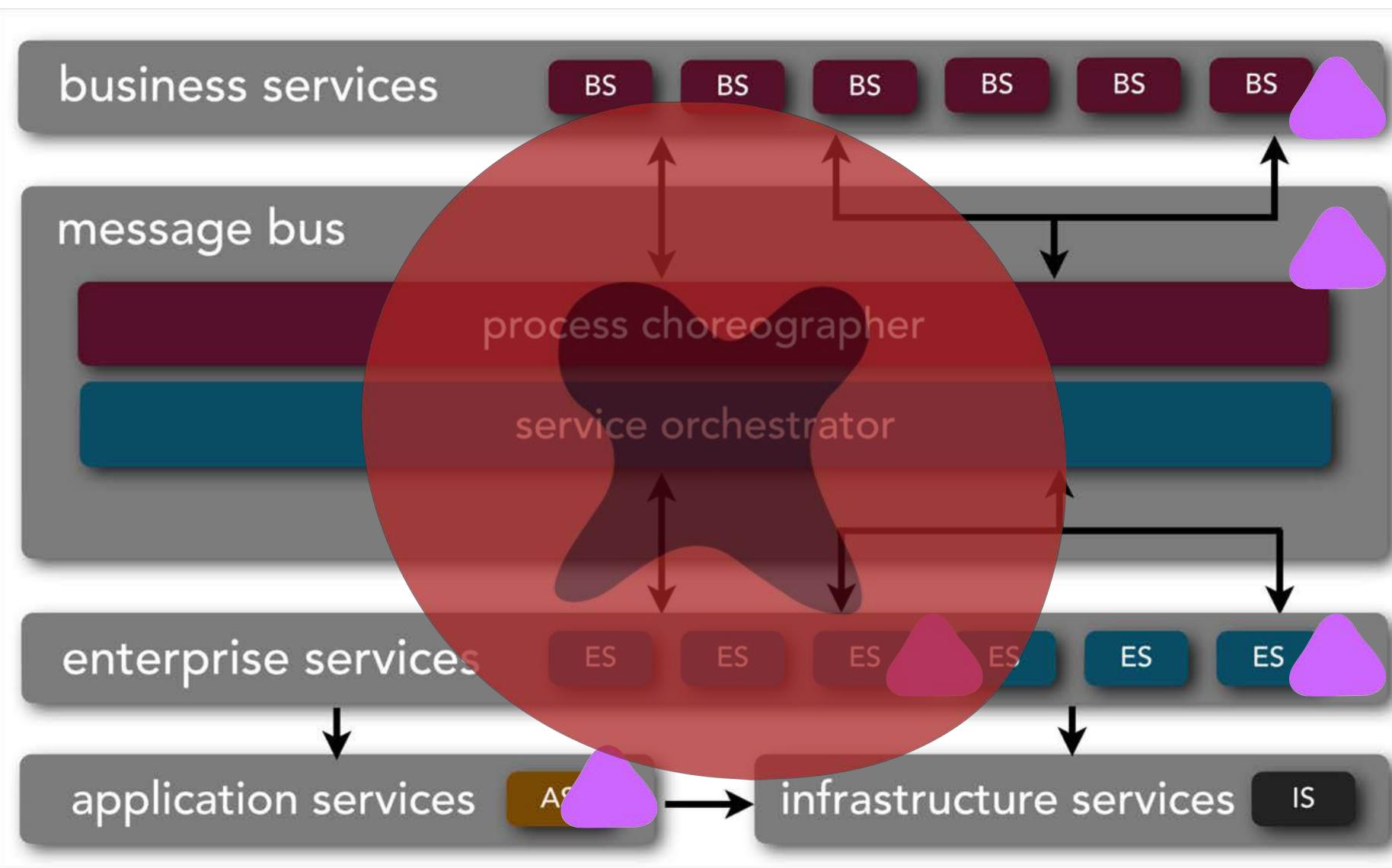
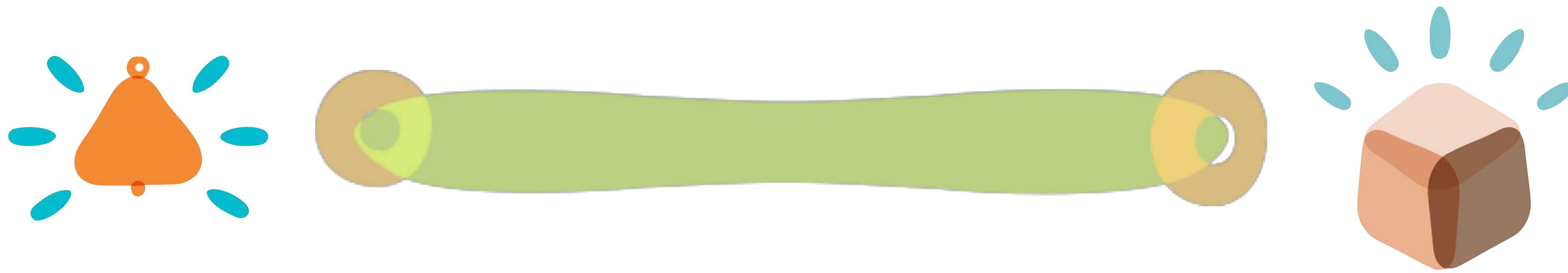
smart endpoints, dumb pipes



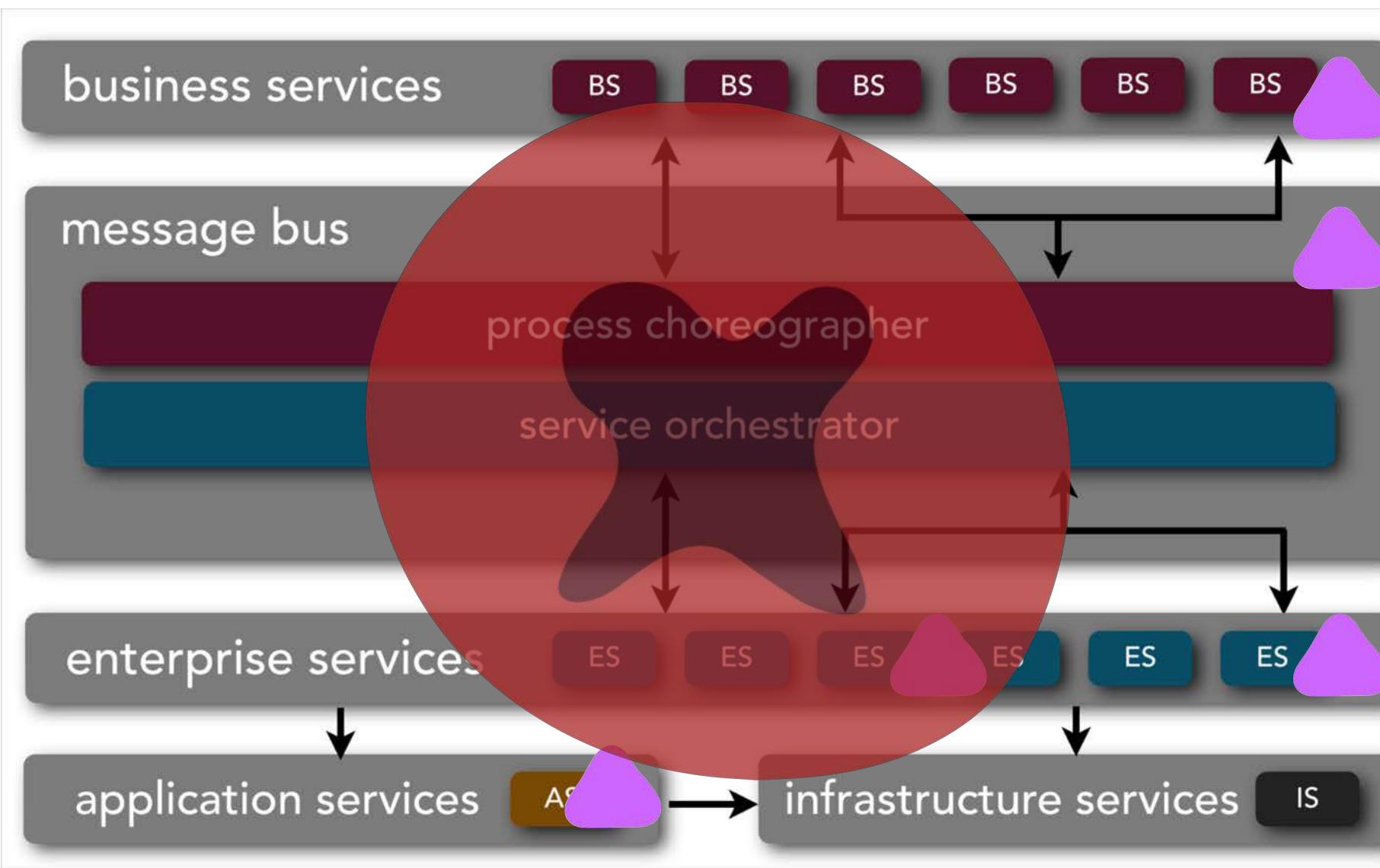
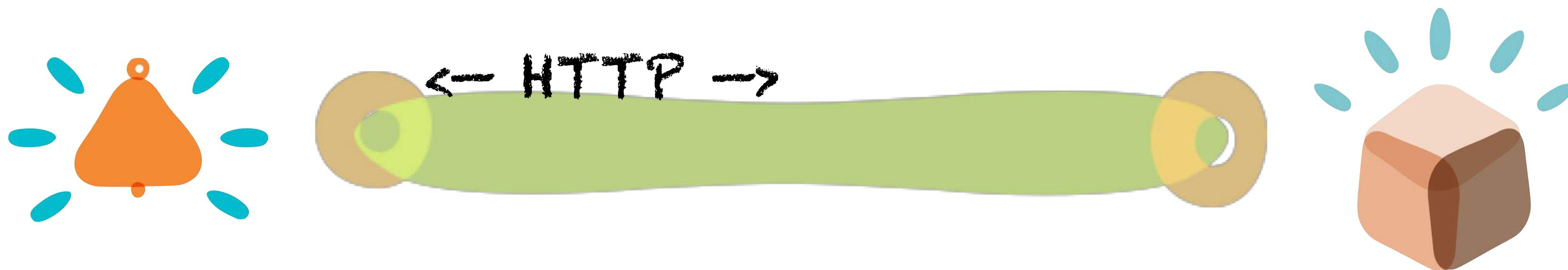
smart endpoints, dumb pipes



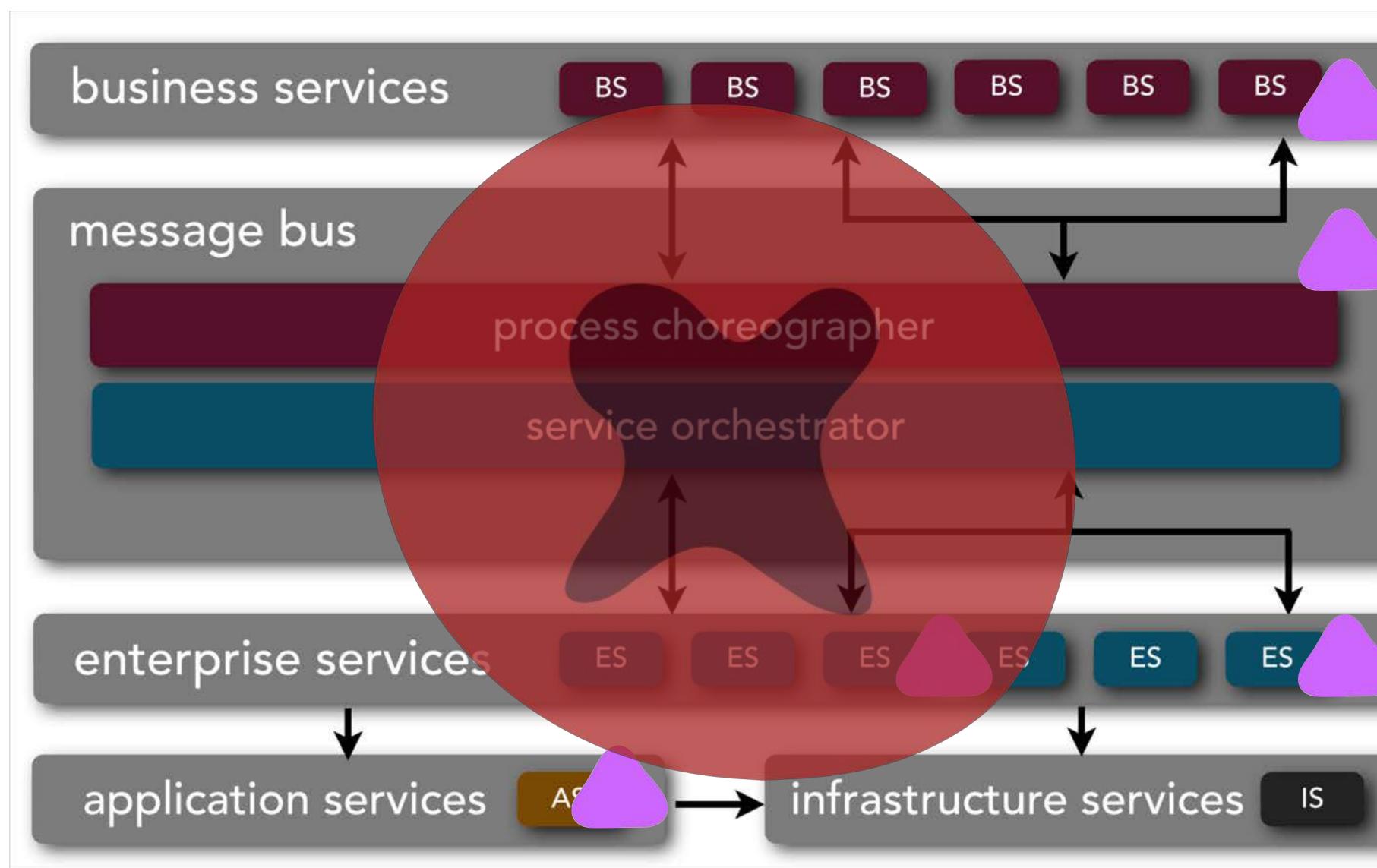
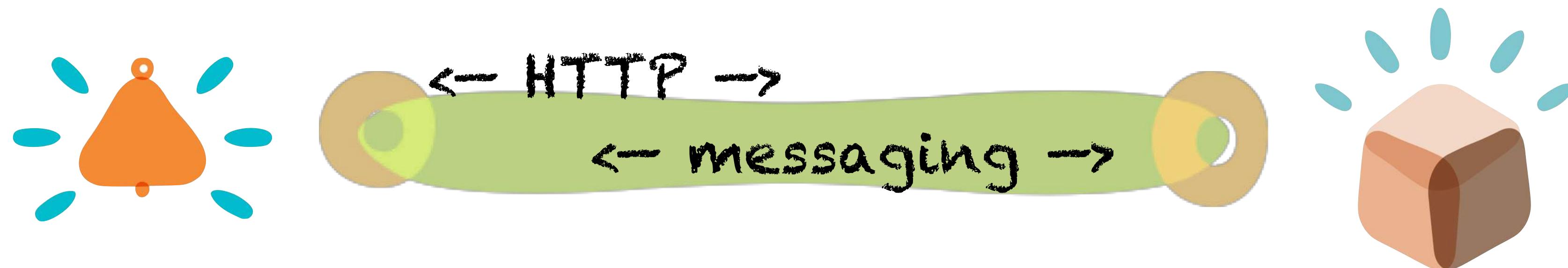
smart endpoints, dumb pipes



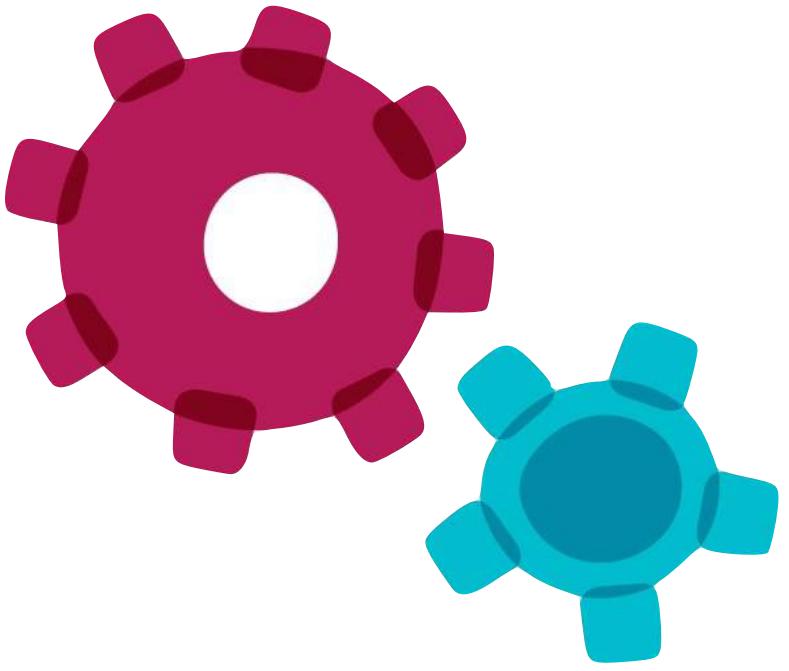
smart endpoints, dumb pipes



smart endpoints, dumb pipes

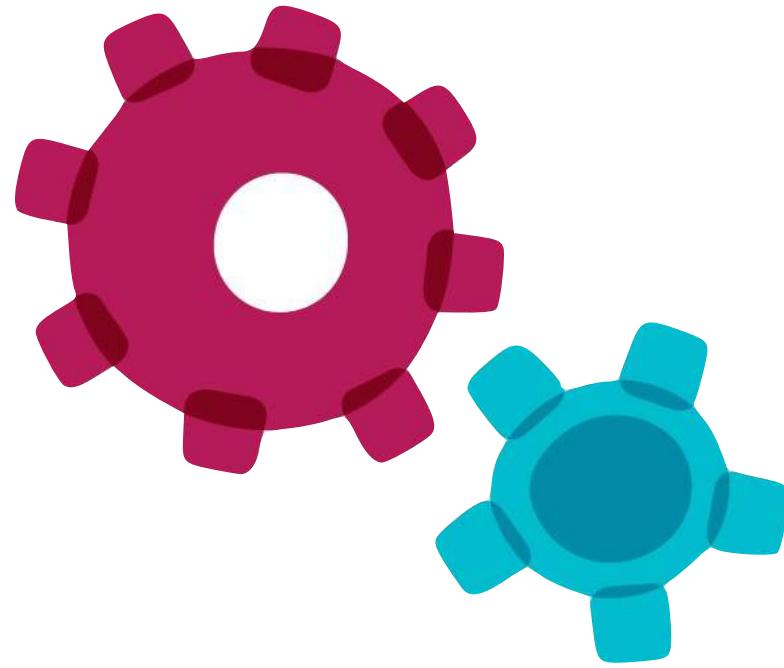


standardize on integration, not platform



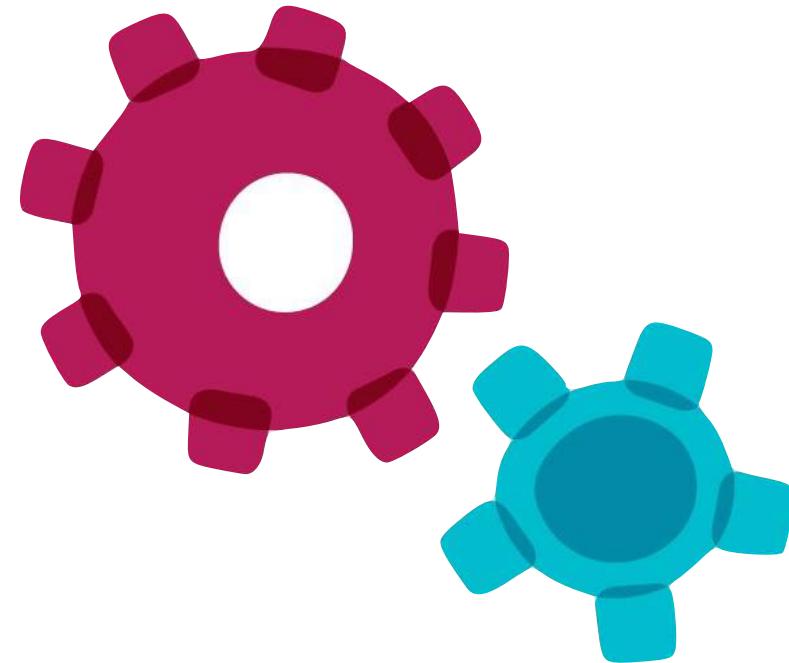
standardize on integration, not platform

embrace polyglot solutions where sensible



standardize on integration, not platform

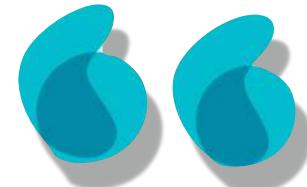
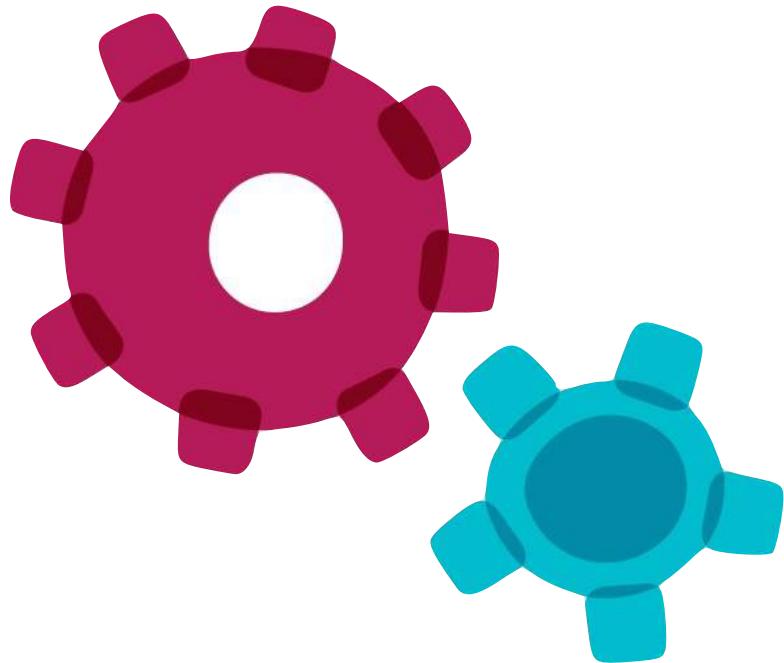
embrace polyglot solutions where sensible



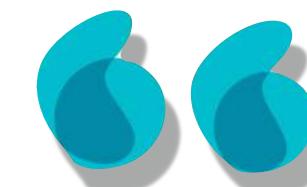
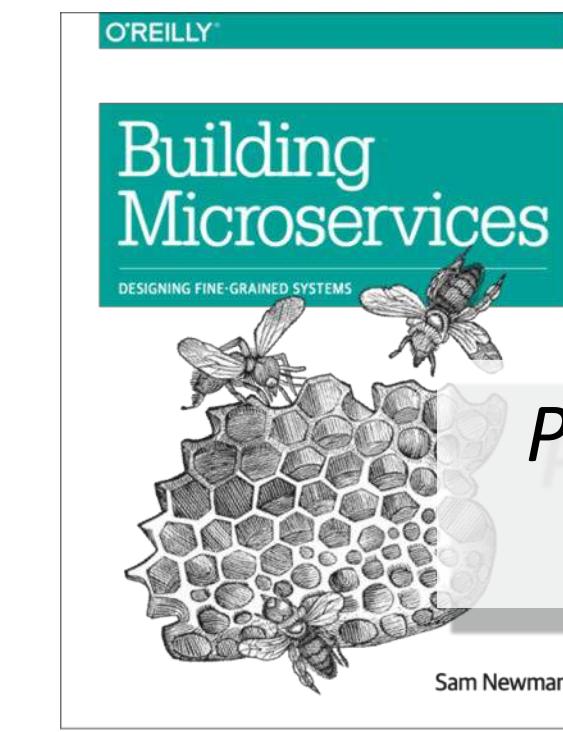
Standardize in the gaps between services - be flexible about what happens inside the boxes

standardize on integration, not platform

embrace polyglot solutions where sensible

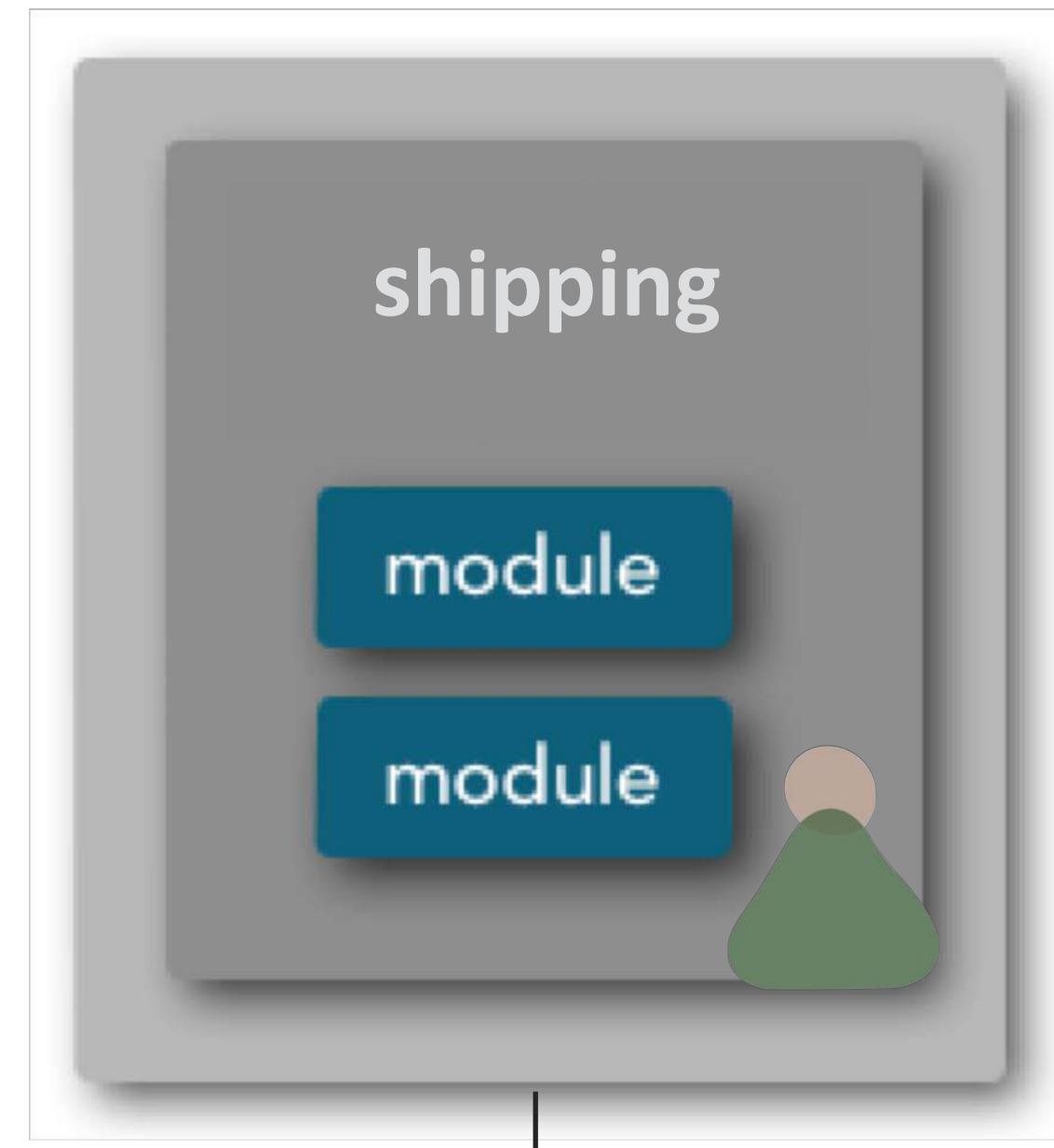
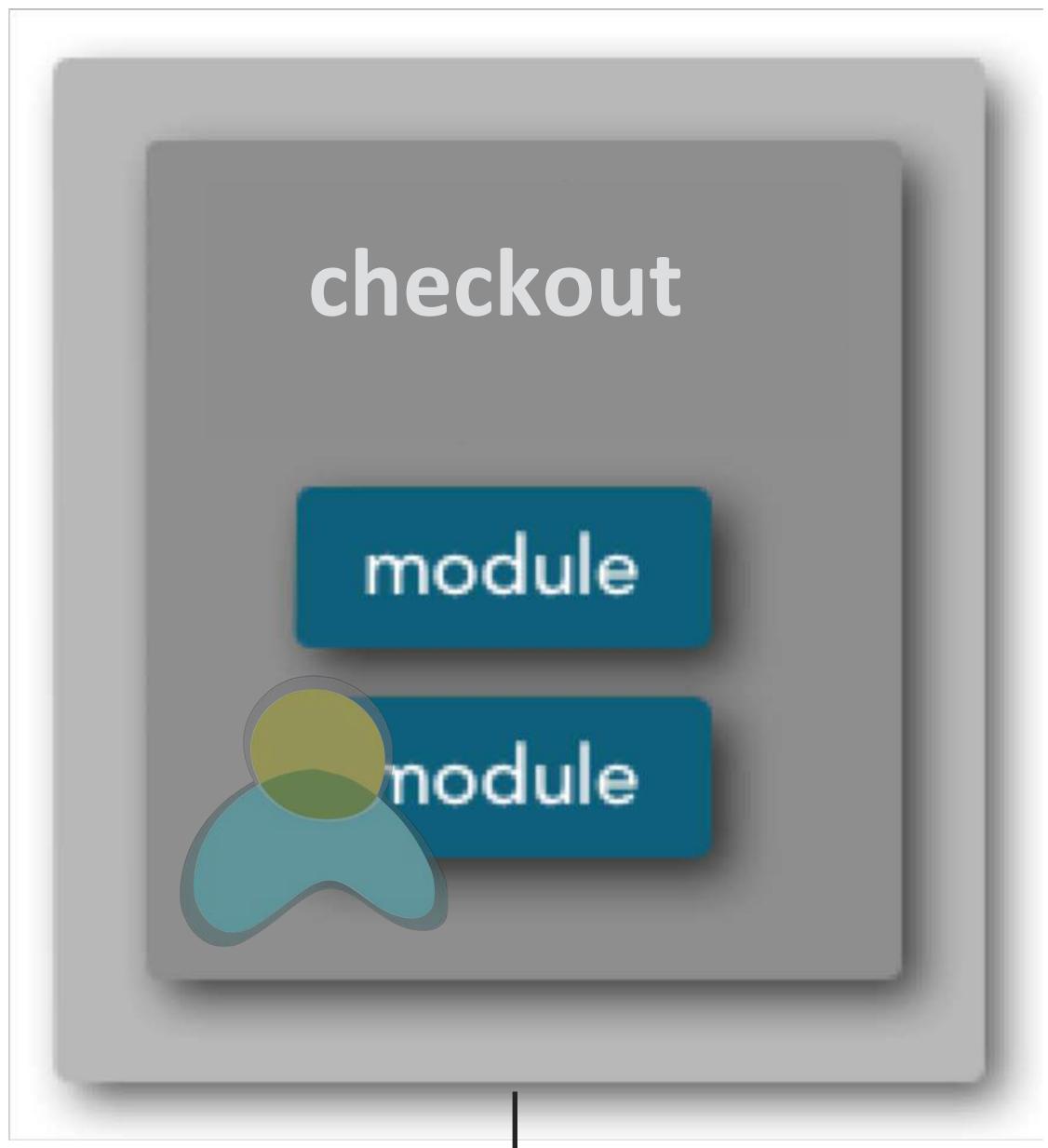


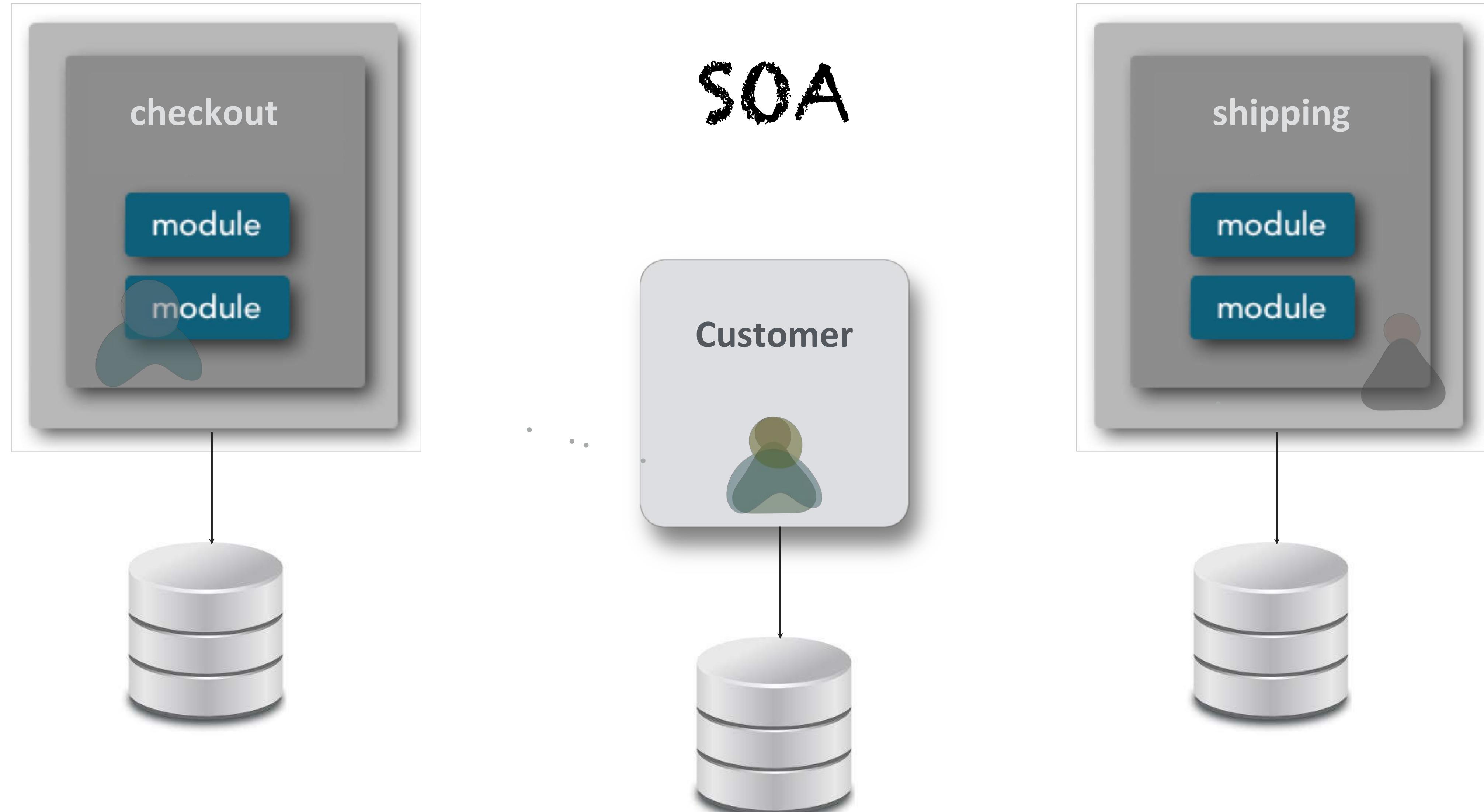
Standardize in the gaps between services - be flexible about what happens inside the boxes

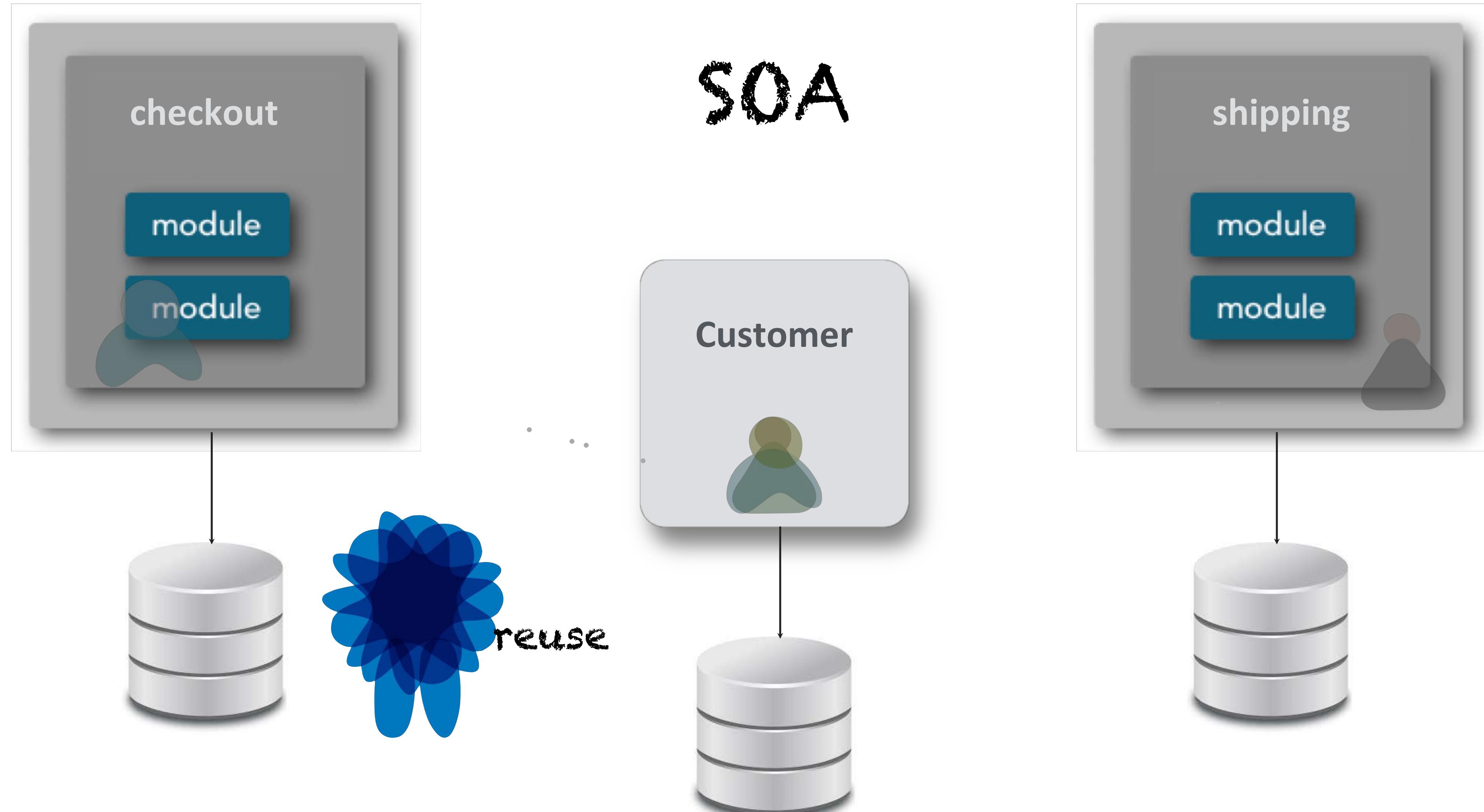


Pick some sensible conventions, and stick with them.

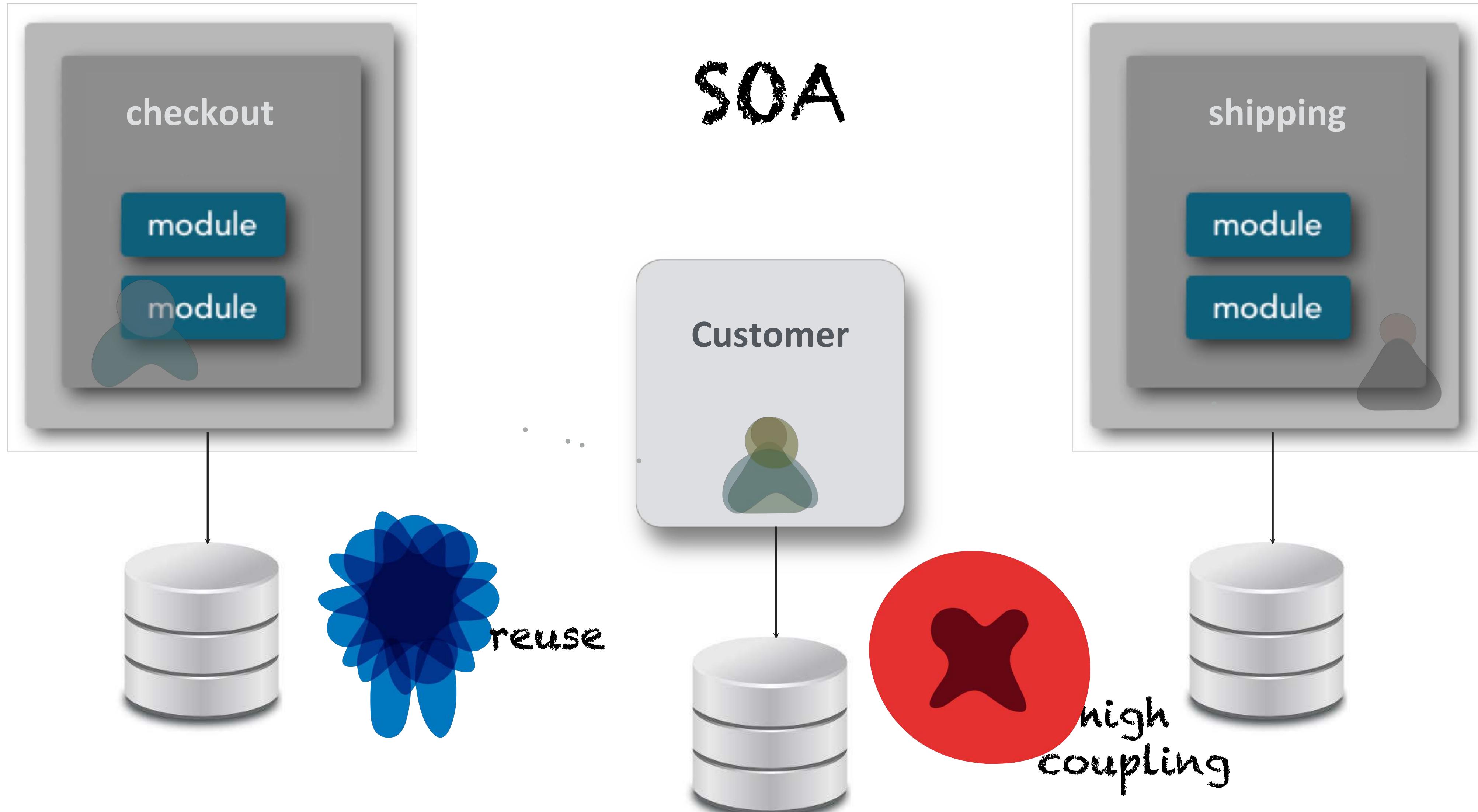
SOA



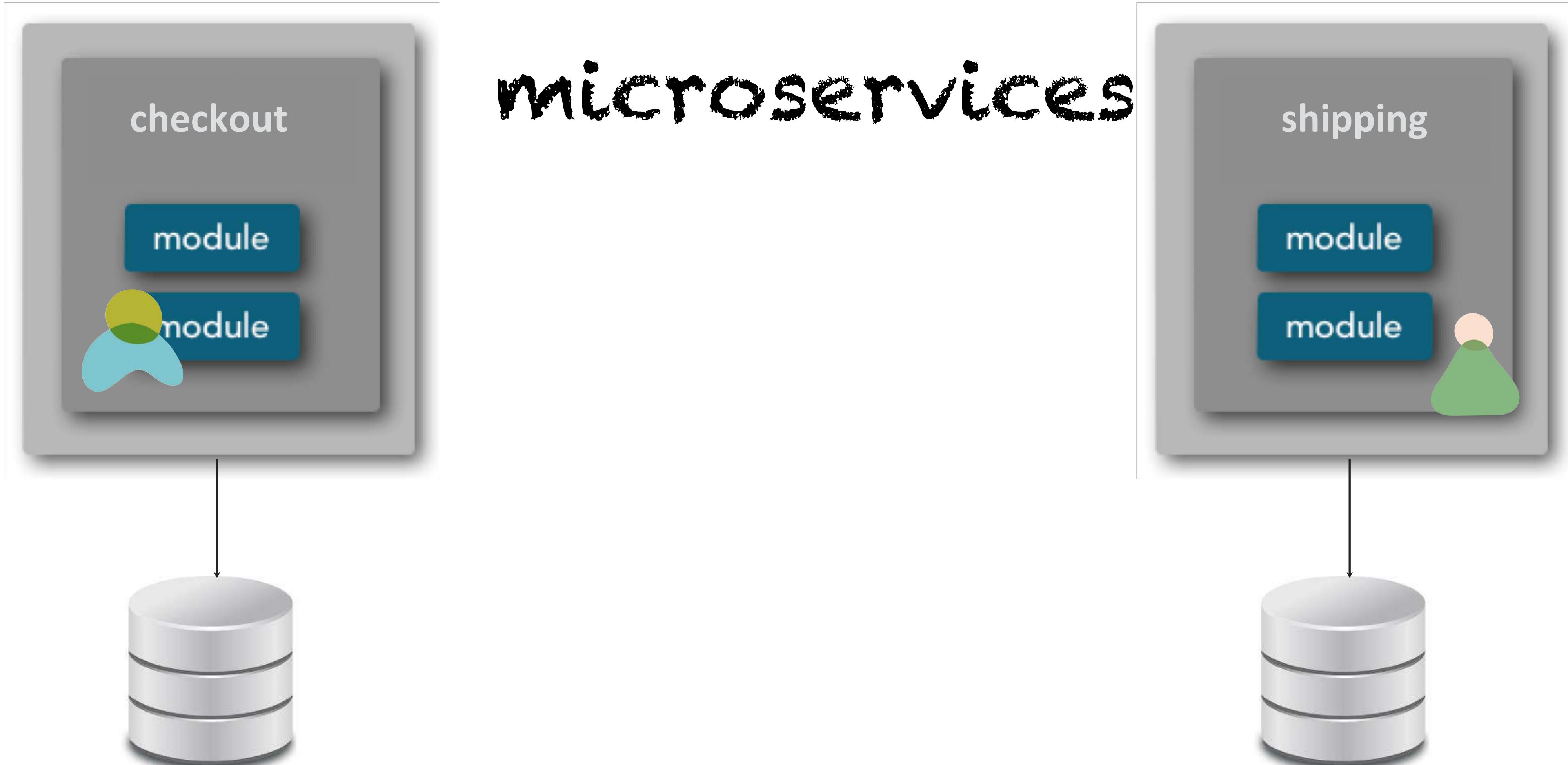


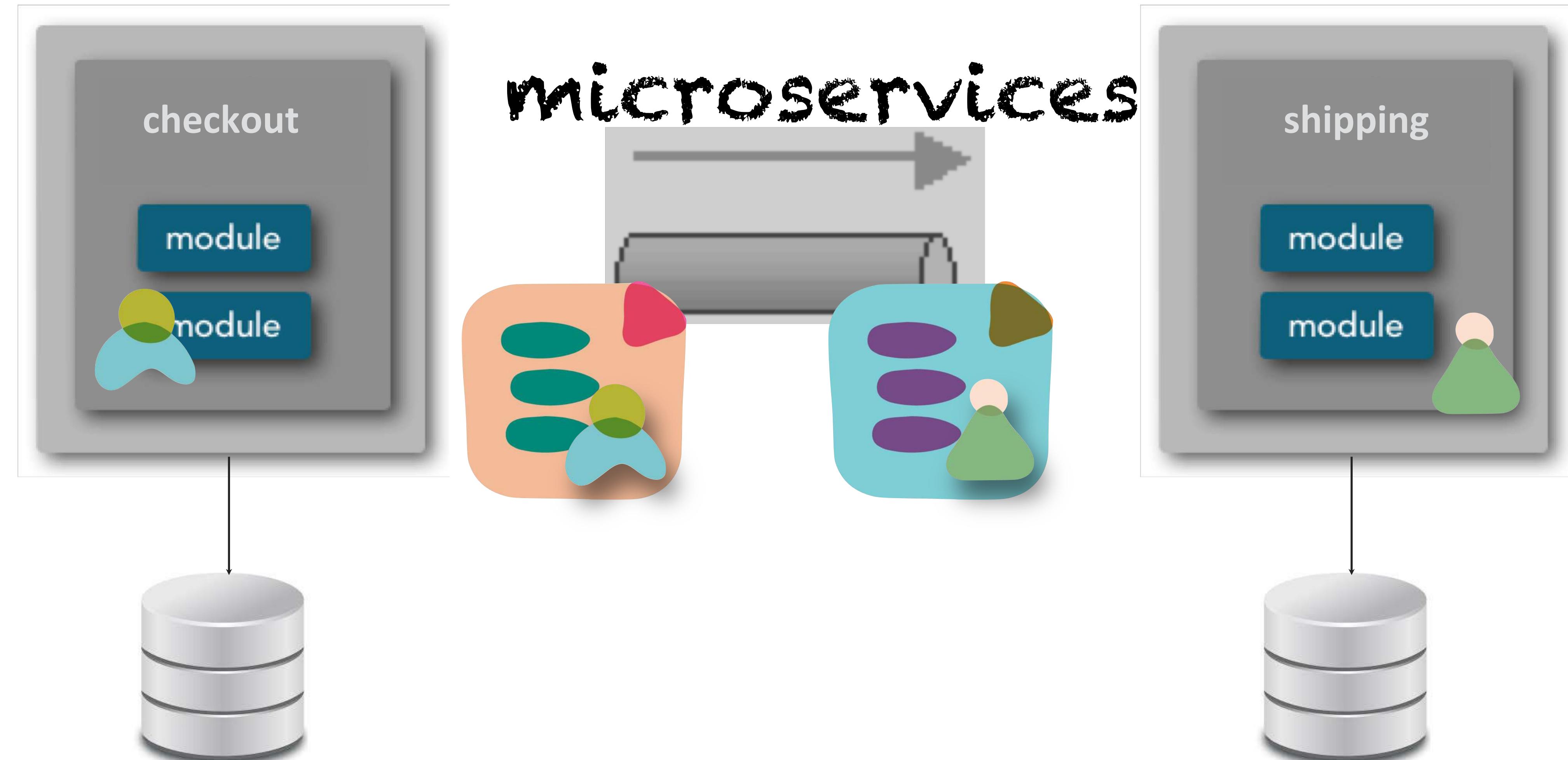


SOA

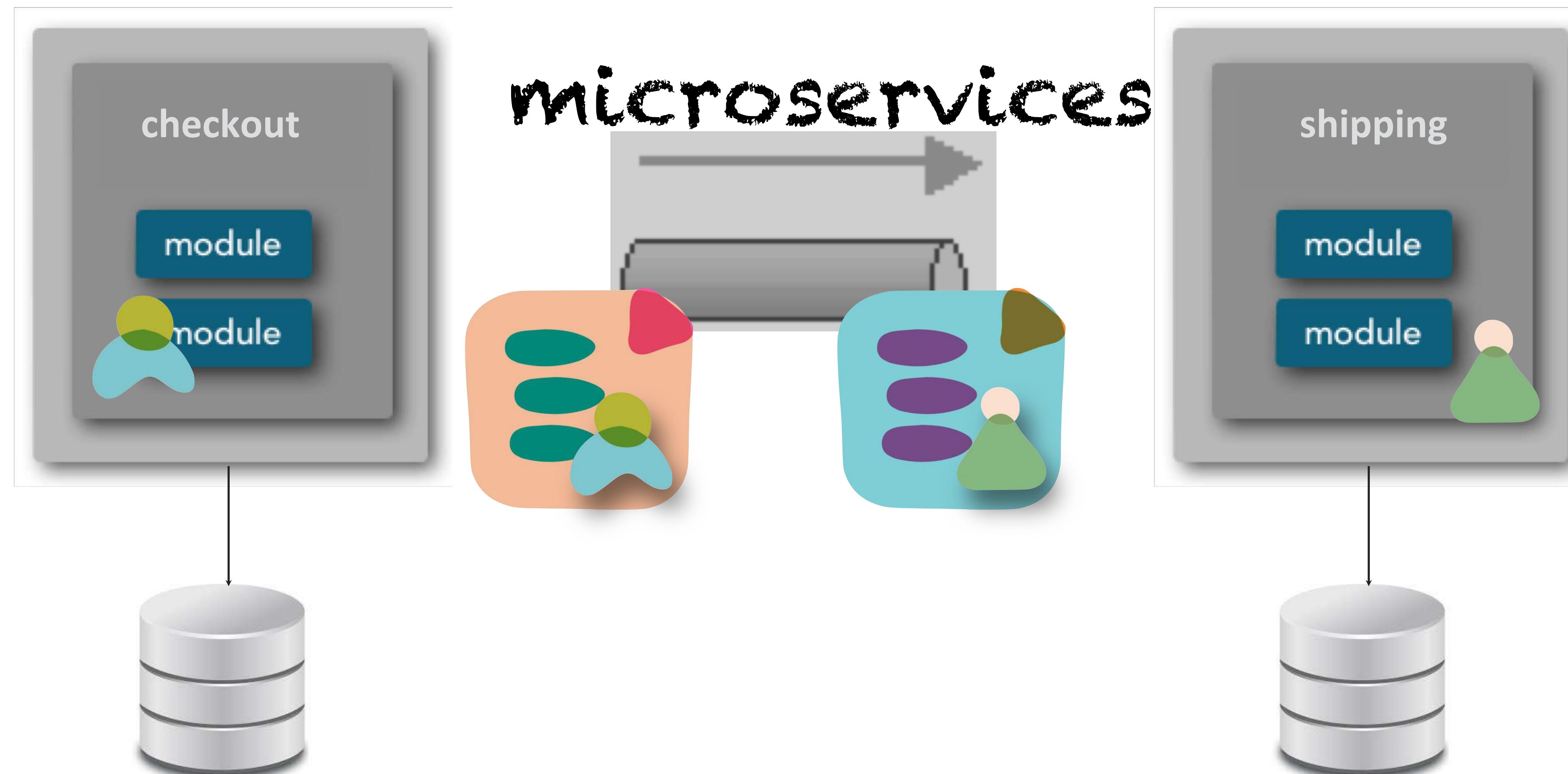


microservices





prefer duplication over coupling

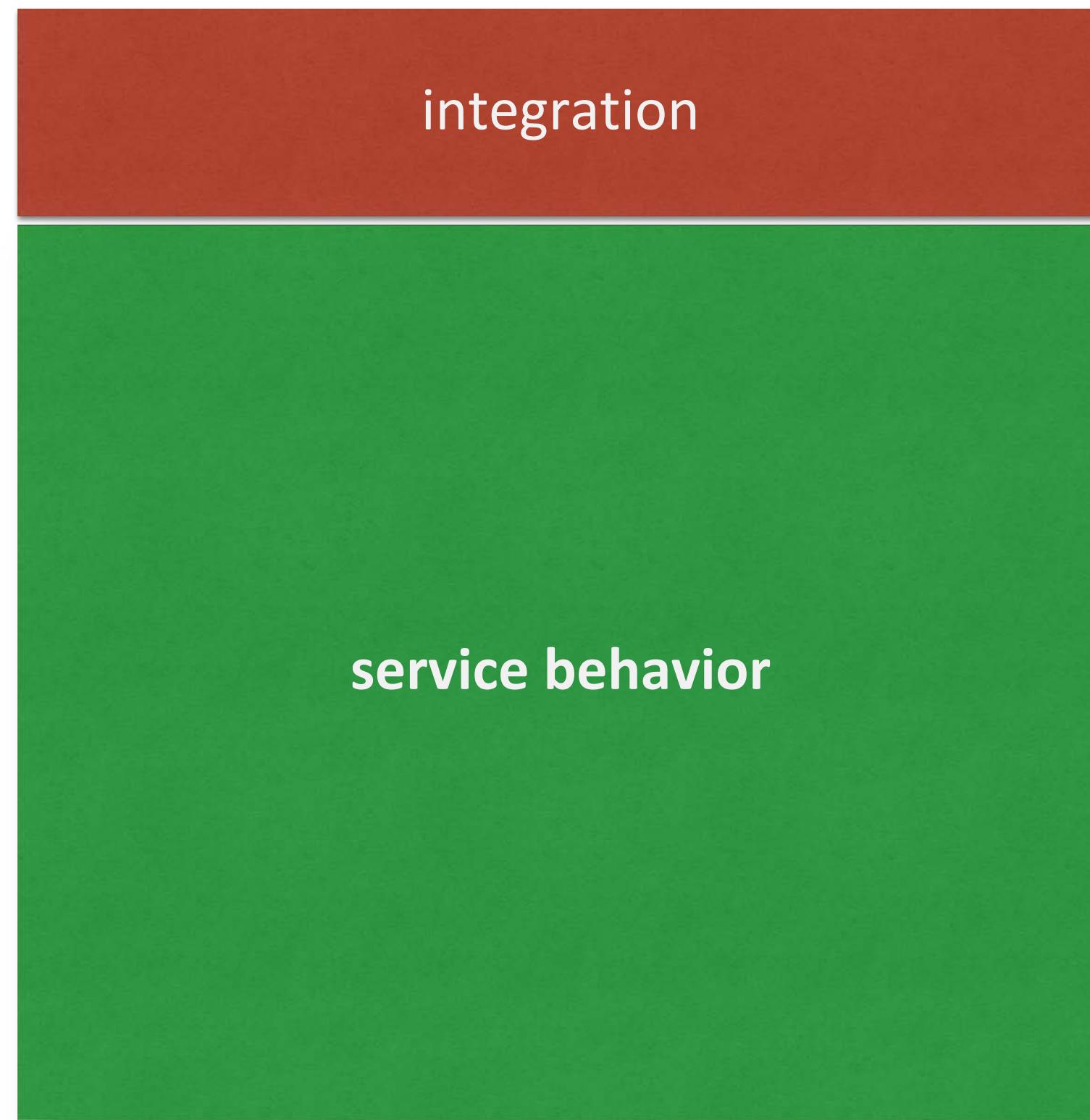


technical consistency

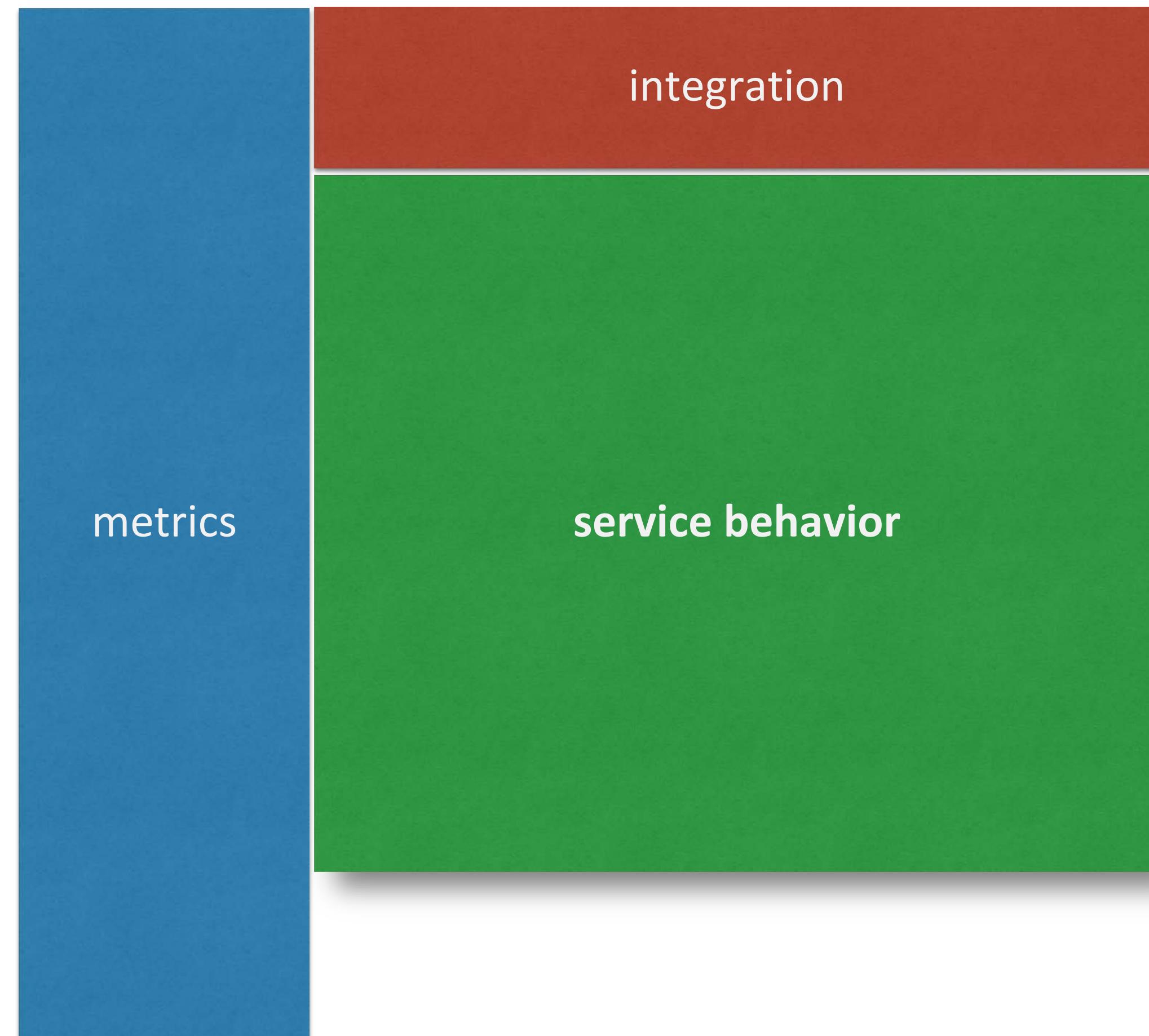


service behavior

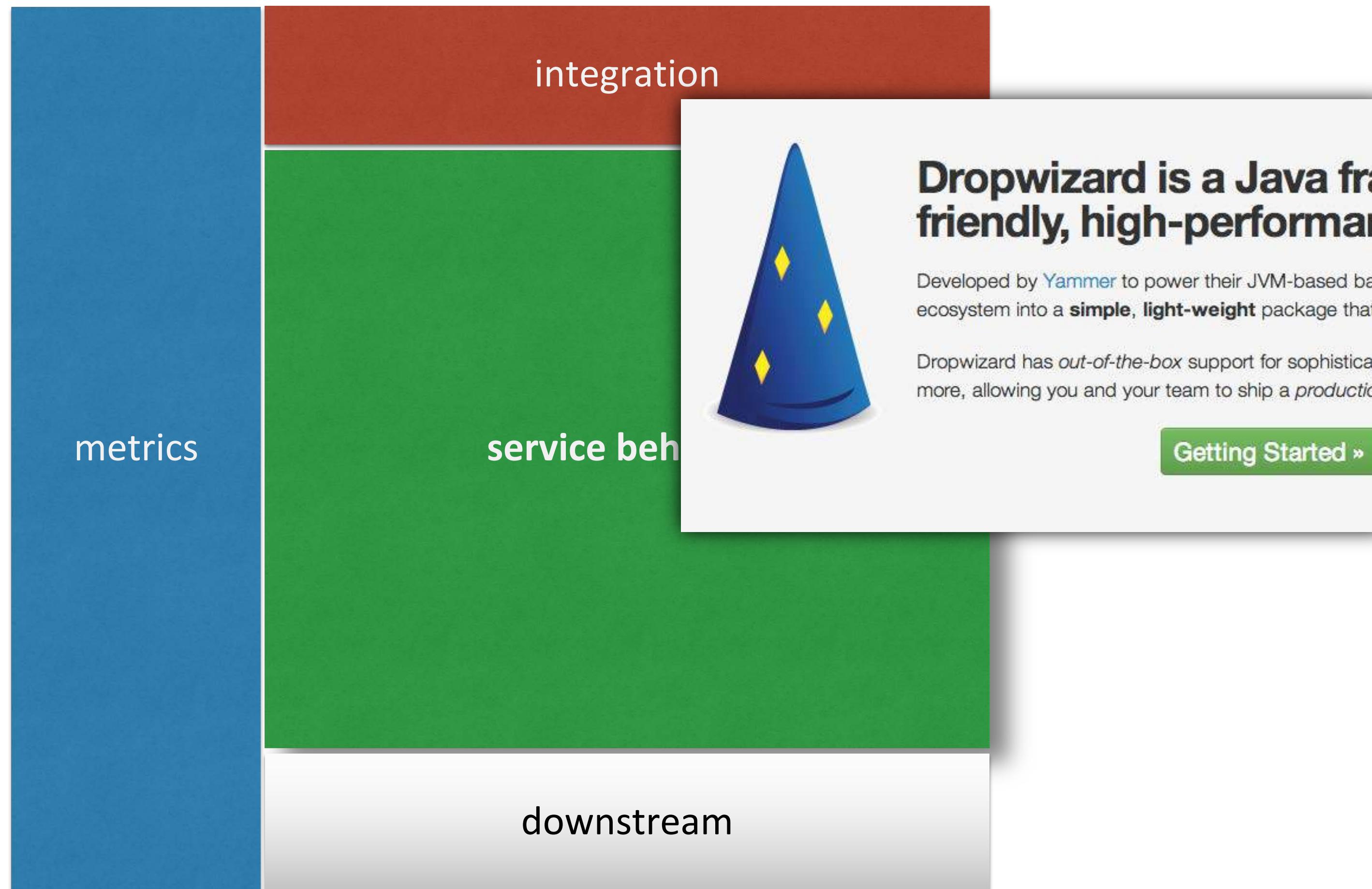
technical consistency



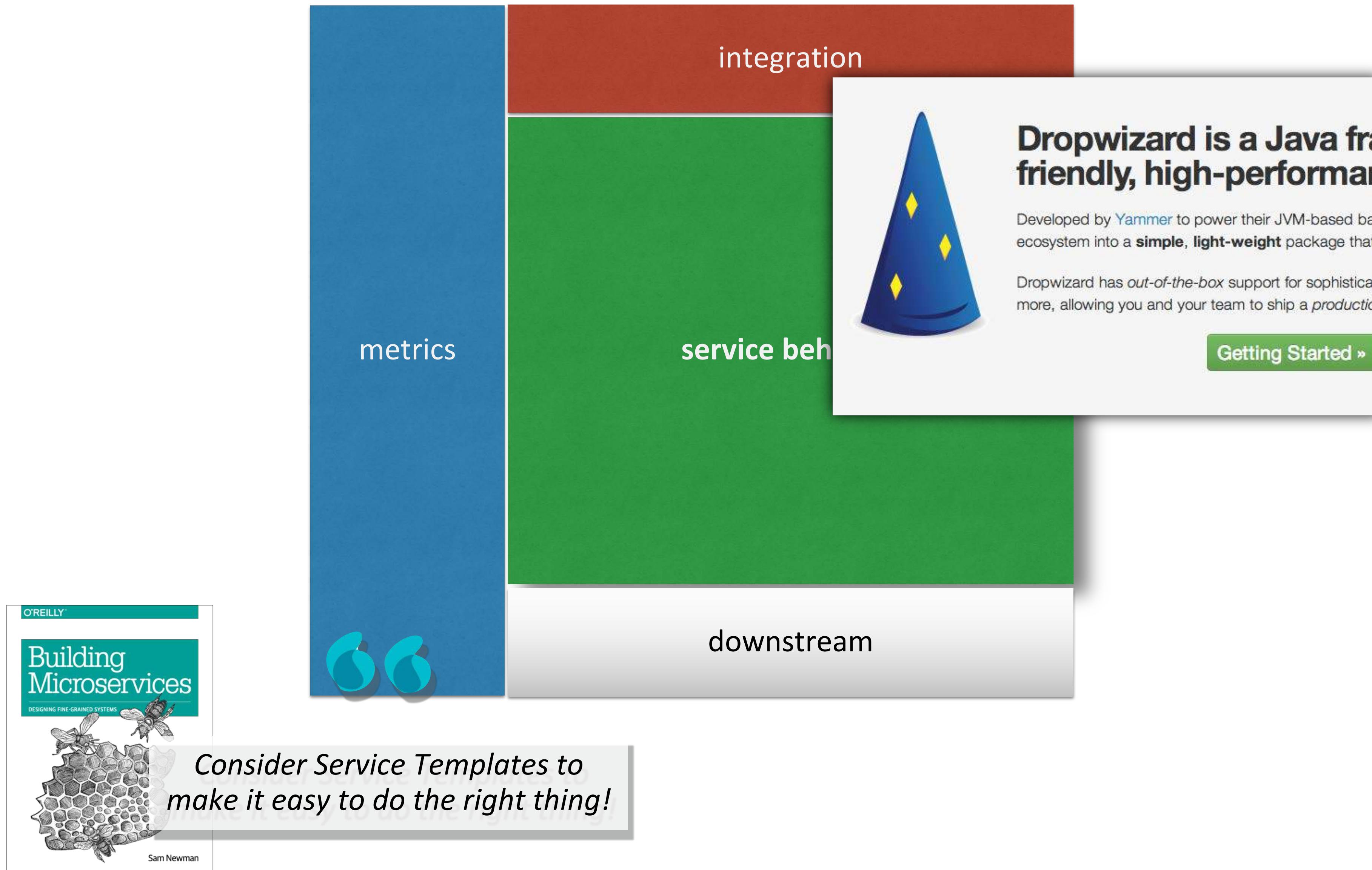
technical consistency



technical consistency



technical consistency



the rise of service meshes

Phil Calçado [Microservices Patterns](#) [About](#)

Pattern: Service Mesh

Aug 3, 2017

• Microservices • Distributed Systems • Service Mesh • Patterns •

Since their first introduction many decades ago, we learnt that distributed systems enable use cases we couldn't even think about before them, but they also introduce all sorts of new issues.

When these systems were rare and simple, engineers dealt with the added complexity by minimising the number of remote interactions. The safest way to handle distribution has been to avoid it as much as possible, even if that meant duplicated logic and data across various systems.

But our needs as an industry pushed us even further, from a few larger central computers to hundreds and thousands of small services. In this new world, we've had to start taking our head out of the sand and tackling the new challenges and open questions, first with ad-hoc solutions done in a case-by-case manner and subsequently with something more sophisticated. As we find out more about the problem domain and design better solutions, we start crystallising some of the most common needs into patterns, libraries, and eventually platforms.

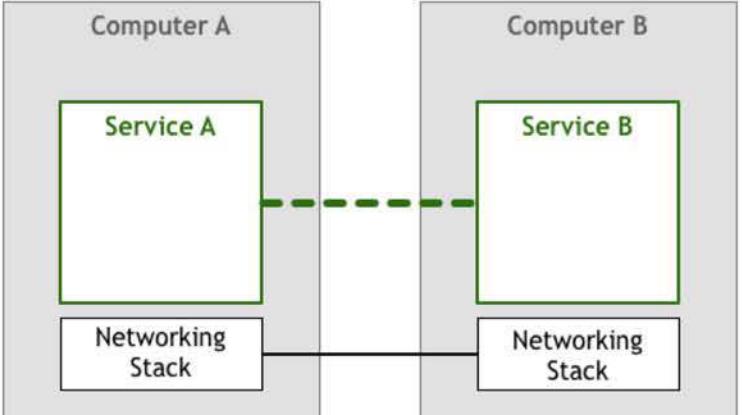
What happened when we first started networking computers

Since people first thought about getting two or more computers to talk to each other, they envisioned something like this:

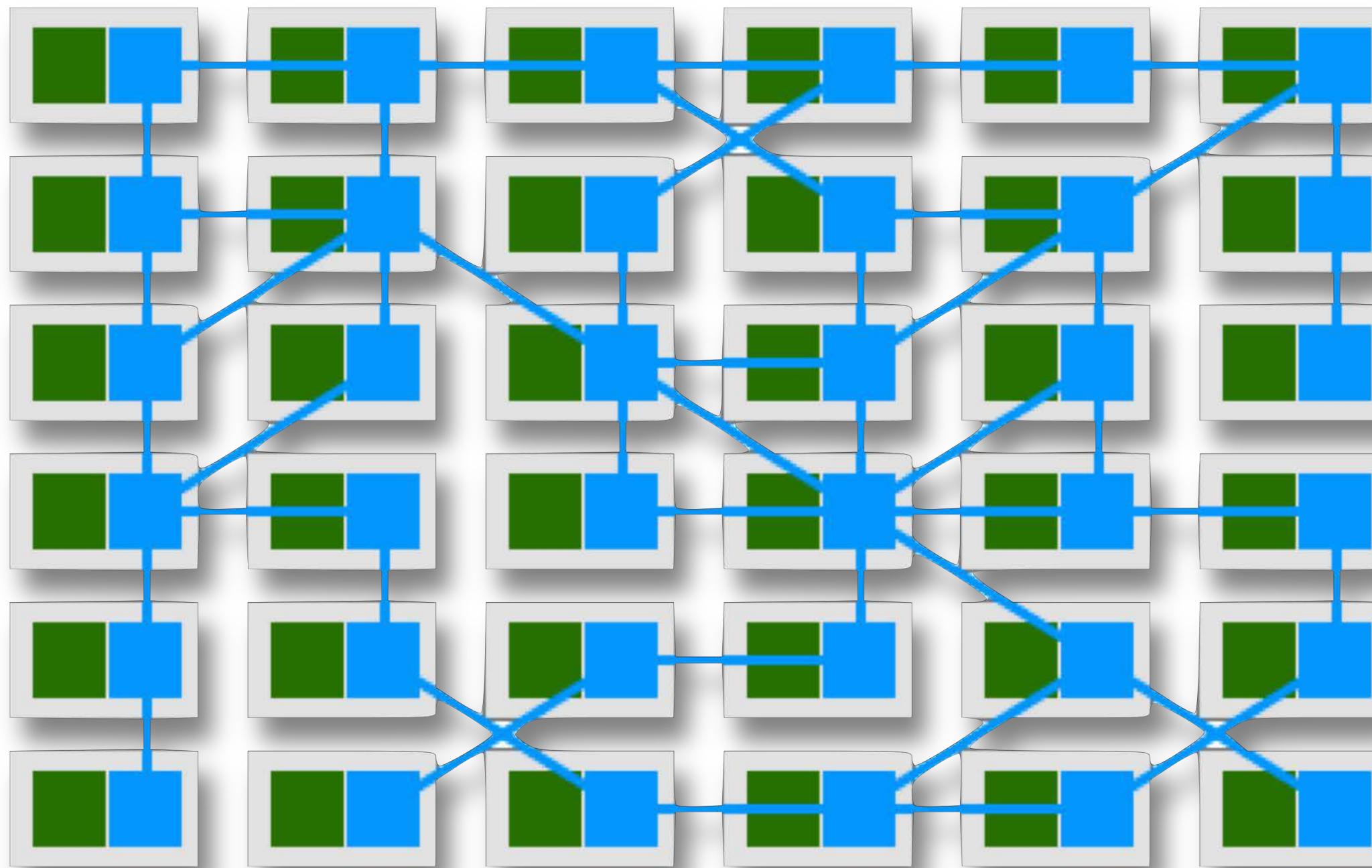


```
graph LR; SA[Service A] --- DB[Service B]
```

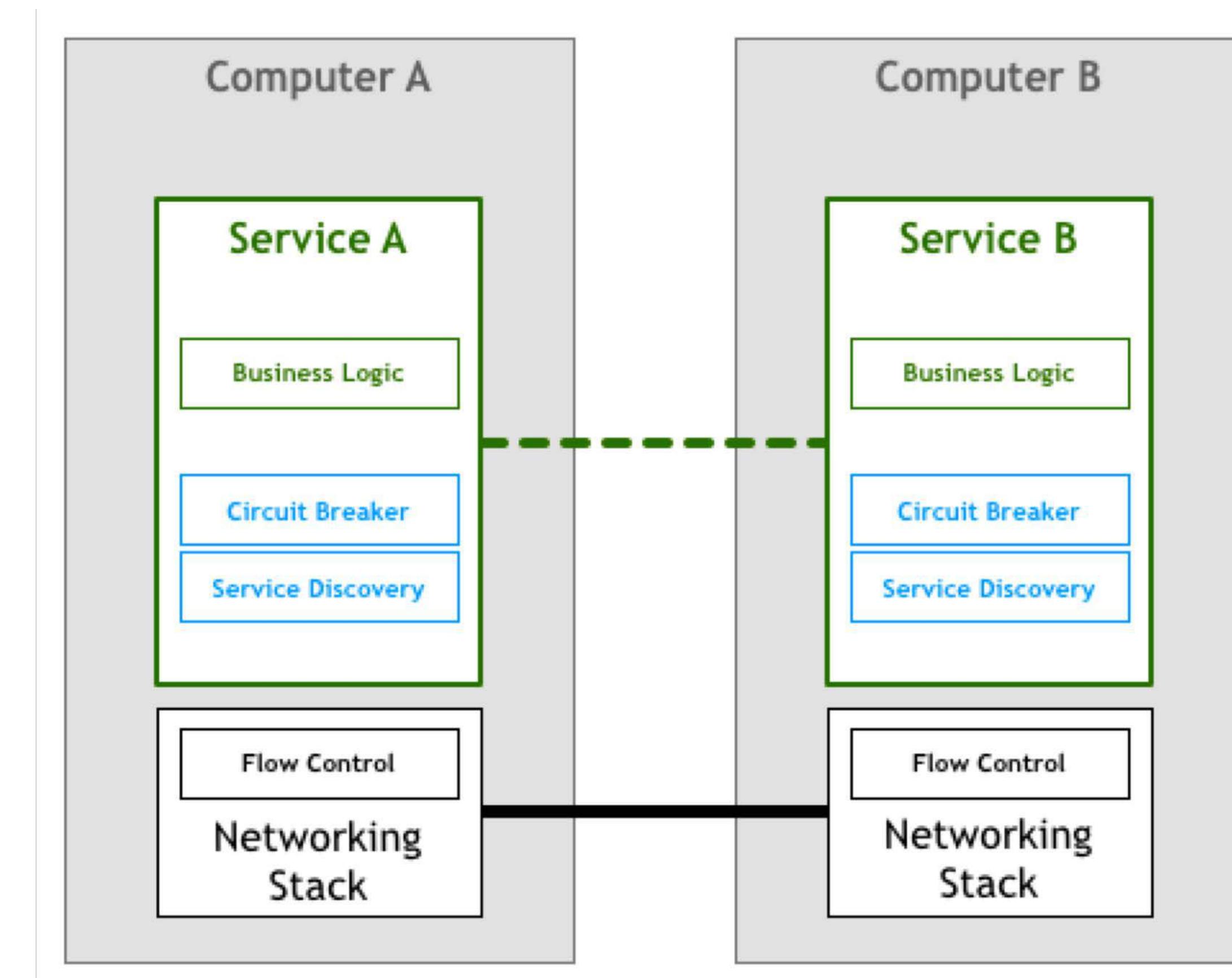
A service talks to another to accomplish some goal for an end-user. This is an obviously oversimplified view, as the many layers that translate between the bytes your code manipulates and the electric signals that are sent and received over a wire are missing. The abstraction is sufficient for our discussion, though. Let's just add a bit more detail by showing the networking stack as a distinct component:



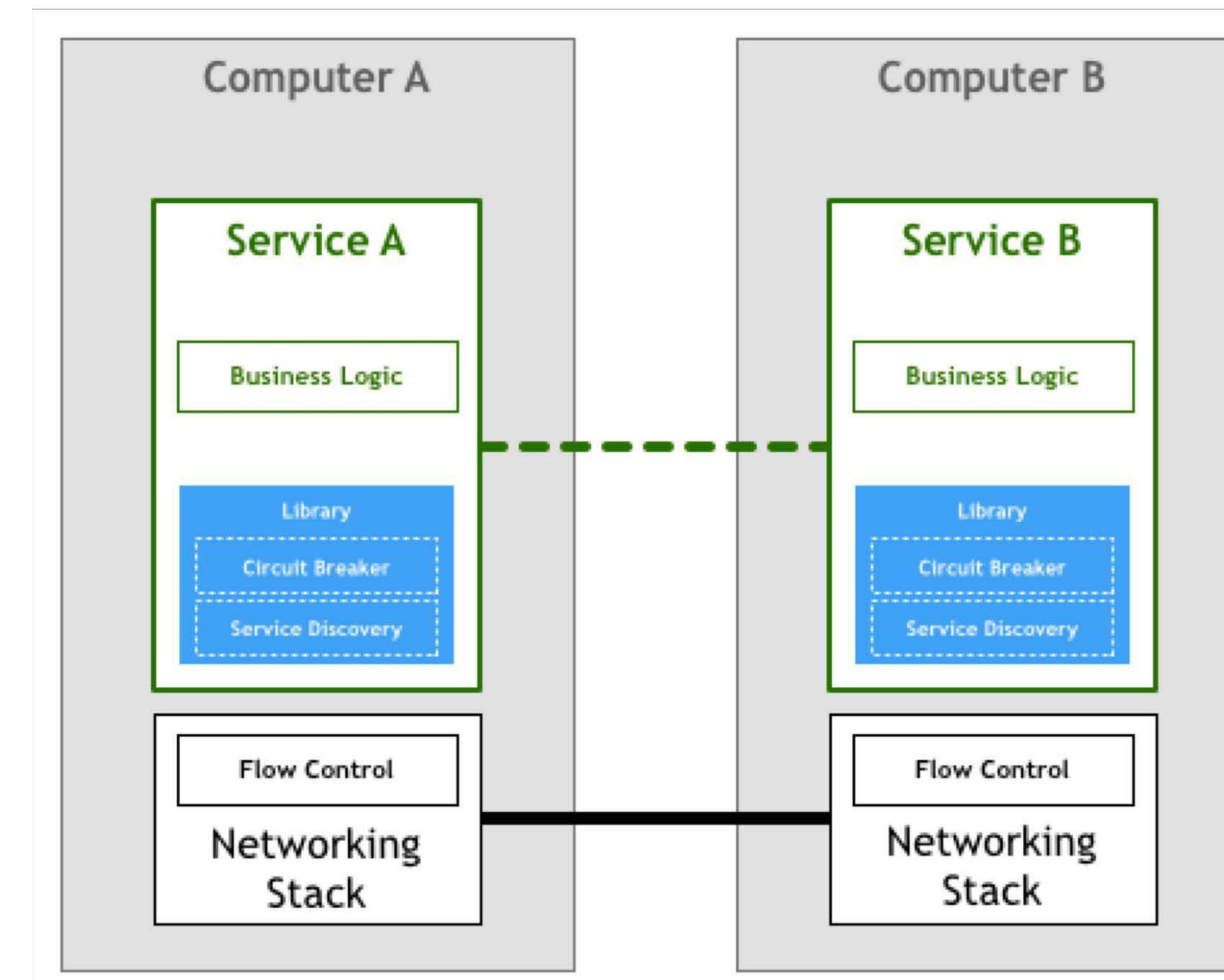
```
graph LR; subgraph CA [Computer A]; SA[Service A]; NS1[Networking Stack]; end; subgraph CB [Computer B]; SB[Service B]; NS2[Networking Stack]; end; SA --- DB[Service B]; NS1 --- NS2
```



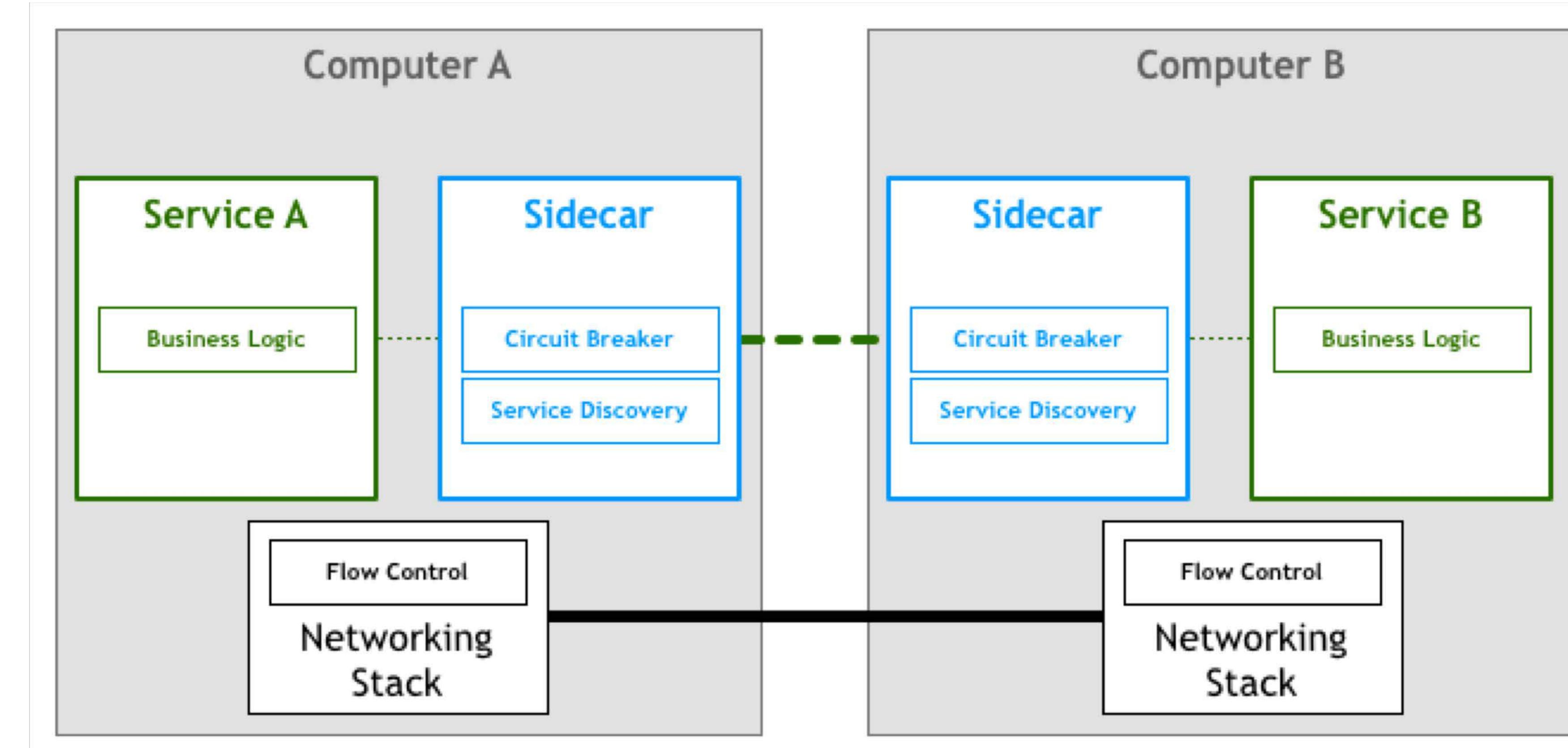
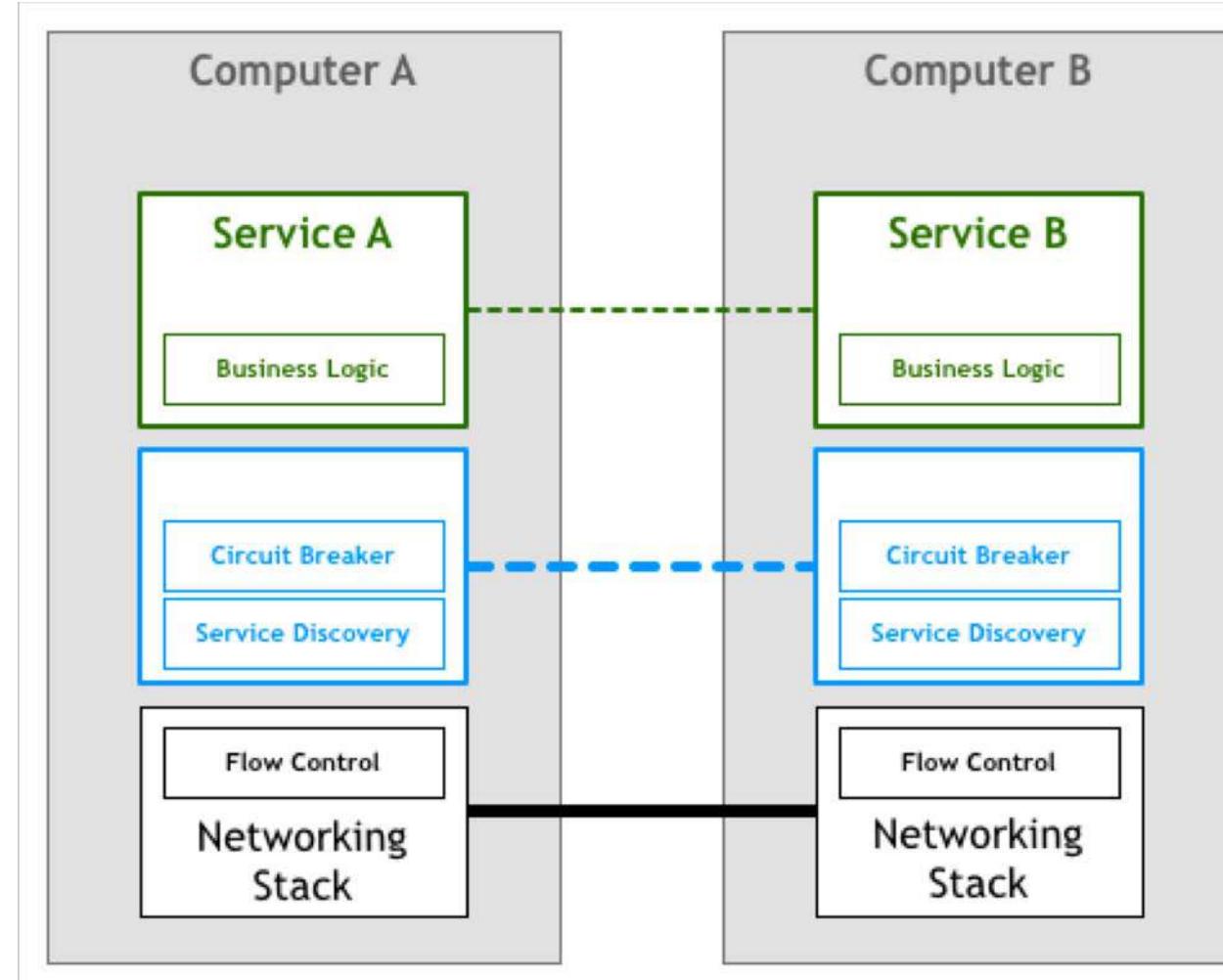
the rise of service meshes



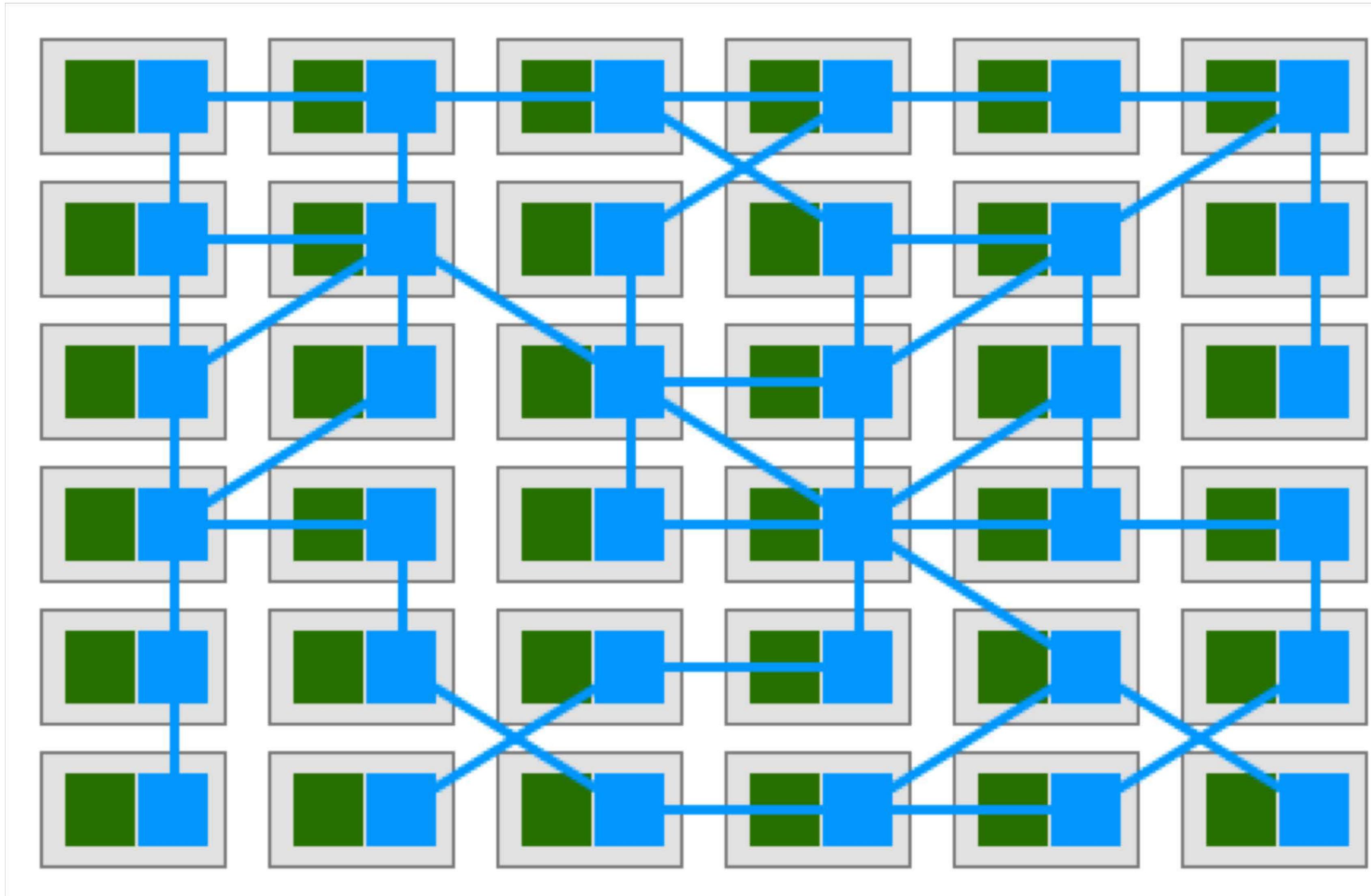
the rise of service meshes



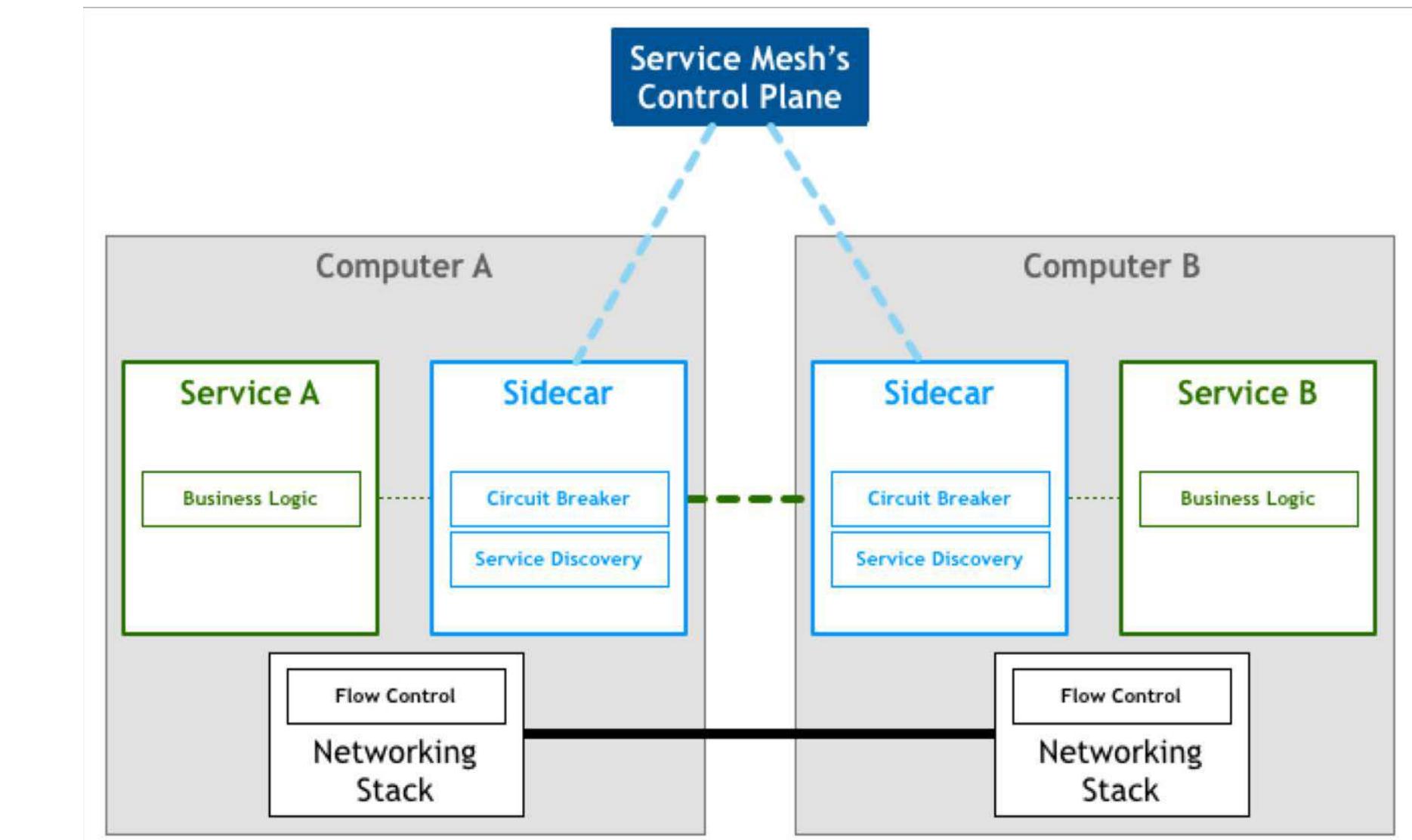
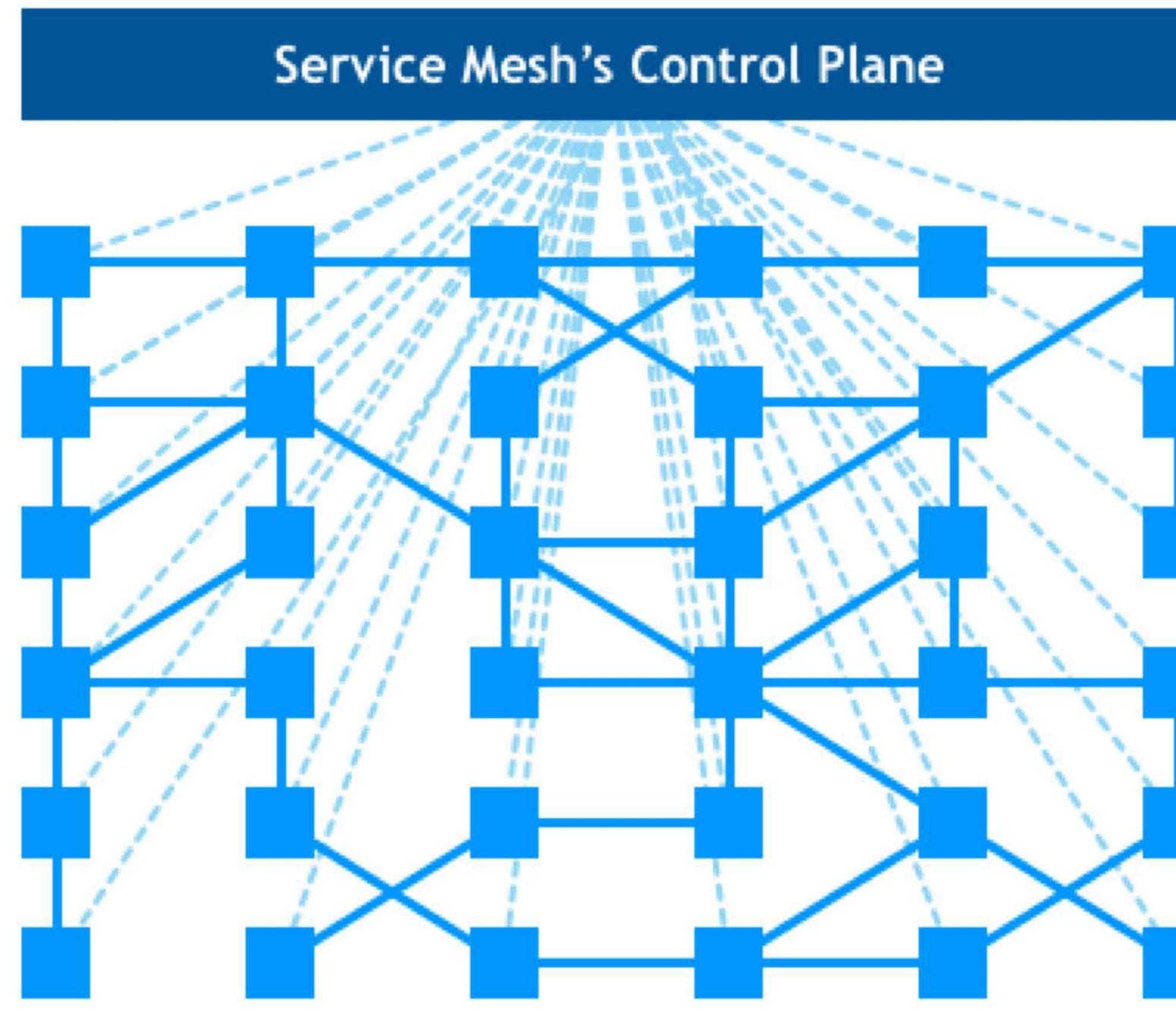
the rise of service meshes



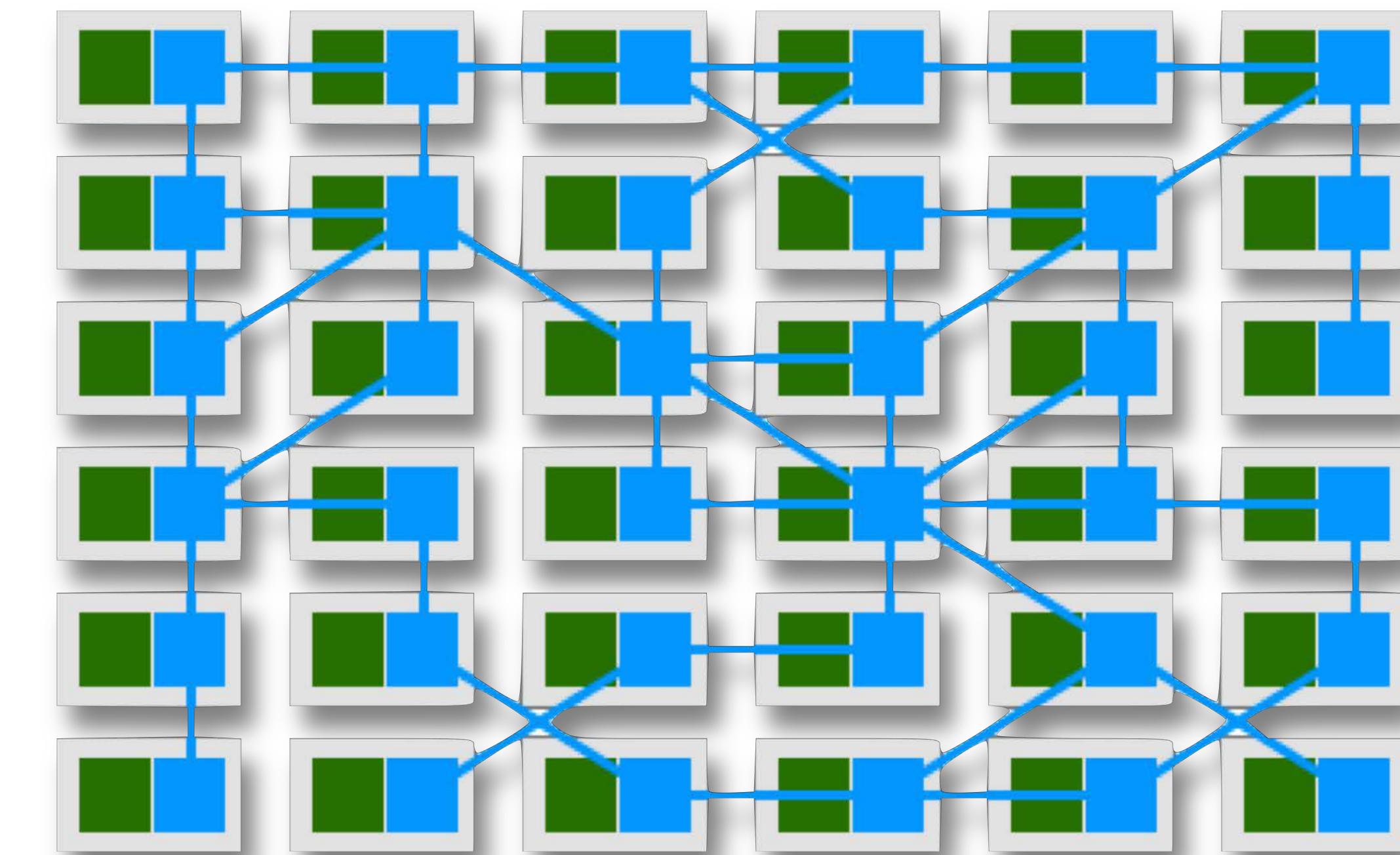
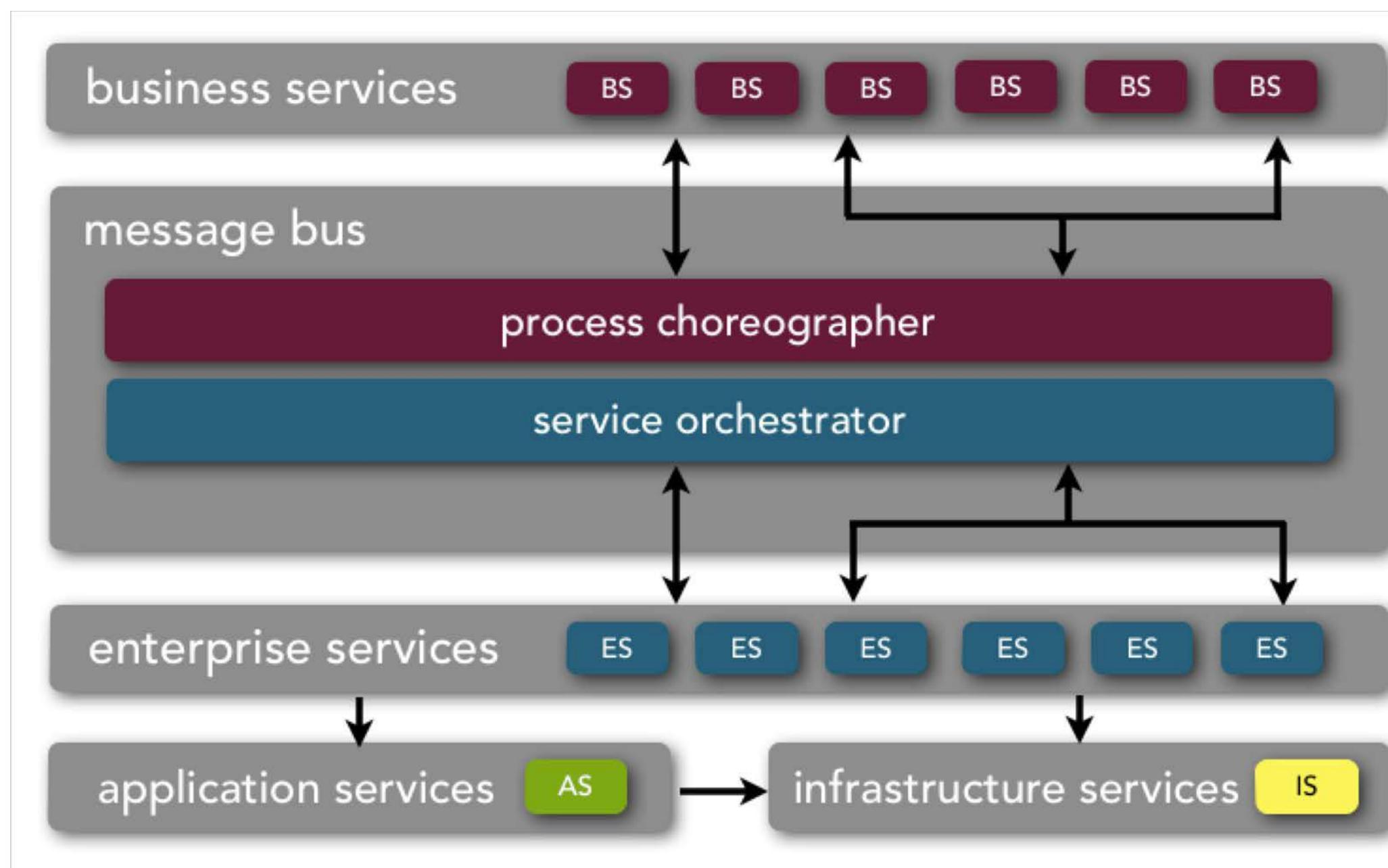
the rise of service meshes



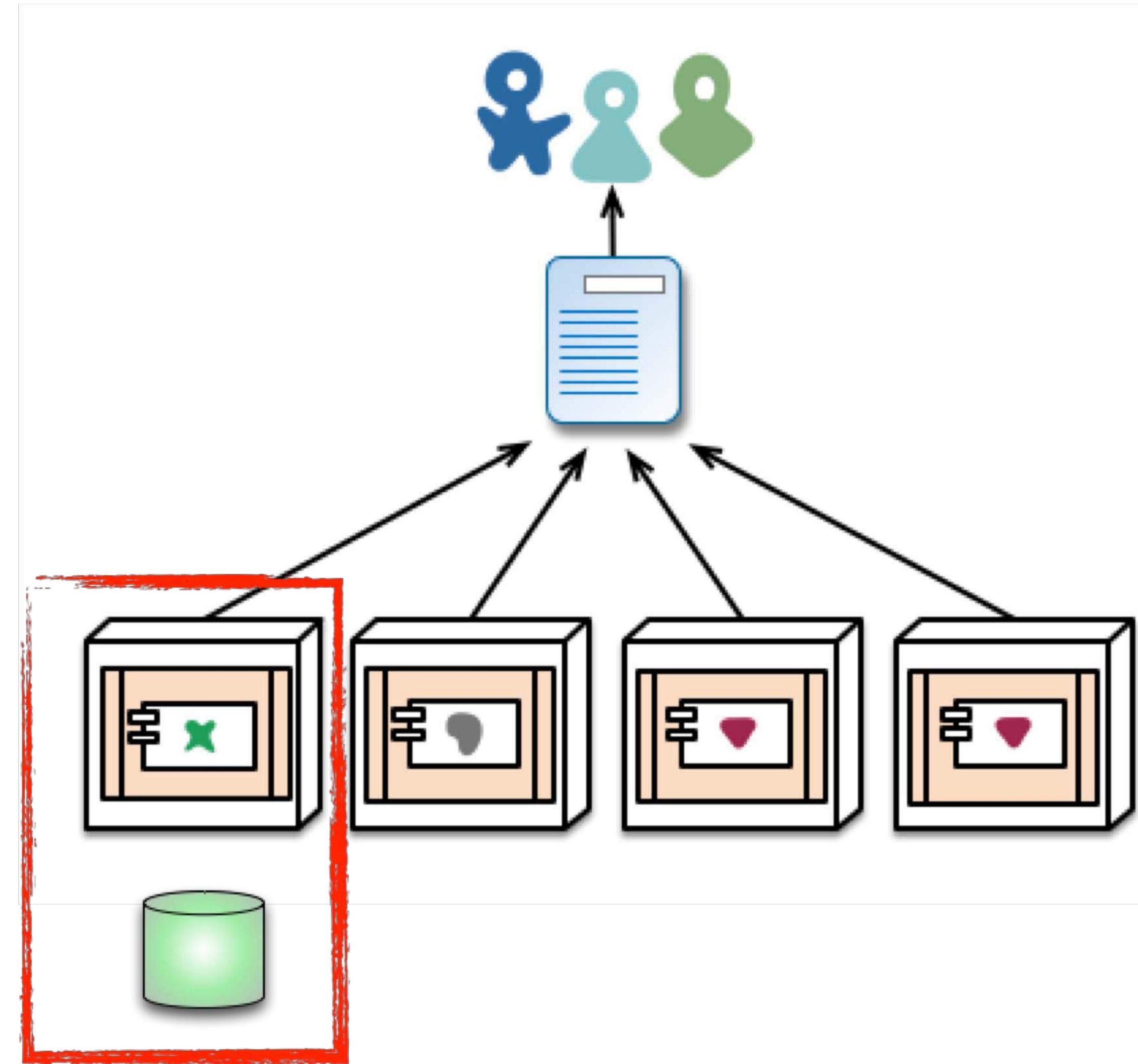
the rise of service meshes



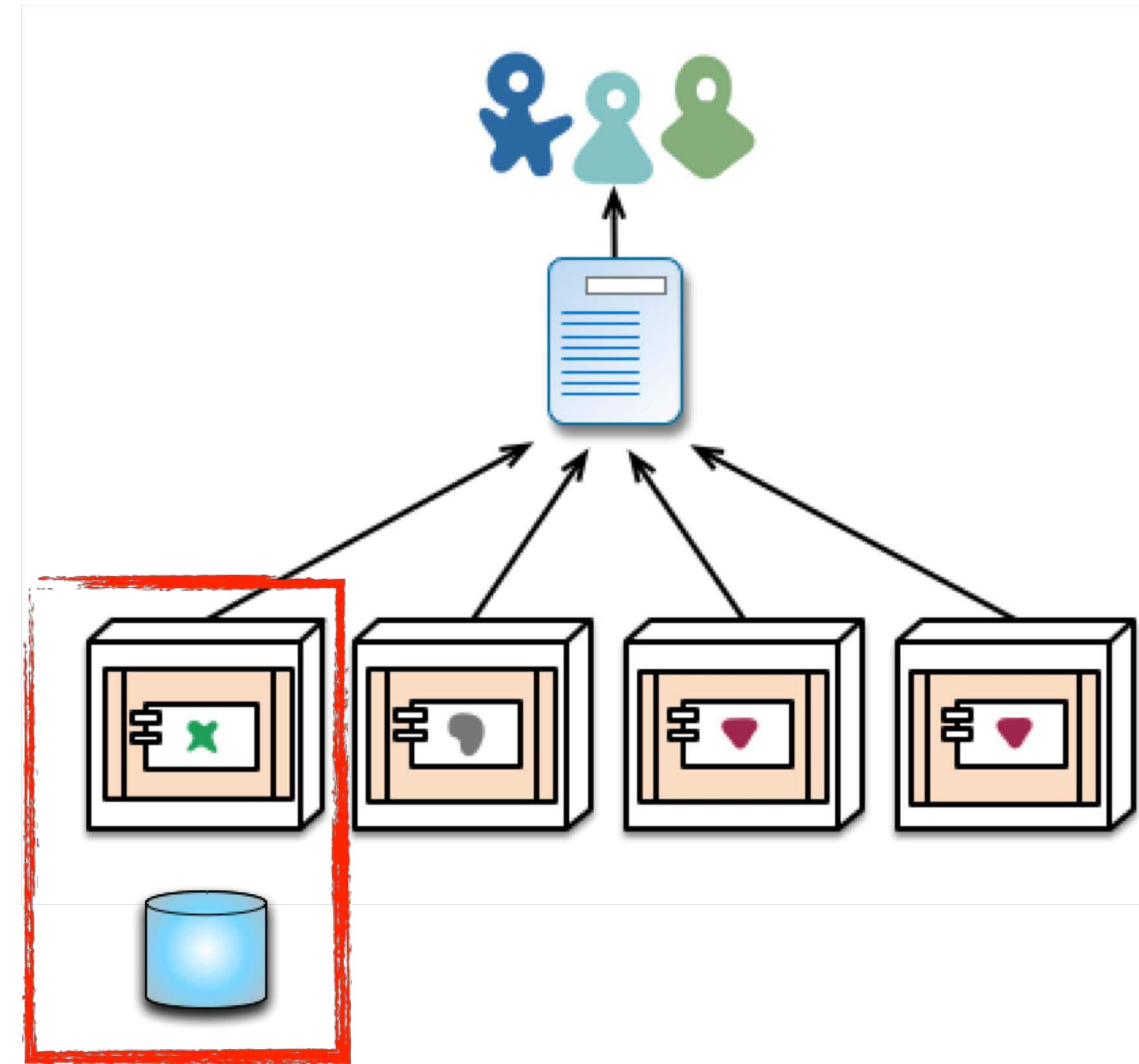
two reuse solutions



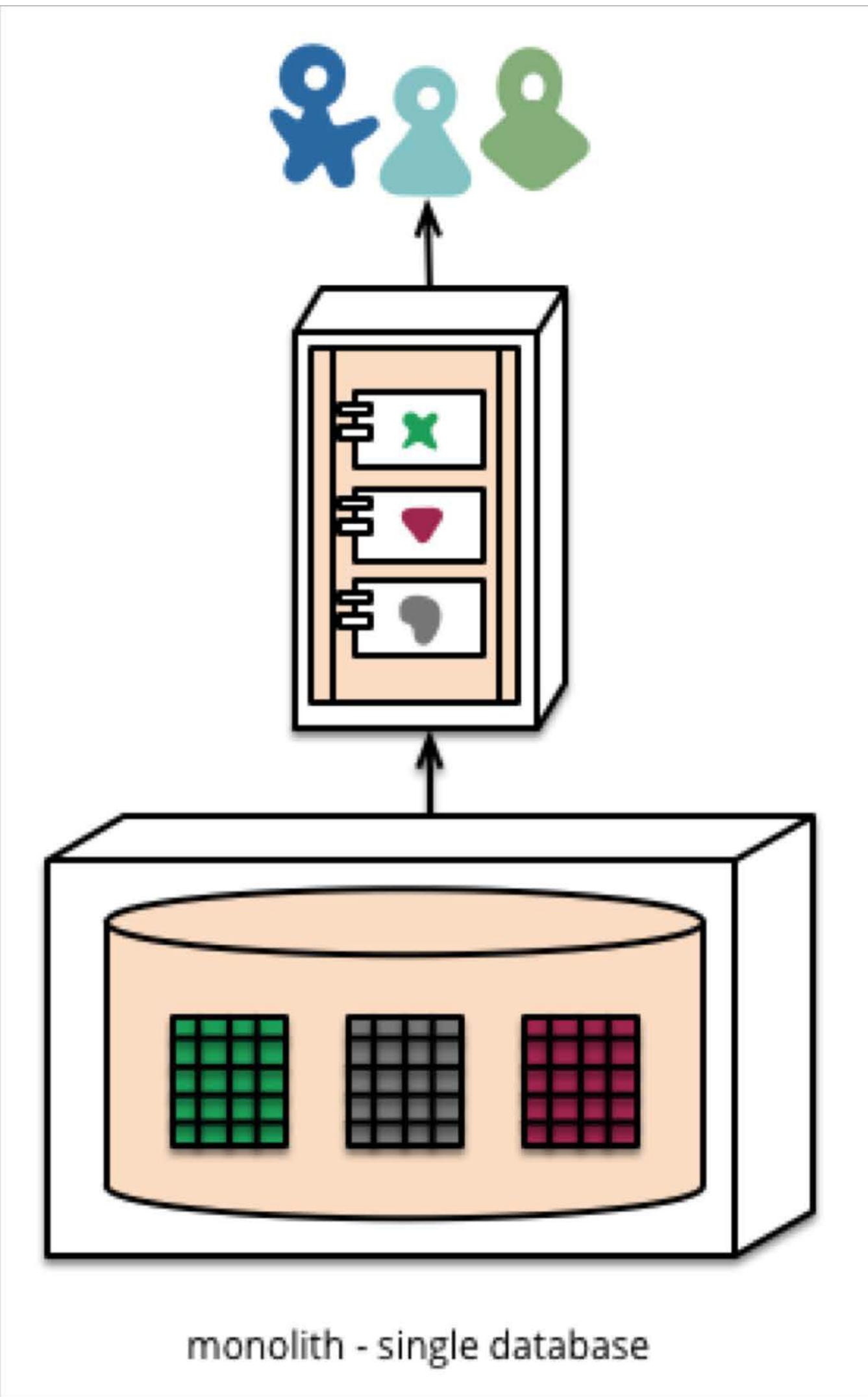
decentralized governance



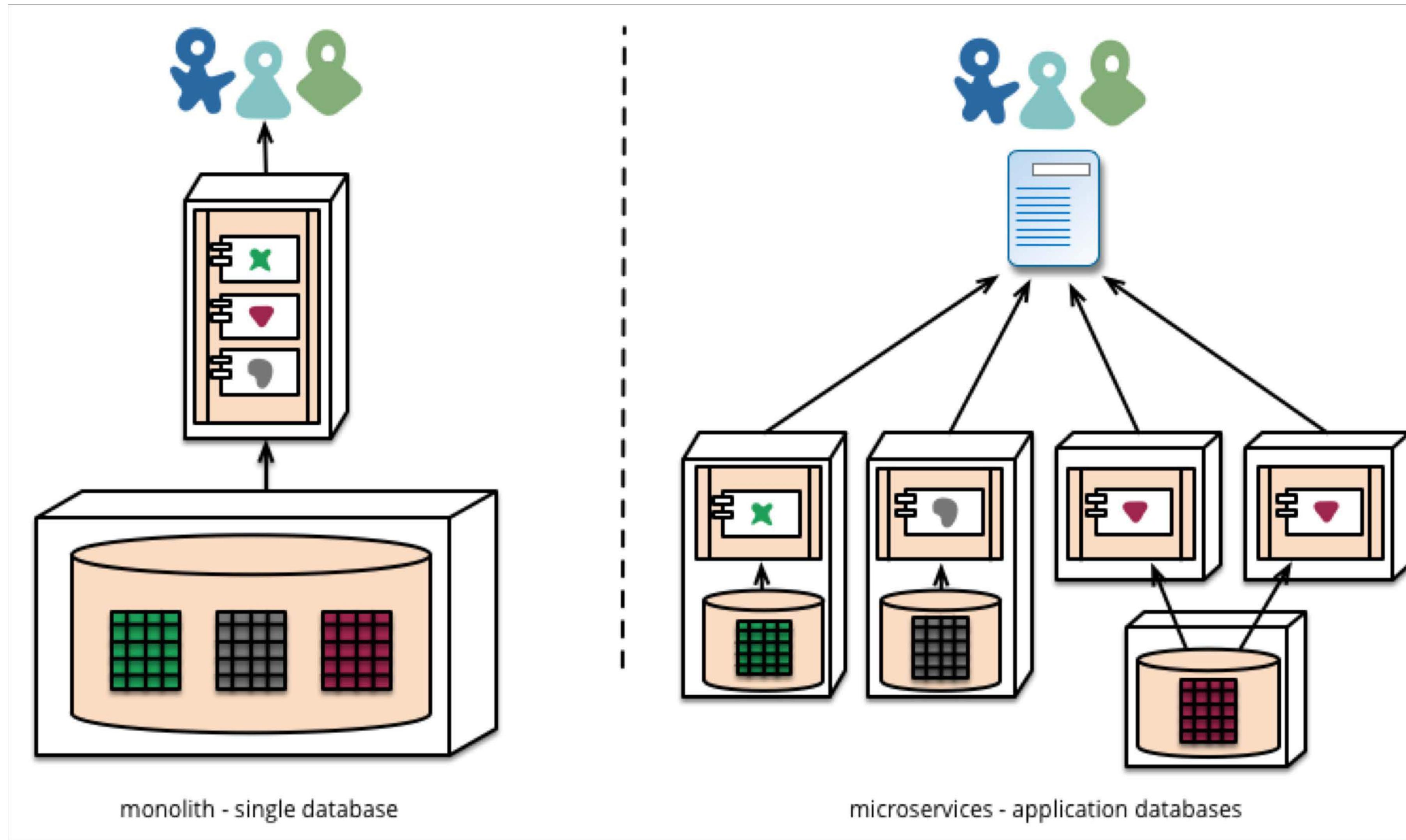
decentralized governance



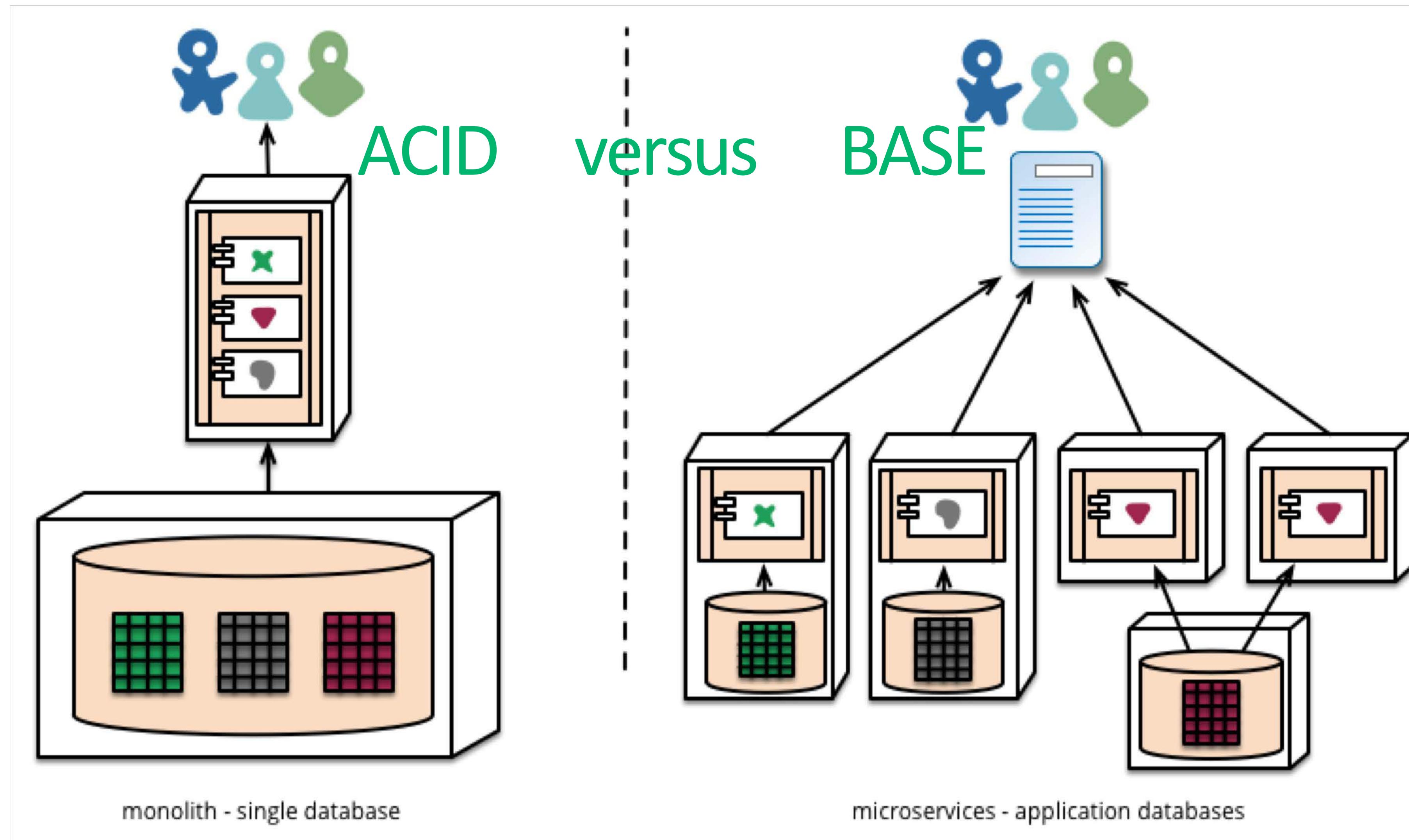
decentralized data management



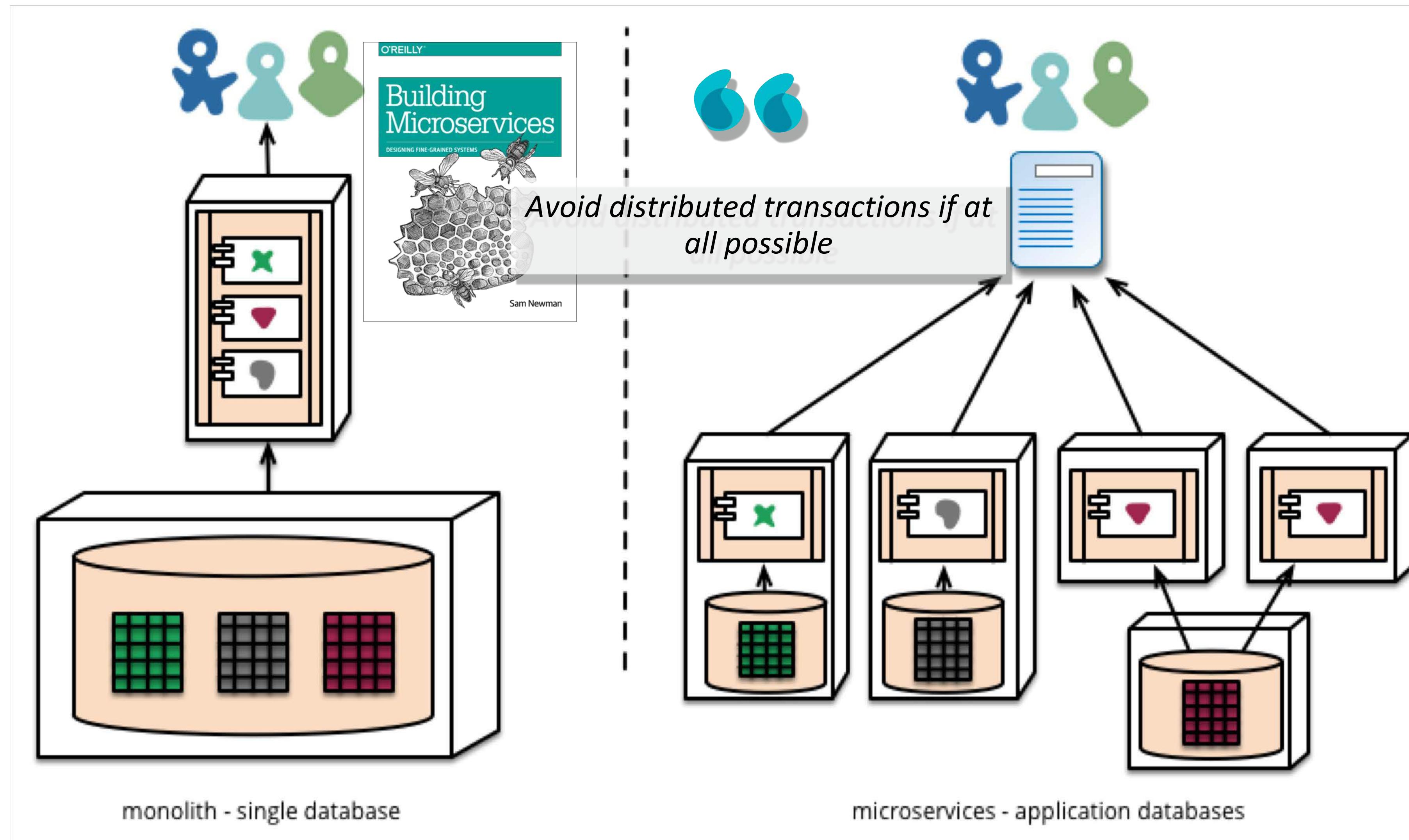
decentralized data management



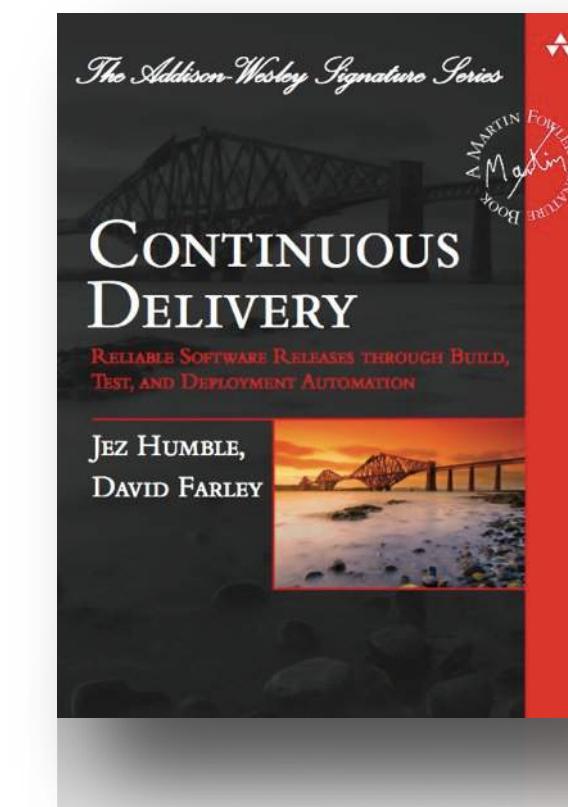
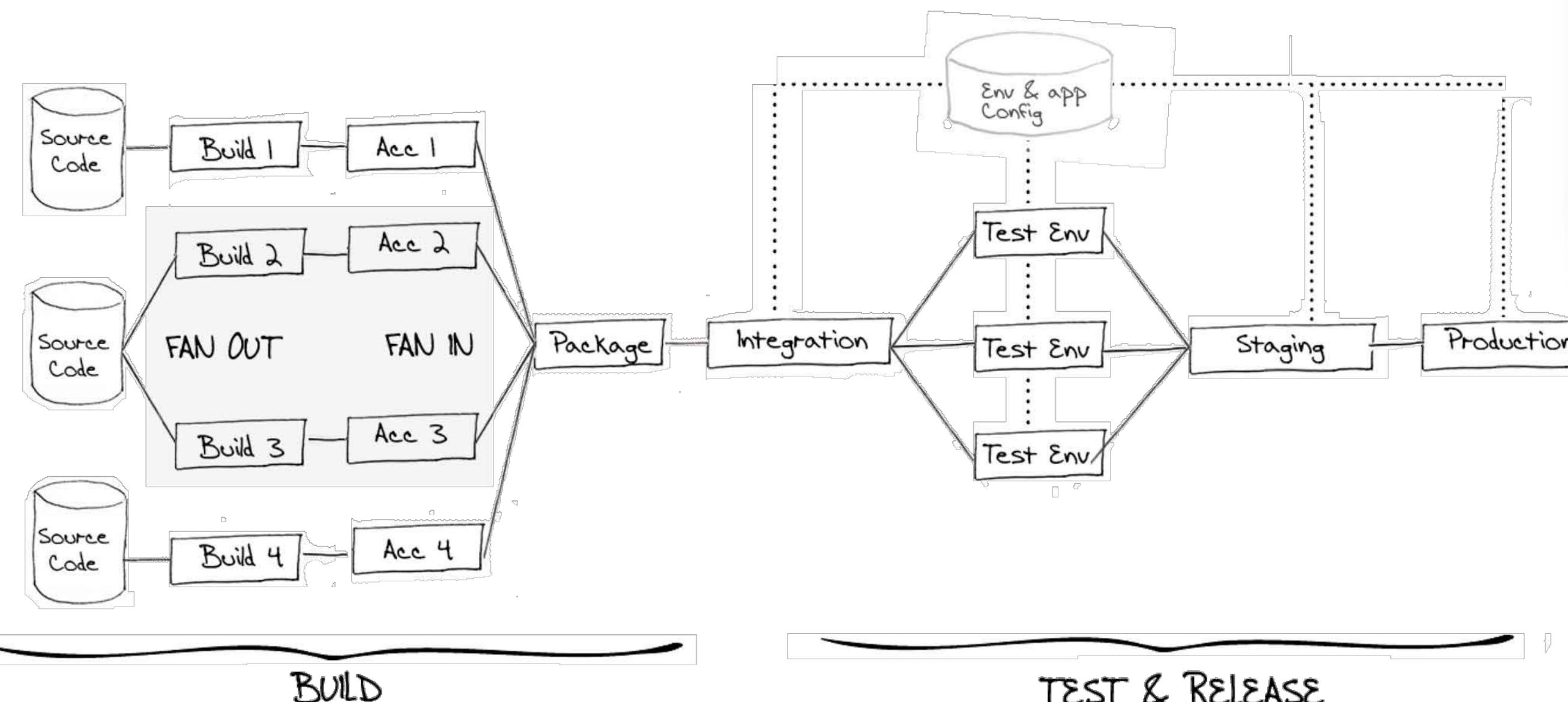
decentralized data management



decentralized data management

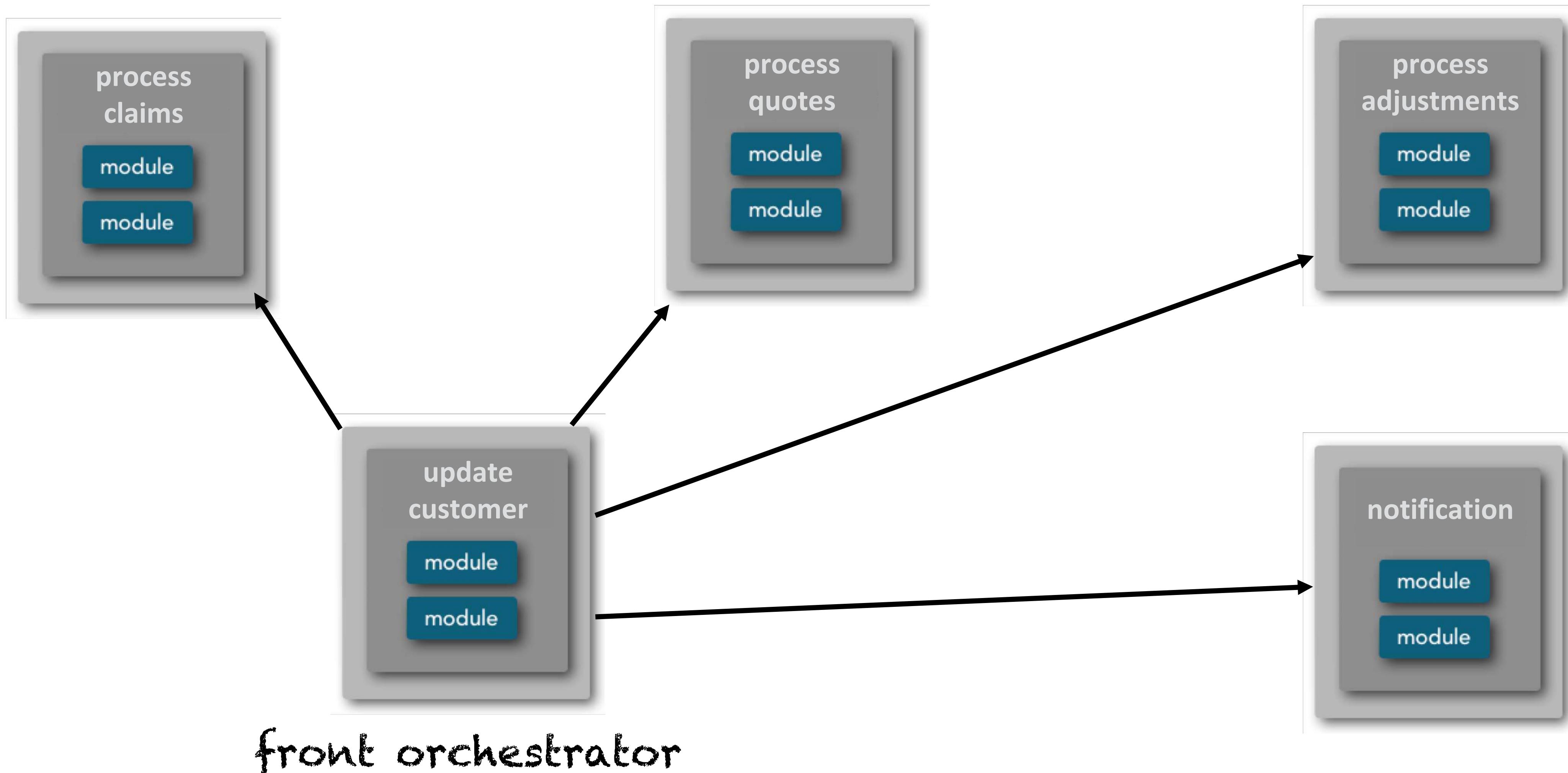


infrastructure automation



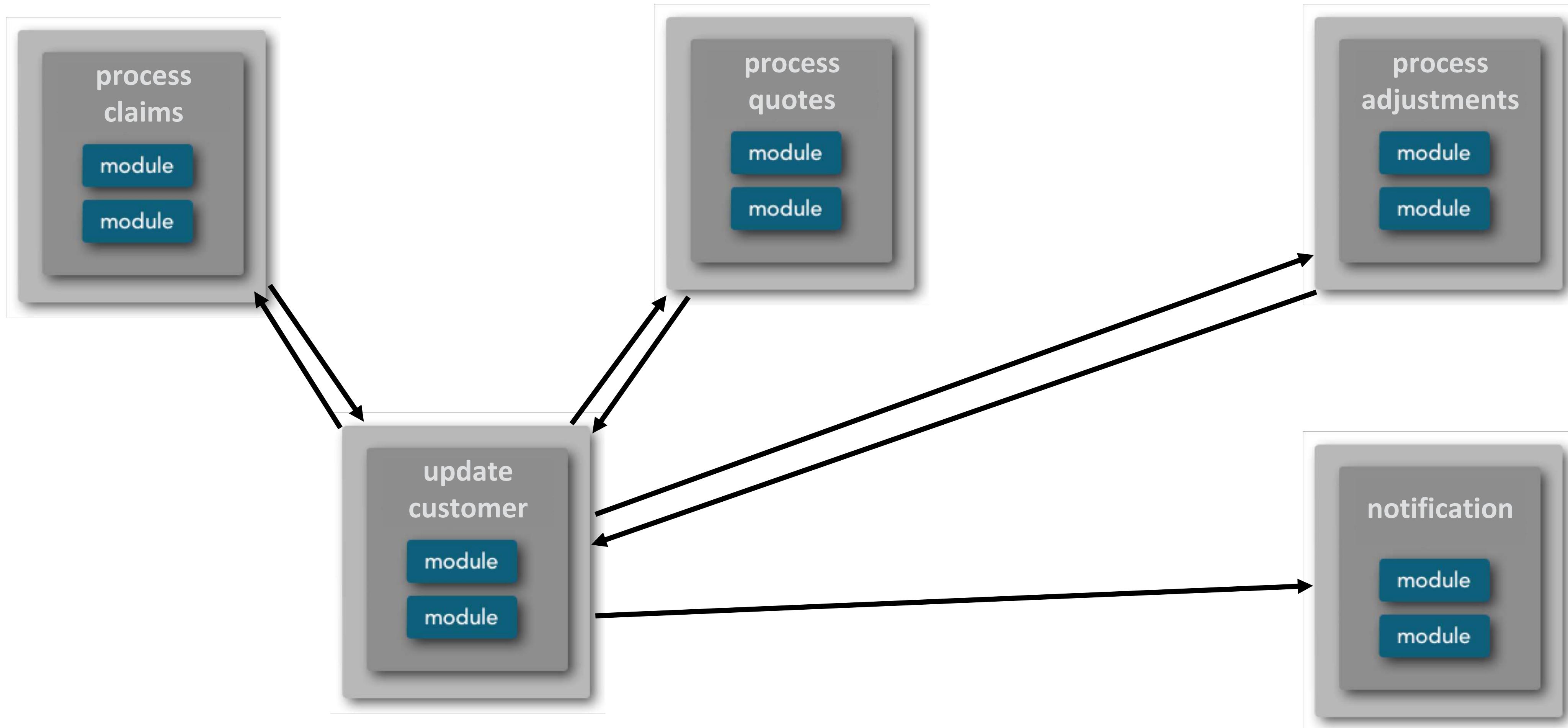
microservices architecture

service orchestration



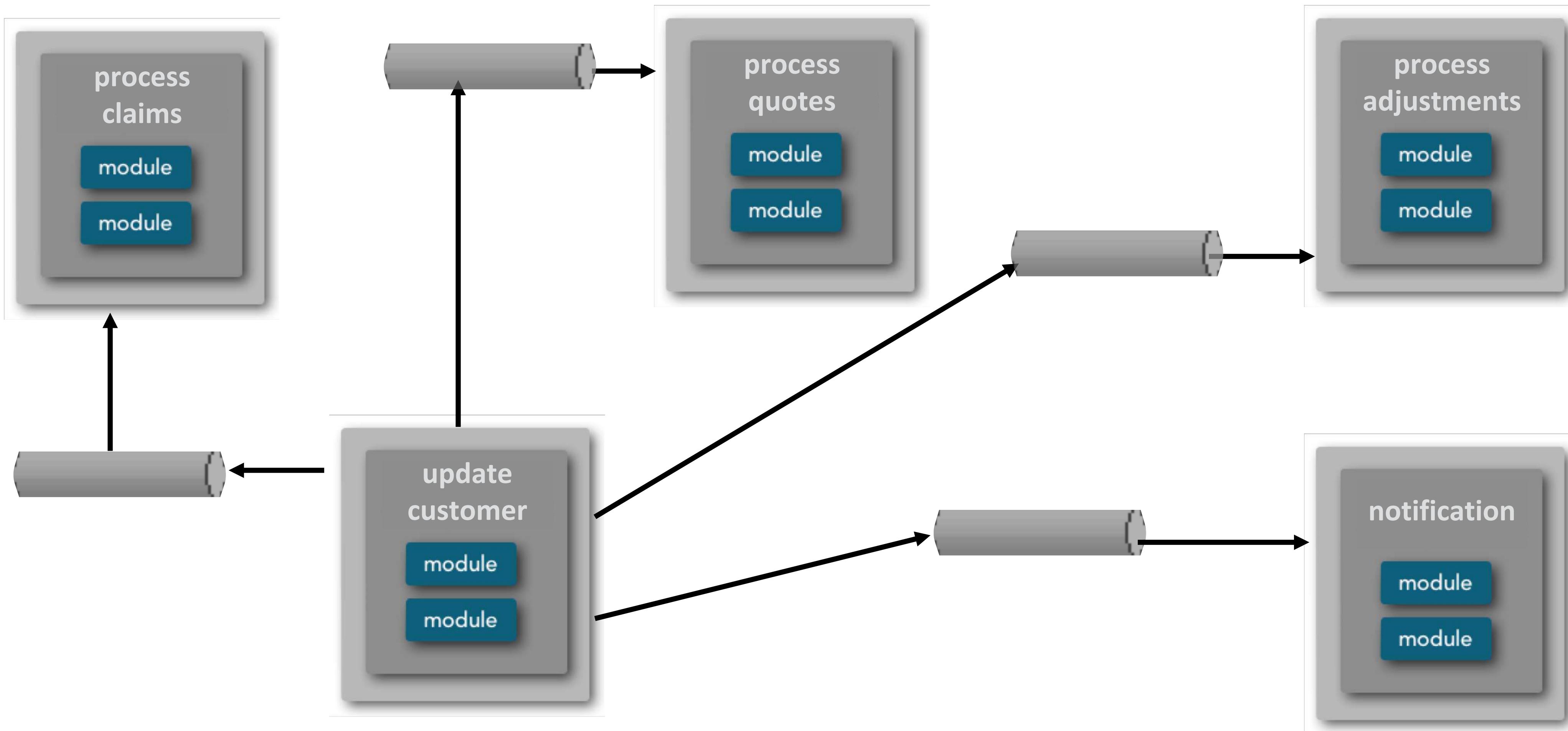
microservices architecture

service orchestration



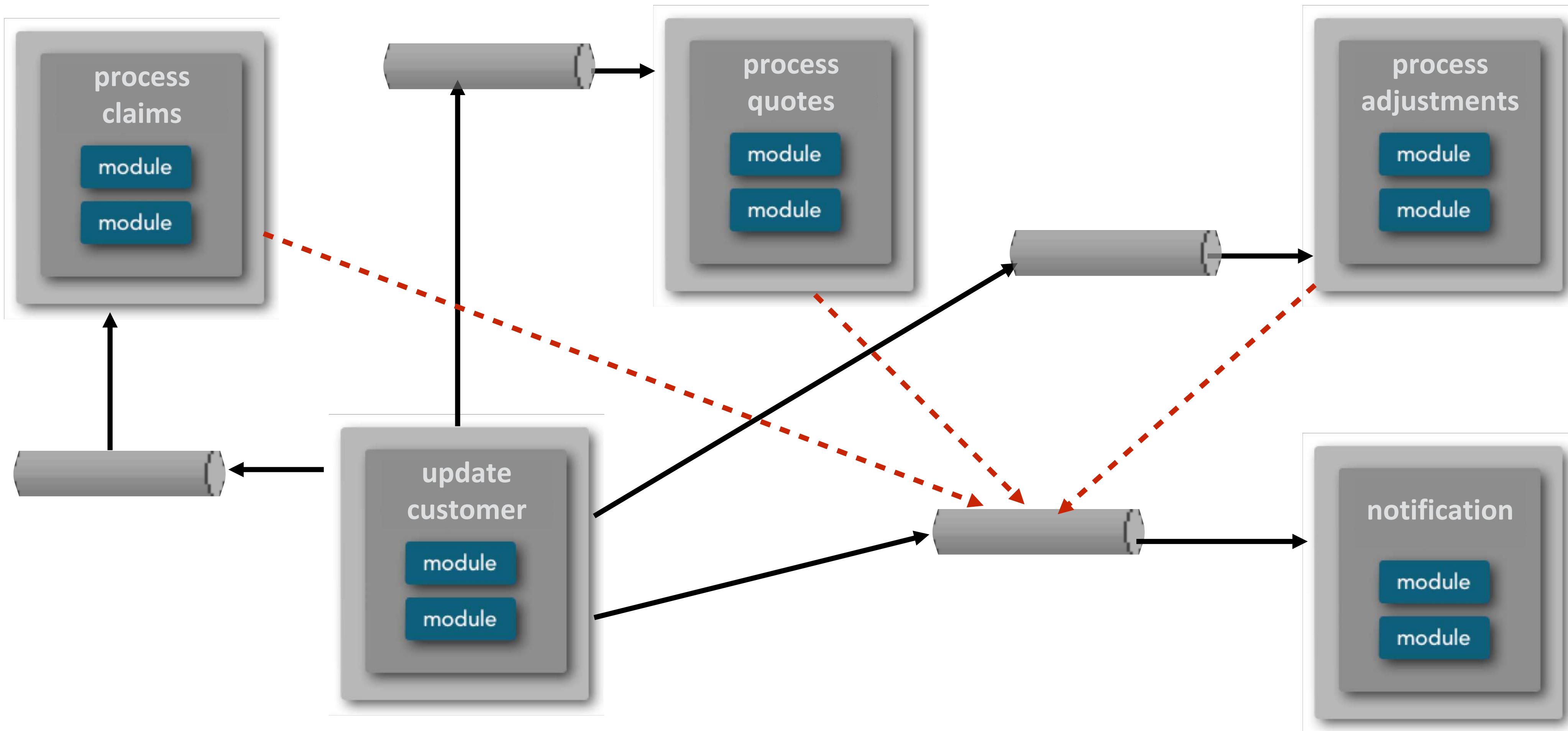
microservices architecture

service orchestration



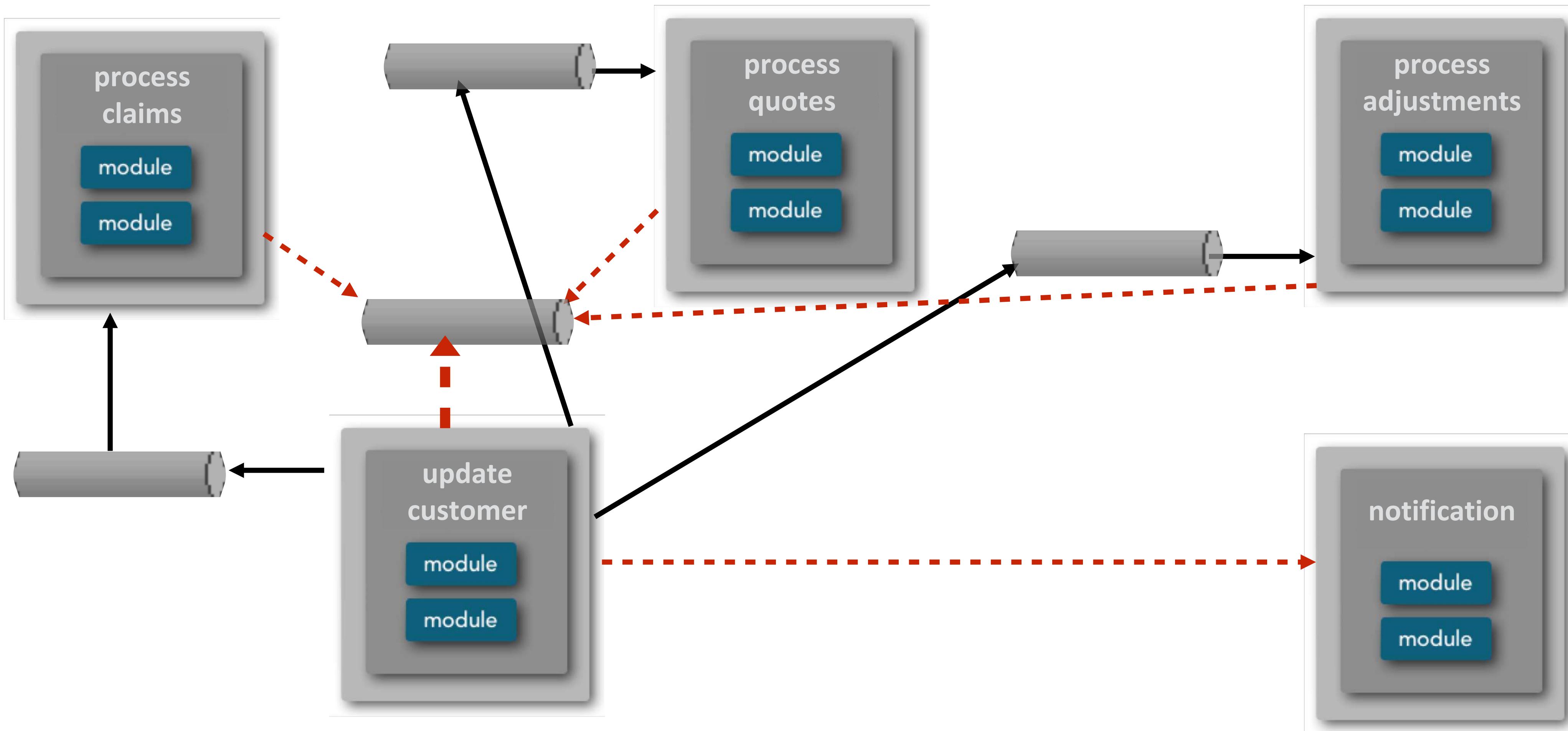
microservices architecture

service orchestration

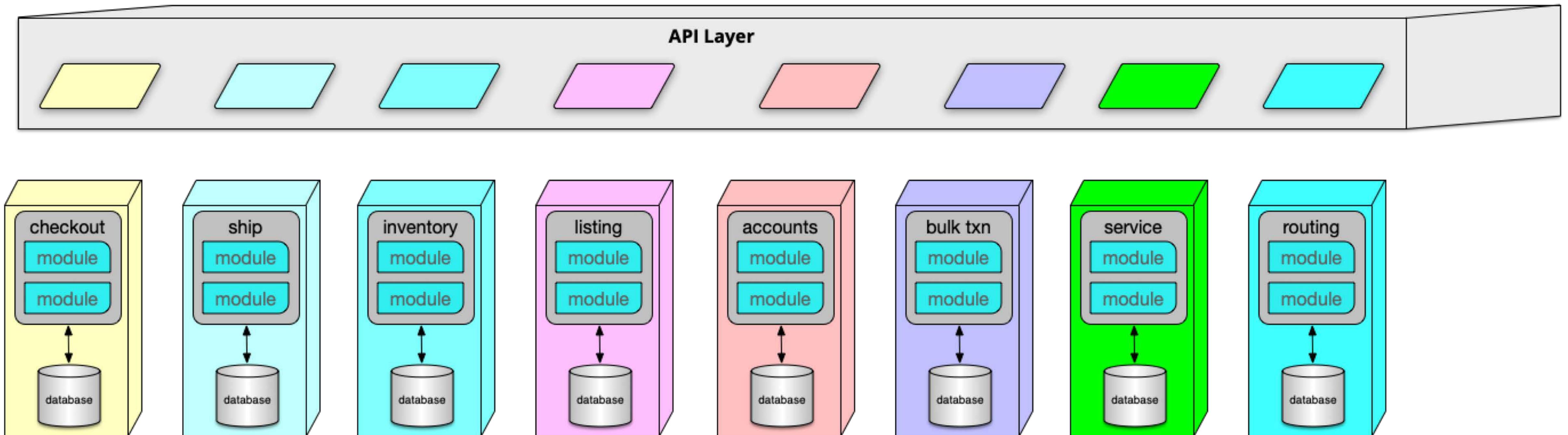


microservices architecture

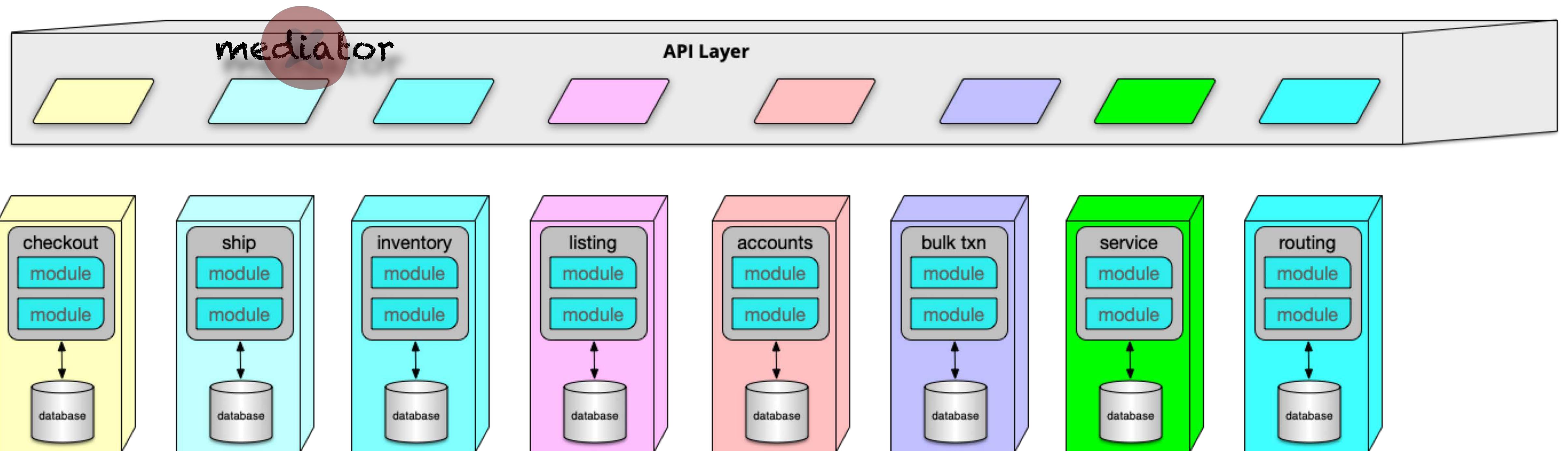
service orchestration



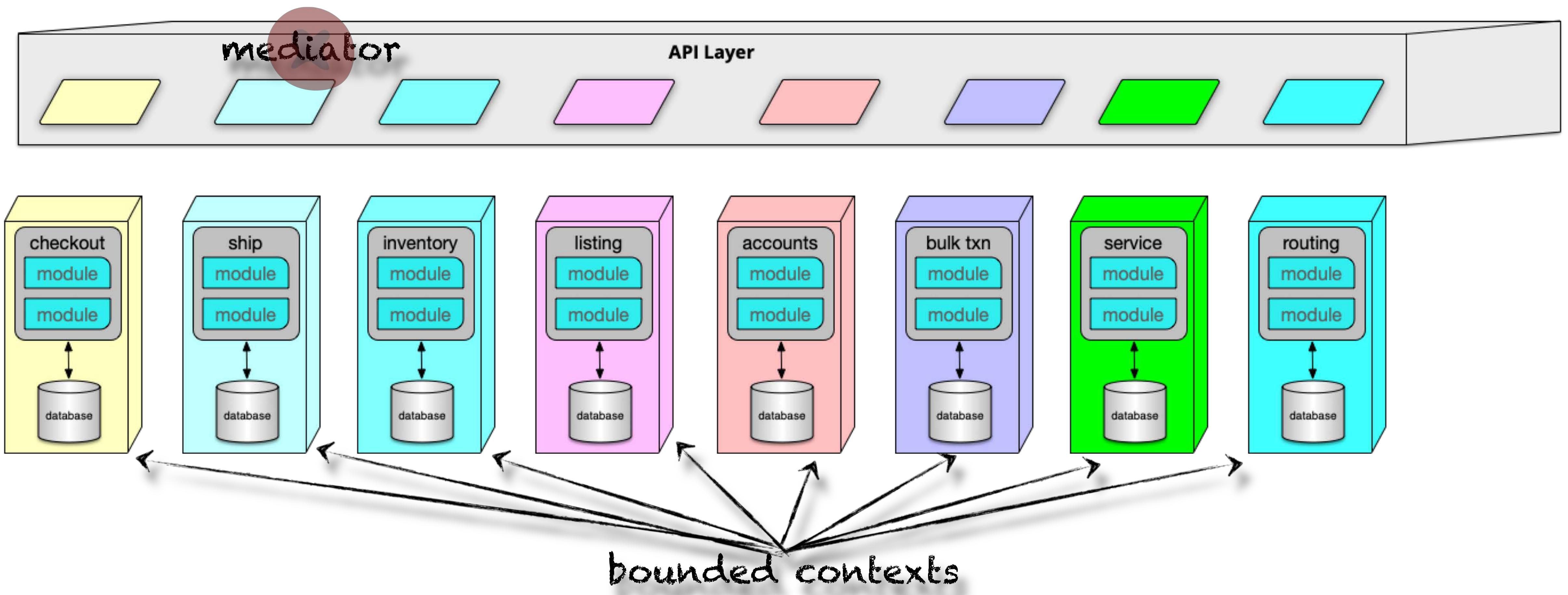
microservices



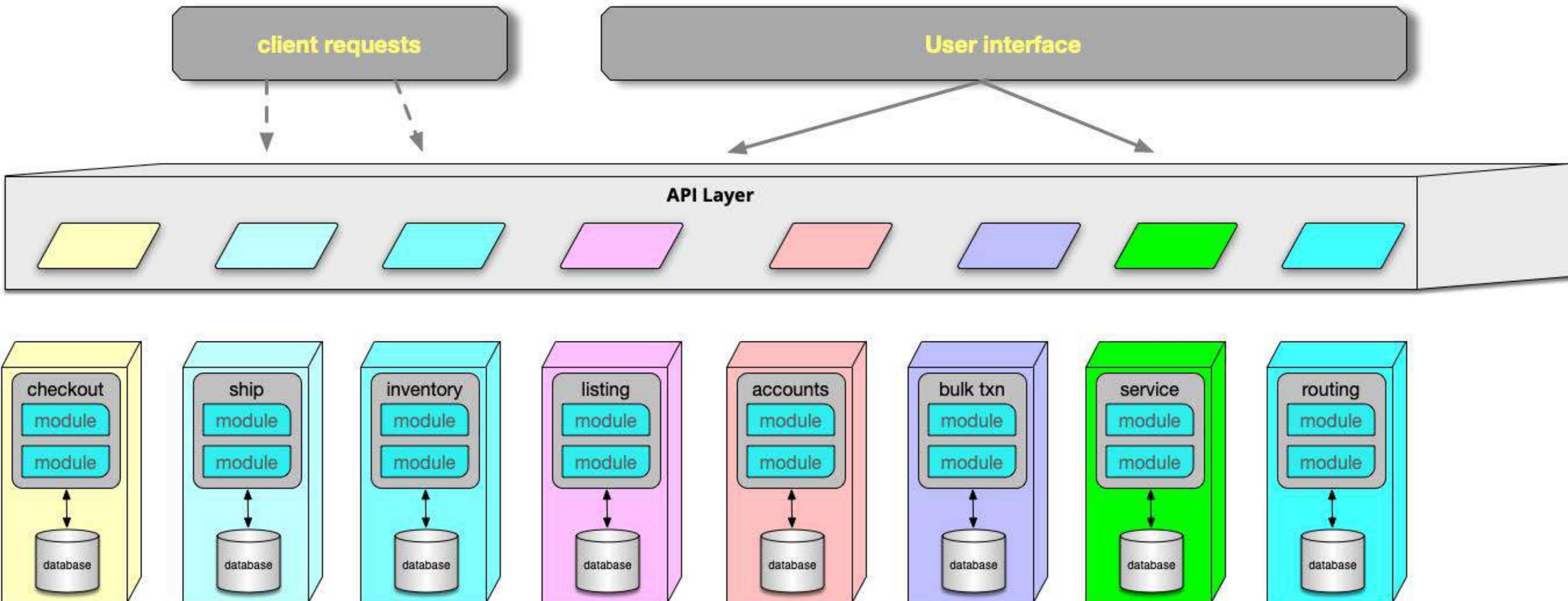
microservices



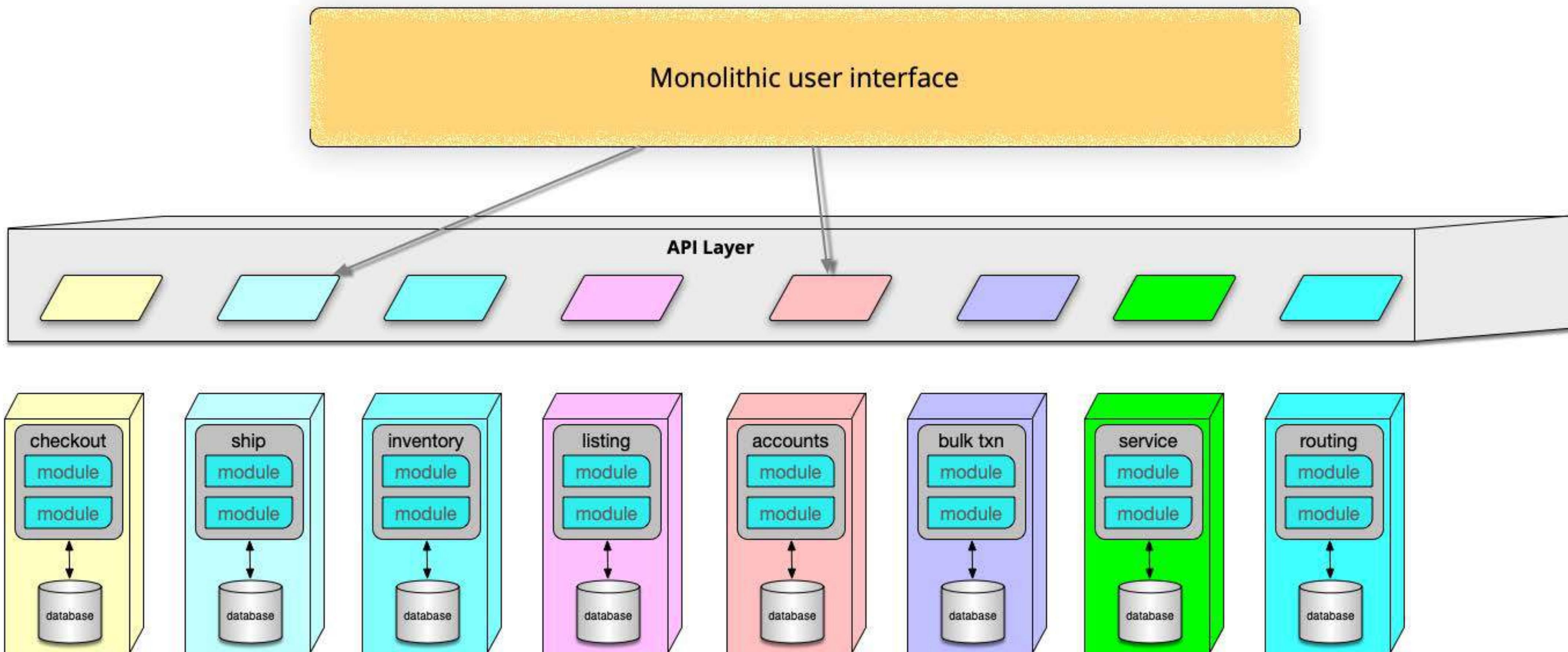
microservices



microservices

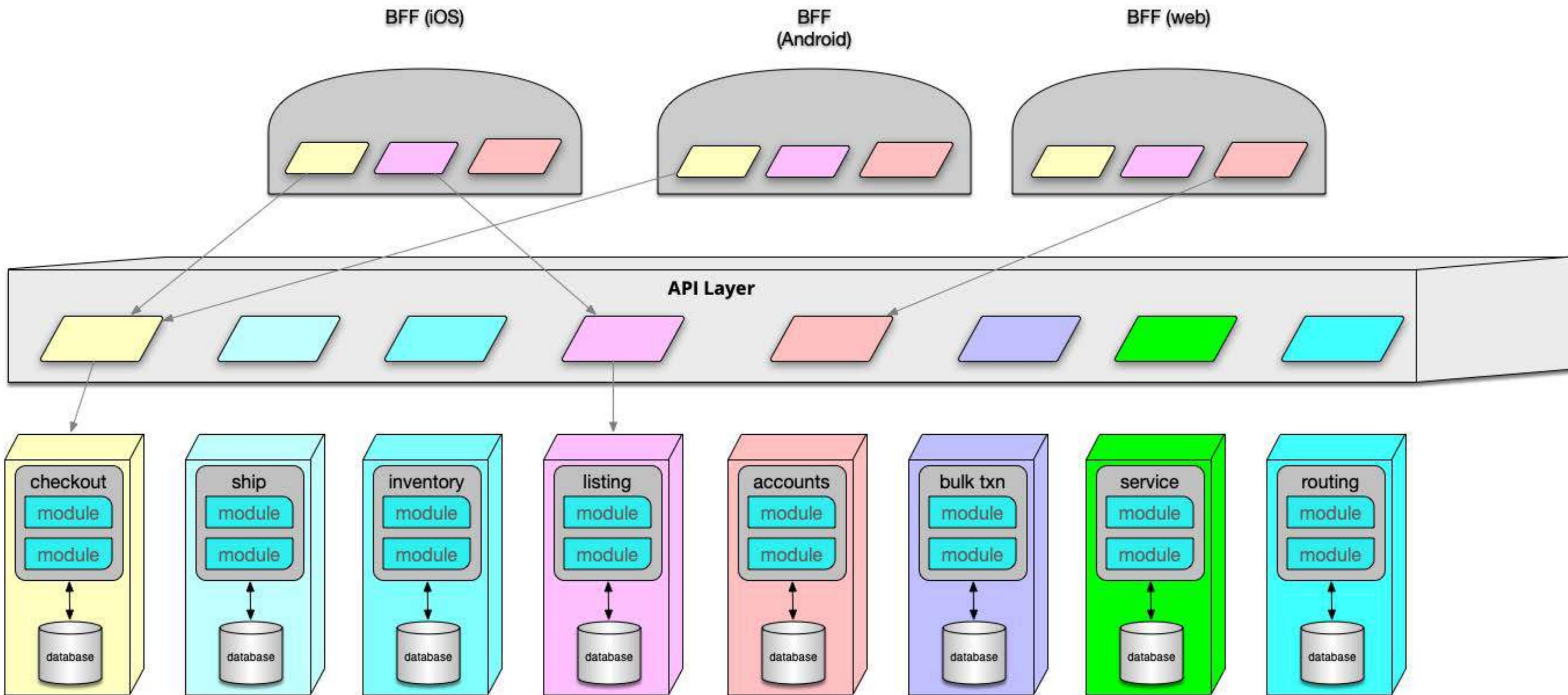


microservices

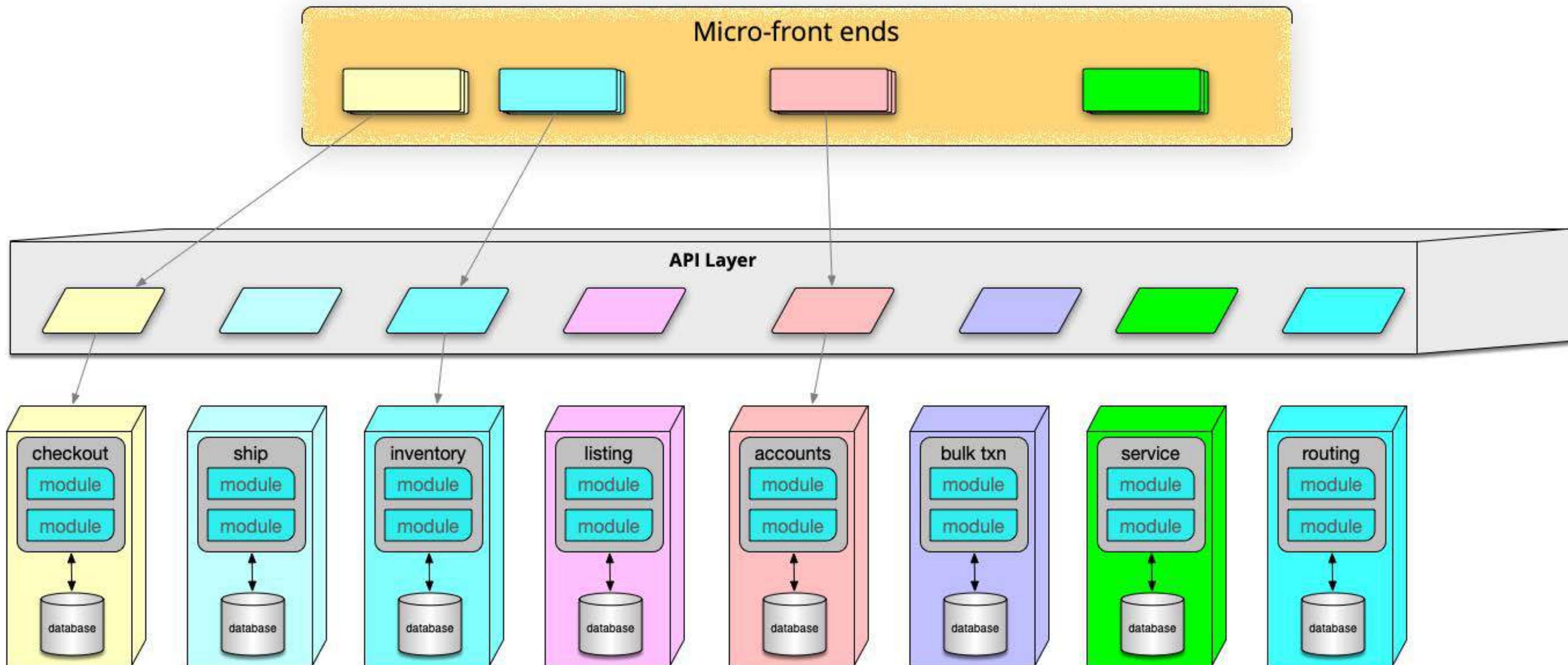


microservices: BFF

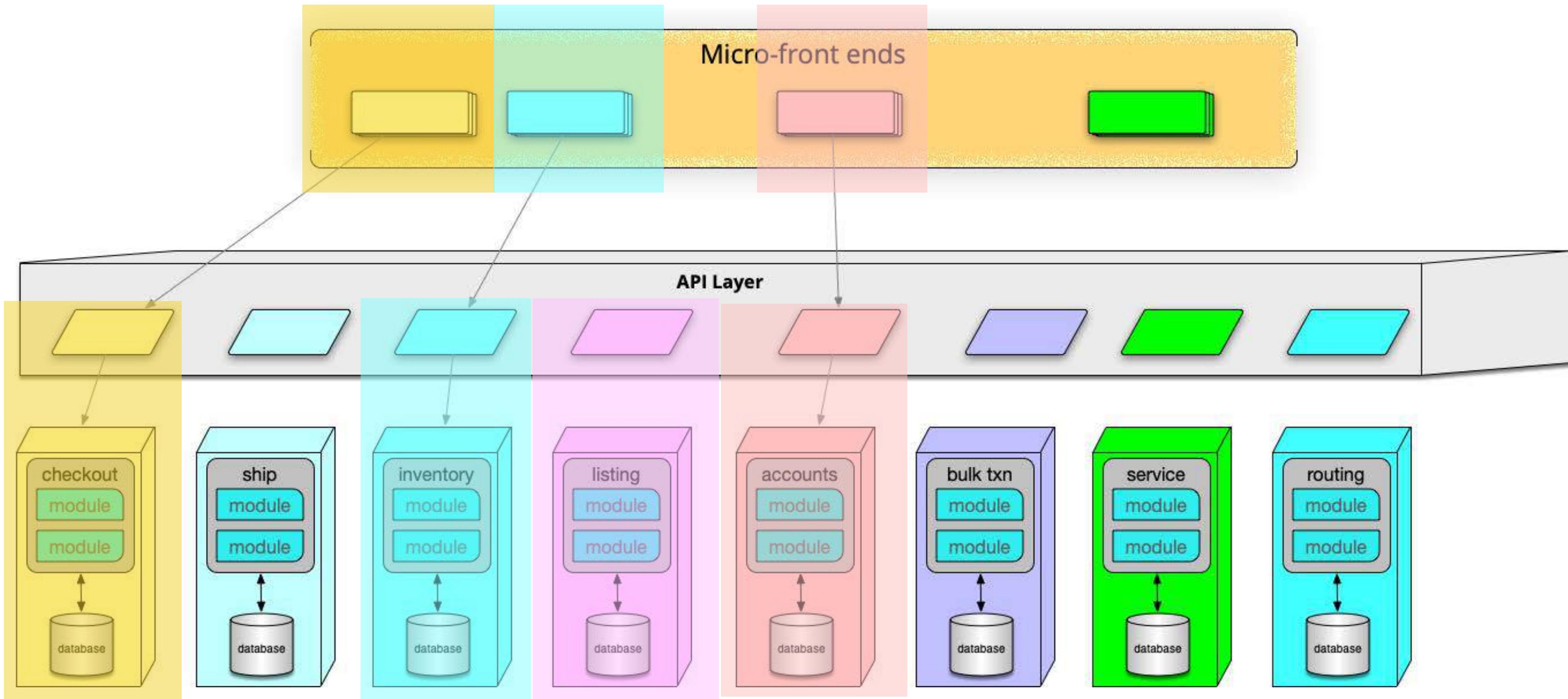
<https://samnewman.io/patterns/architectural/bff/>

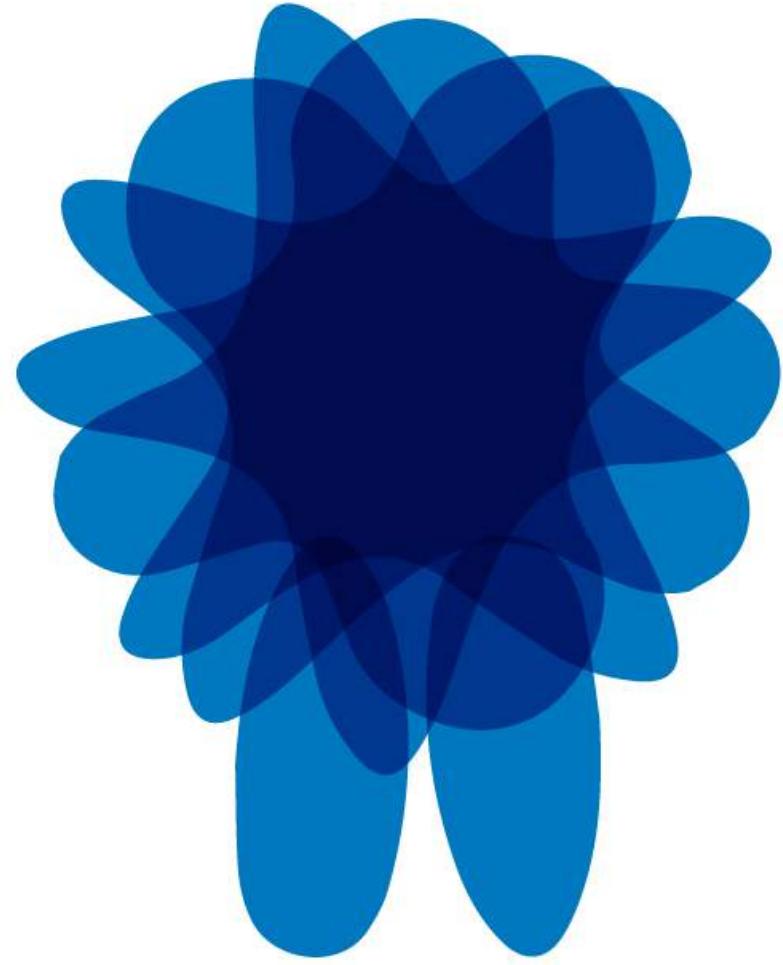


microservices



microservices quanta

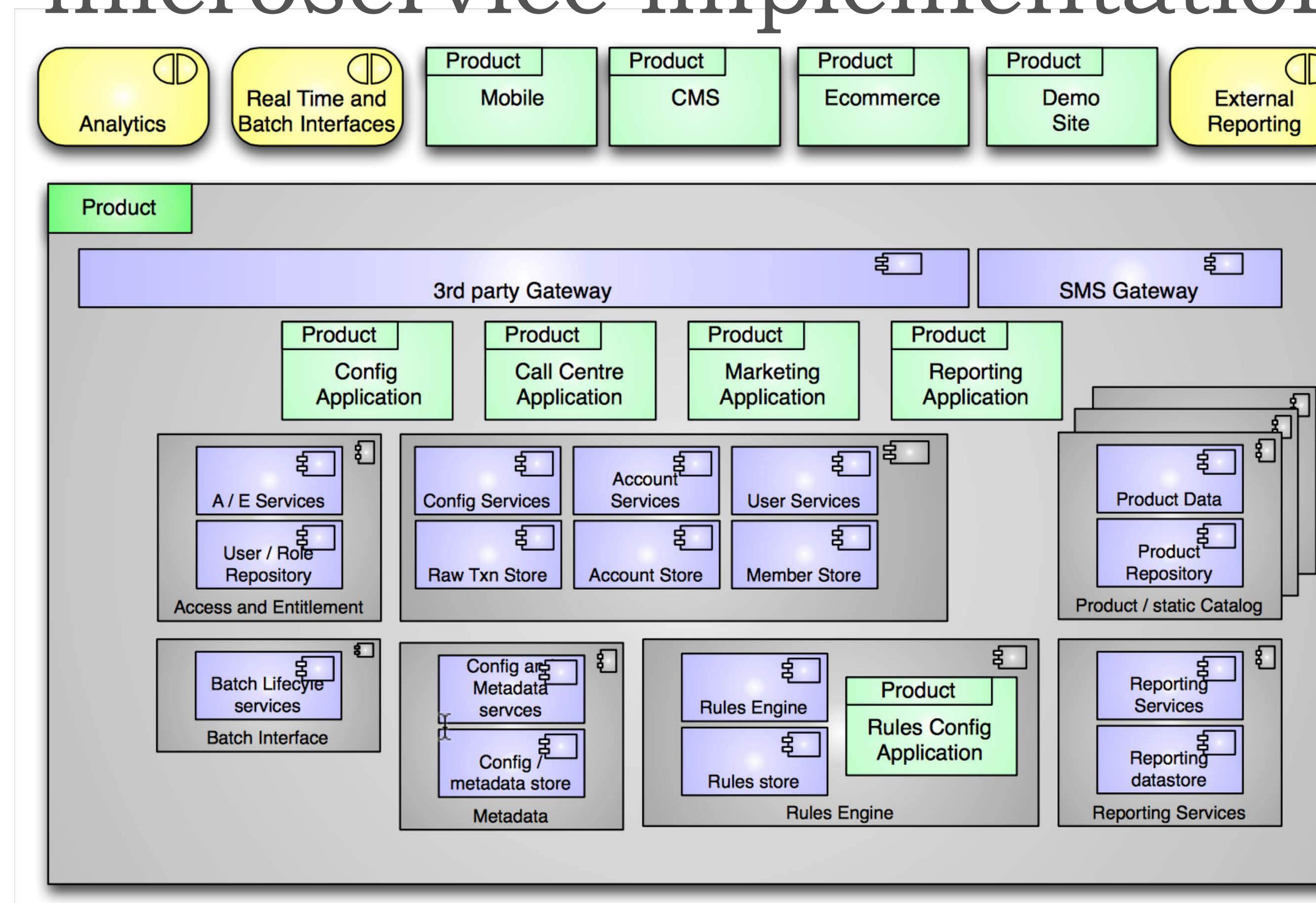




Support ▲

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

microservice implementation



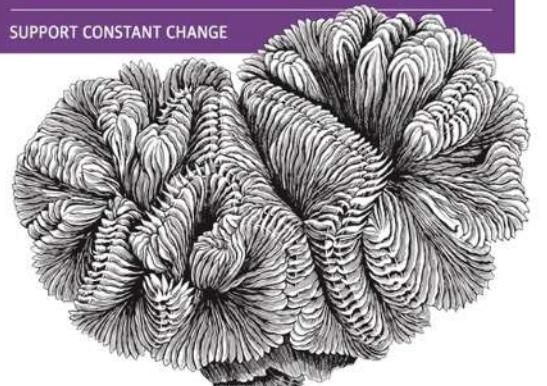
<http://2012.33degree.org/pdf/JamesLewisMicroServices.pdf>

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

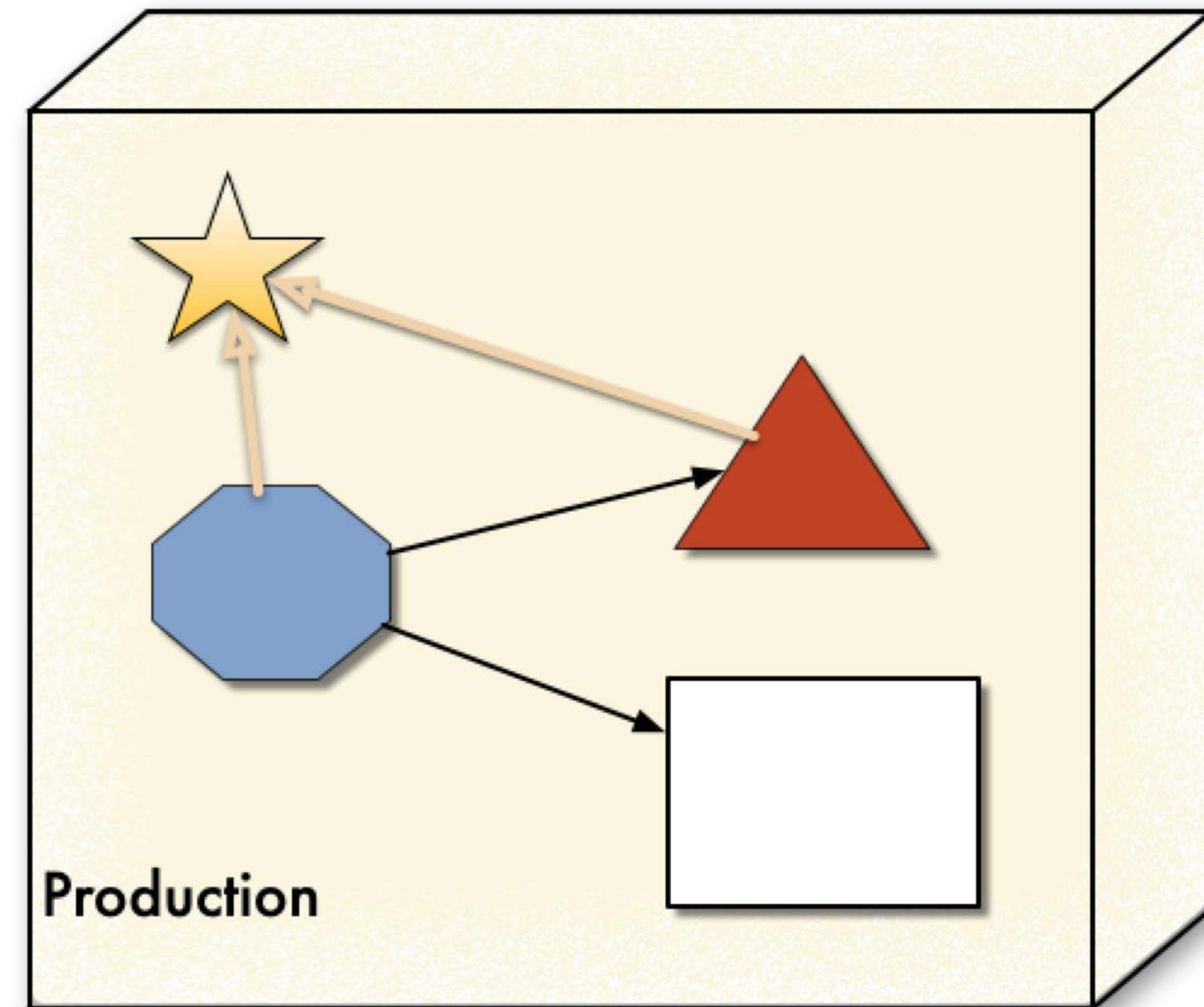
O'REILLY®

Building Evolutionary Architectures

SUPPORT CONSTANT CHANGE

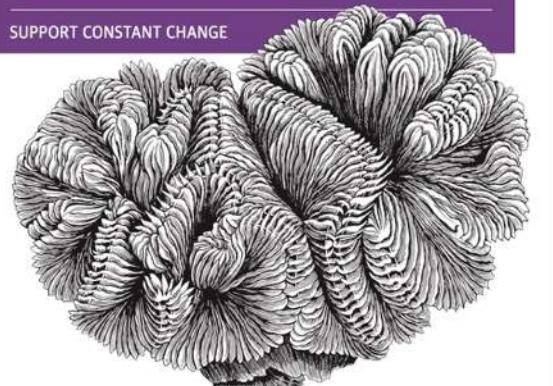


Neal Ford, Rebecca Parsons & Patrick Kua

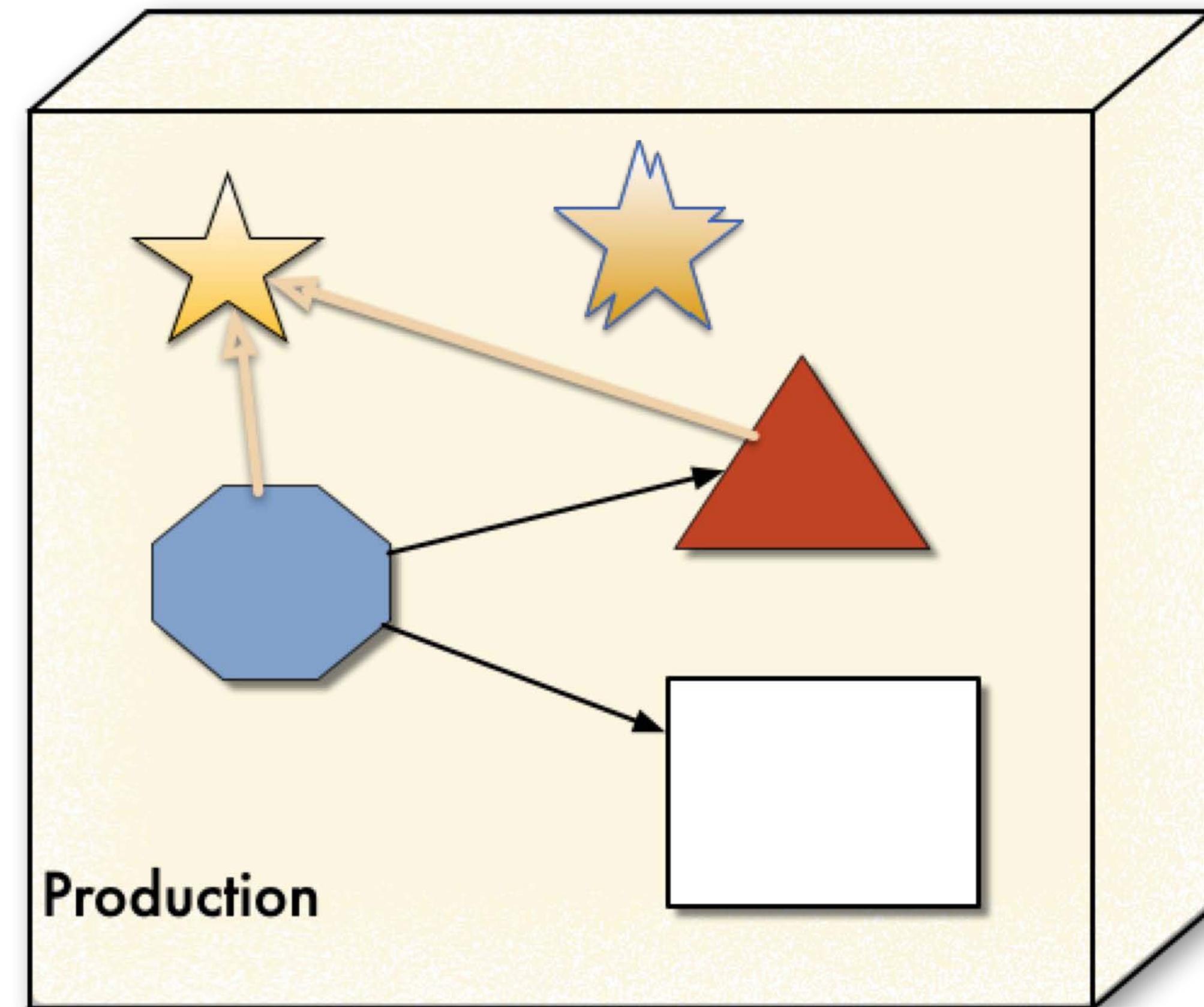


Building Evolutionary Architectures

SUPPORT CONSTANT CHANGE



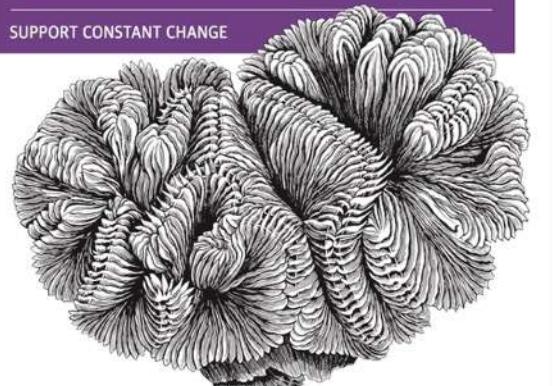
Neal Ford, Rebecca Parsons & Patrick Kua



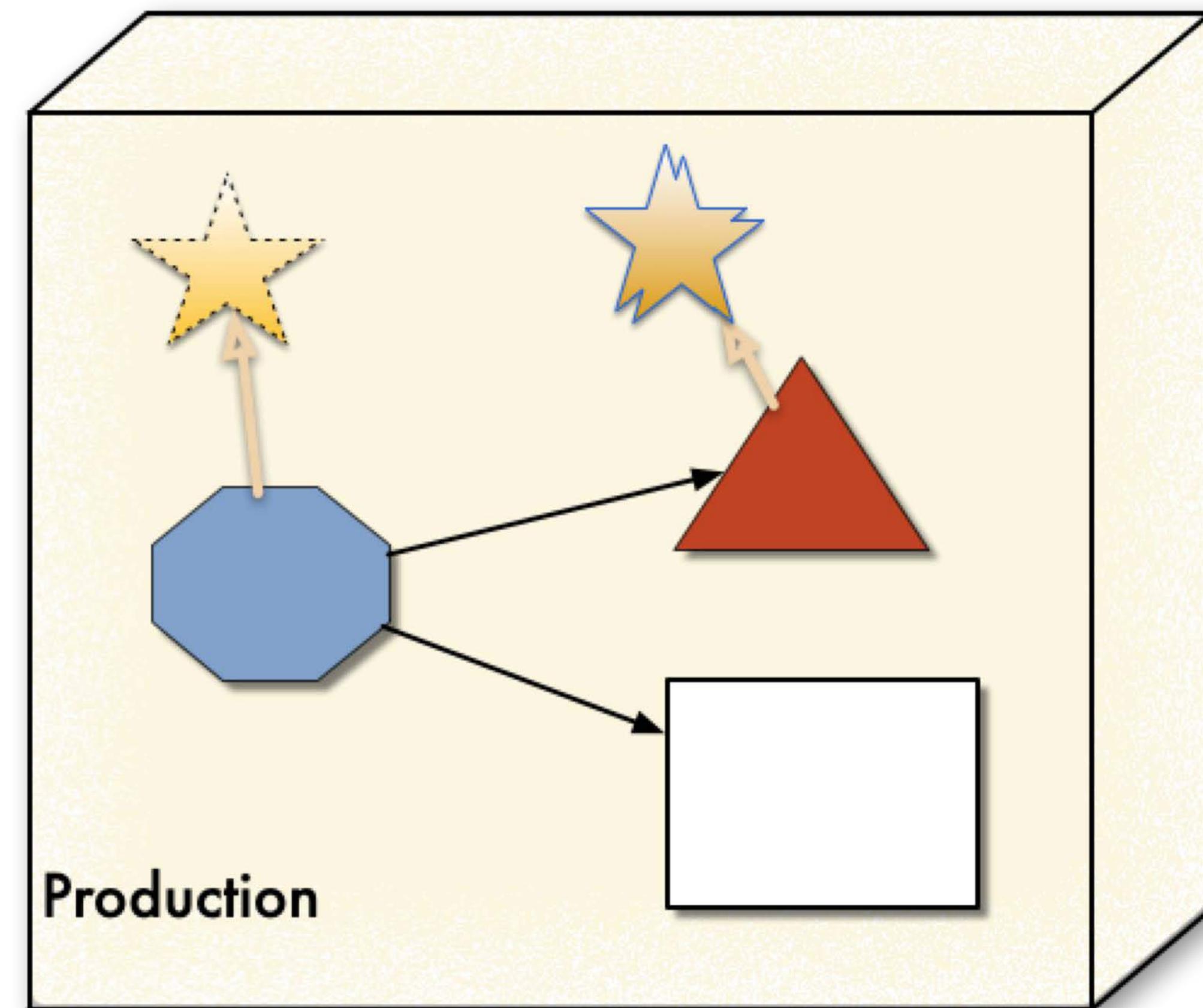
O'REILLY®

Building Evolutionary Architectures

SUPPORT CONSTANT CHANGE

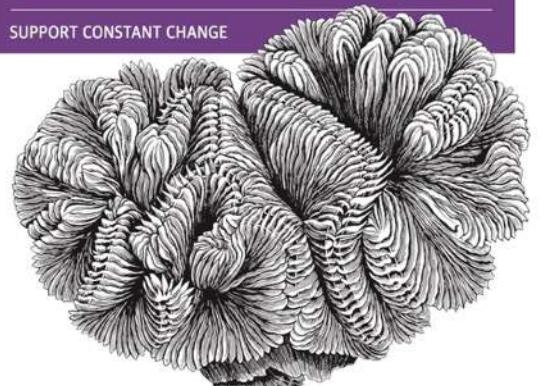


Neal Ford, Rebecca Parsons & Patrick Kua

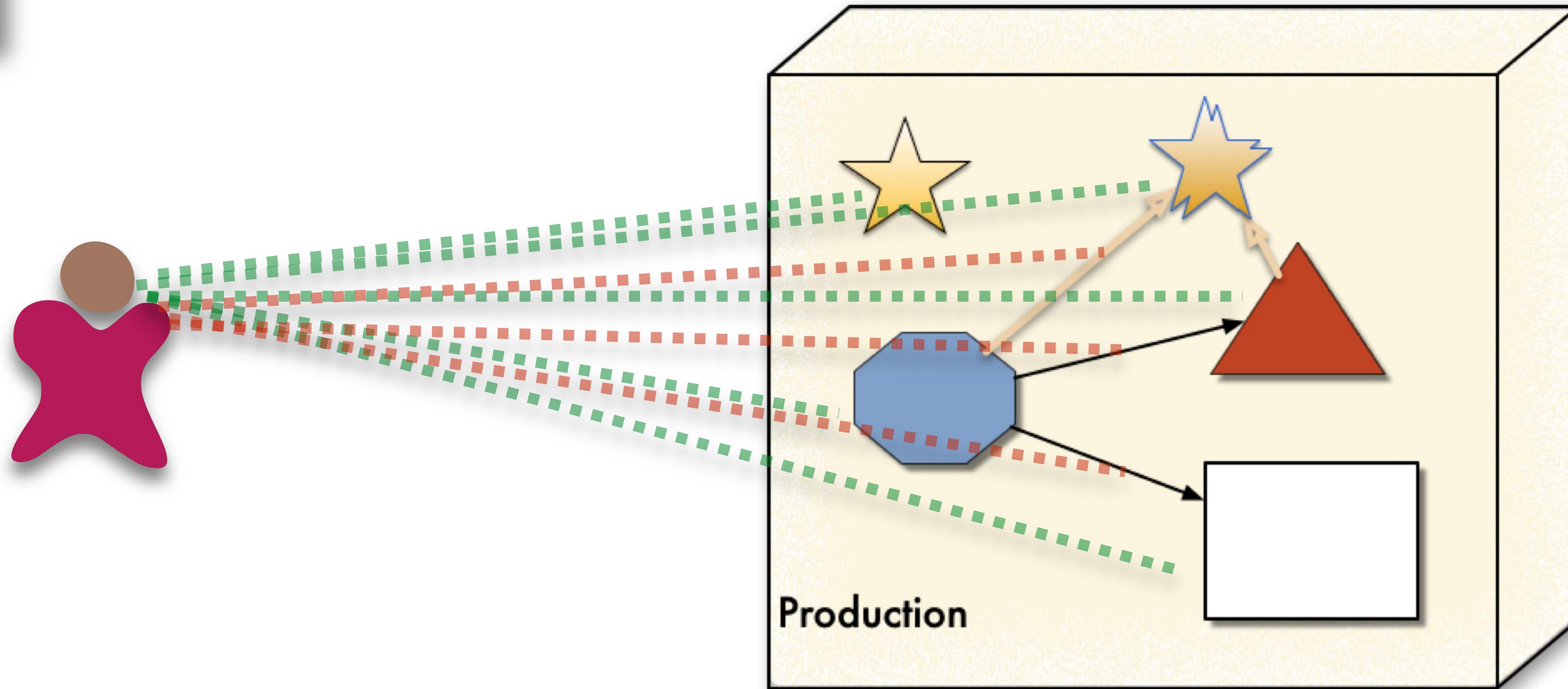


Building Evolutionary Architectures

SUPPORT CONSTANT CHANGE

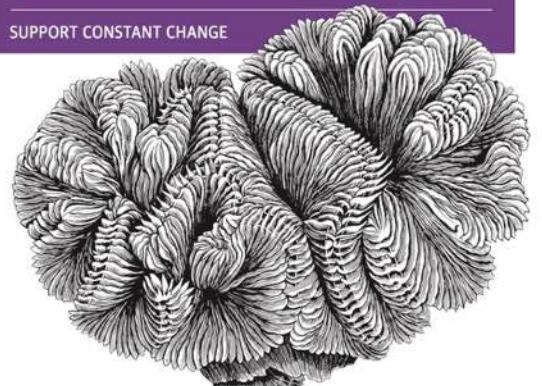


Neal Ford, Rebecca Parsons & Patrick Kua

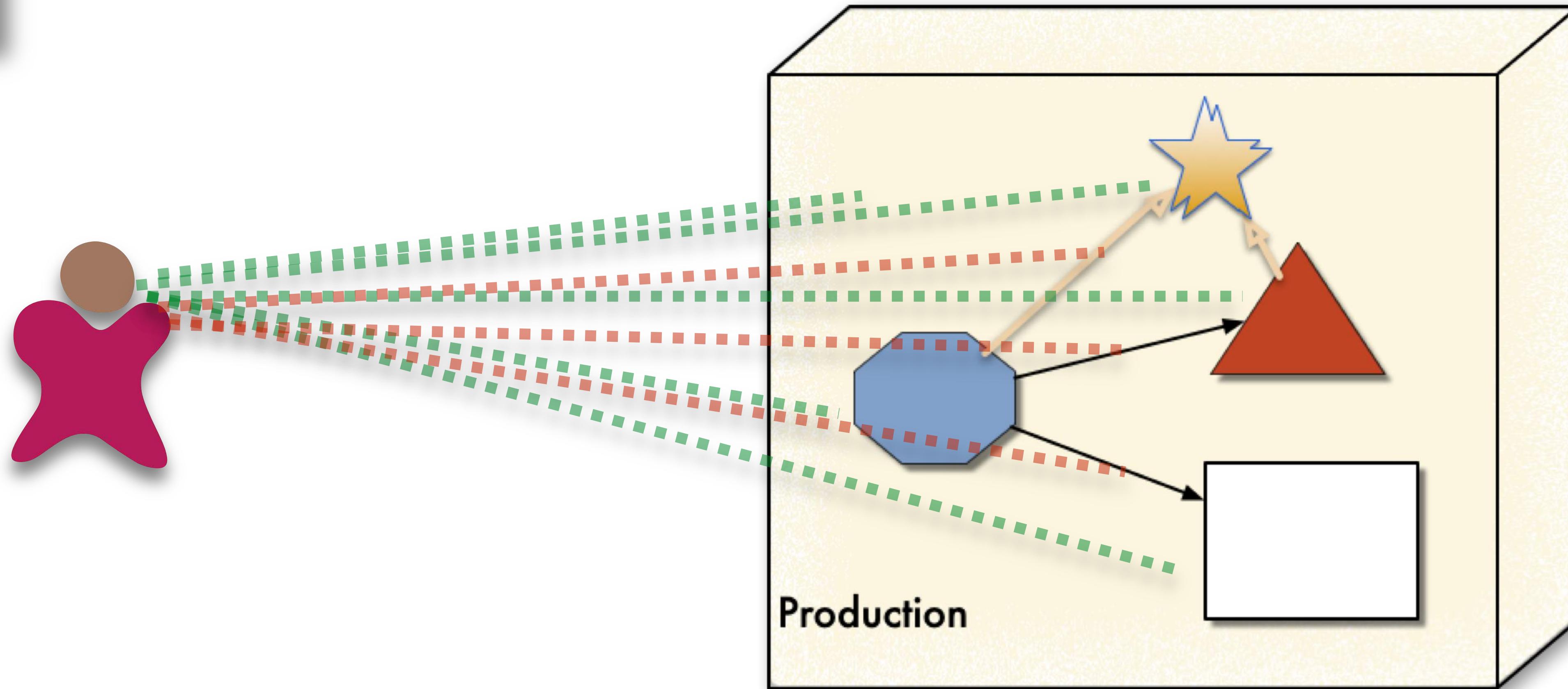


Building Evolutionary Architectures

SUPPORT CONSTANT CHANGE

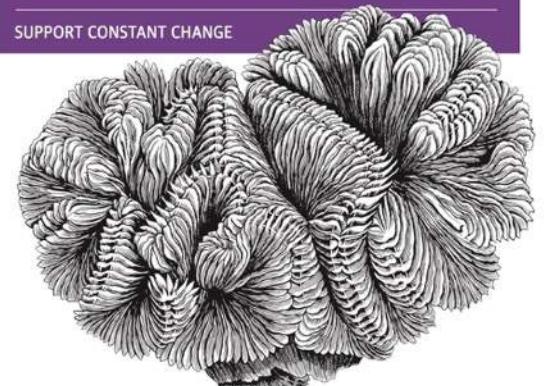


Neal Ford, Rebecca Parsons & Patrick Kua

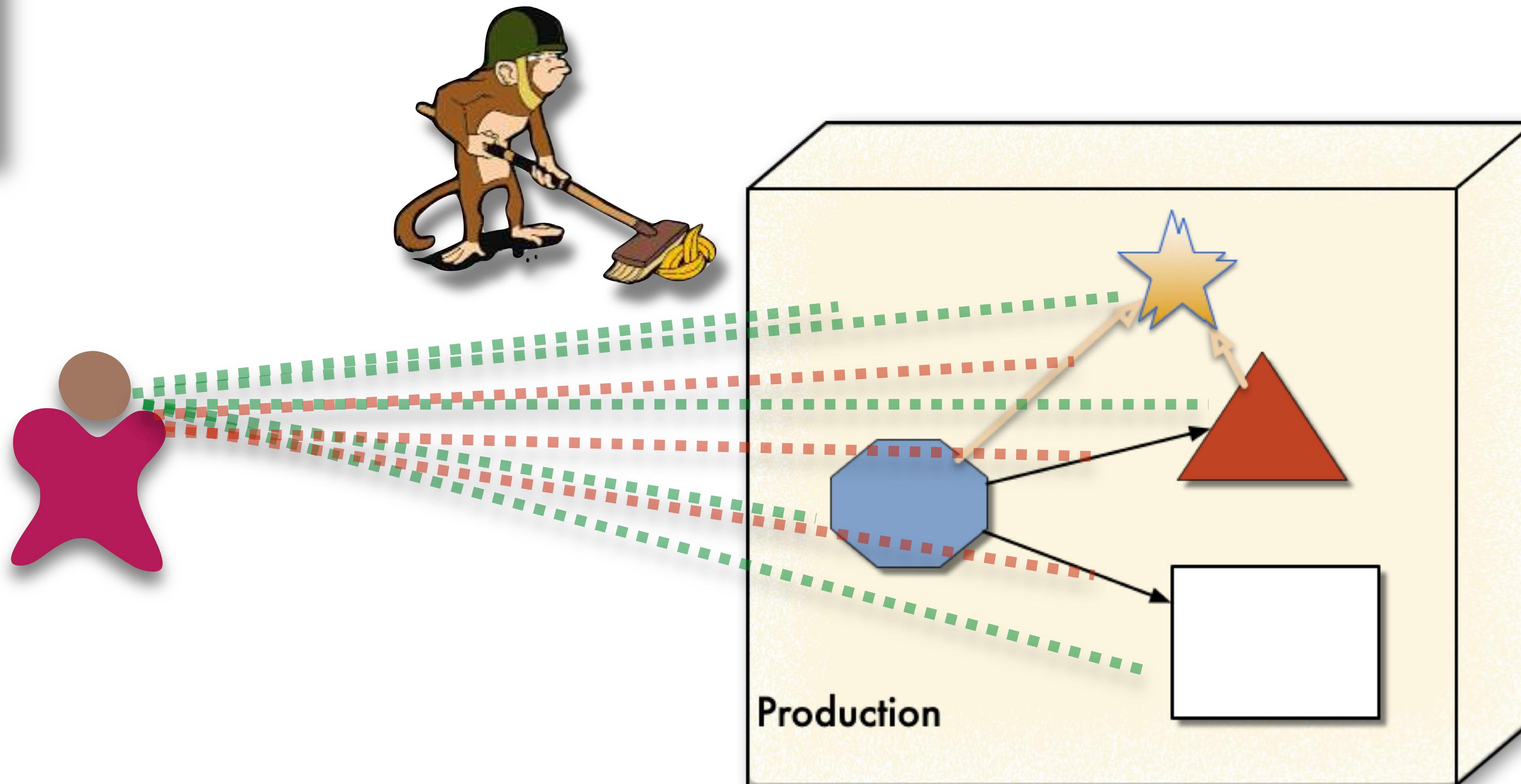


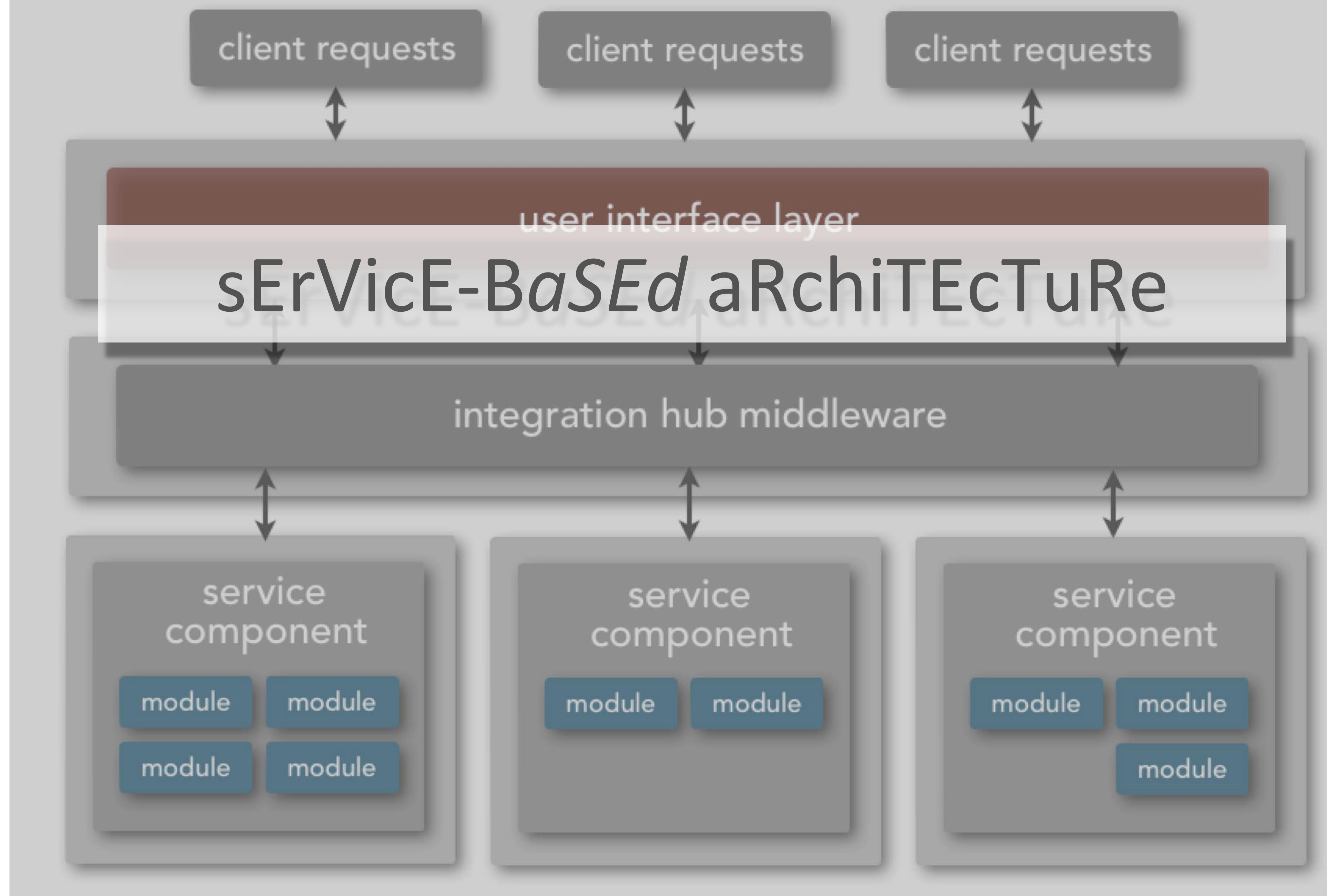
Building Evolutionary Architectures

SUPPORT CONSTANT CHANGE



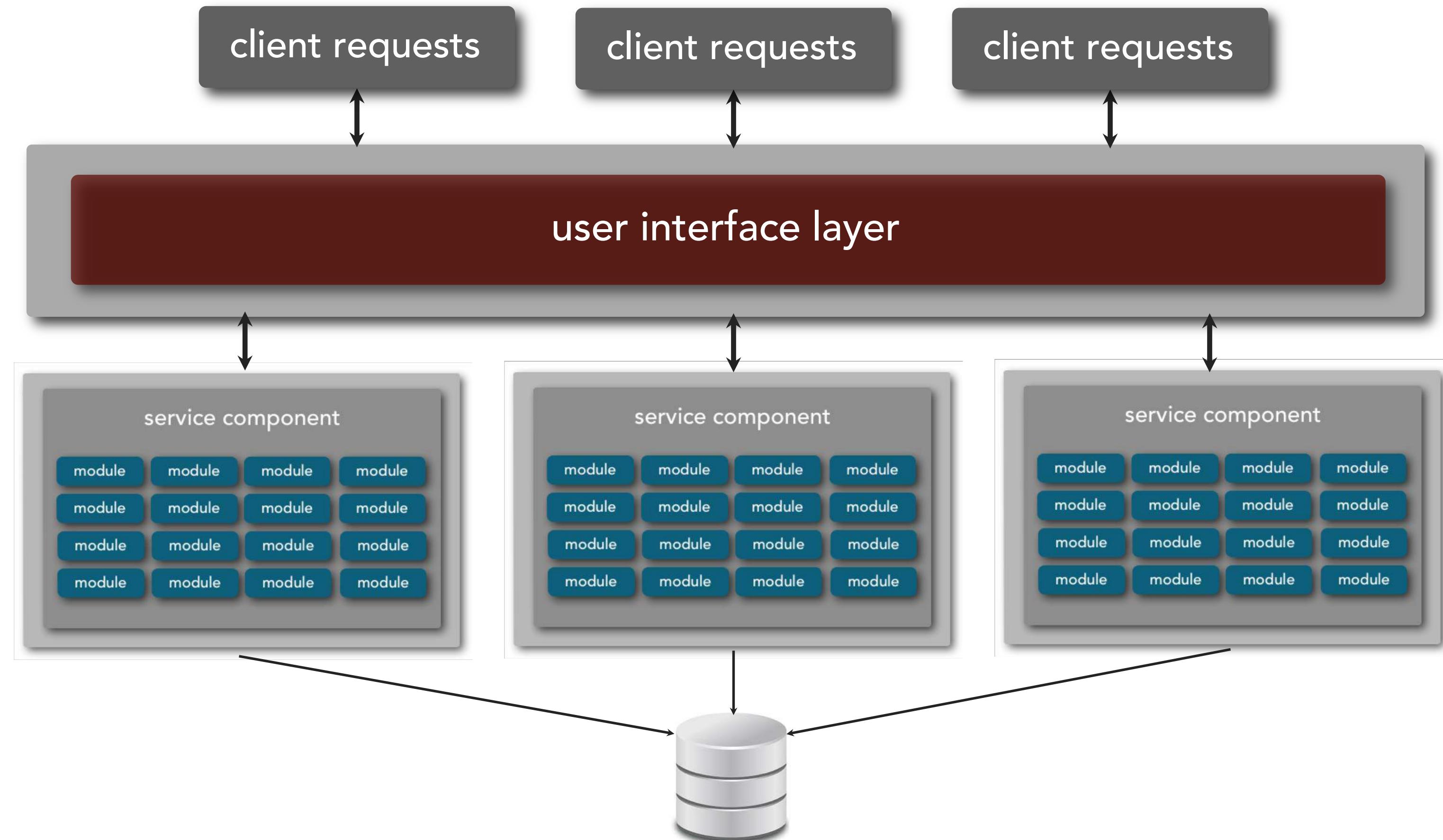
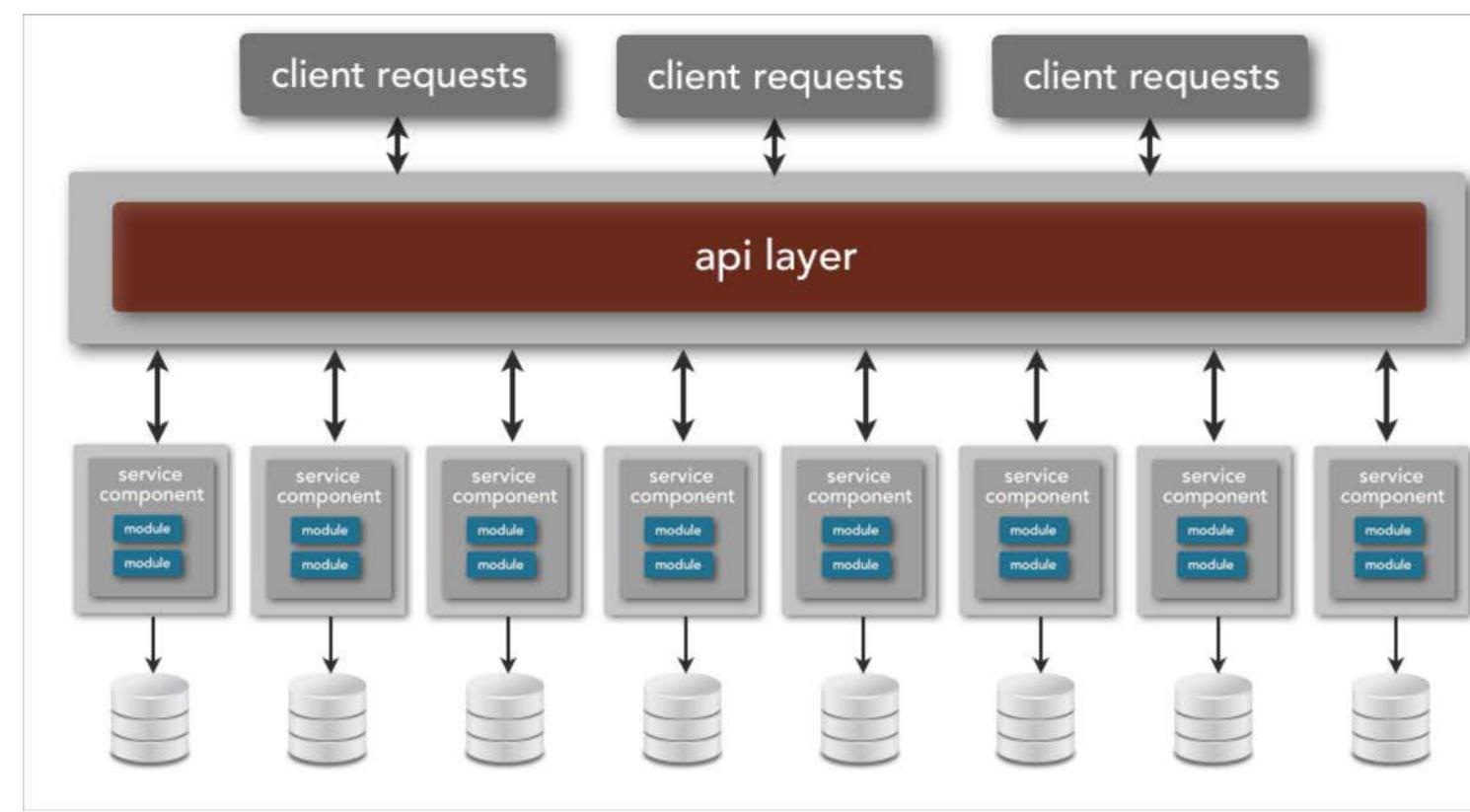
Neal Ford, Rebecca Parsons & Patrick Kua





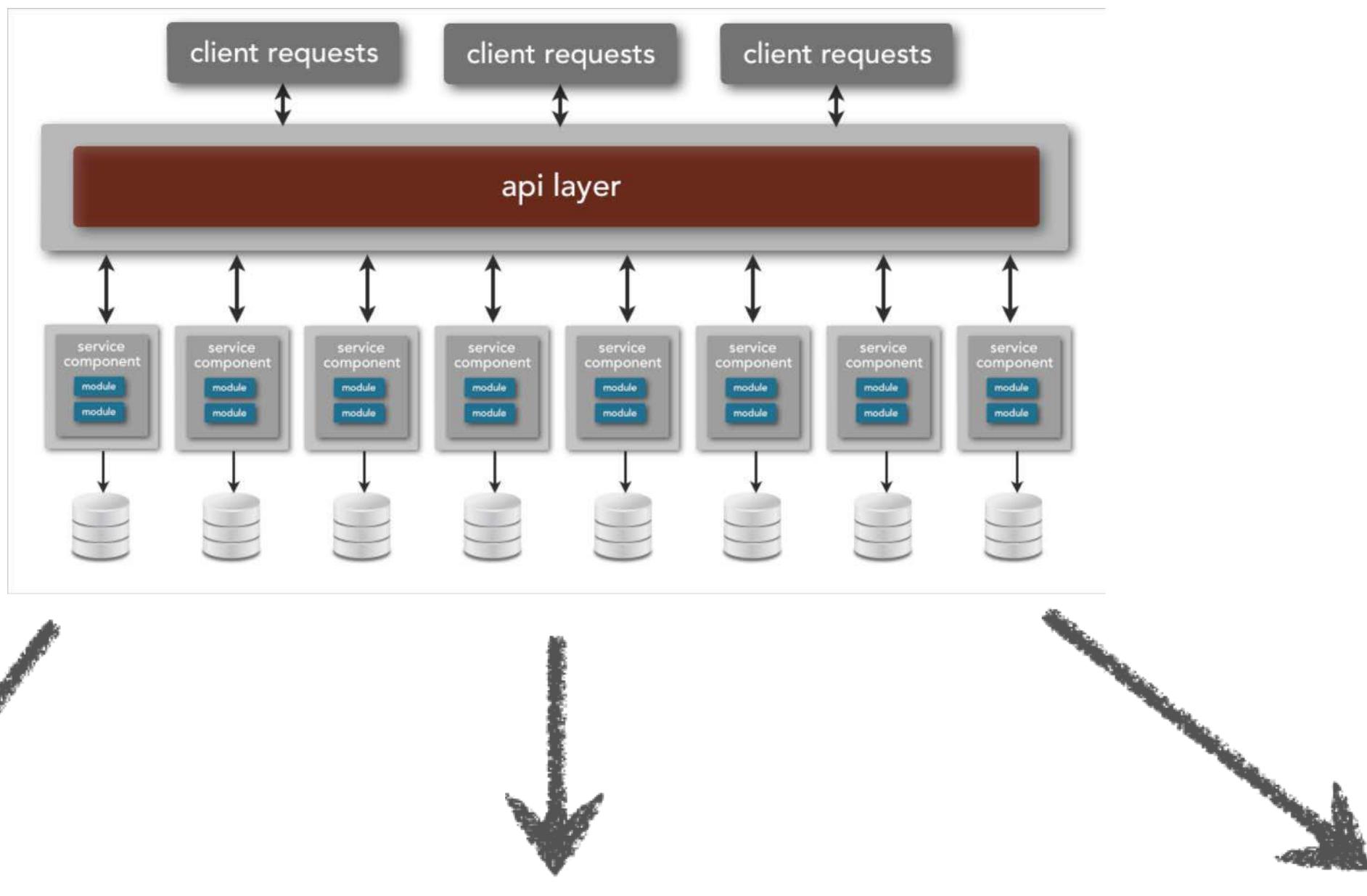
service-based architecture

the goals of microservices...

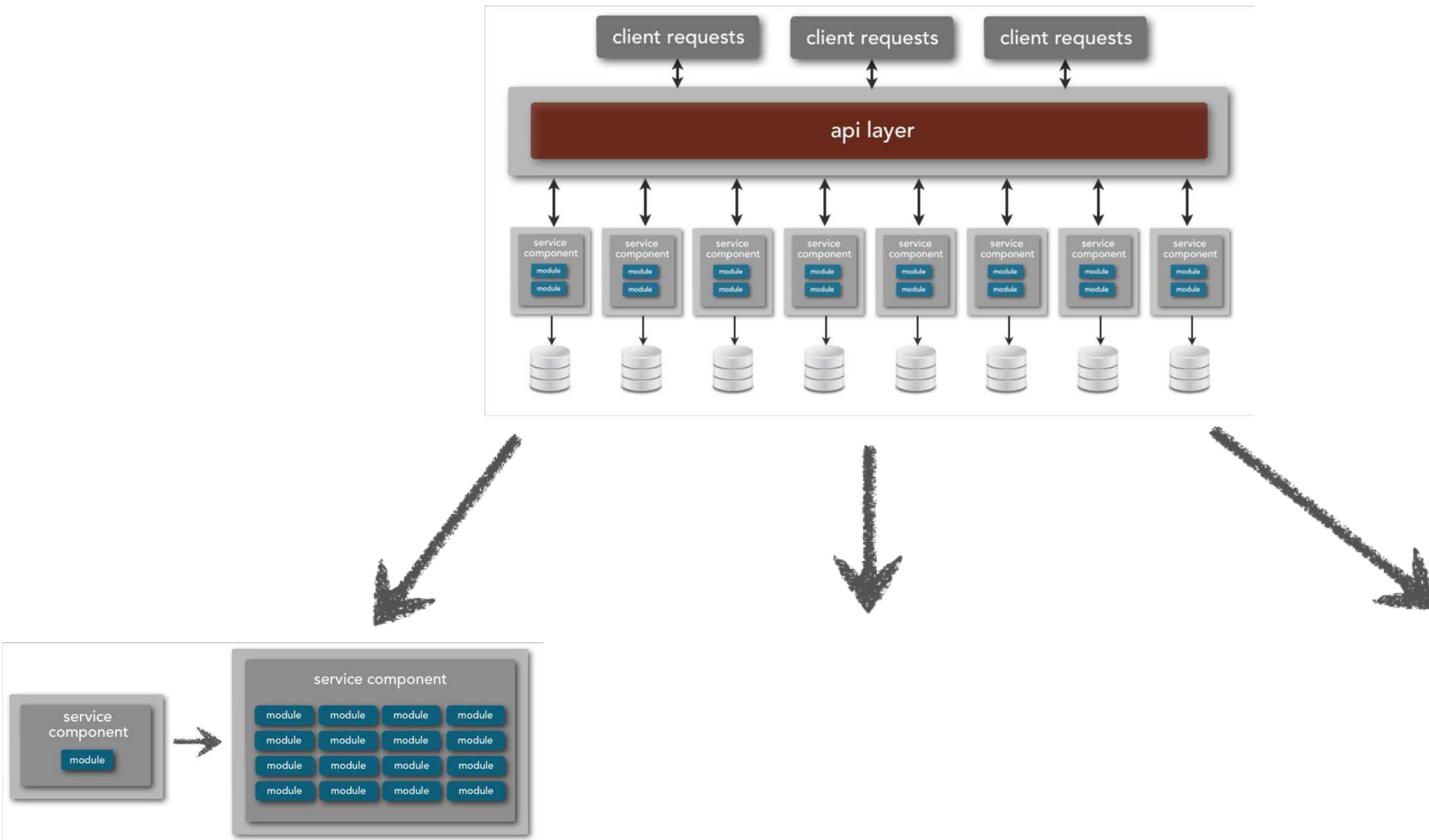


...with real-world constraints.

service-based architecture

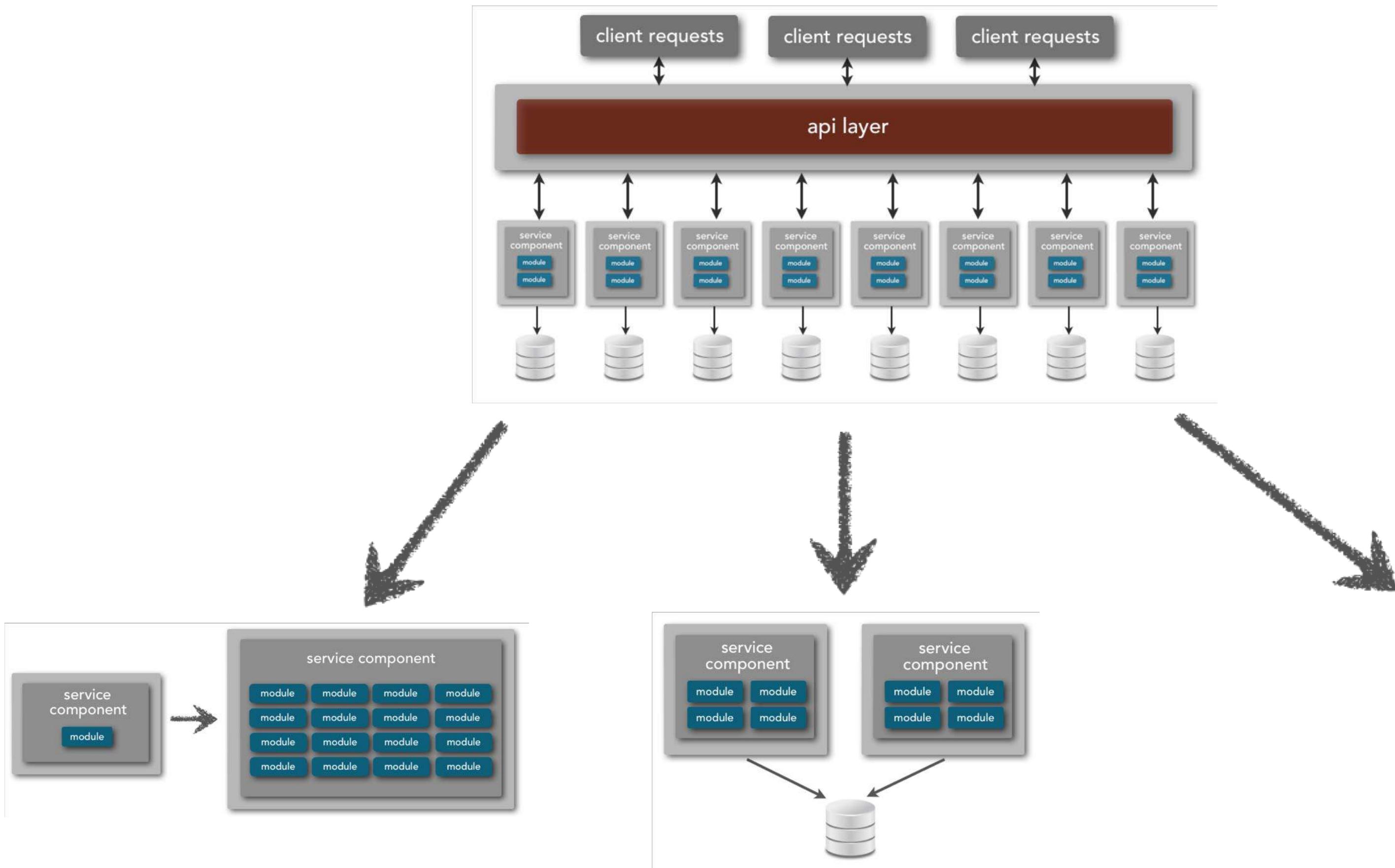


service-based architecture



service
granularity

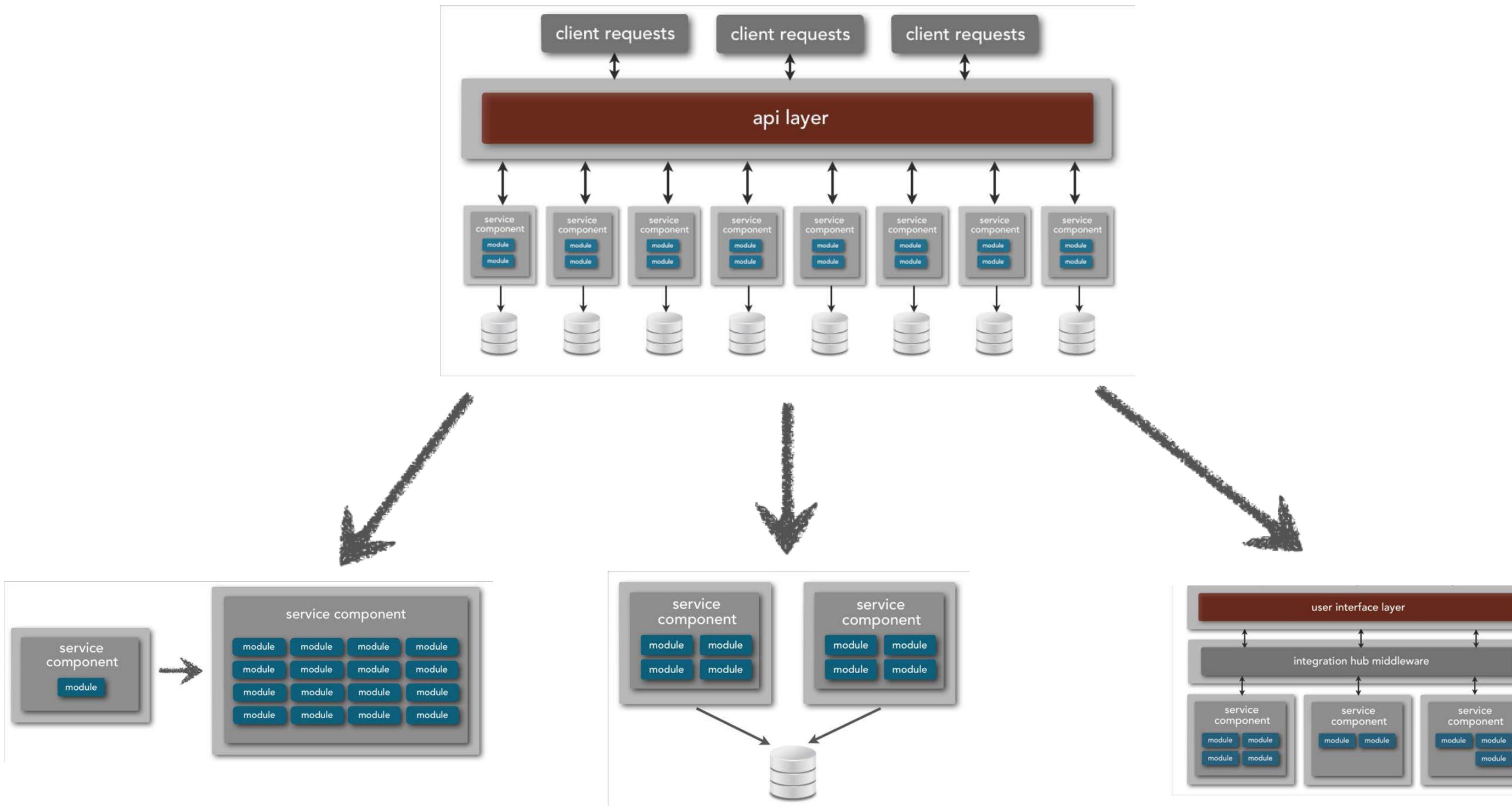
service-based architecture



service
granularity

database
scope

service-based architecture



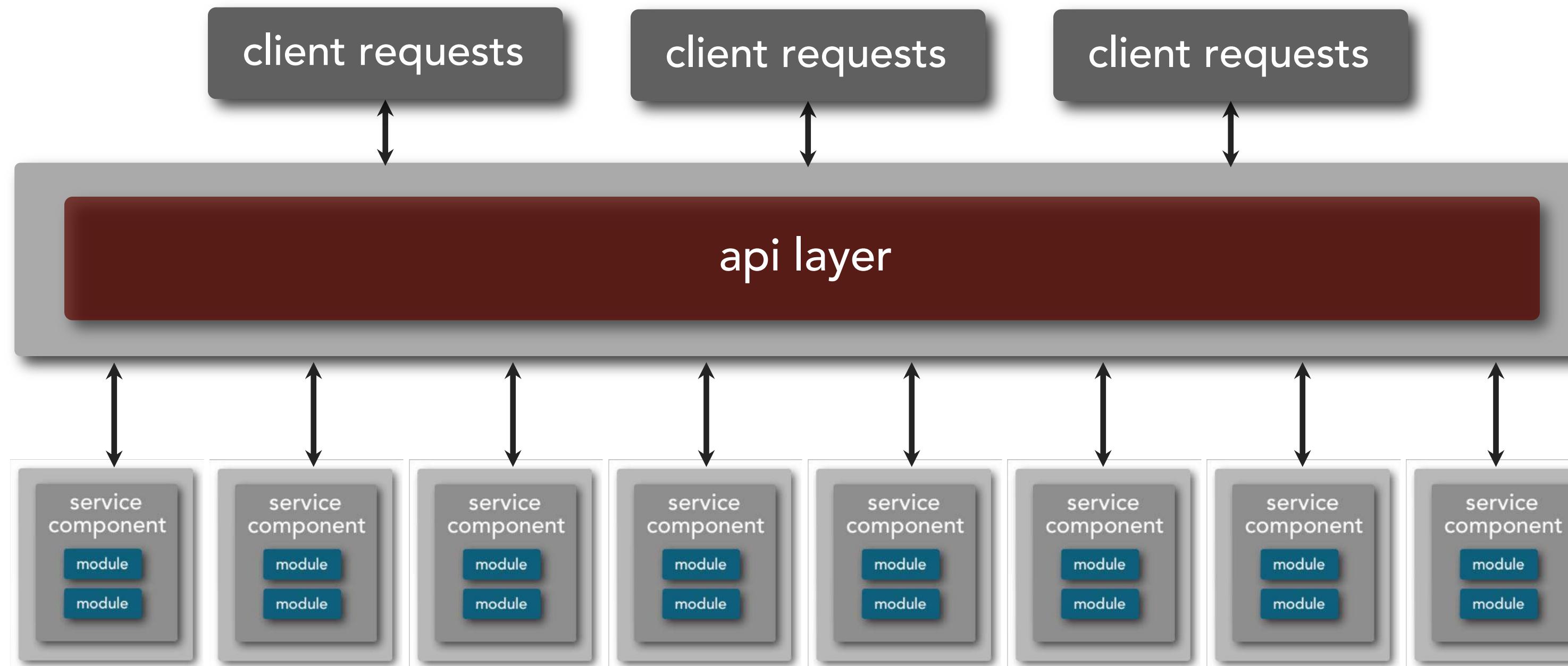
service
granularity

database
scope

integration
hub

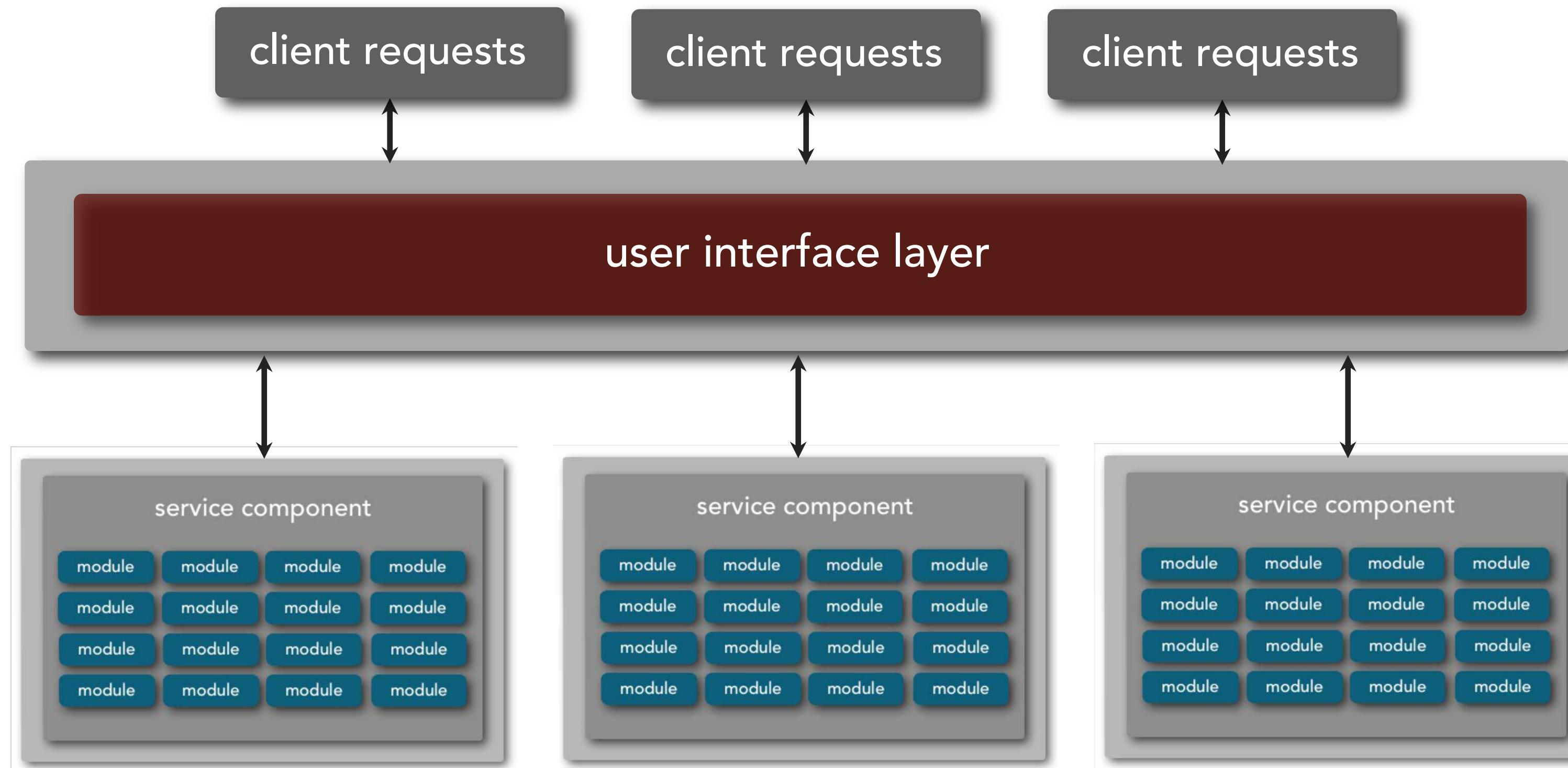
service-based architecture

service granularity



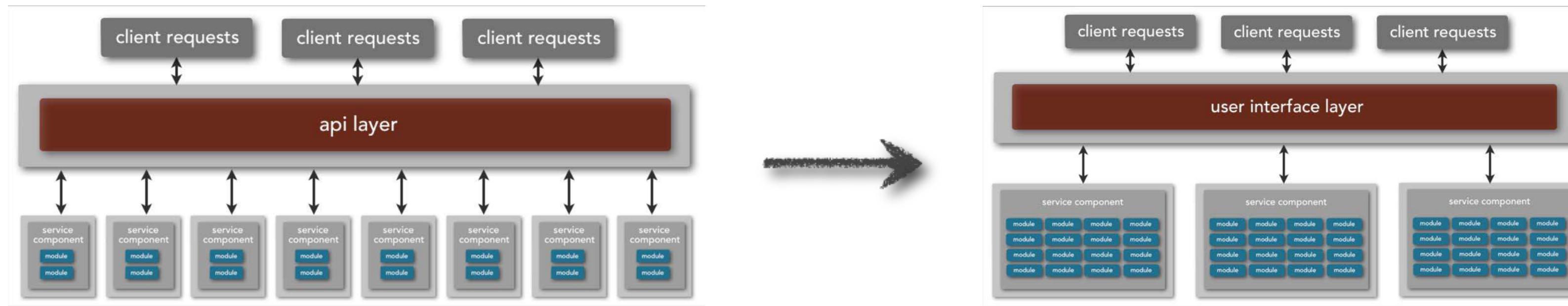
service-based architecture

service granularity



service-based architecture

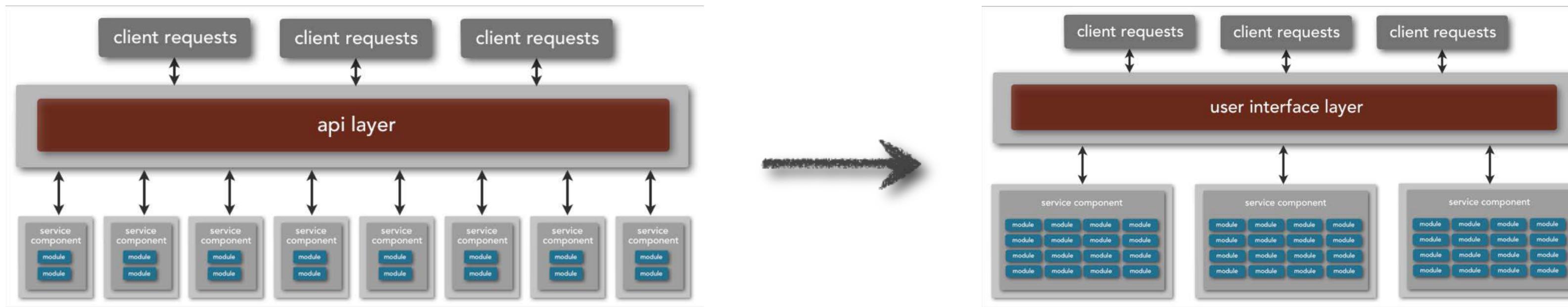
service granularity



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

service-based architecture

service granularity



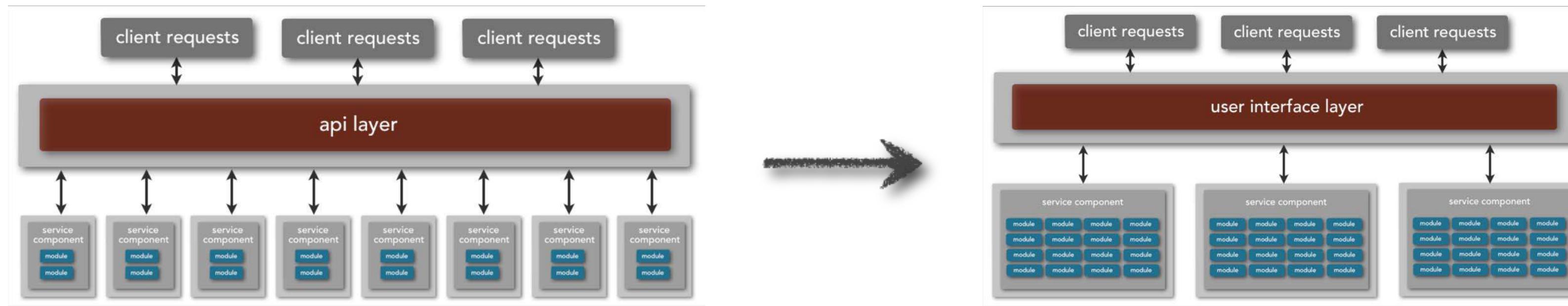
single-purpose micro-service to "portion of the application"
macro-service



macro-services resolves orchestration and transactional issues

service-based architecture

service granularity



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



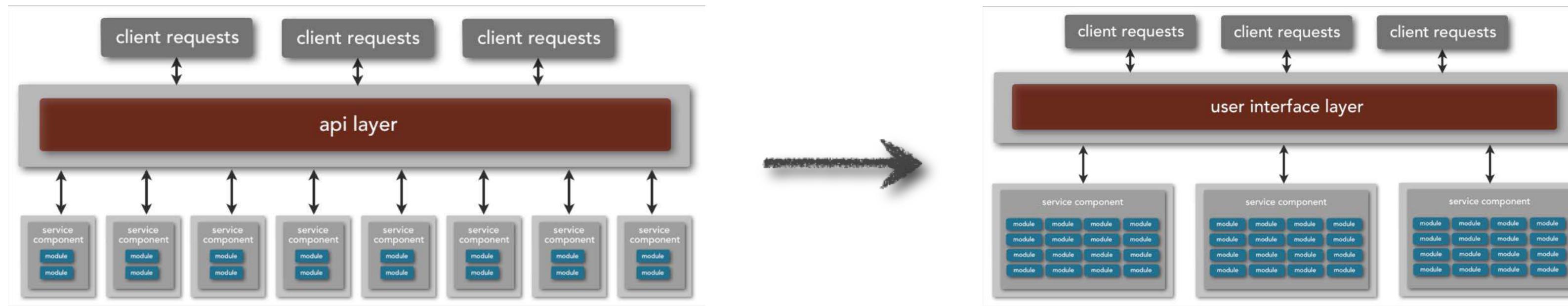
macro-services resolves orchestration and transactional issues



allows for complex business processing within a service context

service-based architecture

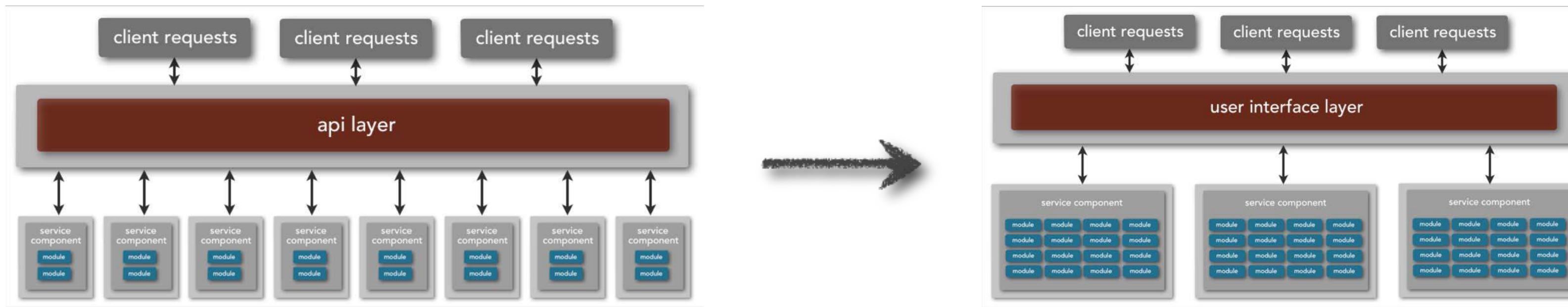
service granularity



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

service-based architecture

service granularity



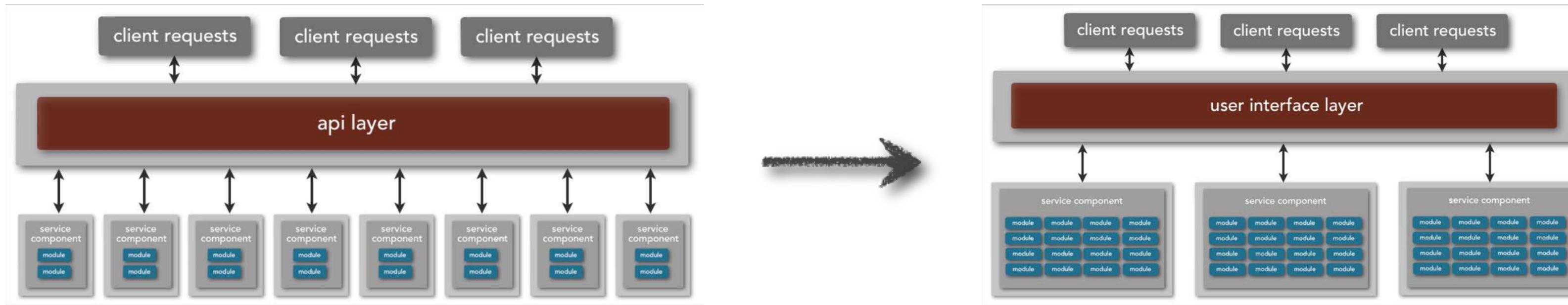
single-purpose micro-service to "portion of the application"
macro-service



services become harder to develop and test

service-based architecture

service granularity



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



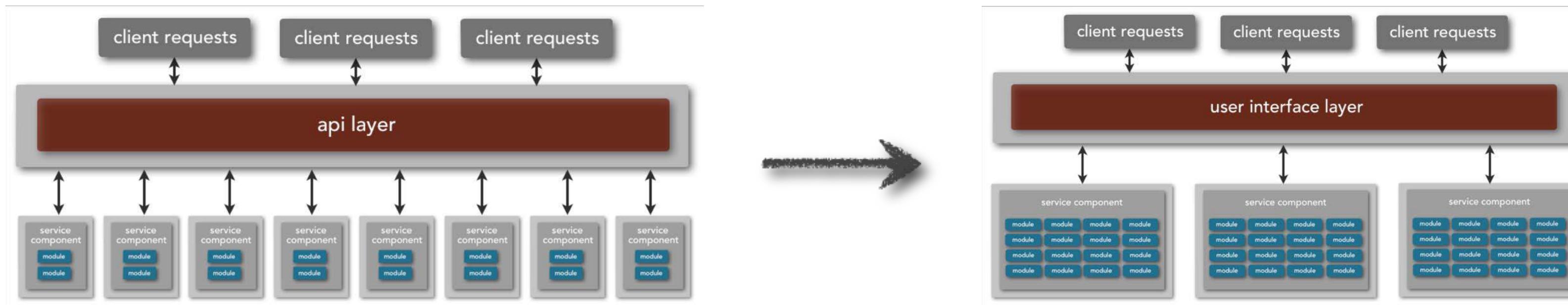
services become harder to develop and test



deployment pipeline requires more planning

service-based architecture

service granularity

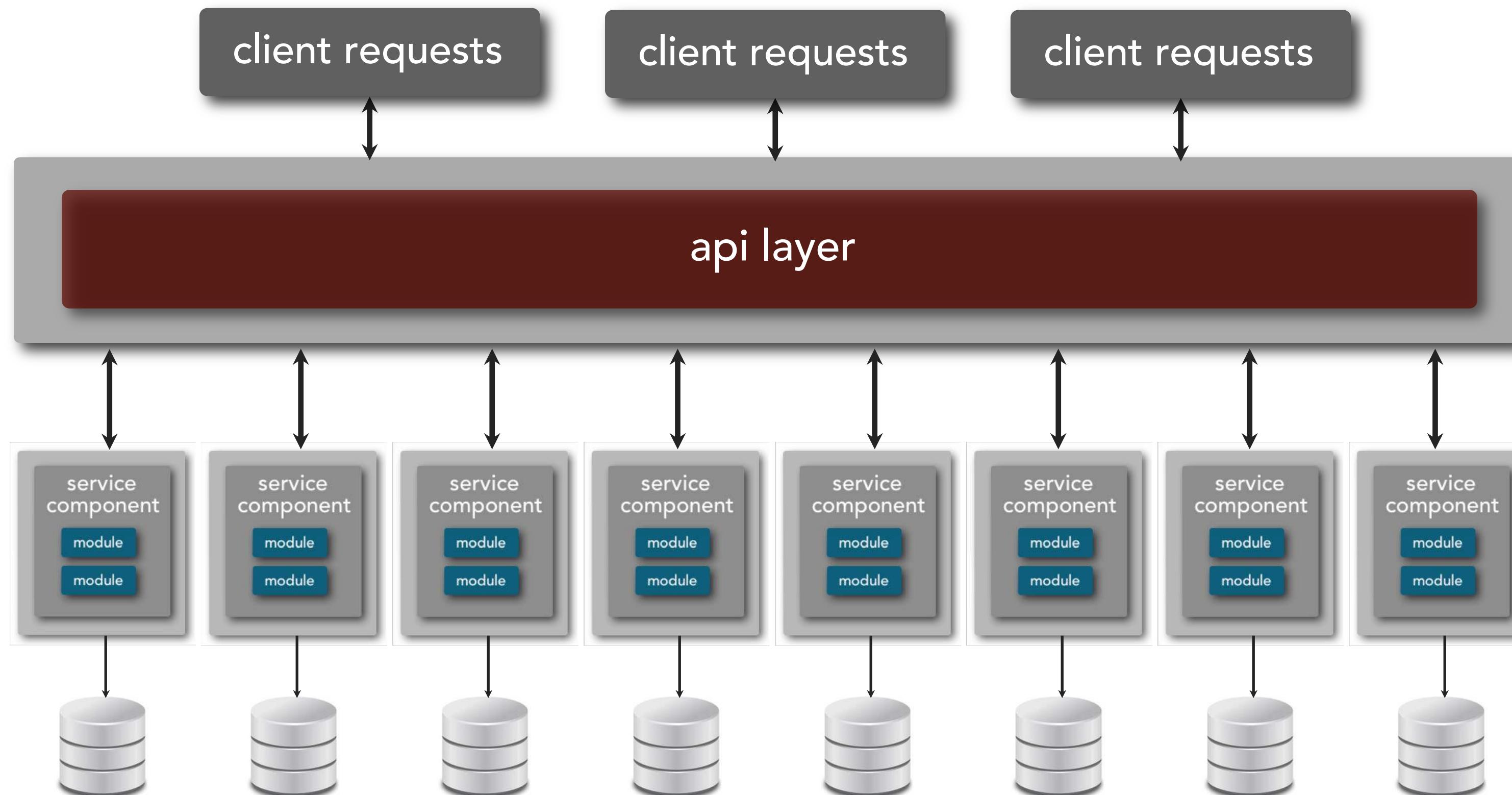


single-purpose micro-service to "portion of the application"
macro-service

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

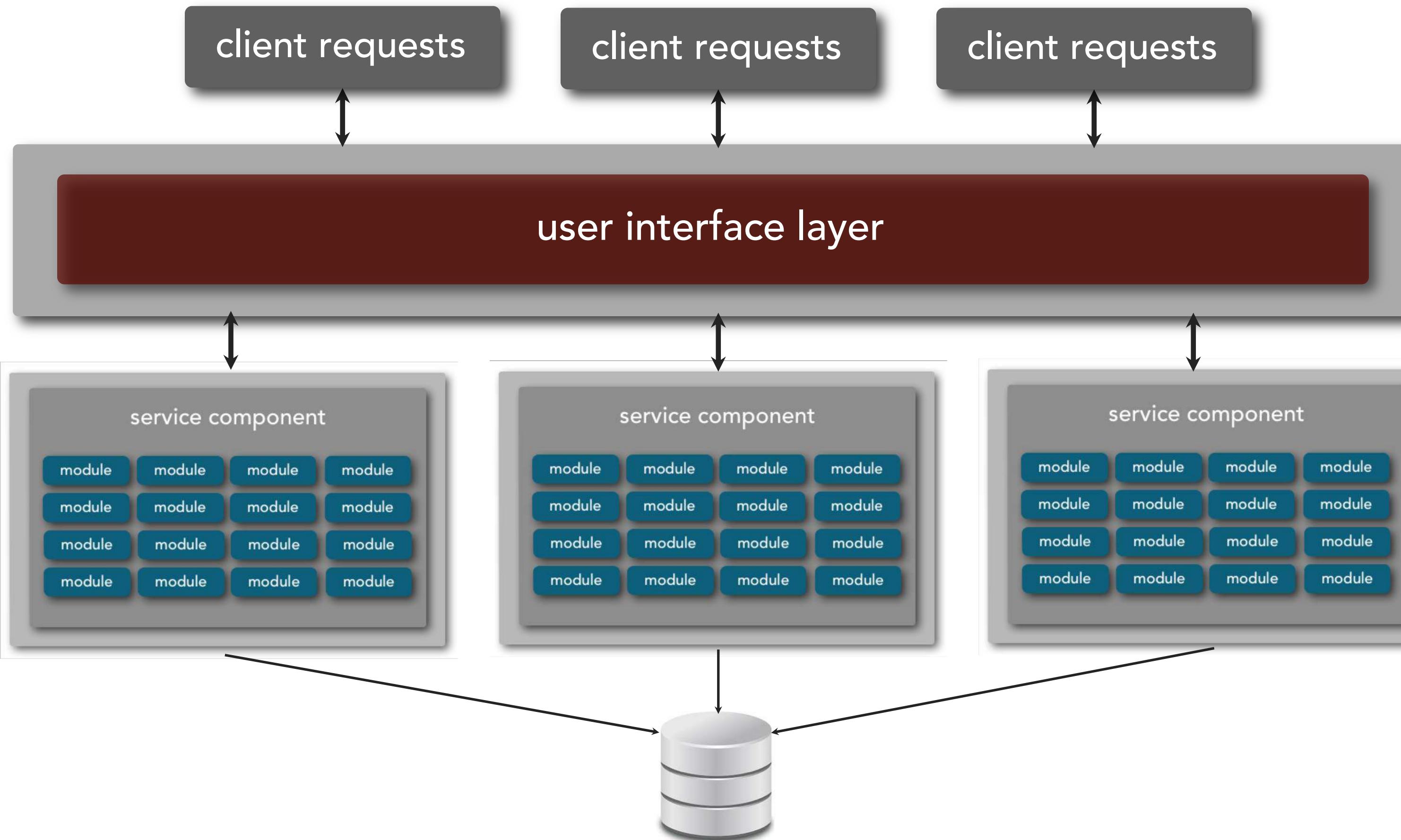
service-based architecture

database scope



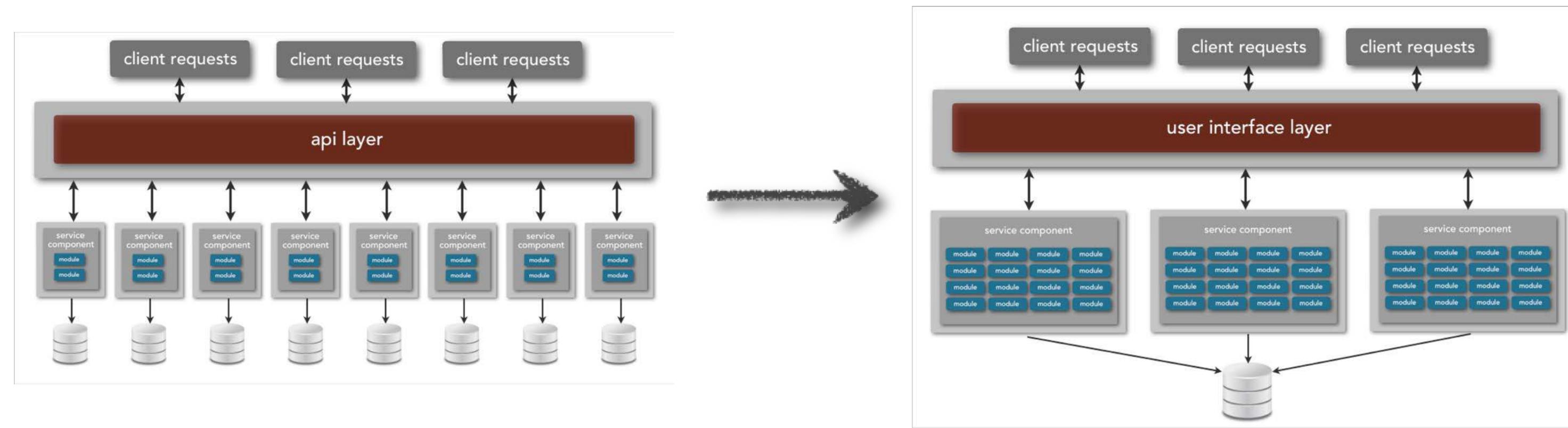
service-based architecture

database scope



service-based architecture

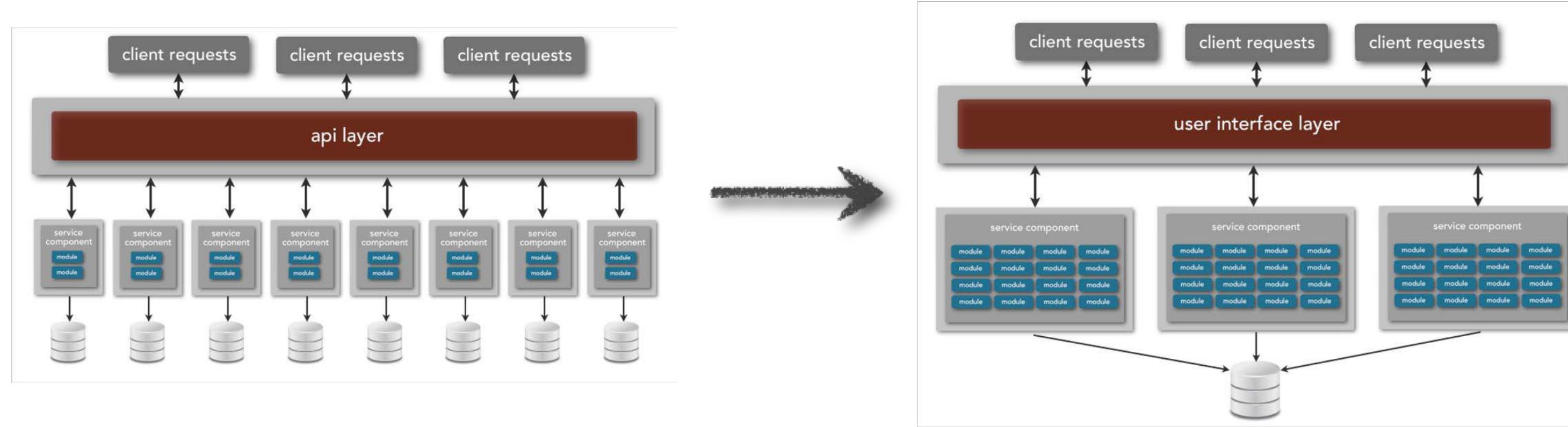
database scope



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

service-based architecture

database scope



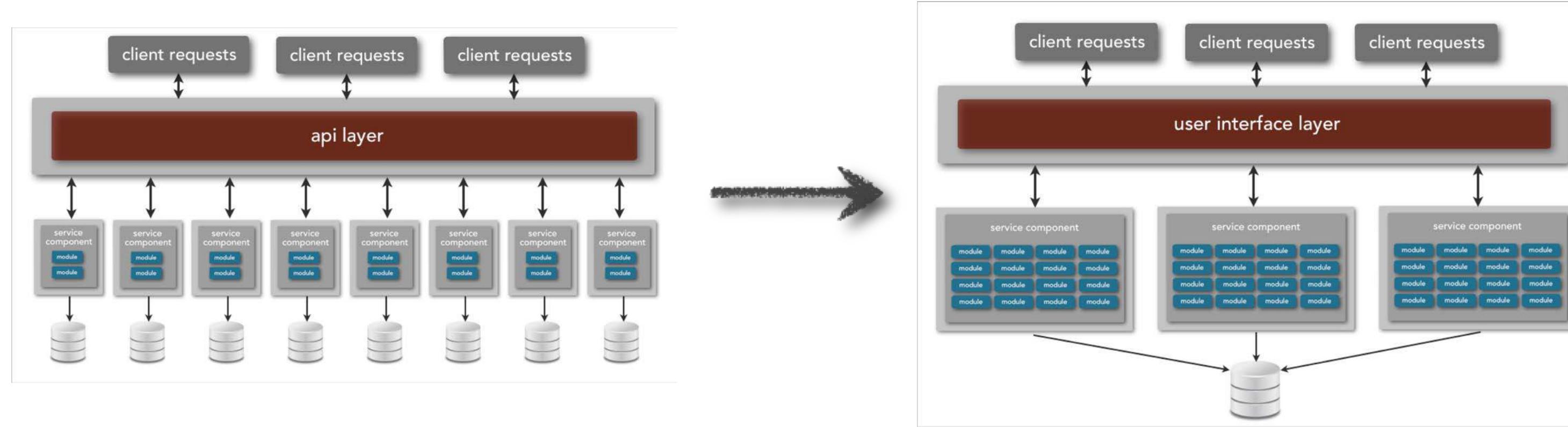
single-purpose service-based database to globally shared application database



reduces service orchestration and contract dependencies

service-based architecture

database scope



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



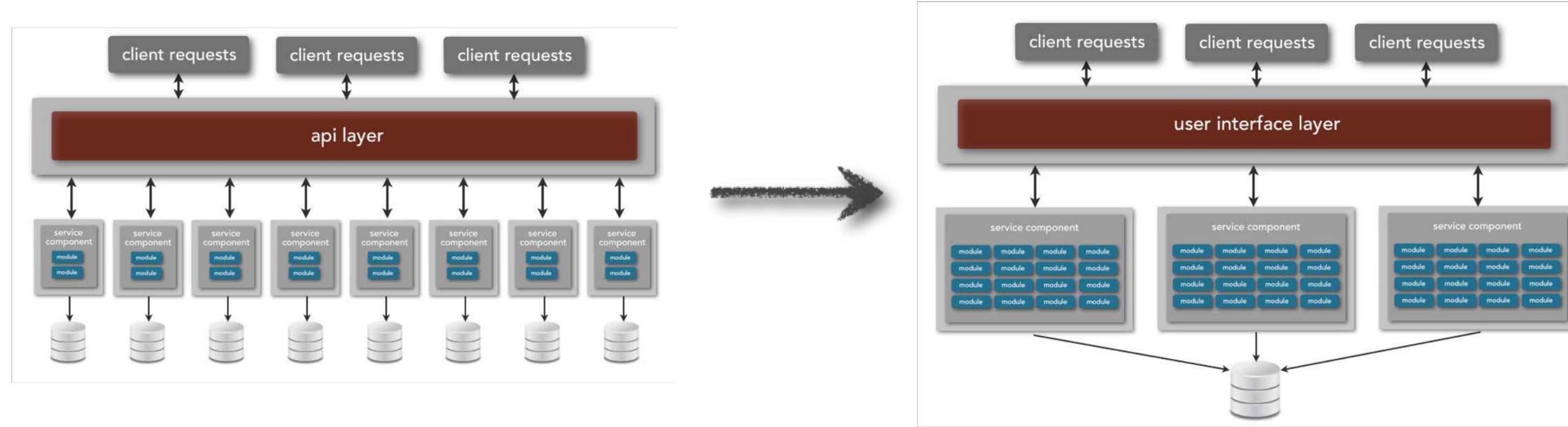
reduces service orchestration and contract dependencies



improves performance due to fewer remote calls

service-based architecture

database scope

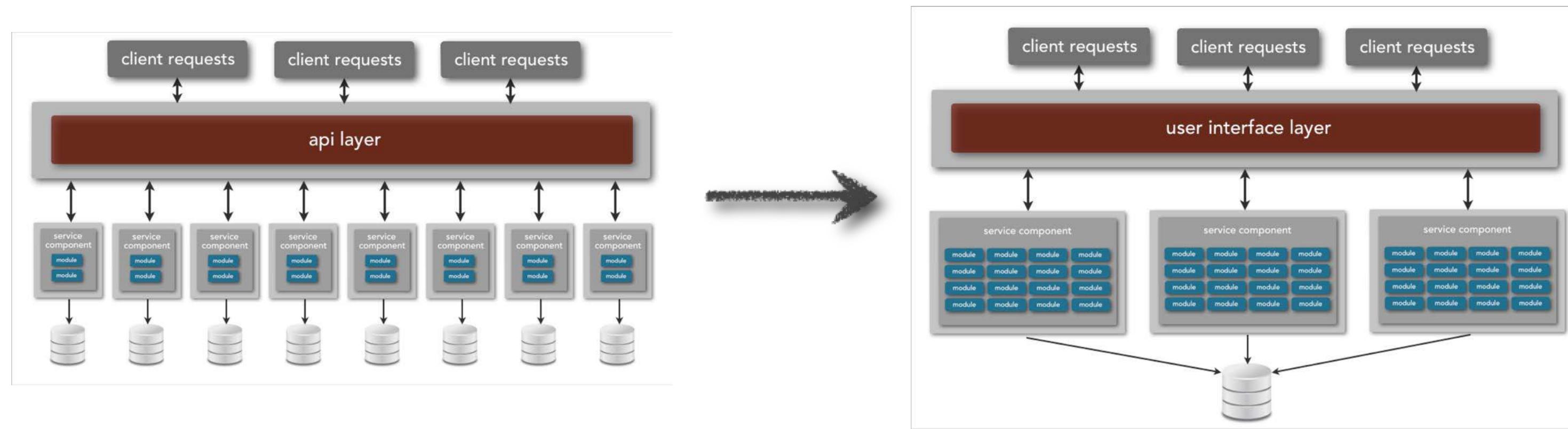


single-purpose service-based database to globally shared application database

- 👍 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



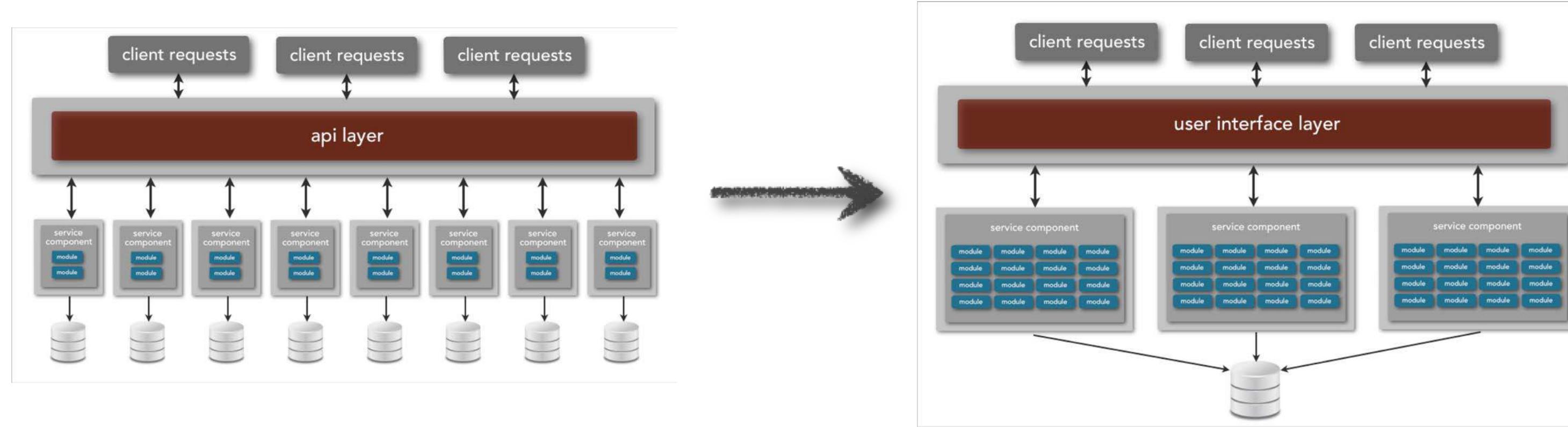
single-purpose service-based database to globally shared application database



looser bounded context of services

service-based architecture

database scope



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



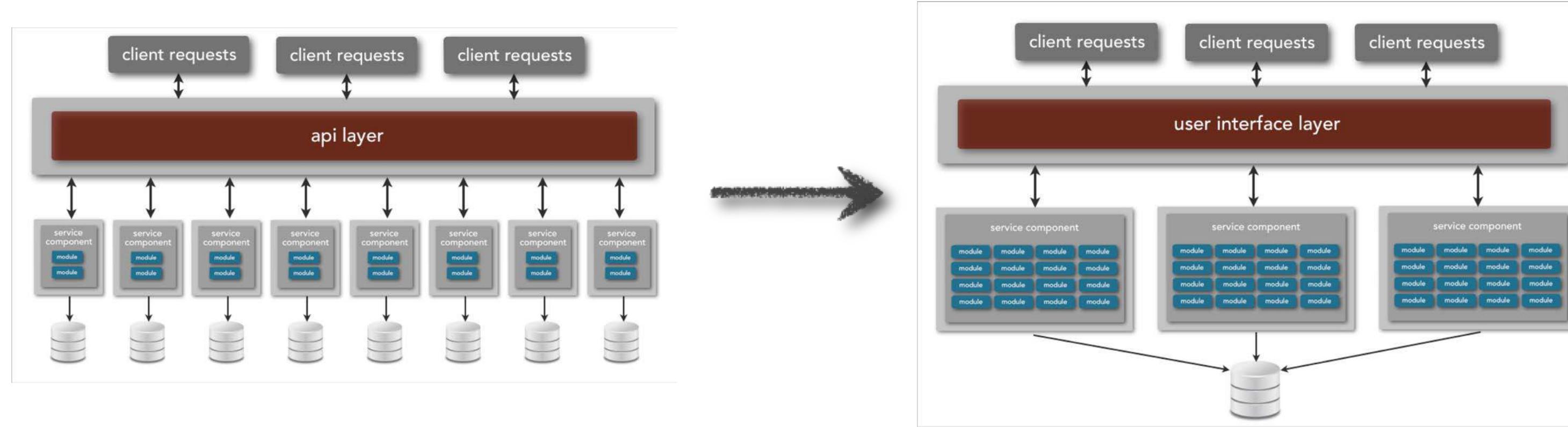
looser bounded context of services



tighter service coupling based on schema

service-based architecture

database scope



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



looser bounded context of services



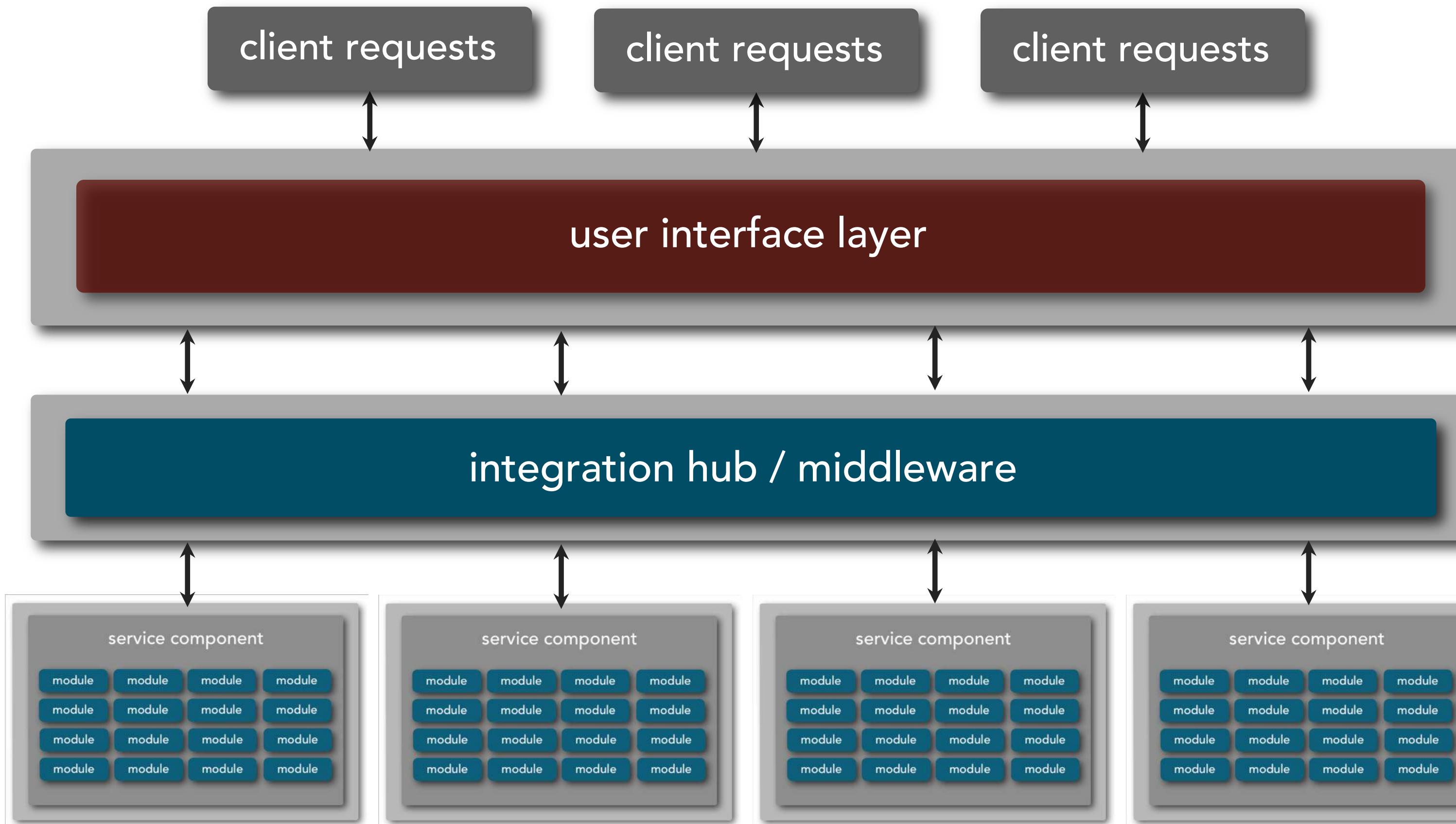
tighter service coupling based on schema



schema changes become expensive and difficult

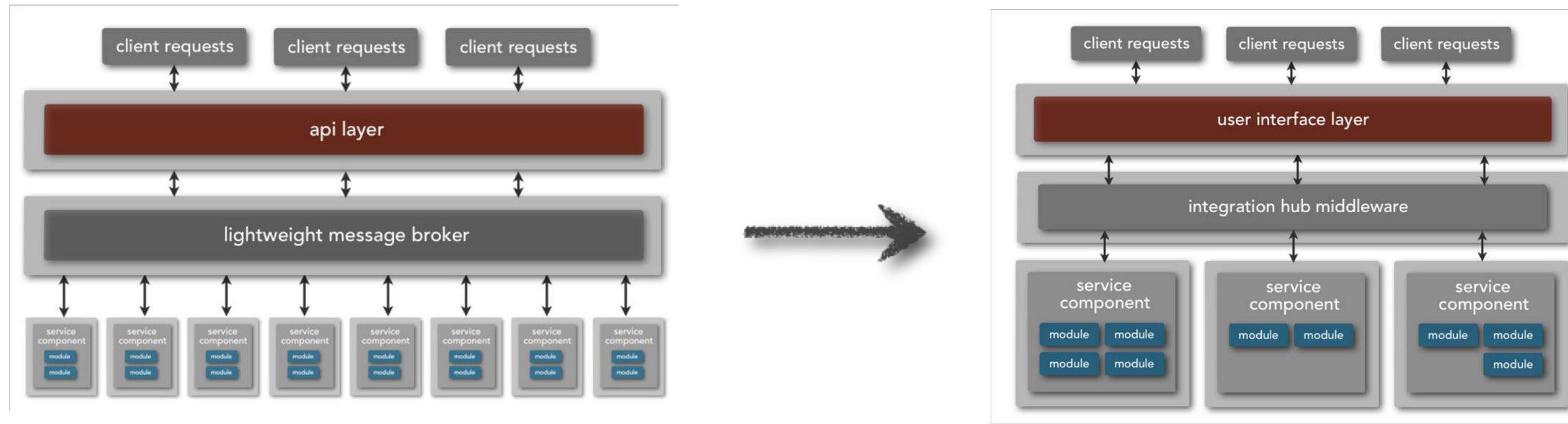
service-based architecture

integration hub



service-based architecture

integration hub



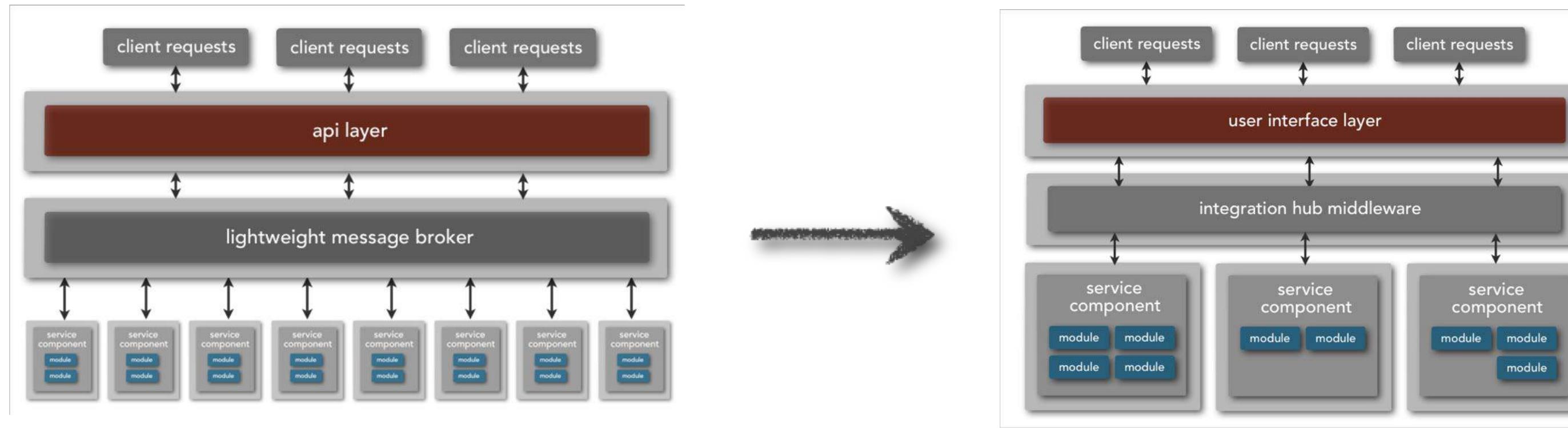
adding integration hub, towards mediator



allows for transformation of contract differences

service-based architecture

integration hub



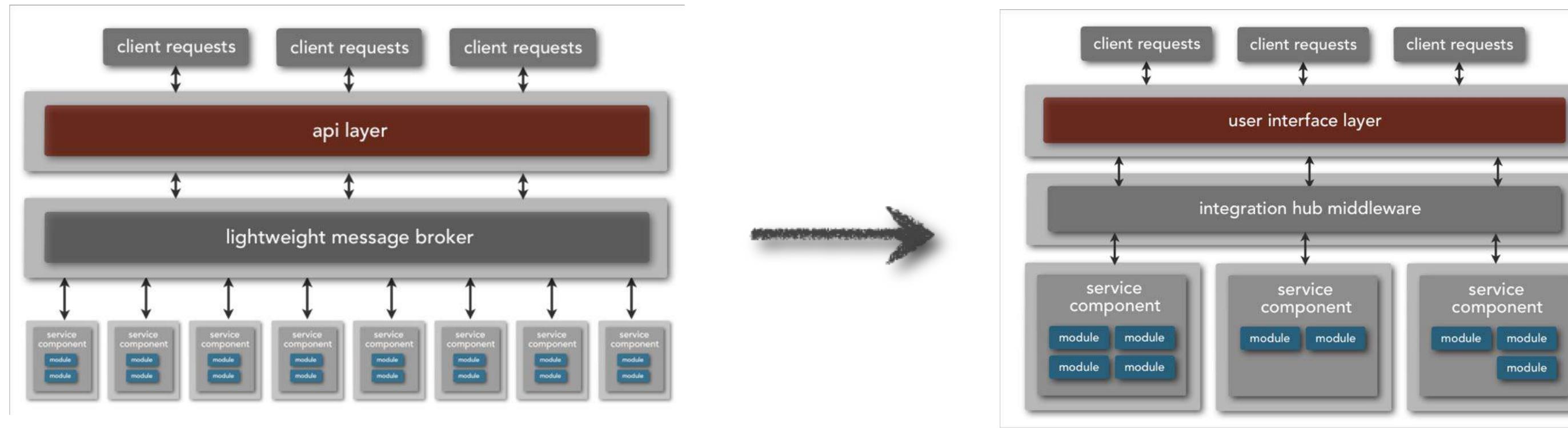
adding integration hub, towards mediator

👍 allows for transformation of contract differences

👍 allows for non-transactional orchestration of services

service-based architecture

integration hub

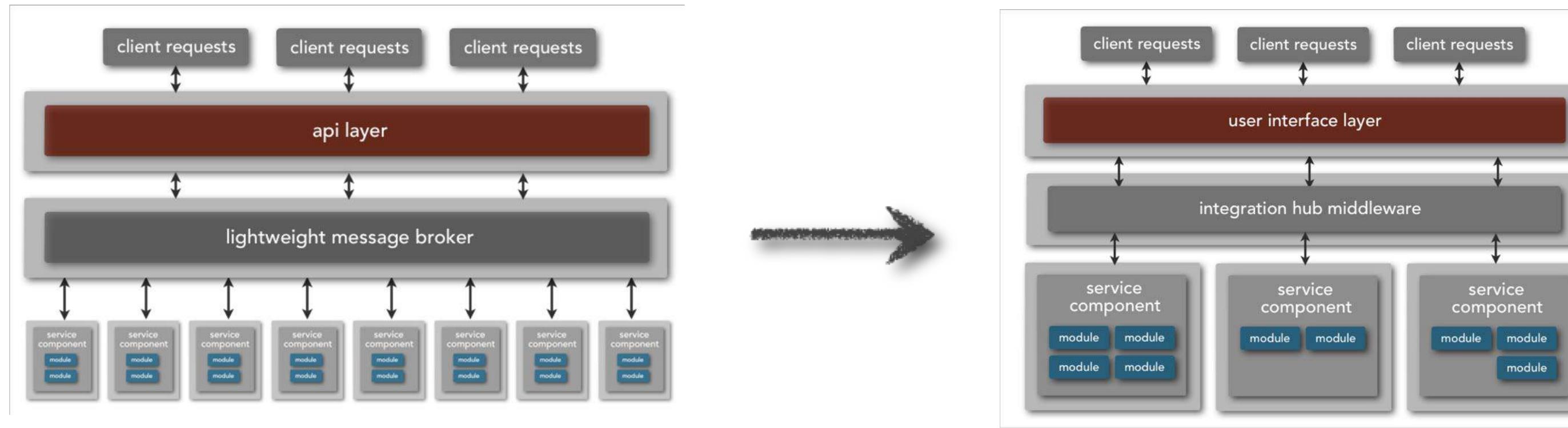


adding integration hub, towards mediator

- 👍 allows for transformation of contract differences
- 👍 allows for non-transactional orchestration of services
- 👍 allows for protocol-agnostic heterogeneous interoperability

service-based architecture

integration hub

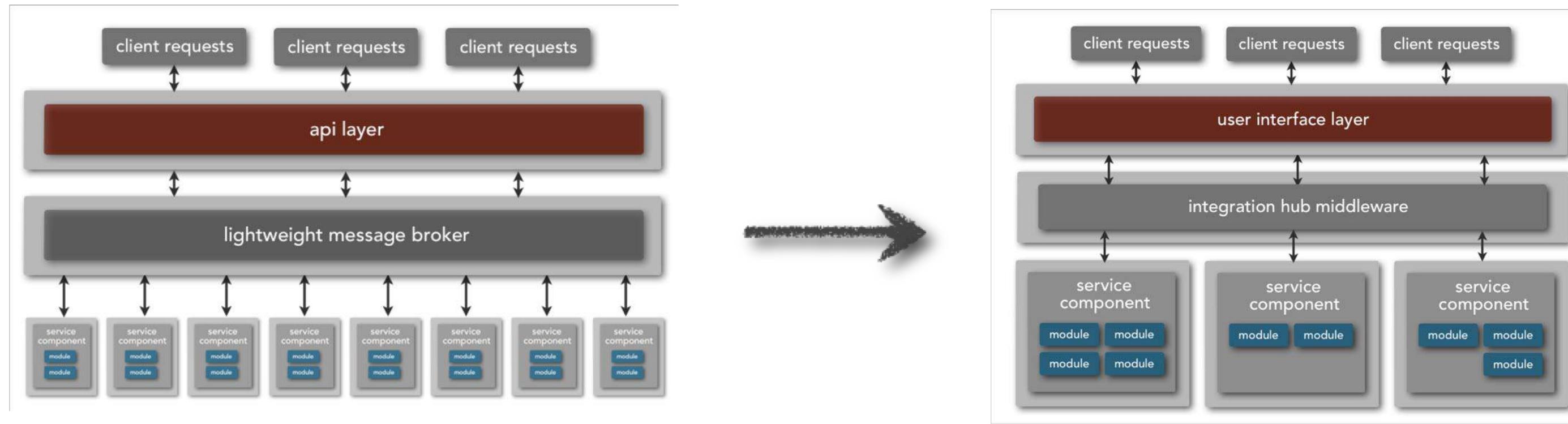


adding integration hub, towards mediator

- 👍 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



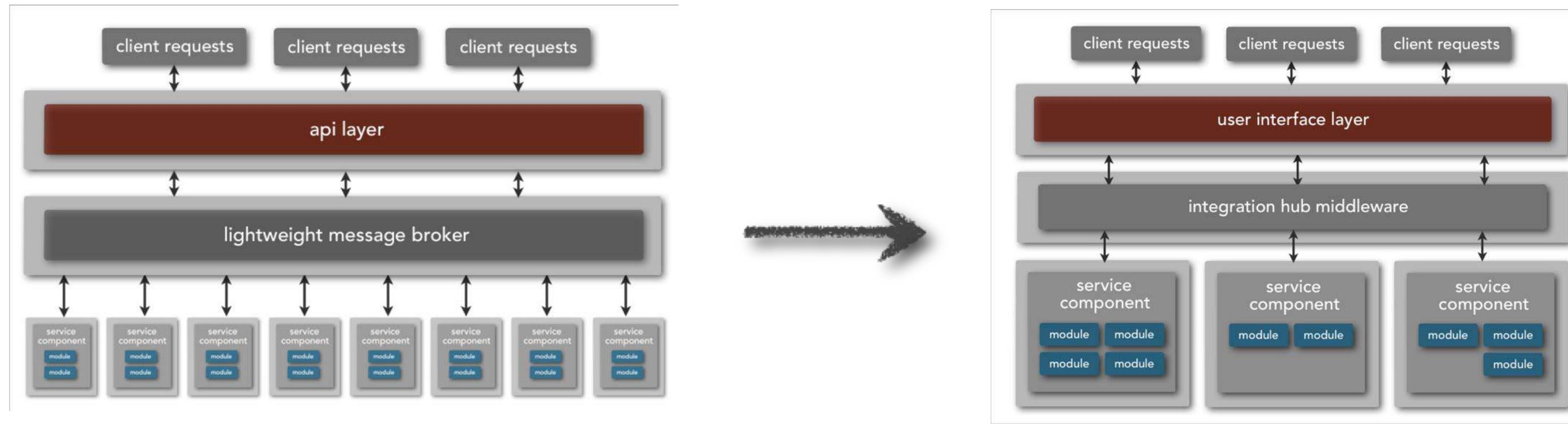
adding integration hub, towards mediator



decrease in overall performance

service-based architecture

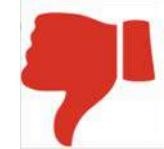
integration hub



adding integration hub, towards mediator



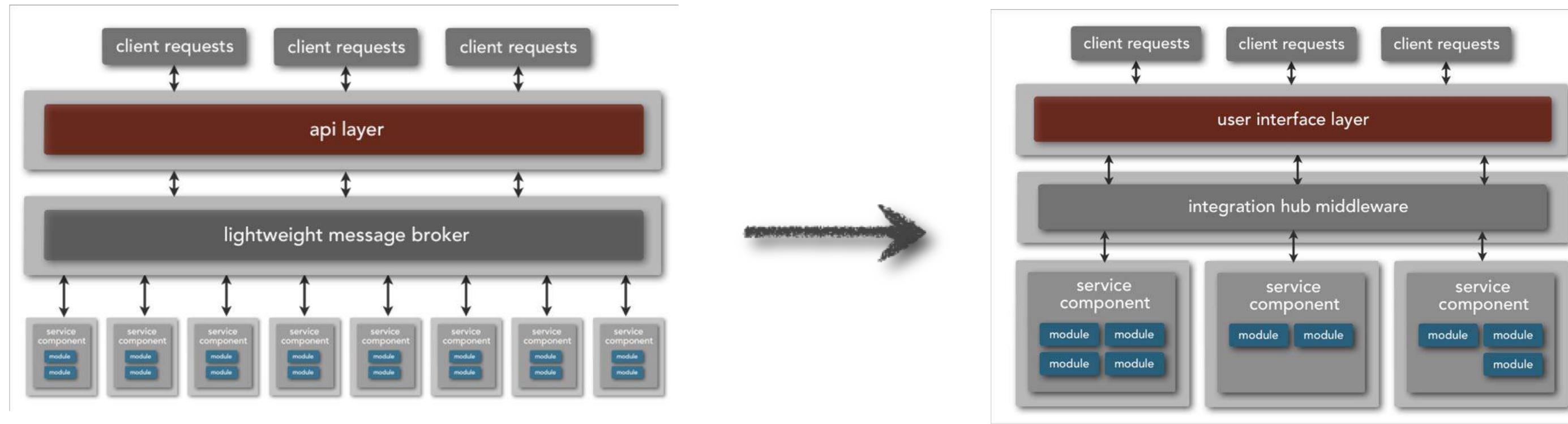
decrease in overall performance



added complexity and cost

service-based architecture

integration hub

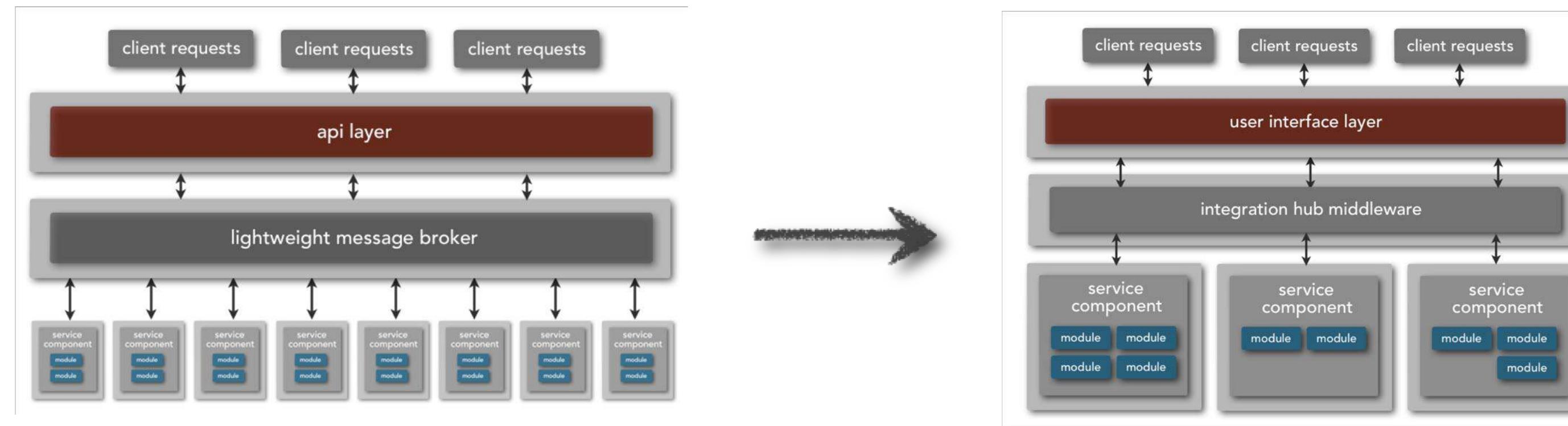


adding integration hub, towards mediator

- 👎 decrease in overall performance
- 👎 added complexity and cost
- 👎 increased need for governance

service-based architecture

integration hub

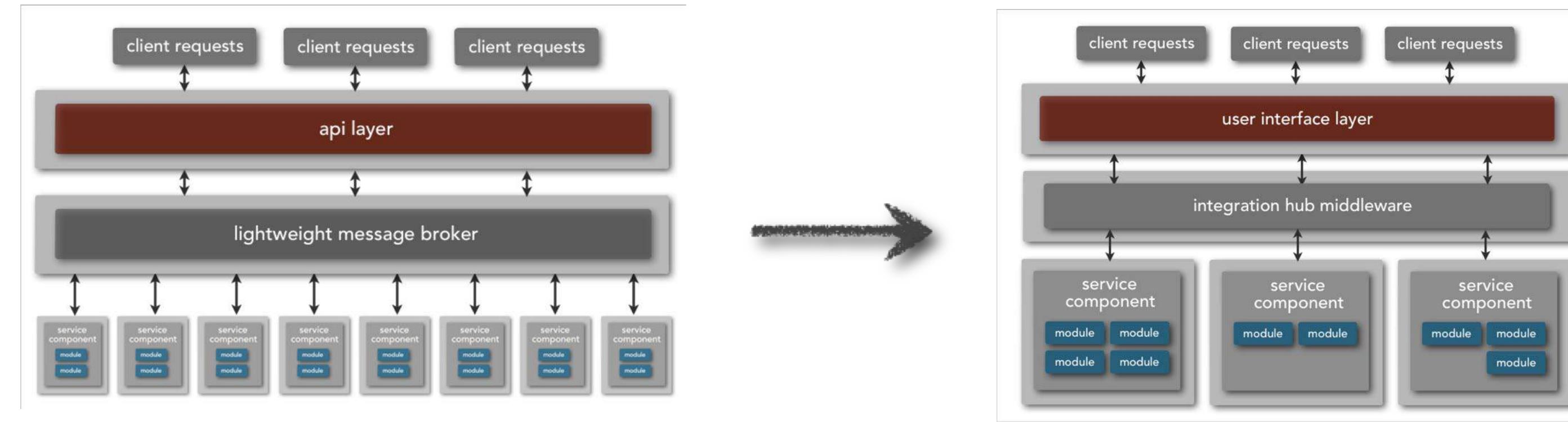


adding integration hub, towards mediator

- 👎 decrease in overall performance
- 👎 added complexity and cost
- 👎 increased need for governance
- 👎 deployment pipeline requires much more planning

service-based architecture

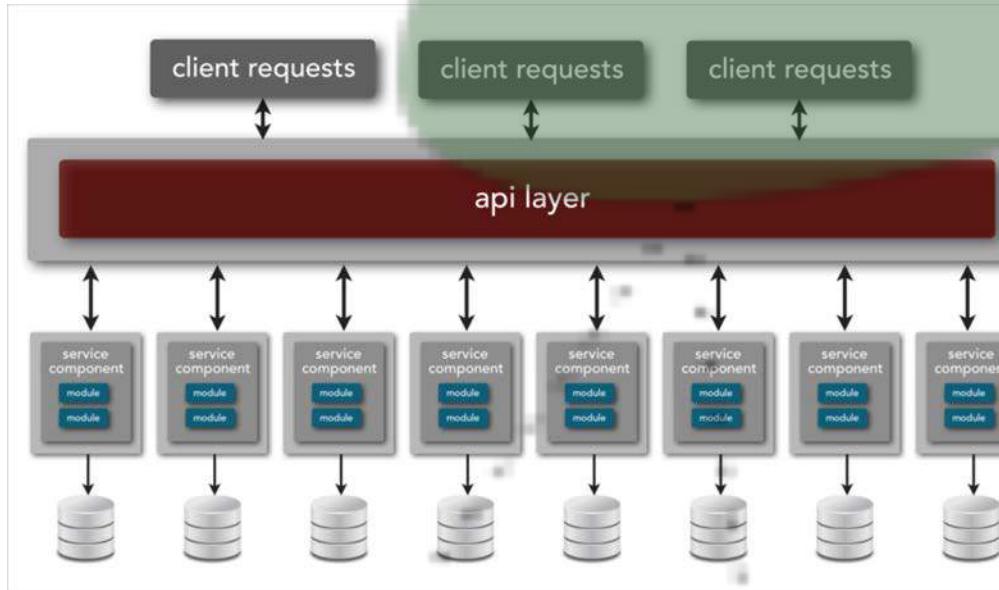
integration hub



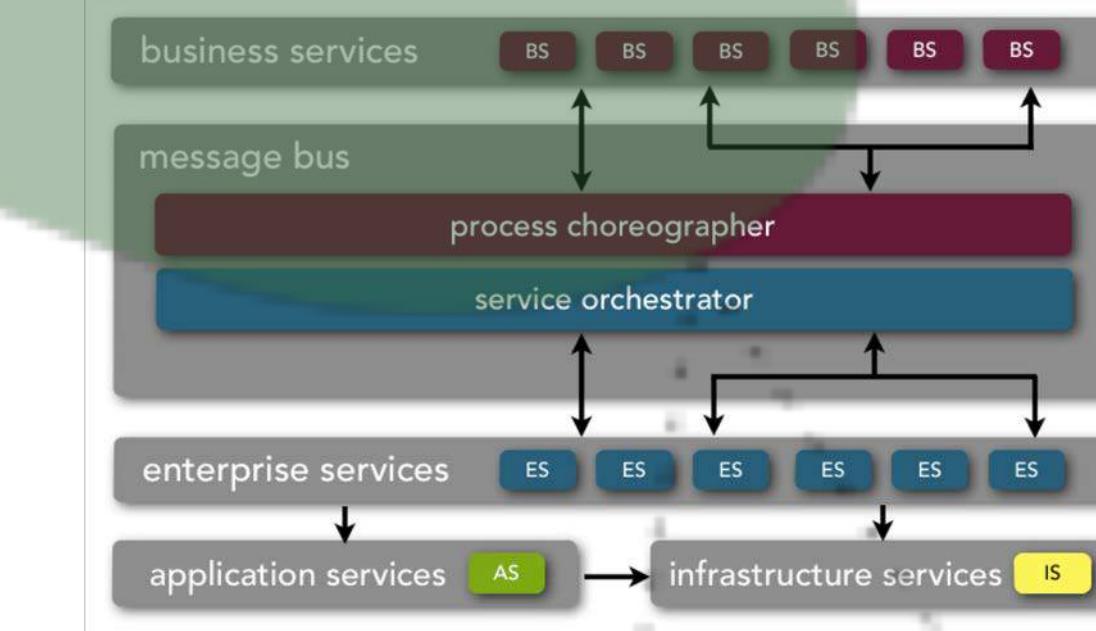
adding integration hub, towards mediator

- 👎 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

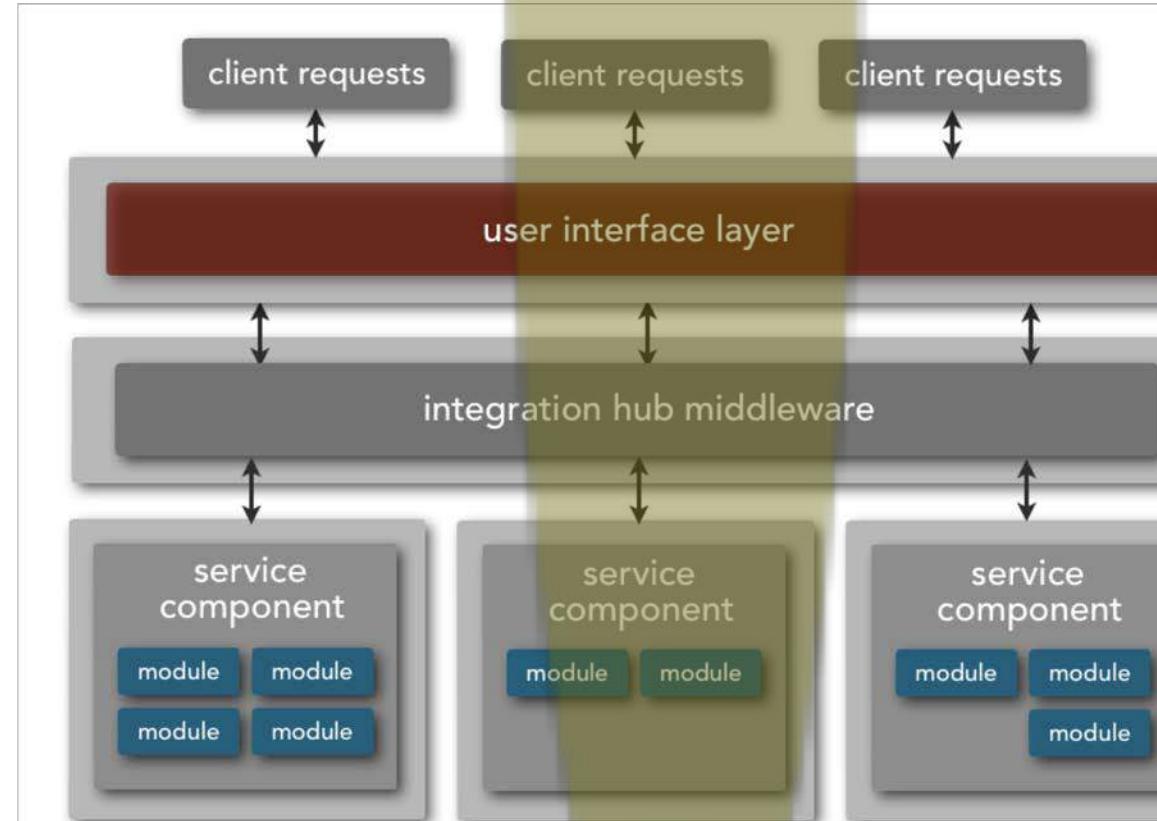
comparing:



Micro



Service-oriented

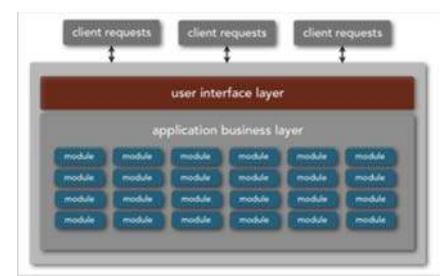


Service-based

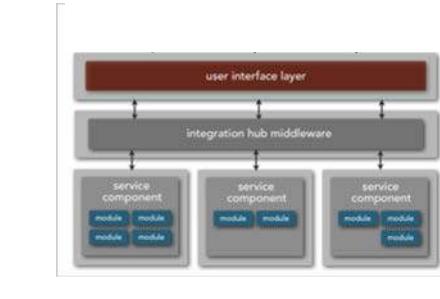
characteristics differences

overall agility

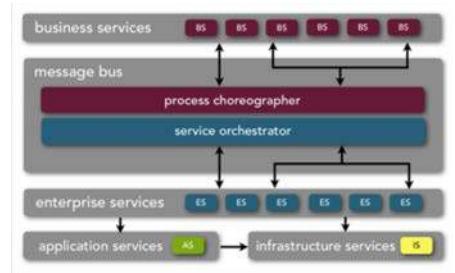
ability to respond quickly to constant change in both business and technology



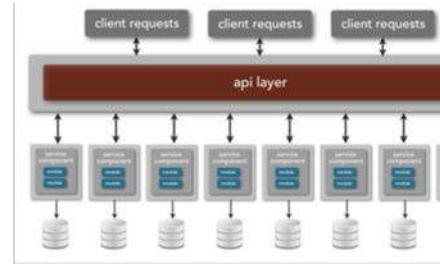
monolithic architecture



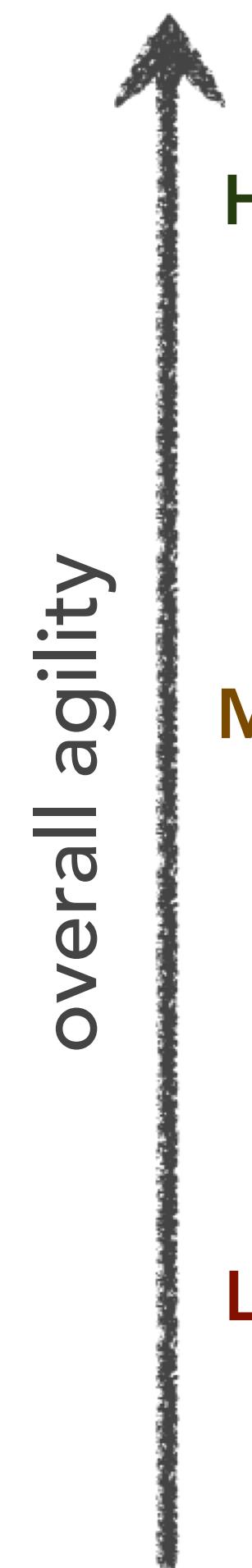
service-based architecture



service-oriented architecture



microservices architecture



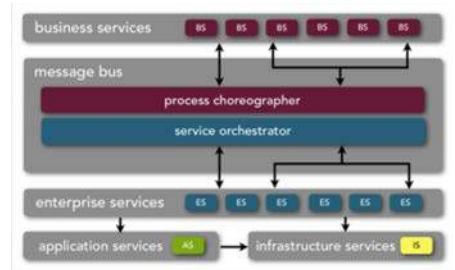
characteristics differences

overall agility

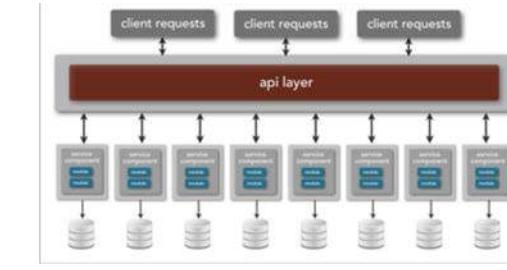
ability to respond quickly to constant change in both business and technology



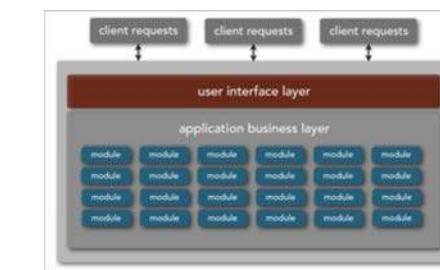
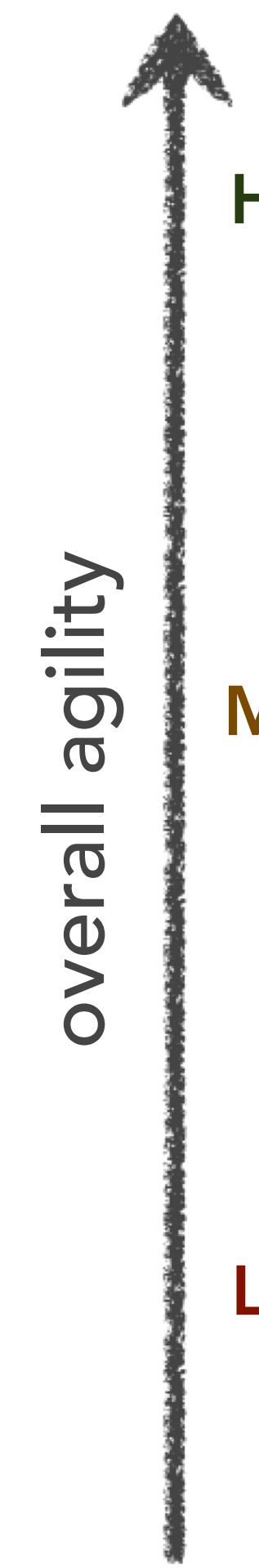
service-based architecture



service-oriented architecture



microservices architecture

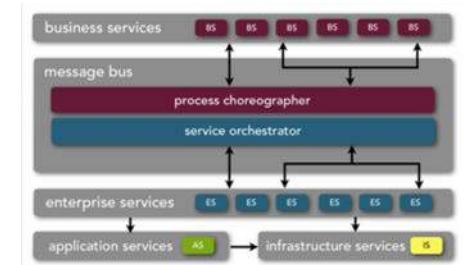


monolithic architecture

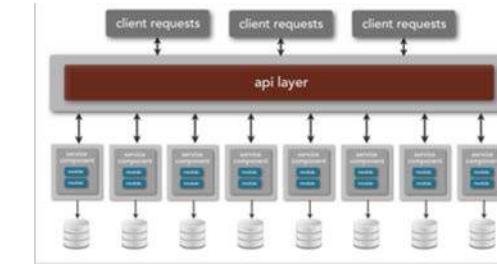
characteristics differences

overall agility

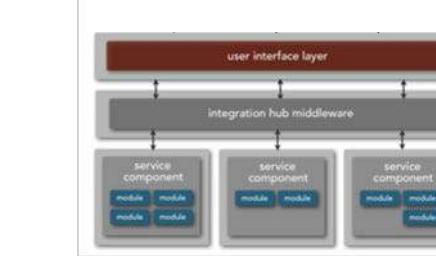
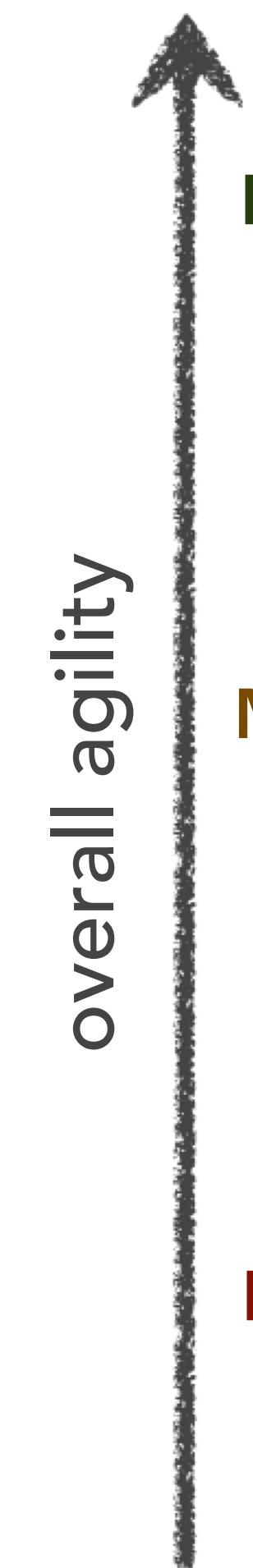
ability to respond quickly to constant change in both business and technology



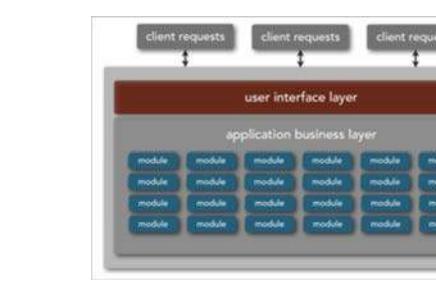
service-oriented
architecture



microservices
architecture



service-based
architecture

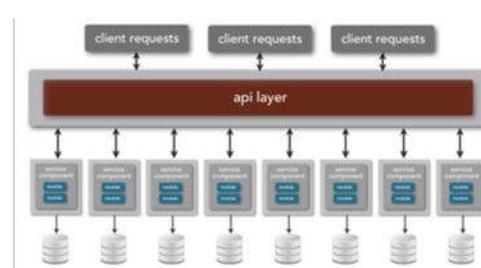


monolithic
architecture

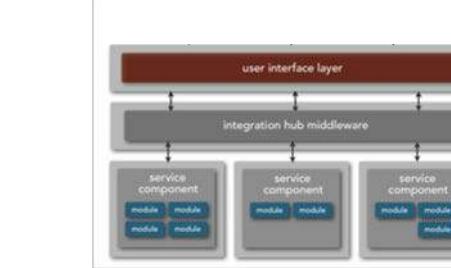
characteristics differences

overall agility

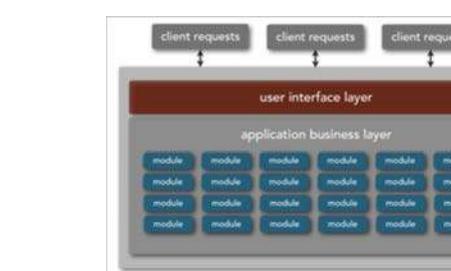
ability to respond quickly to constant change in both business and technology



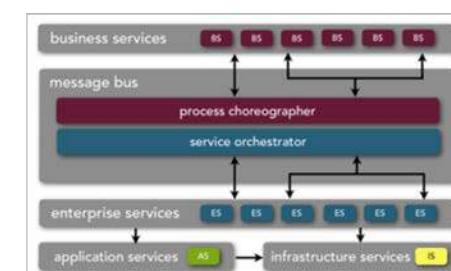
microservices
architecture



service-based
architecture



monolithic
architecture

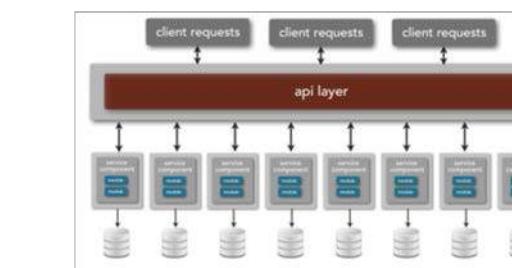
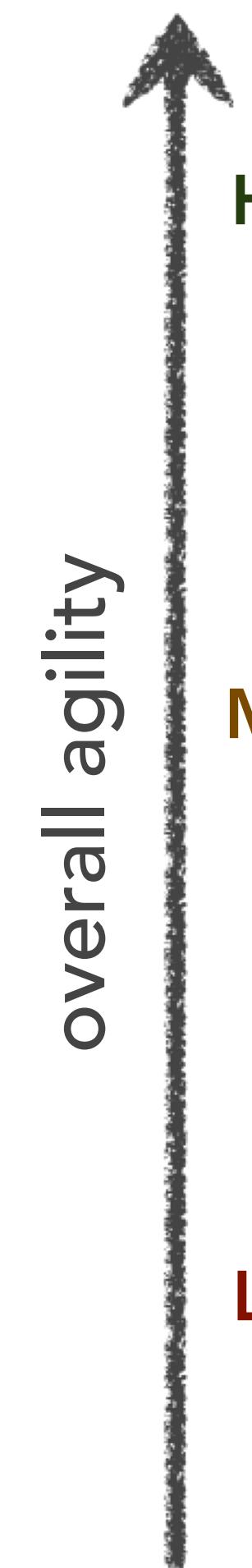


service-oriented
architecture

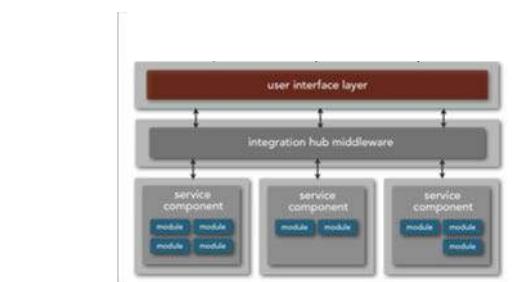
characteristics differences

overall agility

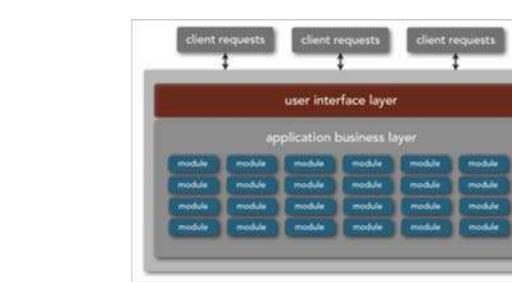
ability to respond quickly to constant change in both business and technology



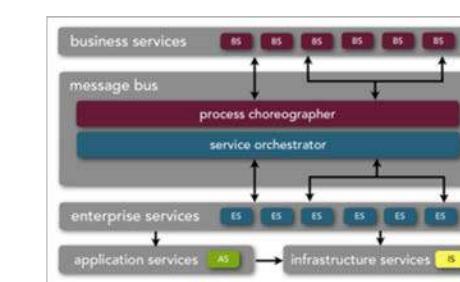
microservices
architecture



service-based
architecture



monolithic
architecture

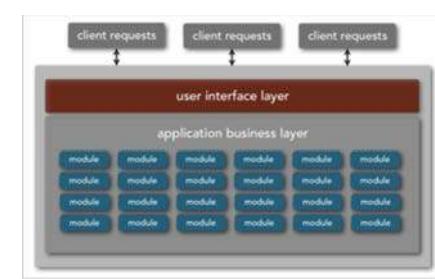


service-oriented
architecture

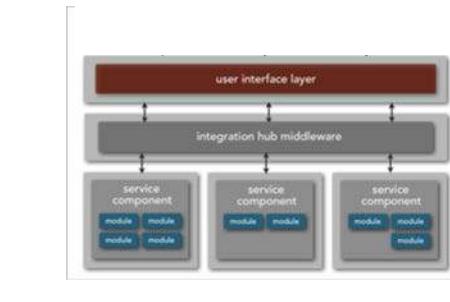
characteristics differences

ease of deployment

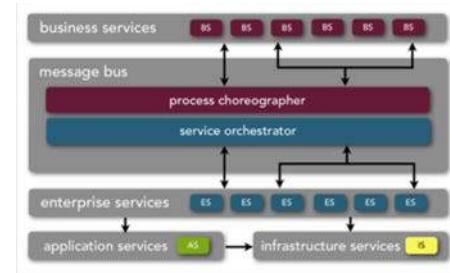
promotes an effective and fast deployment pipeline; features are quick and easy to deploy



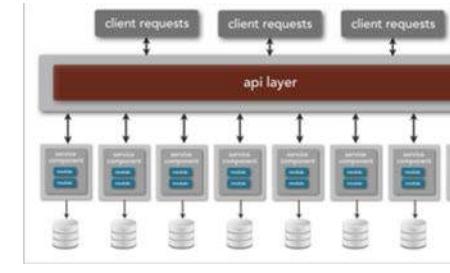
monolithic
architecture



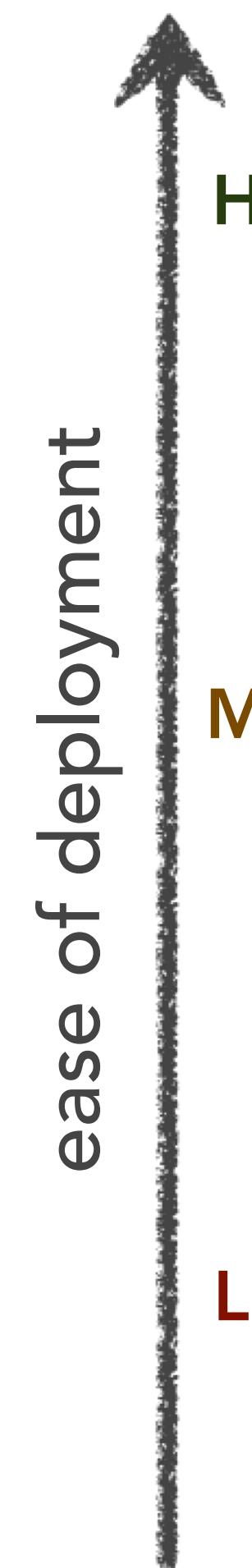
service-based
architecture



service-oriented
architecture



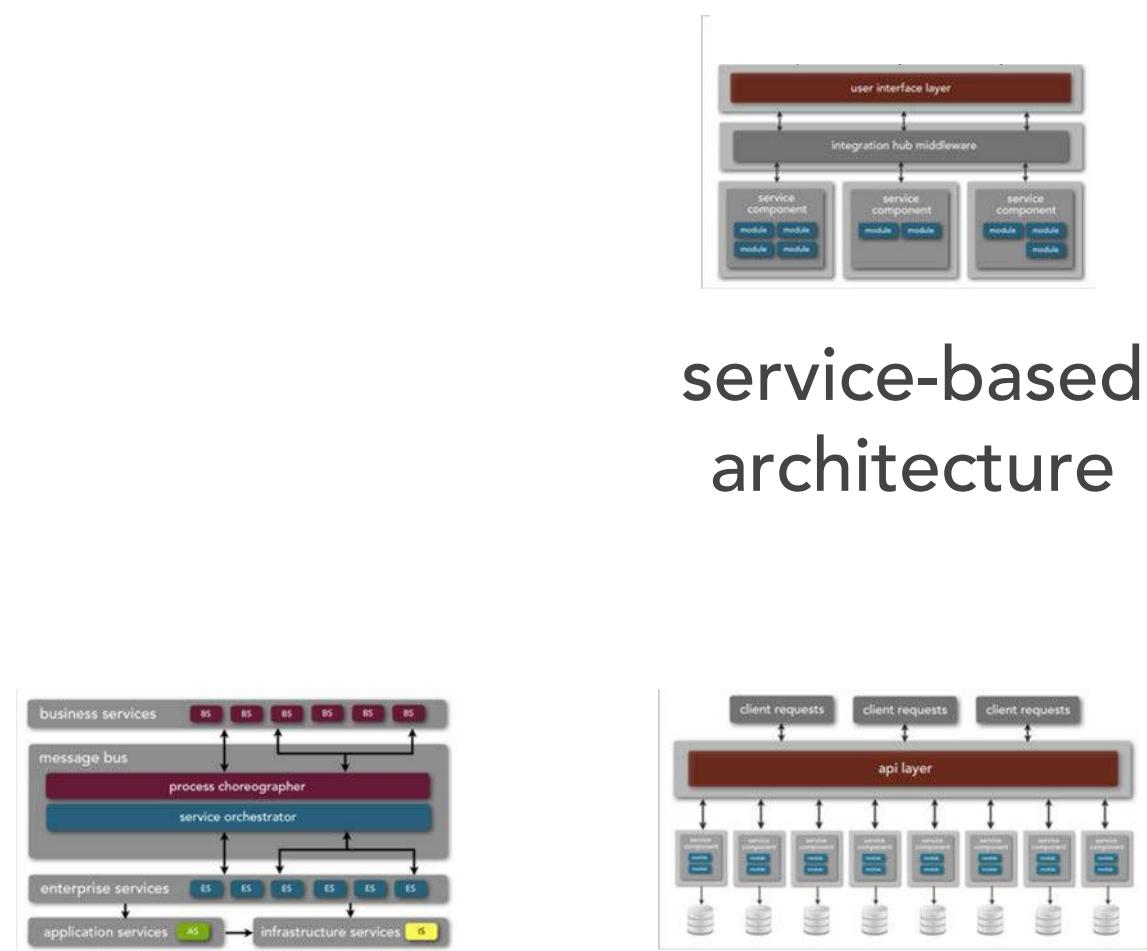
microservices
architecture



characteristics differences

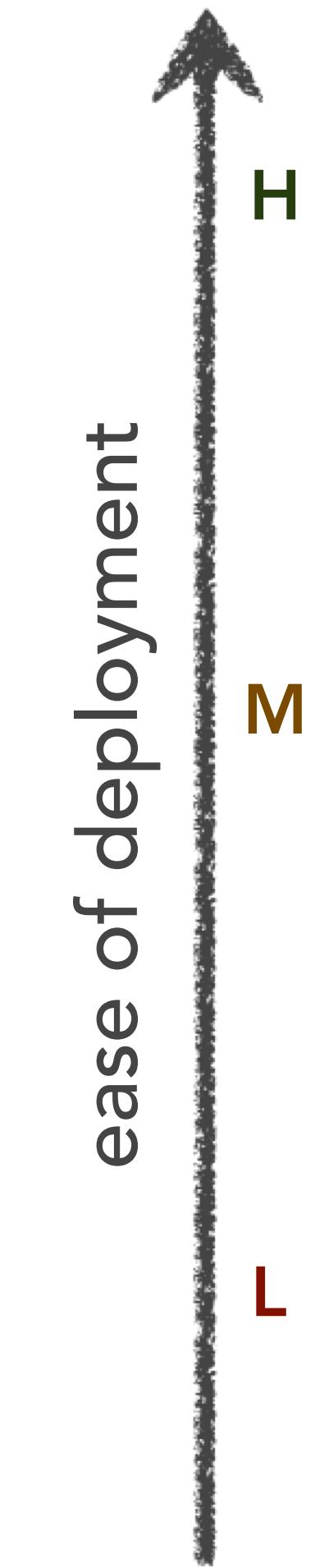
ease of deployment

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



service-oriented
architecture

microservices
architecture

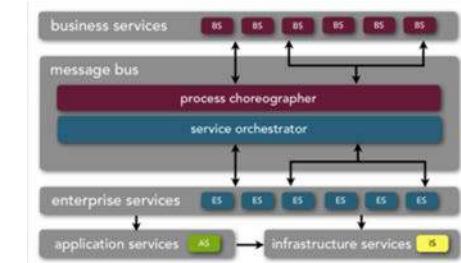


monolithic
architecture

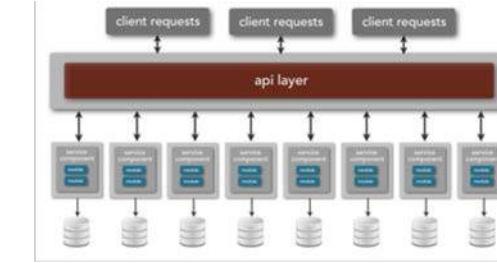
characteristics differences

ease of deployment

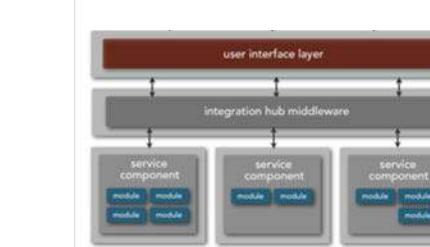
promotes an effective and fast deployment pipeline; features are quick and easy to deploy



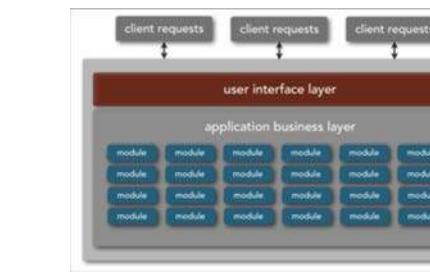
service-oriented
architecture



microservices
architecture



service-based
architecture

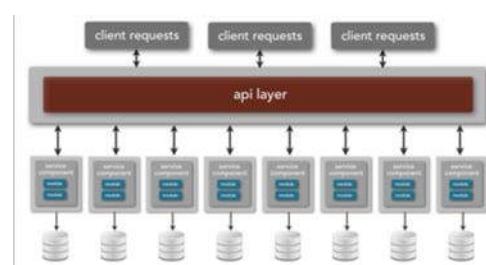


monolithic
architecture

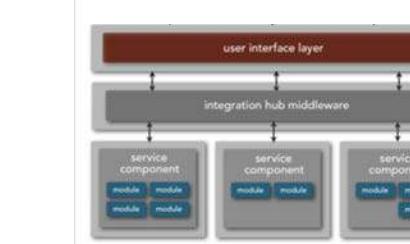
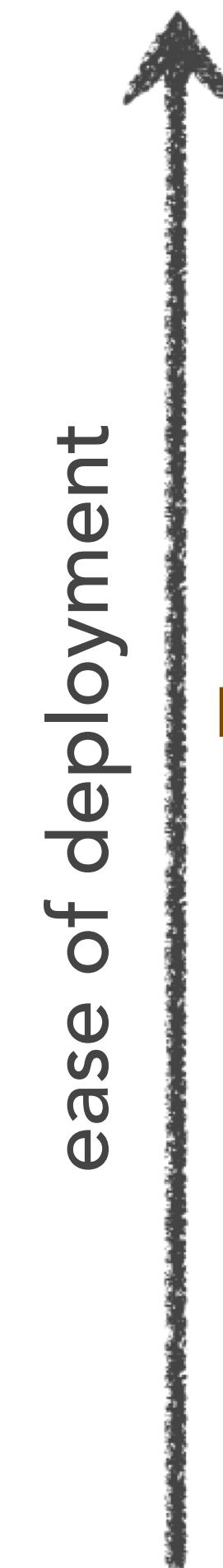
characteristics differences

ease of deployment

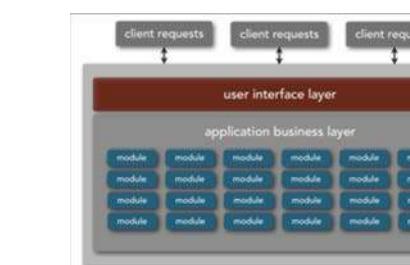
promotes an effective and fast deployment pipeline; features are quick and easy to deploy



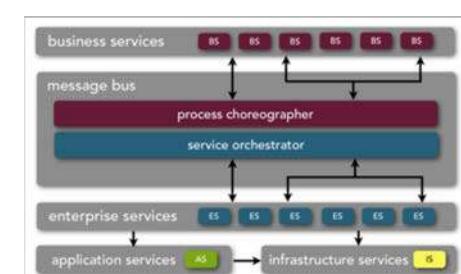
microservices
architecture



service-based
architecture



monolithic
architecture

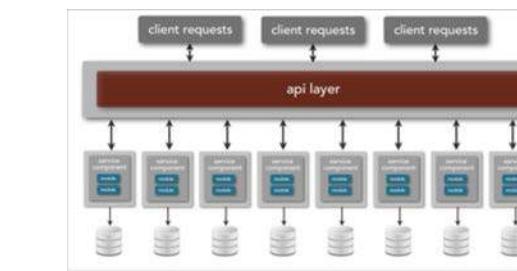


service-oriented
architecture

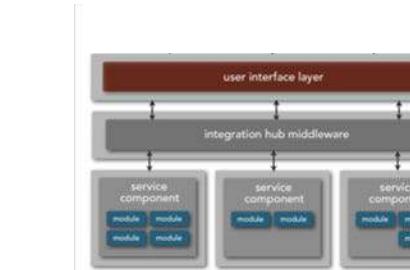
characteristics differences

ease of deployment

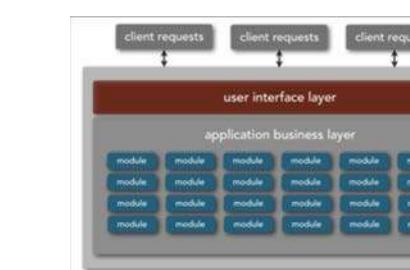
promotes an effective and fast deployment pipeline; features are quick and easy to deploy



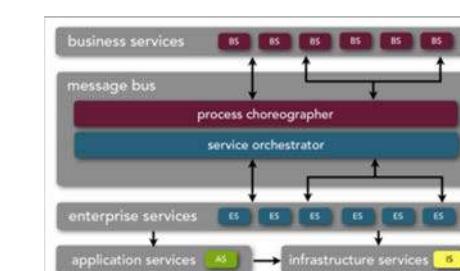
microservices
architecture



service-based
architecture



monolithic
architecture

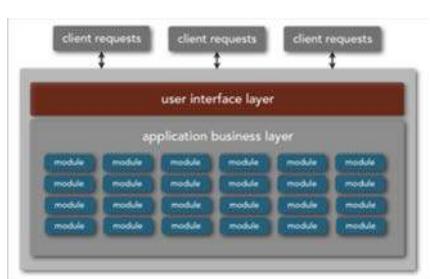


service-oriented
architecture

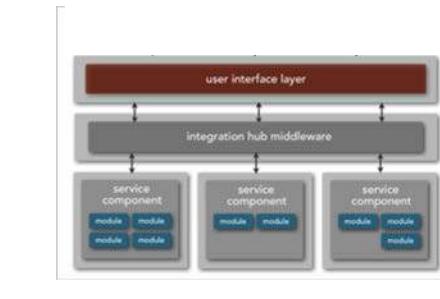
characteristics differences

ease of testing

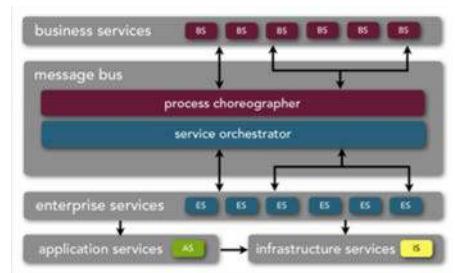
ease at which features can be tested and verified; confidence level in completeness of testing



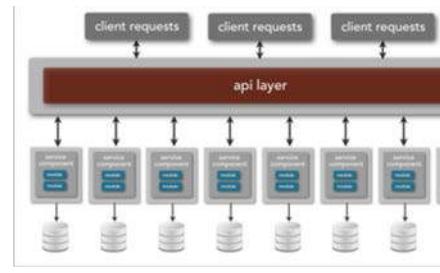
monolithic
architecture



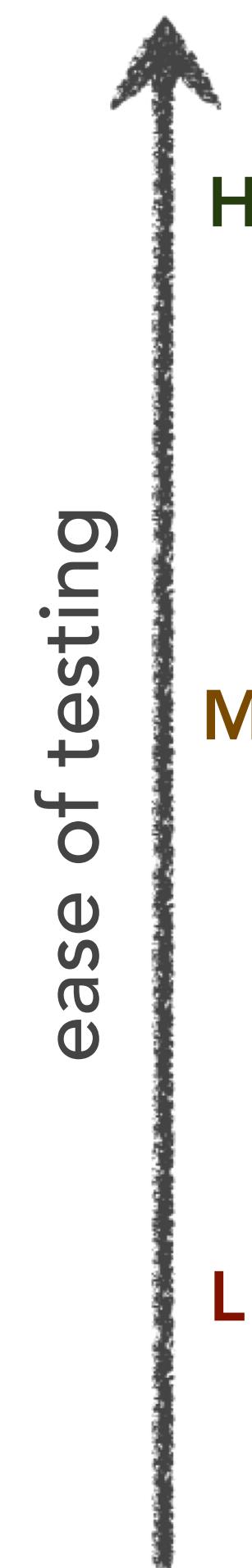
service-based
architecture



service-oriented
architecture



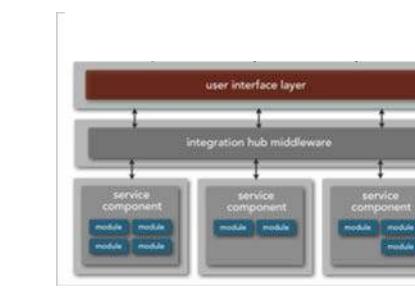
microservices
architecture



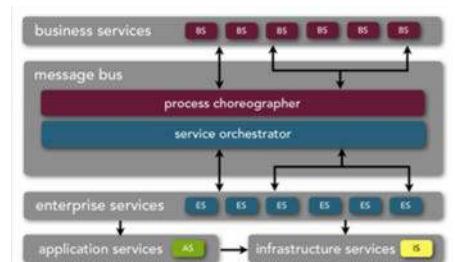
characteristics differences

ease of testing

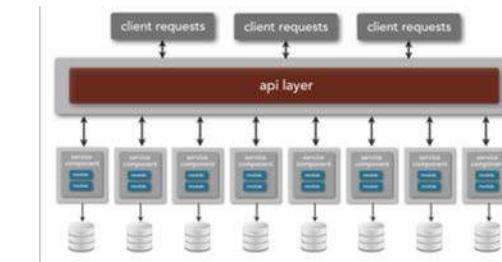
ease at which features can be tested and verified; confidence level in completeness of testing



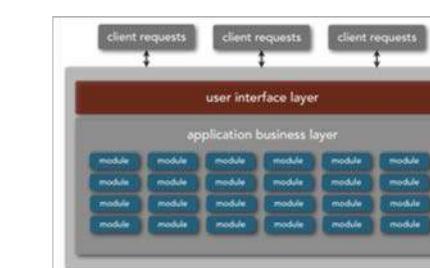
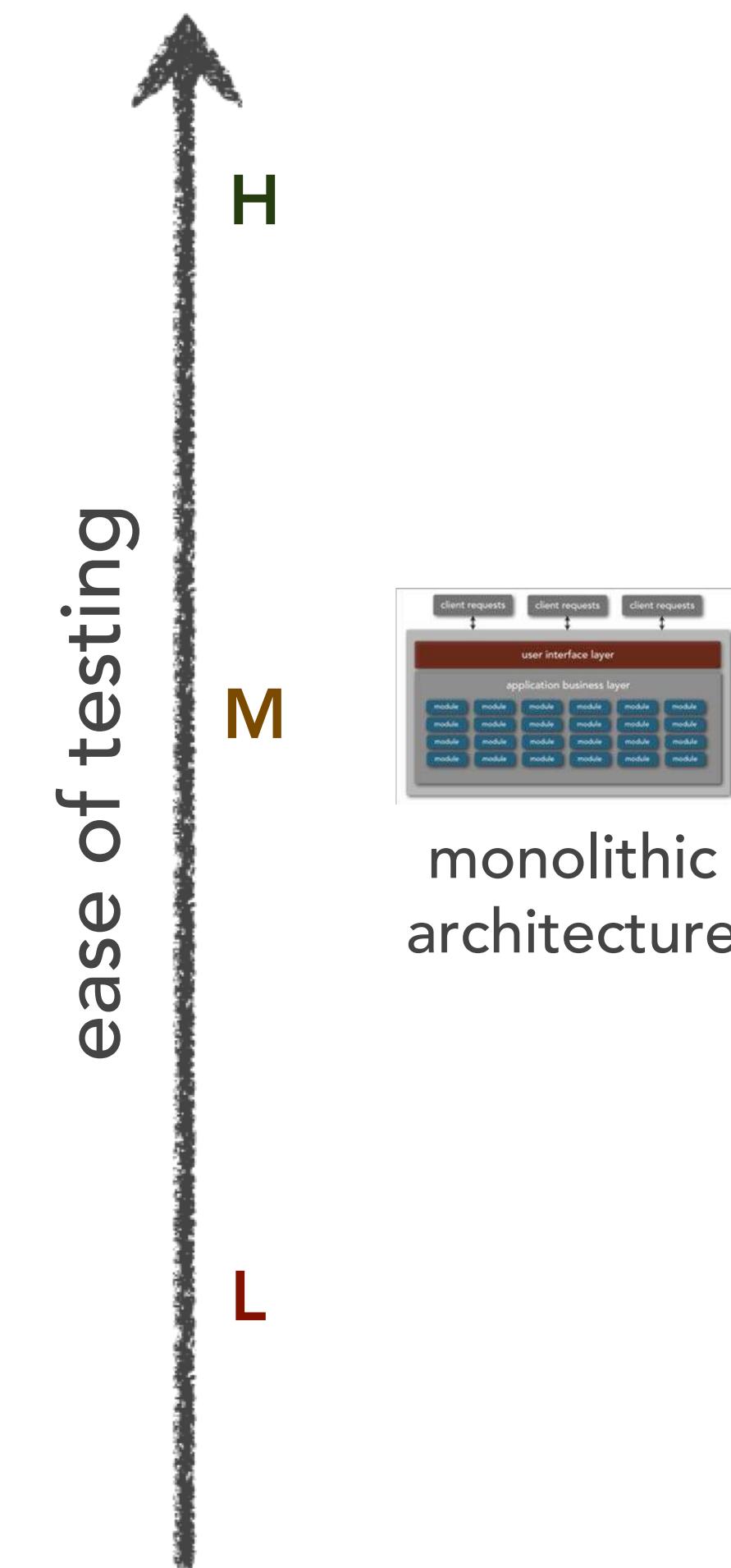
service-based architecture



service-oriented architecture



microservices architecture

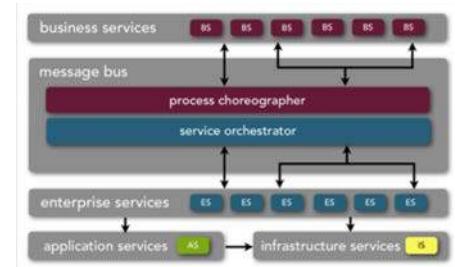


monolithic architecture

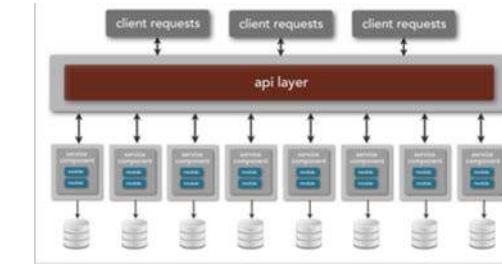
characteristics differences

ease of testing

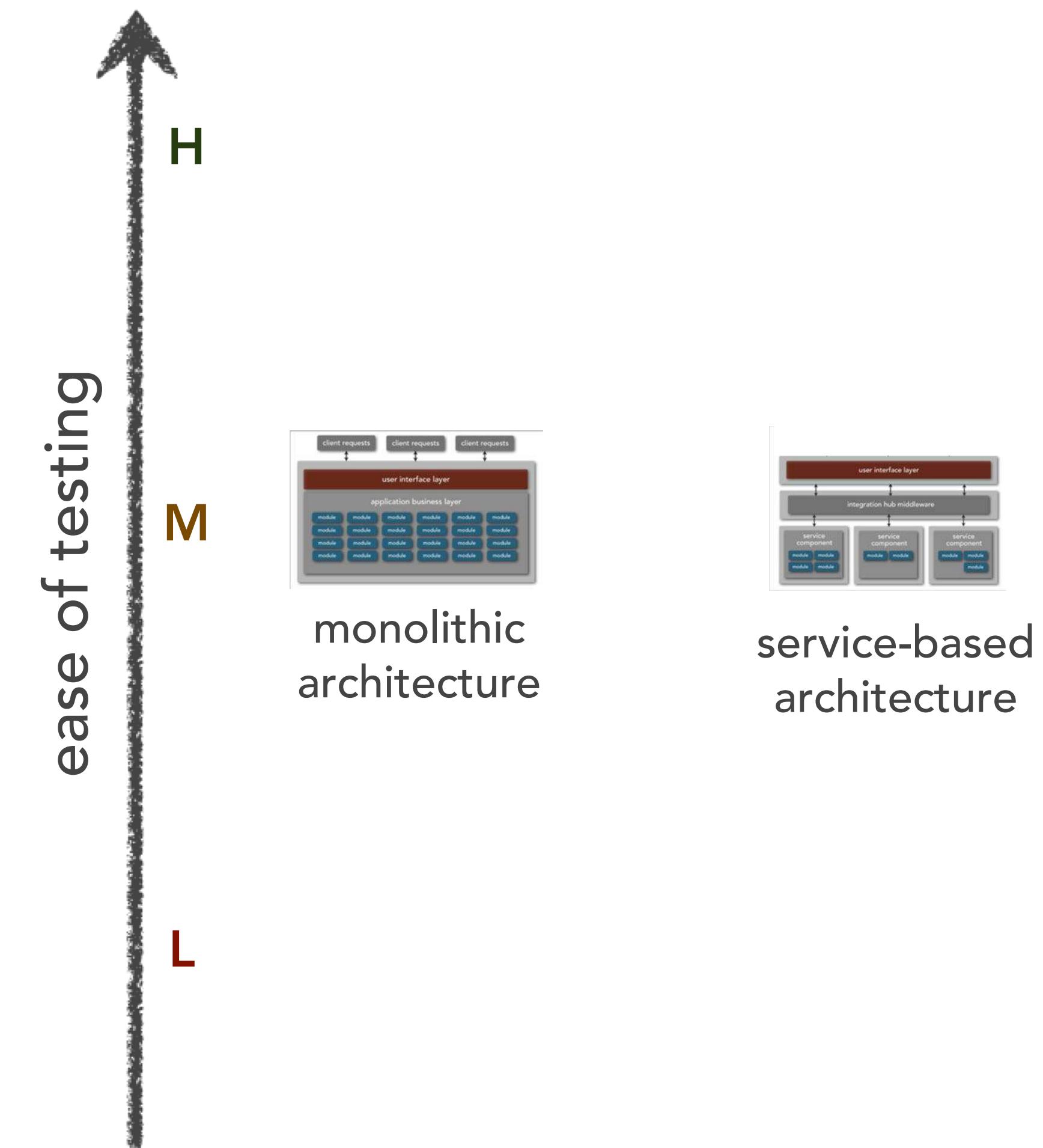
ease at which features can be tested and verified; confidence level in completeness of testing



service-oriented
architecture



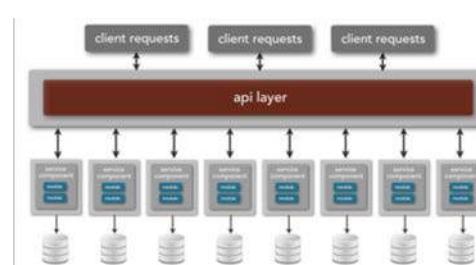
microservices
architecture



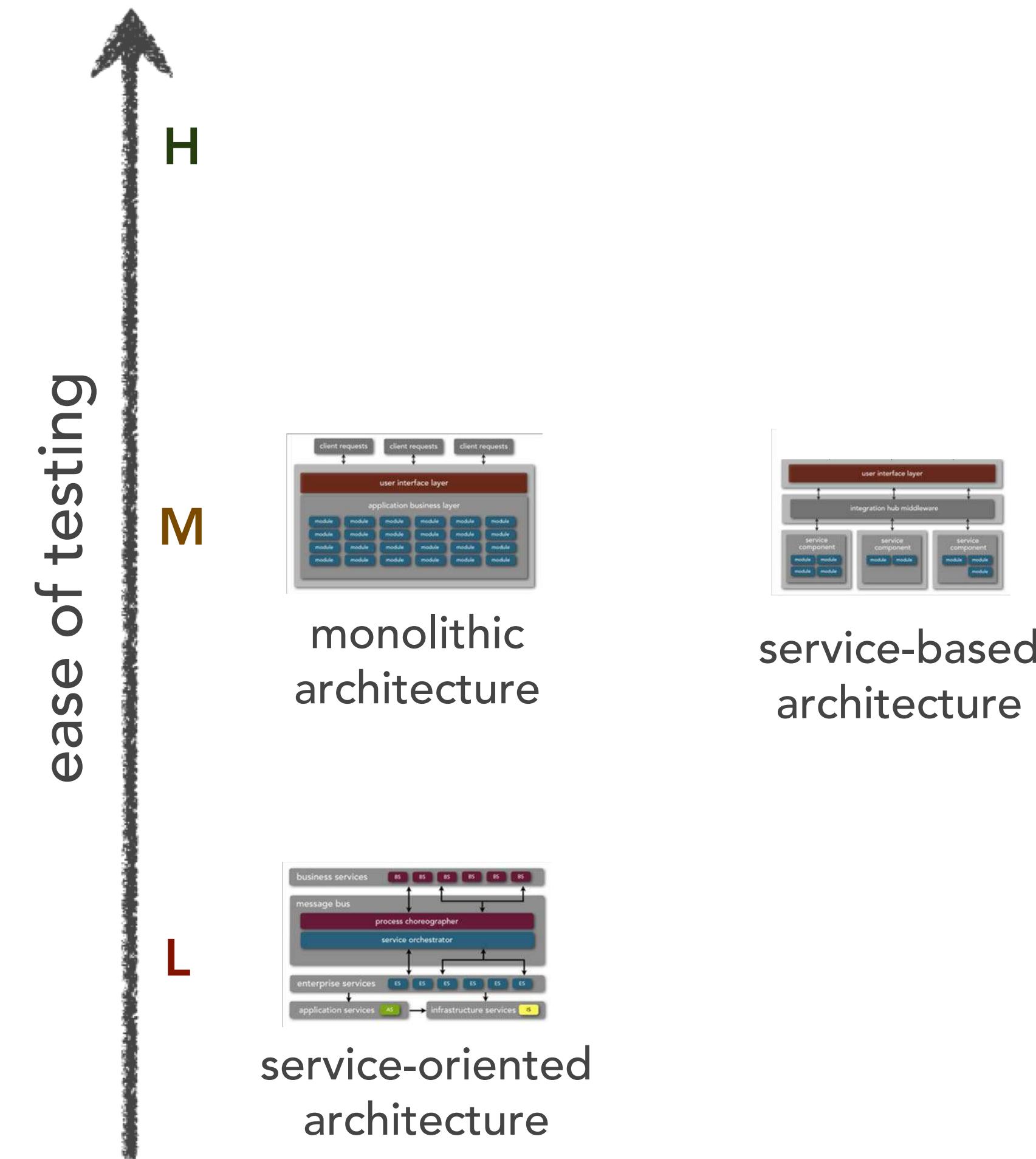
characteristics differences

ease of testing

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



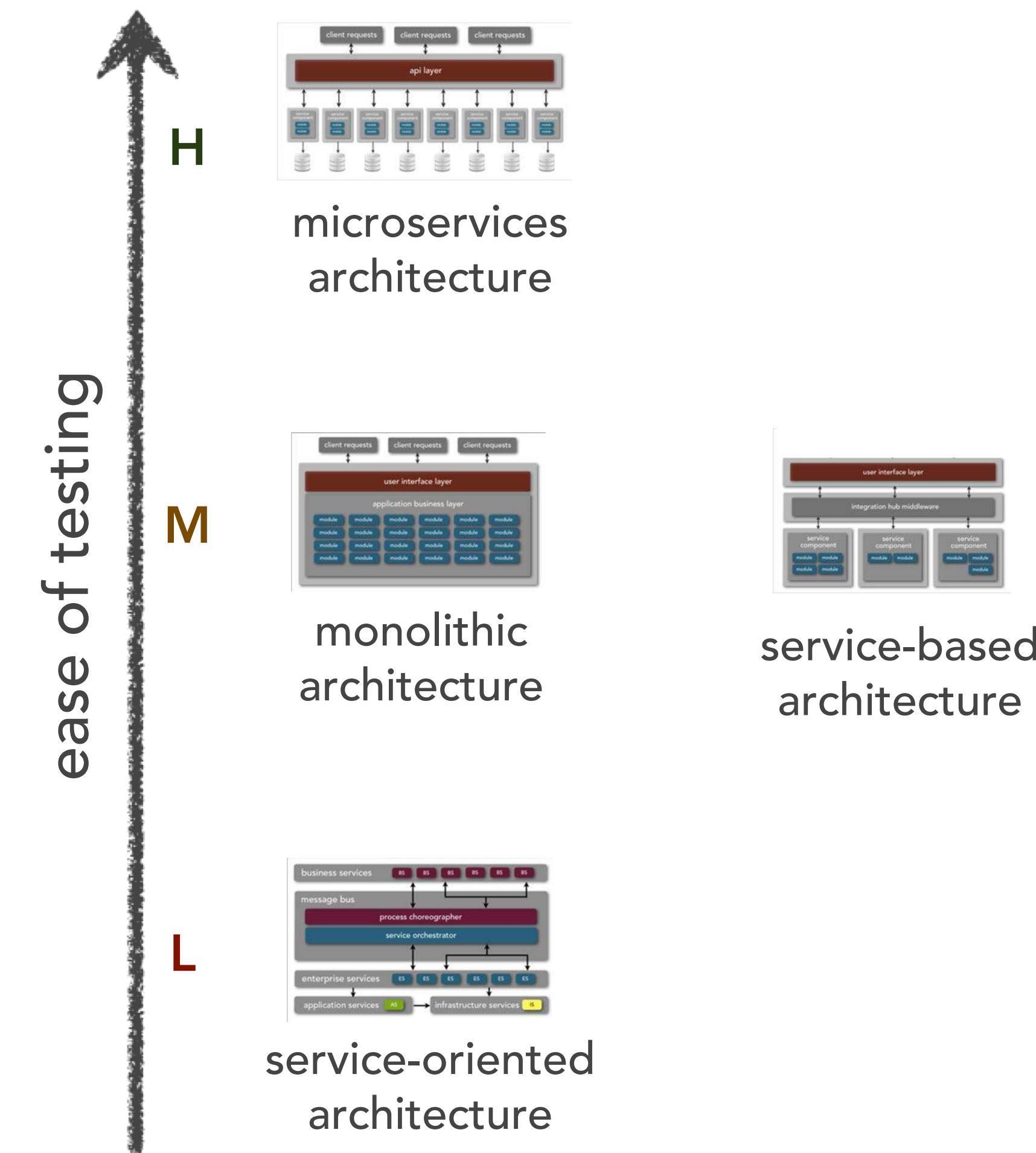
microservices
architecture



characteristics differences

ease of testing

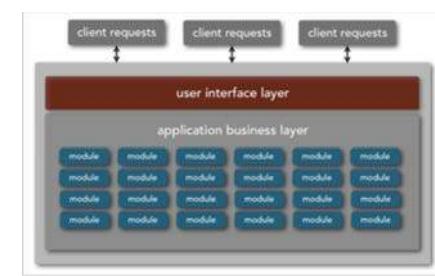
ease at which features can be tested and verified; confidence level in completeness of testing



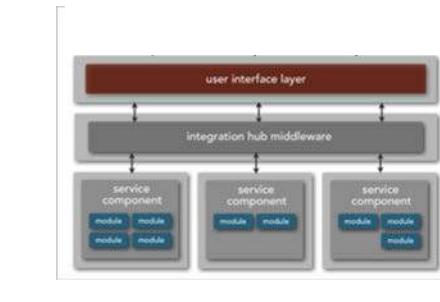
characteristics differences

overall performance

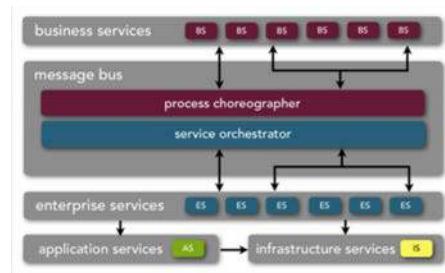
which patterns relatively promote better performing applications?



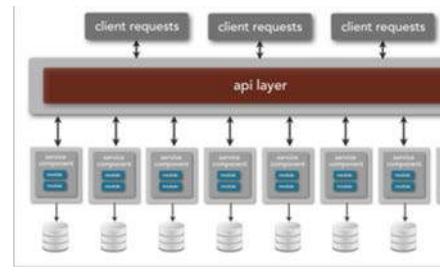
monolithic
architecture



service-based
architecture



service-oriented
architecture



microservices
architecture



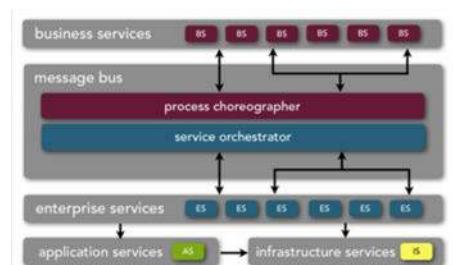
characteristics differences

overall performance

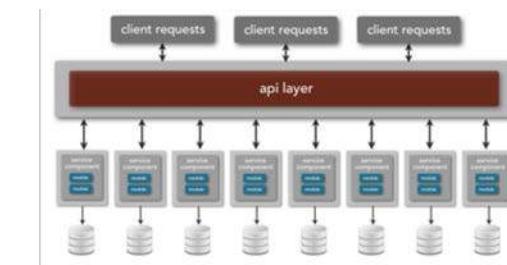
which patterns relatively promote better performing applications?



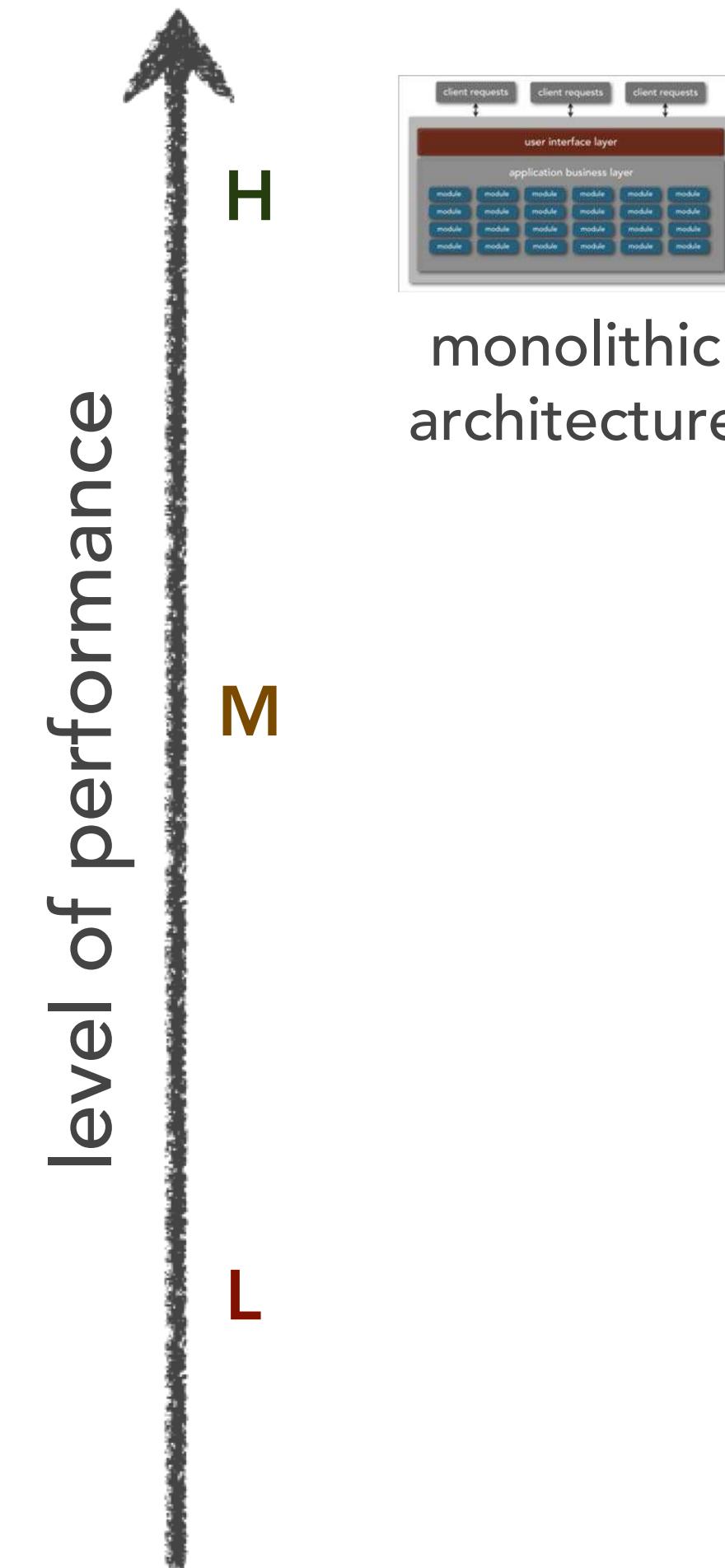
service-based architecture



service-oriented architecture



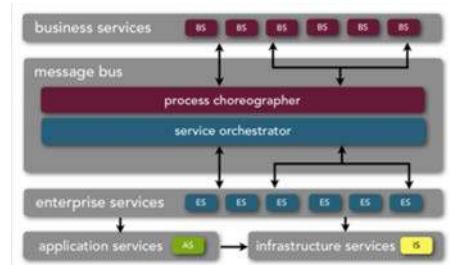
microservices architecture



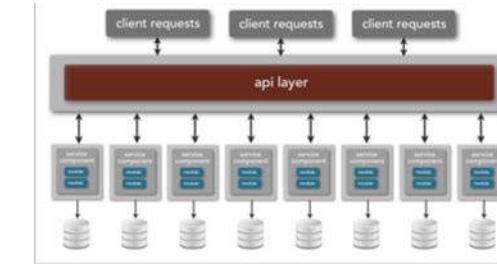
characteristics differences

overall performance

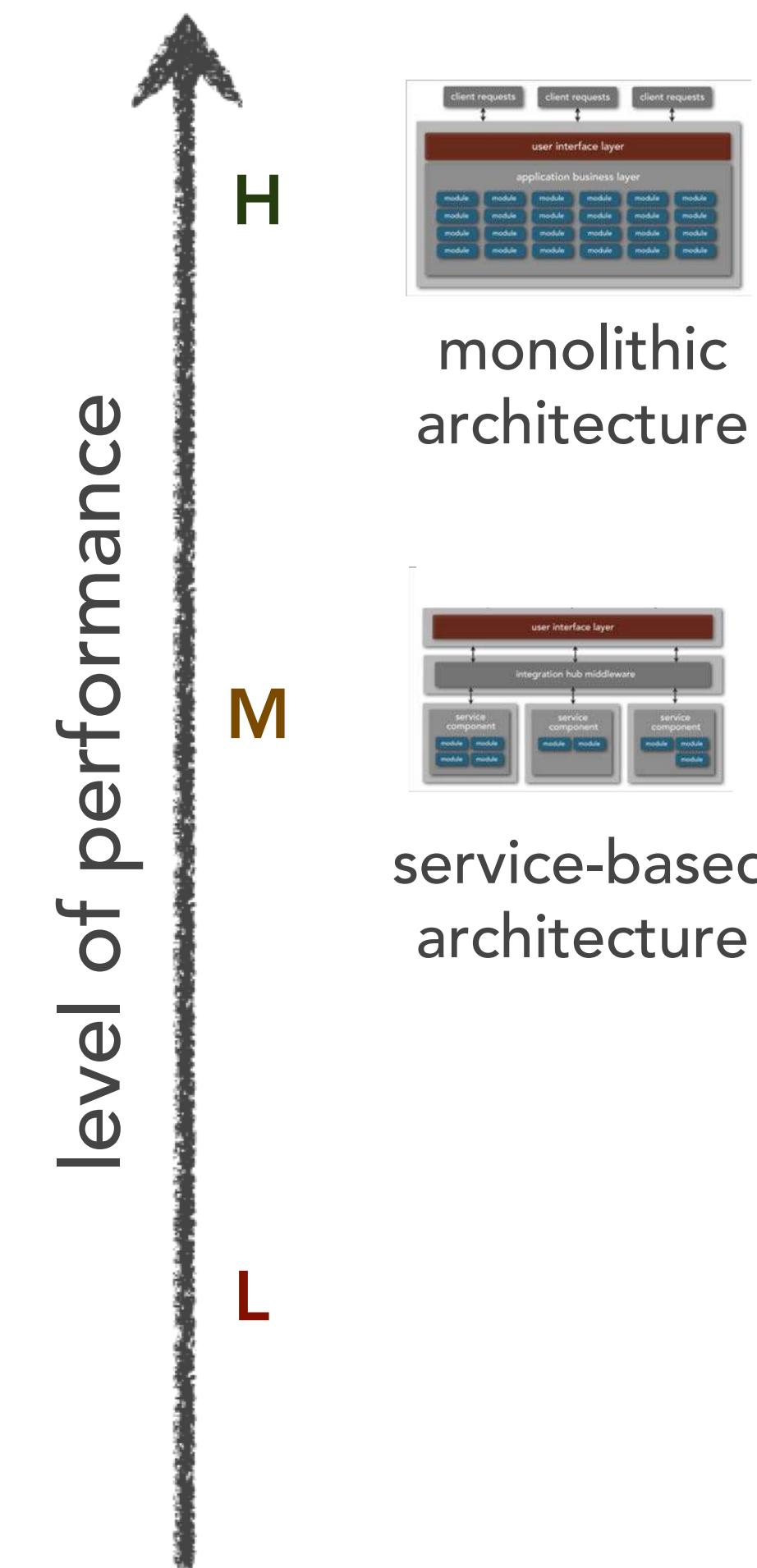
which patterns relatively promote better performing applications?



service-oriented
architecture



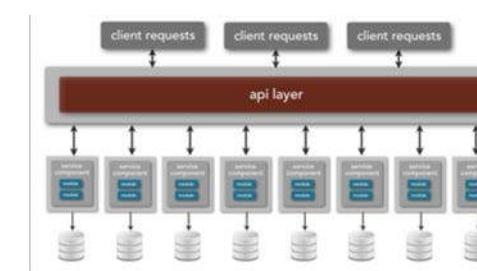
microservices
architecture



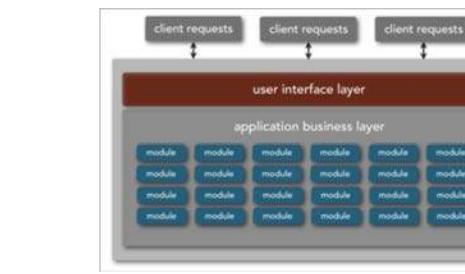
characteristics differences

overall performance

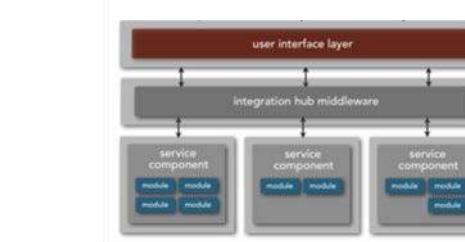
which patterns relatively promote better performing applications?



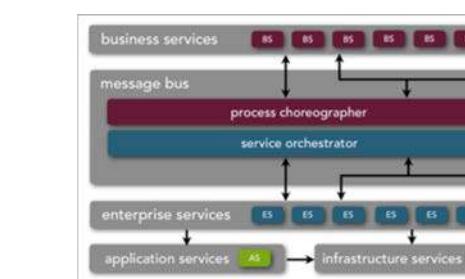
microservices
architecture



monolithic
architecture



service-based
architecture

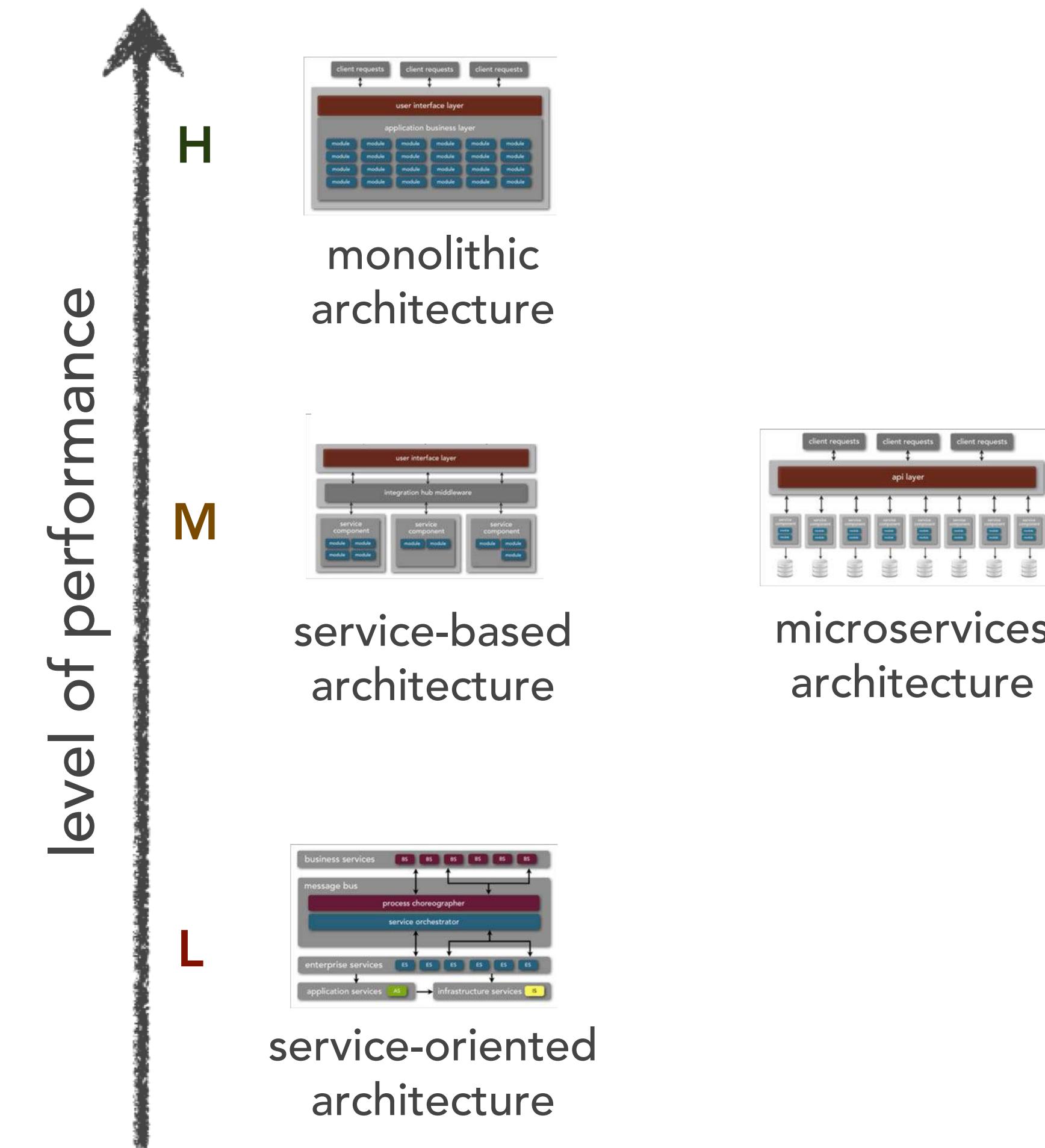


service-oriented
architecture

characteristics differences

overall performance

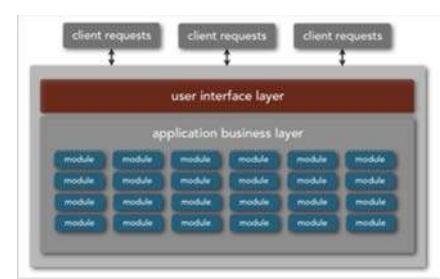
which patterns relatively promote better performing applications?



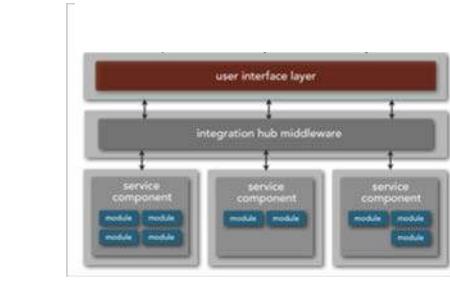
characteristics differences

overall scalability

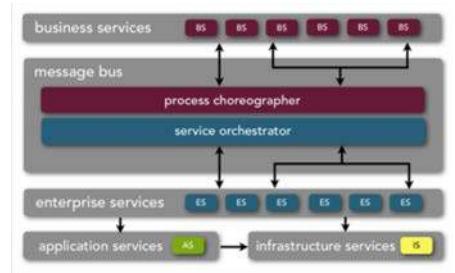
how well does the architecture pattern lend itself to highly scalable applications?



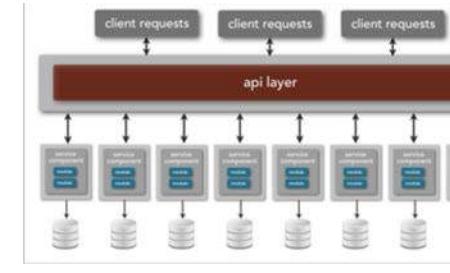
monolithic
architecture



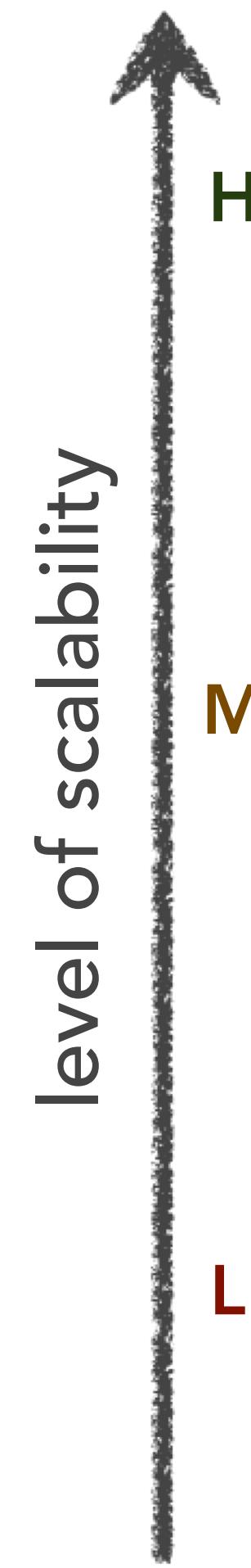
service-based
architecture



service-oriented
architecture



microservices
architecture



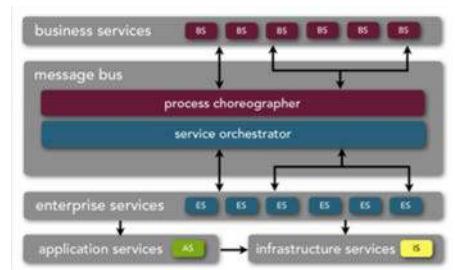
characteristics differences

overall scalability

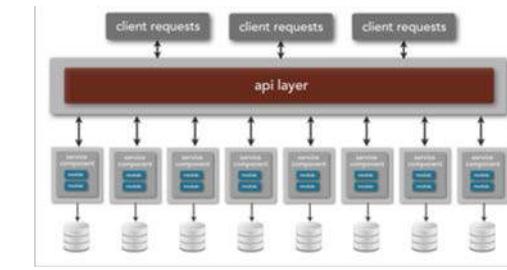
how well does the architecture pattern lend itself to highly scalable applications?



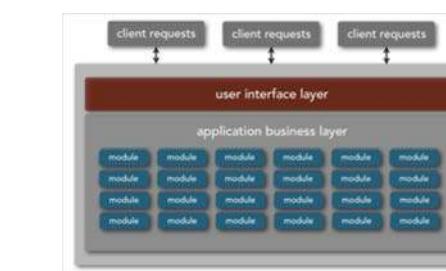
service-based
architecture



service-oriented
architecture



microservices
architecture

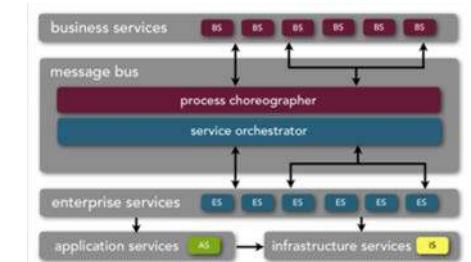


monolithic
architecture

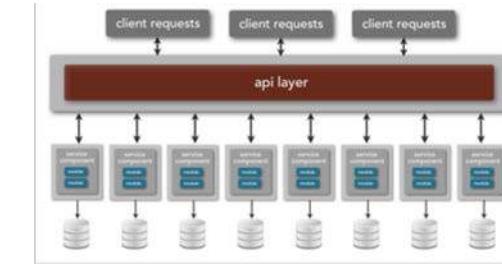
characteristics differences

overall scalability

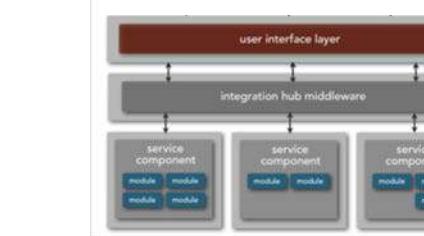
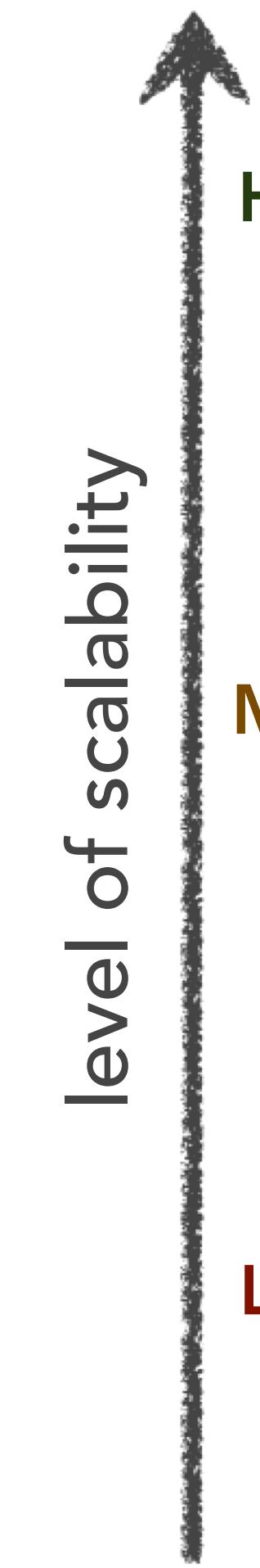
how well does the architecture pattern lend itself to highly scalable applications?



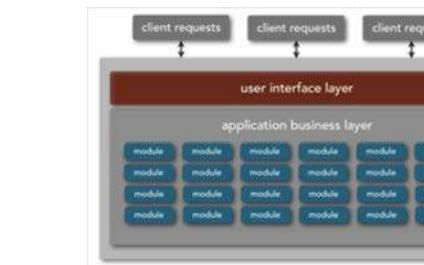
service-oriented
architecture



microservices
architecture



service-based
architecture

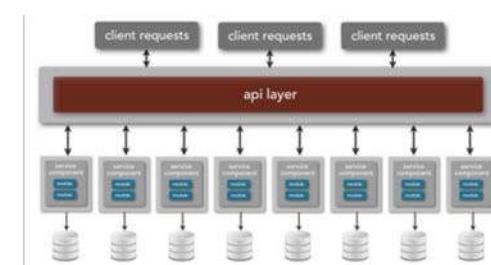


monolithic
architecture

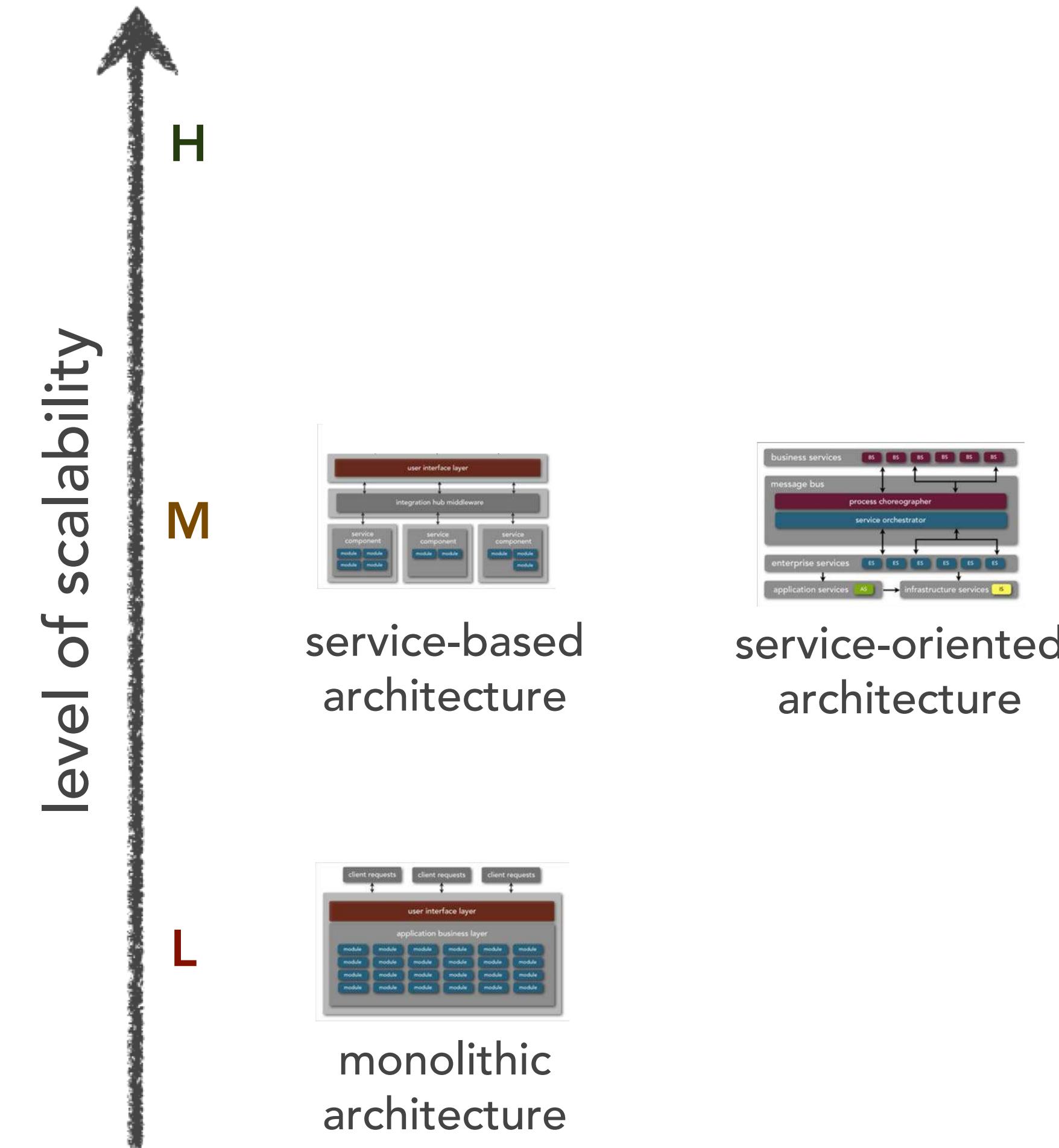
characteristics differences

overall scalability

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



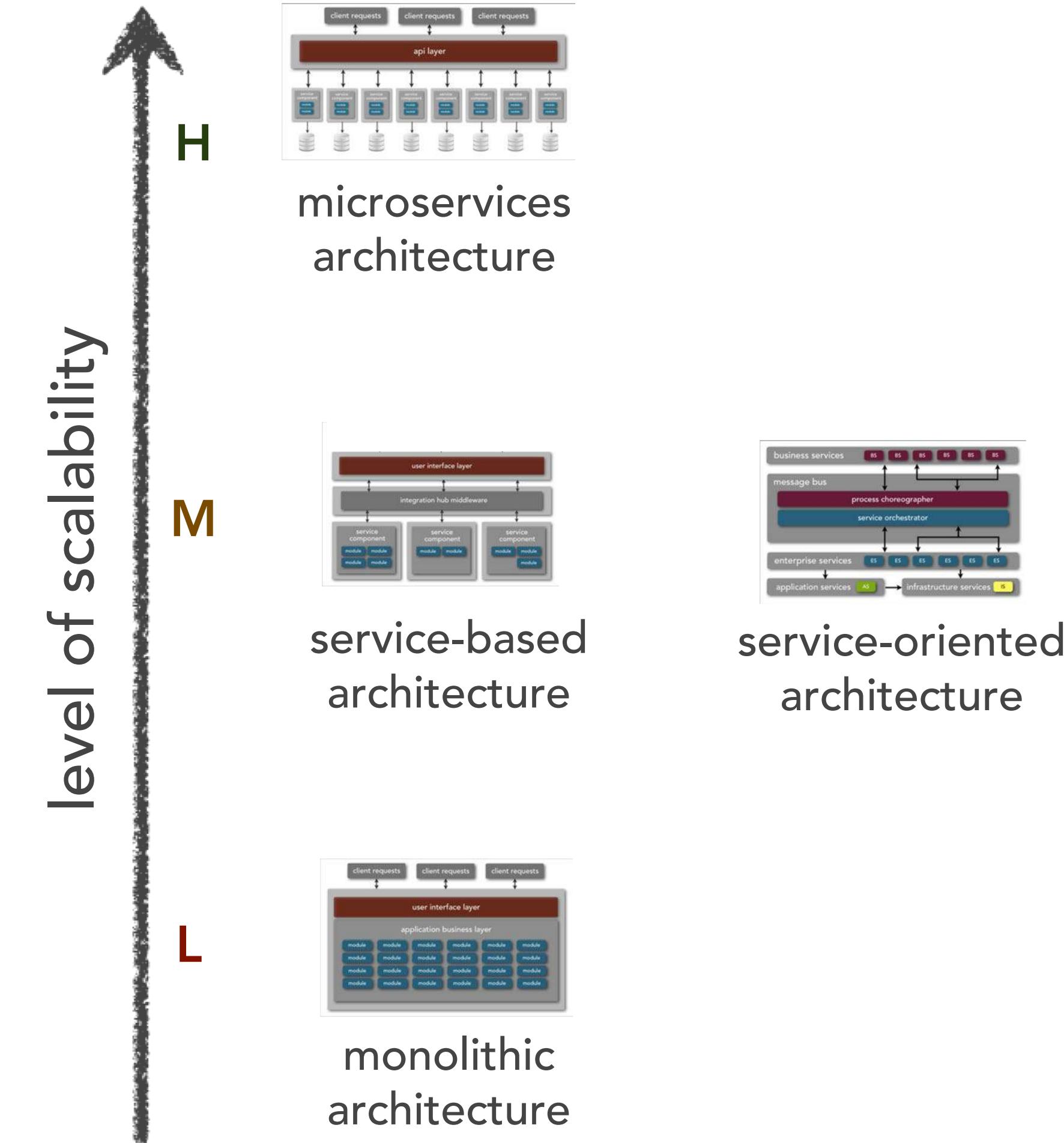
microservices
architecture



characteristics differences

overall scalability

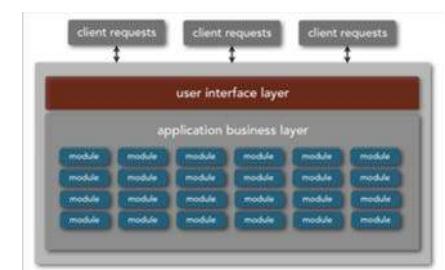
how well does the architecture pattern lend itself to highly scalable applications?



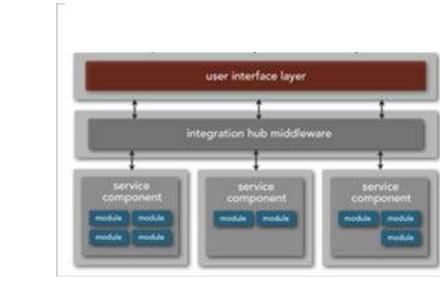
characteristics differences

overall simplicity

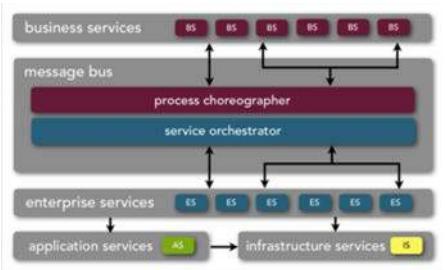
level of complexity in
applications implemented
using the architecture pattern



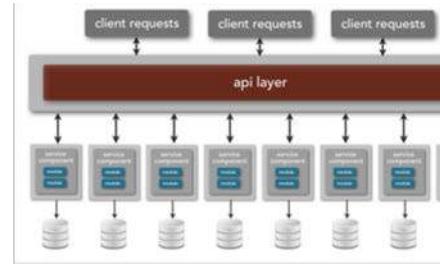
monolithic
architecture



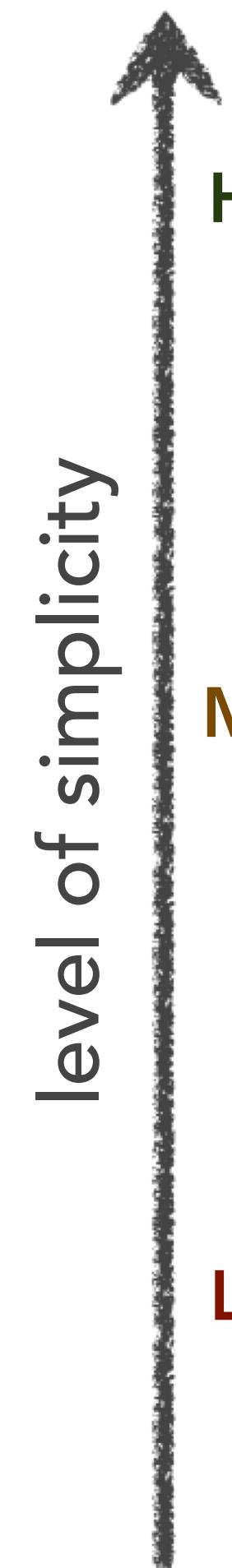
service-based
architecture



service-oriented
architecture



microservices
architecture



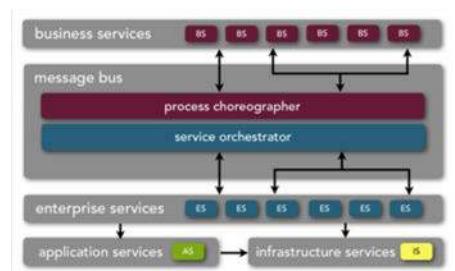
characteristics differences

overall simplicity

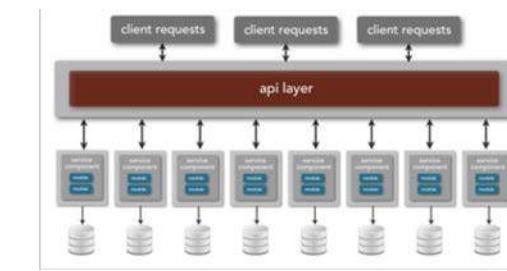
level of complexity in
applications implemented
using the architecture pattern



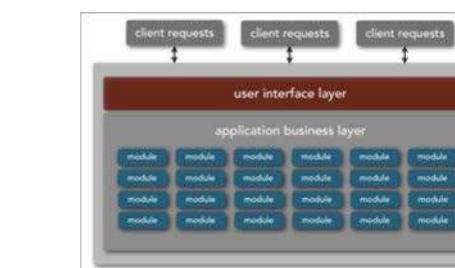
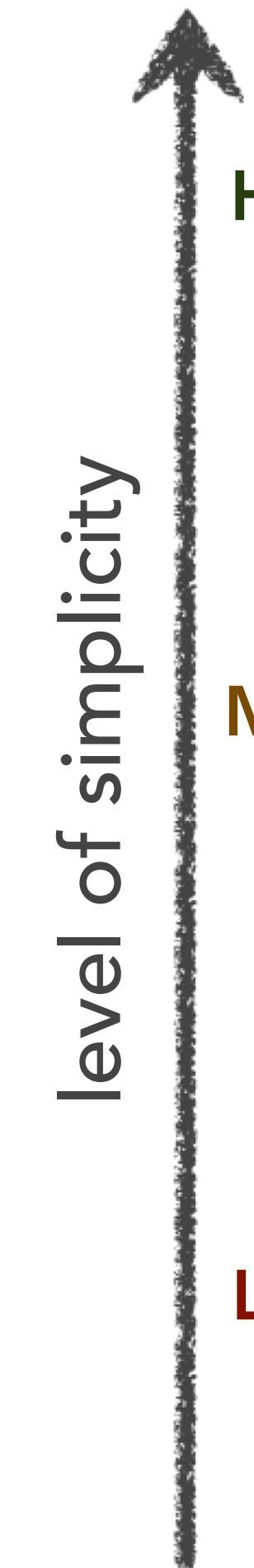
service-based
architecture



service-oriented
architecture



microservices
architecture

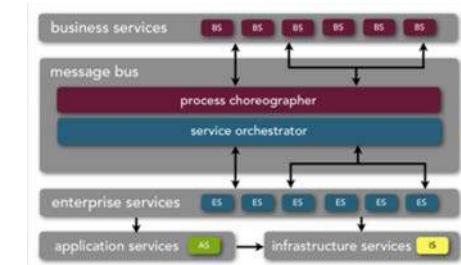


monolithic
architecture

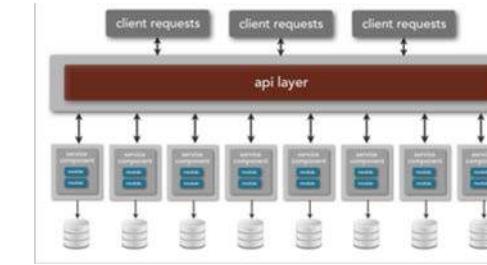
characteristics differences

overall simplicity

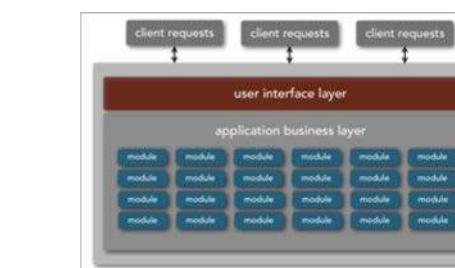
level of complexity in
applications implemented
using the architecture pattern



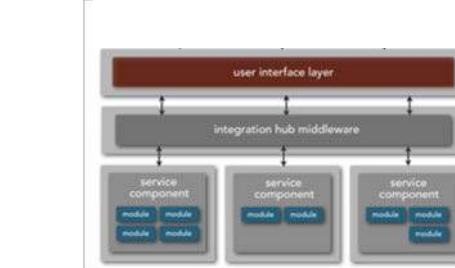
service-oriented
architecture



microservices
architecture



monolithic
architecture

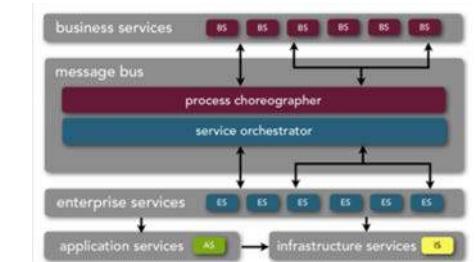


service-based
architecture

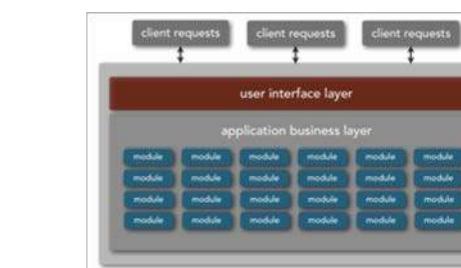
characteristics differences

overall simplicity

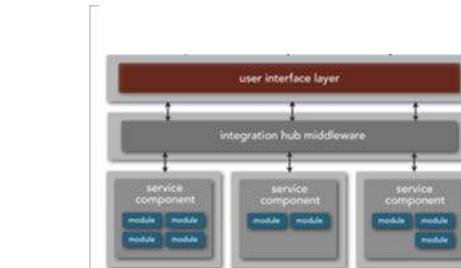
level of complexity in
applications implemented
using the architecture pattern



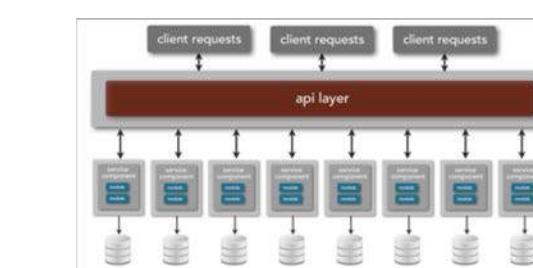
service-oriented
architecture



monolithic
architecture



service-based
architecture

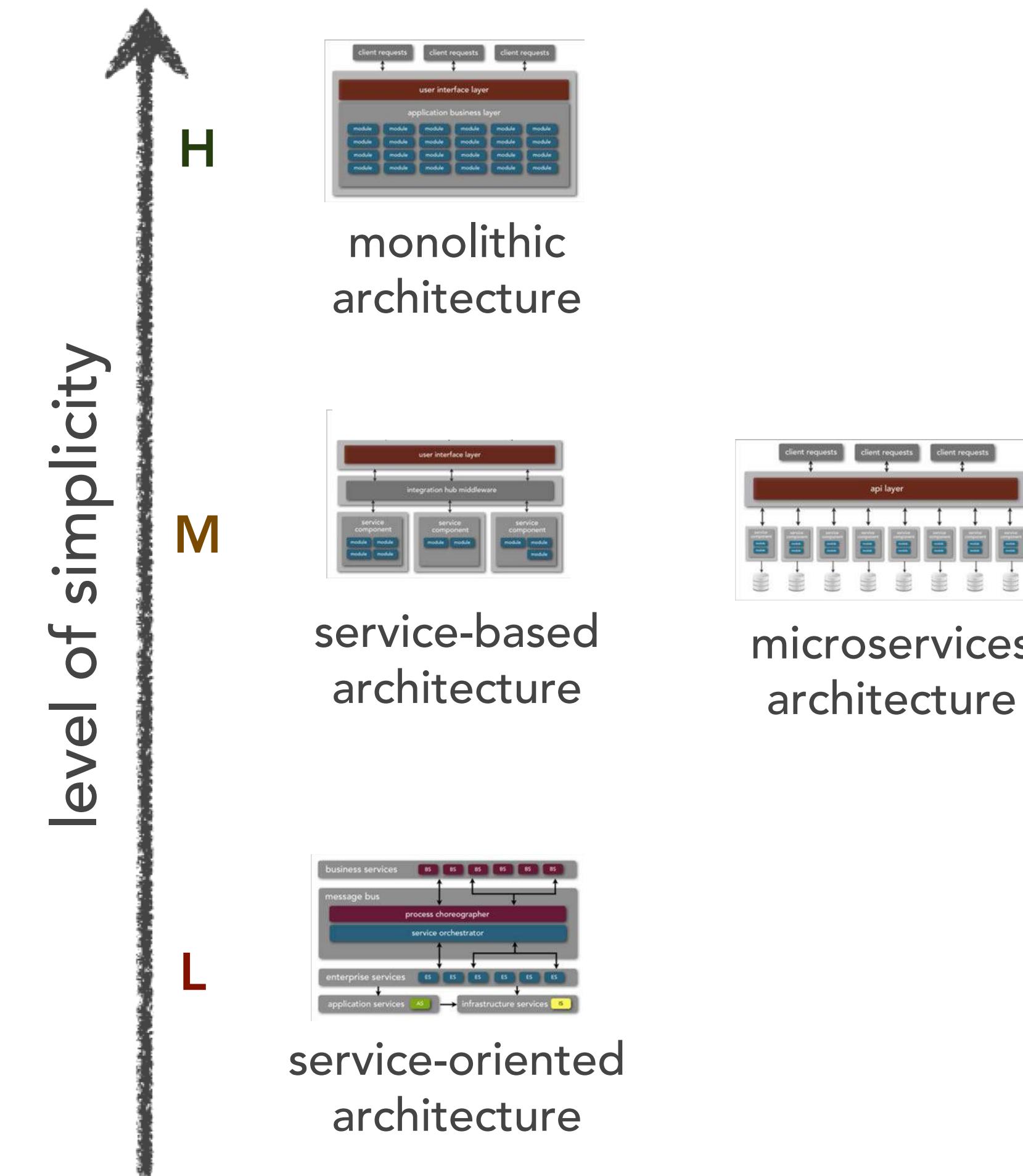


microservices
architecture

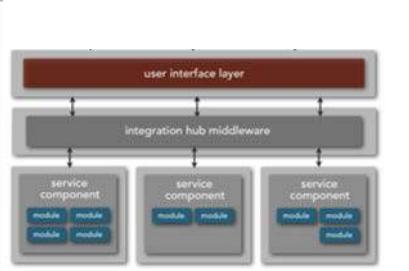
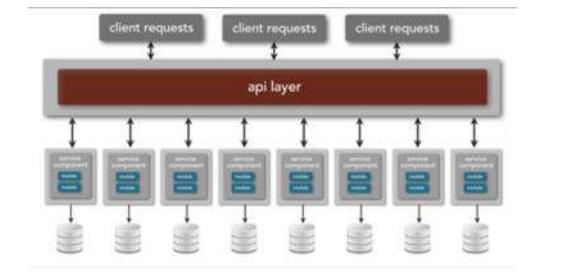
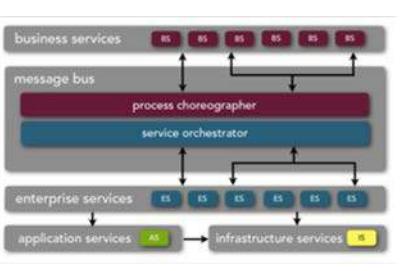
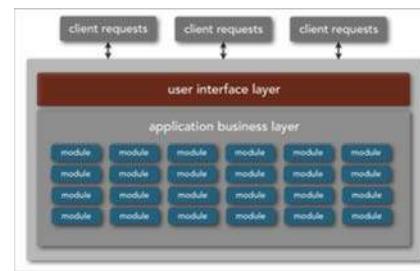
characteristics differences

overall simplicity

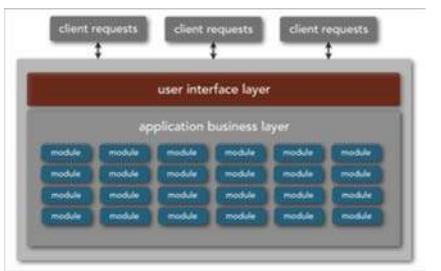
level of complexity in
applications implemented
using the architecture pattern



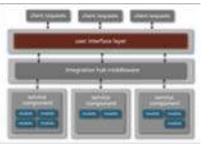
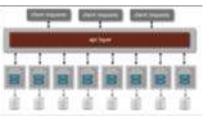
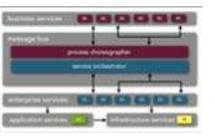
Which ?!?



Which ?!?



Monolith

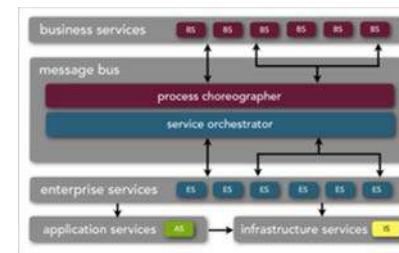


default

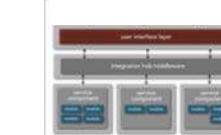
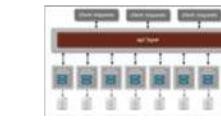
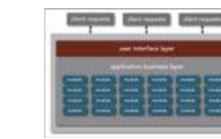
easy to understand/build

doesn't scale well in any dimension

Which ?!?



Service-oriented



service taxonomy

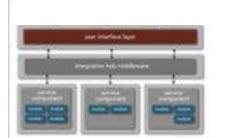
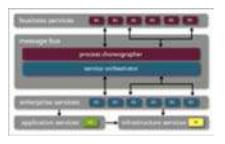
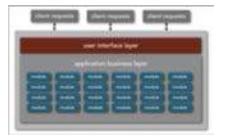
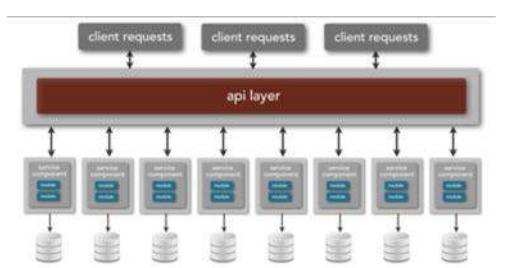
high degree of (potential) reuse

highly compatible in integration-heavy environments

operationally complex

Which ?!?

Microservices

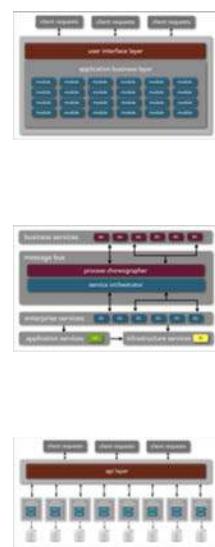


incremental change

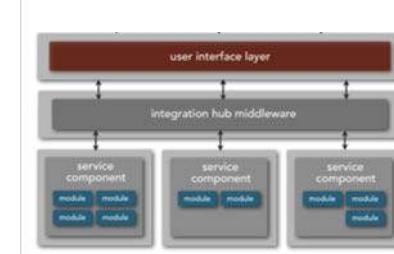
highly evolvable

complex interactions

Which ?!?



service-based

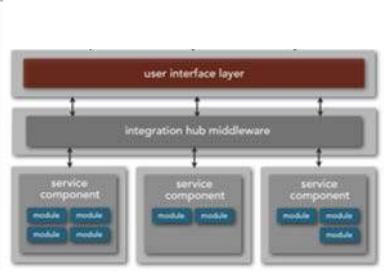
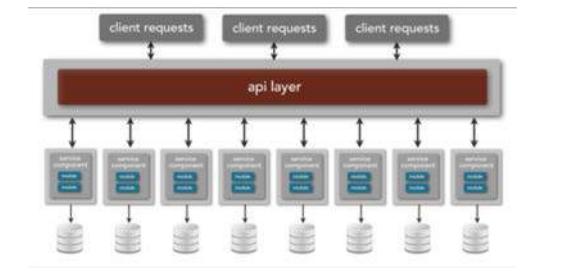
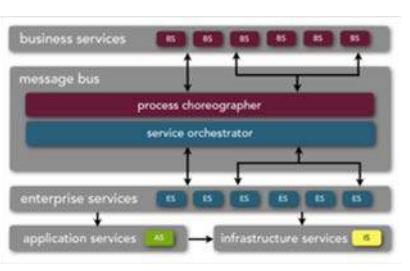
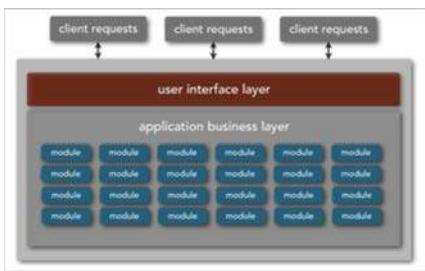


best target for migration

good compromise

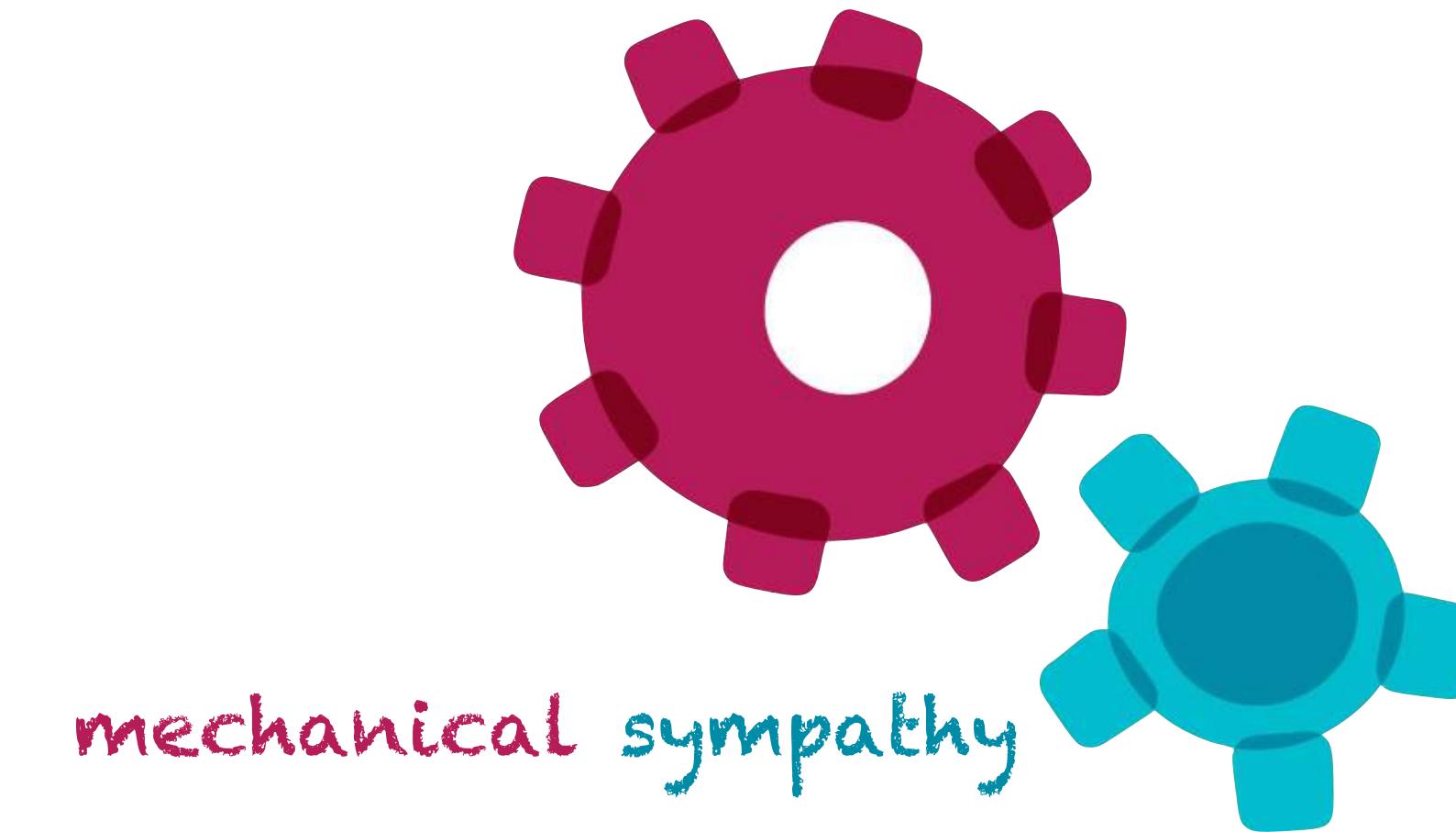
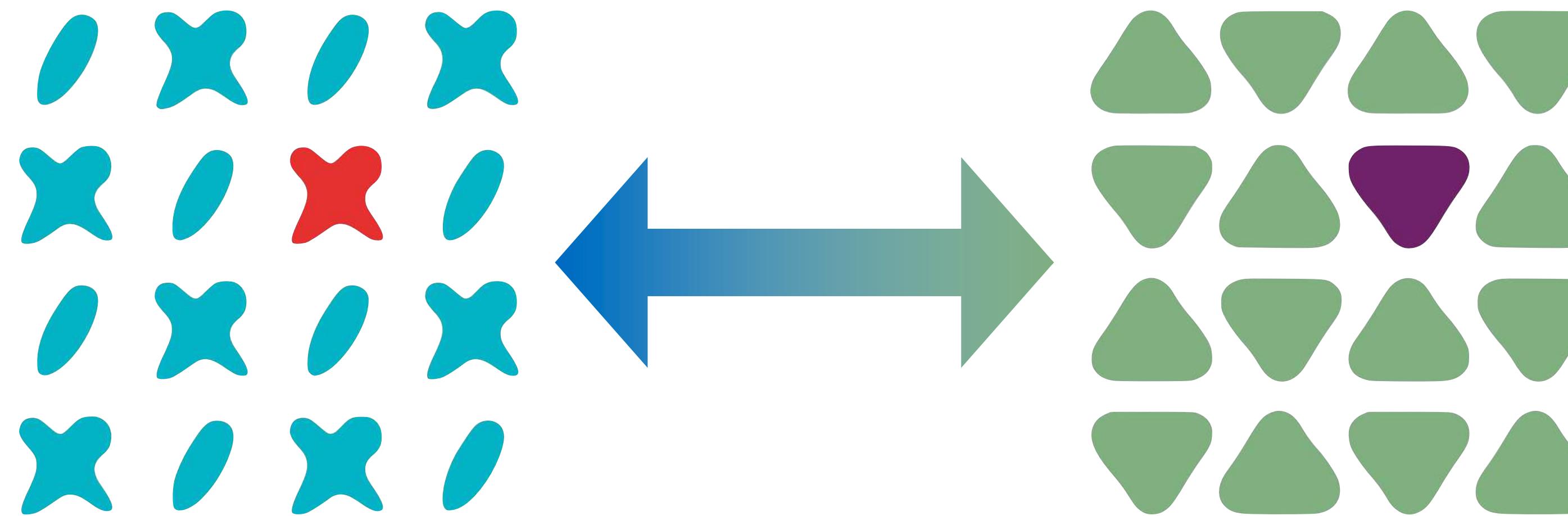
domain-centric without going μ service

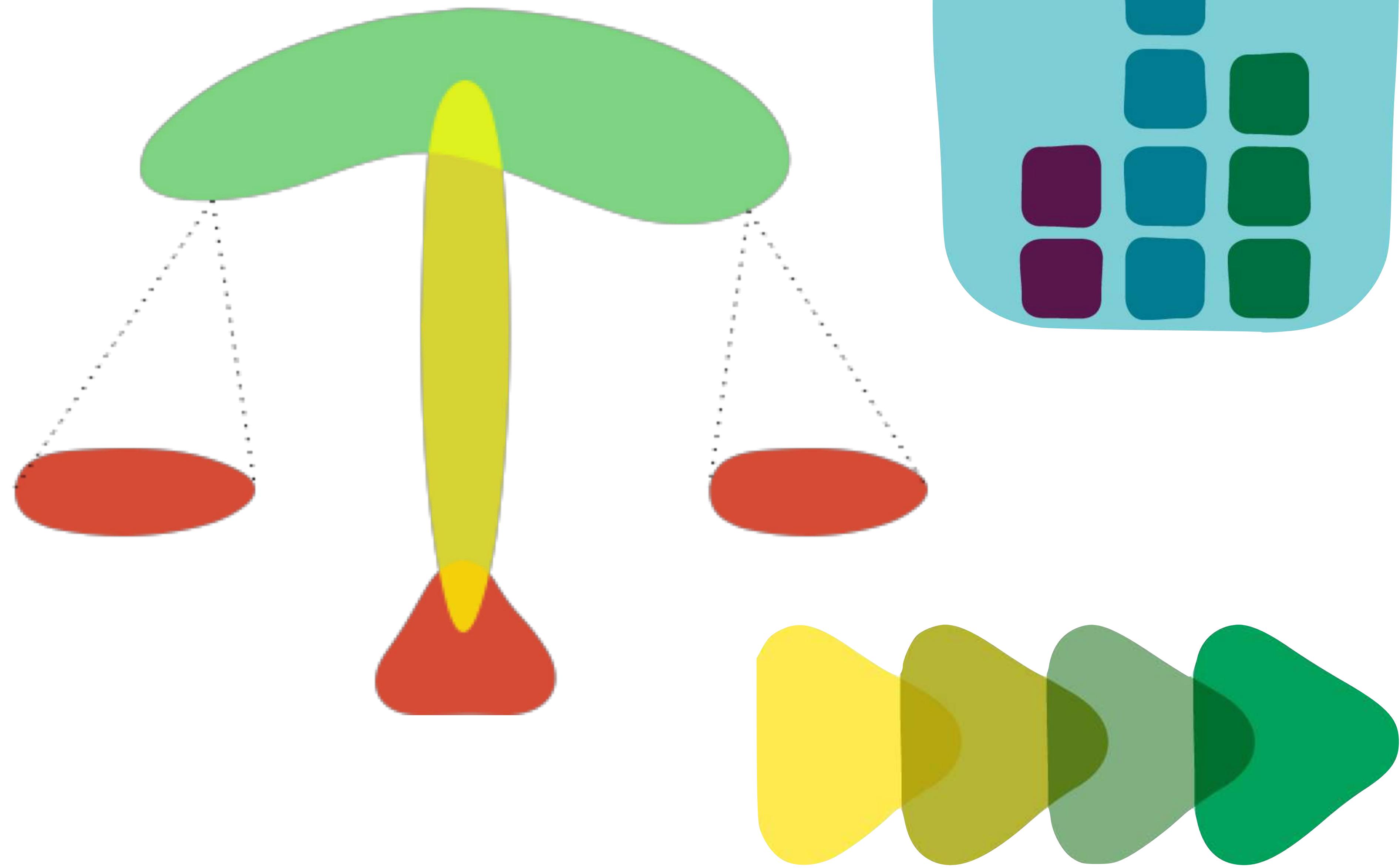
Which ?!?



it depends

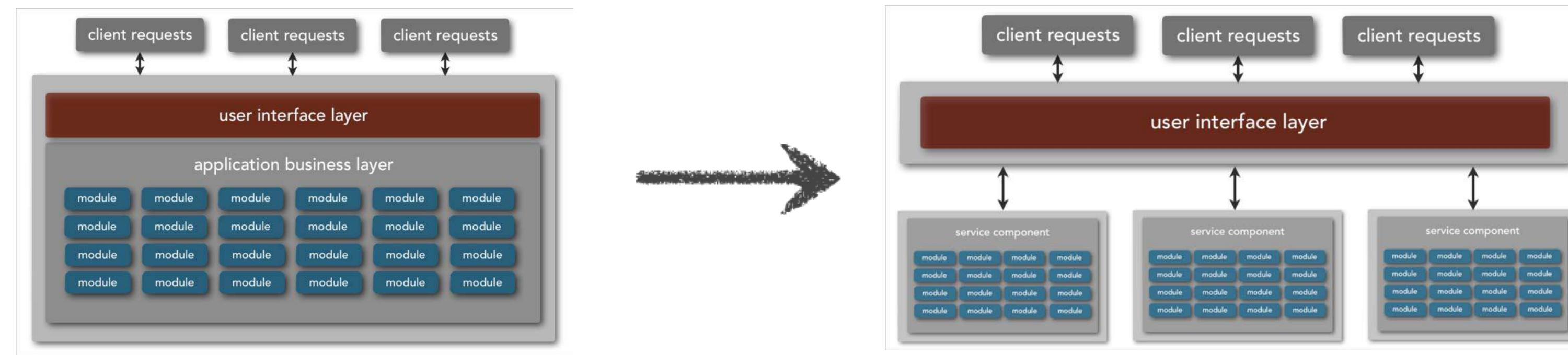
domain/architecture isomorphism





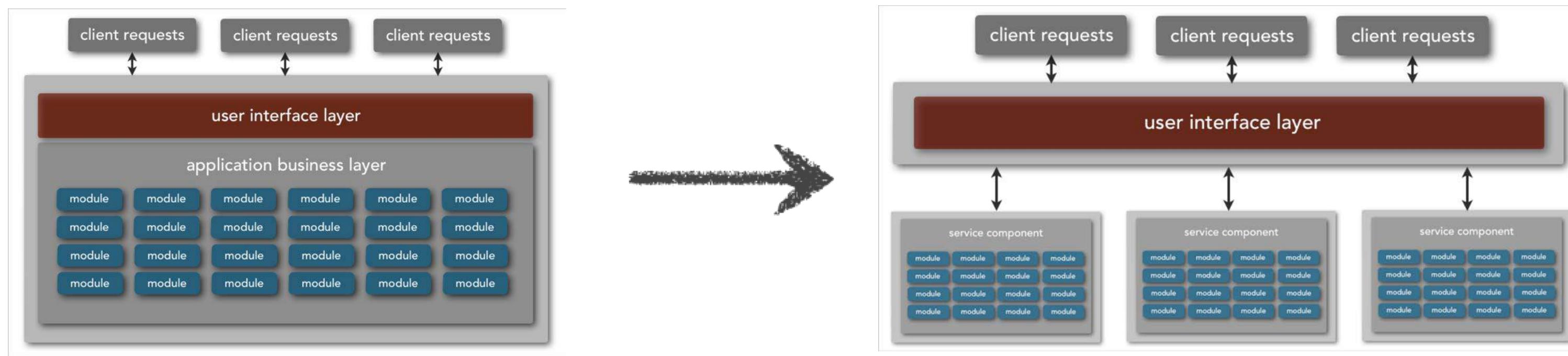
migration paths

monolithic application to service-based architecture



migration paths

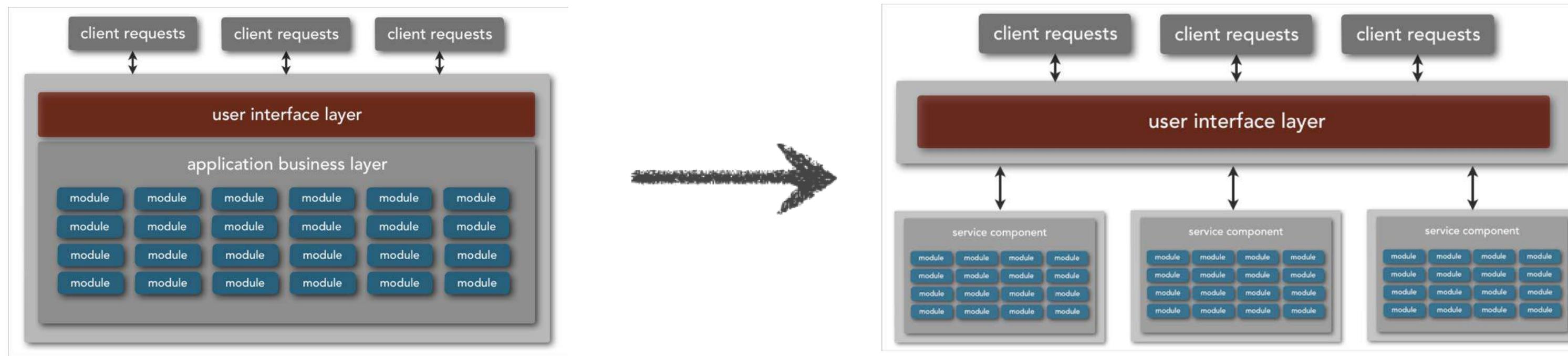
monolithic application to service-based architecture



— deployment

migration paths

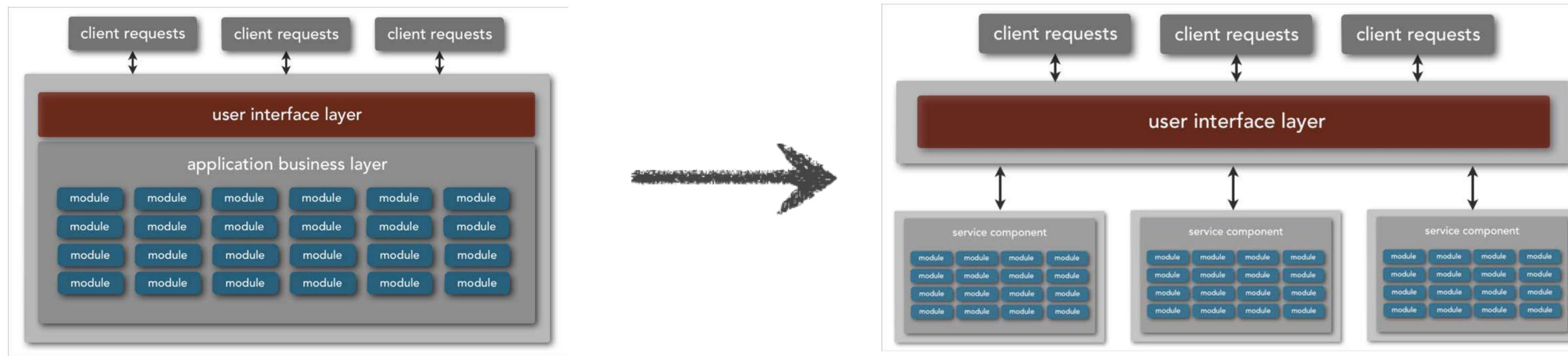
monolithic application to service-based architecture



___deployment
___reliability and robustness

migration paths

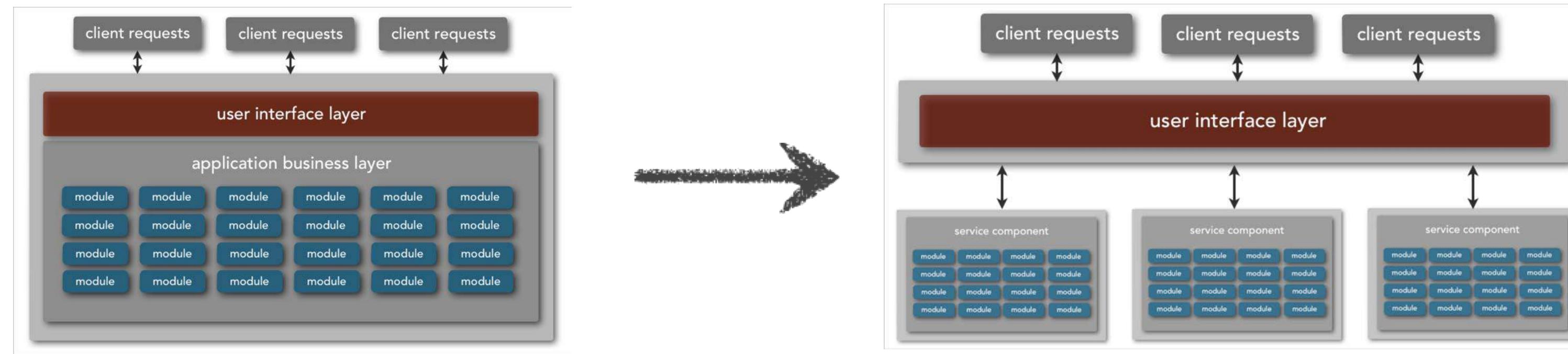
monolithic application to service-based architecture



___deployment
___reliability and robustness
___testability

migration paths

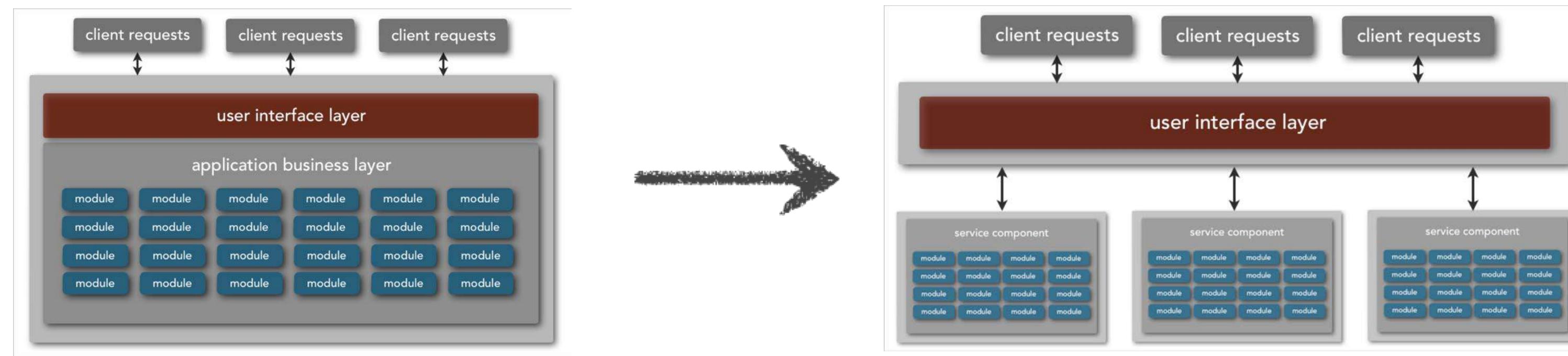
monolithic application to service-based architecture



- ___ deployment
- ___ reliability and robustness
- ___ testability
- ___ scalability

migration paths

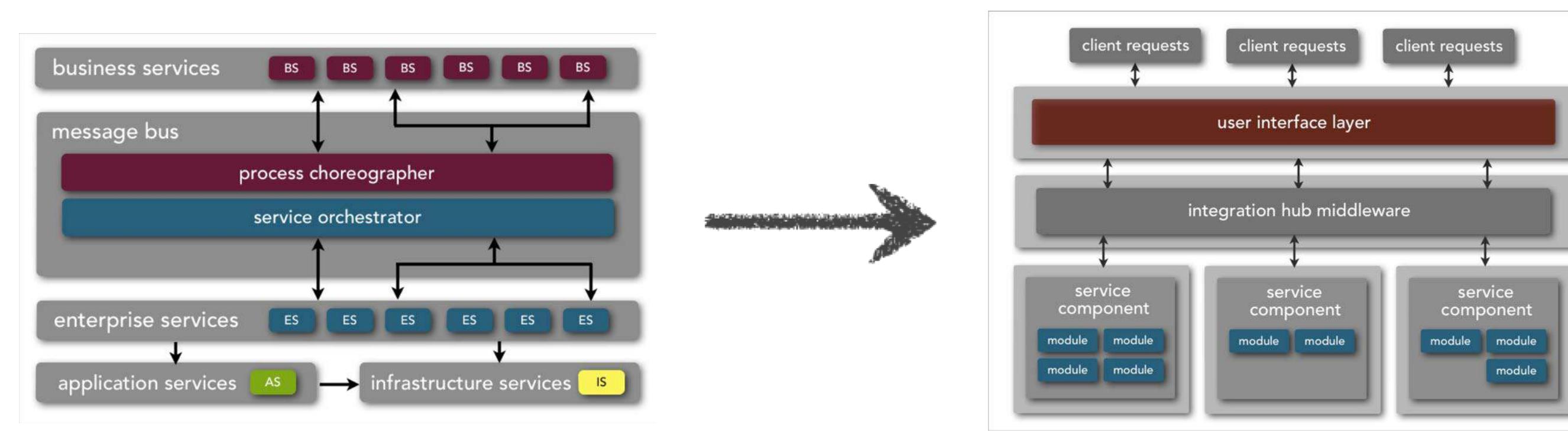
monolithic application to service-based architecture



- ___ deployment
- ___ reliability and robustness
- ___ testability
- ___ scalability
- ___ application size

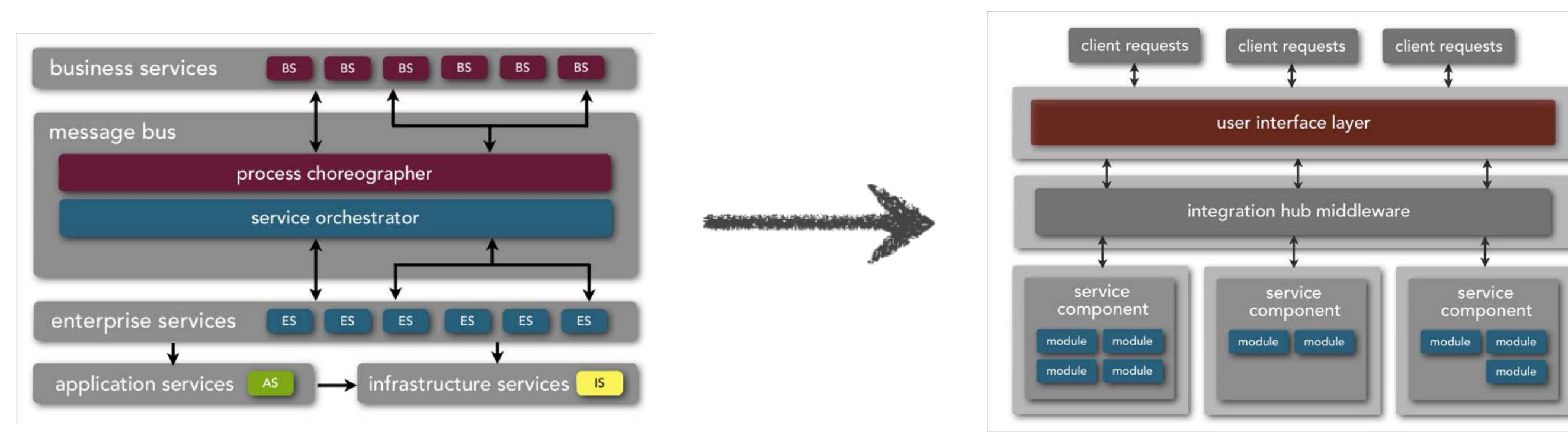
migration paths

soa to service-based architecture



migration paths

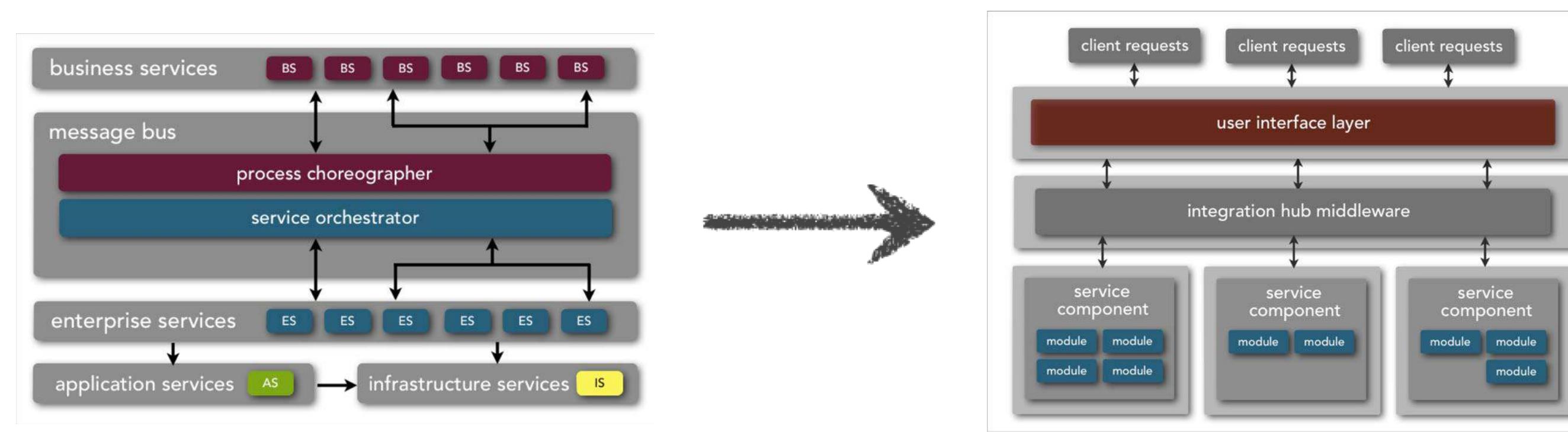
soa to service-based architecture



complexity

migration paths

soa to service-based architecture

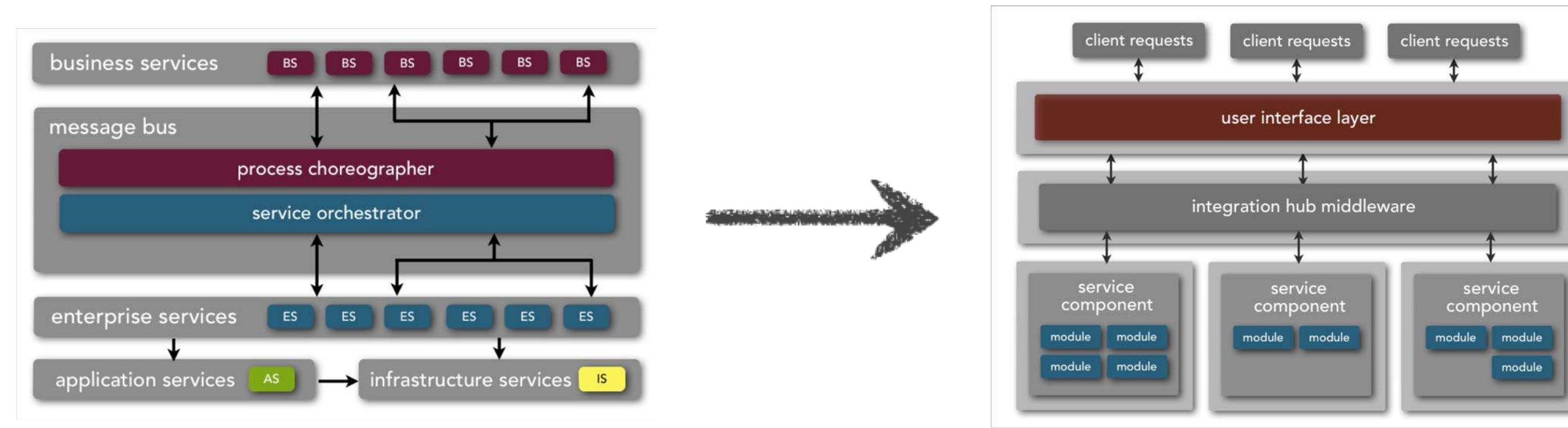


— complexity

— business involvement

migration paths

soa to service-based architecture



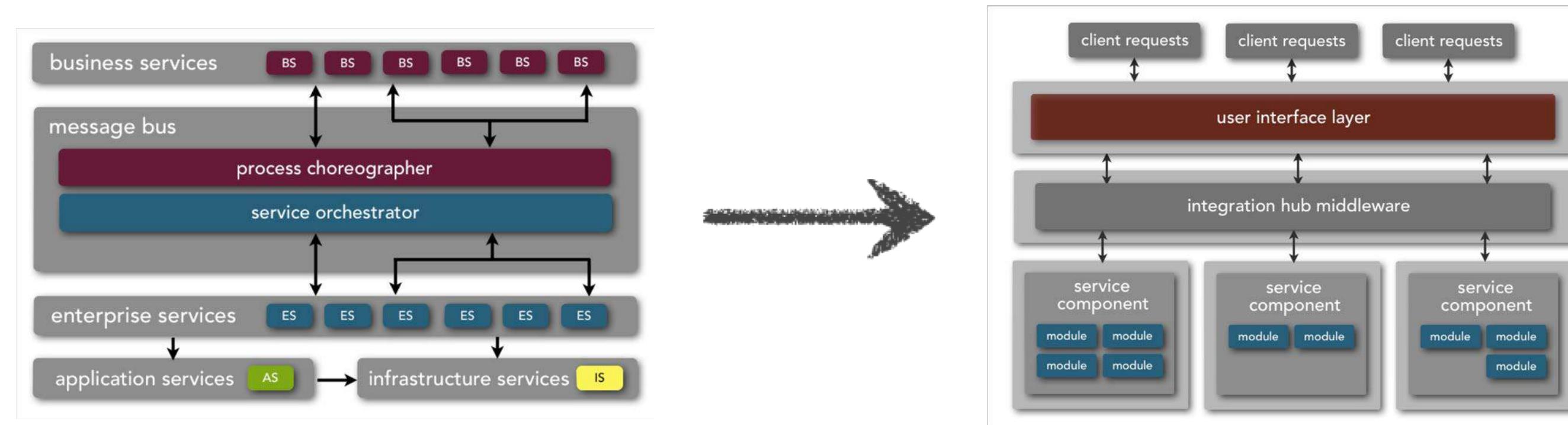
complexity

business involvement

cost

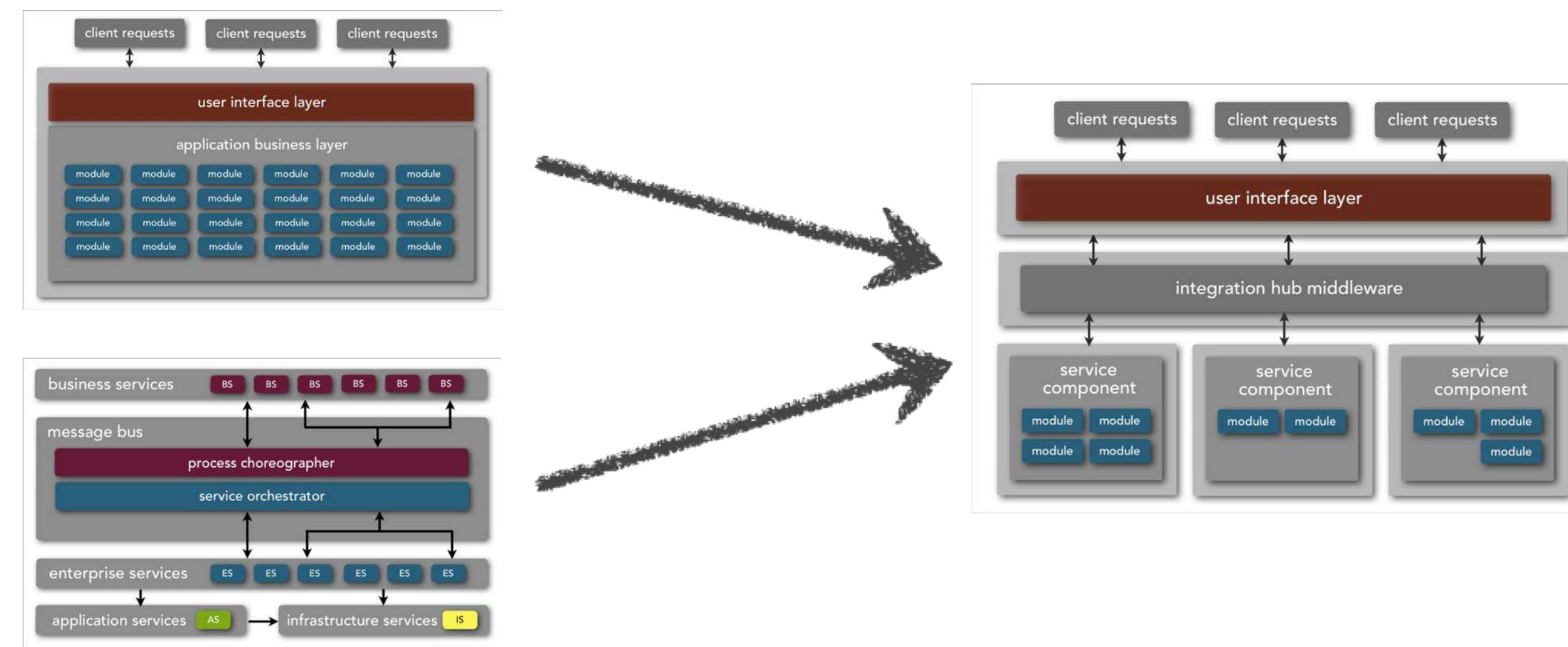
migration paths

soa to service-based architecture

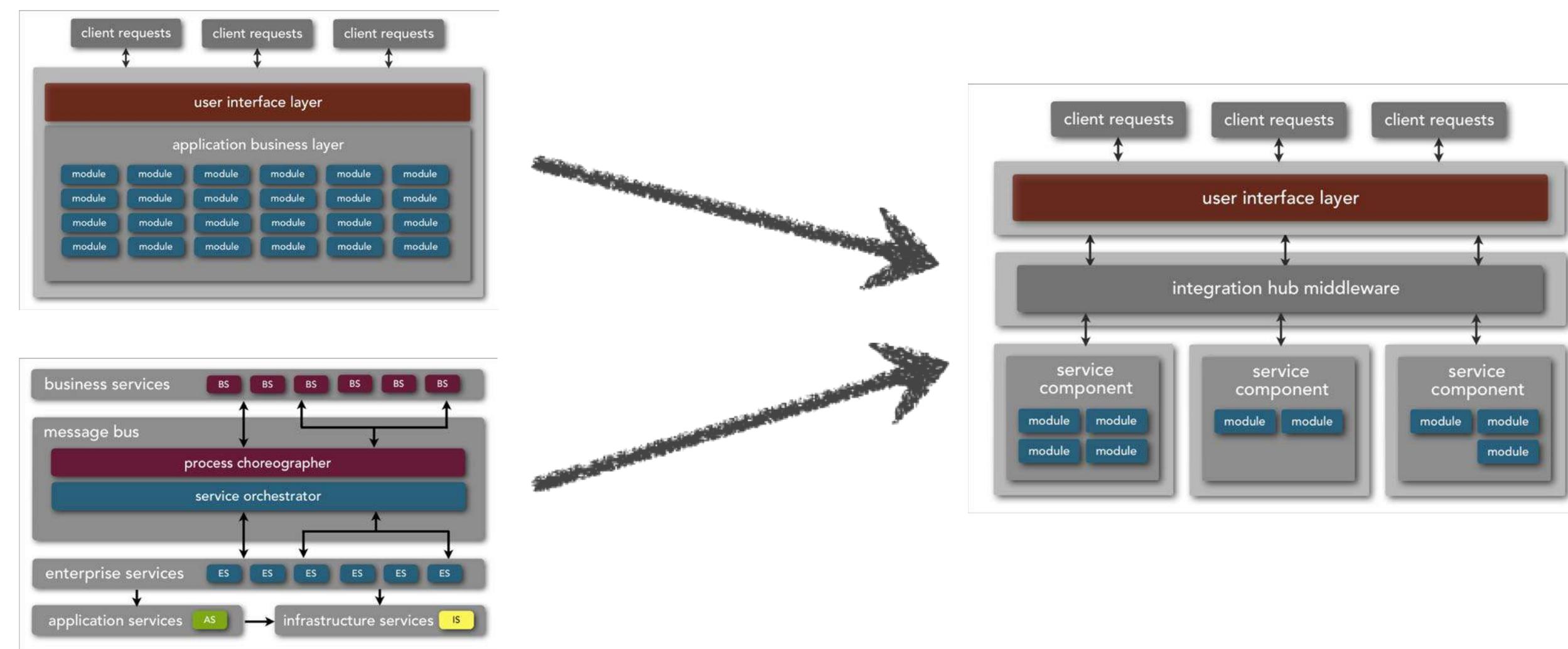


- complexity
- business involvement
- cost
- overkill

migration paths business drivers

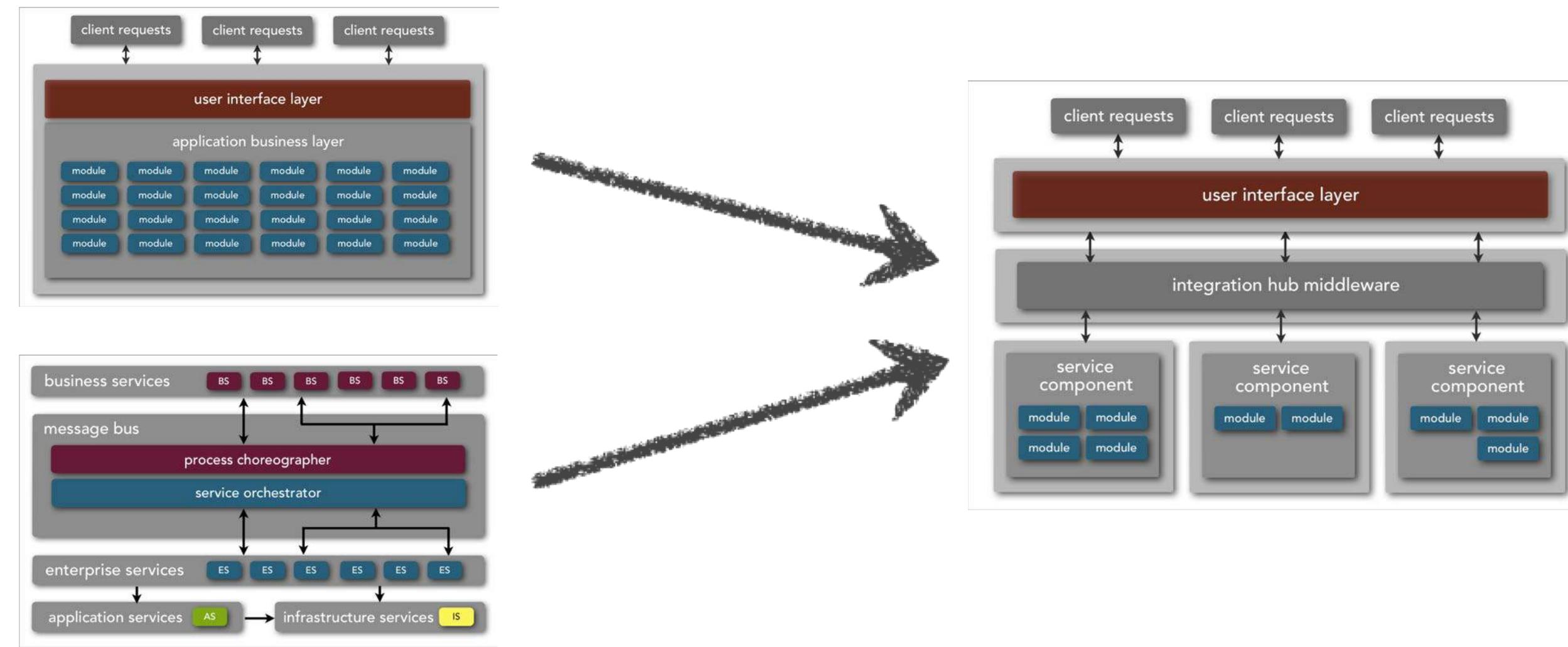


migration paths business drivers



—faster time to market

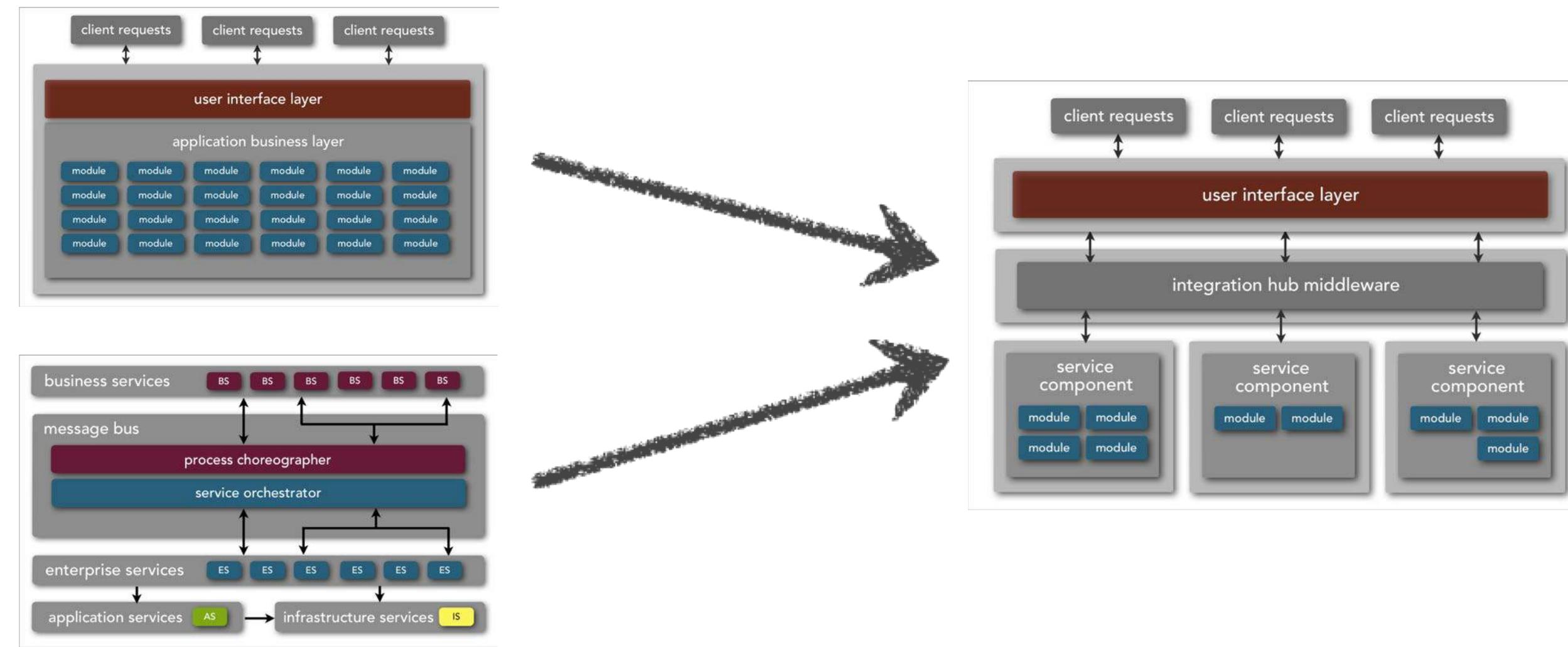
migration paths business drivers



— faster time to market
— improved reliability and quality

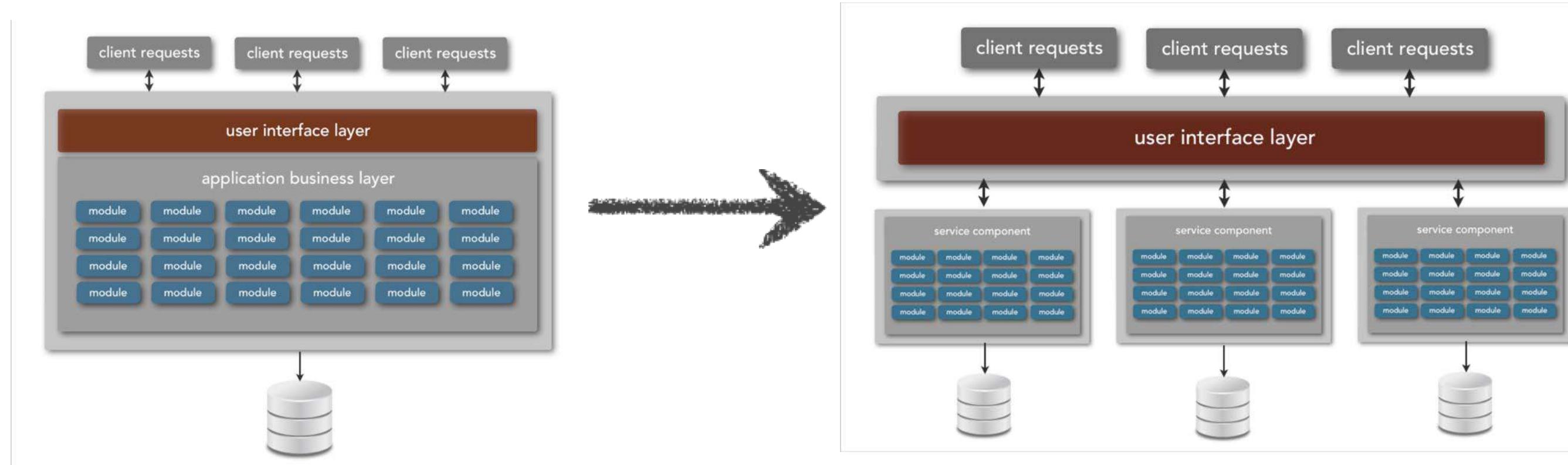
migration paths

business drivers



- __ faster time to market
- __ improved reliability and quality
- __ reduced costs

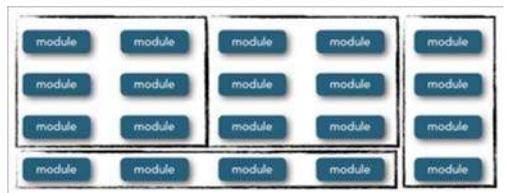
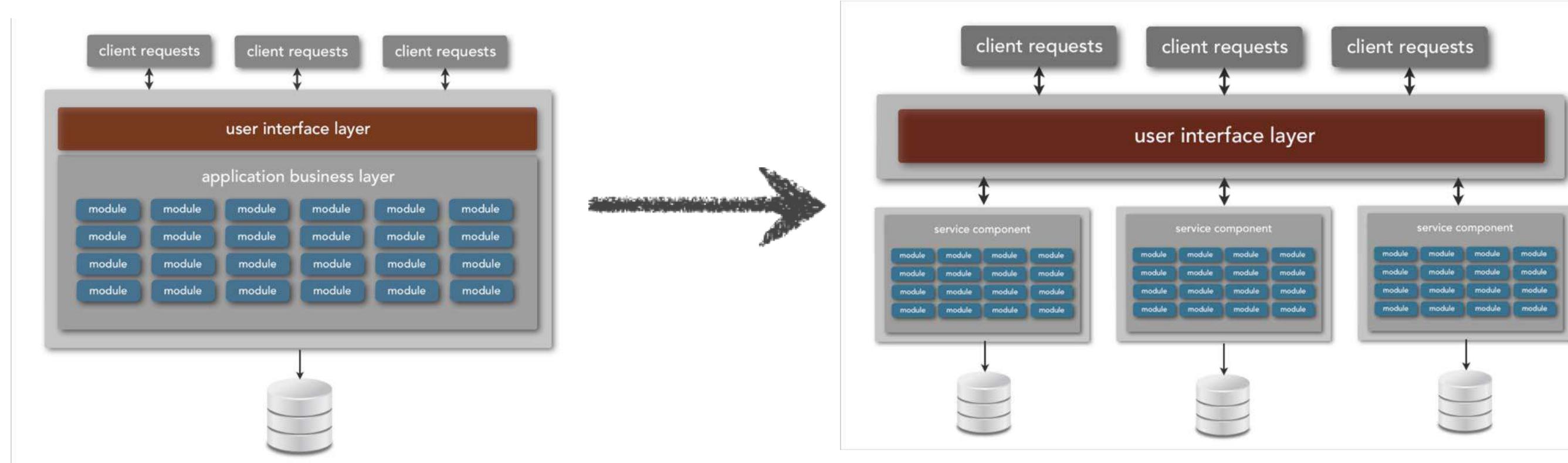
migration challenges



share everything
architecture

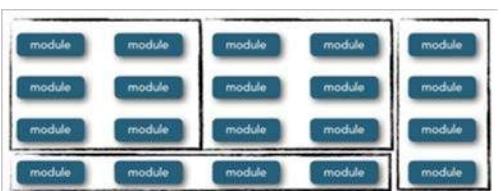
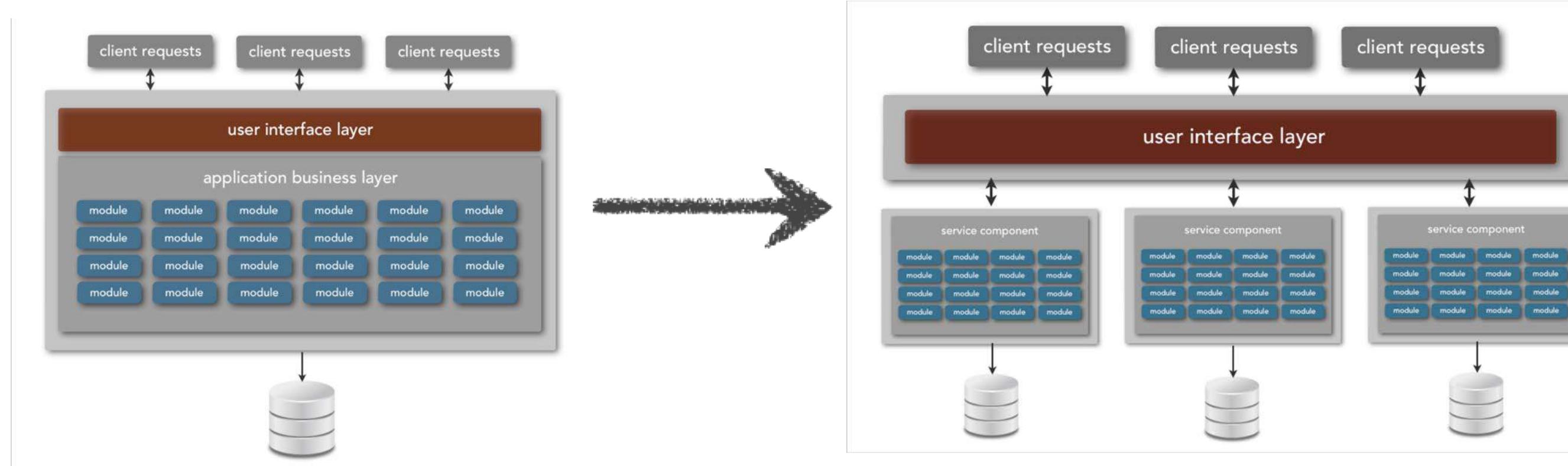
share as little as
possible architecture

migration challenges

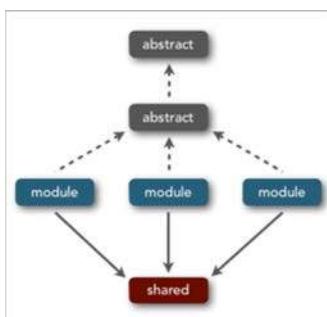


service component granularity and transactional boundaries

migration challenges

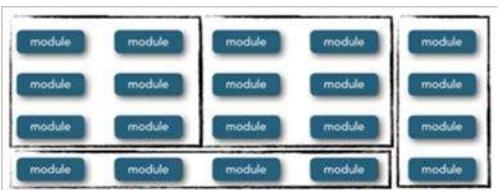
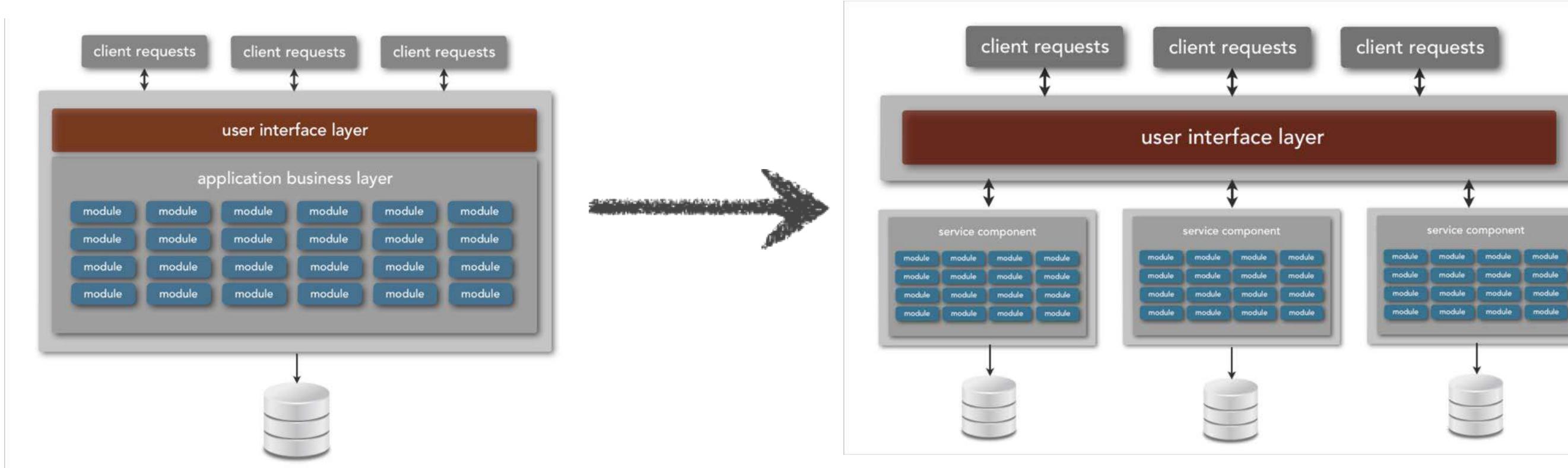


service component granularity and transactional boundaries

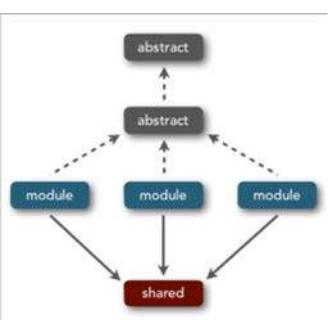


shared services, modules, and object hierarchies

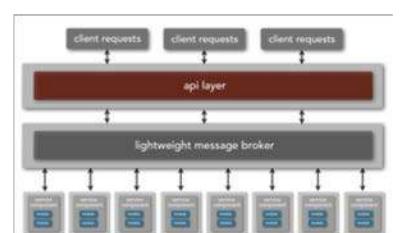
migration challenges



service component granularity and transactional boundaries

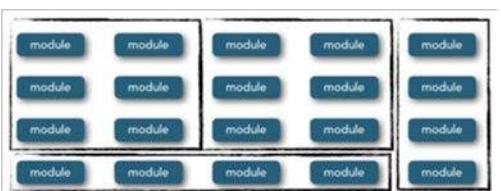
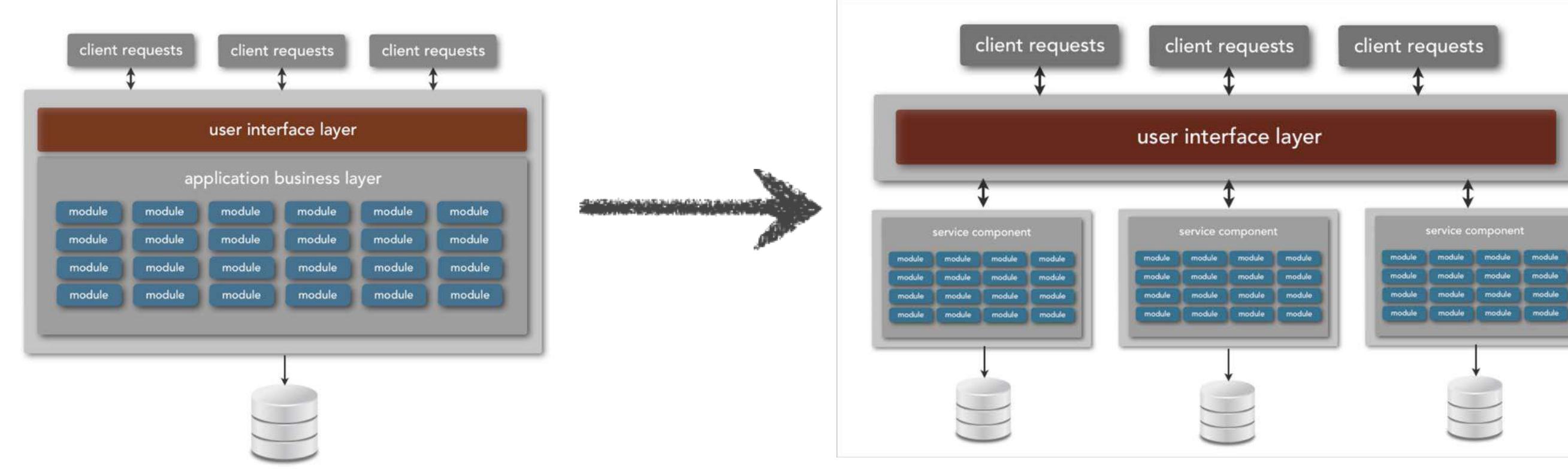


shared services, modules, and object hierarchies

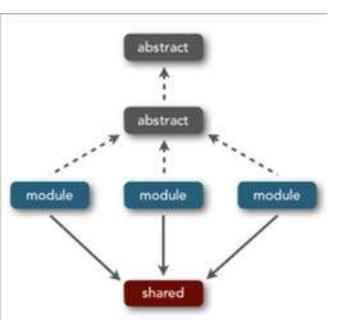


distributed architecture and remote service issues

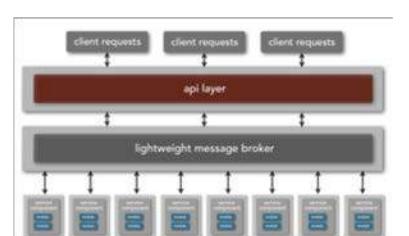
migration challenges



service component granularity and transactional boundaries



shared services, modules, and object hierarchies



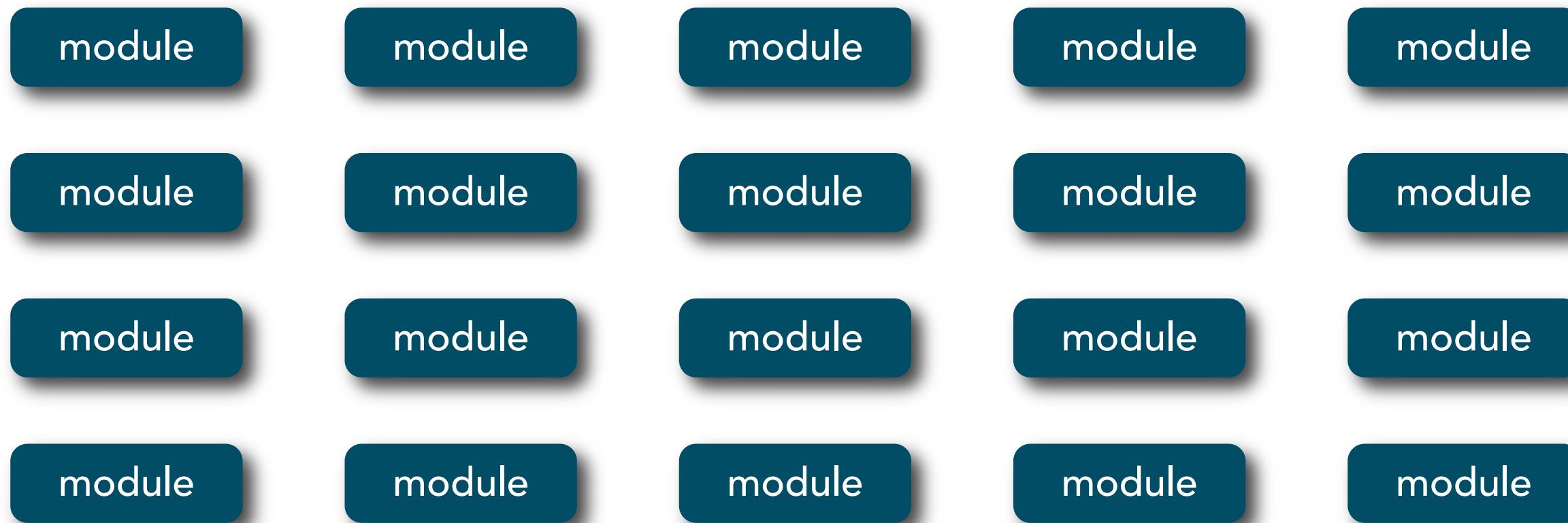
distributed architecture and remote service issues



large shared relational database

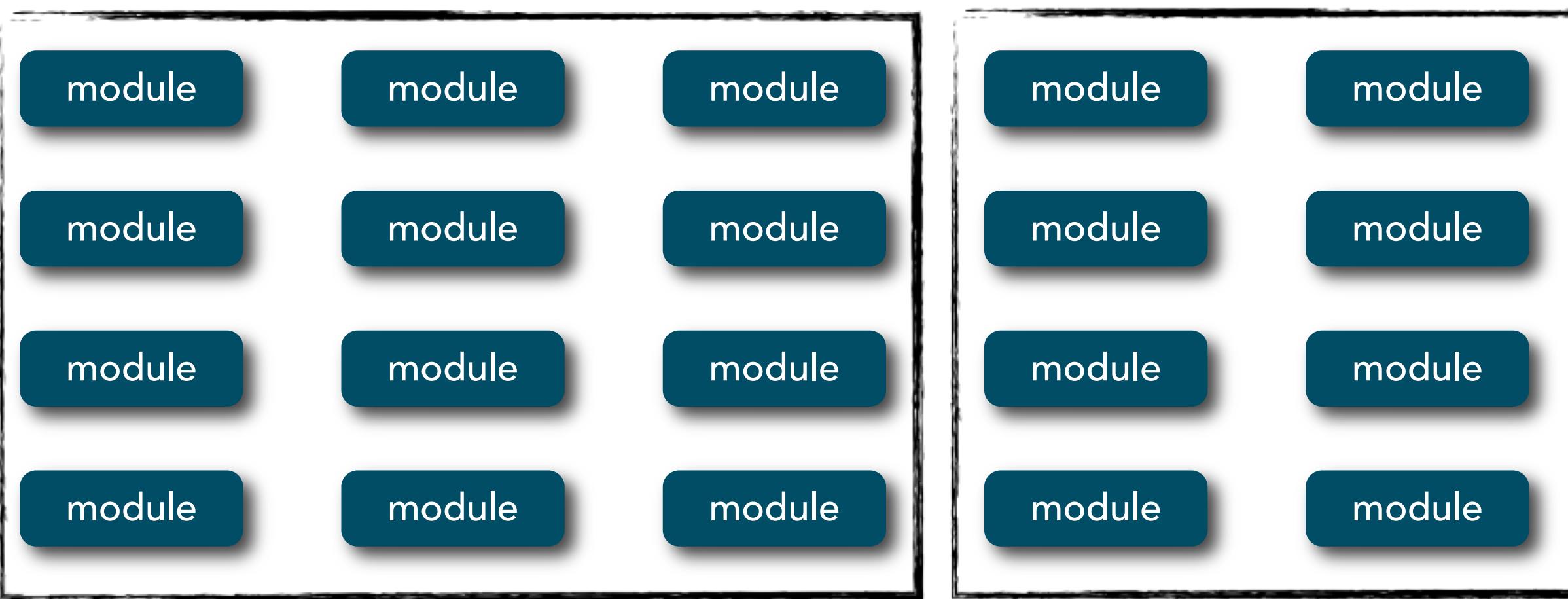
migration challenges

service component granularity



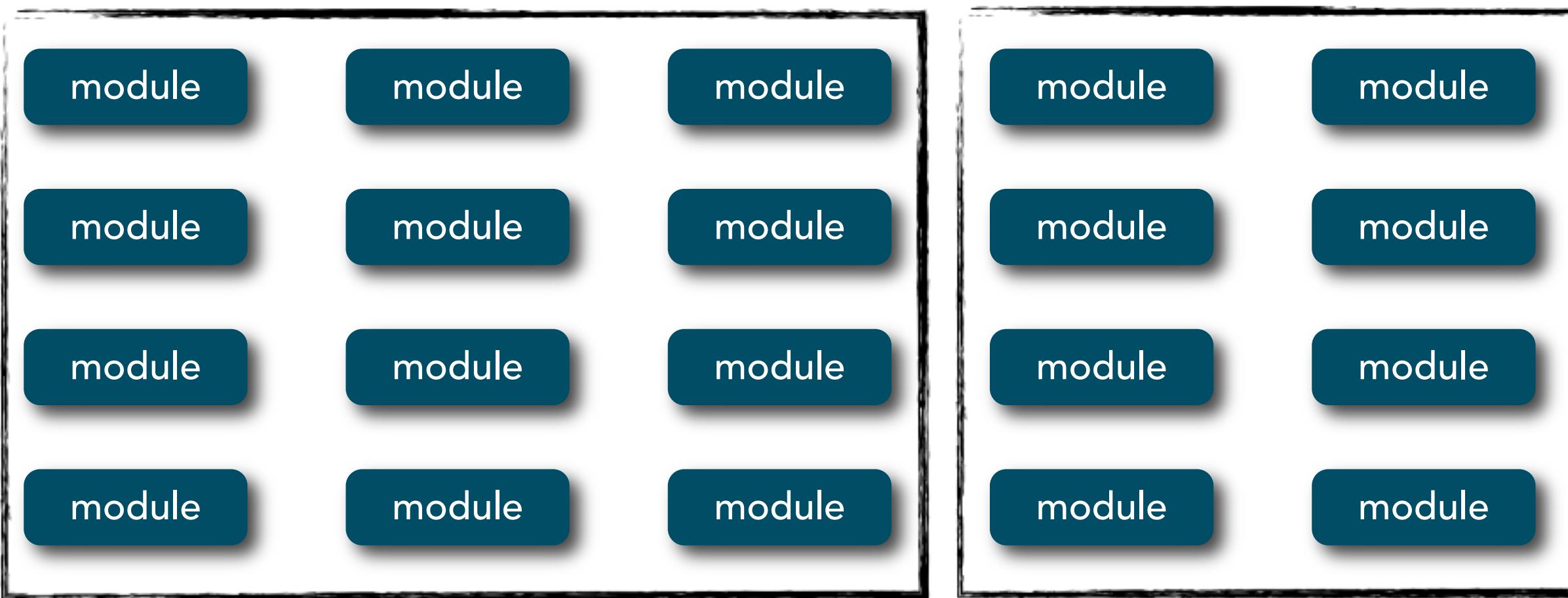
migration challenges

service component granularity



migration challenges

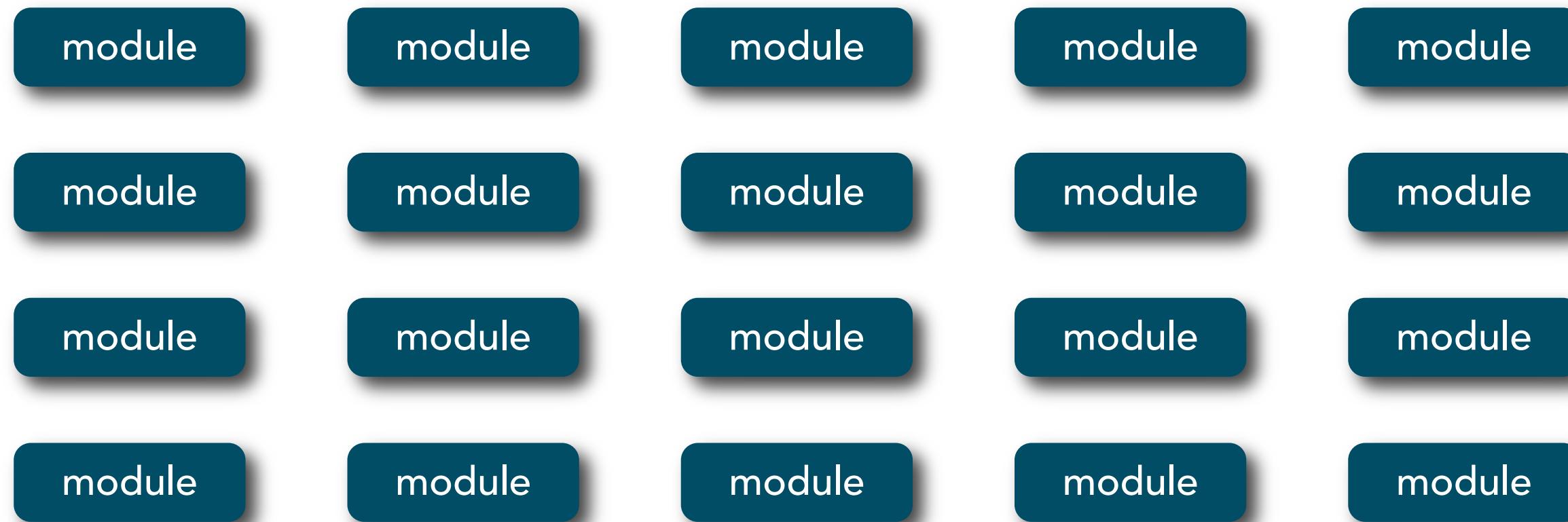
service component granularity



coarse-grained service components address transactional issues but may not achieve your desired goals

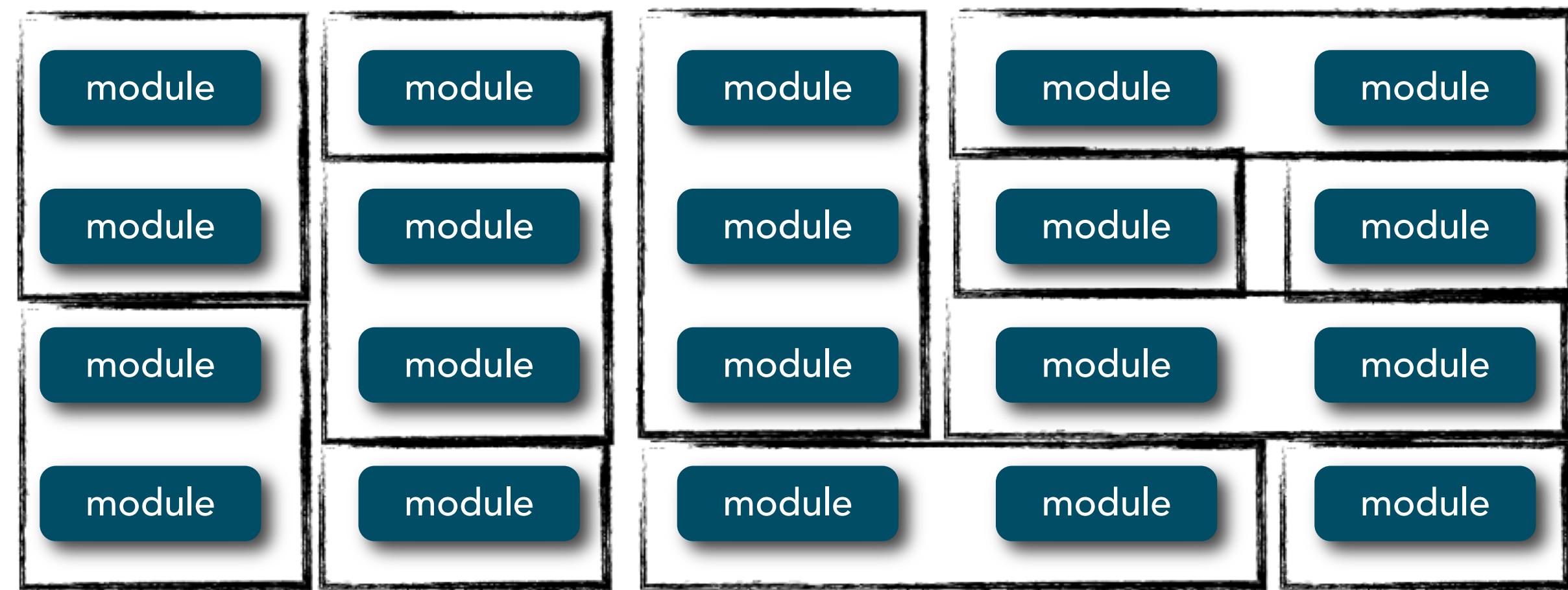
migration challenges

service component granularity



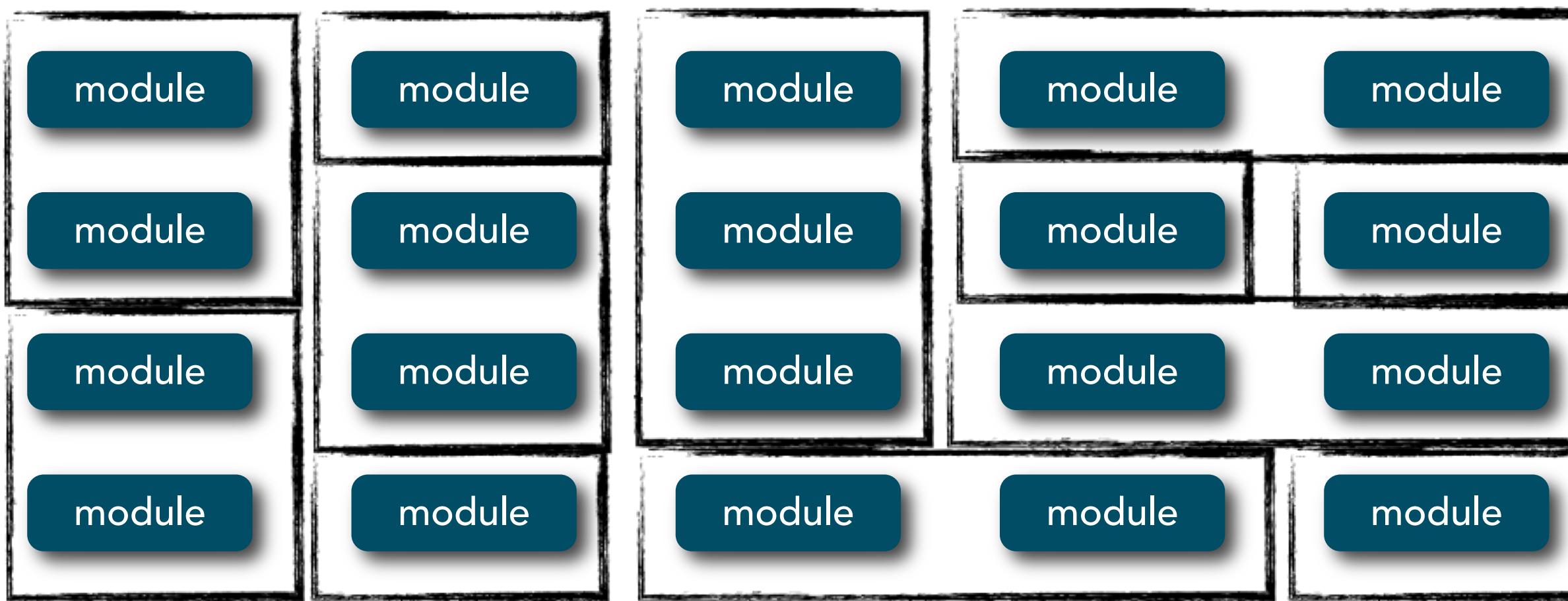
migration challenges

service component granularity



migration challenges

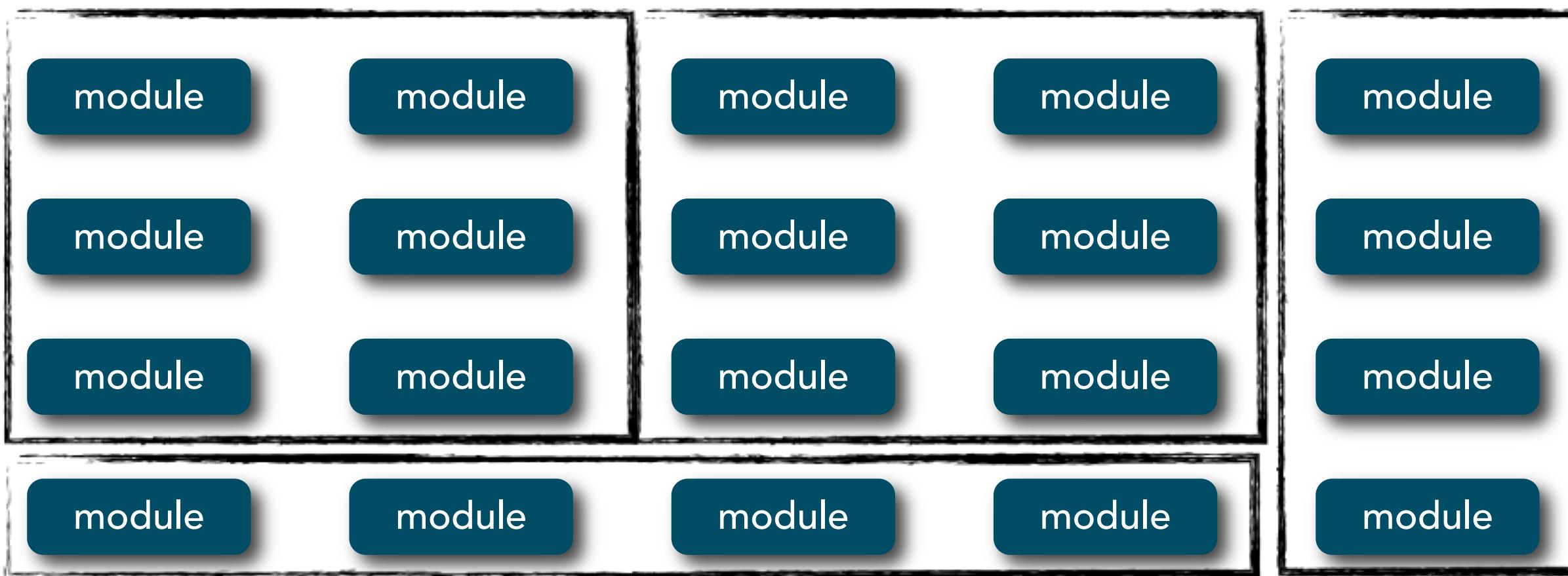
service component granularity



fine-grained service components may lead to too much orchestration and inter-dependency between service components

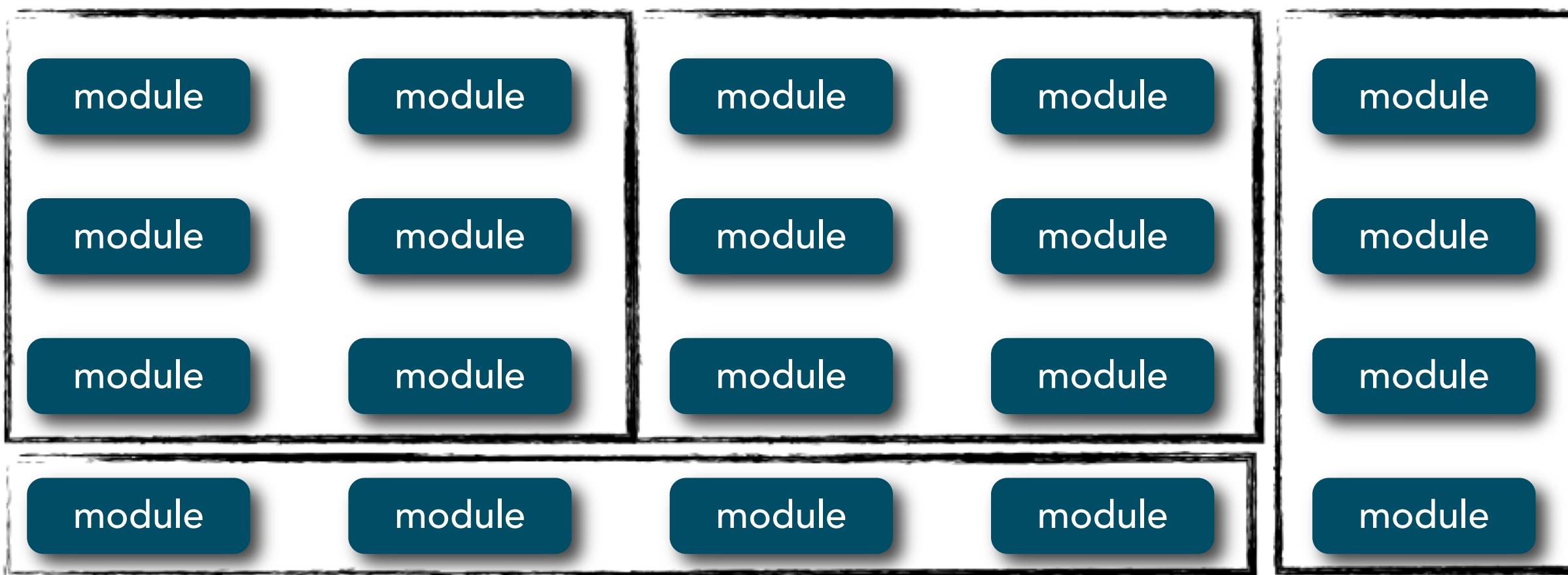
migration challenges

service component granularity



migration challenges

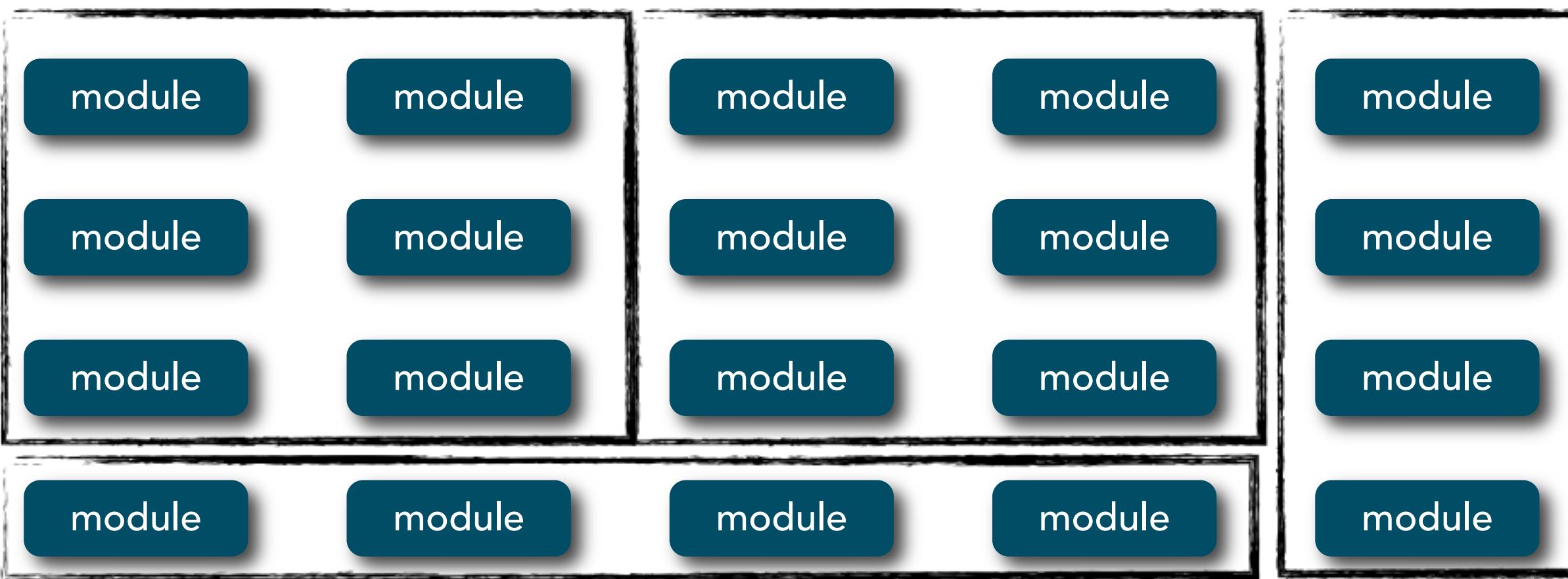
service component granularity



business functionality groupings

migration challenges

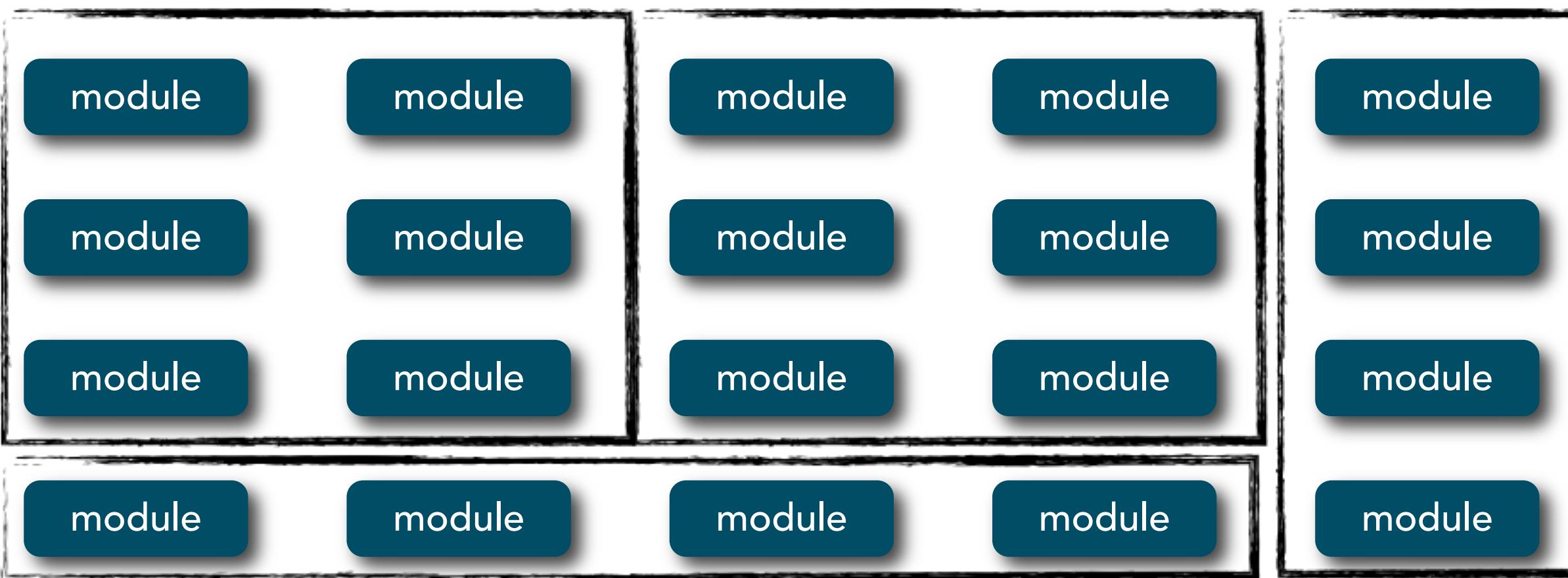
service component granularity



business functionality groupings
transactional boundaries

migration challenges

service component granularity



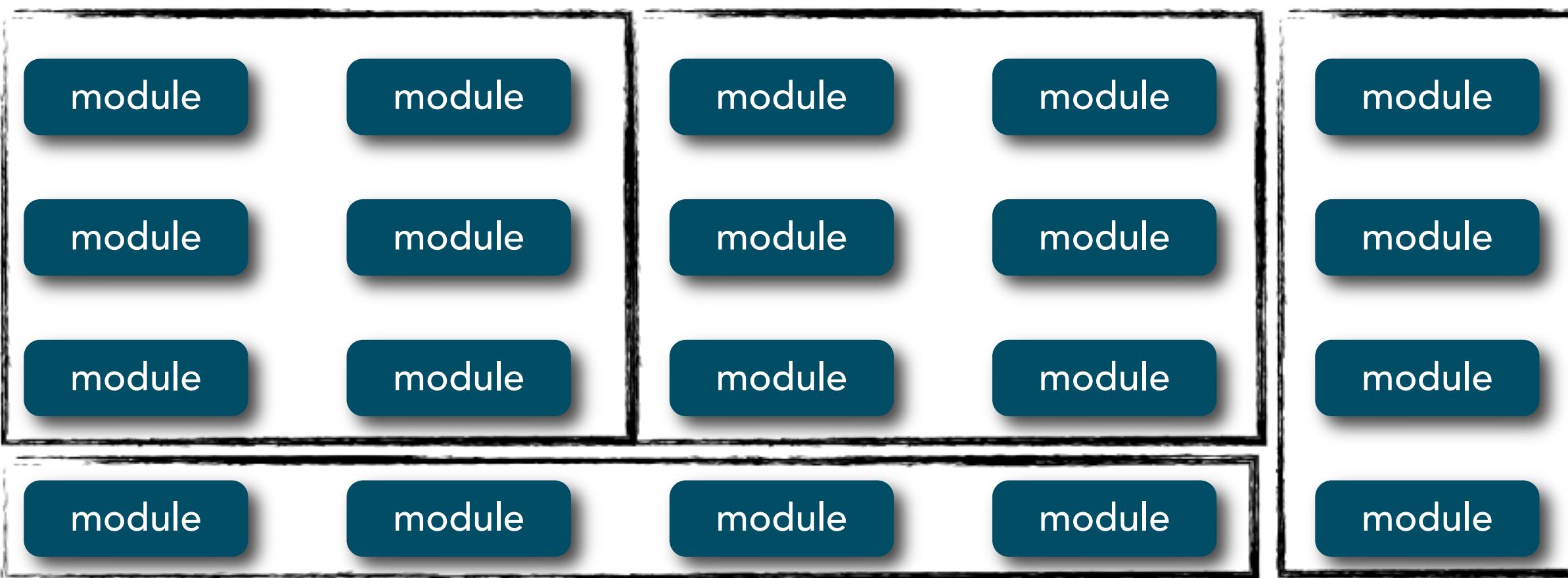
business functionality groupings

transactional boundaries

deployment goals

migration challenges

service component granularity



business functionality groupings

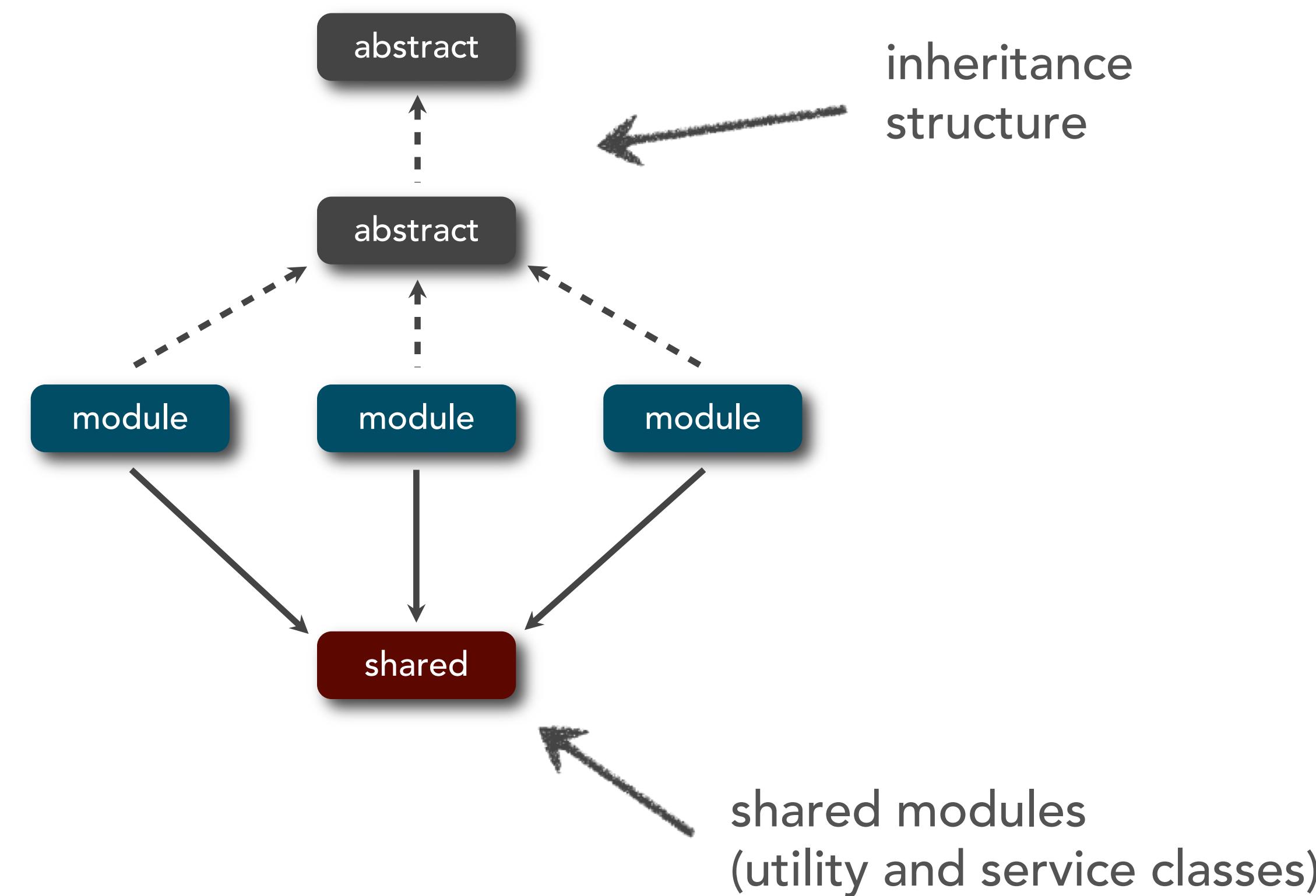
transactional boundaries

deployment goals

scalability needs

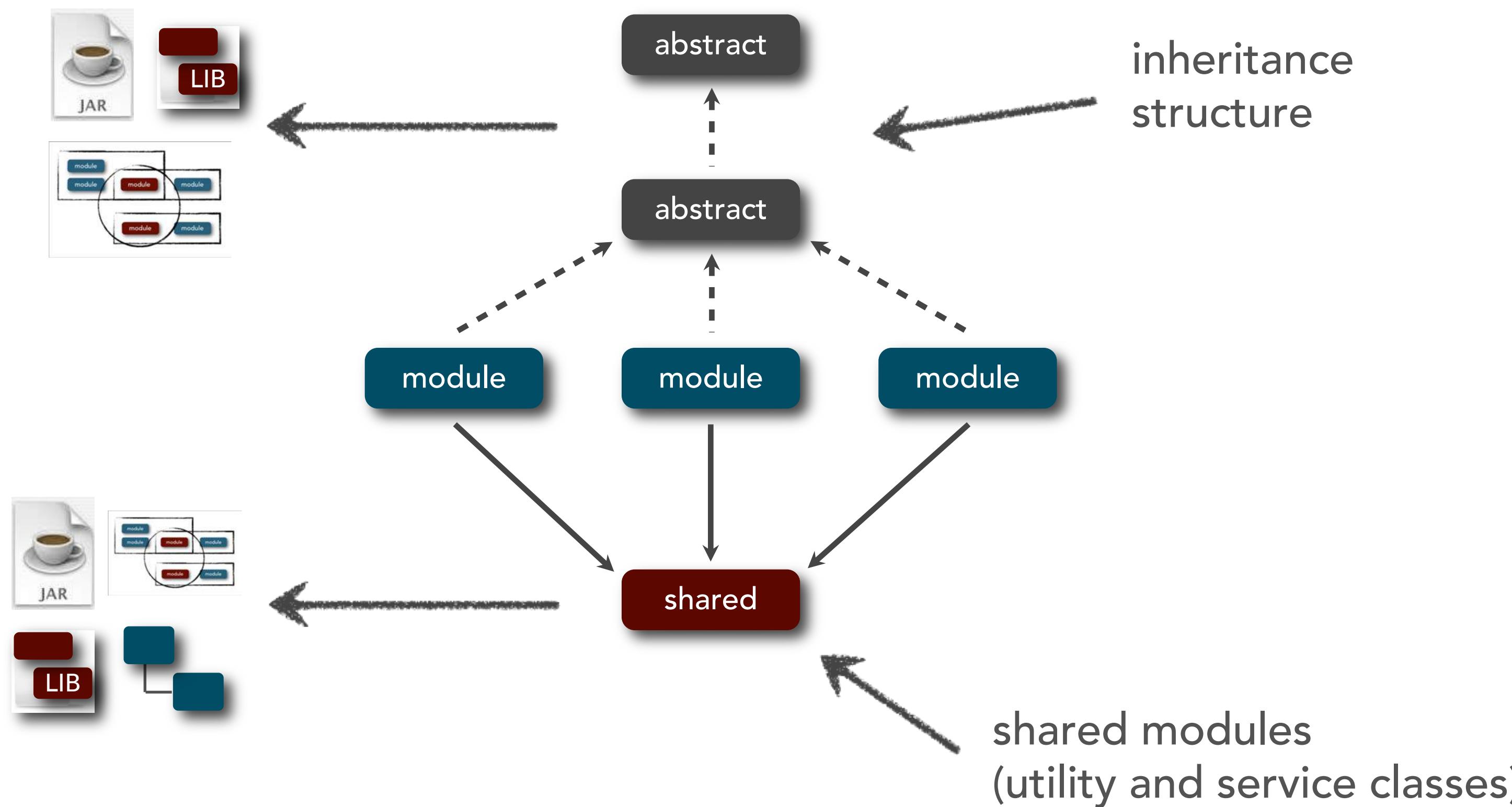
migration challenges

shared components and object hierarchies



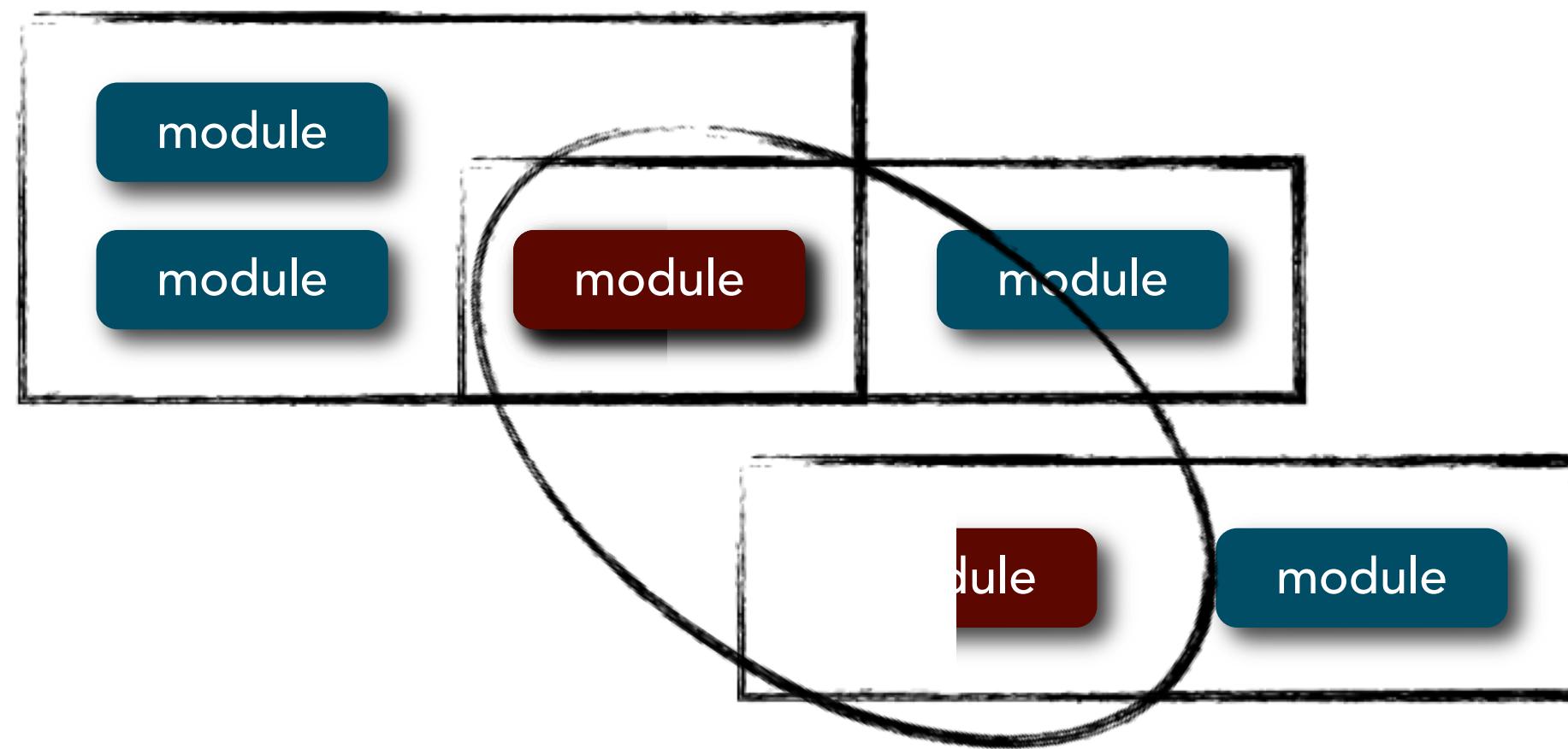
migration challenges

shared components and object hierarchies



migration challenges

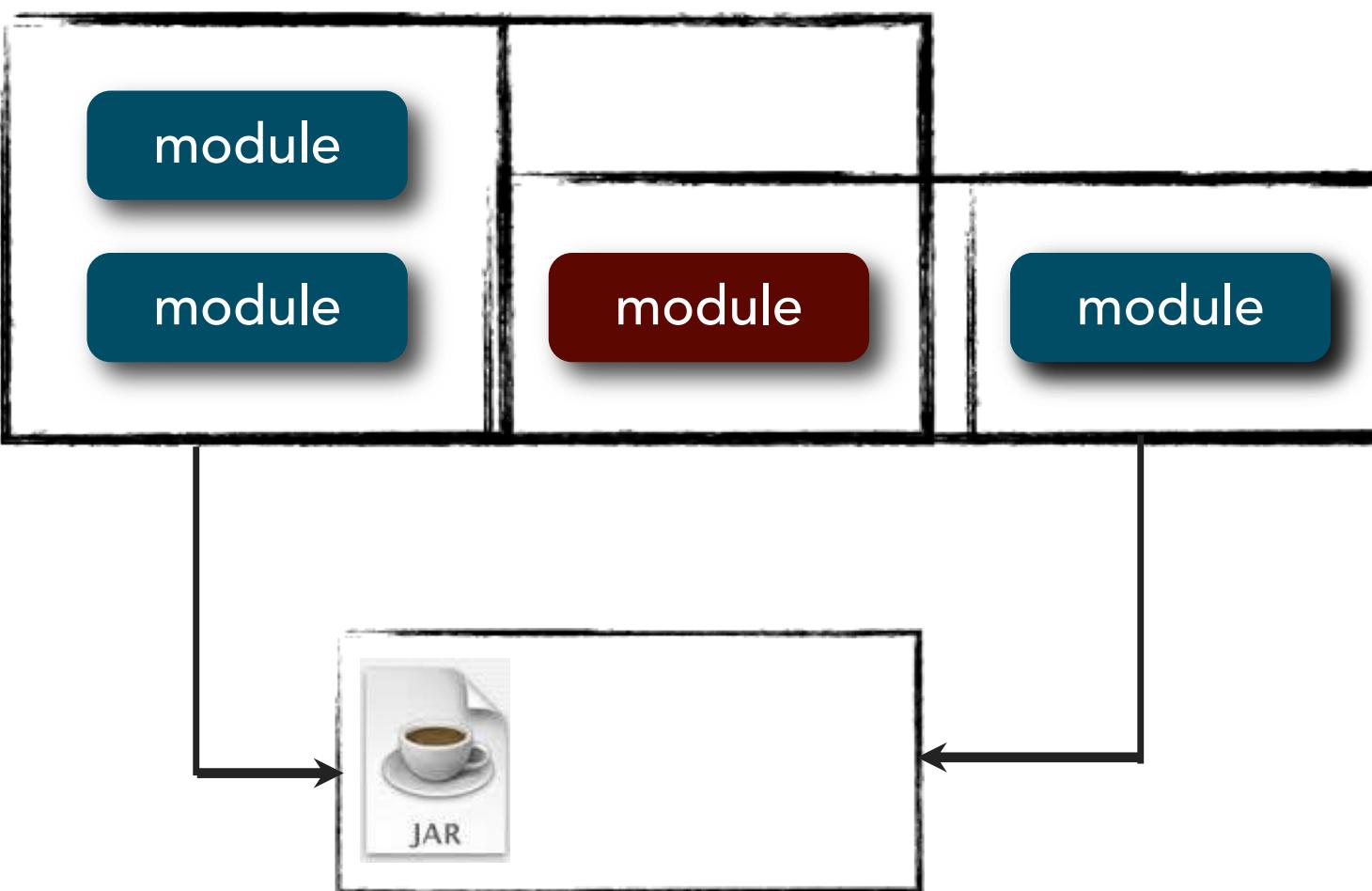
shared components and object hierarchies



modules can be split....

migration challenges

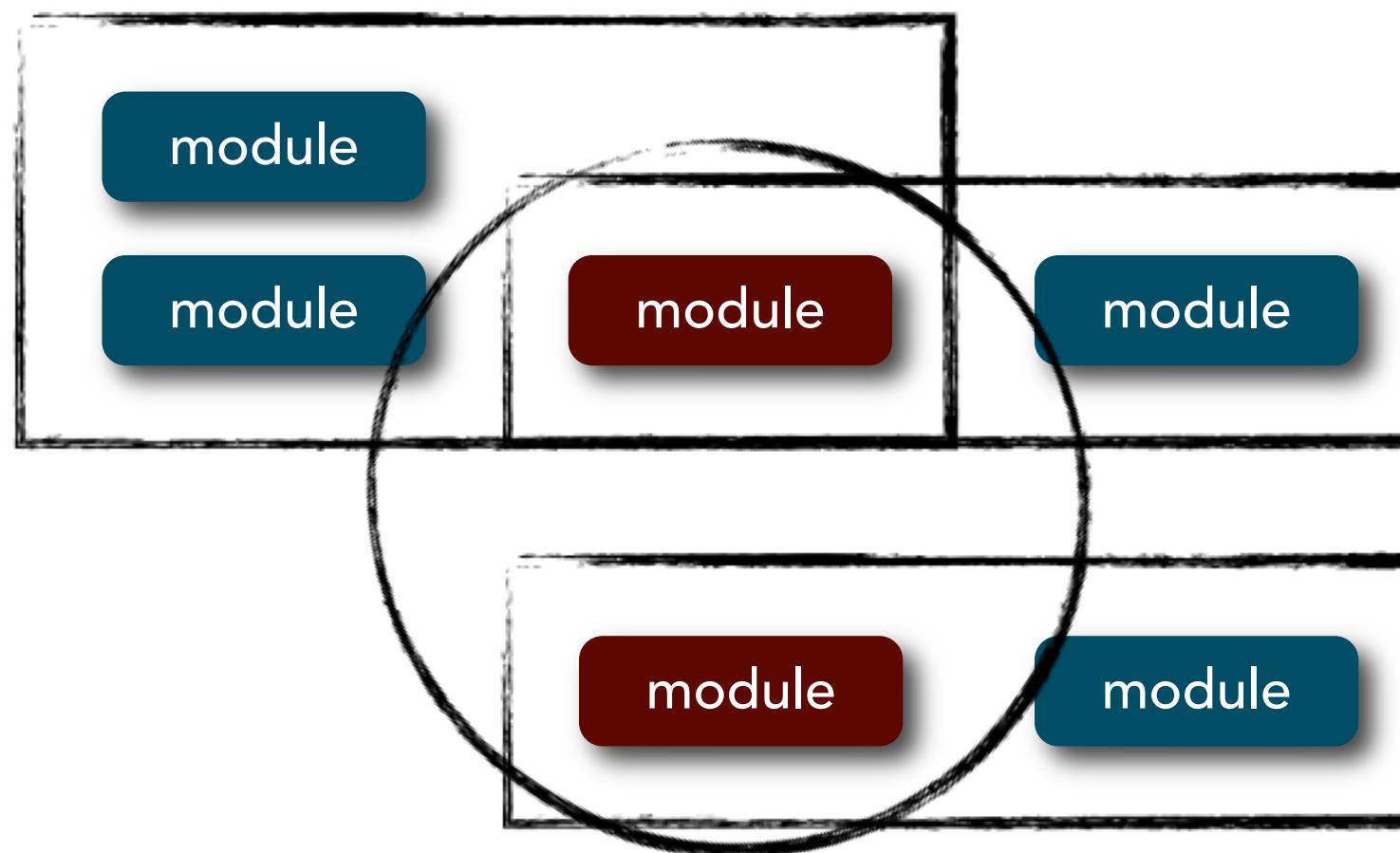
shared components and object hierarchies



or moved into a shared library or jar file...

migration challenges

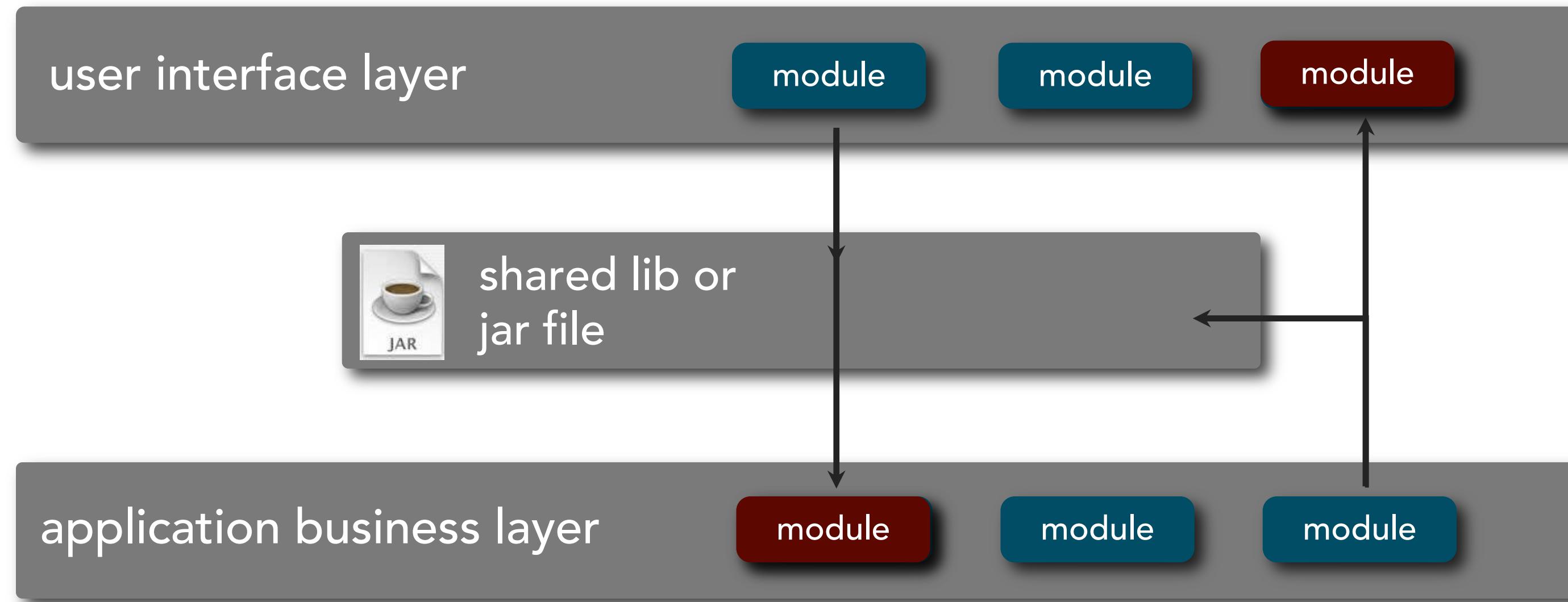
shared components and object hierarchies



or replicated in each service component

migration challenges

shared components and object hierarchies

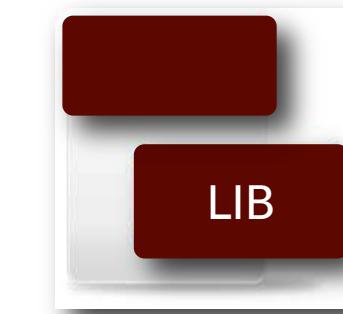


migration challenges

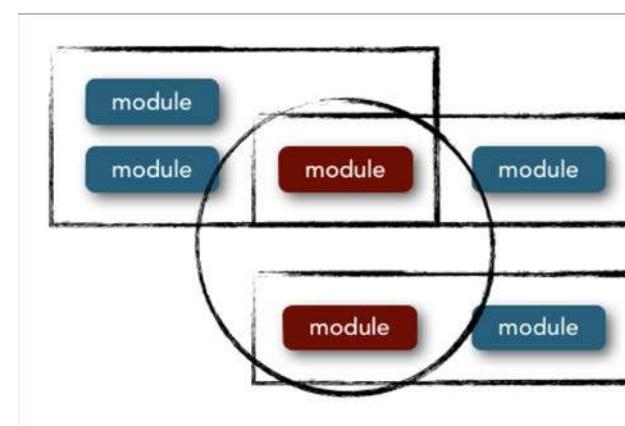
shared component techniques



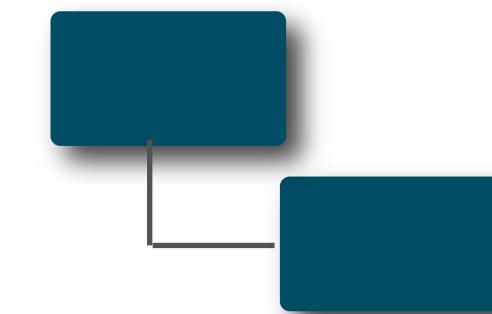
jar /dll
(compile or runtime)



shared library
(compile time)

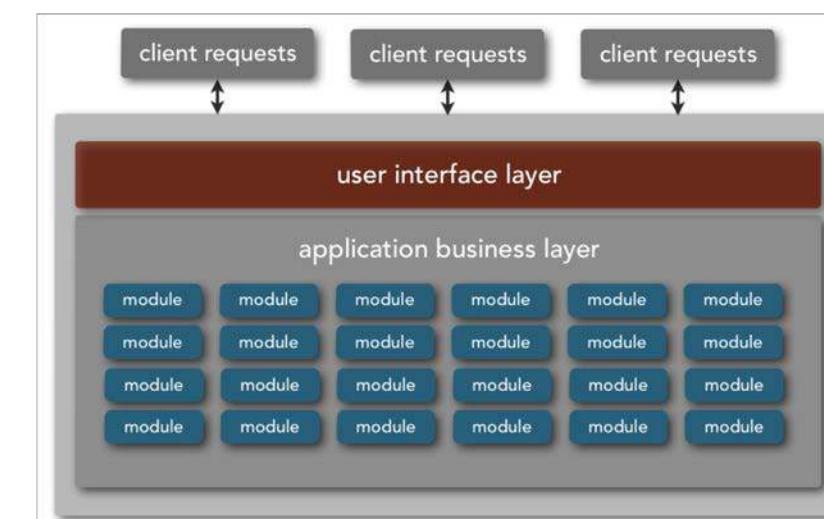


code replication

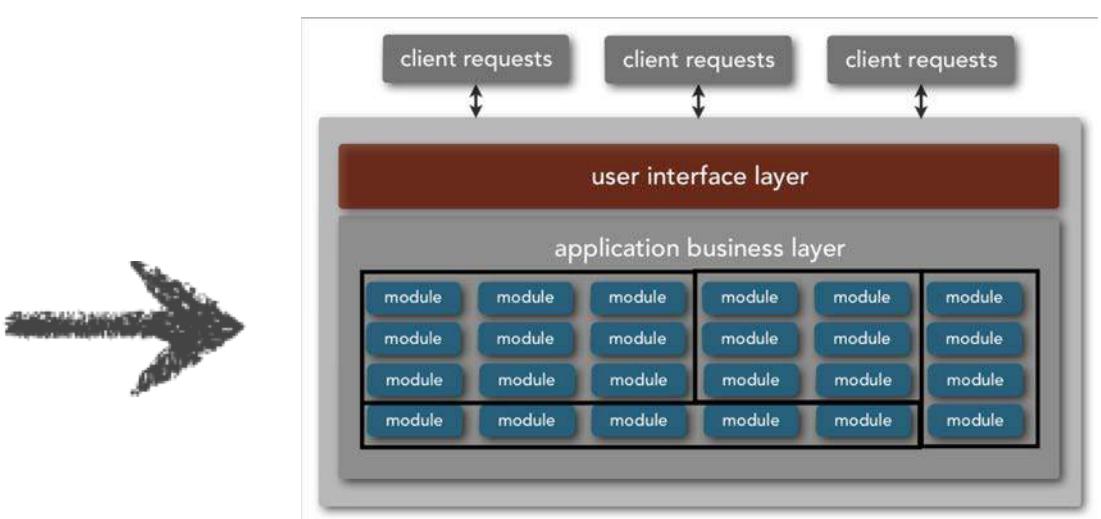


remote services

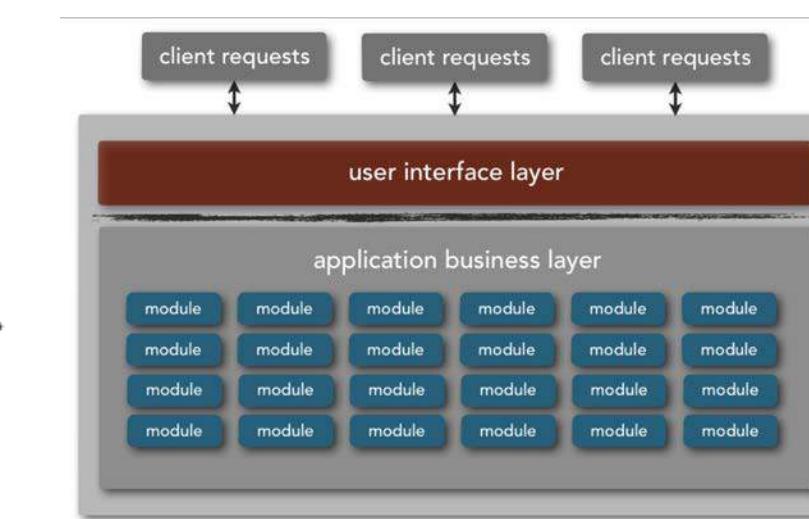
migration steps - refactoring



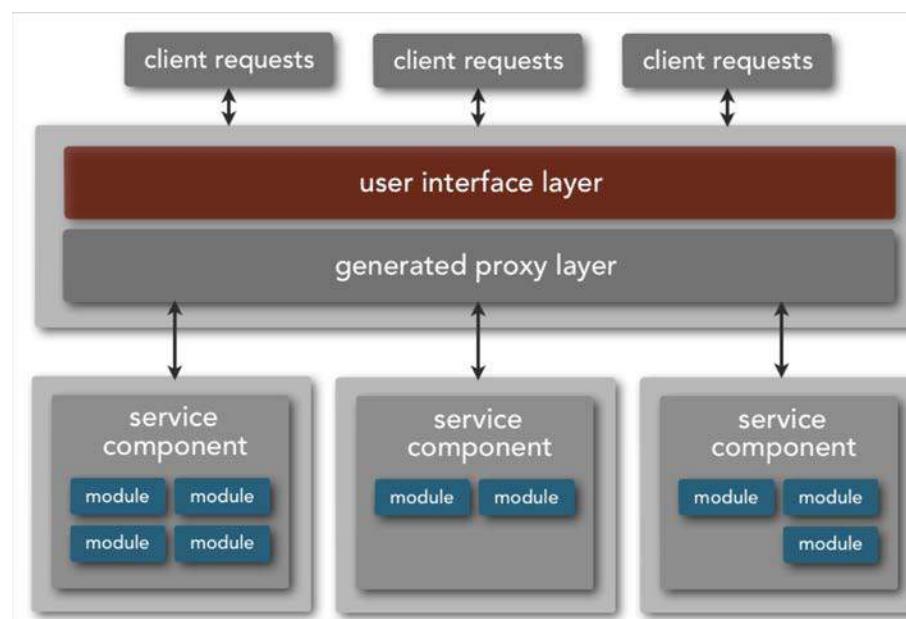
monolith (start)



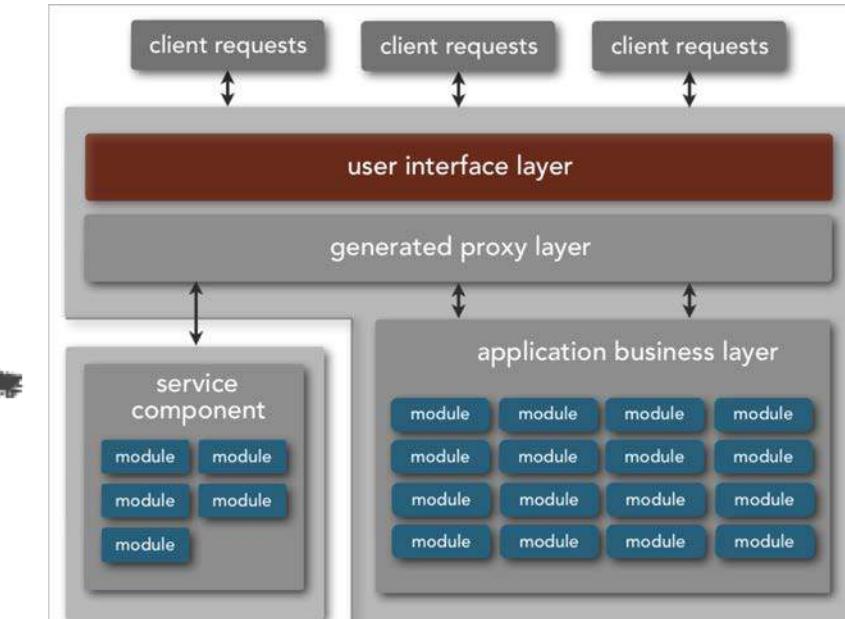
identification



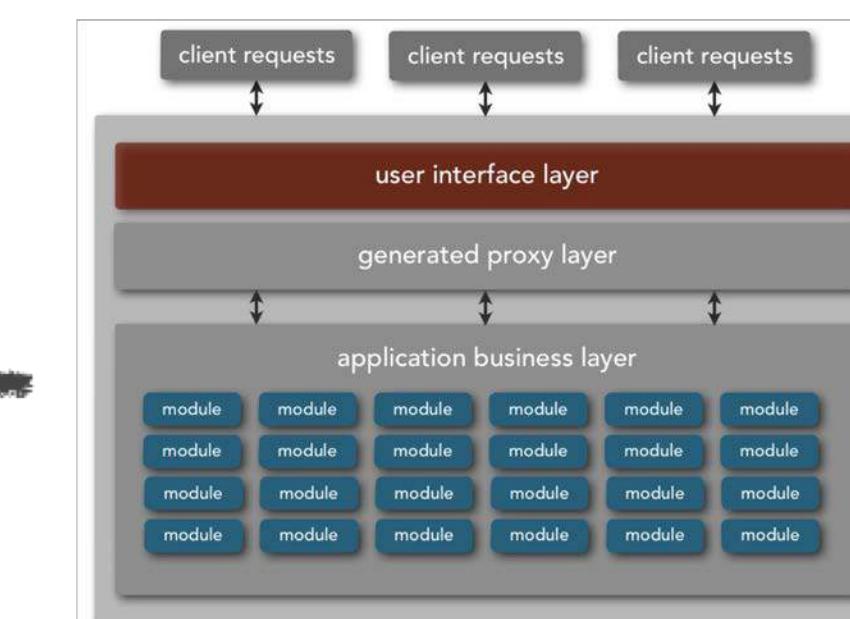
separation



microservices (end)



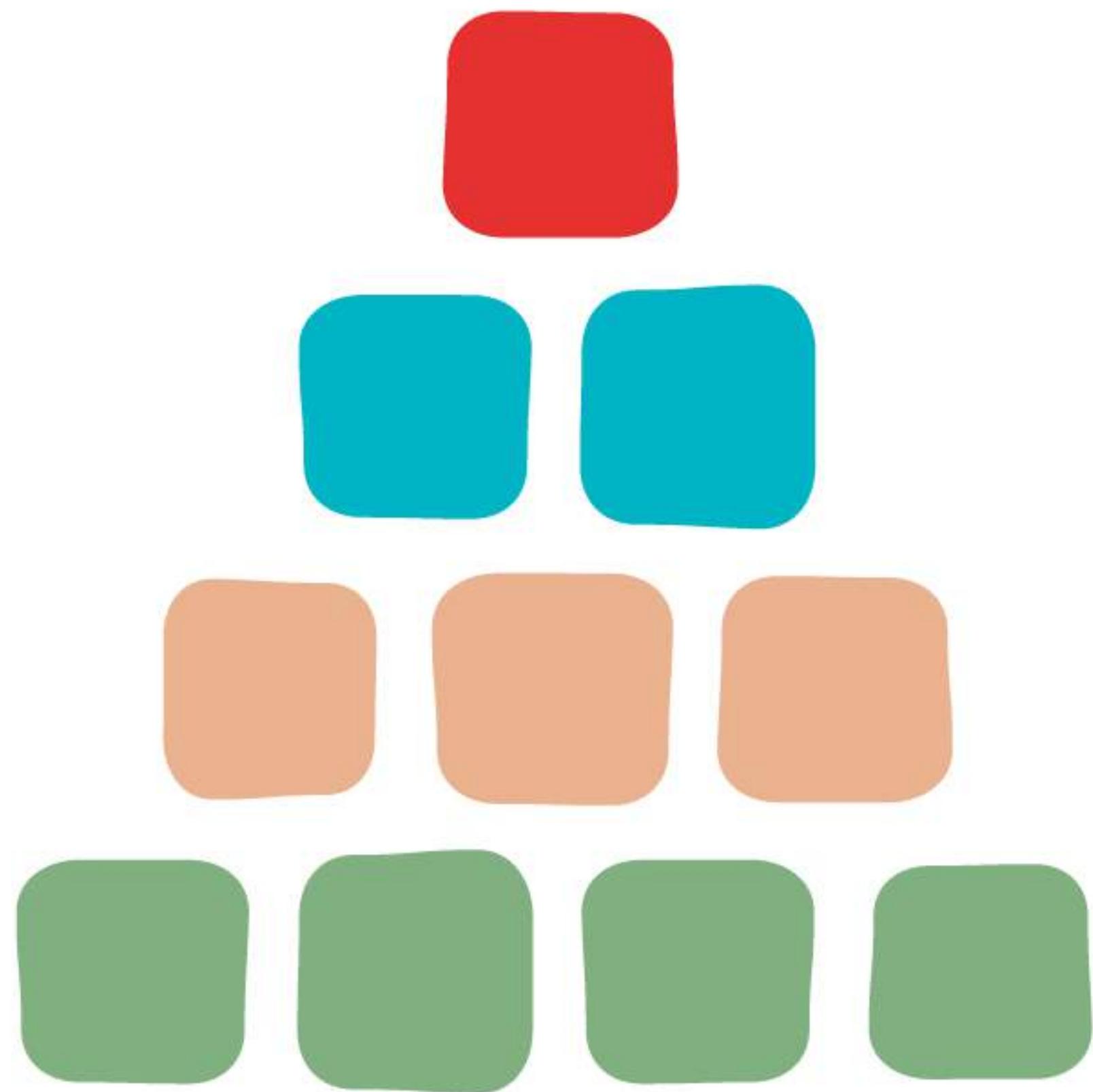
migration



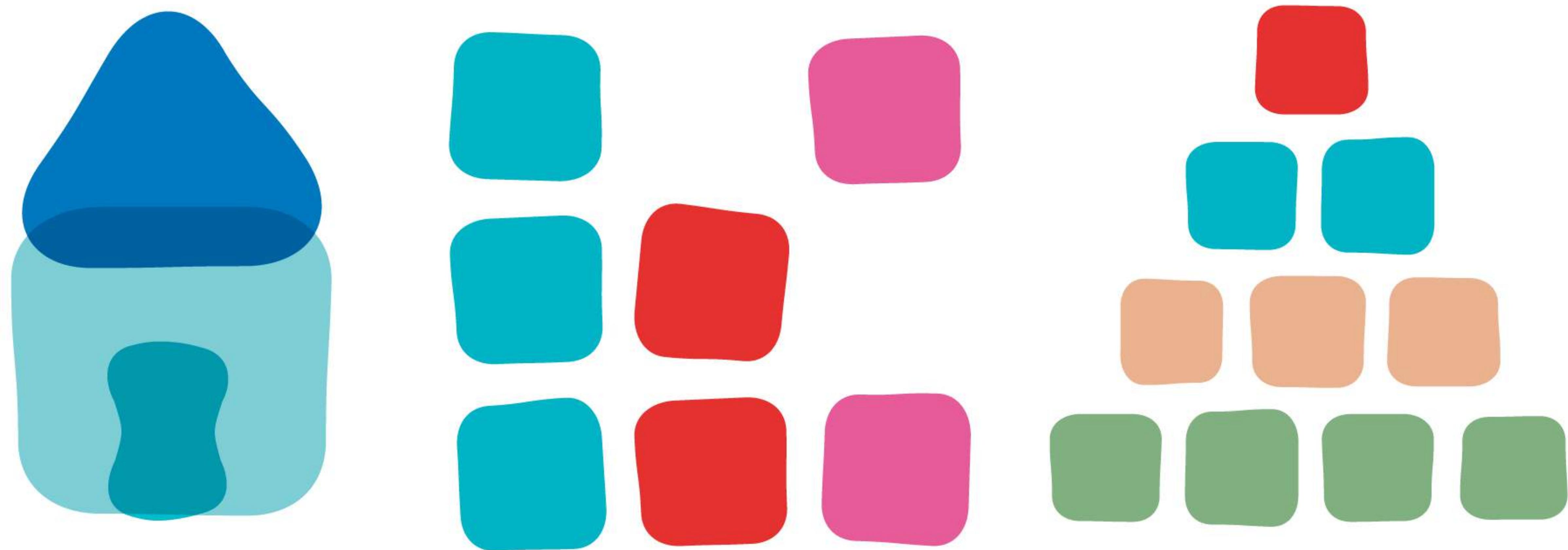
lookup proxy

migration advice

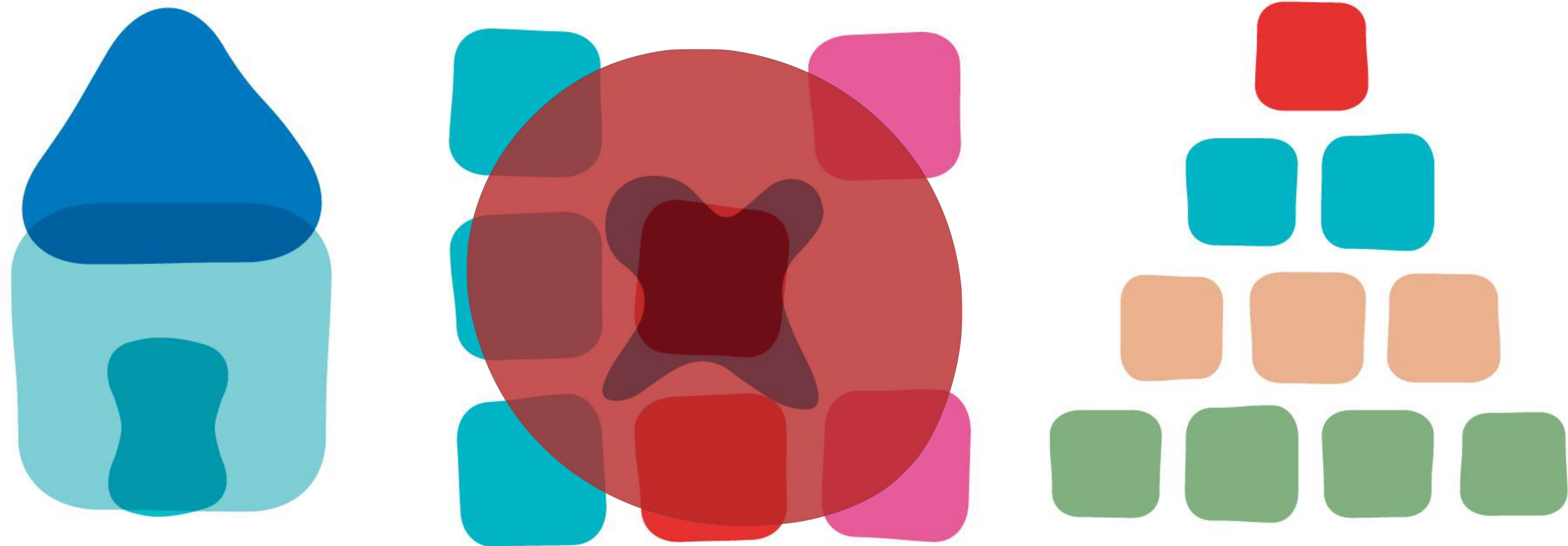
Domain/Architecture Isomorphism



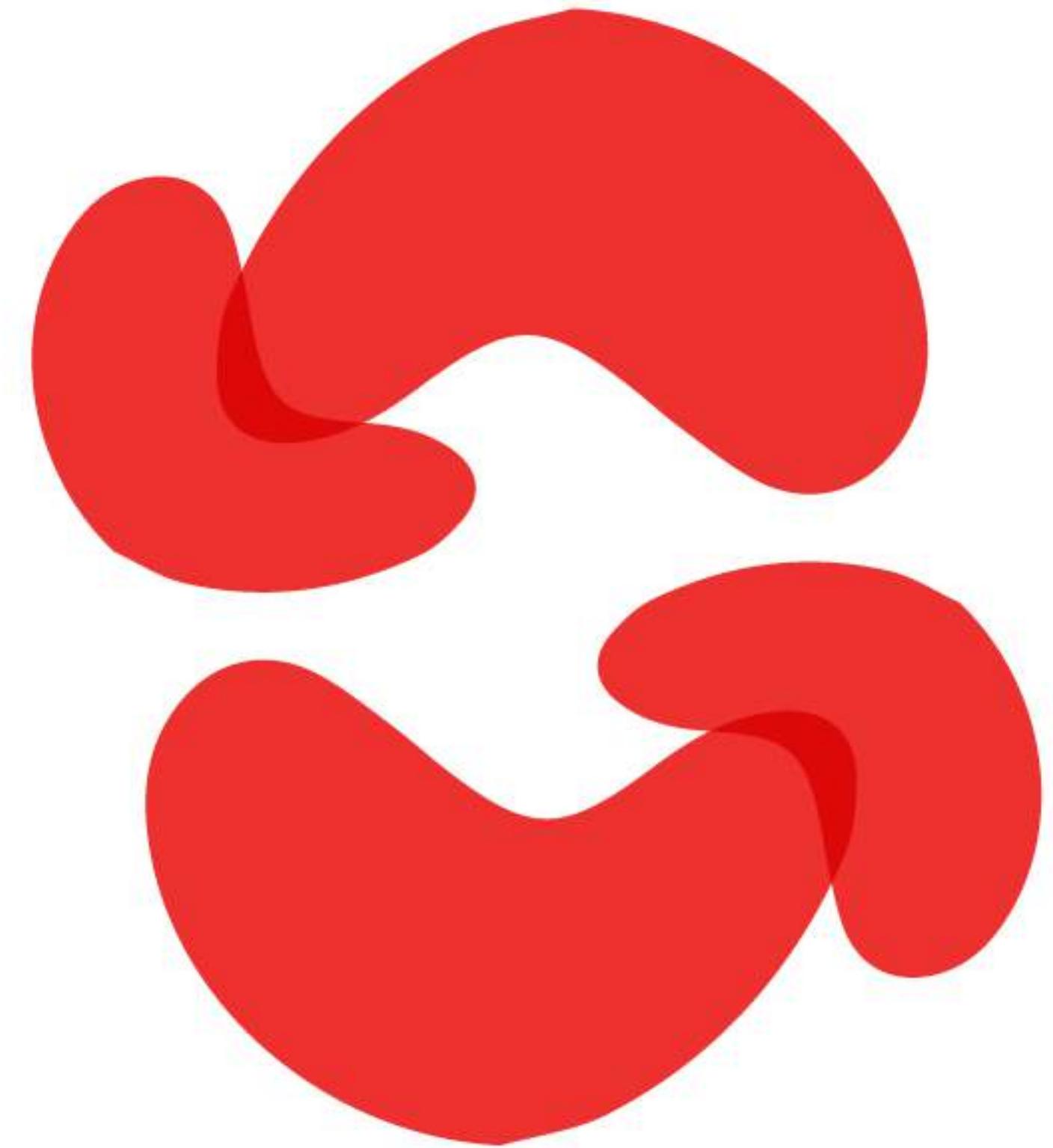
Domain/Architecture Isomorphism



Domain/Architecture Isomorphism



iteration



cOmPaRiNg SeRViCE-bASeD aRchiTecTureS



ThoughtWorks®

NEAL FORD

Director / Software Architect / Meme Wrangler



@neal4d

nealford.com