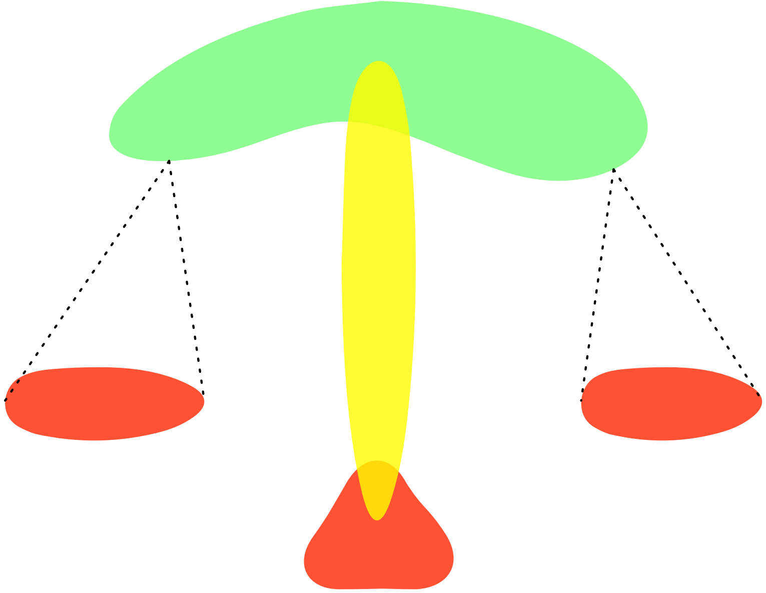
# Comparing Service-based Architectures



ThoughtWorks®

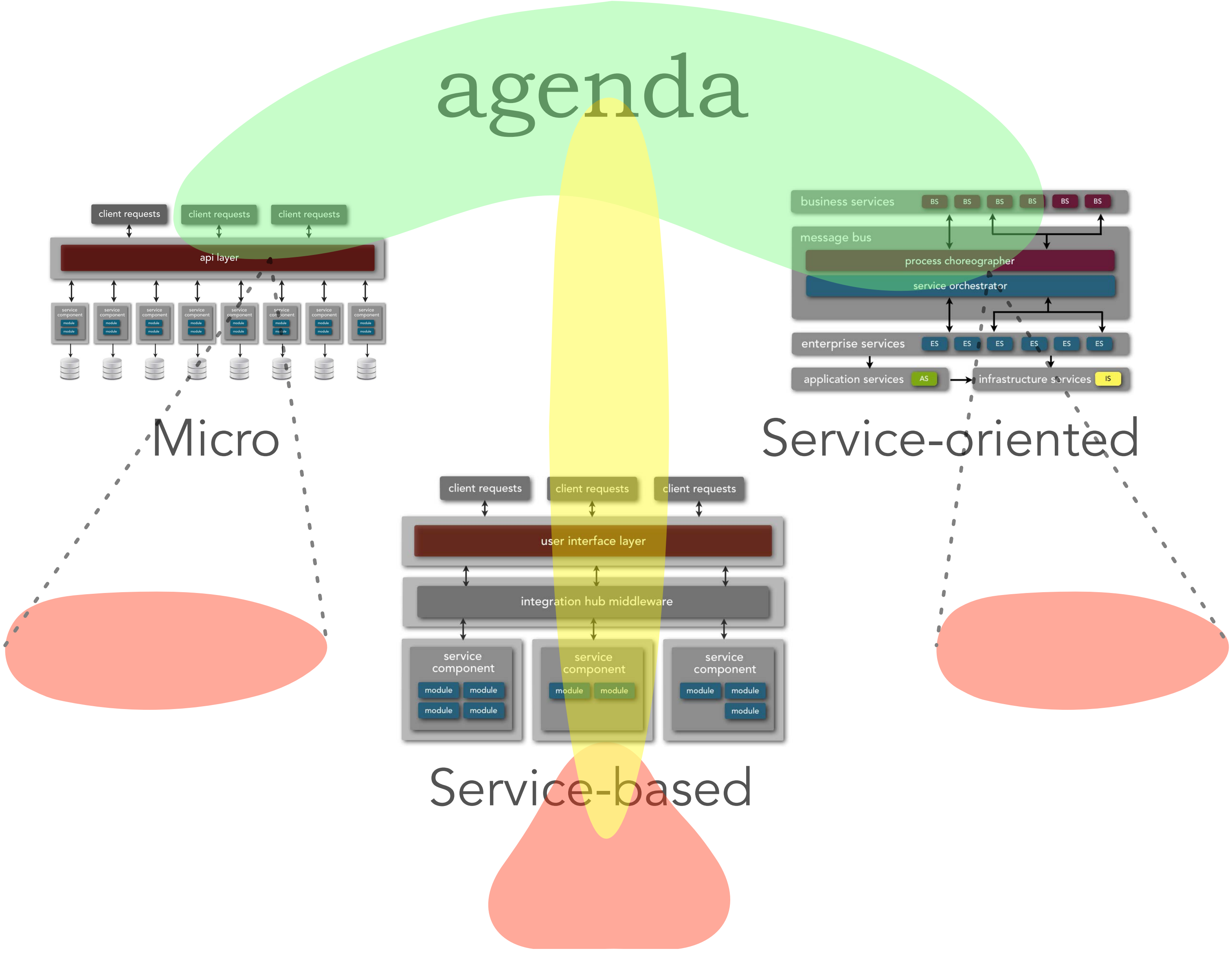
NEAL FORD

*Director / Software Architect / Meme Wrangler*

*@neal4d*

*nealford.com*

# agenda

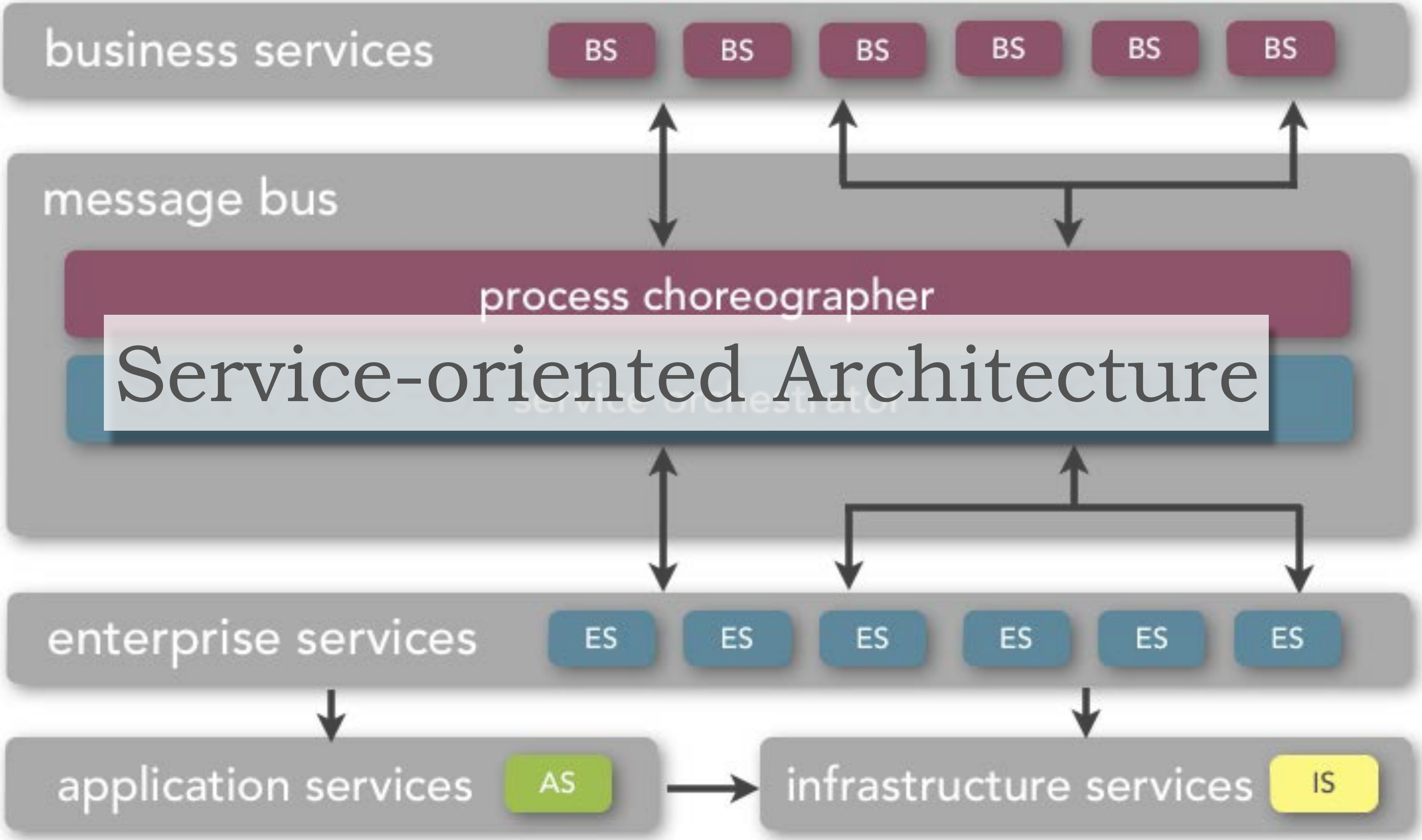


Micro

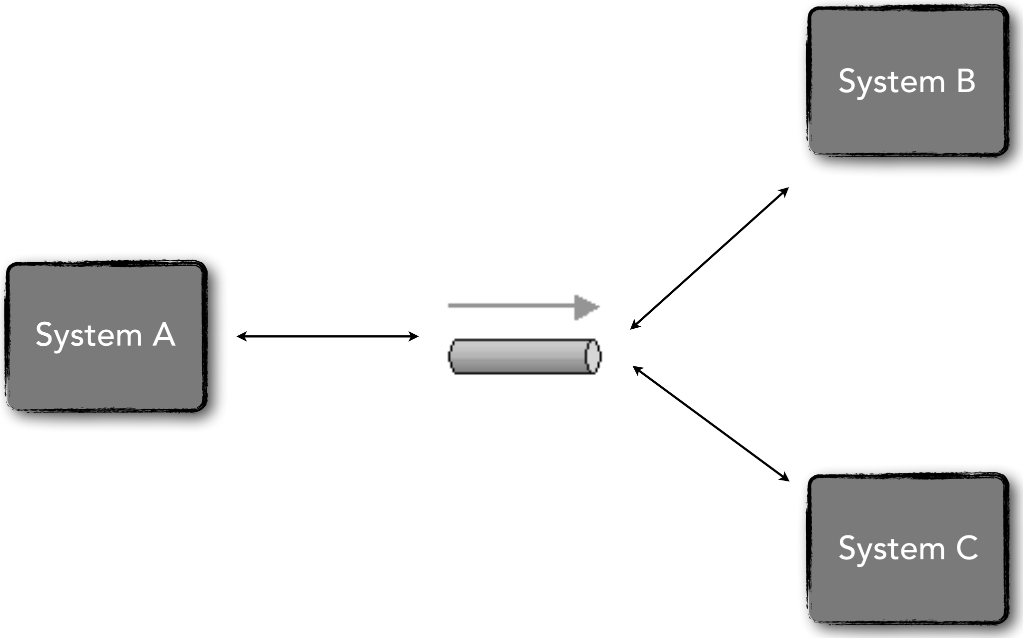
Service-oriented

Service-based

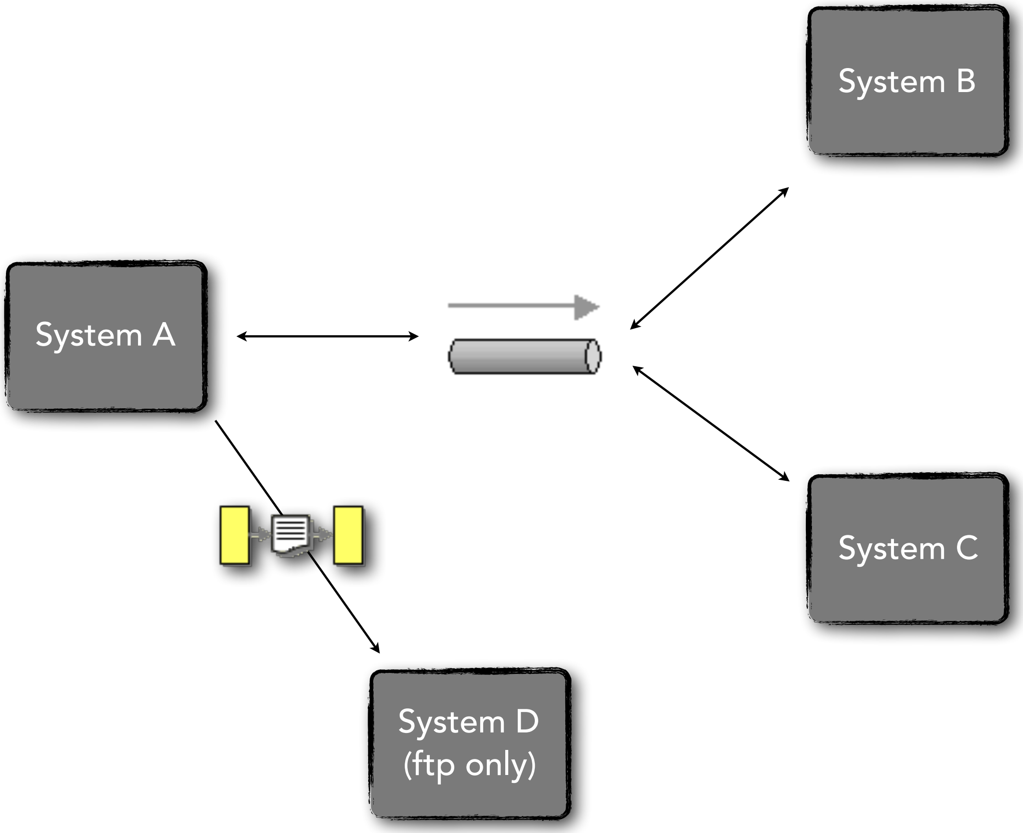
# Service-oriented Architecture



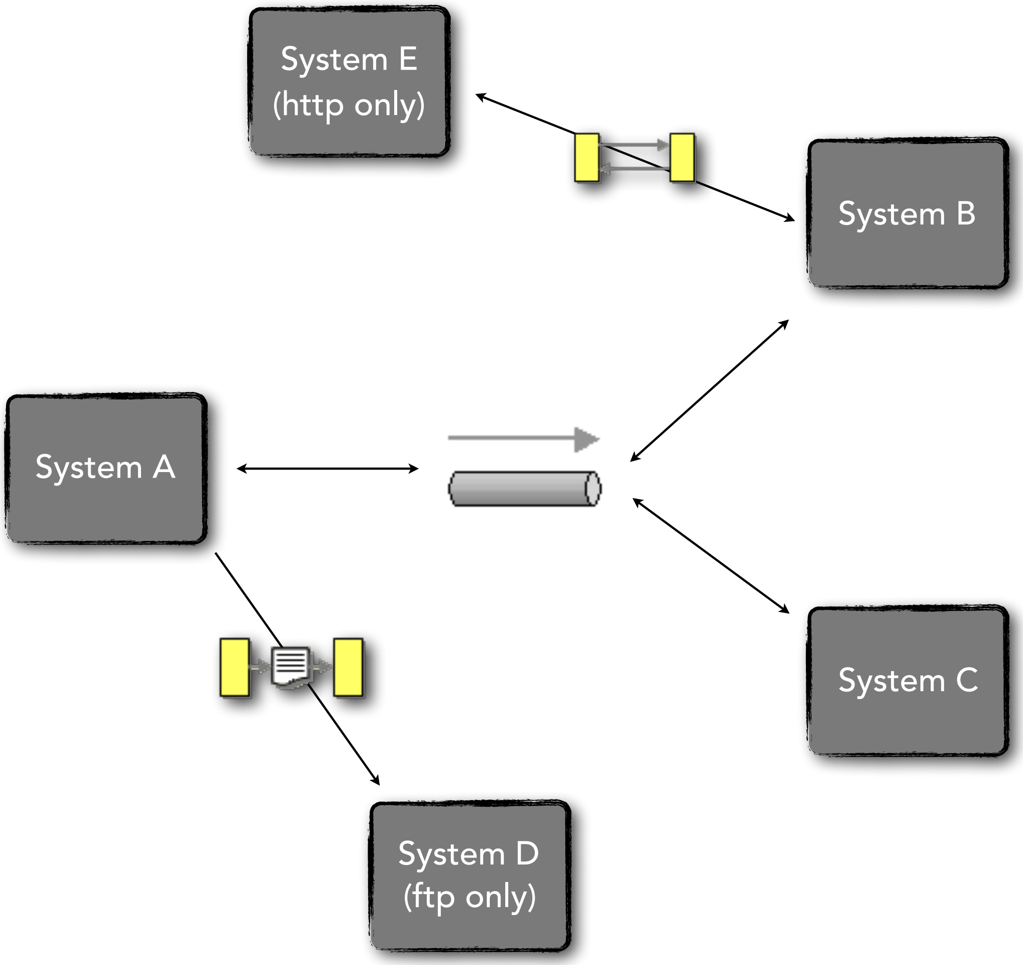
# origins: hubs



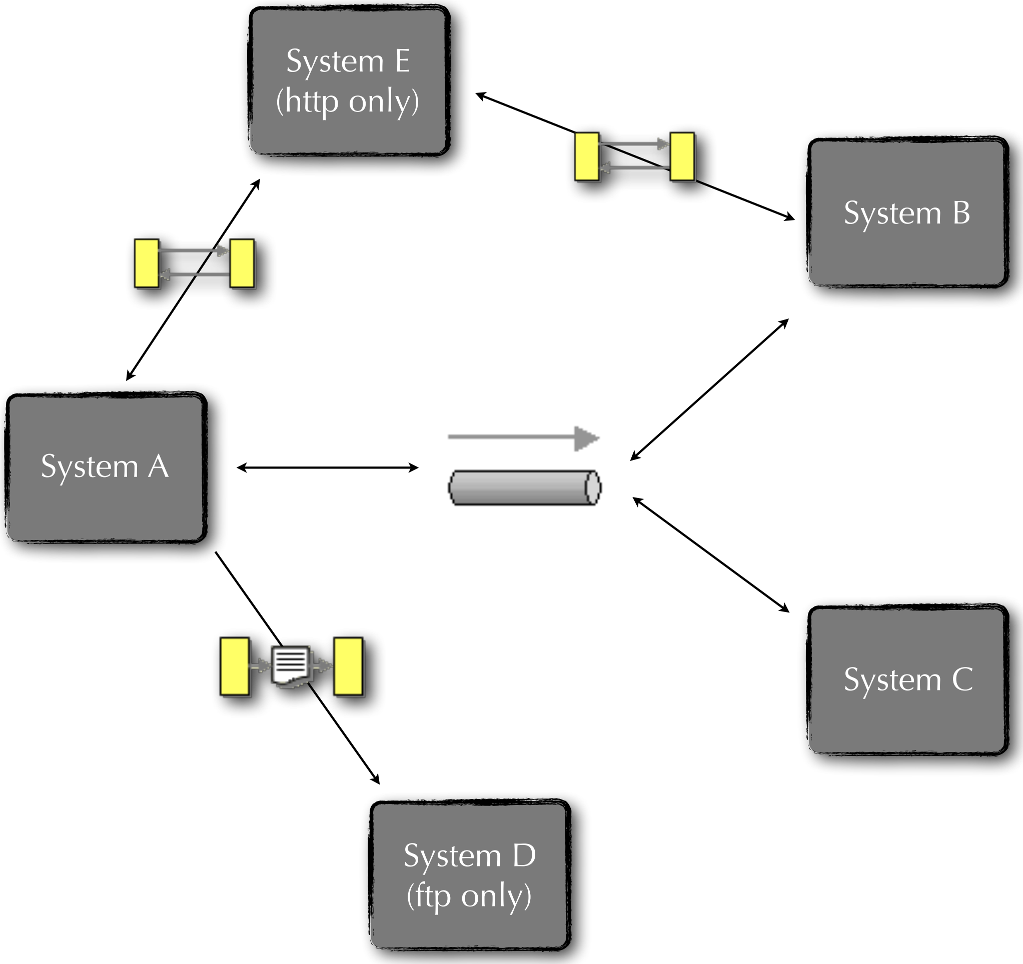
# origins: hubs



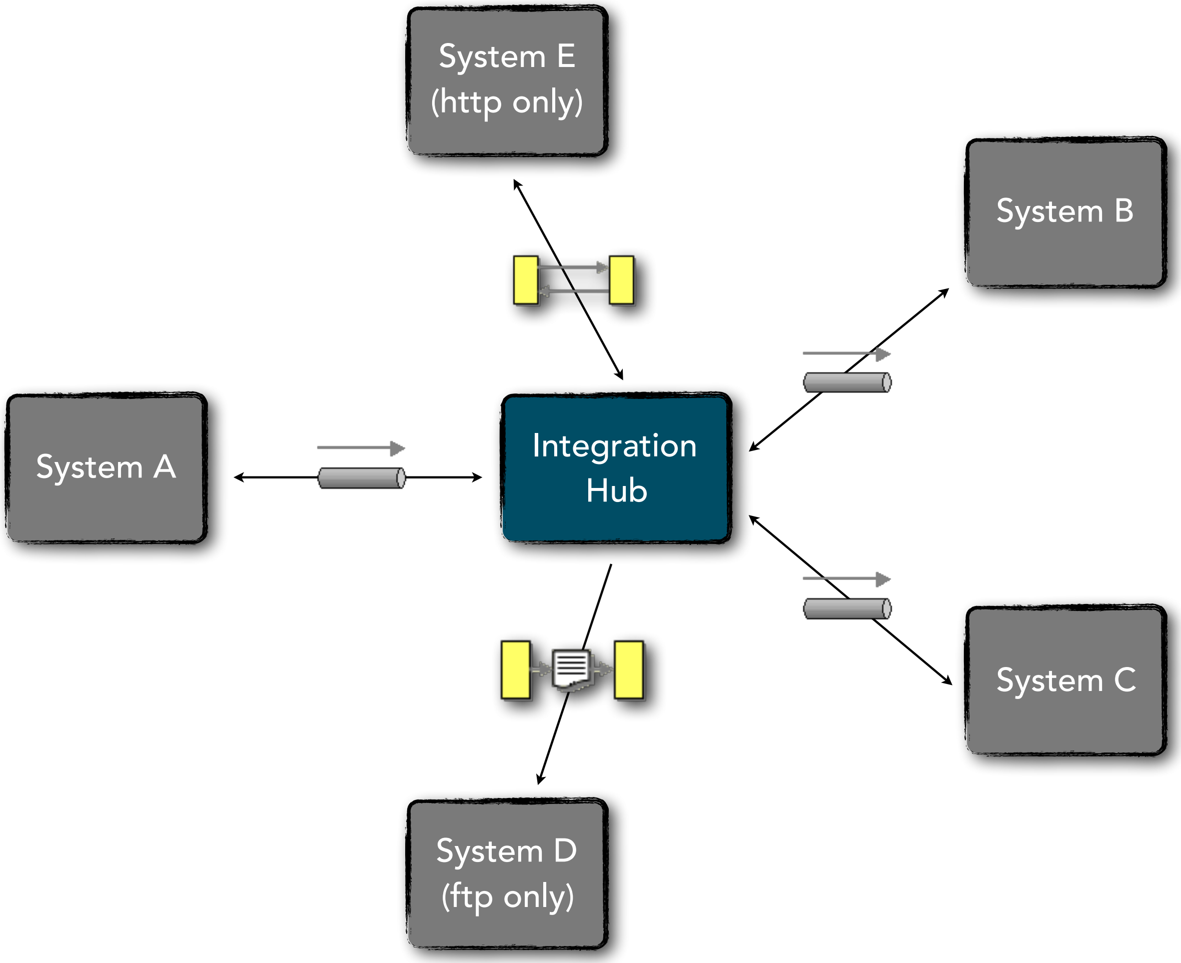
# origins: hubs



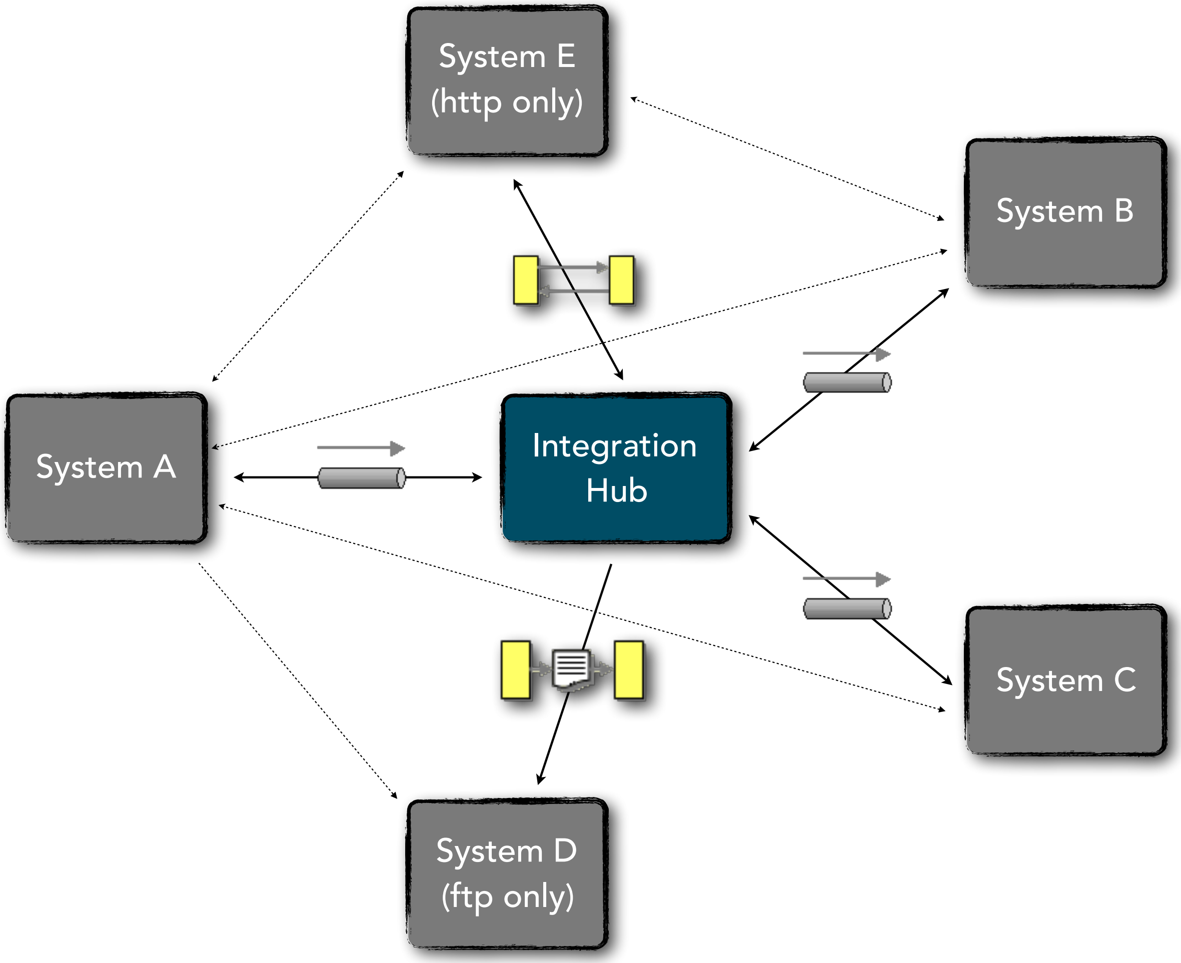
# origins: hubs



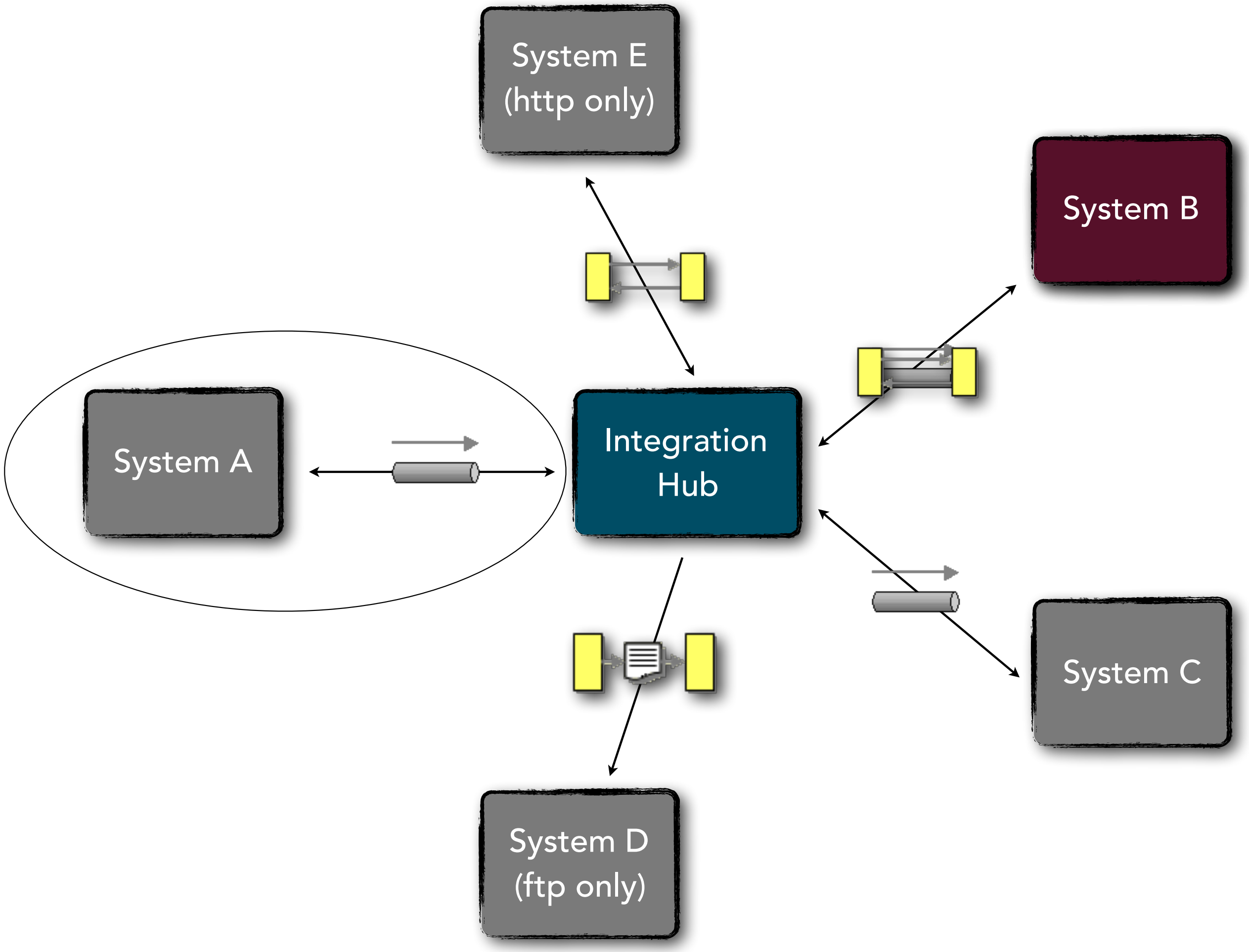
# origins: hubs



# origins: hubs

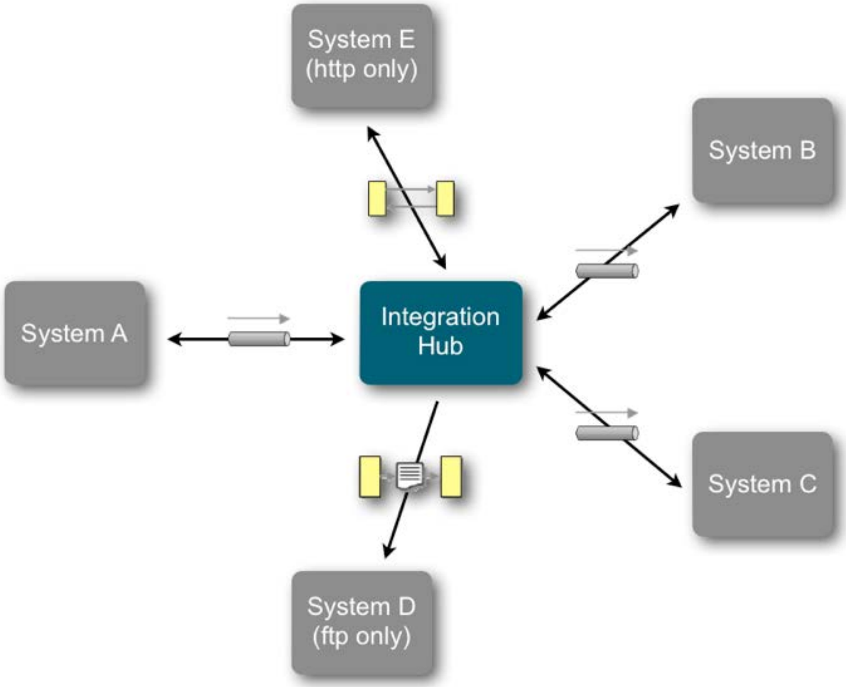


# origins: hubs

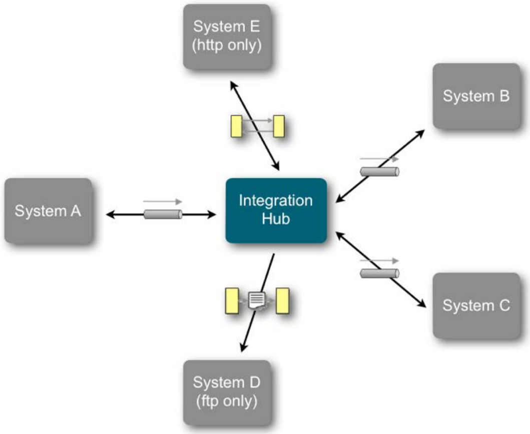


# origins: hubs

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



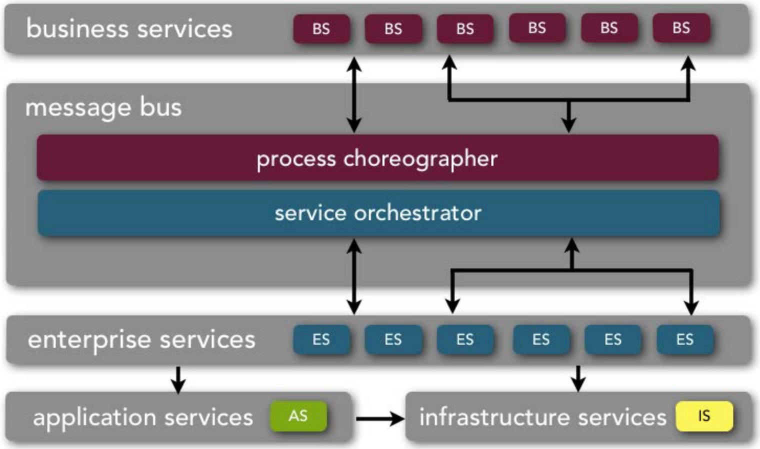
# orchestration



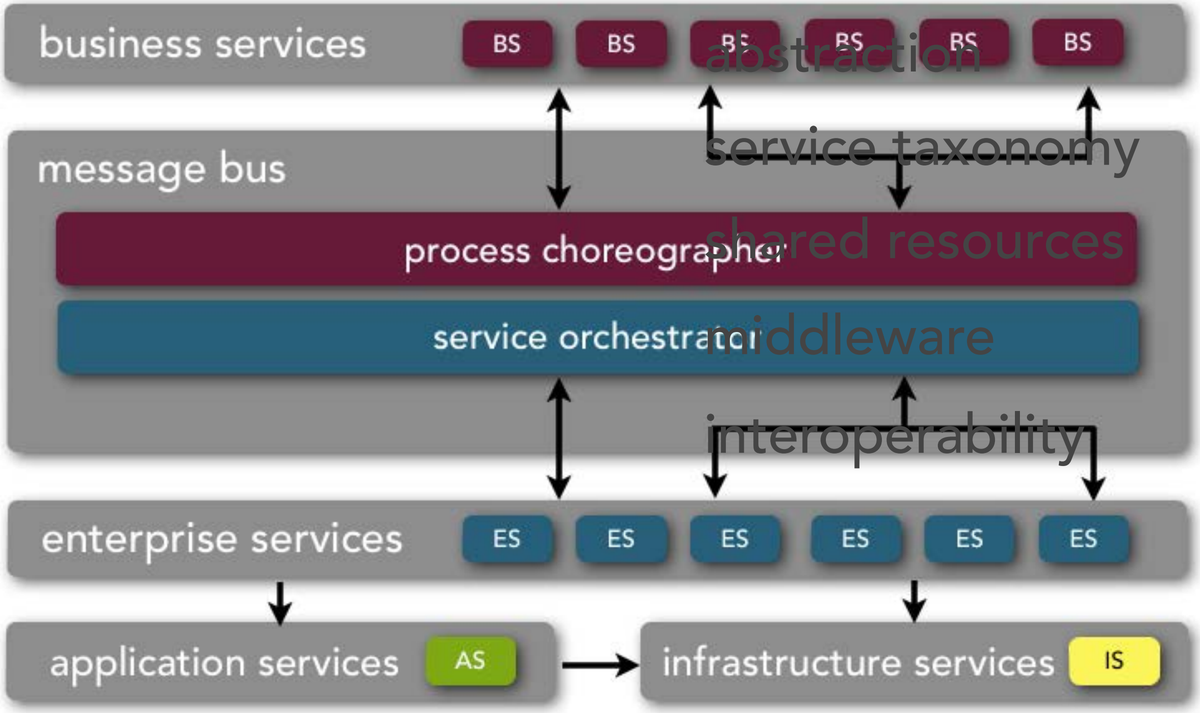
hub

intelligent hub

service oriented architecture /
enterprise service bus pattern



# service-oriented architecture



abstraction

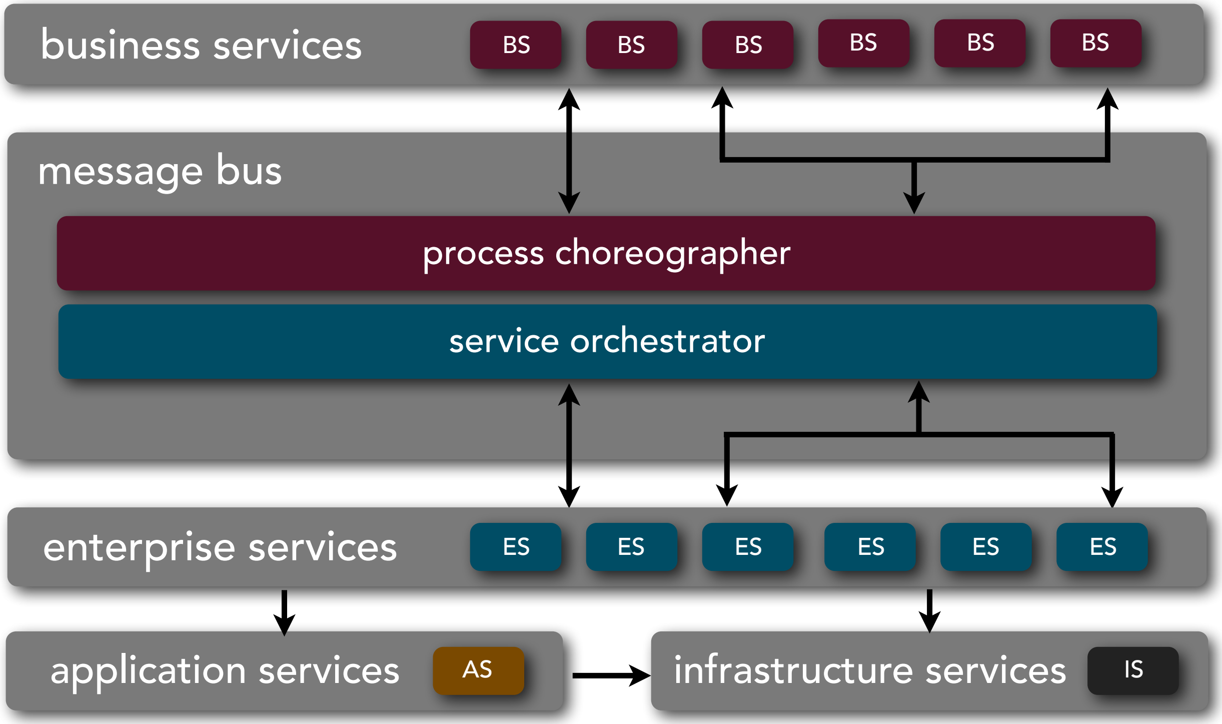
service taxonomy

shared resources

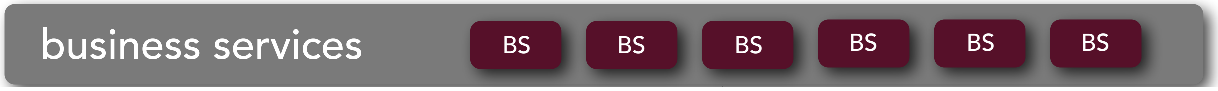
middleware

interoperability

# service-oriented architecture



# 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

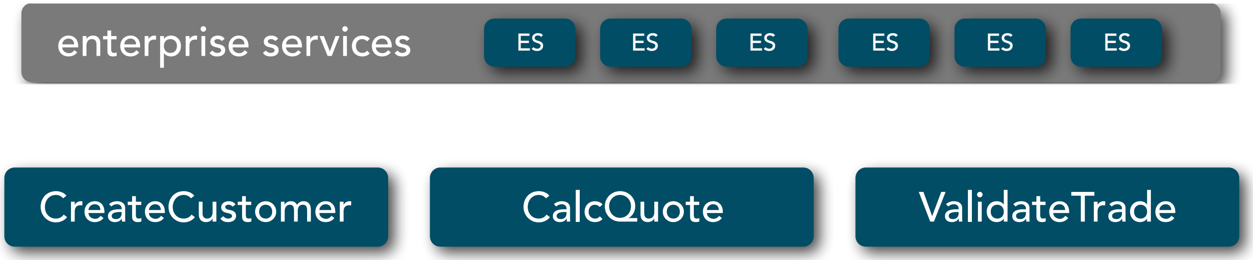
owned by shared services teams

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

owned by application teams

concrete

application-level

fine-grained

bound to a specific application context

AddDriver

UpdateAddress

CalcSalesTax

application services AS

# 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

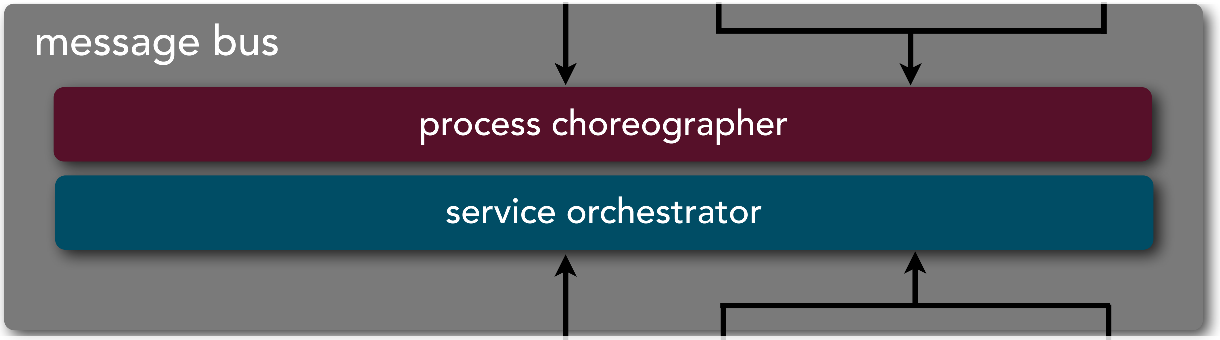
WriteAudit

CheckUserAccess

LogError

infrastructure services IS

# service-oriented architecture



mediation and routing

process choreography

service orchestration

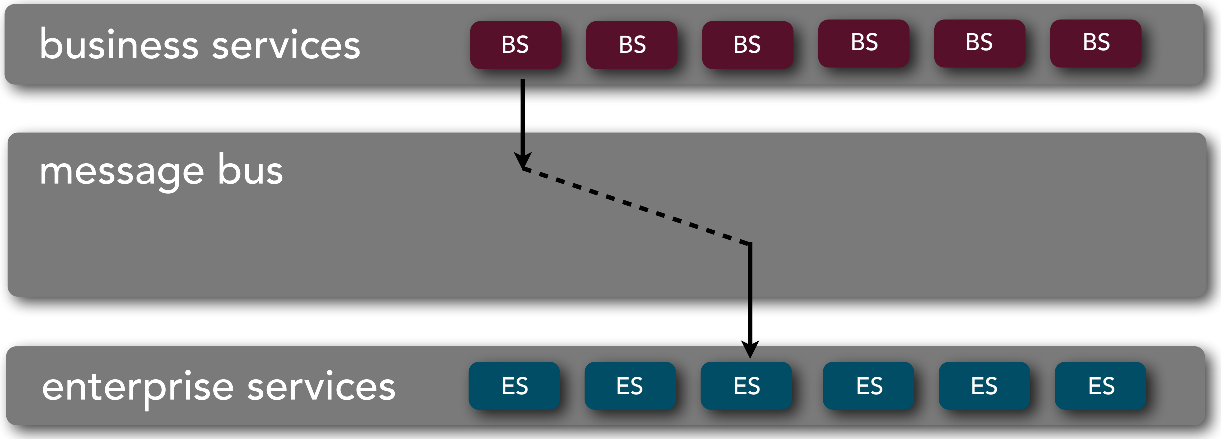
message enhancement

message transformation

protocol transformation

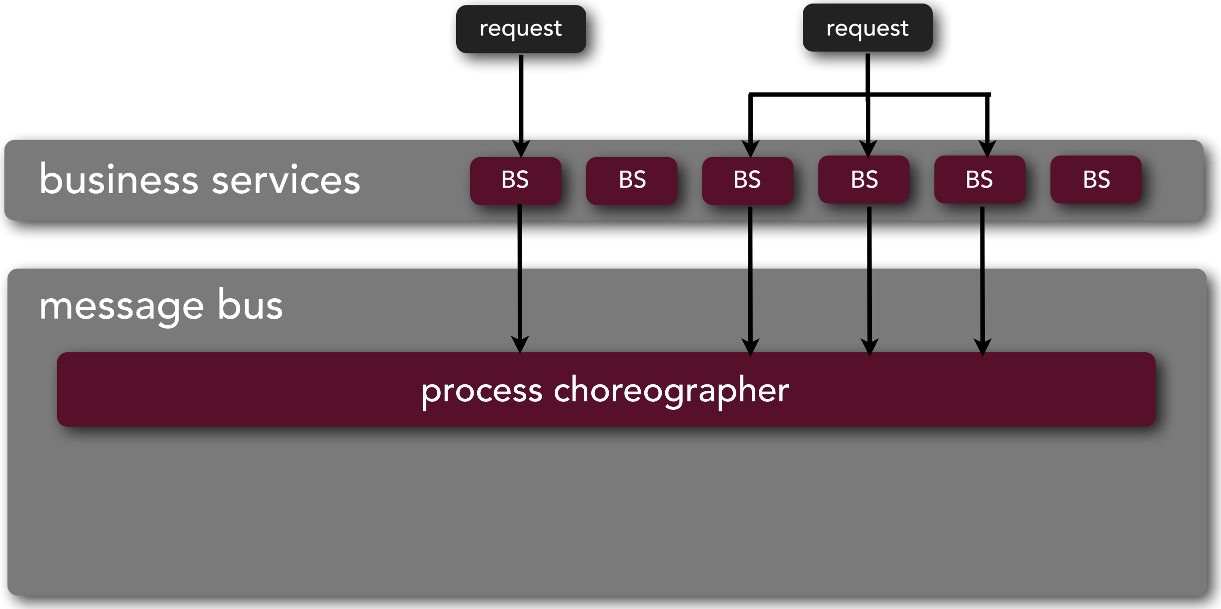
# service-oriented architecture

## mediation and routing



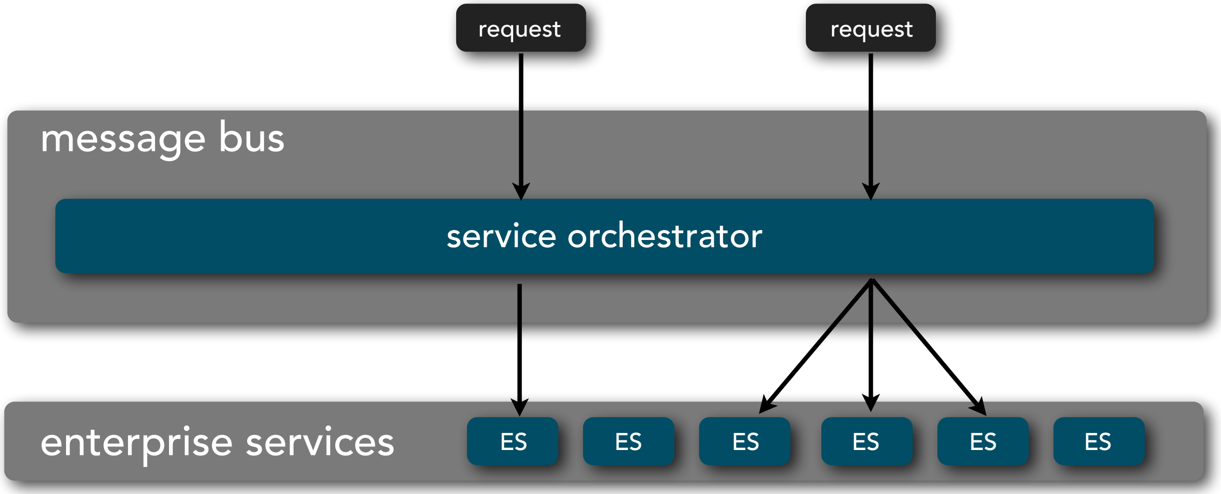
# service-oriented architecture

## process choreography



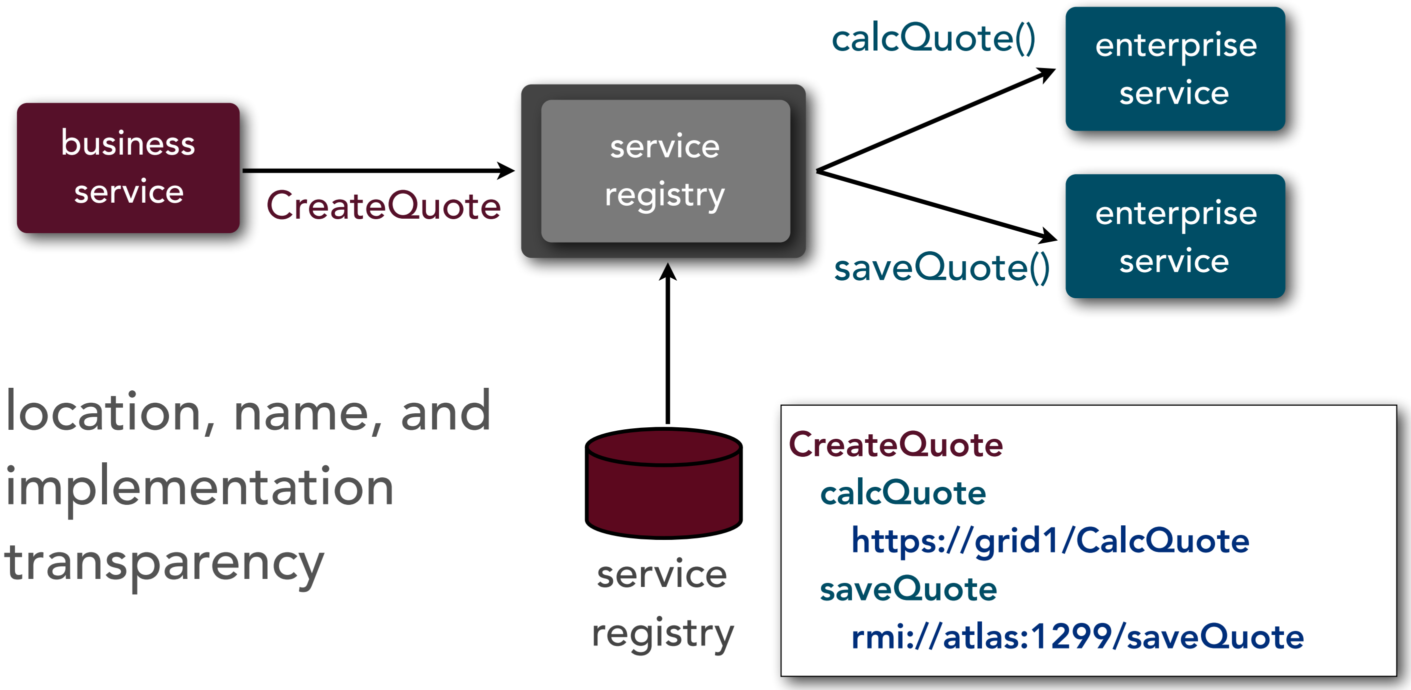
# service-oriented architecture

## service orchestration



# service-oriented architecture

## service registry



business service

CreateQuote

service registry

calcQuote()

saveQuote()

enterprise service

enterprise service

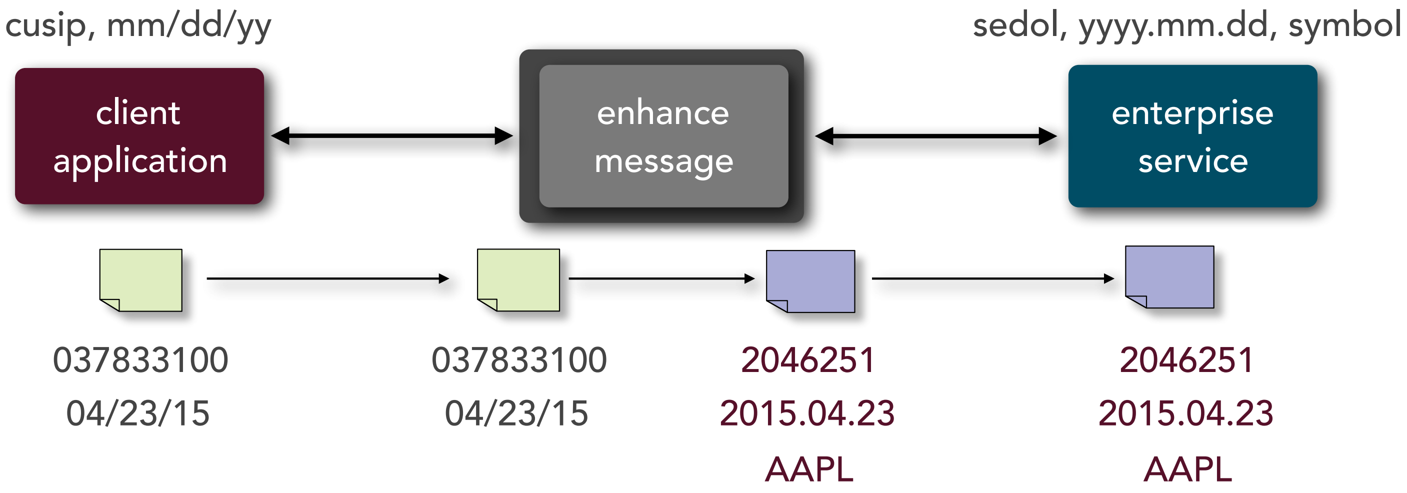
service registry

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

location, name, and implementation transparency

# service-oriented architecture

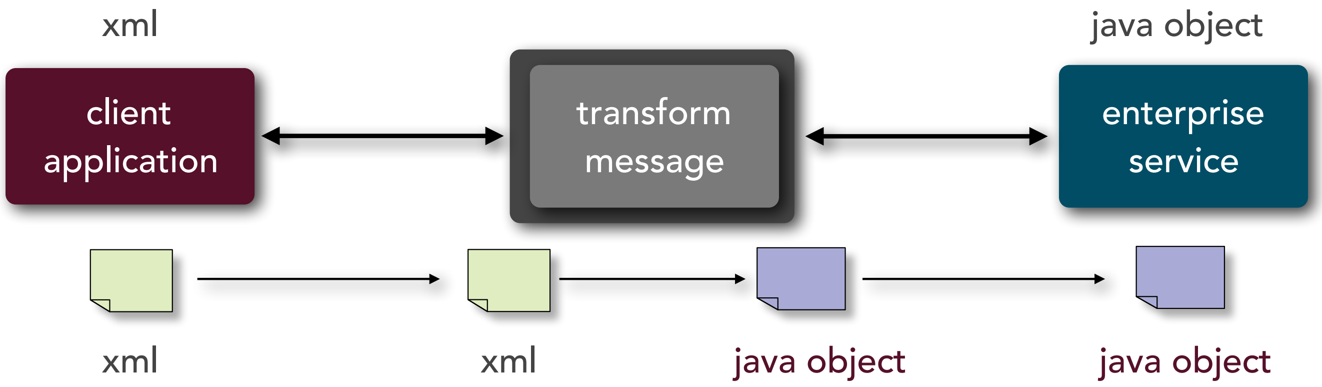
## message enhancement



contract decoupling

# service-oriented architecture

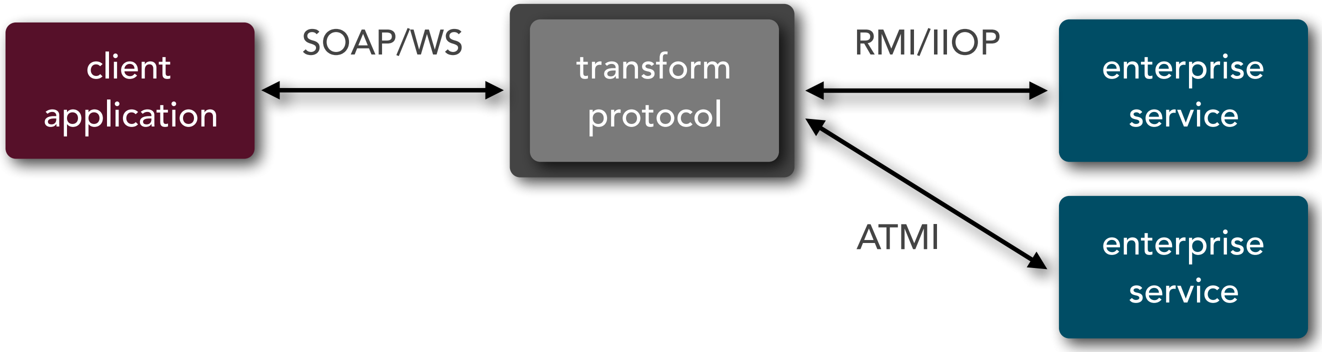
## message transformation



contract decoupling

# service-oriented architecture

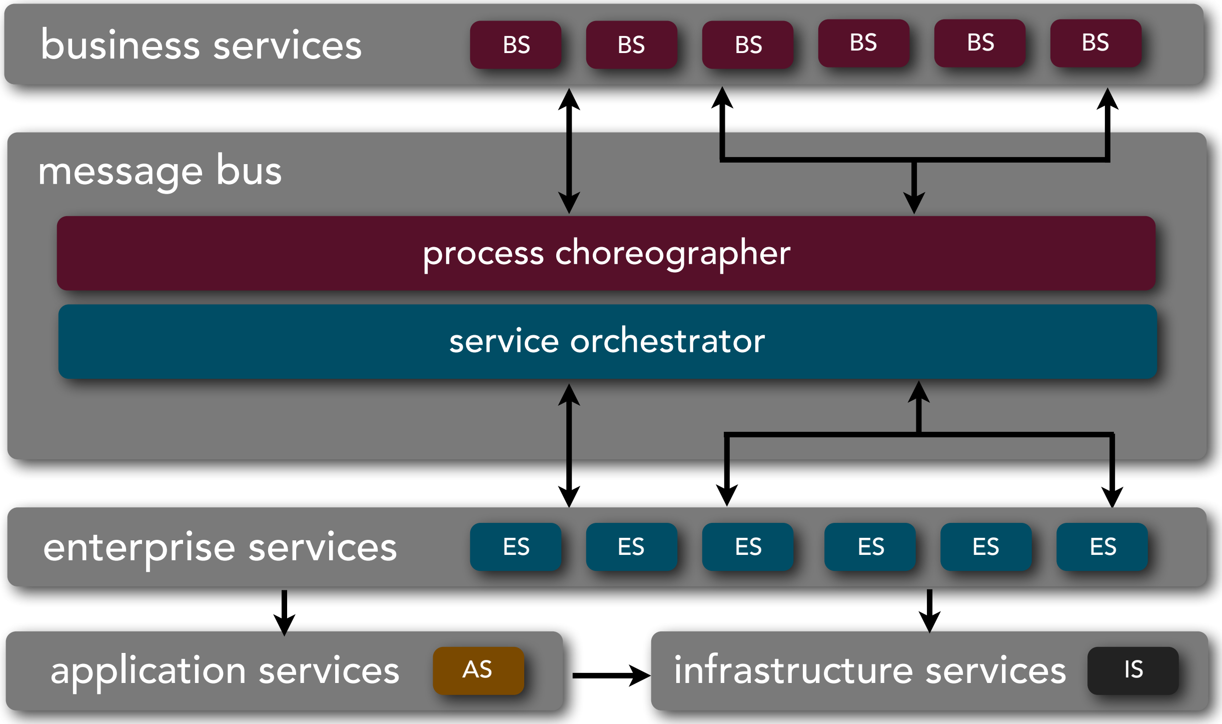
## protocol transformation



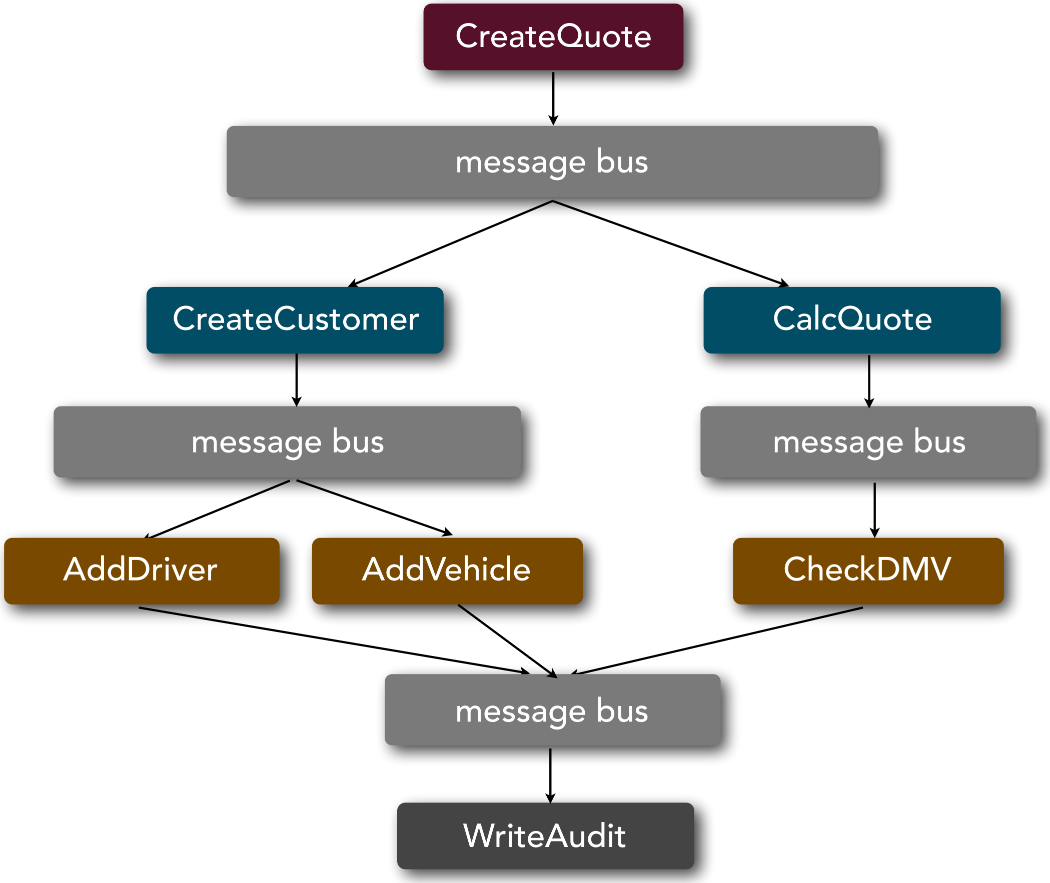
access decoupling

implementation transparency

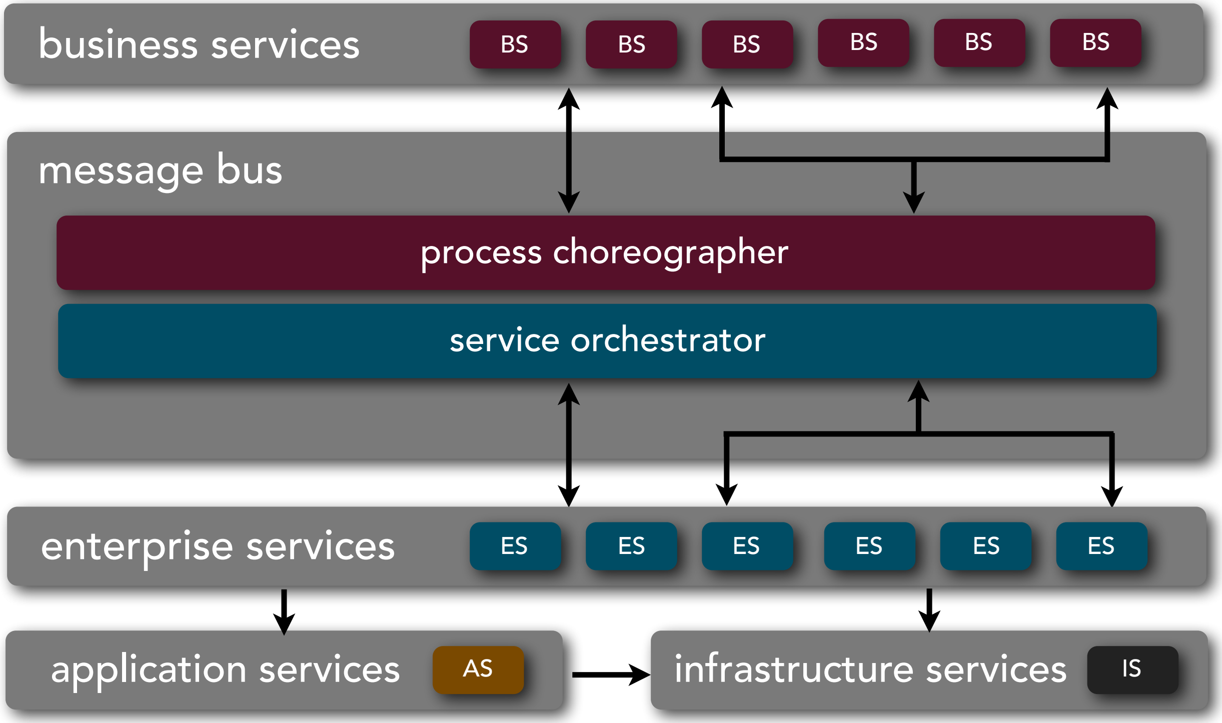
# service-oriented architecture



# service-oriented architecture

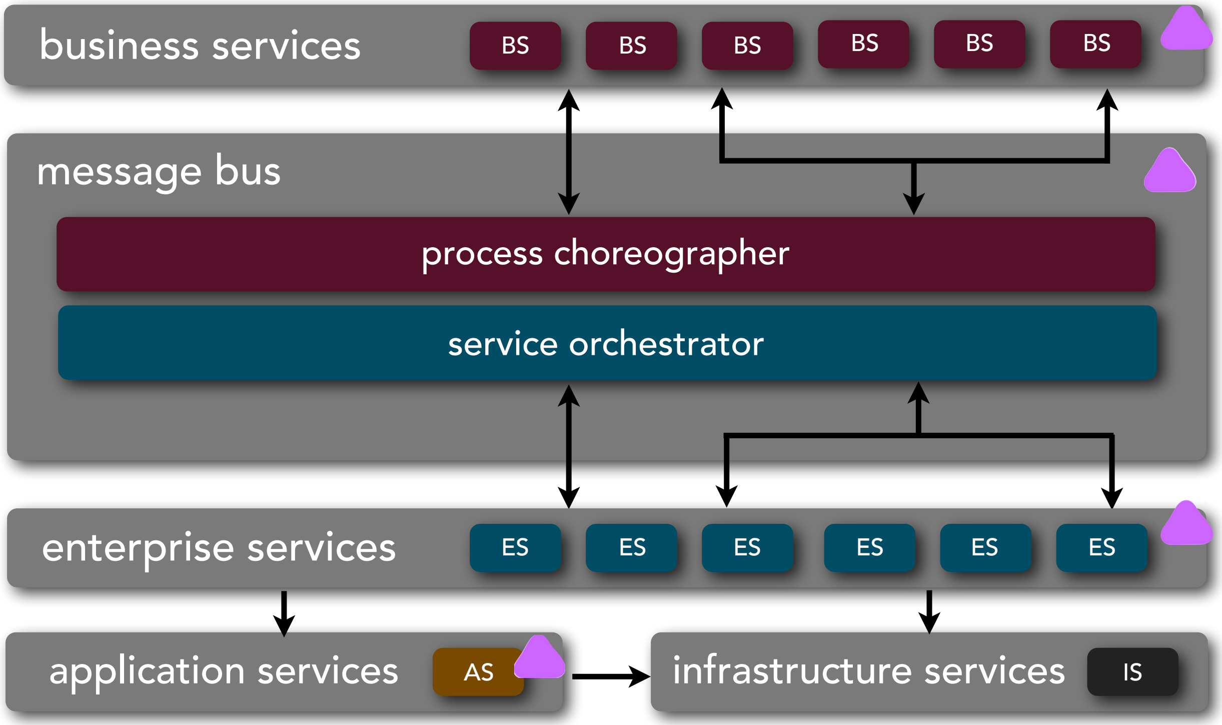


# service-oriented architecture



- maximize reuse
- maximize canonicality

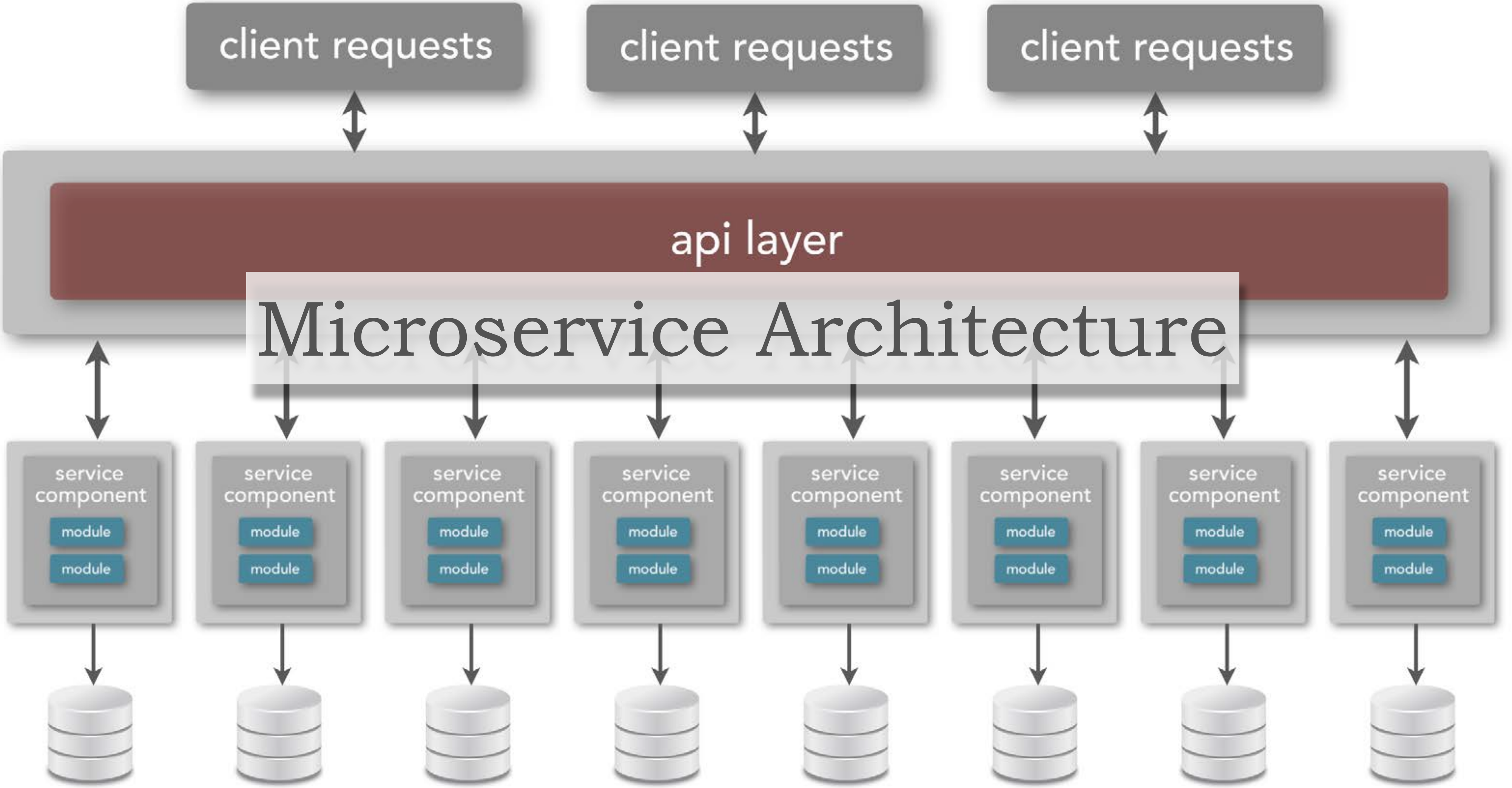
# service-oriented architecture



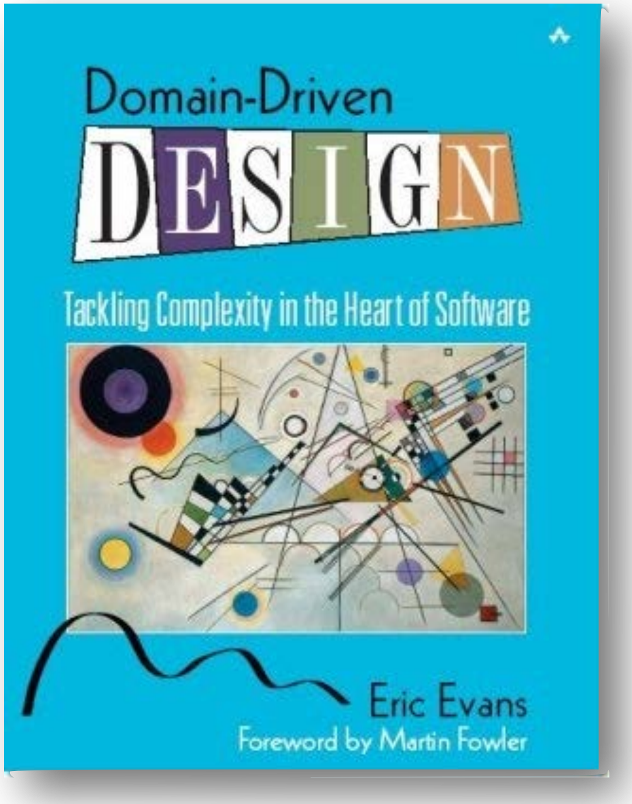
✖ incremental change

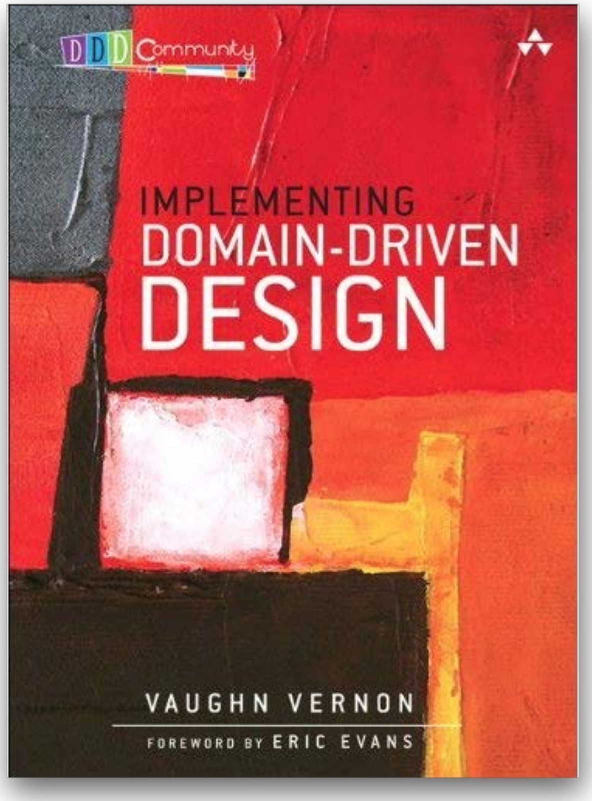
✖ operationally coupled

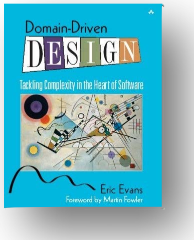
# Microservice Architecture



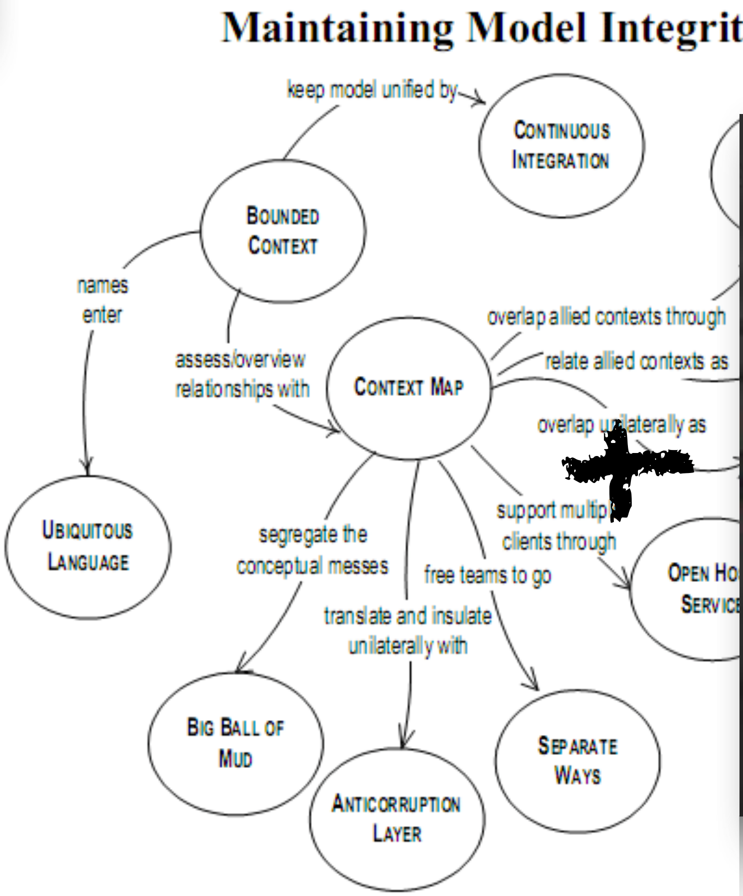
# domain driven design


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

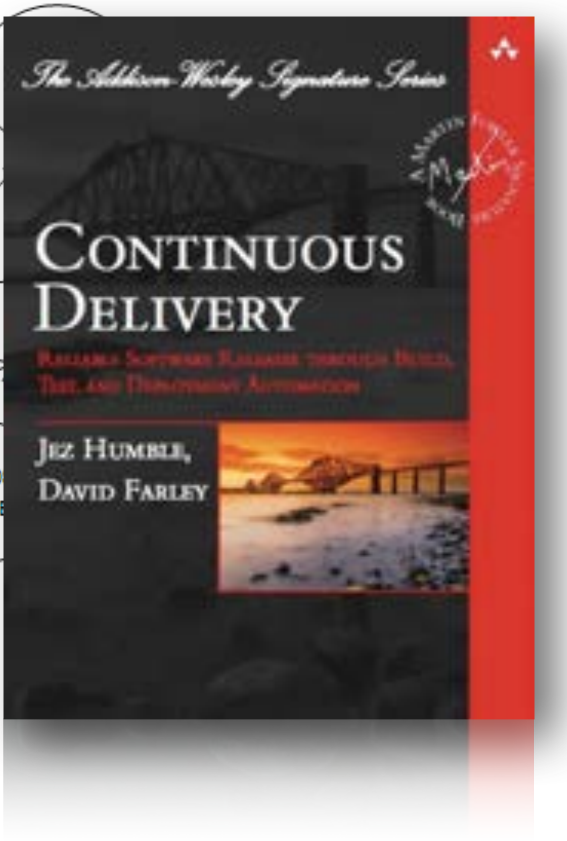



Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

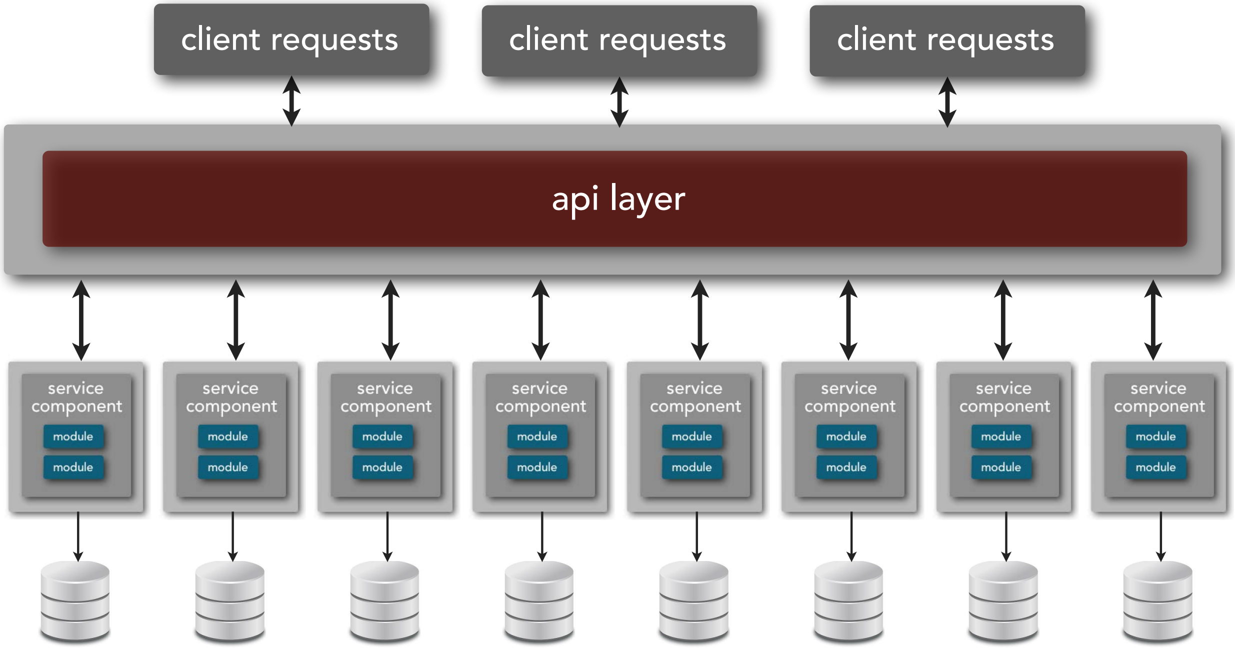
# bounded context


Maintaining Model Integrity
keep model unified by
CONTINUOUS INTEGRATION
BOUNDED CONTEXT
names enter
UBIQUITOUS LANGUAGE
assess/overview relationships with
CONTEXT MAP
overlap allied contexts through
relate allied contexts as
overlap unilaterally as
support multiple clients through
OPEN HO
SERVICE
segregate the conceptual messes
free teams to go
translate and insulate unilaterally with
BIG BALL OF MUD
ANTICORRUPTION LAYER
SEPARATE WAYS

+

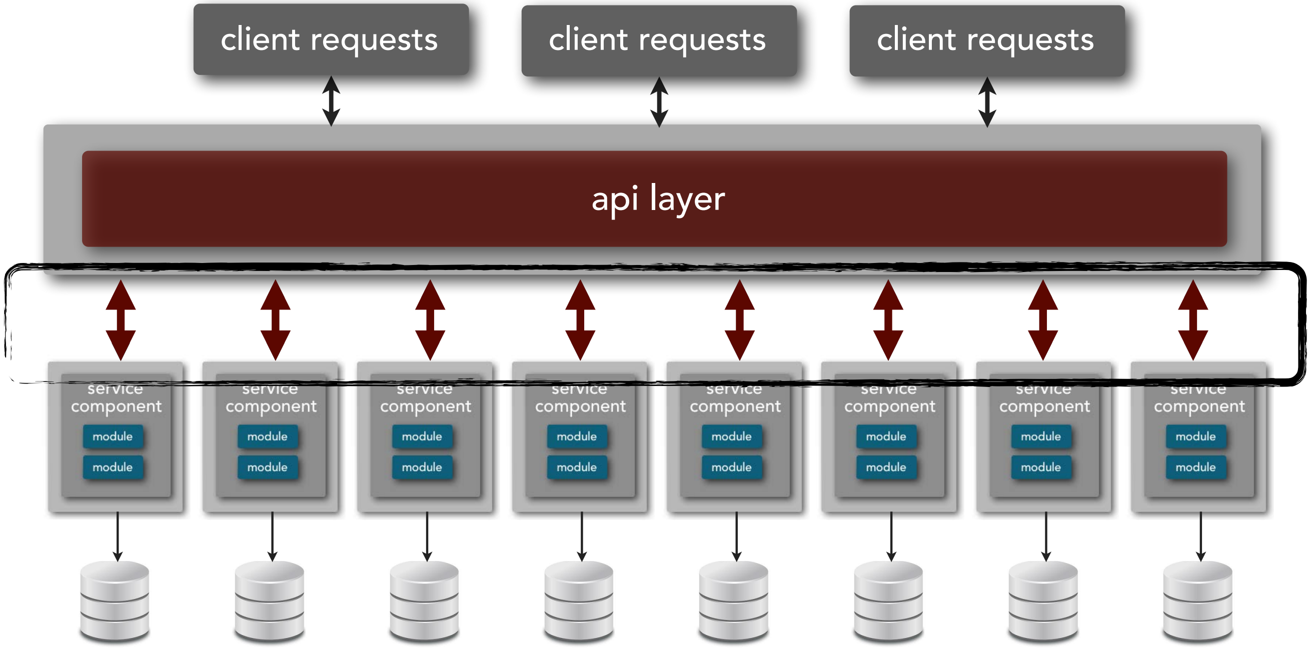

The Addison Wesley Signature Series
A Martin Fowler Signature Book
CONTINUOUS DELIVERY
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble, David Farley

# microservices architecture



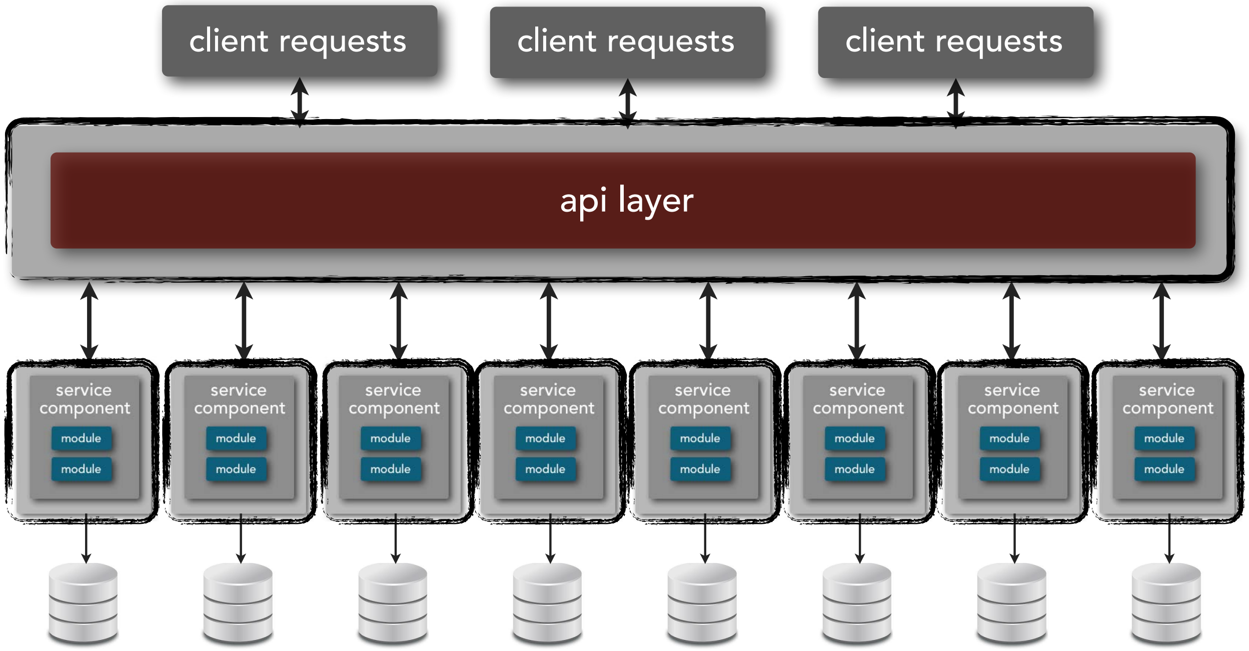
# microservices architecture

## distributed architecture



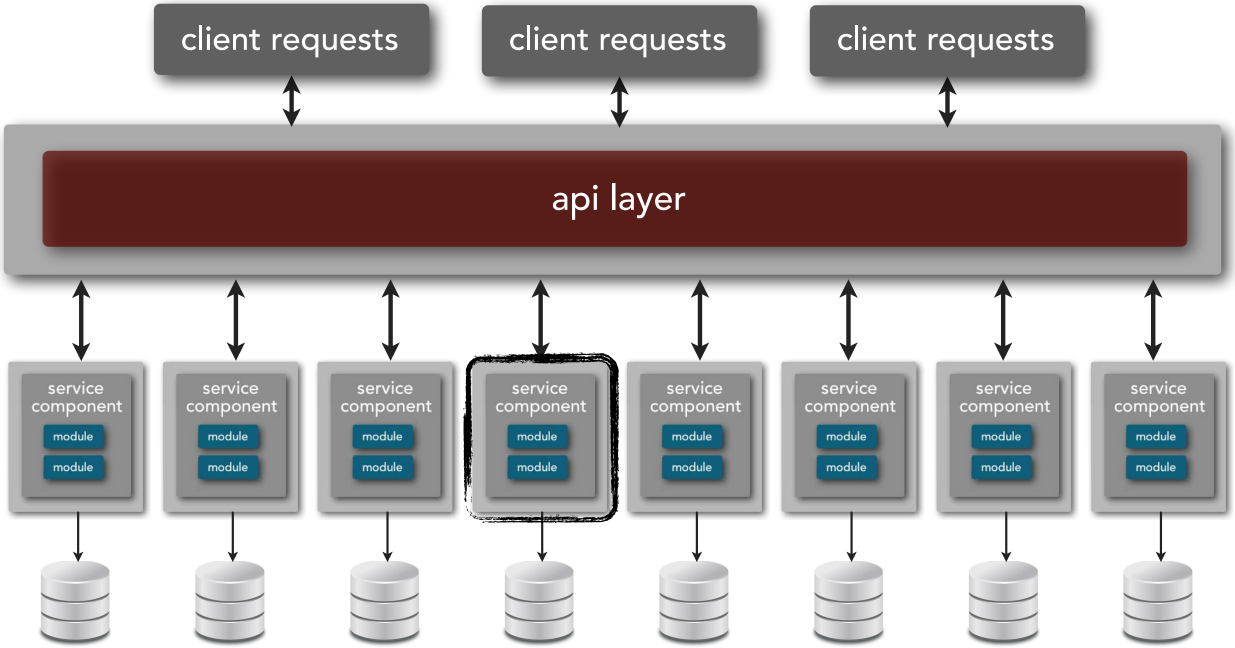
# microservices architecture

## separately deployed components



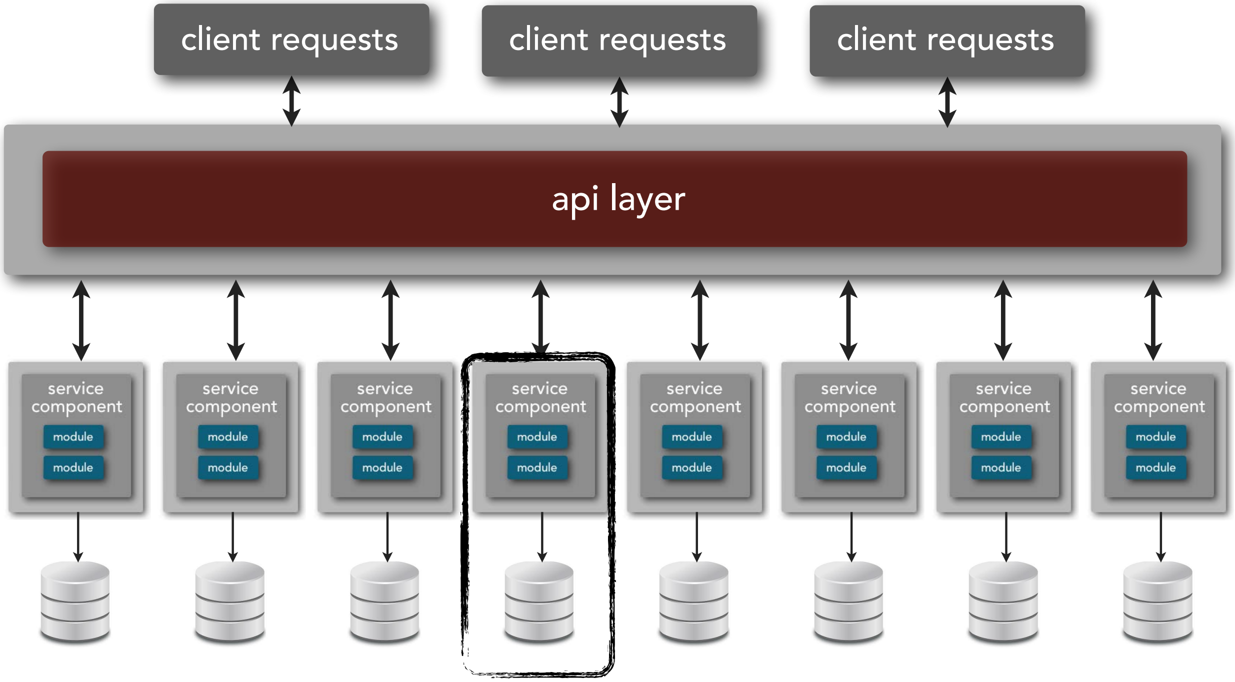
# microservices architecture

## service component

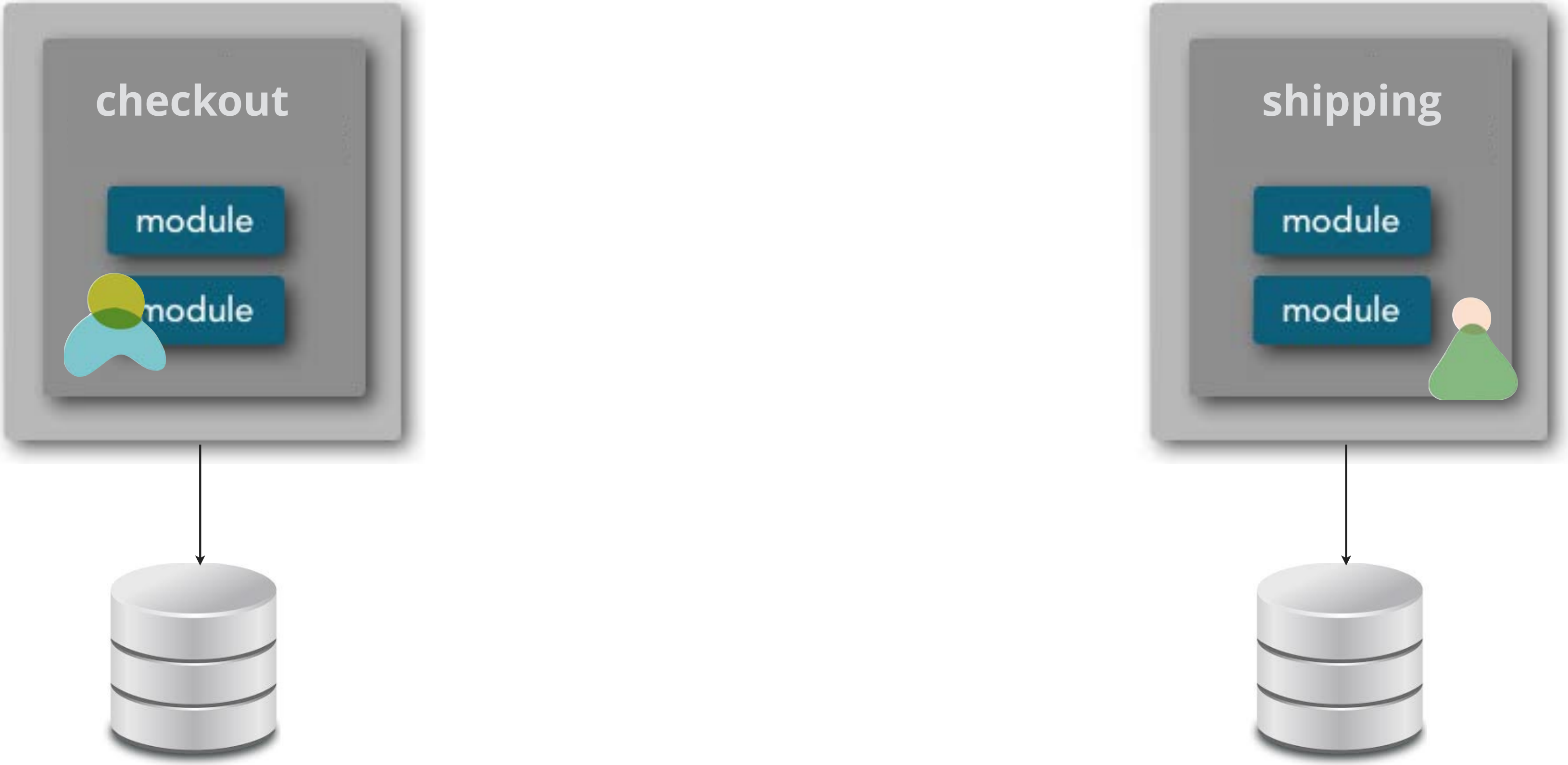


# microservices architecture

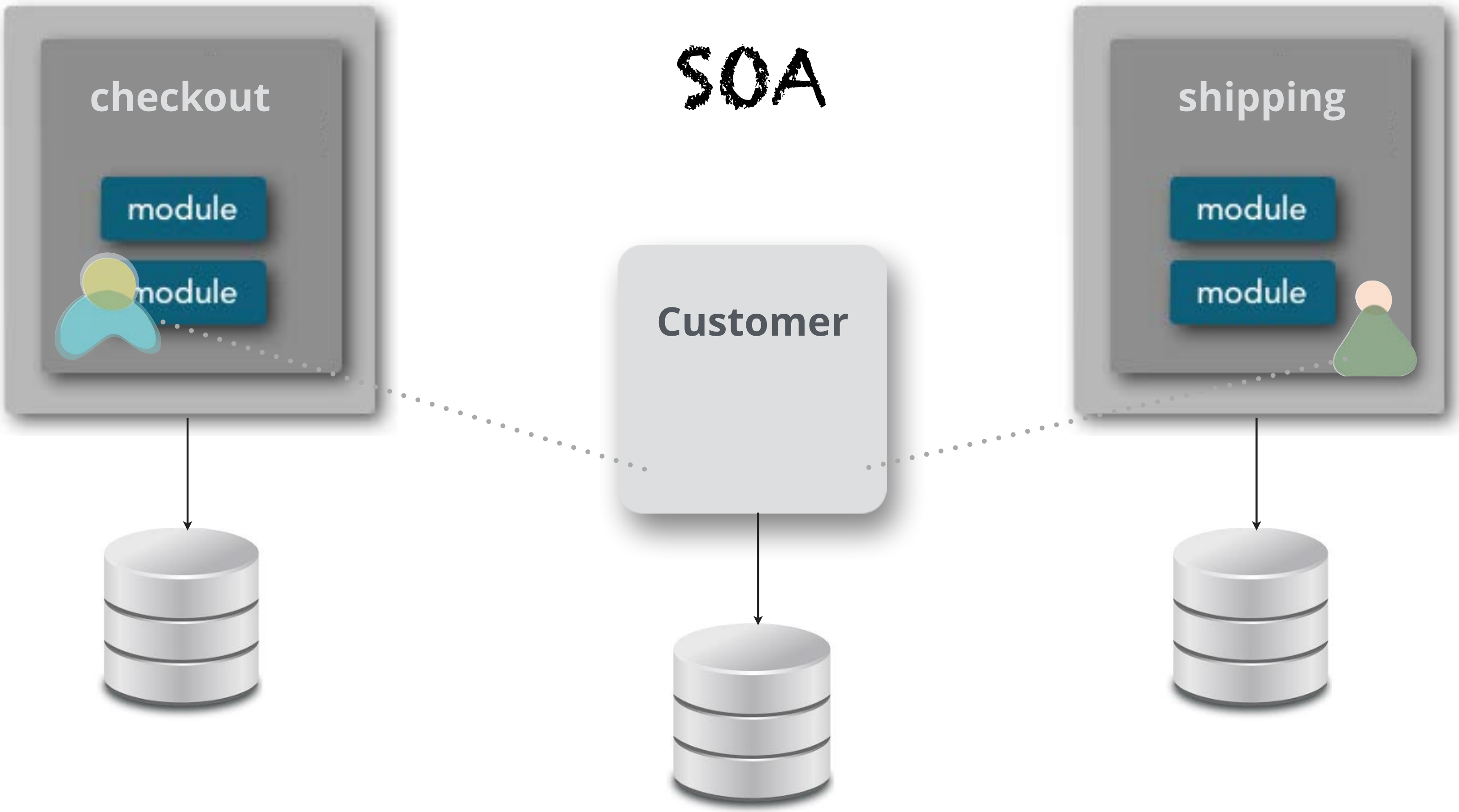
## bounded context



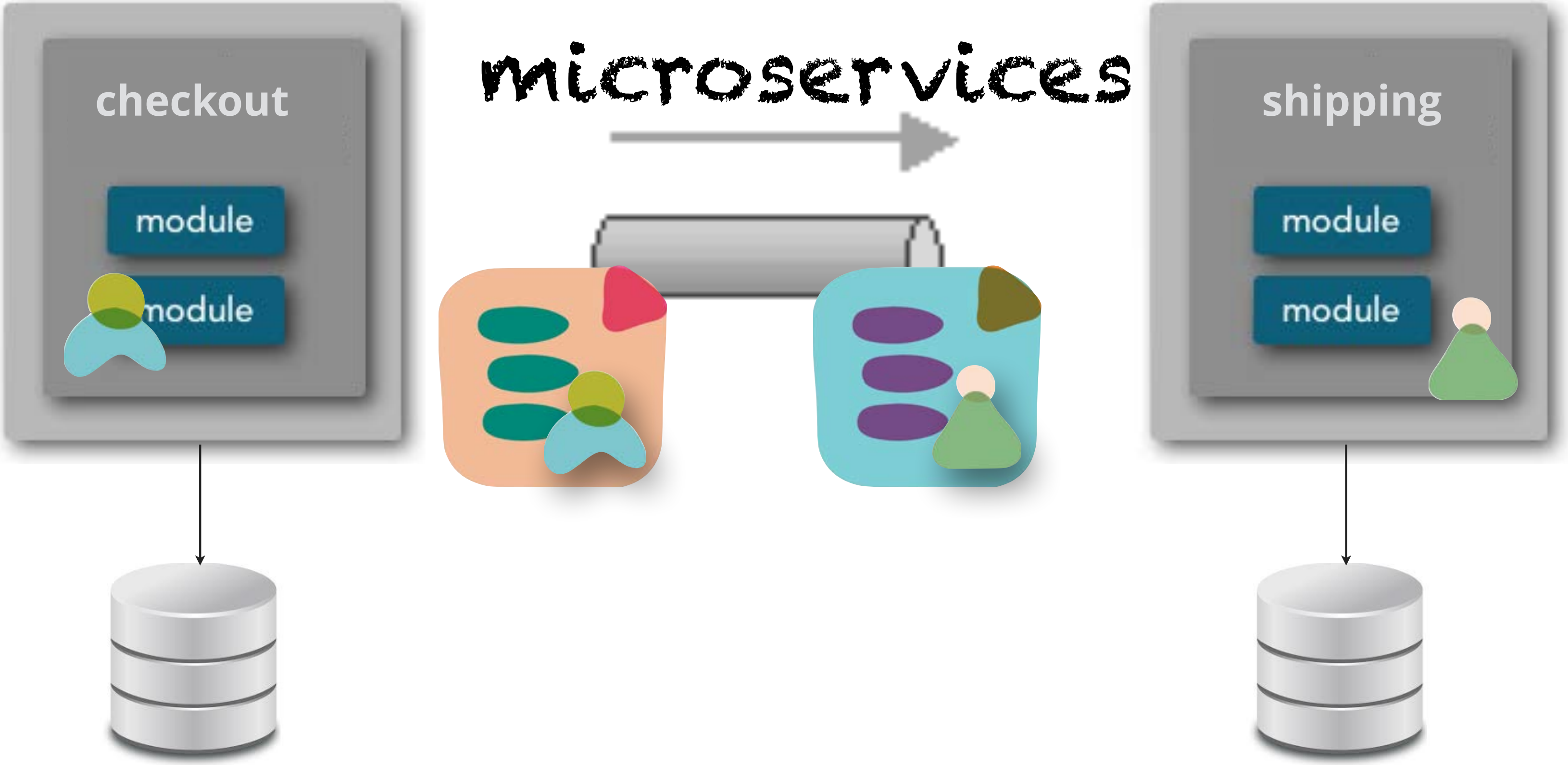
# bounded context ≠ entity



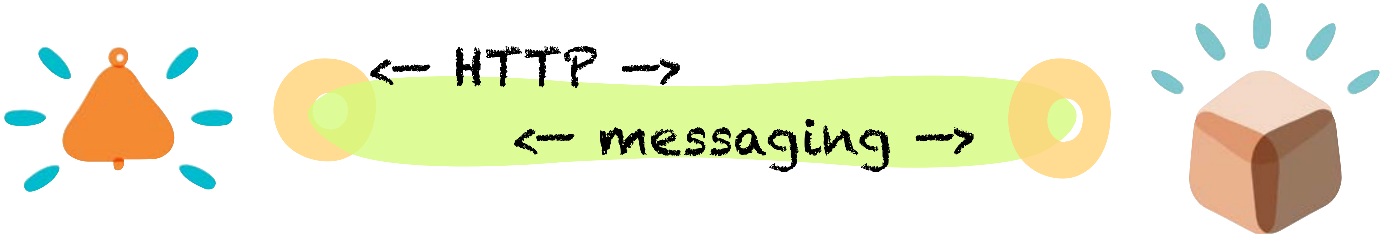
# prefer duplication over coupling

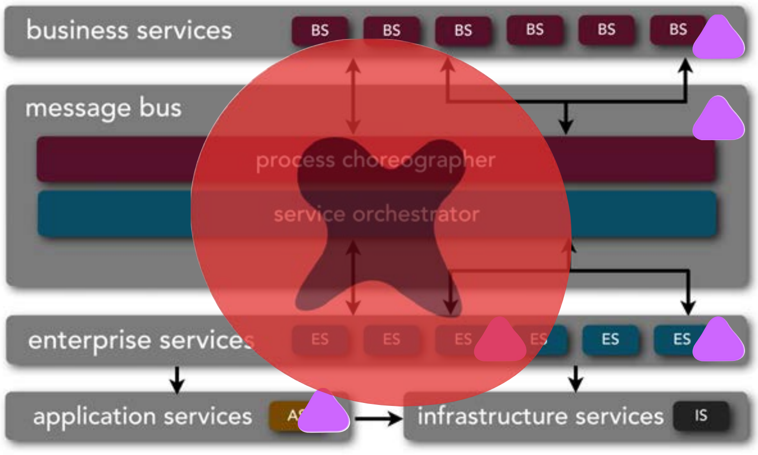


# prefer duplication over coupling



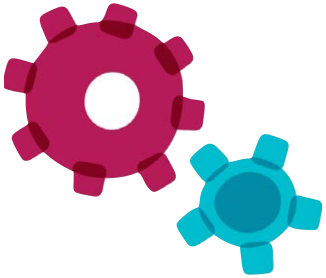
# smart endpoints, dumb pipes

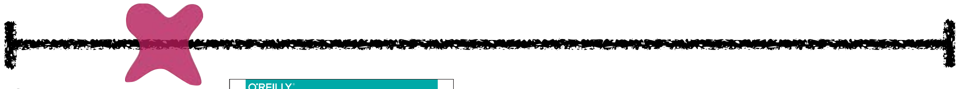



business services
BS
BS
BS
BS
BS
BS
message bus
process choreographer
service orchestrator
enterprise services
ES
ES
ES
ES
ES
ES
application services
AS
infrastructure services
IS

# standardize on integration, not platform

## embrace polyglot solutions where sensible





too few languages/platforms

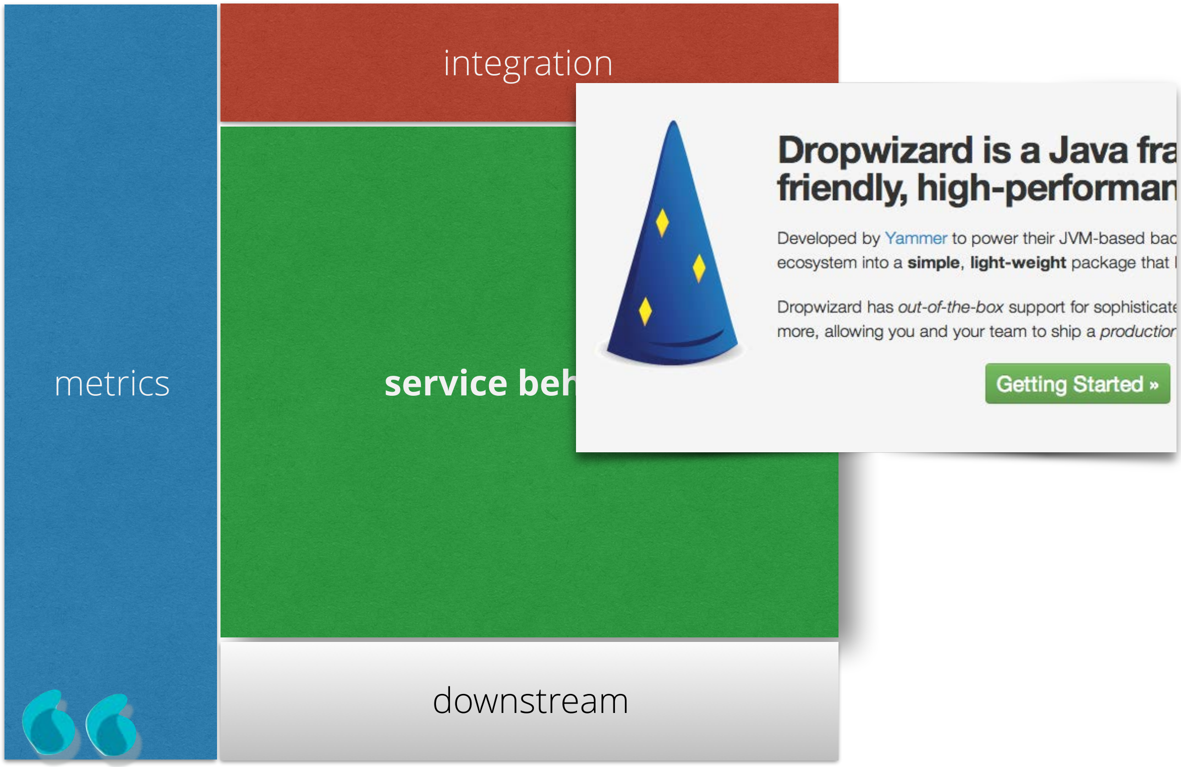
too many languages/platforms

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

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

*Pick some sensible conventions, and stick with them.*

# technical consistency



integration

metrics

service beh

downstream

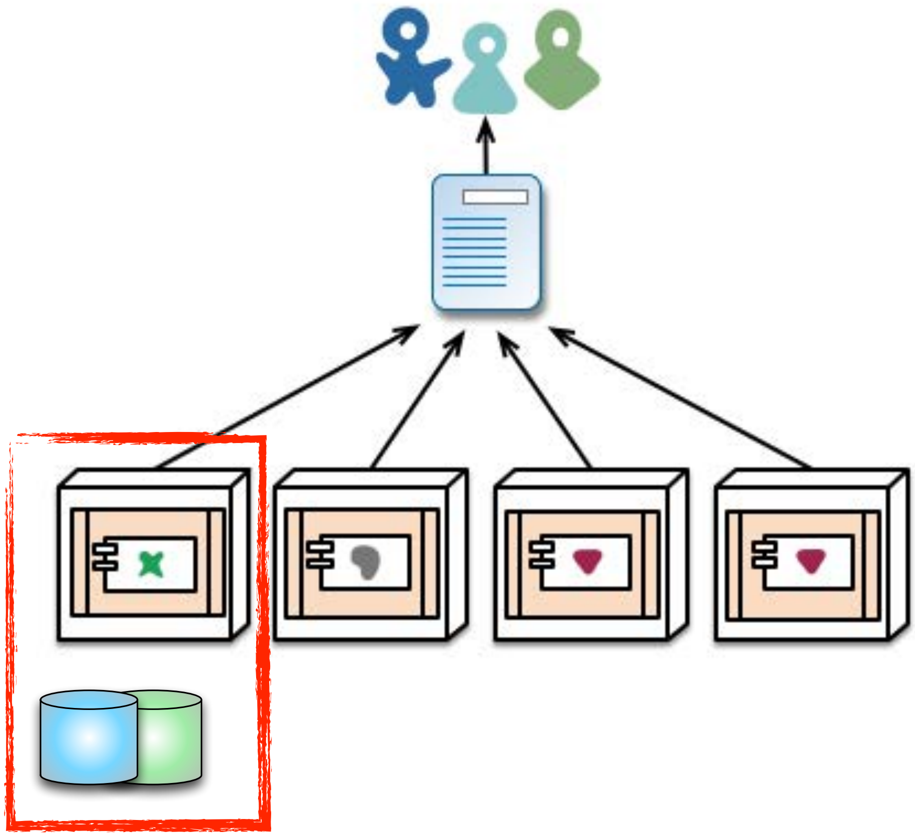
**Dropwizard is a Java fra friendly, high-performan**

Developed by Yammer to power their JVM-based bac ecosystem into a **simple**, **light-weight** package that l

Dropwizard has *out-of-the-box* support for sophisticate more, allowing you and your team to ship a *productior*

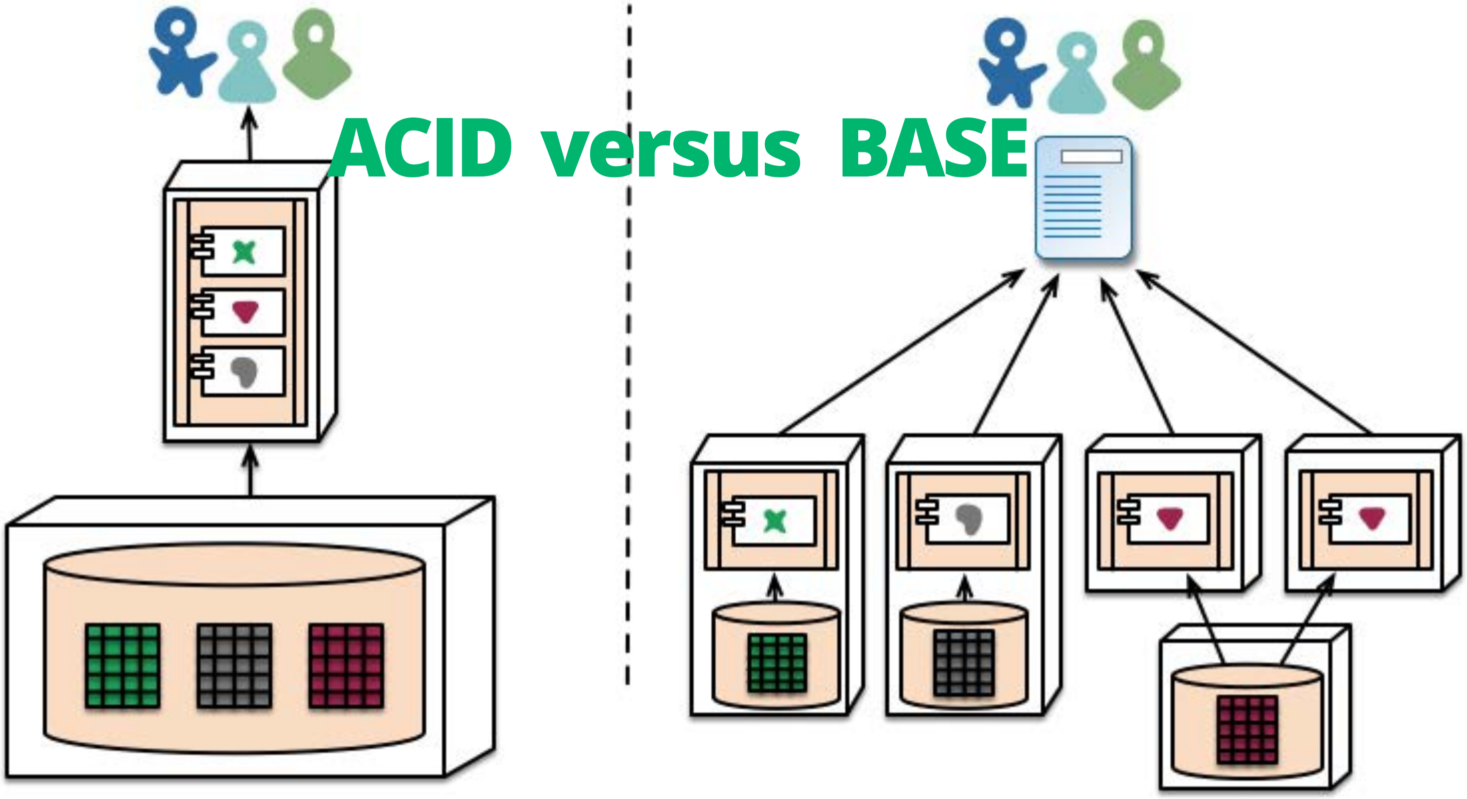
*Consider Service Templates to make it easy to do the right thing!*

# decentralized governance

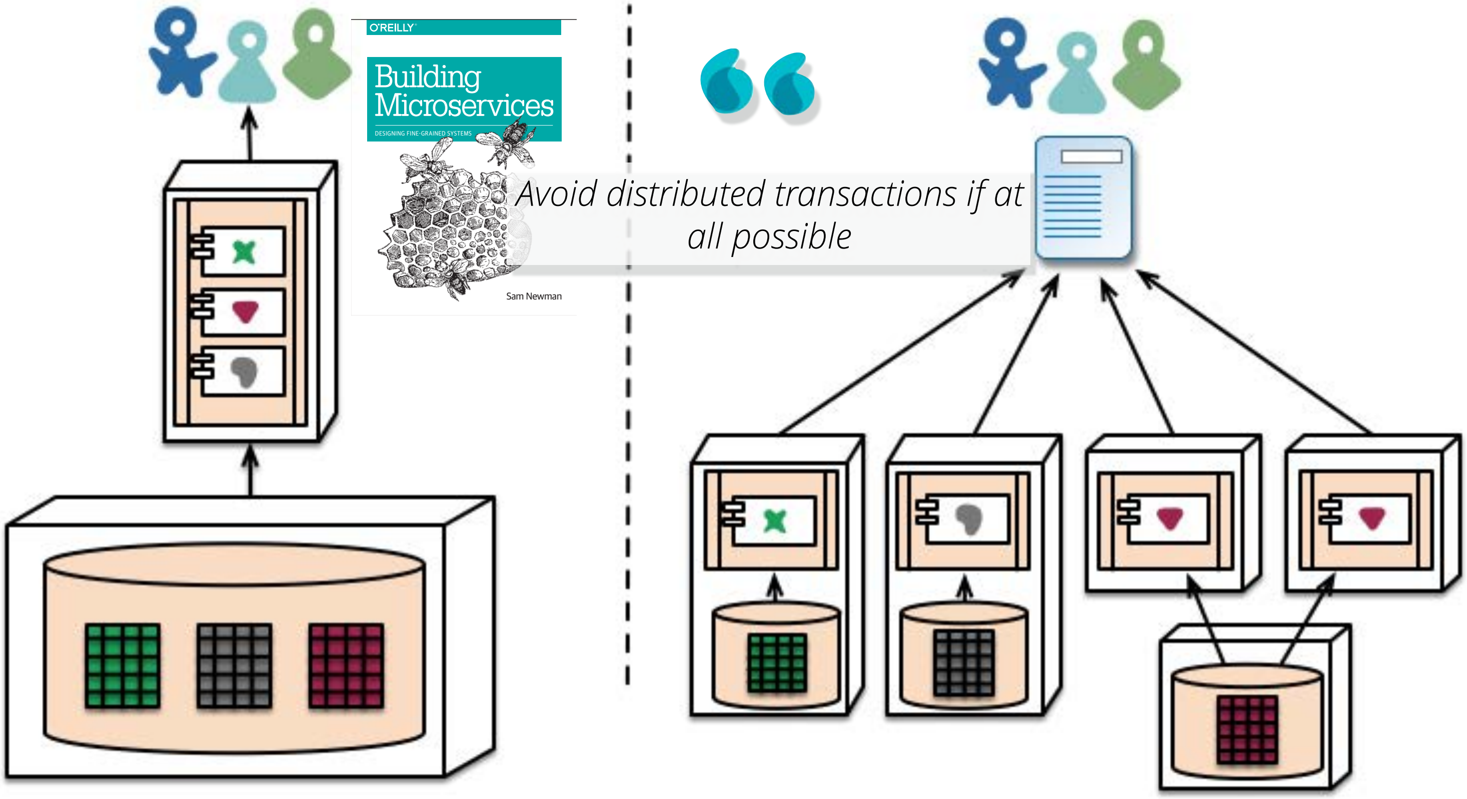


# decentralized data management

## ACID versus BASE


monolith - single database
microservices - application databases

# decentralized data management

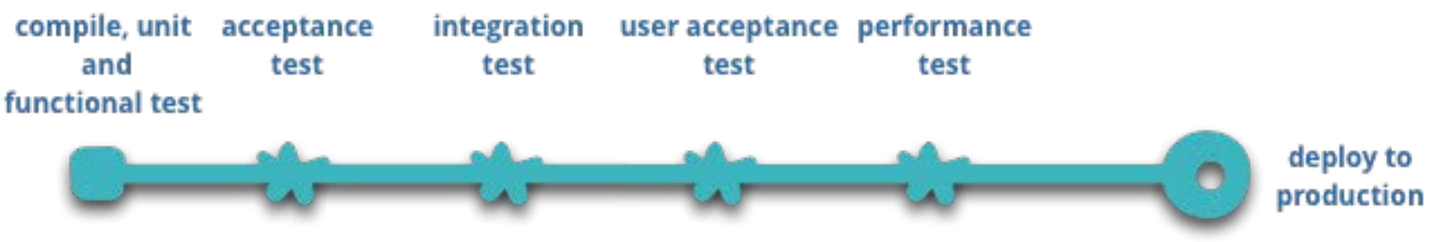


*Avoid distributed transactions if at all possible*

monolith - single database

microservices - application databases

# infrastructure automation



compile, unit and functional test

acceptance test

integration test

user acceptance test

performance test

deploy to production

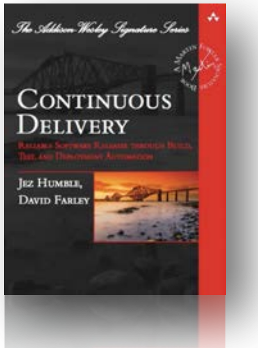
run on build machine

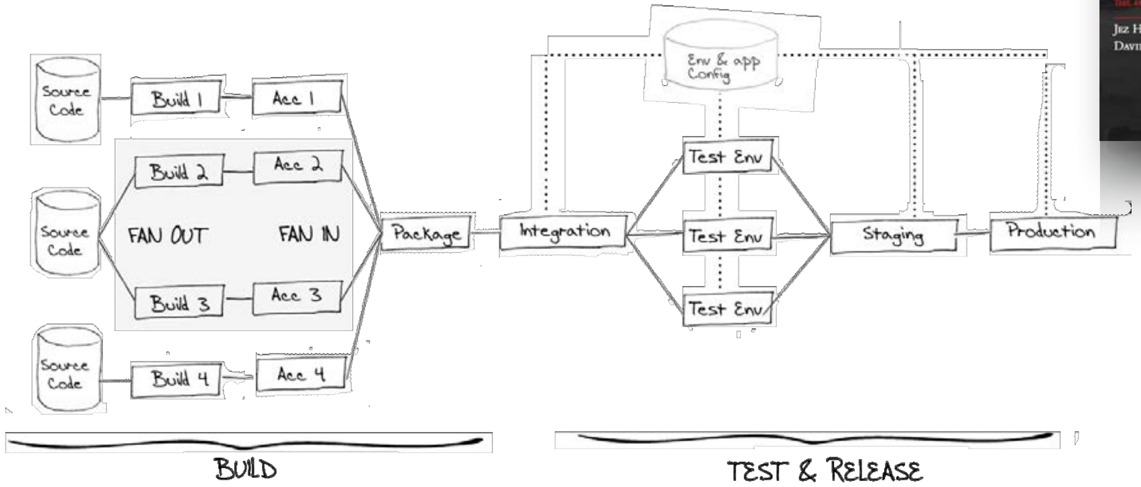
deployed on build machine

deployed to integration environment

deployed to UAT environment

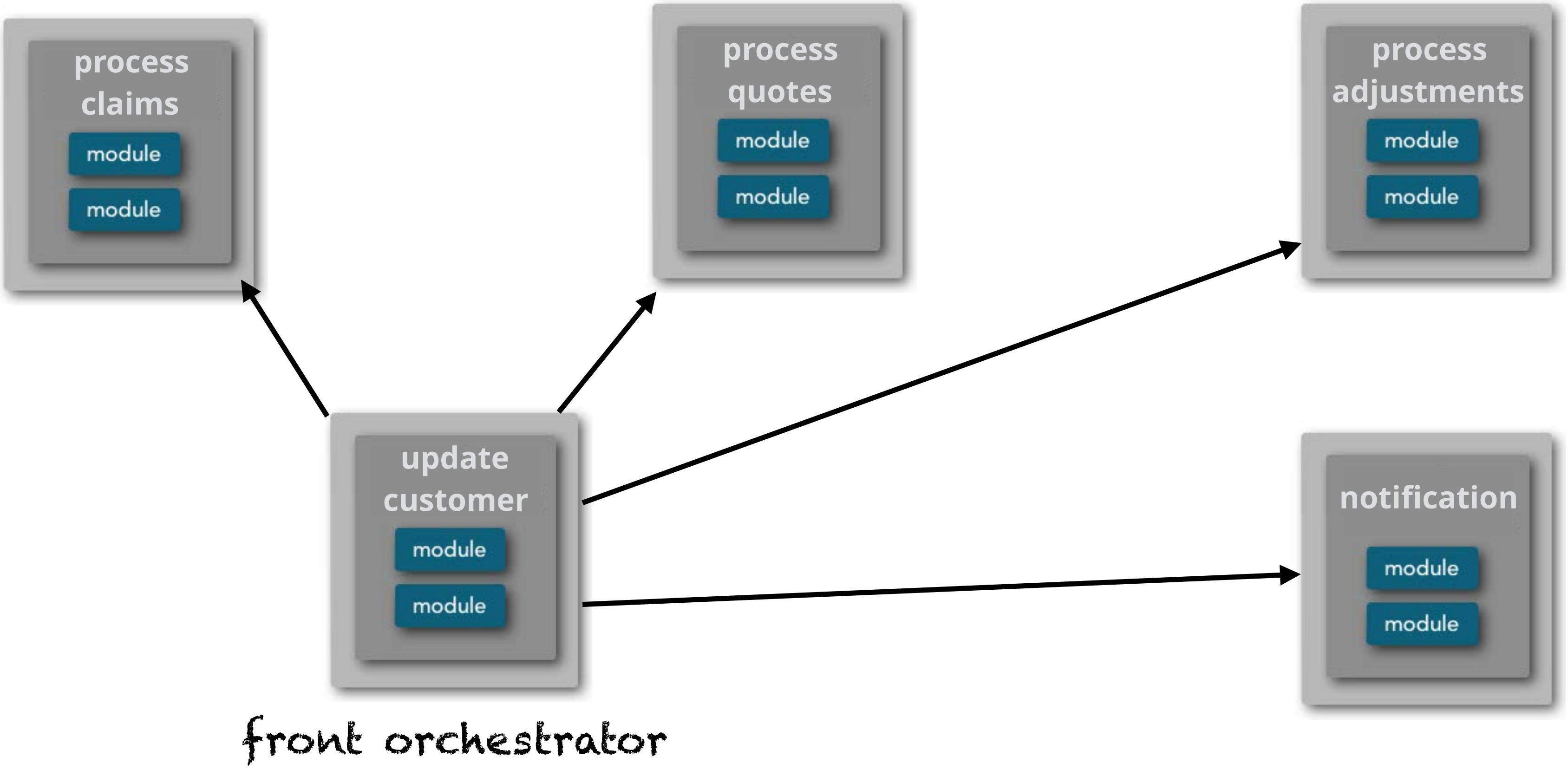
deployed to performance environment


The Addison-Wesley Signature Series
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble, David Farley



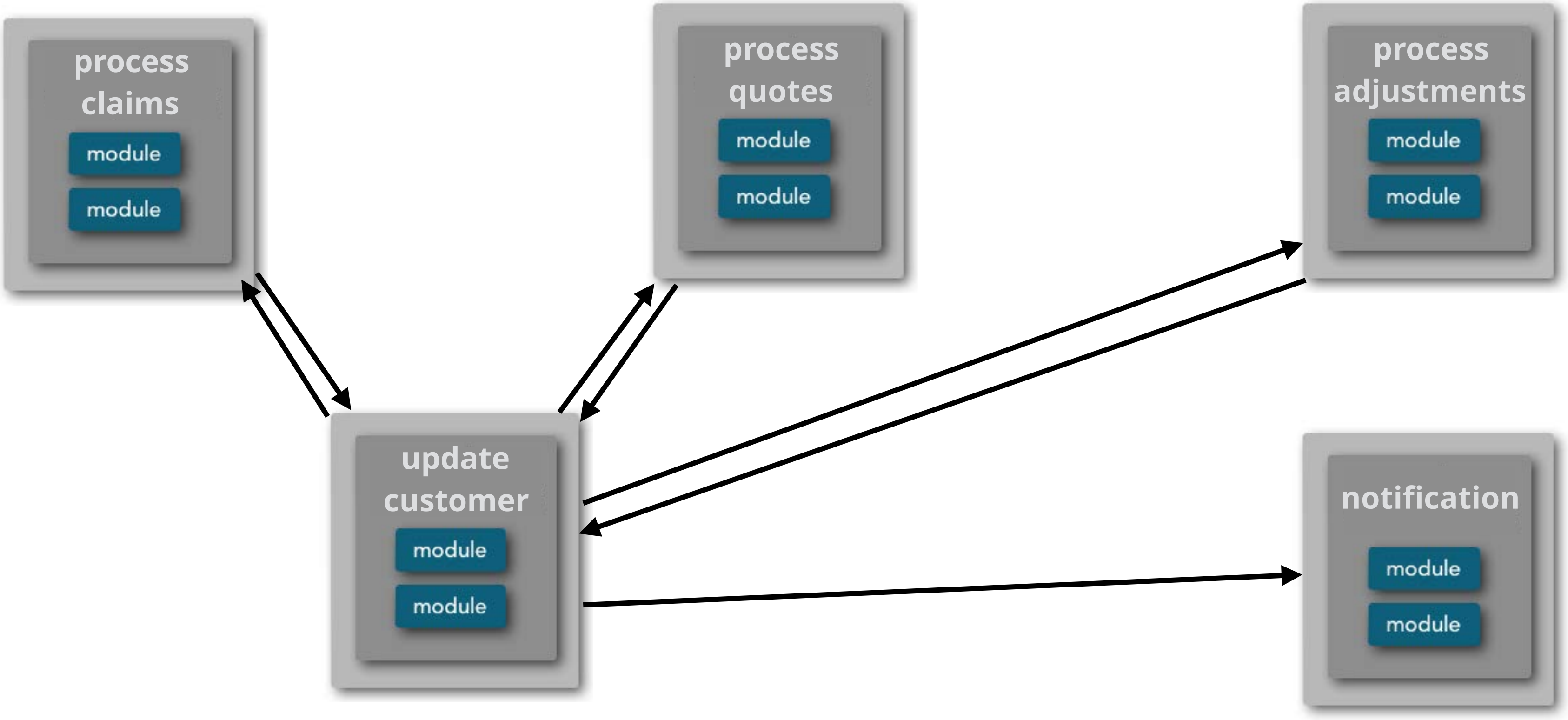
# microservices architecture

## service orchestration



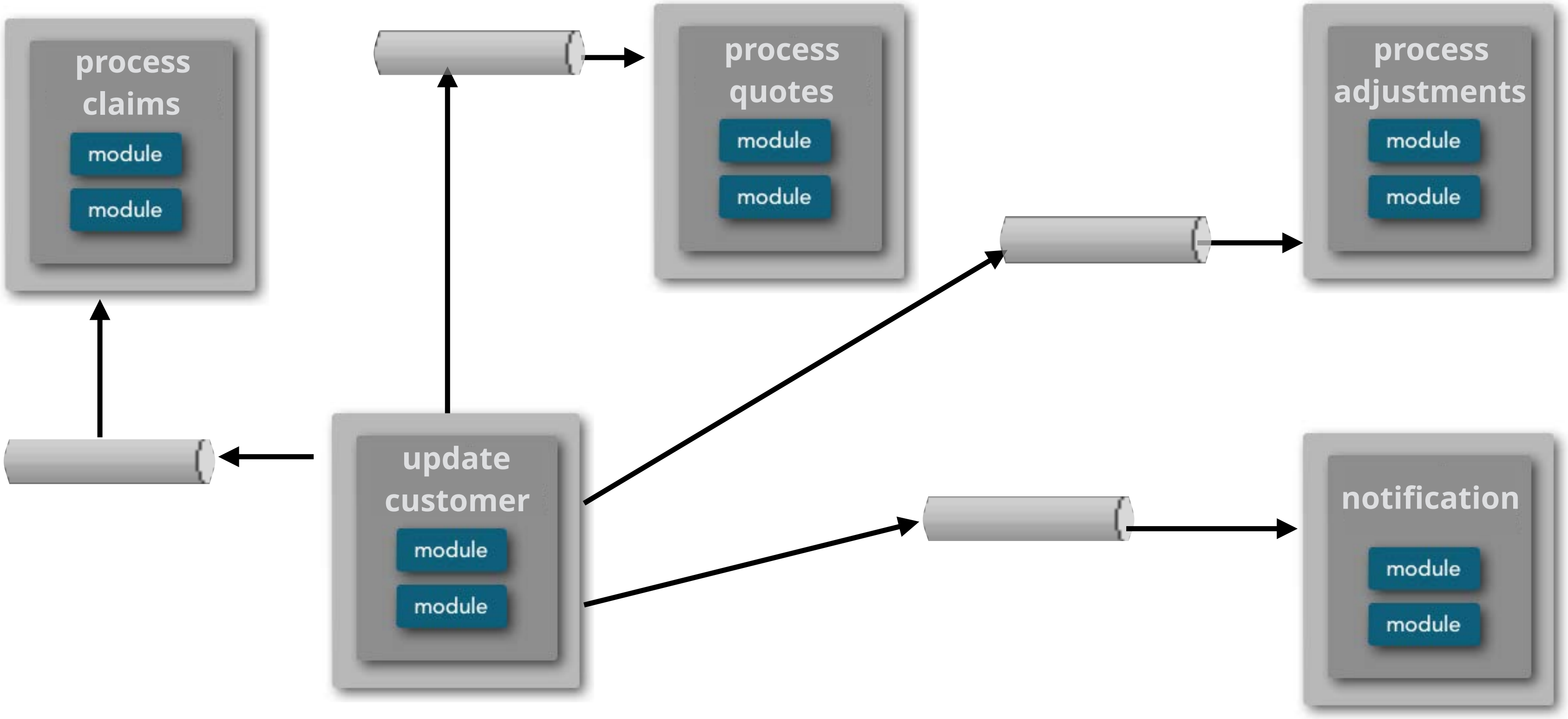
# microservices architecture

## service orchestration



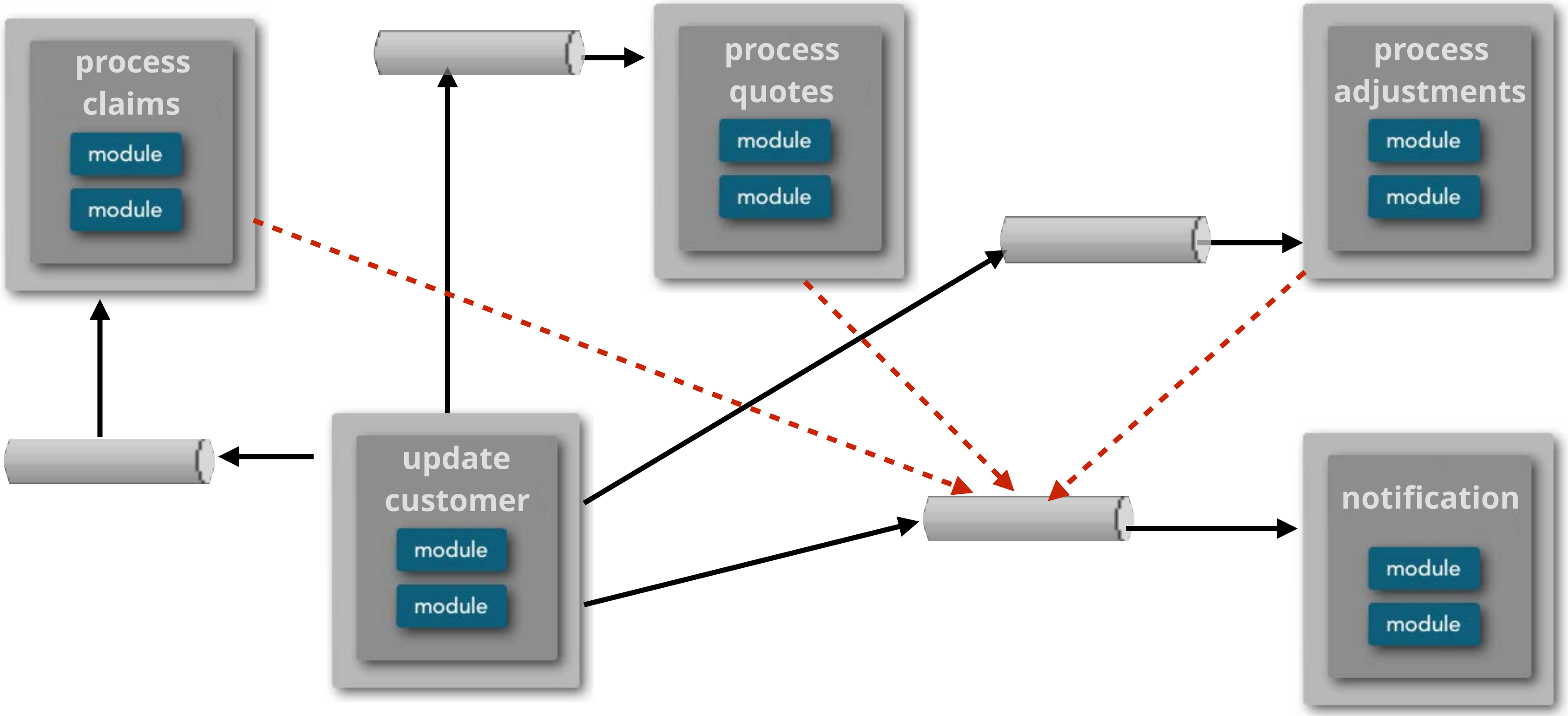
# microservices architecture

## service orchestration



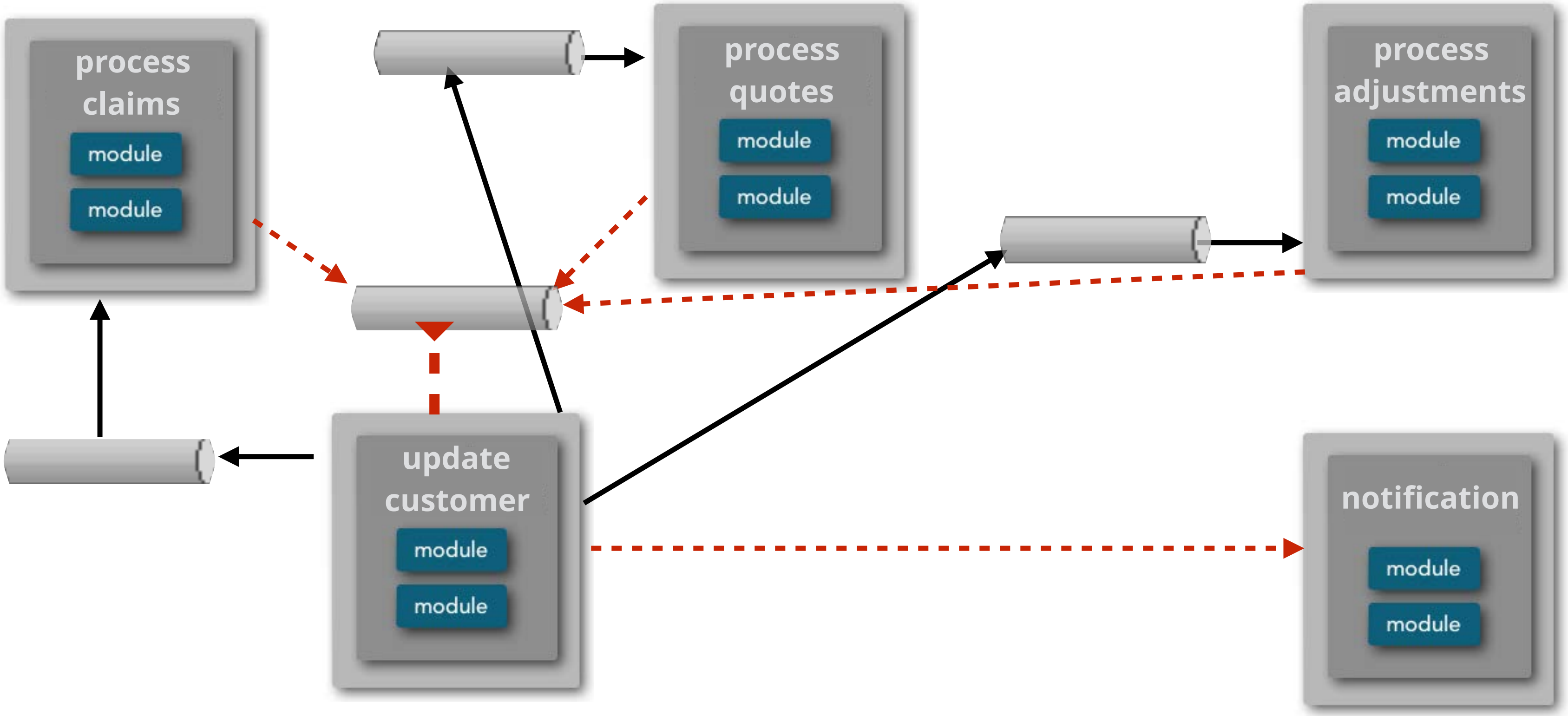
# microservices architecture

## service orchestration



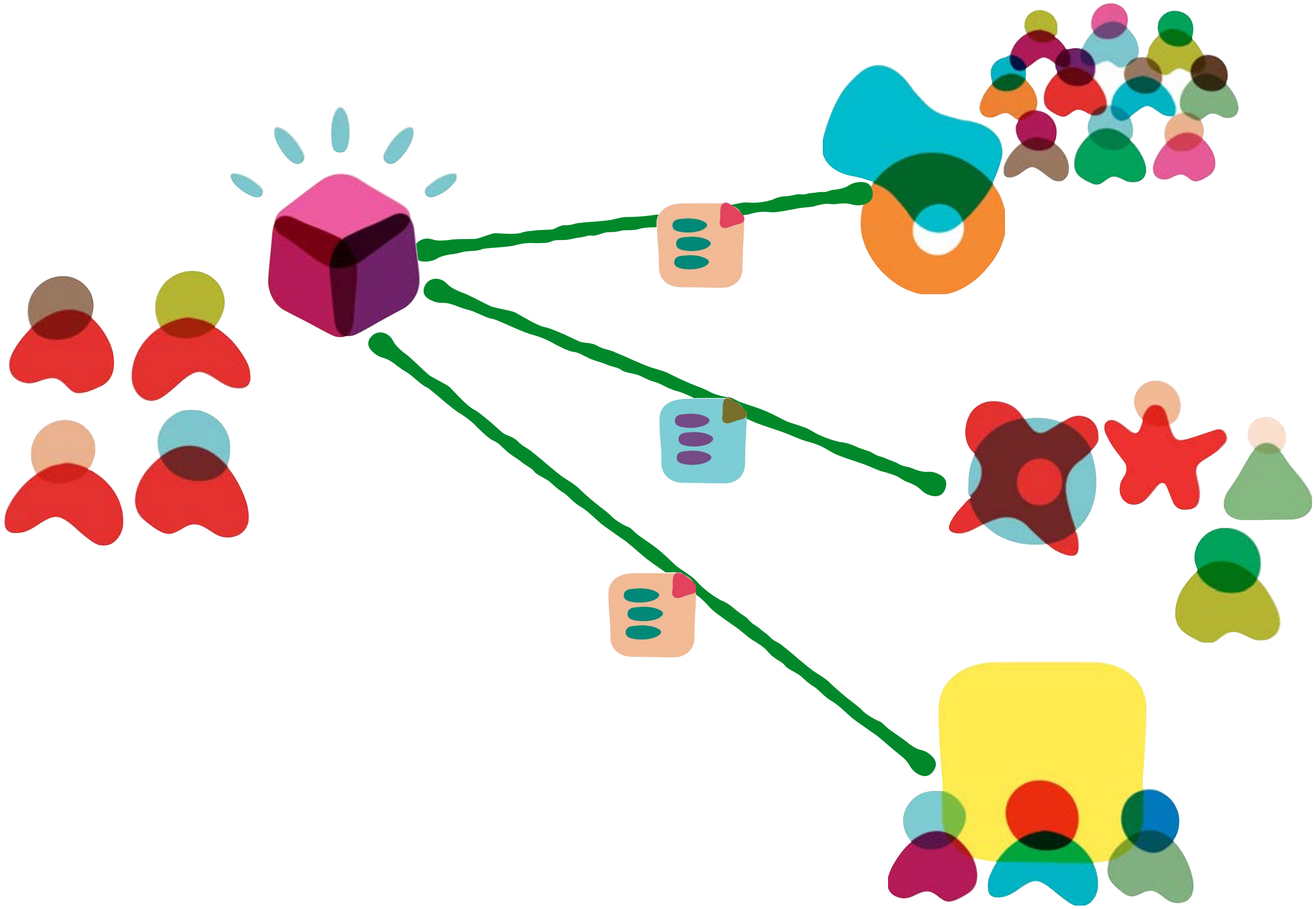
# microservices architecture

## service orchestration

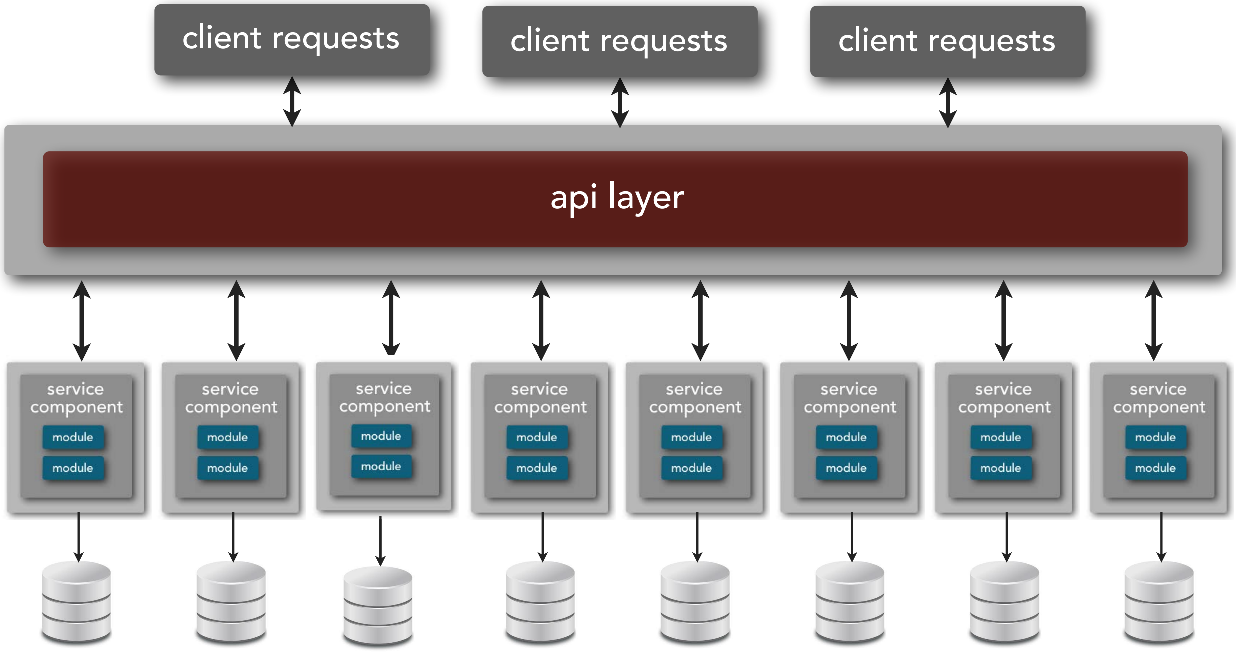


# consumer driven contracts

http://martinfowler.com/articles/consumerDrivenContracts.html



# microservices architecture

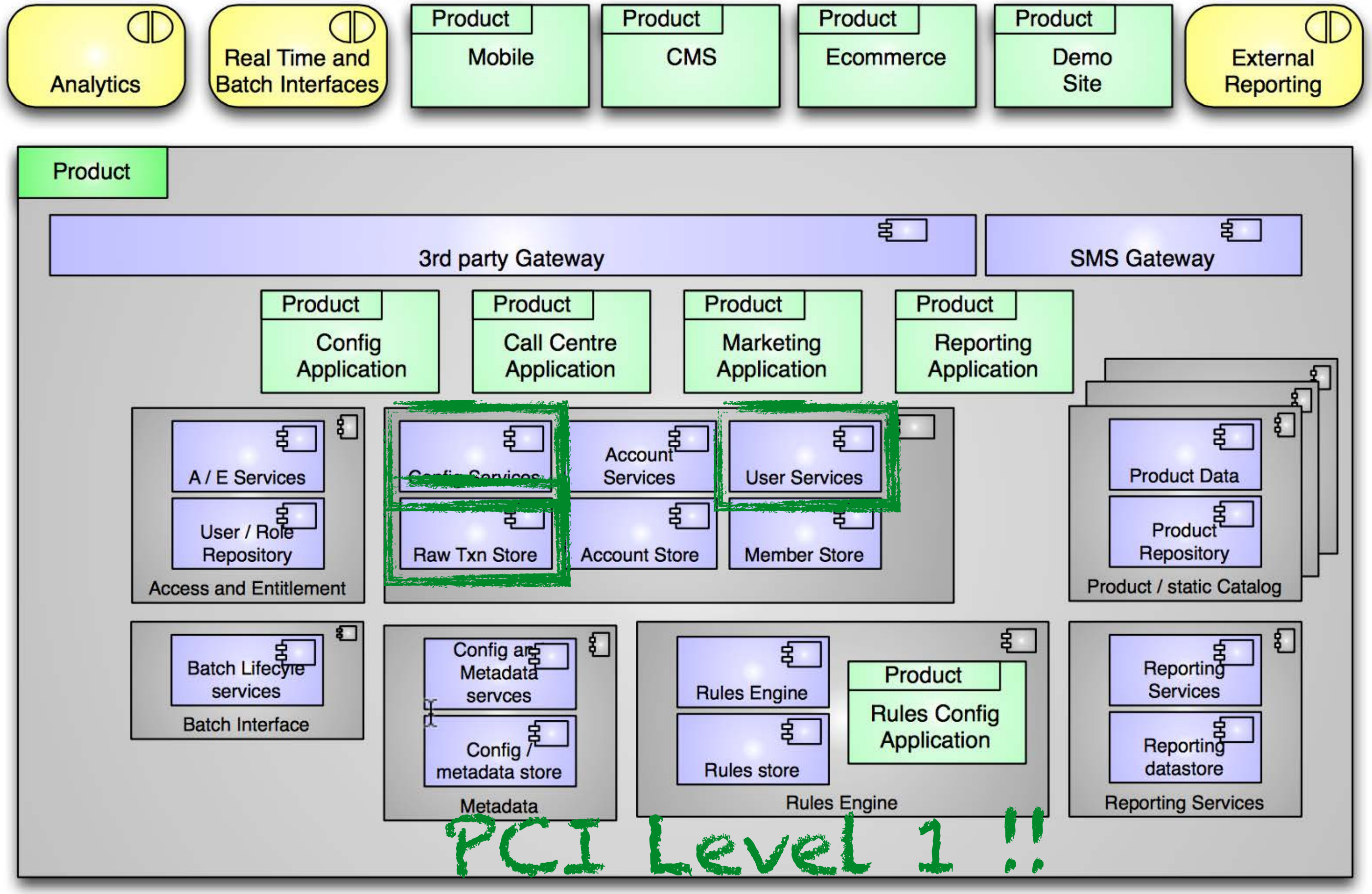


maximize easy evolution

# support △

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

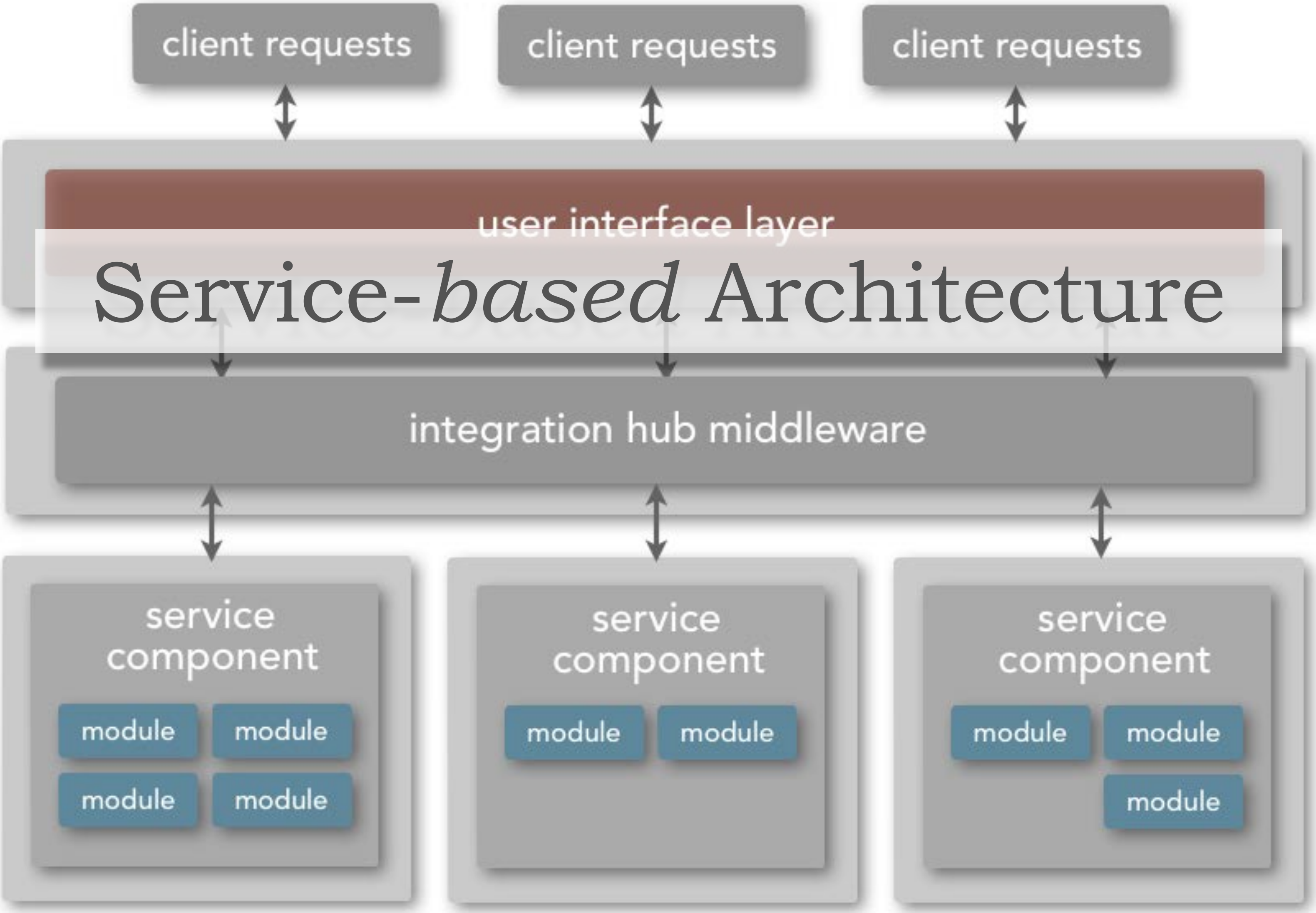
# microservice implementation



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

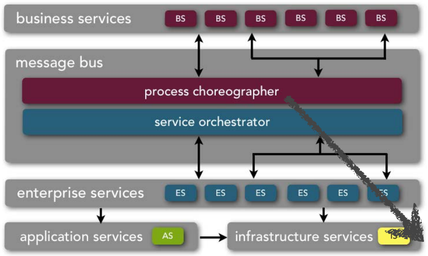
http://www.infoq.com/presentations/Micro-Services

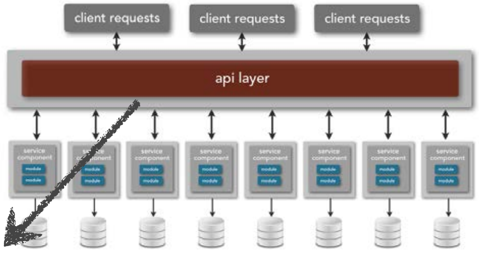
# Service-*based* Architecture



# service-based architecture

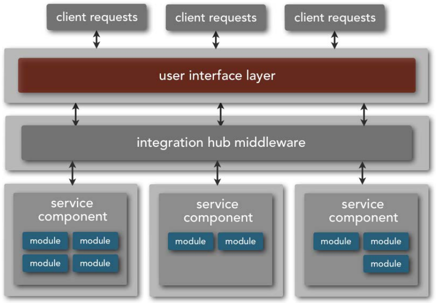
## is there a middle ground?





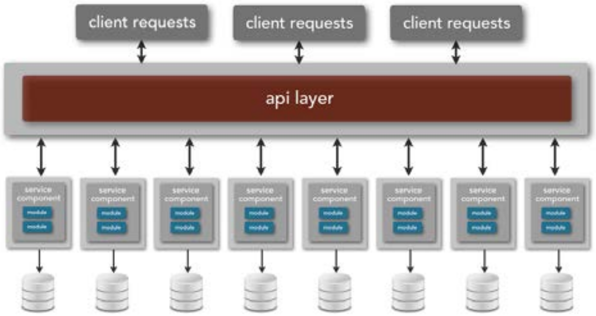
service-orie
architectu

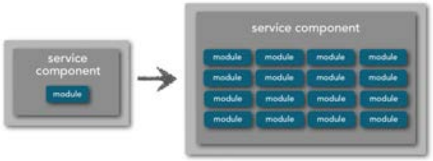
icroservices
rchitecture



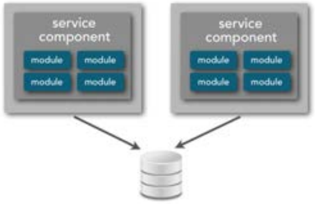
service-based
architecture

# service-based architecture

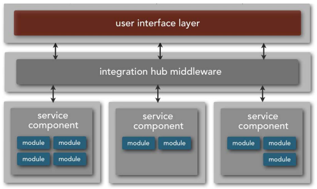




service granularity



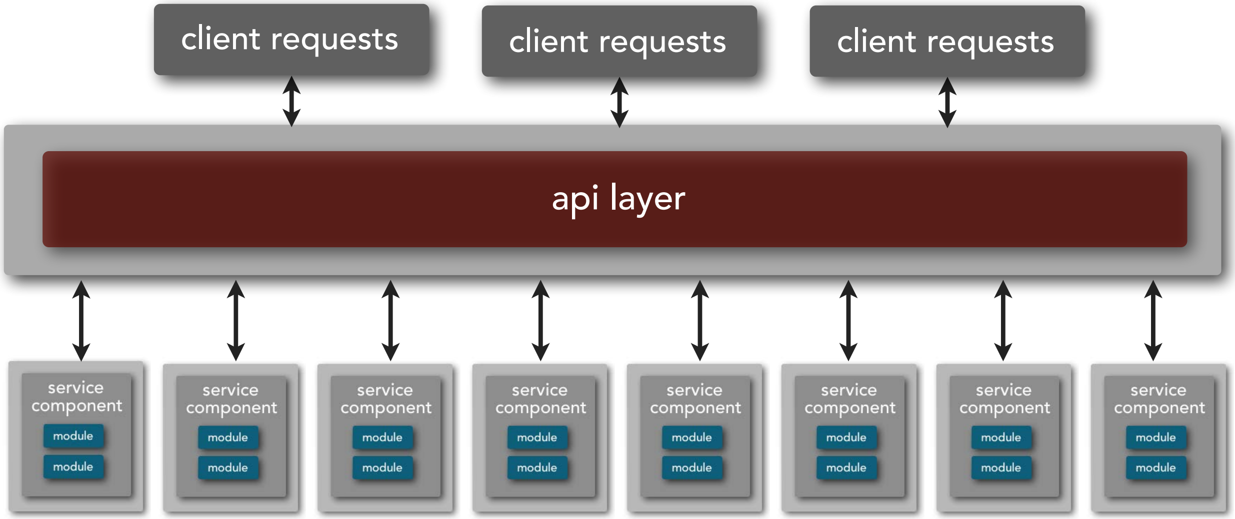
database scope



integration hub

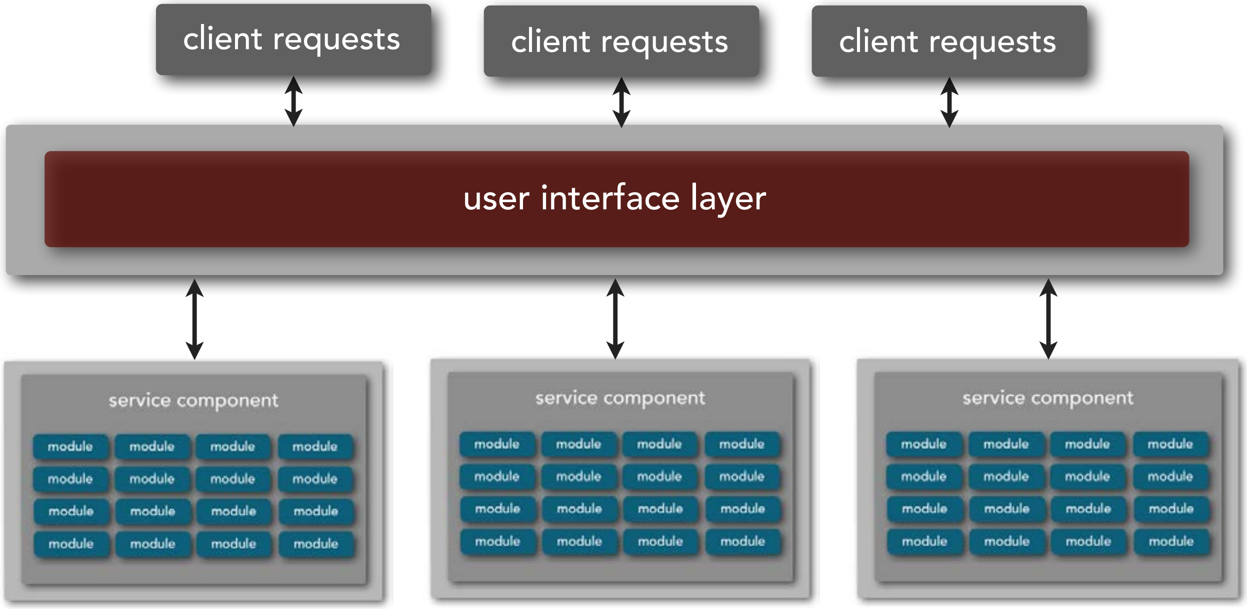
# service-based architecture

## service granularity



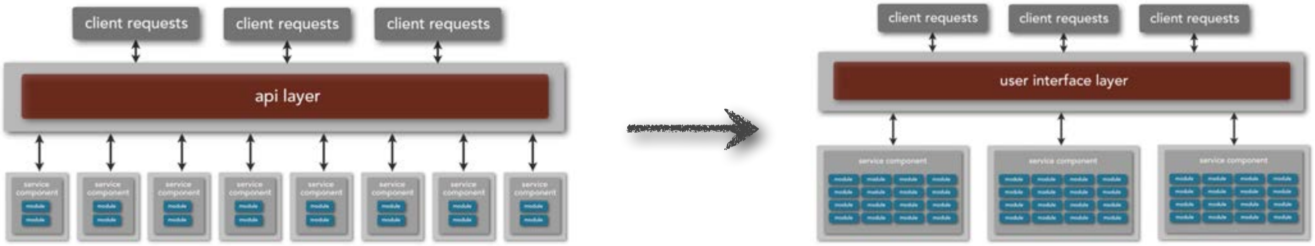
# service-based architecture

## service granularity



# service-based architecture

## service granularity

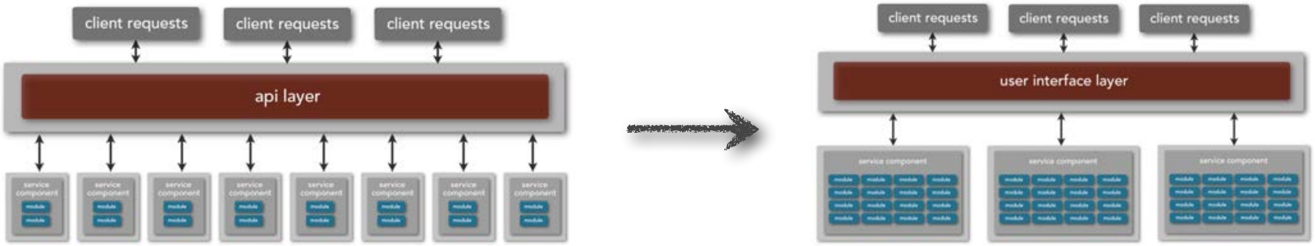


single-purpose micro-service to "portion of the application" macro-service

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

# service-based architecture

## service granularity



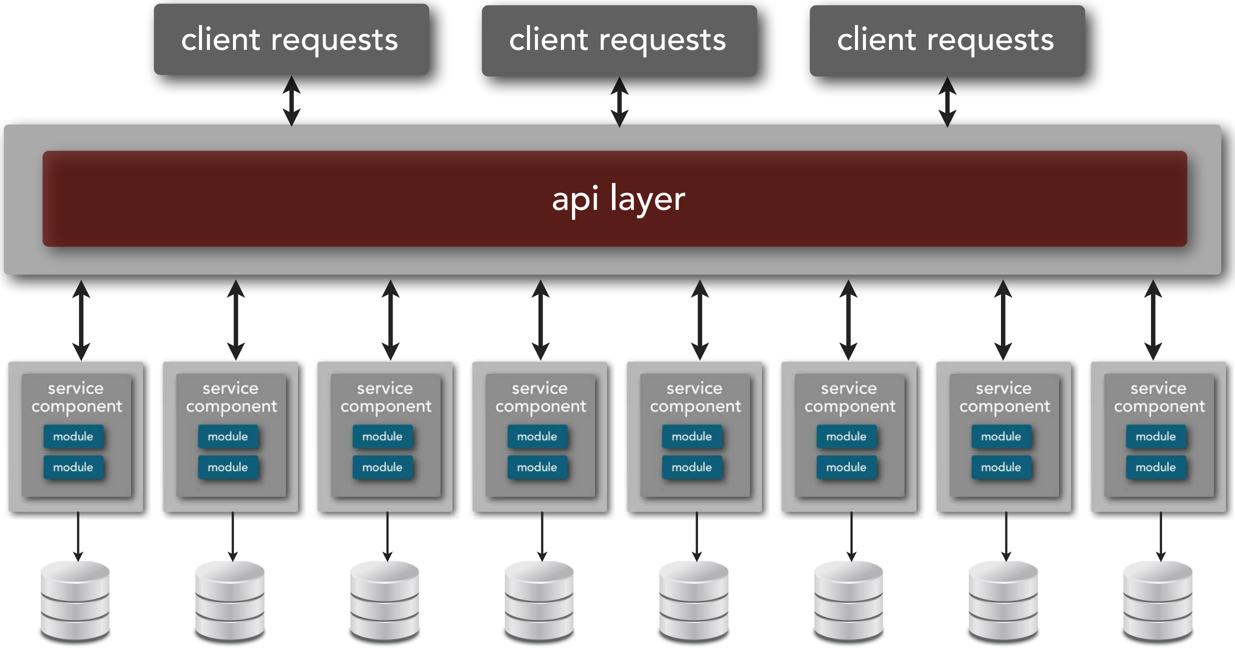
single-purpose micro-service to "portion of the application" macro-service

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

-

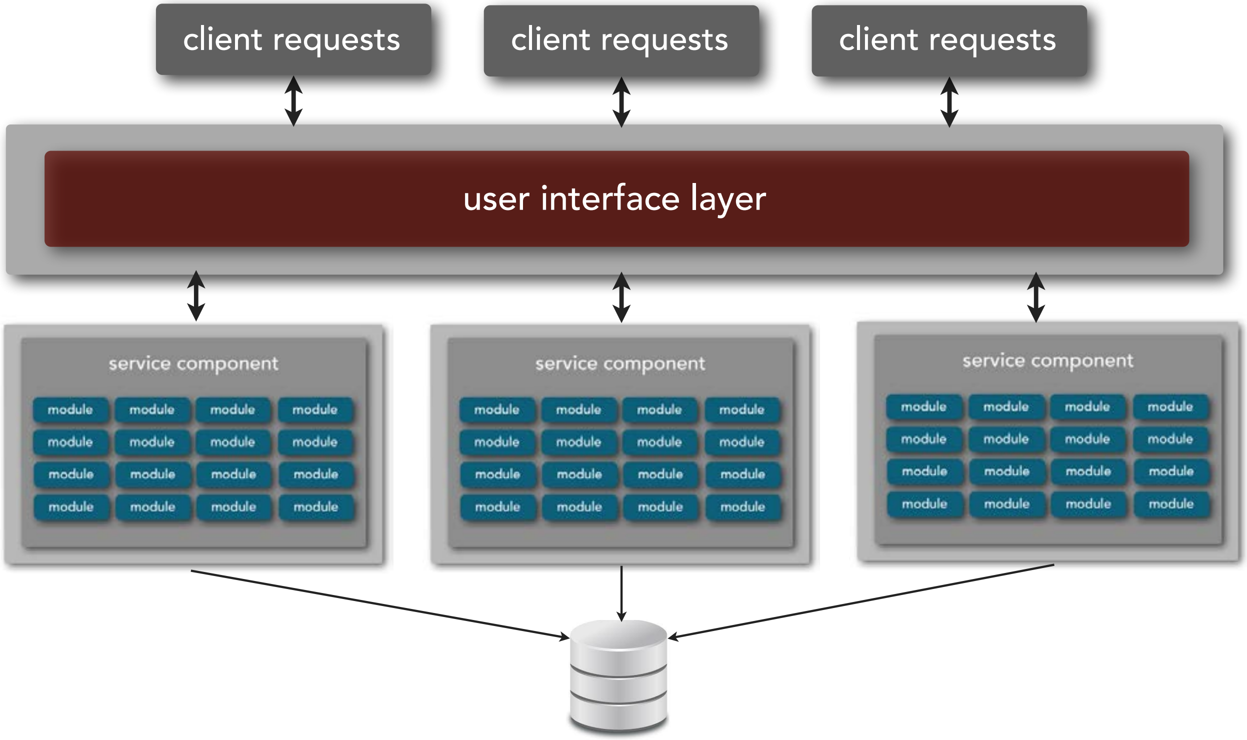
# service-based architecture

## database scope



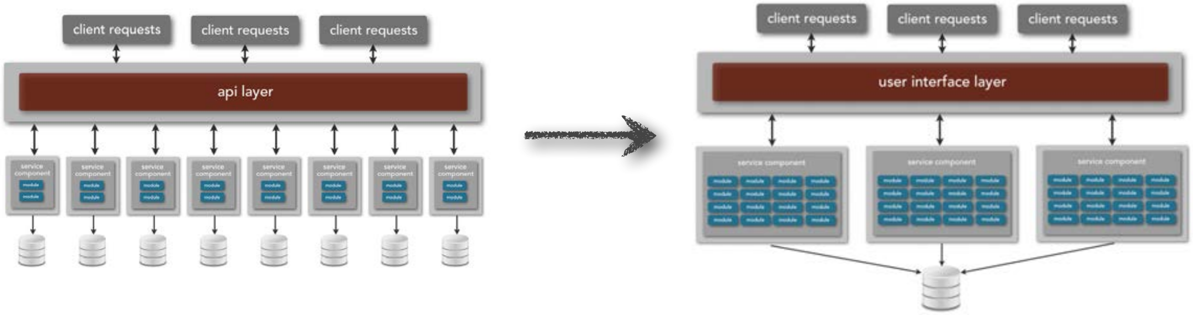
# service-based architecture

## database scope


client requests
client requests
client requests
user interface layer
service component
module module module module
module module module module
module module module module
module module module module
service component
module module module module
module module module module
module module module module
module module module module
service component
module module module module
module module module module
module module module module
module module module module

# service-based architecture

## database scope

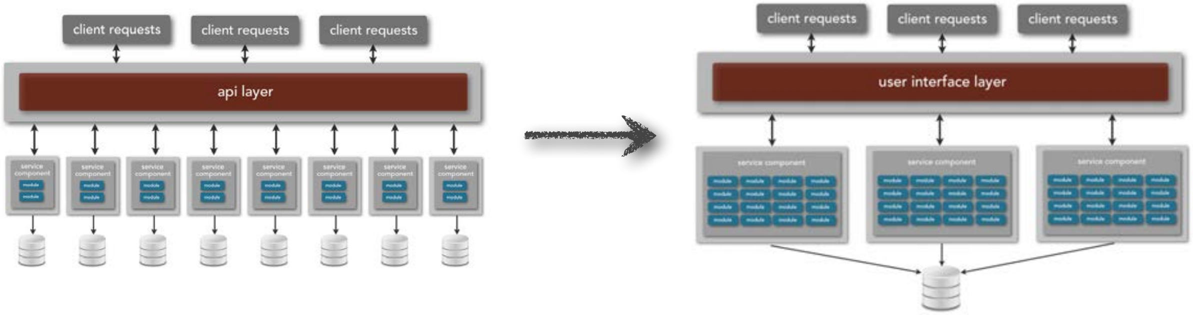


single-purpose service-based database to globally shared application database

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

# service-based architecture

## database scope

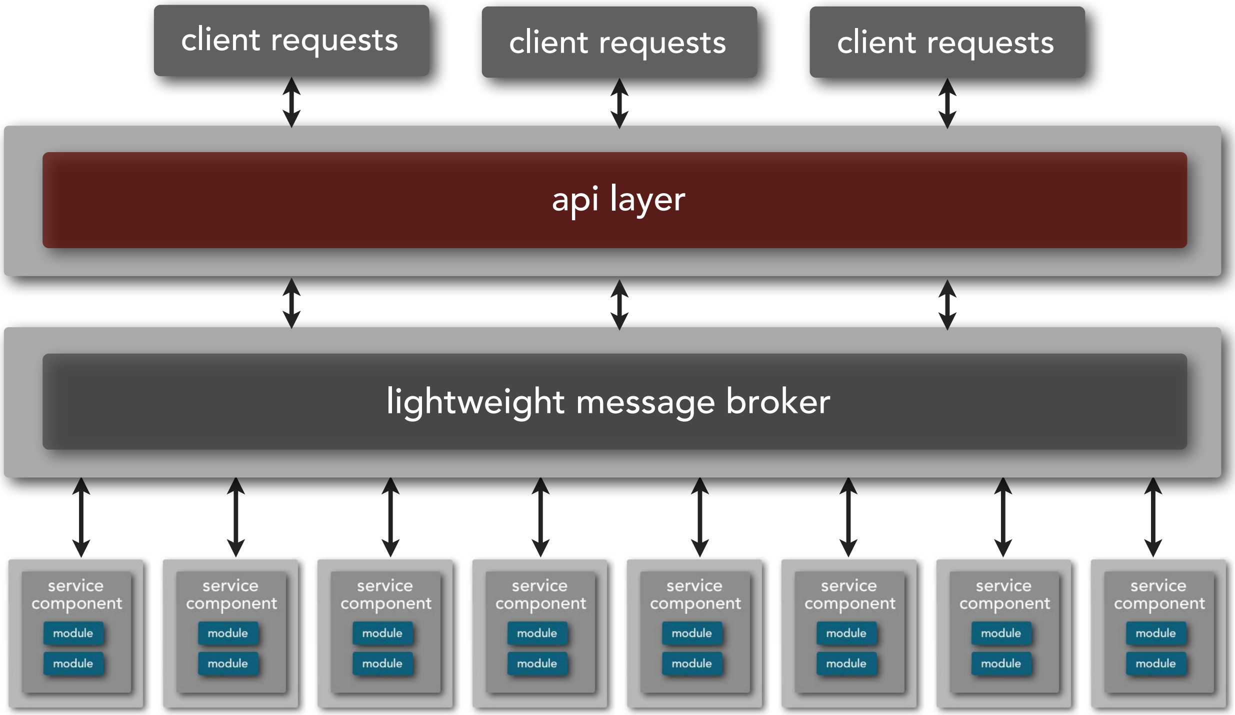


single-purpose service-based database to globally shared application database

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

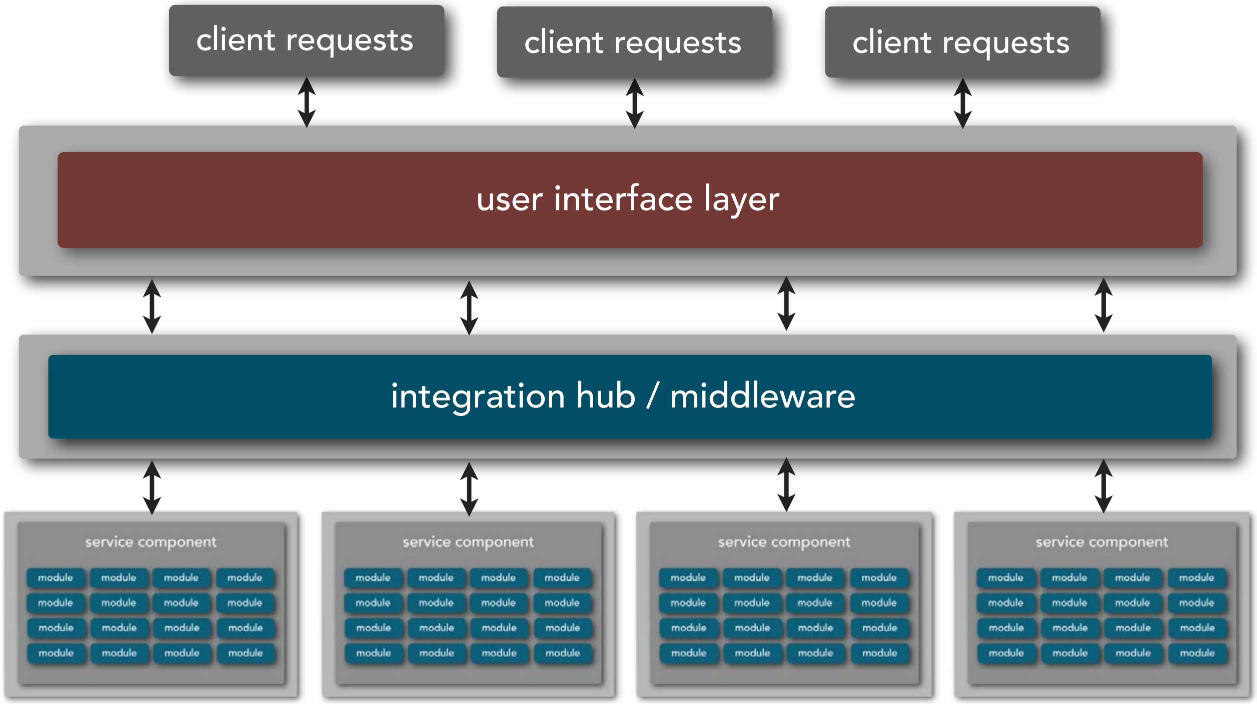
# service-based architecture

## integration hub



# service-based architecture

## integration hub



client requests

client requests

client requests

user interface layer

integration hub / middleware

service component

module module module module
module module module module
module module module module
module module module module

service component

module module module module
module module module module
module module module module
module module module module

service component

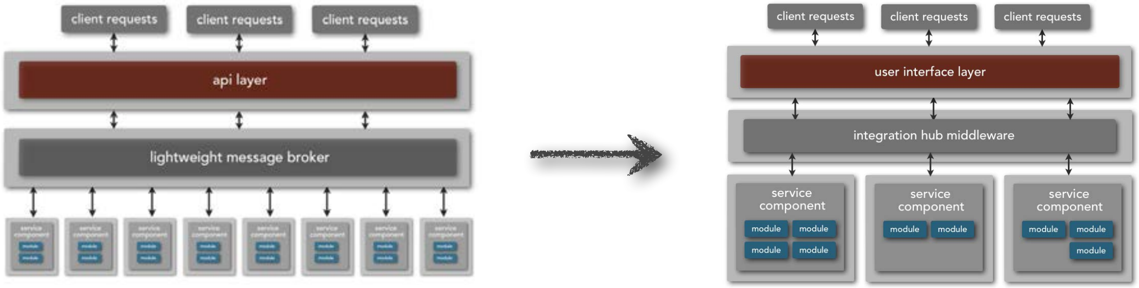
module module module module
module module module module
module module module module
module module module module

service component

module module module module
module module module module
module module module module
module module module module

# service-based architecture

## integration hub

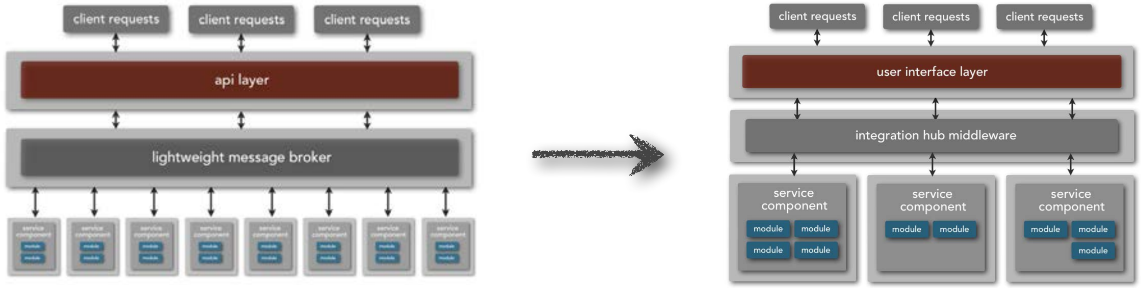


lightweight message broker to heavier integration hub

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

# service-based architecture

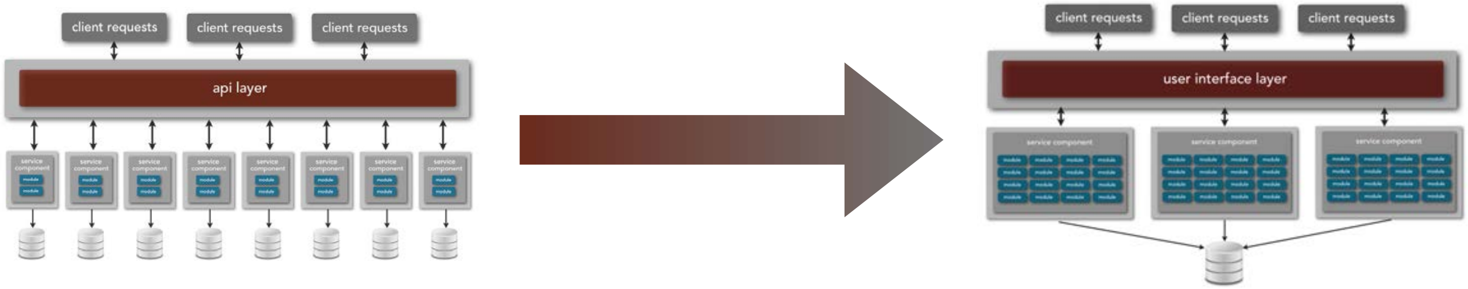
## integration hub



lightweight message broker to heavier integration hub

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

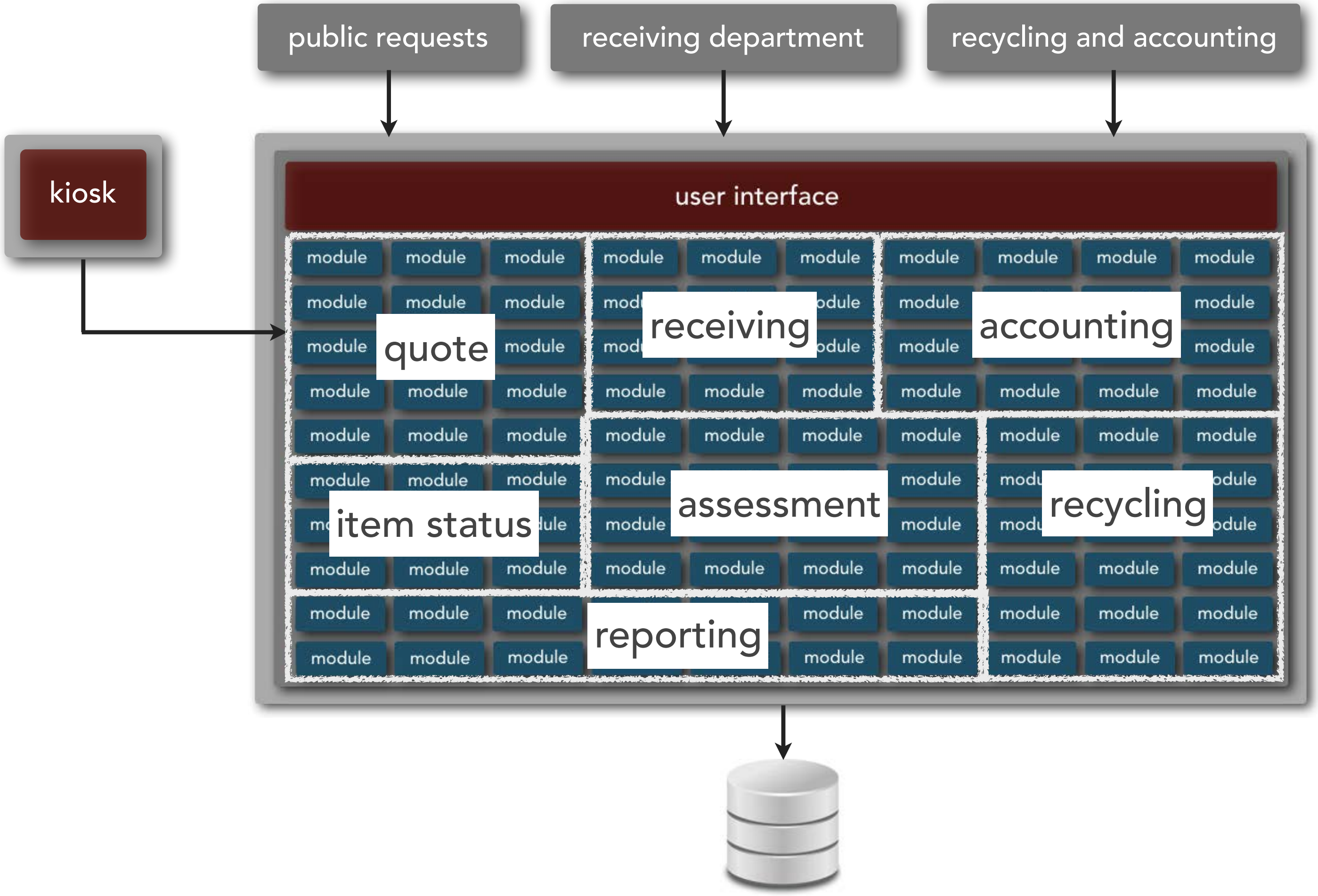
# what people really mean when they say “microservice”:



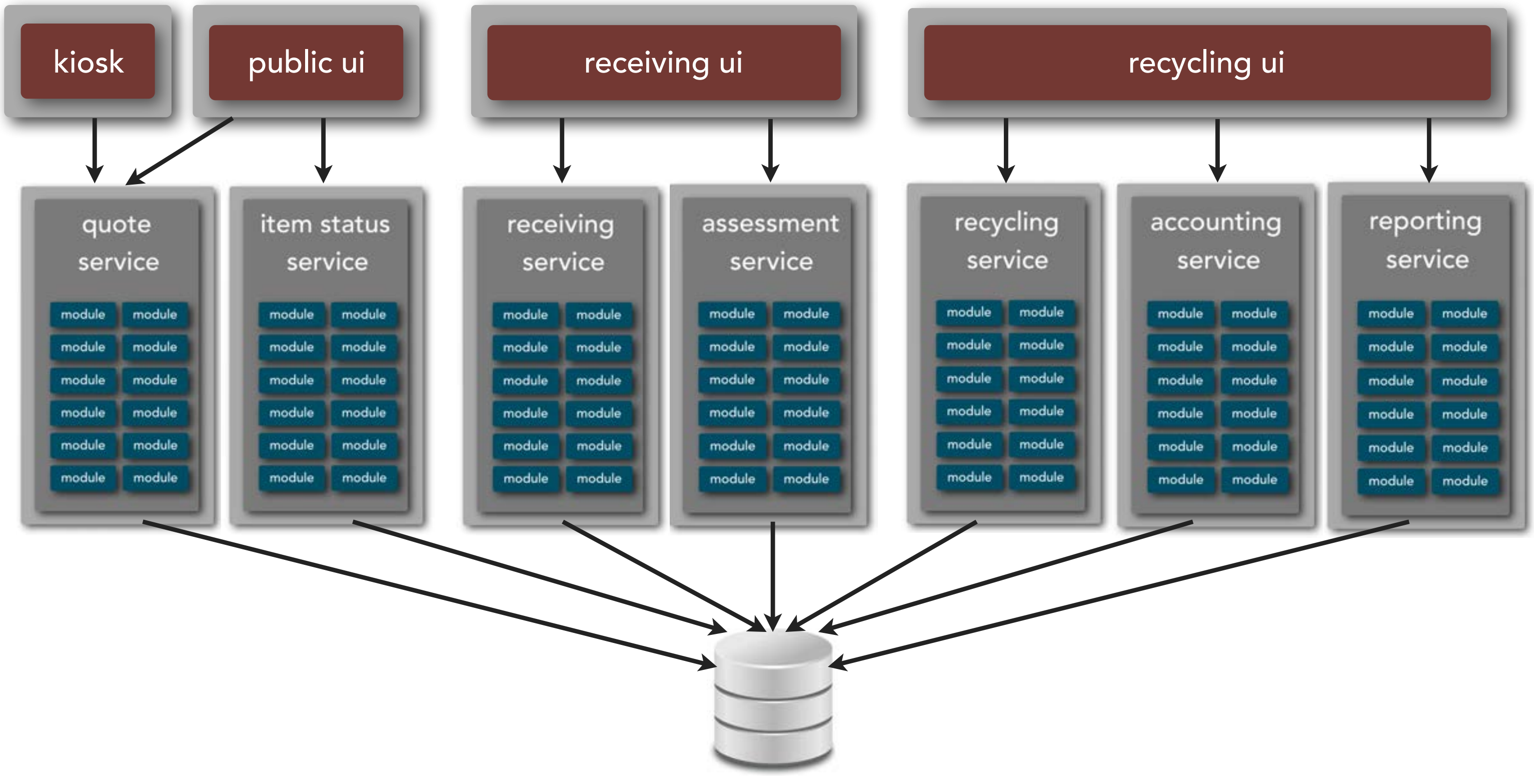
A domain-centric service based architecture with modern DevOps practices.

# migrating architectures

# electronics recycling application

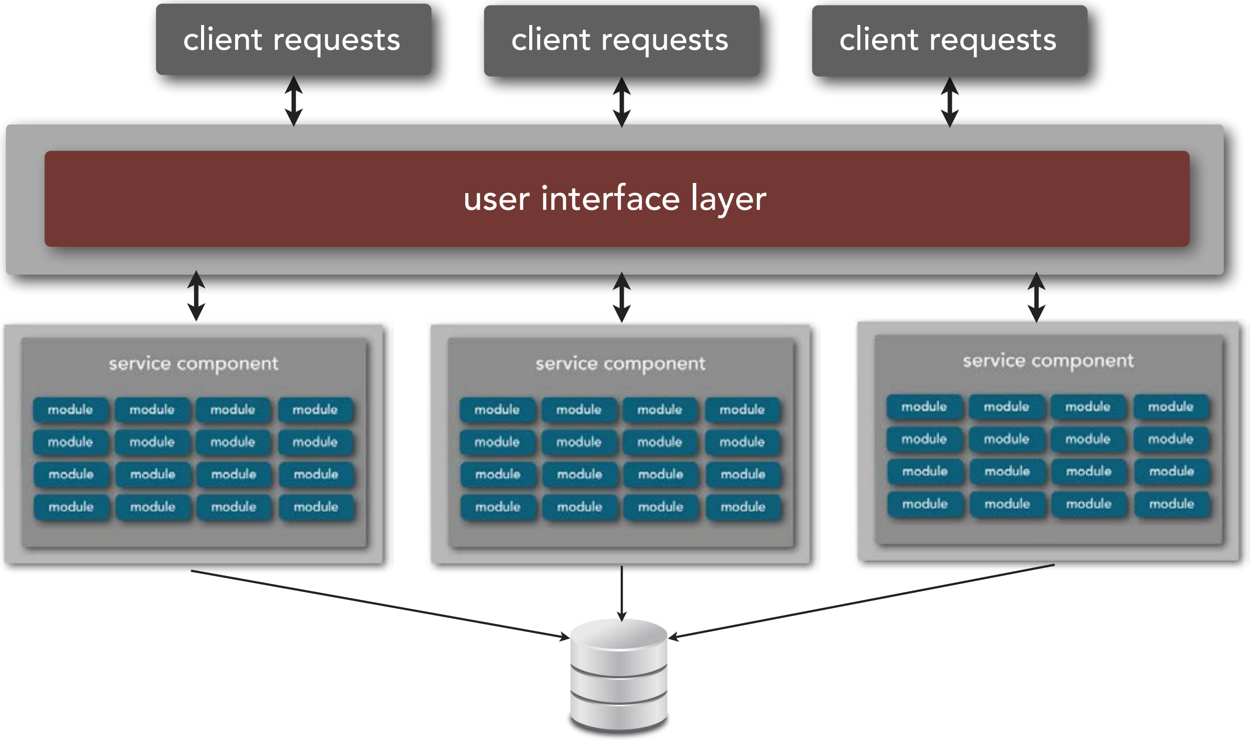


# electronics recycling application



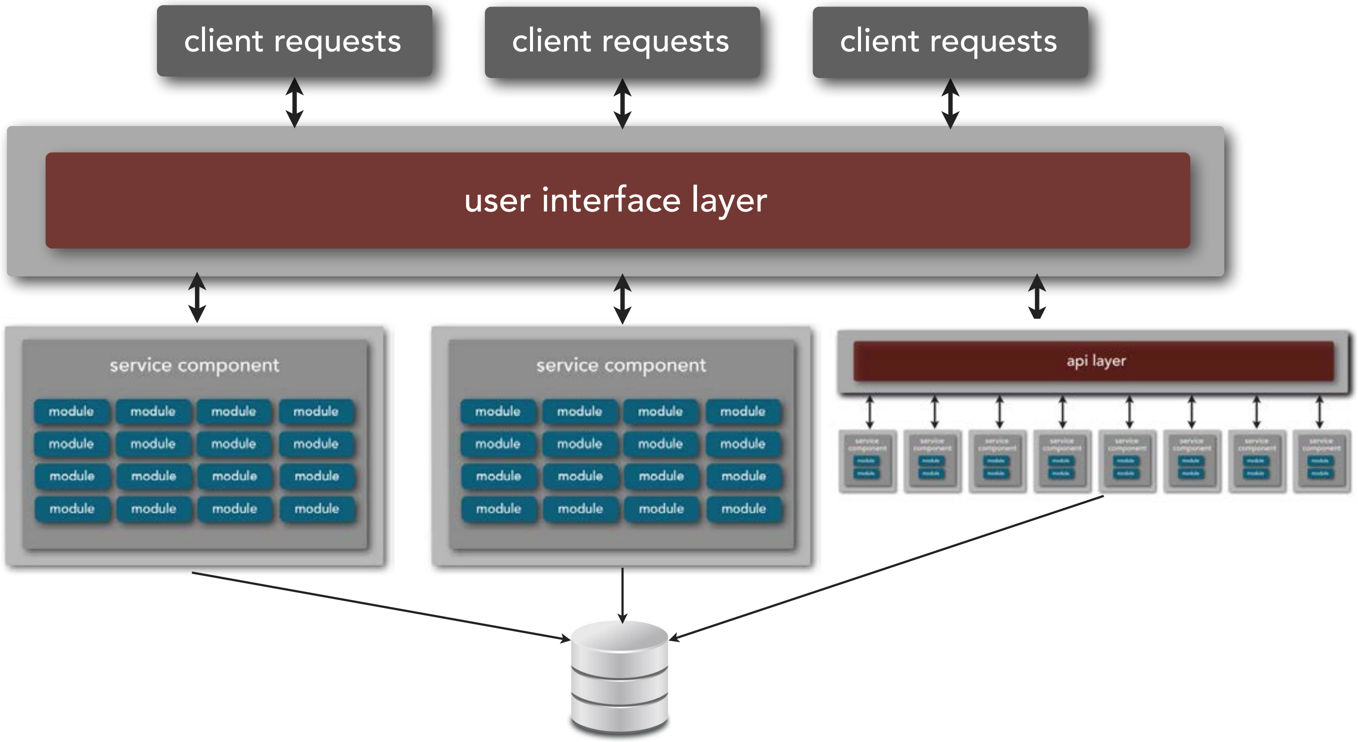
# service-based architecture

## adding microservices


client requests
client requests
client requests
user interface layer
service component
module module module module
module module module module
module module module module
module module module module
service component
module module module module
module module module module
module module module module
module module module module
service component
module module module module
module module module module
module module module module
module module module module

# service-based architecture

## adding microservices



client requests

client requests

client requests

user interface layer

service component

module module module module
module module module module
module module module module
module module module module

service component

module module module module
module module module module
module module module module
module module module module

api layer

service component
module
module

service component
module
module

service component
module
module

service component
module
module

service component
module
module

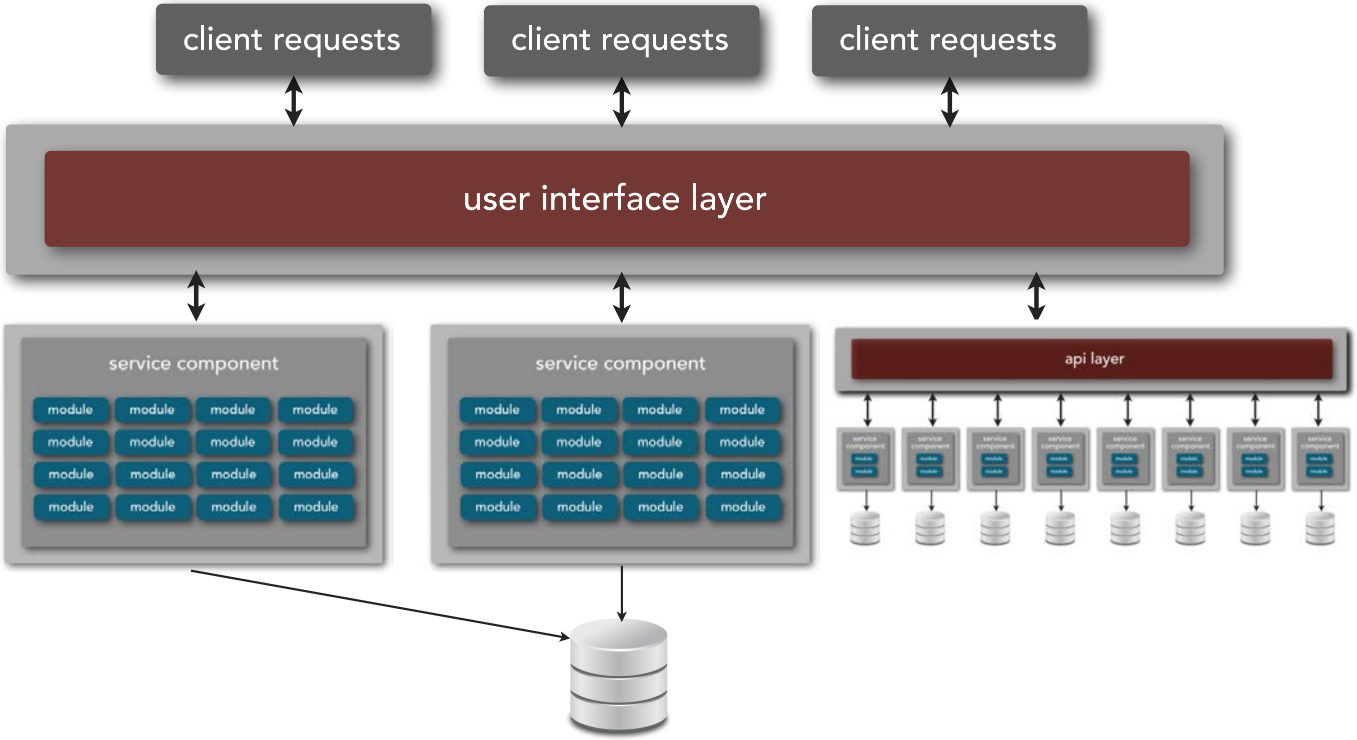
service component
module
module

service component
module
module

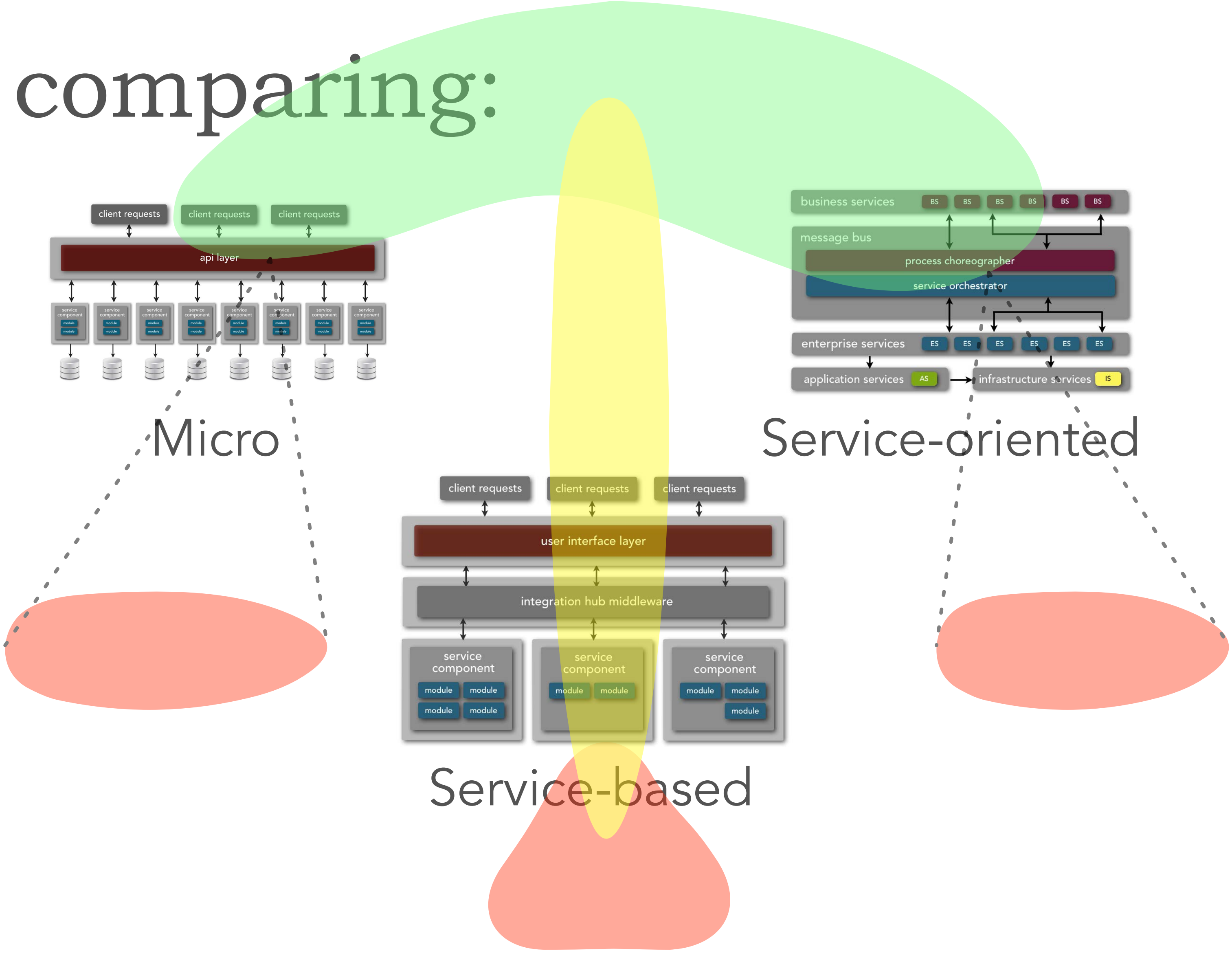
service component
module
module

# service-based architecture

## adding microservices


client requests
client requests
client requests
user interface layer
service component
module module module module
module module module module
module module module module
module module module module
service component
module module module module
module module module module
module module module module
module module module module
api layer
service component
module
module

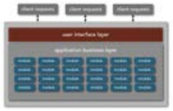
# comparing:


client requests
client requests
client requests
api layer
service component
module
Micro
business services
BS
message bus
process choreographer
service orchestrator
enterprise services
ES
application services
AS
infrastructure services
IS
Service-oriented
client requests
client requests
client requests
user interface layer
integration hub middleware
service component
module
Service-based

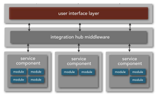
# characteristics differences

## overall agility

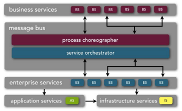
ability to respond quickly to constant change in both business and technology



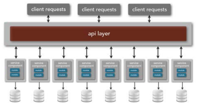
monolithic architecture



service-based architecture



service-oriented architecture



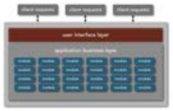
microservices architecture

H
M
L
overall agility

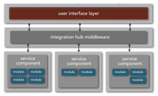
# characteristics differences

## ease of deployment

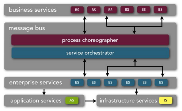
promotes an effective and fast deployment pipeline; features are quick and easy to deploy



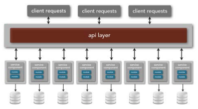
monolithic architecture



service-based architecture



service-oriented architecture



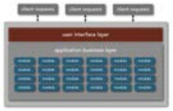
microservices architecture

ease of deployment
H
M
L

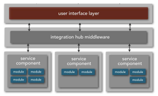
# characteristics differences

## ease of testing

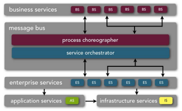
ease at which features can be tested and verified; confidence level in completeness of testing



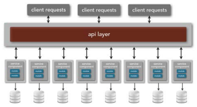
monolithic architecture



service-based architecture



service-oriented architecture



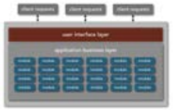
microservices architecture

ease of testing
H
M
L

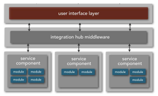
# characteristics differences

## overall performance

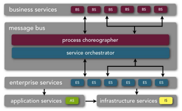
which patterns relatively promote better performing applications?



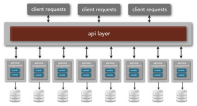
monolithic architecture



service-based architecture



service-oriented architecture



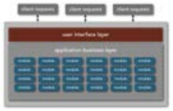
microservices architecture

level of performance
H
M
L

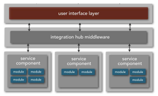
# characteristics differences

## overall scalability

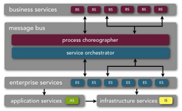
how well does the architecture pattern lend itself to highly scalable applications?



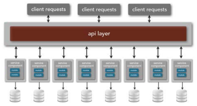
monolithic architecture



service-based architecture



service-oriented architecture



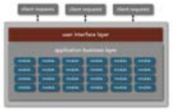
microservices architecture

level of scalability
H
M
L

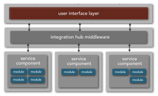
# characteristics differences

## overall simplicity

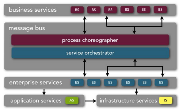
level of complexity in applications implemented using the architecture pattern



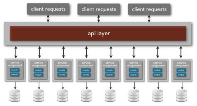
monolithic architecture



service-based architecture



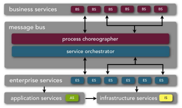
service-oriented architecture



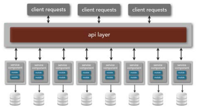
microservices architecture

level of simplicity
H
M
L

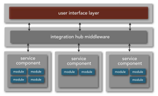
# service differences



service-oriented architecture



microservices architecture



service-based architecture

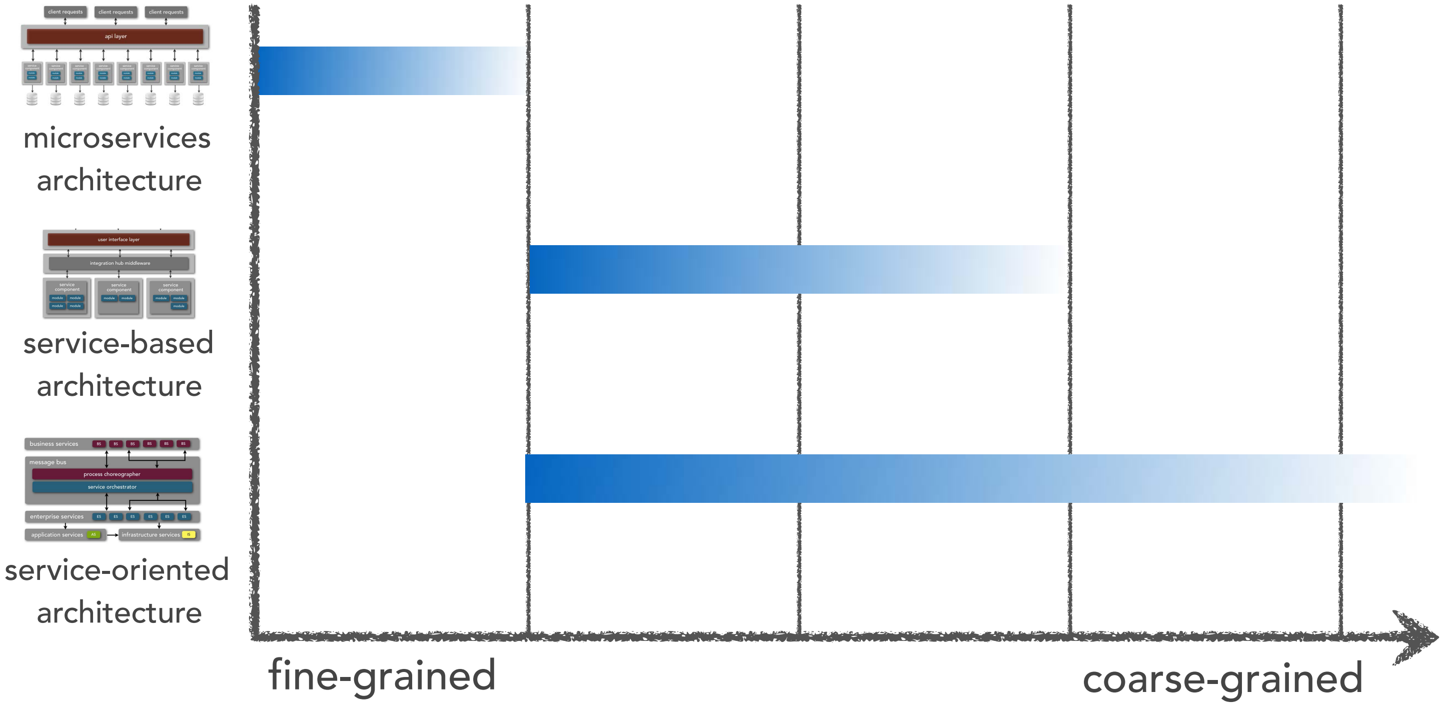
service granularity

service numbers

# service differences

## service granularity

what is the typical granularity of services within this pattern?



microservices architecture

service-based architecture

service-oriented architecture

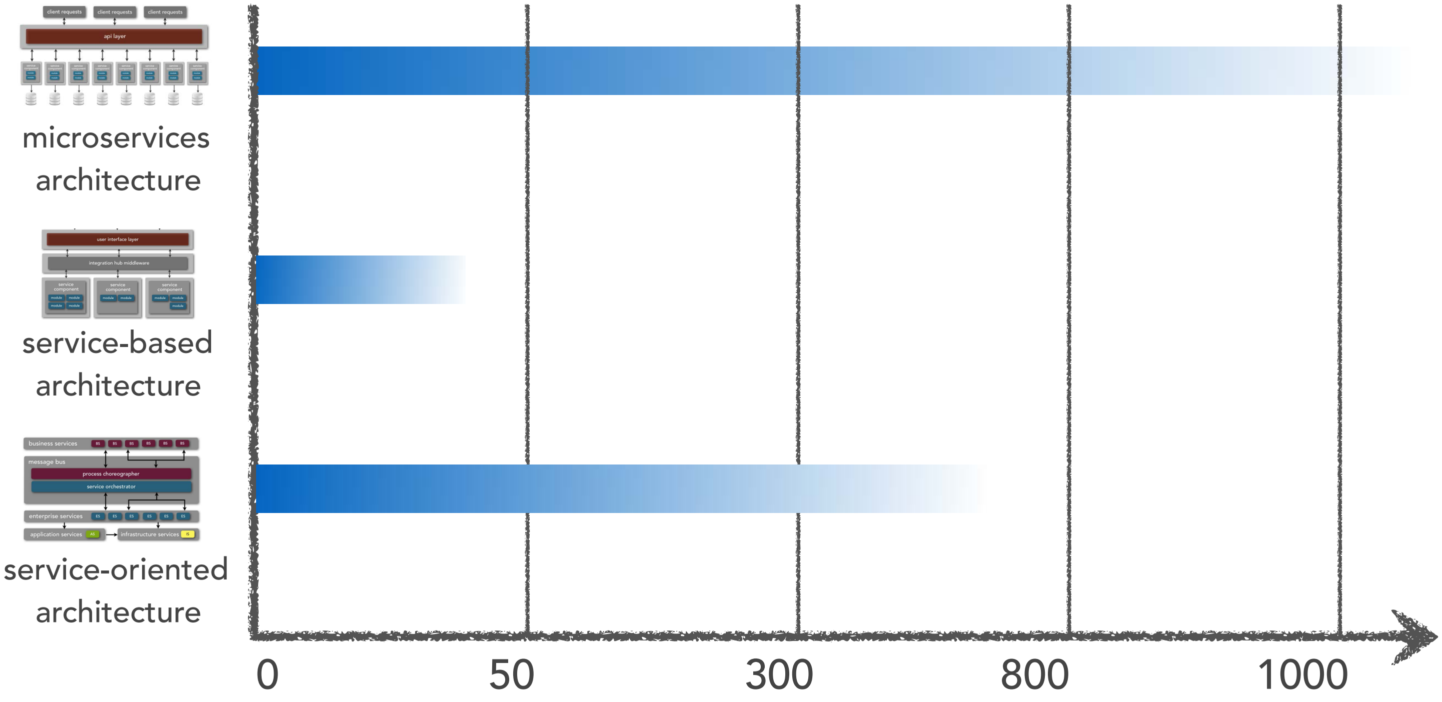
fine-grained

coarse-grained

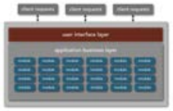
# service differences

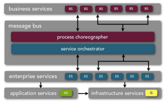
## service numbers

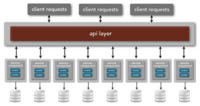
what is the typical upper limit of the number of services found?

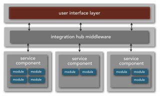


# Which ?!?

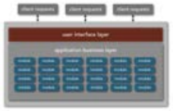

client requests
client requests
client requests
user interface layer
application business layer
module


business services
BS
BS
BS
BS
BS
BS
message bus
process choreographer
service orchestrator
enterprise services
ES
ES
ES
ES
ES
ES
application services
AS
infrastructure services
IS

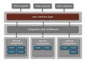

client requests
client requests
client requests
api layer
service component
module


user interface layer
integration hub middleware
service component
service component
service component
module

# Which ?!?



Monolith

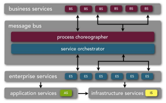


default

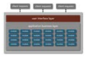
easy to understand/build

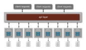
doesn’t scale well in any dimension

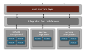
# Which ?!?



Service-oriented







service taxonomy

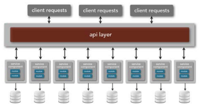
high degree of (potential) reuse

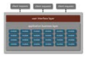
highly compatible in integration-heavy environments

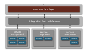
operationally complex

# Which ?!?

Microservices





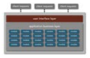


incremental change

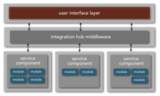
highly evolvable

complex interactions

# Which ?!?



service-based

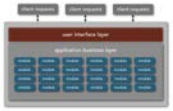


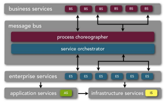
best target for migration

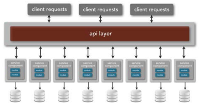
good compromise

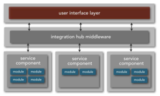
domain-centric without going ***µ***service

# Which ?!?



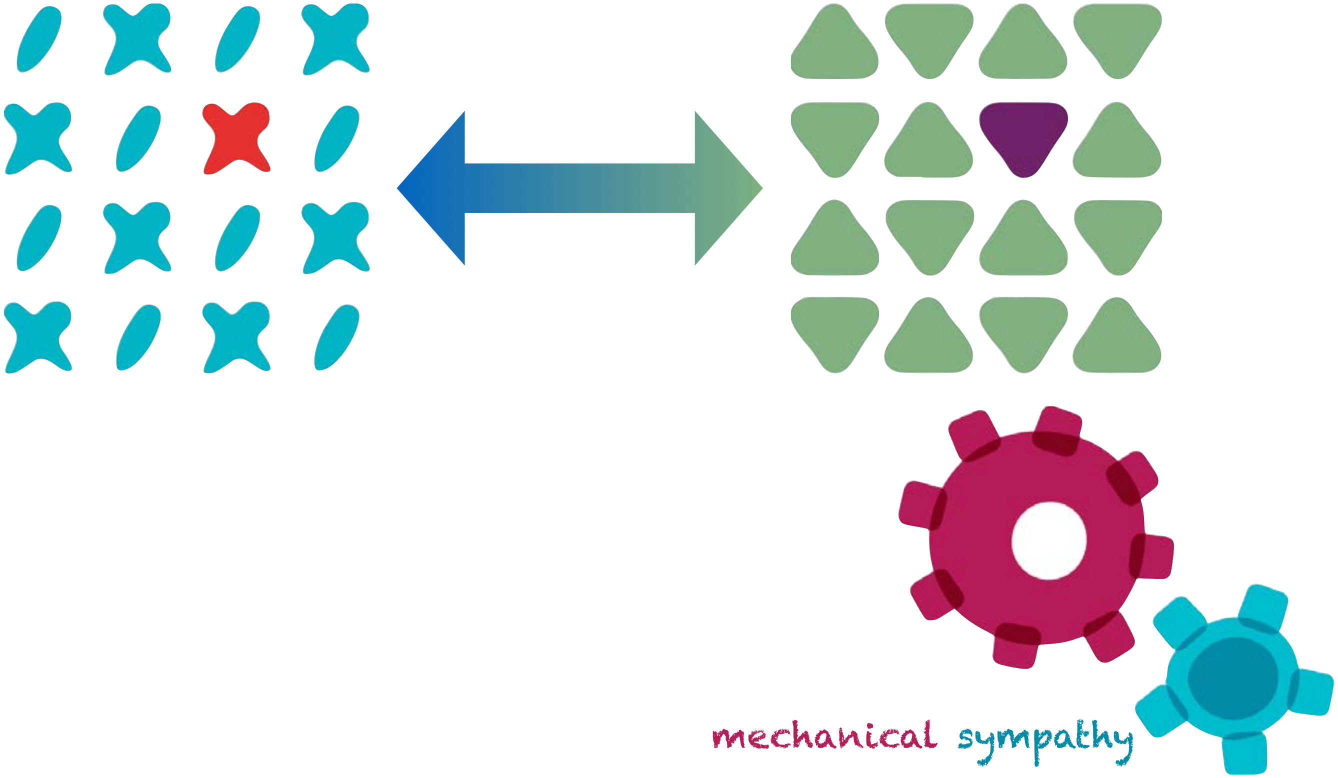




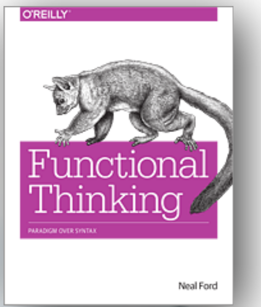


it depends

# domain/architecture isomorphism



Presentation PATTERNS
Techniques for Crafting Better Presentations
NEAL FORD | MATTHEW MCCULLOUGH | NATHANIEL SCHUTTA


O'REILLY
Functional Thinking
PARADIGM OVER SYNTAX
Neal Ford

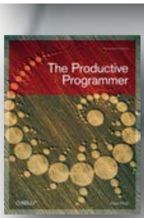
nealford.com

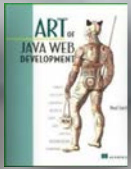
@neal4d

ThoughtWorks

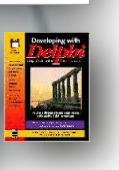
**NEAL FORD**

Director / Software Architect / Meme Wrangler


The Productive Programmer

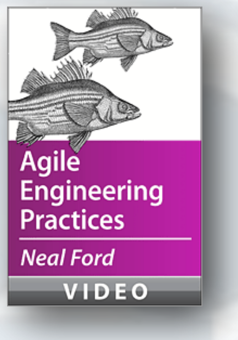

ART of JAVA WEB DEVELOPMENT

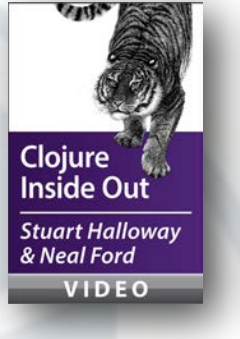
Borland JBuilder 3 Unleashed

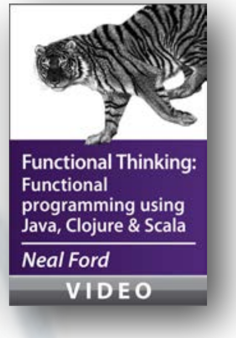

Developing with Delphi

nealford.com/books

nealford.com/videos


Agile Engineering Practices
Neal Ford
VIDEO

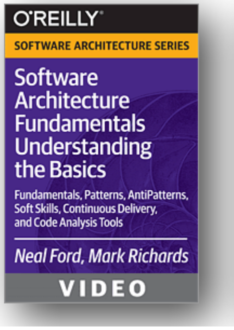

Clojure Inside Out
Stuart Halloway & Neal Ford
VIDEO

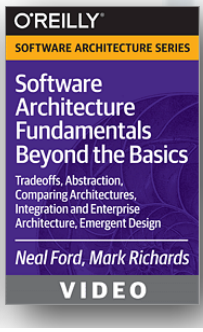

Functional Thinking: Functional programming using Java, Clojure & Scala
Neal Ford
VIDEO

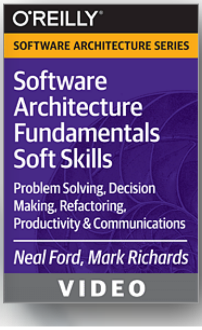
O'REILLY®

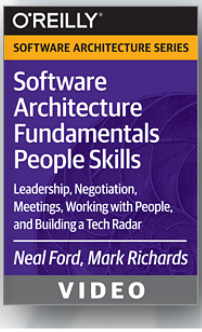
SOFTWARE ARCHITECTURE SERIES

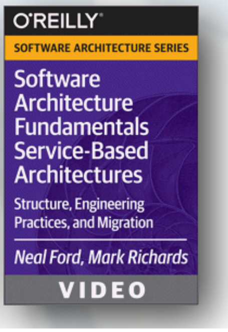
www.oreilly.com/software-architecture-video-training-series.html

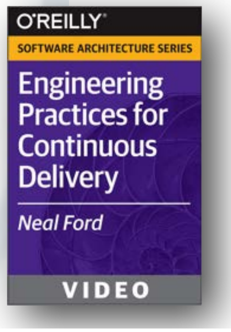

O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Understanding the Basics
Fundamentals, Patterns, AntiPatterns, Soft Skills, Continuous Delivery, and Code Analysis Tools
Neal Ford, Mark Richards
VIDEO


O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Beyond the Basics
Tradeoffs, Abstraction, Comparing Architectures, Integration and Enterprise Architecture, Emergent Design
Neal Ford, Mark Richards
VIDEO


O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Soft Skills
Problem Solving, Decision Making, Refactoring, Productivity & Communications
Neal Ford, Mark Richards
VIDEO


O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals People Skills
Leadership, Negotiation, Meetings, Working with People, and Building a Tech Radar
Neal Ford, Mark Richards
VIDEO


O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Service-Based Architectures
Structure, Engineering Practices, and Migration
Neal Ford, Mark Richards
VIDEO


O'REILLY
SOFTWARE ARCHITECTURE SERIES
Engineering Practices for Continuous Delivery
Neal Ford
VIDEO