Software Architecture Fundamentals Workshop

Part 1: From Developer to Architect

Mark Richards
Independent Consultant
Hands-on Enterprise / Integration Architect
Published Author / Conference Speaker

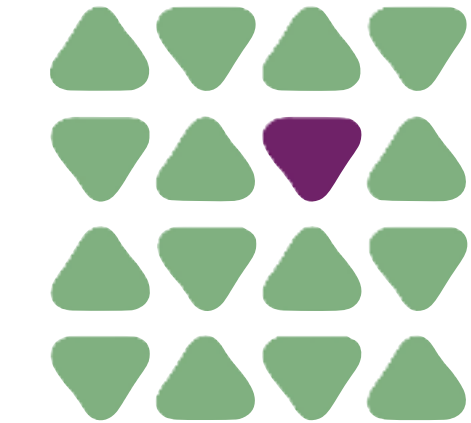
http://www.wmrichards.com
http://www.linkedin.com/pub/mark-richards/0/121/5b9

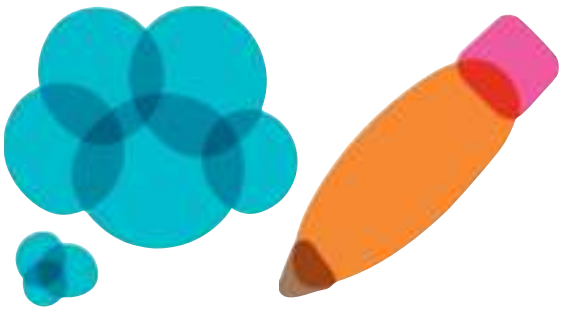
ThoughtWorks®

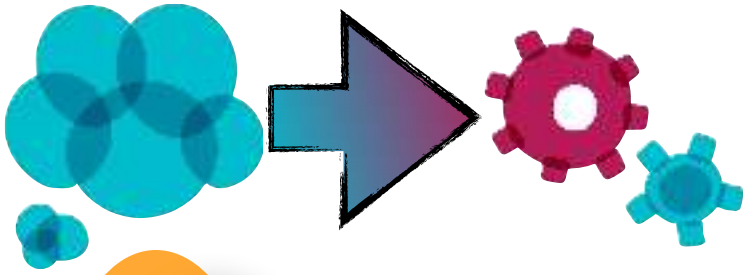
NEAL FORD

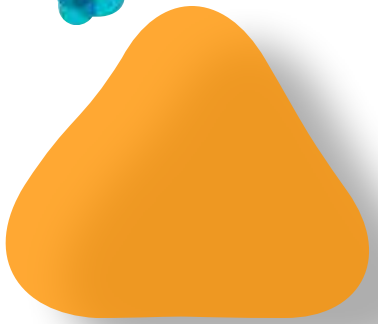
Director / Software Architect / Meme Wrangler

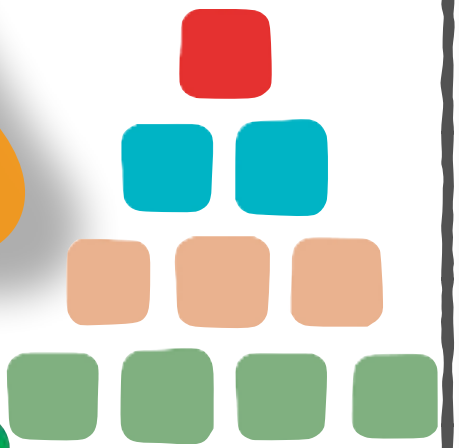
agenda

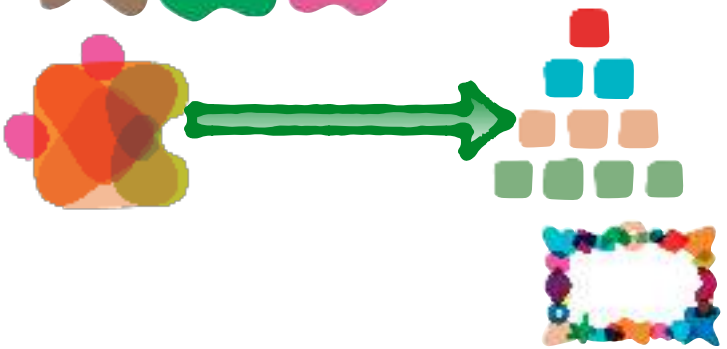












nealford.com/katas/

Best Jobs in America

2015

CNNMoney/PayScale's top 100 careers with big growth, great pay and satisfying work.

1. Software Architect

NEXT

Median pay: $124,000
Top pay: $169,000
10-year job growth: 23%

Christopher Felpel, software architect consultant

In the same way an architect designs a house, software architects lay out a design plan for new programs. That usually means leading a team of developers and engineers, and making sure all the pieces come together to make fully-functioning software.

What's great: New problems come up all the time and new technologies arise, making each day different, and keeping professionals in demand. "I'm pinged at least once or twice a week for new opportunities," said software architect Christopher Felpel. "There's just a lot of work out there, and that doesn't surprise me." --*Jillian Eugenios*

Quality of life ratings:
Personal satisfaction: A | Benefit to society: B | Telecommuting: A | Low stress: A

100 Best Jobs in America


Top 100: Full list


Top-paying jobs


Fastest-growing jobs

“Programmers know the benefits of everything and the tradeoffs of nothing.”

Architects must understand both.

software architecture?

"the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces."

Rational Unified Process definition, working off the IEEE definition

http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

software architecture?

Architecture is the highest level concept of the expert developers.

"In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called 'architecture.' This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers."

http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

software architecture?

developers

product
owner

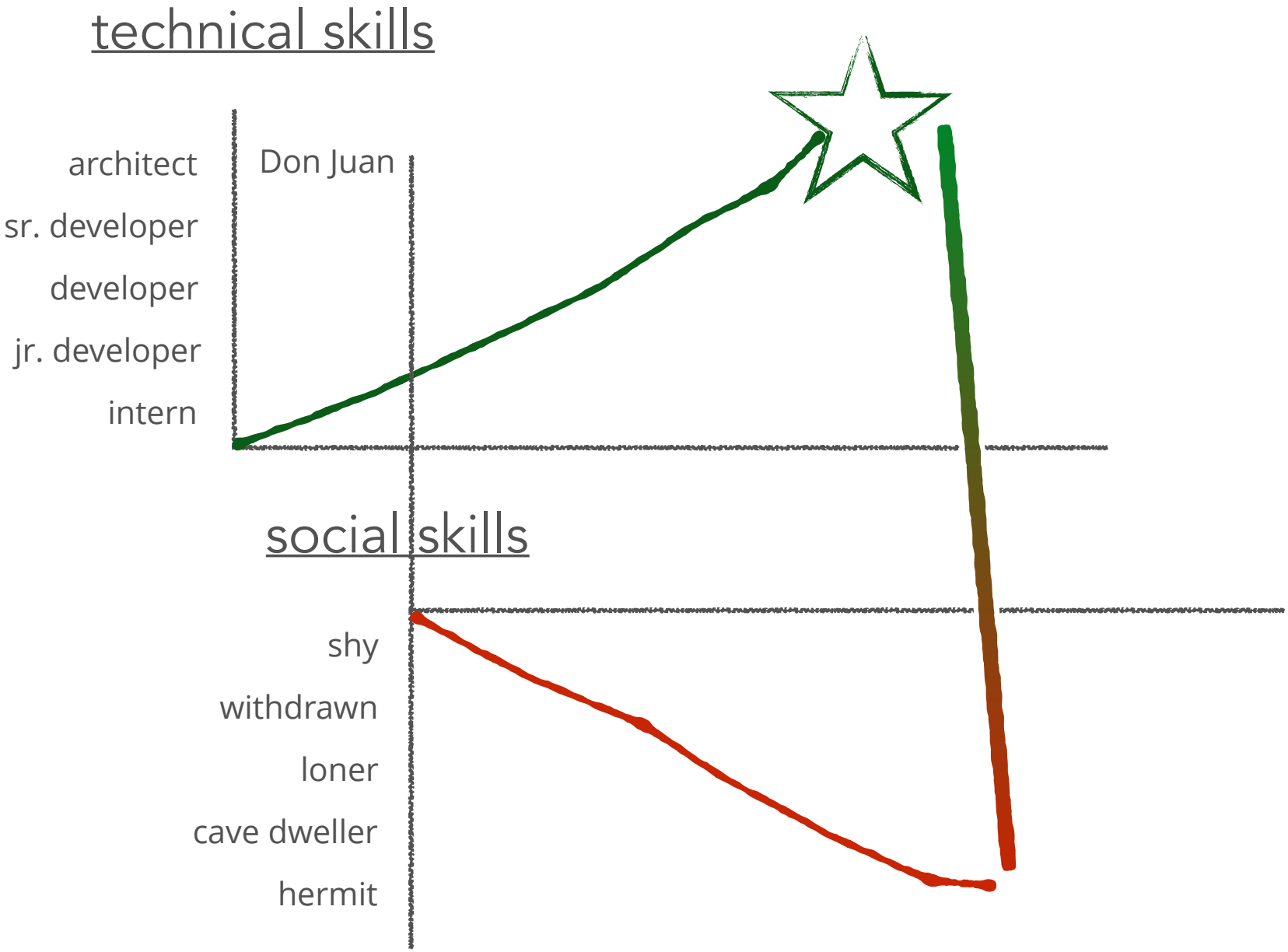
Architecture is about the important stuff.
Whatever that is.

Martin Fowler

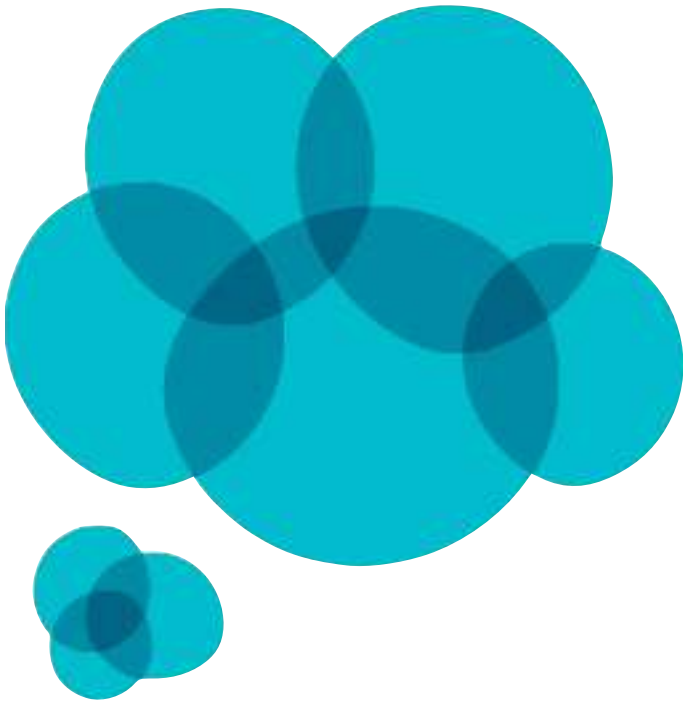
operations

http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

soft skills

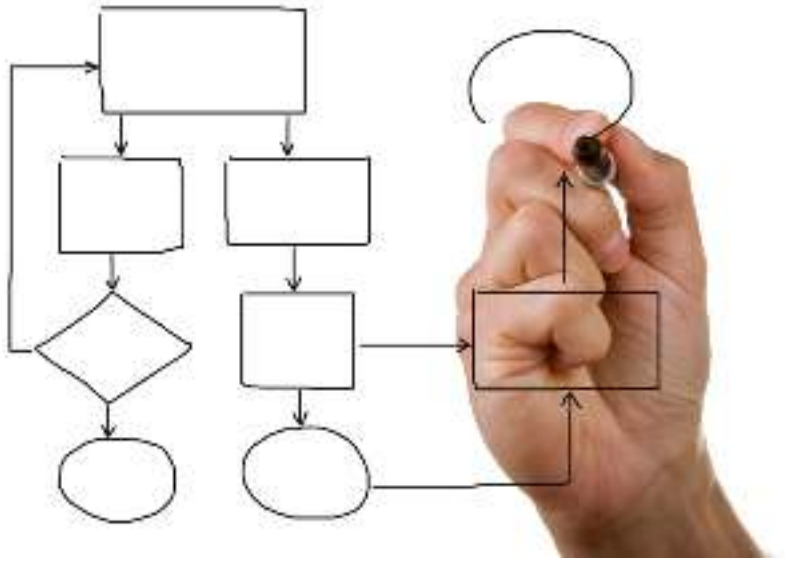

technical skills
architect
sr. developer
developer
jr. developer
intern
Don Juan
social skills
shy
withdrawn
loner
cave dweller
hermit

Decisions

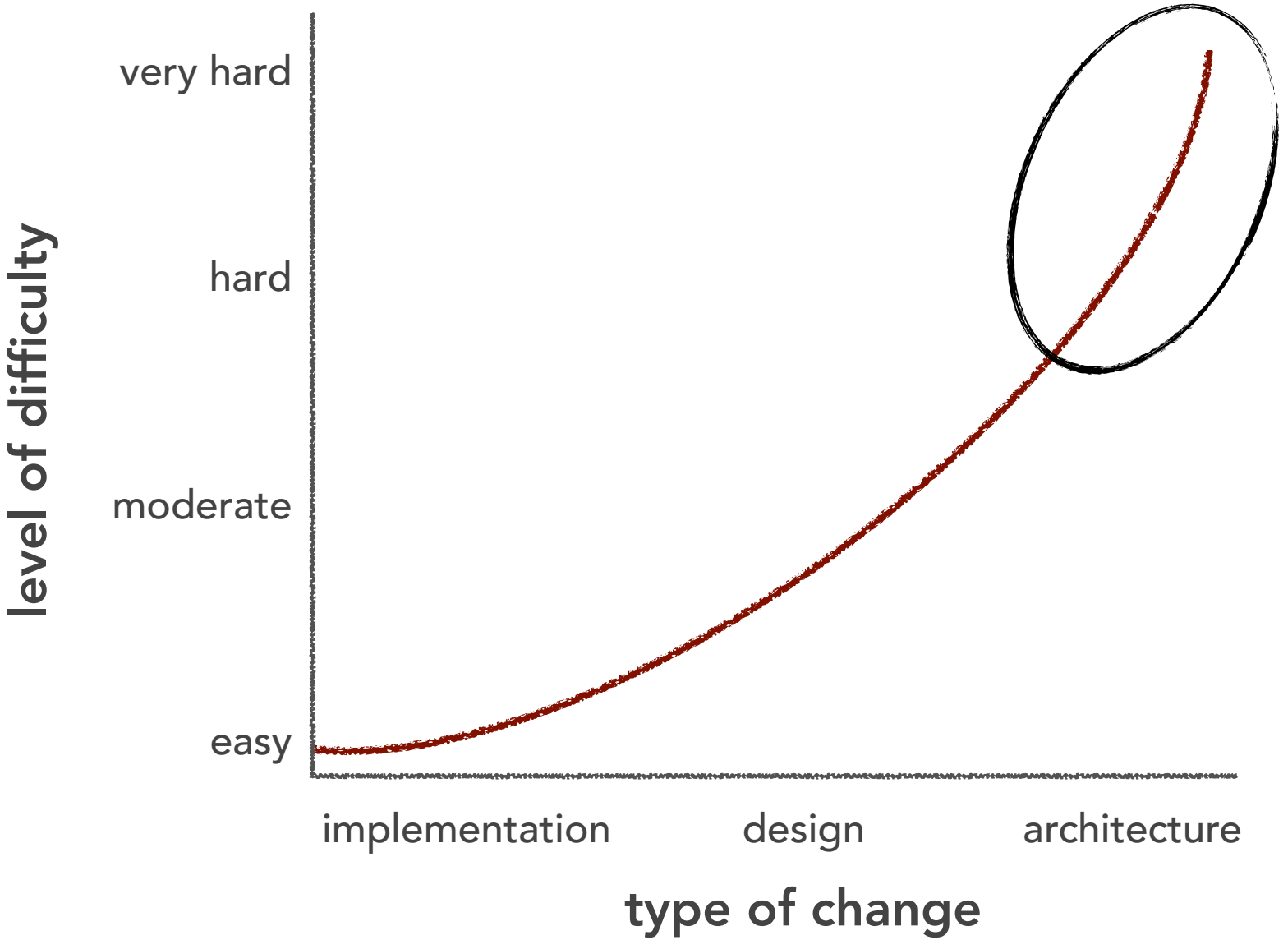


architecture decisions

what is an architecture decision?

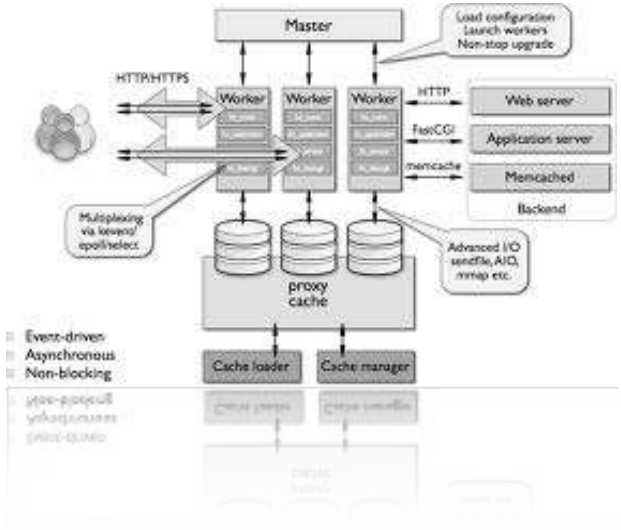


architecture decisions



architecture decisions

an architect is responsible for defining the architecture and design principles used to guide technology decisions



architecture decisions

the decision to use java server faces as your web framework

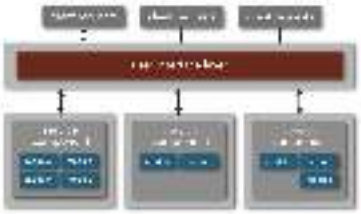
vs.

the decision to use a web-based user interface for your application

architecture decisions

the decision to use rest to communicate between distributed components

vs.



the decision that components should be distributed remotely for better scalability

justifying
architecture decisions

justifying decisions

groundhog day anti-pattern

no one understands why a decision was made so it keeps getting discussed over and over and over...

justifying decisions

the scenario

internal client

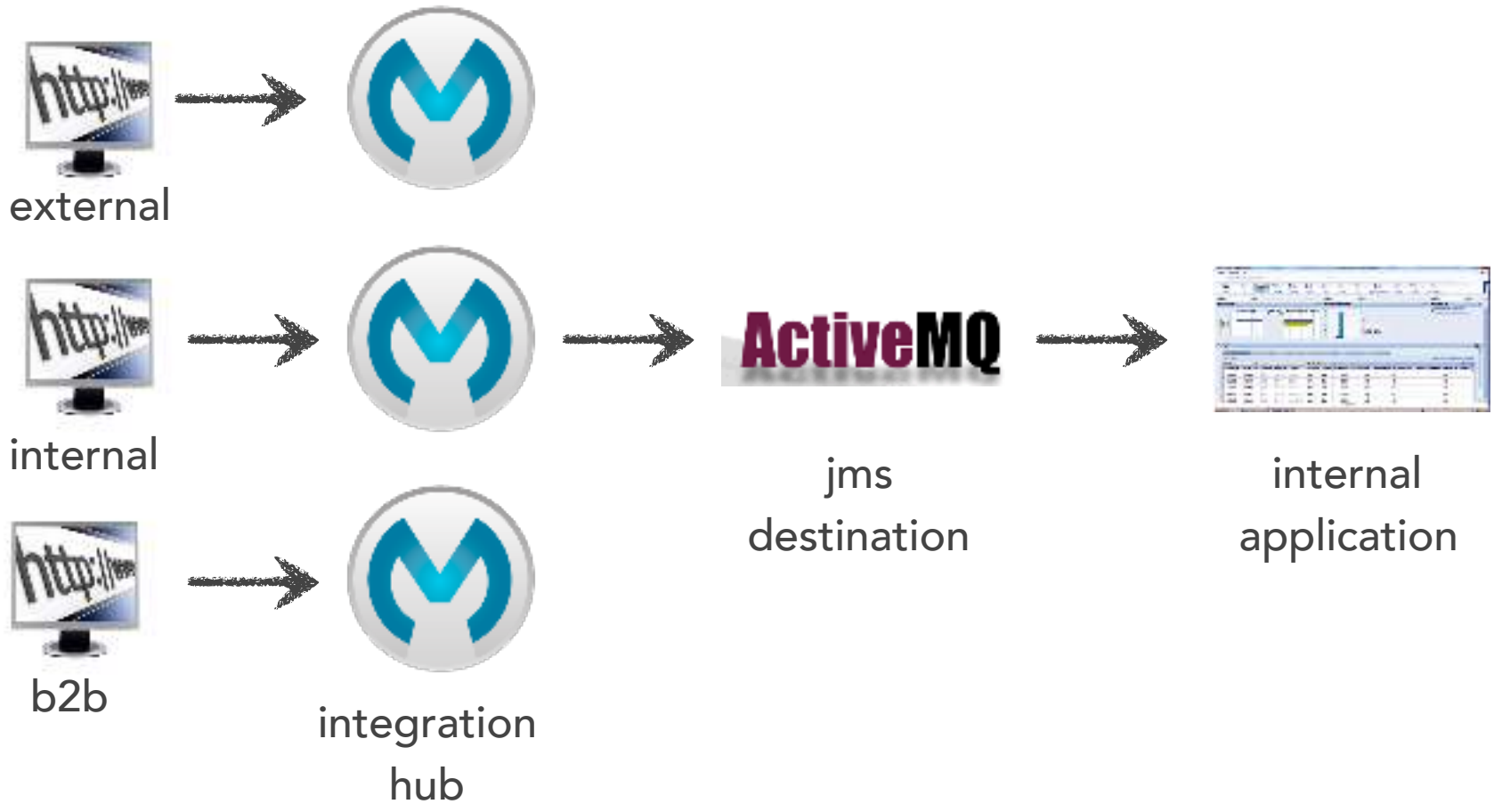
integration hub

jms destination

internal application

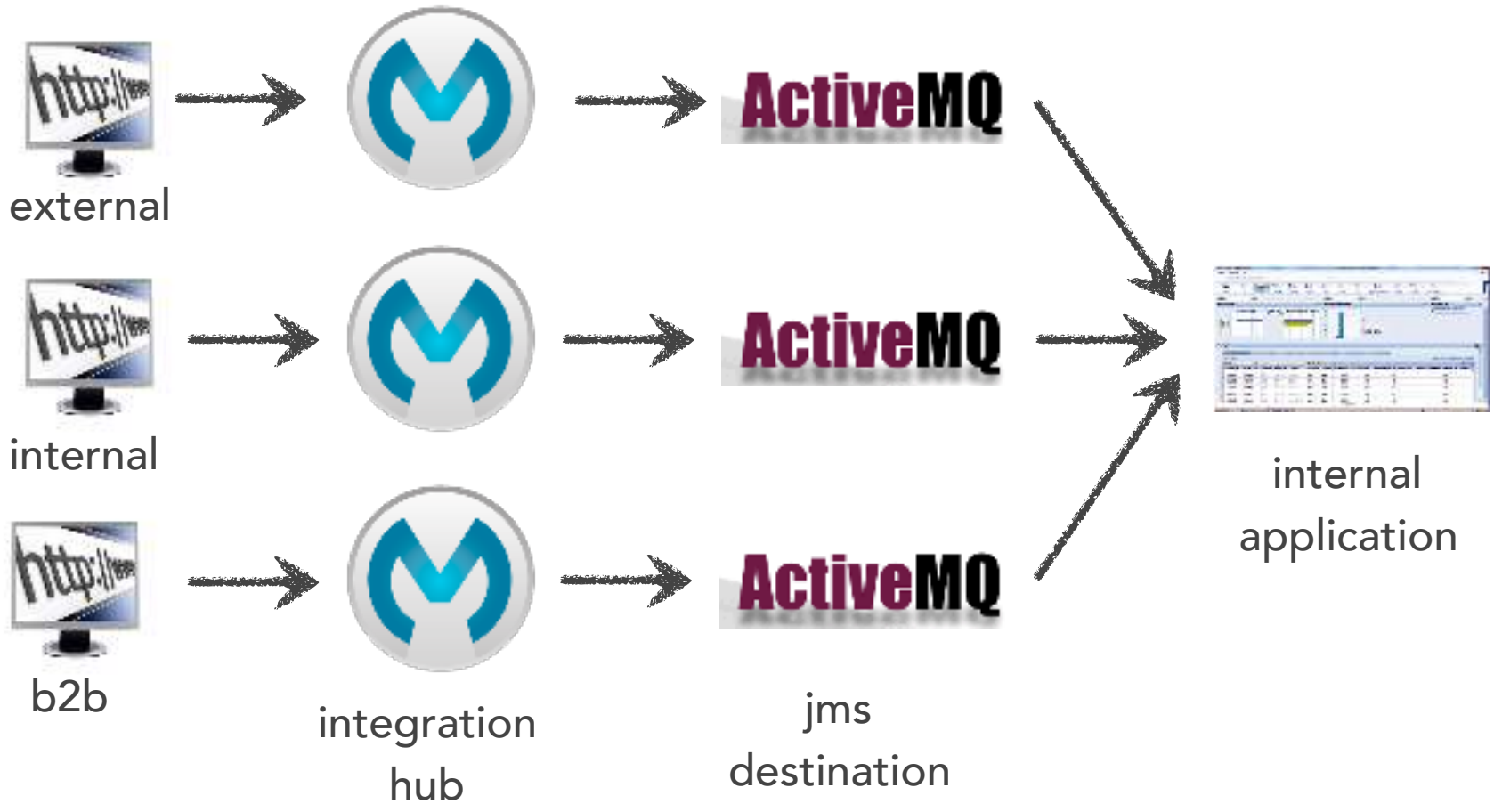
justifying decisions

the requirement: you need to federate the hub



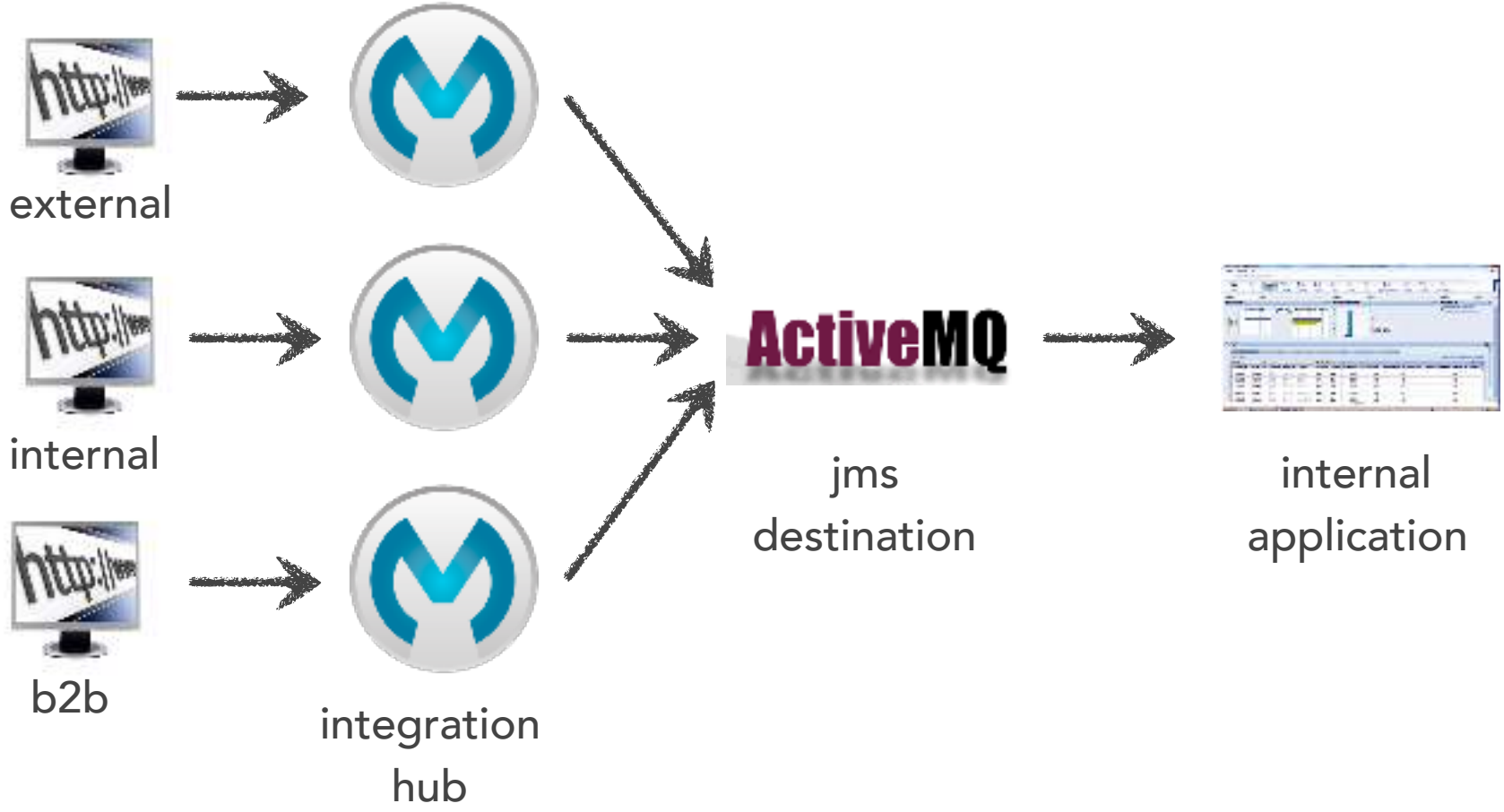
justifying decisions

the decision: dedicated broker instances?



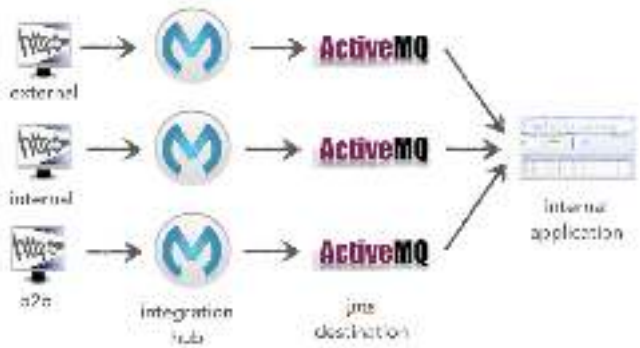
justifying decisions

the decision: centralized broker



justifying decisions

identify the conditions and constraints



conditions and constraints:

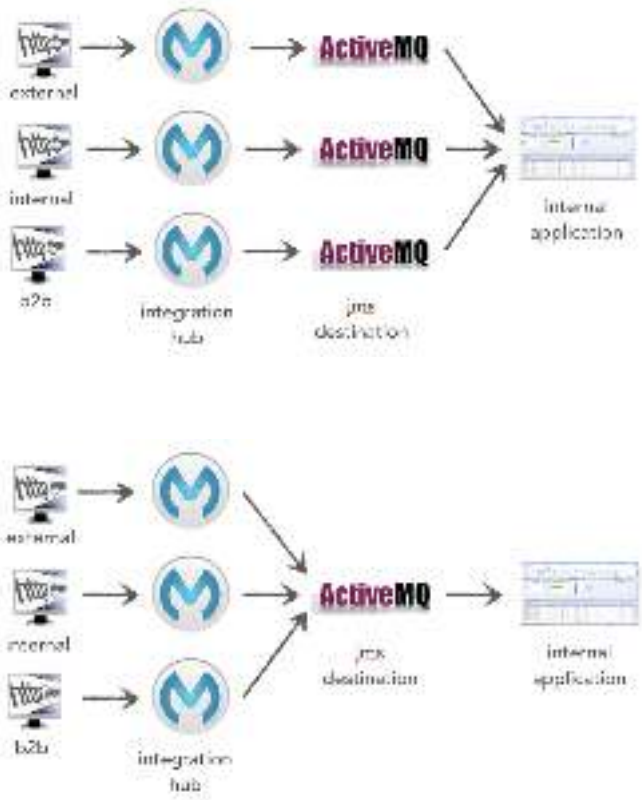
broker only used for hub access

low transaction volumes expected

application logic may be shared between different types of client applications (e.g., internal and external)

justifying decisions

analyze each option based on conditions



considerations:

broker usage and purpose

overall message throughput

internal application coupling

single point of failure

performance bottleneck

justifying decisions

architecture decision:

centralized broker

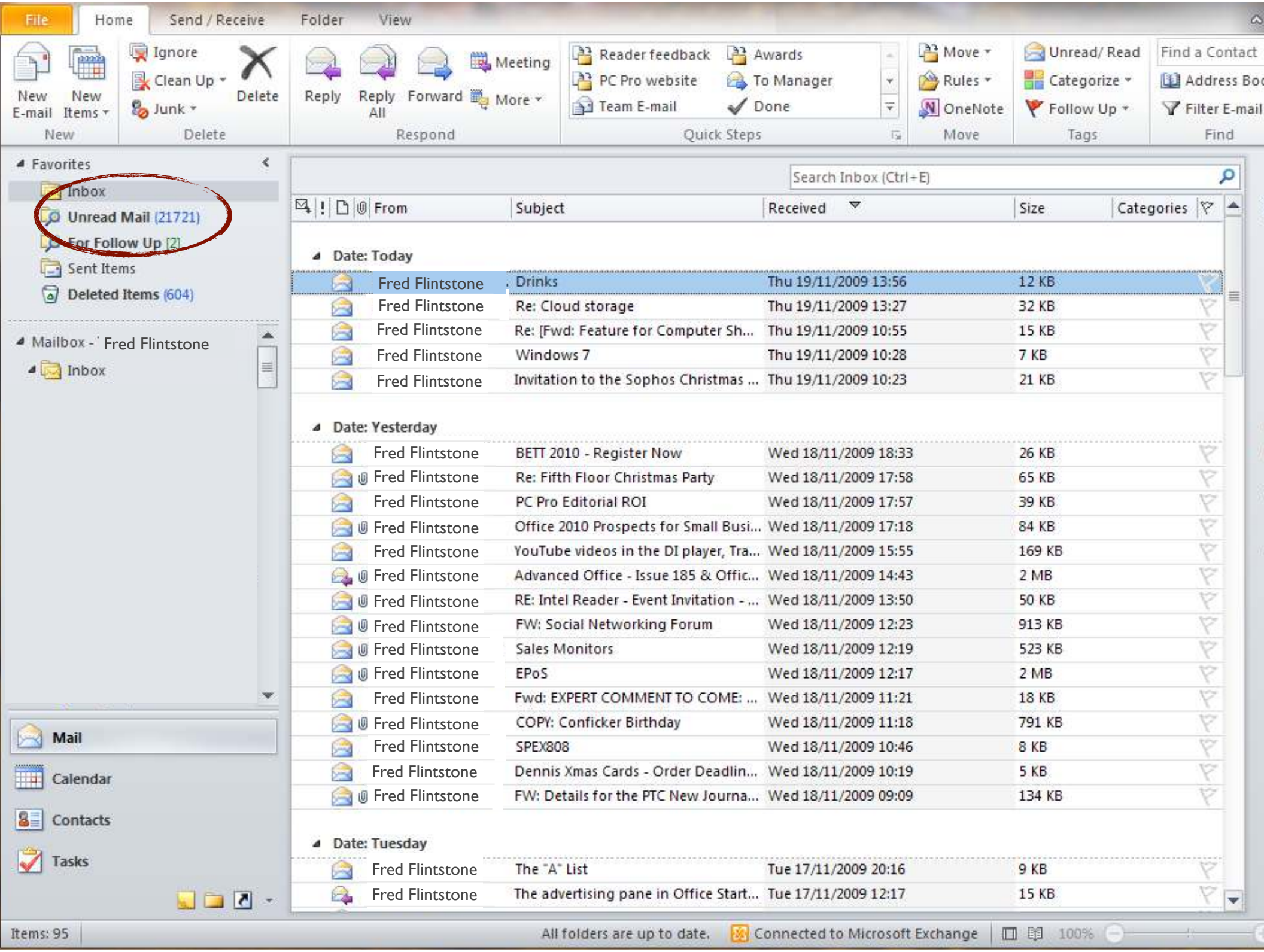
justification:

the internal applications should not have to know from which broker instance the request came from.

only a single broker connection is needed, allowing for the expansion of additional hub instances with no application changes.

due to low request volumes the performance bottleneck is not an issue; single point of failure can be addressed through failover nodes or clustering.

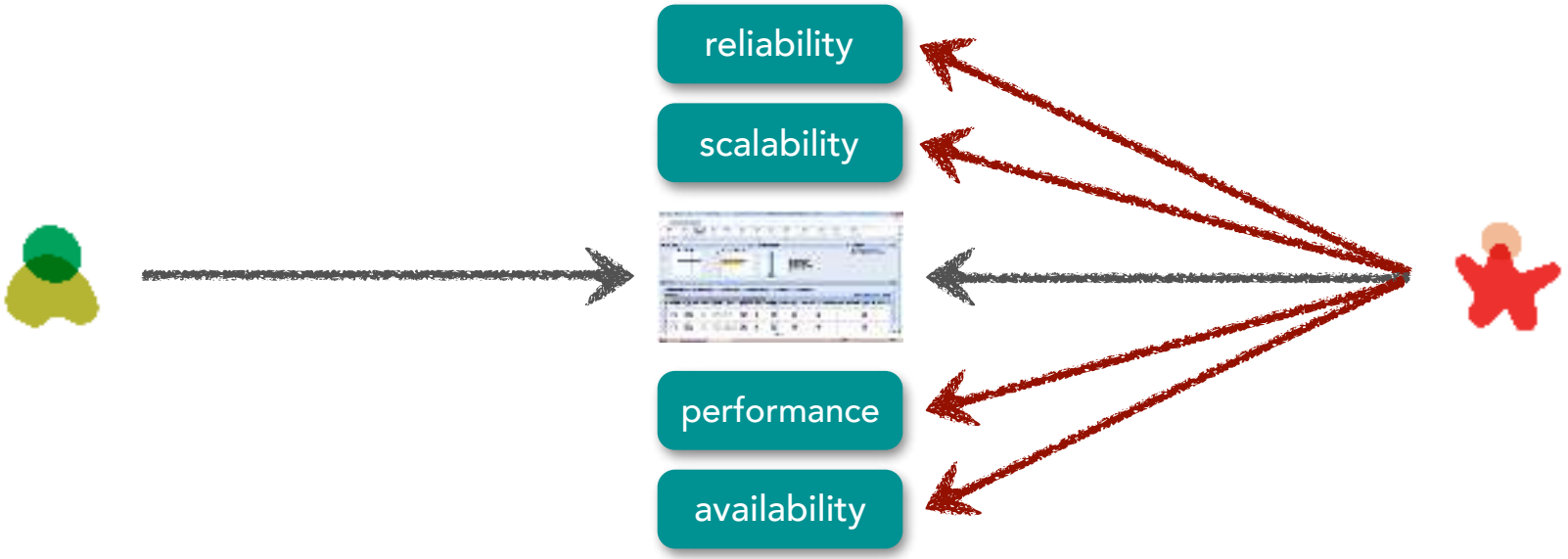
documenting and communicating architecture decisions


File
Home
Send / Receive
Folder
View
New E-mail
New Items
Ignore
Clean Up
Junk
Delete
Reply
Reply All
Forward
Meeting
More
Reader feedback
PC Pro website
Team E-mail
Awards
To Manager
Done
Move
Rules
OneNote
Unread/ Read
Categorize
Follow Up
Find a Contact
Address Boo
Filter E-mail
New
Delete
Respond
Quick Steps
Move
Tags
Find
Favorites
Inbox
Unread Mail (21721)
For Follow Up [2]
Sent Items
Deleted Items (604)
Mailbox - Fred Flintstone
Inbox
Search Inbox (Ctrl+E)
From
Subject
Received
Size
Categories
Date: Today
Fred Flintstone Drinks Thu 19/11/2009 13:56 12 KB
Fred Flintstone Re: Cloud storage Thu 19/11/2009 13:27 32 KB
Fred Flintstone Re: [Fwd: Feature for Computer Sh... Thu 19/11/2009 10:55 15 KB
Fred Flintstone Windows 7 Thu 19/11/2009 10:28 7 KB
Fred Flintstone Invitation to the Sophos Christmas ... Thu 19/11/2009 10:23 21 KB
Date: Yesterday
Fred Flintstone BETT 2010 - Register Now Wed 18/11/2009 18:33 26 KB
Fred Flintstone Re: Fifth Floor Christmas Party Wed 18/11/2009 17:58 65 KB
Fred Flintstone PC Pro Editorial ROI Wed 18/11/2009 17:57 39 KB
Fred Flintstone Office 2010 Prospects for Small Busi... Wed 18/11/2009 17:18 84 KB
Fred Flintstone YouTube videos in the DI player, Tra... Wed 18/11/2009 15:55 169 KB
Fred Flintstone Advanced Office - Issue 185 & Offic... Wed 18/11/2009 14:43 2 MB
Fred Flintstone RE: Intel Reader - Event Invitation - ... Wed 18/11/2009 13:50 50 KB
Fred Flintstone FW: Social Networking Forum Wed 18/11/2009 12:23 913 KB
Fred Flintstone Sales Monitors Wed 18/11/2009 12:19 523 KB
Fred Flintstone EPoS Wed 18/11/2009 12:17 2 MB
Fred Flintstone Fwd: EXPERT COMMENT TO COME: ... Wed 18/11/2009 11:21 18 KB
Fred Flintstone COPY: Conficker Birthday Wed 18/11/2009 11:18 791 KB
Fred Flintstone SPEX808 Wed 18/11/2009 10:46 8 KB
Fred Flintstone Dennis Xmas Cards - Order Deadlin... Wed 18/11/2009 10:19 5 KB
Fred Flintstone FW: Details for the PTC New Journa... Wed 18/11/2009 09:09 134 KB
Date: Tuesday
Fred Flintstone The "A" List Tue 17/11/2009 20:16 9 KB
Fred Flintstone The advertising pane in Office Start... Tue 17/11/2009 12:17 15 KB
Mail
Calendar
Contacts
Tasks
Items: 95
All folders are up to date.
Connected to Microsoft Exchange
100%

identifying architecture characteristics

architecture characteristics

translation skills



System Quality Attributes

accessibility
accountability
accuracy
adaptability
administrability
affordability
agility
auditability
autonomy
availability
compatibility
composability
configurability
correctness
credibility
customizability
debugability
degradability
determinability
demonstrability
dependability
deployability
discoverability
distributability
durability
effectiveness
efficiency
evolvability
extensibility
failure transparency
fault-tolerance
fidelity
flexibility
inspectability
installability
integrity
interchangeability
interoperability
learnability
maintainability
manageability
mobility
modifiability
modularity
operability
orthogonality
portability
precision
predictability
process capabilities
producibility
provability
recoverability
relevance
reliability
repeatability
reproducibility
resilience
responsiveness
reusability
robustness
safety
scalability
seamlessness
self-sustainability
serviceability
supportability
securability
simplicity
stability
standards compliance
survivability
sustainability
tailorability
testability
timeliness
traceability
transparency
ubiquity
understandability
upgradability
usability

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

architecture characteristics

"our business is constantly changing to meet new demands of the marketplace"

???

architecture characteristics

"due to new regulatory requirements, it is imperative that we complete end-of-day processing in time"

???

architecture characteristics

"we need faster time to market to remain competitive"

???

architecture characteristics

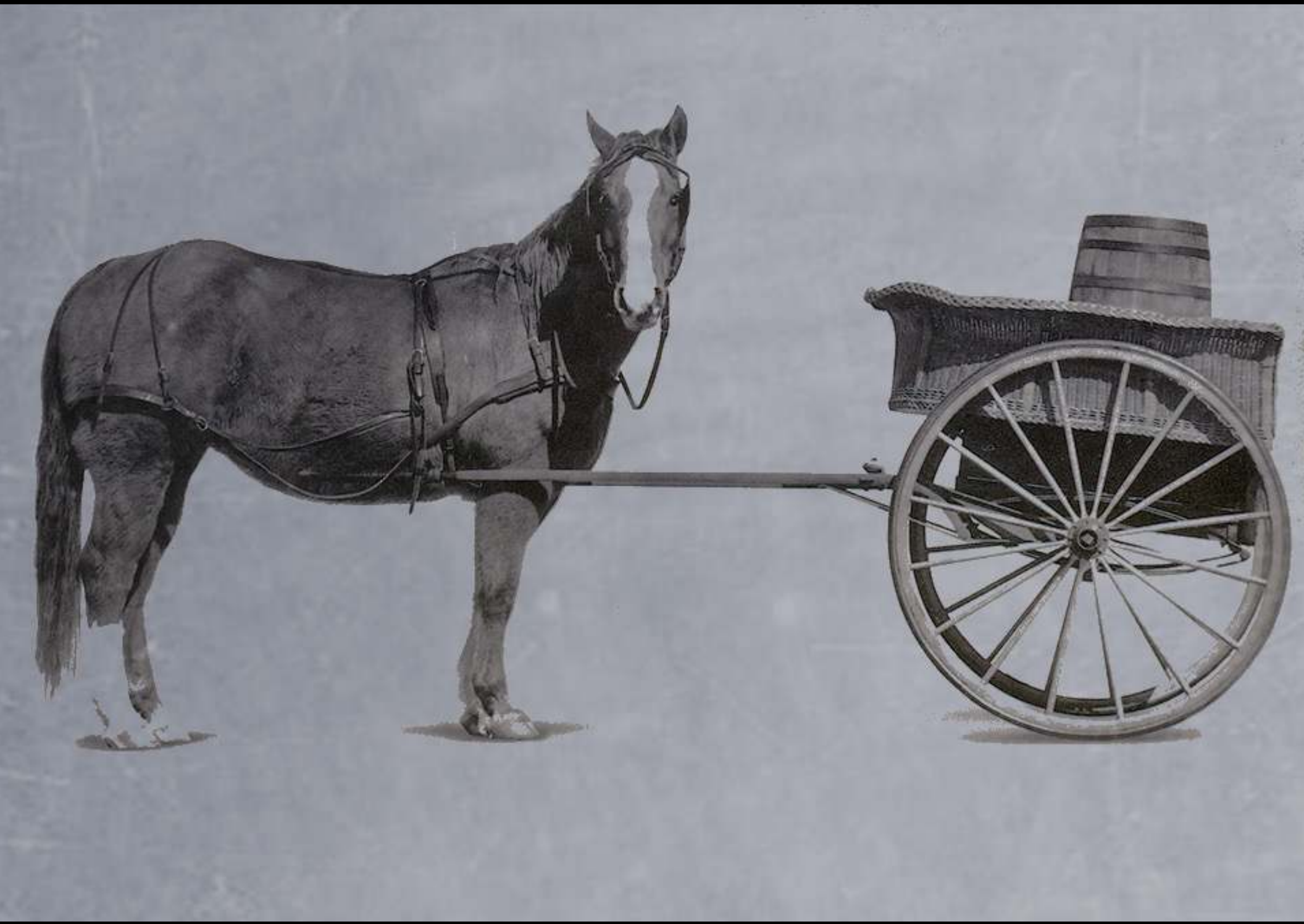
“our plan is to engage heavily in mergers and acquisitions in the next three years”

???

architecture characteristics

"we have a very tight timeframe and budget for this project"

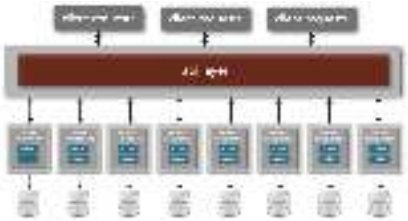
???



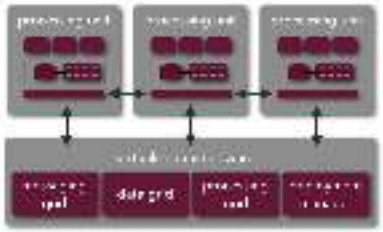
architecture characteristics

feasibility

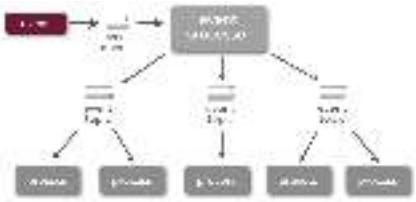
agility



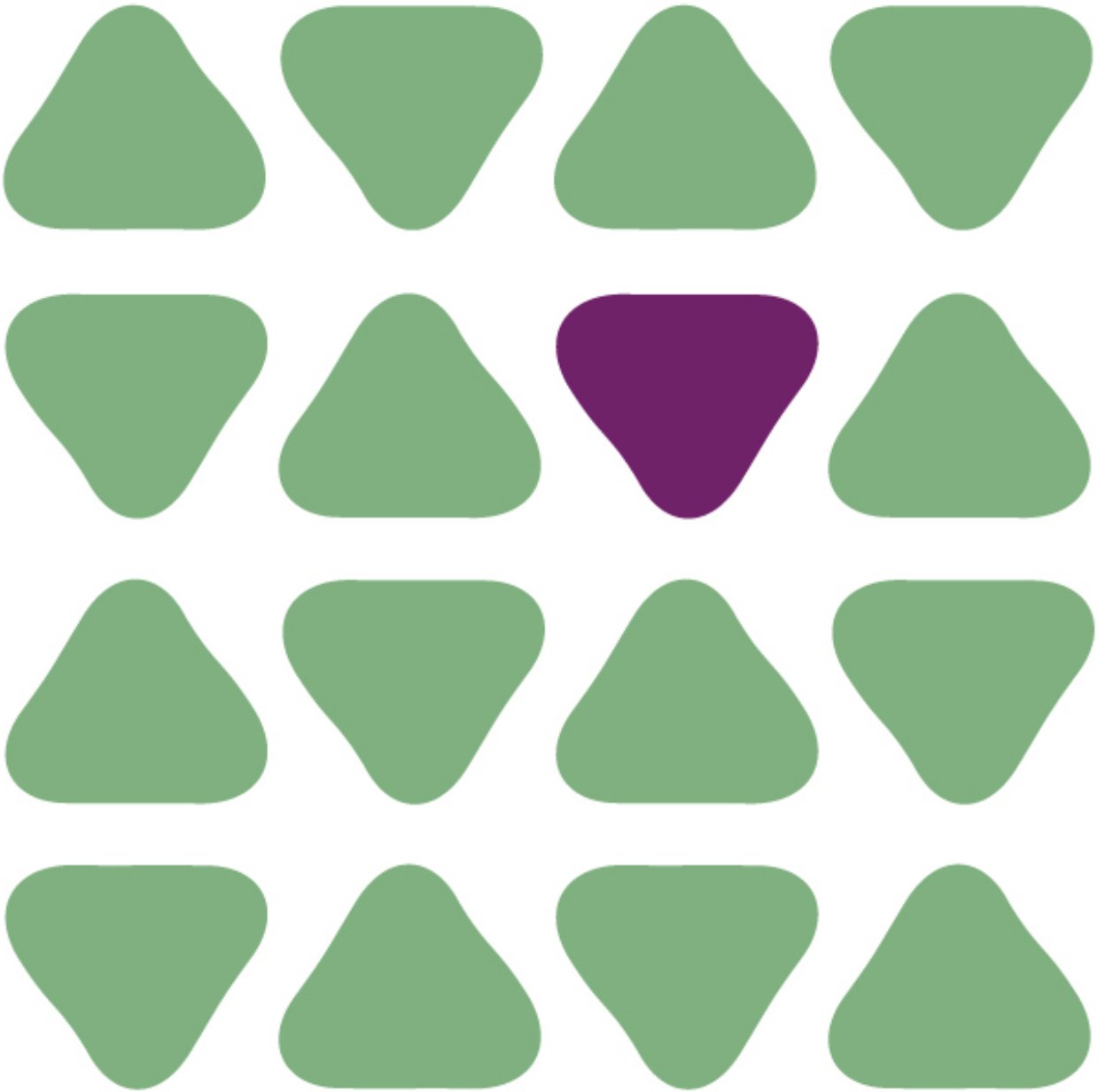
elasticity



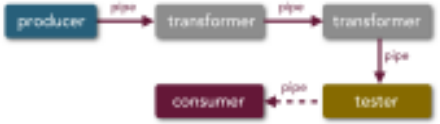
scalability

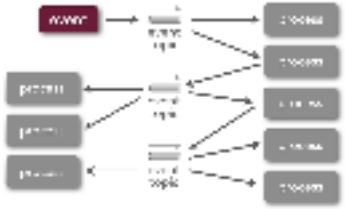


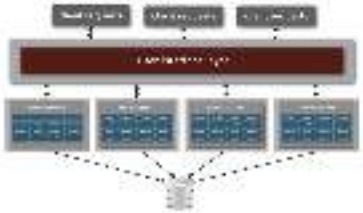
Architecture Patterns

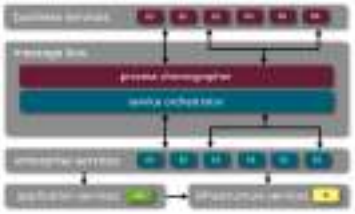


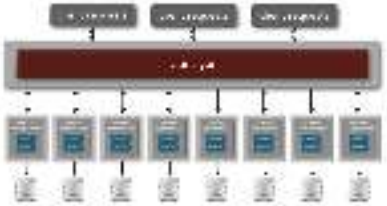
architecture patterns help define the basic characteristics and behavior of the application

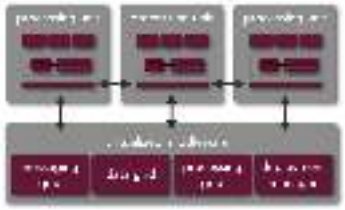












architecture pattern classification

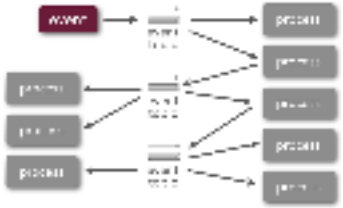
monolithic

layered

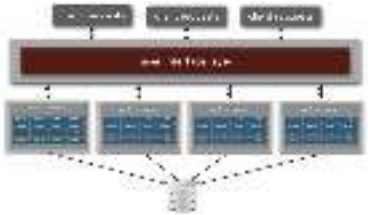
microkernel

pipeline

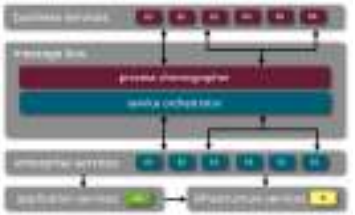
distributed



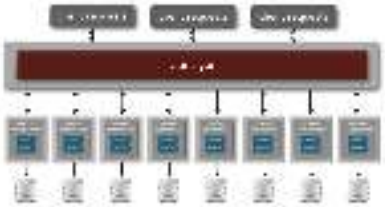
event-driven



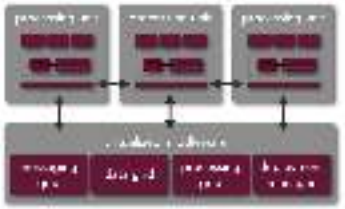
service-based



service-oriented



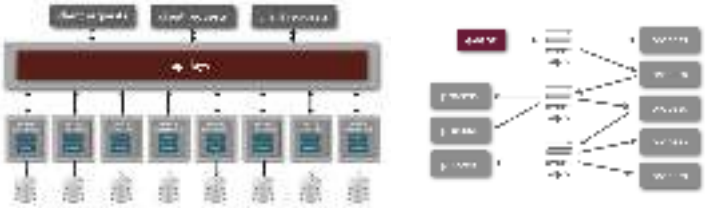
microservices



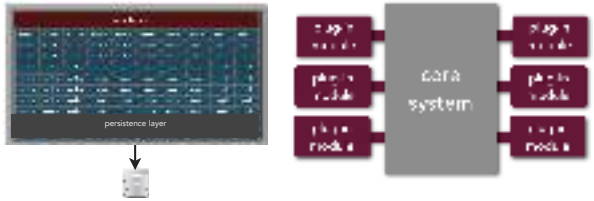
space-based

architecture pattern hybrids

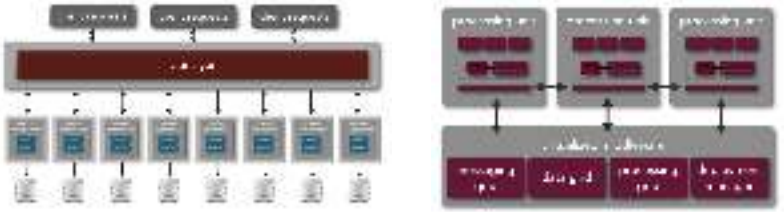
event-driven layered



event-driven microservices



layered microkernel

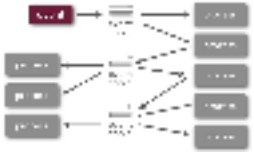


space-based microservices

architecture pattern roadmap

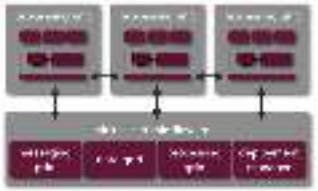
layered architecture

microkernel architecture

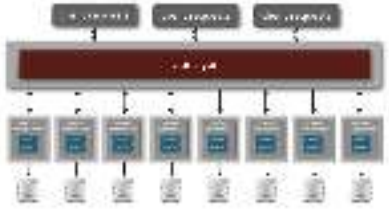


event-driven architecture

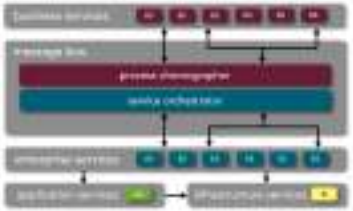
pipeline architecture



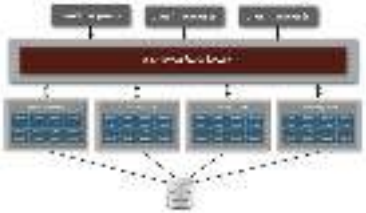
space-based architecture



microservices architecture



service-oriented architecture



service-based architecture

architecture pattern roadmap

layered architecture

microkernel architecture

event-driven architecture

pipeline architecture

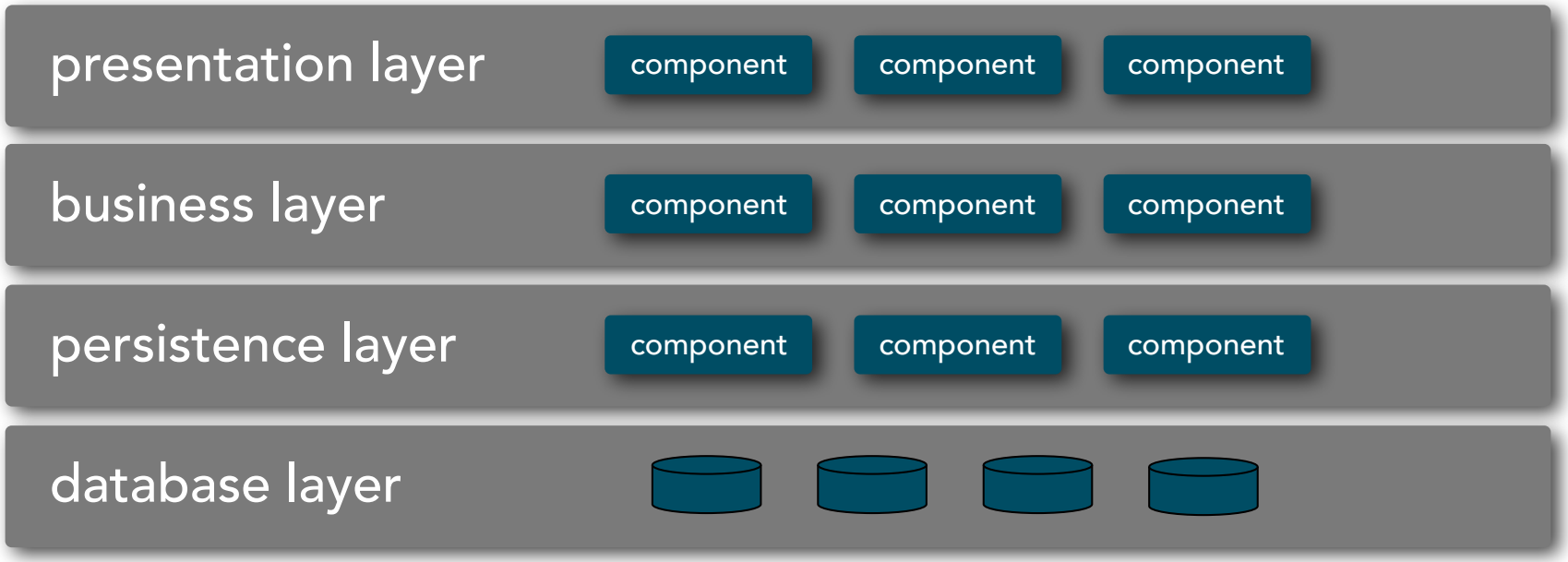
space-based architecture

microservices architecture

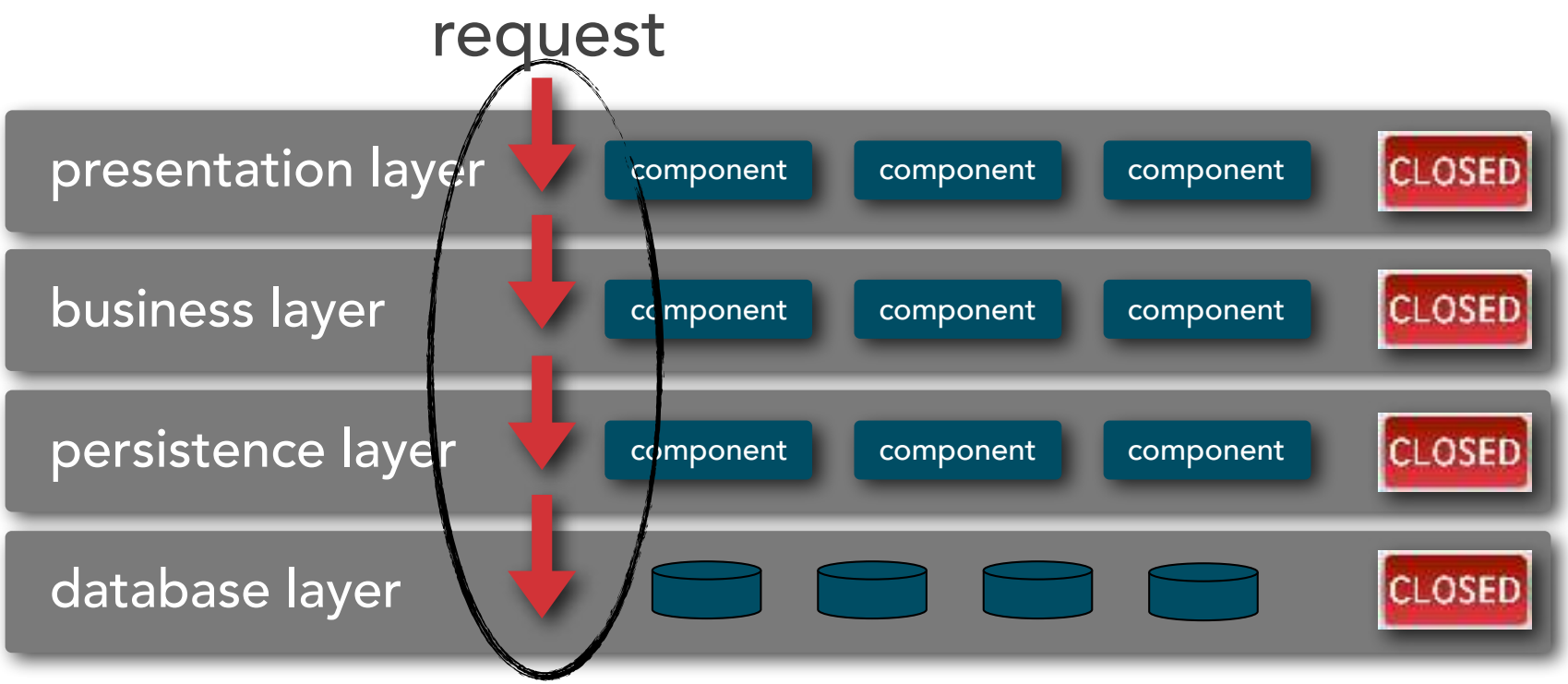
service-oriented architecture

service-based architecture

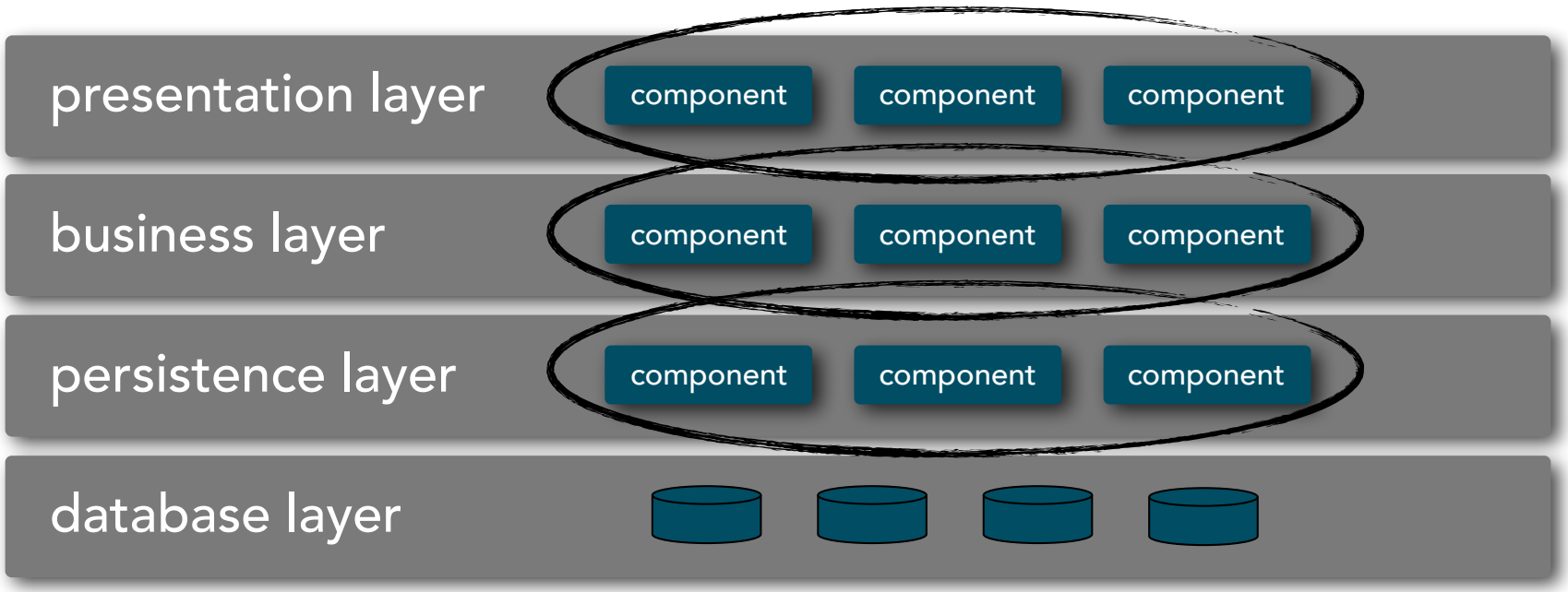
layered architecture



layered architecture

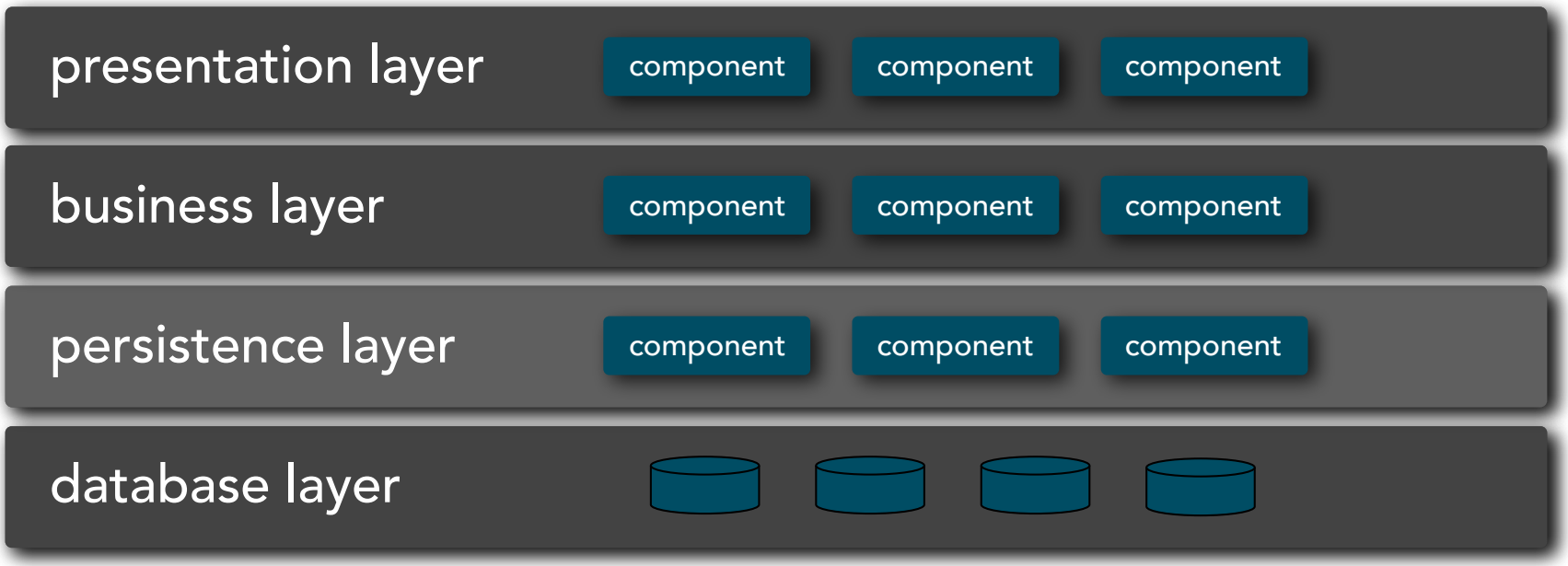


layered architecture



separation of concerns

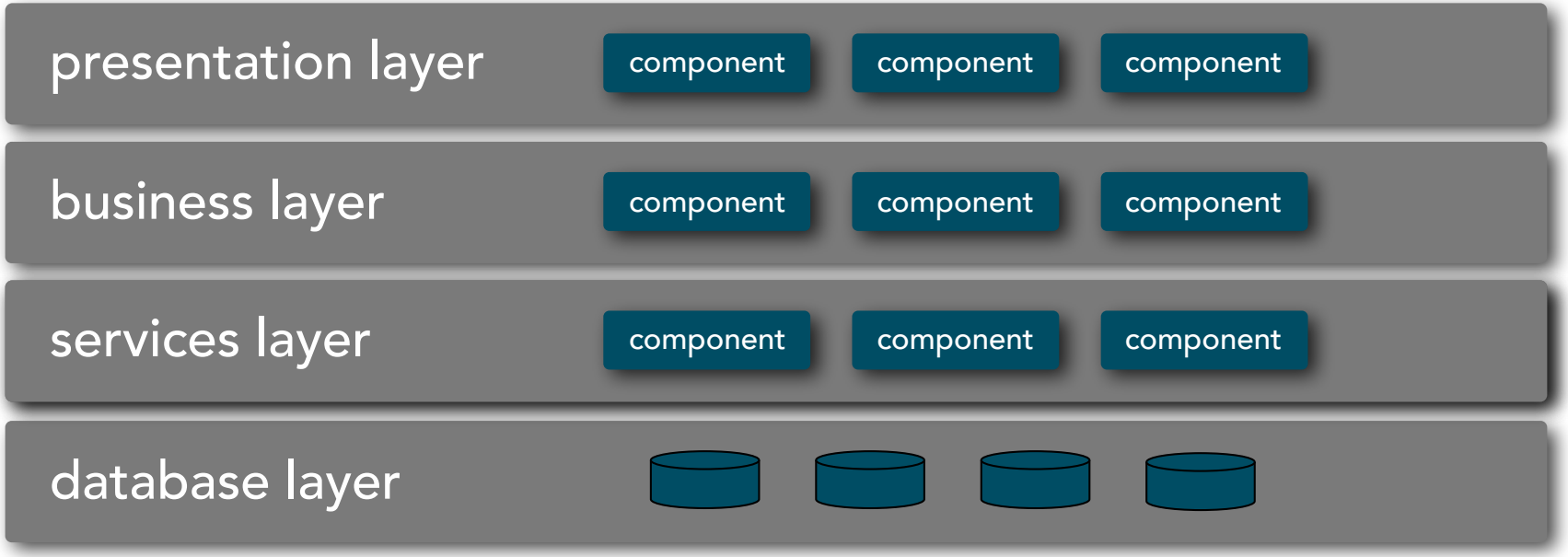
layered architecture



layers of isolation

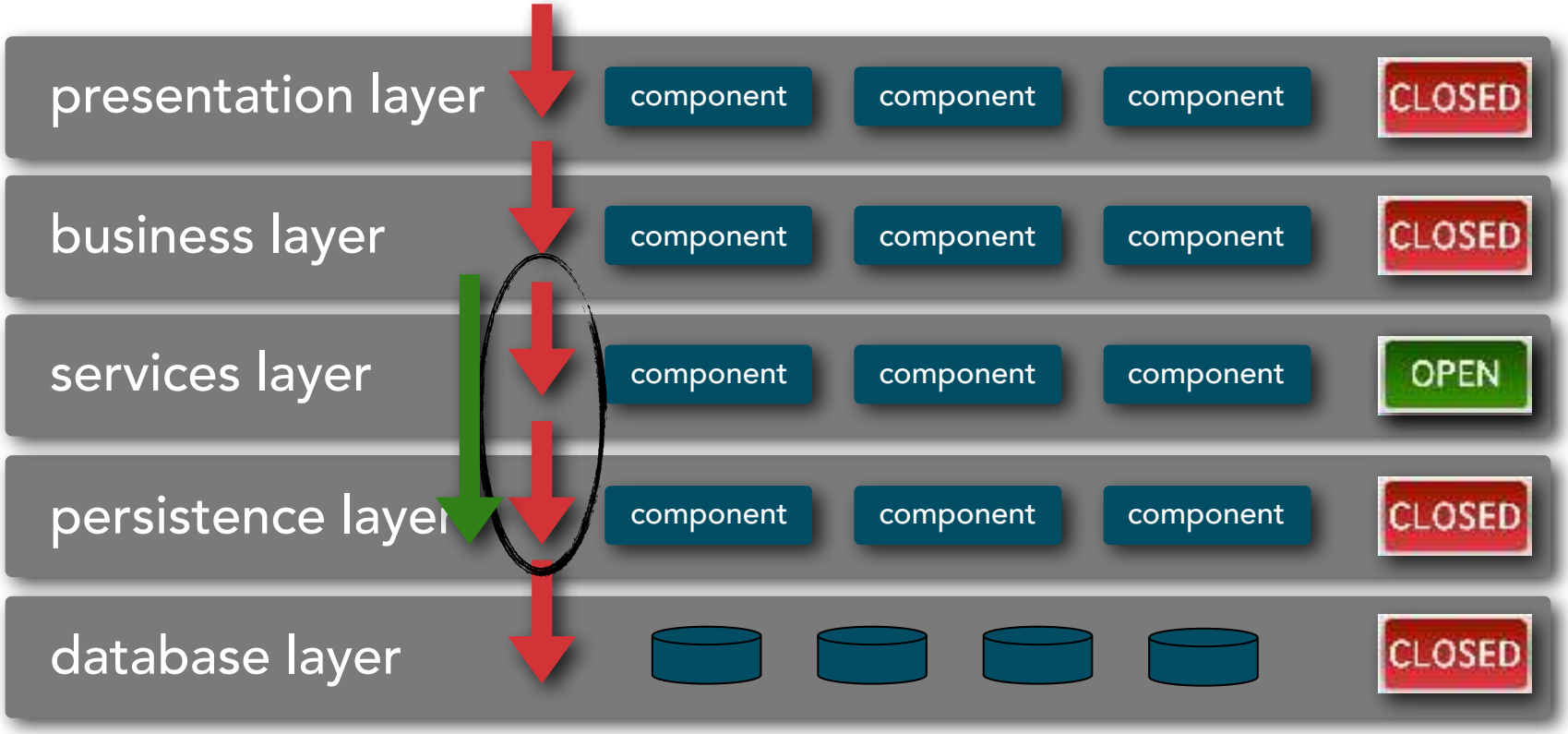
layered architecture

hybrids and variants

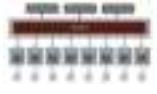


layered architecture

hybrids and variants



layered architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
							
							
							
							
							
							

architecture pattern roadmap

layered architecture

microkernel architecture

event-driven architecture

pipeline architecture

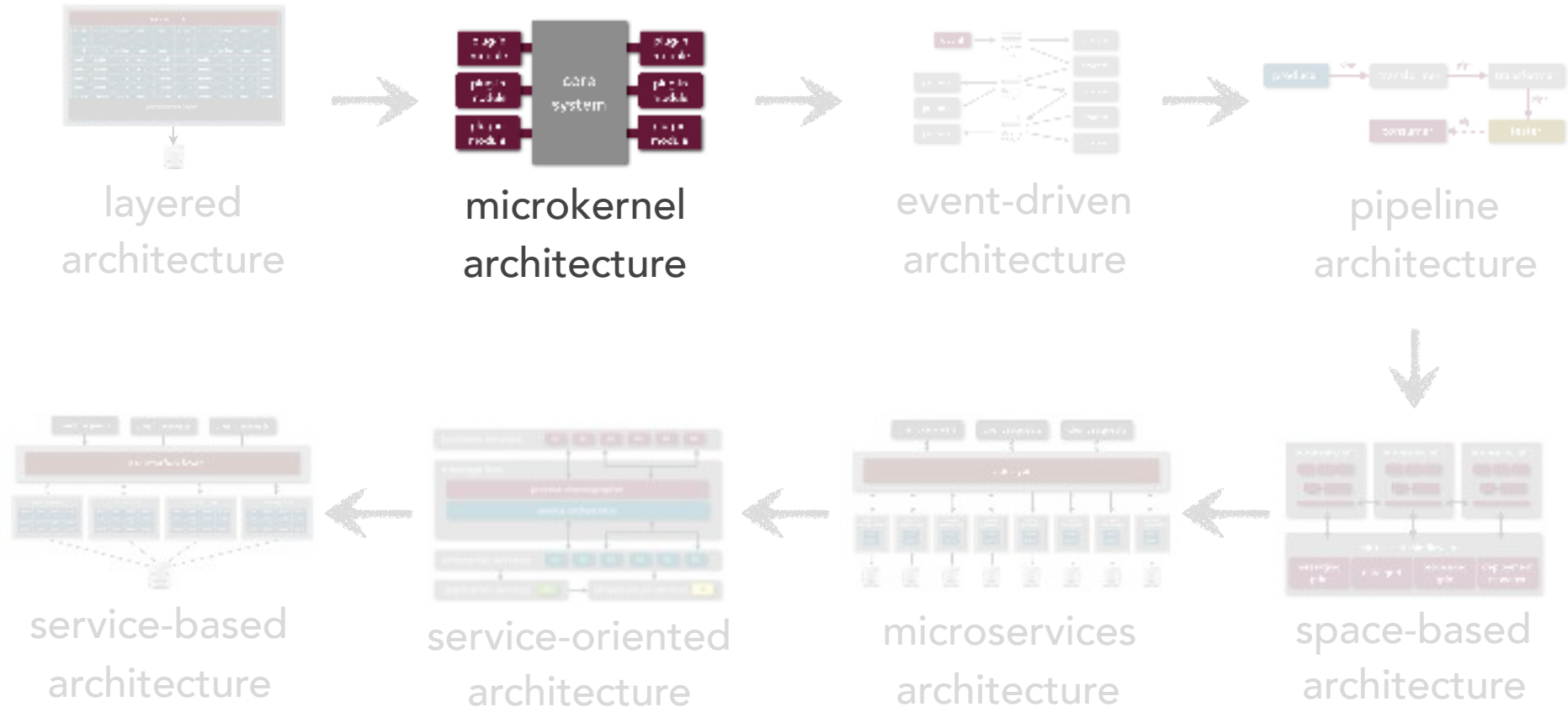
space-based architecture

microservices architecture

service-oriented architecture

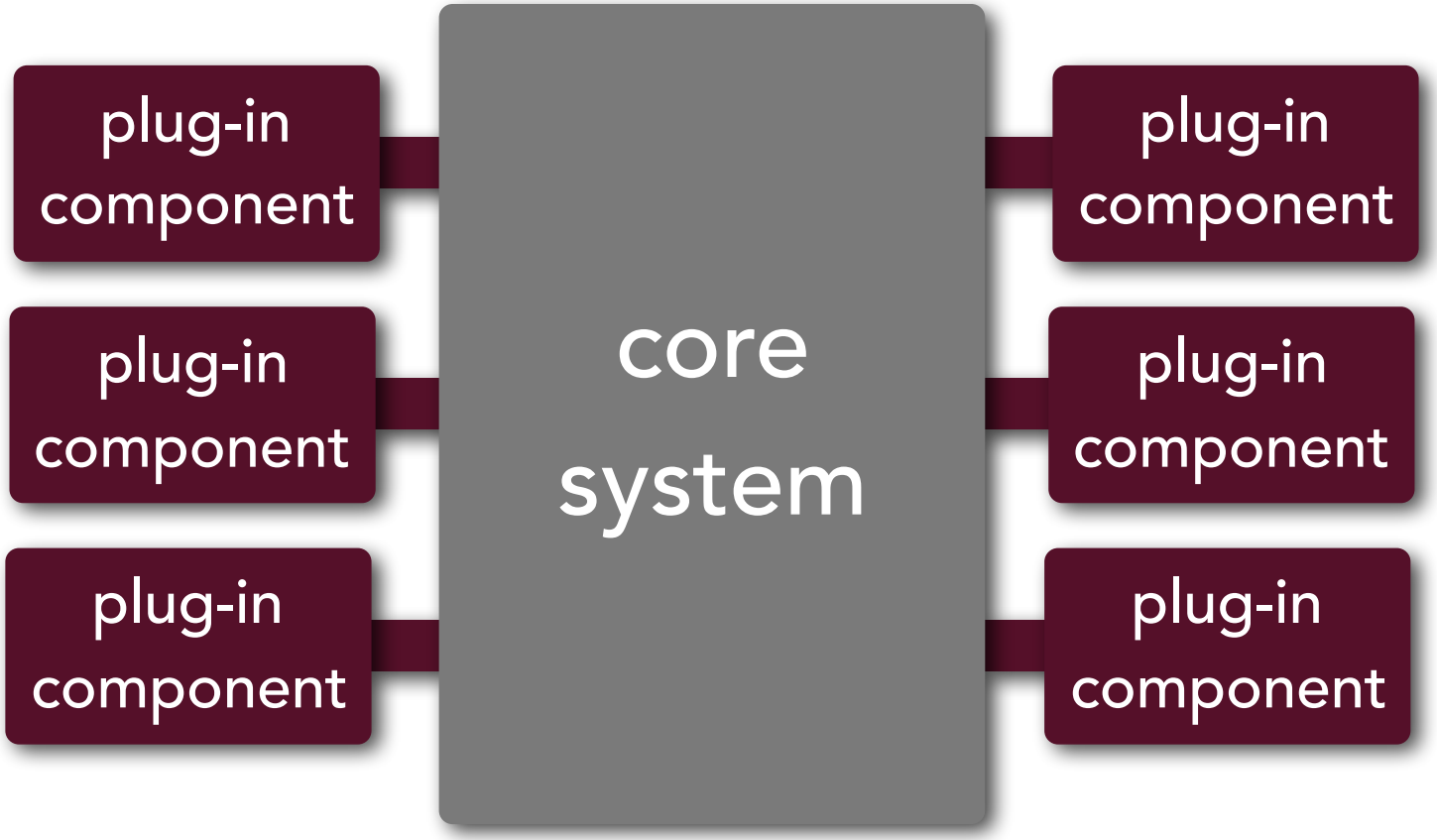
service-based architecture

architecture pattern roadmap



microkernel architecture

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



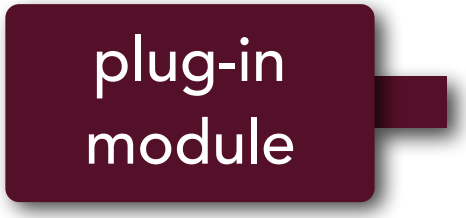
microkernel architecture

architectural components

minimal functionality to run system

general business rules and logic

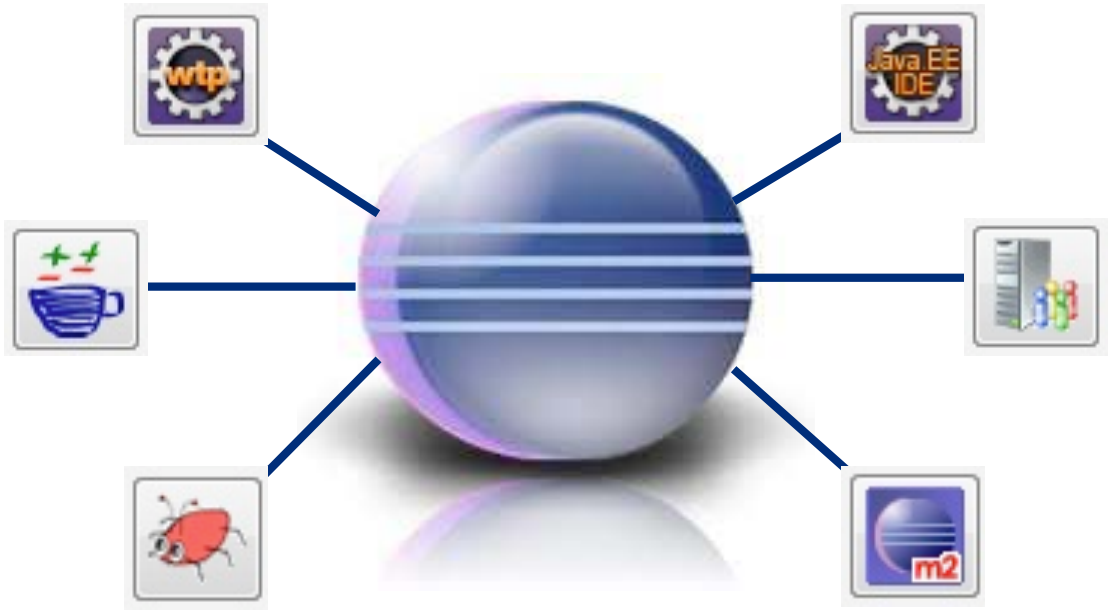
no custom processing



standalone independent module

specific additional rules or logic

microkernel architecture



microkernel architecture

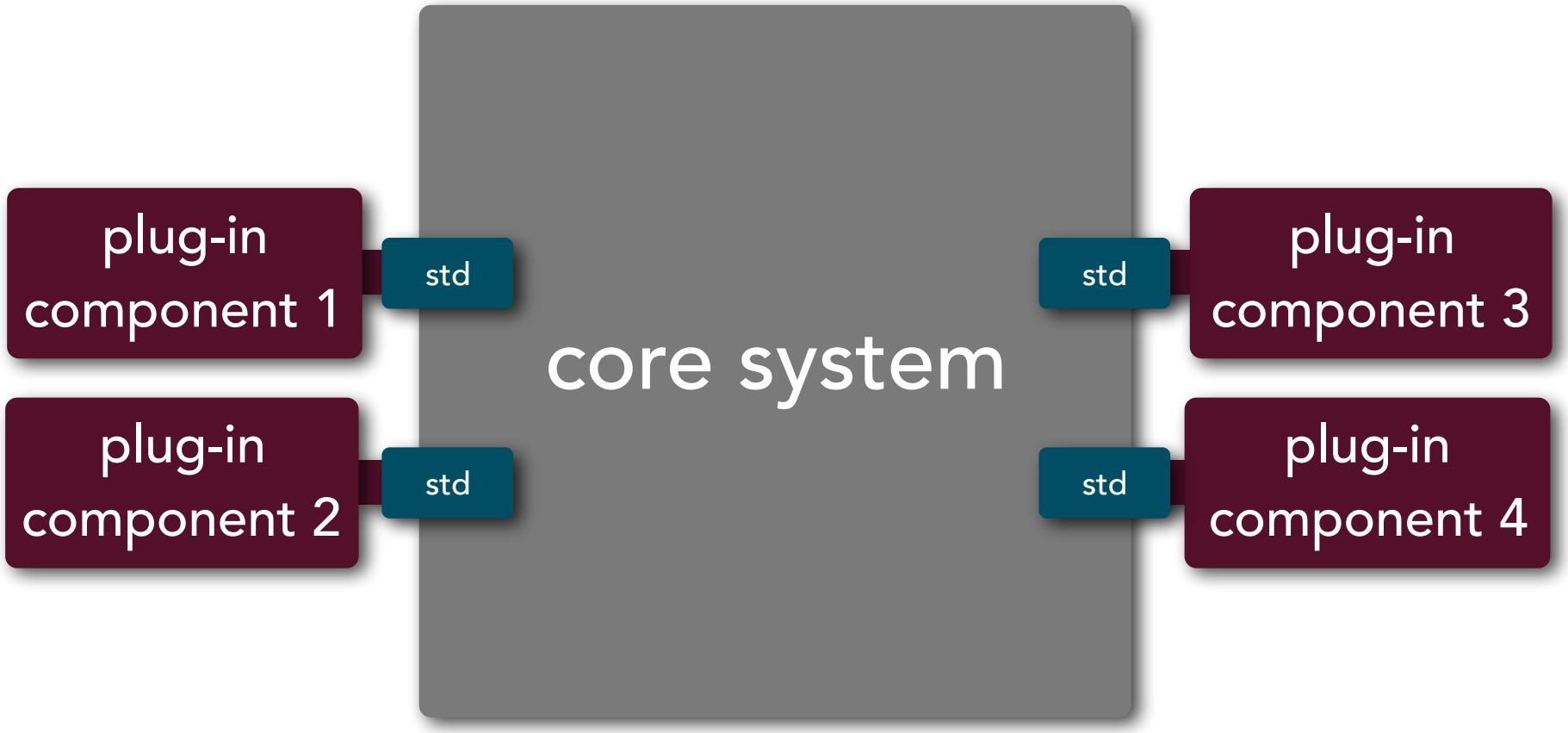
claims processing

microkernel architecture

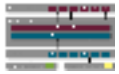
registry

microkernel architecture

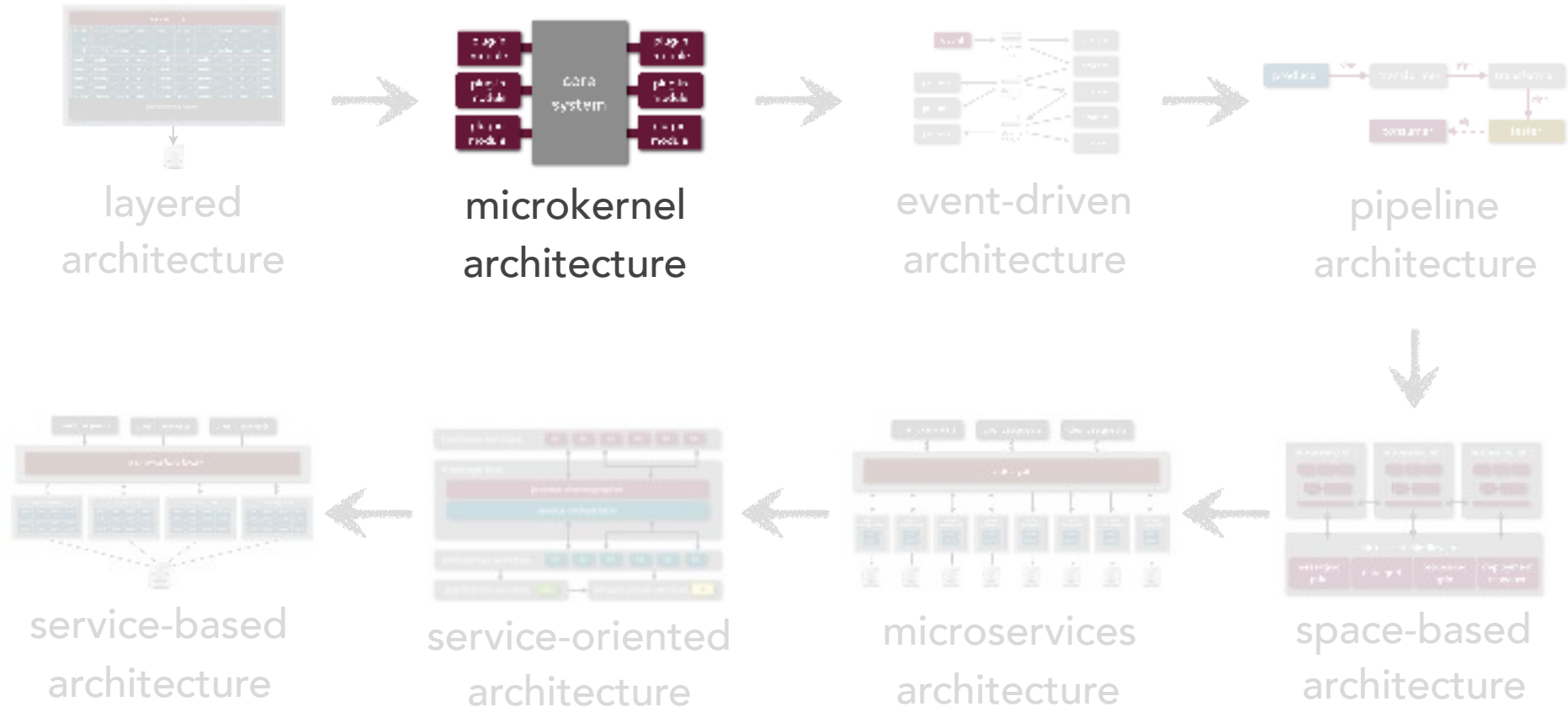
plug-in contracts



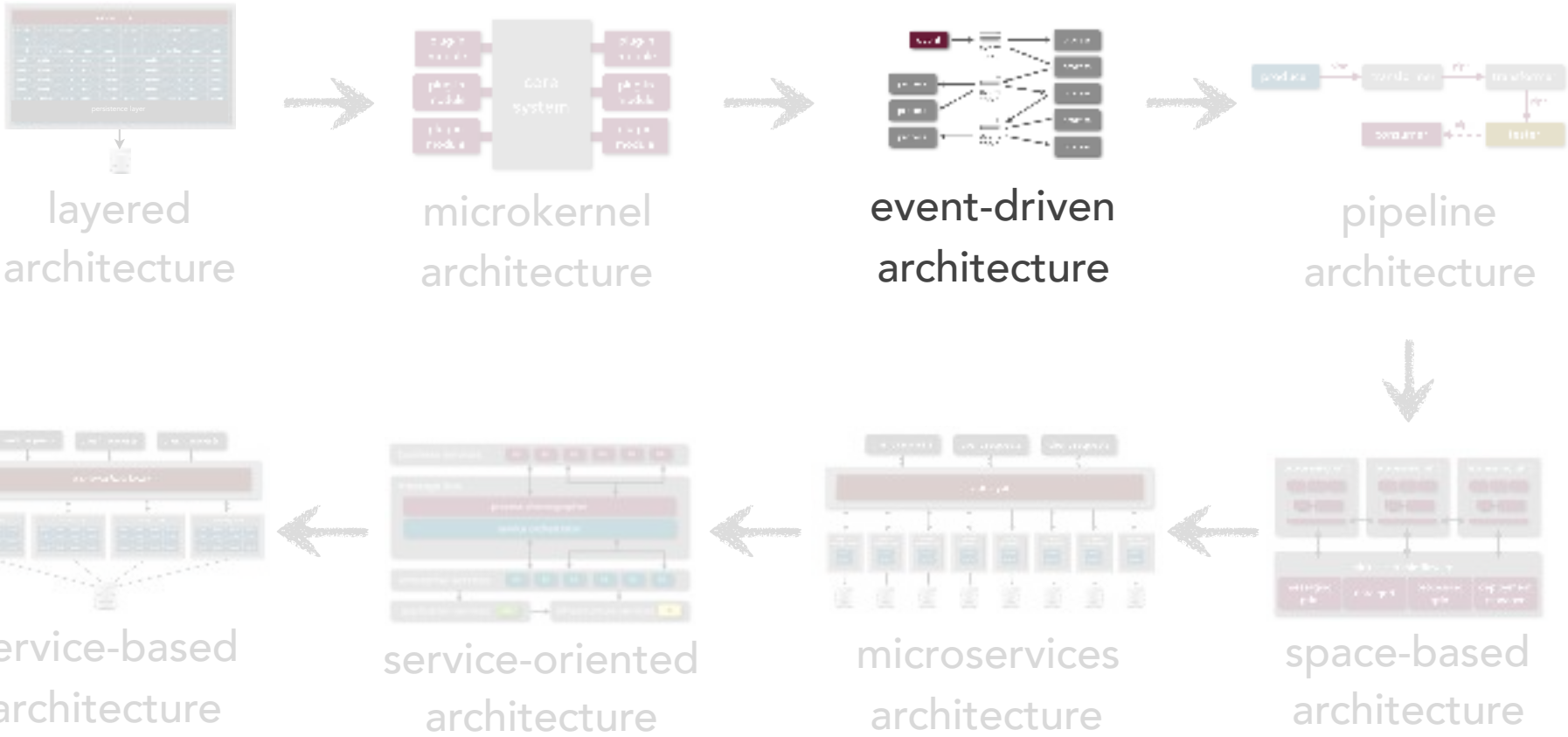
microkernel architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👎	$$
							
							
							
							
							
							

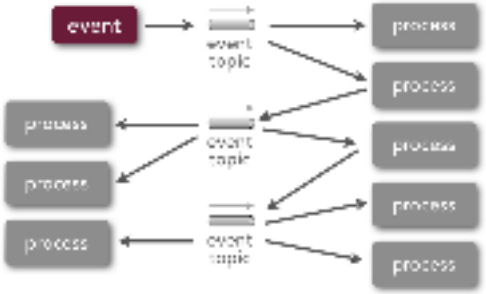
architecture pattern roadmap



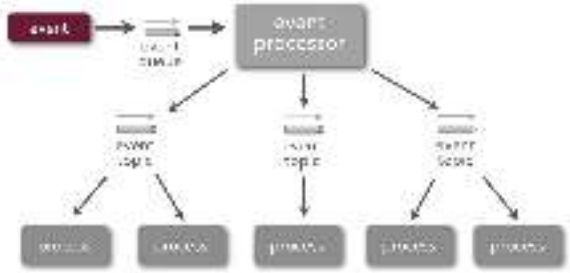
architecture pattern roadmap



event-driven architecture



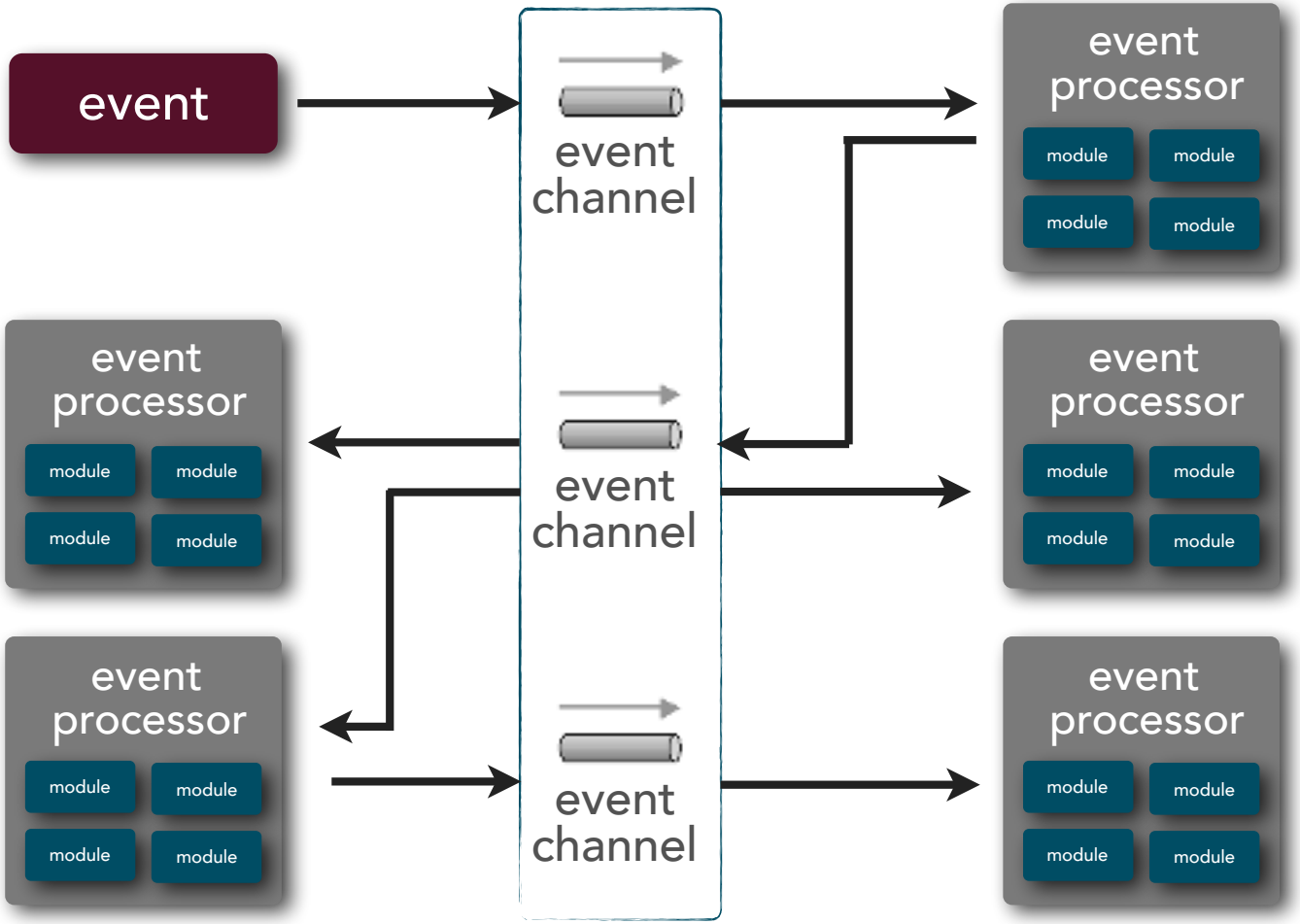
broker topology



mediator topology

event-driven architecture

broker topology



event-driven architecture

you move...

you moved!

customer process

change address

change address

quote process

recalc quote

claims process

update claims

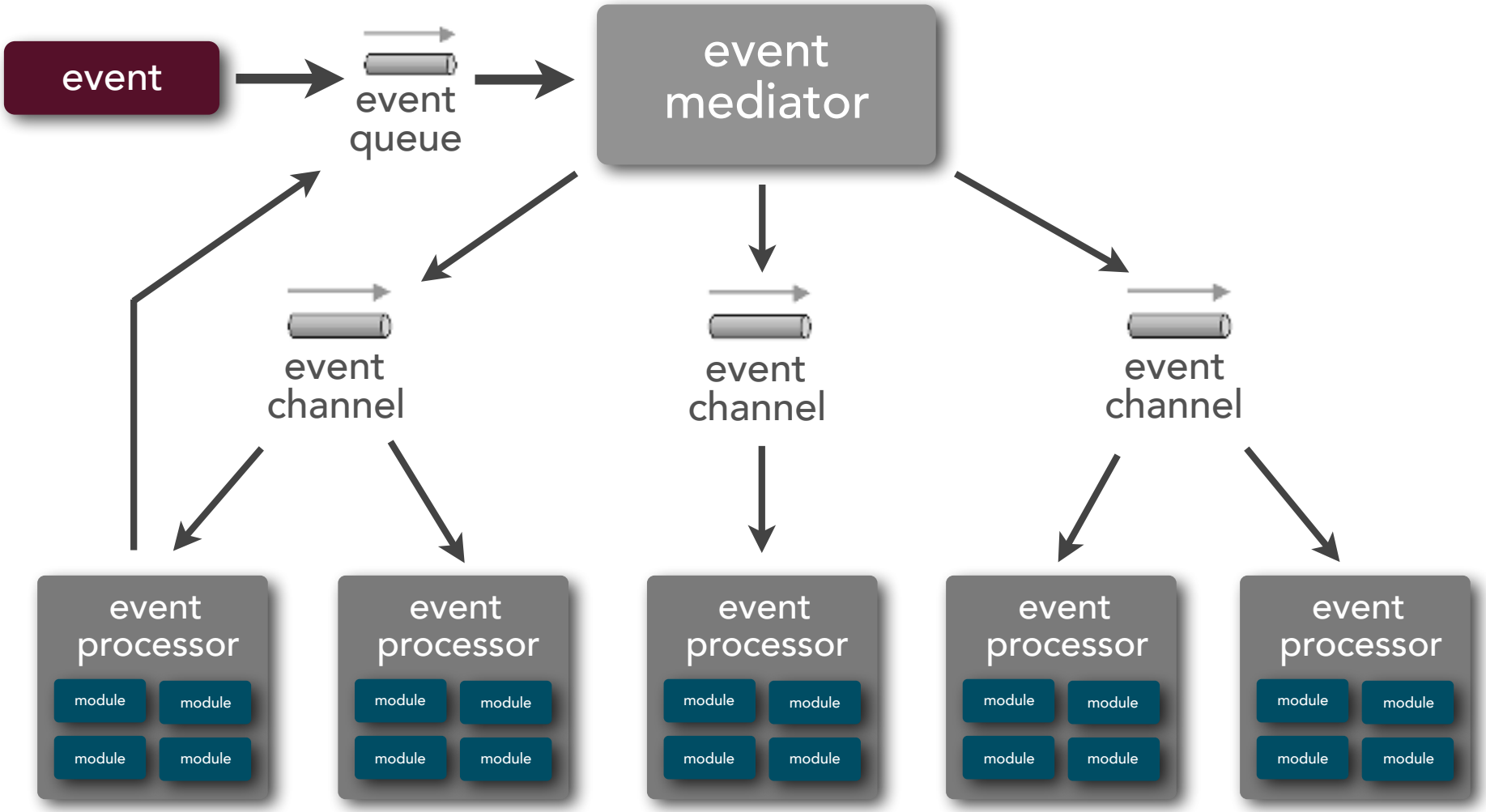
update claims

notification process

adjustment process

event-driven architecture

mediator topology



event-driven architecture

you move...

you moved

process engine

change address

recalc quote

update claims

adjust claims

notify insured

customer process

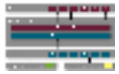
quote process

claims process

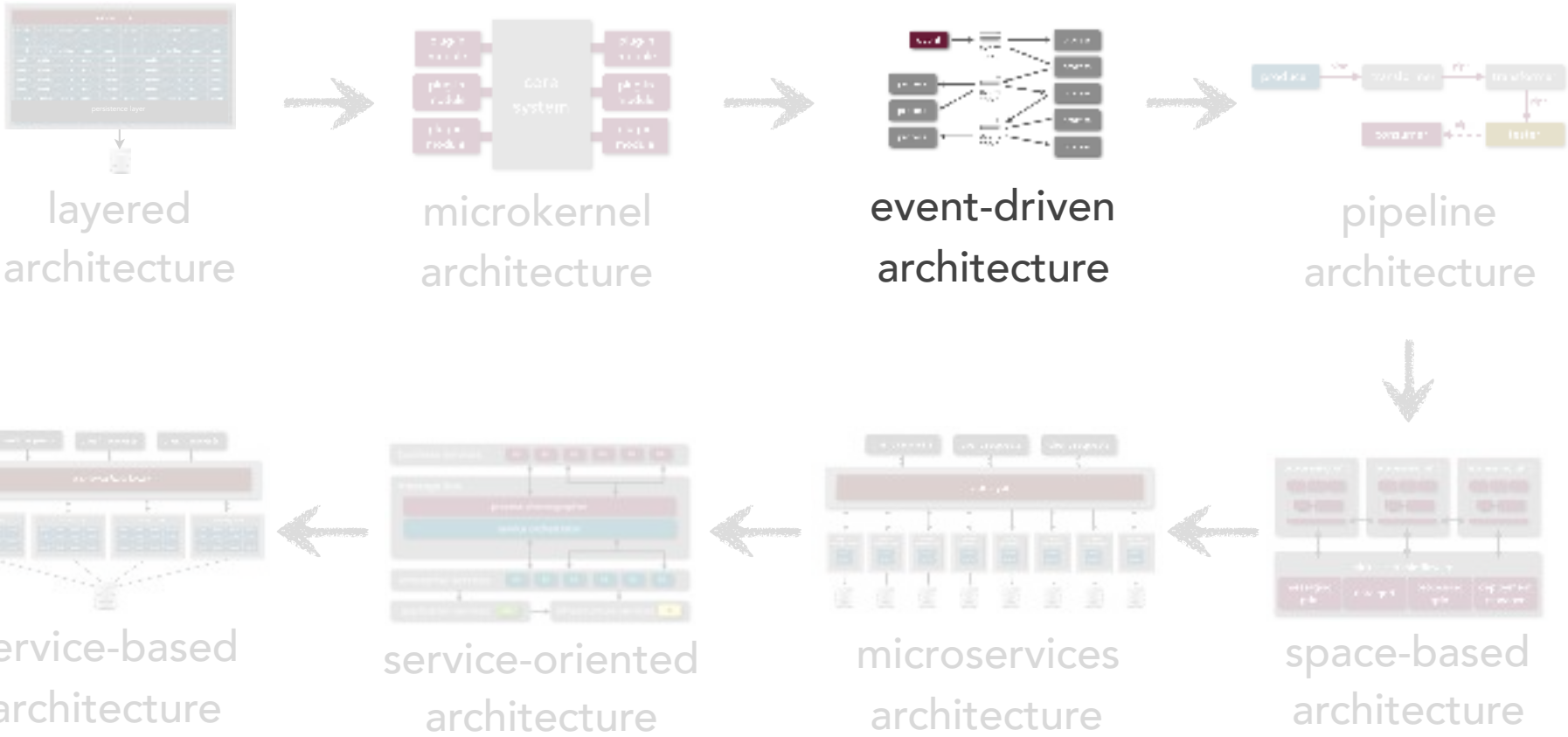
adjustment process

notification process

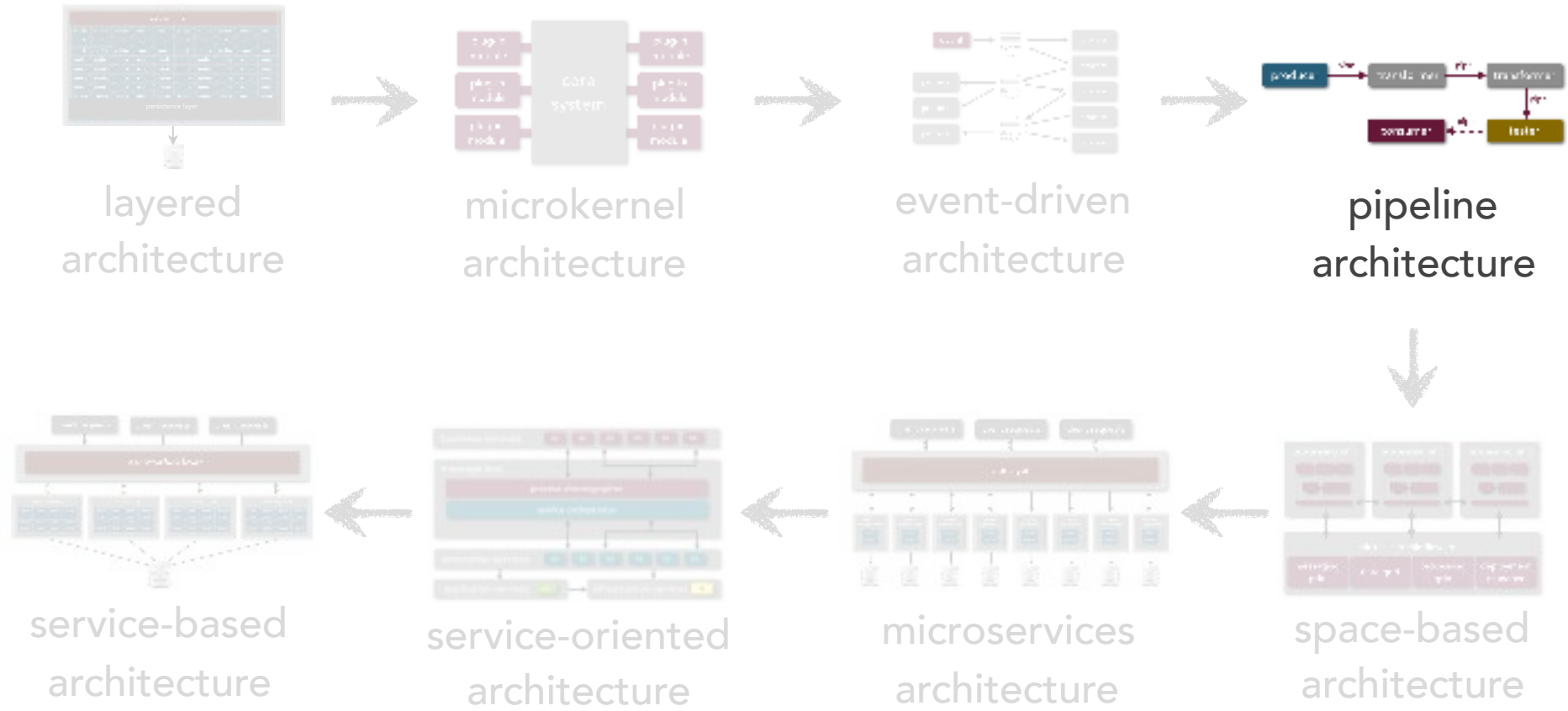
event-driven architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👎	$$
	👍	👍	👎	👍	👍	👎	$$$
							
							
							
							
							

architecture pattern roadmap

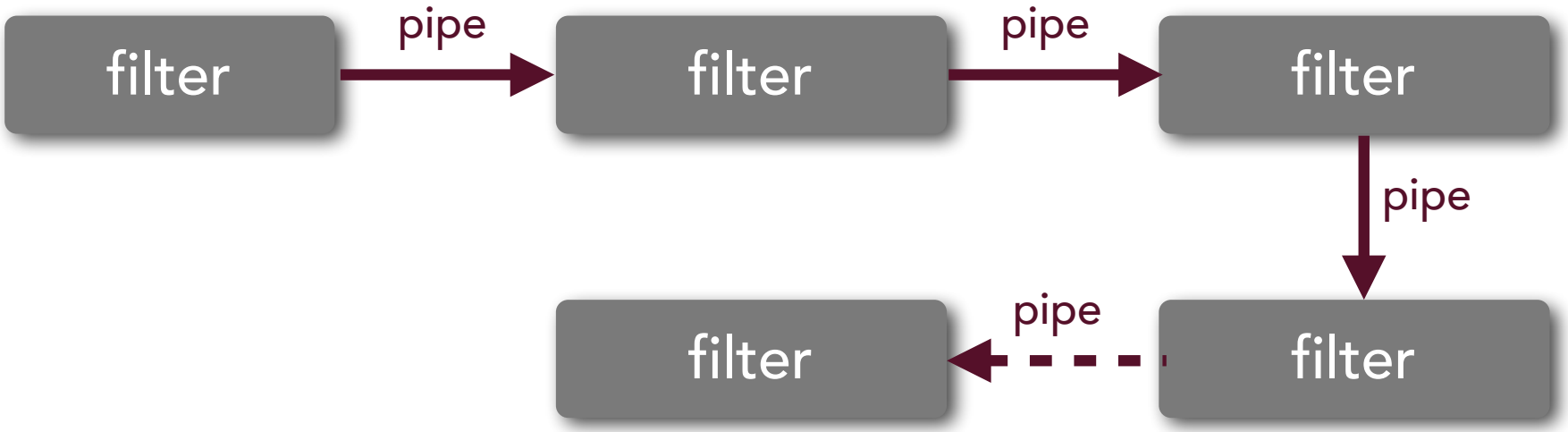


architecture pattern roadmap



pipeline architecture

(a.k.a. pipe and filter architecture)



pipeline architecture

pipes

uni-directional only

usually point-to-point for high performance, but could be message-based for scalability

payload can be any type (text, bytes, object, etc.)

pipeline architecture

filters

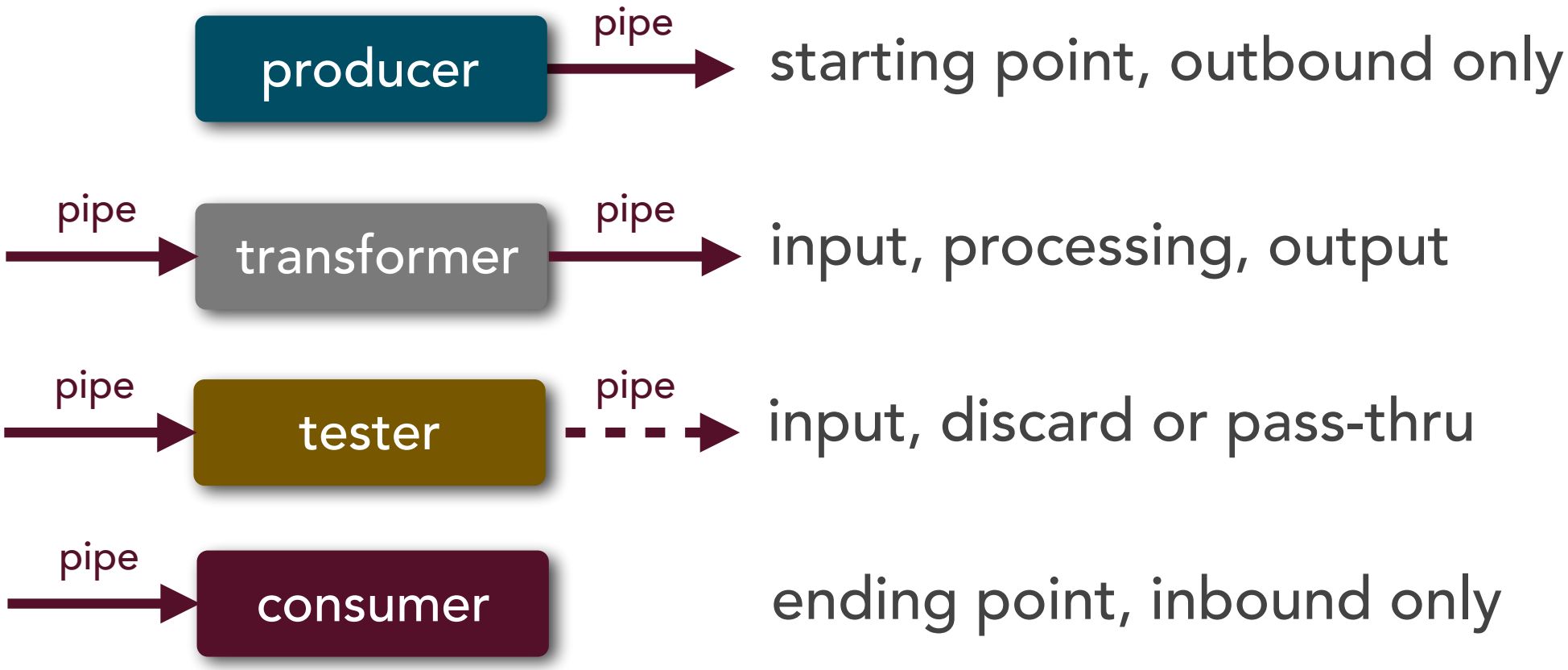
self-contained and independent from other filters

usually designed to perform a single specific task

four filter types (producer, consumer, transformer, and tester

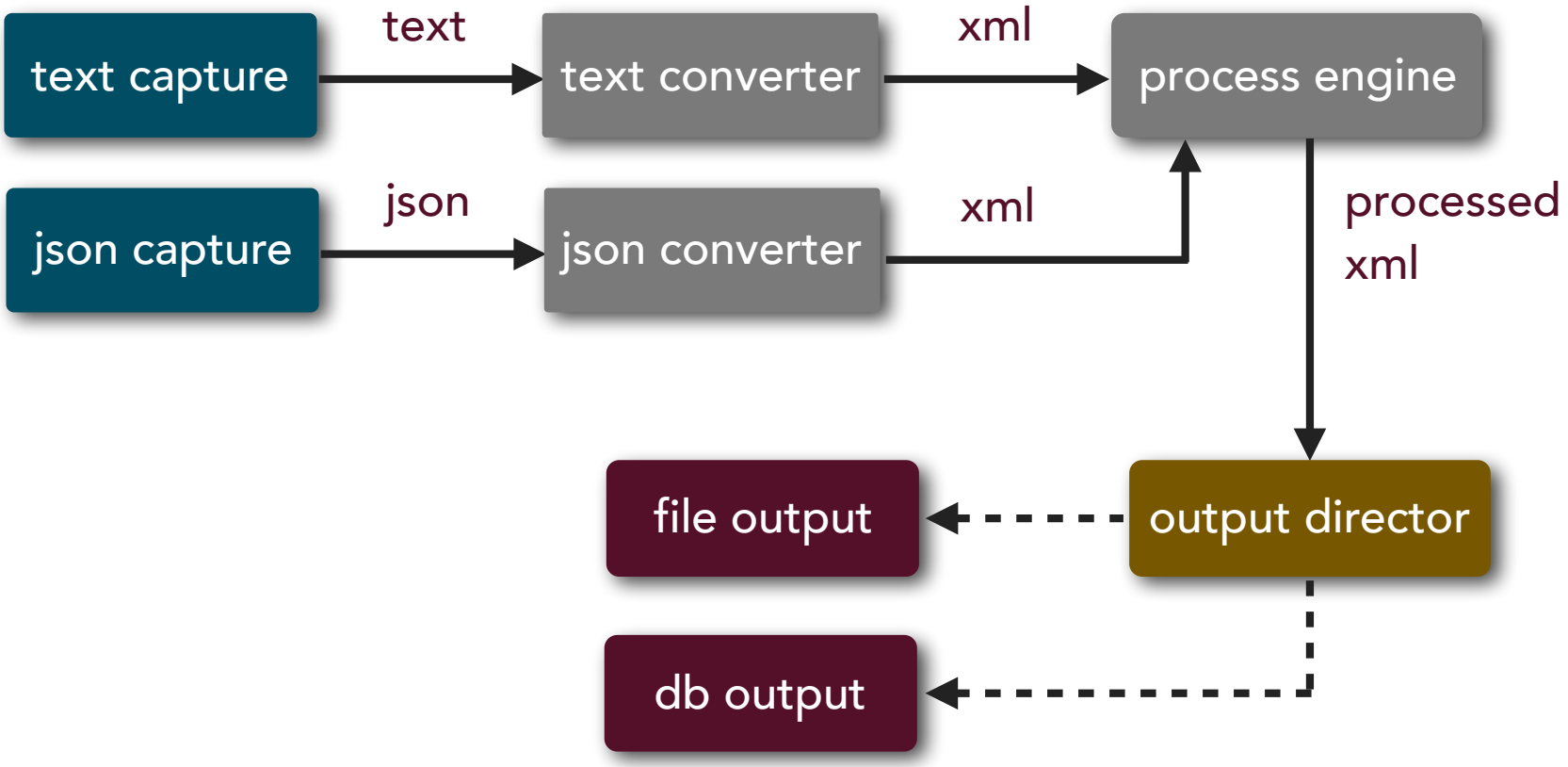
pipeline architecture

filters

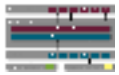


pipeline architecture

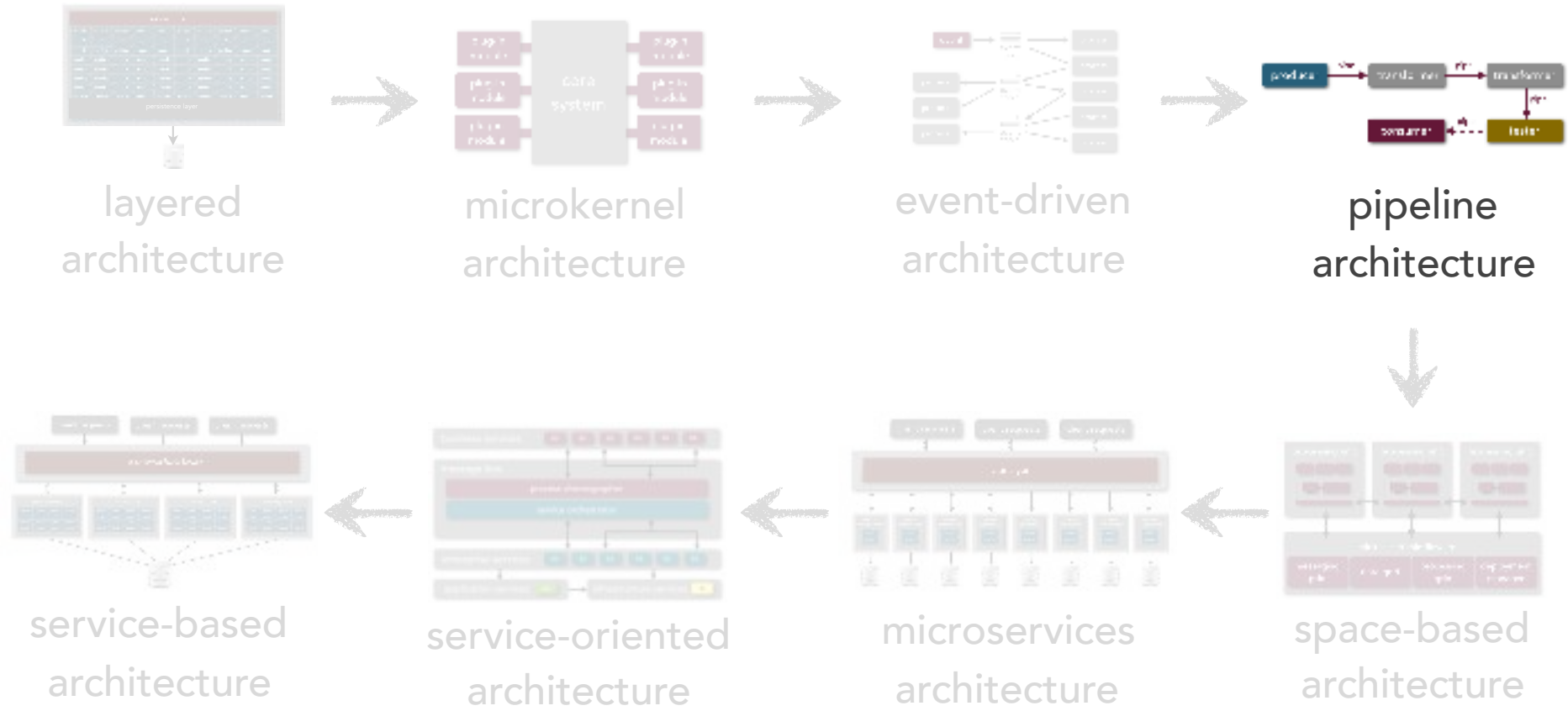
example: capture data in multiple formats, process the data, and send to multiple outputs



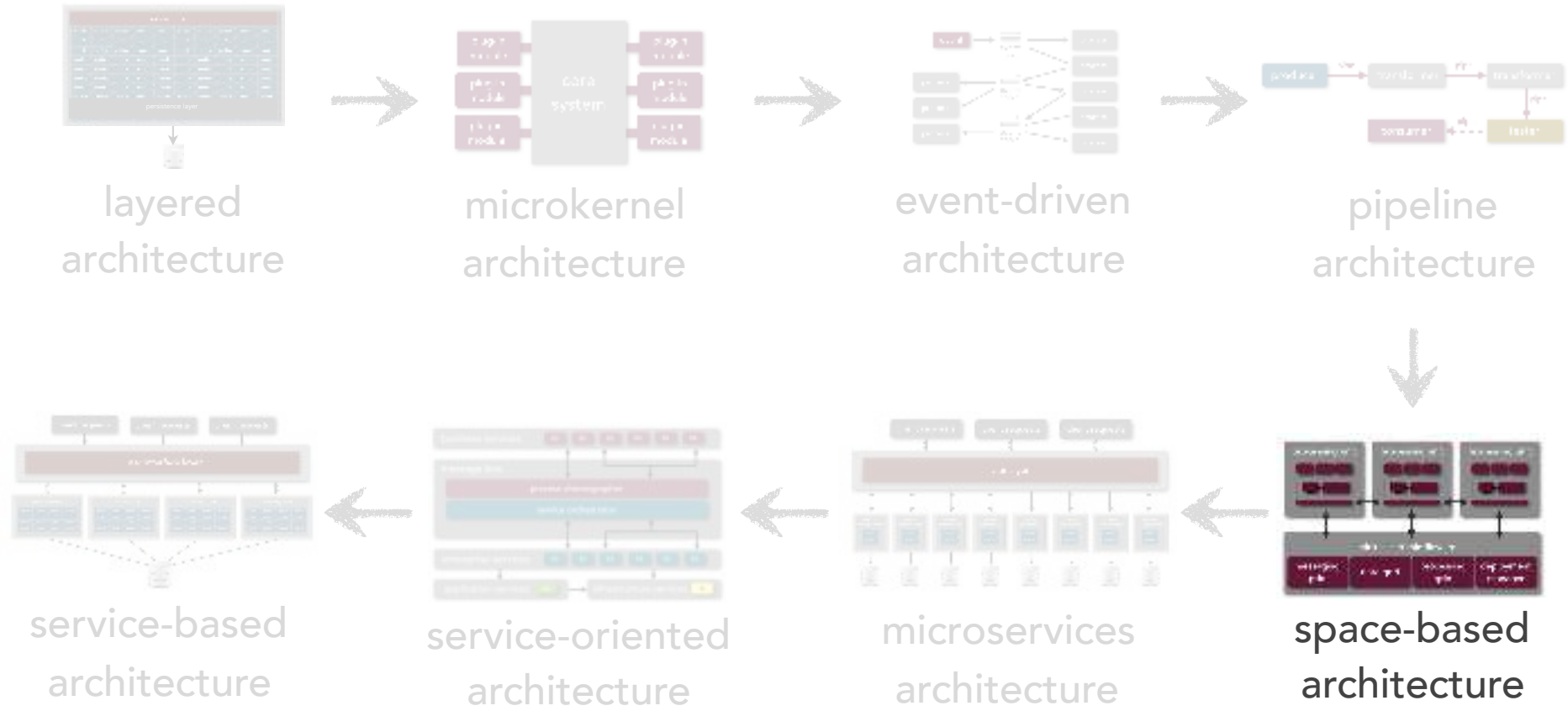
pipeline architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👎	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
							
							
							
							

architecture pattern roadmap

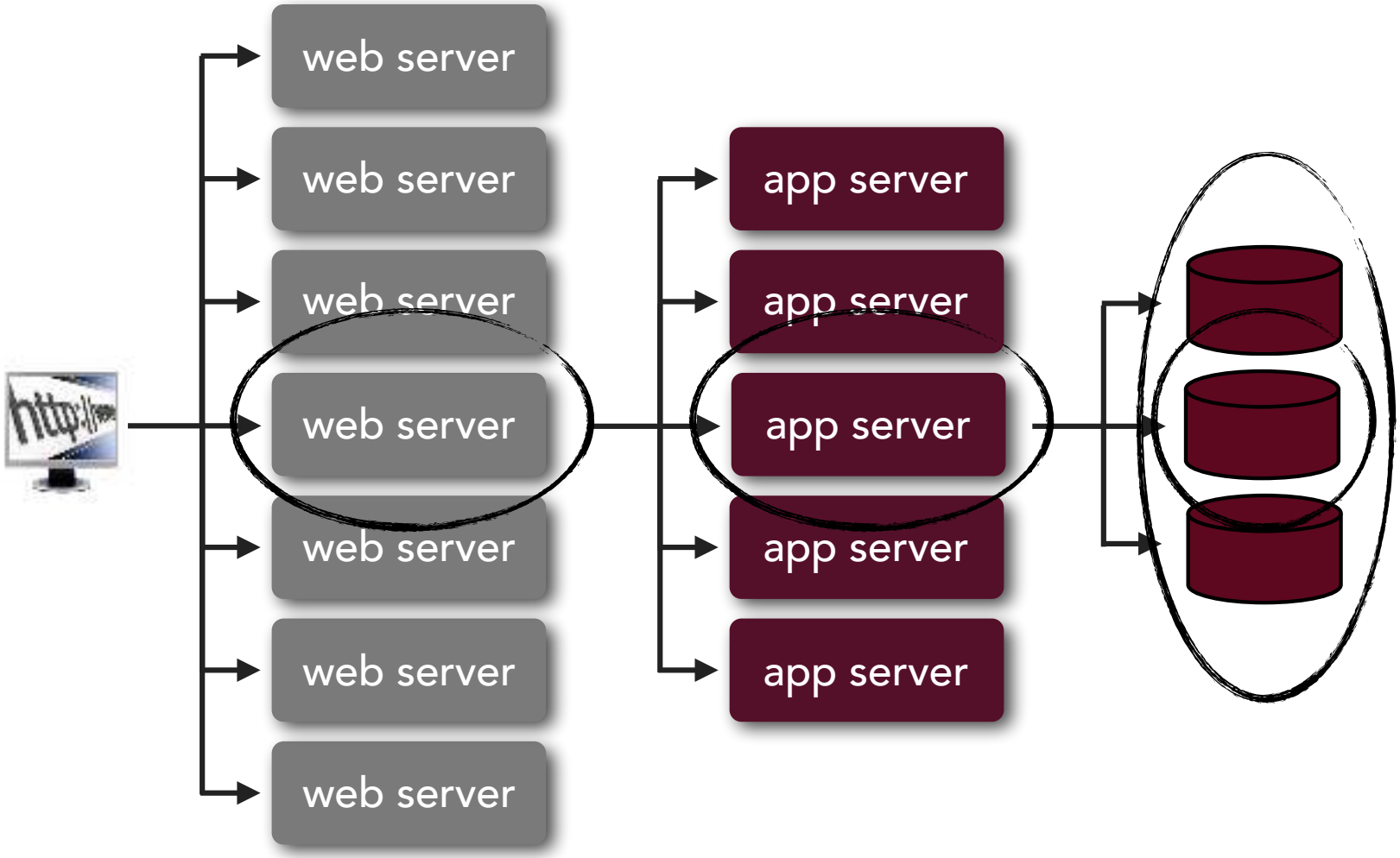


architecture pattern roadmap

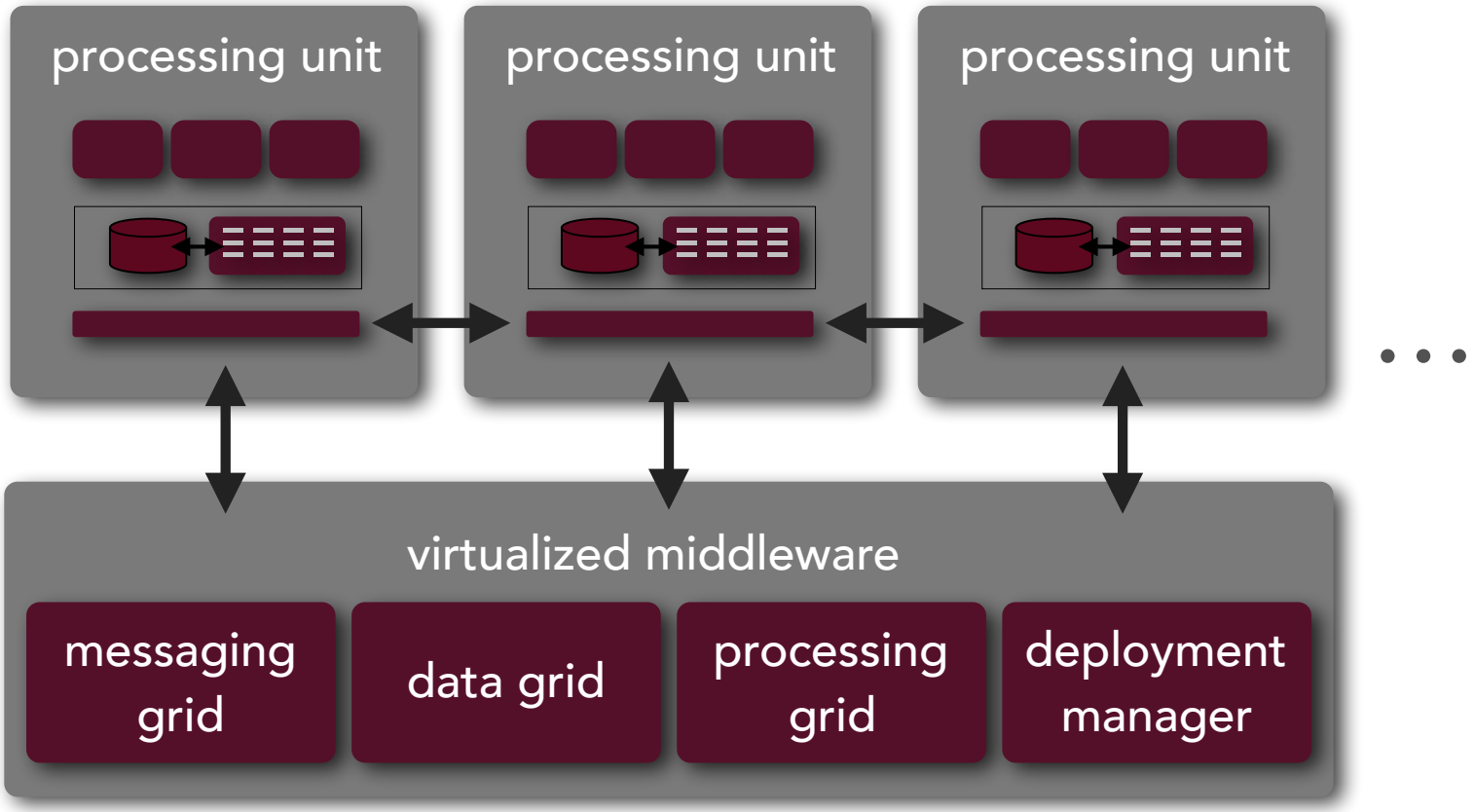


space-based architecture

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

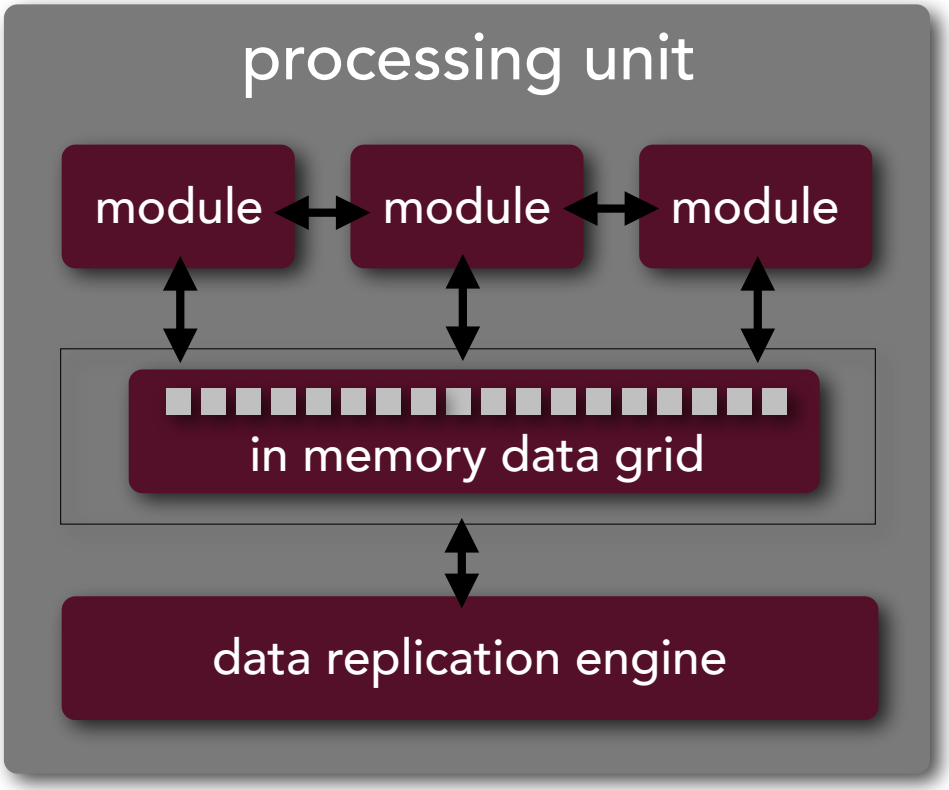


space-based architecture



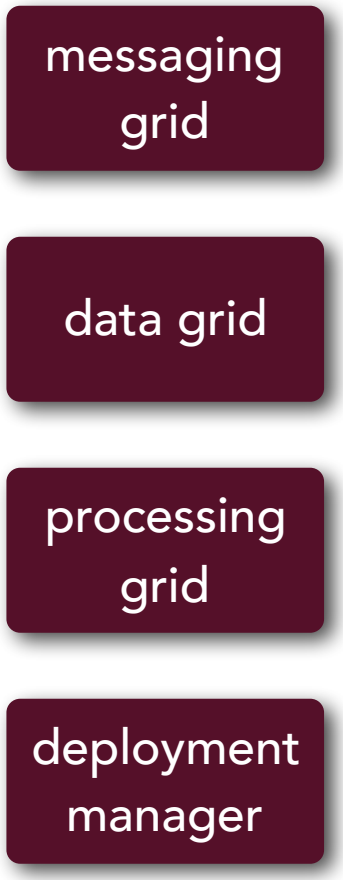
space-based architecture

processing unit



space-based architecture

middleware

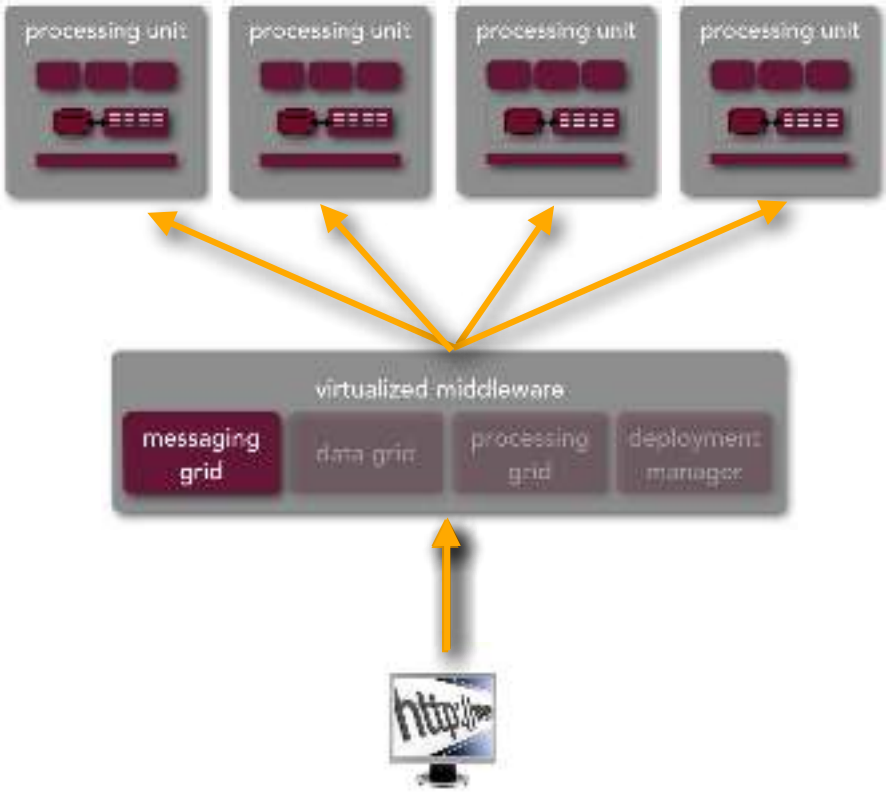


space-based architecture

middleware

- messaging grid
- data grid
- processing grid
- deployment manager

manages input request and session

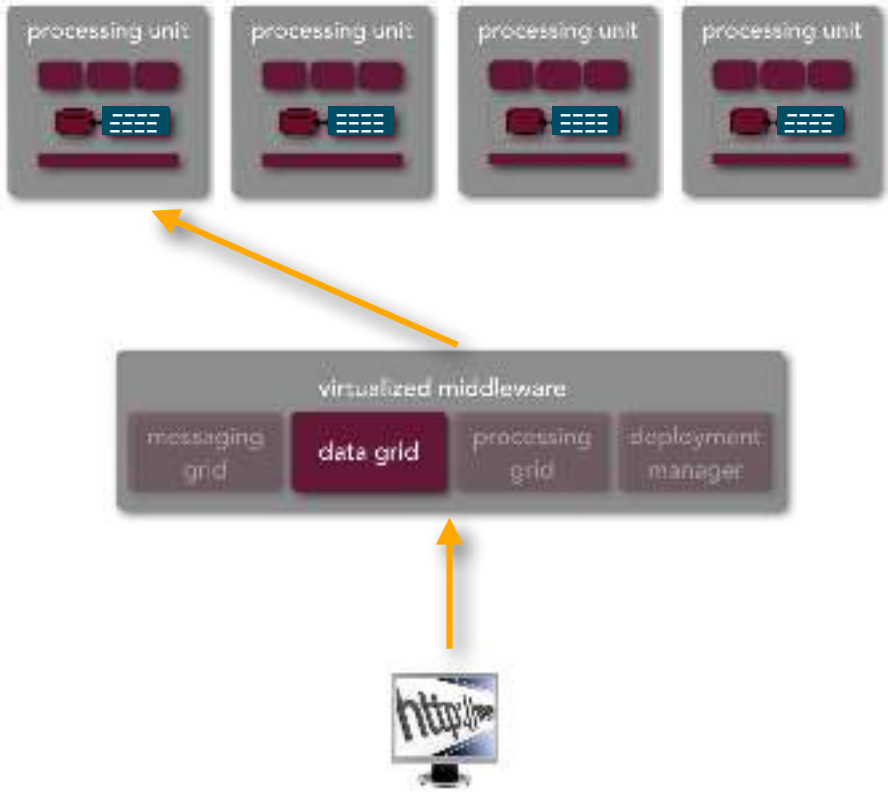


space-based architecture

middleware

manages data replication between processing units

- messaging grid
- **data grid**
- processing grid
- deployment manager

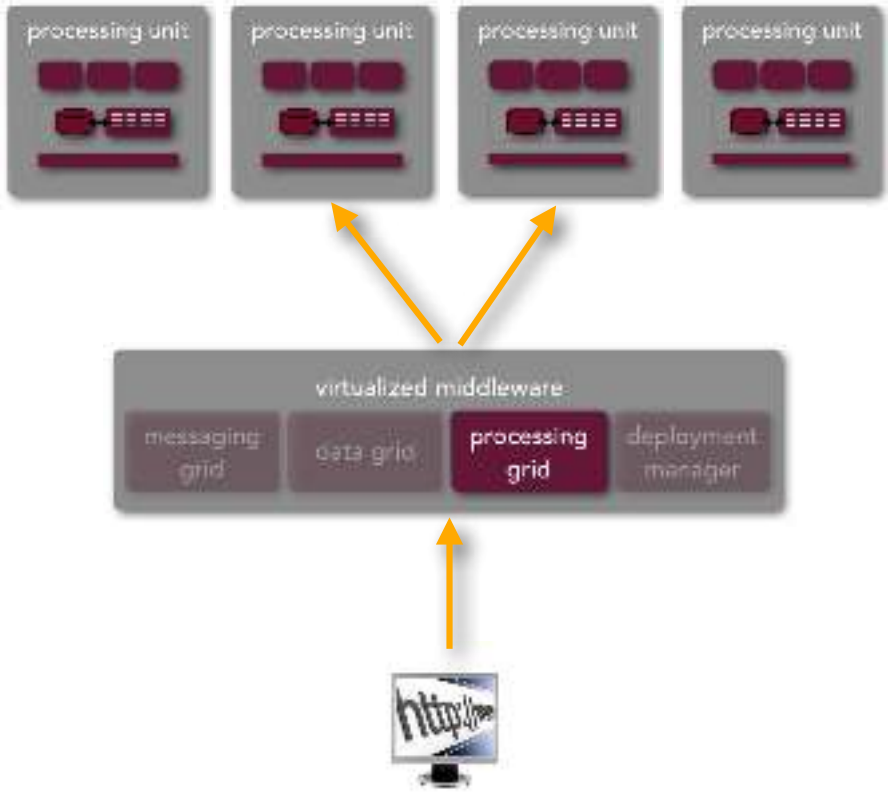


space-based architecture

middleware

manages distributed
request processing

- messaging grid
- data grid
- processing grid
- deployment manager



space-based architecture

middleware

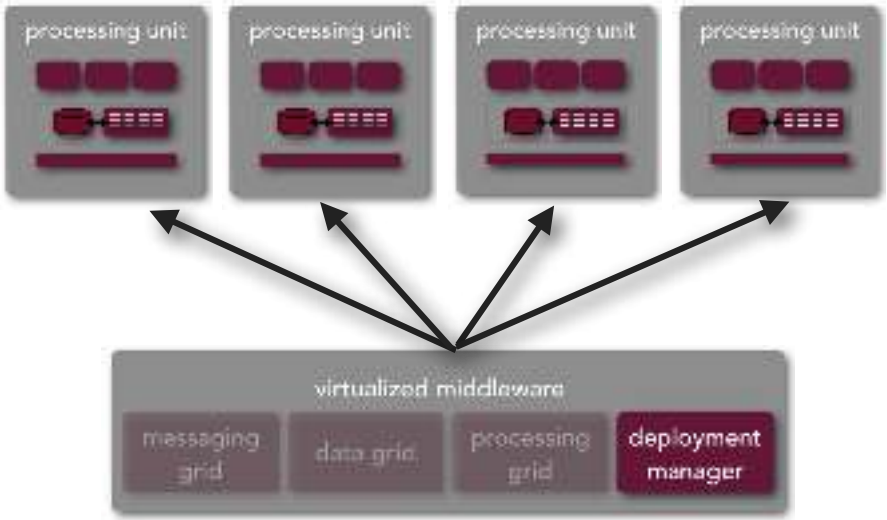
messaging grid

data grid

processing grid

deployment manager

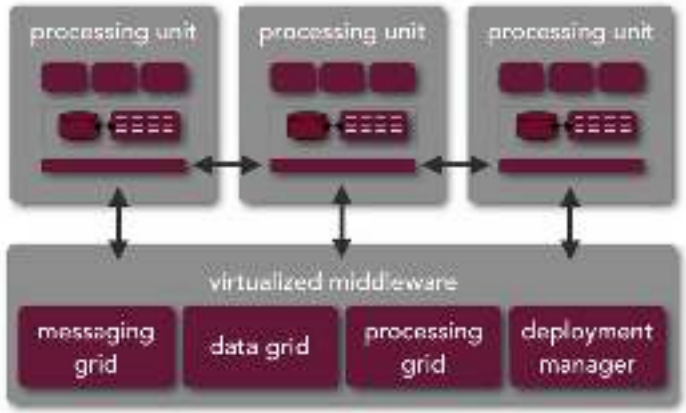
manages dynamic processing unit deployment



space-based architecture

product implementations

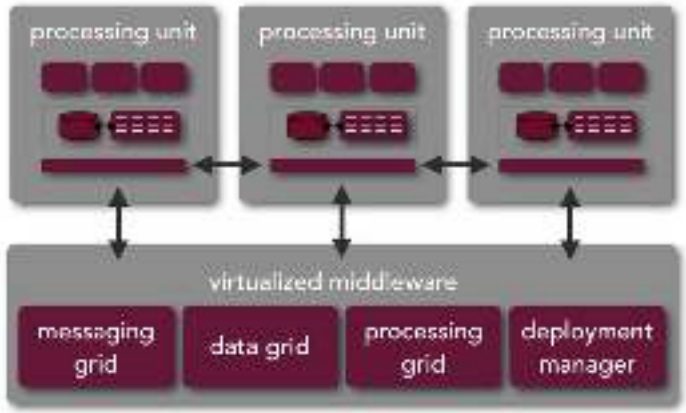
- javaspaces
- gigaspaces
- ibm object grid
- gemfire
- ncache
- oracle coherence



space-based architecture

it's all about variable scalability...

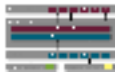
good for applications that have variable load or inconsistent peak times



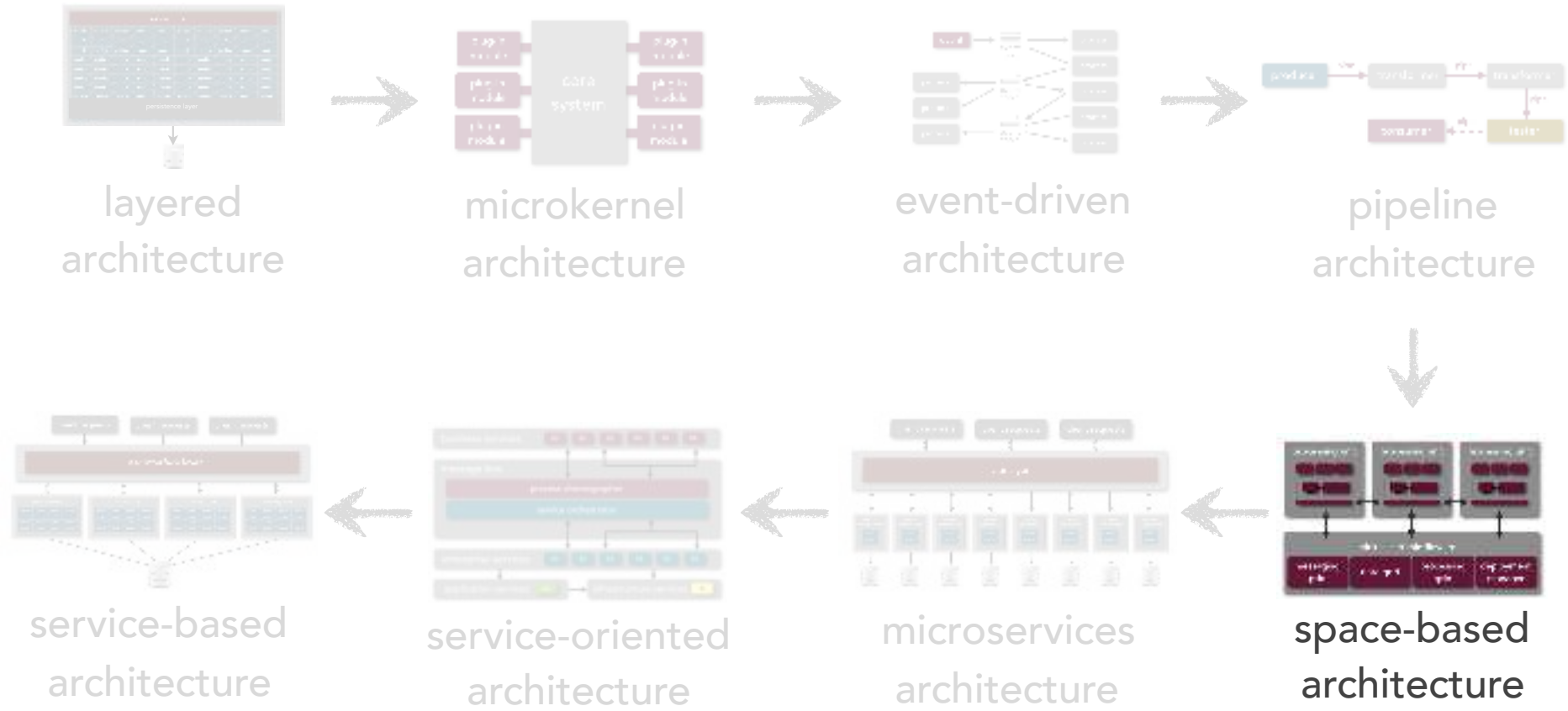
not a good fit for traditional large-scale relational database systems

relatively complex and expensive pattern to implement

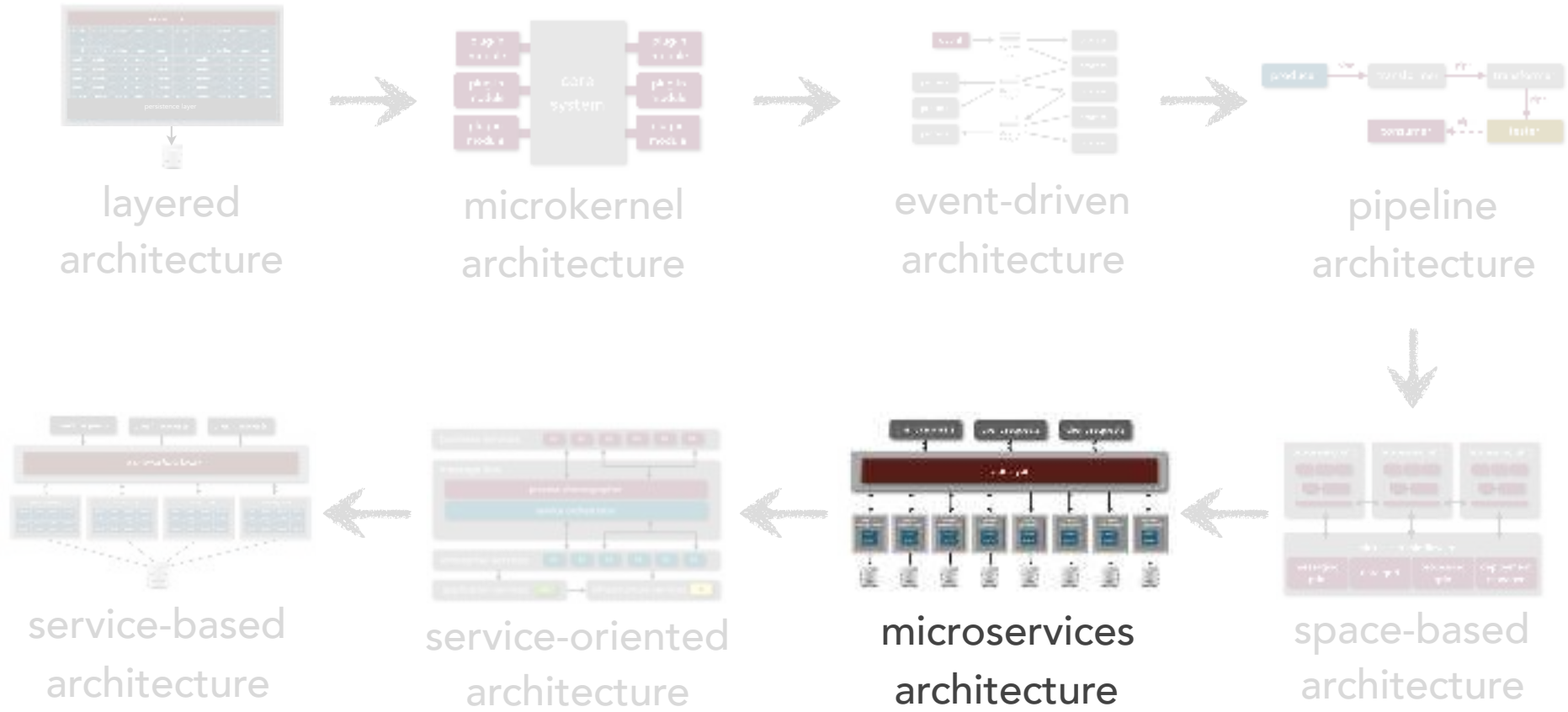
space-based architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👎	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
							
							
							

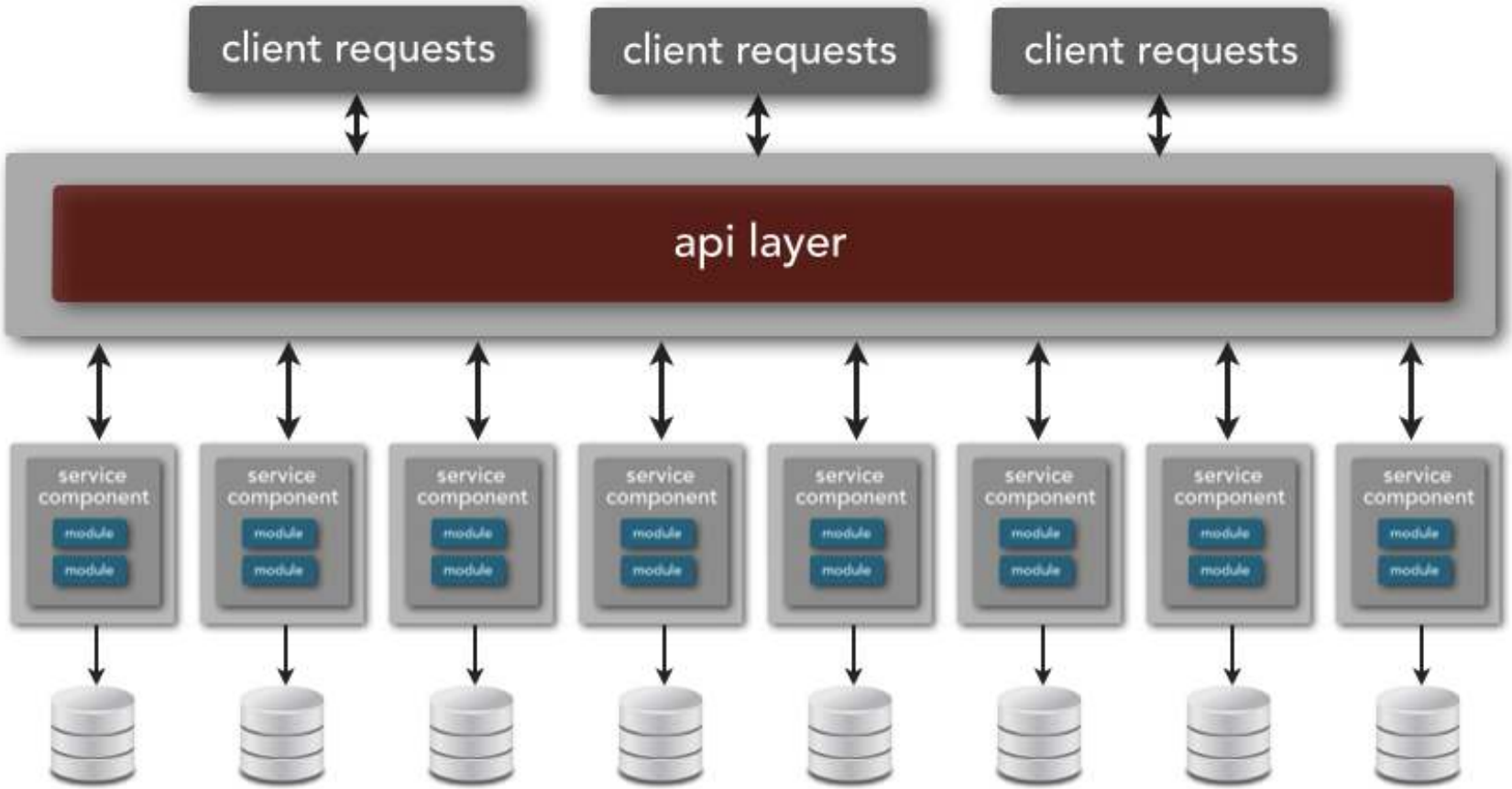
architecture pattern roadmap



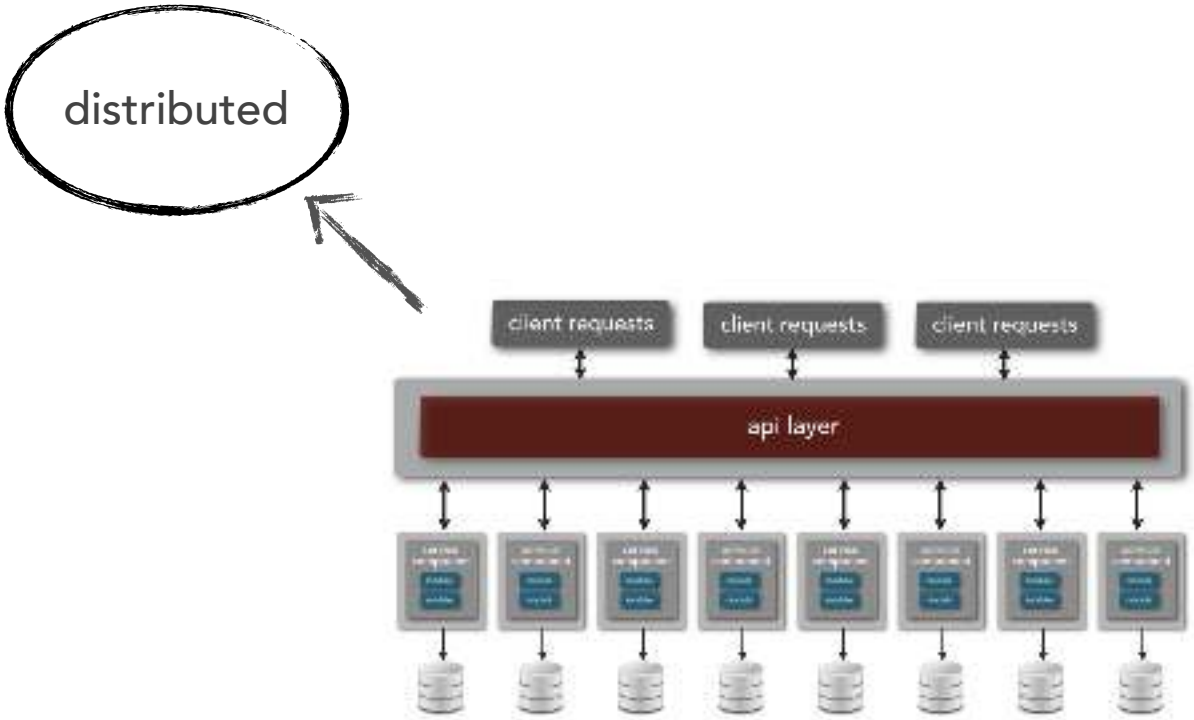
architecture pattern roadmap



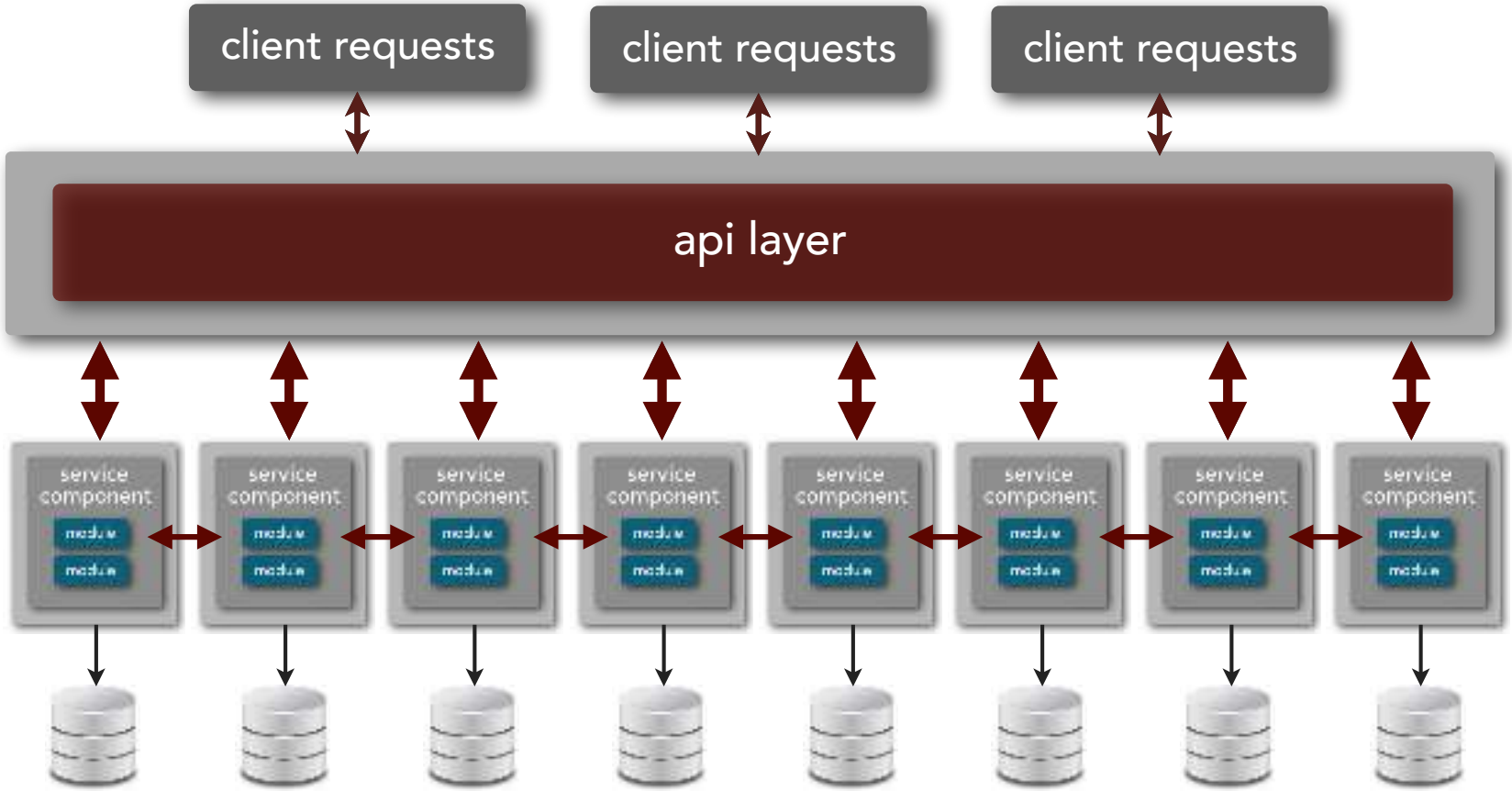
microservices architecture



microservices architecture

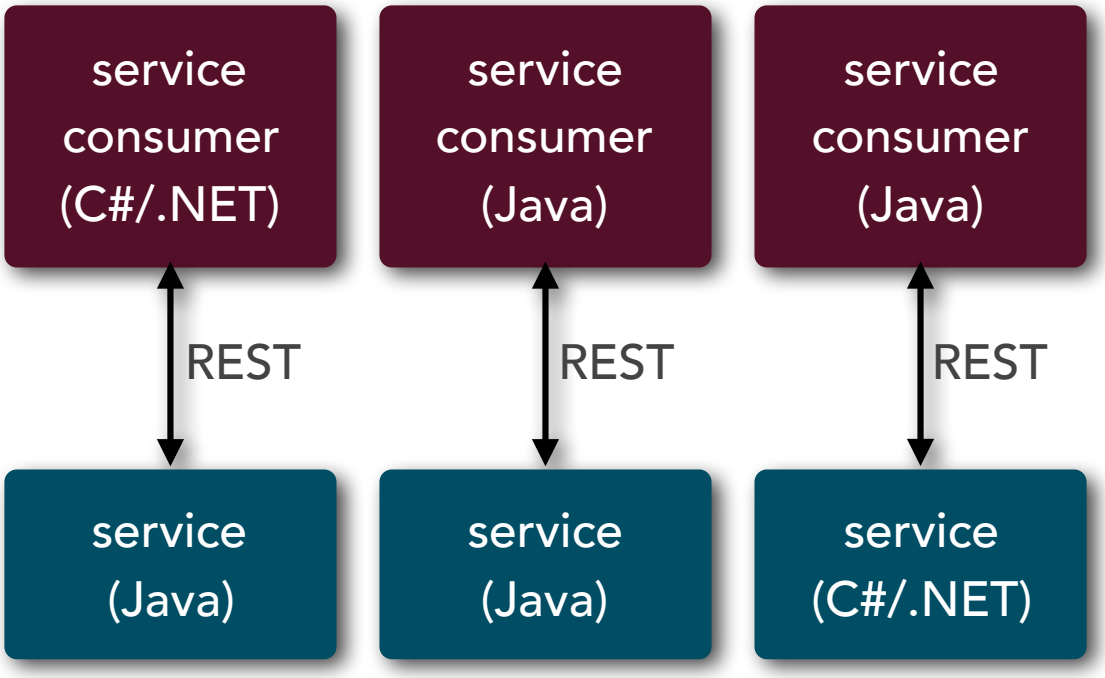


microservices architecture

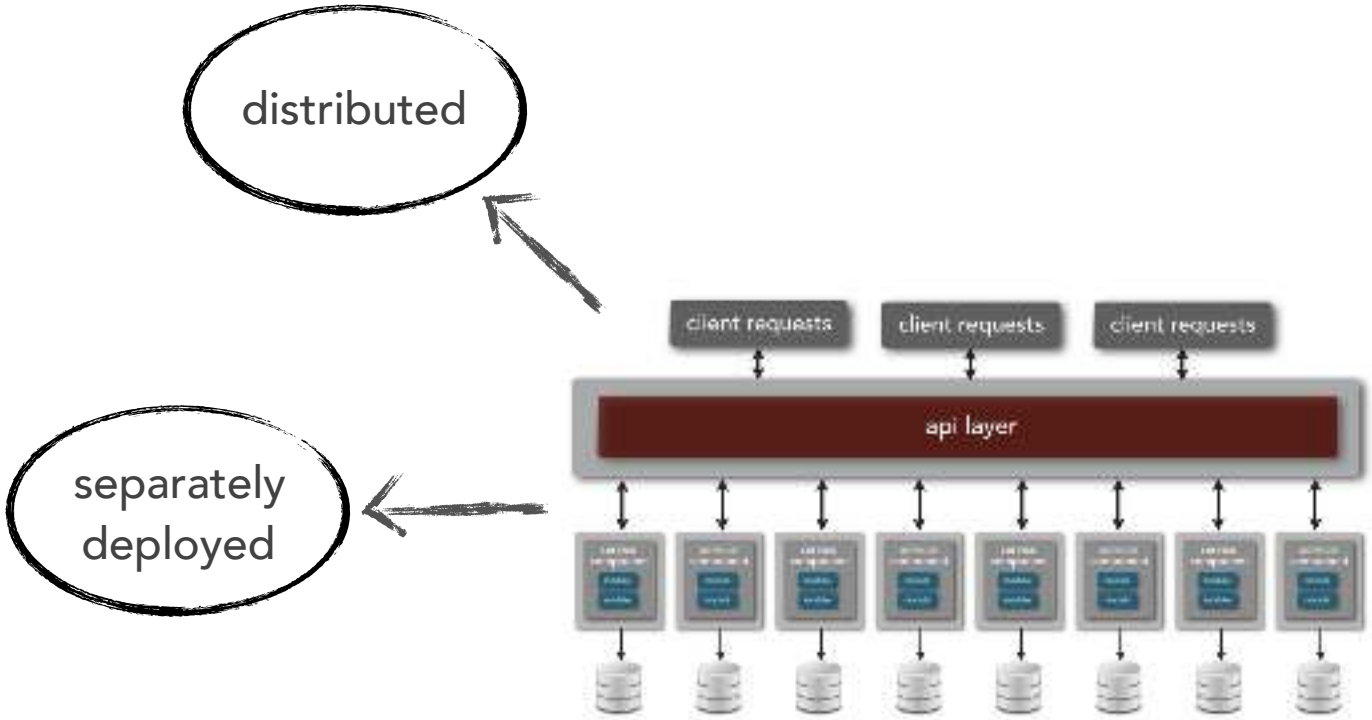


microservices architecture

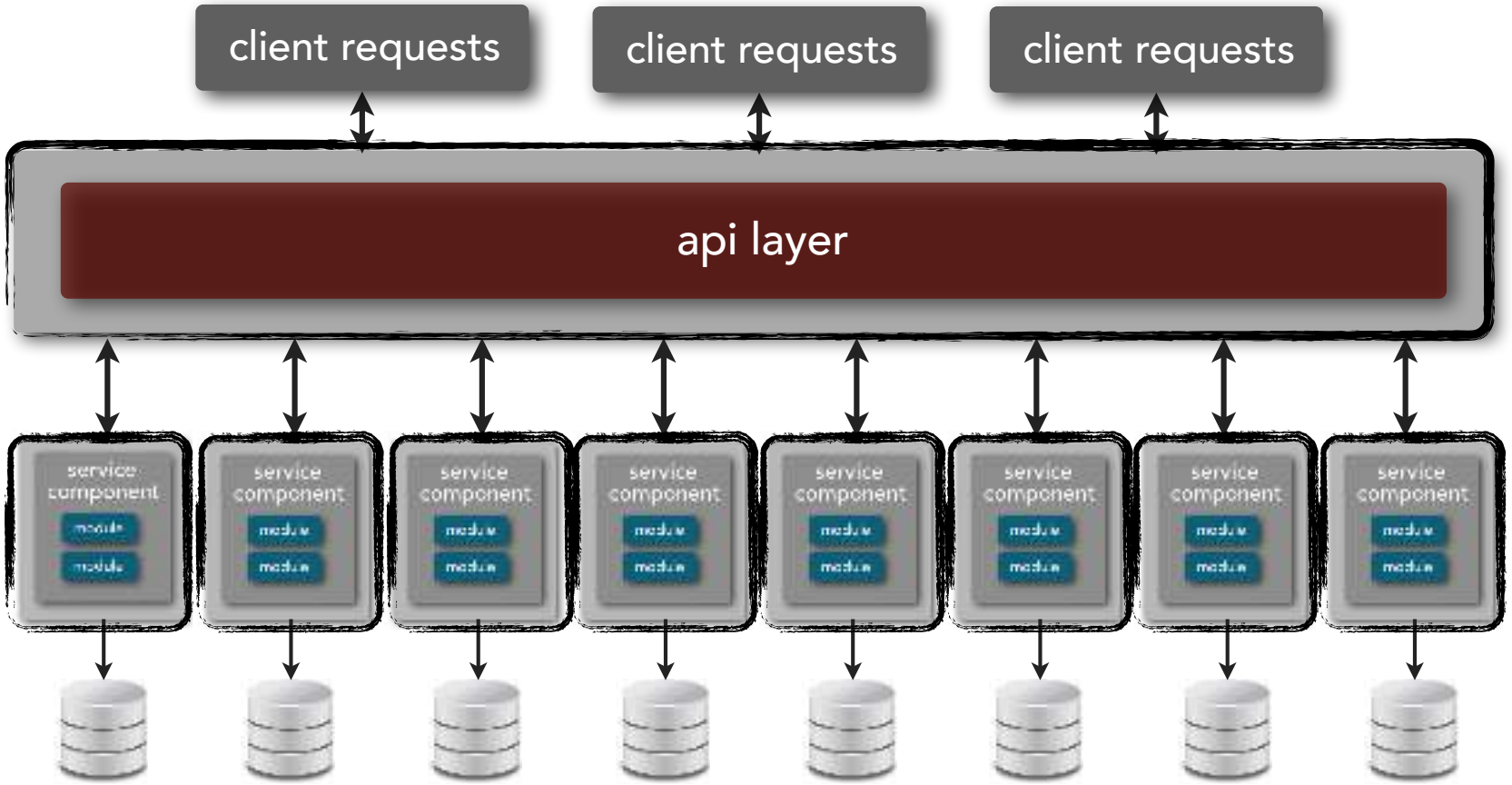
protocol-aware heterogeneous interoperability



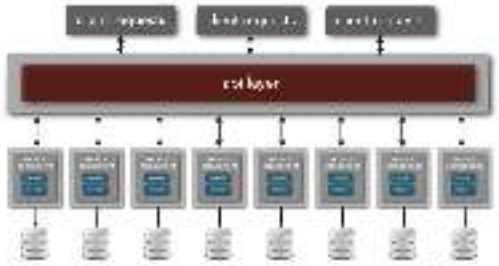
microservices architecture



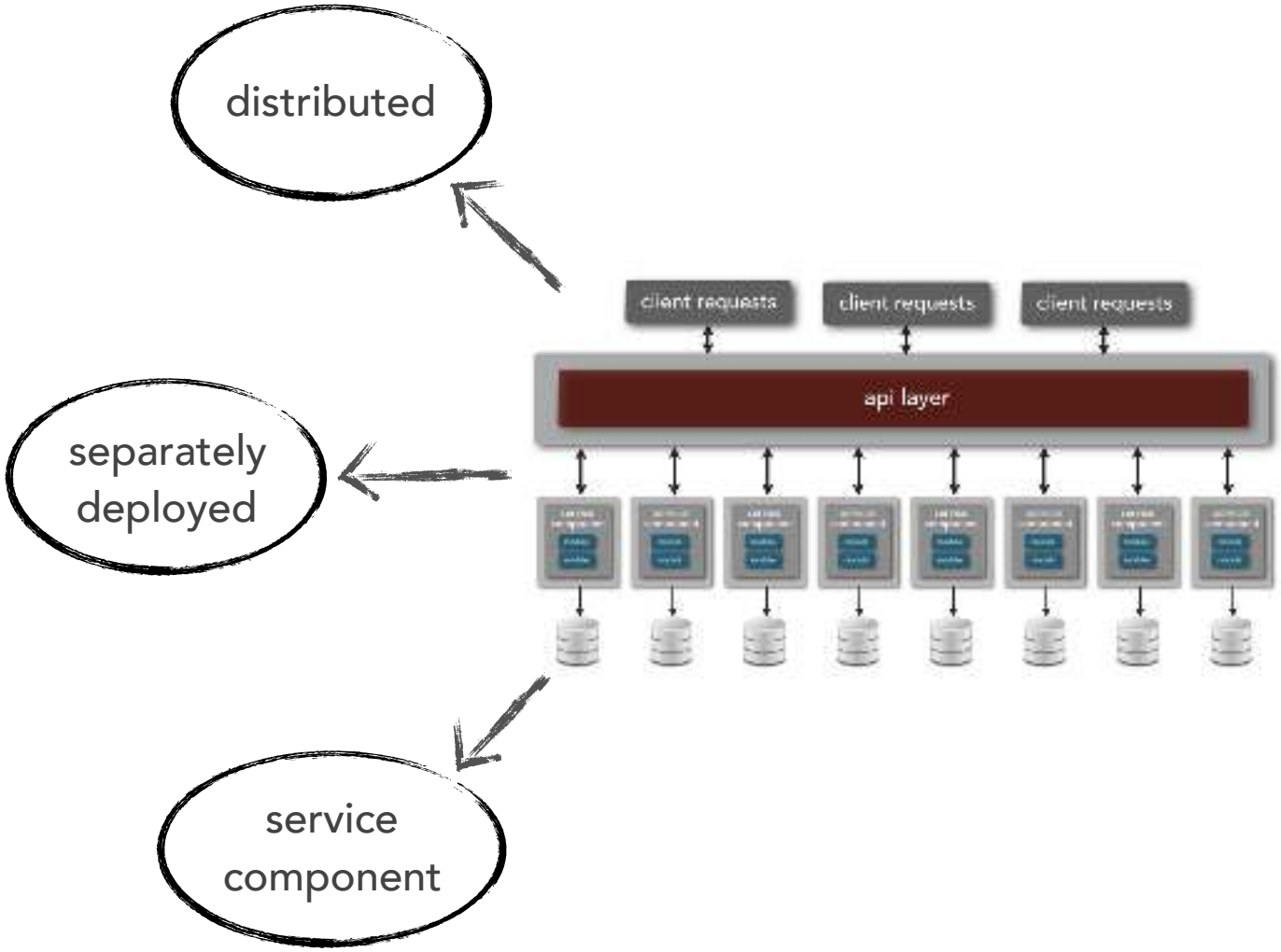
microservices architecture



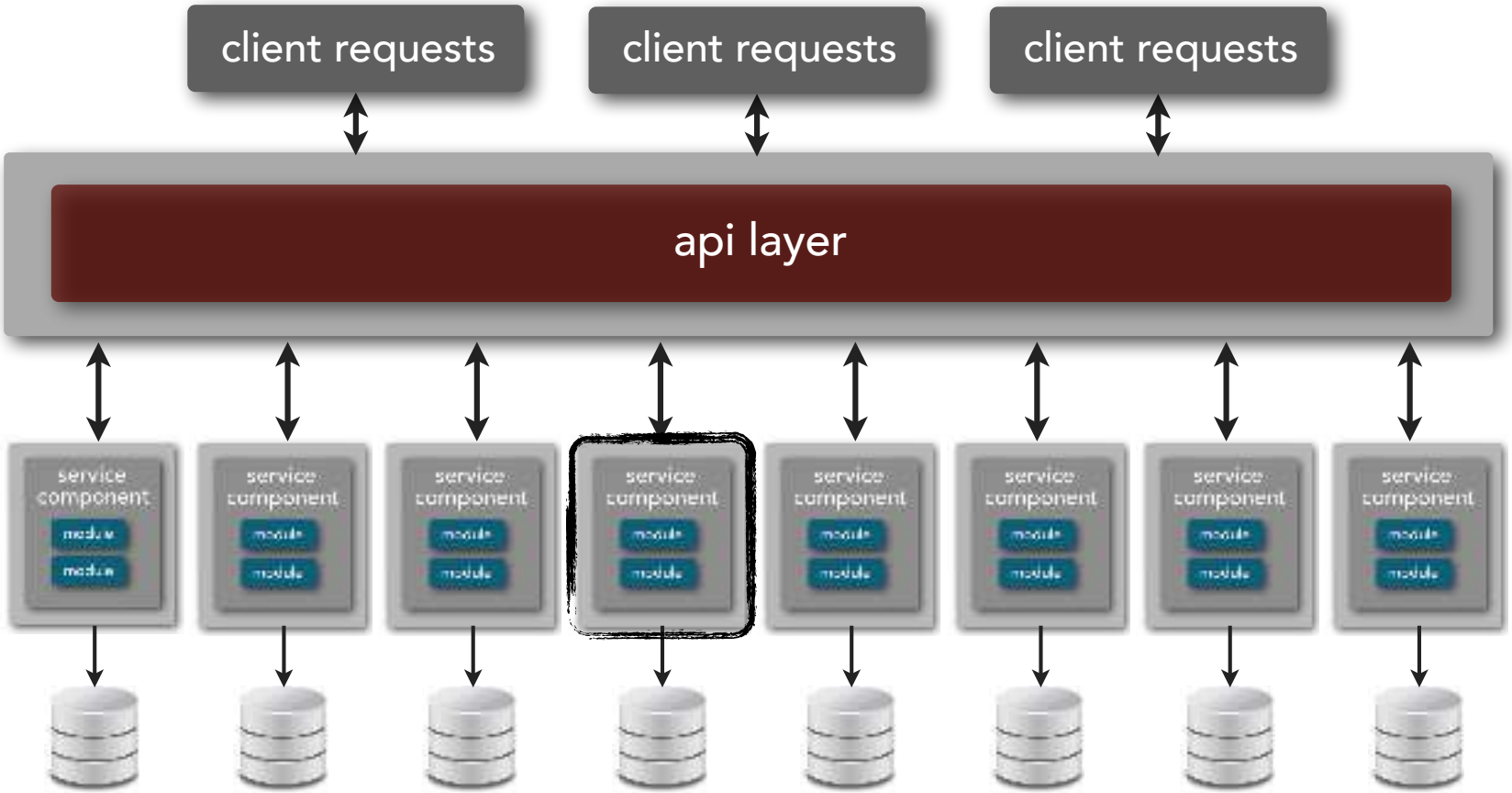
microservices architecture

SERVICE REGISTRY AND DISCOVERY	SECURITY AND COMPLIANCE	LOAD BALANCING
DEPLOYMENT	INTER-SERVICE COMMUNICATION	NETWORK
MONITORING		INFRASTRUCTURE AUTOMATION
CONTINUOUS INTEGRATION		PLATFORM MANAGEMENT
CONTAINER REGISTRY	API MANAGEMENT	SERVICE OPTIMIZATION
CLOUD MANAGEMENT	OPERATING SYSTEM	DATABASE MANAGEMENT

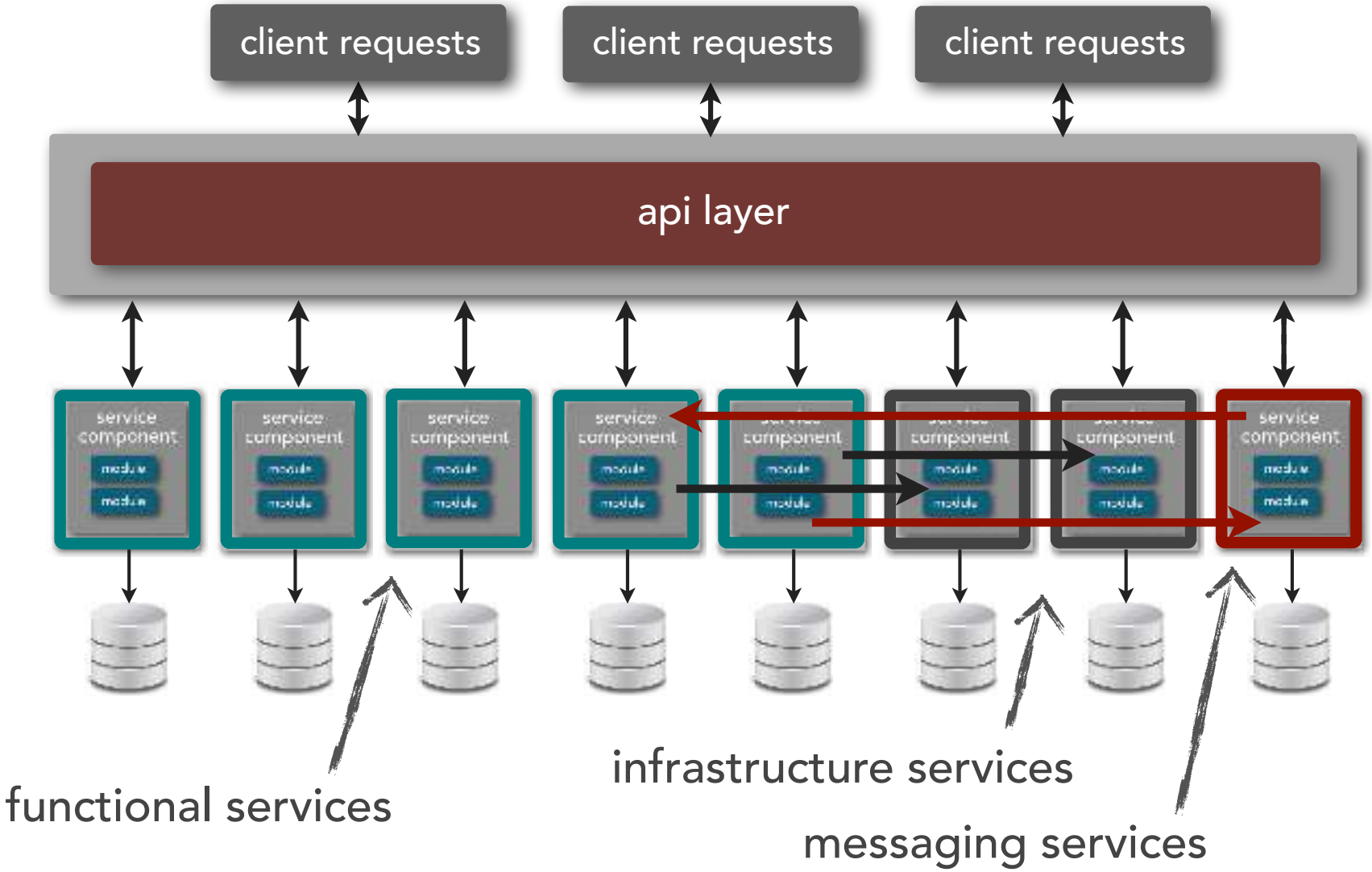
microservices architecture



microservices architecture



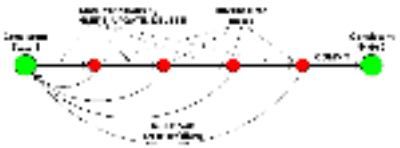
microservices architecture



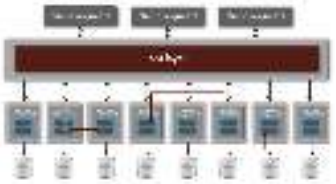
microservices architecture

what is the right size for a microservice?

purpose



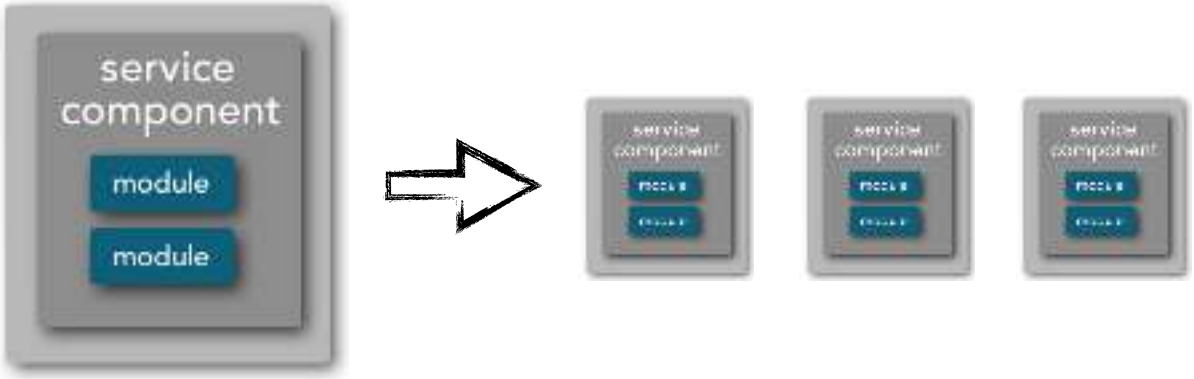
transactions



choreography

microservices architecture

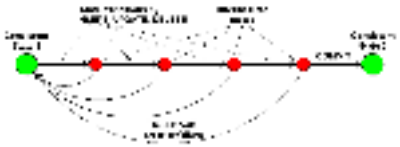
purpose



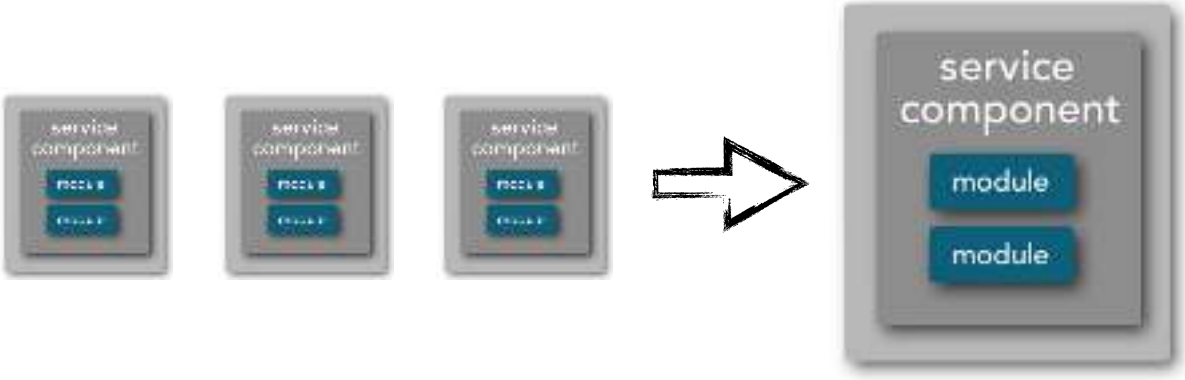
service scope and function

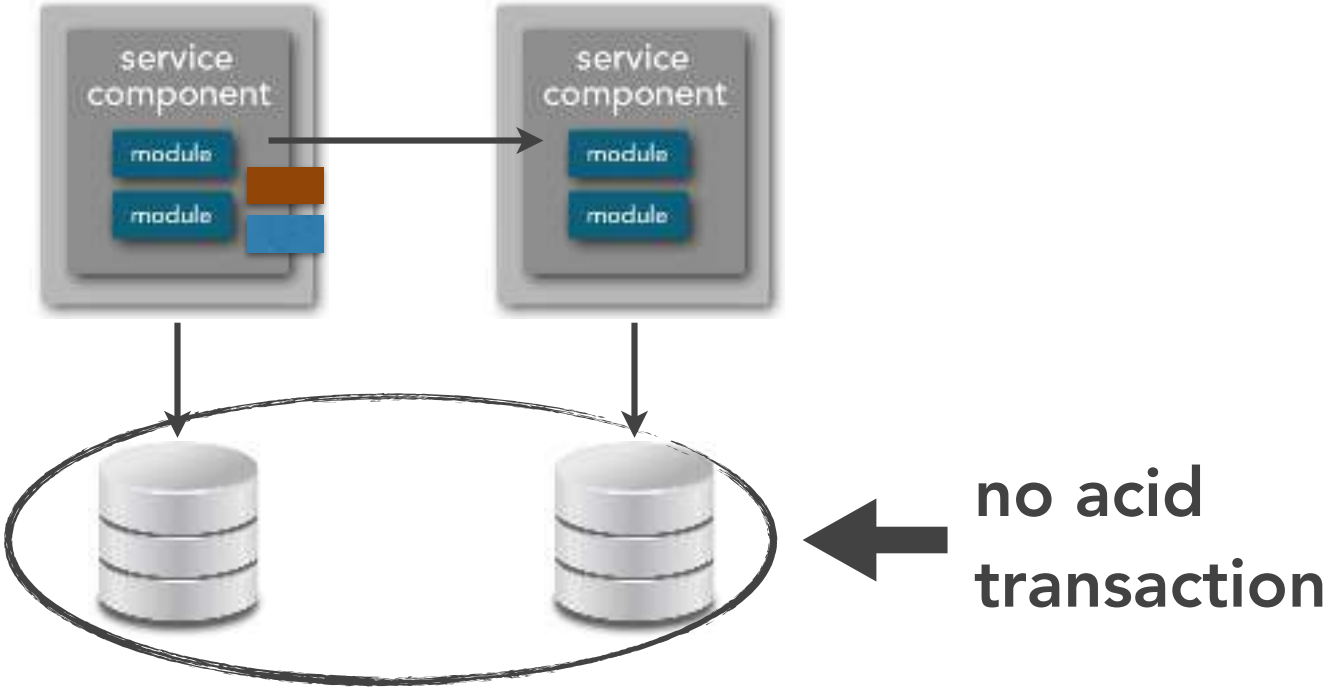
(single-purpose function)

microservices architecture

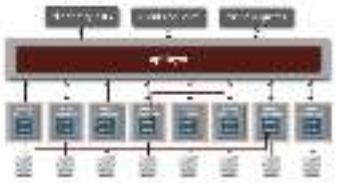


transactions

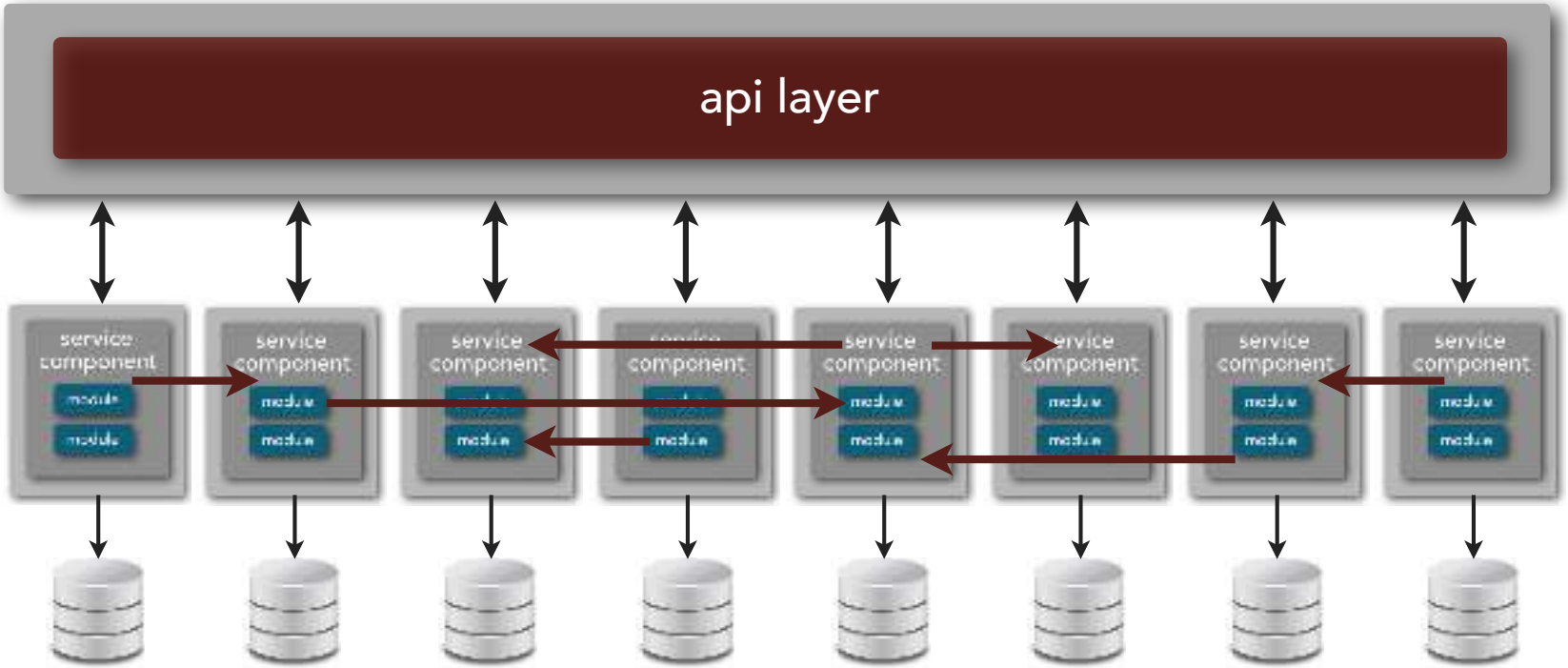




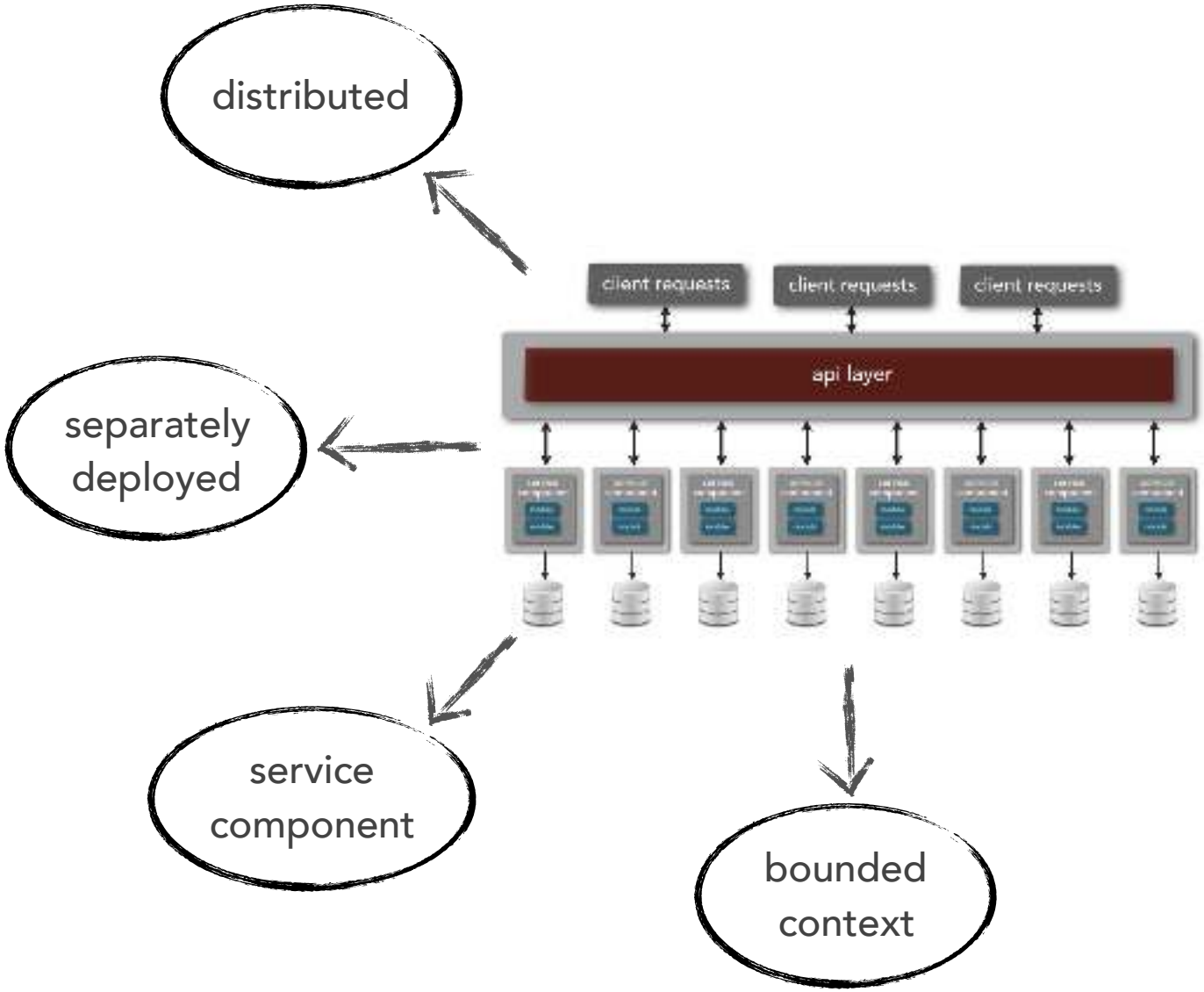
microservices architecture



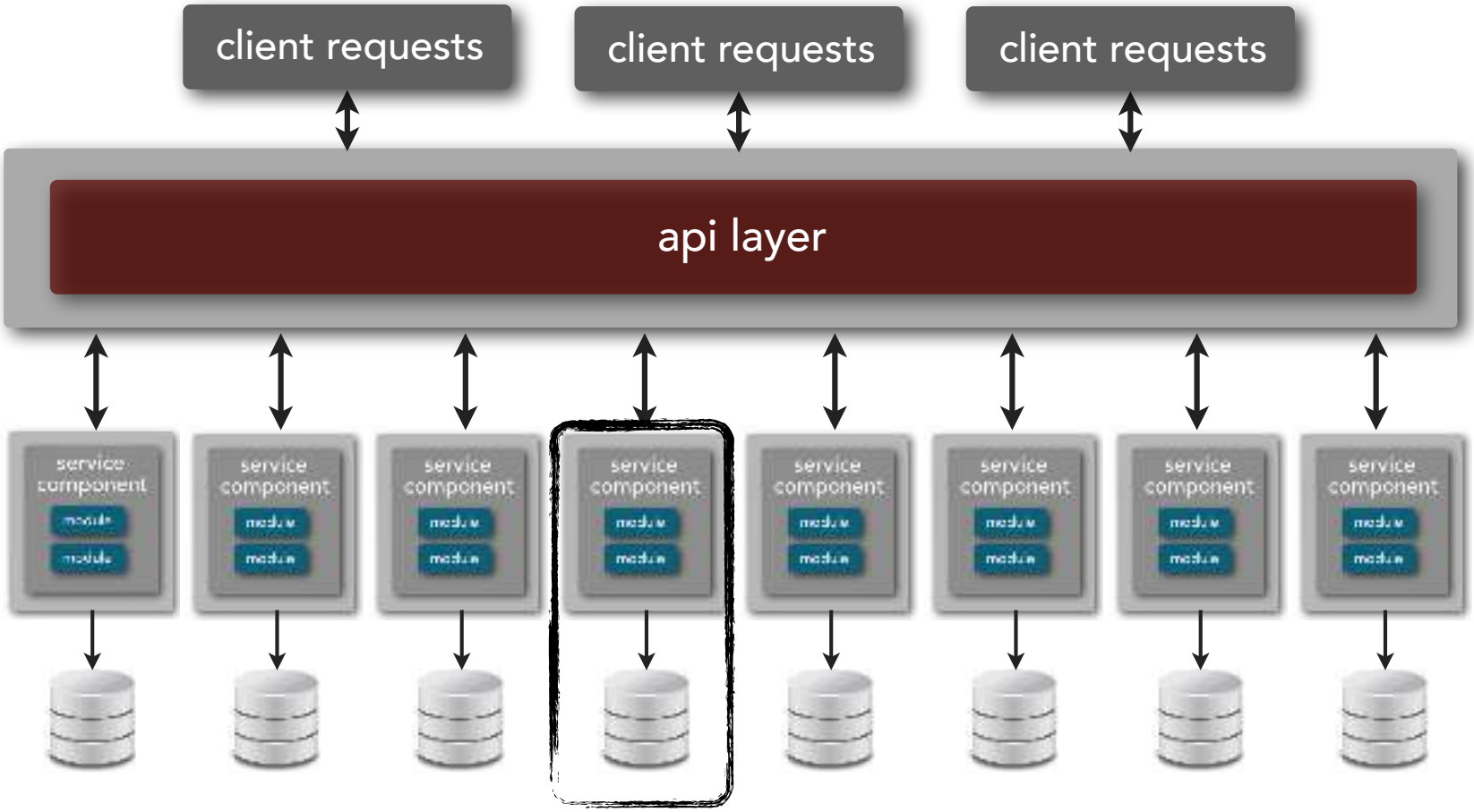
choreography



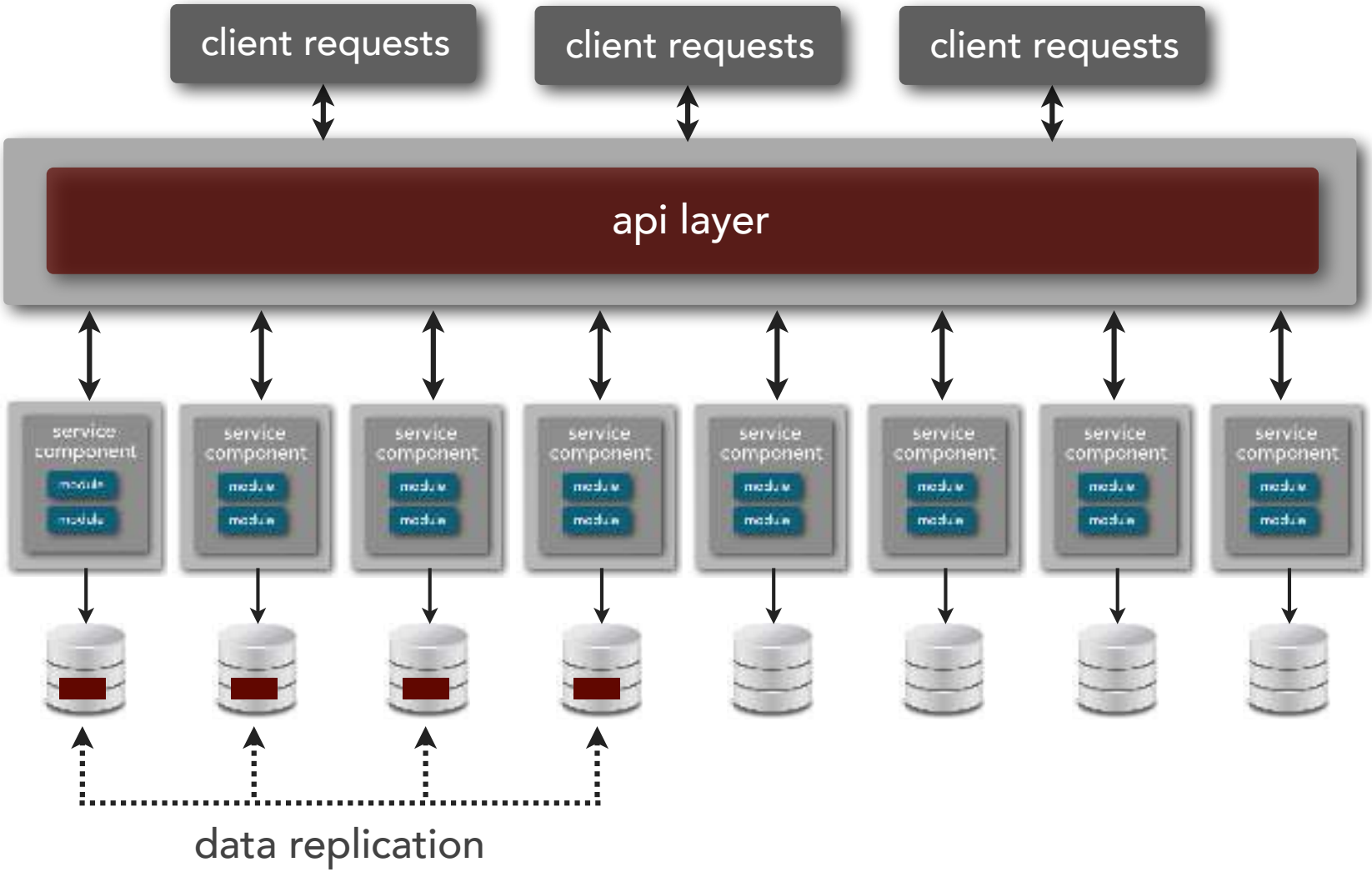
microservices architecture



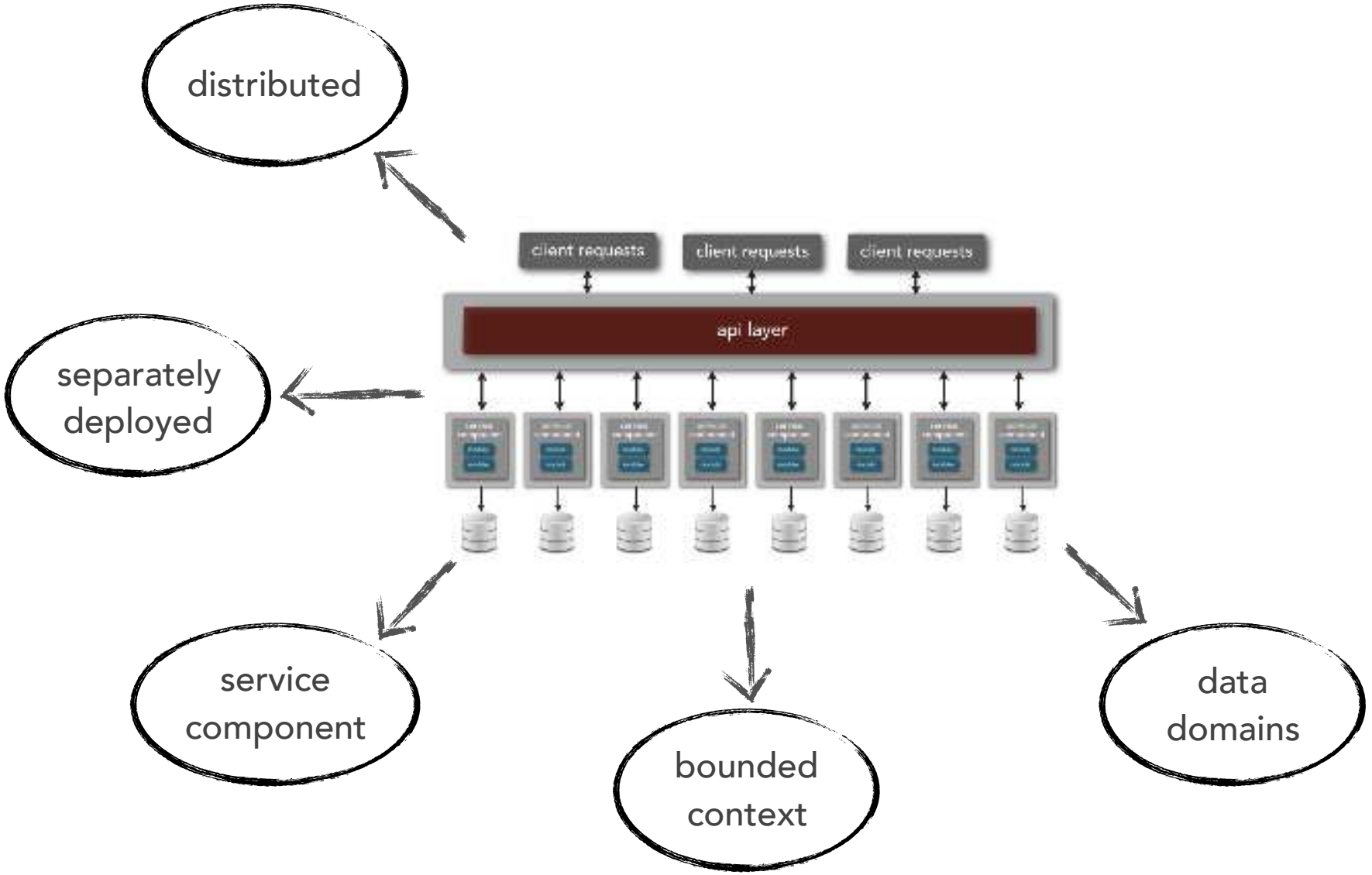
microservices architecture



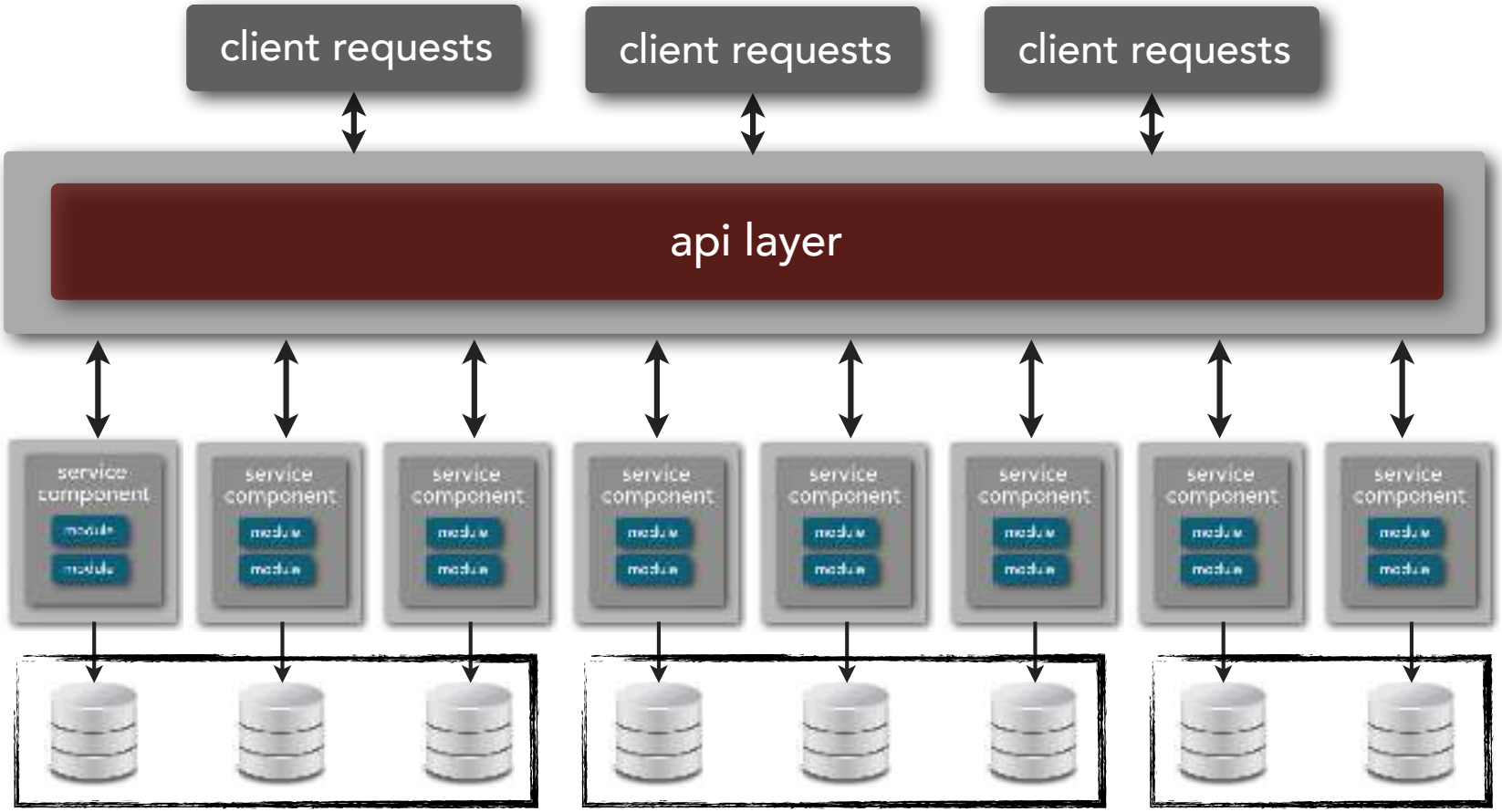
microservices architecture



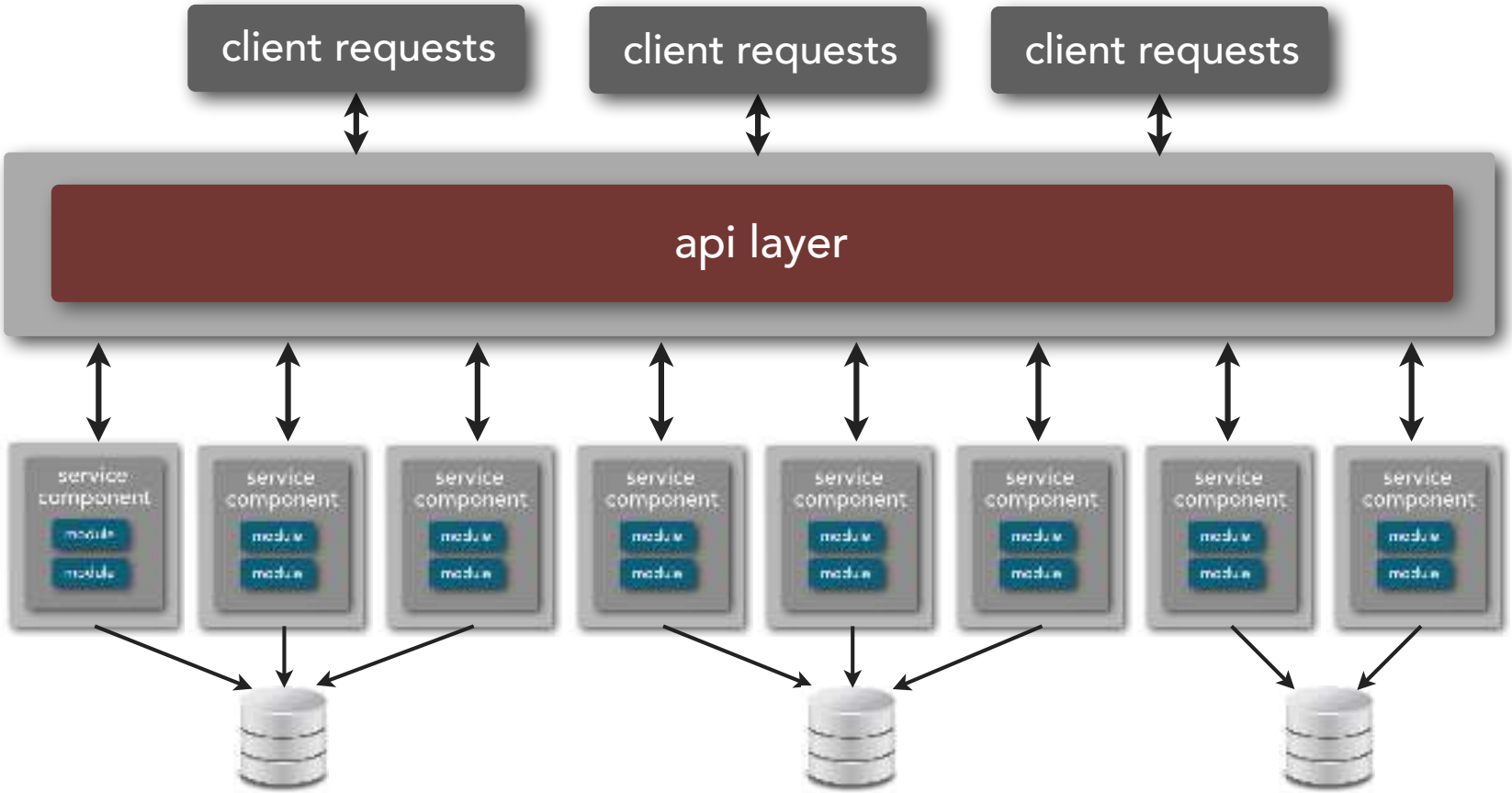
microservices architecture



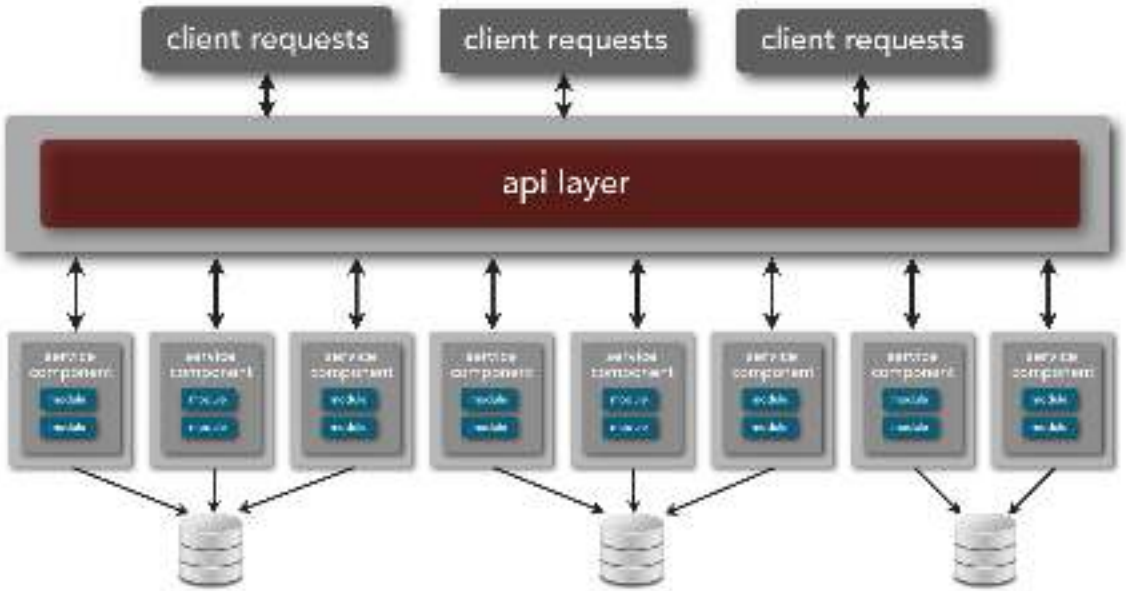
microservices architecture



microservices architecture



microservices architecture

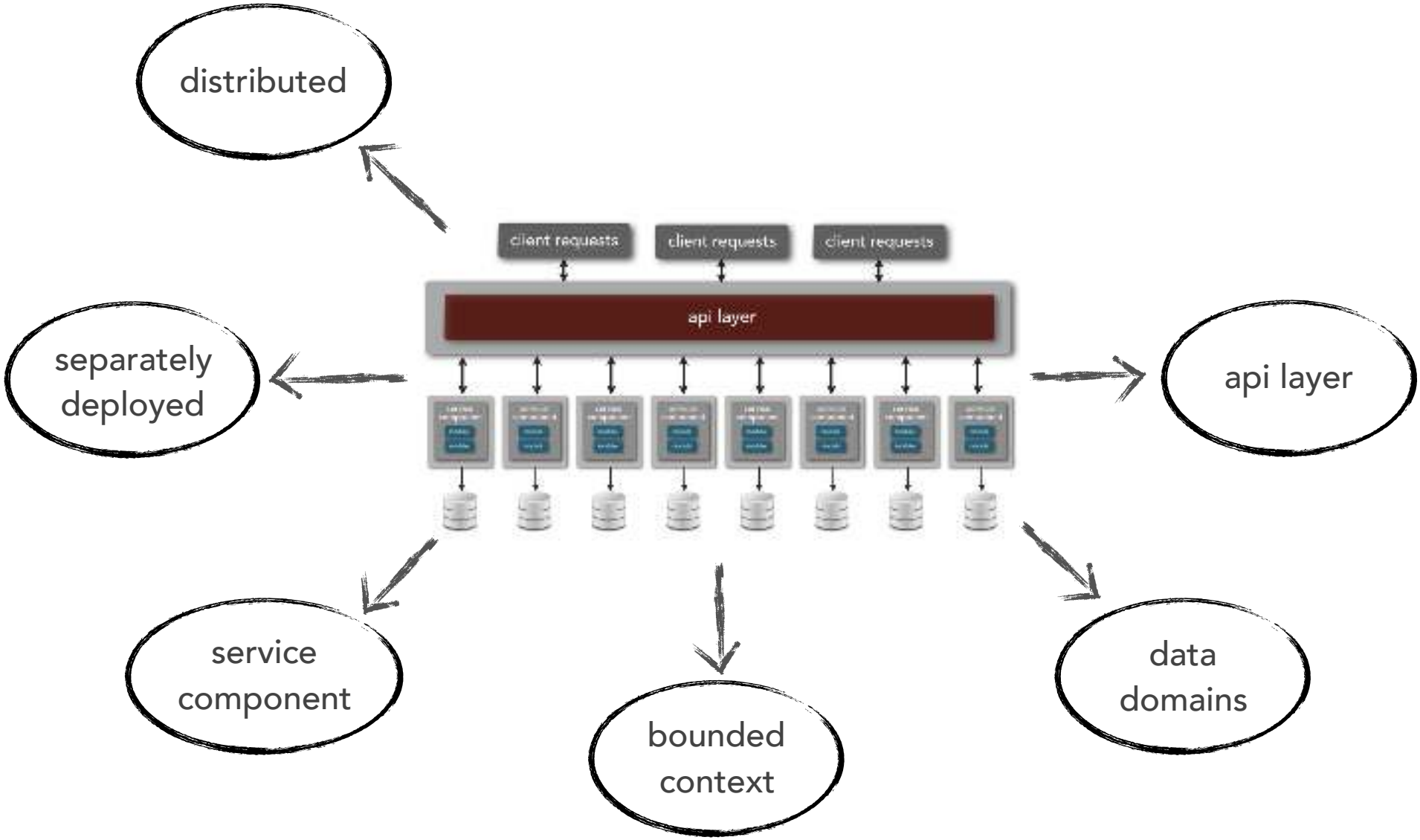


assumes low rate of schema changes (or use of noSQL)

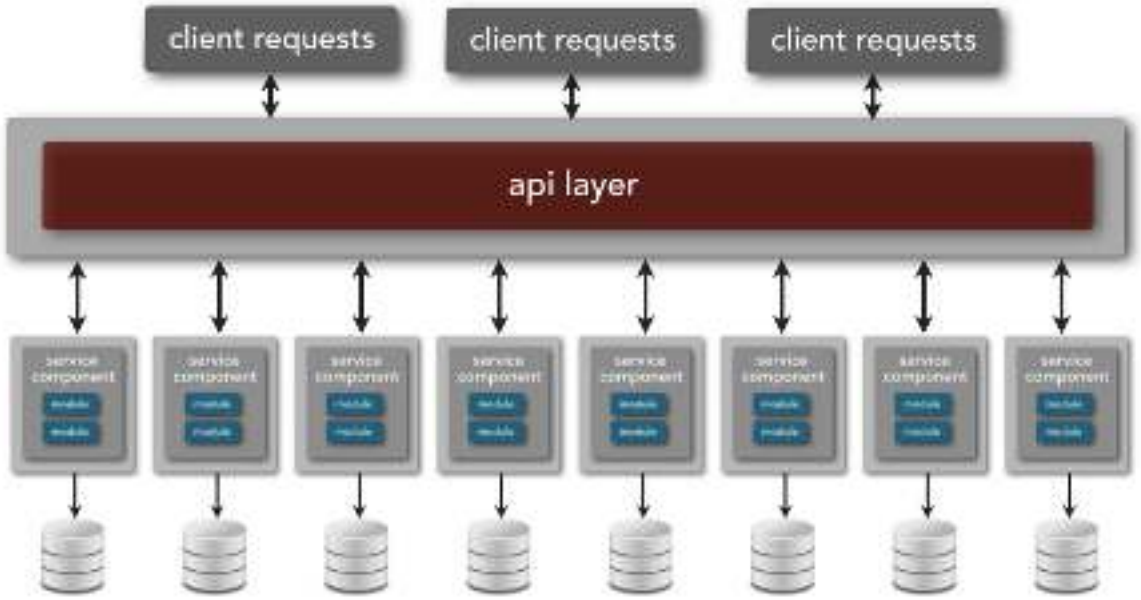
increases performance and overall reliability

reduces data duplication

microservices architecture

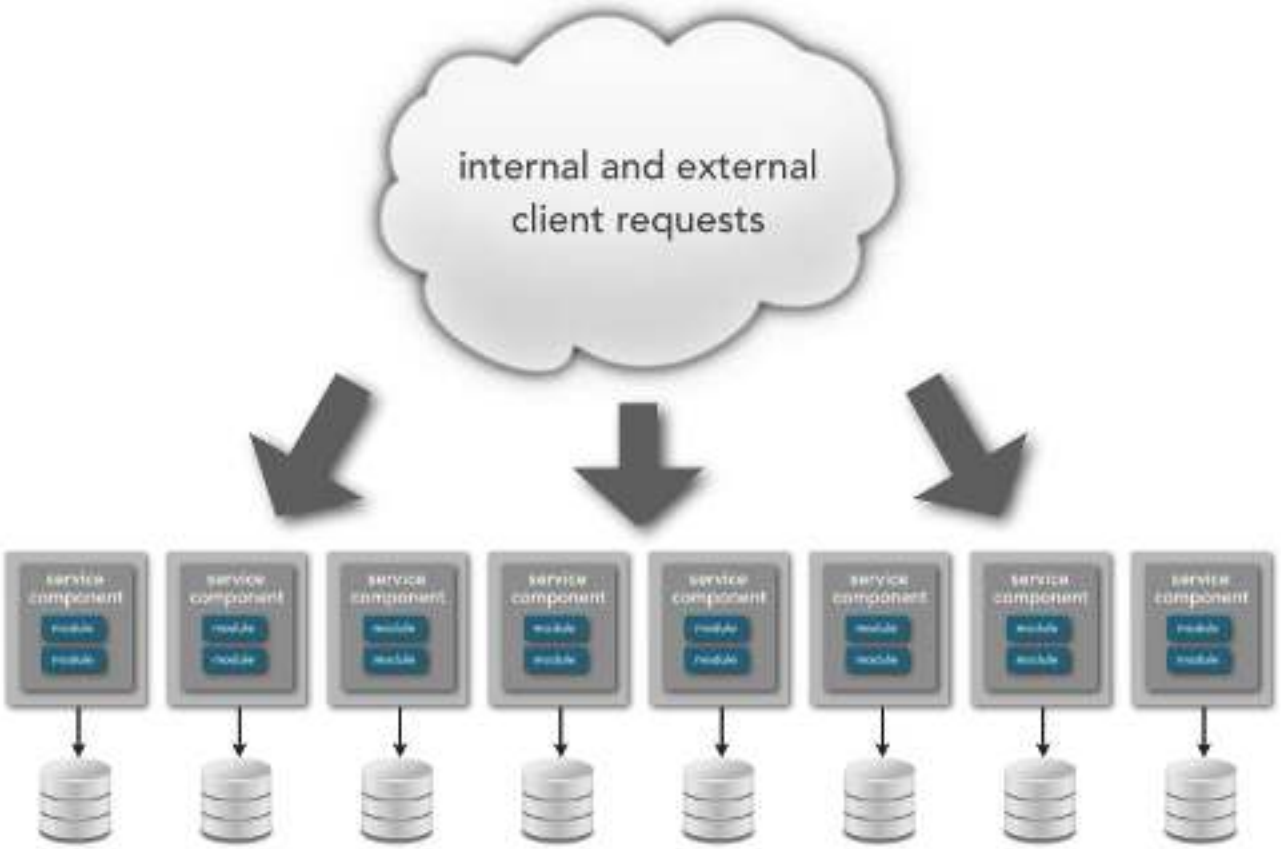


api layer

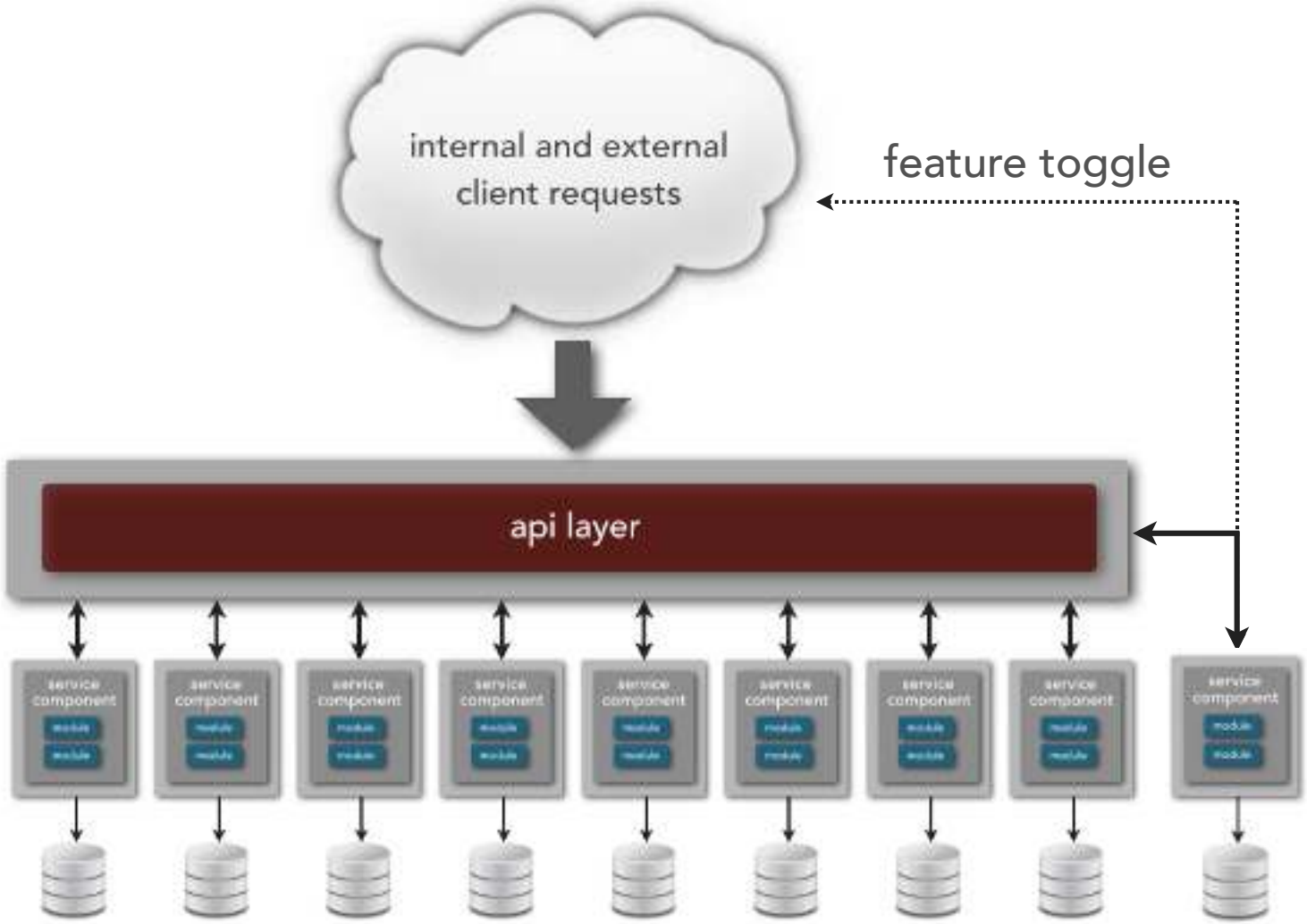


hides the actual endpoint of the service, exposing only those services available for public consumption

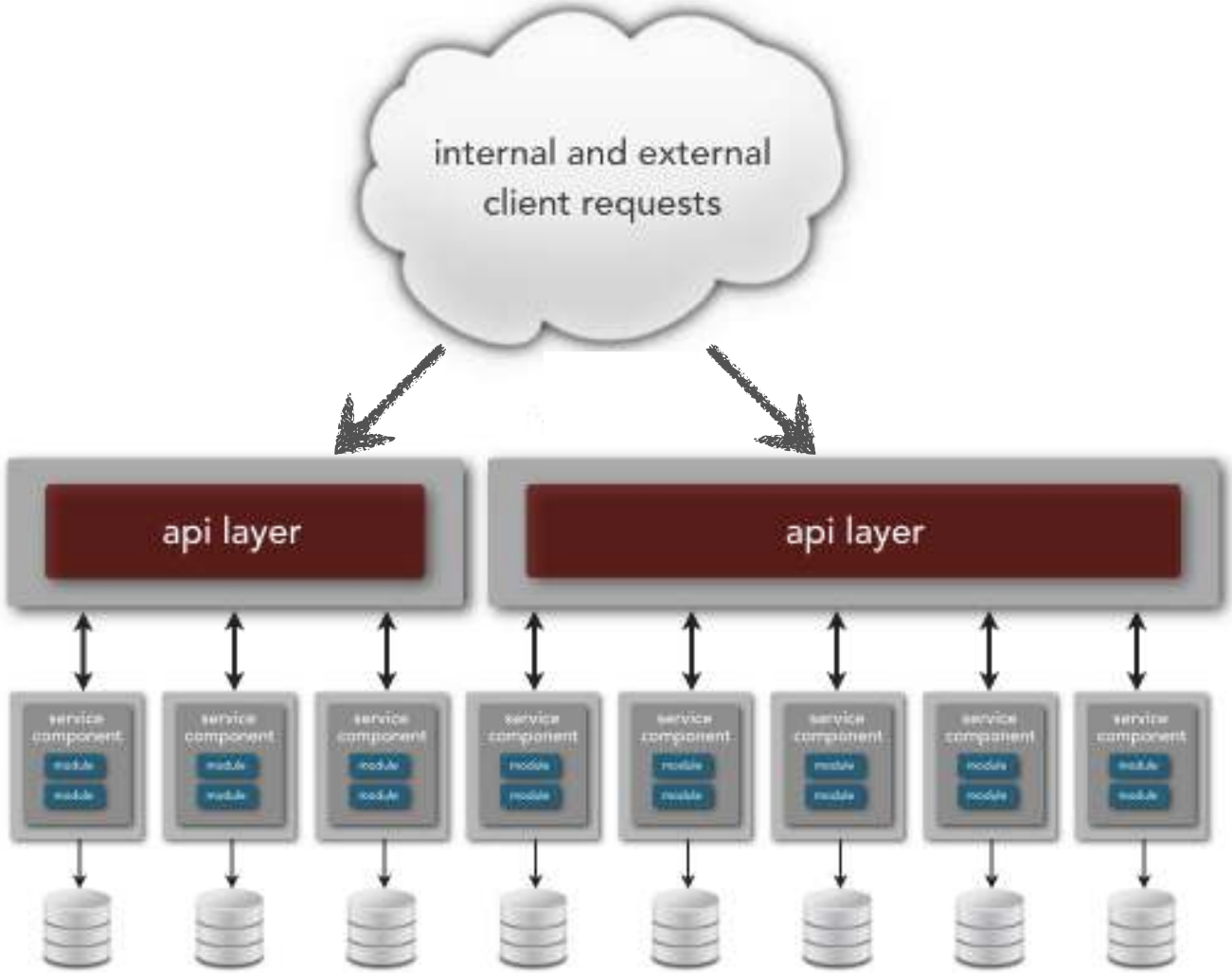
api layer



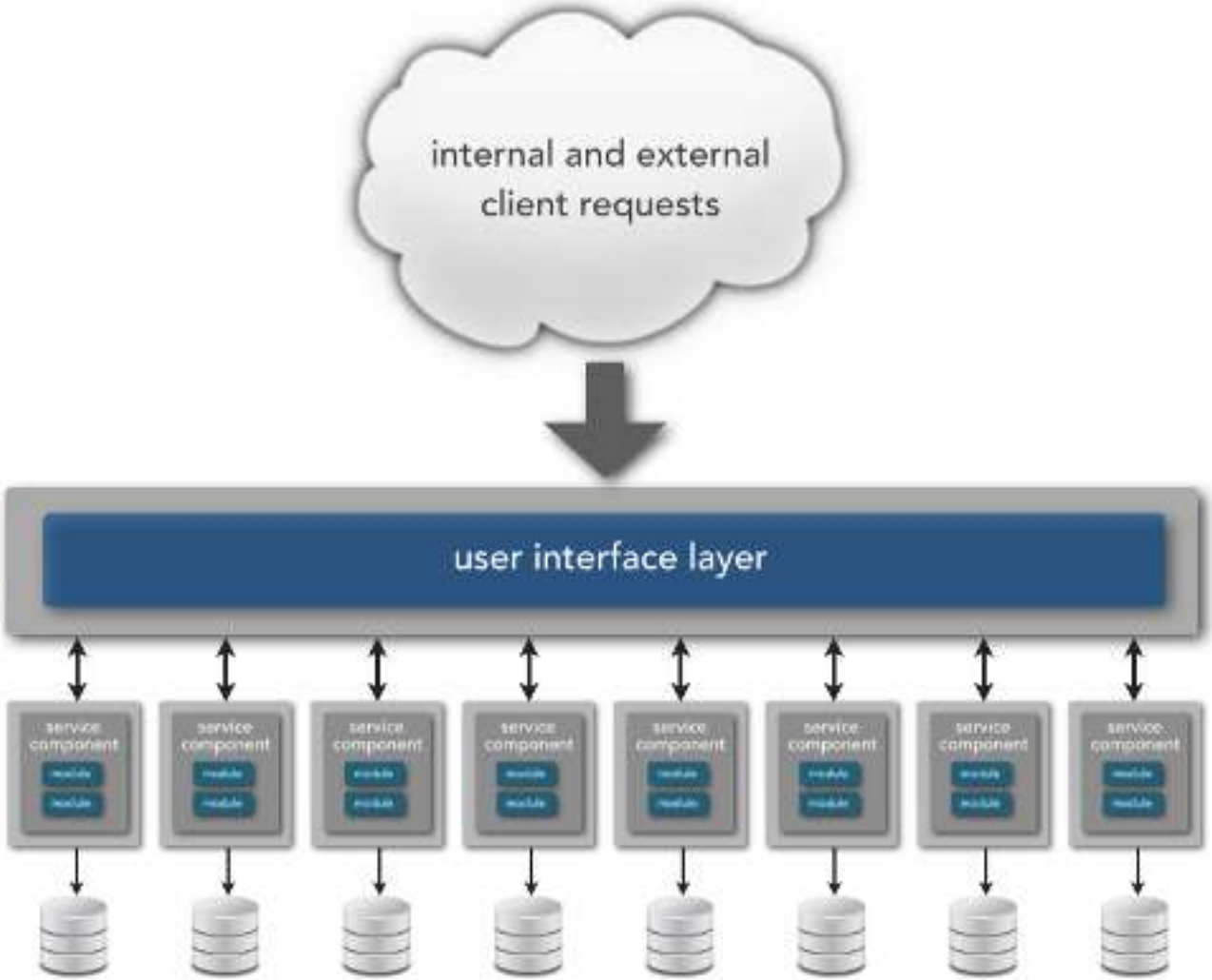
api layer


internal and external
client requests
feature toggle
api layer
service component
module

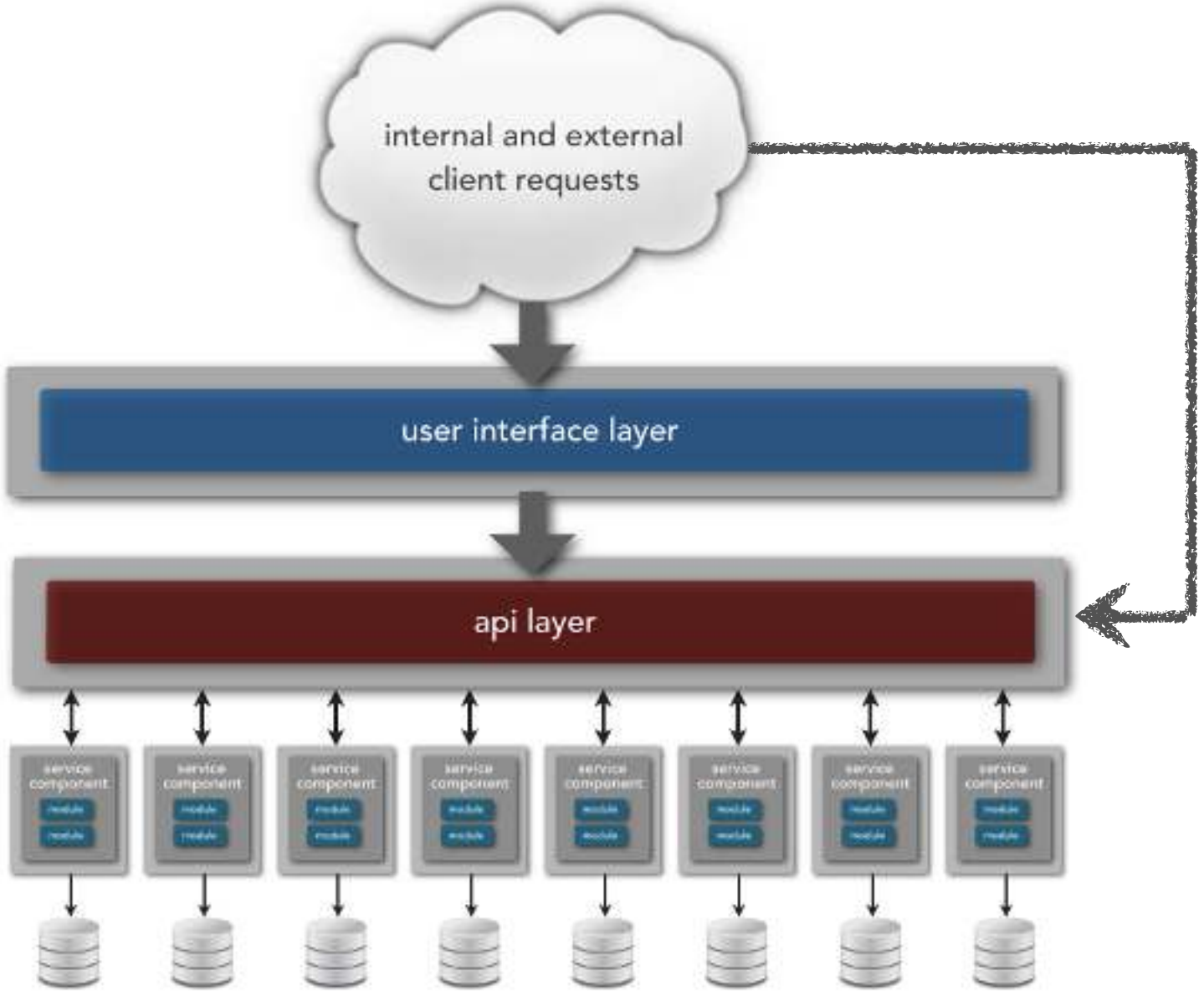
api layer


internal and external
client requests
api layer
api layer
service component
module
module

api layer

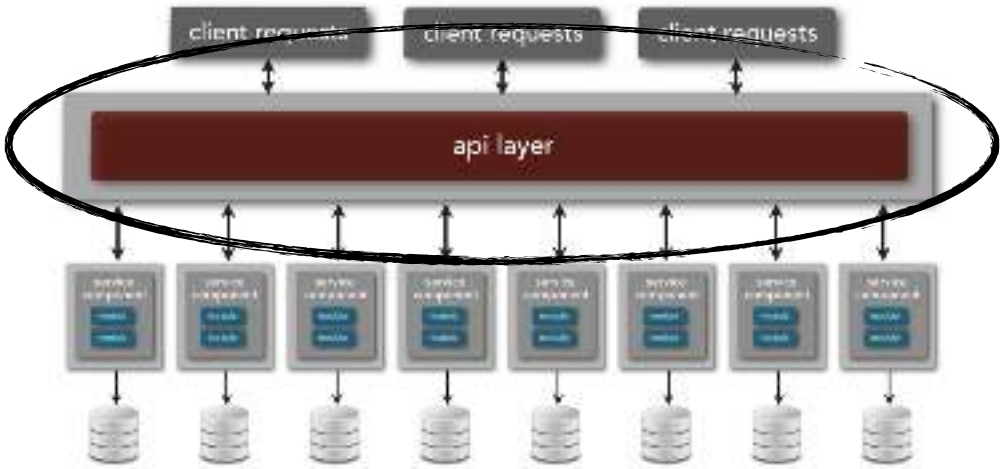

internal and external
client requests
user interface layer
service component
service component
service component
service component
service component
service component
service component
service component

microservices architecture



api layer

endpoint proxy



APACHE HTTP SERVER

HAPROXY

Microsoft IIS

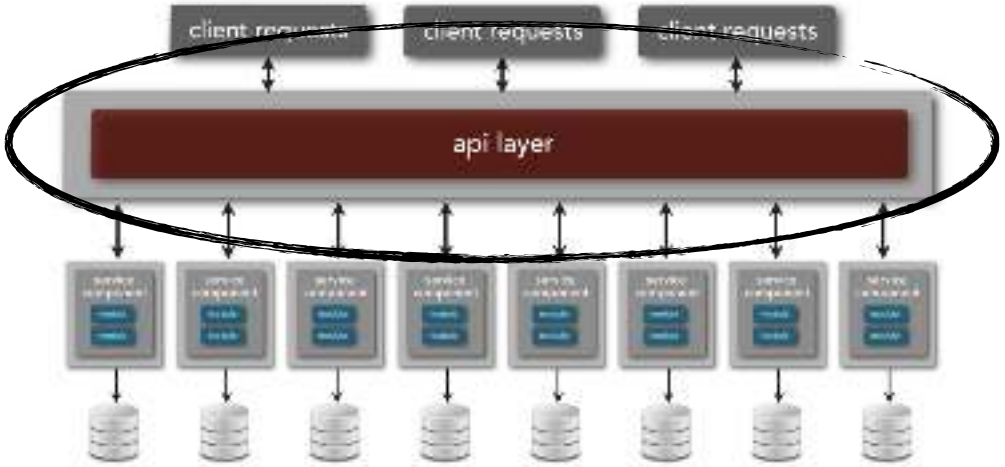
NGINX http server

PROXY SERVER Squid

ZIPROXY

api layer

load balancer



Barracuda

neustar

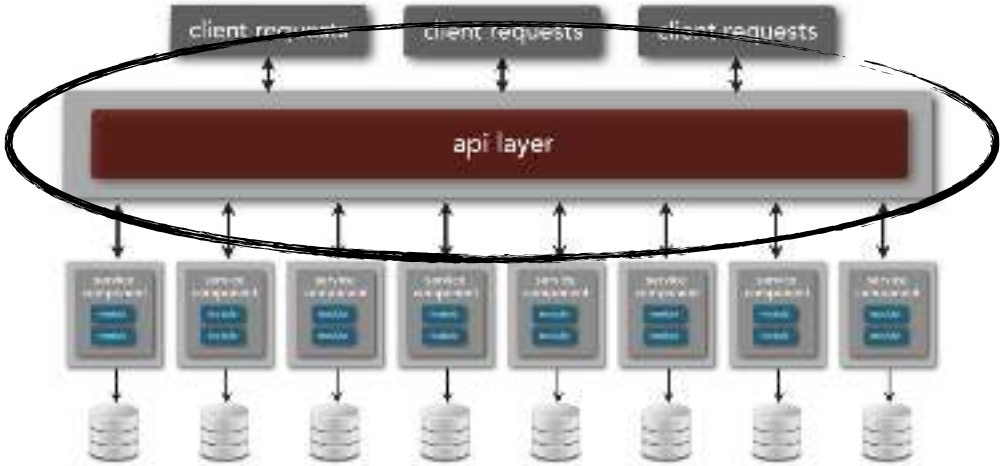
f5

CITRIX

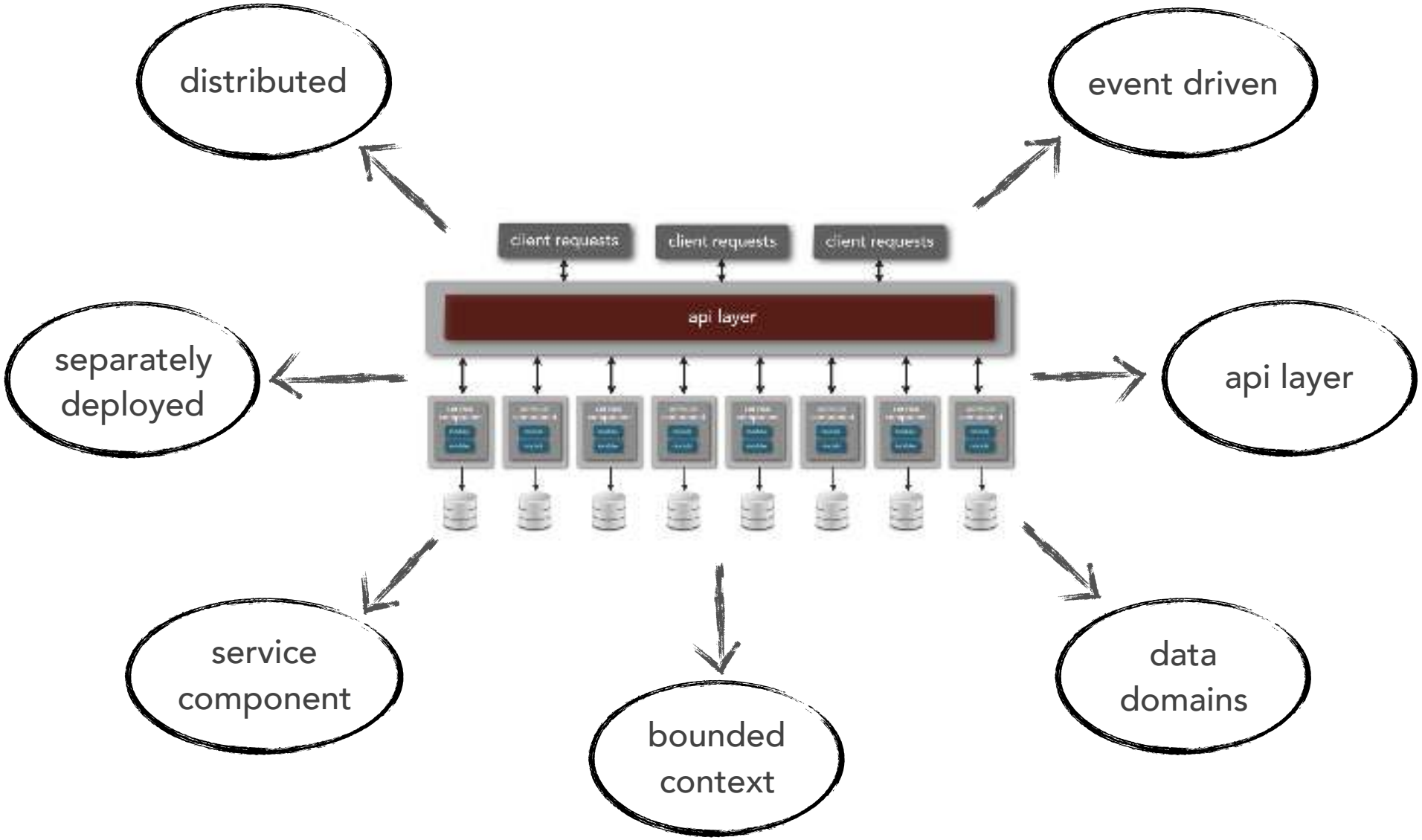
Citrix NetScaler

api layer

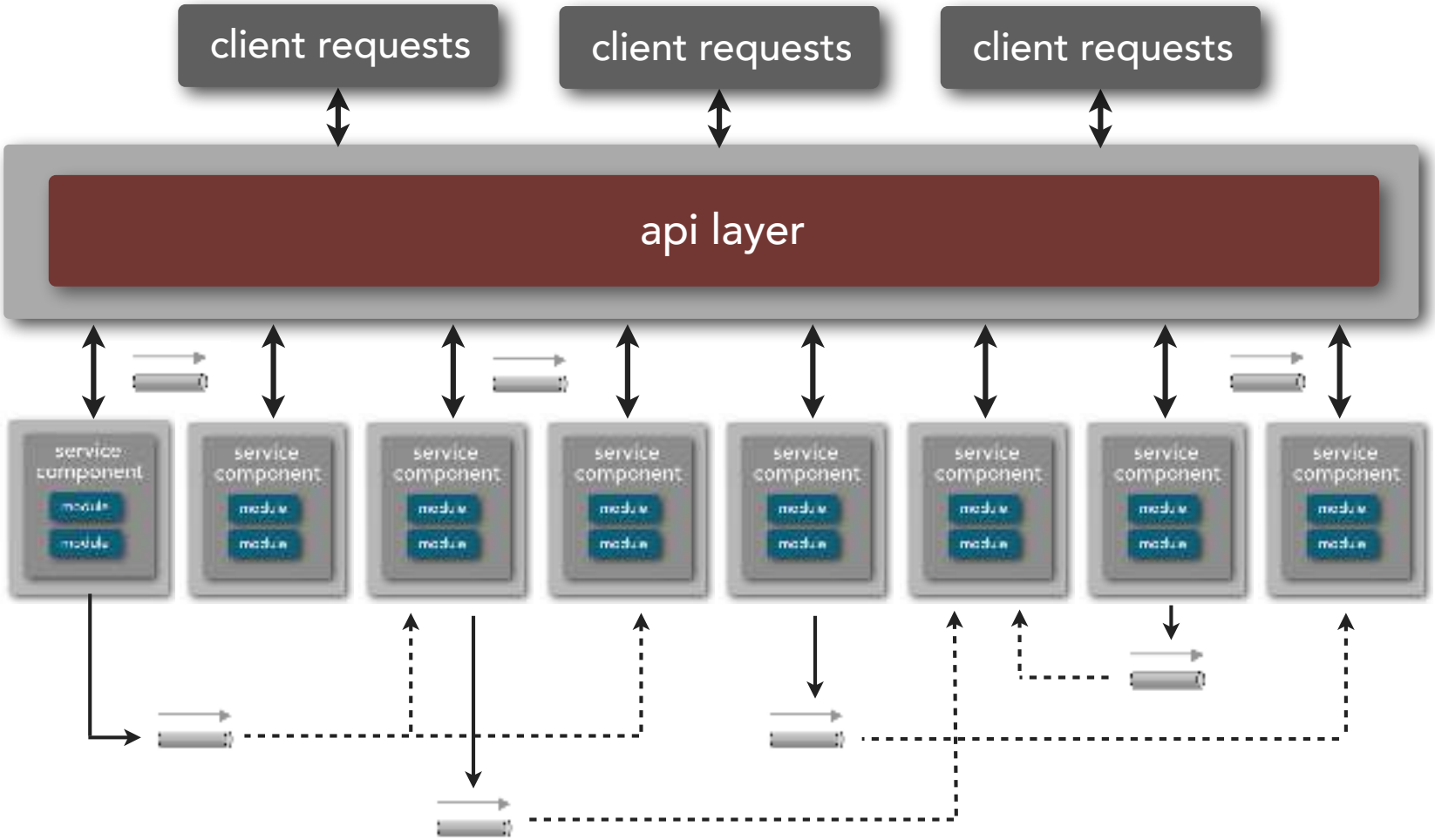
gateway (integration hub)



microservices architecture

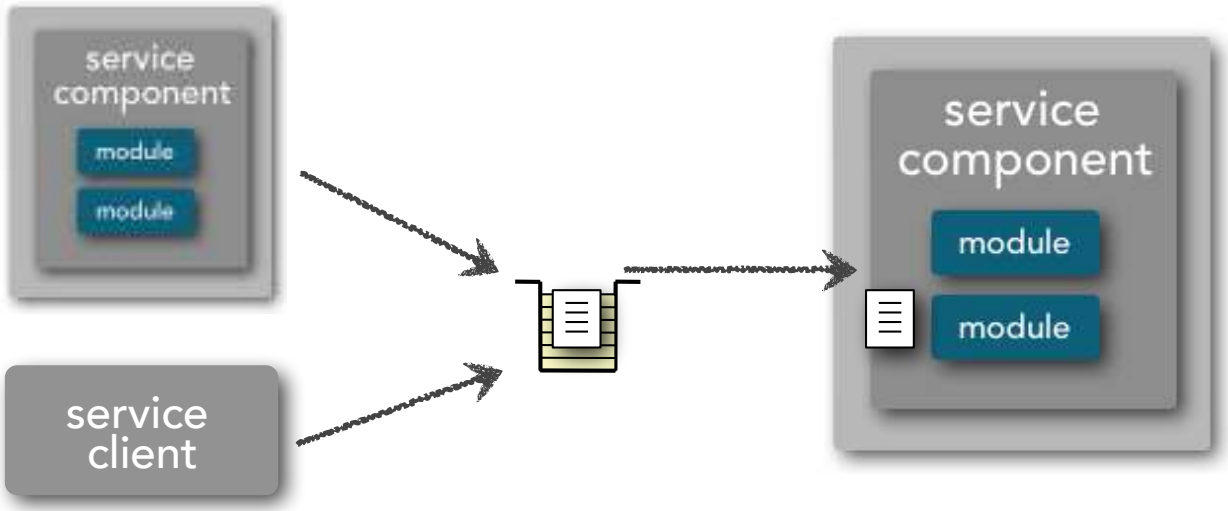


microservices architecture



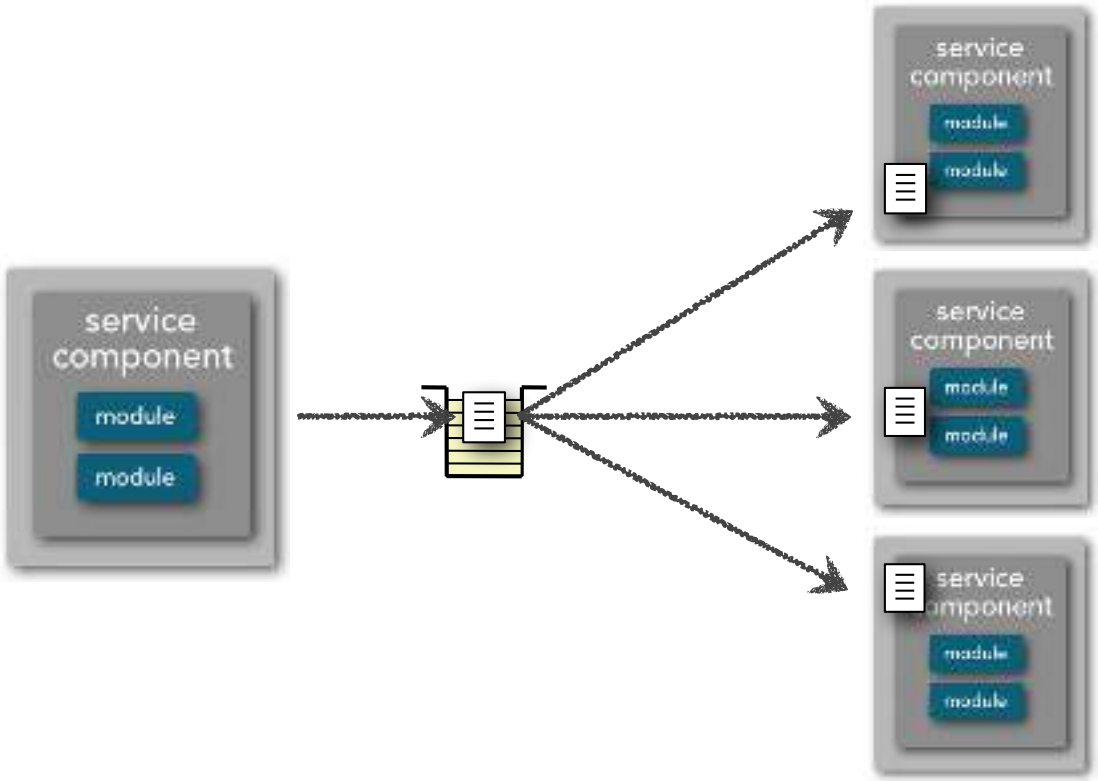
microservices architecture

asynchronous communications



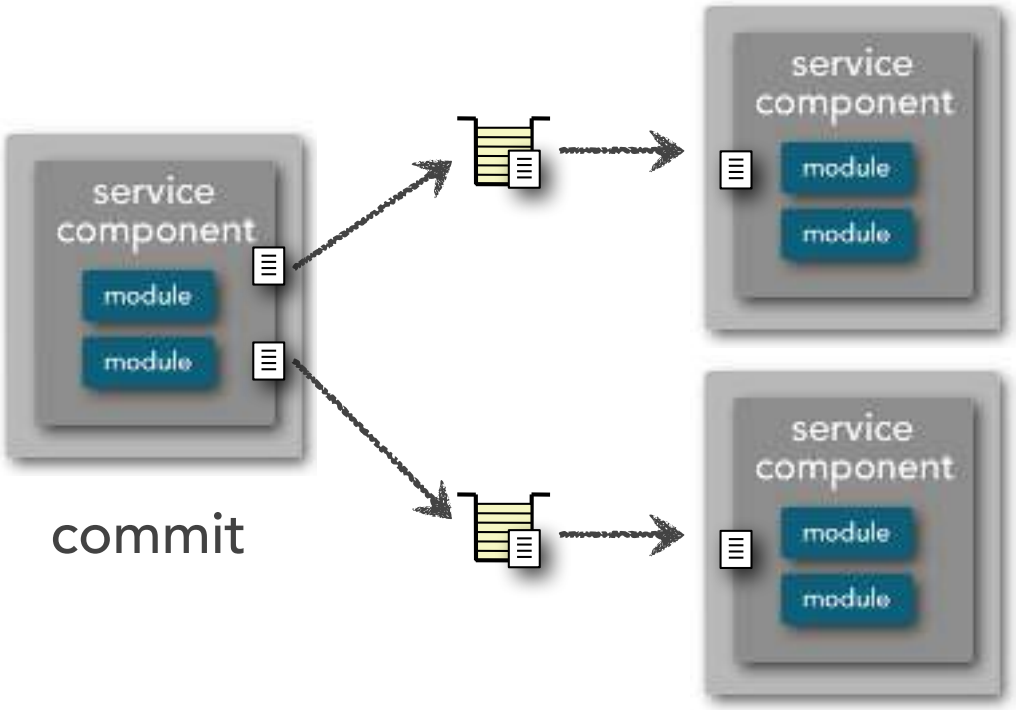
microservices architecture

broadcast capabilities



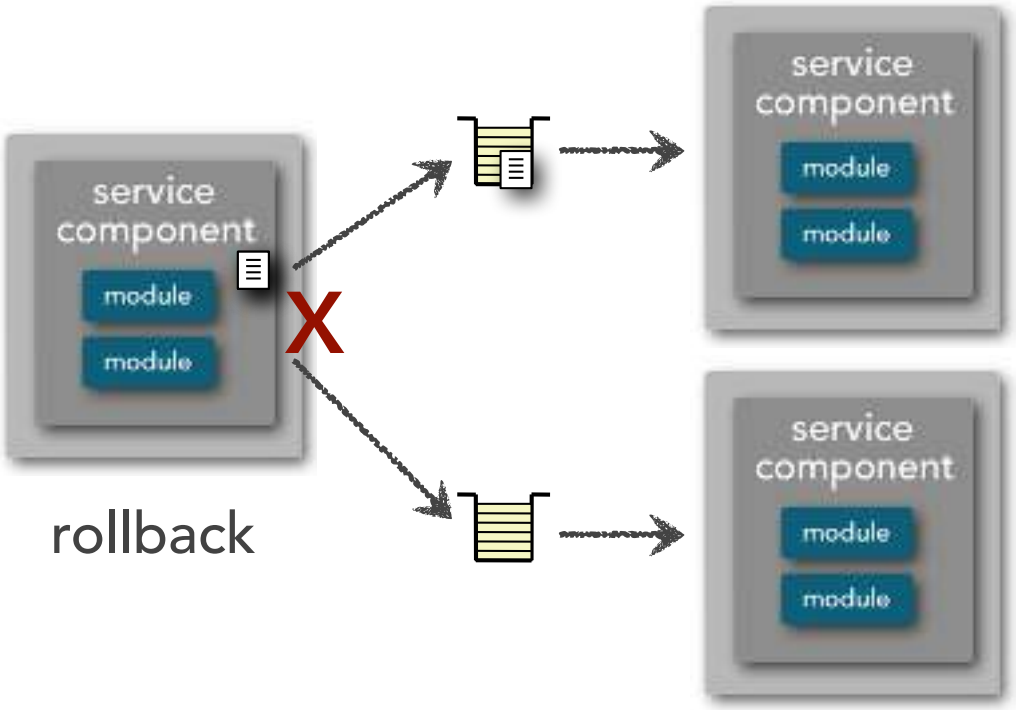
microservices architecture

transactional capabilities

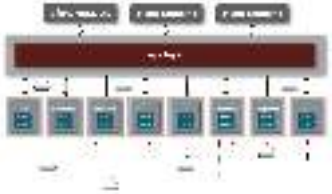


microservices architecture

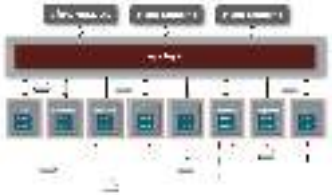
transactional capabilities



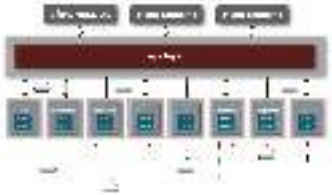
microservices architecture



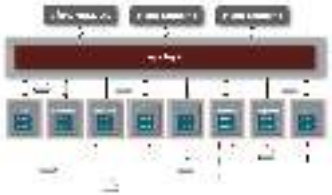
reporting techniques



eventual consistency patterns

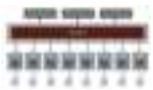
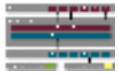


performance tuning



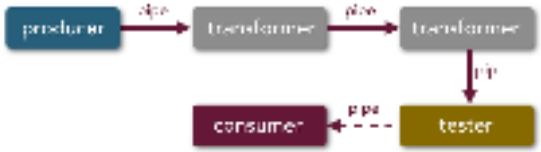
reactive architecture patterns

microservices architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👎	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
							
							

pipeline architecture

pipeline vs. event-driven



synchronous data filtering

always unidirectional

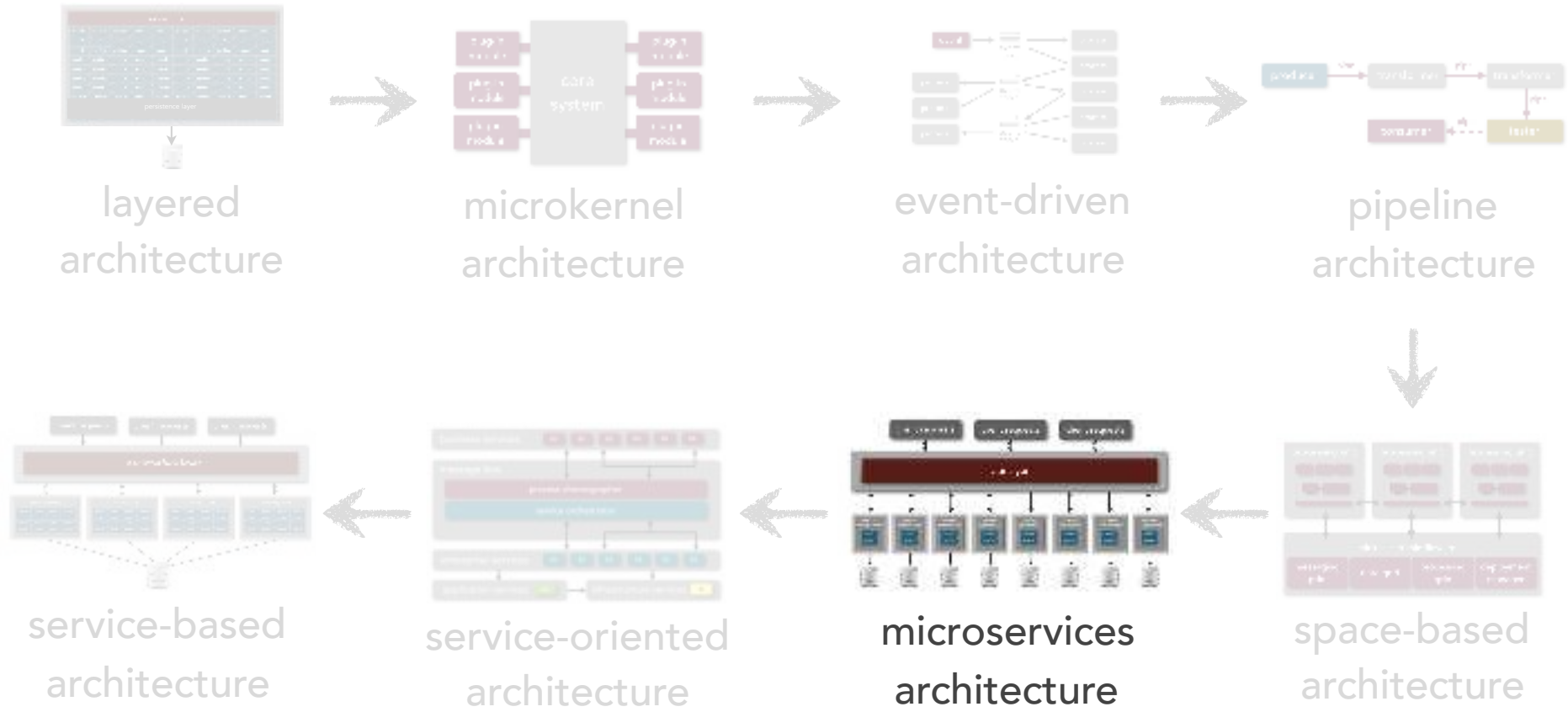
simple single purpose filters

asynchronous event processing

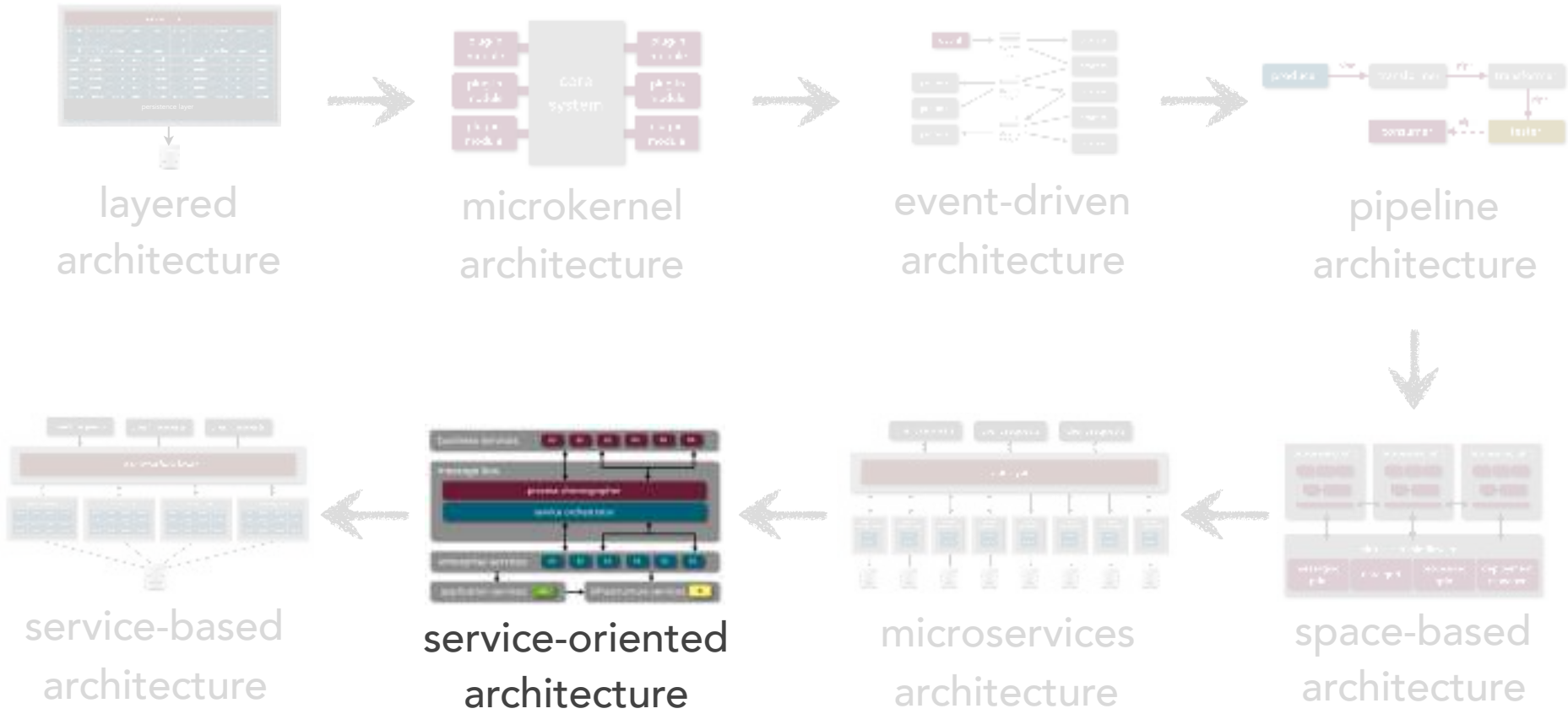
can be request/reply

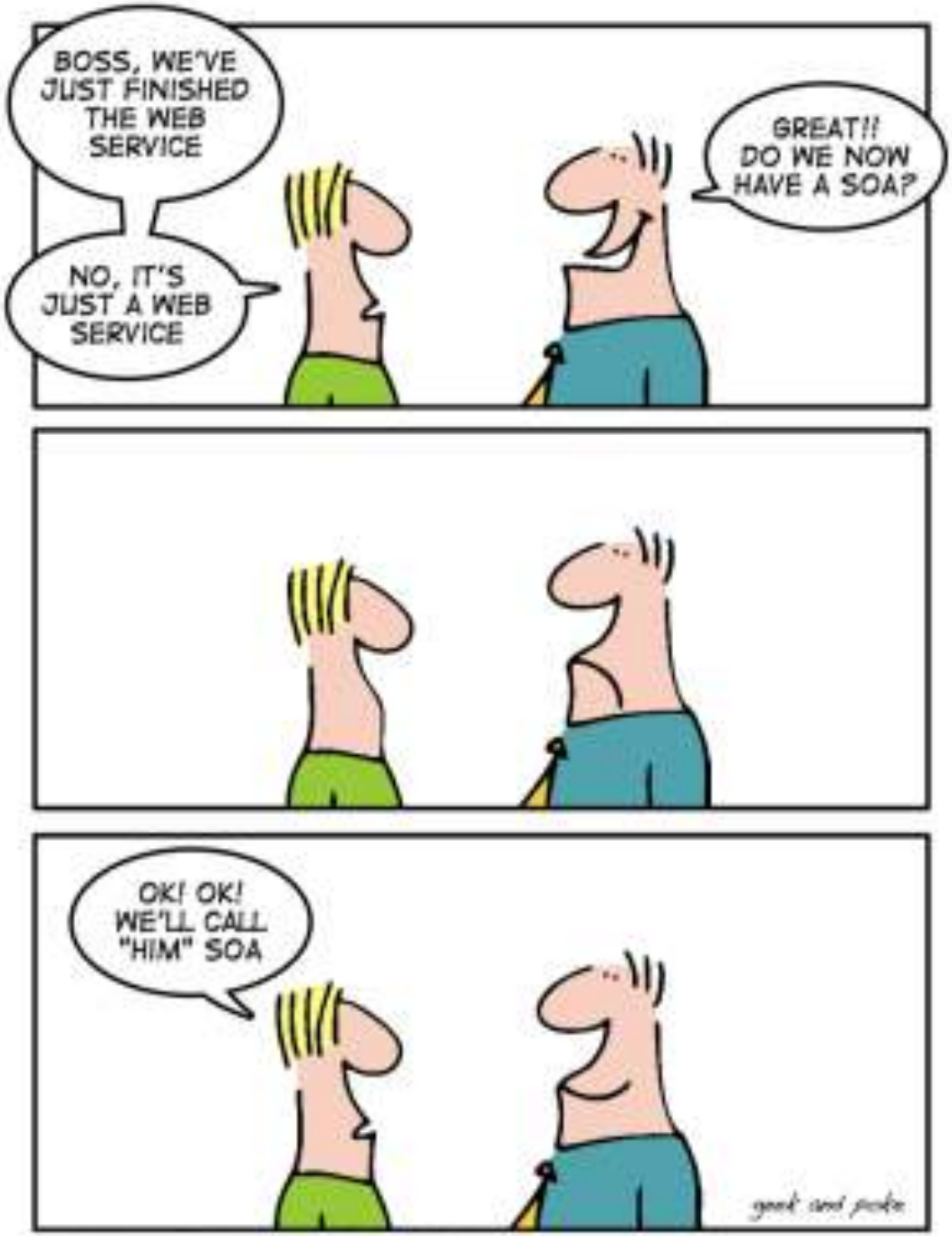
complex multi-purpose processors

architecture pattern roadmap



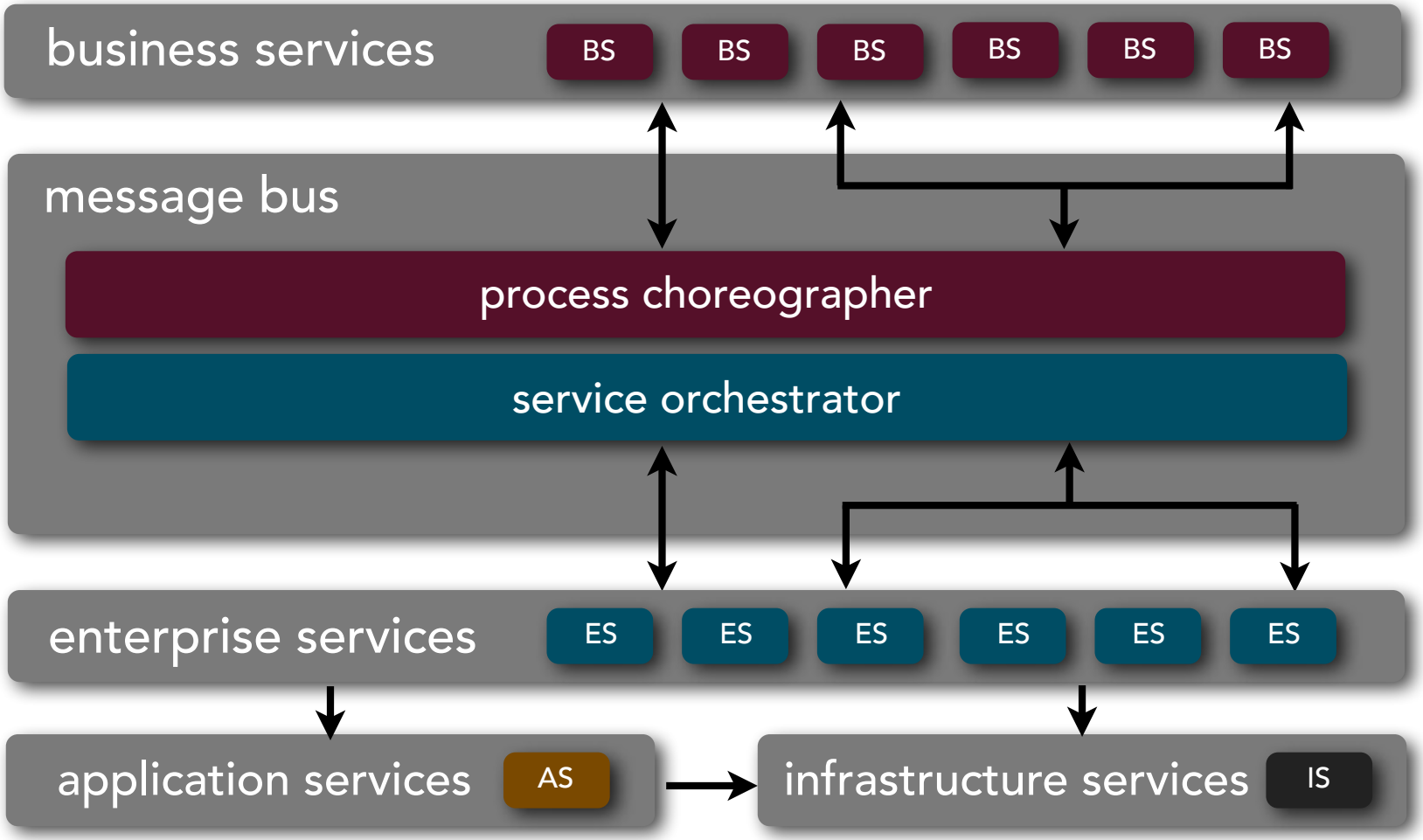
architecture pattern roadmap



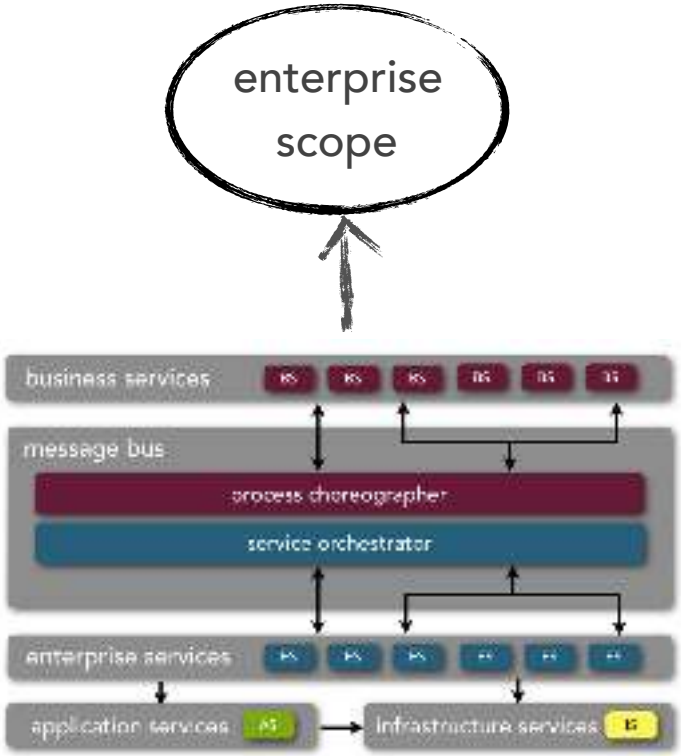


HOW TO GET A SOA

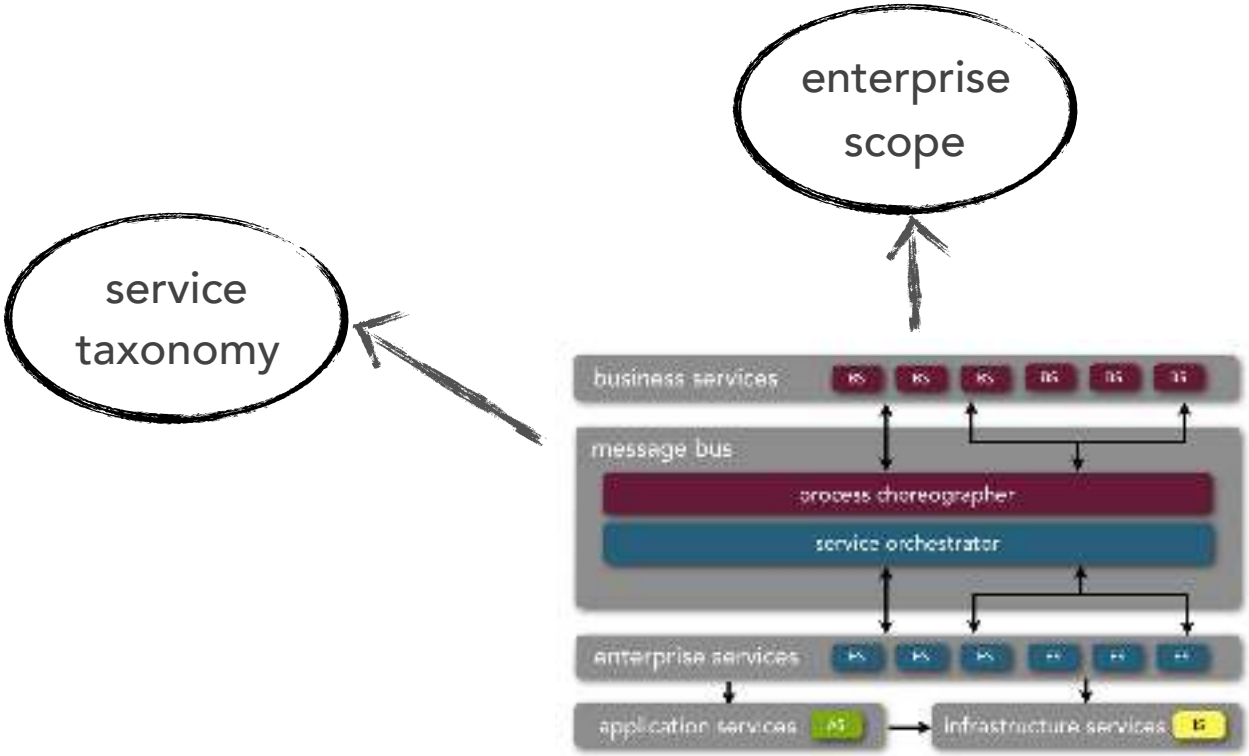
service-oriented architecture



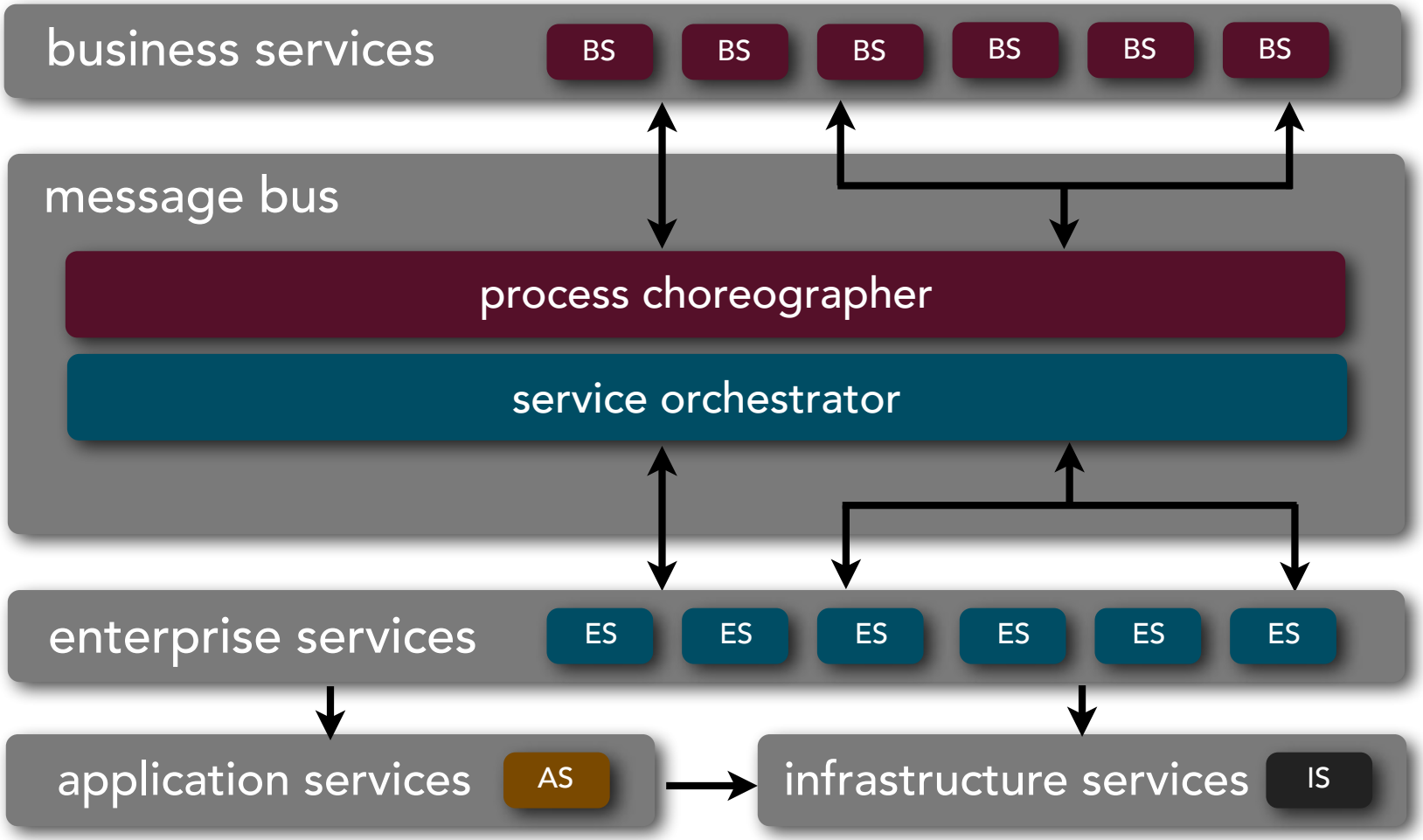
service-oriented architecture



service-oriented architecture

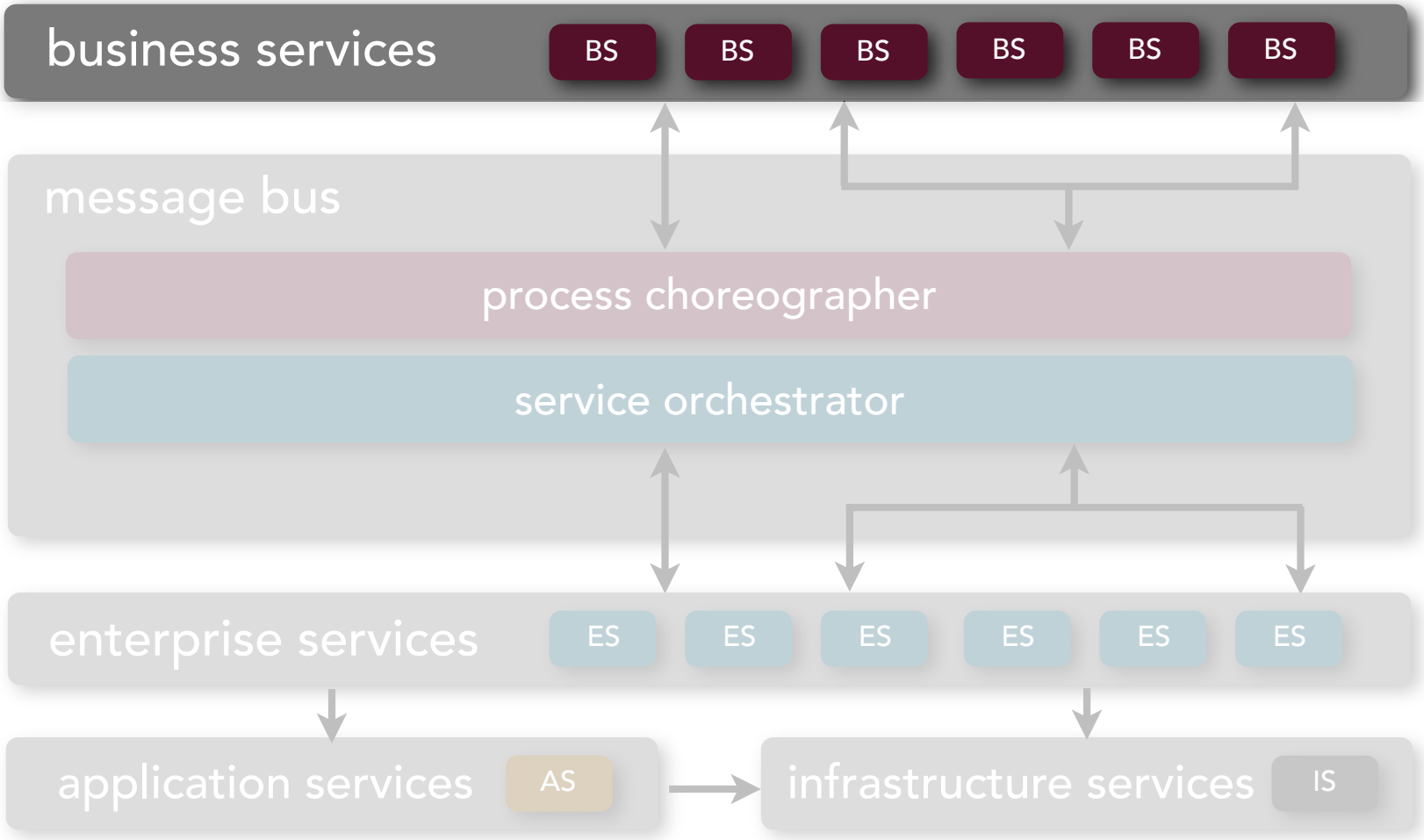


service-oriented architecture



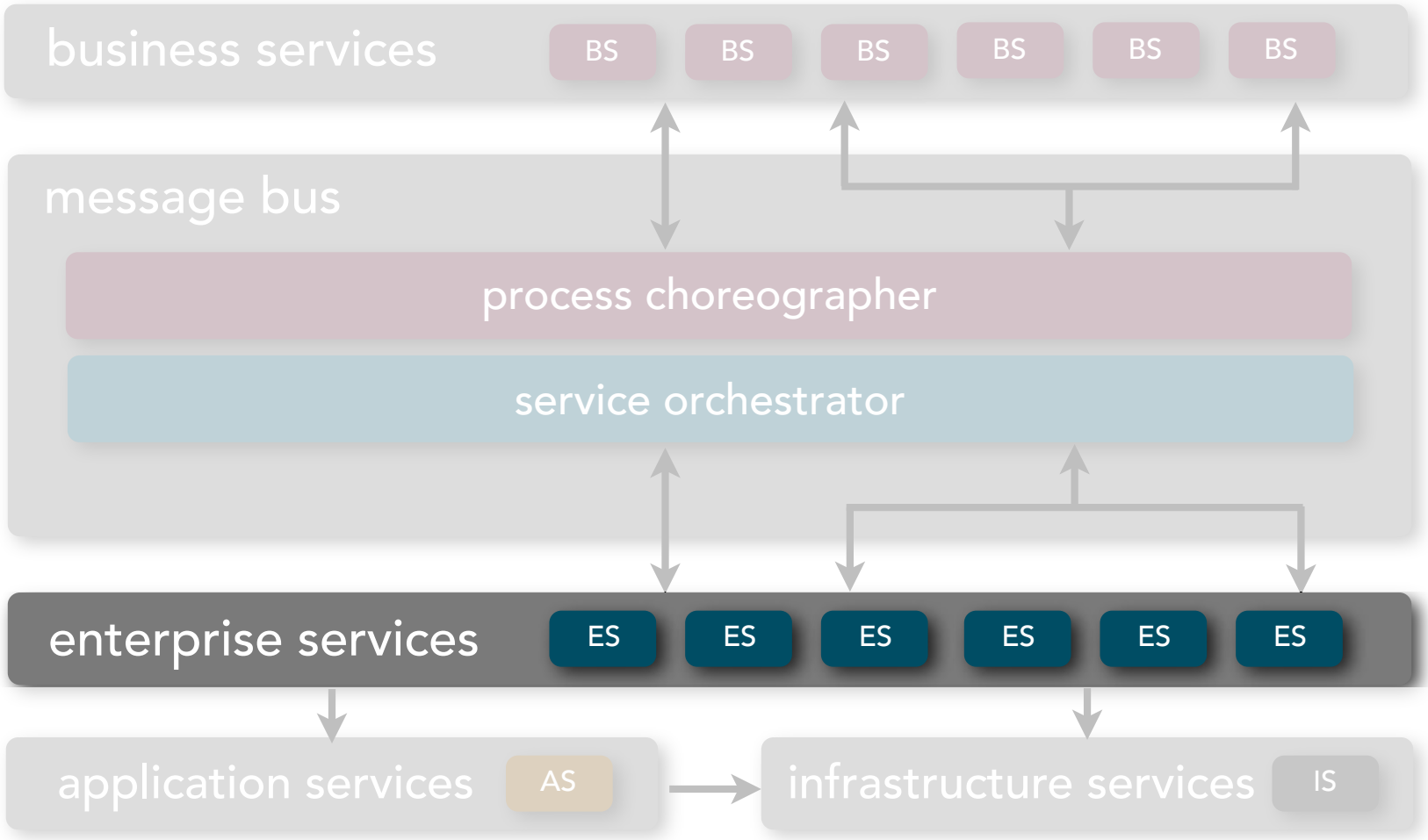
service-oriented architecture

business services



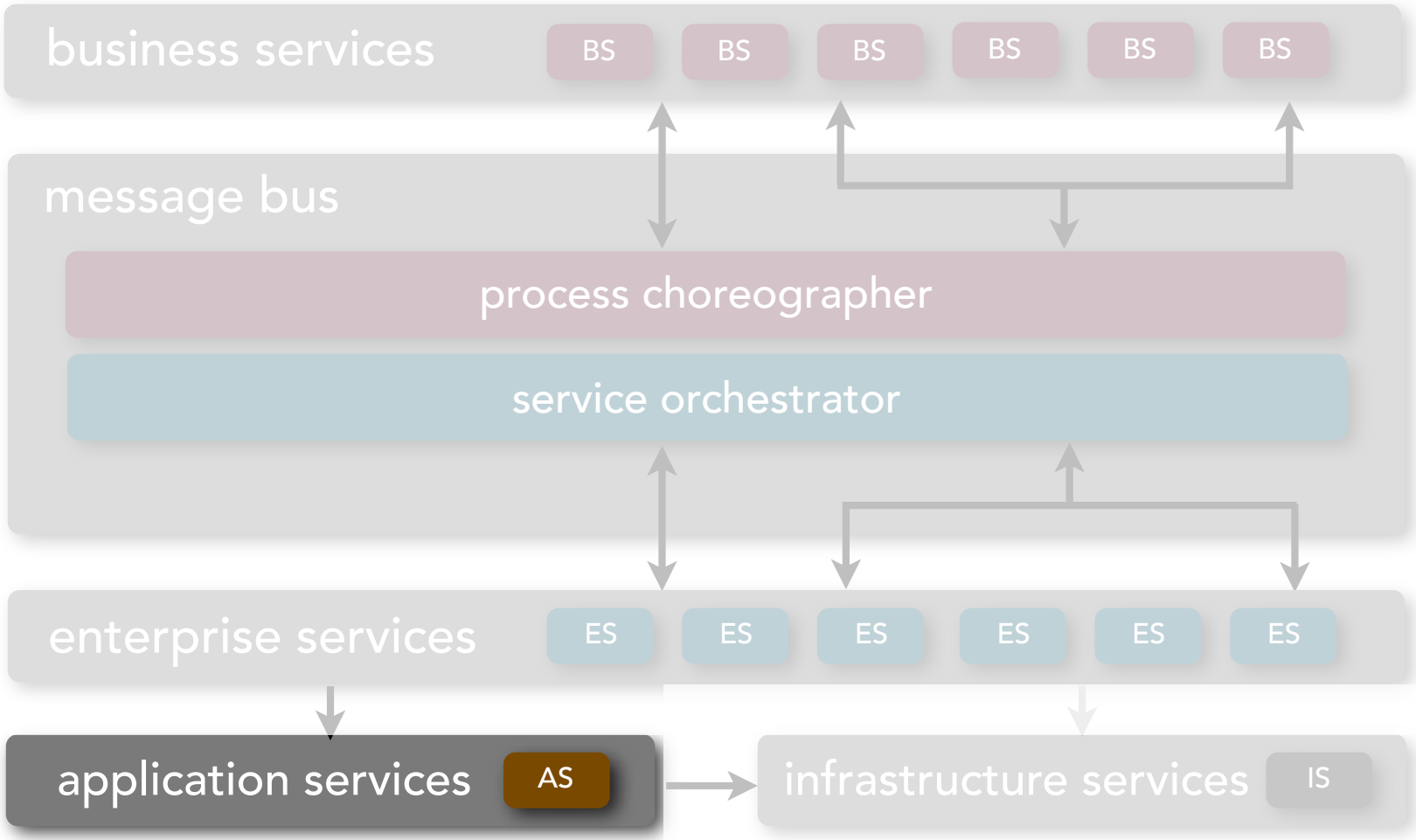
service-oriented architecture

enterprise services



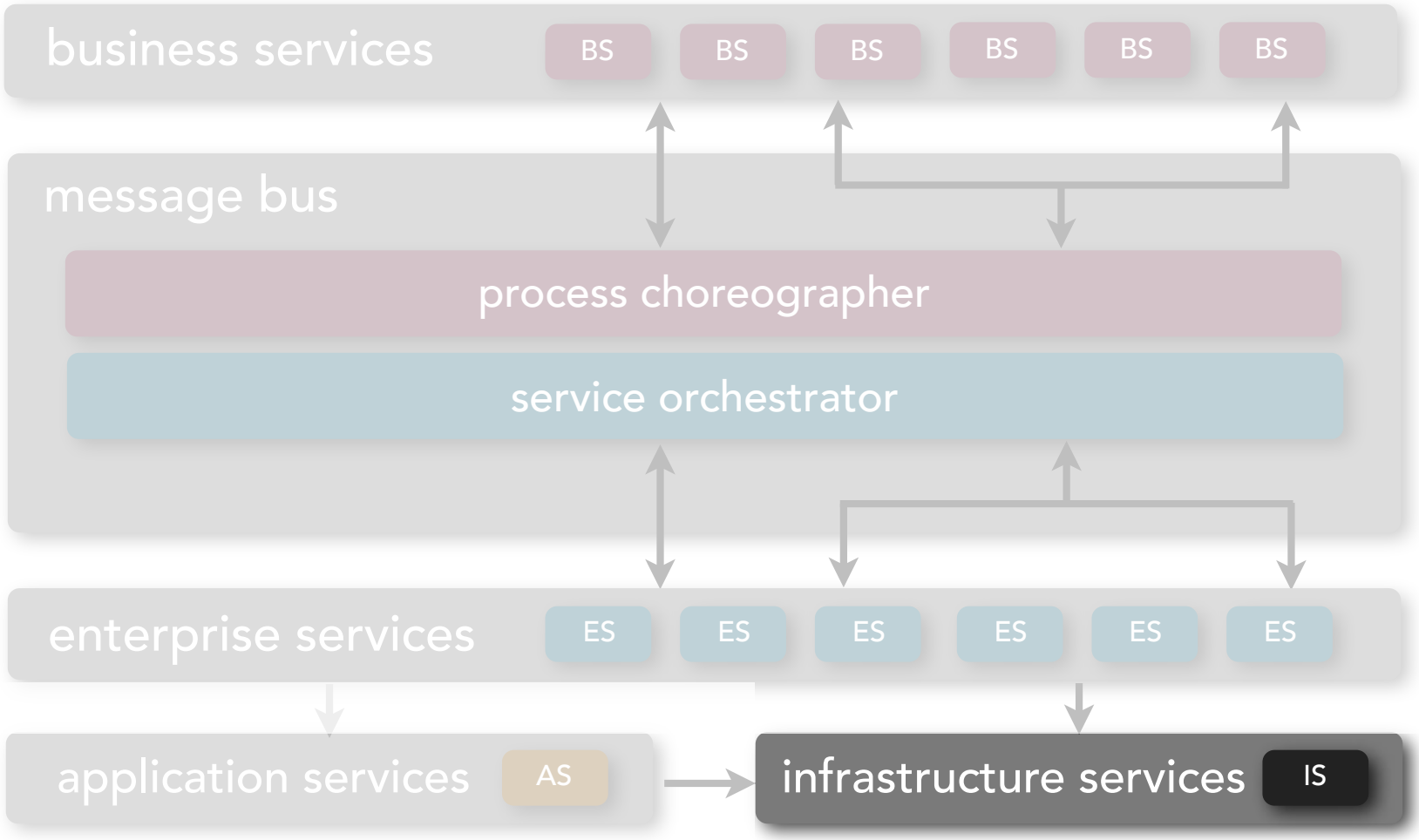
service-oriented architecture

application services

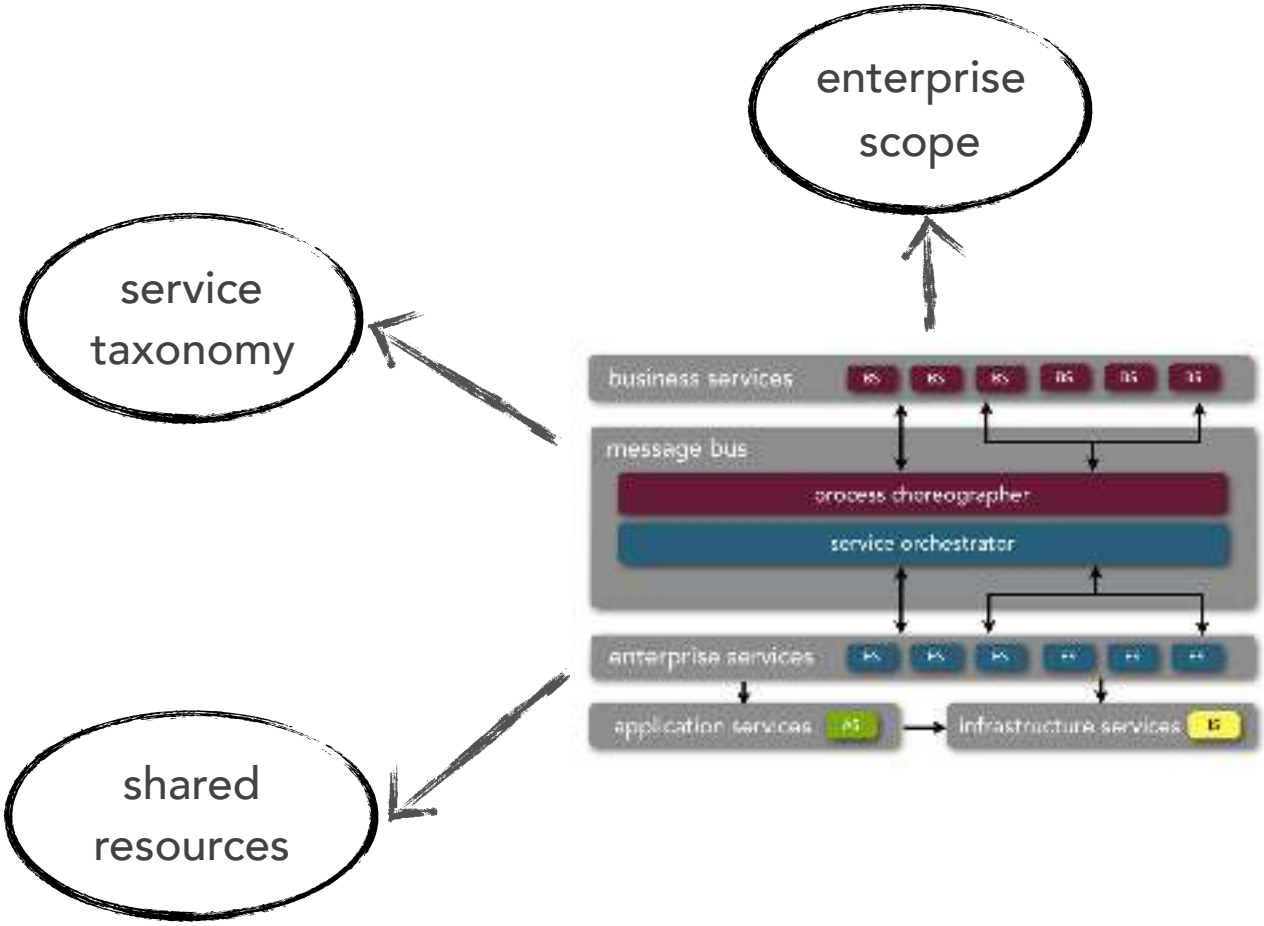


service-oriented architecture

infrastructure services

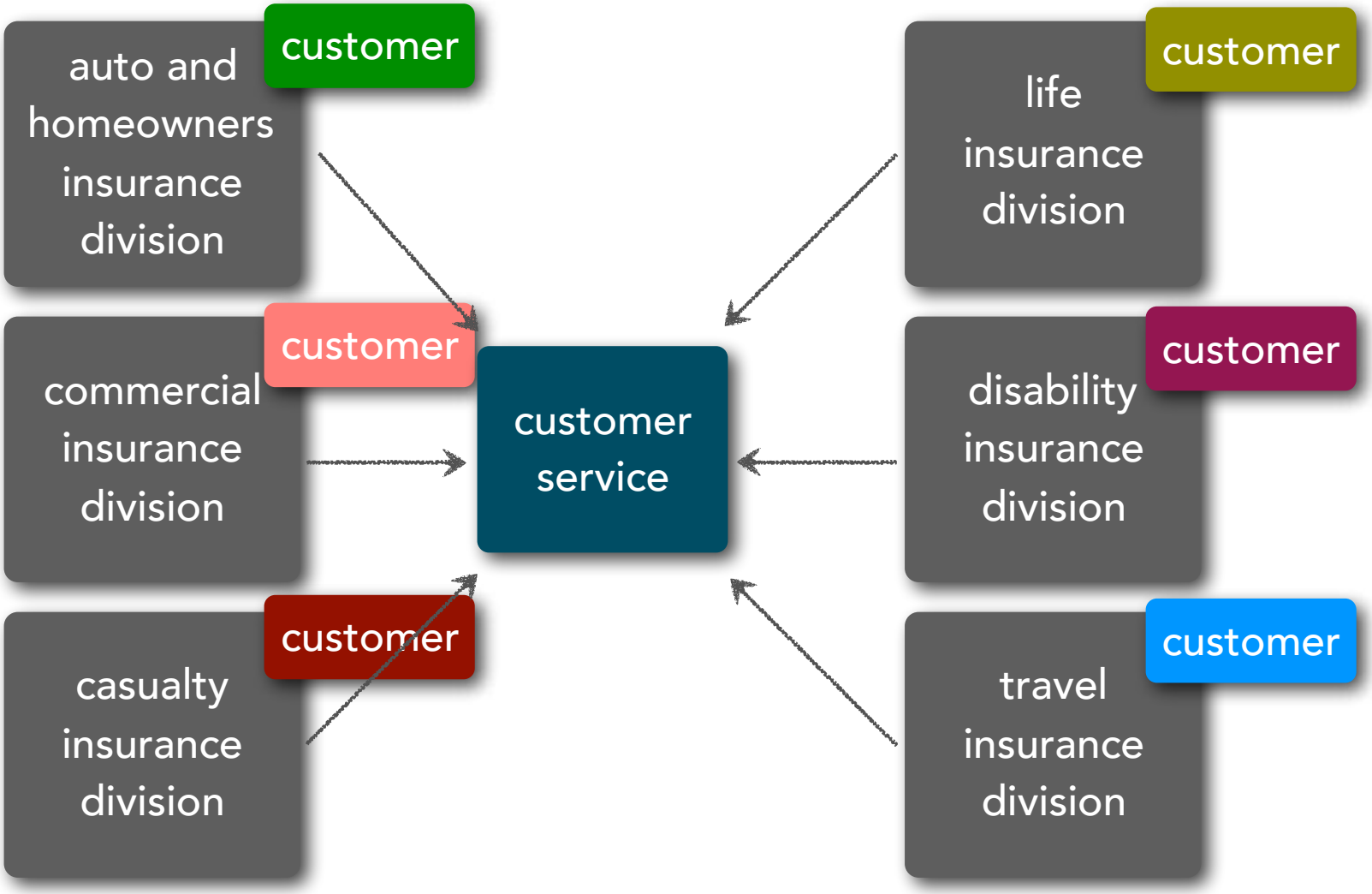


service-oriented architecture

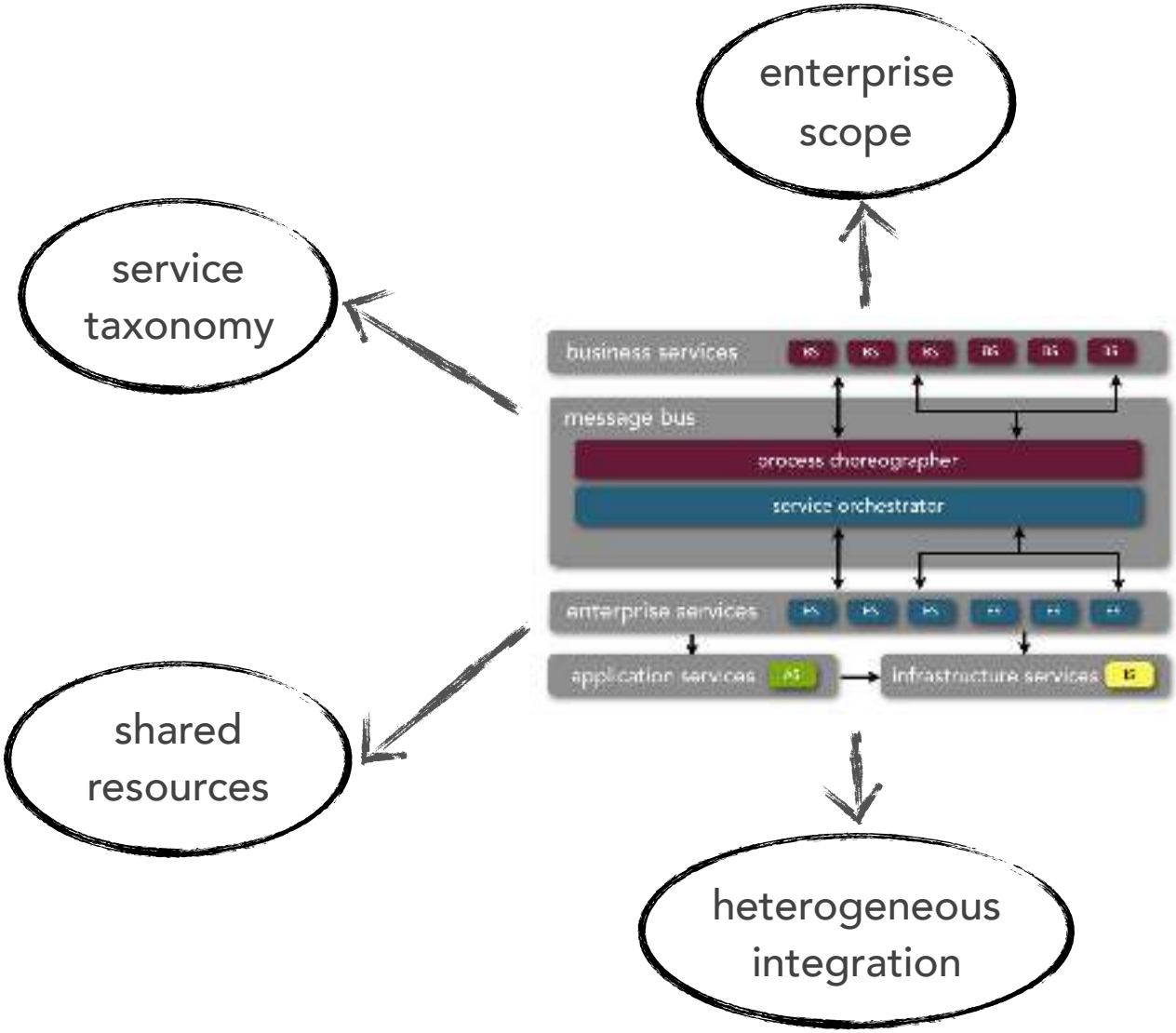


service-oriented architecture

shared resources

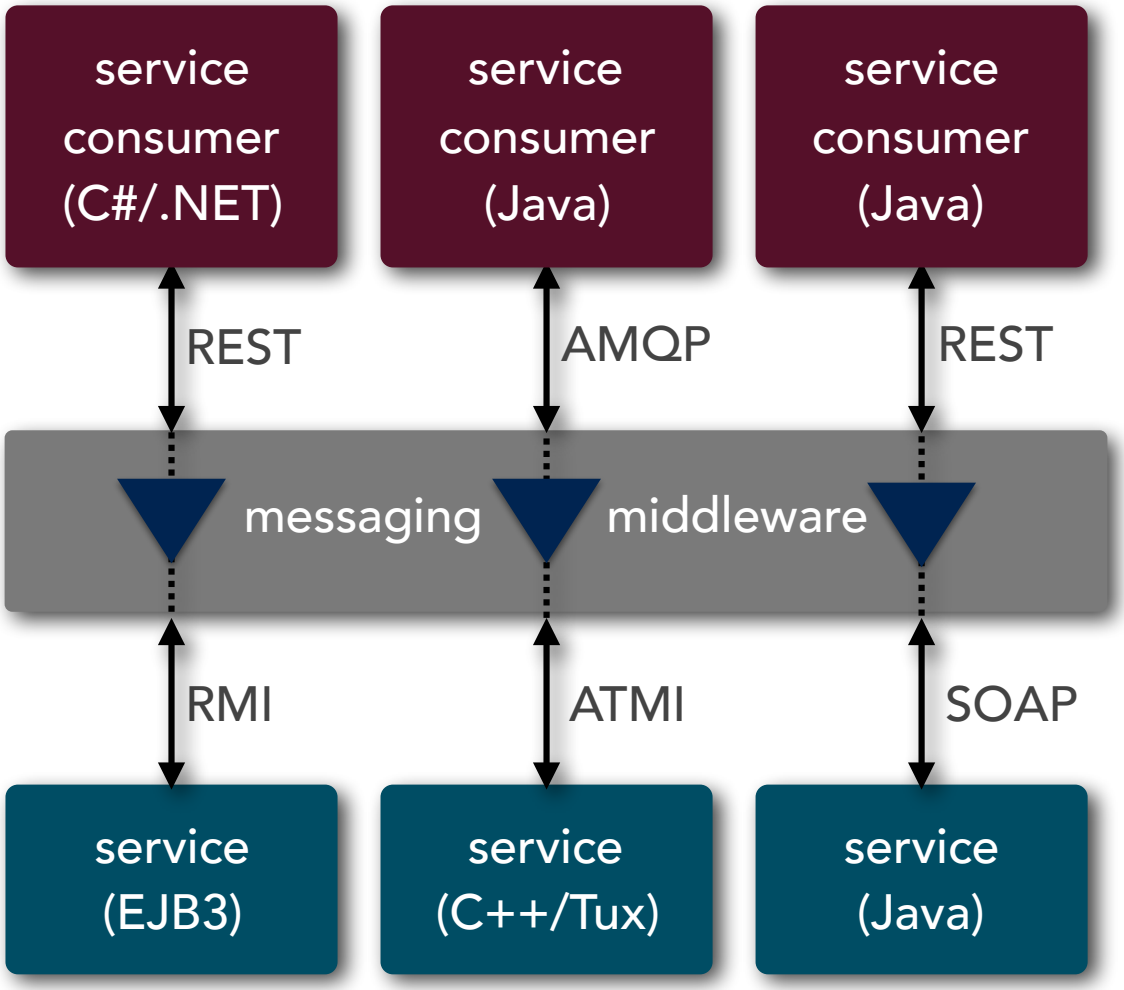


service-oriented architecture

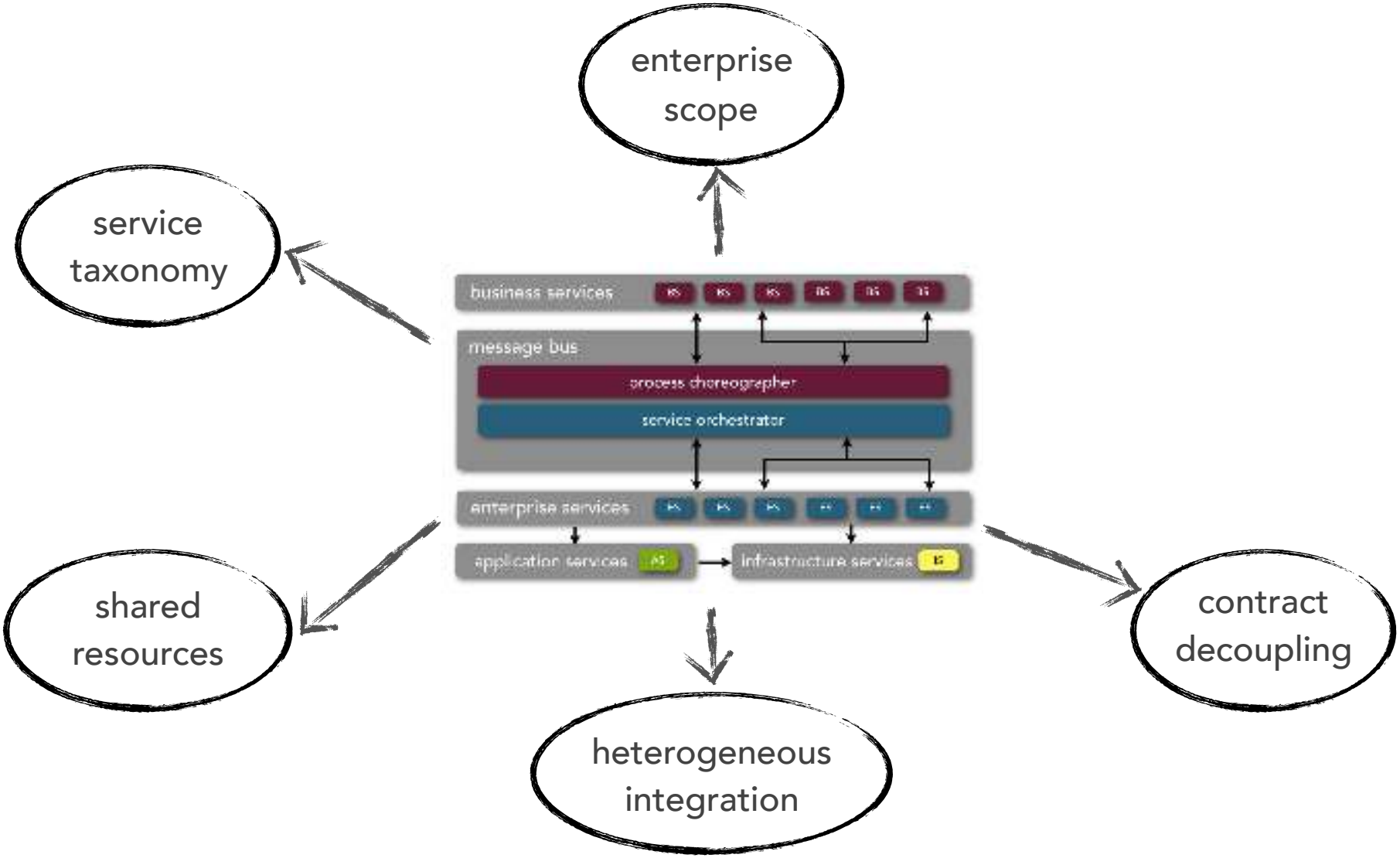


service-oriented architecture

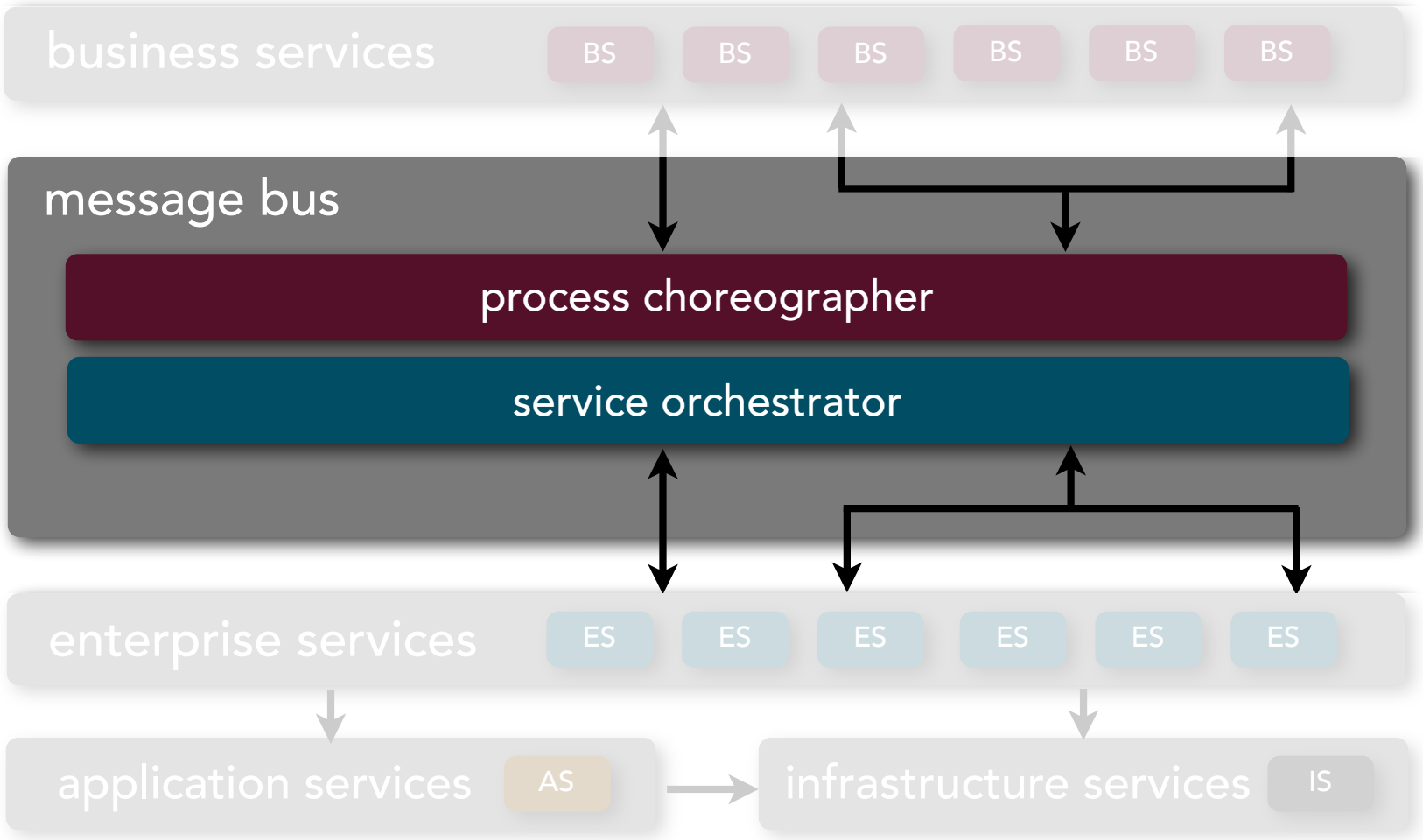
protocol-agnostic heterogeneous interoperability



service-oriented architecture

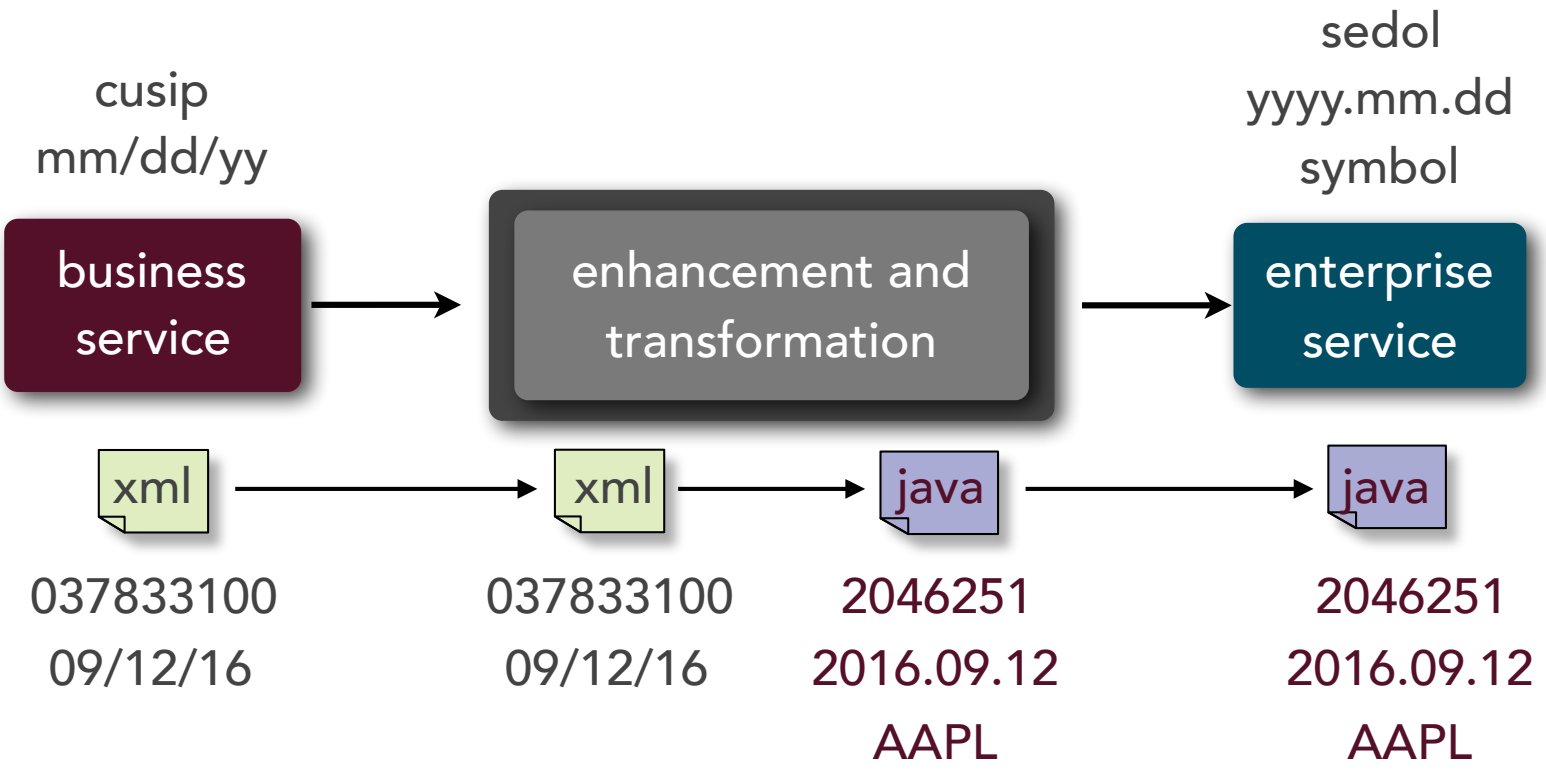


service-oriented architecture

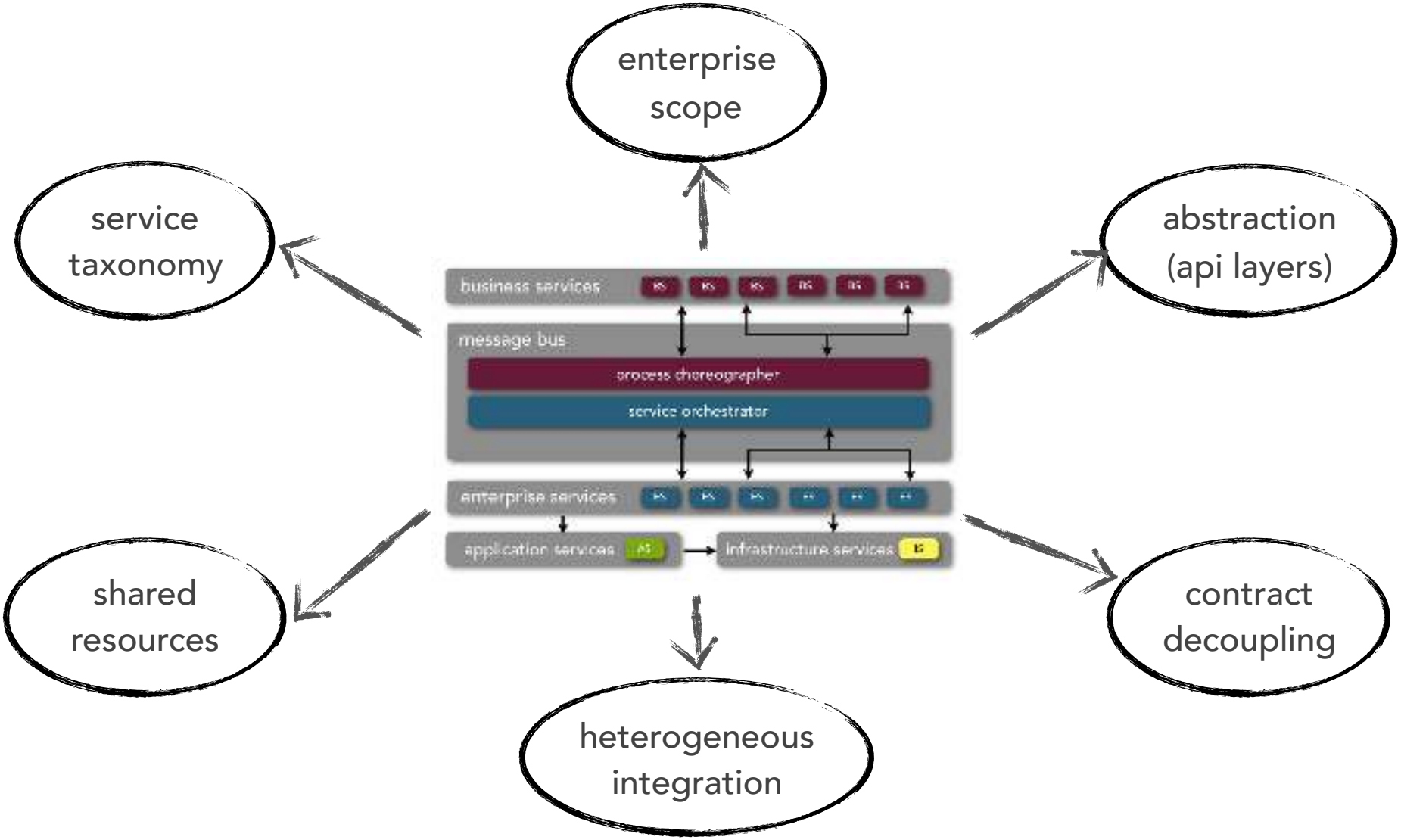


service-oriented architecture

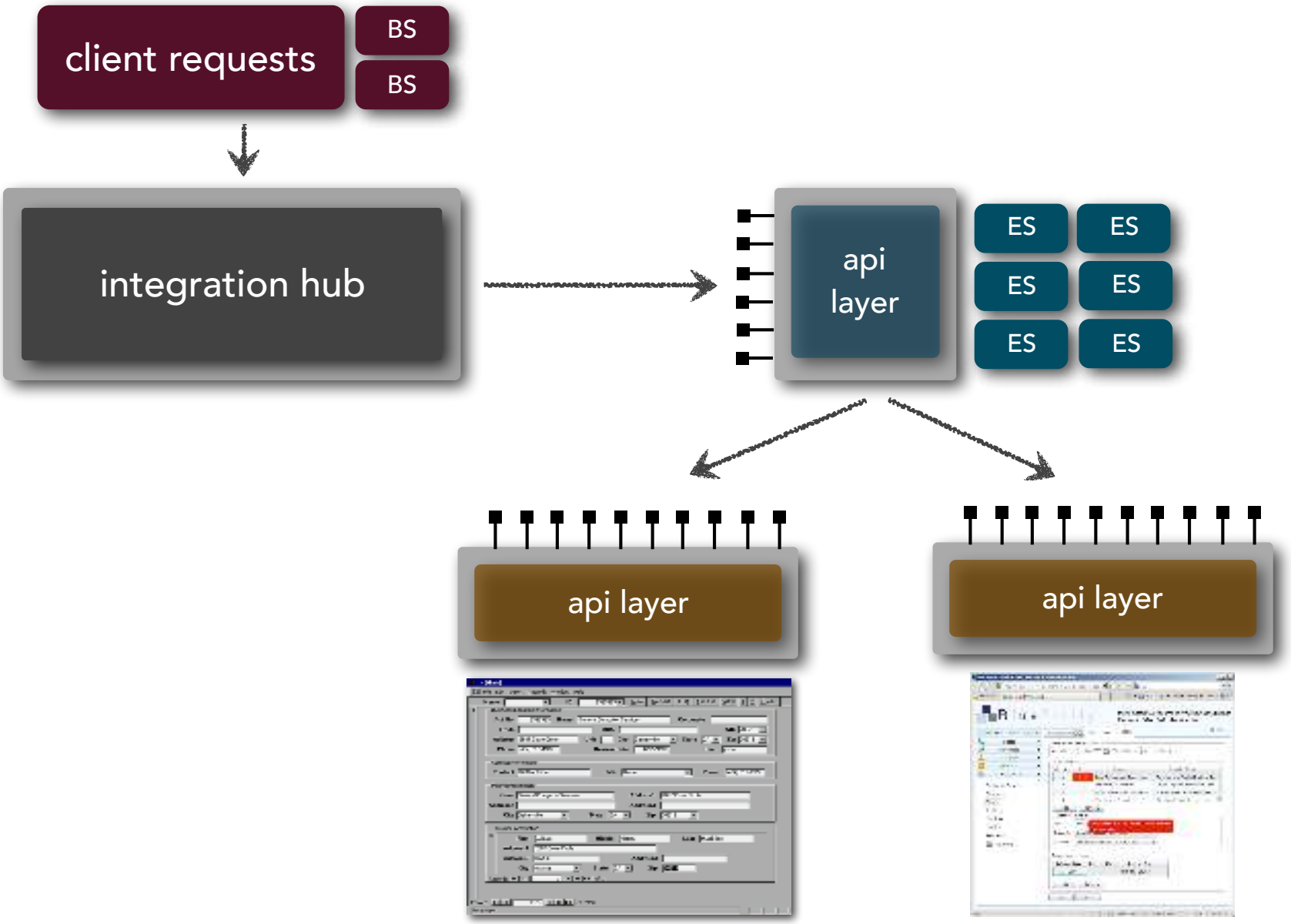
contract decoupling



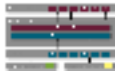
service-oriented architecture



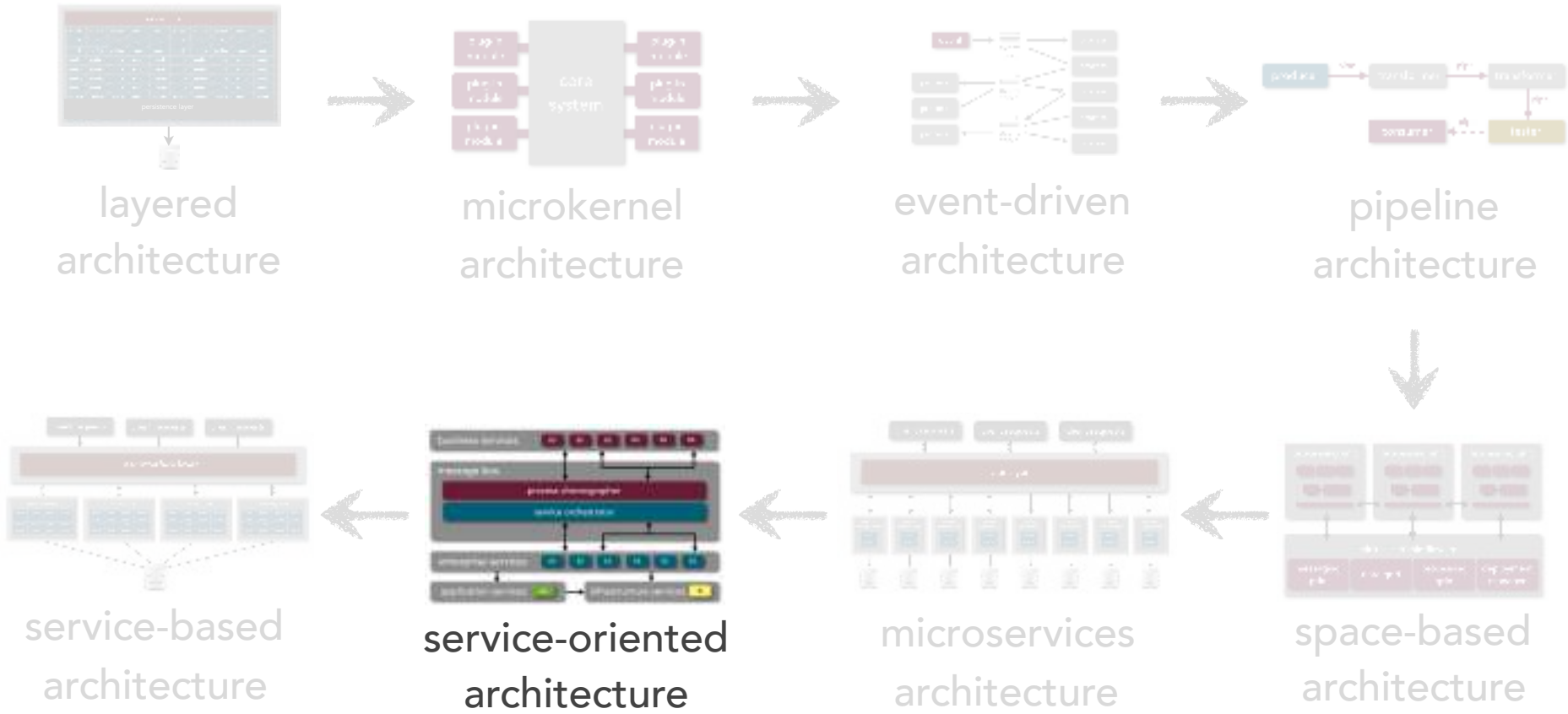
service-oriented architecture



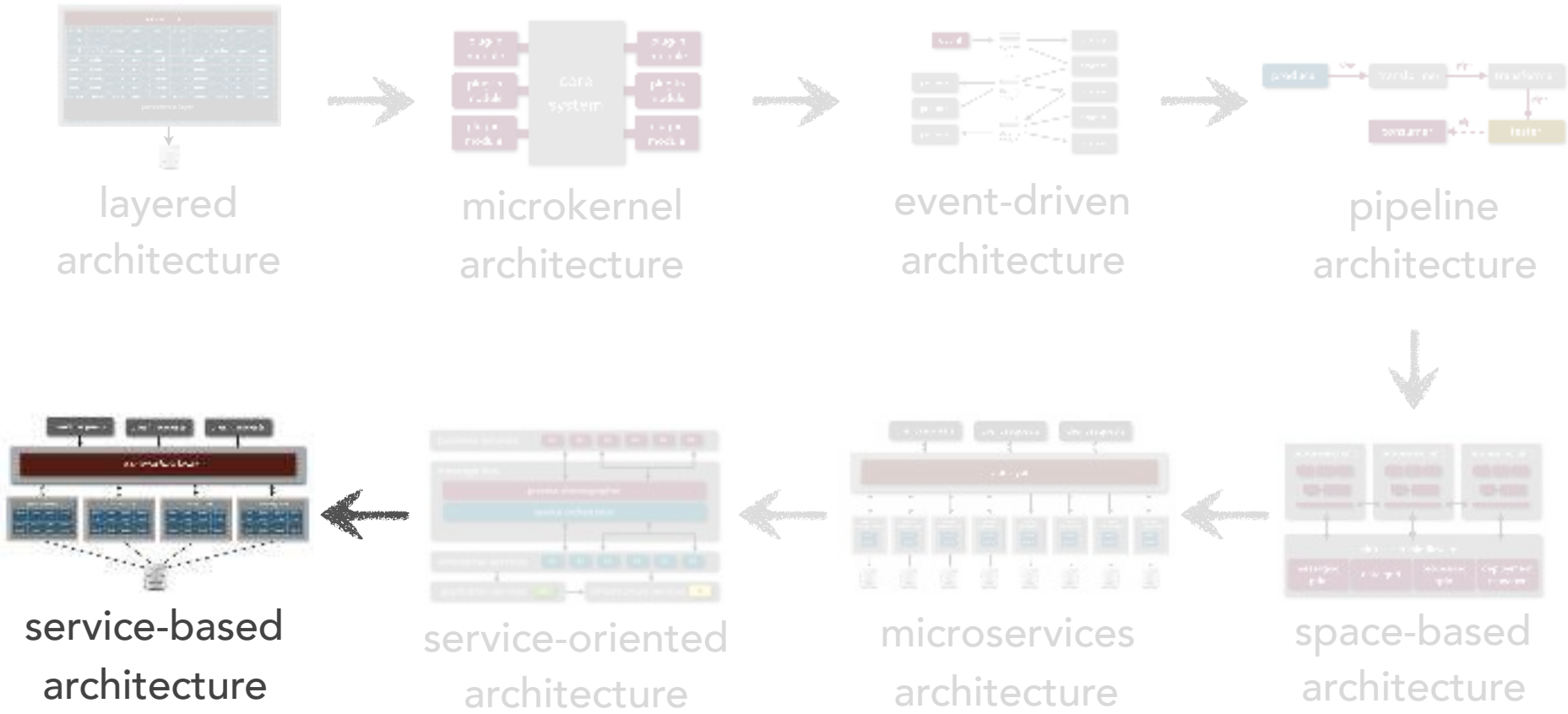
service-oriented architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👎	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
							

architecture pattern roadmap

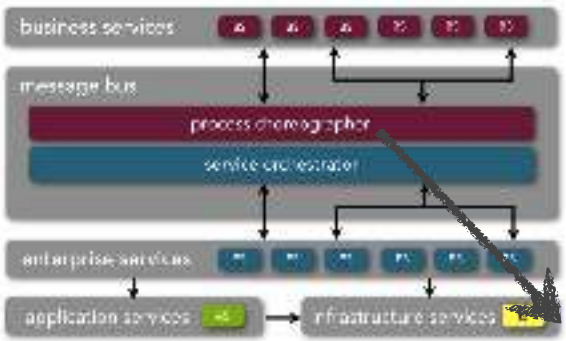


architecture pattern roadmap



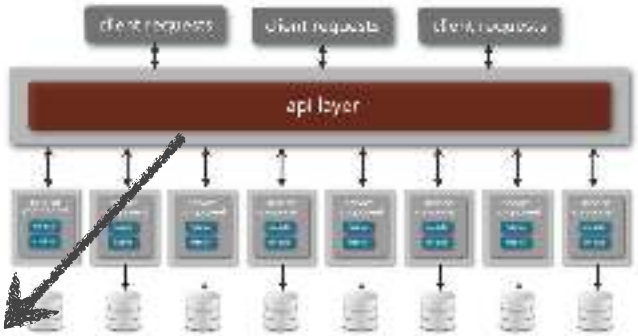
service-based architecture

is there a middle ground?



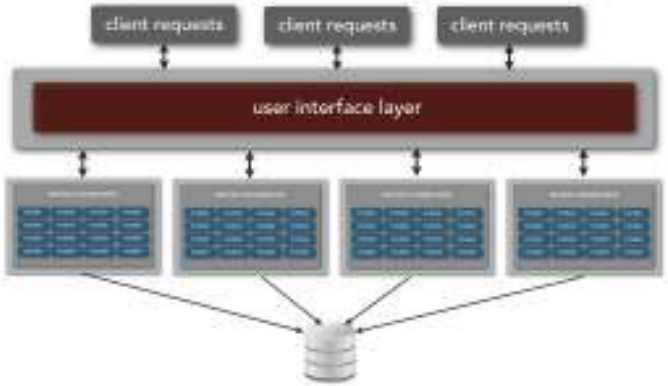
service-ori

architec



roservices

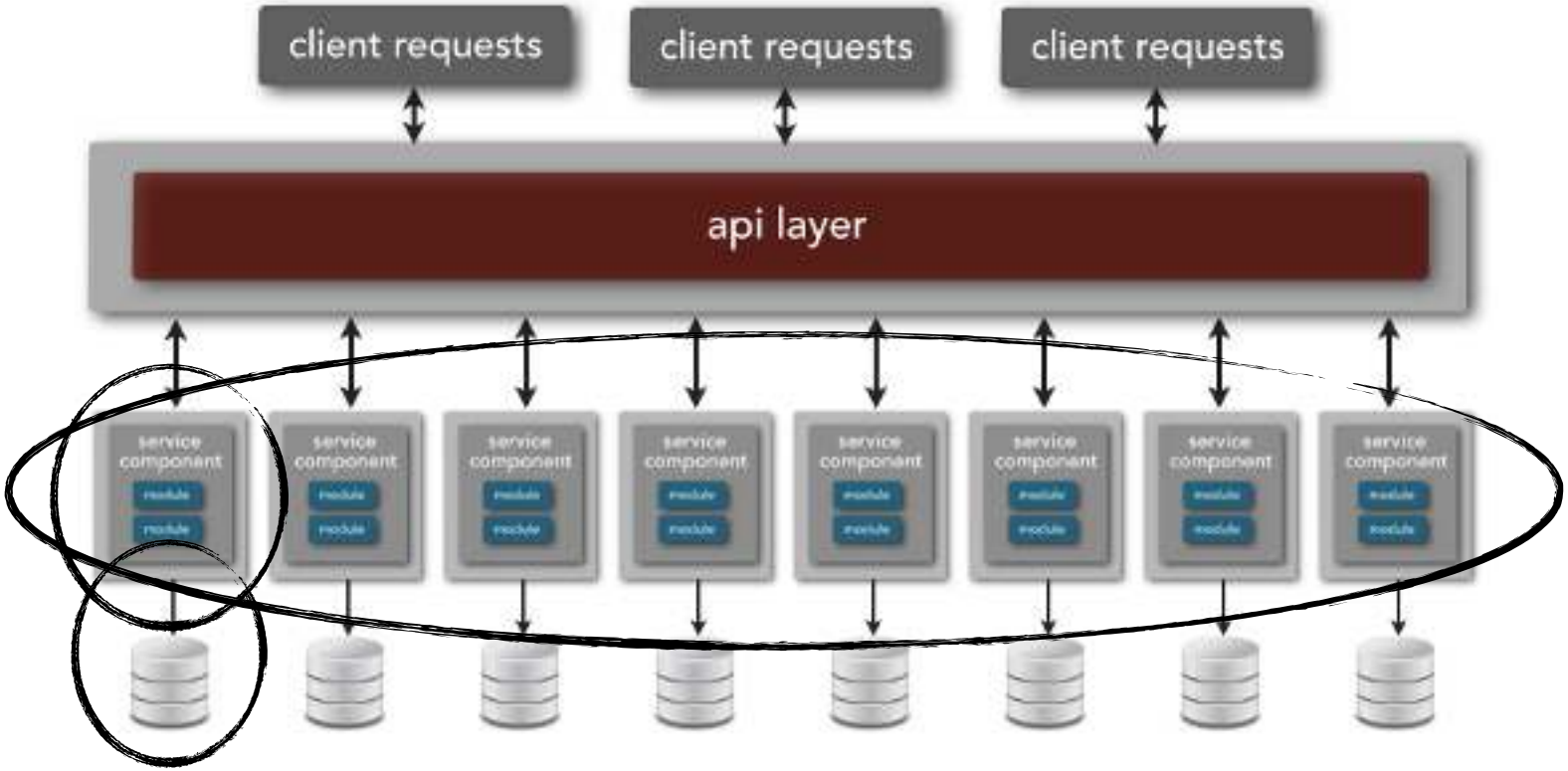
:hitecture



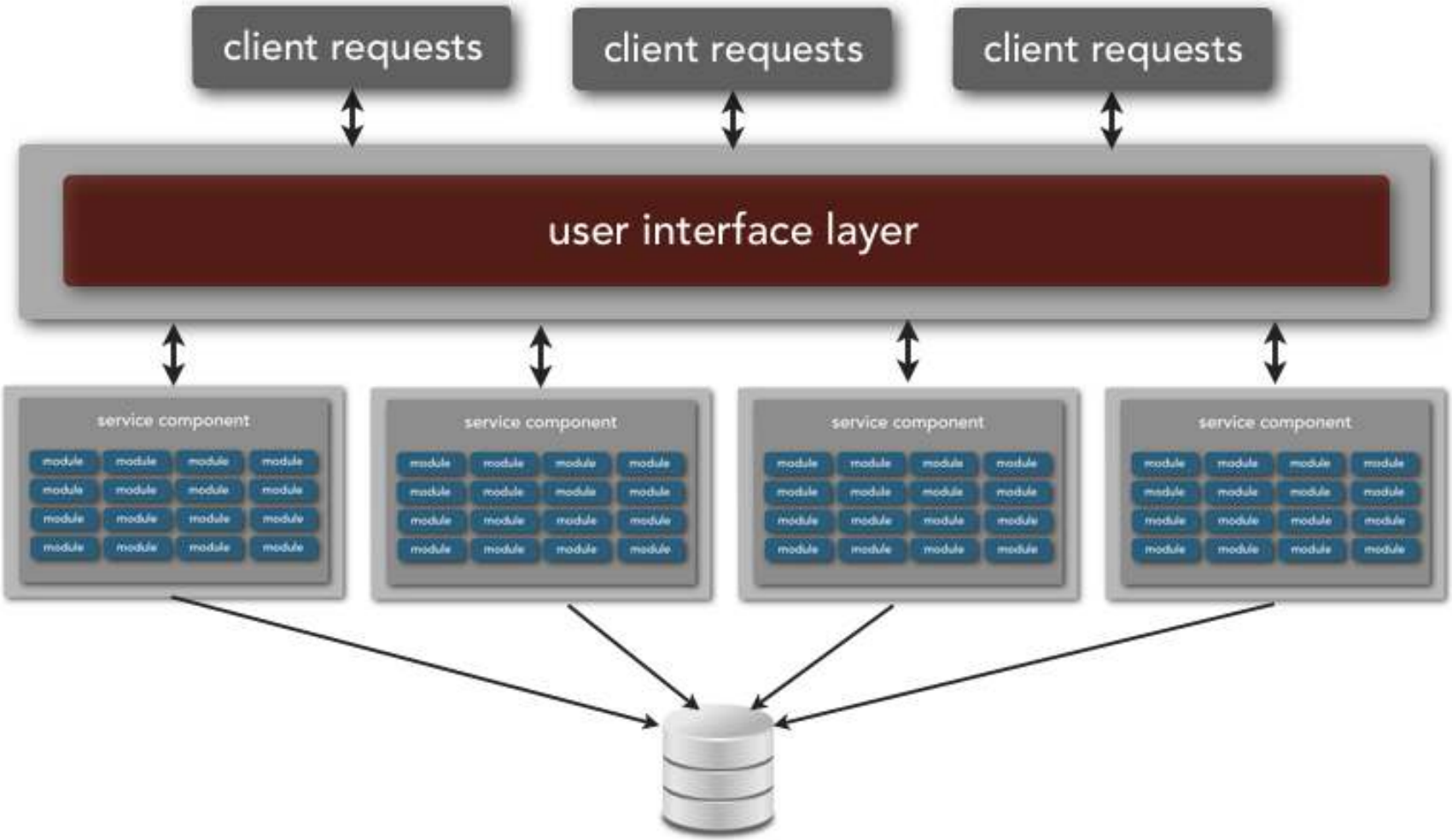
service-based architecture

service-based architecture

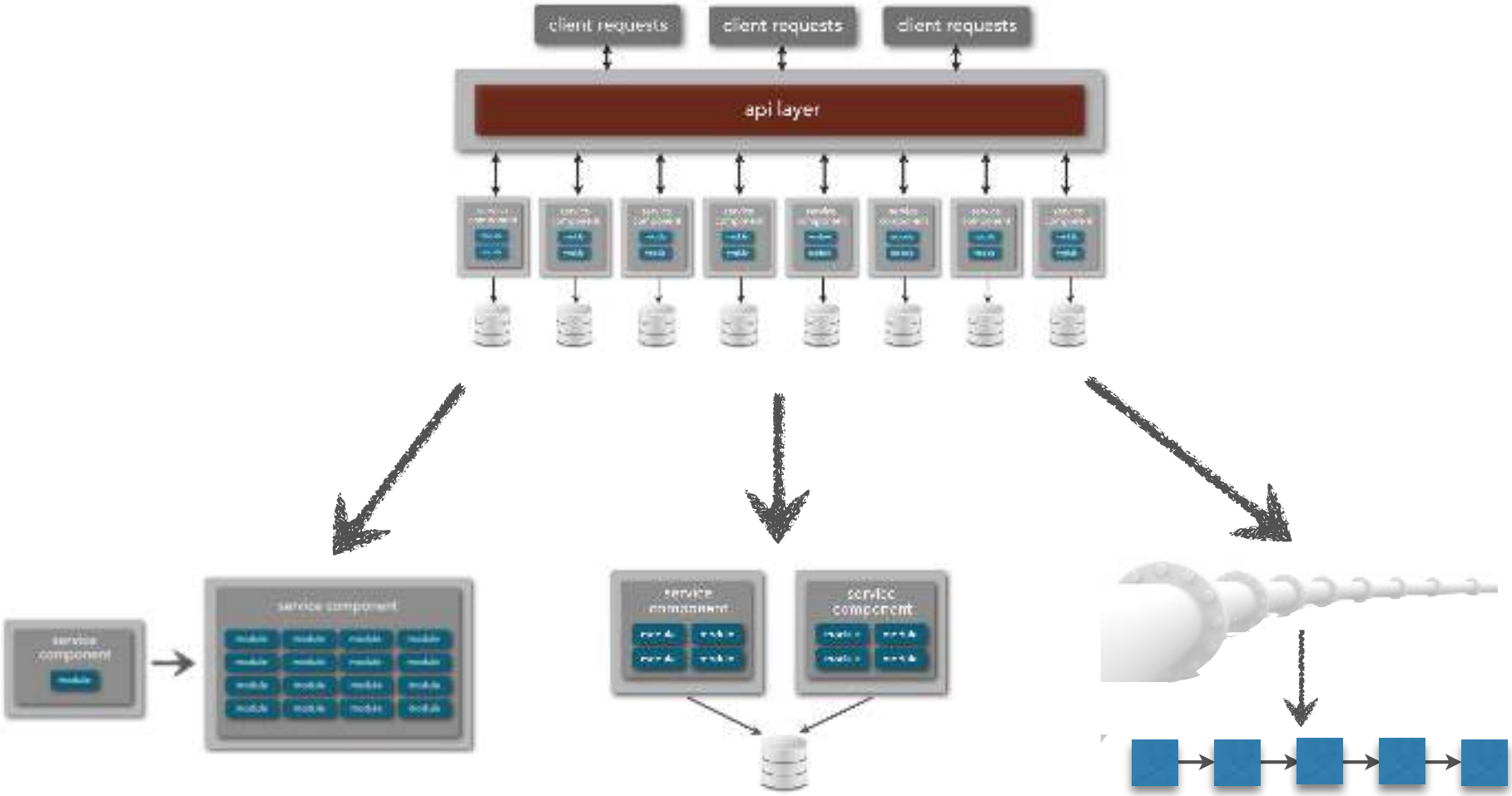
business applications?



service-based architecture



service-based architecture



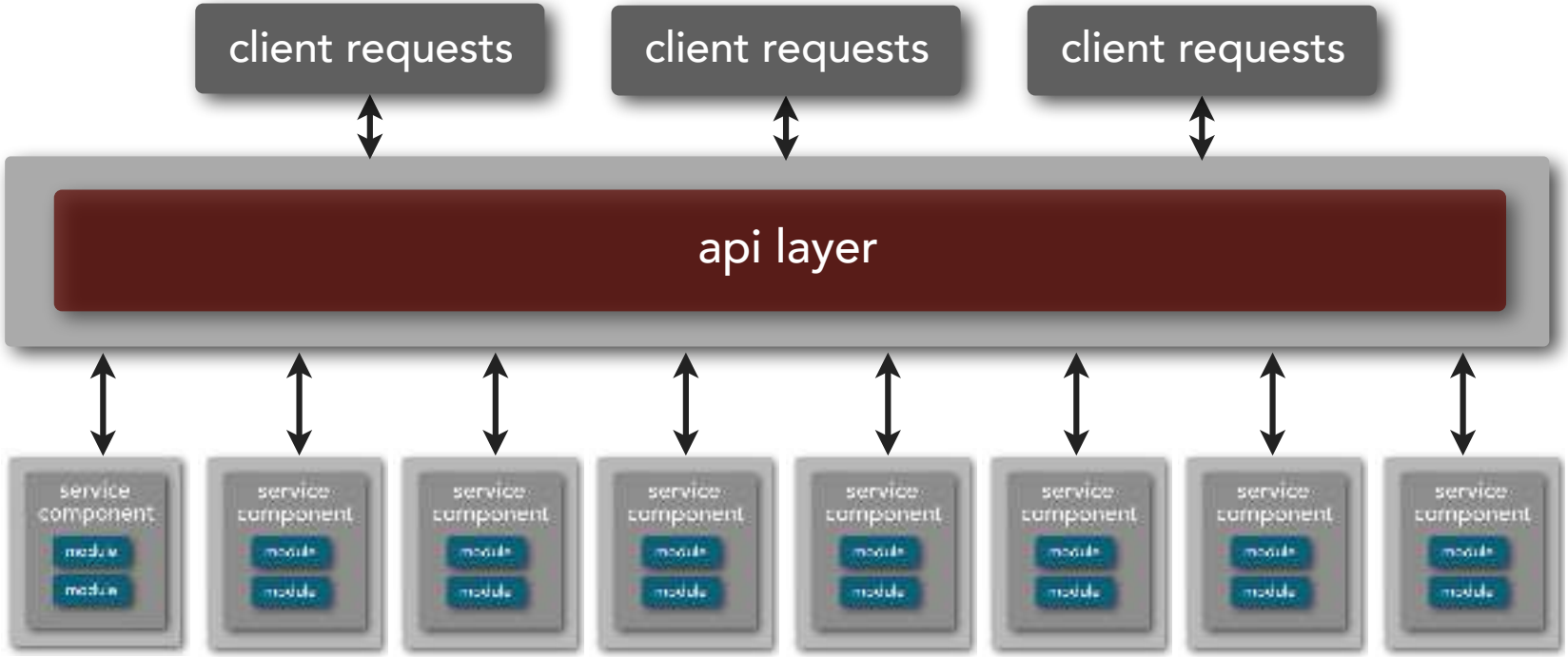
service granularity

database scope

deployment pipeline

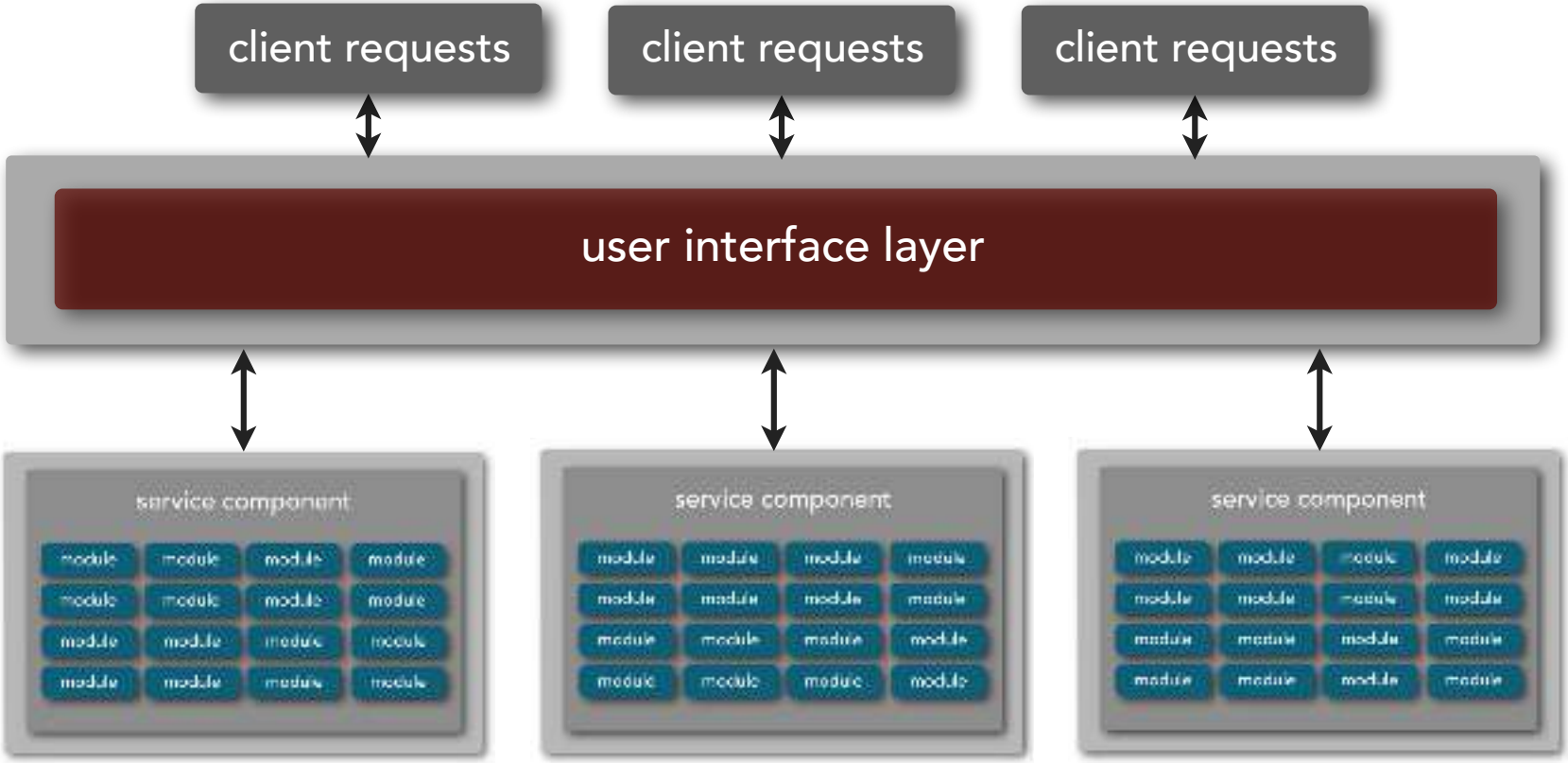
service-based architecture

service granularity



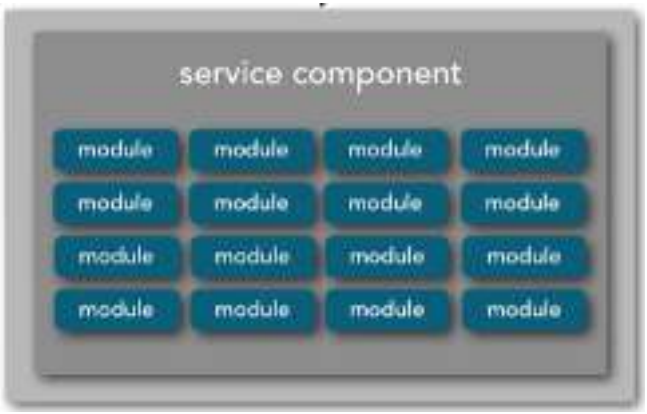
service-based architecture

service granularity



service-based architecture

service granularity

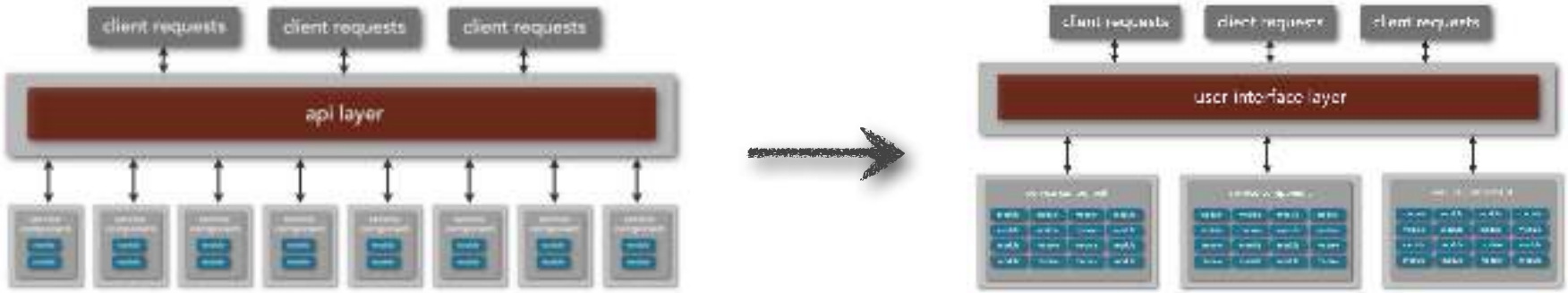


food stamp service
emergency cash service
utility assistance service
child care assist service
health care assist service
nursing facility care service
...

benefit service

service-based architecture

service granularity

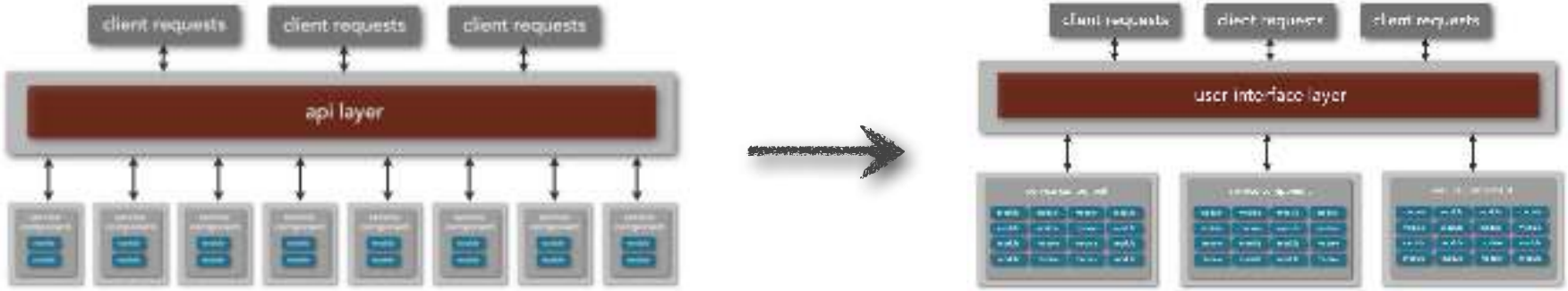


advantages

- unit of work transactional context
- performance and robustness
- domain scope
- shared resources

service-based architecture

service granularity

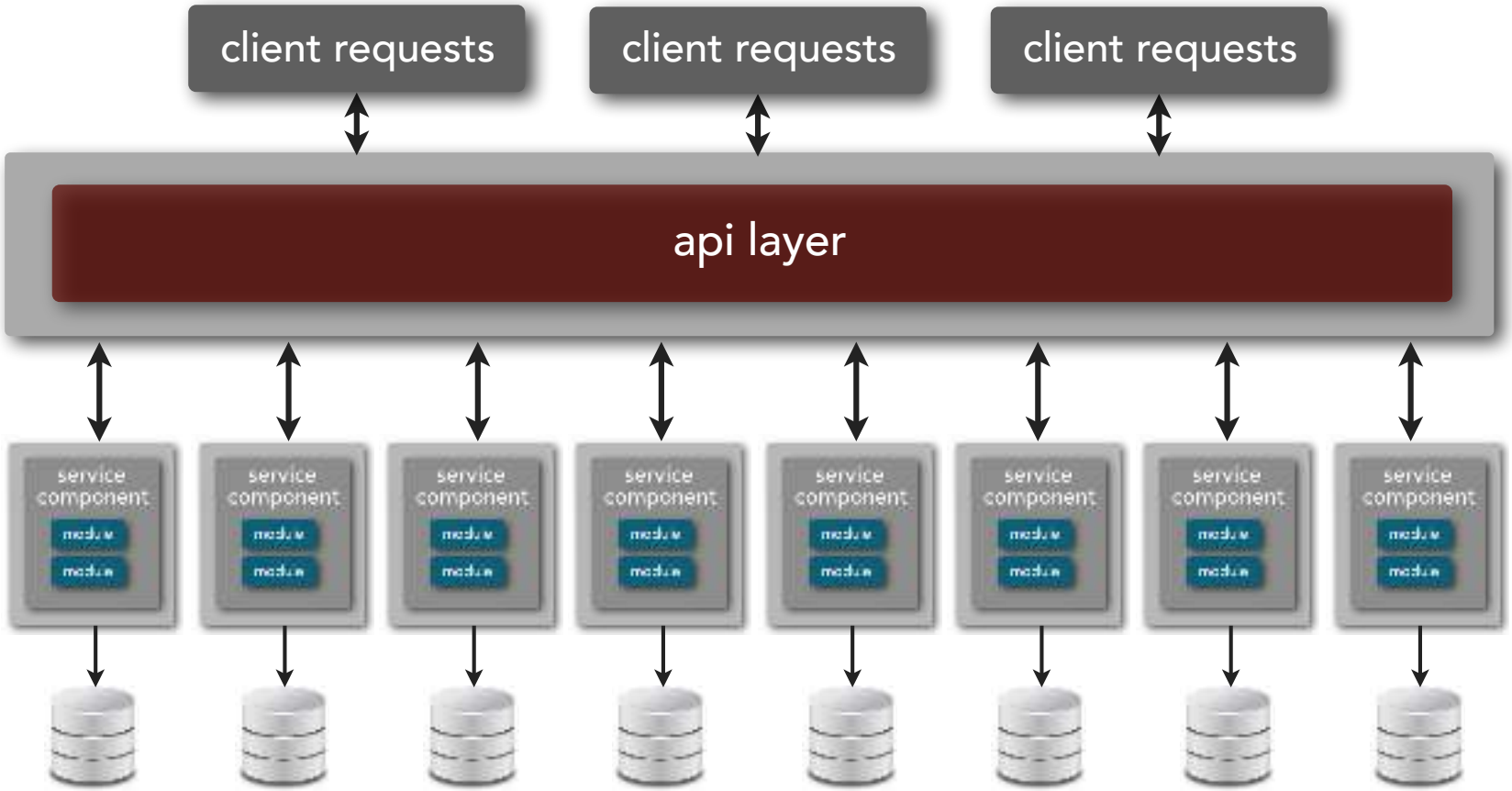


tradeoffs

- services development and testing
- deployment pipeline planning
- change control

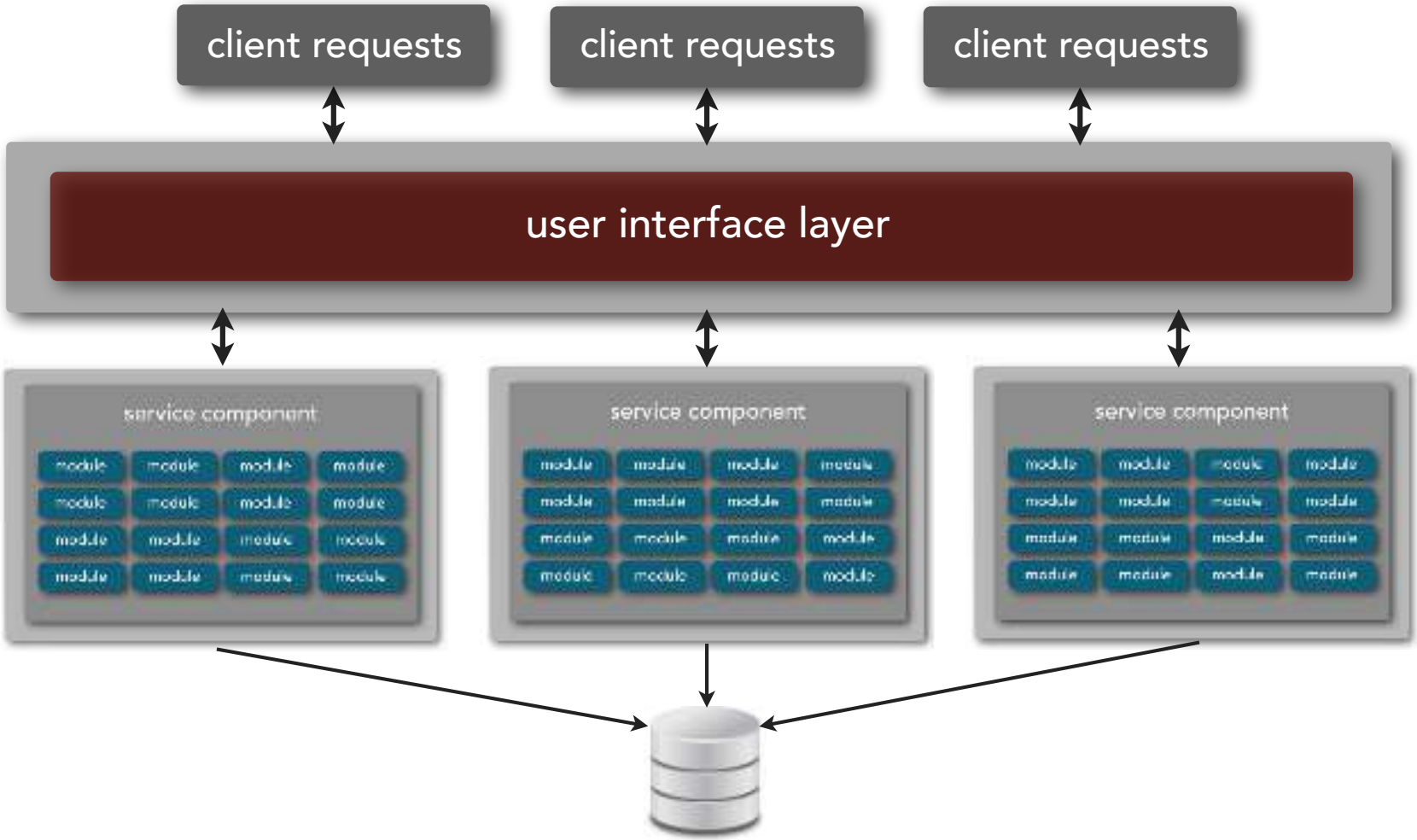
service-based architecture

database scope



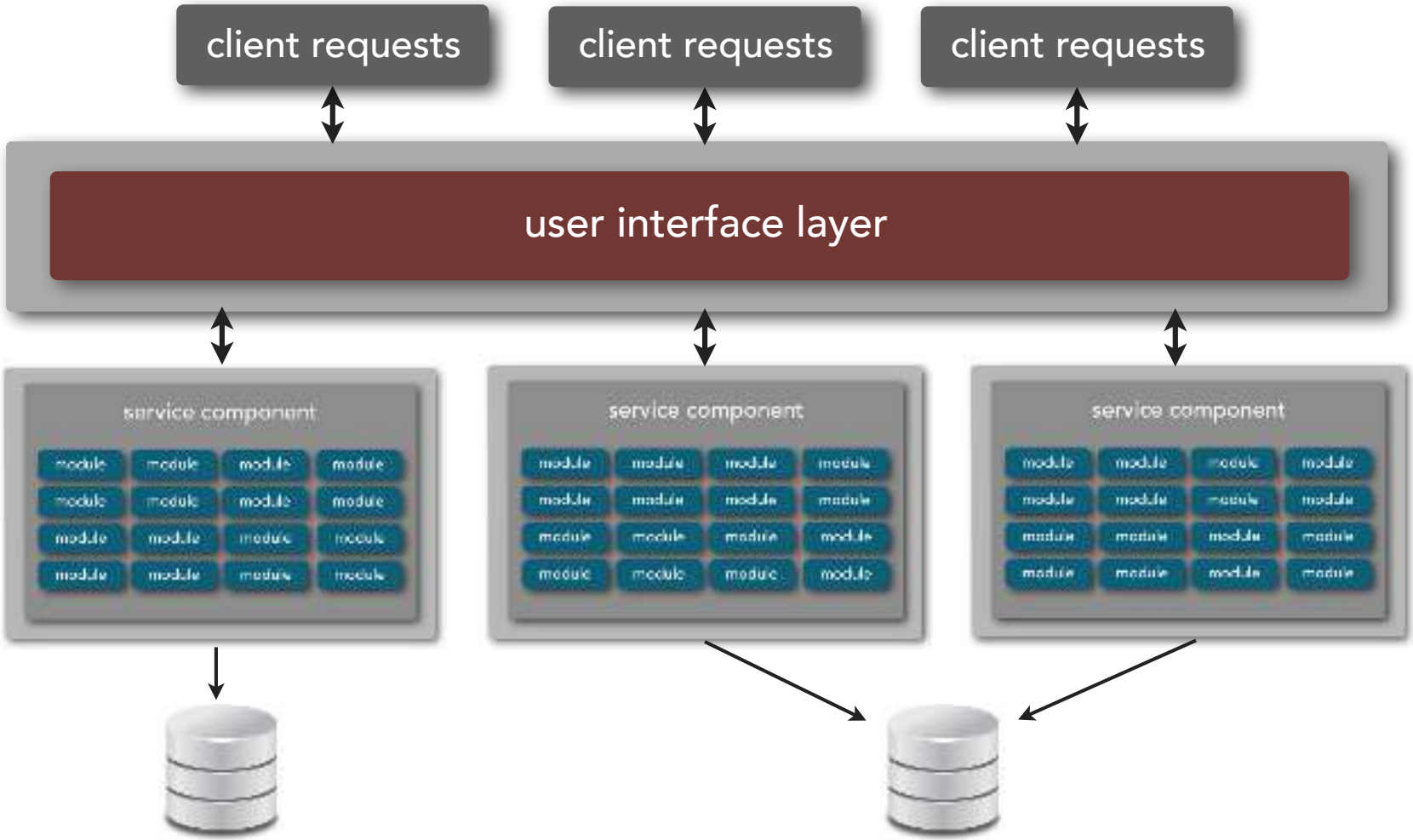
service-based architecture

database scope



service-based architecture

database scope



service-based architecture

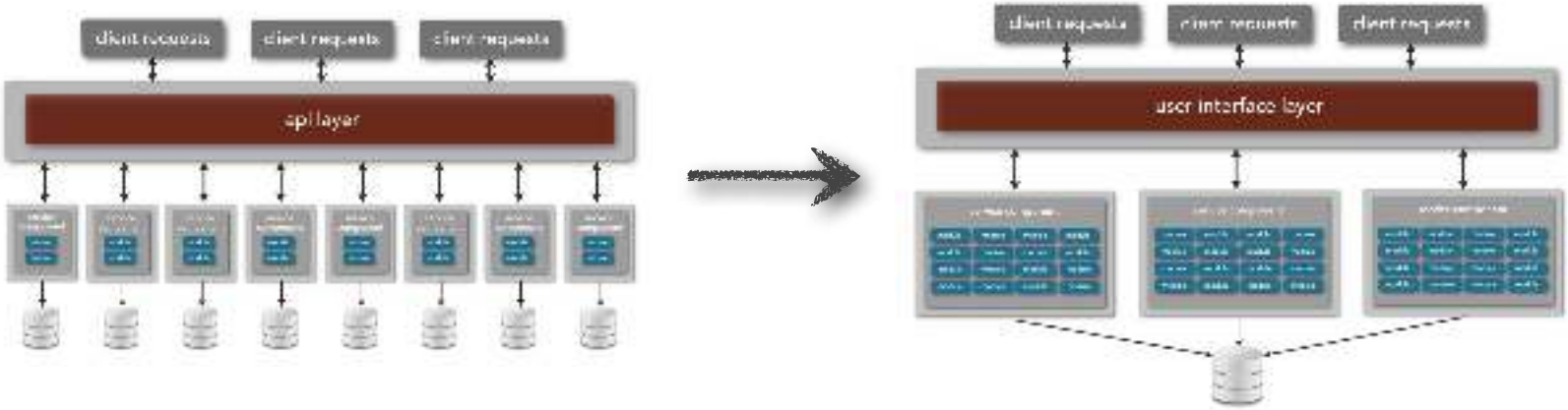
database scope

food stamp db
emergency cash db
utility assistance db
child care assist db
health care assist db
nursing facility care db
...

shared common db

service-based architecture

database scope

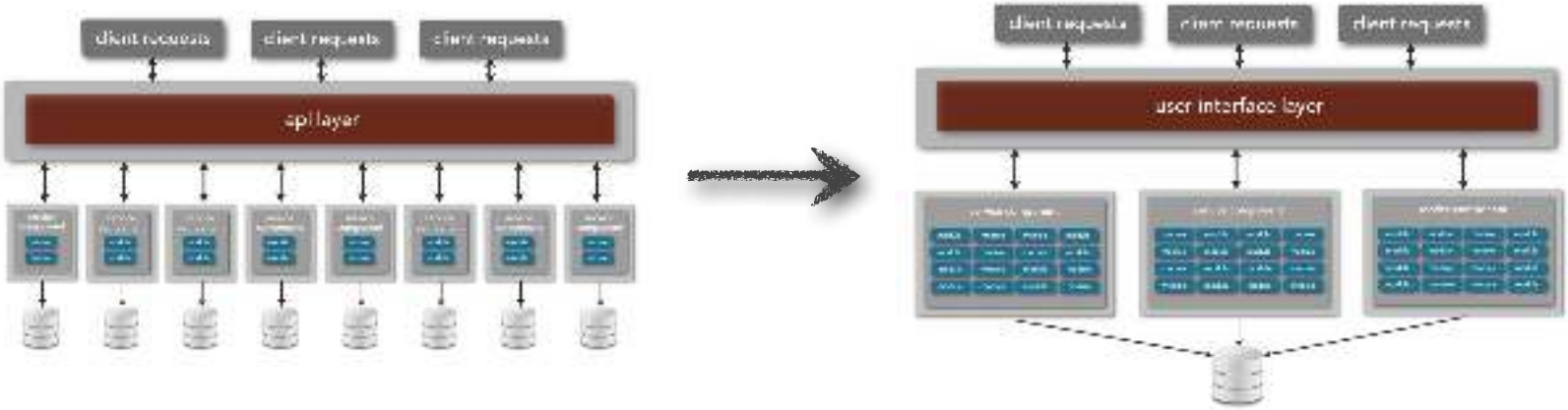


advantages

- performance (joins, orchestration, choreography)
- feasibility

service-based architecture

database scope

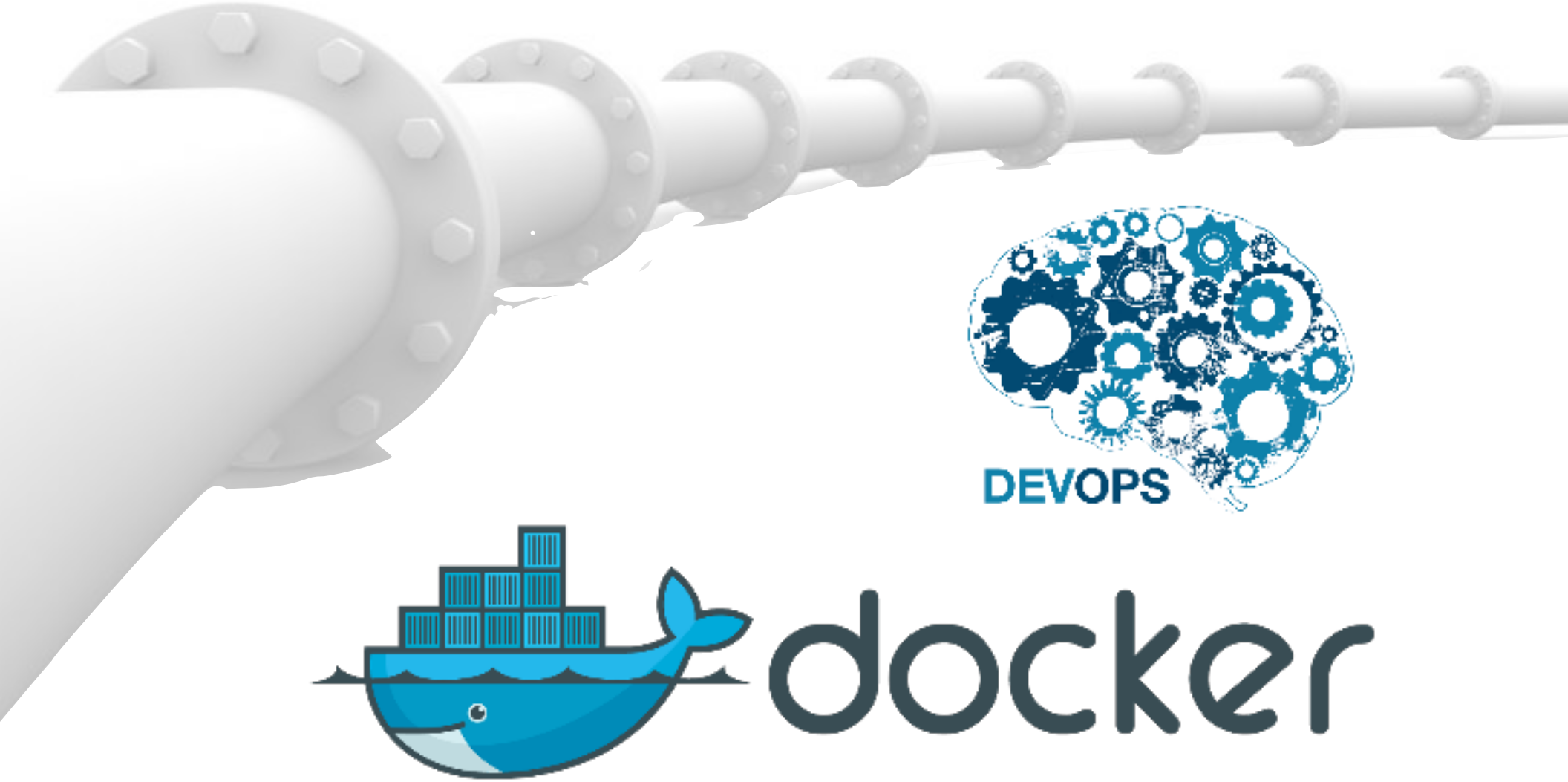


tradeoffs

- bounded context
- service coupling based on schema
- schema changes

service-based architecture

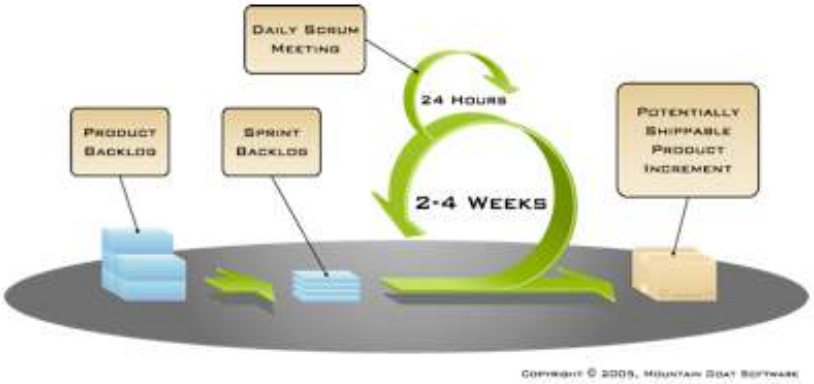
deployment pipeline

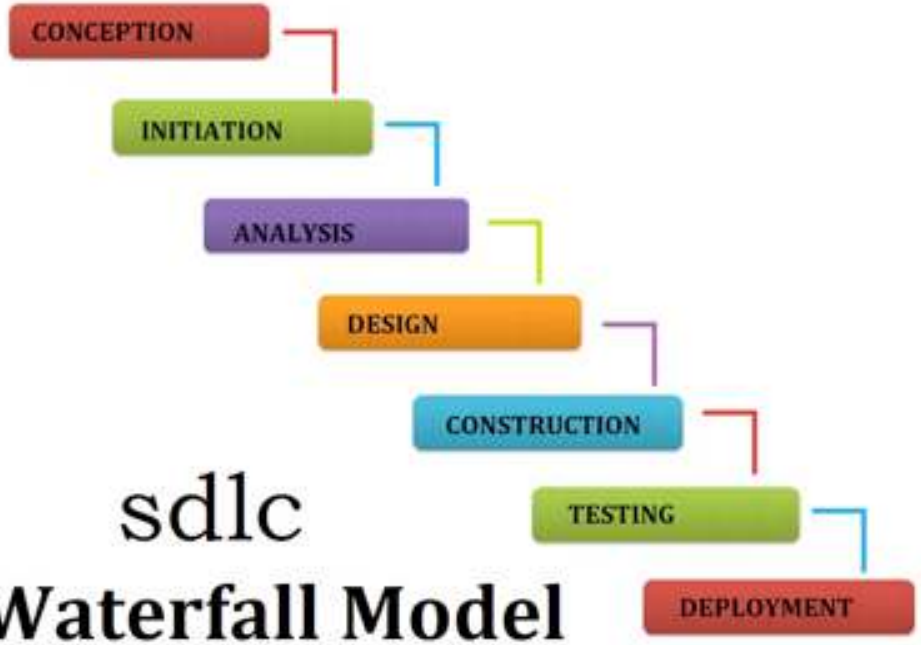


docker

service-based architecture

deployment pipeline



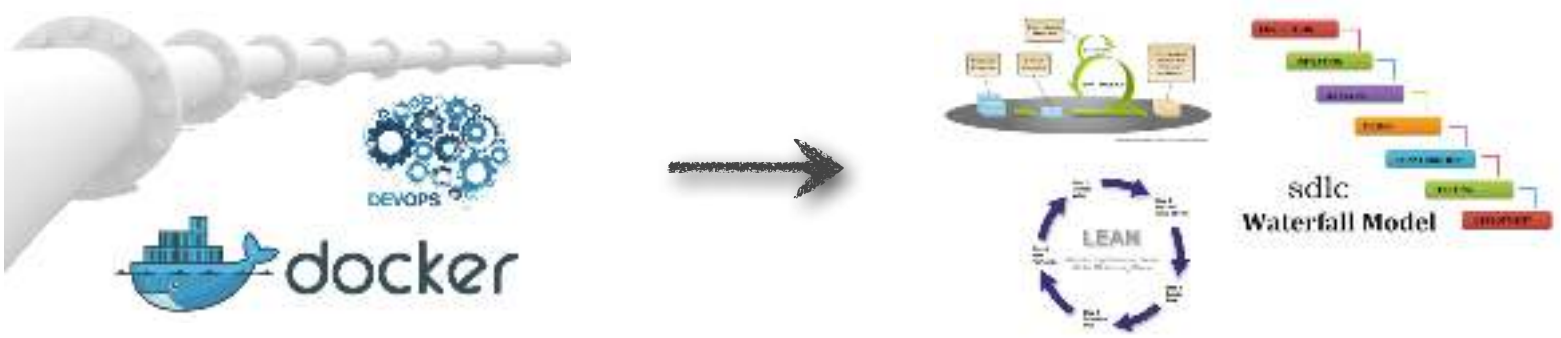


sdlc

Waterfall Model

service-based architecture

deployment pipeline

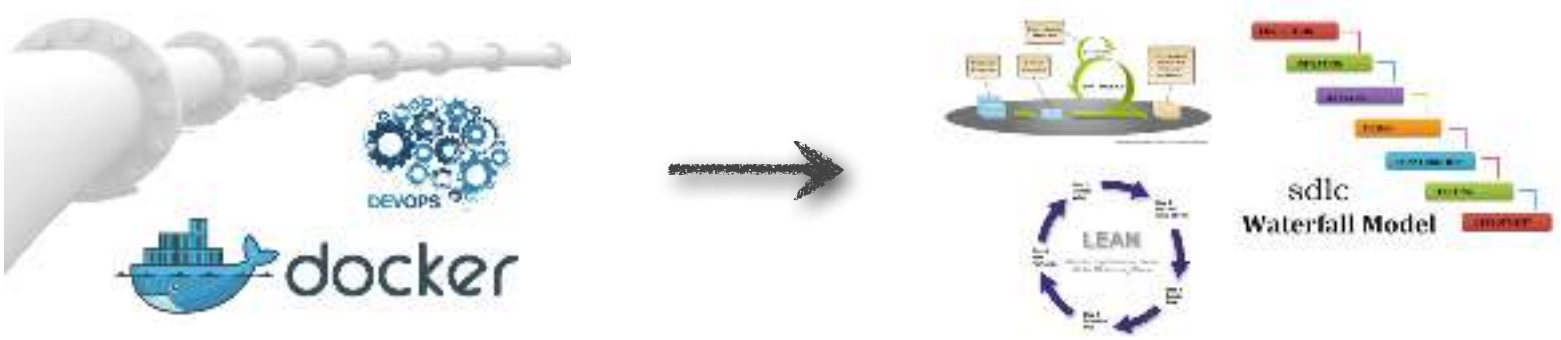


advantages

- no devops complexity
- minimal organizational change

service-based architecture

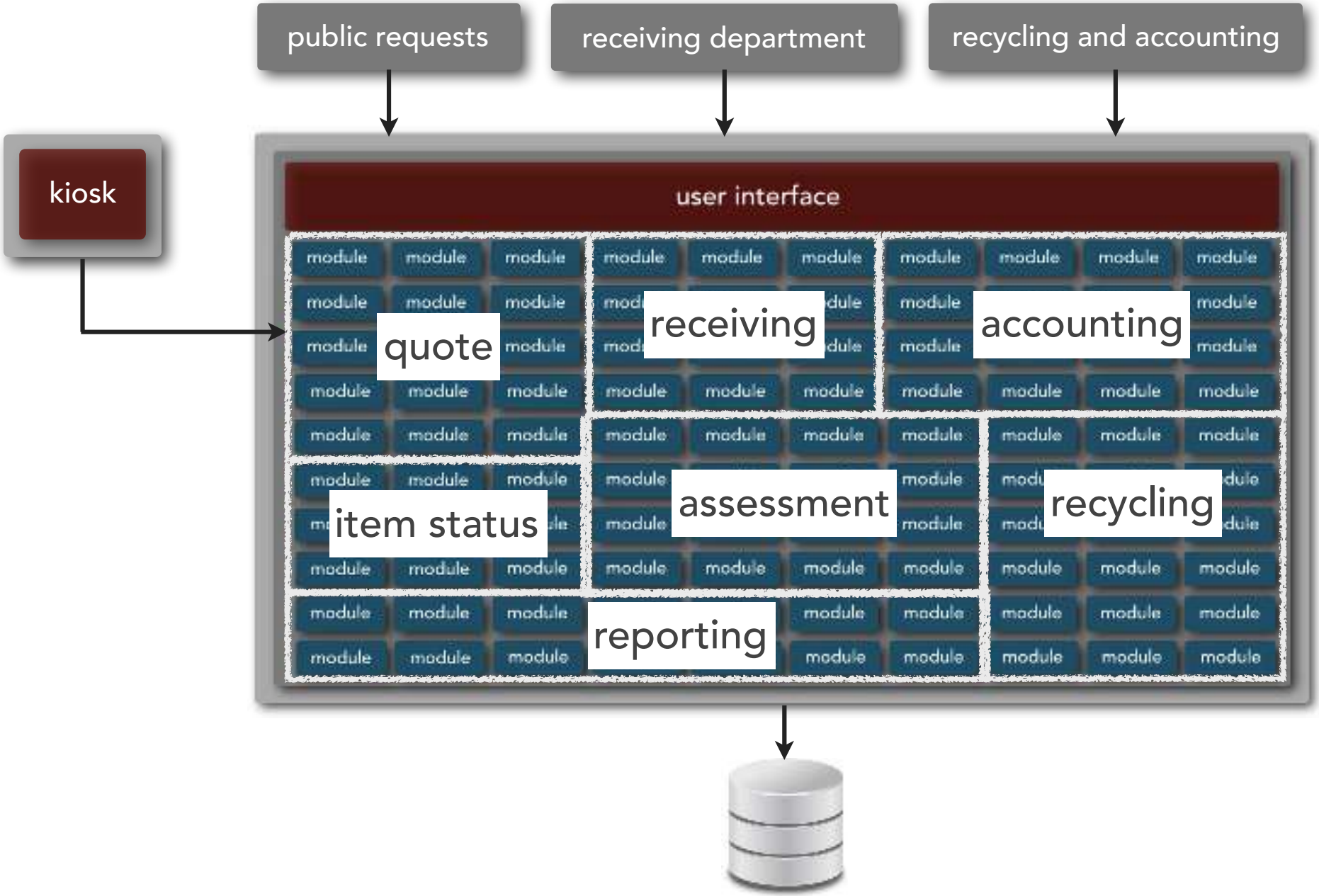
deployment pipeline



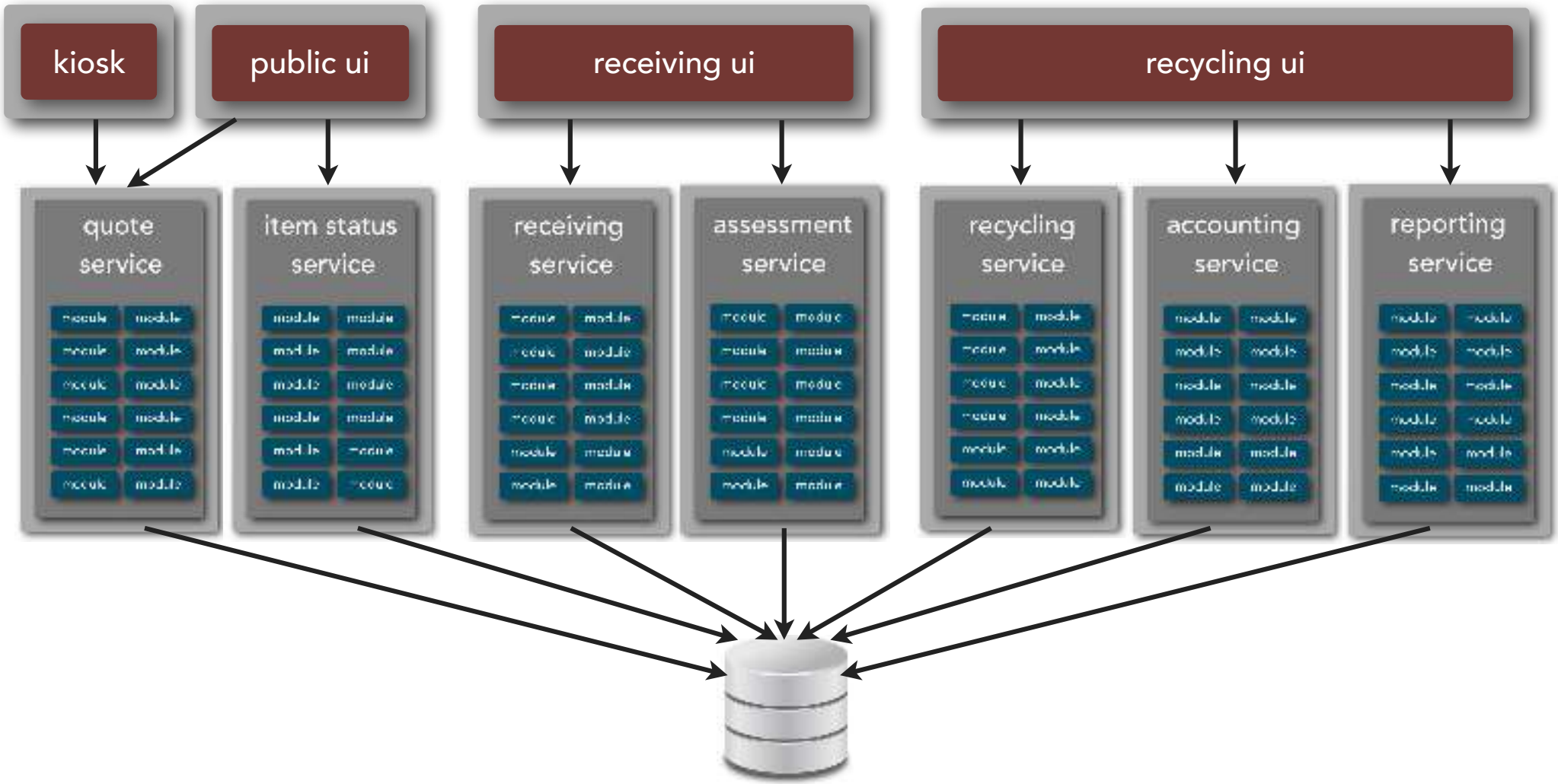
tradeoffs

- lack of quick and effective deployments
- additional risk and coordination needed
- poor continuous delivery model

electronics recycling application

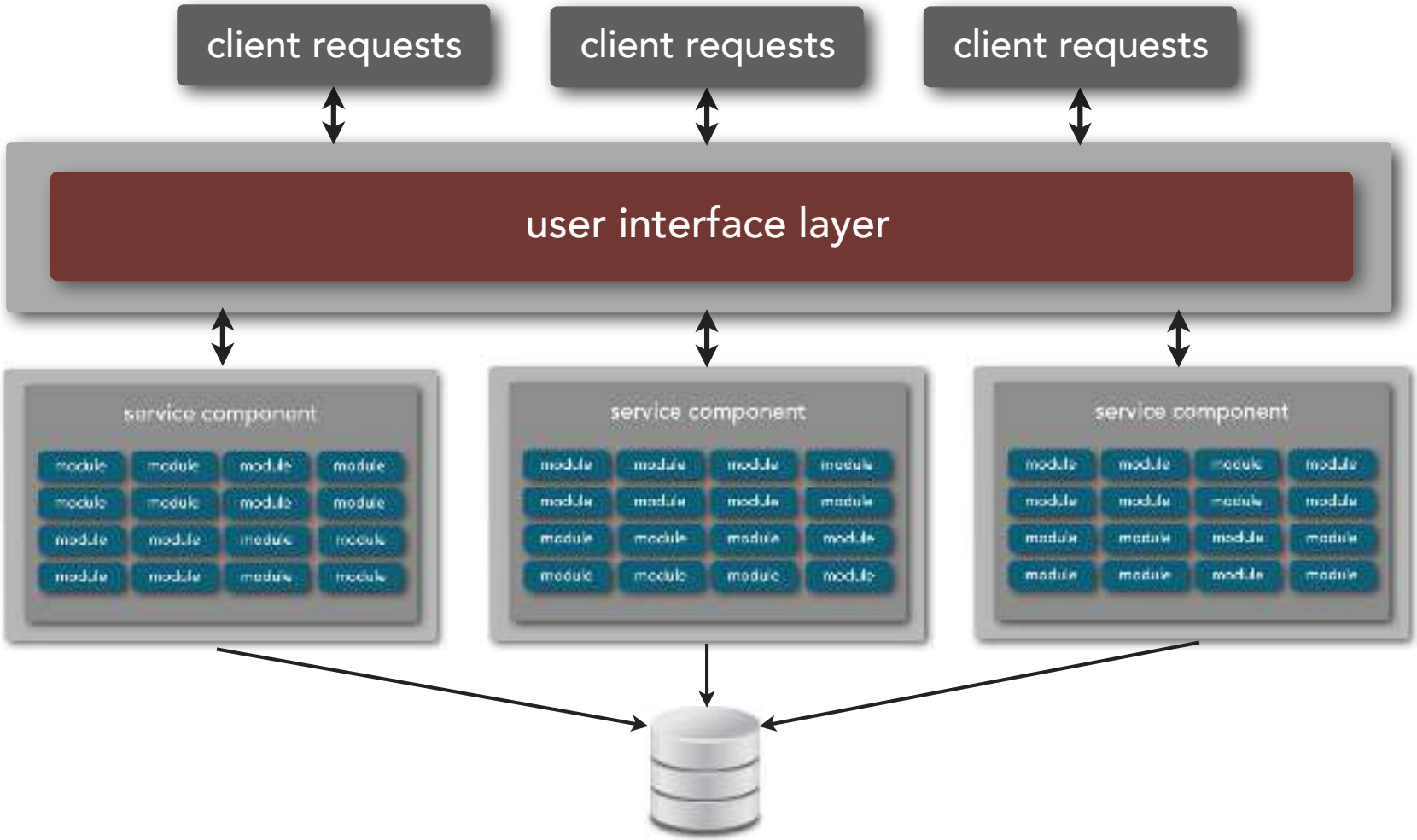


electronics recycling application



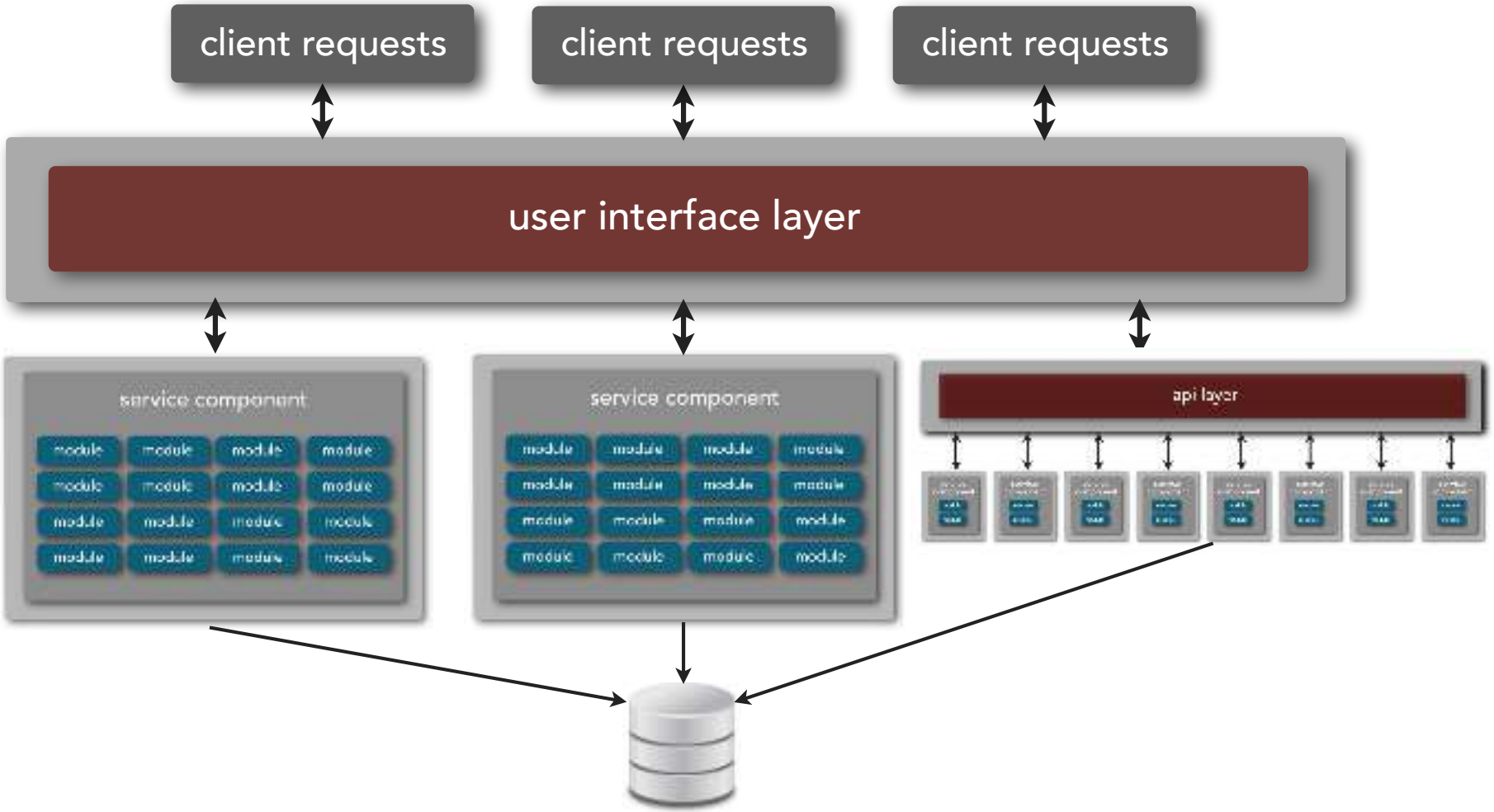
service-based architecture

adding microservices



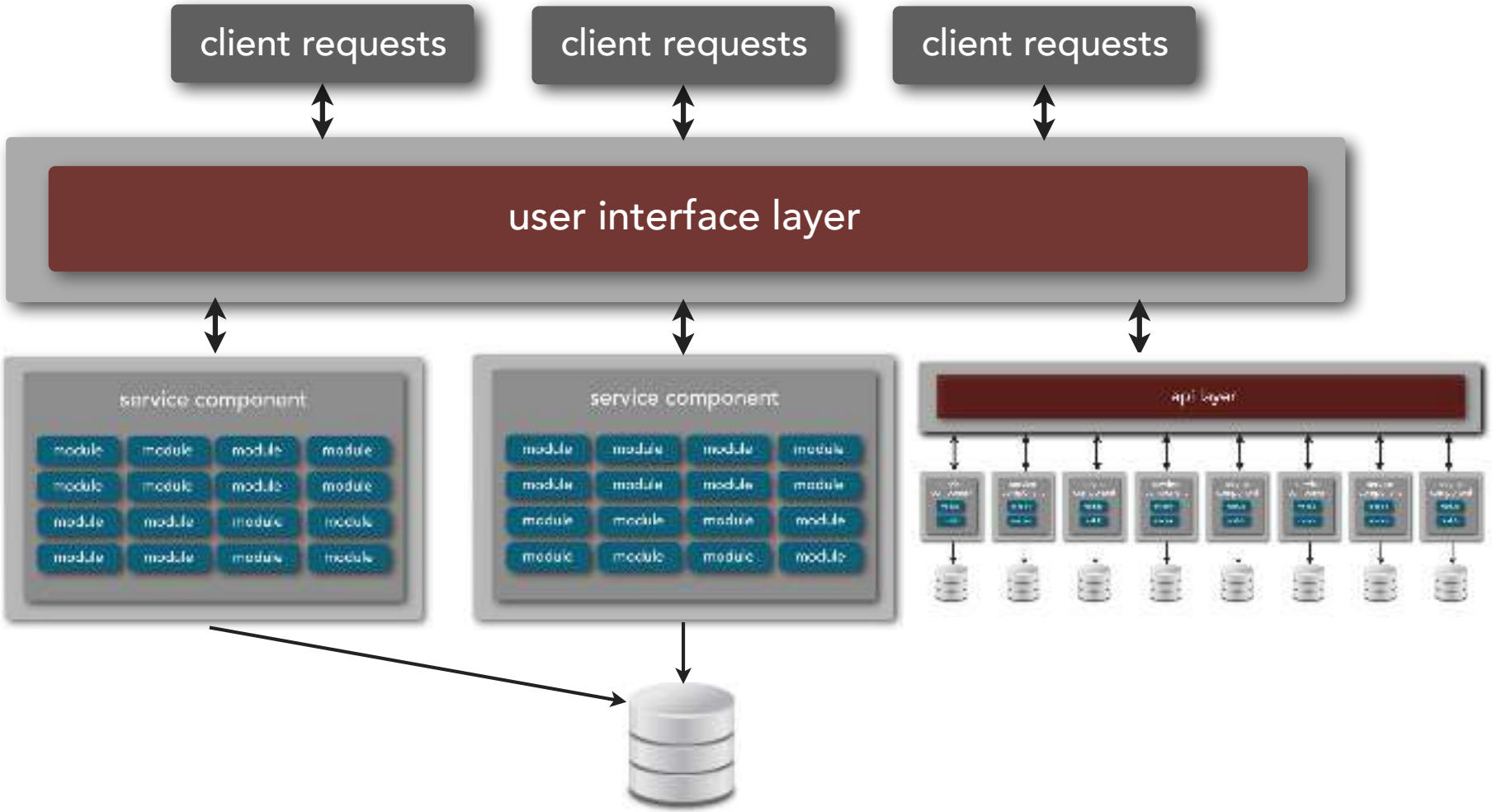
service-based architecture

adding microservices

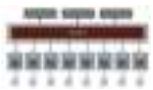


service-based architecture

adding microservices



service-based architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👎	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$


LMAX

LMAX

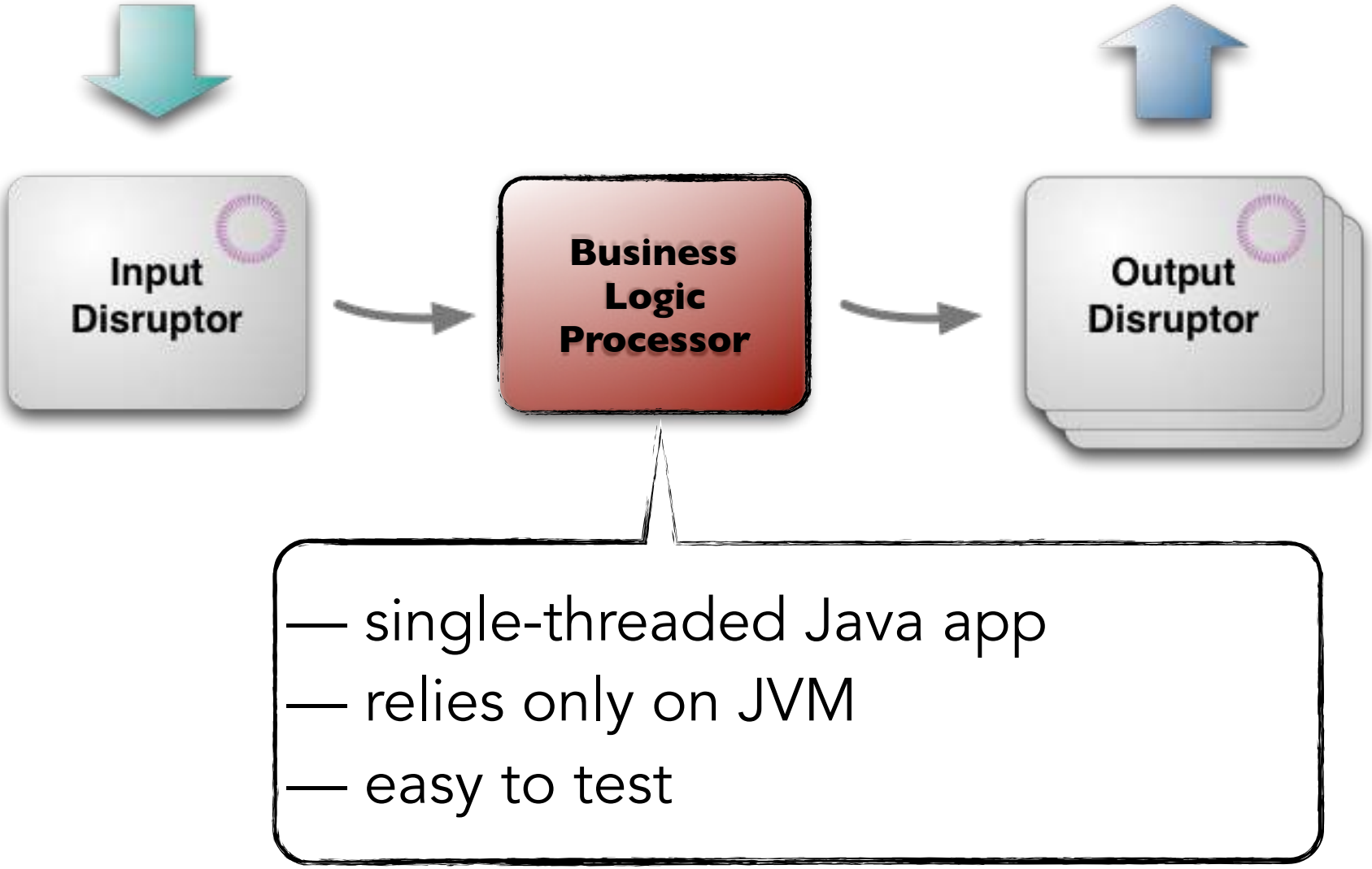
http://martinfowler.com/articles/lmax.html

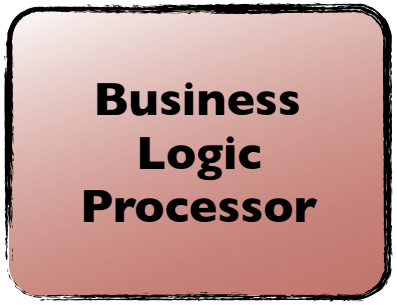
JVM-based retail financial trading platform

centers on Business Logic Processor handling 6,000,000 orders/sec on 1 thread

surrounded by Disruptors, network of lock-less queues

overall structure





business logic processor

in-memory

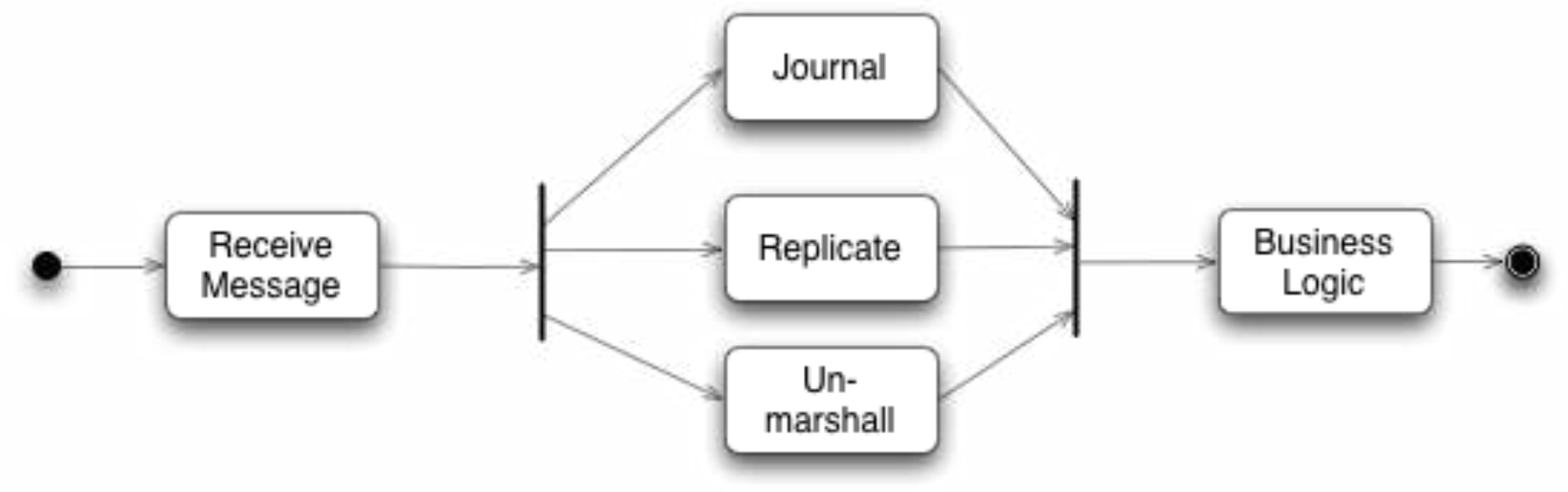
event sourcing via input disruptor

snapshots (full restart—JVM + snapshots — less than 1 min)

multiple instances running

each event processed by multiple processors but only one result used

input/output disruptors



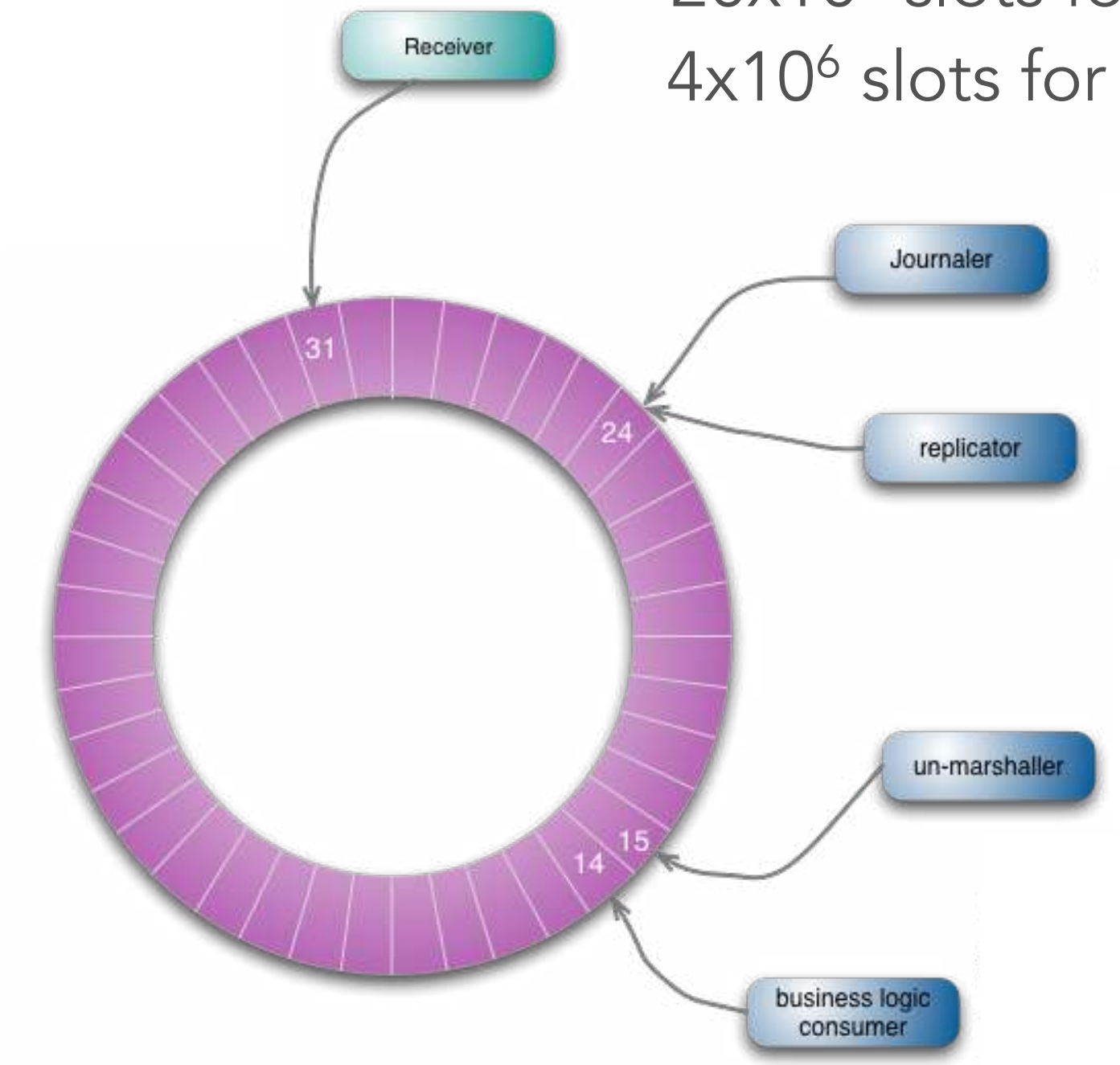
disruptors

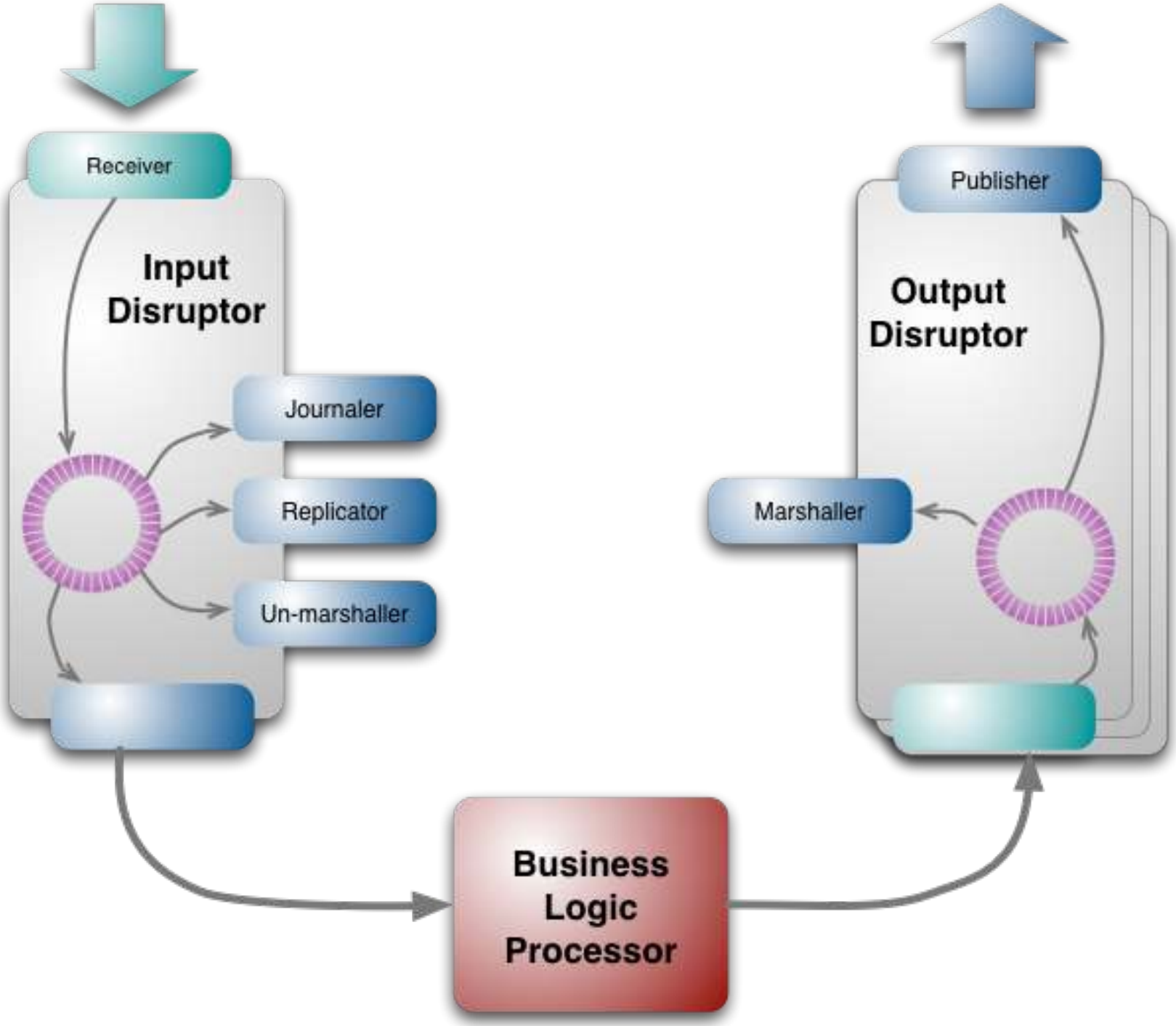
custom concurrency component

multi-cast graph of queues where producers enqueue objects and consumers dequeue in parallel

ring buffer with sequence counters

20x10^6 slots for input buffer
4x10^6 slots for output buffer




Receiver
Input
Disruptor
Journaler
Replicator
Un-marshaller
Business
Logic
Processor
Publisher
Output
Disruptor
Marshaller

“mechanical sympathy”

started with transactions

switched to Actor-based concurrency

hypothesized & measured results

CPU caching is key ➠ single writer principle

architecture katas

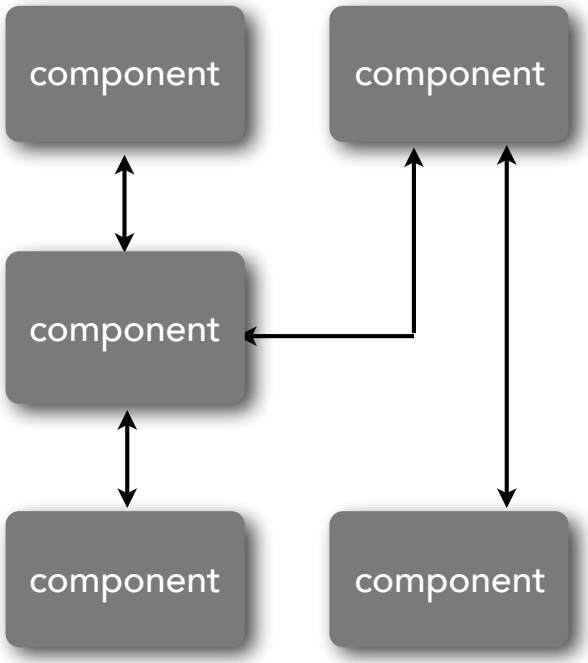
identifying architecture patterns

component-based thinking

component identification and granularity

component identification

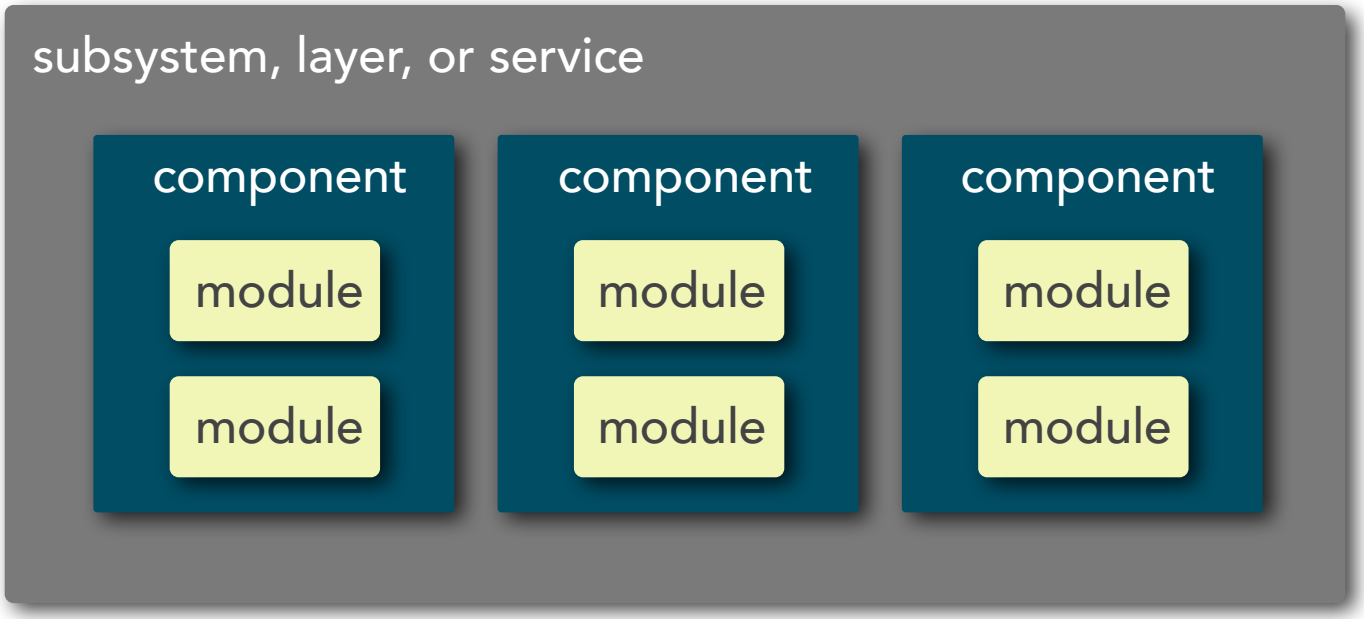
as an architect, you should think about the artifacts within the architecture in terms of *components*



component:
an encapsulated unit of software that has a **well defined interface** and a clear and concise **role and responsibility statement**

component identification

component scope

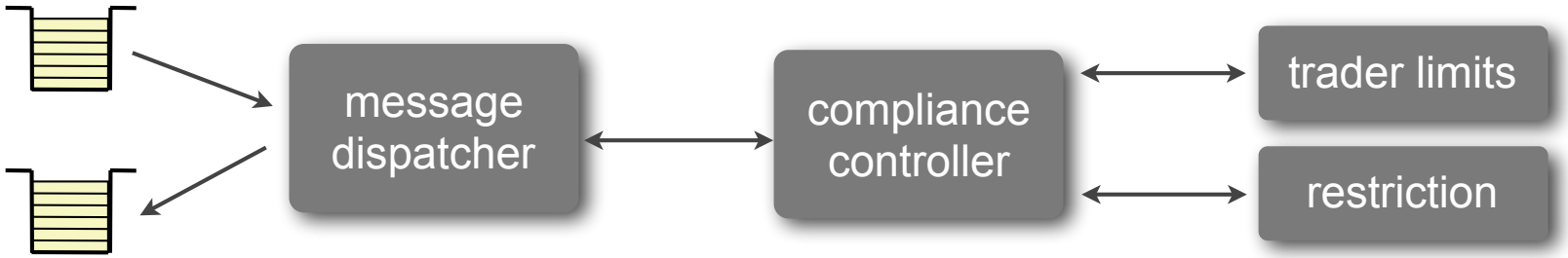


component identification

roles and responsibility model

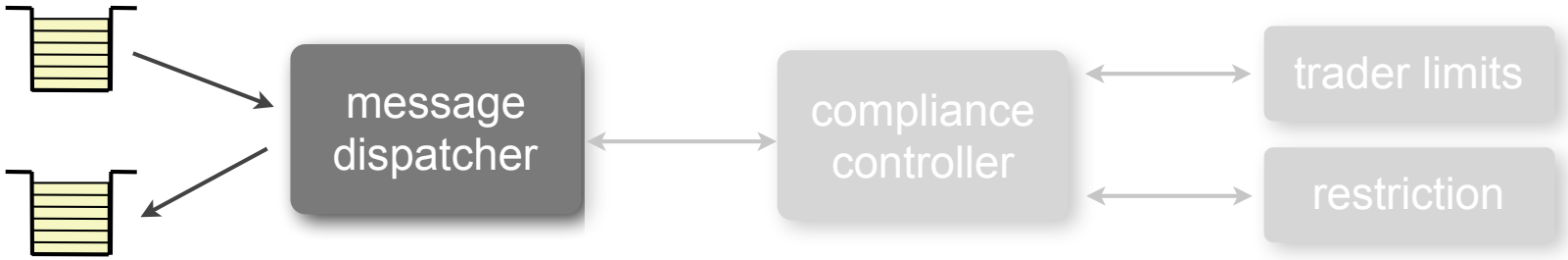
component identification

stock trade order validation



component identification

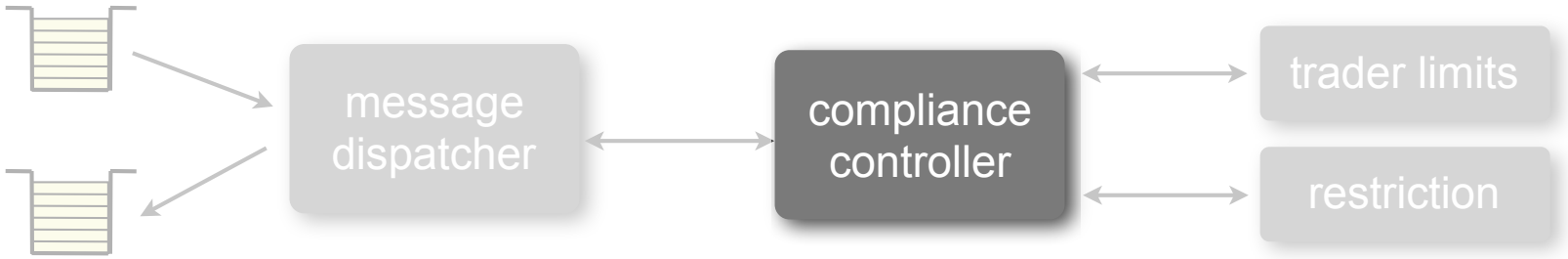
stock trade order validation



responsible for dispatching the trade to the next available controller.

component identification

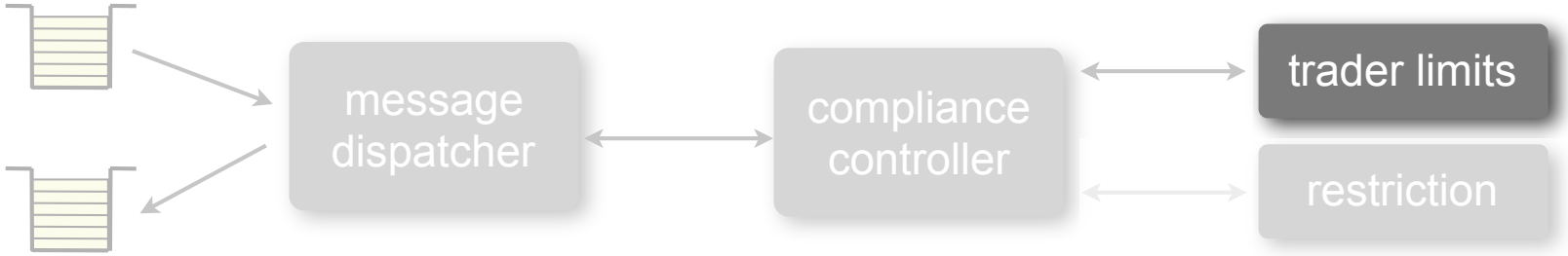
stock trade order validation



responsible for orchestrating the trade order validation process by calling specific compliance processors.

component identification

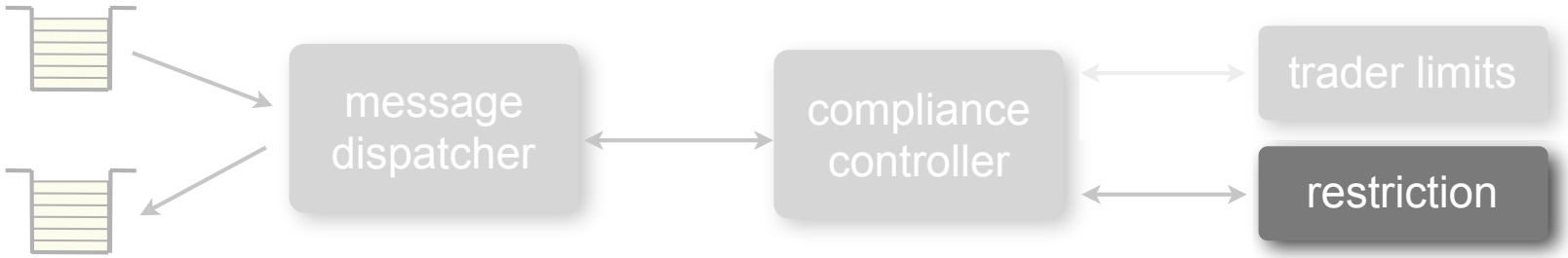
stock trade order validation



responsible for making sure the trader isn't exceeding assigned trader limits for the trade being placed.

component identification

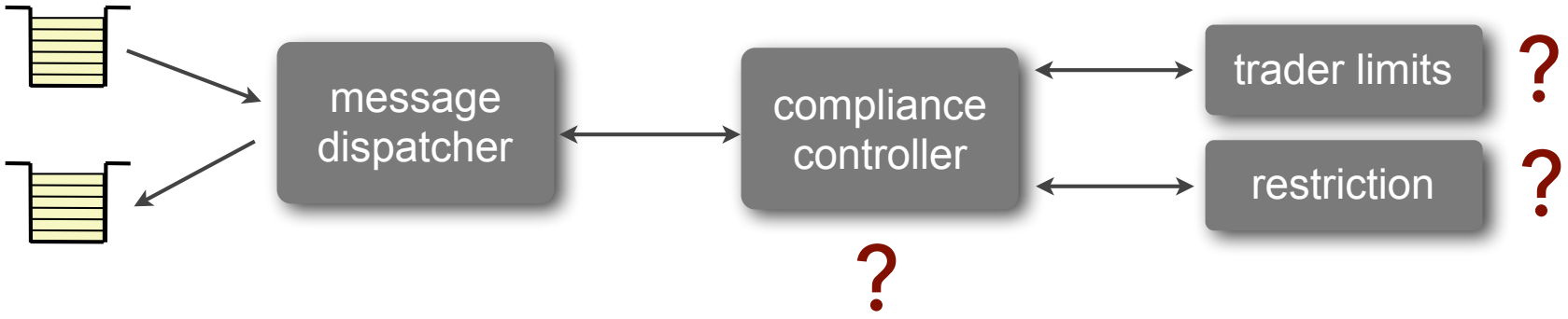
stock trade order validation



responsible for making sure the trade order symbol isn't on the restricted stock list.

component identification

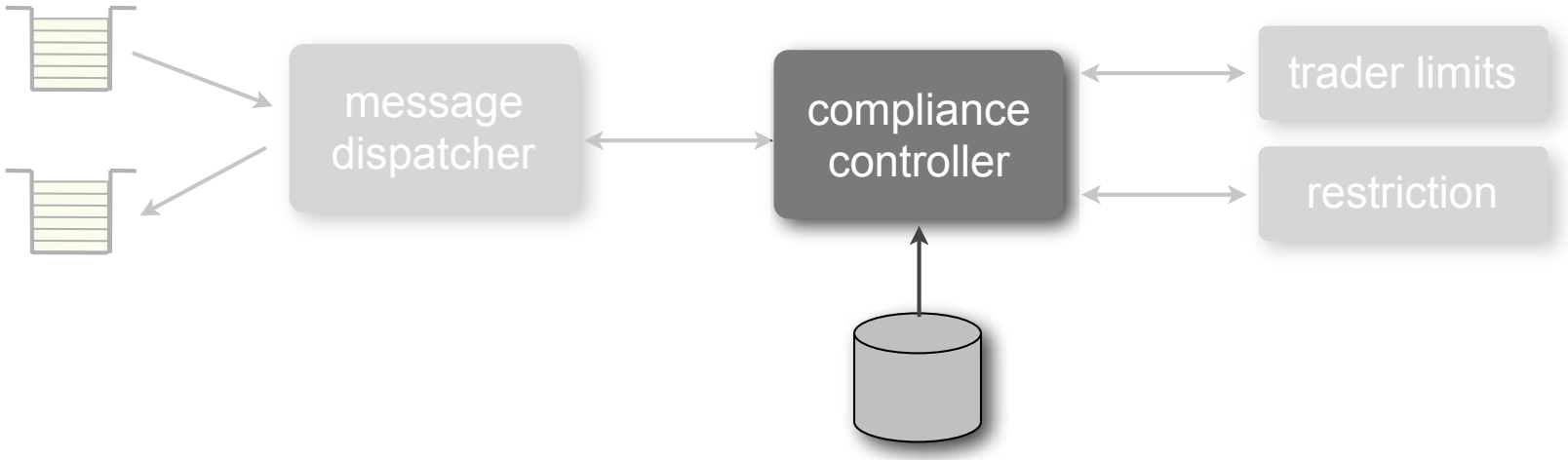
stock trade order validation



who should be responsible for retrieving and caching all of the data needed by the compliance processors?

component identification

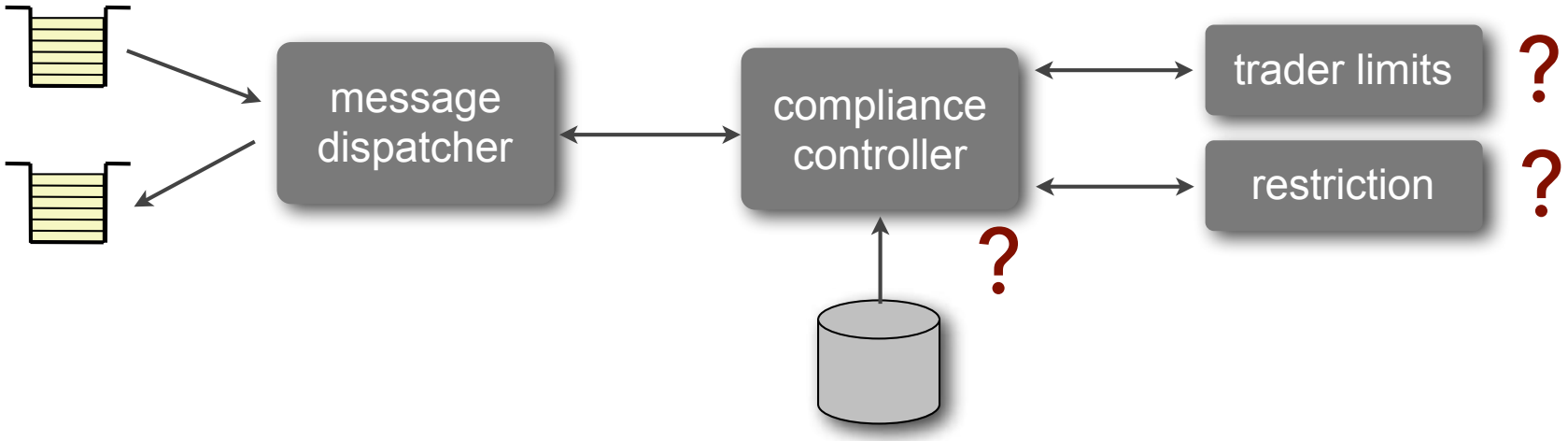
stock trade order validation



responsible for orchestrating the trade order validation process by calling specific compliance processors. also responsible for retrieving and caching all data needed by the compliance processors

component identification

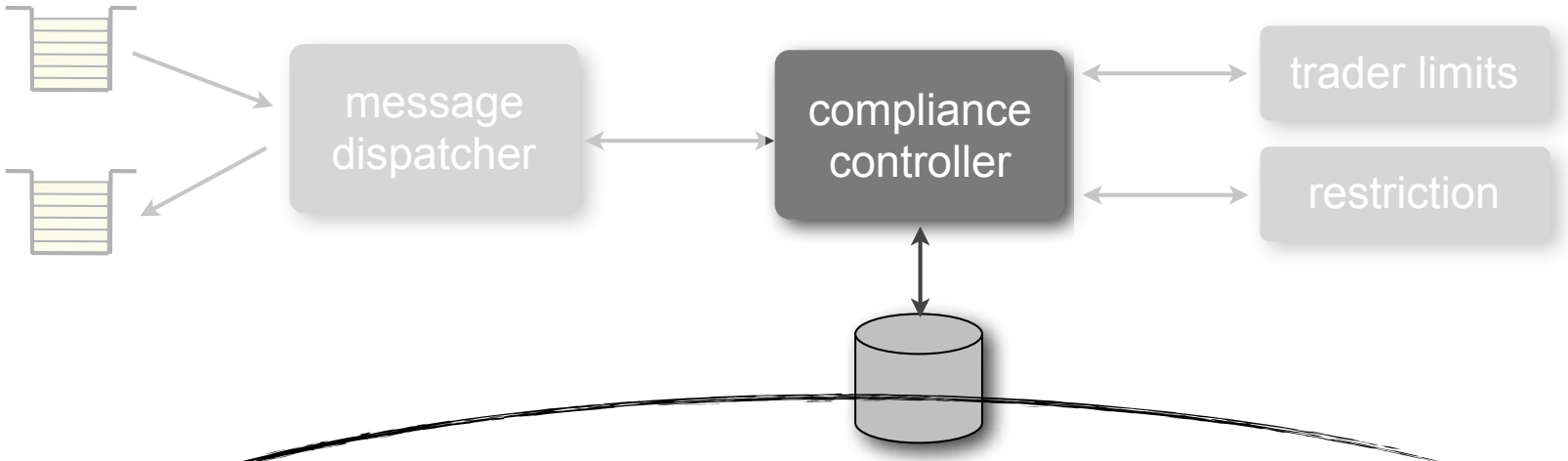
stock trade order validation



who should be responsible for persisting trade validation errors when they occur?

component identification

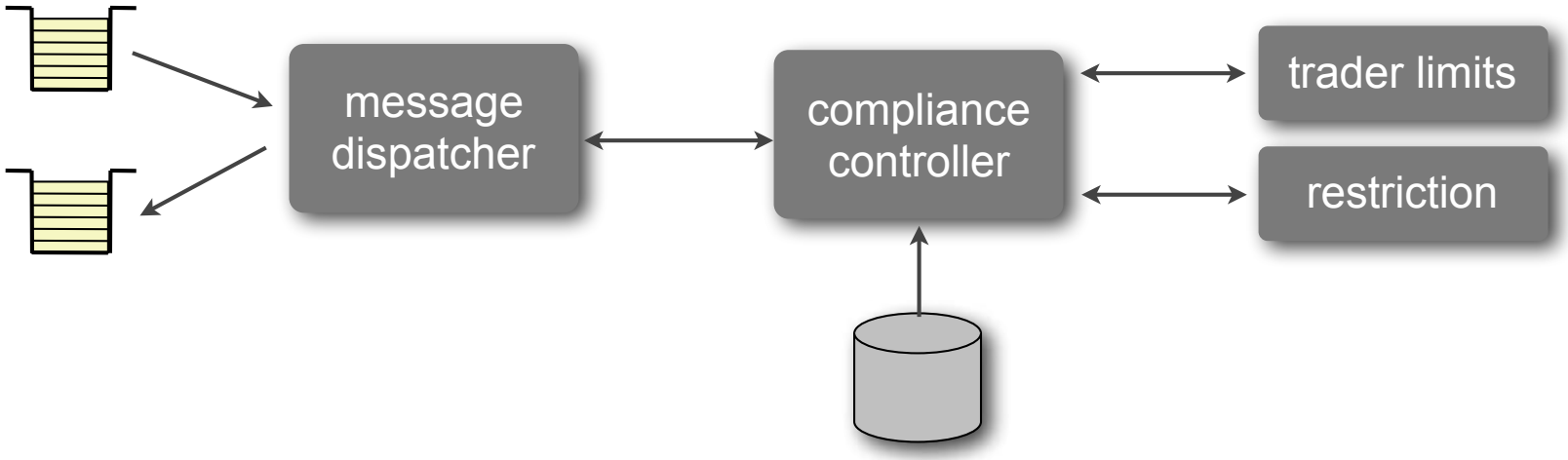
stock trade order validation



responsible for orchestrating the trade order validation process by calling specific compliance processors. also responsible for retrieving and caching all data needed by the compliance processors and persisting all validation errors.

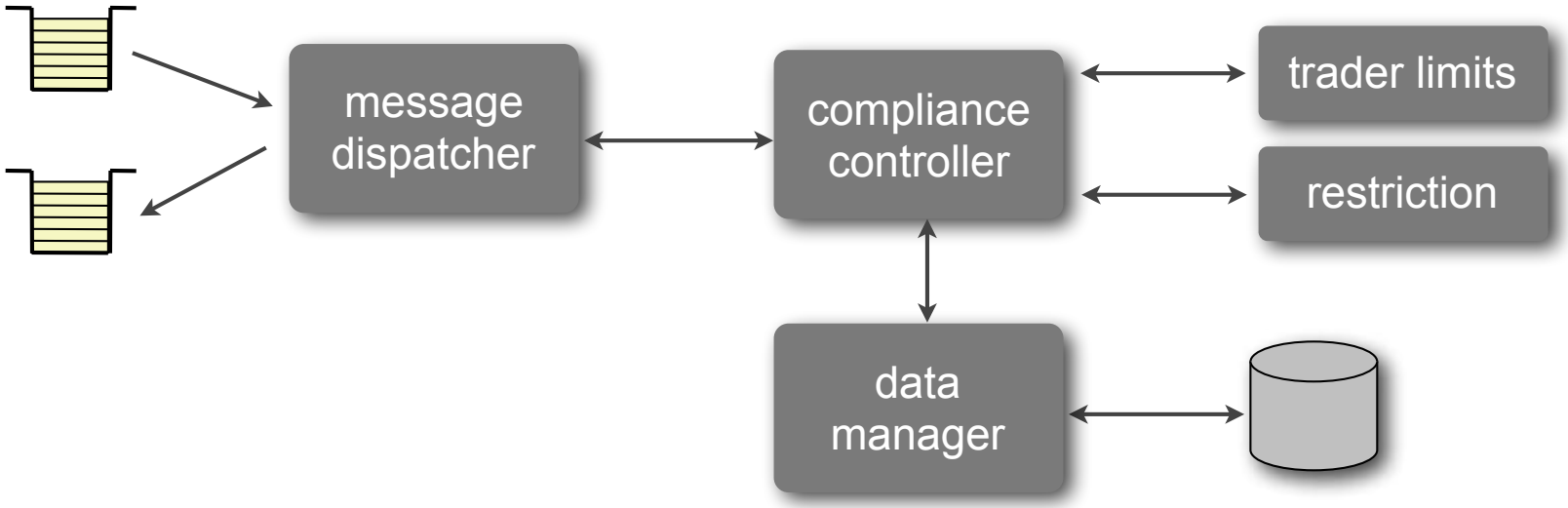
component identification

stock trade order validation



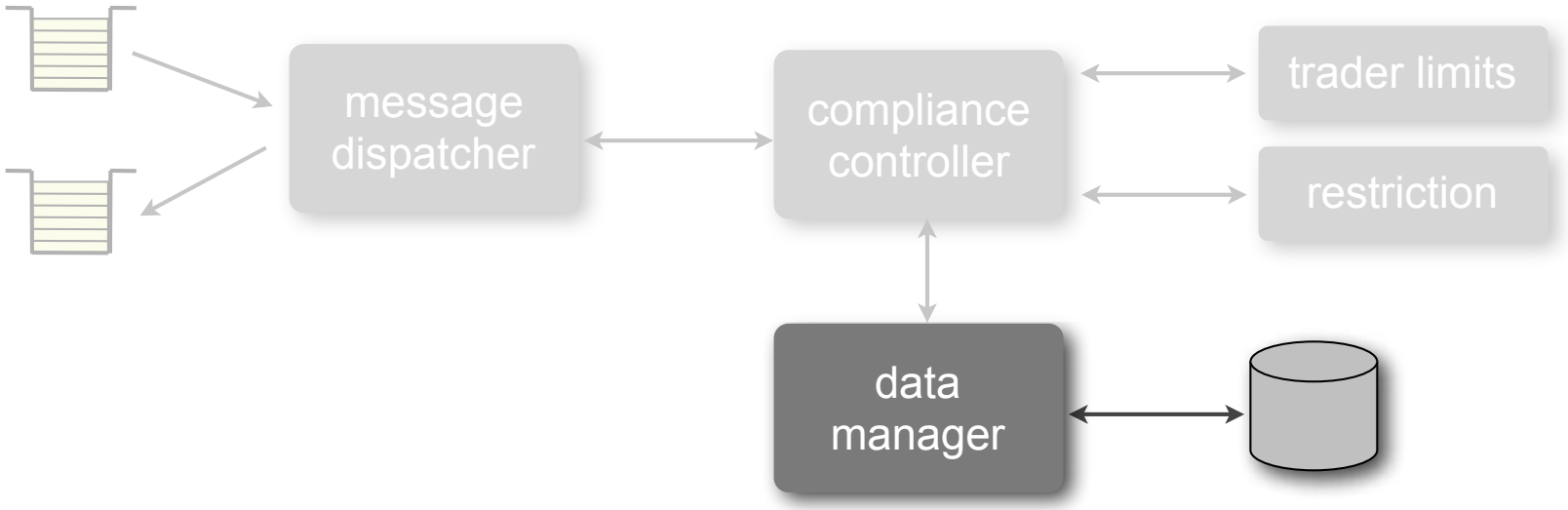
component identification

stock trade order validation



component identification

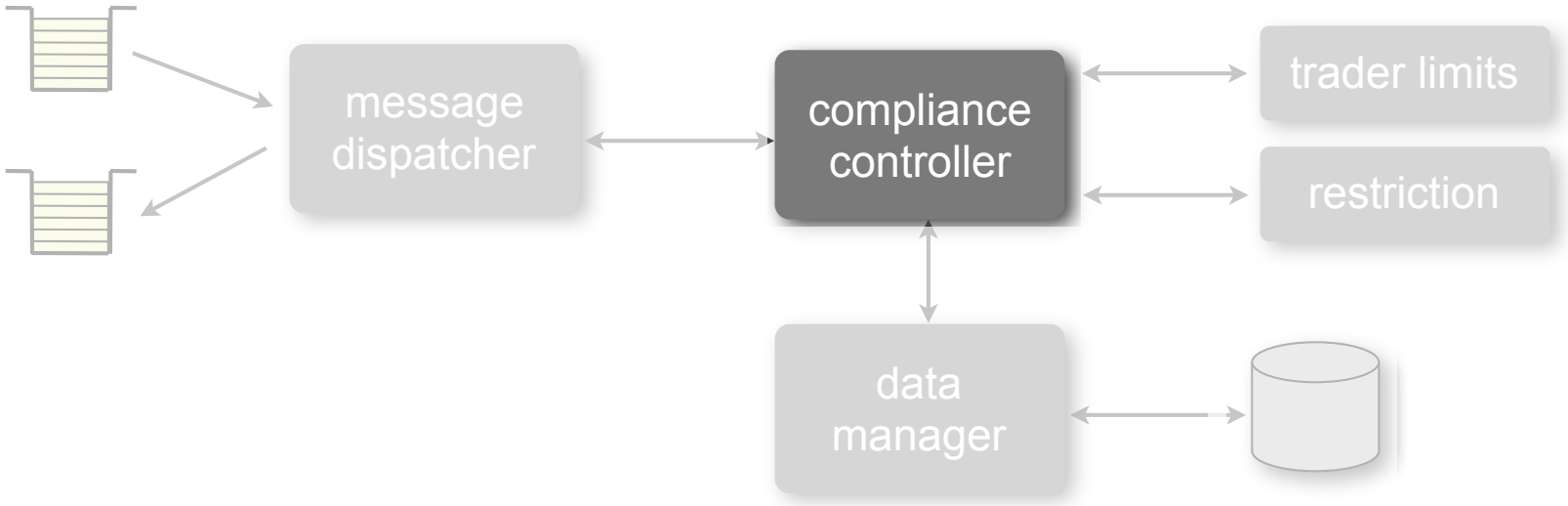
stock trade order validation



responsible for retrieving and caching all data needed by the compliance processors and persisting all validation errors.

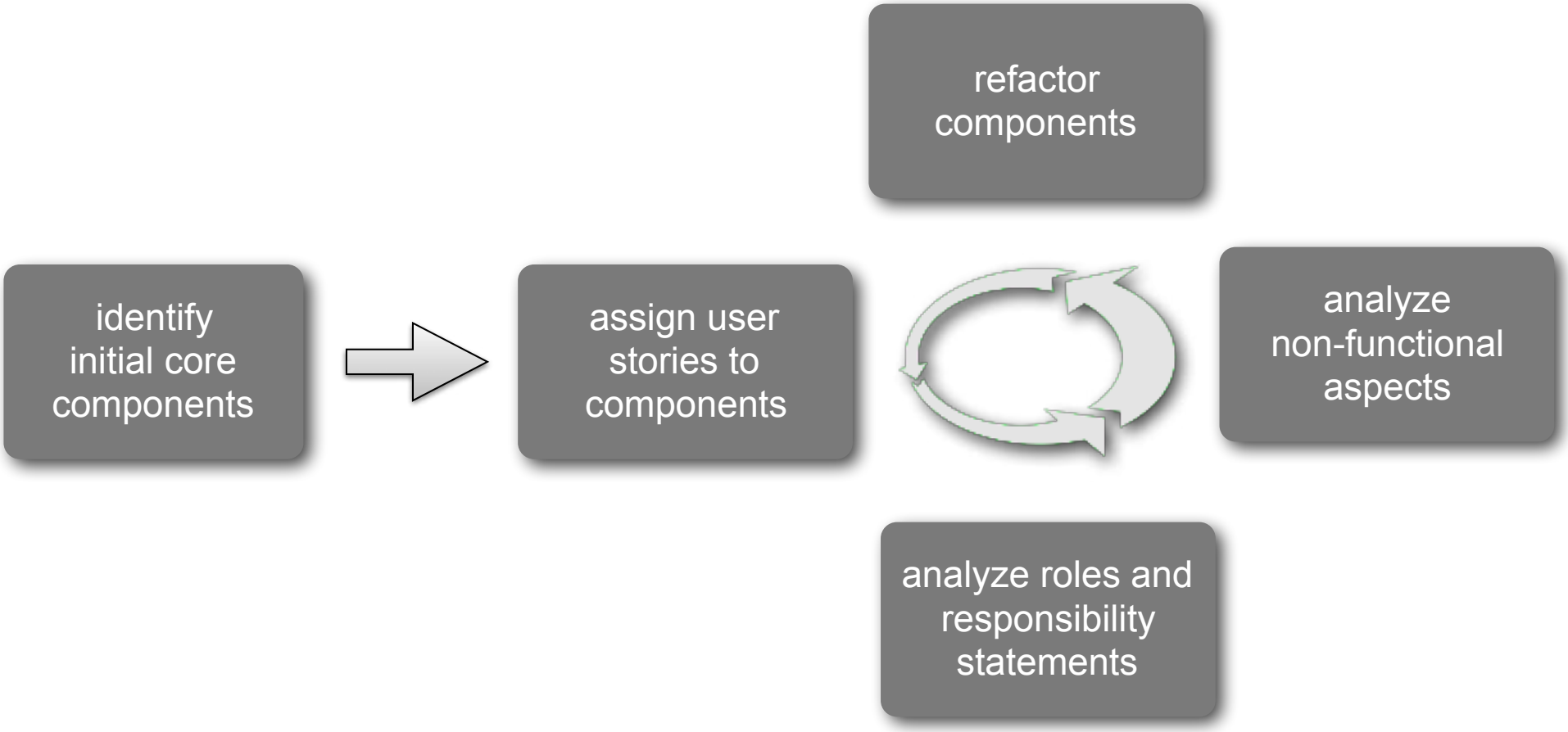
component identification

stock trade order validation



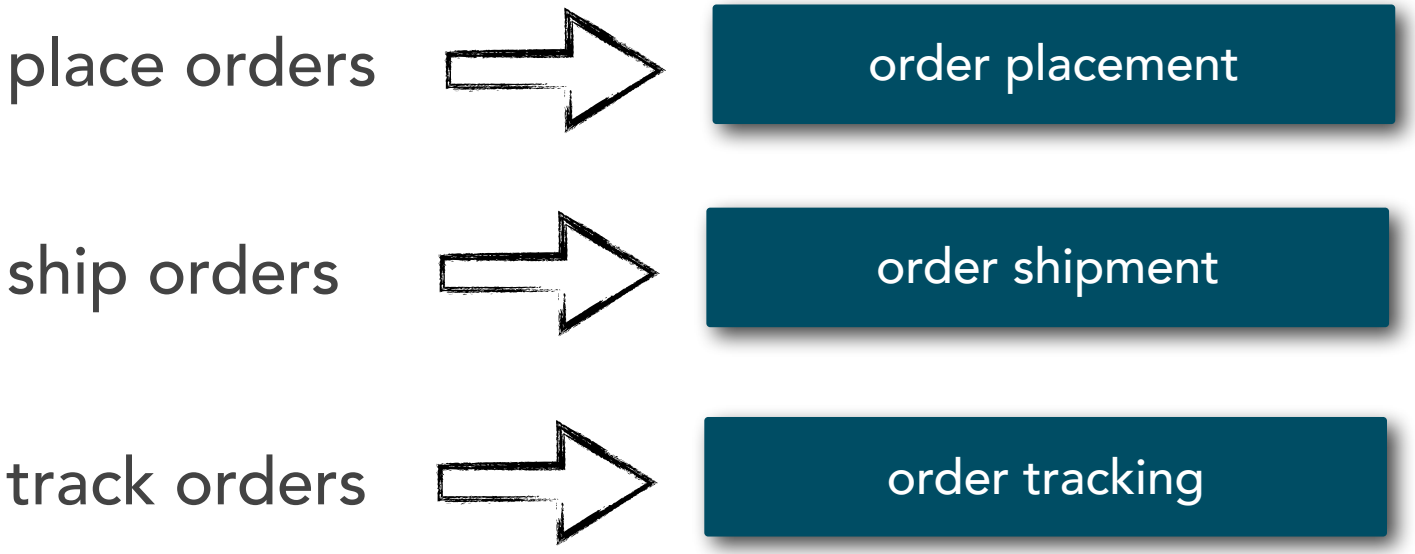
responsible for orchestrating the trade order validation process by calling specific compliance processors.

component identification



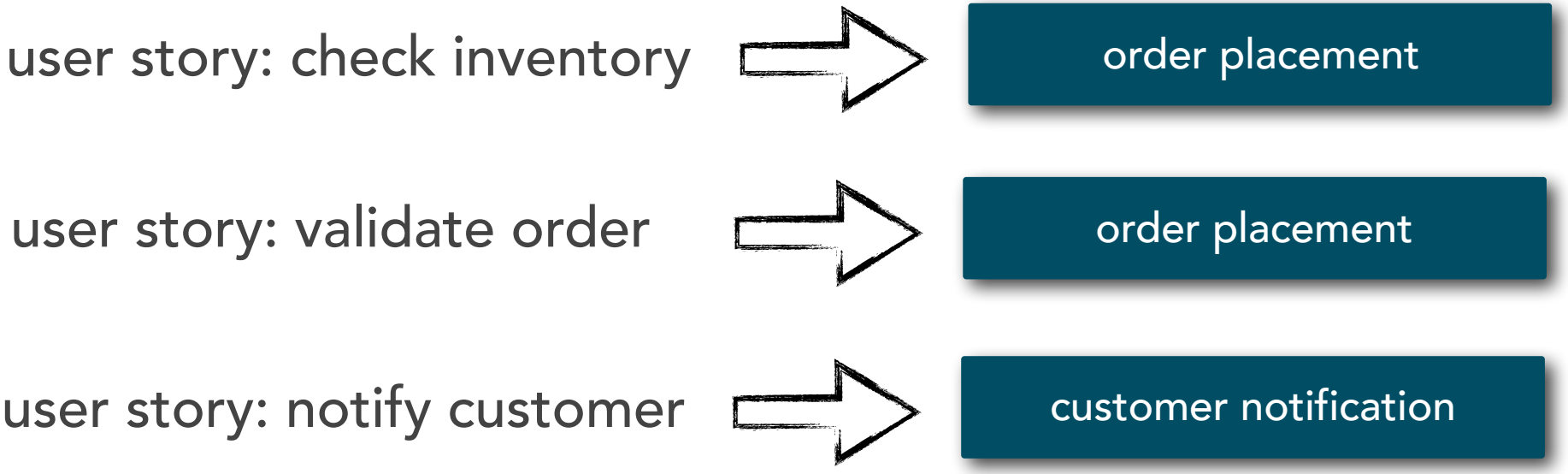
component identification

identify initial components using core functionality



component identification

assign requirements, use cases, or user stories to a component

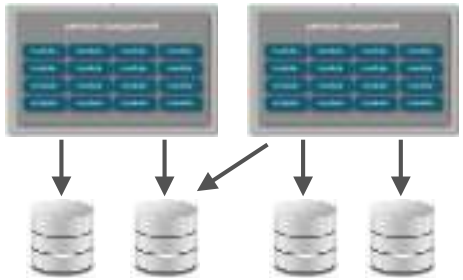


service identification

identify coarse-grained functional areas

create coarse-grained microservices

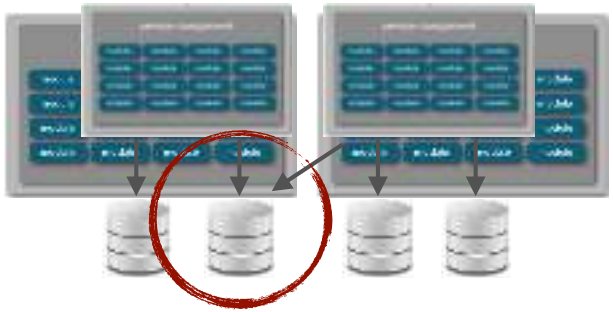
map services to data tables



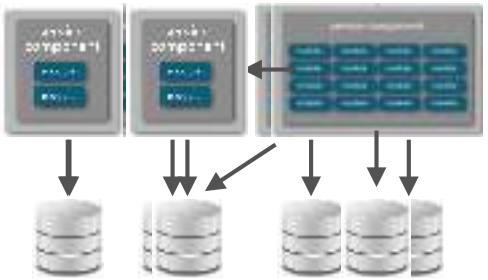
service identification

identify coarse-grained functional areas

identify data overlaps and dependencies



refine services and data



component identification

component granularity

responsible for creating, deleting, and updating orders. also responsible for shipping the order and tracking the order once it has been shipped. this component is also responsible for notifying the customer each time the order status changes.

component identification

component granularity

responsible for creating, deleting, and updating orders.

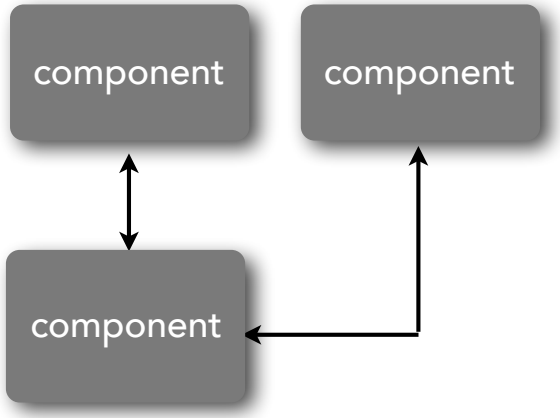
responsible for shipping and tracking orders

responsible for notifying the customer when the order status changes.

component coupling

component coupling

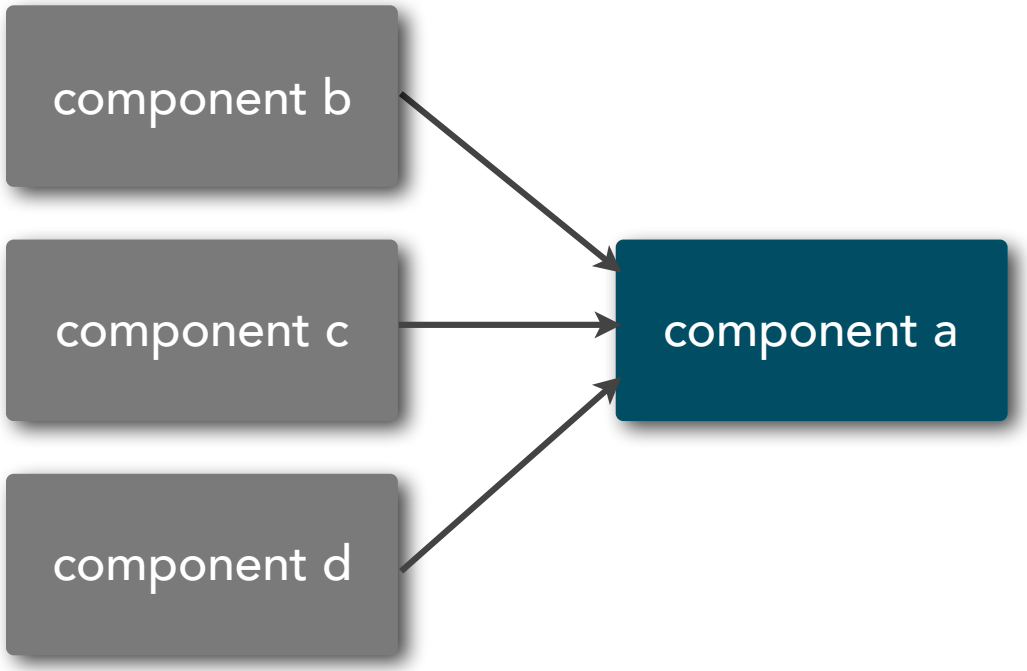
the extent to which components know about each other



component coupling

afferent coupling

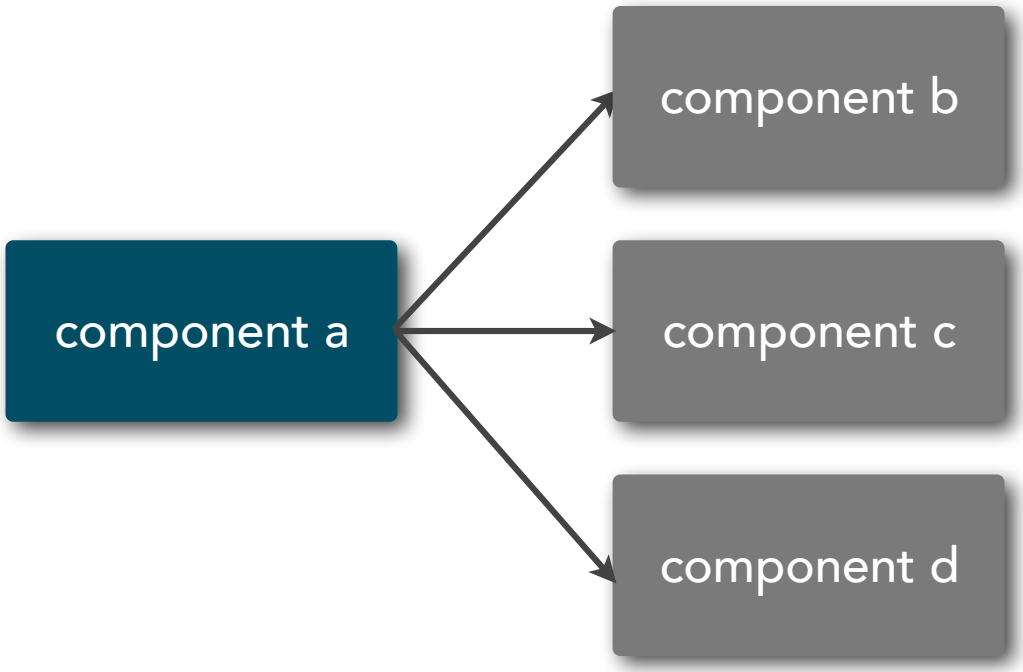
the degree to which other components are dependent on the target component



component coupling

efferent coupling

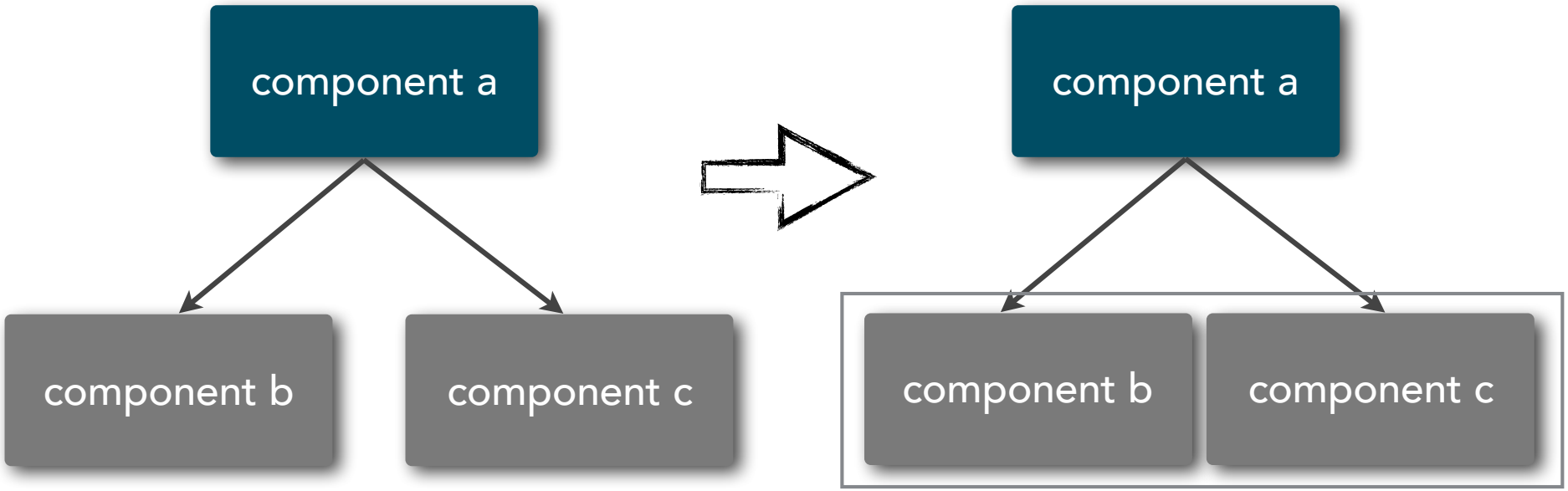
the degree to which the target component is dependent on other components



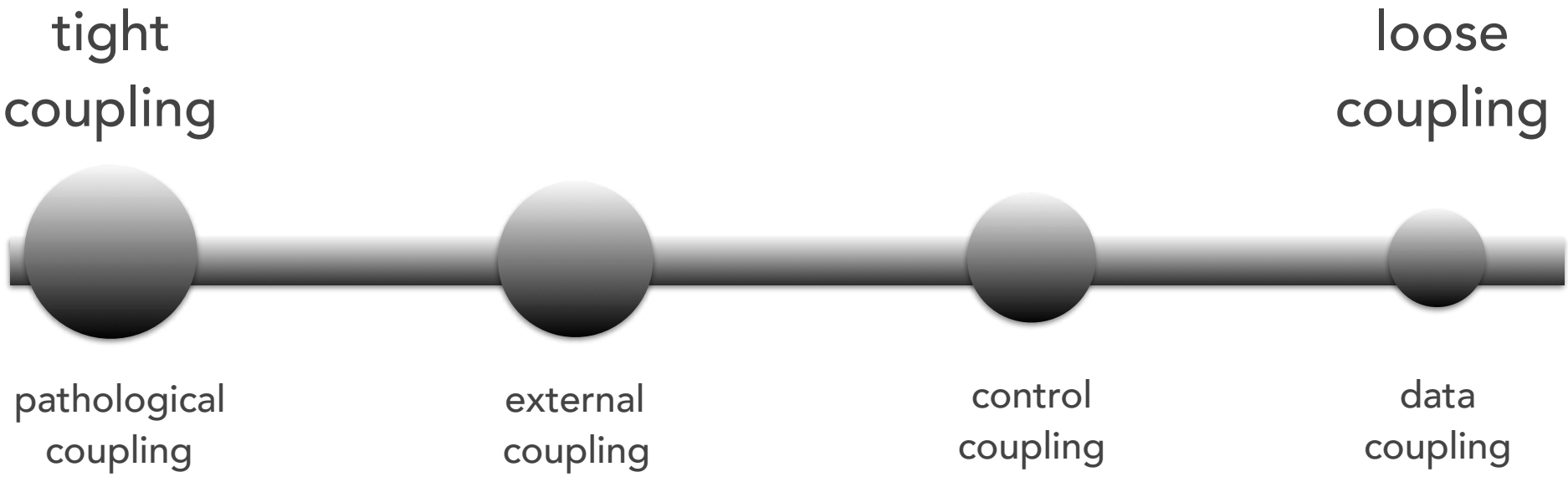
component coupling

temporal coupling

functionality is grouped into one component due to timing dependencies (e.g. transactions)



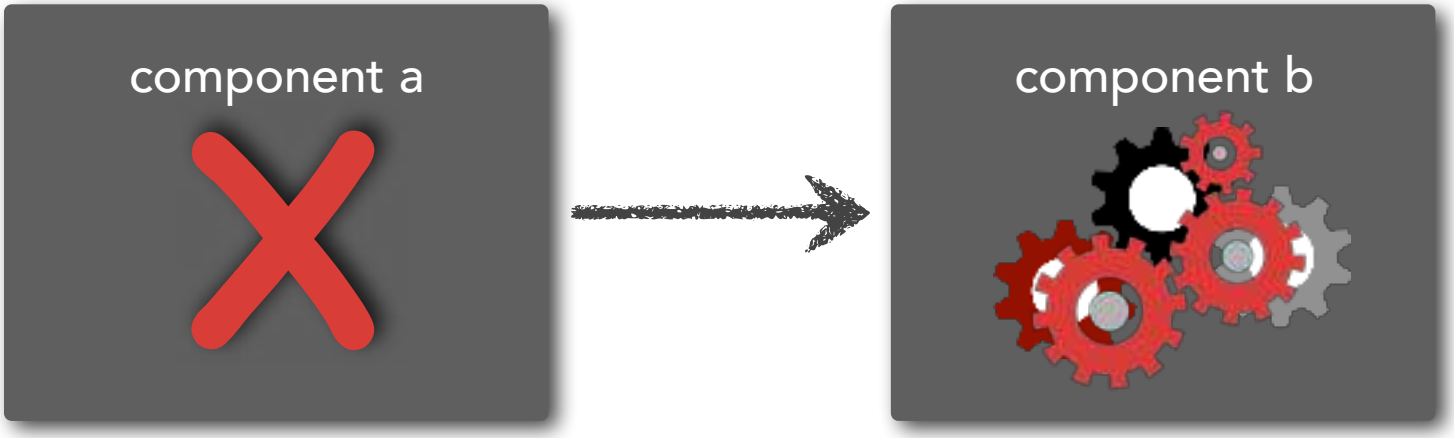
component coupling



component coupling

pathological coupling

one component relies on the inner workings of another component



component coupling

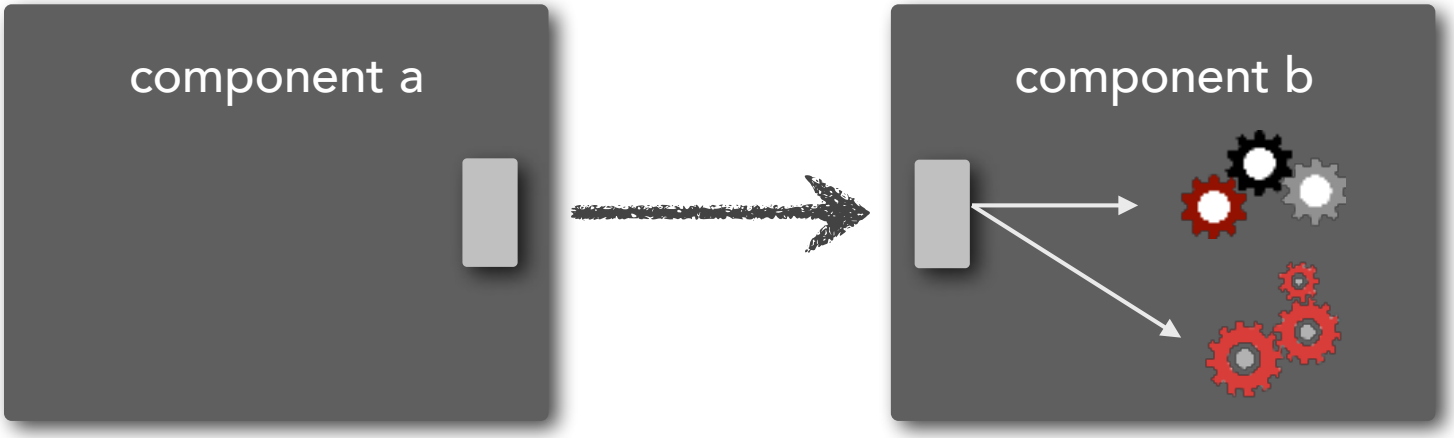
external coupling

multiple components share an externally imposed protocol or data format

component coupling

control coupling

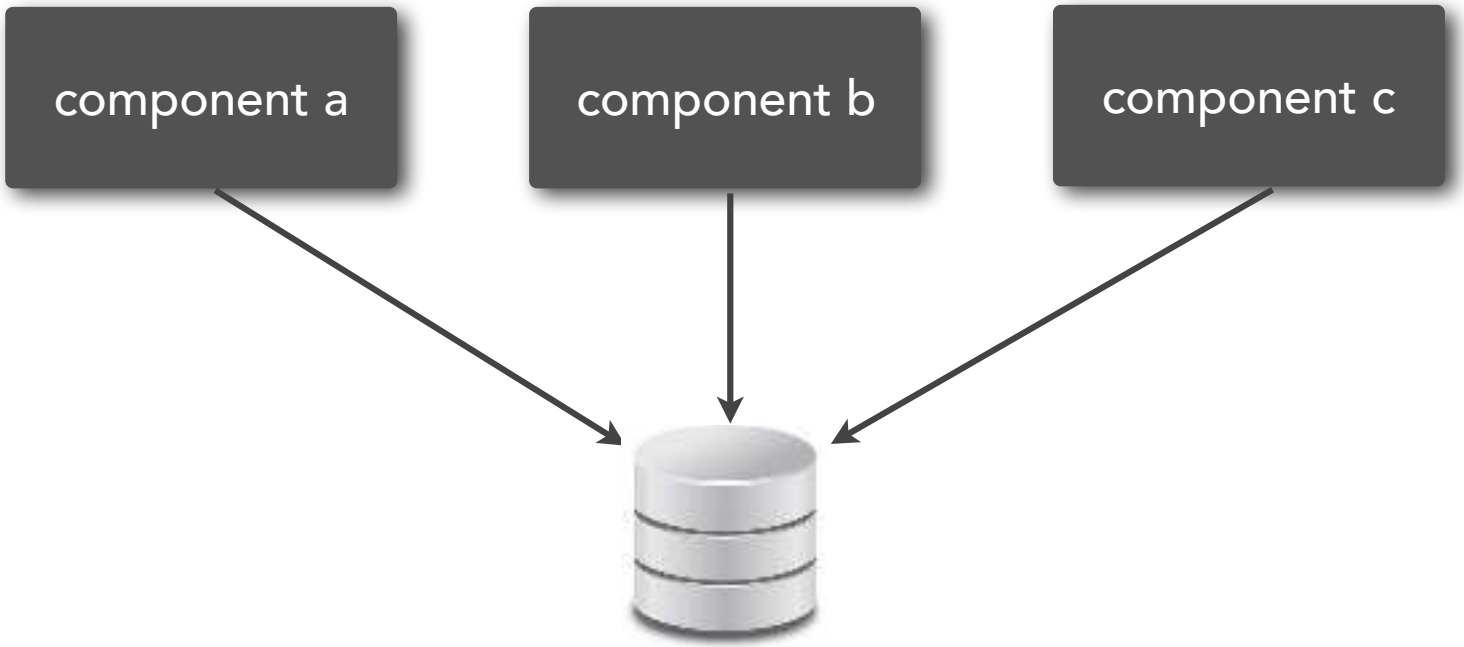
one component passes information to another component on what to do



component coupling

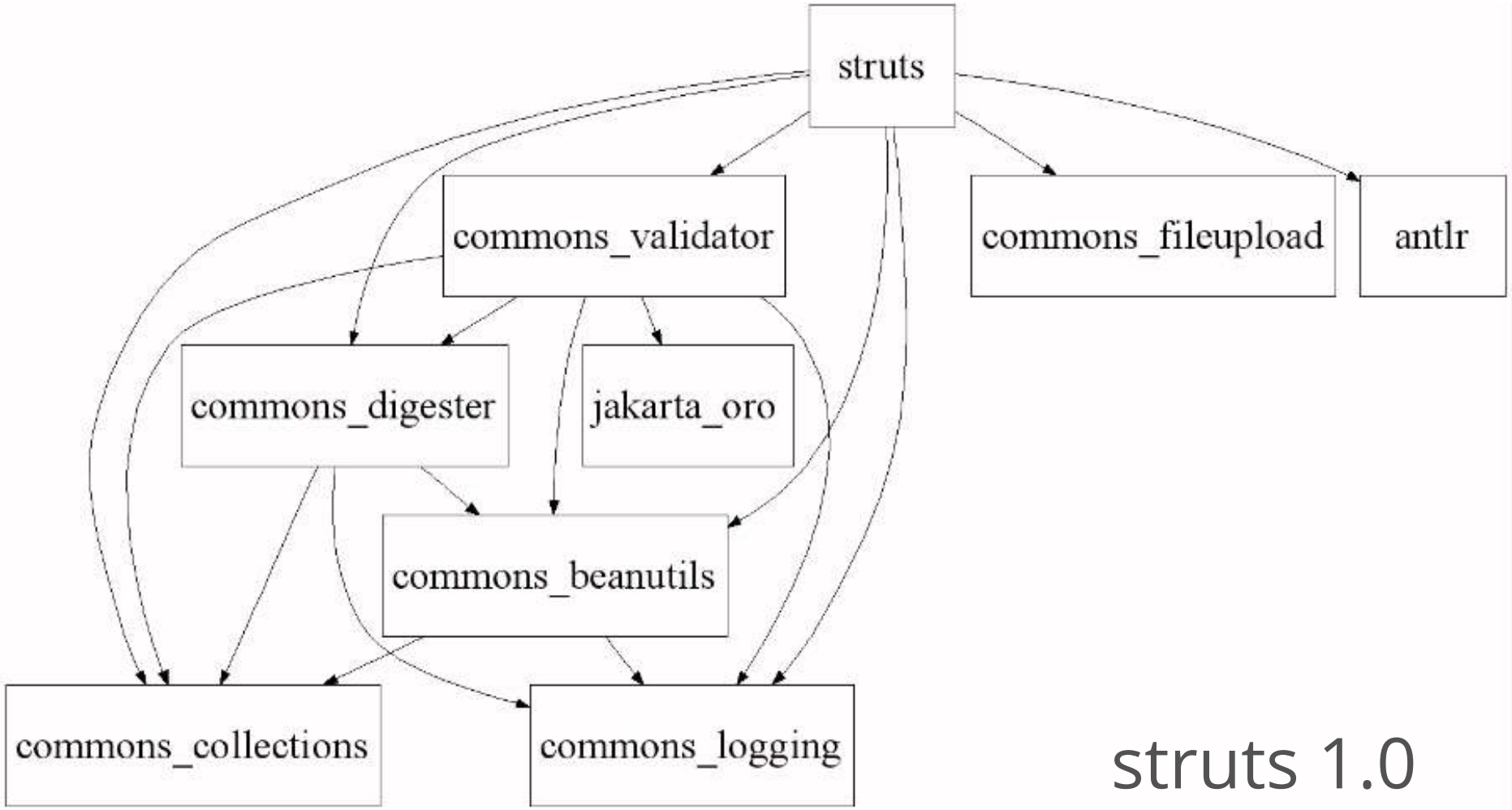
data coupling

the degree to which components are bound to a shared data context

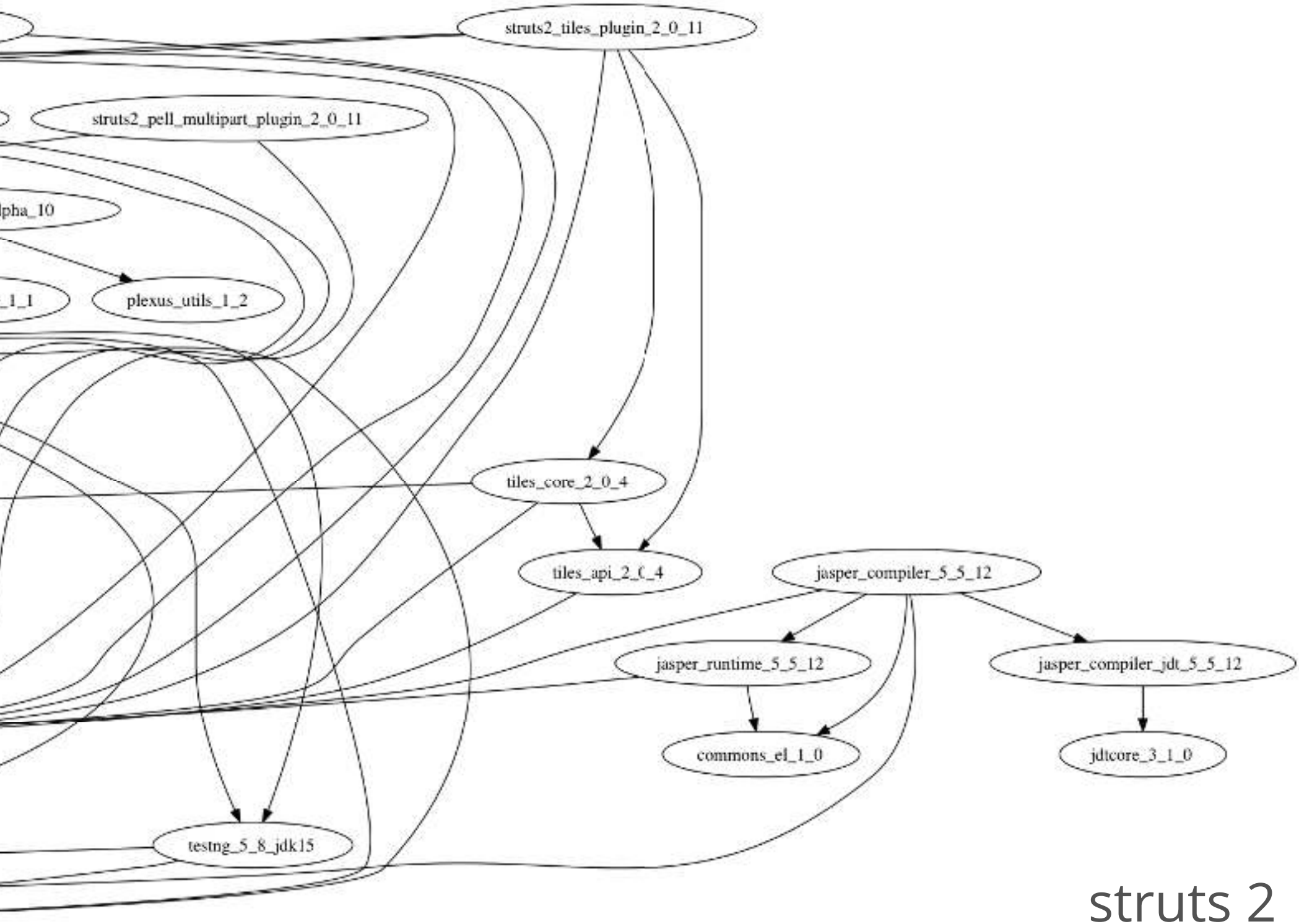


component coupling

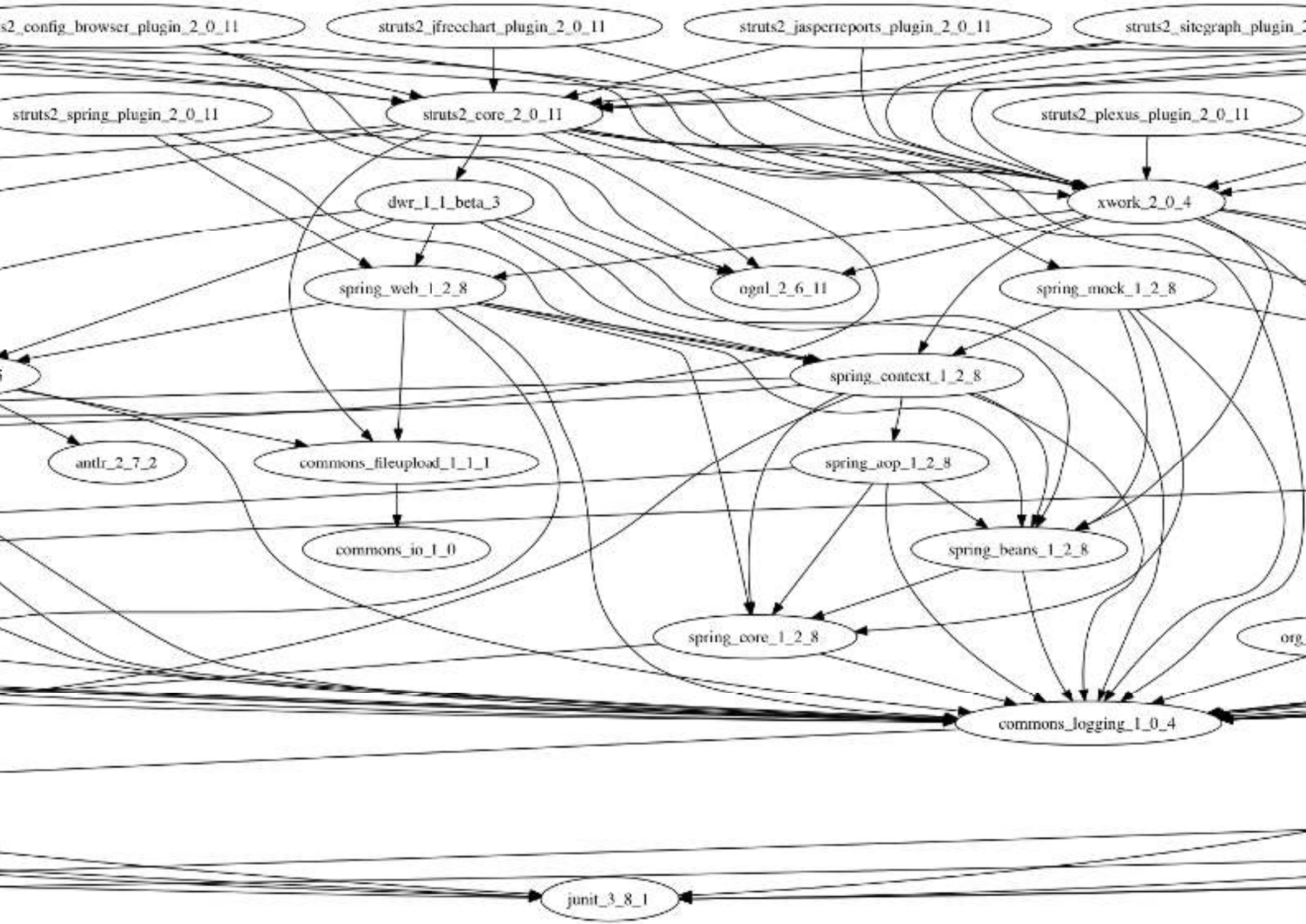
consequences of ignoring...



struts 1.0


struts2_tiles_plugin_2_0_11
struts2_pell_multipart_plugin_2_0_11
plexus_utils_1_2
tiles_core_2_0_4
tiles_api_2_0_4
jasper_compiler_5_5_12
jasper_runtime_5_5_12
jasper_compiler_jdt_5_5_12
commons_el_1_0
jdtcore_3_1_0
testng_5_8_jdk15

struts 2


s2_config_browser_plugin_2_0_11
struts2_jfreechart_plugin_2_0_11
struts2_jasperreports_plugin_2_0_11
struts2_sitegraph_plugin_2
struts2_spring_plugin_2_0_11
struts2_core_2_0_11
struts2_plexus_plugin_2_0_11
dwr_1_1_beta_3
xwork_2_0_4
spring_web_1_2_8
ognl_2_6_11
spring_mock_1_2_8
spring_context_1_2_8
antlr_2_7_2
commons_fileupload_1_1_1
spring_aop_1_2_8
commons_io_1_0
spring_beans_1_2_8
spring_core_1_2_8
org_
commons_logging_1_0_4
junit_3_8_1

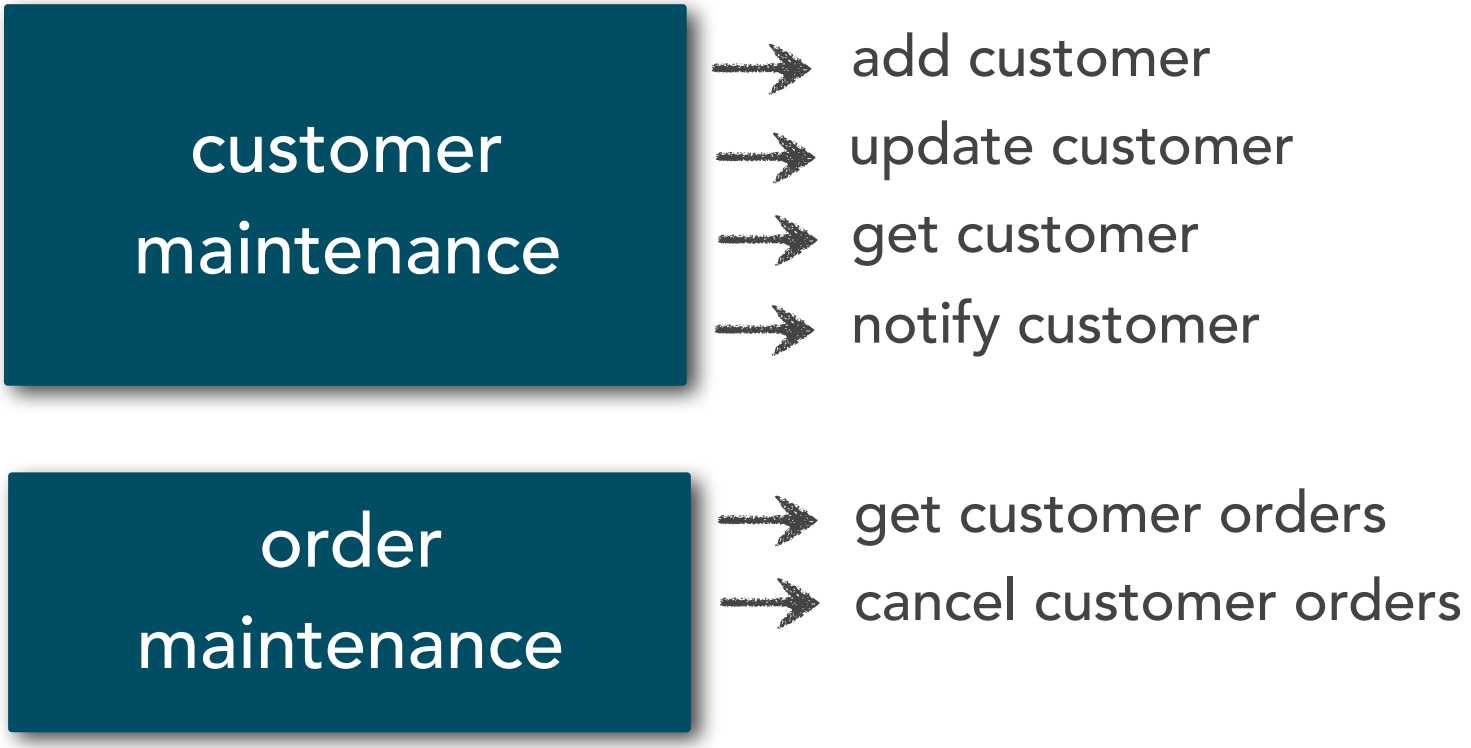
component cohesion

component cohesion

the degree and manner to which the operations of a component are related to one another

component cohesion

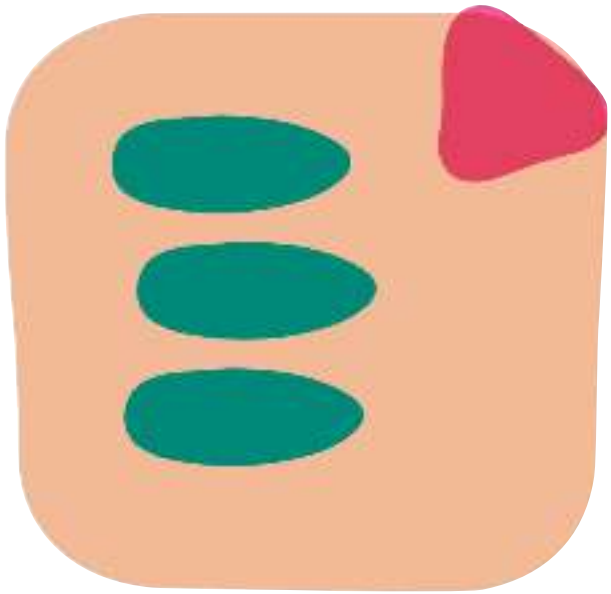
the degree and manner to which the operations of a component are related to one another



architecture katas

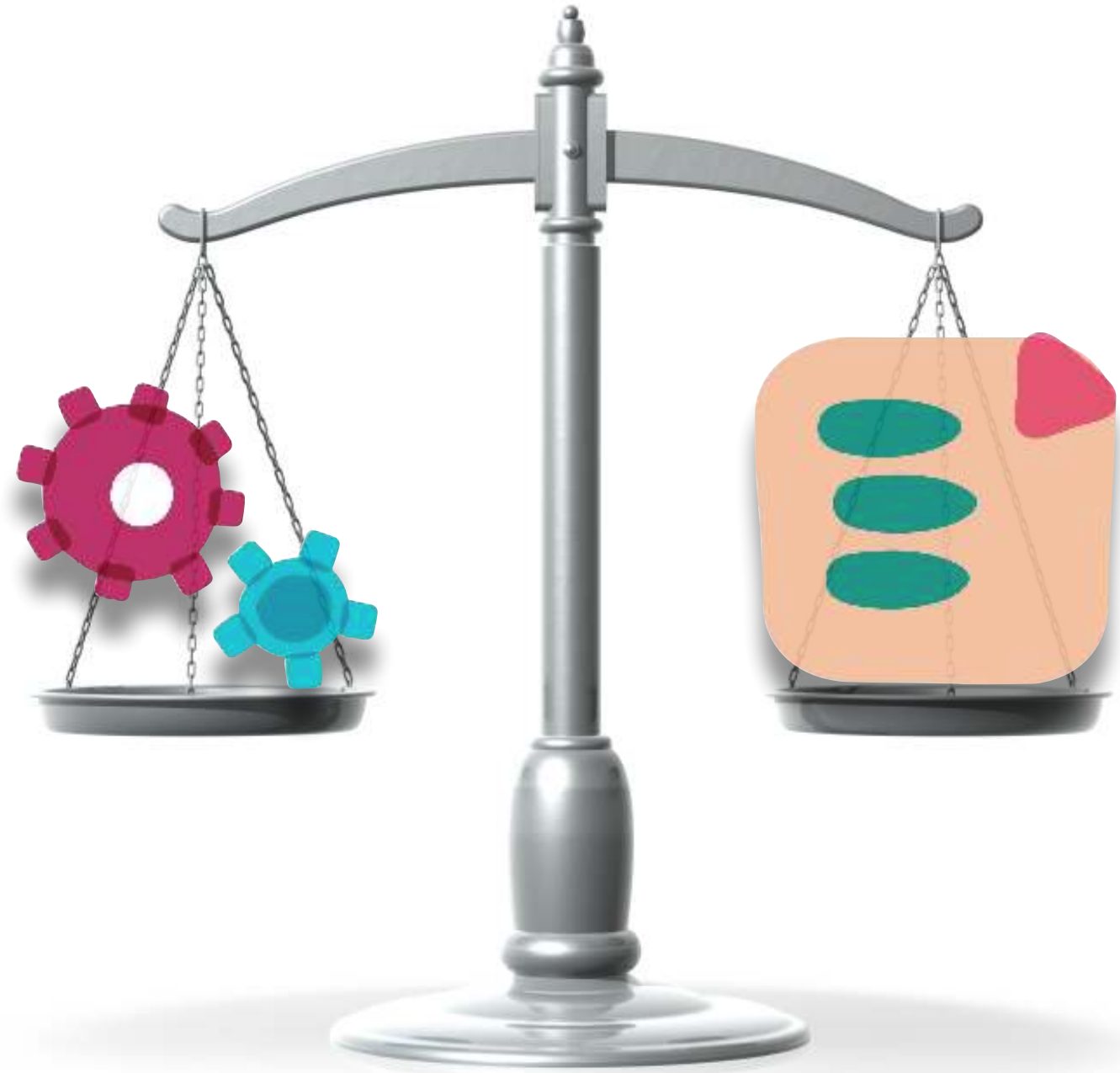
identifying major architecture components

documenting software architectures

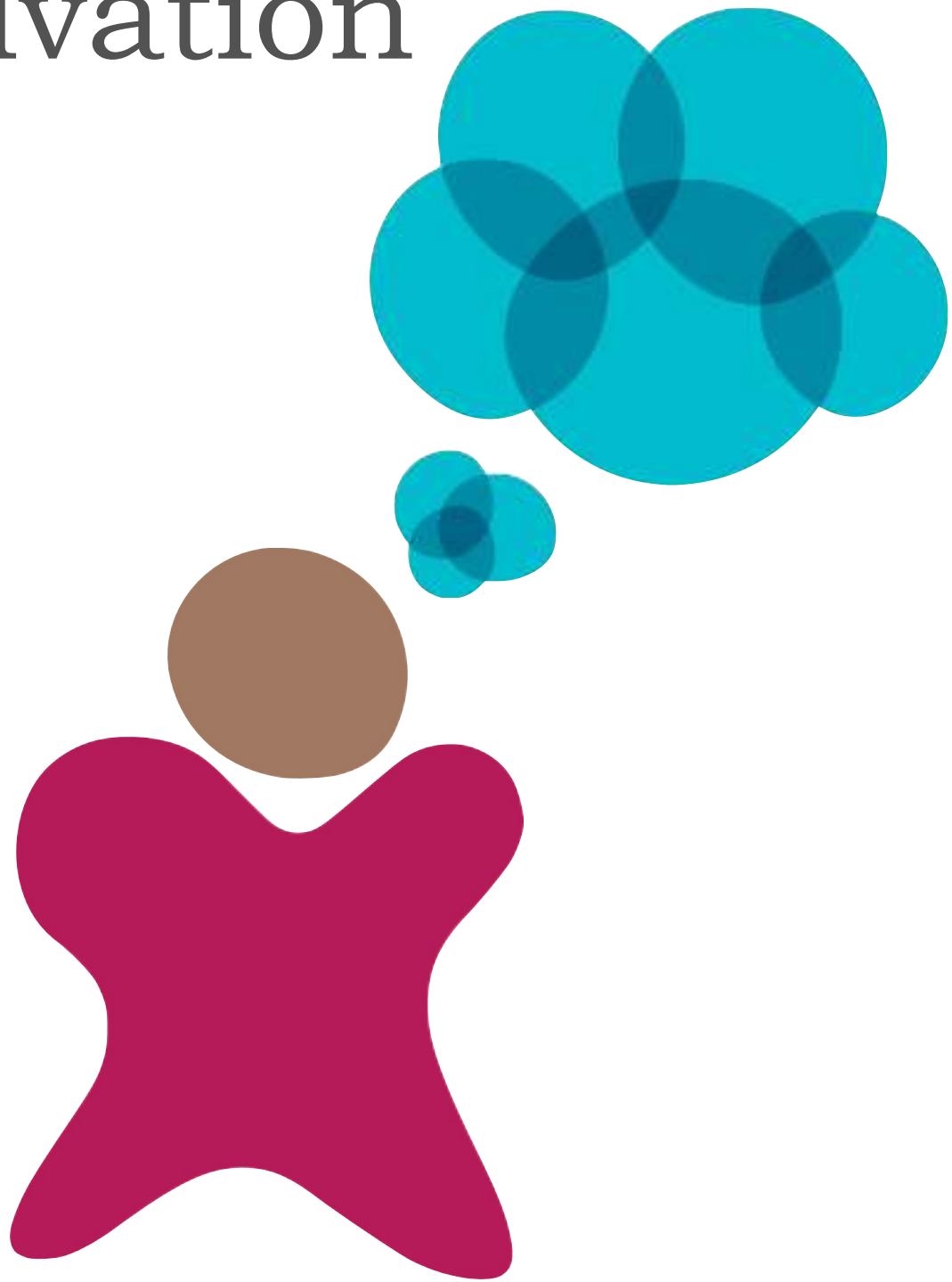


documentation

weight in bytes

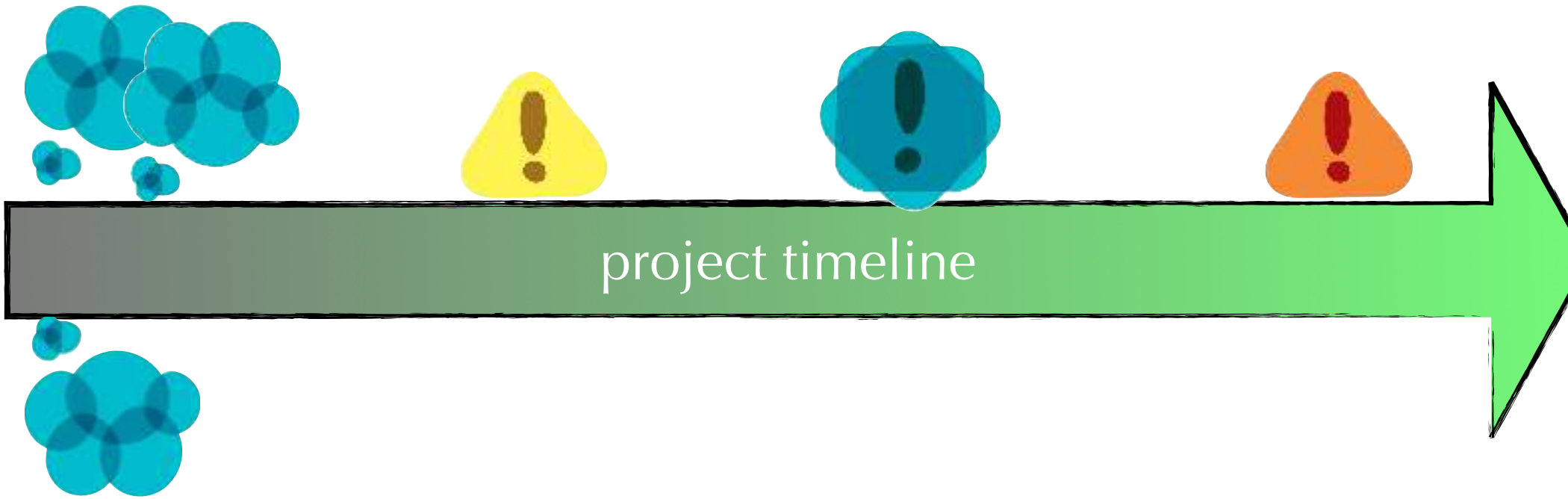


motivation

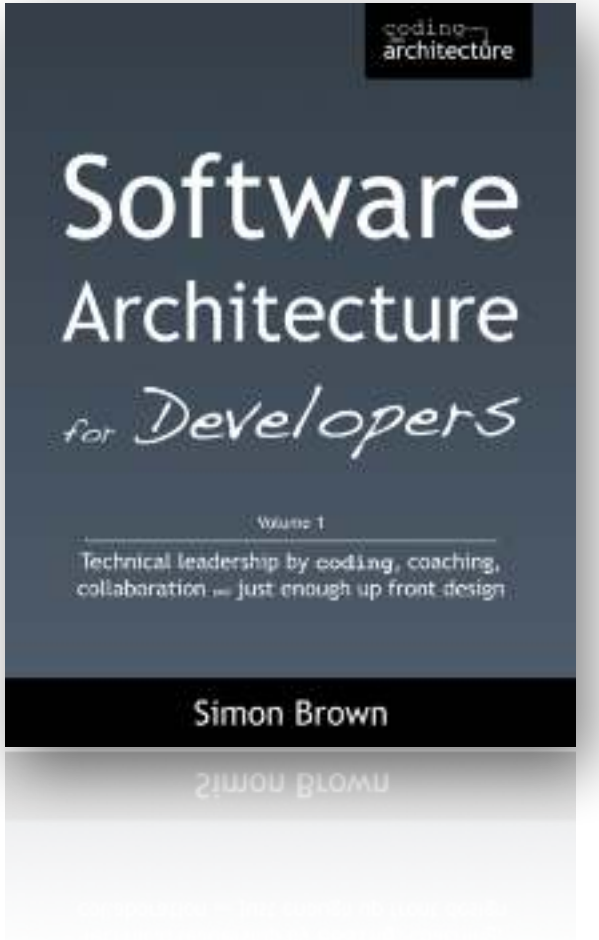


agile architecture

Not all decisions made up front.



diagraming techniques



http://www.codingthearchitecture.com

Titles

Short and meaningful, numbered if diagram order is important

Lines

Favor unidirectional arrows, add descriptive text to provide additional information

Layout

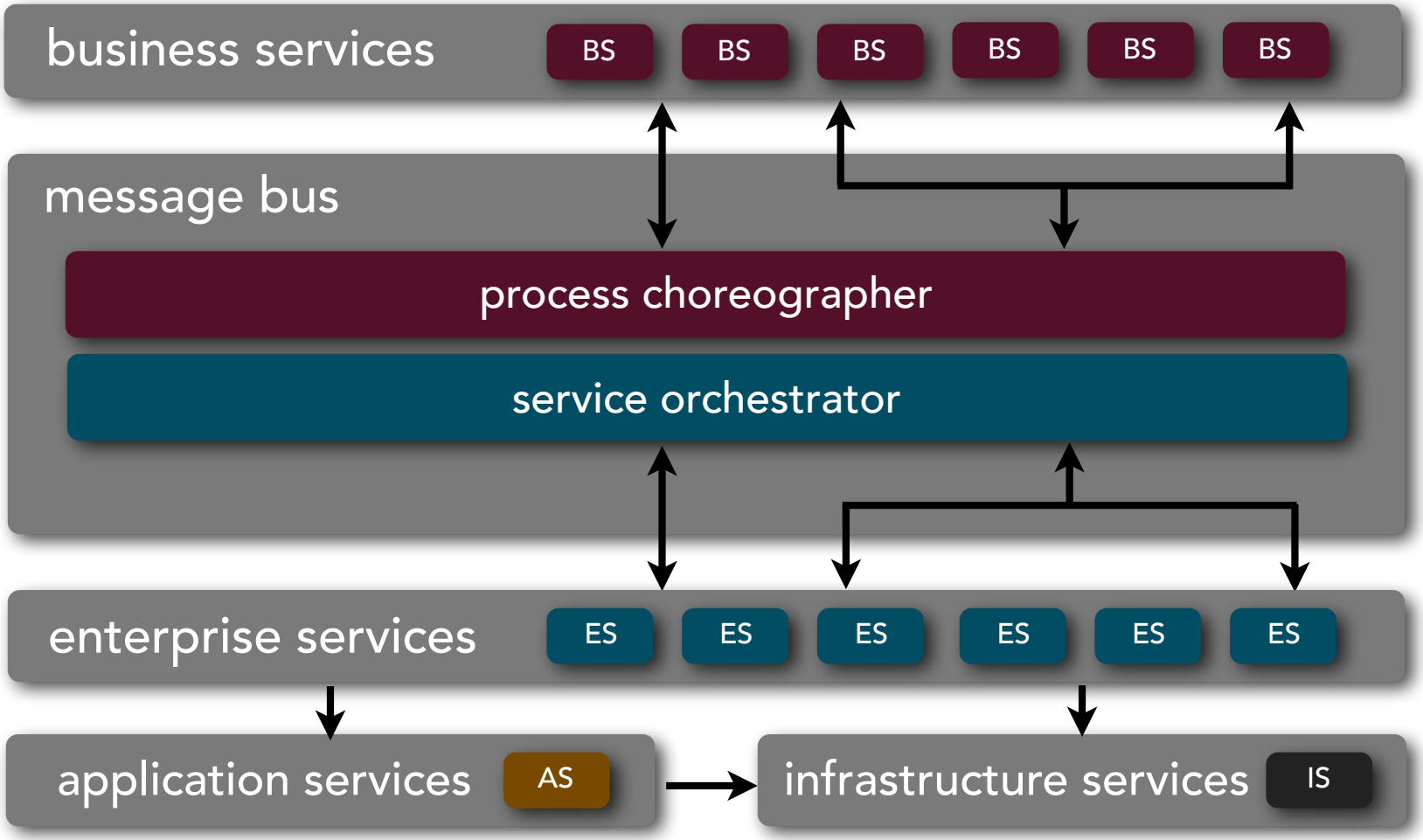
Sticky notes and index cards make a great substitute for drawn boxes, especially early on

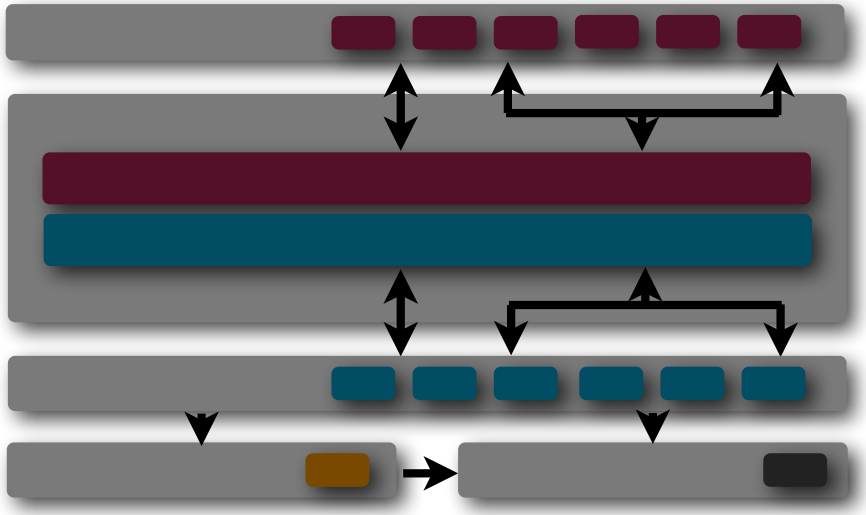
Labels

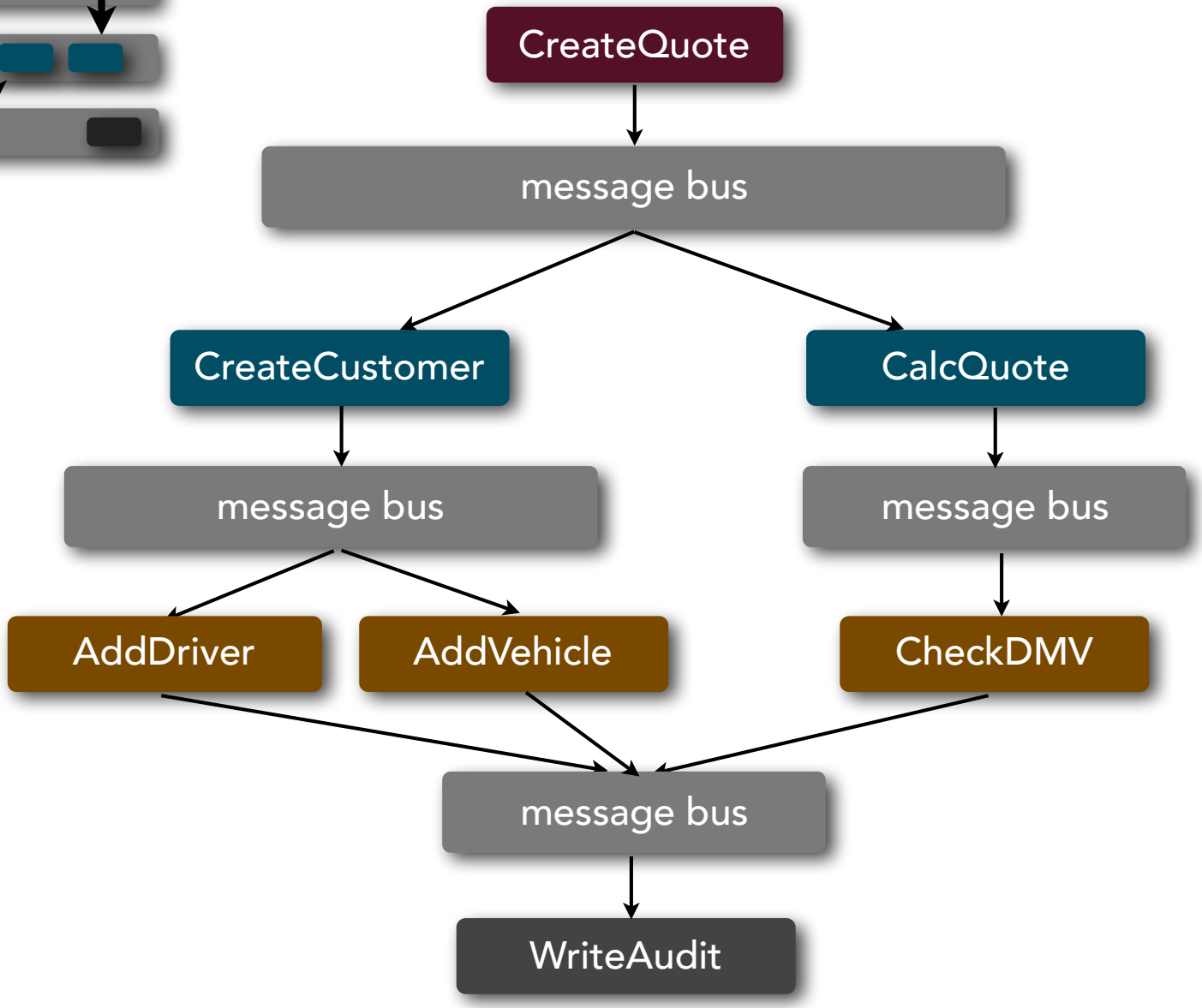
Be wary of using acronyms, especially those related to the business/domain that you work in

Color

Ensure that color coding is made explicit; watch out for color-blindness and black/white printers


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




CreateQuote
message bus
CreateCustomer
CalcQuote
message bus
message bus
AddDriver
AddVehicle
CheckDMV
message bus
WriteAudit

Color

Ensure that color coding is made explicit; watch out for color-blindness and black/white printers

Orientation

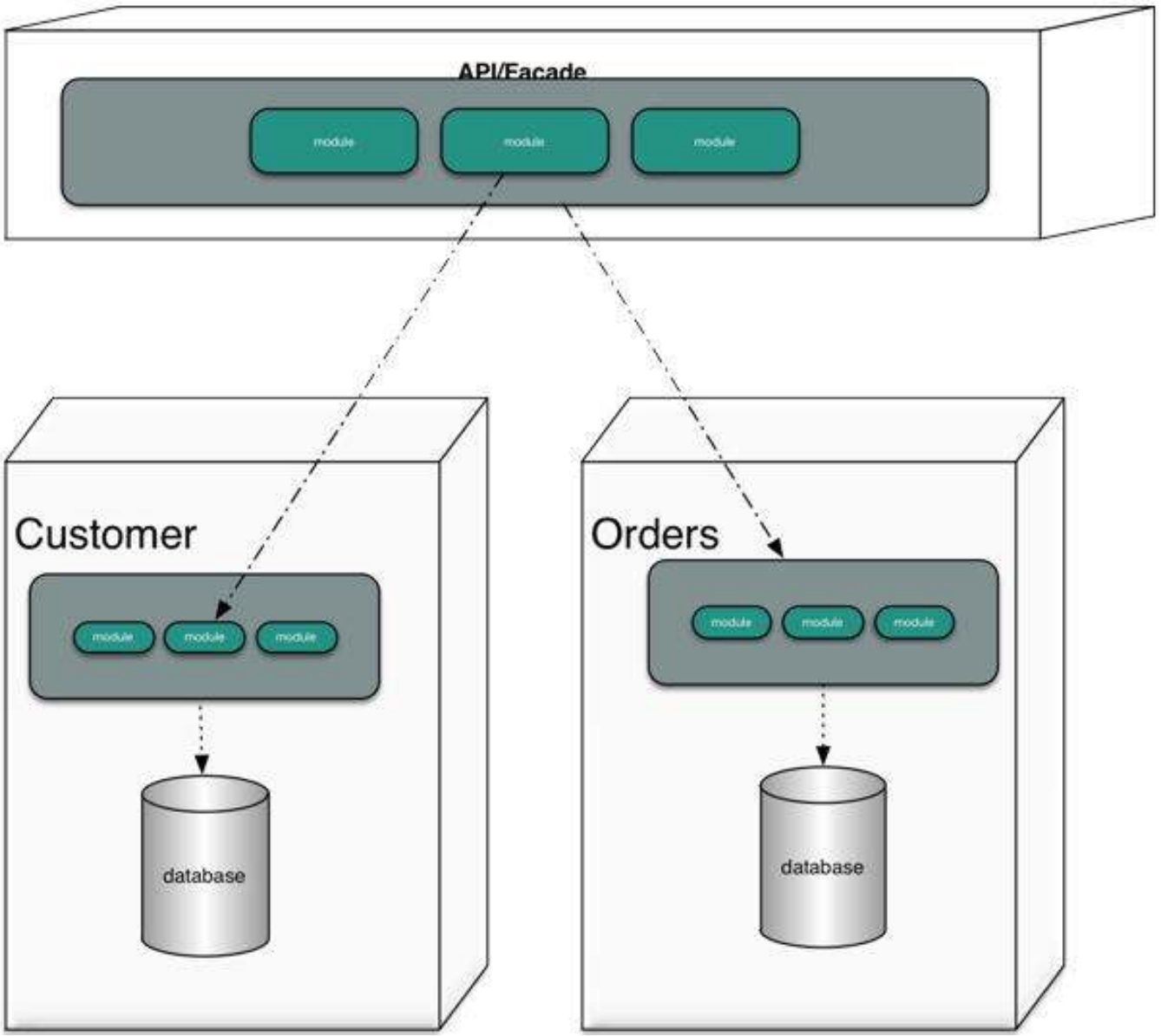
Most important thing in the middle;
be consistent across diagrams

Shapes

Don’t assume that people will understand what different shapes are being used for

Shapes

Be consistent.


API/Facade
module
module
module
Customer
module
module
module
database
Orders
module
module
module
database

Shapes

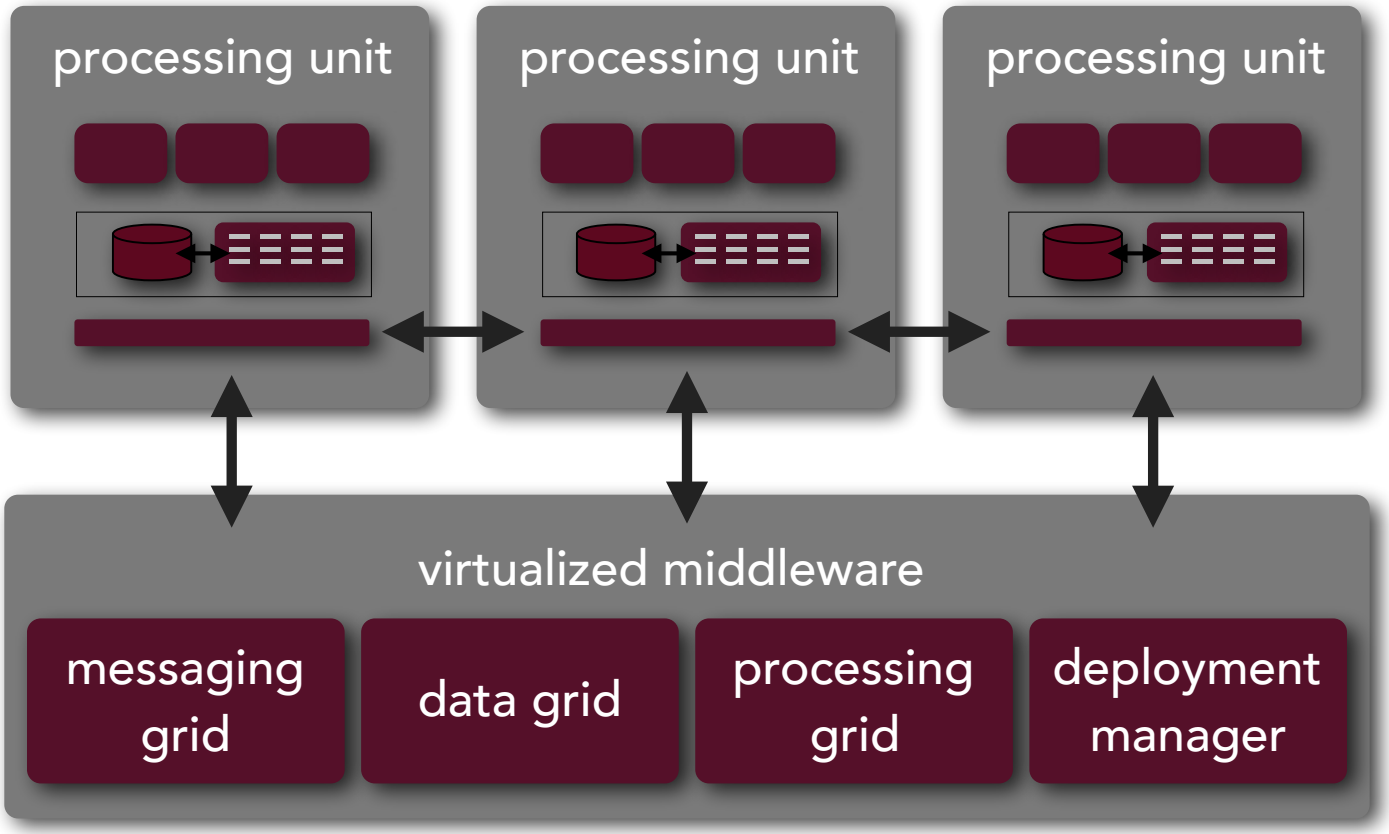
Be consistent.

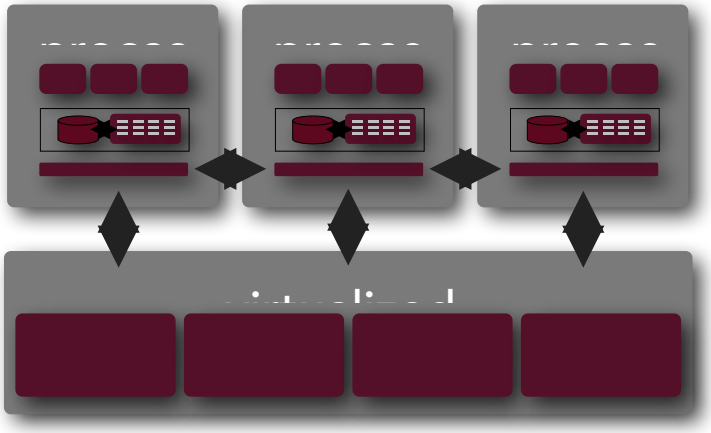
Keys

Explain shapes, lines, colors, borders, acronyms, etc

Representational Consistency

Don’t abruptly change scale on diagrams; provide context for where this portion fits into the bigger picture


processing unit
processing unit
processing unit
virtualized middleware
messaging grid
data grid
processing grid
deployment manager

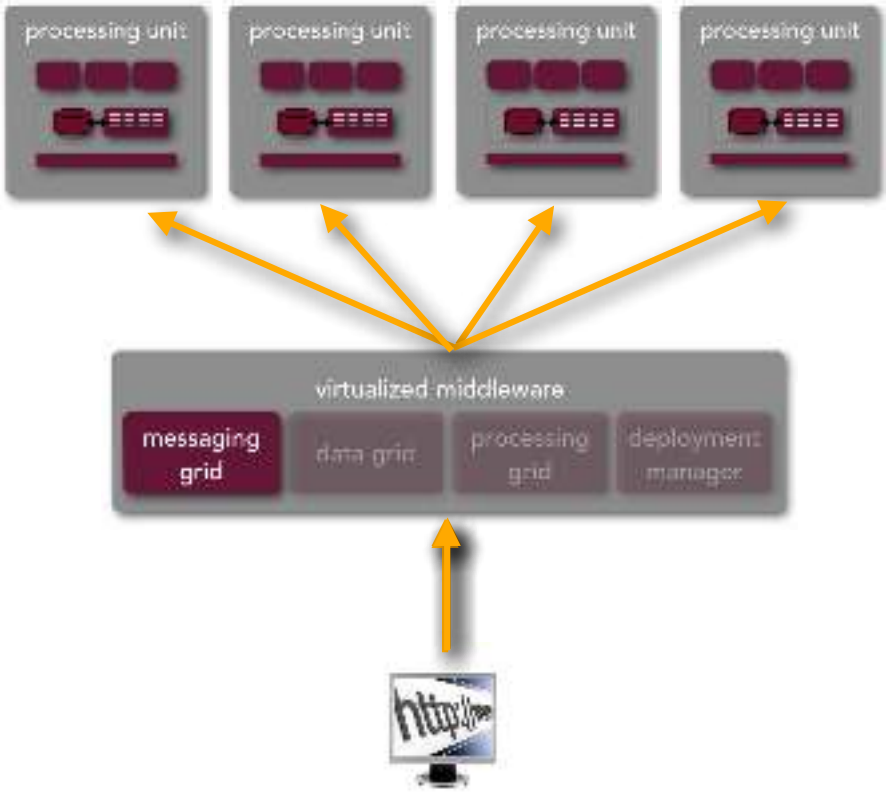


messaging grid

data grid

processing grid

deployment manager

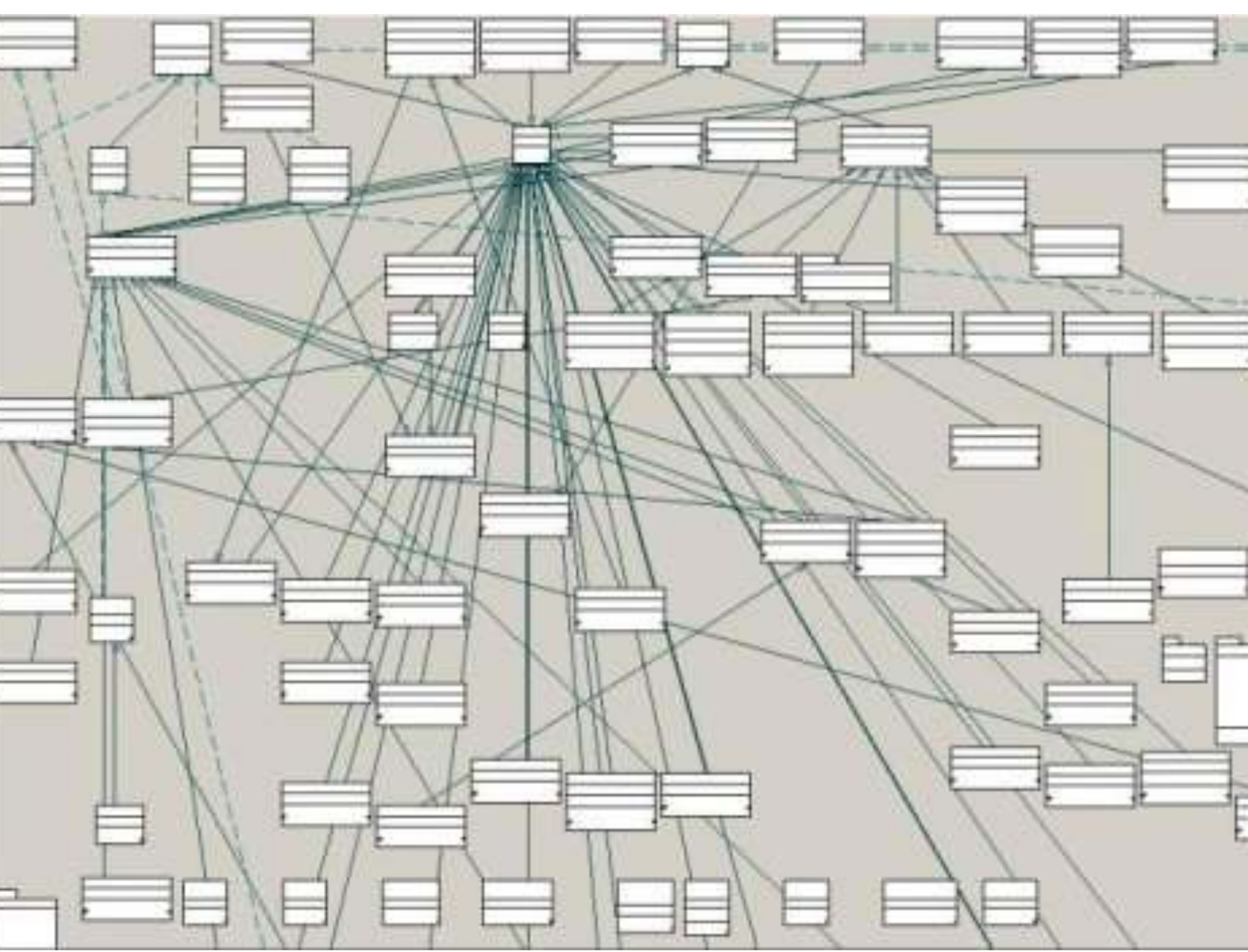


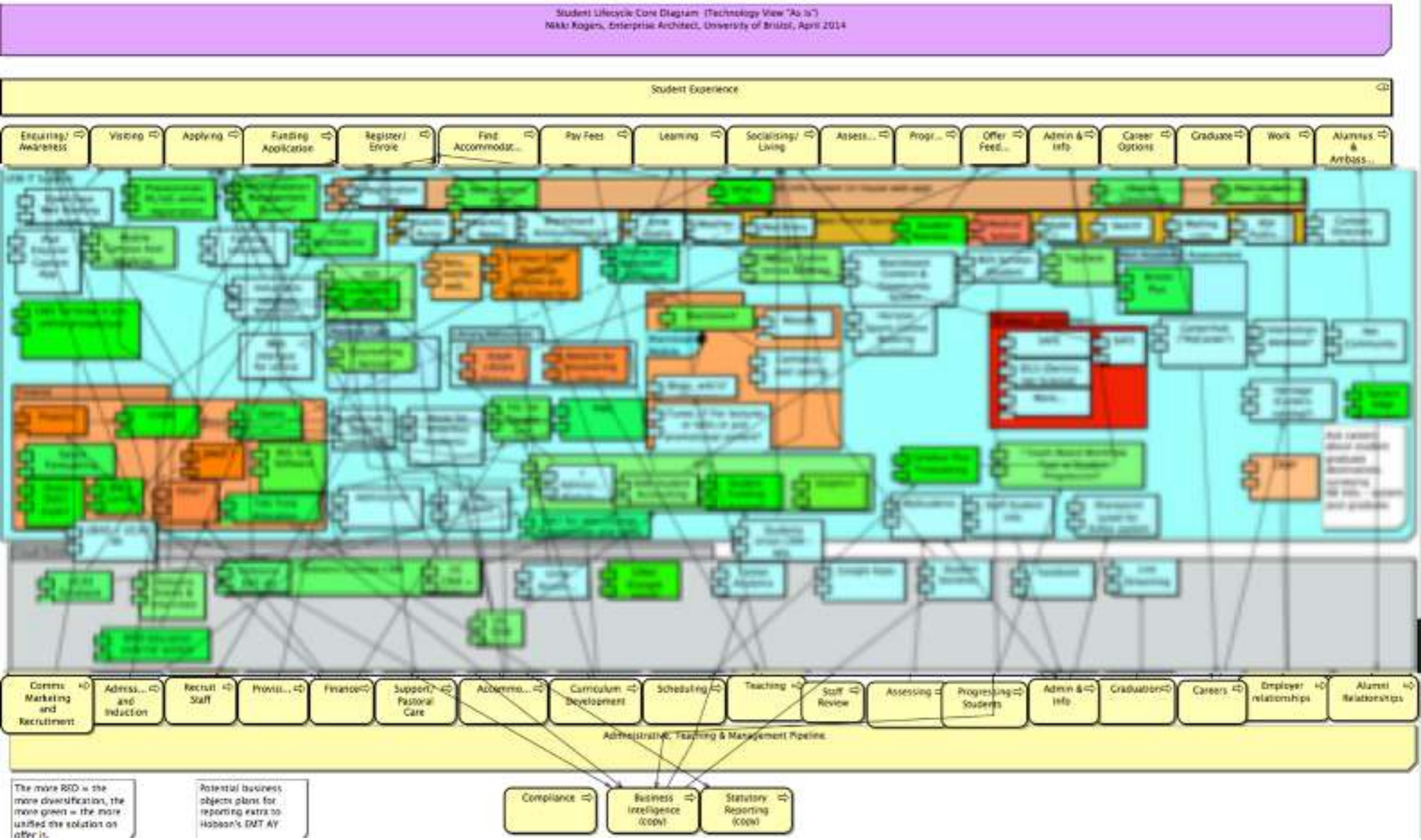
Representational Consistency

Don’t abruptly change scale on diagrams; provide context for where this portion fits into the bigger picture

Comprehensive Diagram

Don’t try to capture the entirety of software architecture in a single diagram; create dimensional views instead




Student Lifecycle Core Diagram (Technology View "As Is")
Nikki Rogers, Enterprise Architect, University of Bristol, April 2014
Student Experience
Enquiring/ Awareness
Visiting
Applying
Funding Application
Register/ Enrole
Find Accommodat...
Pay Fees
Learning
Socialising/ Living
Assess...
Progr...
Offer Feed...
Admin & Info
Career Options
Graduate
Work
Alumnus & Ambass...
Comms Marketing and Recruitment
Admiss... and Induction
Recruit Staff
Provis...
Finance
Support/ Pastoral Care
Accomms...
Curriculum Development
Scheduling
Teaching
Staff Review
Assessing
Progressing Students
Admin & Info
Graduation
Careers
Employer relationships
Alumni Relationships
Administrative, Teaching & Management Pipeline
Compliance
Business Intelligence (copy)
Statutory Reporting (copy)
The more RED = the more diversification, the more green = the more unified the solution on offer is.
Potential business objects plans for reporting extra to Hobson's DMT AY

Comprehensive Diagram

Don’t try to capture the entirety of software architecture in a single diagram; create dimensional views instead

The C4 model

System Context

The system plus users and system dependencies

Containers

The overall shape of the architecture and technology choices

Components

Components and their interactions within a container

Classes (or Code)

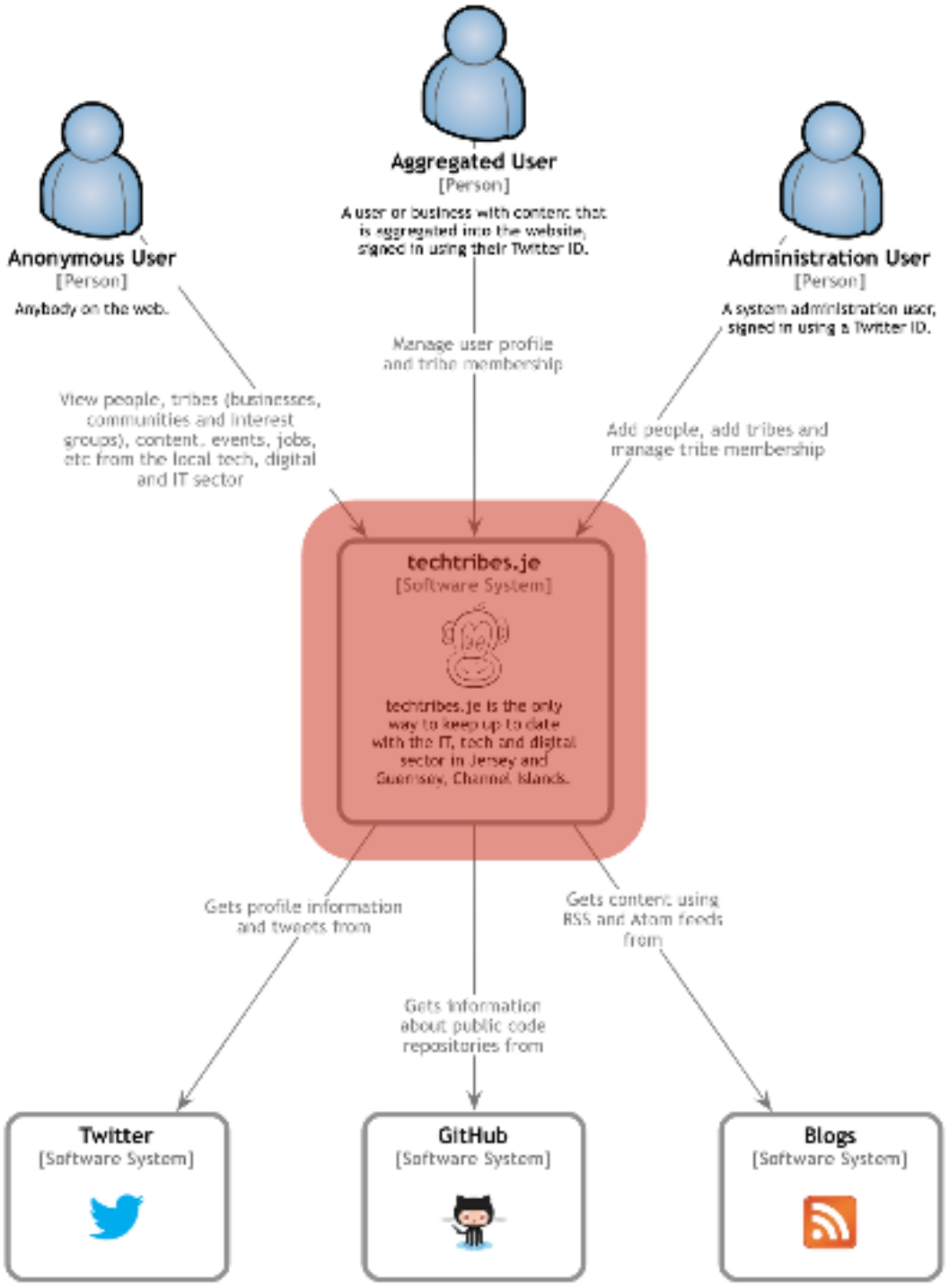
Component implementation details

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



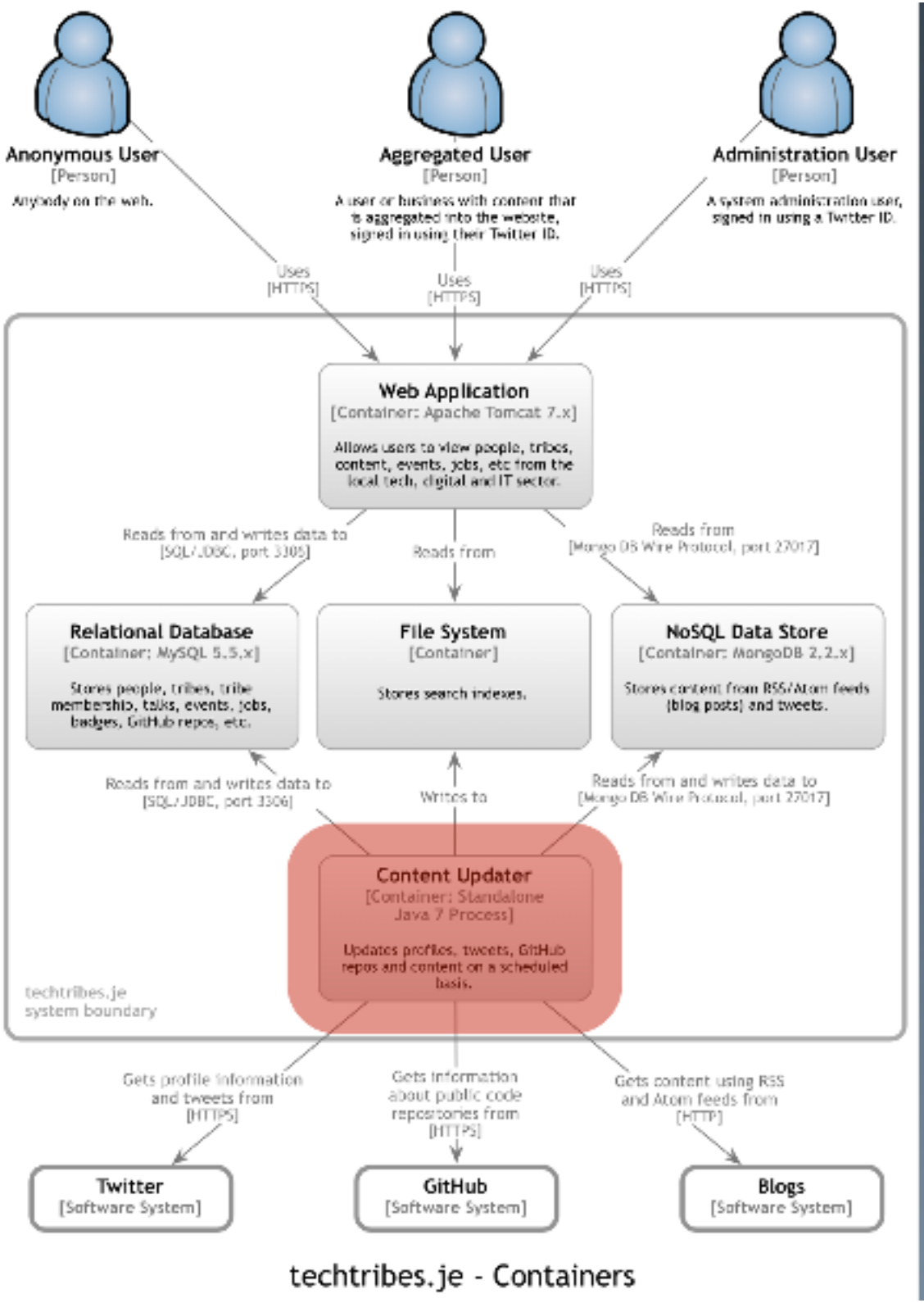
techtribes.je - Context

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)

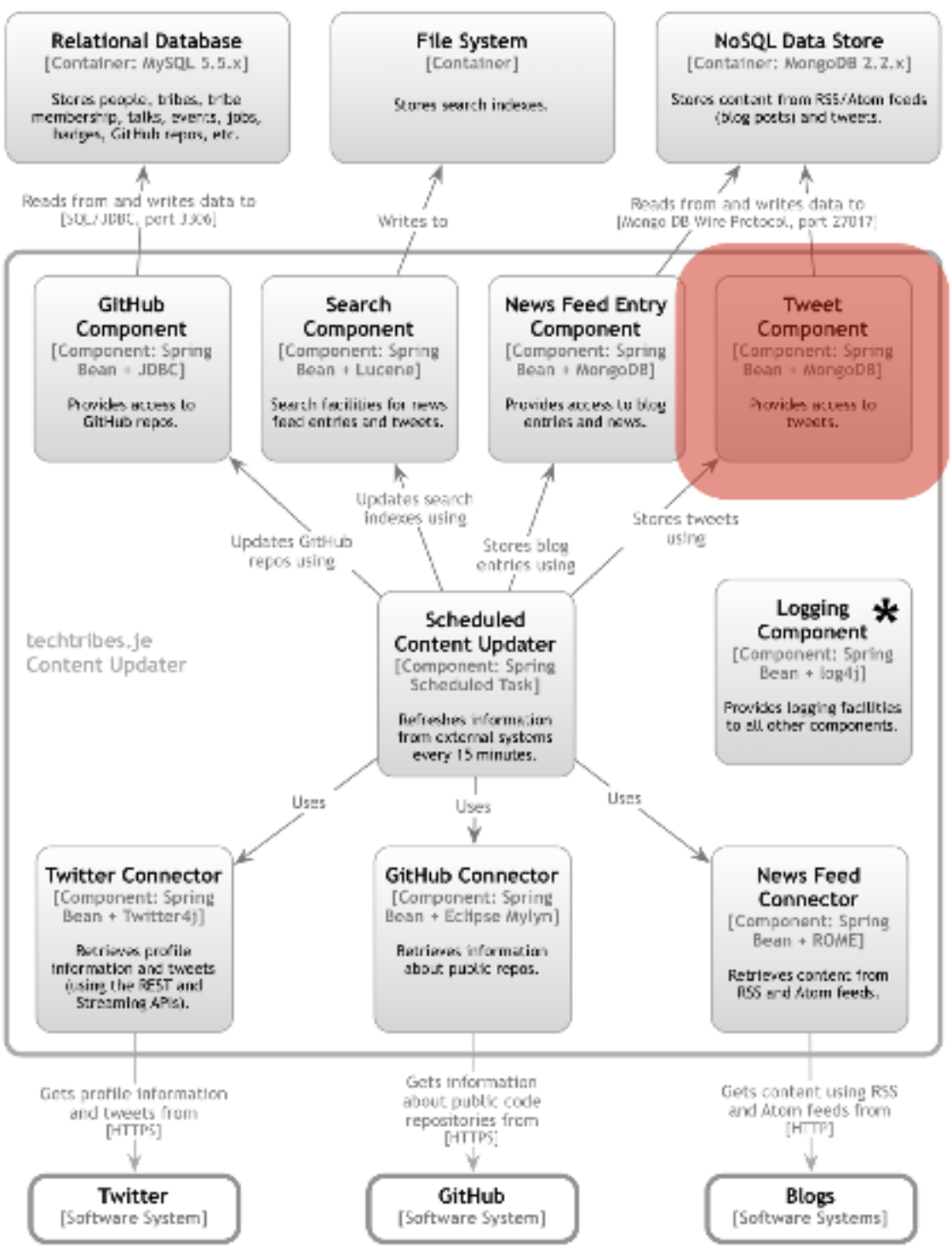


Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)

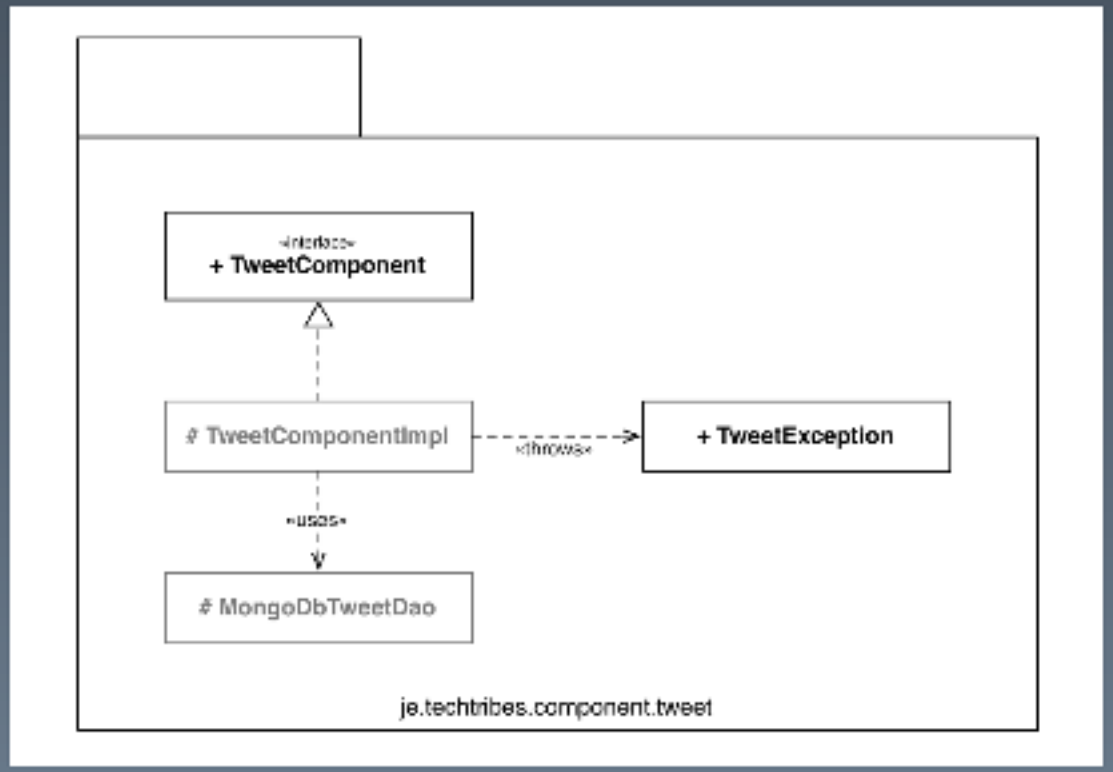


Context diagram
(level 1)

Container diagram
(level 2)

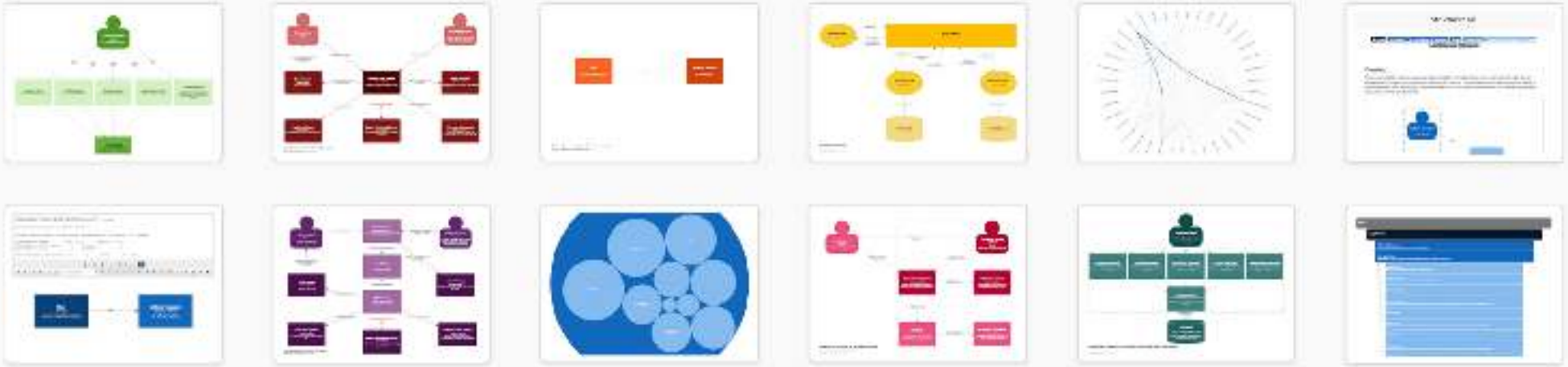
Component diagram
(level 3)

Class diagram
(level 4)



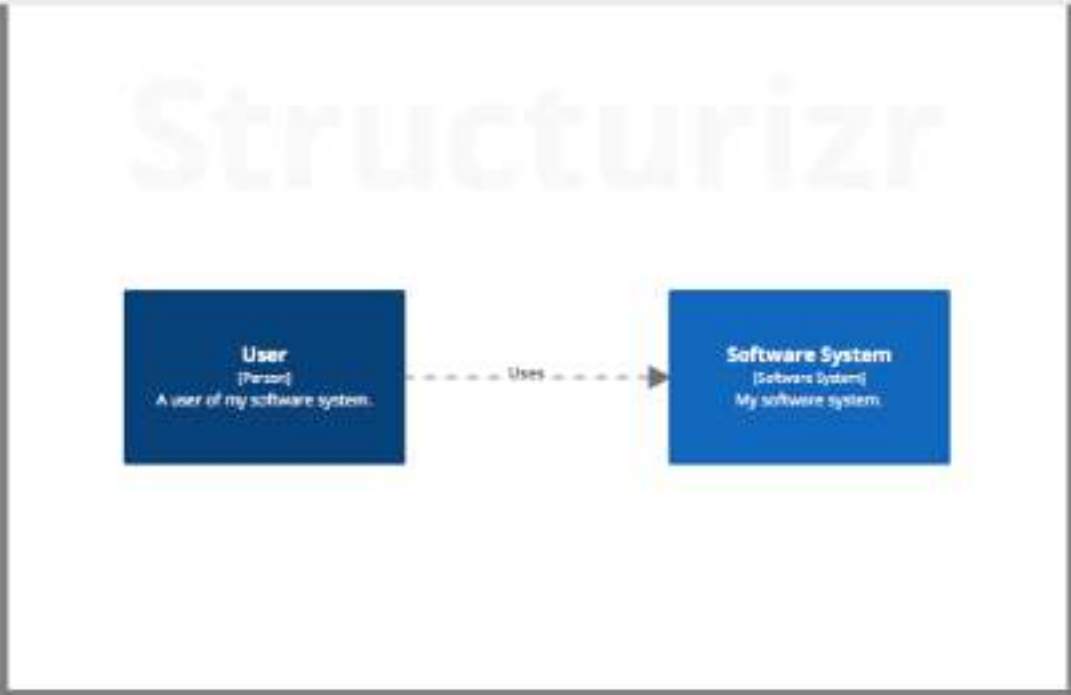
Structurizr

Visualise, document and explore your software architecture



```
Person = User | A user of my software system. | | 277,674
SoftwareSystem = Software System | My software system. | | 1552,674

Relationship = User | Uses | | Software System | |

Diagram = System Context | Software System | A description of this diagram. | A5_Landscape

ElementStyle = Element         |  650 |  400 |         | #ffffff |  36 |
ElementStyle = Software System |      |      | #1168bd |         |     |
ElementStyle = Person          |      |      | #08427b |         |     |
```

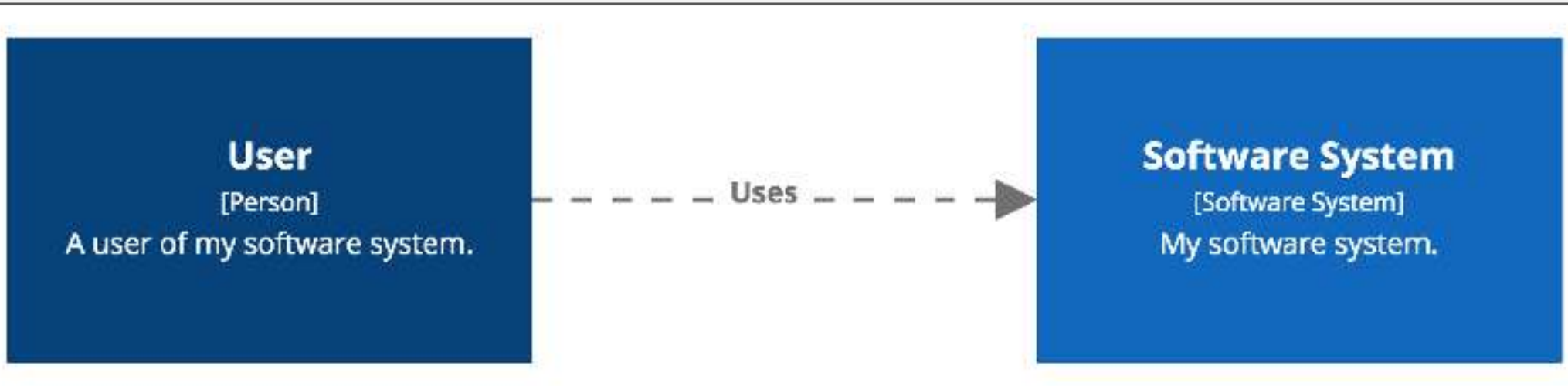
A5 - Landscape



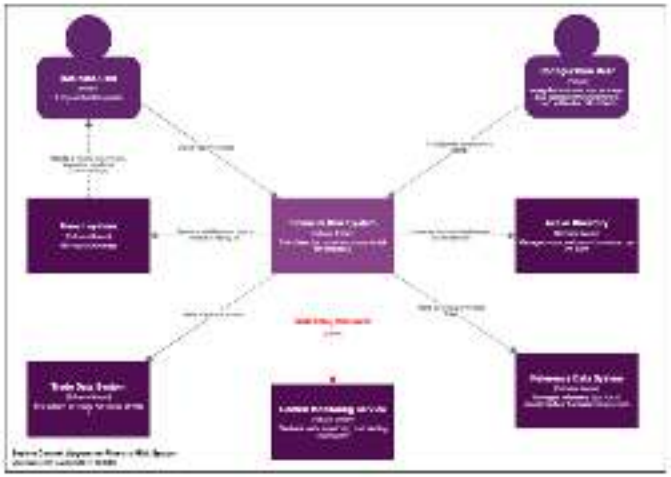
Structurizr for Java

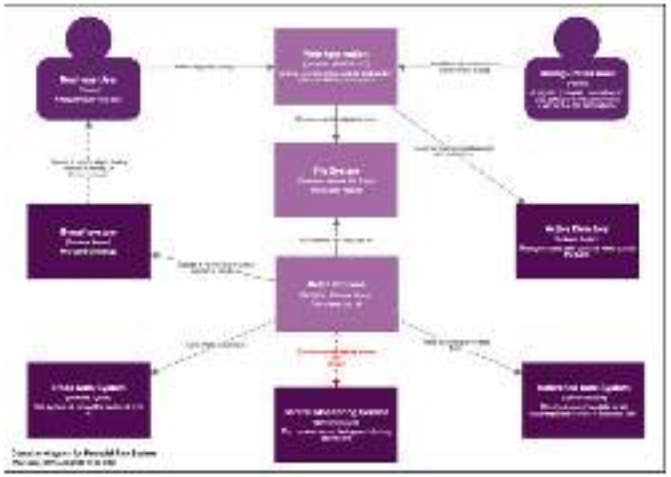
Structurizr for .NET

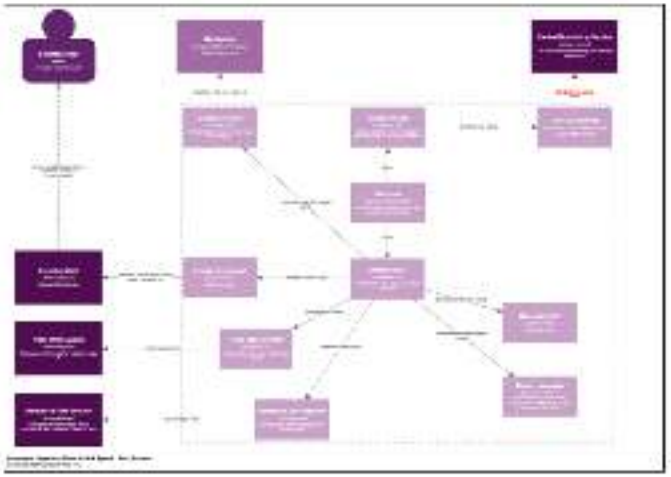
```
Workspace workspace = new Workspace("My model", "This is a model of my software system.");
Model model = workspace.getModel();

Person user = model.addPerson("User", "A user of my software system.");
SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System", "My software system.");
user.uses(softwareSystem, "Uses");

ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createSystemContextView(softwareSystem, "context", "A simple example...");
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

Styles styles = viewSet.getConfiguration().getStyles();
styles.addElementStyle(Tags.SOFTWARE_SYSTEM).background("#1168bd").color("#ffffff");
styles.addElementStyle(Tags.PERSON).background("#08427b").color("#ffffff");

StructurizrClient structurizrClient = new StructurizrClient("key", "secret");
structurizrClient.putWorkspace(1234, workspace);
```

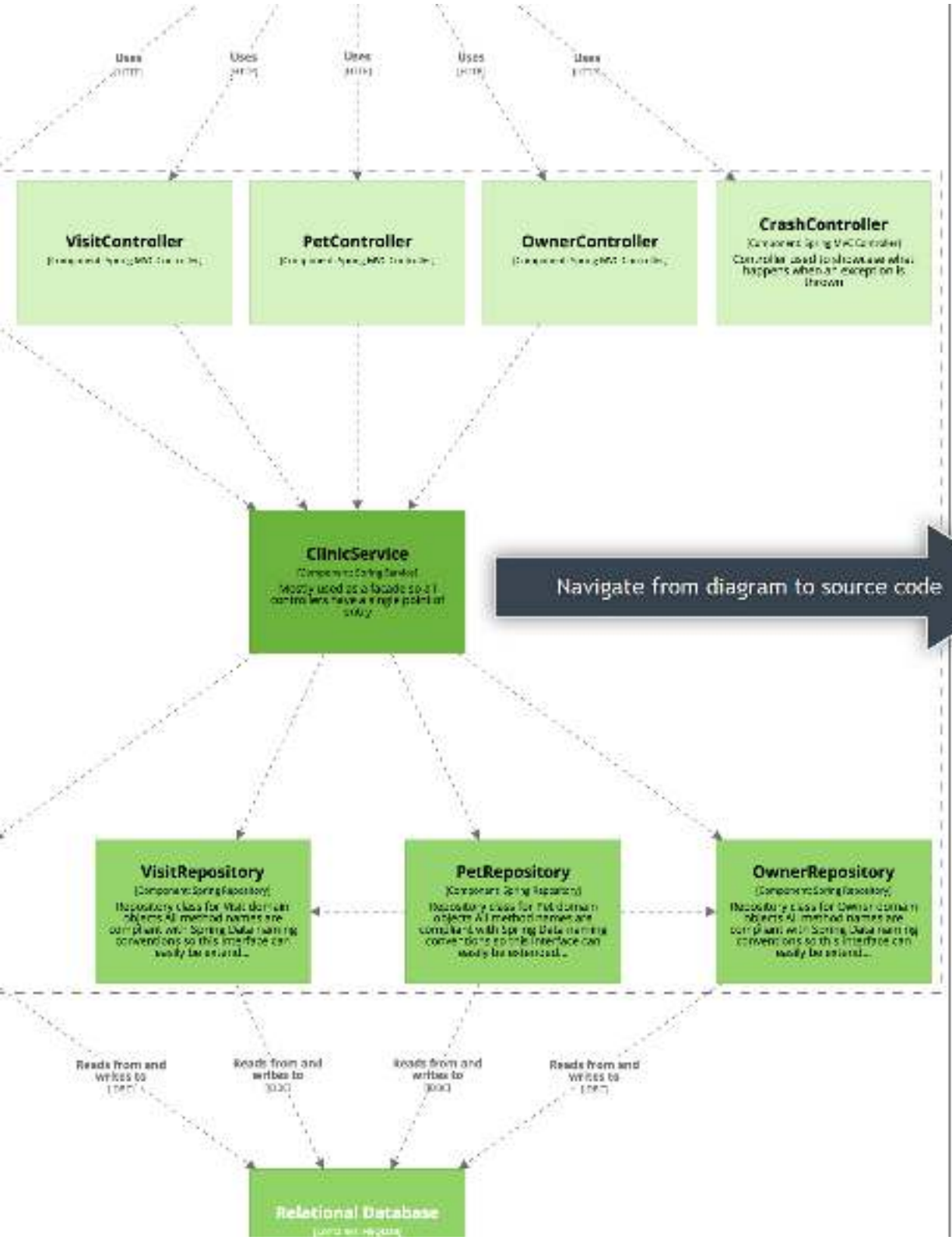


```
static void Main(string[] args)
{
    Workspace workspace = new Workspace("Financial Risk System", "A simple example C4 model based upon the financial risk system arc
    Model.Model model = workspace.Model;

    // create the basic model
    SoftwareSystem financialRiskSystem = model.AddSoftwareSystem(Location.Internal, "Financial Risk System", "Calculates the bank's

    Person businessUser = model.AddPerson(Location.Internal, "Business User", "A regular business user");
    businessUser.Uses(financialRiskSystem, "Views reports using");

    Person configurationUser = model.AddPerson(Location.Internal, "Configuration User", "A regular business user who can also config
    configurationUser.Uses(financialRiskSystem, "Configures parameters using");

    SoftwareSystem tradeDataSystem = model.AddSoftwareSystem(Location.Internal, "Trade Data System", "The system of record for trade
    financialRiskSystem.Uses(tradeDataSystem, "Gets trade data from");

    SoftwareSystem referenceDataSystem = model.AddSoftwareSystem(Location.Internal, "Reference Data System", "Manages reference data
    financialRiskSystem.Uses(referenceDataSystem, "Gets counterparty data from");

    SoftwareSystem emailSystem = model.AddSoftwareSystem(Location.Internal, "E-mail system", "Microsoft Exchange");
    financialRiskSystem.Uses(emailSystem, "Sends a notification that a report is ready to");
    emailSystem.Delivers(businessUser, "Sends a notification that a report is ready to", "E-mail message", InteractionStyle.Asynchro

    SoftwareSystem centralMonitoringService = model.AddSoftwareSystem(Location.Internal, "Central Monitoring Service", "The bank-wid
    financialRiskSystem.Uses(centralMonitoringService, "Sends critical failure alerts to", "SNMP", InteractionStyle.Asynchronous).Ad

    SoftwareSystem activeDirectory = model.AddSoftwareSystem(Location.Internal, "Active Directory", "Manages users and security role
```

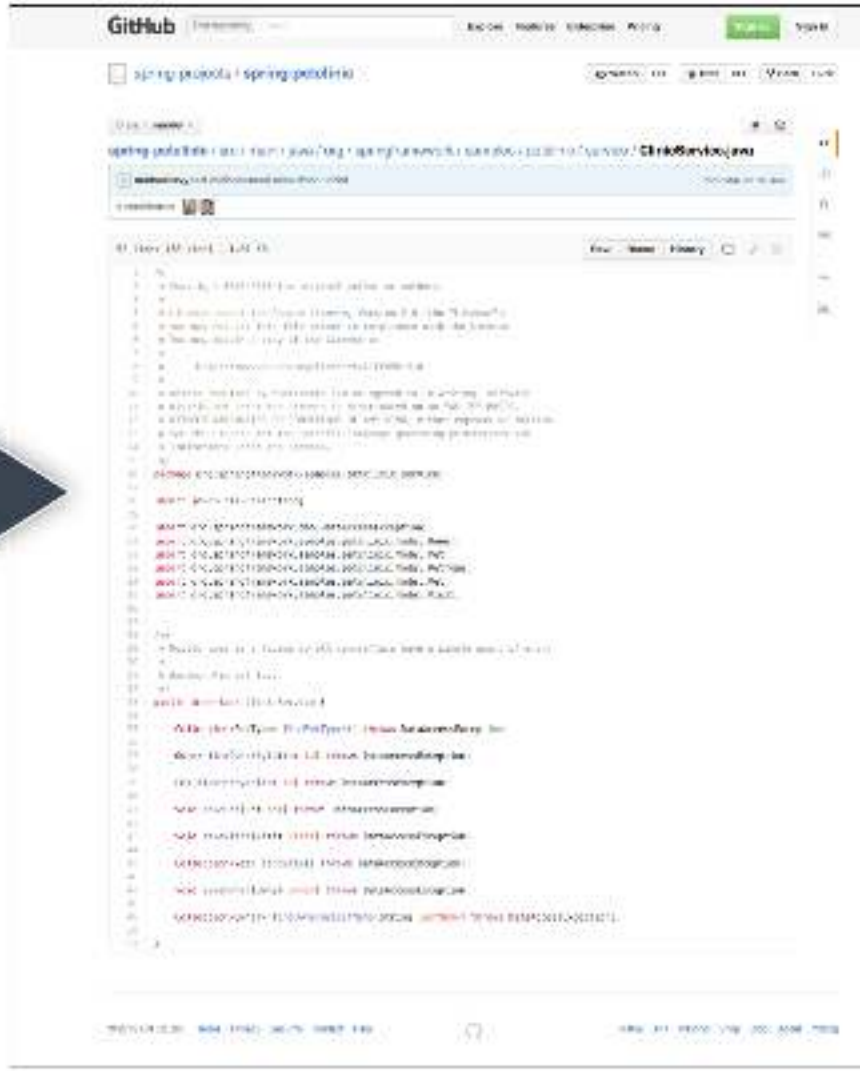






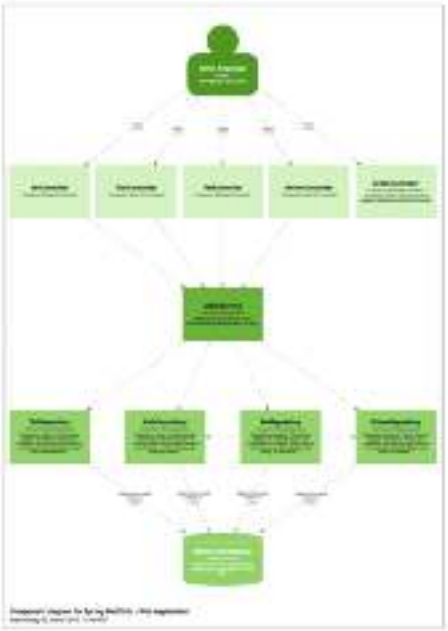


Navigate from diagram to source code



System Context diagram

Container diagram



Component diagram

Source code

Double-click a software system

Double-click a container

Double-click a component

Diagrams are maps

Architecture Decision Records

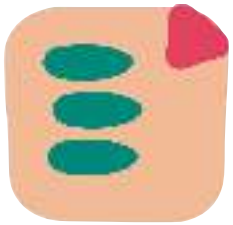
We will keep a collection of records for "architecturally significant" decisions: those that affect the structure, non-functional characteristics, dependencies, interfaces, or construction techniques.

http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions

Architecture Decision Records

Textile

asciidoc



short text file

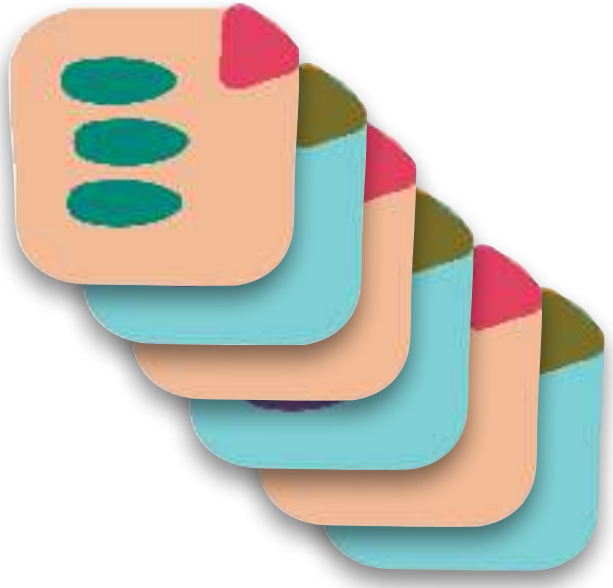
doc/arch/adr-NNN.md

semantic versioning

numbered sequentially

no number reused

reversed ADR kept—
marked superseded



Architecture Decision Records

Title: short noun phrase

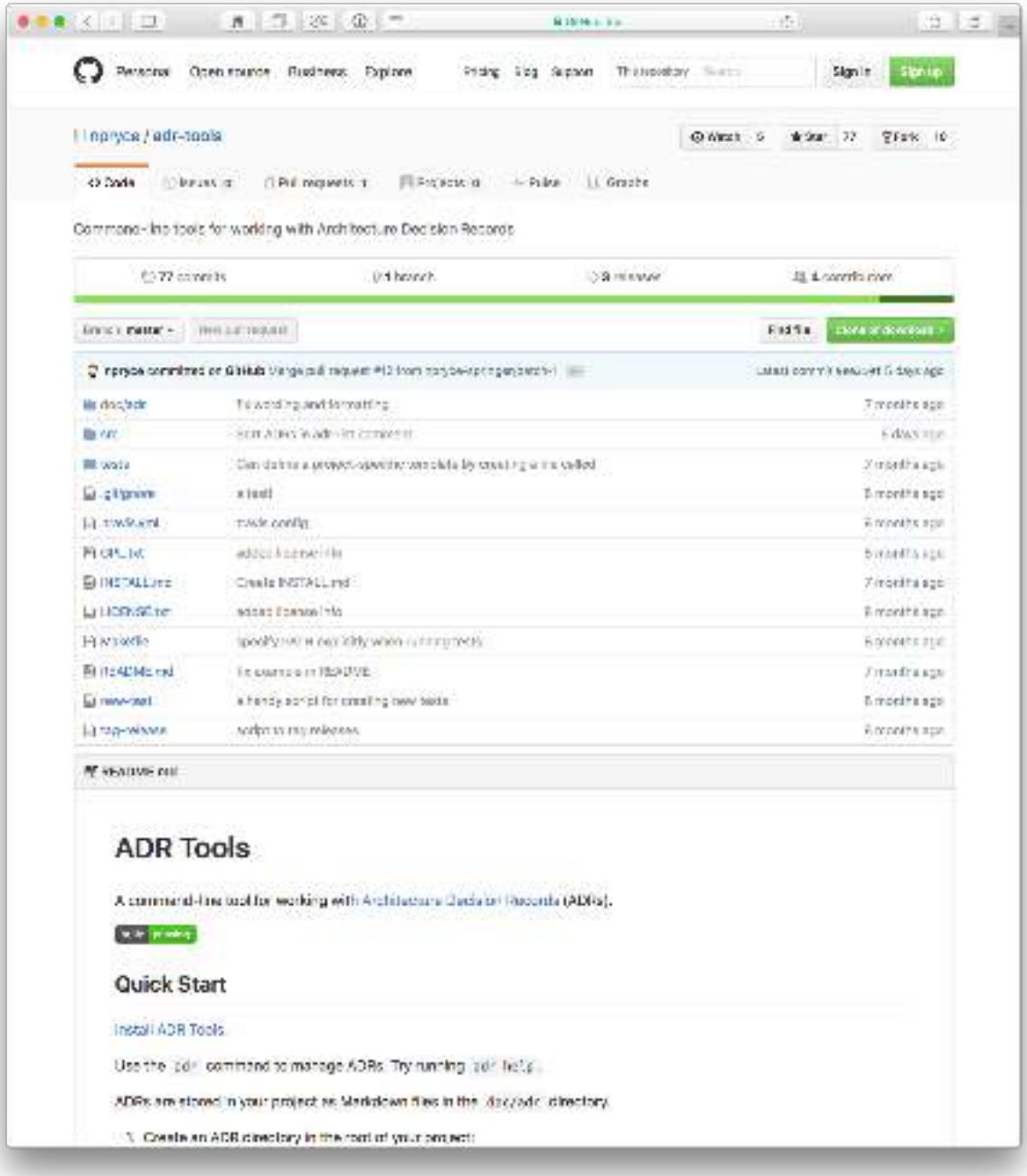
Context: forces at play

Decision: response to forces

Status: `proposed`, `accepted`, `superseded`

Consequences: context after decision is applied

ADR Tool

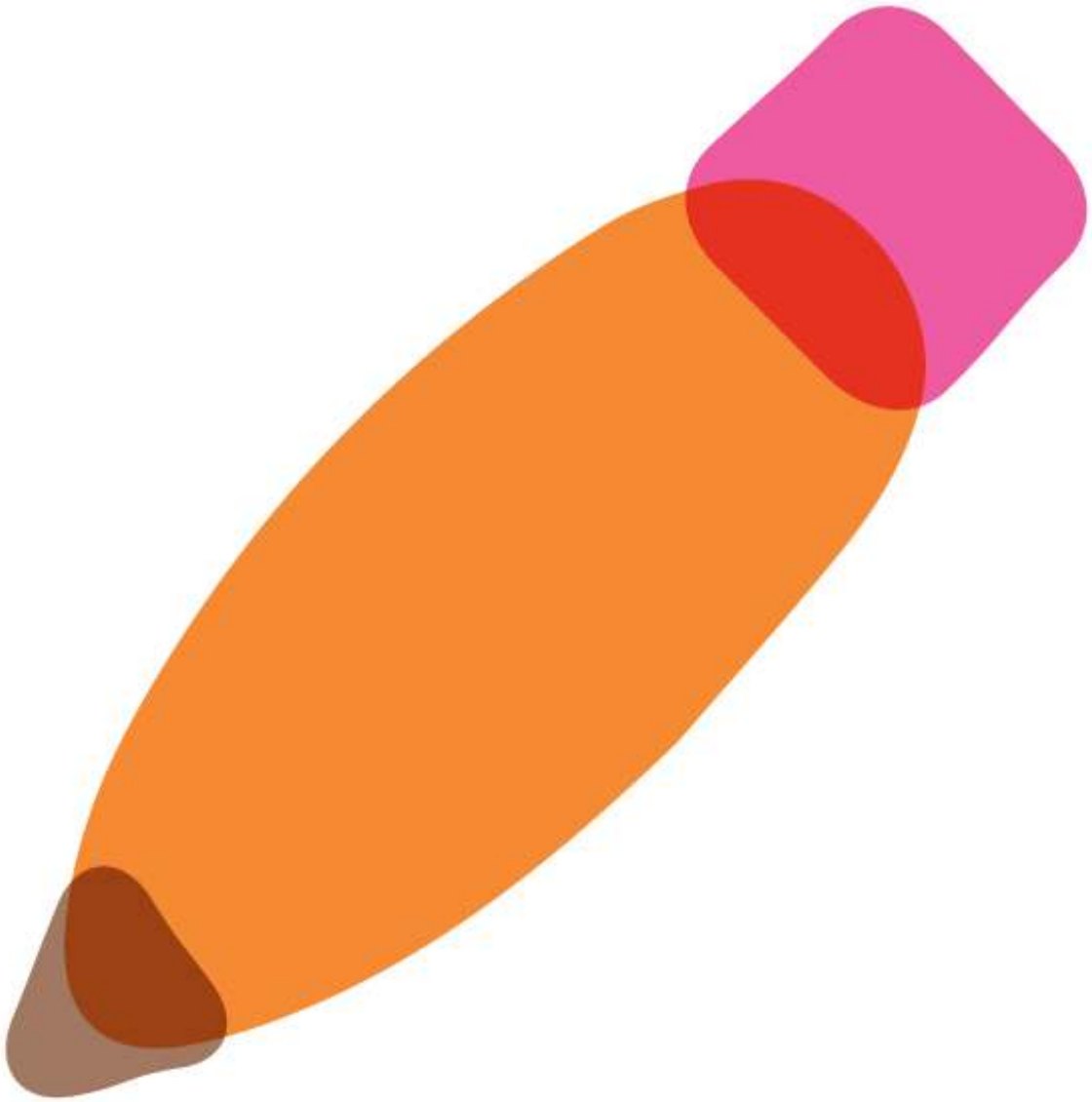


https://github.com/npryce/adr-tools

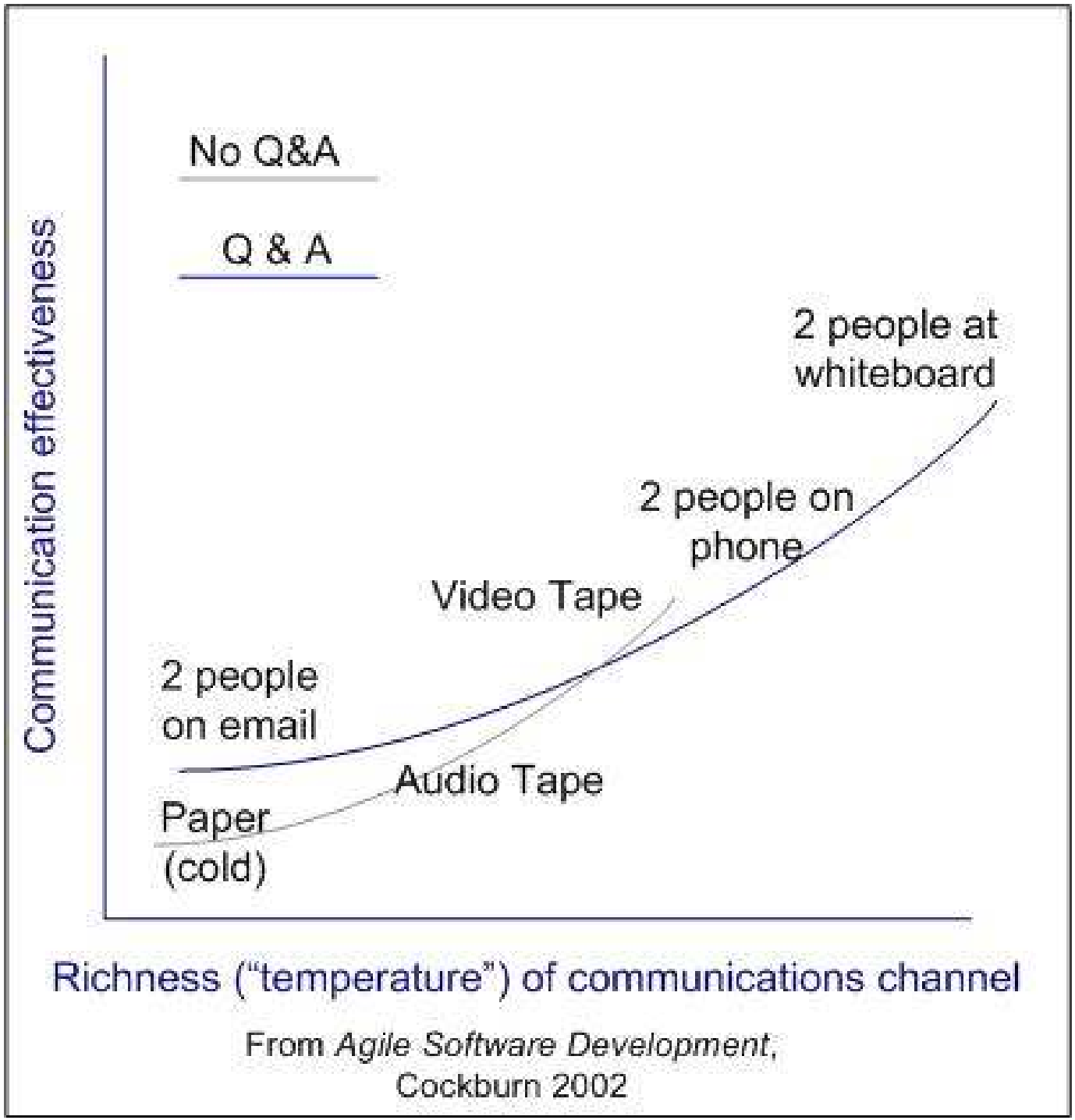
architecture katas

documenting your architecture

Technical Writing Skills

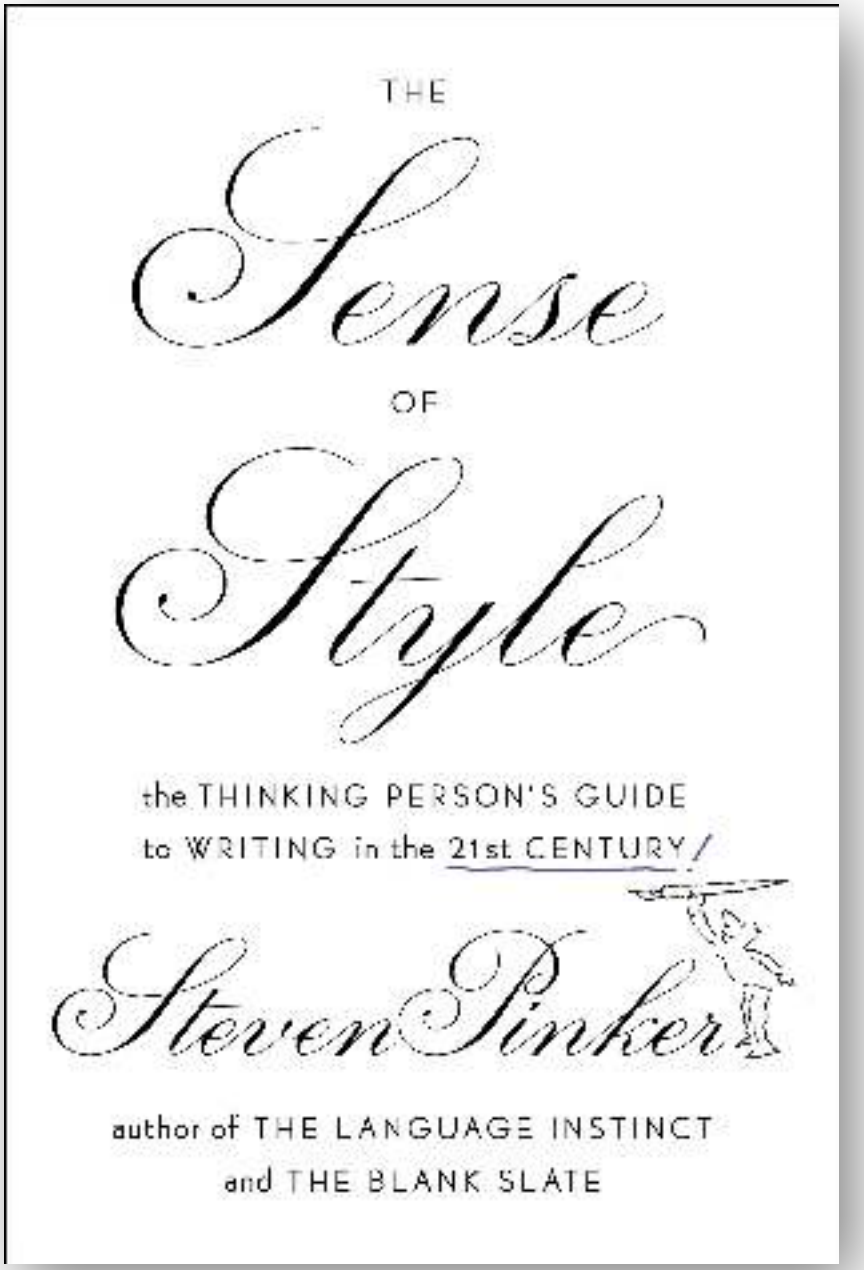


Software is more about communication than technology.



From *Agile Software Development*, Cockburn 2002

Know Your Audience



practical
vs.
classic style

passive voice

Passive voice occurs when you make the object of an action into the subject of a sentence.

Why was the road crossed by the chicken?

examples

The metropolis has been scorched by the dragon's fiery breath.
The dragon scorched the metropolis with his fiery breath.

When her house was invaded, Penelope had to think of ways to delay her remarriage.
After suitors invaded her house, Penelope had to think of ways to delay her remarriage.

more examples

Heart disease is considered the leading cause of death in the United States.

Research points to heart disease as the leading cause of death in the United States.

Researchers have concluded that heart disease is the leading cause of death in the United States.

The balloon is positioned in an area of blockage and is inflated.

The surgeon positions the balloon in an area of blockage and inflates it.

passive voice myths

1. Use of the passive voice constitutes a grammatical error.

2. Any use of "to be" (in any form) constitutes the passive voice.

3. The passive voice always avoids the first person.

4. You should never use the passive voice.

5. I can rely on my grammar checker to catch the passive voice.

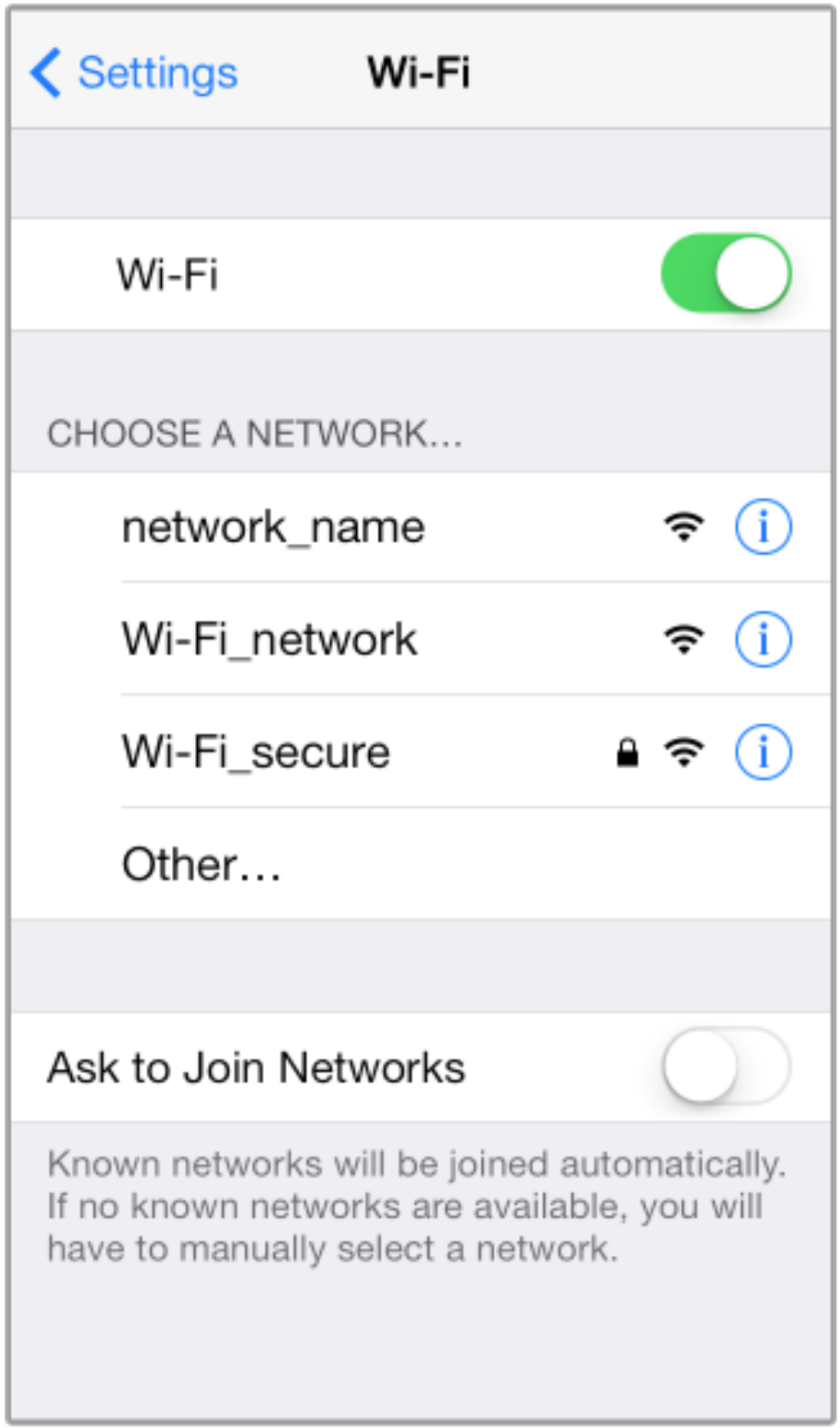
“swindles & perversions”

Mistakes were made.

The Exxon Company accepts that a few gallons might have been spilled.

use of language shapes clarity and meaning

some people use the passive voice to avoid mentioning responsibility for certain actions



it's common

Your phone will join known networks automatically.

Your phone automatically joins known networks.

If no known networks are available, you must manually select a network.

the most important rule:

revise!

revise!

all important documentation

proposals

emails !

all written correspondence

technical writing

simple, declarative sentences

draft & rewrite

…and rewrite and rewrite and rewrite…

spell check!

have someone else read it for clarity

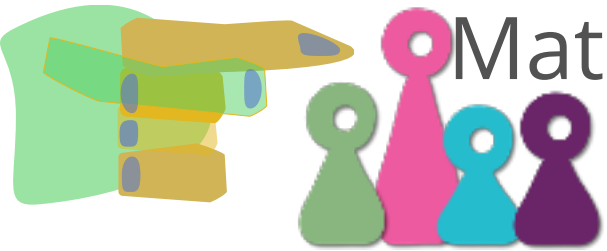
Continuous Delivery

continuous delivery ∩ architect

Architecture is abstract
until operationalized

Understanding
shifting structure.

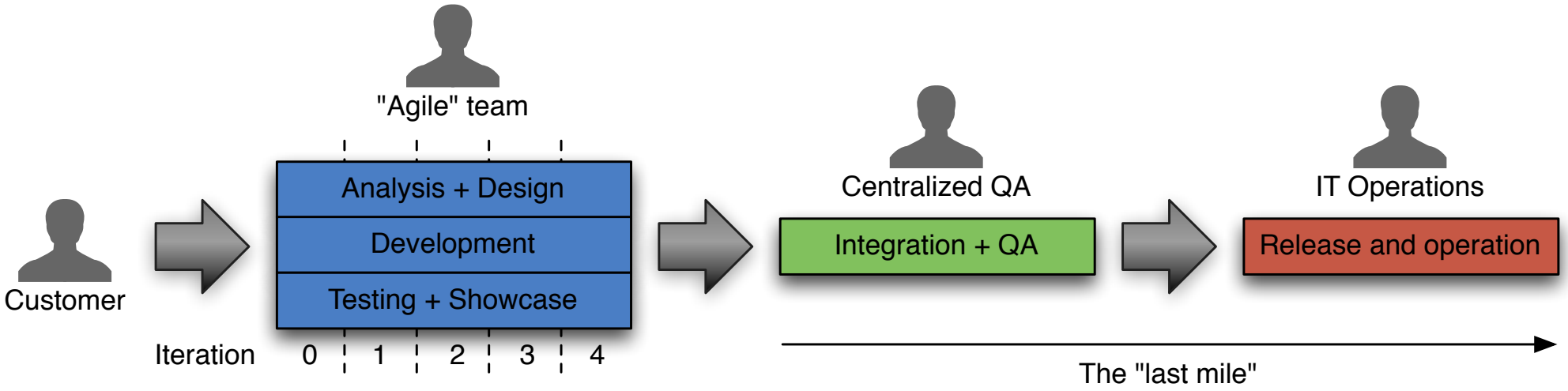
Expanding role
of architect.



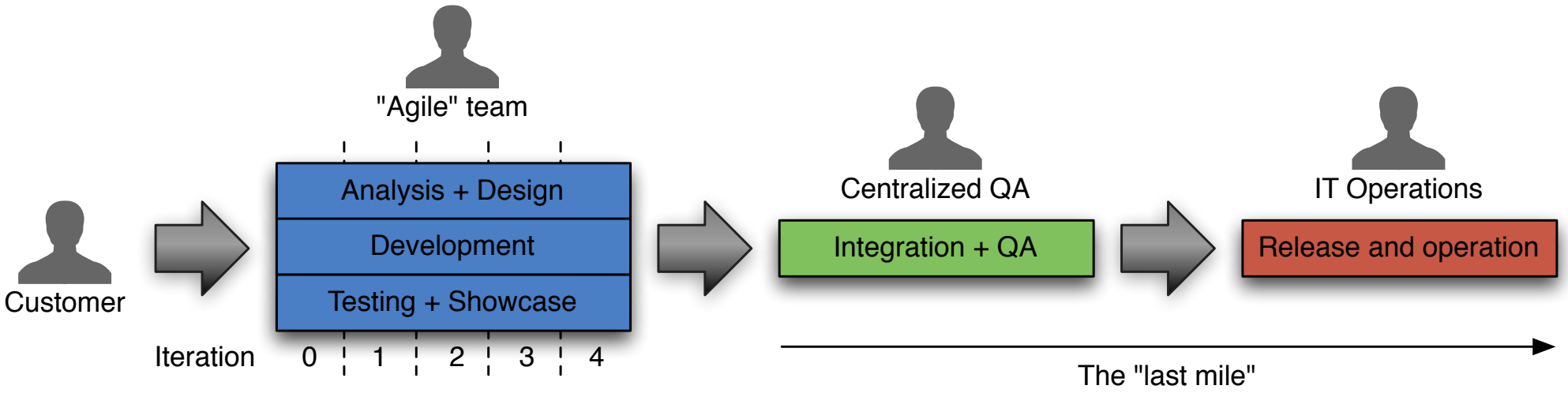
Mature engineering
practices.

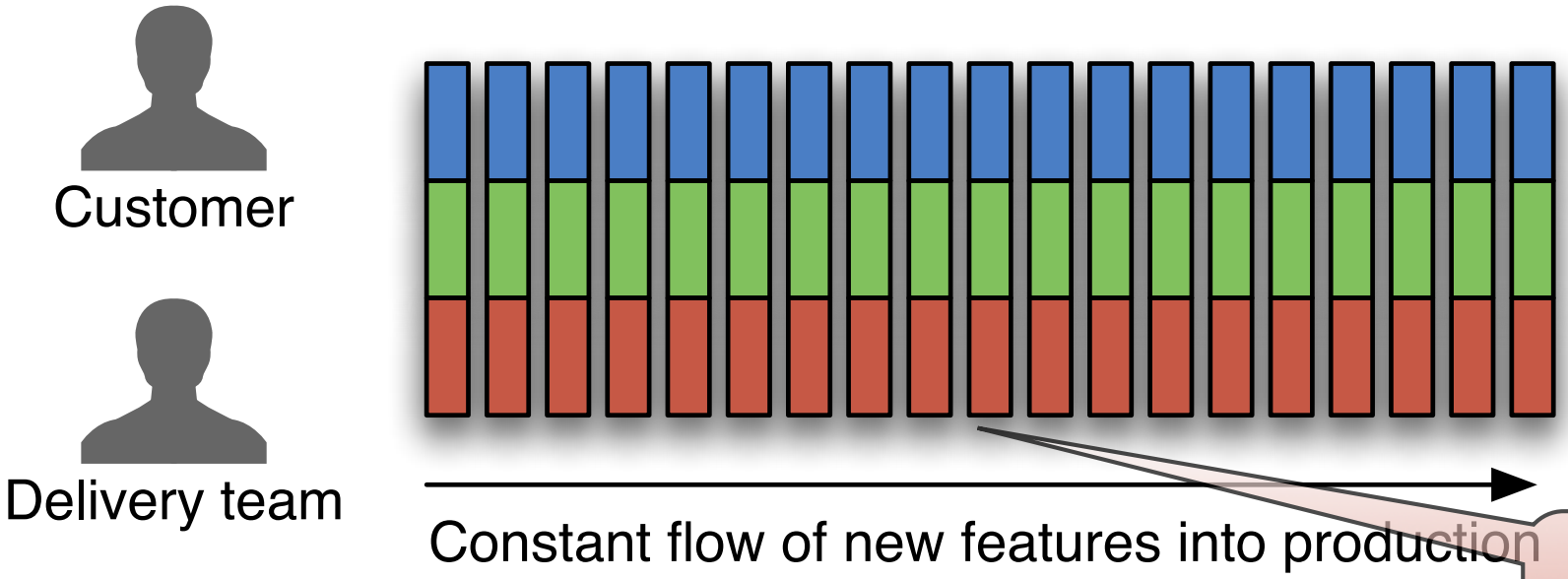
Manage
coupling intelligently.

agile 101


"Agile" team
Analysis + Design
Development
Testing + Showcase
Customer
Iteration
0
1
2
3
4
Centralized QA
Integration + QA
IT Operations
Release and operation
The "last mile"

continuous delivery



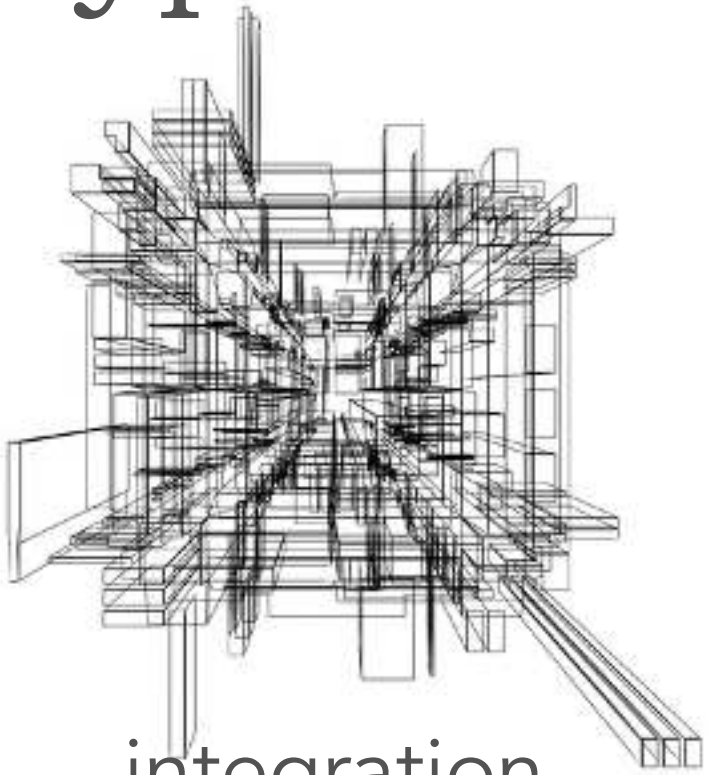


business needs > operational constraints

architecture types

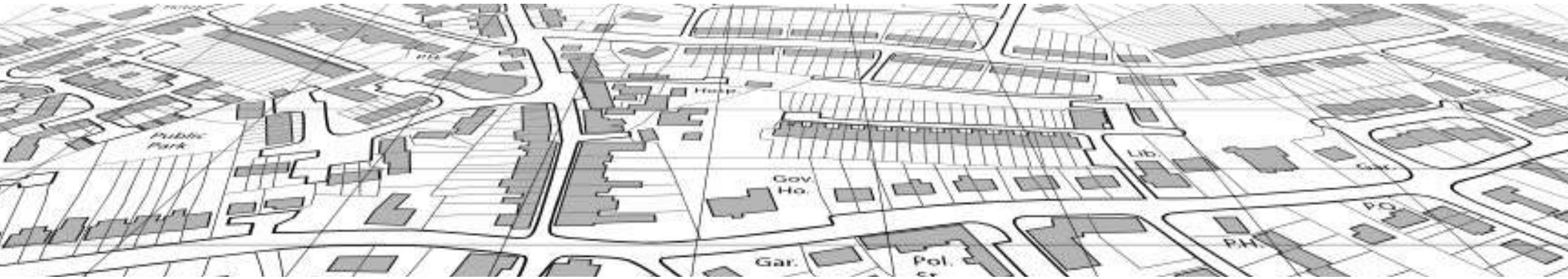
architecture types

application

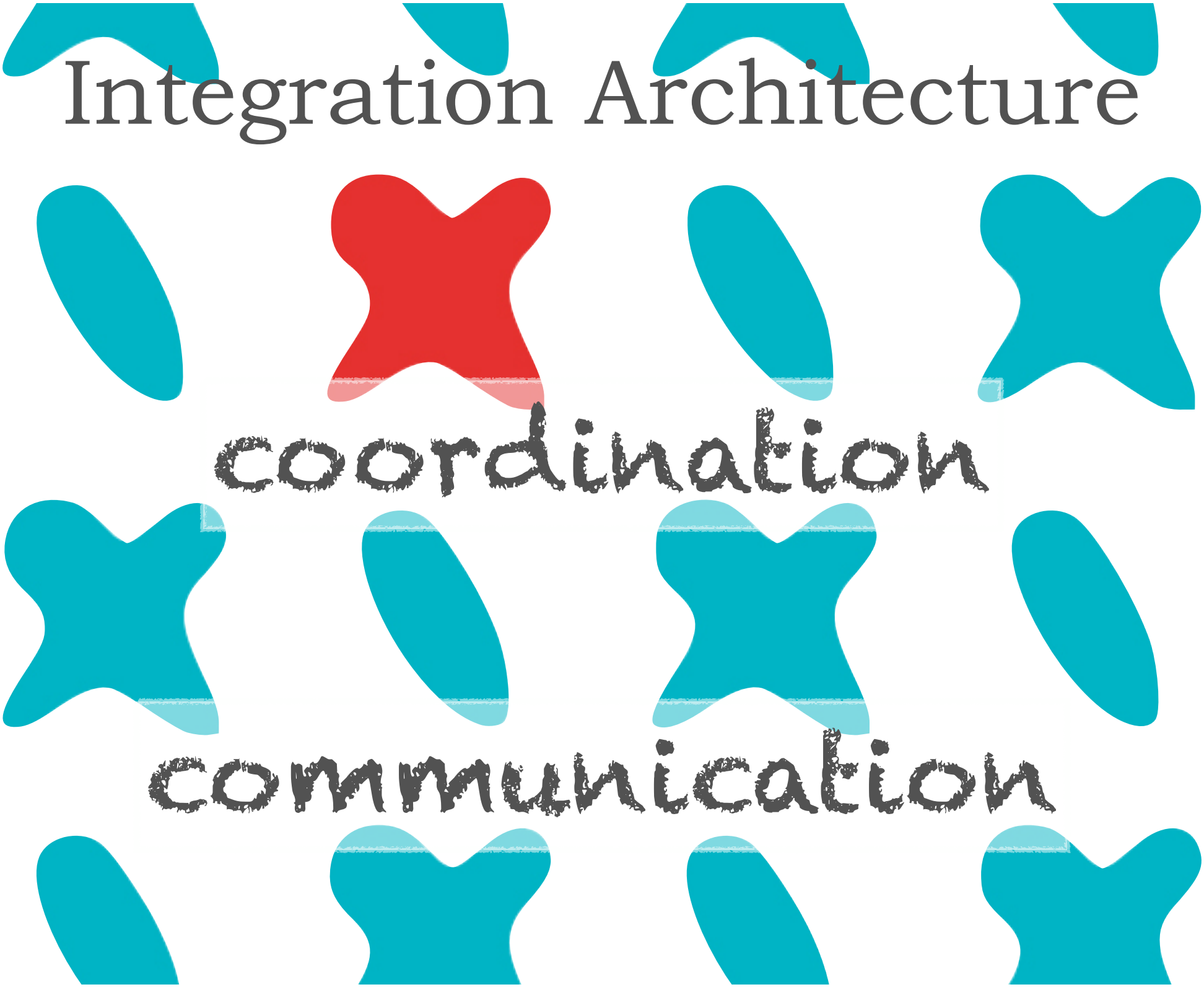


integration

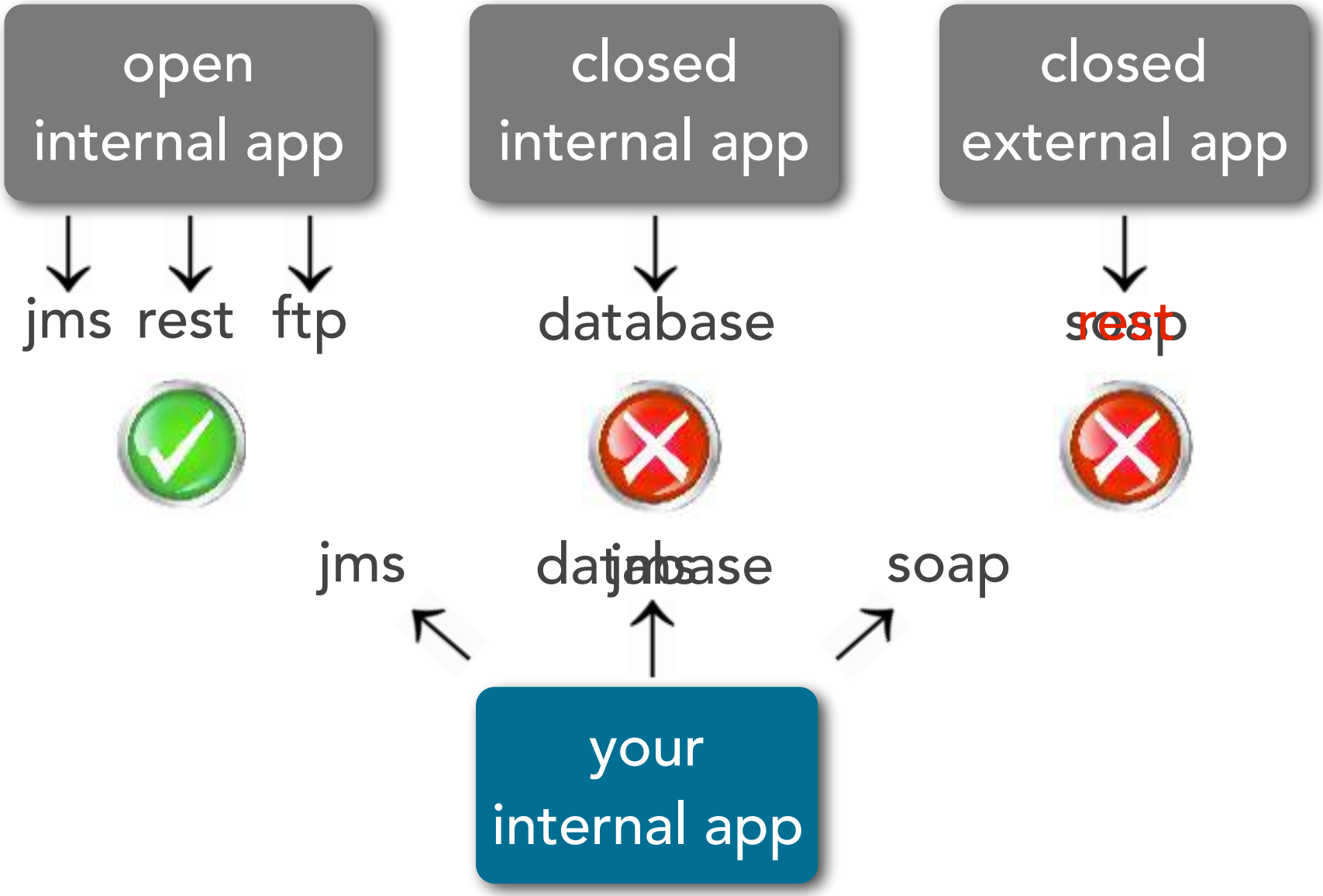
enterprise

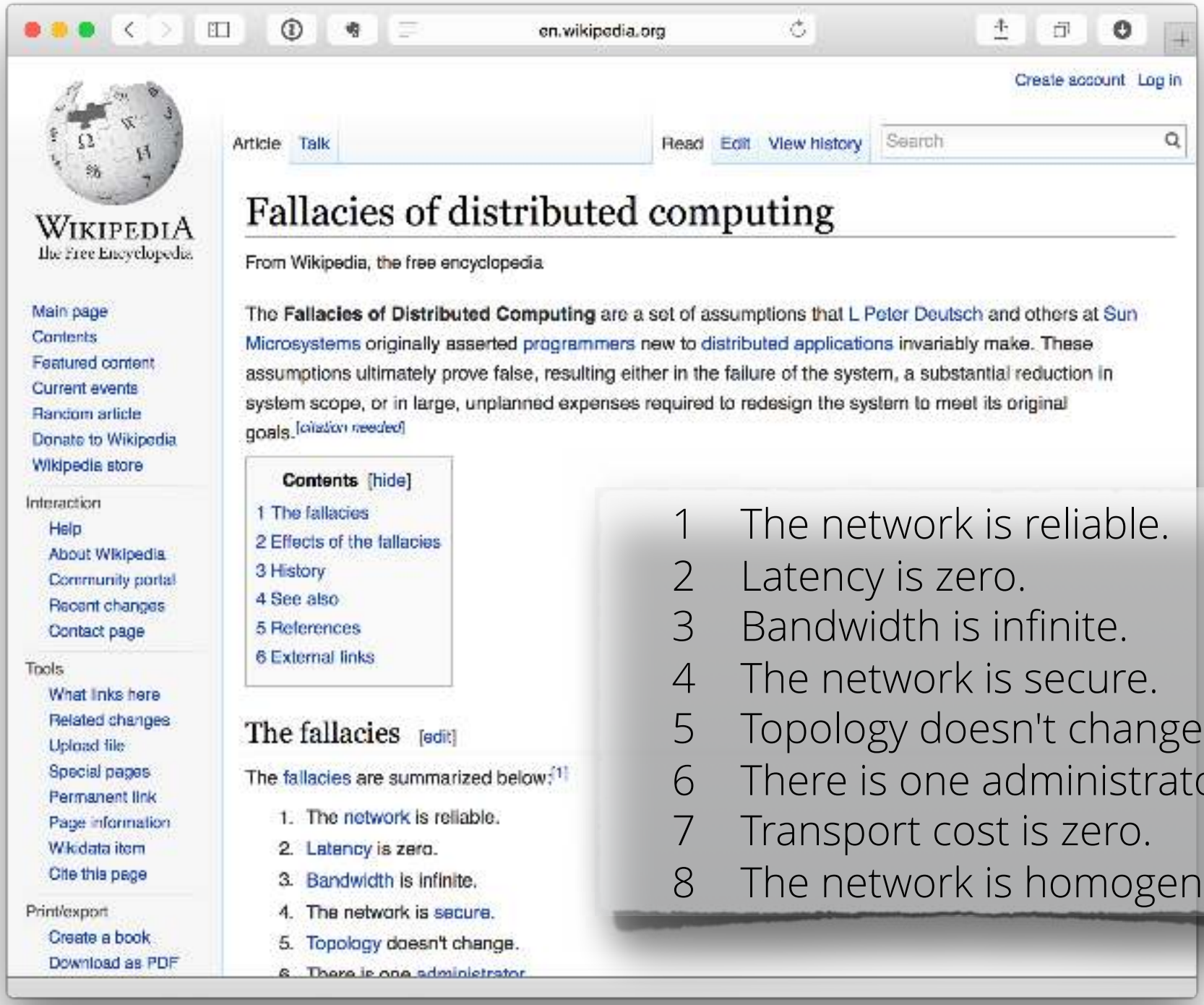


Integration Architecture



challenges

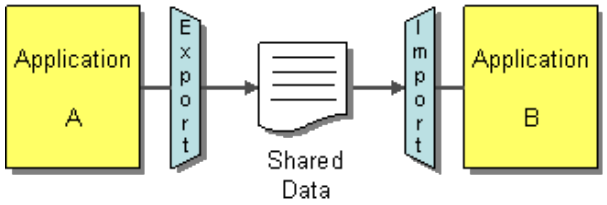

open
internal app
closed
internal app
closed
external app
jms
rest
ftp
database
soap
jms
database
soap
your
internal app



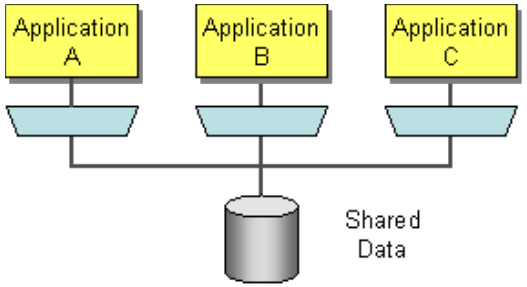
en.wikipedia.org/wiki/Fallacies_of_distributed_computing

integration styles

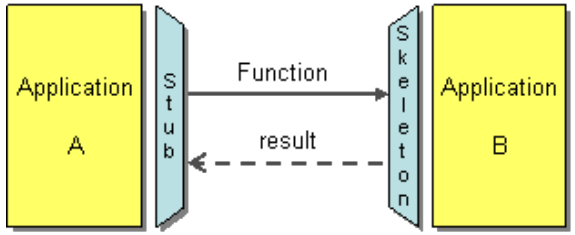
From Enterprise Integration Patterns by Hohpe and Woolf



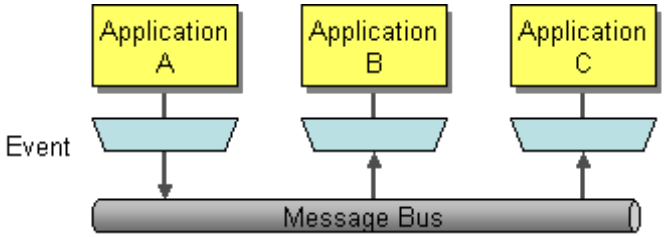
file transfer



shared database

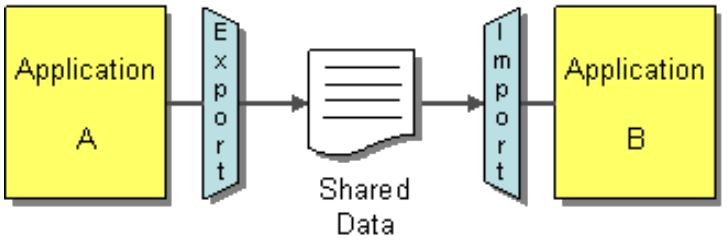


remote procedure invocation



messaging

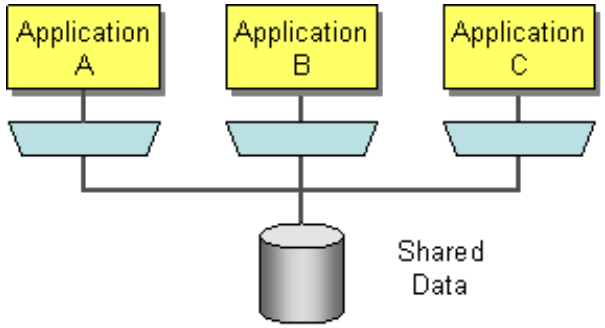
file transfer



universal integration style, integration simplicity, system decoupling and system abstraction

file-based processing is expensive, error processing, timeliness of data synchronization, data-only transfer

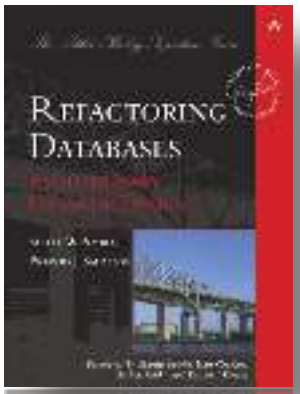
shared database

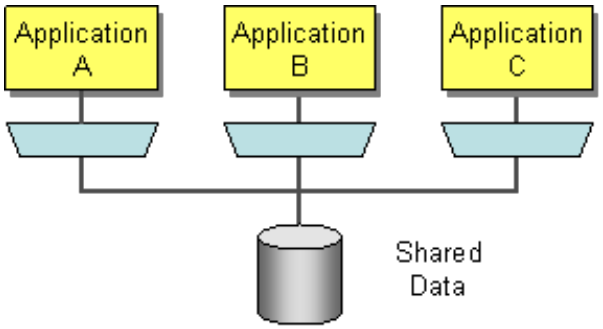


near-universal integration via SQL, system abstraction, system decoupling

cannot use persistence caching (ORM), performance bottleneck issues, schema change issues, data ownership issues

expand/contract pattern





app v205 compatible with db v13 and v14

app v230 compatible with db v14

app v234 compatible with db v14

app v241 compatible with db v14 and 15

app v248 compatible with db v15

DB version 13

DB version 14

DB version 15

app v205 deployed

migrate db to v14

app v230 deployed

app v234 deployed

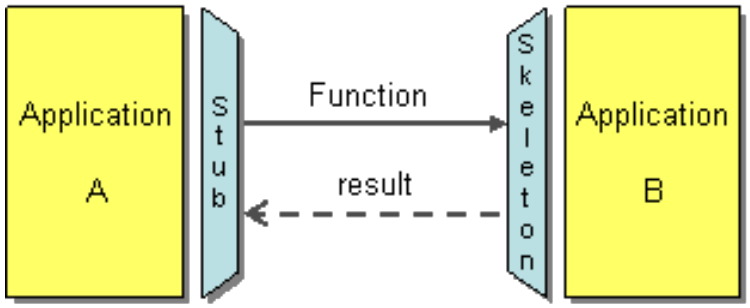
app v234 deployed

migrate db to v15

app v248 deployed

Time

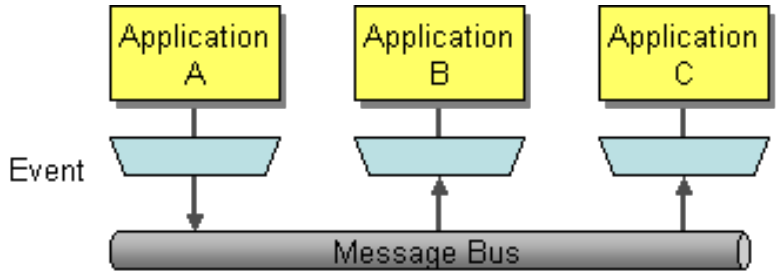
remote procedure



data encapsulation and ownership, external systems integration via web services, mature frameworks and tools

tight system coupling due to dependency on service availability and location knowledge, poor asynchronous communications

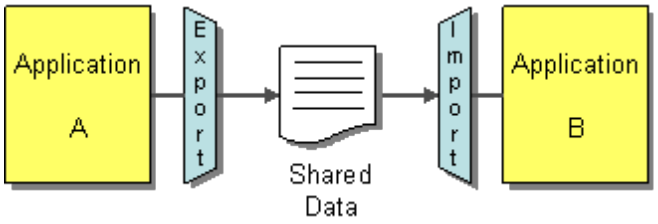
messaging



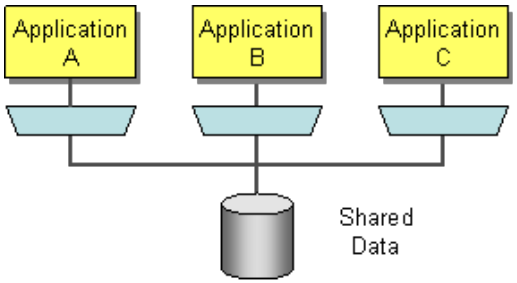
asynchronous and reliable messaging, highly decoupled systems, excellent scalability capabilities, monitoring

external integration beyond firewall, implementation and testing complexity, cross platform standards still evolving

which is the best integration style?

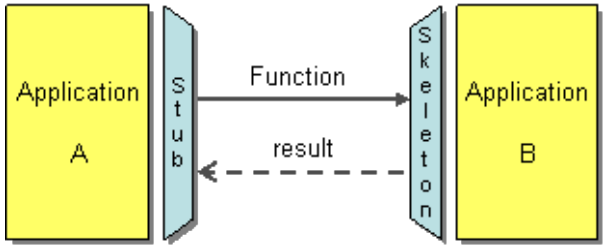


file transfer

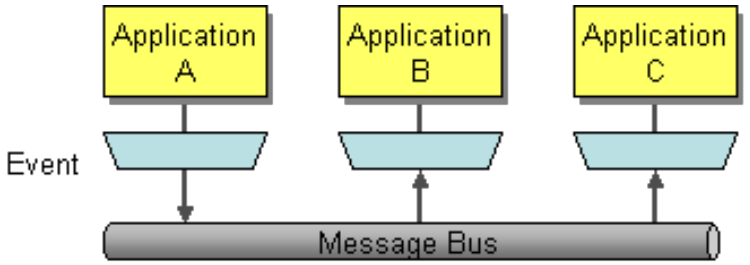


shared database

?



remote procedure invocation



messaging

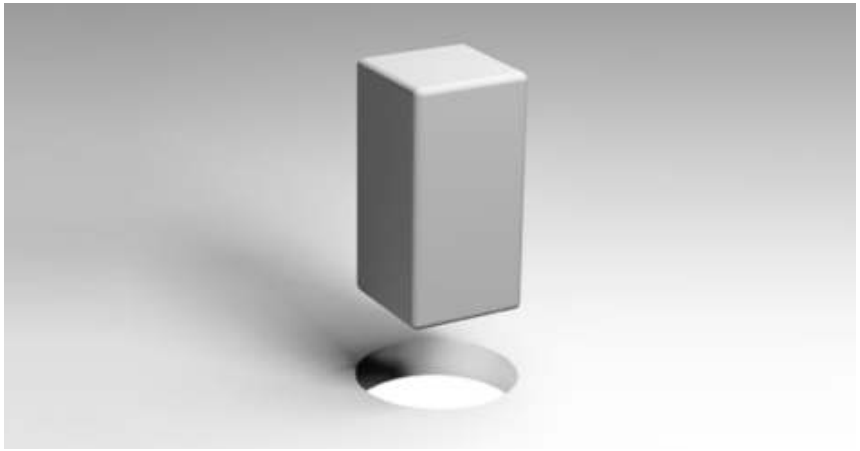
Enterprise Architecture

Enterprise Architecture

enterprise architecture context

Business Strategy and Operating Model

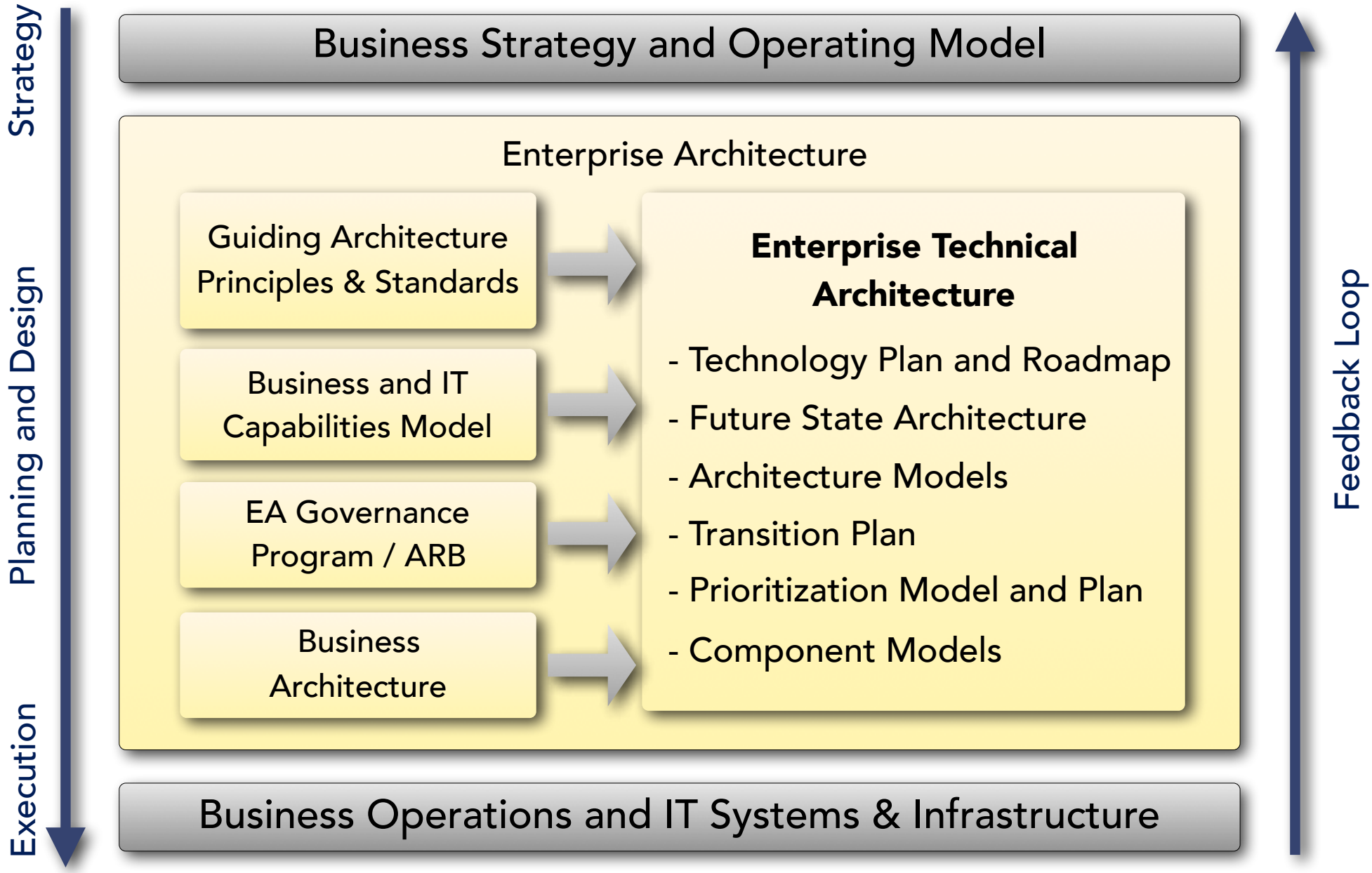
Business Needs



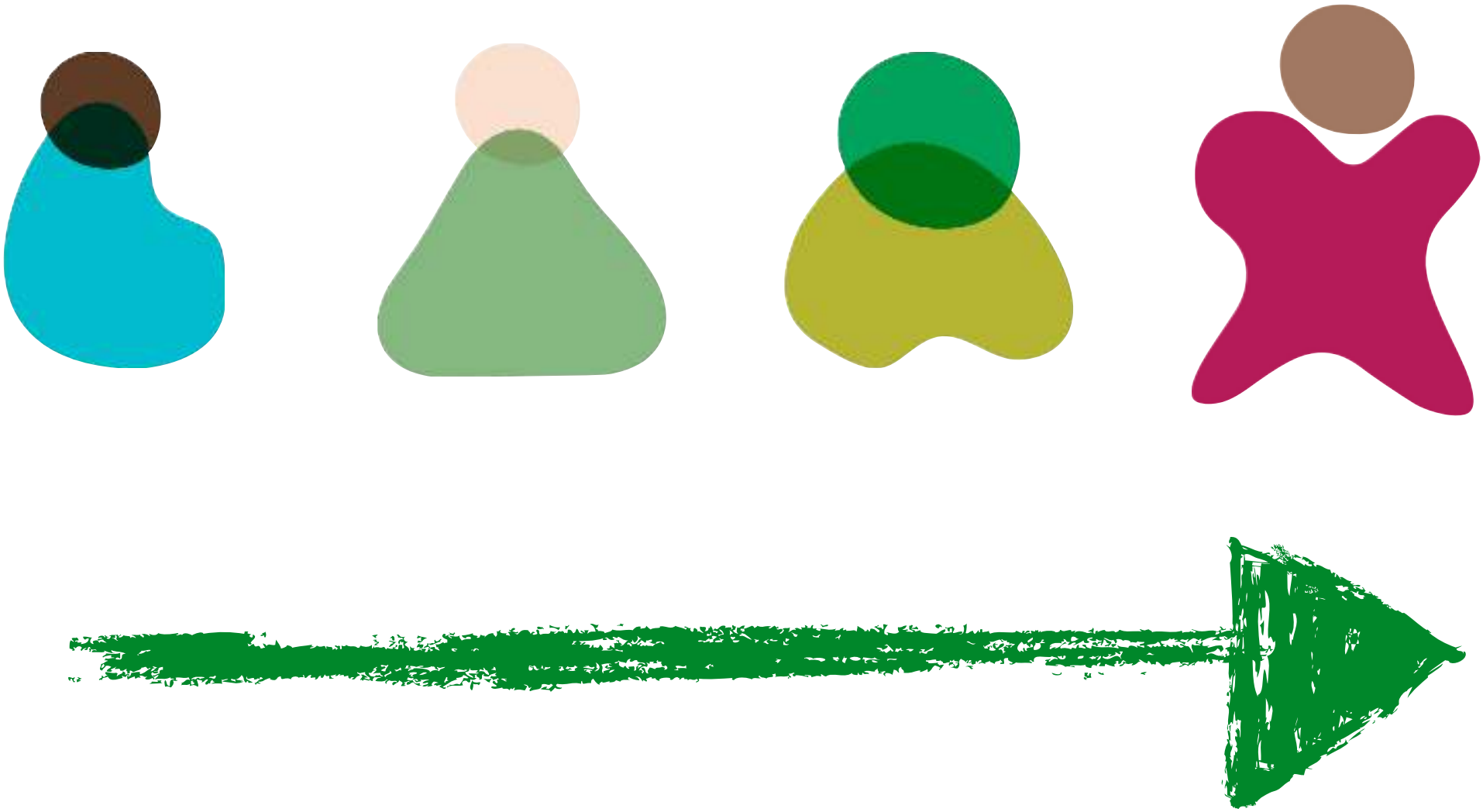
IT Capabilities

Business Operations and IT Systems & Infrastructure

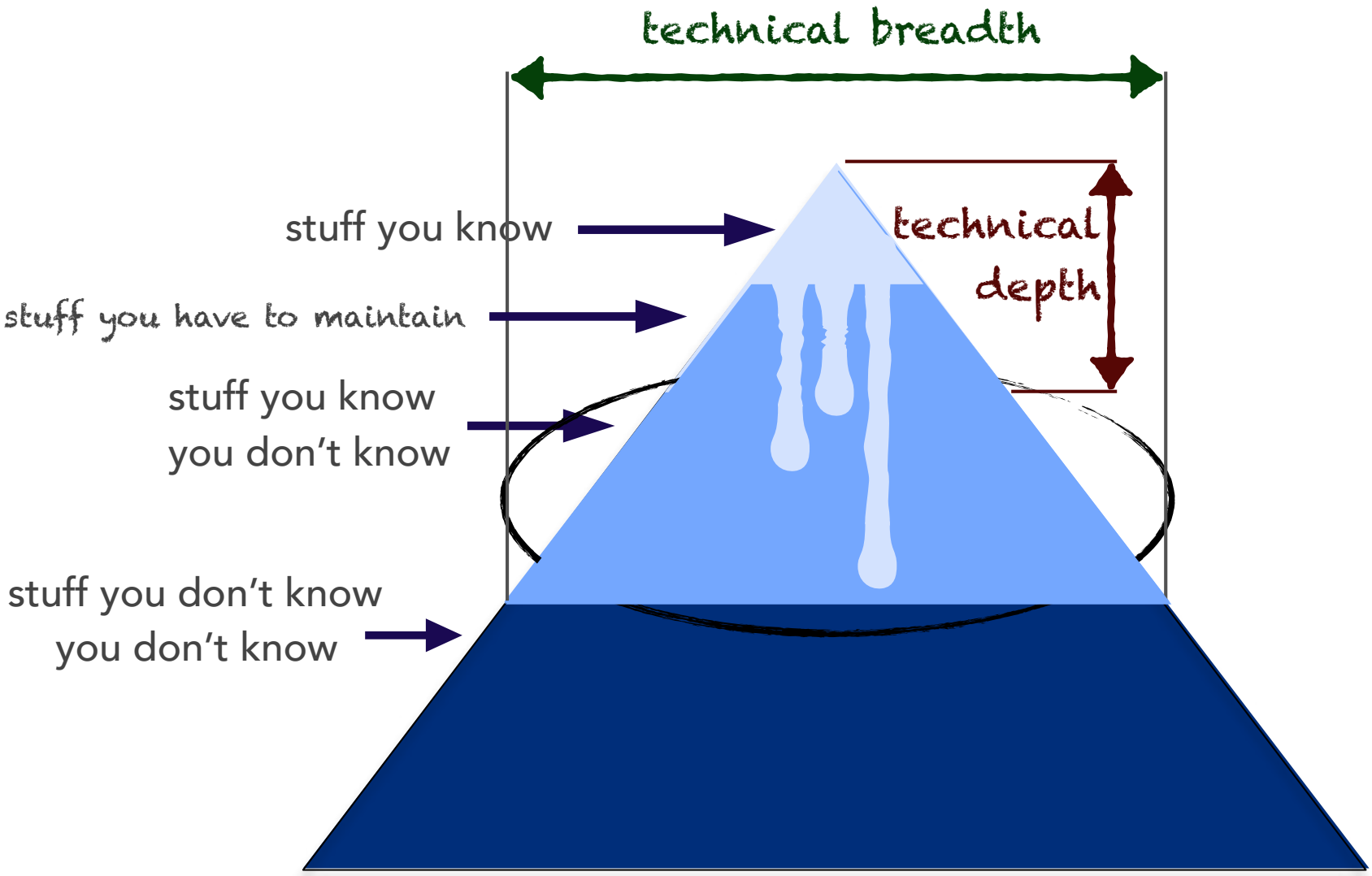
enterprise architecture context



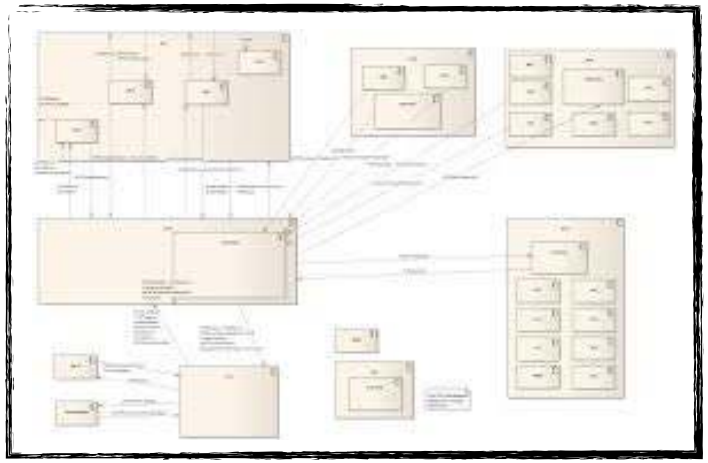
from developer to architect



the knowledge triangle



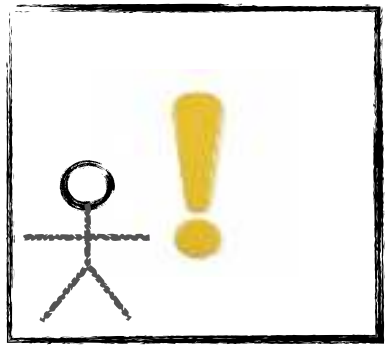
architectural boundaries



there is an art to defining the box that development teams can work in to implement the architecture

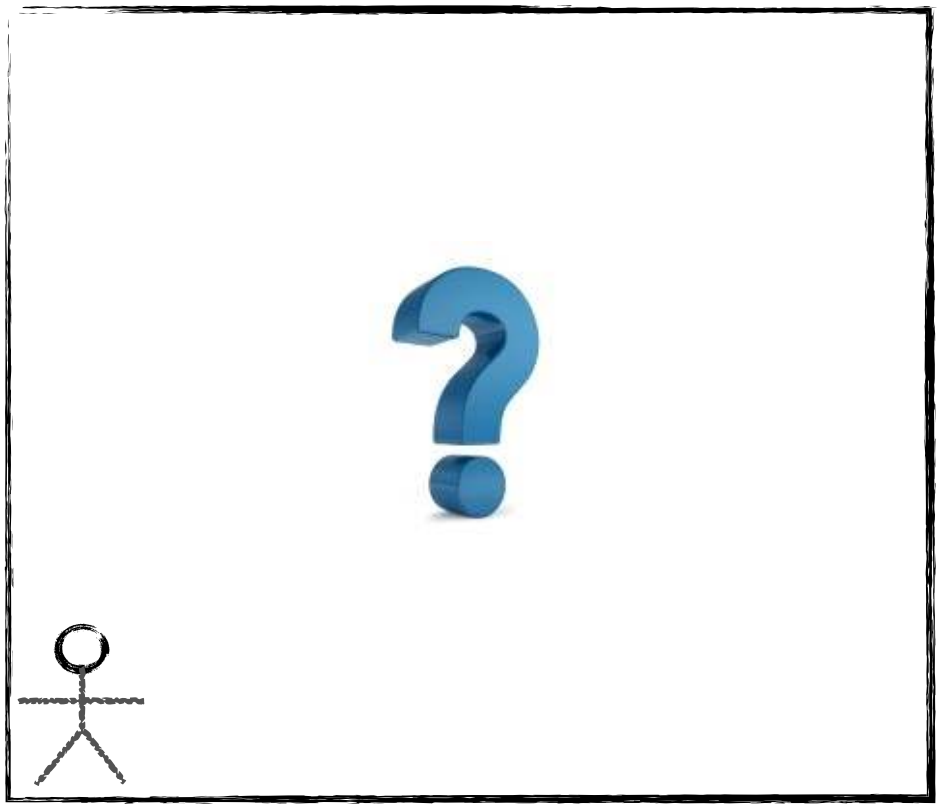
architectural boundaries

tight boundaries



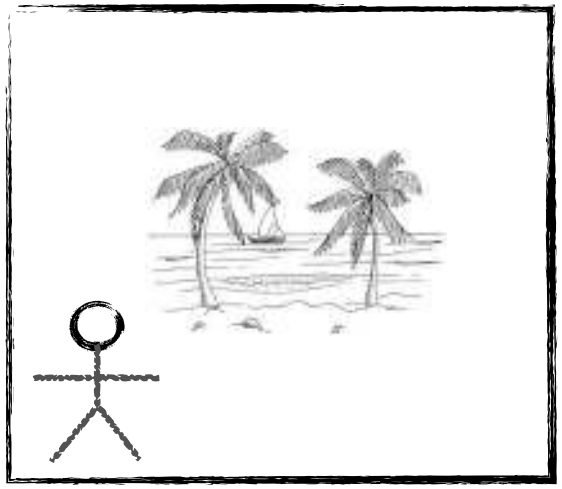
architectural boundaries

loose boundaries



architectural boundaries

appropriate boundaries



architect personalities

control freak architect

architect personalities

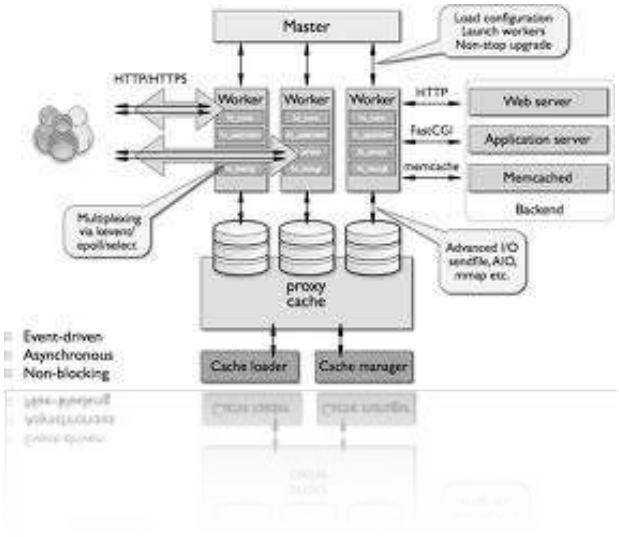
armchair architect

architect personalities

effective architect

controlling the boundaries

the architect defines the architecture and design principles used to guide technology decisions



controlling the boundaries

controlling the layered stack

development team: we decided to incorporate the guava library for the camel case conversion requirement.

"take it out. I only want you to use the core java api for this application. period."

controlling the boundaries

controlling the layered stack

development team: we decided to incorporate the guava library for the camel case conversion requirement.

"guava - that's a cool library name. carry on..."

controlling the boundaries

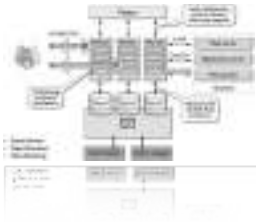
controlling the layered stack

development team: we decided to incorporate the guava library for the camel case conversion requirement.

if that's the only feature you are leveraging, you should just use the java api. if there are other features you can justify, then we can talk about it.

controlling the boundaries

controlling the layered stack



what design principle would you create to manage this type of boundary?

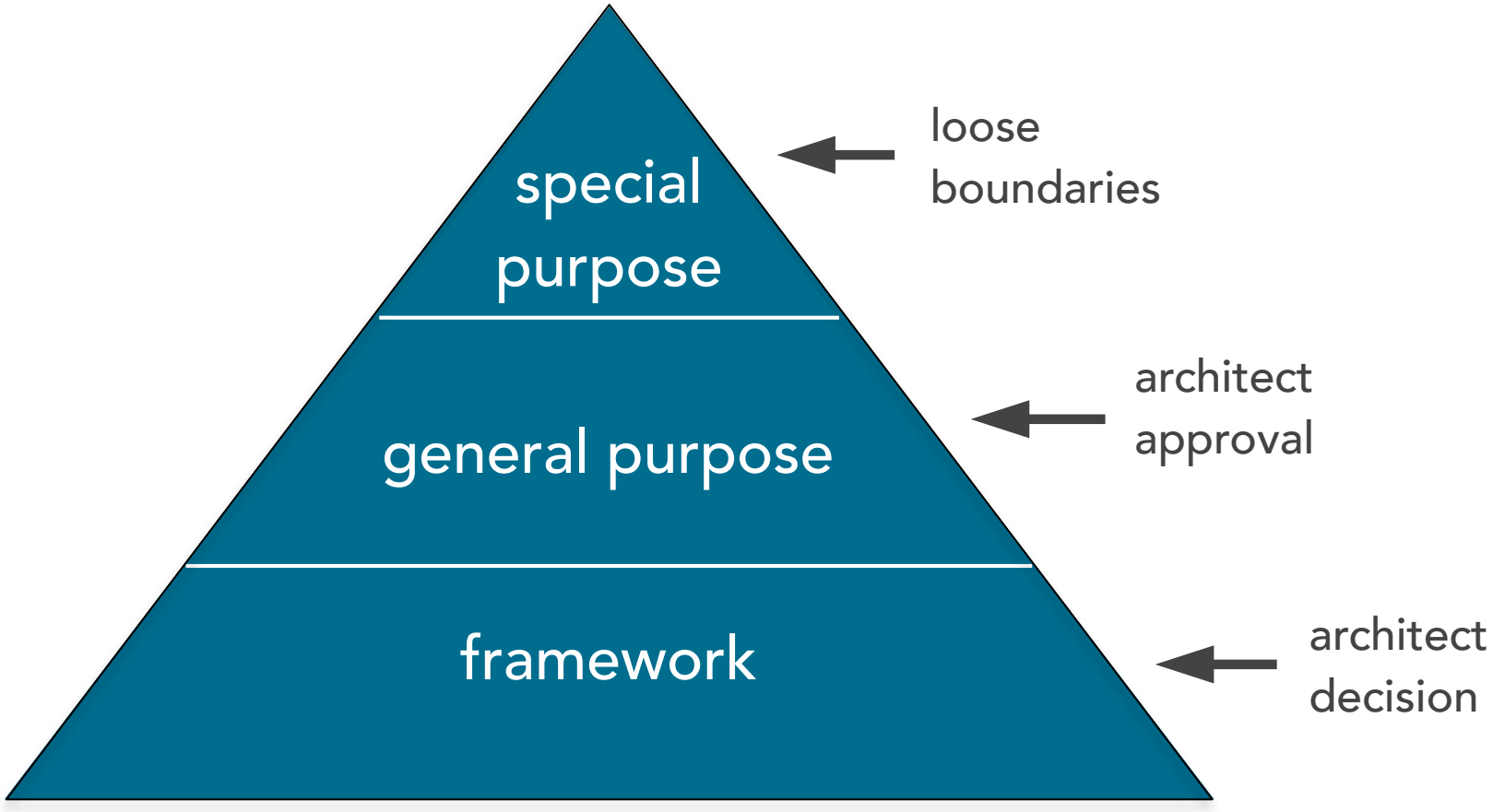
look for overlaps in existing functionality

always seek justification for adding a new library

provide guidance by making it clear what type of libraries need discussion and approval and which ones don't

controlling the boundaries

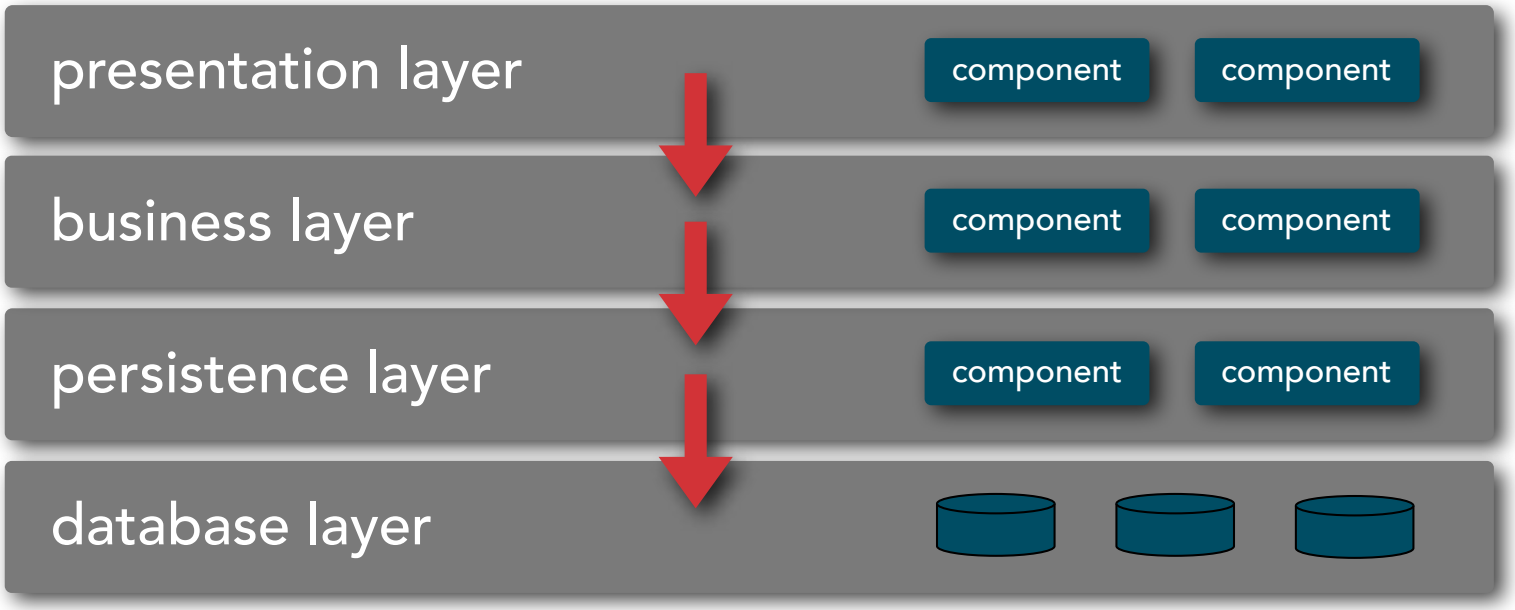
controlling the layered stack



controlling the boundaries

pattern governance

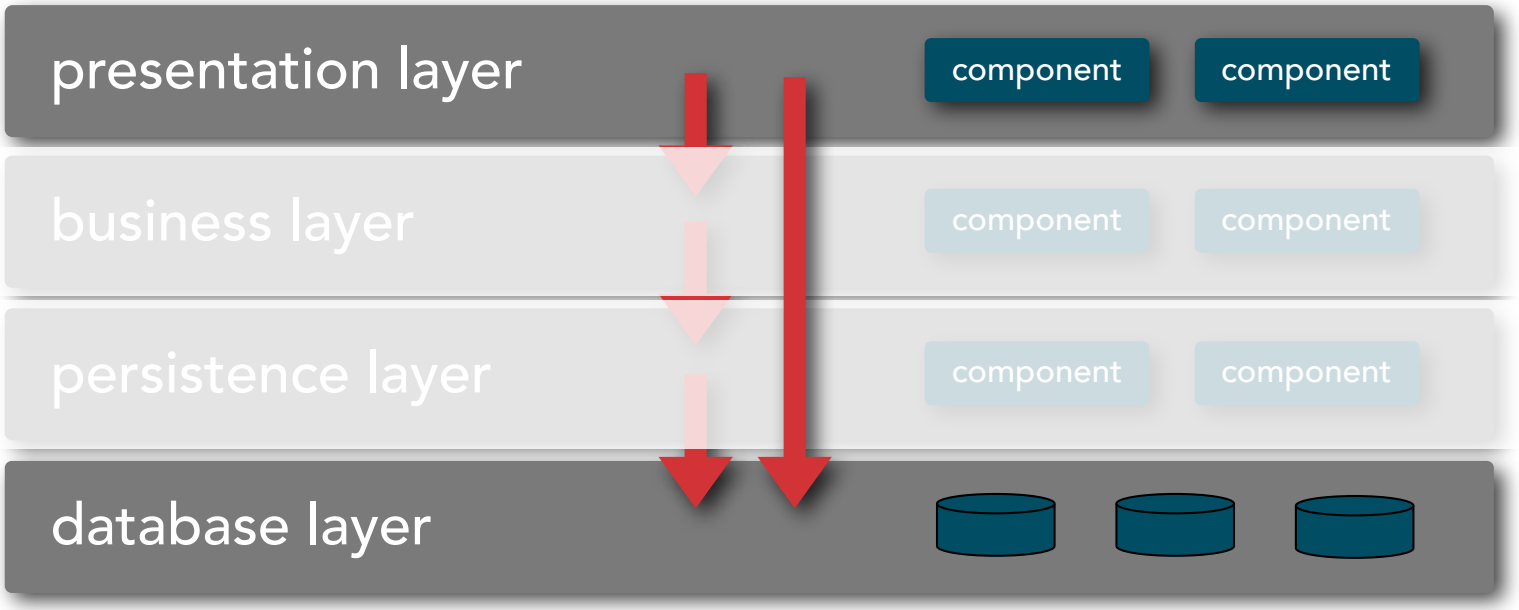
development team: we need better performance - can we access the database directly from the presentation layer?



controlling the boundaries

pattern governance

development team: we need better performance - can we access the database directly from the presentation layer?



controlling the boundaries
pattern governance

development team: we need better performance - can we access the database directly from the presentation layer?

"no."

controlling the boundaries

pattern governance

development team: we need better performance - can we access the database directly from the presentation layer?

"it doesn't matter to me. if you think it would help performance, then go for it."

controlling the boundaries

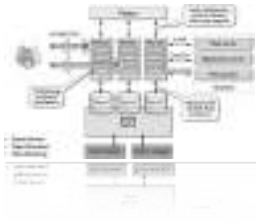
pattern governance

development team: we need better performance - can we access the database directly from the presentation layer?

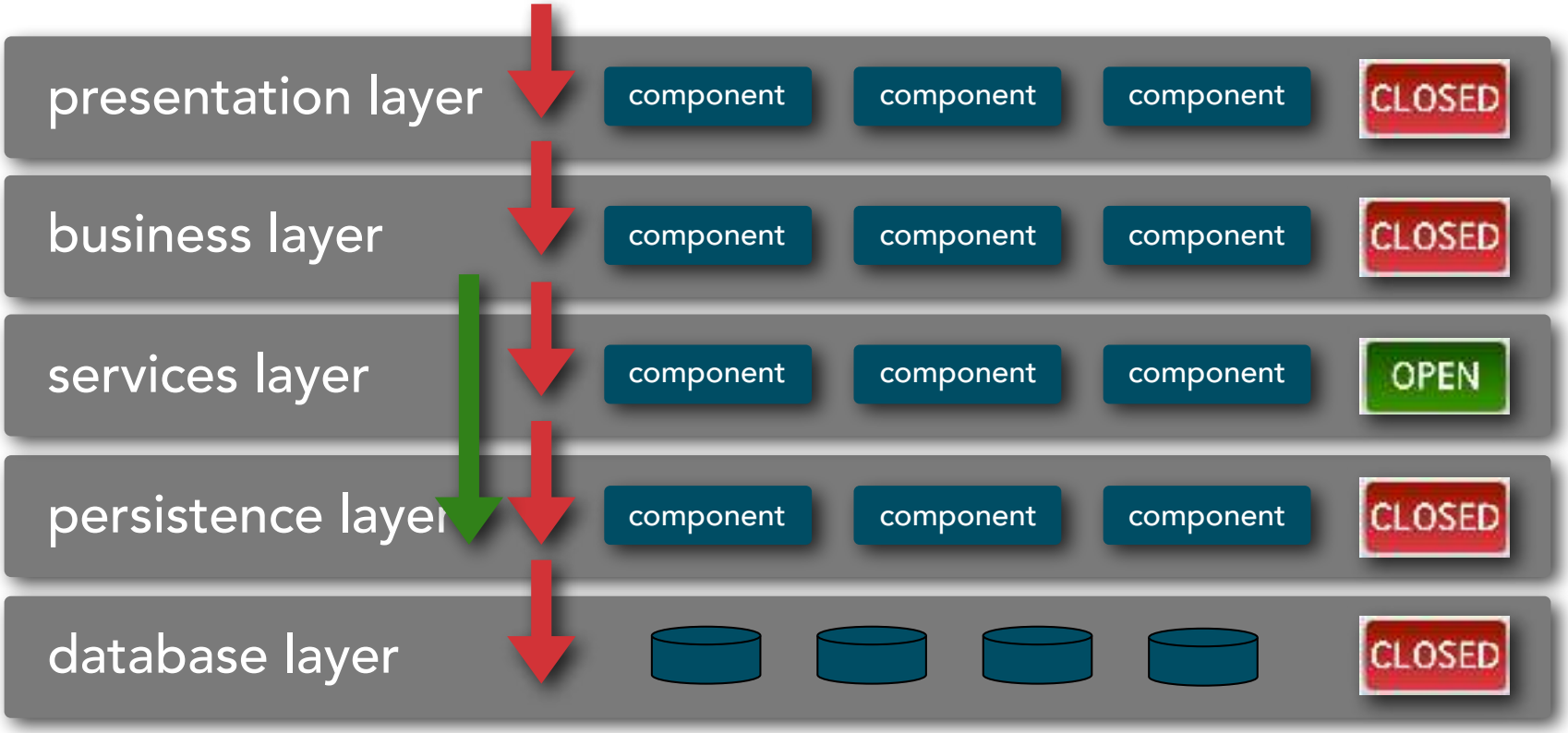
"those layers are closed so that we can better control change through layer isolation, so no. have you been able to identify what might be causing the performance issues?"

controlling the boundaries

pattern governance



what design principle would you create to manage this type of boundary?



controlling the boundaries

pattern governance

what design principle would you create to manage this type of boundary?

clearly document and diagram the architecture

justify your reasons for the architecture decisions

make sure you effectively communicate your decisions

controlling the boundaries

architecture scope

development team: we added some really cool capabilities that might be needed sometime in the future...

"did i tell you to add those capabilities? didn't think so. take them out."

controlling the boundaries

architecture scope

development team: we added some really cool capabilities that might be needed sometime in the future...

"great forward thinking guys! someday the users might need that capability, and now we have it ready..."

controlling the boundaries

architecture scope

development team: we added some really cool capabilities that might be needed sometime in the future...

"let's verify those features with the analysts to see if we need them. if not then we'll take them out. we just need to make sure we don't do anything to prevent that capability from being added in the future."

controlling the boundaries

architecture scope

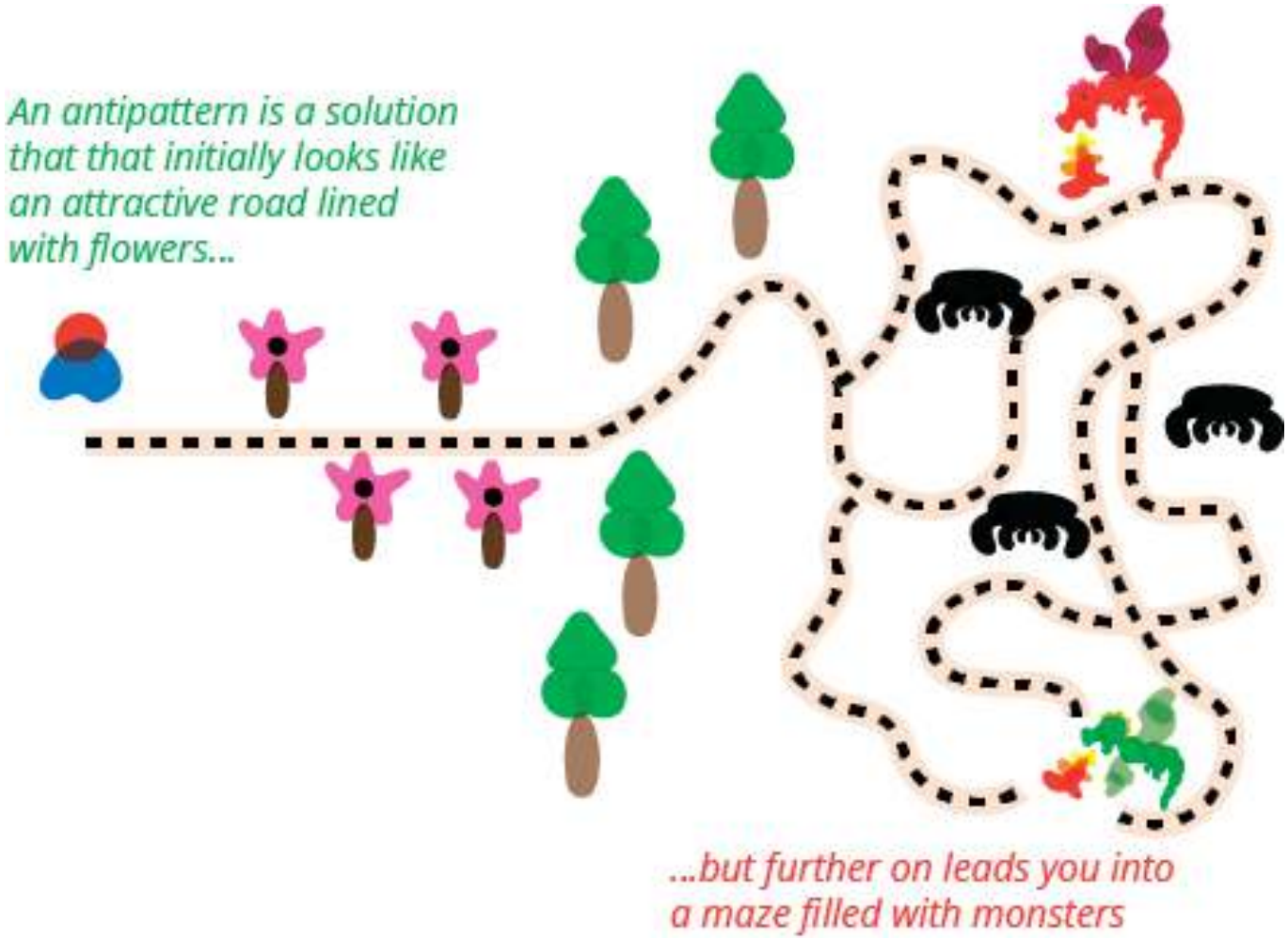
what design principle would you create to manage this type of boundary?

adding additional features over and above the requirements adds additional development, testing, and maintenance time and costs

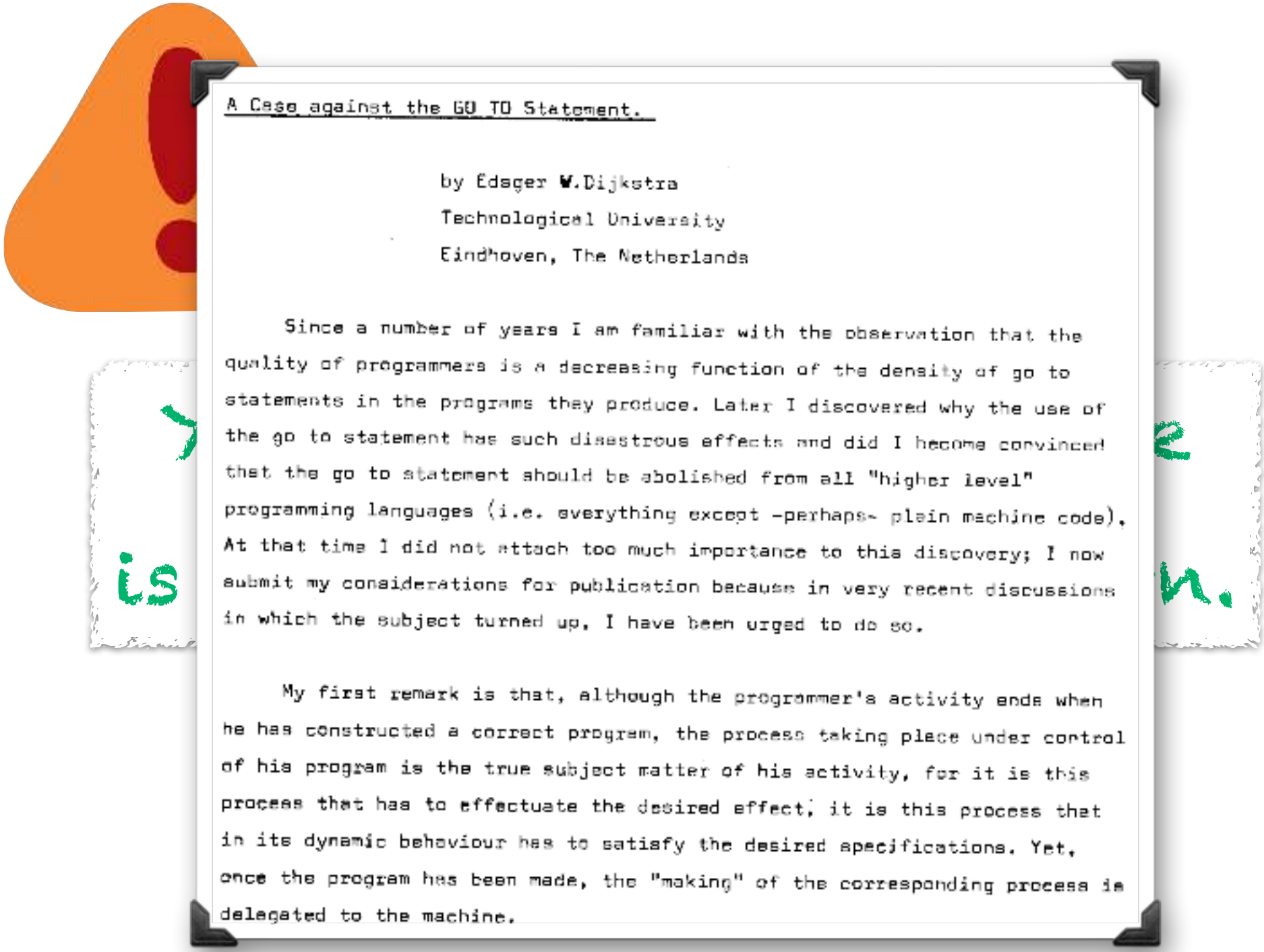
this practice can lead to the *infinity architecture anti-pattern*

document non-required features, verify it with the user community, and ***make sure you don't do anything to restrict that functionality in the future***

Architecture Anti-pattern



Yesterday's best practice is tomorrow's anti-pattern.



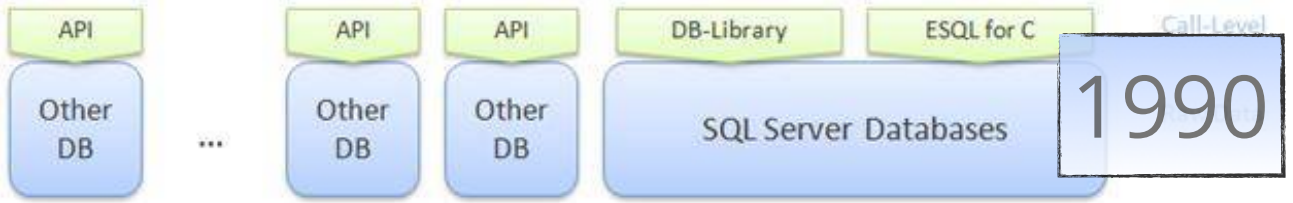
A Case against the GO TO Statement.

by Edsger W.Dijkstra
Technological University
Eindhoven, The Netherlands

Since a number of years I am familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce. Later I discovered why the use of the go to statement has such disastrous effects and did I become convinced that the go to statement should be abolished from all "higher level" programming languages (i.e. everything except -perhaps- plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

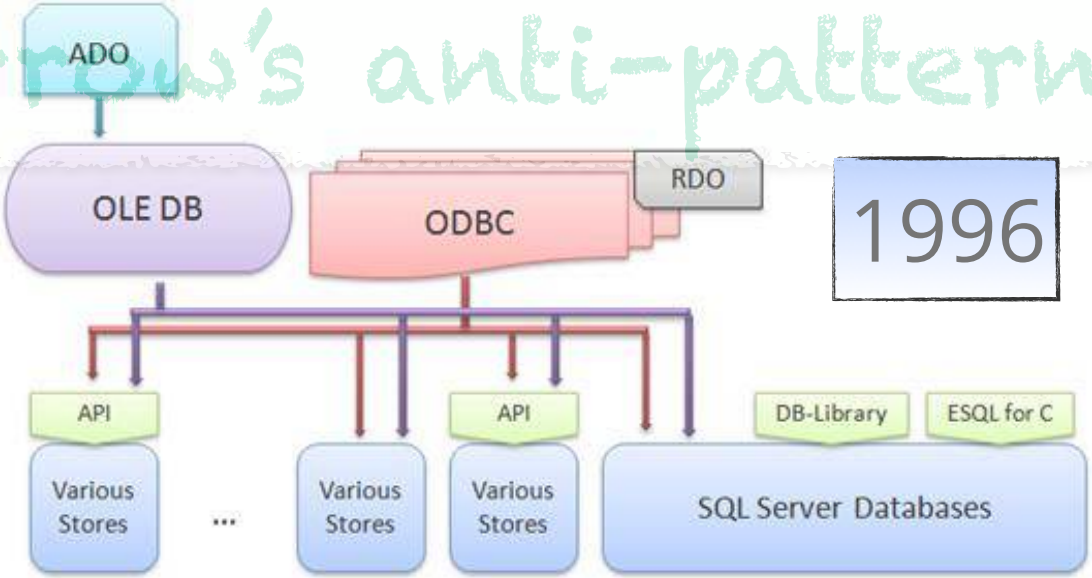
My first remark is that, although the programmer's activity ends when he has constructed a correct program, the process taking place under control of his program is the true subject matter of his activity, for it is this process that has to effectuate the desired effect; it is this process that in its dynamic behaviour has to satisfy the desired specifications. Yet, once the program has been made, the "making" of the corresponding process is delegated to the machine.

www.cs.utexas.edu/users/EWD/ewd02xx/EWD215.PDF

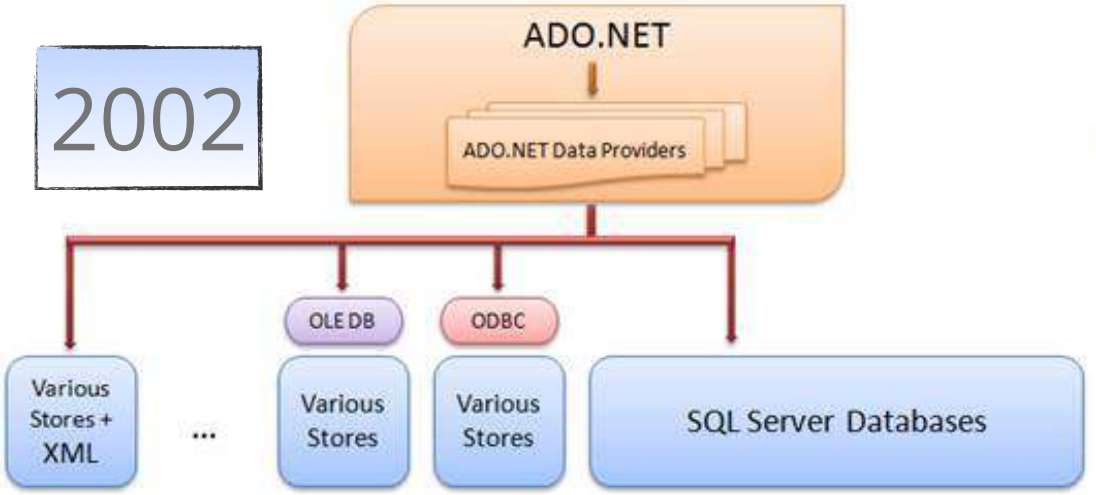


Yesterday's best practice

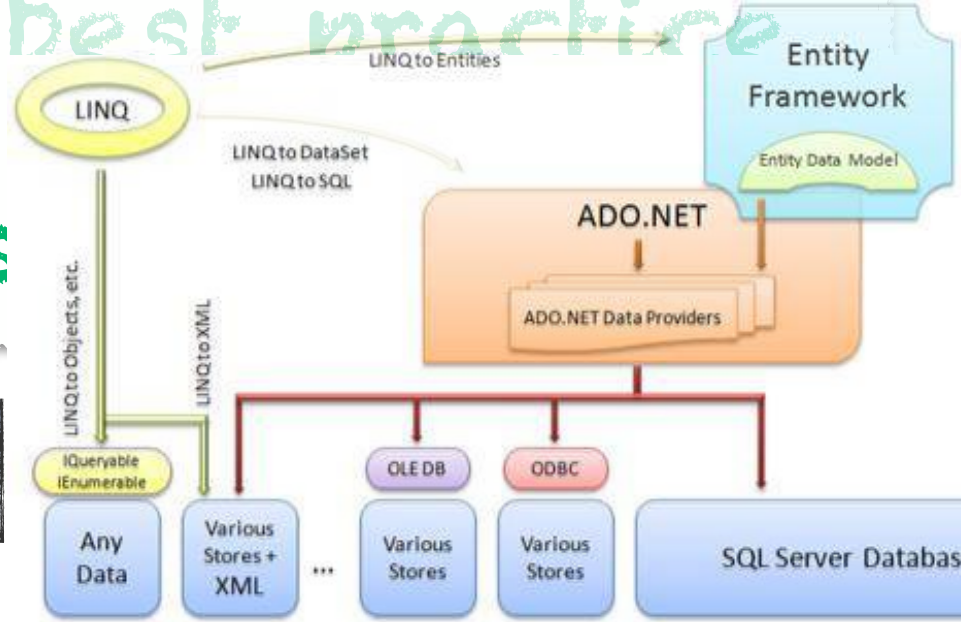
is tomorrow's anti-pattern.



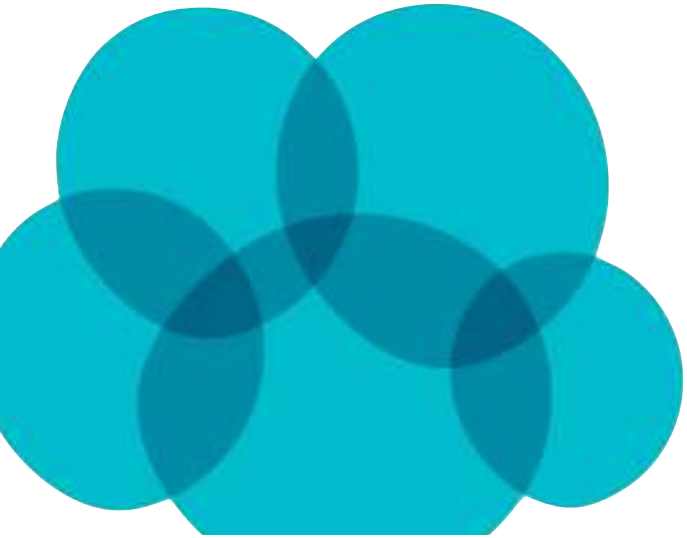
2002



Yesterday's best practice is tomorrow's



2007



2008

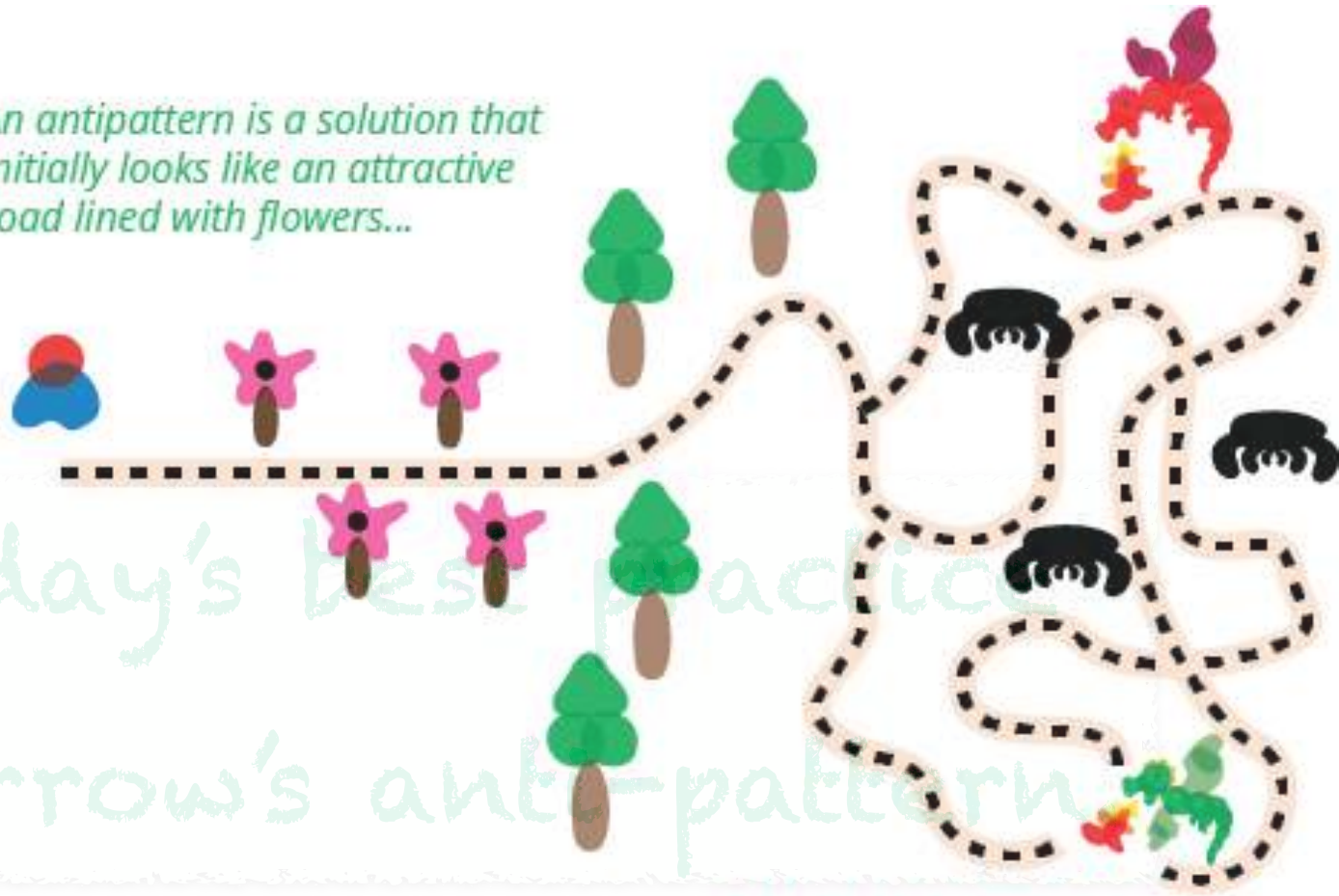
Yesterday's best practice

is tomorrow's anti-pattern.

ENTERPRISE

Java Beans

An antipattern is a solution that initially looks like an attractive road lined with flowers...



Yesterday's best practice

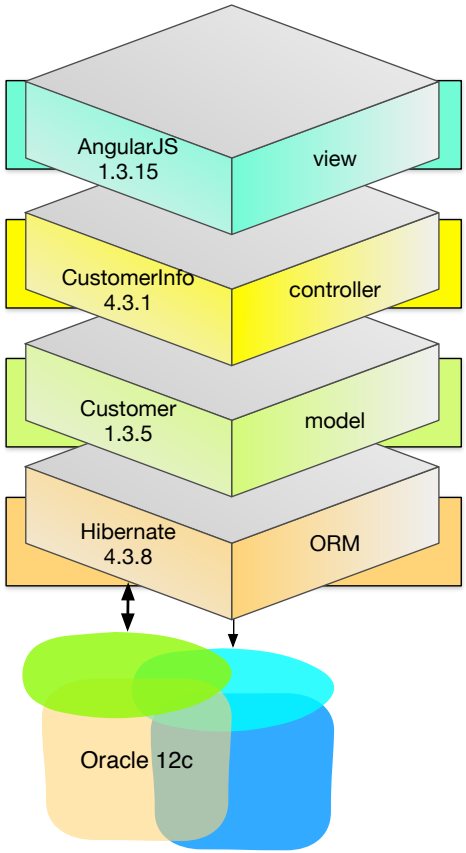
is tomorrow's anti-pattern

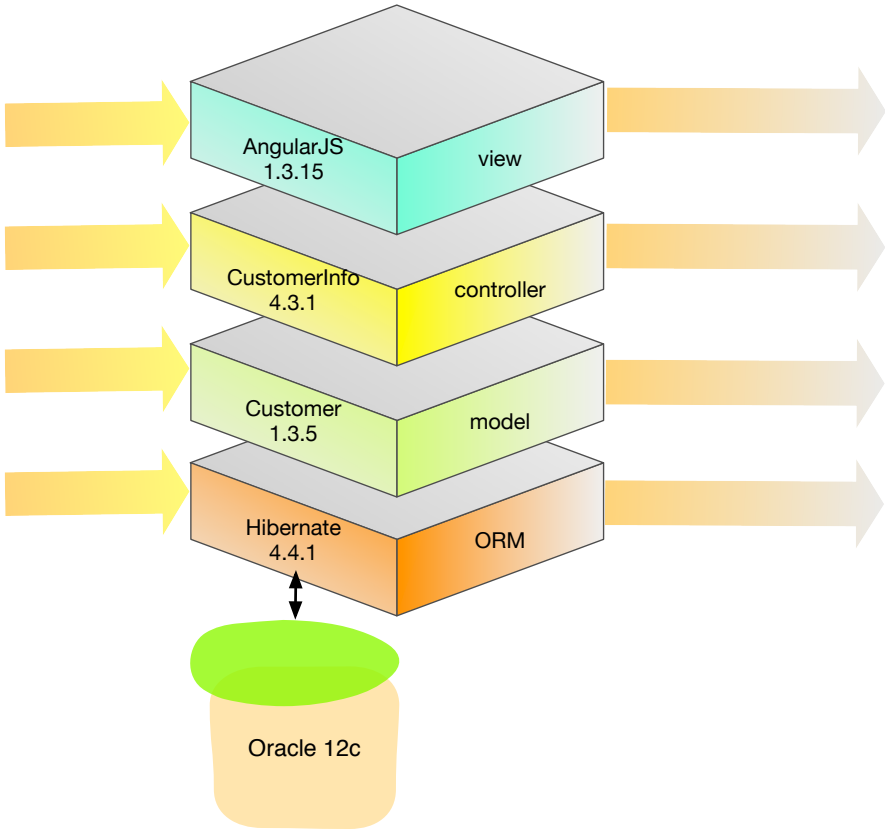
...but further on leads you into a maze filled with monsters

martinfowler.com/bliki/AntiPattern.html

nealford.com/memeagora/2015/03/30/architecture_is_abstract_until_operationalized.html

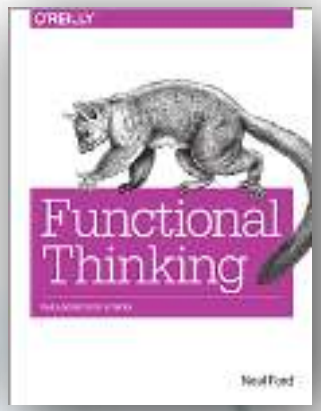
Architecture is abstract until operationalized.





2D 3D 4D

nealford.com/memeagora/2015/03/30/architecture_is_abstract_until_operationalized.html



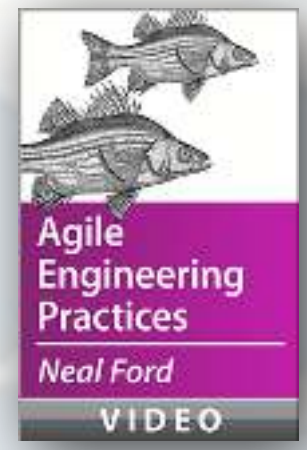
nealford.com

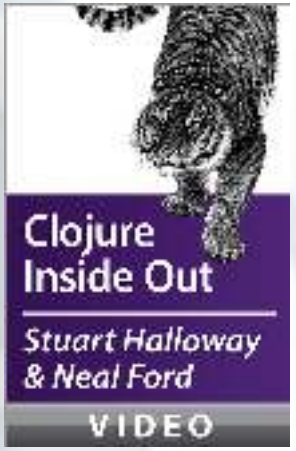
@neal4d

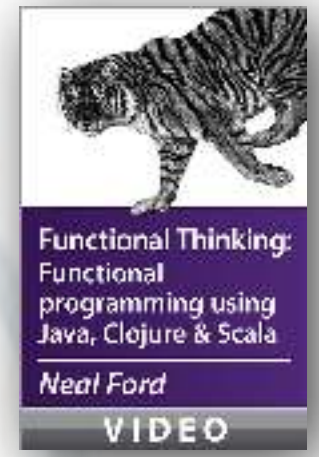
ThoughtWorks

NEAL FORD

Director / Software Architect / Meme Wrangler







O'REILLY®

SOFTWARE ARCHITECTURE SERIES