Creating Software Architectures

Neal Ford
ThoughtWorks
Director / Software Architect / Meme Wrangler
http://www.nealford.com
@neal4d

Mark Richards
Hands-on Software Architect
Published Author / Conference Speaker
http://www.wmrichards.com
https://www.linkedin.com/in/markrichards3
@markrichardssa

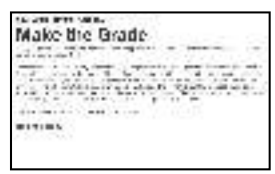
workshop agenda - day 1

course introduction

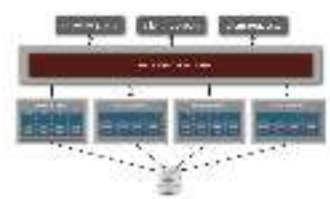
architectural thinking

architecture characteristics

architecture tradeoffs

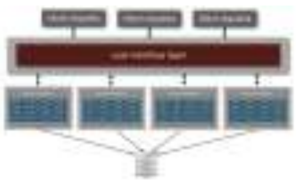


architecture katas



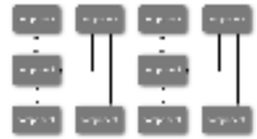
architecture patterns

workshop agenda - day 2



architecture
patterns (cont)

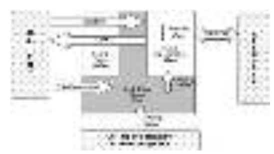
architecture
katas



component-based
thinking

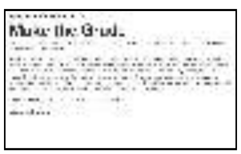
architecture
katas

architecture
decisions



documenting
architectures

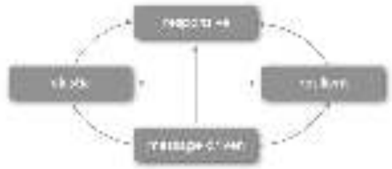
presenting
architectures



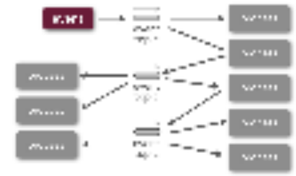
architecture
katas

workshop agenda - day 3

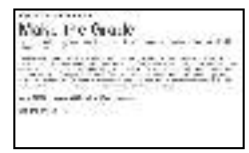
analyzing architectures



reactive architecture



evolutionary architecture



architecture katas

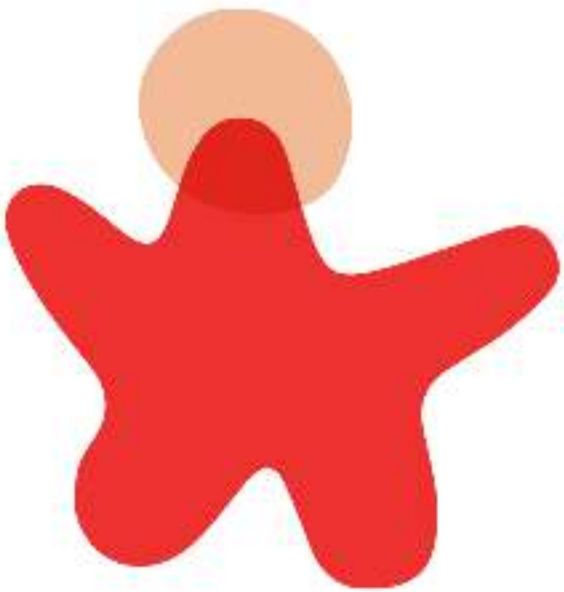
next steps

course slides

http://www.wmrichards.com/sdd-architecture-2016.pdf

password: sdd

attendee introductions



your name

role or title

why are you here?

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?

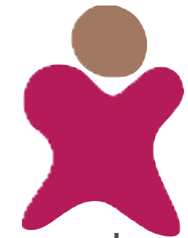
Architecture is the highest level concept of the expert developers.

"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

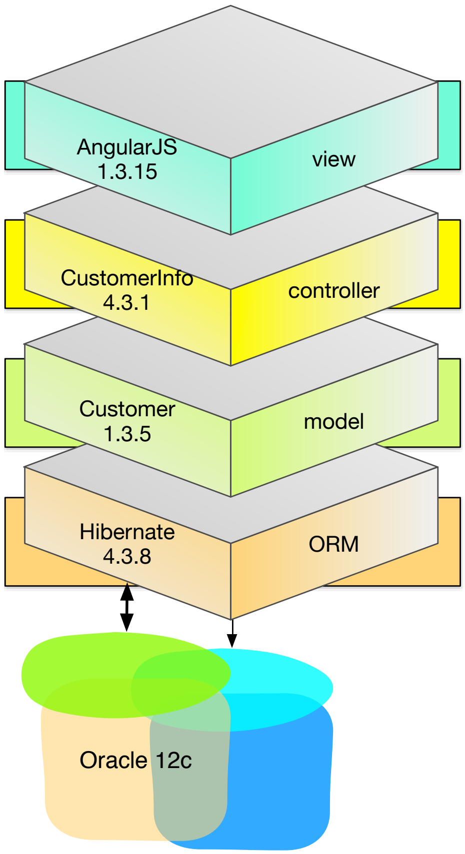
Architecture is about the important stuff.
Whatever that is.

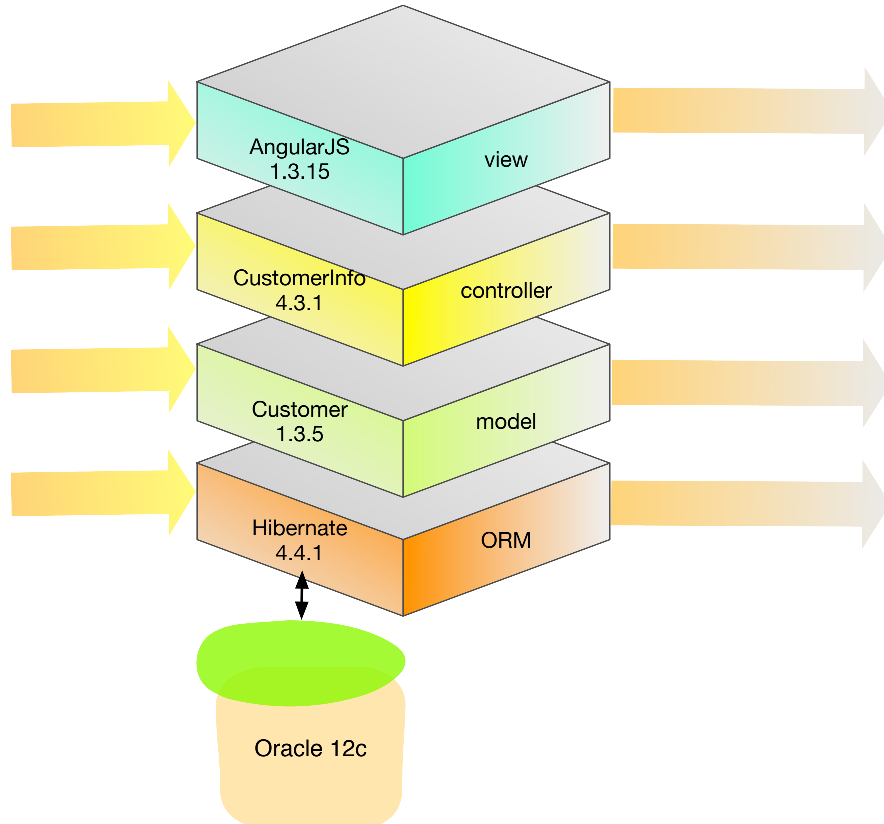
Ralph Johnson

operations

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

Architecture is abstract until operationalized.





2D 3D 4D

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

expectations of an architect

expectations of an architect

Application Architect

Enterprise Architect

Integration Architect

Technical Architect

Information Architect

Security Architect

Data Architect

Network Architect

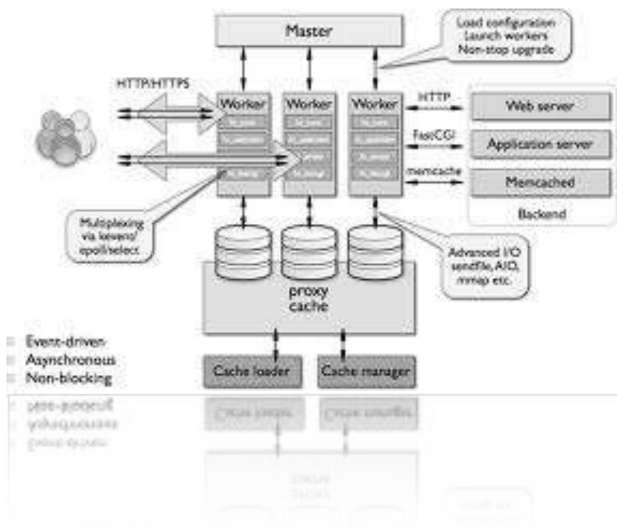
Systems Architect

Solutions Architect

Business Architect

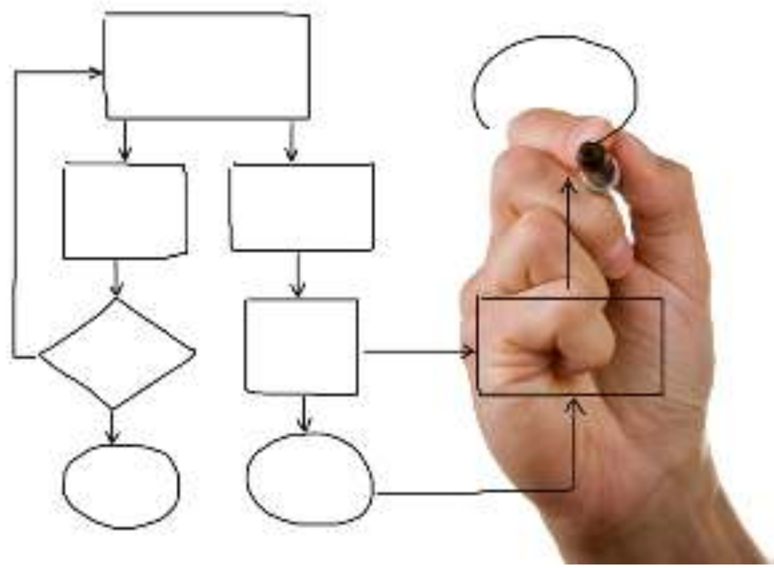
expectations of an architect

define the architecture and design principles to guide technology decisions for the enterprise



expectations of an architect

analyze the current technology environment and recommend solutions for improvement.



expectations of an architect

analyze technology and industry trends and keep current with the latest trends

expectations of an architect

ensure compliance with the architecture

expectations of an architect

have exposure to multiple and diverse technologies, platforms, and environments

expectations of an architect

have a certain level of business domain expertise

expectations of an architect

possess exceptional interpersonal skills, including teamwork, facilitation, and negotiation

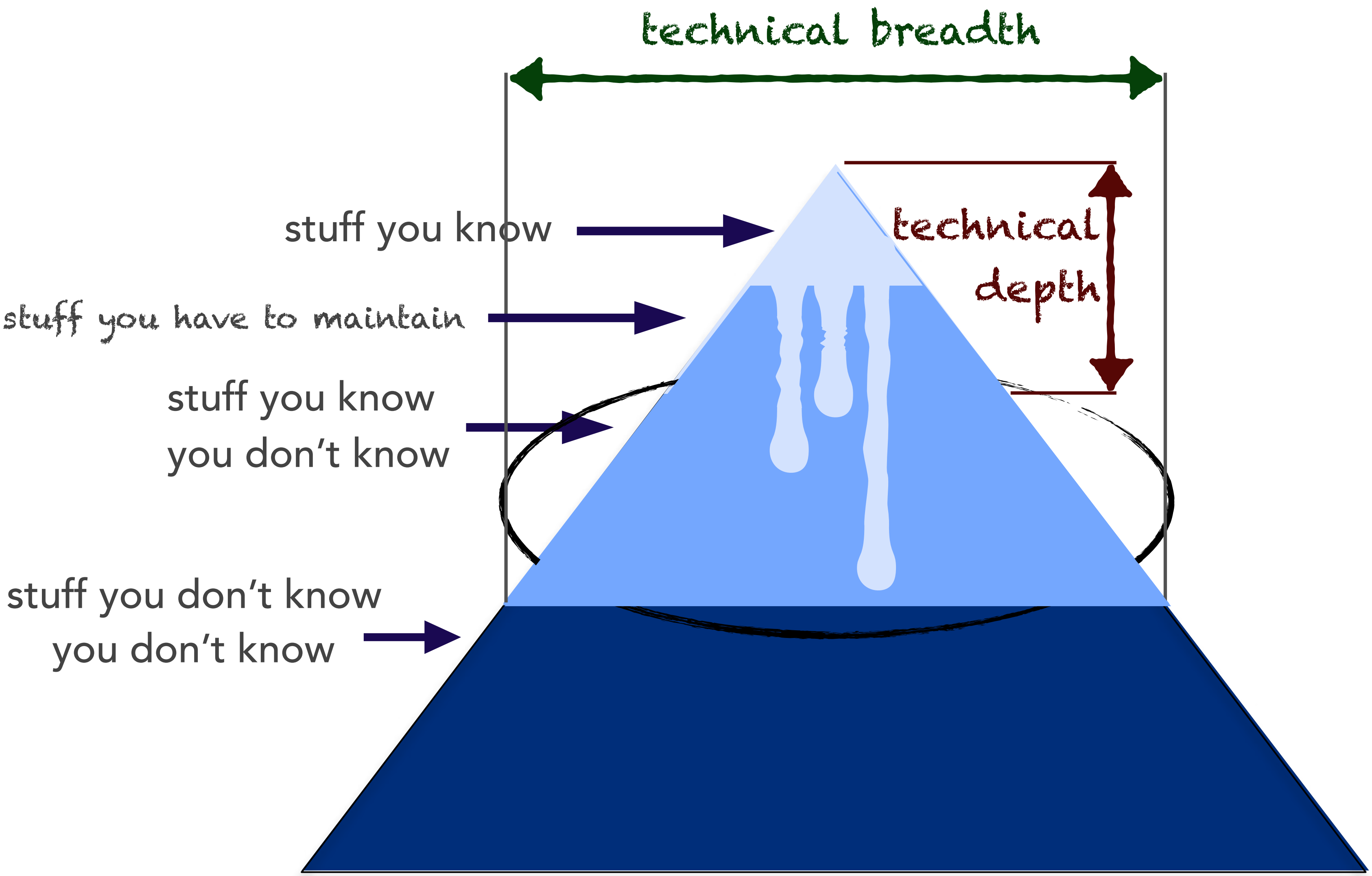
expectations of an architect

understand the political climate of the enterprise and be able to navigate the politics

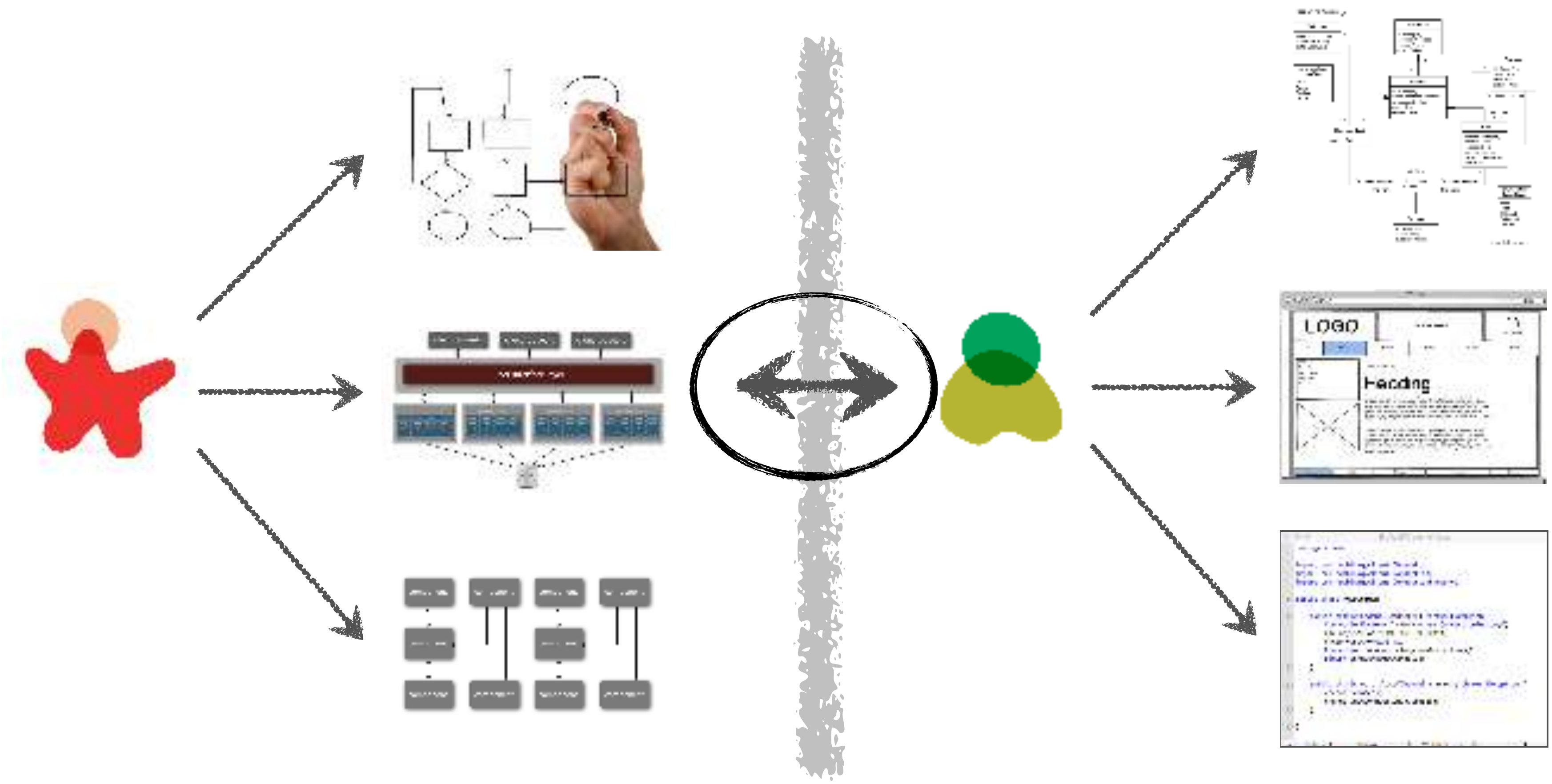

That’s what I do.
I drink, and I
know things.

architectural thinking

technical breadth vs. depth



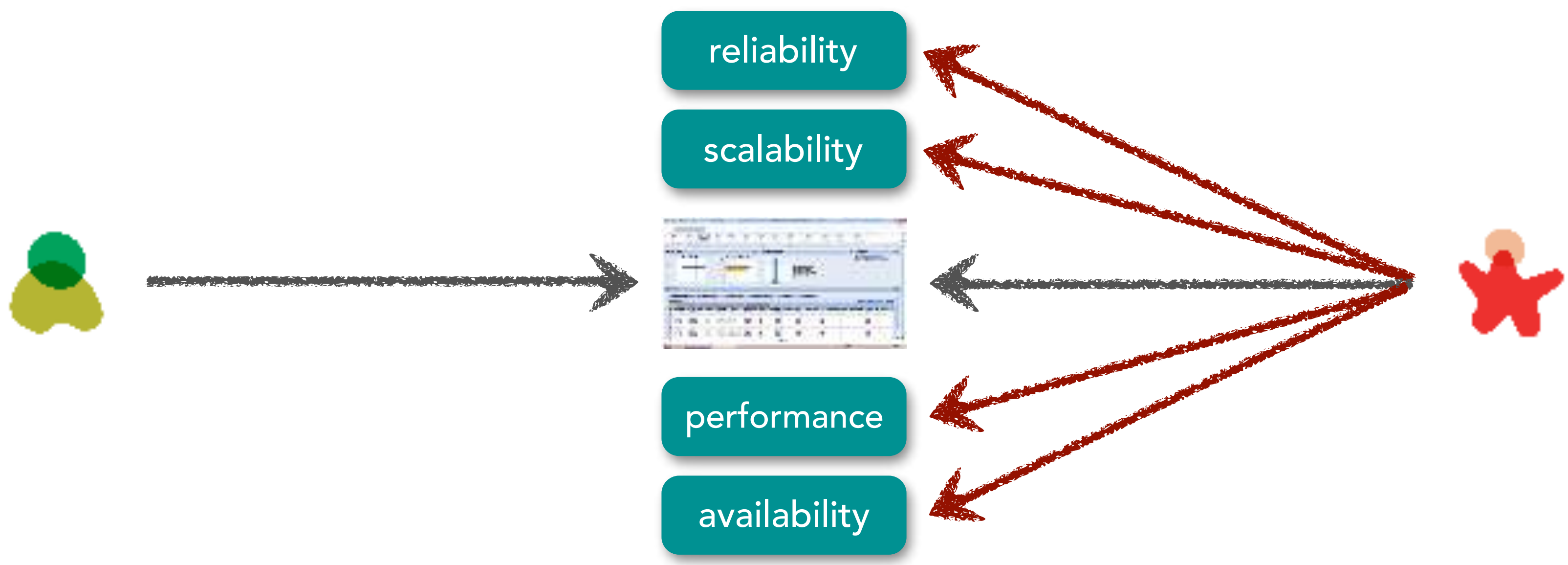
where do you draw the line between architecture and design?



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

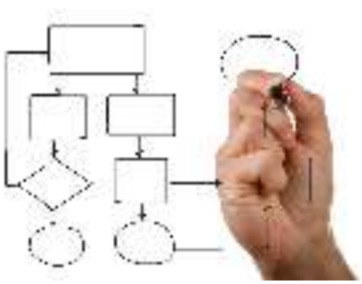
evolvability
extensibility
failure transparency
fault-tolerance
fidelity
flexibility
inspectability
installability
integrity
interchangeability
interoperability
learnability
maintainability
manageability
mobility
modifiability
modularity
operability
orthogonality
portability
precision
predictability
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

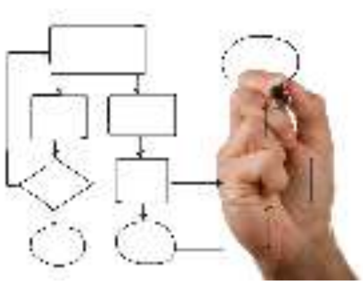
"our business is constantly changing to meet new demands of the marketplace"



???

architecture characteristics

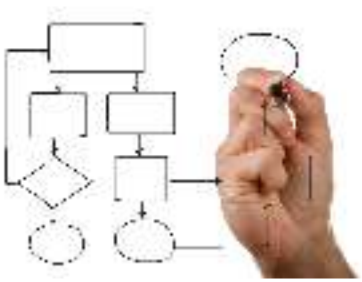
"due to new regulatory requirements, it is imperative that we complete end-of-day processing in time"



???

architecture characteristics

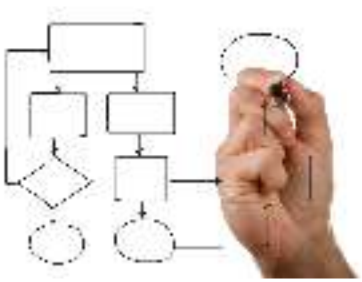
“we need faster time to market to remain competitive”



???

architecture characteristics

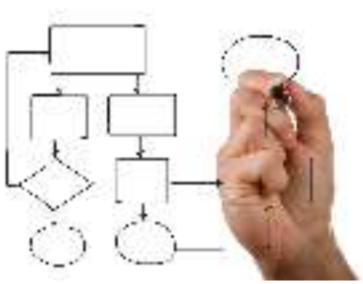
“our plan is to engage heavily in mergers and acquisitions in the next three years”



???

architecture characteristics

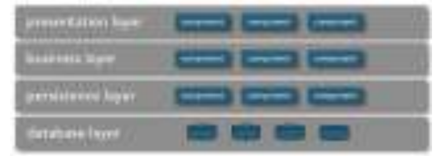
“we have a very tight timeframe and budget for this project”



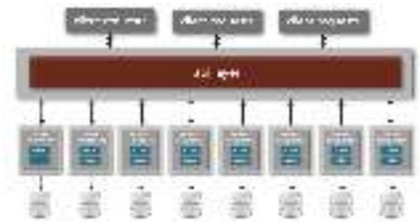
???

architecture characteristics

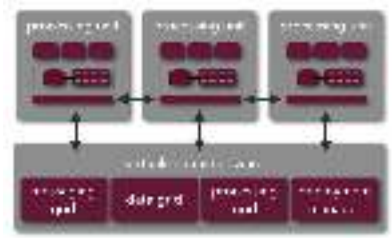
feasibility



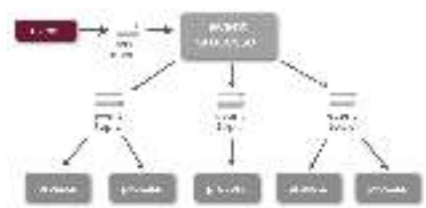
agility

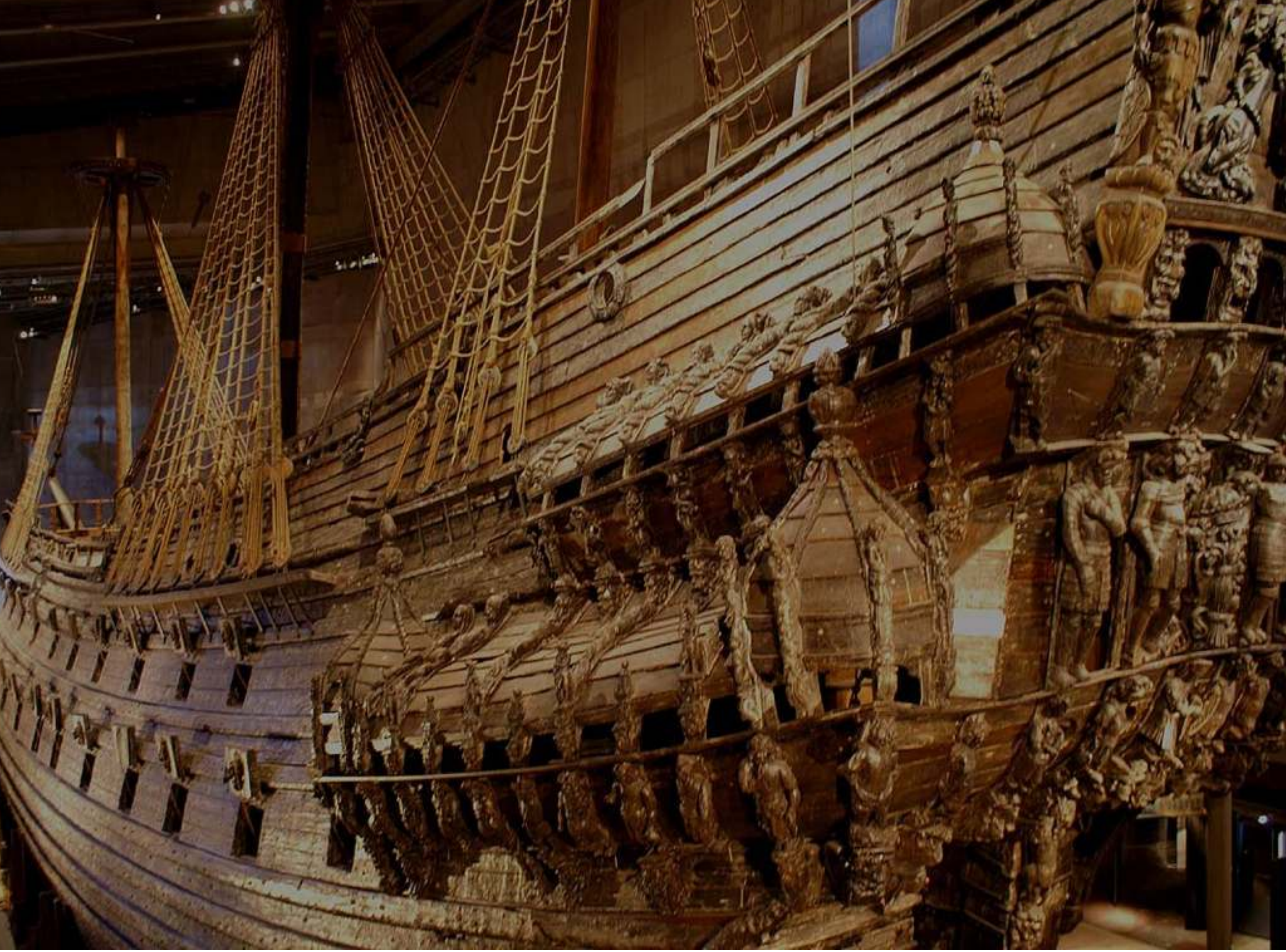


elasticity



scalability





architecture tradeoffs

"we need lightning-fast response time to keep up with the backlog of calls"

performance

"over time we are expecting the entire company to use this system"

scalability

"we are planning to acquire several businesses in the next 5 years"

extensibility agility maintainability

"the budget and timeframe for this system is very, very tight"

feasibility

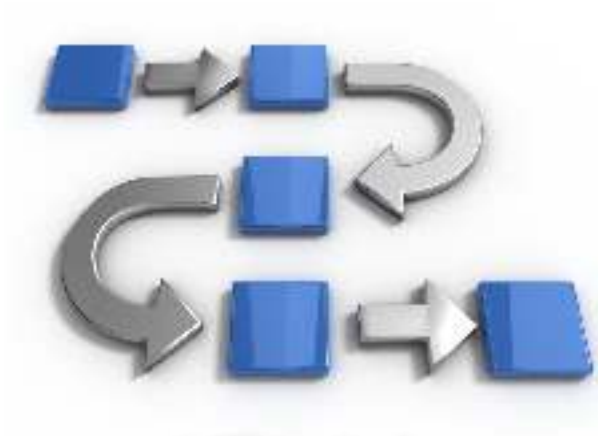
architecture tradeoffs

architecture tradeoff analysis method (ATAM)

proposed architecture →

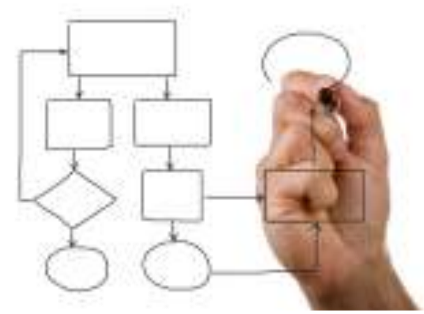
business drivers →

quality attributes →



ATAM

→

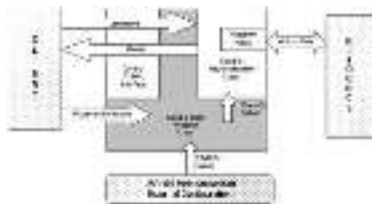


validated and approved architecture

architecture tradeoffs

cost-benefit analysis method (CBAM)

business goals



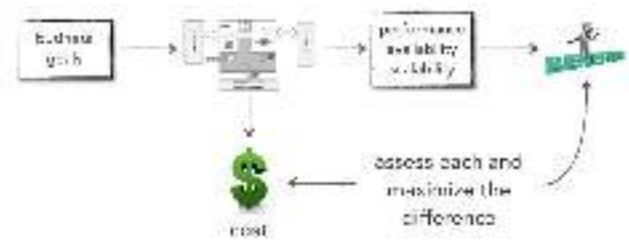
performance
availability
scalability

cost

assess each and maximize the difference

architecture tradeoffs

ATAM



CBAM

Software Architecture
in Practice 3rd Edition,
Bass et.al,
Addison Wesley

Software Engineering Institute

Software Engineering Institute
Digital Library
http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=5177

http://www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm

architecture katas

familiarization and identifying architecture characteristics


Your Architectural Kata is...

Make the Grade

[illegible]

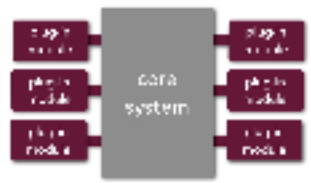
[illegible]

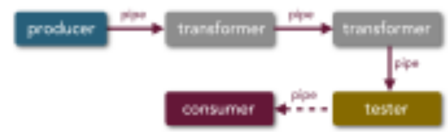
[illegible]

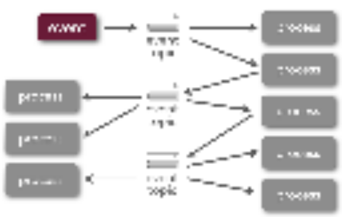
assigned team: 1

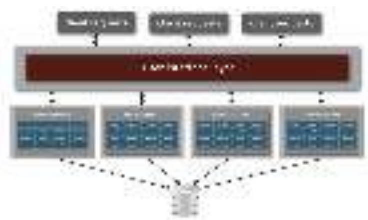
software architecture patterns

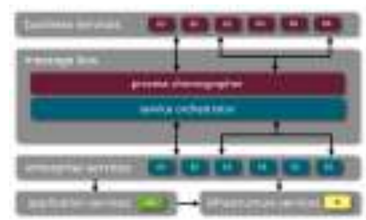
architecture patterns help define the basic characteristics and behavior of the application

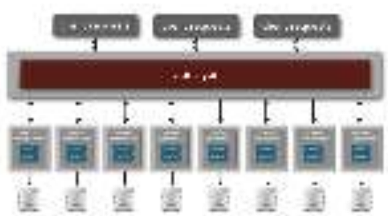


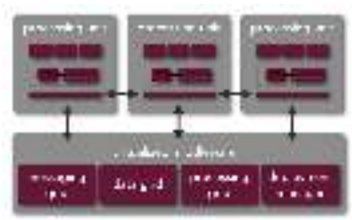








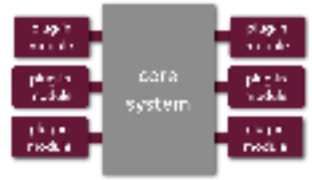




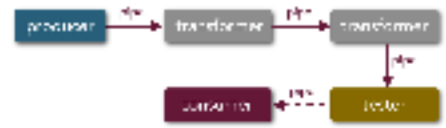
architecture pattern classification

monolithic

layered

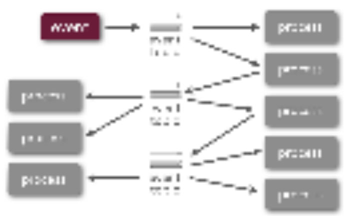


microkernel

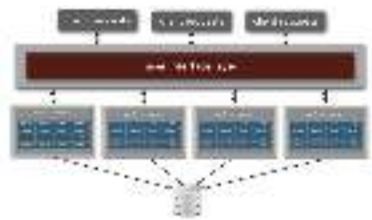


pipeline

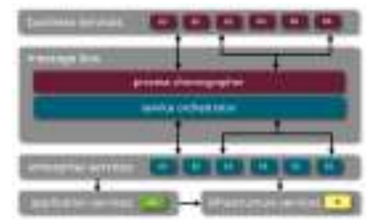
distributed



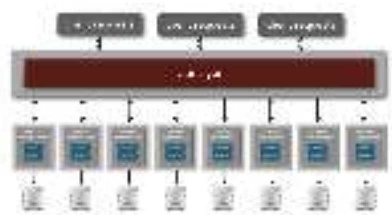
event-driven



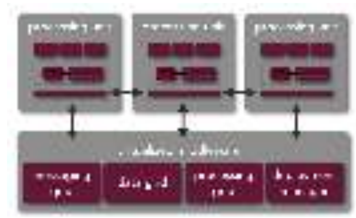
service-based



service-oriented



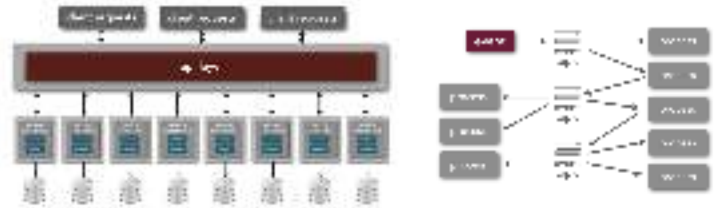
microservices



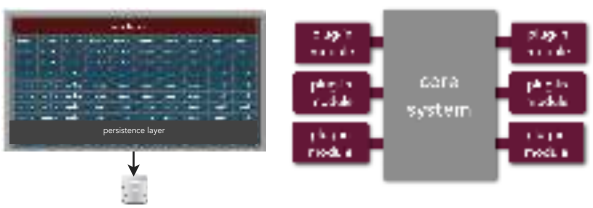
space-based

architecture pattern hybrids

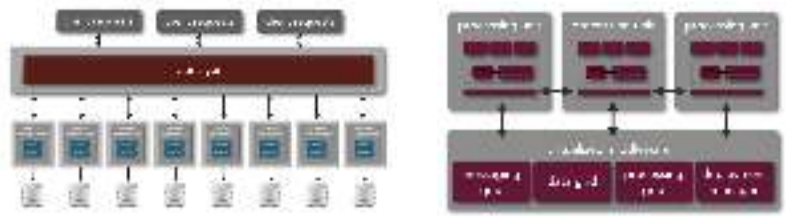
event-driven layered



event-driven microservices



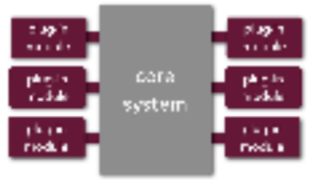
layered microkernel



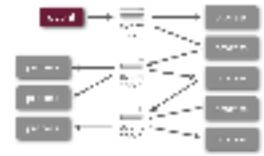
space-based microservices

architecture pattern roadmap

layered architecture

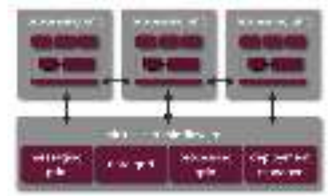


microkernel architecture

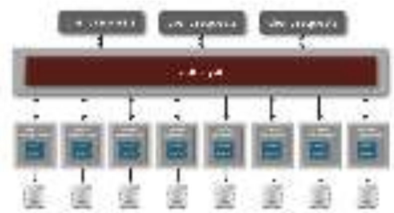


event-driven architecture

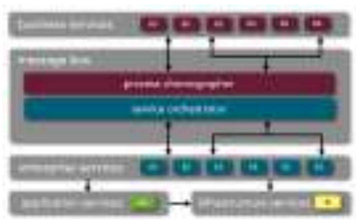
pipeline architecture



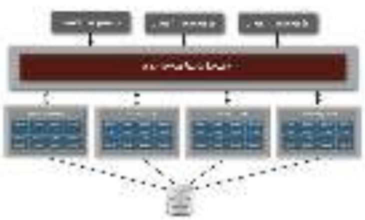
space-based architecture



microservices architecture



service-oriented architecture



service-based architecture

architecture pattern roadmap

layered architecture

microkernel architecture

event-driven architecture

pipeline architecture

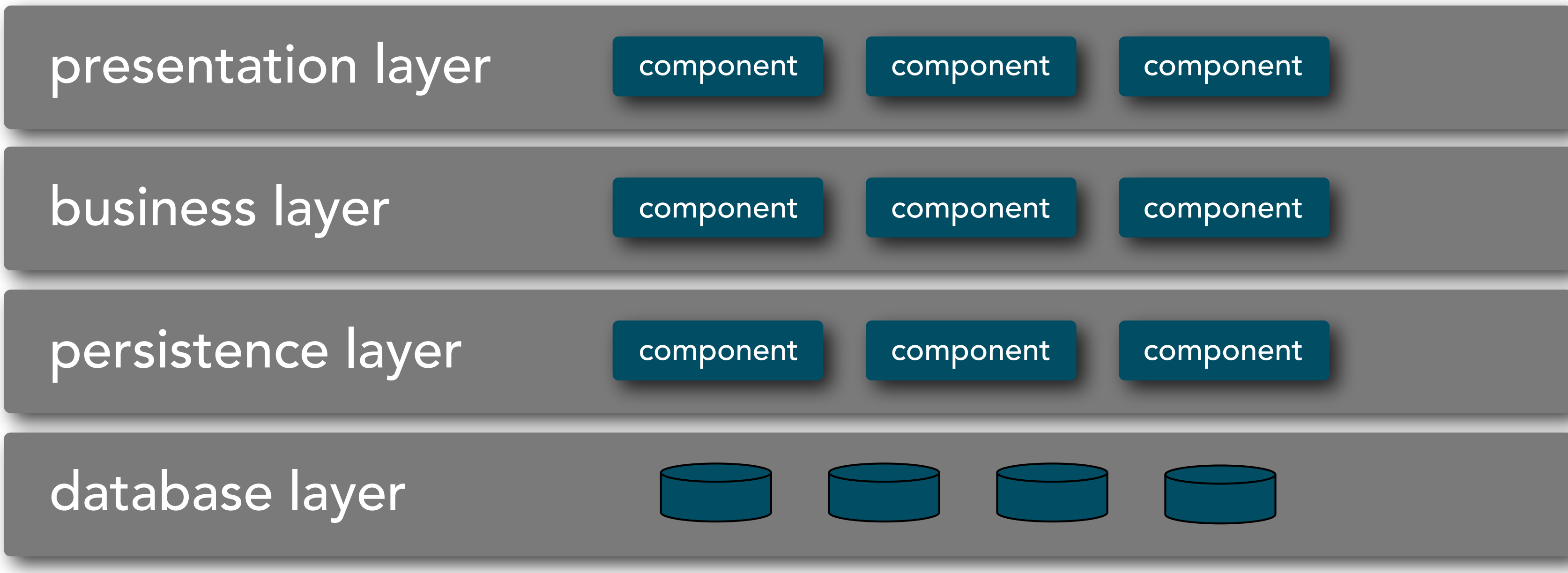
space-based architecture

microservices architecture

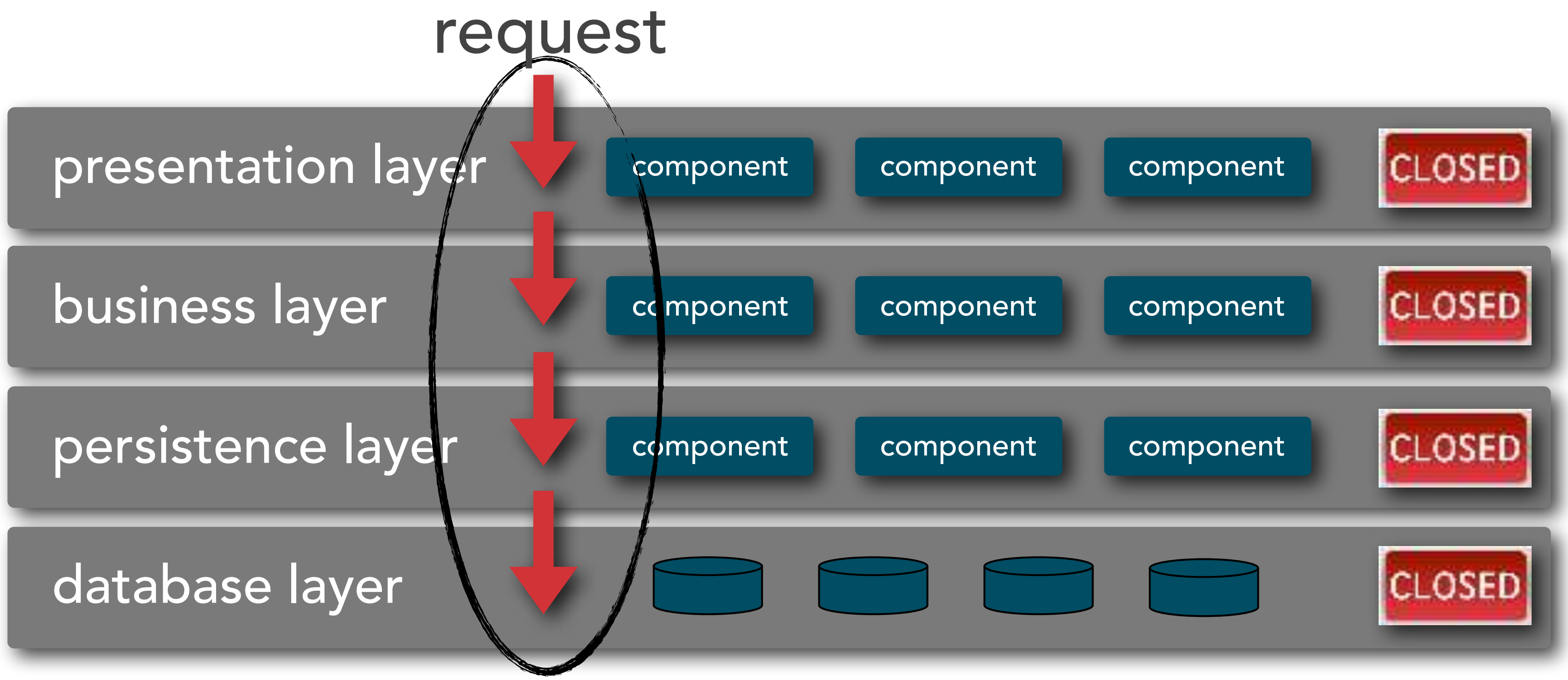
service-oriented architecture

service-based architecture

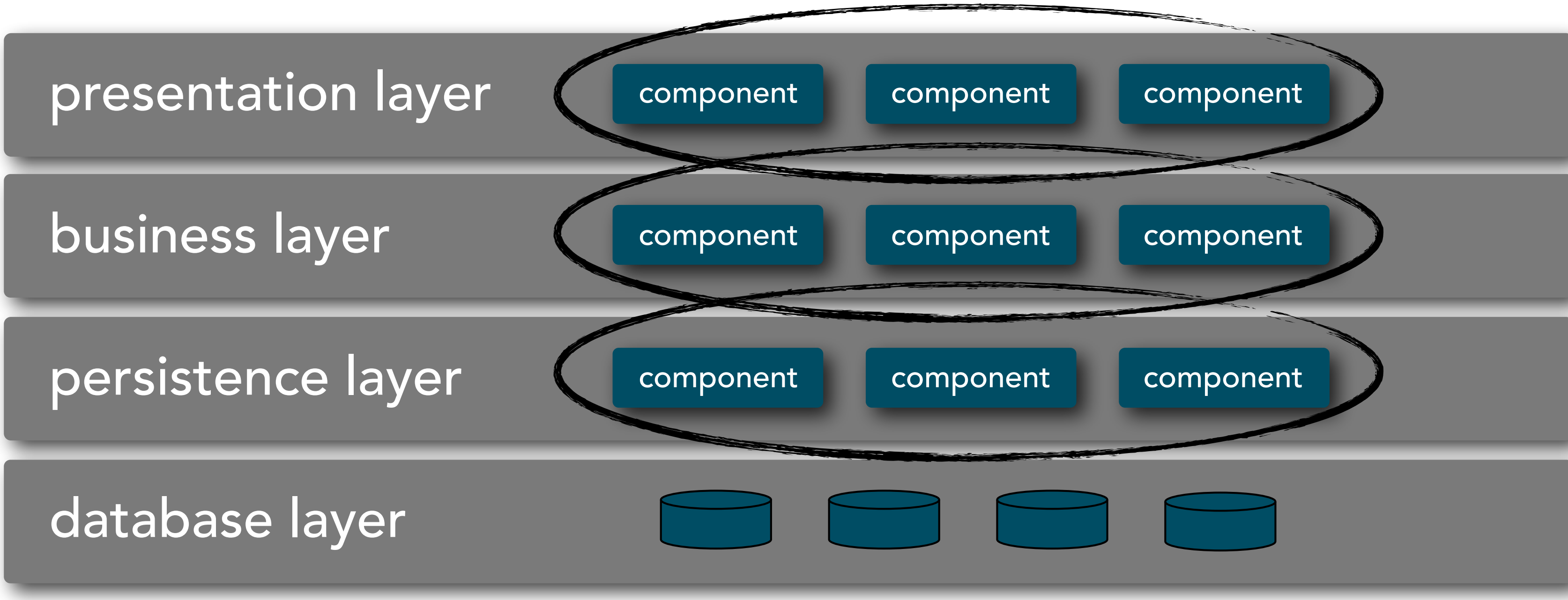
layered architecture



layered architecture

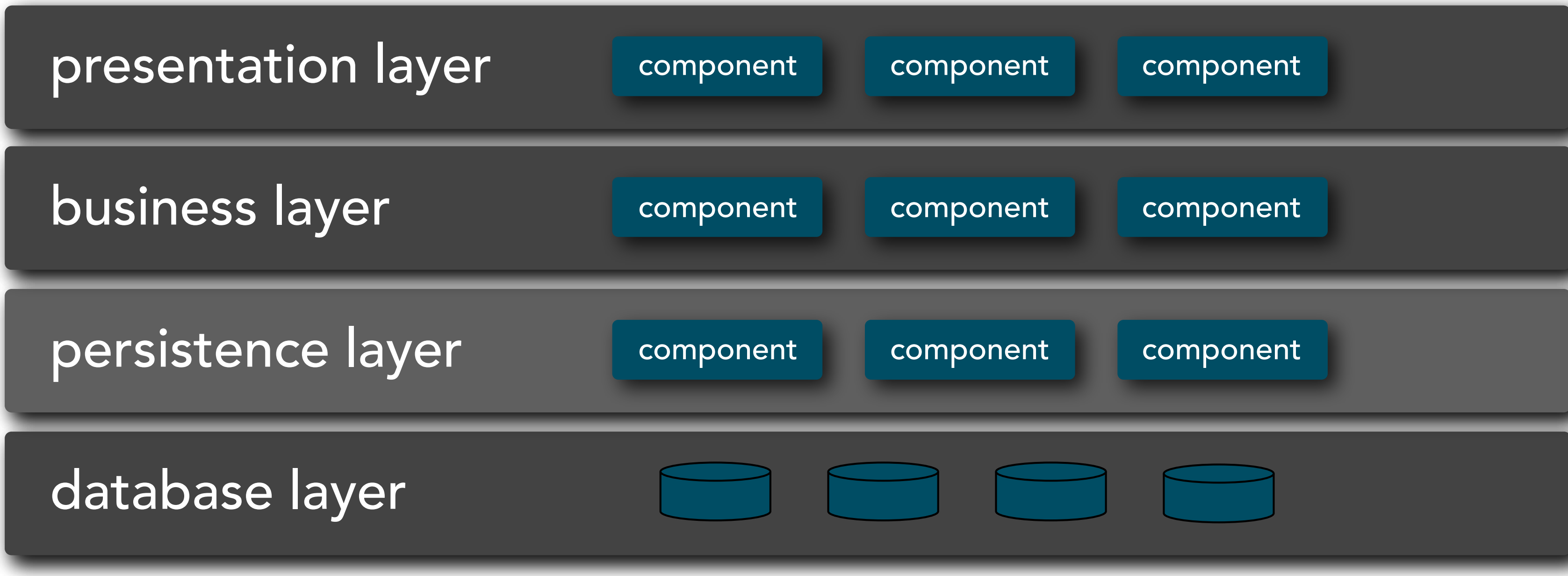


layered architecture



separation of concerns

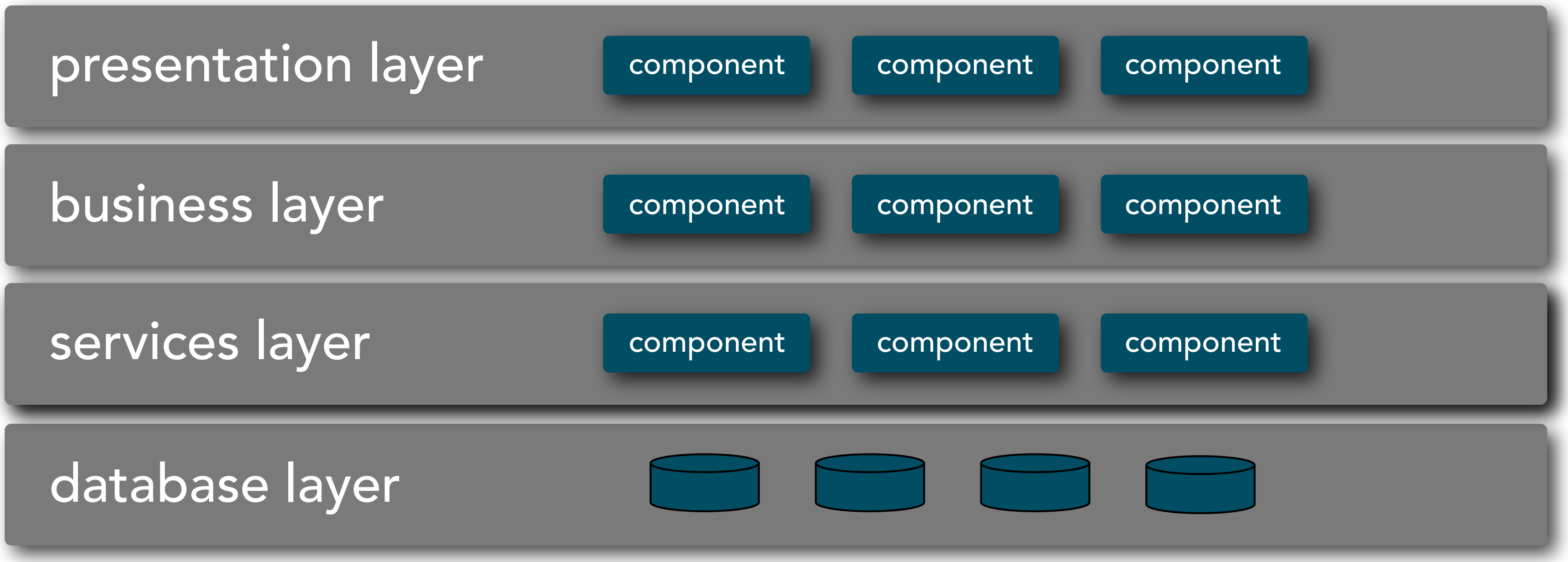
layered architecture



layers of isolation

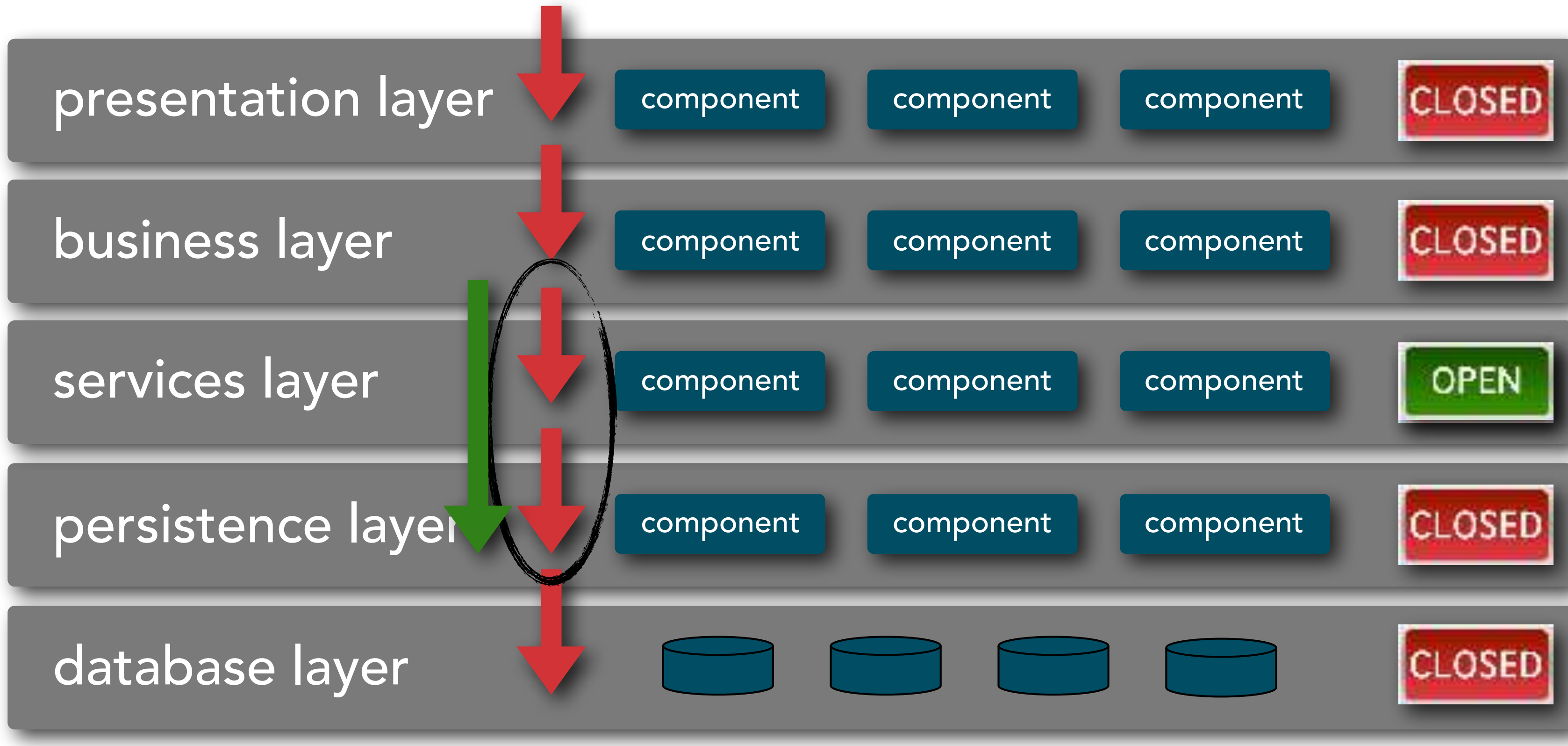
layered architecture

hybrids and variants

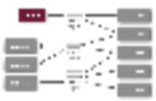
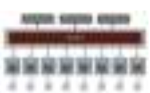
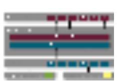


layered architecture

hybrids and variants



layered architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
							
							
							
							
							
							
							
							

architecture pattern roadmap

layered
architecture

microkernel
architecture

event-driven
architecture

pipeline
architecture

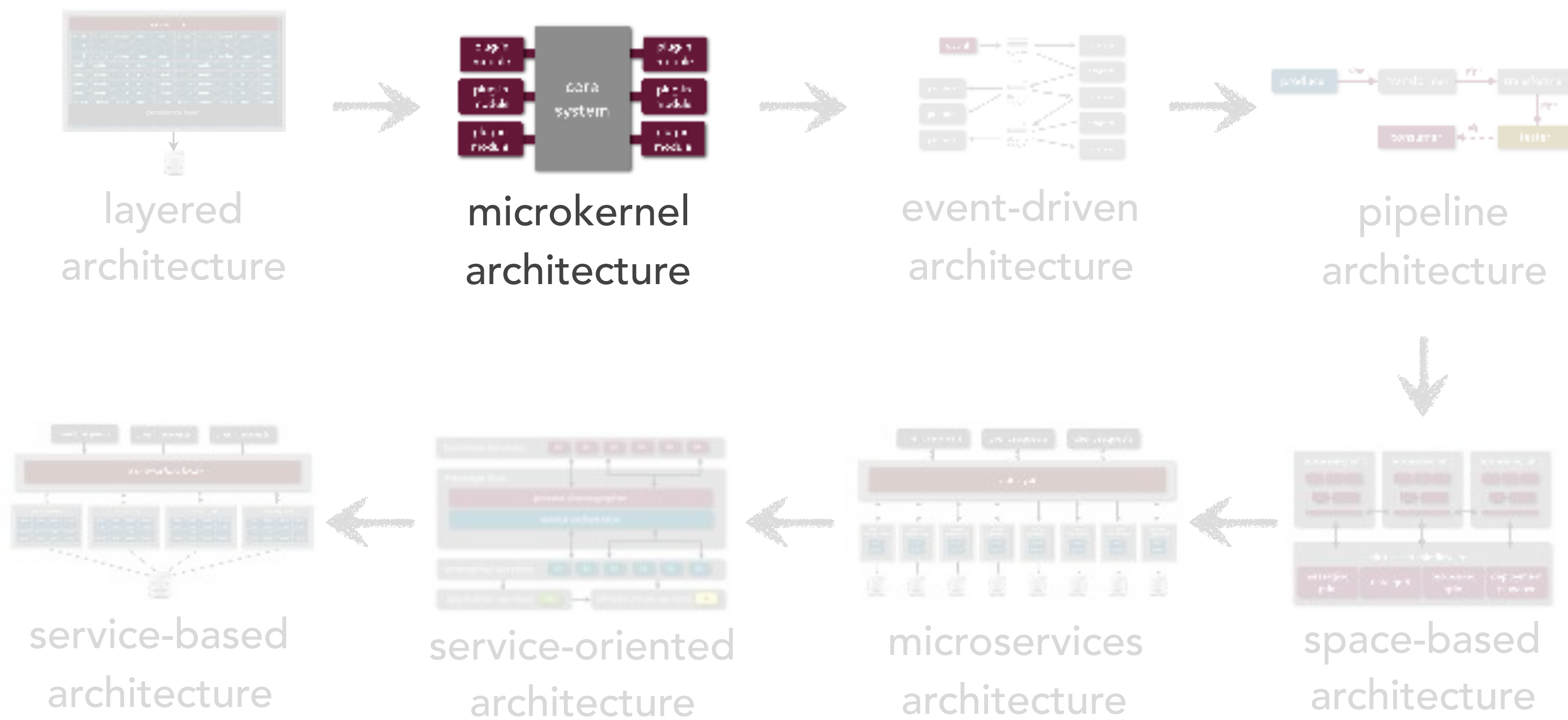
space-based
architecture

microservices
architecture

service-oriented
architecture

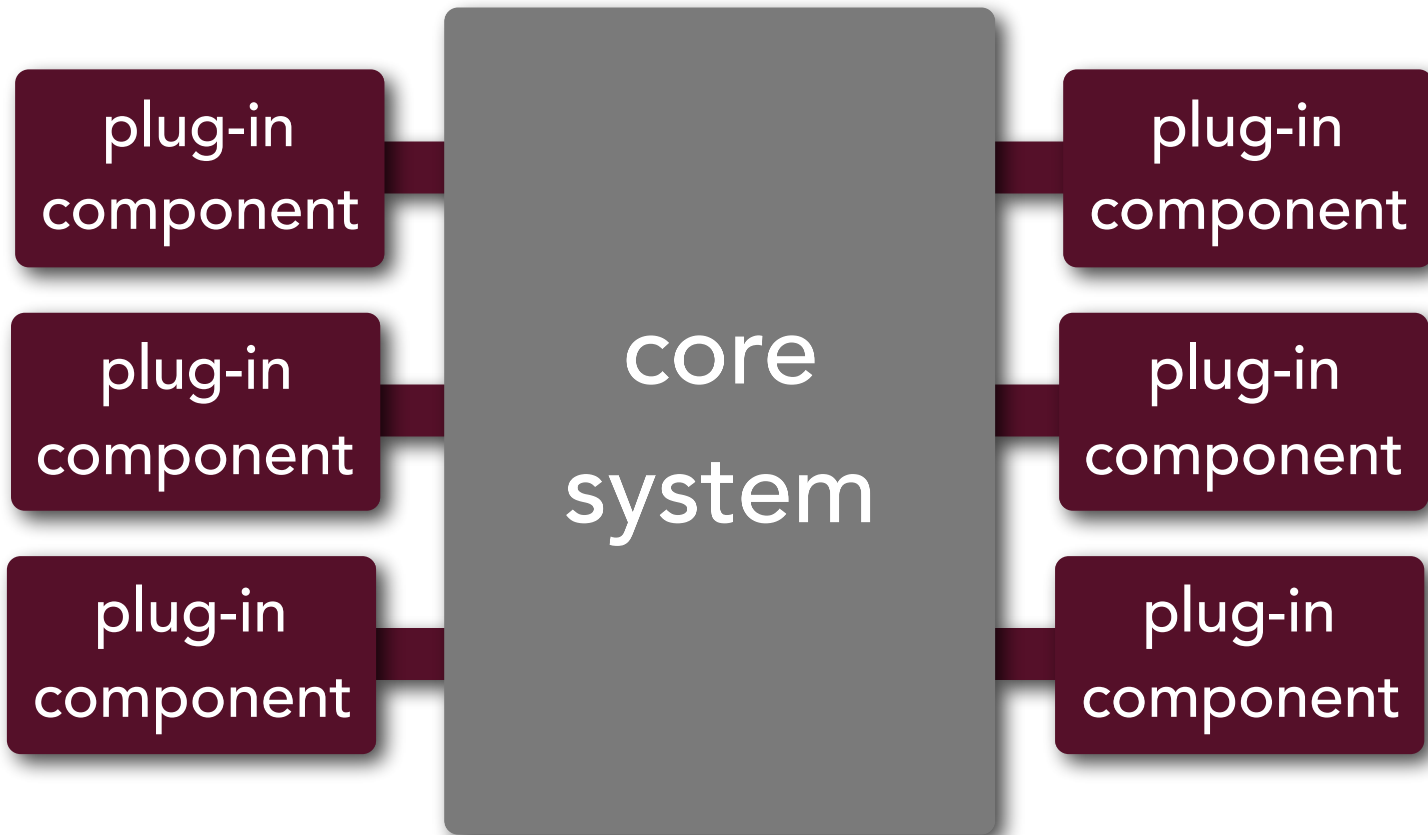
service-based
architecture

architecture pattern roadmap



microkernel architecture

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



microkernel architecture

architectural components

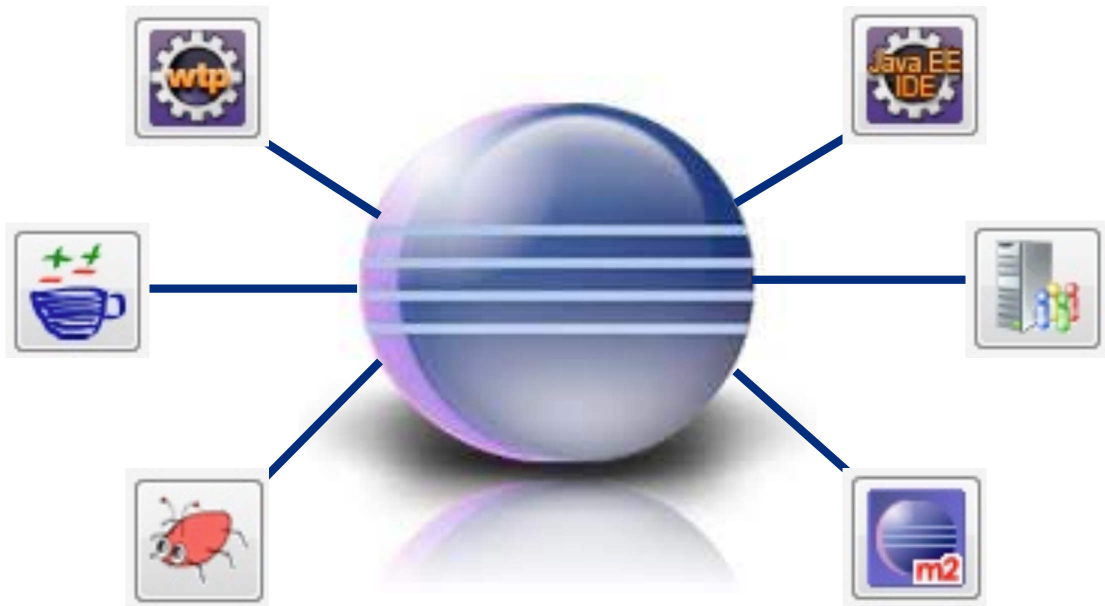
core system

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

plug-in module

- standalone independent module
- specific additional rules or logic

microkernel architecture

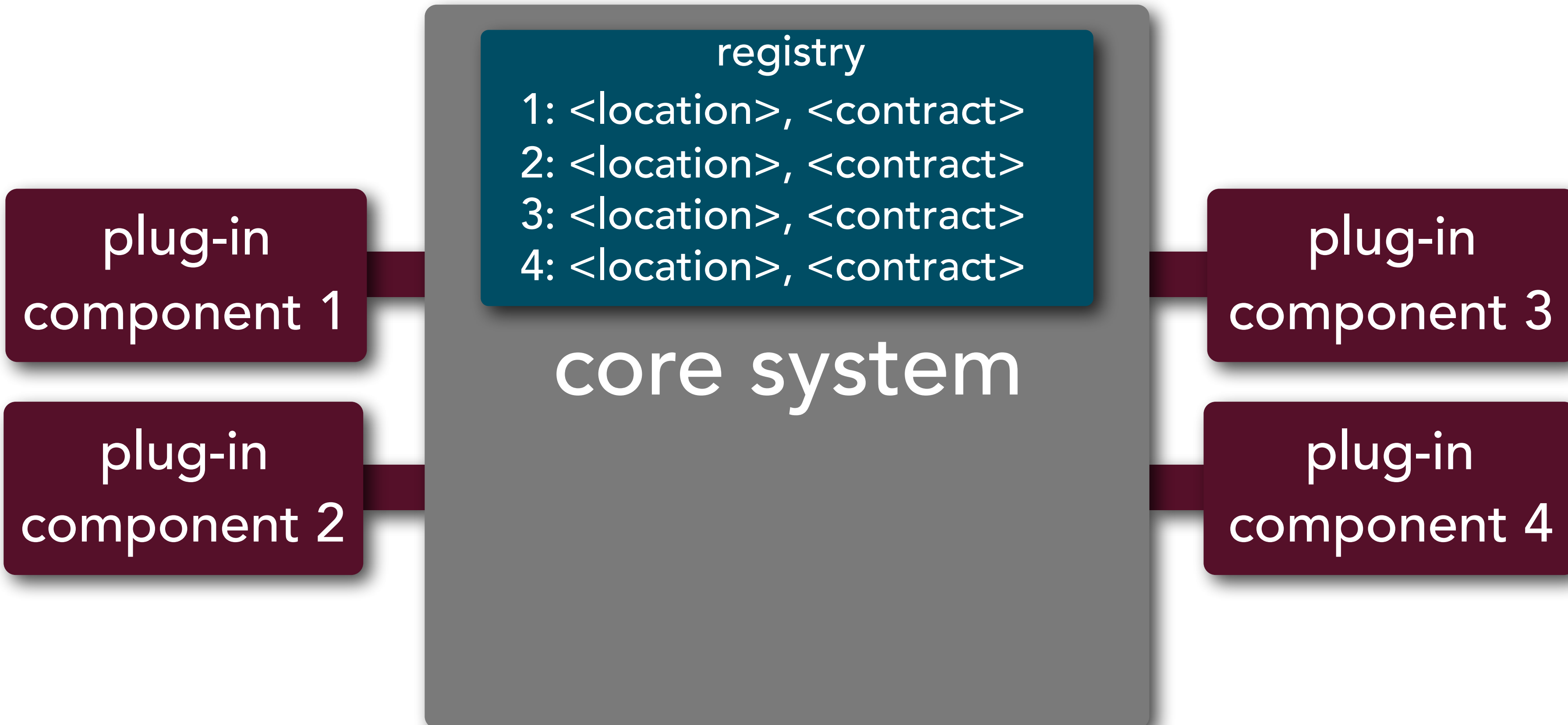


microkernel architecture

claims processing

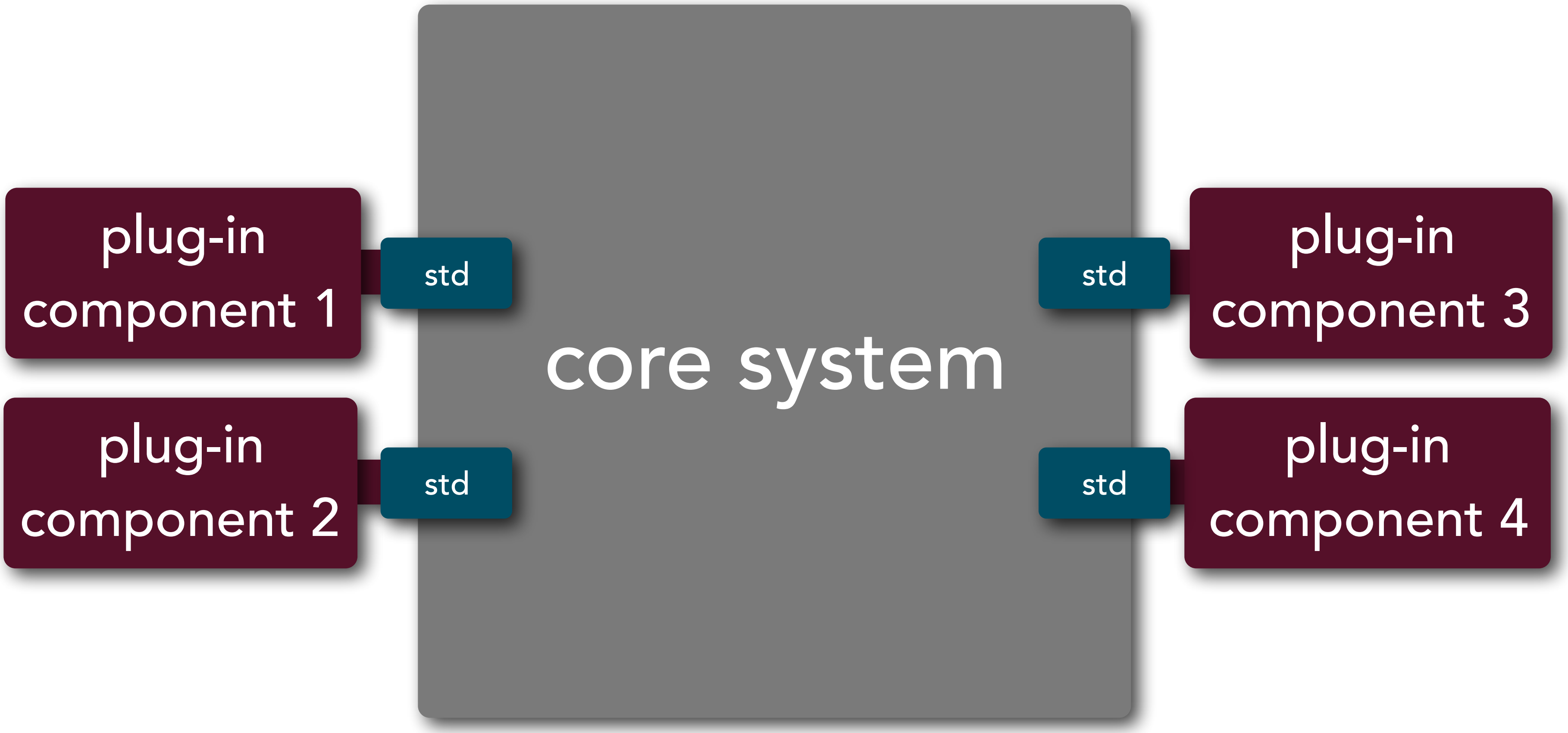
microkernel architecture

registry

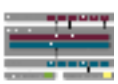


microkernel architecture

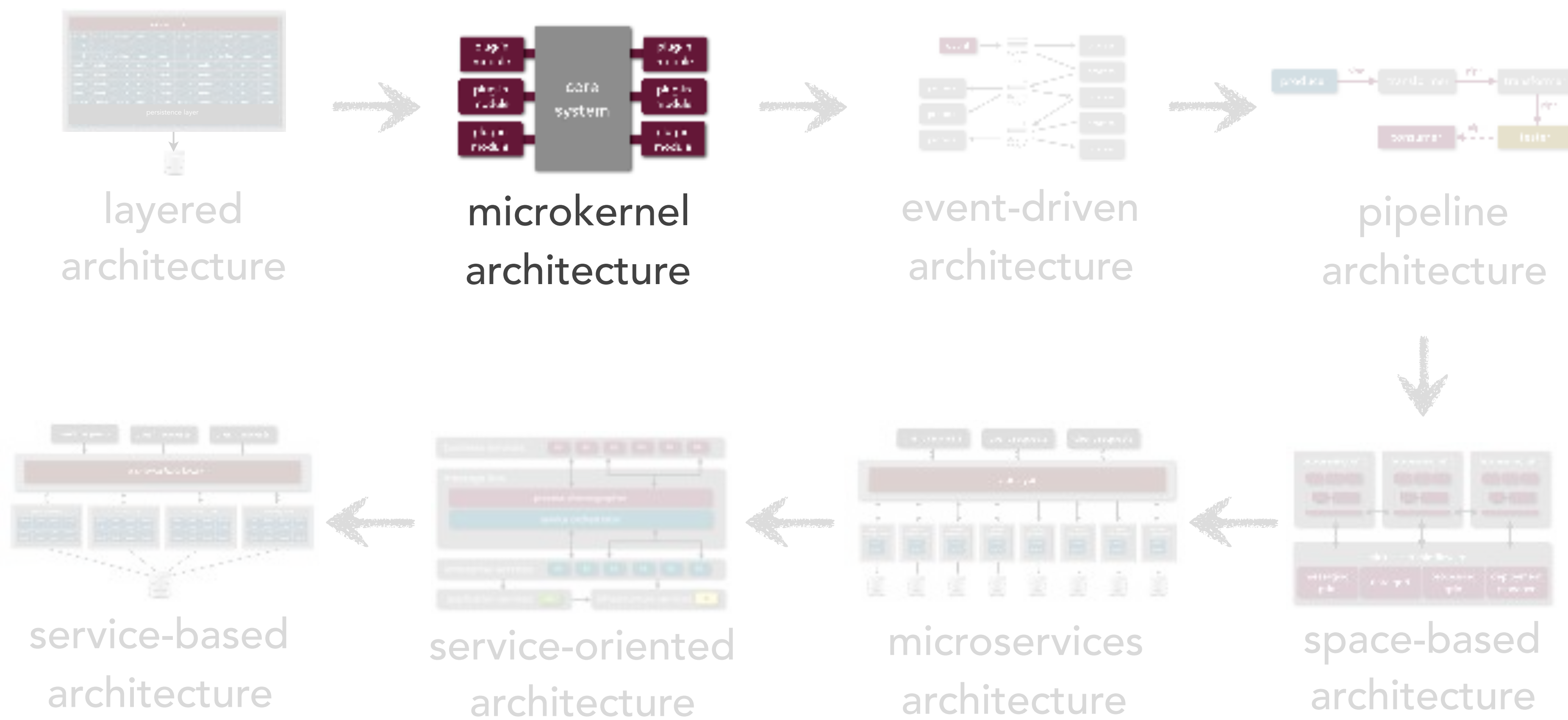
plug-in contracts



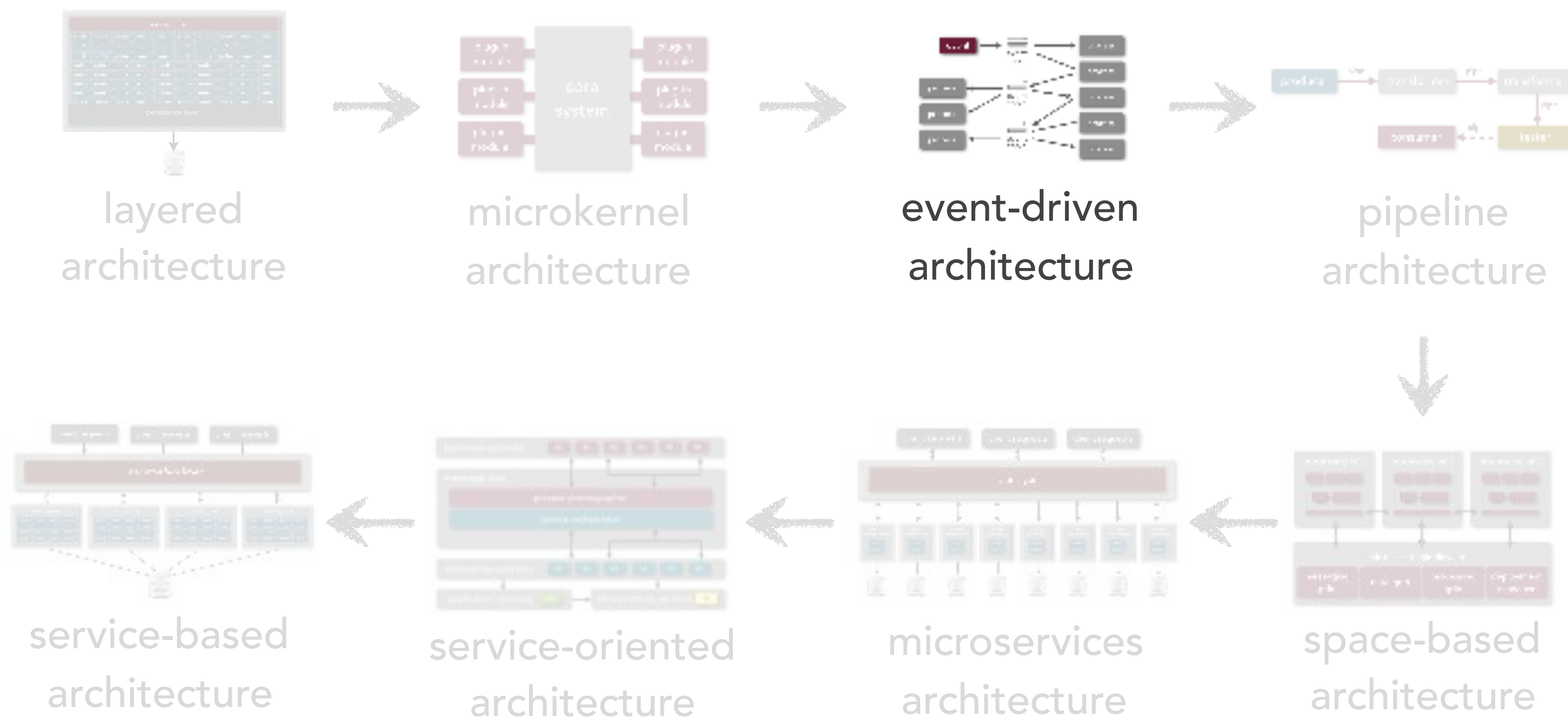
microkernel architecture

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

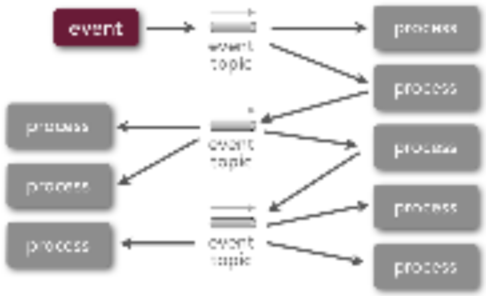
architecture pattern roadmap



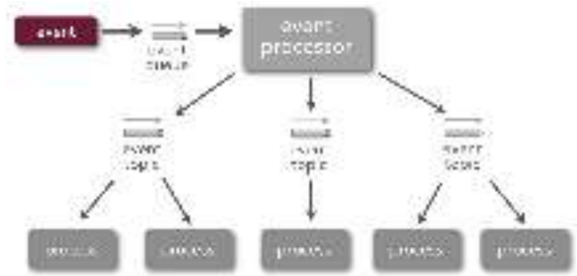
architecture pattern roadmap



event-driven architecture



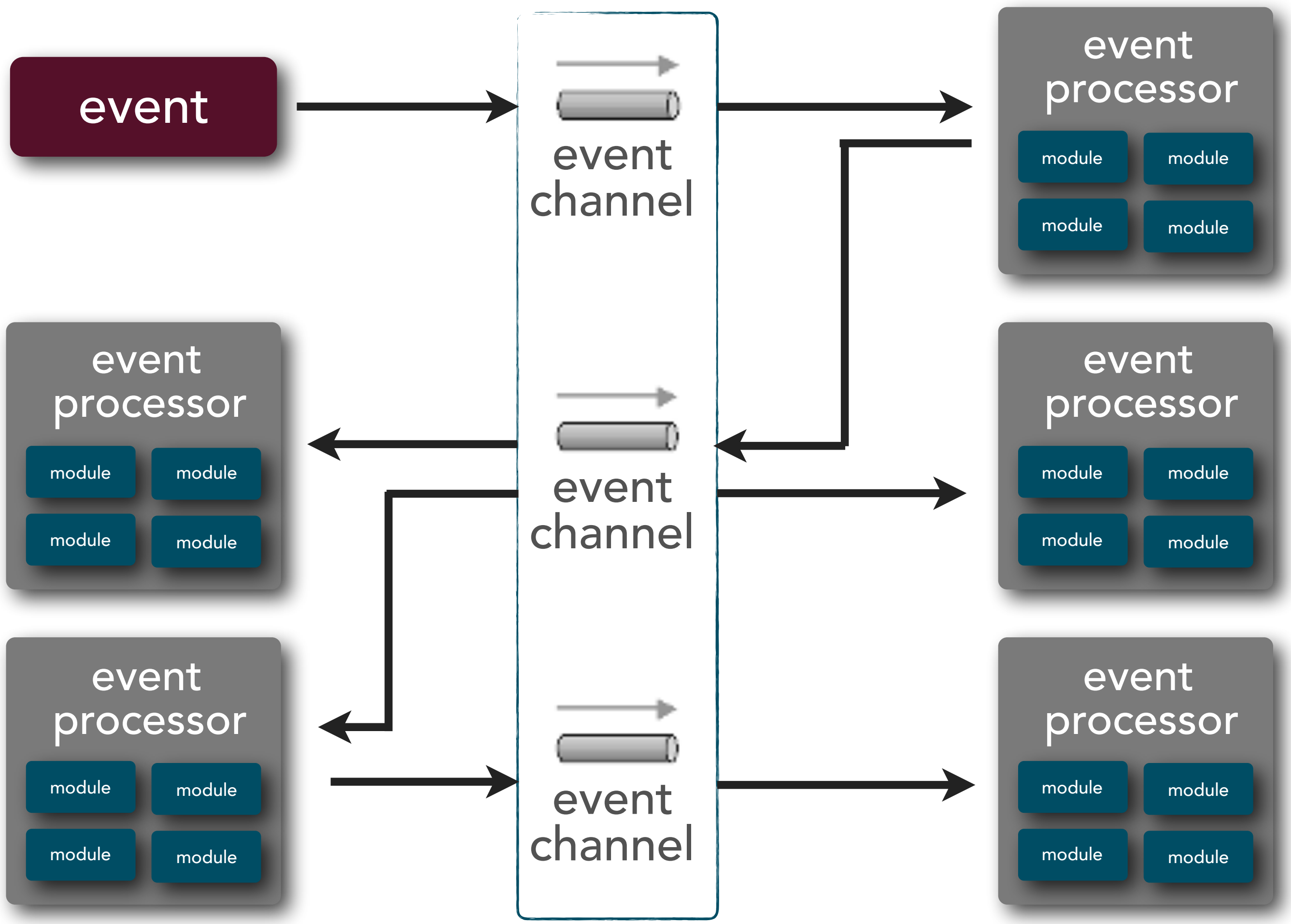
broker topology



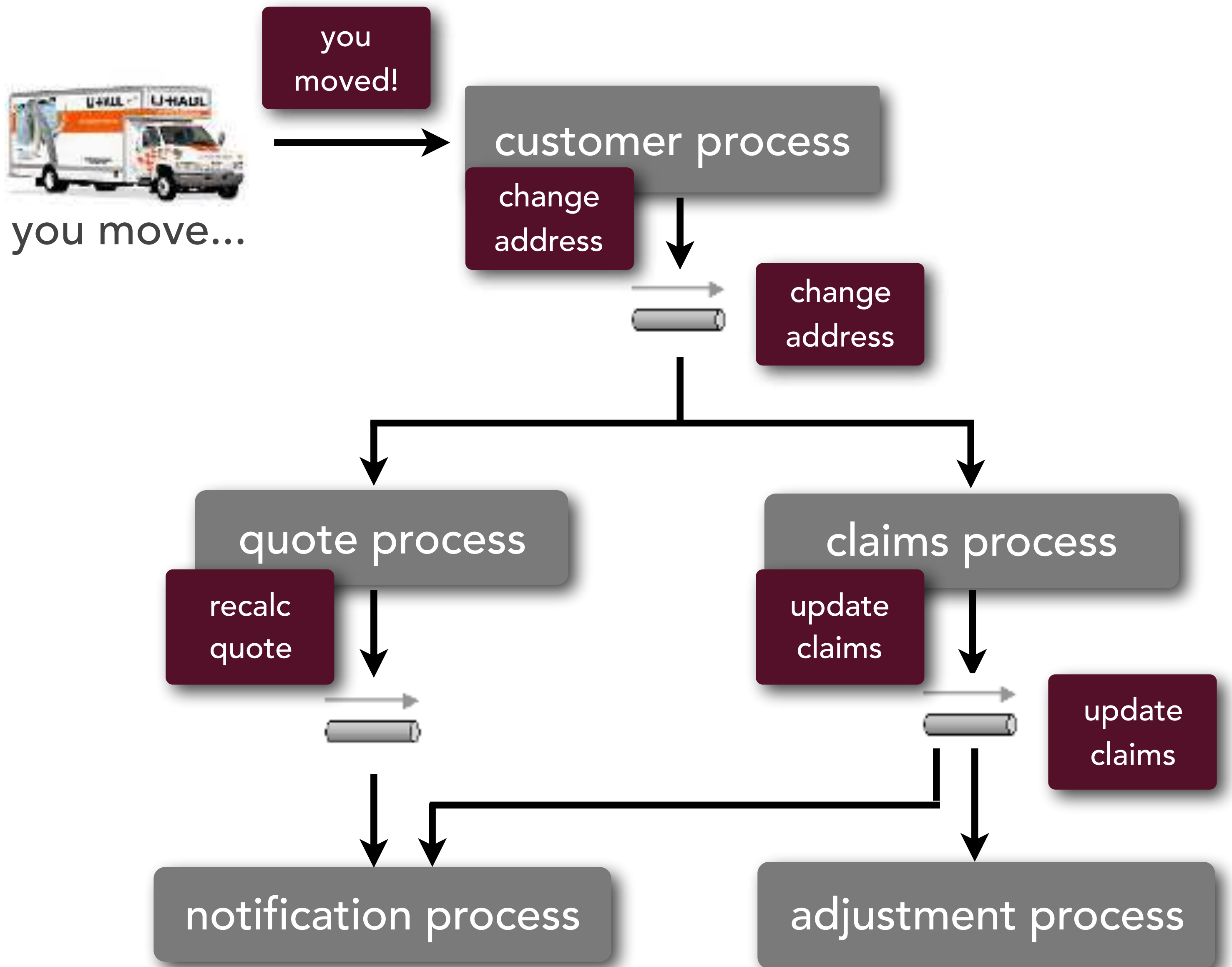
mediator topology

event-driven architecture

broker topology

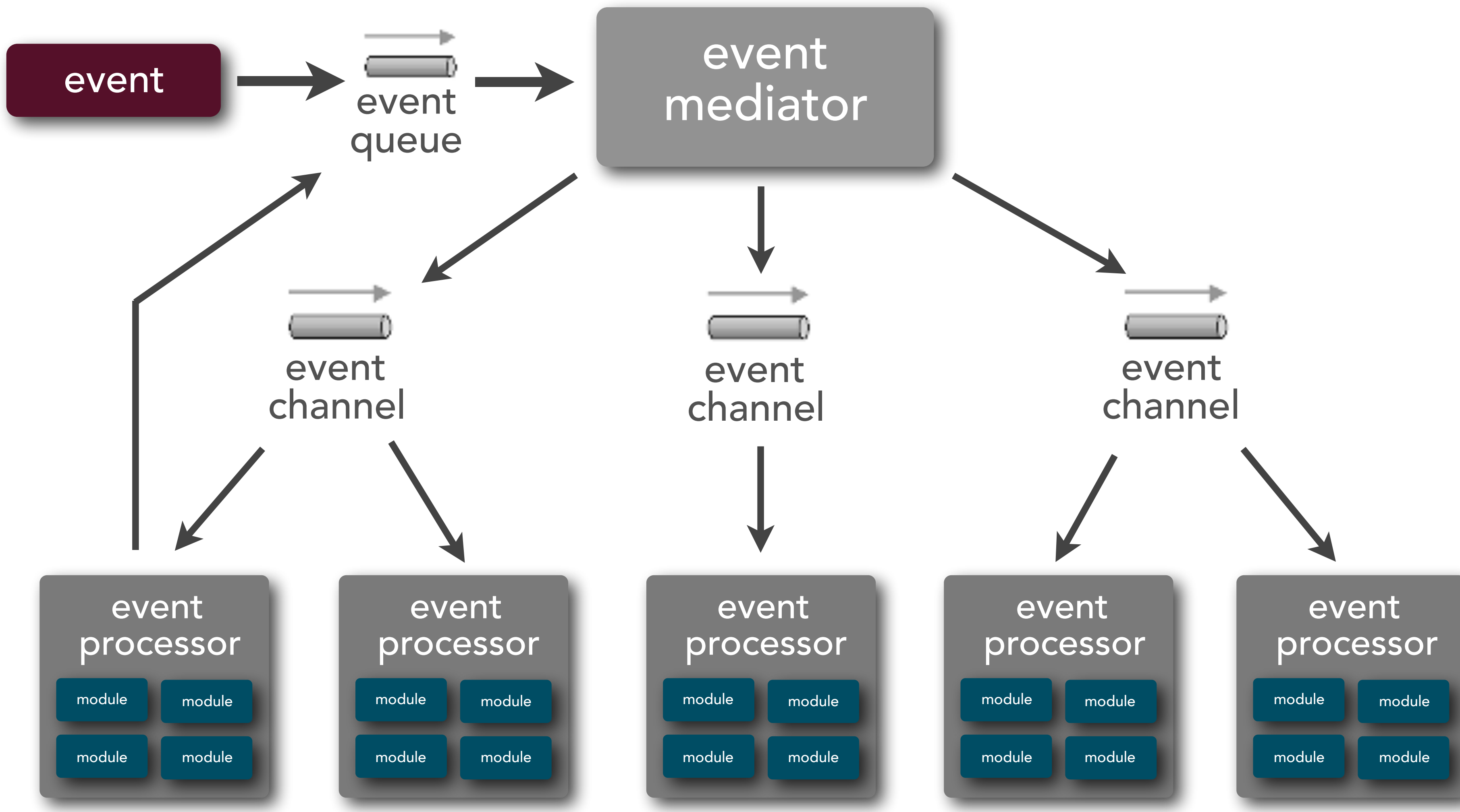


event-driven architecture

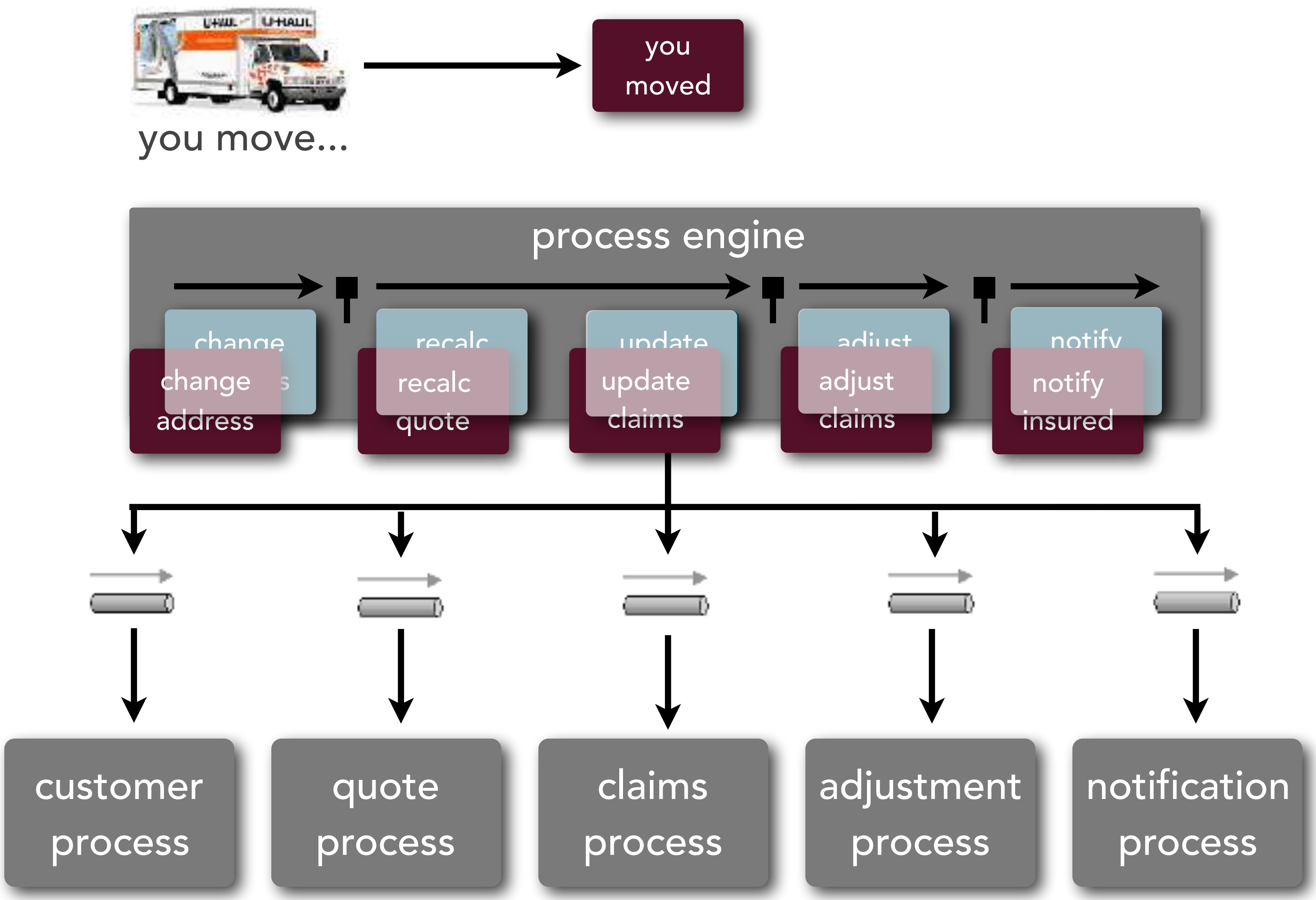


event-driven architecture

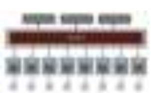
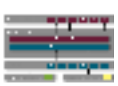
mediator topology



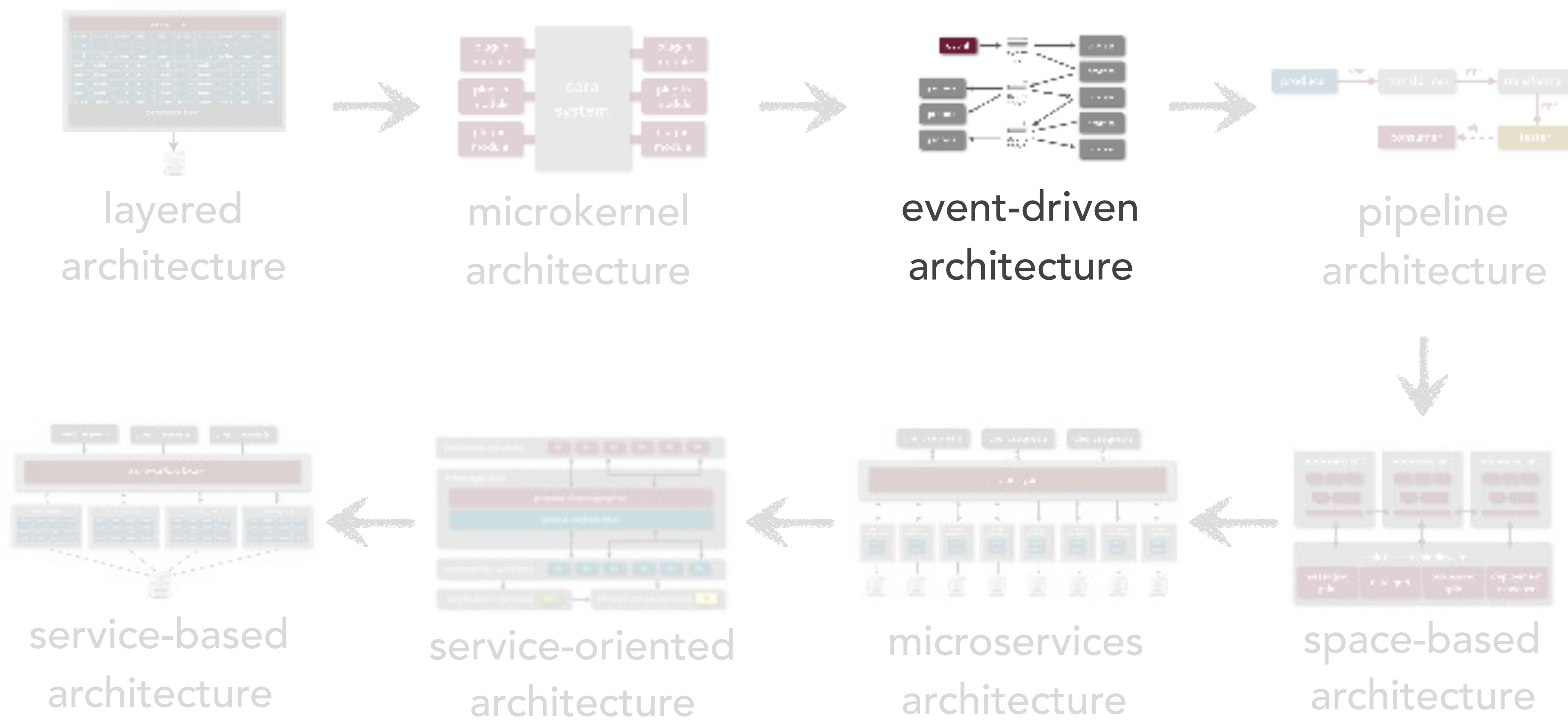
event-driven architecture



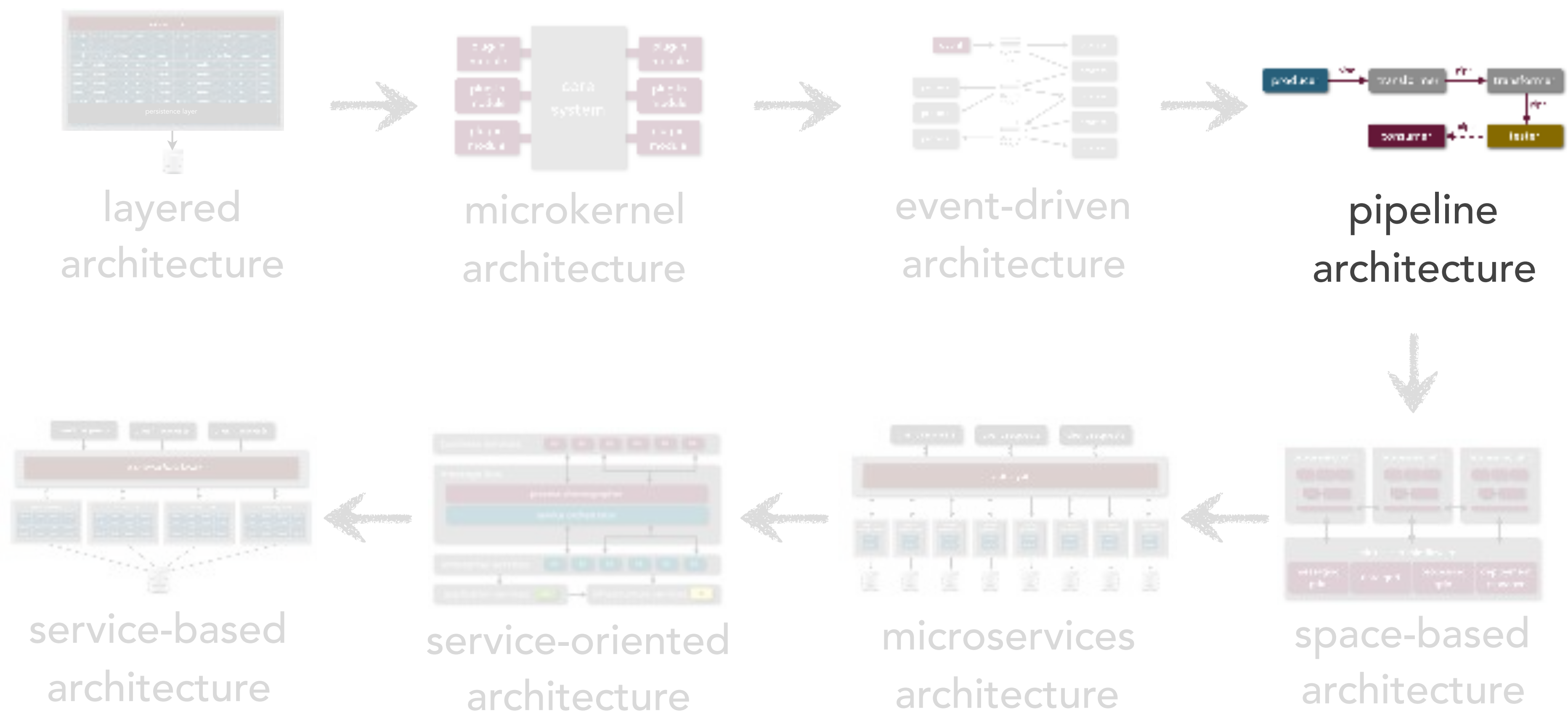
event-driven architecture

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

architecture pattern roadmap

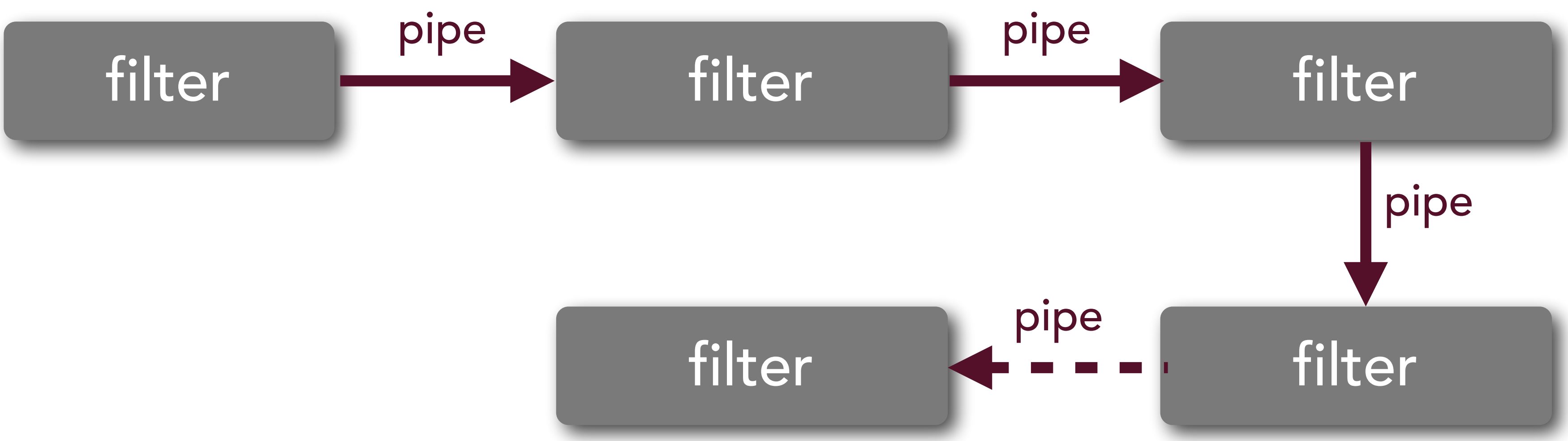


architecture pattern roadmap



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

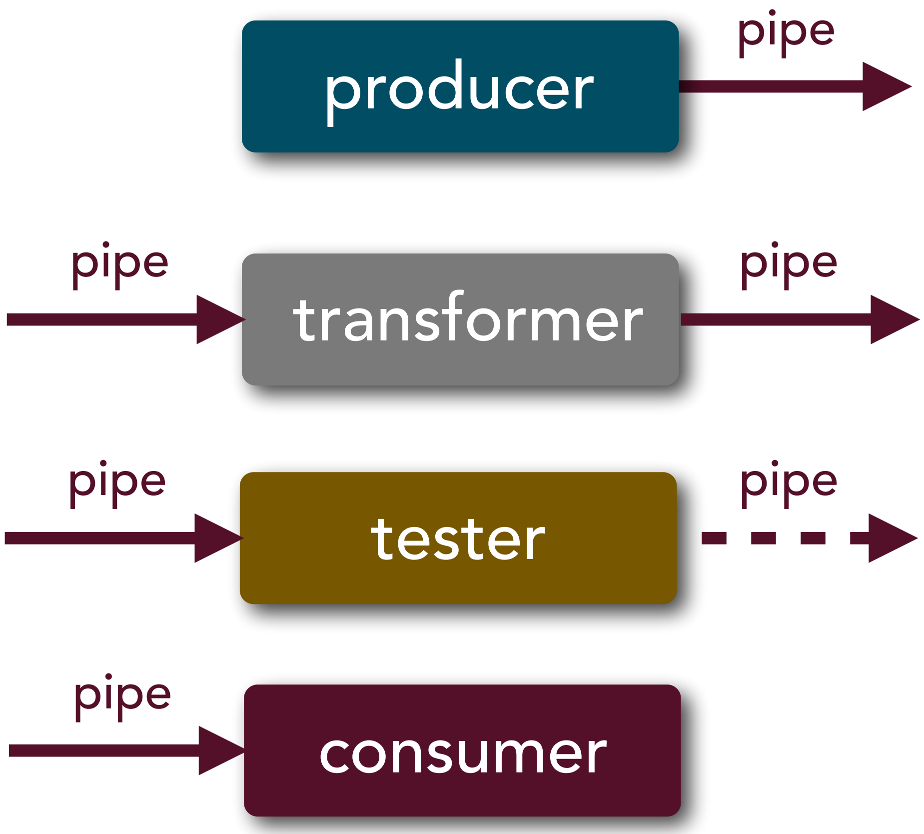
self-contained and independent from other filters

usually designed to perform a single specific task

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

pipeline architecture

filters



starting point, outbound only

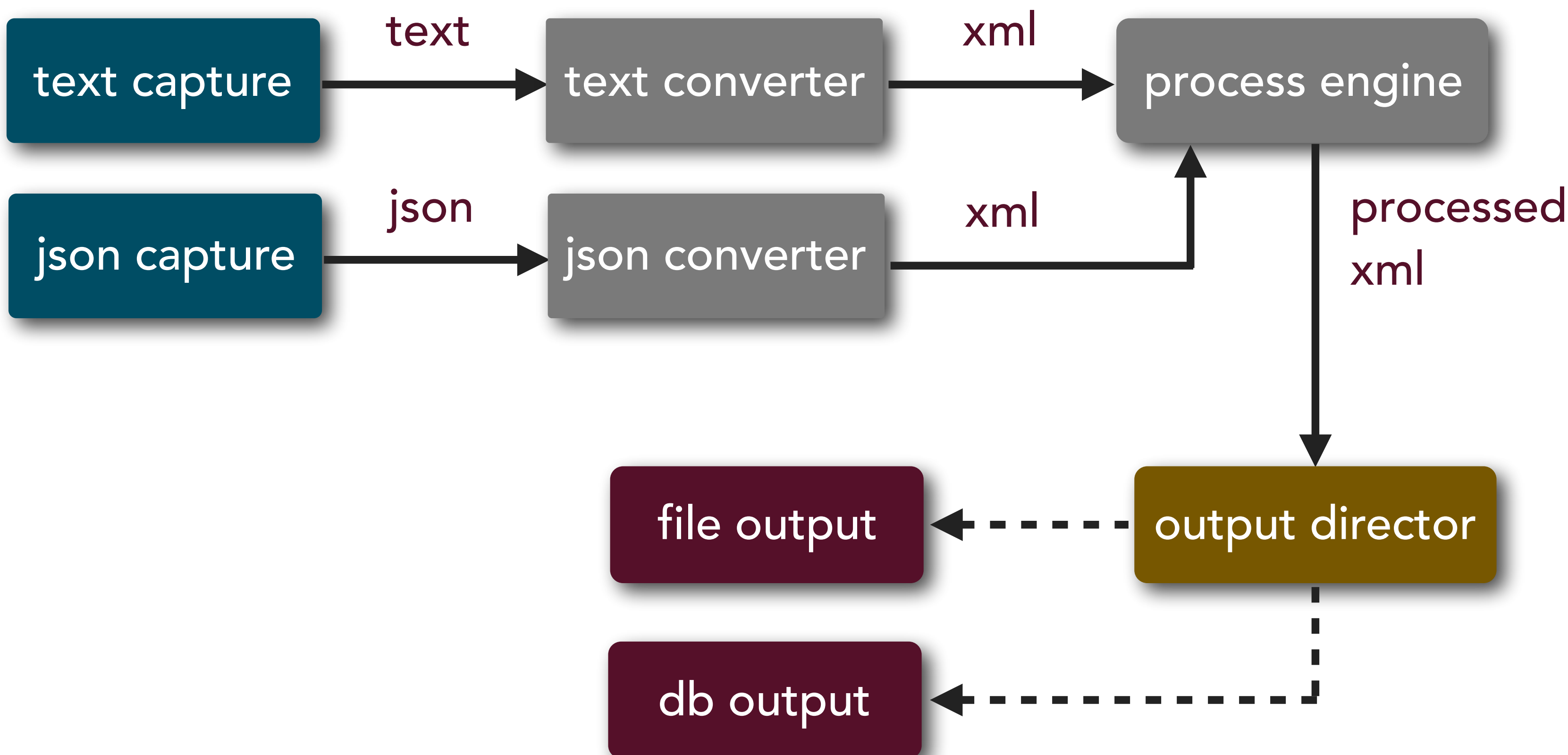
input, processing, output

input, discard or pass-thru

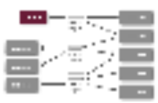
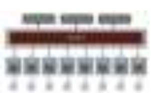
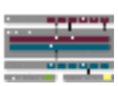
ending point, inbound only

pipeline architecture

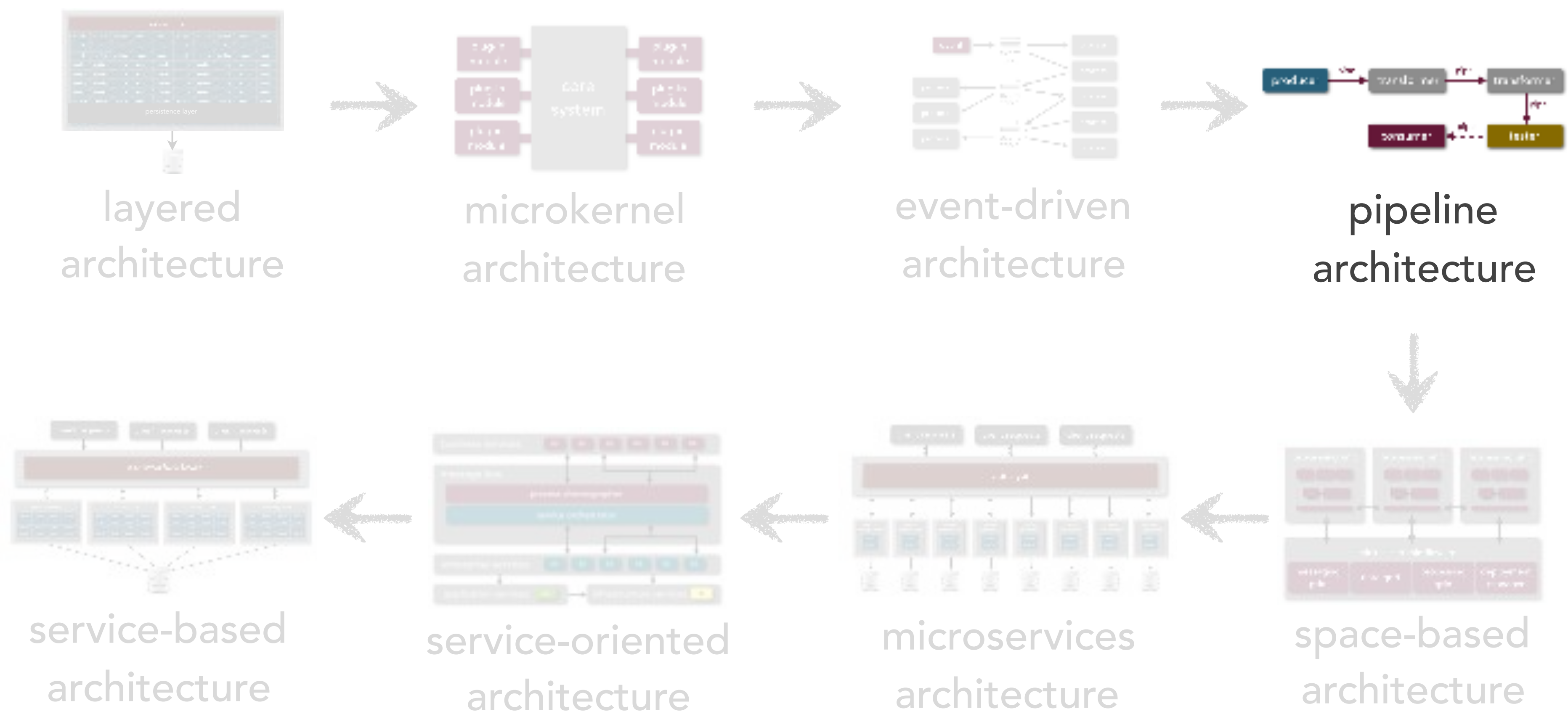
example: capture data in multiple formats, process the data, and send to multiple outputs



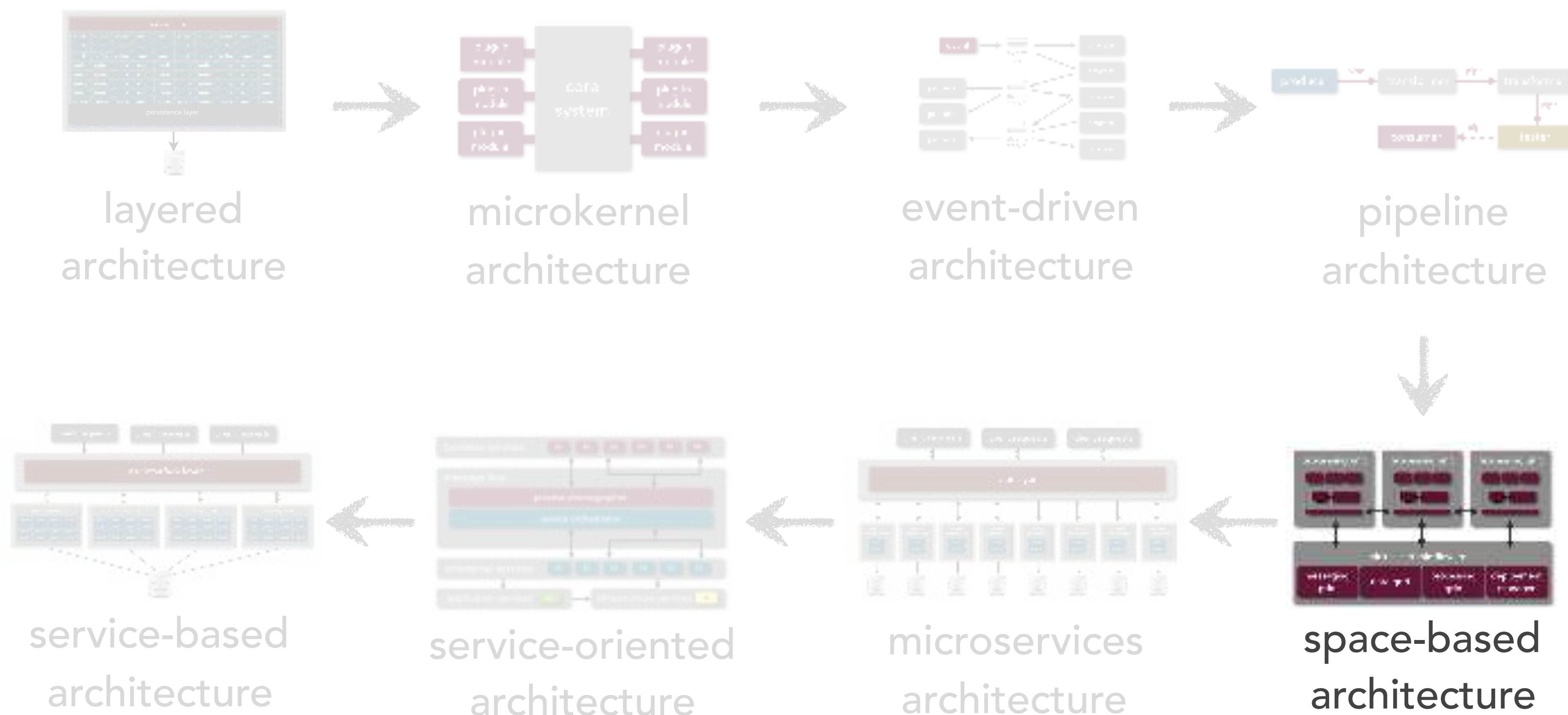
pipeline architecture

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

architecture pattern roadmap

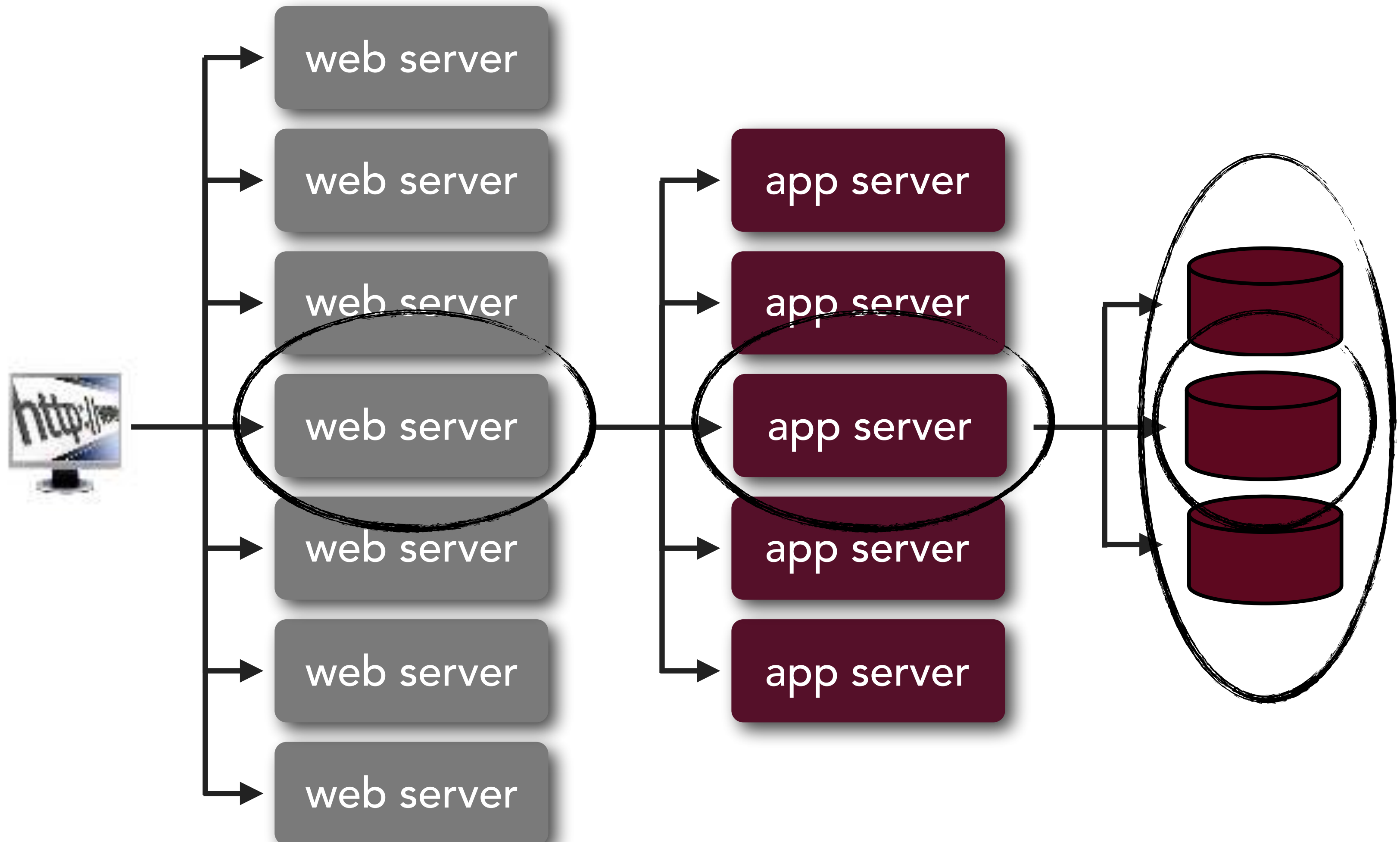


architecture pattern roadmap

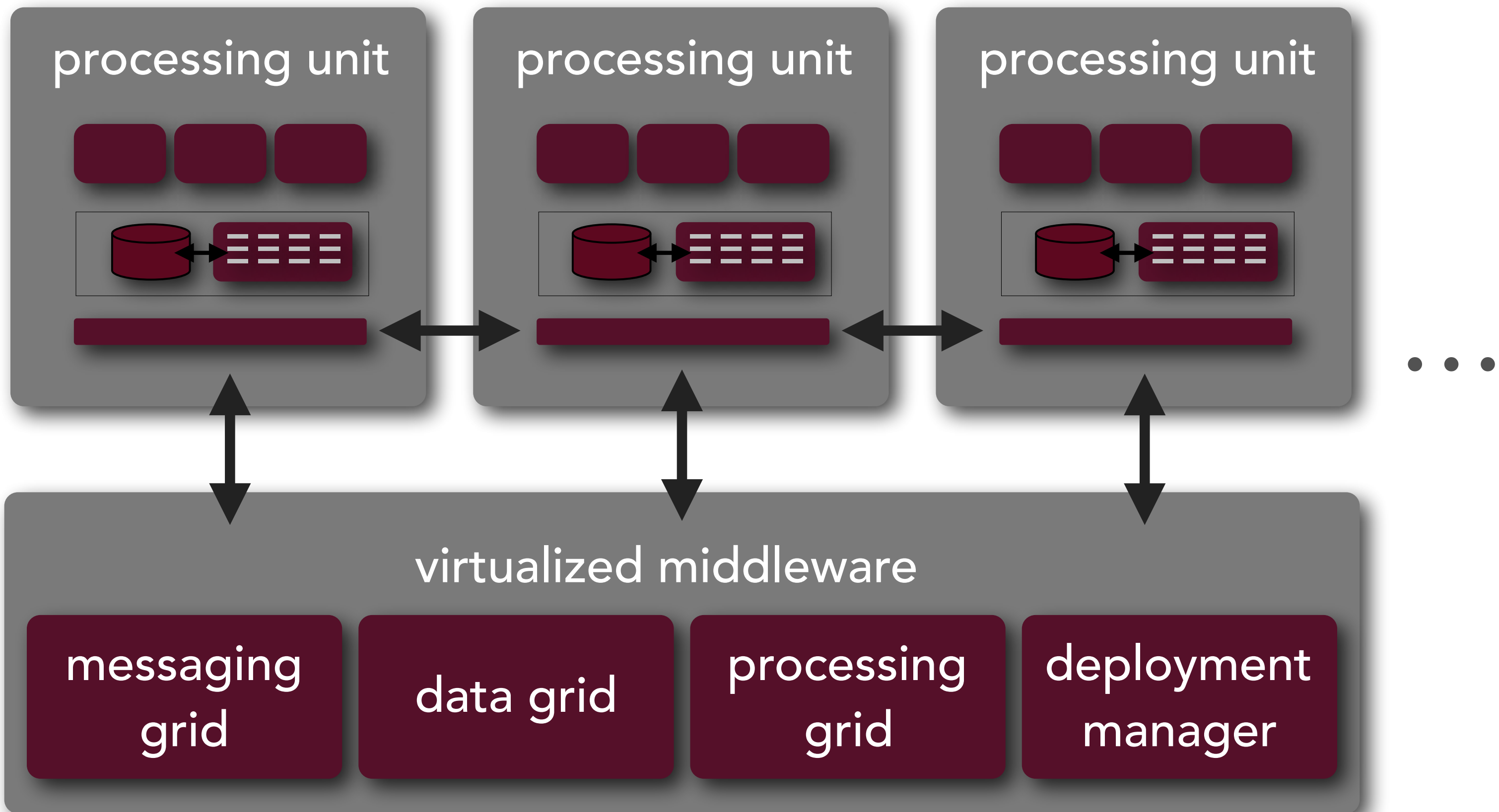


space-based architecture

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

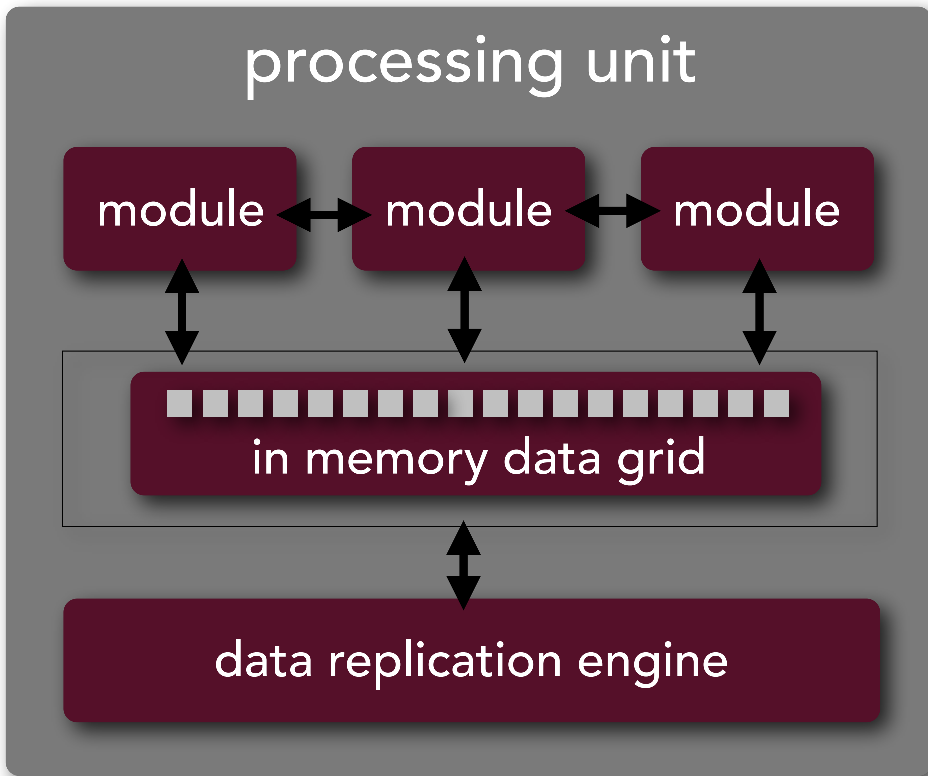


space-based architecture



space-based architecture

processing unit



space-based architecture

middleware

messaging grid

data grid

processing grid

deployment manager

space-based architecture

middleware

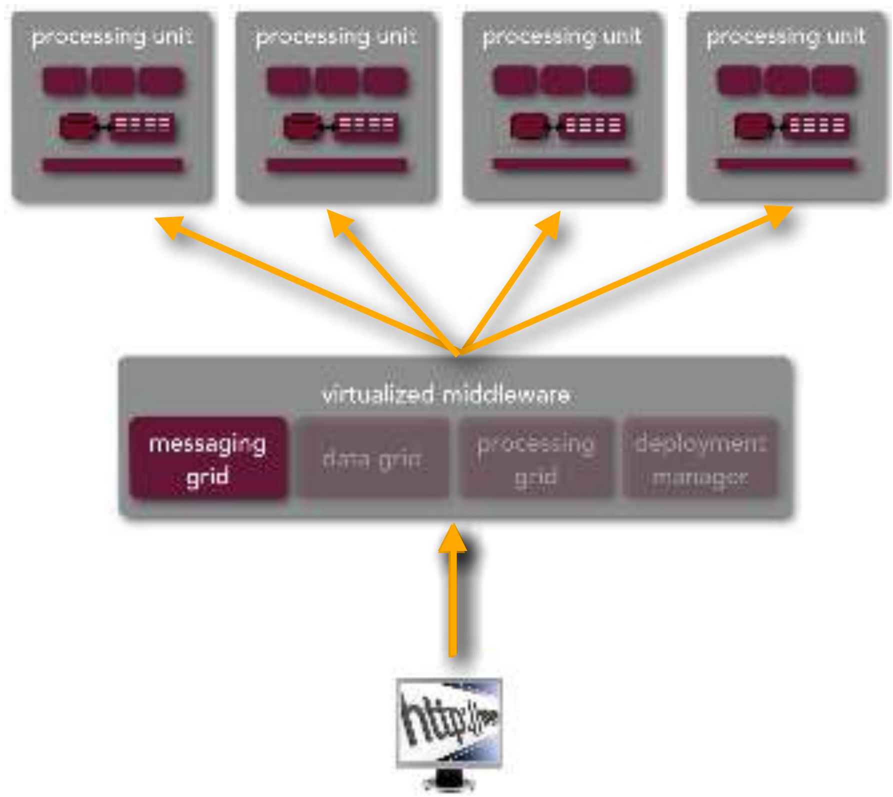
messaging grid

data grid

processing grid

deployment manager

manages input request and session



space-based architecture

middleware

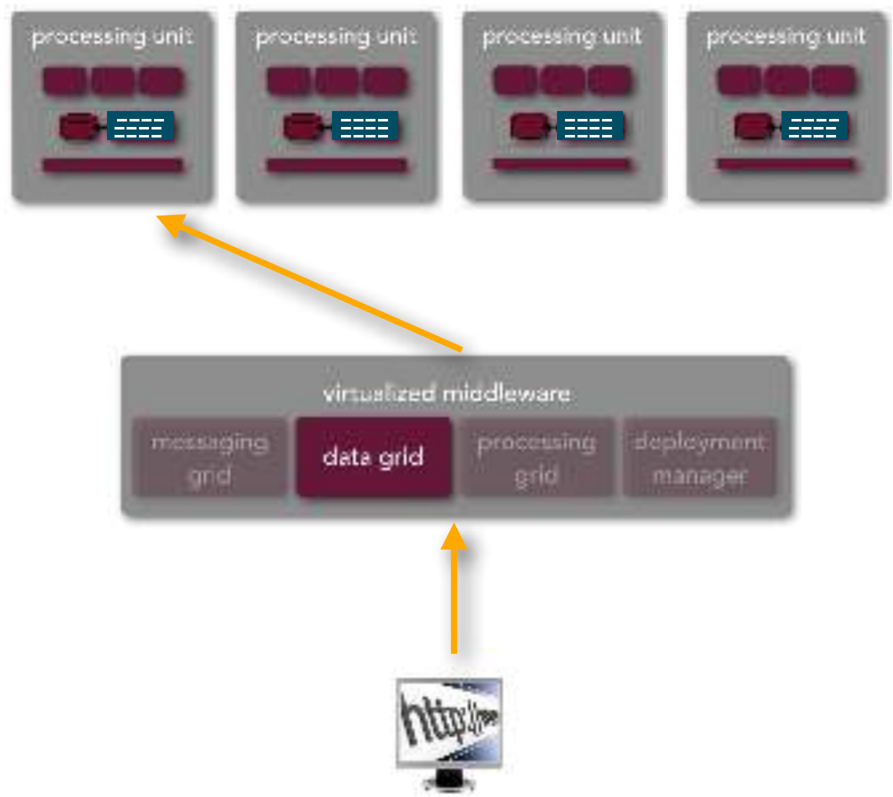
manages data replication between processing units

messaging grid

data grid

processing grid

deployment manager



space-based architecture

middleware

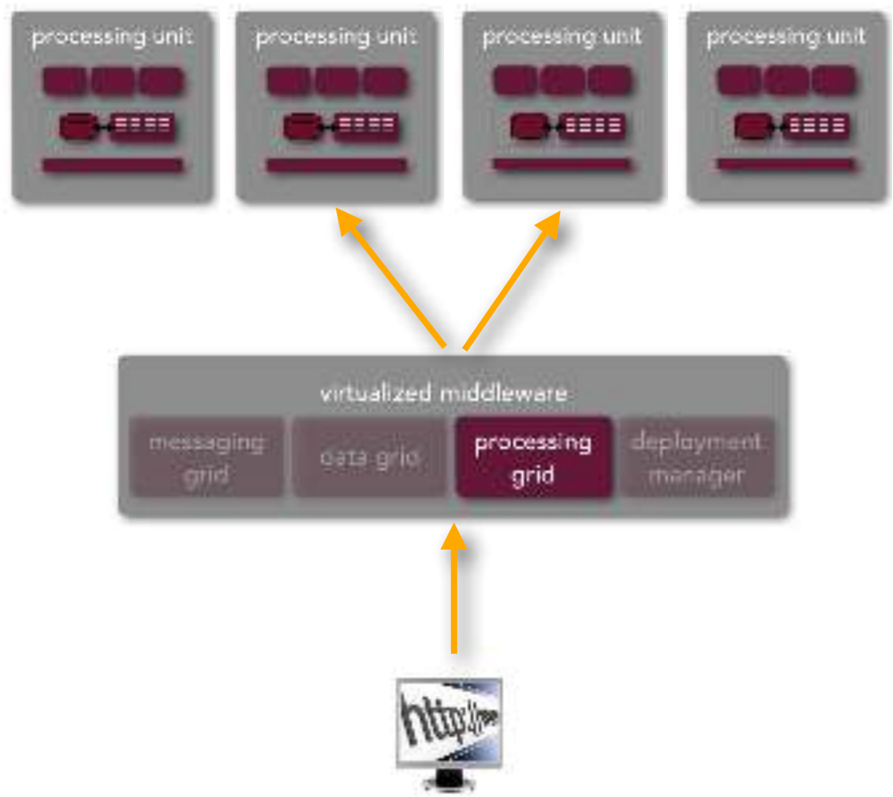
messaging grid

data grid

processing grid

deployment manager

manages distributed request processing



space-based architecture

middleware

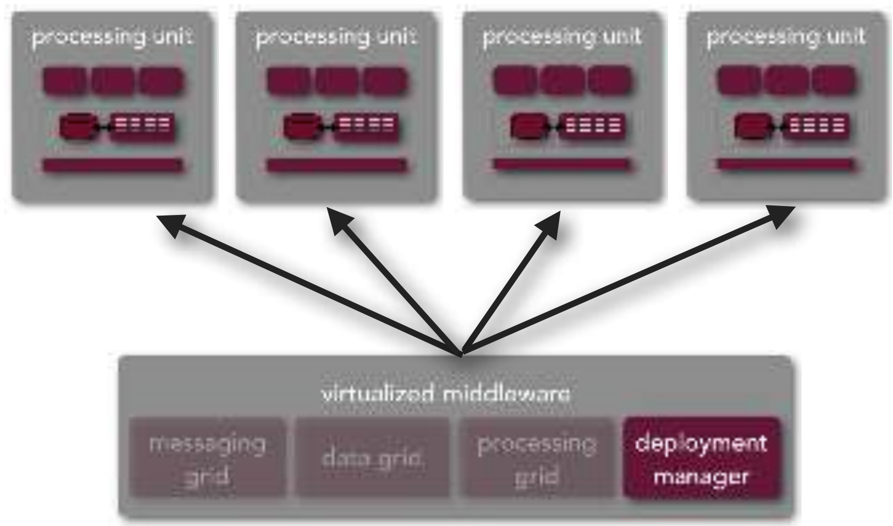
messaging grid

data grid

processing grid

deployment manager

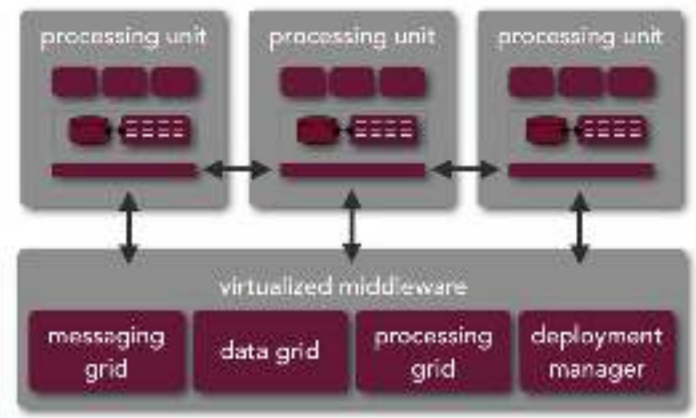
manages dynamic processing unit deployment



space-based architecture

product implementations

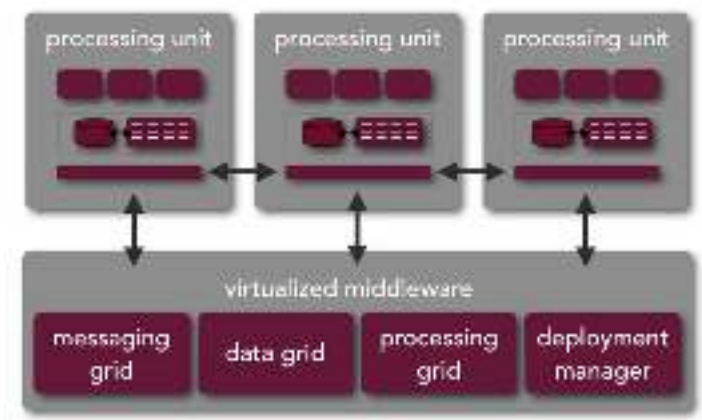
- javaspaces
- gigaspaces
- ibm object grid
- gemfire
- ncache
- oracle coherence



space-based architecture

it's all about variable scalability...

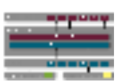
good for applications that have variable load or inconsistent peak times



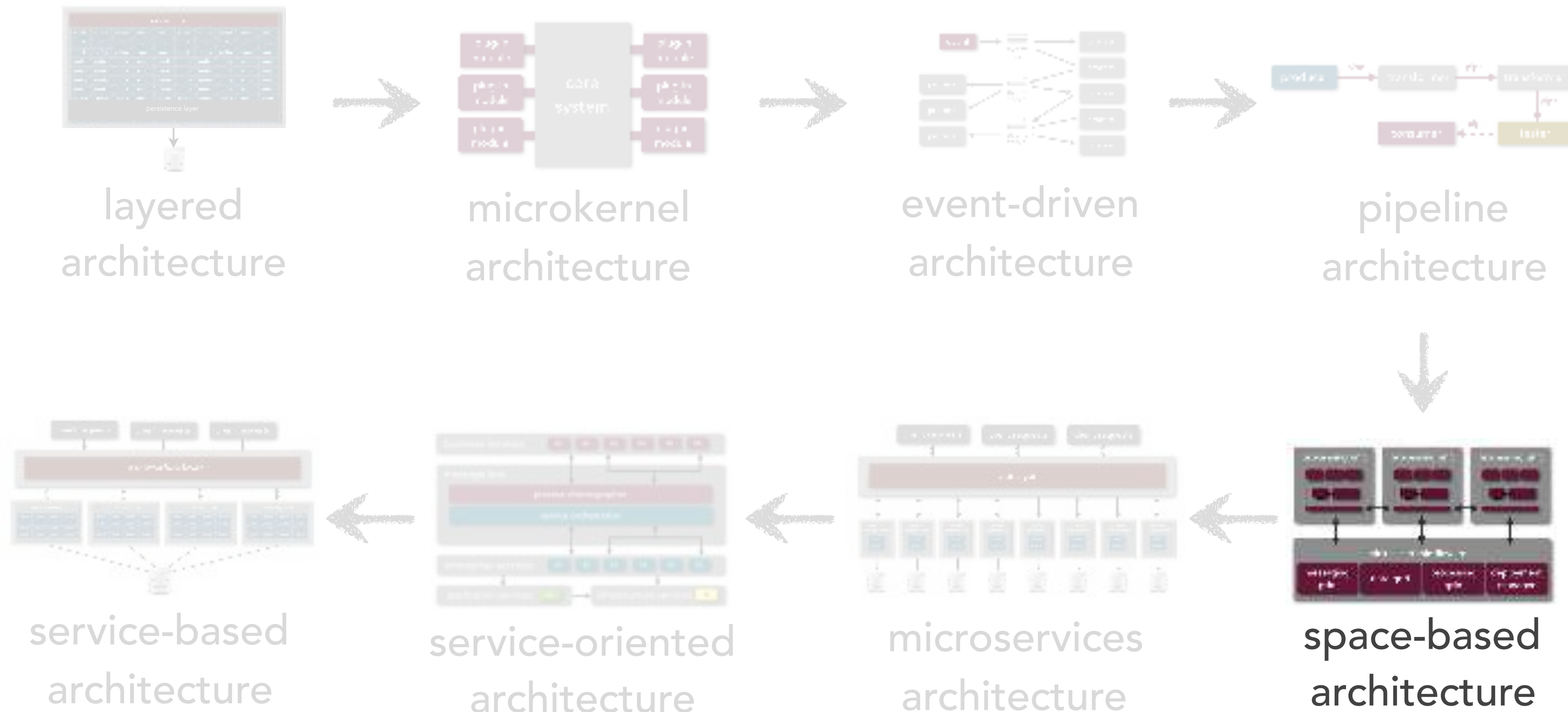
not a good fit for traditional large-scale relational database systems

relatively complex and expensive pattern to implement

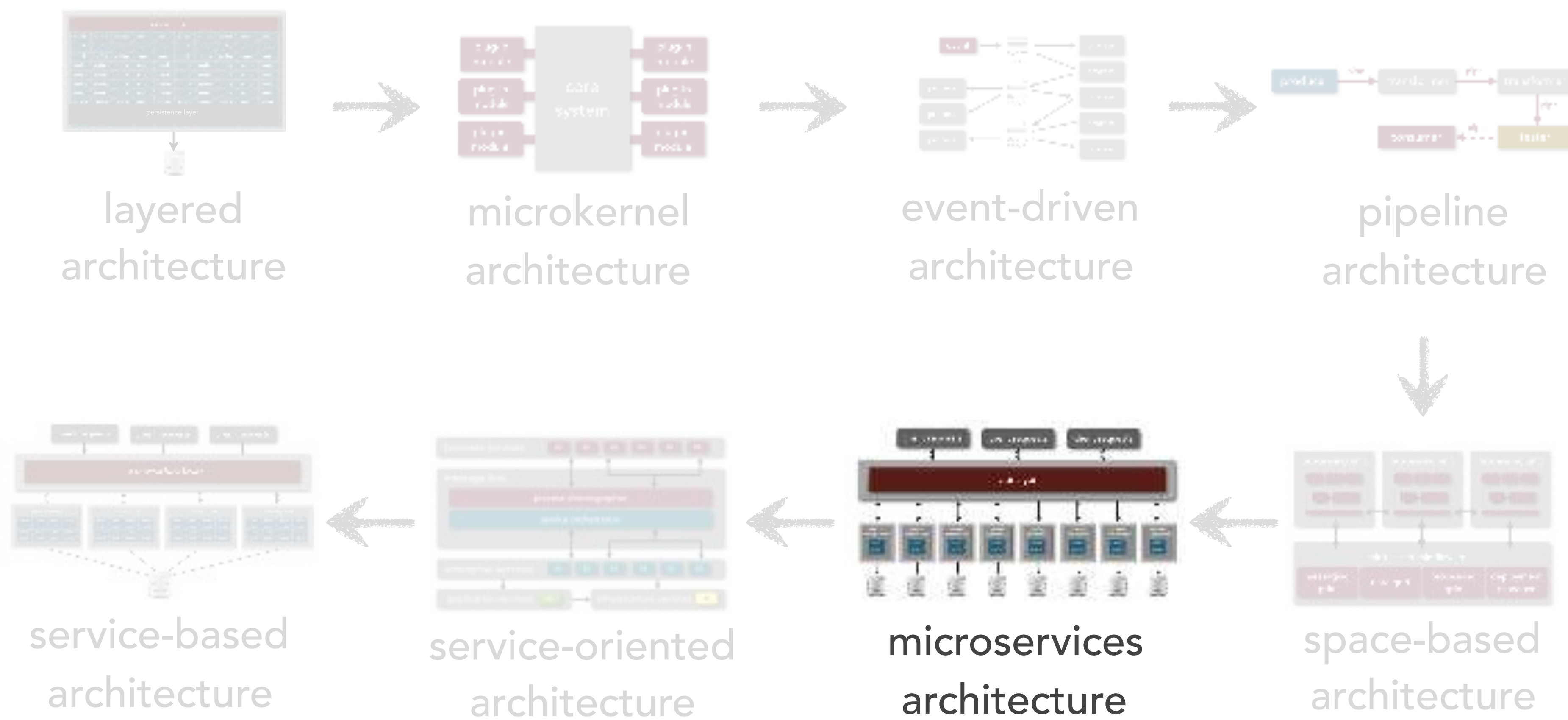
space-based architecture

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

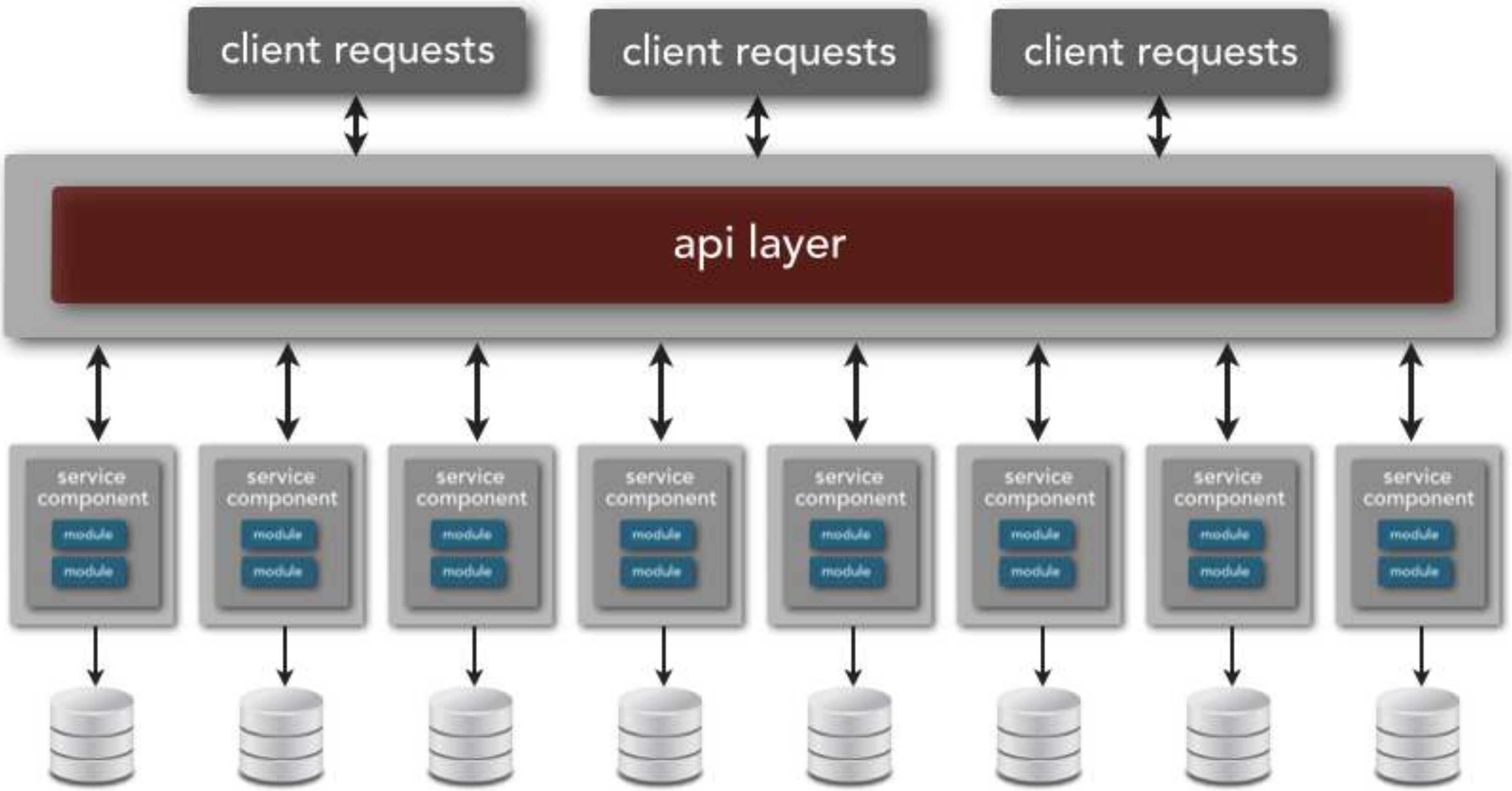
architecture pattern roadmap



architecture pattern roadmap

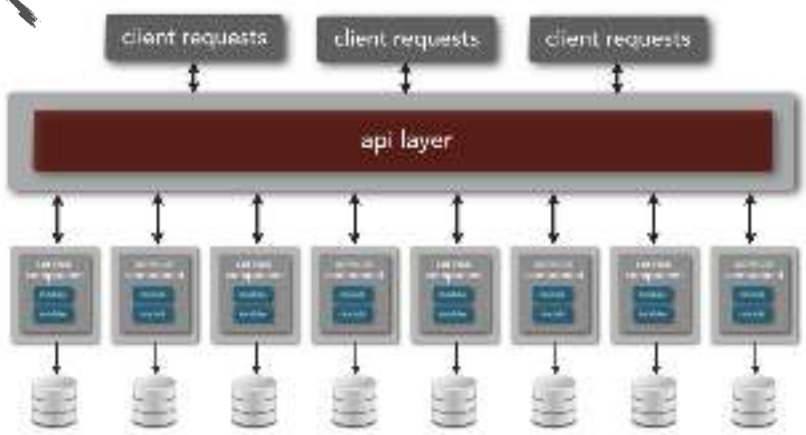


microservices architecture

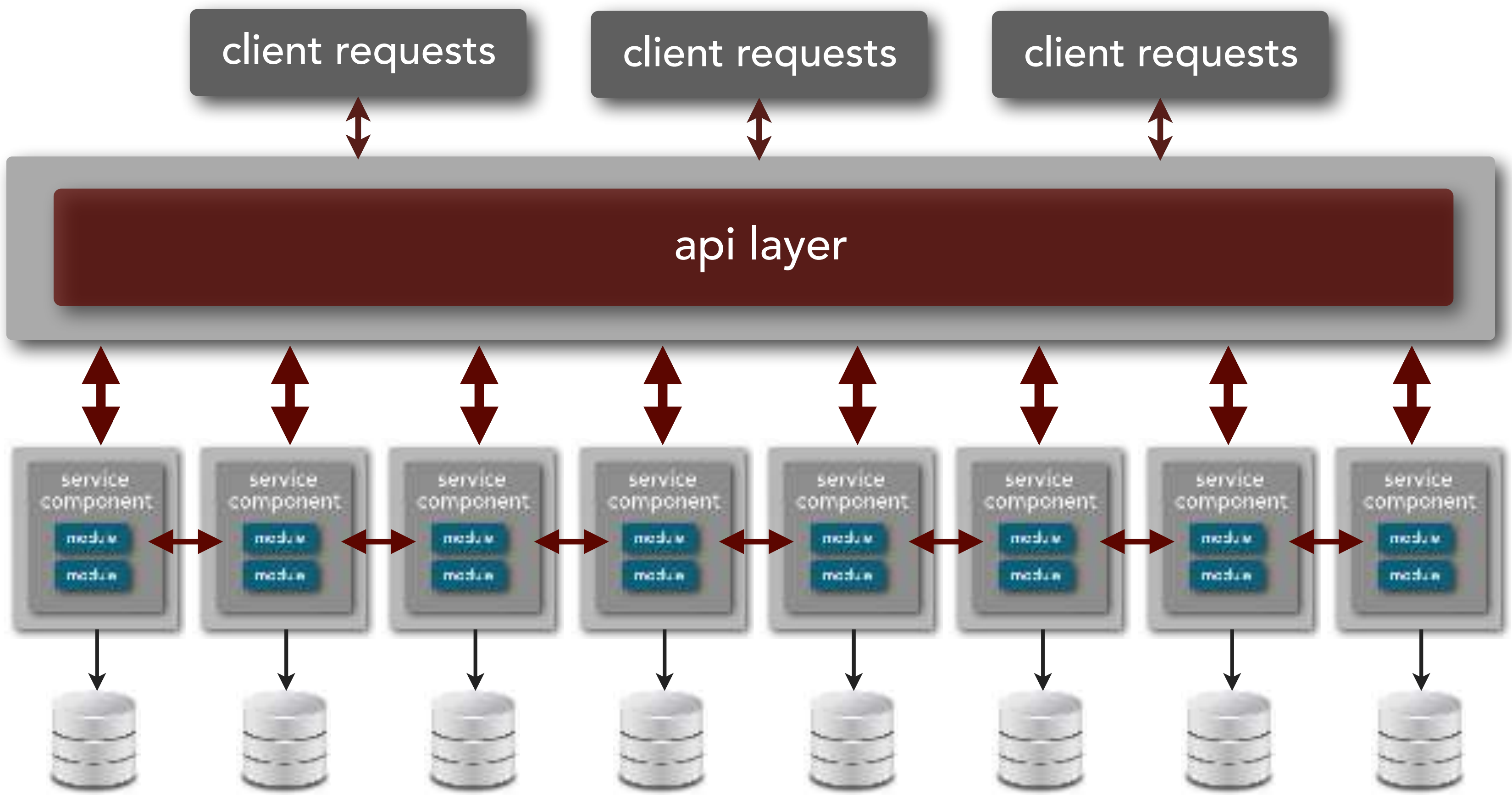


microservices architecture

distributed

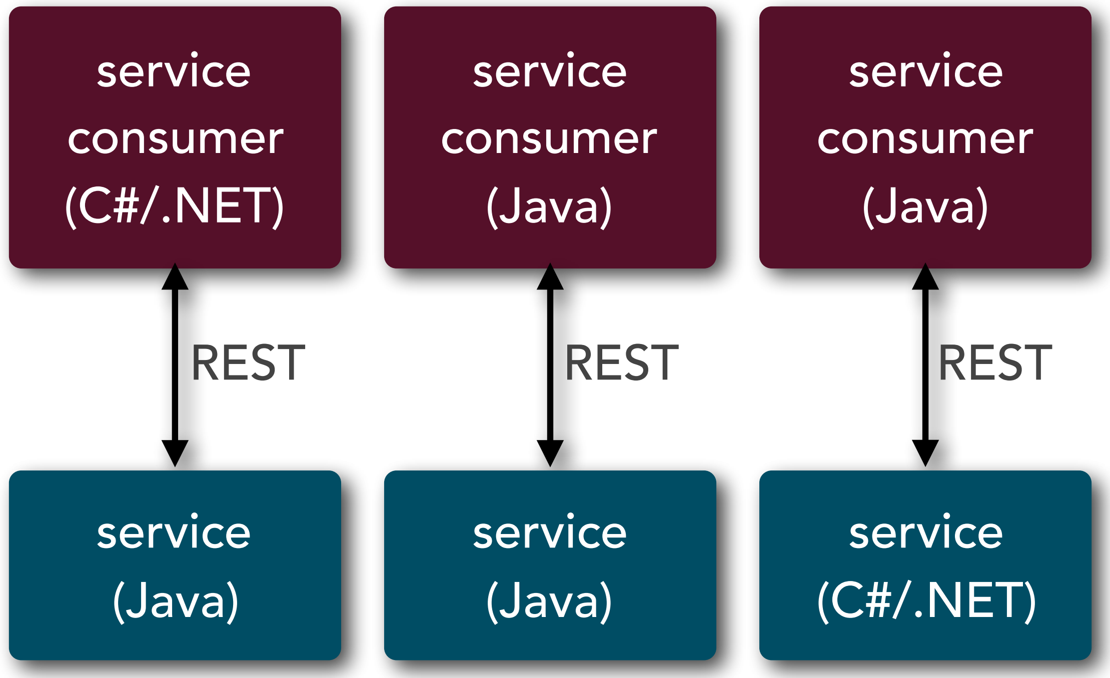


microservices architecture



microservices architecture

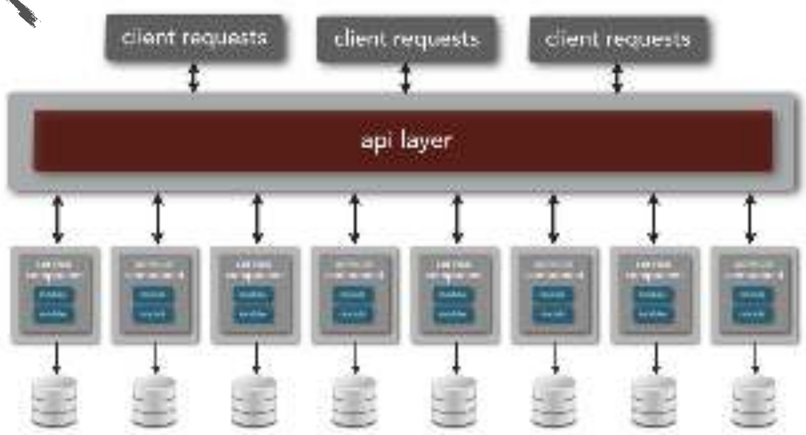
protocol-aware heterogeneous interoperability



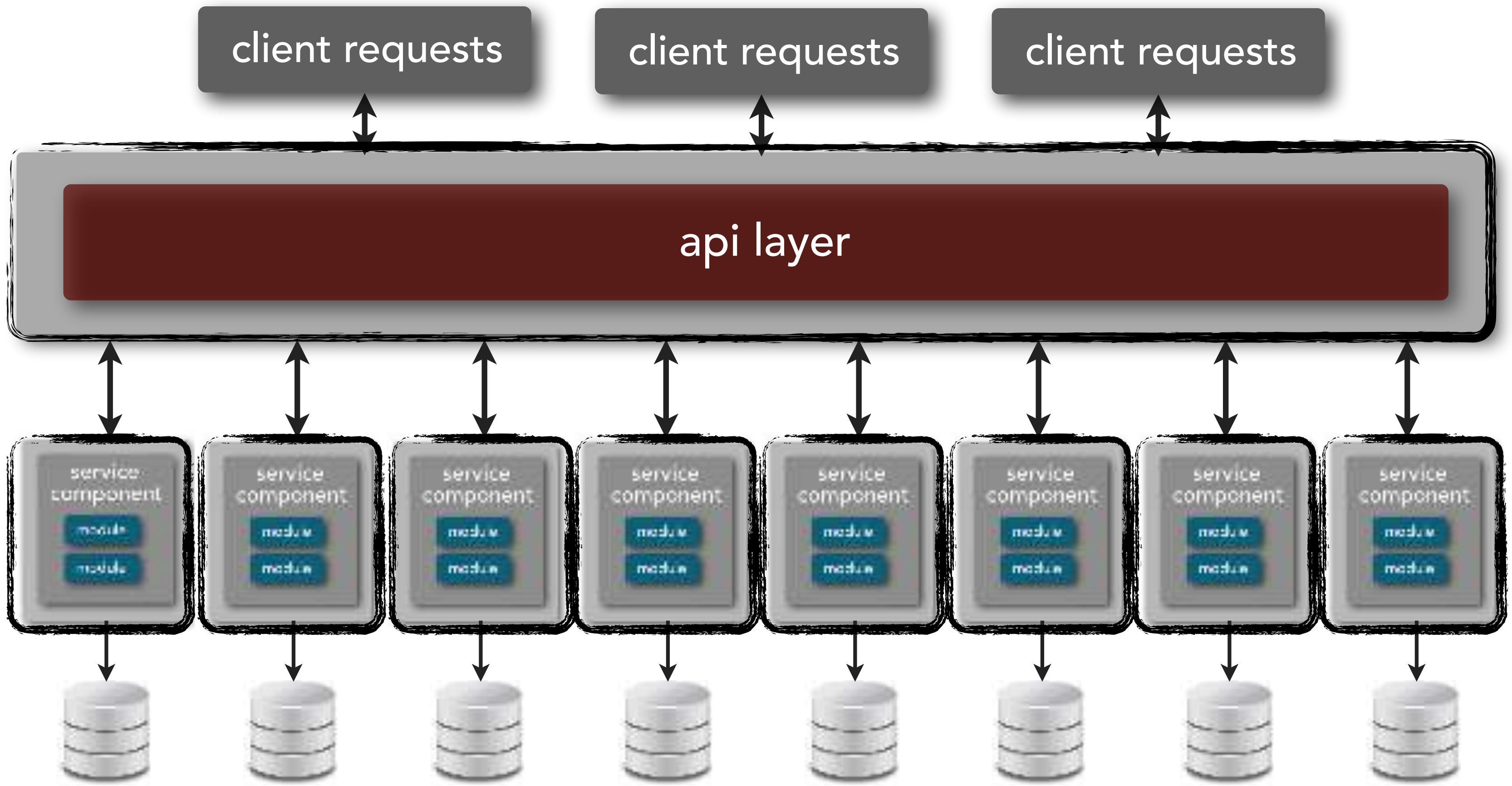
microservices architecture

distributed

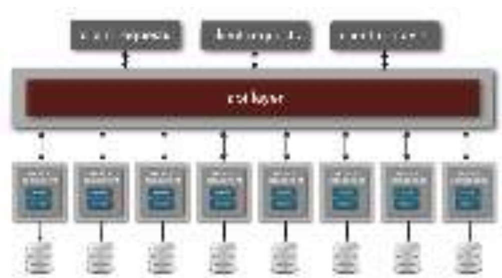
separately deployed



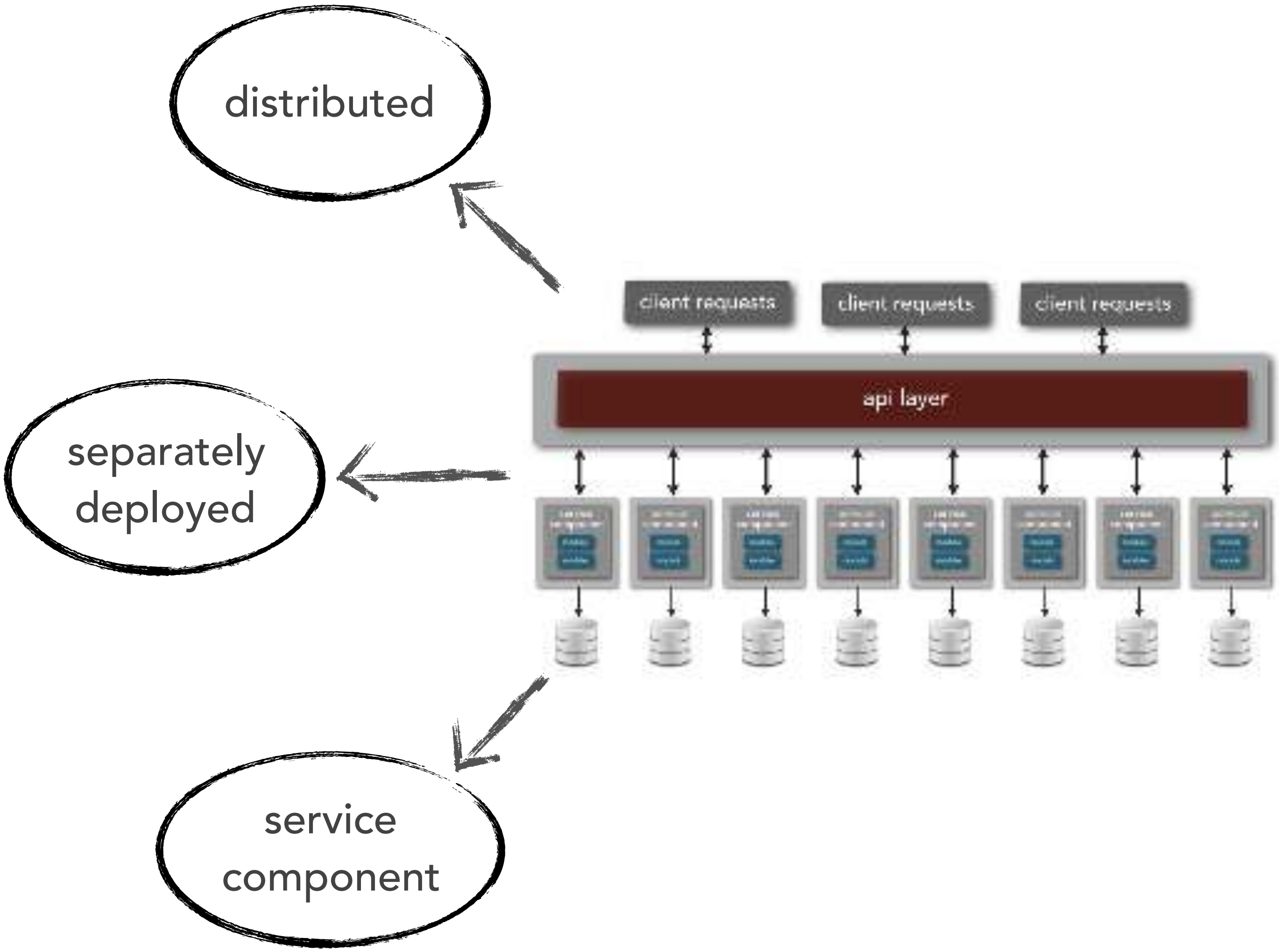
microservices architecture



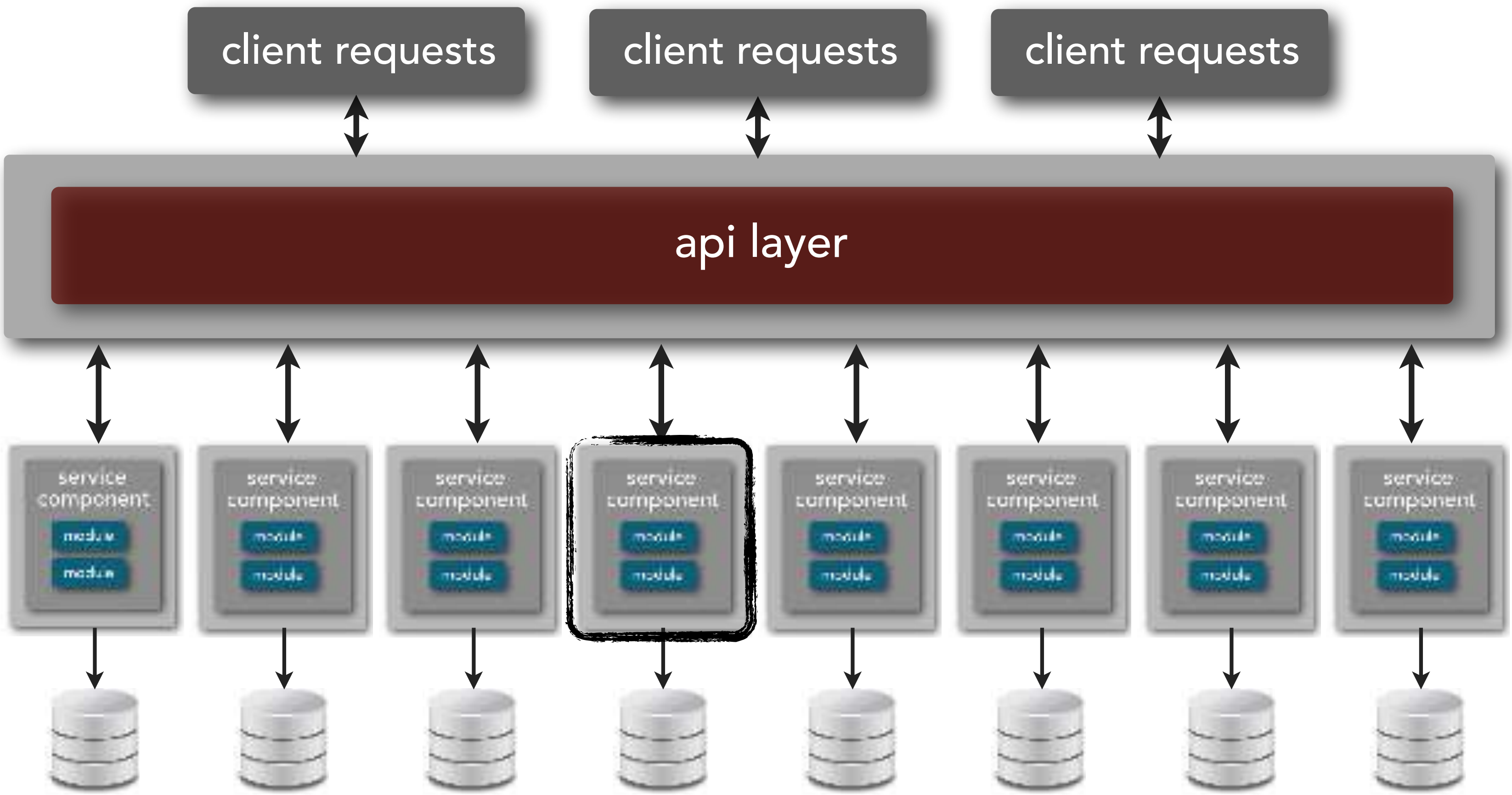
microservices architecture

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

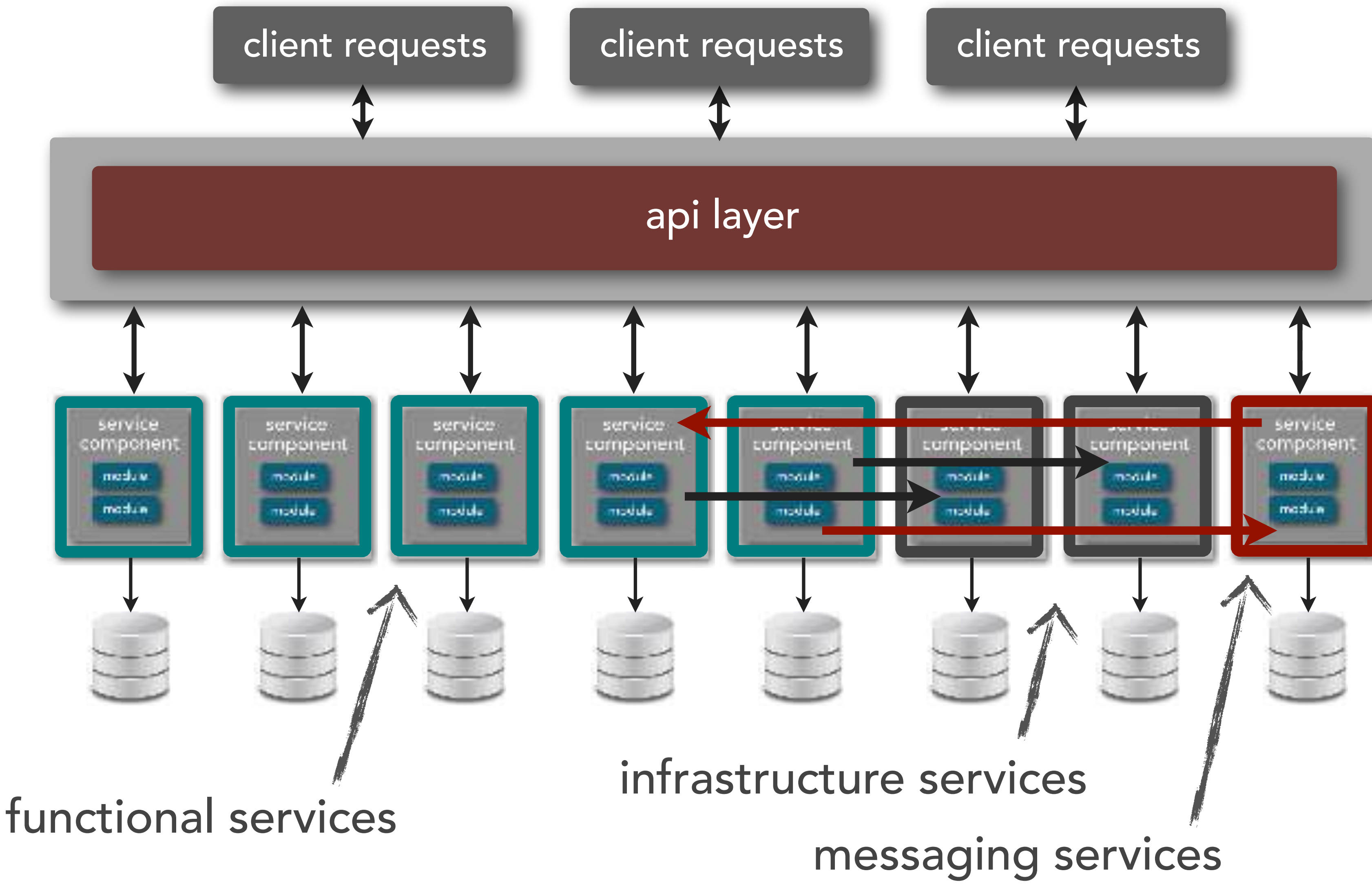
microservices architecture



microservices architecture



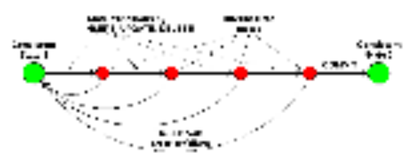
microservices architecture



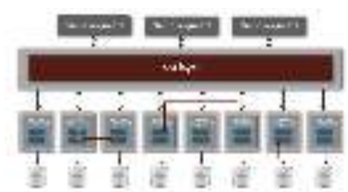
microservices architecture

what is the right size for a microservice?

purpose



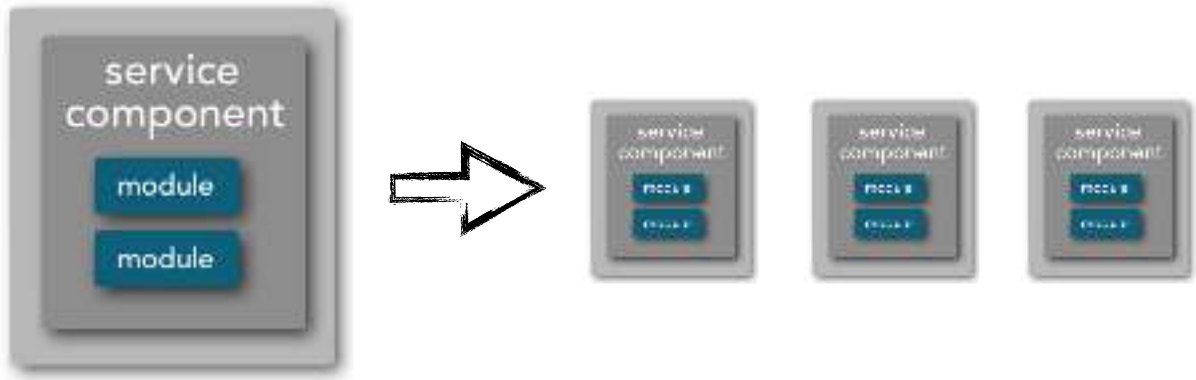
transactions



choreography

microservices architecture

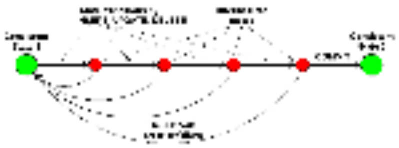
purpose



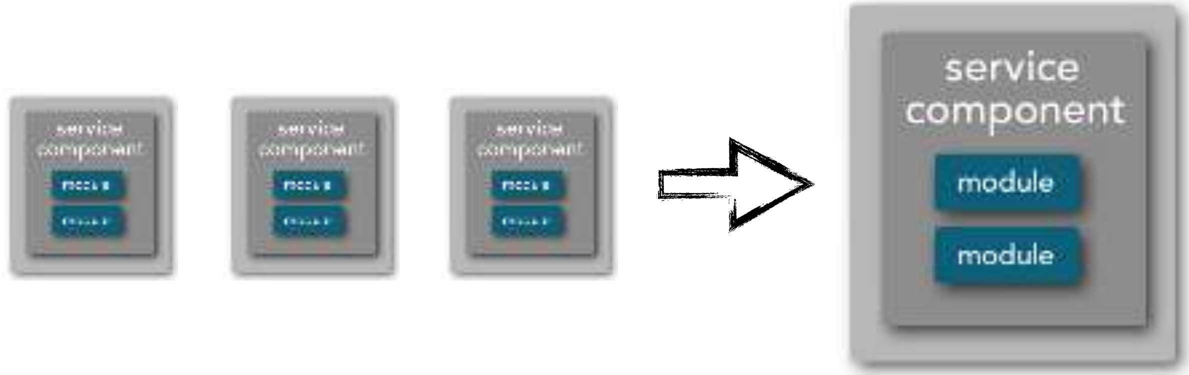
service scope and function

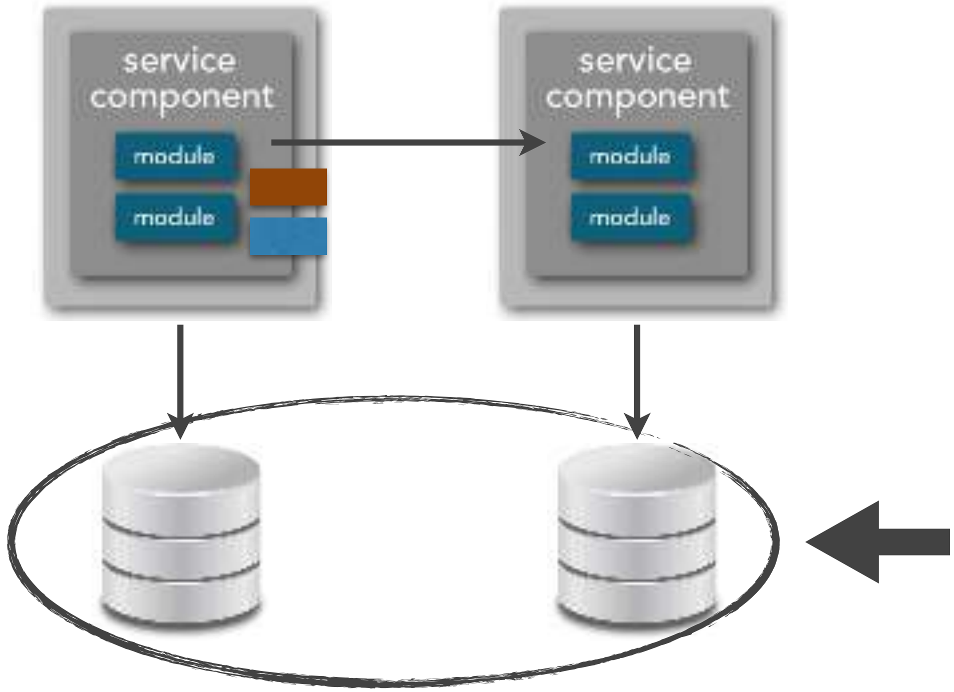
(single-purpose function)

microservices architecture



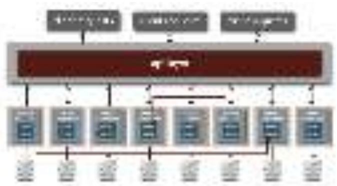
transactions



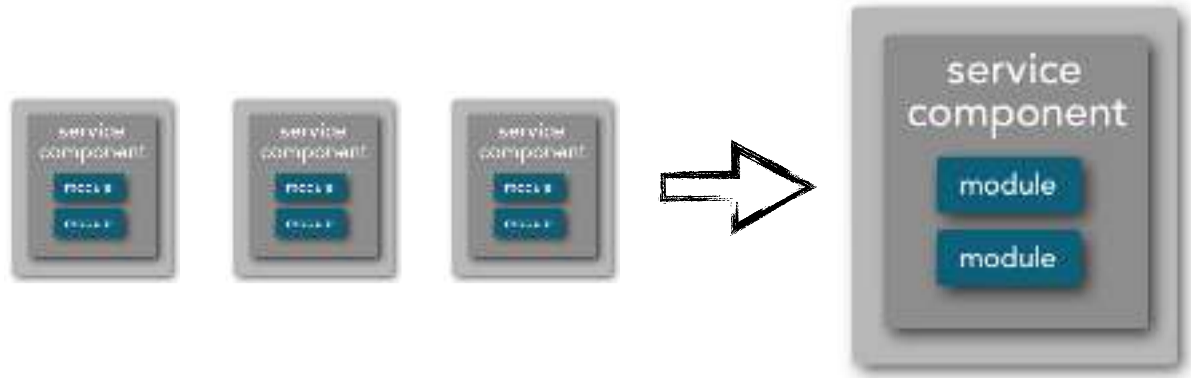


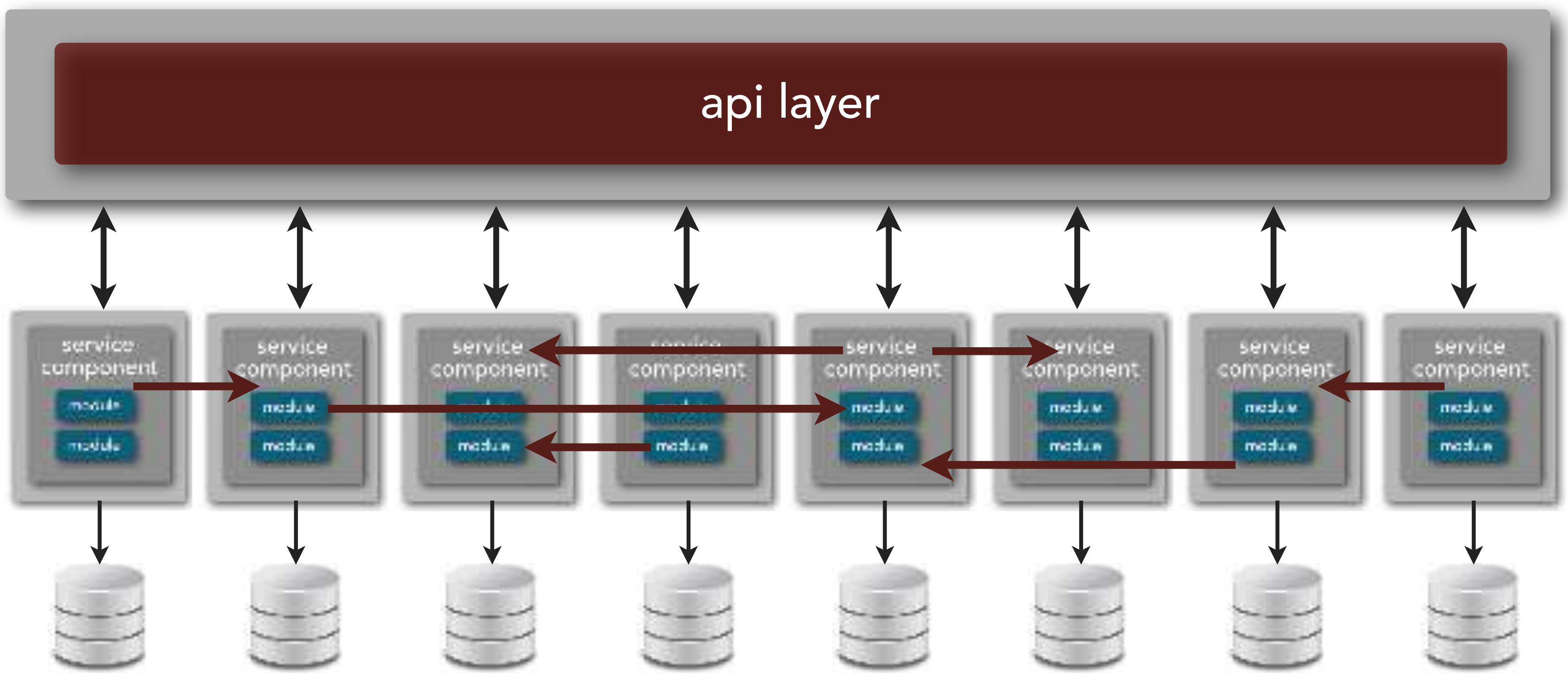
no acid
transaction

microservices architecture

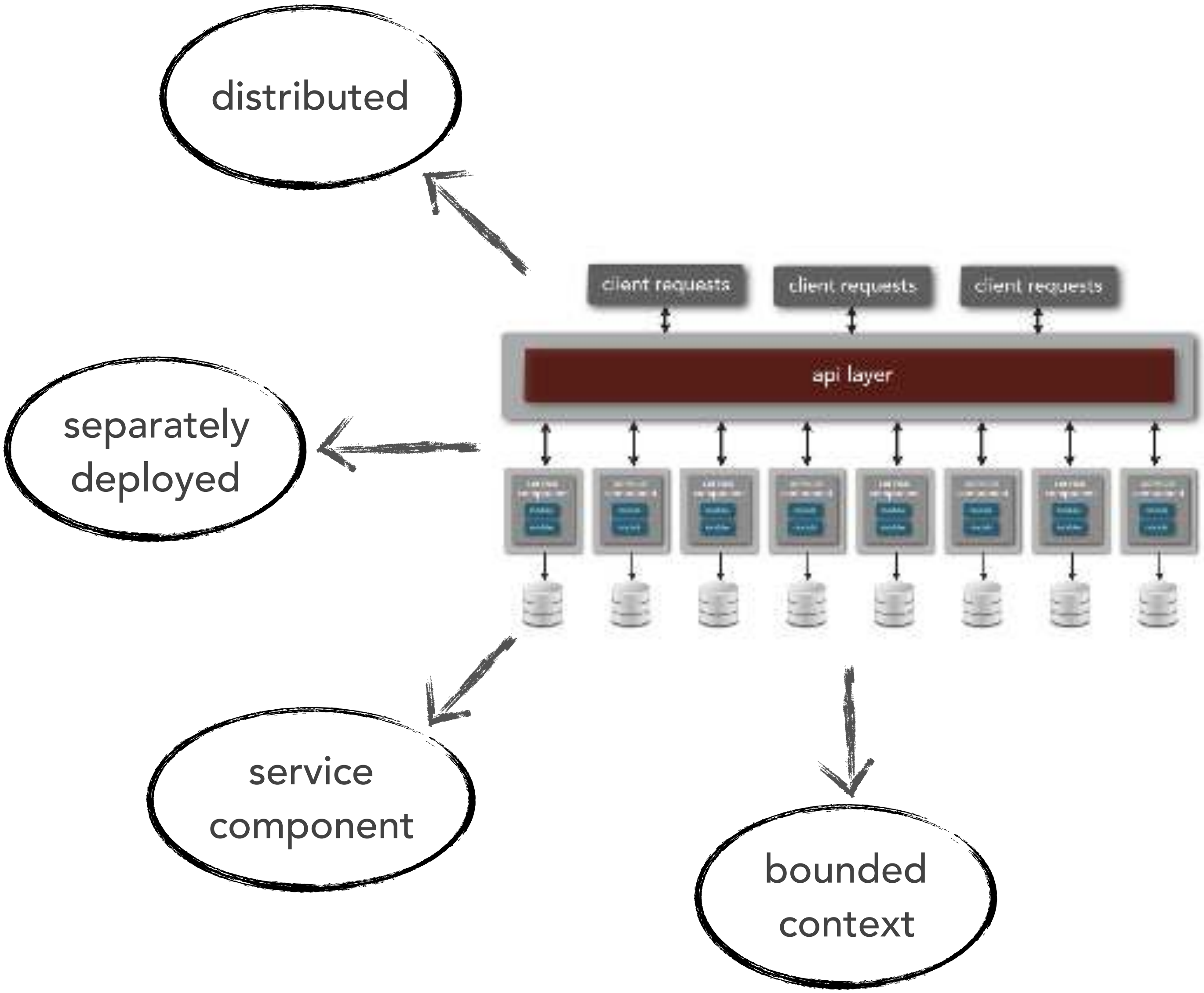


choreography

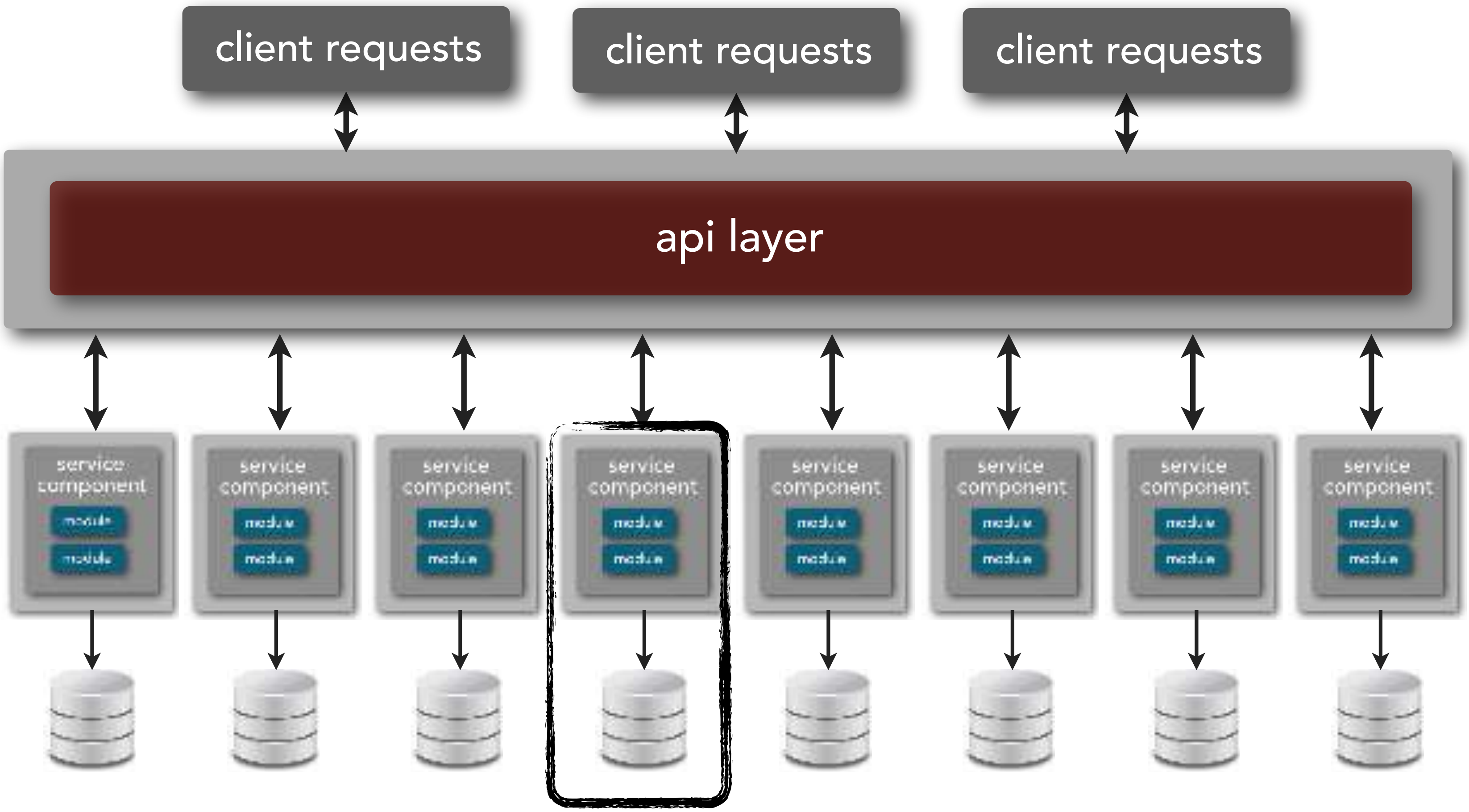




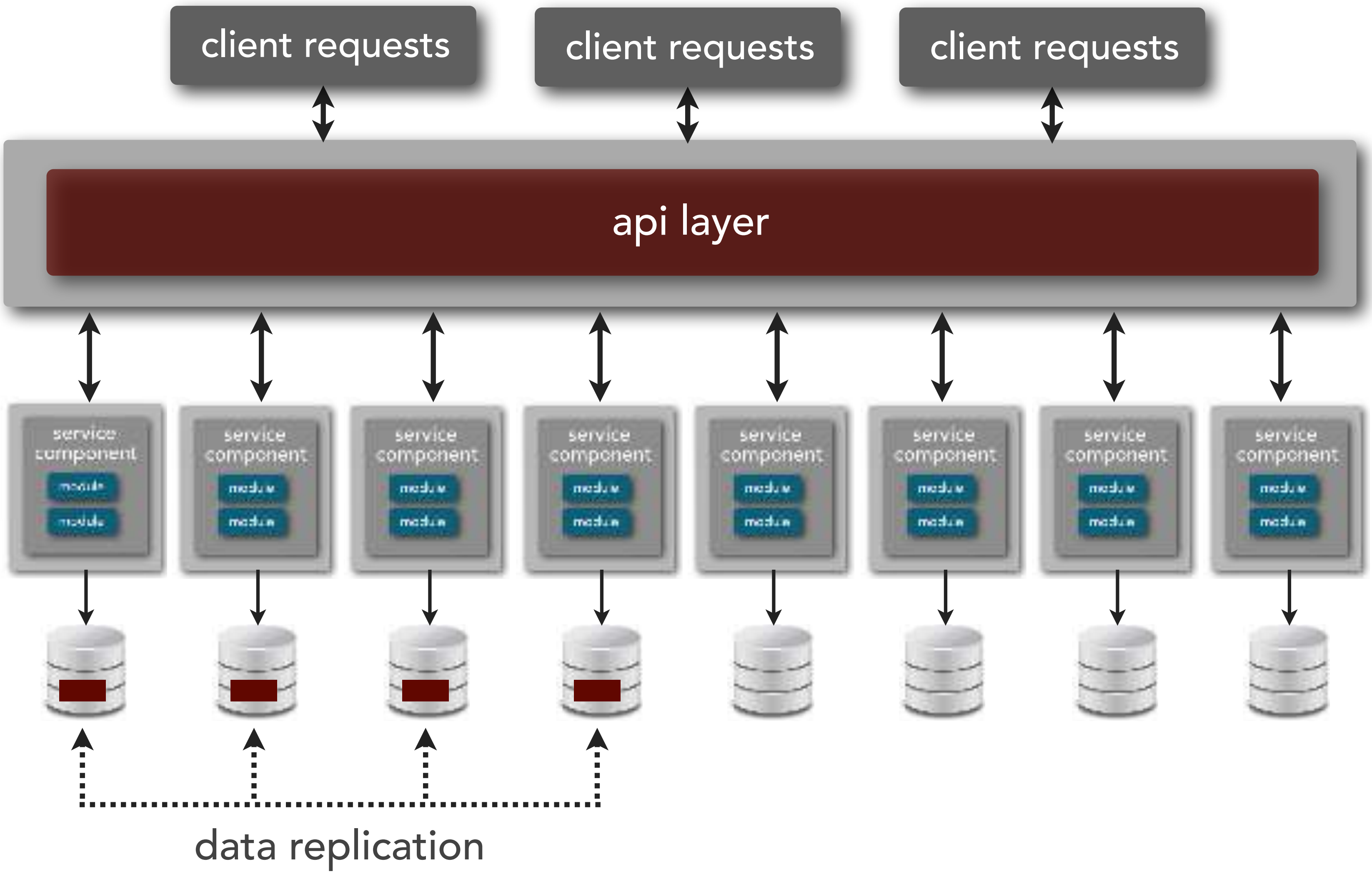
microservices architecture



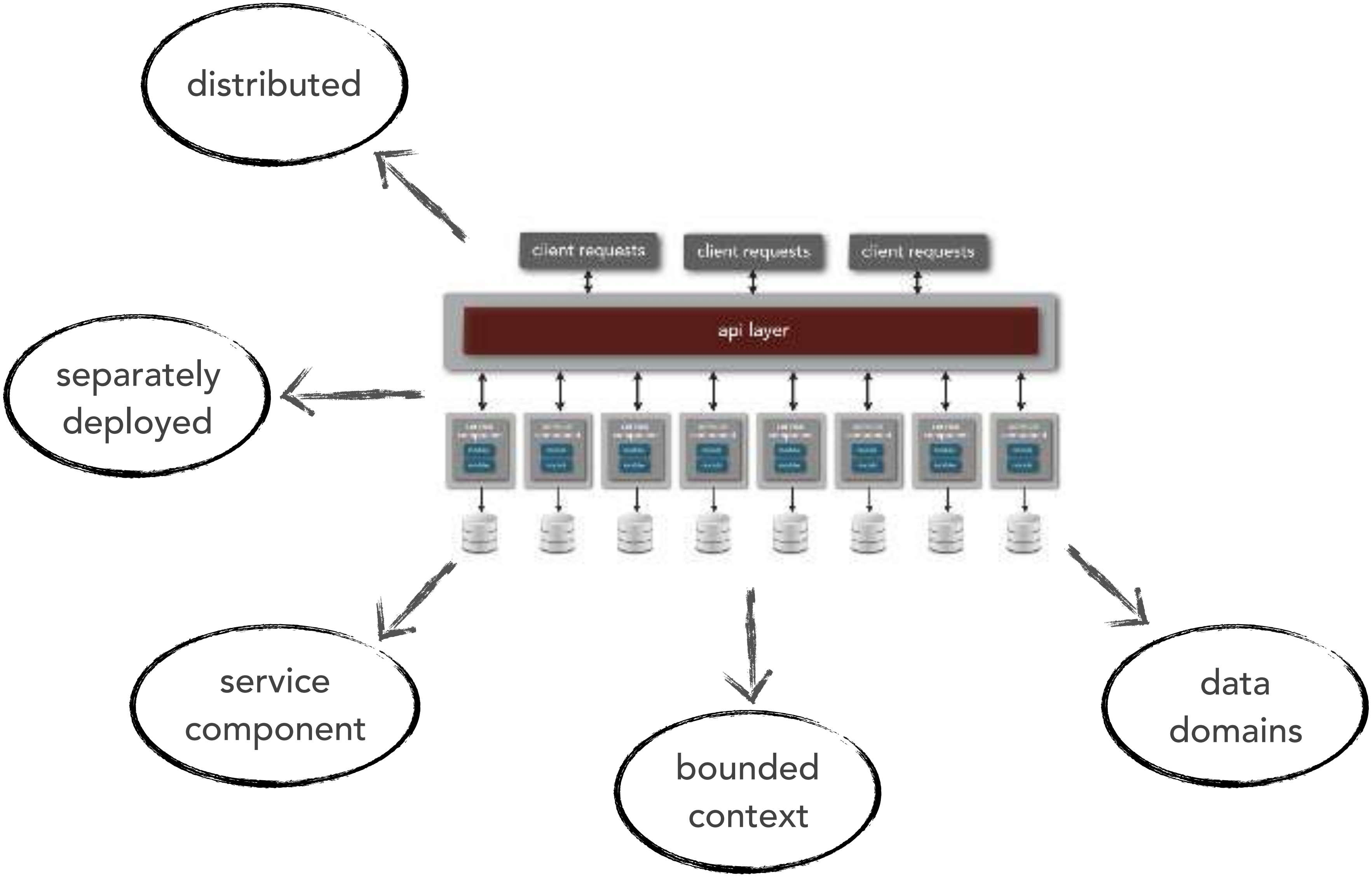
microservices architecture



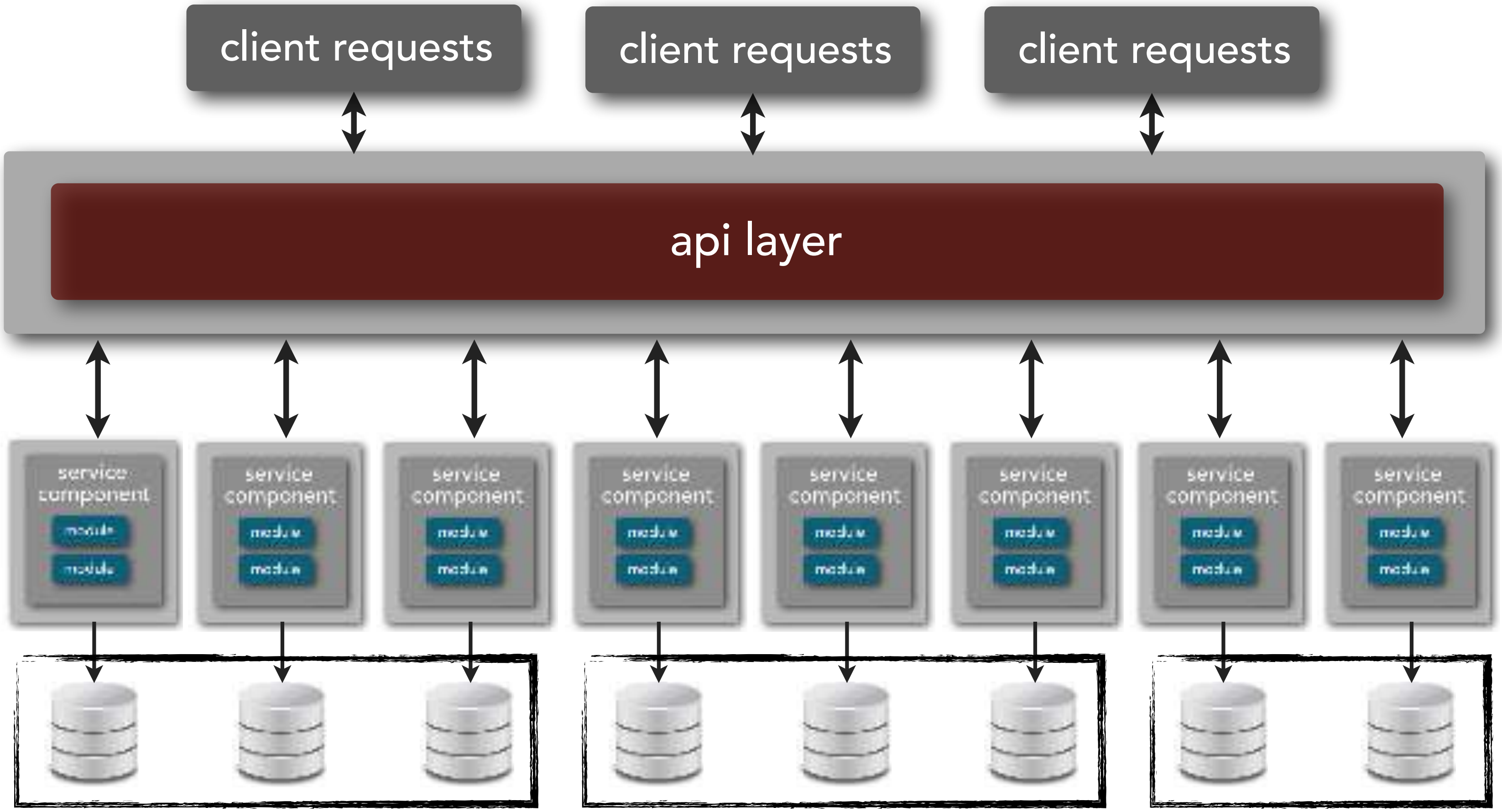
microservices architecture



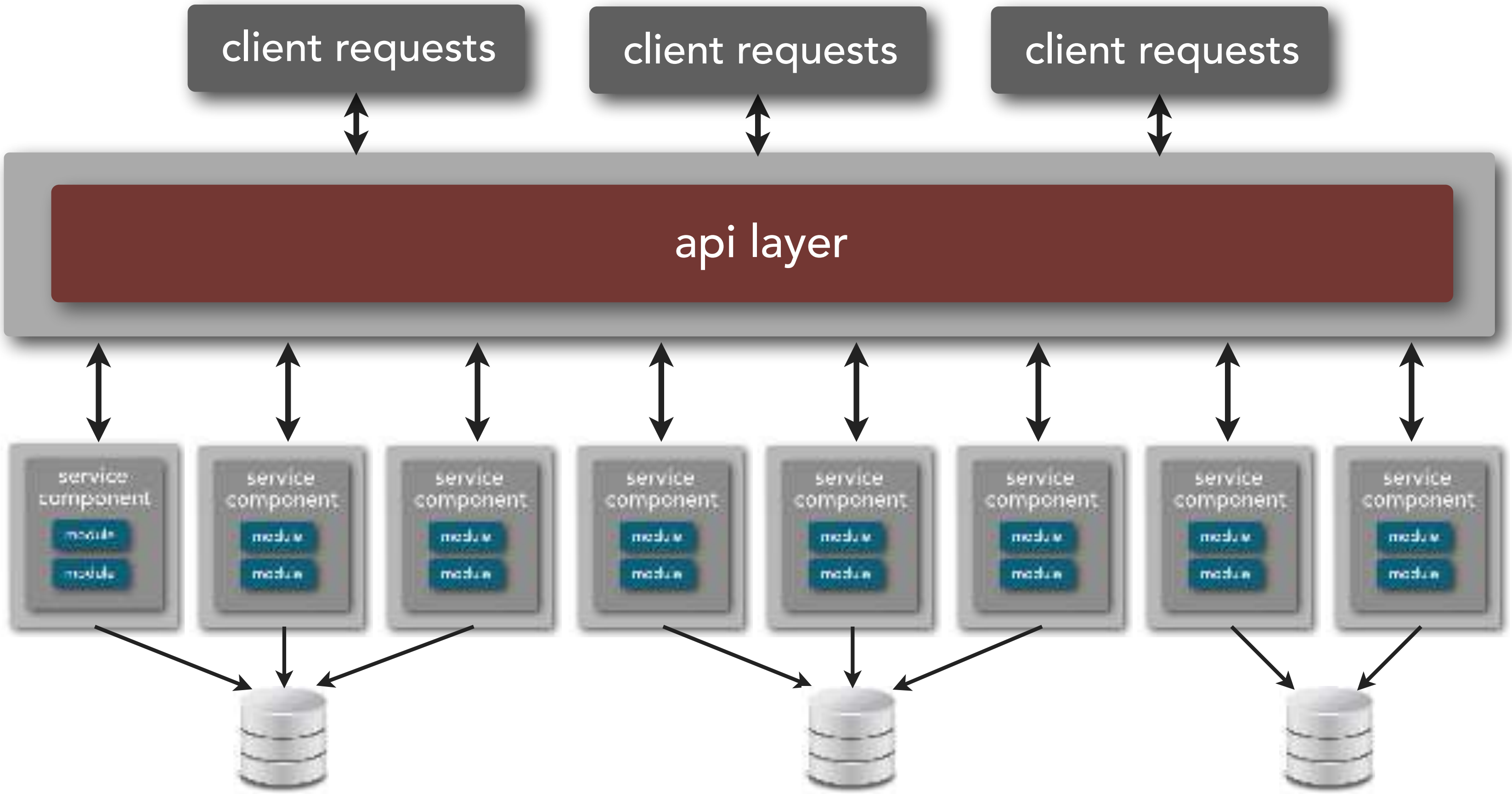
microservices architecture



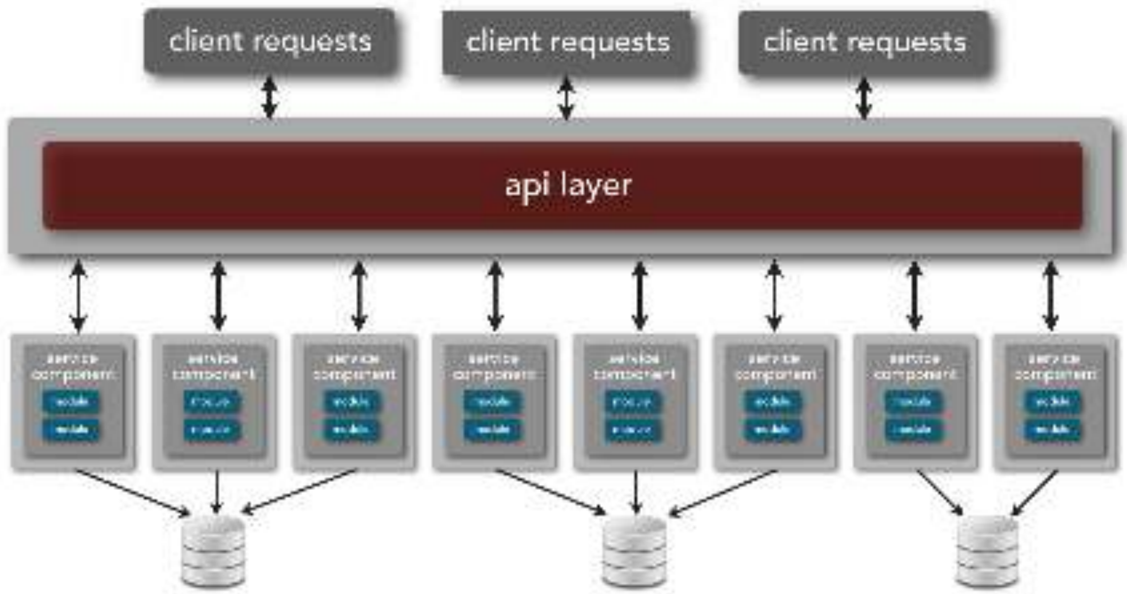
microservices architecture



microservices architecture



microservices architecture

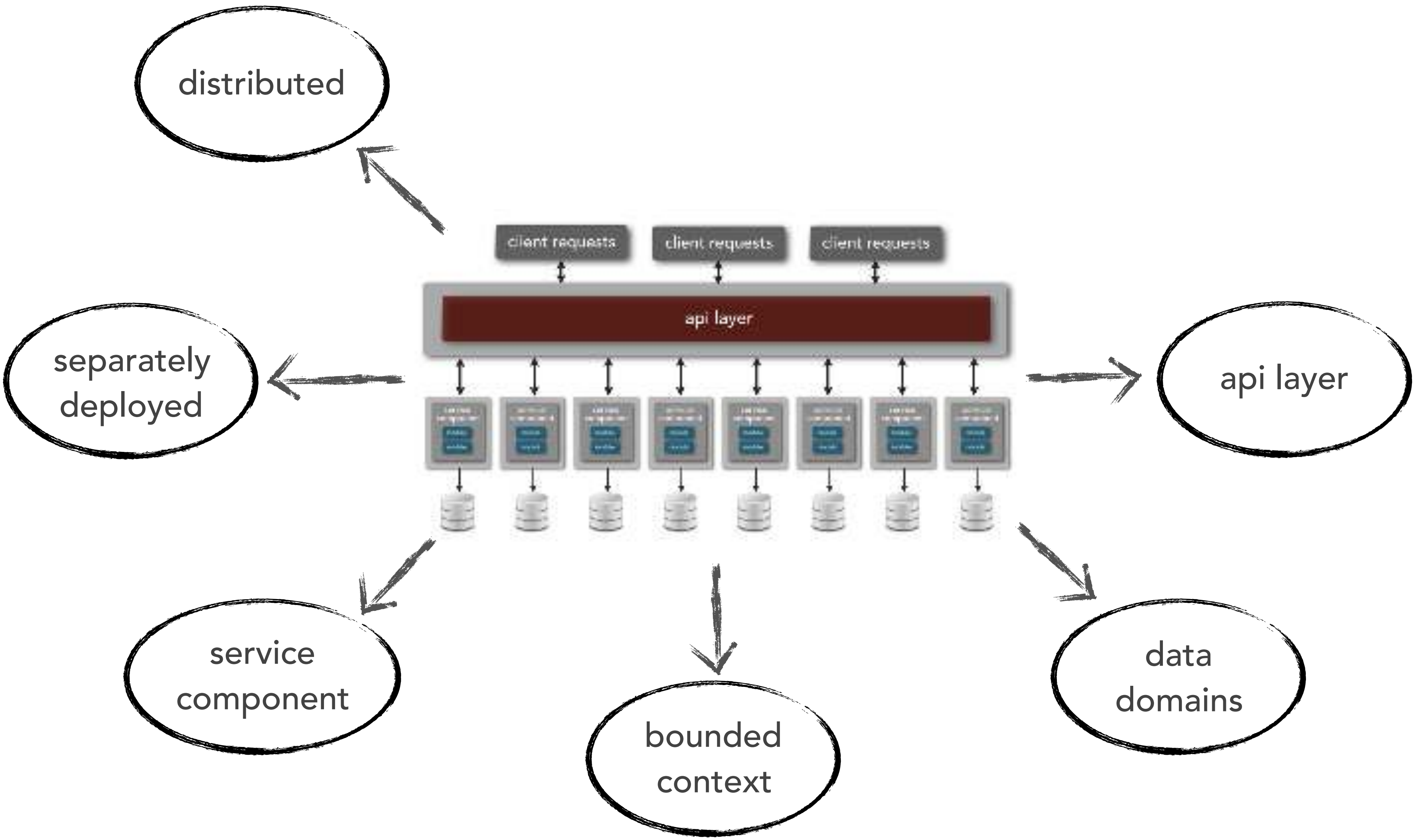


assumes low rate of schema changes (or use of noSQL)

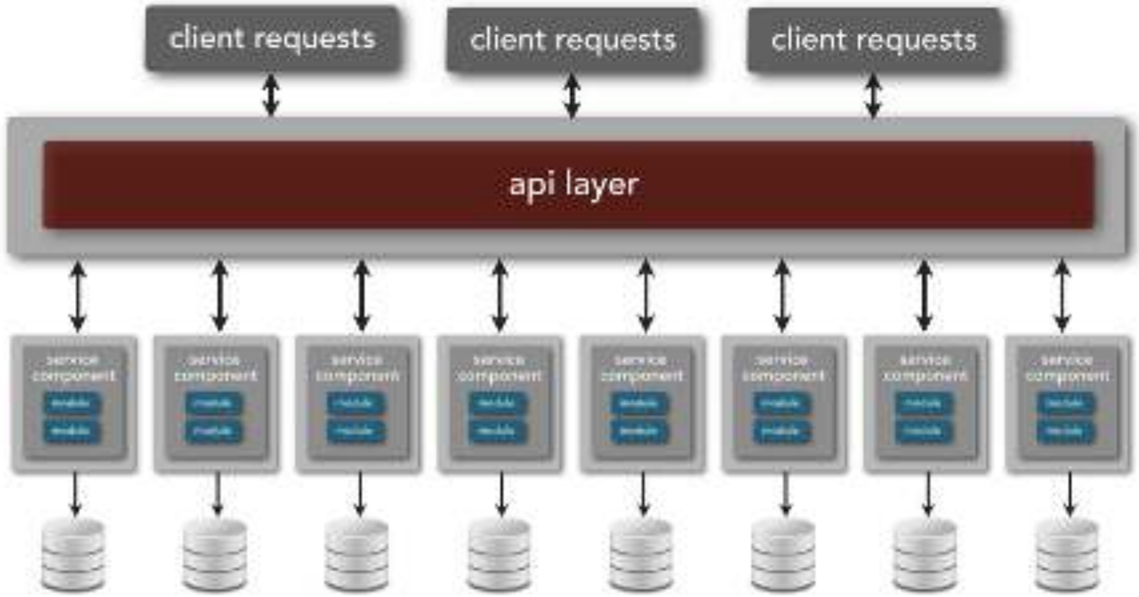
increases performance and overall reliability

reduces data duplication

microservices architecture

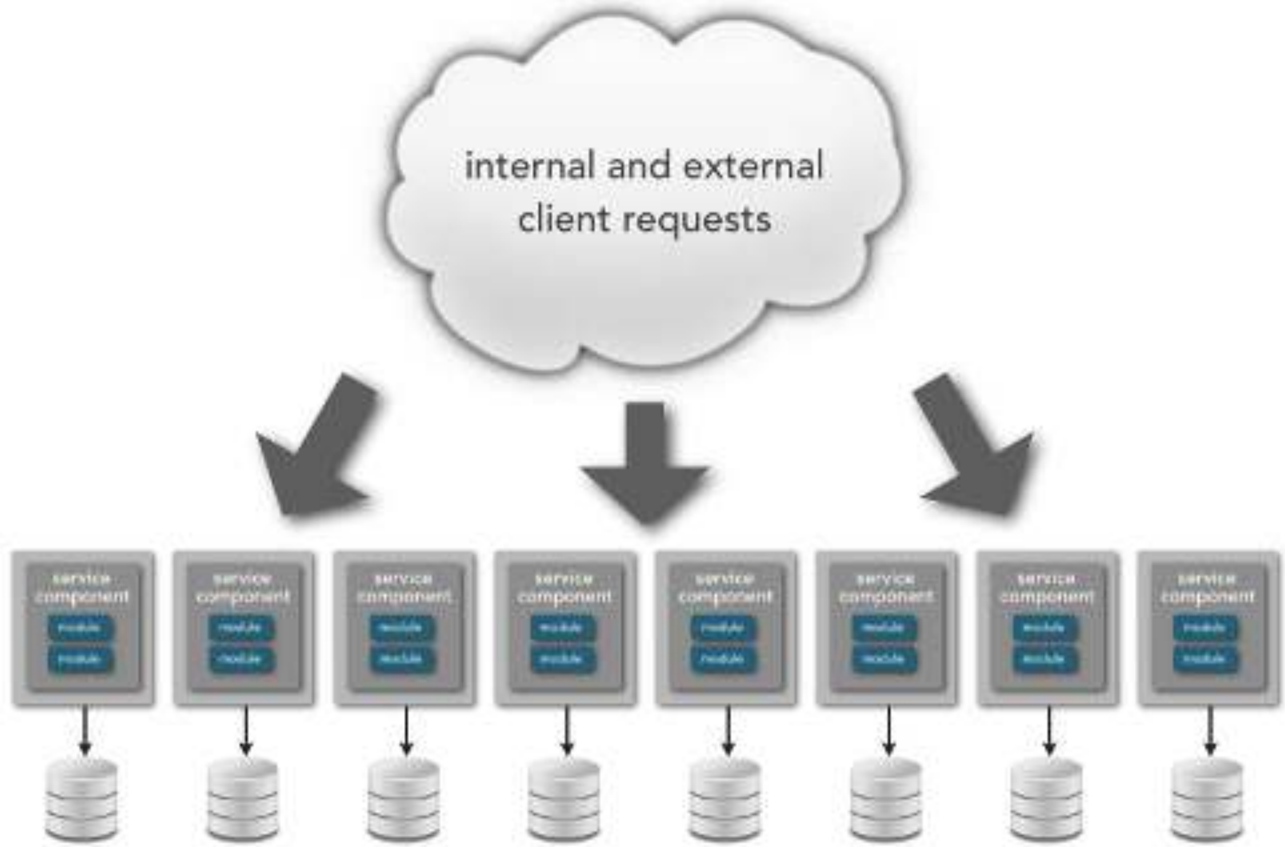


api layer

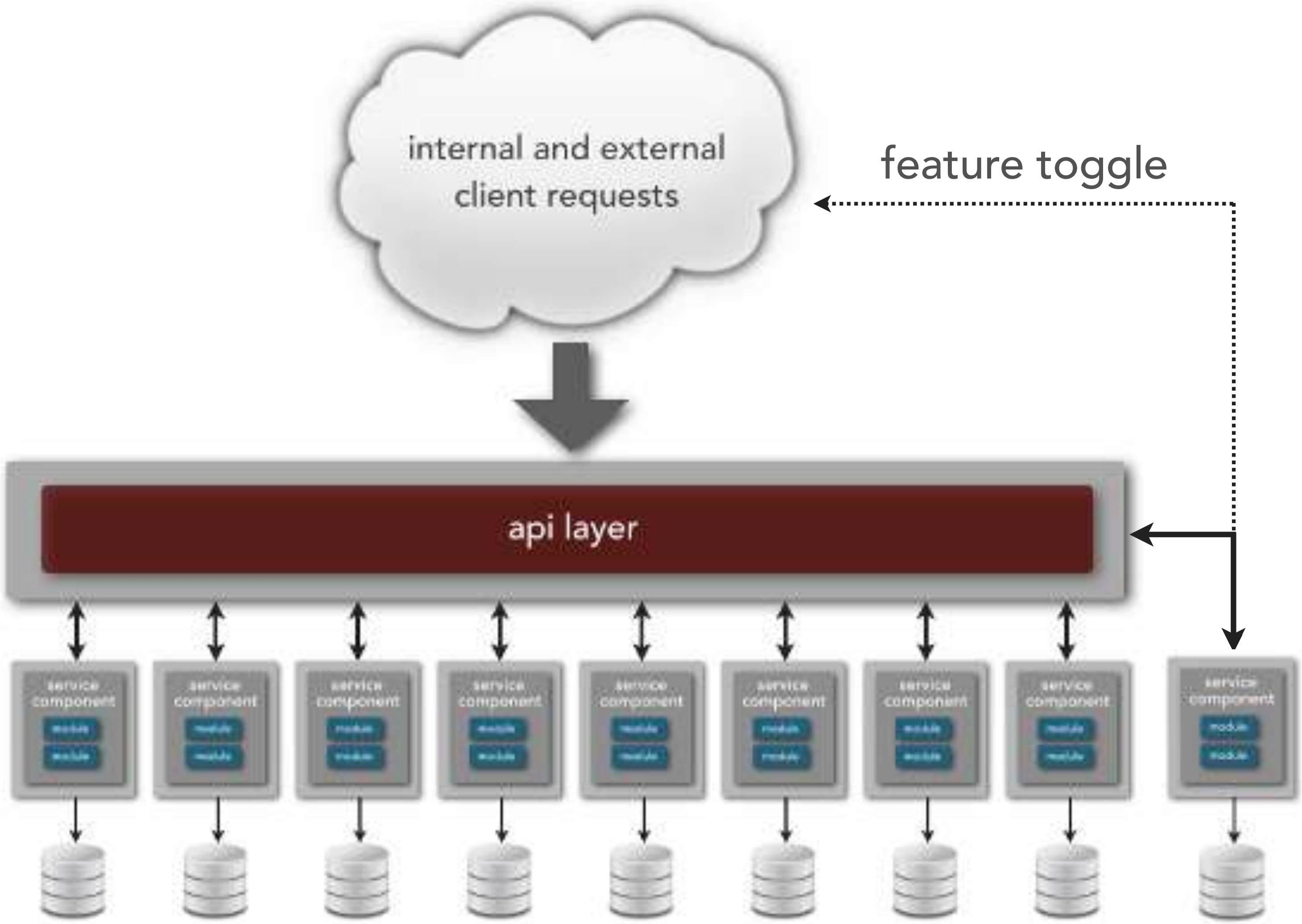


hides the actual endpoint of the service, exposing only those services available for public consumption

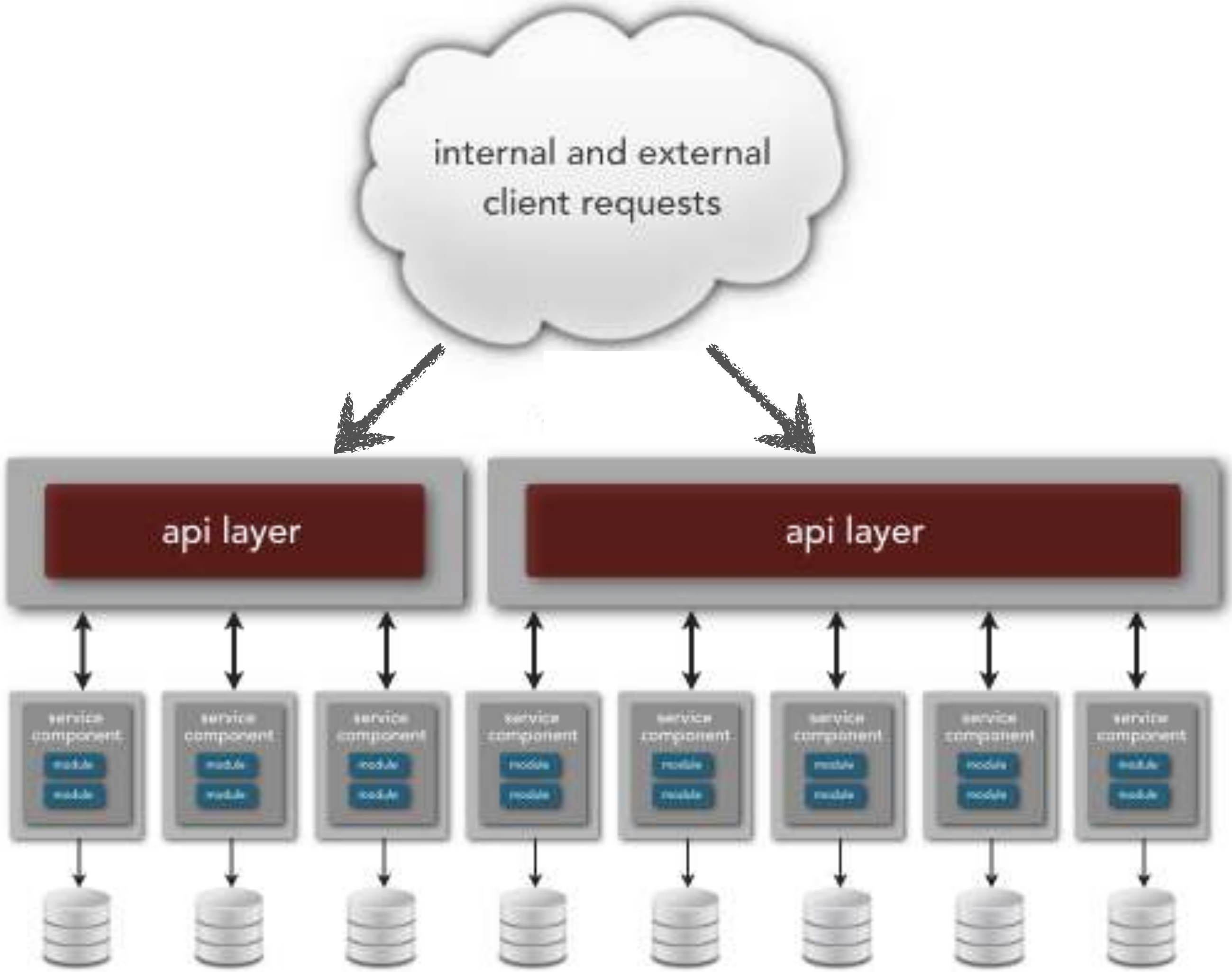
api layer


internal and external
client requests
service component
module
module

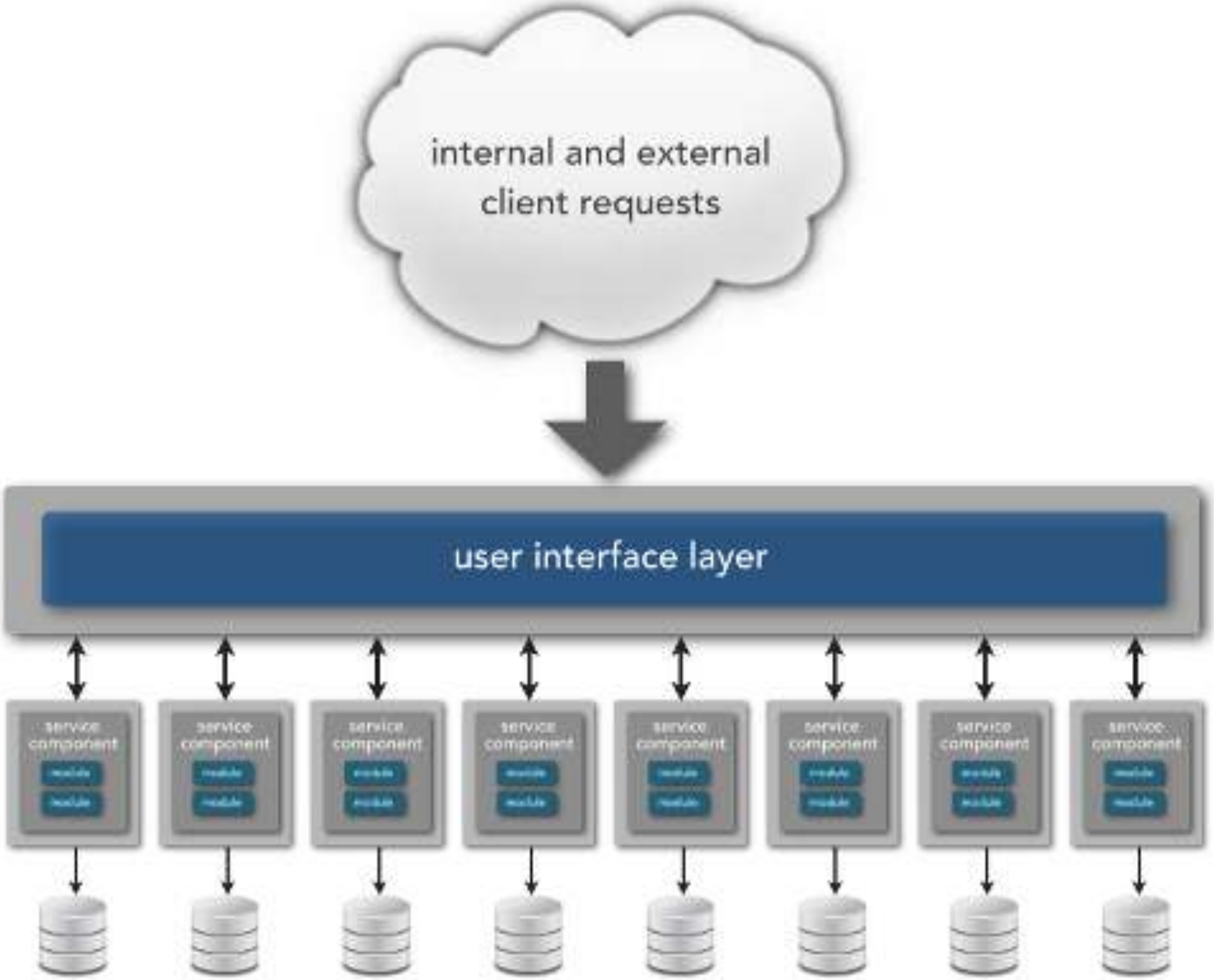
api layer


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

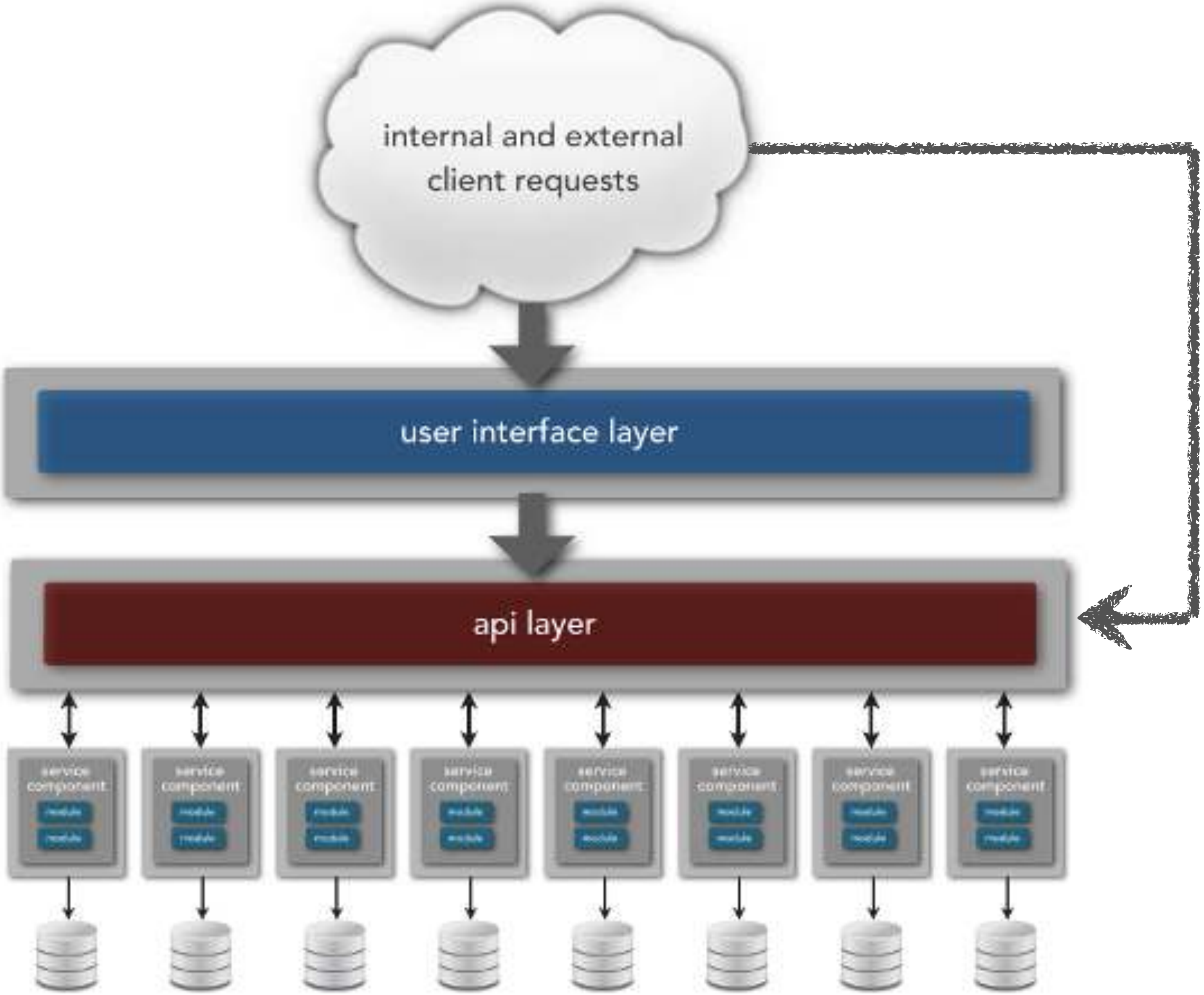
api layer


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

api layer

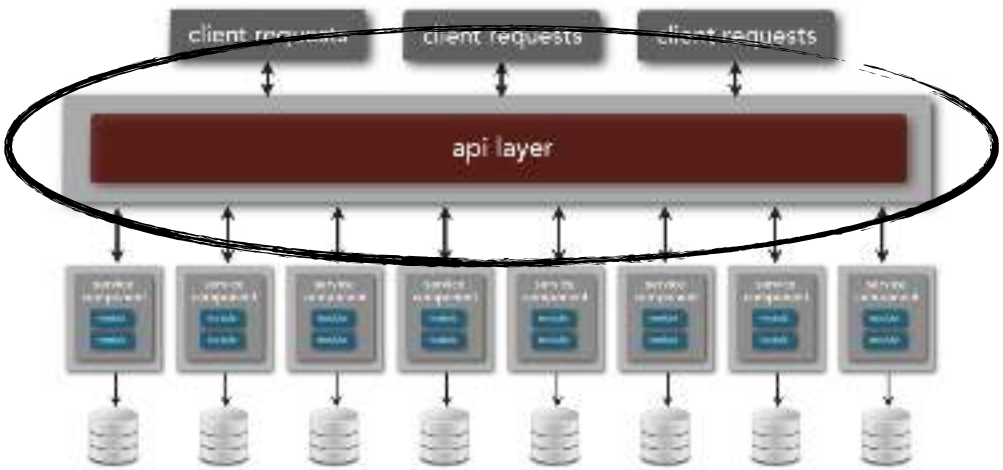

internal and external client requests
user interface layer
service component

microservices architecture



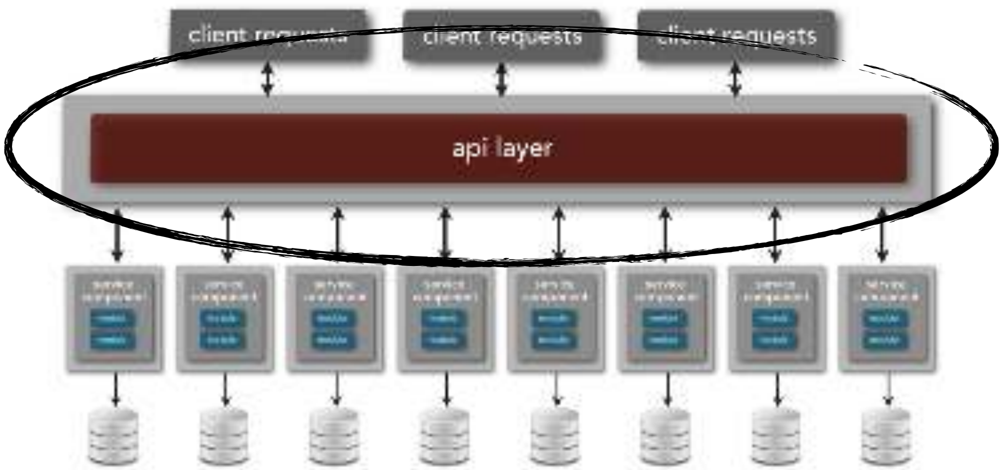
api layer

endpoint proxy



api layer

load balancer



Barracuda

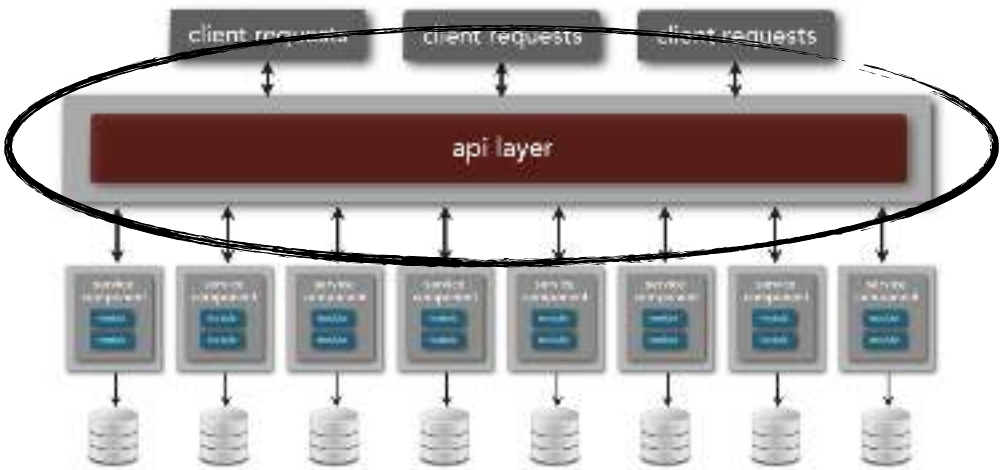
neustar

f5

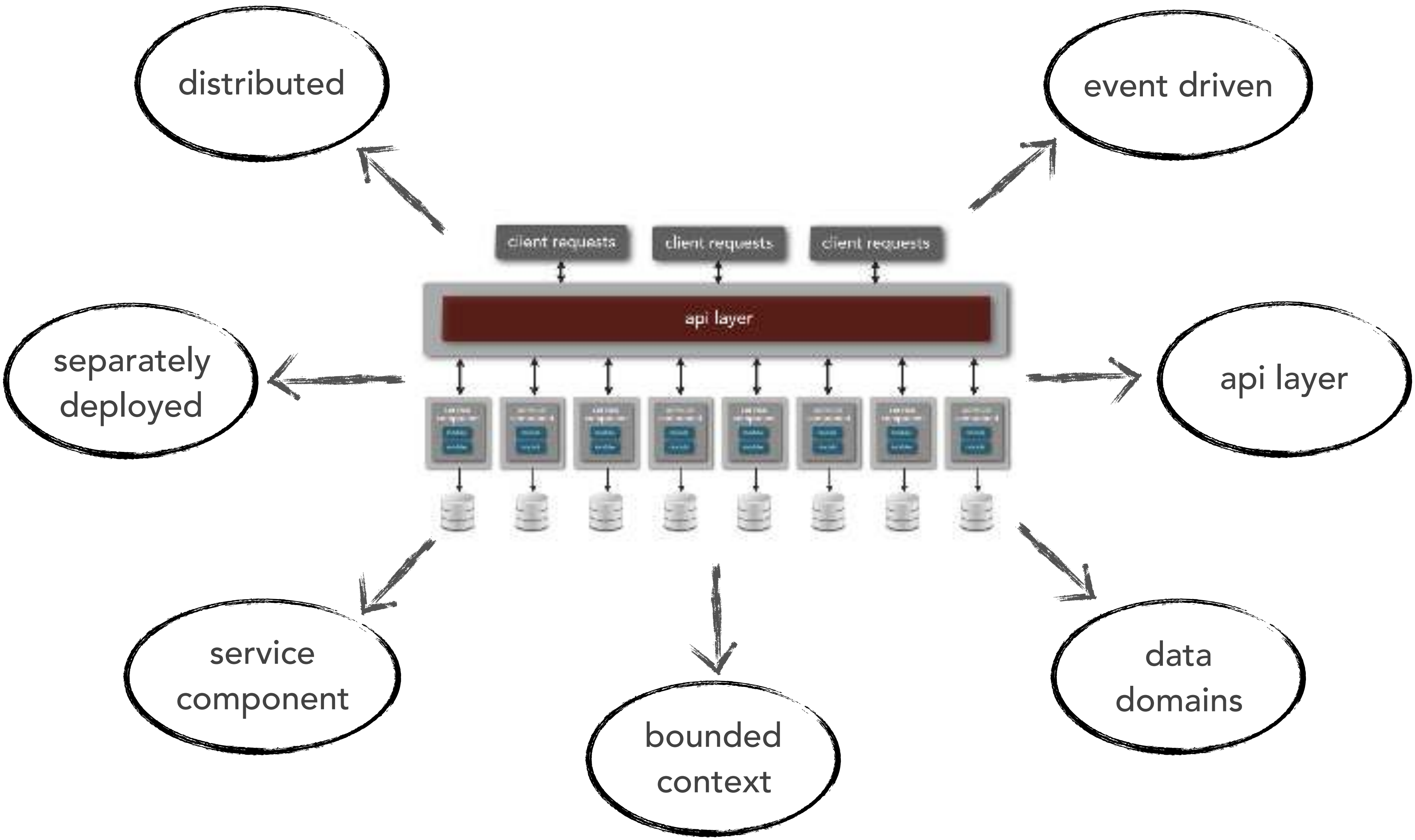
Citrix NetScaler

api layer

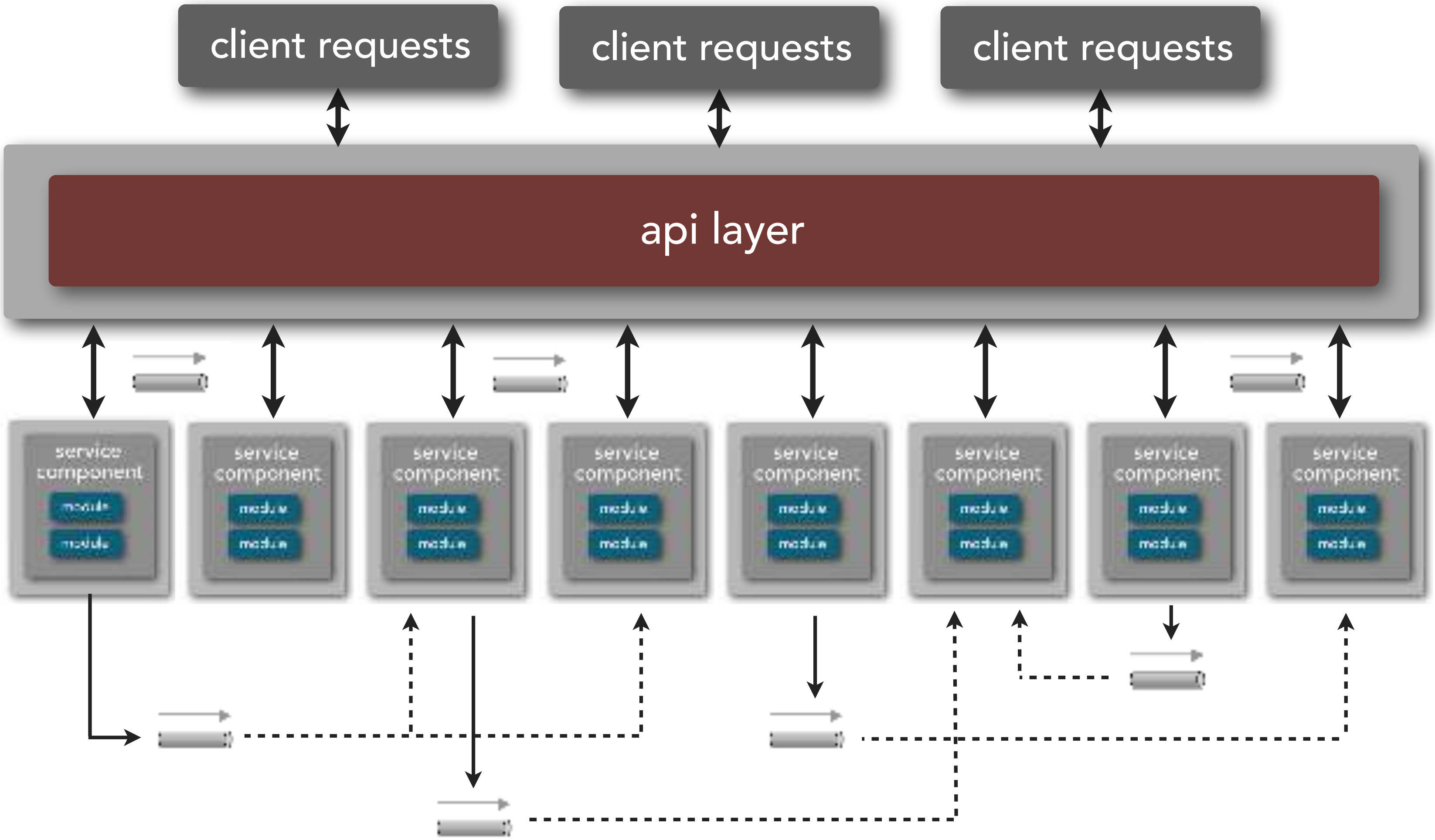
gateway (integration hub)



microservices architecture

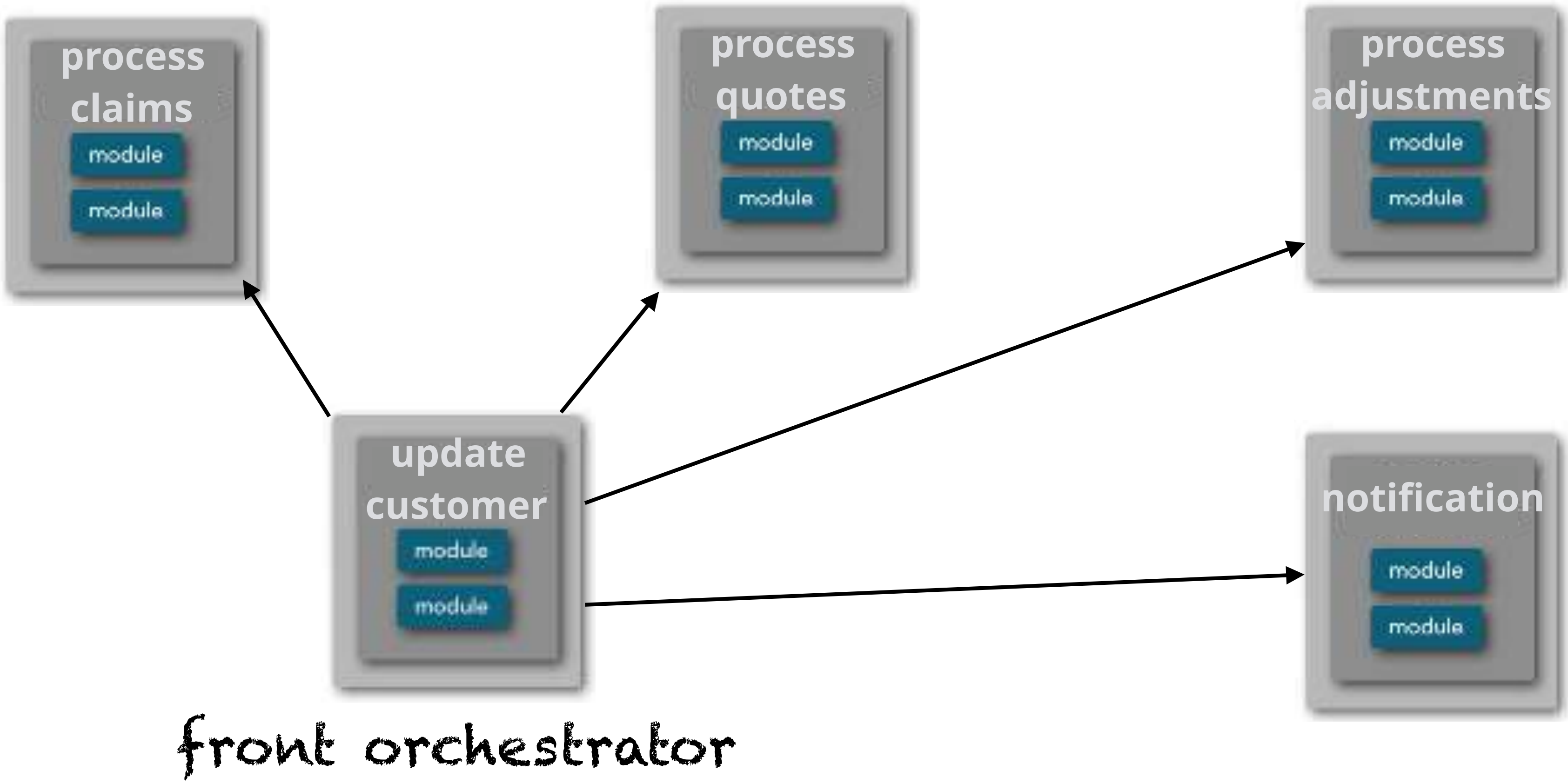


microservices architecture



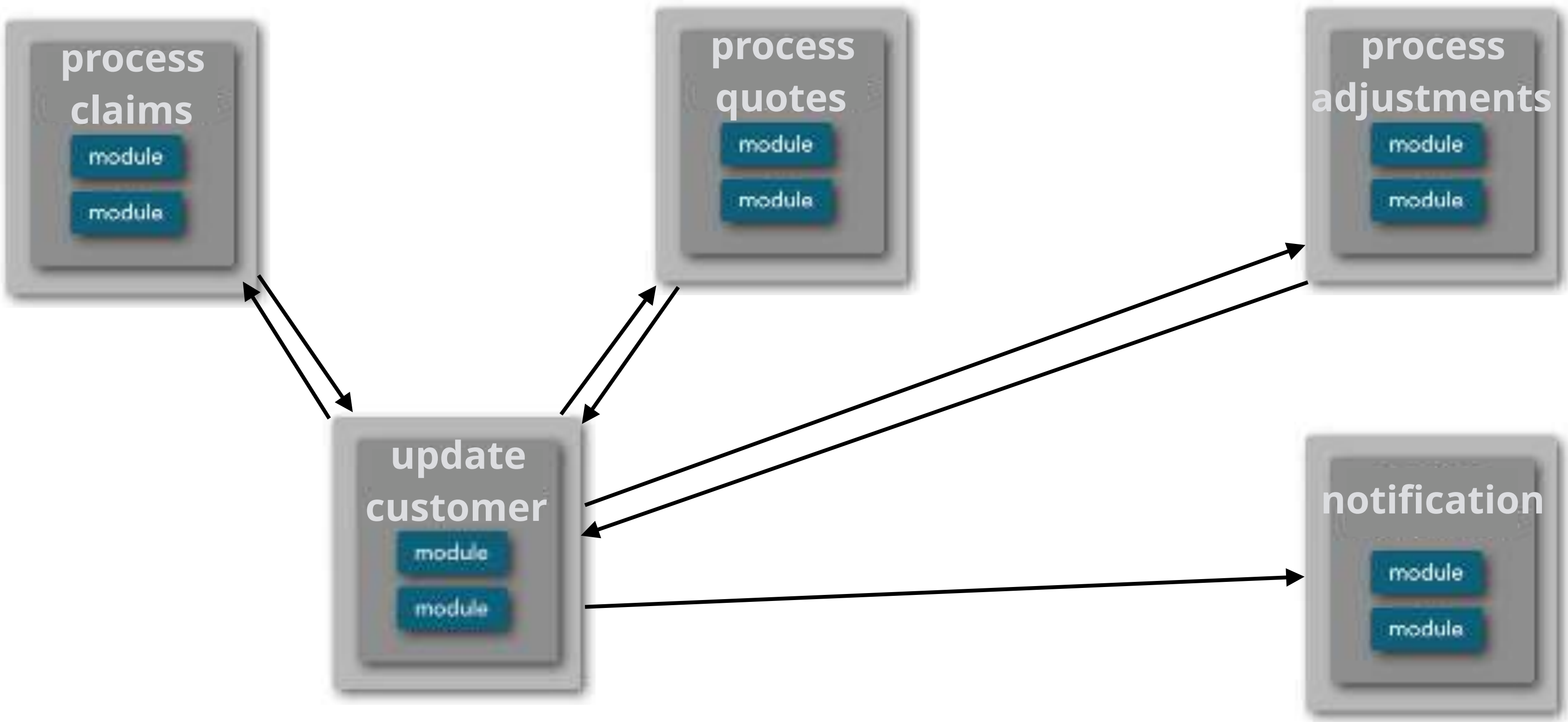
microservices architecture

service orchestration



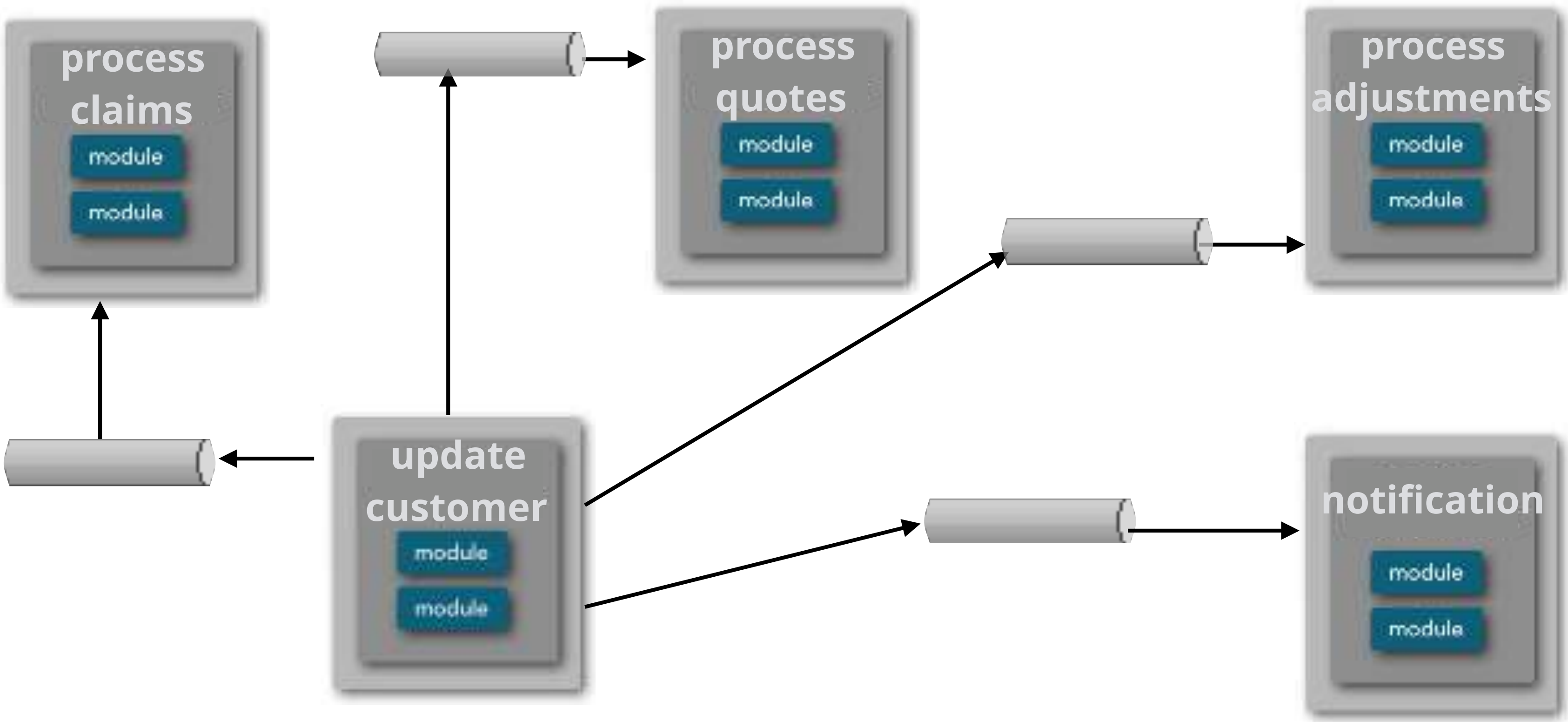
microservices architecture

service orchestration



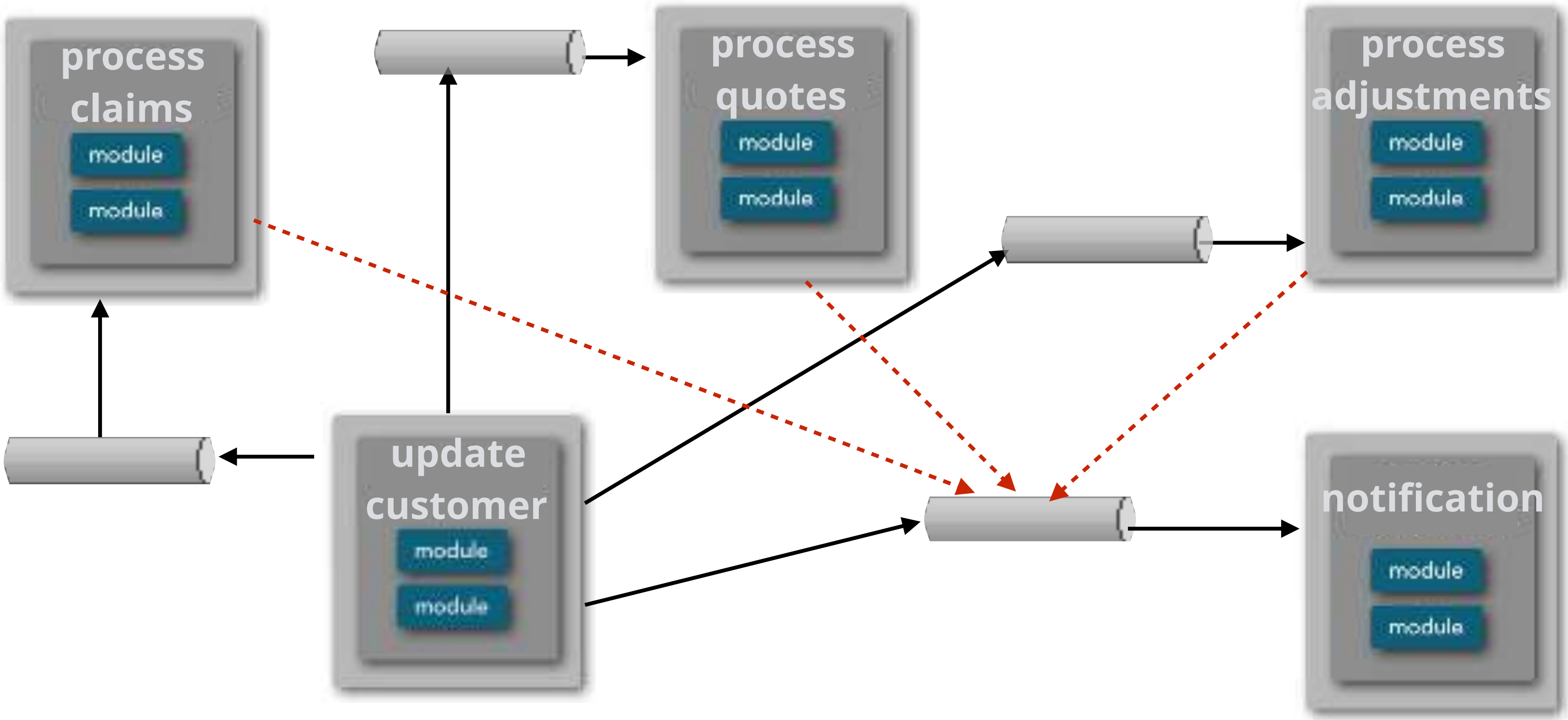
microservices architecture

service orchestration



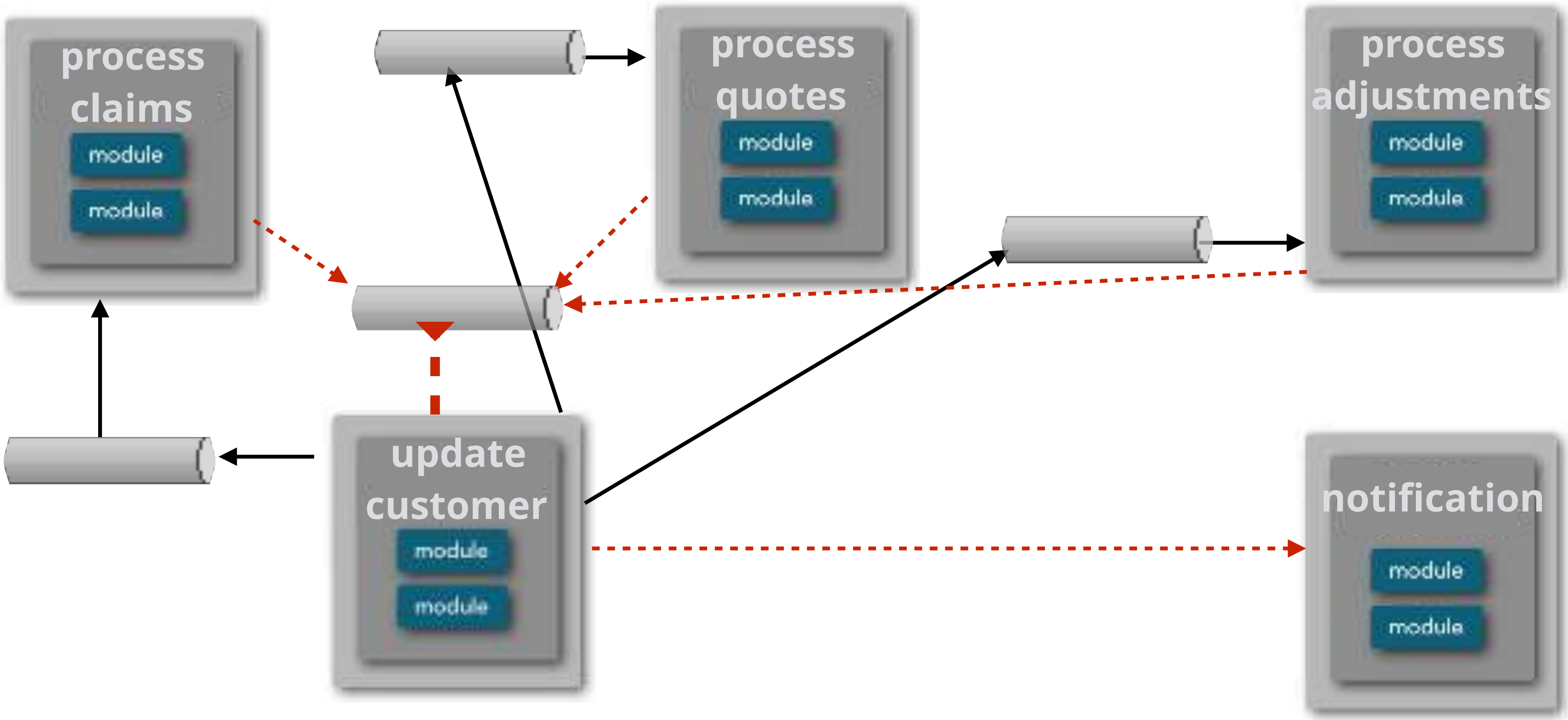
microservices architecture

service orchestration

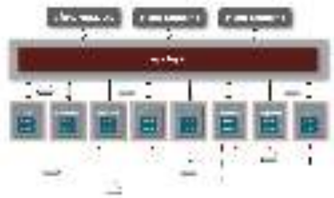


microservices architecture

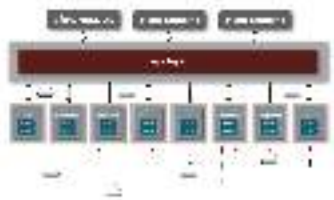
service orchestration



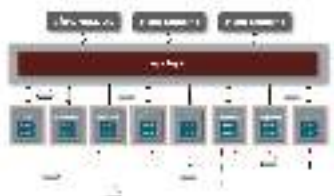
microservices architecture



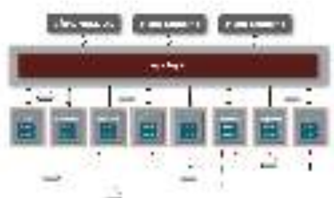
reporting techniques



eventual consistency patterns

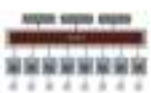
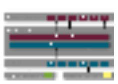


performance tuning



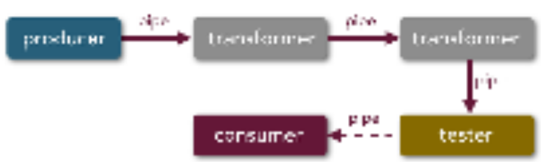
reactive architecture patterns

microservices architecture

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

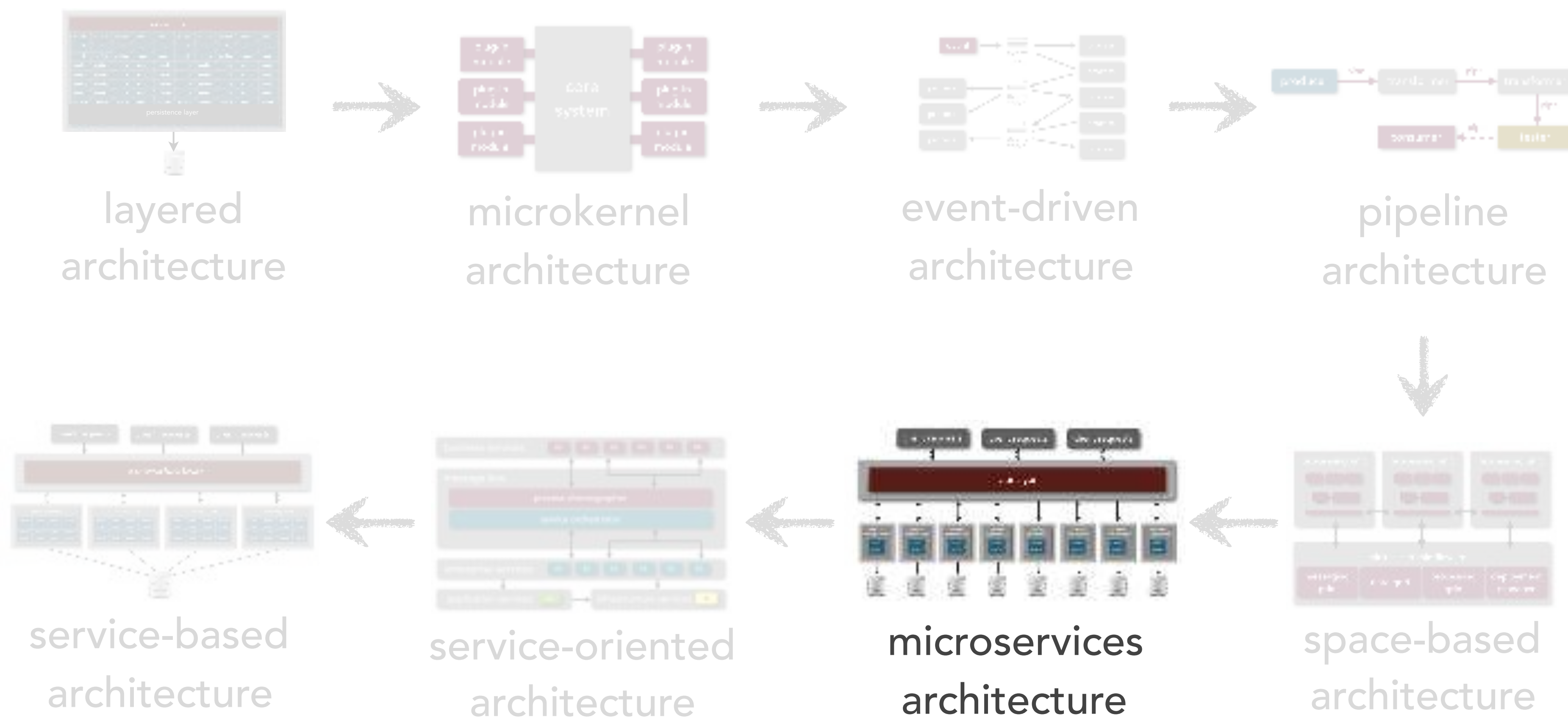
pipeline architecture

pipeline vs. event-driven

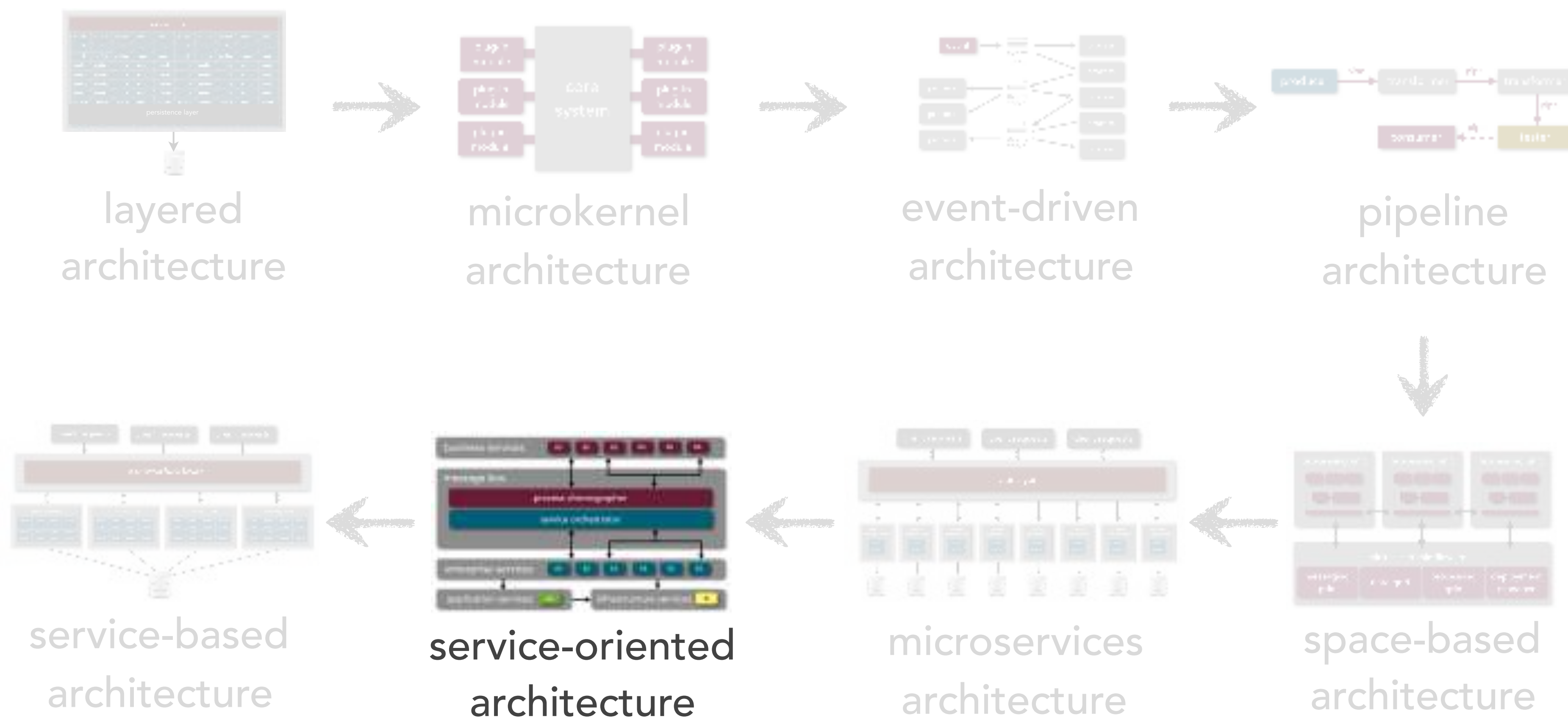


pipeline	event-driven
synchronous data filtering	asynchronous event processing
always unidirectional	can be request/reply
simple single purpose filters	complex multi-purpose processors

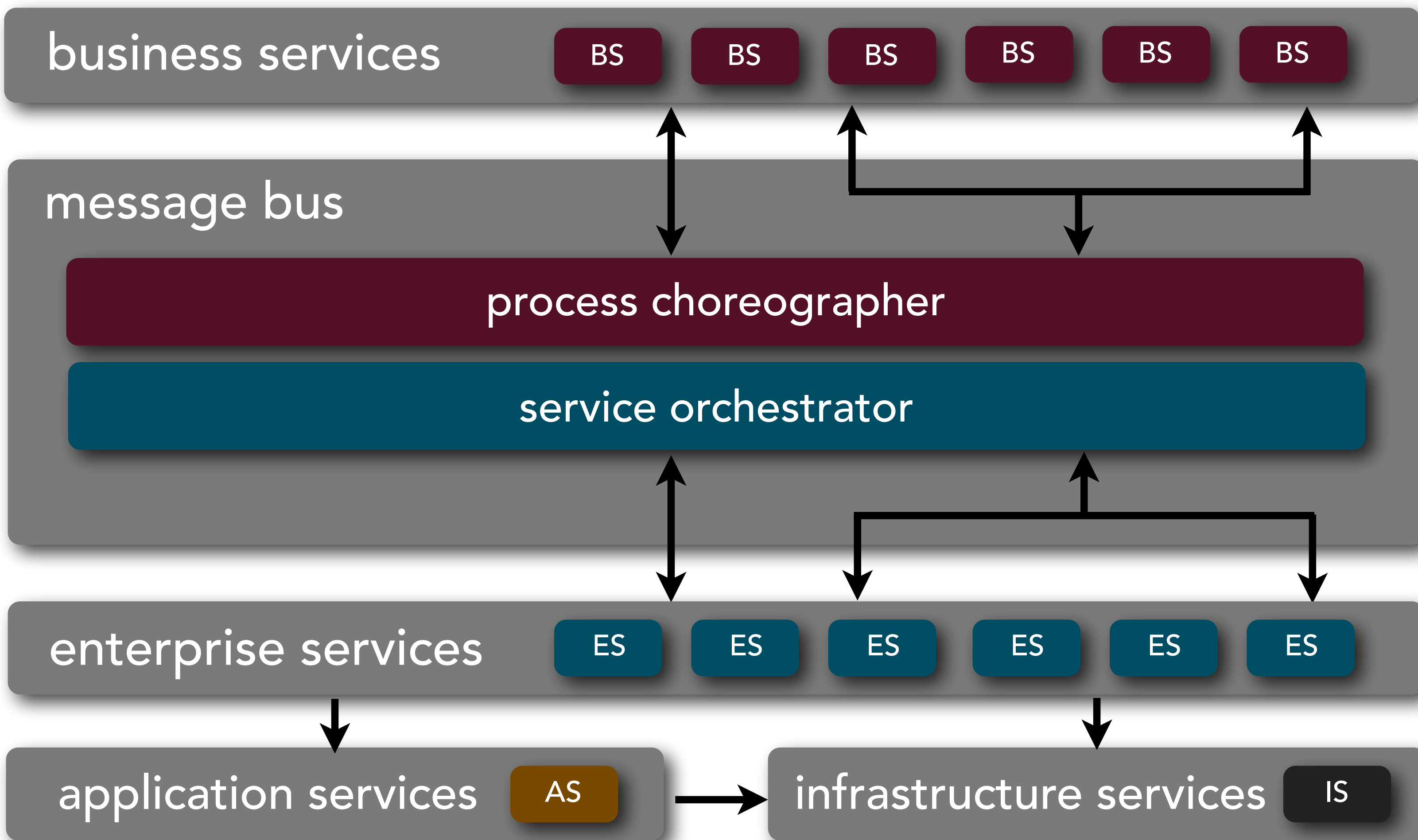
architecture pattern roadmap



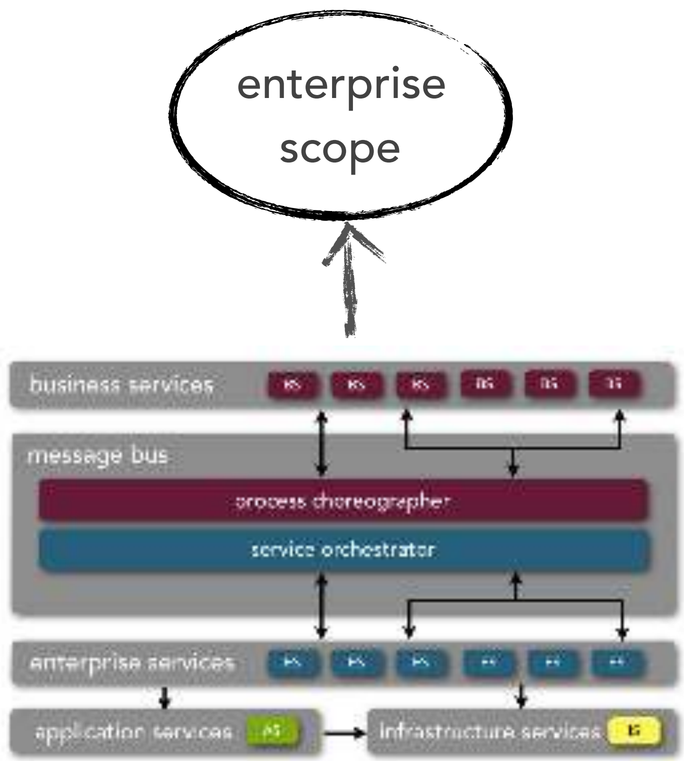
architecture pattern roadmap



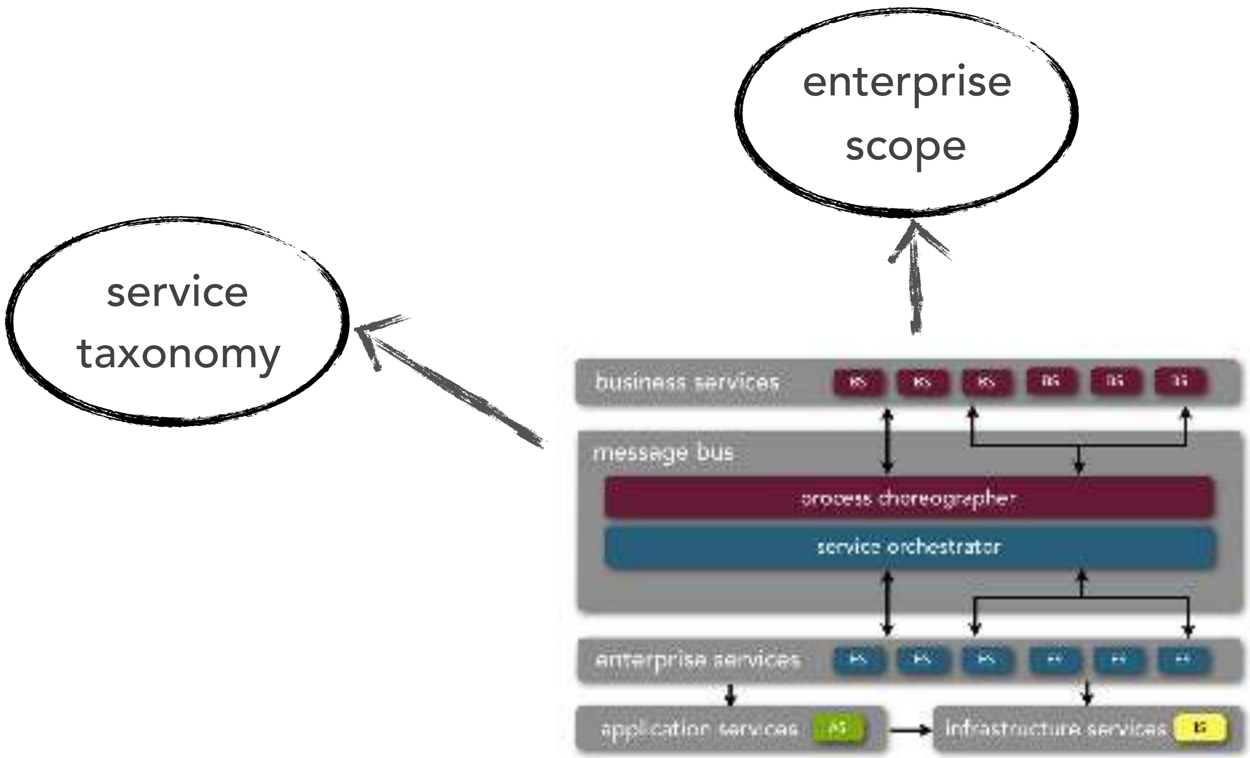
service-oriented architecture



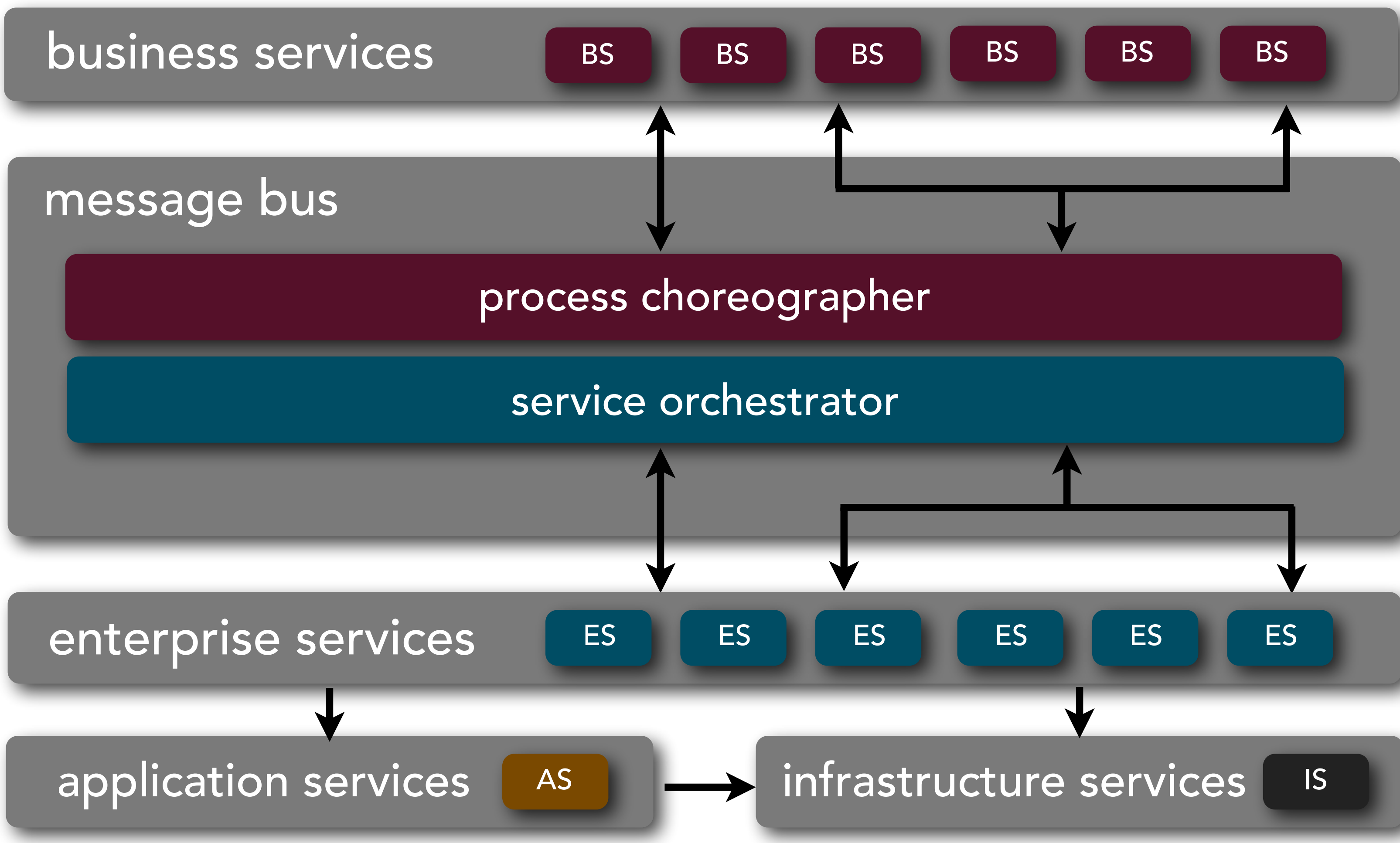
service-oriented architecture



service-oriented architecture

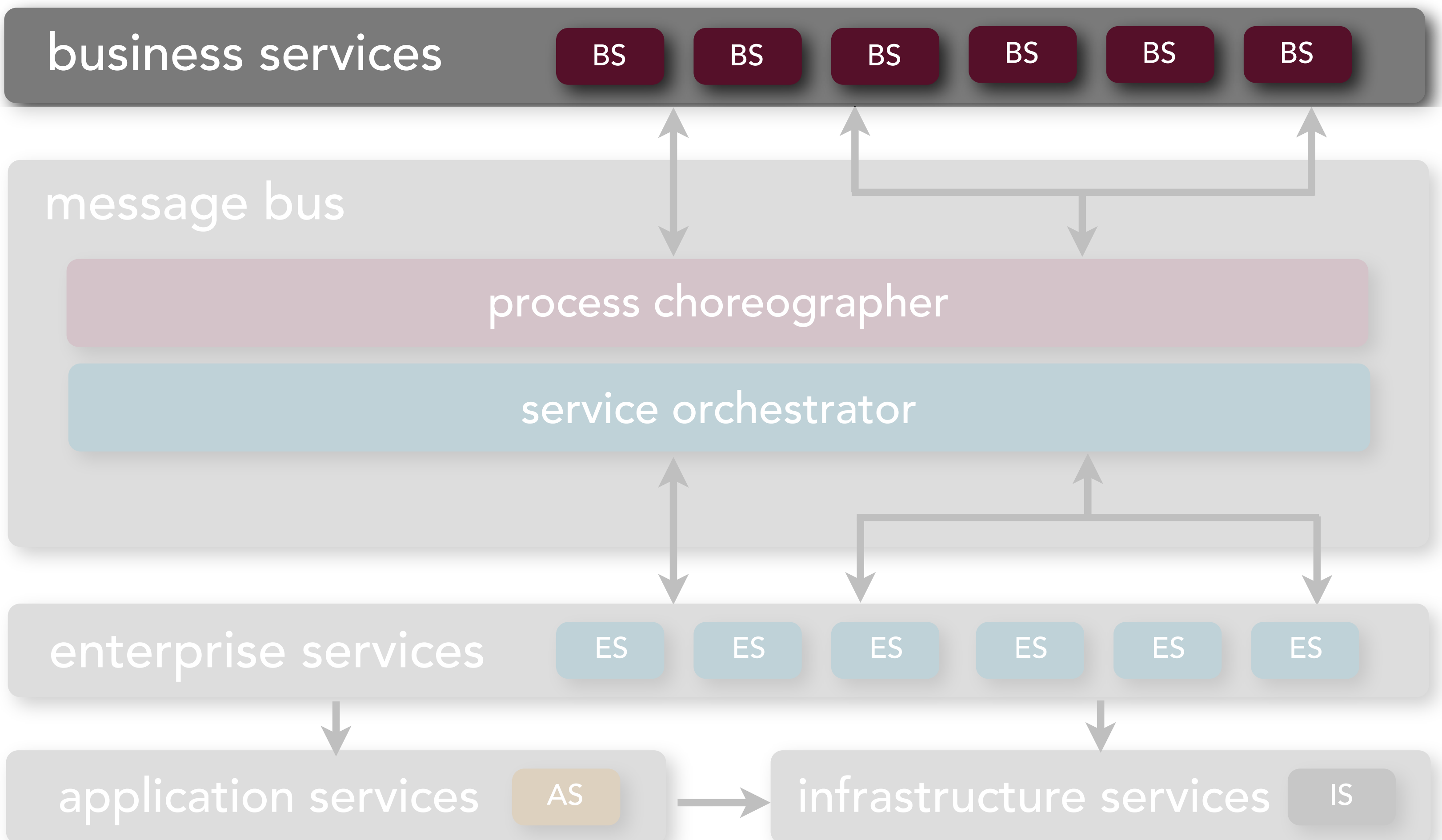


service-oriented architecture



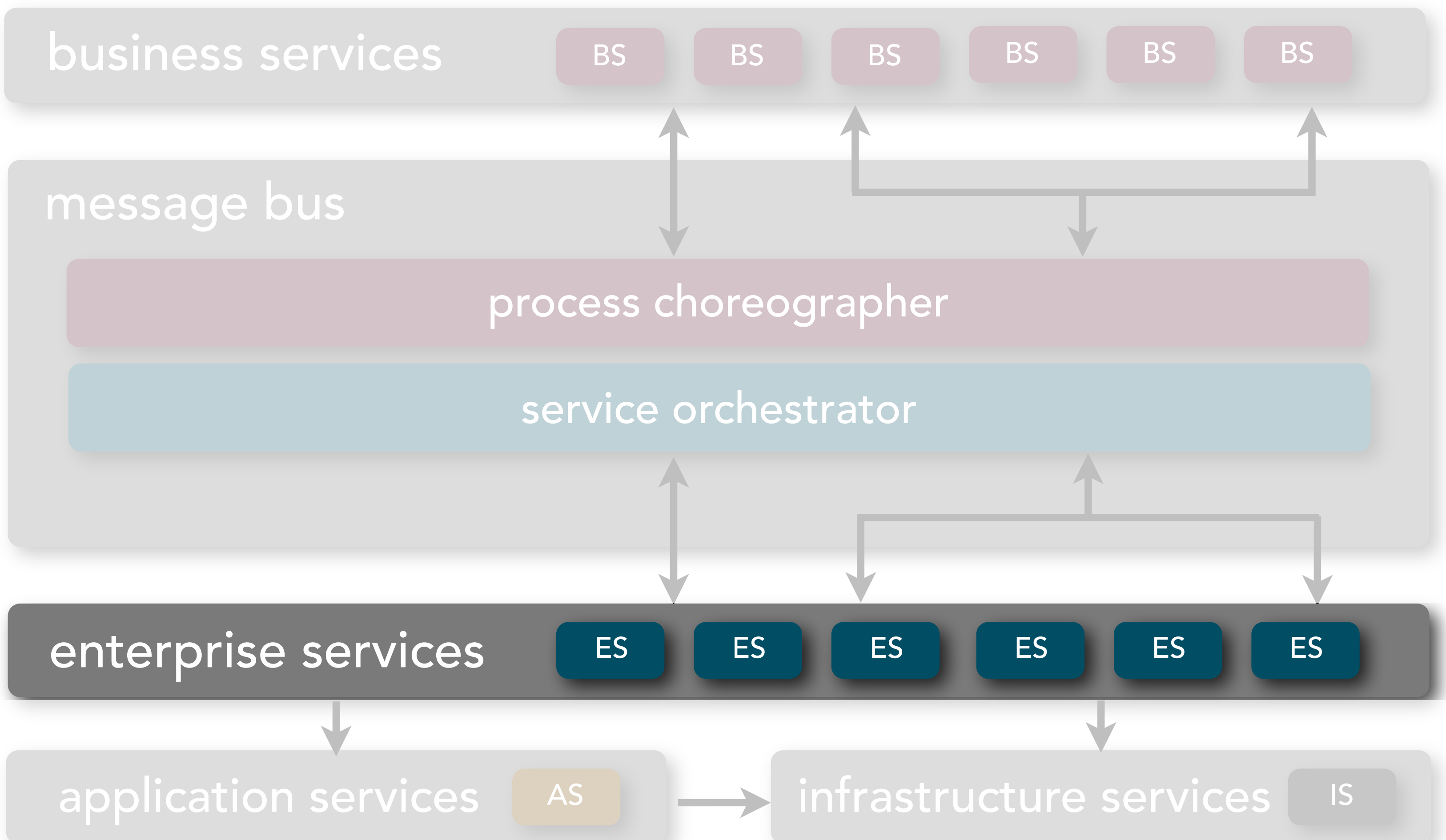
service-oriented architecture

business services



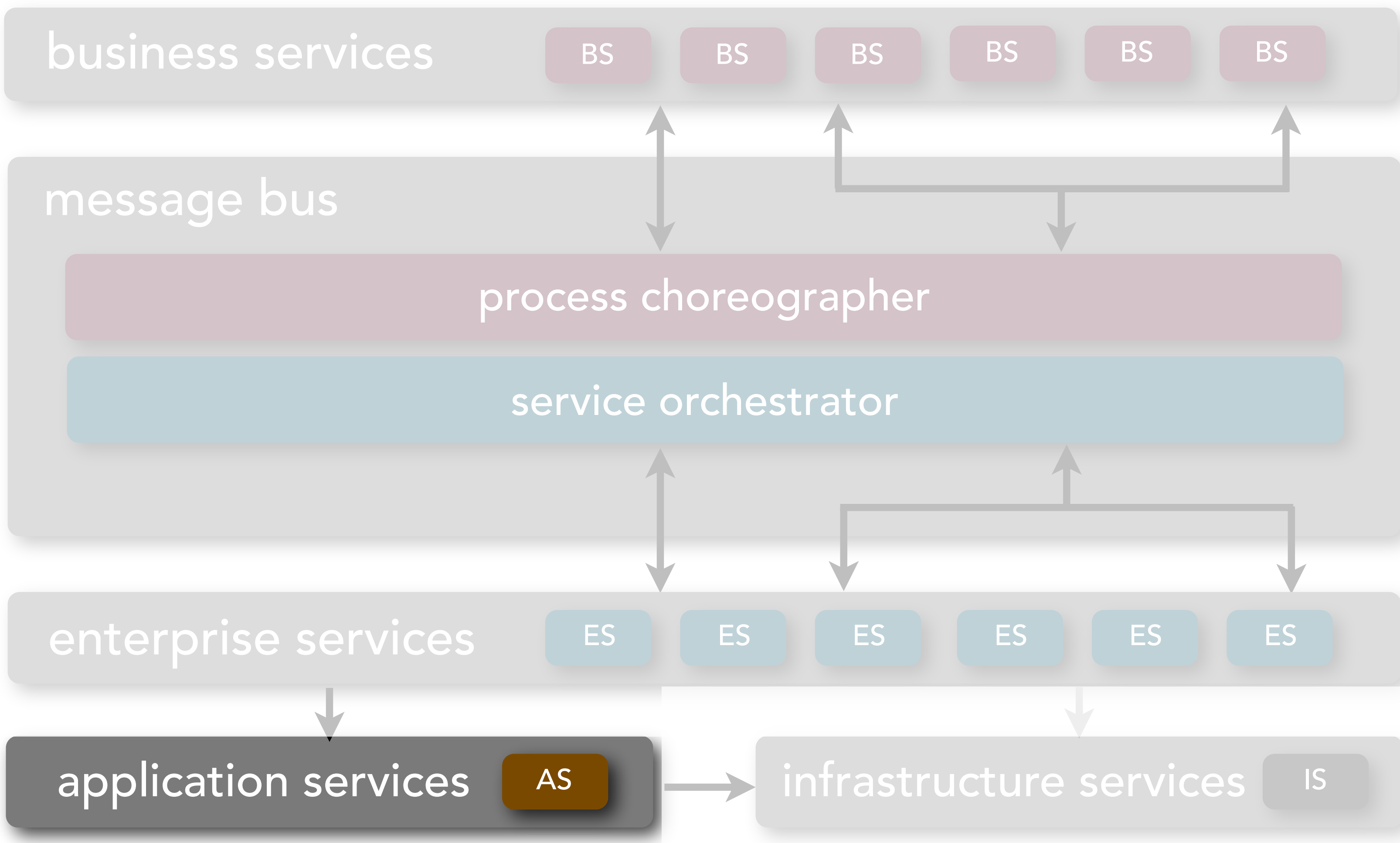
service-oriented architecture

enterprise services



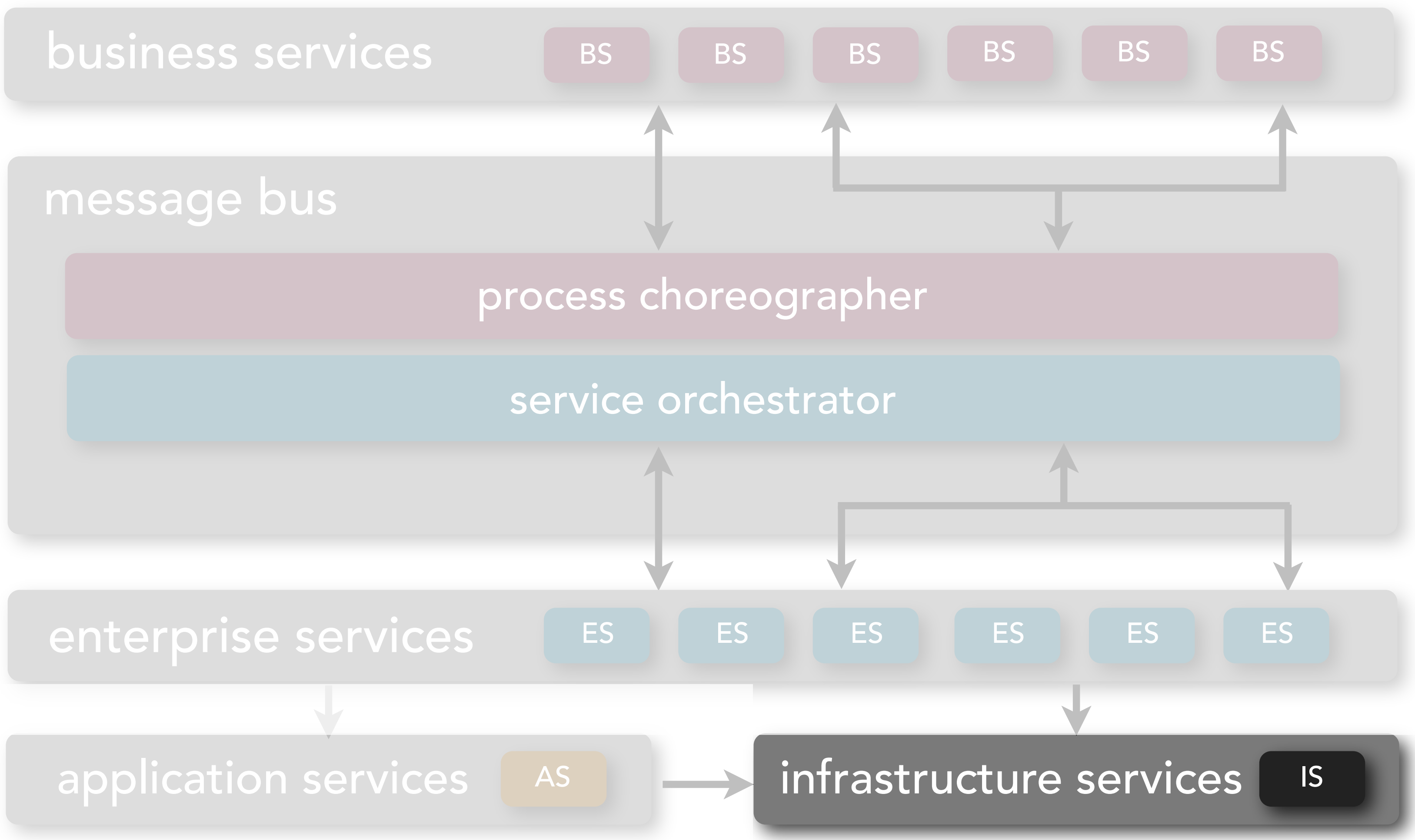
service-oriented architecture

application services

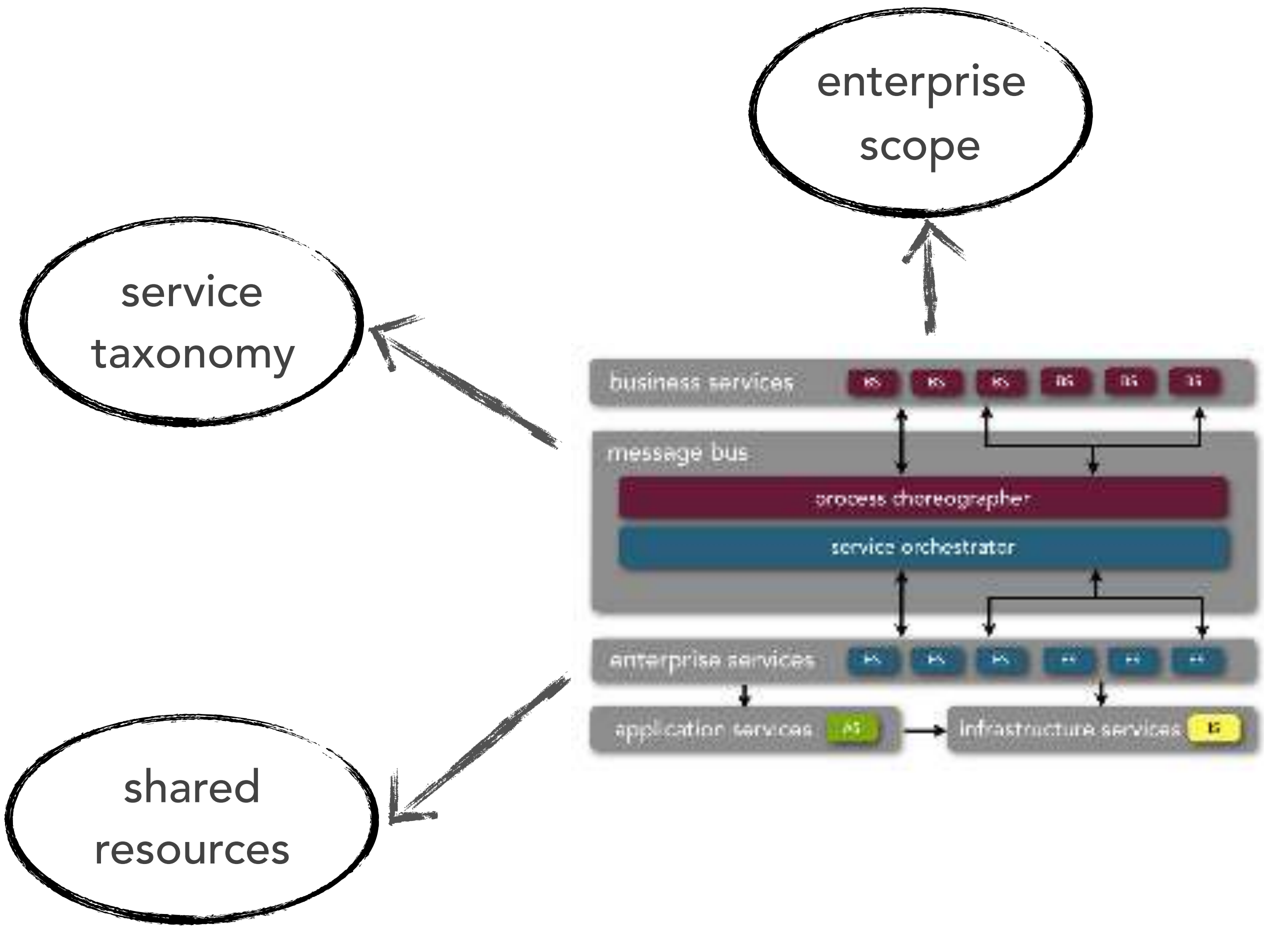


service-oriented architecture

infrastructure services

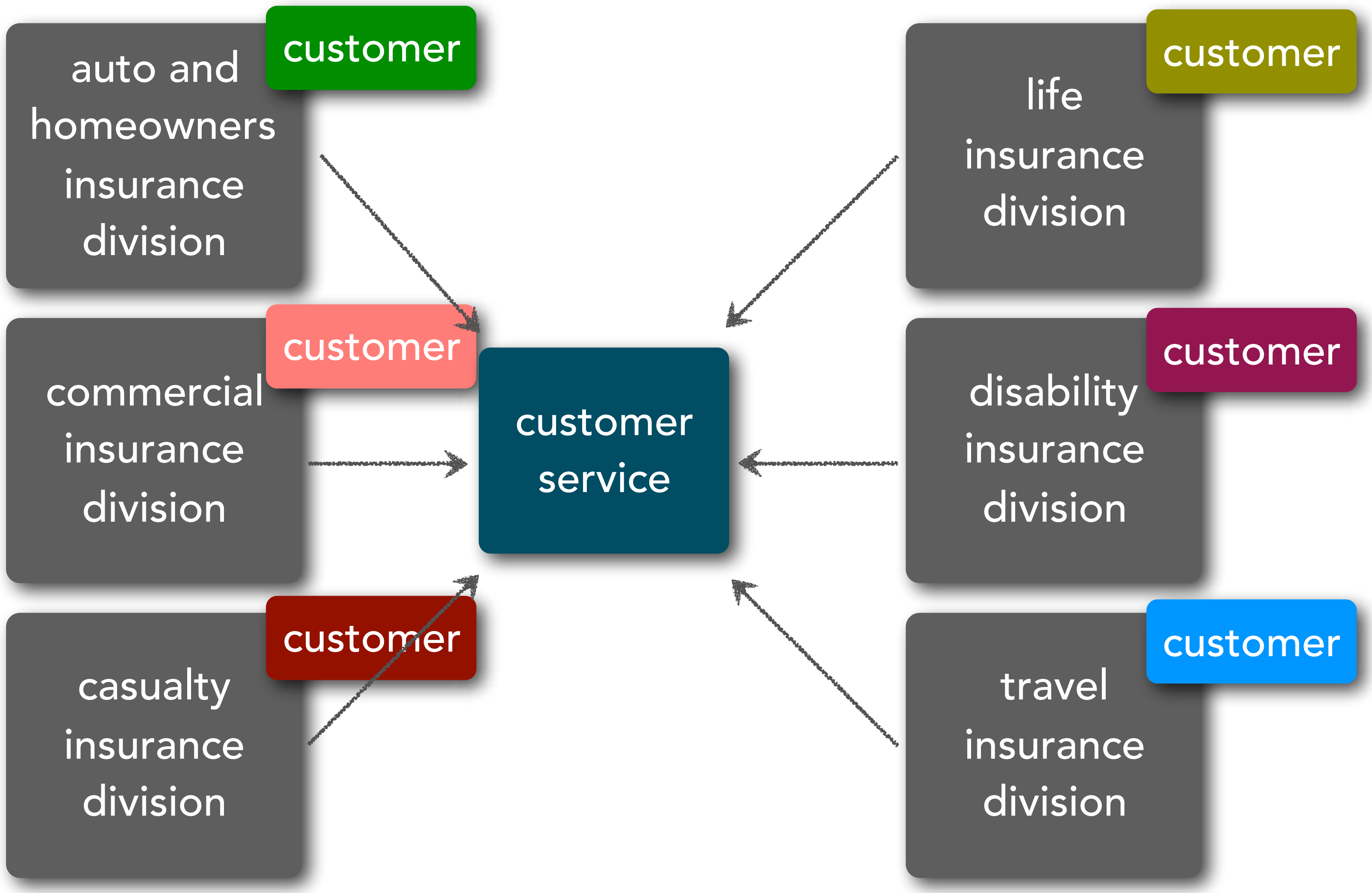


service-oriented architecture

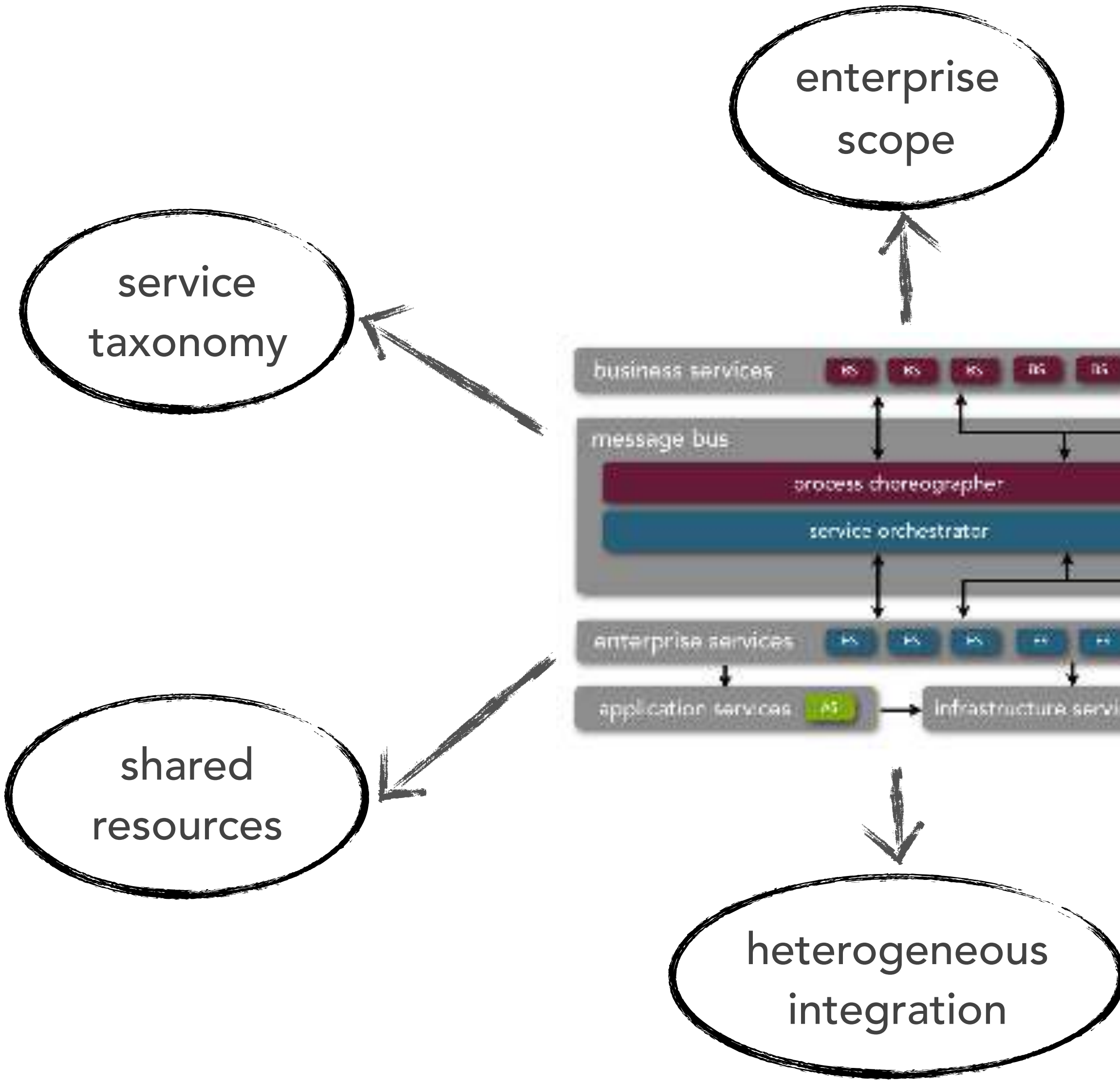


service-oriented architecture

shared resources

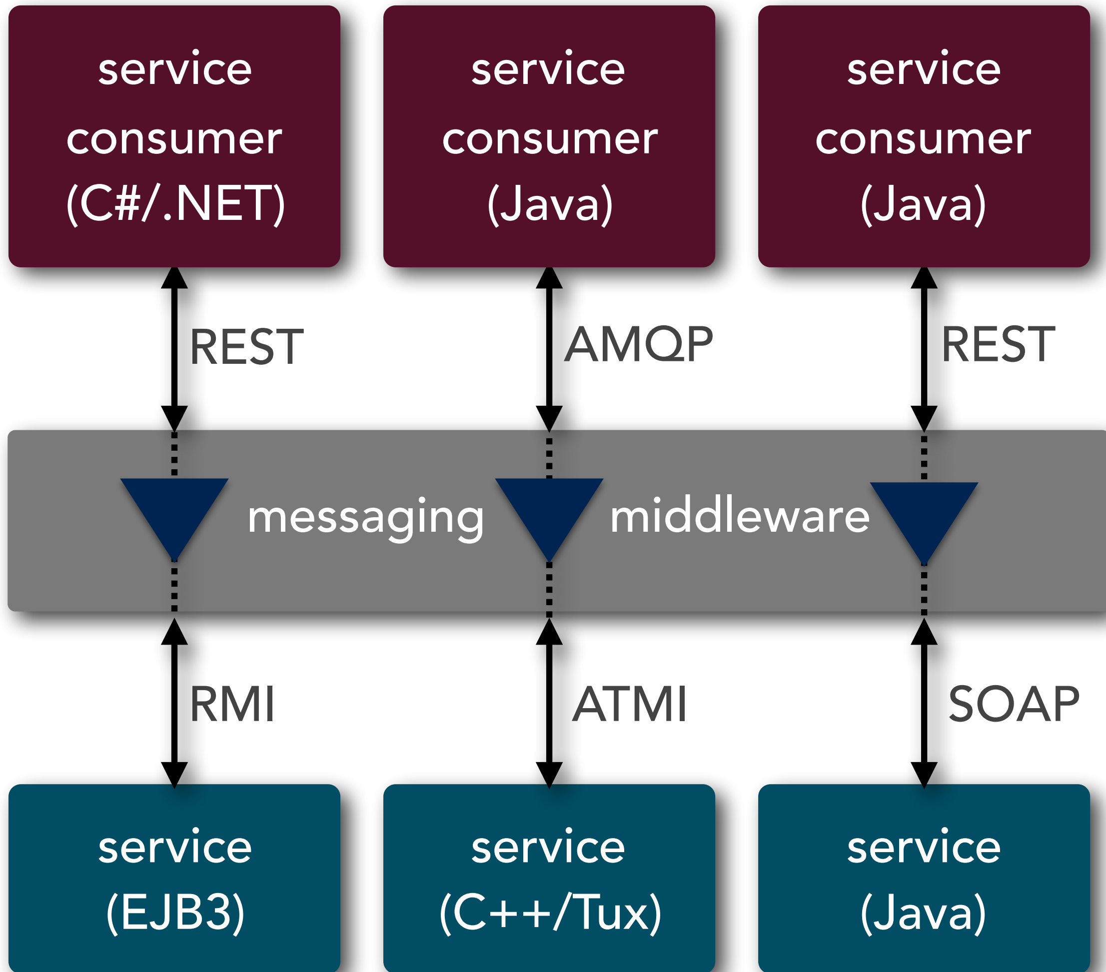


service-oriented architecture

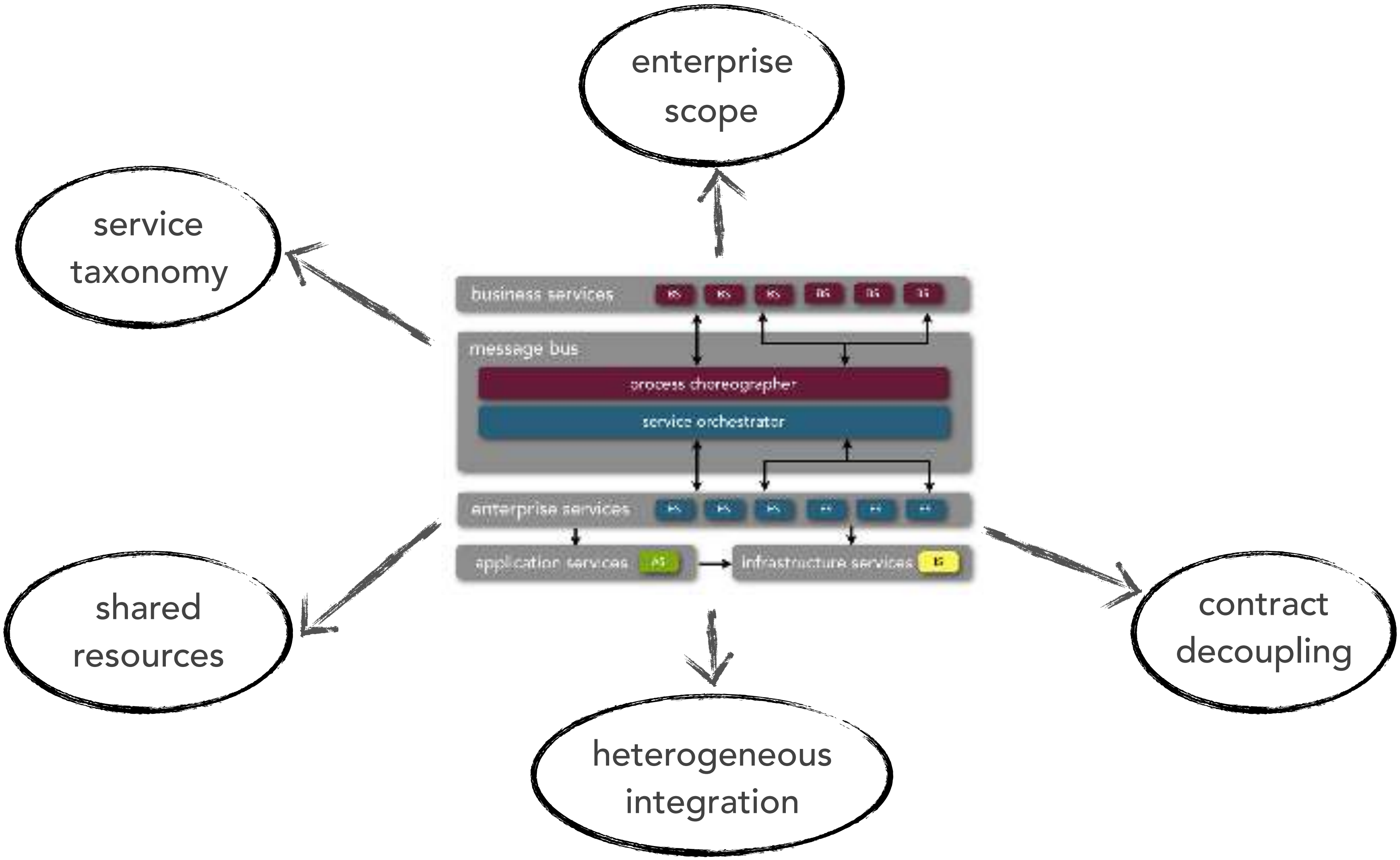


service-oriented architecture

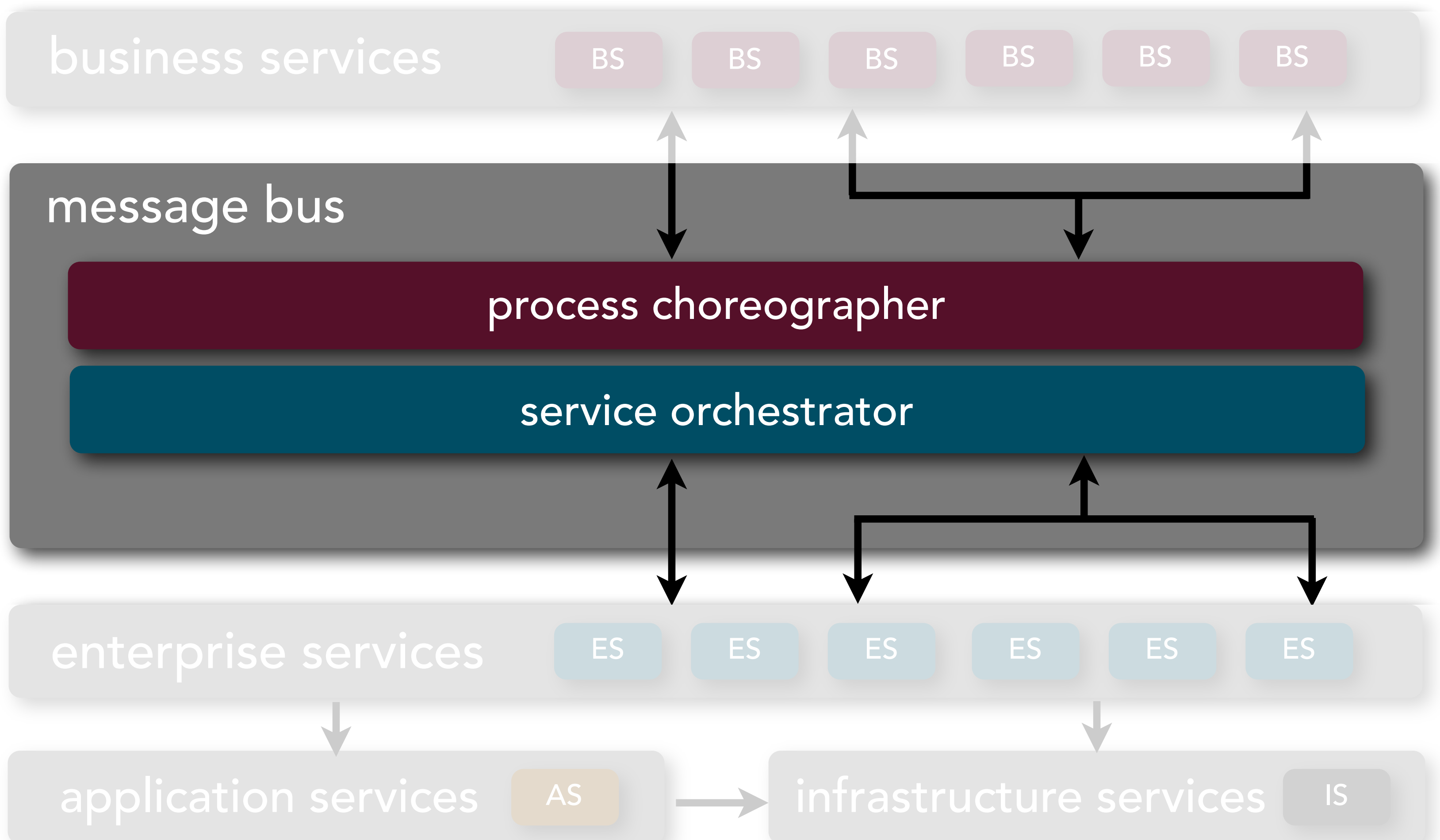
protocol-agnostic heterogeneous interoperability



service-oriented architecture

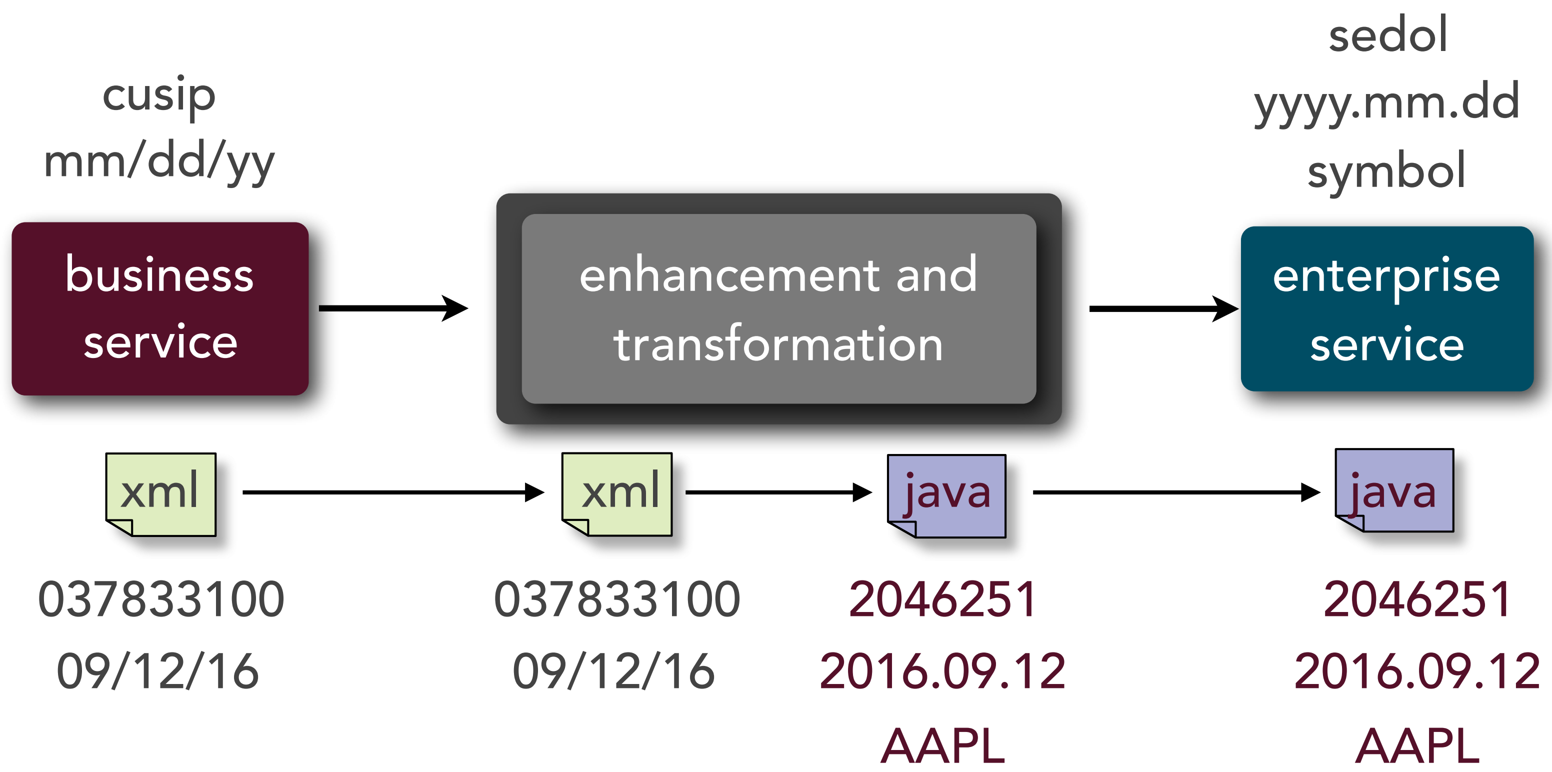


service-oriented architecture



service-oriented architecture

contract decoupling

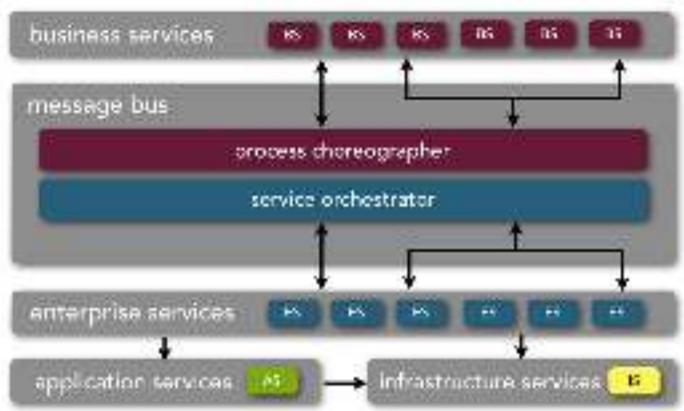


service-oriented architecture

enterprise scope

service taxonomy

abstraction (api layers)

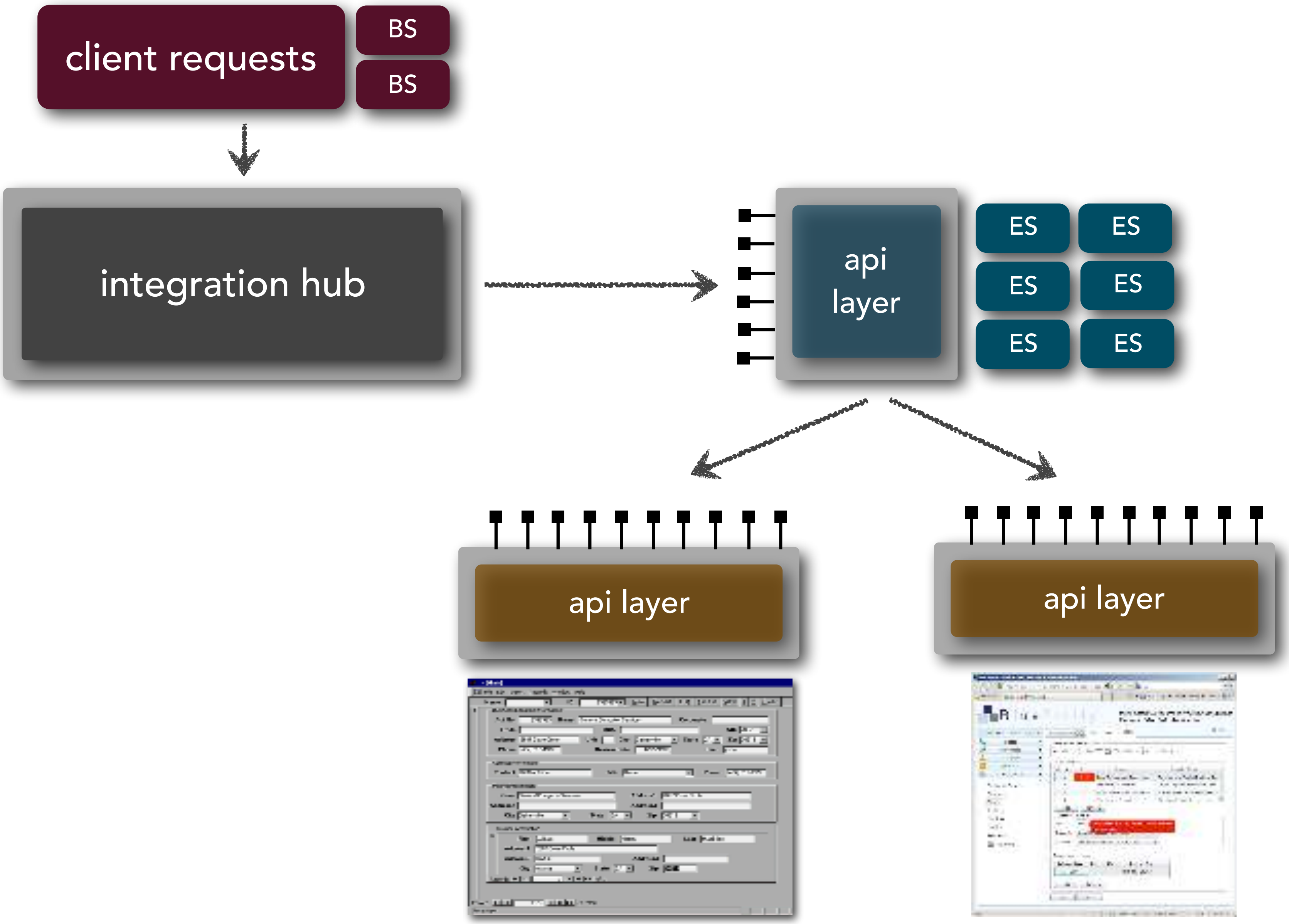


shared resources

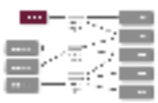
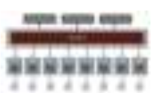
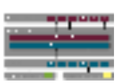
contract decoupling

heterogeneous integration

service-oriented architecture



service-oriented architecture

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

architecture pattern roadmap

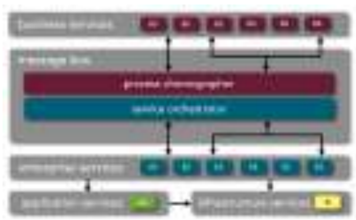
layered
architecture

microkernel
architecture

event-driven
architecture

pipeline
architecture

service-based
architecture

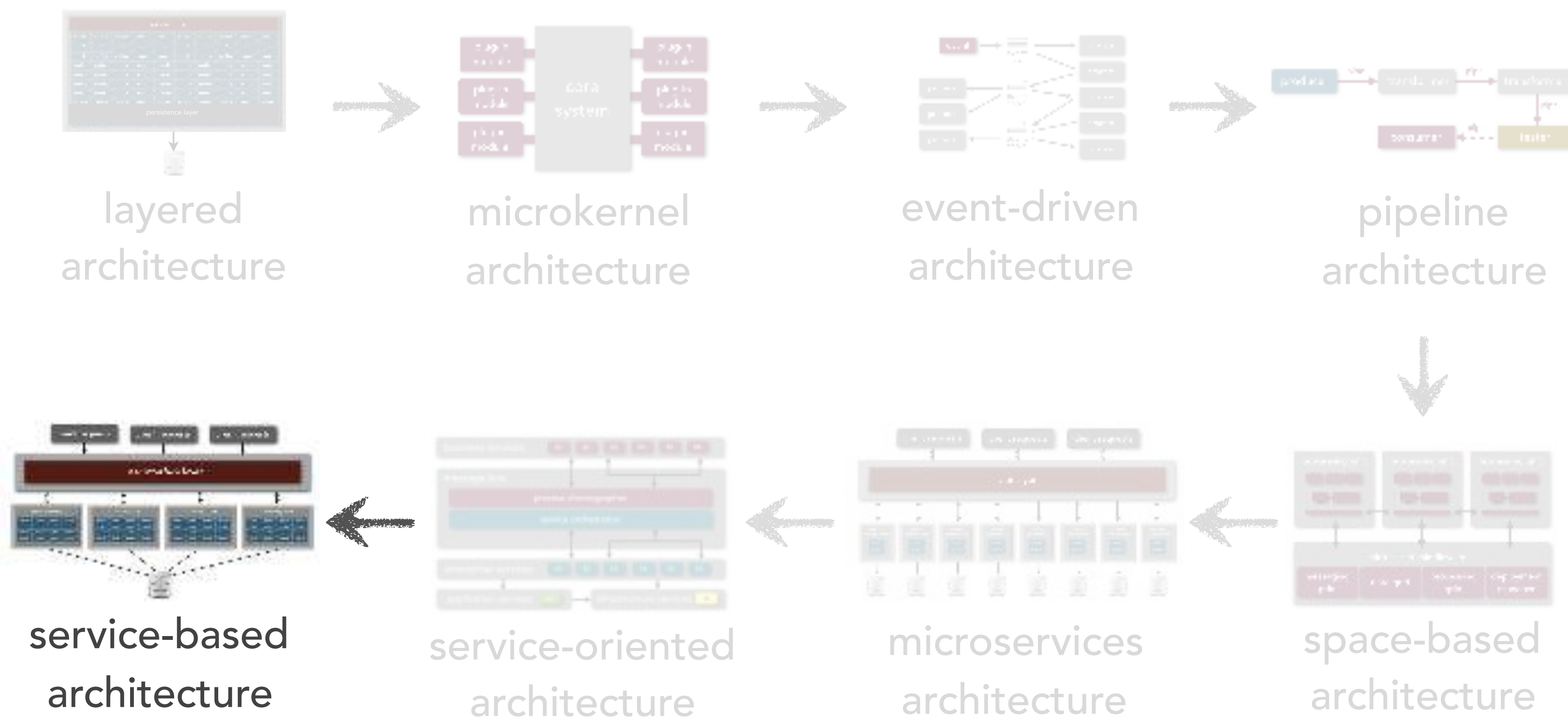


service-oriented
architecture

microservices
architecture

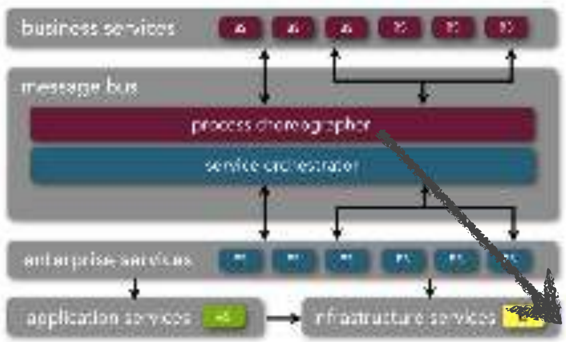
space-based
architecture

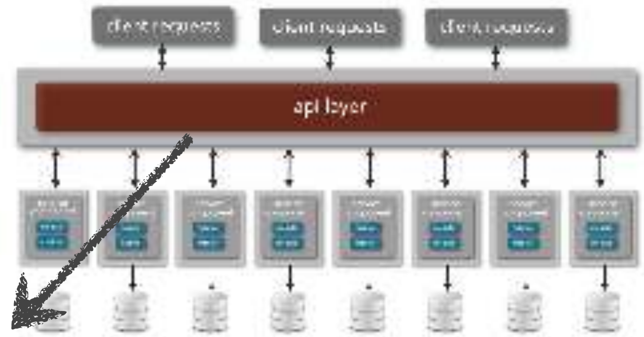
architecture pattern roadmap



service-based architecture

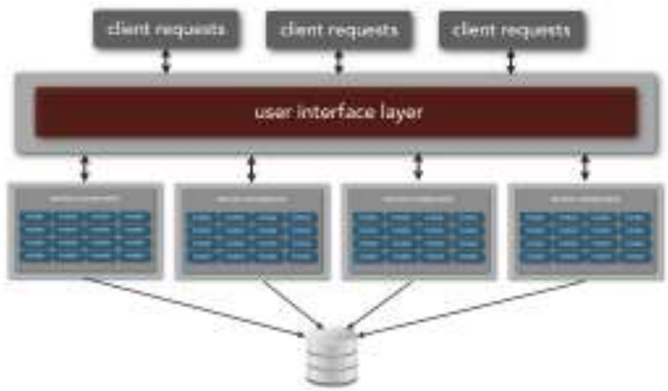
is there a middle ground?





service-ori

architec



roservices

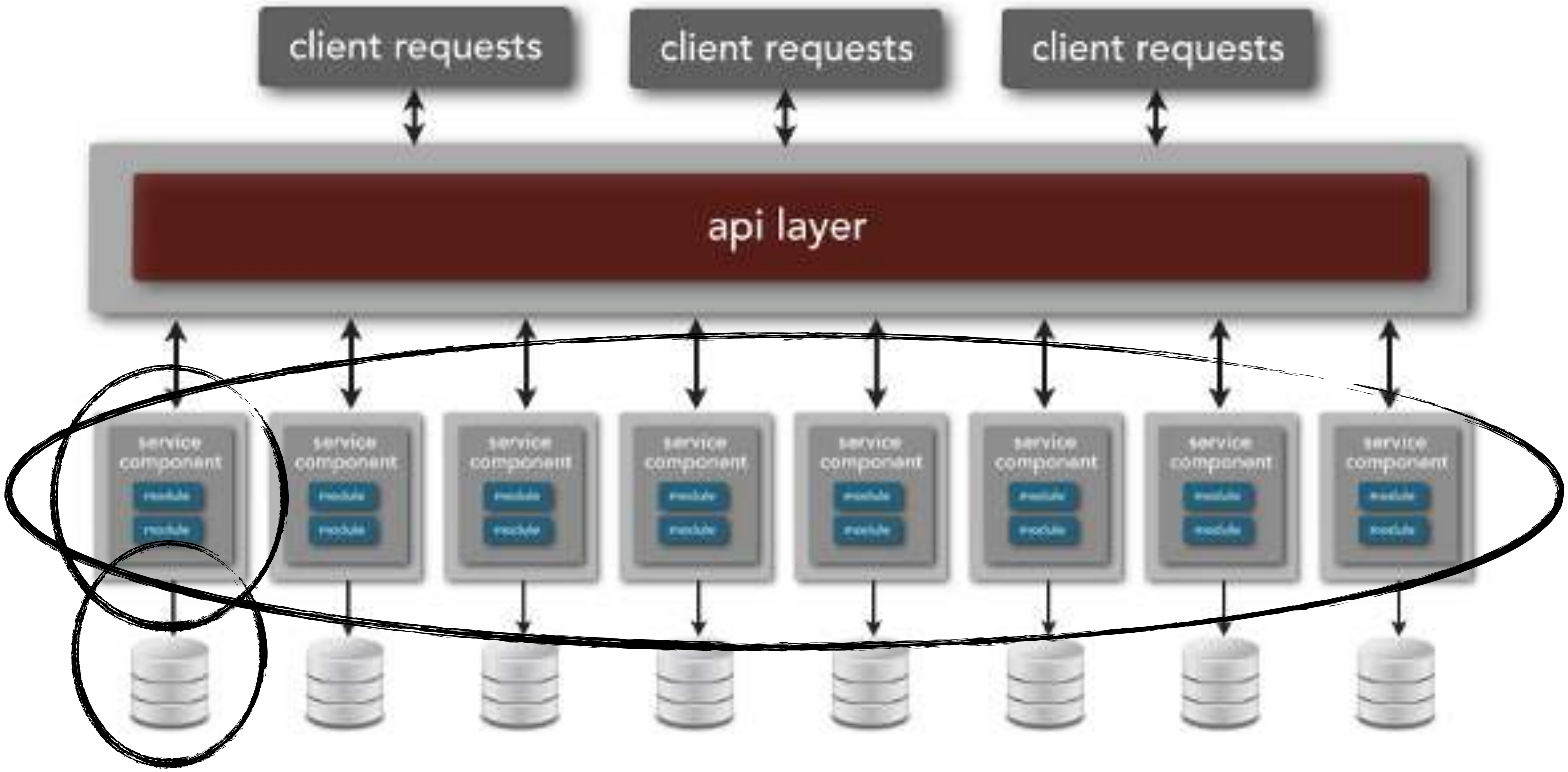
:hitecture

service-based

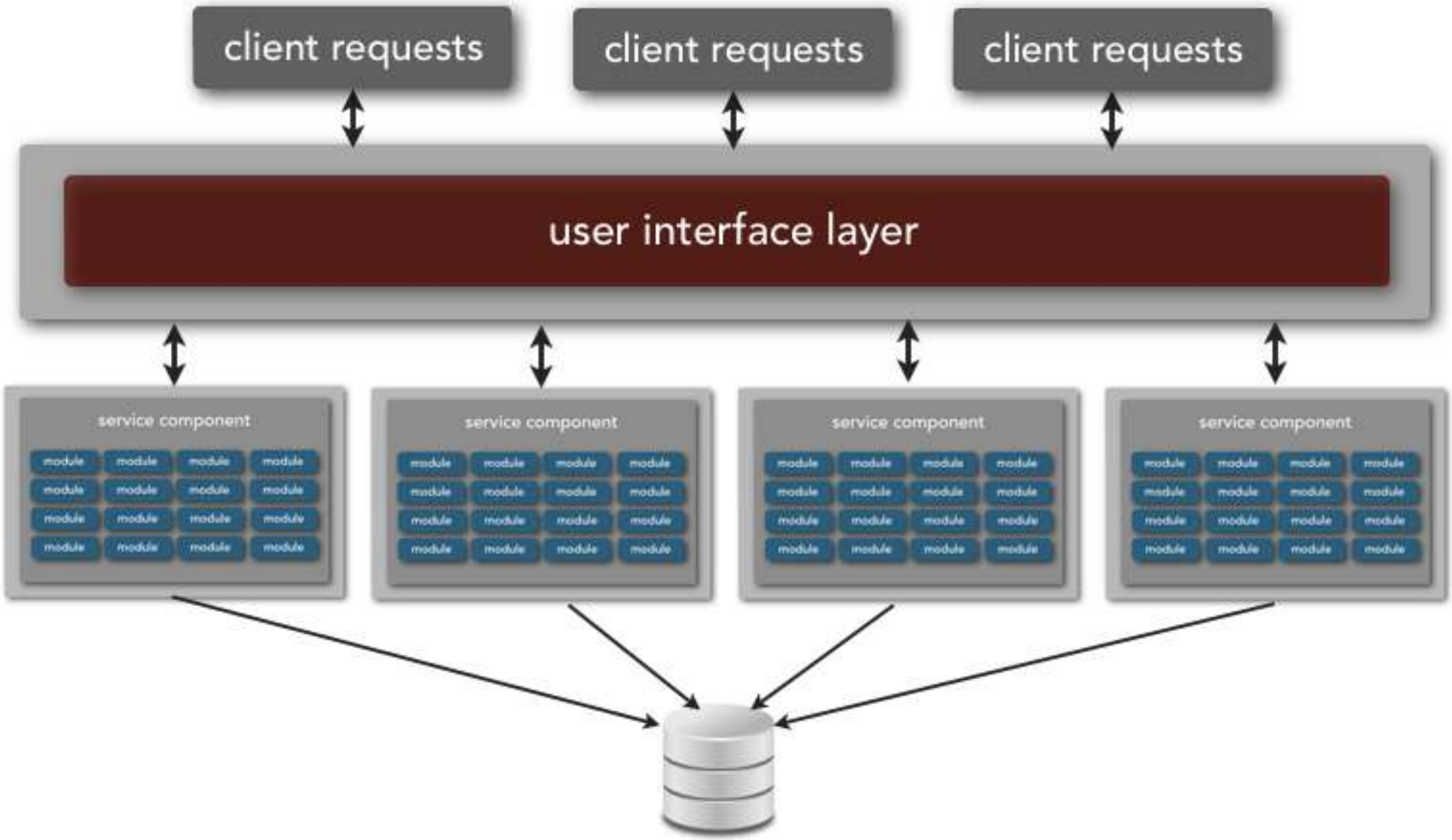
architecture

service-based architecture

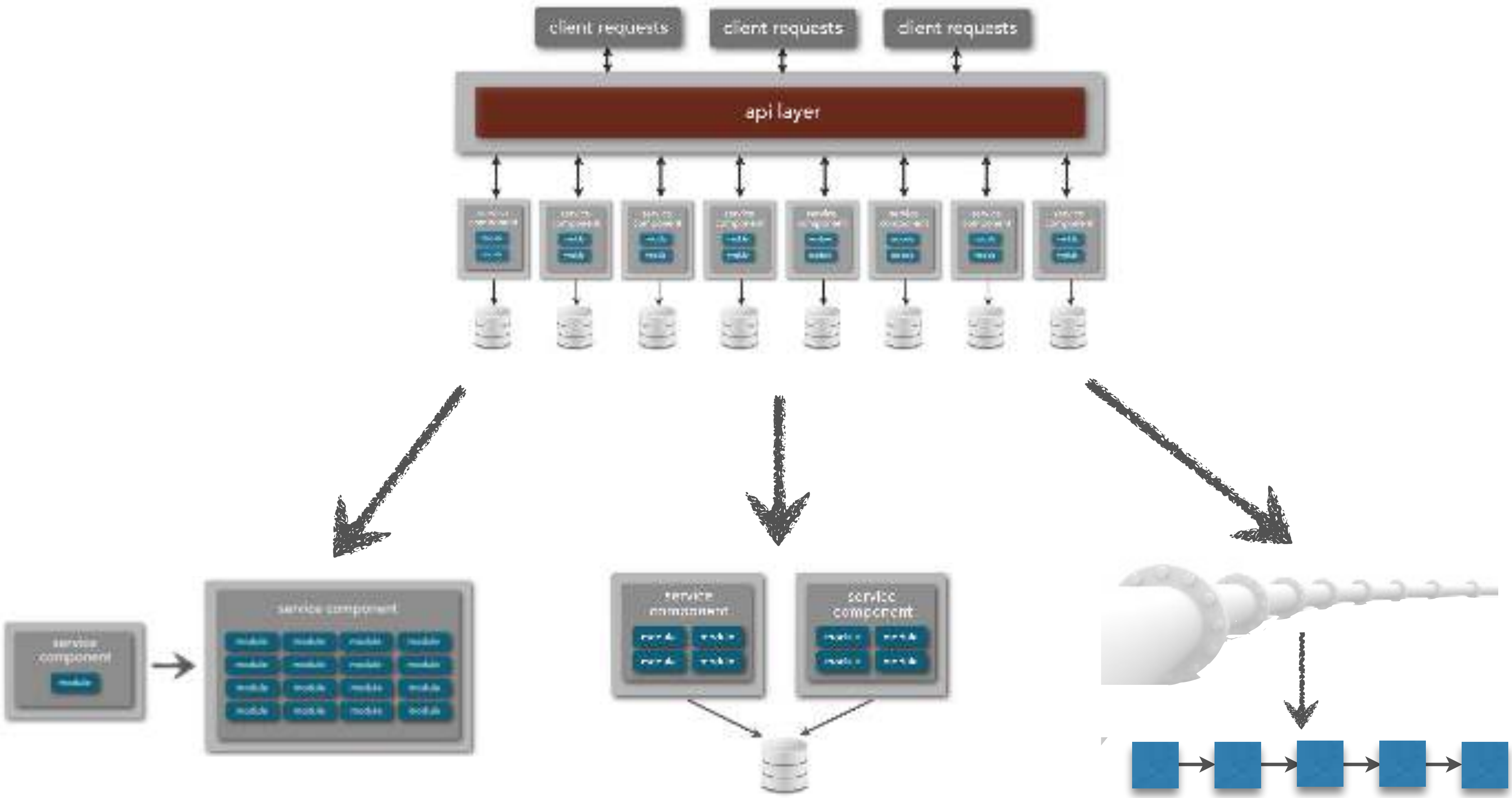
business applications?



service-based architecture



service-based architecture



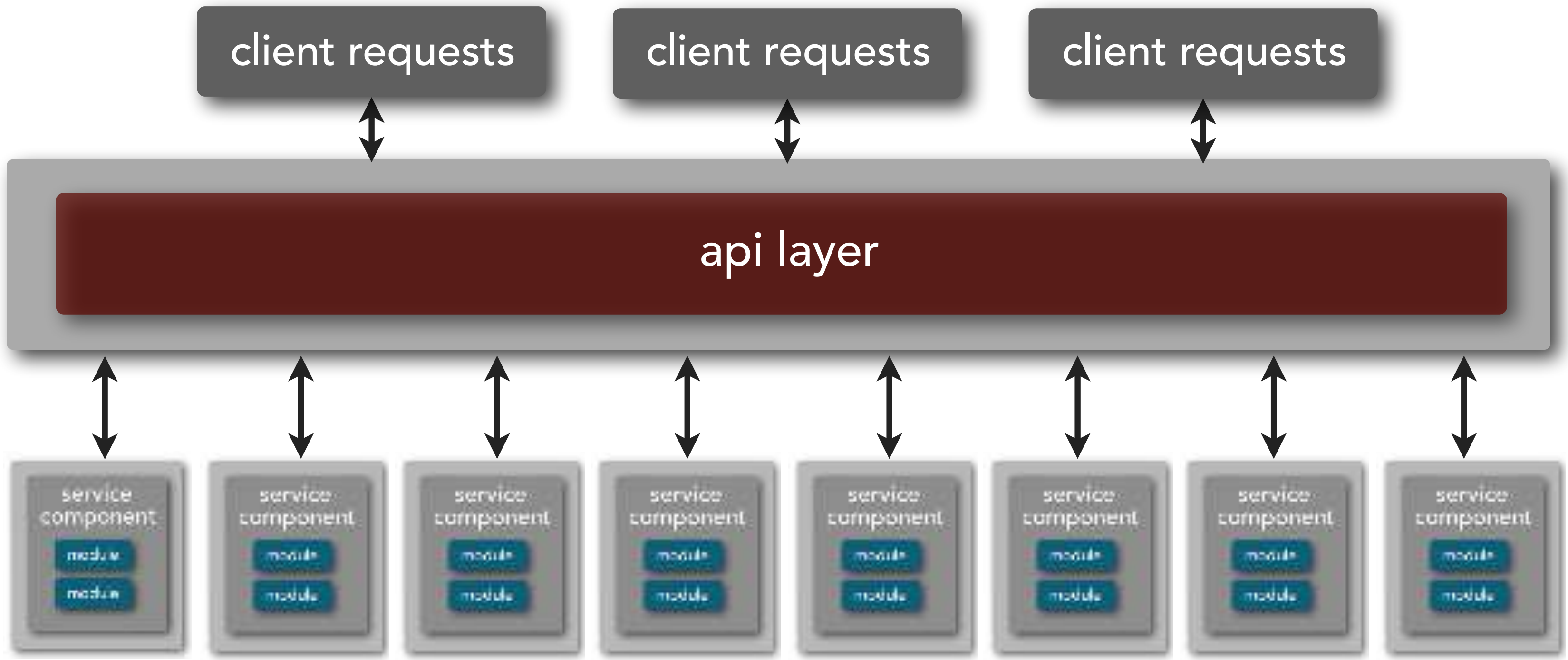
service granularity

database scope

deployment pipeline

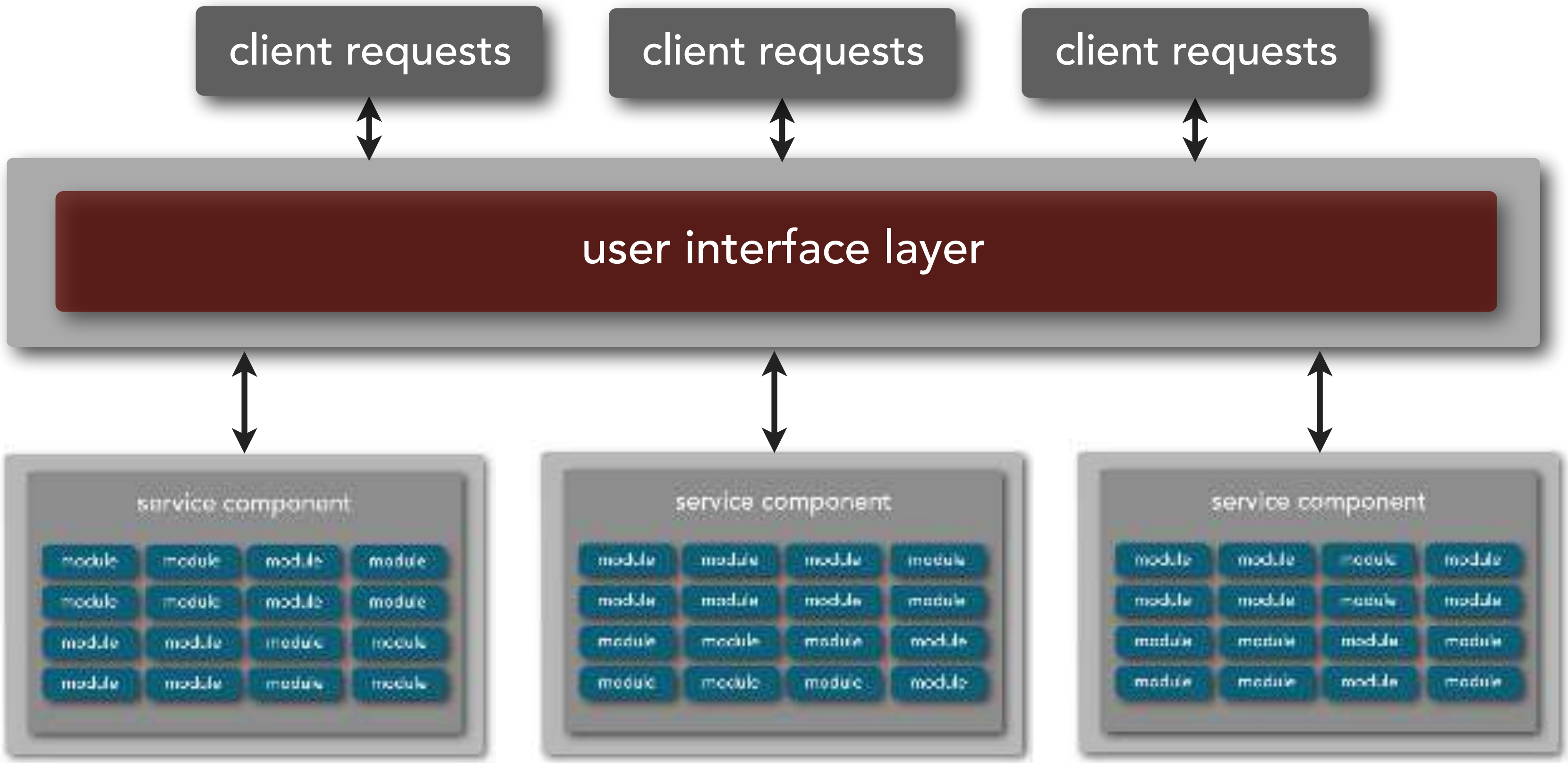
service-based architecture

service granularity



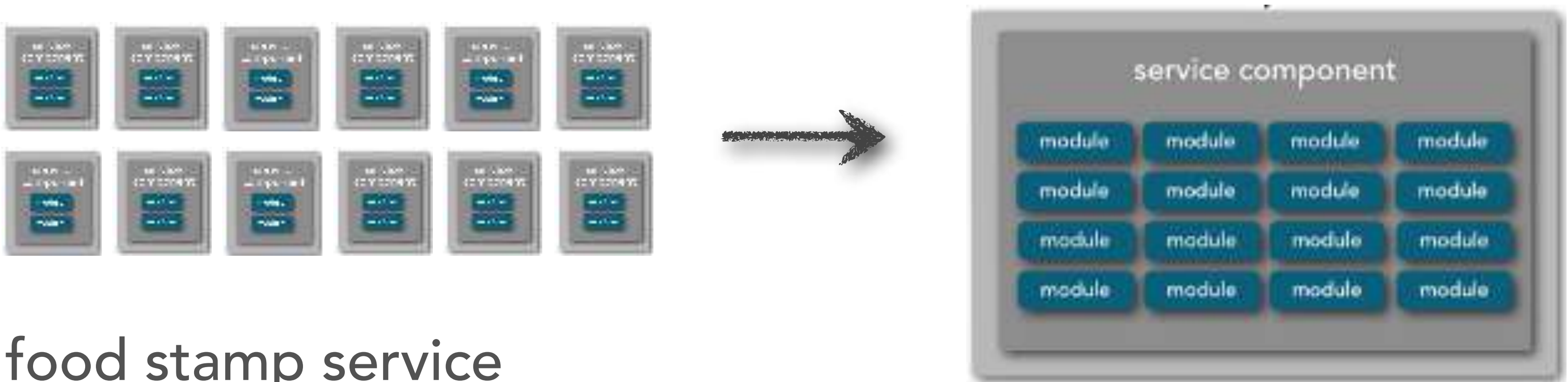
service-based architecture

service granularity



service-based architecture

service granularity

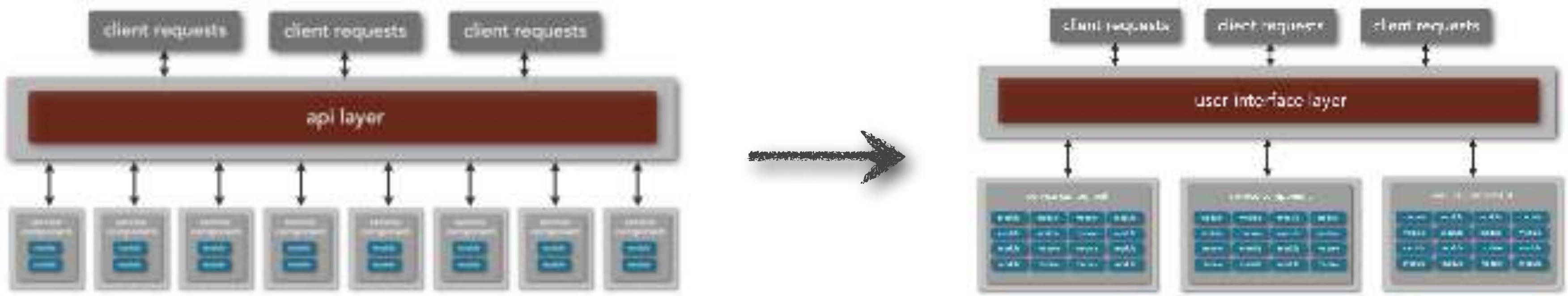


food stamp service
emergency cash service
utility assistance service
child care assist service
health care assist service
nursing facility care service
...

benefit service

service-based architecture

service granularity

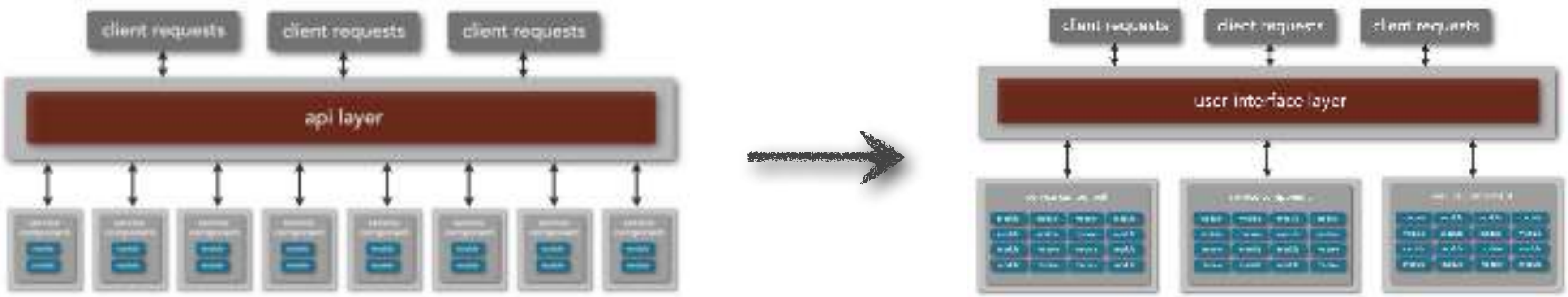


advantages

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

service-based architecture

service granularity

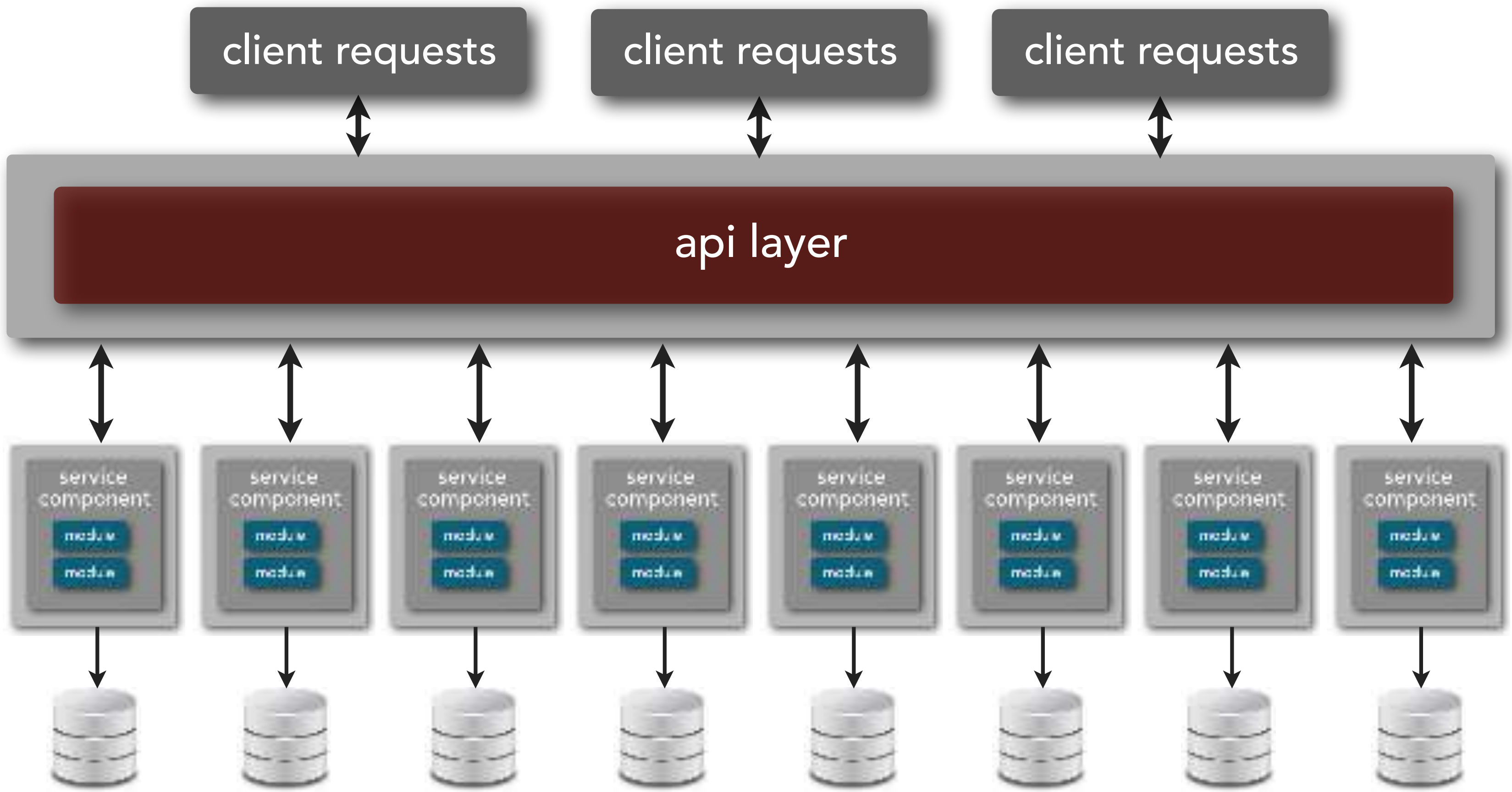


tradeoffs

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

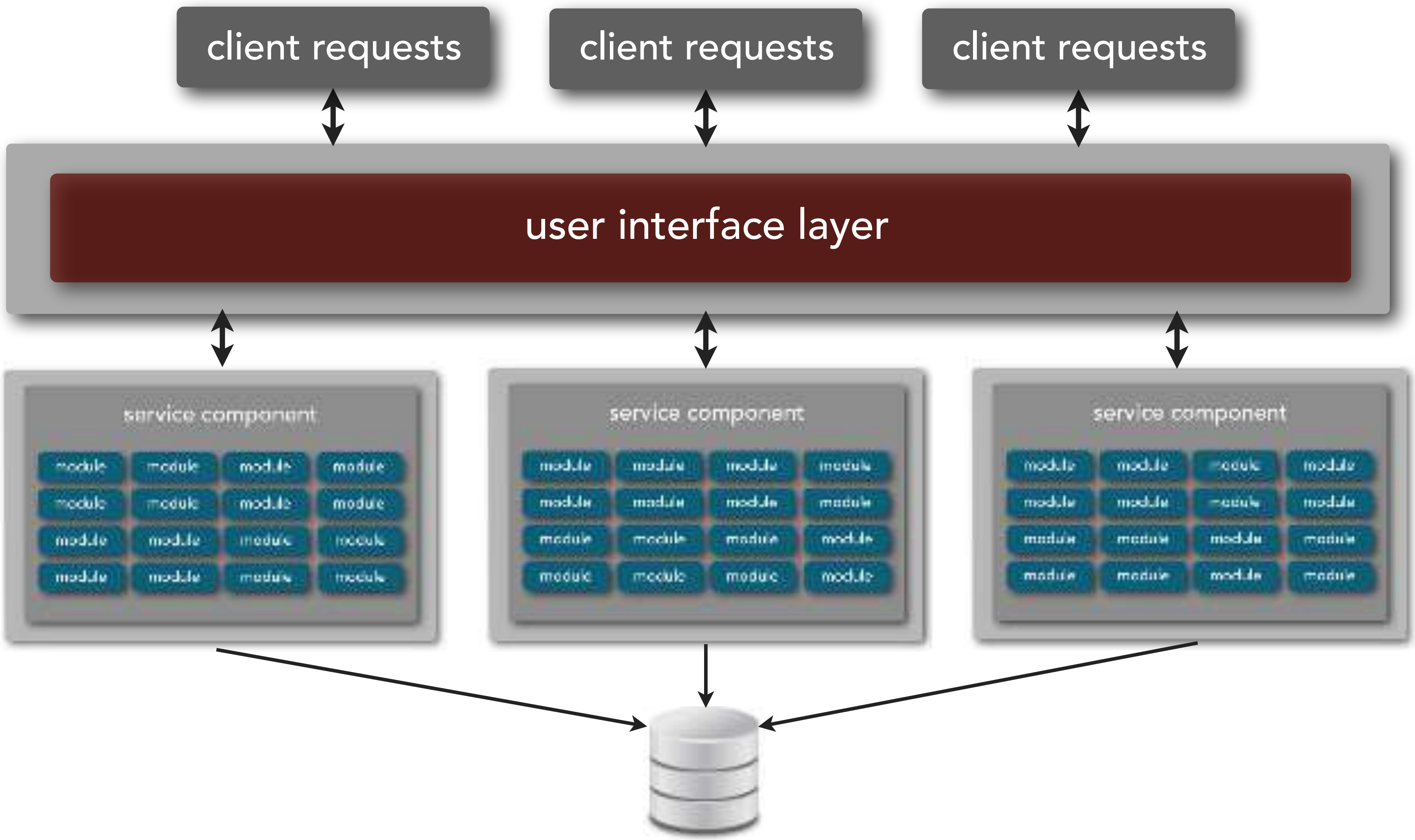
service-based architecture

database scope



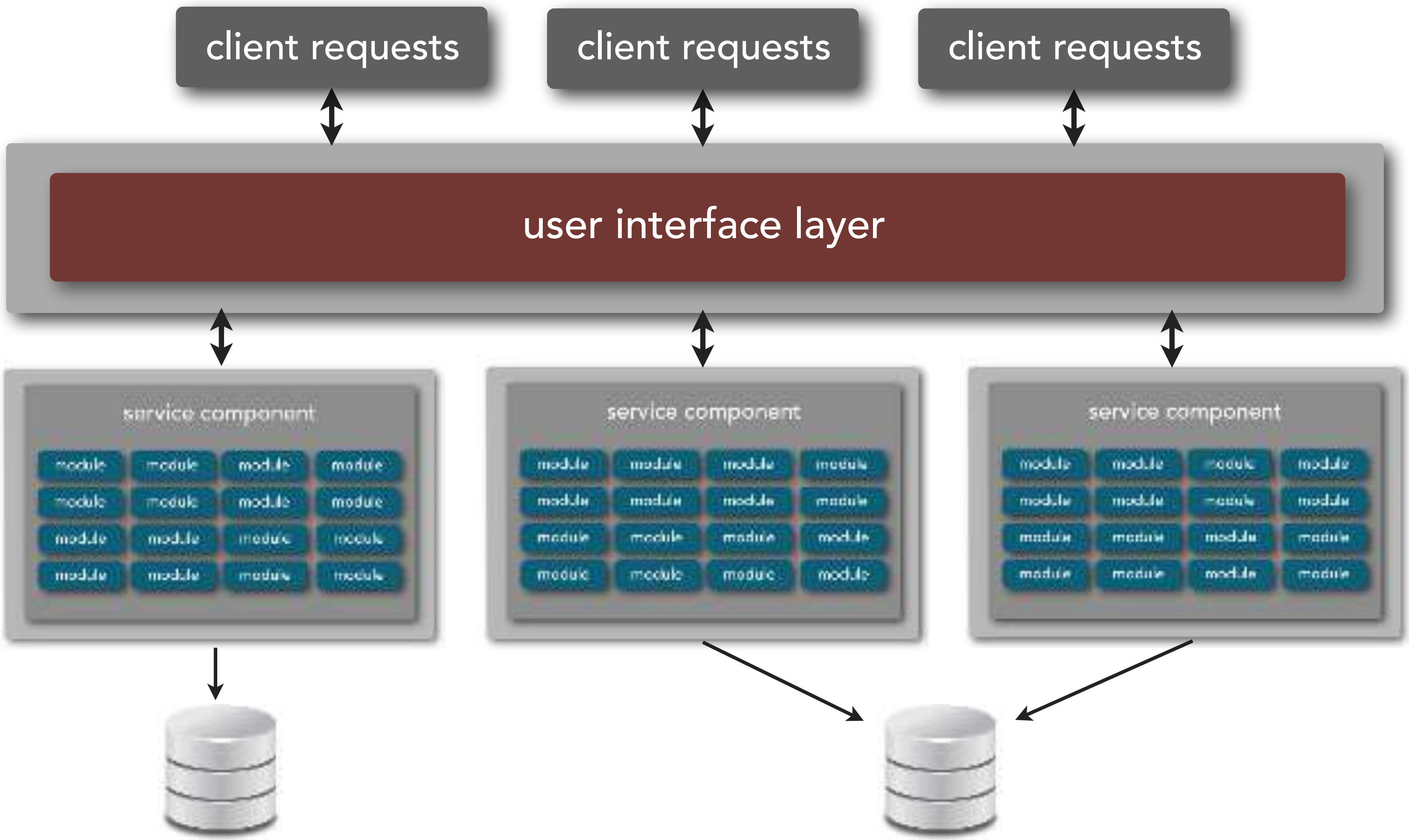
service-based architecture

database scope



service-based architecture

database scope



service-based architecture

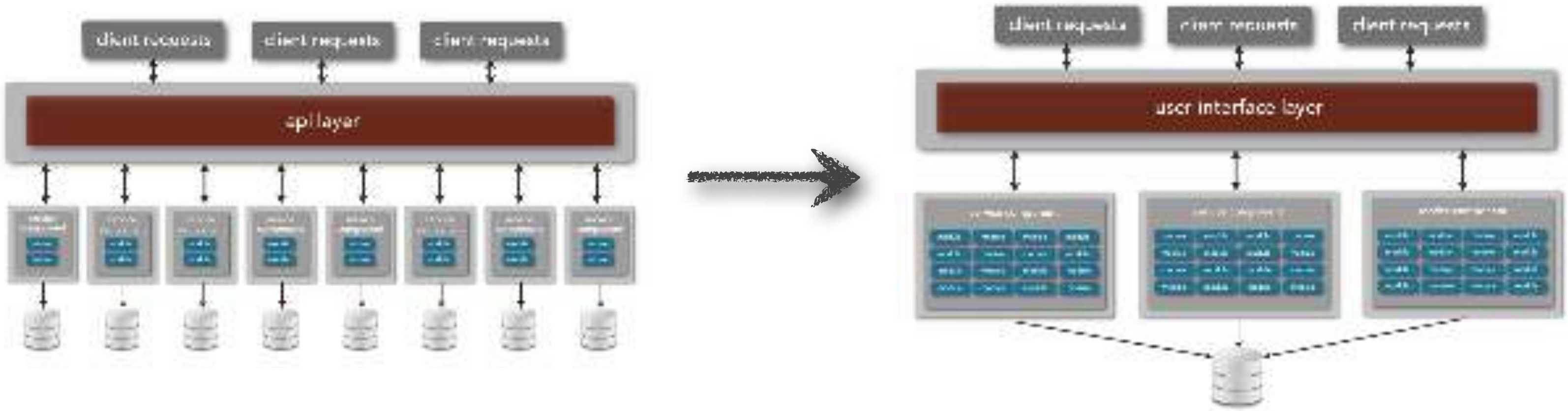
database scope

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

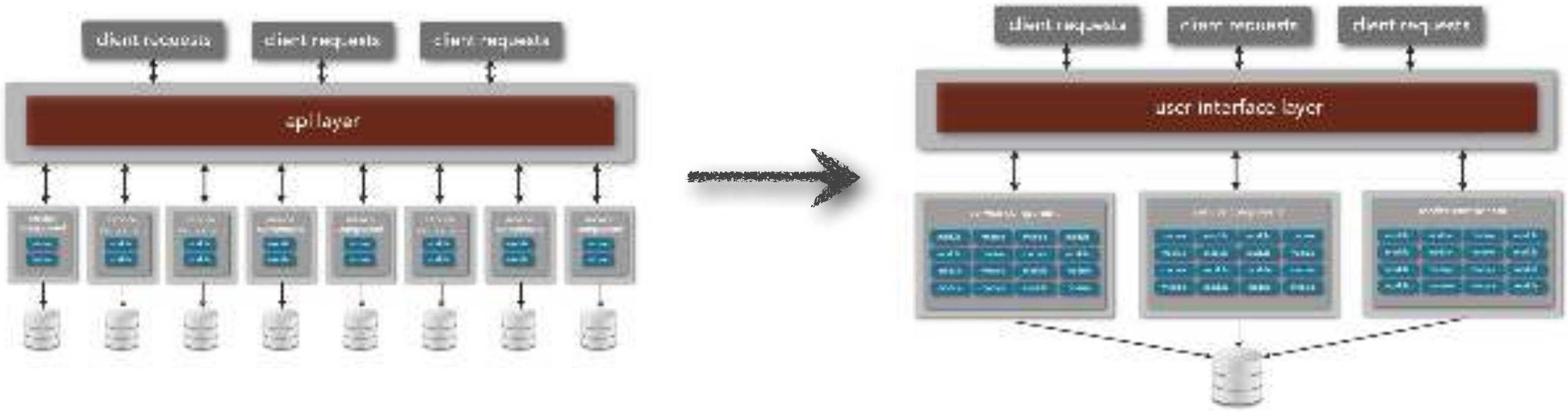


advantages

- performance (joins, orchestration, choreography)
- feasibility

service-based architecture

database scope

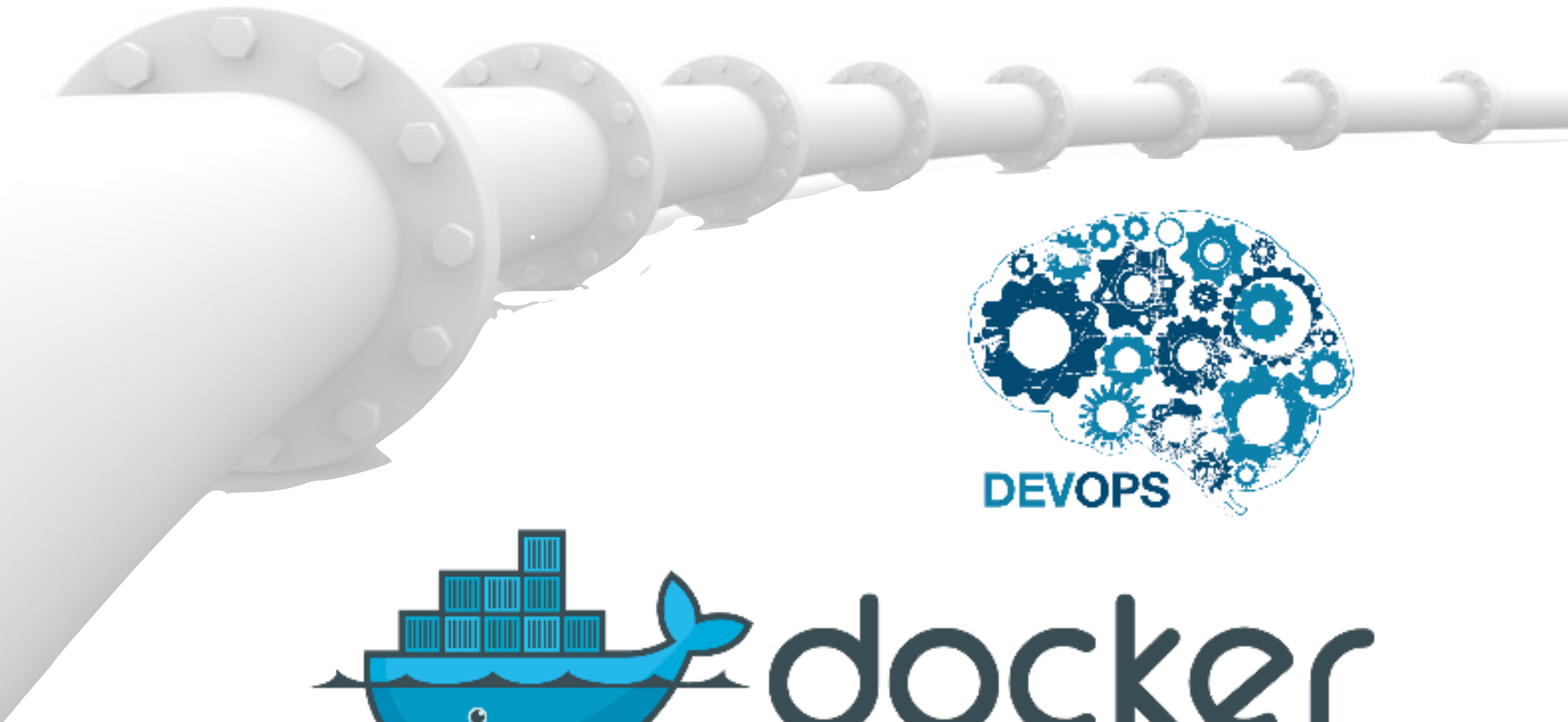


tradeoffs

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

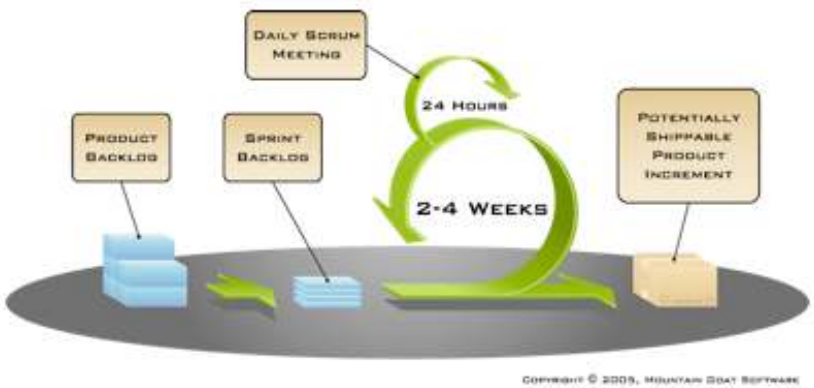
service-based architecture

deployment pipeline



service-based architecture

deployment pipeline



service-based architecture

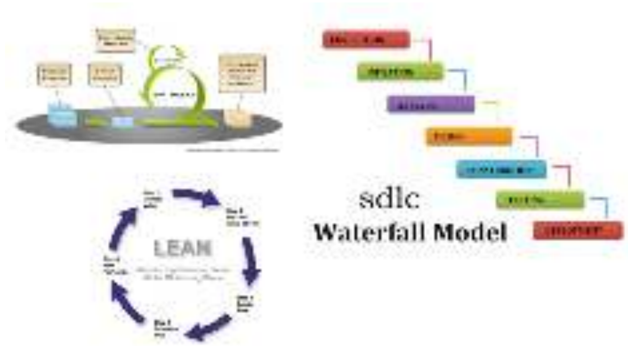
deployment pipeline

advantages

- no devops complexity
- minimal organizational change

service-based architecture

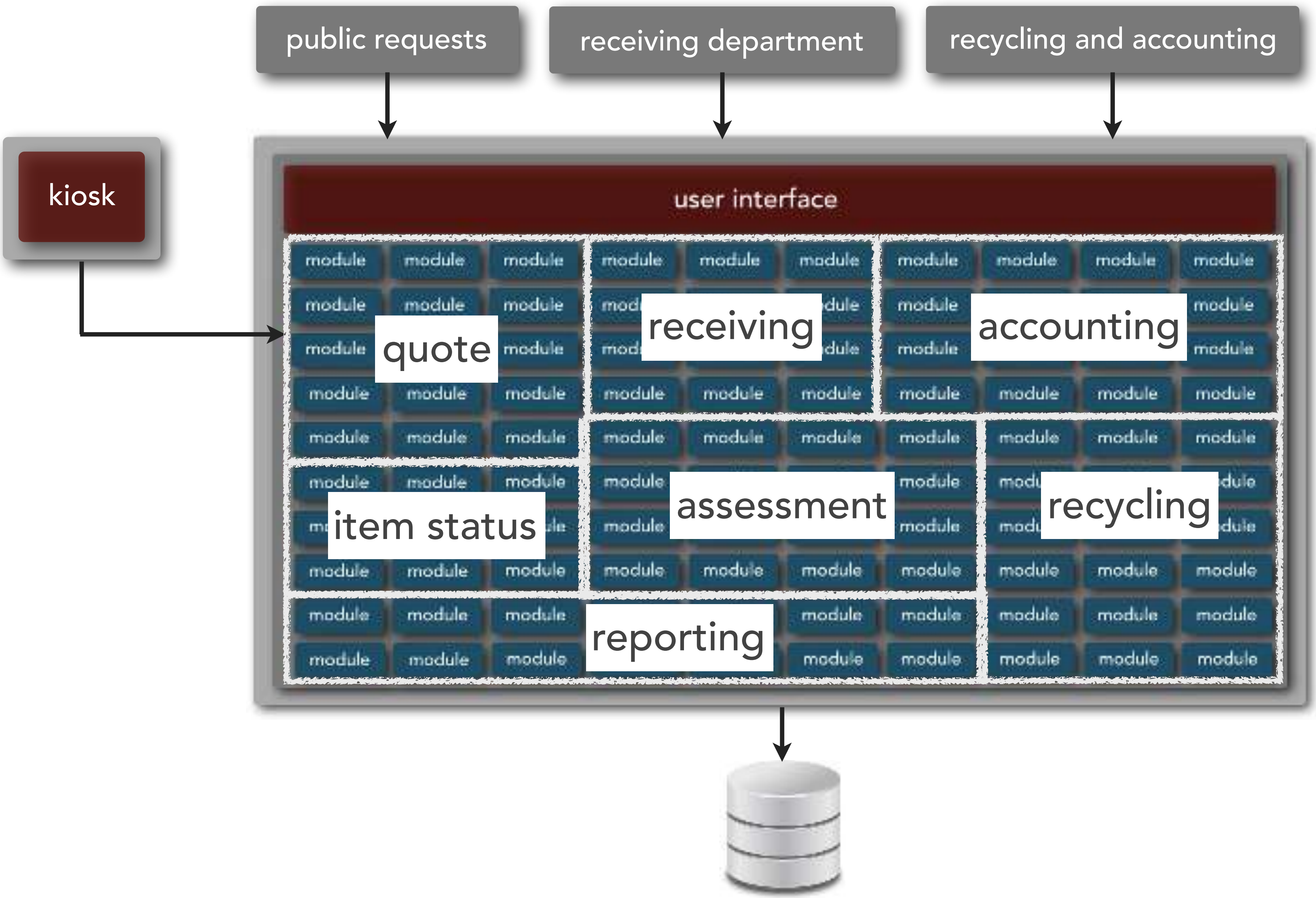
deployment pipeline



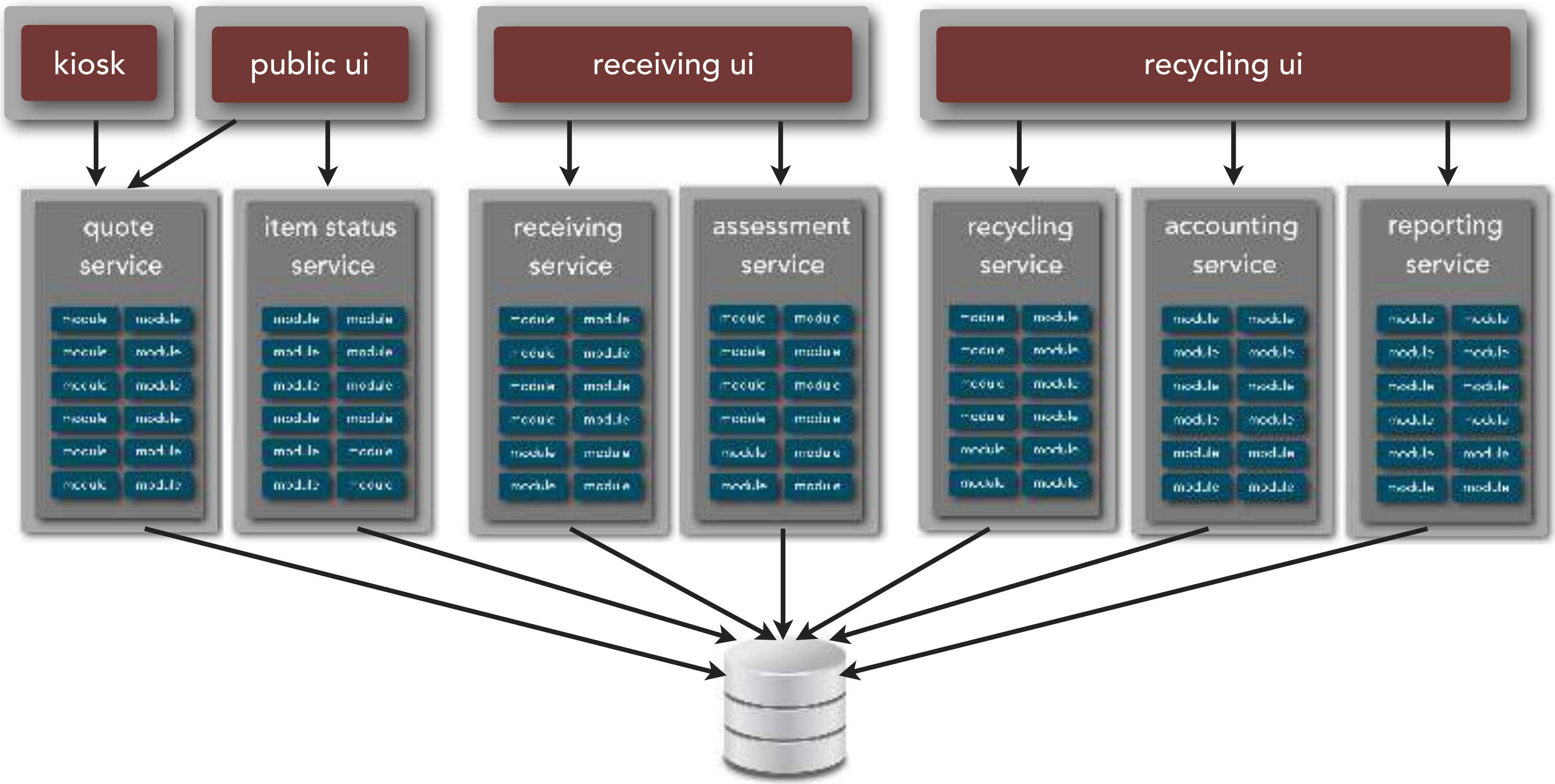
tradeoffs

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

electronics recycling application

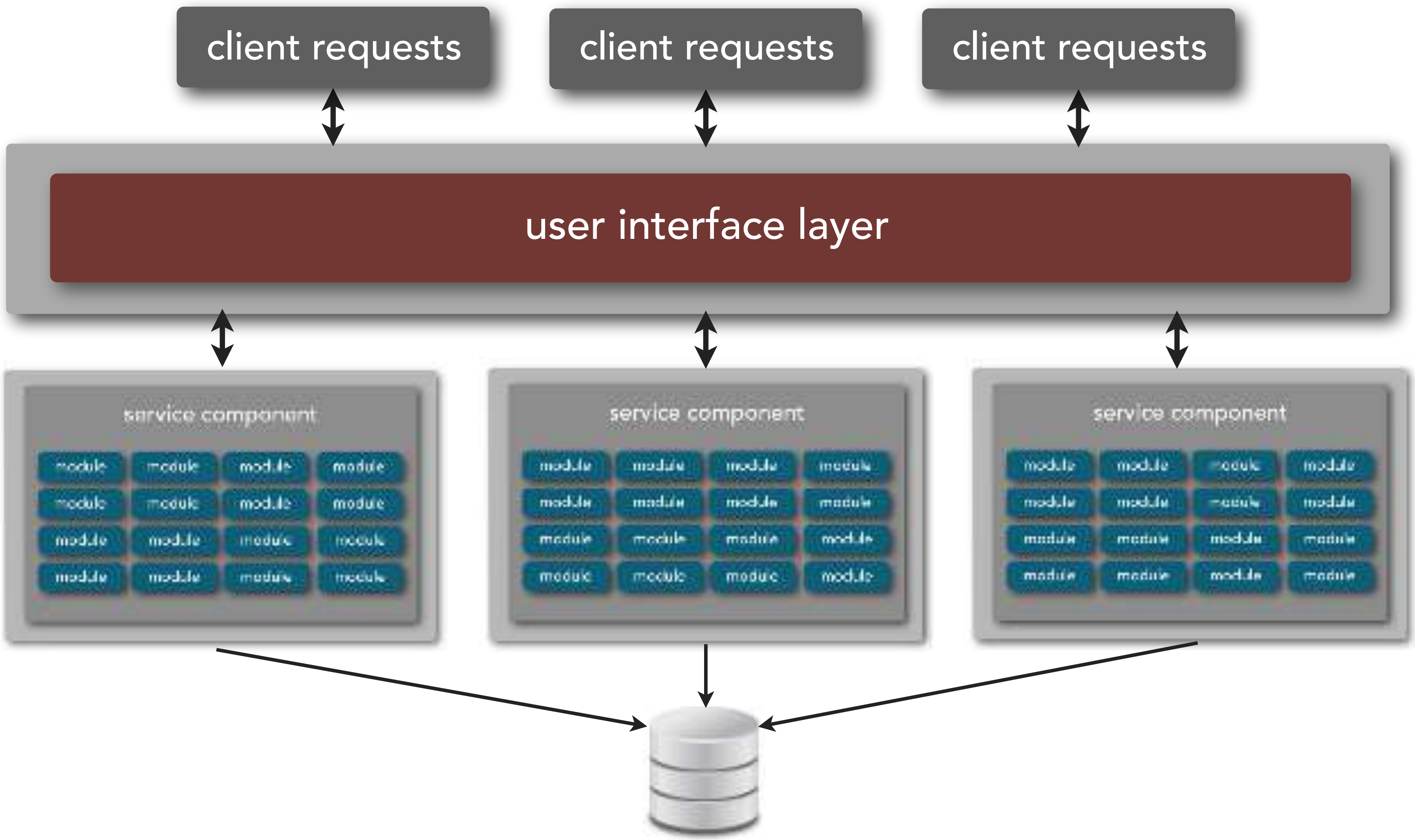


electronics recycling application



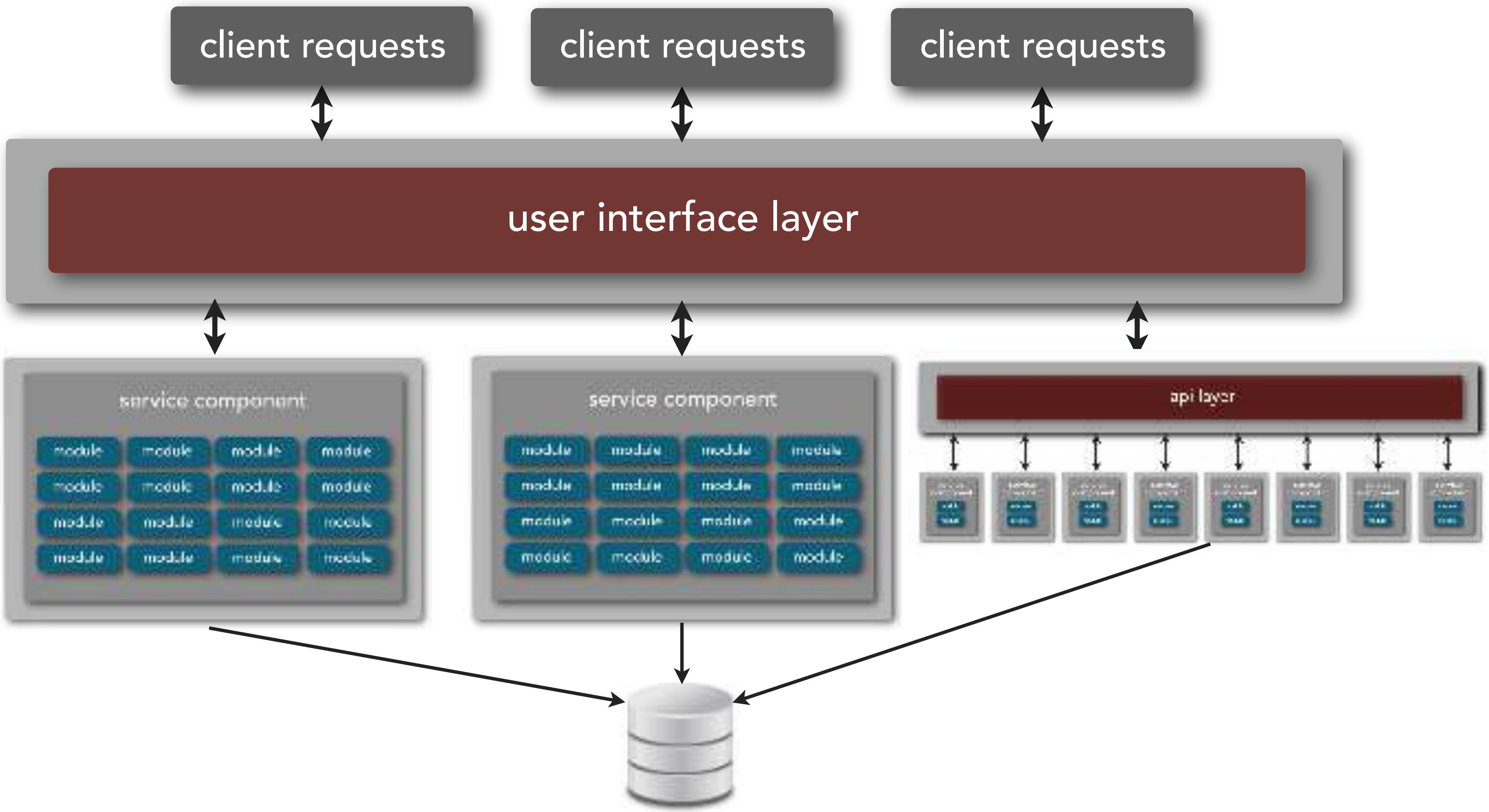
service-based architecture

adding microservices



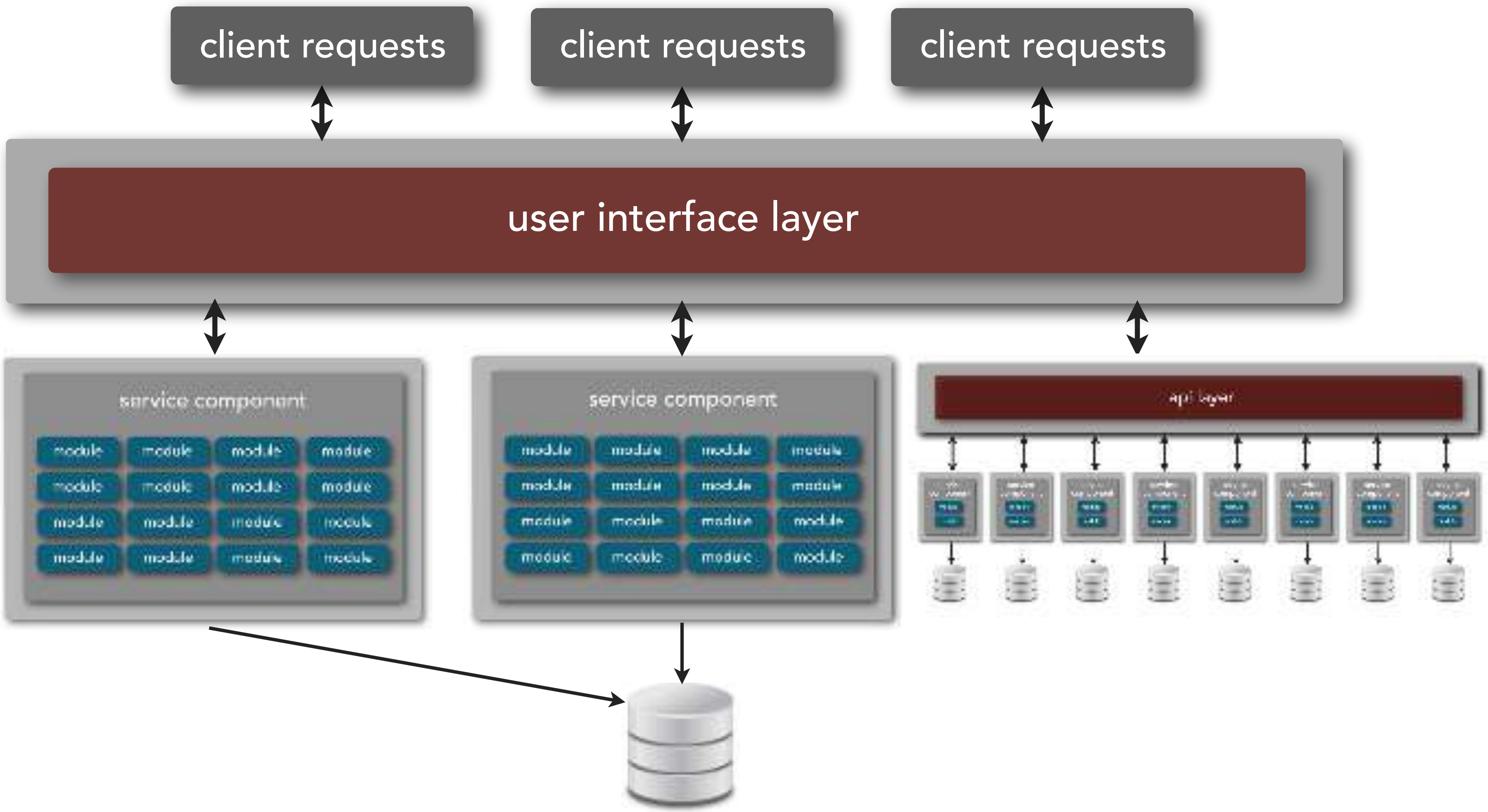
service-based architecture

adding microservices

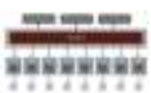
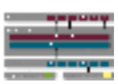


service-based architecture

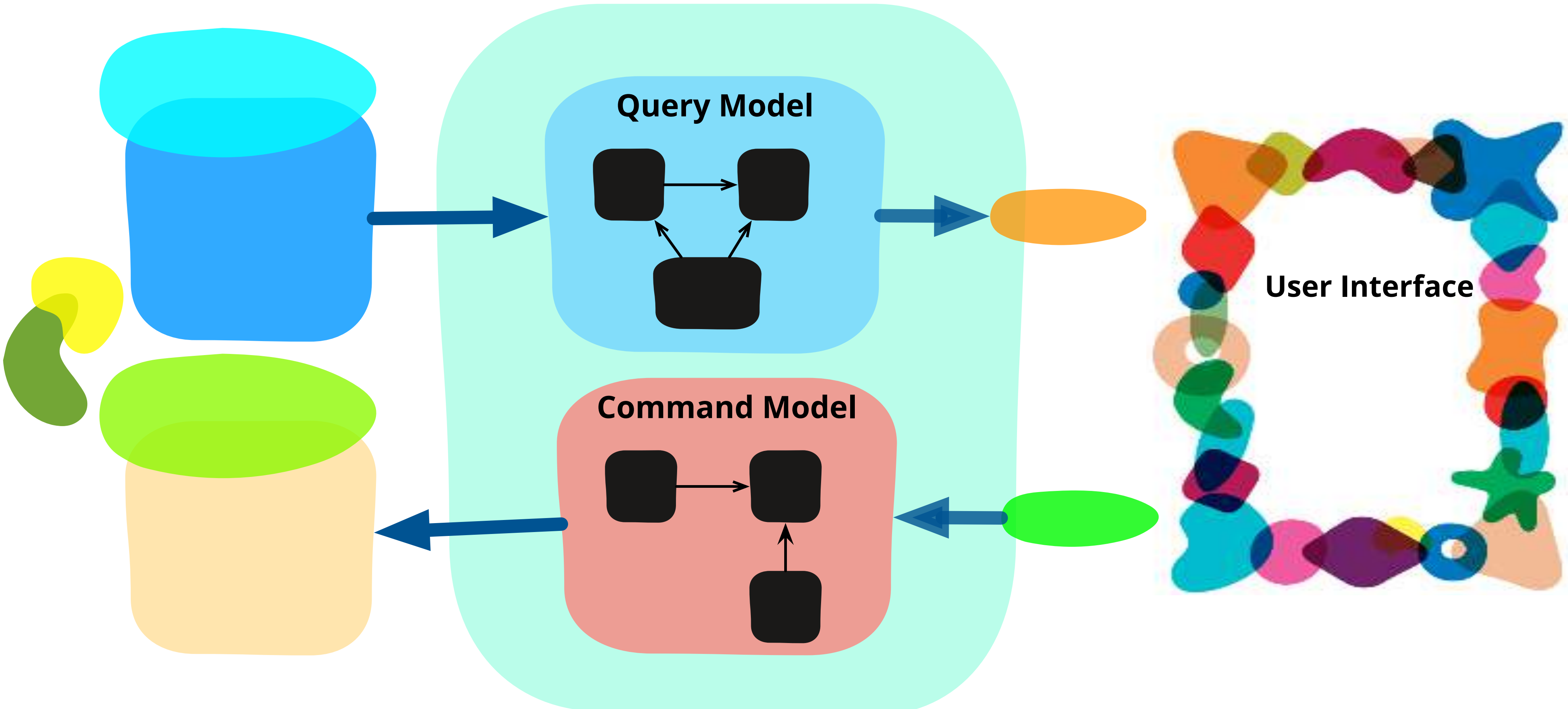
adding microservices



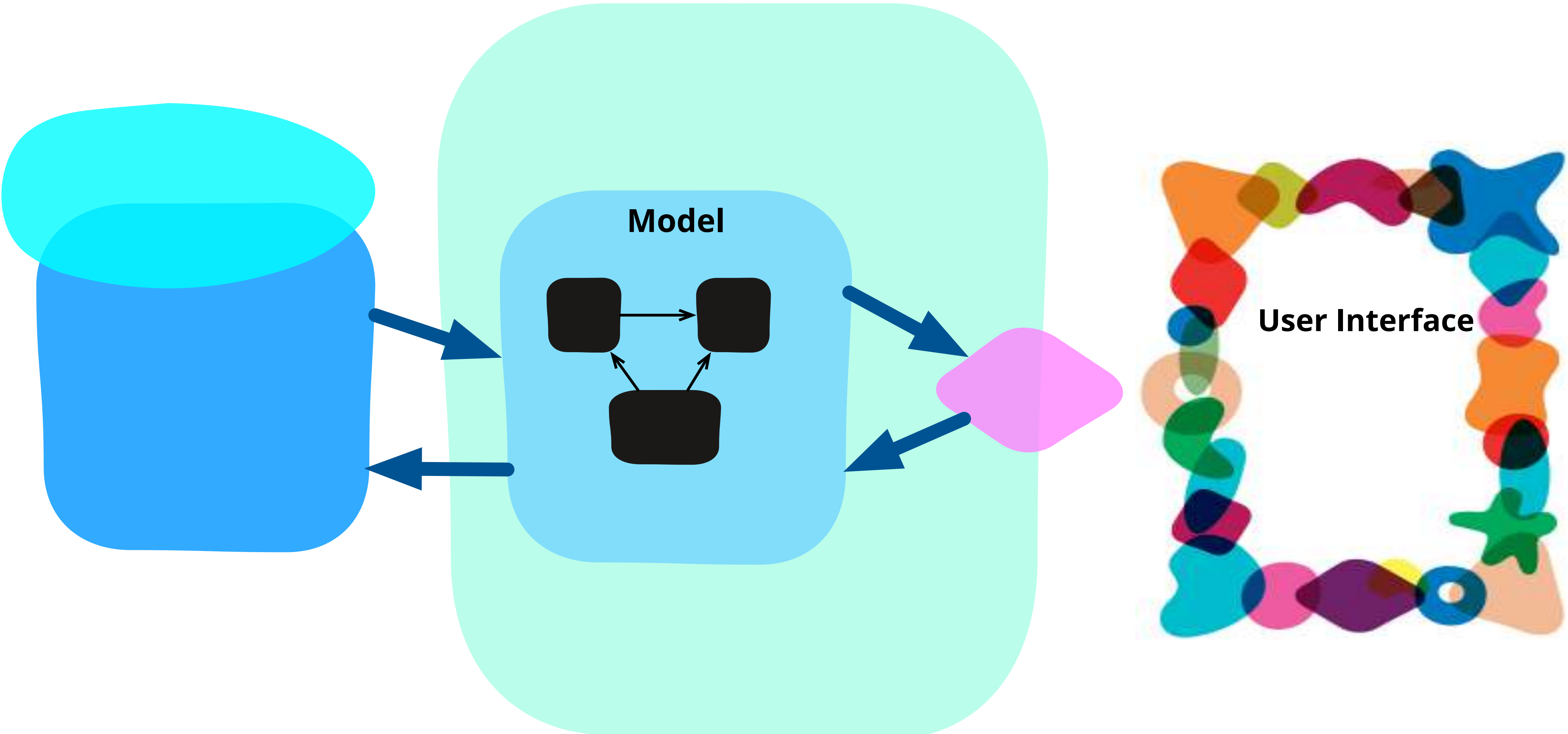
service-based architecture

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

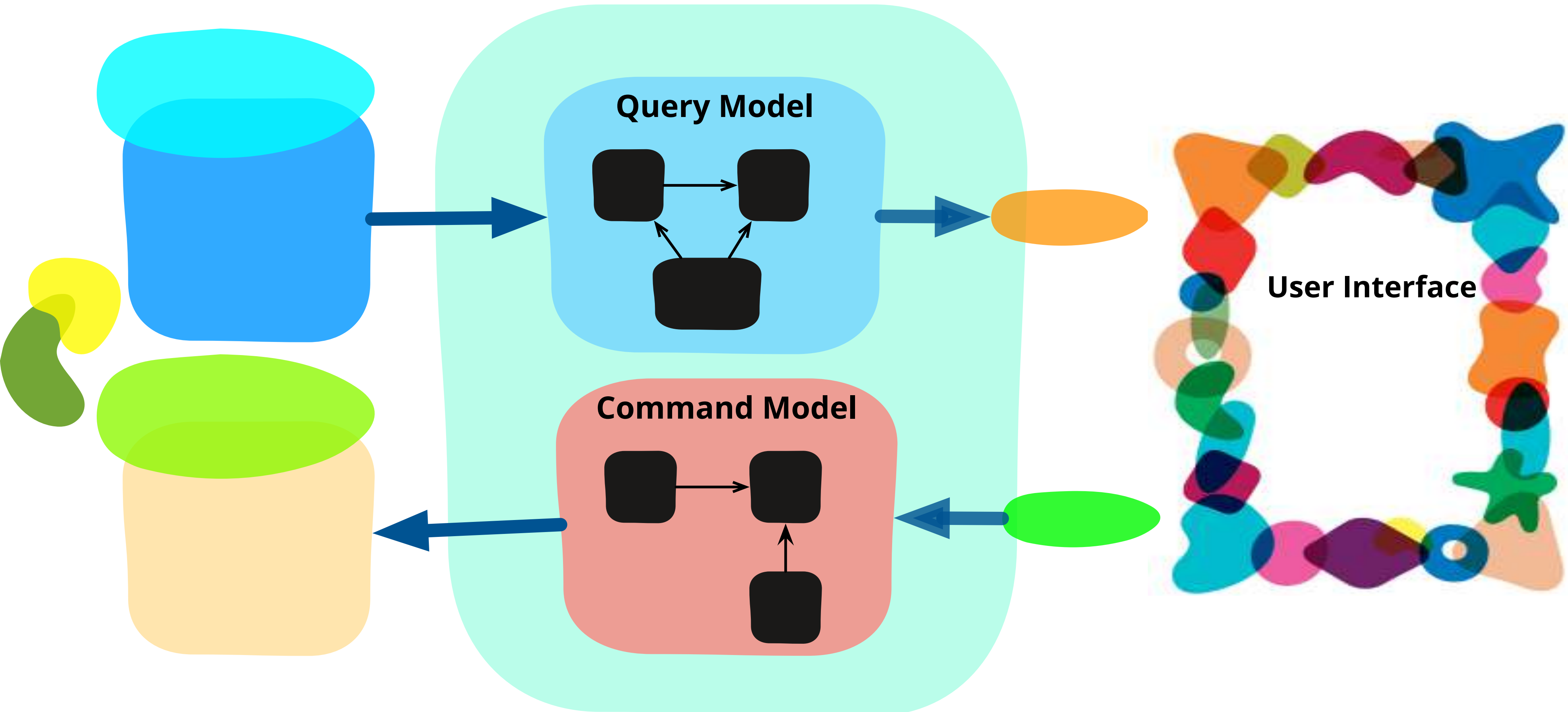
CQRS



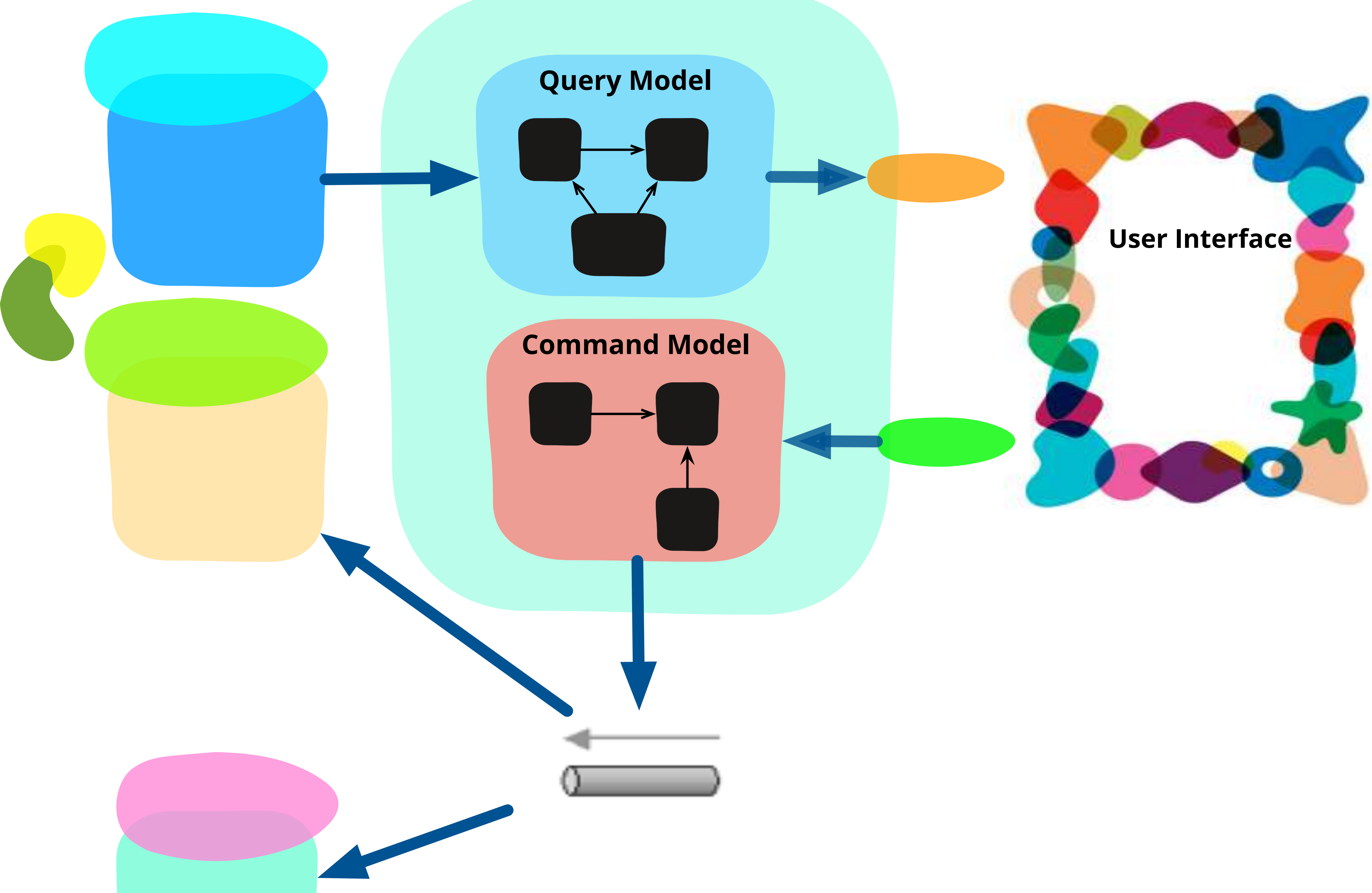
http://codebetter.com/gregyoung/2010/02/16/cqrs-task-based-uis-event-sourcing-agh/



traditional



CQRS Command Query Responsibility Separation



reporting in cqrs

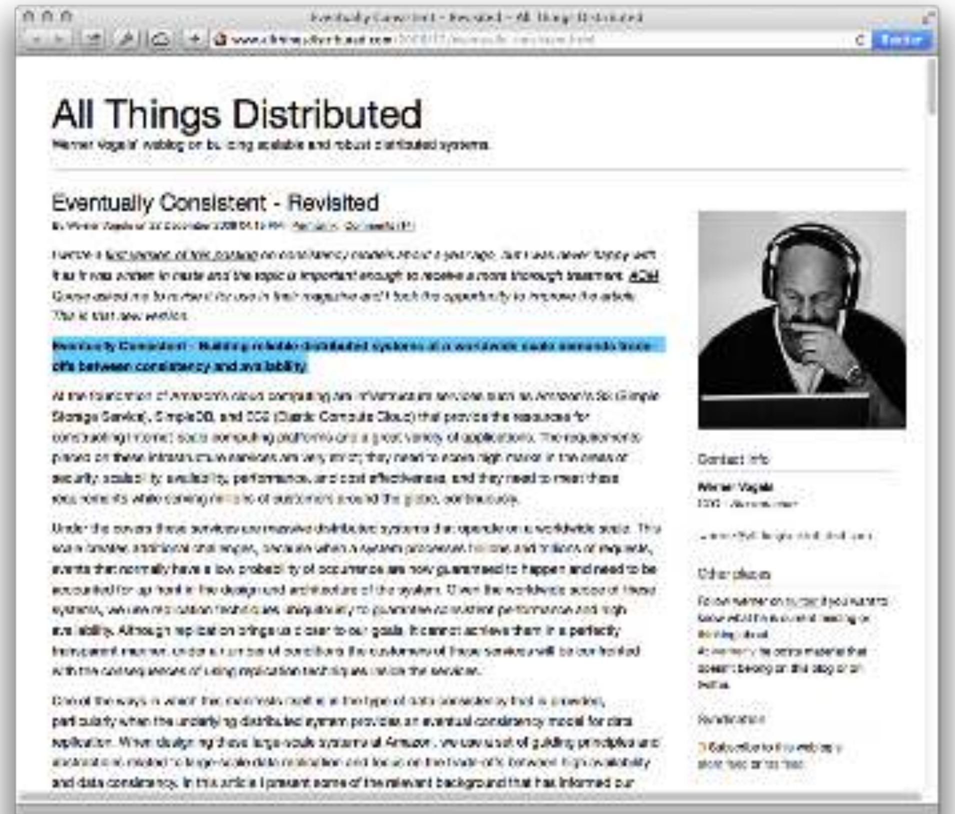
CQRS natural fits

task-based user interface

meshes well with event sourcing

eventual consistency

eventual consistency



"Building reliable distributed systems at a worldwide scale demands trade-offs between consistency and availability."

http://www.allthingsdistributed.com/2008/12/eventually_consistent.html

CQRS natural fits

task-based user interface

meshes well with event sourcing

eventual consistency

consistency or availability
(but never both)

complex or granular domains

LMAX

LMAX

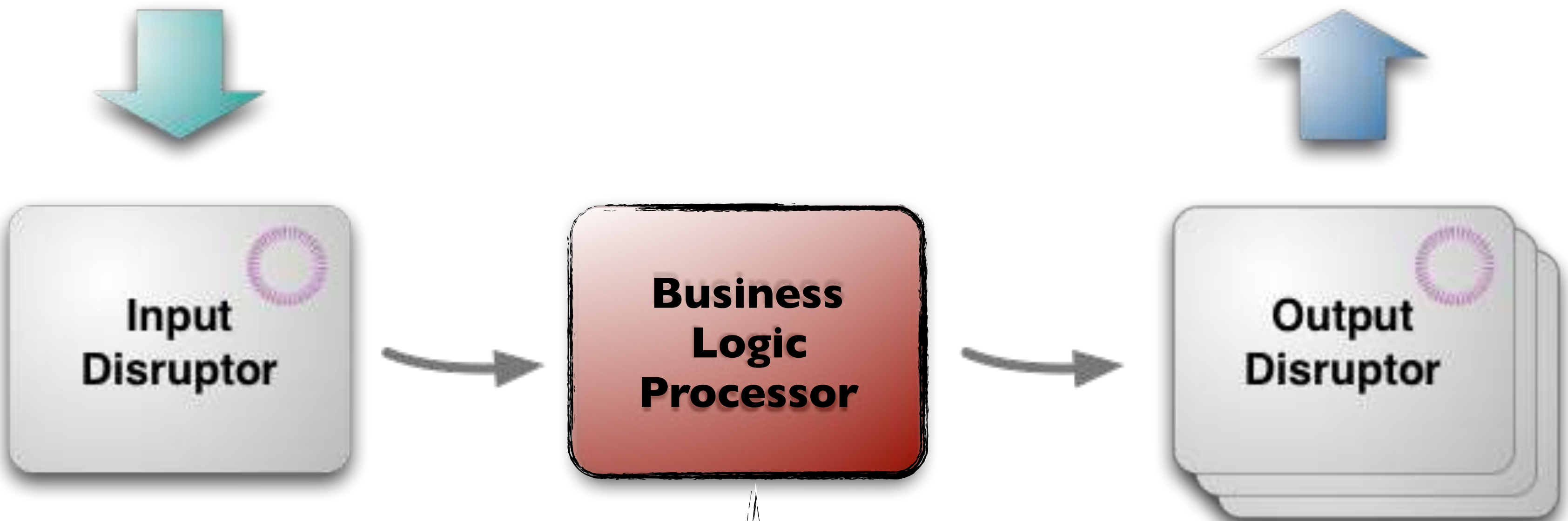
http://martinfowler.com/articles/lmax.html

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



— single-threaded Java app
— relies only on JVM
— easy to test

Business Logic Processor

business logic processor

in-memory

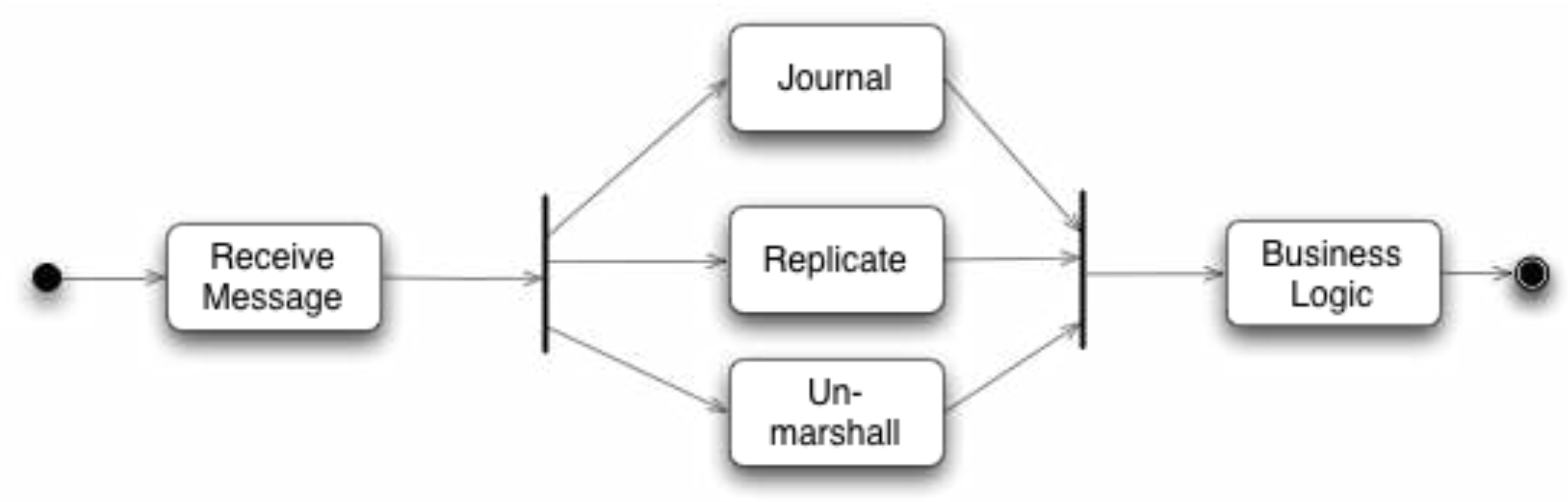
event sourcing via input disruptor

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



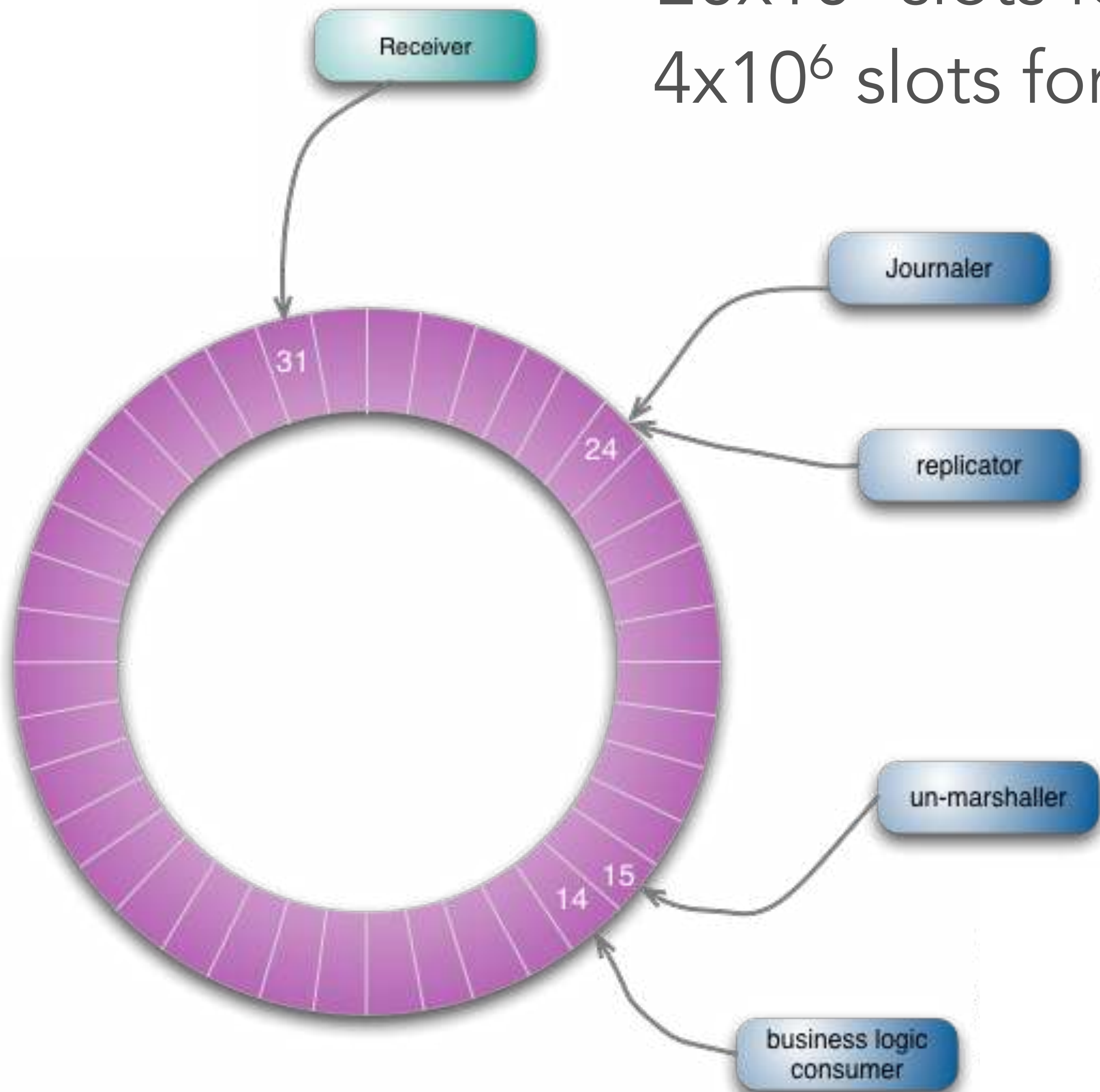
disruptors

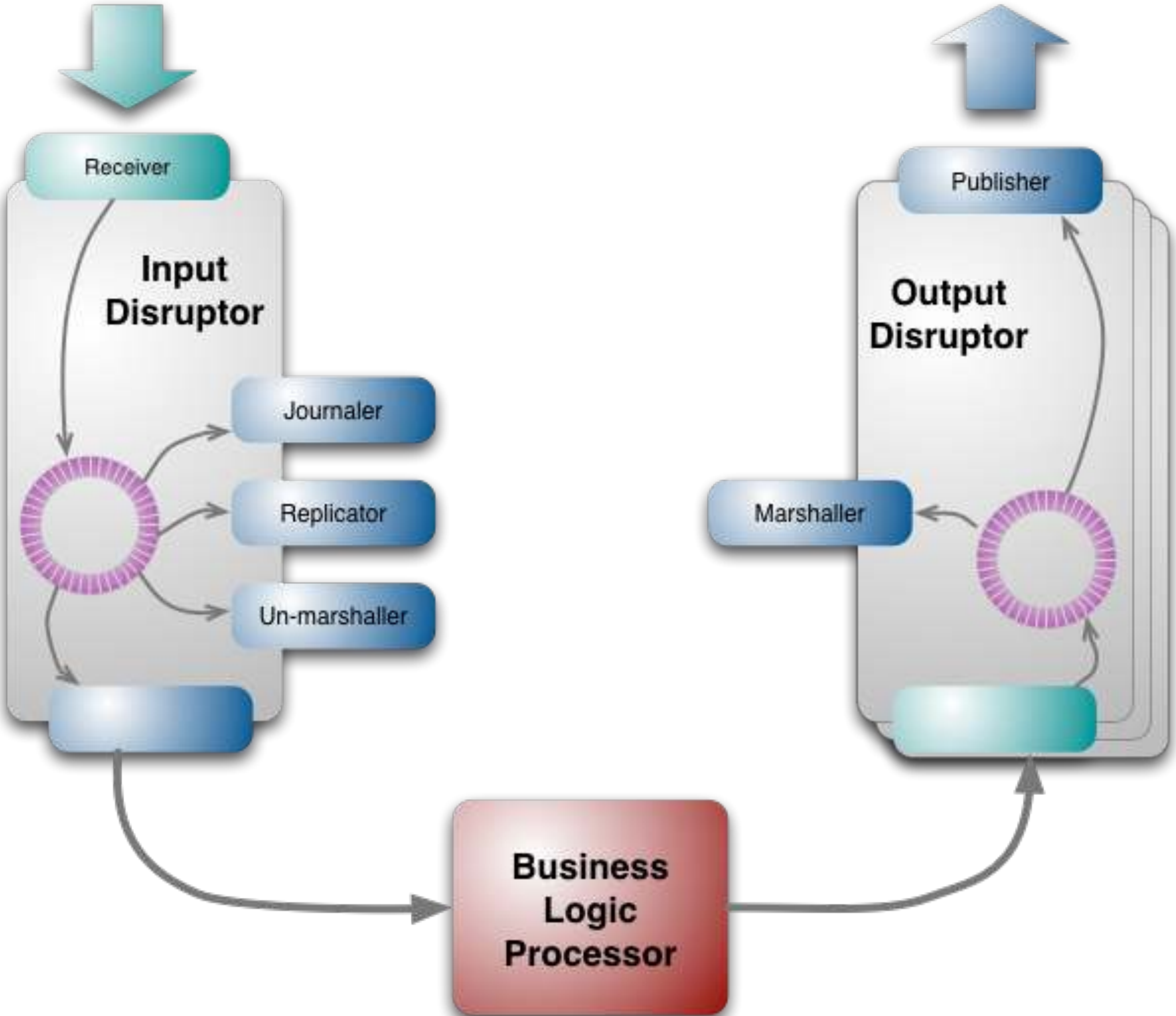
custom concurrency component

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
Output Disruptor
Publisher
Marshaller
Business Logic Processor

“mechanical sympathy”

started with transactions

switched to Actor-based concurrency

hypothesized & measured results

CPU caching is key ➠ single writer principle

architecture katas

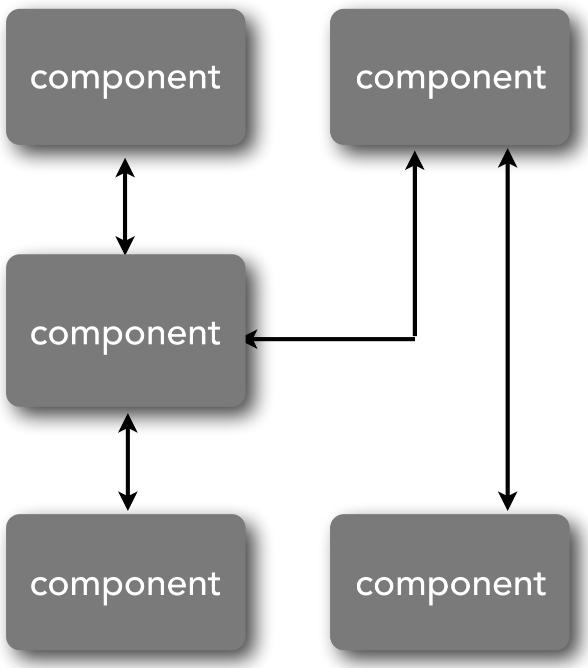
identifying architecture patterns

component-based thinking

component identification and granularity

component identification

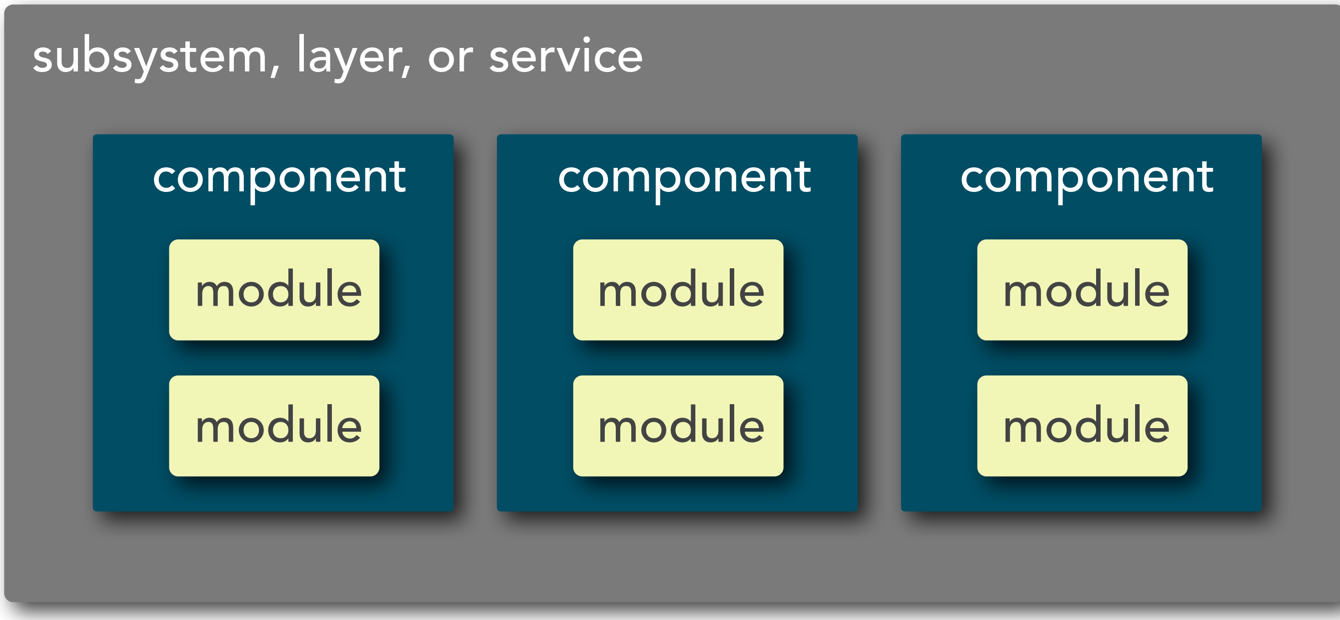
as an architect, you should think about the artifacts within the architecture in terms of *components*



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

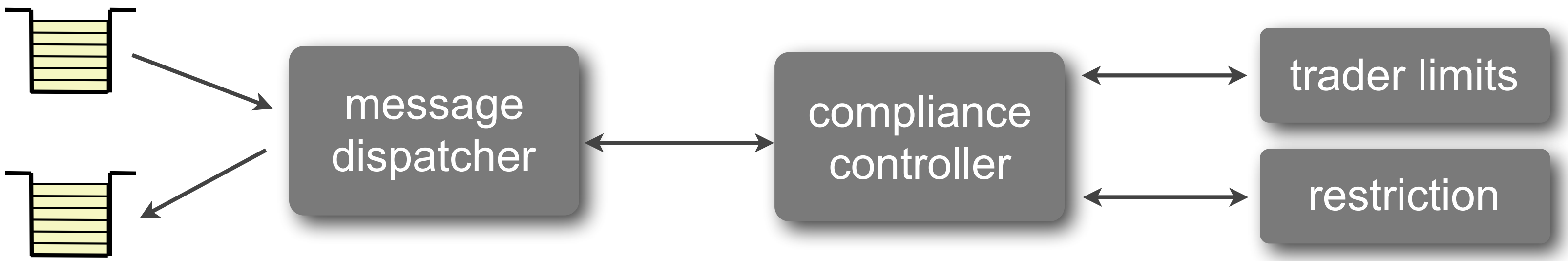


component identification

roles and responsibility model

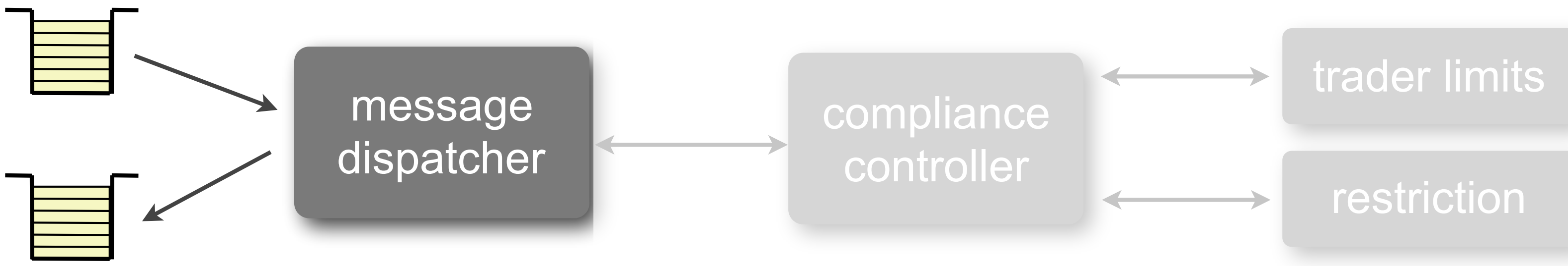
component identification

stock trade order validation



component identification

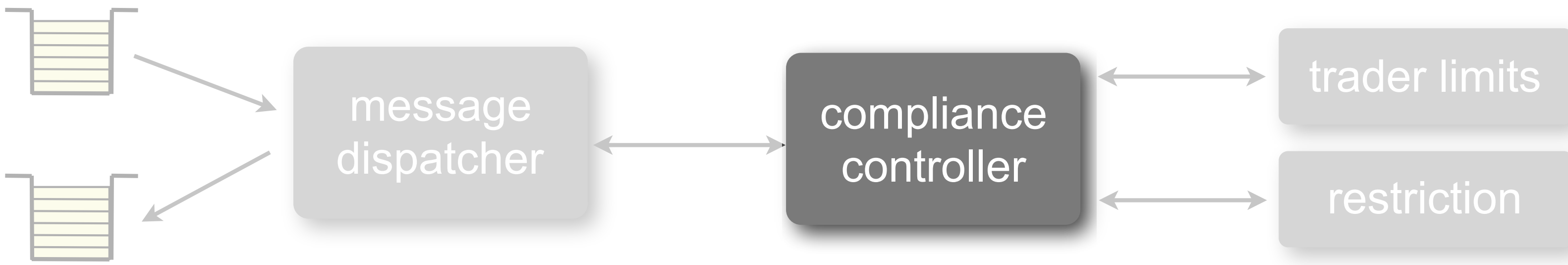
stock trade order validation



responsible for dispatching the trade to the next available controller.

component identification

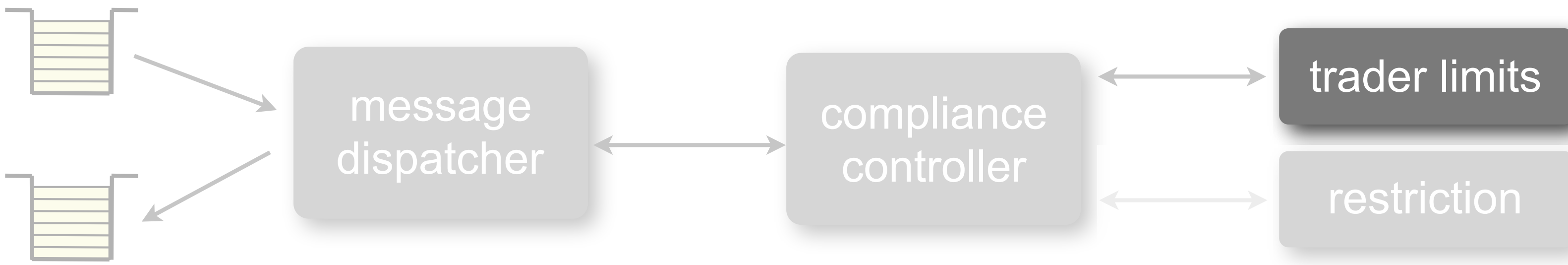
stock trade order validation



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

component identification

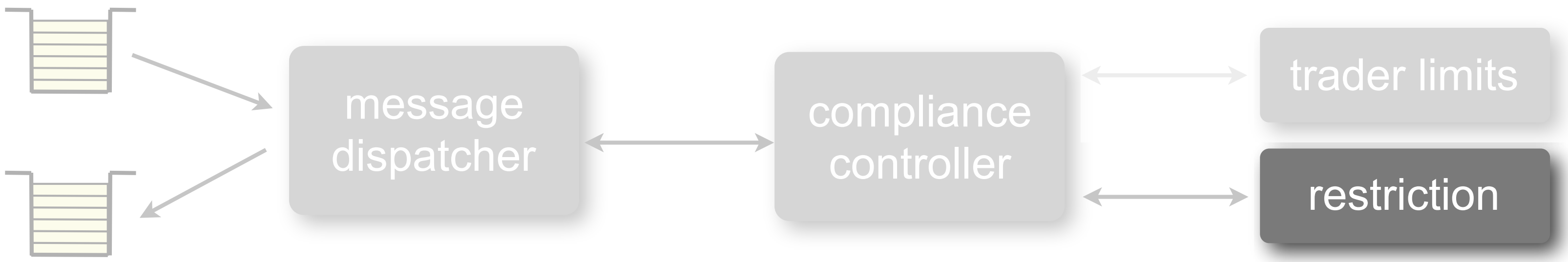
stock trade order validation



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

component identification

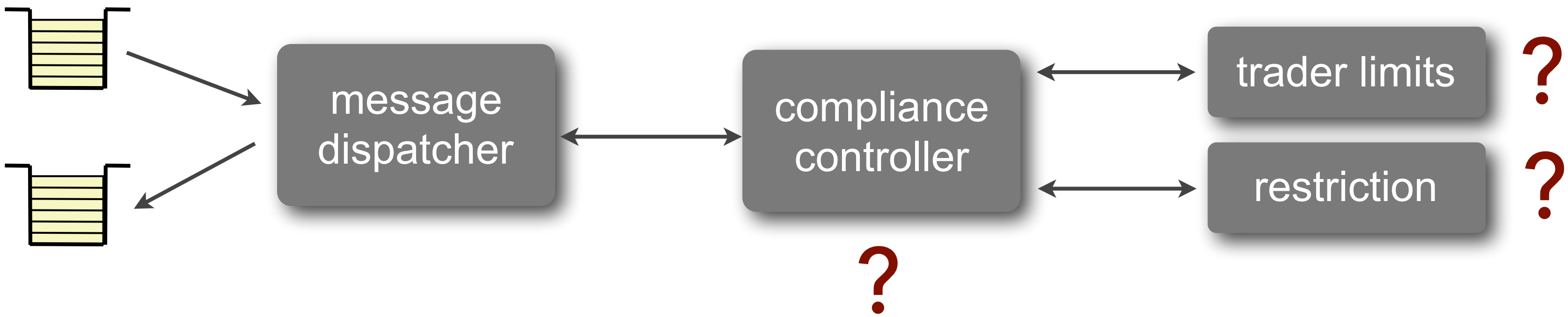
stock trade order validation



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

component identification

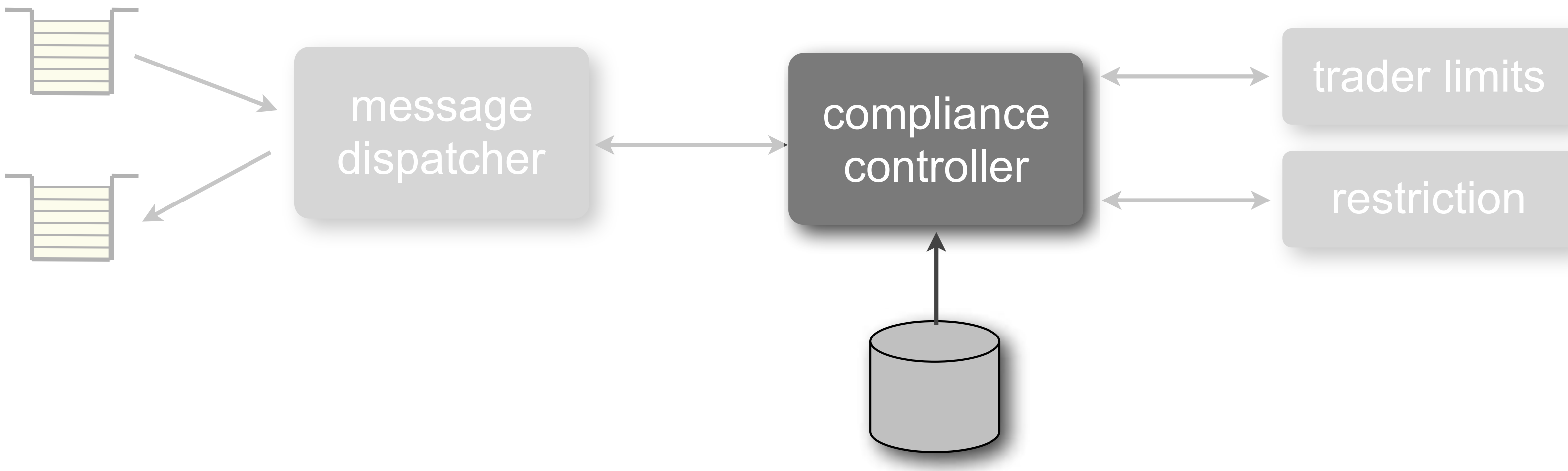
stock trade order validation



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

component identification

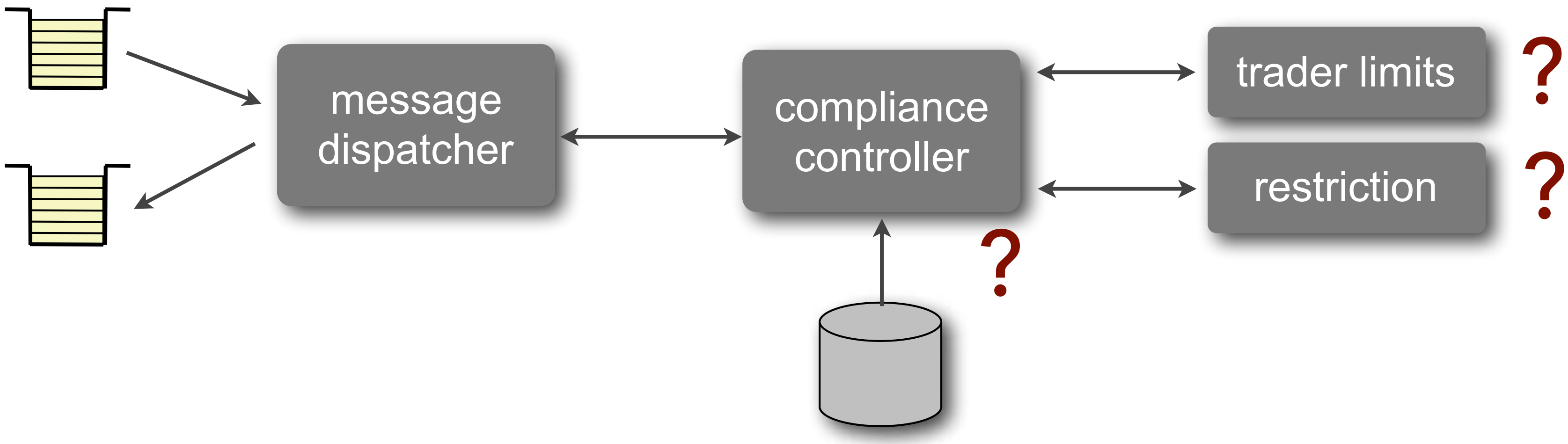
stock trade order validation



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

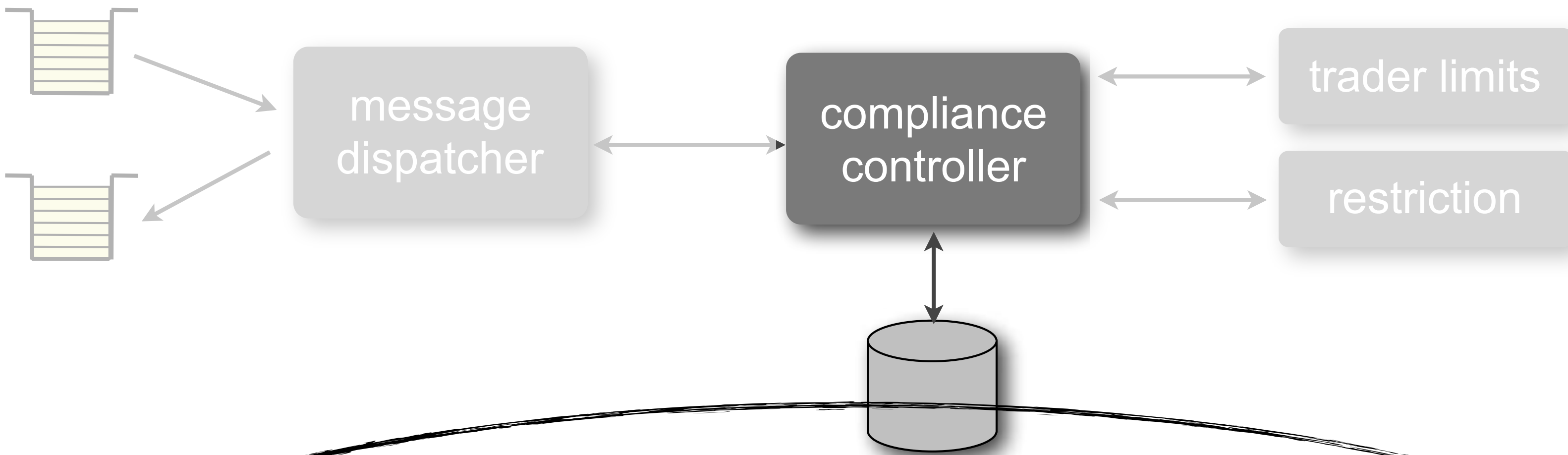
stock trade order validation



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

component identification

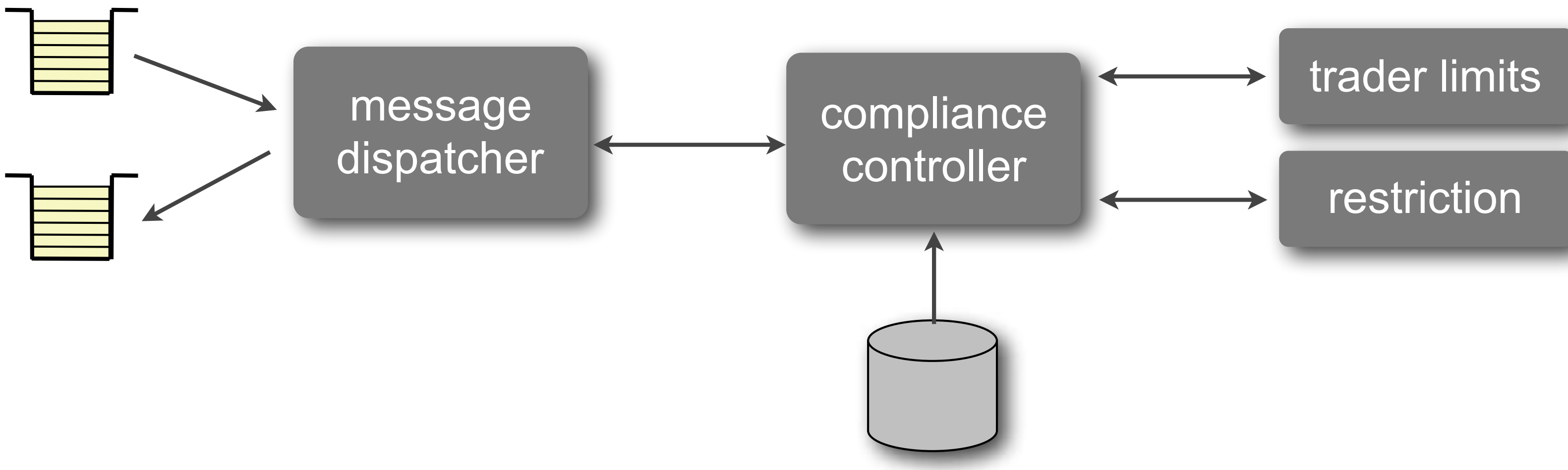
stock trade order validation



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.

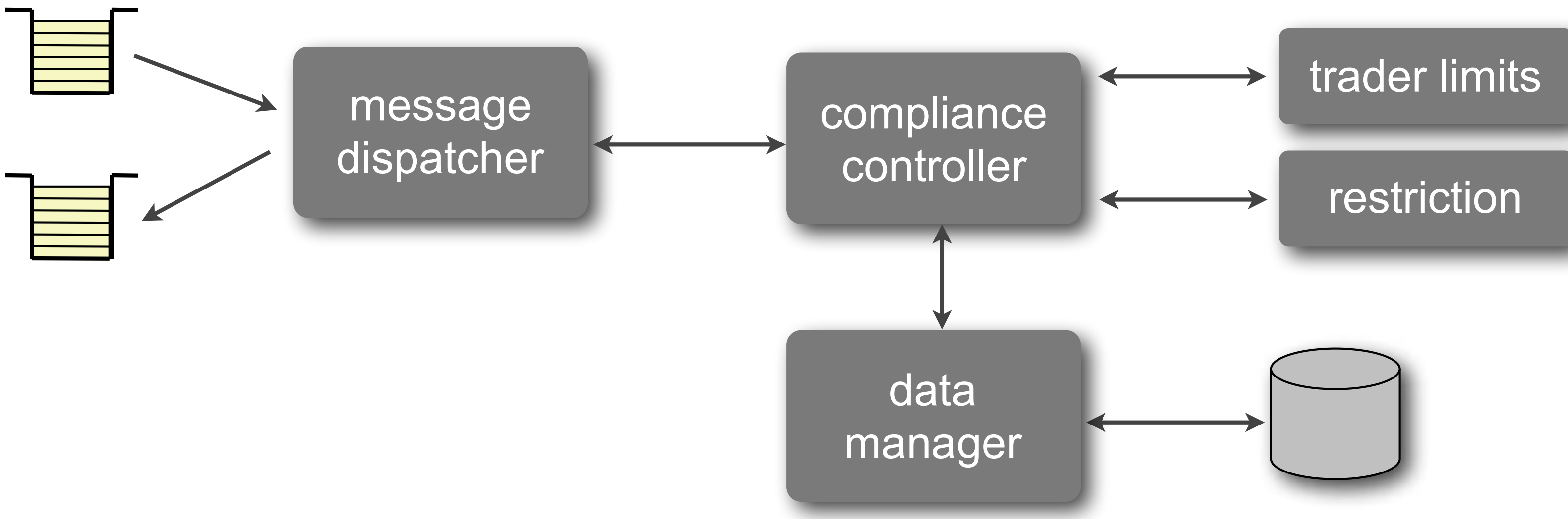
component identification

stock trade order validation



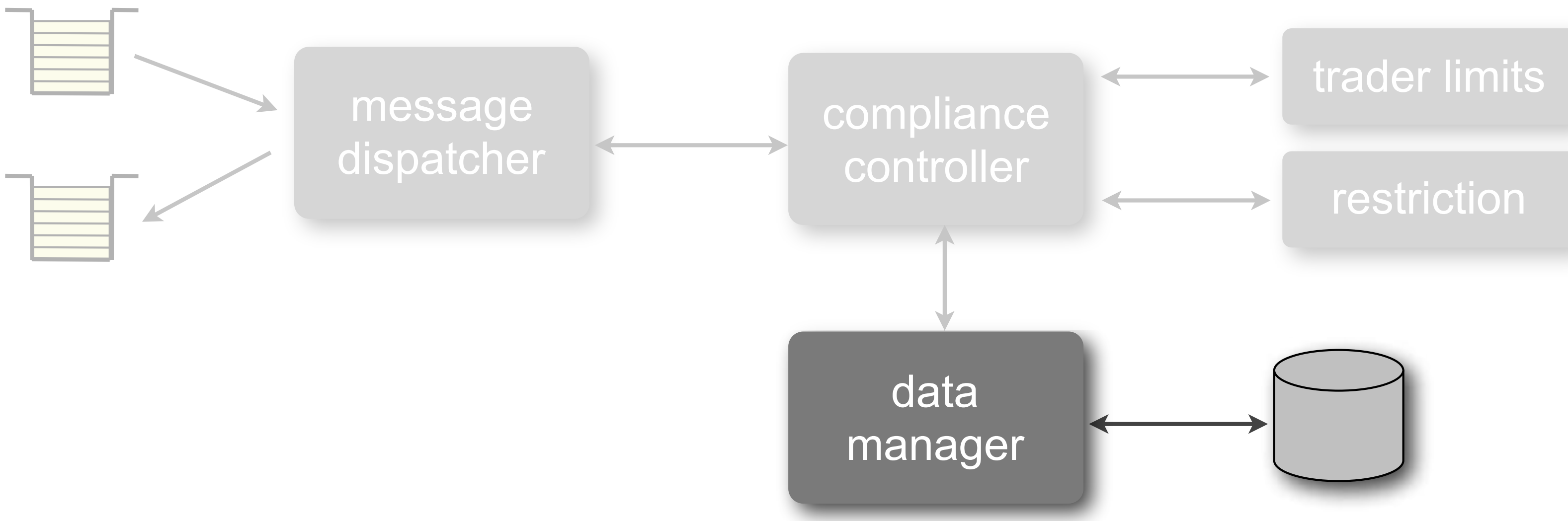
component identification

stock trade order validation



component identification

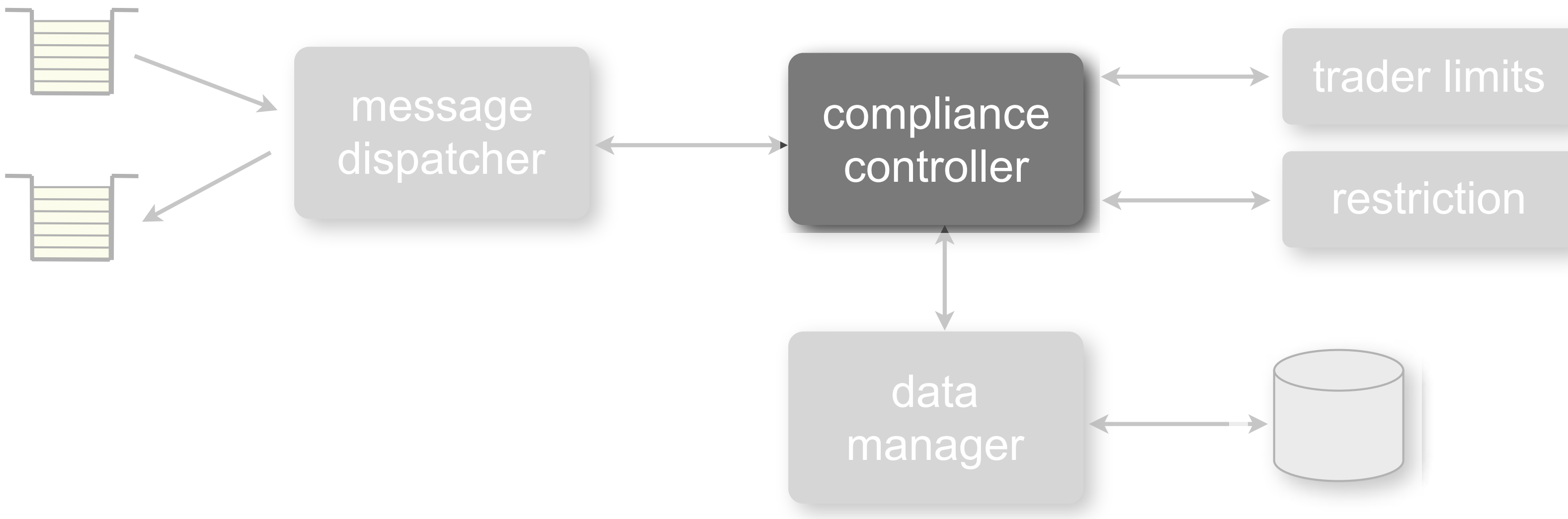
stock trade order validation



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

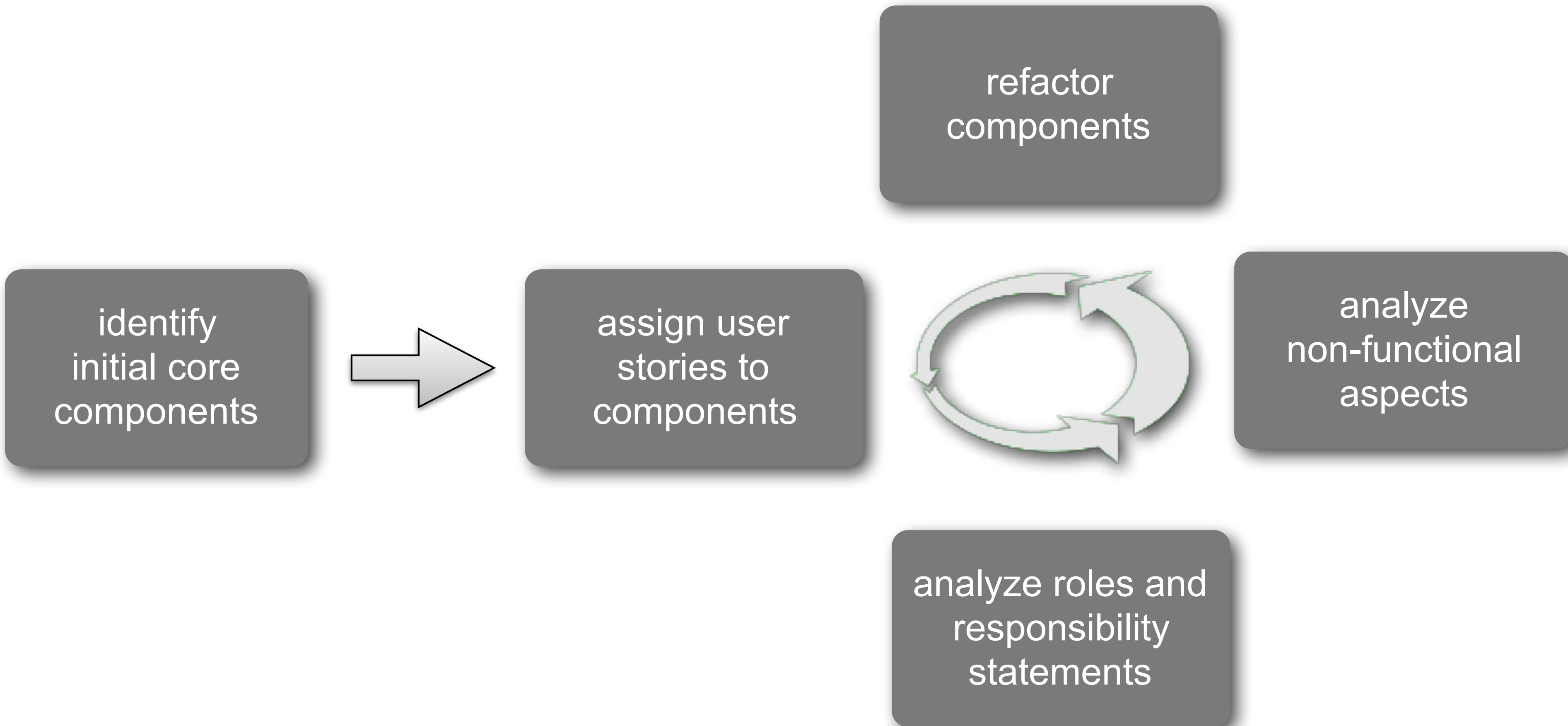
component identification

stock trade order validation



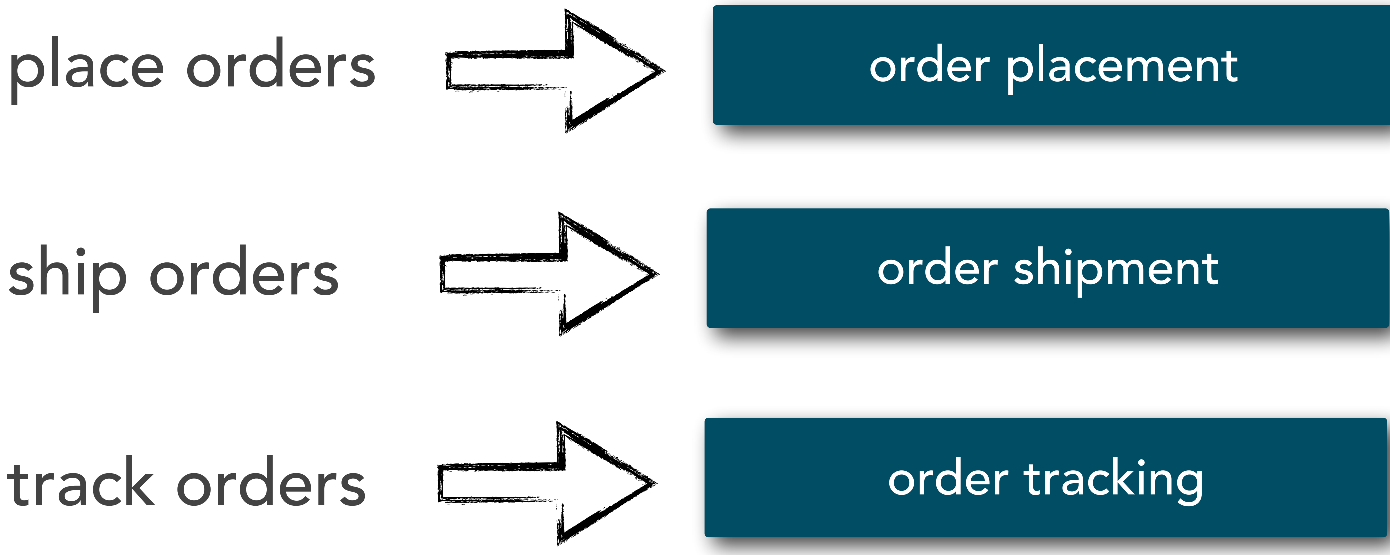
responsible for orchestrating the trade order validation process by calling specific compliance processors.

component identification



component identification

identify initial components using core functionality



component identification

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

user story: check inventory → order placement

user story: validate order → order placement

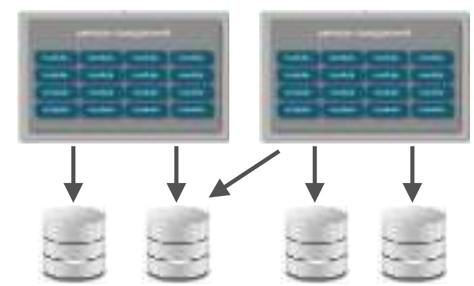
user story: notify customer → customer notification

service identification

identify coarse-grained functional areas

create coarse-grained microservices

map services to data tables

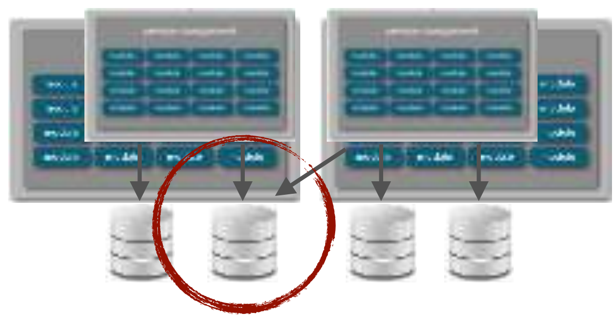


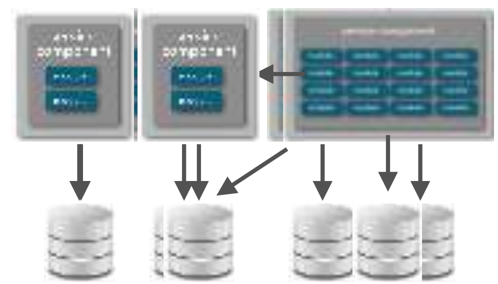
service identification

identify coarse-grained functional areas

identify data overlaps and dependencies

refine services and data





component identification

component granularity

order manager

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

order maintenance

order manager

order shipment

responsible for creating, deleting, and updating orders.

responsible for shipping and tracking orders

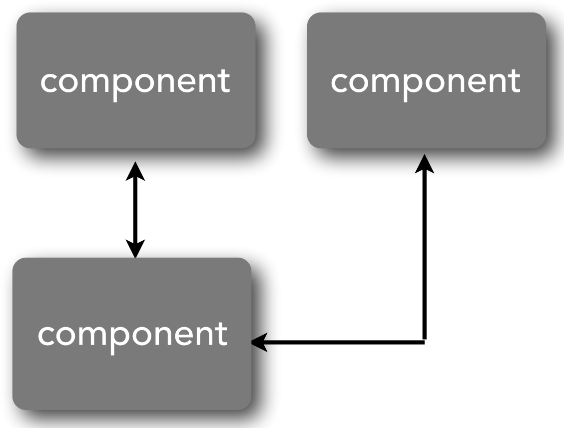
customer notification

responsible for notifying the customer when the order status changes.

component coupling

component coupling

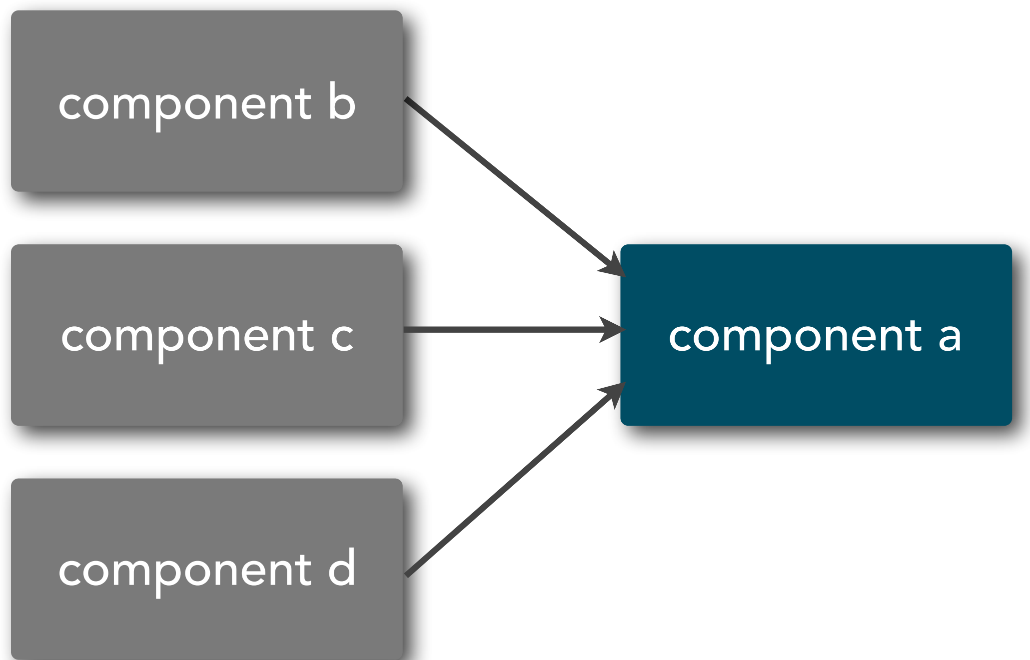
the extent to which components know about each other



component coupling

afferent coupling

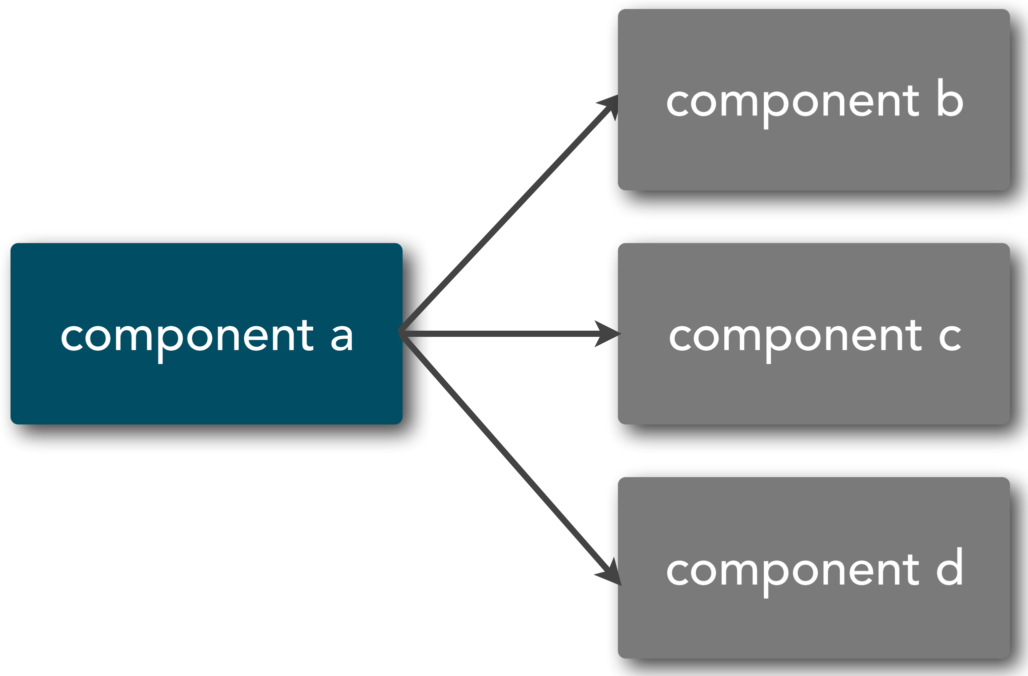
the degree to which other components are dependent on the target component



component coupling

efferent coupling

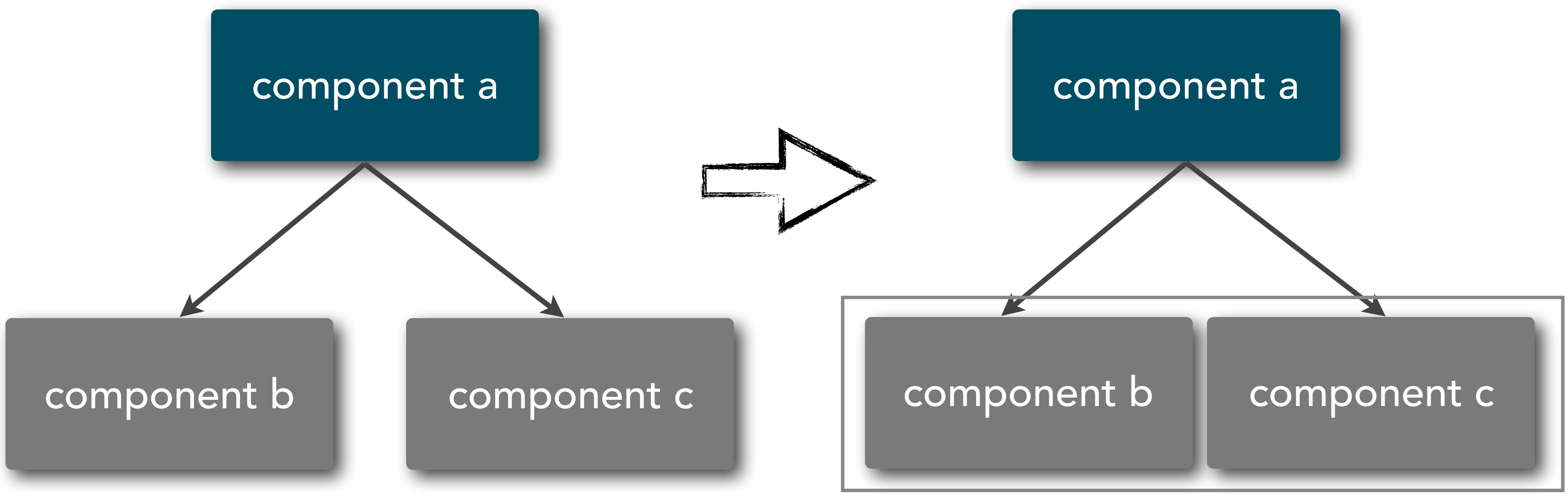
the degree to which the target component
is dependent on other components



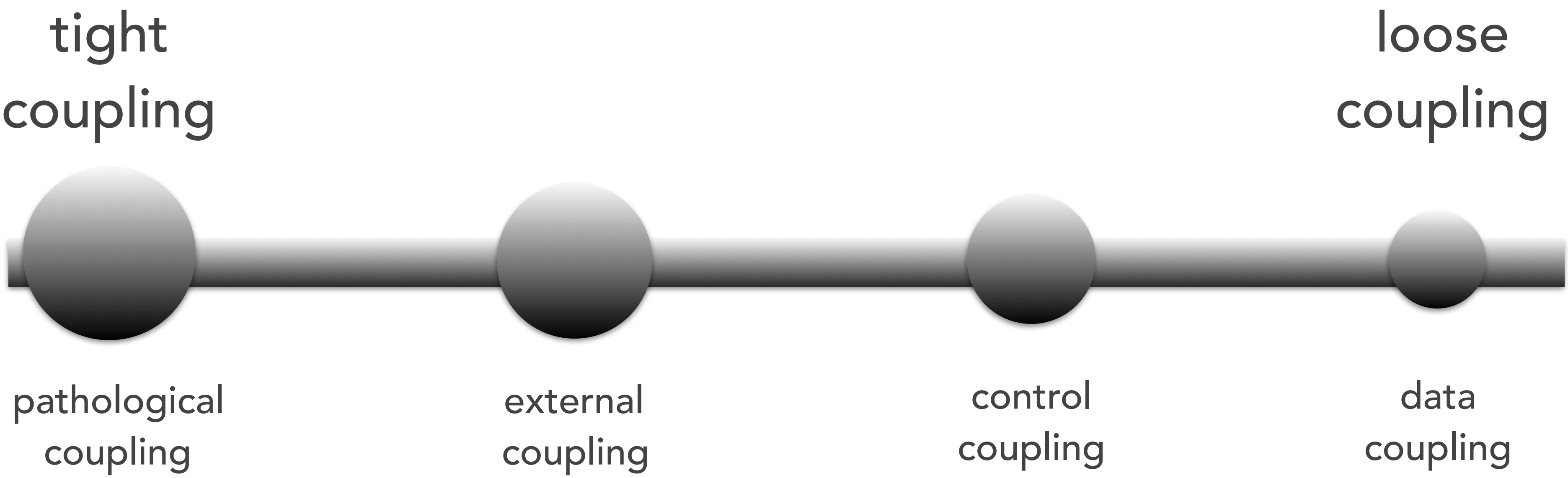
component coupling

temporal coupling

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



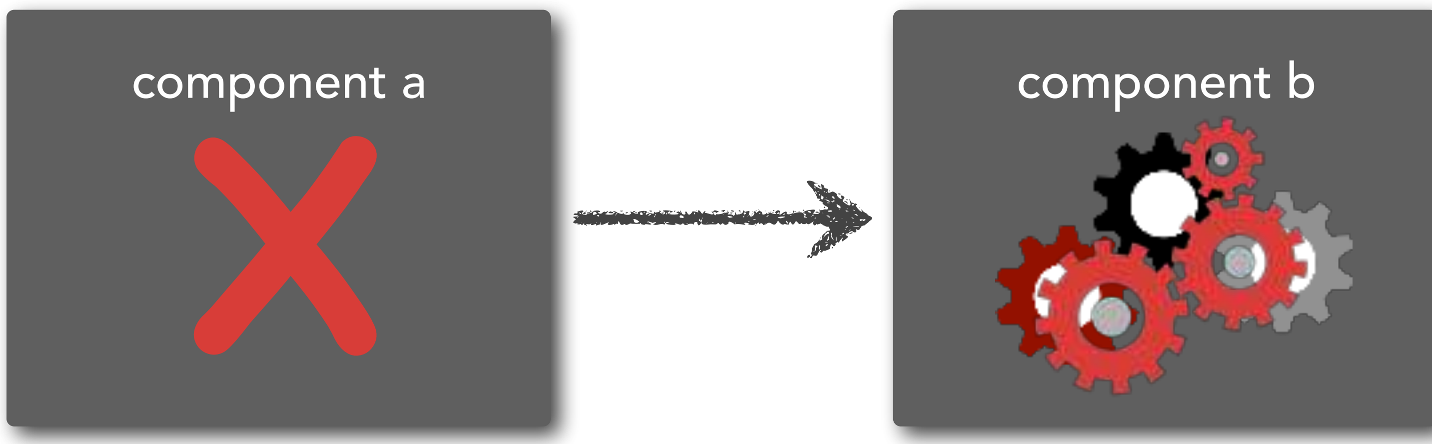
component coupling



component coupling

pathological coupling

one component relies on the inner workings of another component



component coupling

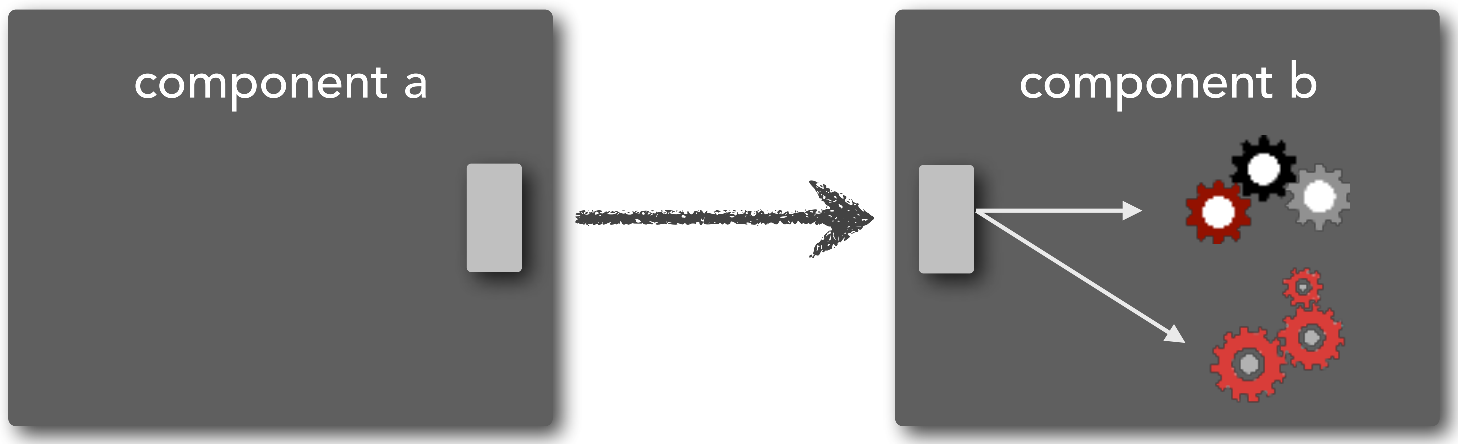
external coupling

multiple components share an externally imposed protocol or data format

component coupling

control coupling

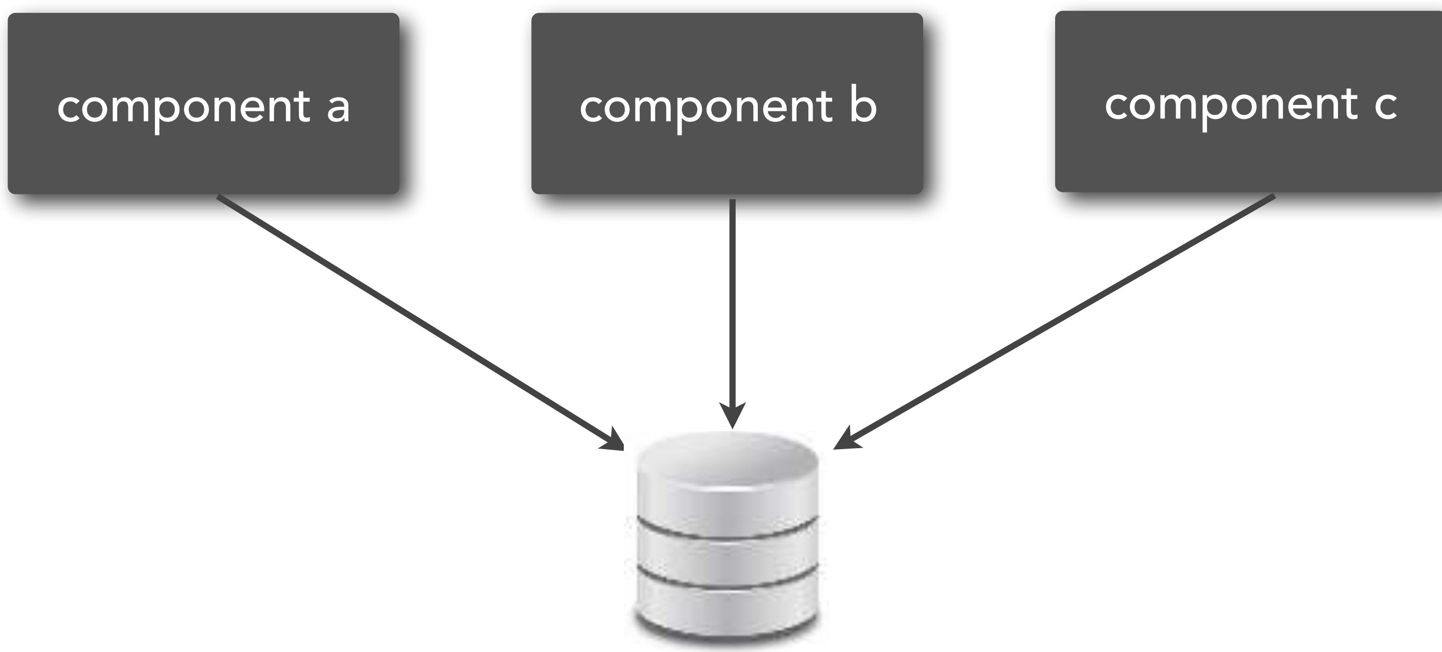
one component passes information to another component on what to do



component coupling

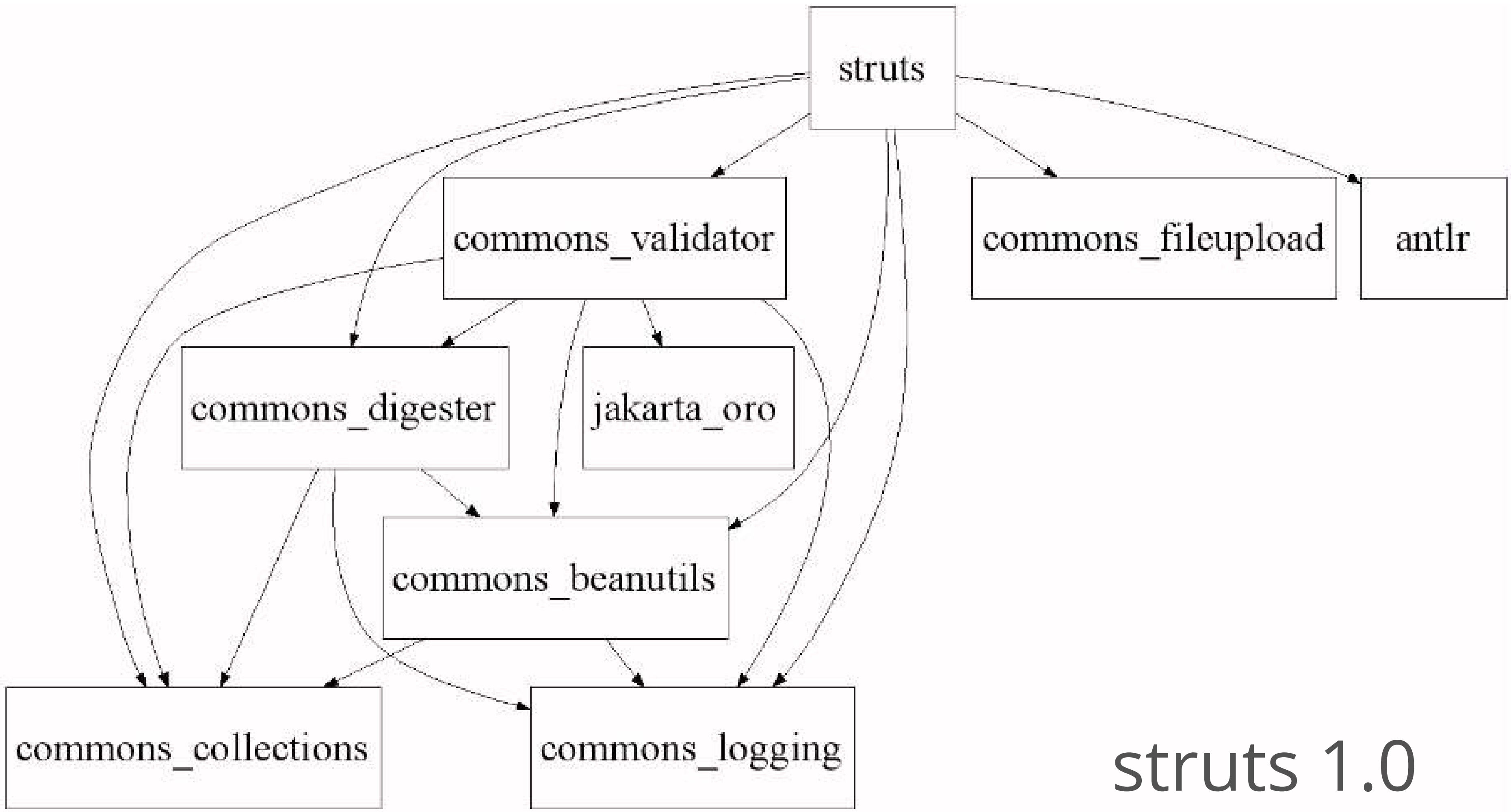
data coupling

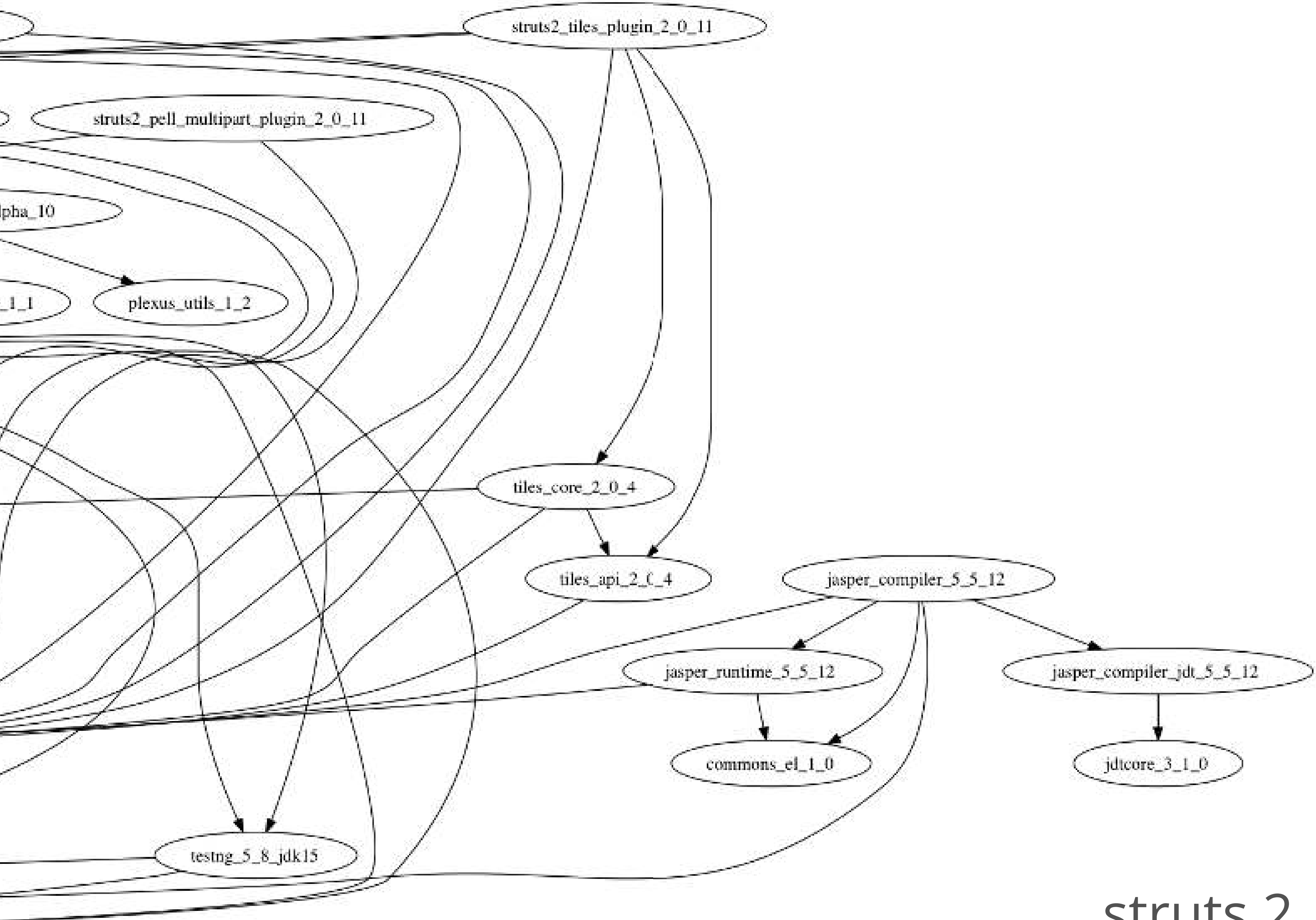
the degree to which components are bound to a shared data context



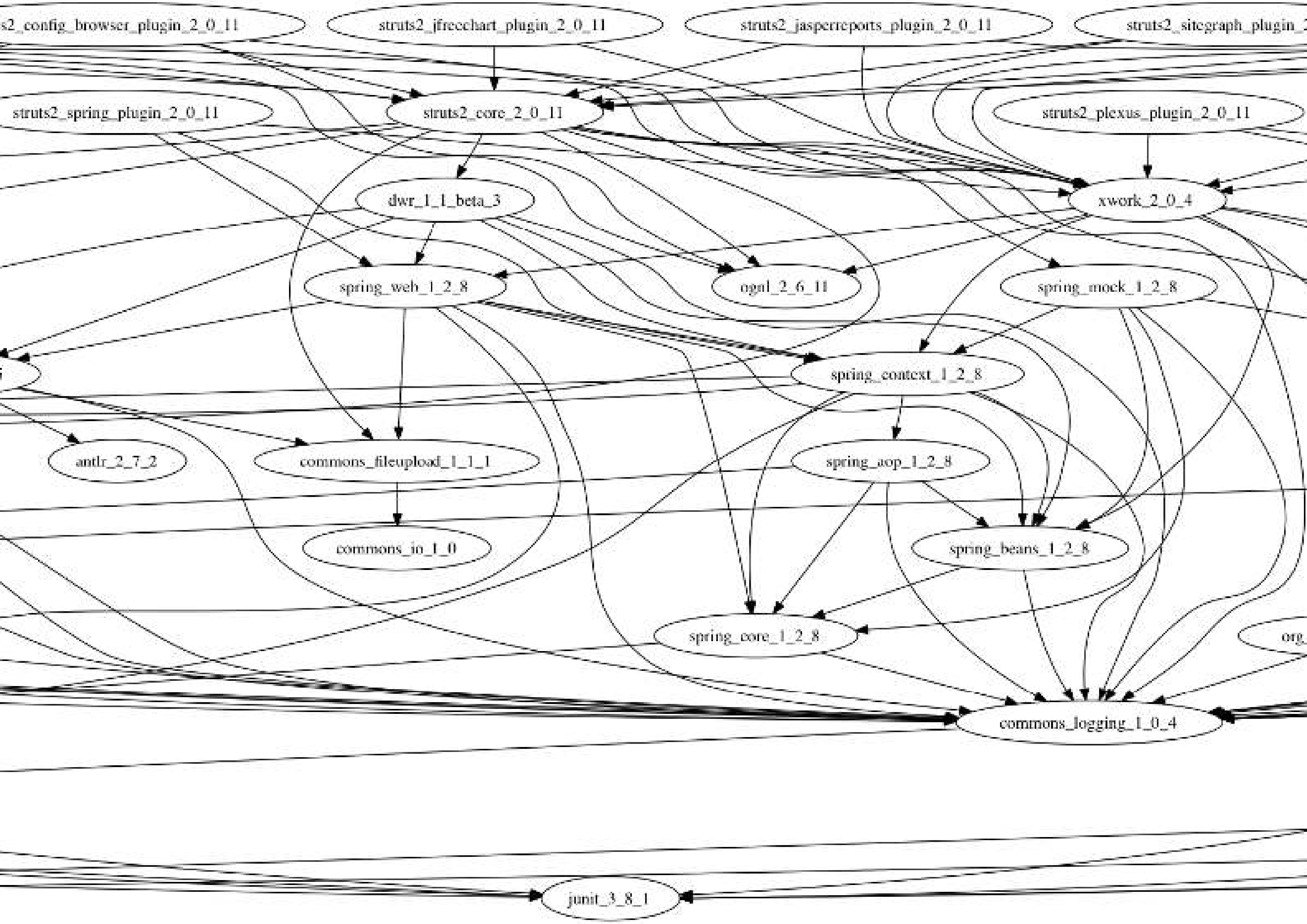
component coupling

consequences of ignoring…




struts2_tiles_plugin_2_0_11
struts2_pell_multipart_plugin_2_0_11
lpha_10
_1_1
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_
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
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

customer maintenance

- add customer
- update customer
- get customer
- notify customer

order maintenance

- get customer orders
- cancel customer orders

architecture katas

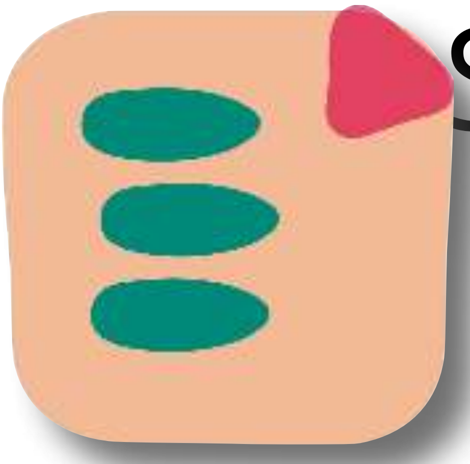
identifying major architecture components

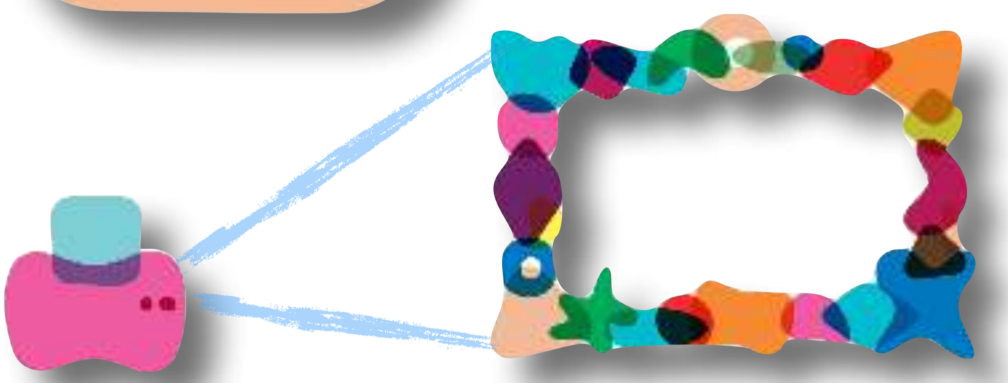

Your Architectural Kata is...

Make the Grade

assigned team: 1

Documenting & Presenting Software Architecture





ThoughtWorks®

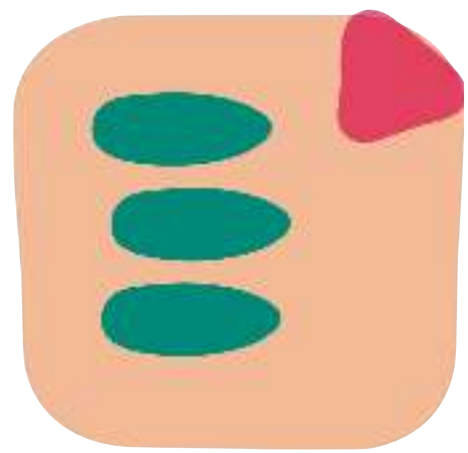
NEAL FORD

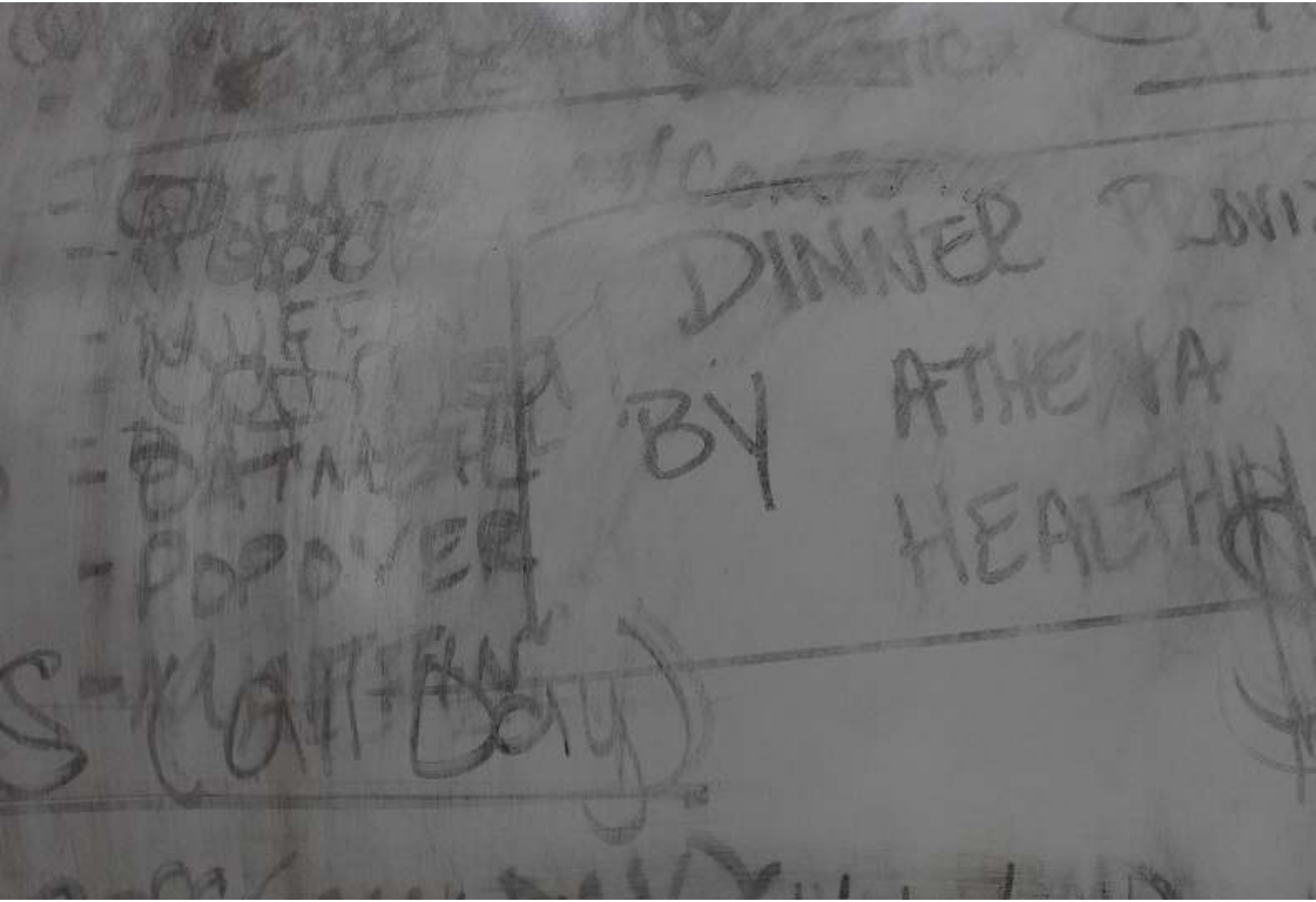
Director / Software Architect / Meme Wrangler

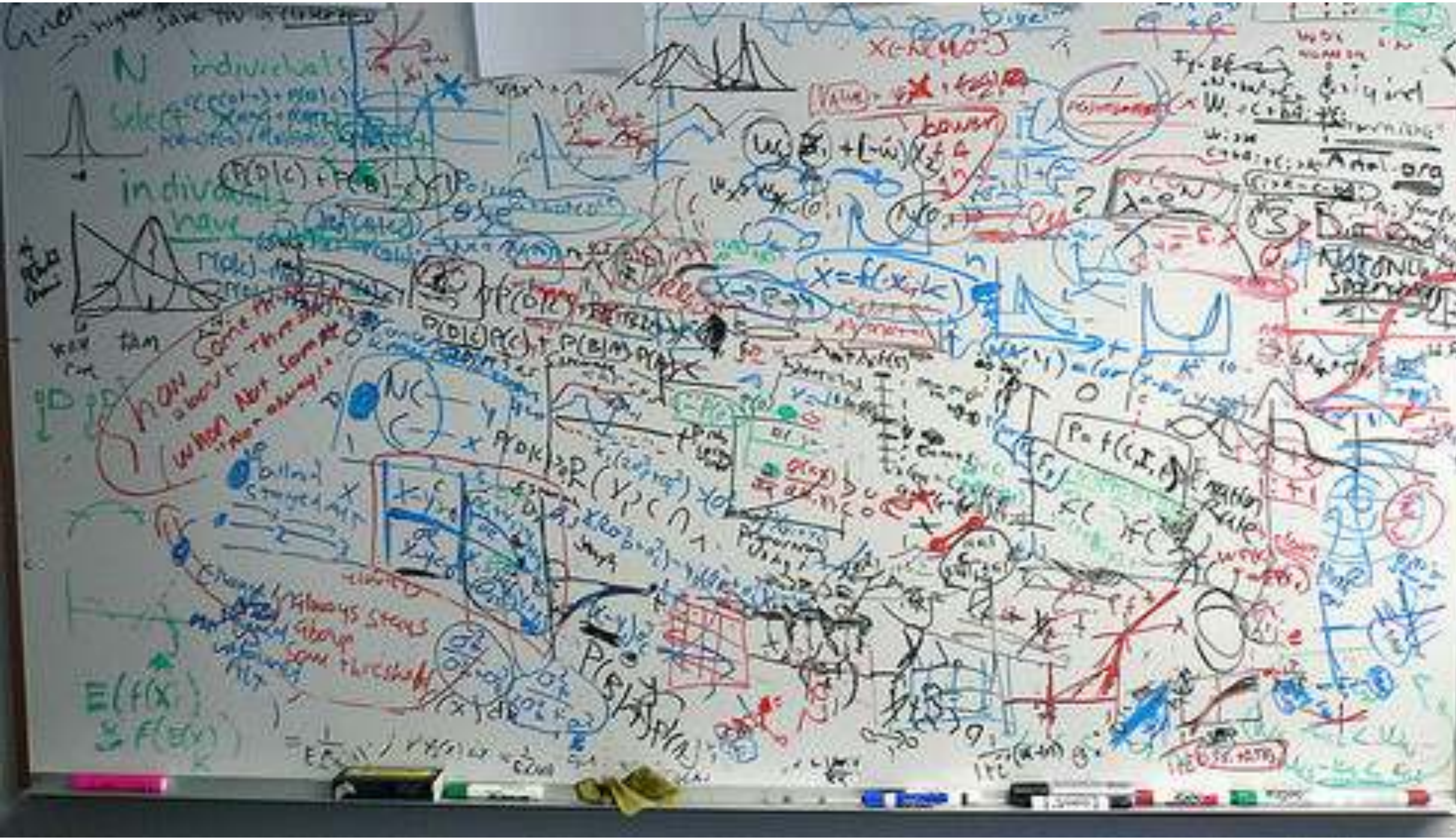
@neal4d

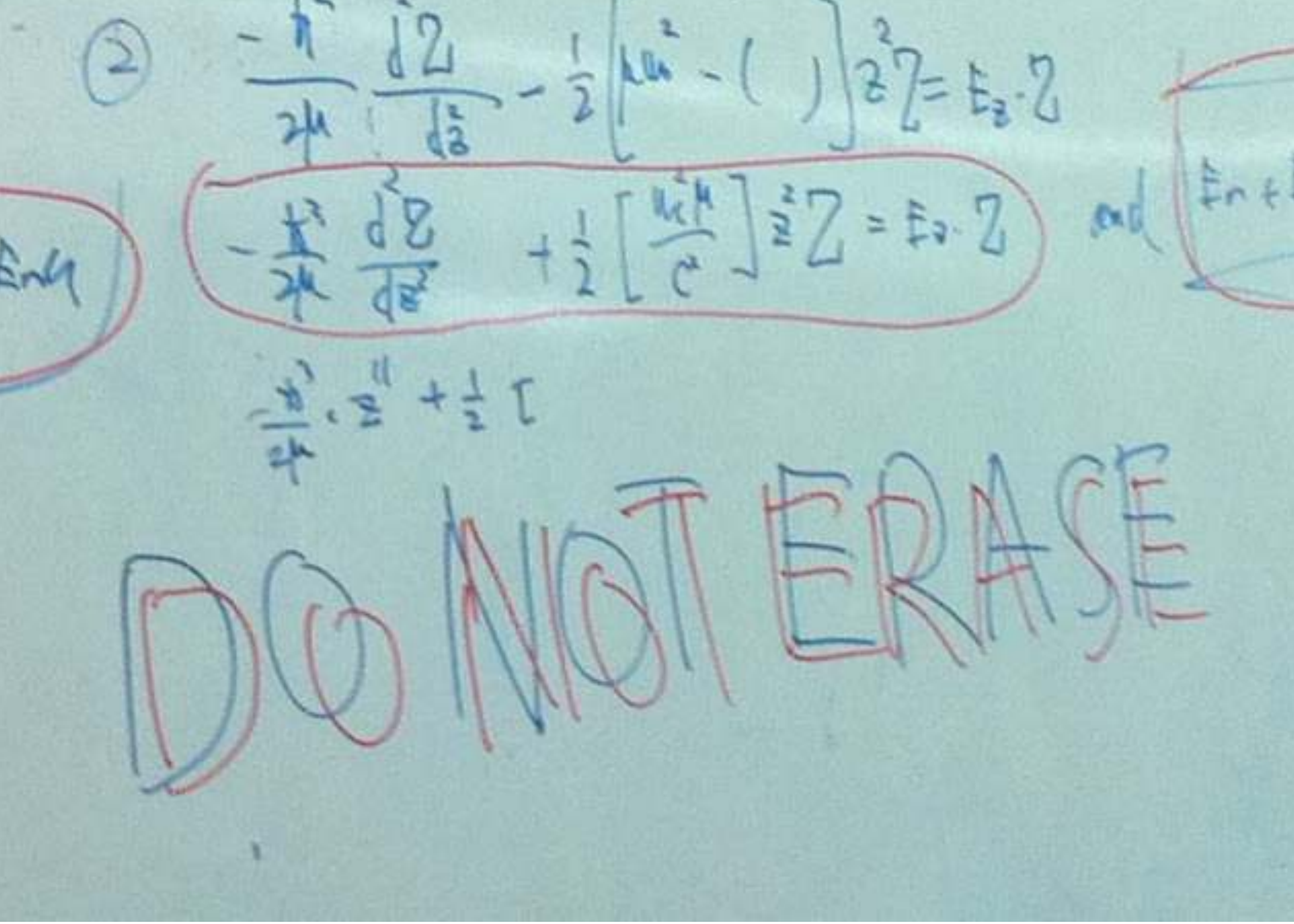
nealford.co

documenting software




DINNER
BY ATHENA
HEALTH


N individuals
always steals
above
some threshold

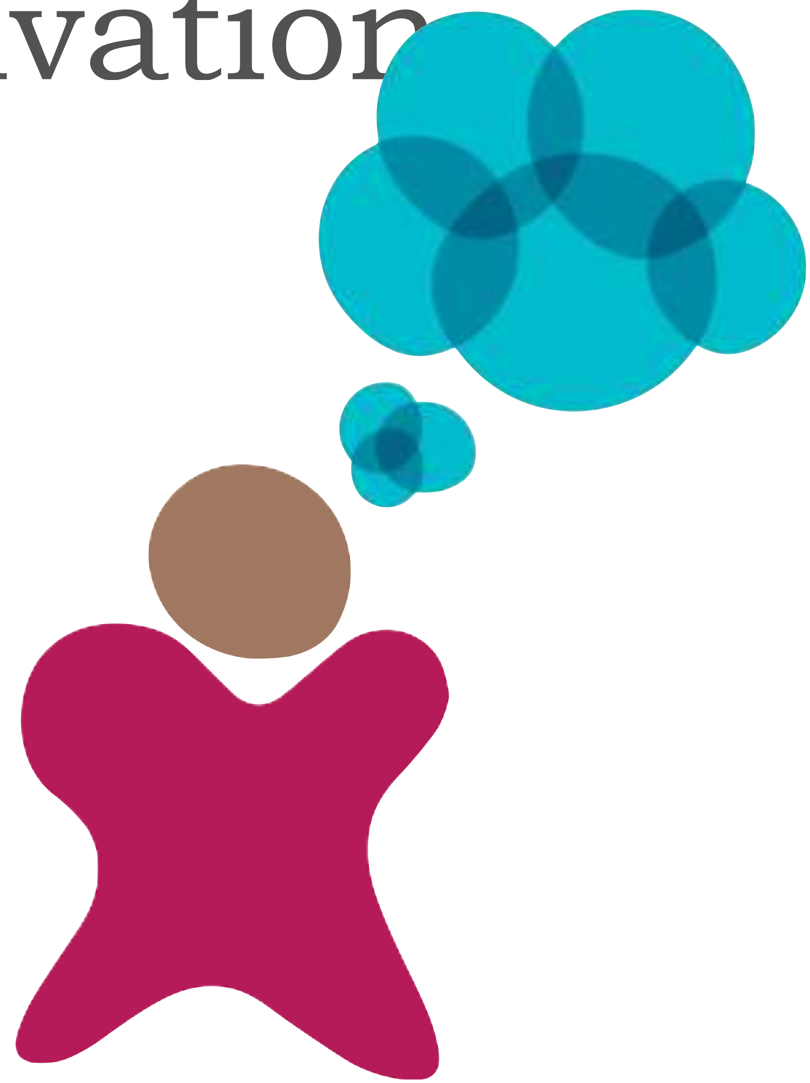

DO NOT ERASE


9:41
Wednesday, September 7

documentation

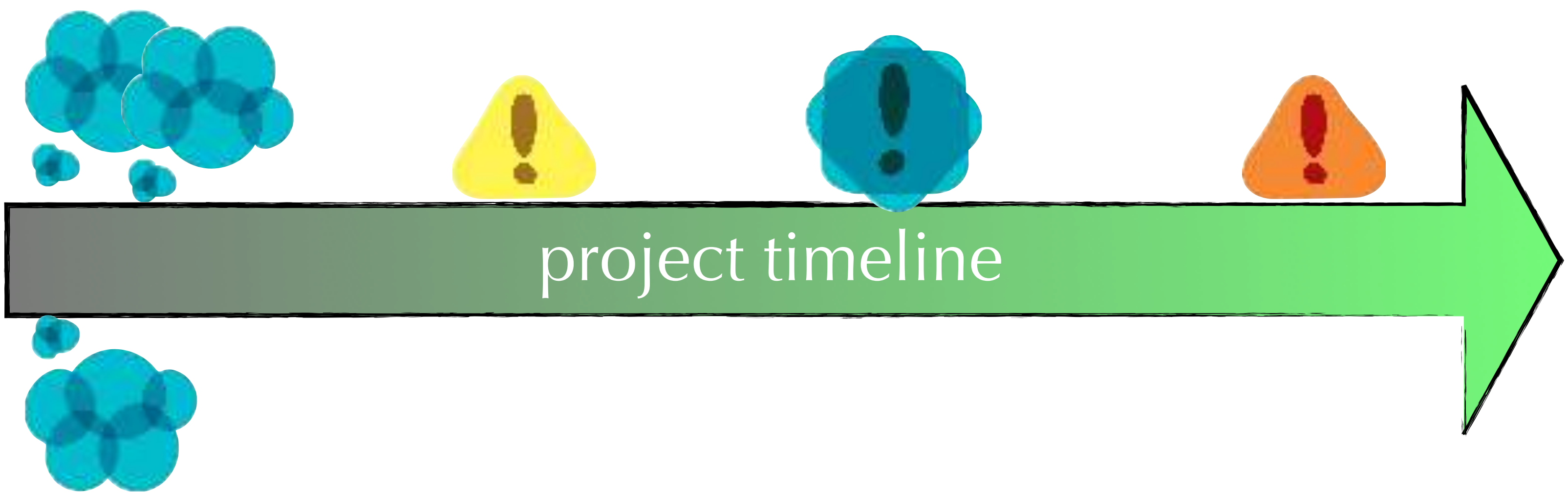
weight in bytes

motivation



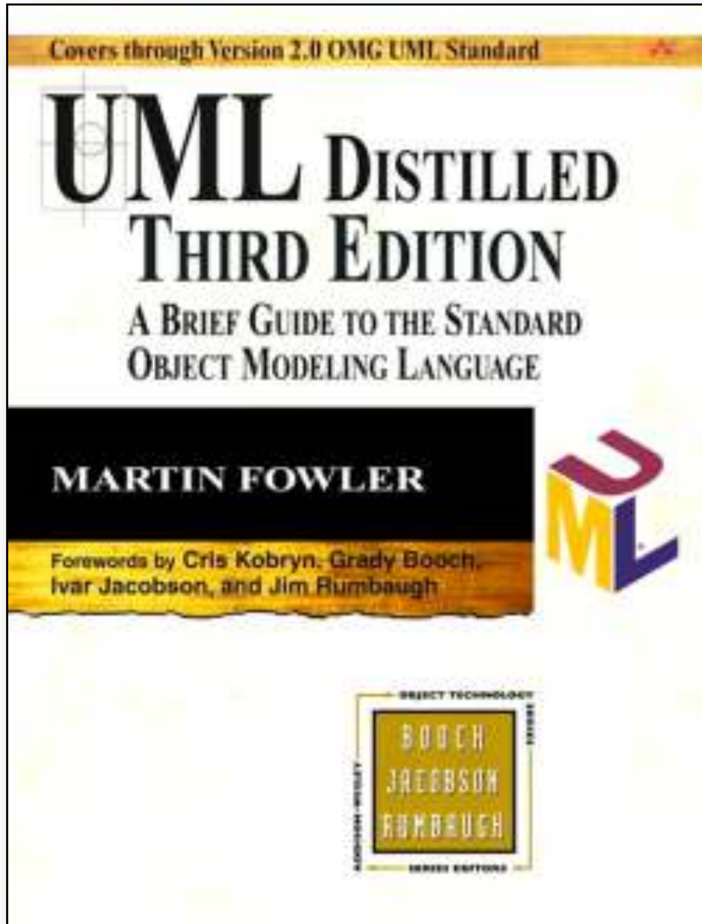
agile architecture

Not all decisions made up front.

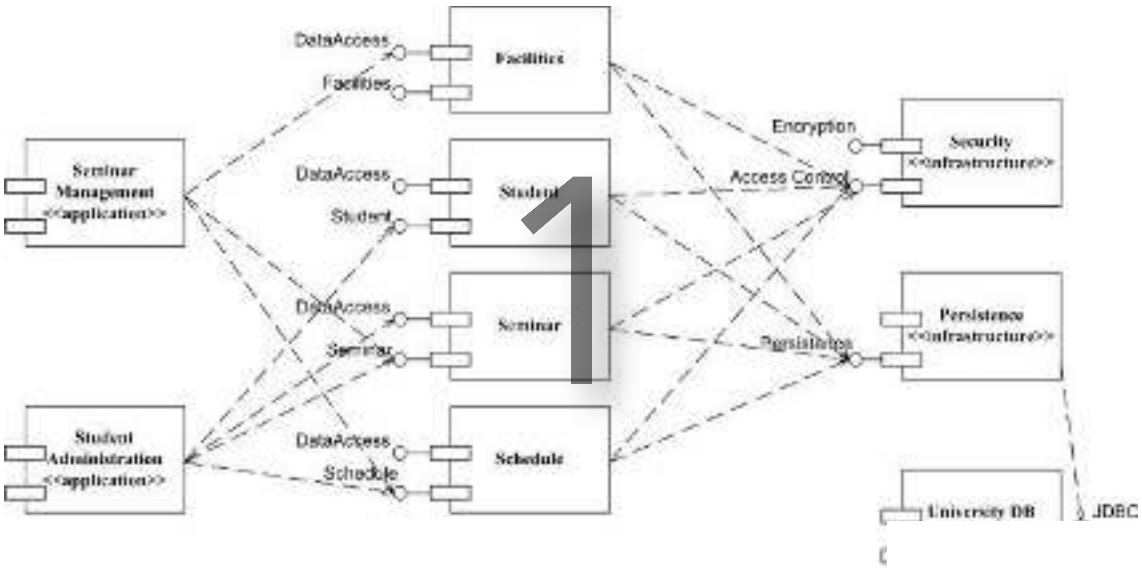


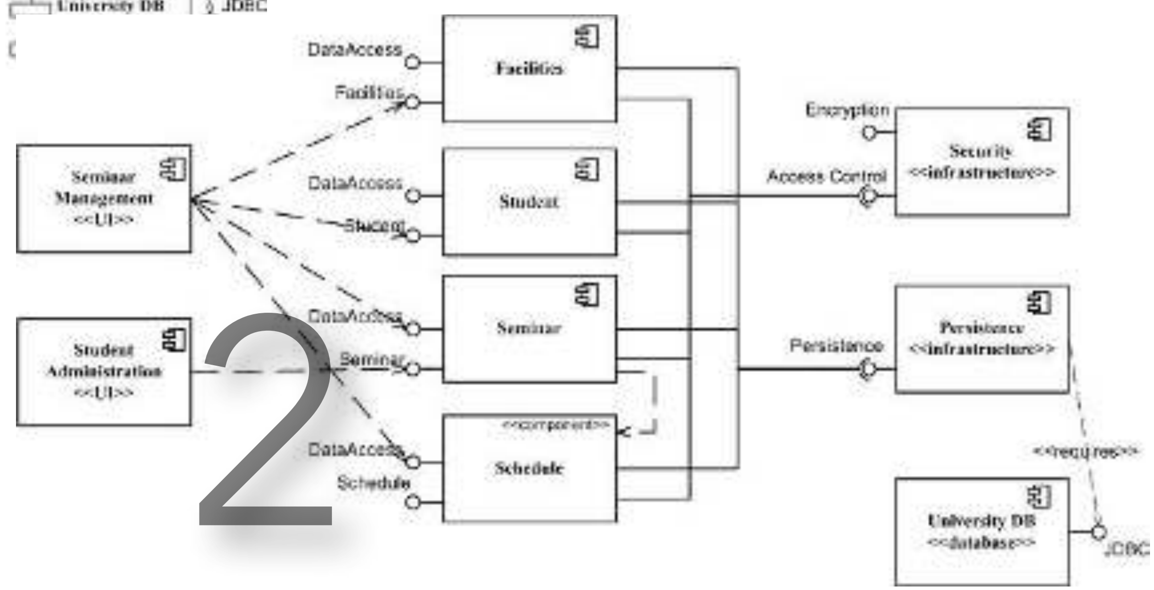
UNIFIED
MODELING
LANGUAGE

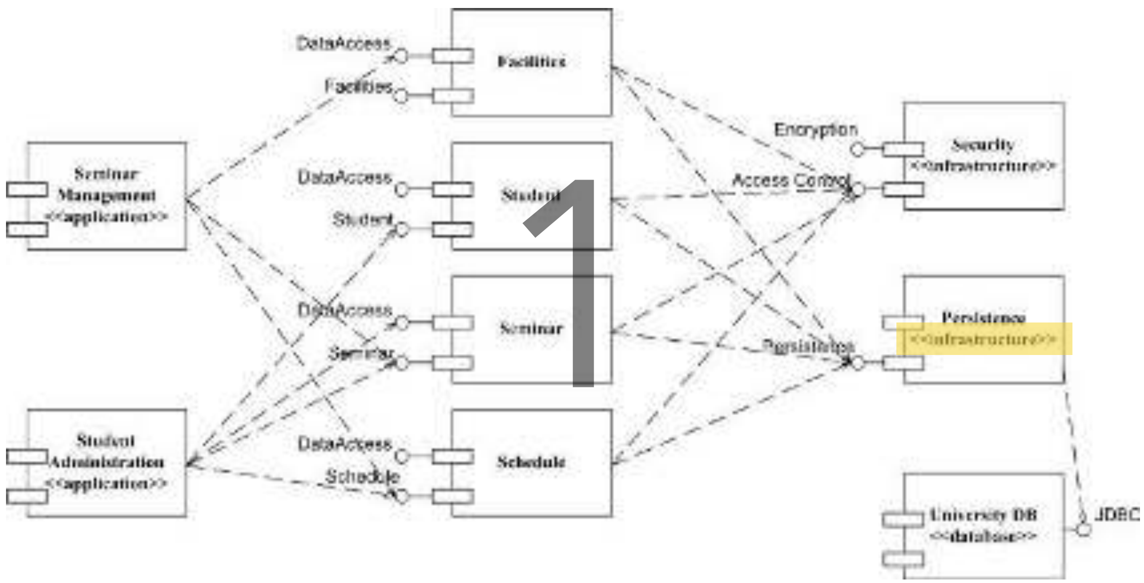
UML™ ???

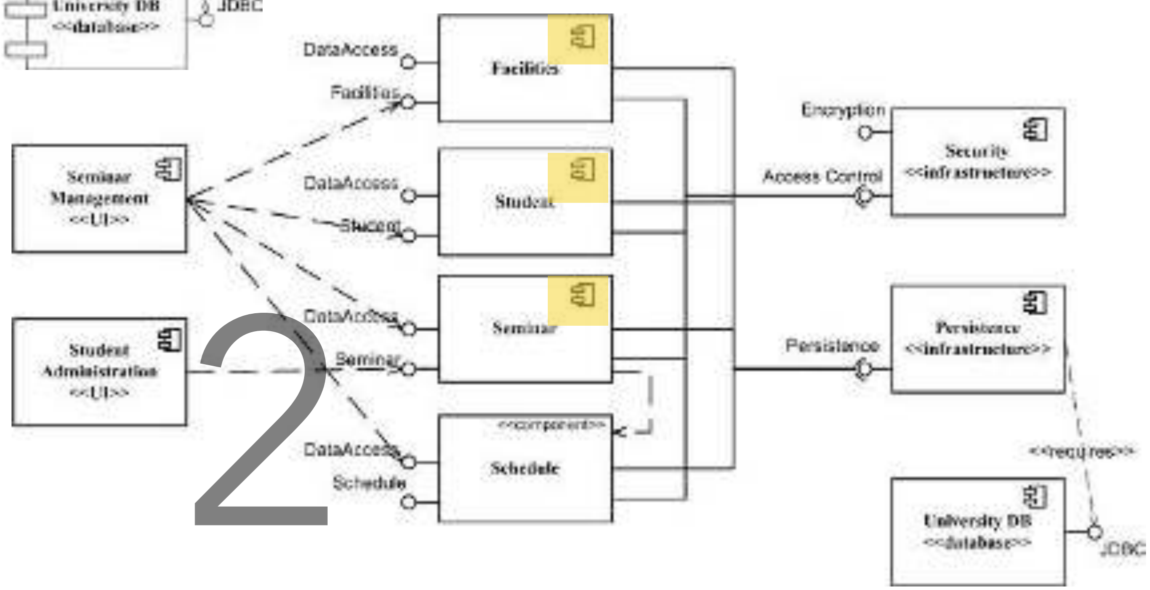


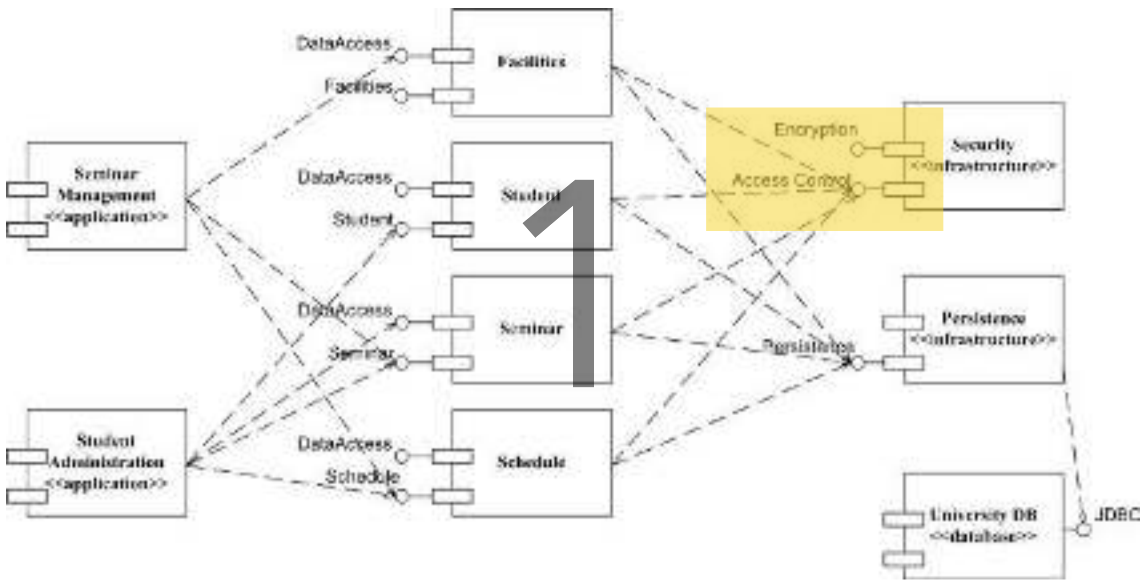
https://martinfowler.com/books/uml.html

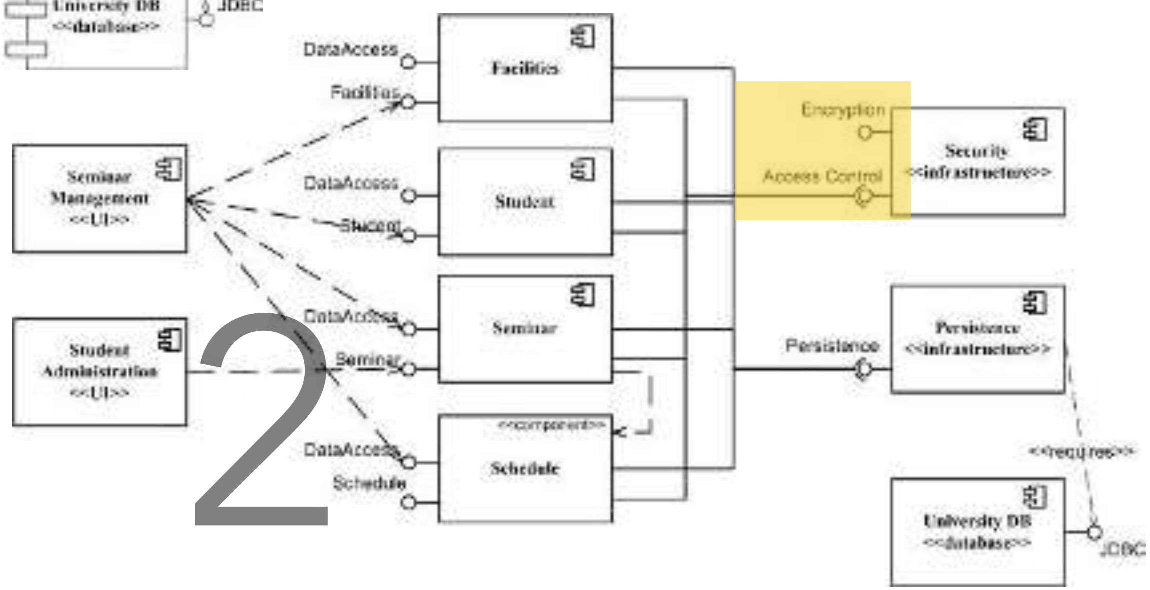

1
Seminar Management <<application>>
Student Administration <<application>>
DataAccess
Facilities
Student
Seminar
Schedule
Encryption
Access Control
Persistence
Security <<infrastructure>>
Persistence <<infrastructure>>
University DB
JDBC

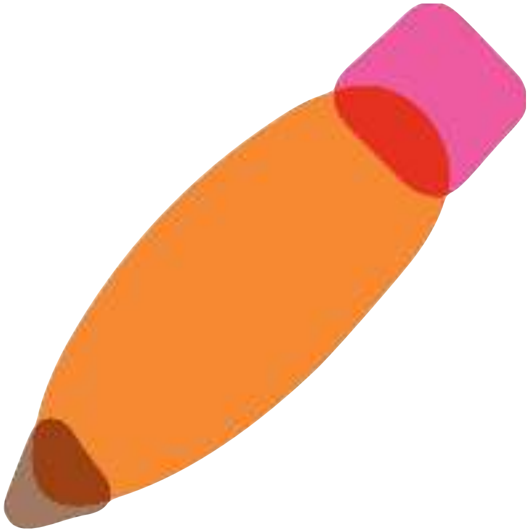

2
Seminar Management <<UI>>
Student Administration <<UI>>
DataAccess
Facilities
Student
Seminar
Schedule
<<component>>
Encryption
Access Control
Persistence
Security <<infrastructure>>
Persistence <<infrastructure>>
<<requires>>
University DB <<database>>
JDBC


1
DataAccess
Facilities
Facilities
Seminar Management <<application>>
DataAccess
Student
Student
DataAccess
Seminar
Seminar
DataAccess
Schedule
Schedule
Student Administration <<application>>
Encryption
Access Control
Security <<infrastructure>>
Persistence <<infrastructure>>
Persistence
University DB <<database>>
JDBC

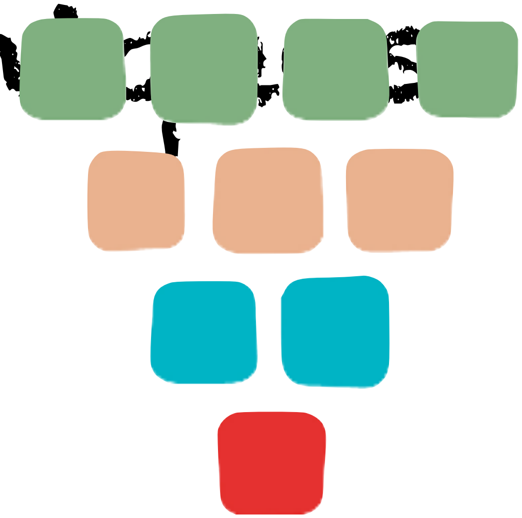

2
DataAccess
Facilities
Facilities
Seminar Management <<UI>>
DataAccess
Student
Student
DataAccess
Seminar
Seminar
Student Administration <<UI>>
DataAccess
Schedule
<<component>>
Schedule
Encryption
Access Control
Security <<infrastructure>>
Persistence
Persistence <<infrastructure>>
<<requires>>
University DB <<database>>
JDBC

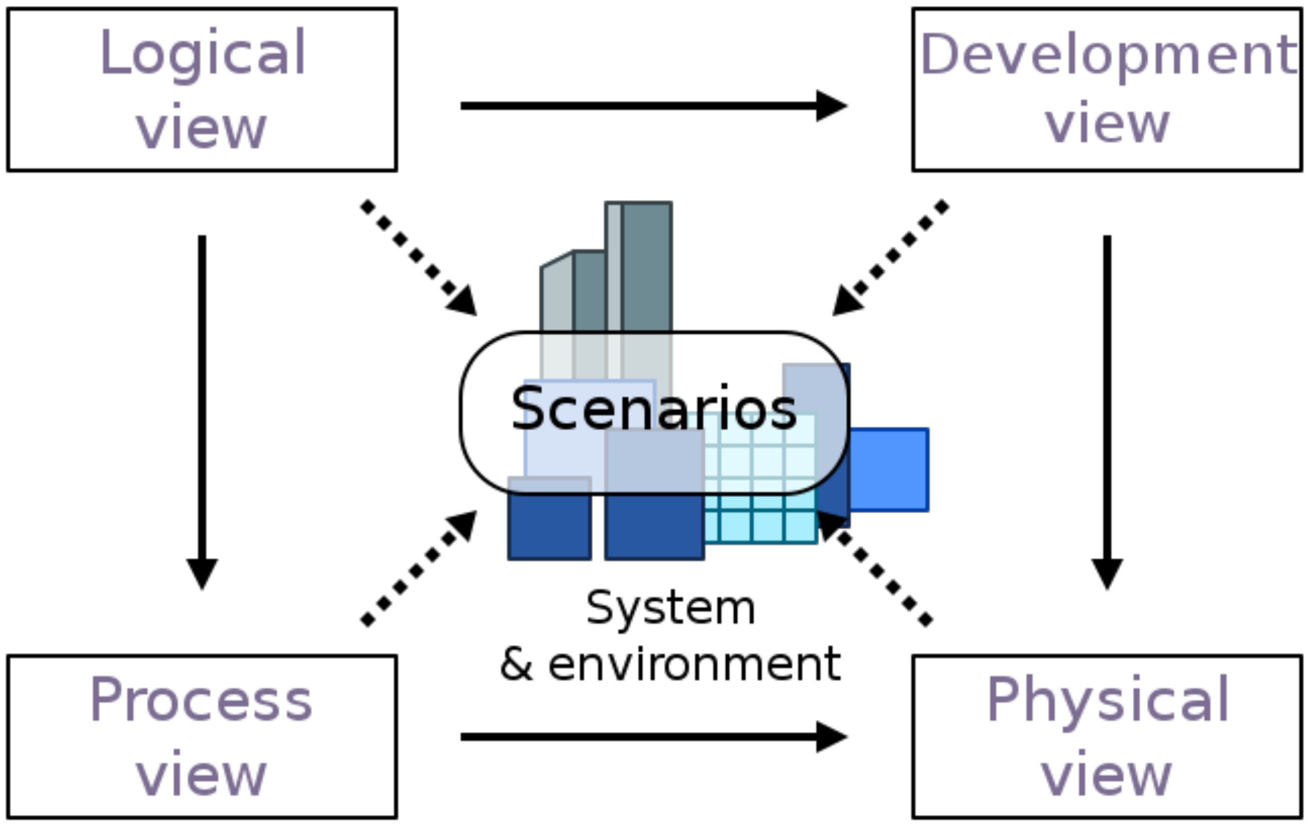

1
Seminar Management <<application>>
Student Administration <<application>>
DataAccess
Facilities
Student
Seminar
Schedule
Encryption
Access Control
Security <<infrastructure>>
Persistence
Persistence <<infrastructure>>
University DB <<database>>
JDBC


2
Seminar Management <<UI>>
Student Administration <<UI>>
DataAccess
Facilities
Student
Seminar
Schedule
<<component>>
Encryption
Access Control
Security <<infrastructure>>
Persistence
Persistence <<infrastructure>>
<<requires>>
University DB <<database>>
JDBC



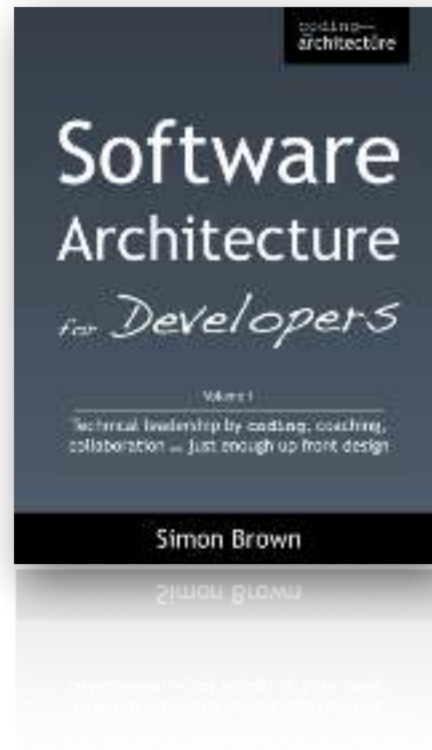
Architectural Diagraming Techniques



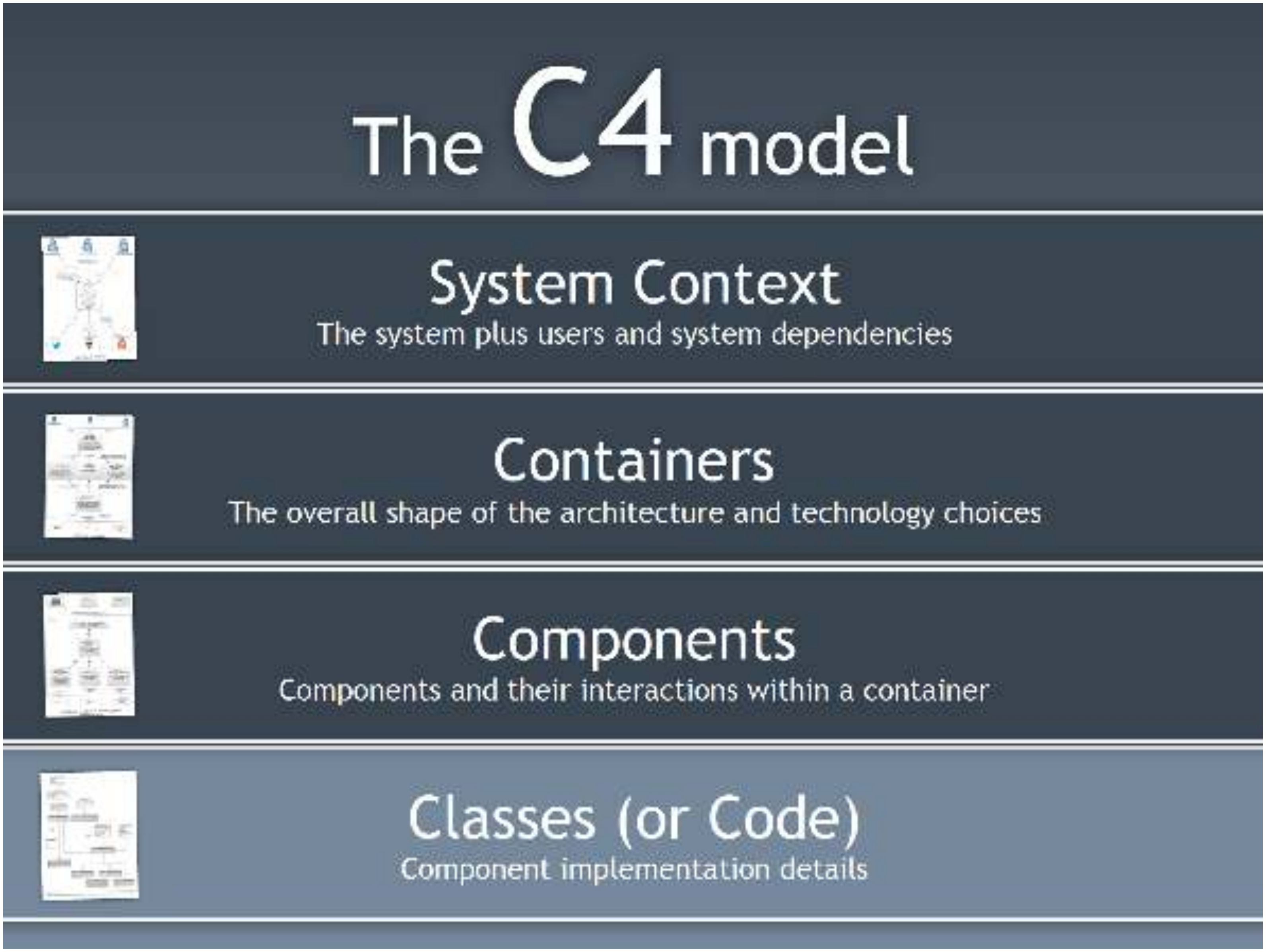


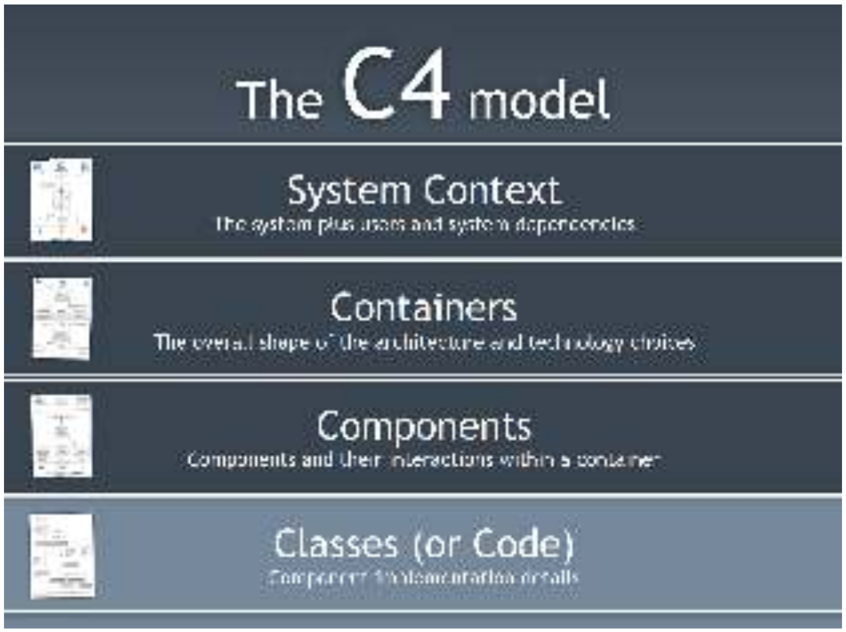
https://en.wikipedia.org/wiki/4+1_architectural_view_model

C4

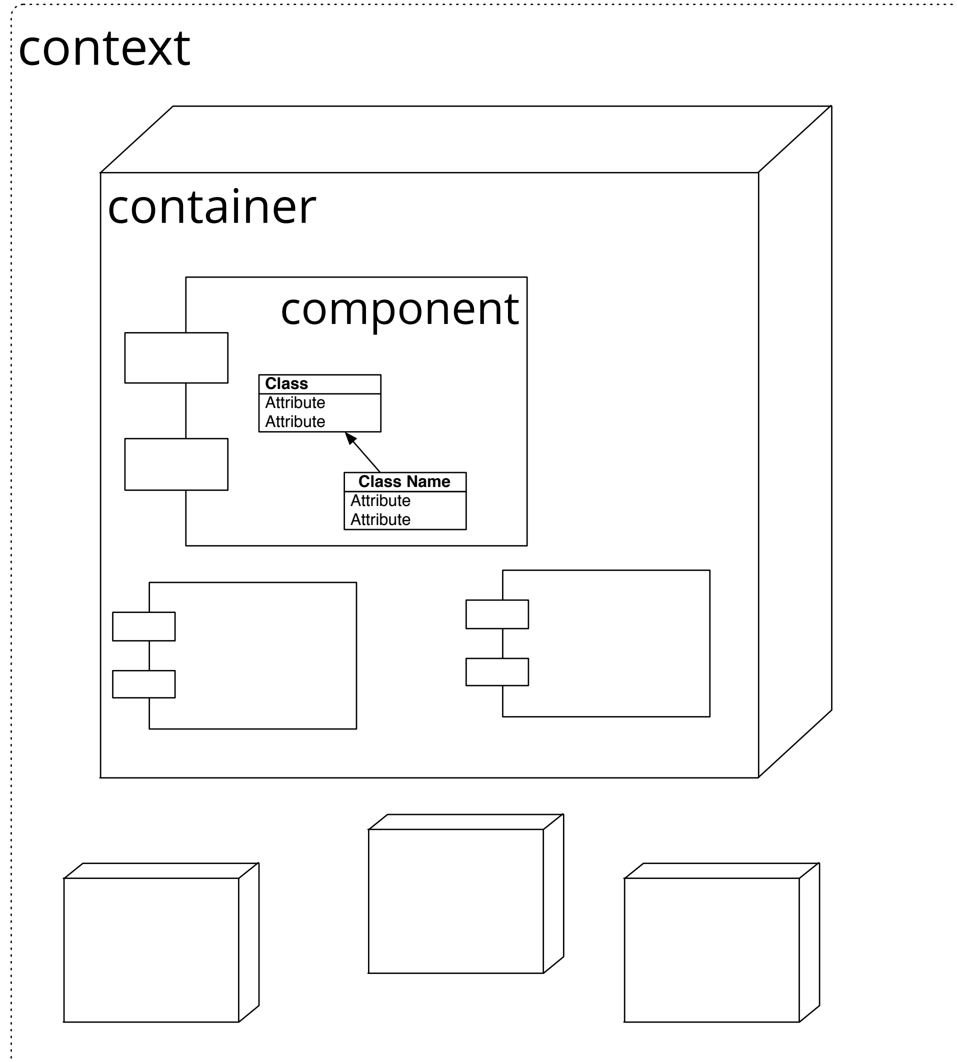


http://www.codingthearchitecture.com

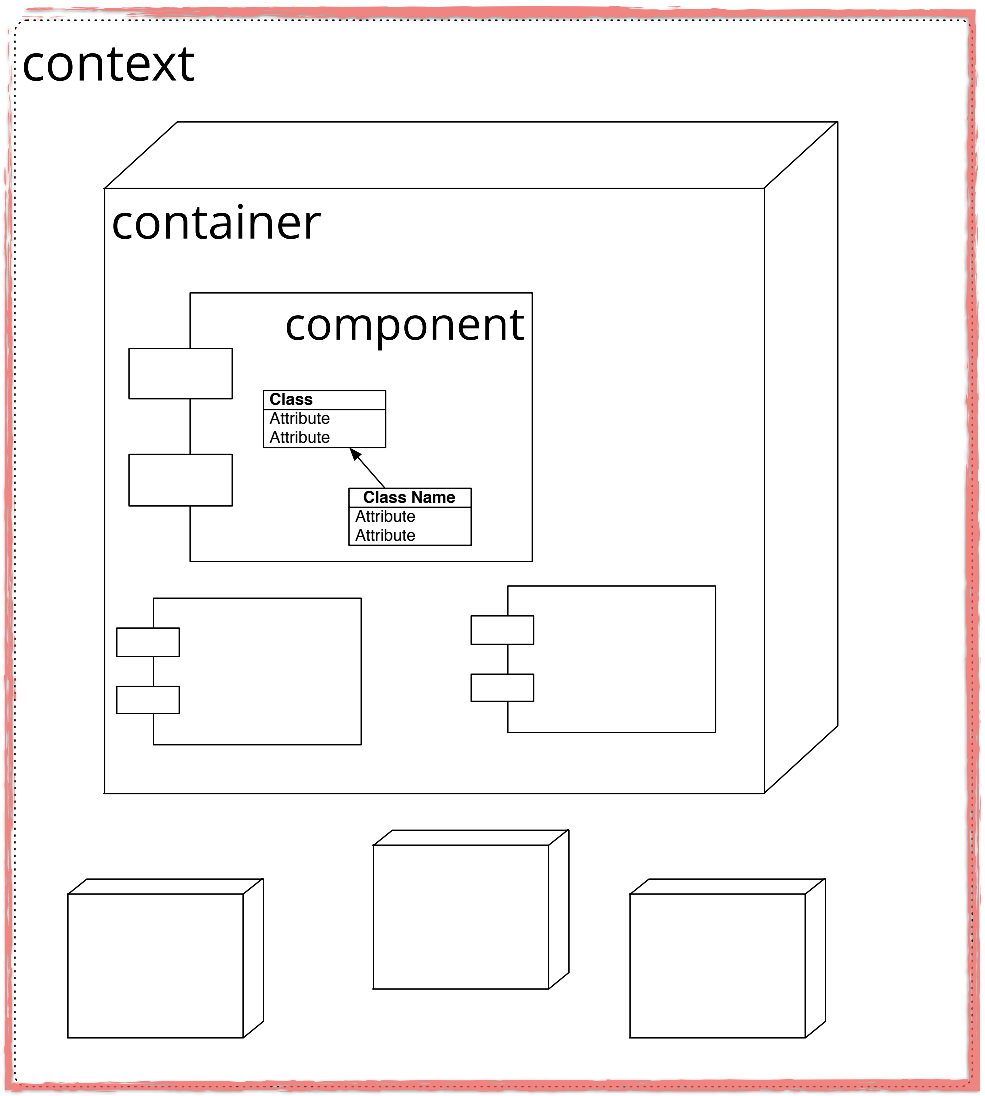

The C4 model
System Context
The system plus users and system dependencies
Containers
The overall shape of the architecture and technology choices
Components
Components and their interactions within a container
Classes (or Code)
Component implementation details

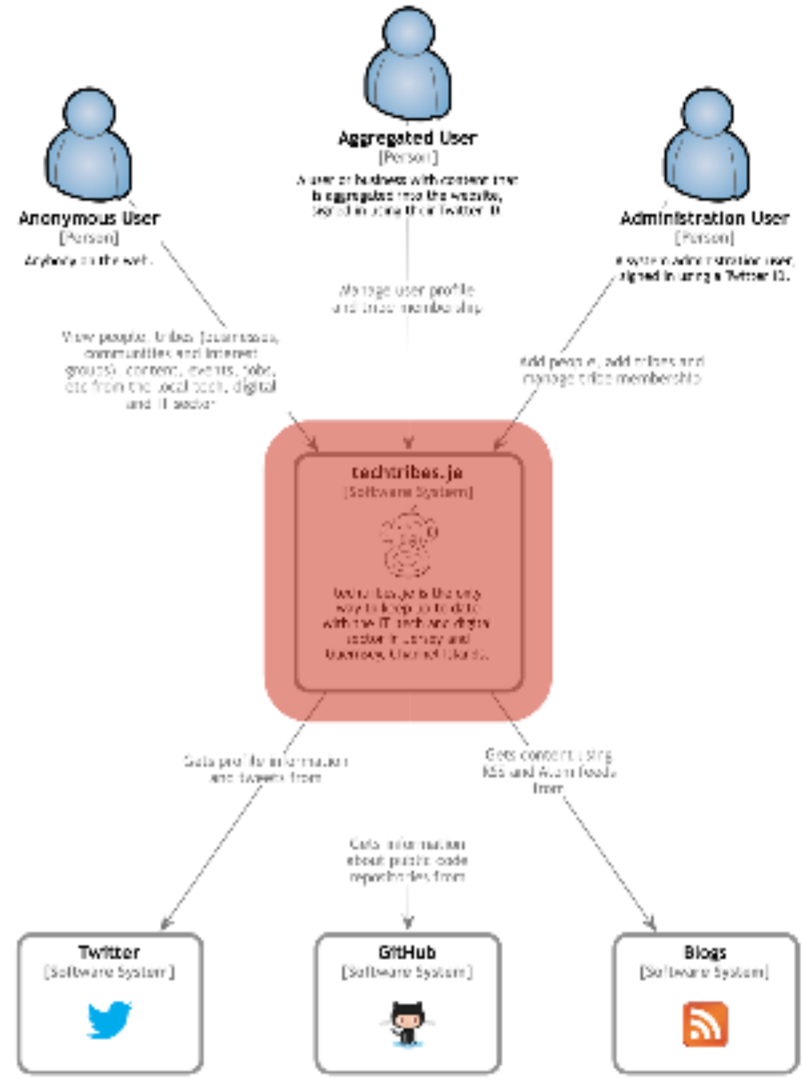


The C4 model
System Context
The system plus users and system dependencies
Containers
The overall shape of the architecture and technology choices
Components
Components and their interactions within a container
Classes (or Code)

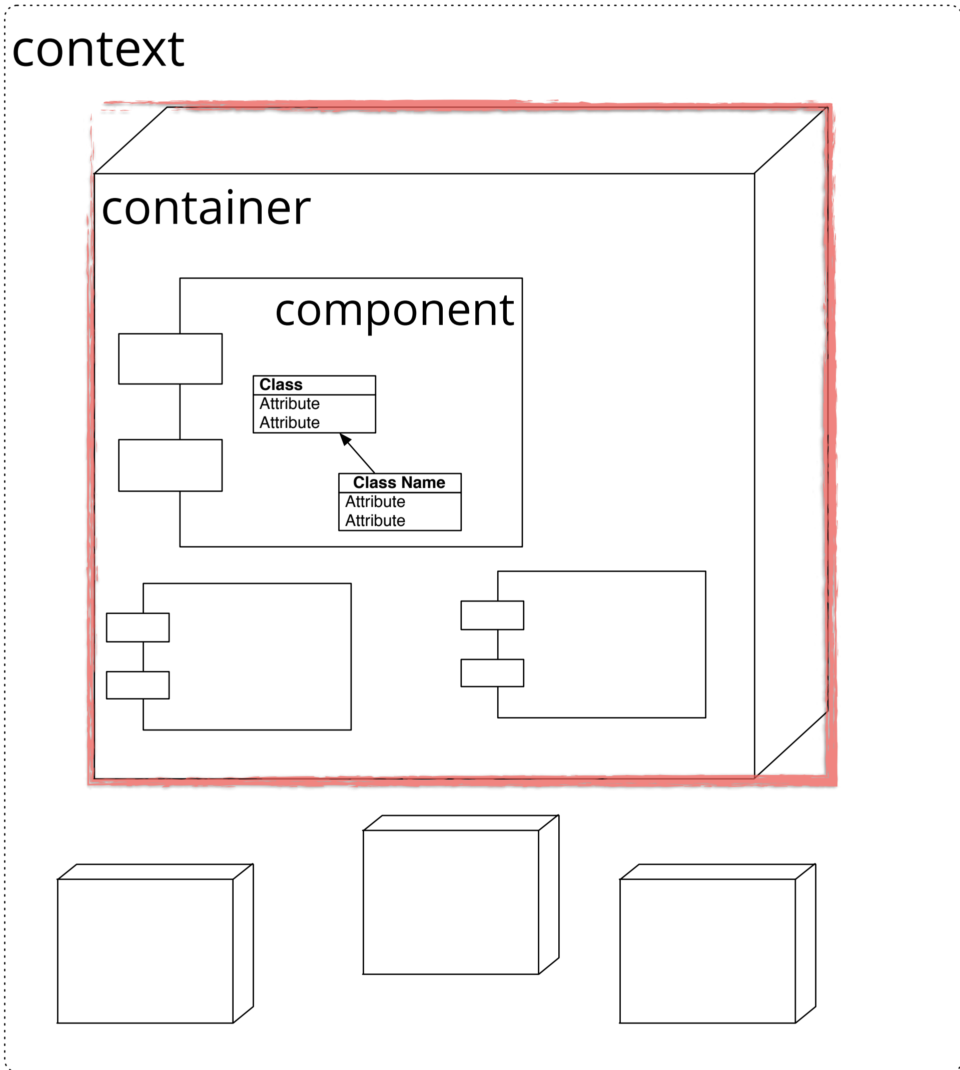


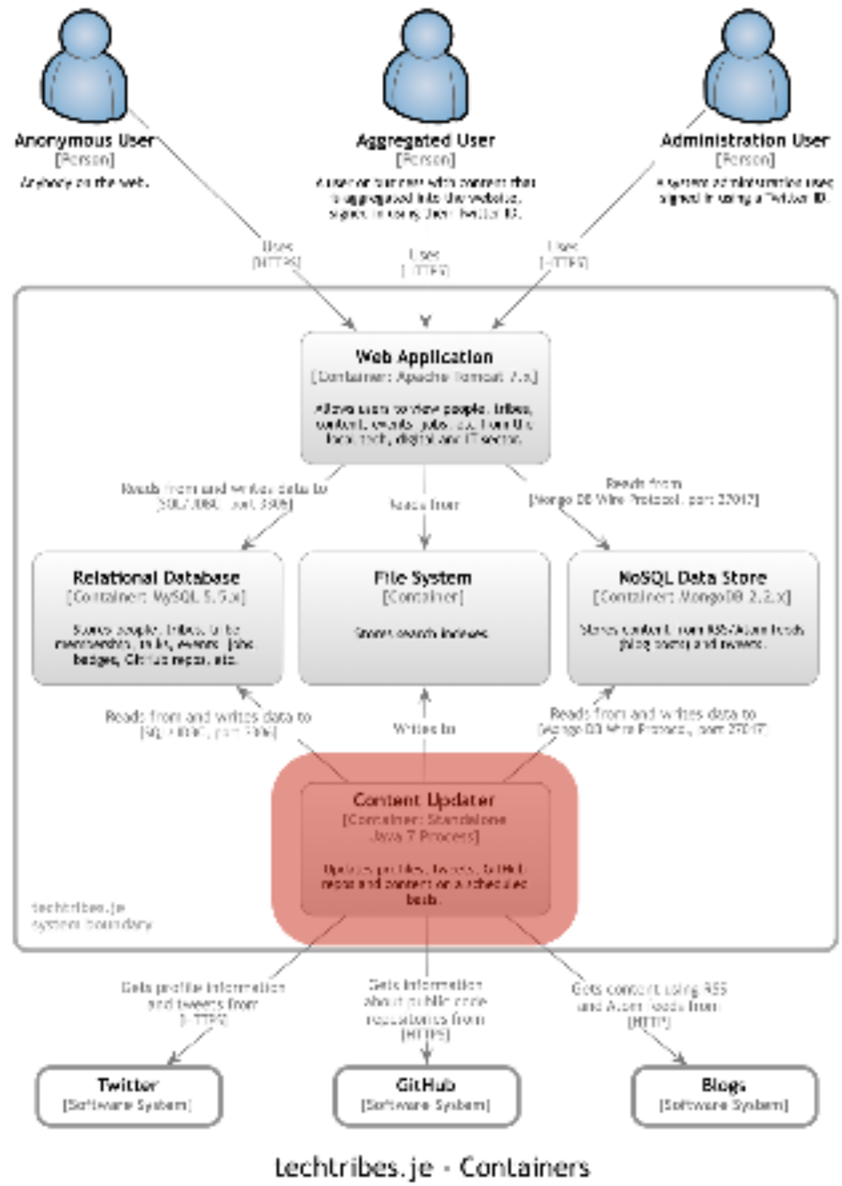
context
container
component
Class
Attribute
Attribute
Class Name
Attribute
Attribute

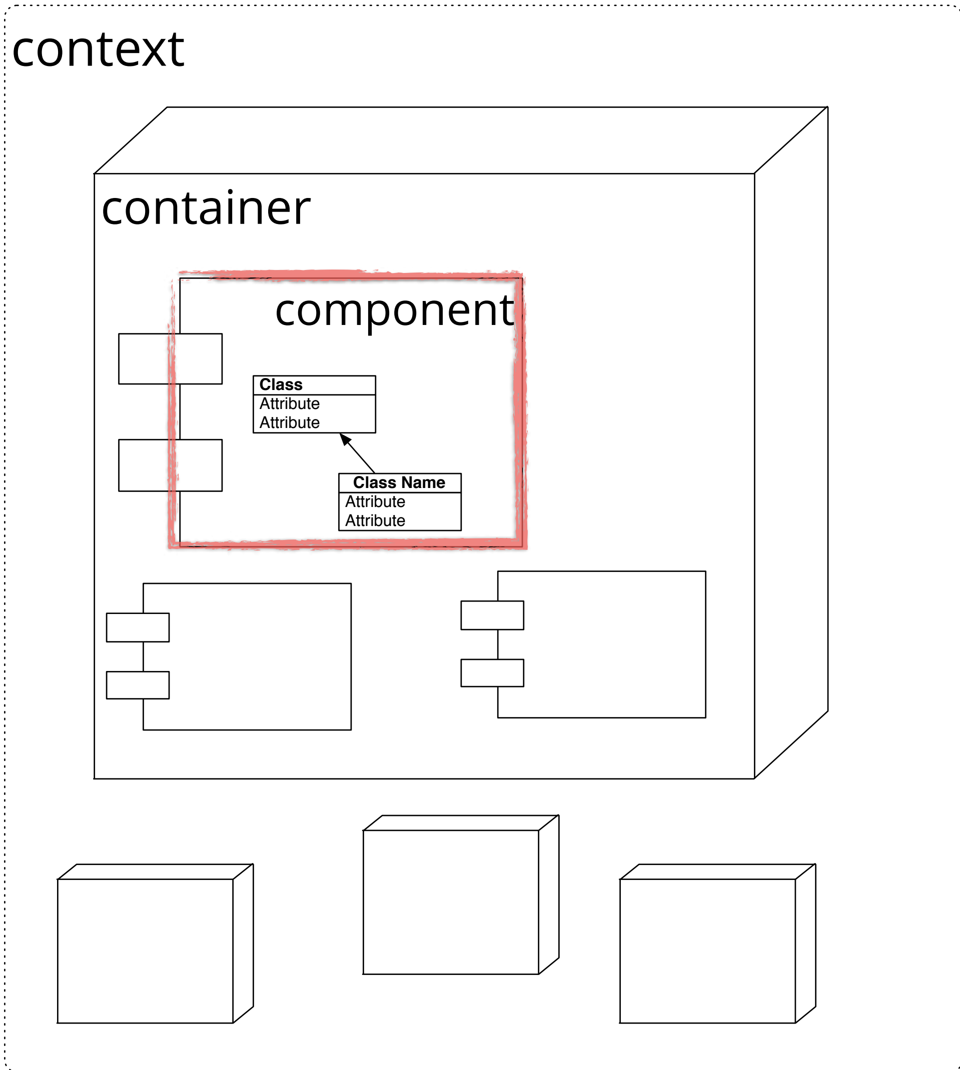


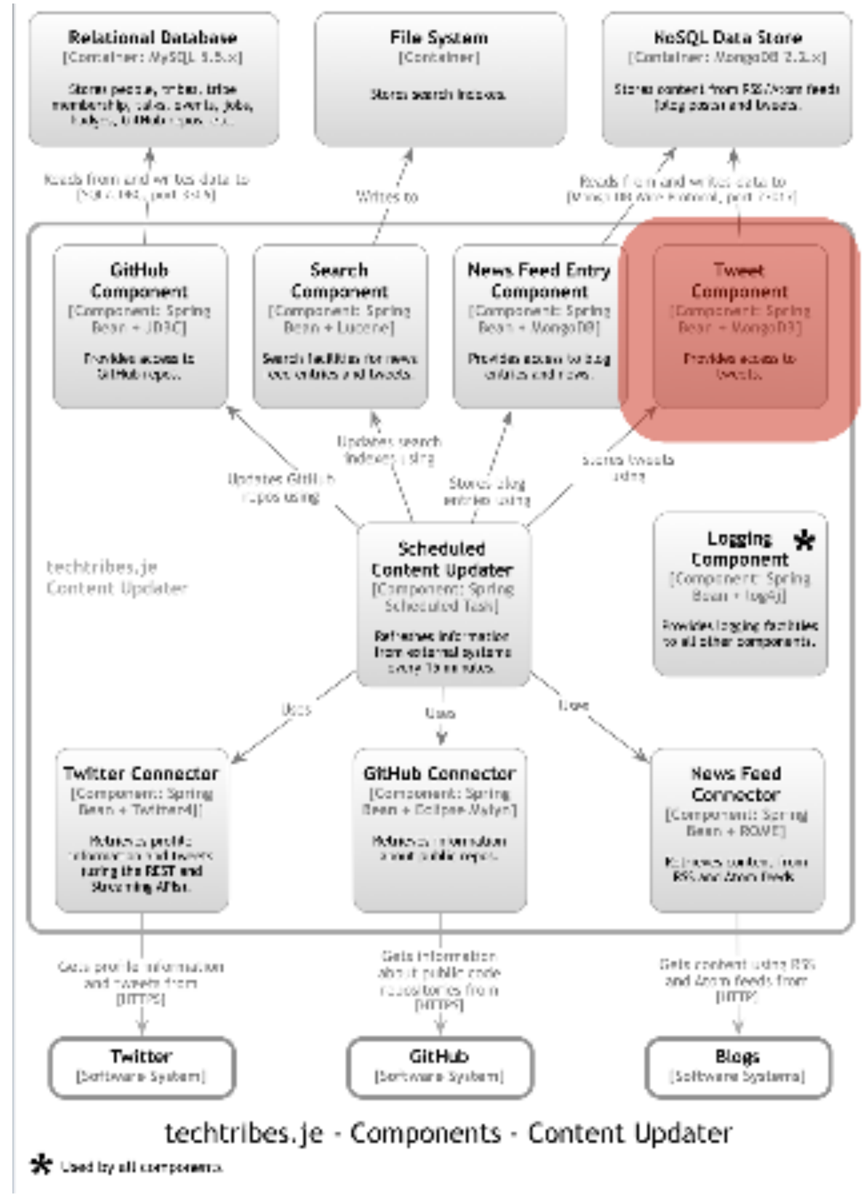


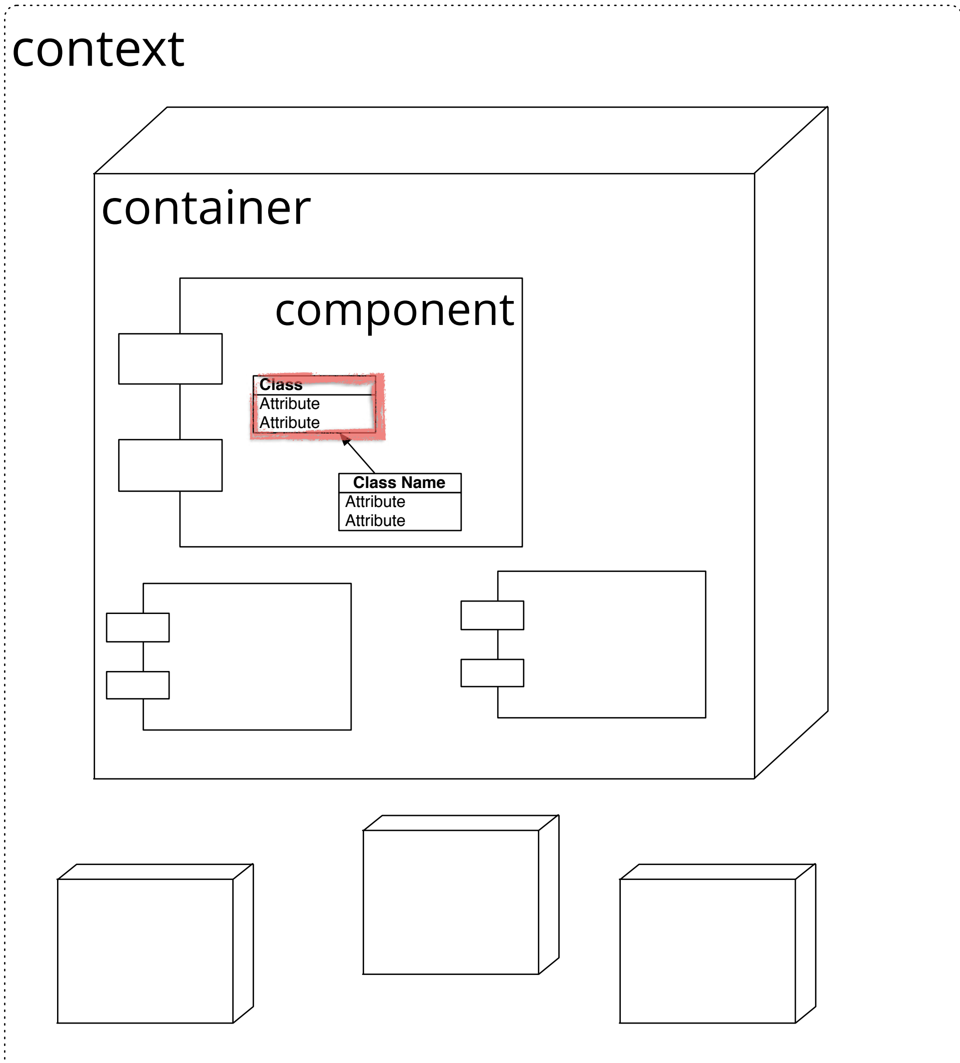
techtribes.je - Context

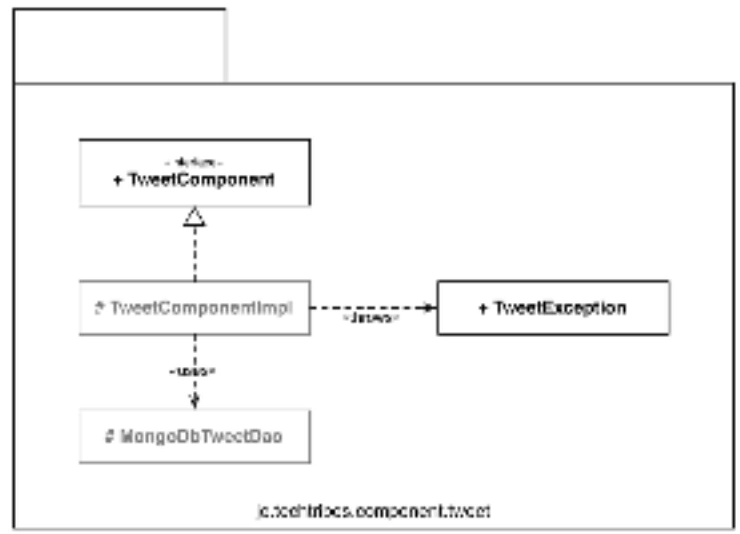

context
container
component
Class
Attribute
Attribute
Class Name
Attribute
Attribute

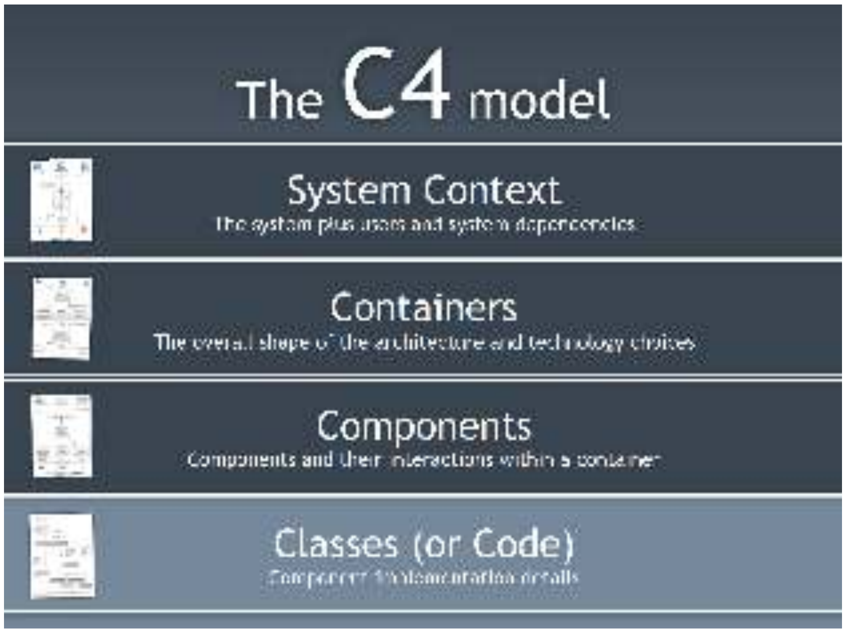

Anonymous User
[Person]
Aggregated User
[Person]
Administration User
[Person]
Web Application
Relational Database
File System
[Container]
NoSQL Data Store
Content Updater
Twitter
[Software System]
GitHub
[Software System]
Blogs
[Software System]
Techtribes.je - Containers

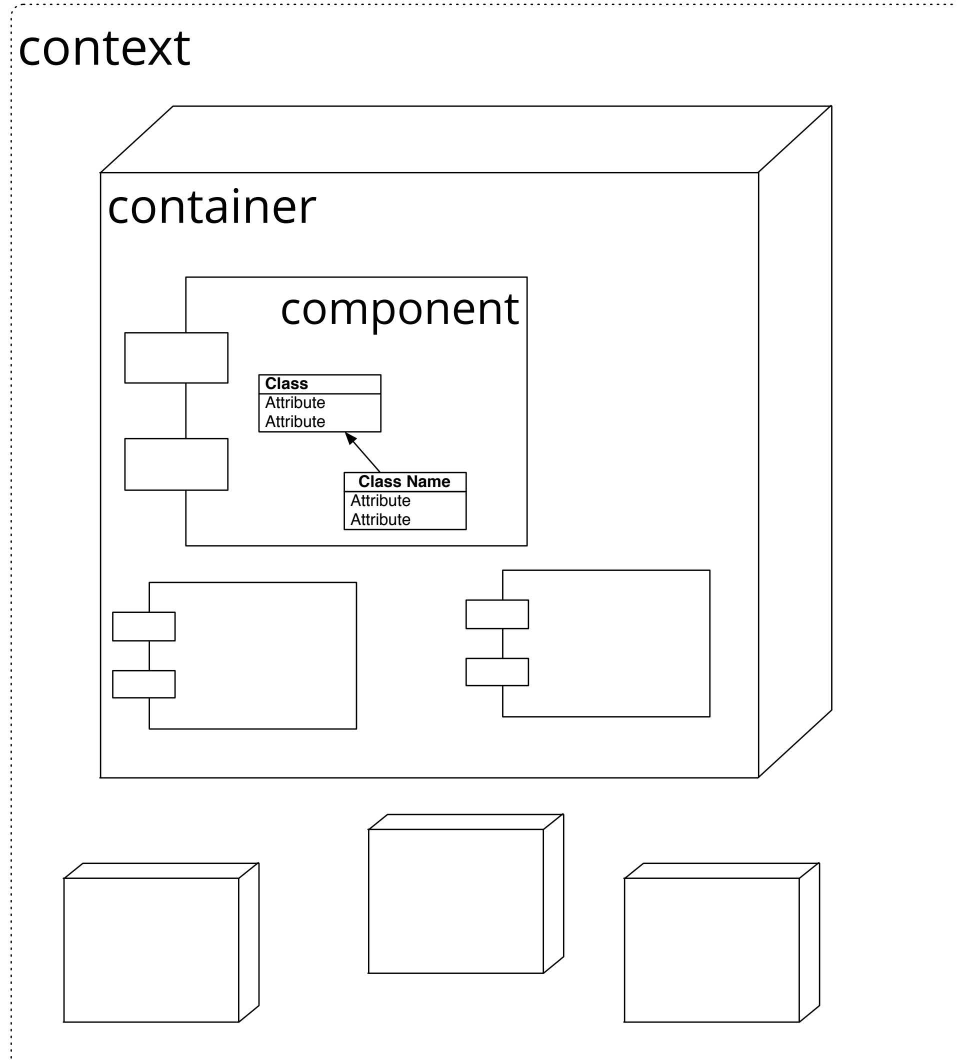

context
container
component
Class
Attribute
Attribute
Class Name
Attribute
Attribute

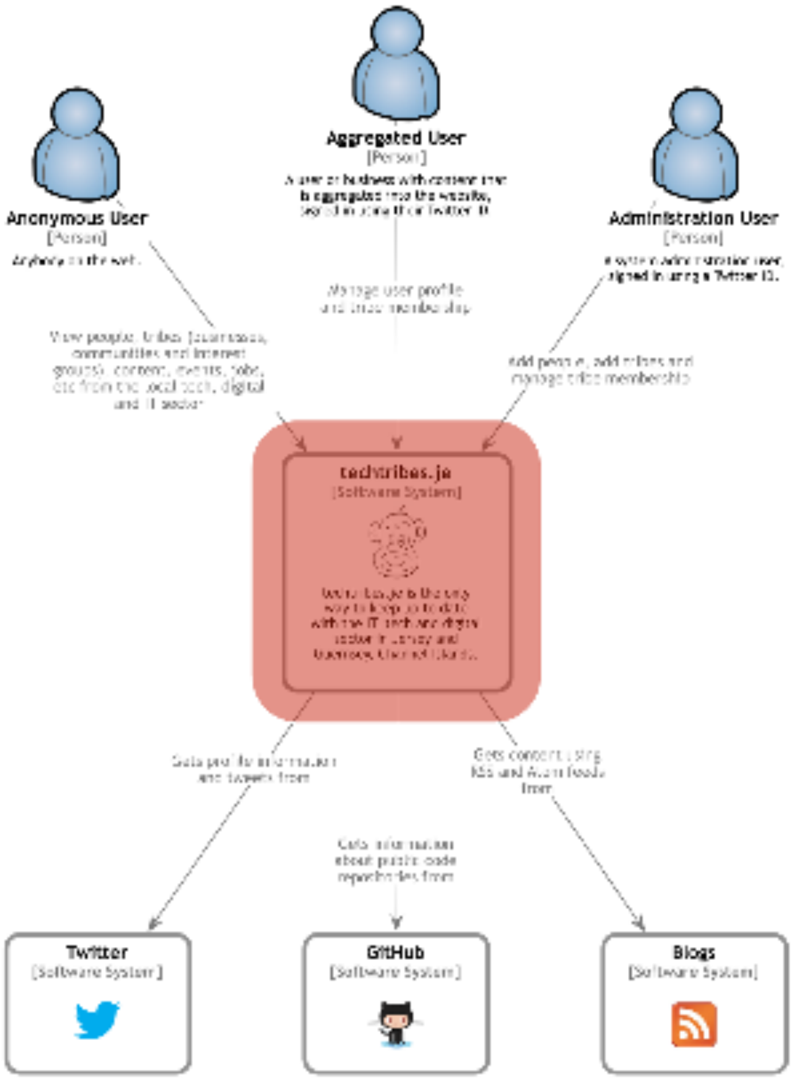

Relational Database
[Container: MySQL 5.5.x]
File System
[Container]
Stores search indexes.
NoSQL Data Store
[Container: MongoDB 2.2.x]
Writes to
GitHub Component
Search Component
News Feed Entry Component
Tweet Component
Scheduled Content Updater
Logging Component
techtribes.je
Content Updater
Uses
Twitter Connector
GitHub Connector
News Feed Connector
Twitter
[Software System]
GitHub
[Software System]
Blogs
[Software System]
techtribes.je - Components - Content Updater
Used by all components.


context
container
component
Class
Attribute
Attribute
Class Name
Attribute
Attribute

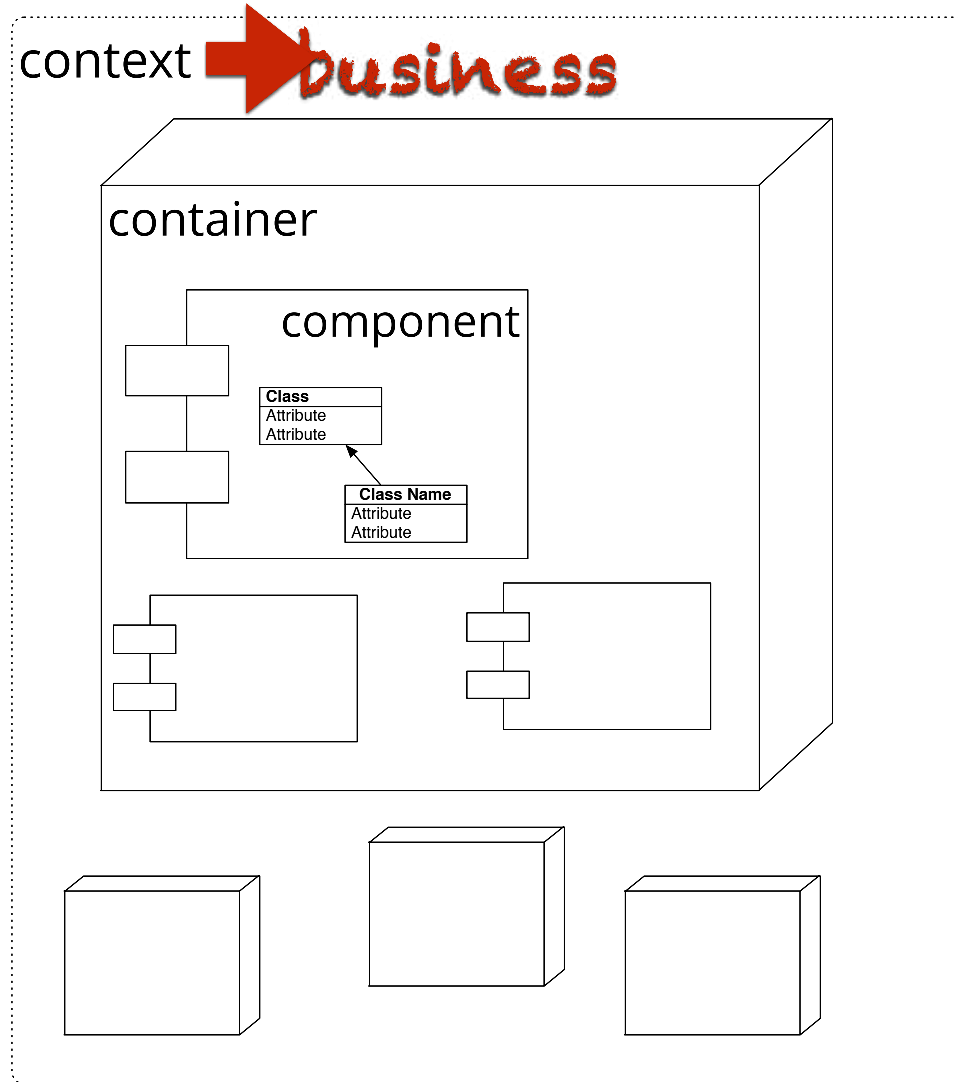

+ TweetComponent
# TweetComponentImpl
+ TweetException
# MongoDbTweetDao


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
container
component
Class
Attribute
Attribute
Class Name
Attribute
Attribute

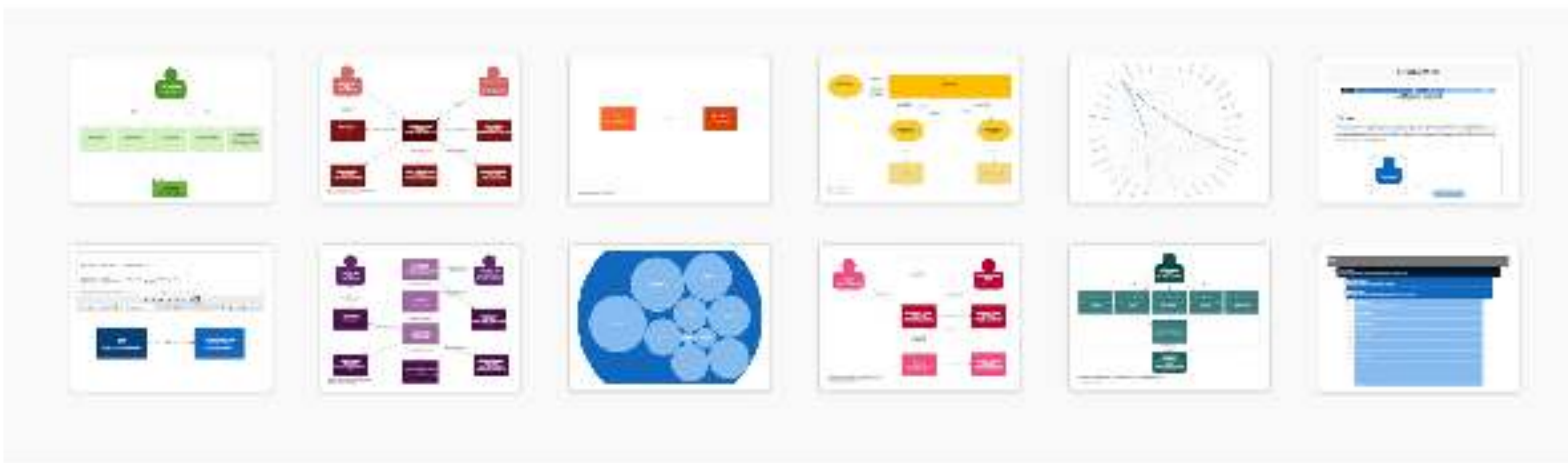


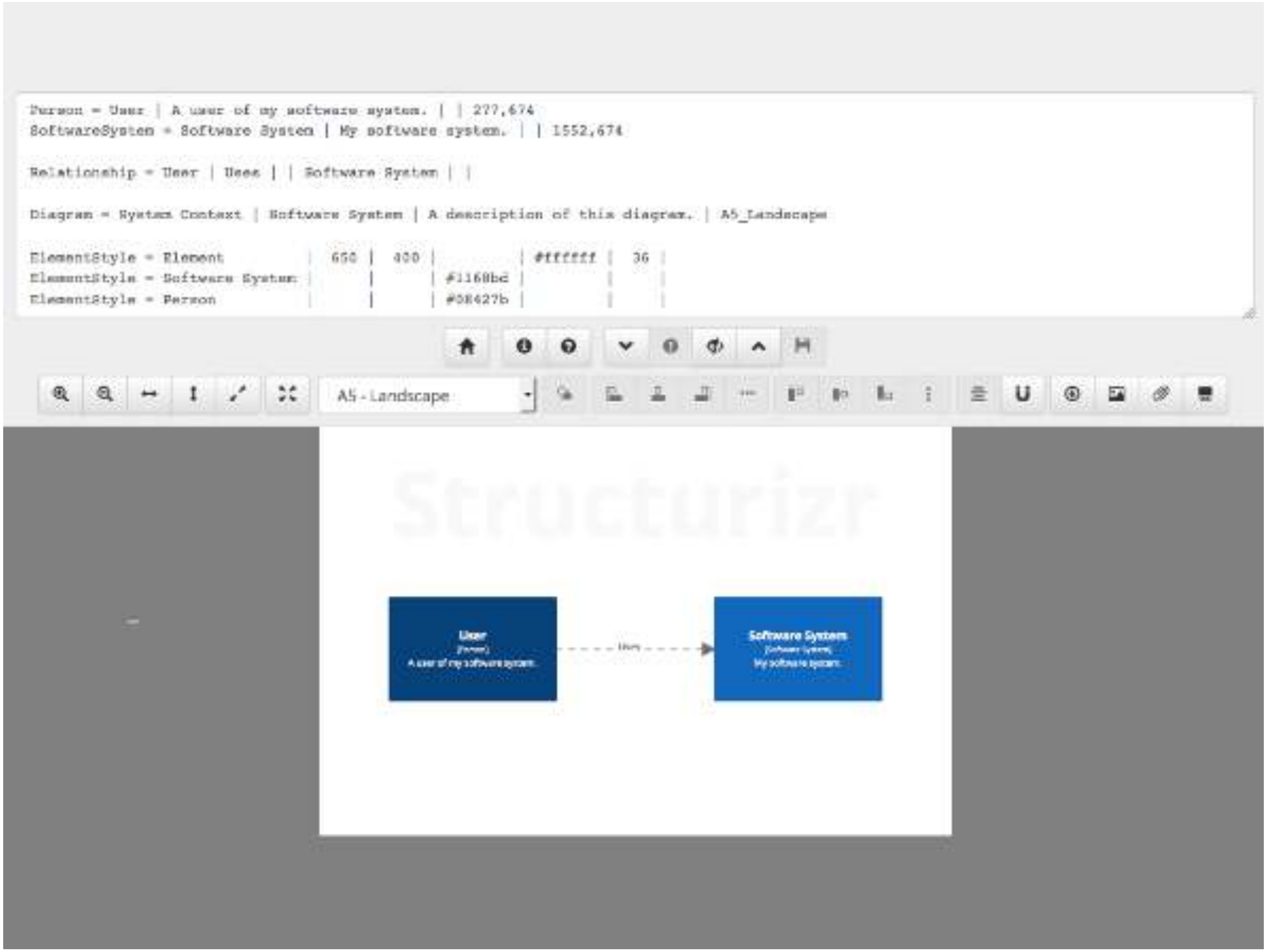
techtribes.je - Context

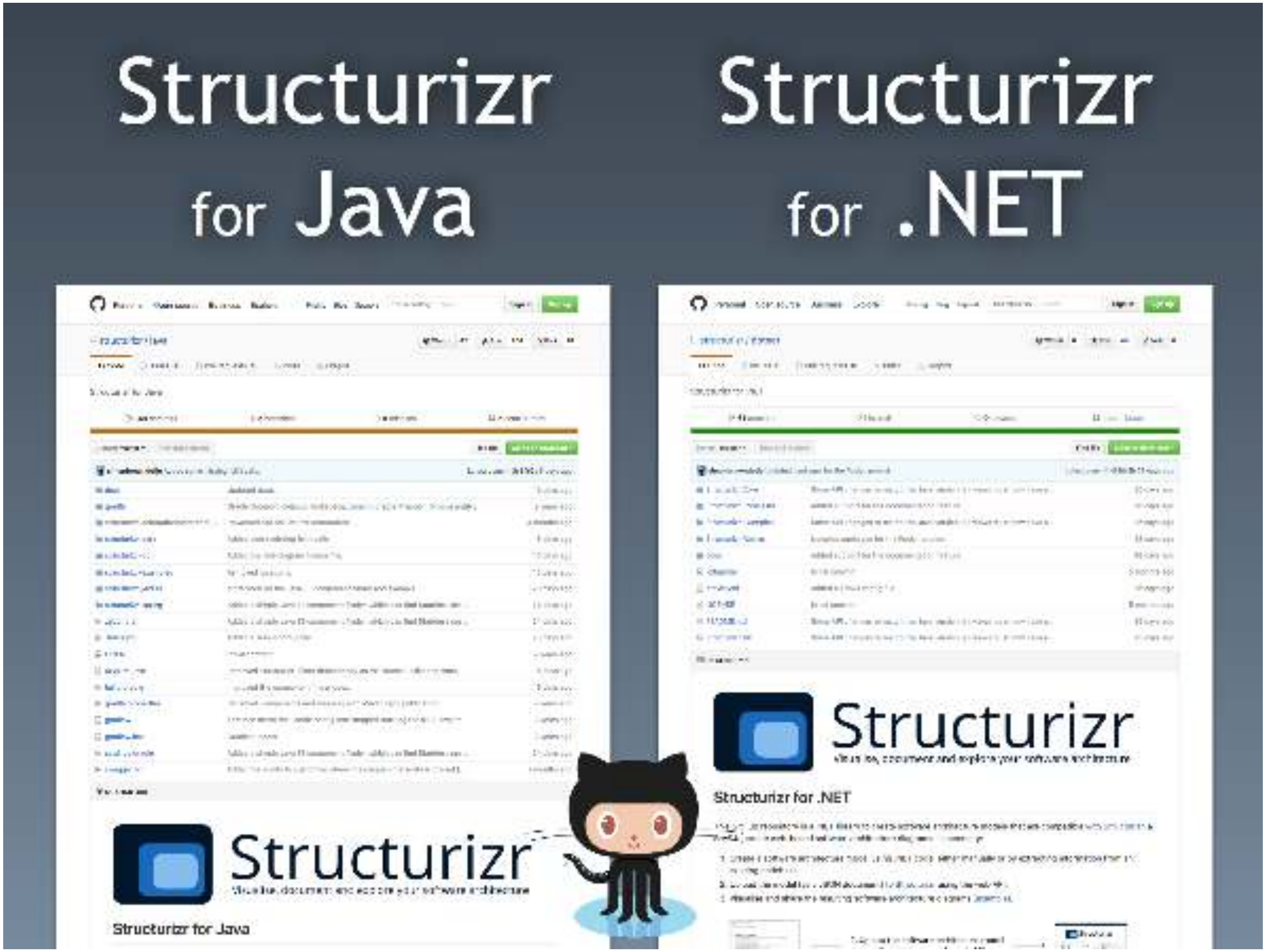


Structurizr

Visualise, document and explore your software architecture

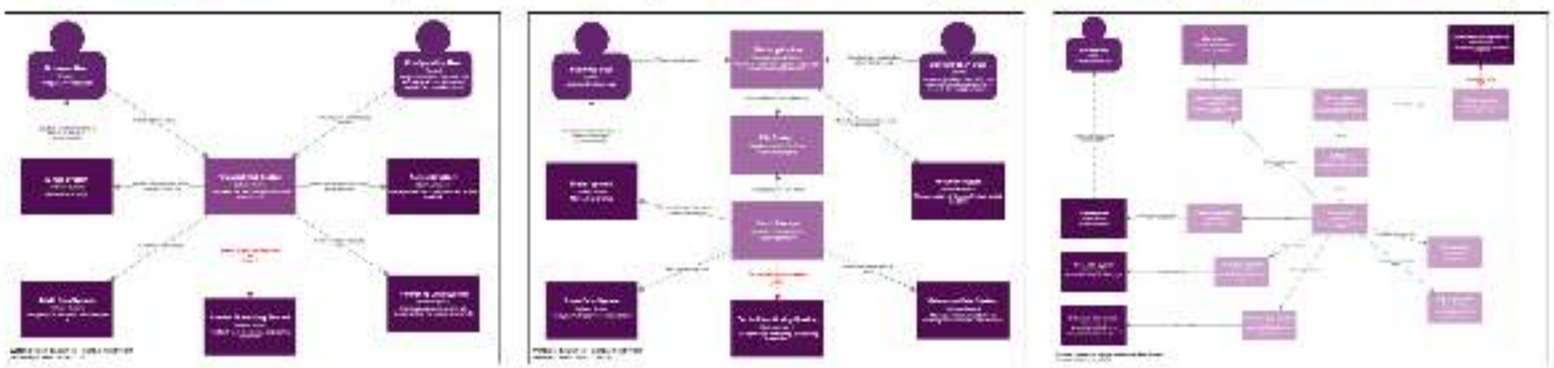


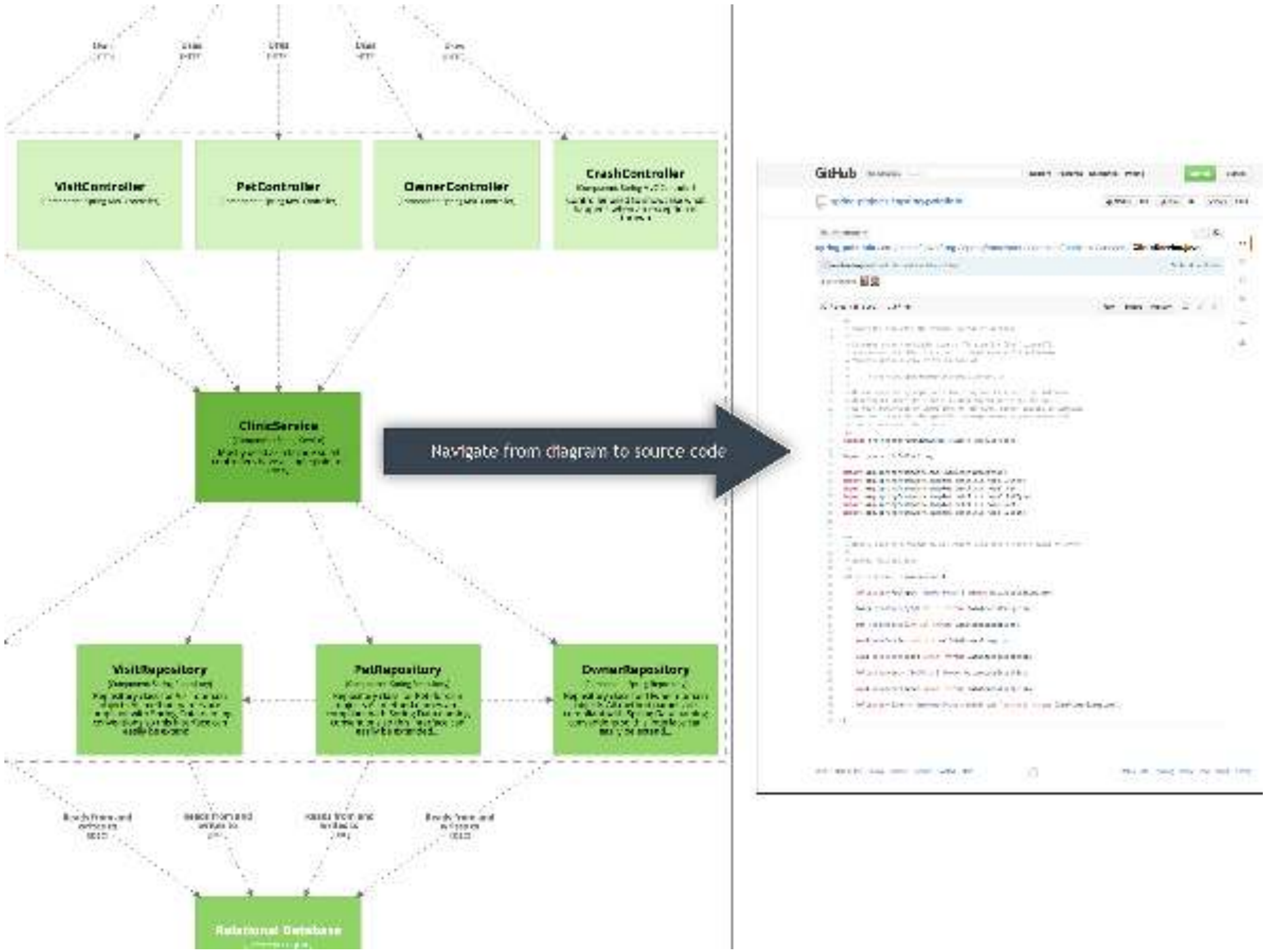

Person = User | A user of my software system. | | 277,674
SoftwareSystem = Software System | My software system. | | 1552,674
Relationship = User | Uses | | Software System | |
Diagram = System Context | Software System | A description of this diagram. | A5_Landscape
ElementStyle = Element | 650 | 400 | | #ffffff | 36 |
ElementStyle = Software System | | | #1168bd | | |
ElementStyle = Person | | | #08427b | | |
A5 - Landscape
Structurizr
User
[Person]
A user of my software system.
Uses
Software System
[Software System]
My software system.


Structurizr for Java
Structurizr for .NET
Structurizr
Structurizr for Java
Structurizr
Structurizr for .NET

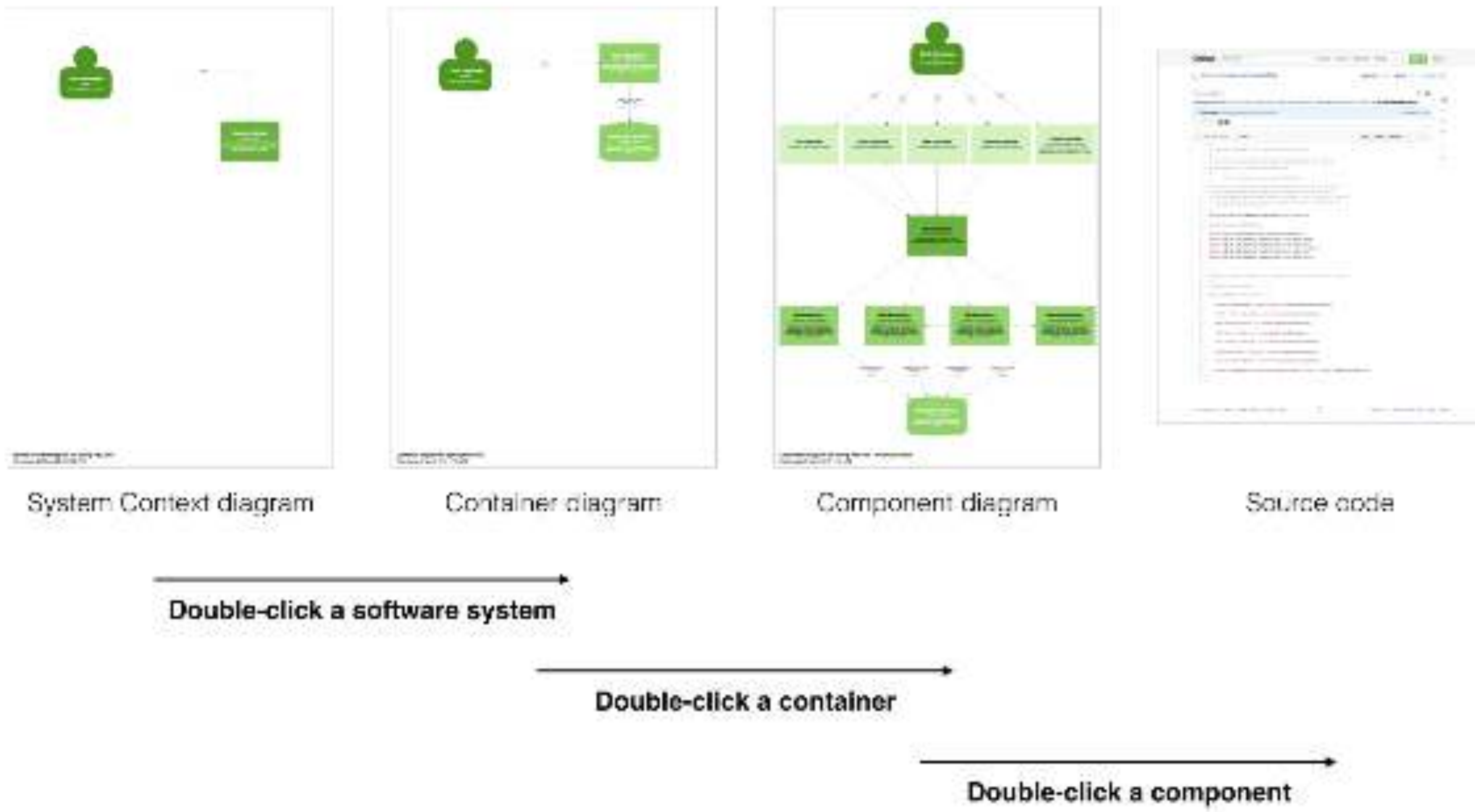
```
Workspace workspace = new Workspace("My model", "This is a model of my software system.");
Model model = workspace.getModel();

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
```



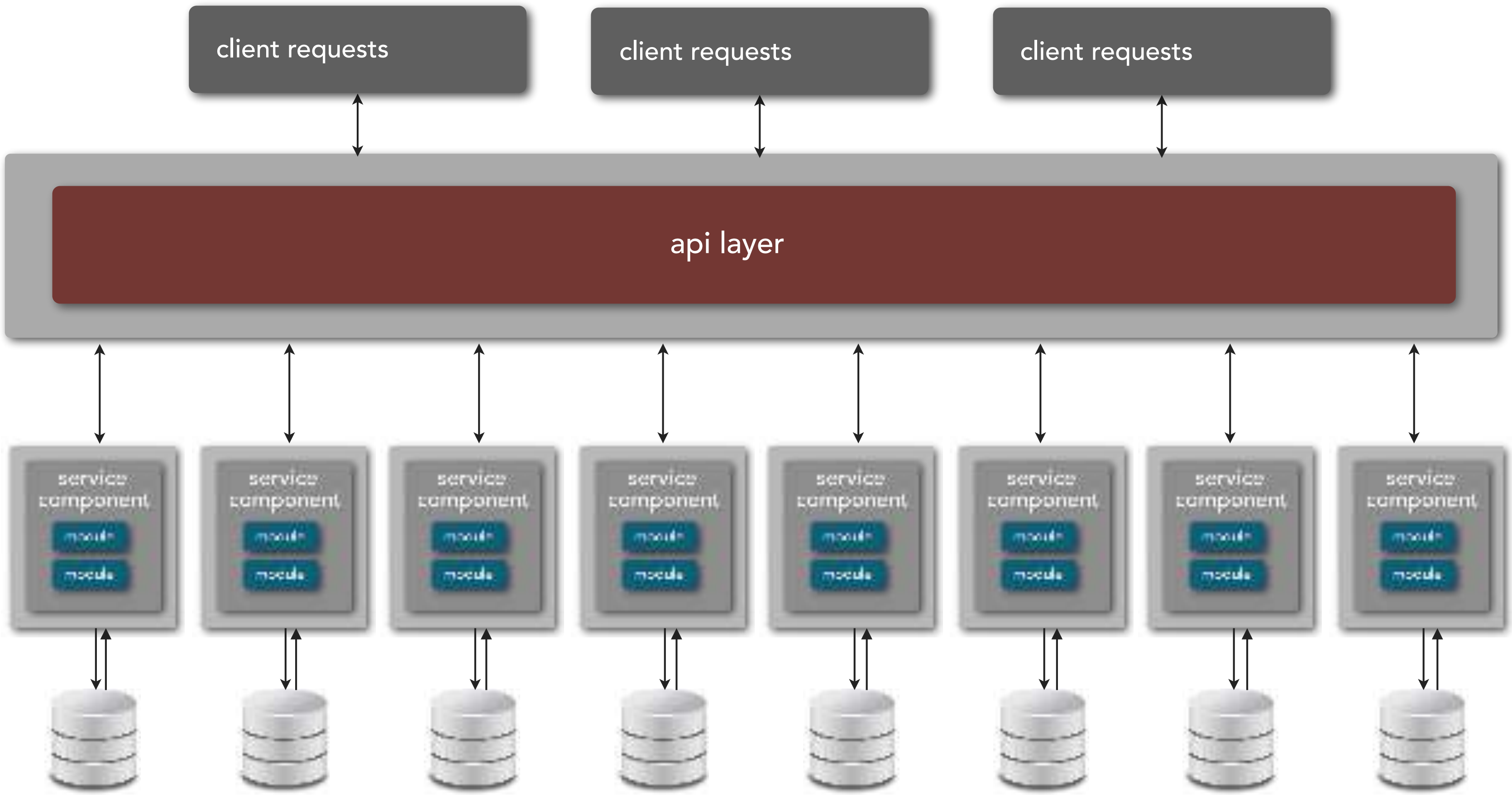


VisitController
PetController
OwnerController
CrashController
ClinicService
Navigate from diagram to source code
VisitRepository
PetRepository
OwnerRepository
Relational Database
GitHub



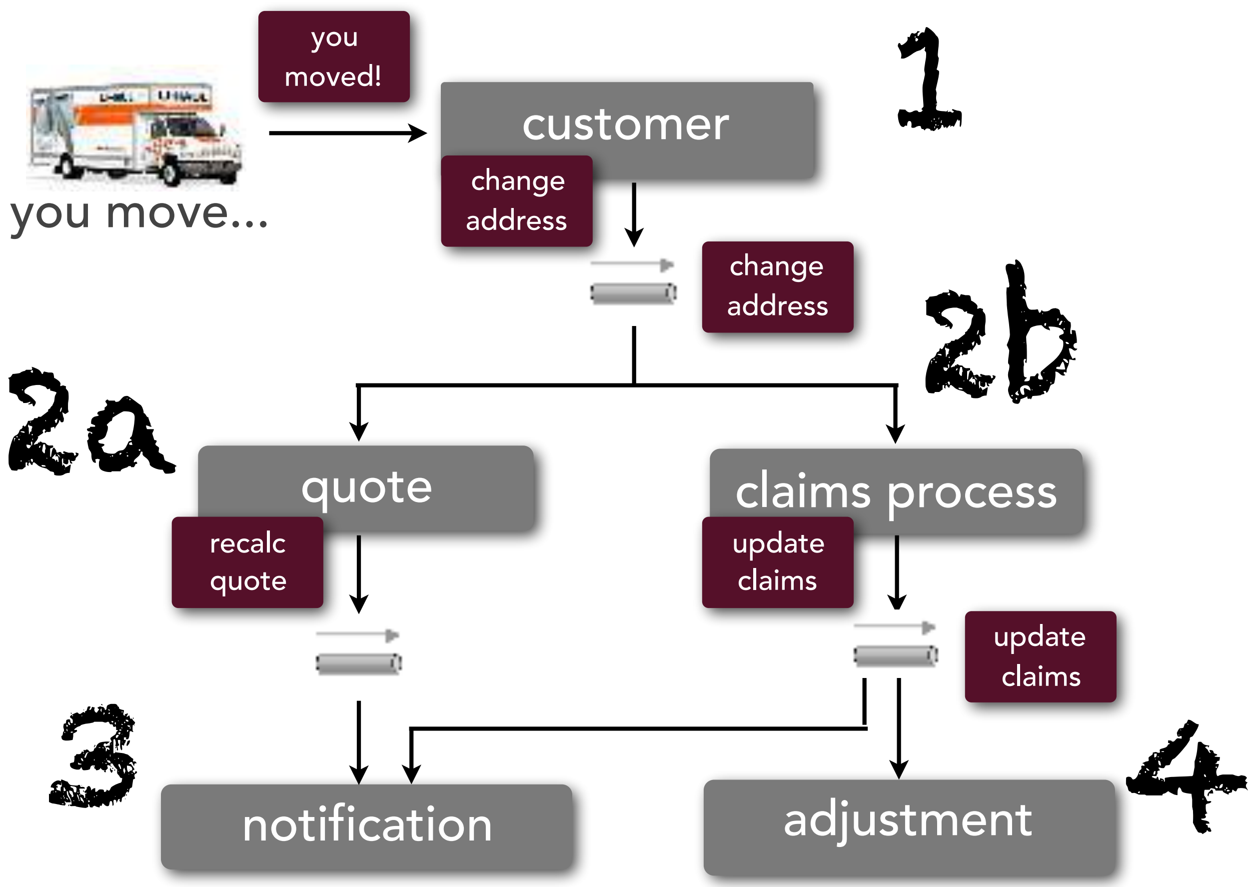
Diagrams are maps

Titles



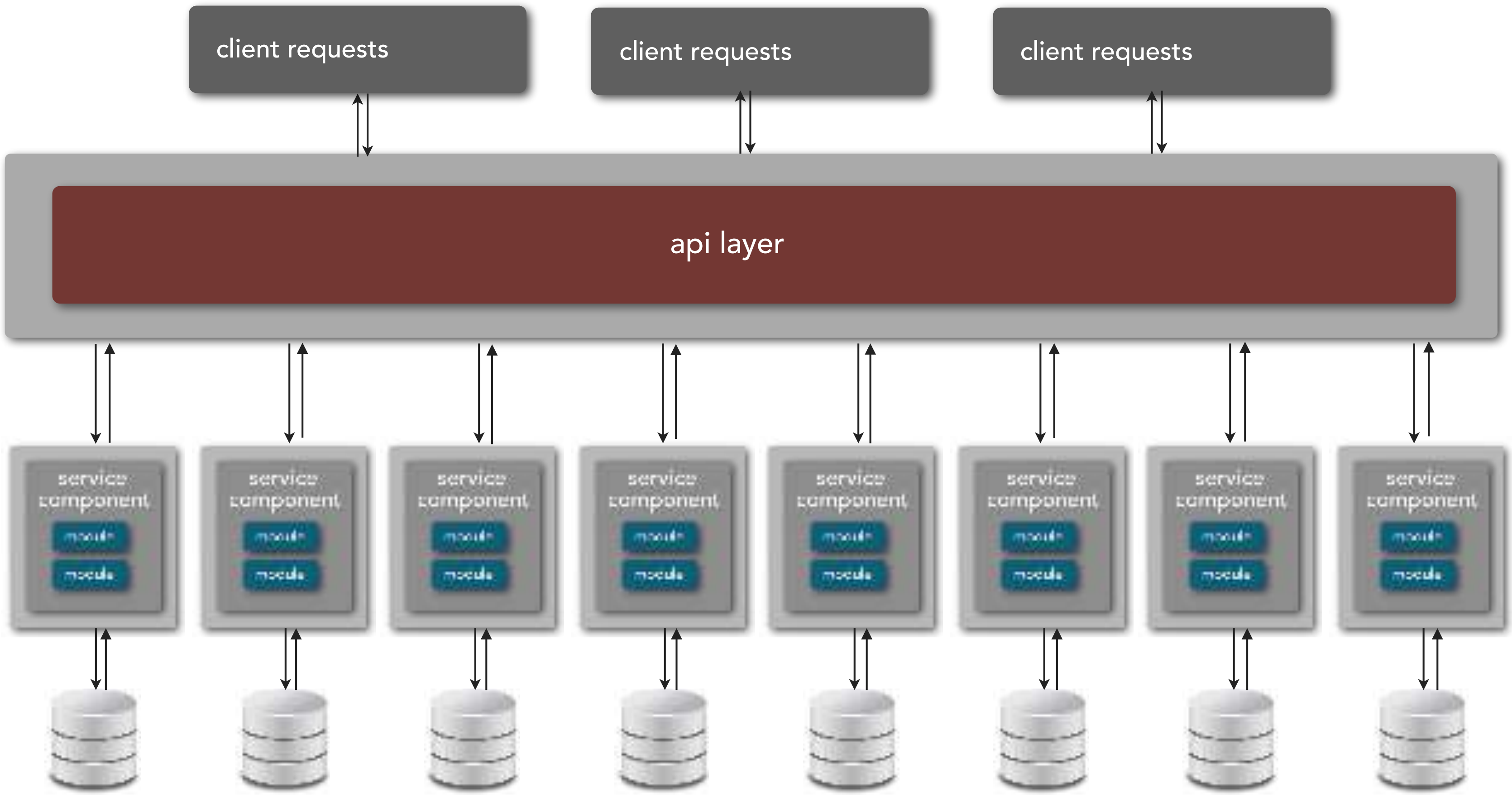
Short and meaningful, numbered if diagram order is

Titles

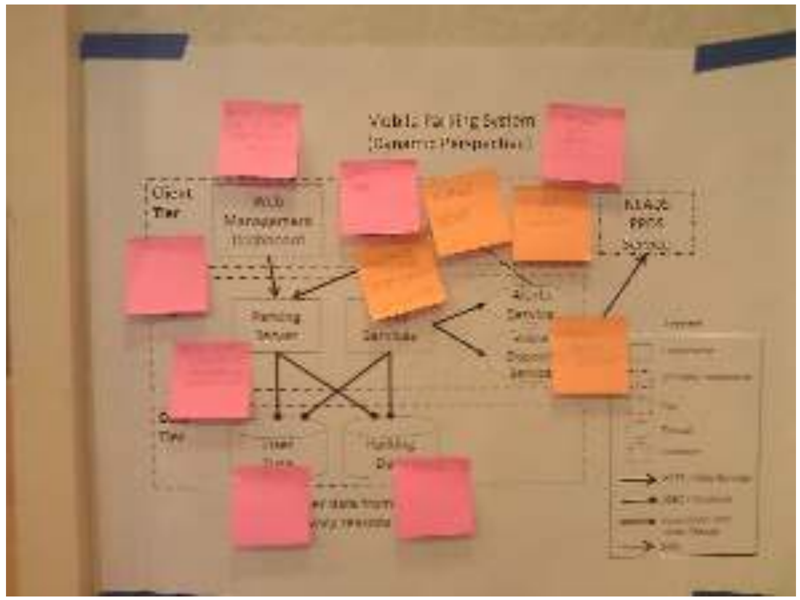


Short and meaningful, numbered if diagram order is

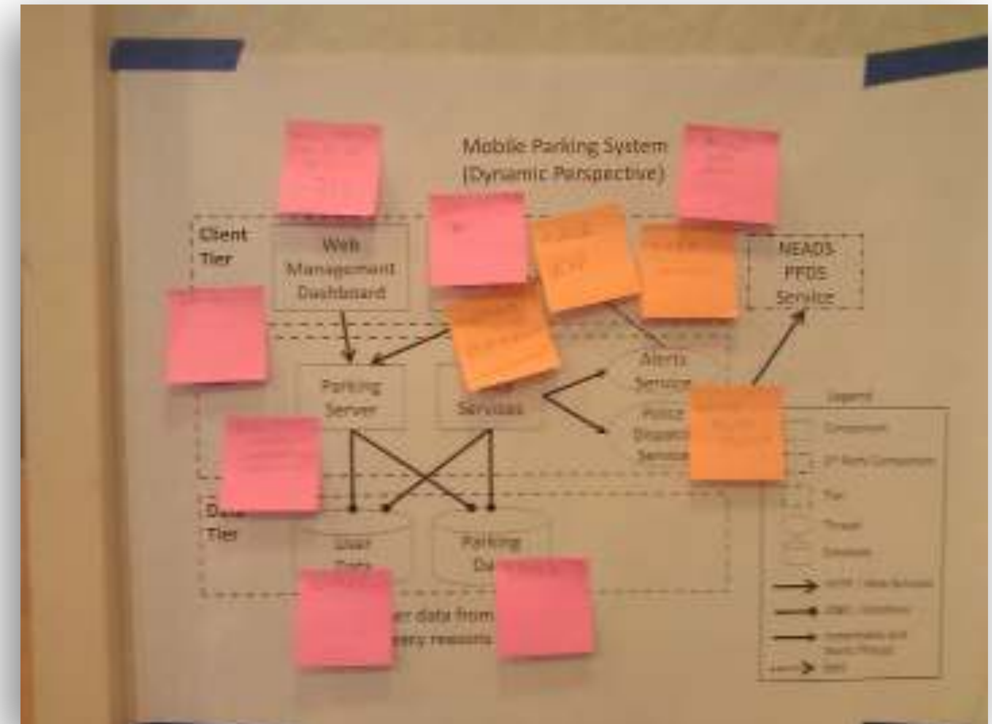
Lines add descriptive text to provide additional inform



Favor unidirectional arrows


Mobile Parking System
(Generic Perspective)
Client
Tier
Web
Management
Parking
Server
Service
Data
Tier
User
Data

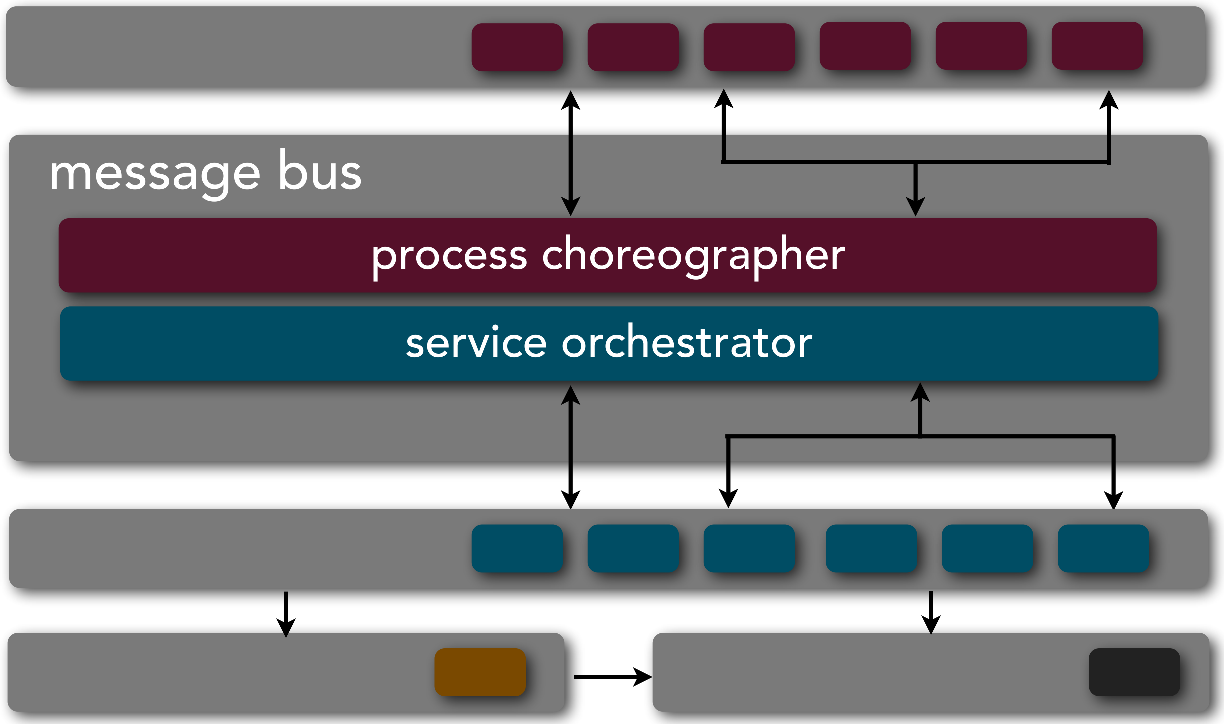
Layout

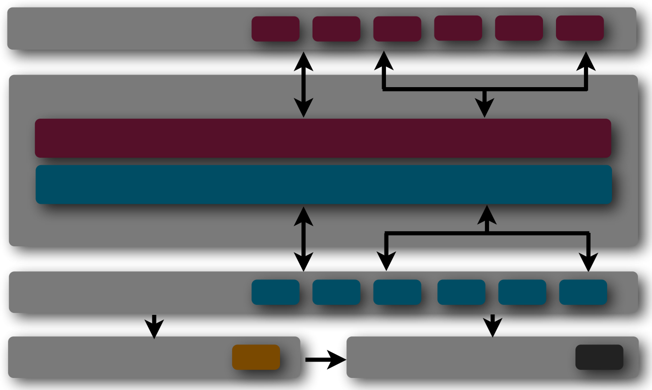


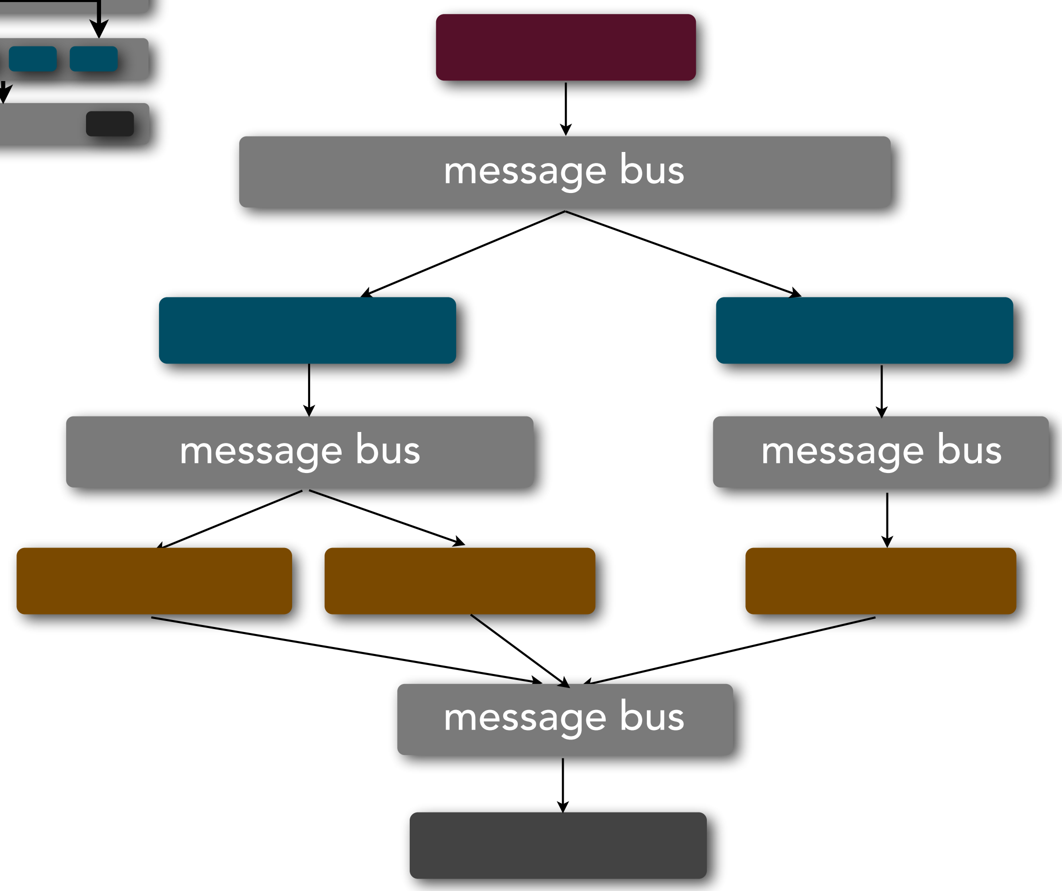
Sticky notes and index cards make a great substitute for drawn

Color

Ensure that color coding is made explicit; watch out for color-


message bus
process choreographer
service orchestrator




message bus
message bus
message bus
message bus

Color

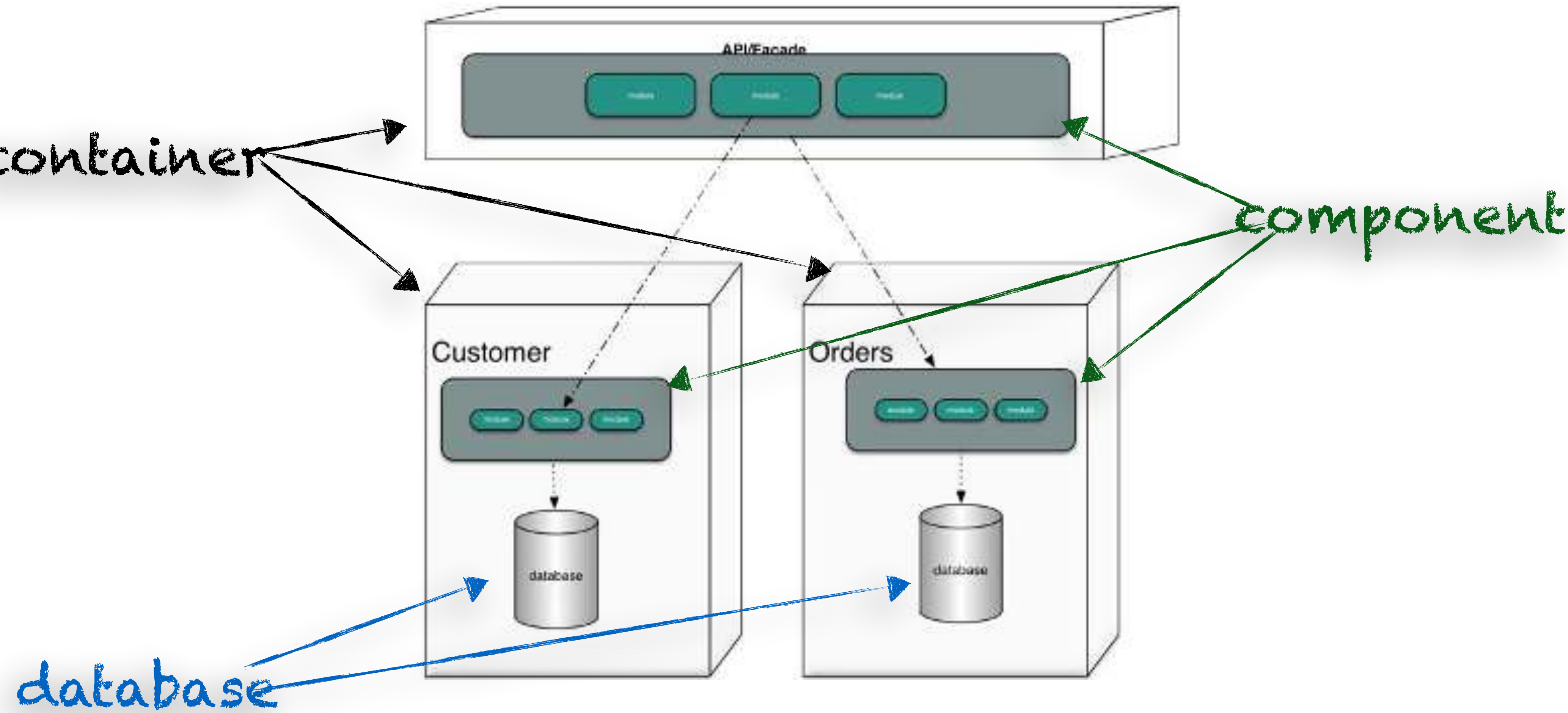
Ensure that color coding is made explicit; watch out for color-

Orientation

Most important thing in the middle;

Shapes

Don’t assume that people will understand what different shapes

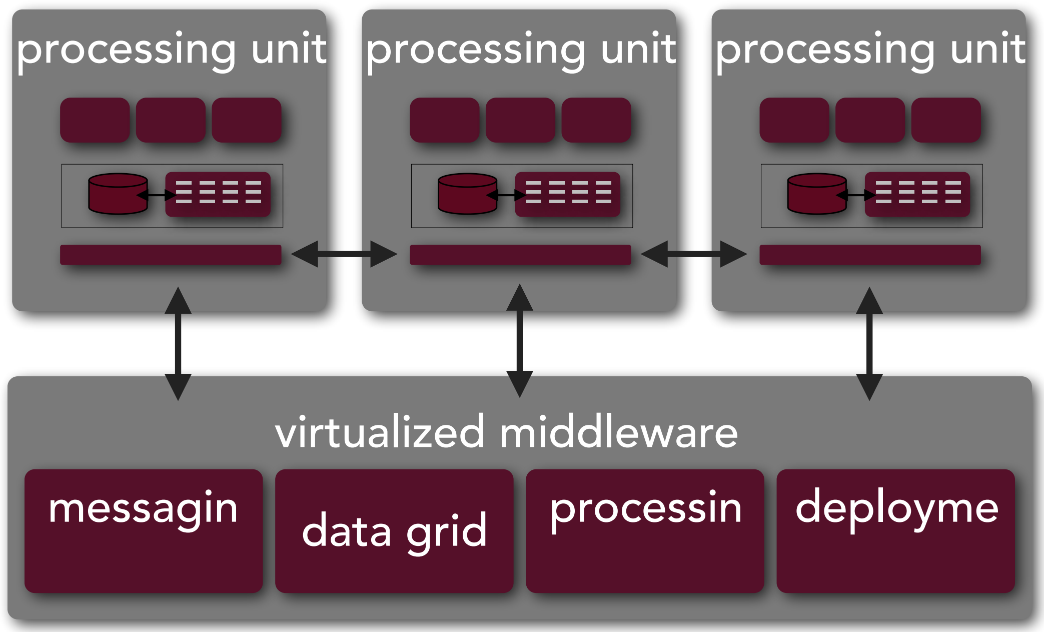

container
API/Facade
component
Customer
Orders
database
database
database

Keys

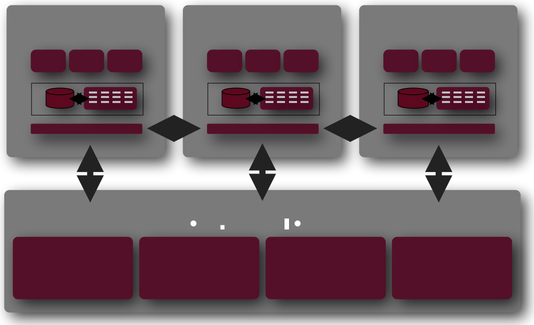
Explain shapes, lines, colors, borders, acronyms, etc

Representational Consistency

Don't abruptly change scale on diagrams; provide context for



processing unit
processing unit
processing unit
virtualized middleware
messagin
data grid
processin
deployme

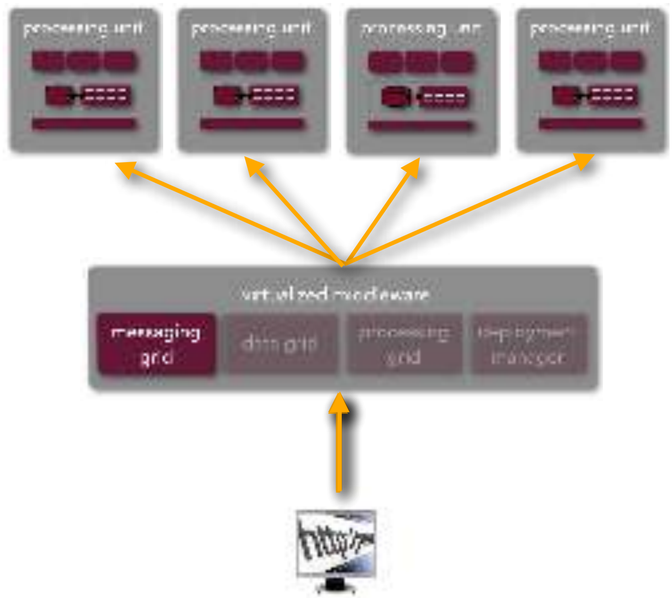


messagin

data grid

processin

deployme

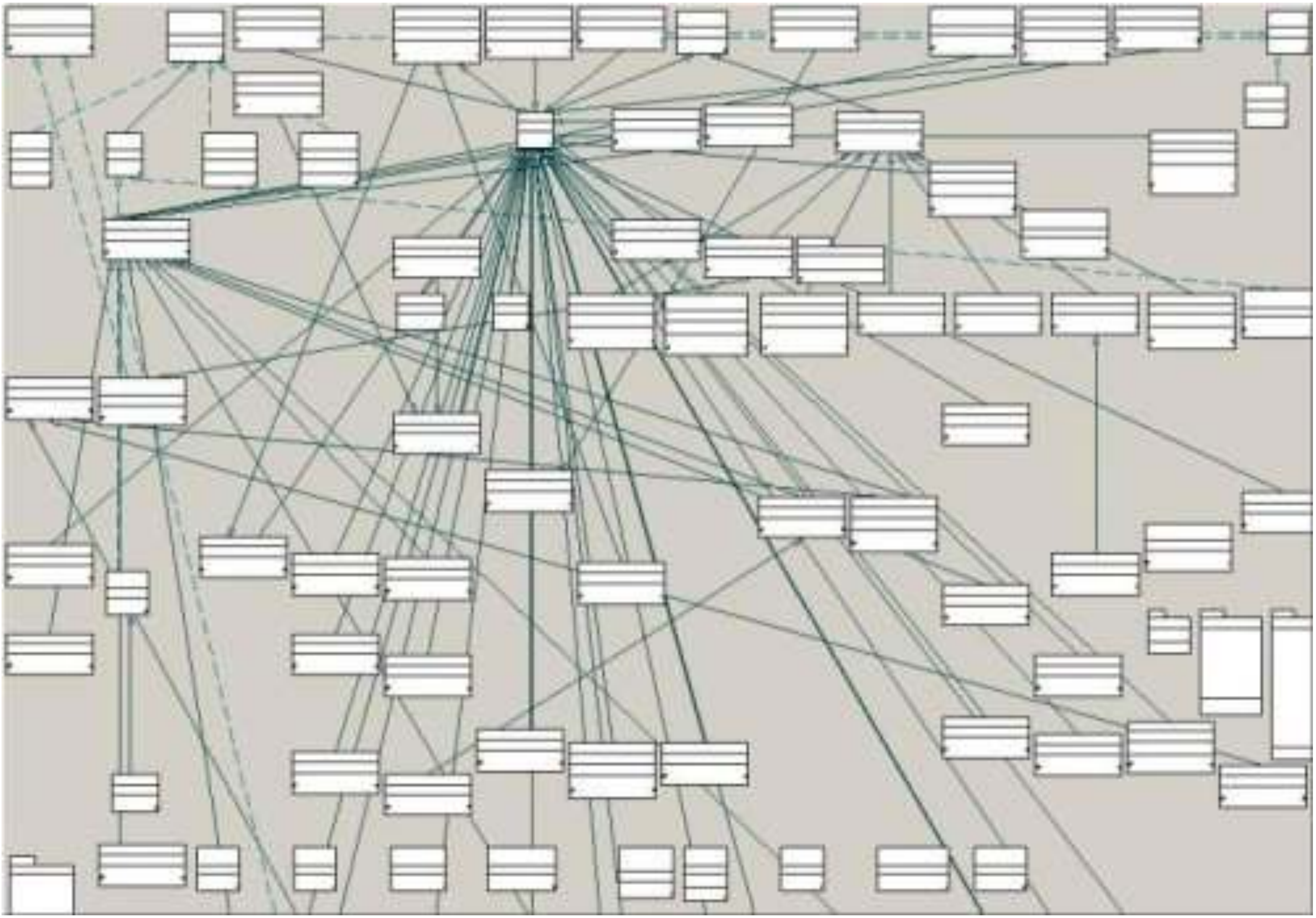


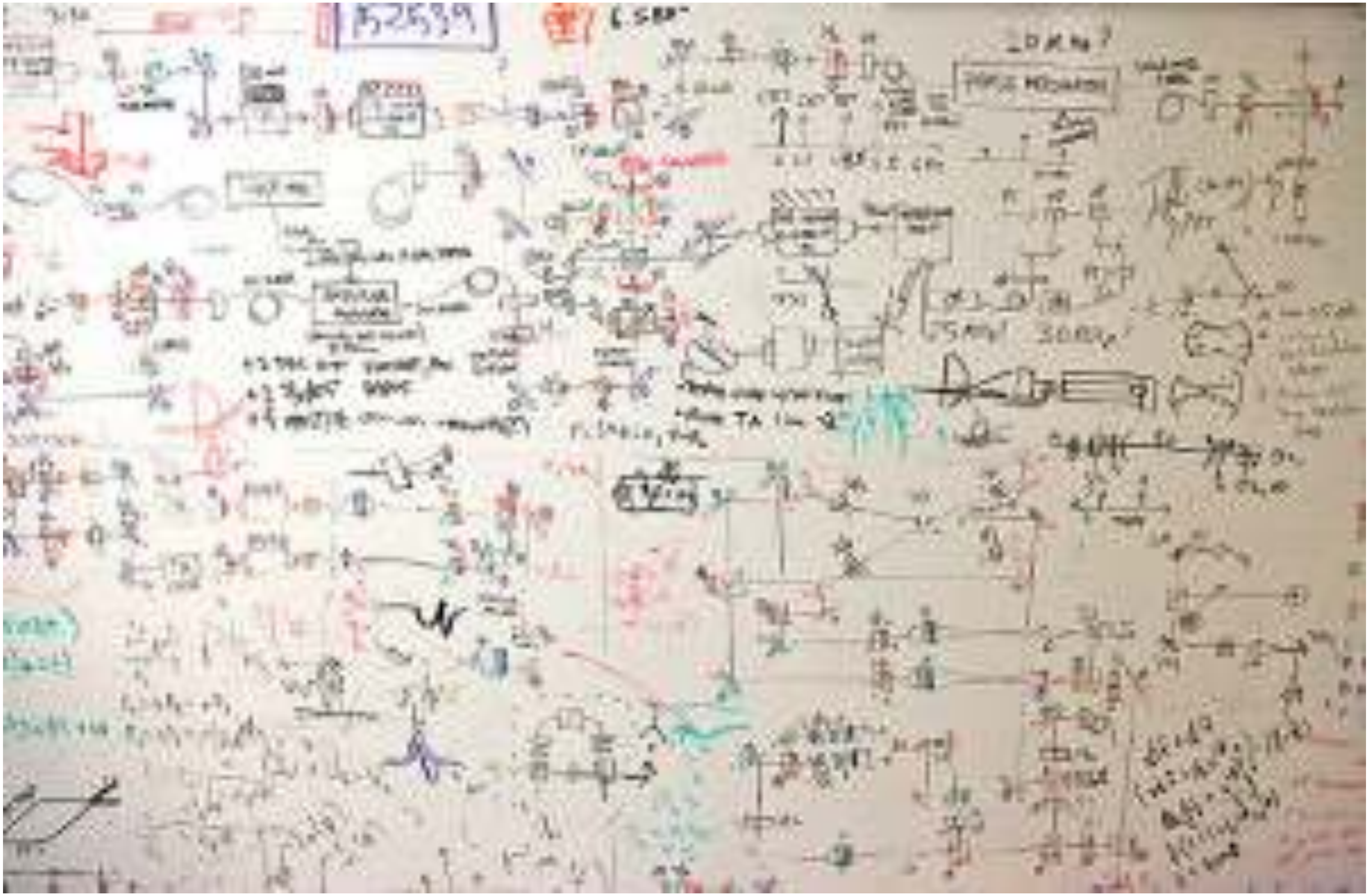
Representational Consistency

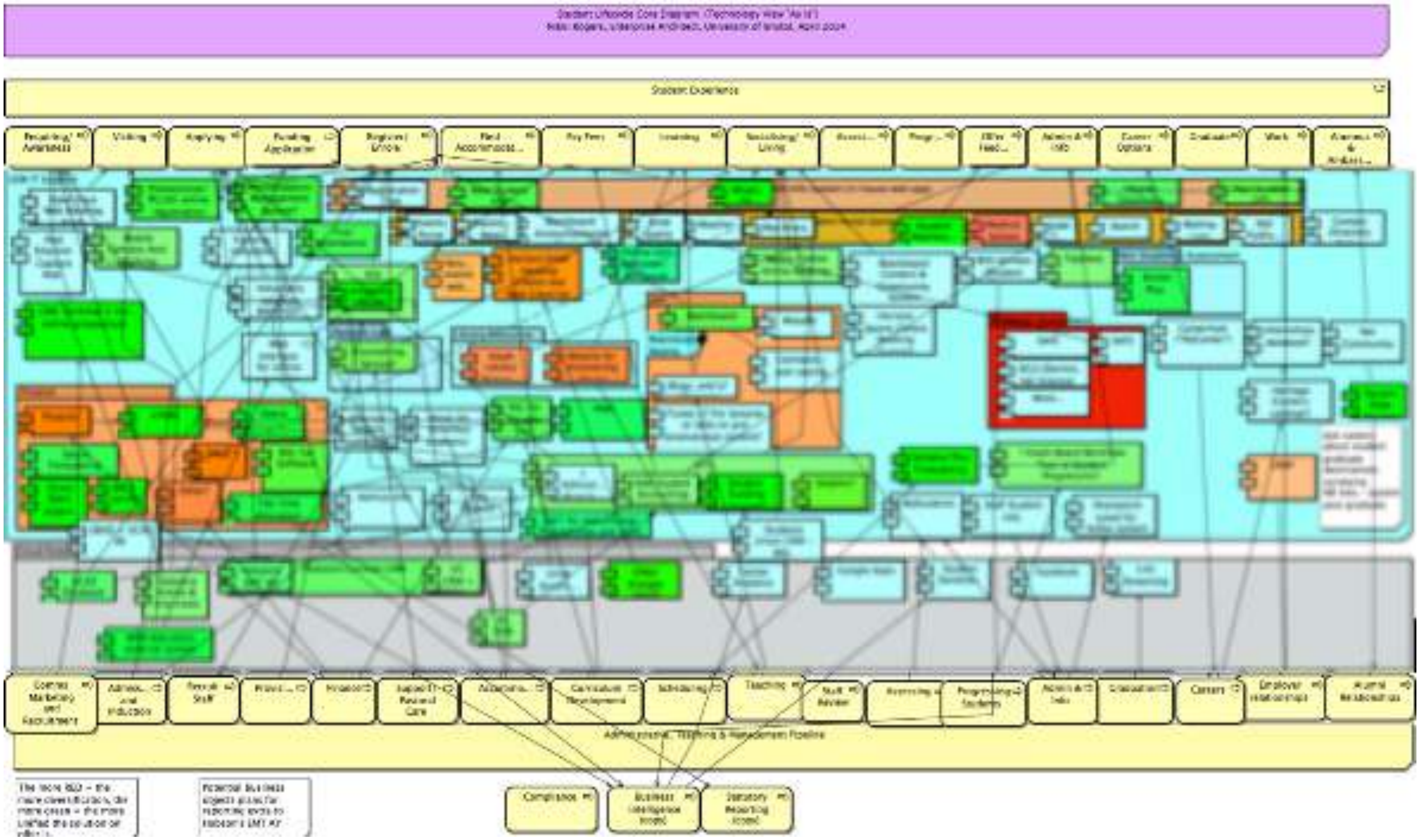
Don’t abruptly change scale on diagrams; provide context for

Comprehensive Diagram

Don’t try to capture the entirety of software architecture in a single



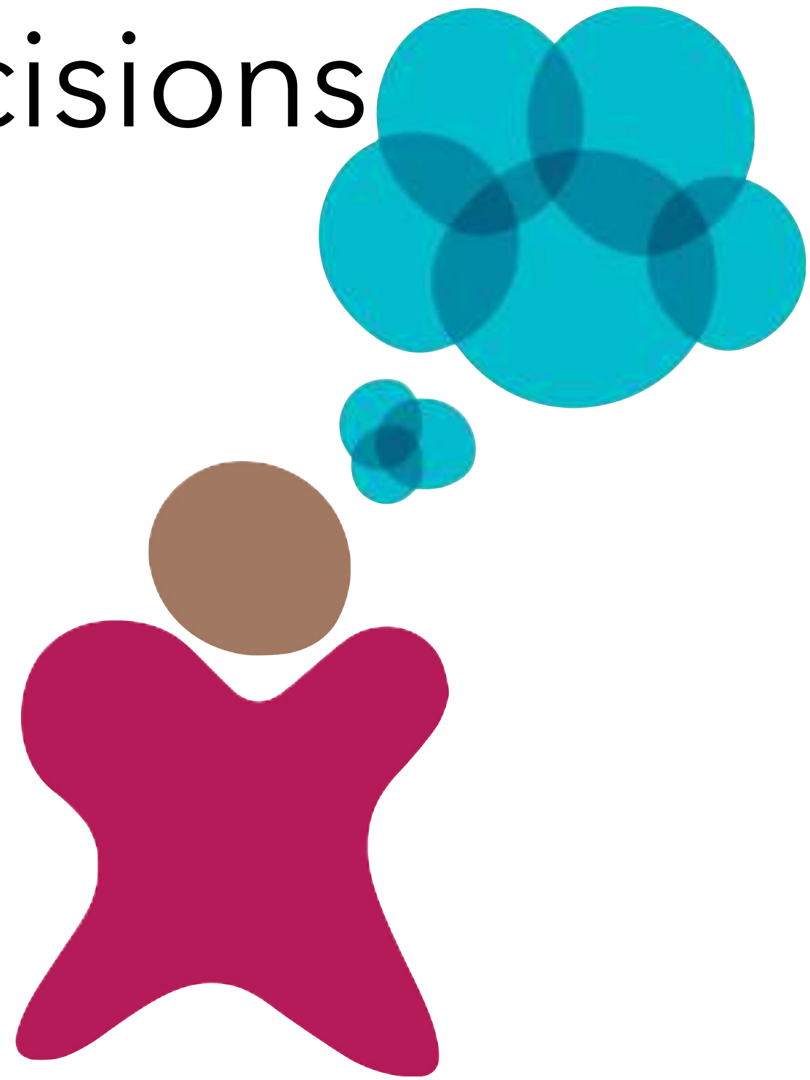




Comprehensive Diagram

Don't try to capture the entirety of software architecture in a single

Decisions



Architecture Decision

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

Textile

asciidoc

short text file

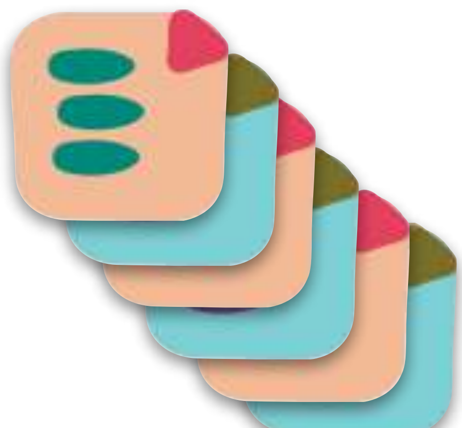
`doc/arch/adr-NNN.md`

semantic

numbered

no number

reversed ADR kept—marked superseded



Architecture Decision

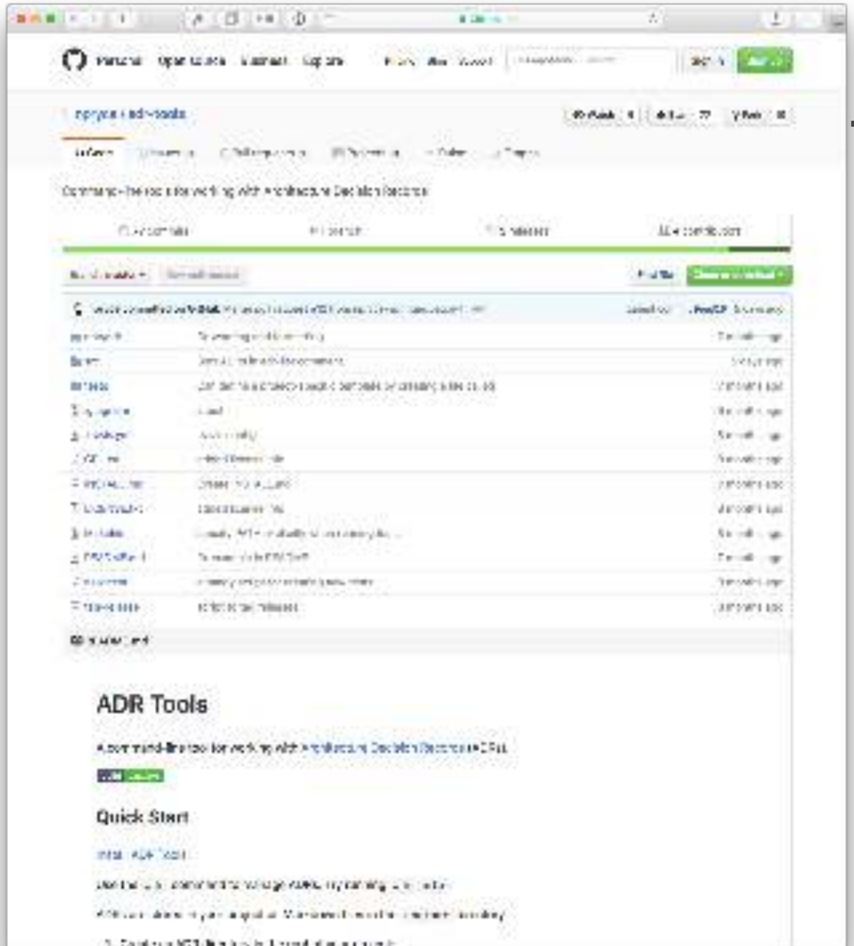
Title: short noun phrase

Context: forces at play

Decision: response to forces

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

ADR Tool



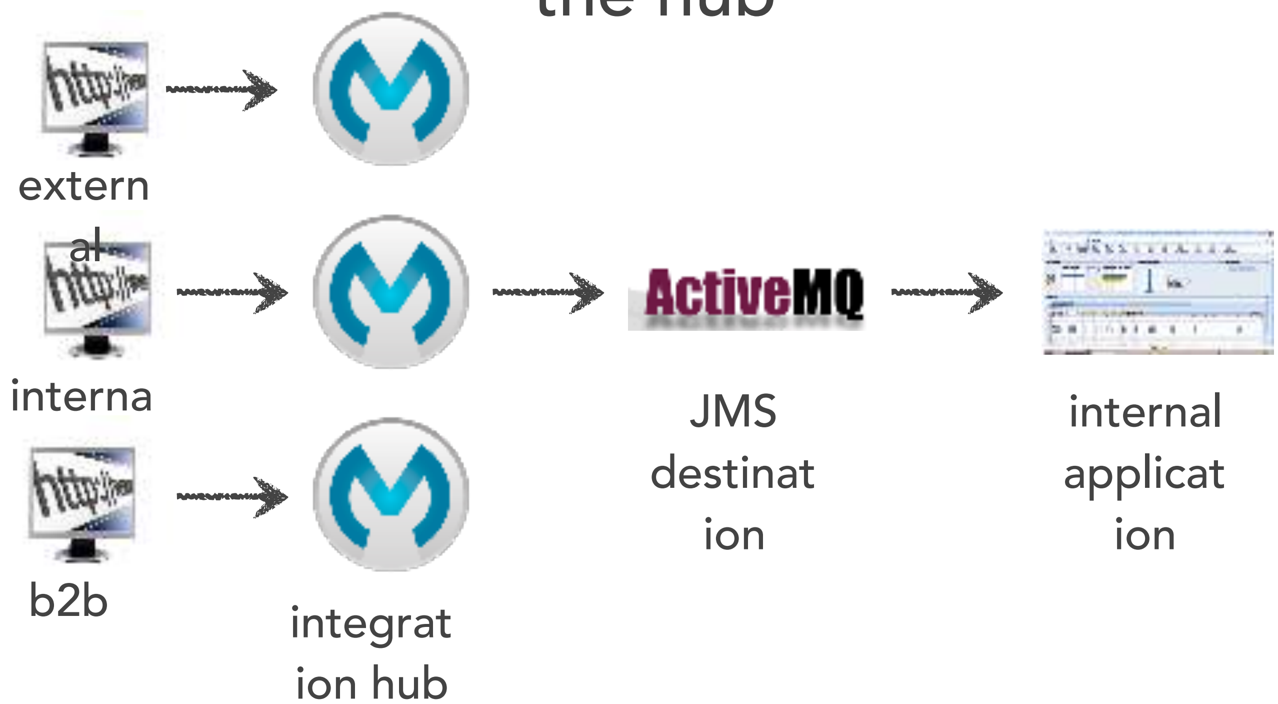
https://github.com/npryce/adr-tools

Justifying Decisions

the scenario

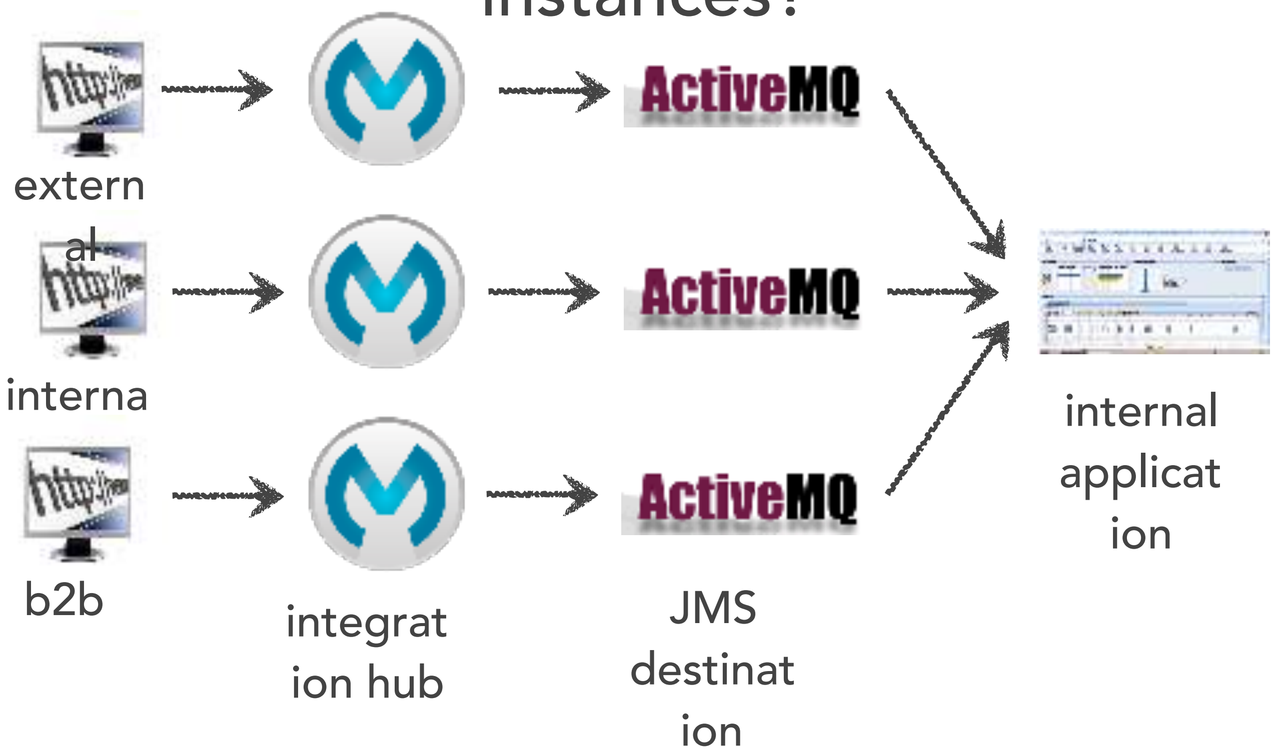
Justifying Decisions

the requirement: you need to federate the hub



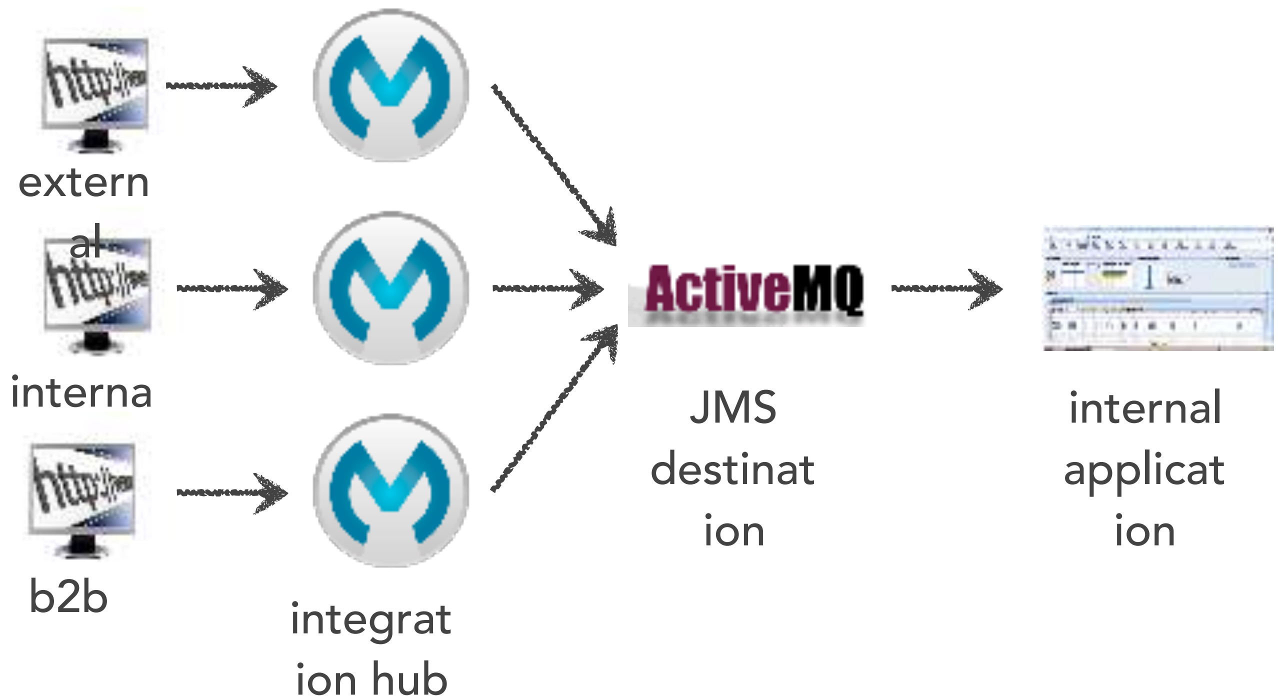
Justifying Decisions

the decision: dedicated broker instances?

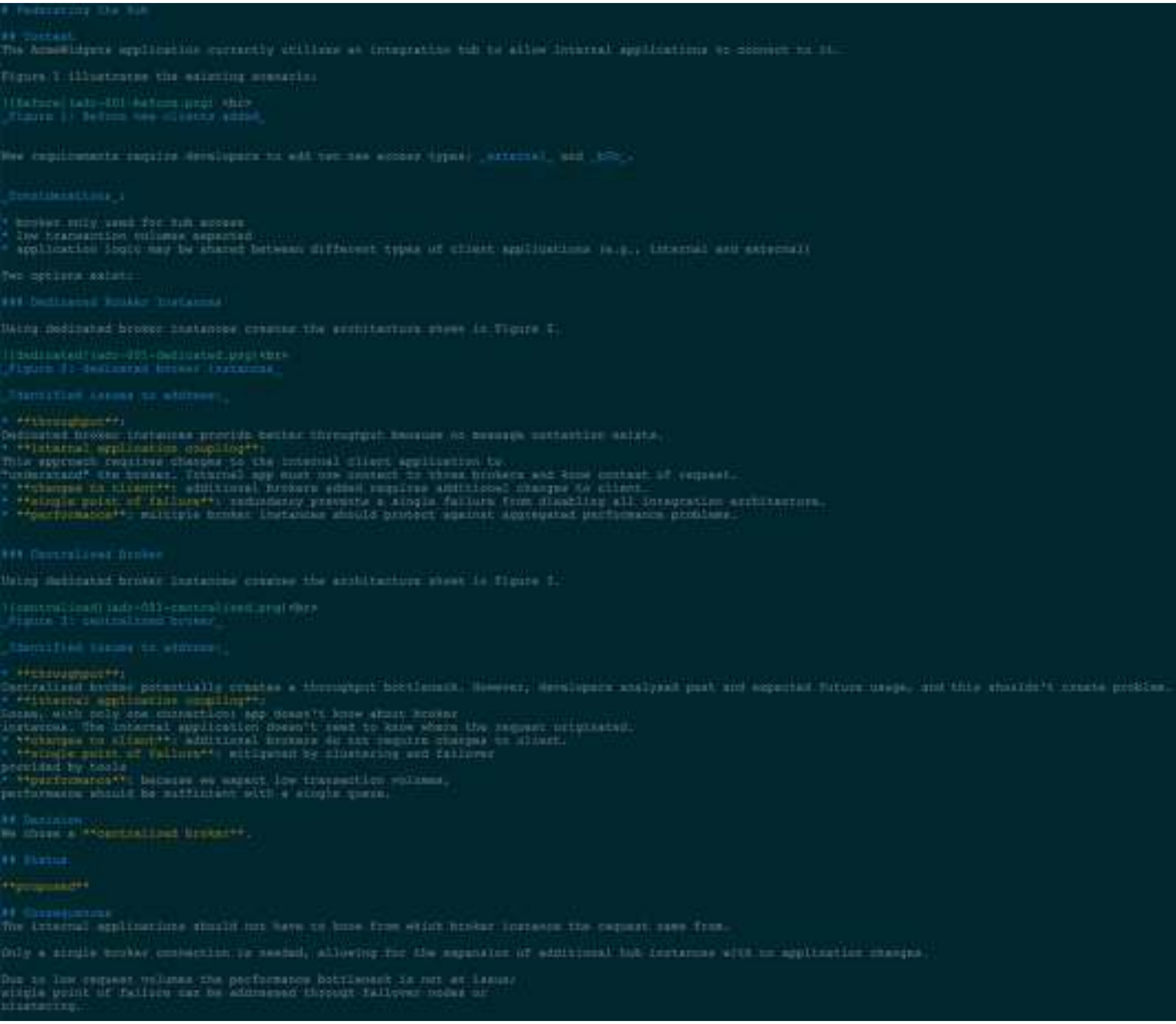


Justifying Decisions

the decision: centralized broker



dr-001.md



```
# Federating the hub

## Context
The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:

![Before](adr-001-before.png) <br>
_Figure 1: Before new clients added_


New requirements require developers to add two new access types: _external_ and _B2B_.


_Considerations_:

* 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)

Two options exist:

### Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

![dedicated](adr-001-dedicated.png)<br>
_Figure 2: Dedicated broker instances_

_Identified issues to address:_

* **throughput**:
Dedicated broker instances provide better throughput because no message contention exists.
* **internal application coupling**:
This approach requires changes to the internal client application to
"understand" the broker. Internal app must now connect to three brokers and know context of request.
* **changes to client**: additional brokers added requires additional changes to client.
* **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
* **performance**: multiple broker instances should protect against aggregated performance problems.


### Centralized Broker

Using dedicated broker instances creates the architecture shown in Figure 3.

![centralized](adr-001-centralized.png)<br>
_Figure 3: centralized broker_

_Identified issues to address:_

* **throughput**:
Centralized broker potentially creates a throughput bottleneck. However, developers analyzed past and expected future usage, and this shouldn't create problems.
* **internal application coupling**:
Issues, with only one connection: app doesn't know about broker
instances. The internal application doesn't need to know where the request originated.
* **changes to client**: additional brokers do not require changes to client.
* **single point of failure**: mitigated by clustering and failover
provided by tools
* **performance**: because we expect low transaction volumes,
performance should be sufficient with a single queue.

## Decision
We chose a **centralized broker**.

## Status

**proposed**

## Consequences
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.
```

```
# Federating the Hub

## Context
The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:

![Before](adr-001-before.png) <br>
_Figure 1: Before new clients added_


New requirements require developers to add two new access types: _external_ and _b2b_.


_Considerations_:

* 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)

Two options exist:

### Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

![dedicated](adr-001-dedicated.png)<br>
_Figure 2: dedicated broker instances_

_Identified issues to address:_

* **throughput**:
Dedicated broker instances provide better throughput because no message contention exists.
* **internal application coupling**:
This approach requires changes to the internal client application to
"understand" the broker. Internal app must now connect to three brokers and know context of request.
* **changes to client**: additional brokers added requires additional changes to client.
* **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
* **performance**: multiple broker instances should protect against aggregated performance problems.


### Centralized Broker

Using dedicated broker instances creates the architecture shown in Figure 3.
```

Federating the Hub

Context

The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:

Figure 1: Before new clients added

New requirements require developers to add two new access types: *external* and *b2b*

Considerations:

- 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)

Two options exist:

Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

Figure 2: dedicated broker instances

Identified issues to address:

- **throughput:** Dedicated broker instances provide better throughput because no message contention exists.
- **internal application coupling:** This approach requires changes to the internal client application to "understand" the broker. Internal app must now connect to three brokers and know context of request.
- **changes in client:** additional brokers added requires additional changes in client.
- **single point of failure:** redundancy prevents a single failure from disabling all integration architecture.
- **performance:** multiple broker instances should protect against aggregated performance problems.

Centralized Broker

Using dedicated broker instance creates the architecture shown in Figure 3.

Figure 3: centralized broker

Identified issues to address:

- **throughput:** Centralized broker potentially creates a throughput bottleneck. However, developers analyzed past and expected future usage, and this shouldn't create problem.
- **internal application coupling:** Loose, with only one connection, app doesn't know about broker instances. The internal application doesn't need to know where the request originated.
- **changes to client:** additional brokers do not require changes to client.
- **single point of failure:** mitigated by clustering and failover provided by tools
- **performance:** because we expect low transaction volumes, performance should be sufficient with a single queue.

Decision

We chose a **centralized broker.**

Status

proposed

Consequences

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.

The Case for

http://asciidoc.org

http://asciidoctor.org

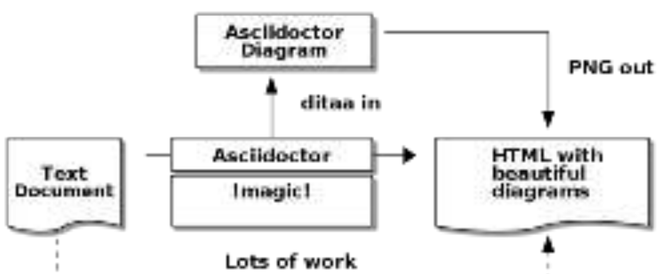
The Case for

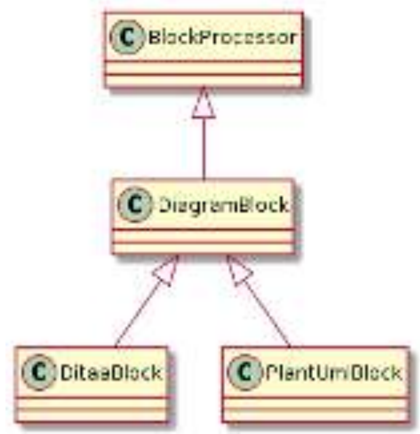
http://asciidoctor.org/docs/asciidoctor-diagram/

The Case for

```
[ditaa]
....
                 +-------------+
                 | Asciidoctor |-------+
                 |   diagram   |       |
                 +-------------+       | PNG out
                     ^                 |
                     | ditaa in        |
                     |                 v
 +--------+   +-------------+    /---------------\
 |        | --+ Asciidoctor +--> |               |
 |  Text  |   +-------------+    |   Beautiful   |
 |Document|   |   !magic!   |    |    Output     |
 |     {d}|   |             |    |               |
 +---+----+   +-------------+    \---------------/
     :                                   ^
     |          Lots of work             |
     +-----------------------------------+
....
```

```
[plantuml, diagram-classes, png]
....
class BlockProcessor
class DiagramBlock
class DitaaBlock
class PlantUmlBlock

BlockProcessor <|-- DiagramBlock
DiagramBlock <|-- DitaaBlock
DiagramBlock <|-- PlantUmlBlock
....
```



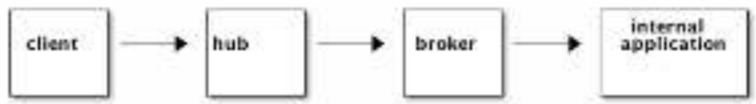


Federating the Hub

Context

The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:



New requirements require developers to add two new access types: *external* and *b2b*.

Considerations:

- 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)

Two options exist:

Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

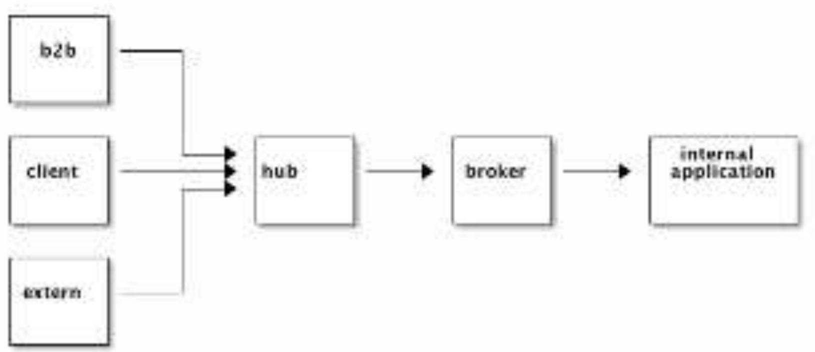


Figure 2: dedicated broker instances

Identified issues to address:

- **throughput**: Dedicated broker instances provide better throughput because no message contention exists.
- **internal application coupling**: This approach requires changes to the internal client application to "understand" the broker. Internal app must now connect to three brokers and know context of request.
- **changes to client**: additional brokers added requires additional changes to client.
- **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
- **performance**: multiple broker instances should protect against aggregated performance problems.

Centralized Broker

Using dedicated broker instances creates the architecture shown in Figure 3.

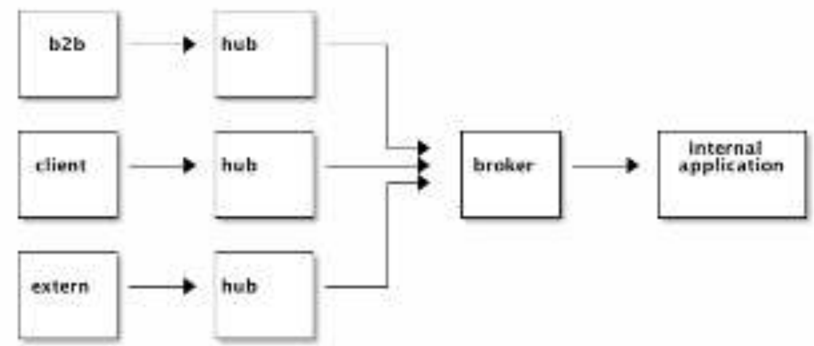


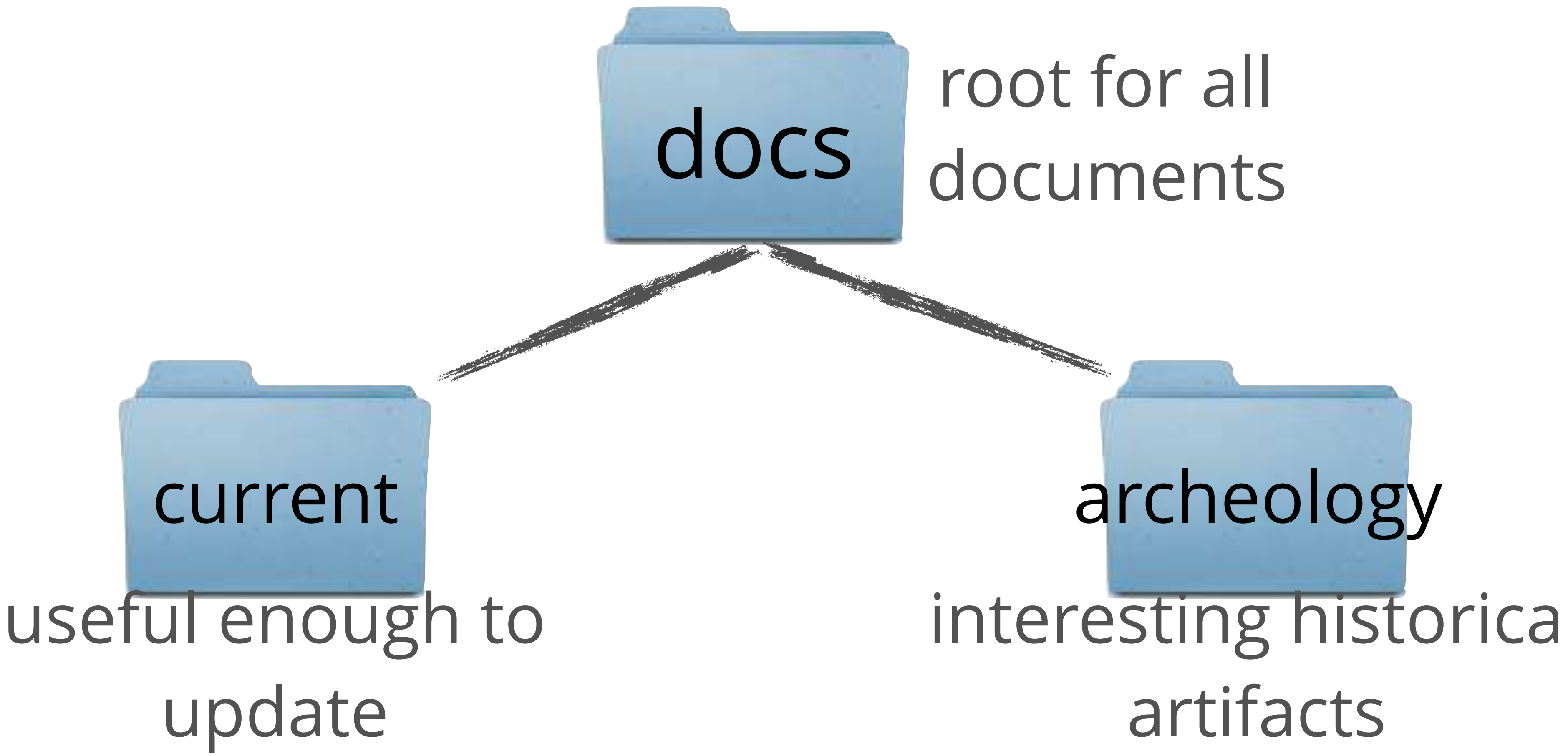
Figure 3: centralized broker

Identified issues to address:

- **throughput**: Centralized broker potentially creates a throughput bottleneck. However, developers analyzed past and expected future usage, and this shouldn't create problem.
- **internal application coupling**: Loose, with only one connection; app doesn't know about broker instances. The internal application doesn't need to know where the request originated.
- **changes to client**: additional brokers do not require changes to client.
- **single point of failure**: mitigated by clustering and failover provided by tools
- **performance**: because we expect low transaction volumes, performance should be sufficient with a single queue.

```
_Considerations_:

* broker only used for hub access
* low transaction volumes expected
* [illegible]

Two options exist:

### Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

[ditaa]
....
+--------+
|        |
|  b2b   |-------
|        |      |
+--------+      |
                |
+--------+      |    +--------+     +--------+     +------------+
|        |      ---->|        |     |        |     |  internal  |
|client  |---------->|hub     |---->|broker  |---->|application |
|        |      ---->|        |     |        |     |            |
+--------+      |    +--------+     +--------+     +------------+
                |
+--------+      |
|        |      |
|partner |-------
|        |
+--------+
....

_Figure 2: dedicated broker instances_

_Identified issues to address:_

* **throughput**:
Dedicated broker instances provide better throughput because no message contention exists.
* **internal application coupling**:
This approach requires changes to the internal client application to
"understand" the broker. Internal app must now connect to three brokers and keep context of request.
* **changes to client**: additional brokers added requires additional changes to client.
* **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
* **performance**: multiple broker instances should protect against aggregated performance problems.

### Centralized Broker

Using centralized broker instances creates the architecture shown in Figure 3.

[ditaa]
....
+--------+       +--------+
|        |       |        |
|  b2b   |------>|hub     |-------
|        |       |        |      |
+--------+       +--------+      |
                                 |
+--------+       +--------+      |    +--------+     +------------+
|        |       |        |      --->|        |     |  internal  |
|client  |------>|hub     |--------->|broker  |---->|application |
|        |       |        |      --->|        |     |            |
+--------+       +--------+      |    +--------+     +------------+
                                 |
+--------+       +--------+      |
|        |       |        |      |
|partner |------>|hub     |-------
|        |       |        |
+--------+       +--------+
....
```

Archeology


docs
root for all documents
current
useful enough to update
archeology
interesting historical artifacts

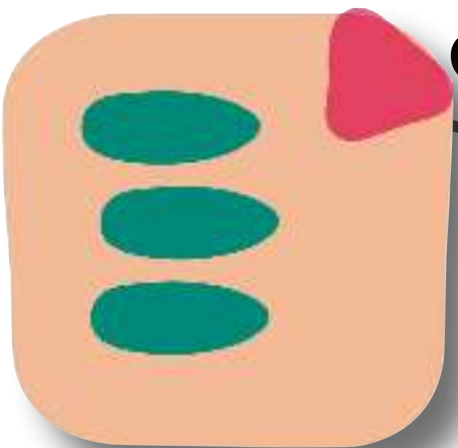
Rules for Documentation

1. useful <u>now</u>

2. as little as possible

3. always accurate

Documenting & Presenting Software Architecture



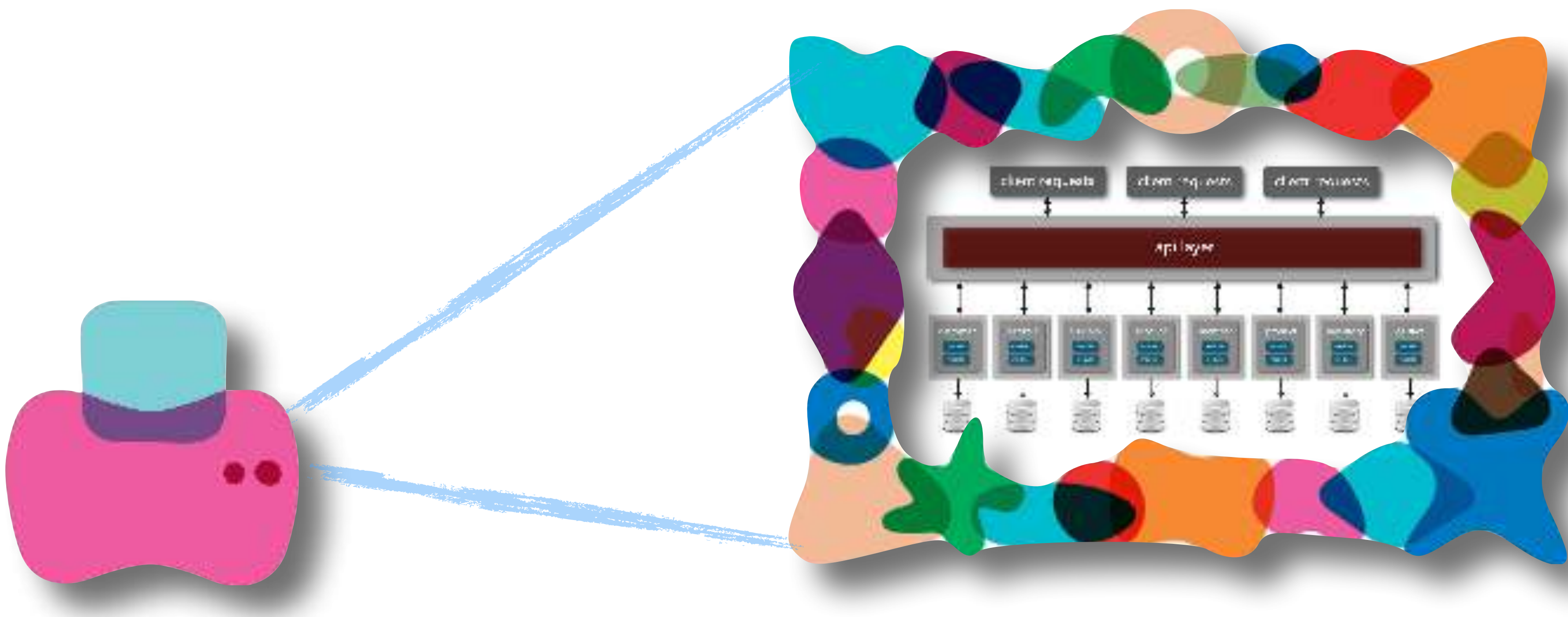
ThoughtWorks

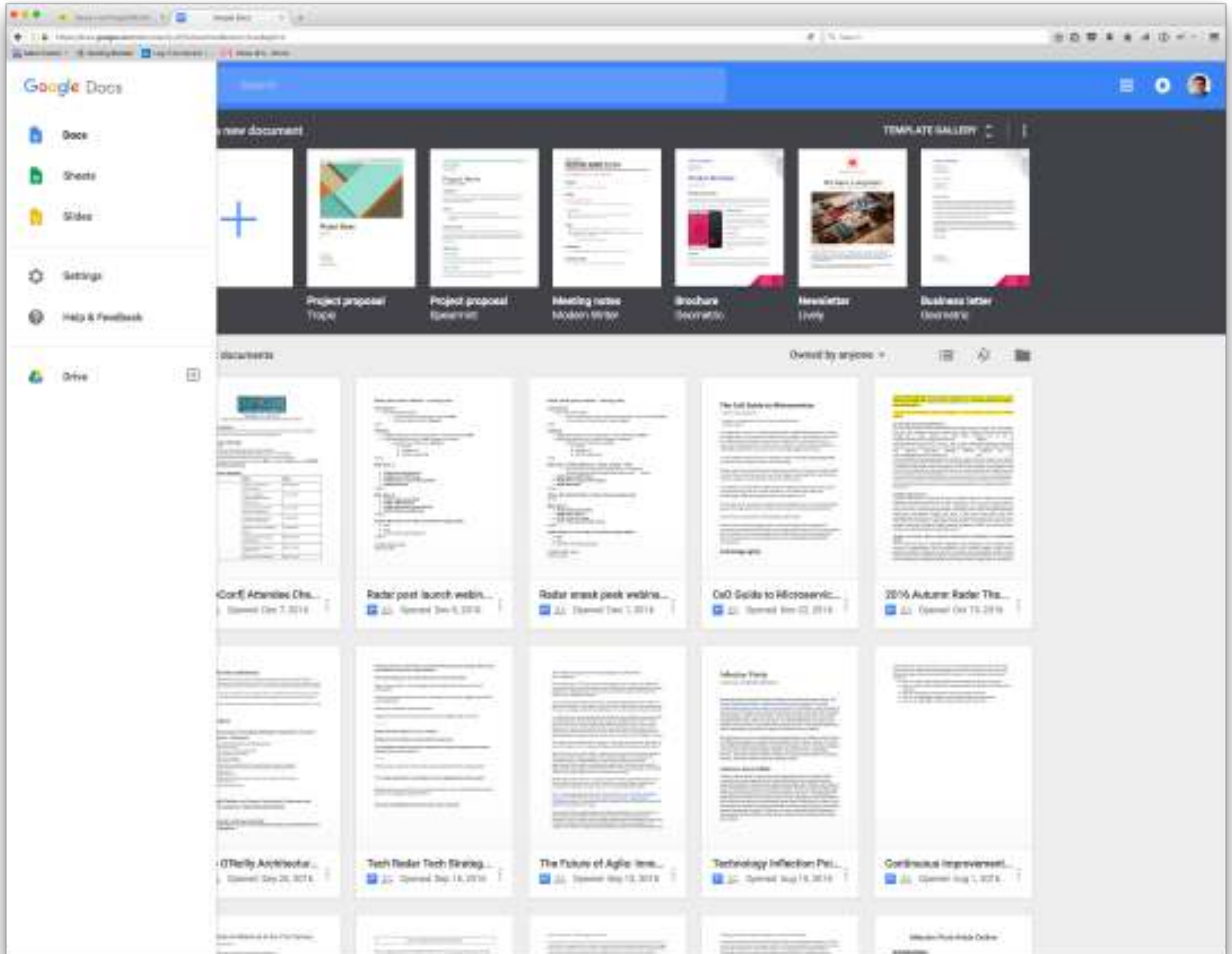
NEAL FORD

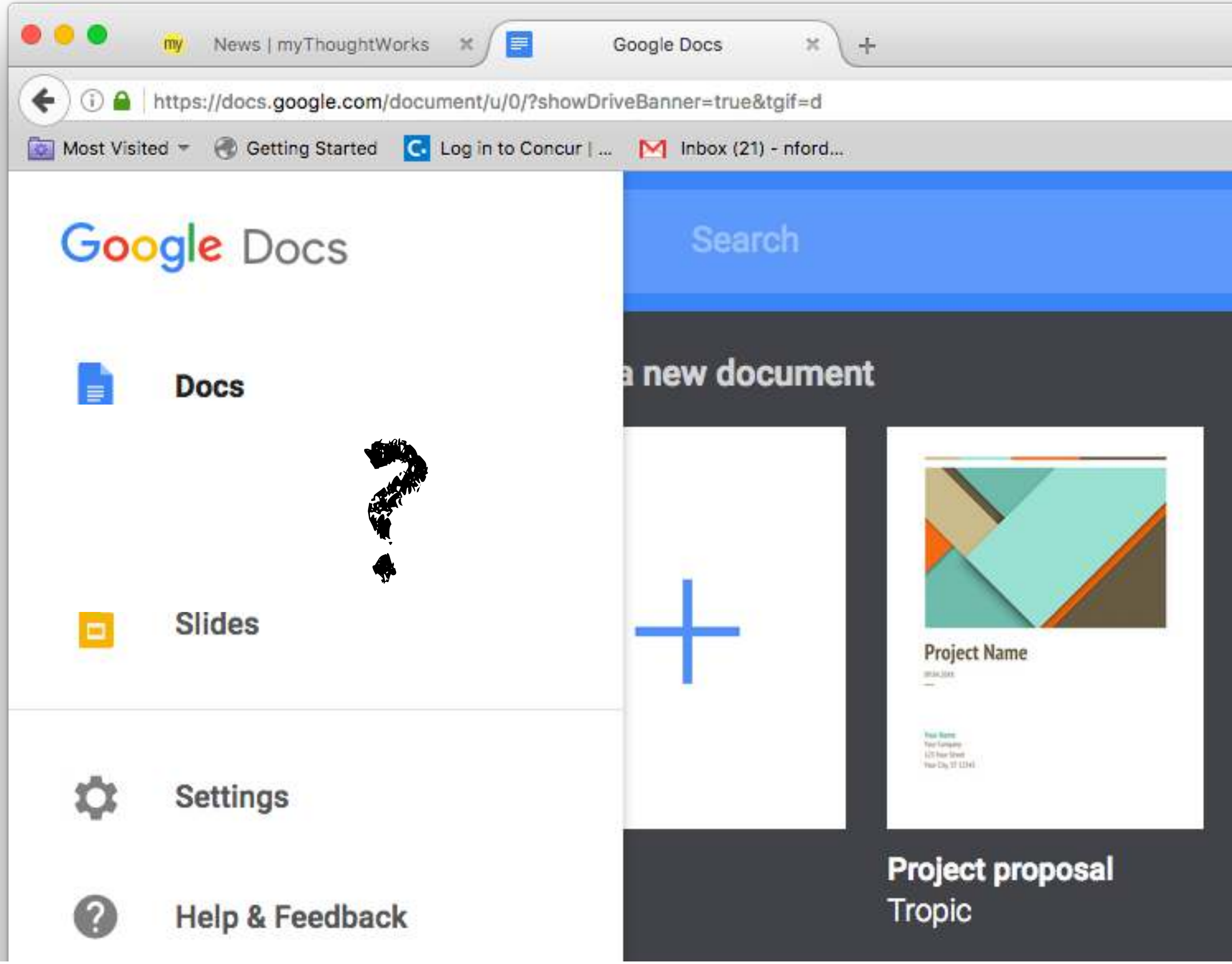
Director / Software Architect / Meme Wrangler

@neal4d
nealford.com

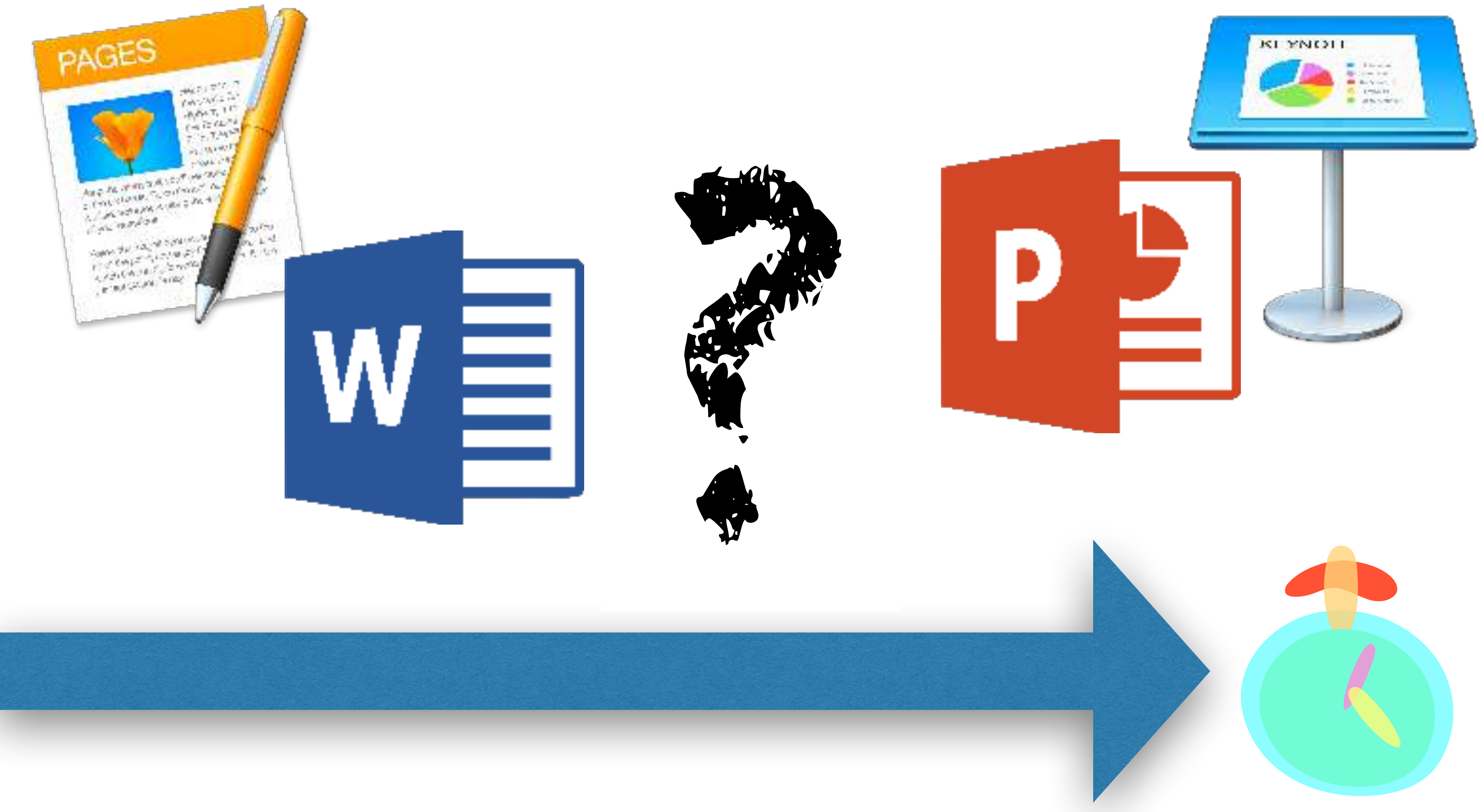
Presenting Software Architecture

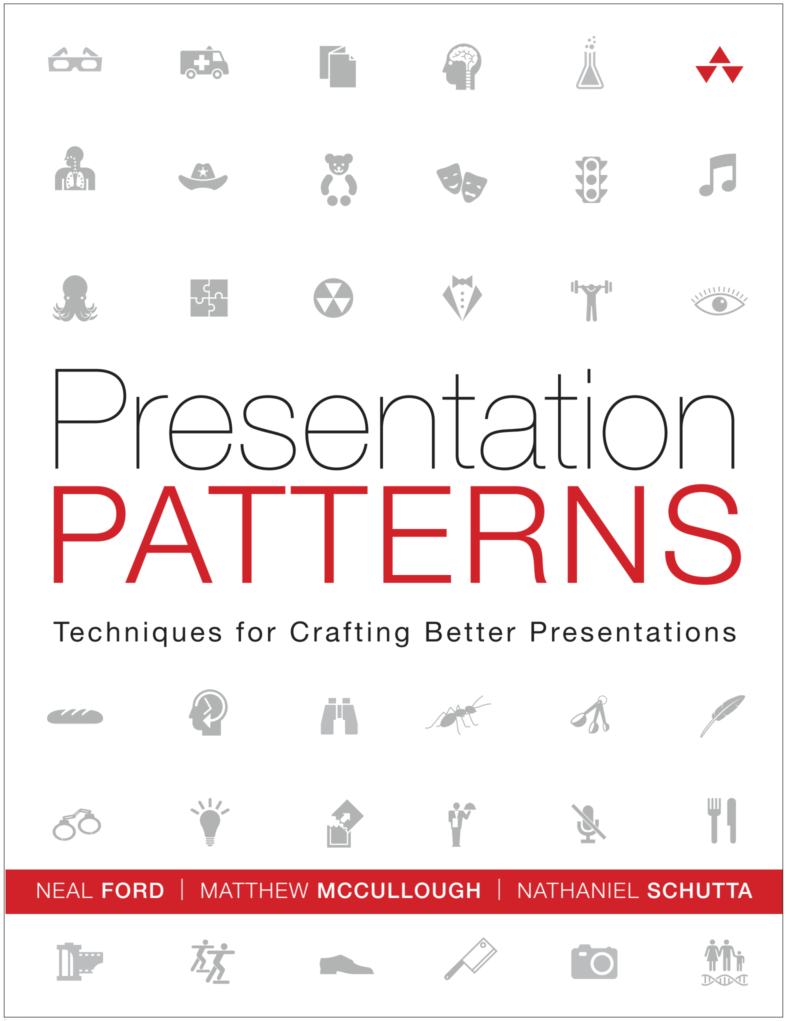


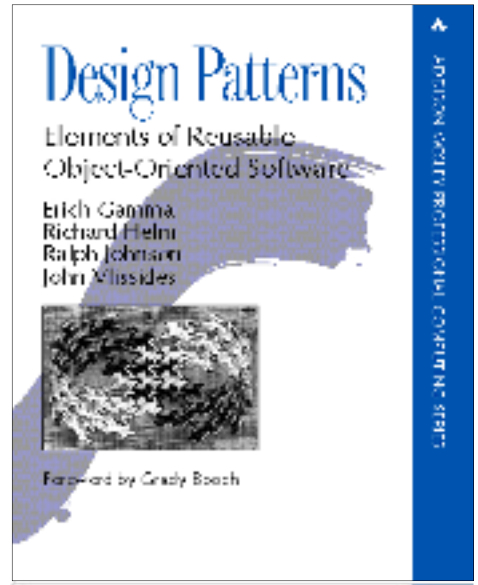

Google Docs
Docs
Sheets
Slides
Settings
Help & Feedback
Drive
new document
TEMPLATE GALLERY
Project proposal
Tropic
Project proposal
Spearmint
Meeting notes
Modern Writer
Brochure
Geometric
Newsletter
Lively
Business letter
Geometric
documents
Owned by anyone



News | myThoughtWorks
Google Docs
https://docs.google.com/document/u/0/?showDriveBanner=true&tgif=d
Most Visited
Getting Started
Log in to Concur | ...
Inbox (21) - nford...
Google Docs
Search
Docs
a new document
Slides
Project Name
Settings
Help & Feedback
Project proposal
Tropic


PAGES

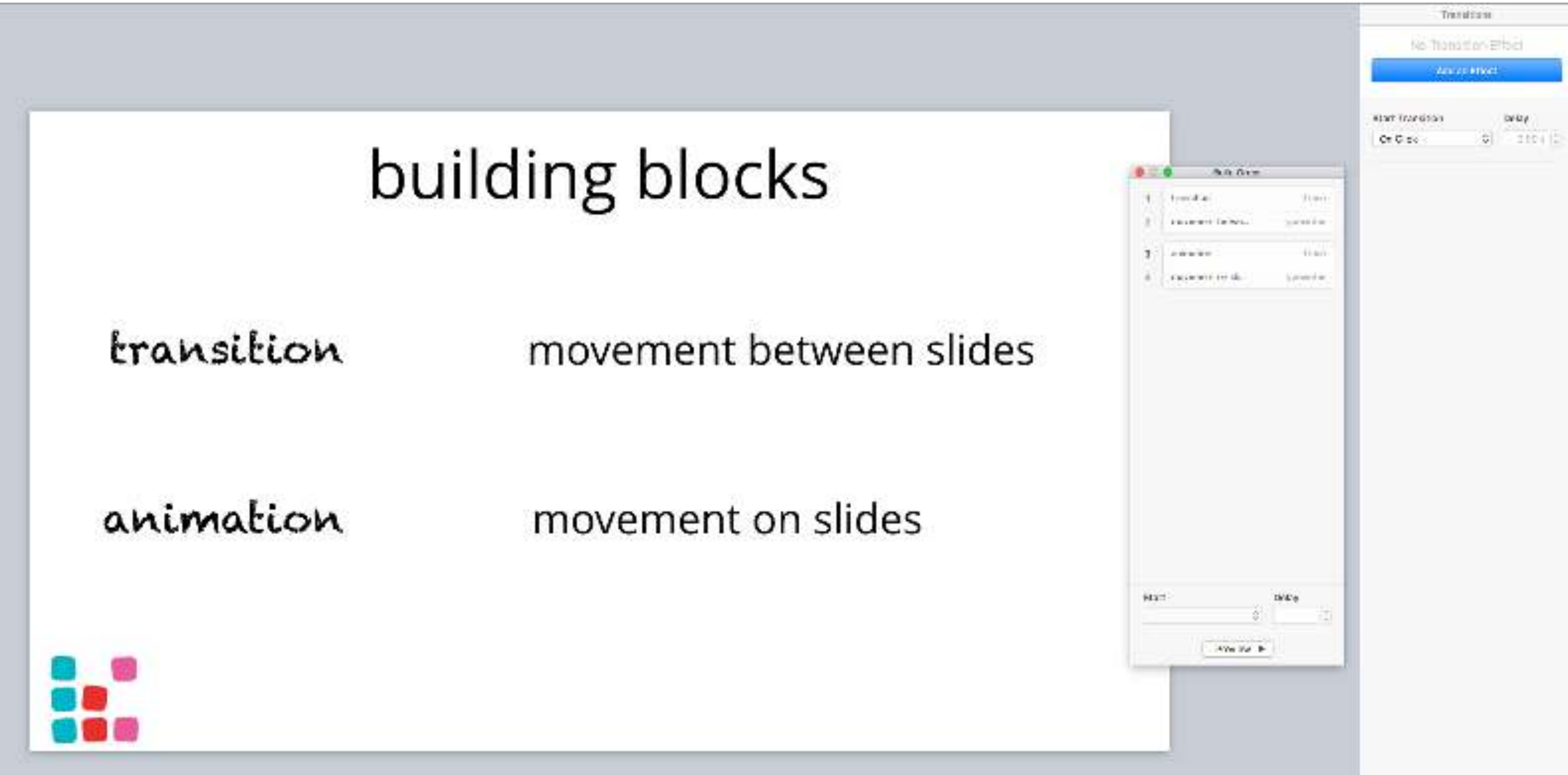

Presentation
PATTERNS
Techniques for Crafting Better Presentations
NEAL FORD | MATTHEW MCCULLOUGH | NATHANIEL SCHUTTA


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch

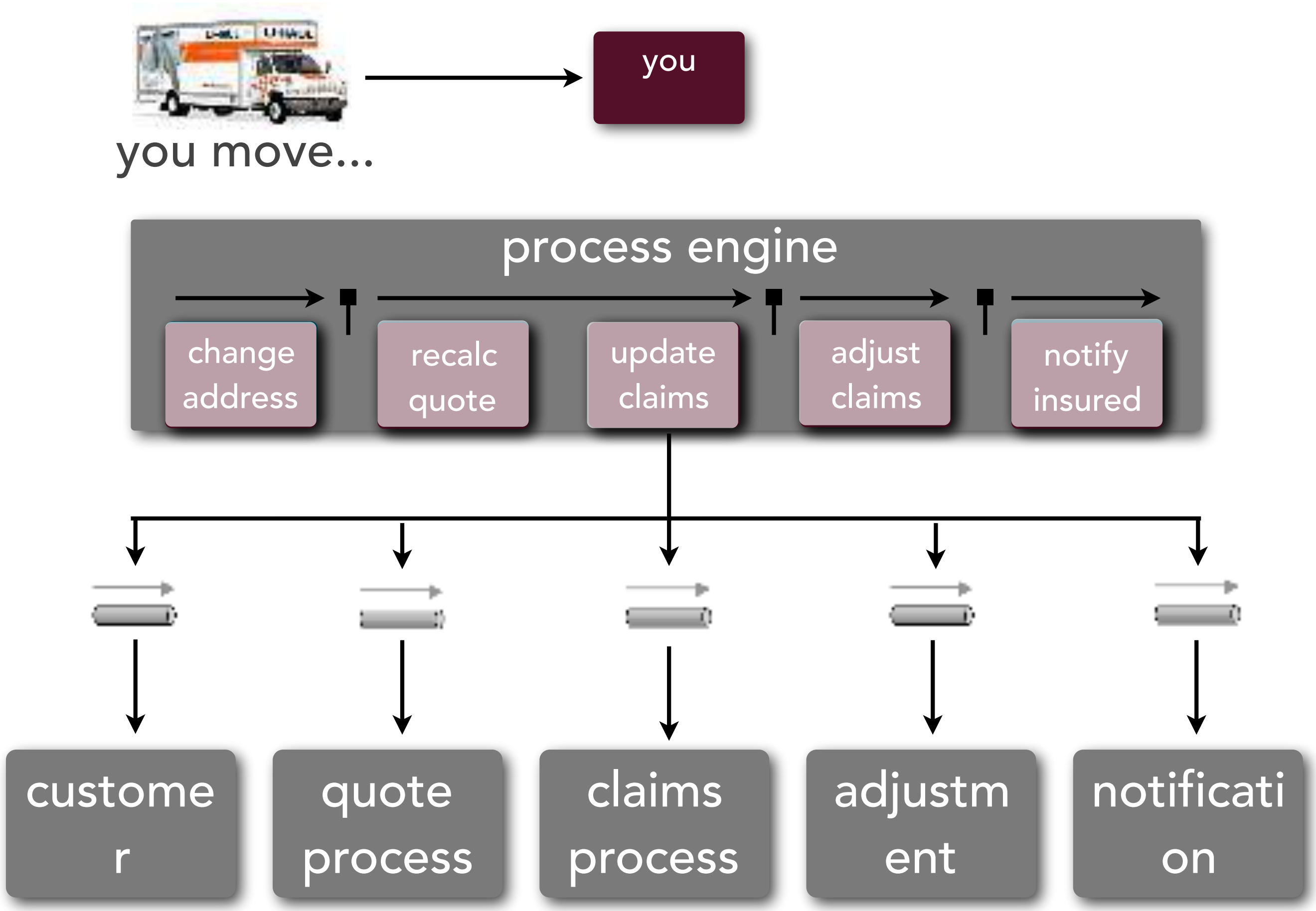
building blocks

building blocks

transition	movement between slides
animation	movement on slides


building blocks
transition
movement between slides
animation
movement on slides

animations



you move...
you
process engine
change address
recalc quote
update claims
adjust claims
notify insured
customer
quote process
claims process
adjustment
notification

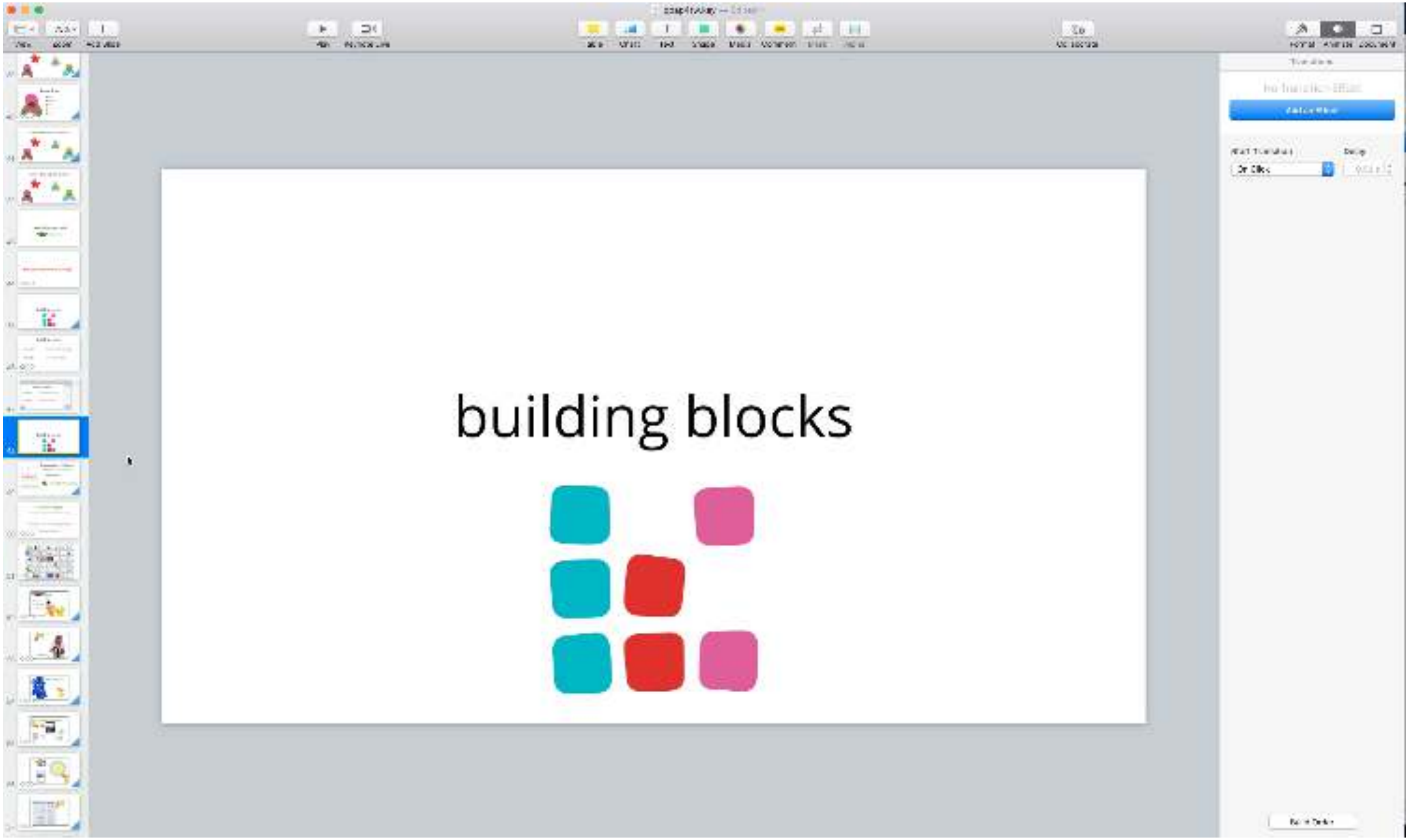
building blocks

building blocks

transition — movement between slides

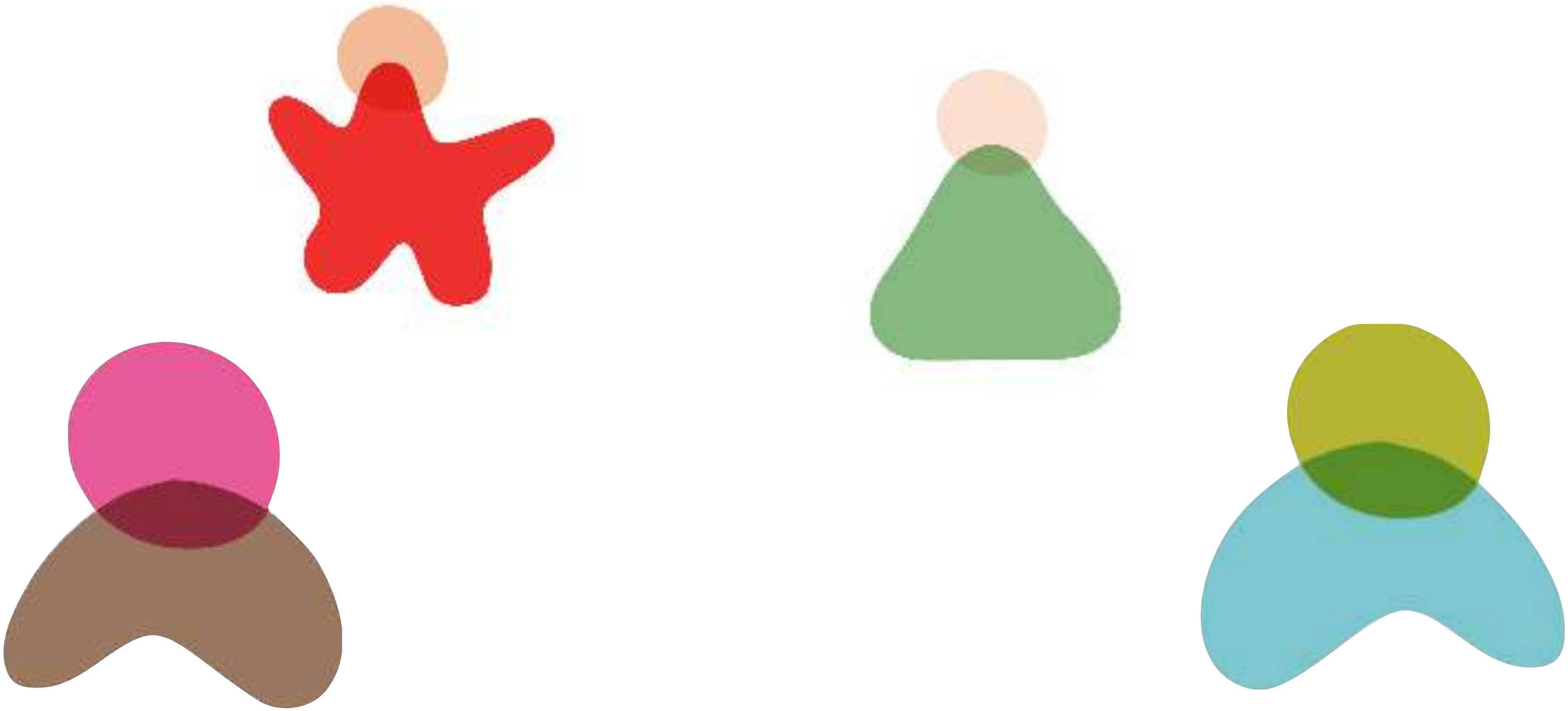
animation — movement on slides

Magic Move

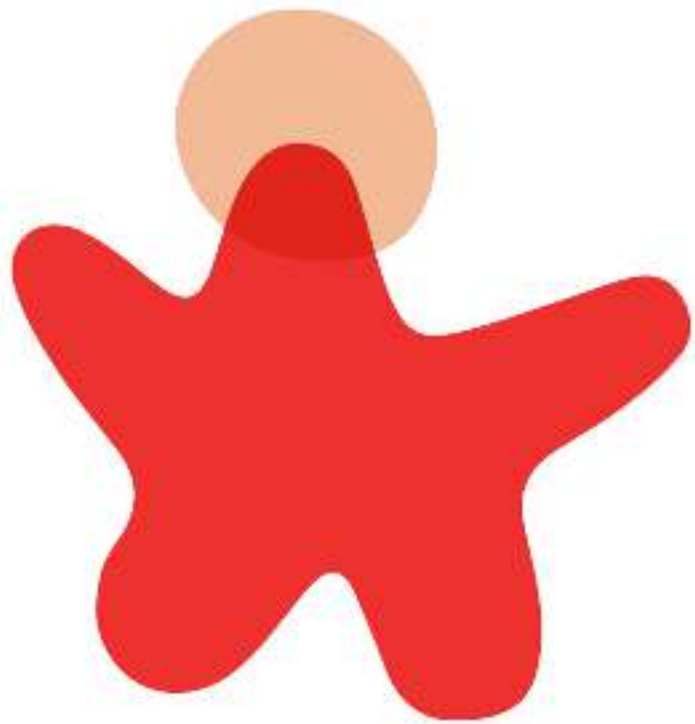

building blocks

Magic Move for tools that dont have Magic Move

Magic Move Version

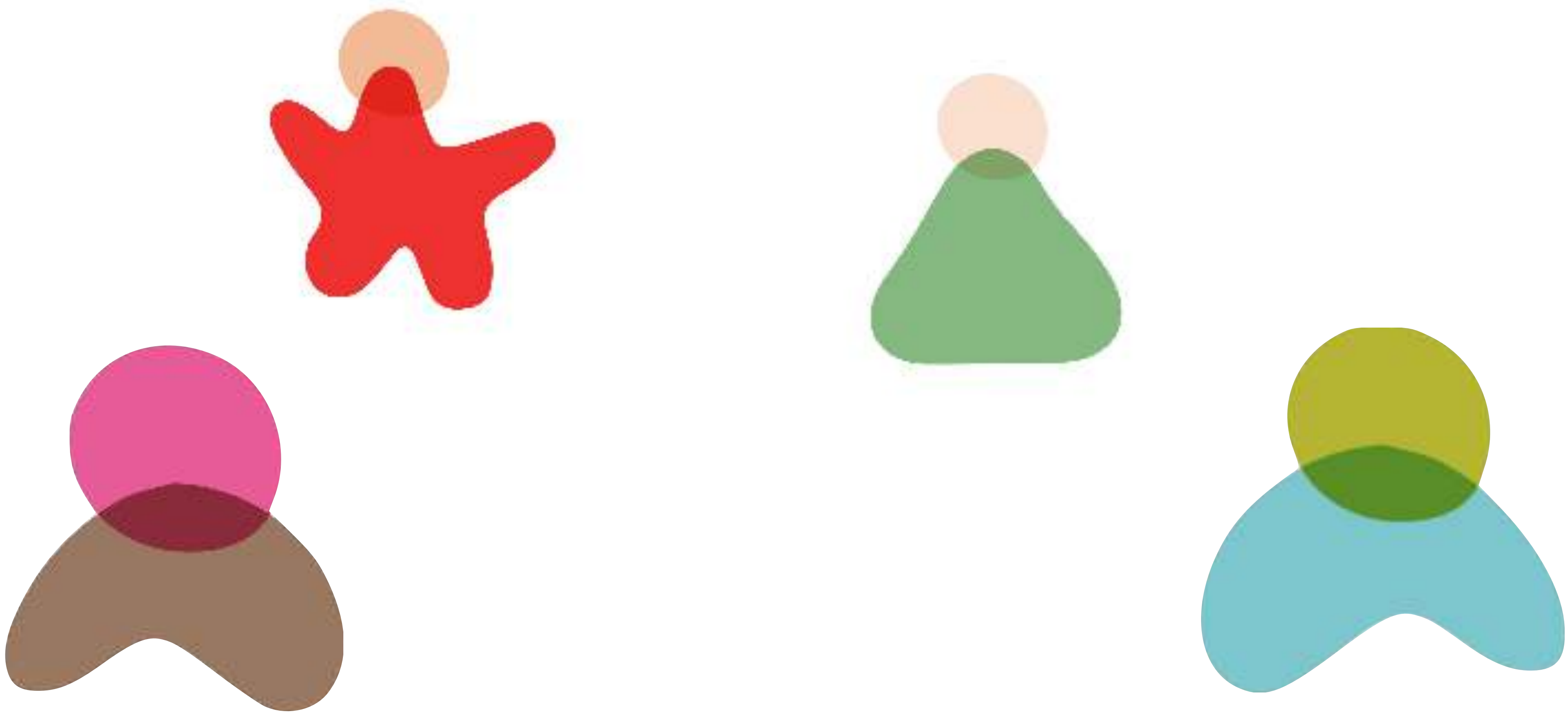


Red Shirt

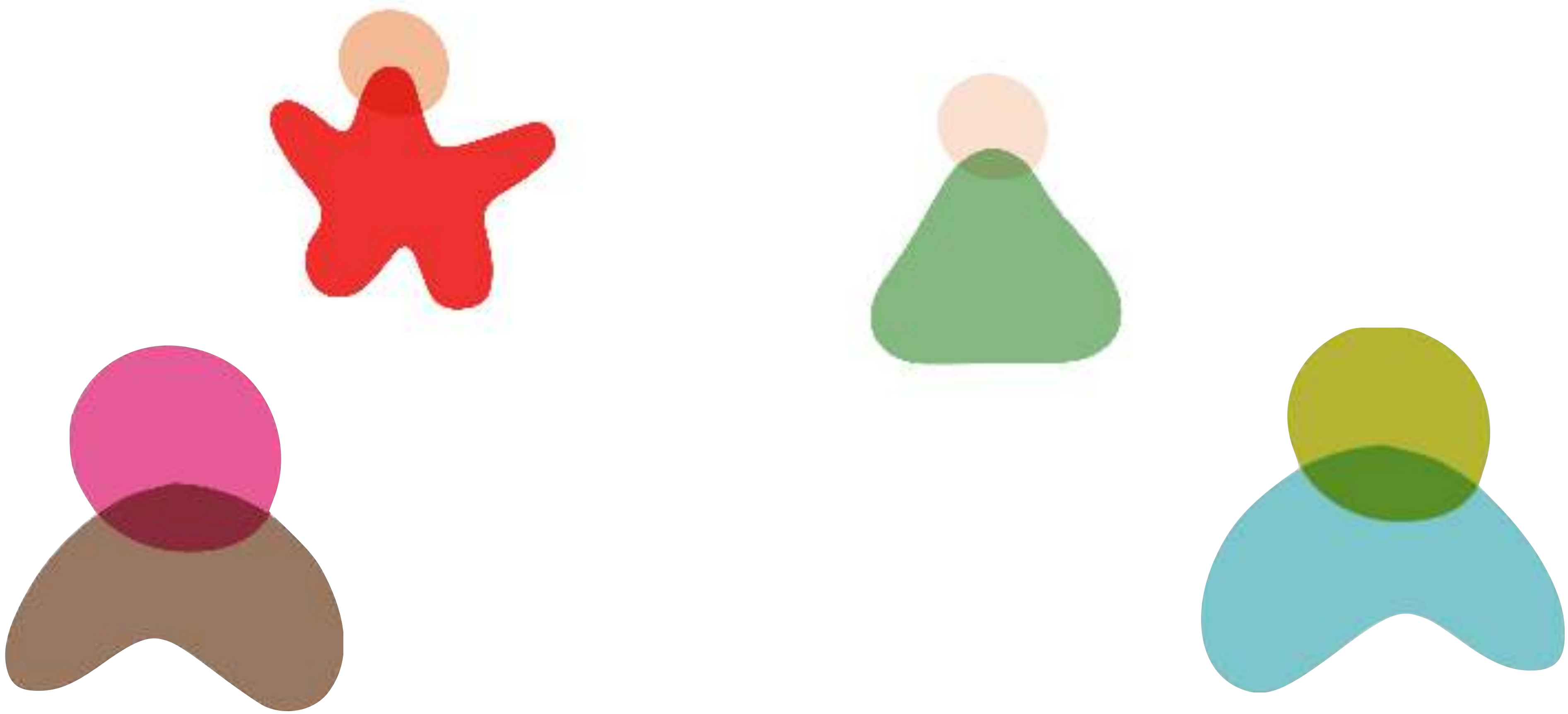


- ☑ Fact 1
- ☑ Another Fact
- ☑ Fact 3
- ☑ Yet Another Fact
- ☑ Fact 5

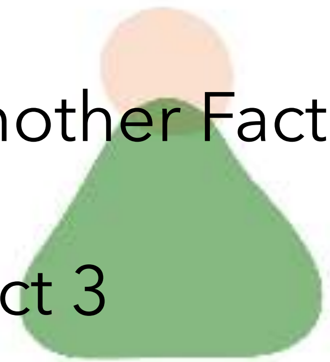
Magic Move Version



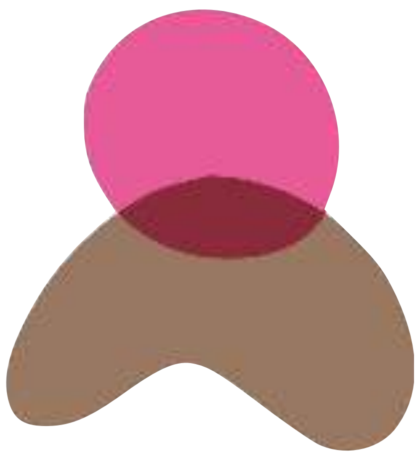
Non-*Magic Move* Version

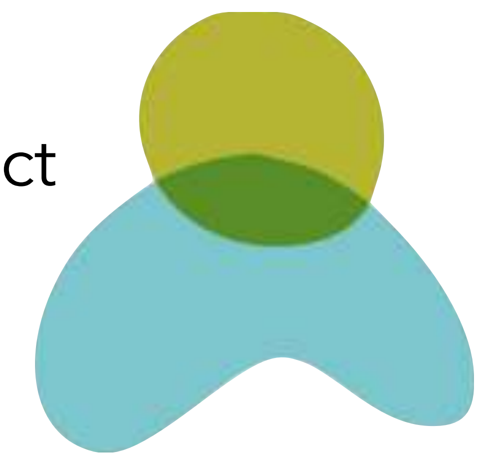


Non-*Magic Move* Version

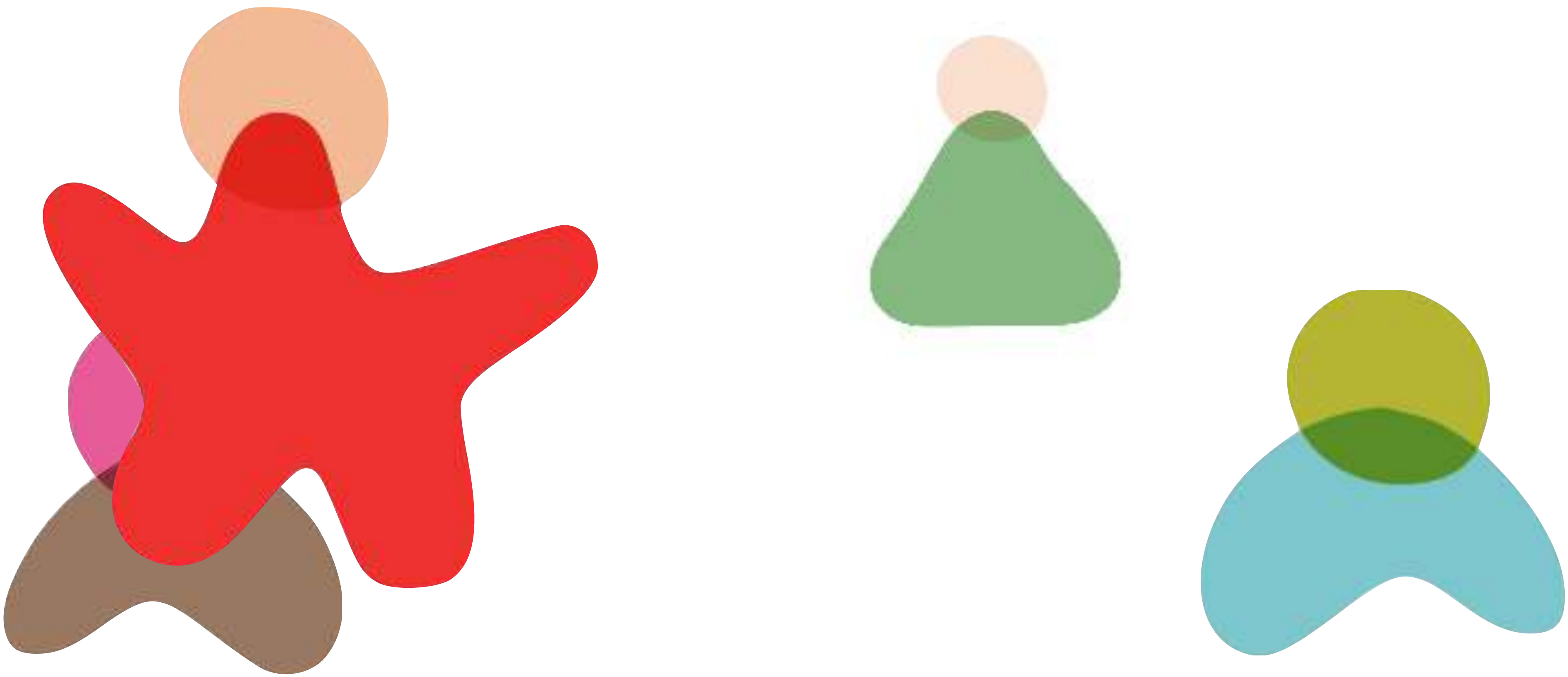


- [x] Fact 1
- [x] Another Fact
- [x] Fact 3
- [x] Yet Another Fact
- [x] Fact 5

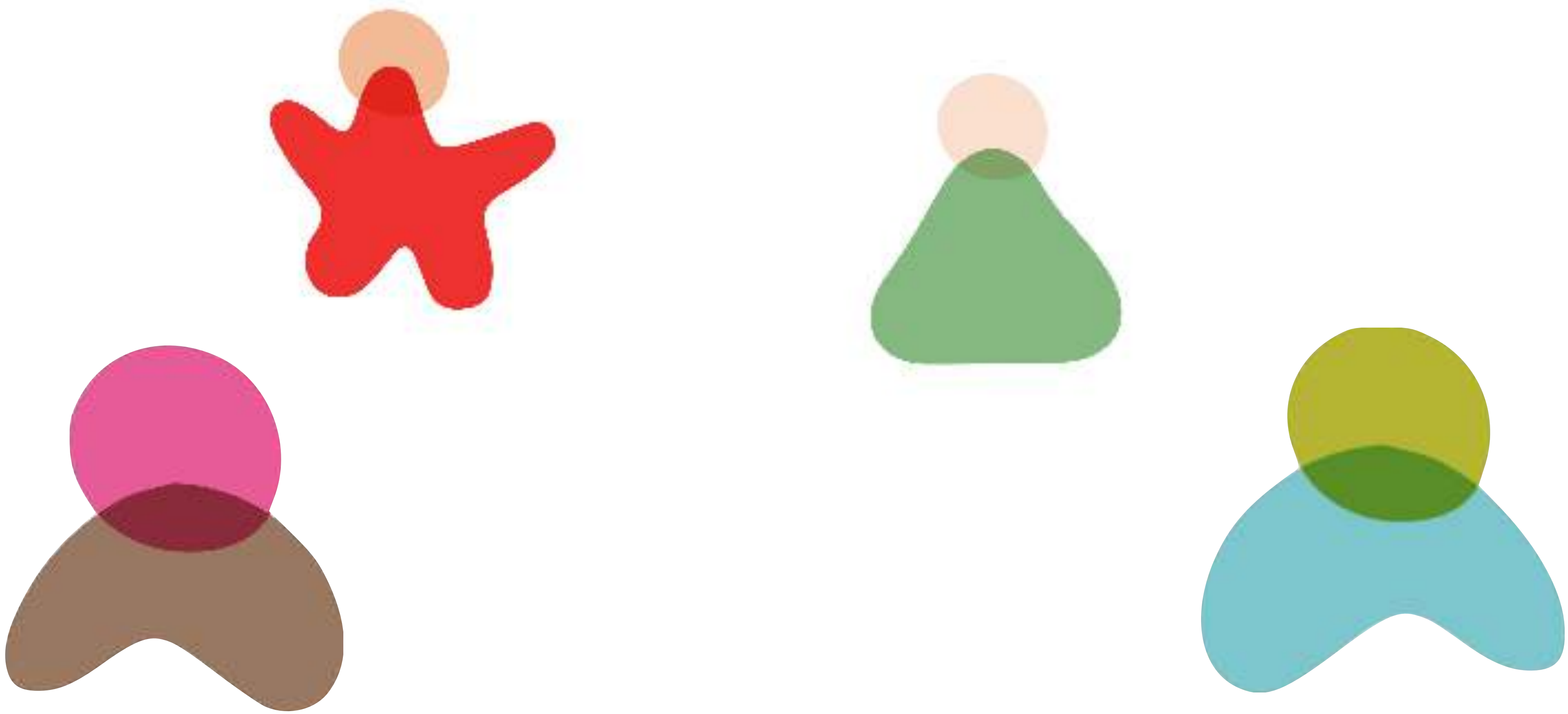




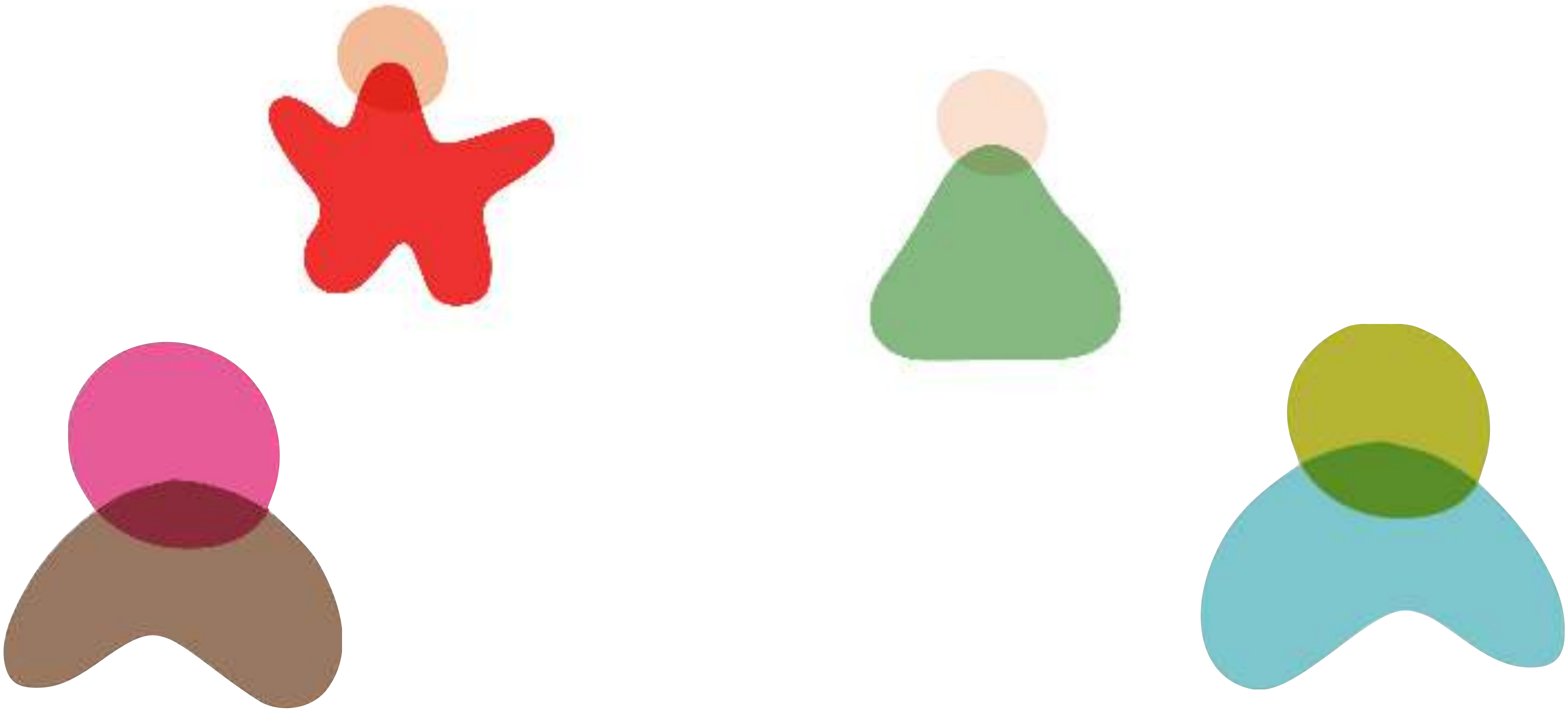
Non-*Magic Move* Version

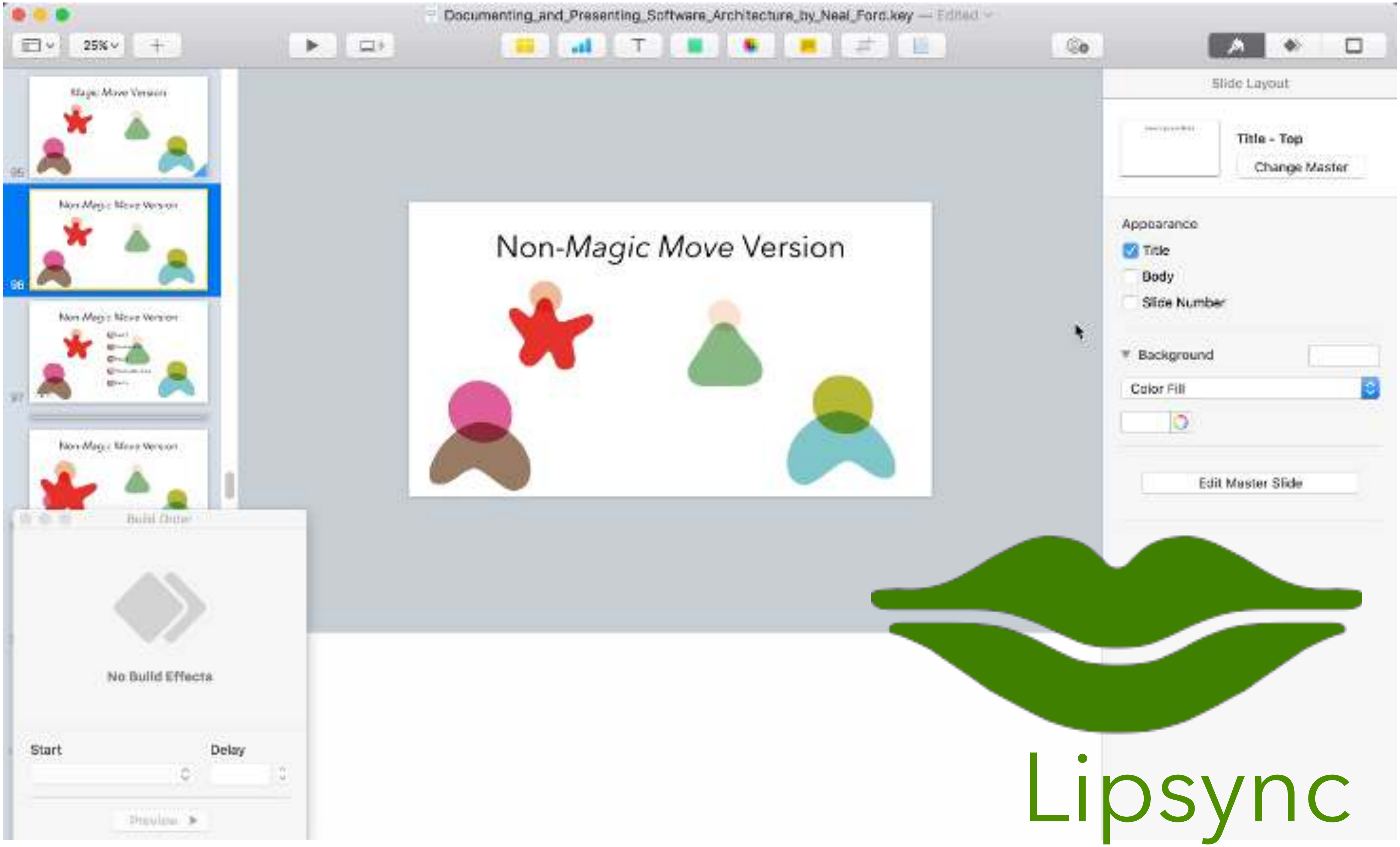


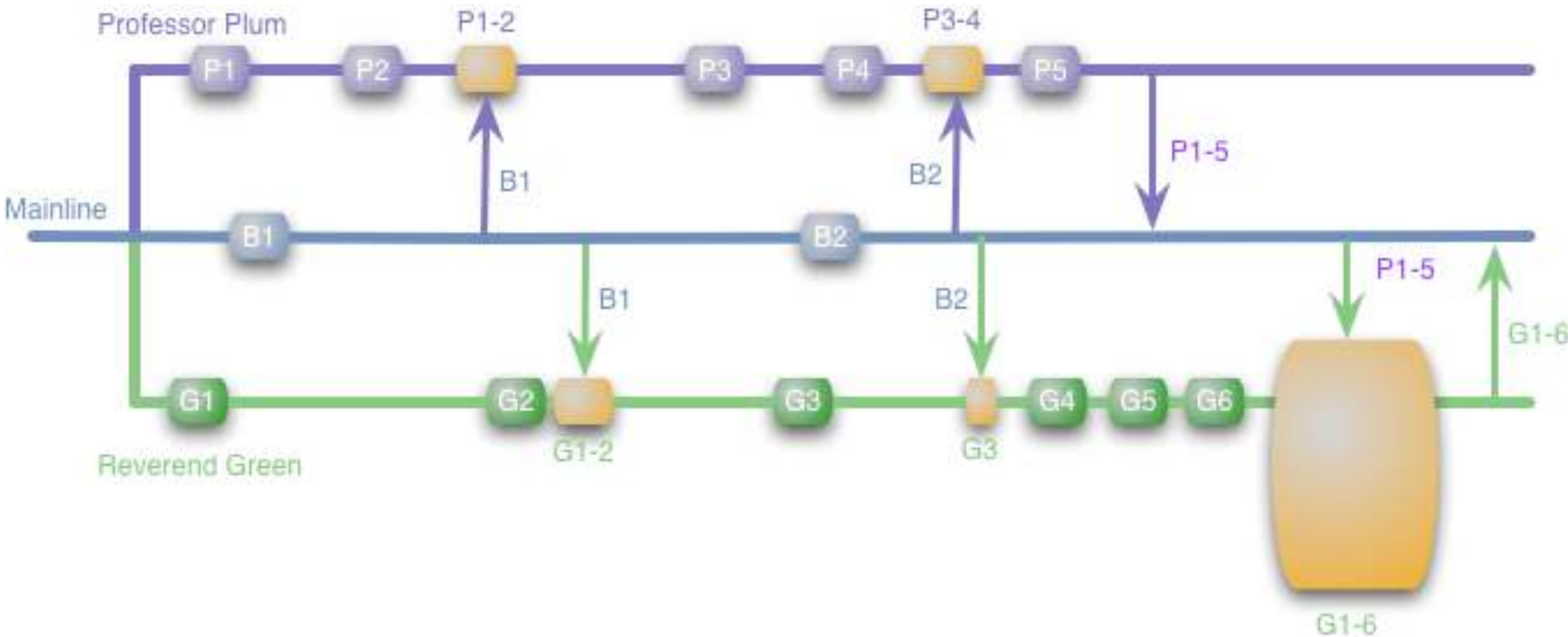
Non-*Magic Move* Version



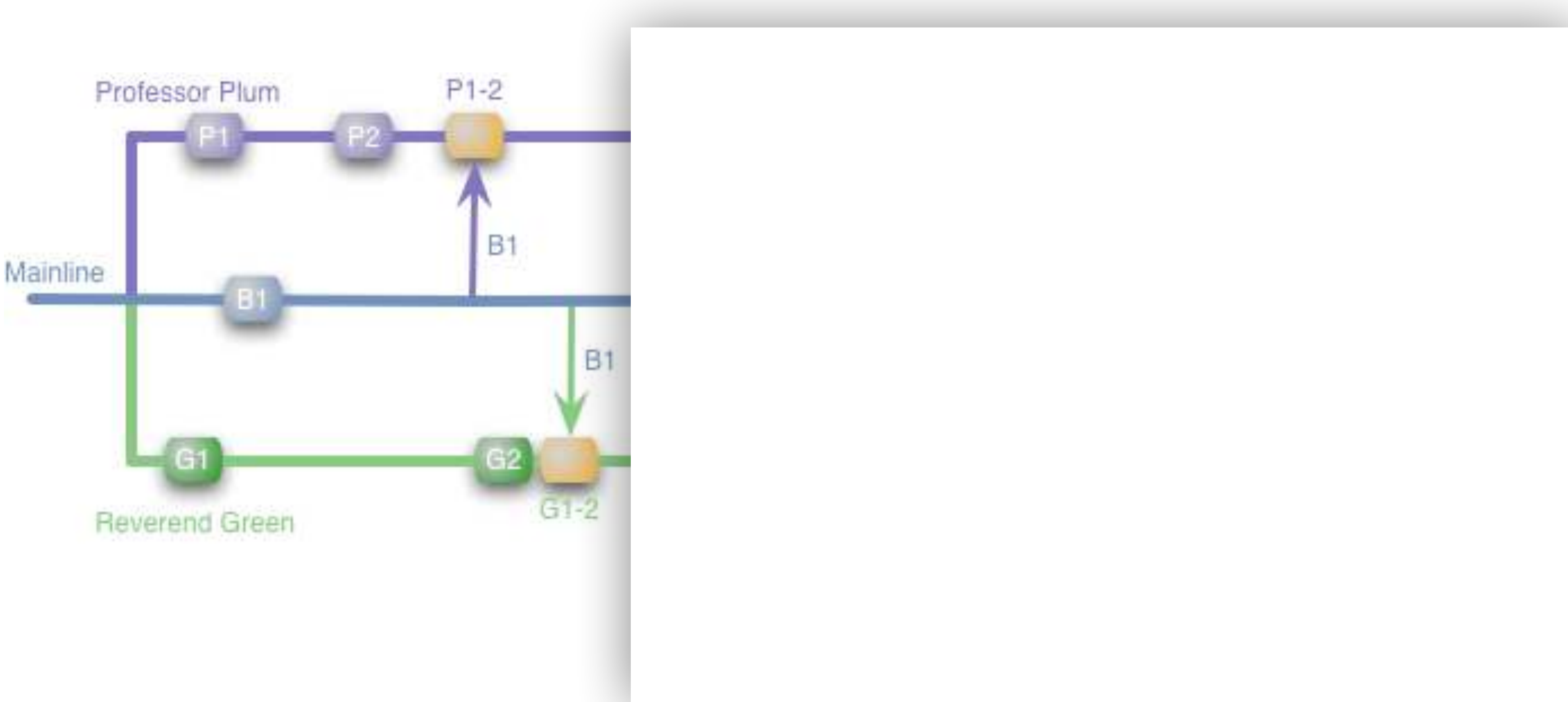
Non-*Magic Move* Version

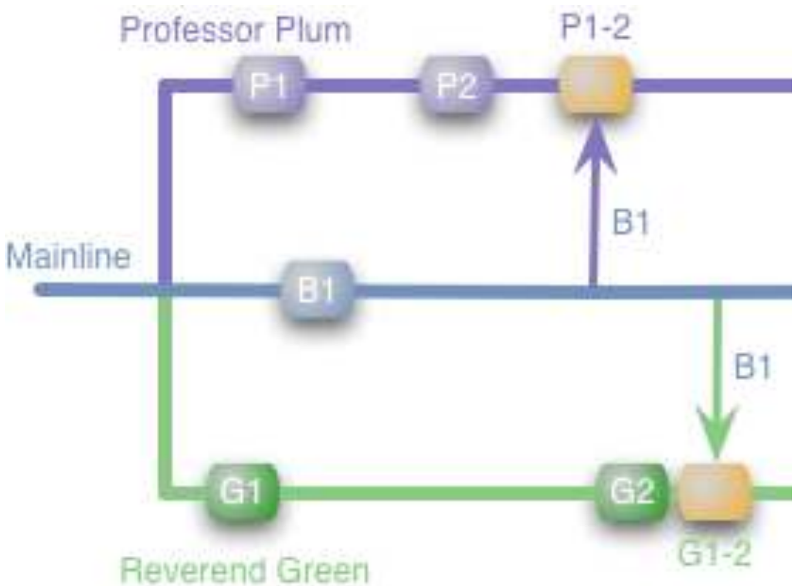


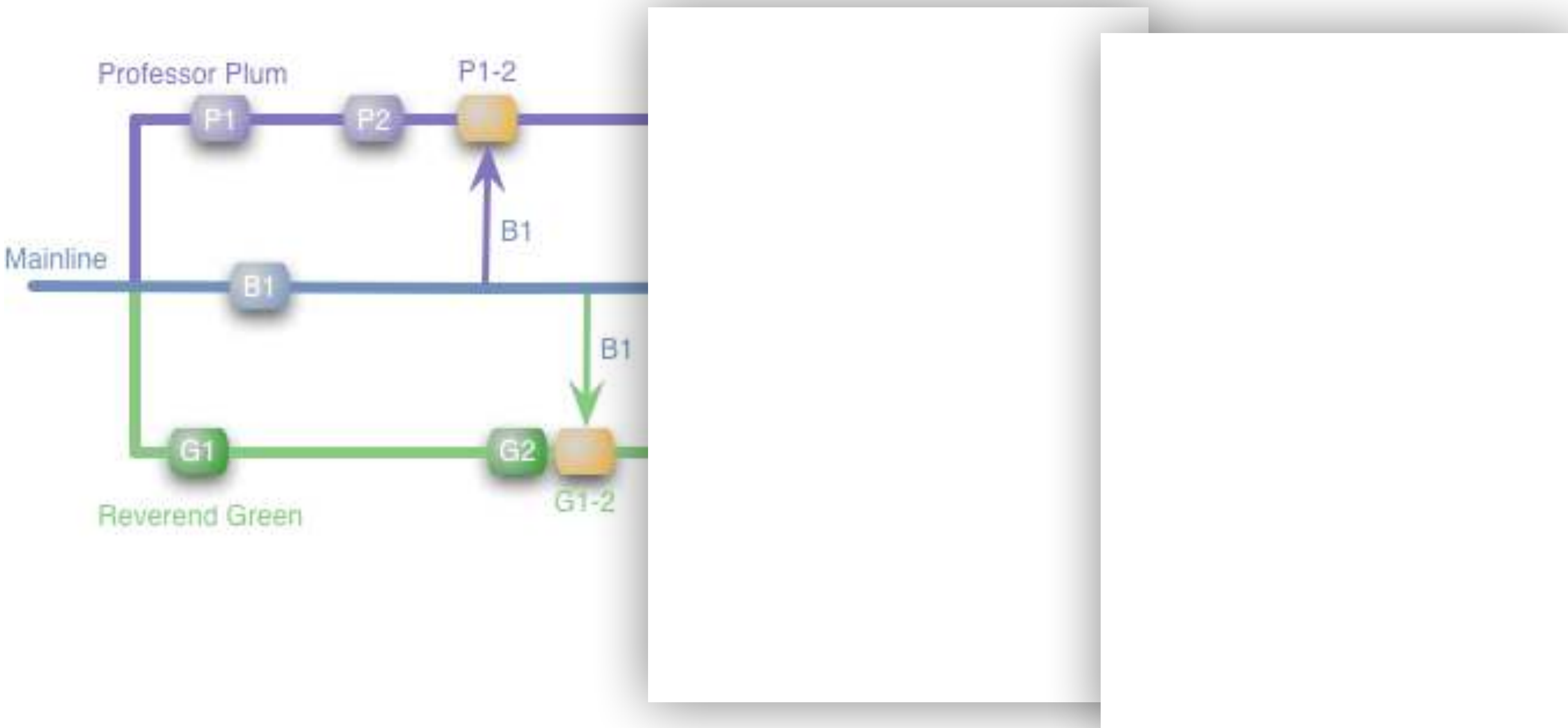

Documenting_and_Presenting_Software_Architecture_by_Neal_Ford.key — Edited
25%
Magic Move Version
Non-Magic Move Version
Non-Magic Move Version
Non-Magic Move Version
Non-Magic Move Version
Slide Layout
Title - Top
Change Master
Appearance
Title
Body
Slide Number
Background
Color Fill
Edit Master Slide
Build Order
No Build Effects
Start
Delay
Preview
Lipsync



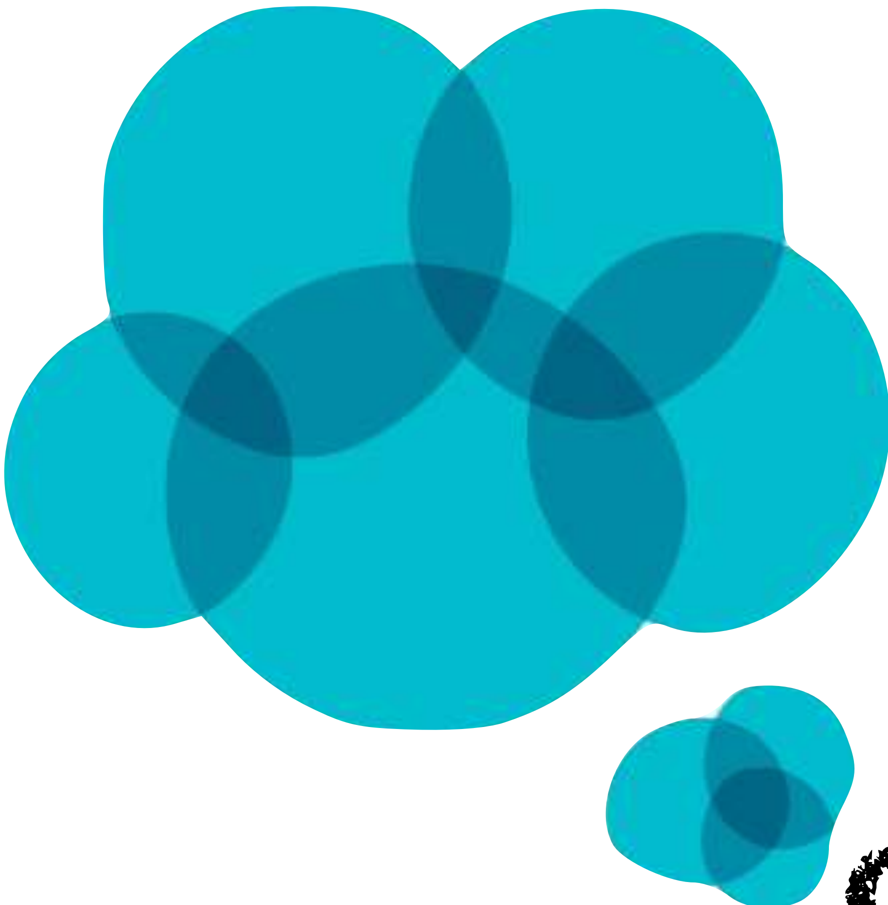
https://martinfowler.com/bliki/FeatureBranch.html


Professor Plum
P1
P2
P1-2
B1
Mainline
B1
B1
G1
G2
G1-2
Reverend Green

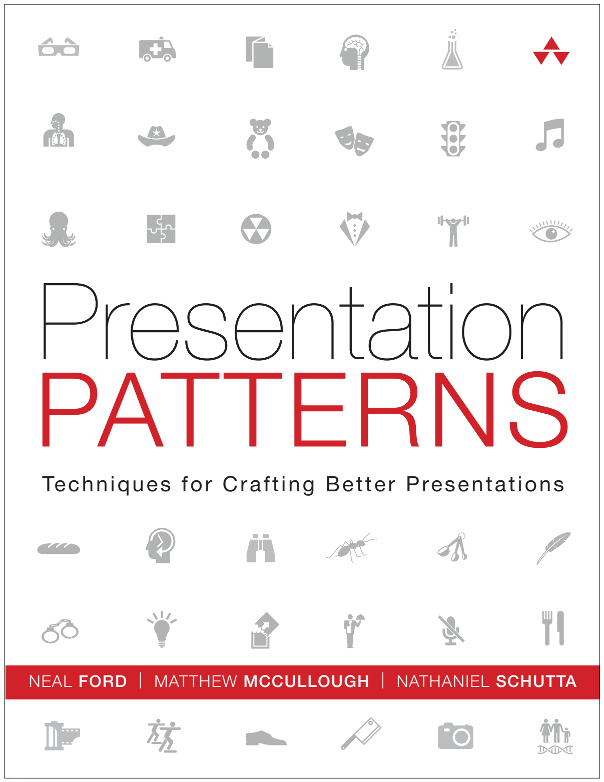

Professor Plum
P1-2
P1
P2
B1
Mainline
B1
B1
G1
G2
G1-2
Reverend Green



Professor Plum
P1
P2
P1-2
B1
Mainline
B1
B1
G1
G2
G1-2
Reverend Green



creativity



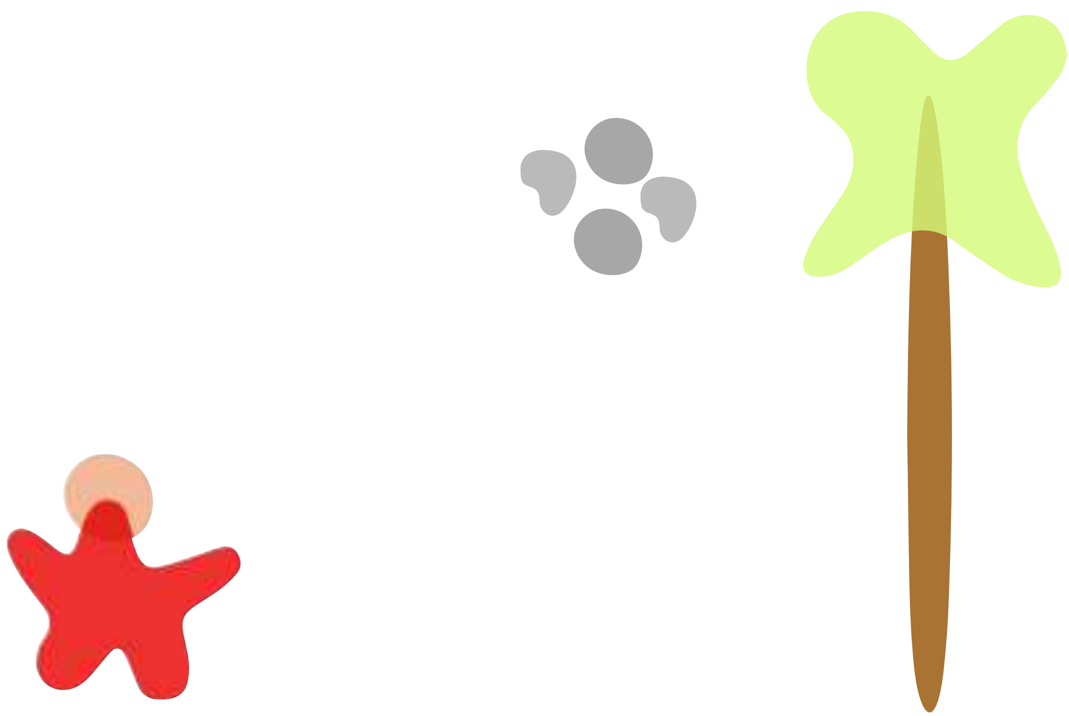
Presentation Patterns

Building Blocks for Perfect Presentations

Creativity Patterns:

Narrative Arc

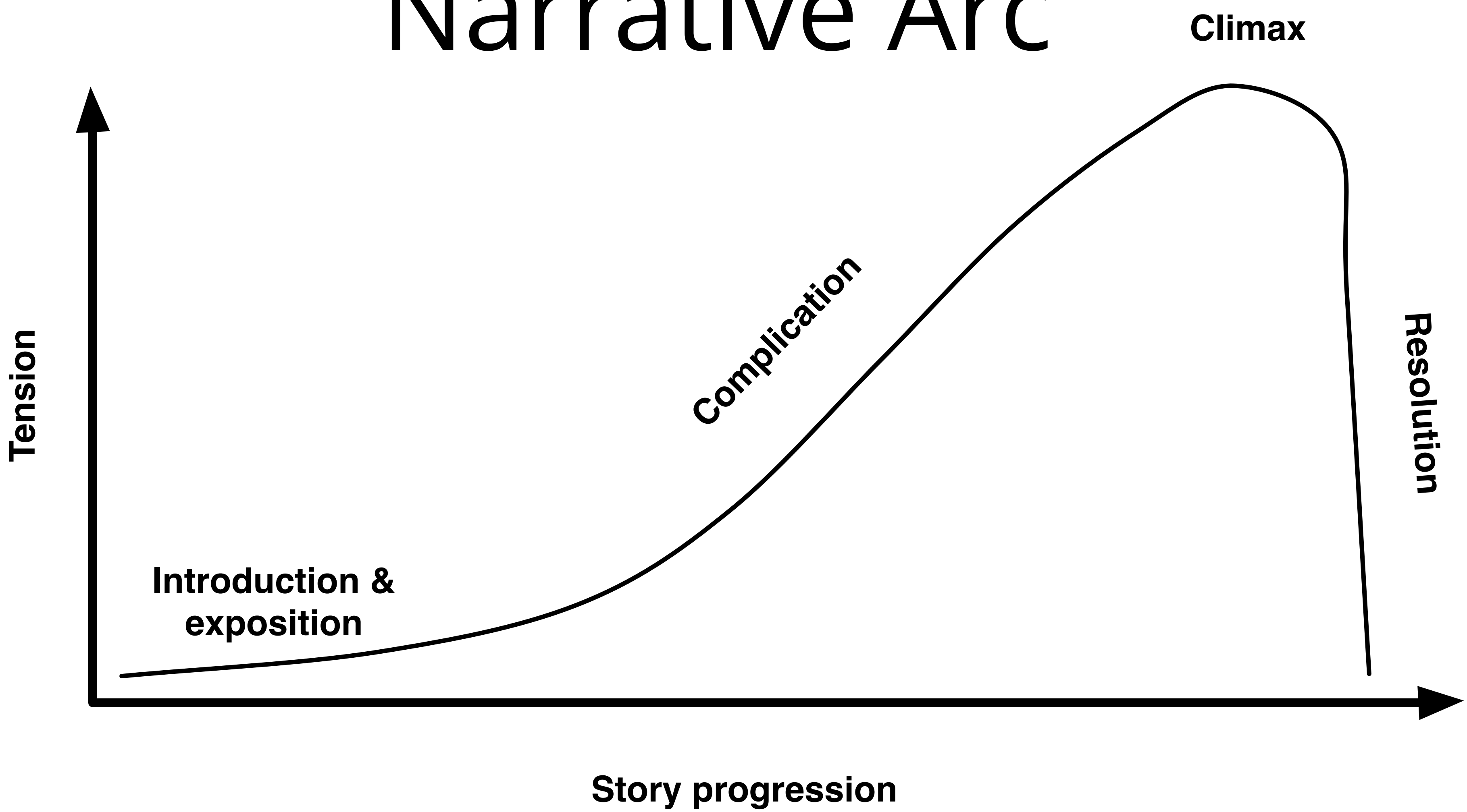
Presentations are a form of …



"Get your protagonist up a tree. Throw rocks at him. Then get him down."

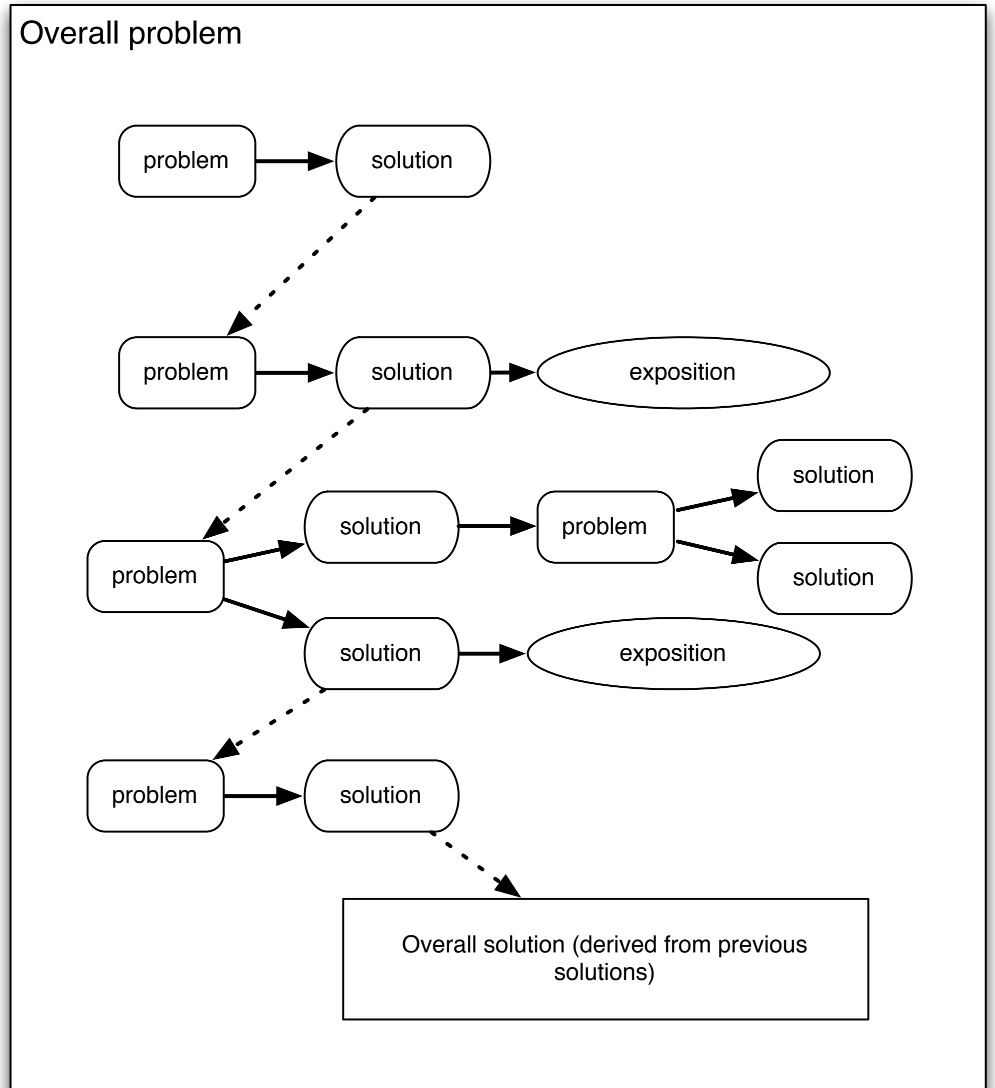
–Syd Fields

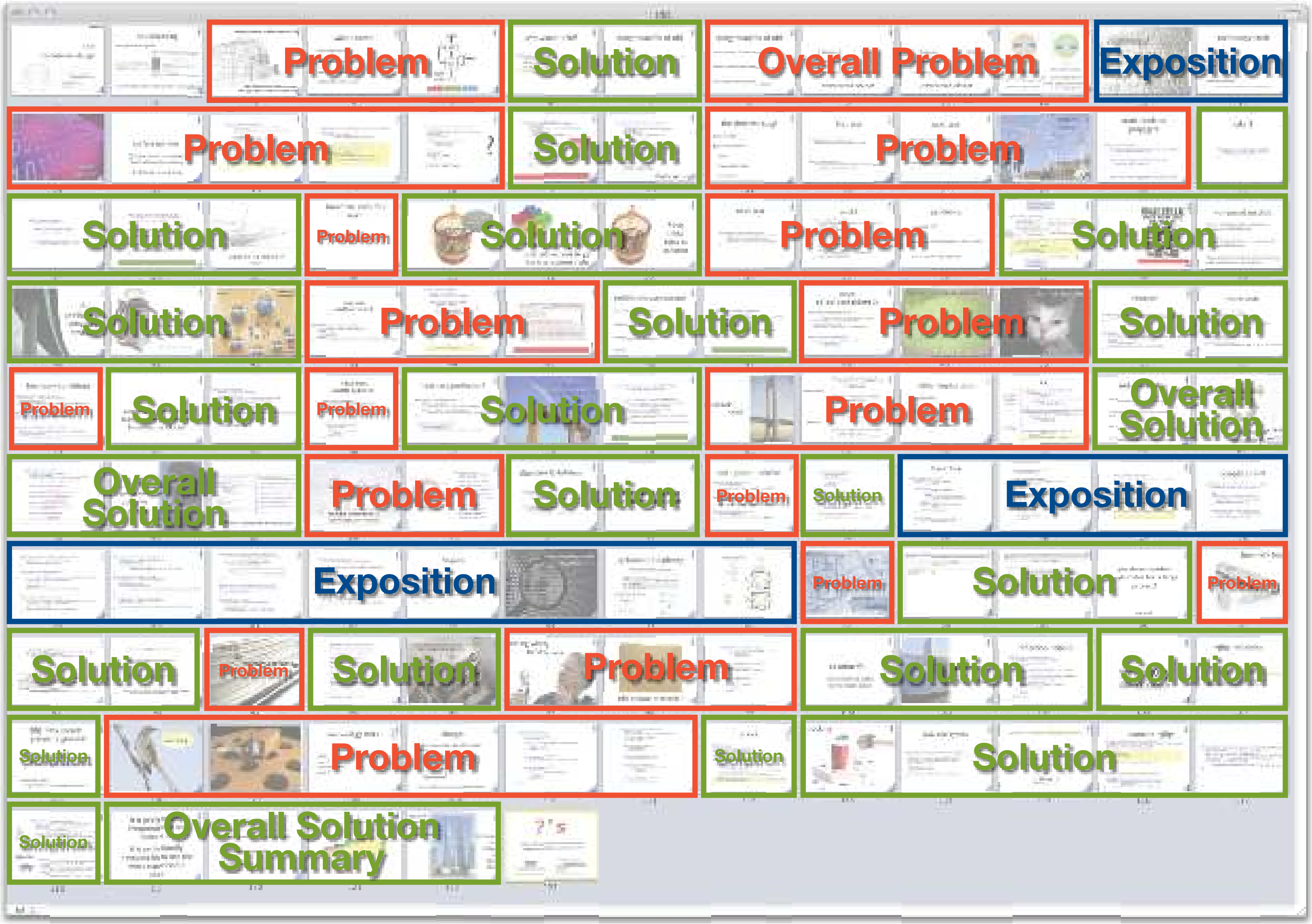
Narrative Arc



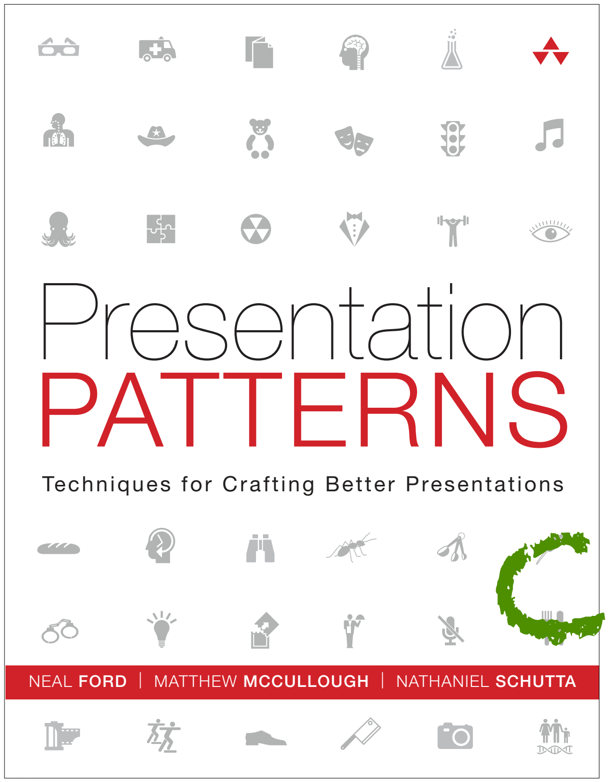

Why So Many Superheroes?

Technical Narrative?





Problem
Solution
Overall Problem
Exposition
Problem
Solution
Problem
Solution
Problem
Solution
Problem
Solution
Solution
Problem
Solution
Problem
Solution
Problem
Solution
Problem
Solution
Problem
Overall Solution
Overall Solution
Problem
Solution
Problem
Solution
Exposition
Exposition
Problem
Solution
Problem
Solution
Problem
Solution
Problem
Solution
Solution
Solution
Problem
Solution
Solution
Solution
Overall Solution Summary



Presentation Patterns

Building Blocks for Perfect Presentations

Creativity Patterns:

Concurrent Creatio

Concurrent Creation

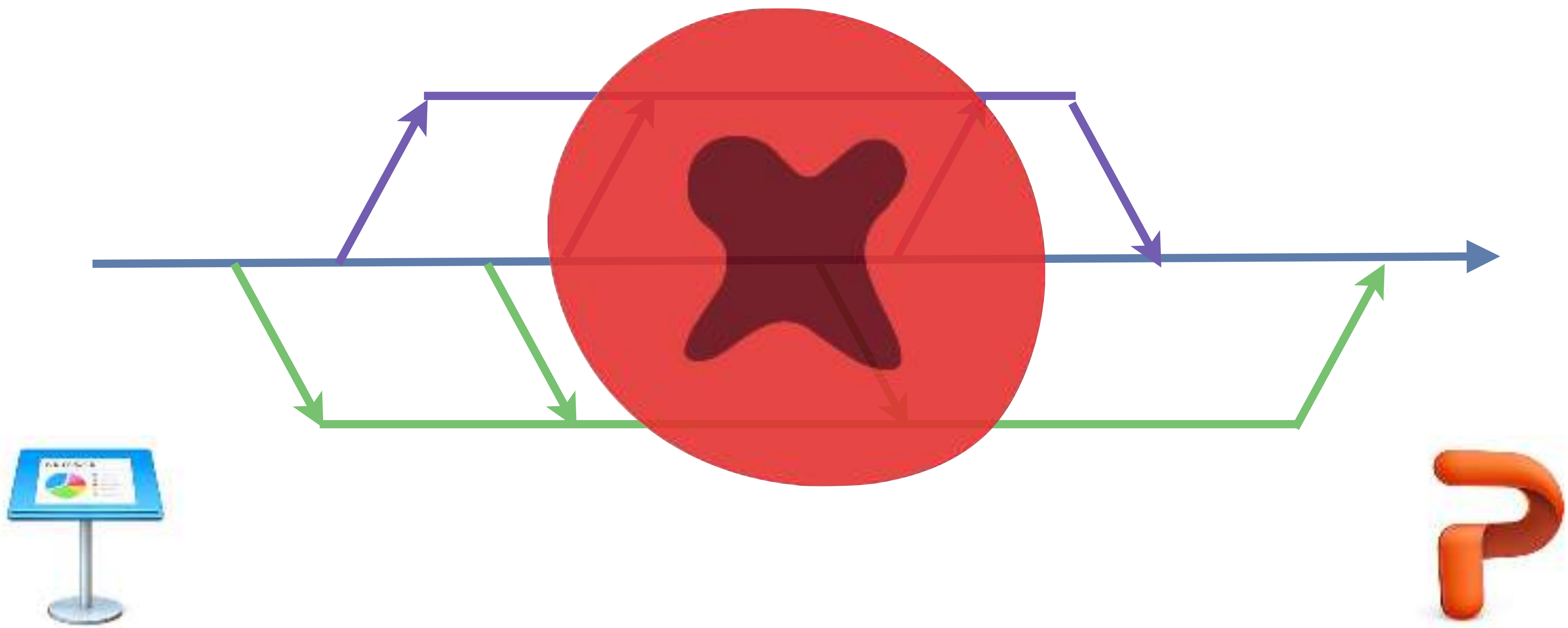
Don't feel compelled to create your presentation materials in the same order as the presentation itself.

Concurrent Creation

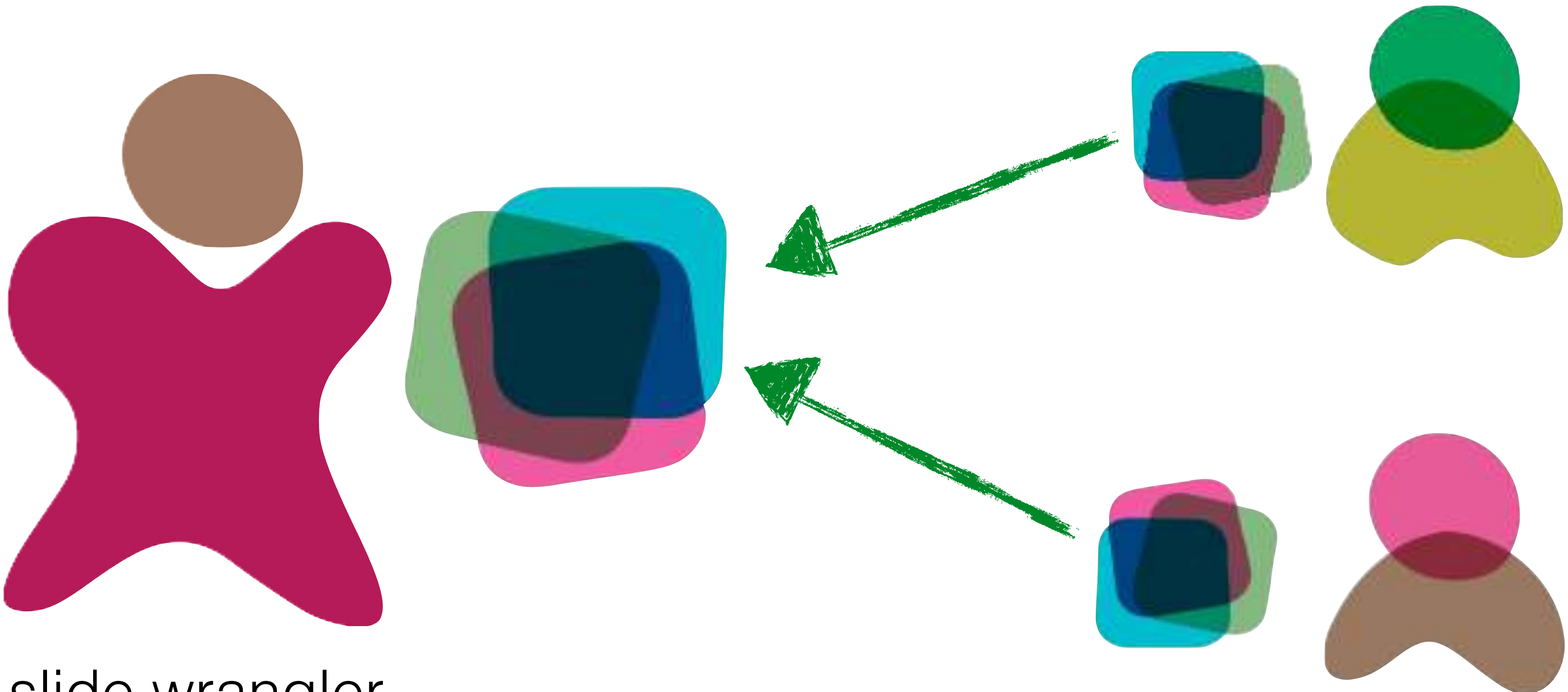
Don't feel compelled to create your presentation materials in the same order as the presentation itself.

When creating a presentation as a group, follow certain practices to retain sanity.

When creating a presentation as a group, follow certain practices to retain sanity.



Sanity-saving Practices

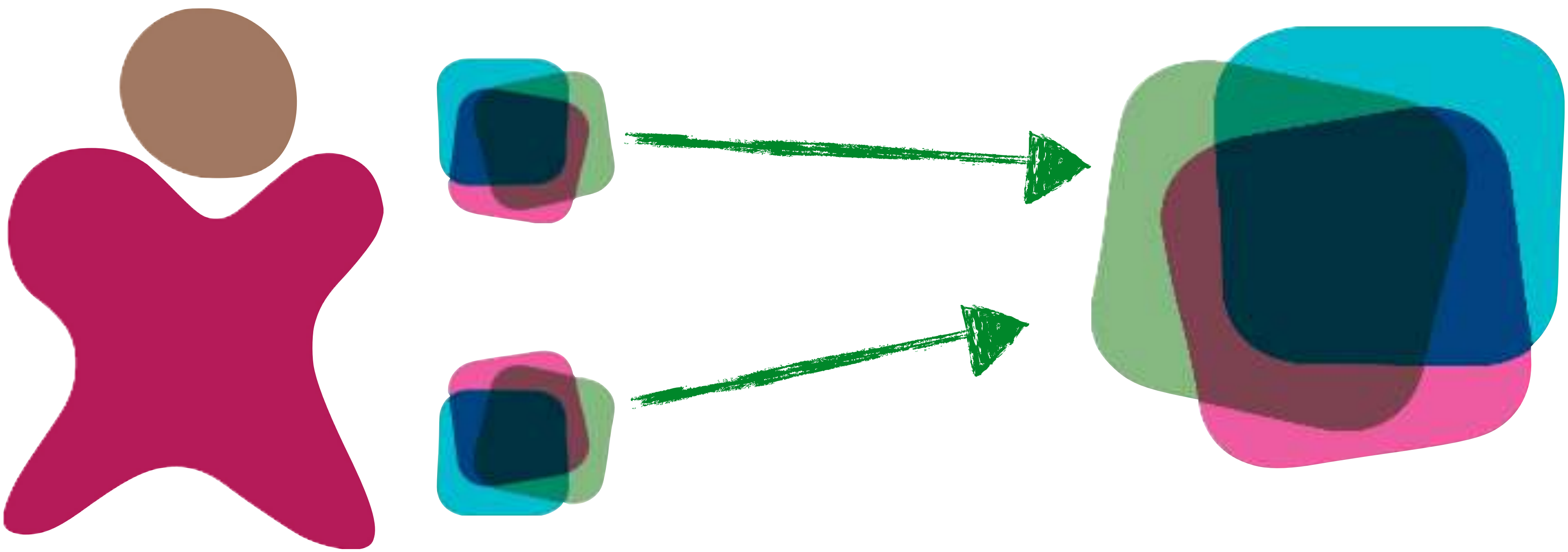


slide wrangler

Sanity-saving Practices

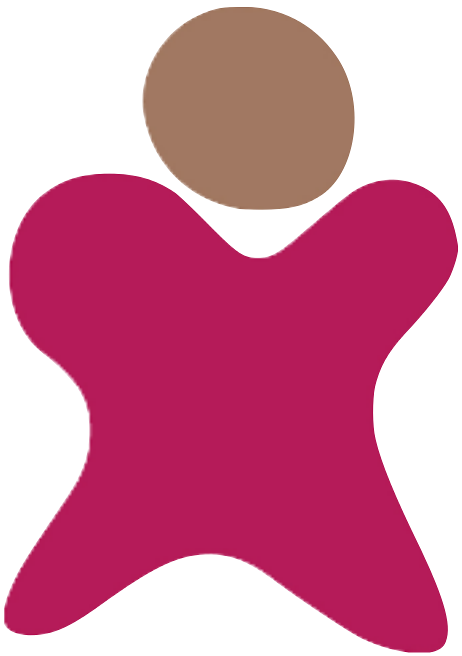
Sanity-saving Practices

Sanity-saving Practices

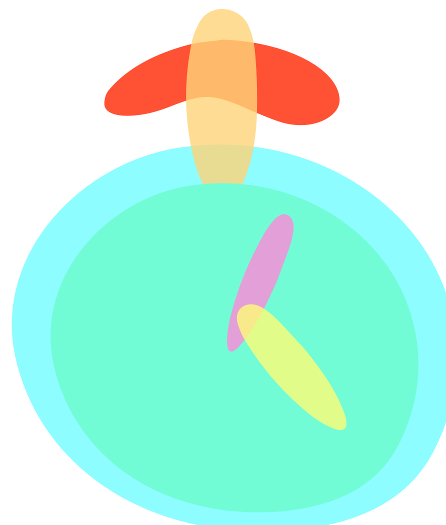


slide wrangler

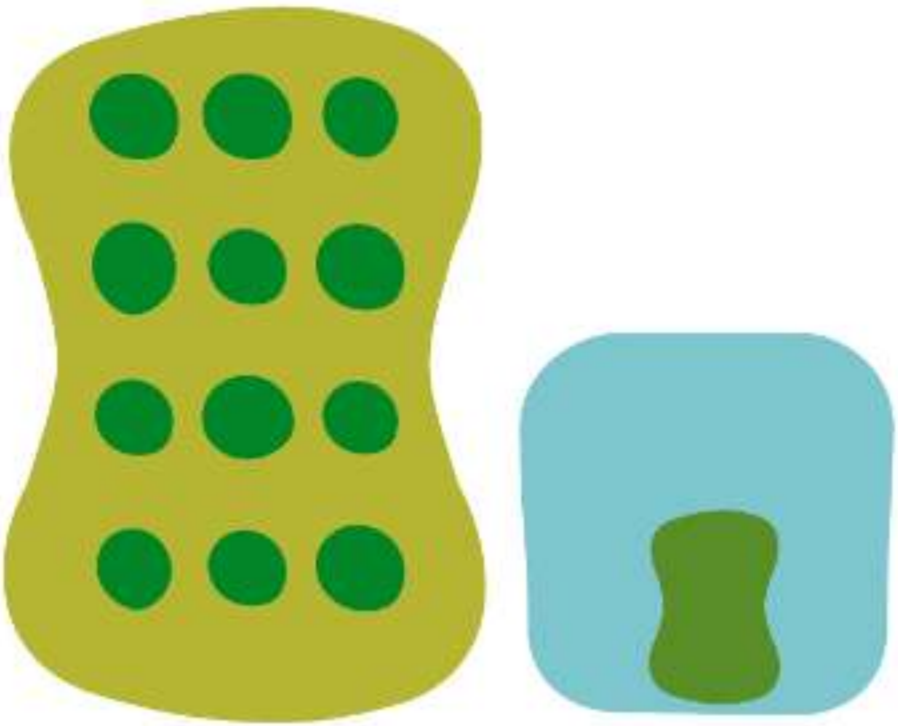
Set a Deadline (with Teeth)



slide wrangler



Known Uses

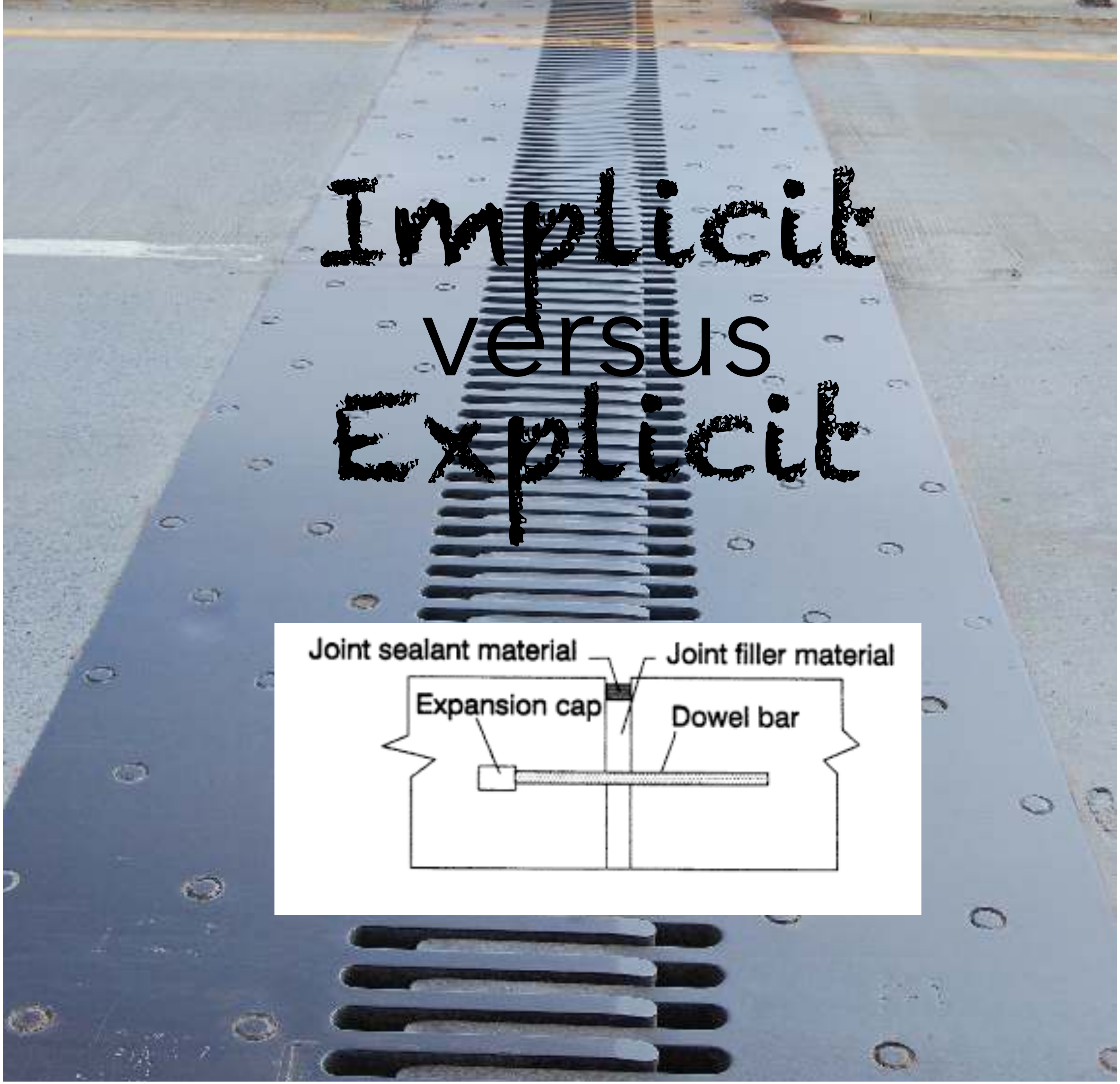


every
corporation
everywhere!

Expansion Joints

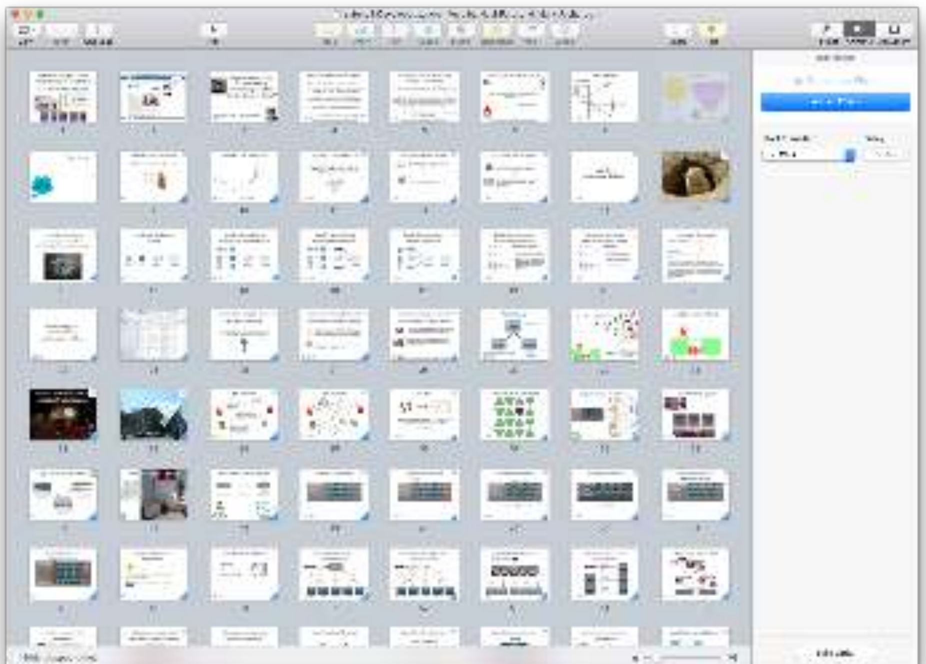
Also know as:

Goldilocks; Short, Medium, Long

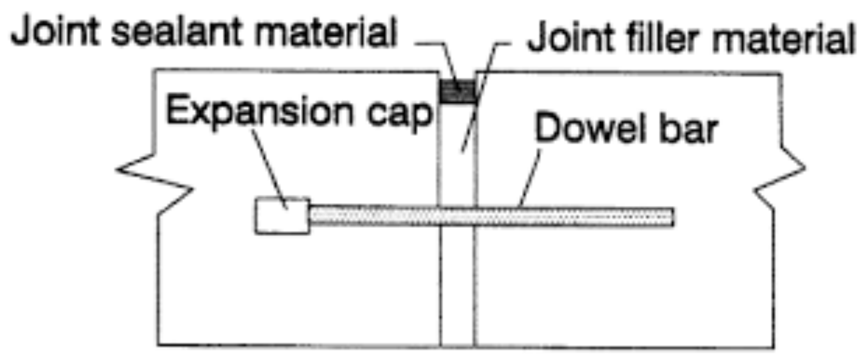

Implicit
versus
Explicit
Joint sealant material
Joint filler material
Expansion cap
Dowel bar

Implicit Expansion Joints

practice skipping gracefully



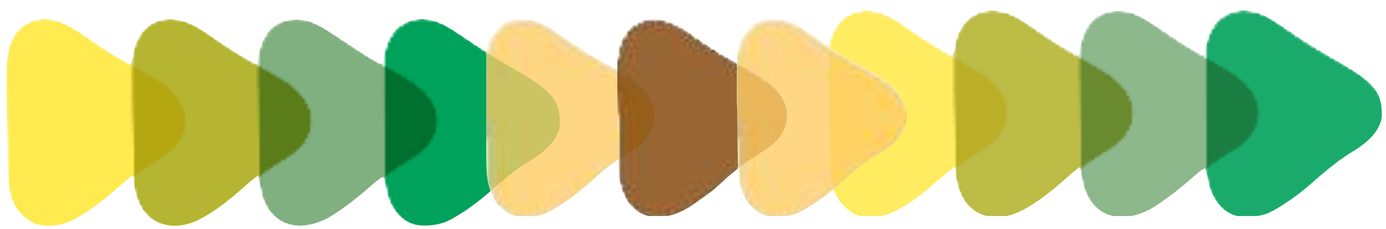
Implicit
versus
Explicit



Explicit

Expansion Joints

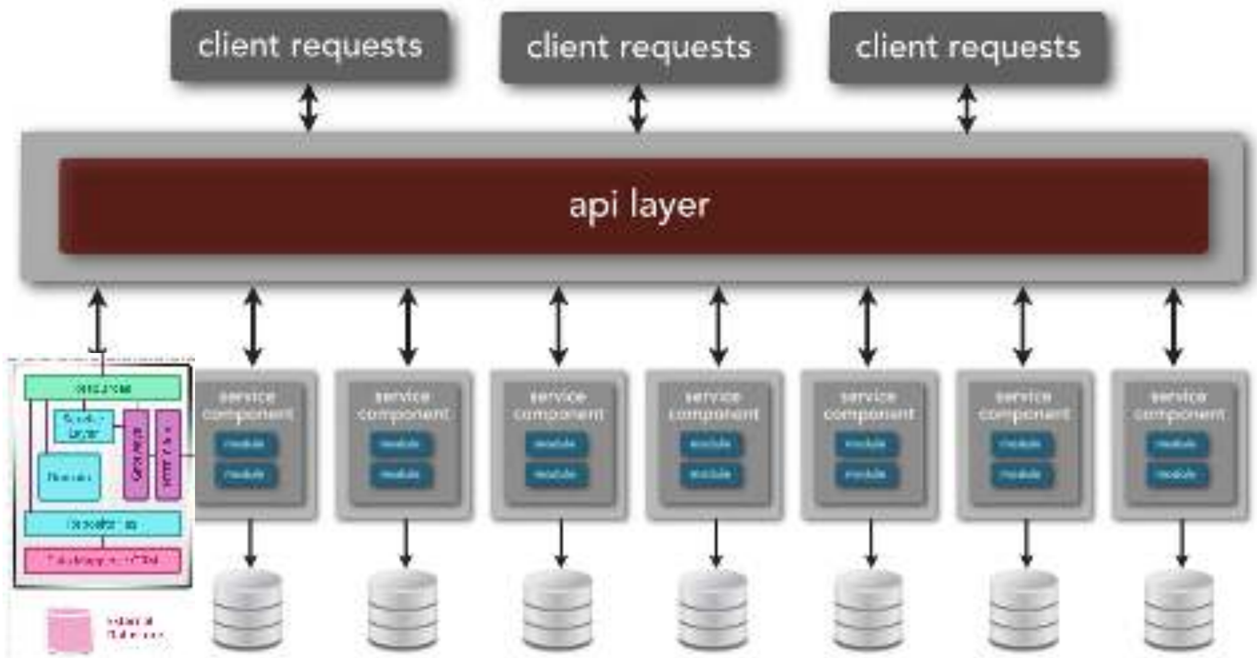
makes sense within
the Narrative Arc



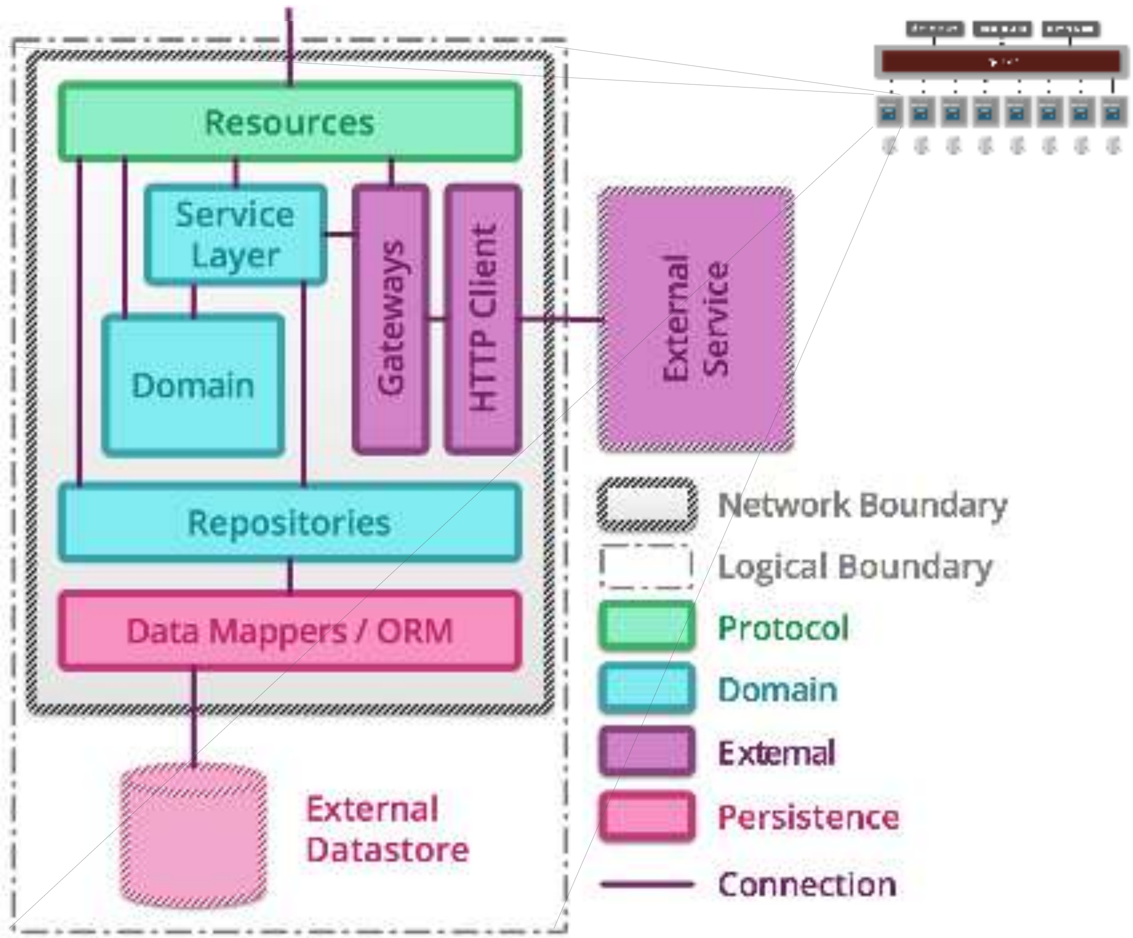
same (or closely related)
Unifying Visual Theme

multi-purpose

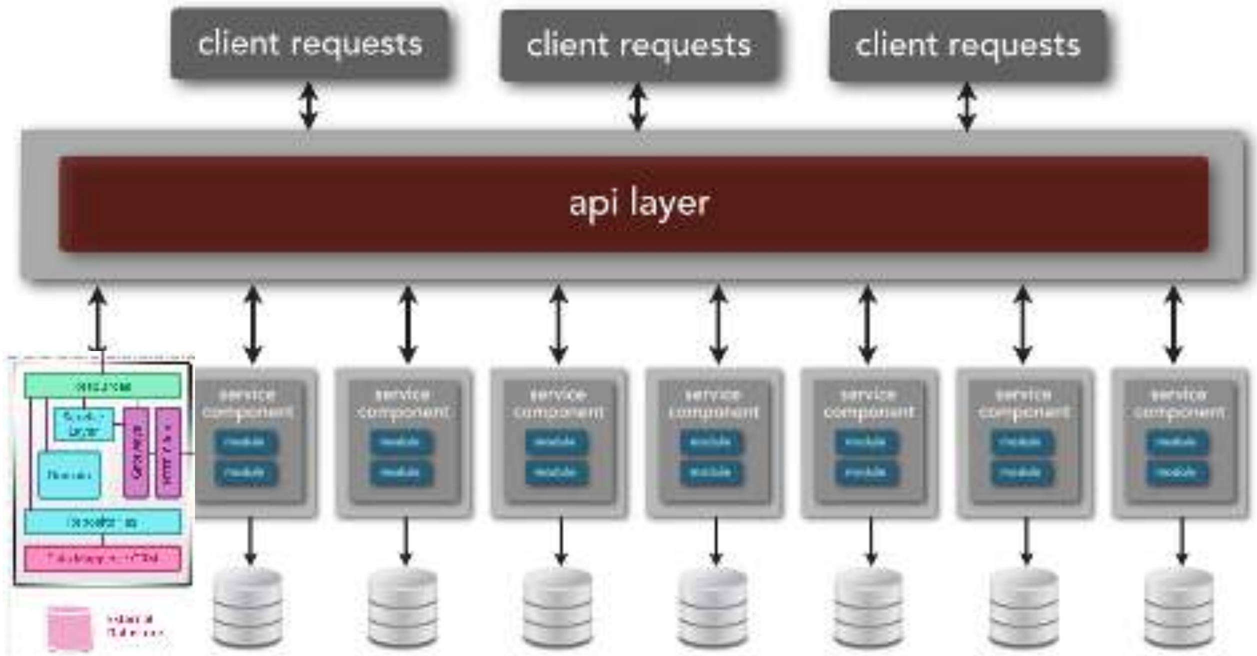
representational

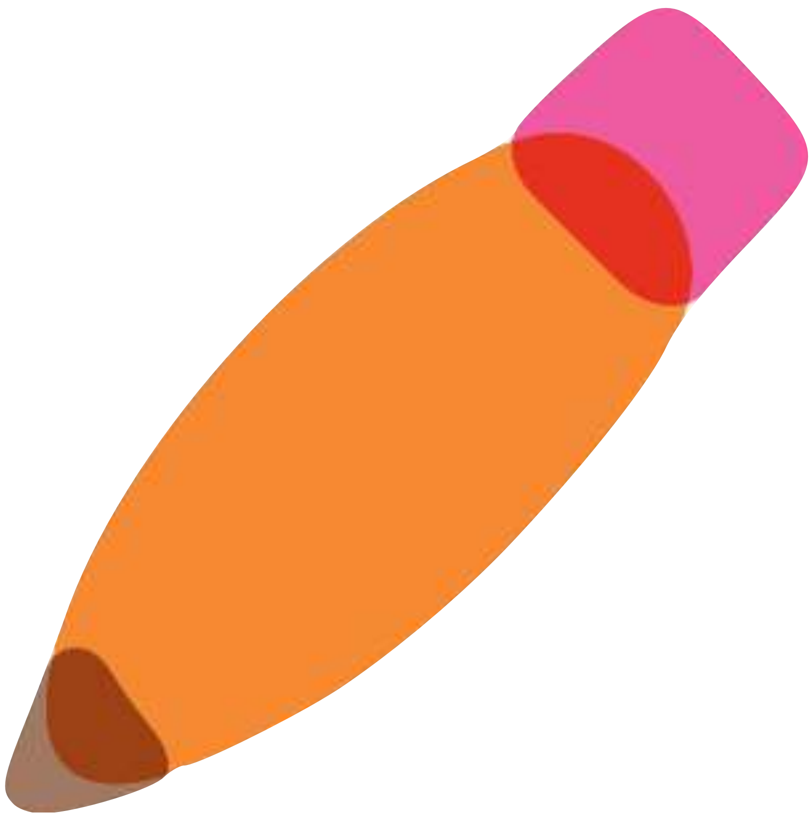


representational



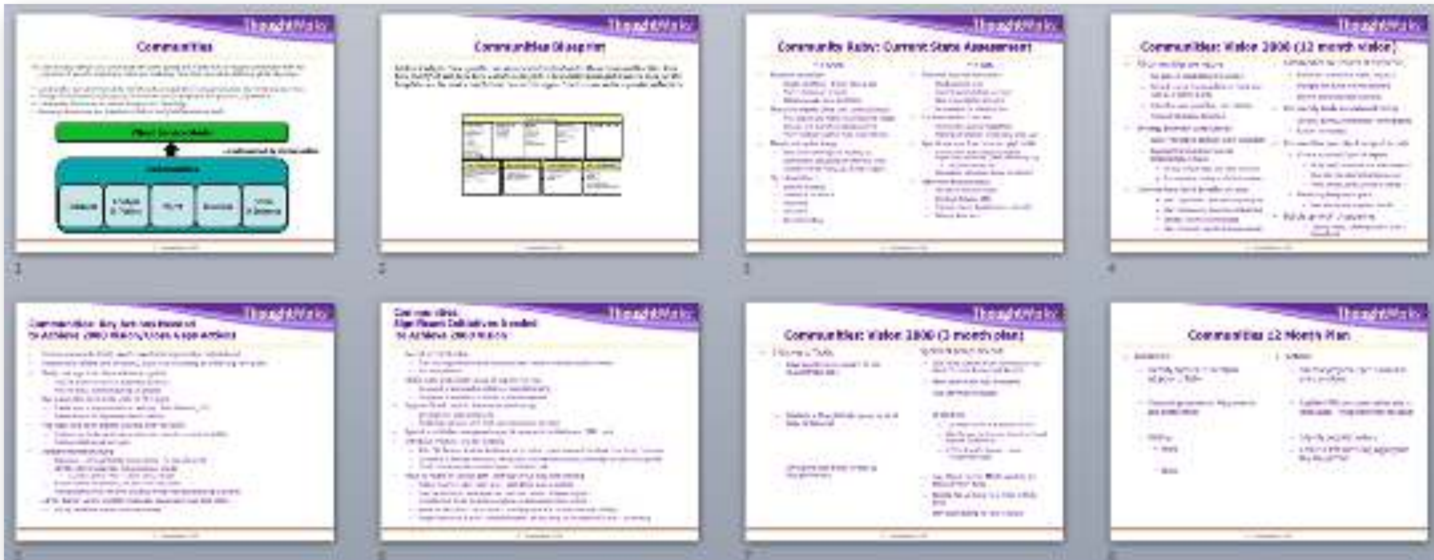
representational





creation

Cookie Cutter



~~some~~ most ideas > 1 slide

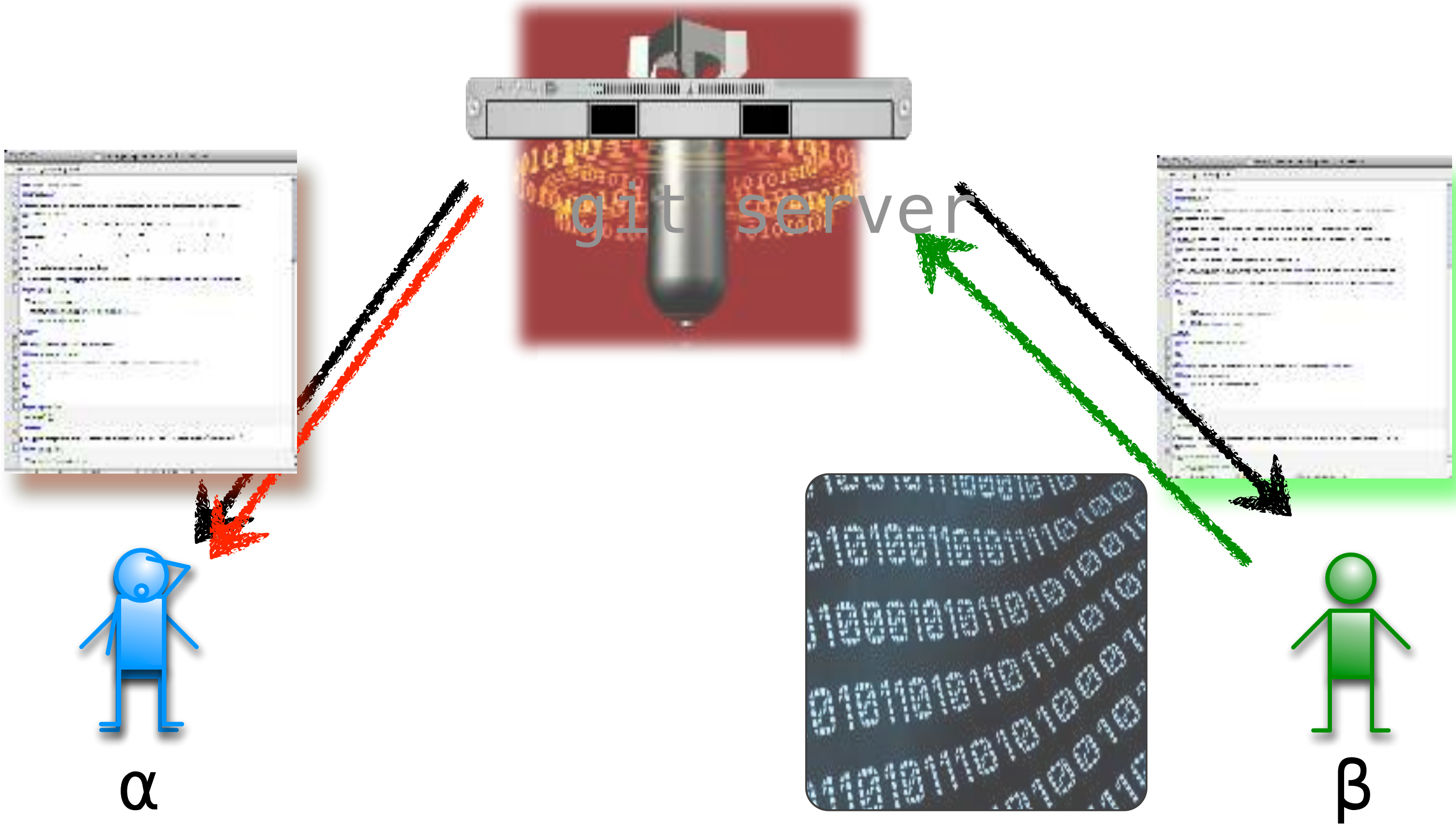
don't allow the tool to alter the message

auto-size tex

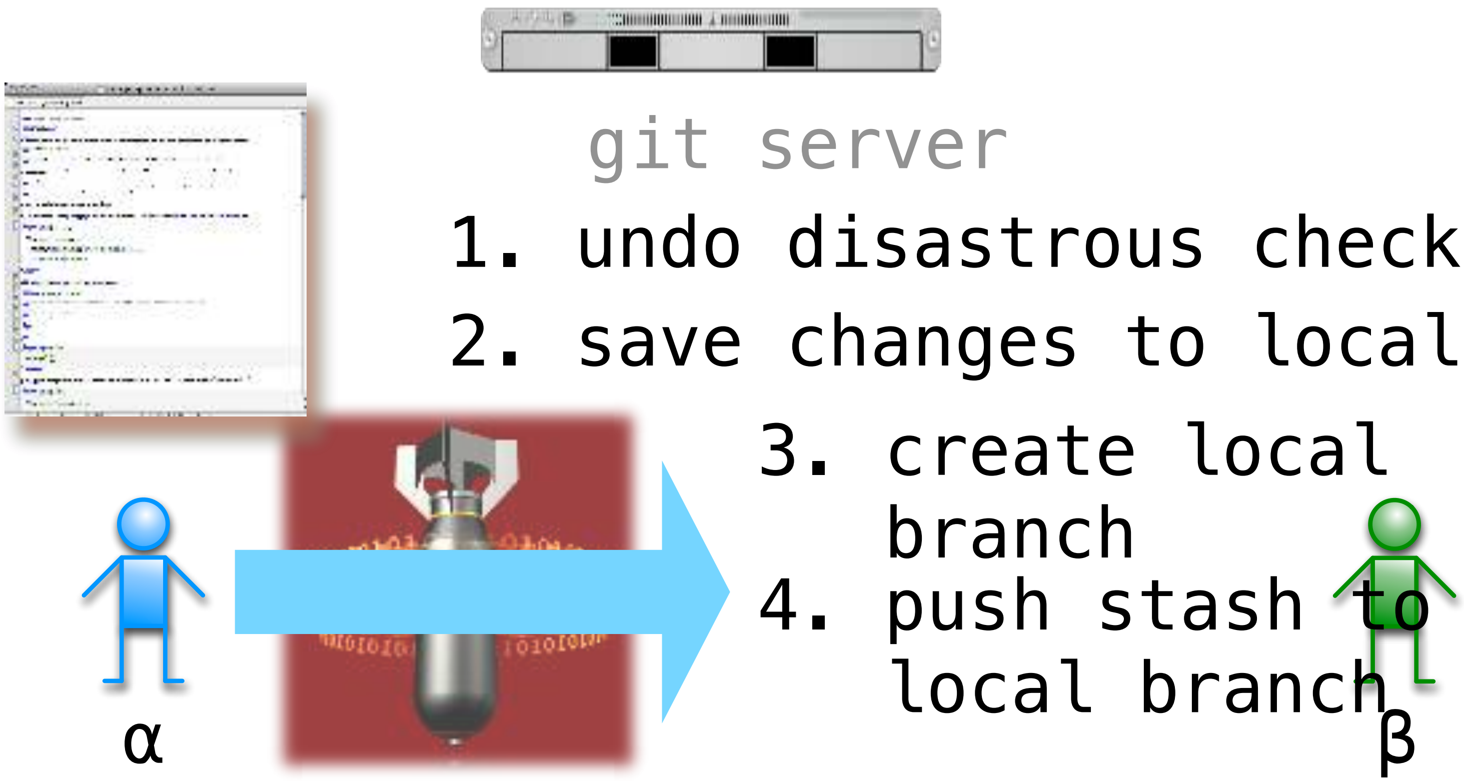
evil!

evil!

git magic



git magic

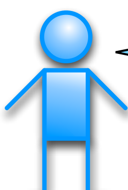


1. undo disastrous checkout
2. save changes to local stash
3. create local branch
4. push stash to local branch

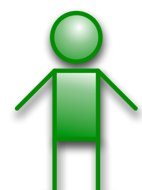
git magic

git server

5. push local branch to remote branch



6. you broke it –

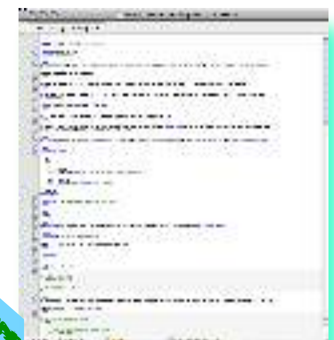


α

β

git magic

git server

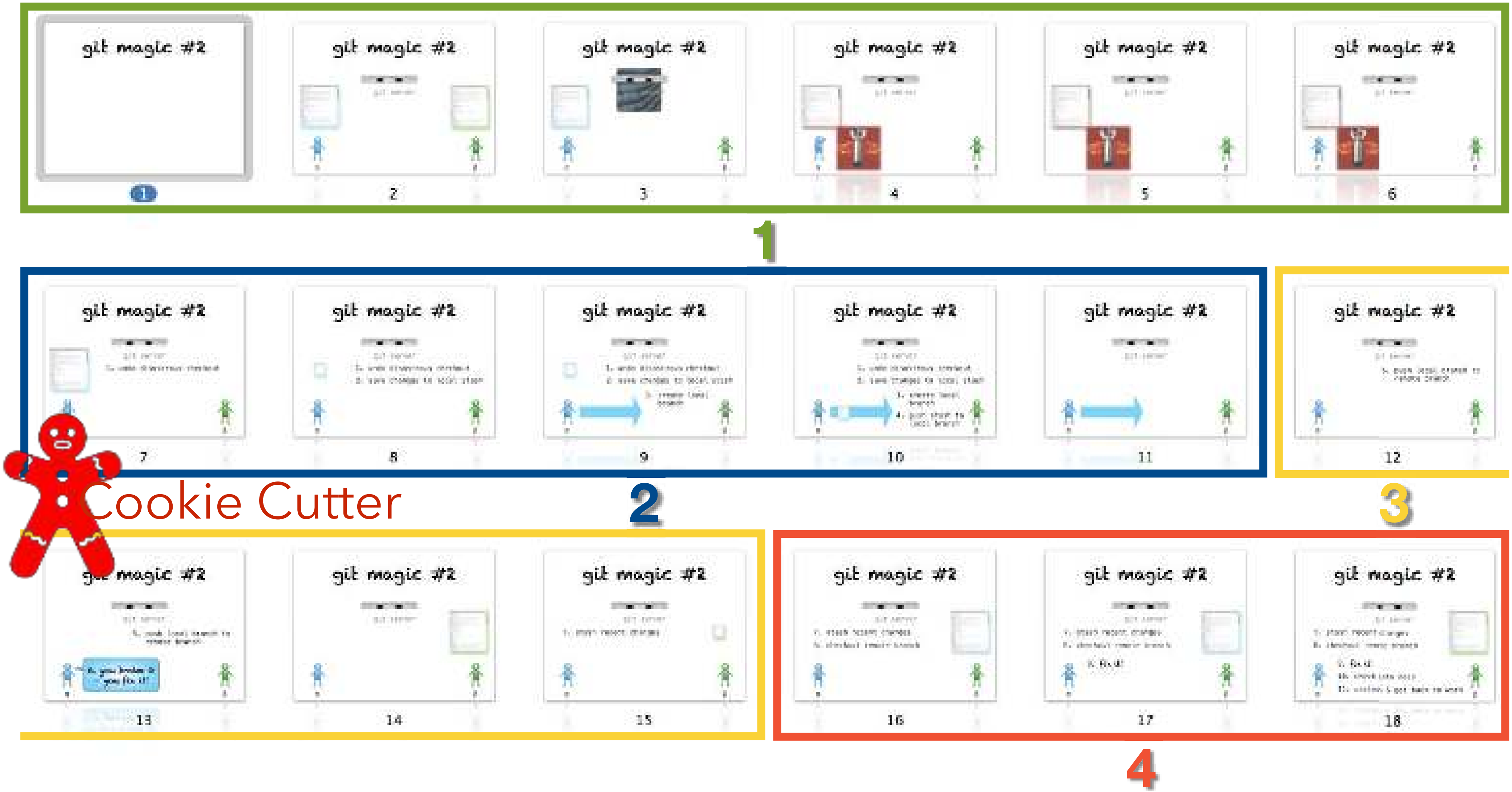


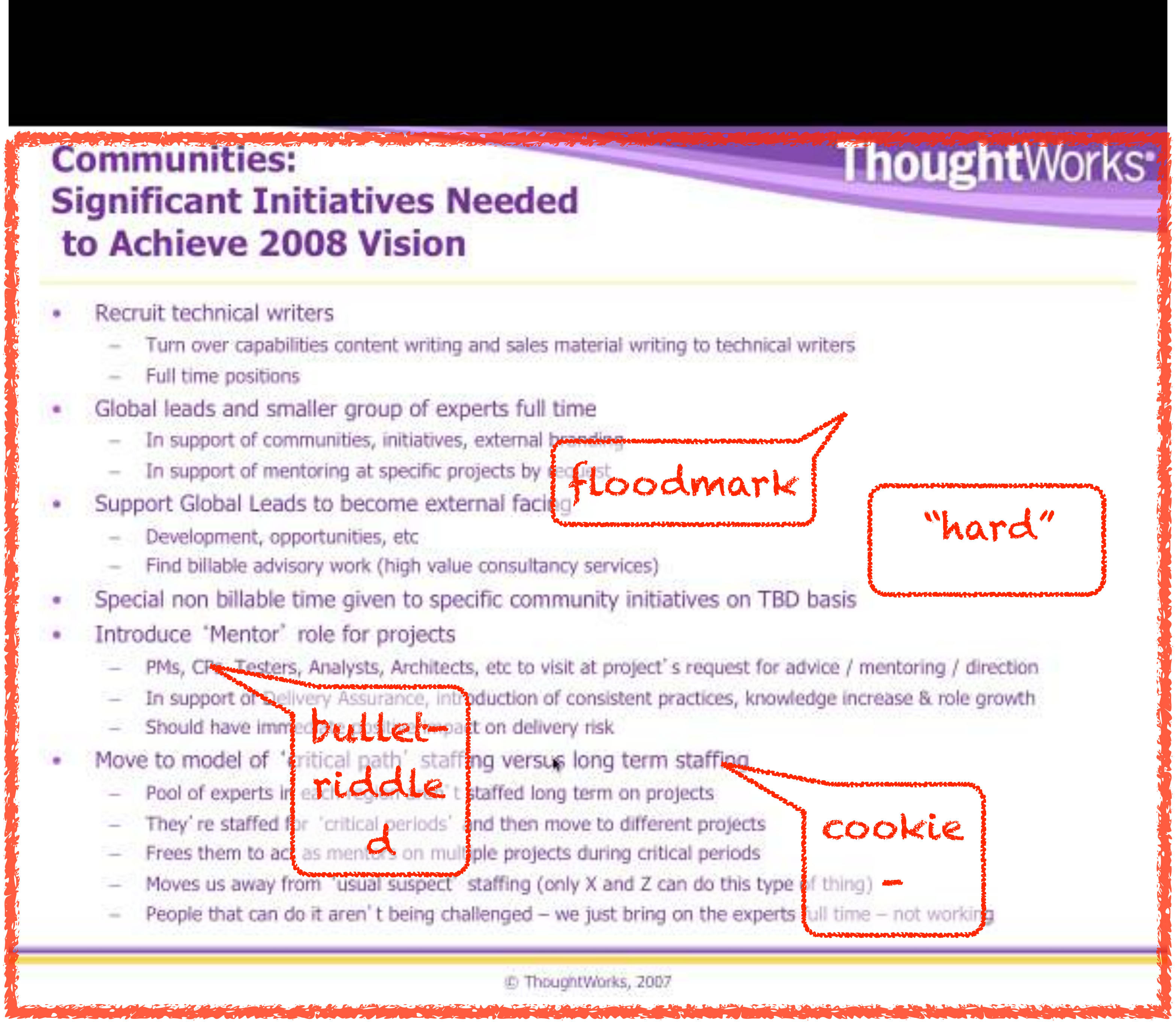
7. stash recent changes
8. checkout remote branch

9. fix it!
10. check into main
11. unstash & get back to work

α

β


git magic #2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Cookie Cutter


Communities:
Significant Initiatives Needed
to Achieve 2008 Vision
ThoughtWorks
• Recruit technical writers
– Turn over capabilities content writing and sales material writing to technical writers
– Full time positions
• Global leads and smaller group of experts full time
– In support of communities, initiatives, external branding
– In support of mentoring at specific projects by request
• Support Global Leads to become external facing
– Development, opportunities, etc
– Find billable advisory work (high value consultancy services)
• Special non billable time given to specific community initiatives on TBD basis
• Introduce 'Mentor' role for projects
– PMs, CPs, Testers, Analysts, Architects, etc to visit at project's request for advice / mentoring / direction
– In support of Delivery Assurance, introduction of consistent practices, knowledge increase & role growth
– Should have imm... t on delivery risk
• Move to model of 'critical path' staffing versus long term staffing
– Pool of experts in ... t staffed long term on projects
– They're staffed for 'critical periods' and then move to different projects
– Frees them to act as mentors on multiple projects during critical periods
– Moves us away from 'usual suspect' staffing (only X and Z can do this type of thing)
– People that can do it aren't being challenged – we just bring on the experts full time – not working
© ThoughtWorks, 2007
floodmark
"hard"
bullet-
riddle
d
cookie
-

Why So Many Bullets?

- Both presenters and audience expect it.
- Title + Bullets is often the default template.
- Inexperienced speaker's rely on bullets as speaker notes
- Easy to bang together in a conference room while > 4 people are talking

When a Slide Full of Text Appears

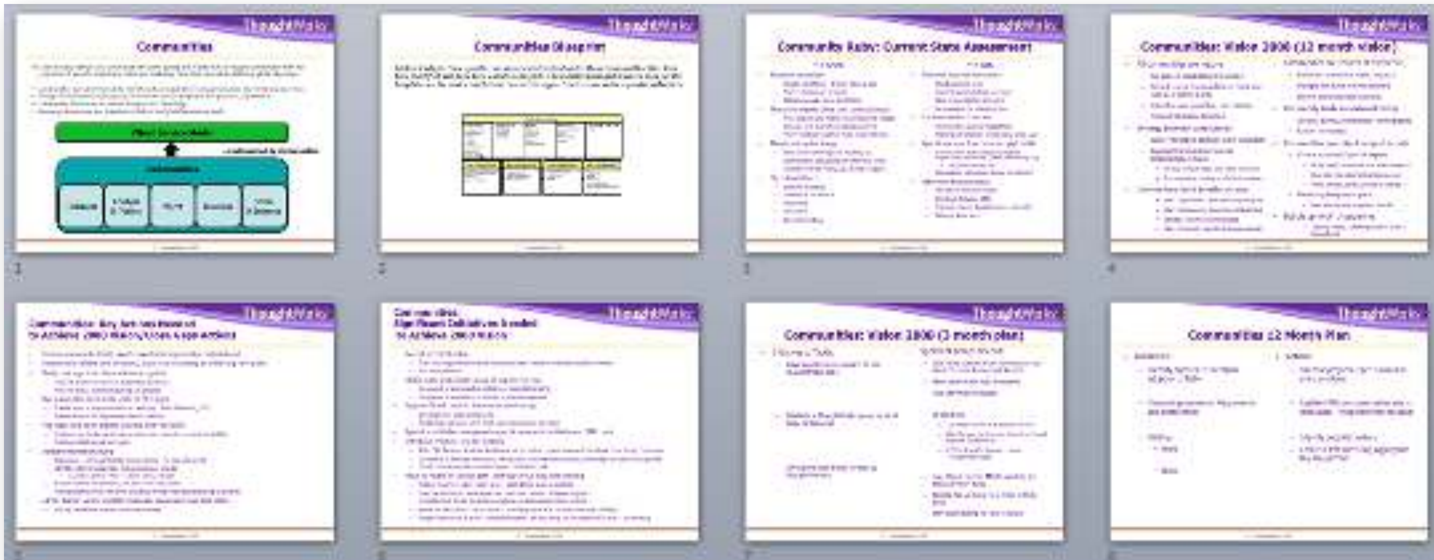
- Everyone in the audience
- Reads the entire thing right away
- You can’t help it
- Now, the presenter spends the next five minutes
- slowly reading what you’ve already read

Bullet Riddled Corps

When showing a slide, pause for a moment to allow the audience to read it before you continue.

– Multiple terrible presentation guides

Cookie Cutter



most

some ideas > 1 slide

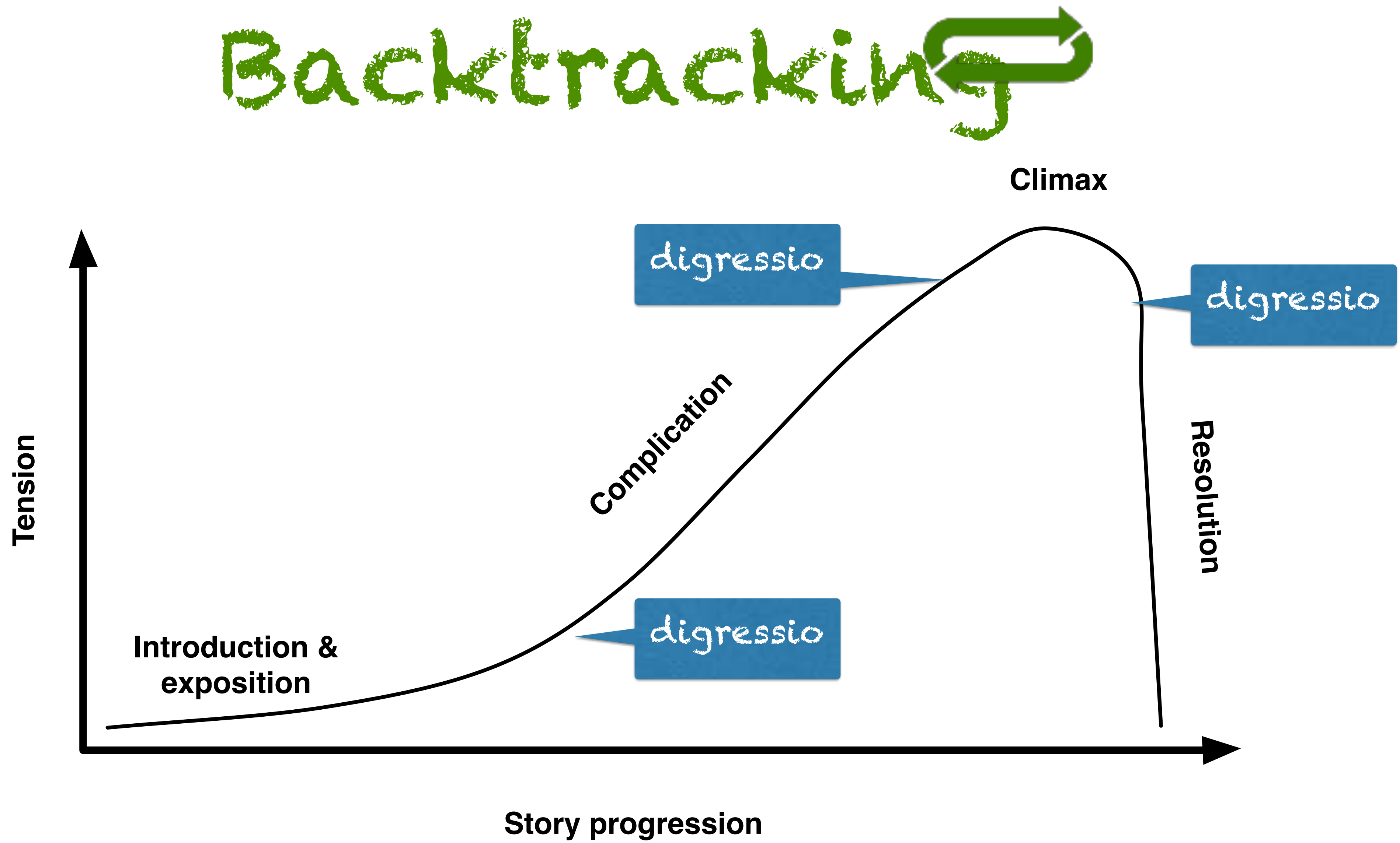
don't allow the tool to alter the message

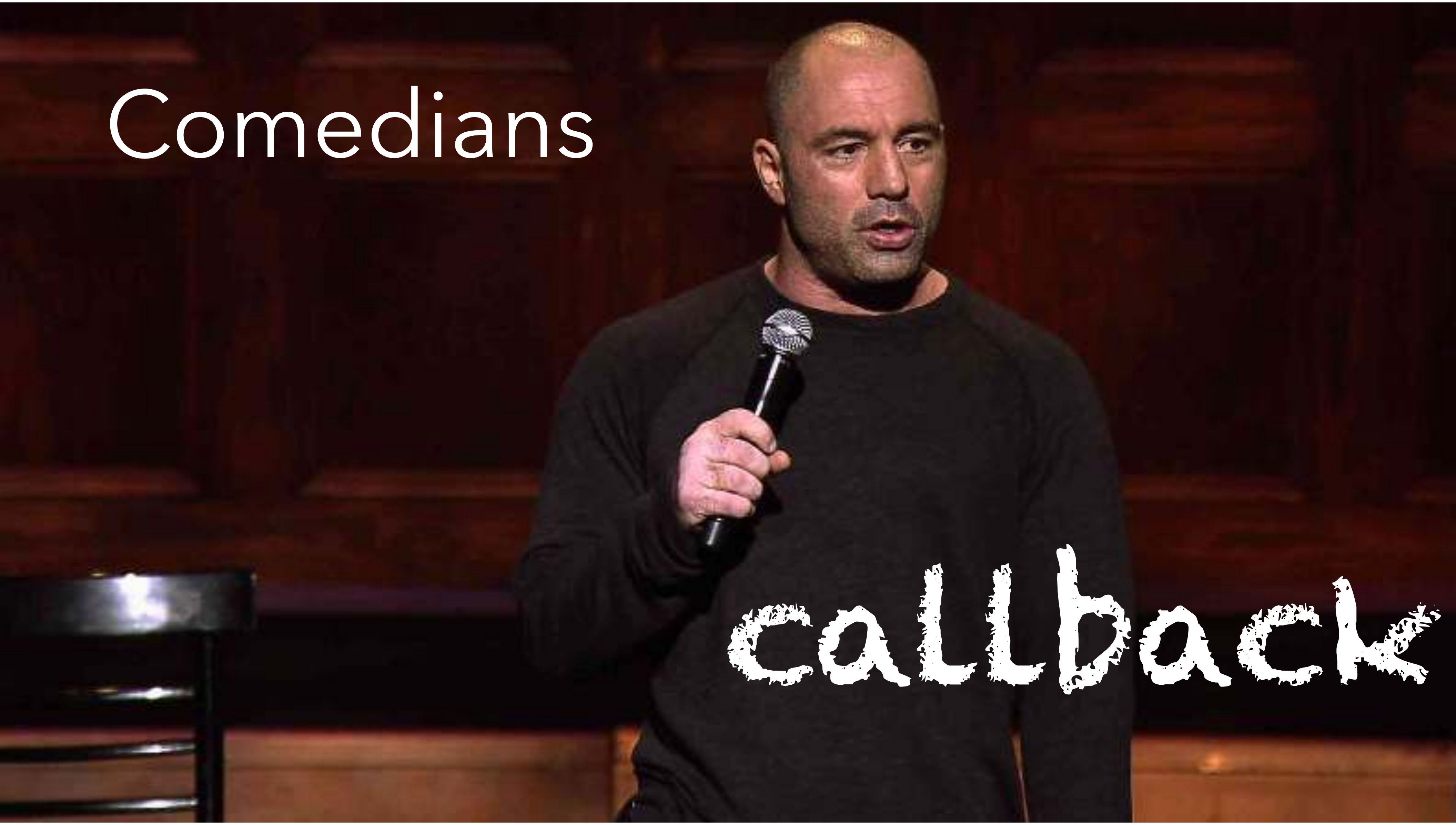
evil!

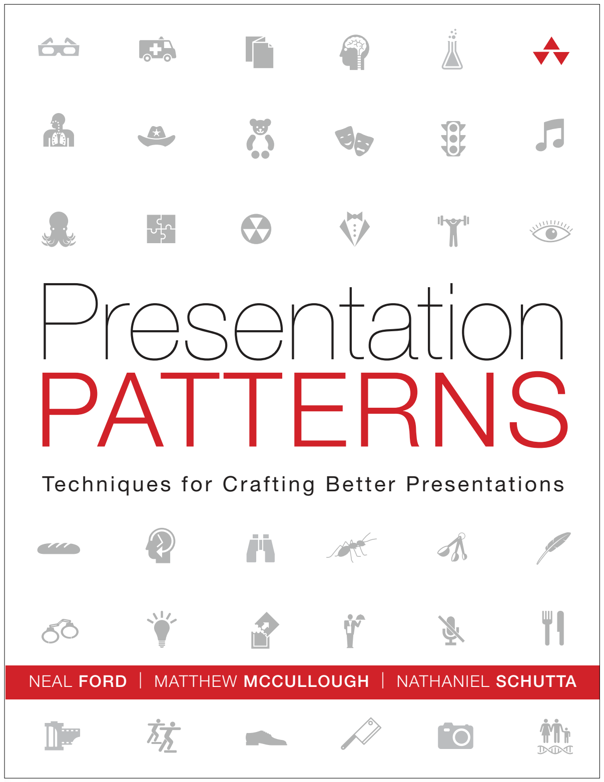
Backtracking

auto-size tex

evil!


Backtracking
Climax
digressio
digressio
Complication
Tension
Resolution
Introduction &
exposition
digressio
Story progression


Comedians
callback



Presentation Patterns

Building Blocks for Perfect Presentations

Slide Creation Patterns:

Floodmarks

Also know as:

Death by Advertising, Marketing Mania, Kudzu Log

Floodmarks

Floodmarks represent extraneous background imagery featured on every slide.

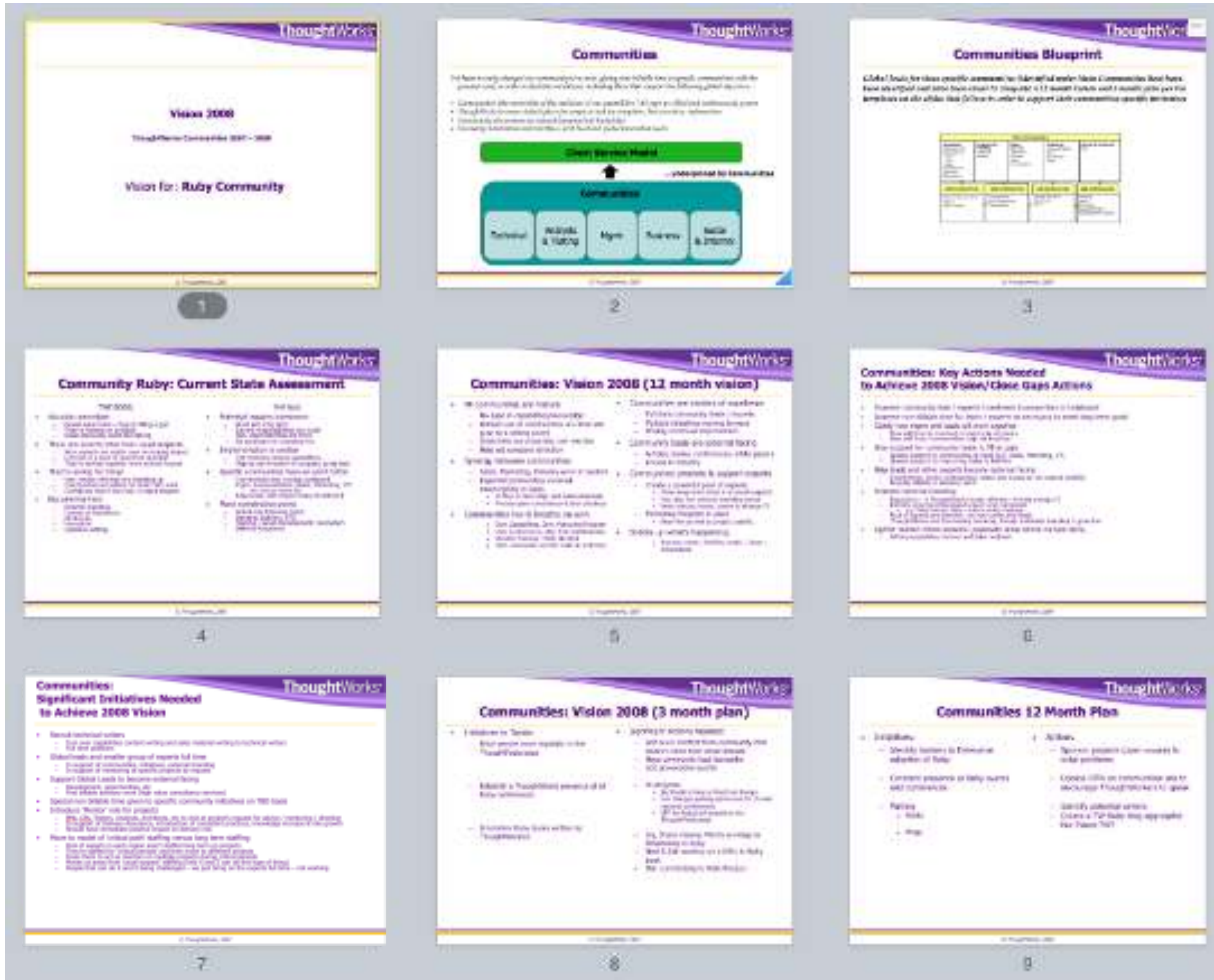
Building Web Services with Java

Neal Ford

CTO, The DSW Group, Ltd.

Tuesday, September 10, 2002

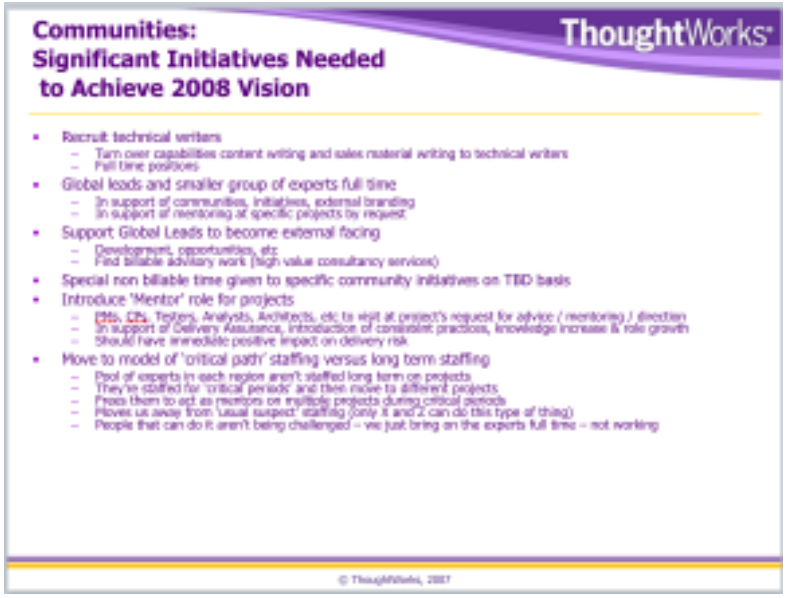
Floodmarks as Constraint



Communities: Significant Initiatives Needed to Achieve 2008 Vision

ThoughtWorks®

- Recruit technical writers
  - Turn over capabilities content writing and sales material writing to technical writers
  - Full time positions
- Global leads and smaller group of experts full time
  - In support of communities, initiatives, external branding
  - In support of mentoring at specific projects by request
- Support Global Leads to become external facing
  - Development, opportunities, etc
  - Find billable advisory work (high value consultancy services)
- Special non billable time given to specific community initiatives on TBD basis
- Introduce 'Mentor' role for projects
  - PMs, CPs, Testers, Analysts, Architects, etc to visit at project's request for advice / mentoring / direction
  - In support of Delivery Assurance, introduction of consistent practices, knowledge increase & role growth
  - Should have immediate positive impact on delivery risk
- Move to model of 'critical path' staffing versus long term staffing
  - Pool of experts in each region aren't staffed long term on projects
  - They're staffed for 'critical periods' and then move to different projects
  - Frees them to act as mentors on multiple projects during critical periods
  - Moves us away from 'usual suspect' staffing (only X and Z can do this type of thing)
  - People that can do it aren't being challenged – we just bring on the experts full time – not working



- Floodmark artificially compresses the headline.

- Lines at top and bottom artificially constrain information space.

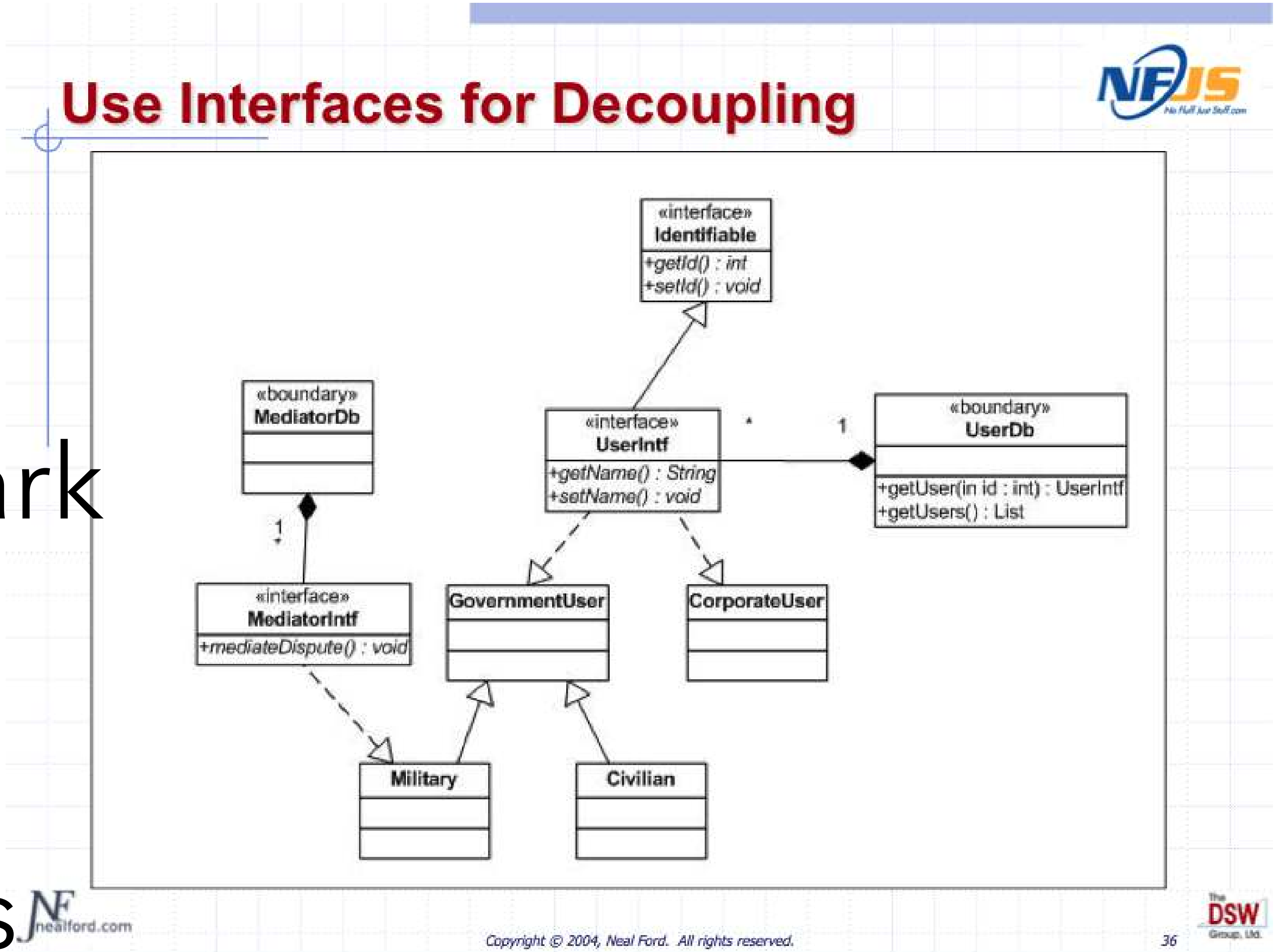
  - What happens to large images?

  - Makes slide transitions obvious

- Copyright and company logo on each slide negates intended effect.

Charred Trail

Floodmarks
+
Images



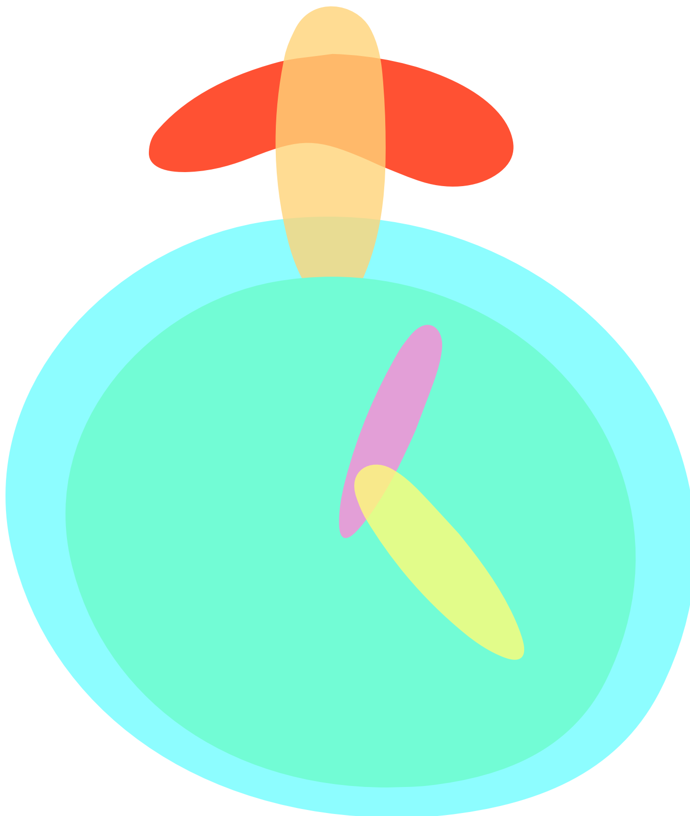
Should You Ever Floodmark?

Bookend slides

Anytime you want to *remind* your audience of branding

At the end

temporal



Presentation

Presenter controls exposition rate.

Infodeck

Reader controls exposition rate.

Slideument

MARTIN FOWLER 1/25

ThoughtWorks

Testing Strategies in a Microservice Architecture

There has been a shift in service based architectures over the last few years towards smaller, more focussed "micro" services. There are many benefits with this approach such as the ability to independently deploy, scale and maintain each component and parallelize development across multiple teams. However, once these additional network partitions have been introduced, the testing strategies that applied for monolithic in process applications need to be reconsidered.

Here, we plan to discuss a number of approaches for managing the additional testing complexity of multiple independently deployable components as well as how to have tests and the application remain correct despite having multiple teams each acting as guardians for different services.

18 November 2014

Toby Clemson is a developer at ThoughtWorks with a passion for building large scale distributed business systems. He has worked on projects in four continents and is currently based in New York.

My thanks to **Martin Fowler** for his continued support in compiling this infodeck. Thanks also to **Danilo Sato, Dan Coffman, Steven Lowe, Chris Ford, Mark Taylor, Praful Todkar, Sam Newman** and **Marcos Matos** for their feedback and contributions.

Hints for using this deck

http://martinfowler.com/articles/microservice-testing

Best Compromise Slideument

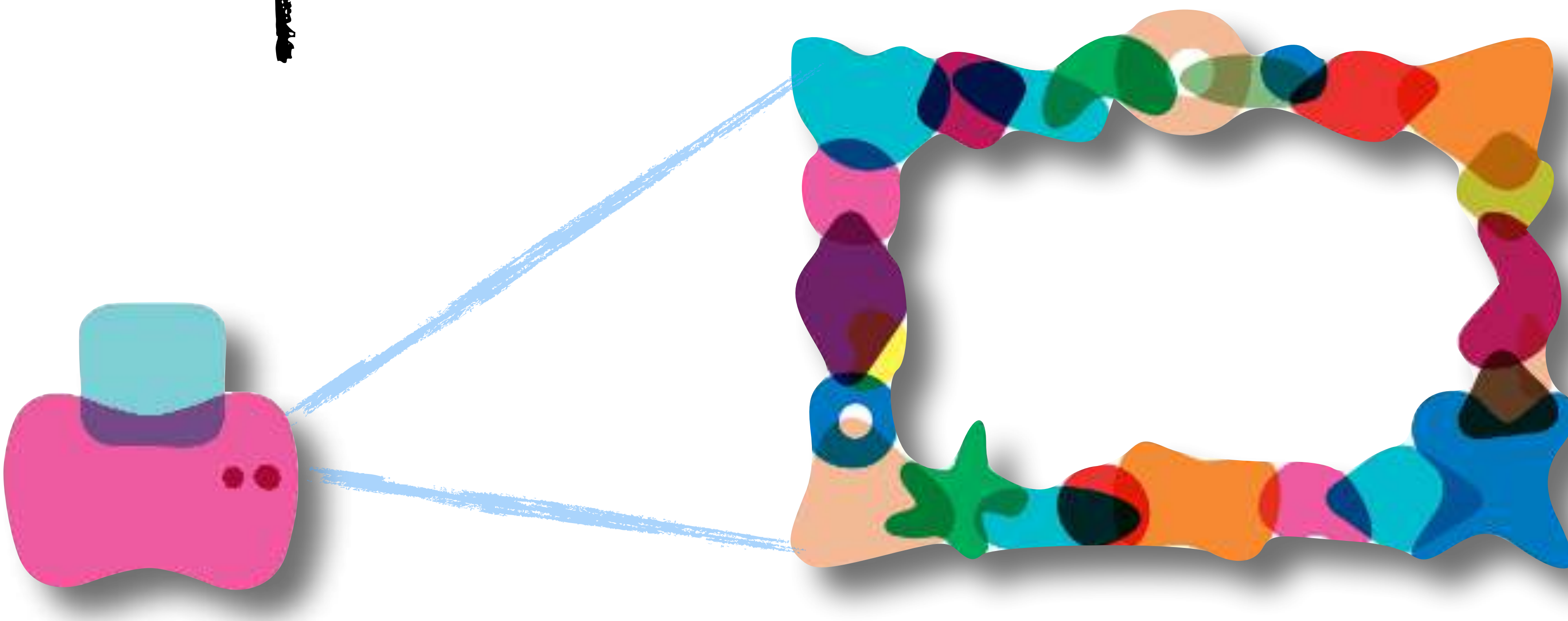
Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and

Problems:

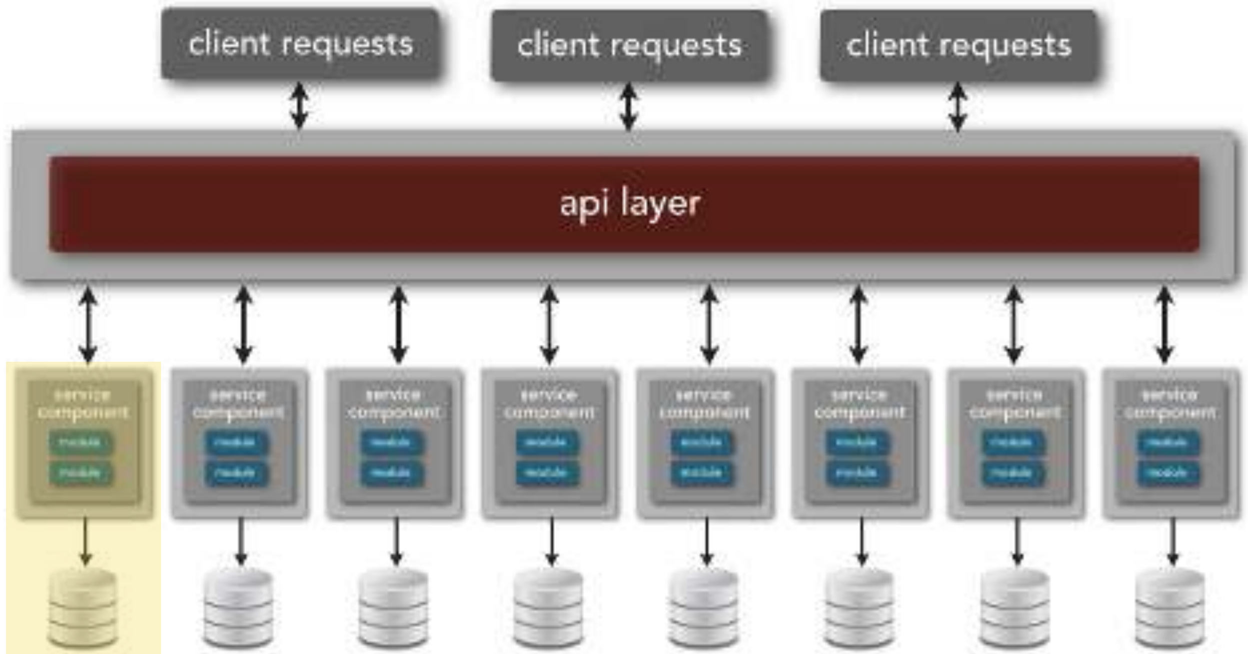
- mechanically difficult to write prose in speaker's notes
- people tends towards Bullet Riddled Corpse summaries

(this slide intentionally left blank)

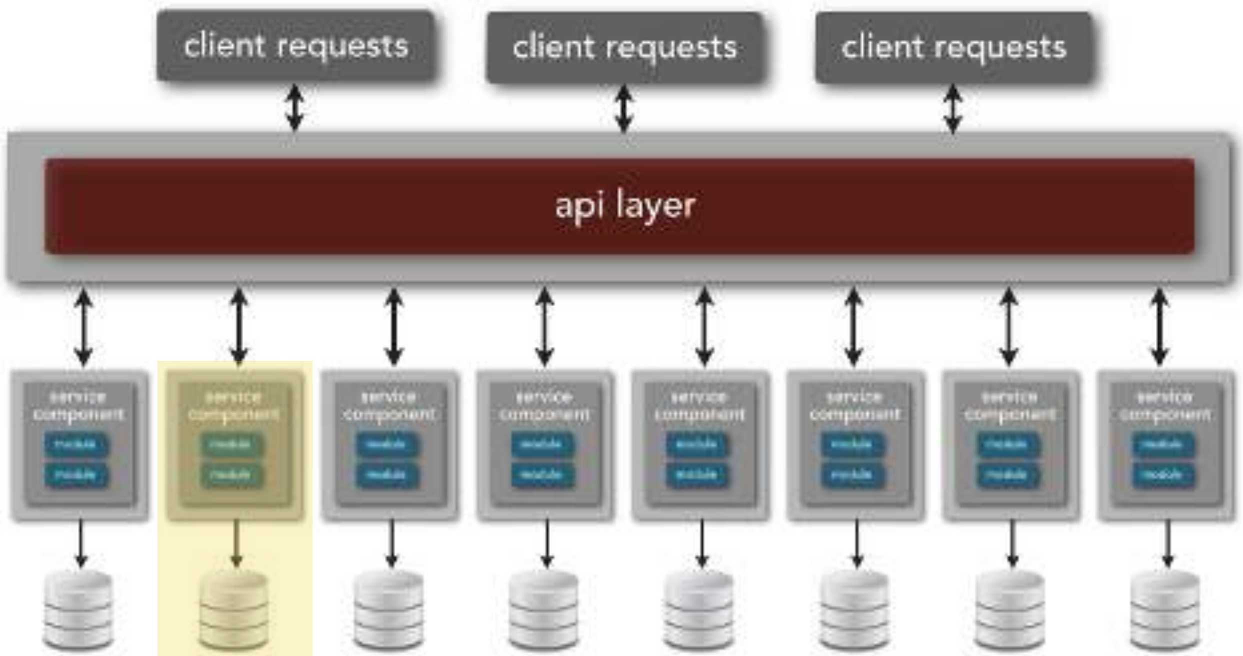
demos vs presentations



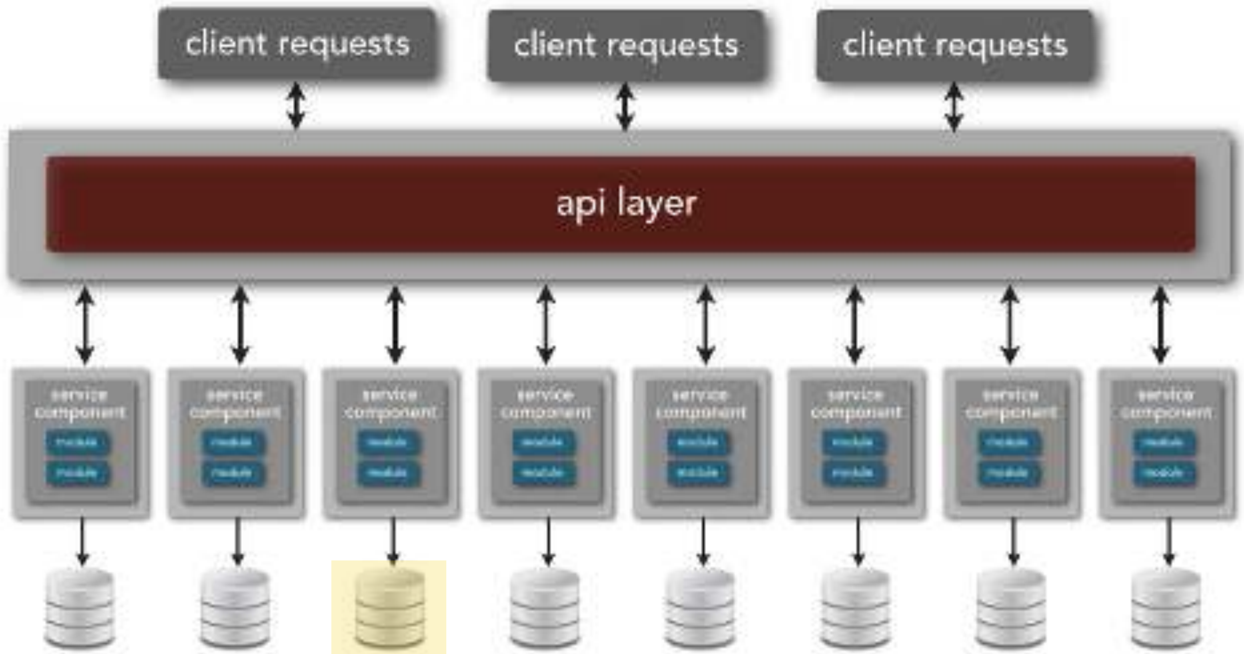
traveling highlights



traveling highlights

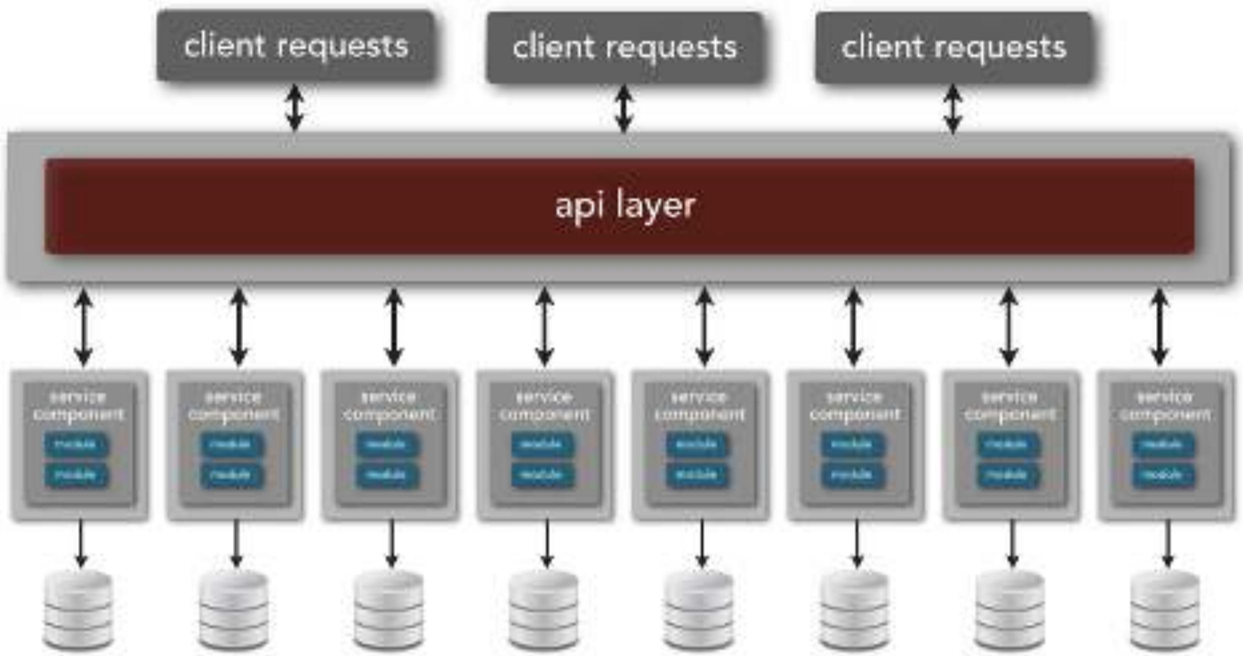


traveling highlights

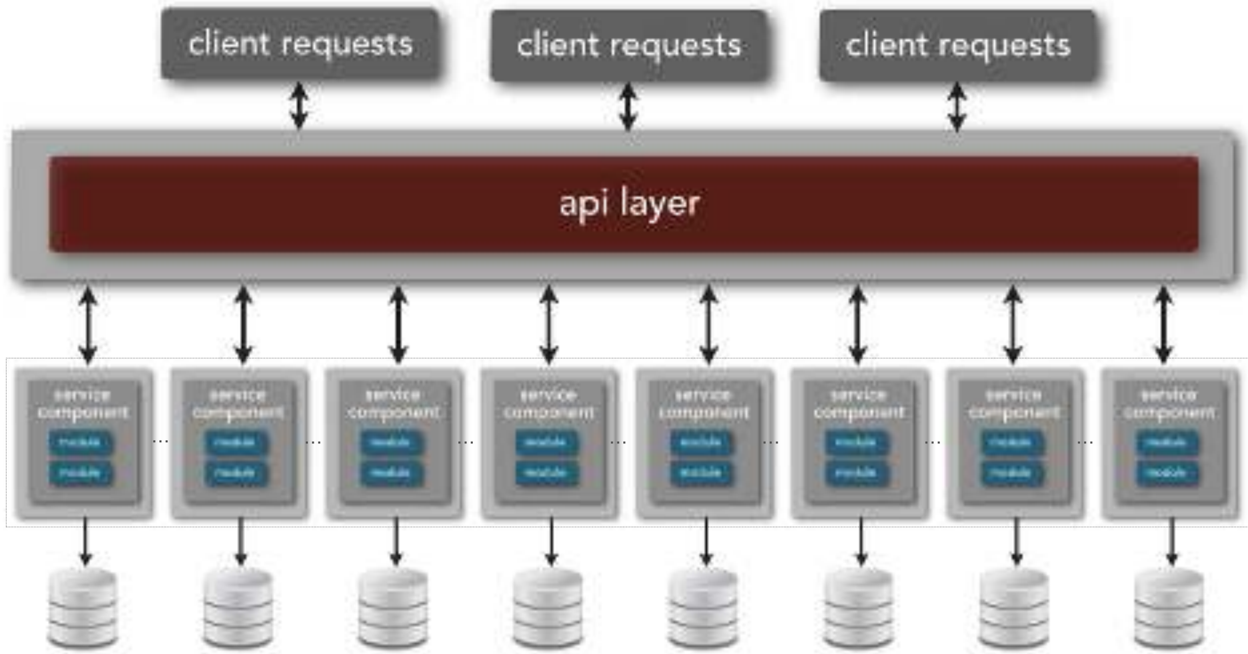



traveling highlights
client requests
client requests
client requests
api layer

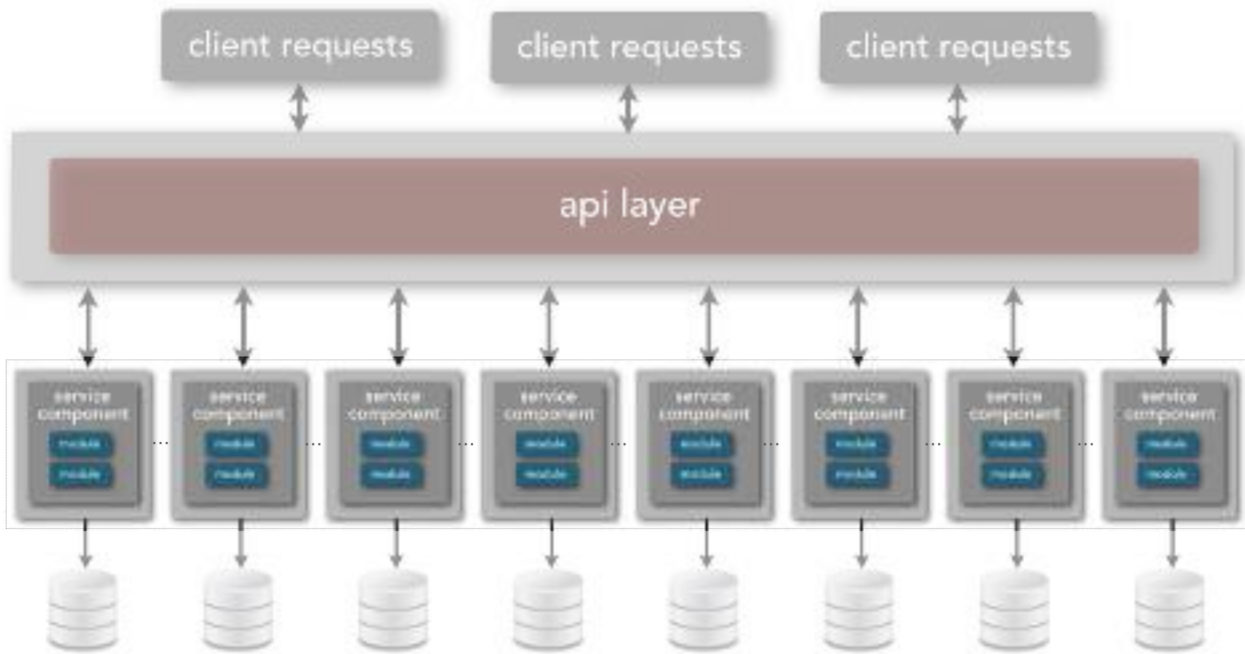
dimensions


client requests
client requests
client requests
api layer
service component
service component
service component
service component
service component
service component
service component
service component

dimensions

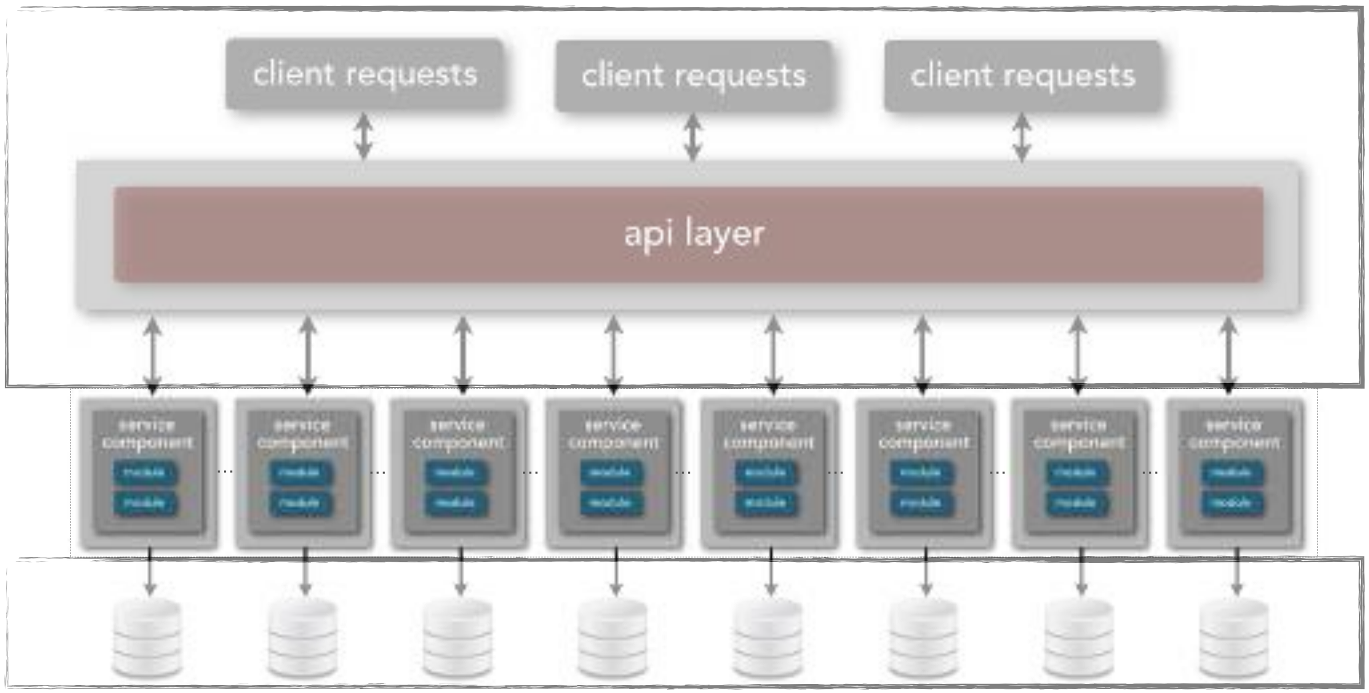

client requests
client requests
client requests
api layer

dimensions

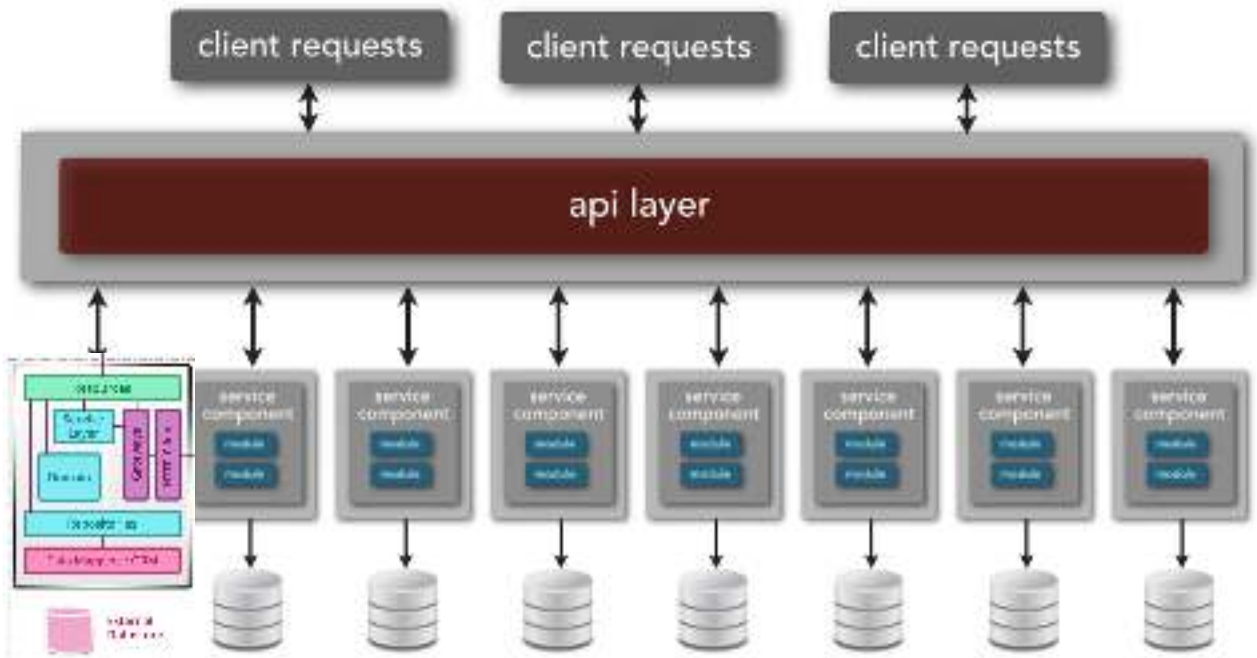


opacity shift

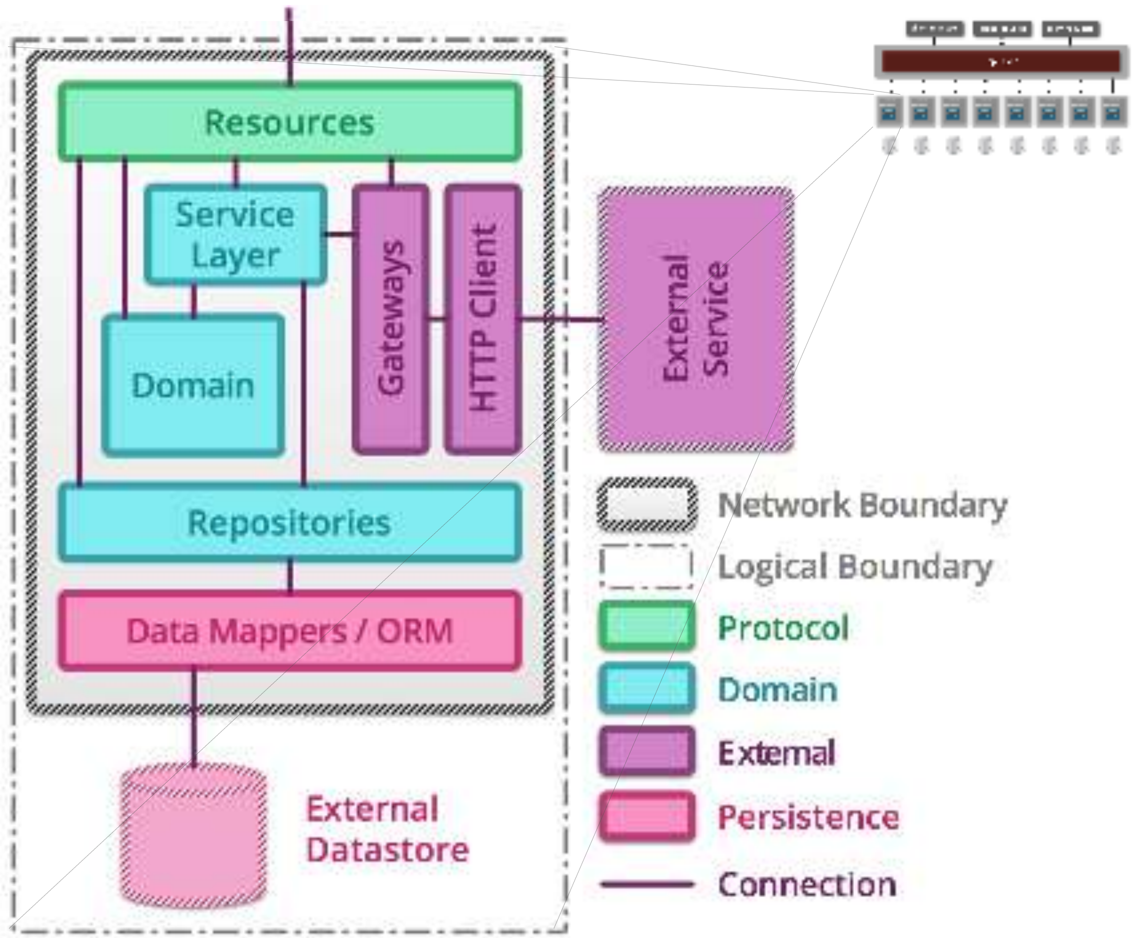
dimensions


client requests
client requests
client requests
api layer
service component
service component
service component
service component
service component
service component
service component
service component

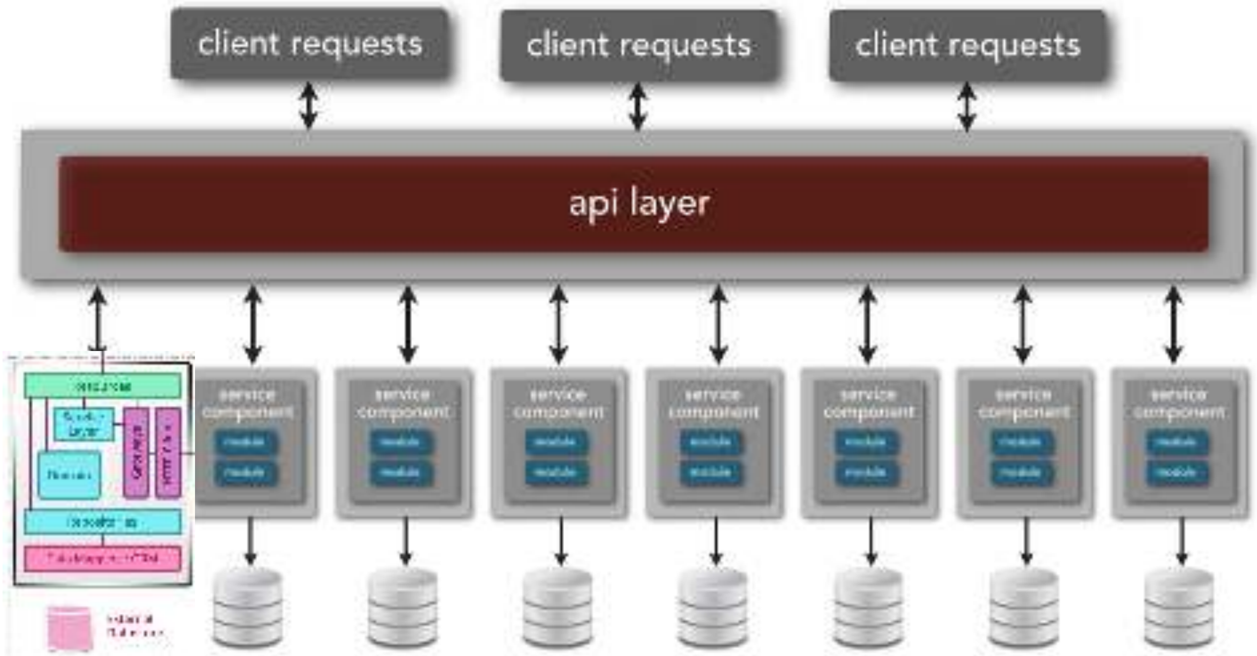
representational



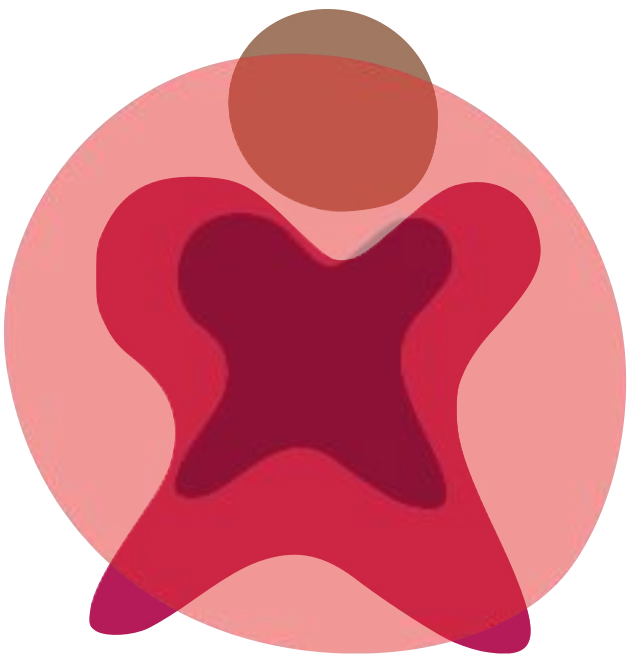
representational



representational

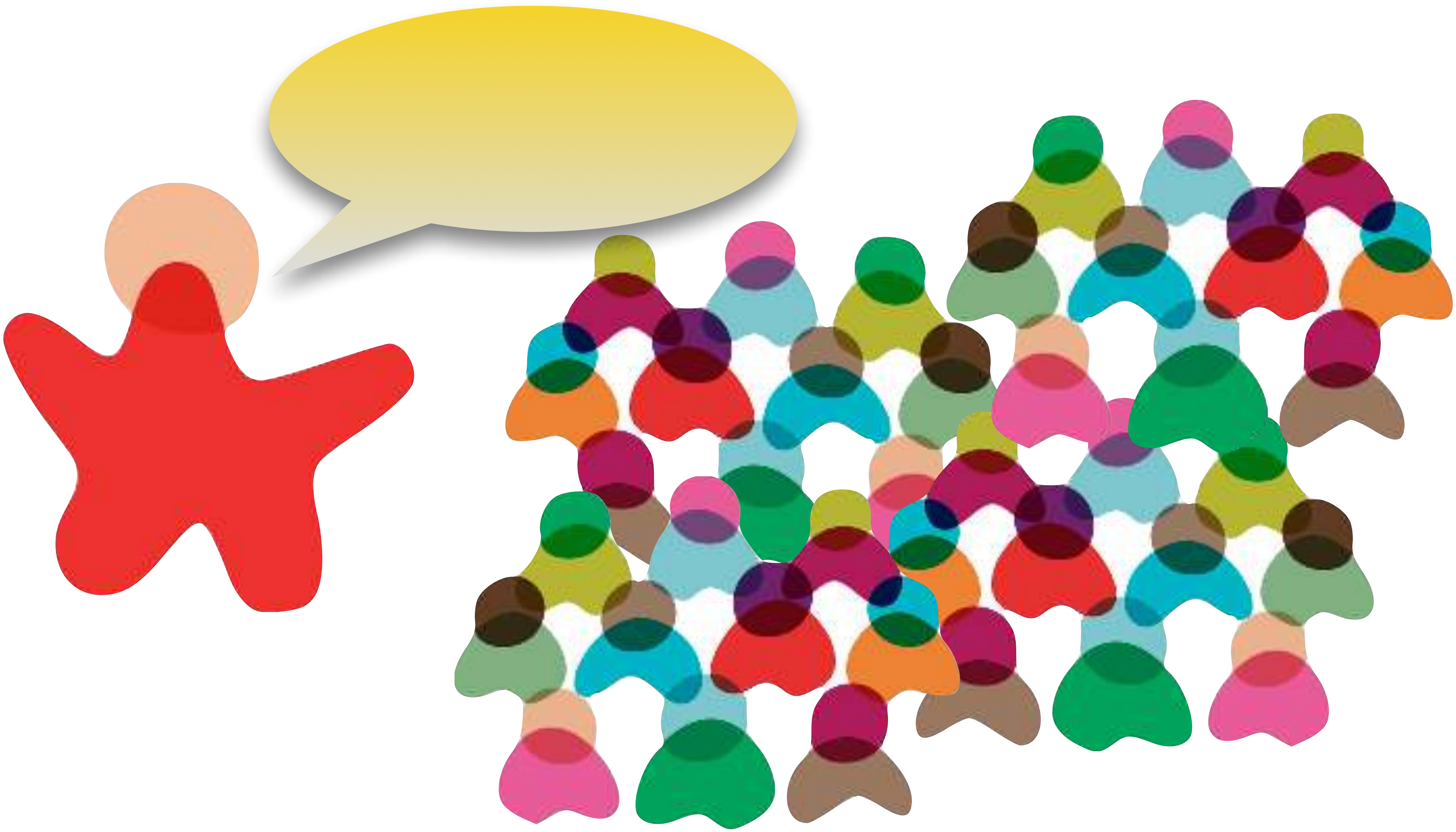


performance anti-patterns

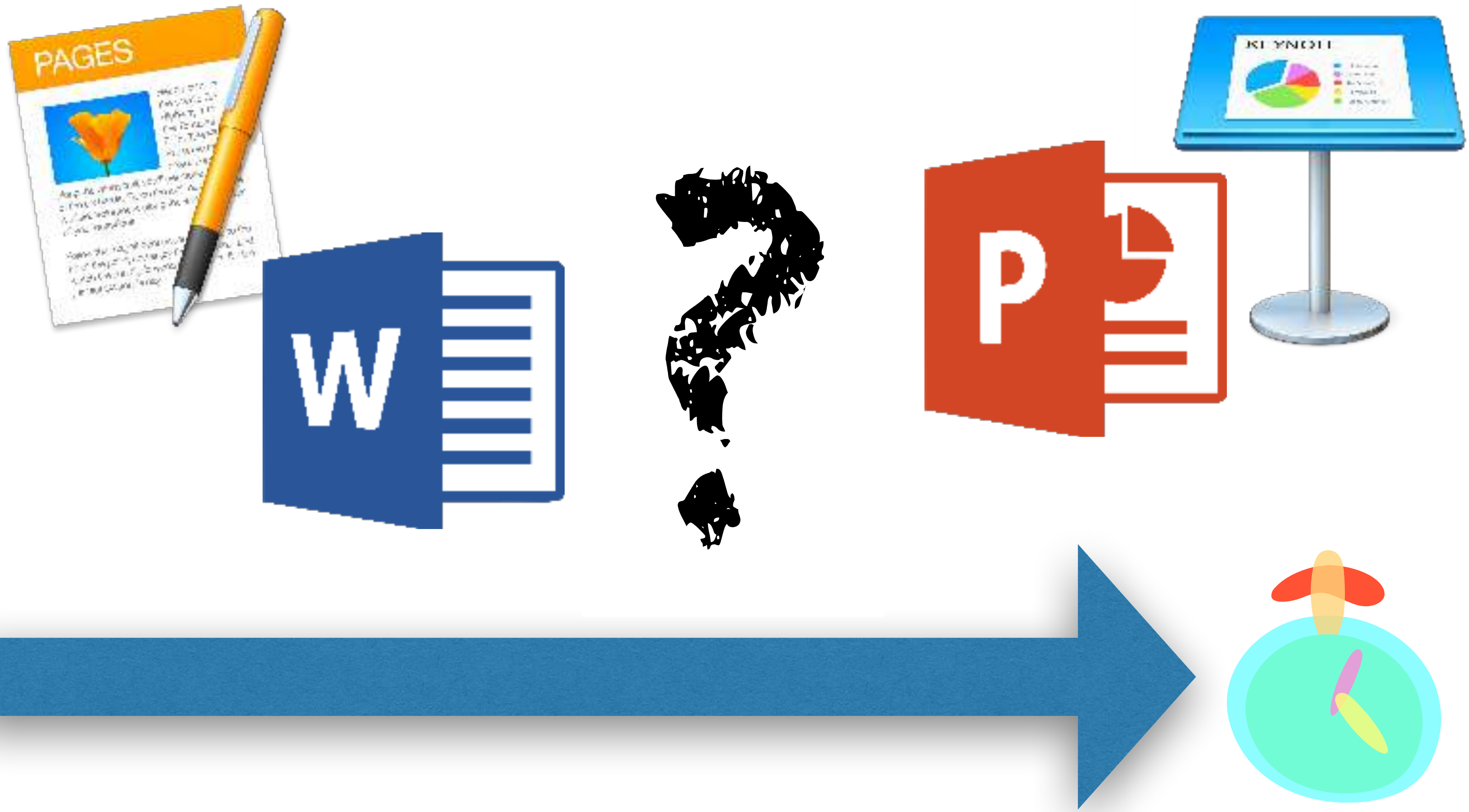


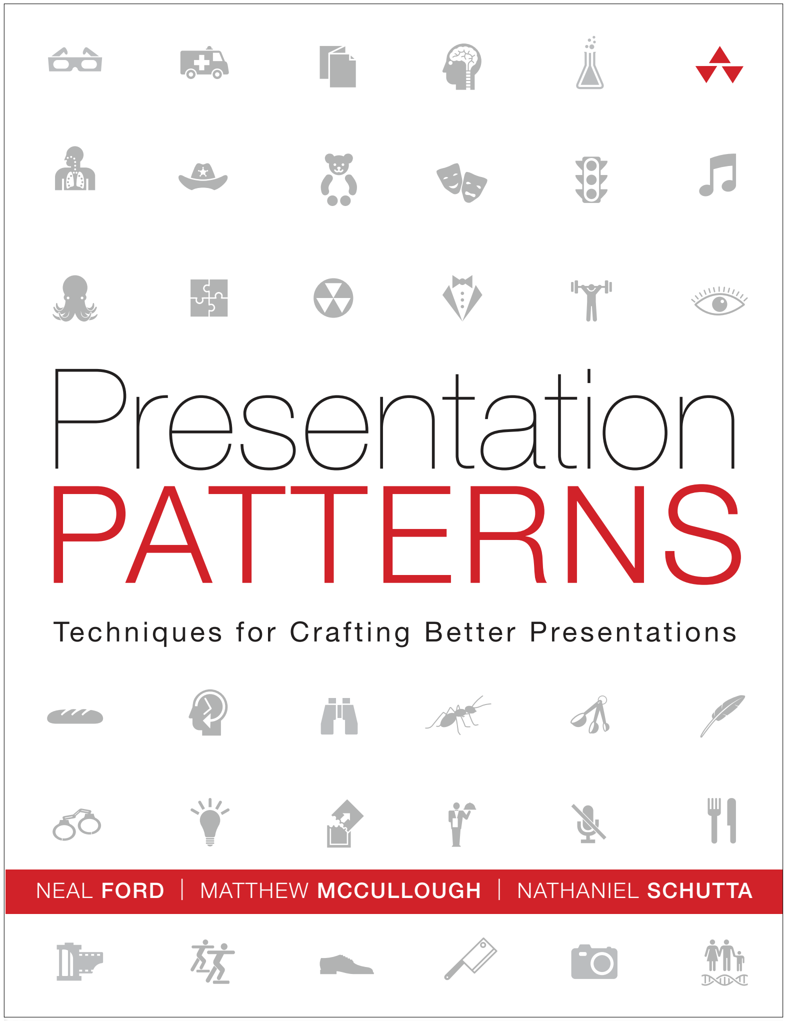
Going Meta

To bore your audience (at best) and annoy it (at worst), talk about your presentation within the presentation.



−'s	+'s
Shortchanging useful topic time	
Talking about something only you care about	0
Negative foreshadowing	


PAGES



ttp://presentationpatterns.com

architecture katas

documenting and presenting your architecture

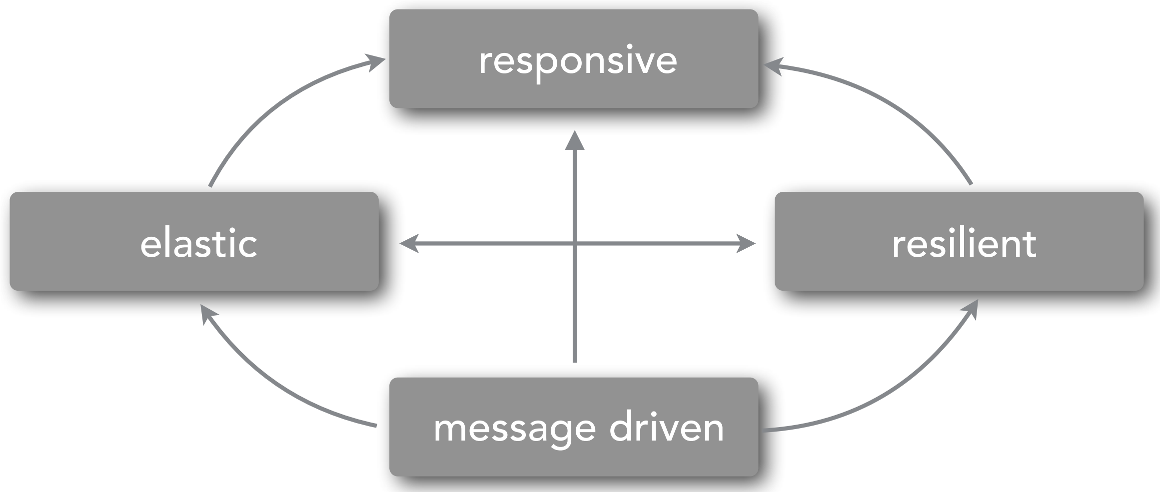
Reactive Architecture

source code

https://github.com/wmr513/reactive

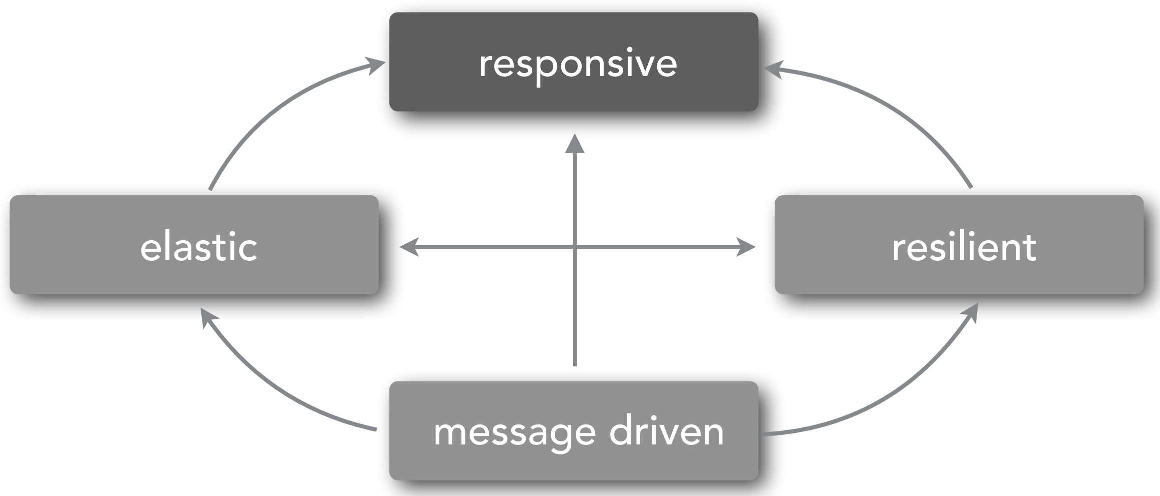
reactive architecture

reactive manifesto



reactive architecture

reactive manifesto

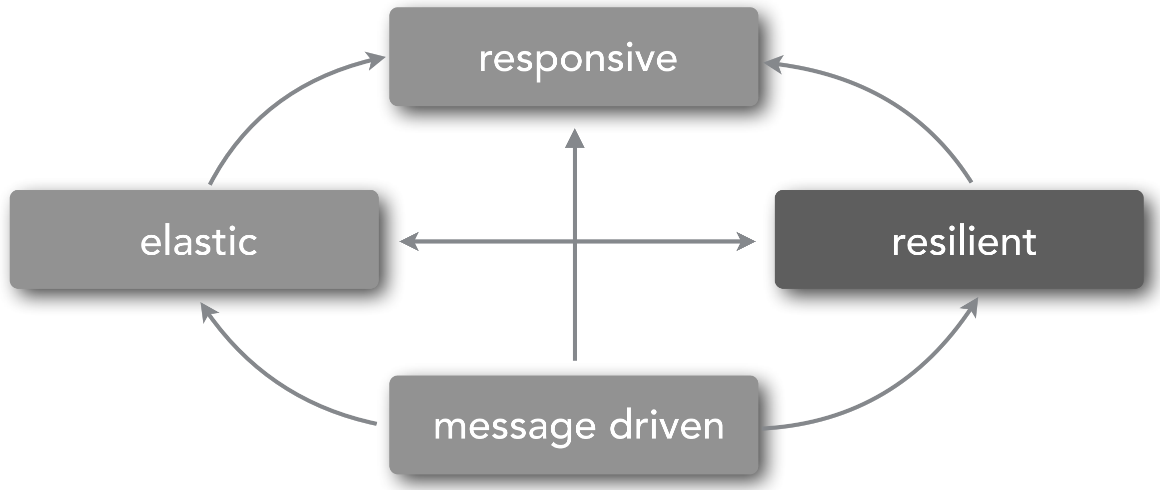


the system responds in a consistent, rapid, and timely manner whenever possible

how the system reacts to users

reactive architecture

reactive manifesto

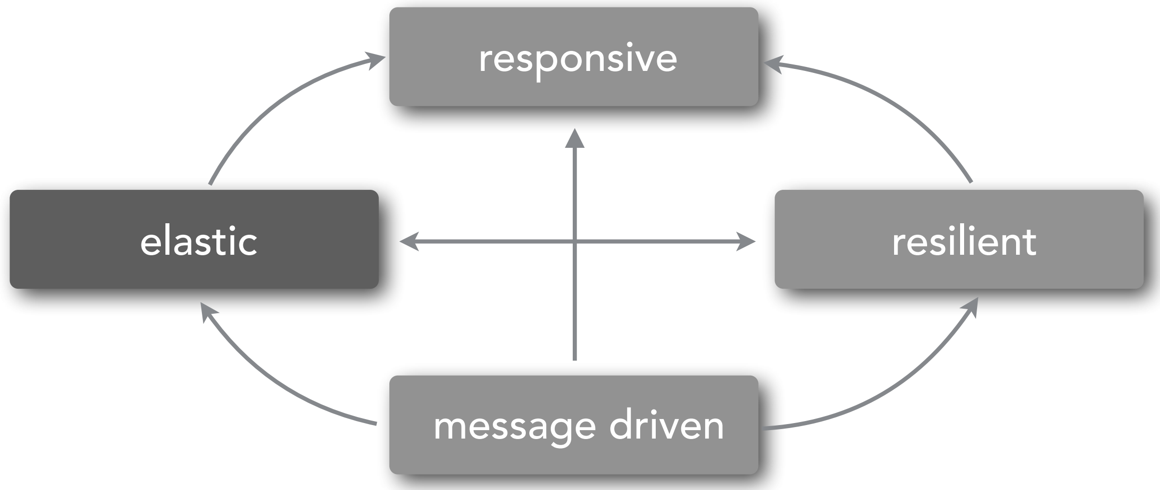


the system stays responsive after a failure through replication, containment, isolation, and delegation

how the system reacts to failures

reactive architecture

reactive manifesto

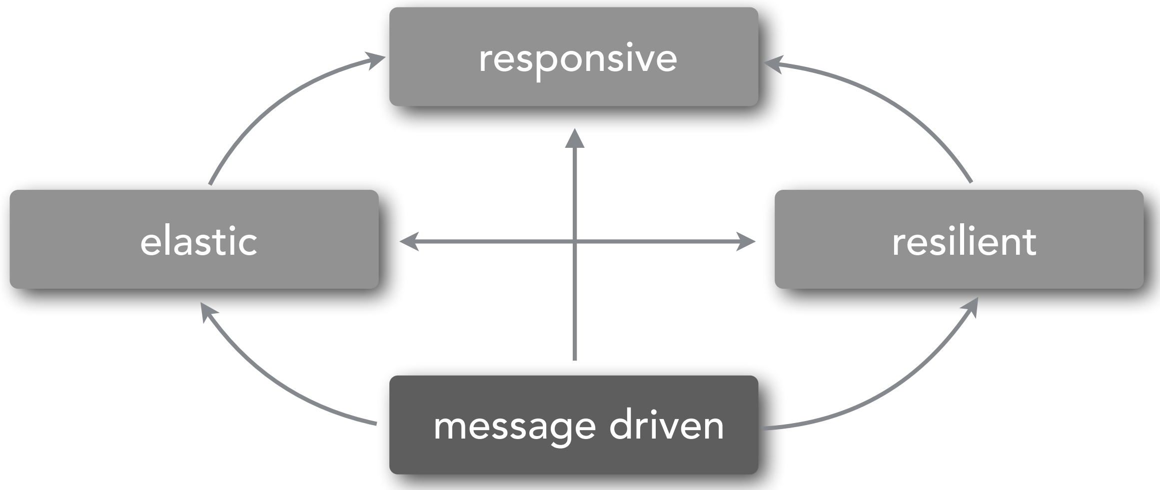


the system stays responsive under varying workload

how the system reacts to load

reactive architecture

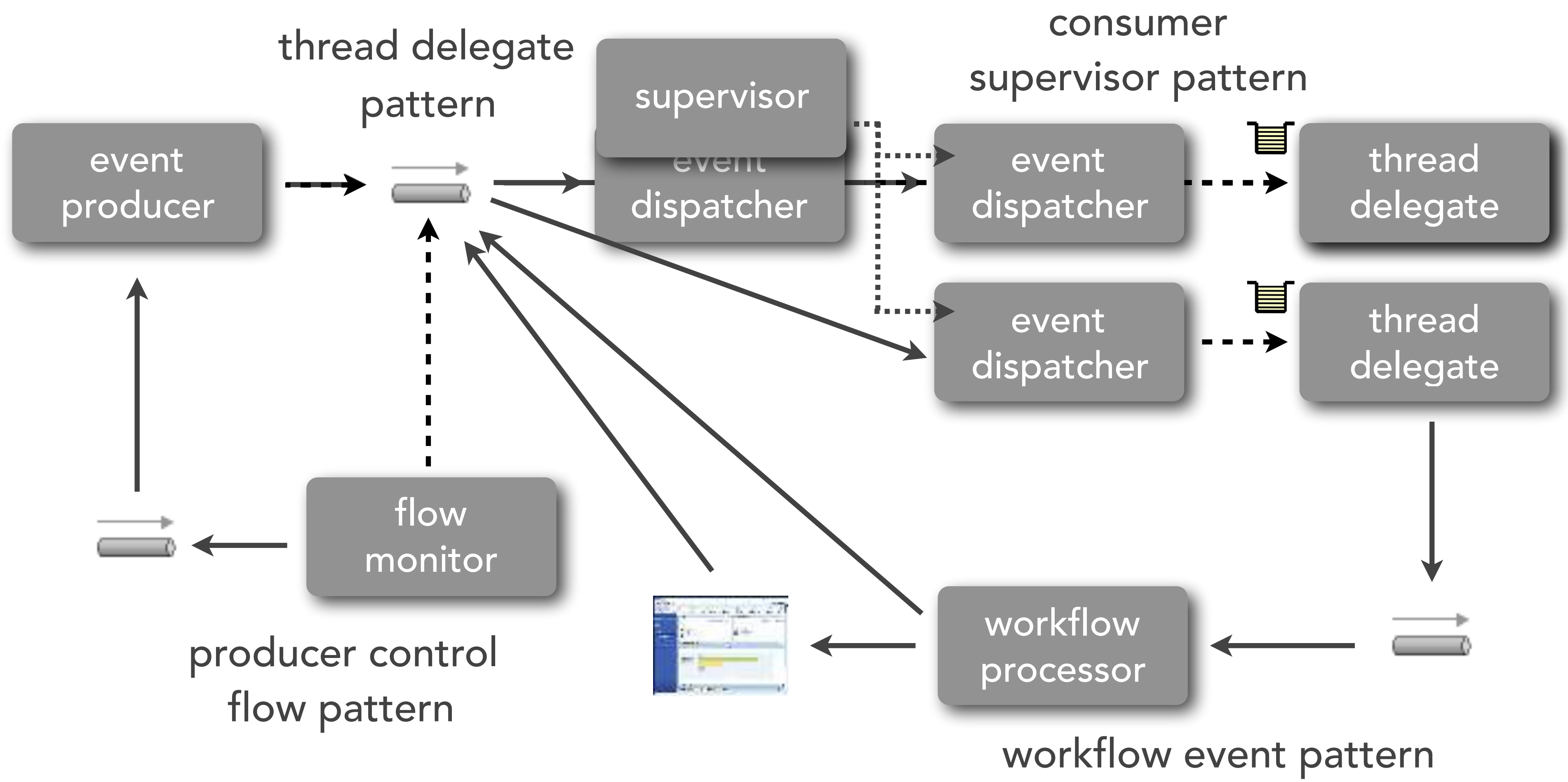
reactive manifesto



the system relies on asynchronous messaging to ensure loose coupling, isolation, location transparency, and error delegation

how the system reacts to events

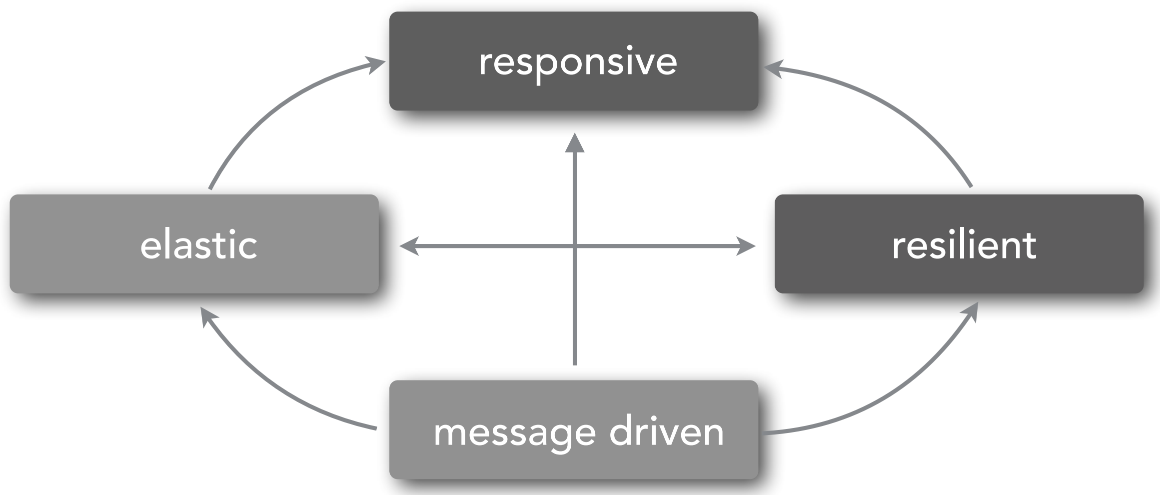
reactive patterns for self-healing systems



Thread Delegate Pattern

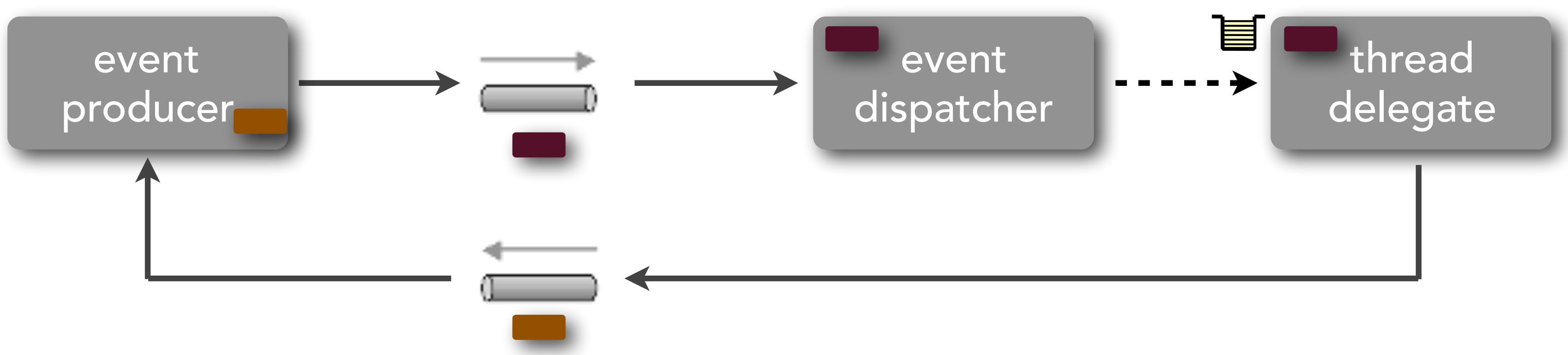
thread delegate pattern

how can you ensure timely and consistent response time as your system grows?



thread delegate pattern

how can you ensure timely and consistent response time as your system grows?

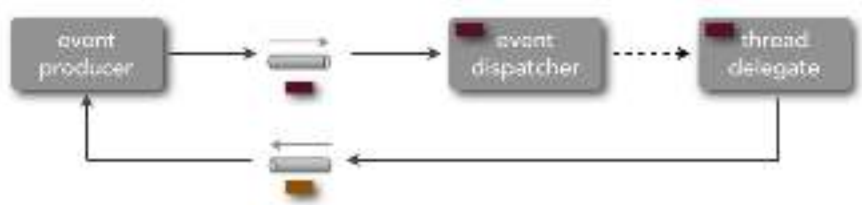


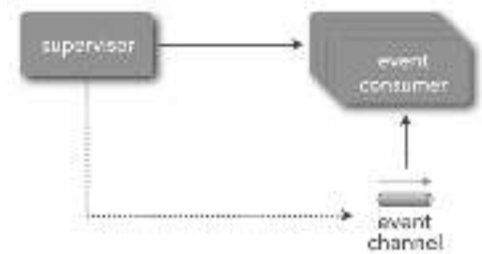
thread delegate pattern

let’s see the issue…

thread delegate pattern

thread delegate vs. consumer supervisor

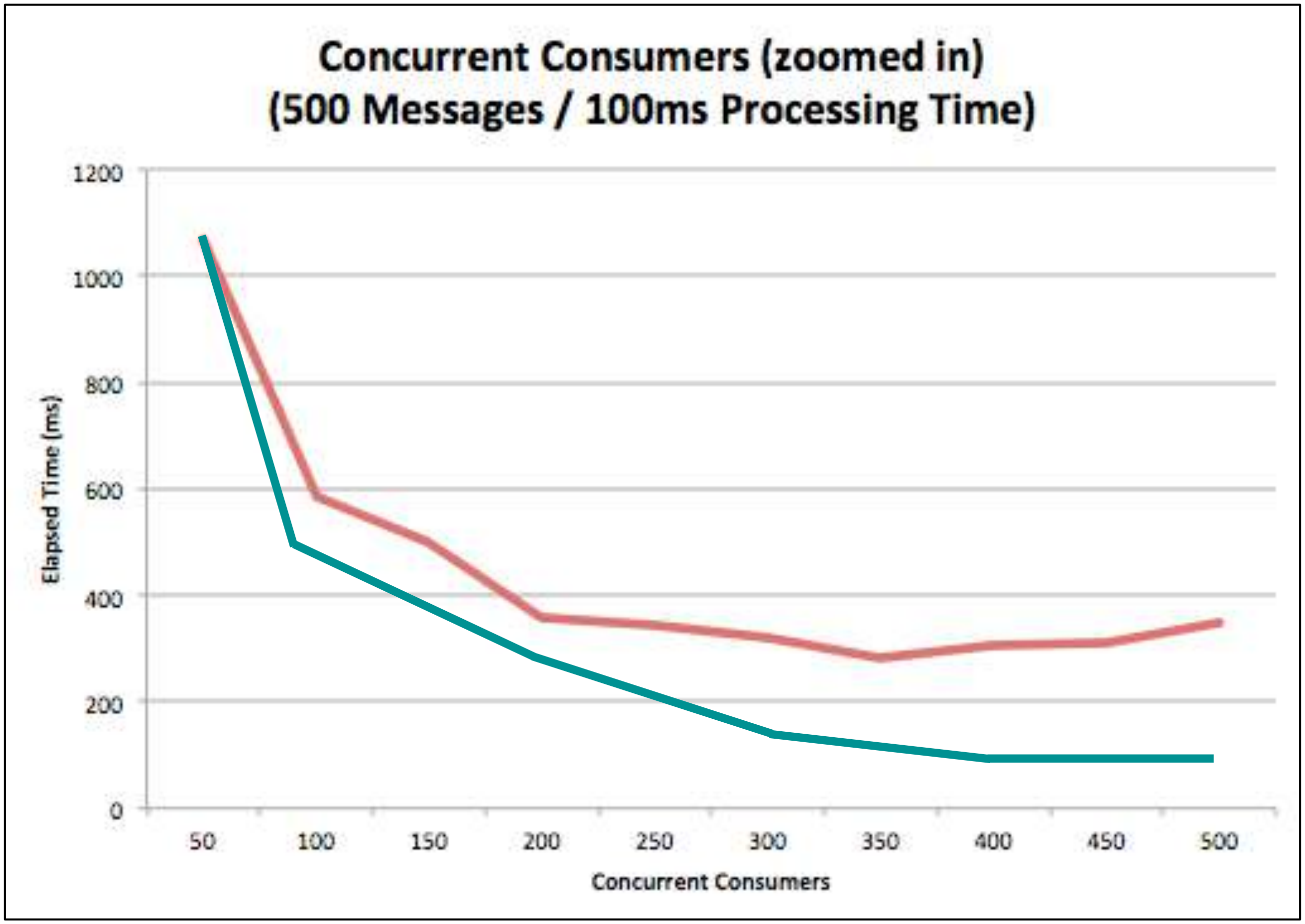




thread delegate	consumer supervisor
scalability	elasticity
consistent consumers	variable consumers
decoupled event processors	coupled event processors
near-linear performance	diminishing performance

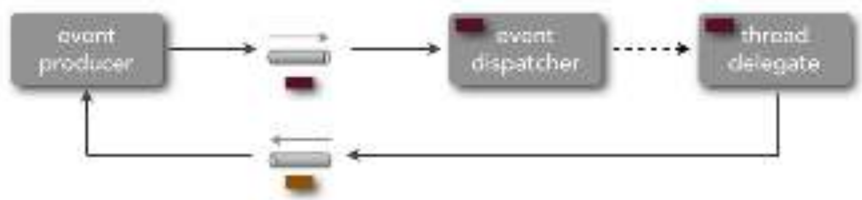
thread delegate pattern

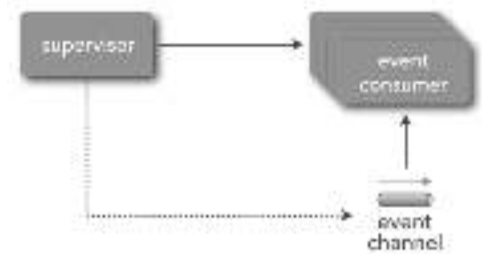
thread delegate vs. consumer supervisor



thread delegate pattern

thread delegate vs. consumer supervisor





scalability

consistent consumers

decoupled event processors

near-linear performance

can preserve message order

elasticity

variable consumers

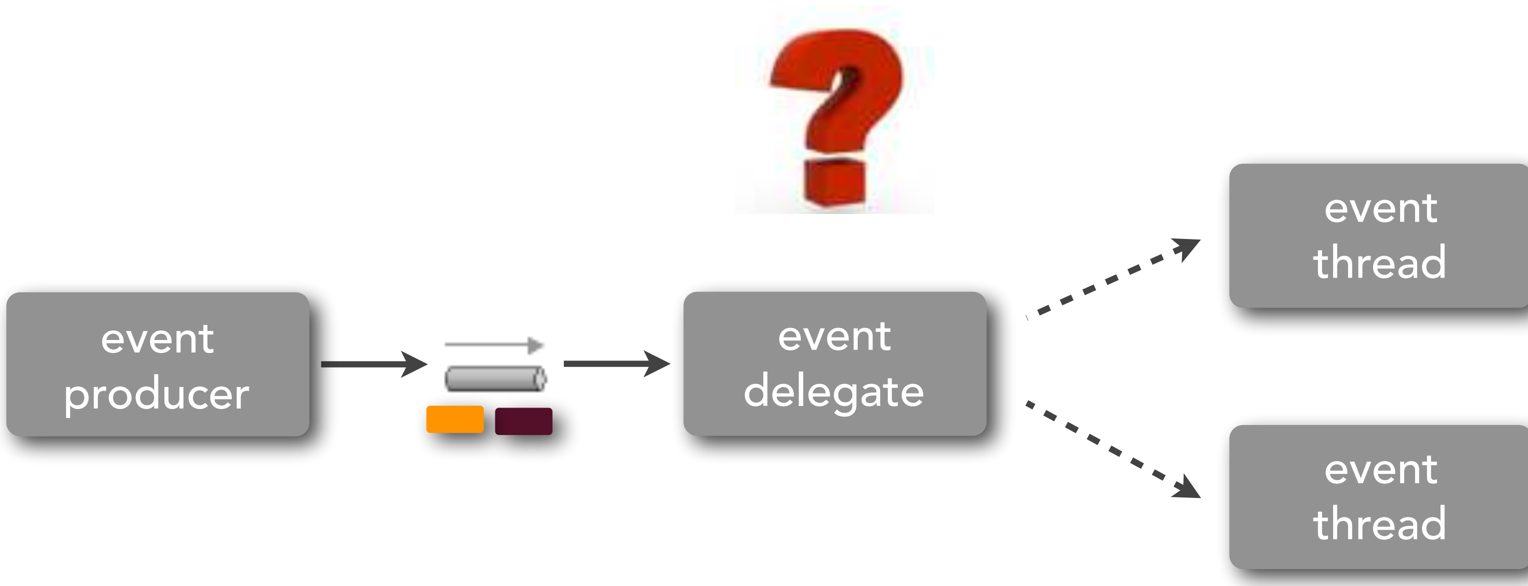
coupled event processors

diminishing performance

message order not preserved

thread delegate pattern

preserving message order



thread delegate pattern

preserving message order

premise: not every message must be ordered, but rather messages *within a context* must be ordered

```
1. PLACE  AAPL A-136 2,000,000.00
2. CANCEL AAPL A-136 2,000,000.00
3. REBOOK AAPL A-136 1,800,000.00
```

1, 2, 3

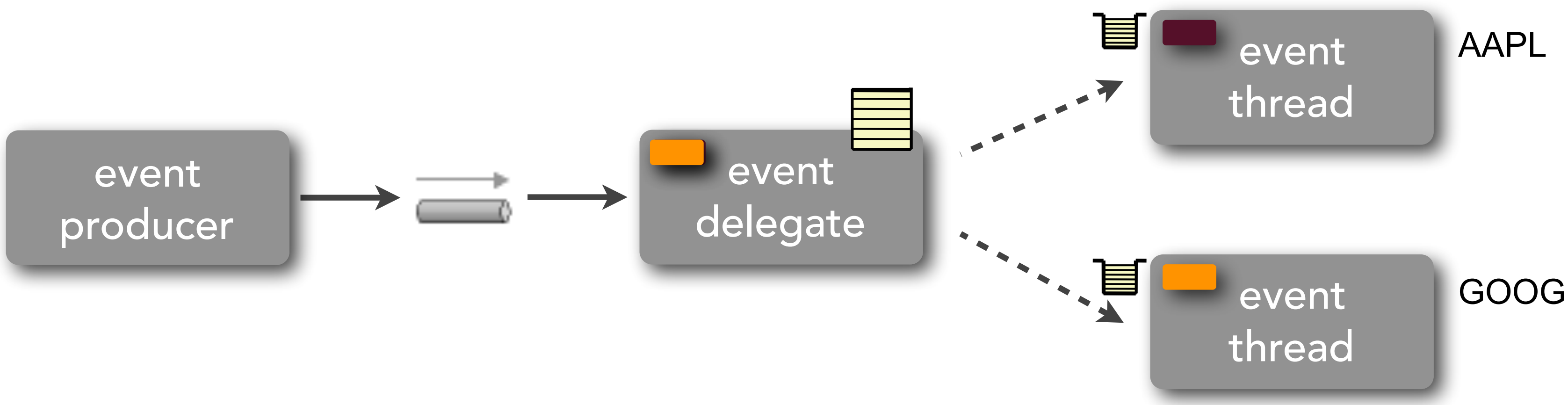
```
1. PLACE  AAPL A-136 2,000,000.00
2. PLACE  GOOG V-976   650,000.00
3. CANCEL GOOG V-976   650,000.00
4. CANCEL AAPL A-136 2,000,000.00
5. REBOOK AAPL A-136 1,800,000.00
6. REBOOK GOOG V-976   600,000.00
```

1, 4, 5

2, 3, 6

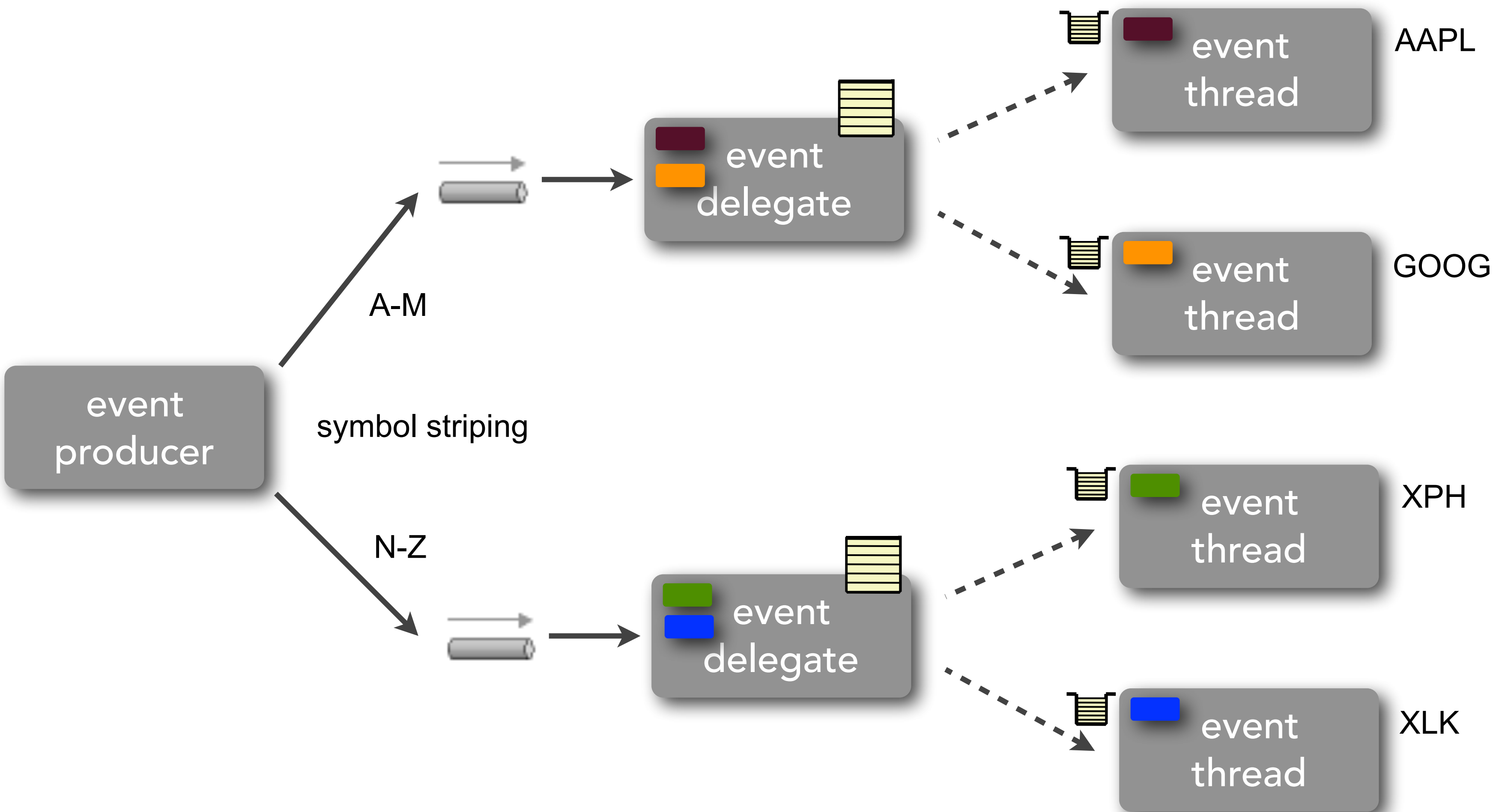
thread delegate pattern

preserving message order



thread delegate pattern

preserving message order



thread delegate pattern

Dispatcher

```
while (true) {
  //get the next message from the queue
  //get next available thread
  //send message to thread (or start new thread)
}
```

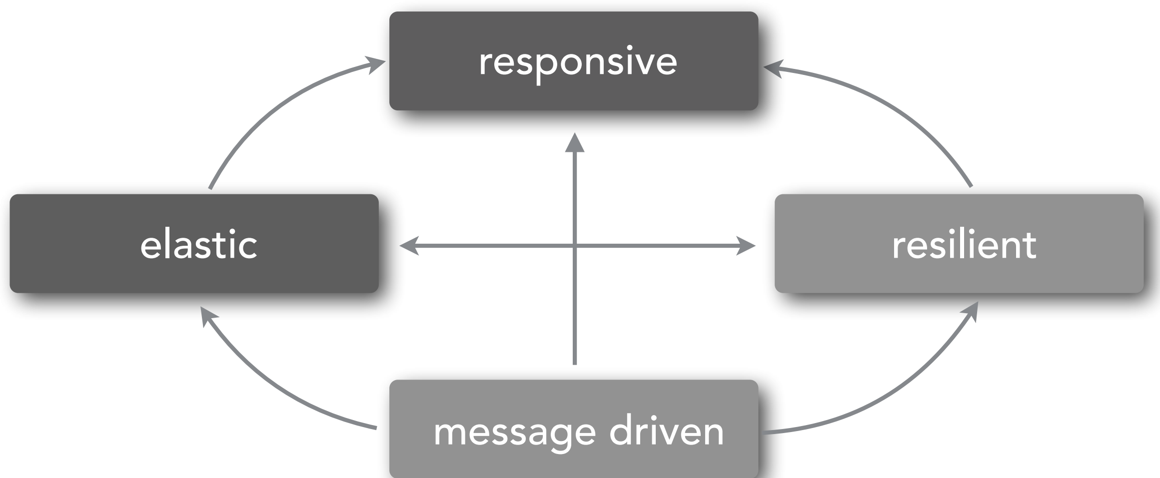
thread delegate pattern

let's see the result…

Consumer Supervisor Pattern

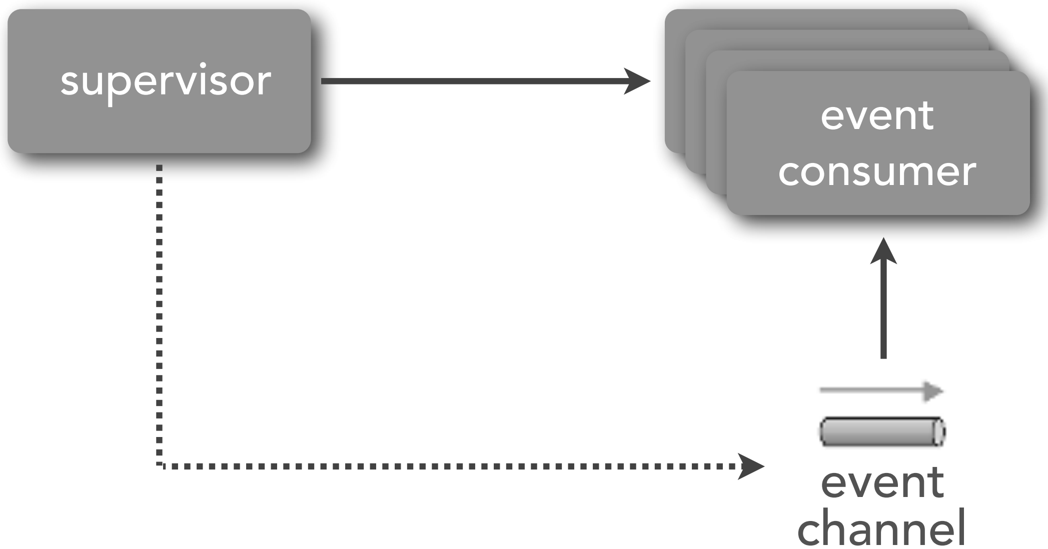
consumer supervisor pattern

how can you react to varying changes in load to event consumers to ensure consistent response time?



consumer supervisor pattern

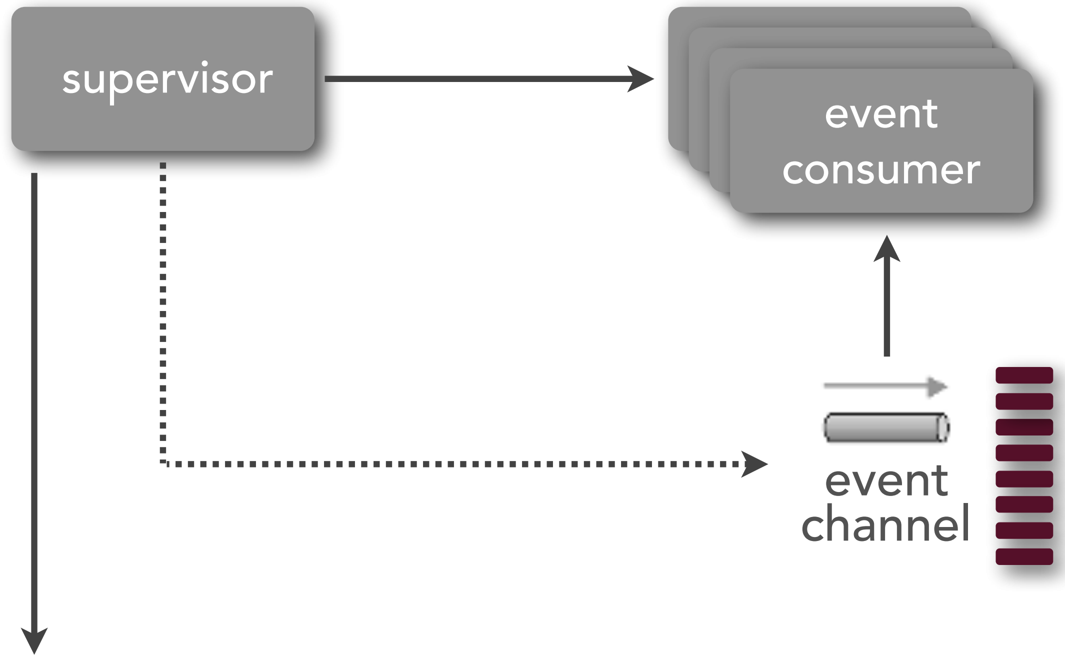
how can you react to varying changes in load to event consumers to ensure consistent response time?



consumer supervisor pattern

let’s see the issue….

consumer supervisor pattern



periodically monitor queue depth

determine consumers needed (e.g., depth/2)

apply max threshold

add or remove consumers

consumer supervisor pattern

Supervisor.java

```
List<MyConsumer> consumers = new ArrayList<MyConsumer>();

private void startConsumer() {
   MyConsumer consumer = new MyConsumer();
   consumers.add(consumer);
   new Thread() { public void run() {
      consumer.startup(connection);
   }}.start();
}

private void stopConsumer() {
  if (consumers.size() > 1) {
    AMQPConsumer consumer = consumers.get(0);
    consumer.shutdown();
    consumers.remove(consumer);
   }
 }
```

consumer supervisor pattern

Supervisor.java

```
public void execute() throws Exception {
   startConsumer();
   while (true) {
      Thread.sleep(1000);
      long queueDepth = getMessageCount("trade.eq.q");
      long consumersNeeded = queueDepth/2;
      long diff = Math.abs(consumersNeeded - consumers.size());
      for (int i=0;i<diff;i++) {
         if (consumersNeeded > consumers.size())
            startConsumer();
         else
            stopConsumer();
      }
   }
}
```

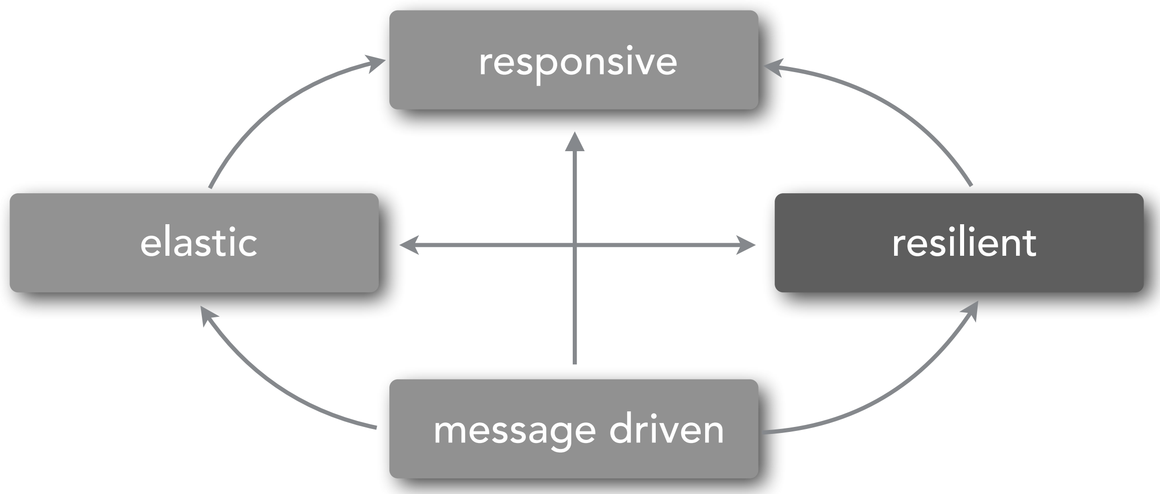
consumer supervisor pattern

let’s see the result…

Workflow Event Pattern

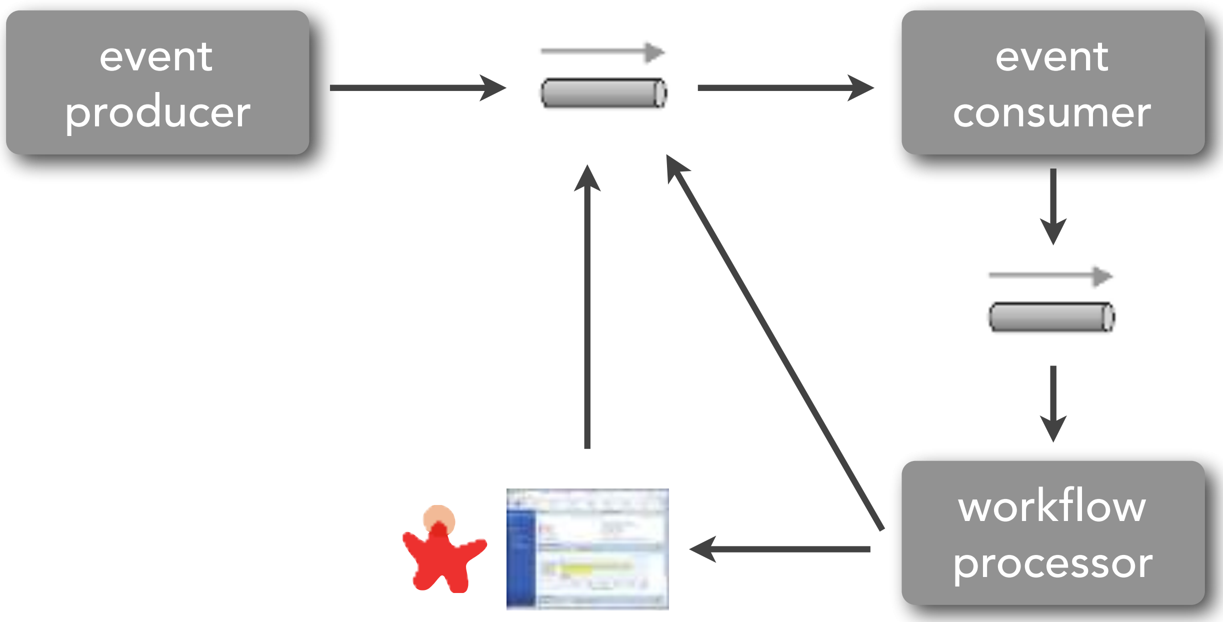
workflow event pattern

how can you handle error conditions without failing the transaction?



workflow event pattern

how can you handle error conditions without failing the transaction?



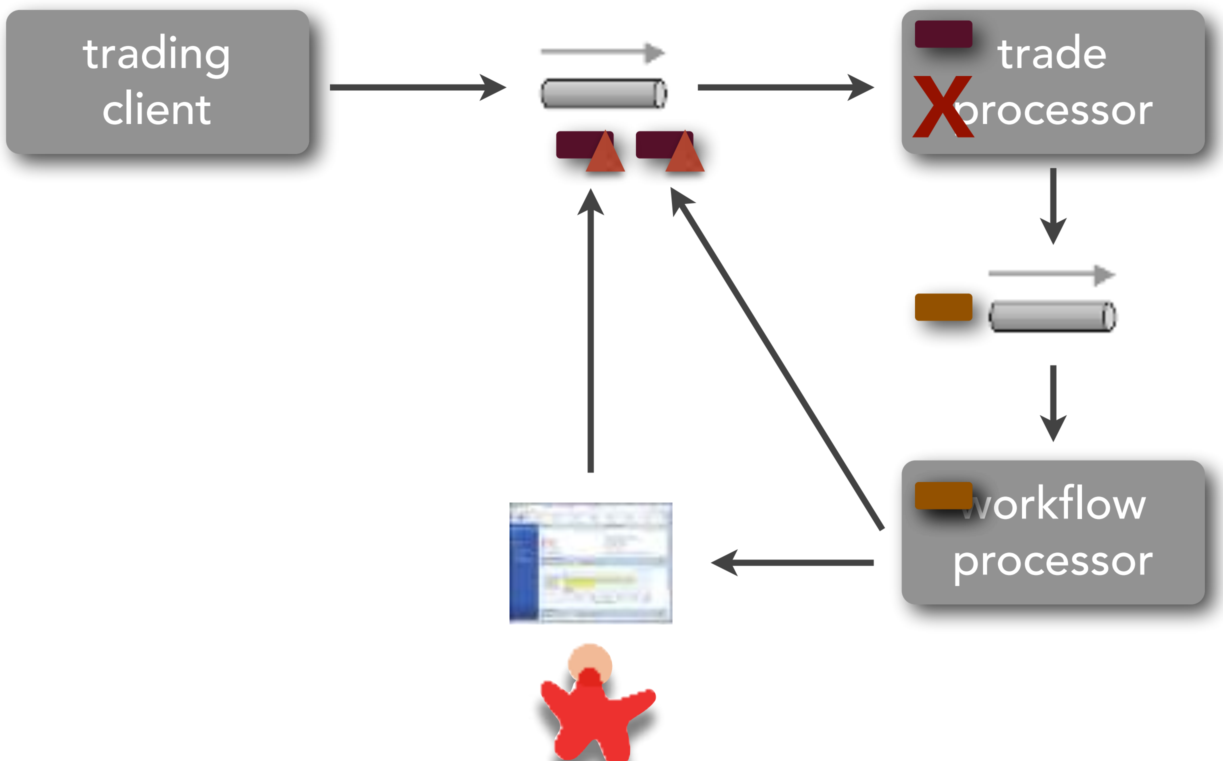
workflow event pattern

let’s see the issue…

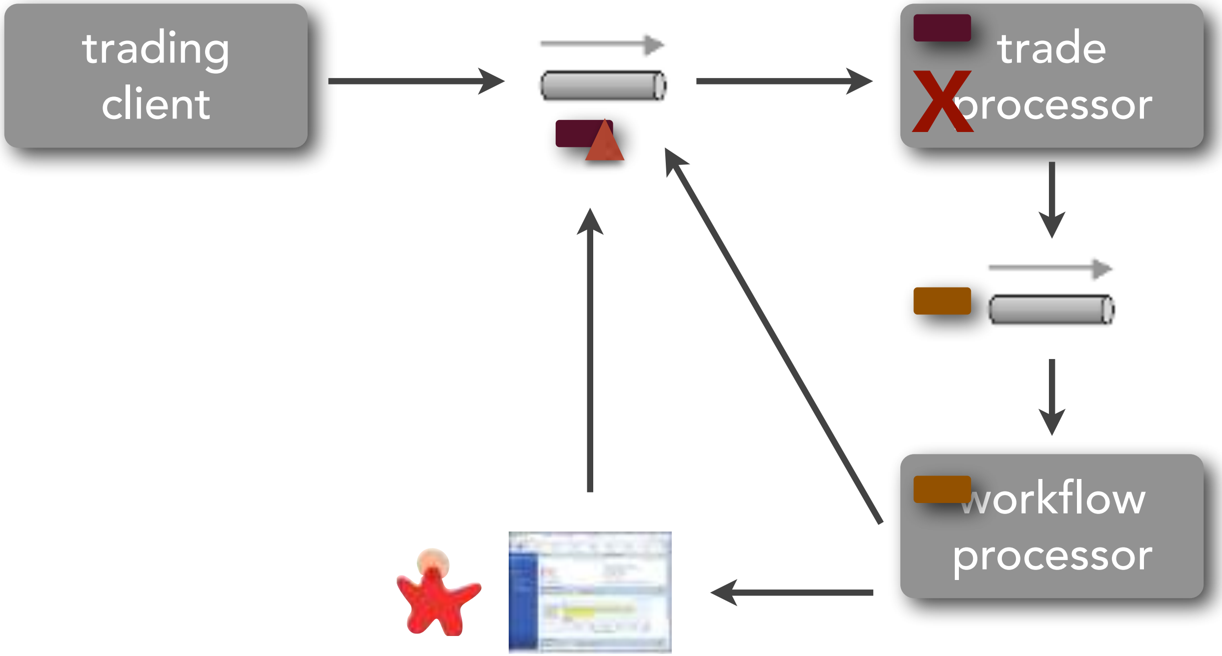
workflow event pattern

example

while asynchronously processing trades an error occurs with one of the trade orders



workflow event pattern



send error to queue

programmatically fix error

resubmit to processing queue

---

send error to dashboard

human fixes error

resubmit to processing queue

workflow event pattern

Workflow.java

```
//get next message from queue
String newMsg = msg.substring(0, msg.indexOf(" shares"));
//resubmit message
```

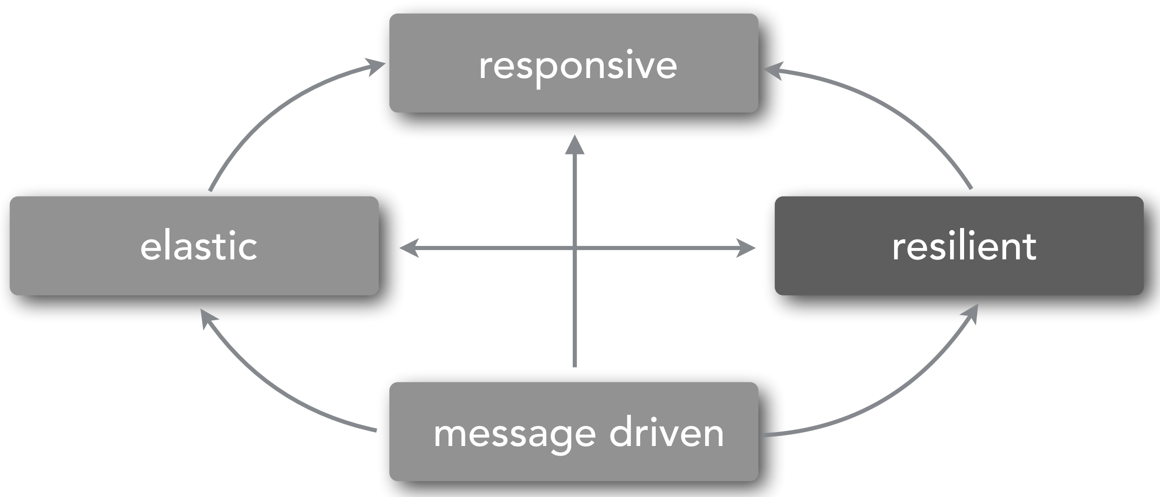
consumer supervisor pattern

let's see the result...

Producer Control Flow Pattern

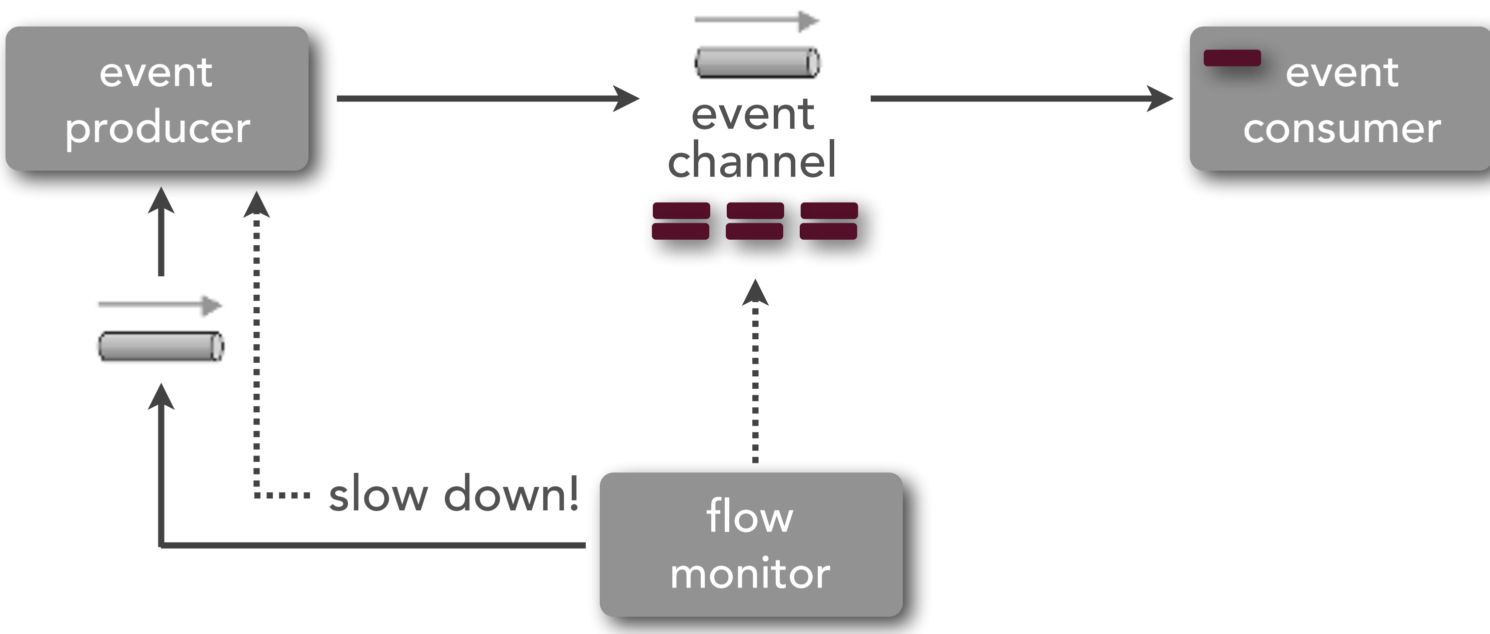
producer control flow pattern

how can you slow down message producers when the messaging system becomes overwhelmed?



producer control flow pattern

how can you slow down message producers when the messaging system becomes overwhelmed?



producer control flow pattern

how can you slow down message producers when the messaging system becomes overwhelmed?

RabbitMQ™

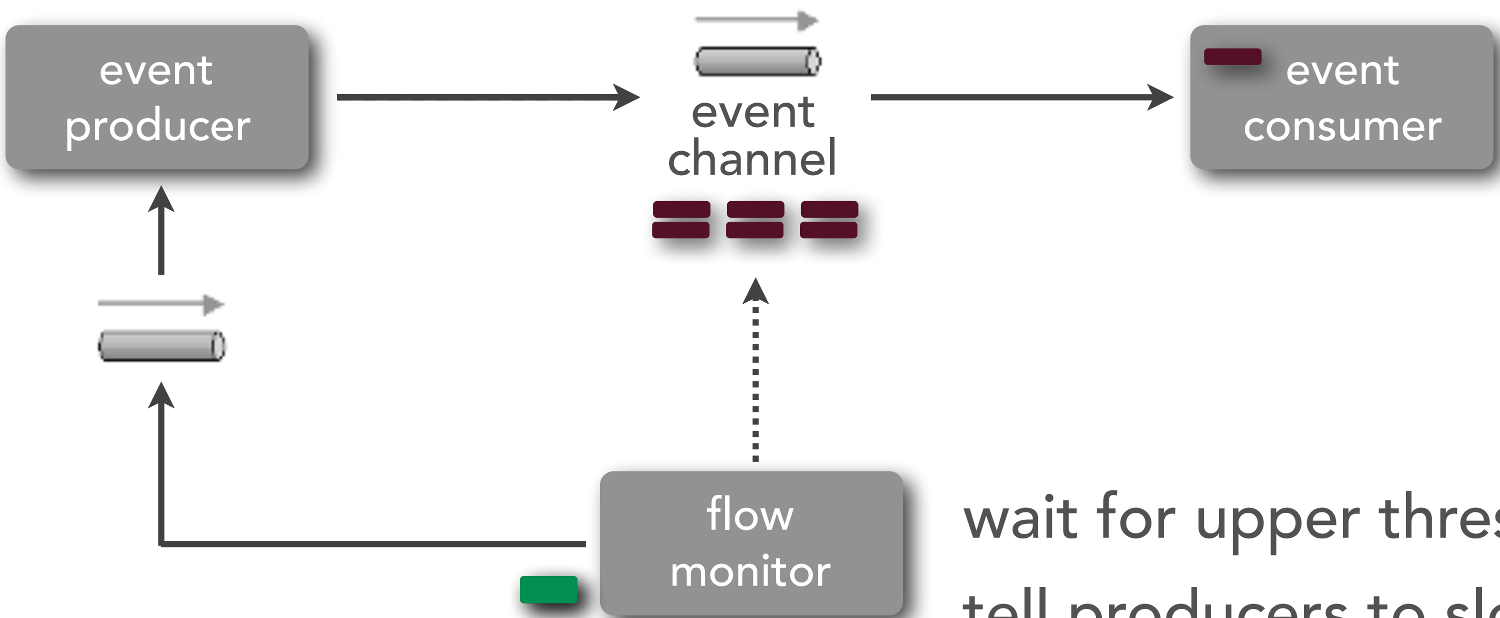
ActiveMQ

shutdown (broker) vs. slowdown (pattern)

producer control flow pattern

let's see the issue….

producer control flow pattern



wait for upper threshold

tell producers to slow down

wait for lower threshold

tell producers to resume

producer control flow pattern

FlowMonitor.java

```java
public void execute() throws Exception {
   long threshold = 10;
   boolean controlFlow = false;
   while (true) {
      Thread.sleep(3000);
      long queueDepth = getMessageCount("trade.eq.q");
      if (queueDepth > threshold && !controlFlow) {
         controlFlow = enableControlFlow(channel);
      } else if (queueDepth <= (threshold/2) && controlFlow) {
         controlFlow = disableControlFlow(channel);
      }
   }
}
```

producer control flow pattern

FlowMonitor.java

```
private boolean enableControlFlow(Channel channel) {
   byte[] msg = String.valueOf(true).getBytes();
   //send message to producer
   return true;
}


private boolean disableControlFlow(Channel channel) {
   byte[] msg = String.valueOf(false).getBytes();
   //send message to producer
   return false;
}
```

producer control flow pattern

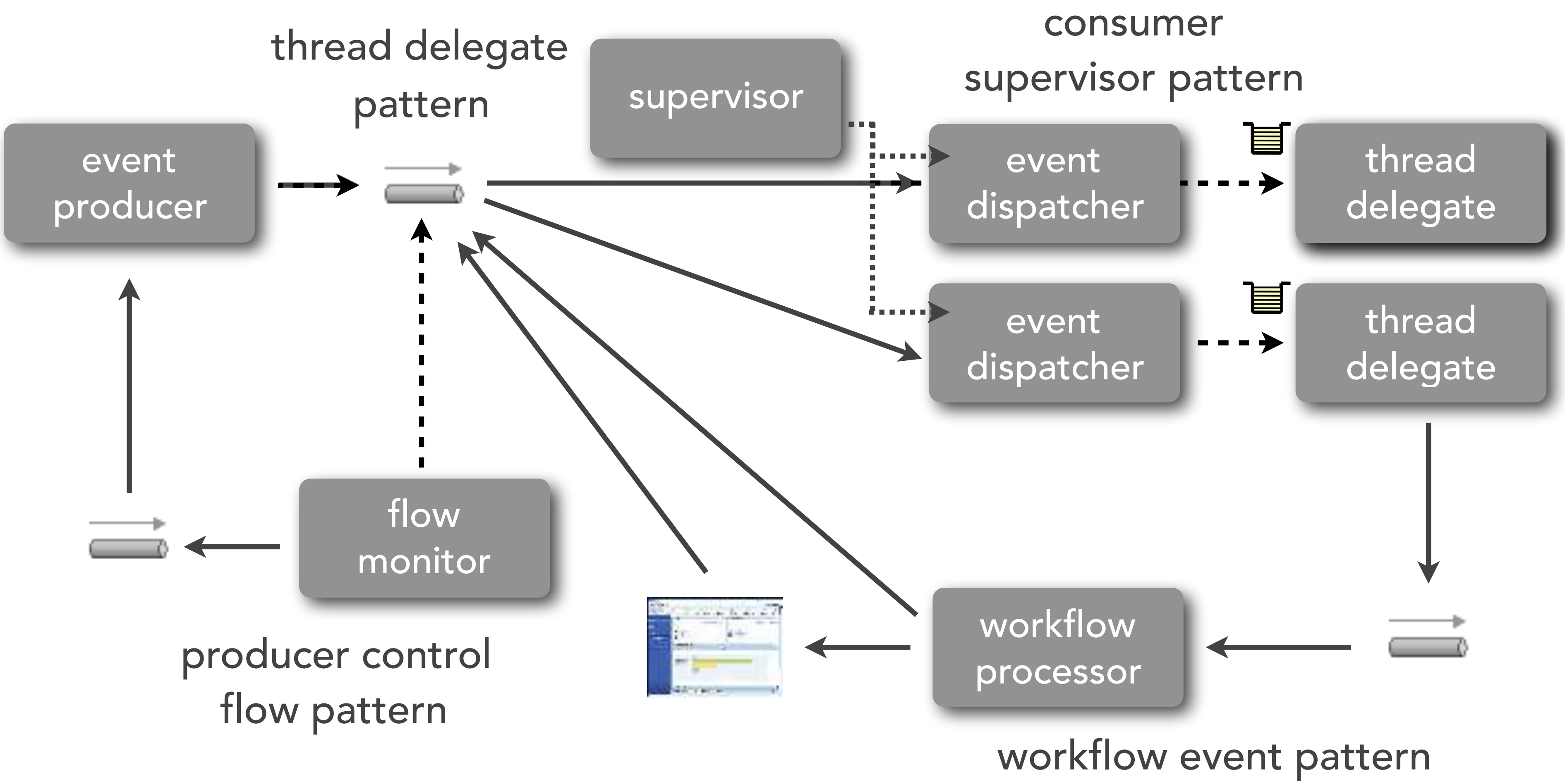
Producer.java

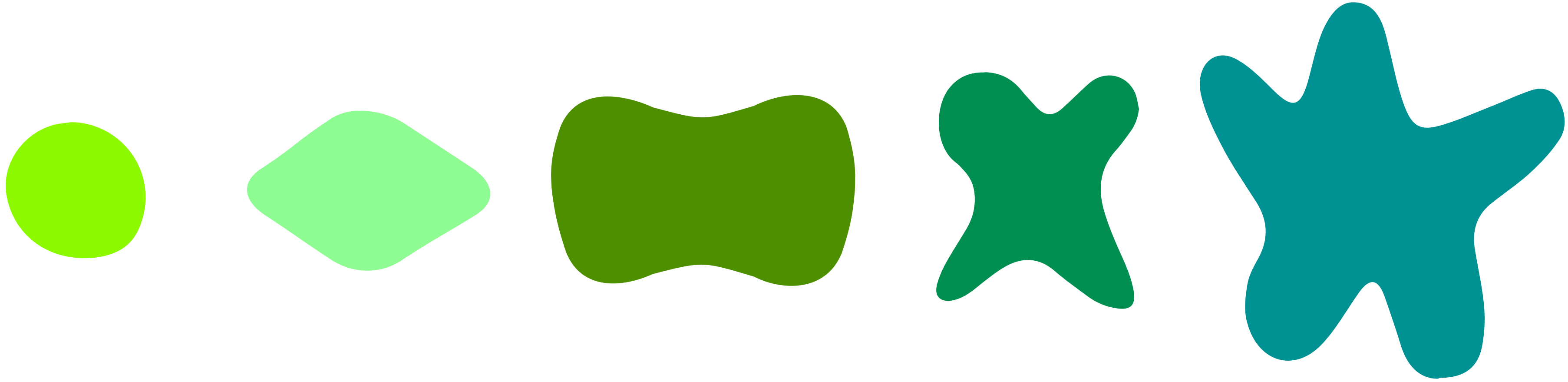
```
public void startListener() {
   new Thread() {
     public void run() {
     while (true) {
        //wait for message from flow monitor
        boolean controlFlow =
          new Boolean(new String(msg.getBody())).booleanValue();
        synchronized(delay) { delay = controlFlow ? 3000 : 0; }
      }
   }}.start();
}

private void produceMessages() {
   Thread.sleep(delay);
   //send trade to queue...

}
```

producer control flow pattern

let’s see the result…

reactive patterns for self-healing systems





Evolutionary Architectures

ThoughtWorks

NEAL FORD

Director / Software Architect / Meme Wrangler

@neal4d
nealford.com

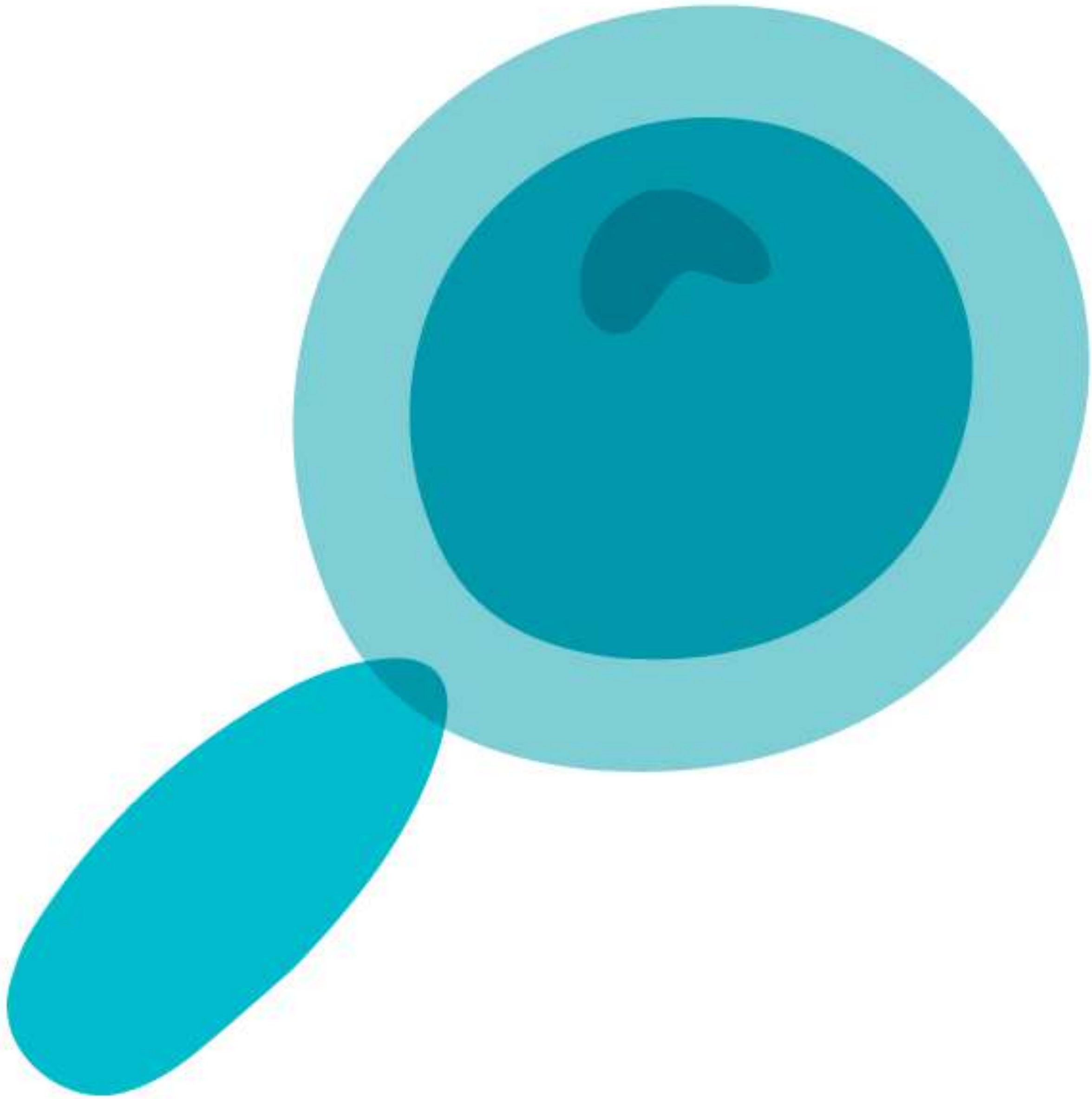
with Rebecca Parsons & Pat Kua

Rebecca Parsons

Pat Kua

Neal Ford

Photos by Martin Fowler:
http://martinfowler.com/albums/ThoughtWorkers/



Dynamic Equilibrium

Definition :

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.

Dimensions of Architecture:

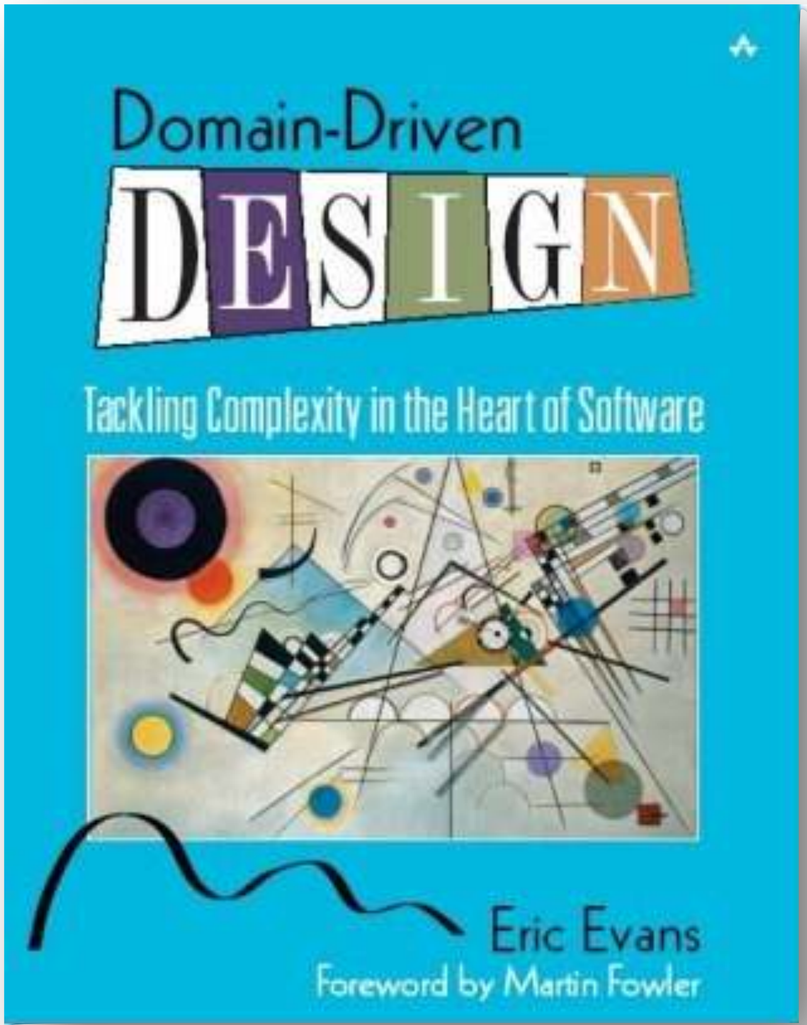
Technical: The implementation parts of the architecture

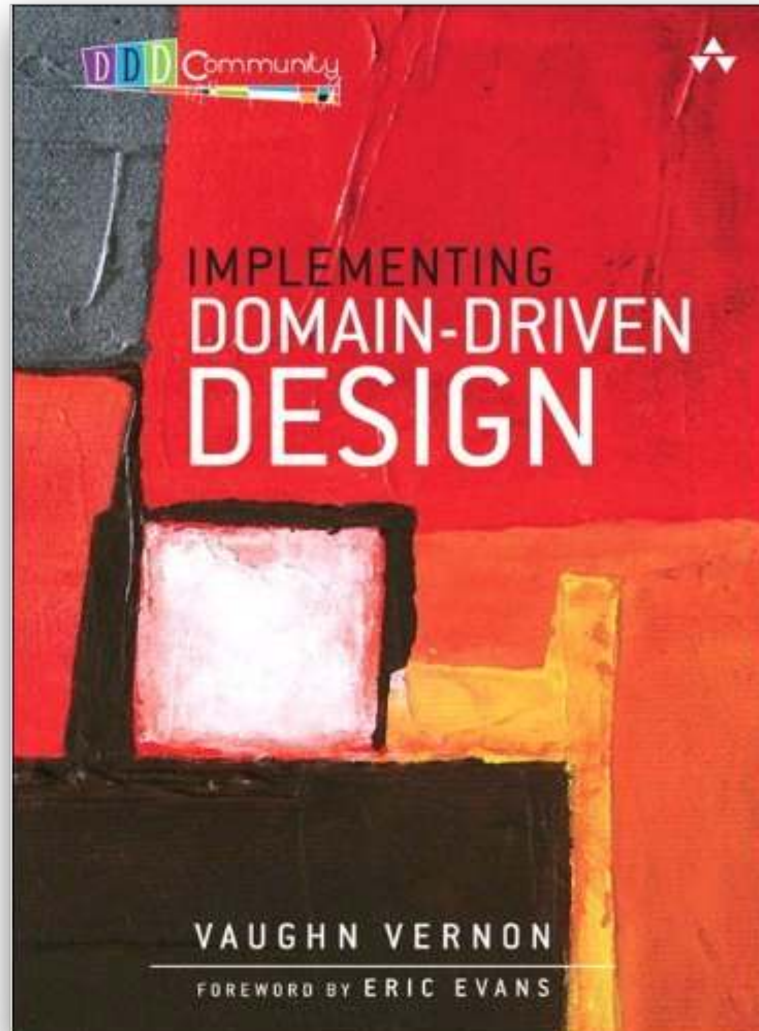
Data: Database schemas, table layouts, optimization planning, etc.

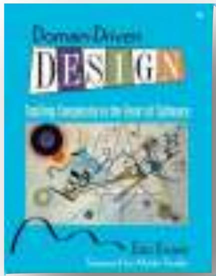
Security: Defines security policies, guidelines, and specifies tools to help uncover deficiencies.

Domain:

Domain Driven Design

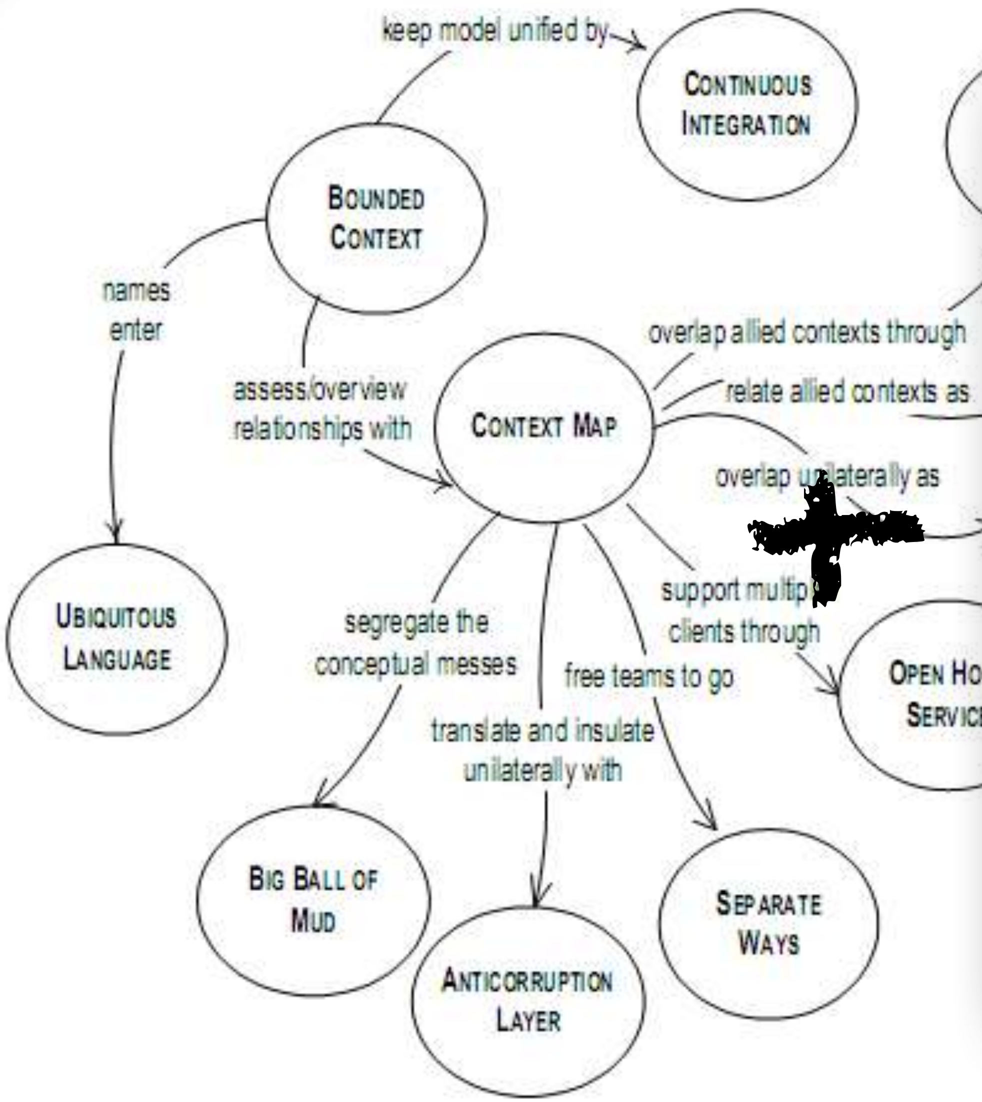


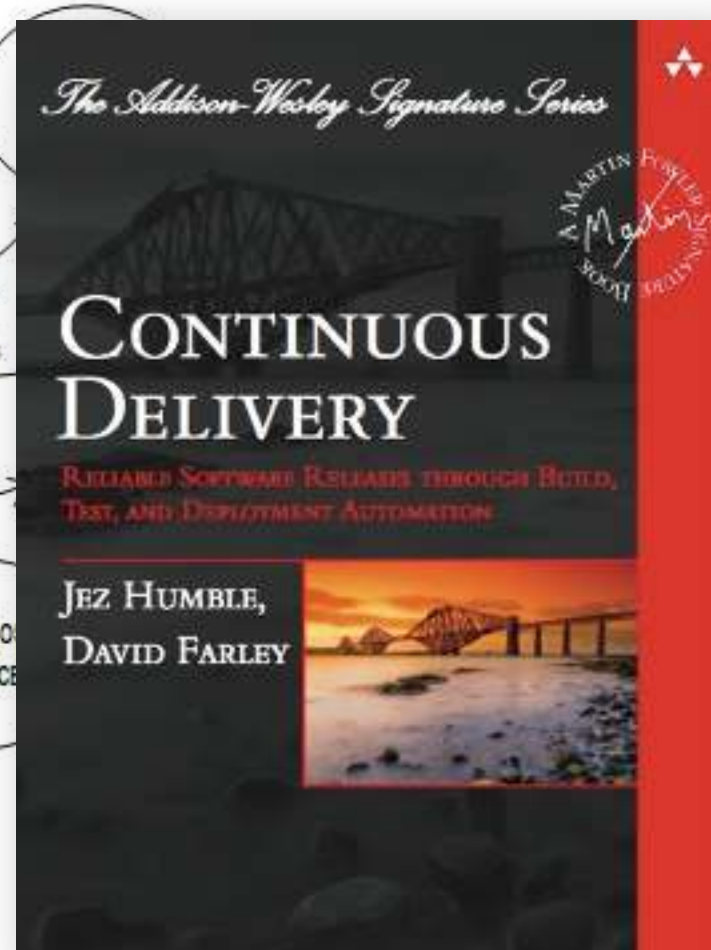




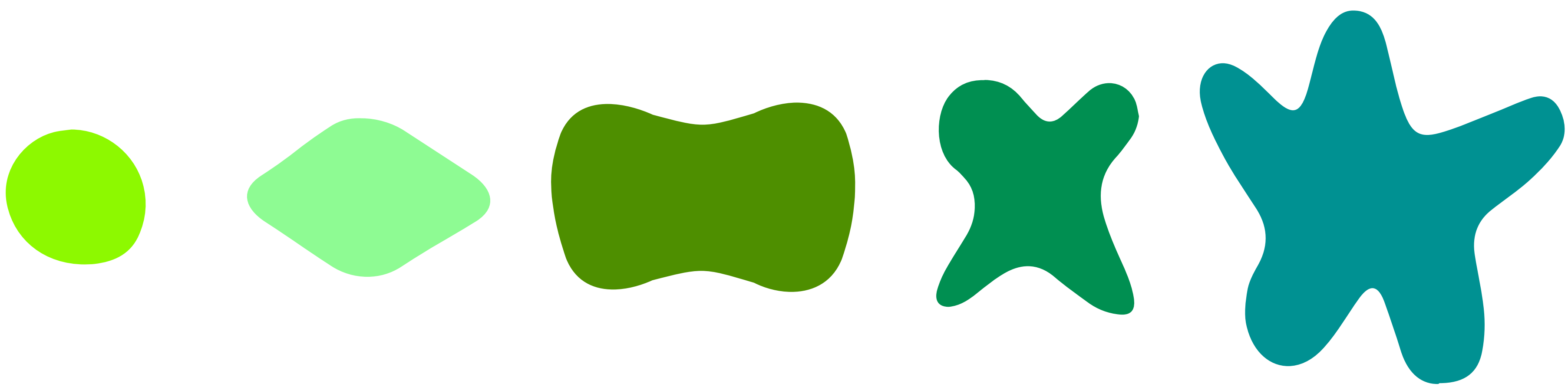
Bounded Context

Maintaining Model Integrity

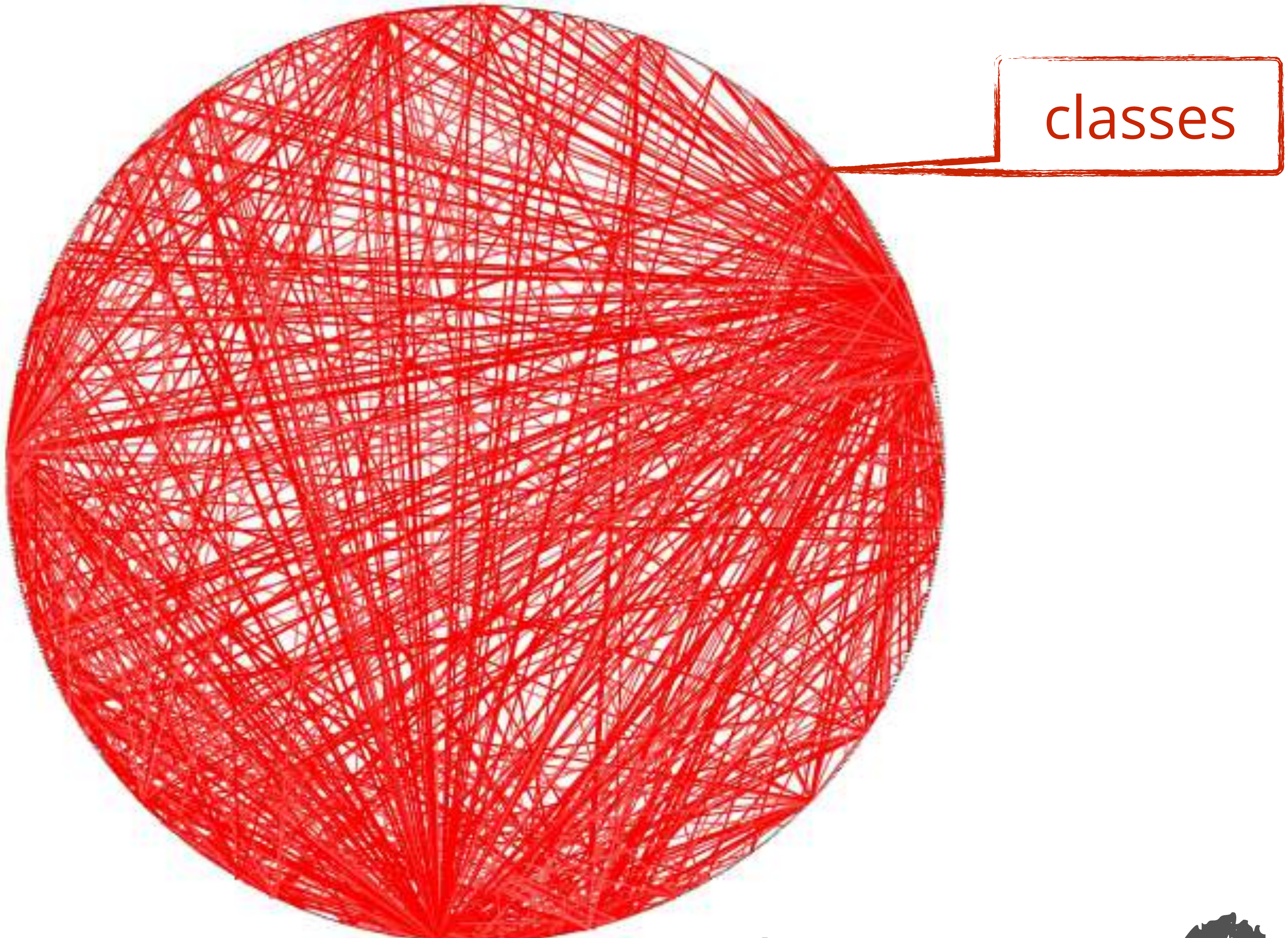




Evolvability of Architectures



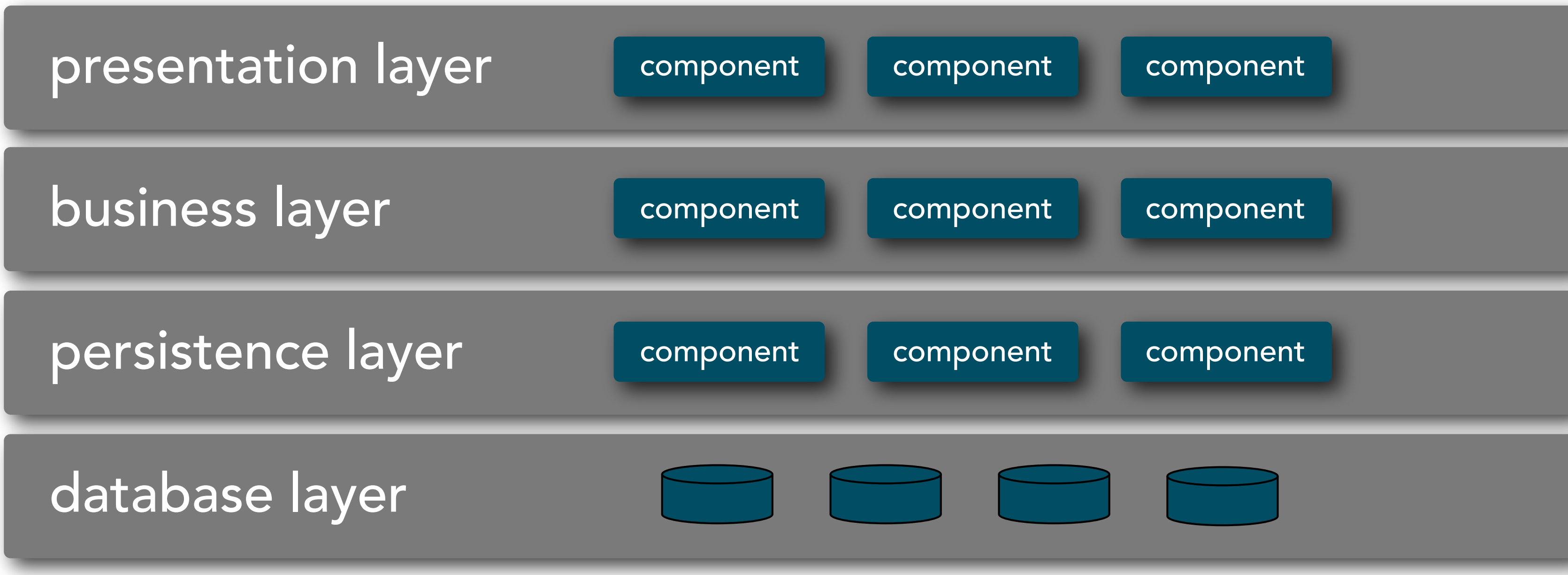
Big Ball of Mud



coupling connections

dimensions :

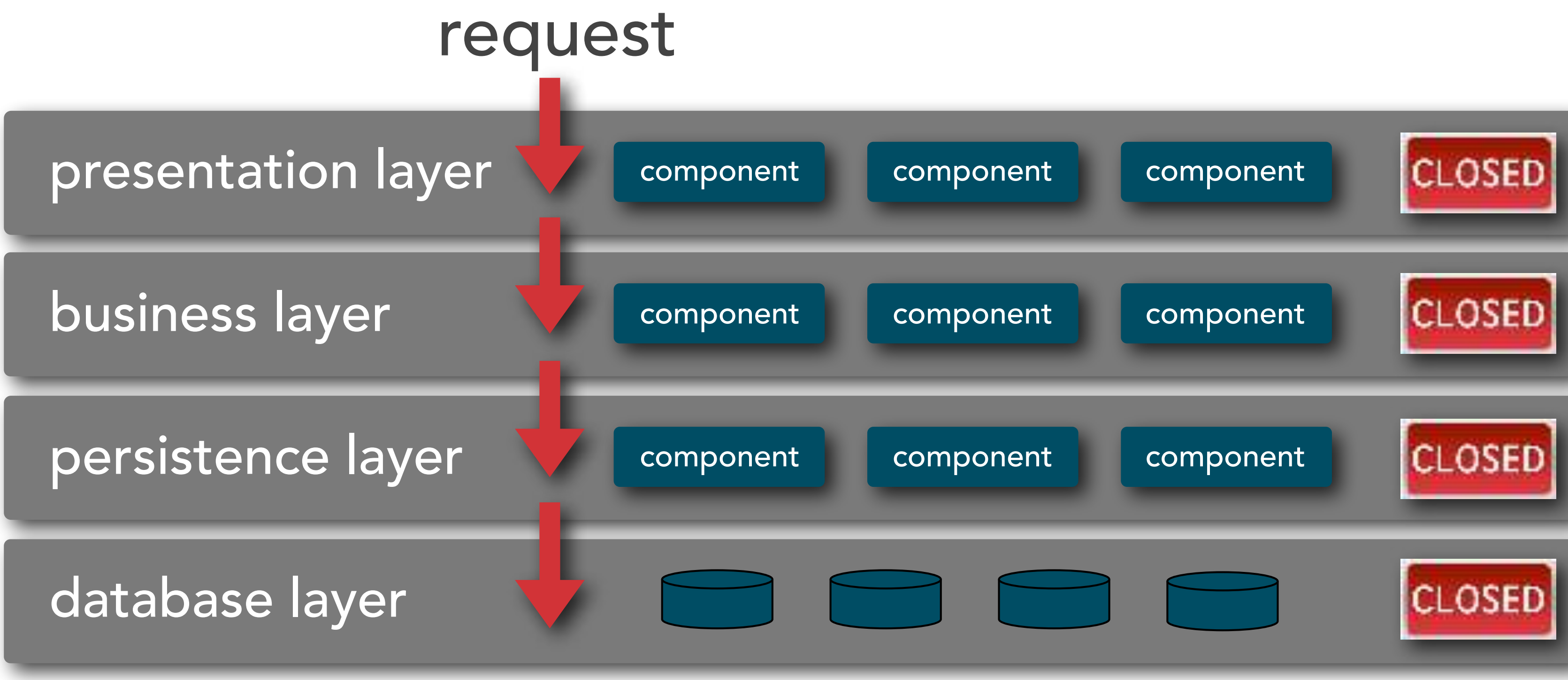
Layered Architecture



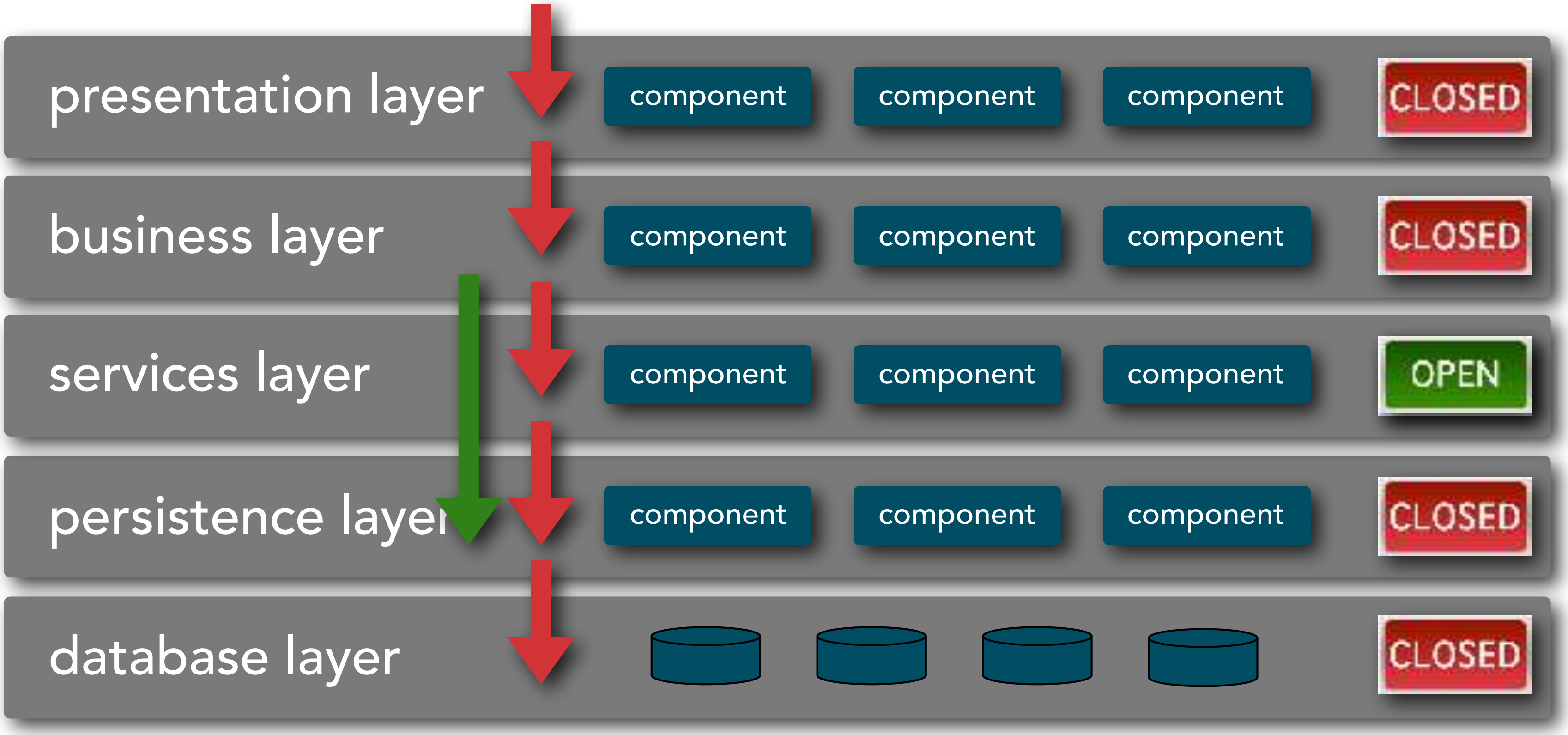
opportunities: 4

dimensions : 1

Layered Architecture



Layered Architecture



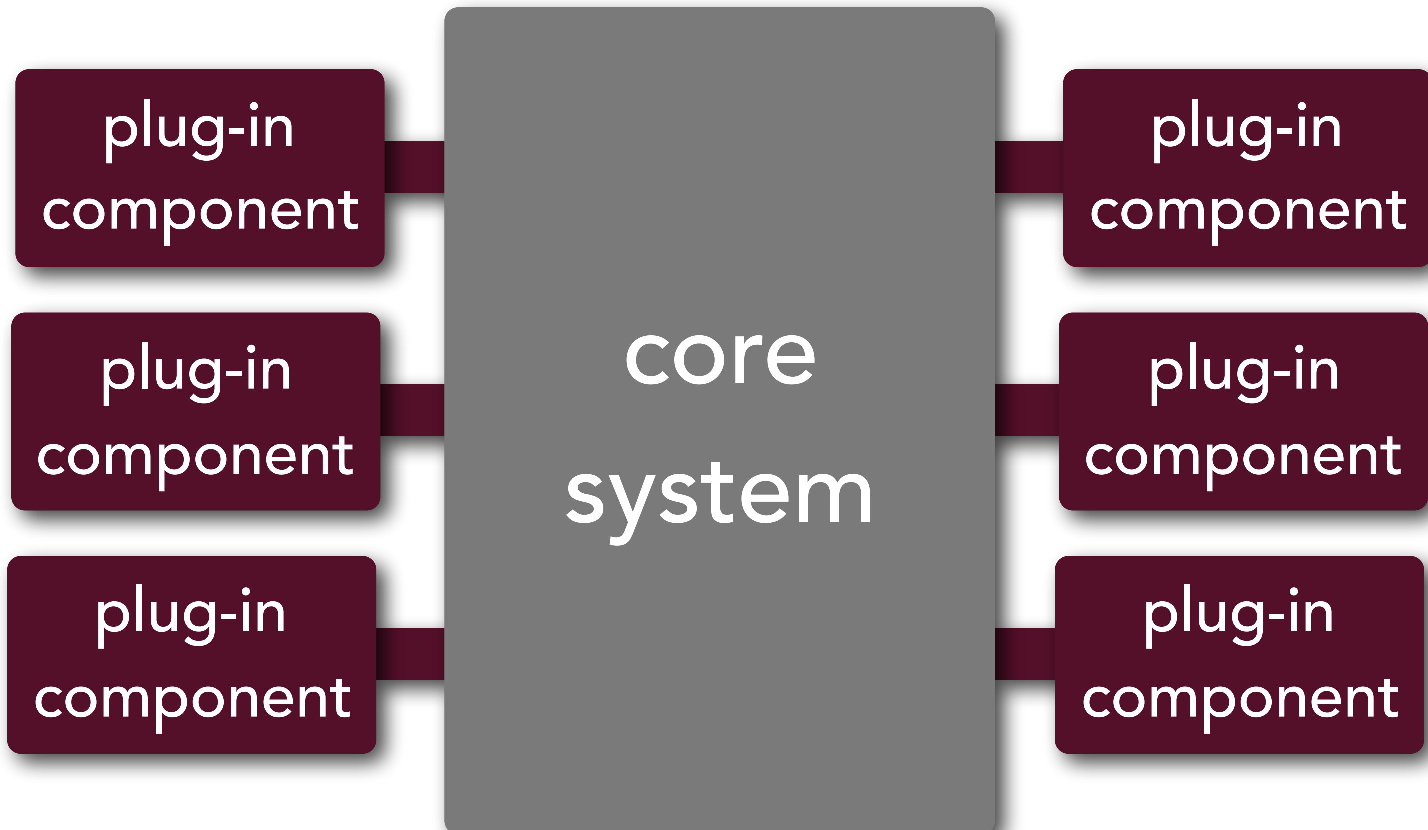
opportunities for evolution = $L - (2 \times L^o)$

L : # of layers

L^o : # of open layers

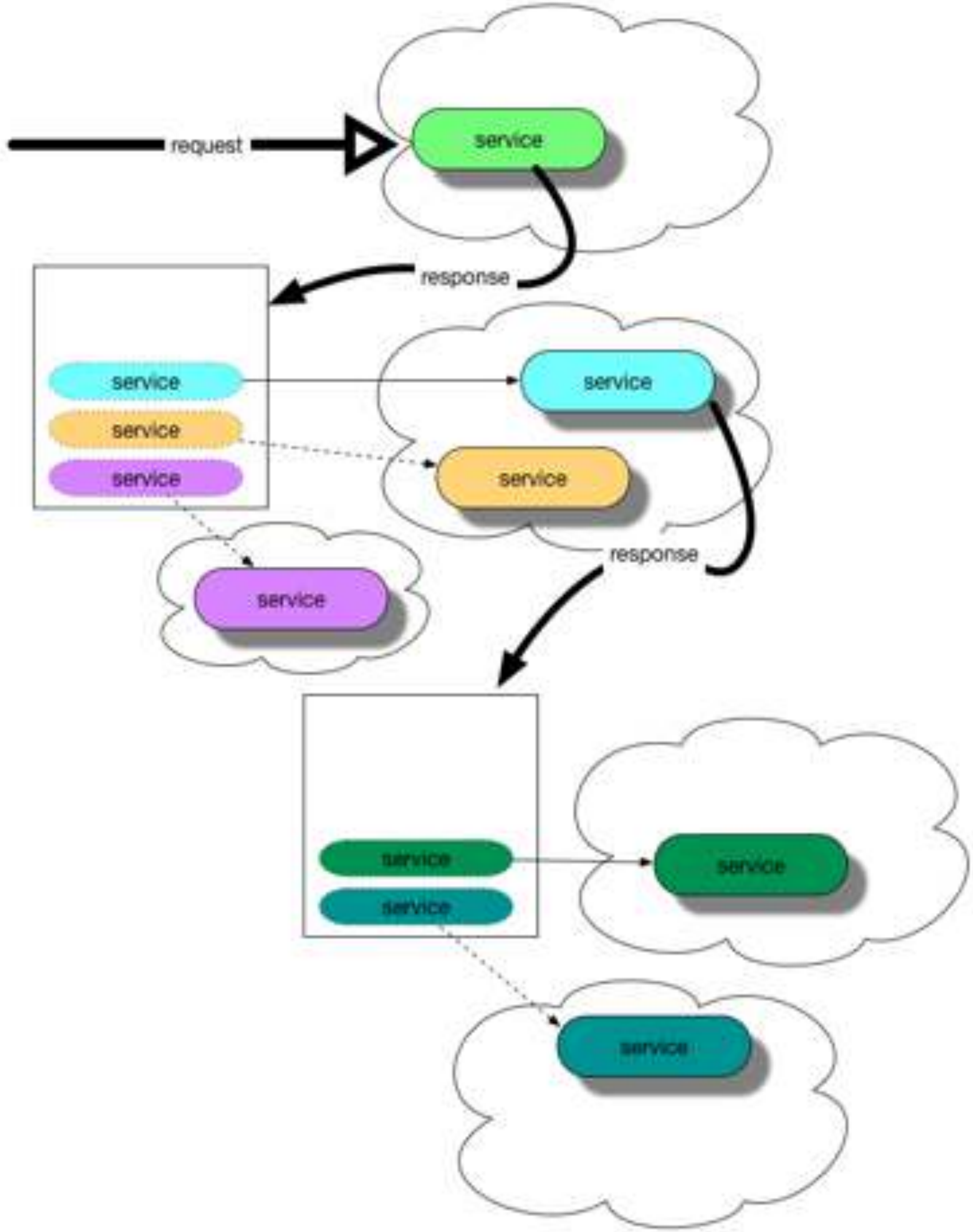
dimensions : 1

Microkernel



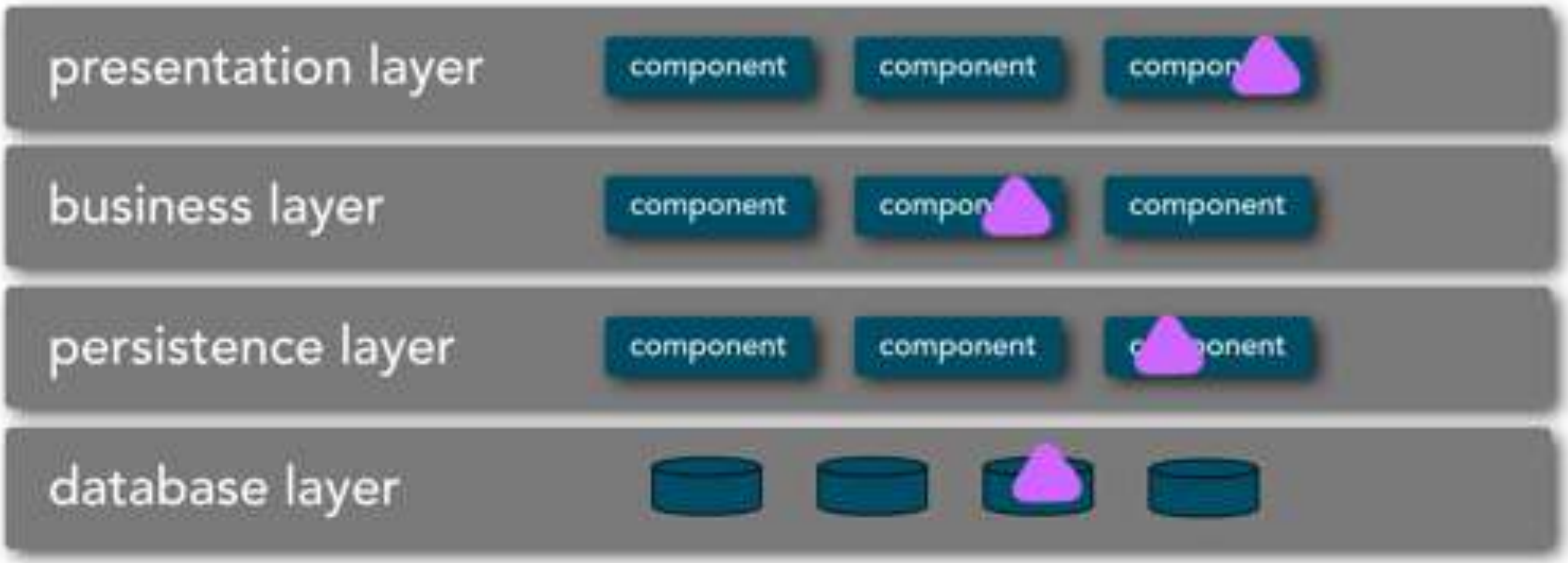
dimensions : 1

REST

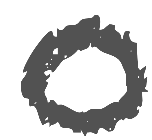


dimensions : 1

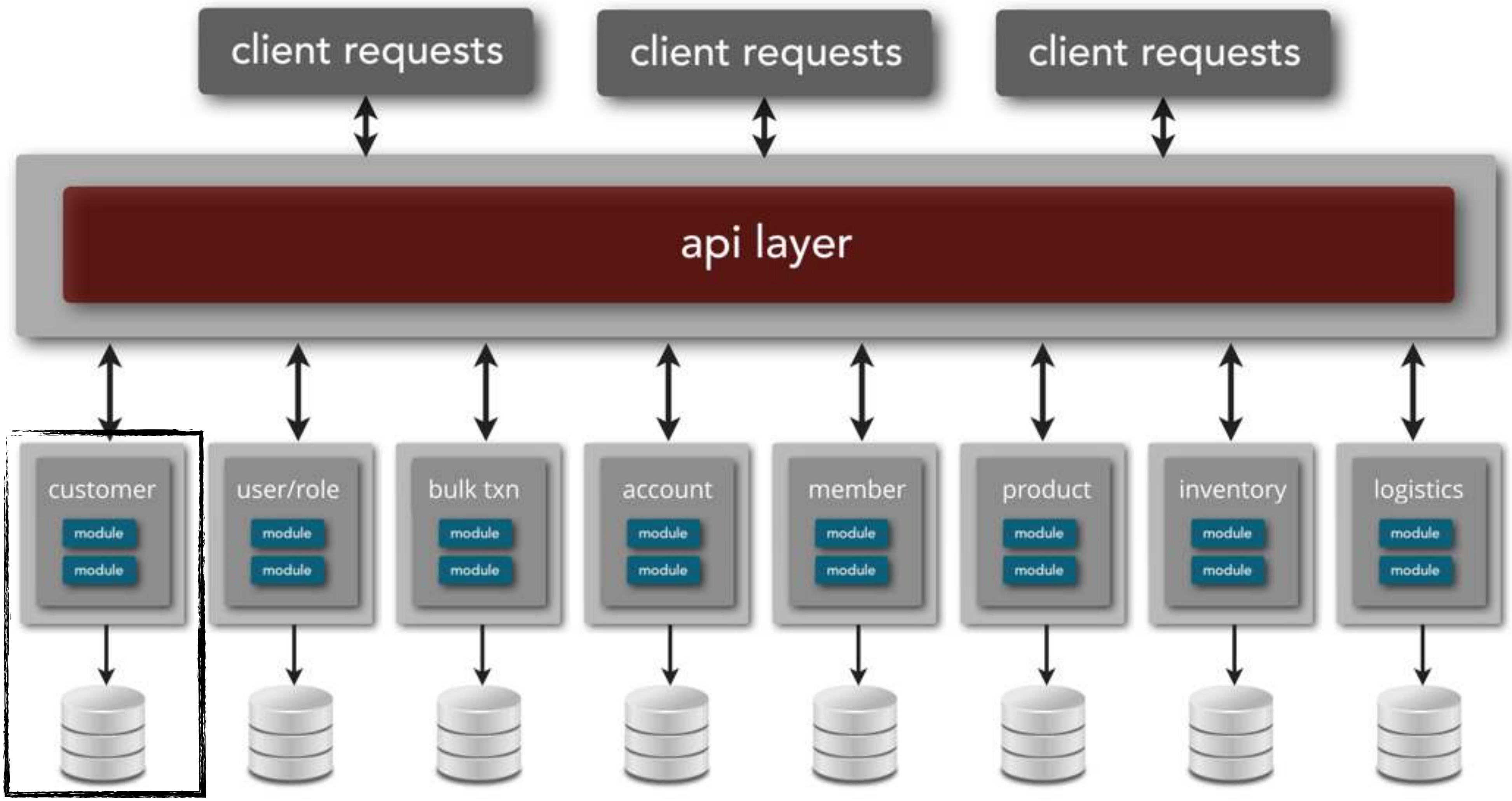
Domain Shift



domain dimensions :



Microservices



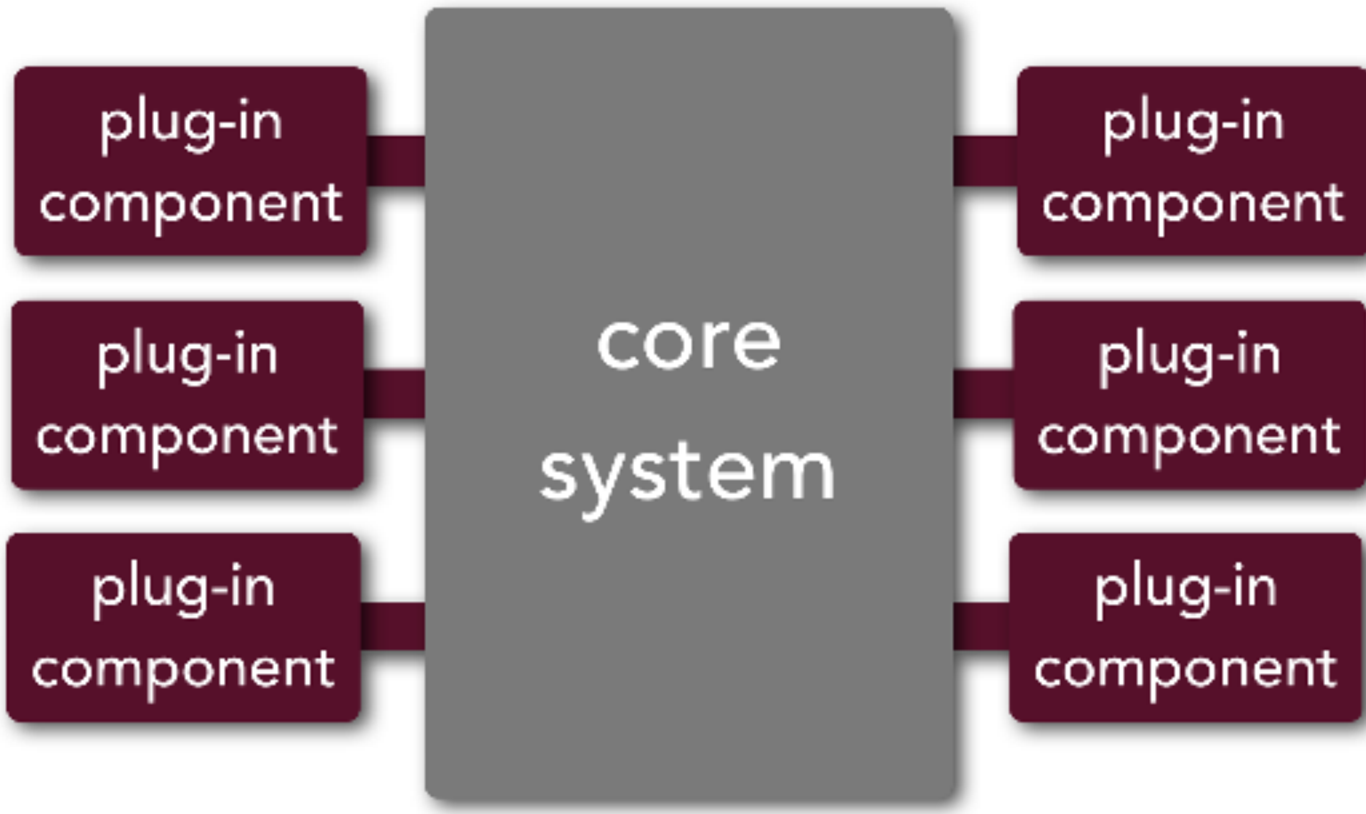
evolutionary architecture dimensions :

Definition :

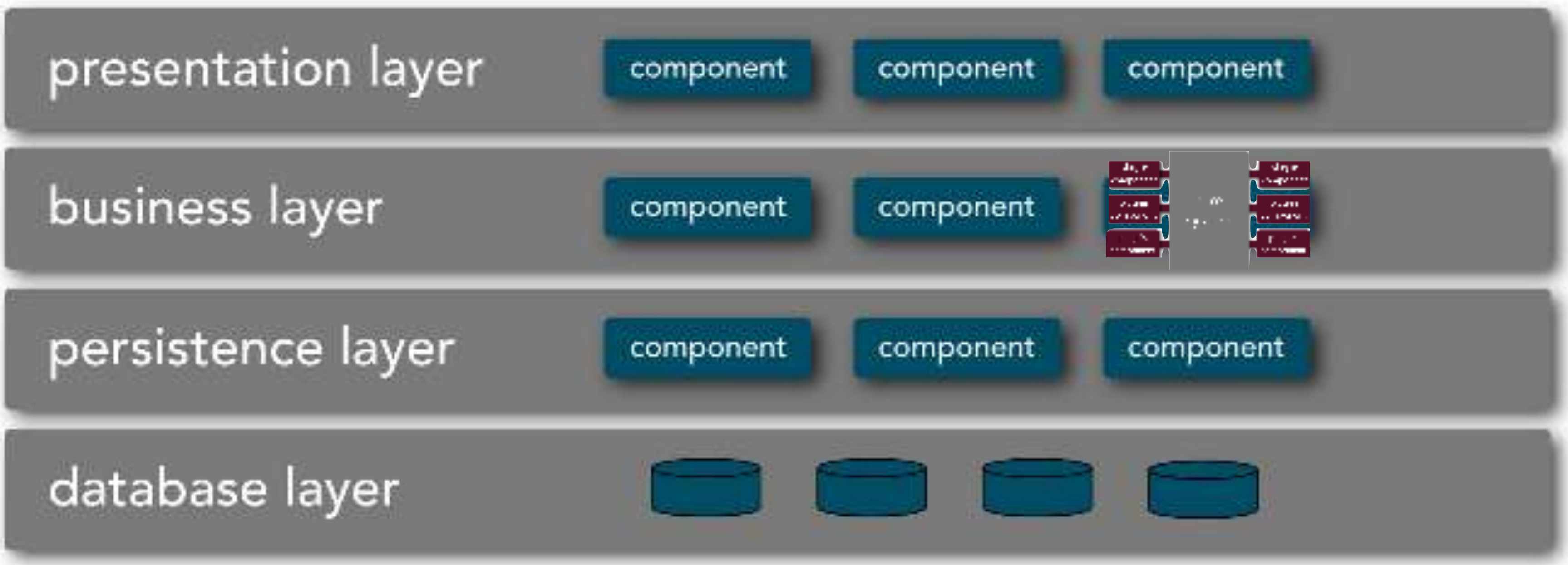
evolutionary architecture

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.

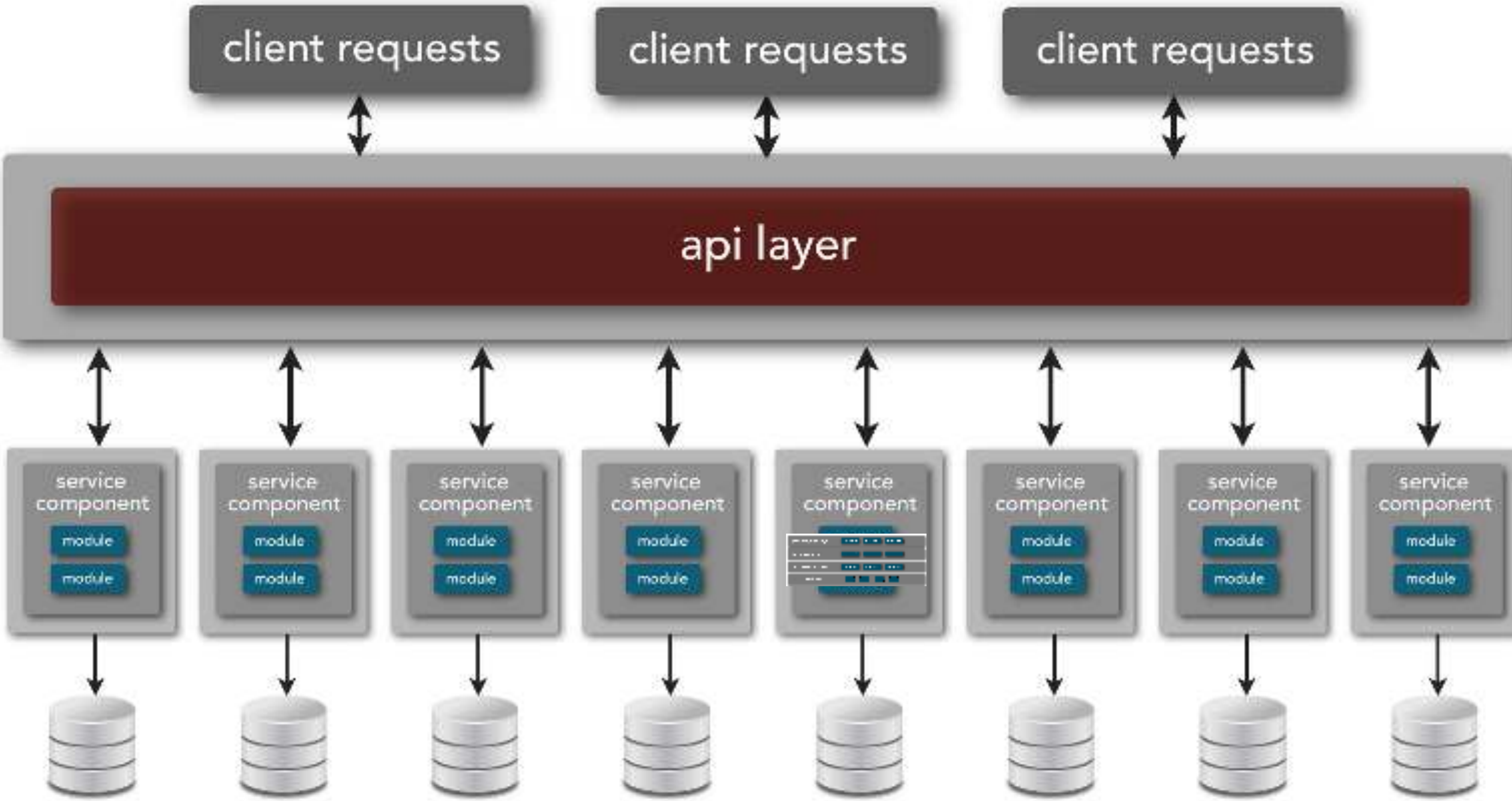
Composability



Composability



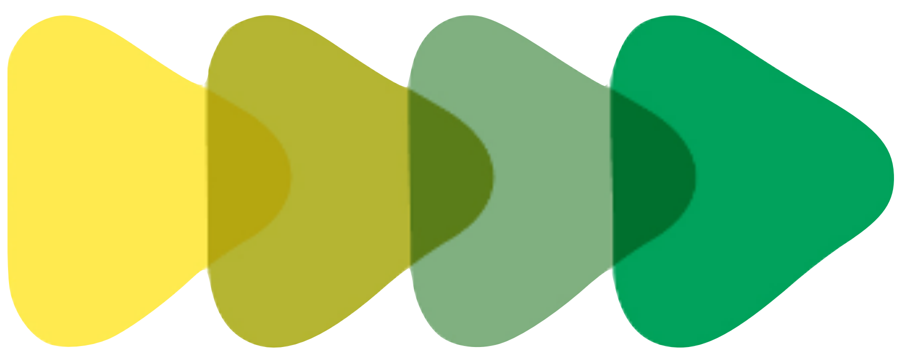
Composability



Definition :

evolutionary architecture

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.

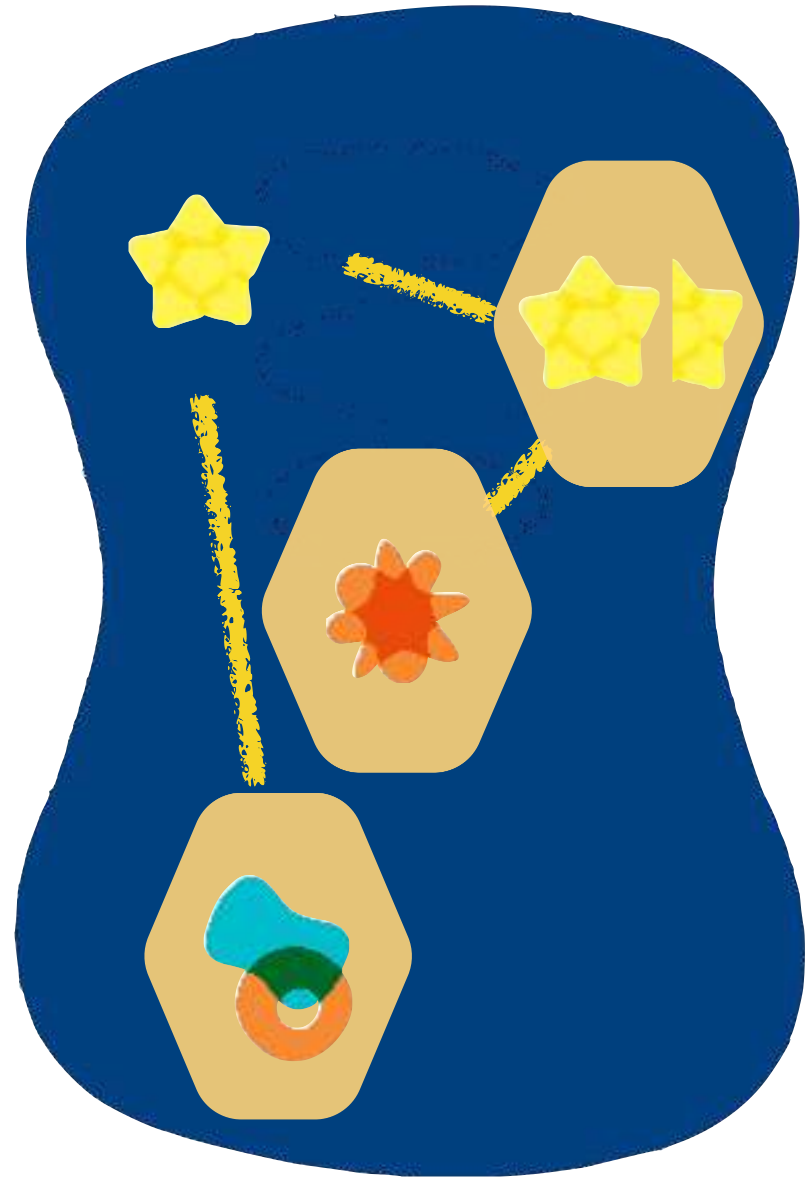


Incremental Change

Components are *deployed*.

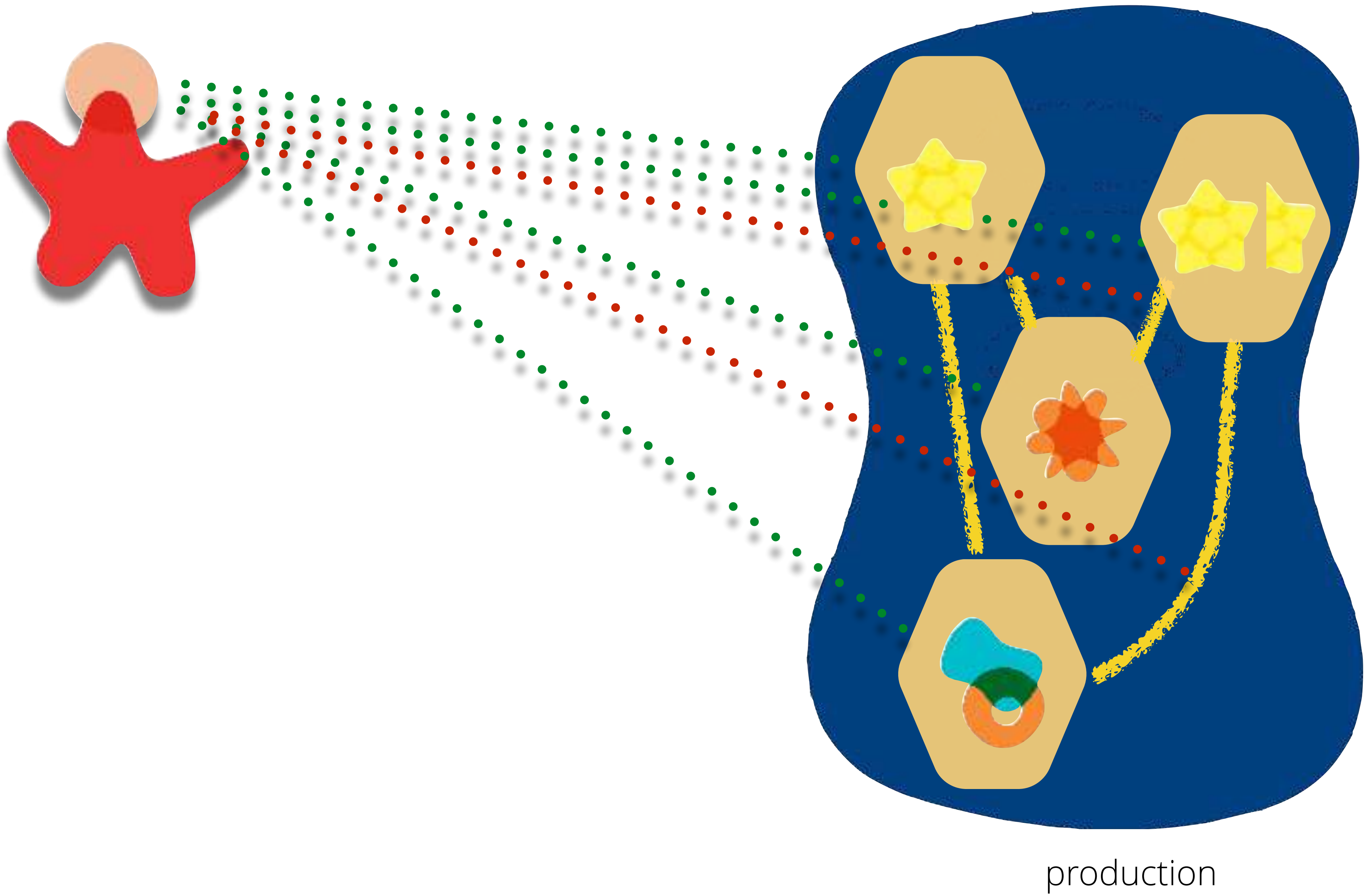
Features are *released*.

Applications consist of *routing*.



production

Incremental Change

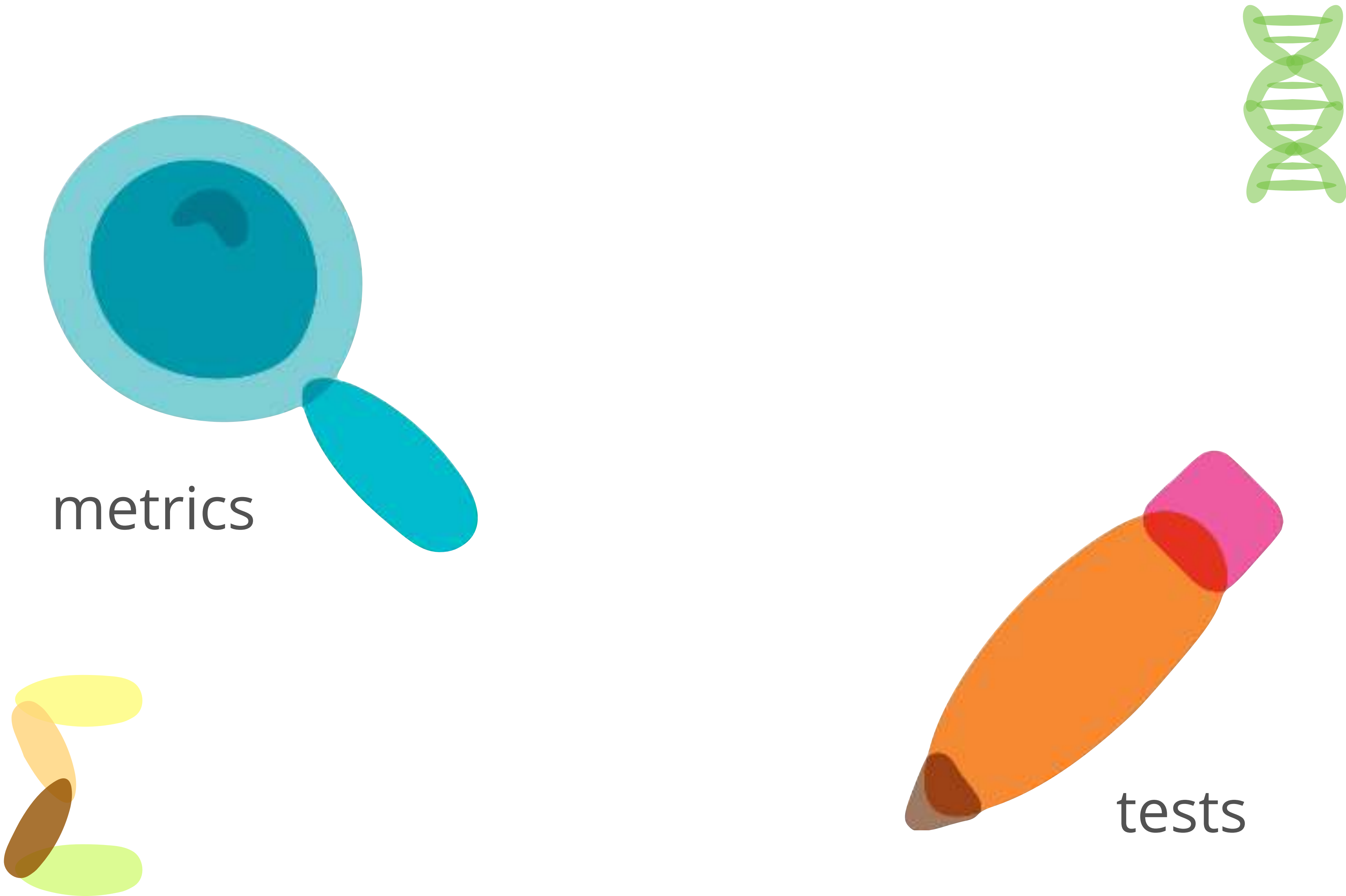


Definition :

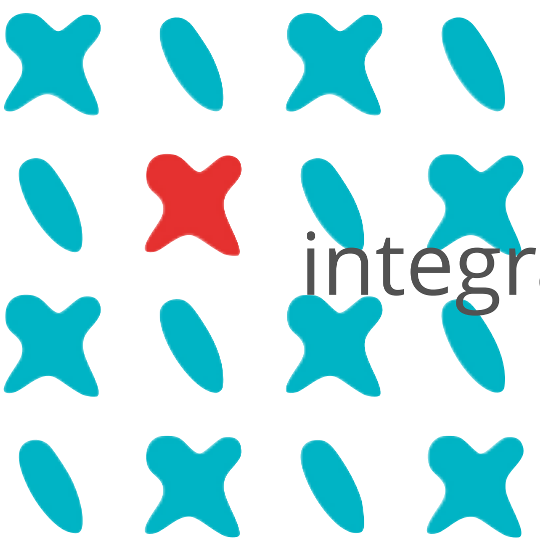
evolutionary architecture

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.

Architecture Fitness Function

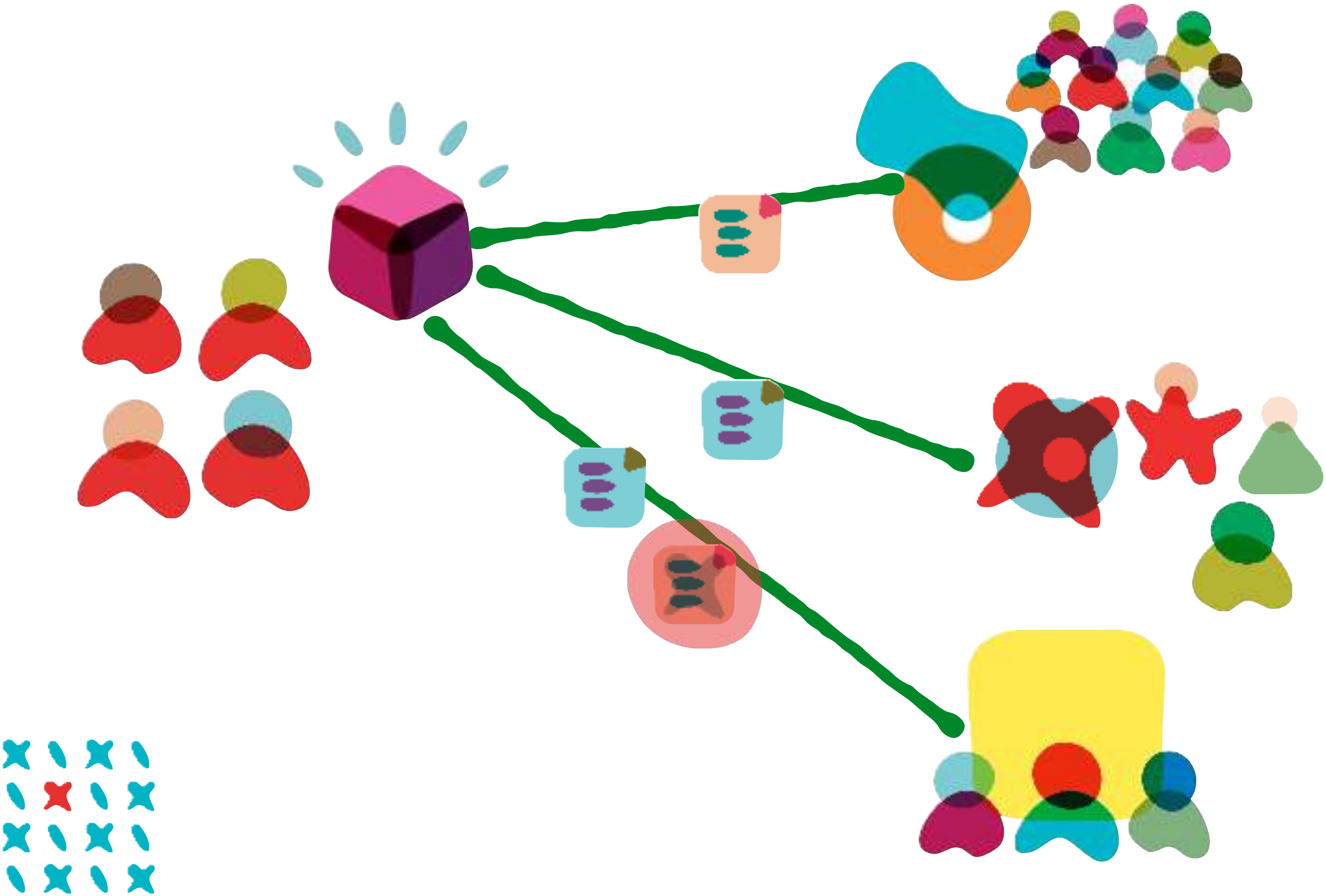


Scope



Consumer Driven Contracts

martinfowler.com/articles/consumerDrivenContracts.html



Scope

application

integration

process

Cycle Fitness Function

```java
/**
 * Tests that a package dependency cycle does not
 * exist for any of the analyzed packages.
 */
public void testAllPackages() {

    Collection packages = jdepend.analyze();

    assertEquals("Cycles exist",
                 false, jdepend.containsCycles());
}
```

clarkware.com/software/JDepend.html

Coupling Fitness Function

```java
protected void setUp() throws IOException {
    jdepend = new JDepend();
    jdepend.addDirectory("/path/to/project/util/classes");
    jdepend.addDirectory("/path/to/project/ejb/classes");
    jdepend.addDirectory("/path/to/project/web/classes");
}


public void testMatch() {
    DependencyConstraint constraint = new DependencyConstraint();
    JavaPackage ejb  = constraint.addPackage("com.xyz.ejb");
    JavaPackage web  = constraint.addPackage("com.xyz.web");
    JavaPackage util = constraint.addPackage("com.xyz.util");

    ejb.dependsUpon(util);
    web.dependsUpon(util);

    jdepend.analyze();

    assertEquals("Dependency mismatch",
             true, jdepend.dependencyMatch(constraint));
}
```

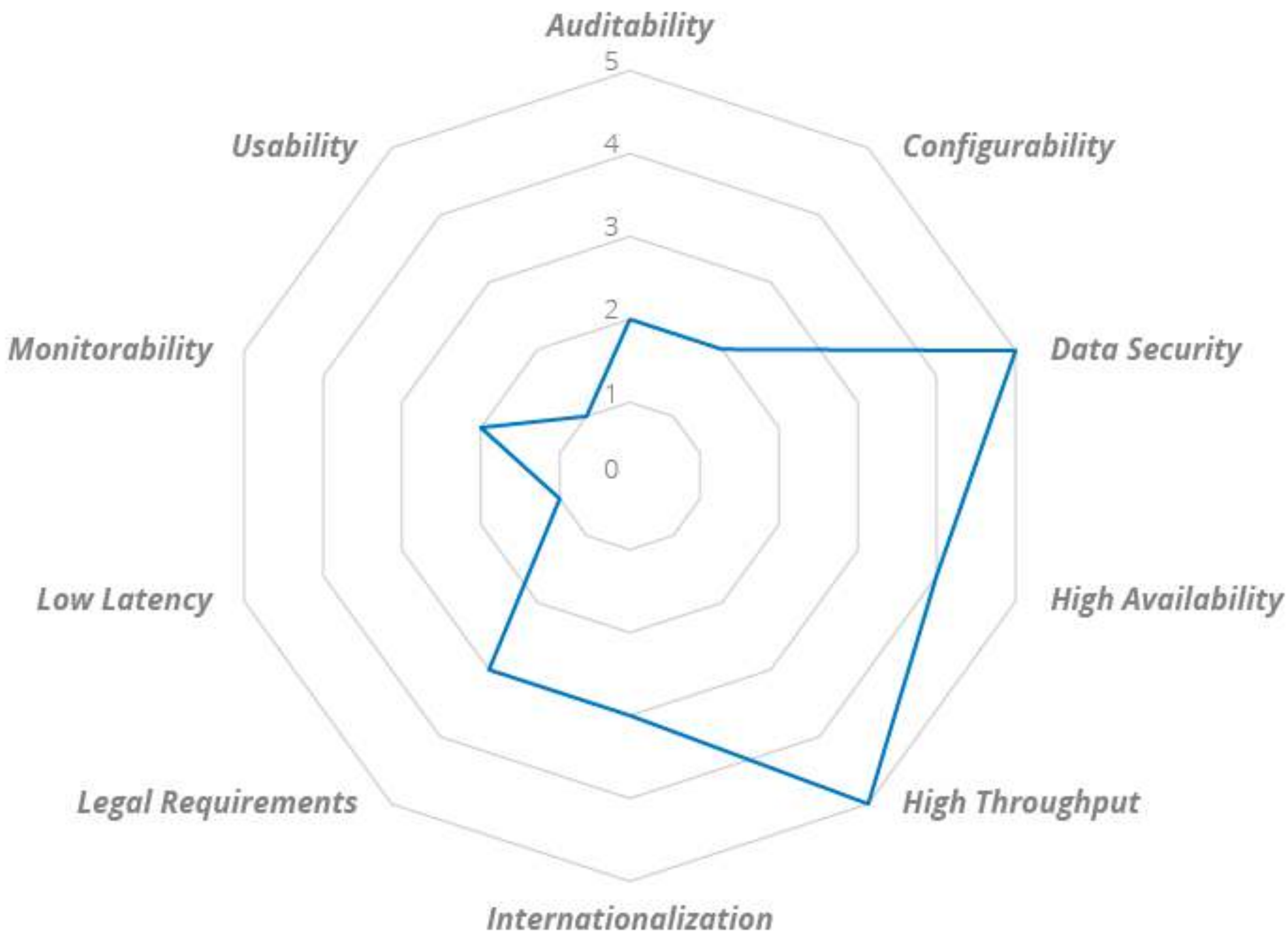
Scope

application

integration

process

Fitness Function Fit



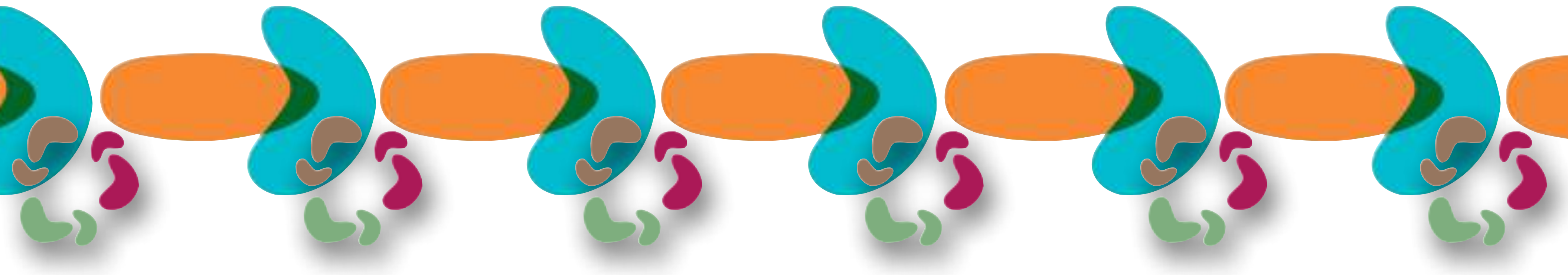
Guided Evolution

Definition :

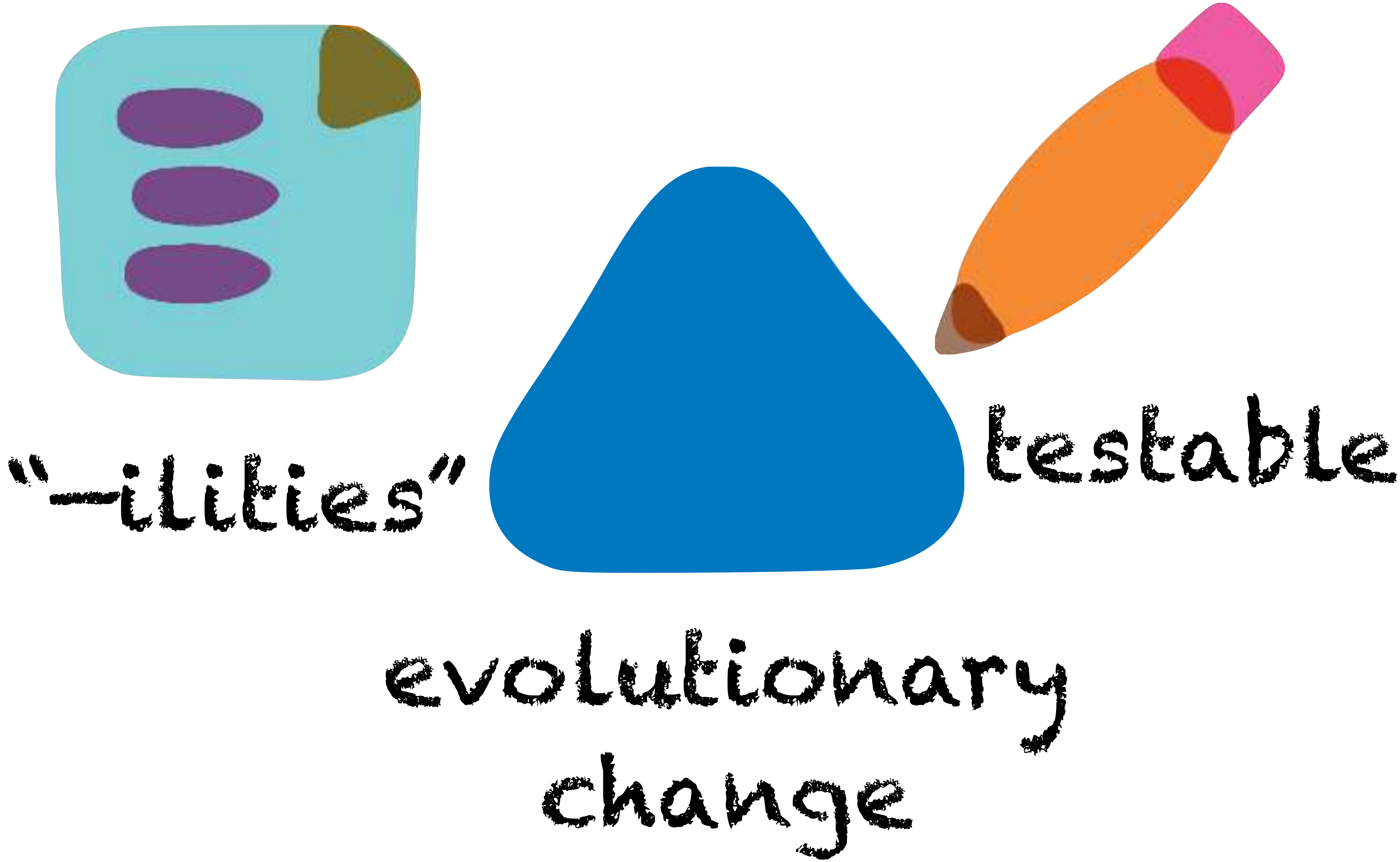
evolutionary architecture

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.

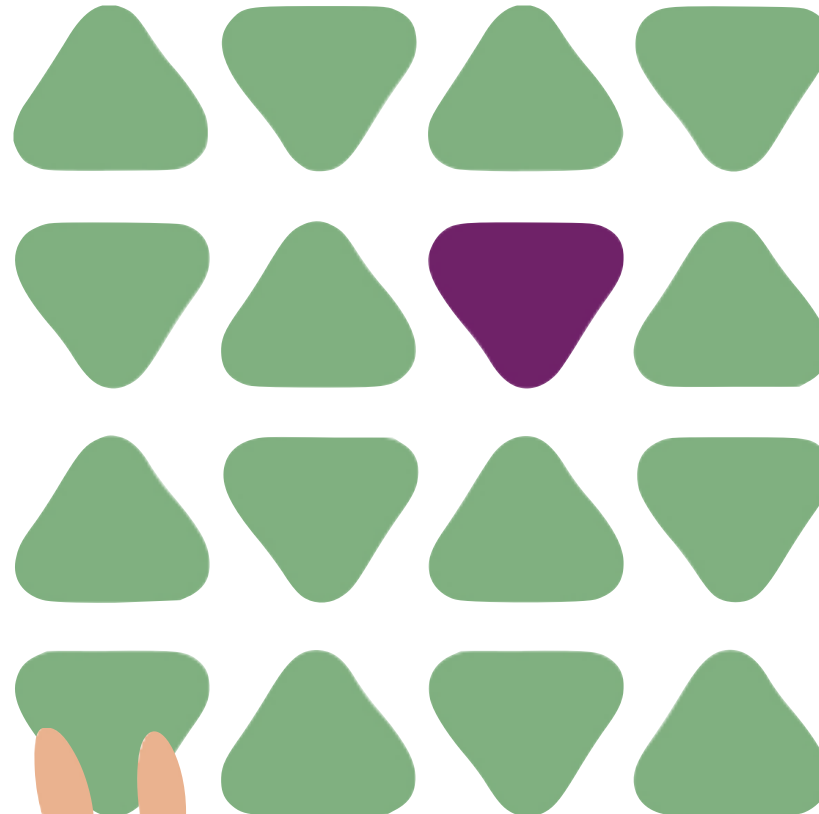
utilizing evolutionary architecture



1. choose dimensions



2. identify fitness functions

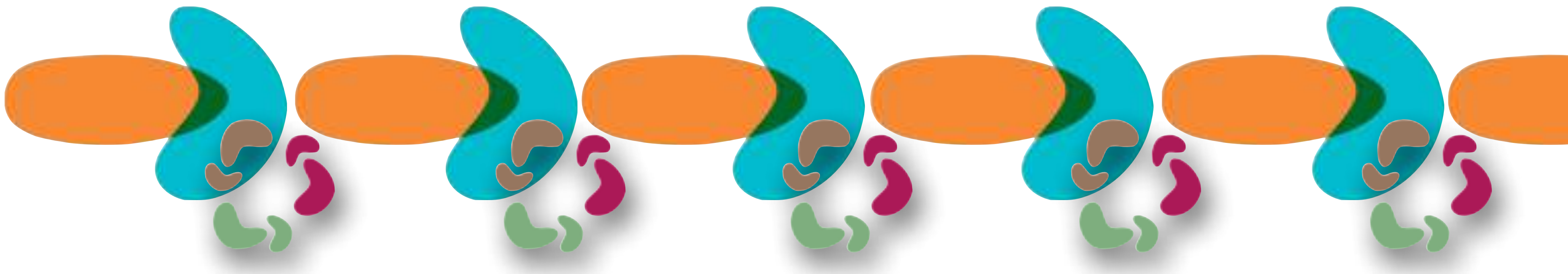


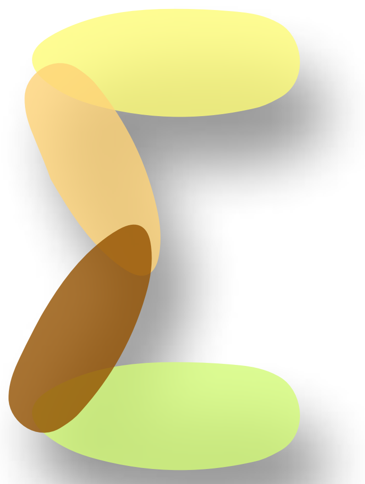
atomic

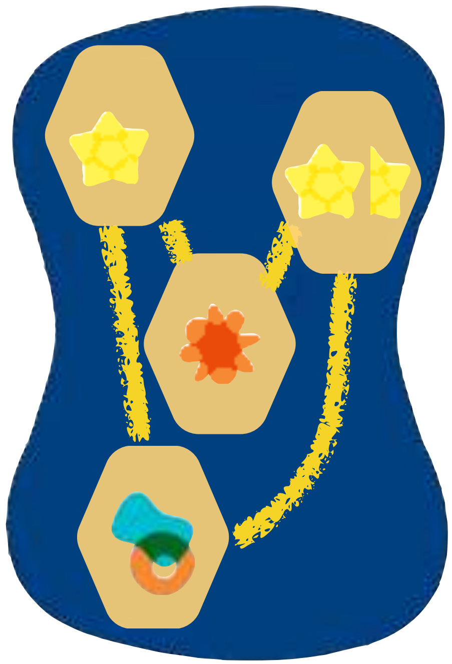
holistic

automated / manual

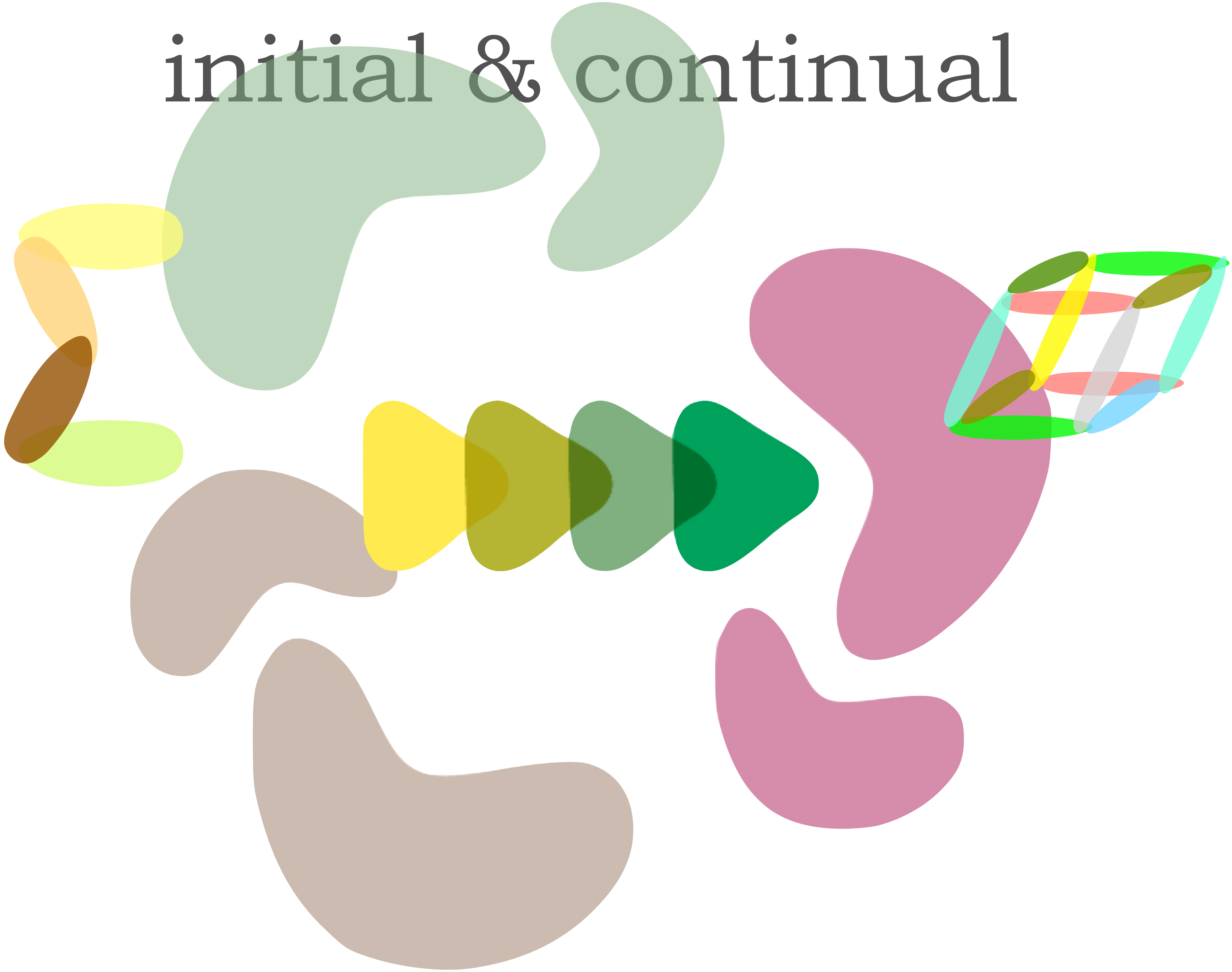
3. apply incremental change







initial & continual



architecture katas

identifying evolutionary architecture factors

Your Architectural Kata is...

Make the Grade

[illegible]

[illegible]

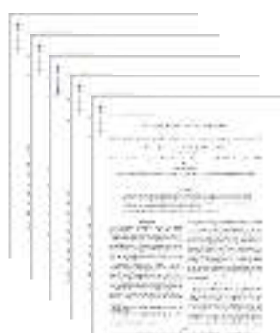
[illegible]

assigned team: 1

next steps

next steps

technical breadth

personal radar

certification
nomination packet

next steps

InfoQ

www.infoq.com

DZone

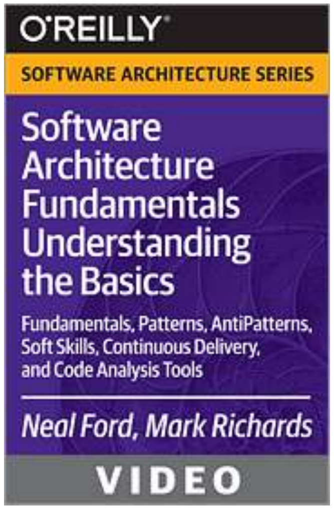
www.dzone.com

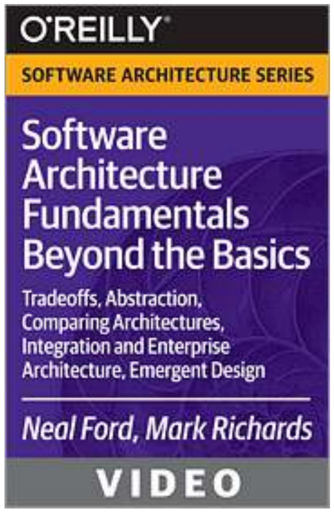
ThoughtWorks

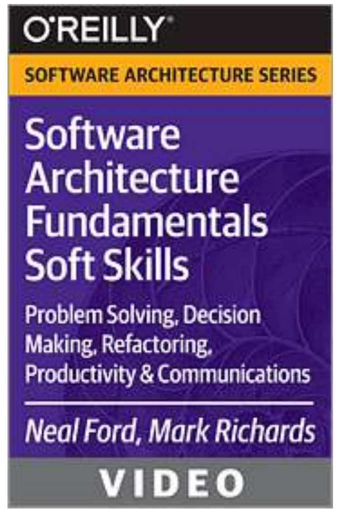
https://www.thoughtworks.com/radar

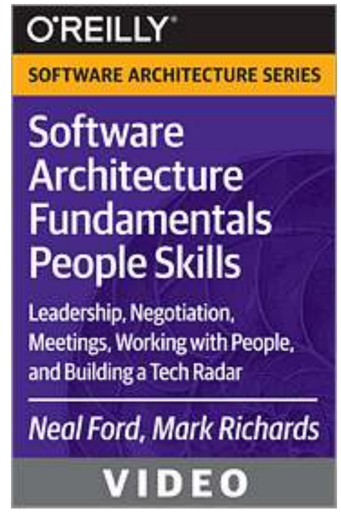
next steps

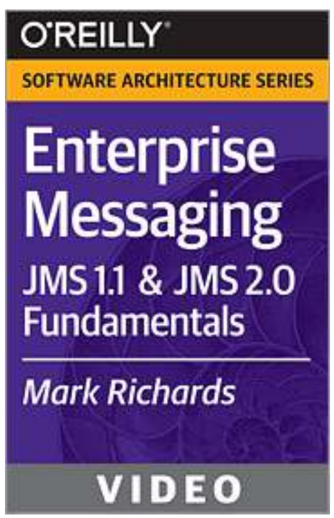
Software Architecture Fundamentals Video Series
Enterprise Messaging Video Series

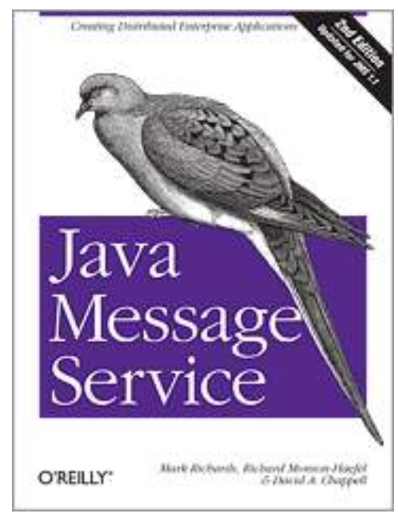




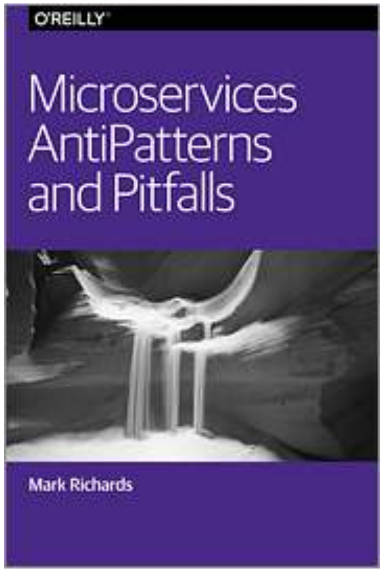


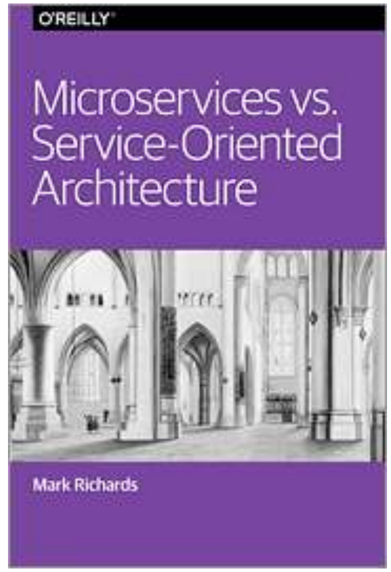






O’Reilly Free Reports





http://www.wmrichards.com/publications

Microservices AntiPatterns and Pitfalls Video

Larger Cover

Microservices AntiPatterns and Pitfalls

Learning to Avoid Costly Mistakes

By Mark Richards

Publisher: O'Reilly Media
Final Release Date: July 2016
Run time: 4 hours 9 minutes

0.0

Write a Review

Microservices is an increasingly popular architecture style that promotes scalability and ease of testing and deployment through small, highly distributed service components. It may sound like the correct architecture for your situation, but if you're new to microservices, how do you really know? Understanding microservices'...

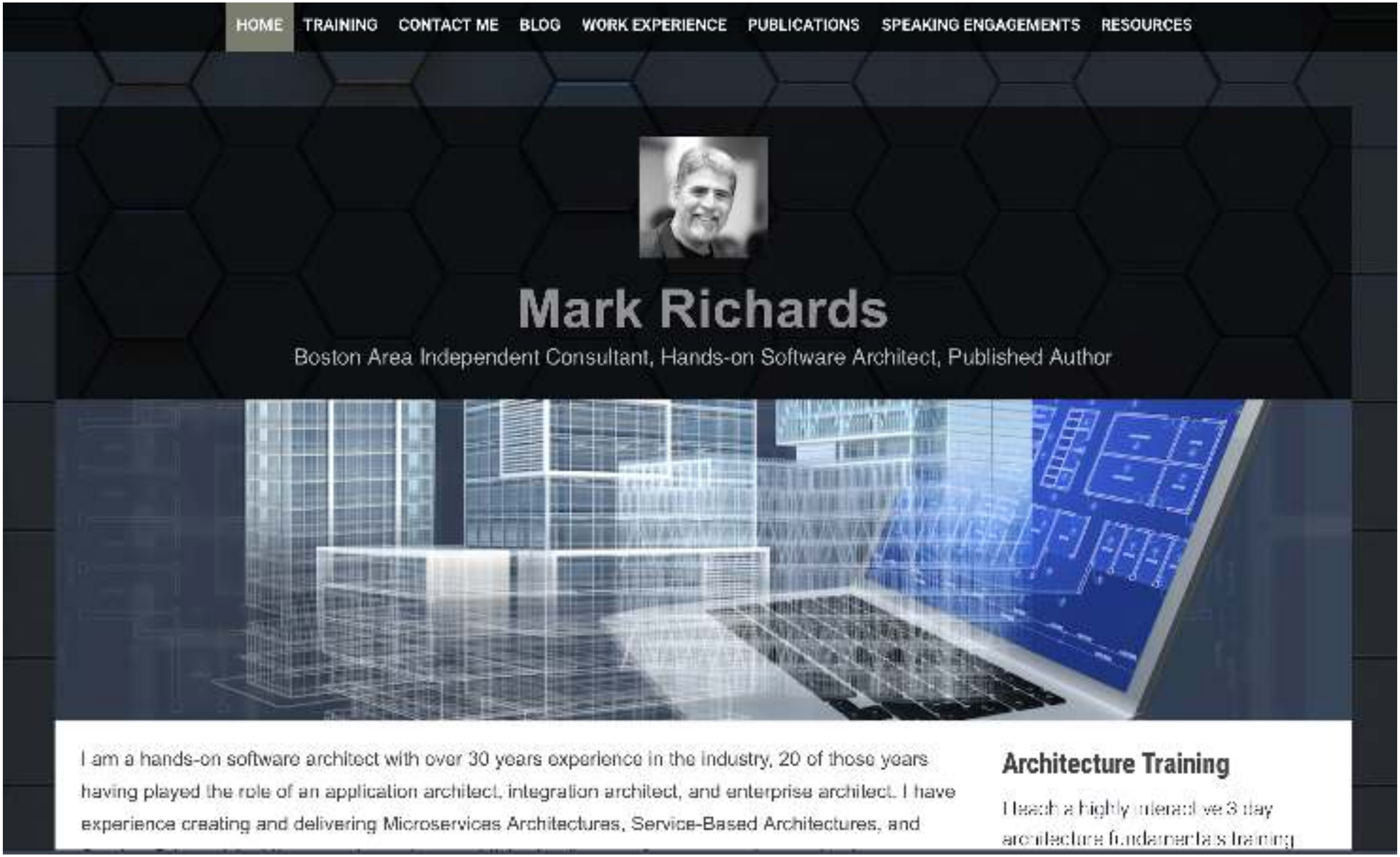
Full description

Start This Video Training for Free

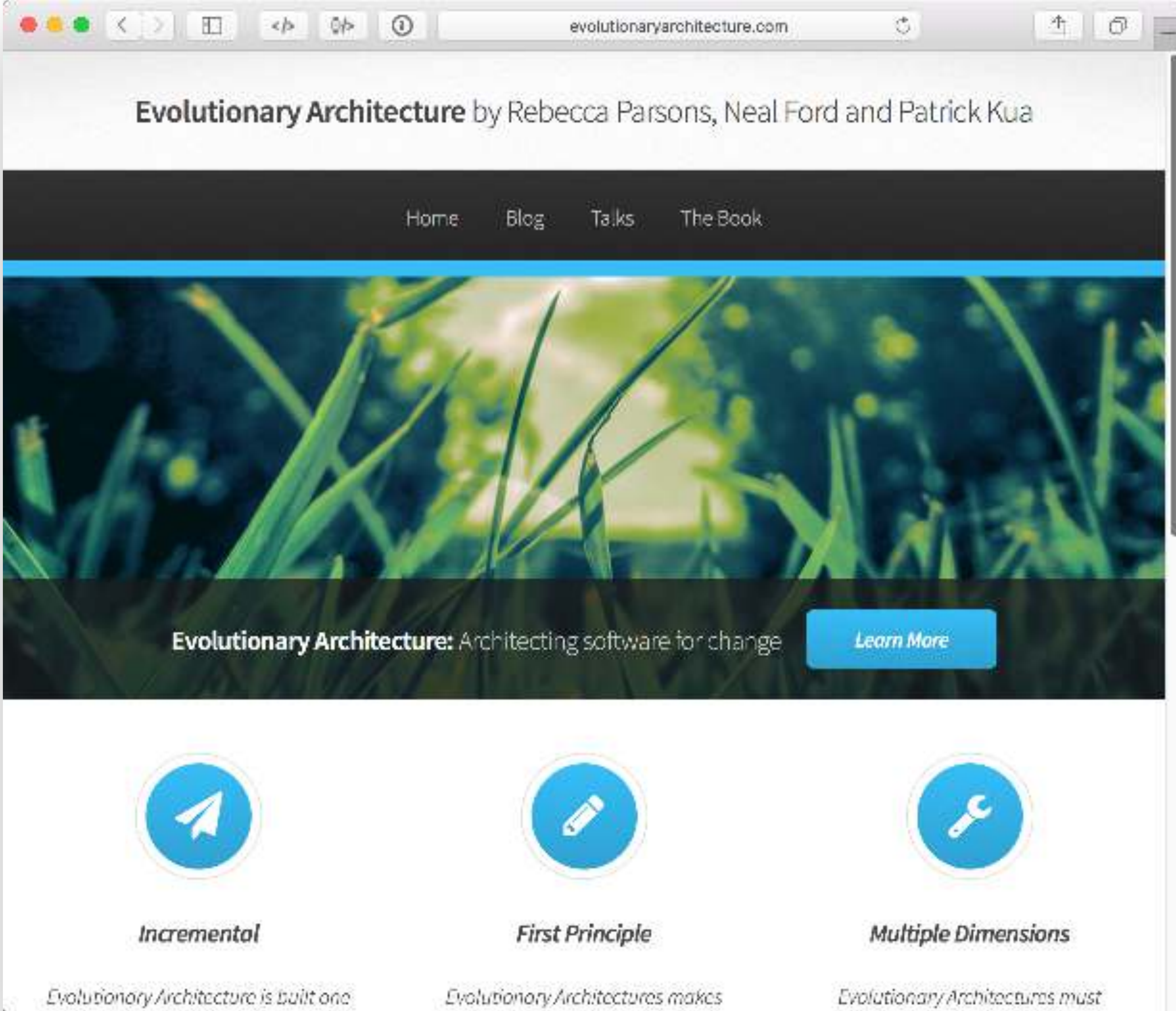
View the links in the TOC below.

http://shop.oreilly.com/product/0636920052876.do#

www.wmrichards.com




Presentation PATTERNS
O'REILLY
Functional Thinking
Neal Ford
nealford.com
@neal4d
ThoughtWorks
NEAL FORD
ART of JAVA WEB DEVELOPMENT
JBuilder 3 Unleashed
Developing with Delphi
O'REILLY
SOFTWARE ARCHITECTURE SERIES
Agile Engineering Practices
Neal Ford
VIDEO
Clojure Inside Out
Stuart Halloway & Neal Ford
VIDEO
Functional Thinking: Functional programming using Java, Clojure & Scala
Neal Ford
VIDEO
O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Understanding the Basics
Neal Ford, Mark Richards
VIDEO
O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Beyond the Basics
Neal Ford, Mark Richards
VIDEO
O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Soft Skills
Problem Solving, Decision Making, Refactoring, Productivity & Communications
Neal Ford, Mark Richards
VIDEO
O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals People Skills
Neal Ford, Mark Richards
VIDEO
O'REILLY
SOFTWARE ARCHITECTURE SERIES
Software Architecture Fundamentals Service-Based Architectures
Structure, Engineering Practices, and Migration
Neal Ford, Mark Richards
VIDEO
O'REILLY
SOFTWARE ARCHITECTURE SERIES
Engineering Practices for Continuous Delivery
Neal Ford
VIDEO



http://evolutionaryarchitecture.com

Creating Software Architectures

Neal Ford
ThoughtWorks
Director / Software Architect / Meme Wrangler
http://www.nealford.com
@neal4d

Mark Richards
Hands-on Software Architect
Published Author / Conference Speaker
http://www.wmrichards.com
https://www.linkedin.com/in/markrichards3
@markrichardssa

SDD DEEP DIVE 2016
Software Design & Development
London, 8-10 November 2016