Microservices Architecture and Design

Mark Richards

Independent Consultant

Hands-on Software Architect / Published Author

Founder, DeveloperToArchitect.com

http://www.wmrichards.com

https://www.linkedin.com/in/markrichards3

@markrichardssa

Apache ZooKeeper™

CONSUL

ANSIBLE

cloudstack
open source cloud computing

docker

CHEF™

kubernetes
by Google

Dropwizard

Prometheus

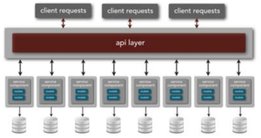
spring
boot

gain a full understanding of microservices from an architecture and design standpoint

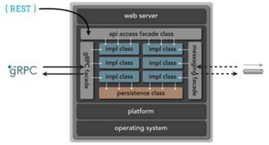
gain an understanding of the implications and challenges of the microservices architecture style

validate whether microservices is the right choice for your company and understand the alternatives

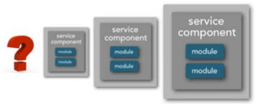
course agenda



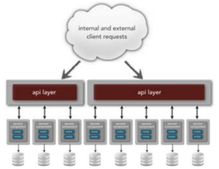
architecture characteristics



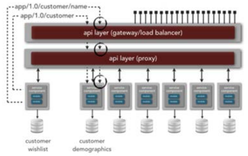
service design



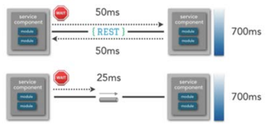
service granularity



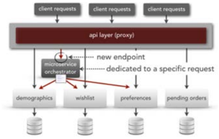
api gateways



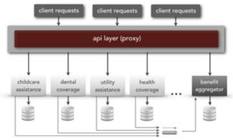
communication challenges



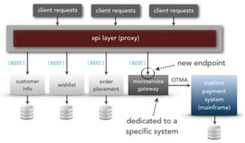
event-driven microservices



service orchestration

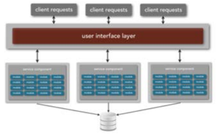


service aggregation

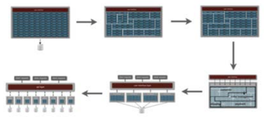


service adapter

distributed transactions



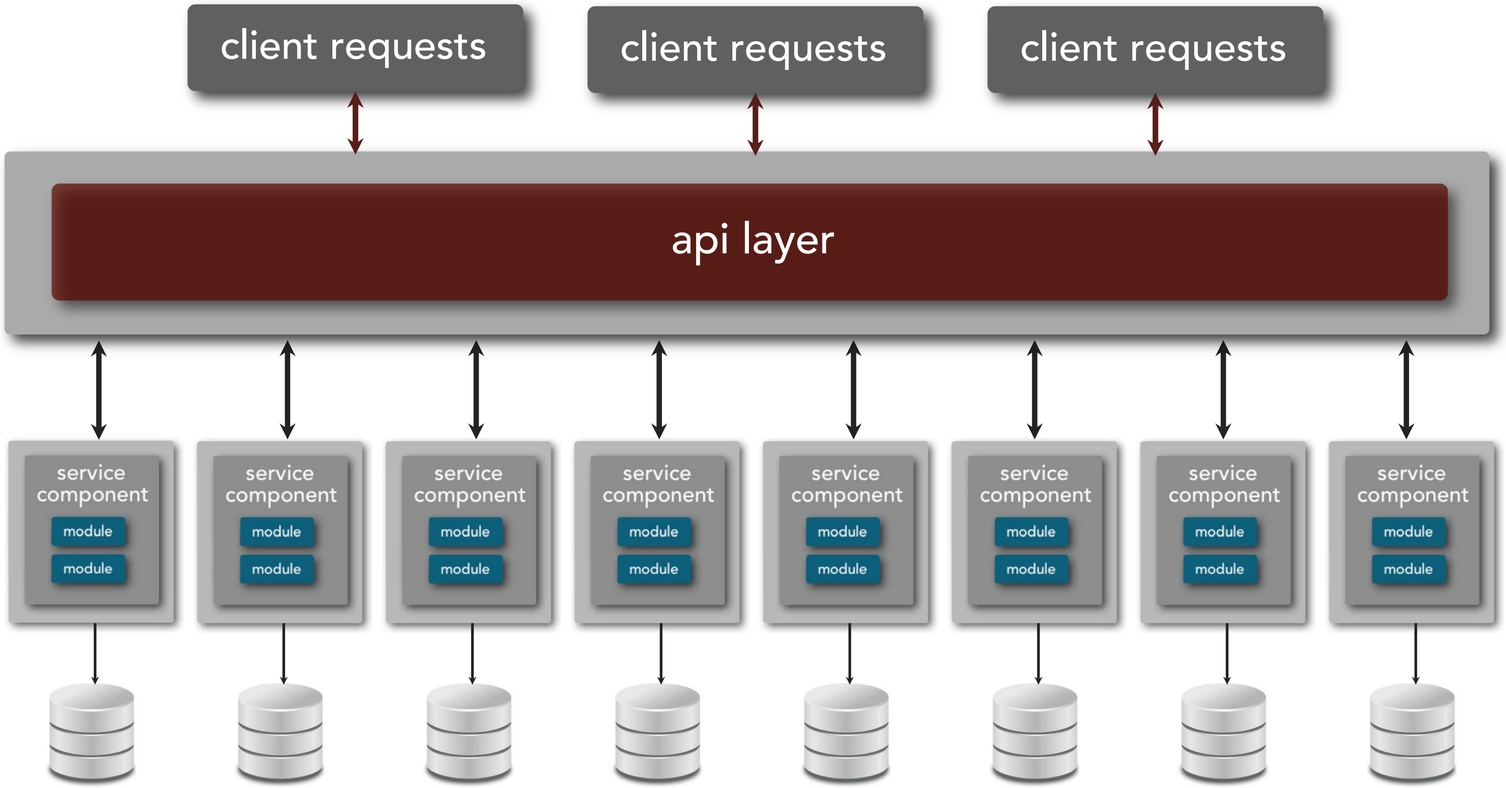
hybrid architectures



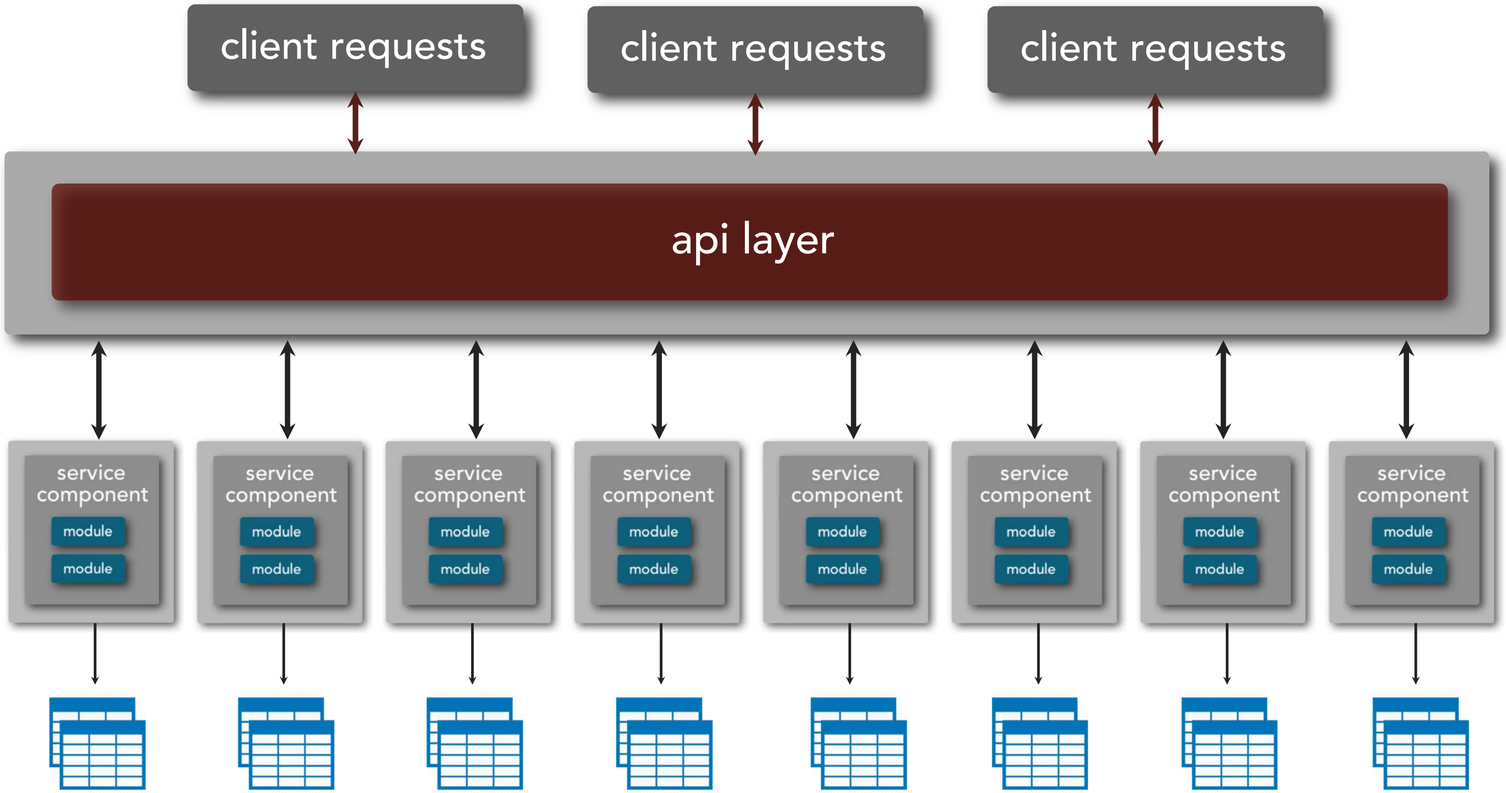
migration patterns

Architecture Characteristics

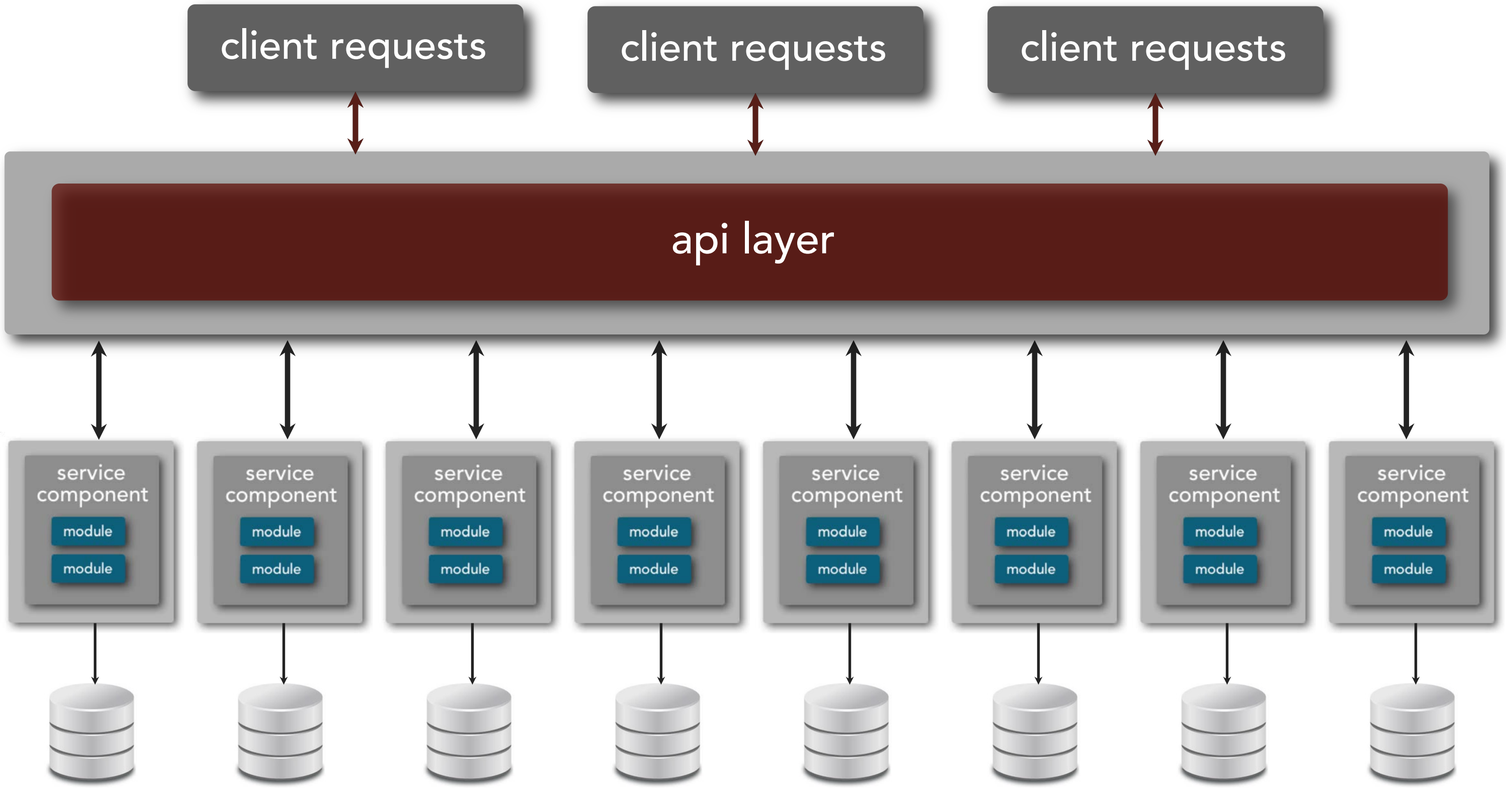
architecture characteristics



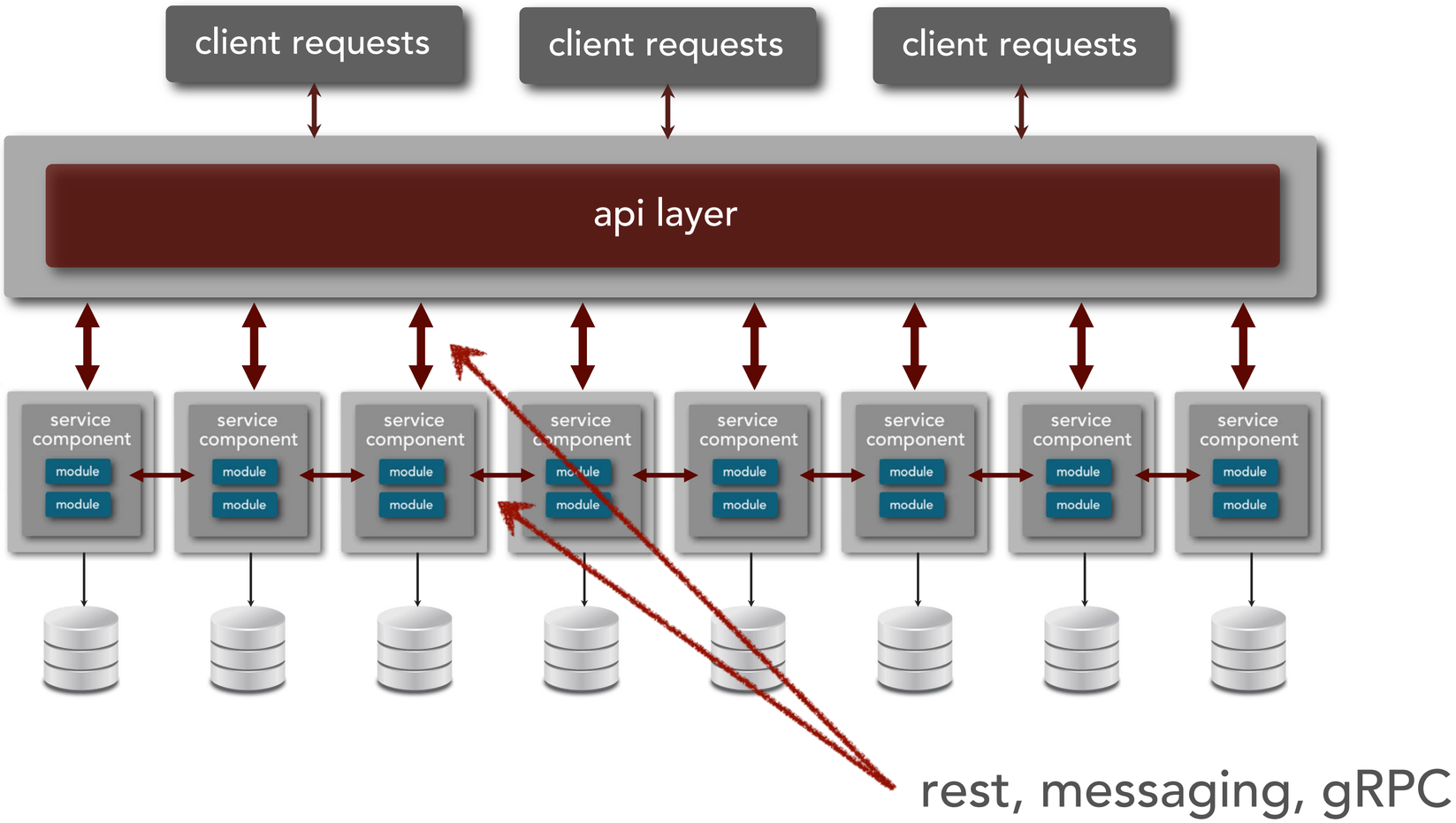
architecture characteristics



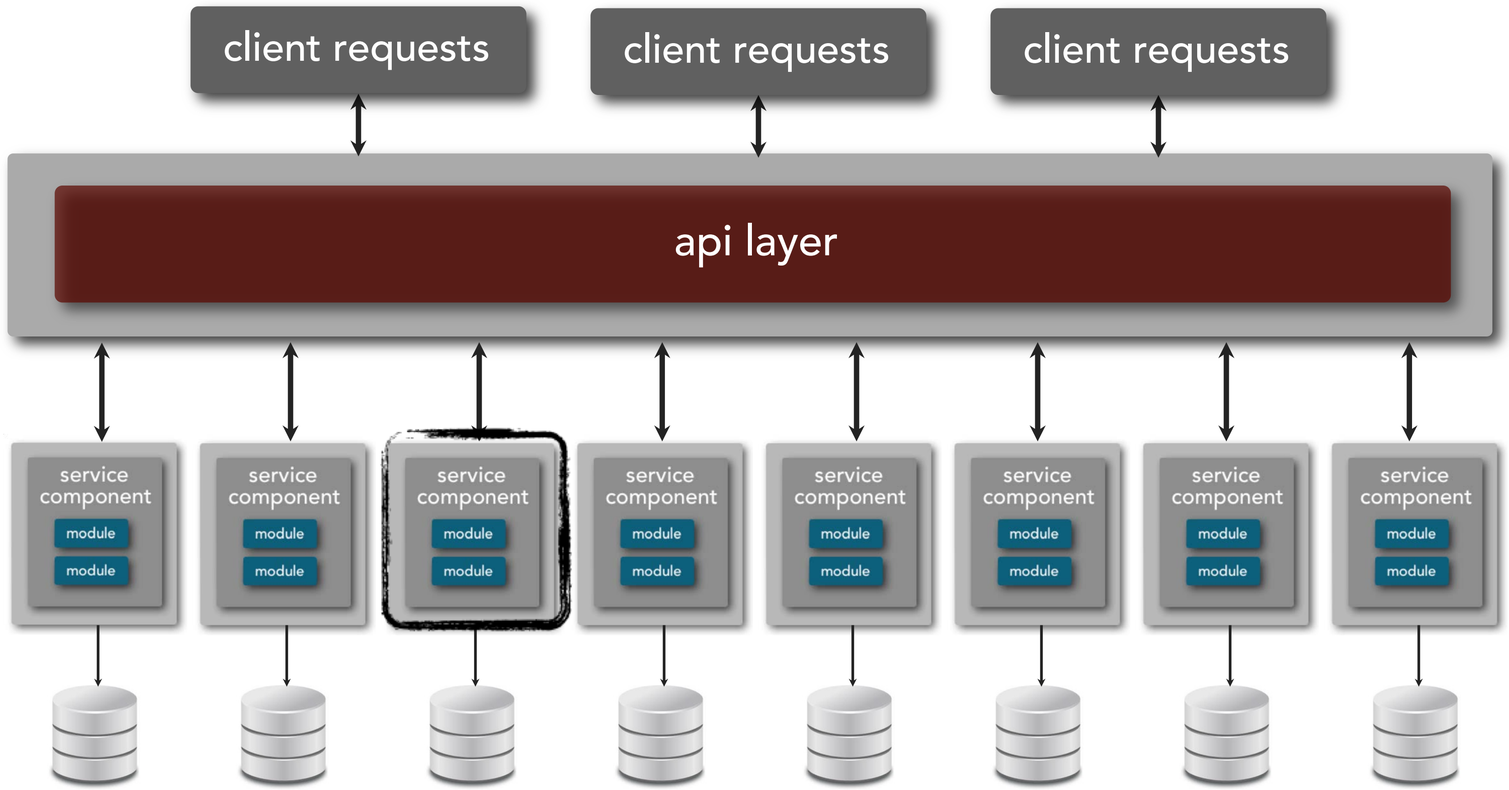
architecture characteristics



architecture characteristics

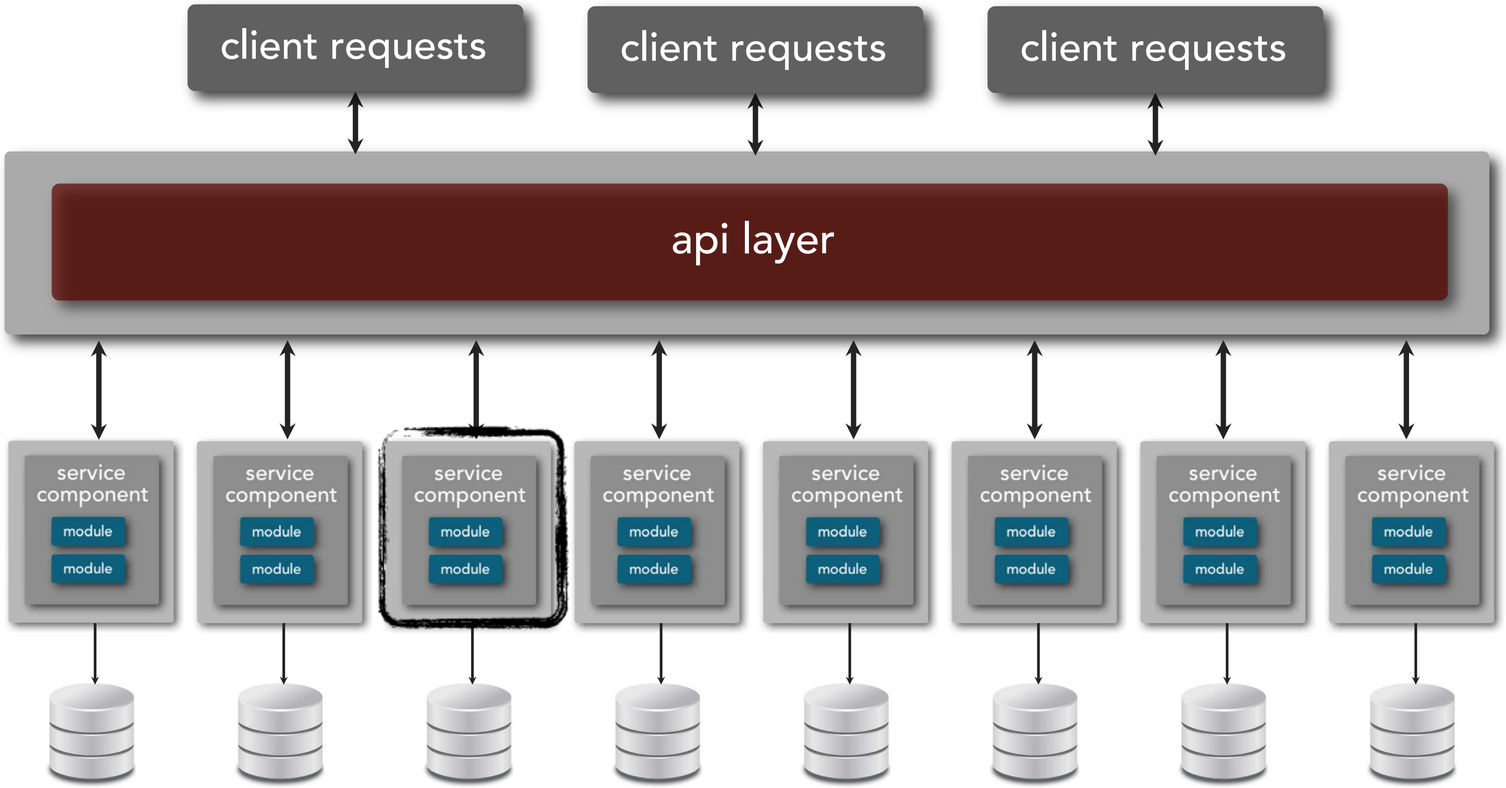


architecture characteristics



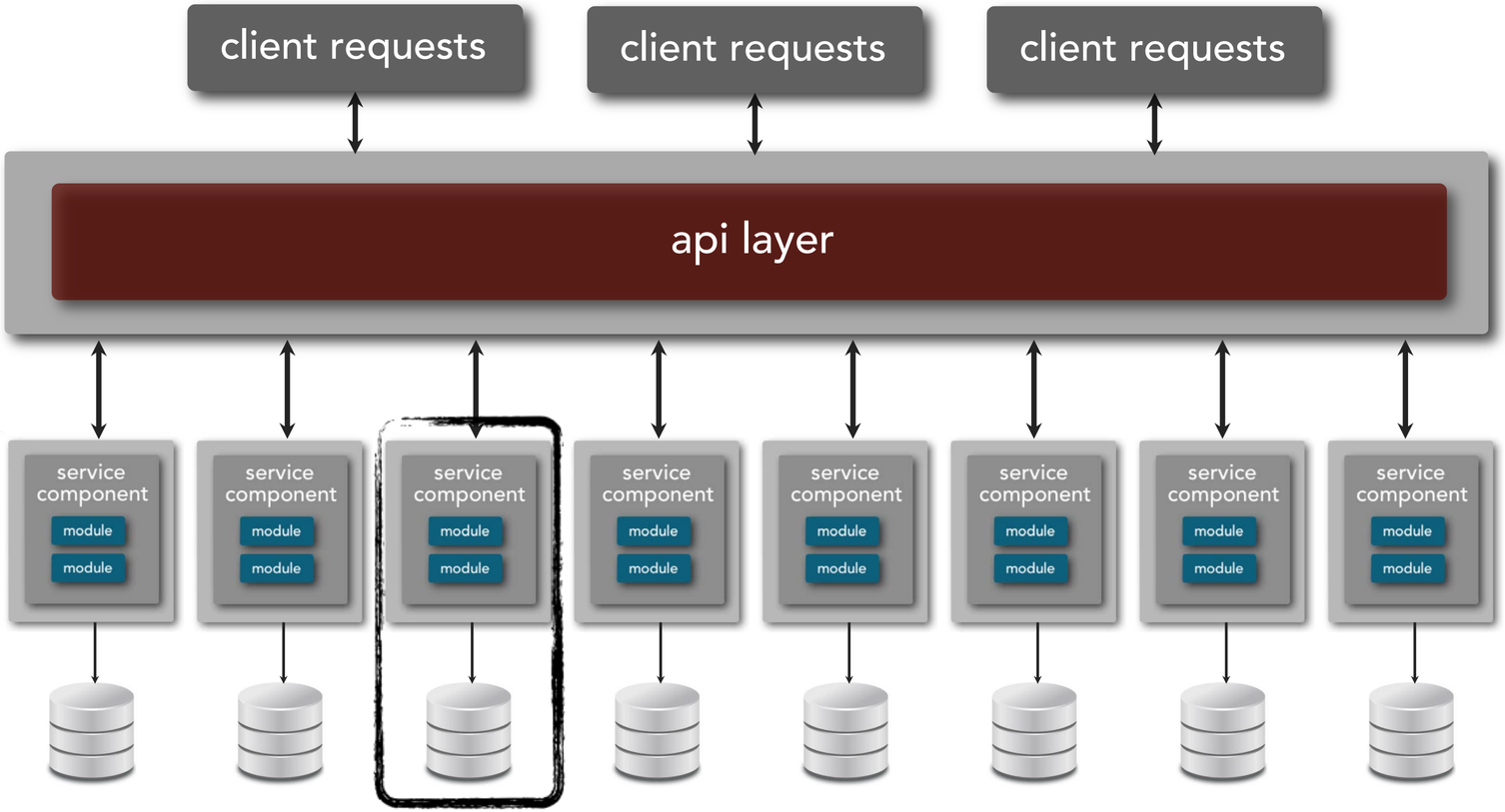
single-purpose, separately deployed unit of software that does *one thing* really well

architecture characteristics



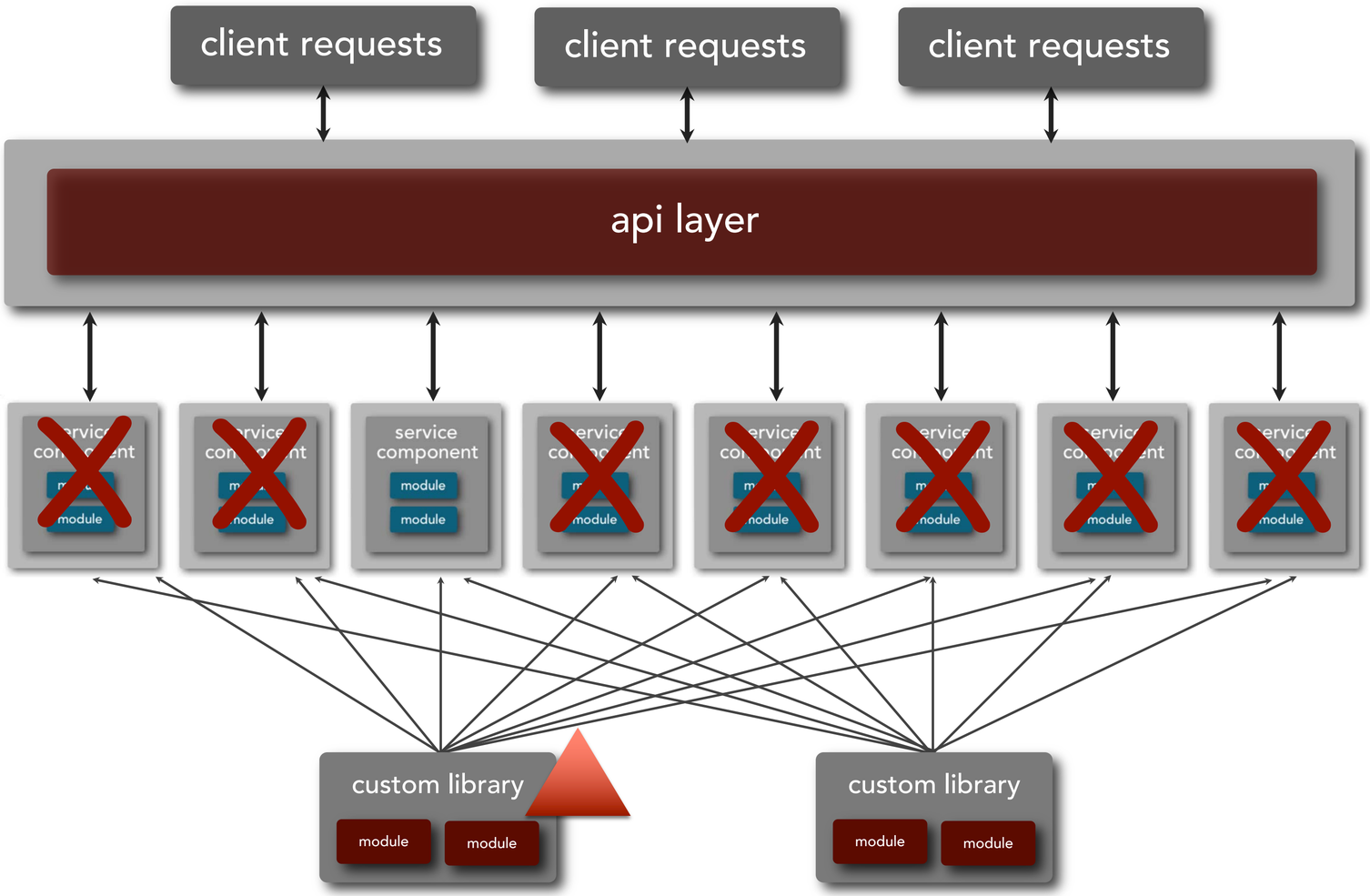
100's to 1000's of services…

architecture characteristics

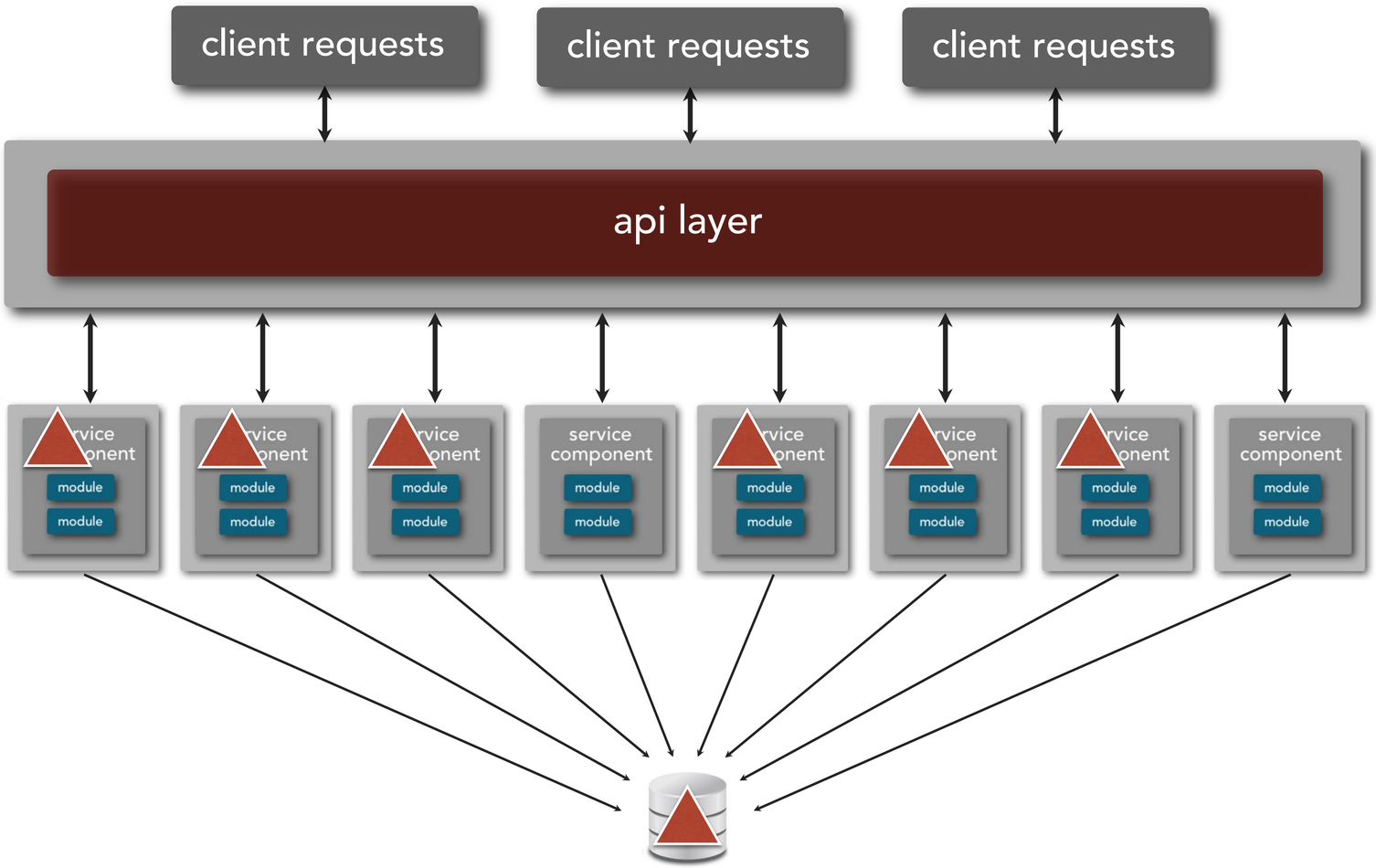


bounded context (share nothing architecture)

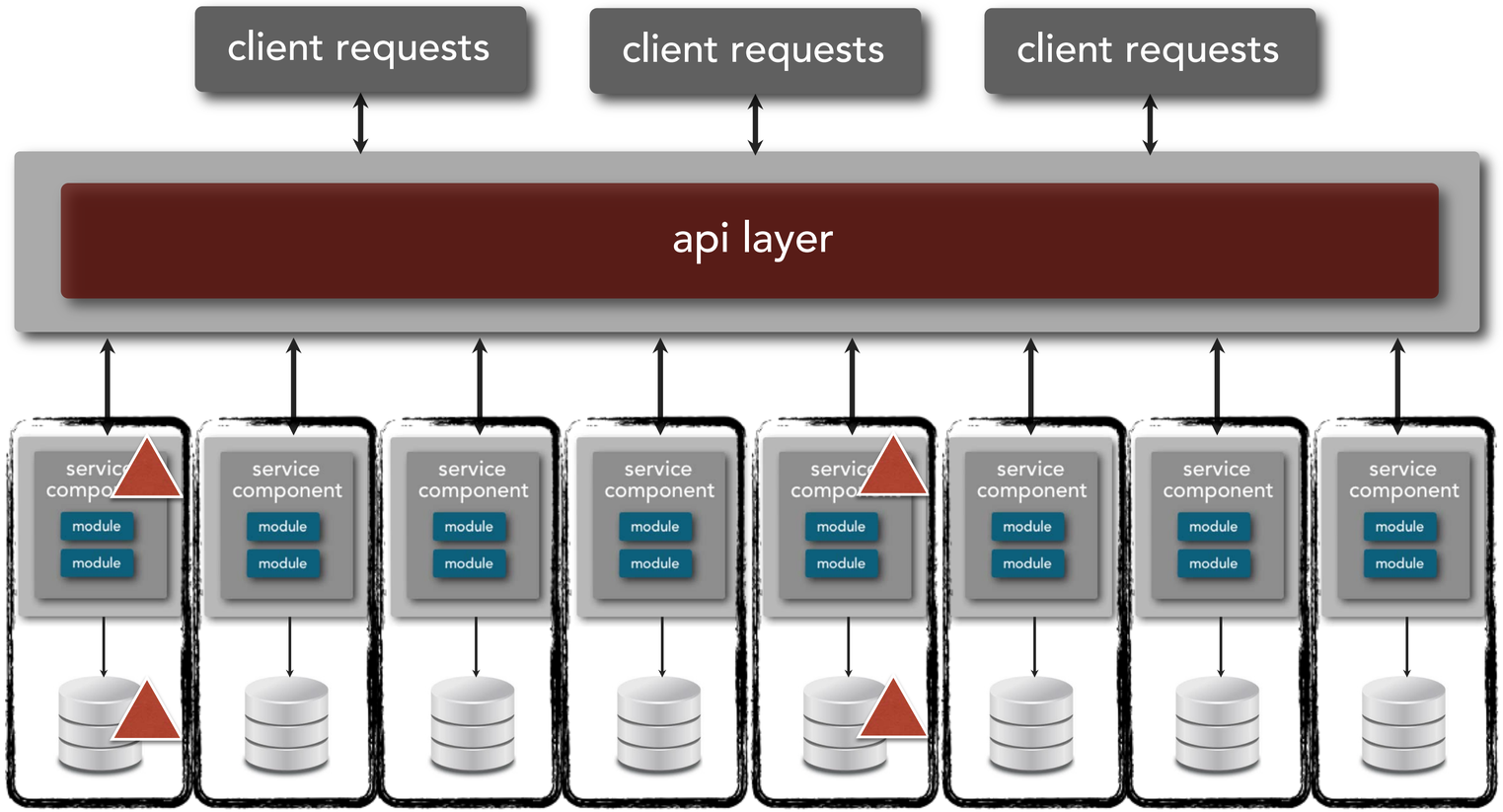
architecture characteristics



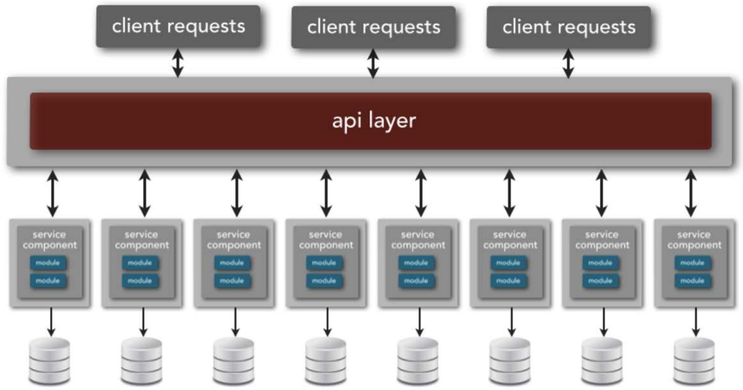
architecture characteristics

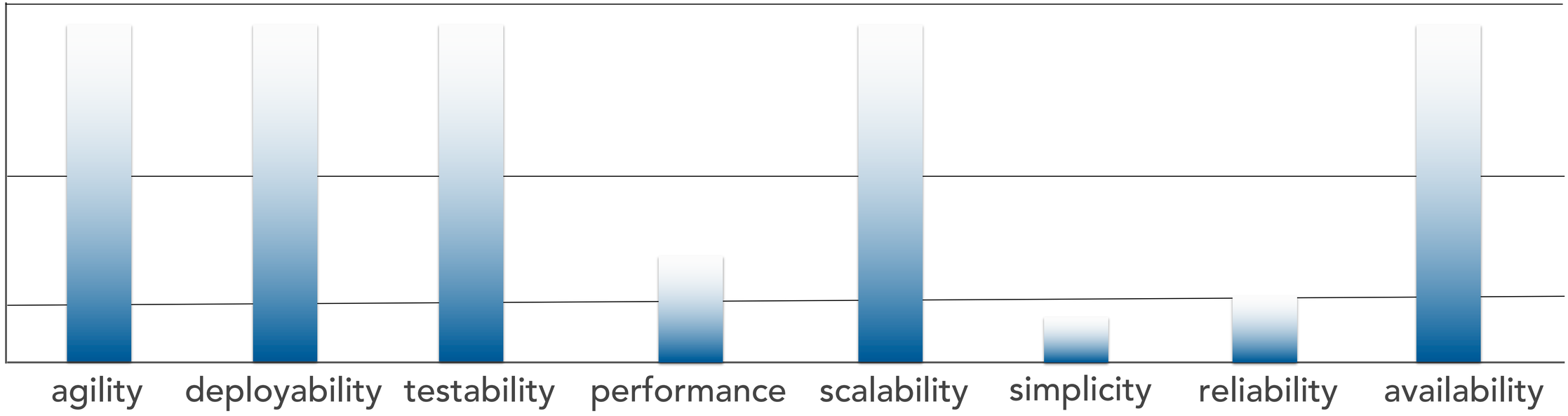


architecture characteristics



architecture characteristics





en.wikipedia.org

Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF

Fallacies of distributed computing

From Wikipedia, the free encyclopedia

The **Fallacies of Distributed Computing** are a set of assumptions that L Peter Deutsch and others at Sun Microsystems originally asserted programmers new to distributed applications invariably make. These assumptions ultimately prove false, resulting either in the failure of the system, a substantial reduction in system scope, or in large, unplanned expenses required to redesign the system to meet its original goals.[*citation needed*]

Contents [hide]

1 The fallacies
2 Effects of the fallacies
3 History
4 See also
5 References
6 External links

The fallacies [edit]

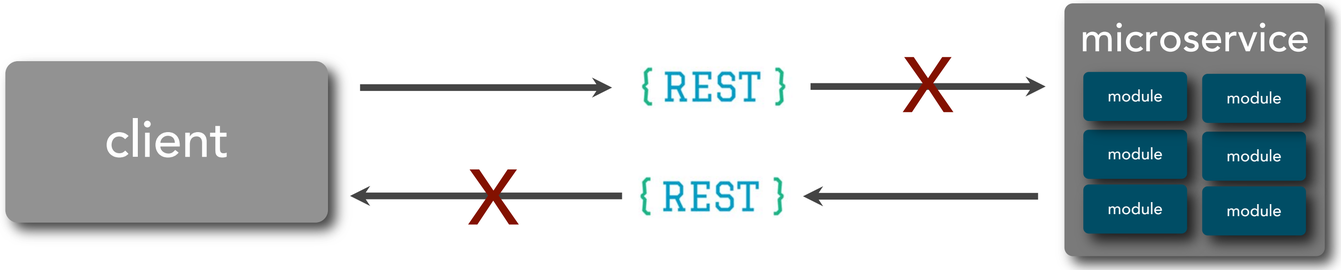
The fallacies are summarized below:[1]

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator

en.wikipedia.org/wiki/Fallacies_of_distributed_computing

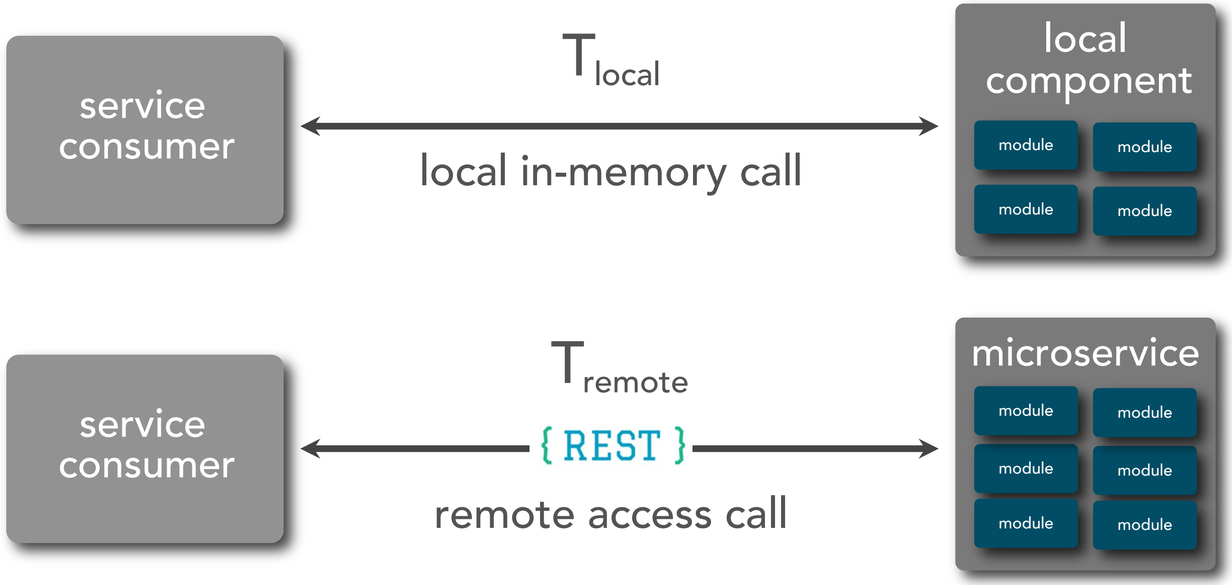
distributed architecture

1. the network is reliable



distributed architecture

2. latency is zero



$T_{remote} > T_{local}$

distributed architecture

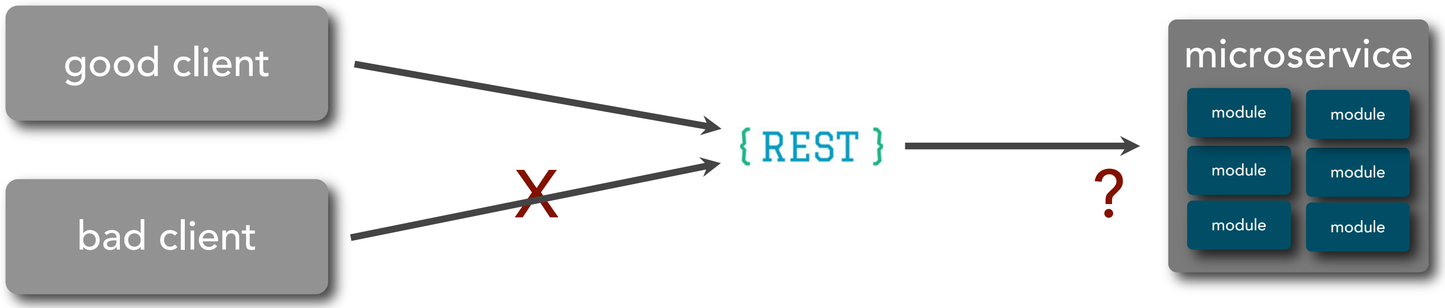
3. bandwidth is infinite

distributed architecture

3. bandwidth is infinite

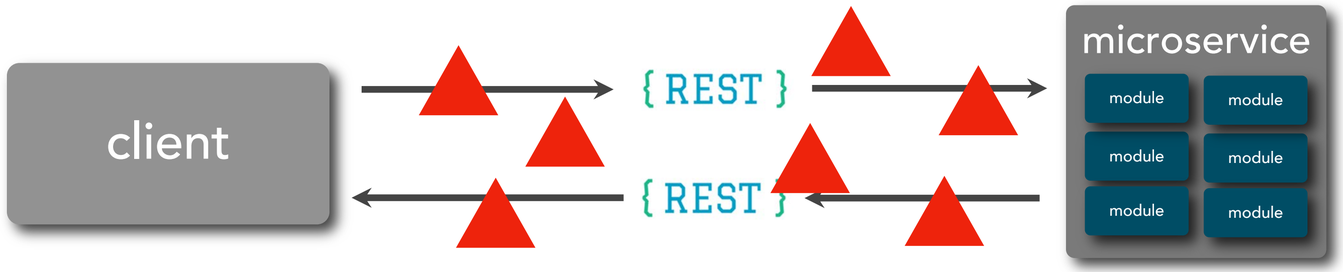
distributed architecture

4. the network is secure



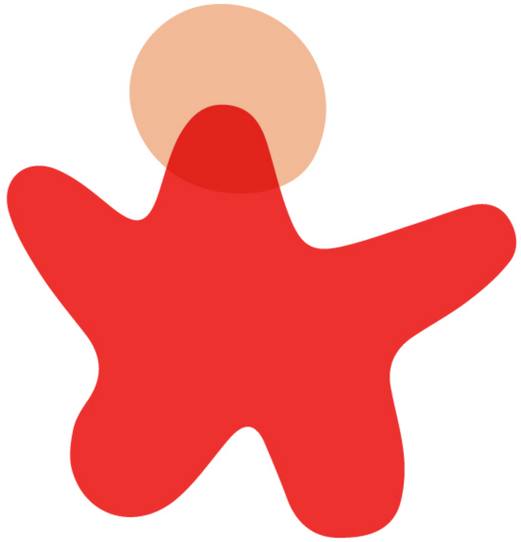
distributed architecture

5. topology doesn't change



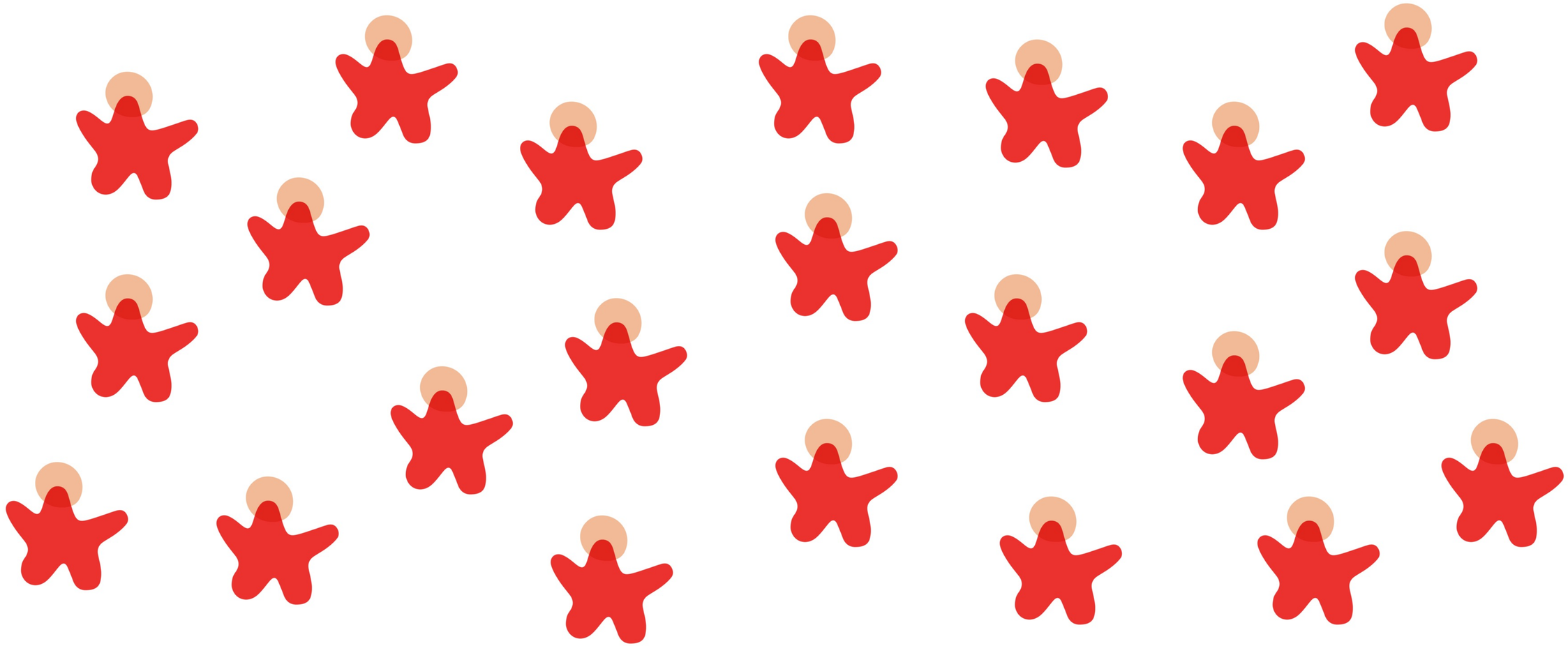
distributed architecture

6. there is one administrator



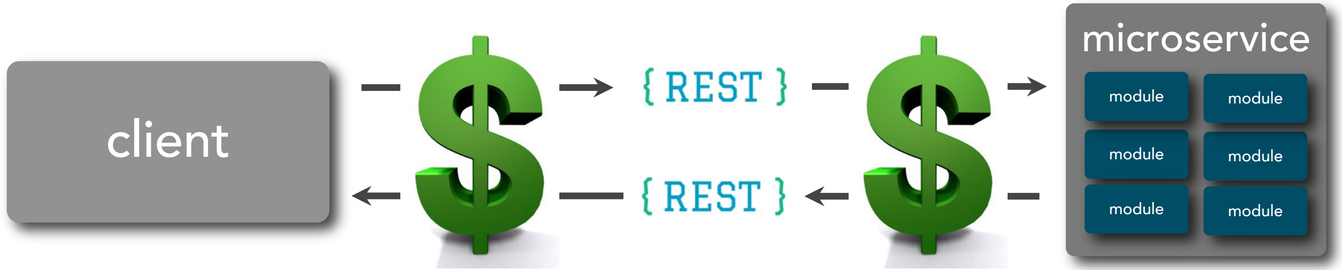
distributed architecture

6. there is one administrator



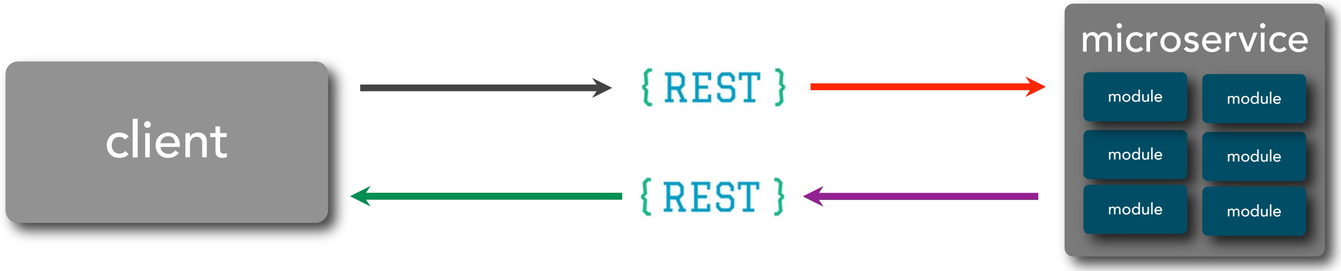
distributed architecture

7. transport cost is zero



distributed architecture

8. the network is homogeneous



en.wikipedia.org

Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF

Fallacies of distributed computing

From Wikipedia, the free encyclopedia

The **Fallacies of Distributed Computing** are a set of assumptions that L Peter Deutsch and others at Sun Microsystems originally asserted programmers new to distributed applications invariably make. These assumptions ultimately prove false, resulting either in the failure of the system, a substantial reduction in system scope, or in large, unplanned expenses required to redesign the system to meet its original goals.[*citation needed*]

Contents [hide]

1 The fallacies
2 Effects of the fallacies
3 History
4 See also
5 References
6 External links

The fallacies [edit]

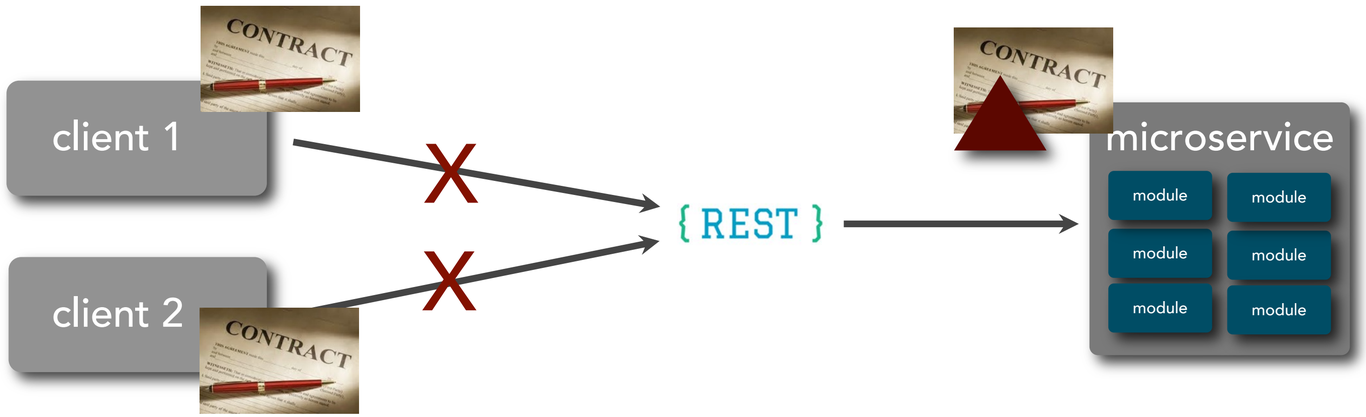
The fallacies are summarized below:[1]

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator

en.wikipedia.org/wiki/Fallacies_of_distributed_computing

distributed architecture

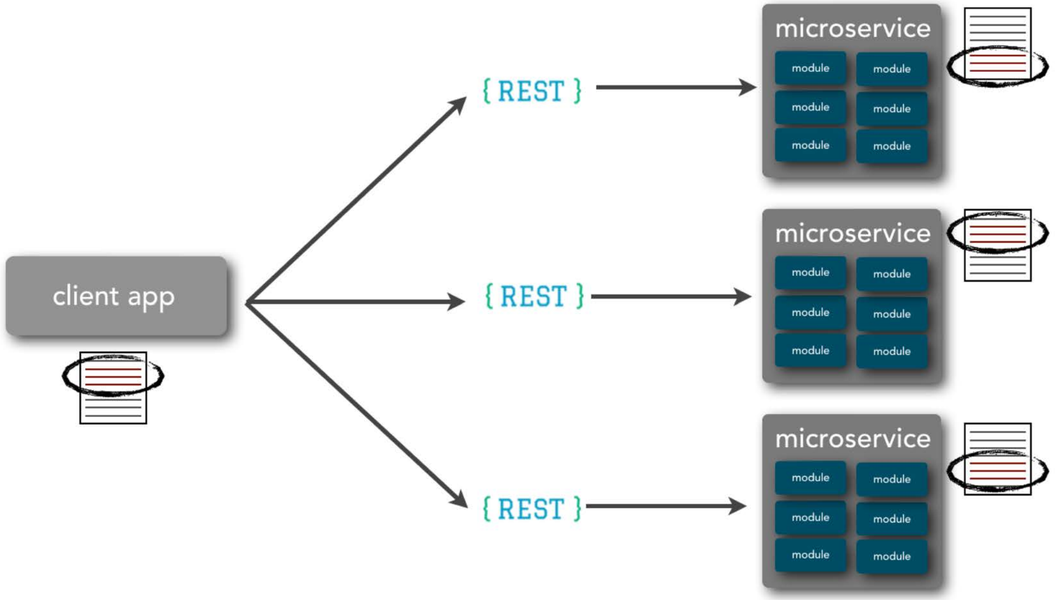
contract creation, maintenance, versioning, and coordination



https://www.developertoarchitect.com/lessons/lesson41.html

distributed architecture

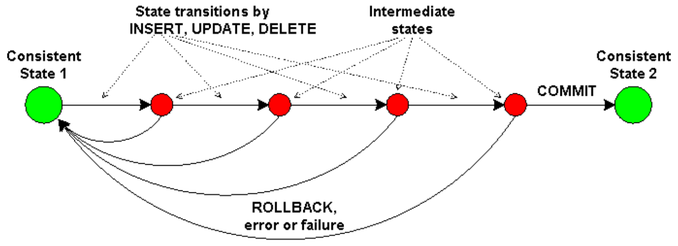
distributed logging facilities to provide a holistic view of a transaction

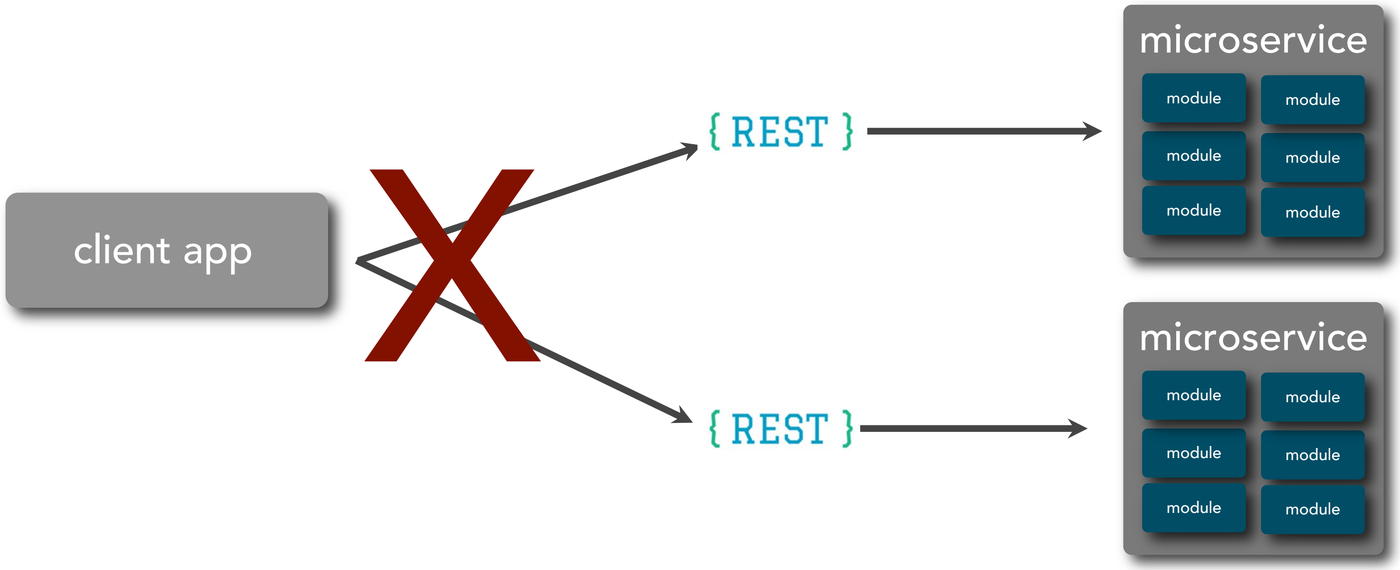


https://www.developertoarchitect.com/lessons/lesson4.html

distributed architecture

distributed transaction management

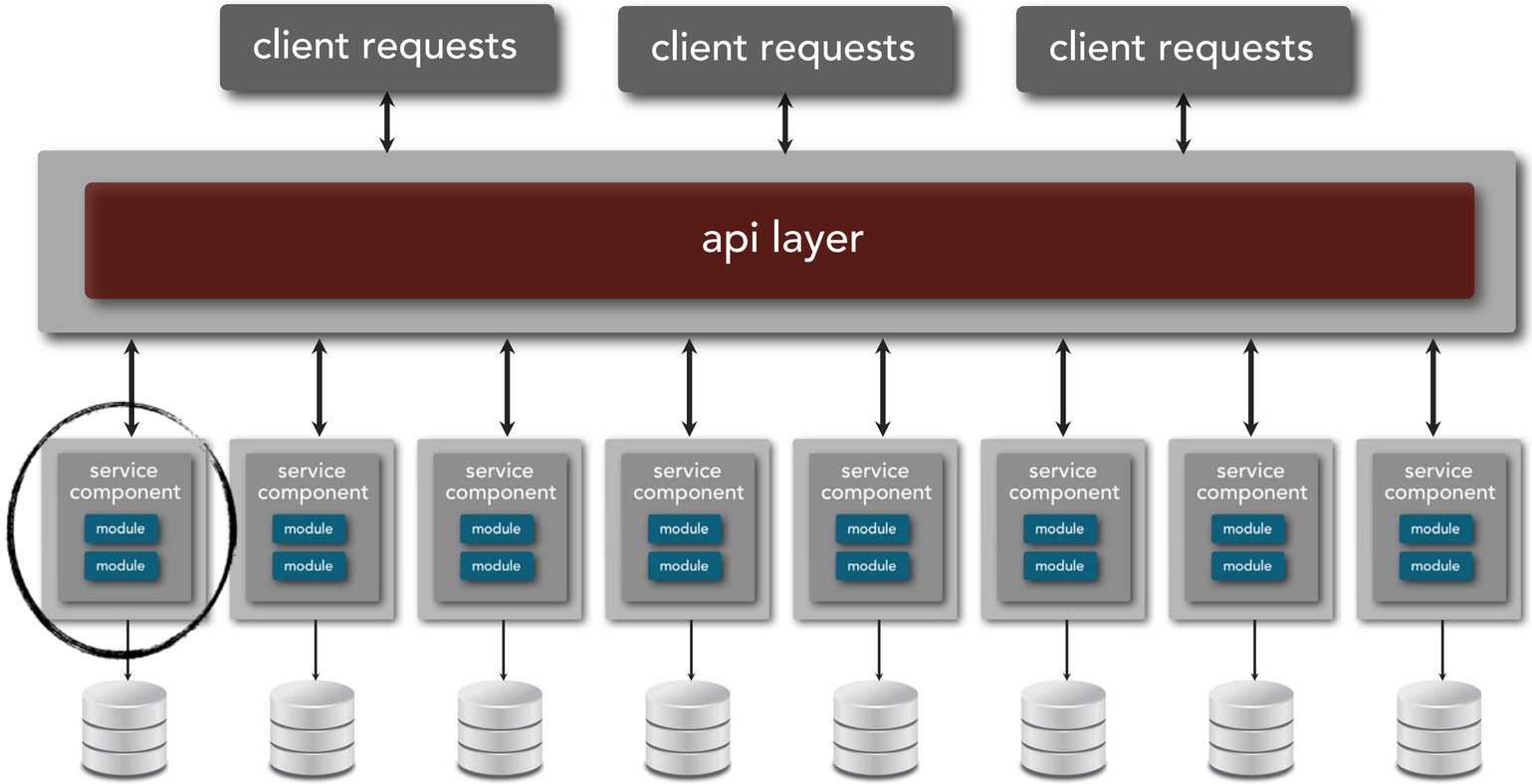




Service Design

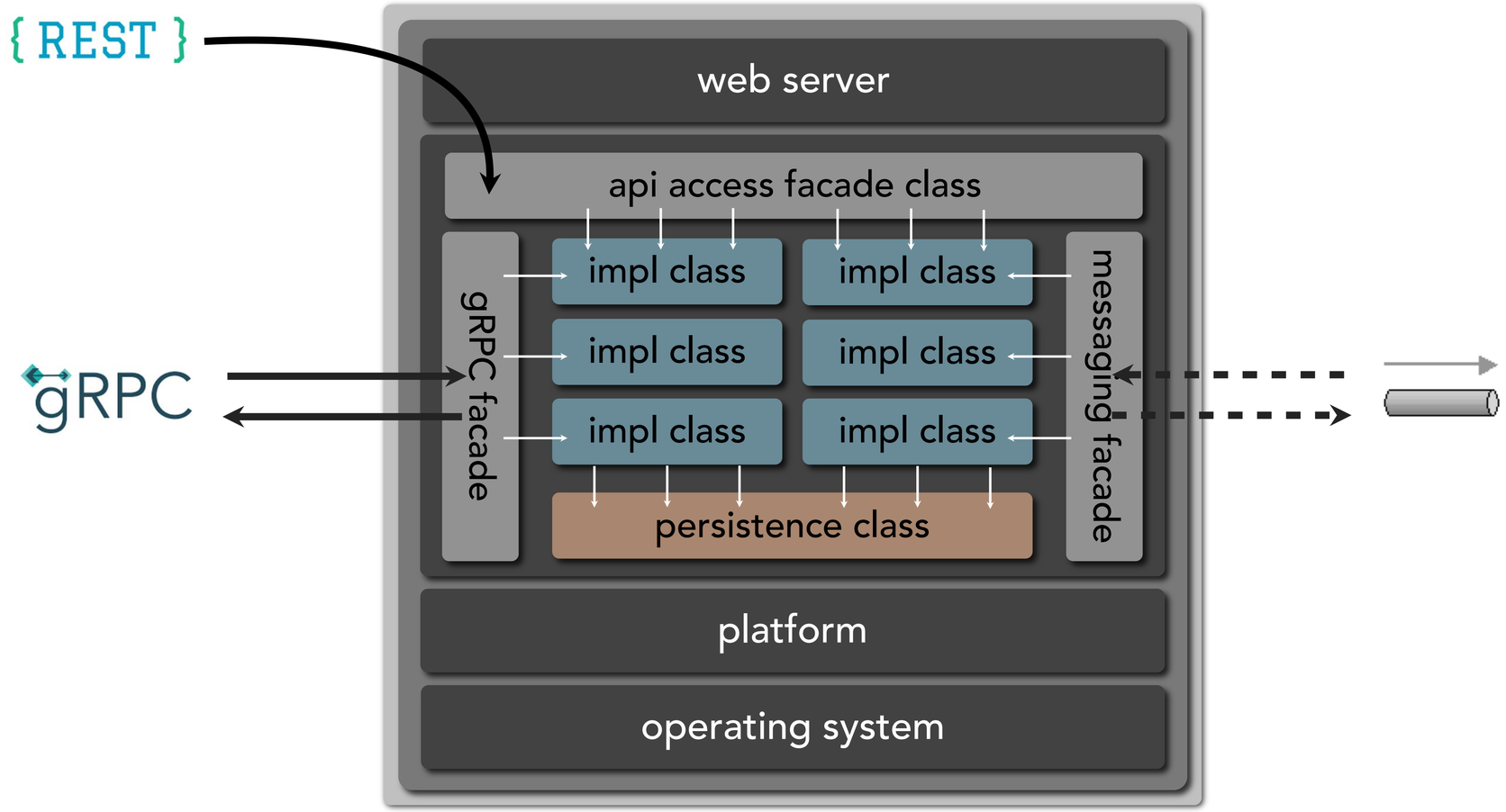
microservices architecture

service definition



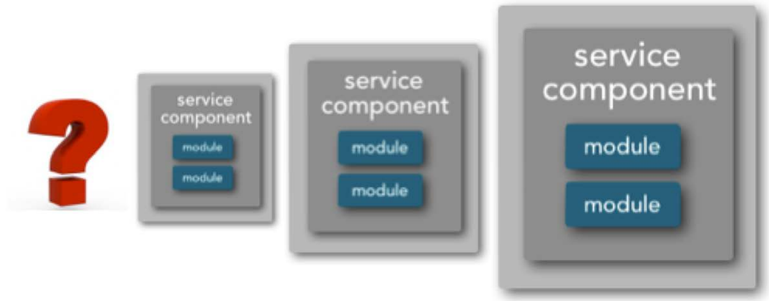
microservices architecture

micro-hexagonal design **to support** *protocol agnostic processing*

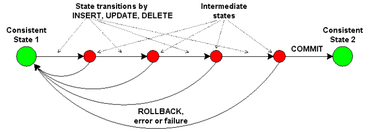


Service Granularity

service granularity

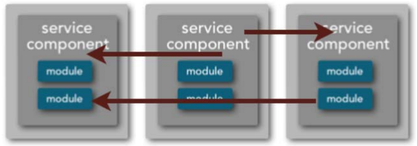


service functionality



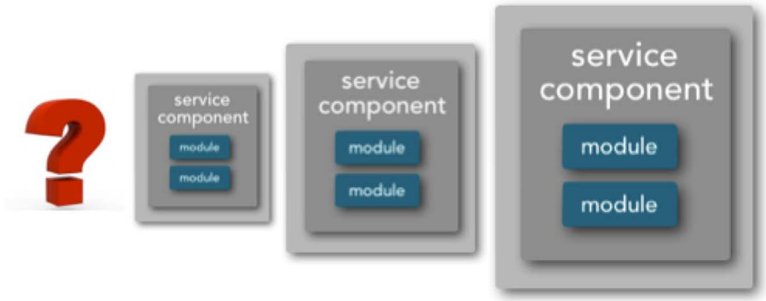
database transactions

data dependencies

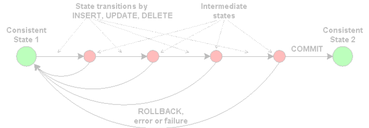


workflow and choreography

service granularity



service functionality



database transactions

data dependencies

workflow and choreography

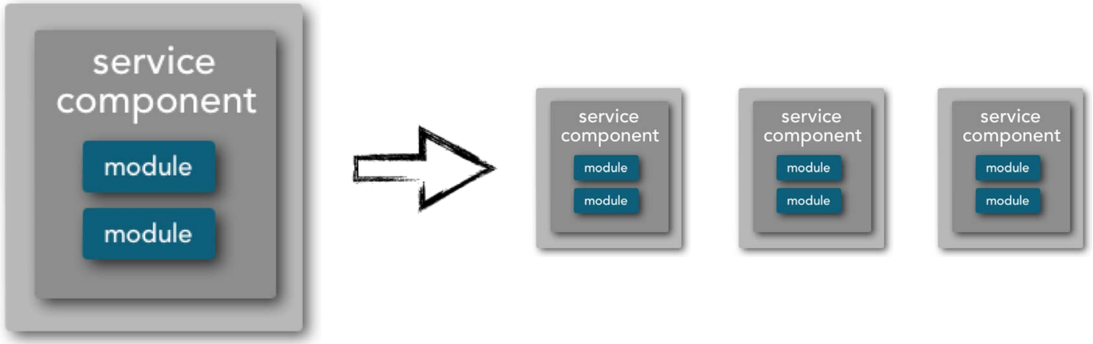
service granularity

service
functionality

service scope and function
(single-purpose function)

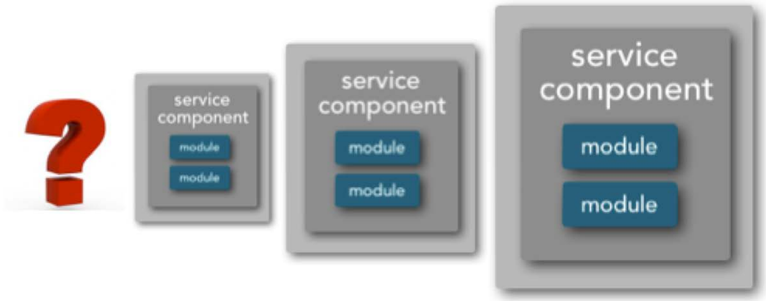
service granularity

service
functionality

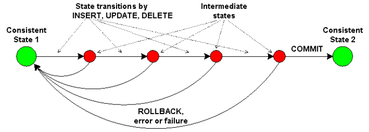


service scope and function
(single-purpose function)

service granularity



service functionality

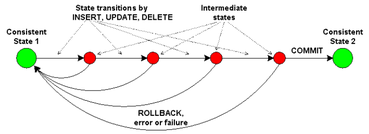


database transactions

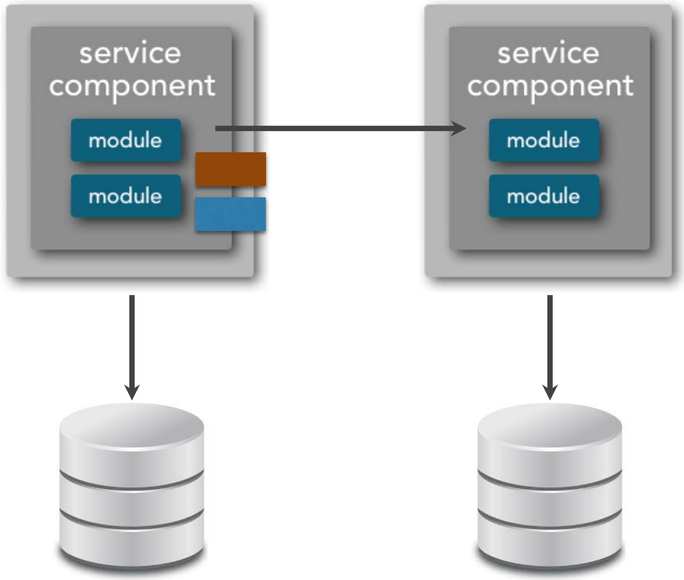
data dependencies

workflow and choreography

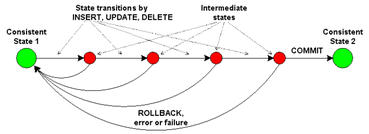
service granularity



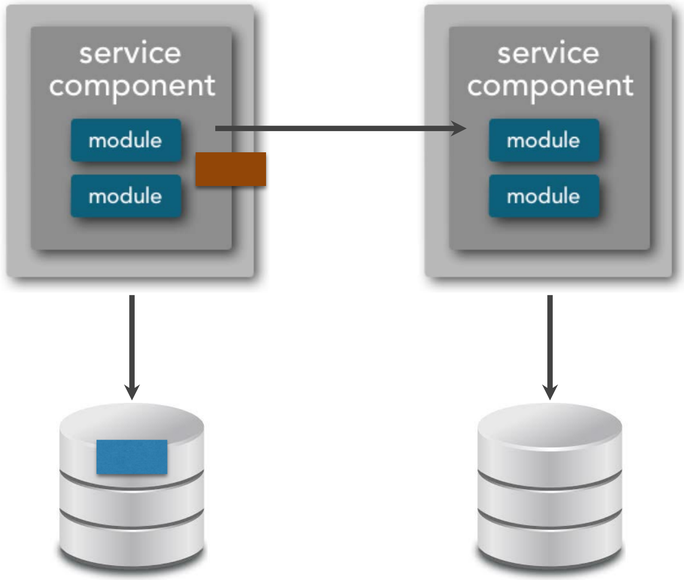
database
transactions



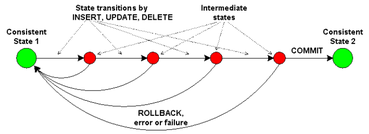
service granularity



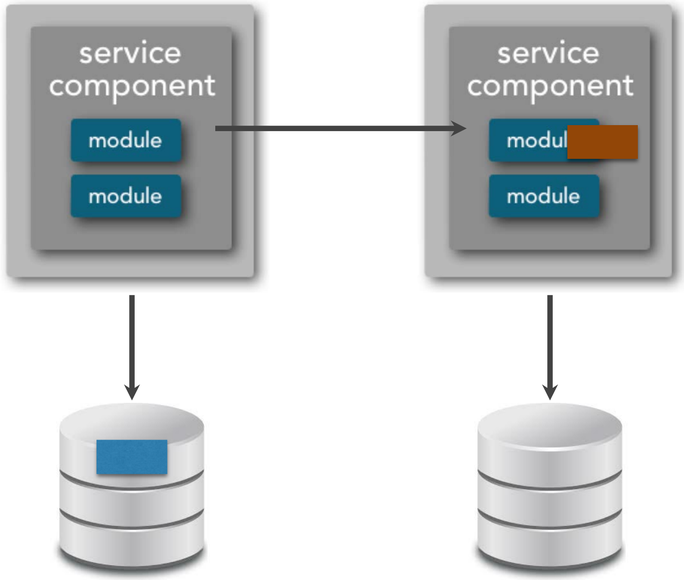
database
transactions



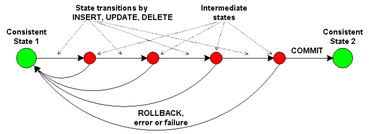
service granularity



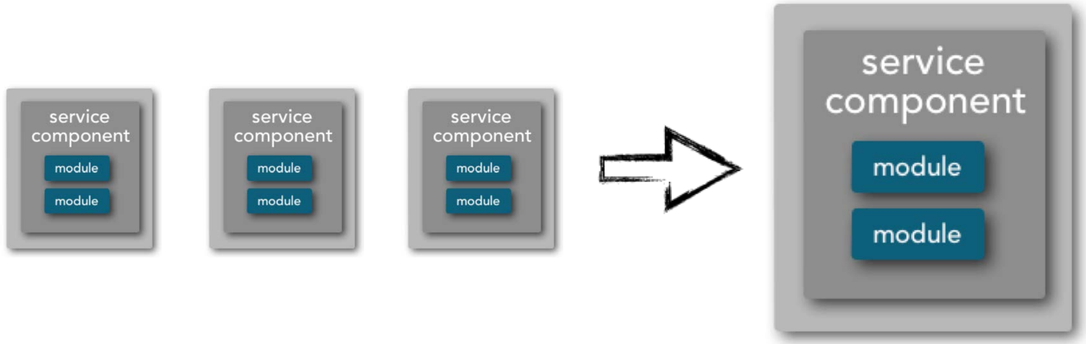
database
transactions

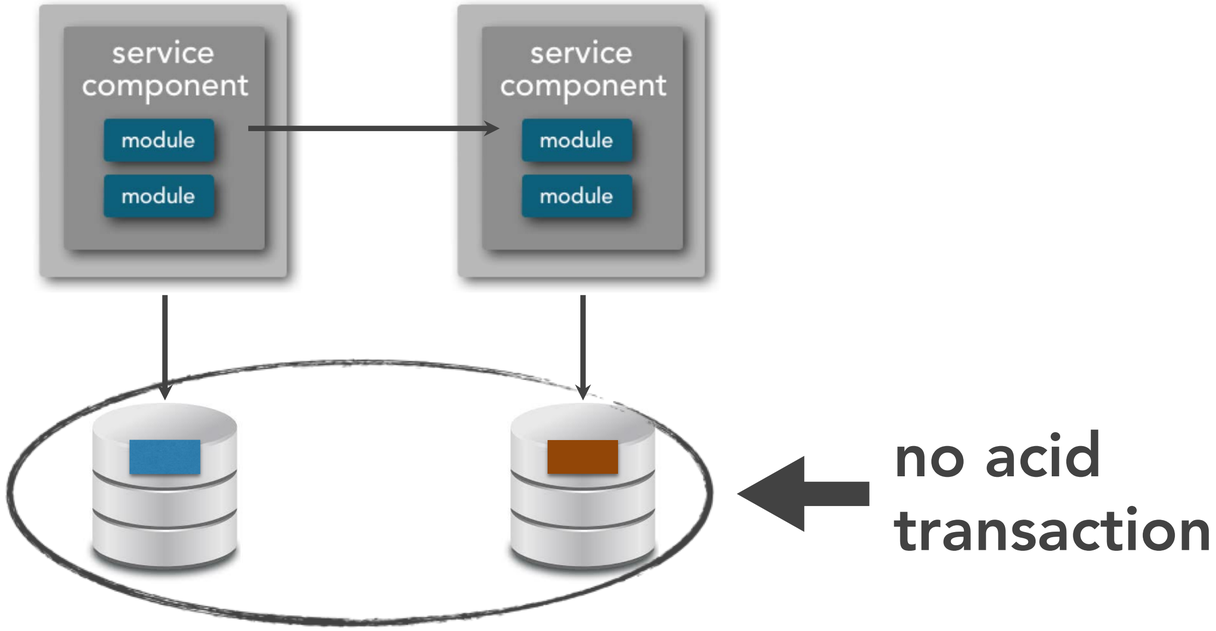


service granularity

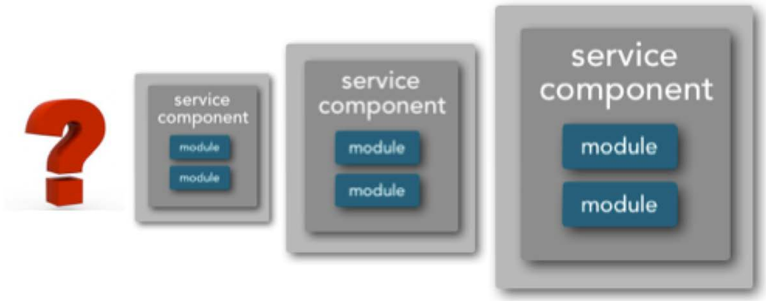


database
transactions

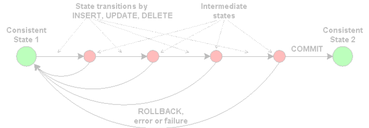




service granularity



service functionality

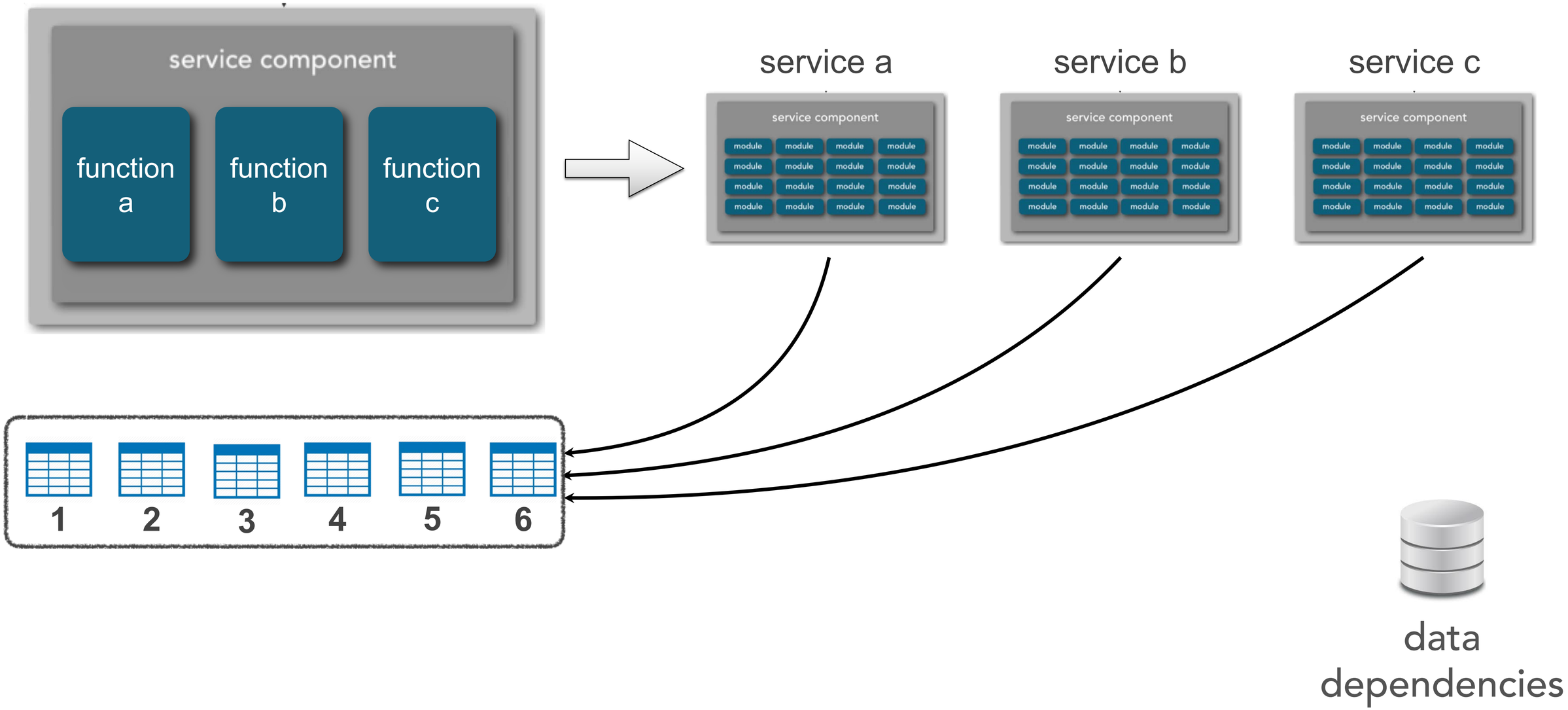


database transactions

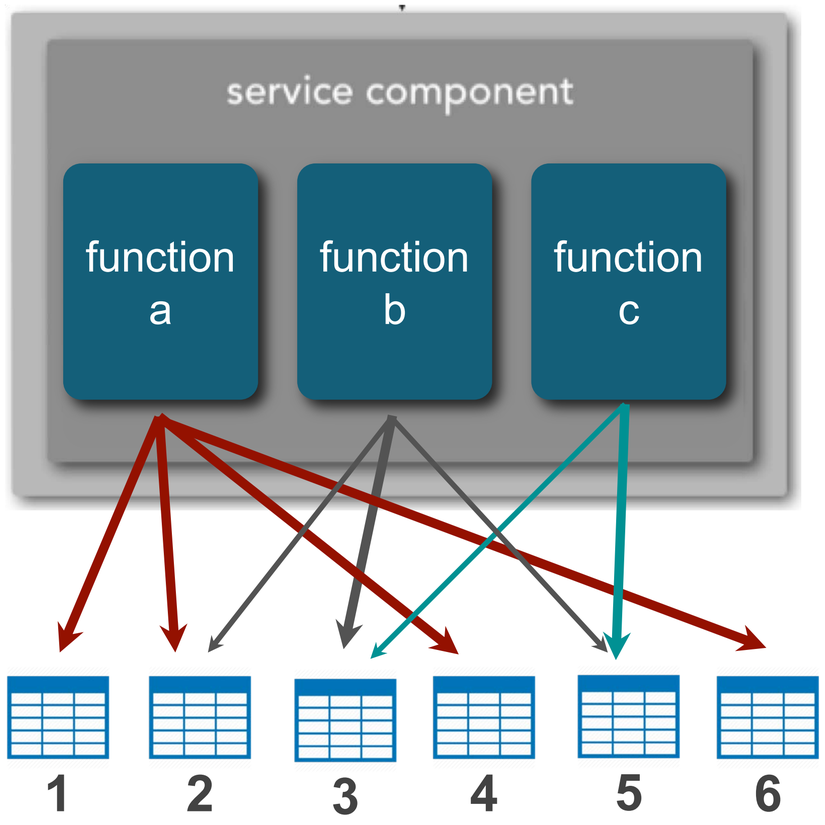
data dependencies

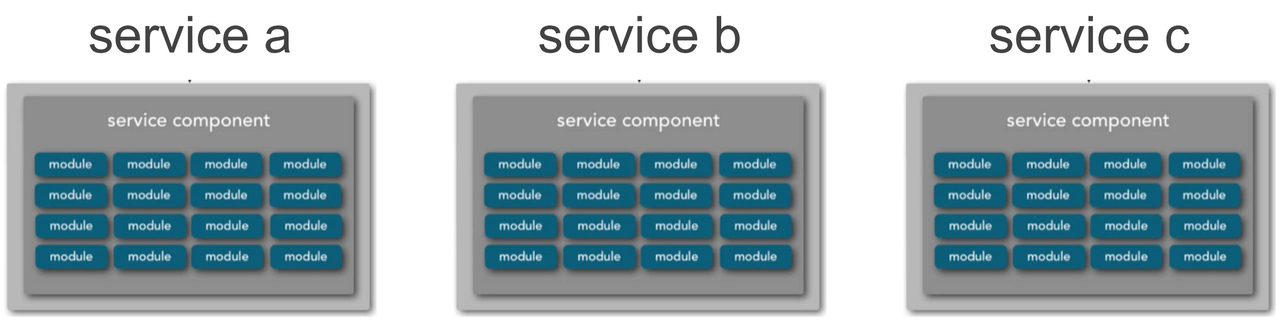
workflow and choreography

service granularity



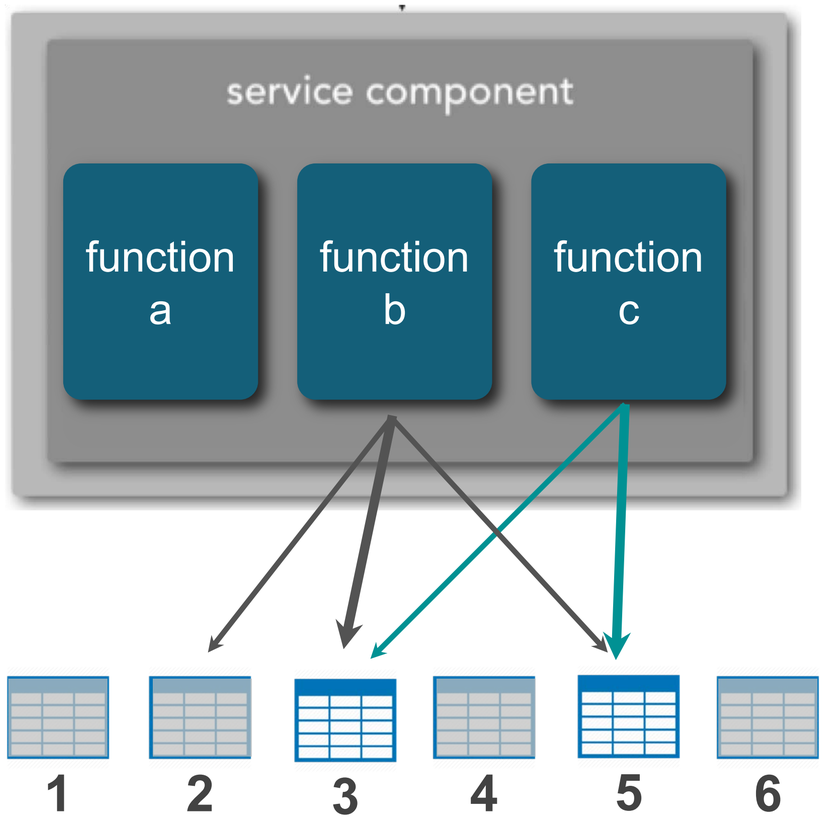
service granularity

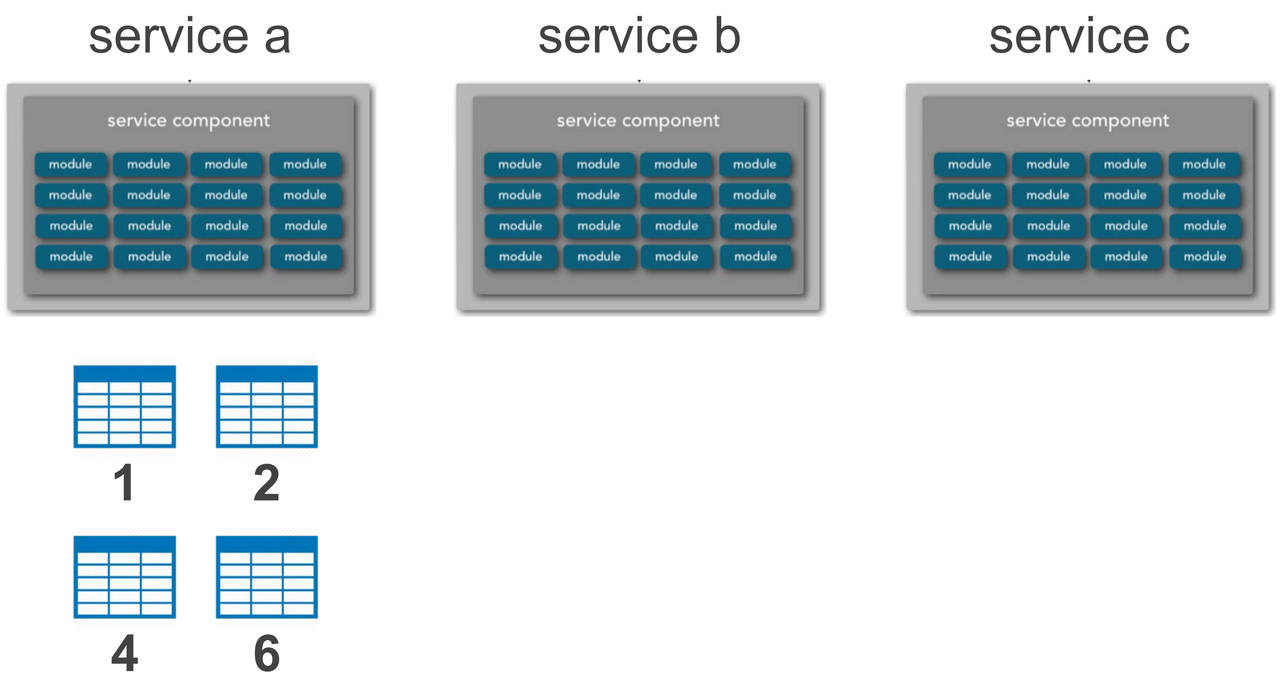




data dependencies

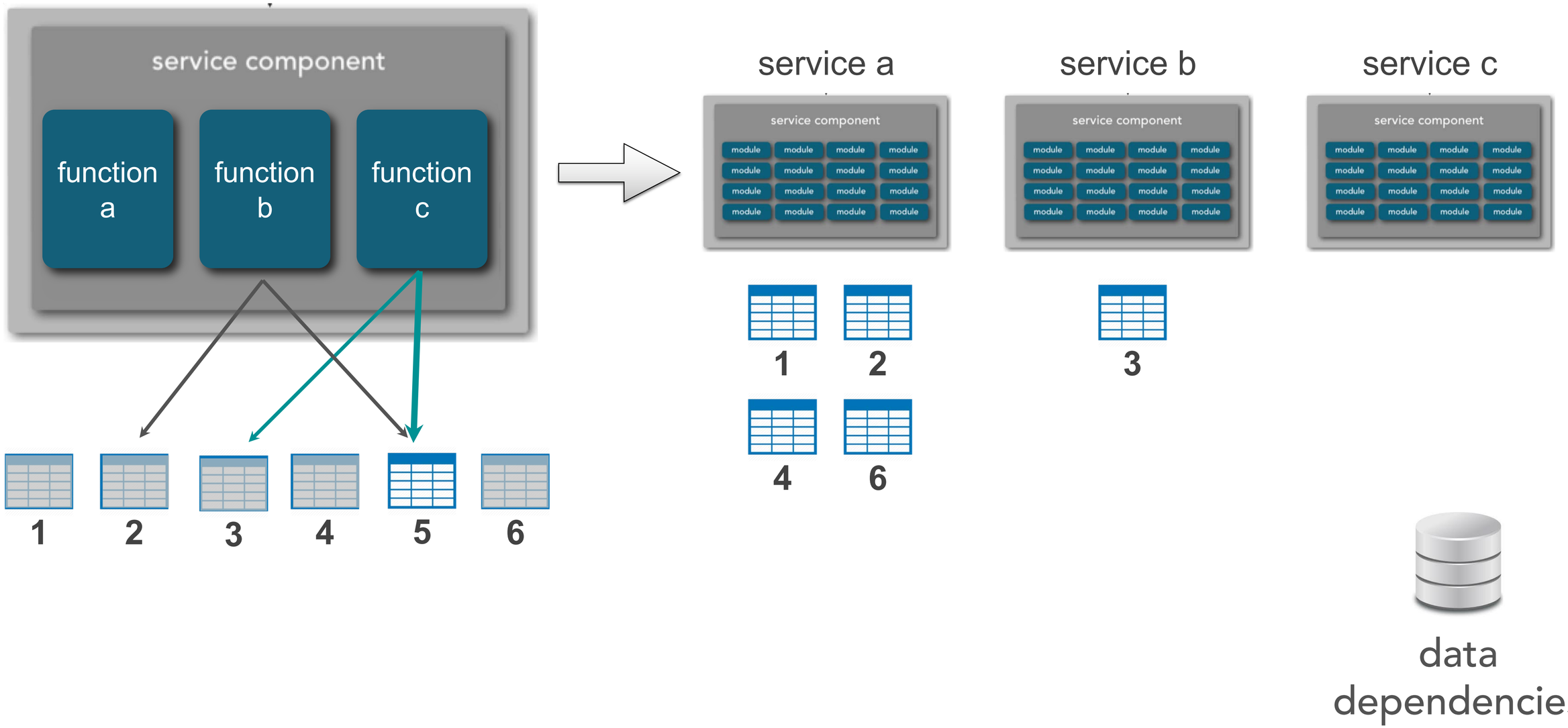
service granularity



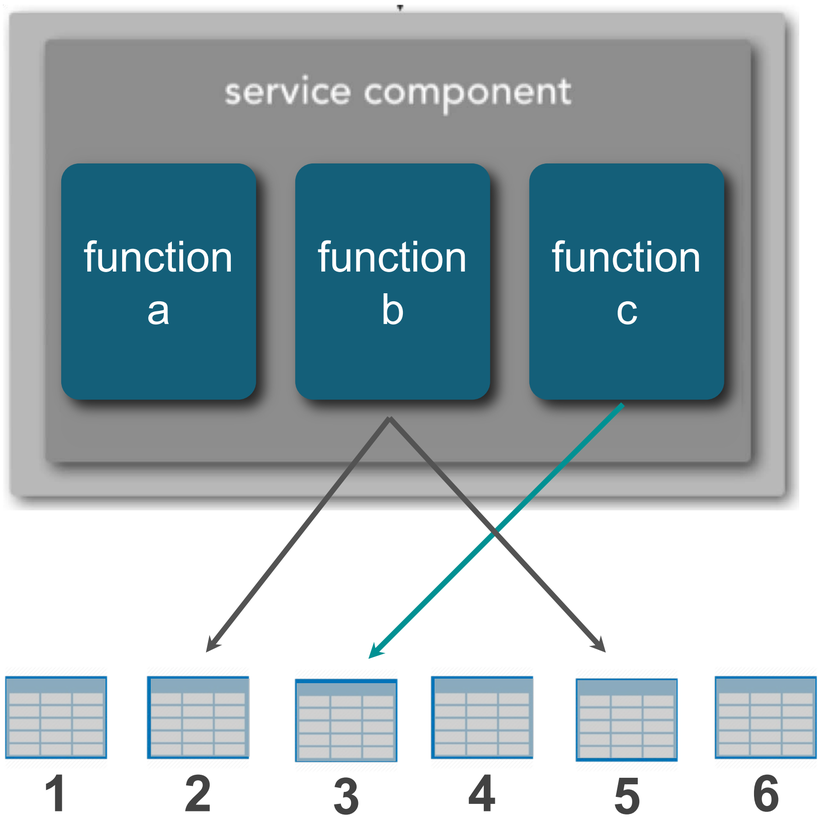


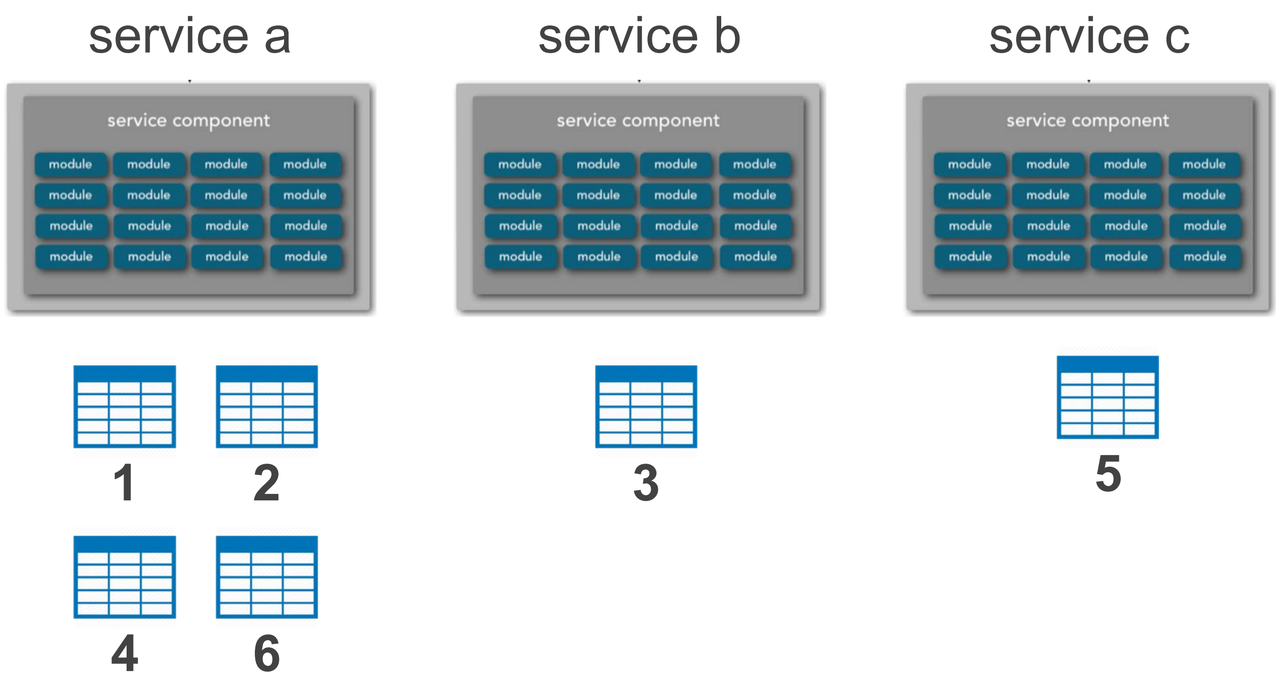
data dependencies

service granularity



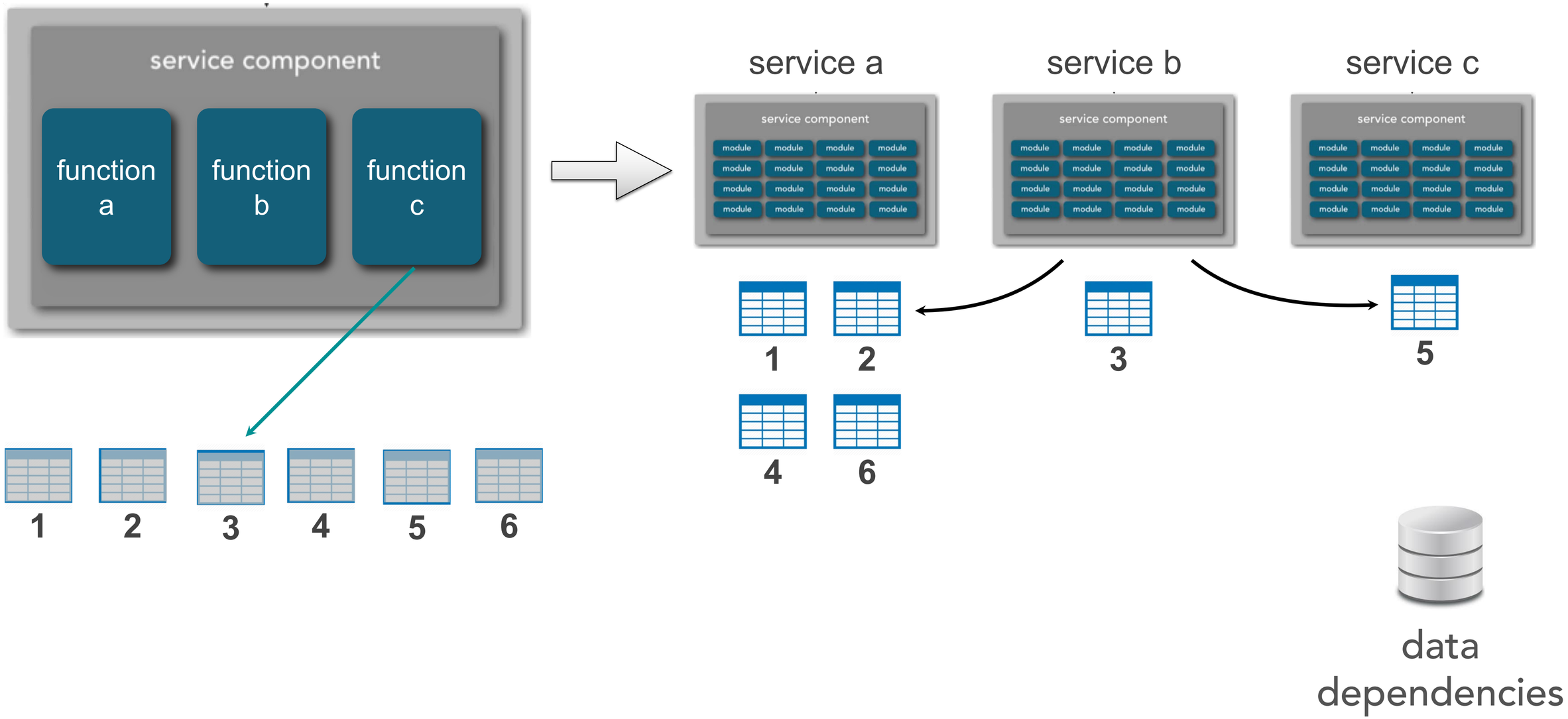
service granularity



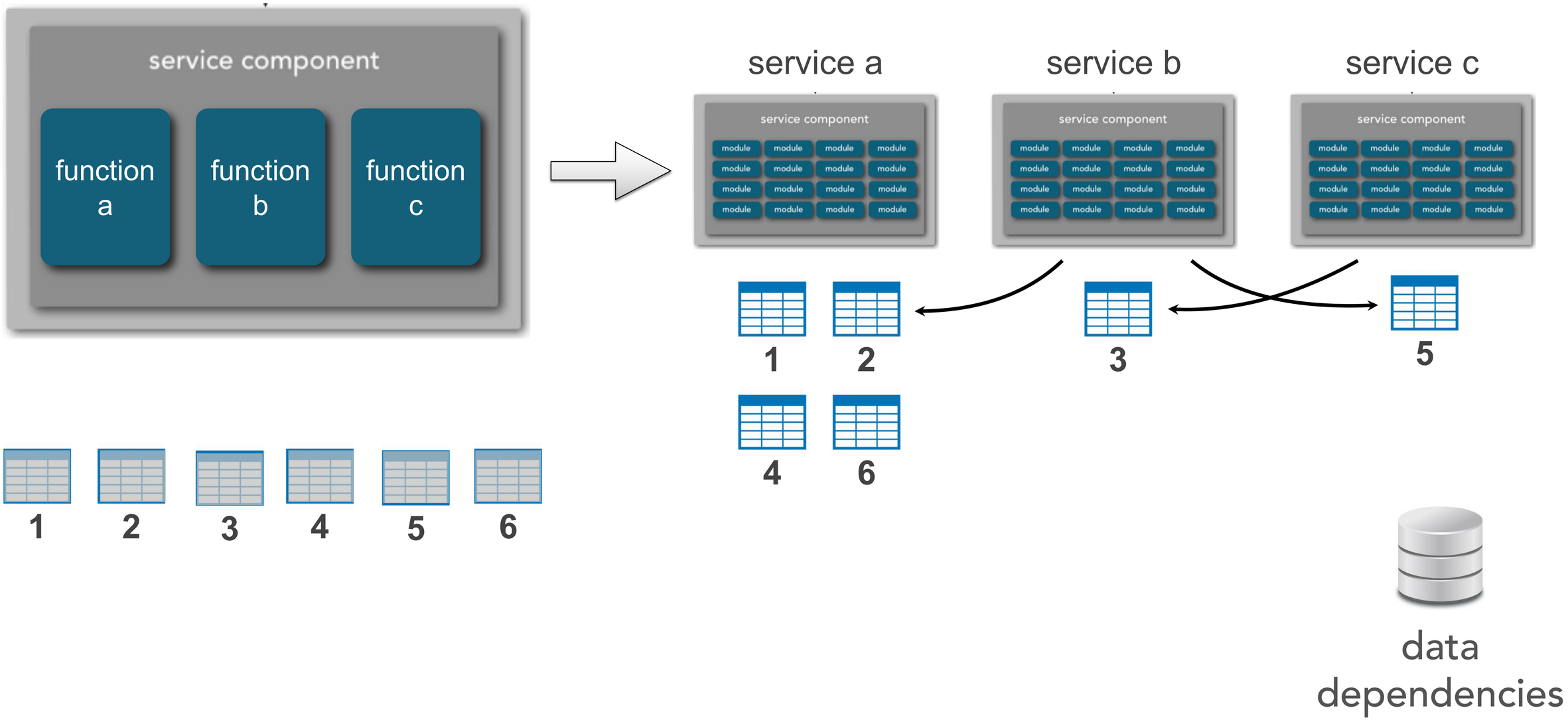


data dependencies

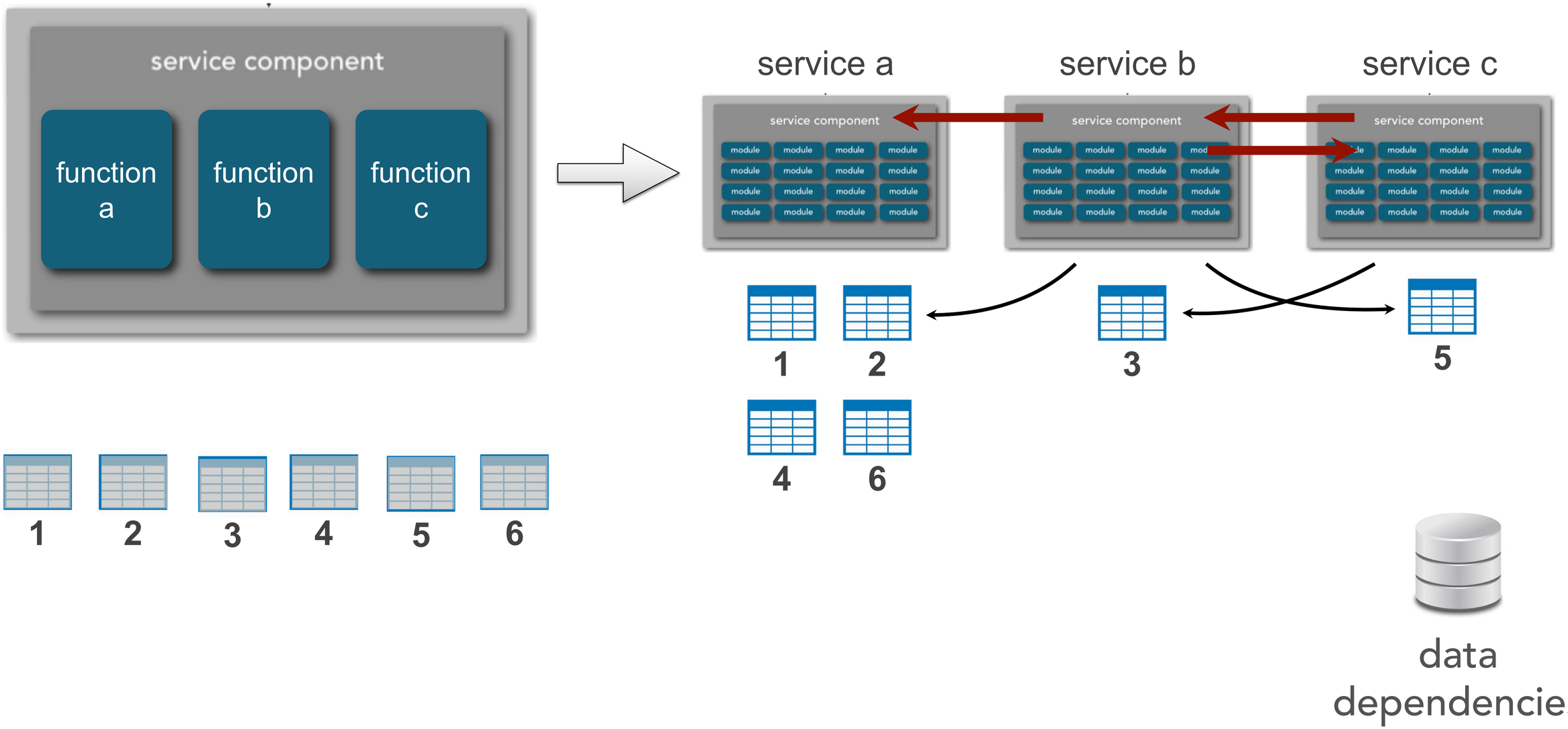
service granularity



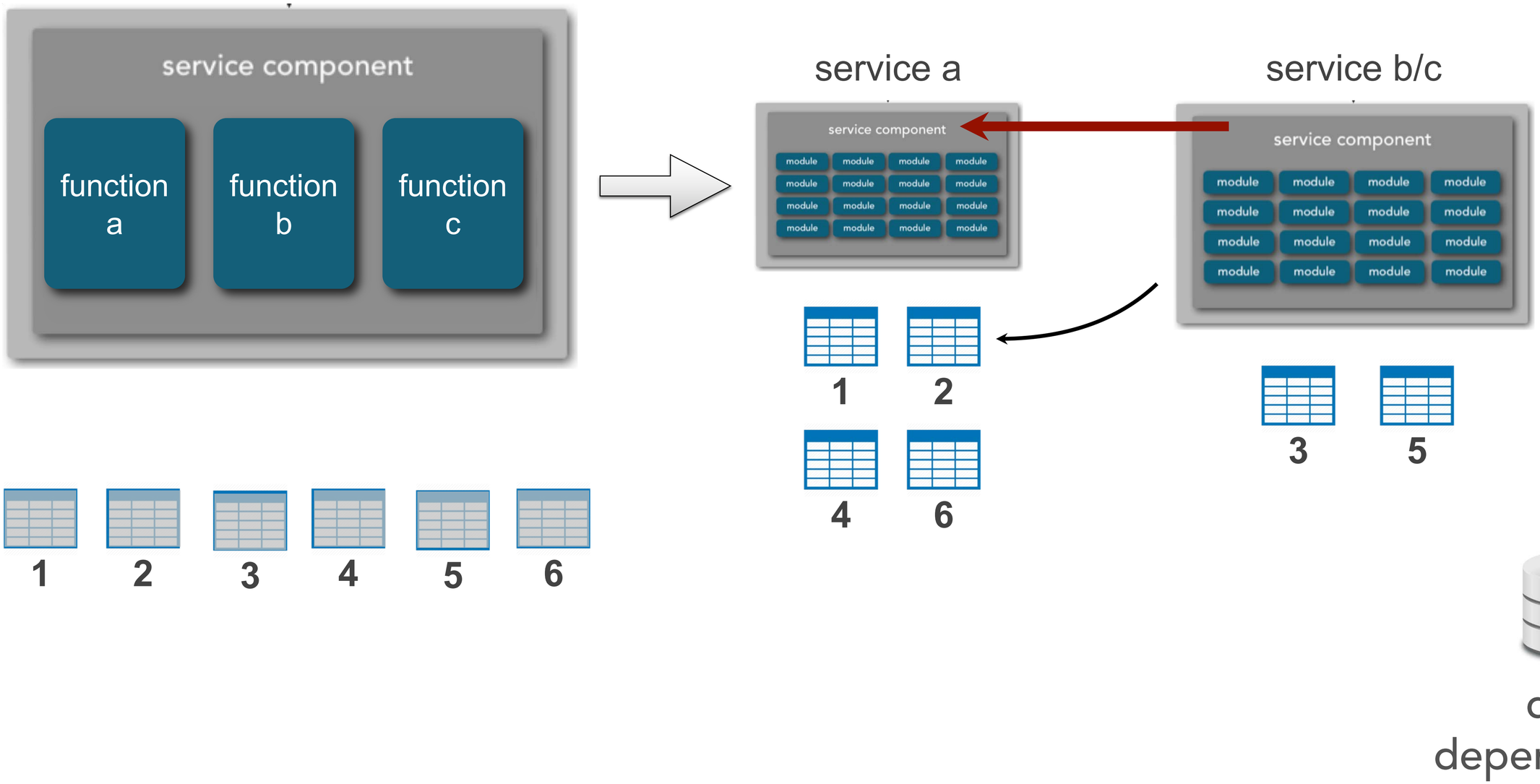
service granularity



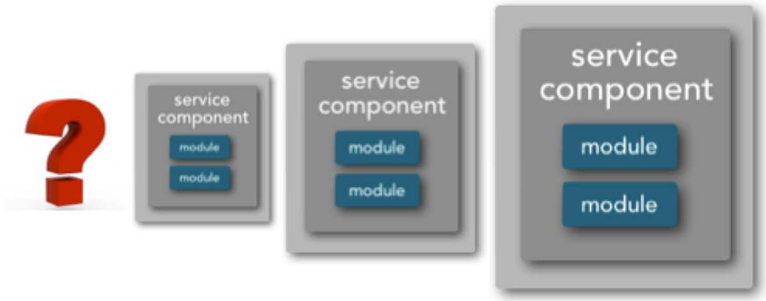
service granularity



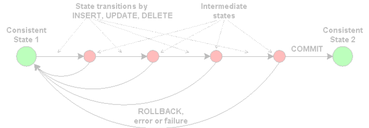
service granularity



service granularity

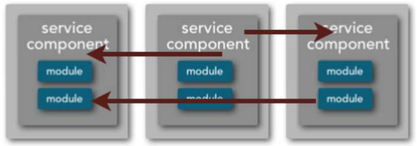


service functionality



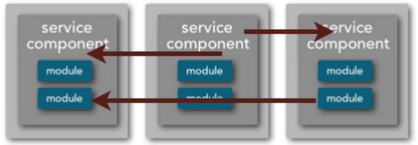
database transactions

data dependencies

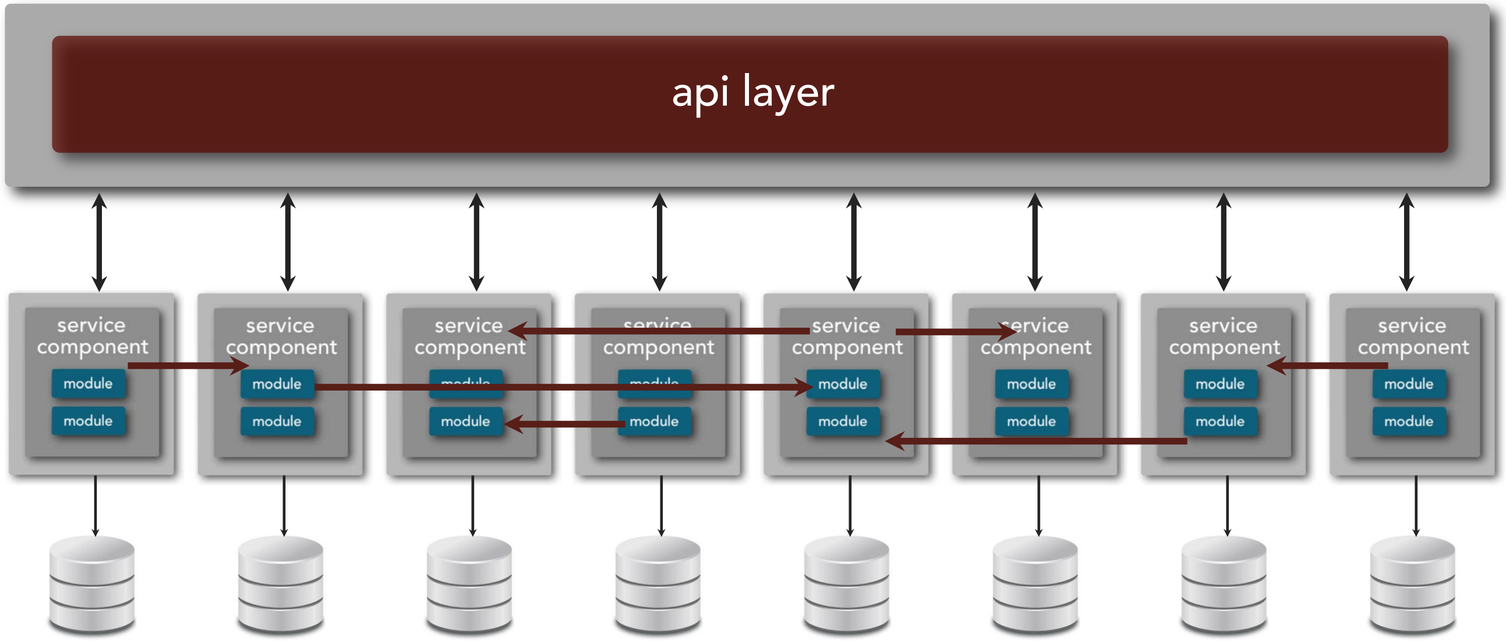


workflow and choreography

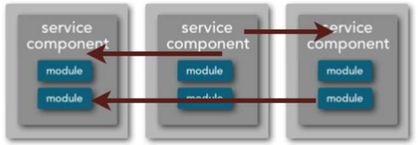
service granularity



workflow and choreography

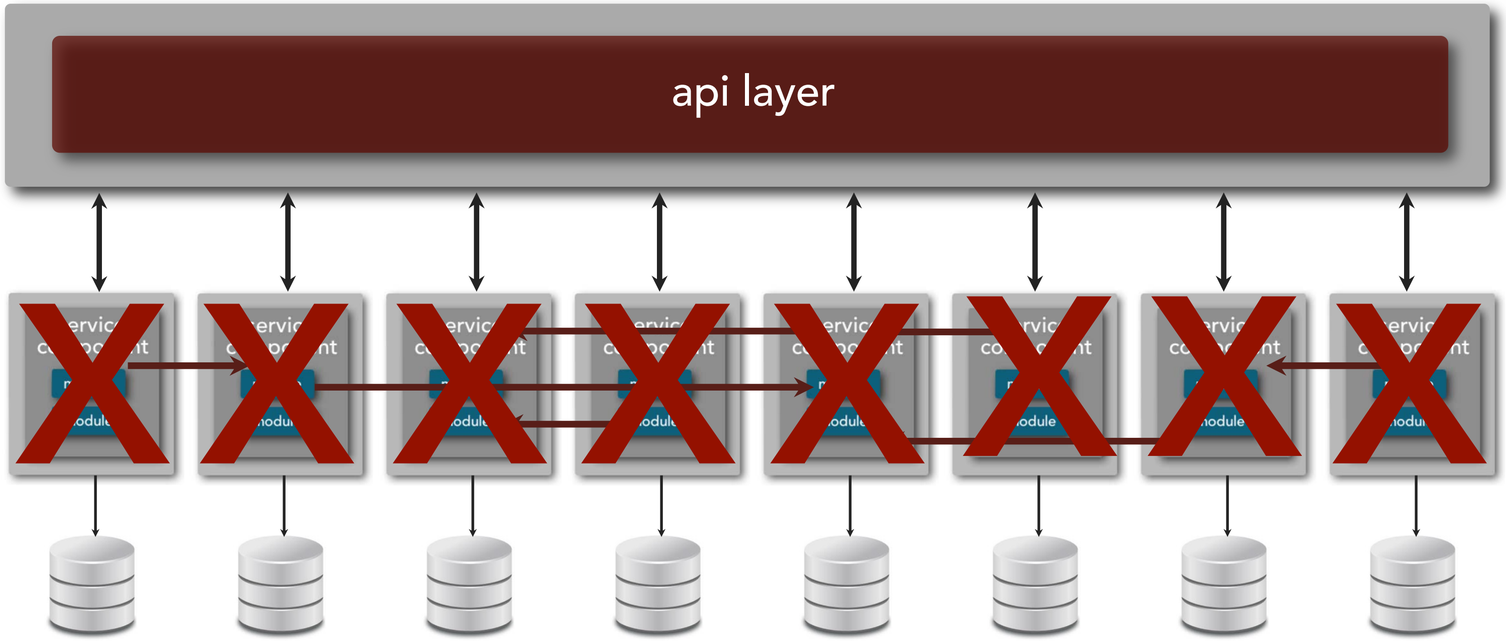


service granularity

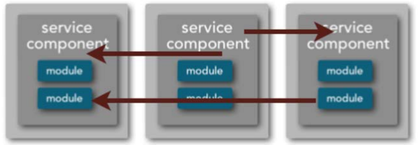


workflow and choreography

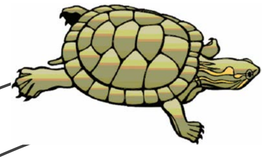
fault tolerance (availability)



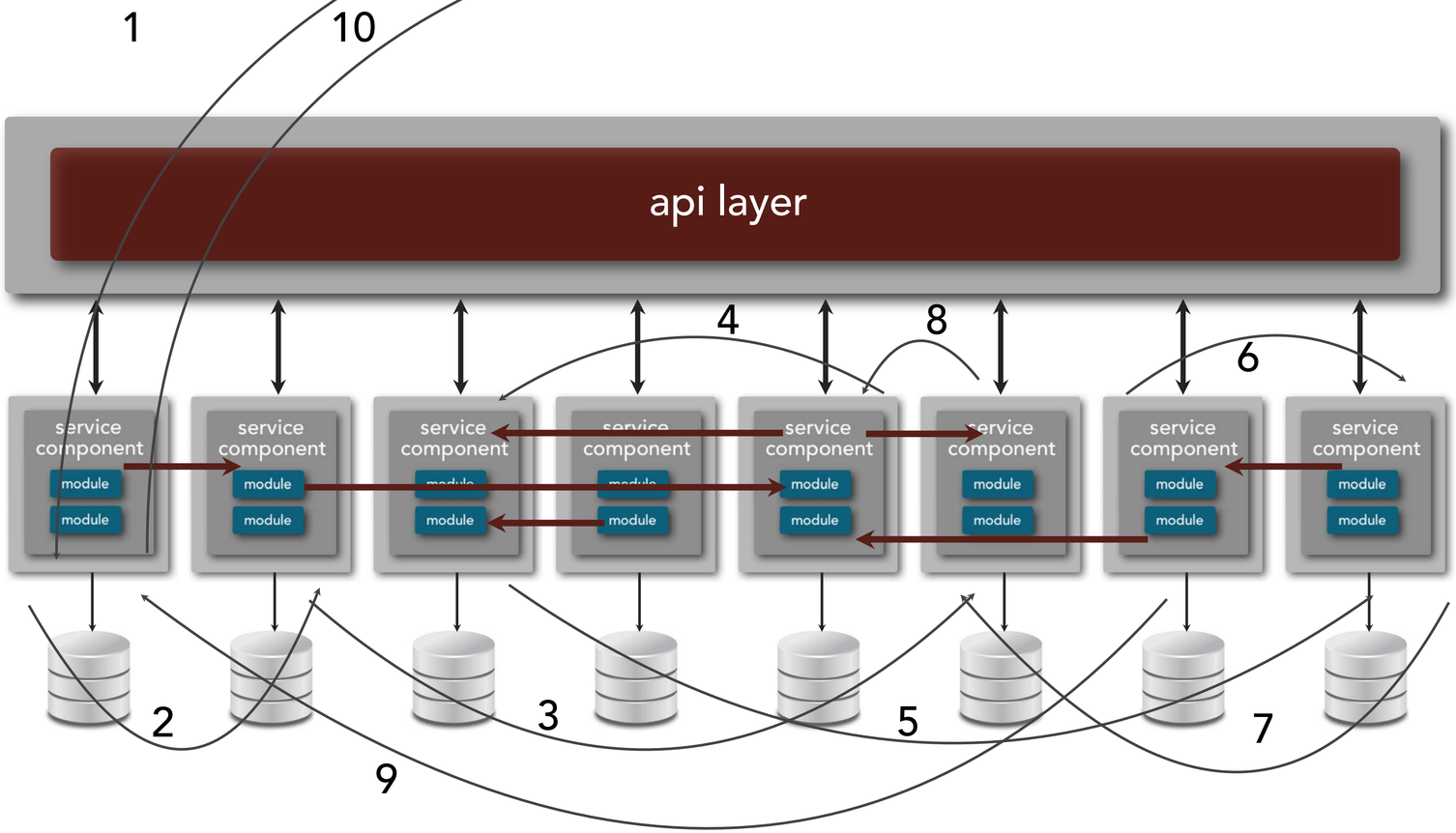
service granularity



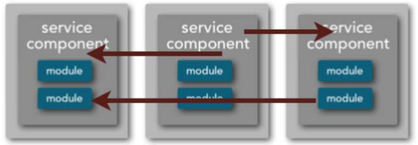
workflow and choreography



performance due to latency and security

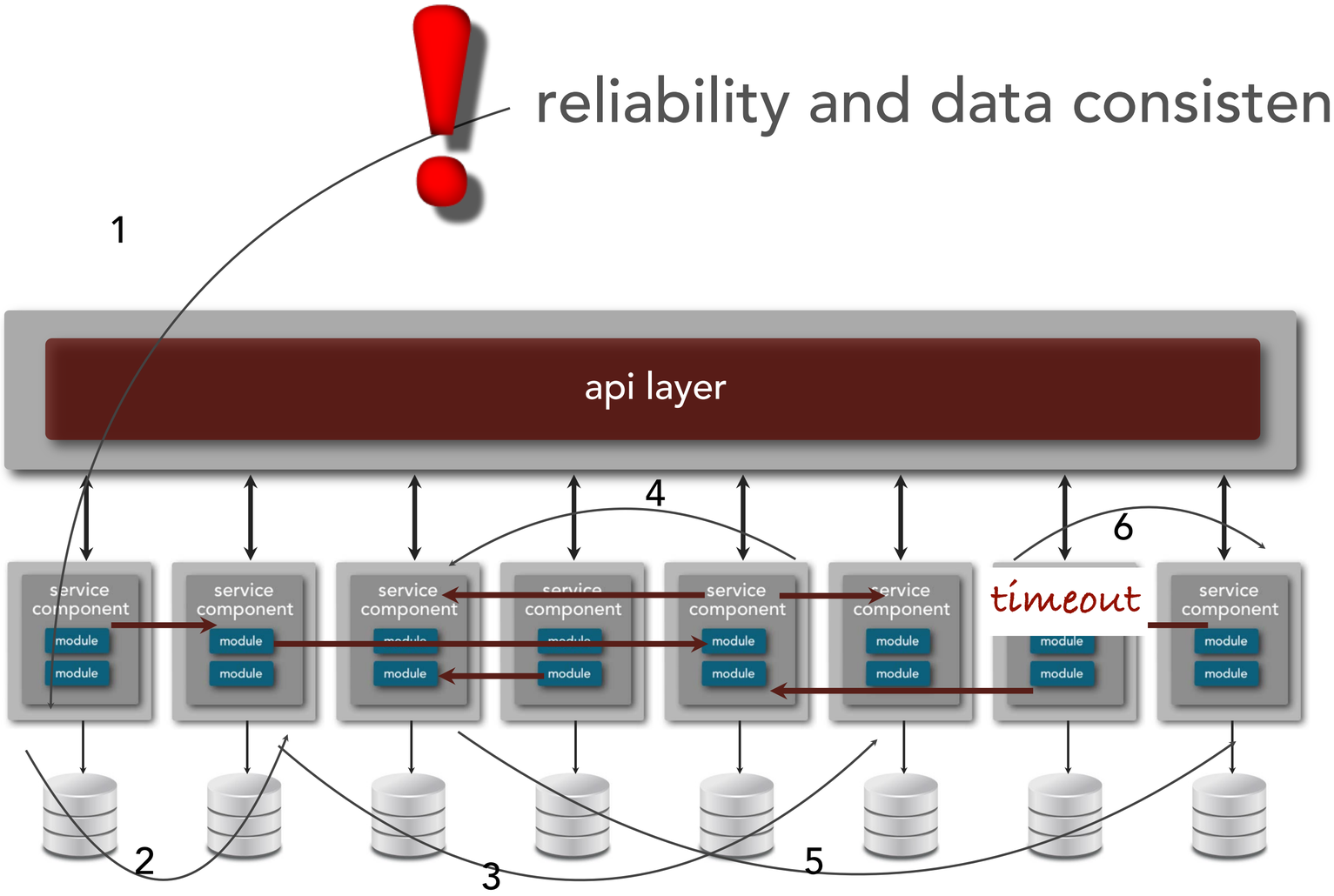


service granularity

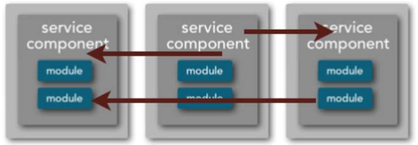


workflow and choreography

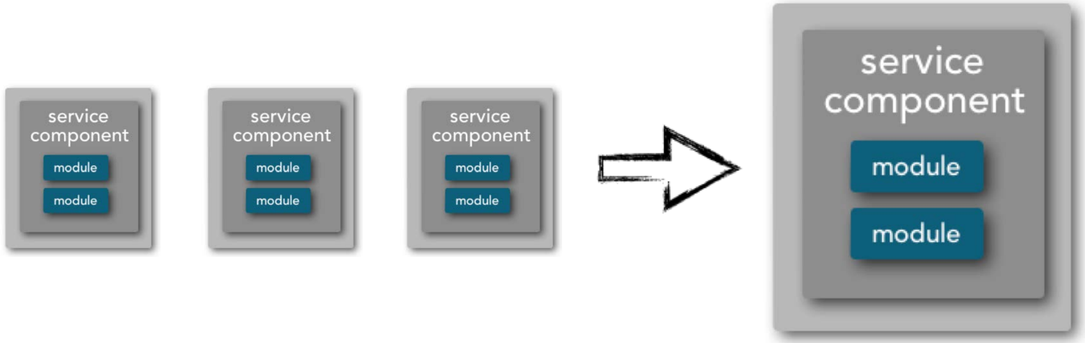
reliability and data consistency

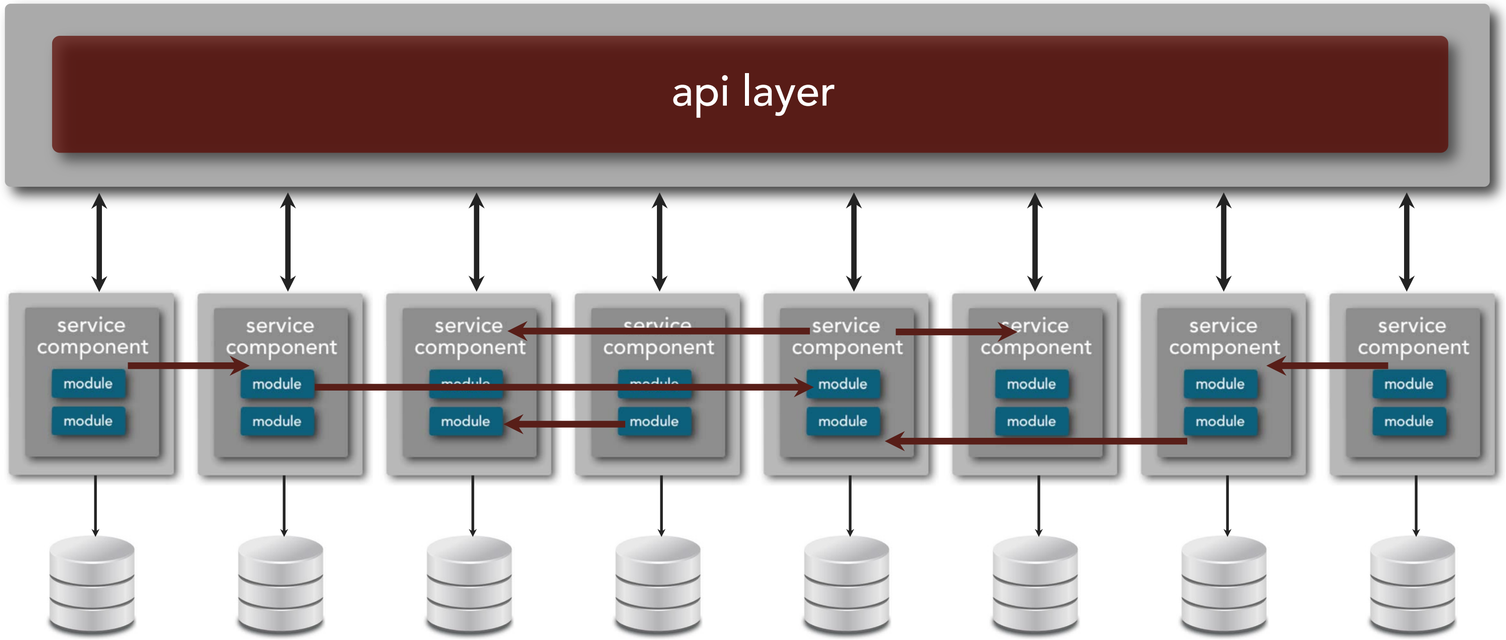


service granularity

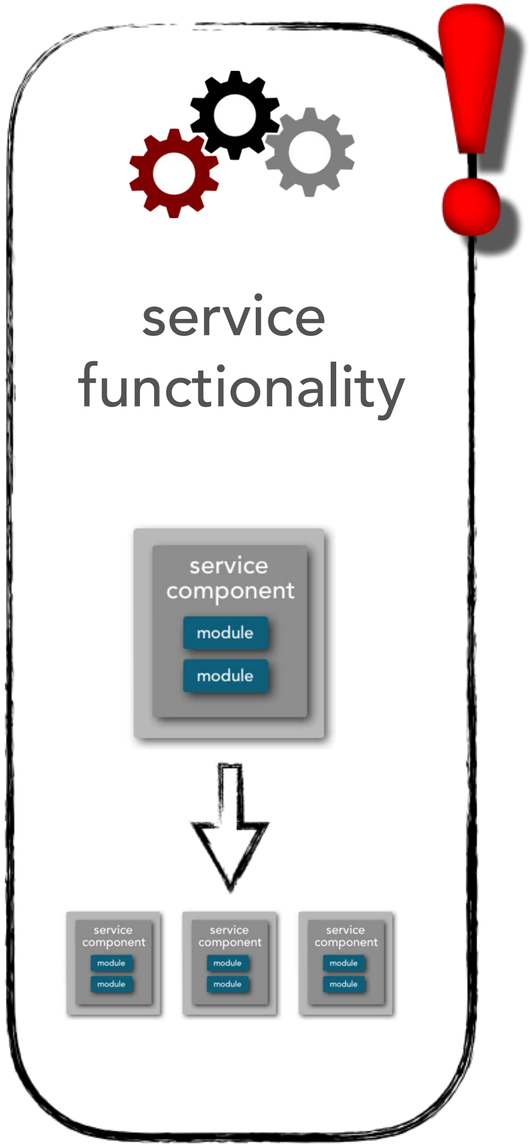


workflow and choreography

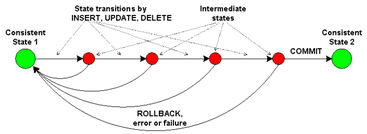




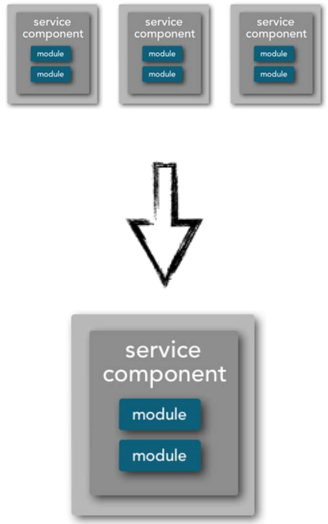
service granularity



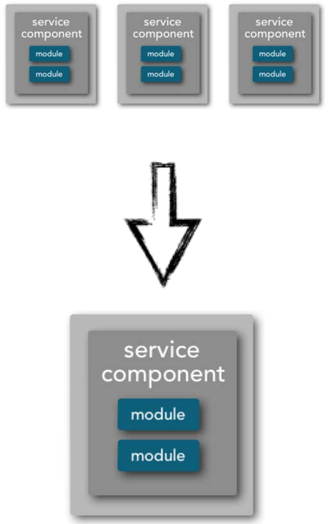
service functionality

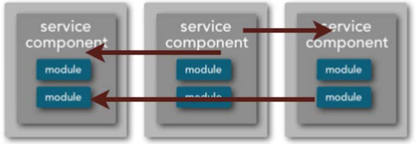


database transactions

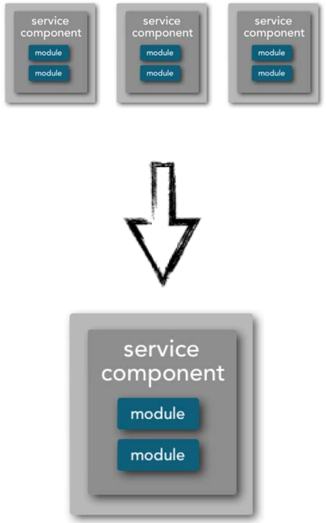


data dependencies



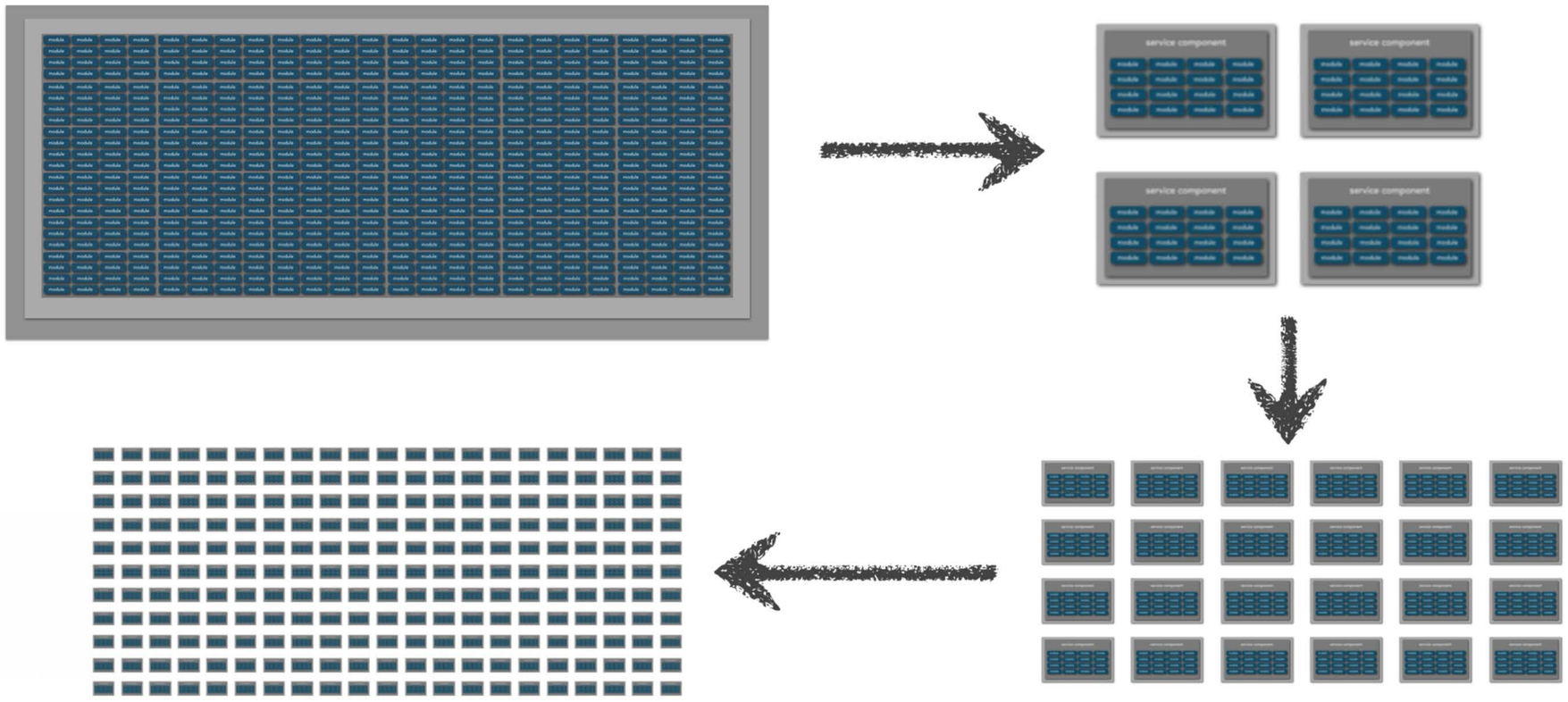


workflow and choreography



service granularity

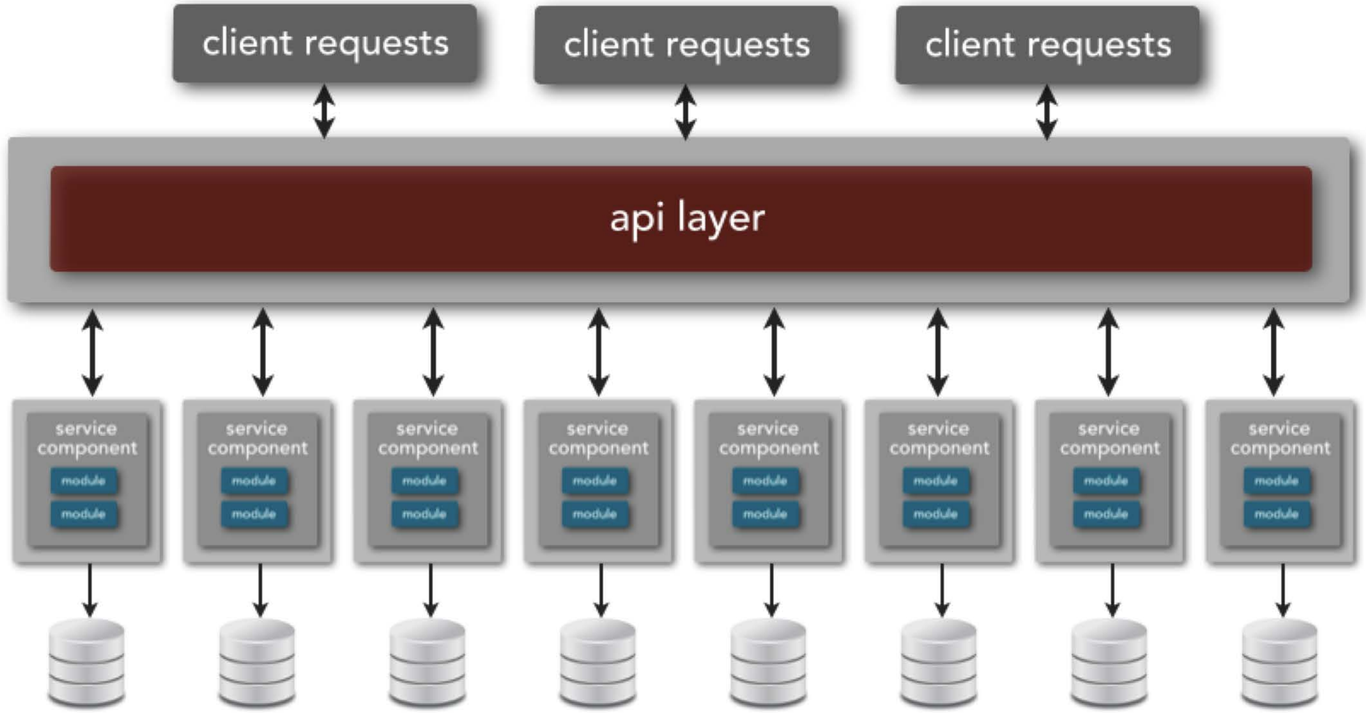
get granularity right by iterating



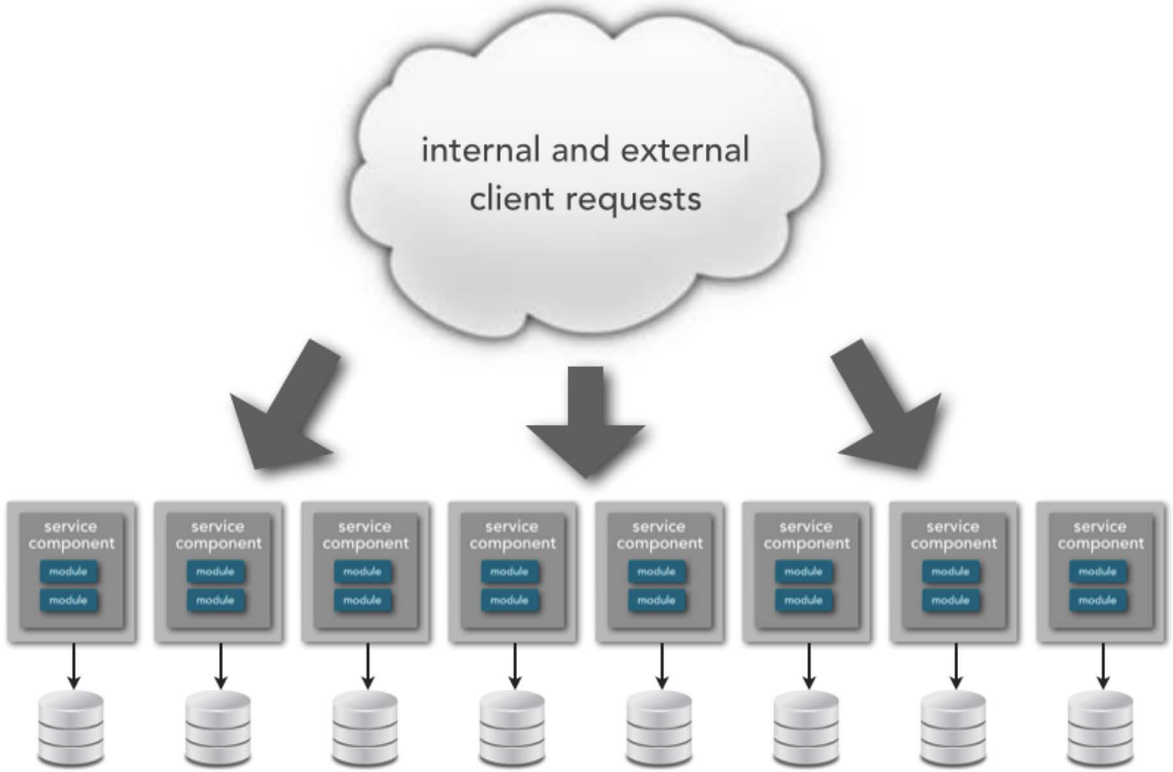
API Layer

api layer

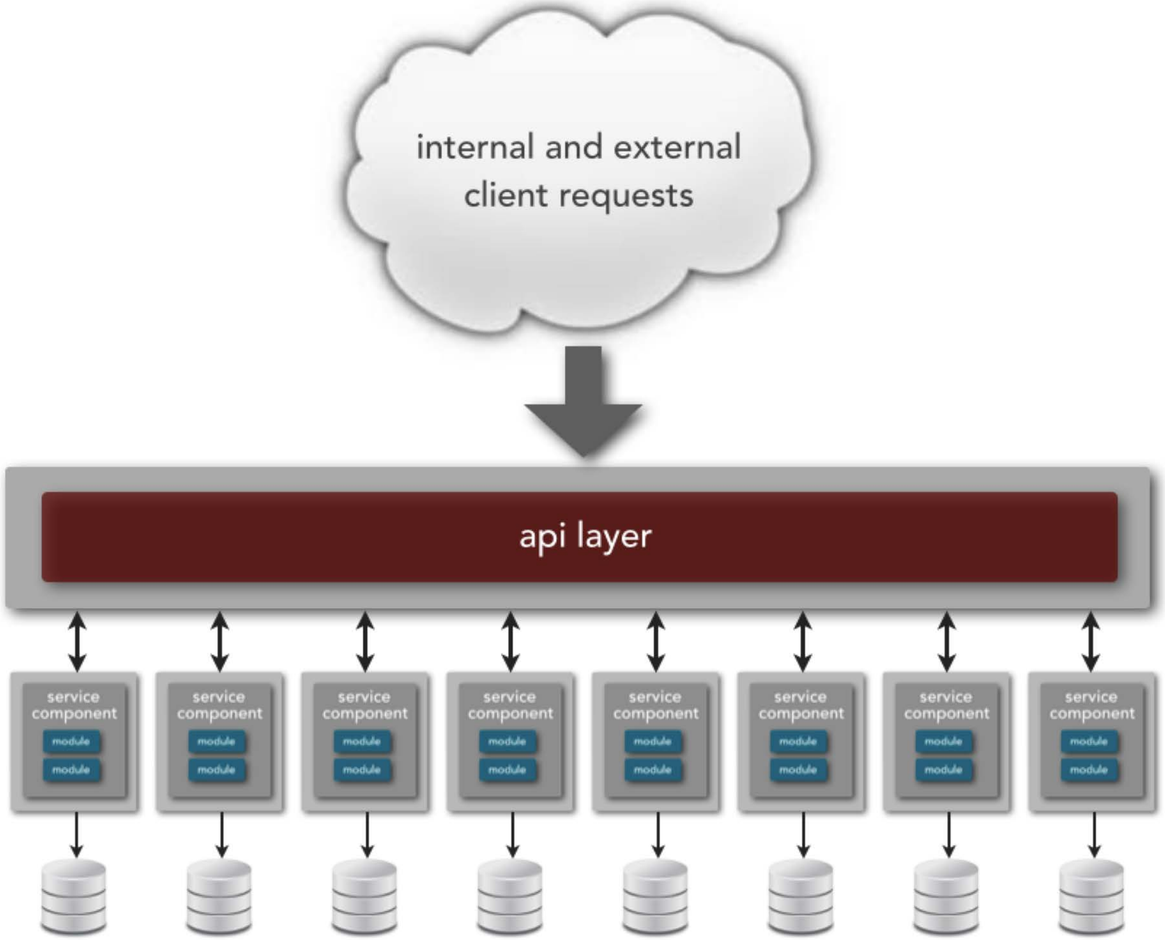
hides the actual endpoint of the service, exposing only those services available for public consumption



api layer

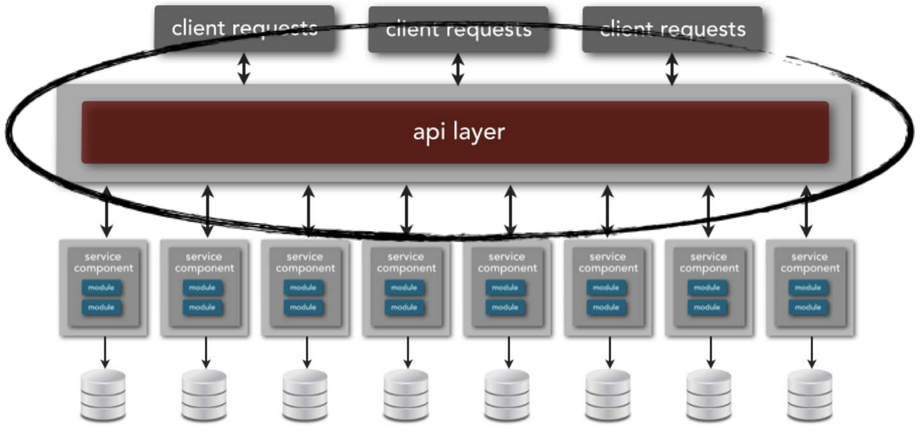

internal and external
client requests
service component
module
module

api layer


internal and external
client requests
api layer
service component
module
module

api layer

endpoint proxy



APACHE HTTP SERVER

HAPROXY

Microsoft IIS

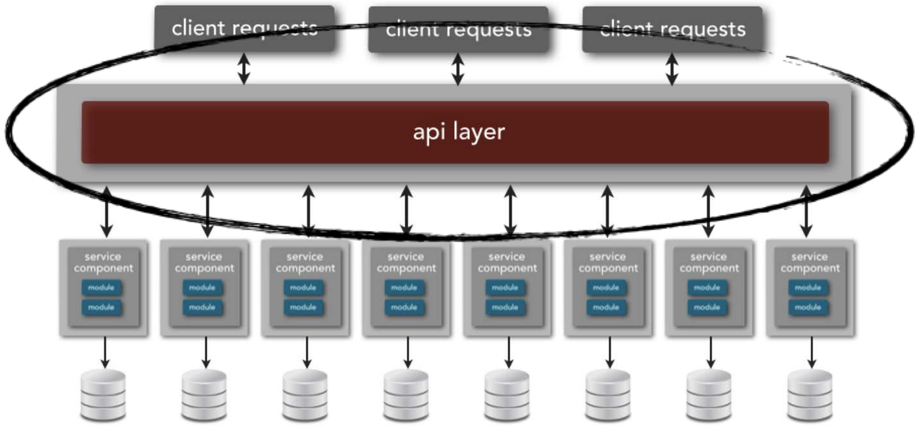
NGINX http server

PROXY SERVER Squid

ZIPROXY

api layer

gateway (integration hub)



Apache Camel

muleESB

JBoss by Red Hat

IBM

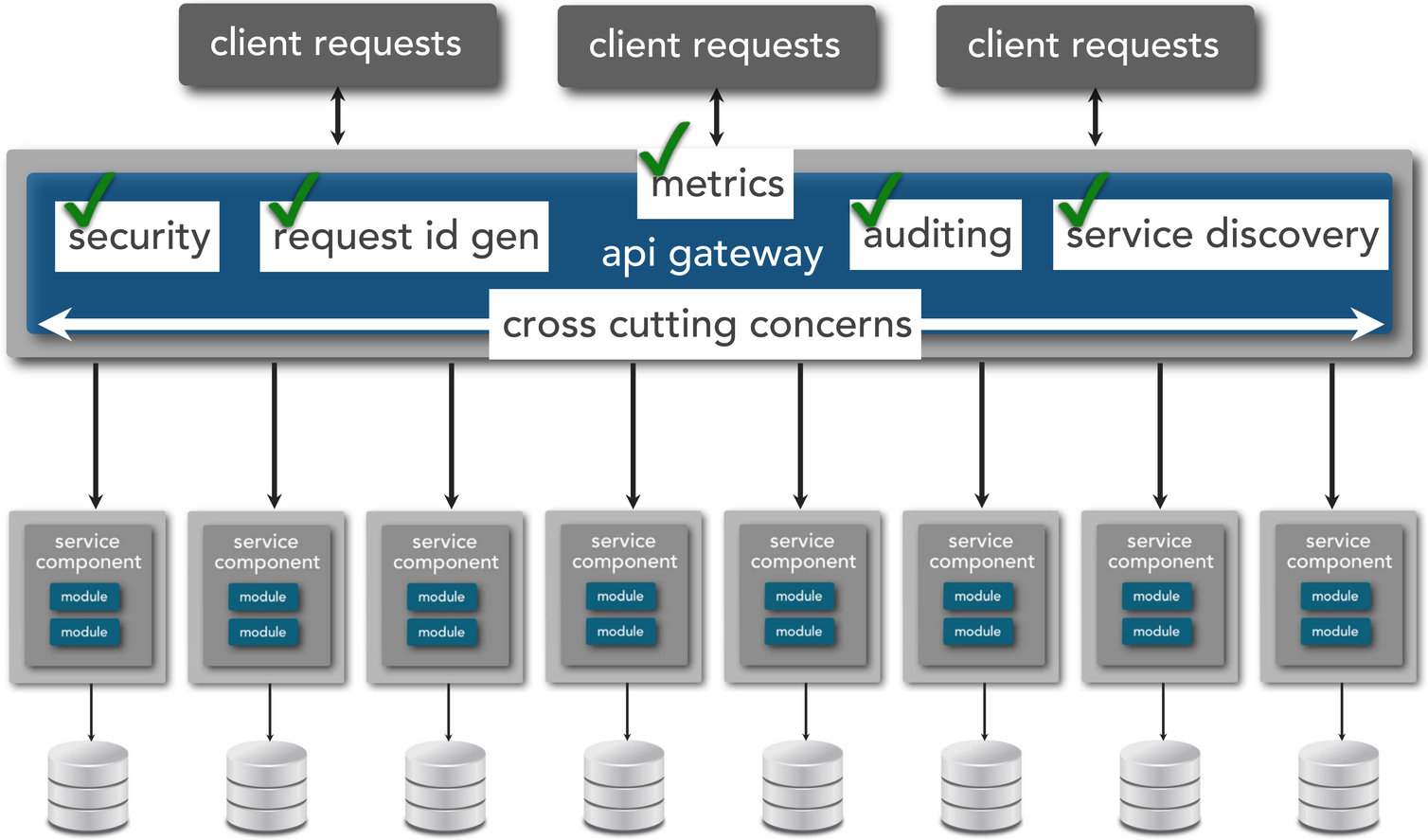
ORACLE

Microsoft BizTalk Server

ca technologies API Gateway

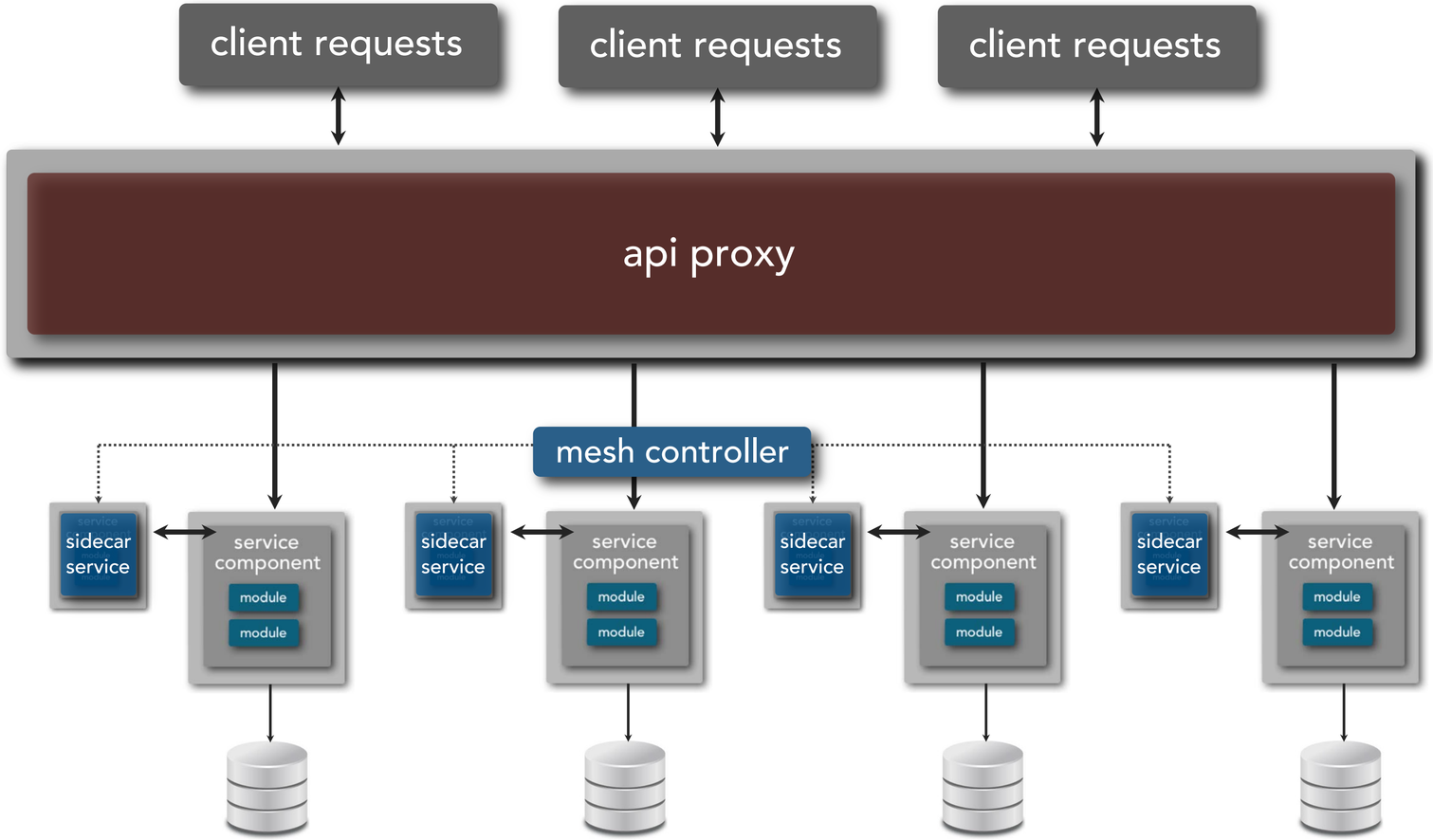
api layer

gateway (integration hub)



api layer

service mesh



Istio

CONDUIT

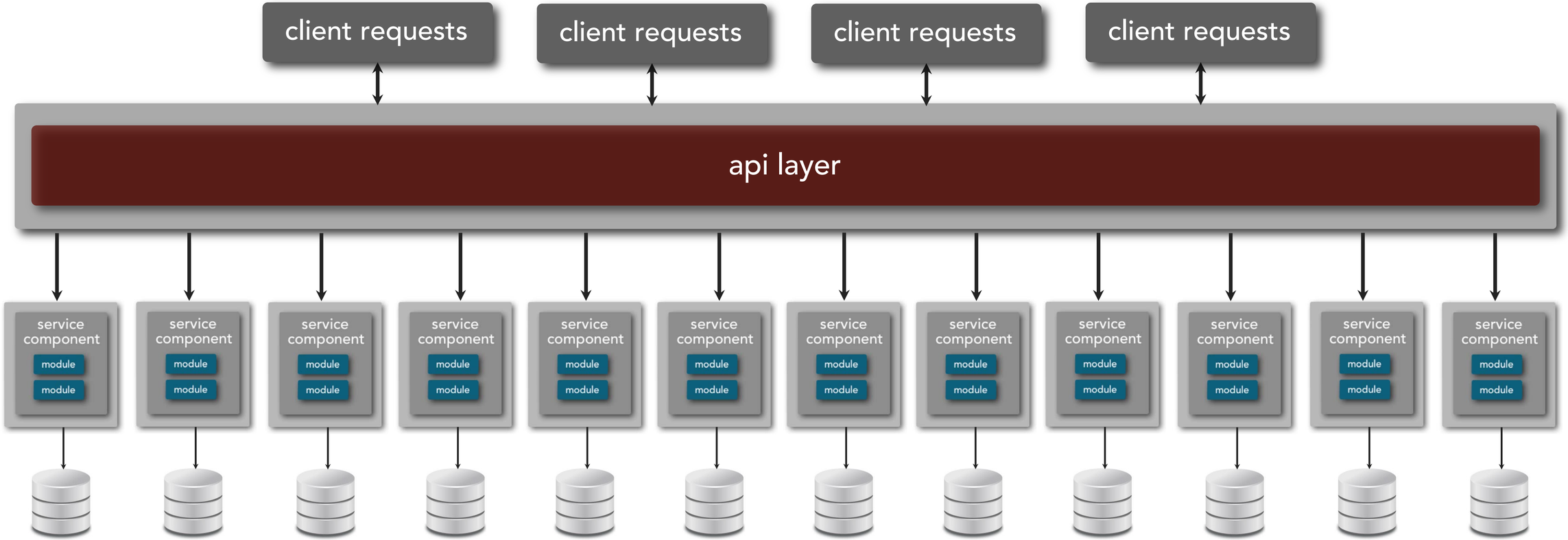
kubernetes

envoy

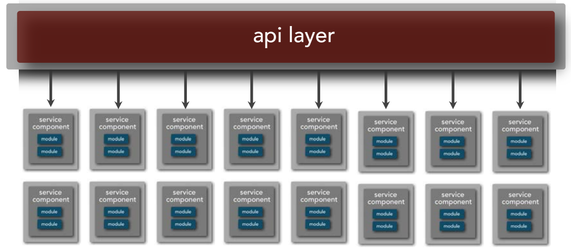
linkerd

ASPEN MESH

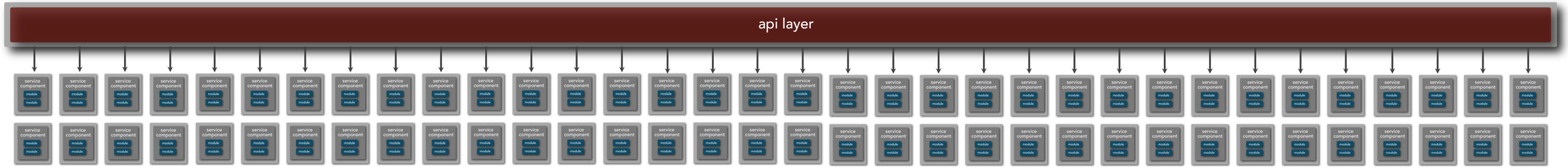
service domains



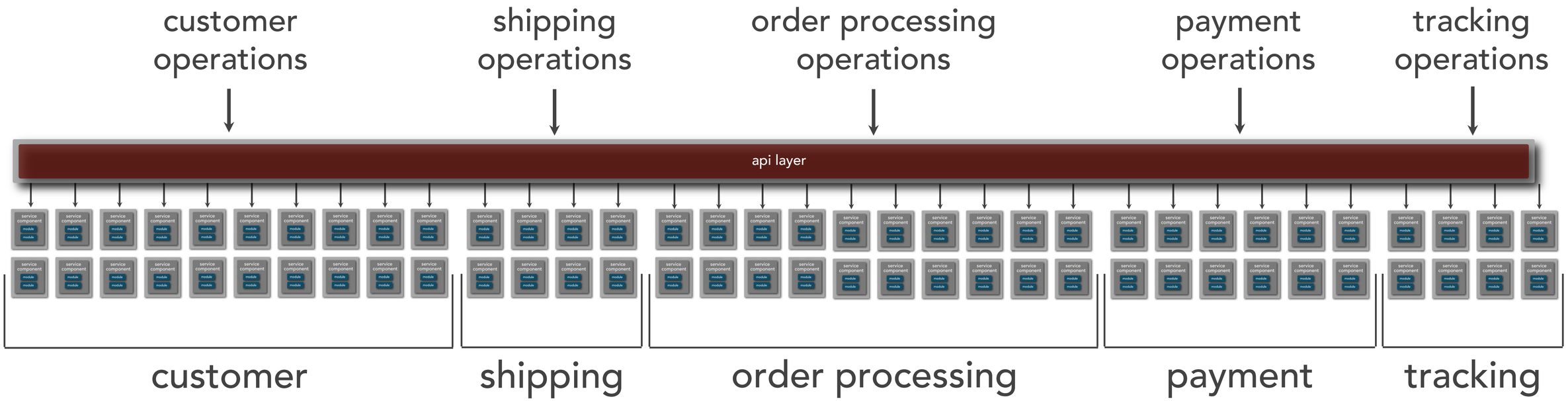
service domains



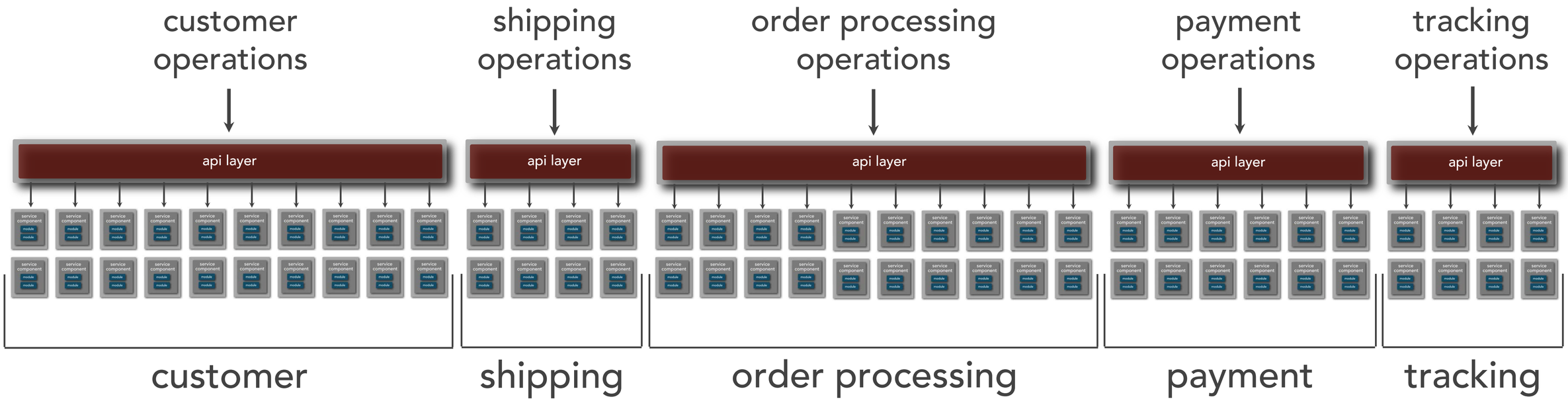
service domains



service domains

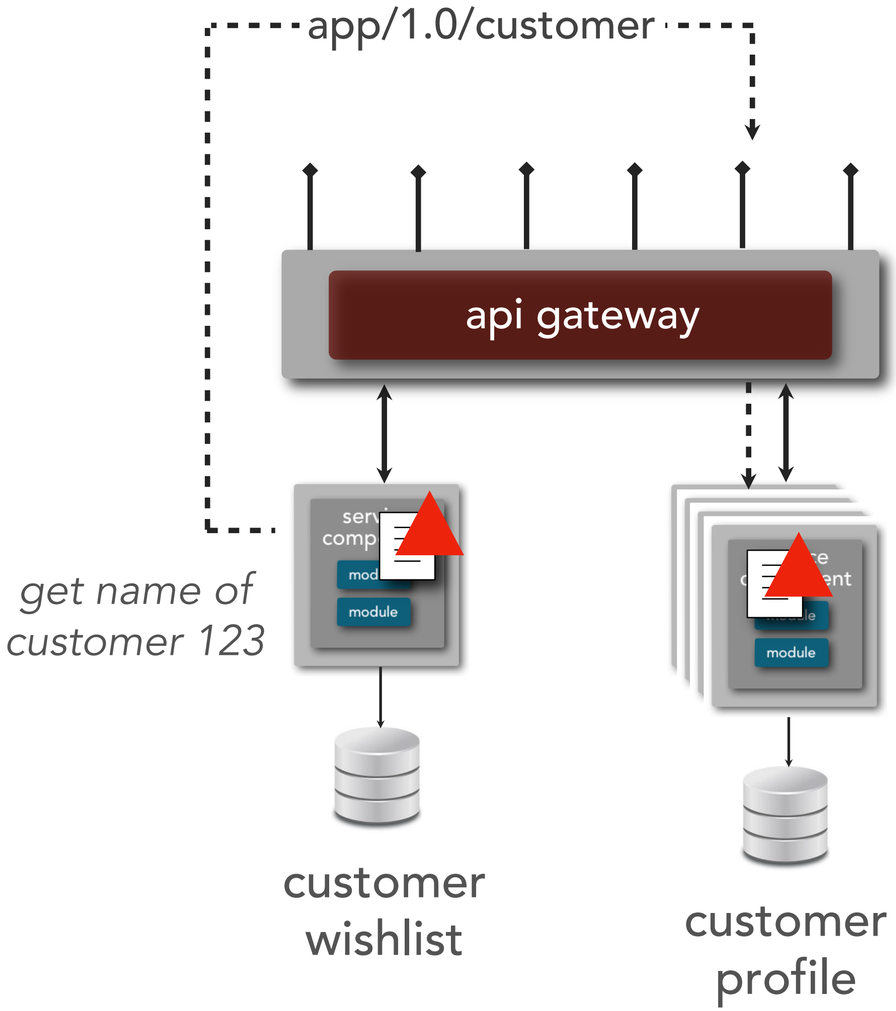


service domains



Communication Challenges

communication challenges



stamp coupling

app/1.0/customer

45 attributes
500K

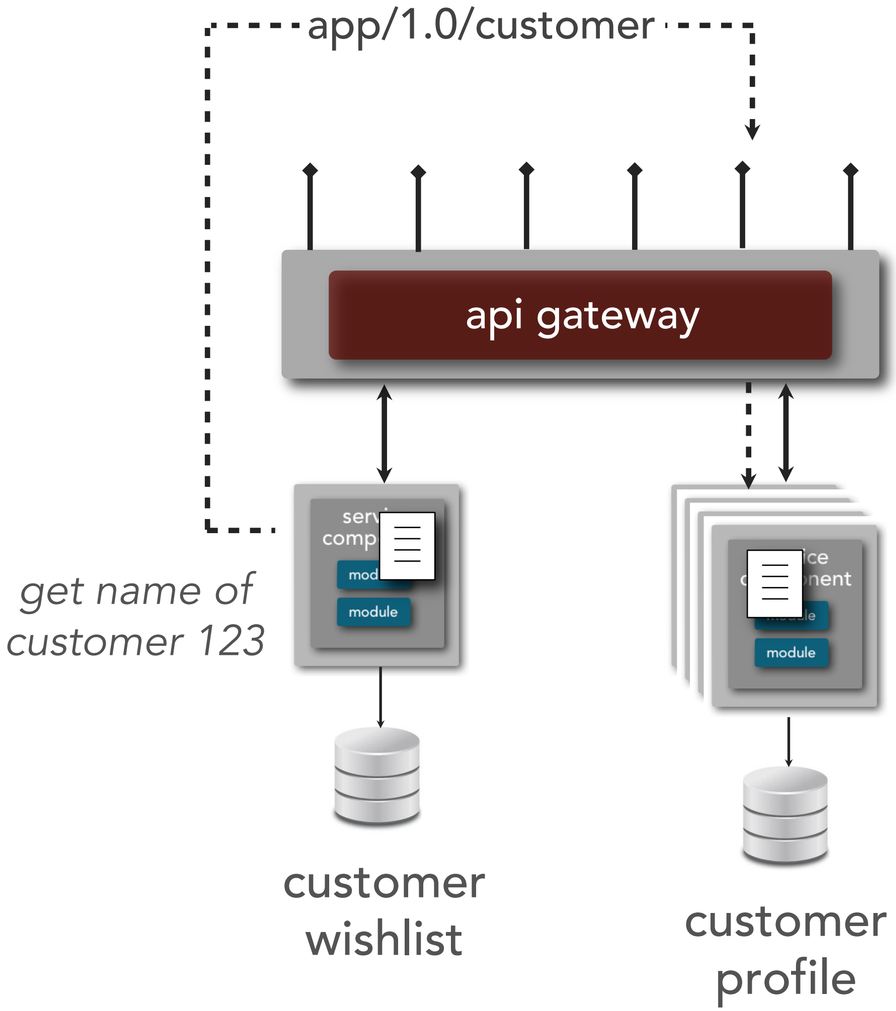
1 attributes
200 bytes

2000/sec

1,000,000kb/sec

400kb/sec

communication challenges

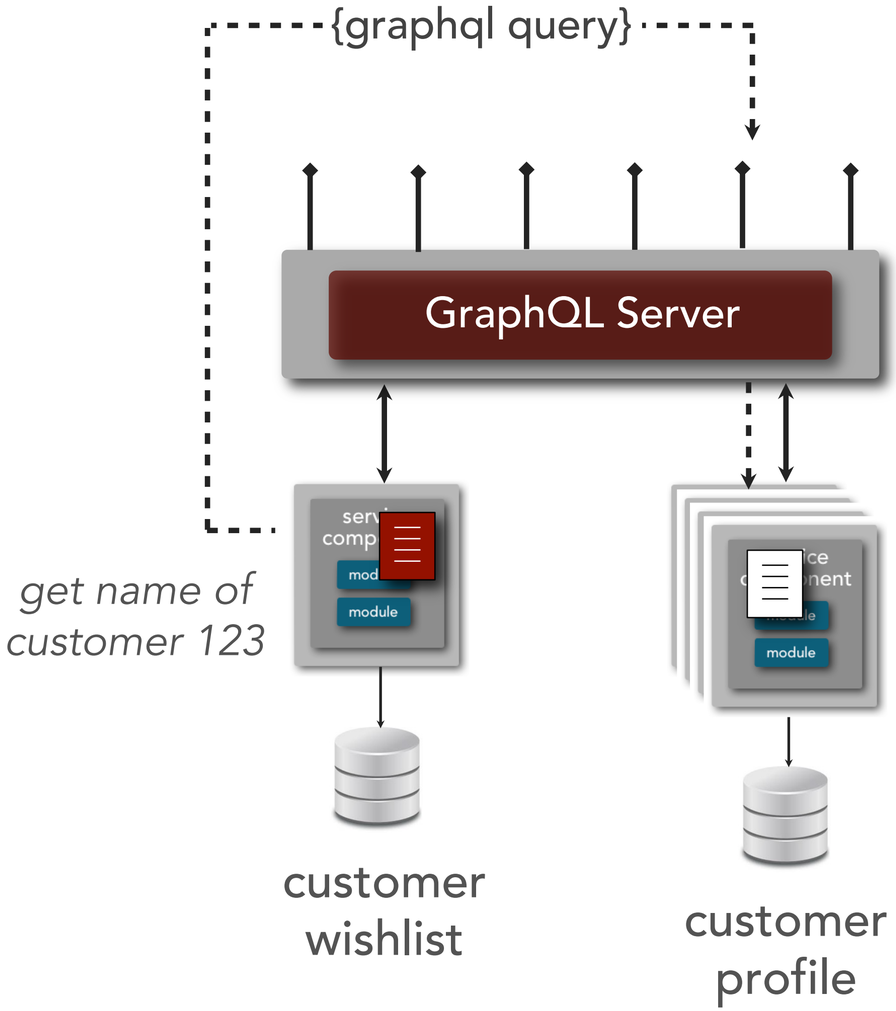


field selector

app/1.0/customer?fields=name

- ✗ stamp coupling
- ✓ bandwidth

communication challenges



GraphQL

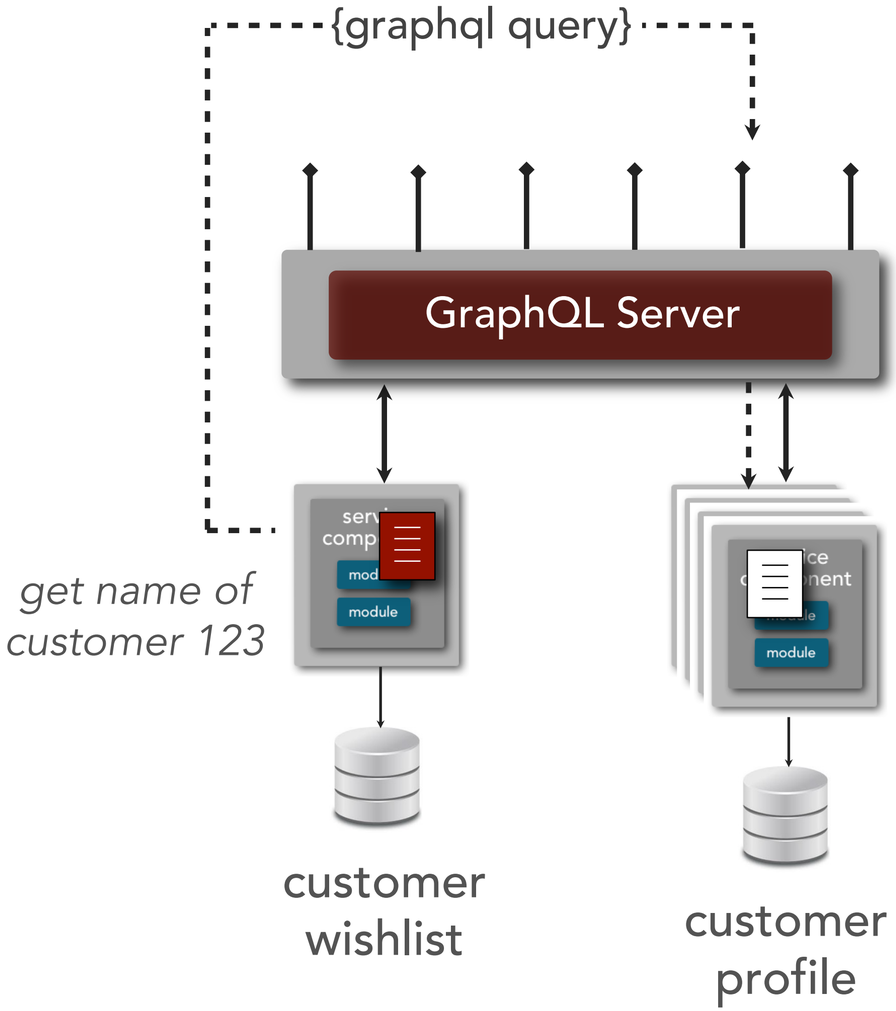
customer wishlist

```
type Profile {
    name: String
}
```

customer profile

```
type Profile {
    name: String
    addr1: String
    addr2: String
    country: String
    ...
}
```

communication challenges



GraphQL

customer wishlist

```
type Profile {
    name: String
}
```

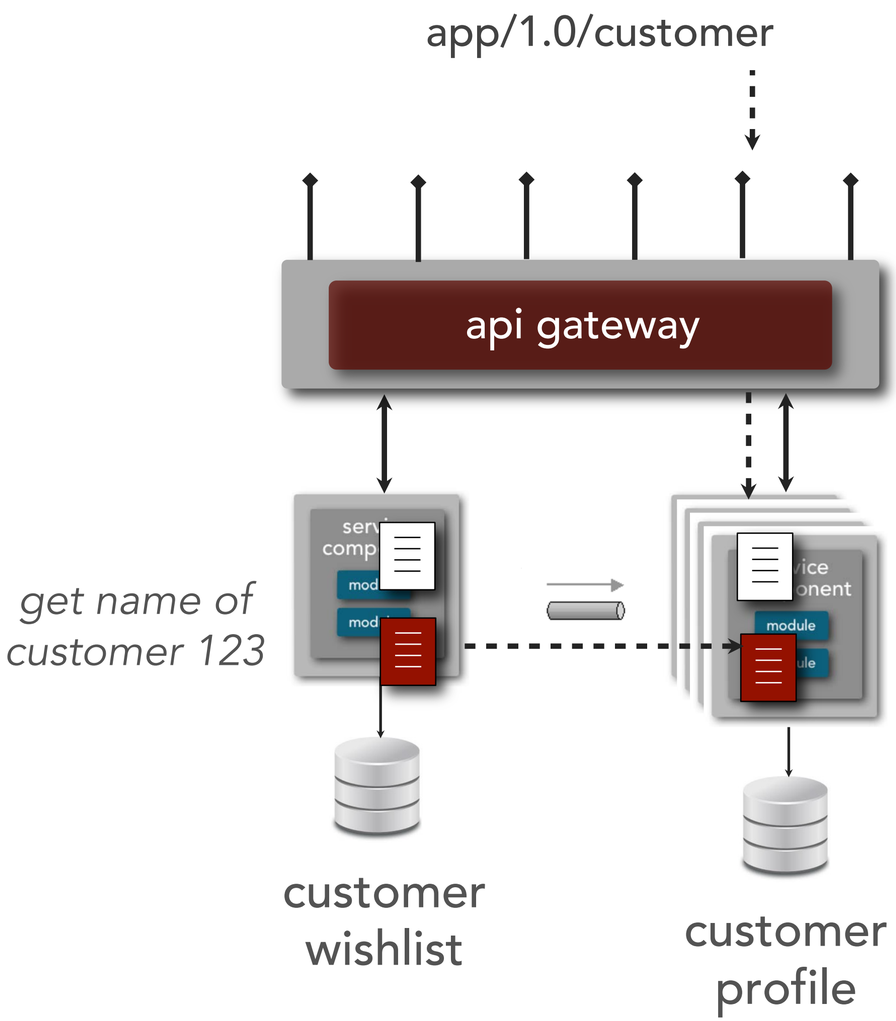
customer profile

```
type Profile {
    name: String
    addr1: String
    addr2: String
    addr3: String
    country: String
    ...
}
```

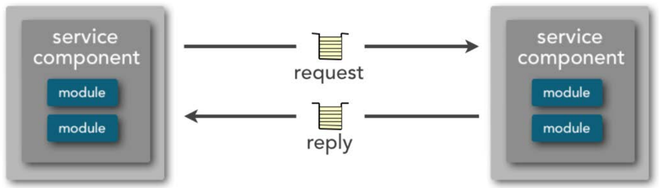
✓ stamp coupling

✗ bandwidth

communication challenges



messaging



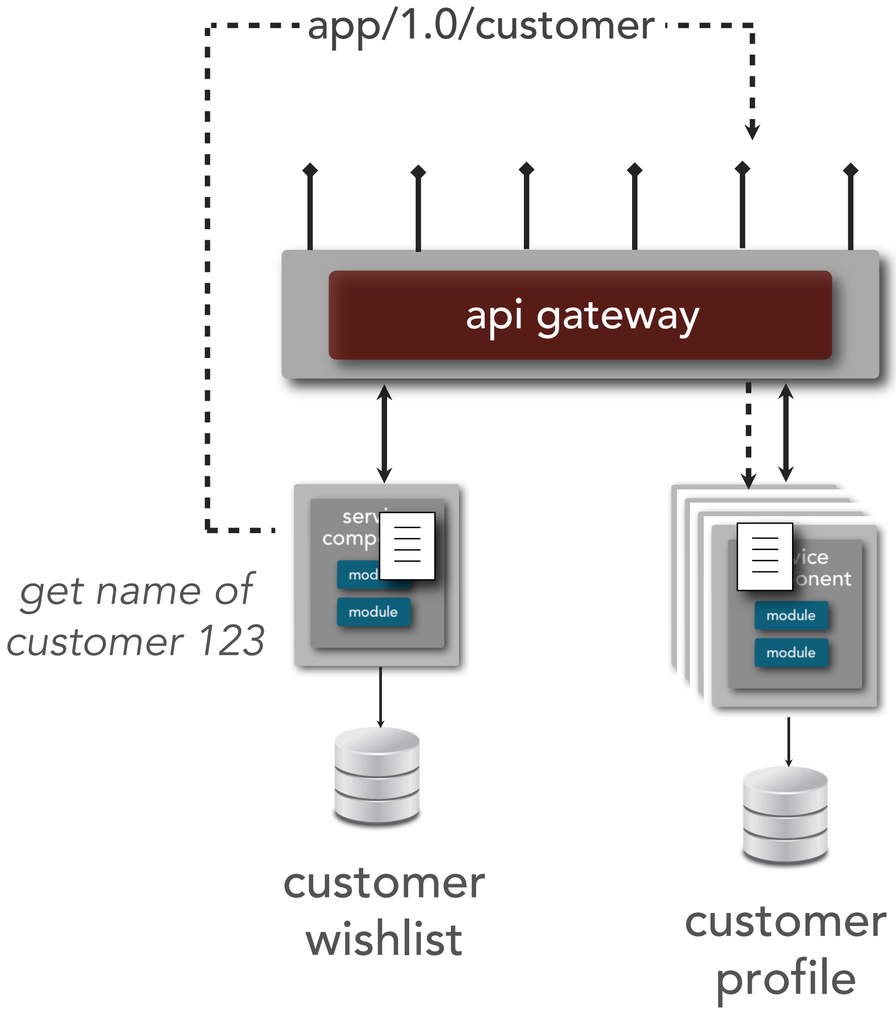
app/1.0/customer (REST): 45 attributes, 500kb

send/receive from queue: 1 attributes, 200b

✓ stamp coupling

✓ bandwidth

communication challenges



value-driven contracts

customer wishlist

[name: String]

customer profile

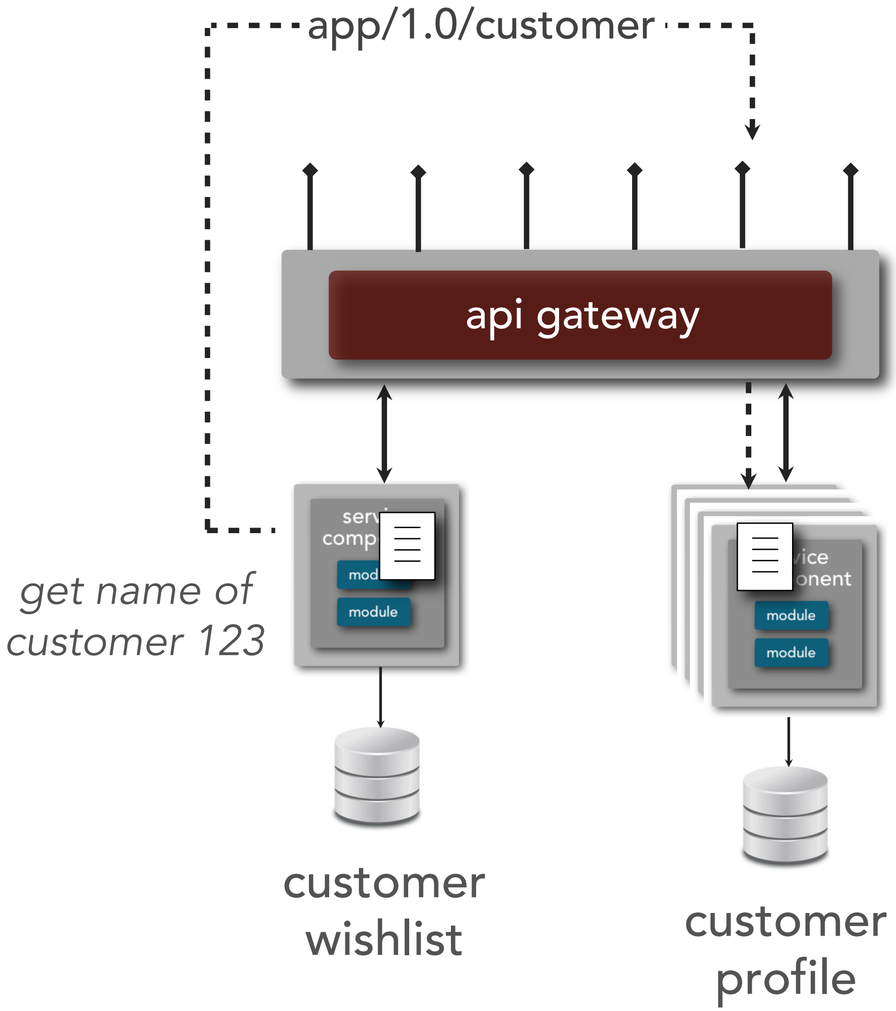
[name: String]
[addr1: String]
[addr2: String]
[country: String]
…

```
System.out.println("Starting Apache Ignite...")
CacheConfiguration cfg = new CacheConfiguration
cfg.setCacheMode(CacheMode.LOCAL);
IgniteConfiguration ic = new IgniteConfiguratio
ic.setClientMode(false);
Ignite ignite = Ignition.start(ic);
IgniteCache<String, String> cache = ignite.get
System.out.println("Apache Ignite started");
System.out.println();

String customerId = getCustomerIdFromRequest();
String newName = getNameFromRequest();
```

wishlist test

communication challenges



value-driven contracts

customer wishlist

[name: String]

customer profile

[name: String]
[addr1: String]
[addr2: String]
[addr3: String]
[country: String]
…

✓ stamp coupling

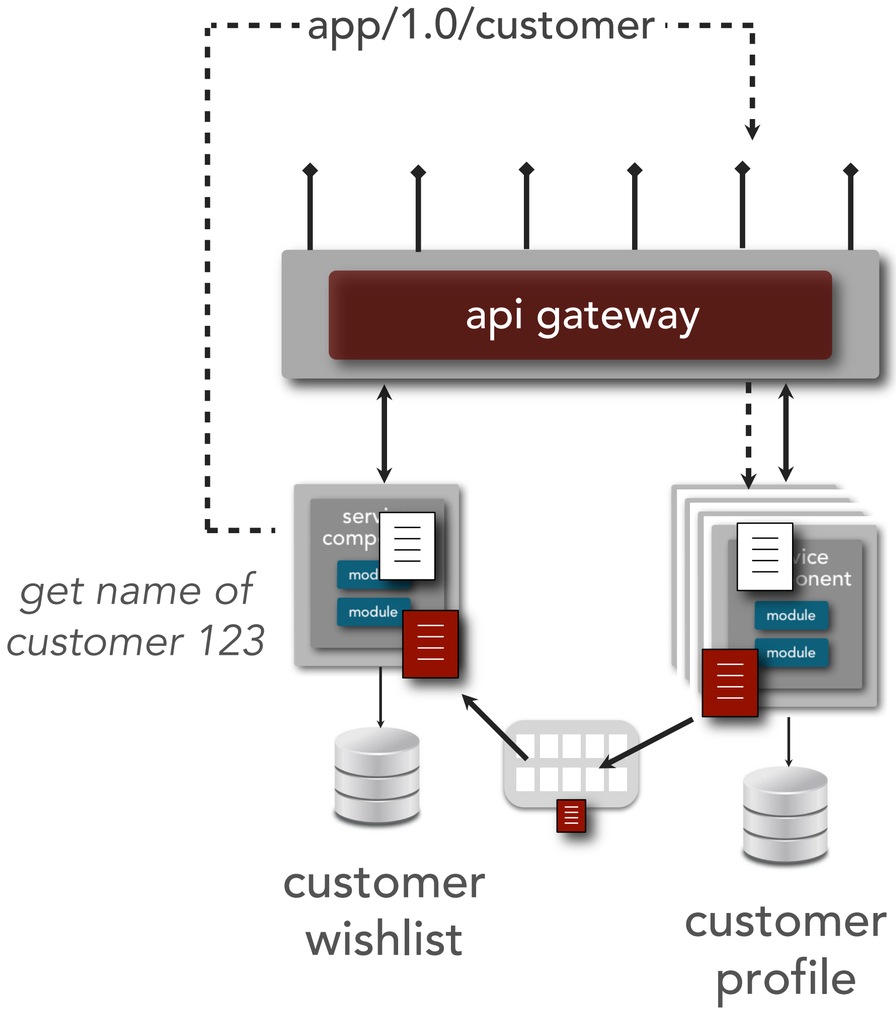
✓ bandwidth

```
System.out.println("Starting Apache Ignite...")
CacheConfiguration cfg = new CacheConfiguration
cfg.setCacheMode(CacheMode.LOCAL);
IgniteConfiguration ic = new IgniteConfiguratio
ic.setClientMode(false);
Ignite ignite = Ignition.start(ic);
IgniteCache<String, String> cache = ignite.get
System.out.println("Apache Ignite started");
System.out.println();

String customerId = getCustomerIdFromRequest();
String newName = getNameFromRequest();
```

wishlist test

communication challenges



replicated or distributed caching

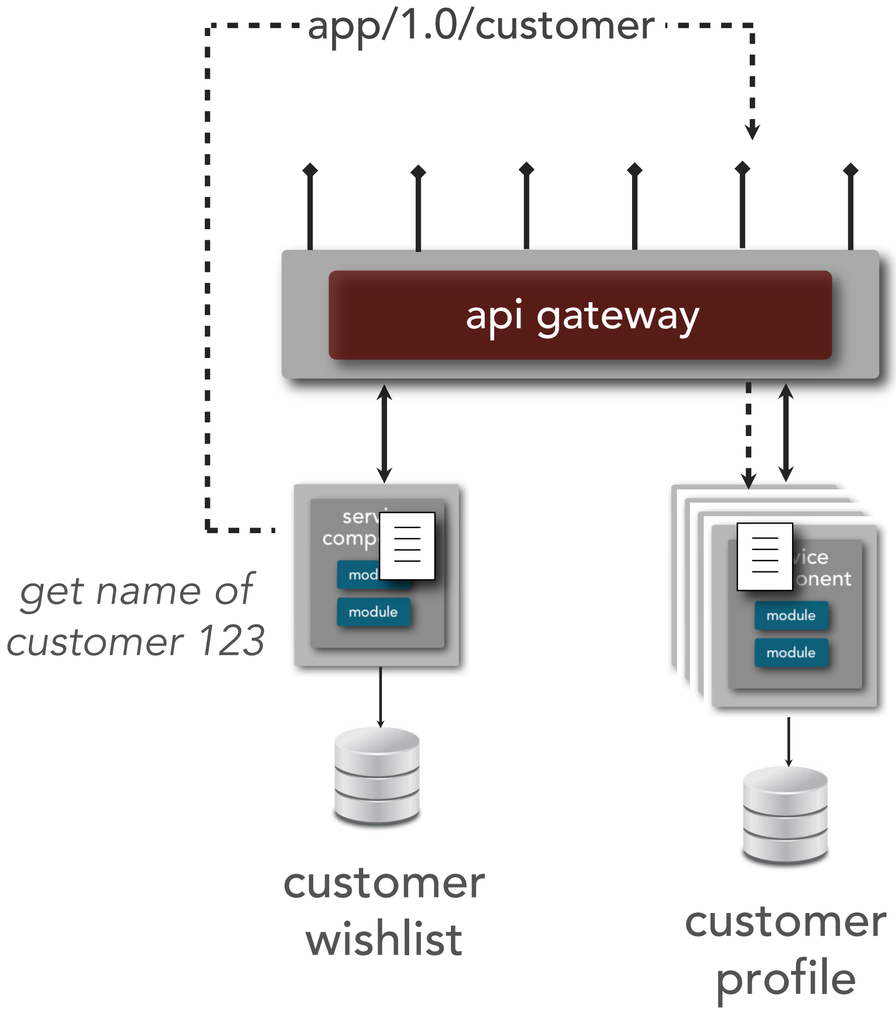
name cache

[name: String]

✓ stamp coupling

✓ bandwidth

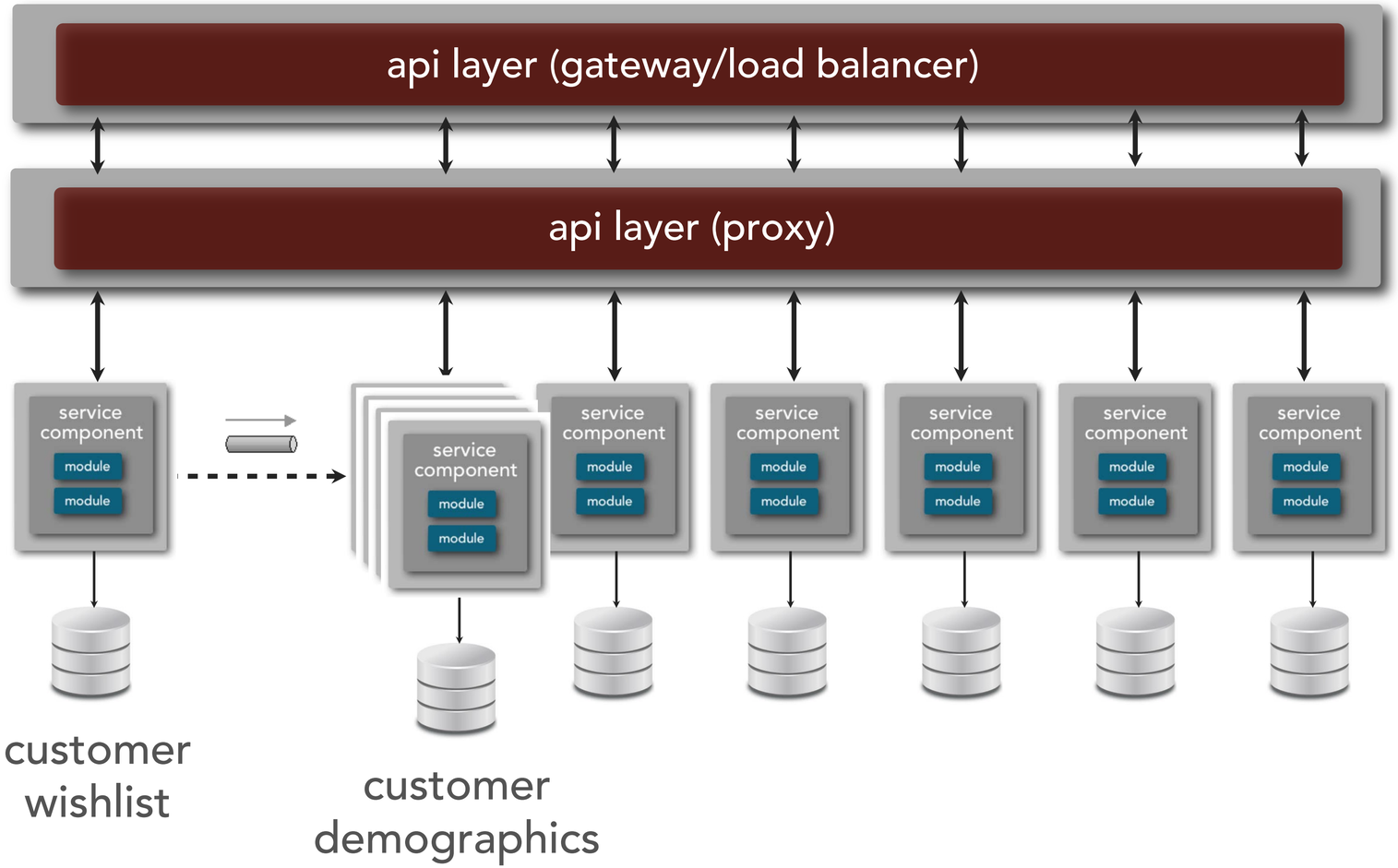
communication challenges



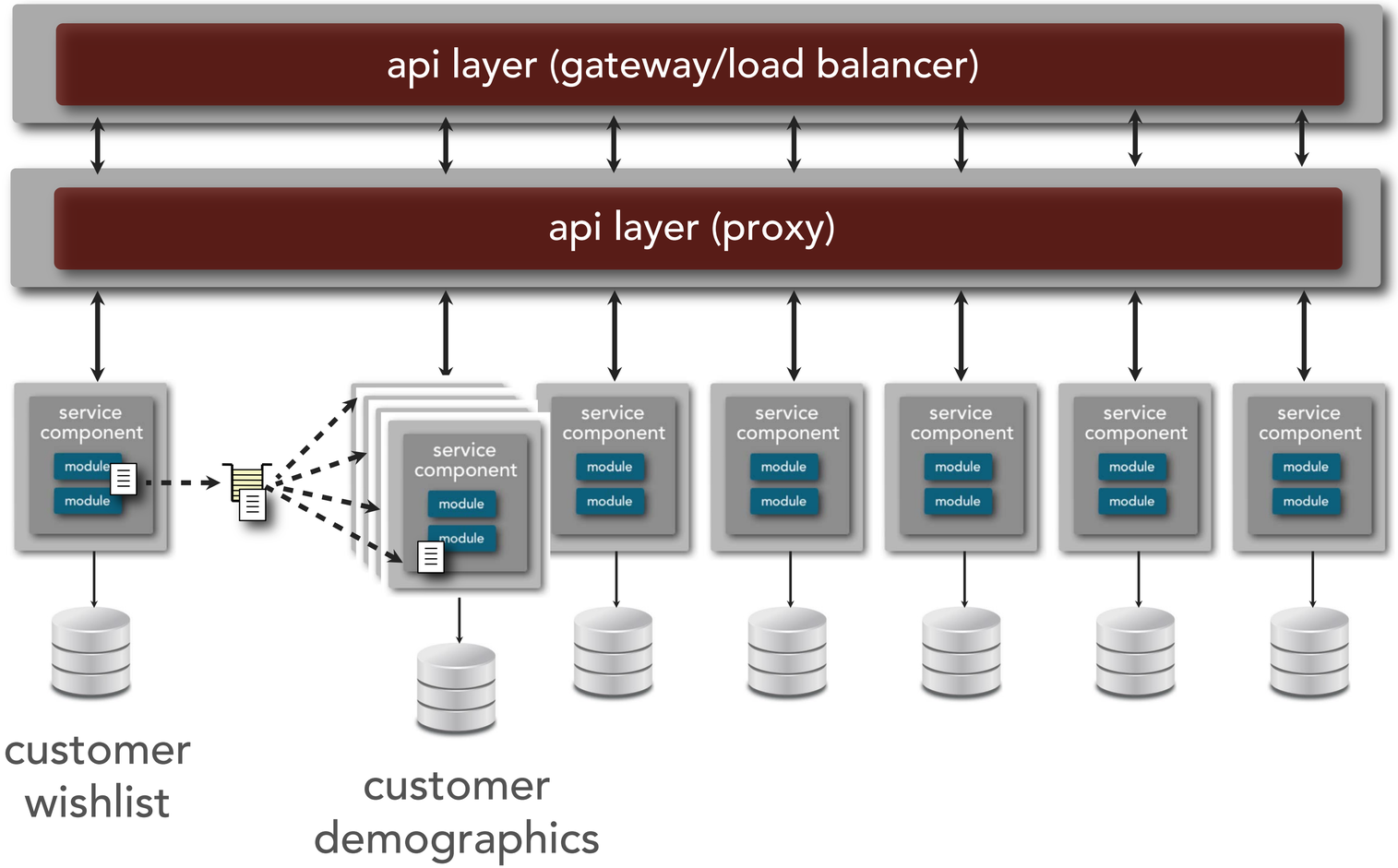
	stamp coupling	bandwidth
field selector	✗	✓
graphql	✓	✗
messaging	✓	✓
value-driven contracts	✓	✓
caching	✓	✓

Event-Driven Microservices

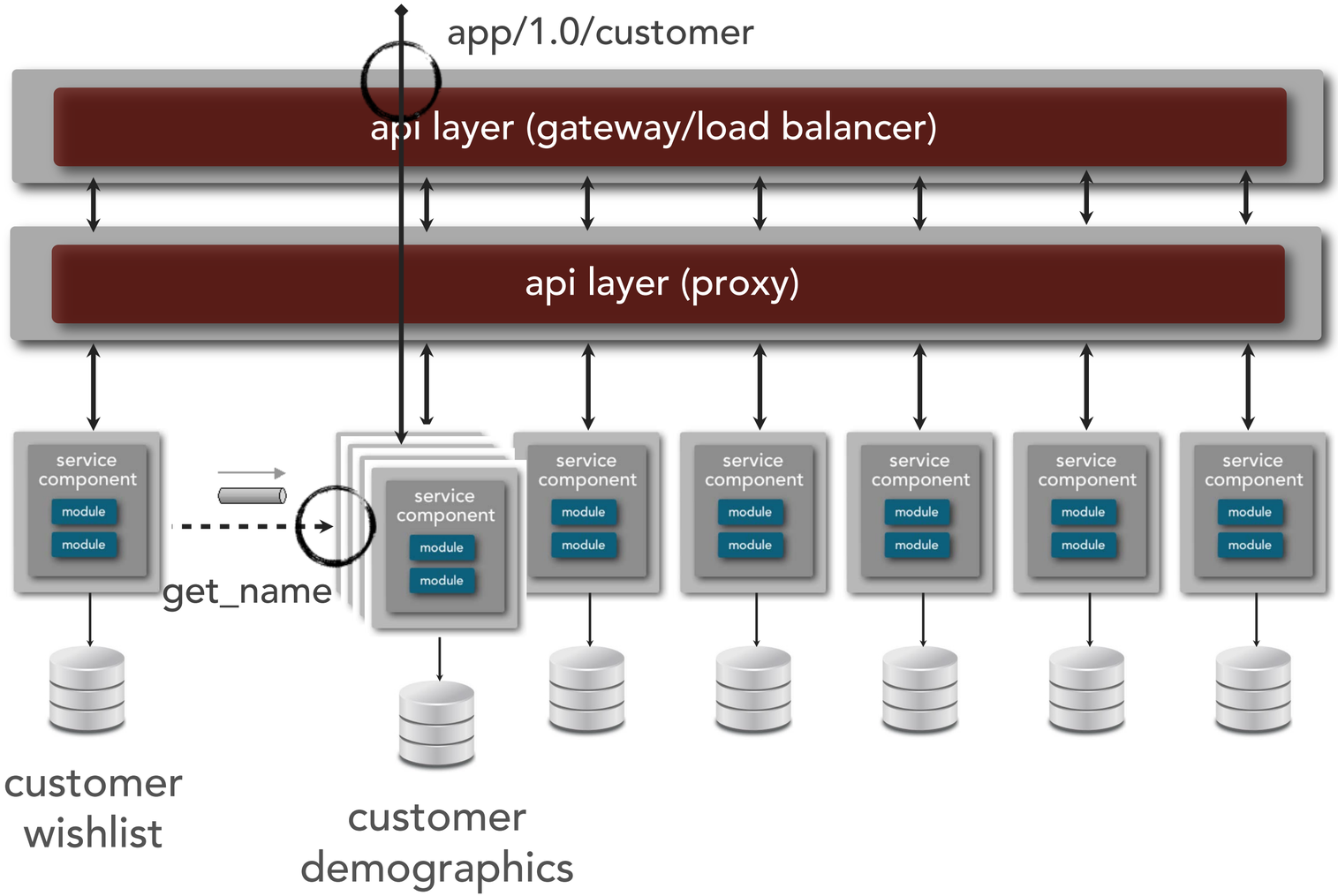
event-driven microservices



event-driven microservices

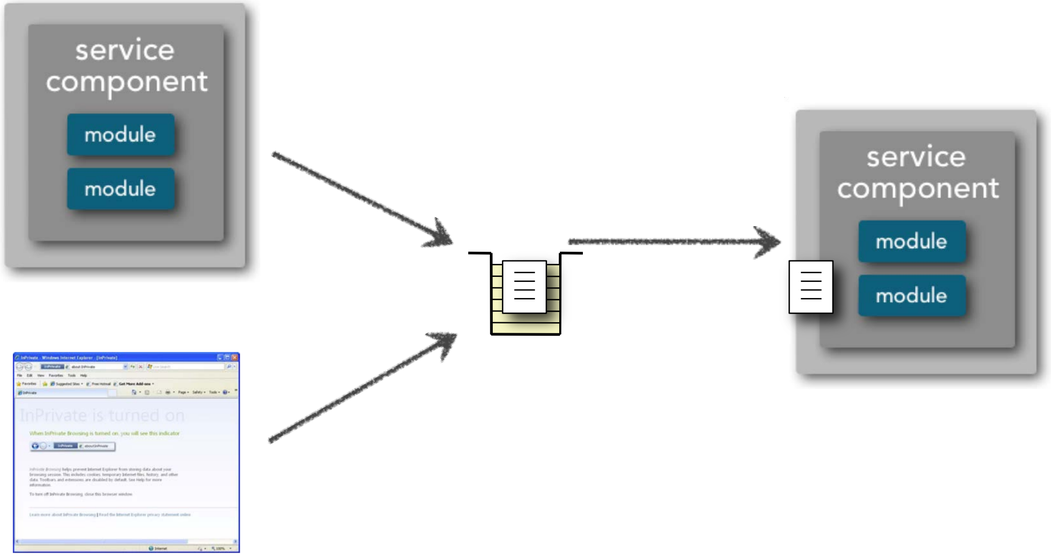


event-driven microservices



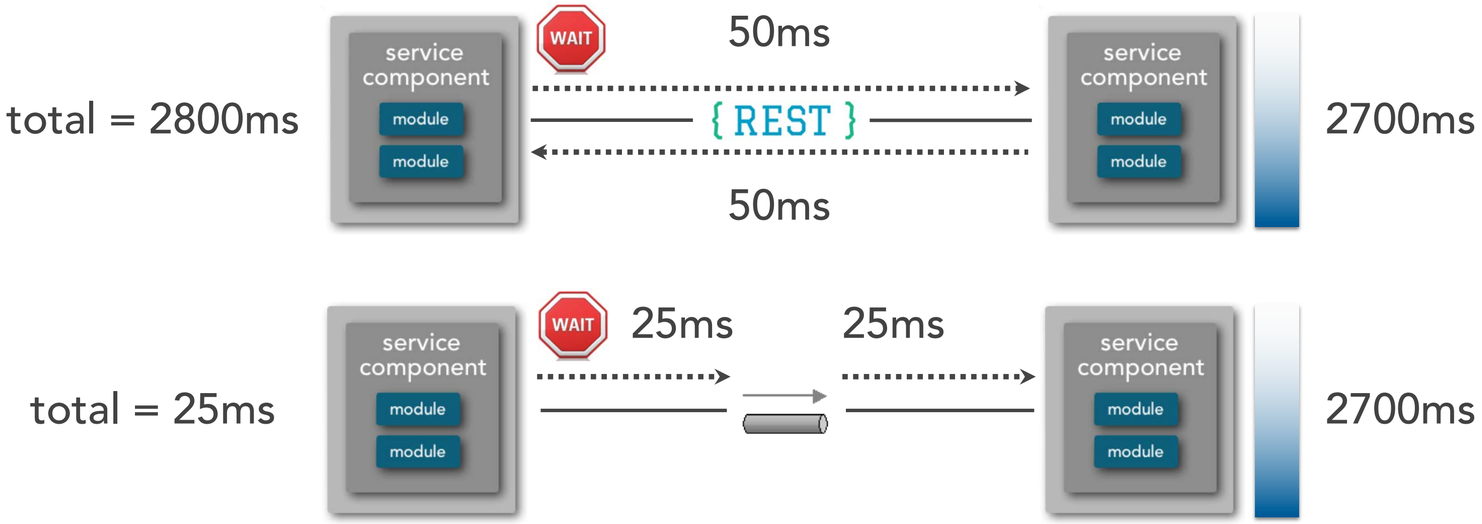
event-driven microservices

asynchronous capabilities



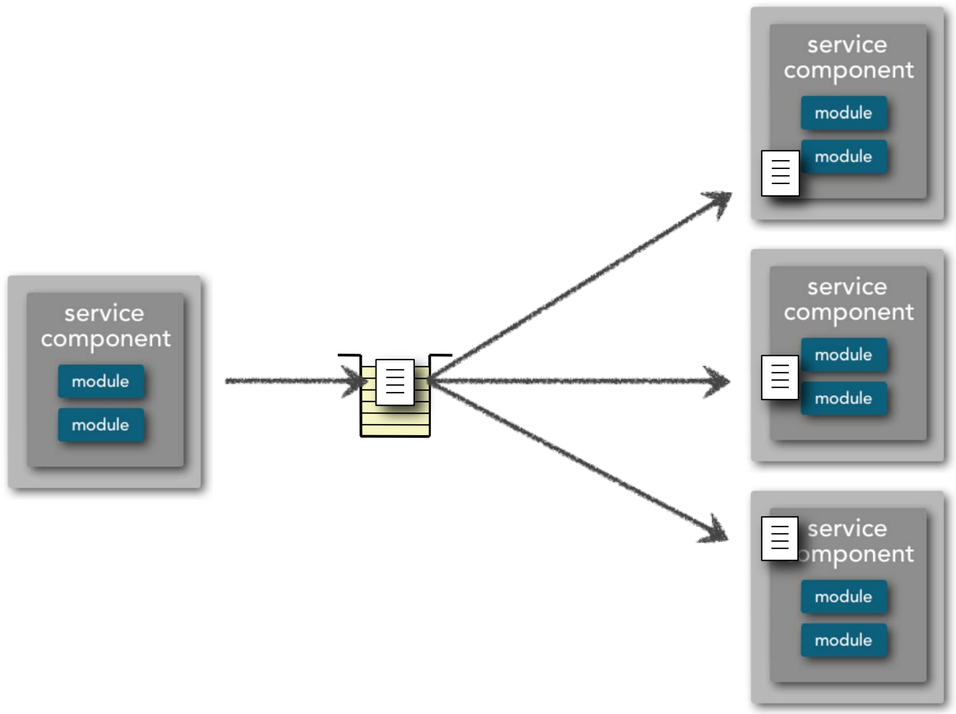
event-driven microservices

asynchronous capabilities



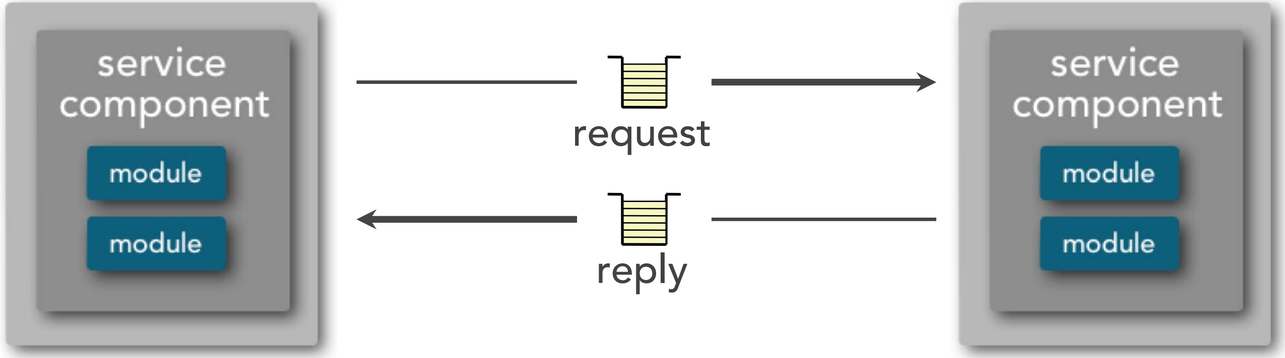
event-driven microservices

broadcast capabilities



event-driven microservices

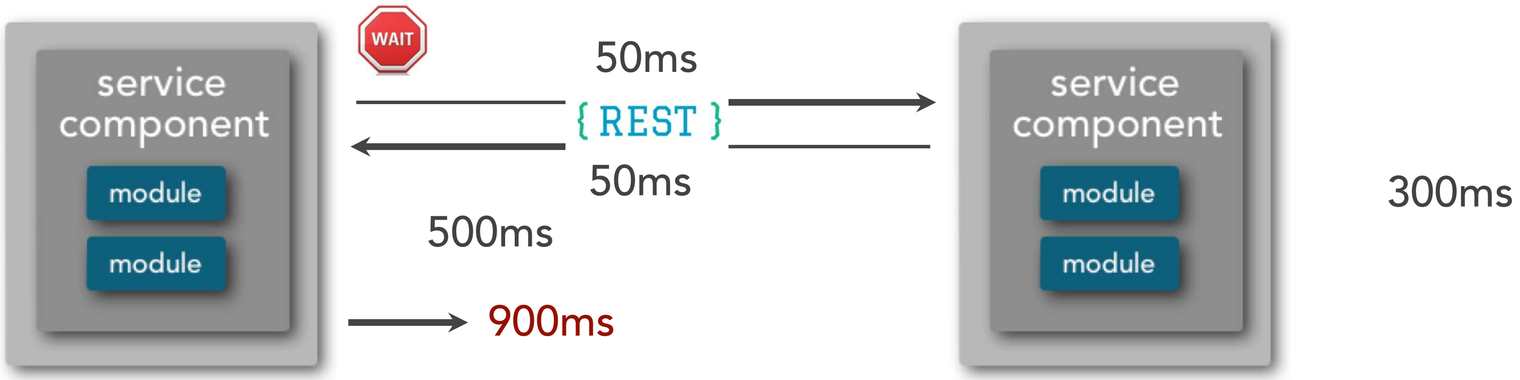
request/reply

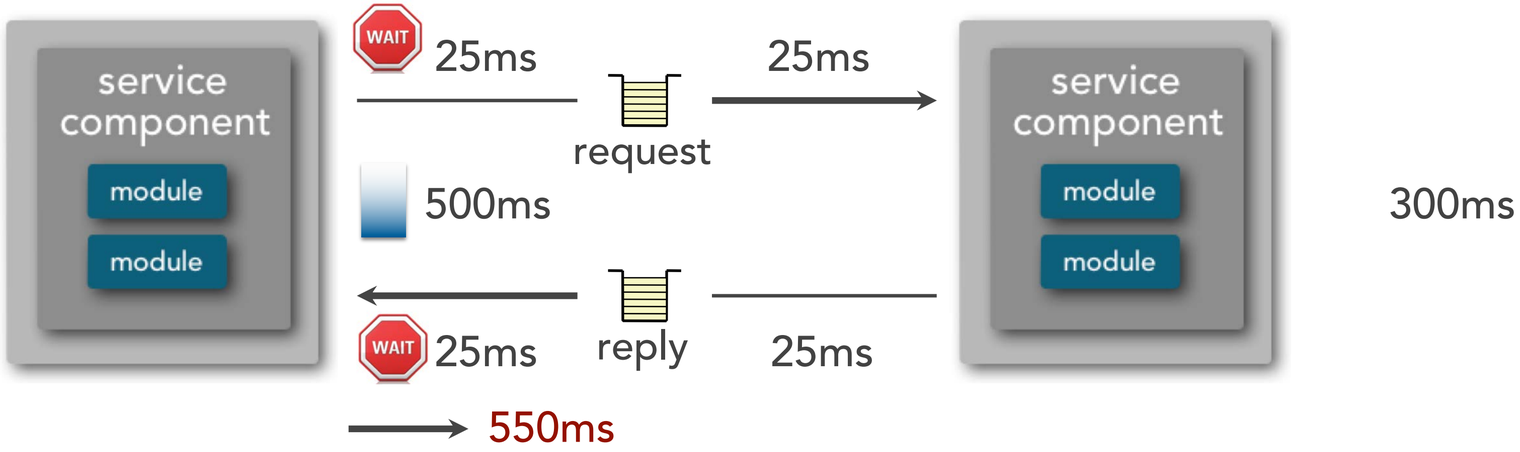


https://www.developertoarchitect.com/lessons/lesson1.html

event-driven microservices

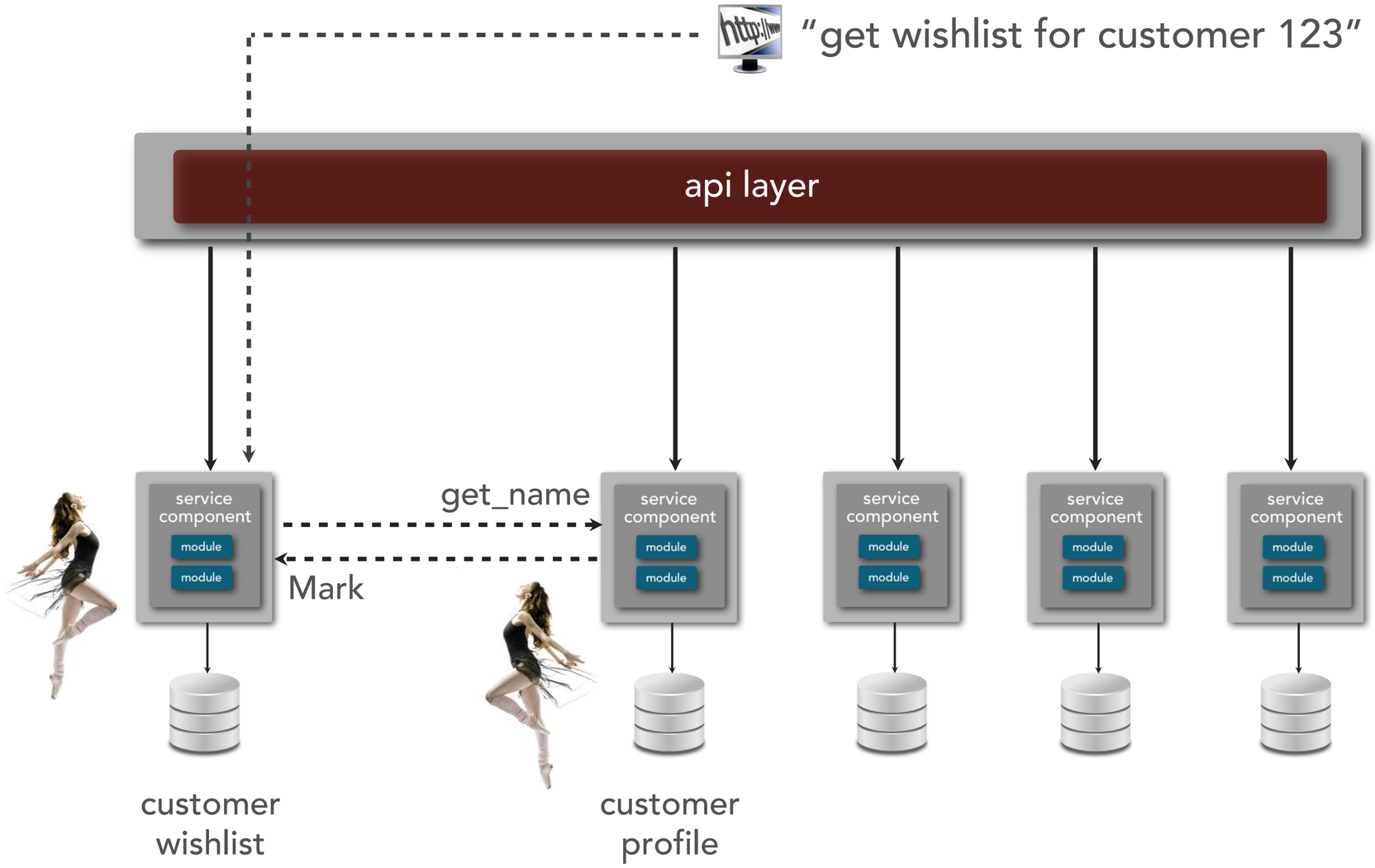
request/reply



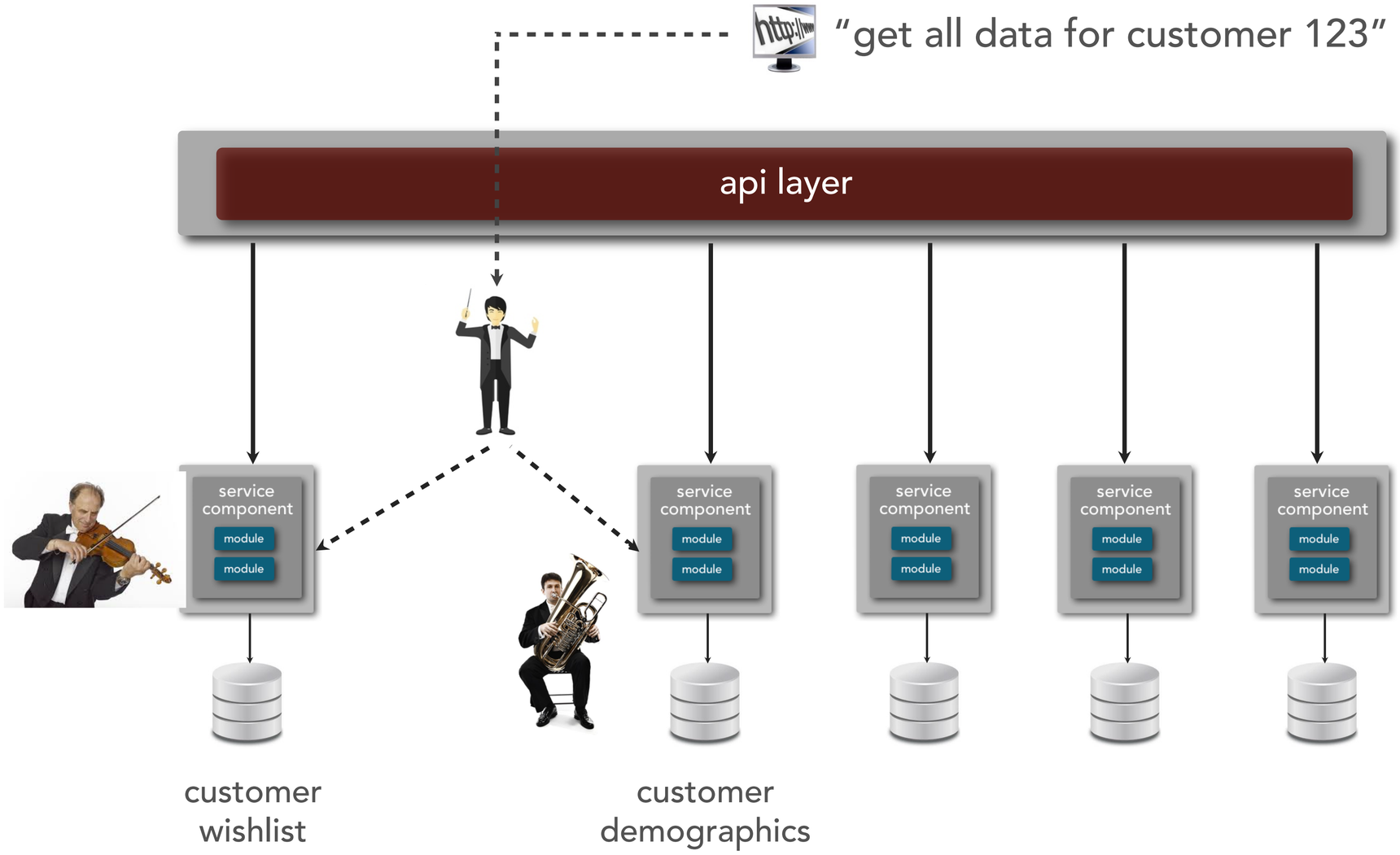


Orchestration vs. Choreography

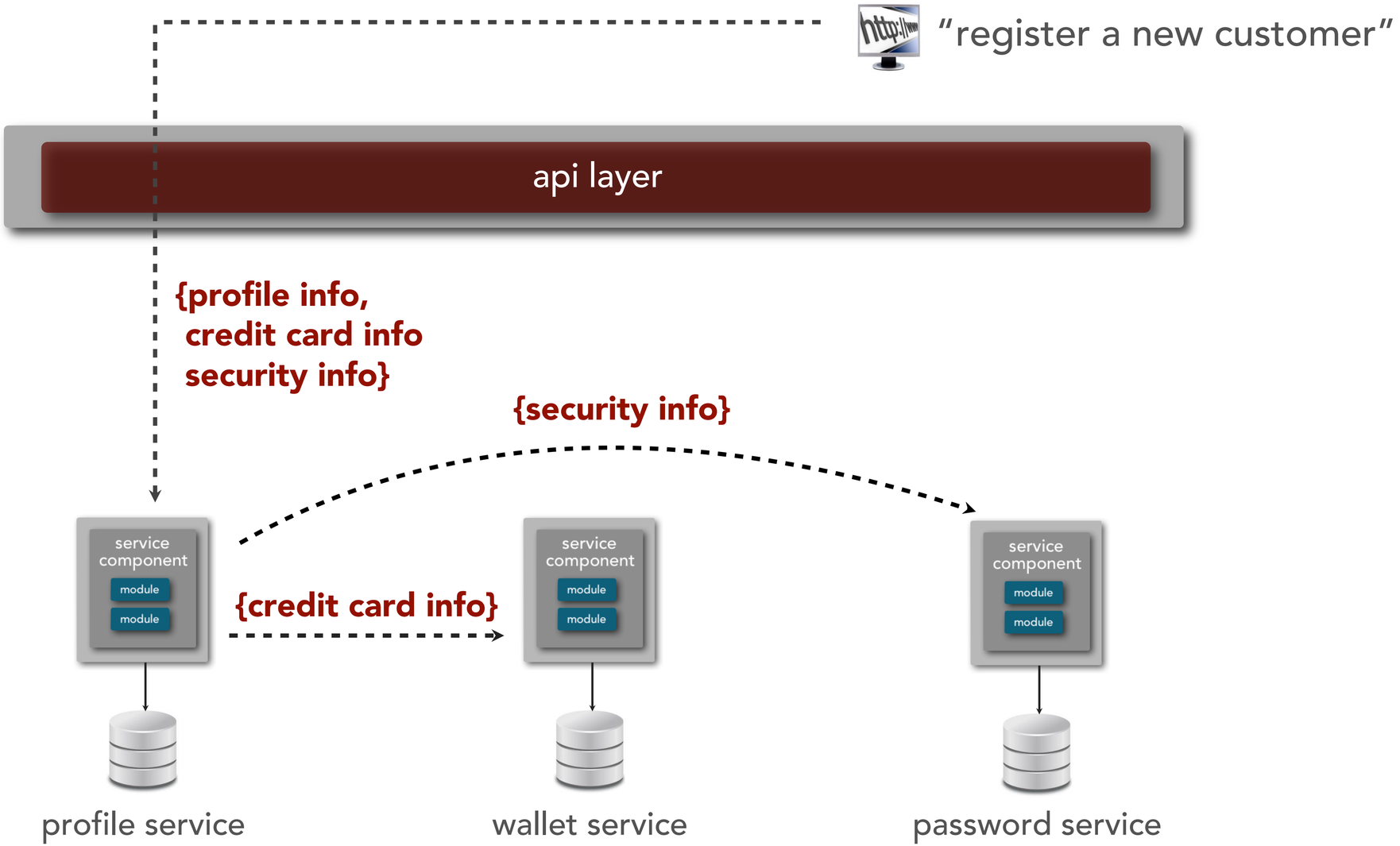
orchestration vs. choreography



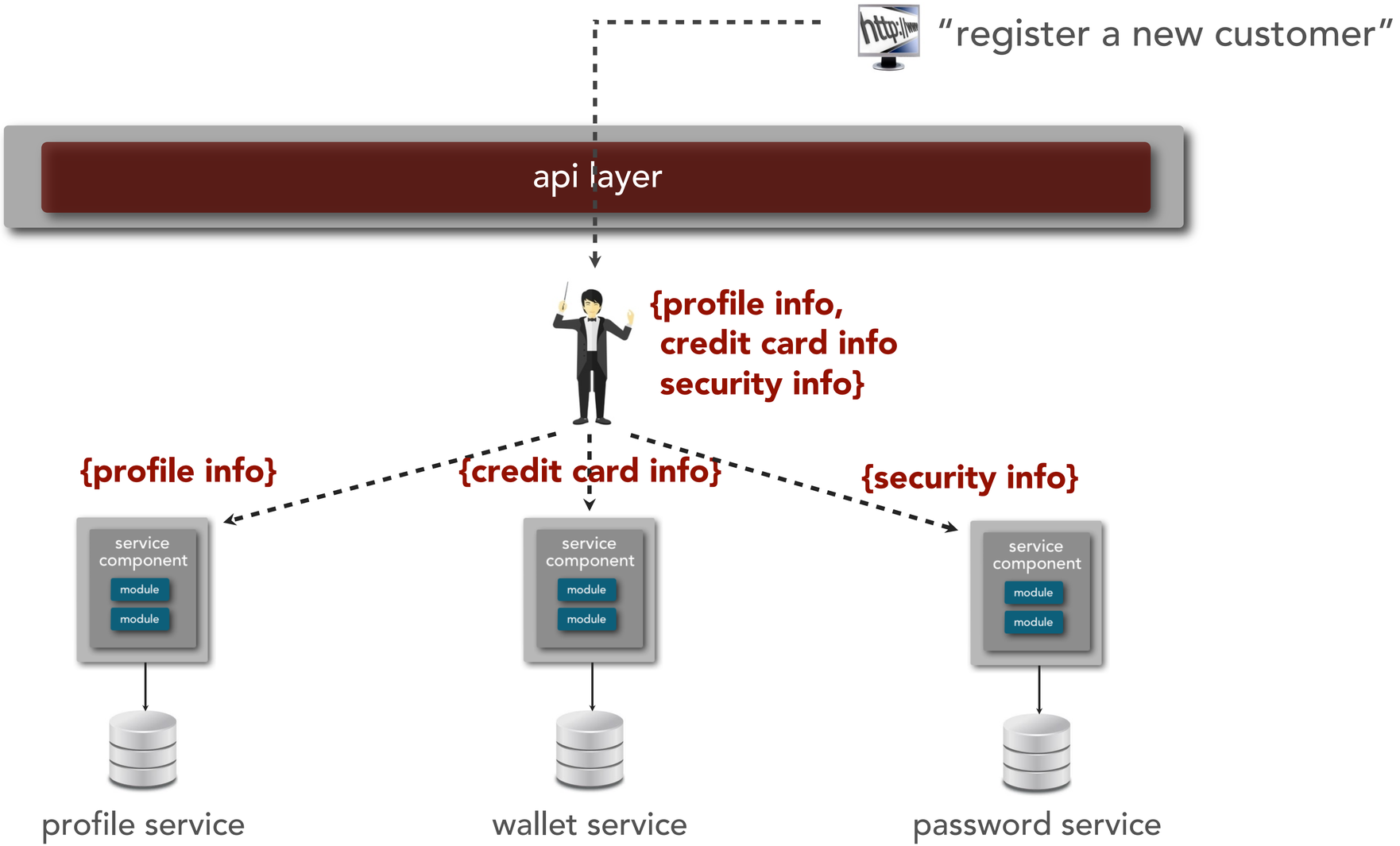
orchestration vs. choreography



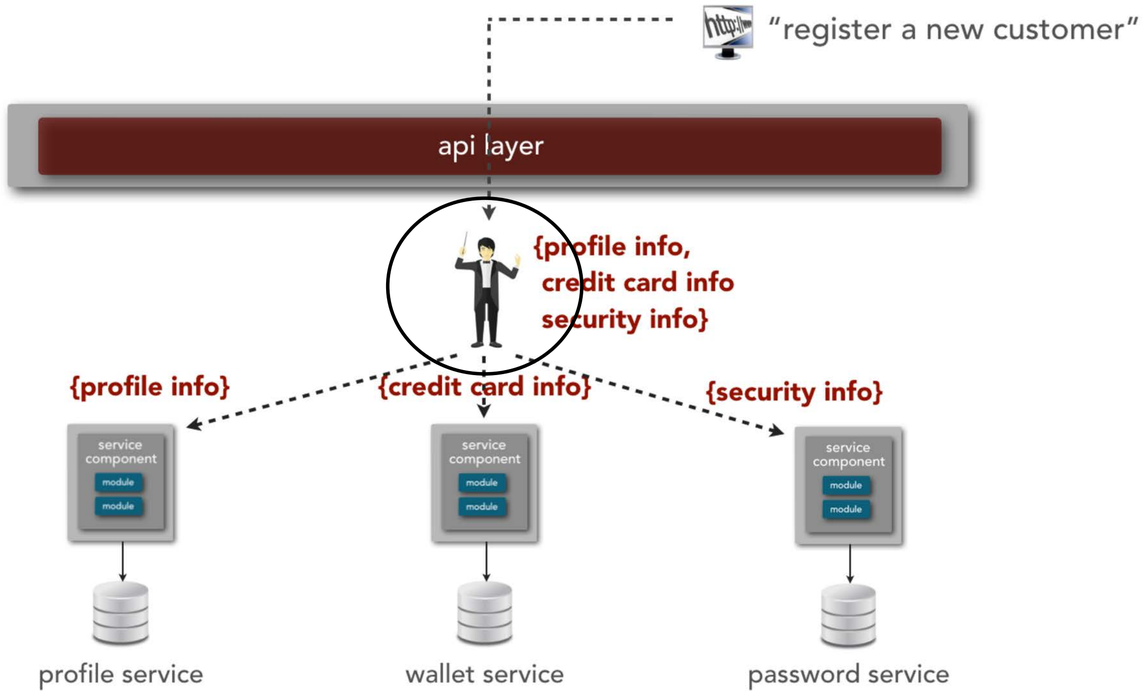
orchestration vs. choreography

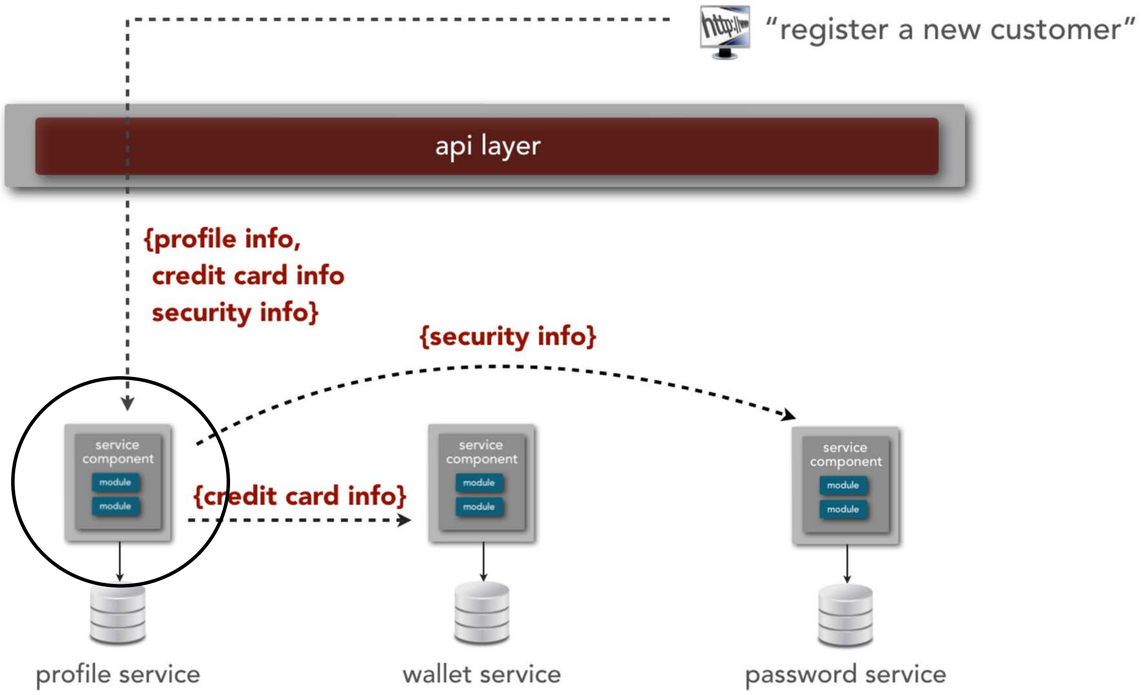


orchestration vs. choreography



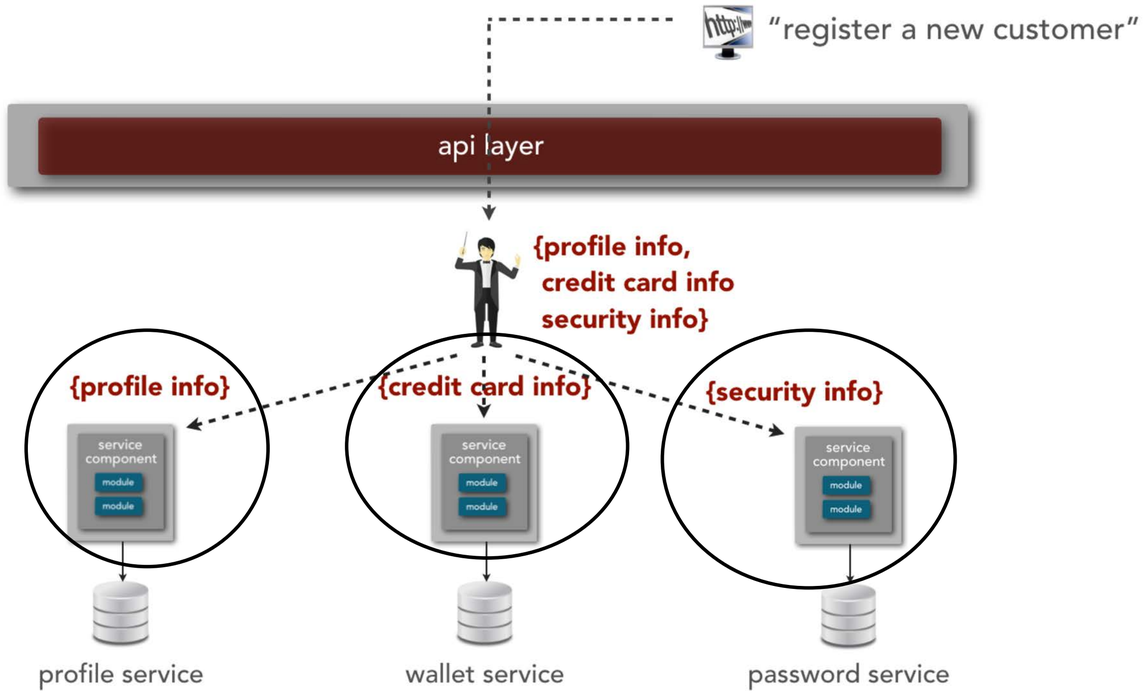
orchestration vs. choreography

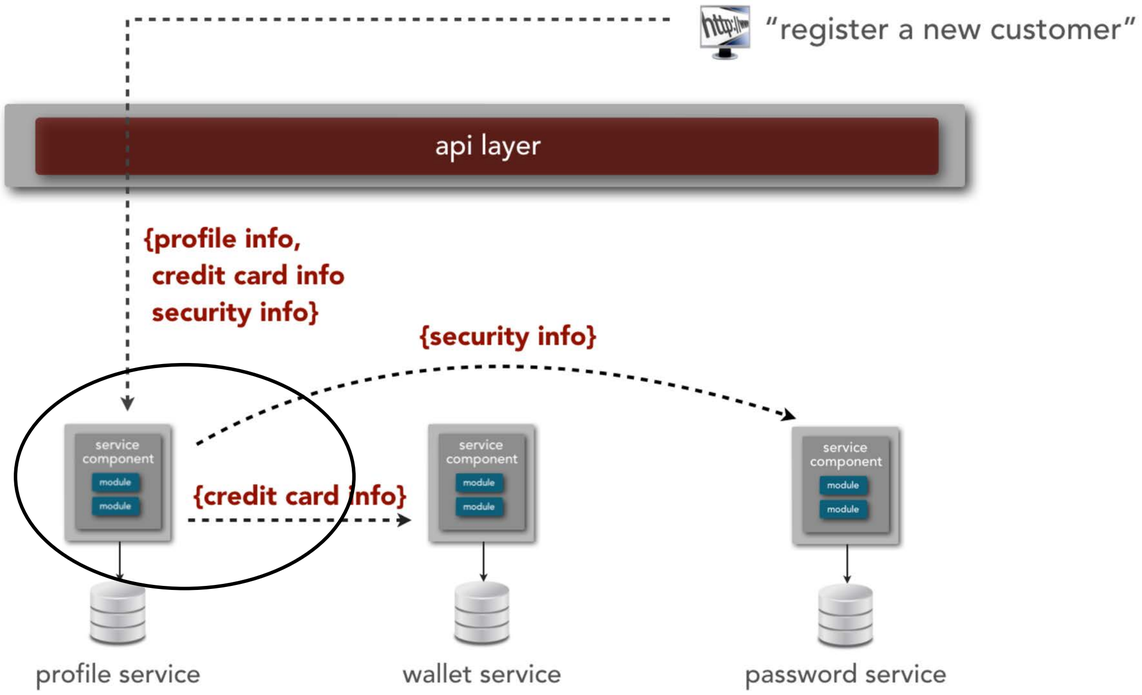




workflow ownership

orchestration vs. choreography



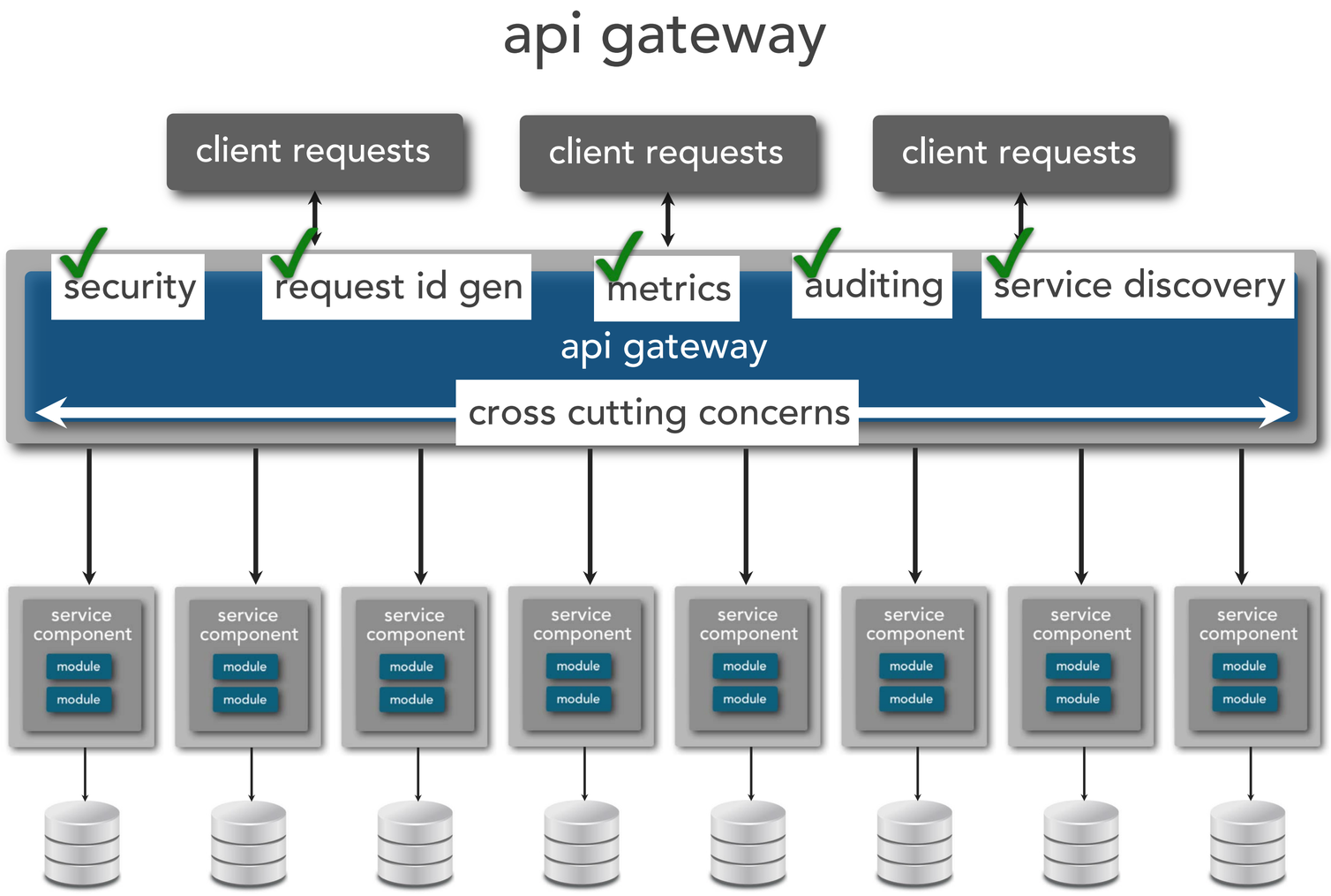


workflow ownership

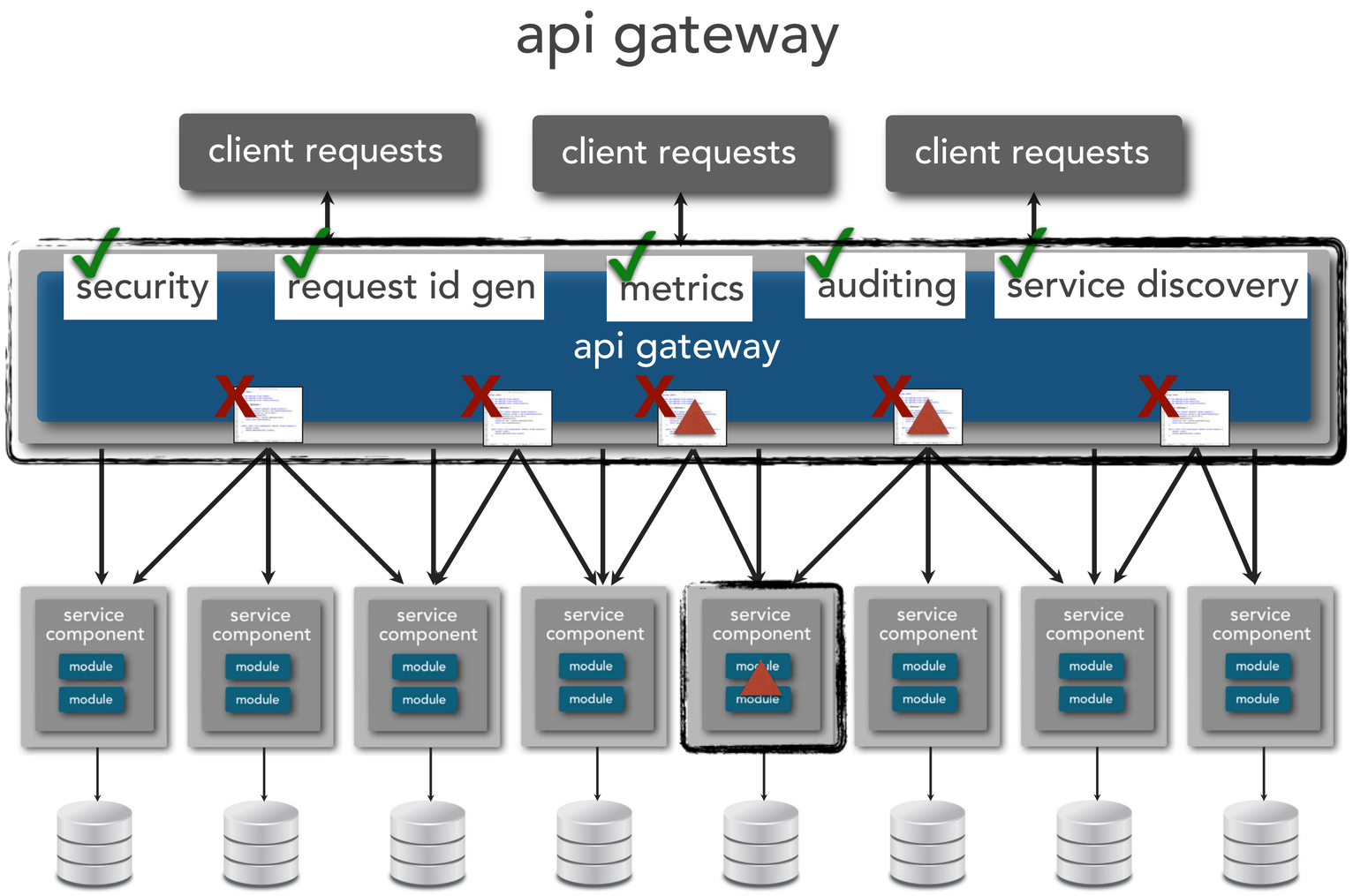
data access and ownership

Orchestration Pattern

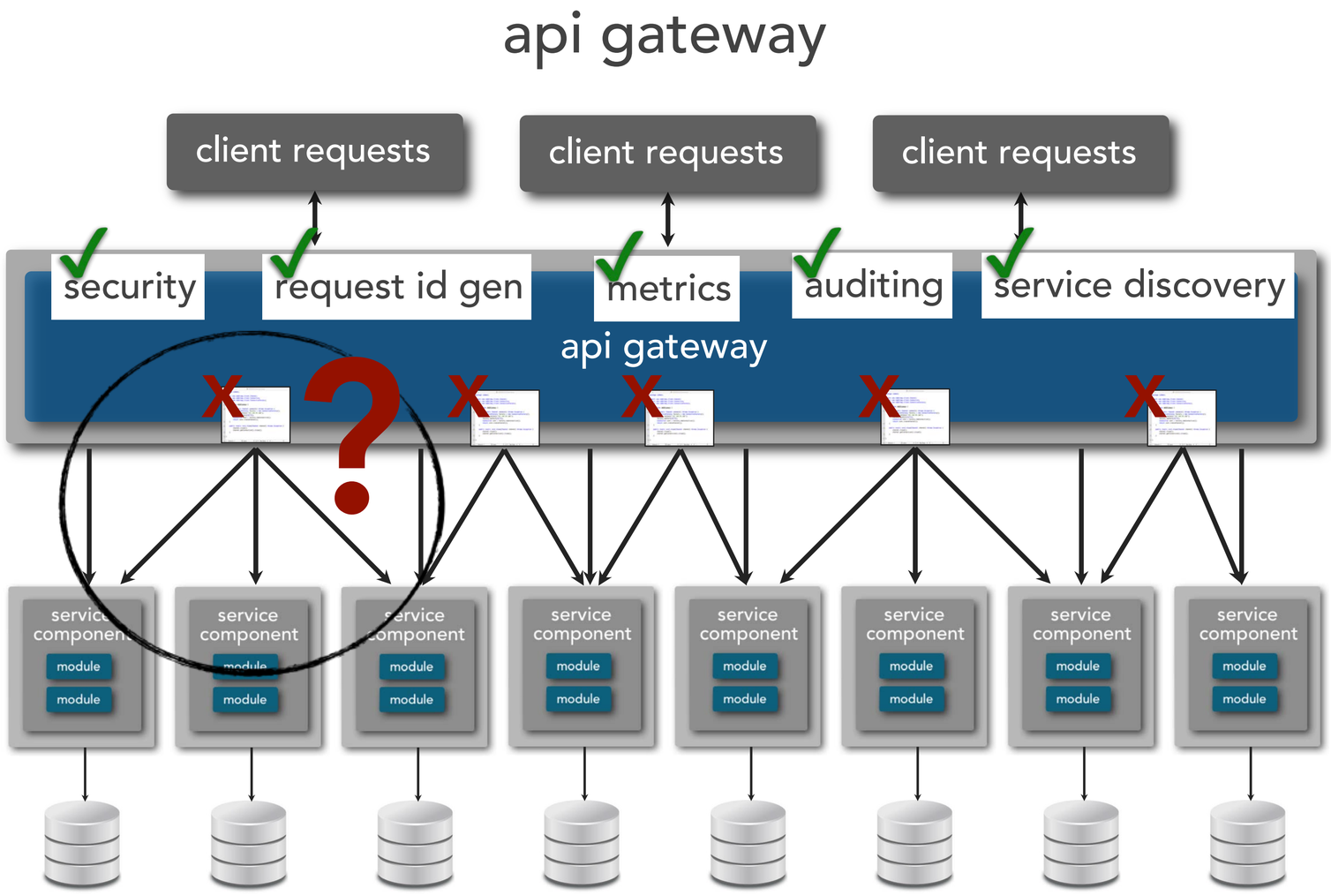
microservices orchestration pattern



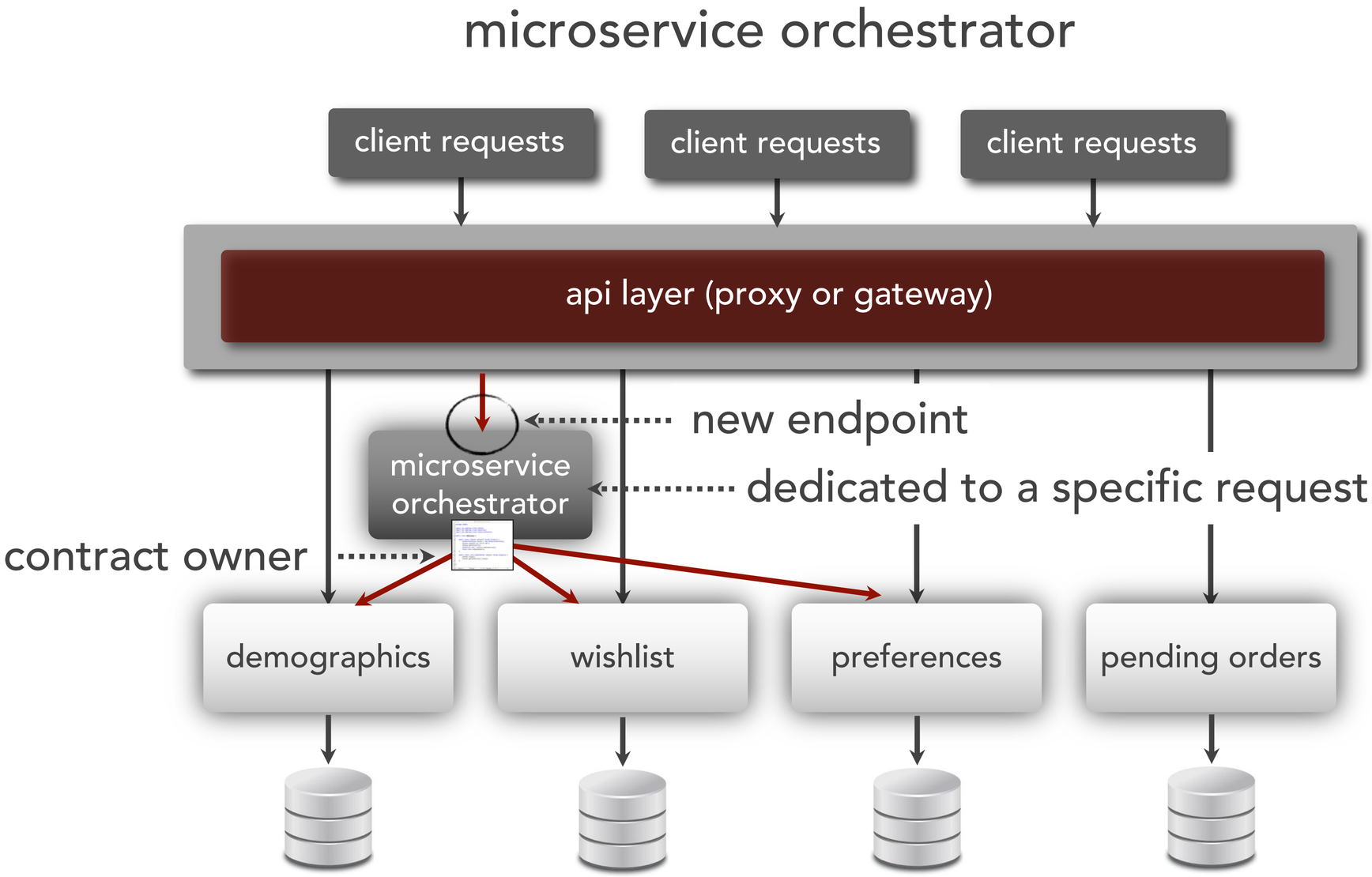
microservices orchestration pattern



microservices orchestration pattern

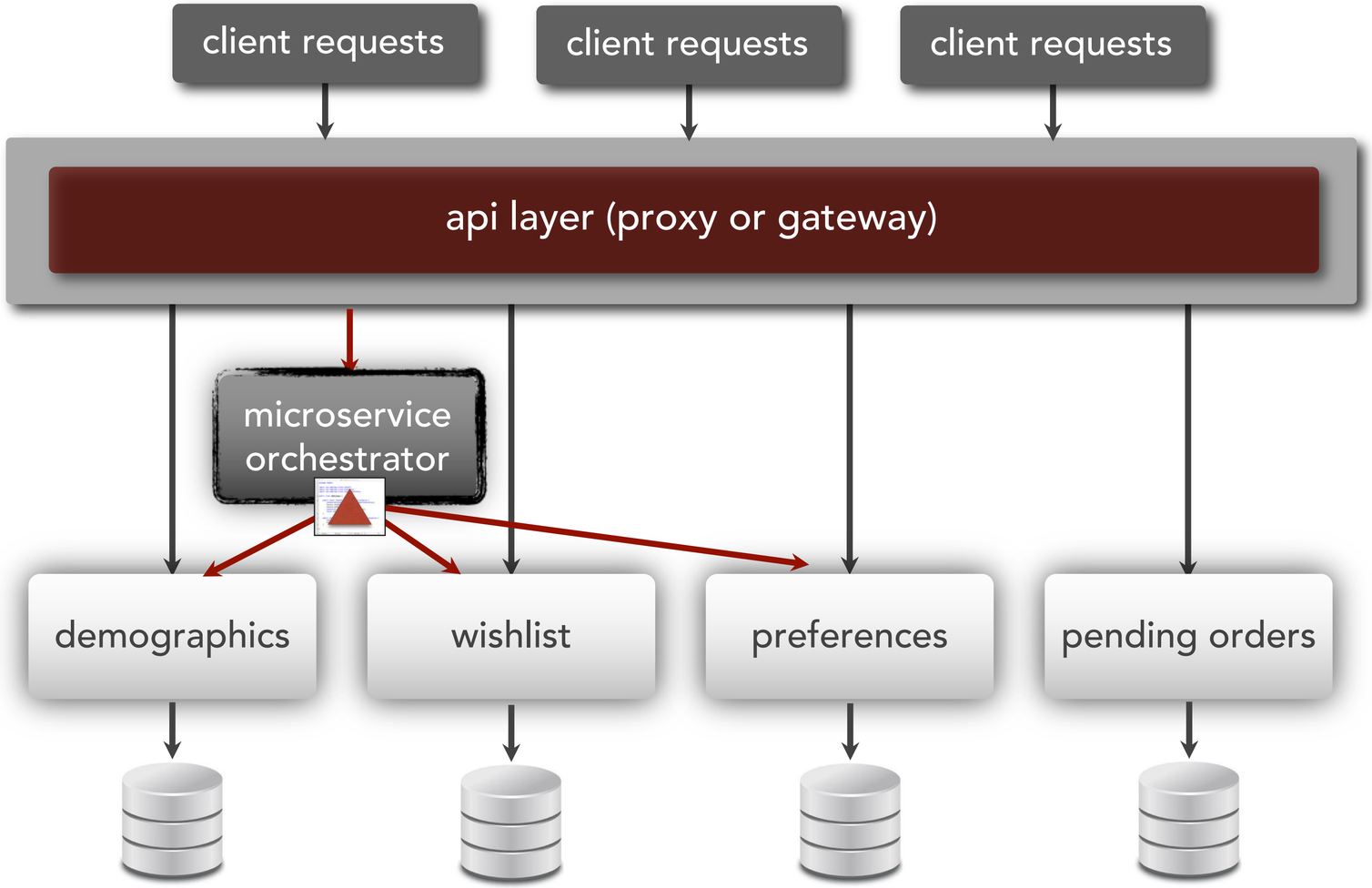


microservices orchestration pattern



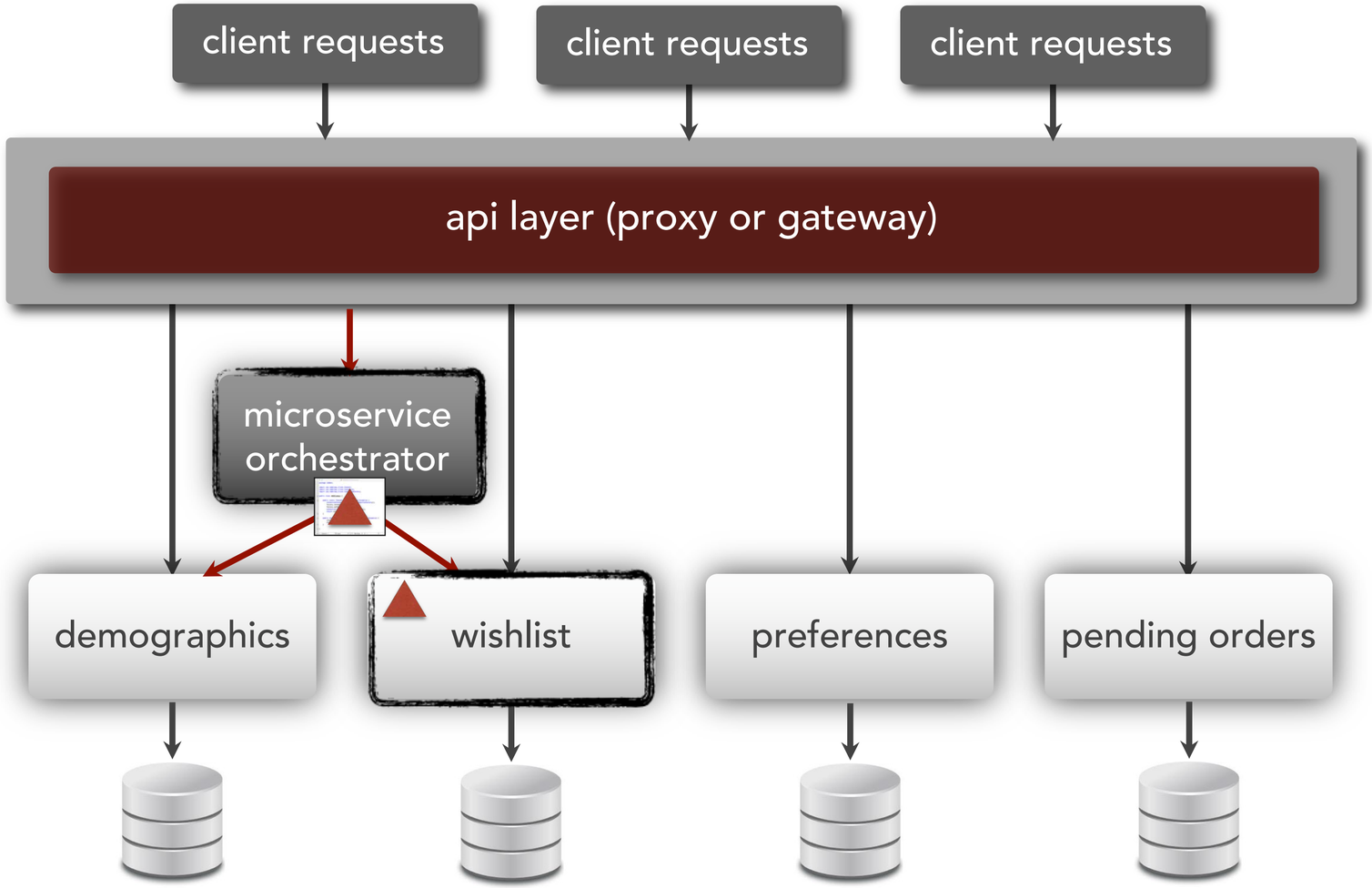
microservices orchestration pattern

microservice orchestrator



microservices orchestration pattern

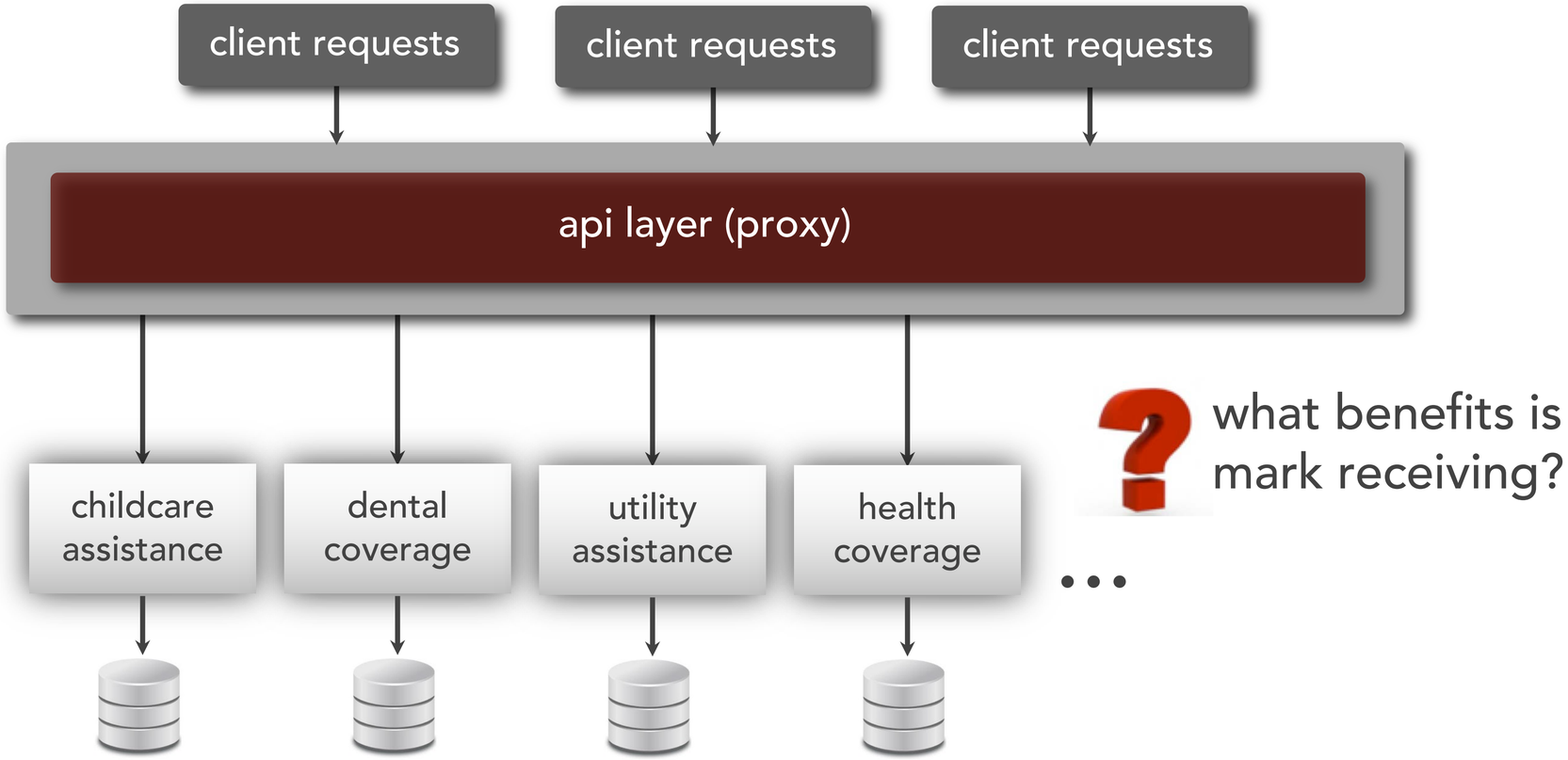
microservice orchestrator



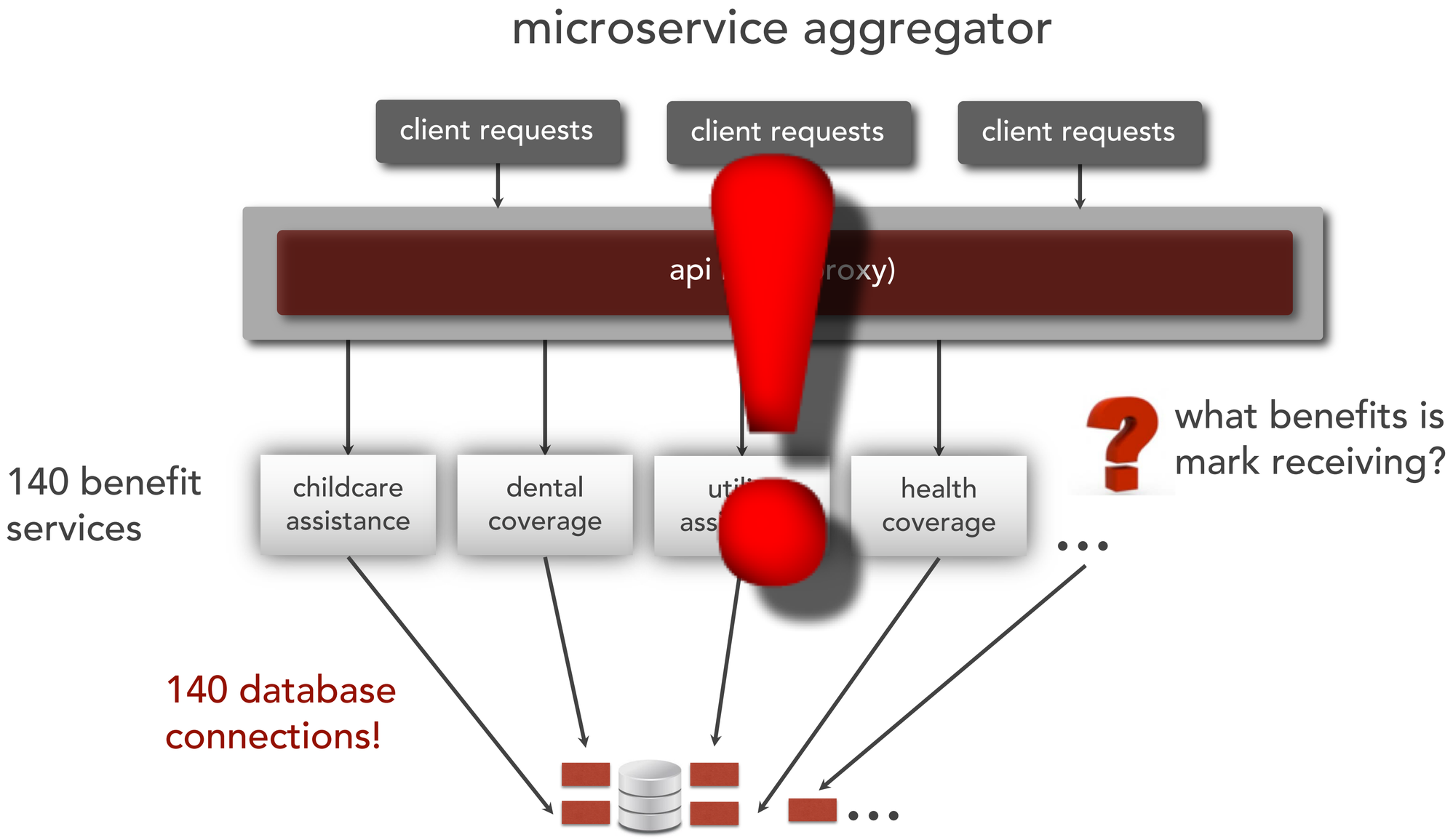
Aggregation Pattern

microservices aggregation pattern

microservice aggregator

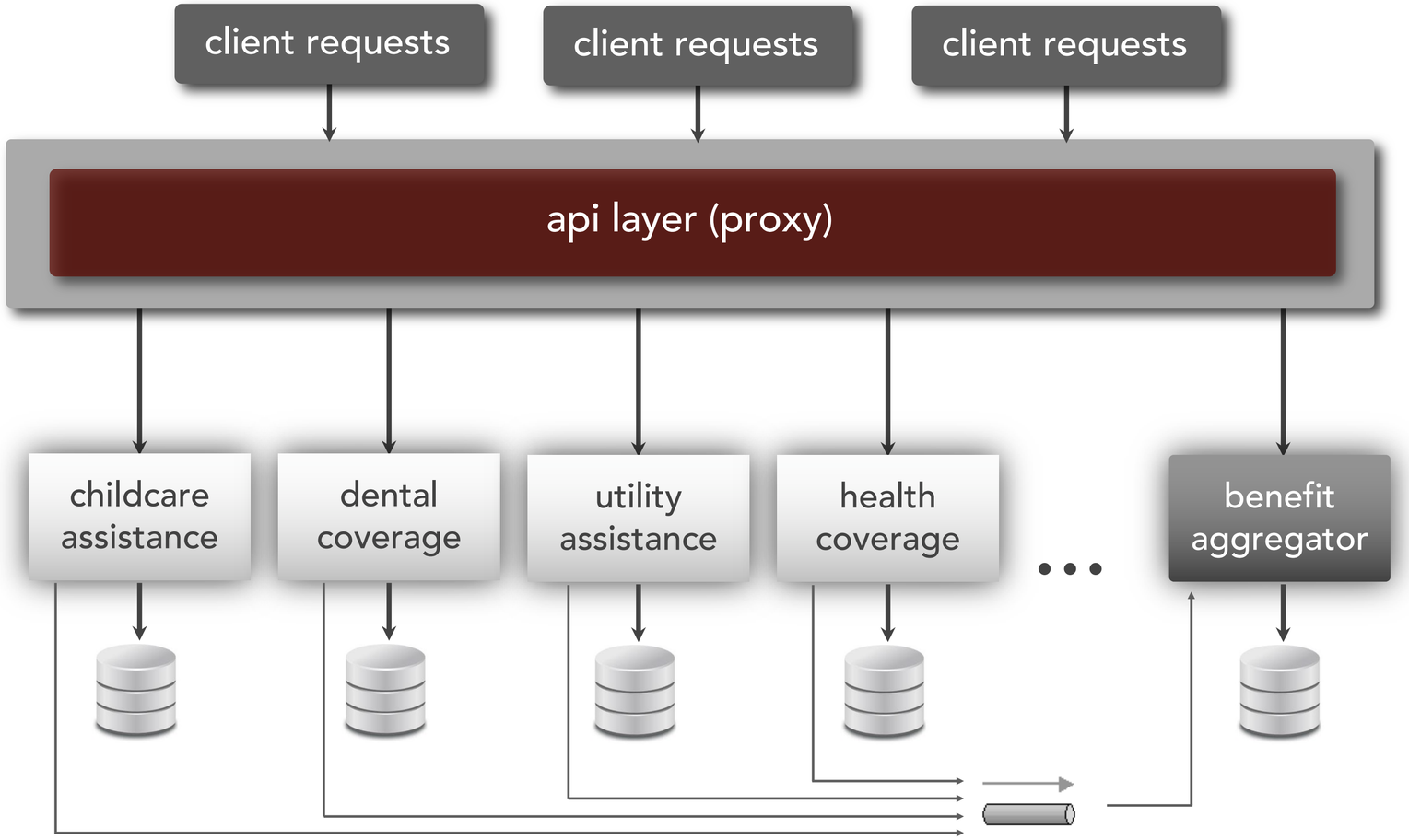


microservices aggregation pattern

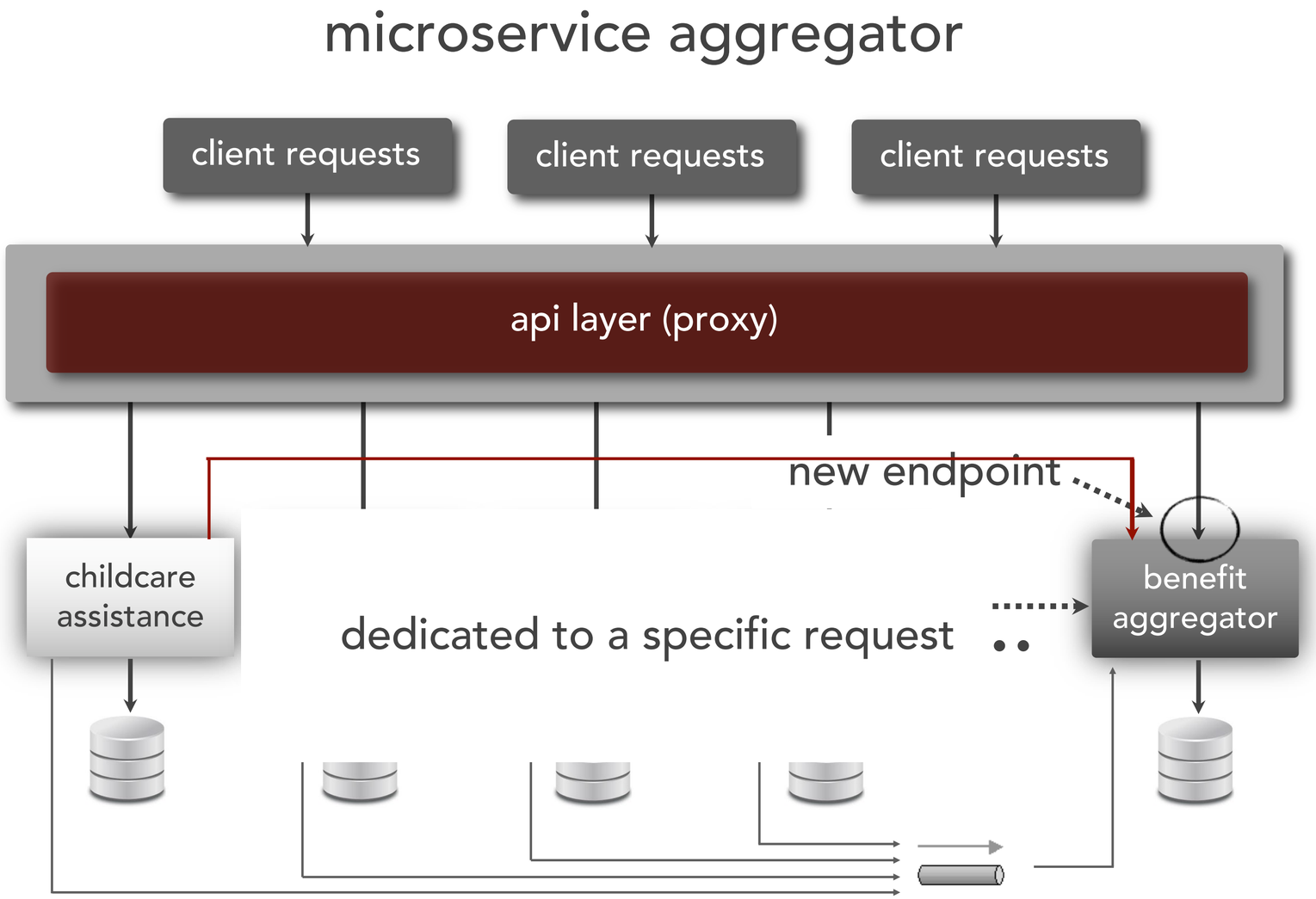


microservices aggregation pattern

microservice aggregator



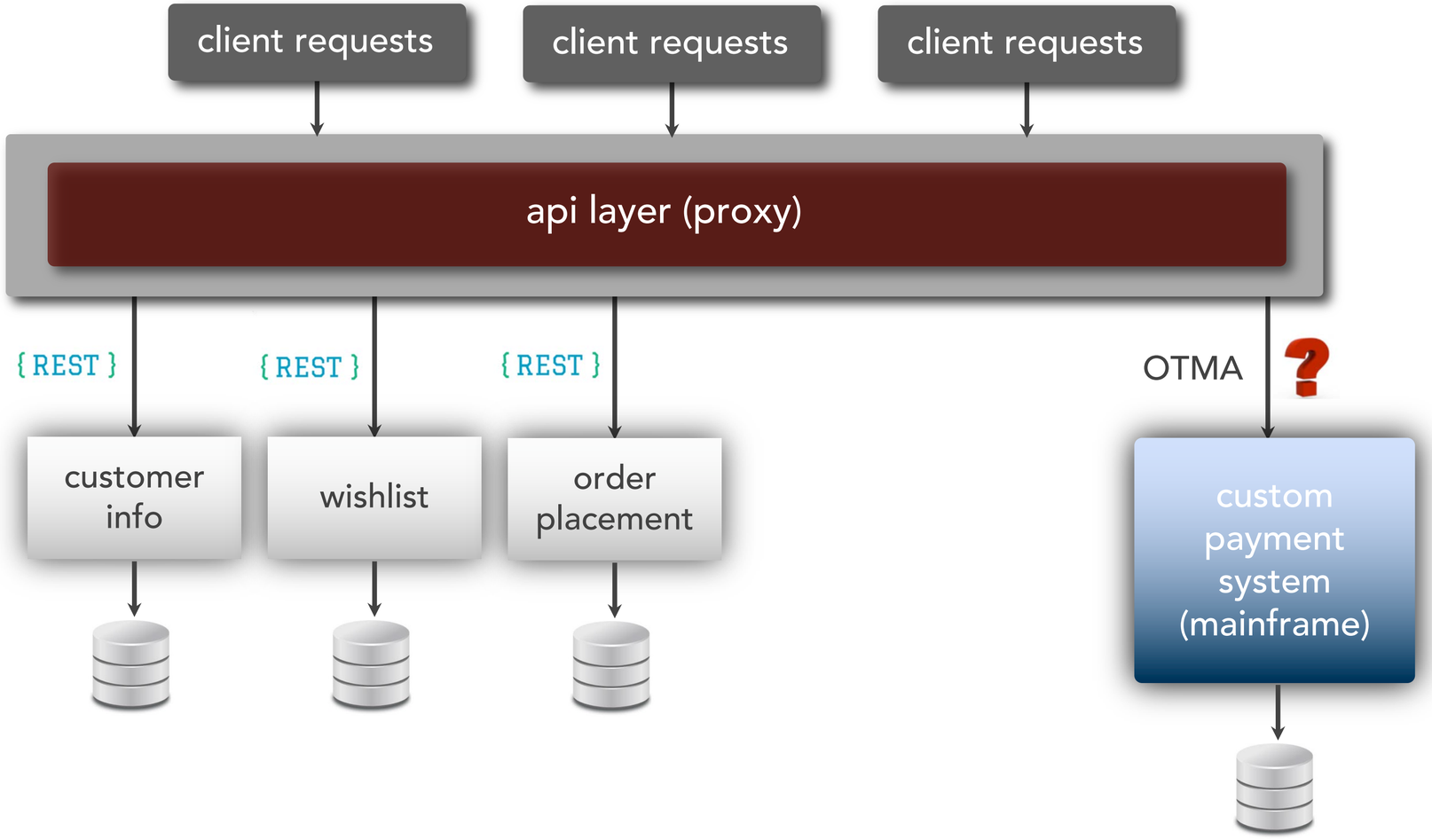
microservices aggregation pattern



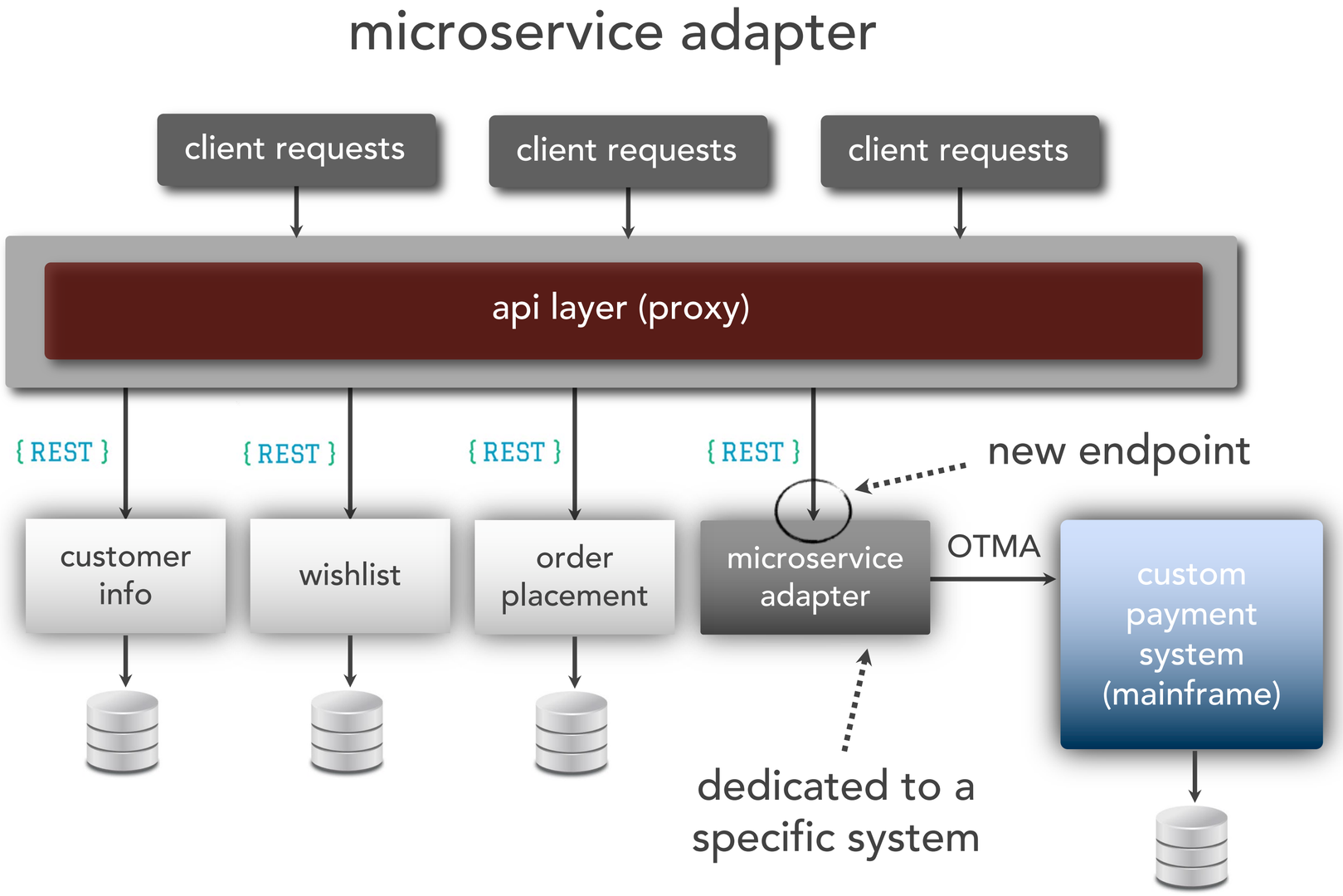
Adapter Pattern

microservices adapter pattern

microservice adapter

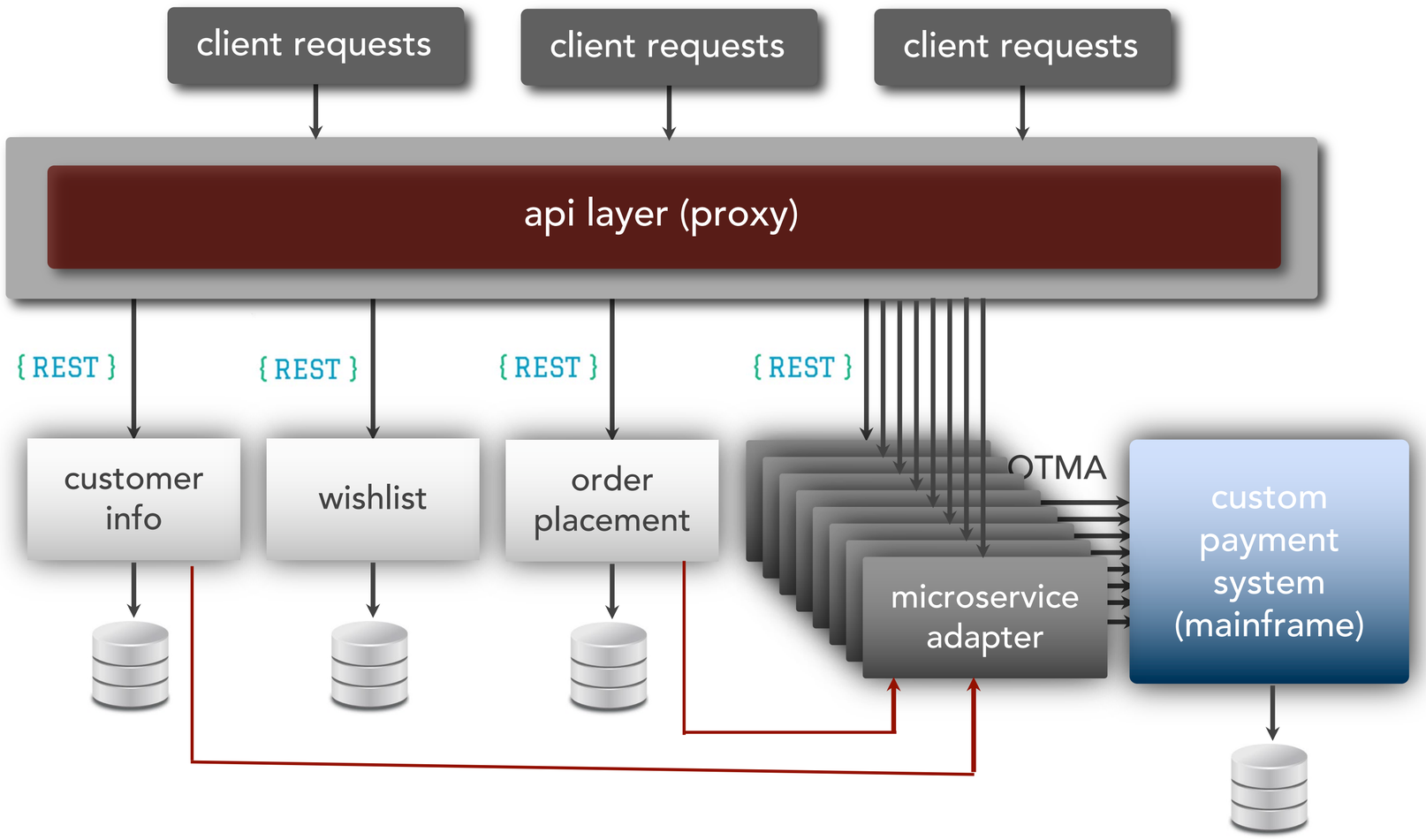


microservices adapter pattern



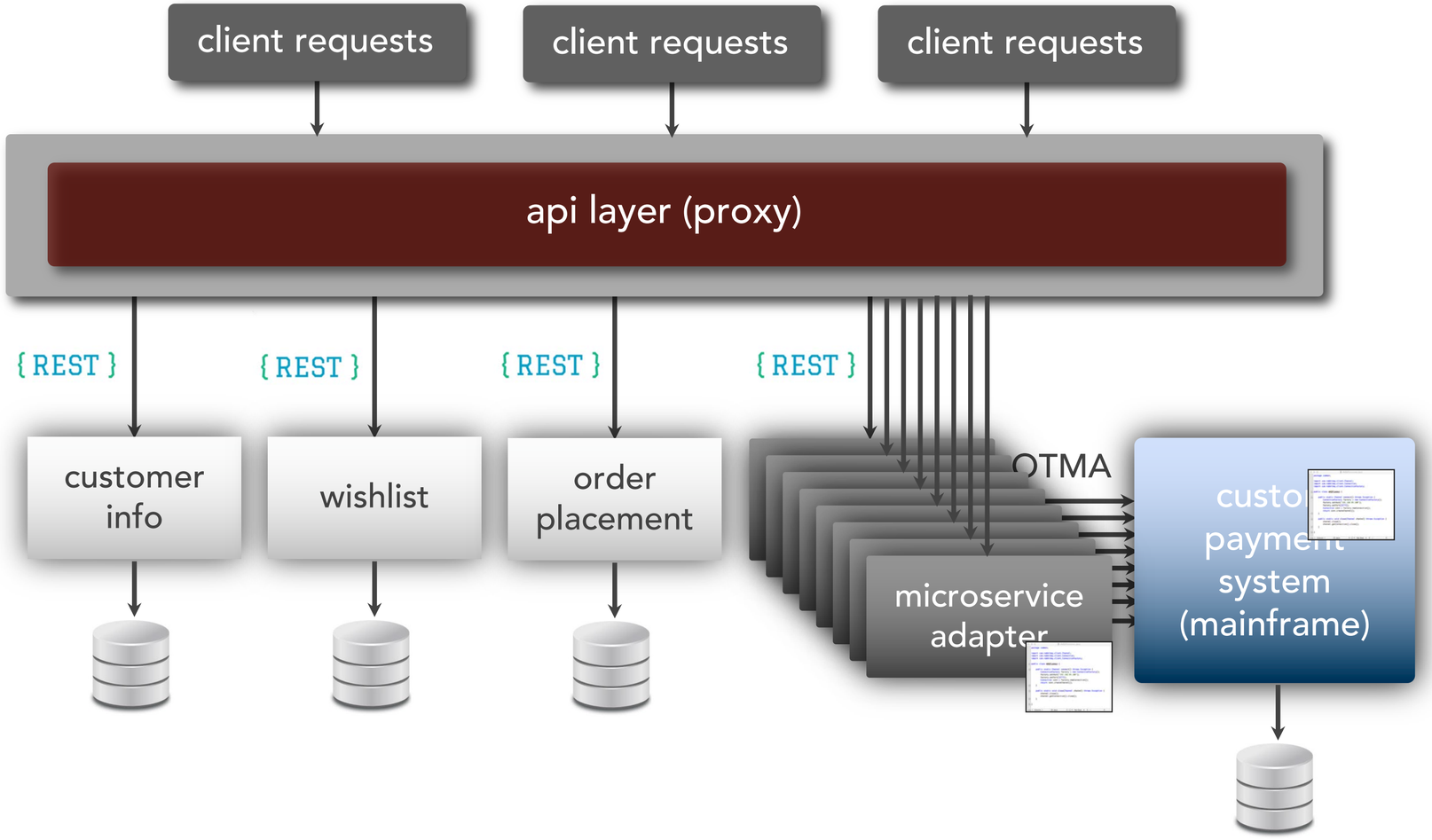
microservices adapter pattern

microservice adapter



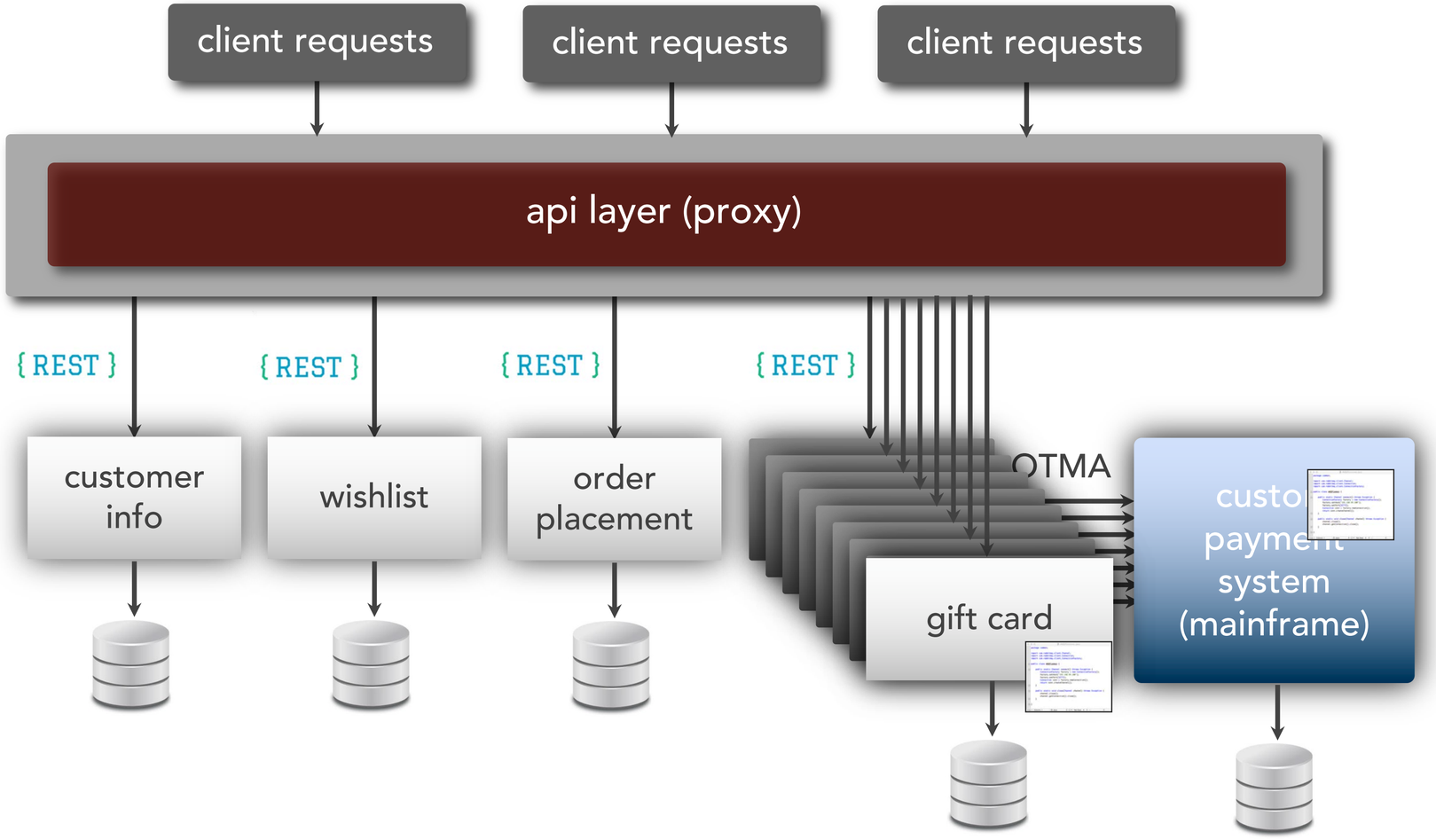
microservices adapter pattern

microservice adapter



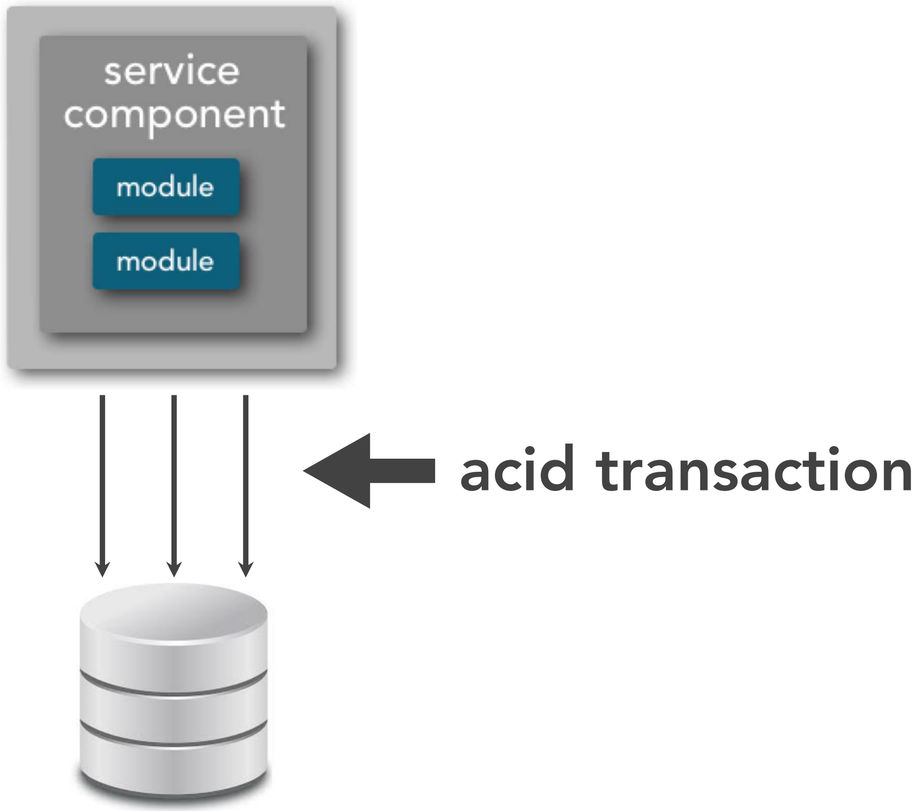
microservices adapter pattern

microservice adapter

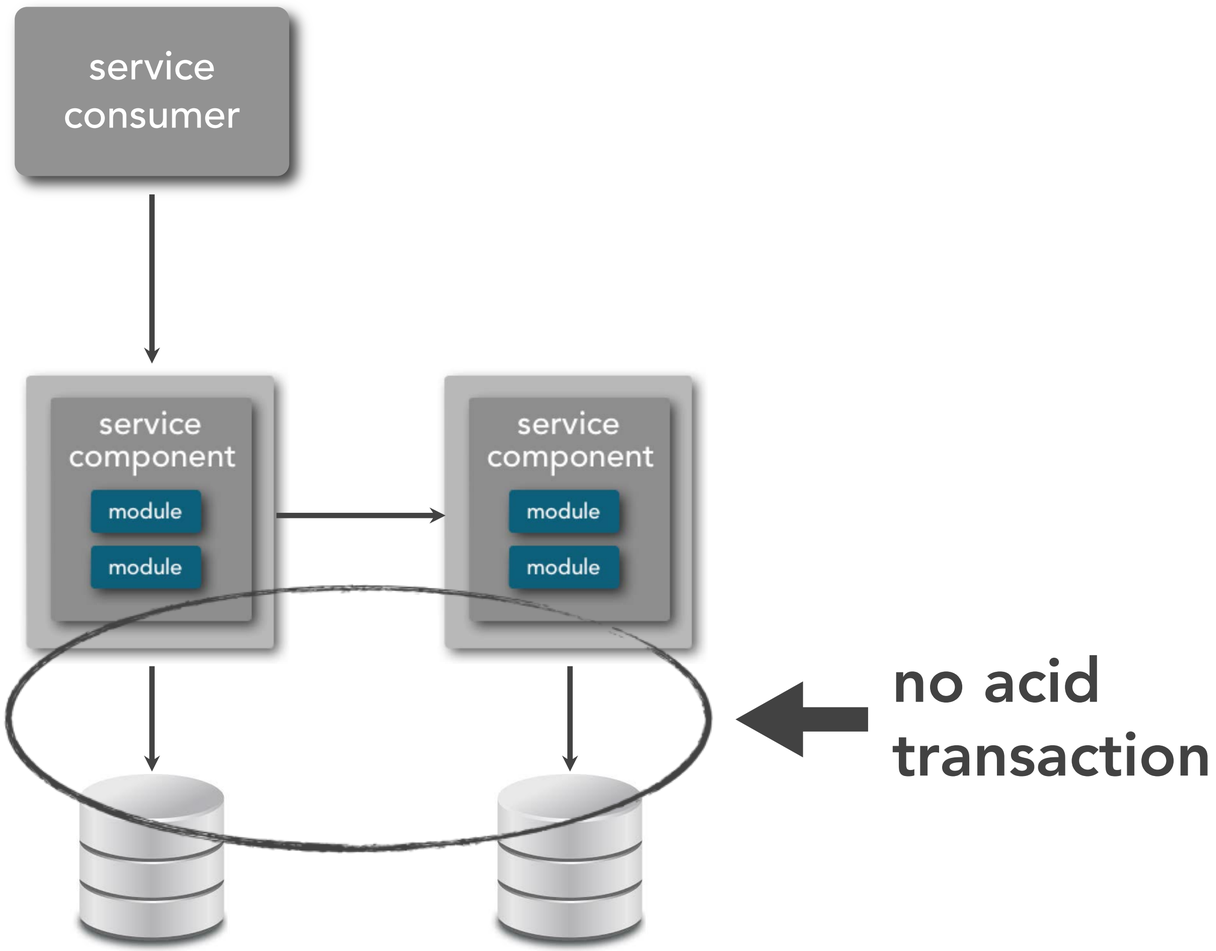


Distributed Transactions

distributed transactions

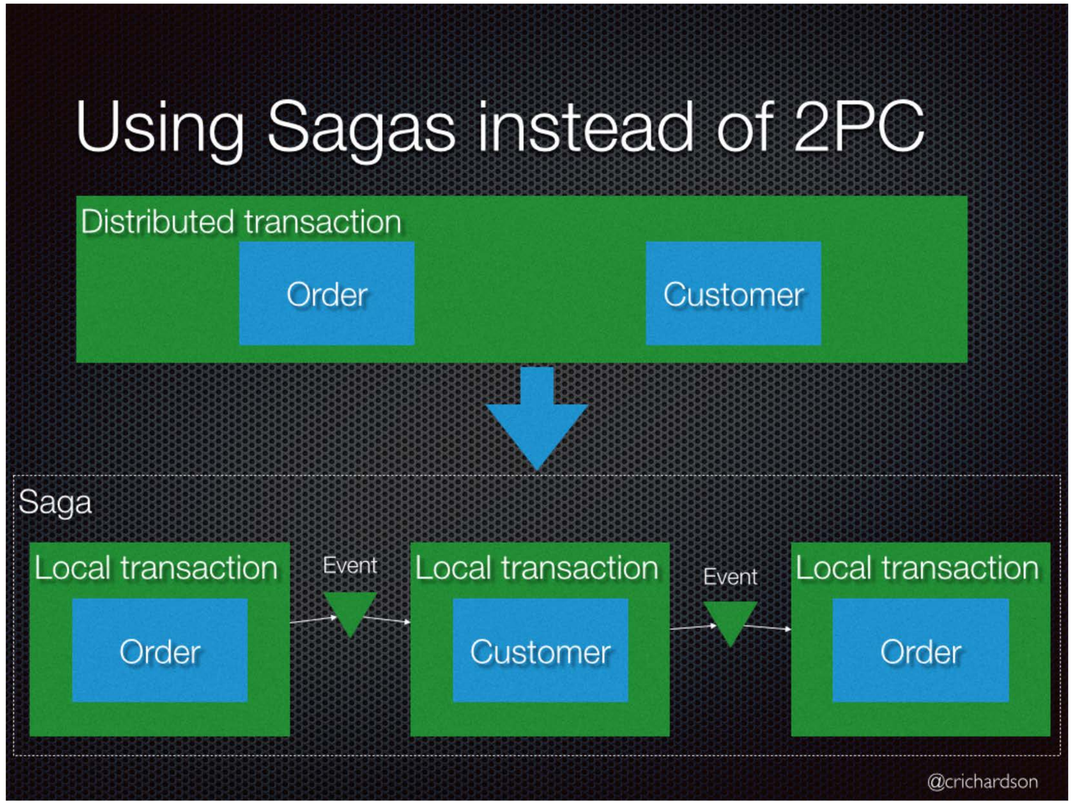


distributed transactions

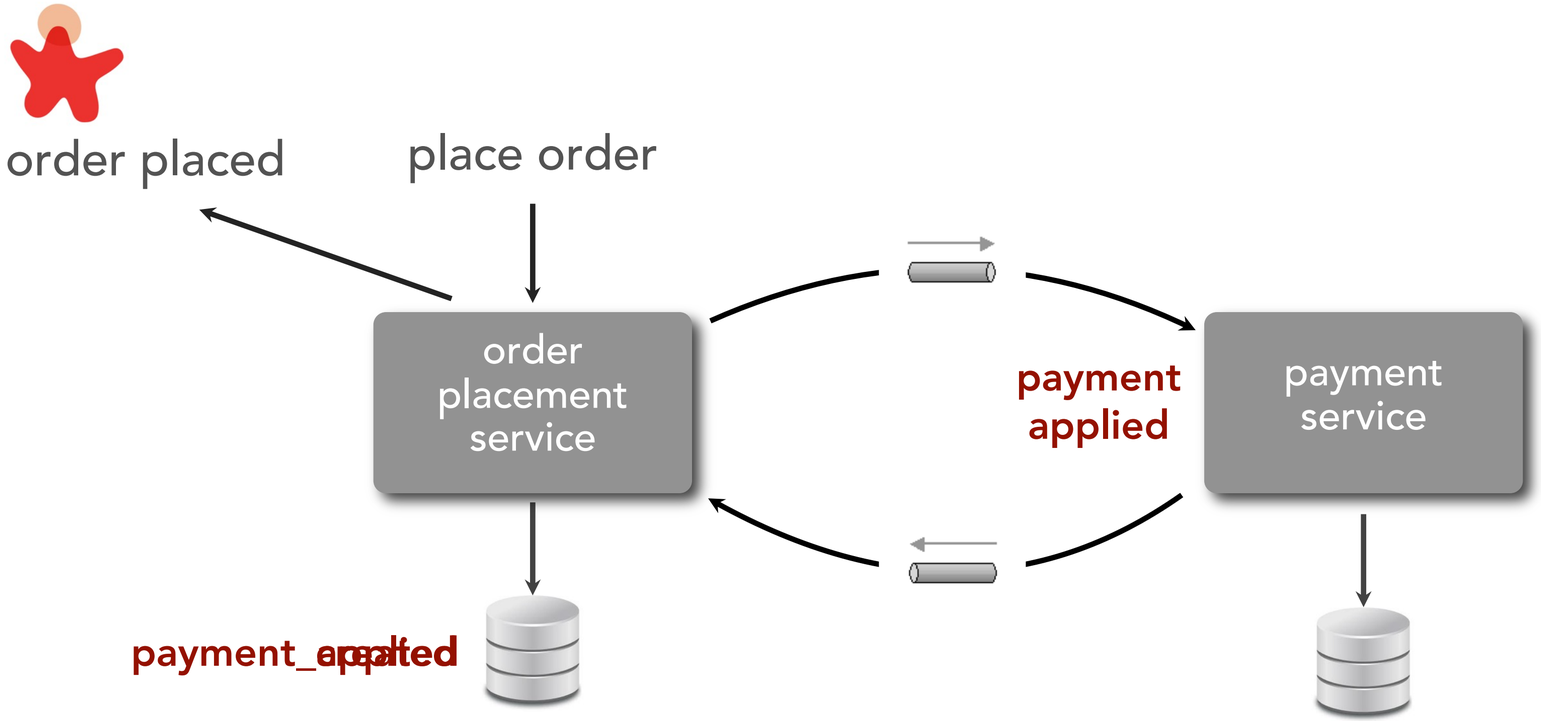


distributed transactions

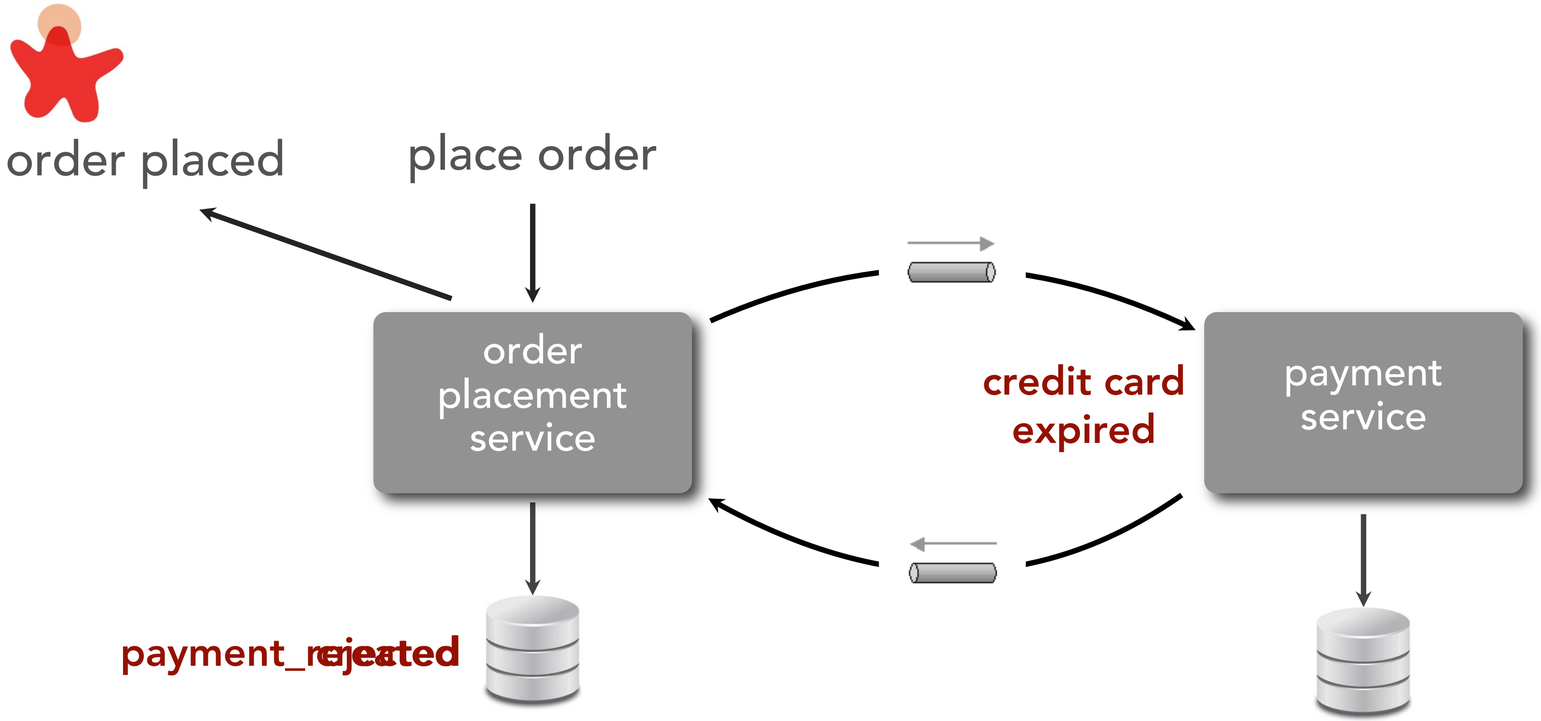
https://microservices.io/patterns/data/saga.html



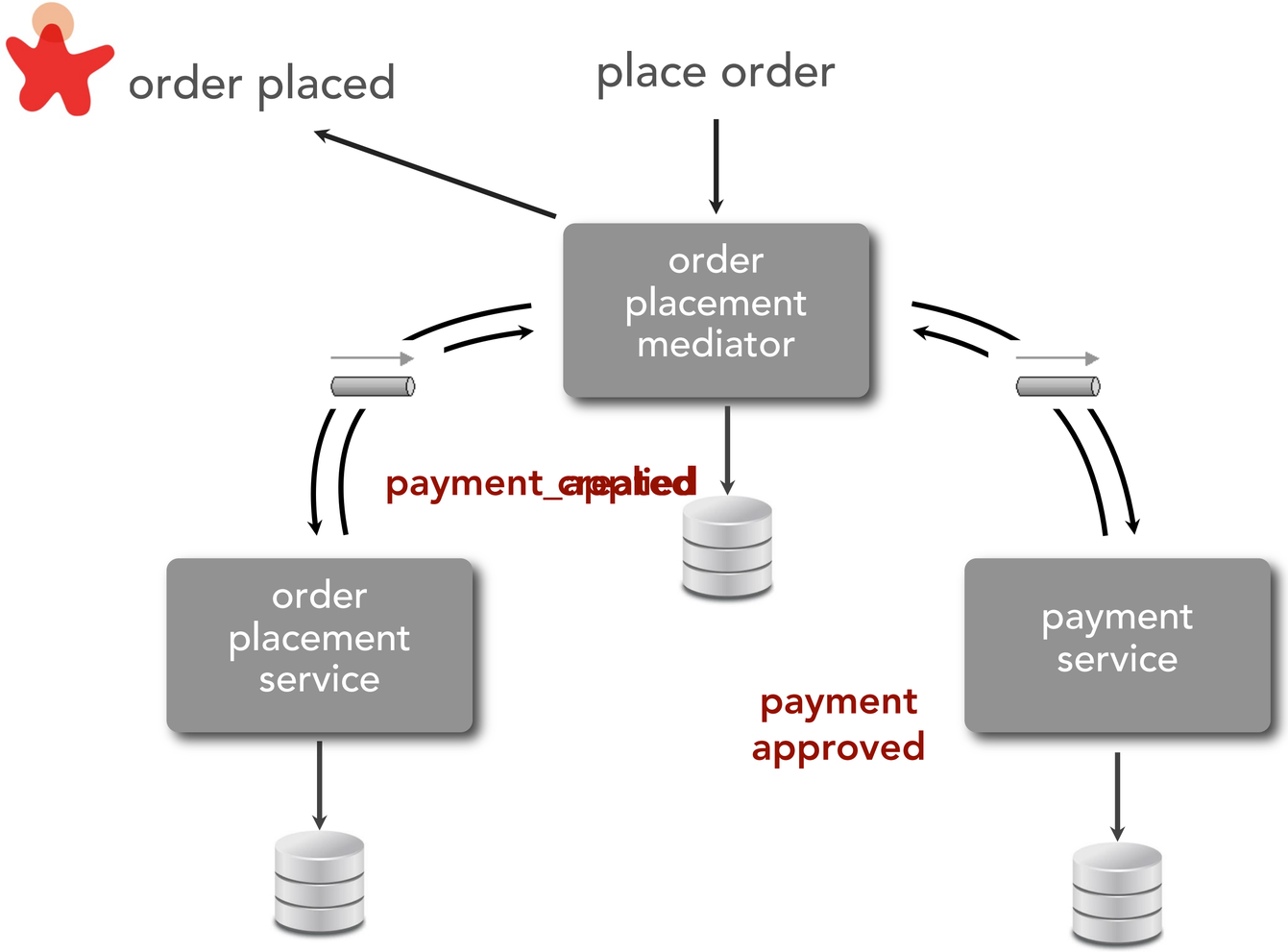
distributed transactions



distributed transactions



distributed transactions



distributed transactions

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Saga {
    public Transaction[] value();

    public enum Transaction {
        ORDER_PLACEMENT,
        ORDER_CANCEL,
        NEW_CUSTOMER
    }
}
```

distributed transactions

```
[System.AttributeUsage(System.AttributeTargets.Class)]
public class Saga : System.Attribute {

    public Transaction[] transaction;

    public enum Transaction {
        ORDER_PLACEMENT,
        ORDER_CANCEL,
        NEW_CUSTOMER
    };
}
```

distributed transactions

```
@ServiceEntrypoint
@Saga(Transaction.ORDER_PLACEMENT)
public class OrderPlacementService {
    ...
}

@ServiceEntrypoint
@Saga({Transaction.ORDER_PLACEMENT,)
       Transaction.ORDER_CANCEL})
public class PaymentService {
    ...
}
```

distributed transactions

```
@ServiceEntrypoint
@Saga(Transaction.ORDER_PLACEMENT)
public class OrderPlacementService {
    ...
}
```

only those transactions that span multiple services are sagas!

distributed transactions

ACID - atomicity, consistency, isolation, durability

BASE - basic availability, soft state, eventual consistency

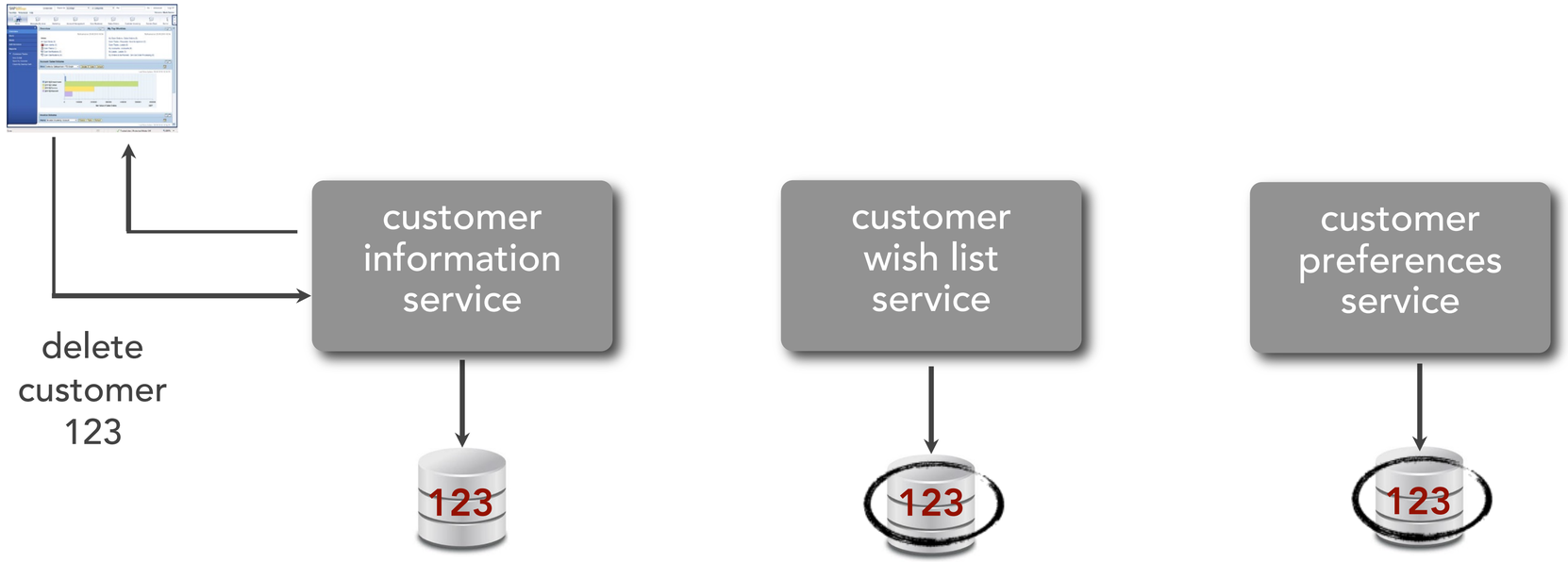
background async

request-based

event-based

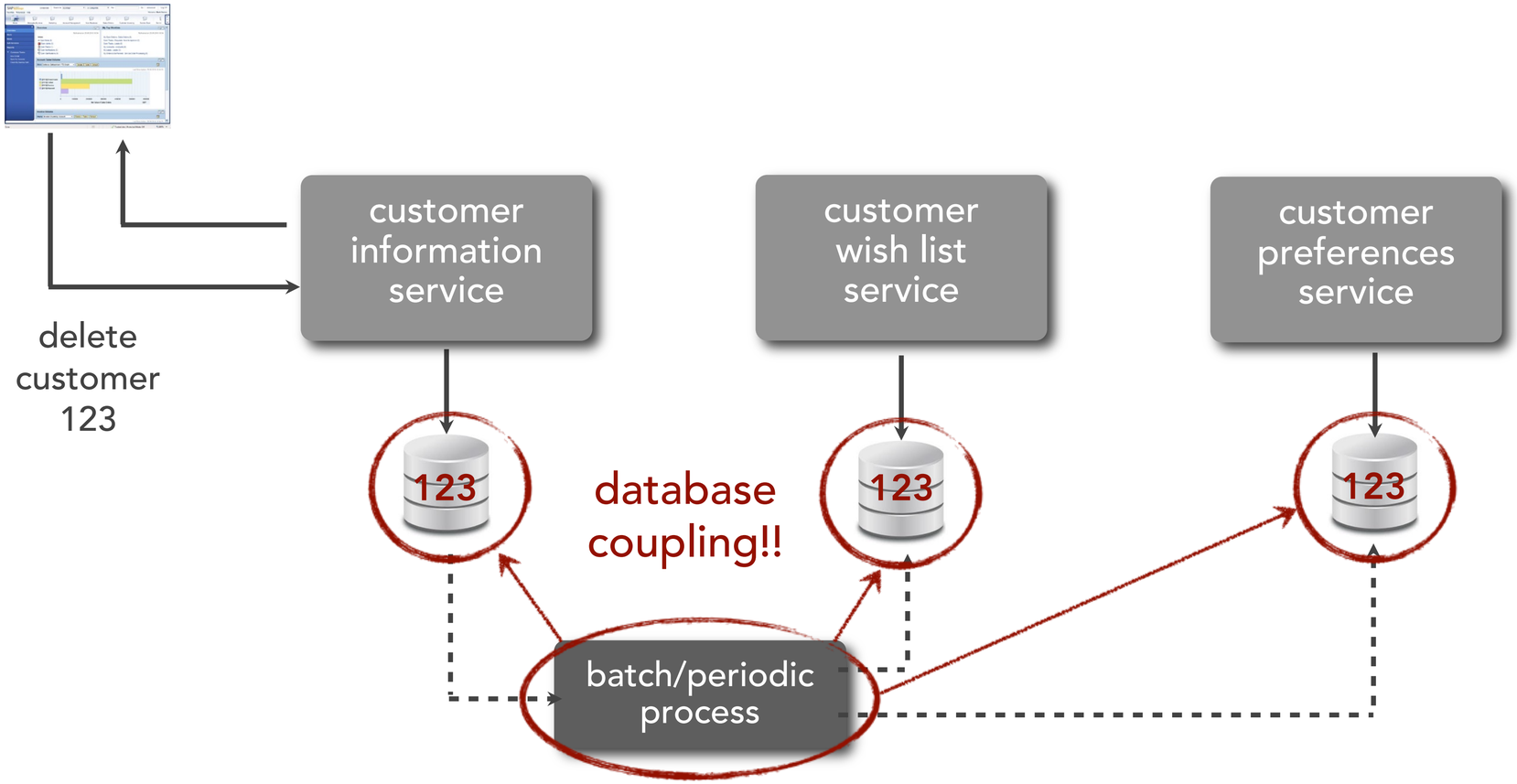
distributed transactions

the problem...



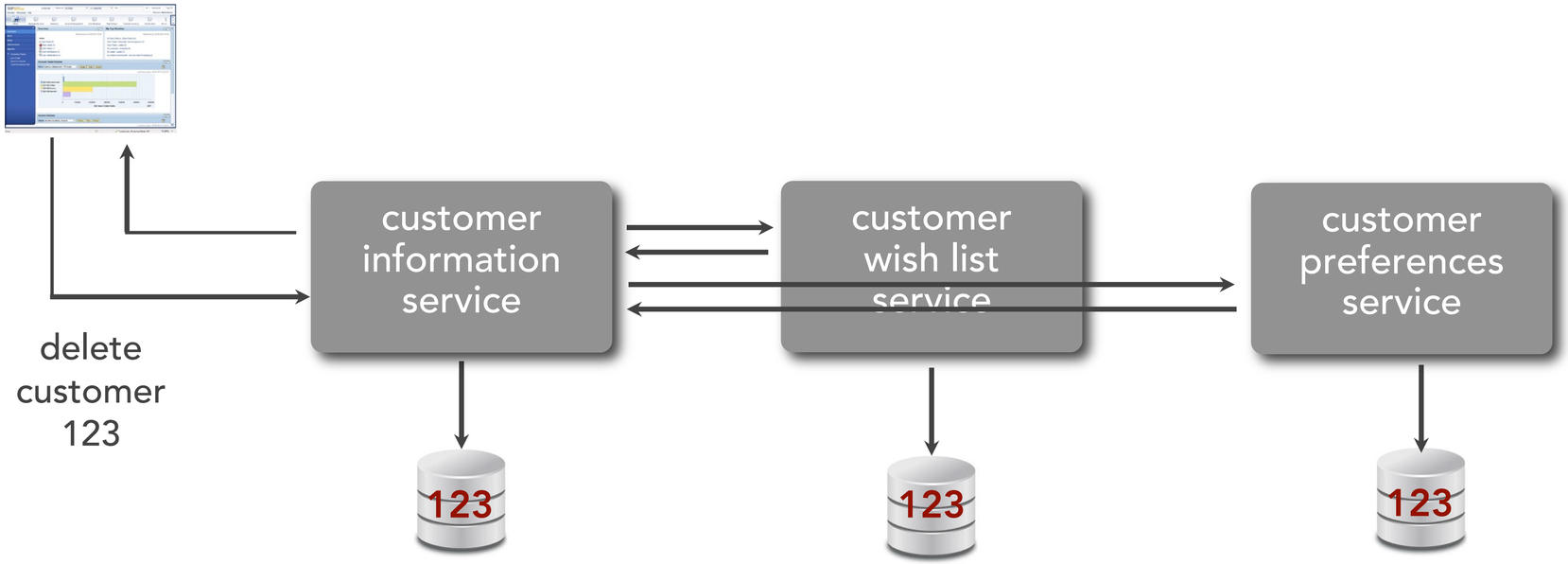
distributed transactions

background synchronization



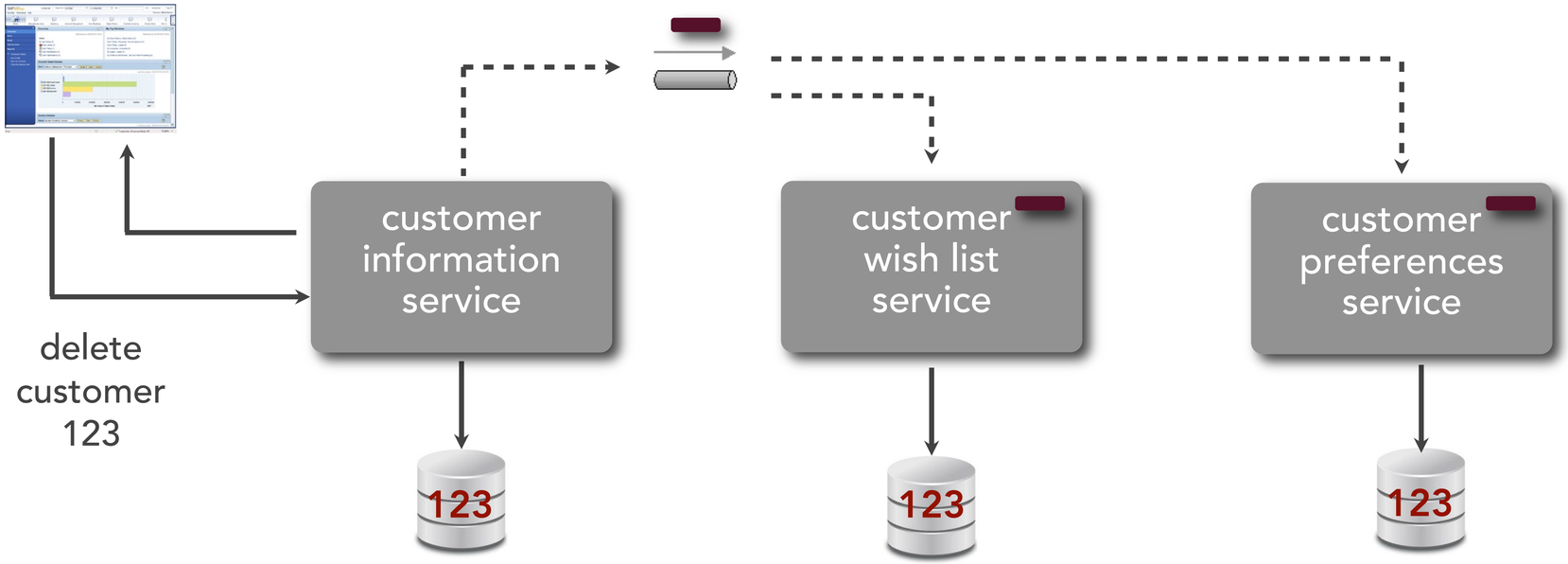
distributed transactions

request-based data synchronization



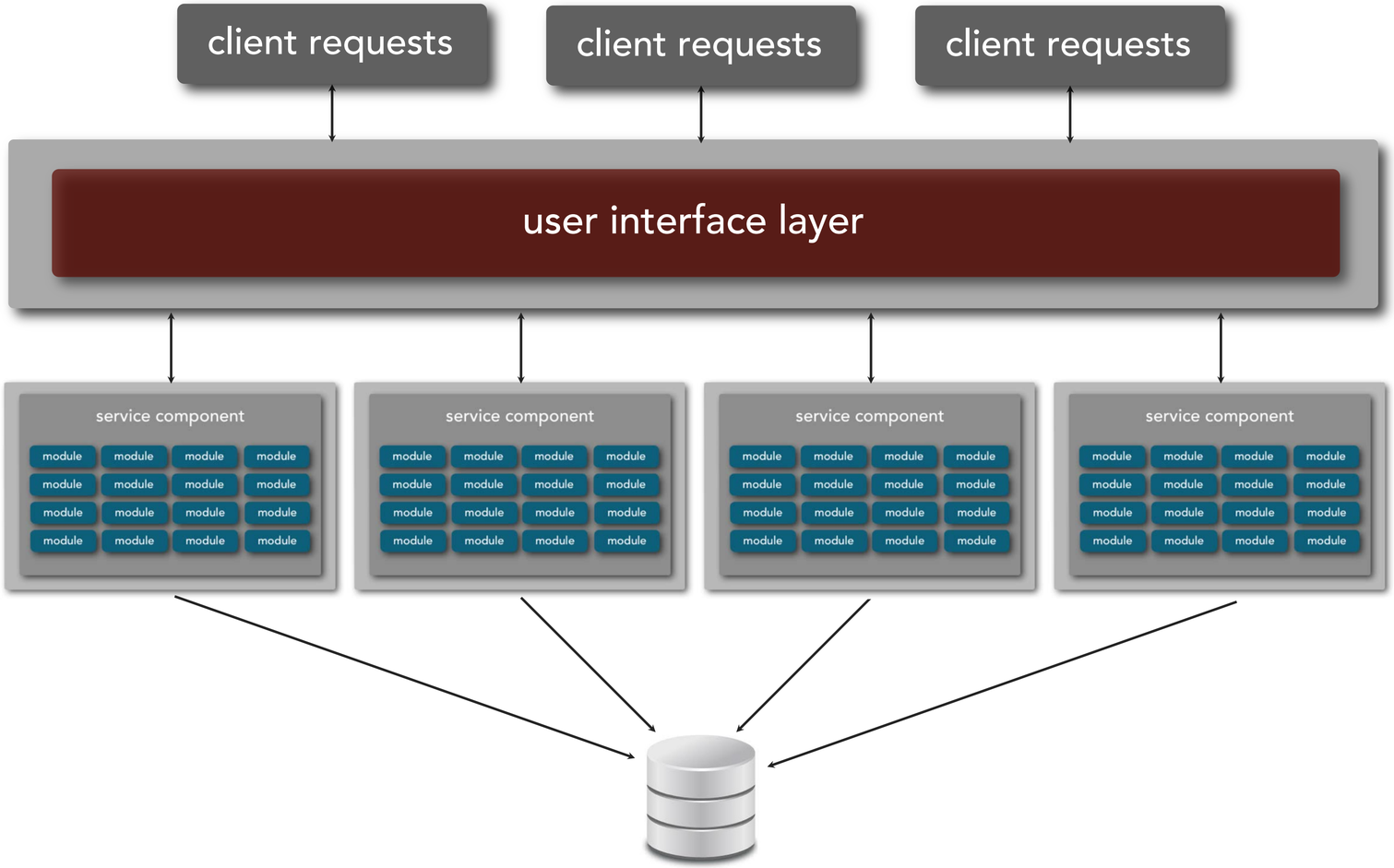
distributed transactions

event-based data synchronization

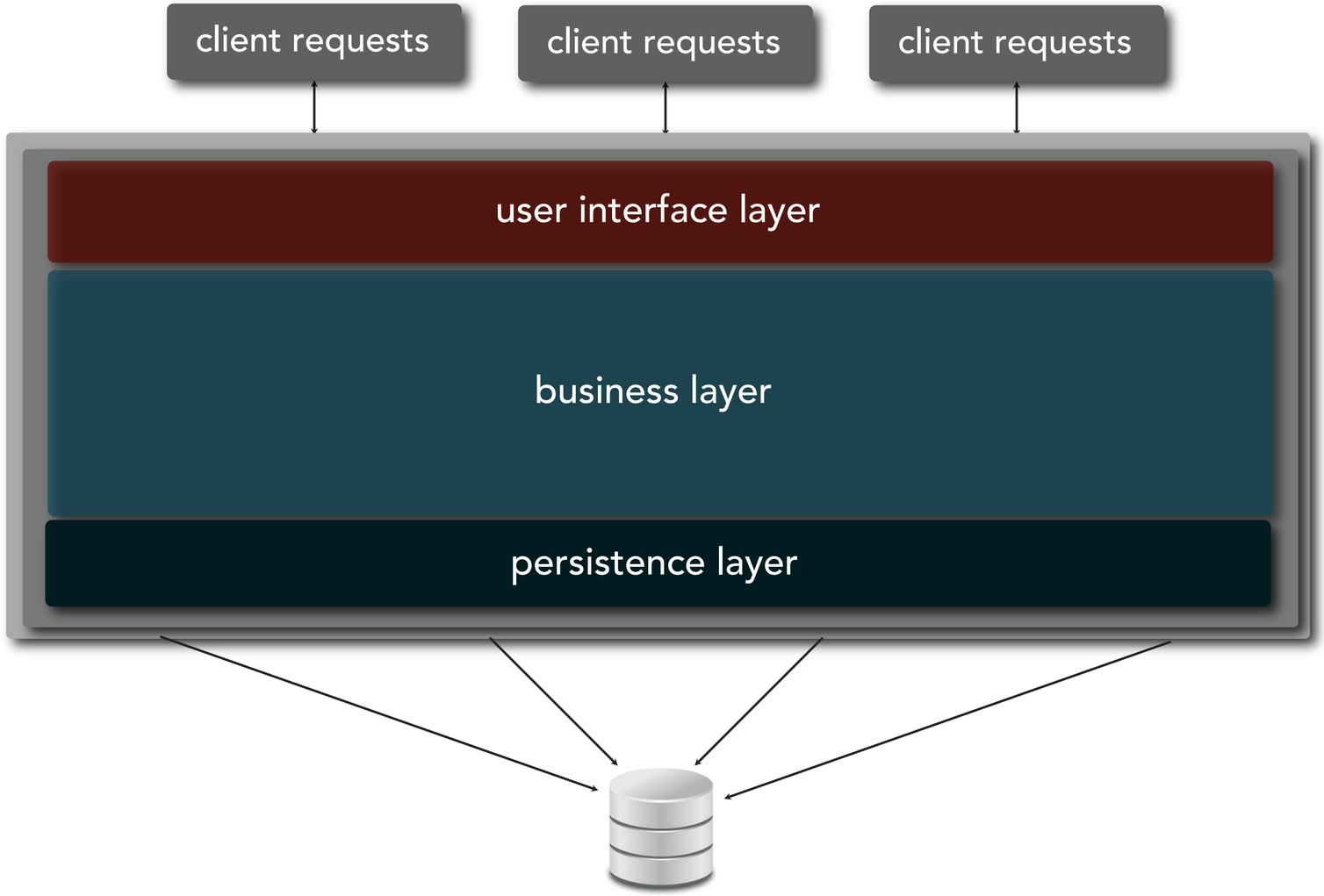


Hybrid Architectures

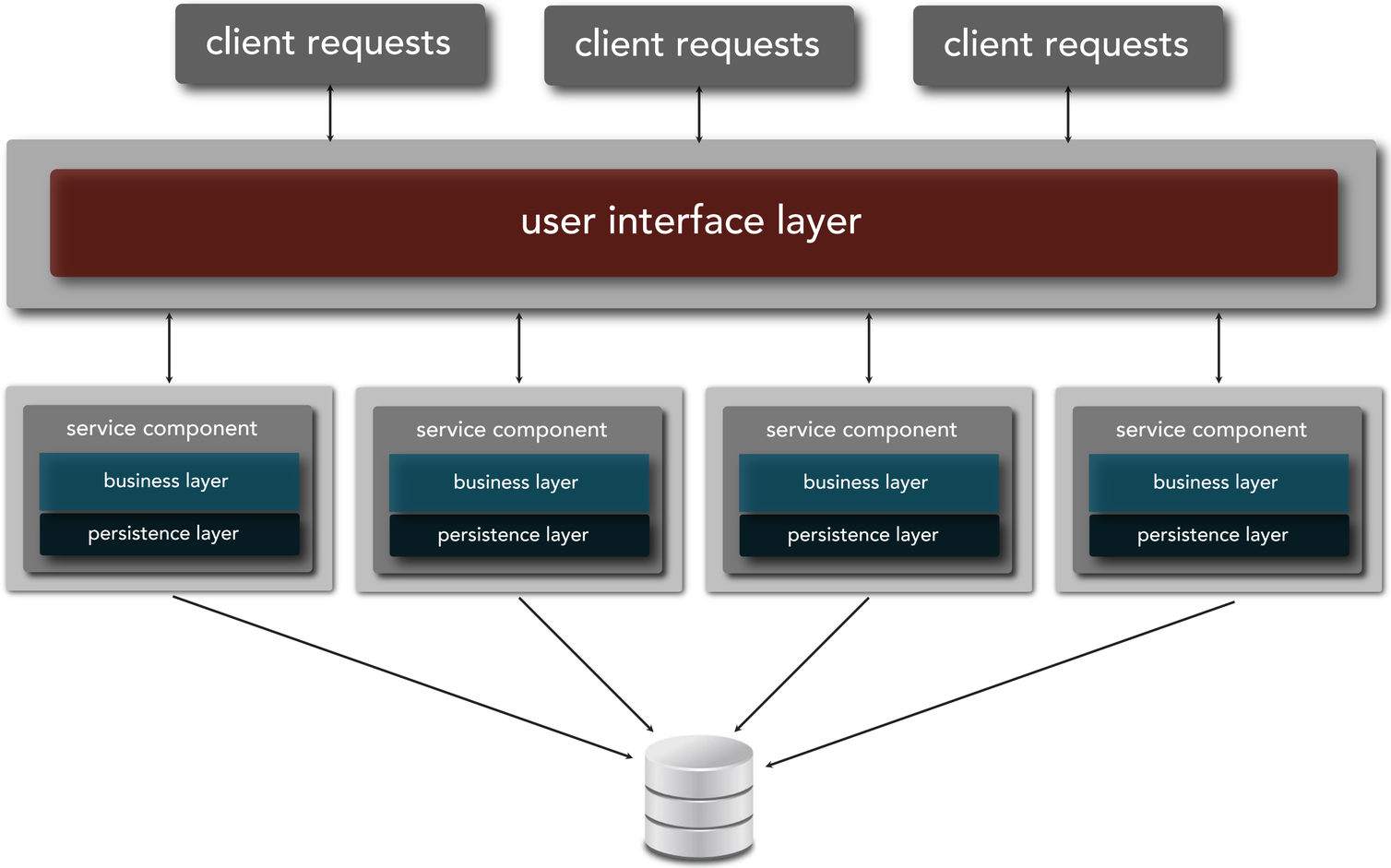
service-based architecture



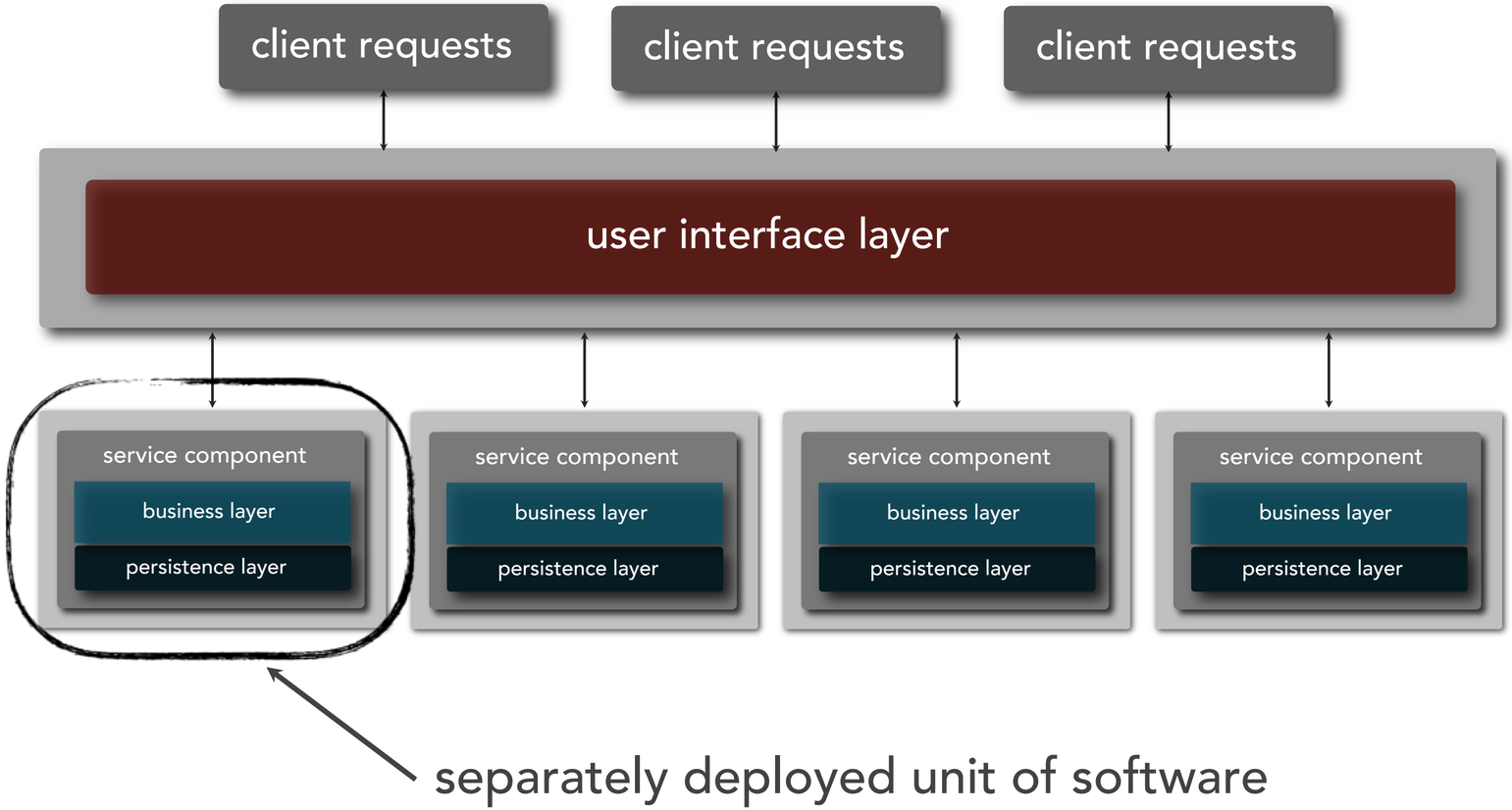
service-based architecture



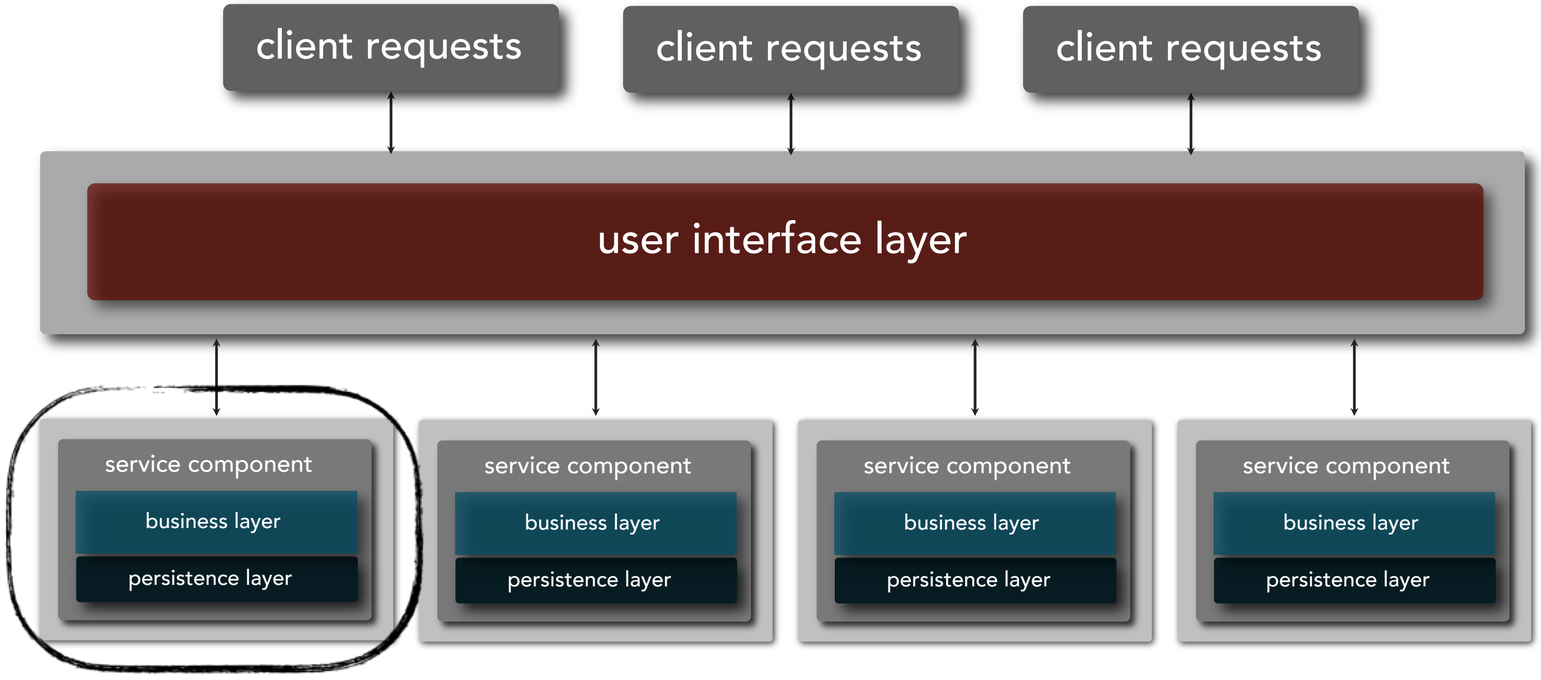
service-based architecture



service-based architecture

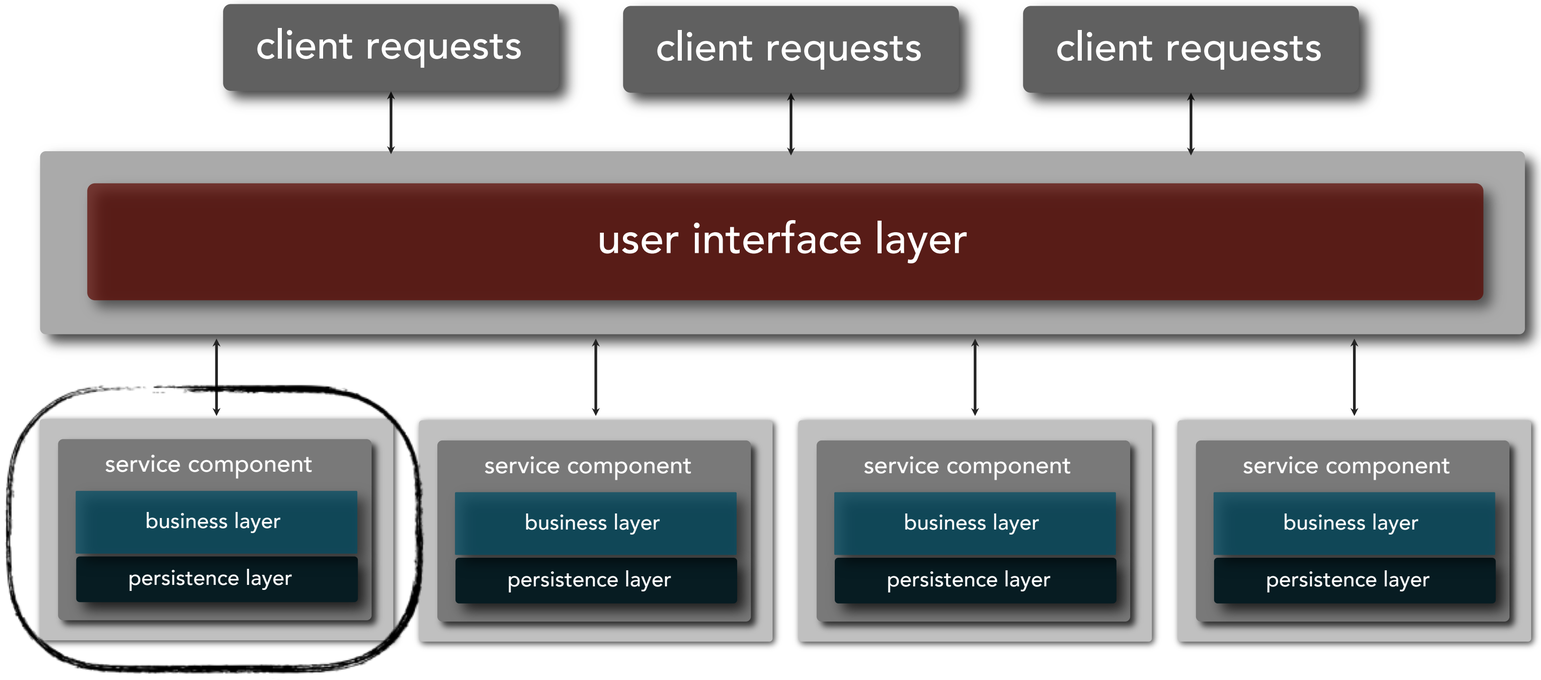


service-based architecture



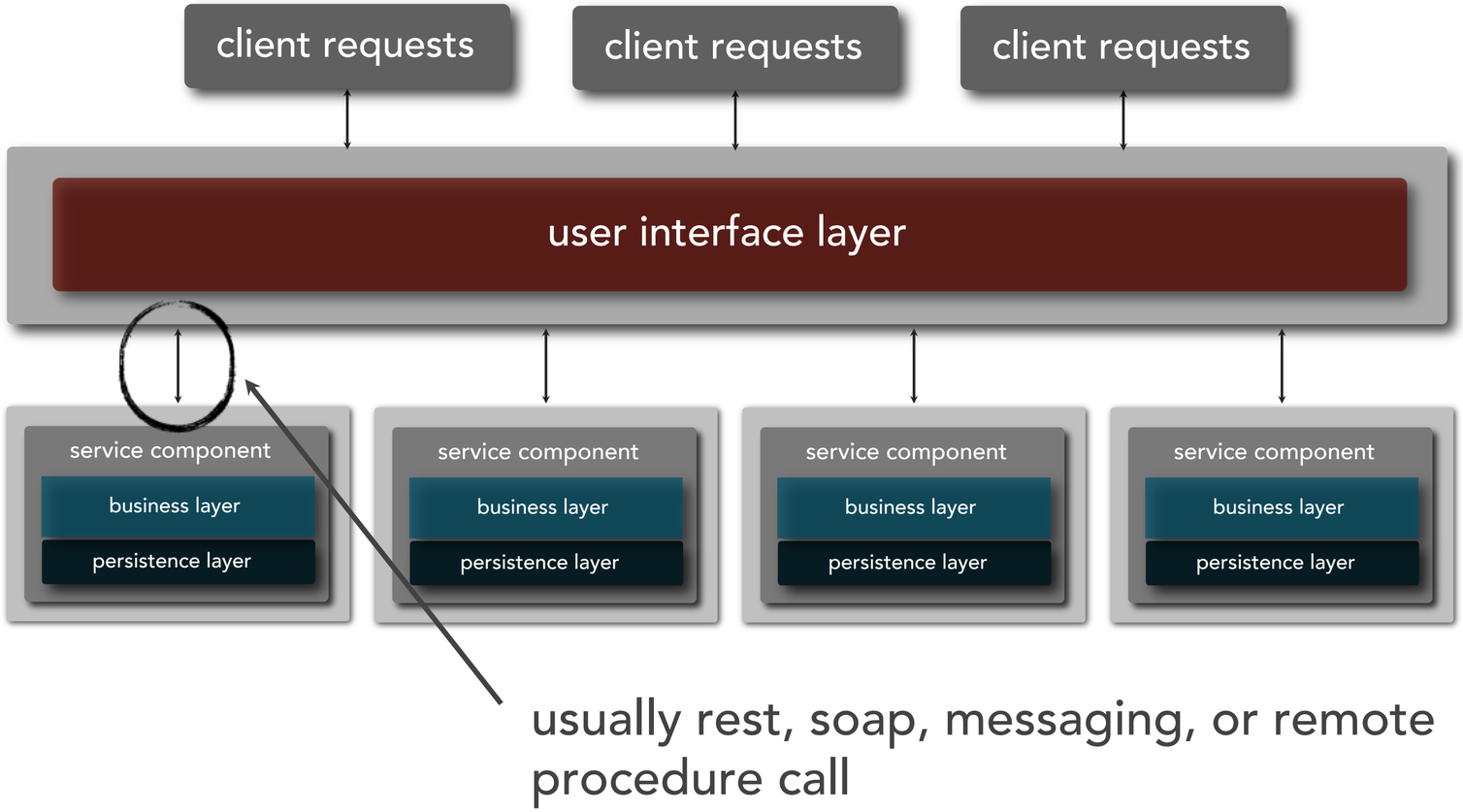
same deployment unit as original application
(ear, assembly, etc.)

service-based architecture

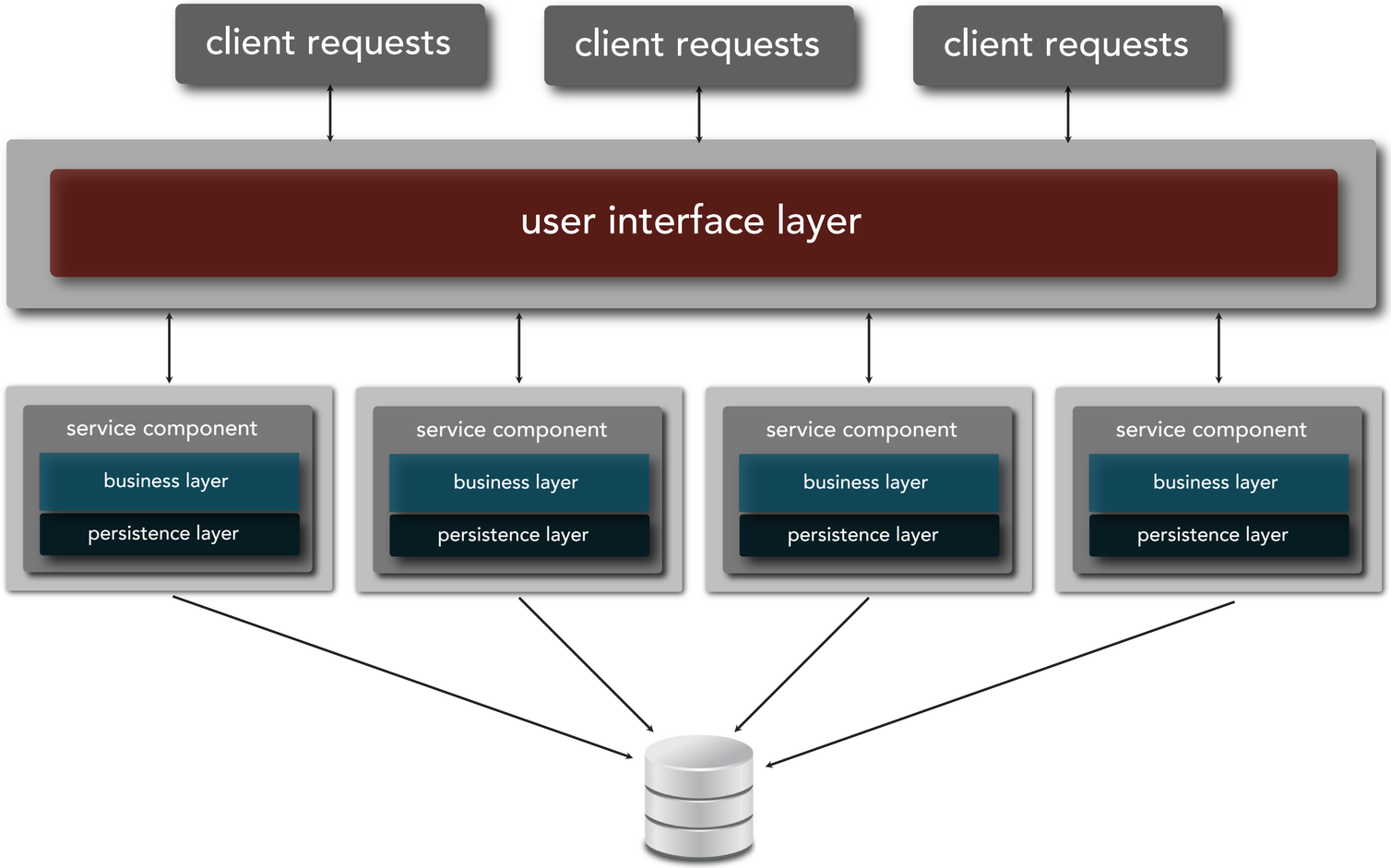


independent self-contained portion of the application (6 - 12 services)

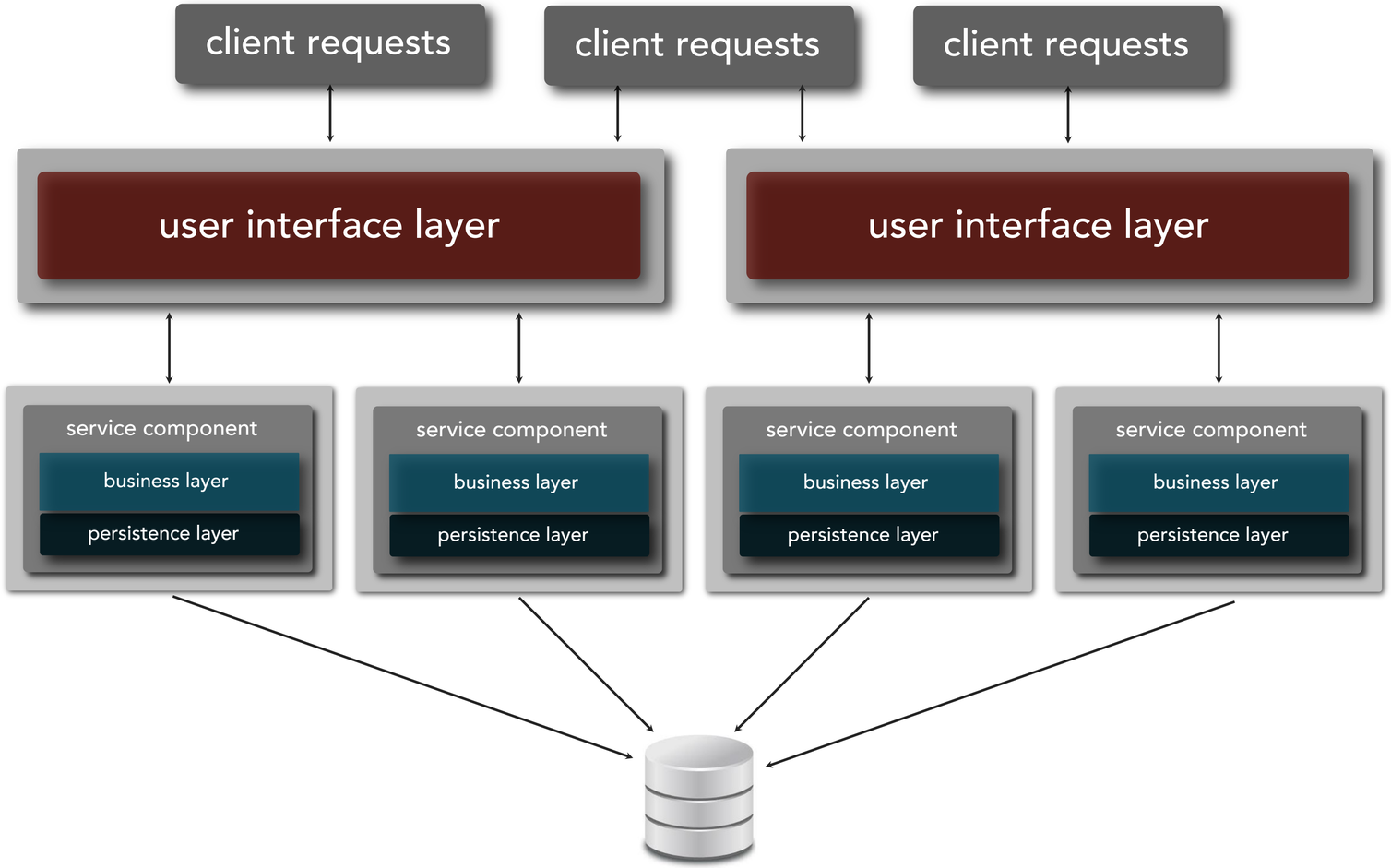
service-based architecture



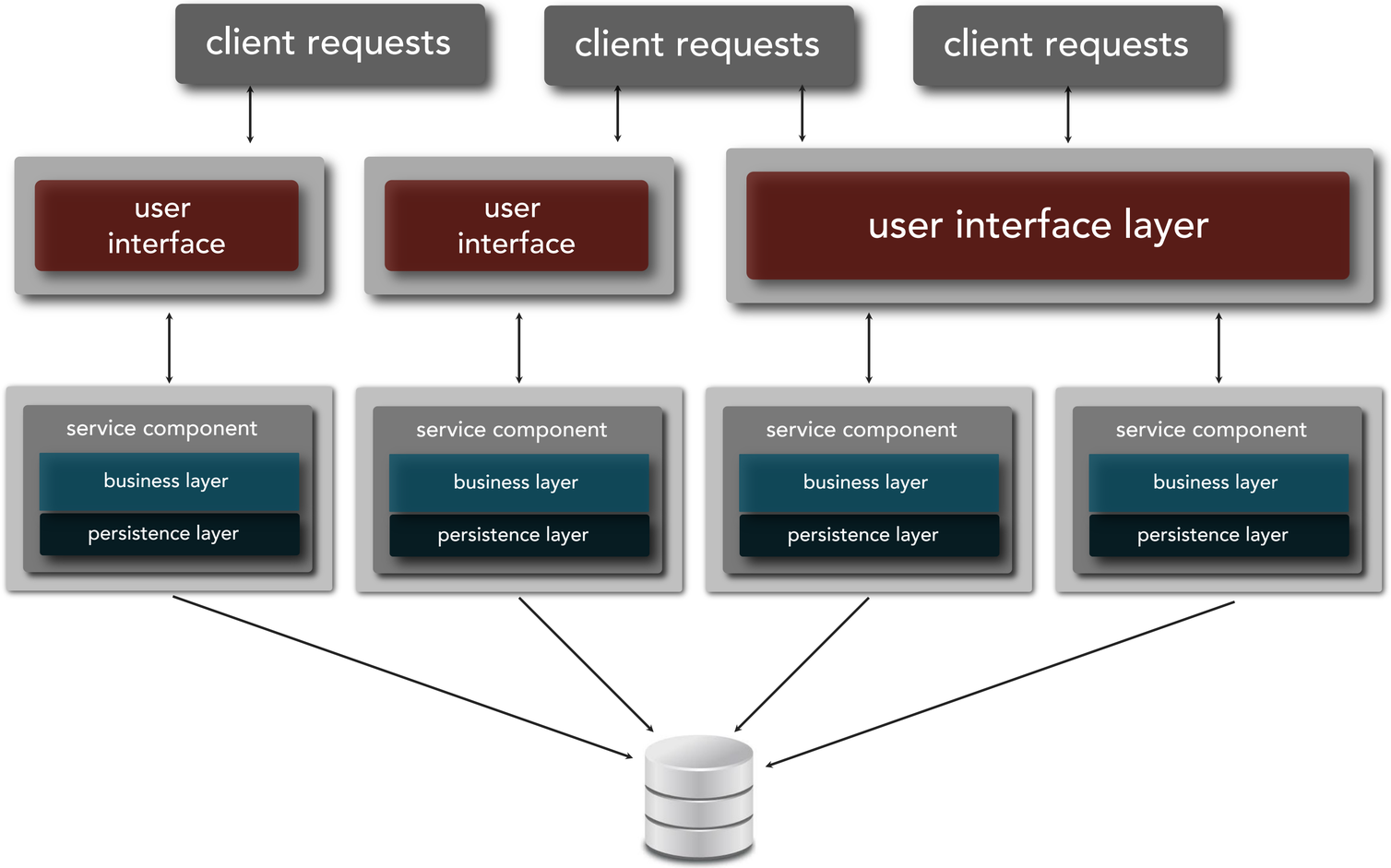
service-based architecture



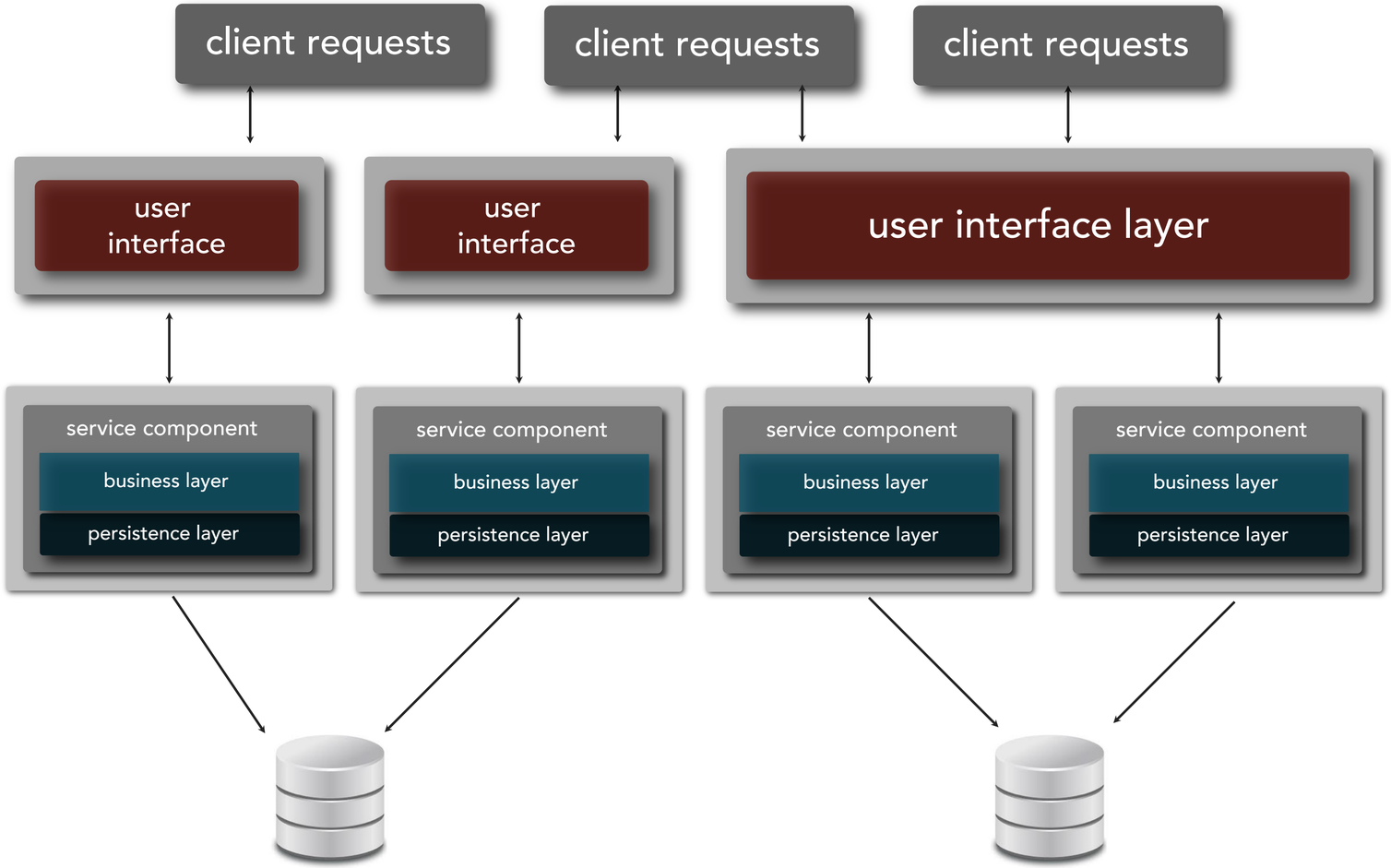
service-based architecture



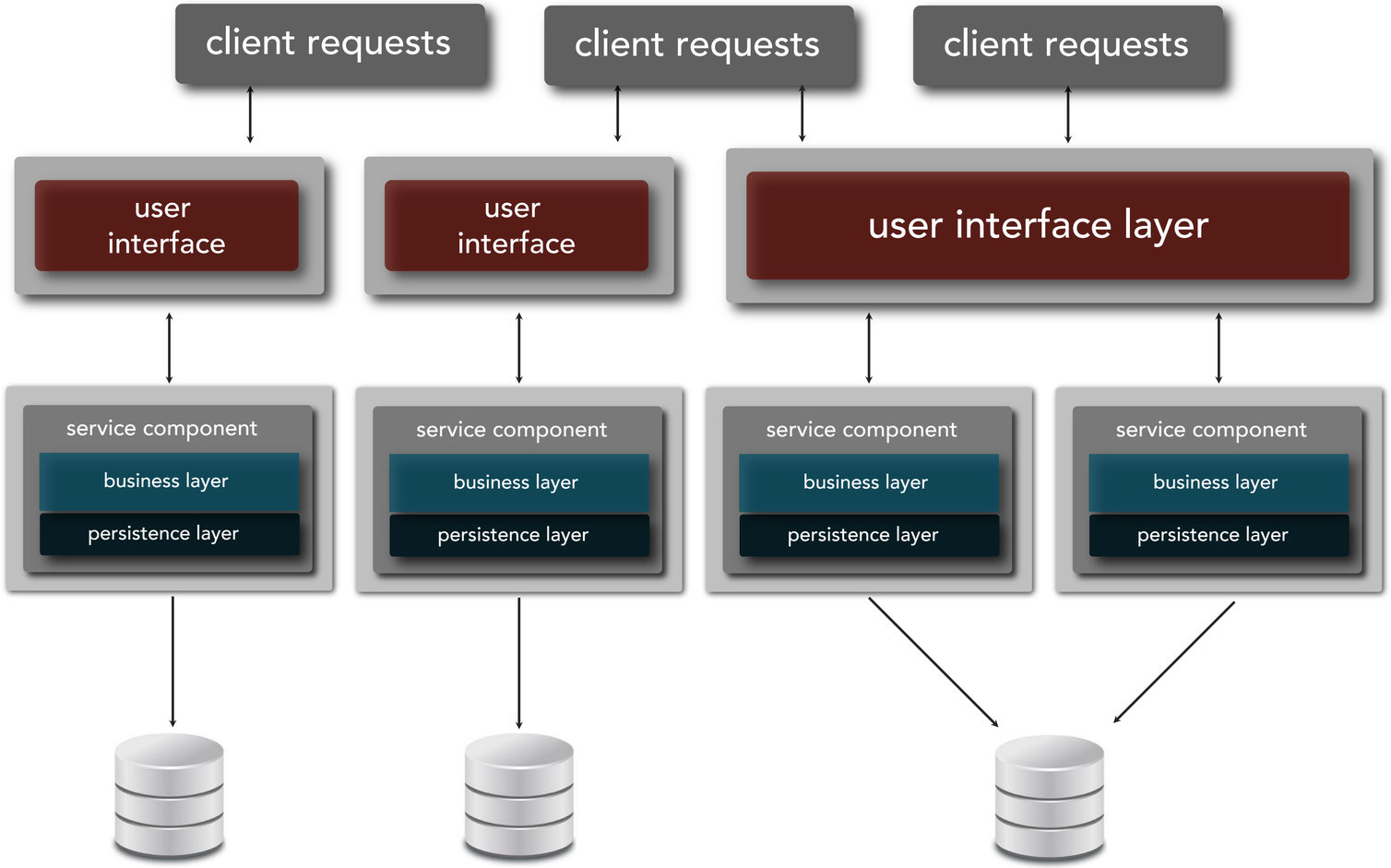
service-based architecture



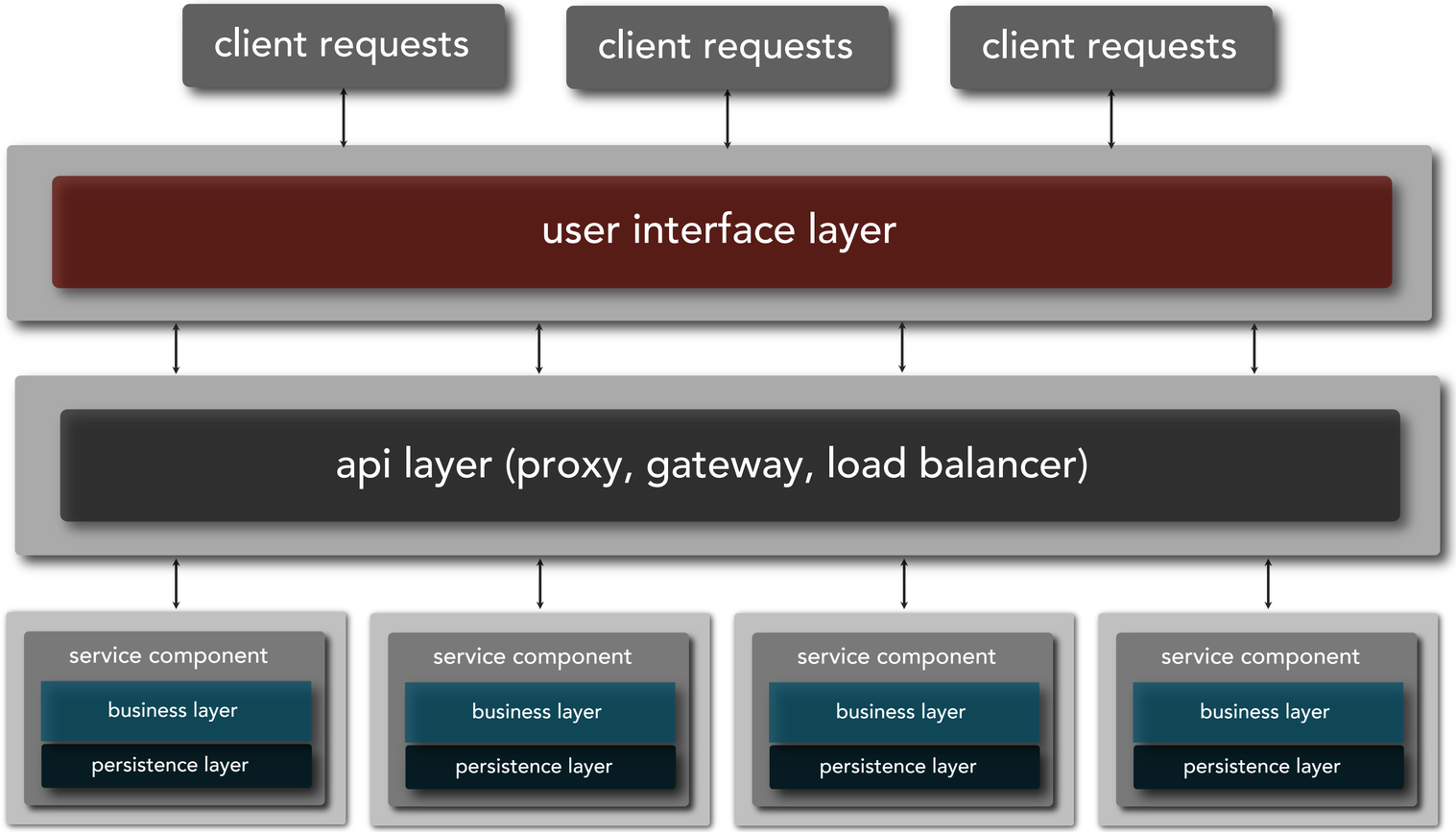
service-based architecture



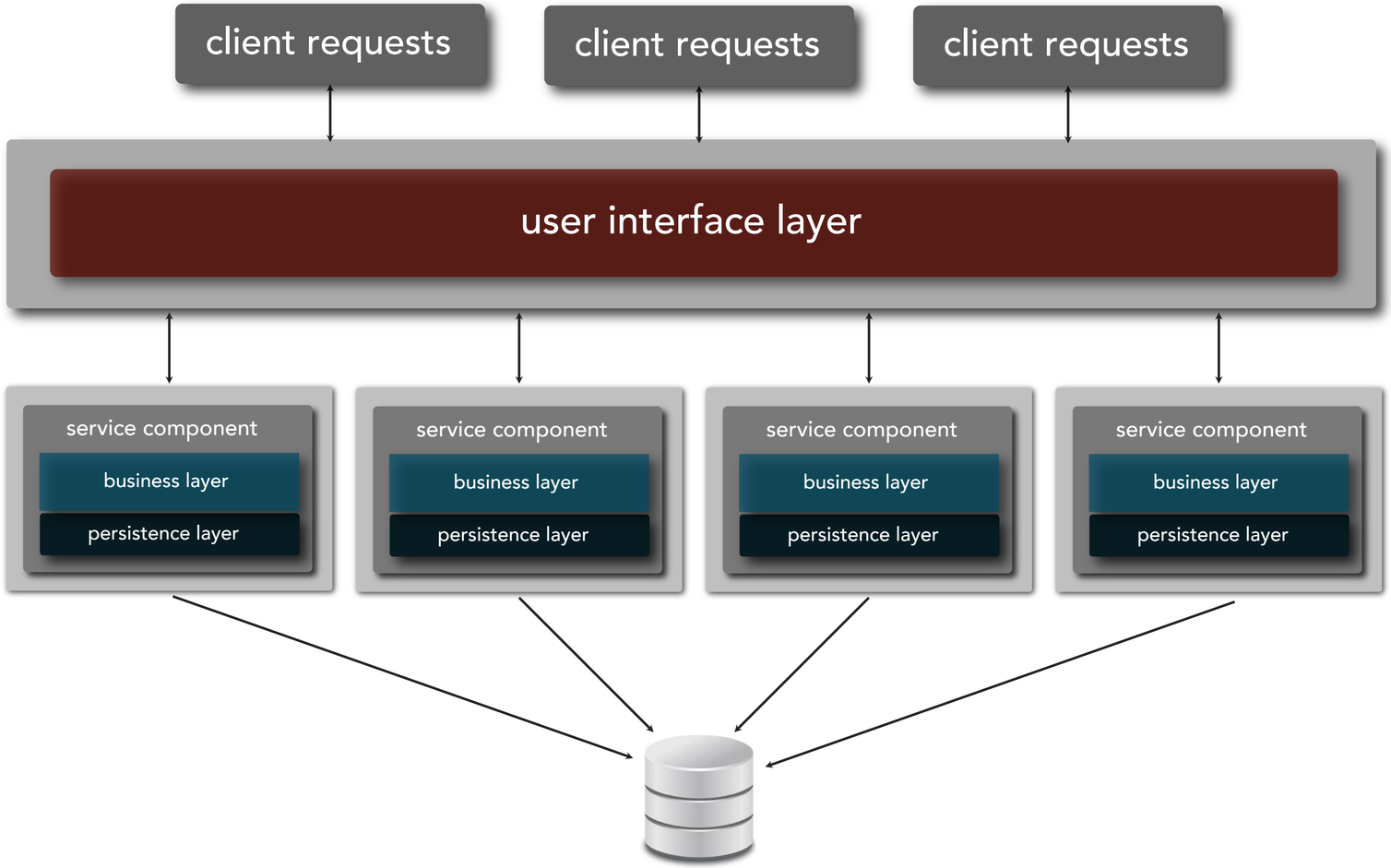
service-based architecture



service-based architecture



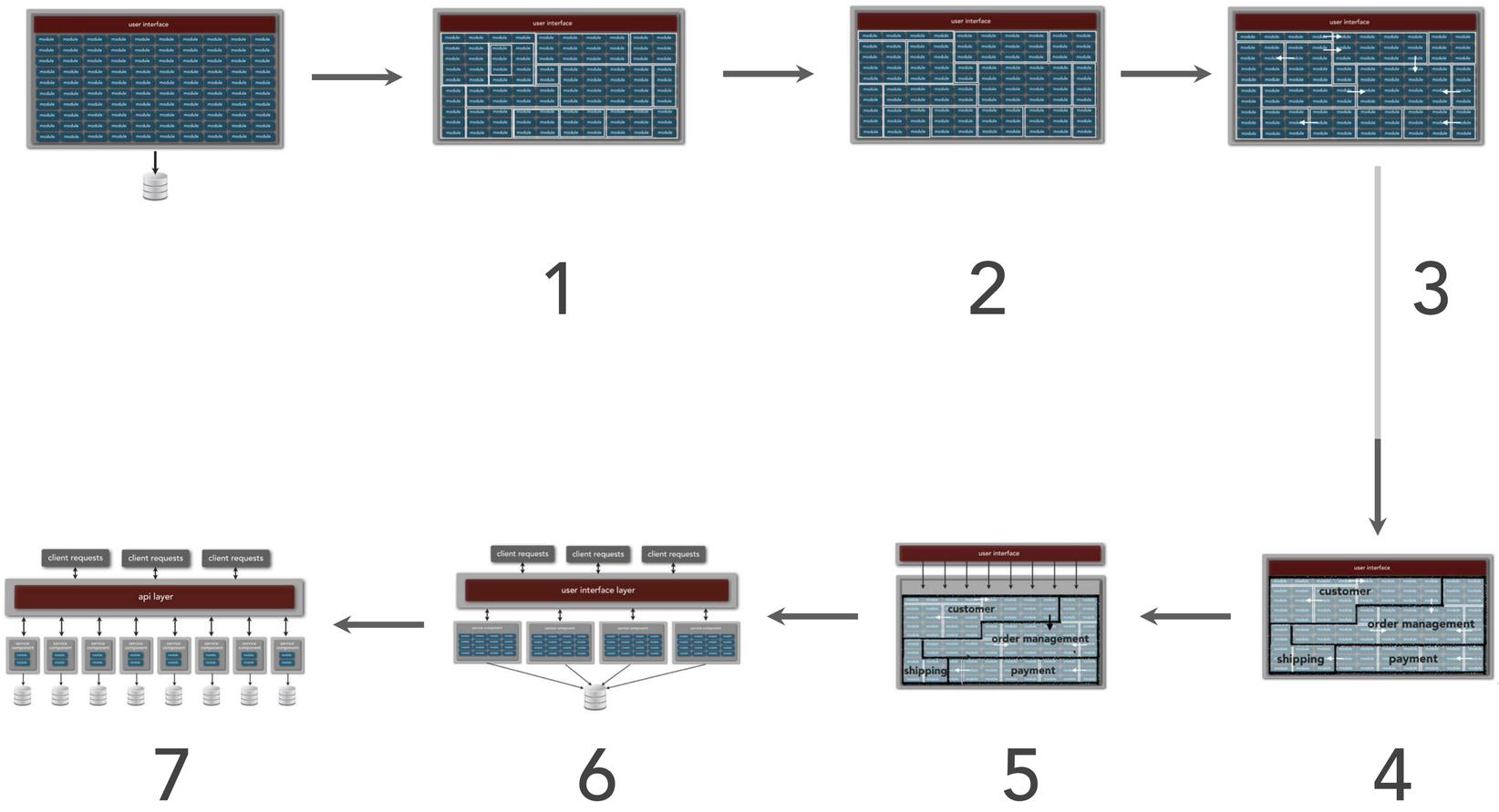
service-based architecture



Migration Patterns

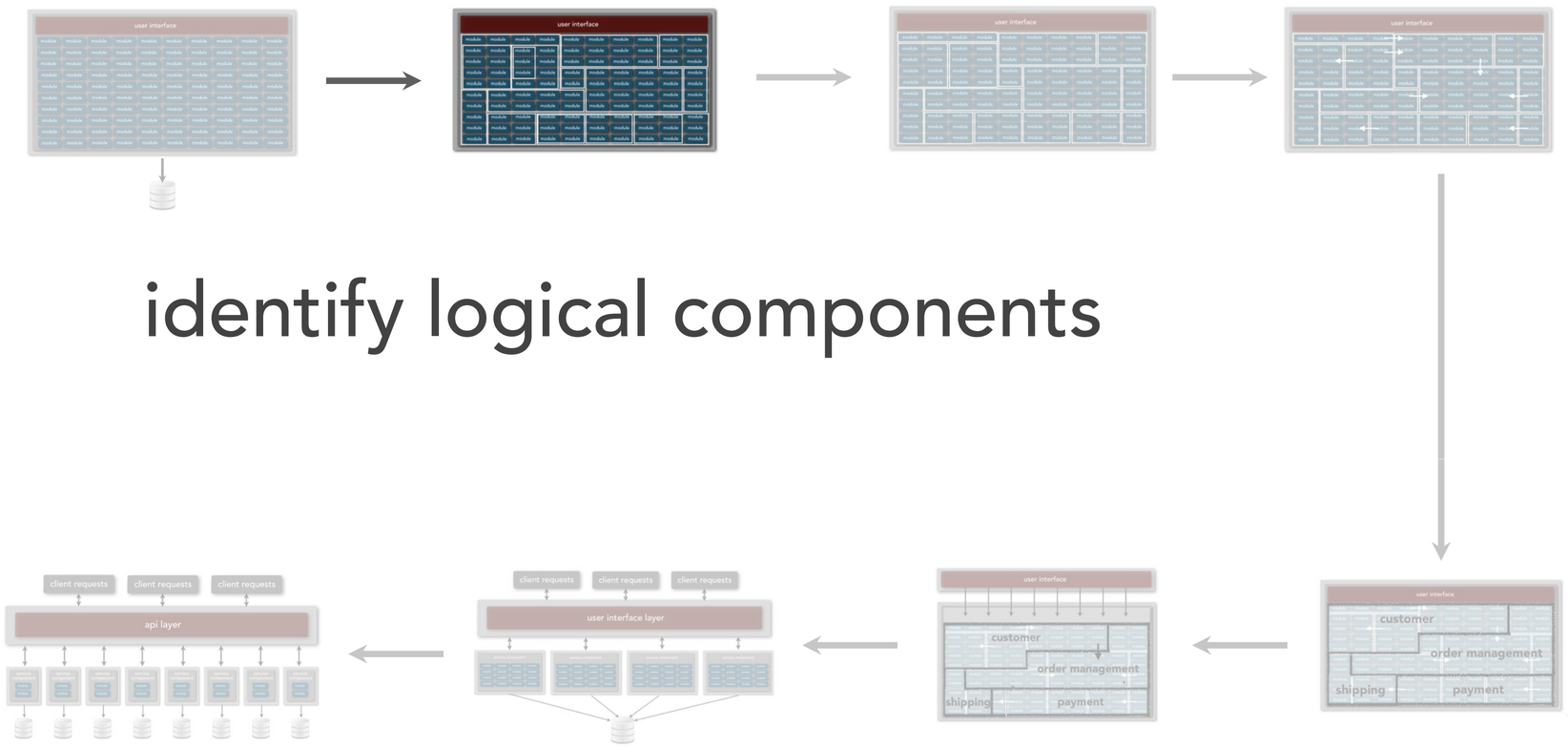
migration patterns

microservices roadmap



migration patterns

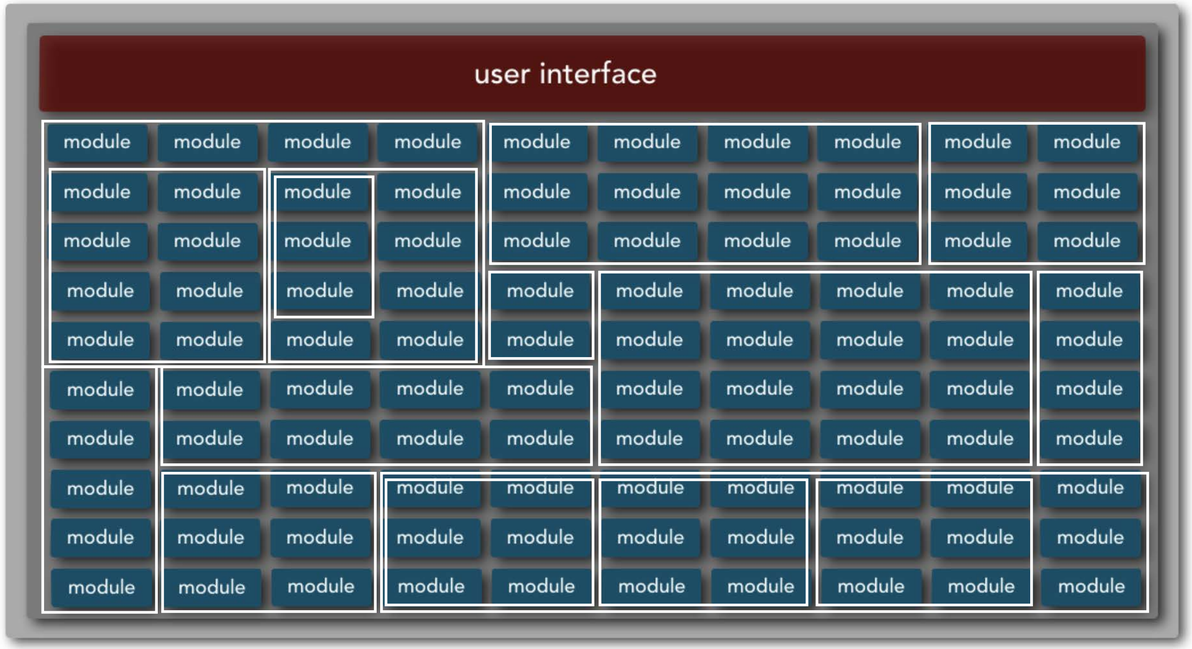
microservices roadmap



identify logical components

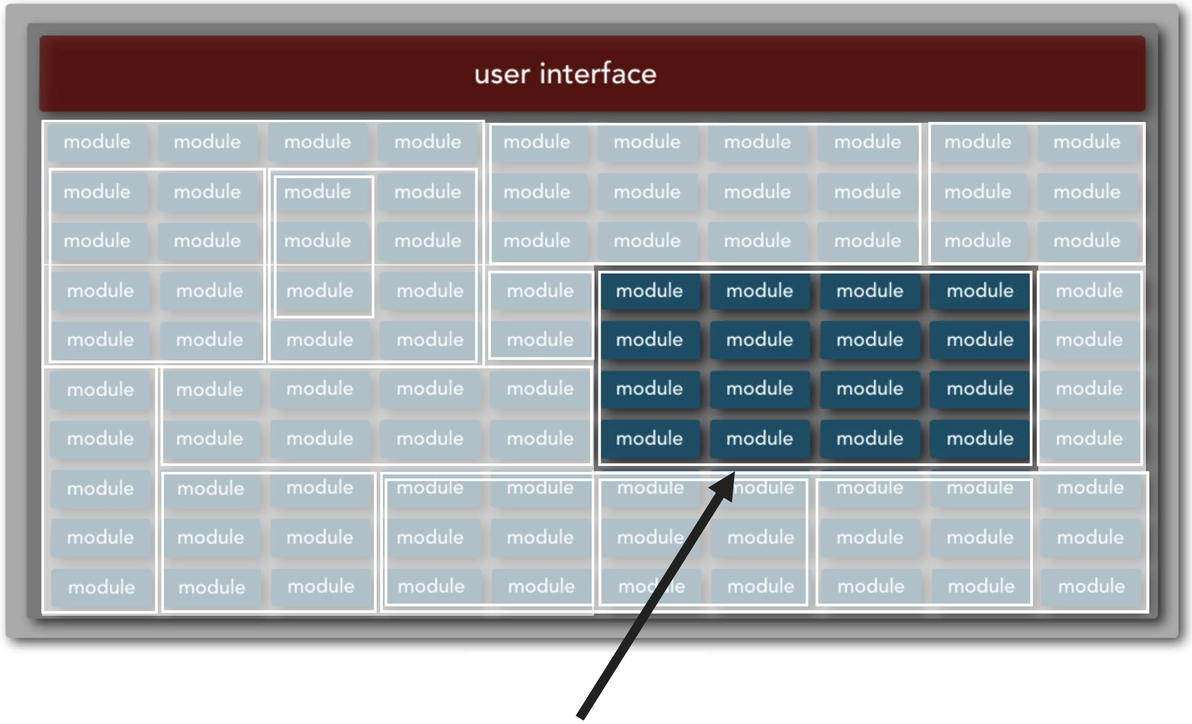
migration patterns

identify logical components



migration patterns

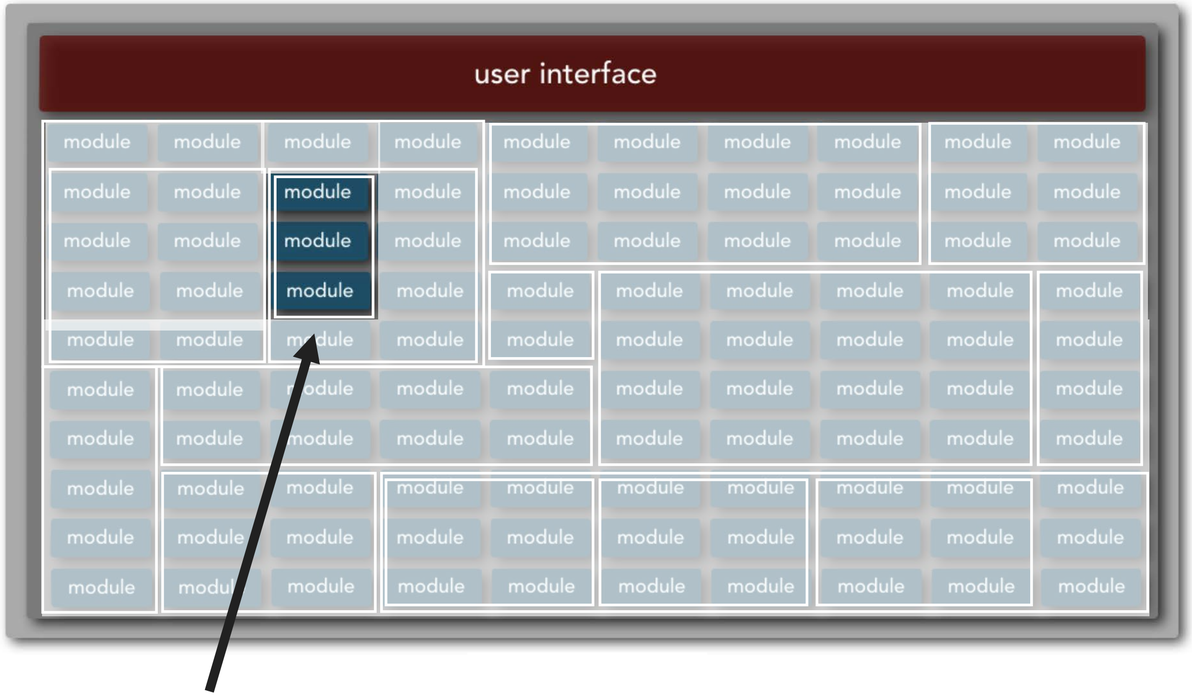
identify logical components



app.payment

migration patterns

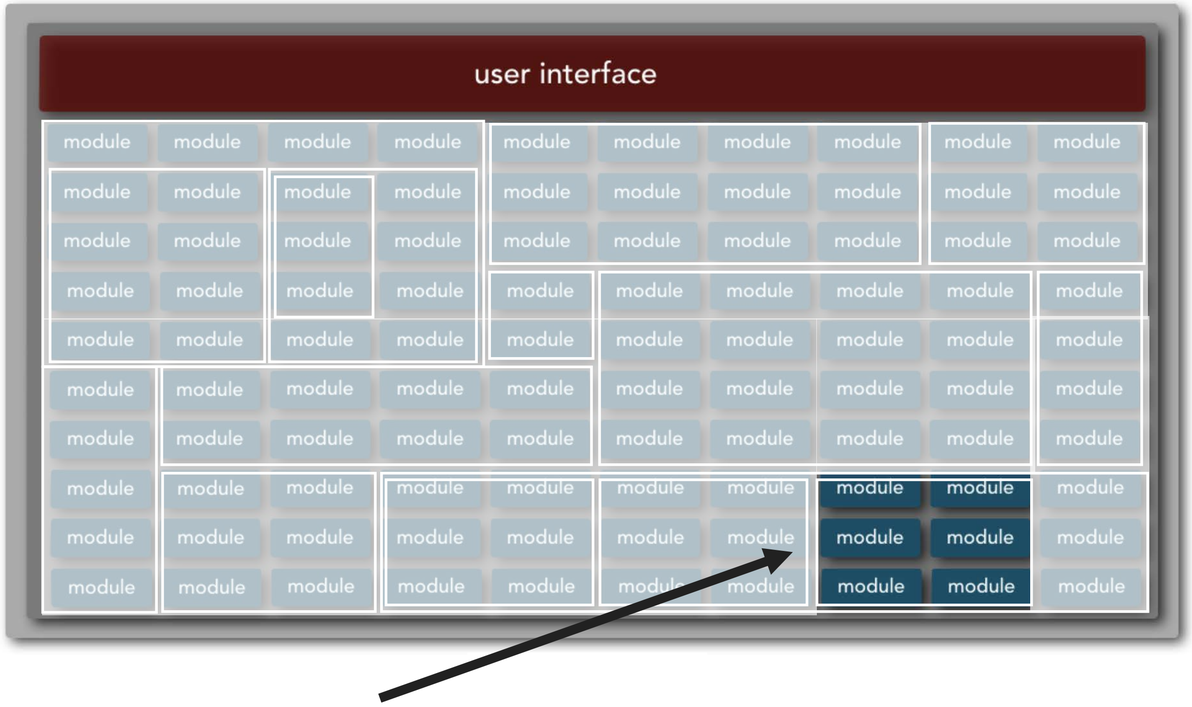
identify logical components



app.customer.notification.sms

migration patterns

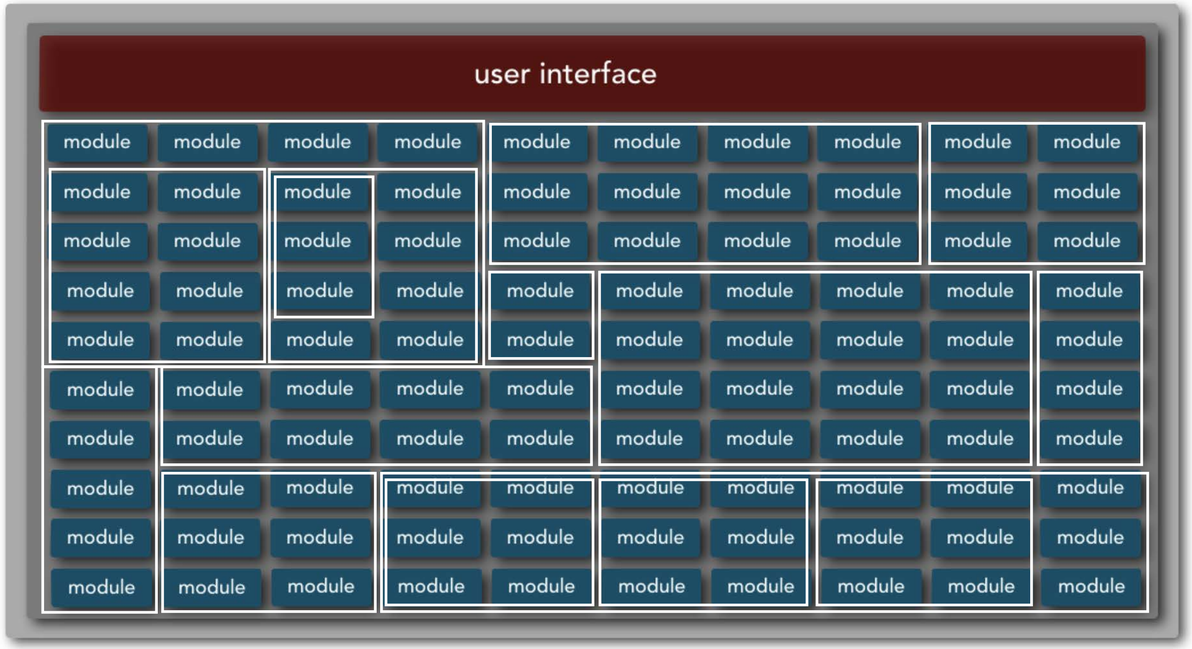
identify logical components



app.order.history

migration patterns

identify logical components



Component Size Analysis
Total Statements: 129304
Total Components: 27
Avg Statements Per Component: 4789
Standard Deviation: 10535

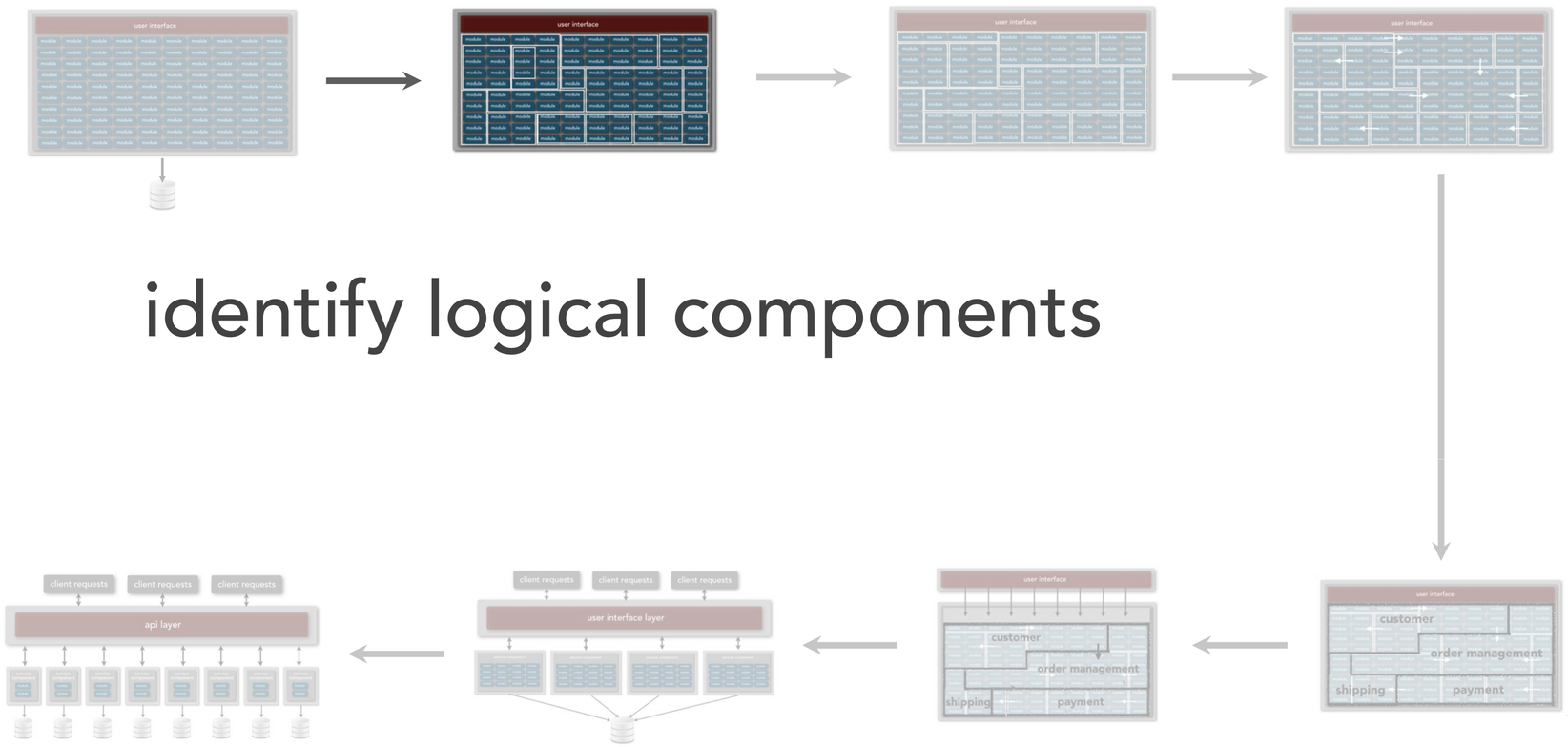
%	stmts	files	sdev	root	ca	ce	ct	component
1	53	3	1	*	24	2	26	business.framework.bos
41	54183	581	5		20	10	30	business.framework.bos.ae
3	4787	46	1		5	5	10	business.framework.bos.bi
7	9245	97	1		0	5	5	business.framework.bos.bv
3	5023	79	1		5	11	16	business.framework.bos.cm
2	2721	29	1		2	8	10	business.framework.bos.cn
6	8511	46	1		16	4	20	business.framework.bos.cr
3	4694	48	1		6	7	13	business.framework.bos.cs
4	5174	73	1		4	10	14	business.framework.bos.di
2	3387	71	1		2	7	9	business.framework.bos.dx
1	2234	83	1		3	5	8	business.framework.bos.ed
5	7002	77	1		2	9	11	business.framework.bos.es
1	1015	41	1		7	6	13	business.framework.bos.fw
1	421	13	1		0	5	5	business.framework.bos.hm
1	385	6	1		0	7	7	business.framework.bos.mi
1	163	7	1		0	3	3	business.framework.bos.pr

Component Size Analysis
Total Statements: 129304
Total Components: 27
Avg Statements Per Component: 4789
Standard Deviation: 10535

%	stmts	files	sdev	root	ca	ce	ct	component
1	53	3	1	*	24	2	26	business.framework.bos
41	54183	581	5		20	10	30	business.framework.bos.ae
3	4787	46	1		5	5	10	business.framework.bos.bi
7	9245	97	1		0	5	5	business.framework.bos.bv
3	5023	79	1		5	11	16	business.framework.bos.cm
2	2721	29	1		2	8	10	business.framework.bos.cn
6	8511	46	1		16	4	20	business.framework.bos.cr
3	4694	48	1		6	7	13	business.framework.bos.cs
4	5174	73	1		4	10	14	business.framework.bos.di
2	3387	71	1		2	7	9	business.framework.bos.dx
1	2234	83	1		3	5	8	business.framework.bos.ed
5	7002	77	1		2	9	11	business.framework.bos.es
1	1015	41	1		7	6	13	business.framework.bos.fw
1	421	13	1		0	5	5	business.framework.bos.hm
1	385	6	1		0	7	7	business.framework.bos.mi
1	163	7	1		0	3	3	business.framework.bos.pr

migration patterns

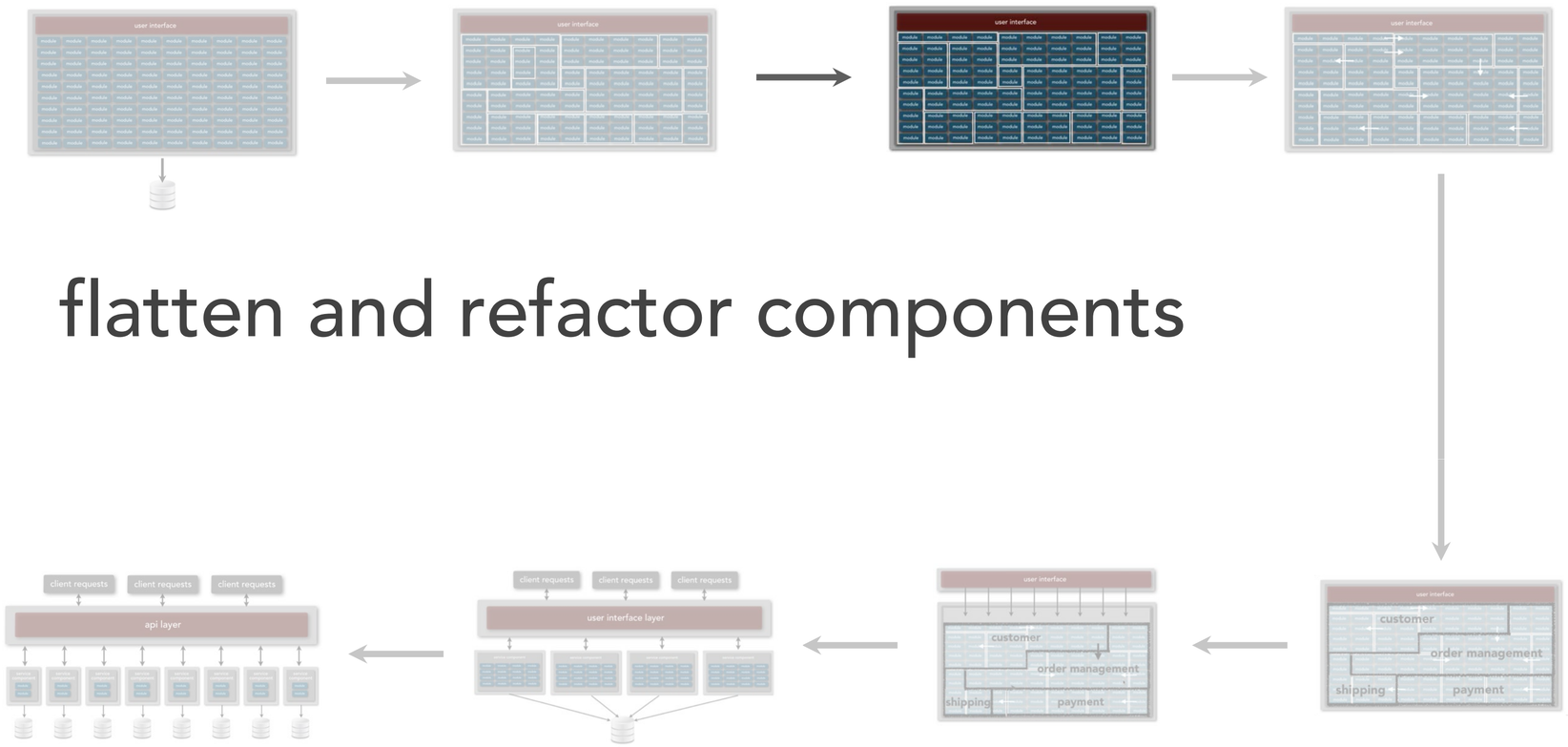
microservices roadmap



identify logical components

migration patterns

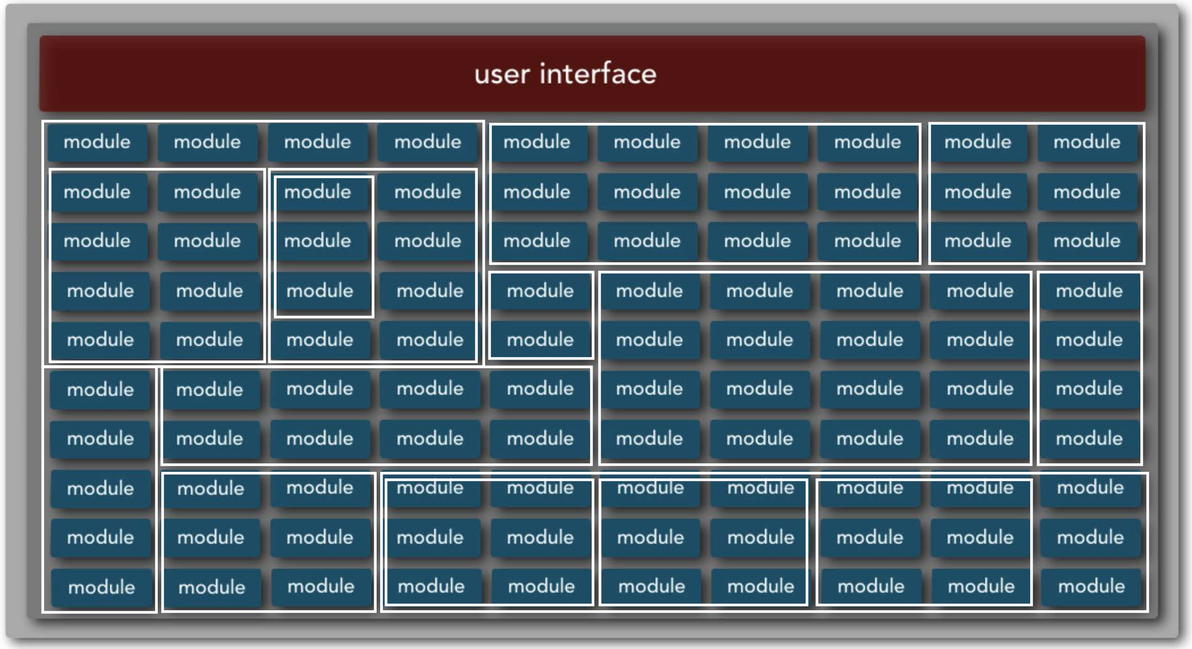
microservices roadmap



flatten and refactor components

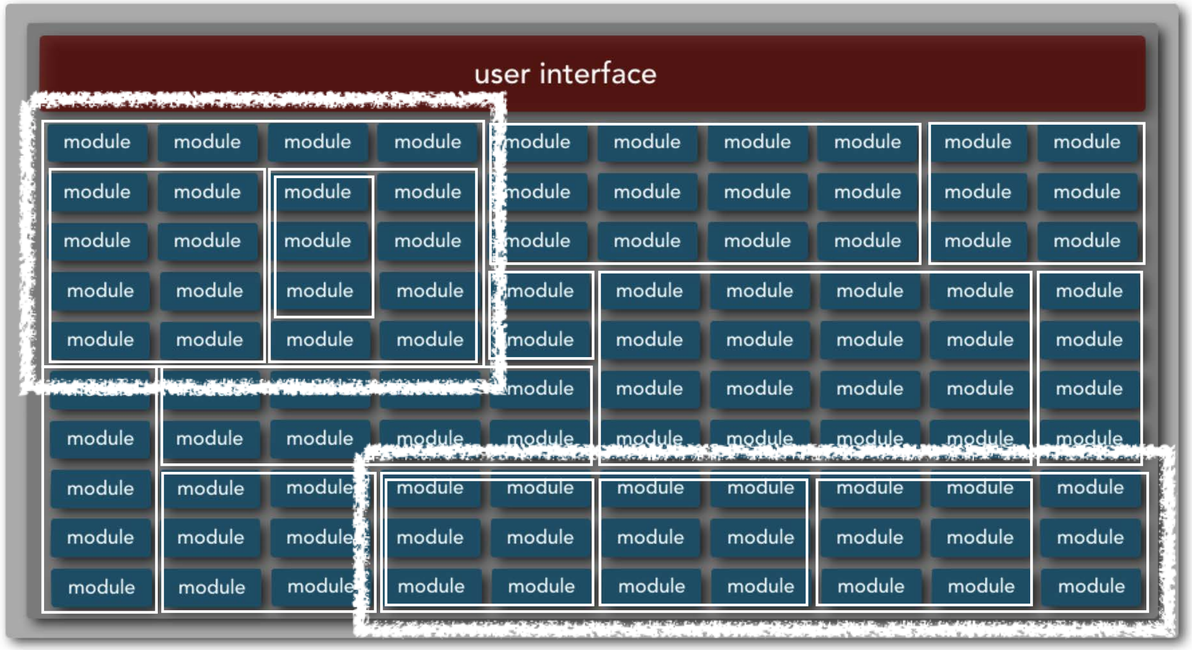
migration patterns

flatten and refactor components



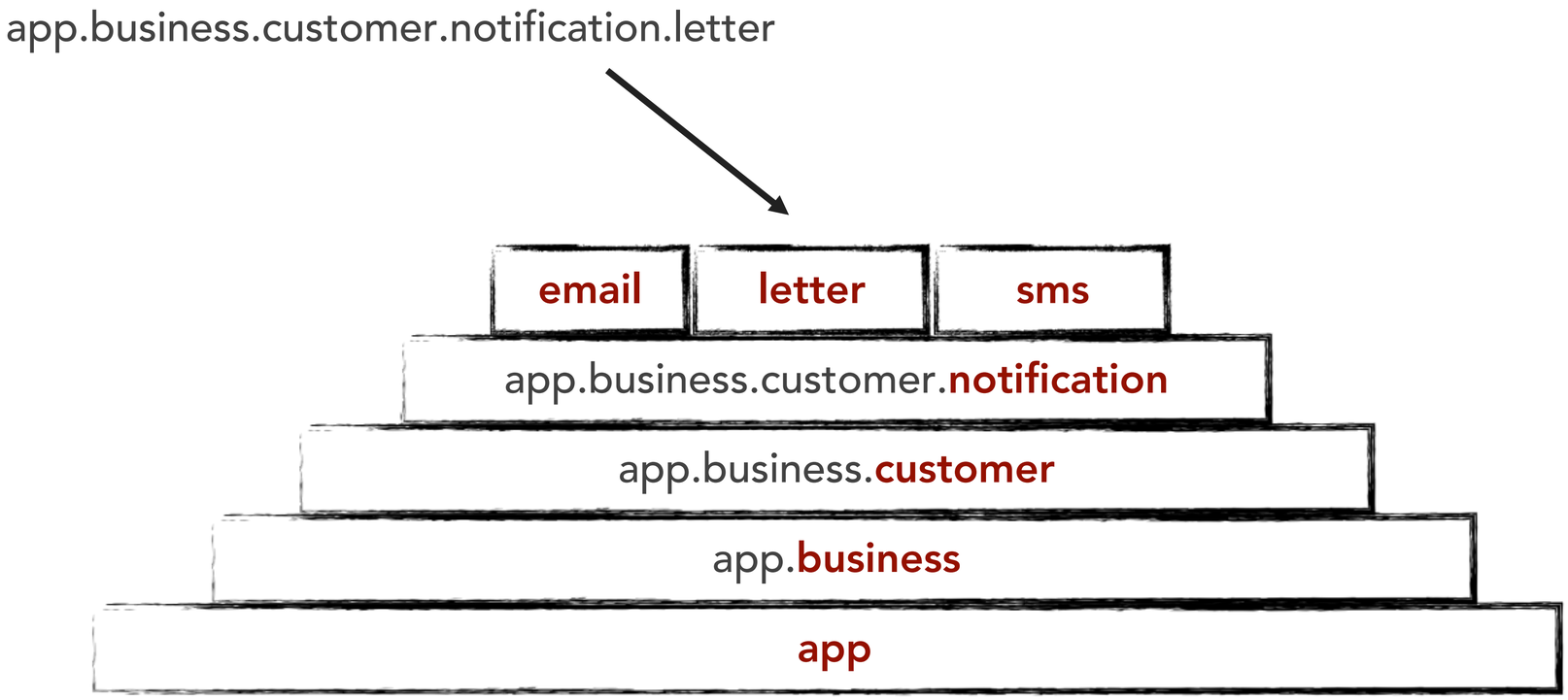
migration patterns

flatten and refactor components



migration patterns

flatten and refactor components



migration patterns

flatten and refactor components

customer info classes

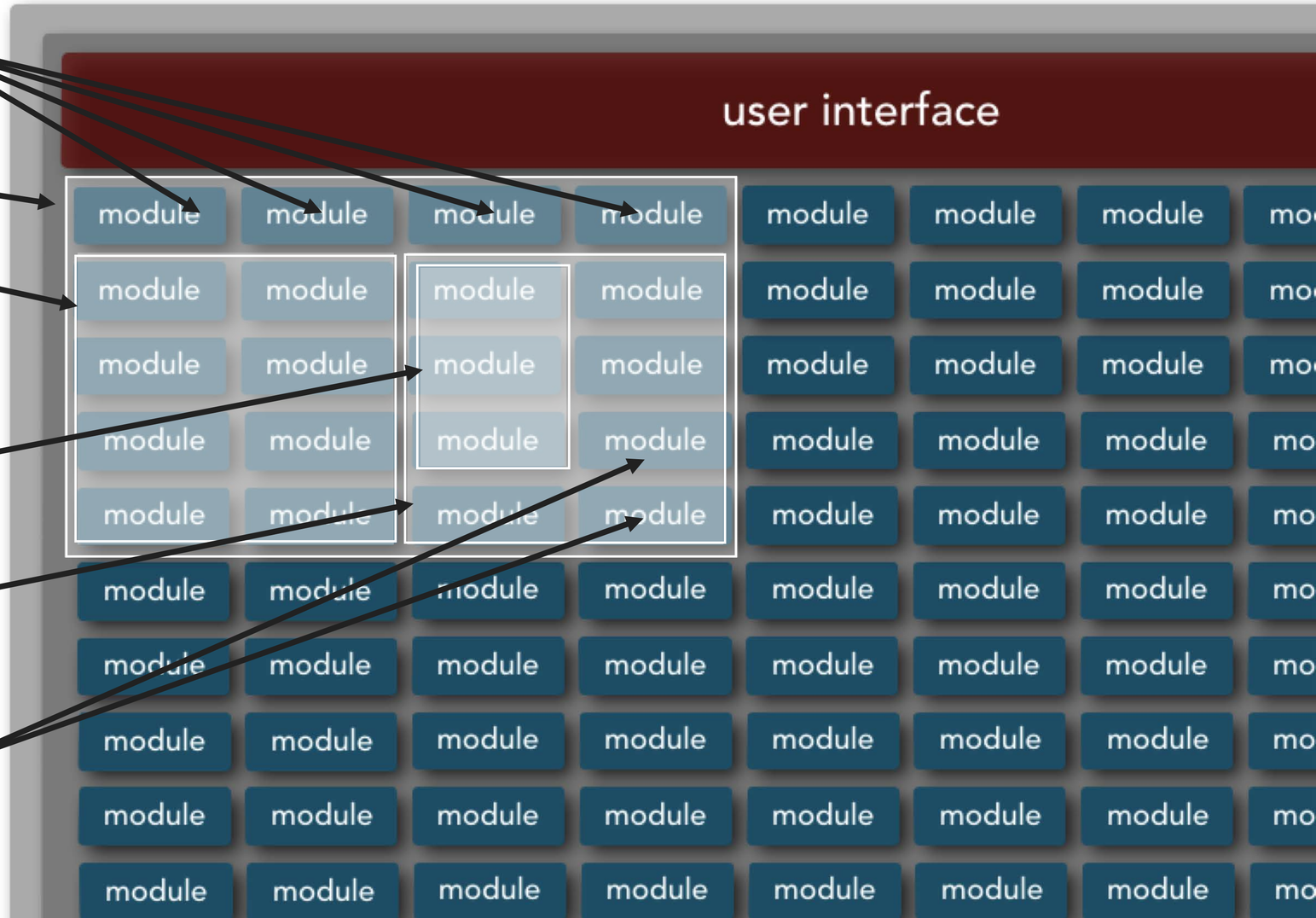
app.customer

app.customer.preferences

app.customer.notification.sms

app.customer.notification

other notification classes
(email, letter, etc.)



migration patterns

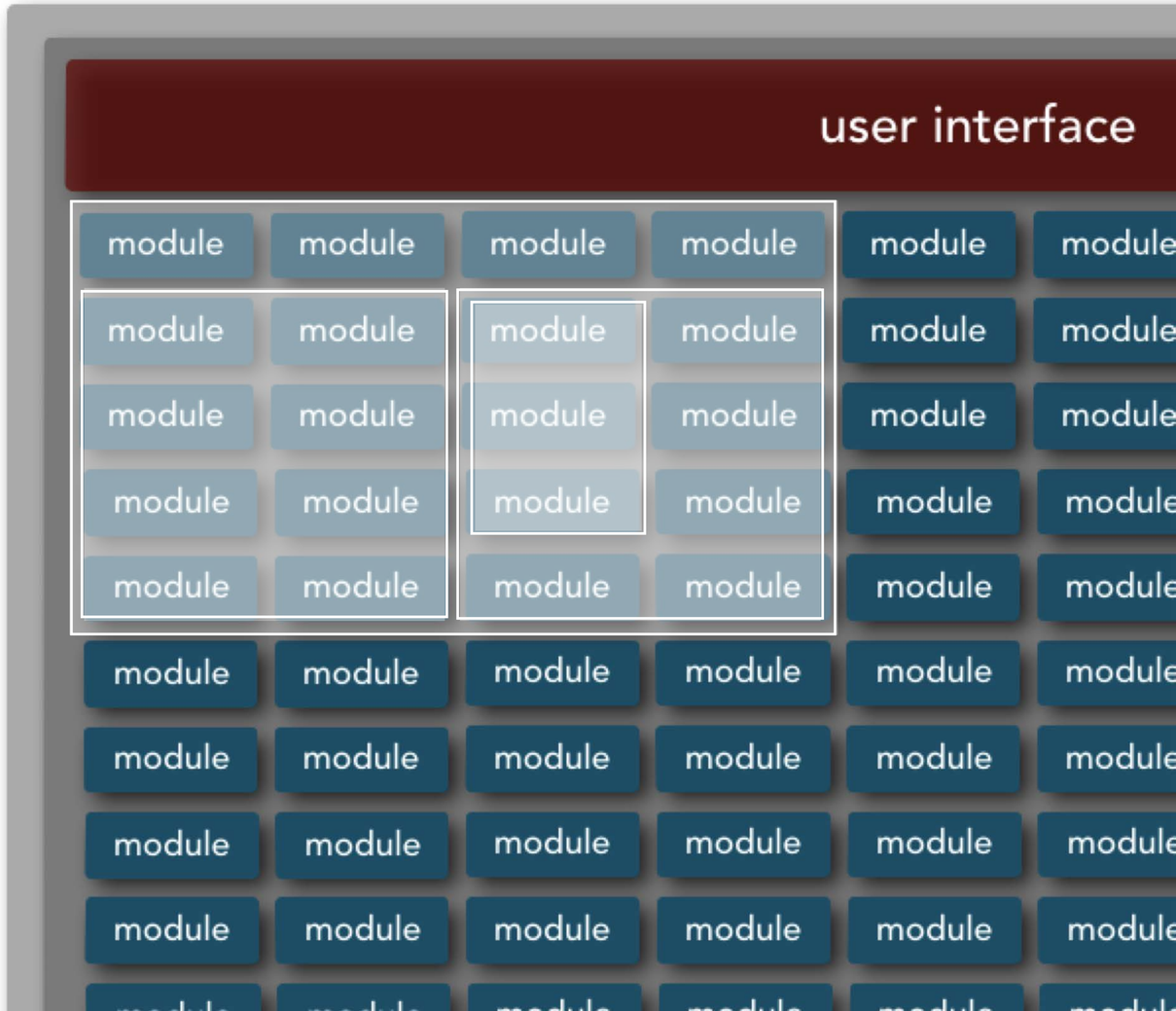
flatten and refactor components

app.customer

app.customer.notification

app.customer.notification.sms

app.customer.preferences



migration patterns

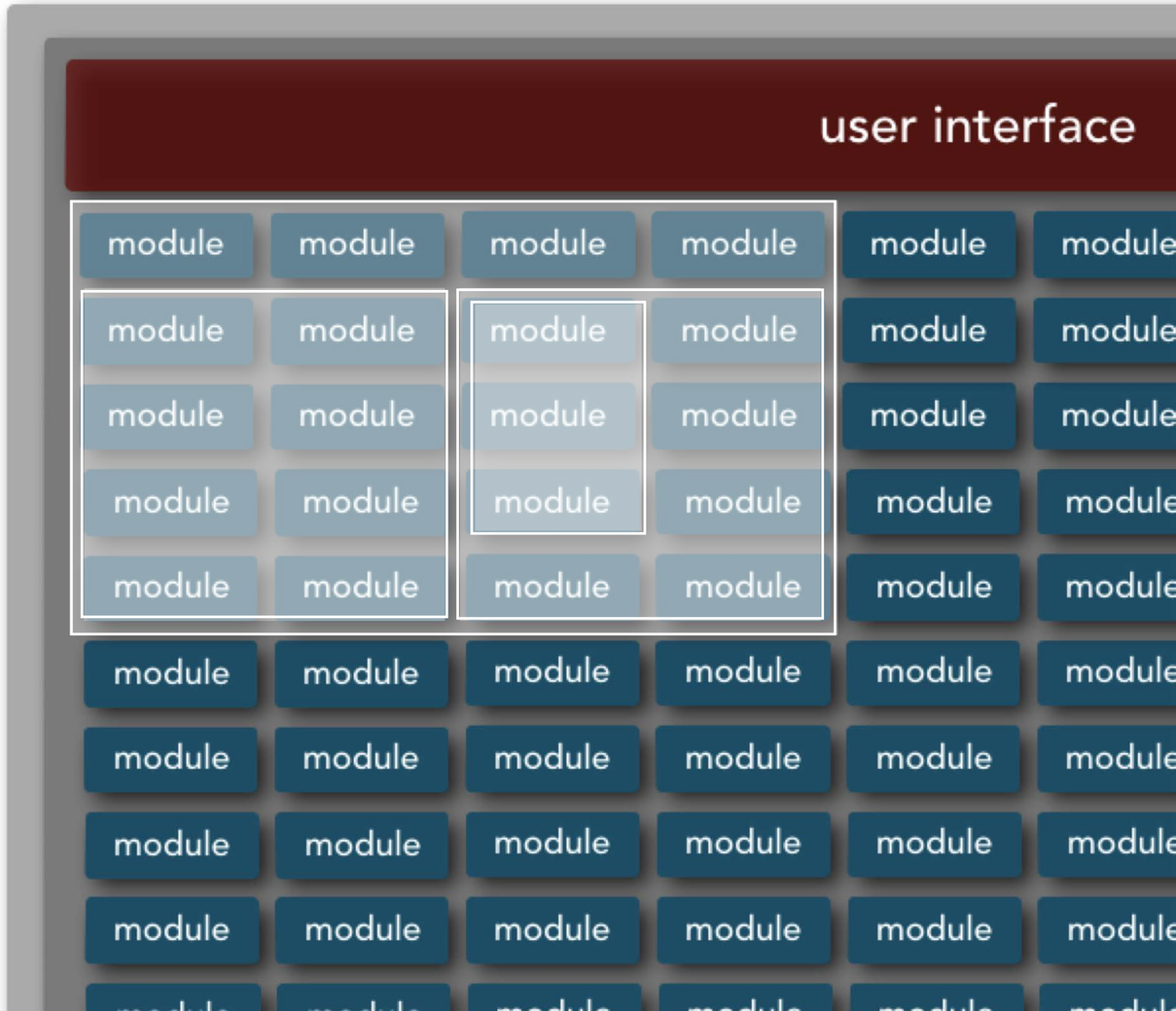
flatten and refactor components

app.customer

app.customer.notification

app.customer.notification.sms

app.customer.preferences



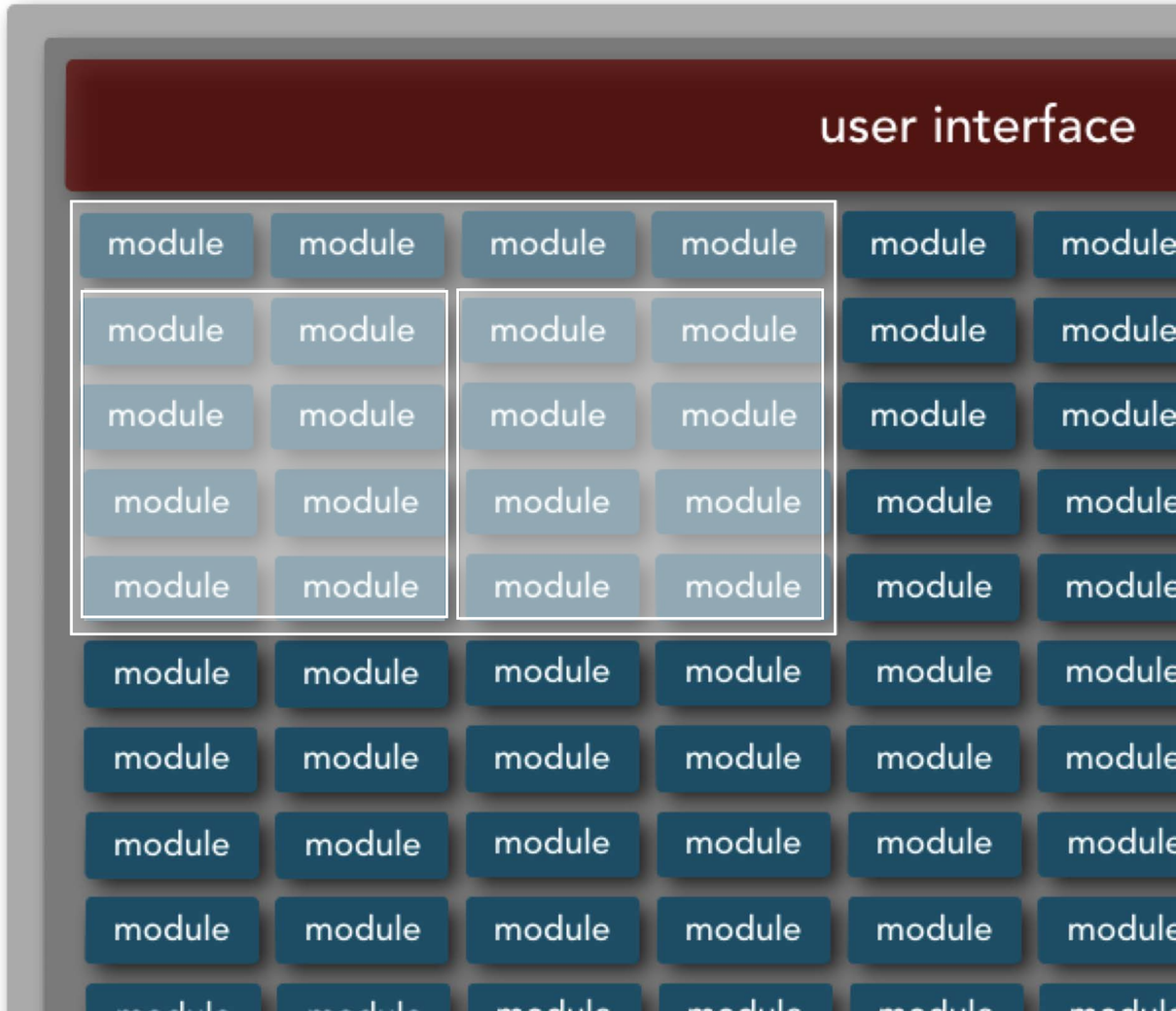
migration patterns

flatten and refactor components

app.customer

app.customer.notification

app.customer.preferences



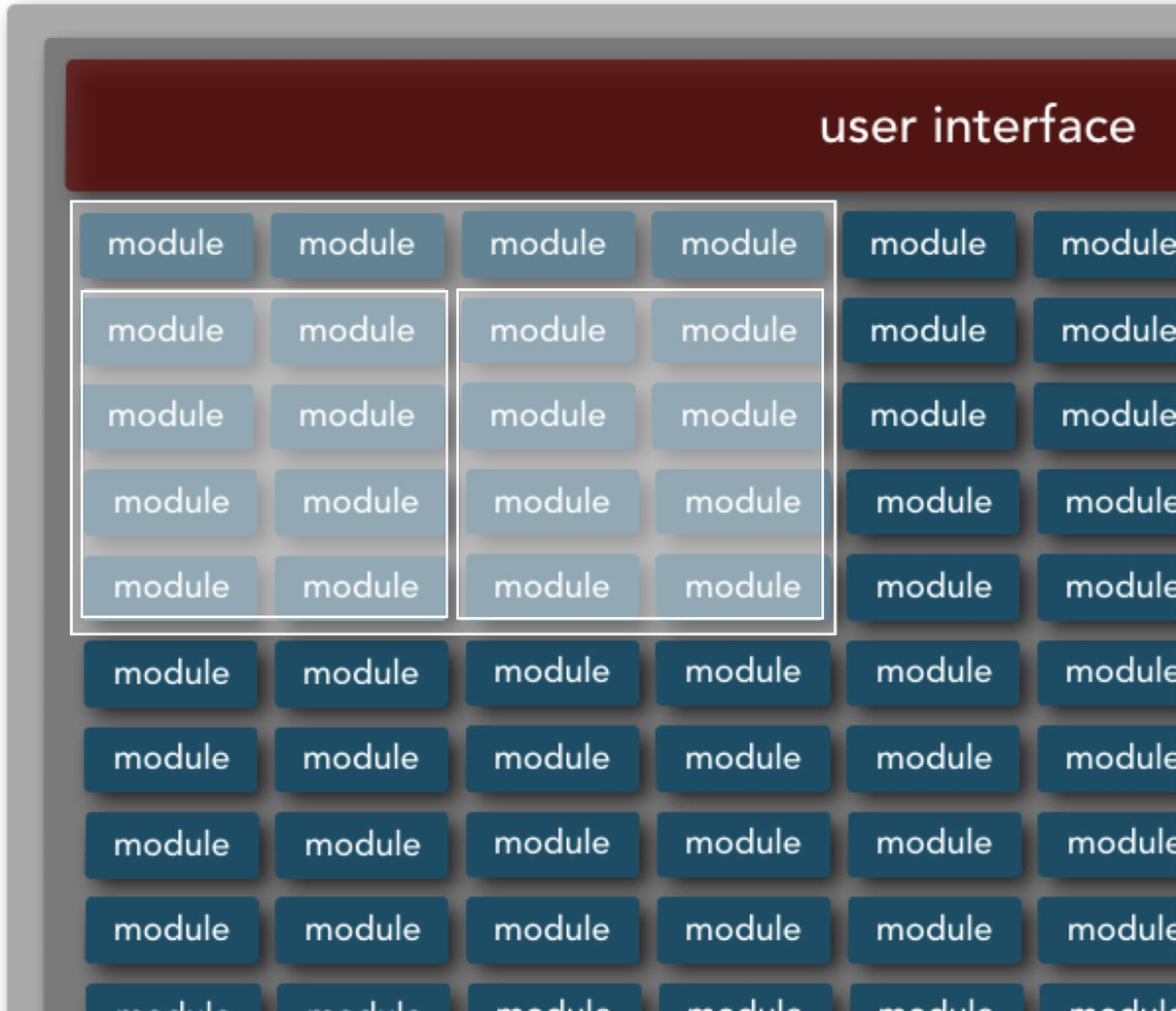
migration patterns

flatten and refactor components

app.customer

app.customer.notification

app.customer.preferences



migration patterns

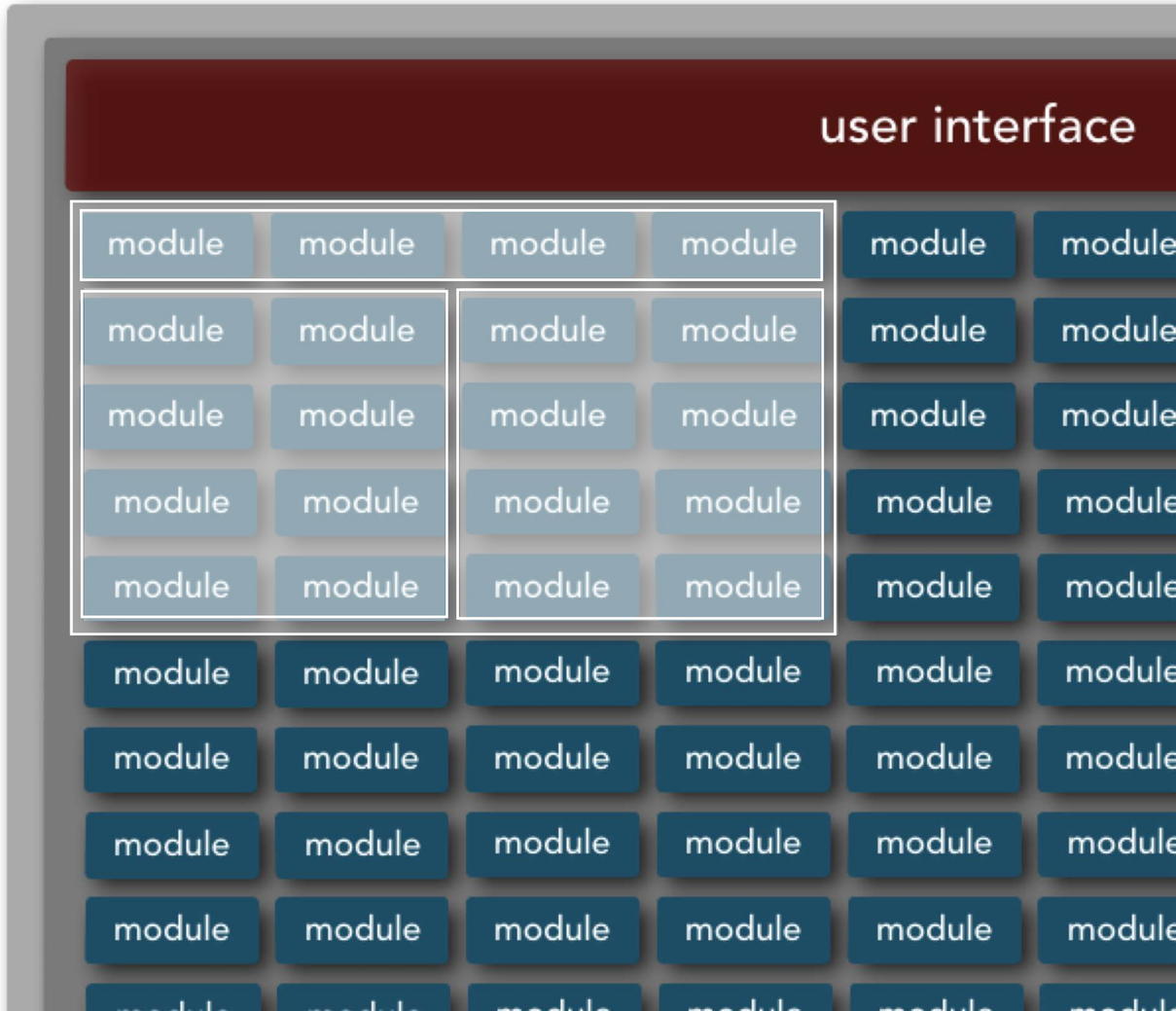
flatten and refactor components

app.customer

app.customer.notification

app.customer.preferences

app.customer.info



migration patterns

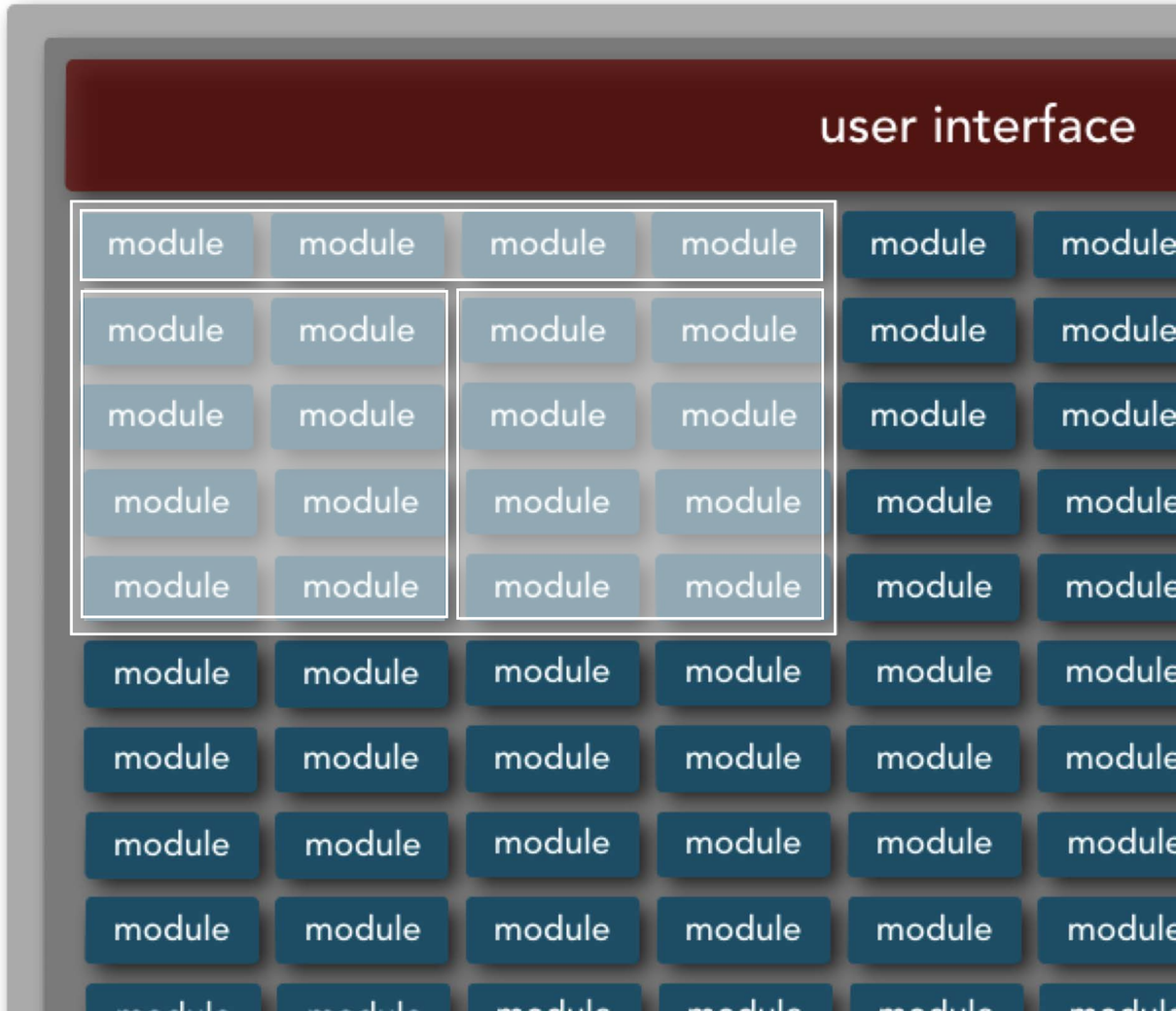
flatten and refactor components

app.customer

app.customer.notification

app.customer.preferences

app.customer.info



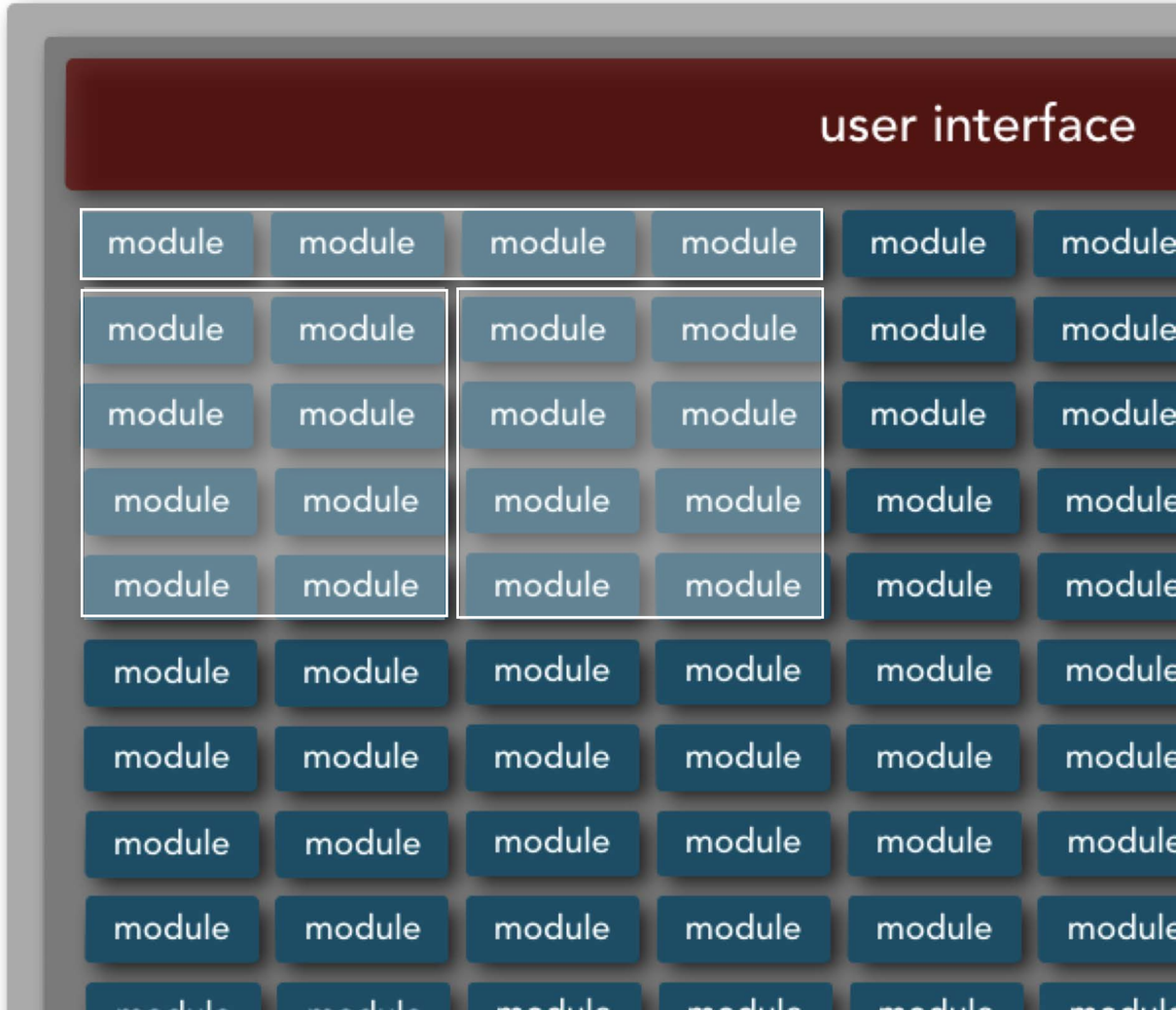
migration patterns

flatten and refactor components

app.customer.notification

app.customer.preferences

app.customer.info

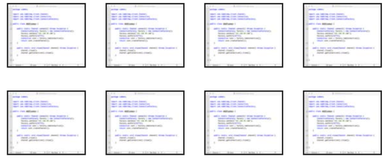


migration patterns

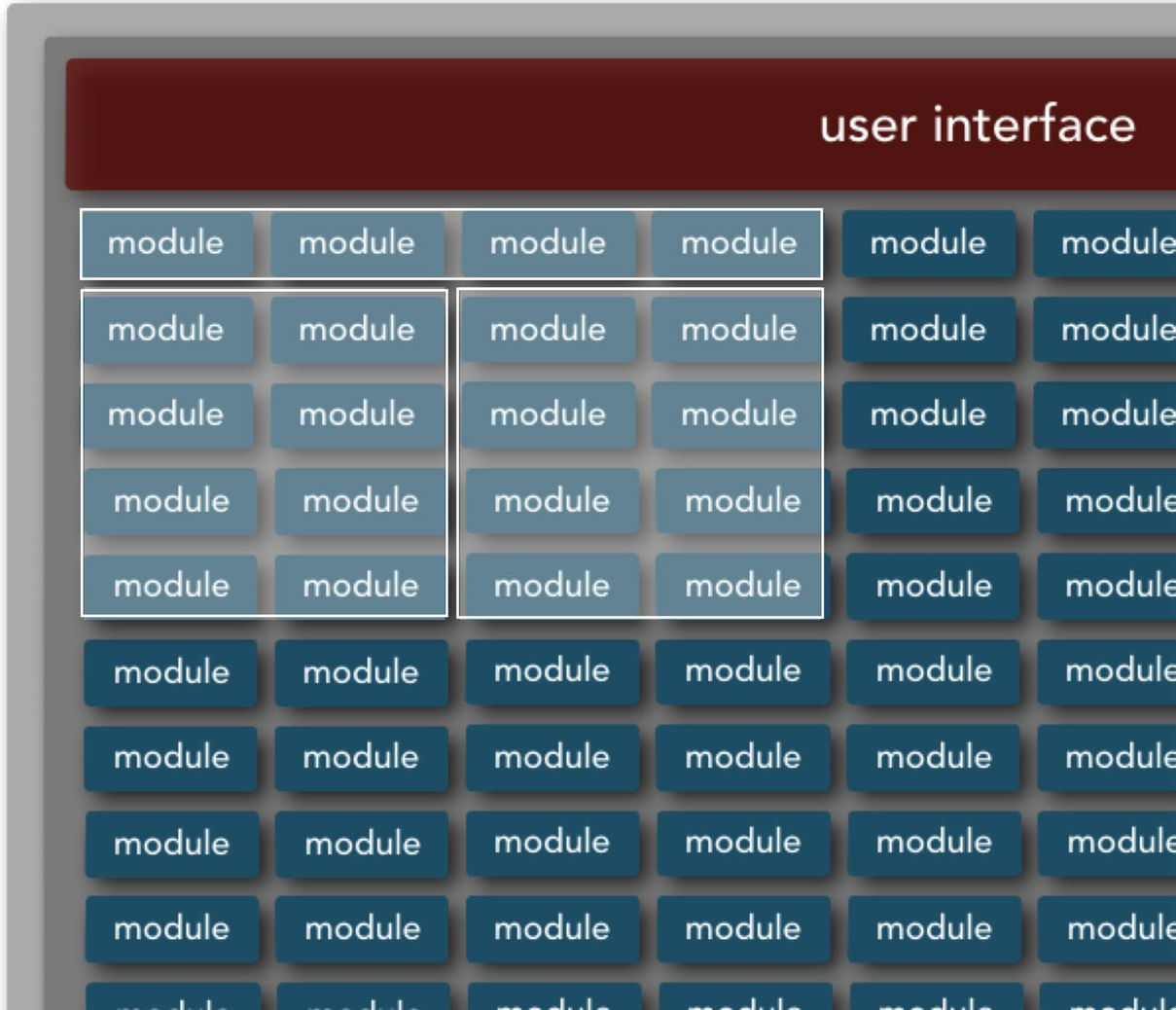
flatten and refactor components

app.customer.notification

app.customer.preferences



app.customer.info



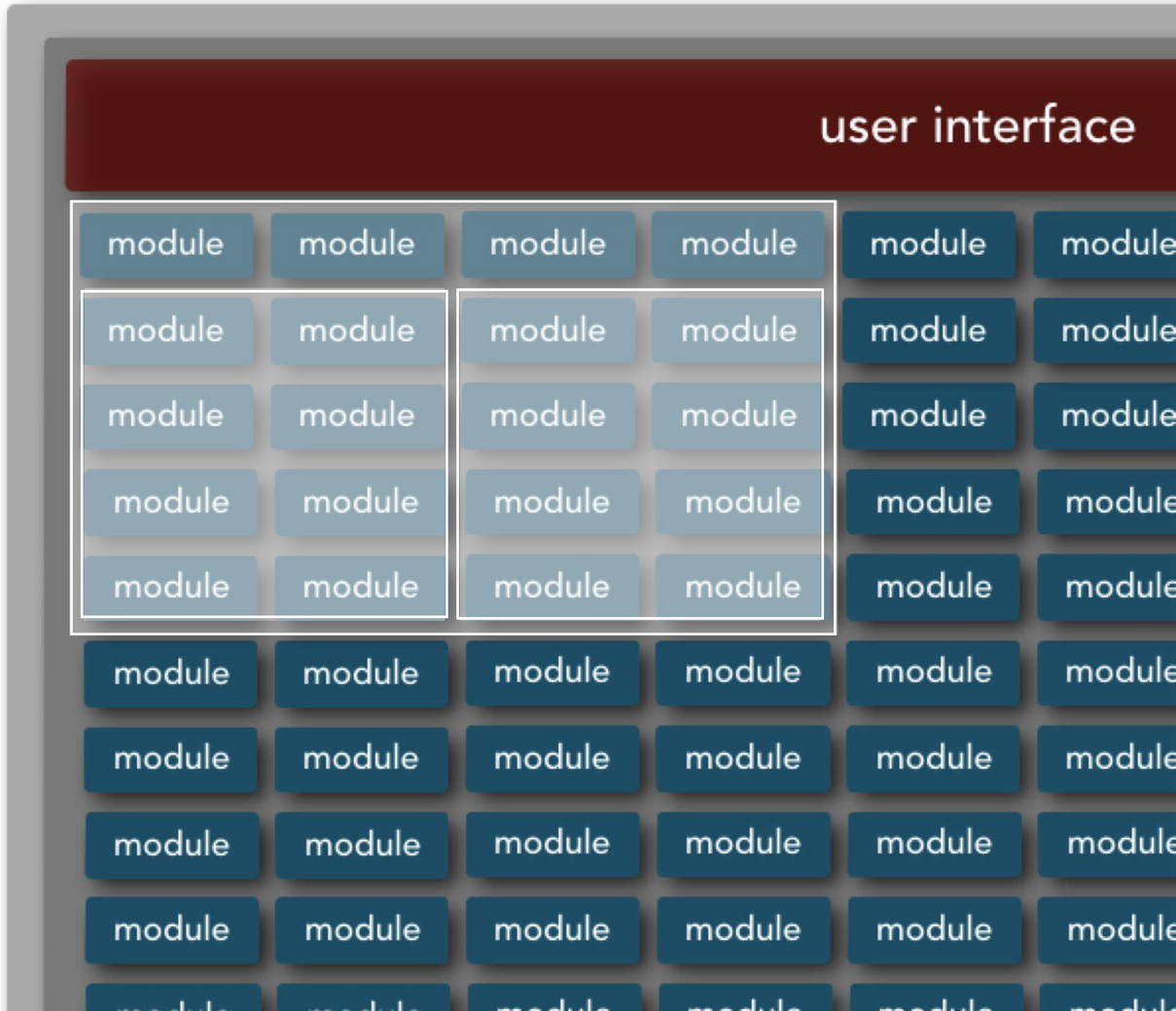
migration patterns

flatten and refactor components

app.customer

app.customer.notification

app.customer.preferences



migration patterns

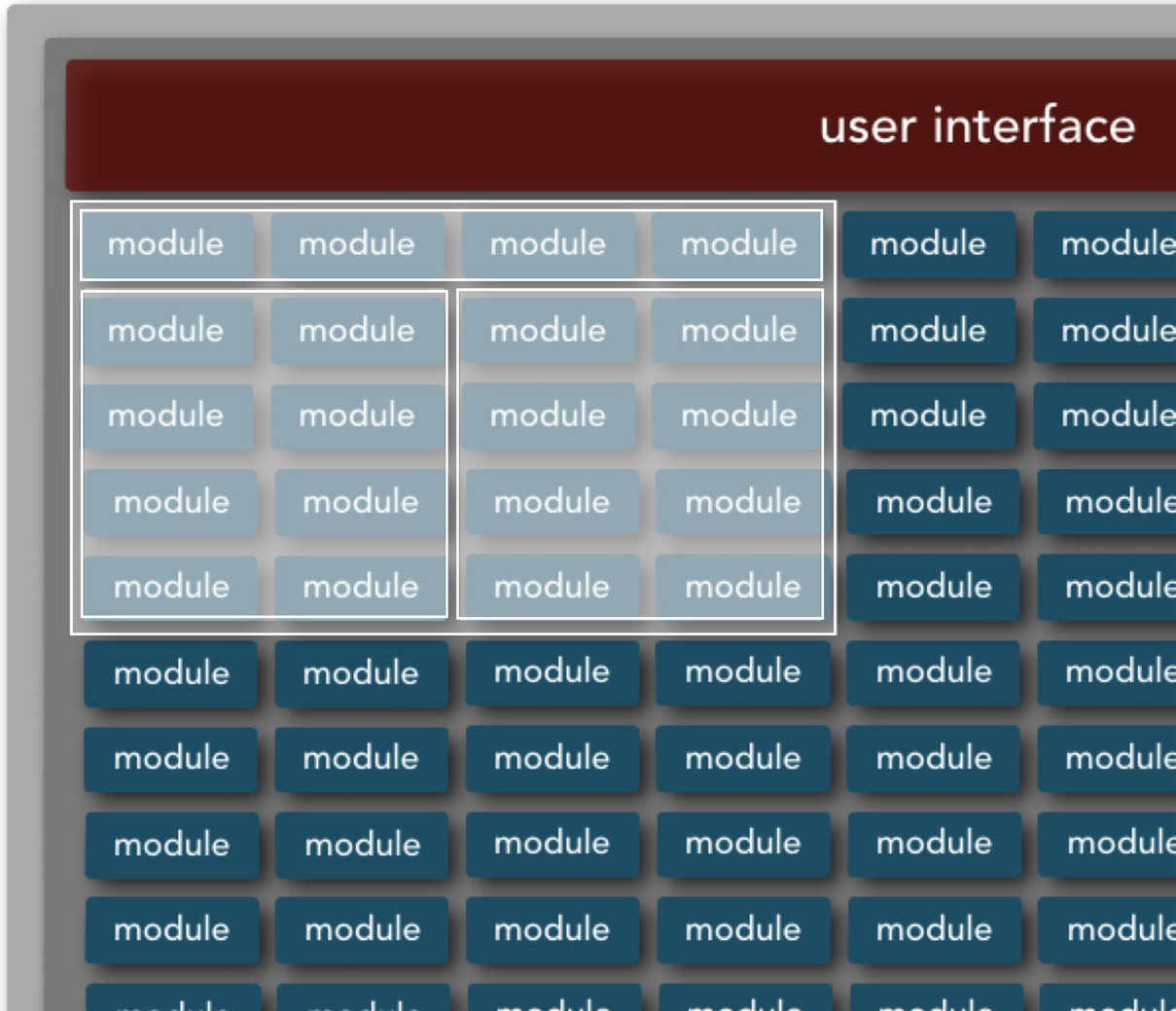
flatten and refactor components

app.customer

app.customer.notification

app.customer.preferences

app.customer.shared



migration patterns

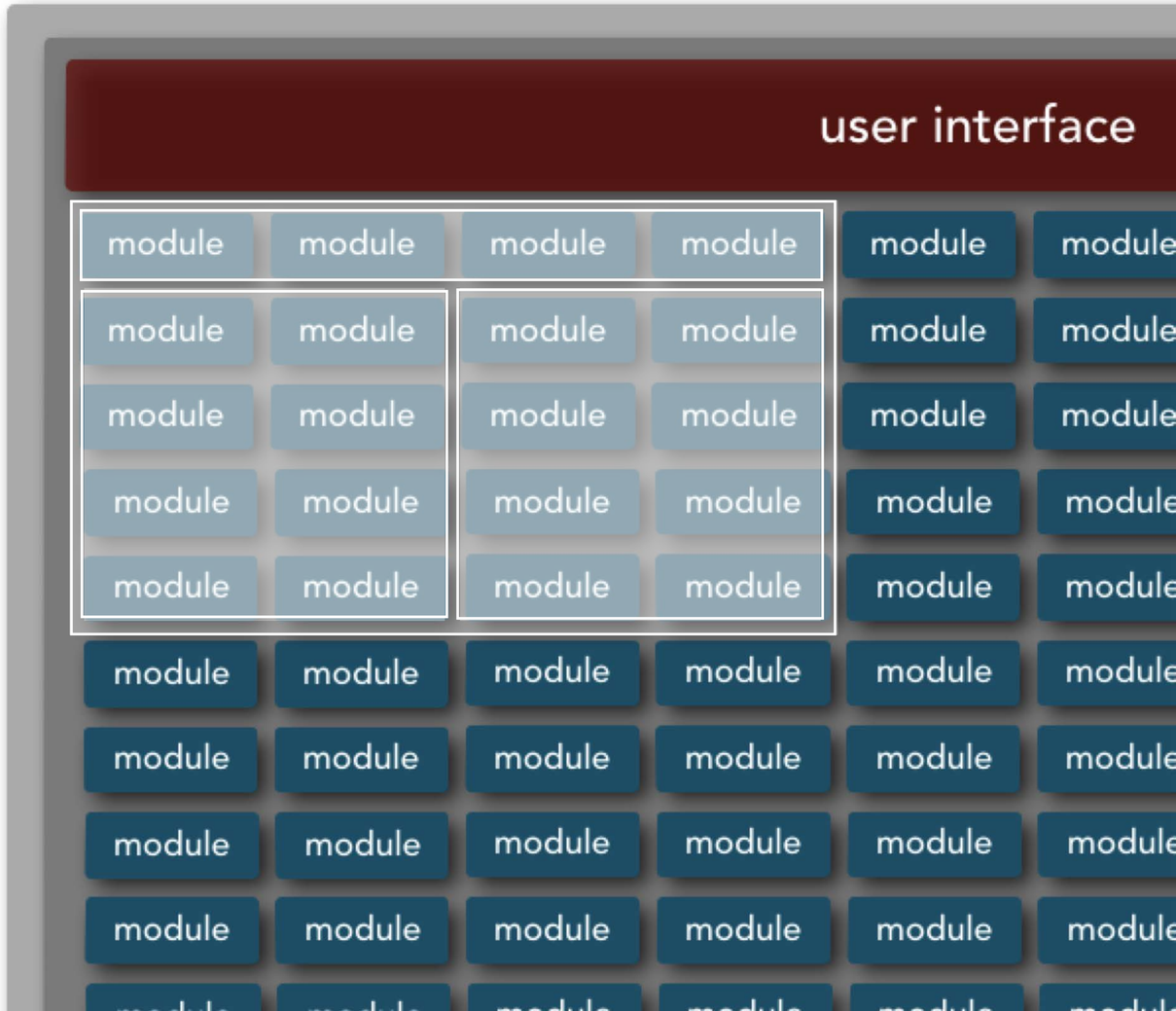
flatten and refactor components

app.customer

app.customer.notification

app.customer.preferences

app.customer.shared



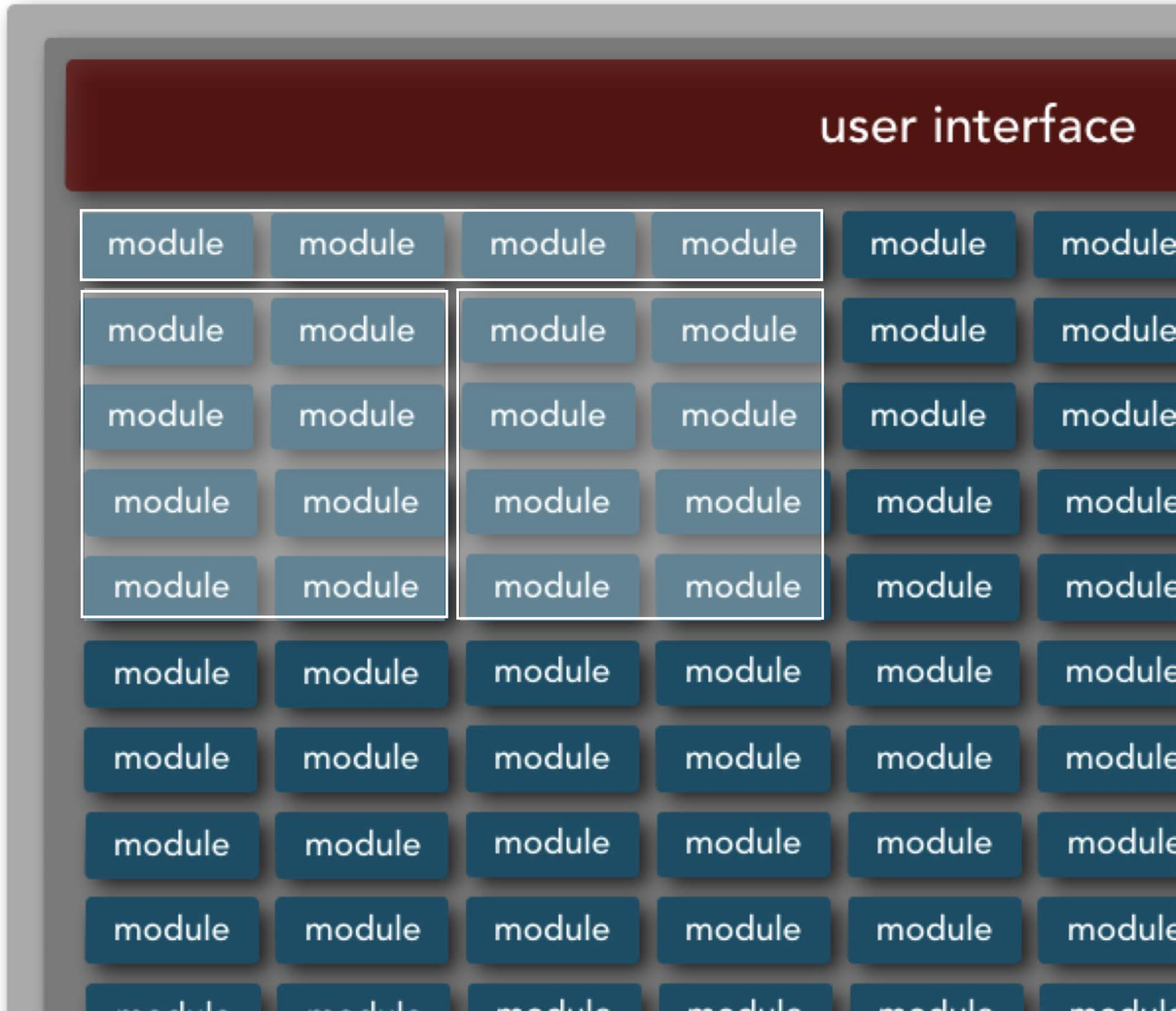
migration patterns

flatten and refactor components

app.customer.notification

app.customer.preferences

app.customer.shared

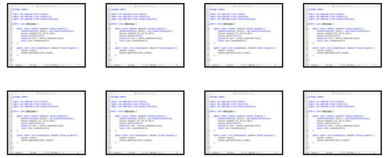


migration patterns

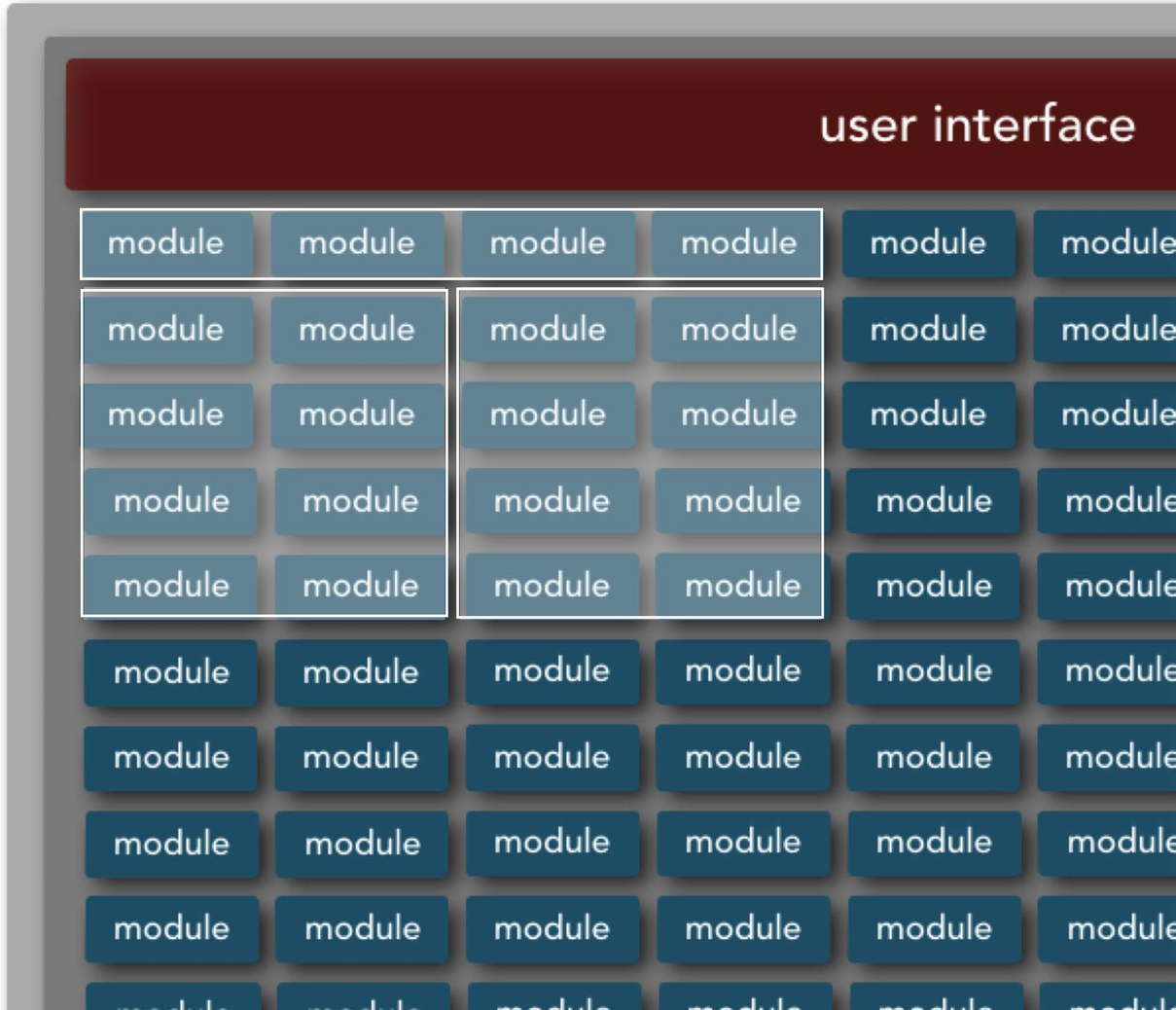
flatten and refactor components

app.customer.notification

app.customer.preferences



app.customer.shared



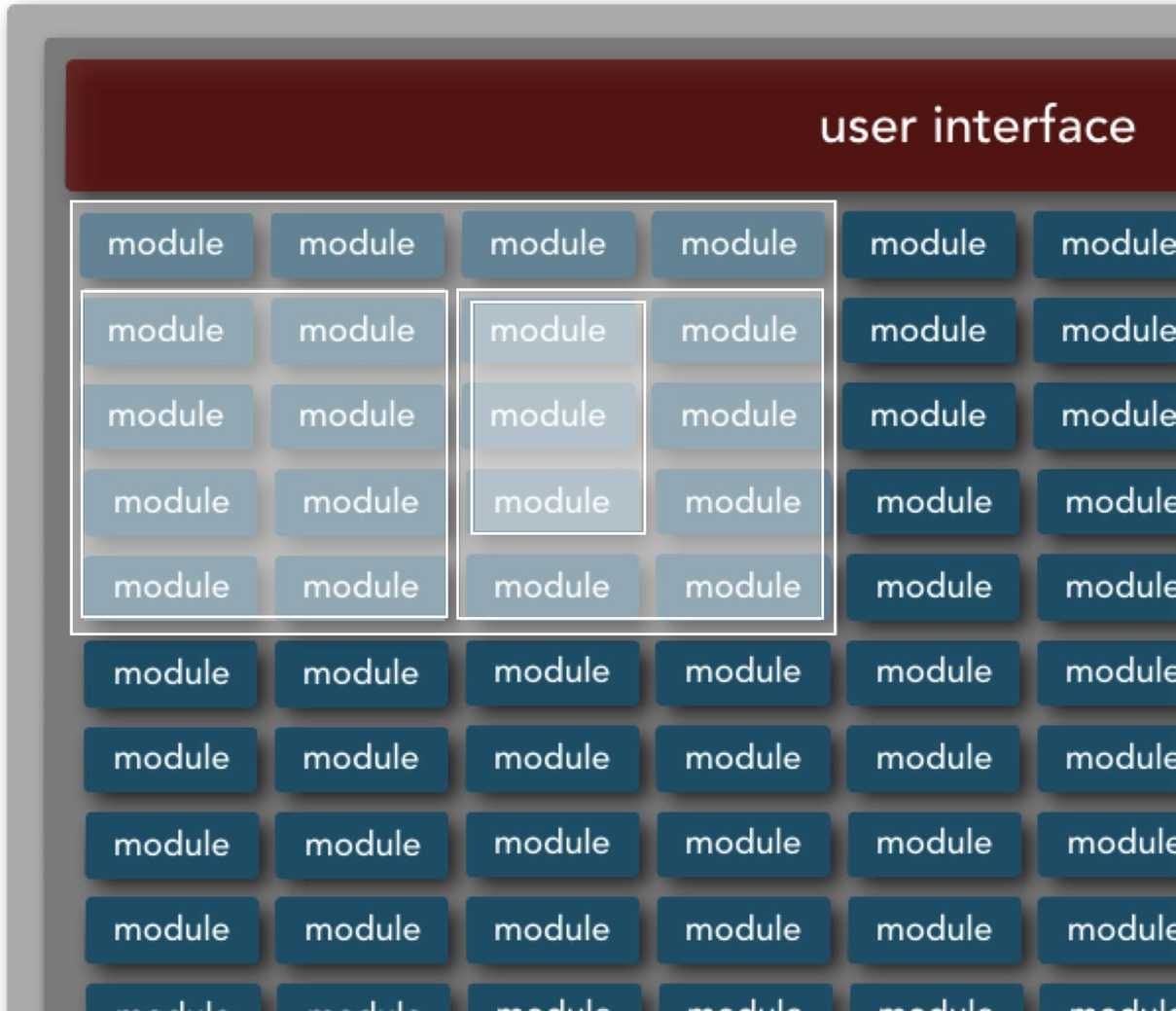
migration patterns

flatten and refactor components

app.customer

app.customer.notification

app.customer.notification.sms



migration patterns

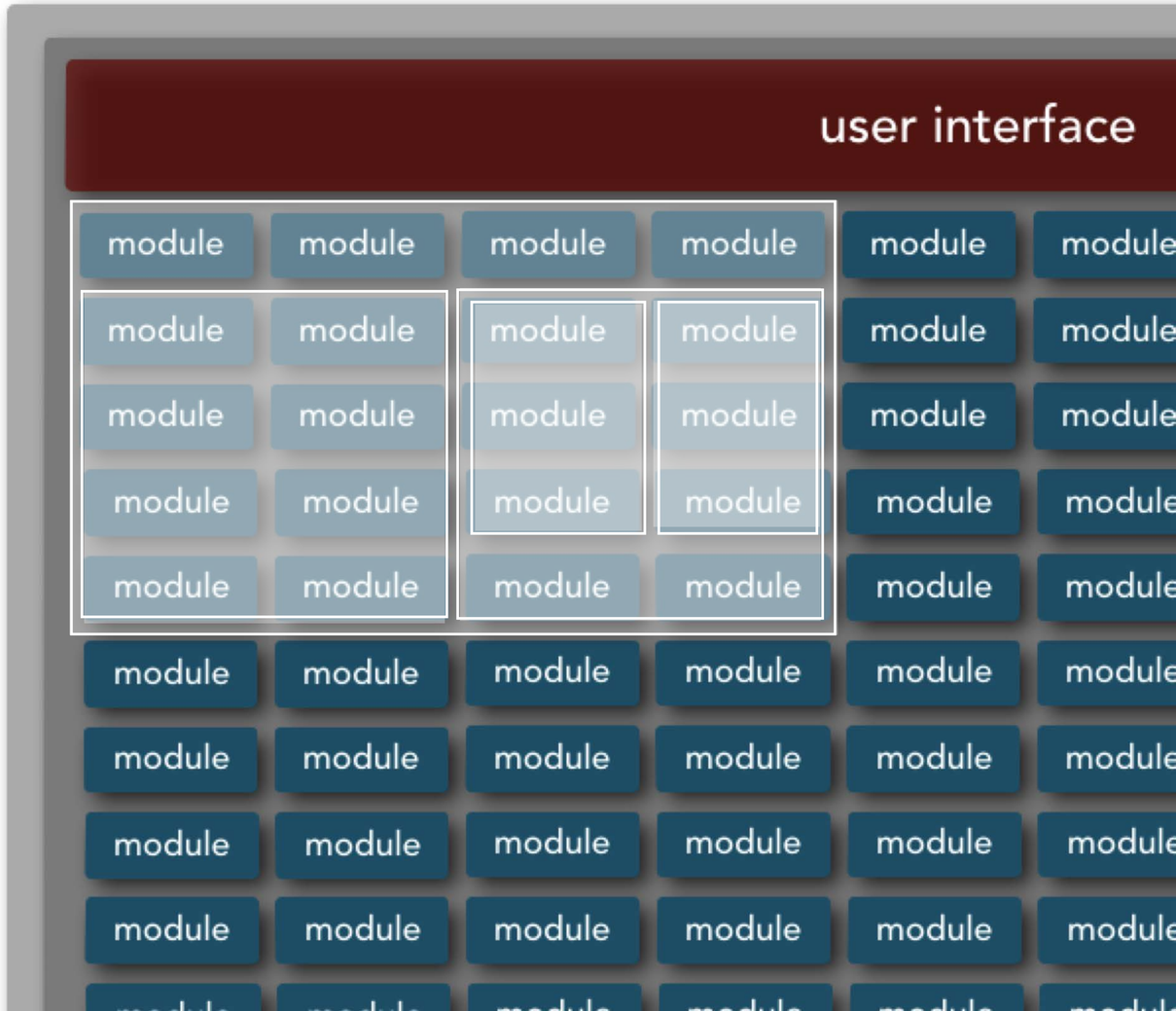
flatten and refactor components

app.customer

app.customer.notification

app.customer.notification.sms

app.customer.notification.letter



migration patterns

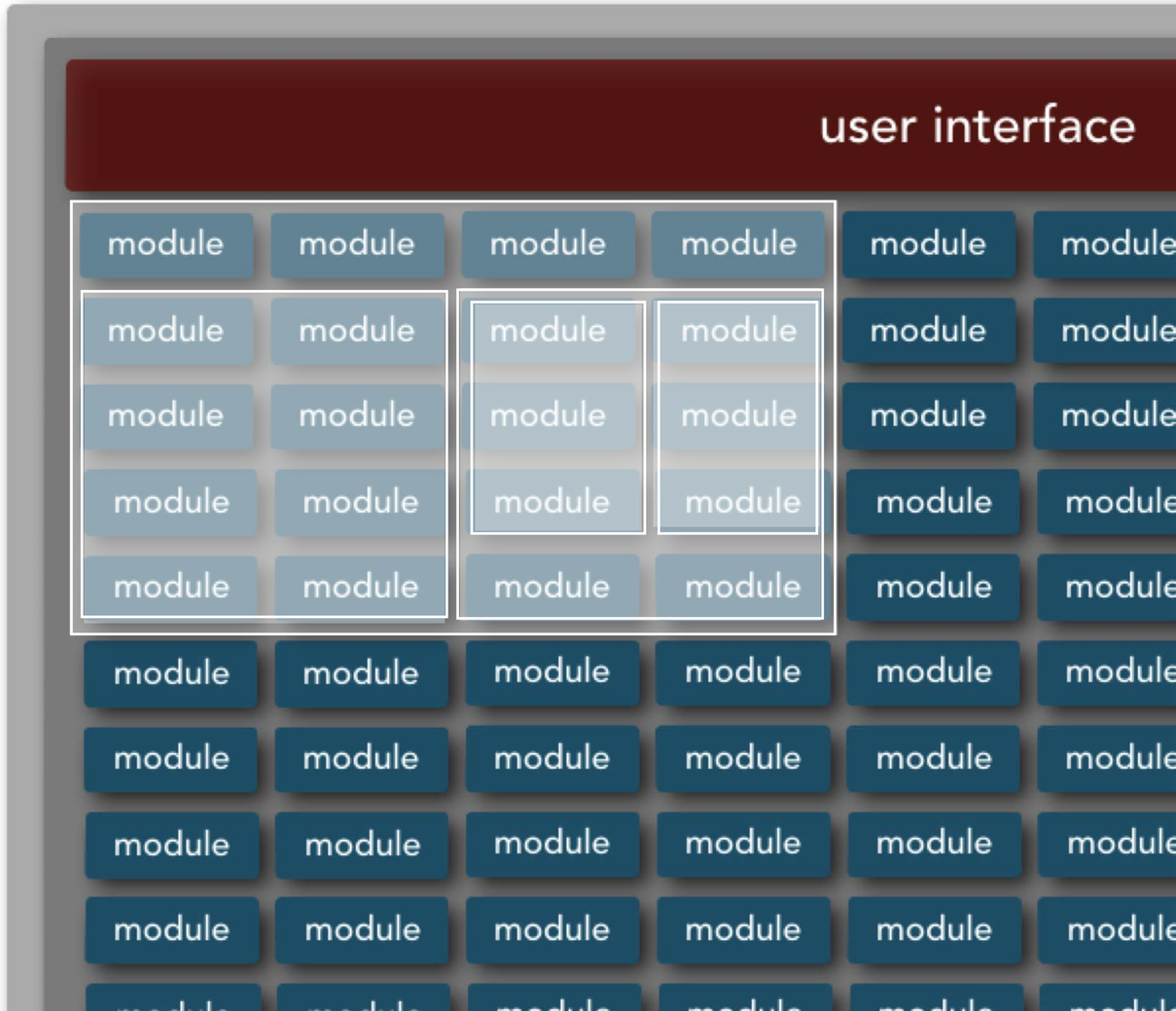
flatten and refactor components

app.customer

app.customer.notification

app.customer.notification.sms

app.customer.notification.letter



migration patterns

flatten and refactor components

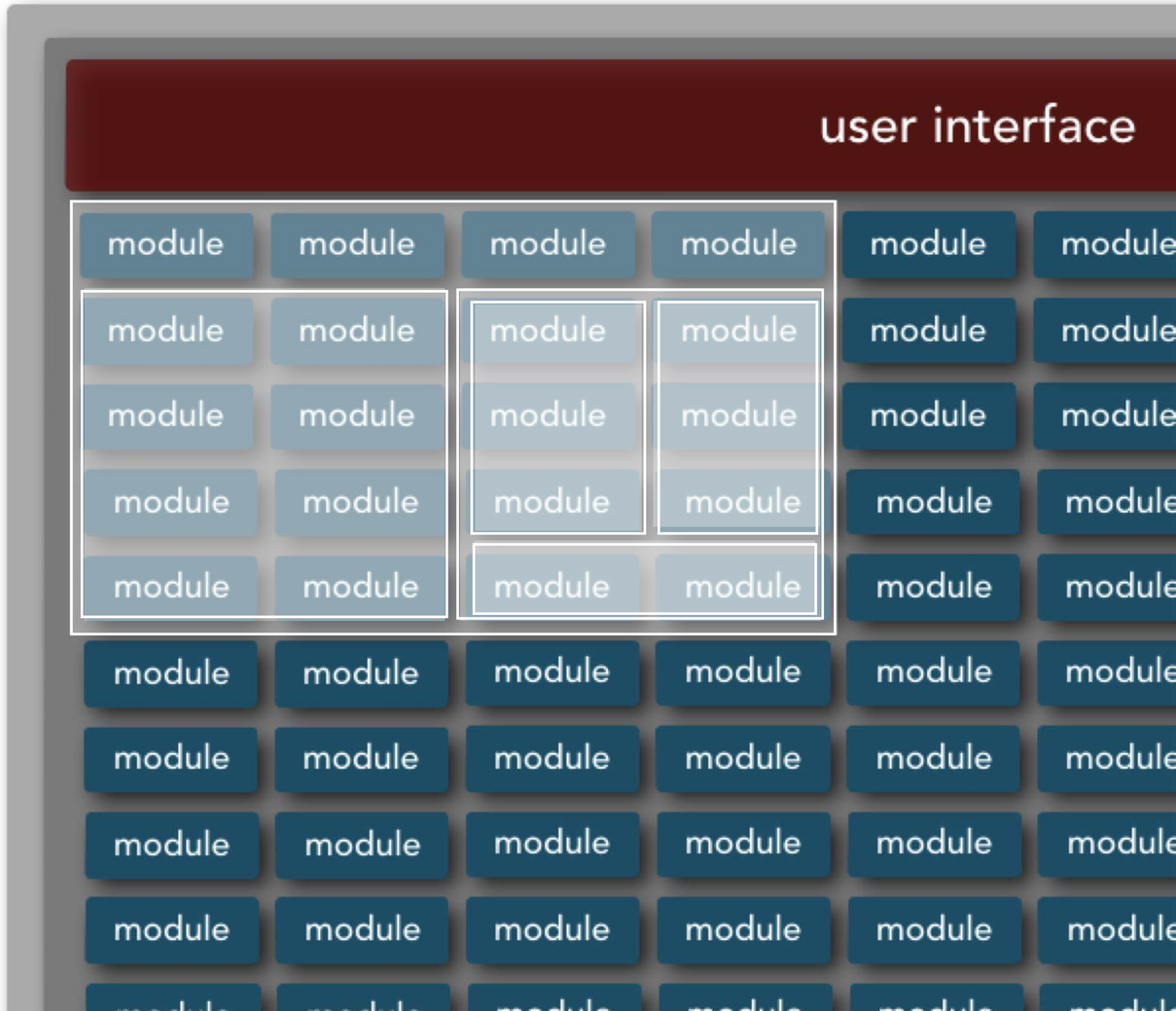
app.customer

app.customer.notification

app.customer.notification.sms

app.customer.notification.letter

app.customer.notification.email



migration patterns

flatten and refactor components

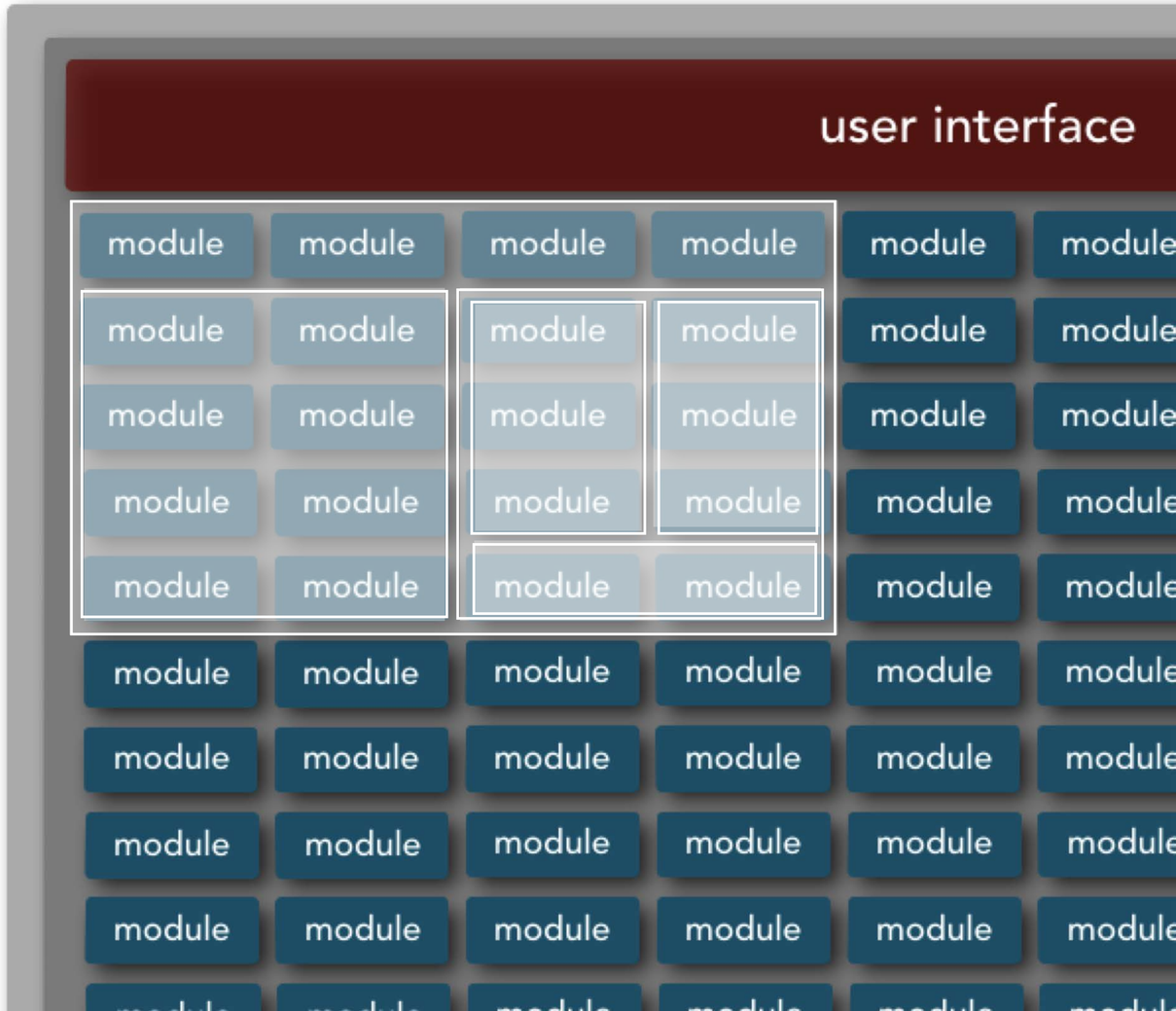
app.customer

app.customer.notification

app.customer.notification.sms

app.customer.notification.letter

app.customer.notification.email



migration patterns

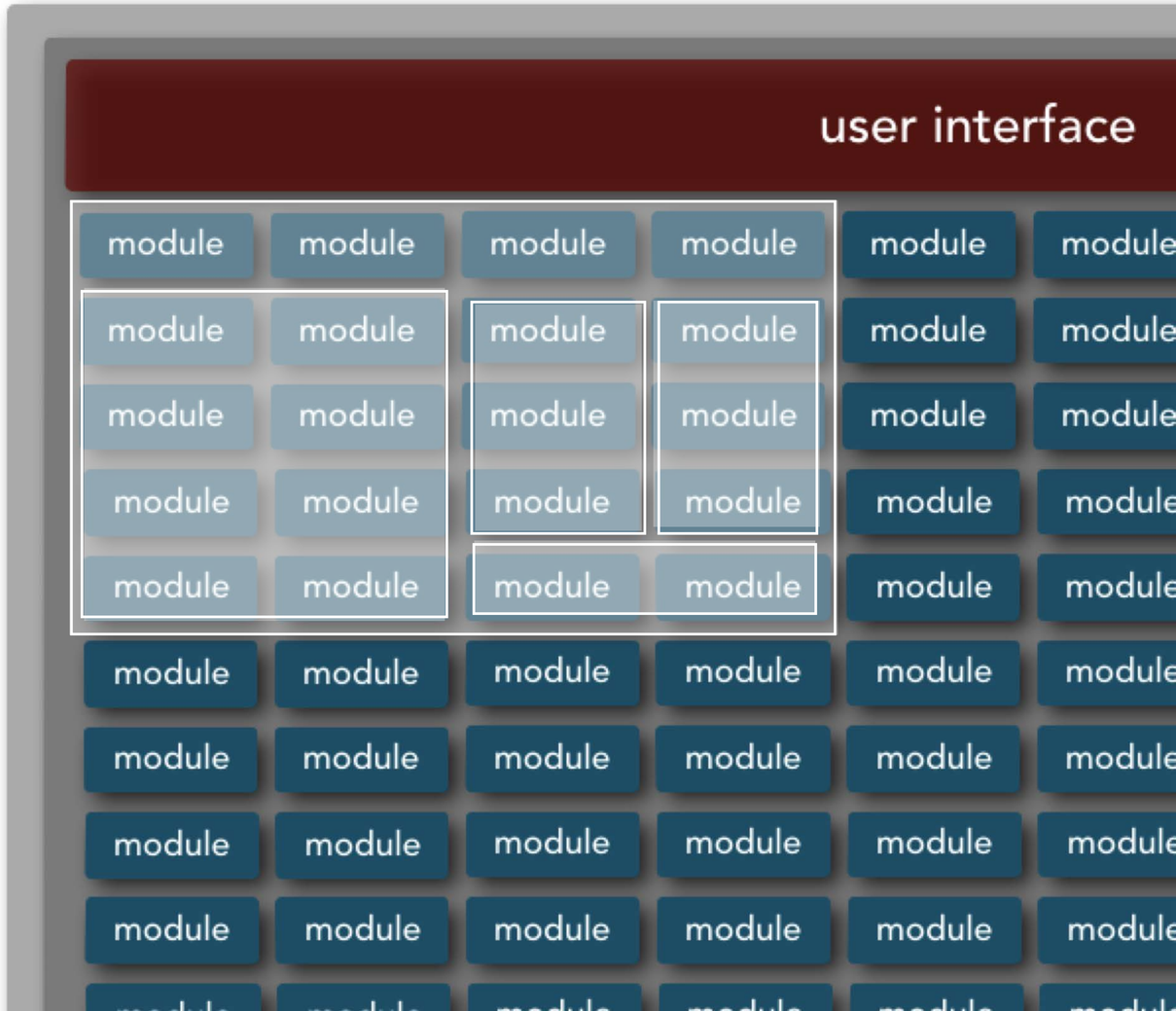
flatten and refactor components

app.customer

app.customer.notification.sms

app.customer.notification.letter

app.customer.notification.email



migration patterns

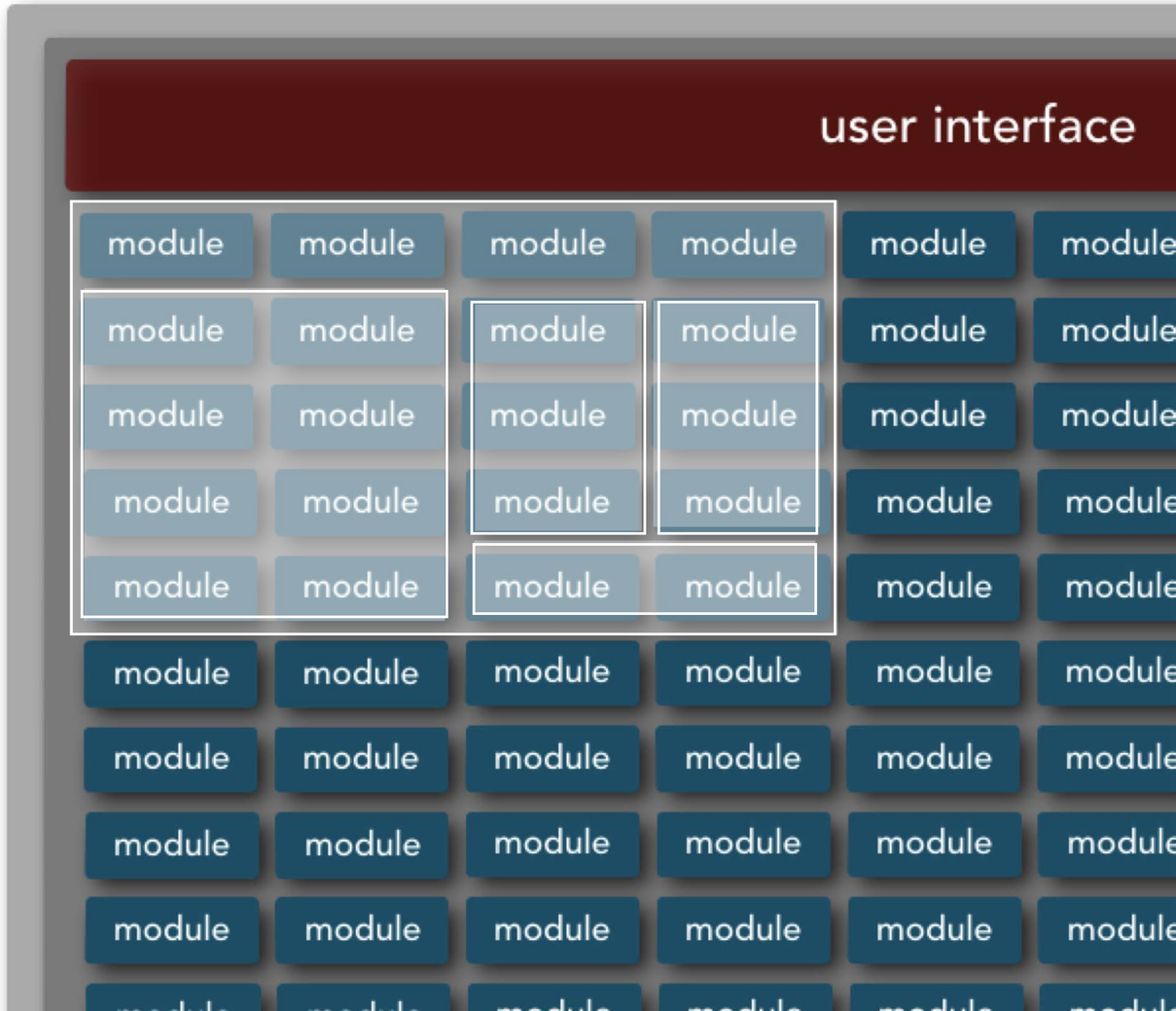
flatten and refactor components

app.customer

app.customer.notification.sms

app.customer.notification.letter

app.customer.notification.email



migration patterns

flatten and refactor components

customer info classes

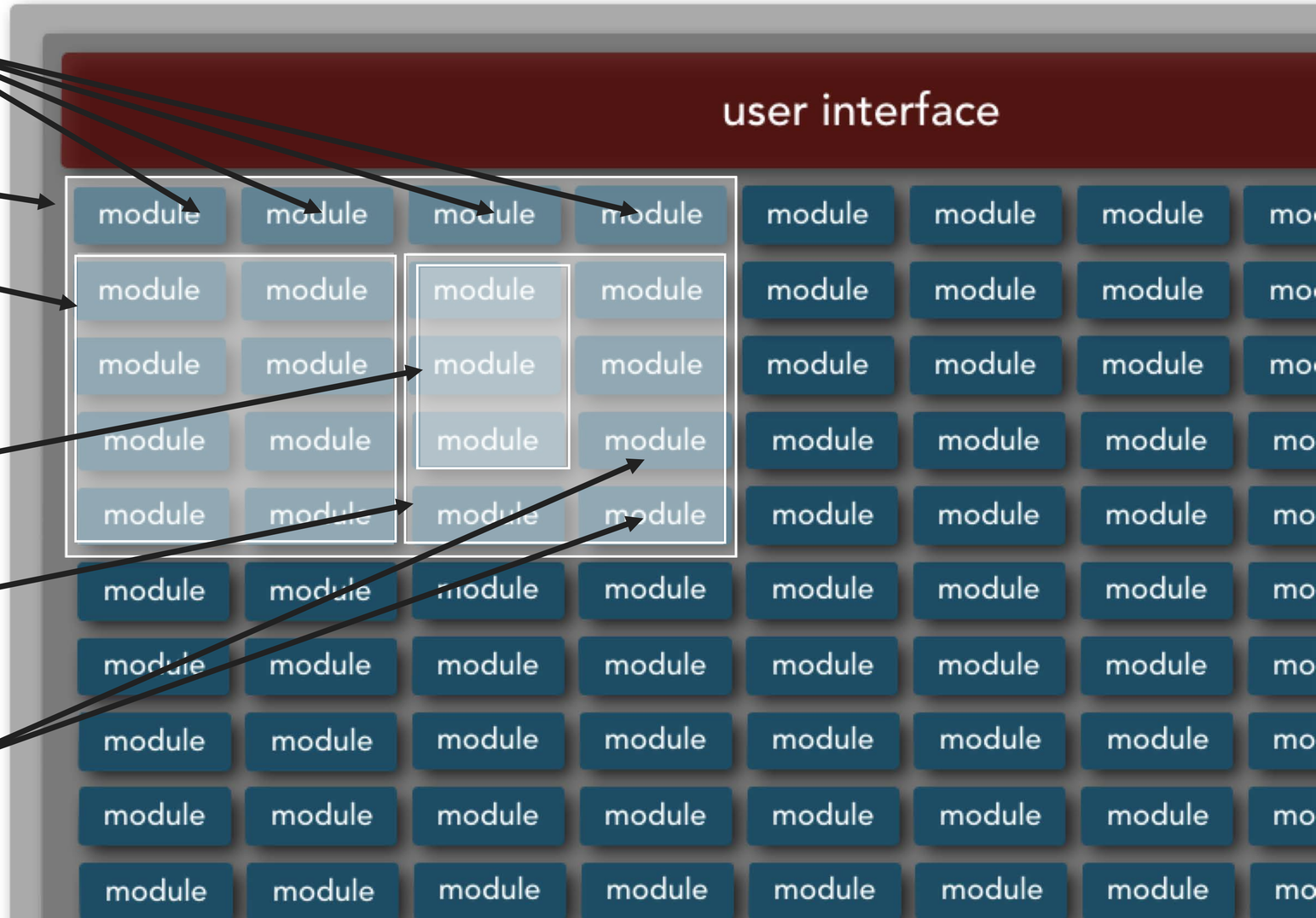
app.customer

app.customer.preferences

app.customer.notification.sms

app.customer.notification

other notification classes
(email, letter, etc.)



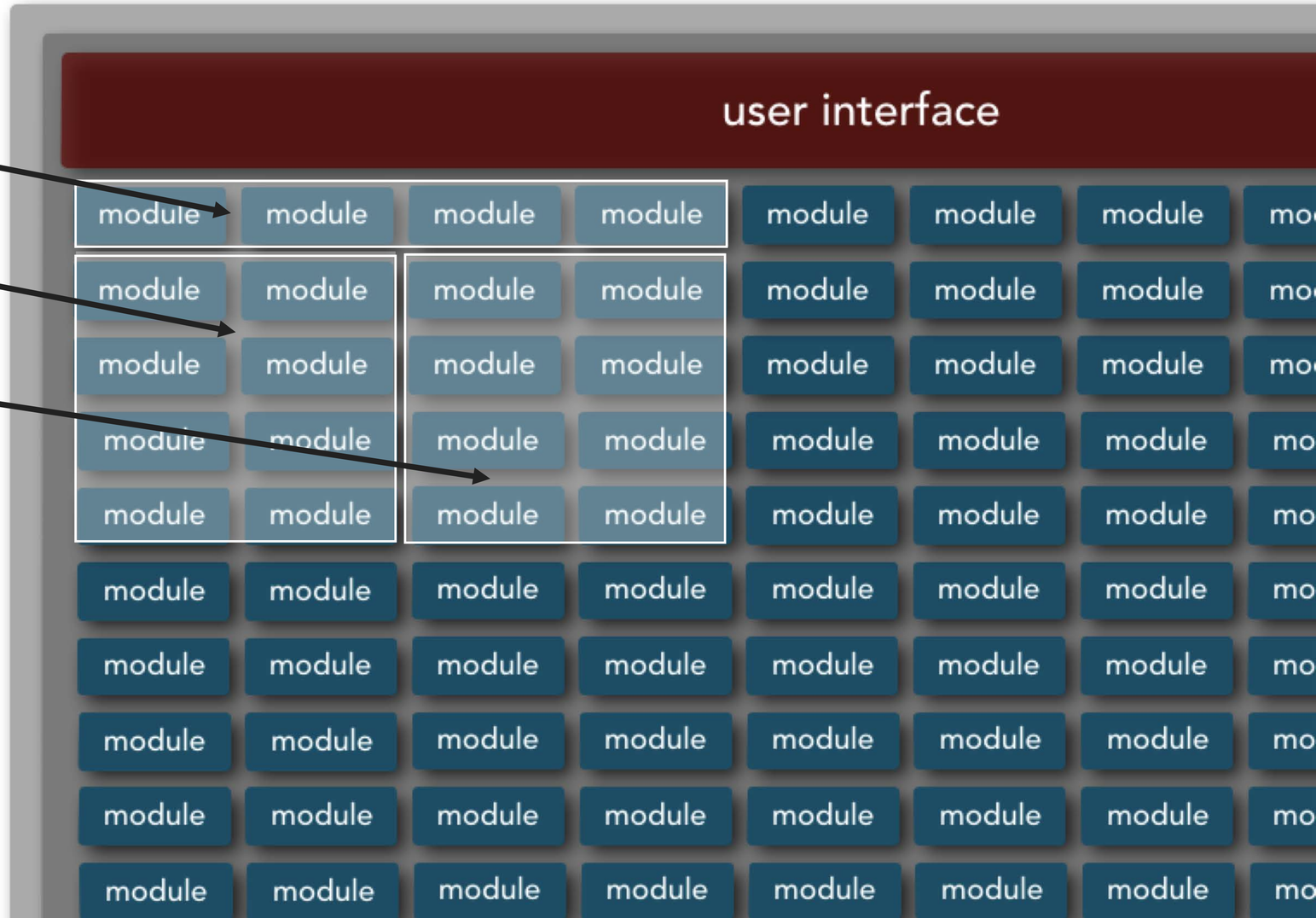
migration patterns

flatten and refactor components

app.customer.info

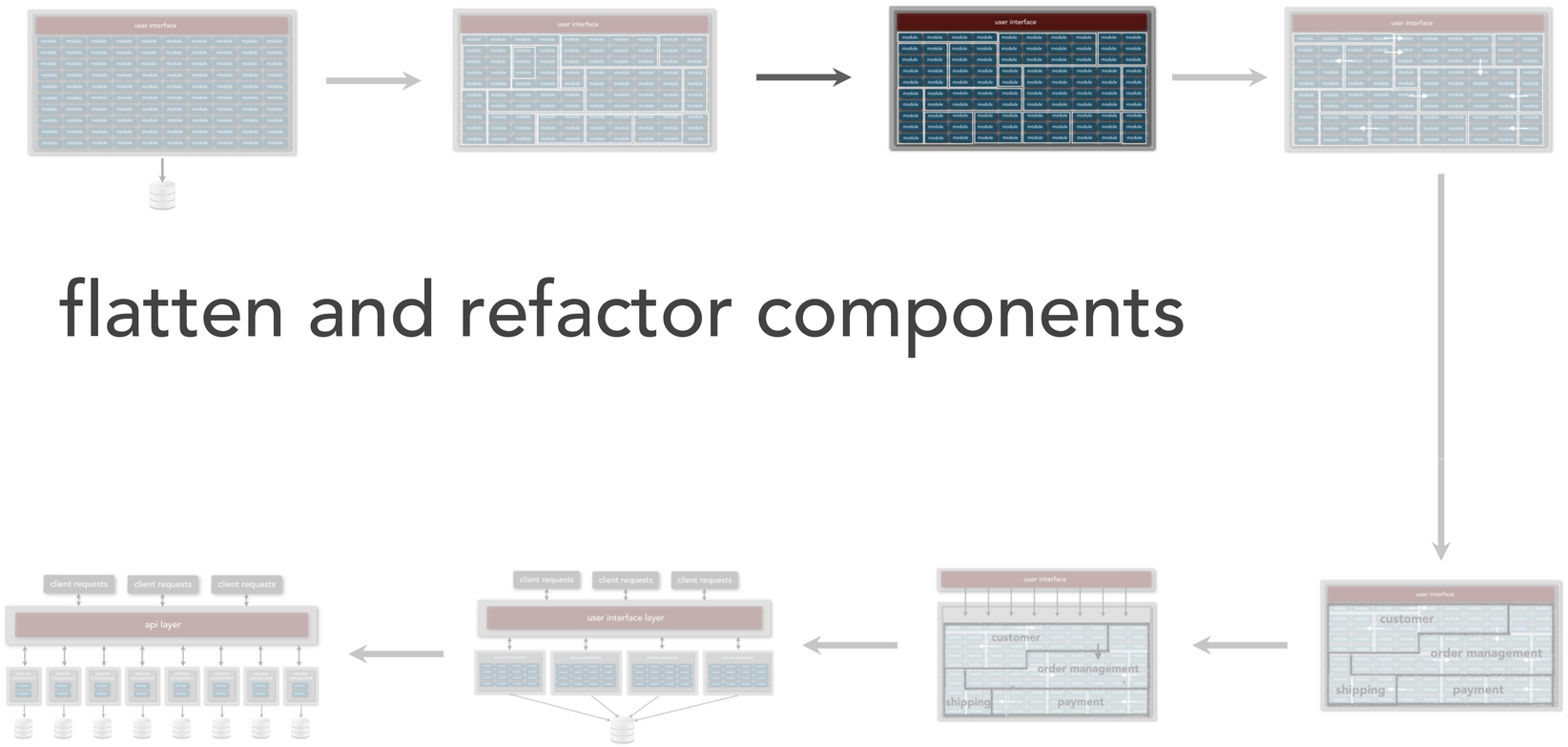
app.customer.preferences

app.customer.notification



migration patterns

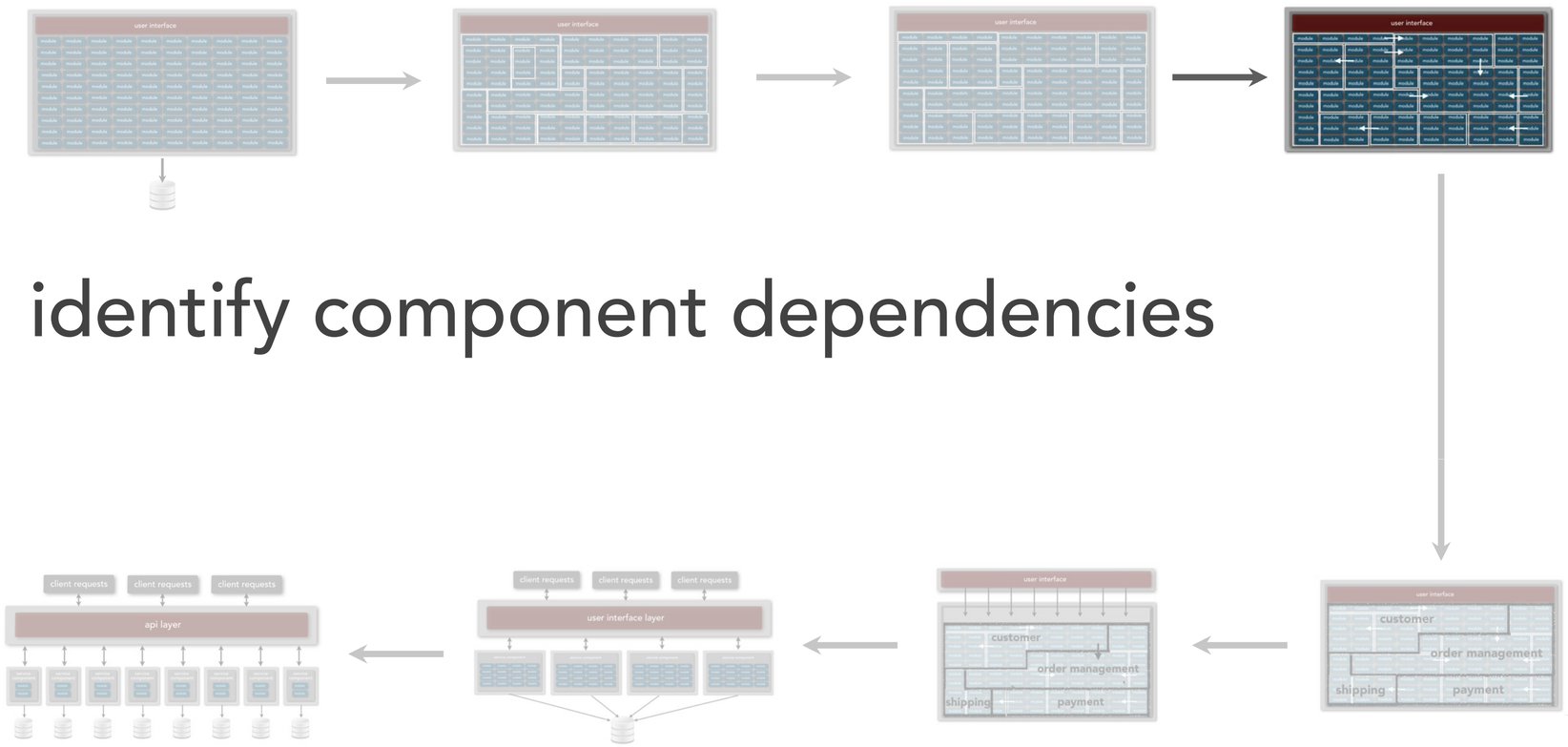
microservices roadmap



flatten and refactor components

migration patterns

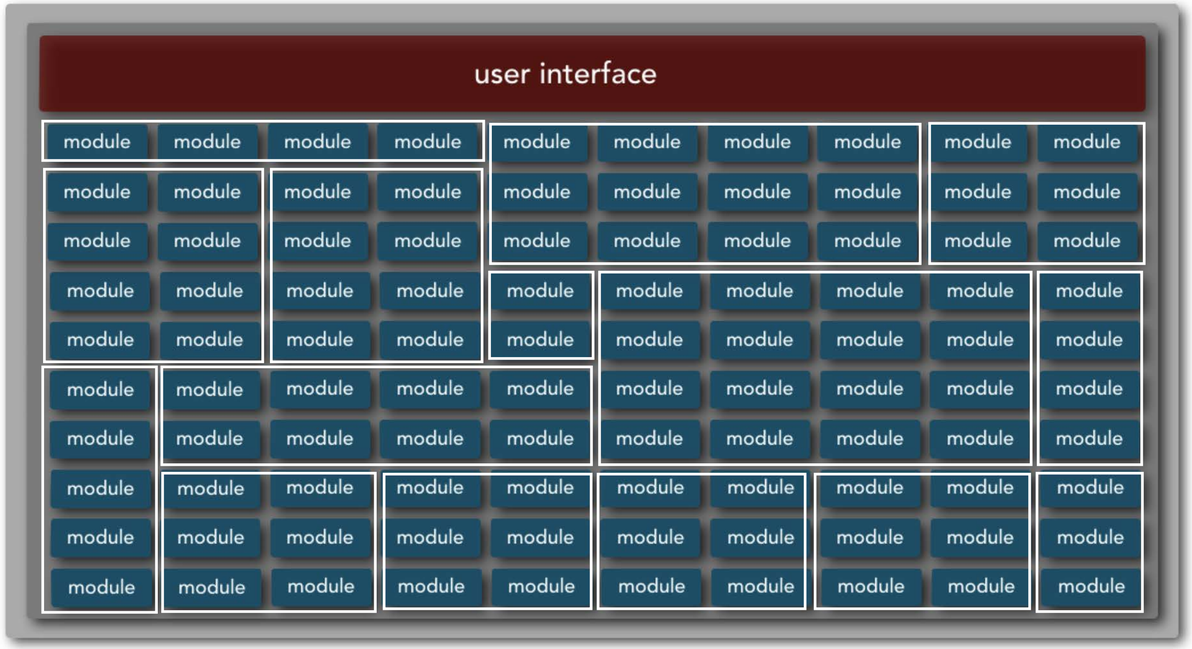
microservices roadmap



identify component dependencies

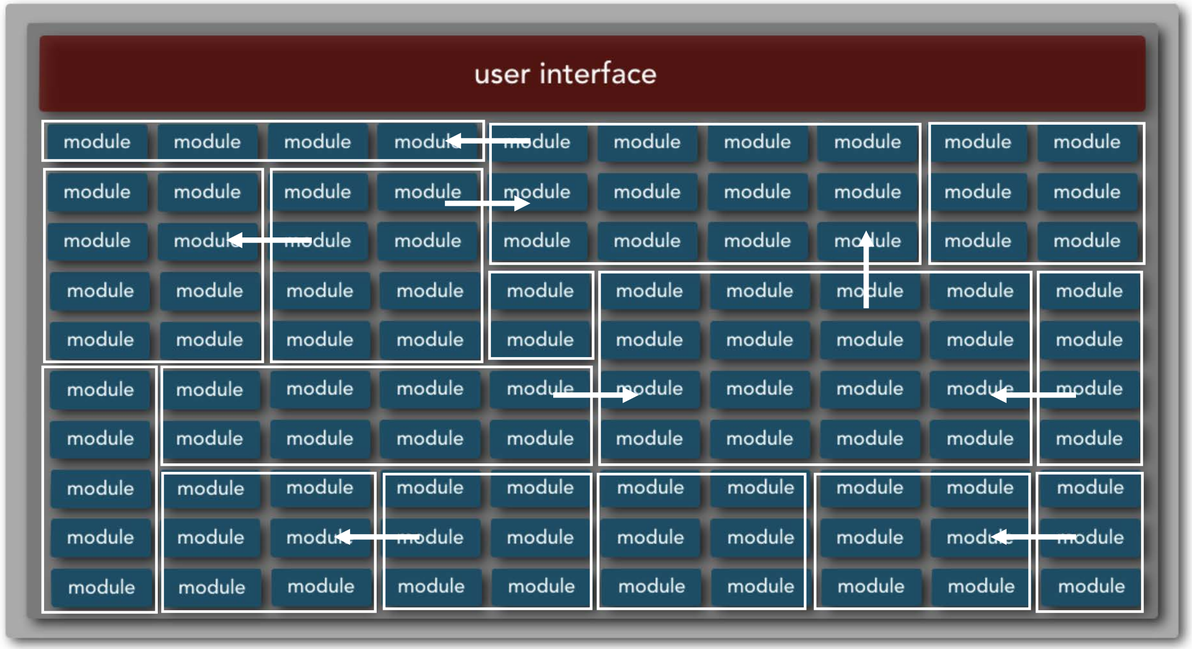
migration patterns

identify components dependencies



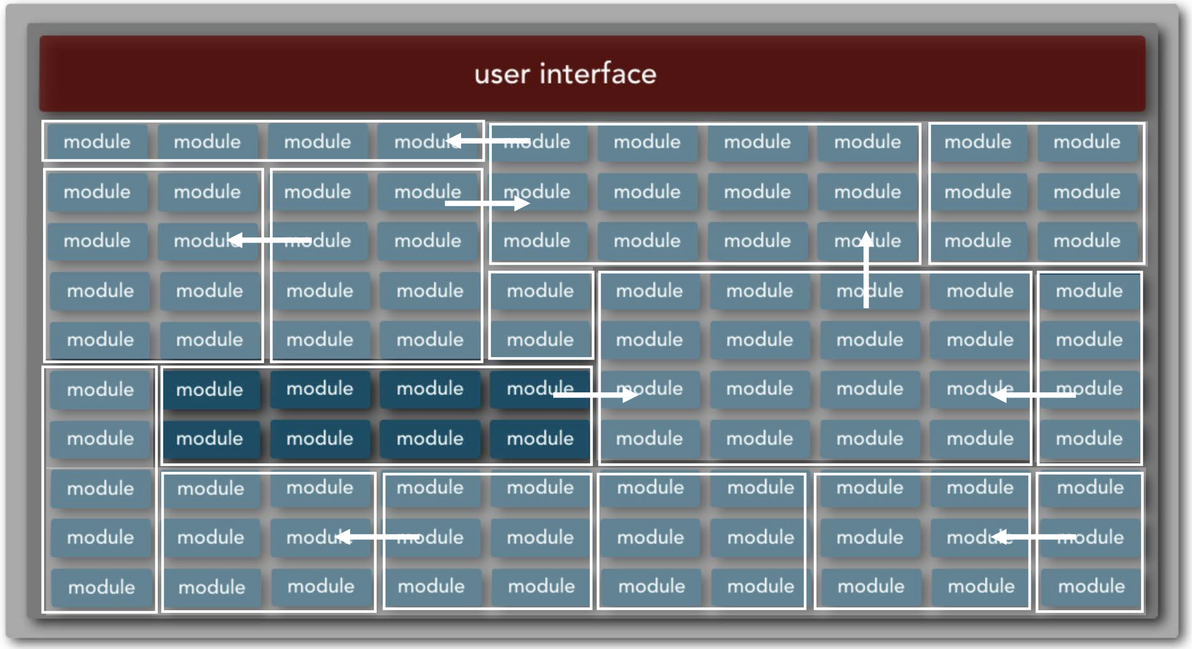
migration patterns

identify components dependencies



migration patterns

identify components dependencies

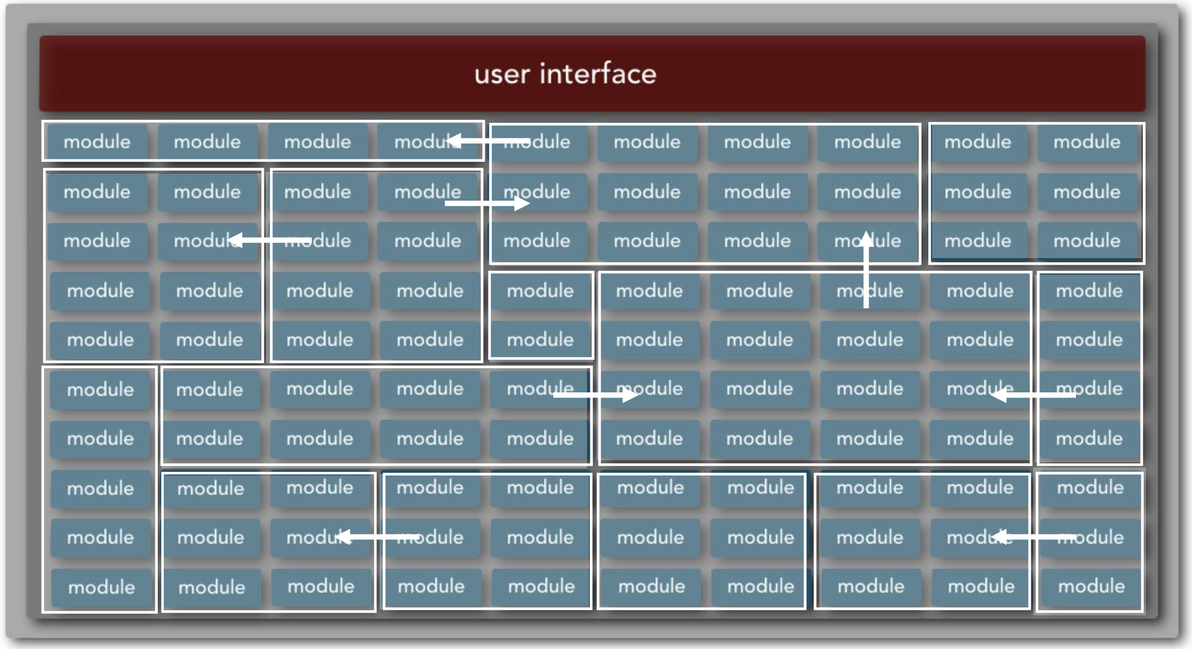


app.shipping.fulfillment.OrderShipment app.payment.invoicing.CustomerInvoice

import

migration patterns

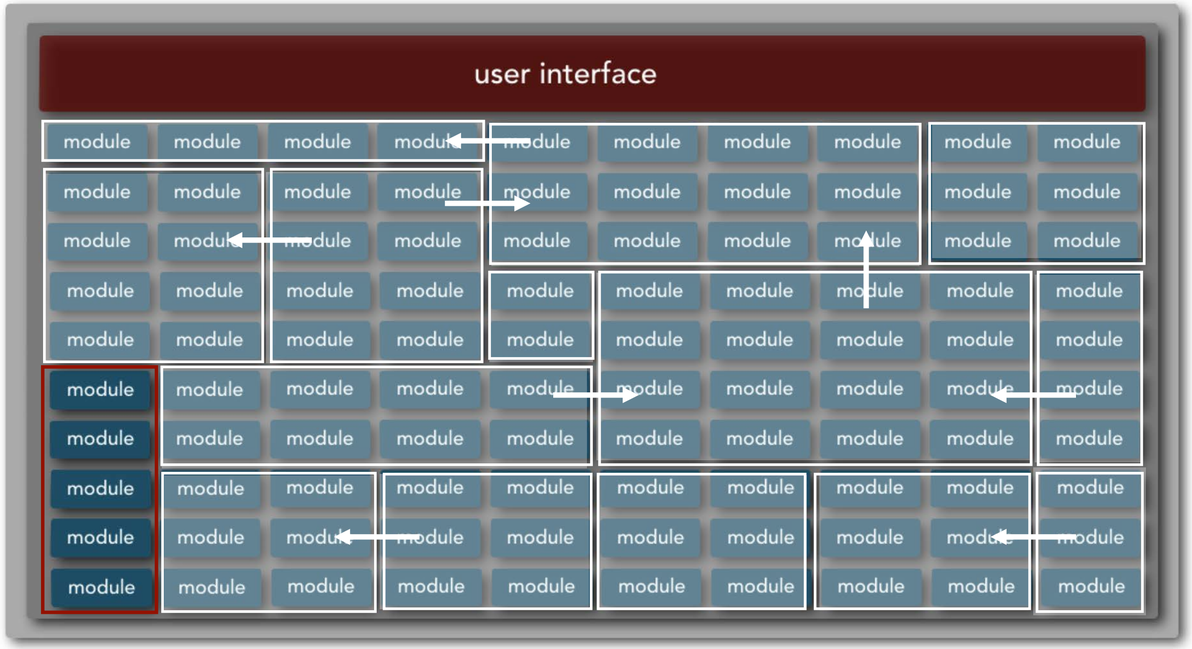
identify components dependencies



component coupling analysis

migration patterns

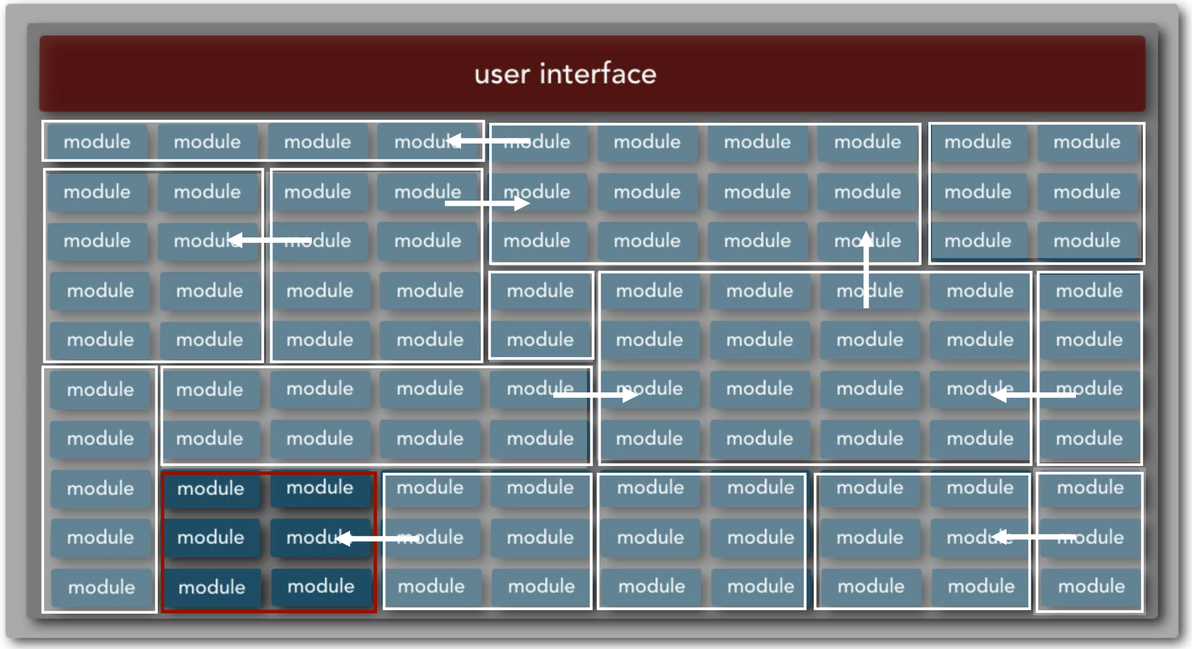
identify components dependencies



component coupling analysis

migration patterns

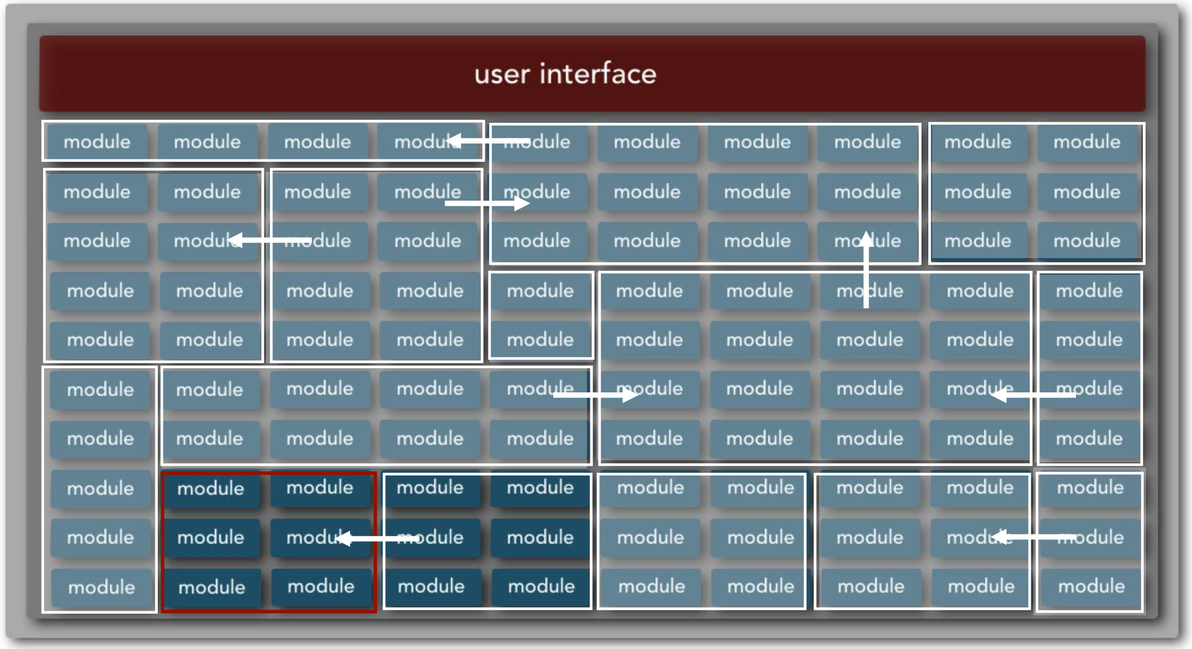
identify components dependencies



component coupling analysis

migration patterns

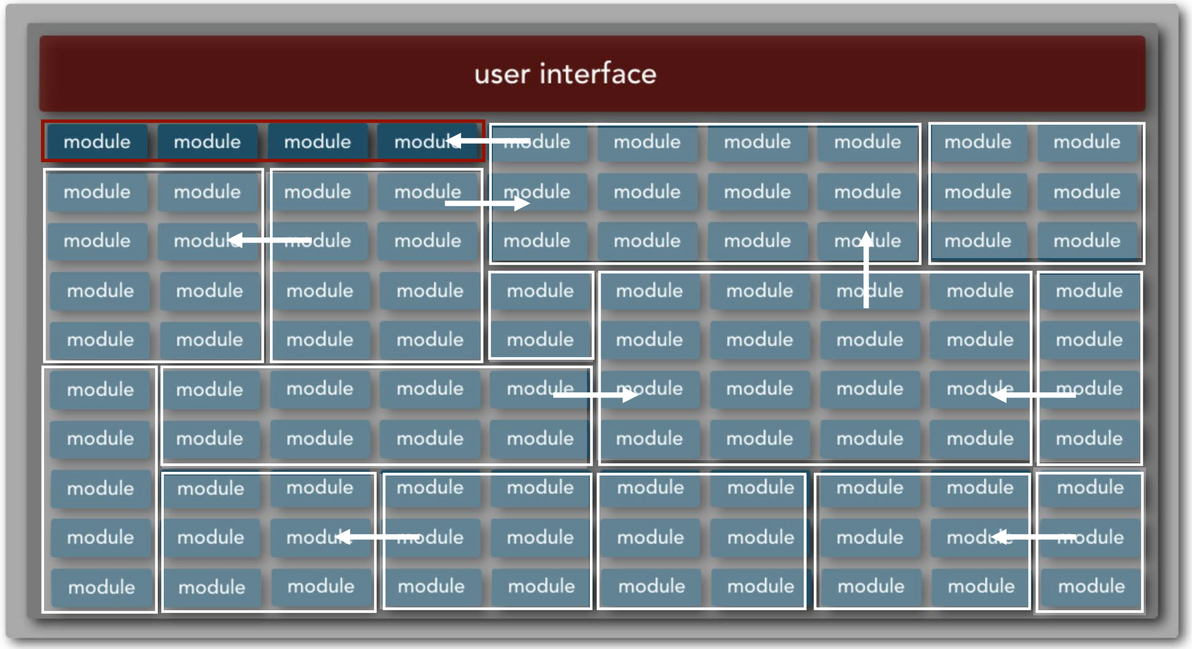
identify components dependencies



component coupling analysis

migration patterns

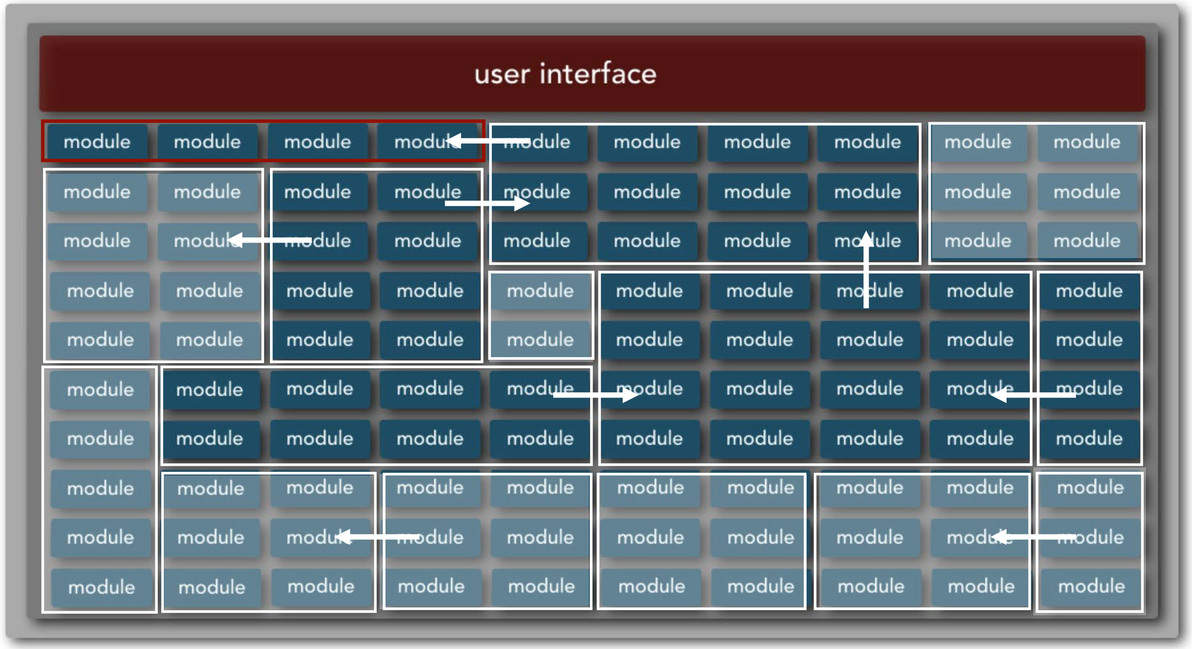
identify components dependencies



component coupling analysis

migration patterns

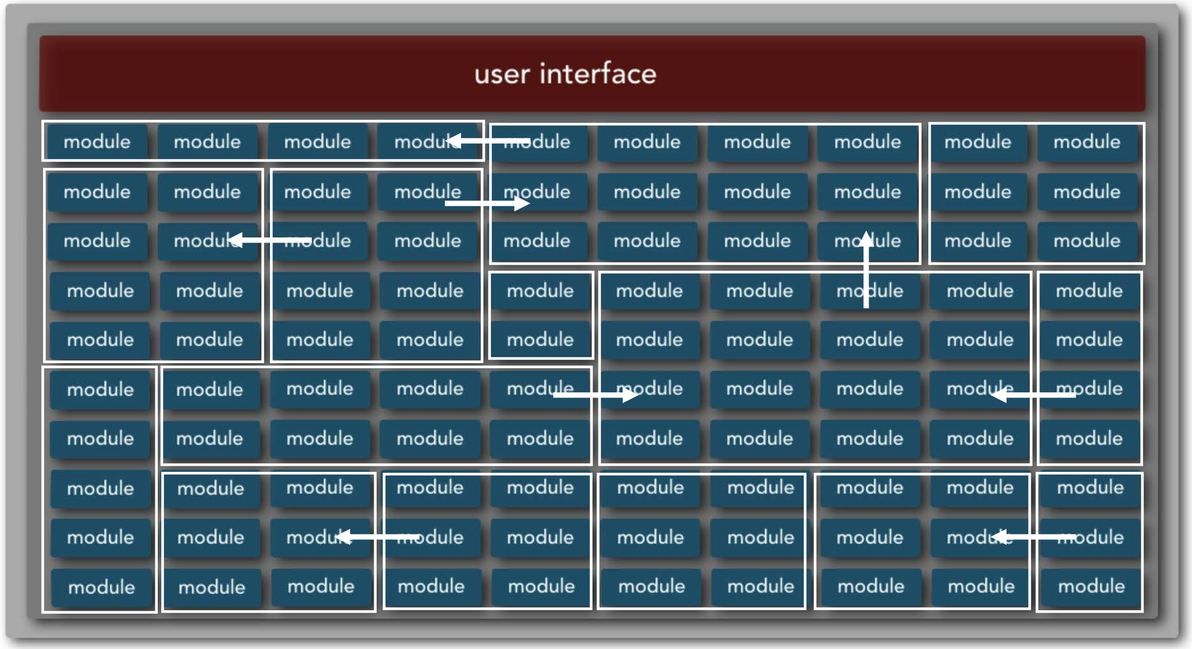
identify components dependencies



component coupling analysis

migration patterns

identify components dependencies

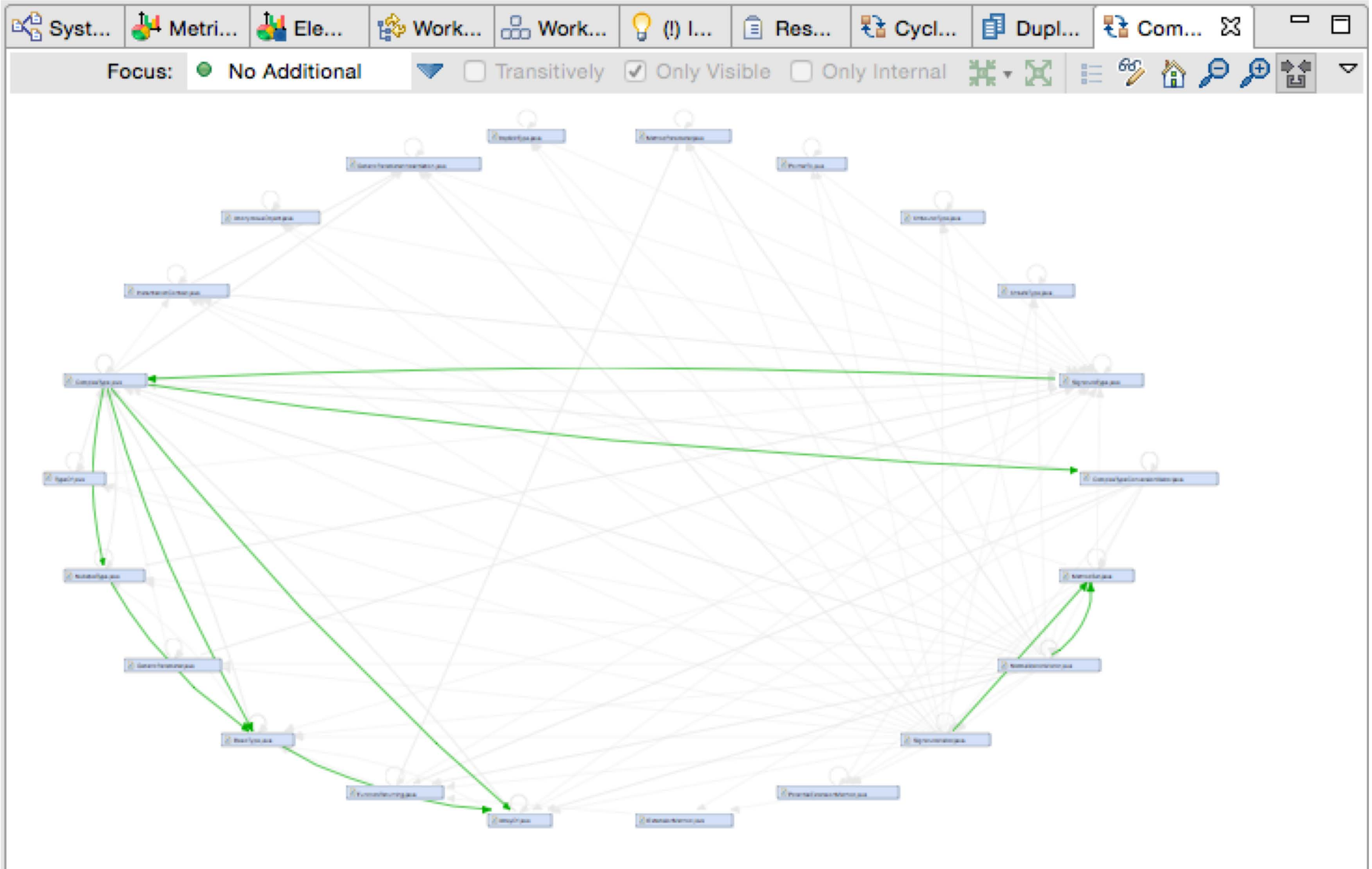


Sonargraph

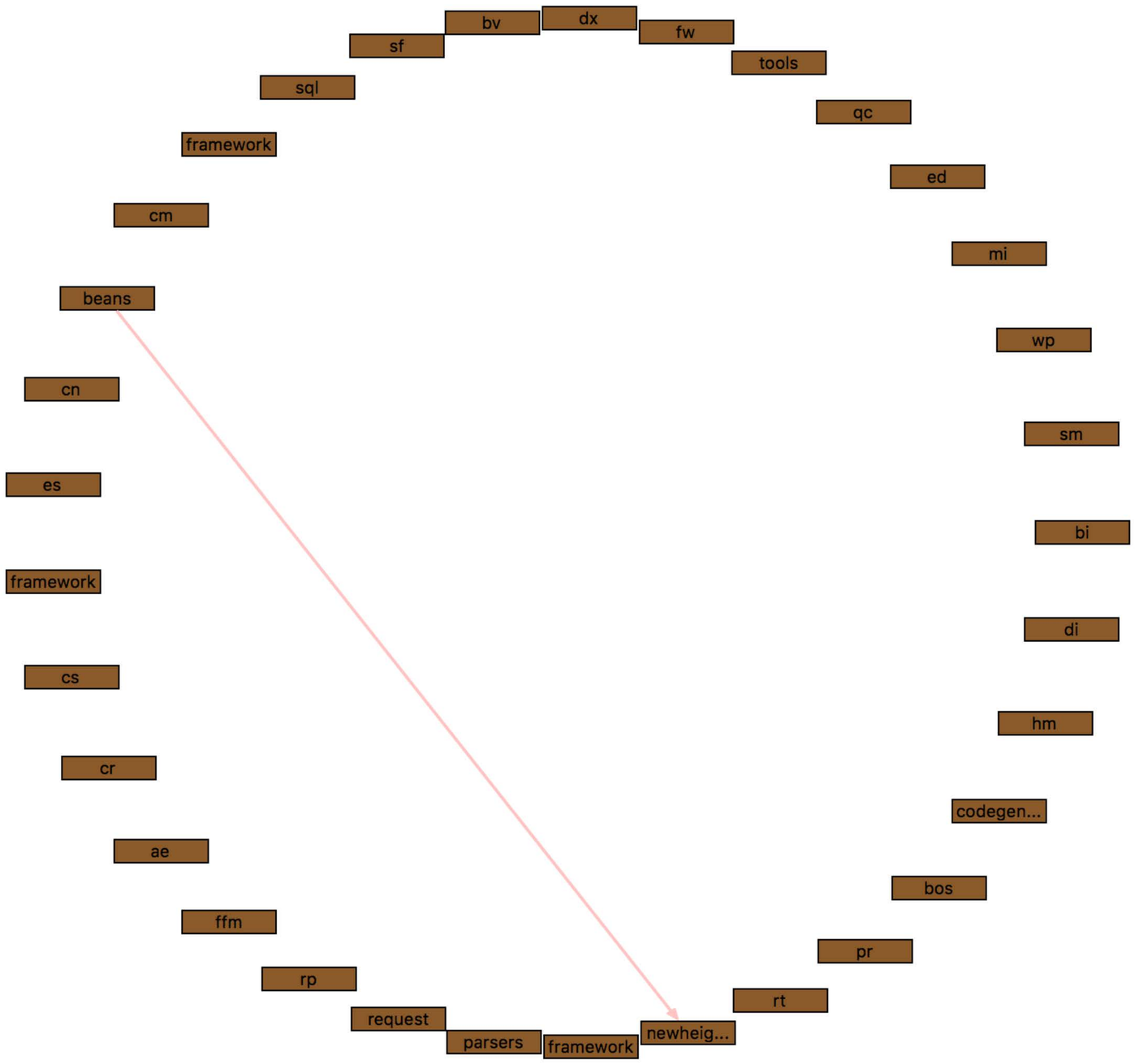
http://www.hello2morrow.com/products/sonargraph

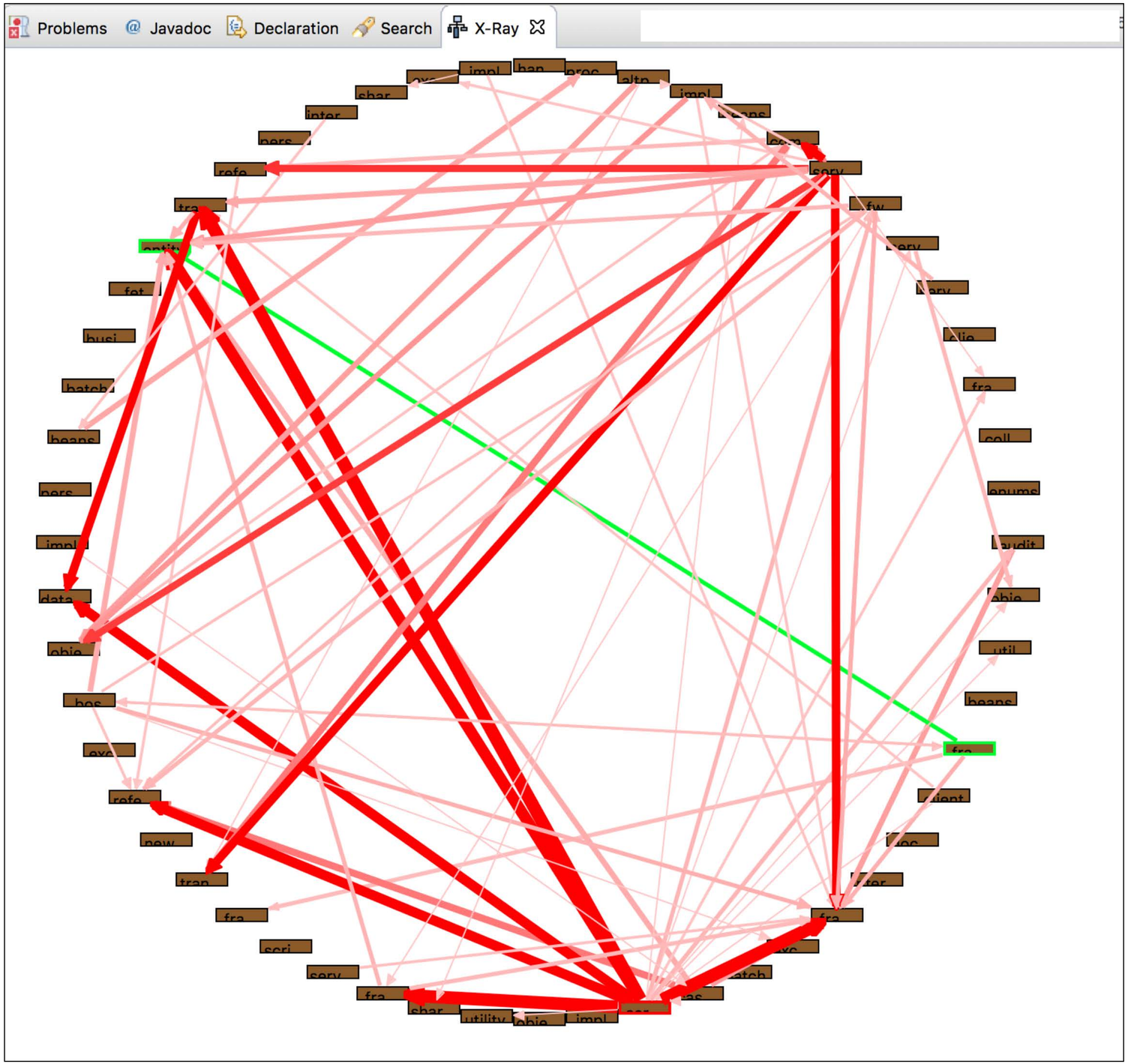
HELLO2MORROW
Empowering Software Craftsmanship

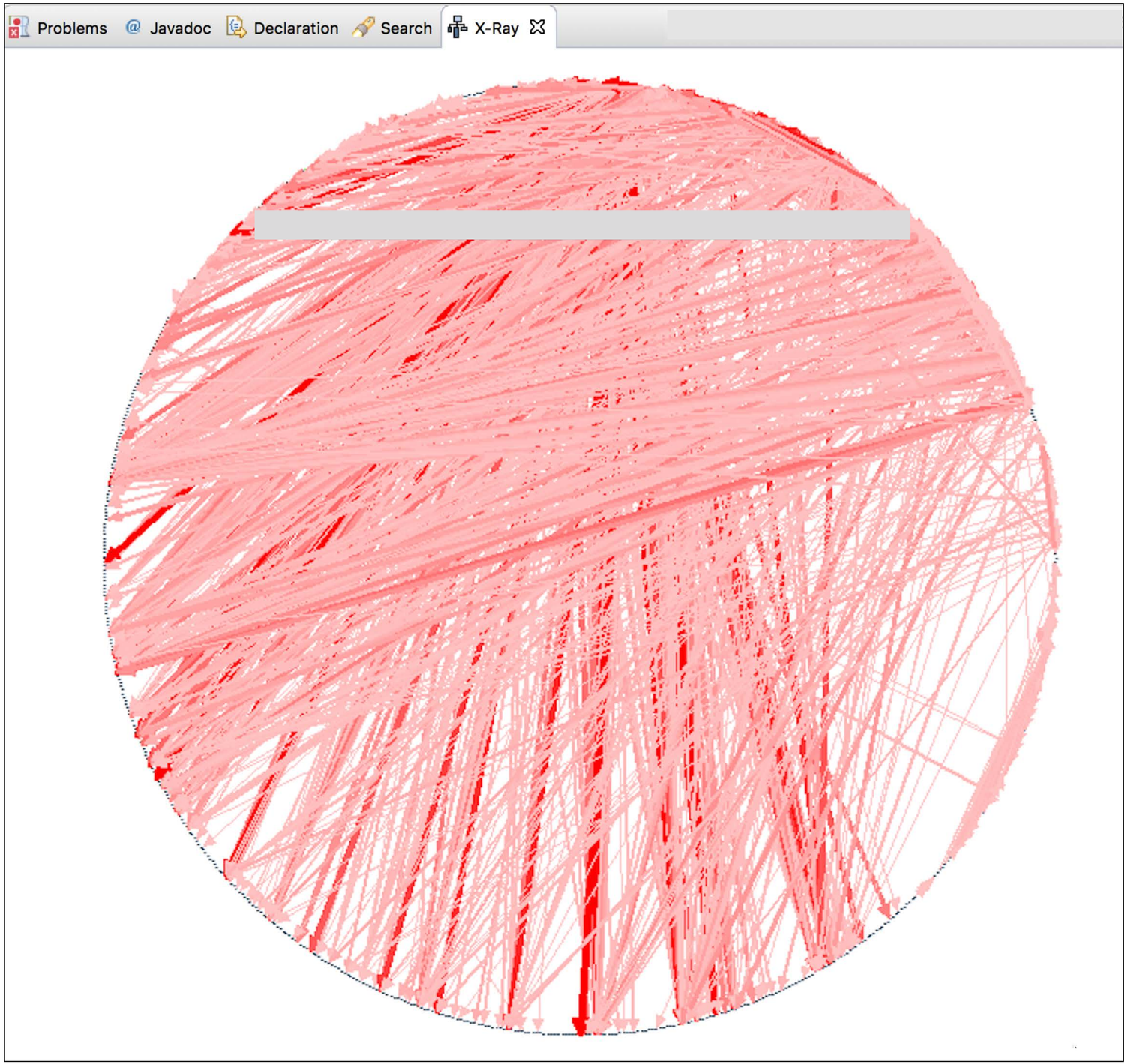
Sonargraph-Explorer	Java, C#, Python3
Sonargraph-Architect	Java, C#, C++, C, Python3


Syst...
Metri...
Ele...
Work...
Work...
(!) I...
Res...
Cycl...
Dupl...
Com...
Focus:
No Additional
Transitively
Only Visible
Only Internal

Problems | Javadoc | Declaration | Search | X-Ray

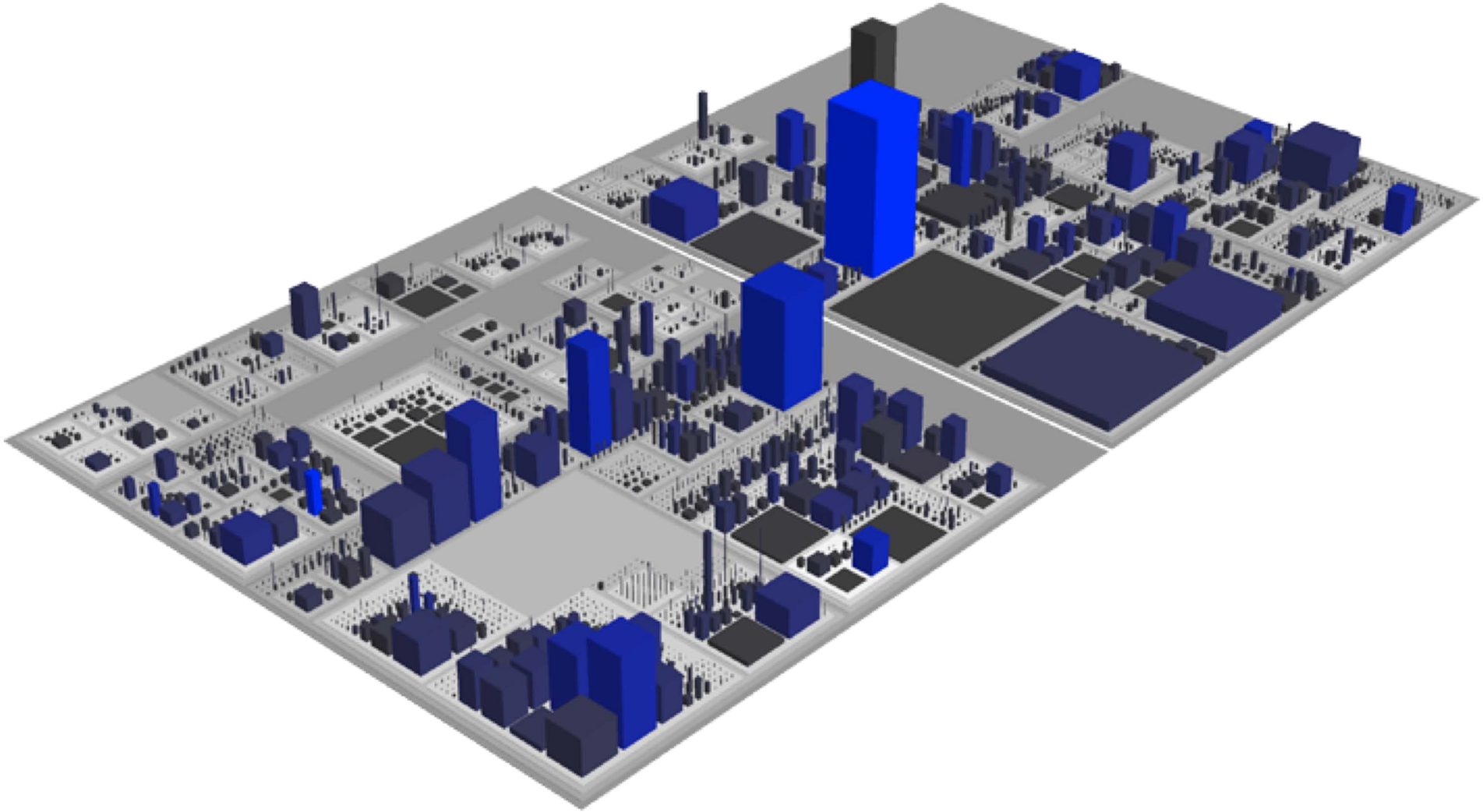



Problems
Javadoc
Declaration
Search
X-Ray


Problems
Javadoc
Declaration
Search
X-Ray

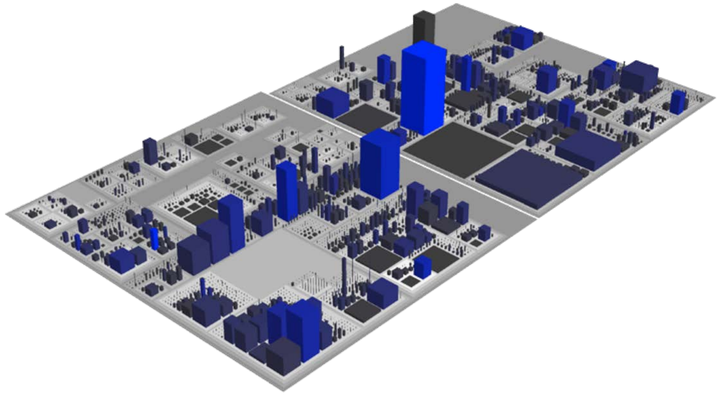
Code City

http://www.inf.usi.ch/phd/wettel/codecity.html



Code City

http://www.inf.usi.ch/phd/wettel/codecity.html



https://www.developertoarchitect.com/downloads/iplasma.zip

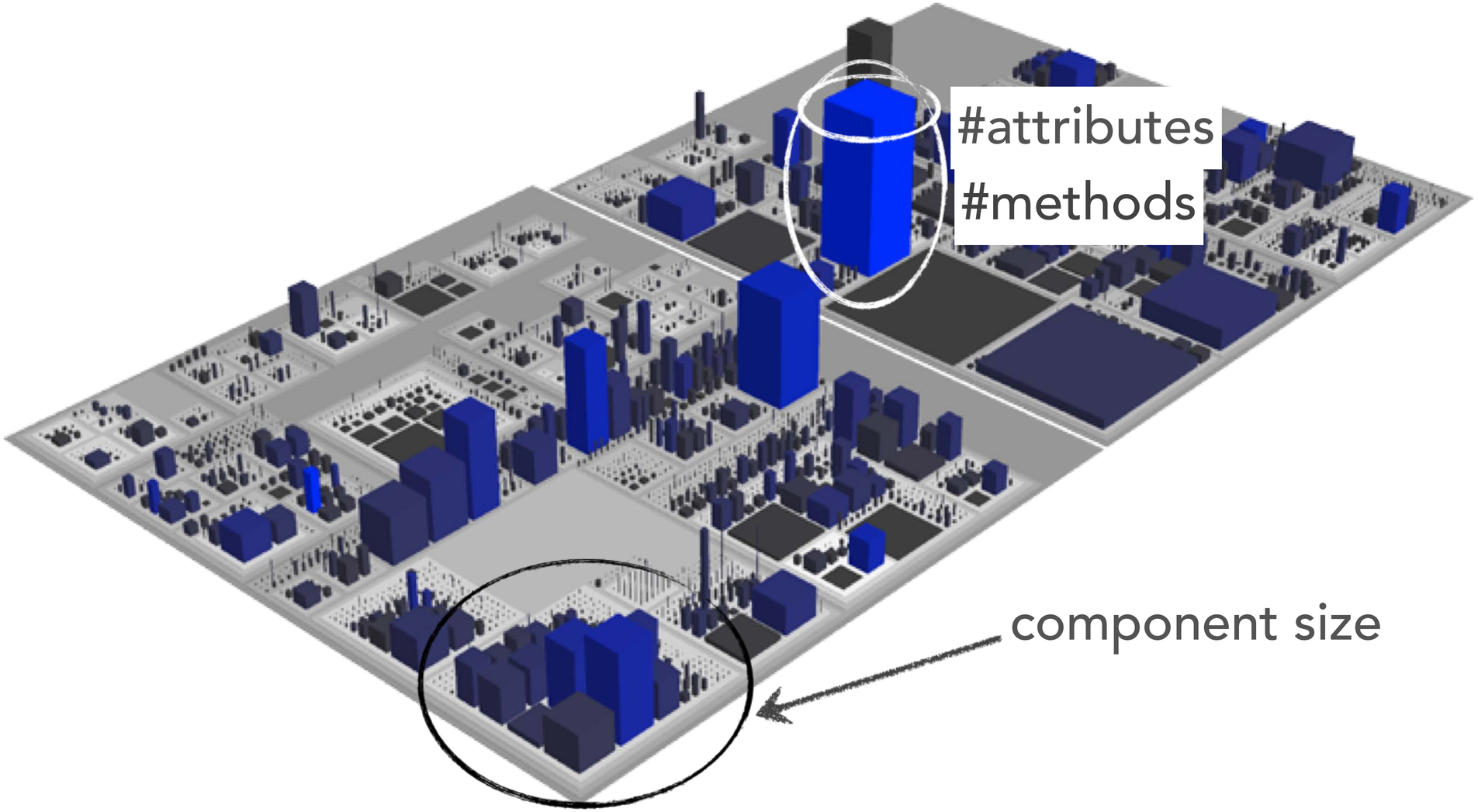
iPlasma6/tools/iPlasma/insider.sh (.bat)

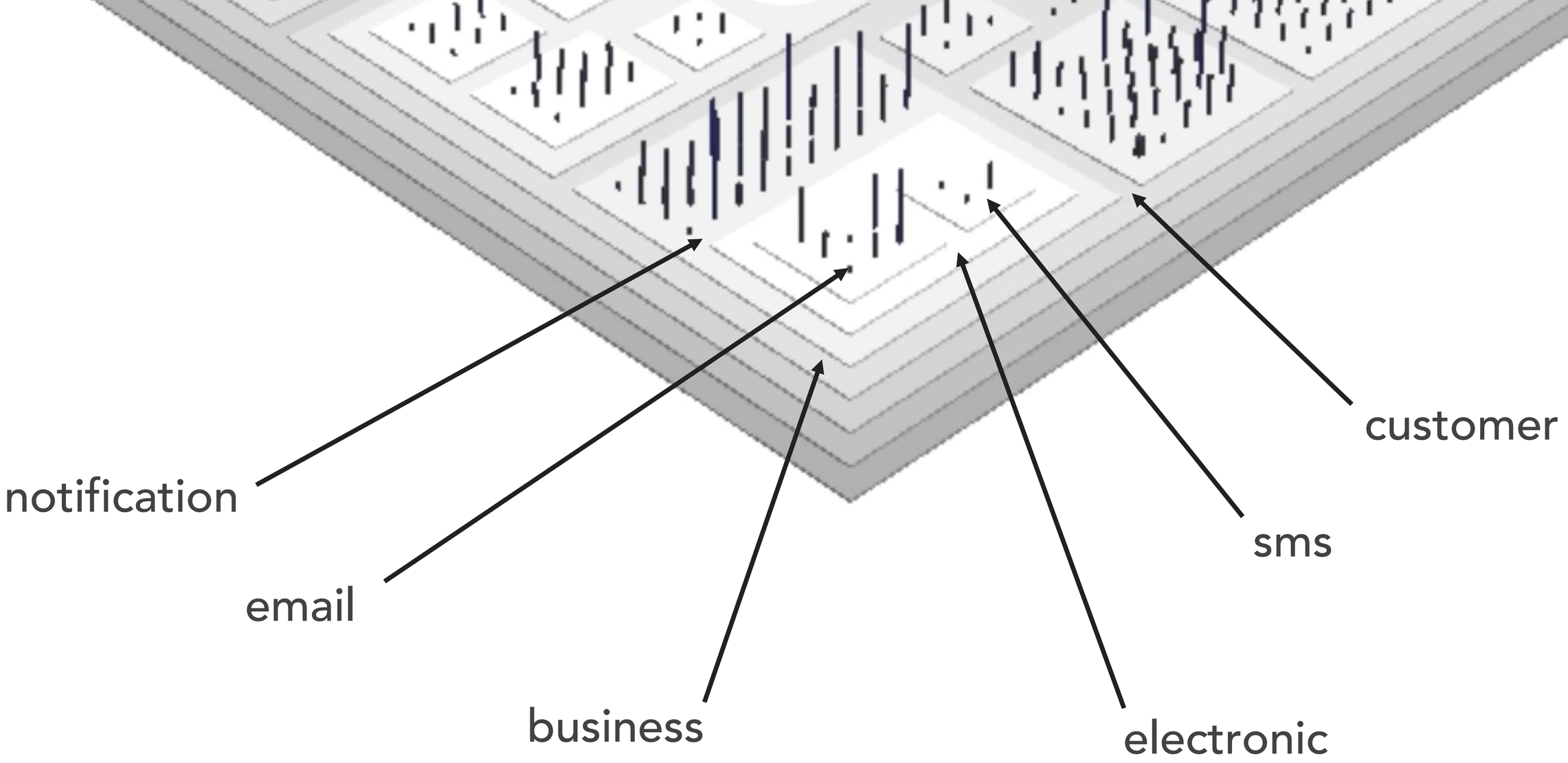
https://www.developertoarchitect.com/downloads/codecity-mac.zip

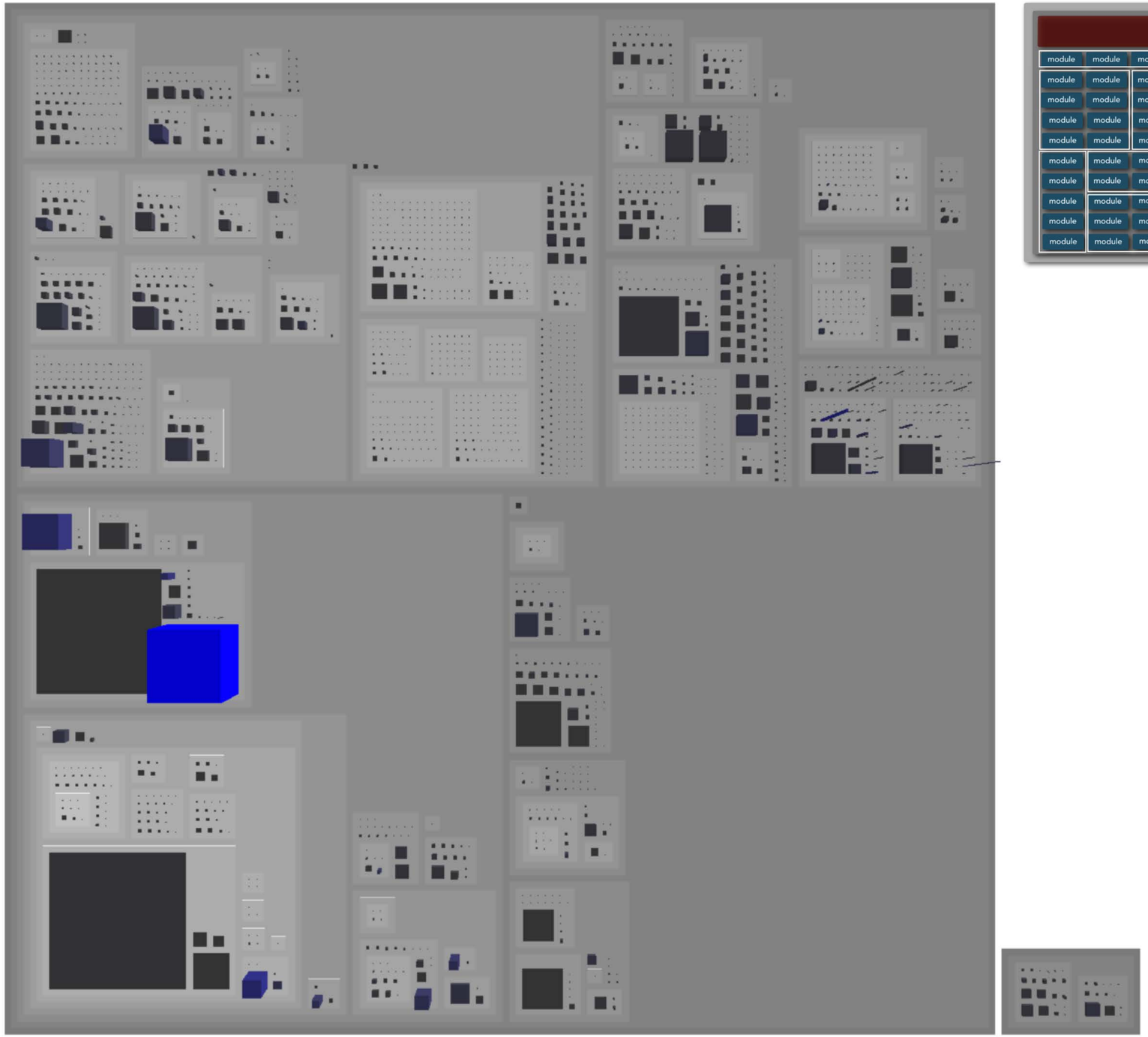
https://www.developertoarchitect.com/downloads/codecity-win.zip

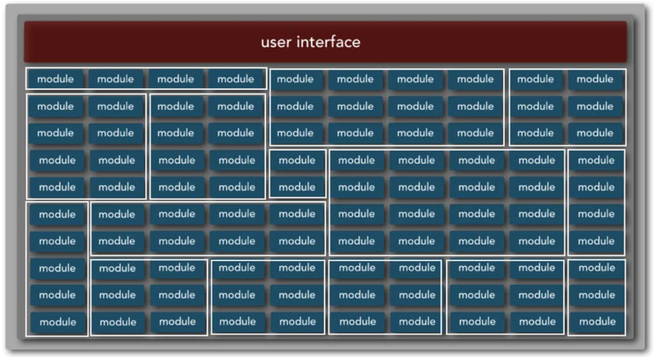
Code City

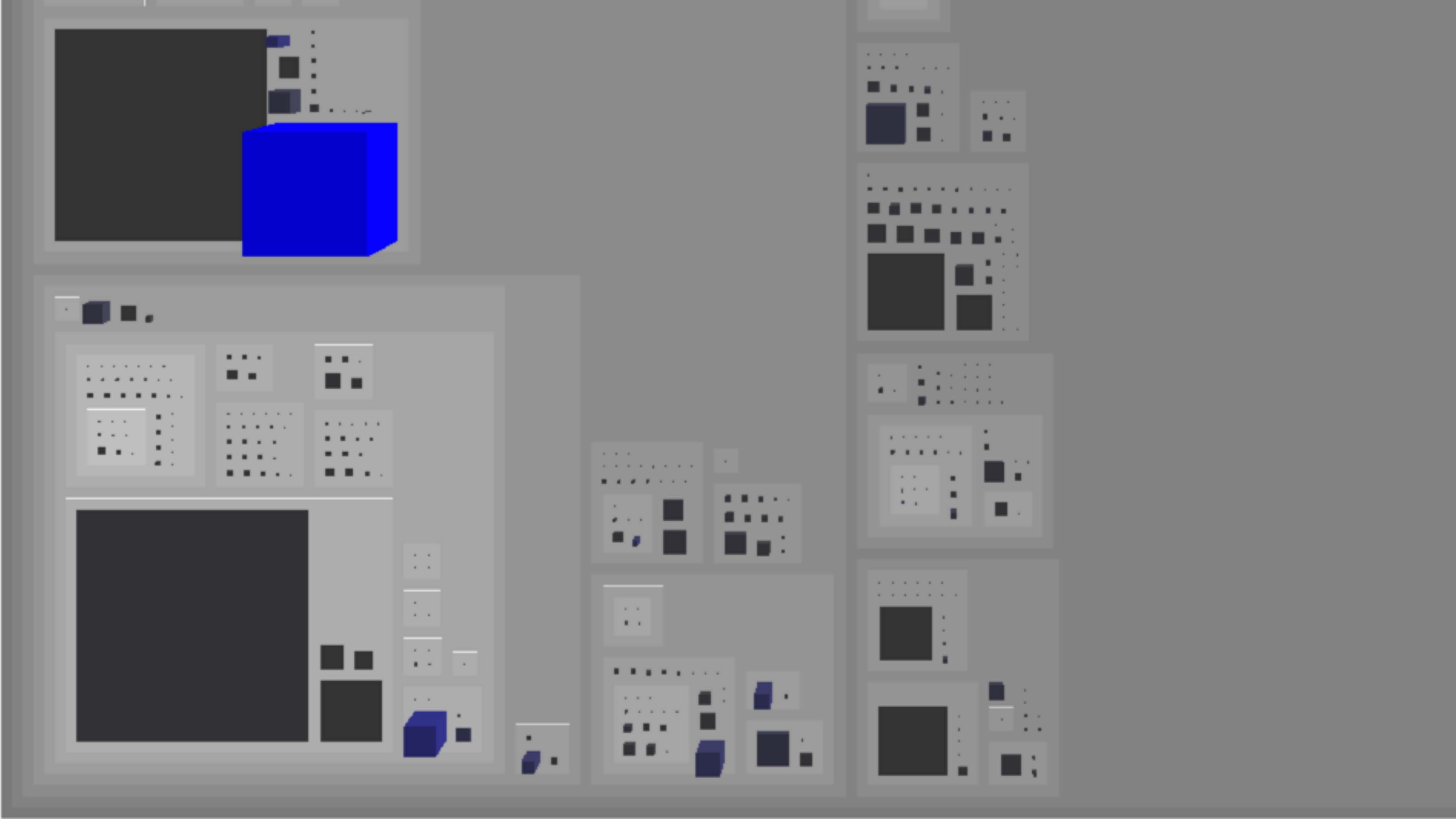
http://www.inf.usi.ch/phd/wettel/codecity.html

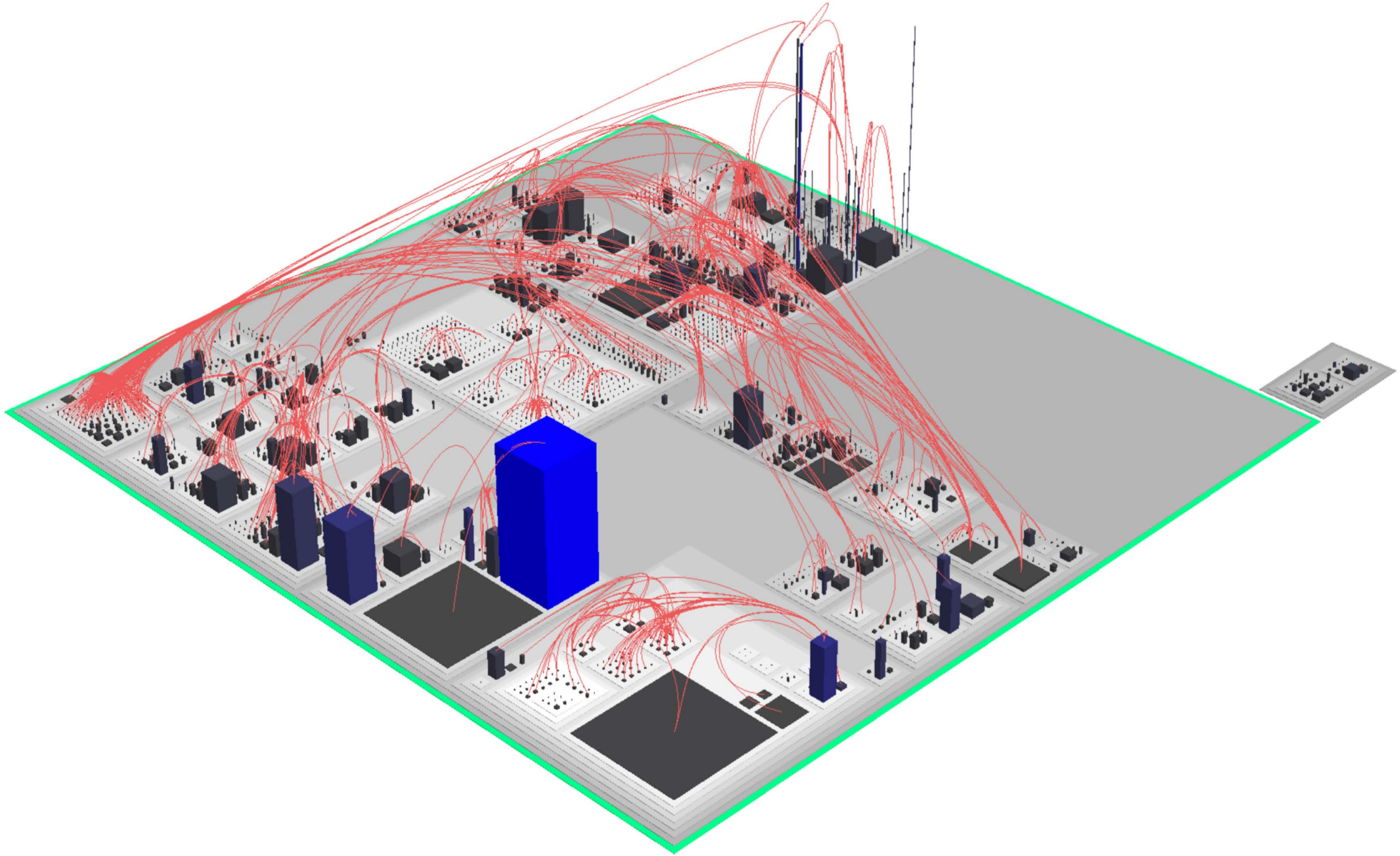


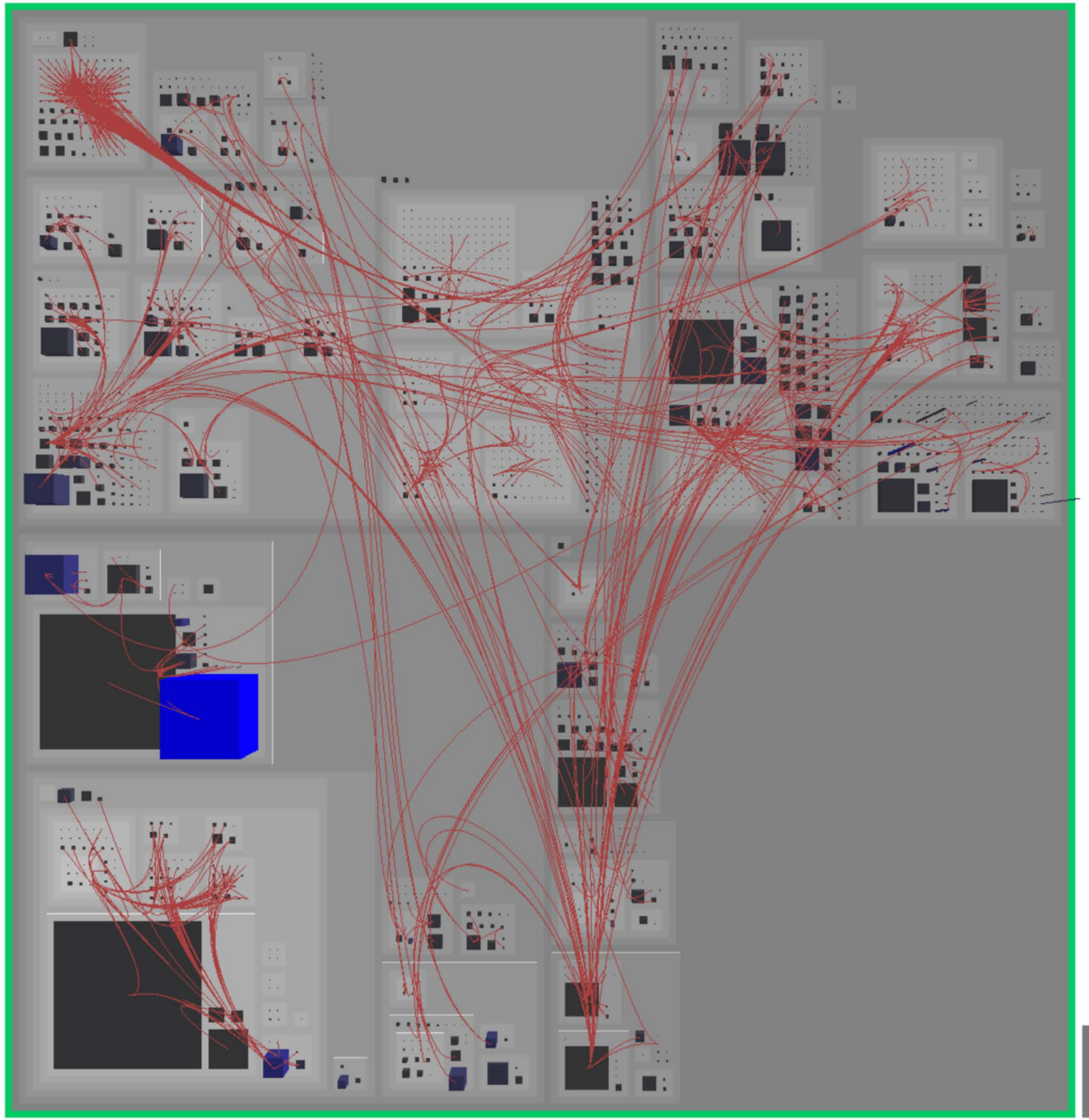

notification
email
business
electronic
sms
customer

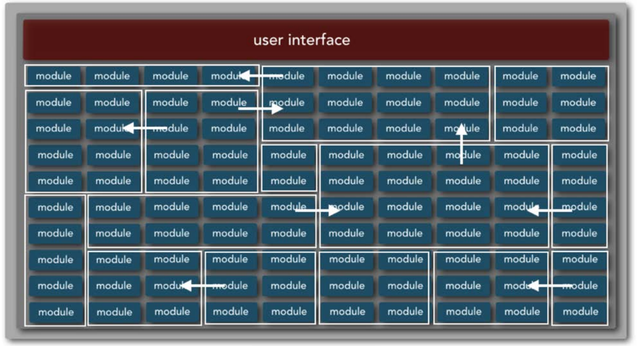


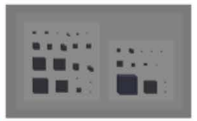

user interface
module

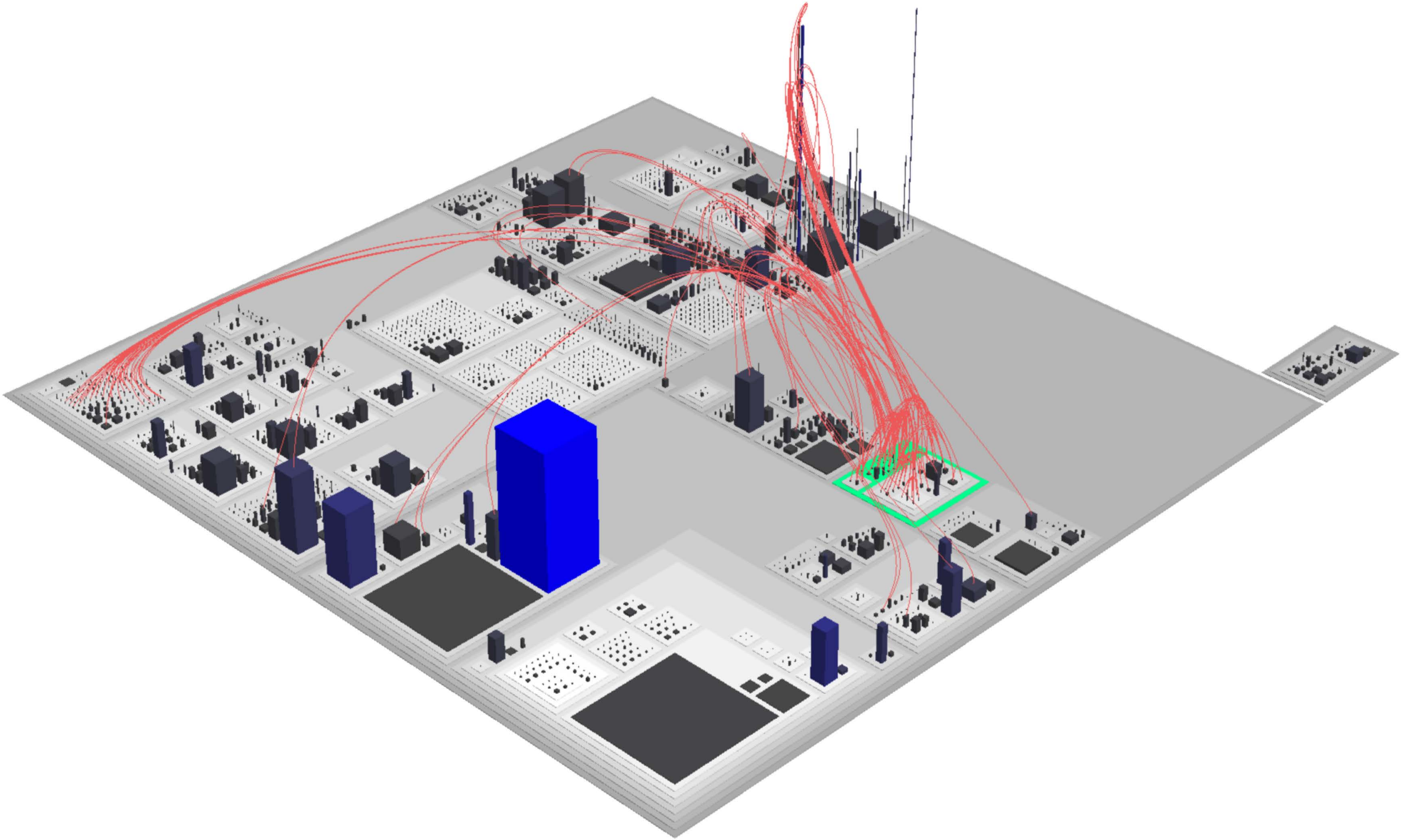




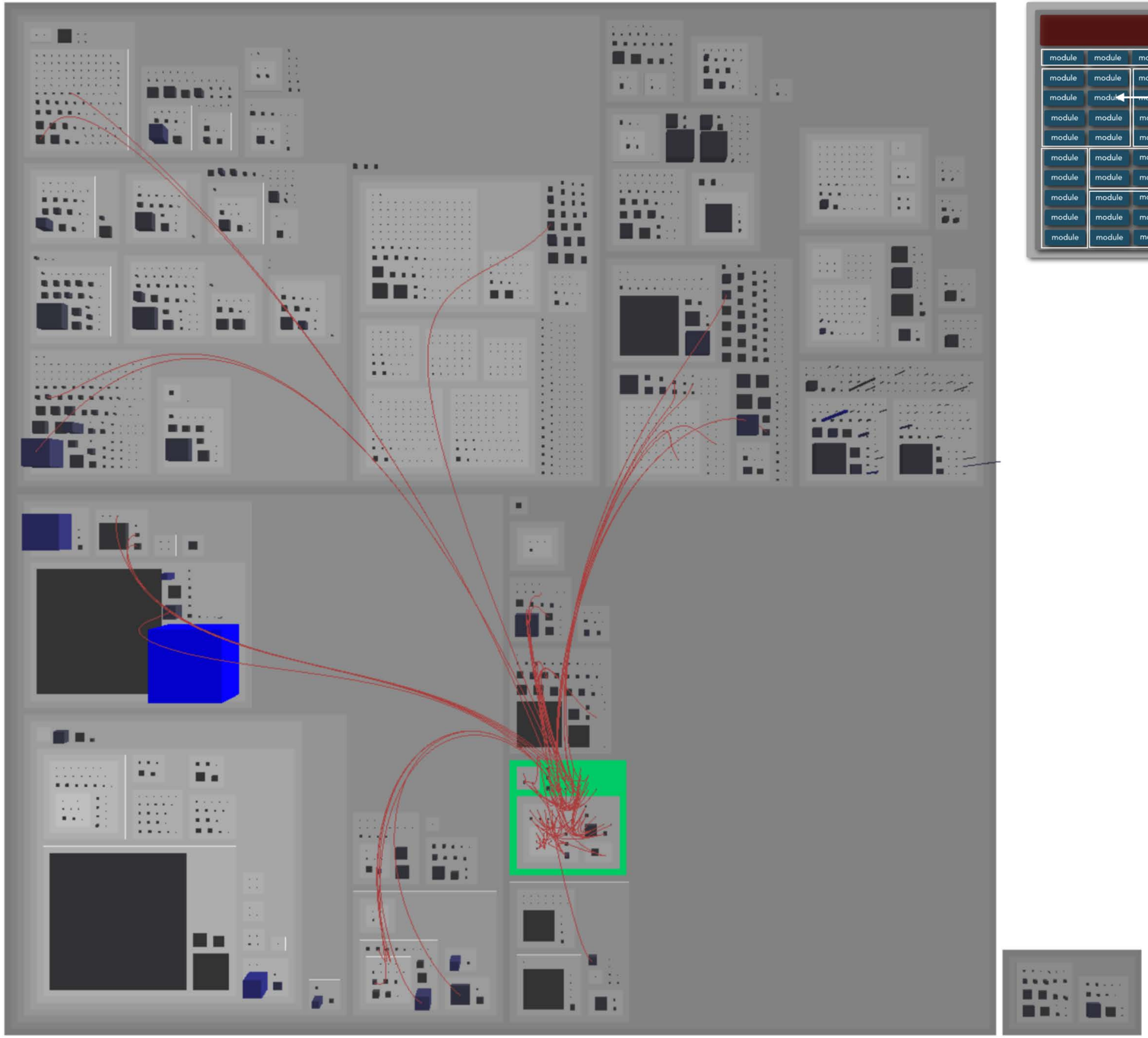


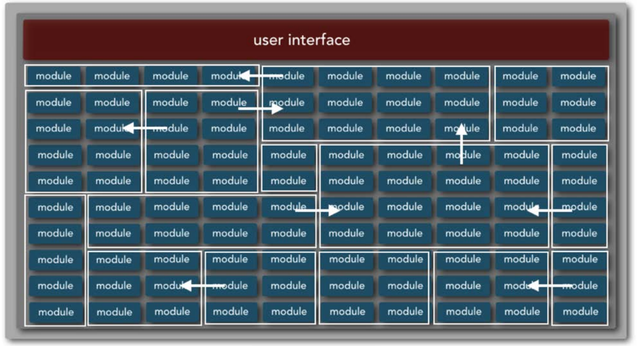

user interface
module



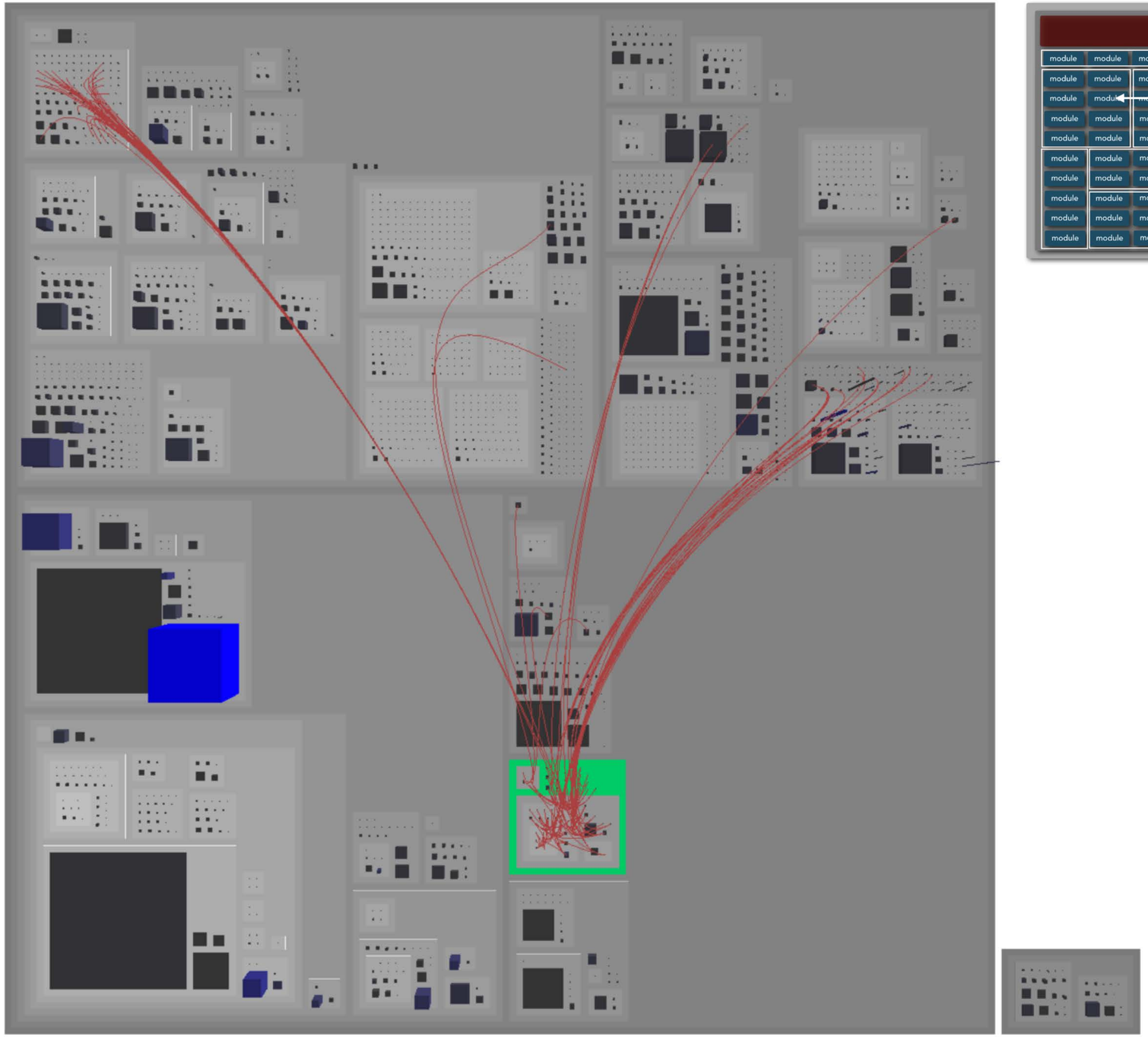


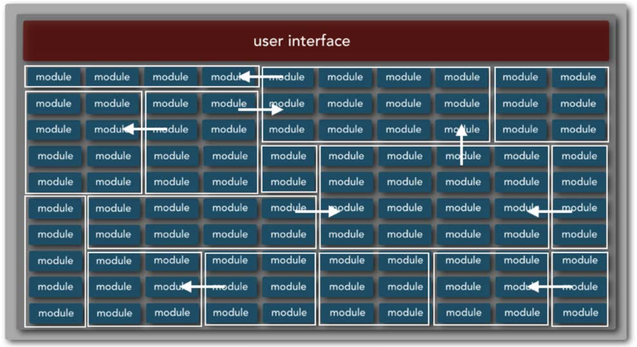
incoming dependencies





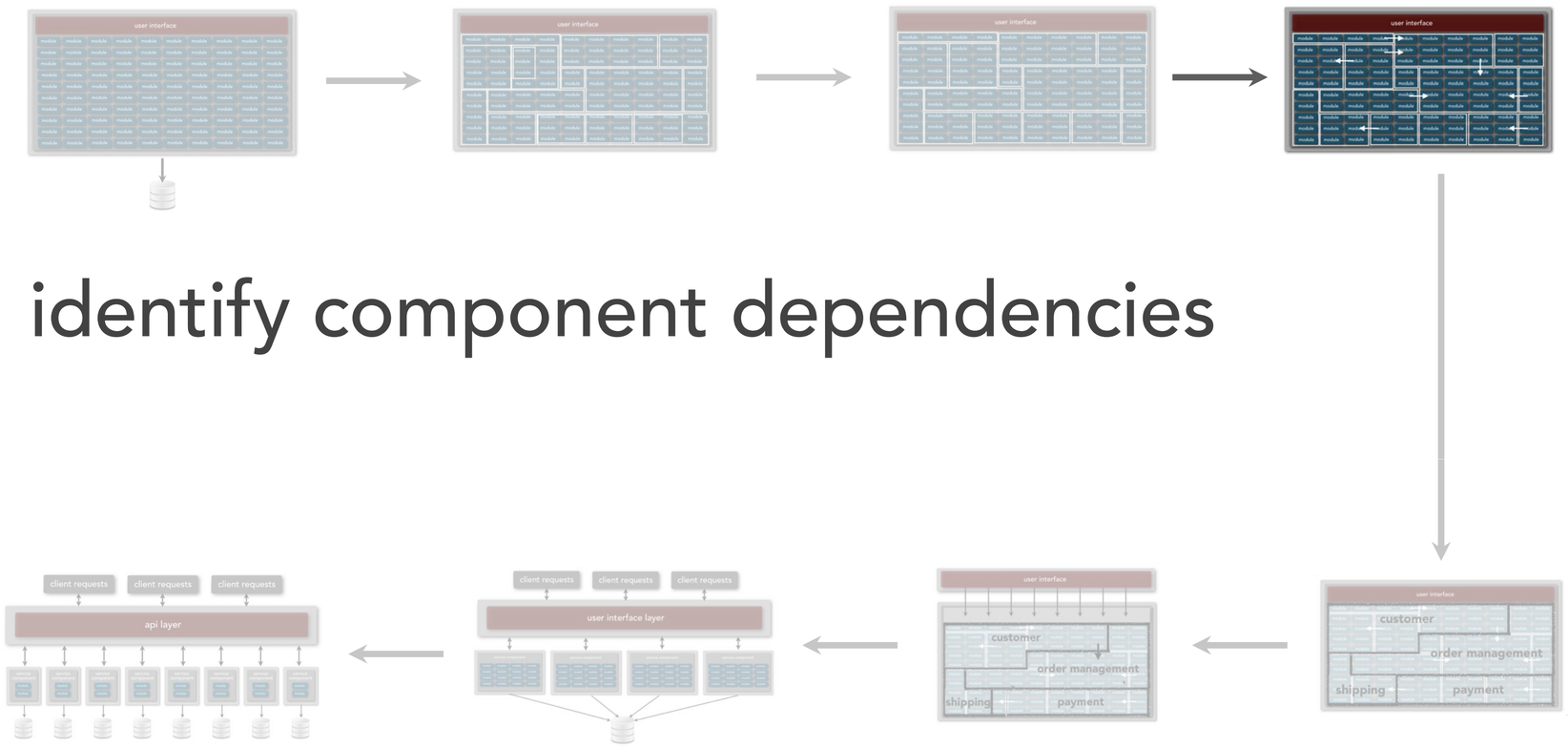
outgoing dependencies





migration patterns

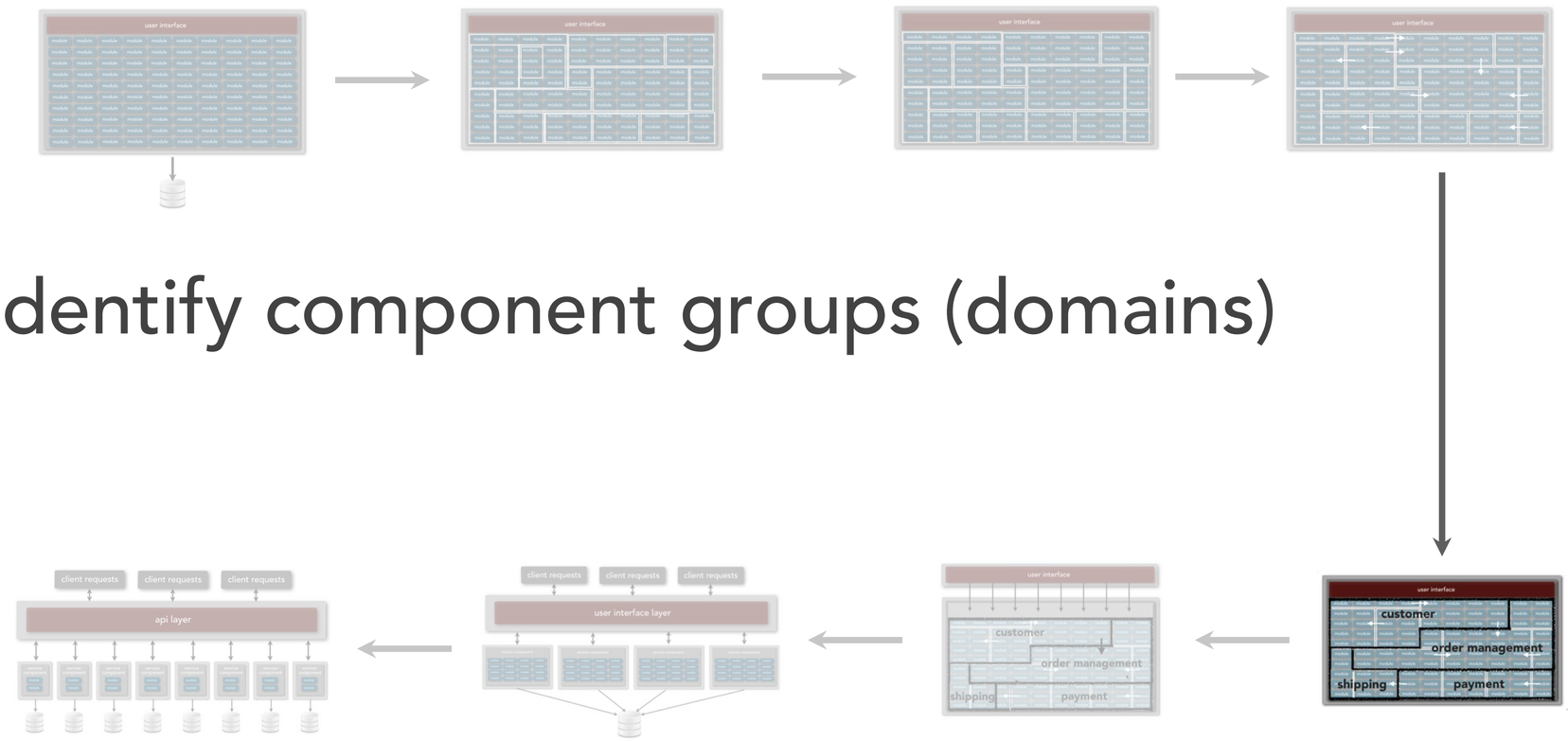
microservices roadmap



identify component dependencies

migration patterns

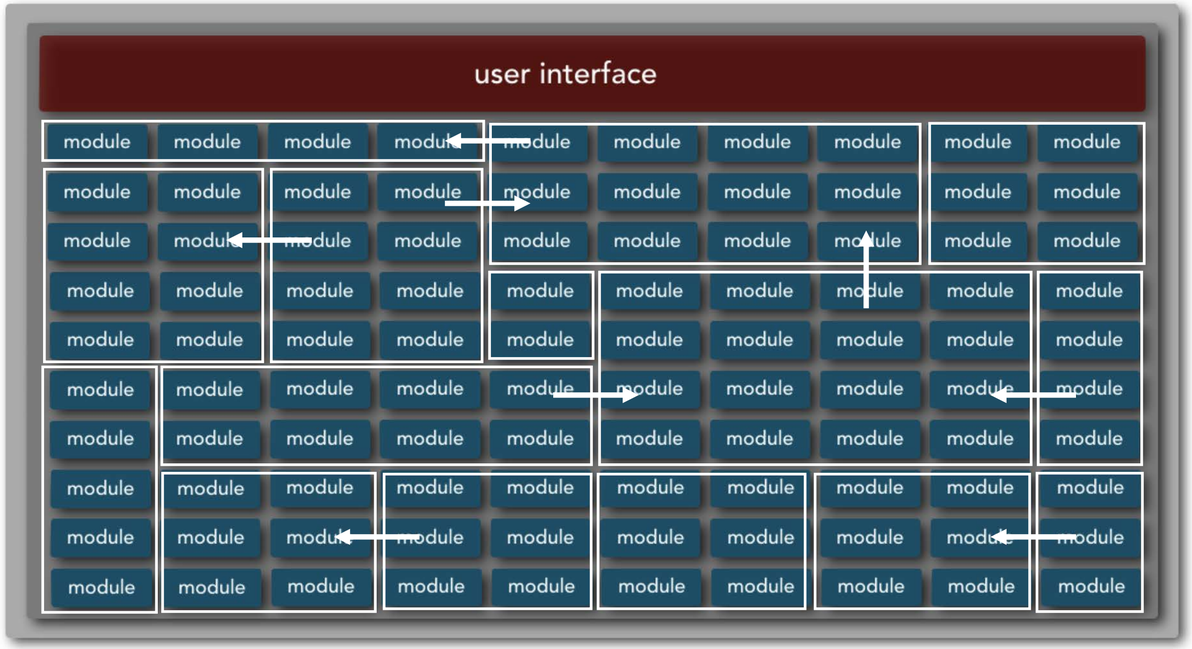
microservices roadmap



identify component groups (domains)

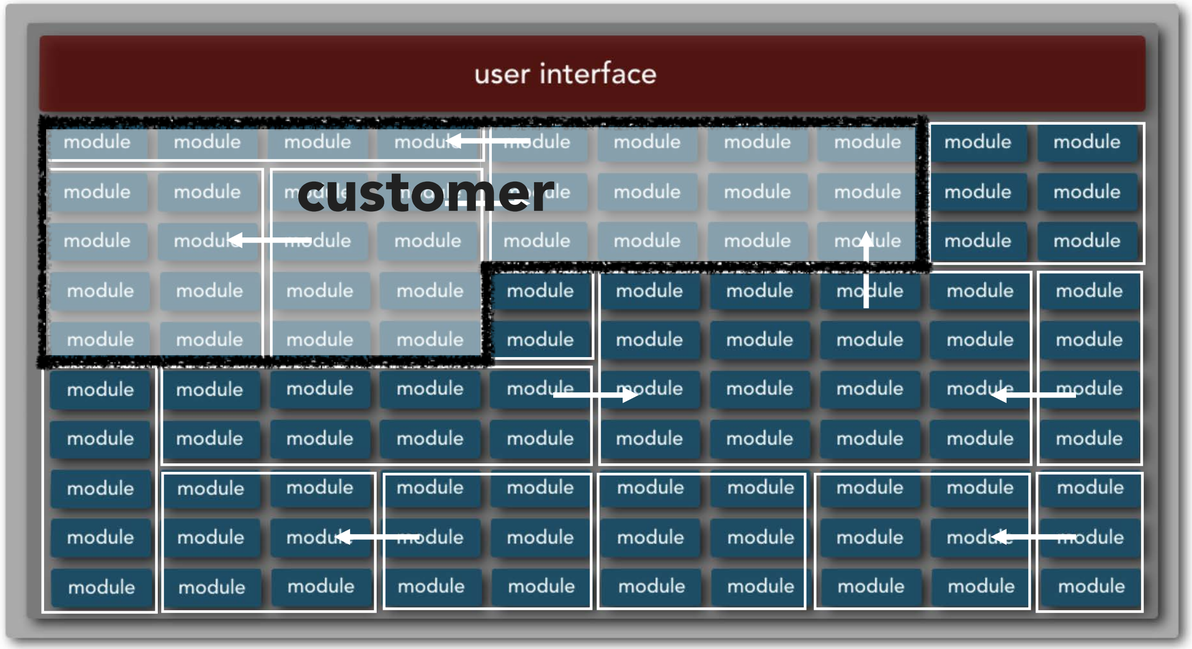
migration patterns

identify component groups (domains)



migration patterns

identify component groups (domains)



app.customer.info

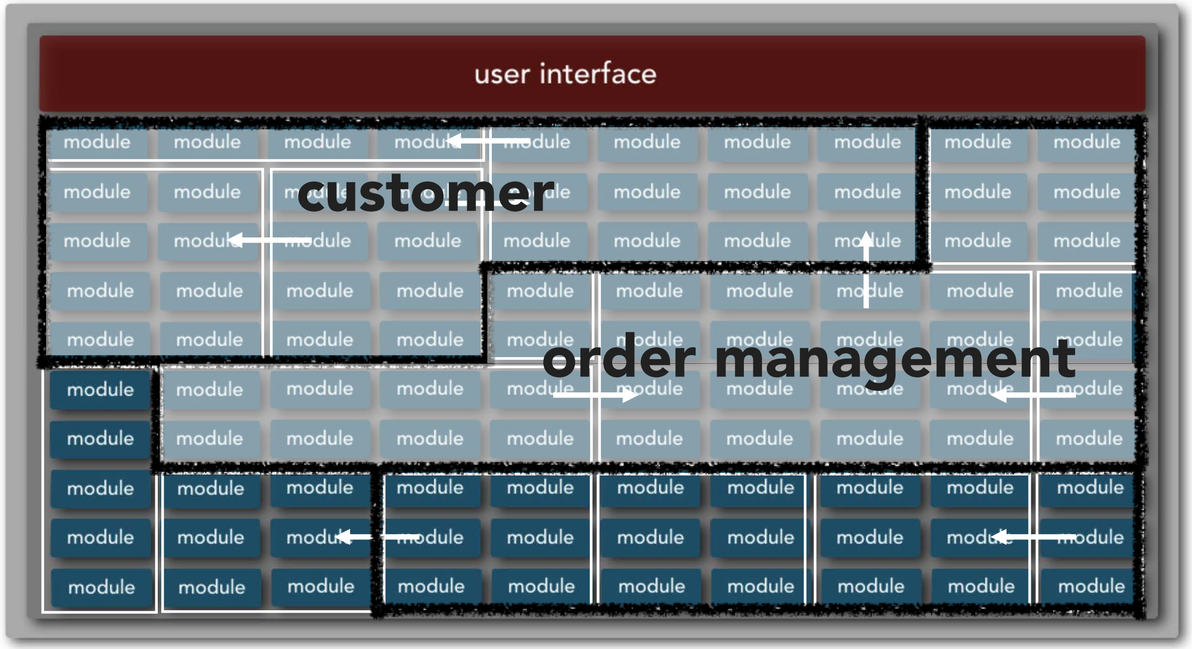
app.customer.notification

app.customer.preferences

app.customer.tracking

migration patterns

identify component groups (domains)



app.order.history

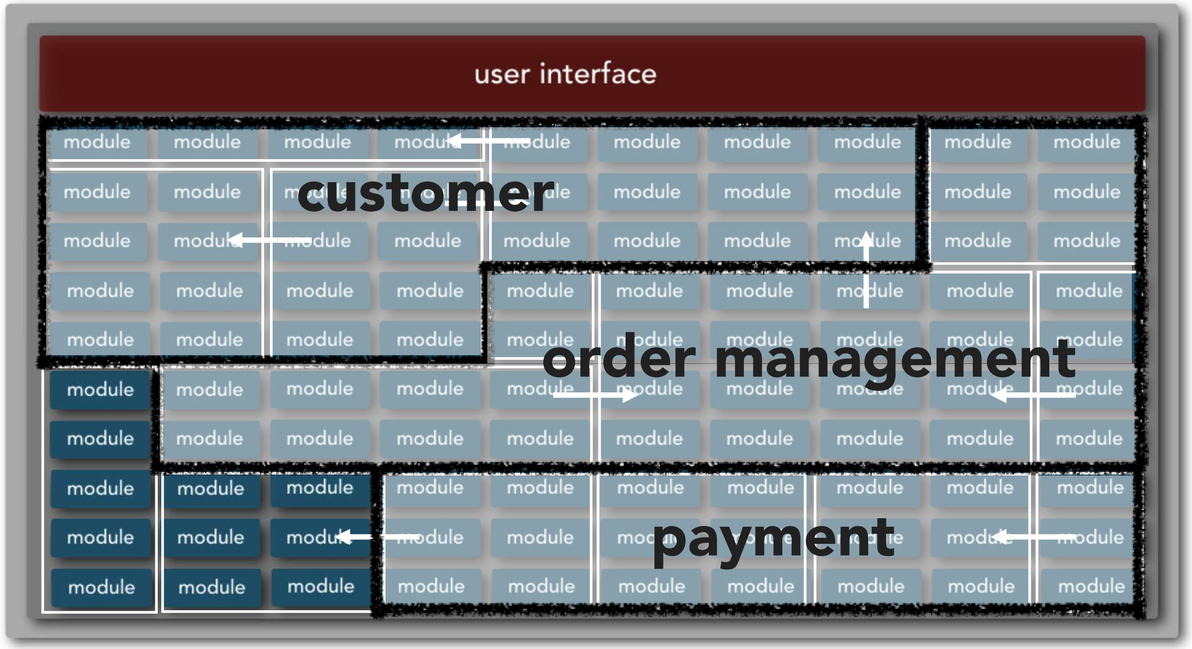
app.order.cancel

app.order.placement

app.customer.wishlist

migration patterns

identify component groups (domains)



app.payment.invoicing

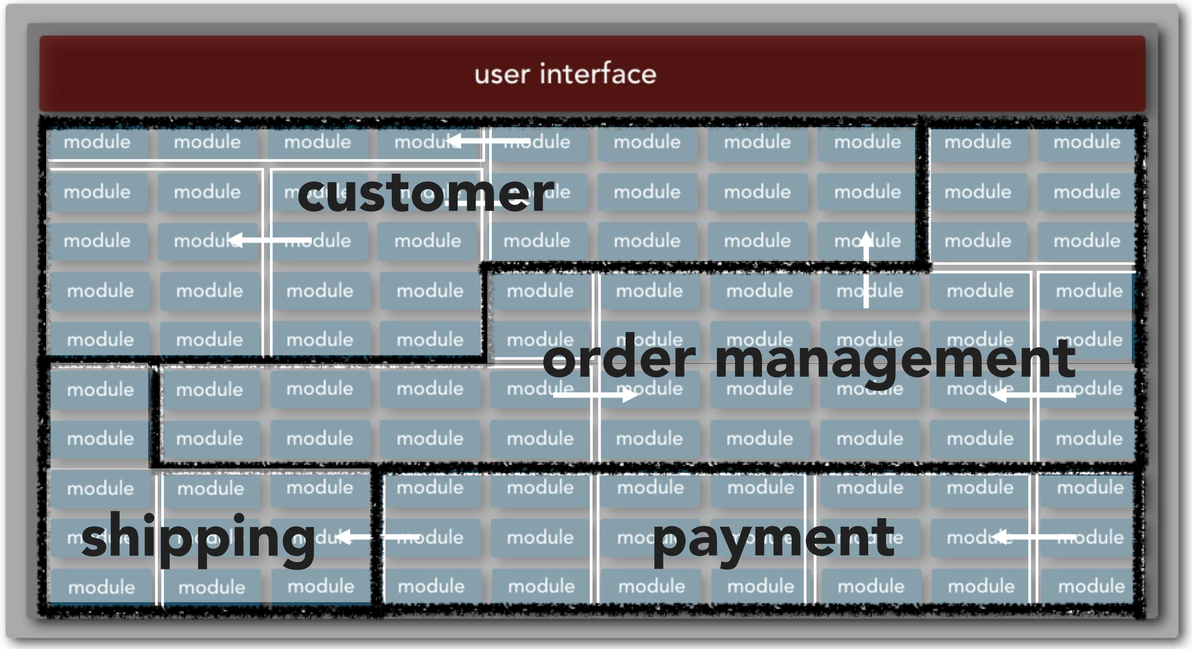
app.payment.history

app.payment.returns

app.payment.credit

migration patterns

identify component groups (domains)

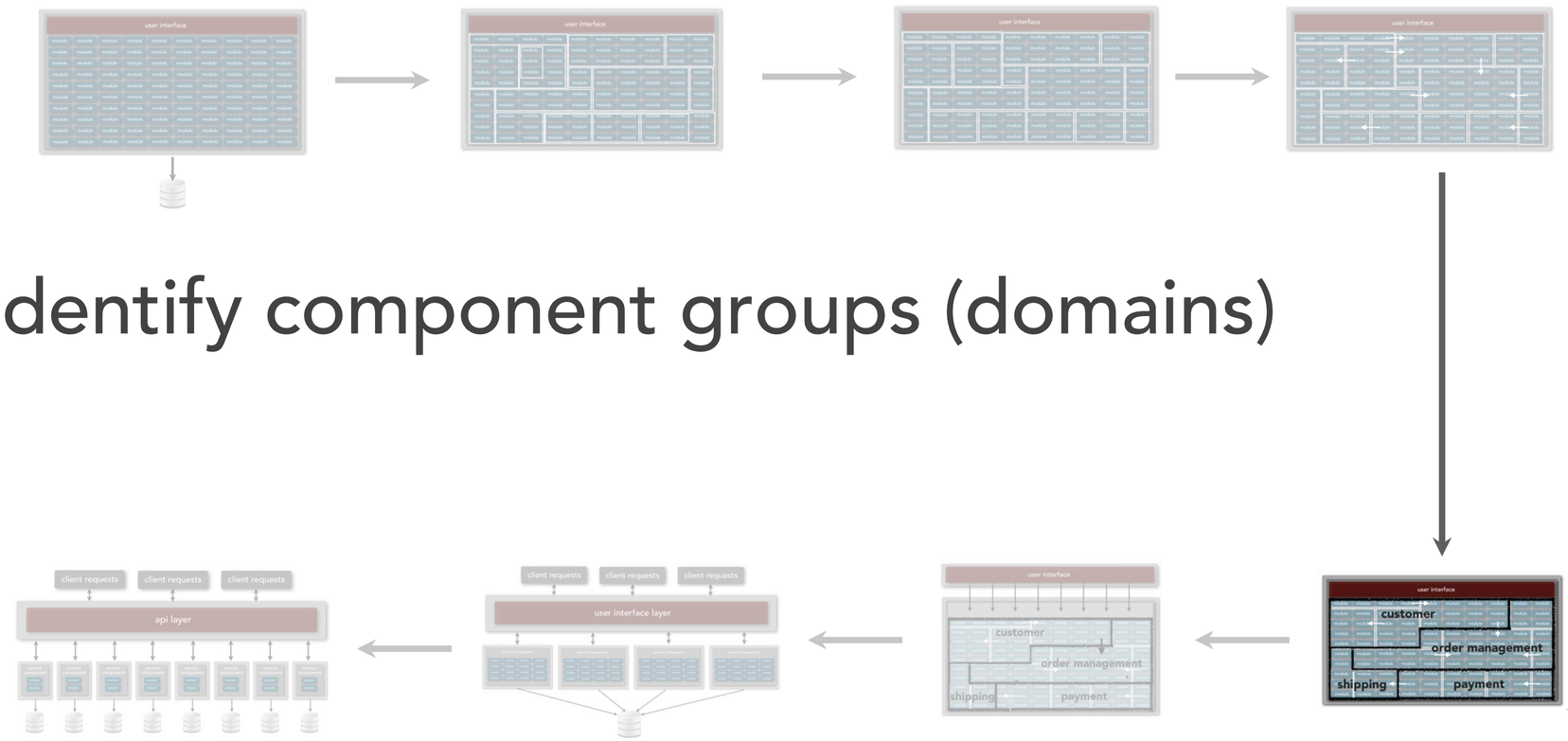


app.shipping.fulfillment

app.shipping.tracking

migration patterns

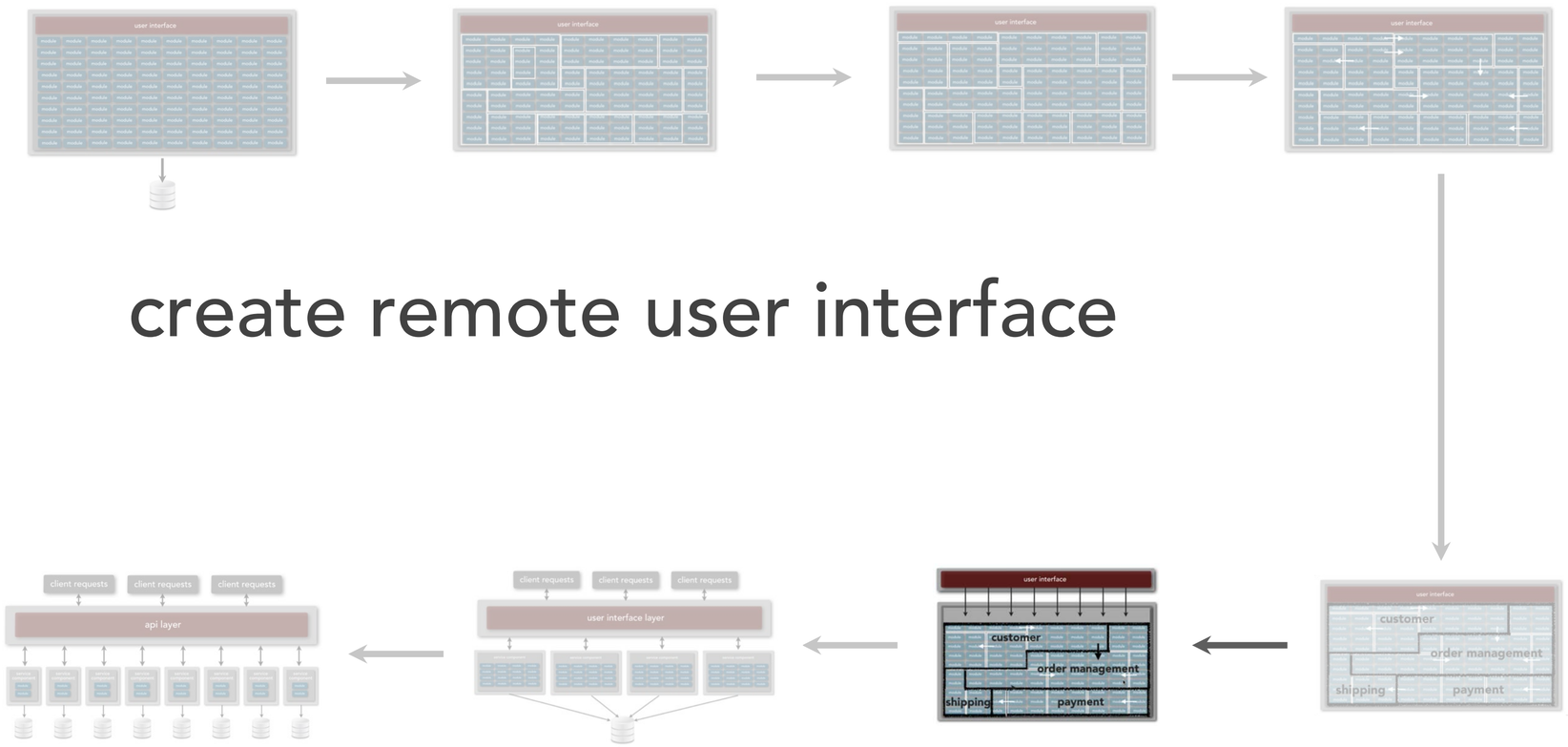
microservices roadmap



identify component groups (domains)

migration patterns

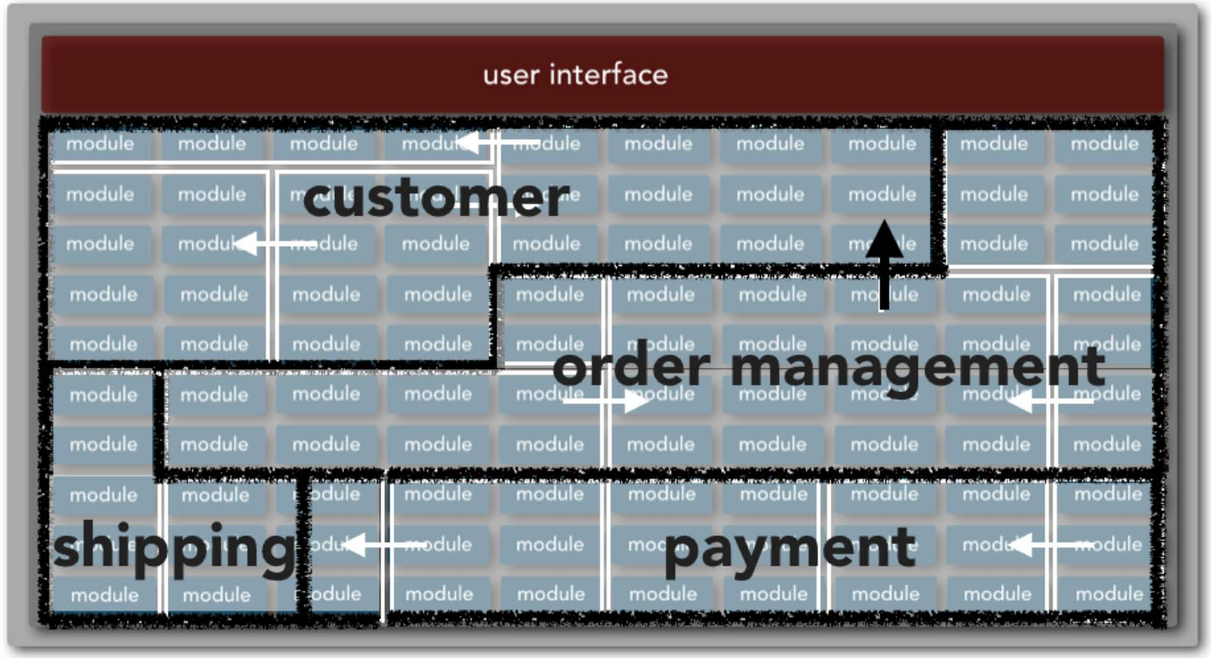
microservices roadmap



create remote user interface

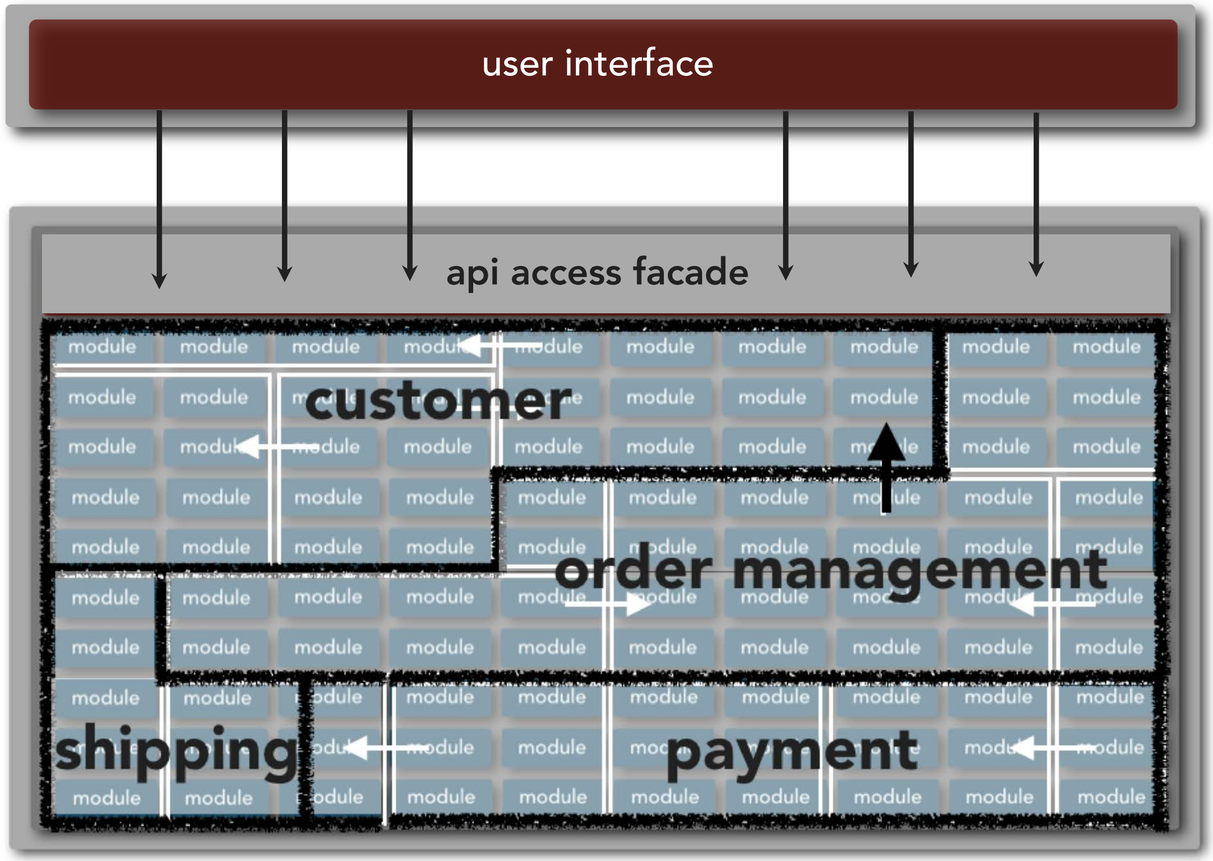
migration patterns

create remote user interface



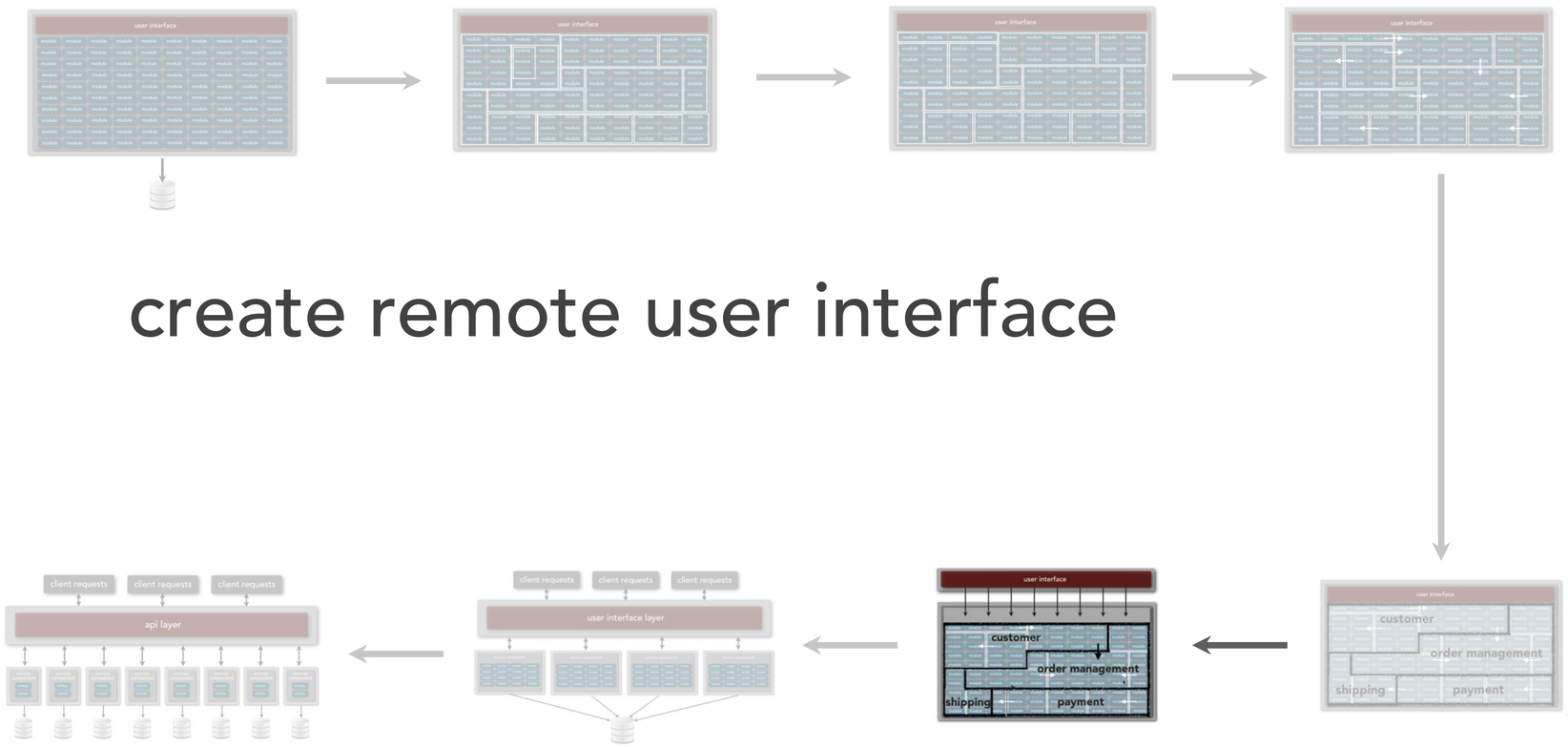
migration patterns

create remote user interface



migration patterns

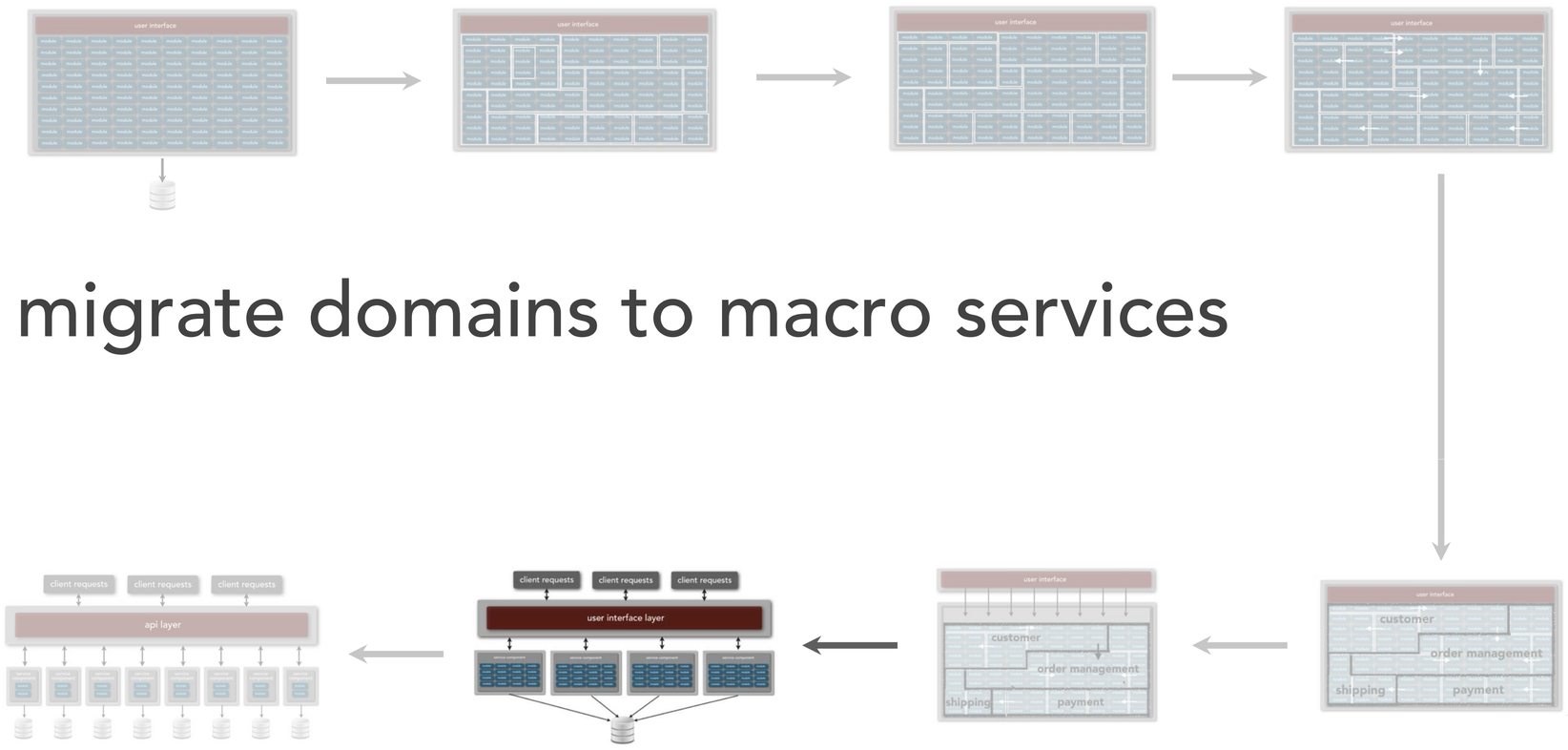
microservices roadmap



create remote user interface

migration patterns

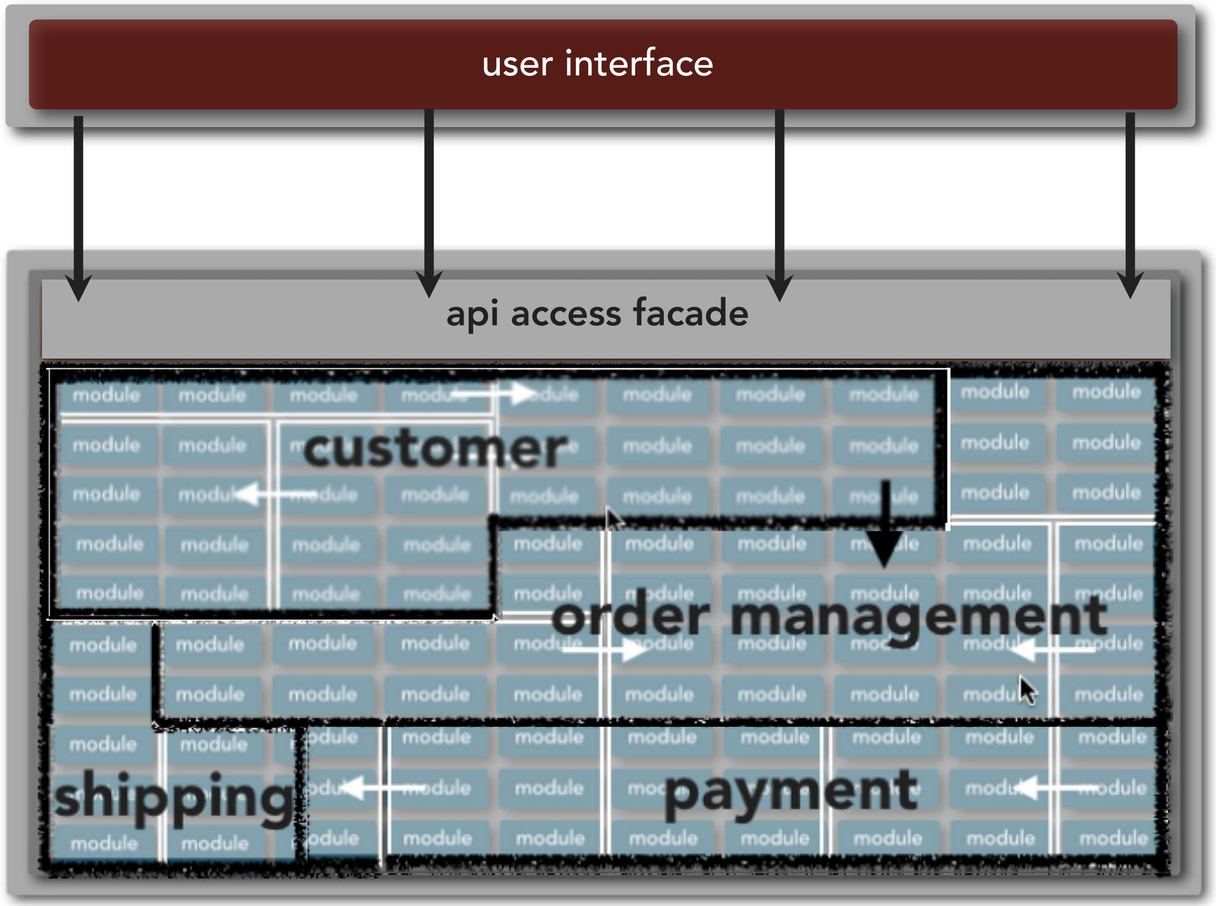
microservices roadmap



migrate domains to macro services

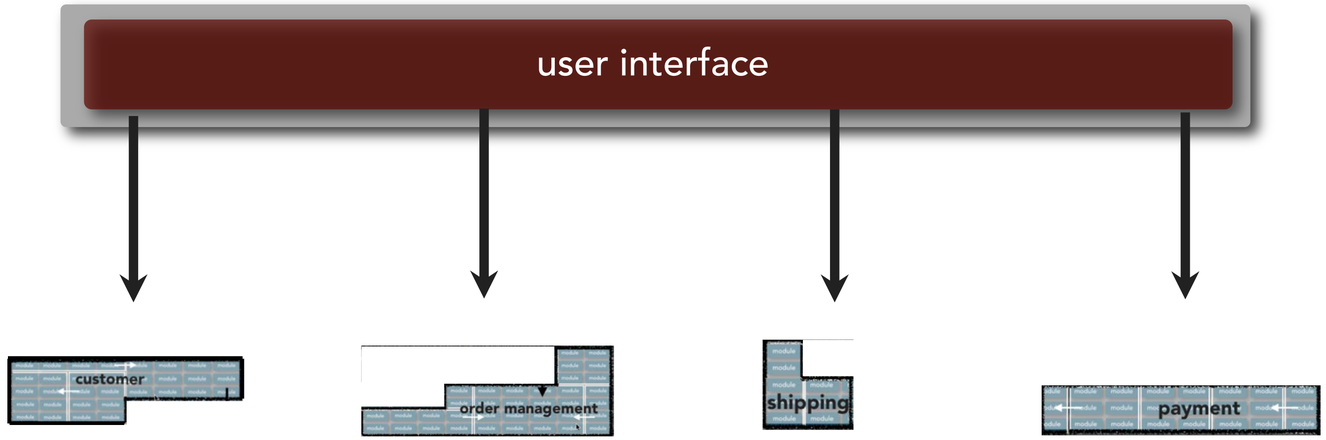
migration patterns

migrate domains to macro services



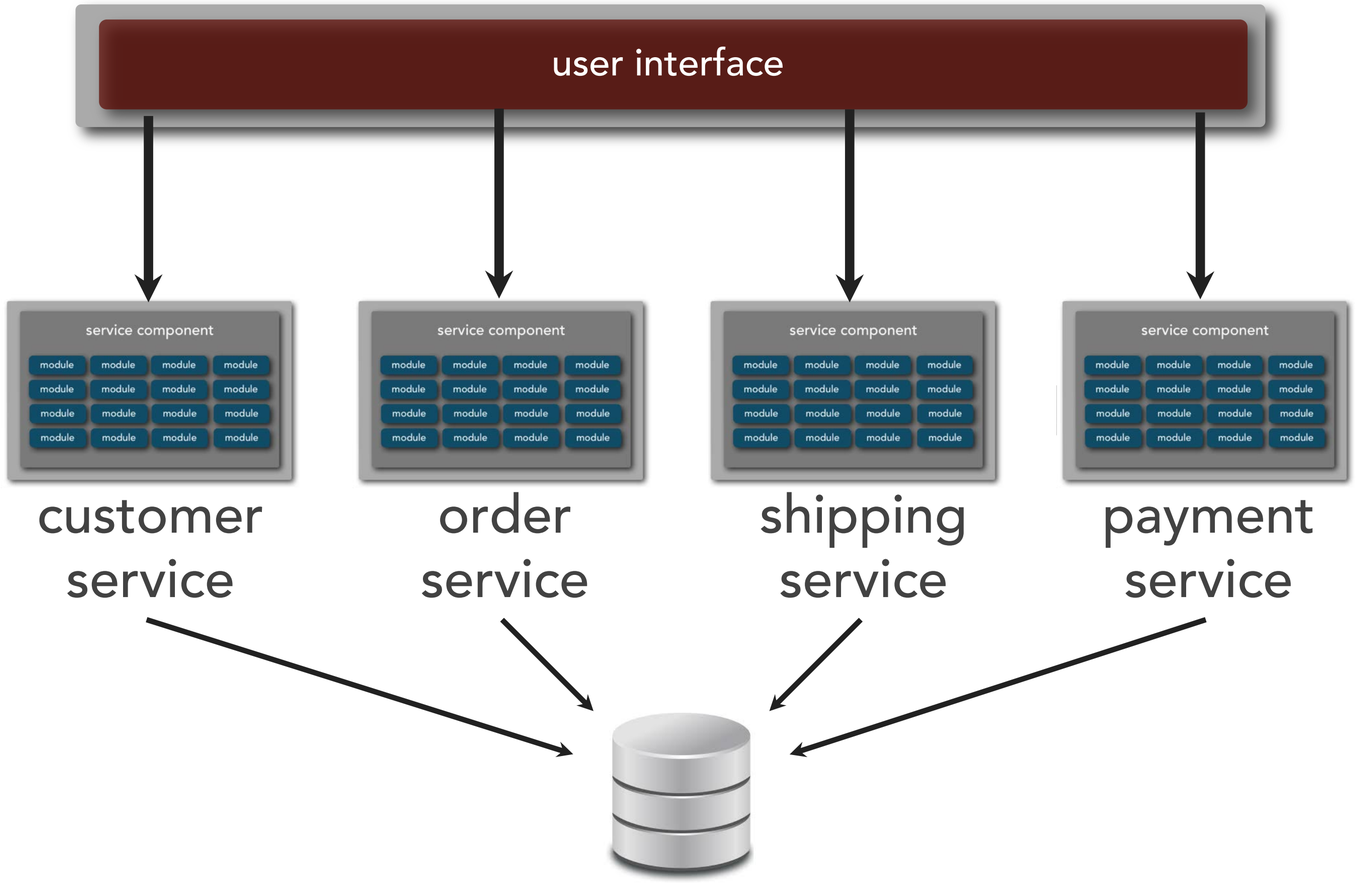
migration patterns

migrate domains to macro services



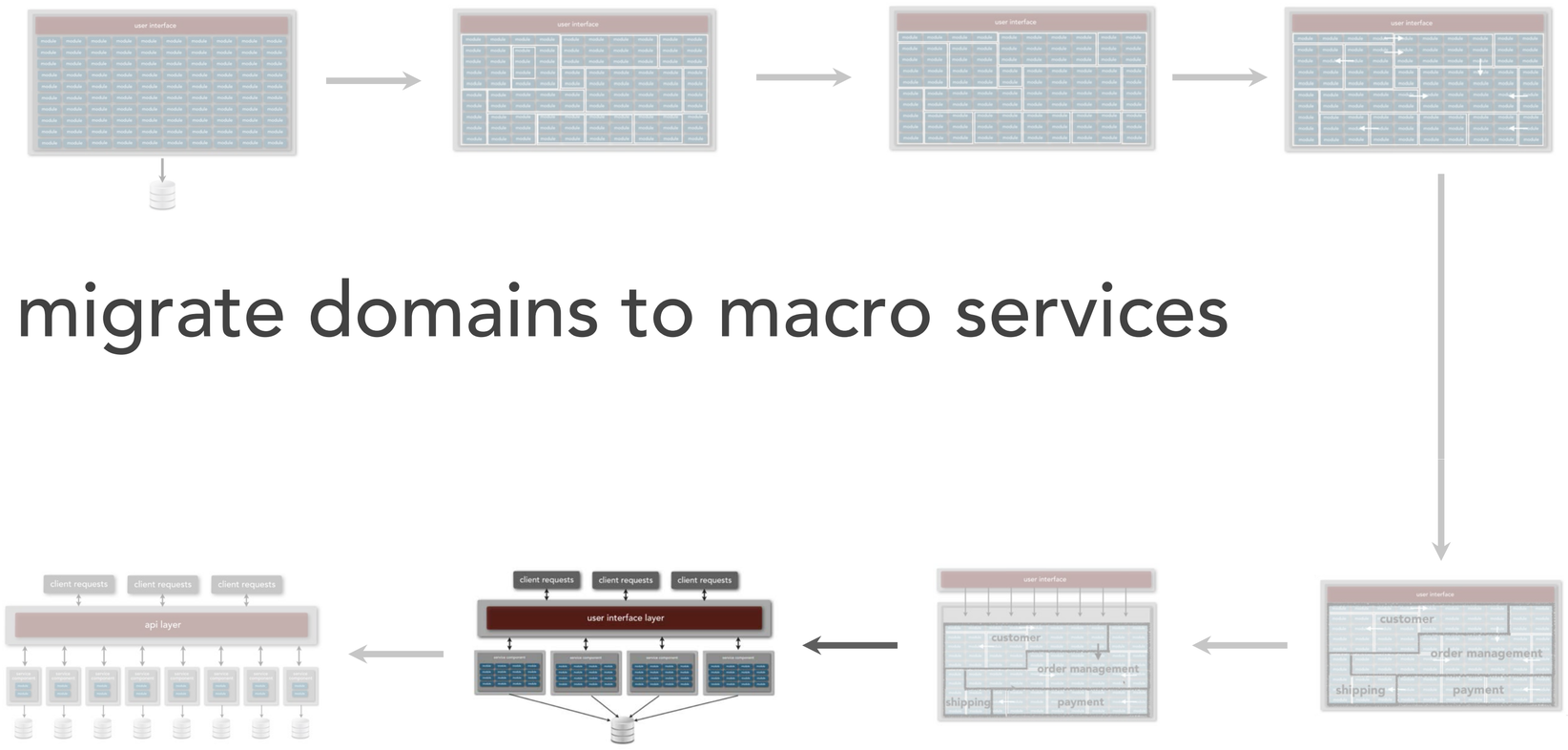
migration patterns

migrate domains to macro services



migration patterns

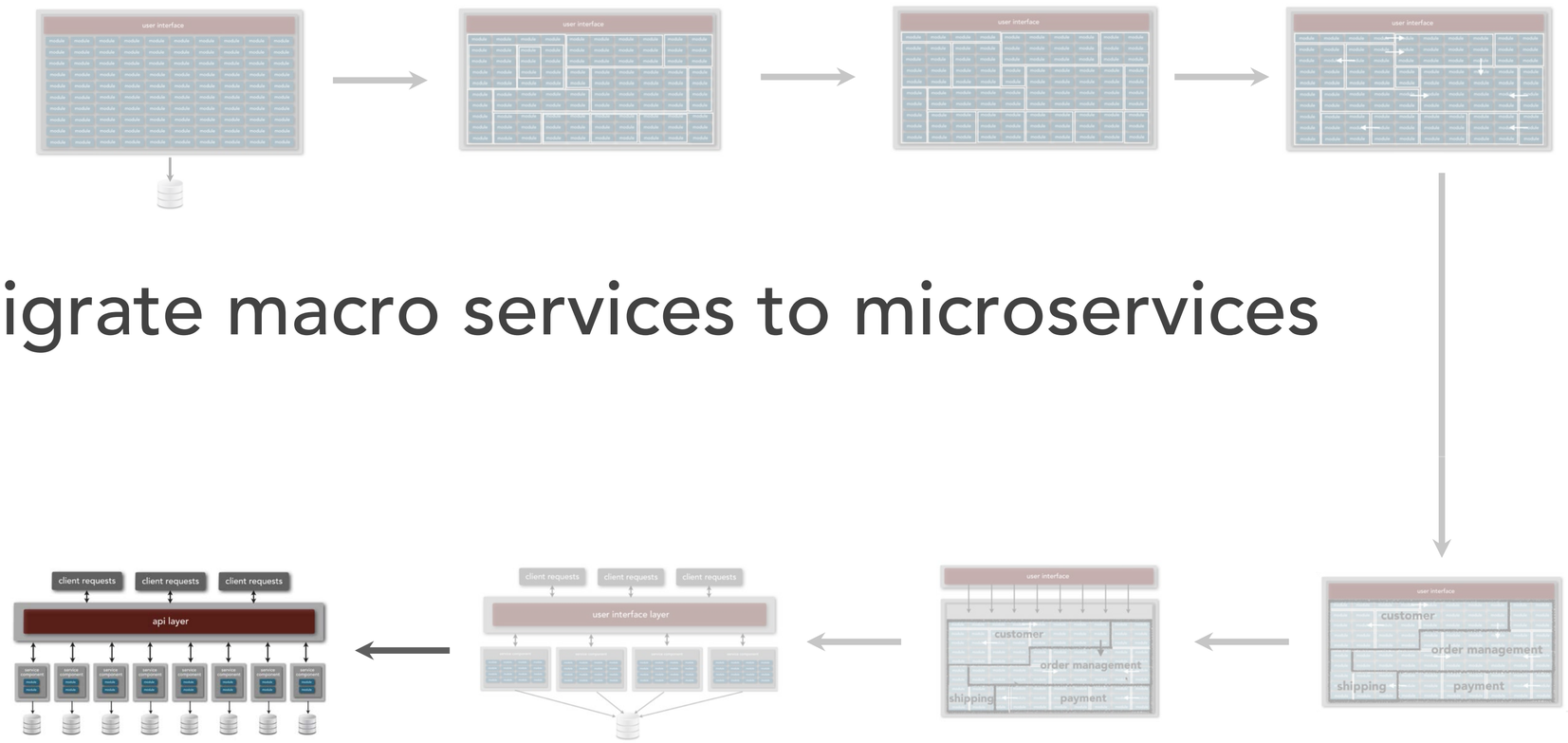
microservices roadmap



migrate domains to macro services

migration patterns

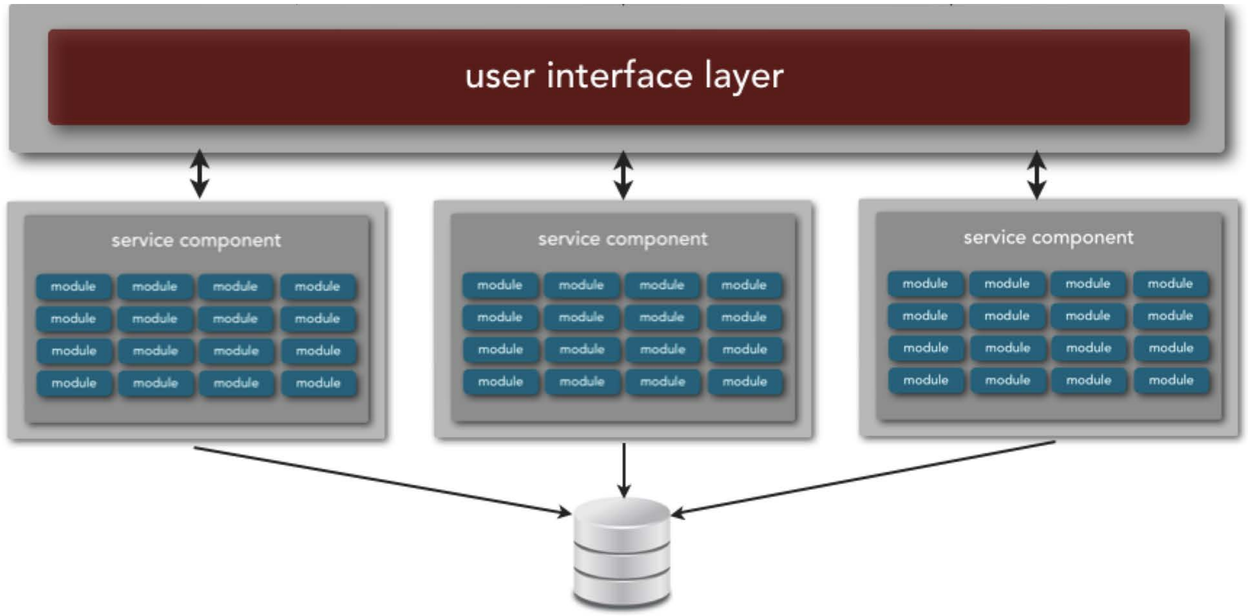
microservices roadmap



migrate macro services to microservices

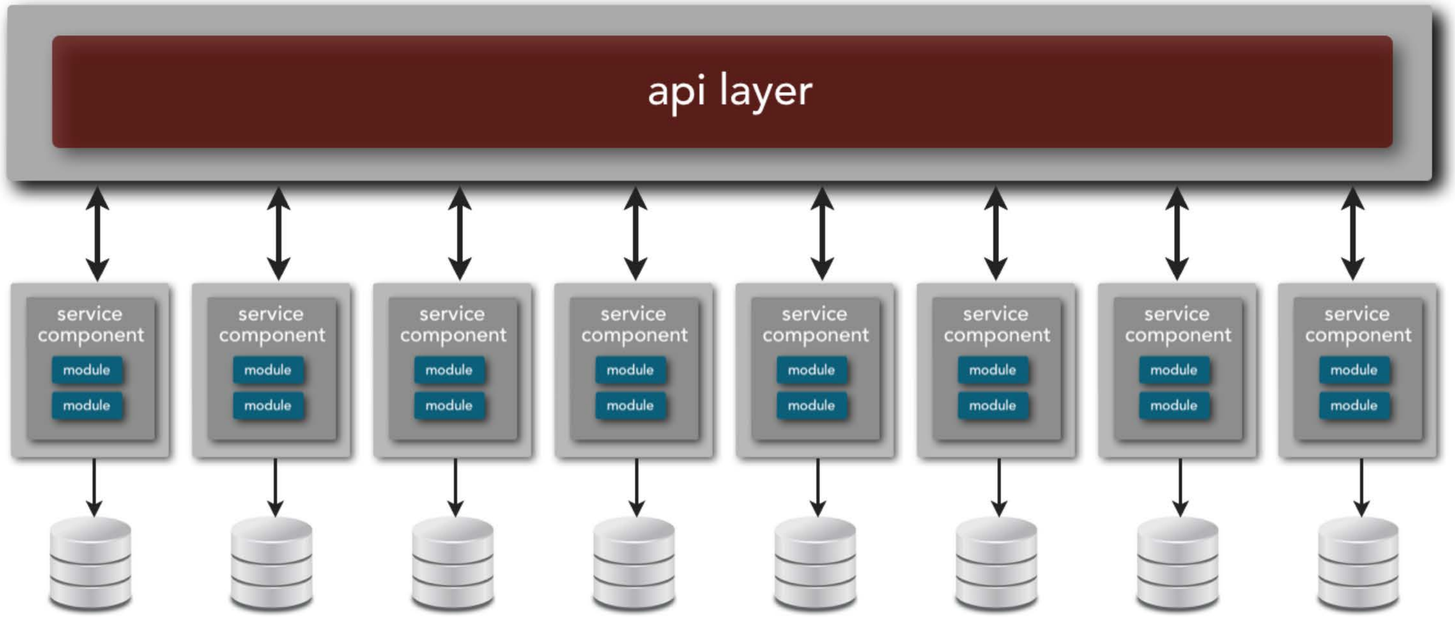
migration patterns

migrate macro services to microservices



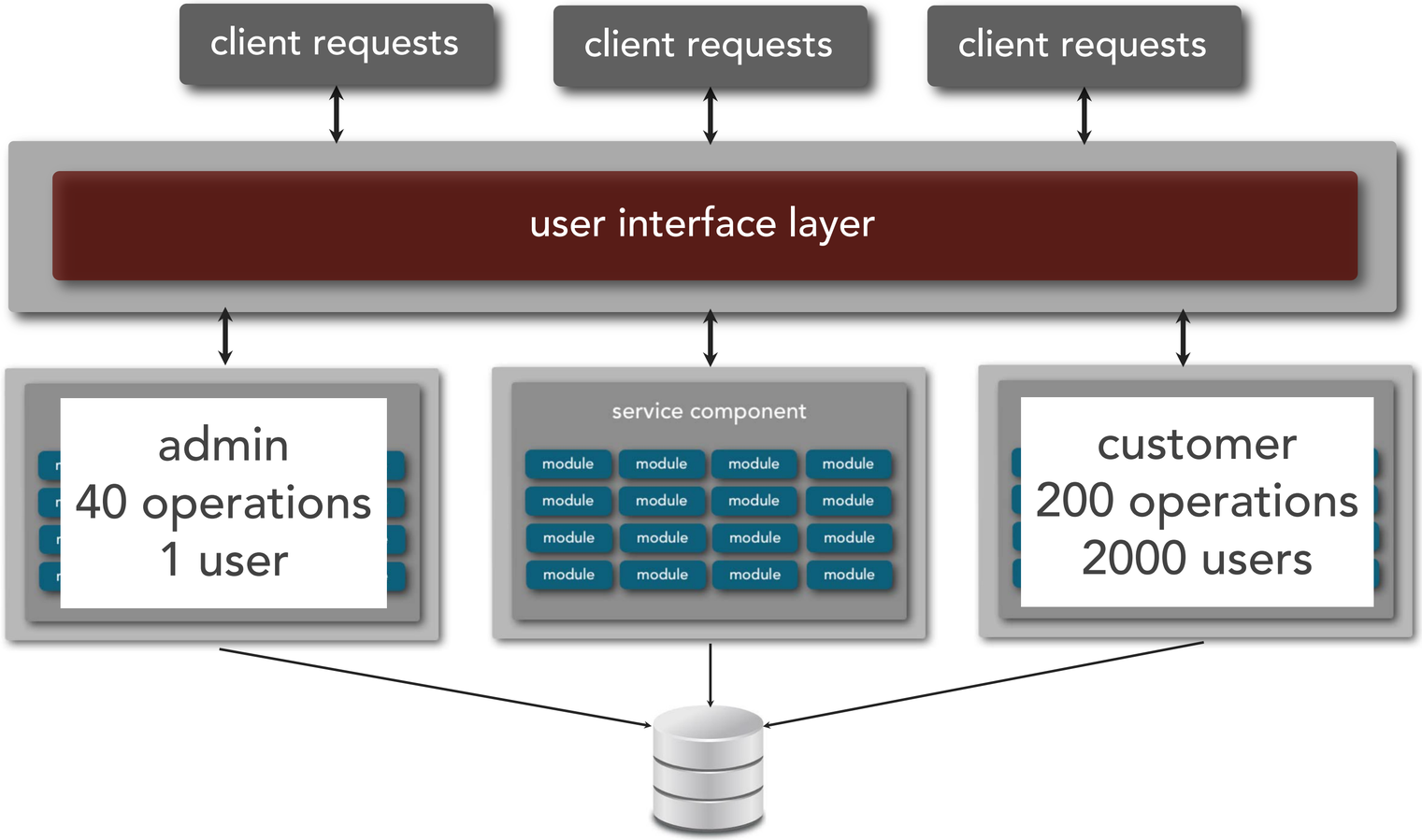
migration patterns

migrate macro services to microservices



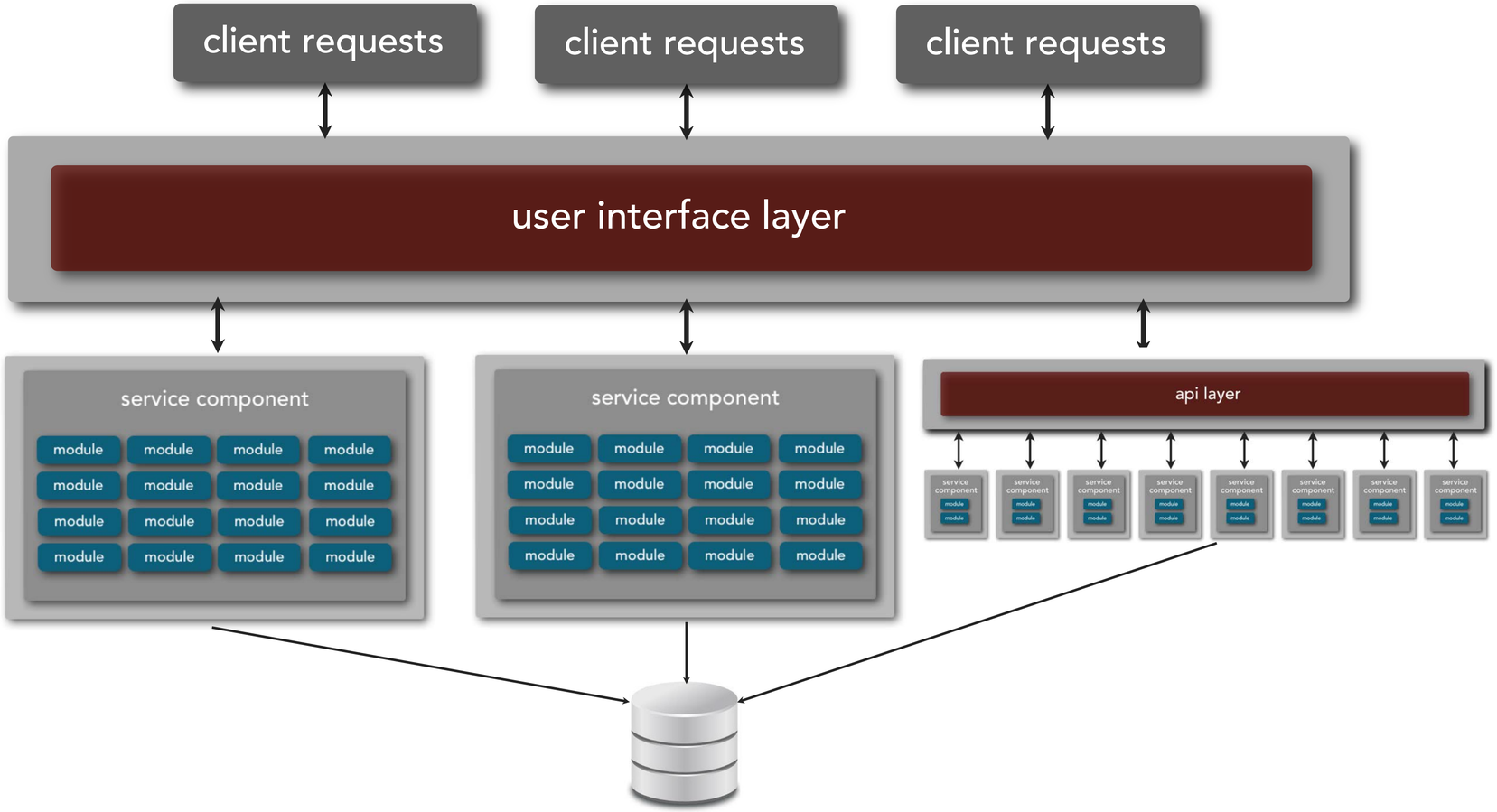
migration patterns

migrate macro services to microservices



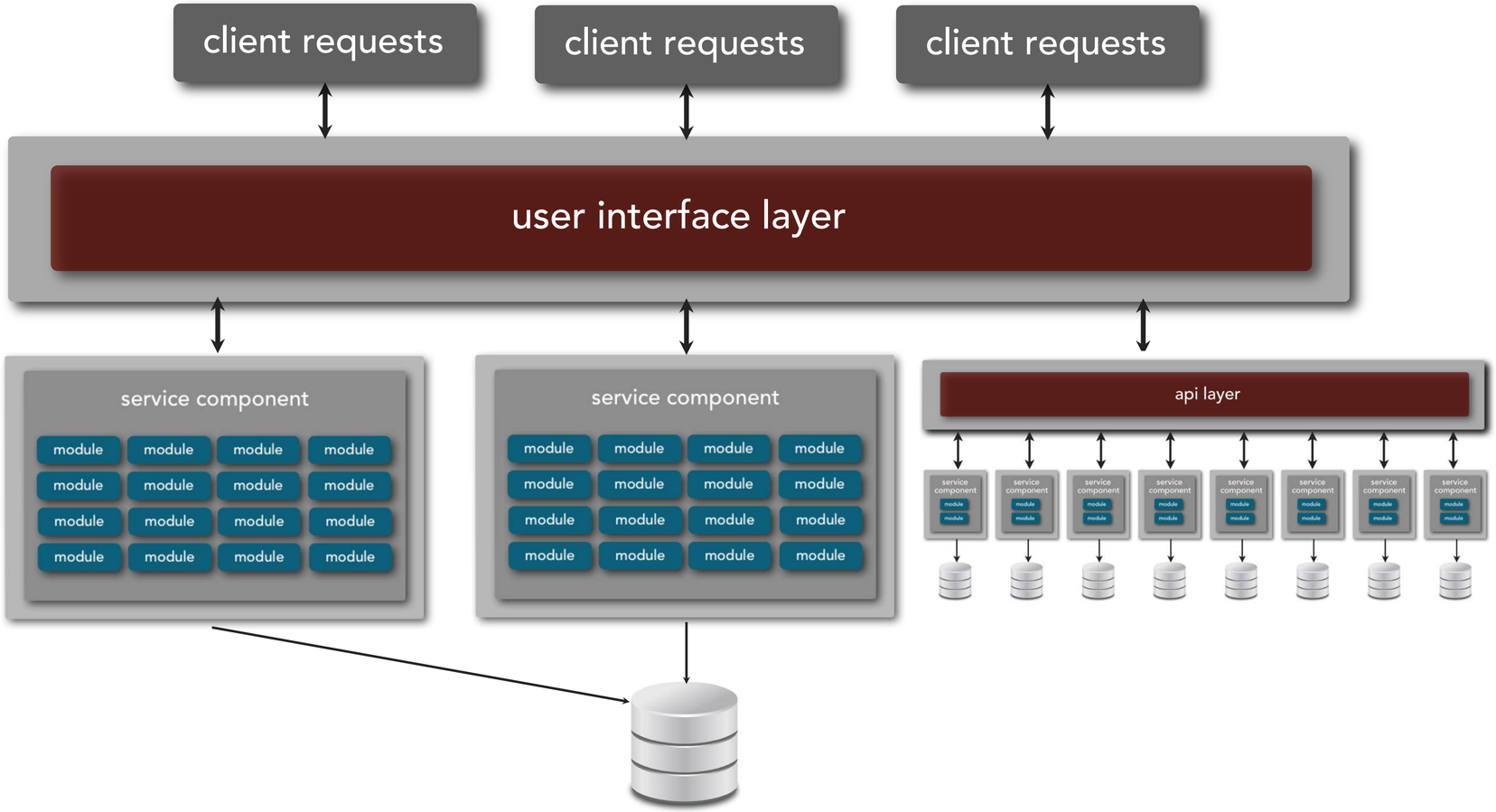
migration patterns

migrate macro services to microservices



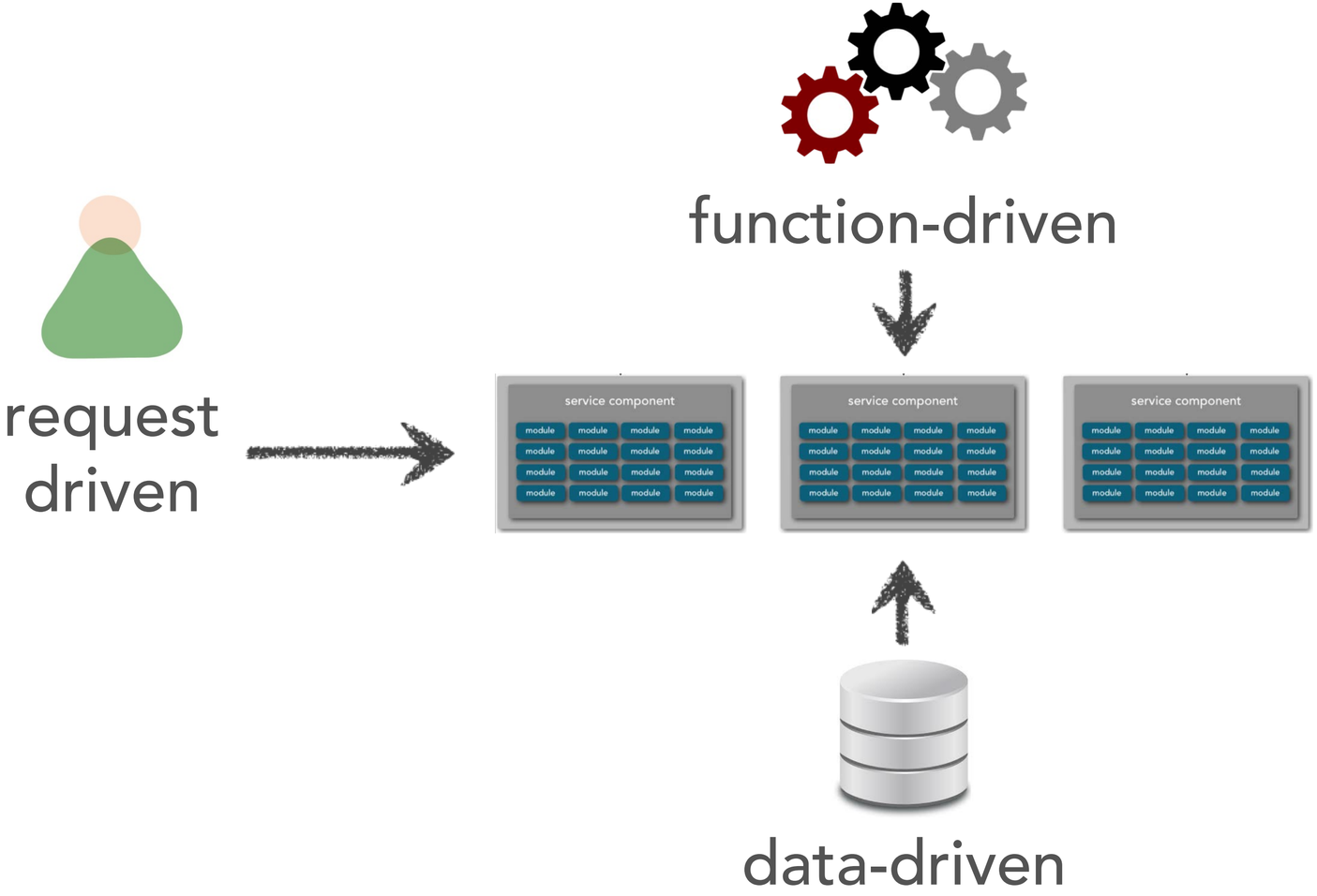
migration patterns

migrate macro services to microservices

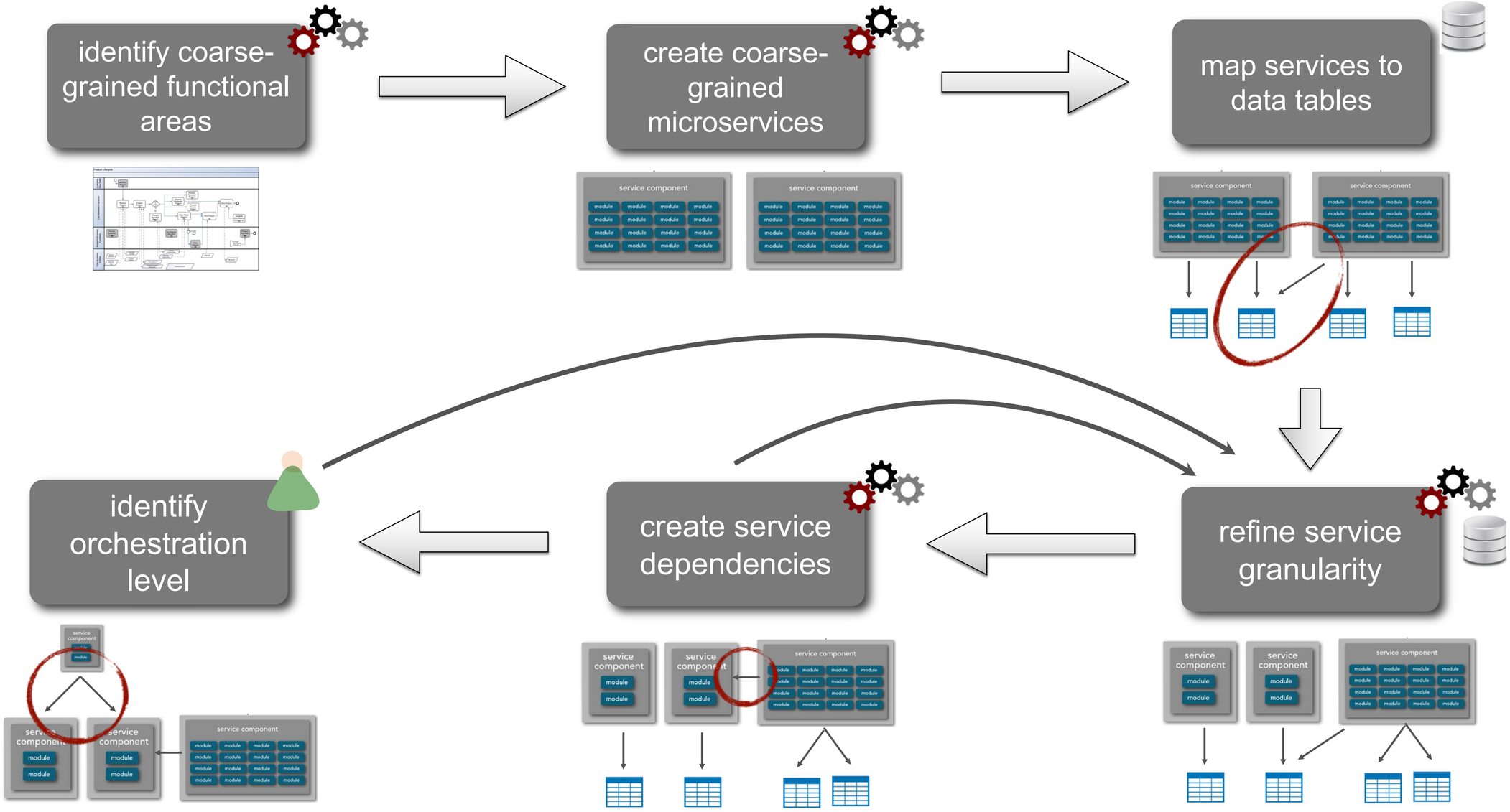


migration patterns

migrate macro services to microservices



migration patterns



Microservices Architecture and Design

Mark Richards

Independent Consultant

Hands-on Software Architect / Published Author

Founder, DeveloperToArchitect.com

http://www.wmrichards.com

https://www.linkedin.com/in/markrichards3

@markrichardssa