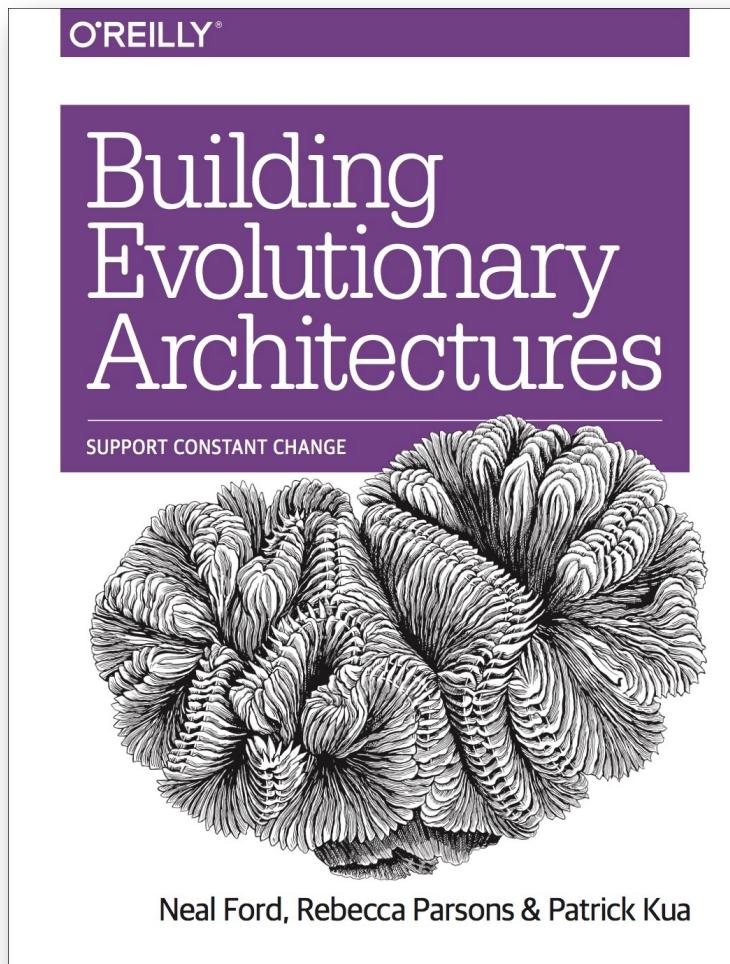


# bUiLdiNG eVoLuTiONaRy ARcHiEcTuREs wOrkSHoP



<http://evolutionaryarchitecture.com/downloads/BEA-1day.pdf>



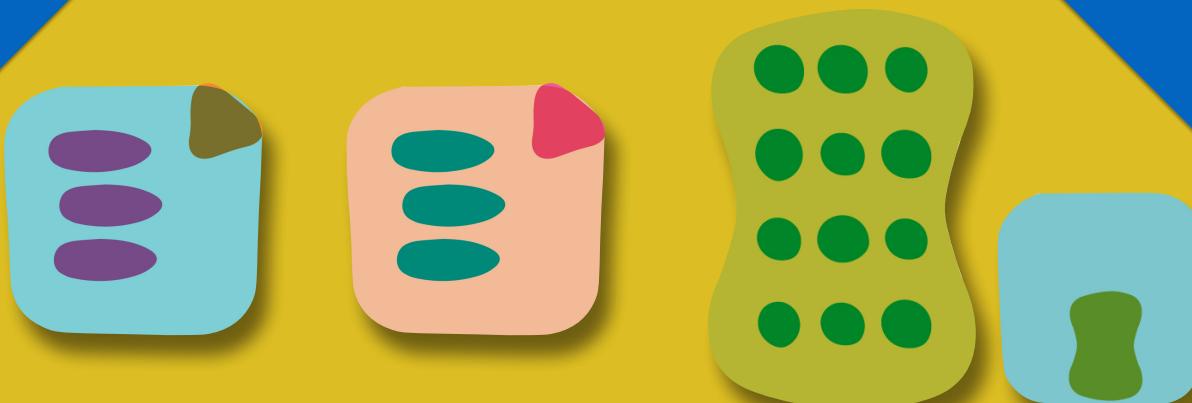
photos by Martin Fowler

# Change

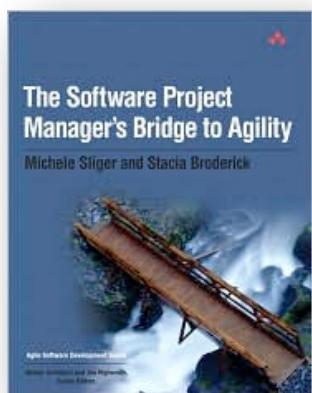
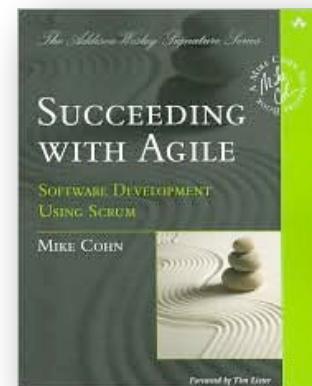
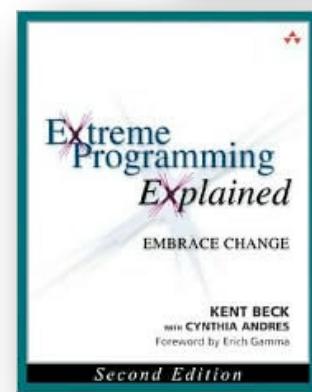
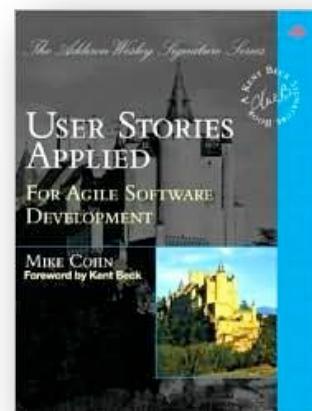
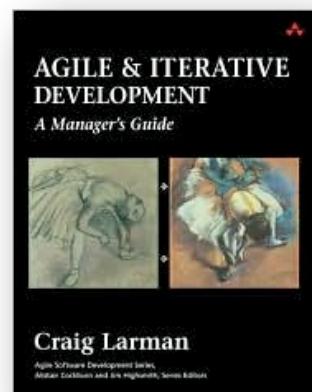
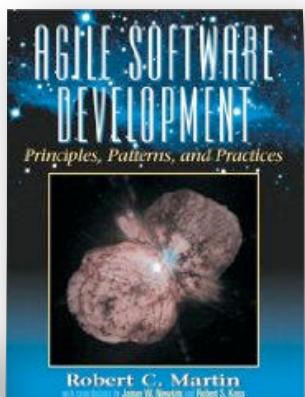
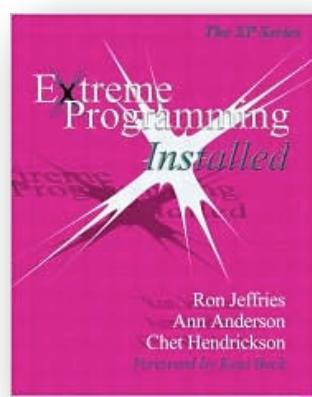
technology  
driven

business driven

technology  
driven



business driven



Agile Software Development with Scrum

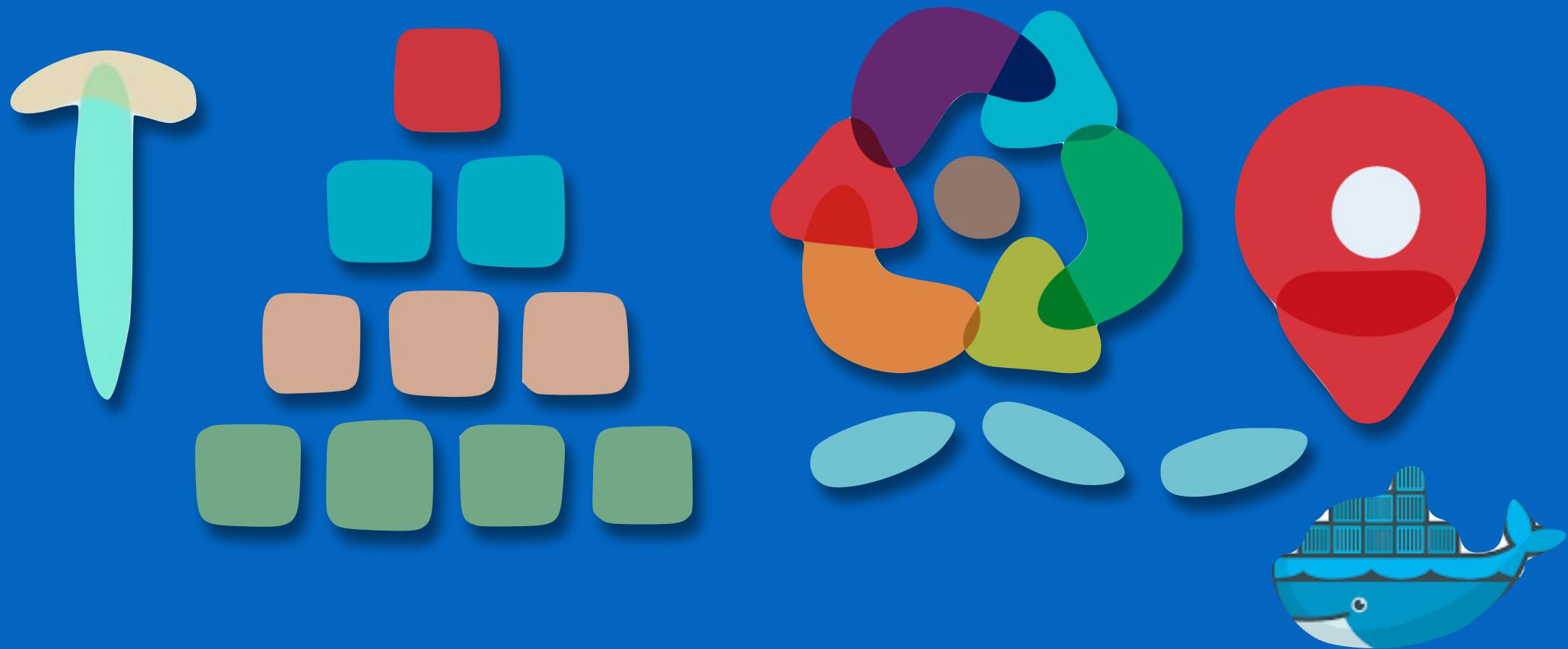
red  
yellow  
green  
blue  
red  
blue  
yellow  
green  
blue

Ken Schwaber ■■■ Mike Beedle

technology  
driven

business driven

# dYNaMic equUiLIBRiuM



# dYNaMic equUiLIBRiuM



everything changes  
all the time!

**How is long term planning  
possible when things constantly  
change in unExpECtEd ways?**

# Agenda

architecture characteristics



definition

fitness functions



fitness function as governance

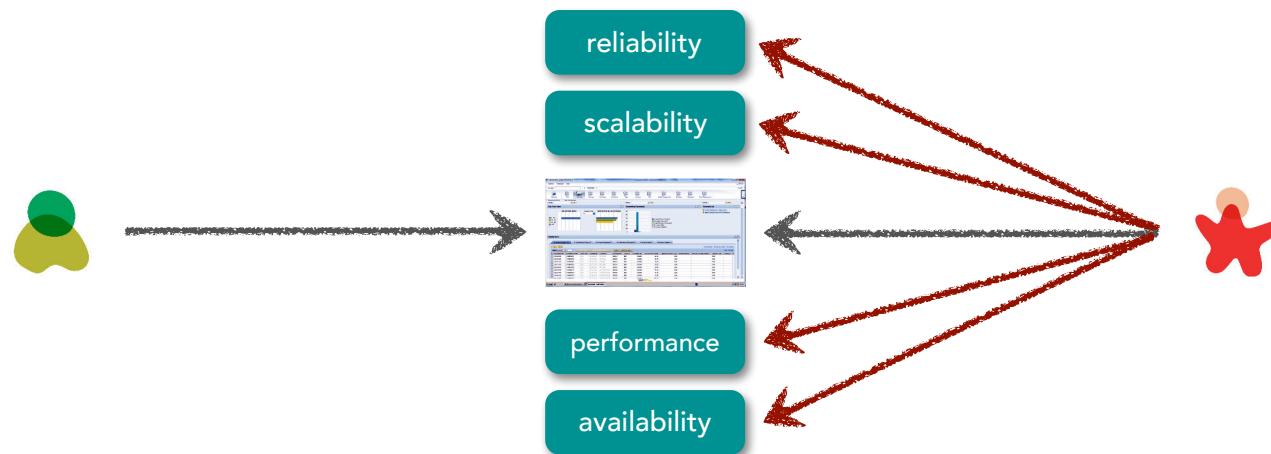
architecture quantum



evolutionary data

implementing  
evolutionary architectures

# architecture characteristics



# architecture characteristics

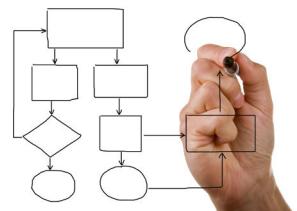
accessibility	evolvability	repeatability
accountability	extensibility	reproducibility
accuracy	failure transparency	resilience
adaptability	fault-tolerance	responsiveness
administrability	fidelity	reusability
affordability	flexibility	robustness
agility	inspectability	safety
auditability	installability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

[https://en.wikipedia.org/wiki/List\\_of\\_system\\_quality\\_attributes](https://en.wikipedia.org/wiki/List_of_system_quality_attributes)

# architecture characteristics



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

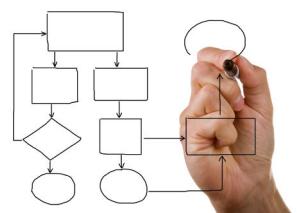


agility, modularity, extensibility,  
maintainability, testability,  
deployability, flexibility, adaptability

# architecture characteristics



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

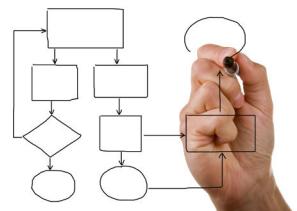


performance, scalability, availability,  
reliability, recoverability, auditability

# architecture characteristics



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

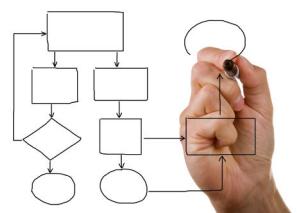


scalability, extensibility, openness,  
standards-based, agility, modularity

# architecture characteristics



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



feasibility





# architecture characteristics

feasibility



agility

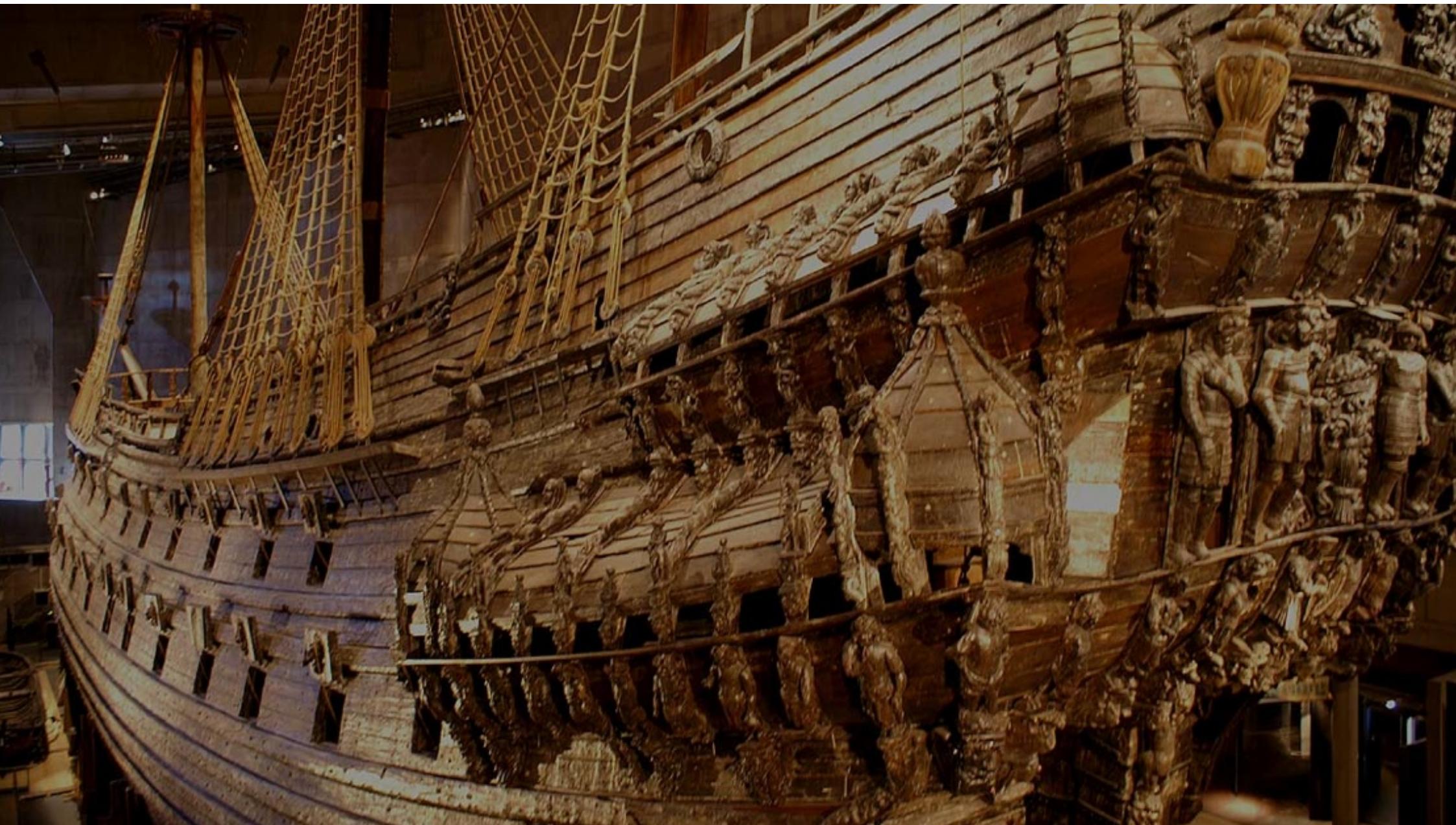


elasticity



scalability







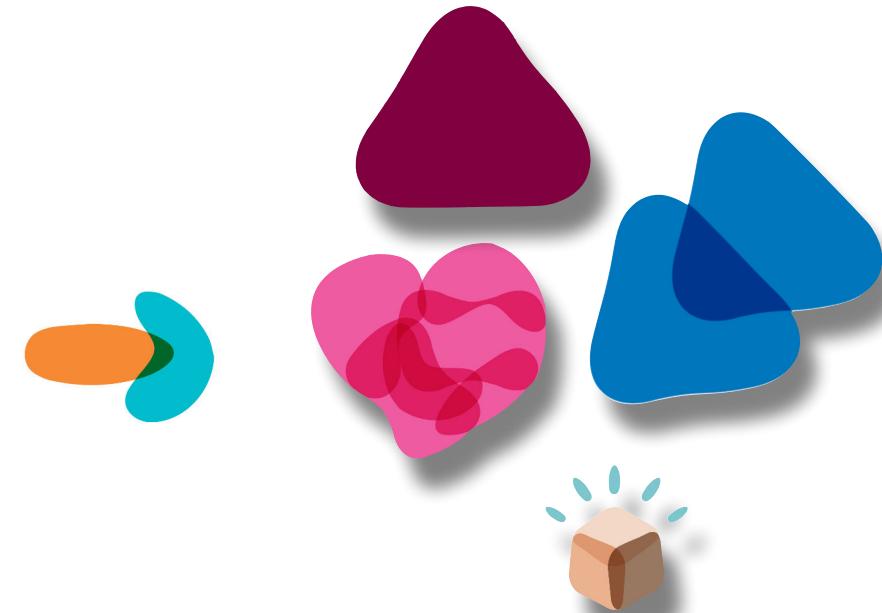
# Identify Architectural Characteristics

Your Architectural Kata is...

## Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale—customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained—this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud



Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

# Going Going Gone!

?

An auction company wants to take their **auctions online** to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - **auctions must be as real-time as possible** ?
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their **auctions online** to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - **auctions must be as real-time as possible**
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

*availability   reliability   performance*

Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability   reliability   performance

Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

*availability reliability performance scalability*

Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.



- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

*availability reliability performance scalability*

Your Architectural Kata is...

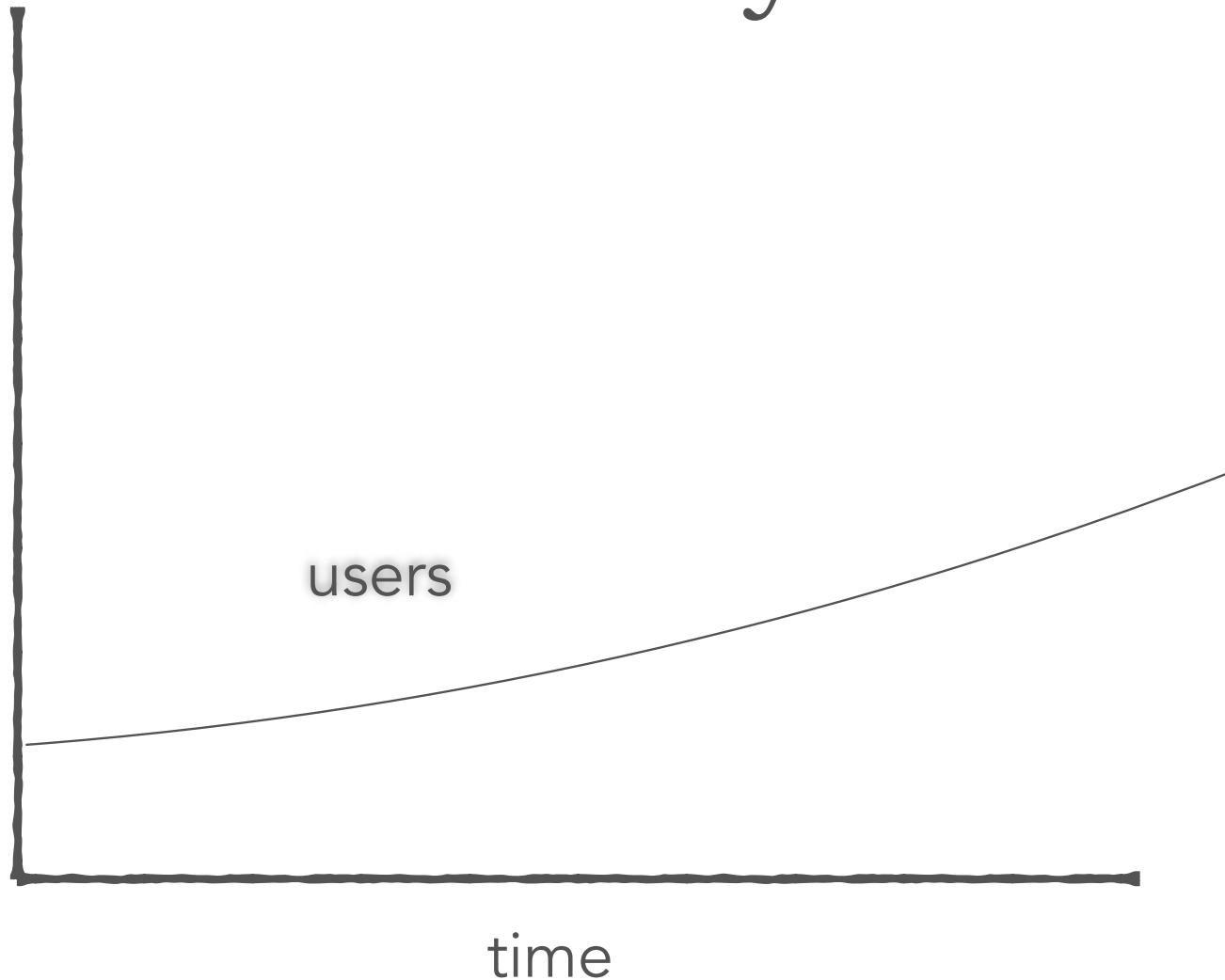
# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

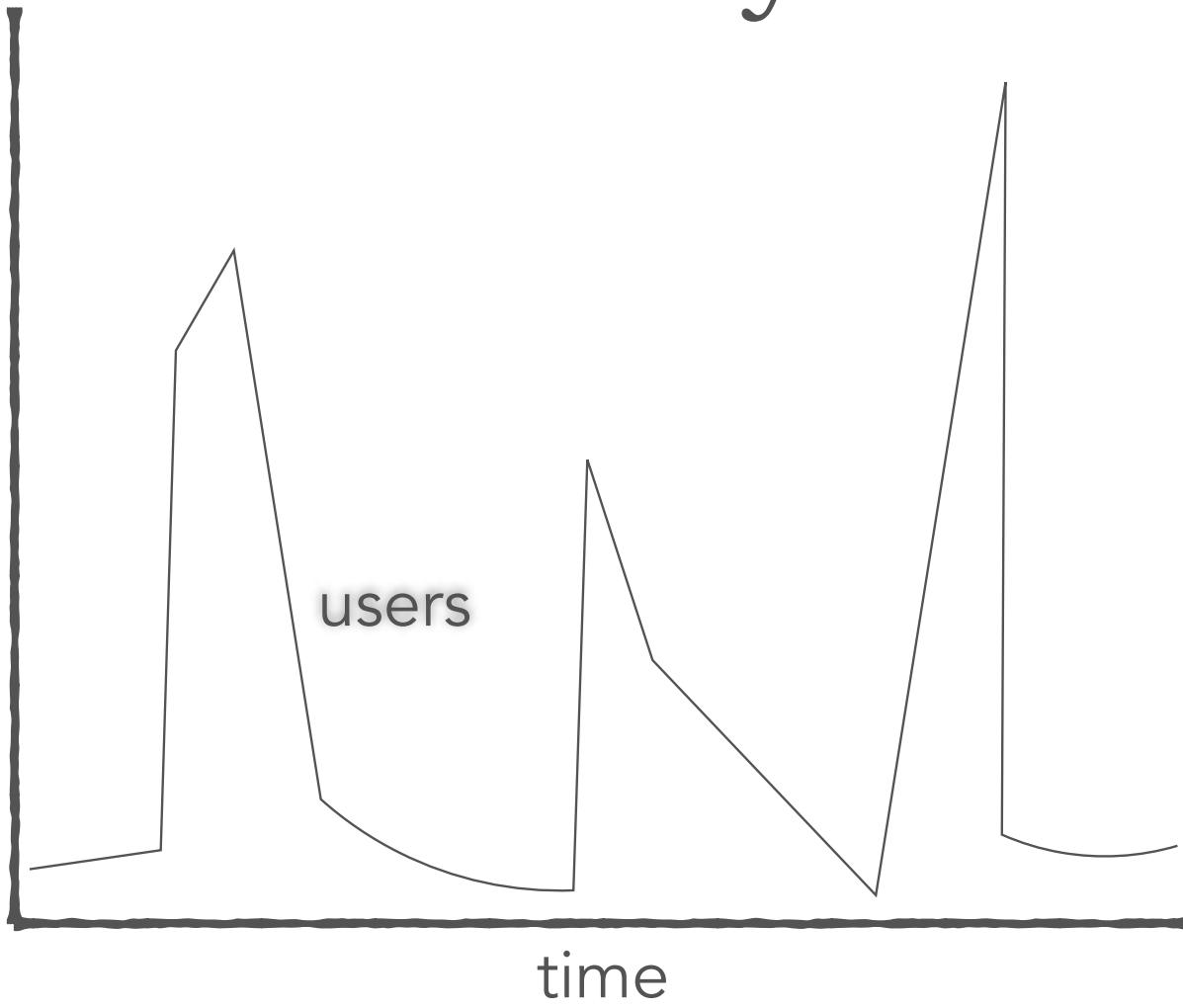
- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

*availability reliability performance scalability elasticity*

# scalability:



# elasticity:



Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index ?
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability elasticity

Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

?

availability reliability performance scalability elasticity

Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability elasticity  
(security)

Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability elasticity  
(security)

Continuous Architecture?

Incremental Architecture?

Agile Architecture?

Adaptable Architecture?

**Evolutionary Architecture**

# Evolutionary Architecture

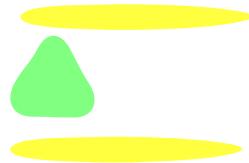
An evolutionary architecture supports  
guided,  
incremental change  
across multiple dimensions.



# Evolutionary Architecture

An evolutionary architecture supports  
guided  
incremental change  
across multiple dimensions.



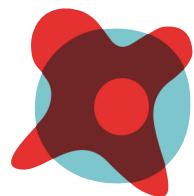
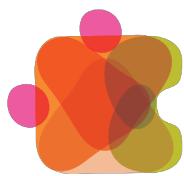


guided

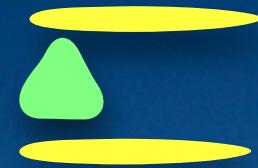
## *evolutionary computing fitness function:*

a particular type of objective function that is used to summarize...how close a given design solution is to achieving the set aims.

# Traveling Salesman Problem



fitness function = length of route

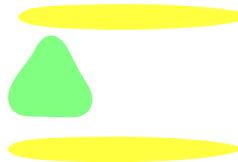


guided





*evolutionary computing fitness function:*  
a particular type of objective function that is  
used to summarize...how close a given  
design solution is to achieving the set aims.

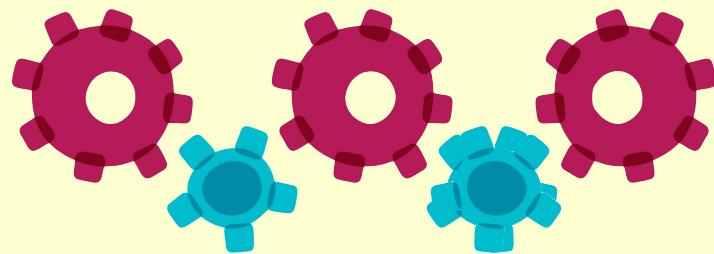


guided

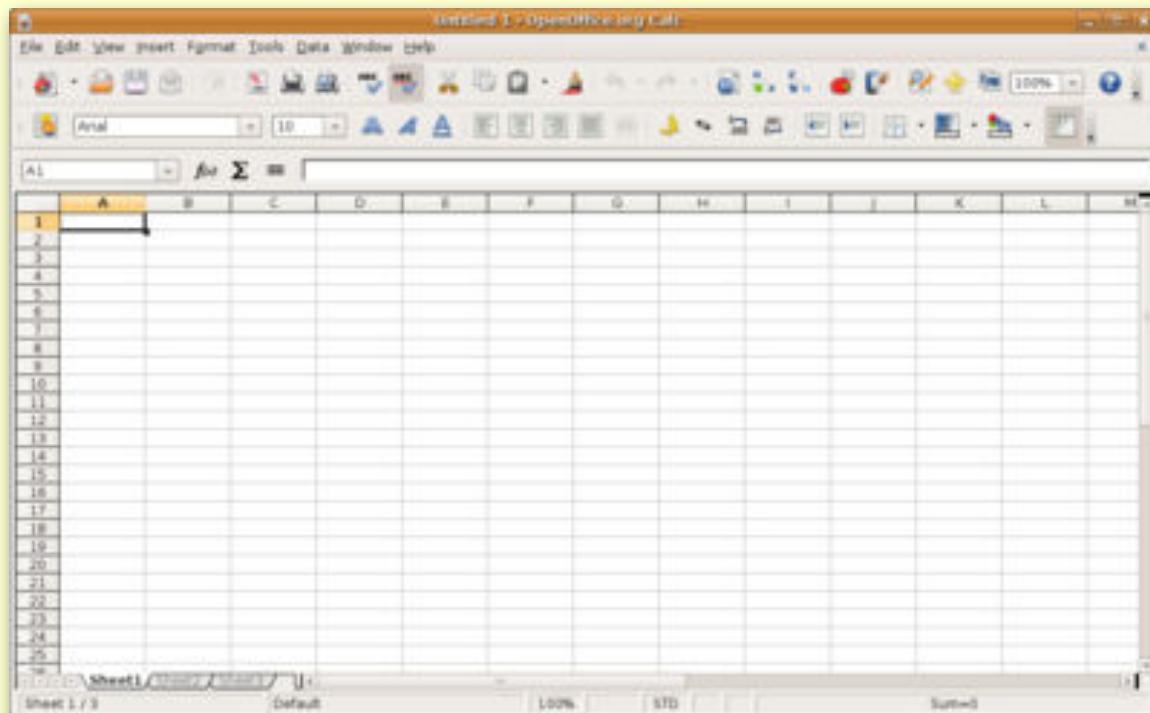
## **architectural fitness function:**

An architectural fitness function provides an objective integrity assessment of some architectural characteristic(s).

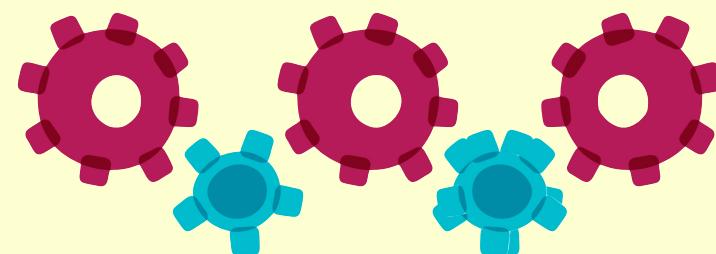
Penultima ↑ e



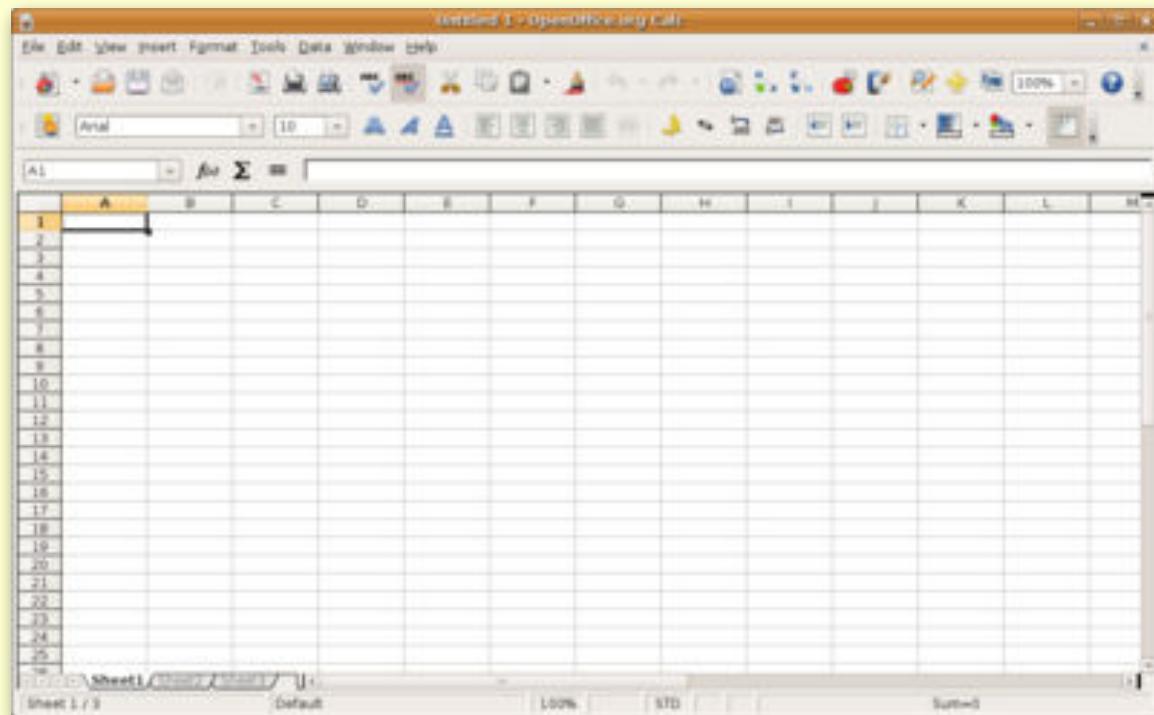
# EA Spreadsheet



Penultima ↑ e

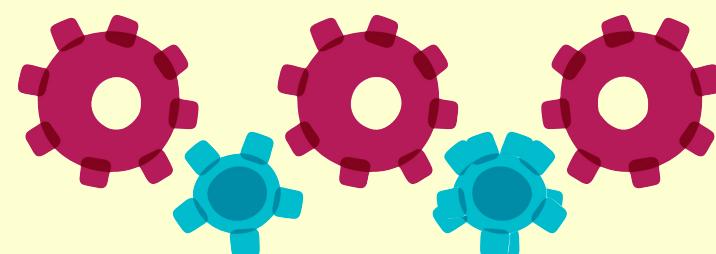


# EA Spreadsheet

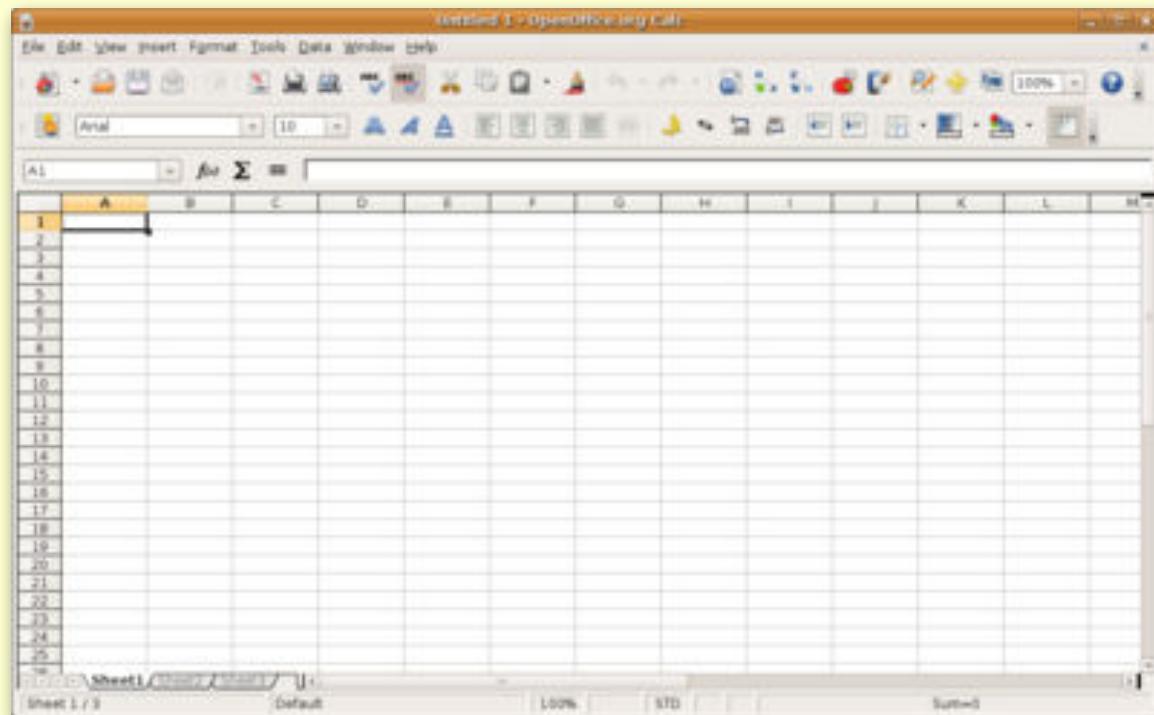


✓ definition

Penultima ↑ e



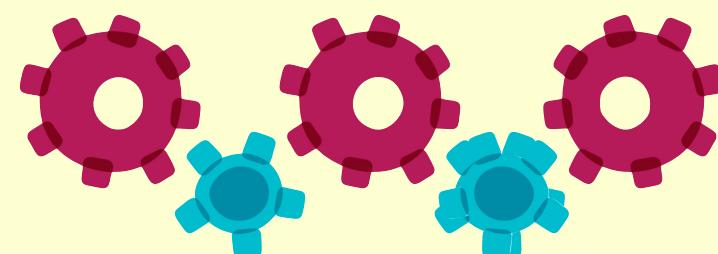
# EA Spreadsheet



✓ definition

! verification

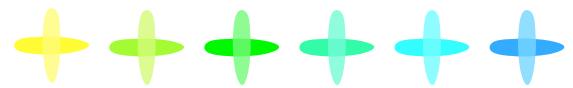
Penultima ↑ e



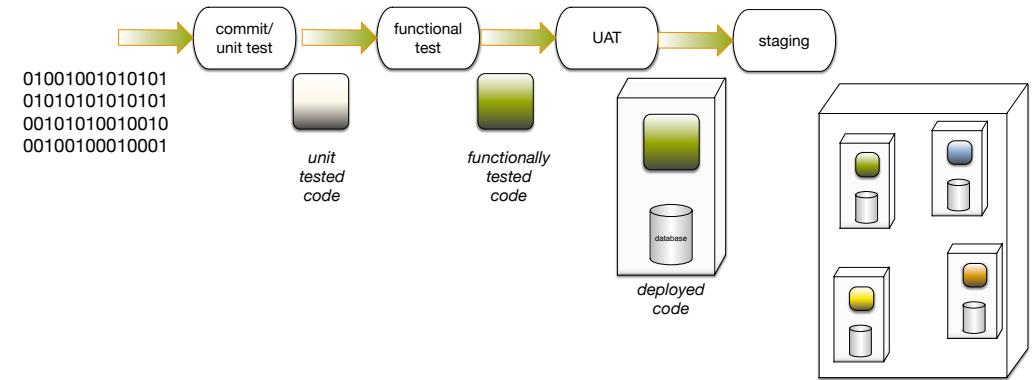
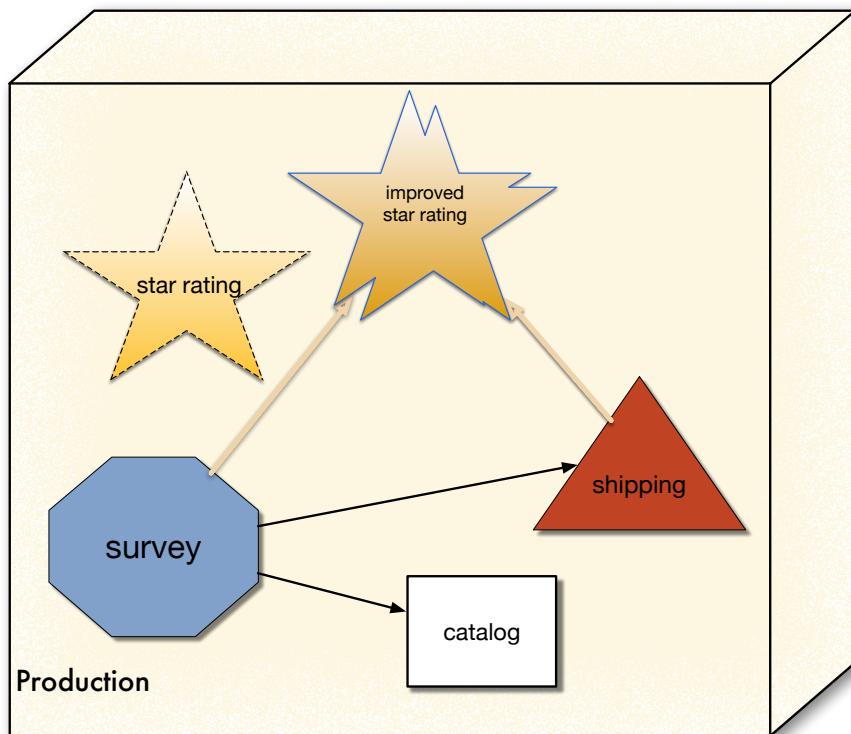
# Evolutionary Architecture

An evolutionary architecture supports  
guided  
incremental change across multiple dimensions.





# incremental



# Evolutionary Architecture

An evolutionary architecture supports  
guided,

incremental change    + + + + +  
across multiple dimensions

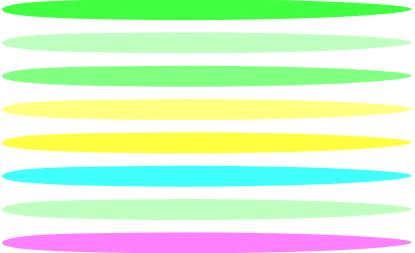


# Evolutionary Architecture

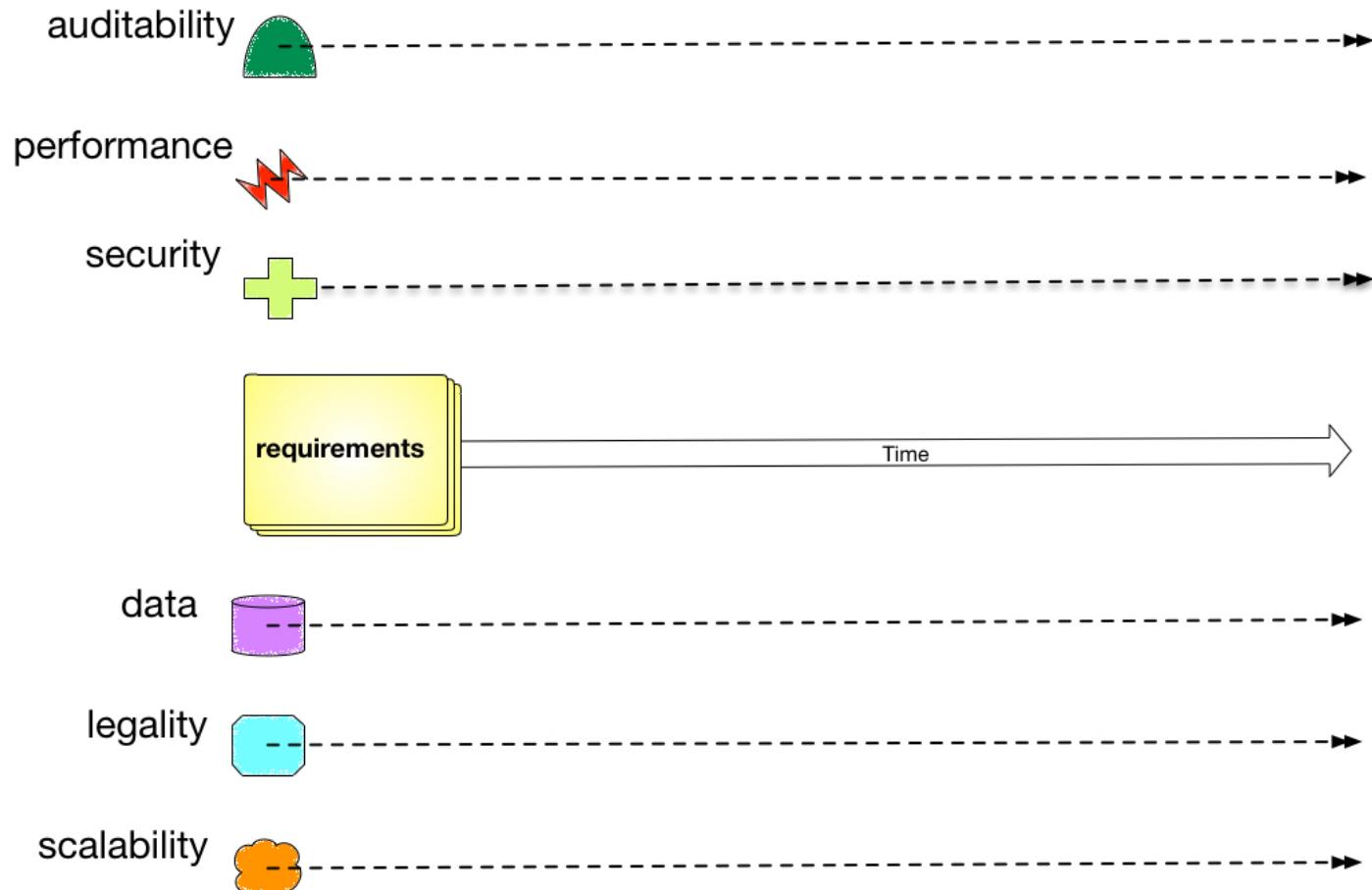
An evolutionary architecture supports  
guided,  
incremental change

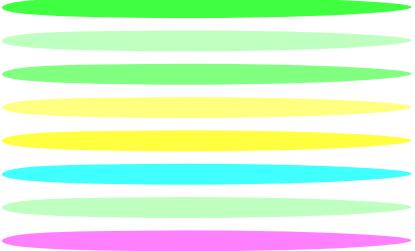
across **multiple dimensions**



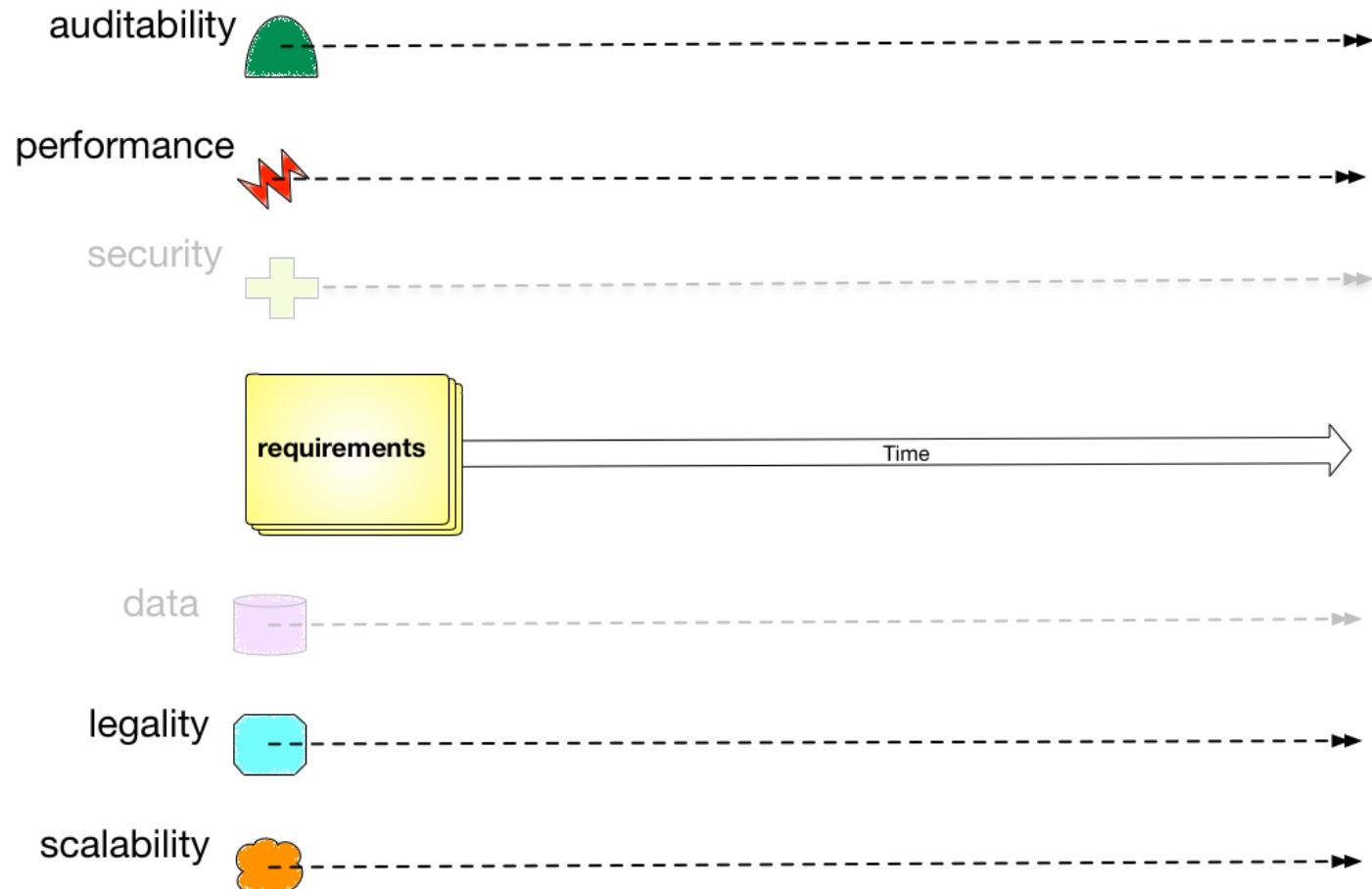


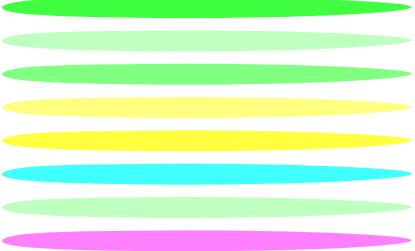
# multiple dimensions



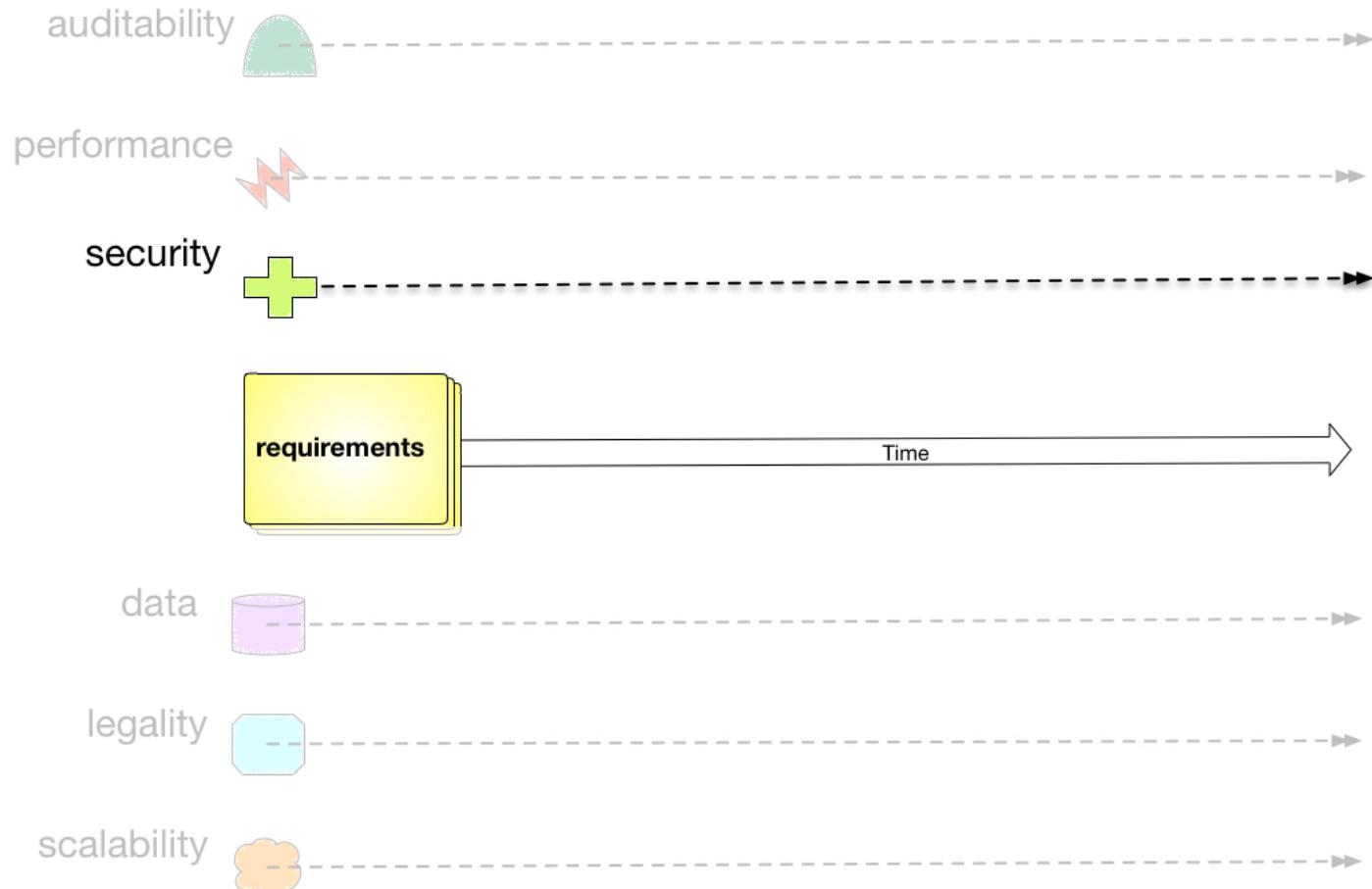


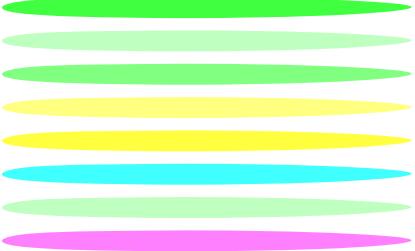
# multiple dimensions



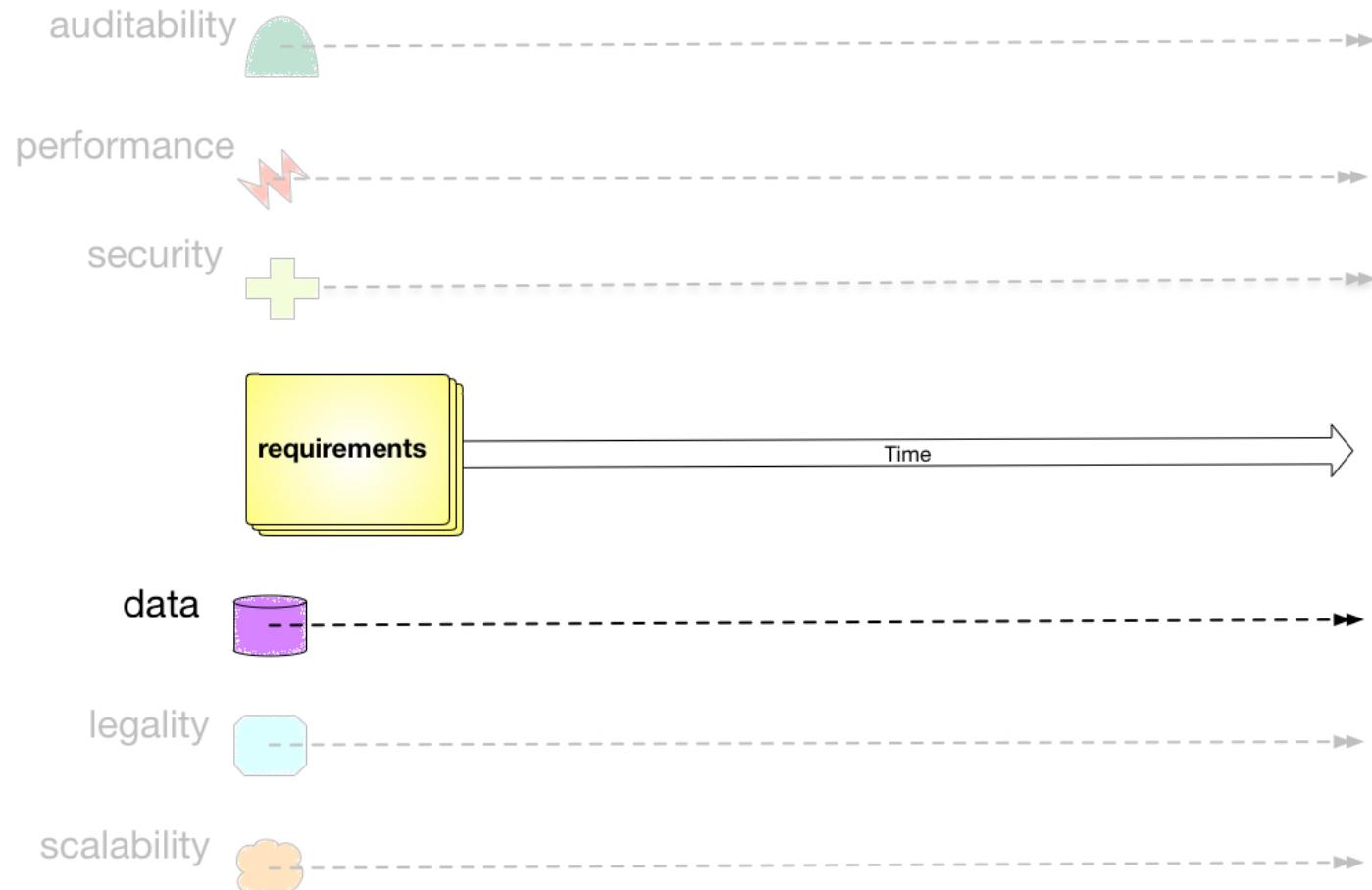


# multiple dimensions





# multiple dimensions



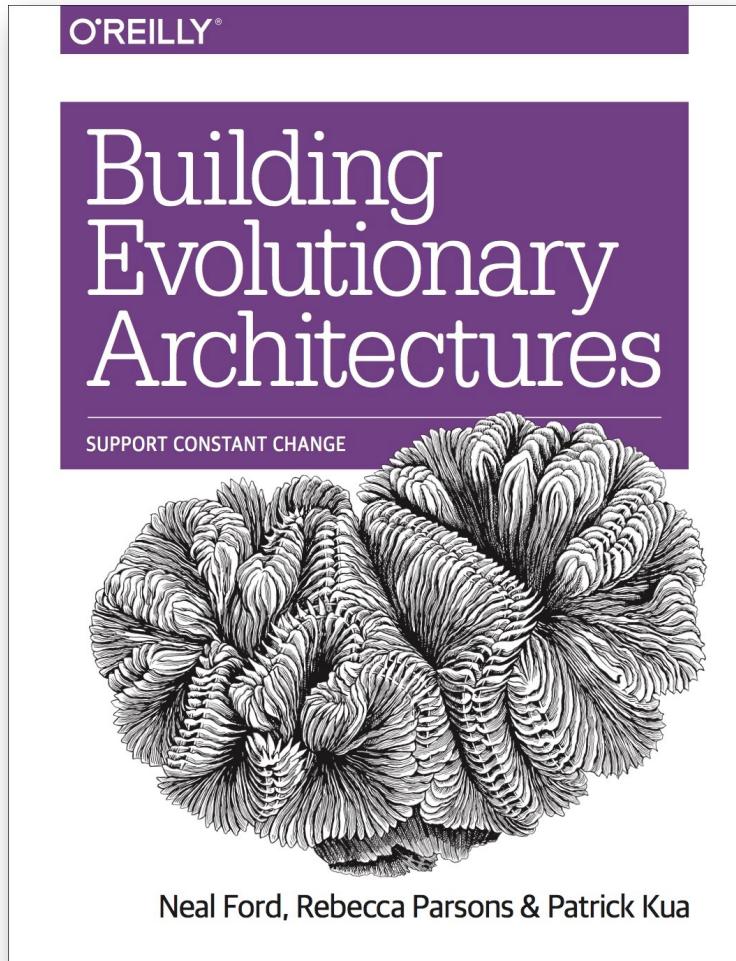
# Evolutionary Architecture

An evolutionary architecture supports  
guided,  
incremental change  
across multiple dimensions.



# Building Evolutionary Architectures

## FITNESS FUNCTIONS



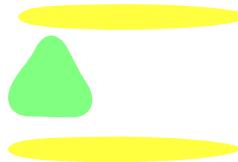
@neal4d  
[nealford.com](http://nealford.com)



# Evolutionary Architecture

An evolutionary architecture supports  
guided  
incremental change  
across multiple dimensions.



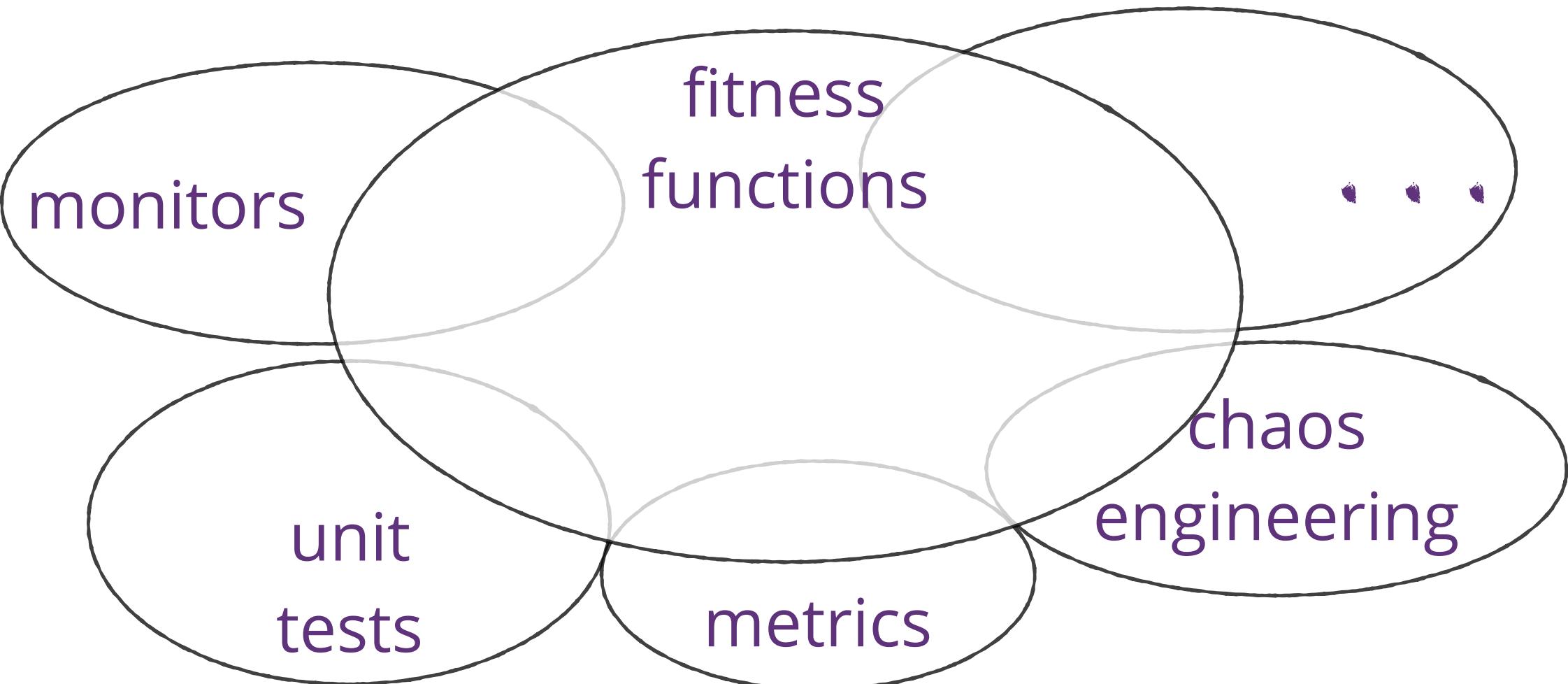


guided

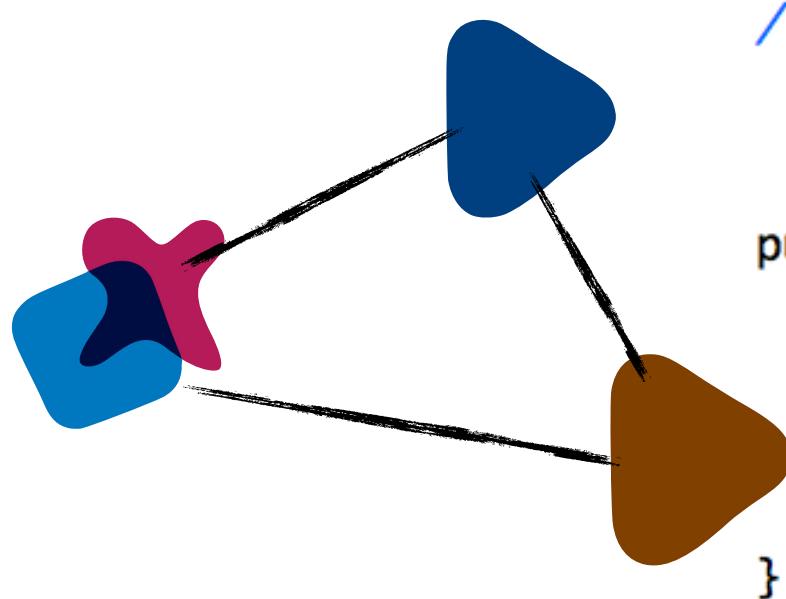
## **architectural fitness function:**

An architectural fitness function provides an objective integrity assessment of some architectural characteristic(s).

# Fitness Functions

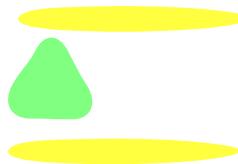


# Cyclic Dependency Function



```
/**  
 * Tests that a package dependency cycle does not  
 * exist for any of the analyzed packages.  
 */  
public void testAllPackages() {  
  
    Collection packages = jdepend.analyze();  
  
    assertEquals("Cycles exist",  
                false, jdepend.containsCycles());  
}
```

[clarkware.com/software/JDepend.html](http://clarkware.com/software/JDepend.html)

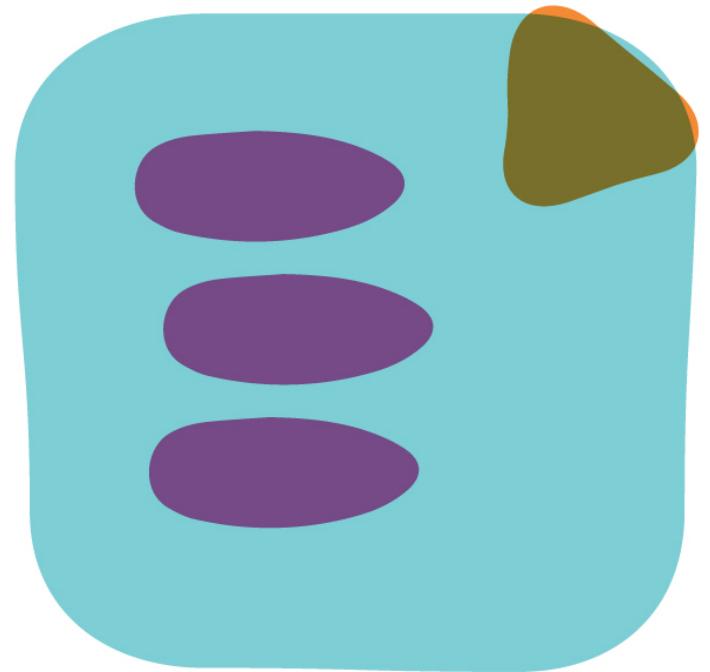


guided

## **architectural fitness function:**

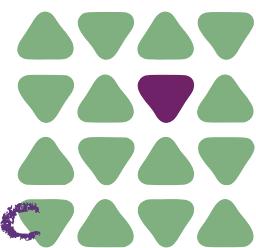
An architectural fitness function provides an objective integrity assessment of some architectural characteristic(s).

# Categories of Fitness Functions



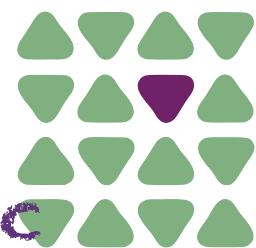
# Scope

run against a singular context and exercise one particular aspect of the architecture.



atomic

run against a shared context and exercise a combination of architectural aspects such as security and scalability



holistic

# Invocation

run based on a particular event:

- developer executing a unit test
- deployment pipeline running tests
- timed task



executes constant verification of architectural aspect(s)

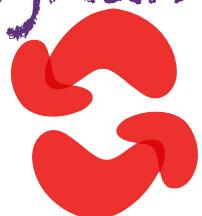
# Result

static



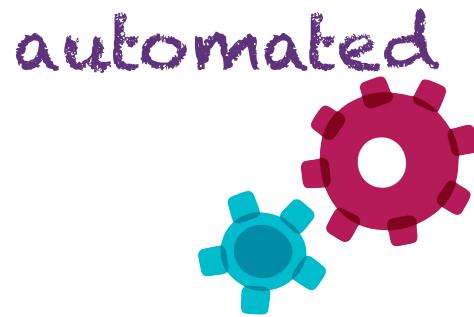
have a fixed result, such as  
a unit test pass/fail binary outcome.

dynamic



rely on a shifting definition based on extra  
context.

# Automation



tests and other verification mechanism that run without human interaction.



must involve at least one human.

# Time

temporal



architects may want to build a time component into assessing fitness

break on upgrade

overdue library update

# Domain?

domain-specific



Some architectures have specific concerns,  
such as special security or regulatory  
requirements

# Domain?

architectural  
characteristic

domain-specific



# Domain?

architectural  
characteristic

domain-specific



problem domain

unit



functional



# Creation



architects will define most fitness functions at project inception as they elucidate the characteristics of the architecture...

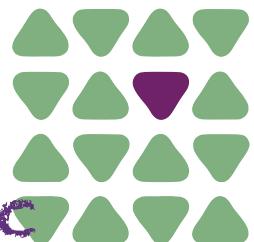
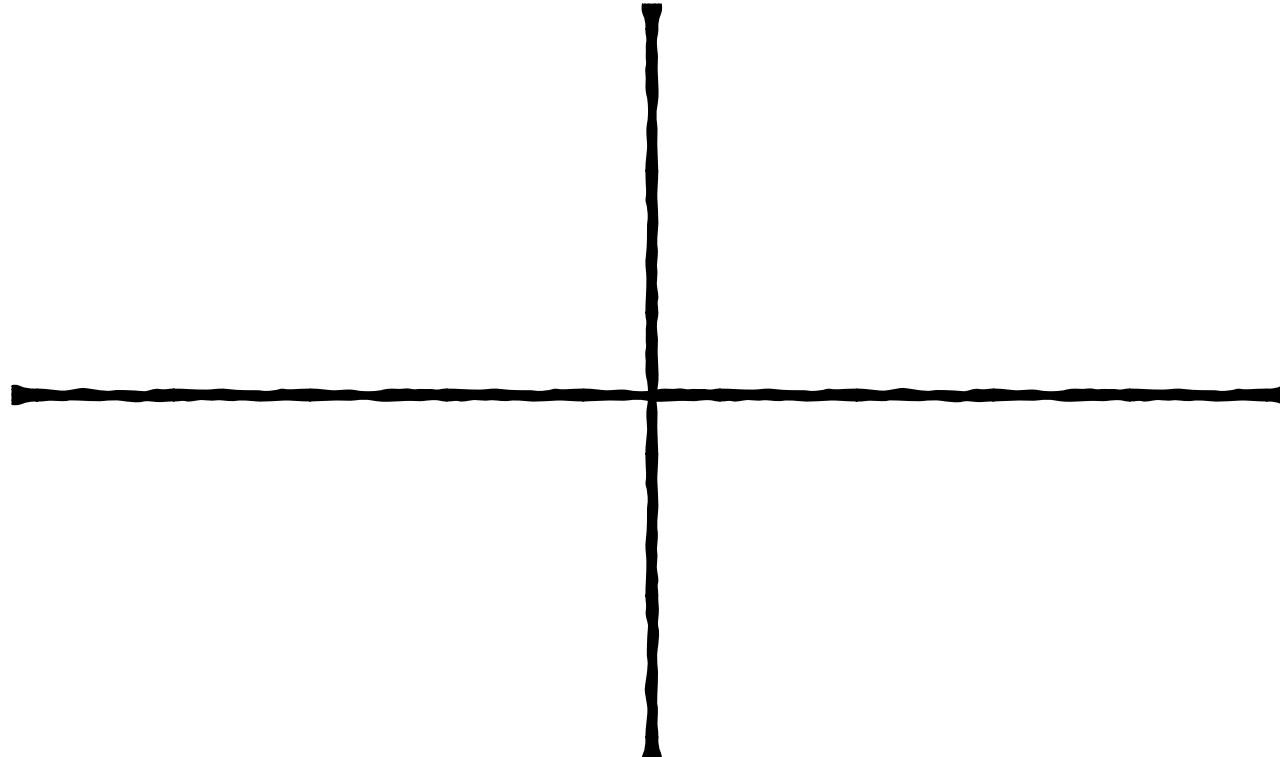


...some fitness functions will emerge during development of the system

# Fitness Function



atomic



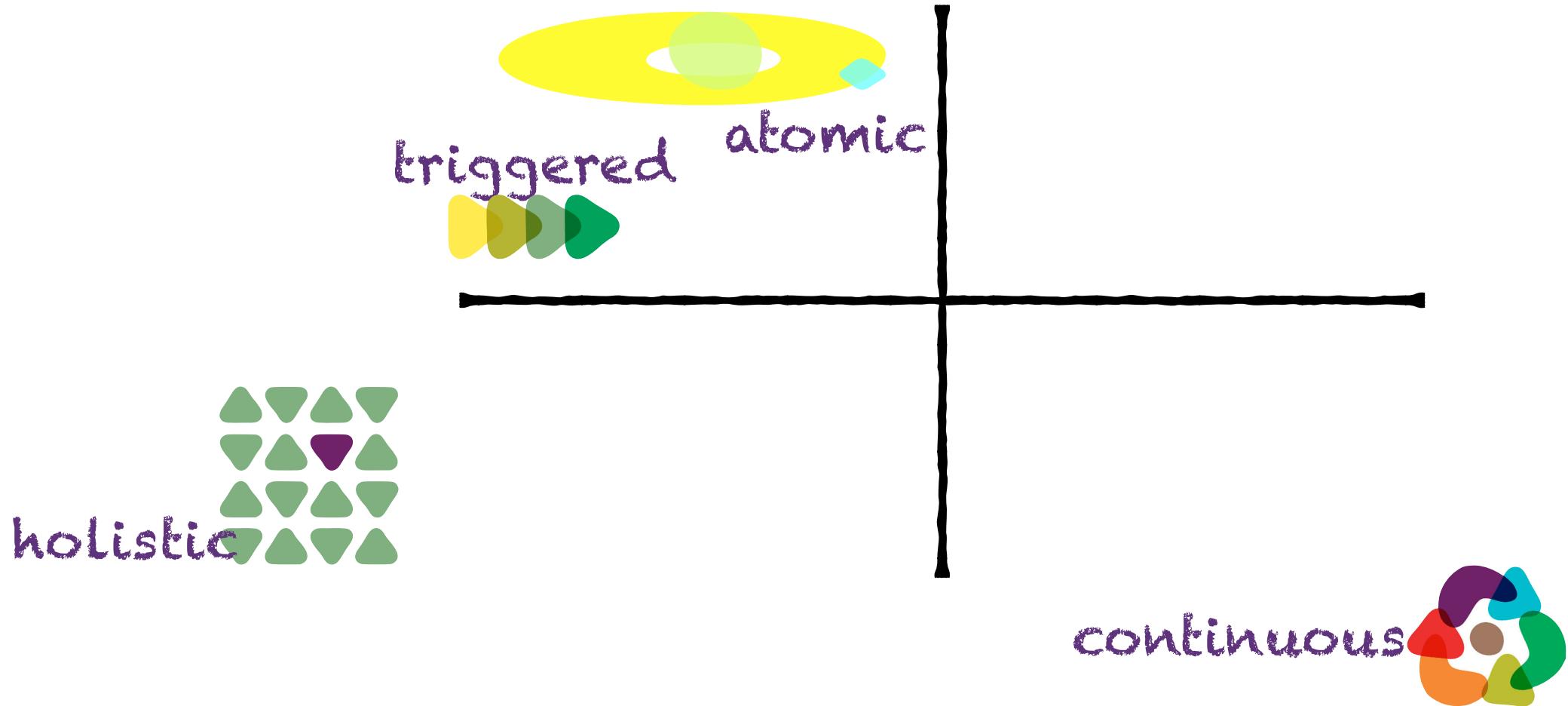
holistic



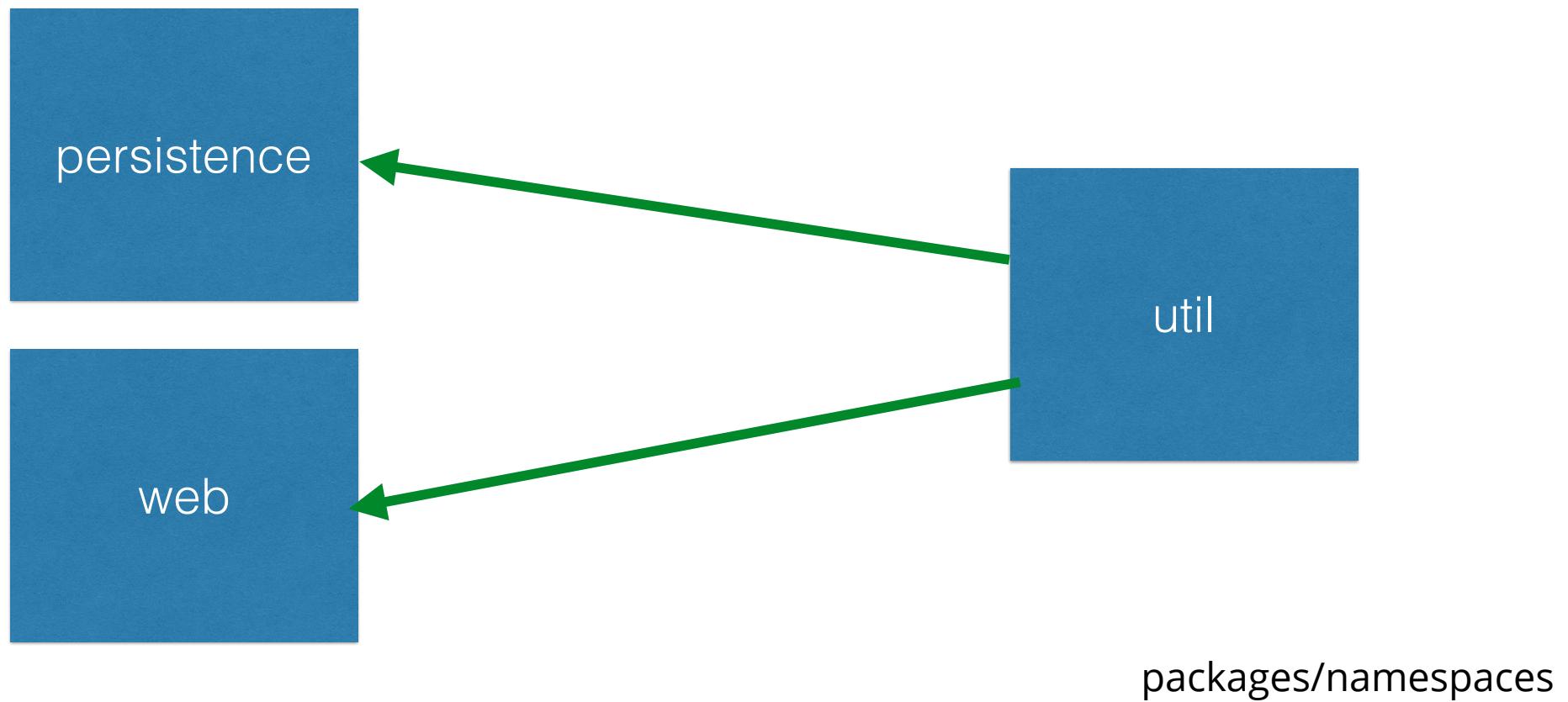
continuous



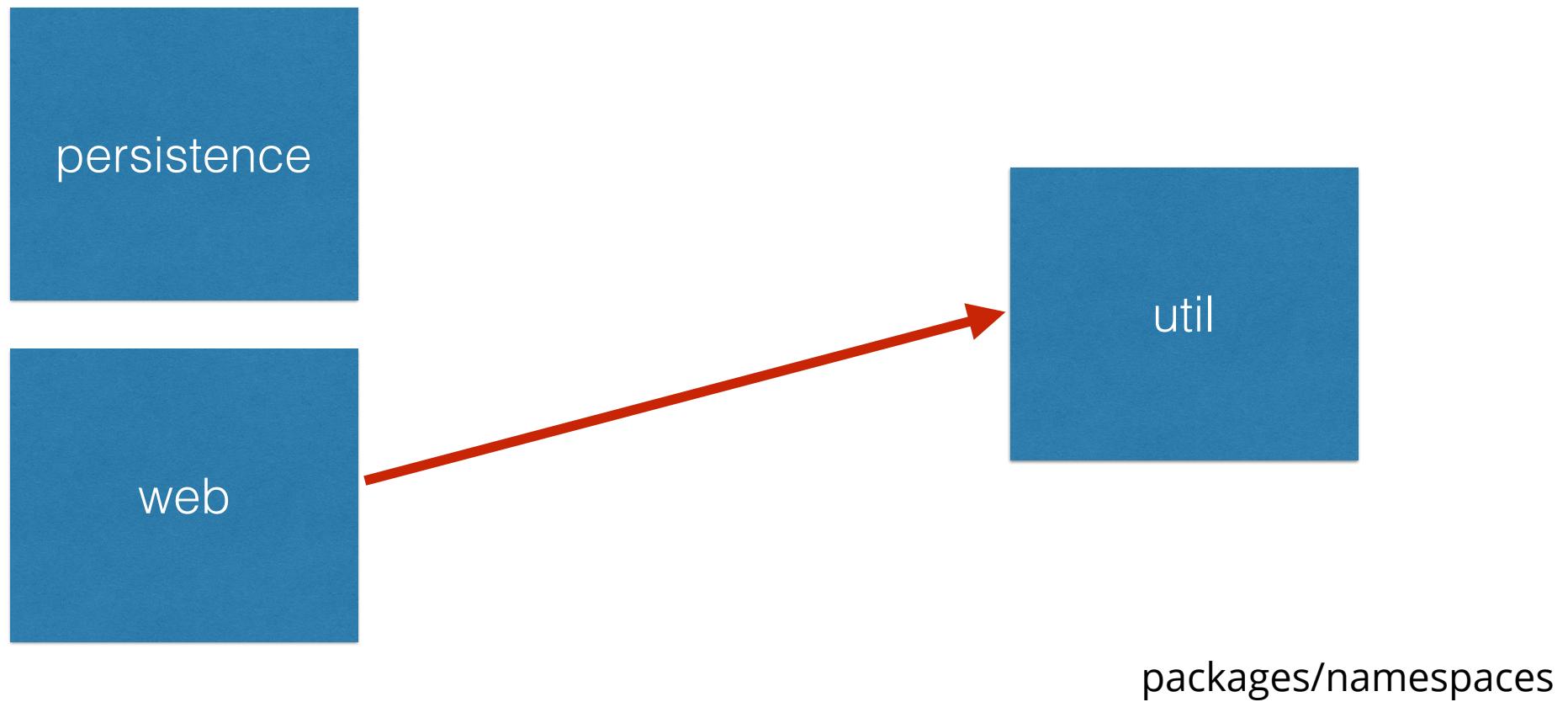
# Fitness Function



# Directionality of Imports



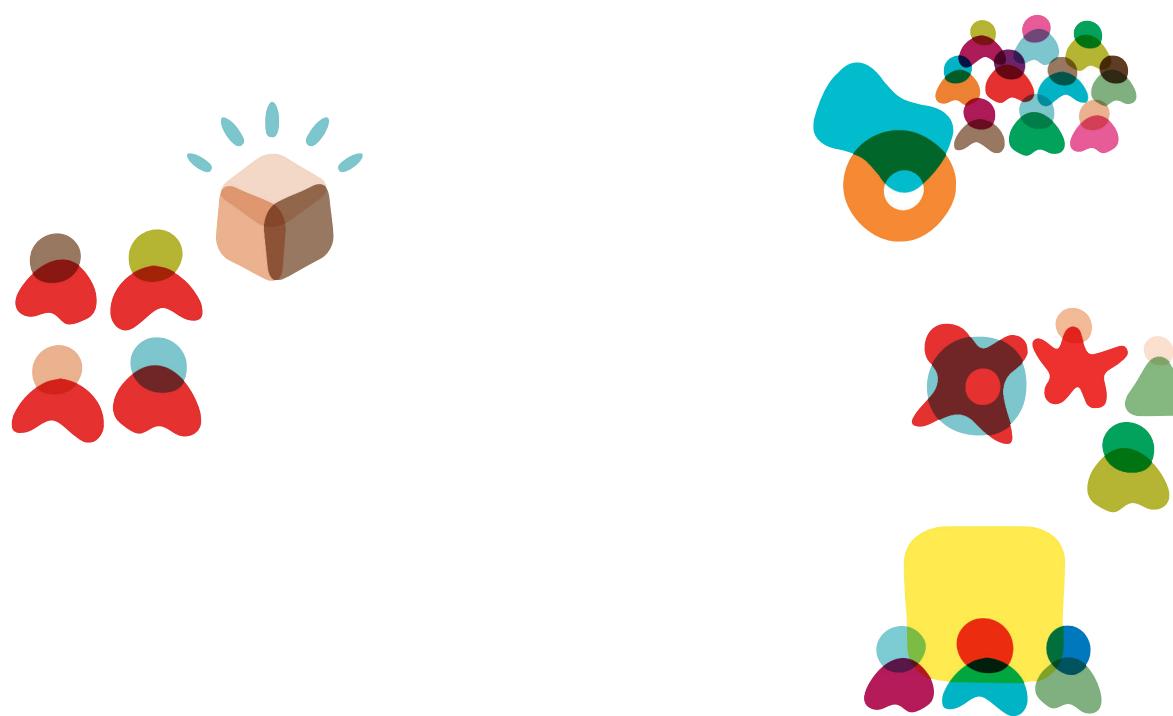
# Directionality of Imports



# Coupling Fitness Function

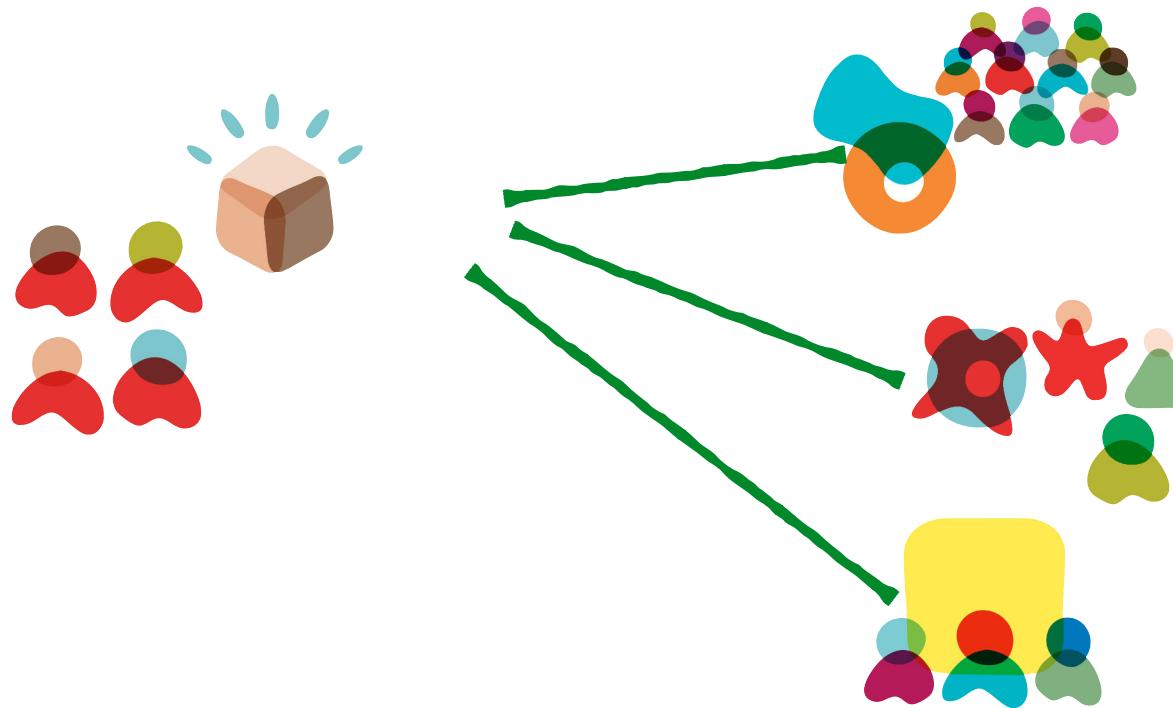
```
public void testMatch() {  
    DependencyConstraint constraint = new DependencyConstraint();  
  
    JavaPackage persistence = constraint.addPackage("com.xyz.persistence");  
    JavaPackage web = constraint.addPackage("com.xyz.web");  
    JavaPackage util = constraint.addPackage("com.xyz.util");  
  
    persistence.dependsUpon(util);  
    web.dependsUpon(util);  
  
    jdepend.analyze();  
  
    assertEquals("Dependency mismatch",  
                true, jdepend.dependencyMatch(constraint));  
}
```

# Consumer Driven Contracts



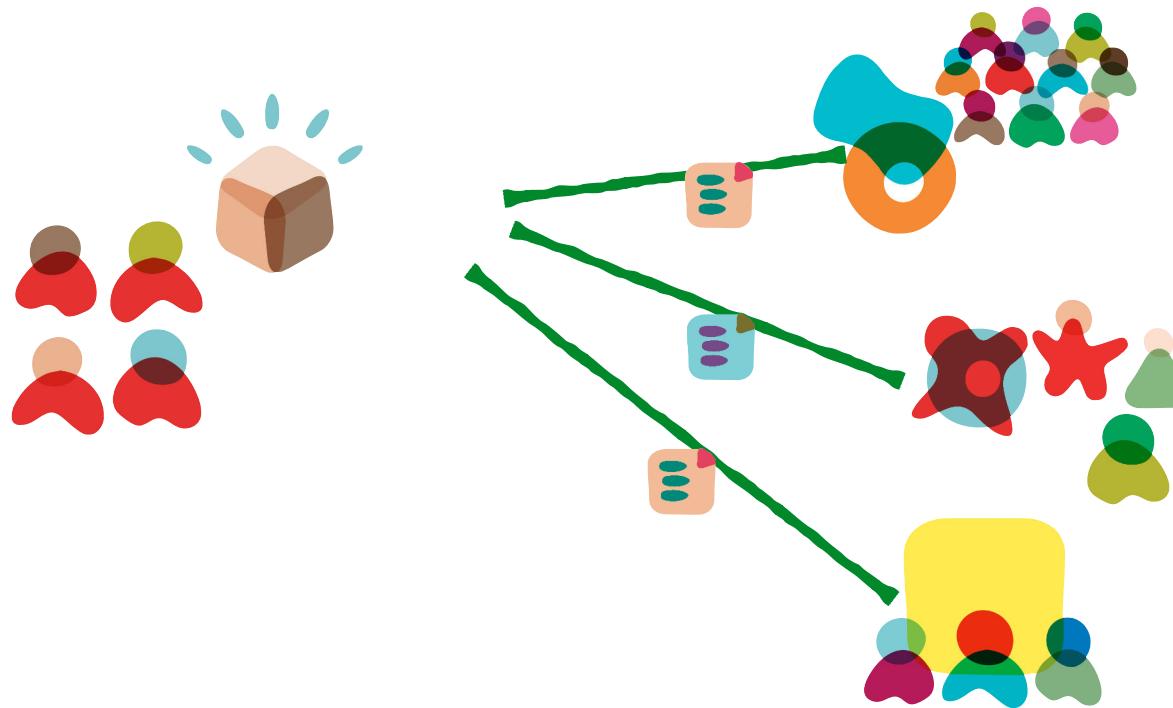
[martinfowler.com/articles/consumerDrivenContracts.html](http://martinfowler.com/articles/consumerDrivenContracts.html)

# Consumer Driven Contracts



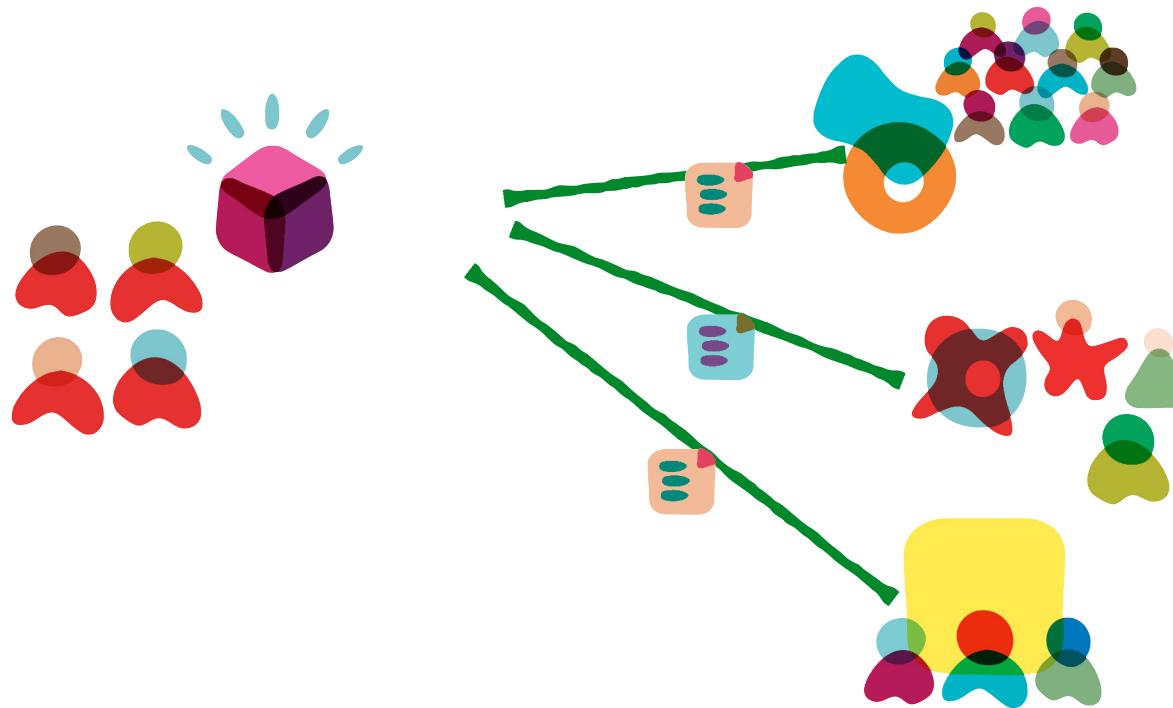
[martinfowler.com/articles/consumerDrivenContracts.html](http://martinfowler.com/articles/consumerDrivenContracts.html)

# Consumer Driven Contracts



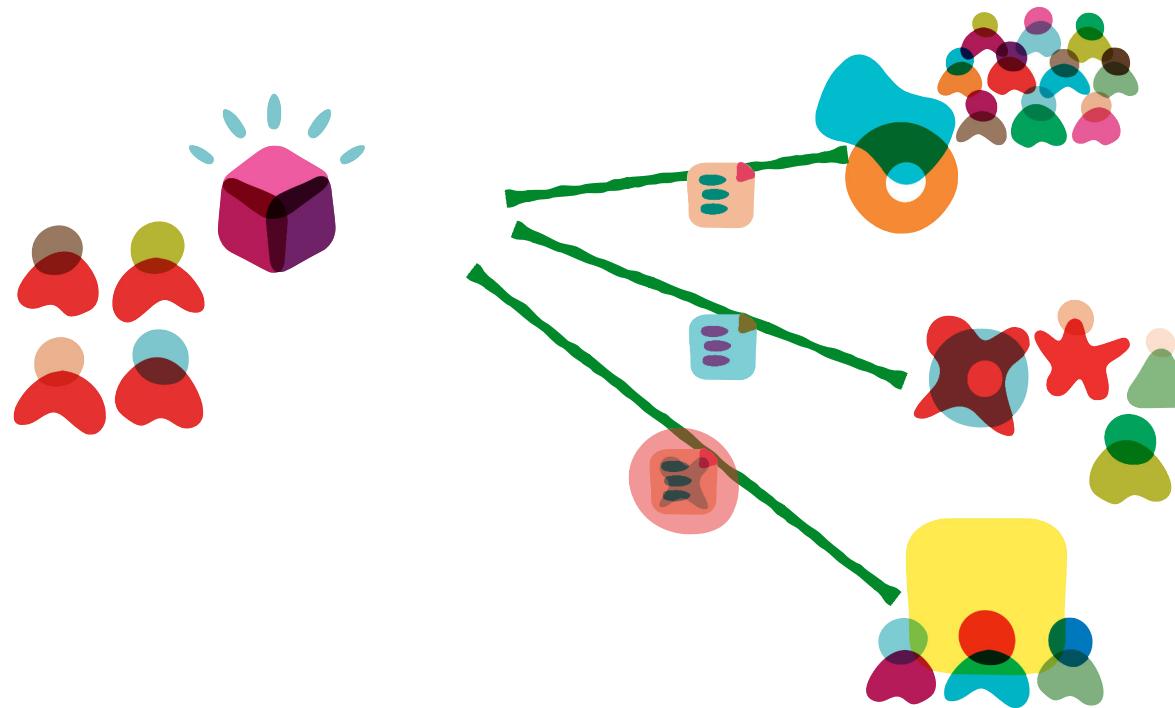
[martinfowler.com/articles/consumerDrivenContracts.html](http://martinfowler.com/articles/consumerDrivenContracts.html)

# Consumer Driven Contracts



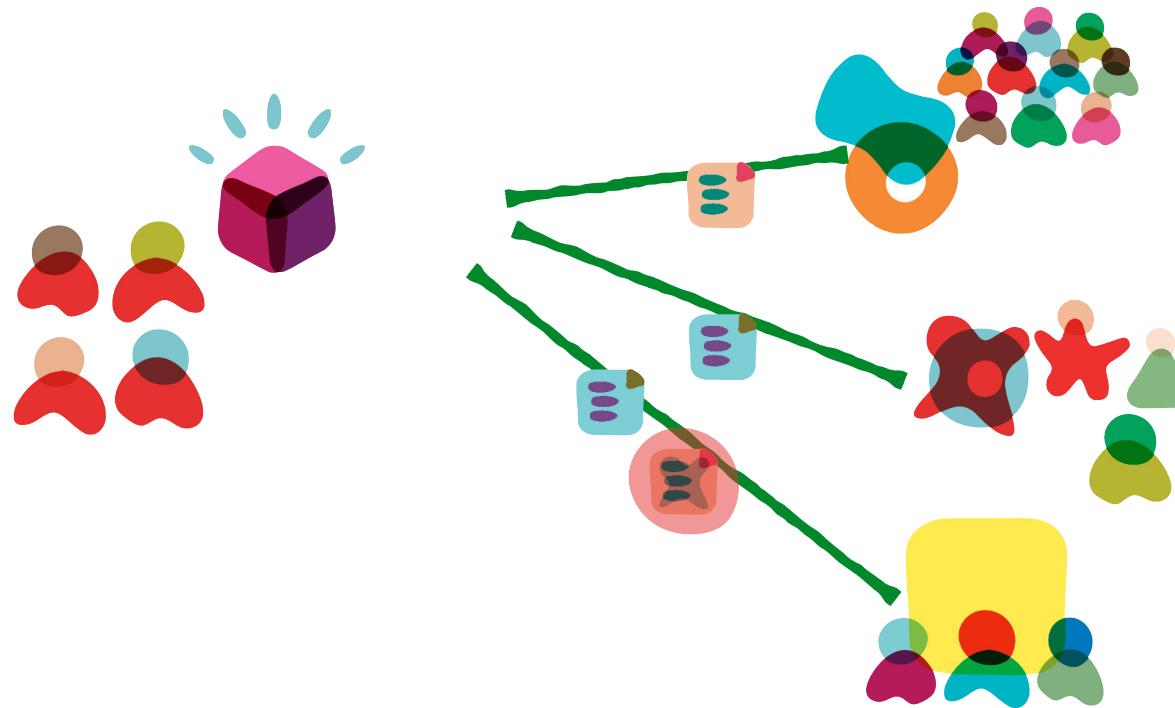
[martinfowler.com/articles/consumerDrivenContracts.html](http://martinfowler.com/articles/consumerDrivenContracts.html)

# Consumer Driven Contracts



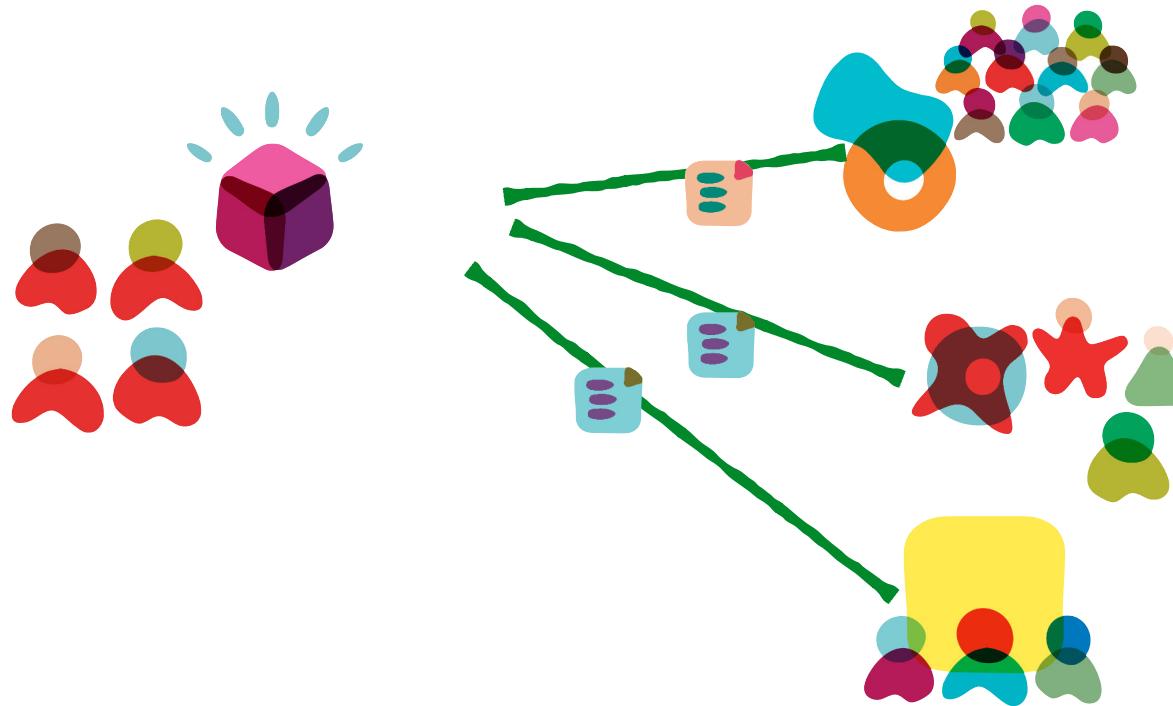
[martinfowler.com/articles/consumerDrivenContracts.html](http://martinfowler.com/articles/consumerDrivenContracts.html)

# Consumer Driven Contracts



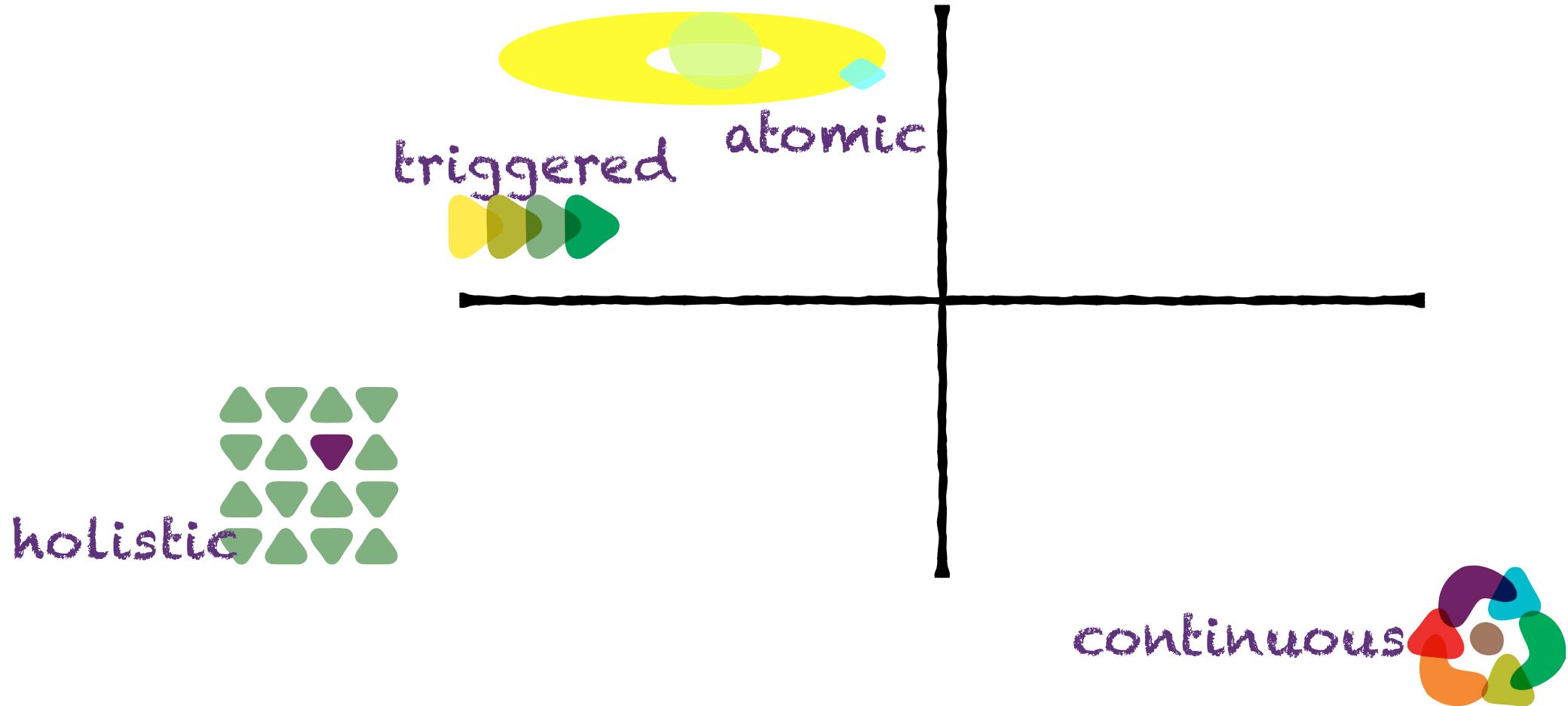
[martinfowler.com/articles/consumerDrivenContracts.html](http://martinfowler.com/articles/consumerDrivenContracts.html)

# Consumer Driven Contracts



[martinfowler.com/articles/consumerDrivenContracts.html](http://martinfowler.com/articles/consumerDrivenContracts.html)

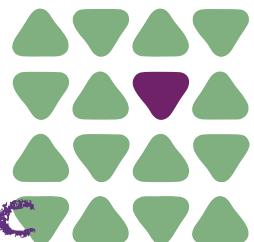
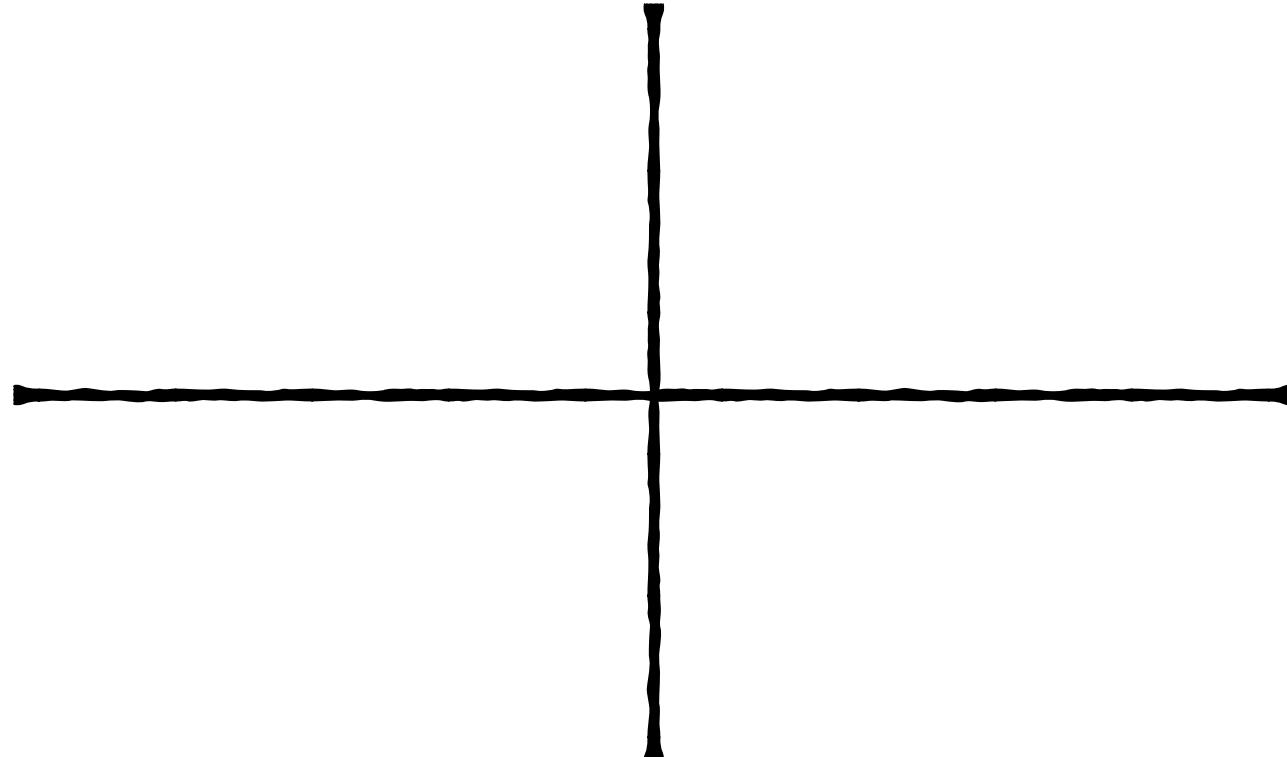
# Fitness Function



# Fitness Function



atomic



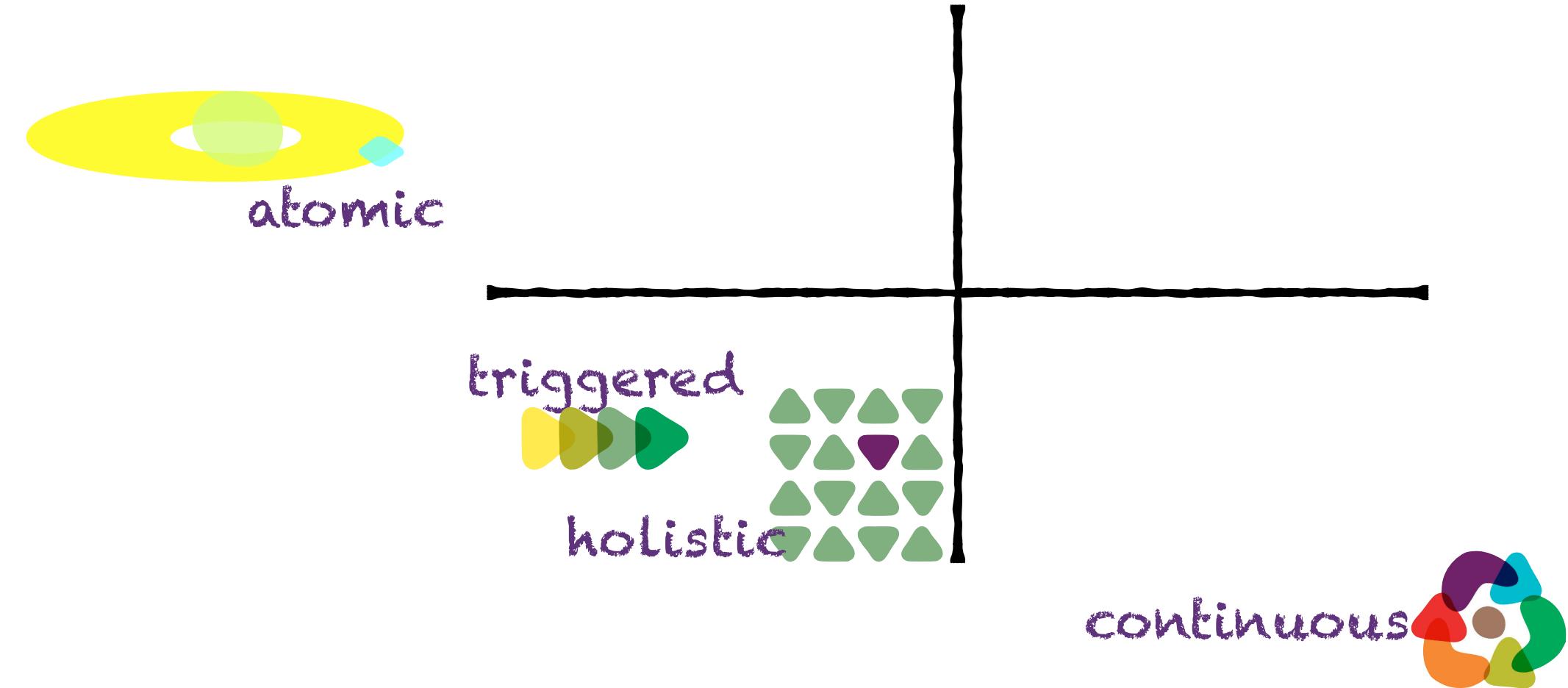
holistic



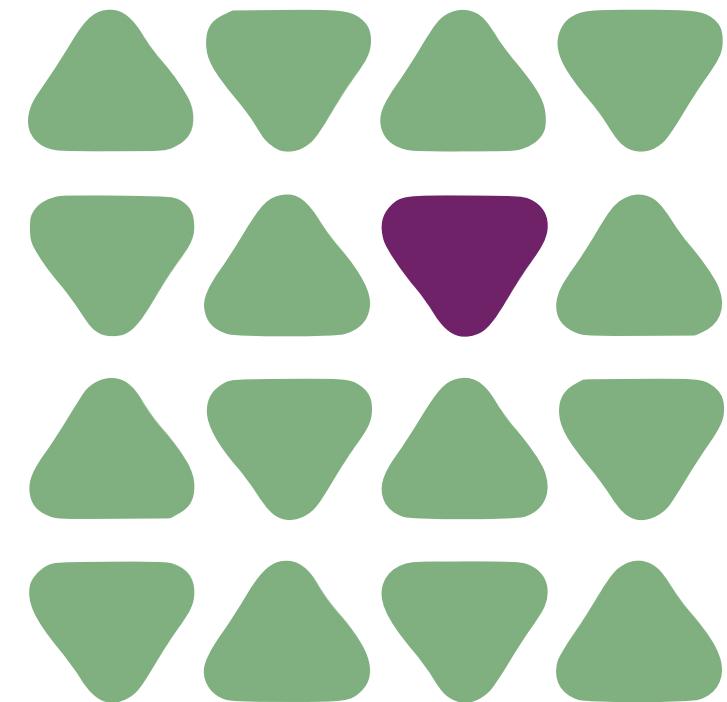
continuous



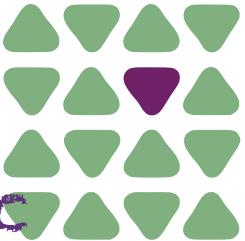
# Fitness Function

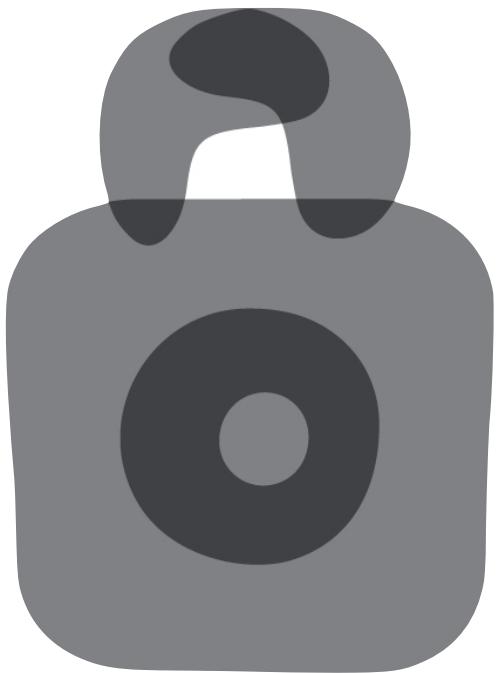


triggered



holistic

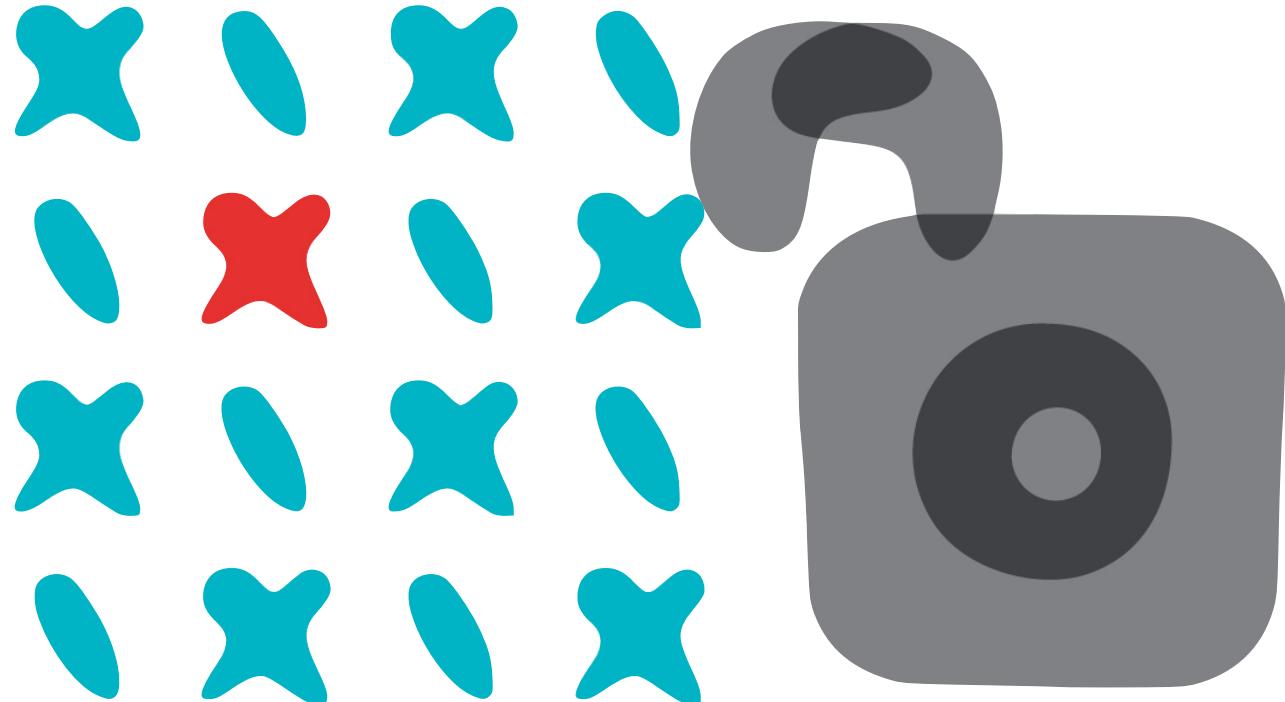
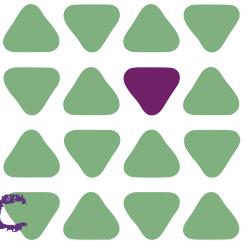




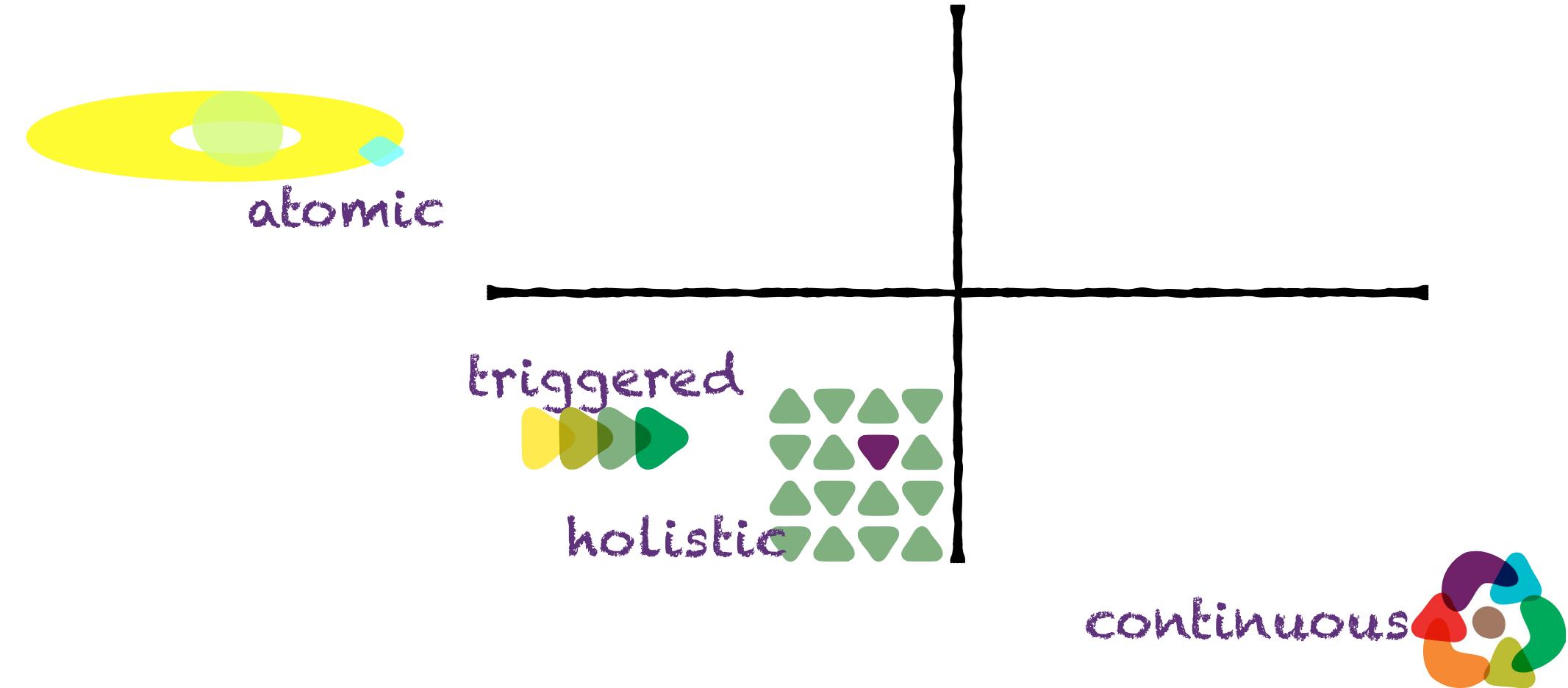
triggered



holistic



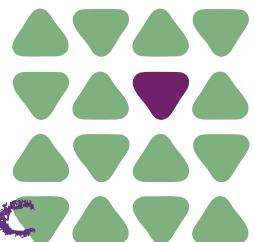
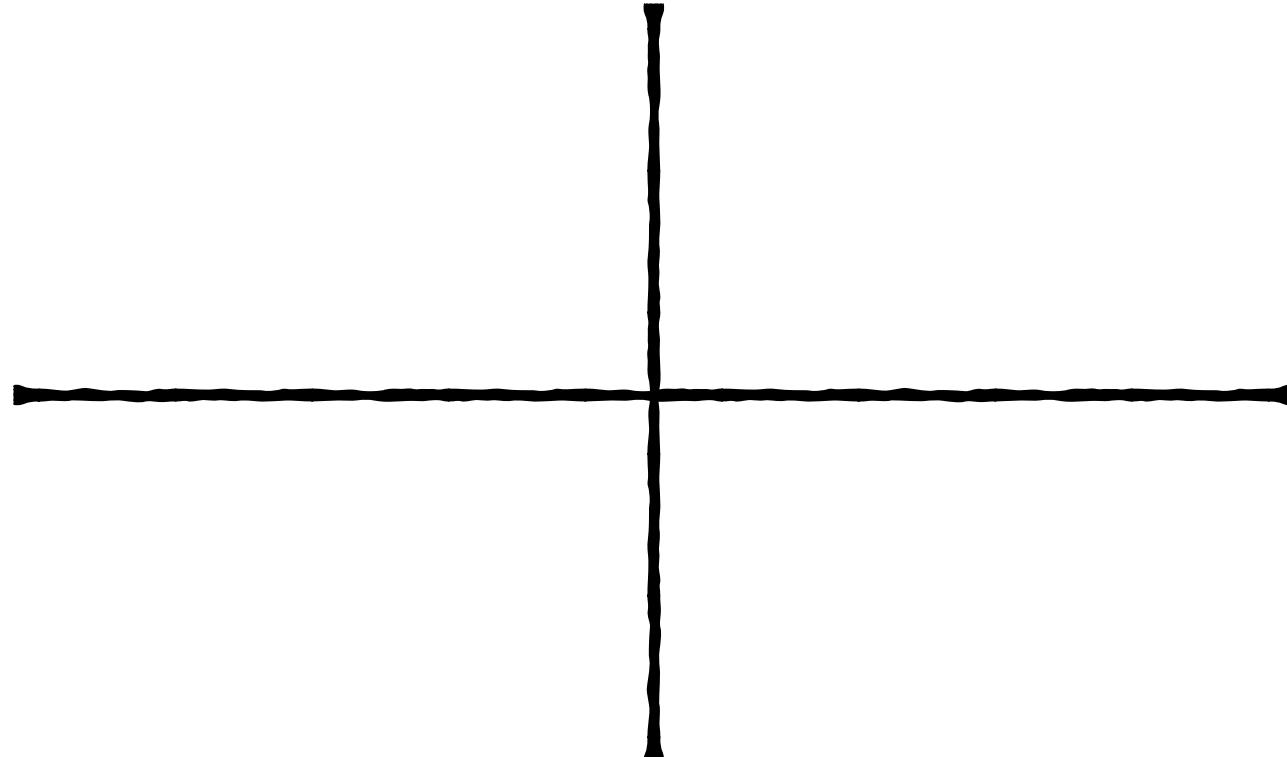
# Fitness Function



# Fitness Function



atomic



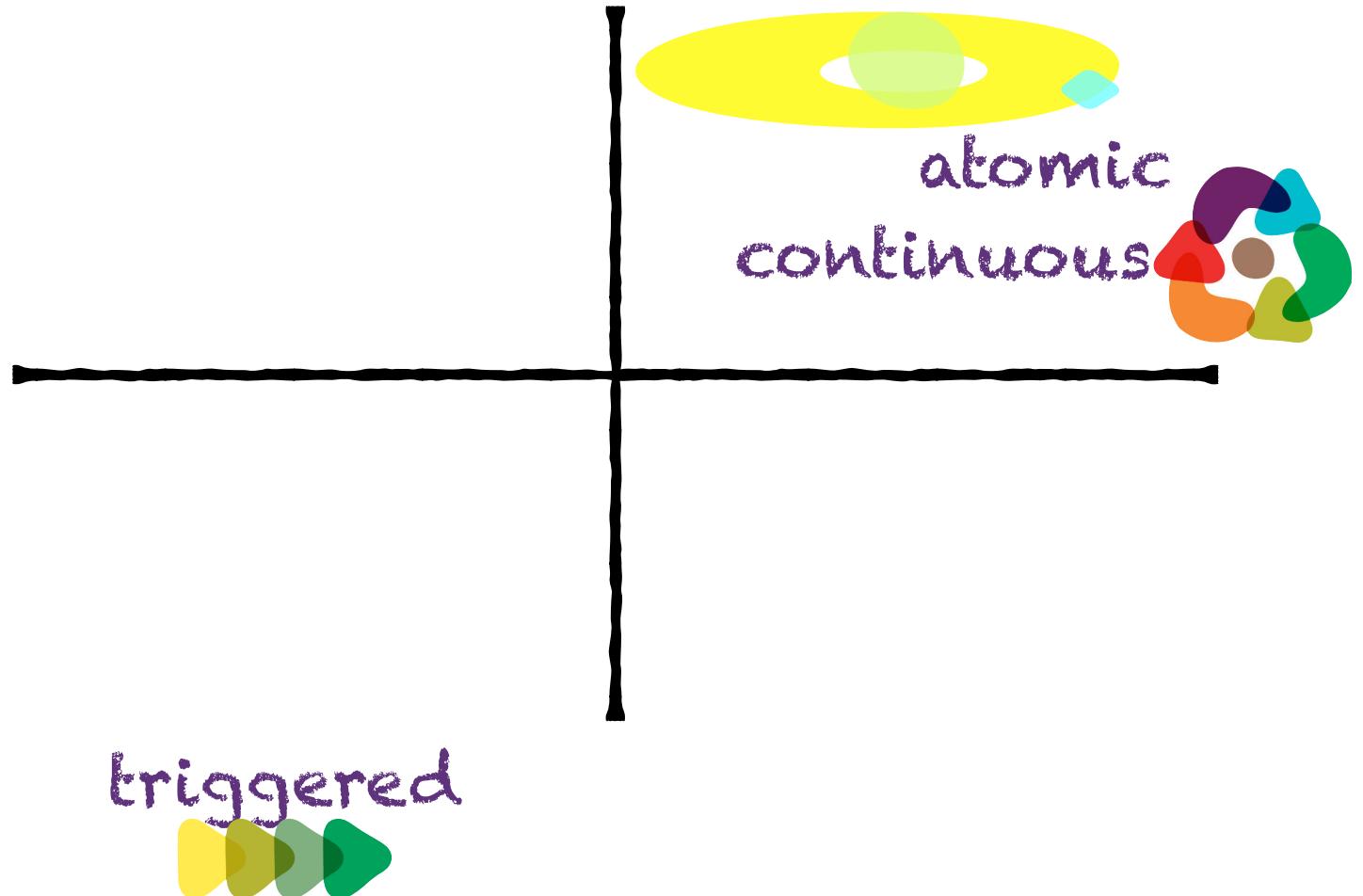
holistic

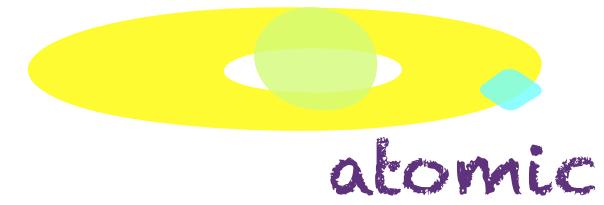


continuous

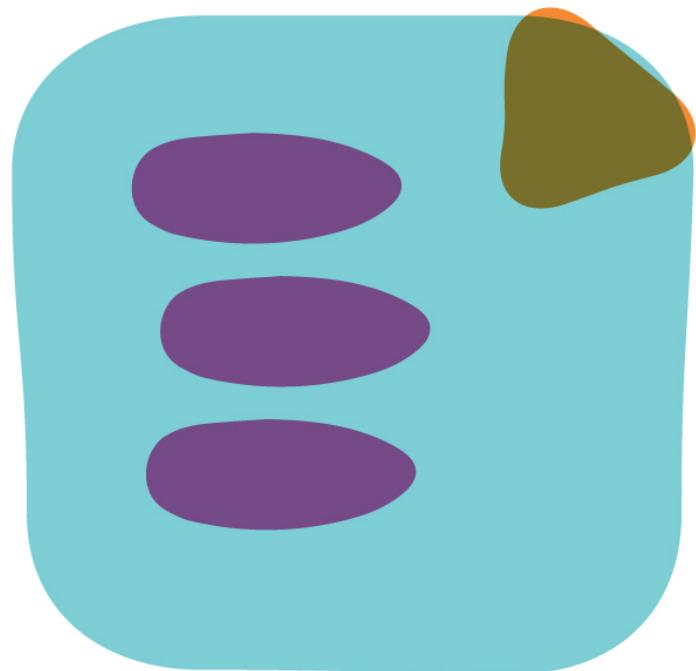
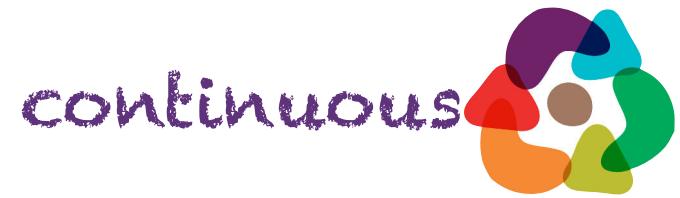


# Fitness Function

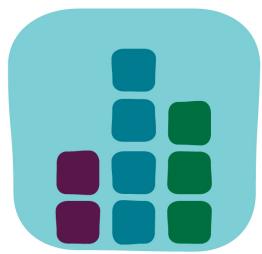




monitoring



logging



# monitoring

Nagios XI Home Views Dashboards Reports Configure Tools Help Admin

Quick View Host Status Summary Service Status Summary Hostgroup Status Summary

**Host Status Summary**

Up	Down	Unreachable	Pending
59	60	3	0
Unhandled		Problems	
64	64	117	

Last Updated: 2017-10-05 16:06:57

**Service Status Summary**

Ok	Warning	Unknown	Critical	Pending
226	12	64	221	2
Unhandled		Problems		
366	367	595		

Last Updated: 2017-10-05 16:06:57

**Hostgroup Status Summary**

**Status Summary For All Host Groups**

Host Group	Hosts	Services
All EMC SAN Hosts (all_emc_hosts)	1 Up	4 Ok, 1 Critical
Firewalls (firewalls)	1 Up	1 Ok
Host Deadpool (host-deadpool)	1 Up, 1 Down, 1 Unreachable	8 Ok, 7 Critical
Linux Servers (linux-servers)	5 Up	52 Ok, 3 Warning, 9 Unknown, 5 Critical
new group (new group)	2 up, 1 Down, 2 Unreachable	58 Ok, 3 Warning, 9 Unknown, 3 Critical
Printers (printers)	1 Up, 2 Down, 2 Unreachable	2 Ok, 2 Critical
Websites (websites)	3 Up	20 Ok, 2 Warning, 2 Critical
Windows Servers (windows-servers)	2 Down	8 Critical

Last Updated: 2017-10-05 16:06:57

**My Graph**

**Top Alert Producers Last 24 Hours**

**Metrics Overview**

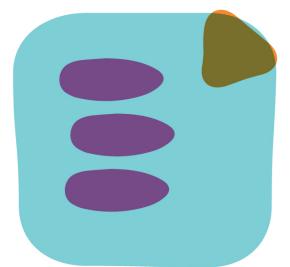
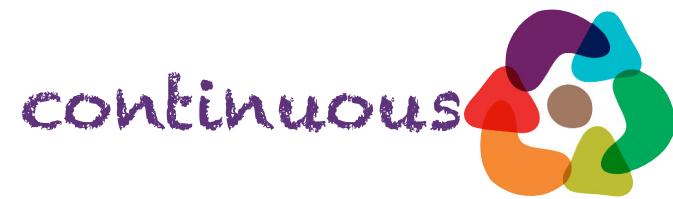
**Disk Usage**

Host	Service	% Utilization	Details
localhost	Root Partition	78.67%	DISK WARNING - free space: / 1207 MB (17% inode=68%):
vs1.nagios.com	/ Disk Usage	37.30%	DISK OK - free space: / 117214 MB (61% inode=99%):
exchange.nagios.org	/ Disk Usage	13.22%	DISK OK - free space: / 68067 MB (86% inode=97%):

Last Updated: 2017-10-05 16:06:58

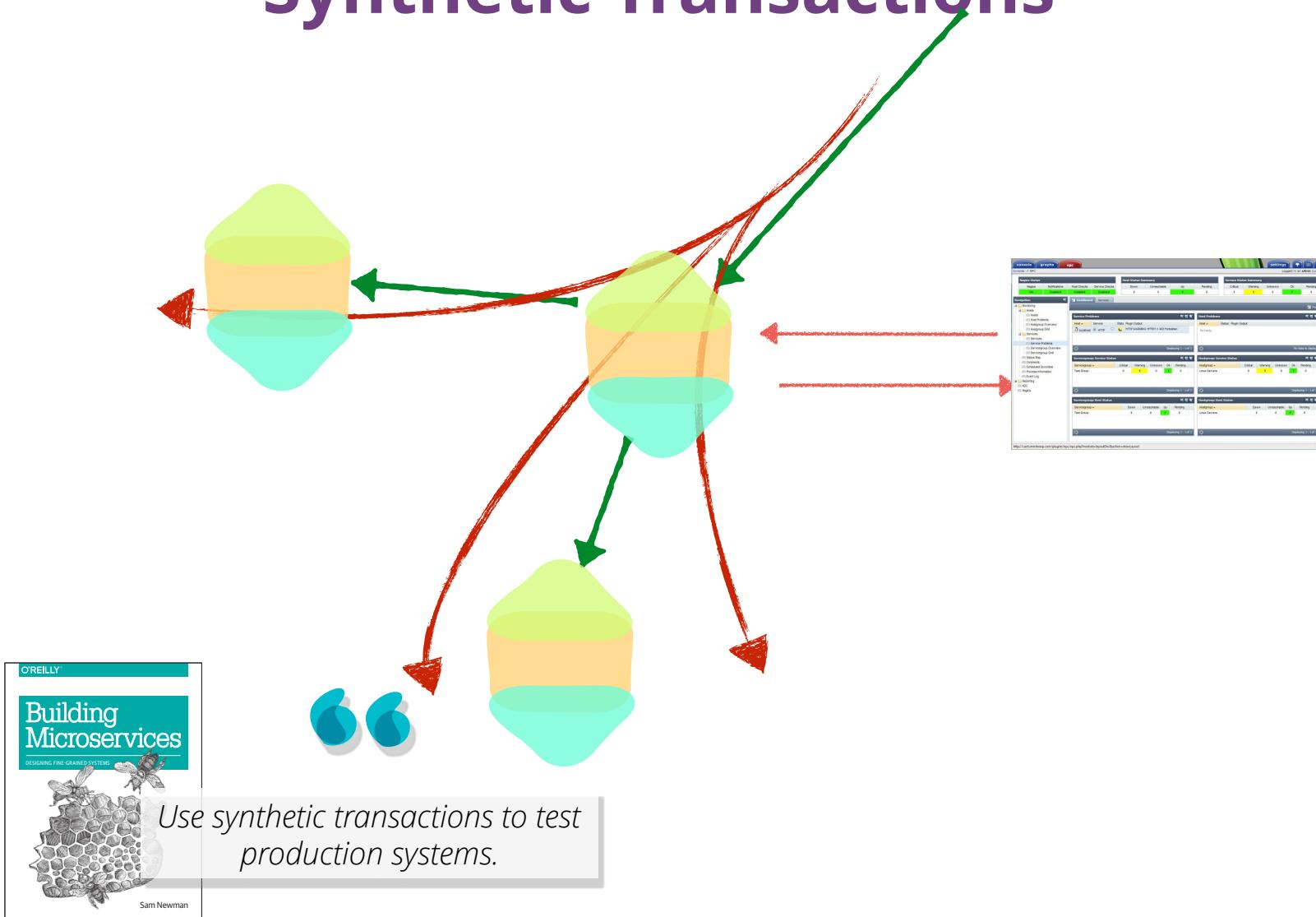
Nagios XI 5.4.10 • Check for Updates

About | Legal | Copyright © 2008-2017 Nagios Enterprises, LLC

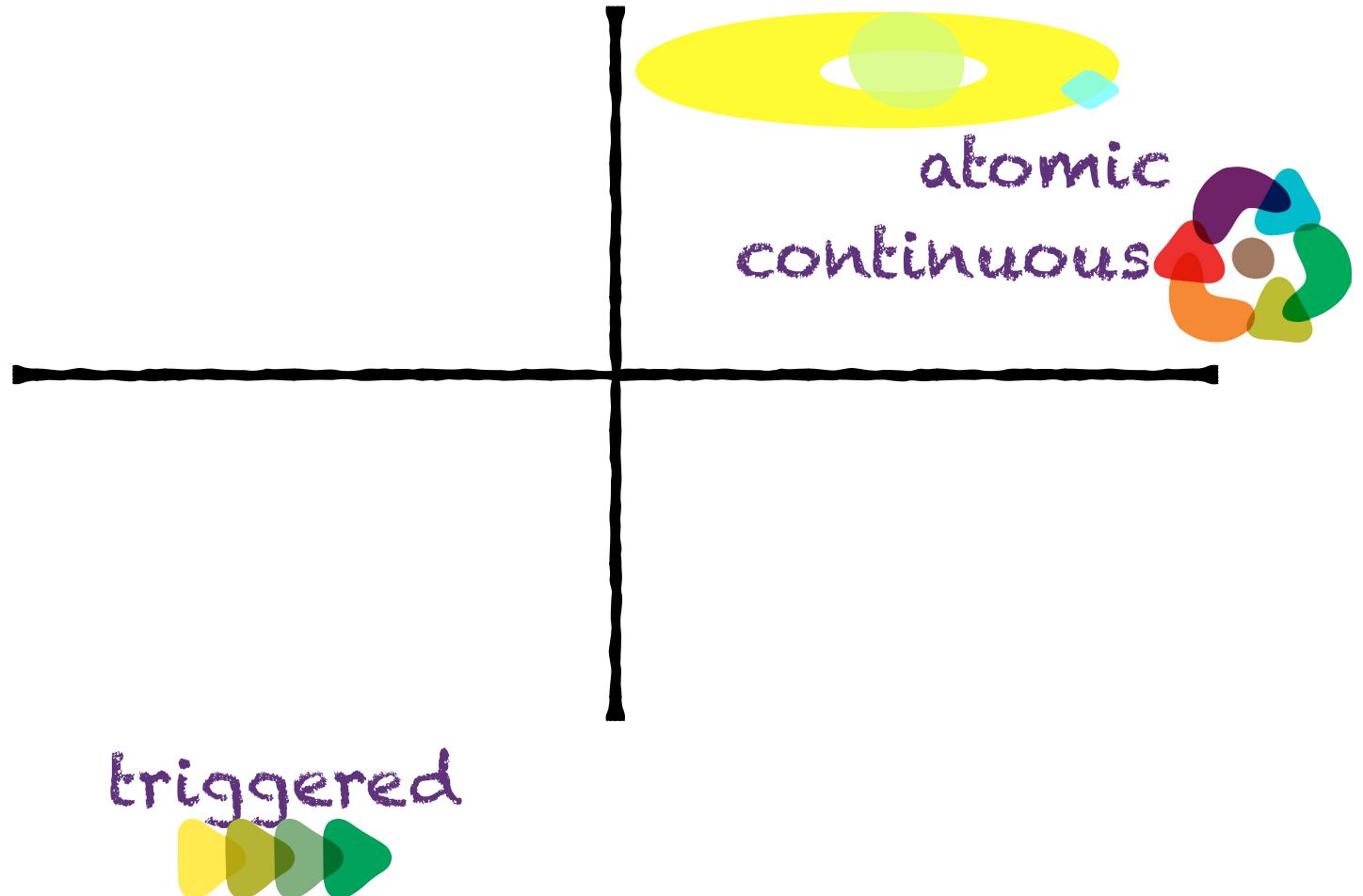


# logging

# Synthetic Transactions



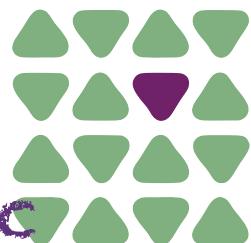
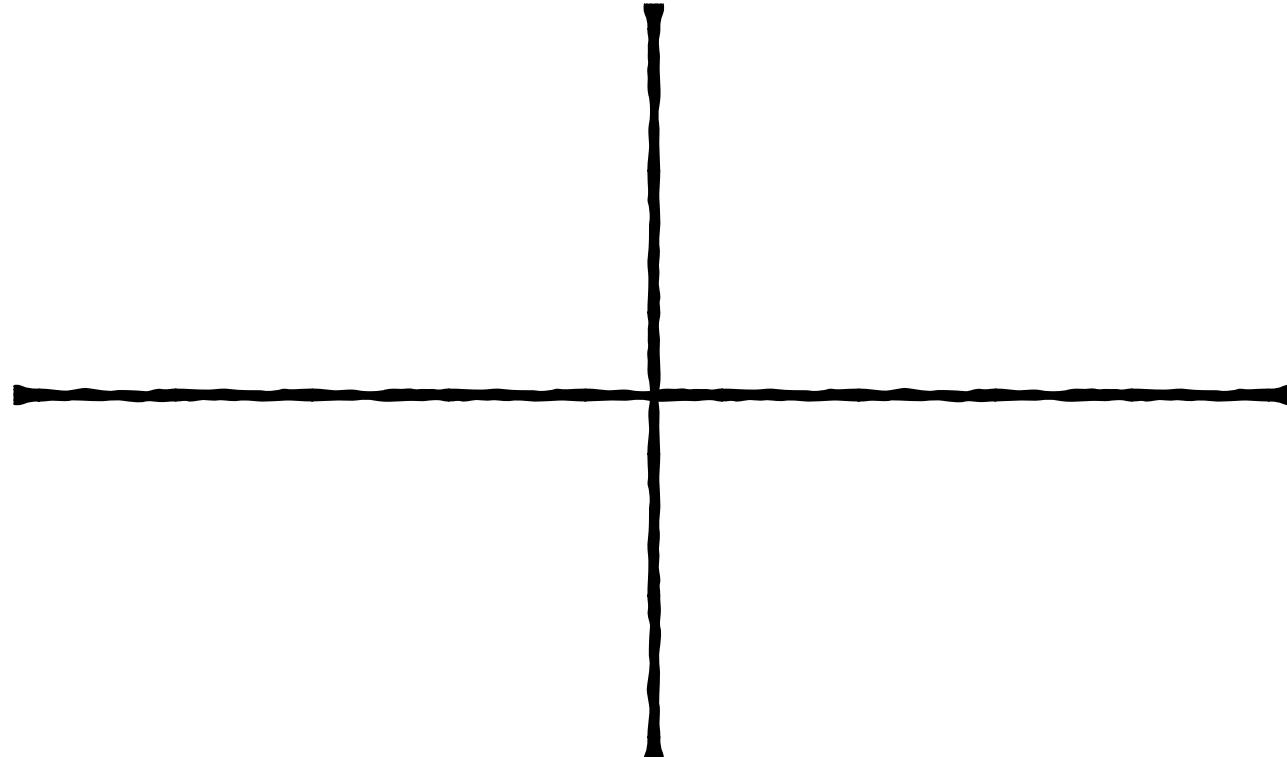
# Fitness Function



# Fitness Function



atomic



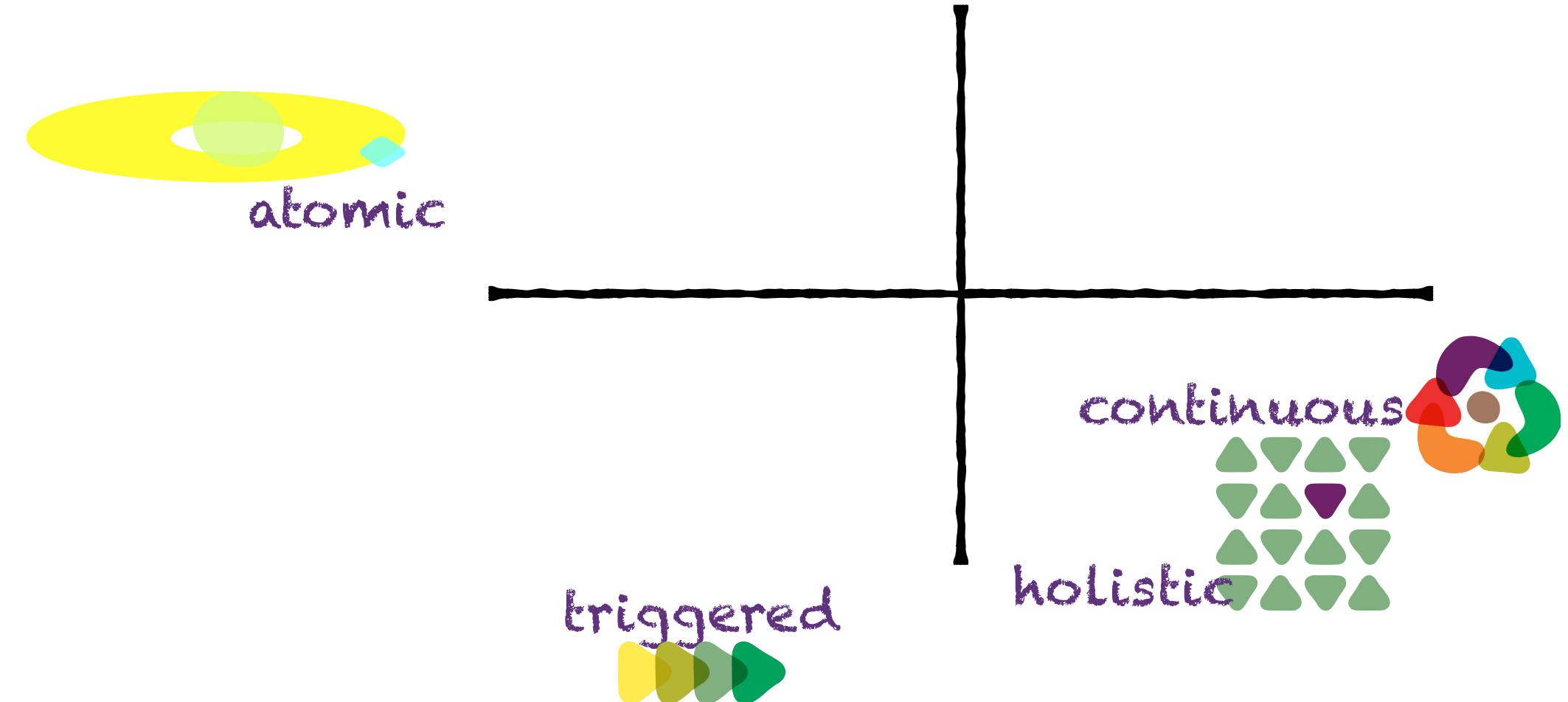
holistic



continuous



# Fitness Function









chaos monkey

# SIAMIAN ARMY



chaos gorilla

# SIAMIAN ARMY



Latency monkey



doctor monkey

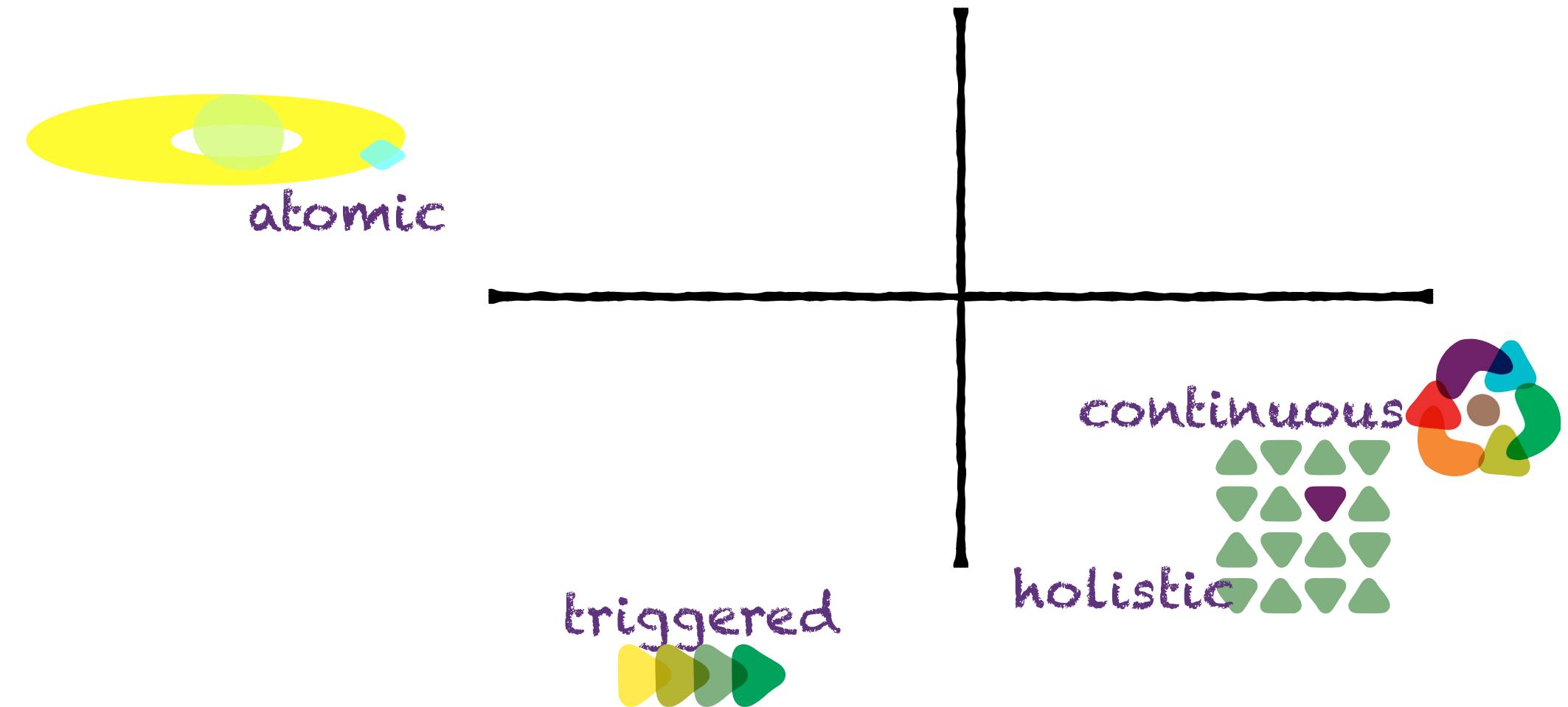


janitor monkey

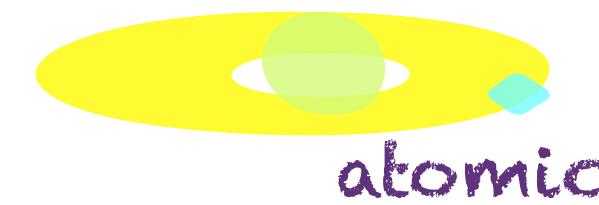
# SIMIAN ARMY



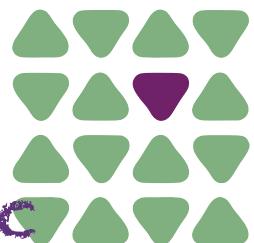
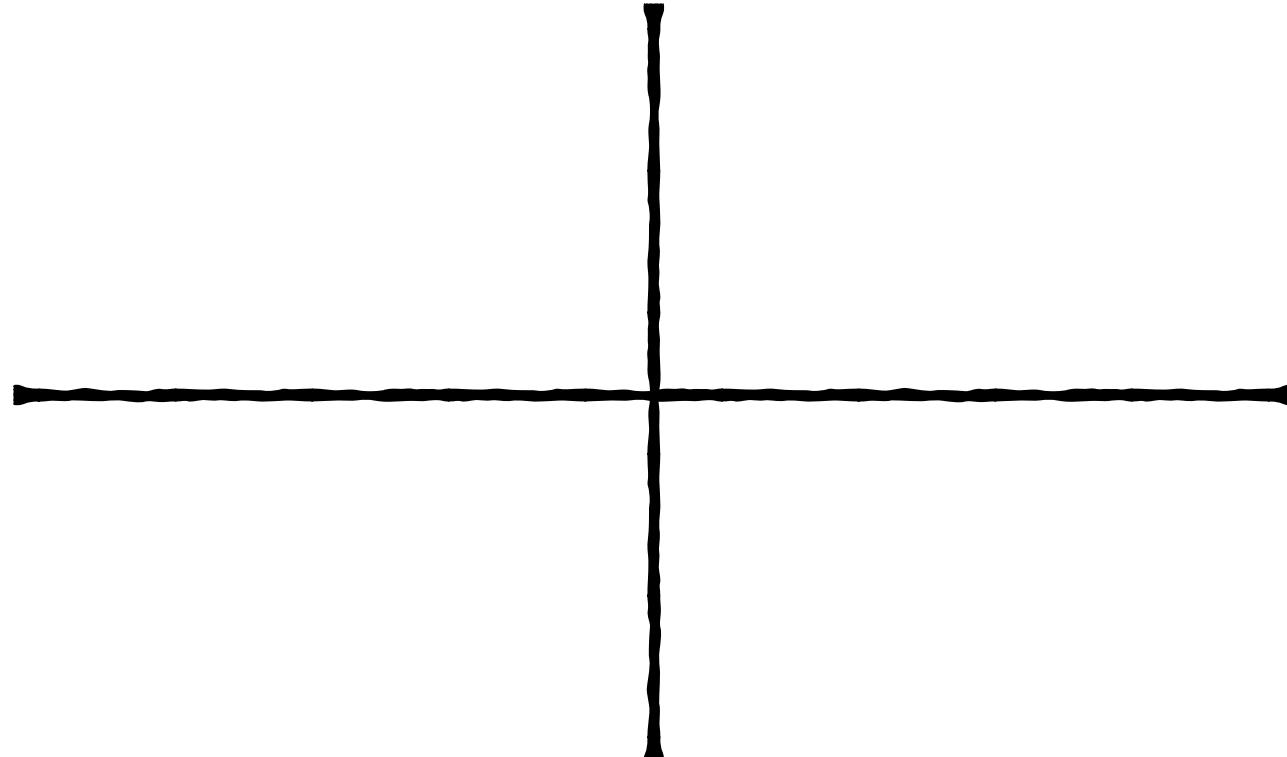
# Fitness Function



# Fitness Function



atomic



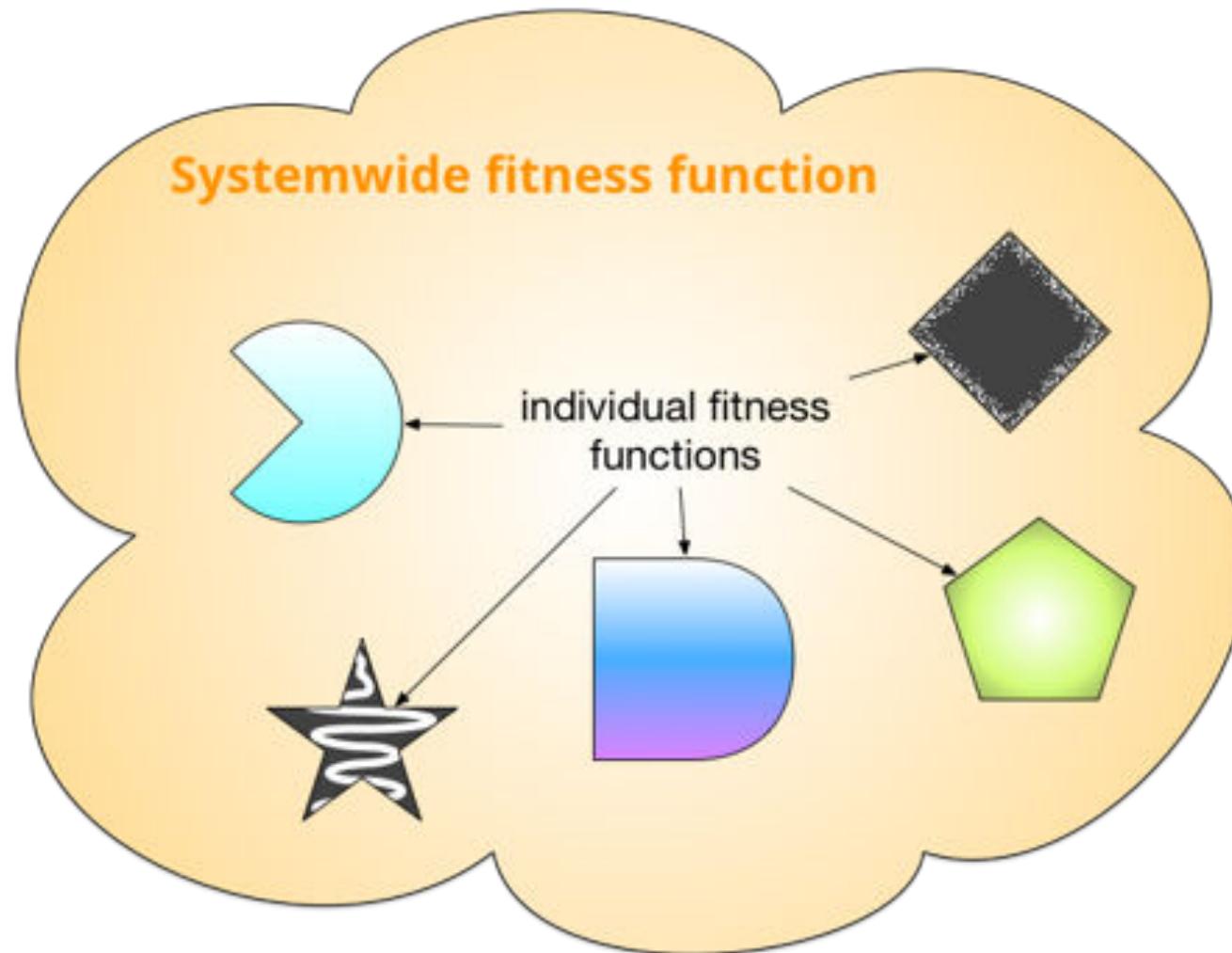
holistic



continuous

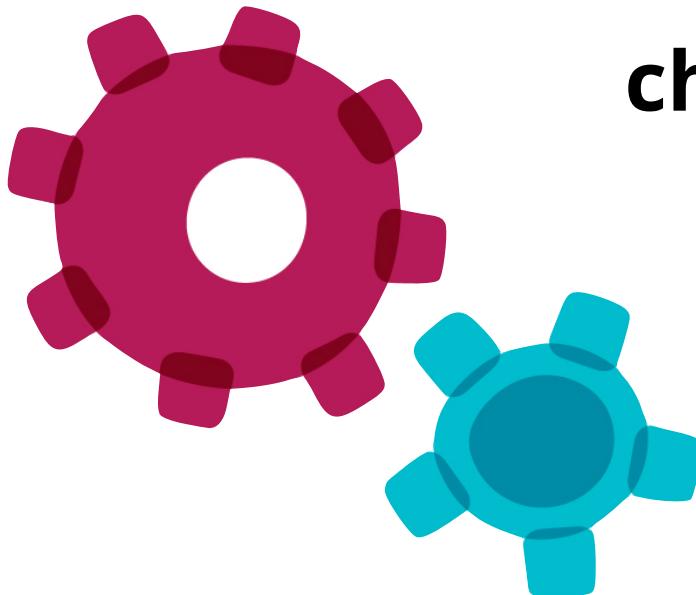


# System-wide Fitness Function



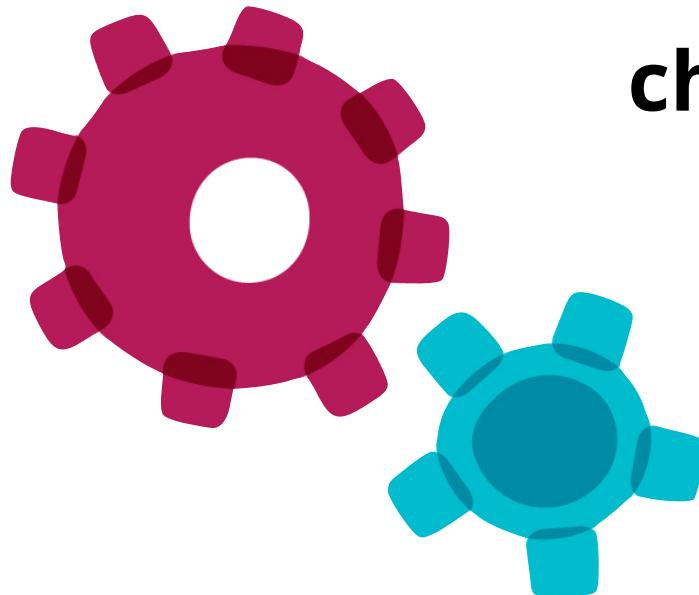
# Implementing Fitness Functions

Protecting architectural  
characteristics

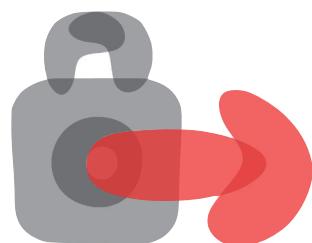


# Implementing Fitness Functions

Protecting architectural  
characteristics



Automating governance



Cyclomatic complexity < 50 for all projects

Naming conventions

immutability

**maintainable?**

(incoming/outgoing)

Controlled afferent/efferent coupling

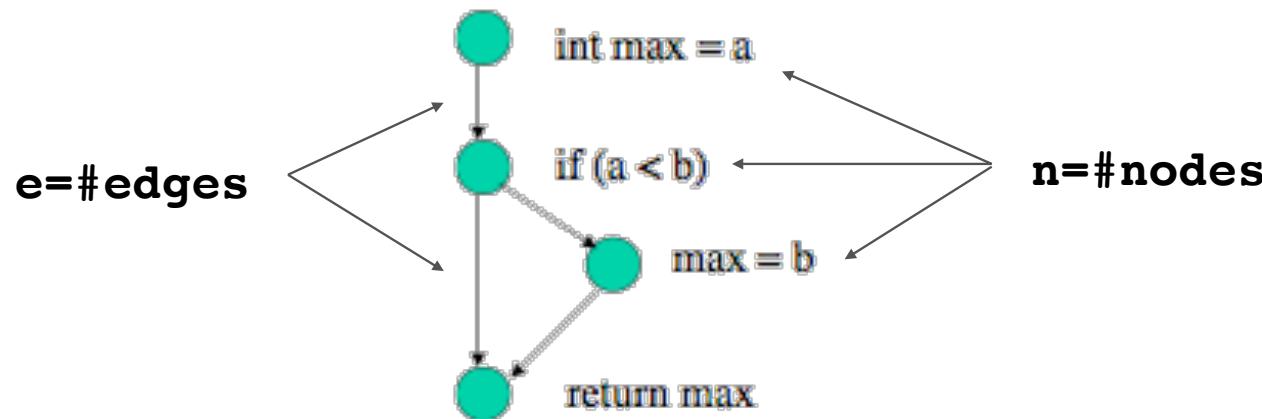
# Concrete Metrics for Architectural Characteristics

# Complexity

## cyclomatic complexity

provides a numeric value representing the complexity of a function or method

$$V(G) = e - n + 2$$

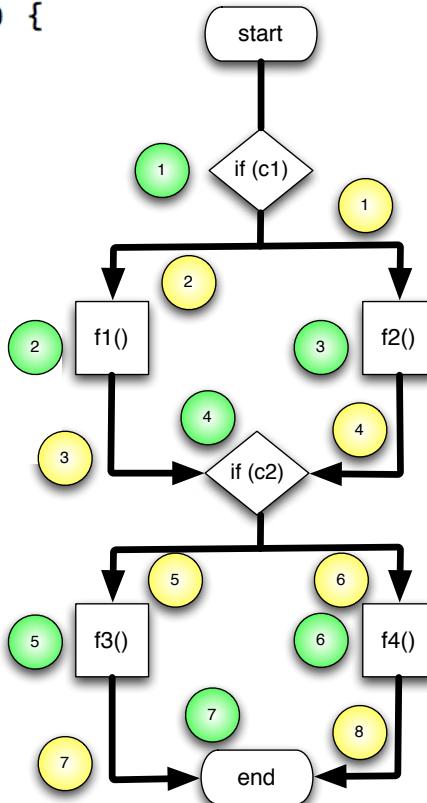


# Complexity

## cyclomatic complexity

```
public void doIt() {  
    if (c1) {  
        f1();  
    } else {  
        f2();  
    }  
    if (c2) {  
        f3();  
    } else {  
        f4();  
    }  
}
```

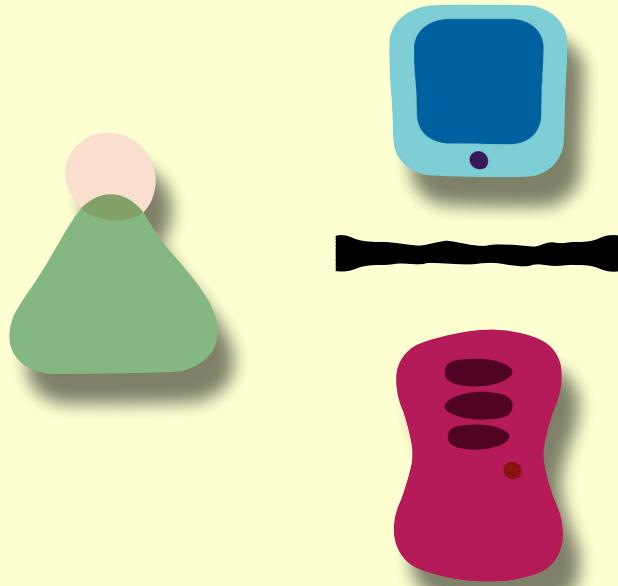
nodes  
edges



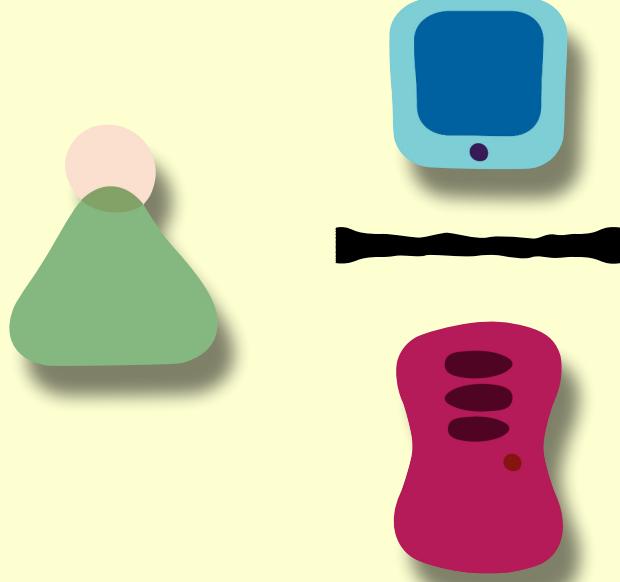
$$V(G) = e - n + 2$$

$$V(G) = 8 - 7 + 2 = 3$$

# Governing Code Quality

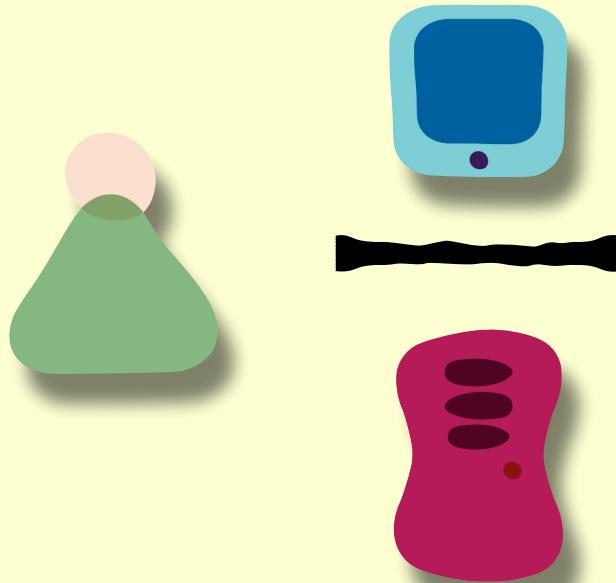


# Governing Code Quality

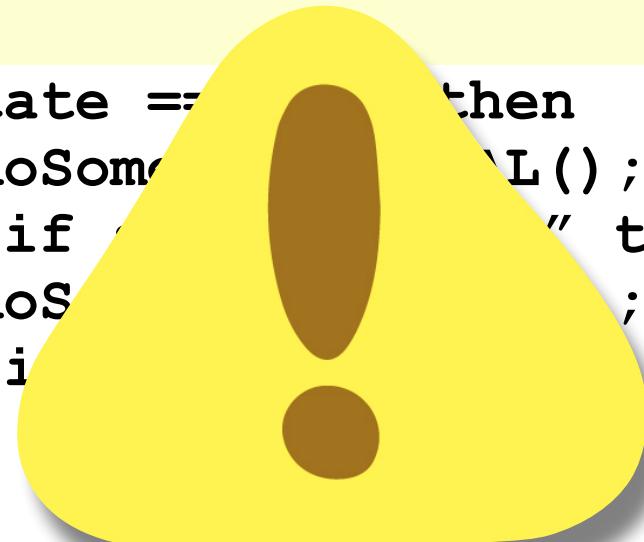


```
if state == "AL" then
    doSomethingForAL();
else if state == "GA" then
    doSomethingForGA();
else if ...
```

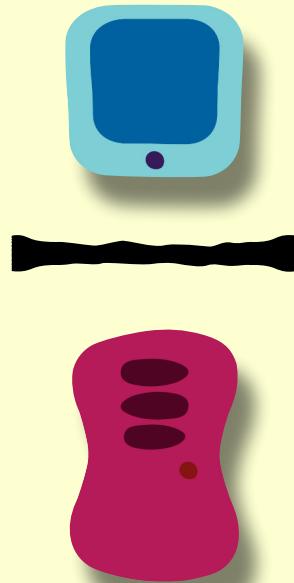
# Governing Code Quality



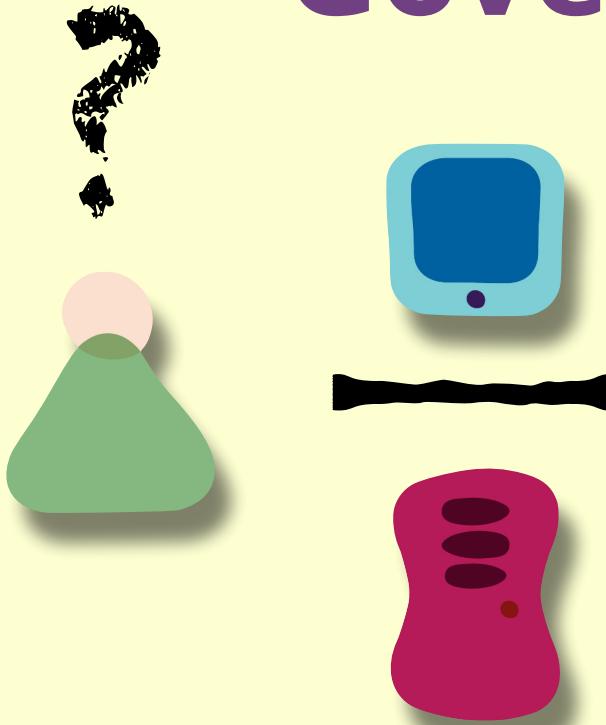
```
if state == "A" then
    doSomethingA();
else if state == "B" then
    doSomethingB();
else if state == "C" then
    doSomethingC();
```



# Governing Code Quality

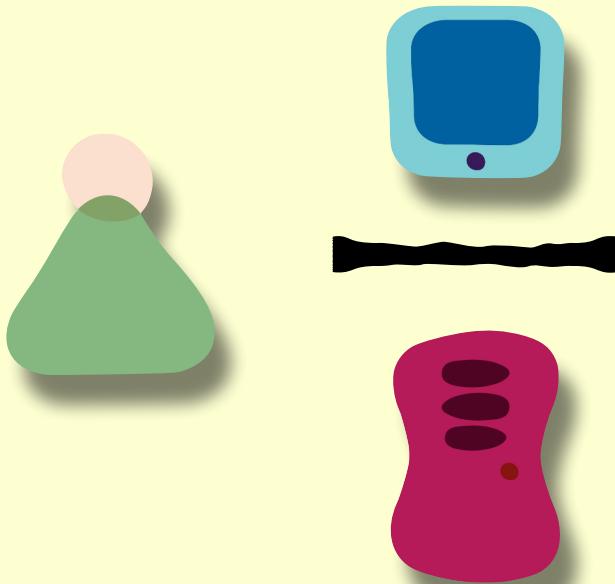


# Governing Code Quality



```
if state == "AL" then
    doSomethingForAL();
else if state == "GA" then
    doSomethingForGA();
else if ...
```

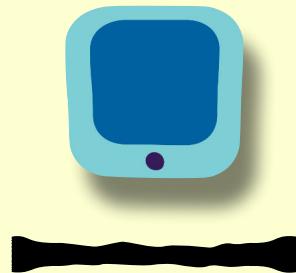
# Governing Code Quality



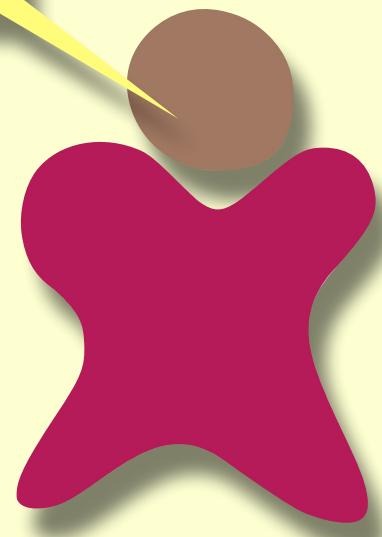
```
if state == "A" then
    doSomethingA();
else if state == "B" then
    doSomethingB();
else if state == "C" then
    doSomethingC();
```



# Governing Code Quality



Strategy Design  
Pattern

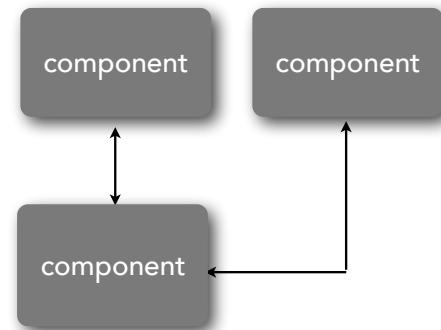


# Metrics

# Component coupling

# Component Coupling

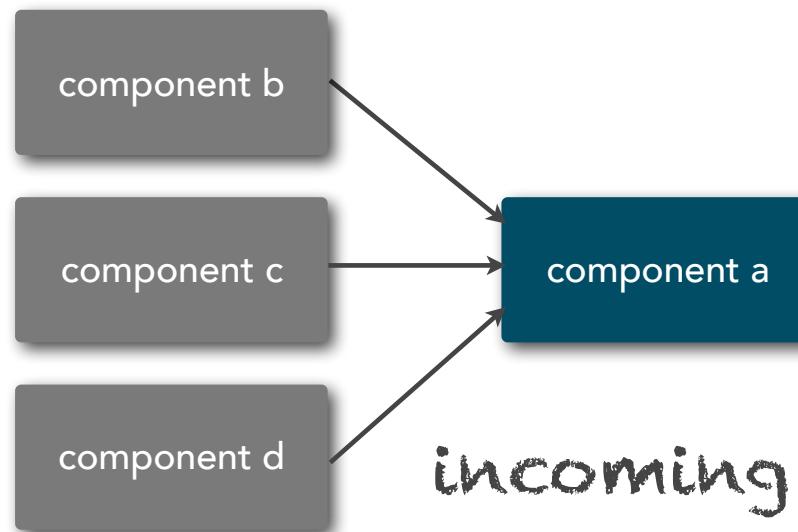
the extent to which components know  
about each other



# Component Coupling

afferent coupling

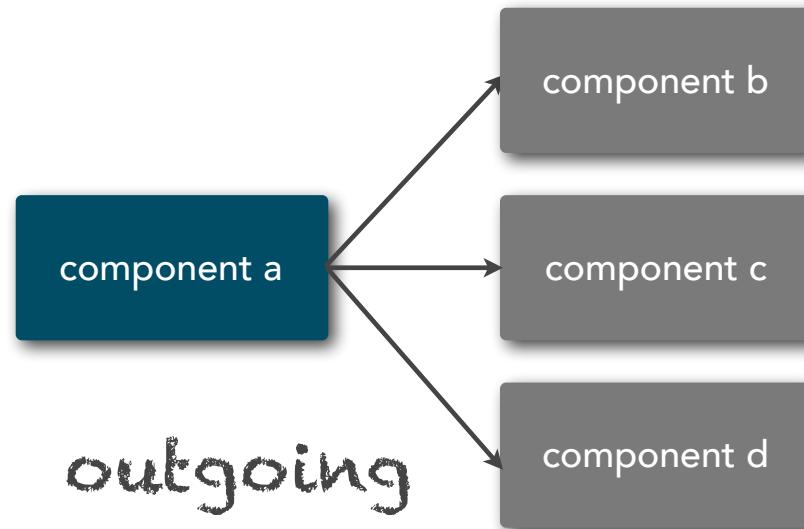
the degree to which other components are dependent on the target component



# Component Coupling

efferent coupling

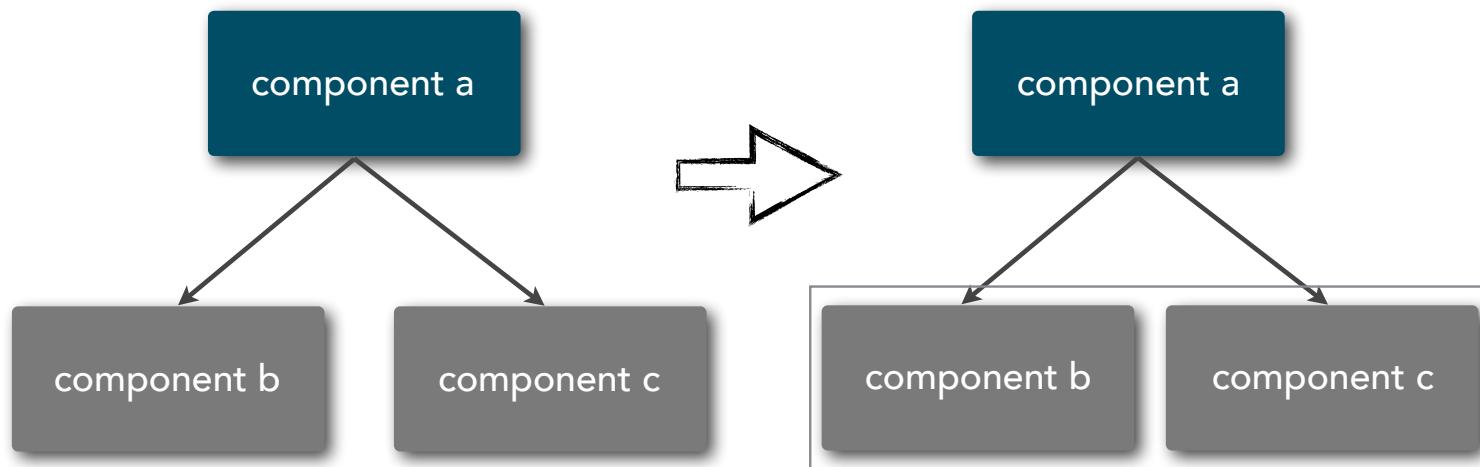
the degree to which the target component  
is dependent on other components



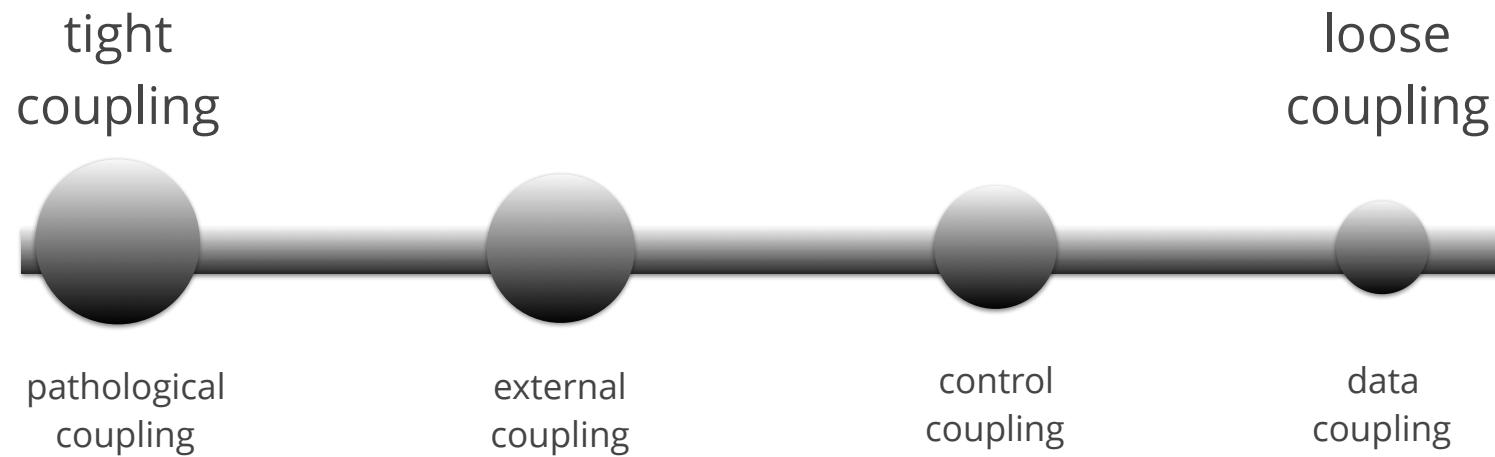
# Component Coupling

temporal coupling

components are coupled due to non-static or timing dependencies



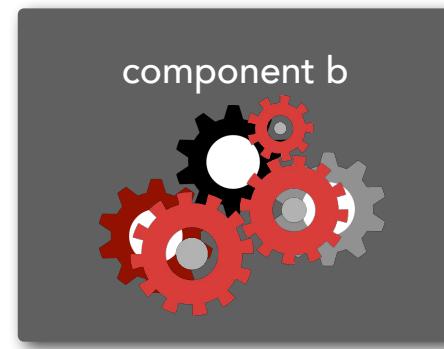
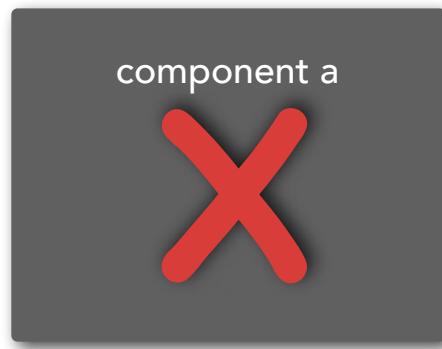
# Component Coupling



# Component Coupling

pathological coupling

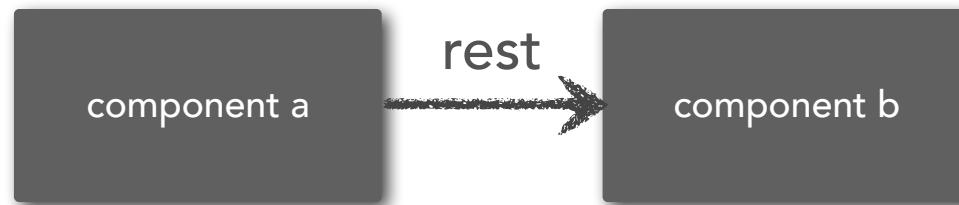
one component relies on the inner workings of another component



# Component Coupling

external coupling

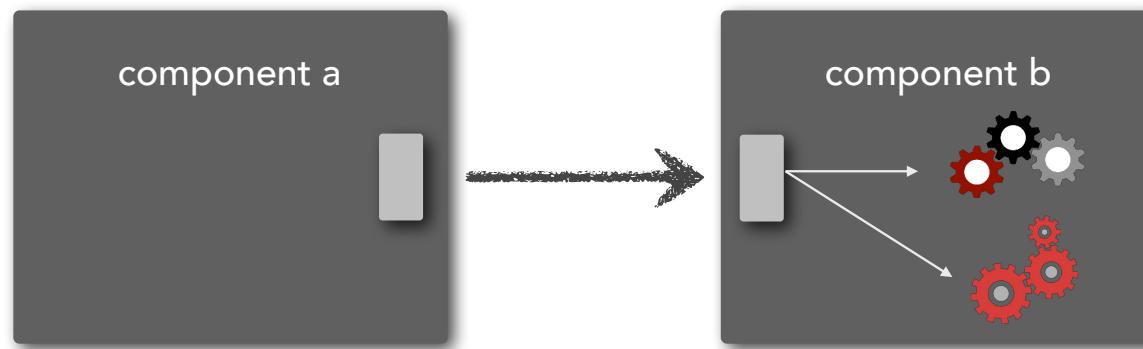
multiple components share an externally imposed protocol or data format



# Component Coupling

control coupling

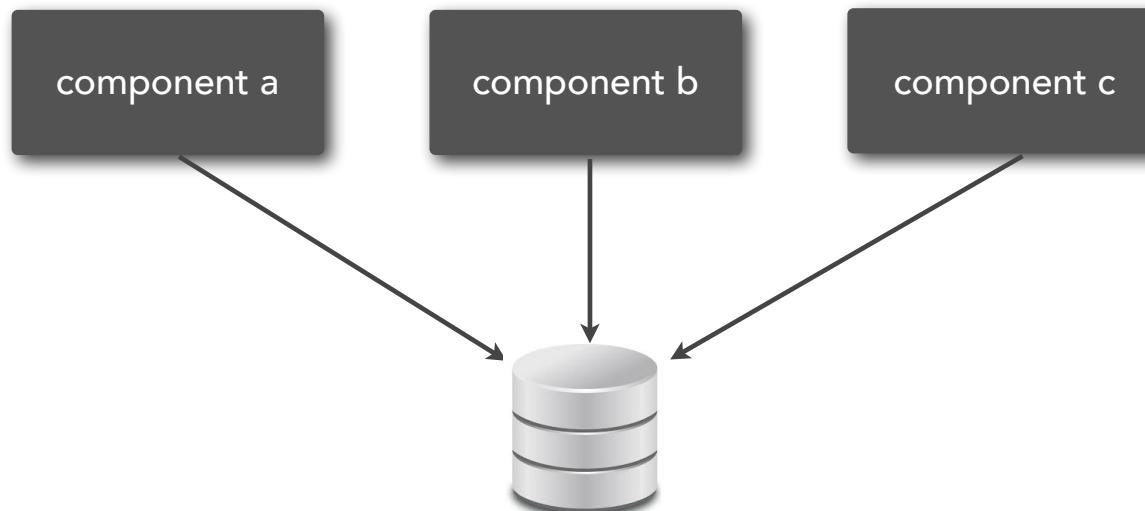
one component passes information to  
another component on what to do



# Component Coupling

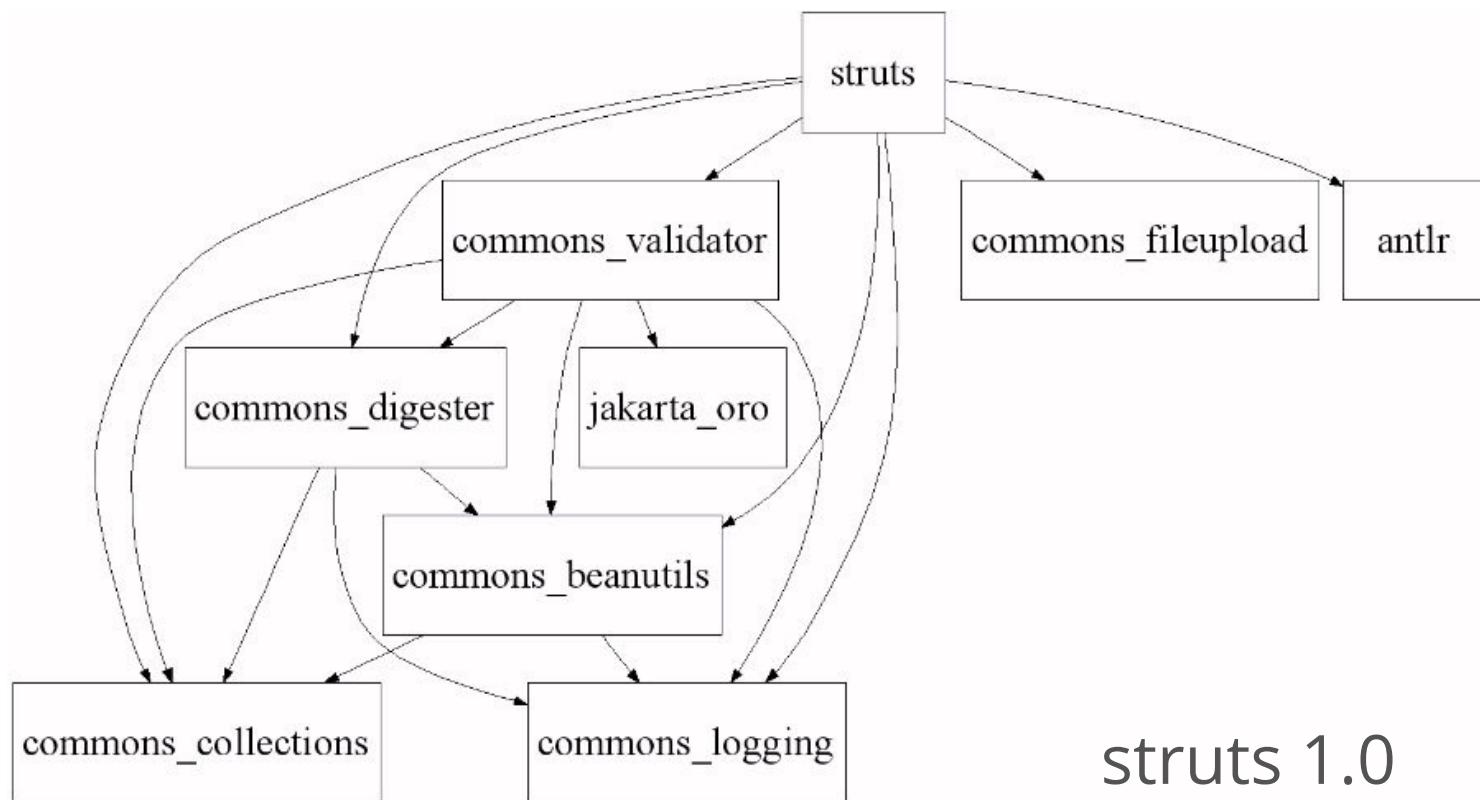
data coupling

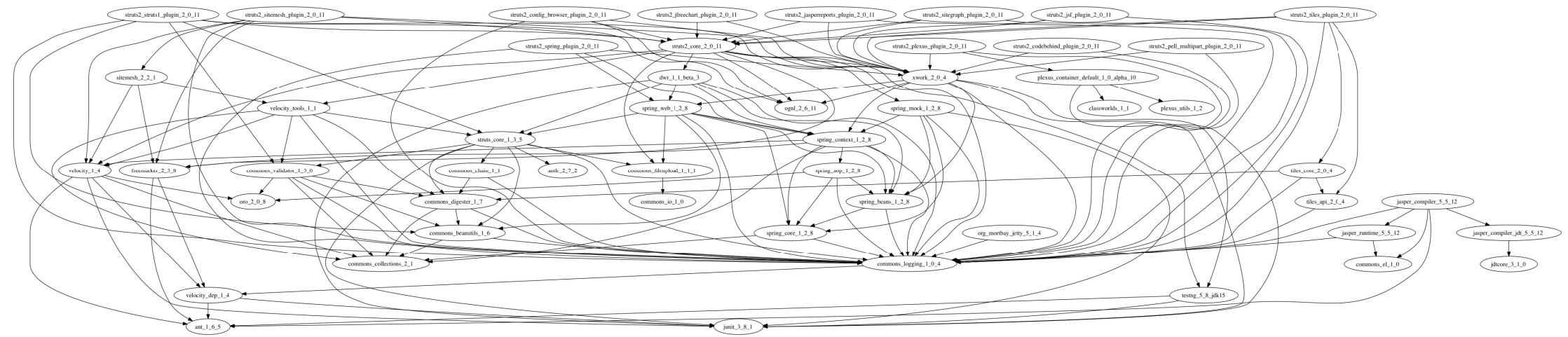
the degree to which components are bound to a shared data context



# Component Coupling

consequences of ignoring...



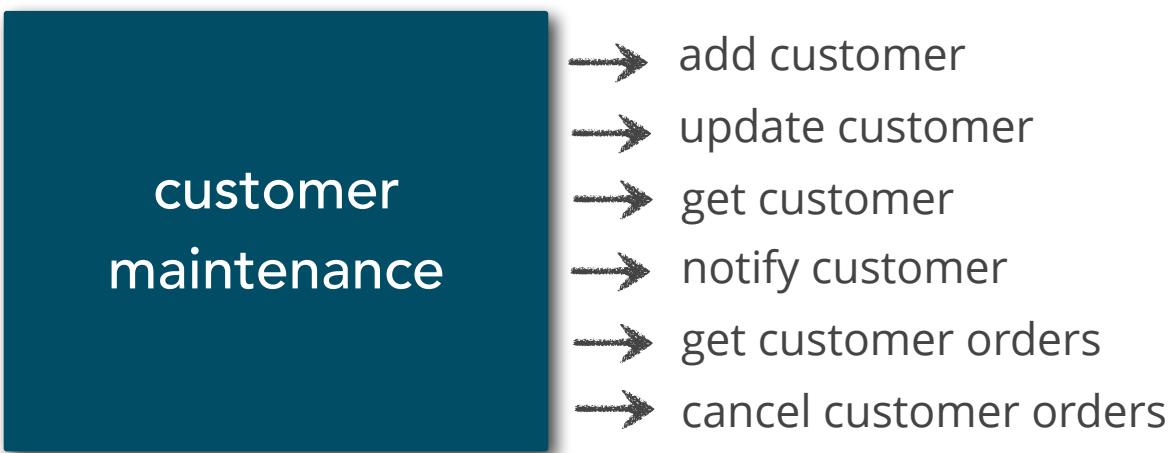


# Component Cohesion

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

# Component Cohesion

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



# Component Cohesion

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



# Component Cohesion

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

customer  
maintenance

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

order  
maintenance

- get customer orders
- cancel customer orders

# Chidamber & Kemerer Metrics

- ✓ DIT (depth of inheritance tree)
- ✓ WMC (weighted methods/class; sum of CC)
- ✓ CE (efferent coupling count)
- ✓ CA (afferent coupling count)

# Chidamber & Kemerer Metrics

✓ LCOM (Lack of Cohesion in Methods)

$$LCOM96b = \frac{1}{a} \sum_{j=1}^a m - \mu(Aj)$$

# Chidamber & Kemerer Metrics

✓ LCOM (Lack of Cohesion in Methods)

The LCOM is a count of the number of method pairs whose similarity is 0 (i.e.  $\sigma()$  is a null set) minus the count of method pairs whose similarity is not zero.

# Chidamber & Kemerer Metrics

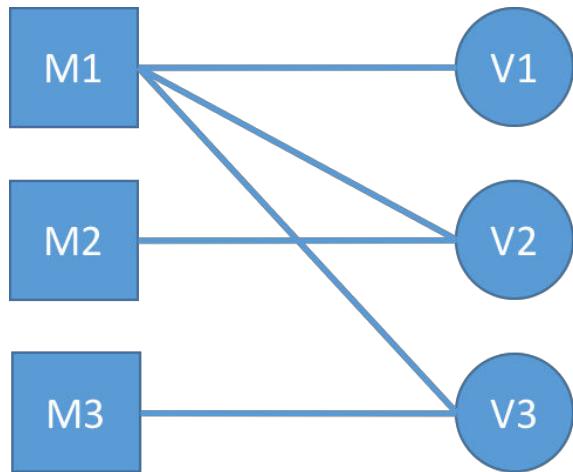
## ✓ LCOM (Lack of Cohesion in Methods)

The LCOM is a count of the number of method pairs whose similarity is 0 (i.e.  $\sigma()$  is a null set) minus the count of method pairs whose similarity is not zero.

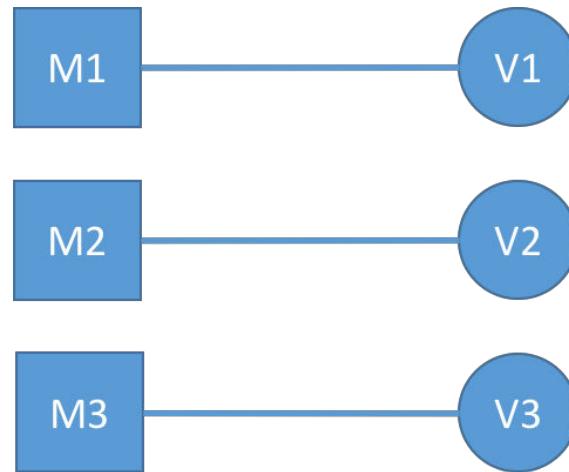
The sum of sets of methods not shared via sharing fields.

# Chidamber & Kemerer Metrics

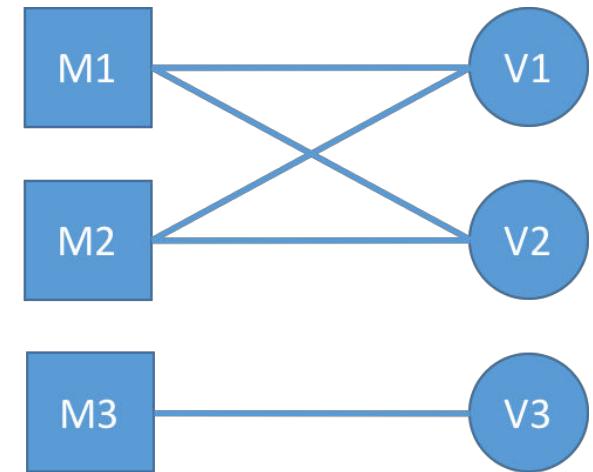
✓ LCOM (Lack of Cohesion in Methods)



(A)



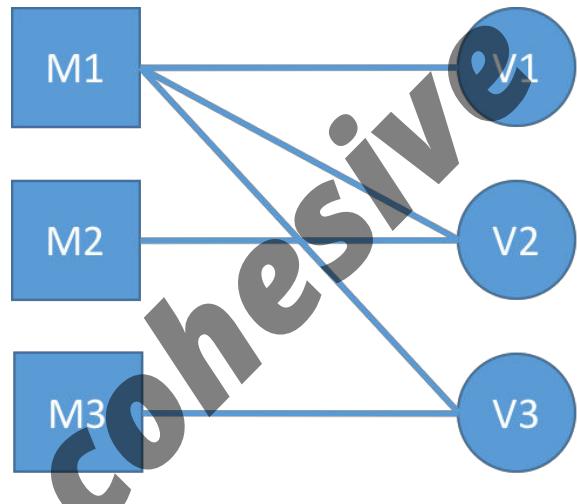
(B)



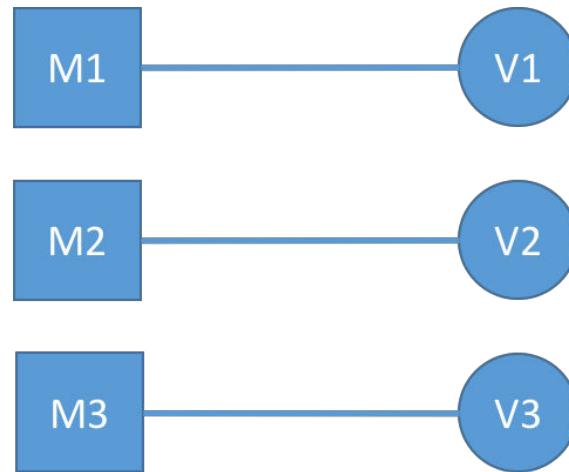
(C)

# Chidamber & Kemerer Metrics

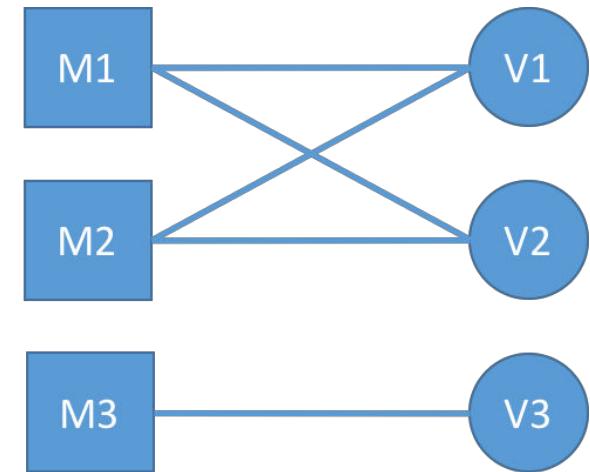
✓ LCOM (Lack of Cohesion in Methods)



(A)



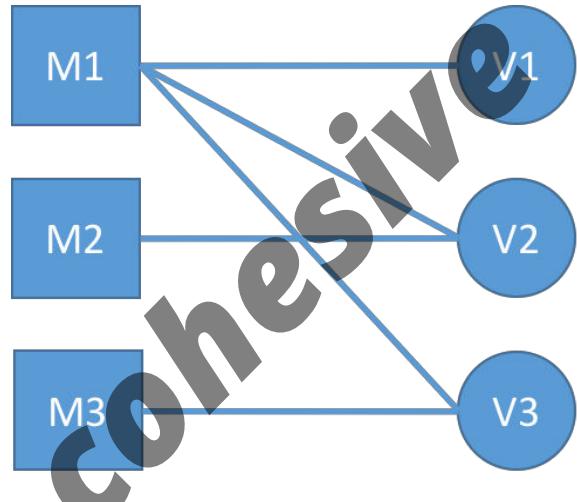
(B)



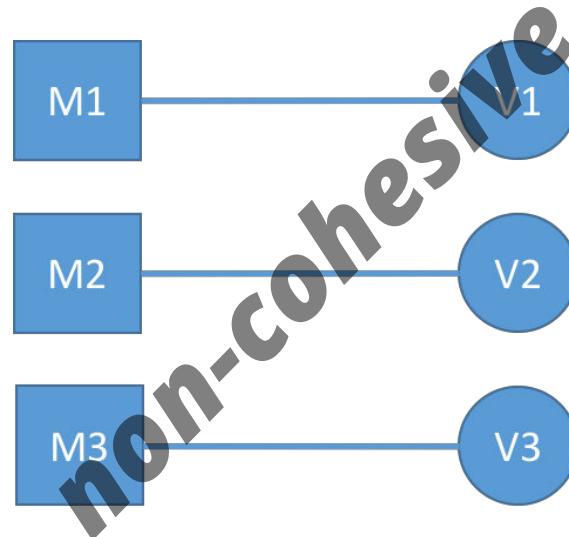
(C)

# Chidamber & Kemerer Metrics

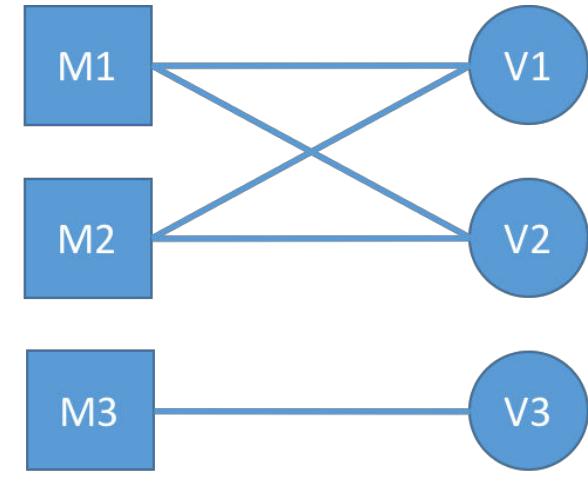
✓ LCOM (Lack of Cohesion in Methods)



(A)



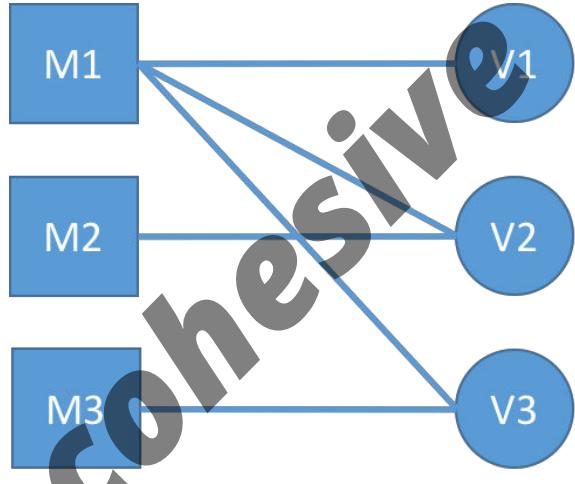
(B)



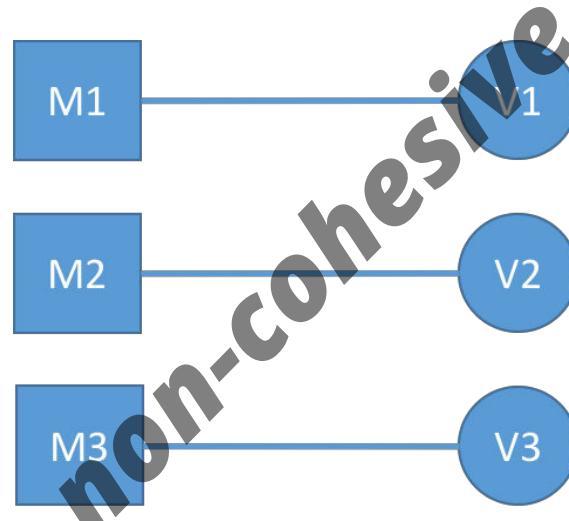
(C)

# Chidamber & Kemerer Metrics

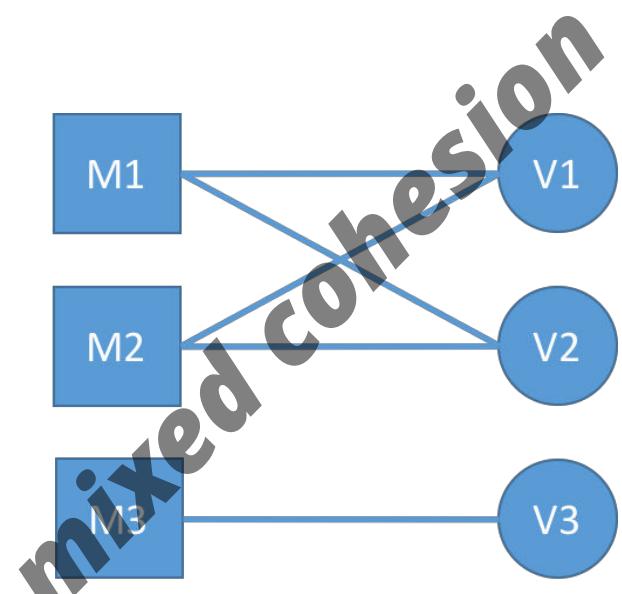
✓ LCOM (Lack of Cohesion in Methods)



(A)



(B)



(C)

# Metrics ∩ Architecture Characteristics

core metrics

✓ number of classes per package

✓ number of lines of source code per package

✓ percent comments (range: 8-20)

✓ max complexity (1+num\_paths thru method; range: 2-8)

✓ average complexity (range: 2.0 - 4.0)

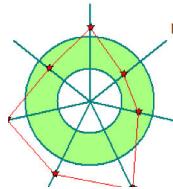
# Metrics ∩ Architecture Characteristics

Chidamber & Kemerer Metrics

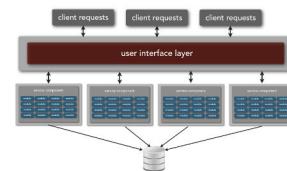
- ✓ DIT (depth of inheritance tree)
- ✓ WMC (weighted methods/class; sum of CC)
- ✓ CE (efferent coupling count)
- ✓ CA (afferent coupling count)

# Metrics ∩ Architecture Characteristics

architecture characteristics mapping



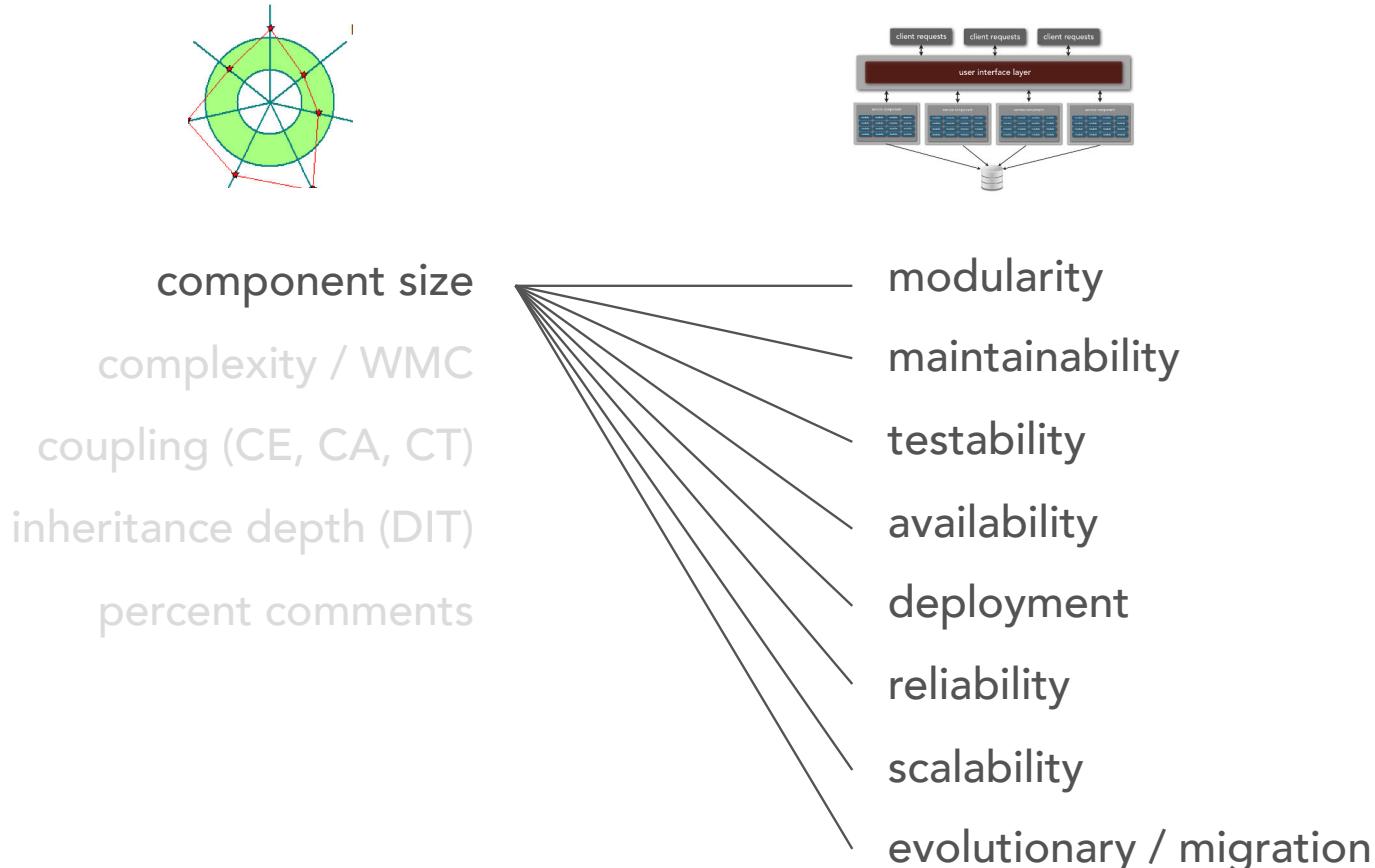
component size  
complexity / WMC  
coupling (CE, CA, CT)  
inheritance depth (DIT)  
percent comments



modularity  
maintainability  
testability  
availability  
deployment  
reliability  
scalability  
evolutionary / migration

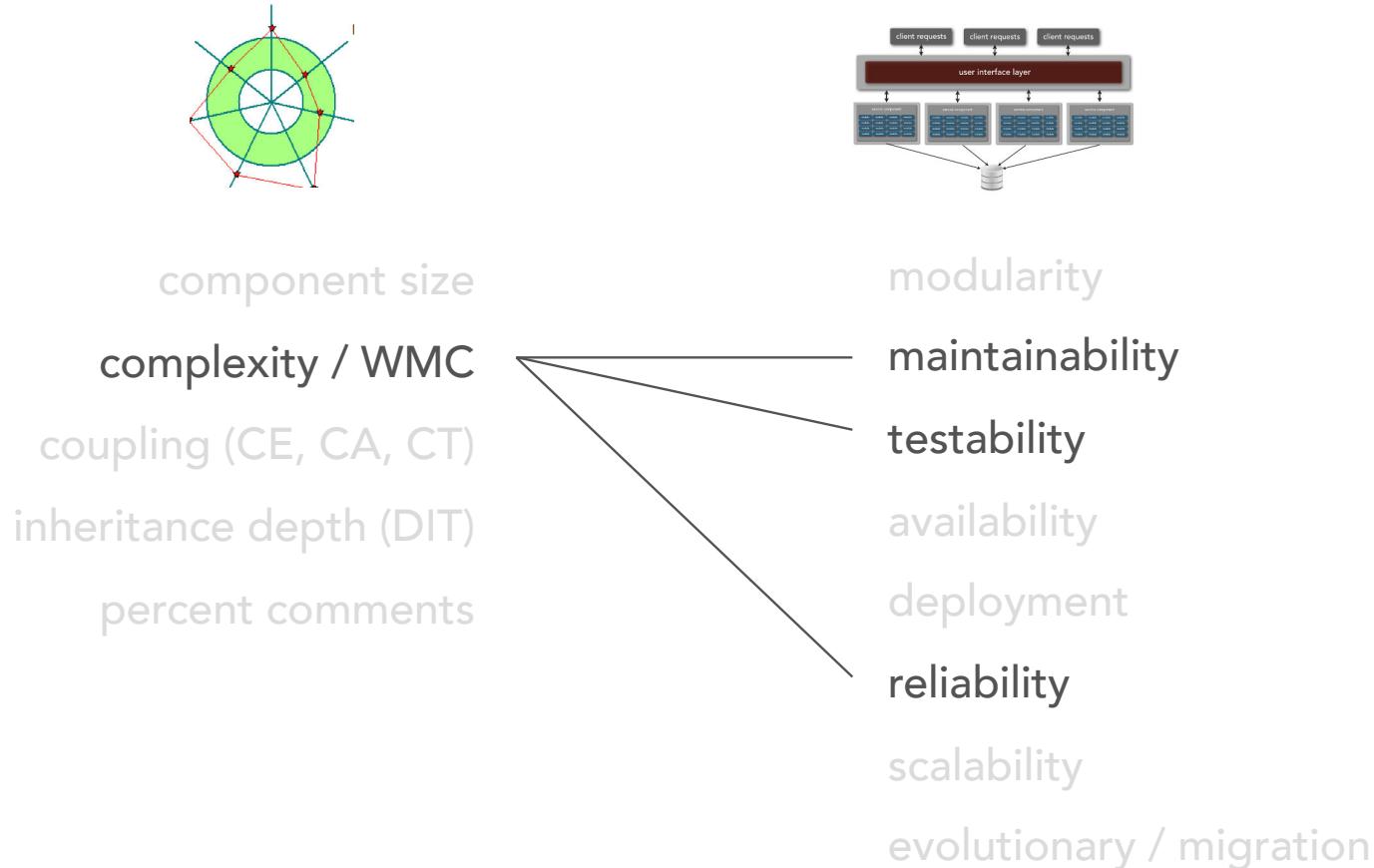
# Metrics ∩ Architecture Characteristics

architecture characteristics mapping



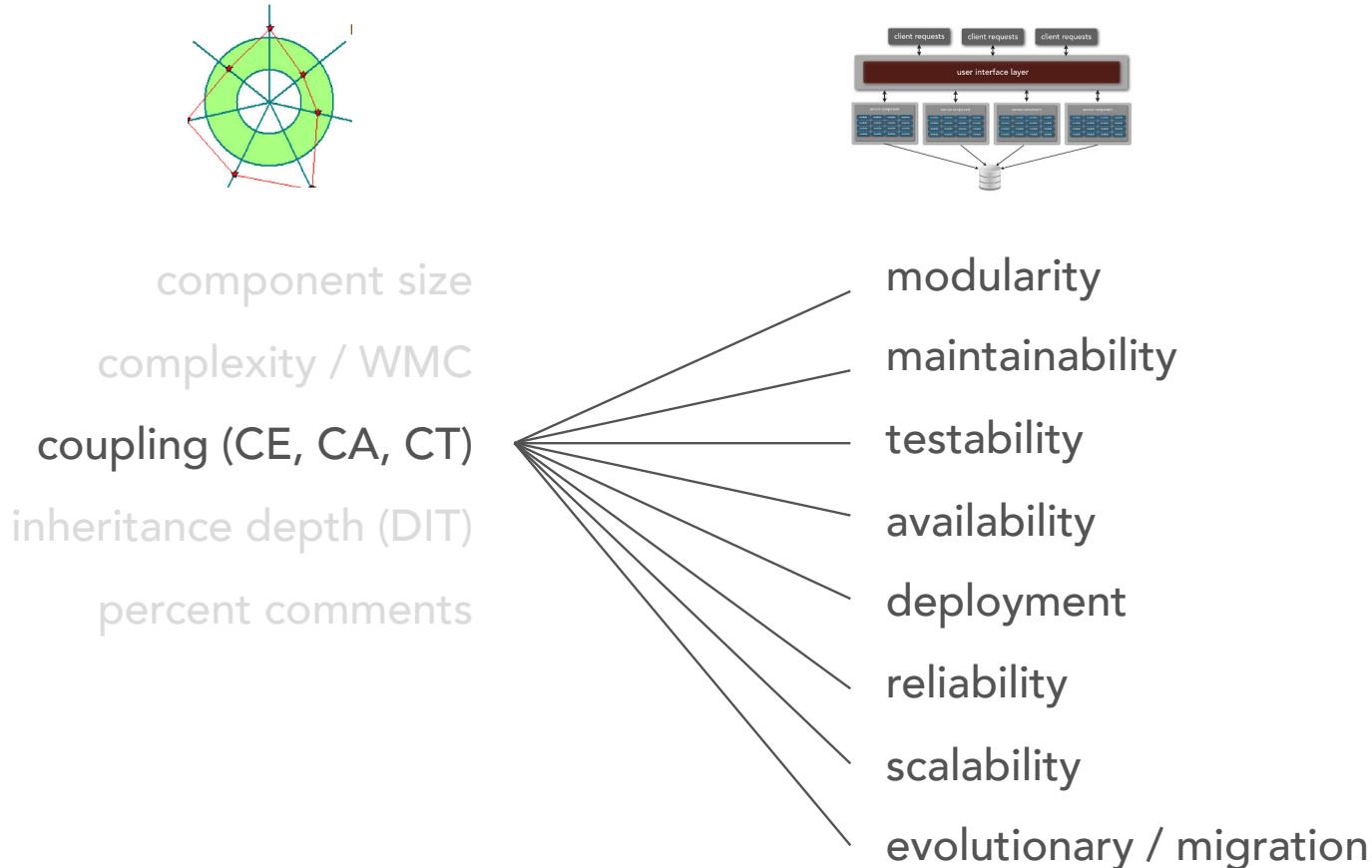
# Metrics ∩ Architecture Characteristics

architecture characteristics mapping



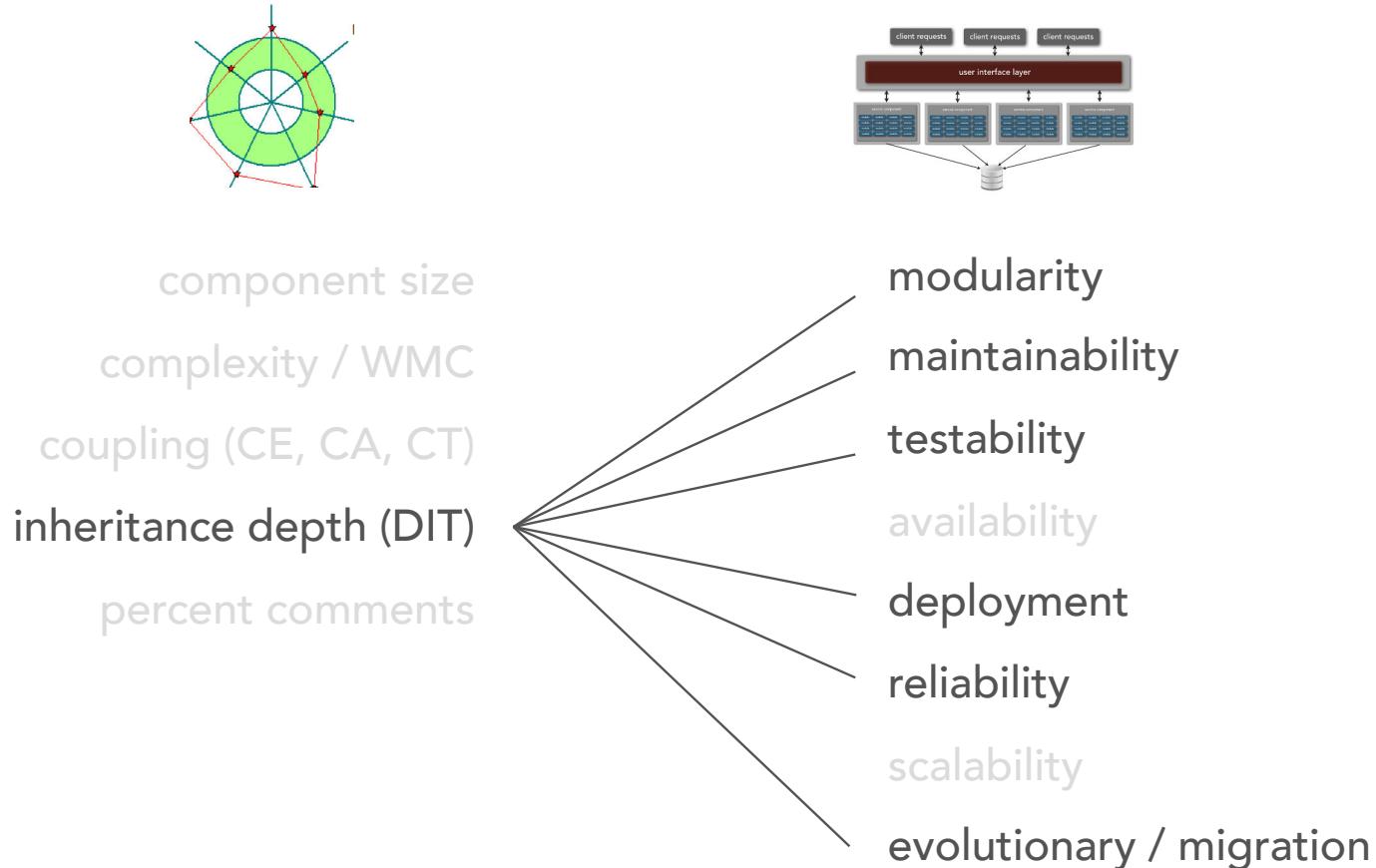
# Metrics ∩ Architecture Characteristics

architecture characteristics mapping



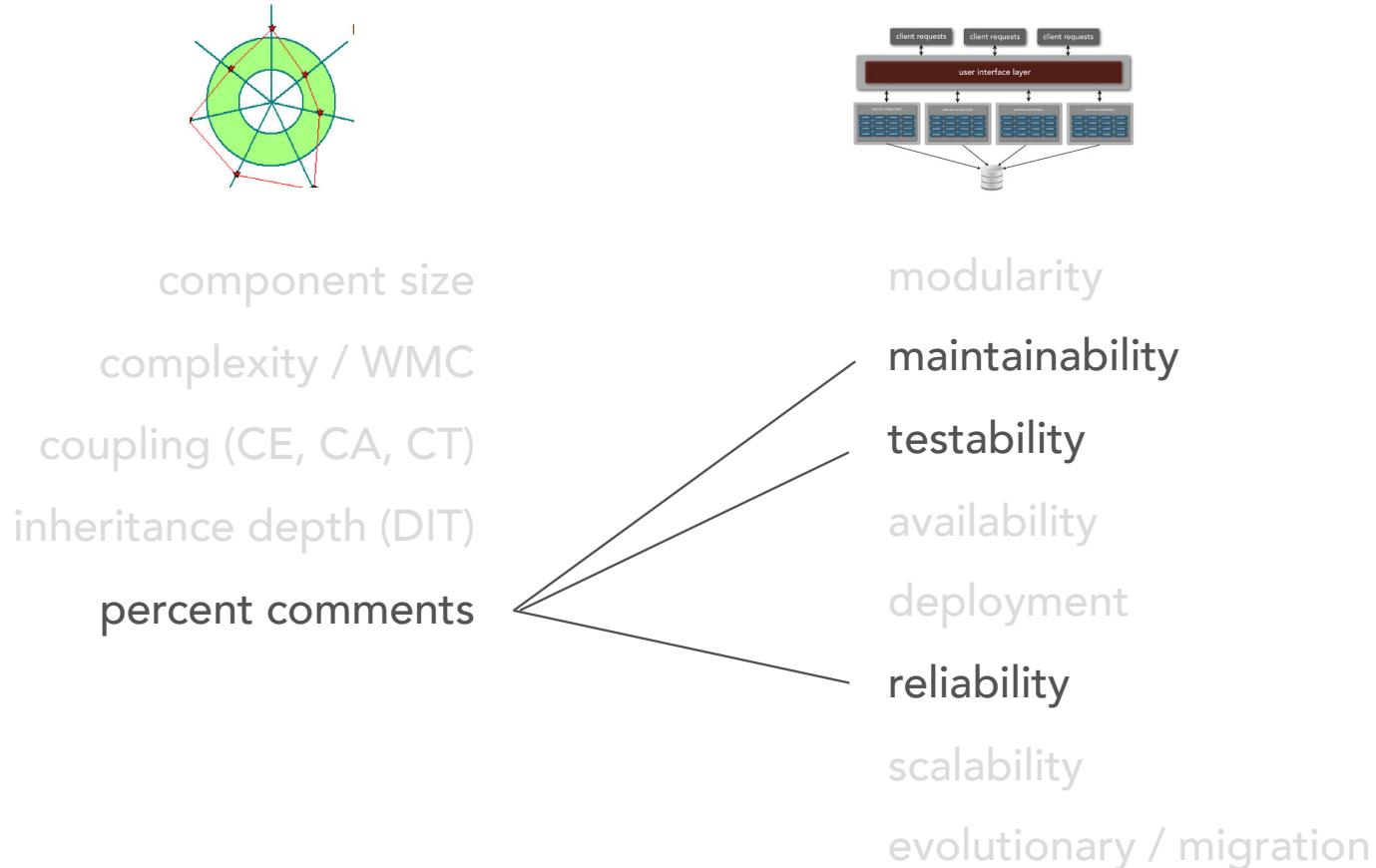
# Metrics ∩ Architecture Characteristics

architecture characteristics mapping



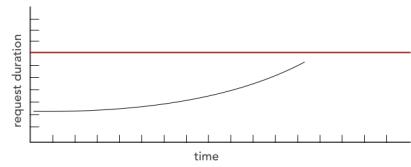
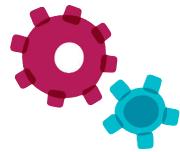
# Metrics ∩ Architecture Characteristics

architecture characteristics mapping

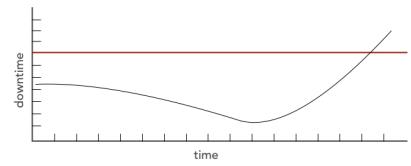
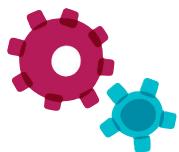




# Performance



measure and track average response times (cumulative)

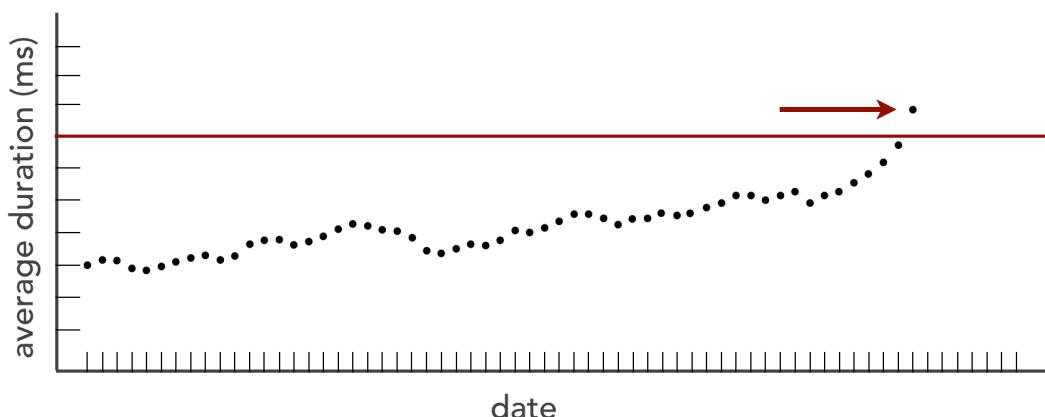


measure and track maximum response times

# Cumulative Average Response Times

(per request, domain, or application context)

trigger alert when the average response times within a particular context exceeds 1400ms



atomic

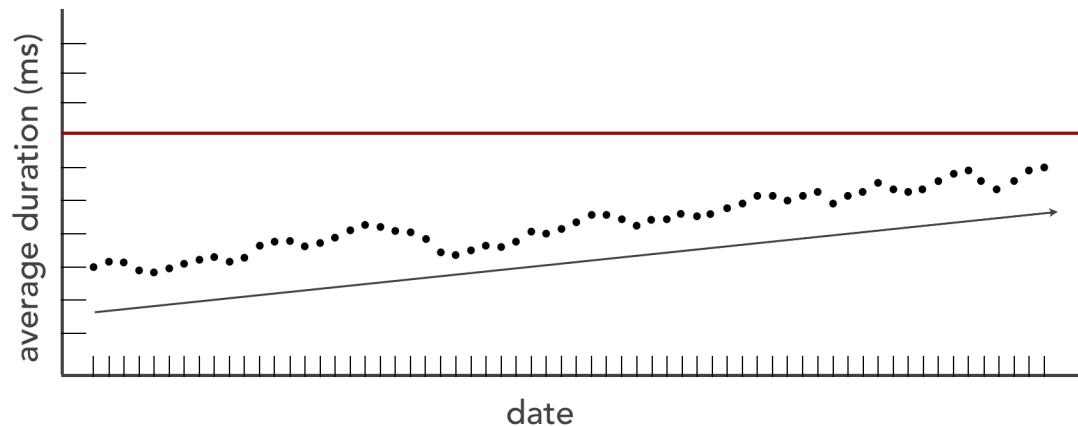


continuous

# Cumulative Average Response Times

(per request, domain, or application context)

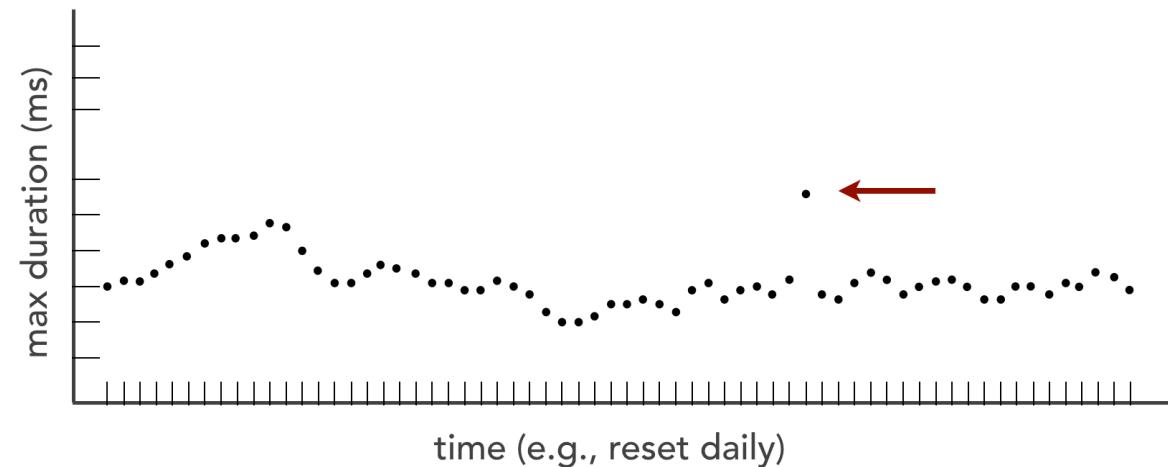
trigger alert when the average response times continues to increase over a 2 month period of time



# Maximum Response Times

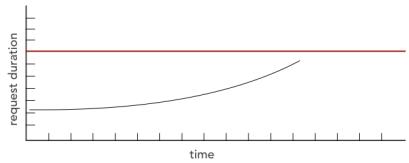
(per request, domain, or application context)

trigger alert anytime the maximum response time of selected requests exceeds 2000ms

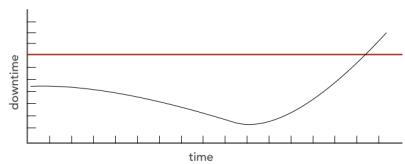




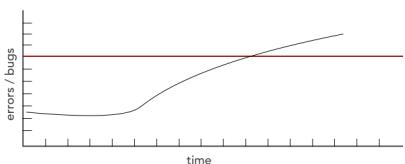
# Scalability



measure and track average response times across number of users or business requests



measure and track timeouts across number of users or business requests



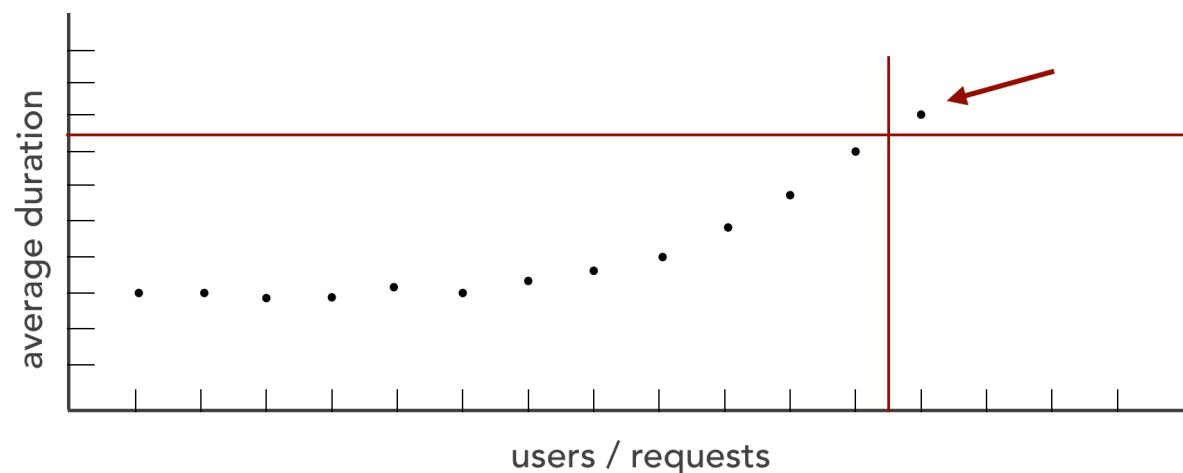
measure and track application faults (crashes) across number of users or business requests



# Average Response Times

(per request, domain, or application context)

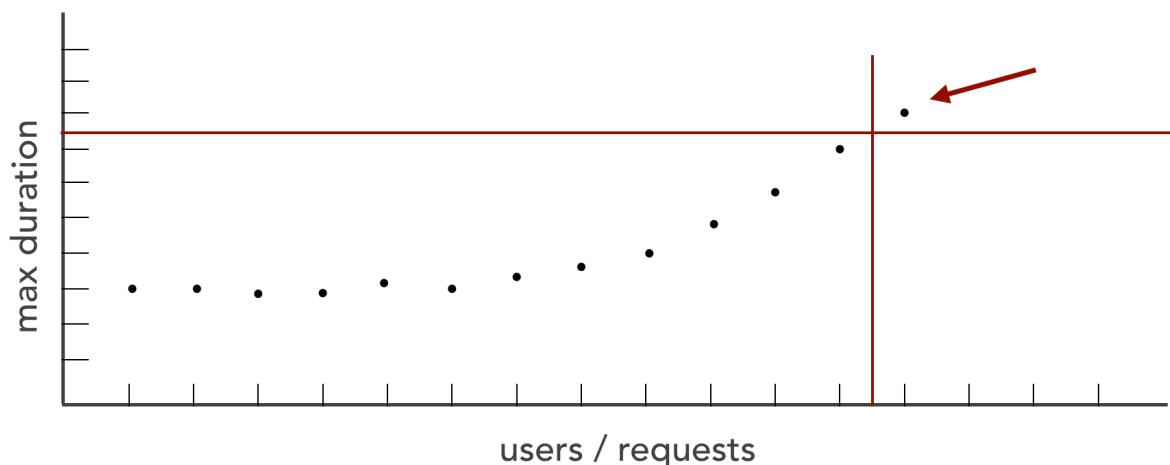
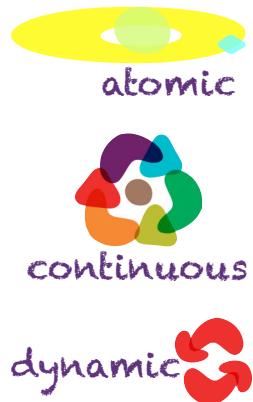
trigger alert if the average response time increases by 10% when the number of users increases over 7%



# Maximum Response Times

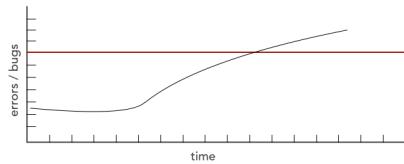
(per request, domain, or application context)

trigger alert if the maximum response time increases over 5% when the number of users increases over 10%

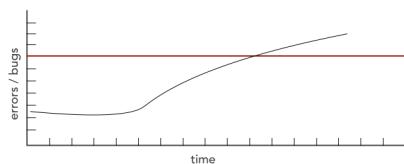




# Fault Tolerance



measure and track number of users impacted  
by a system or service crash

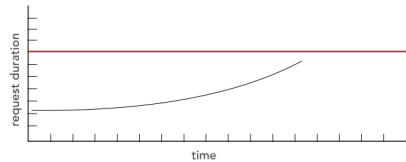


measure and track number of requests  
impacted by a system or service crash

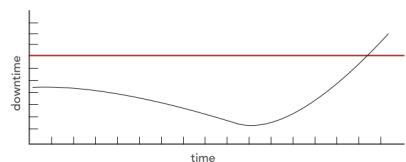


# Deployability

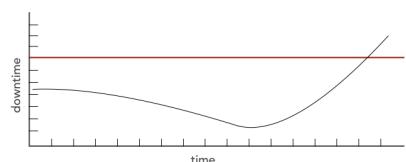
the ease of, risk, and frequency of deployment



measure and track number of actual hours spent to deploy



measure and track failed deployments or errors resulting from deployment

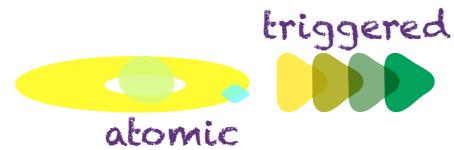
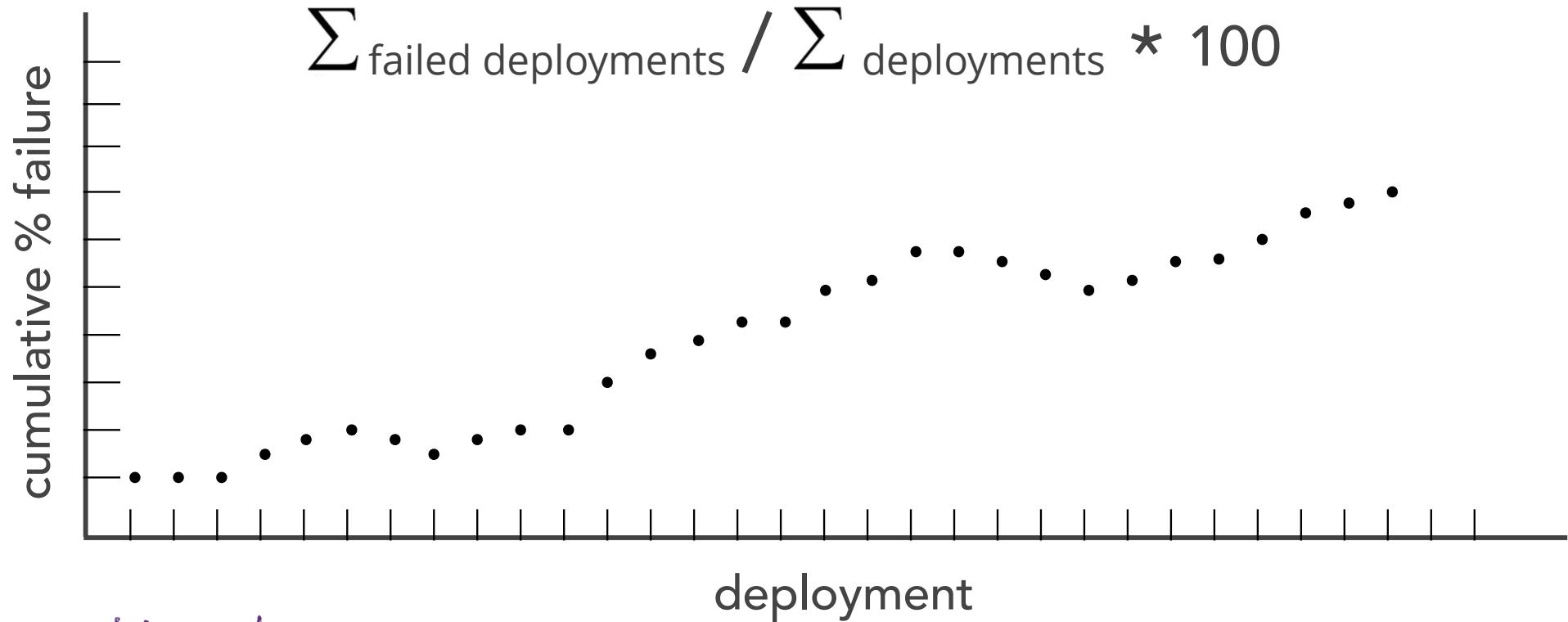


measure and track the frequency of deployment



# Deployment Errors/Failures

(per service, domain, or application context)



# ArchUnit

<https://www.archunit.org/>

The screenshot shows a web browser window displaying the ArchUnit website at https://www.archunit.org/. The page has a blue header bar with the ArchUnit logo and navigation links for Getting Started, Motivation, News, User Guide, API, and About. Below the header is a large blue banner with the text "Persistence" and "Unit test your Java architecture". A subtext below the banner reads: "Start enforcing your architecture within 30 minutes using the test setup you already have." A "Start Now" button is visible. At the bottom of the page, there is a section titled "News" which contains a paragraph about the ArchUnit library.

Persistence

## Unit test your Java architecture

Start enforcing your architecture within 30 minutes using the test setup you already have.

**Start Now**

ArchUnit is a free, simple and extensible library for checking the architecture of your Java code using any plain Java unit test framework. That is, ArchUnit can check dependencies between packages and classes, layers and slices, check for cyclic dependencies and more. It does so by analyzing given Java bytecode, importing all classes into a Java code structure. You can find examples for the current release at [ArchUnit Examples](#) and the sources on [GitHub](#).

### News

# ArchUnit

<https://www.archunit.org/>

coding rules

```
import static com.tngtech.archunit.lang.syntax.ArchRuleDefinition.noClasses;
import static com.tngtech.archunit.library.GeneralCodingRules.ACCESS_STANDARD_STREAMS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING;

public class CodingRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
    }

    @Test
    public void classes_should_not_access_standard_streams_defined_by_hand() {
        noClasses().should(ACCESS_STANDARD_STREAMS).check(classes);
    }

    @Test
    public void classes_should_not_access_standard_streams_from_library() {
        NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS.check(classes);
    }

    @Test
    public void classes_should_not_throw_generic_exceptions() {
        NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS.check(classes);
    }

    @Test
    public void classes_should_not_use_java_util_logging() {
        NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING.check(classes);
    }
}
```

# ArchUnit

@Test

<https://www.archunit.org/>

```
public void classes_should_not_access_standard_streams_from_library() {  
    NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS.check(classes);  
}
```

@Test

```
public void classes_should_not_throw_generic_exceptions() {  
    NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS.check(classes);  
}
```

@Test

```
public void classes_should_not_use_java_util_logging() {  
    NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING.check(classes);  
}
```

coding rules

# ArchUnit

<https://www.archunit.org/>

```
public class InterfaceRules {

    @Test
    public void interfaces_should_not_have_names_ending_with_the_word_interface() {
        JavaClasses classes = new ClassFileImporter().importClasses(
            SomeBusinessInterface.class,
            SomeDao.class
        );

        noClasses().that().areInterfaces().should().haveNameMatching(".*Interface").check(classes);
    }

    @Test
    public void interfaces_should_not_have_simple_class_names_ending_with_the_word_interface() {
        JavaClasses classes = new ClassFileImporter().importClasses(
            SomeBusinessInterface.class,
            SomeDao.class
        );

        noClasses().that().areInterfaces().should().haveSimpleNameContaining("Interface").check(classes);
    }

    @Test
    public void interfaces_must_not_be_placed_in_implementation_packages() {
        JavaClasses classes = new ClassFileImporter().importPackagesOf(SomeInterfacePlacedInTheWrongPackage.class);

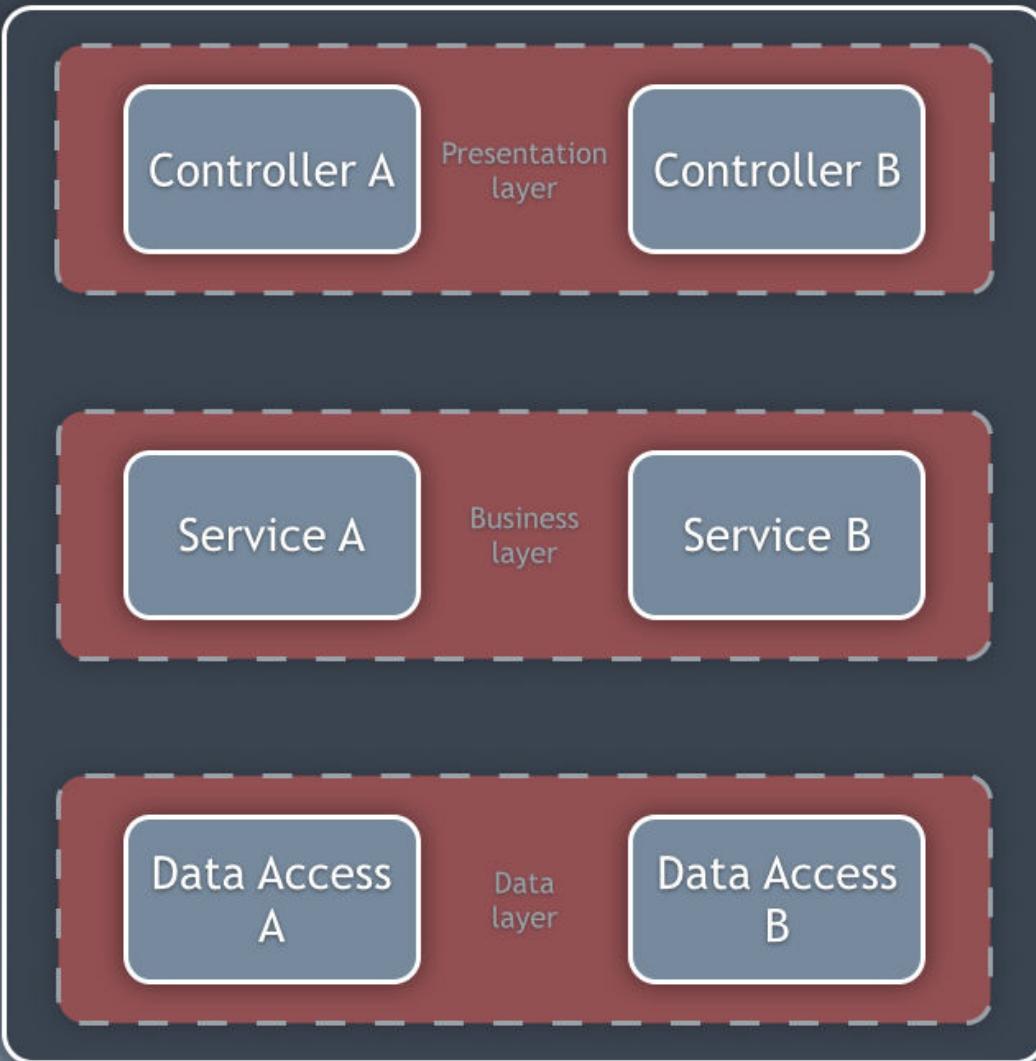
        noClasses().that().resideInAPackage("..impl..").should().beInterfaces().check(classes);
    }
}
```

interface rules

# ArchUnit

<https://www.archunit.org/>

```
public class InterfaceRules {  
  
    @Test  
    public void interfaces_should_not_have_names_ending_with_the_word_interface() {  
        JavaClasses classes = new ClassFileImporter().importClasses(  
            SomeBusinessInterface.class,  
            SomeDao.class  
        );  
  
        noClasses().that().areInterfaces().should().haveNameMatching(".*Interface")  
    }  
  
    @Test  
    public void interfaces_should_not_have_simple_class_names_ending_with_the_word
```



Package by layer (horizontal slicing)

```
public class LayerDependencyRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
    }

    @Test
    public void services_should_not_access_controllers() {
        noClasses().that().resideInAPackage("..service..")
            .should().accessClassesThat().resideInAPackage("..controller..").check(classes);
    }

    @Test
    public void persistence_should_not_access_services() {
        noClasses().that().resideInAPackage("..persistence..")
            .should().accessClassesThat().resideInAPackage("..service..").check(classes);
    }

    @Test
    public void services_should_only_be_accessed_by_controllers_or_other_services() {
        classes().that().resideInAPackage("..service..")
            .should().onlyBeAccessed().byAnyPackage("..controller..", "..service..").check(classes);
    }
}
```

# layer dependency

```
private JavaClasses classes;

@Before
public void setUp() throws Exception {
    classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
}

@Test
public void services_should_not_access_controllers() {
    noClasses().that().resideInAPackage("..service..")
        .should().accessClassesThat().resideInAPackage("..controller..").check(classes);
}

@Test
public void persistence_should_not_access_services() {
    noClasses().that().resideInAPackage("..persistence..")
        .should().accessClassesThat().resideInAPackage("..service..").check(classes);
}
```

# ArchUnit

<https://www.archunit.org/>

```
@Test
public void third_party_class_should_only_be_instantiated_via_workaround() {
    classes().should(notCreateProblematicClassesOutsideOfWorkaroundFactory()
        .as(THIRD_PARTY_CLASS_RULE_TEXT))
        .check(classes);
}

private ArchCondition<JavaClass> notCreateProblematicClassesOutsideOfWorkaroundFactory() {
    DescribedPredicate<JavaCall<?>> constructorCallOfThirdPartyClass =
        target(is(constructor())).and(targetOwner(is(assignableTo(ThirdPartyClassWithProblem.class))));

    DescribedPredicate<JavaCall<?>> notFromWithinThirdPartyClass =
        originOwner(is(not(assignableTo(ThirdPartyClassWithProblem.class)))).forSubType();

    DescribedPredicate<JavaCall<?>> notFromWorkaroundFactory =
        originOwner(is(not(equivalentTo(ThirdPartyClassWorkaroundFactory.class)))).forSubType();

    DescribedPredicate<JavaCall<?>> targetIsIllegalConstructorOfThirdPartyClass =
        constructorCallOfThirdPartyClass.
            and(notFromWithinThirdPartyClass).
            and(notFromWorkaroundFactory);

    return never(callCodeUnitWhere(targetIsIllegalConstructorOfThirdPartyClass));
}
```

governance

# NetArchTest

<https://github.com/BenMorris/NetArchTest/>

```
// Controllers should not directly reference repositories
var result = Types.InCurrentDomain()
    .That()
    .ResideInNamespace("NetArchTest.SampleLibrary.Presentation")
    .ShouldNot()
    .HaveDependencyOn("NetArchTest.SampleLibrary.Data")
    .GetResult().IsSuccessful;
```

# NetArchTest

<https://github.com/BenMorris/NetArchTest/>

```
// Only classes in the data namespace can have a dependency on System.Data
result = Types.InCurrentDomain()
    .That().HaveDependencyOn("System.Data")
    .And().ResideInNamespace(("ArchTest"))
    .Should().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .GetResult().IsSuccessful;
```

# NetArchTest

<https://github.com/BenMorris/NetArchTest/>

```
// All the classes in the data namespace should implement IRepository
result = Types.InCurrentDomain()
    .That().ResideInNamespace("NetArchTest.SampleLibrary.Data")
    .And().AreClasses()
    .Should().ImplementInterface(typeof(IRepository<>))
    .GetResult().IsSuccessful;

// Classes that implement IRepository should have the suffix "Repository"
result = Types.InCurrentDomain()
    .That().ResideInNamespace("NetArchTest.SampleLibrary.Data")
    .And().AreClasses()
    .Should().HaveNameEndingWith("Repository")
    .GetResult().IsSuccessful;

// Classes that implement IRepository must reside in the Data namespace
result = Types.InCurrentDomain()
    .That().ImplementInterface(typeof(IRepository<>))
    .Should().ResideInNamespace("NetArchTest.SampleLibrary.Data")
    .GetResult().IsSuccessful;

// All the service classes should be sealed
result = Types.InCurrentDomain()
    .That().ImplementInterface(typeof(IWidgetService))
    .Should().BeSealed()
    .GetResult().IsSuccessful;
```

# Legality of Open Source Libraries



Penultima ↑ e  


# Legality of Open Source Libraries



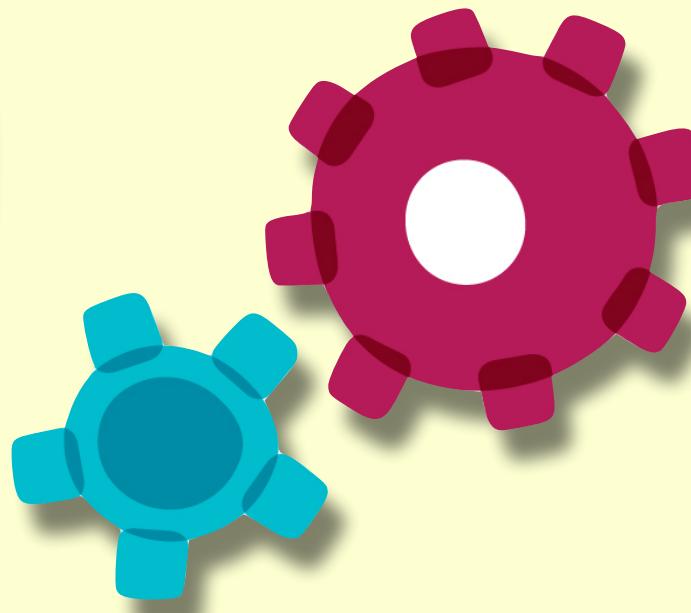
Penultima ↑e  
A row of three small, semi-transparent gears in pink, teal, and light blue colors, positioned below the text.

# Legality of Open Source Libraries



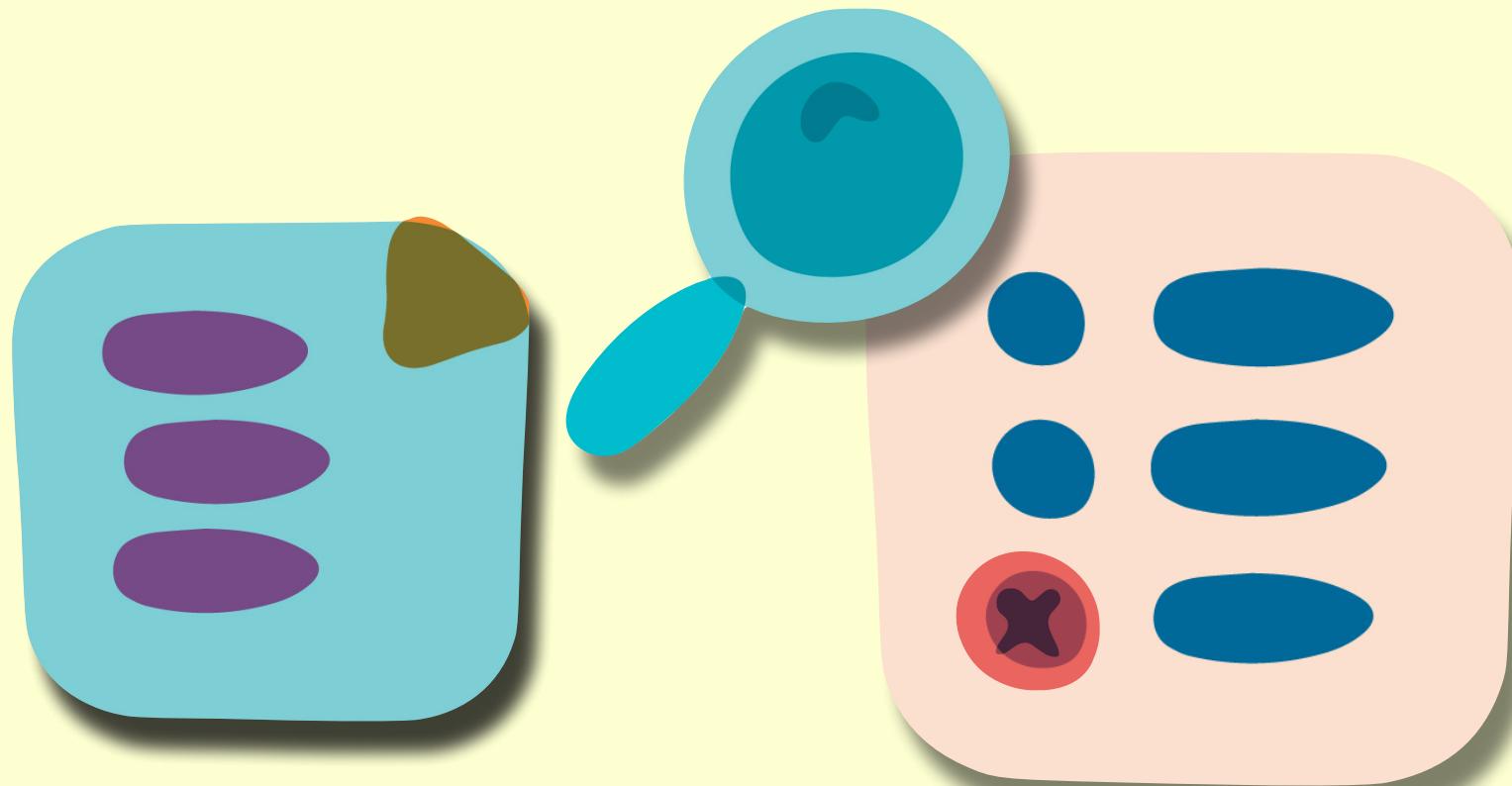
Penultima ↑ e  


# Legality of Open Source Libraries



Penultima ↑ e  


# Legality of Open Source Libraries



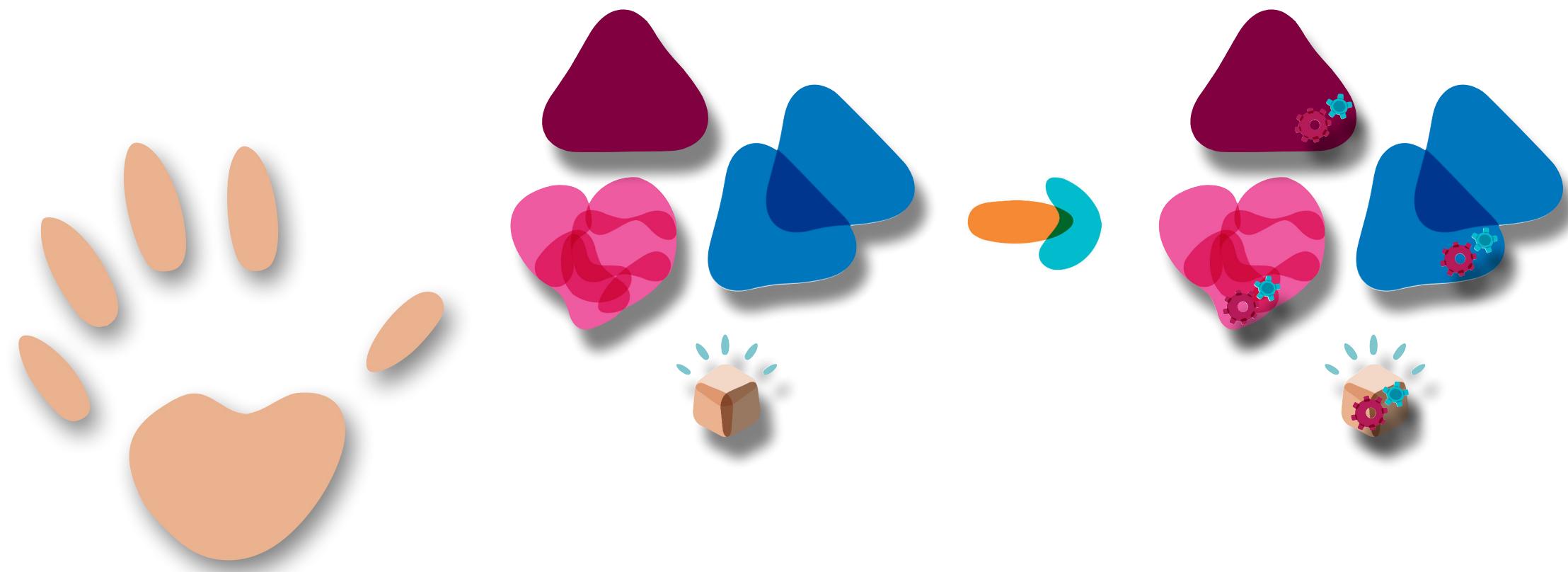
# Penultima

# Legality of Open Source Libraries



Penultima ↑e  


## 2. Identify Fitness Functions





# fitness function:

## No More View Models

PenultimateWidgets has a group of developers who cut their teeth on mainframe programming, where all work is done via a terminal. They have (mostly) switched to modern web development, but architects find that they often create models and workflows directly in web pages rather than support a cleaner MVC separation, and they would like to discourage this behavior.



# fitness function:

## No More View Models

- use PMD (<https://pmd.github.io/>) or Language server protocol (<https://langserver.org/>) to build a tool that scans which classes developers instantiate in view code
  - build a list of restricted model and controller packages
  - if any view code instantiates a class from the restricted list, fail the build

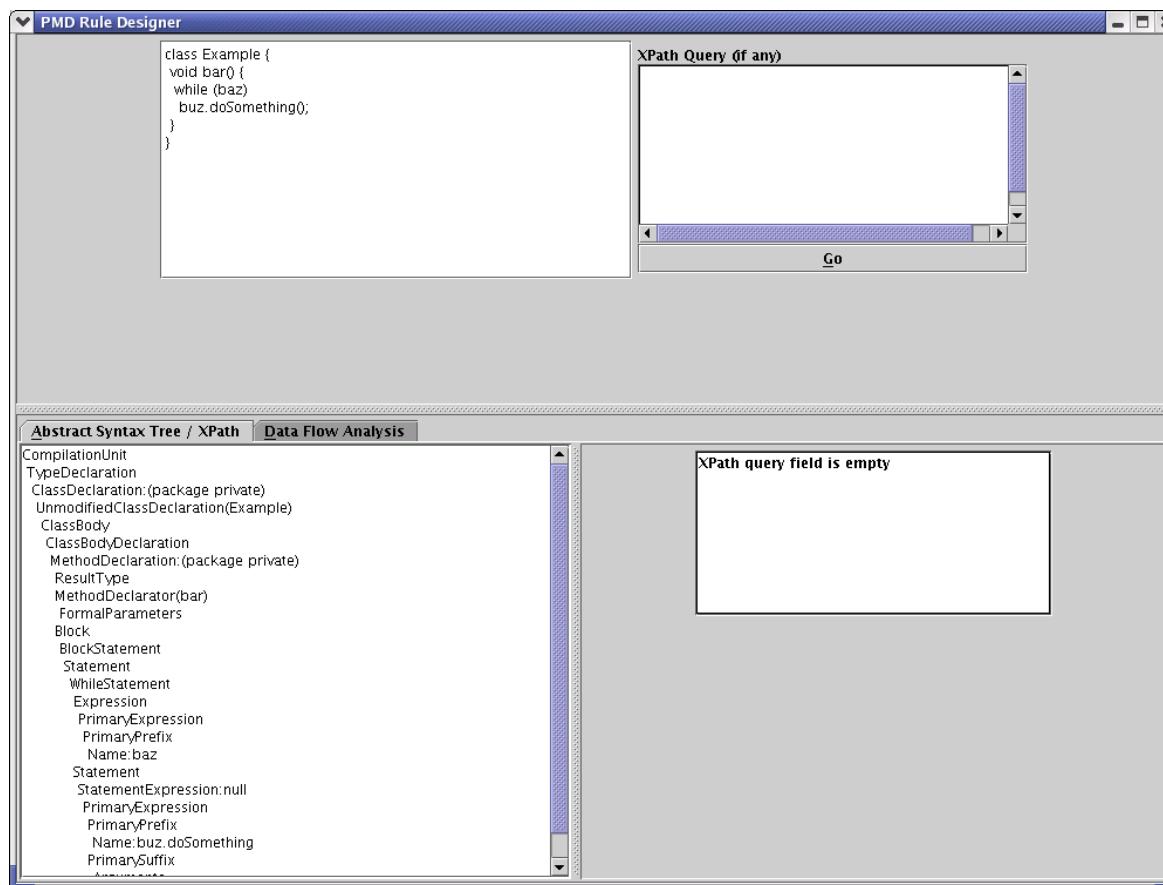
# “Parser” Fitness Function

- 1. Full-blown testing library
- 2. Parser
- 3. Lexer

# “Parser” Fitness Function



<https://pmd.github.io/>



# “Parser” Fitness Function



<https://pmd.github.io/>

```
import net.sourceforge.pmd.lang.ast.*;
import net.sourceforge.pmd.lang.java.ast.*;
import net.sourceforge.pmd.lang.java.rule.*;

public class WhileLoopsMustUseBracesRule extends AbstractJavaRule {
    public Object visit(ASTWhileStatement node, Object data) {
        Node firstStmt = node.jjtGetChild(1);
        if (!hasBlockAsFirstChild(firstStmt)) {
            addViolation(data, node);
        }
        return super.visit(node,data);
    }
    private boolean hasBlockAsFirstChild(Node node) {
        return (node.jjtGetNumChildren() != 0 && (node.jjtGetChild(0) instanceof ASTBlock));
    }
}
```

# “Parser” Fitness Function

- 1. Full-blown testing library



<https://pmd.github.io/>

- 2. Parser
- 3. Lexer

# “Parser” Fitness Function

- 1. Full-blown testing library



- 2. Parser



- 3. Lexer



<https://pmd.github.io/>

# “Parser” Fitness Function

- 1. Full-blown testing library

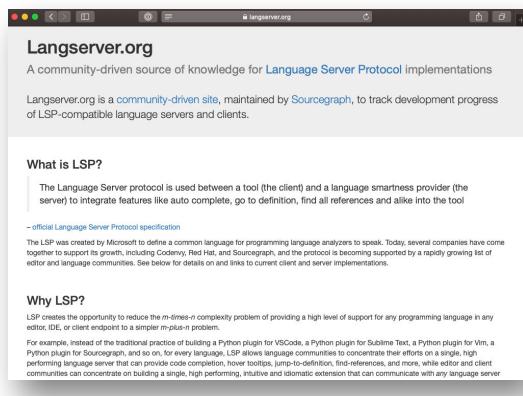


- 2. Parser

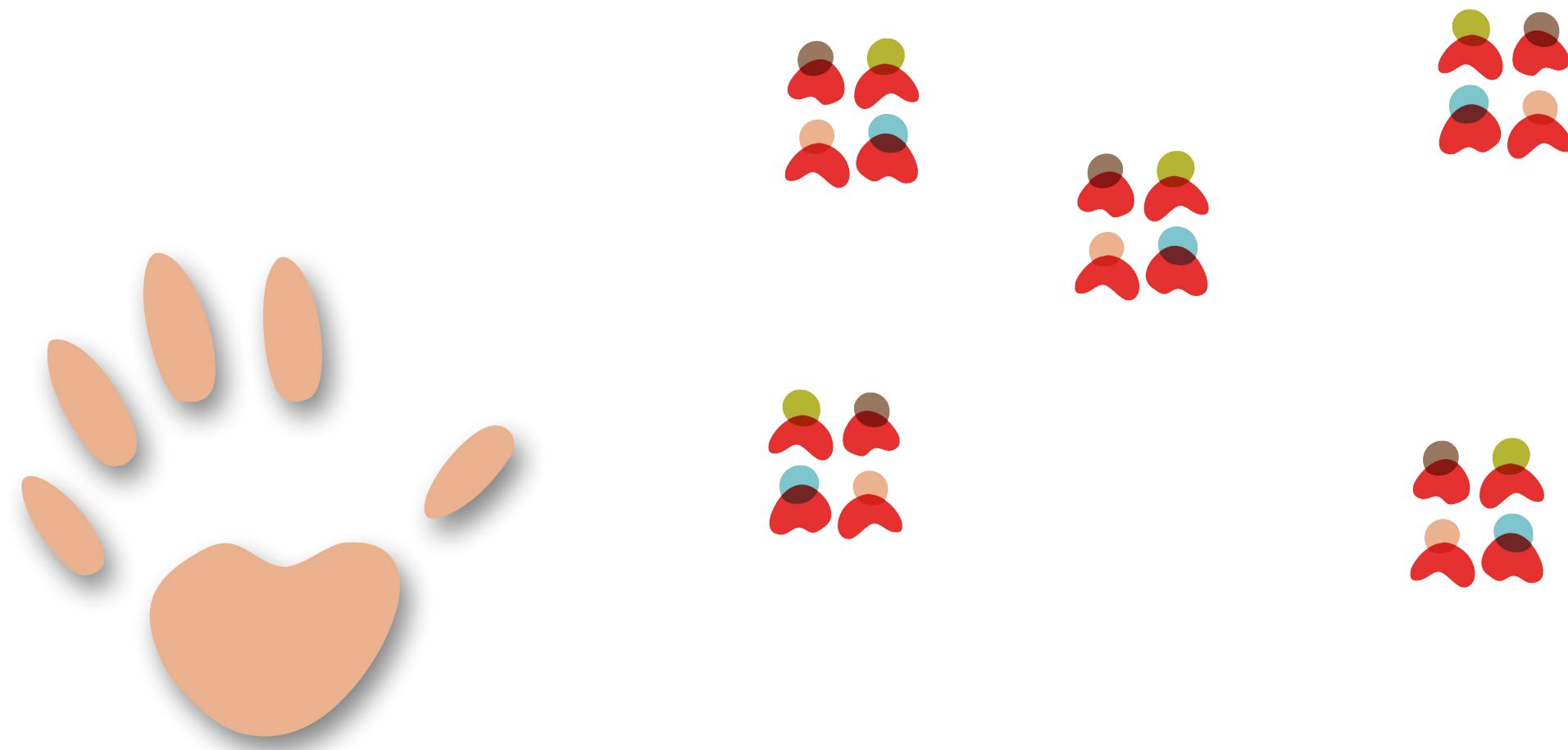


<https://pmd.github.io/>

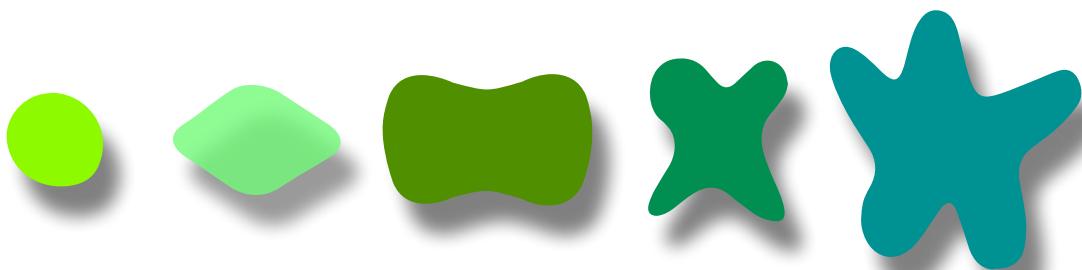
- 3. Lexer



# 3. Fitness Function Katas

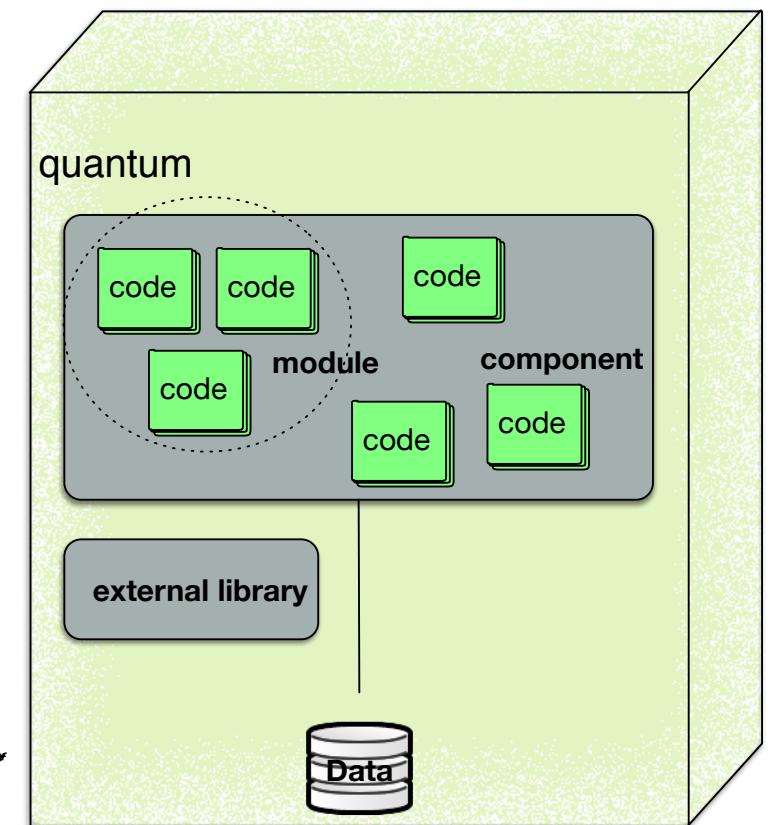


# Two Big Ideas

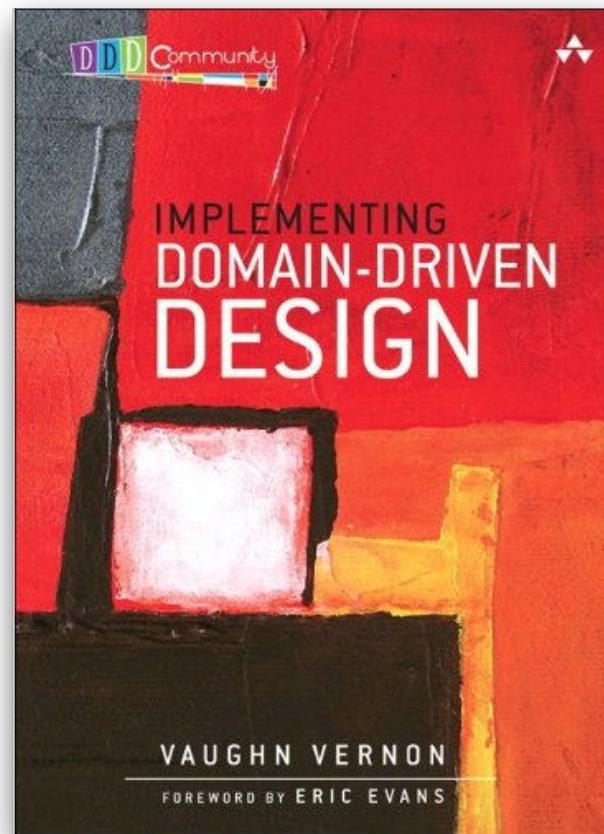
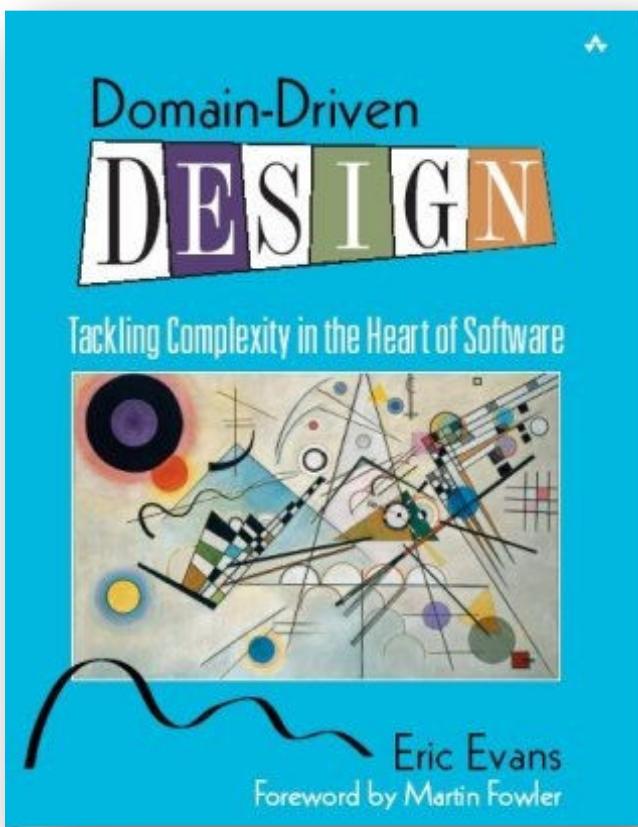


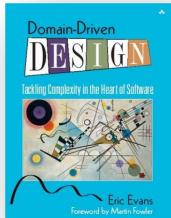
fitness functions for  
evolvability

architectural  
quantum



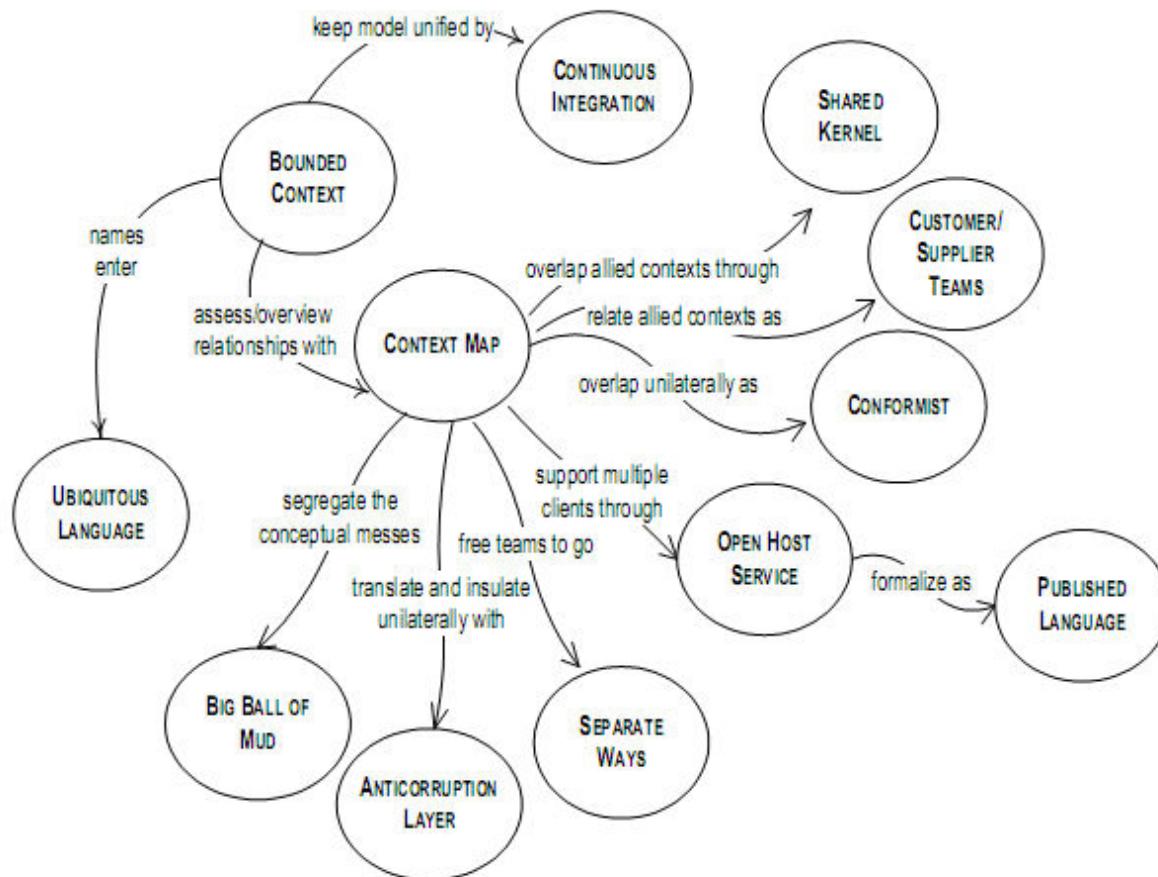
# Domain Driven Design





# Bounded Context

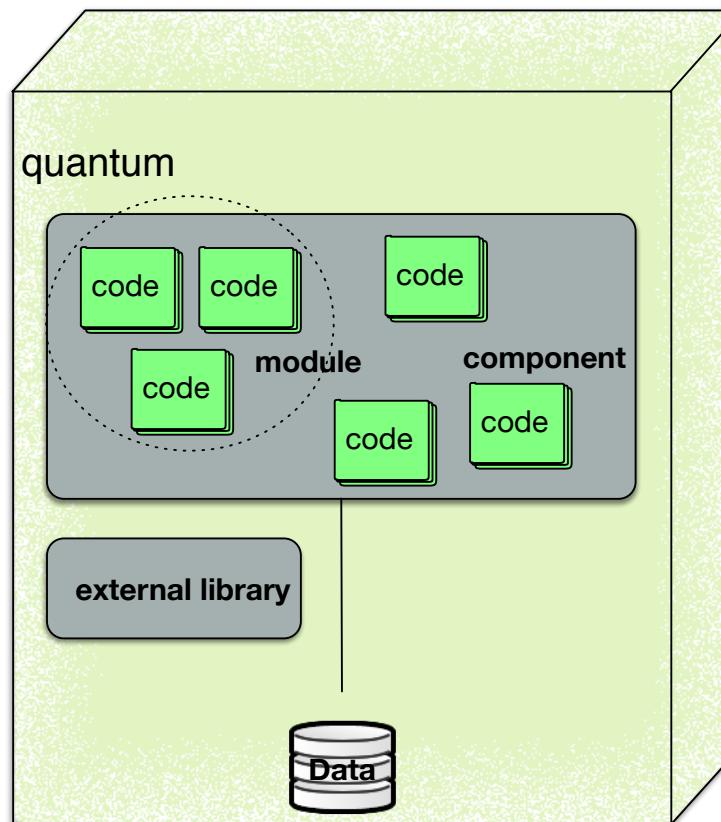
## Maintaining Model Integrity



# Architectural Quantum

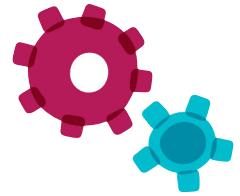
An architectural quantum is an independently deployable component with synchronous cohesion, which includes all the structural elements required for the system to function properly.

# Architectural Quantum



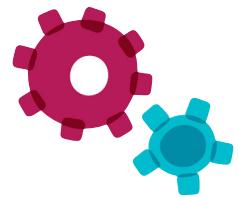
# Why Quantum?

# Why Quantum?

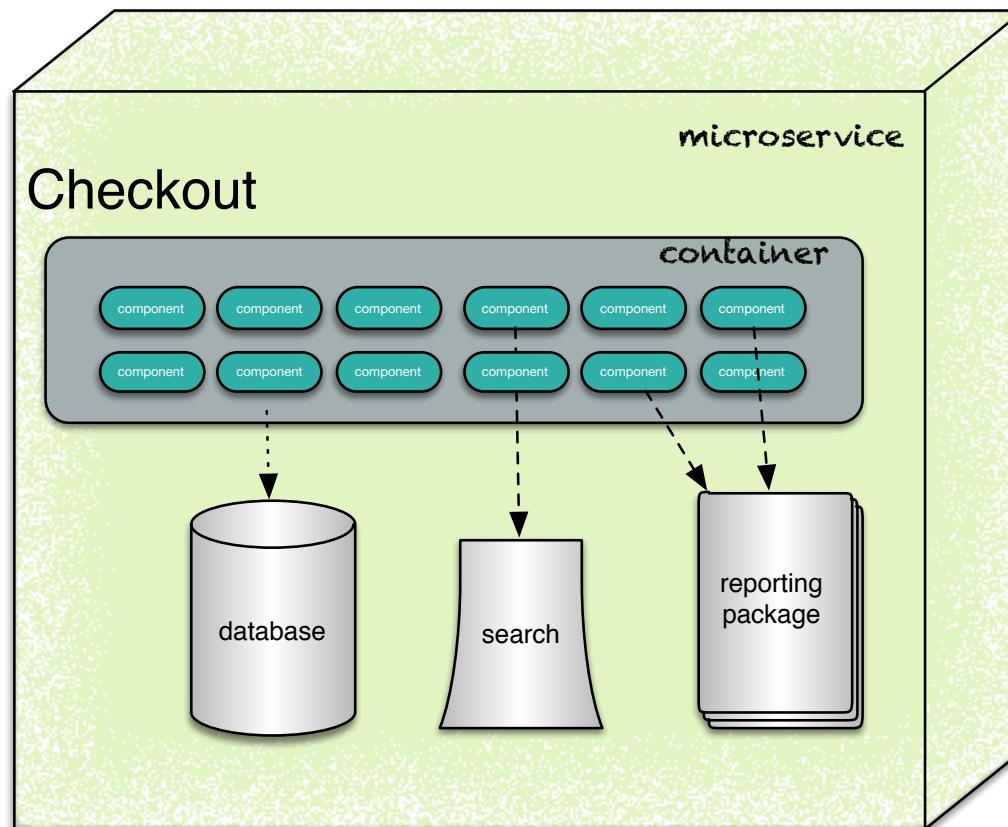


operational view  
of architecture

# Why Quantum?



operational view  
of architecture



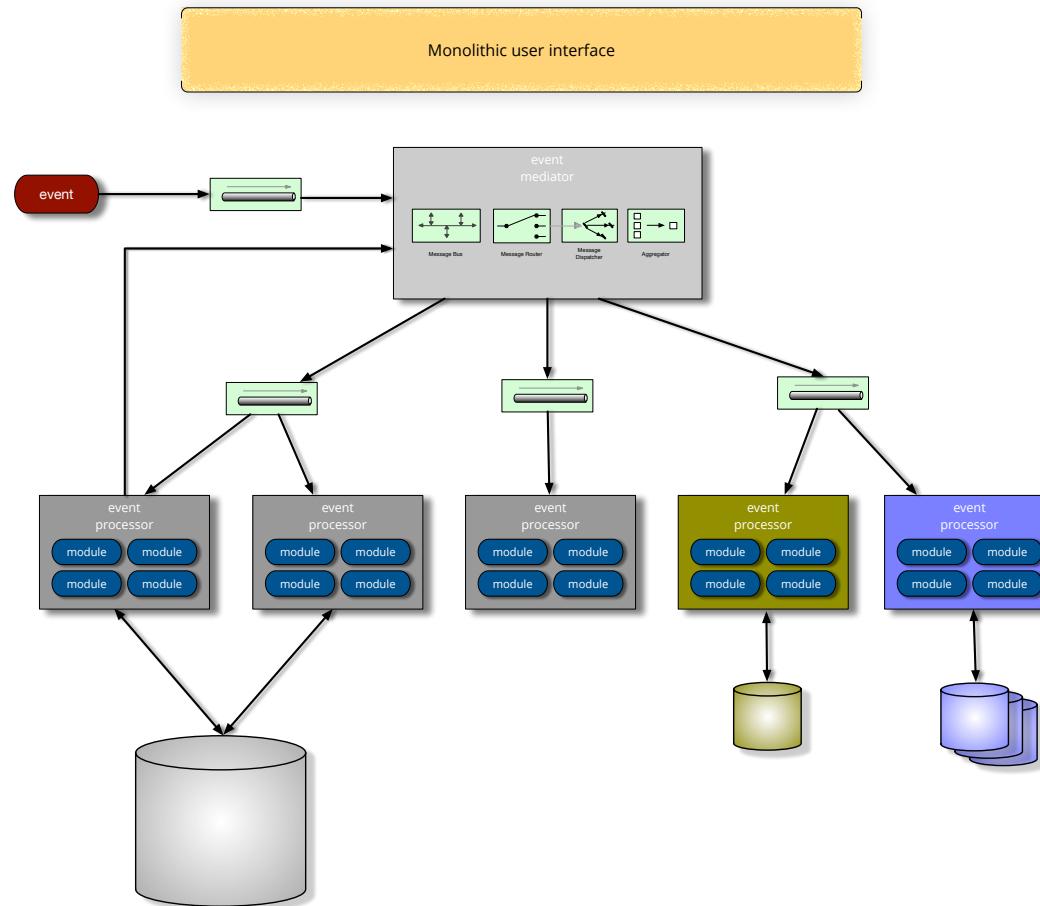
# Why Quantum?

holistic



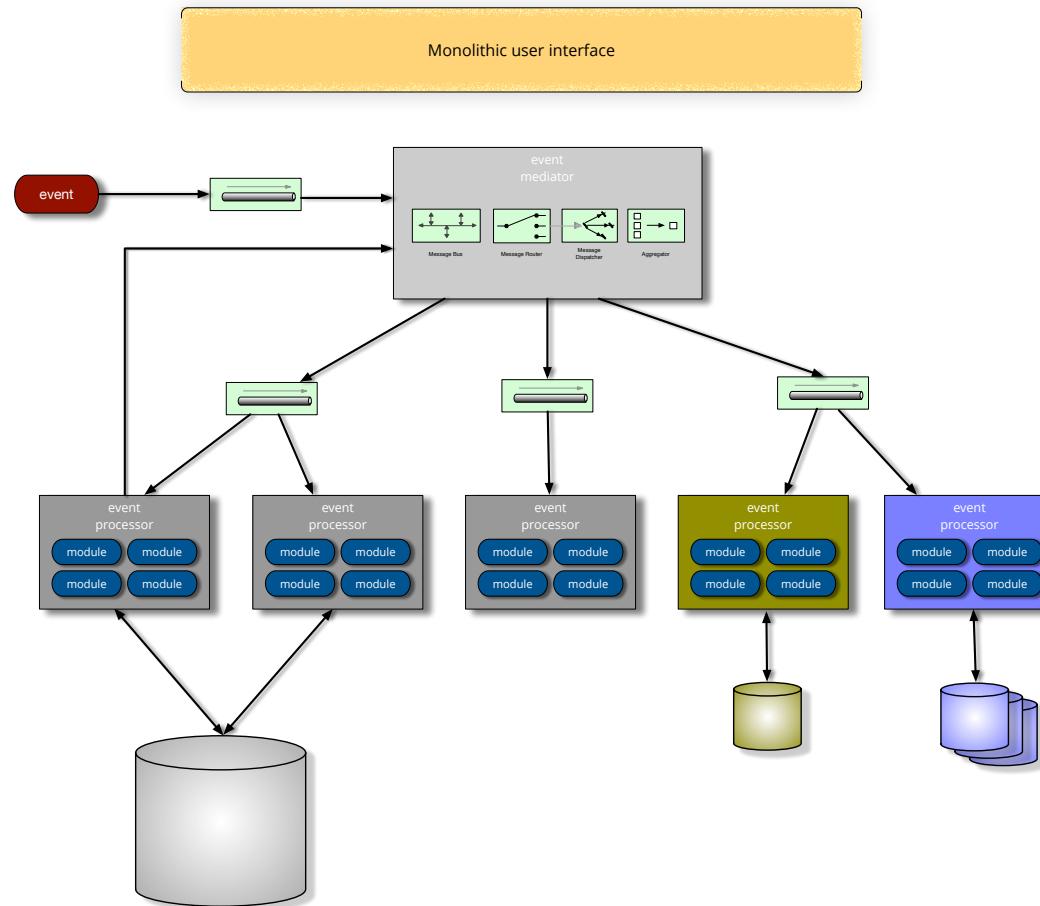
holistic

# Why Quantum?



holistic

# Why Quantum?



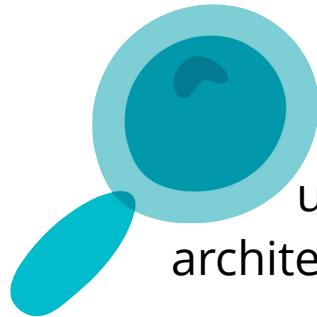
*"multiple dimensions"*



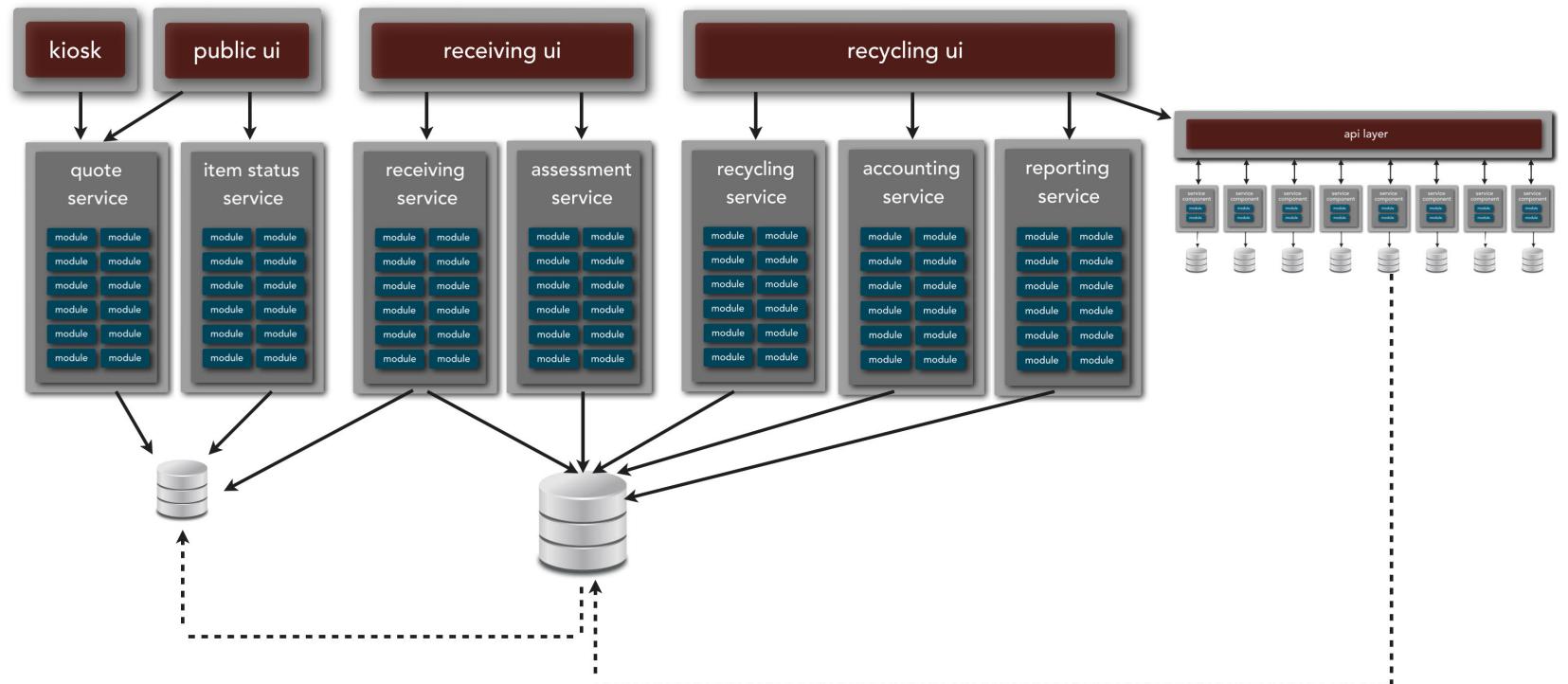
useful for  
architectural analysis

# Why Quantum?

# Why Quantum?



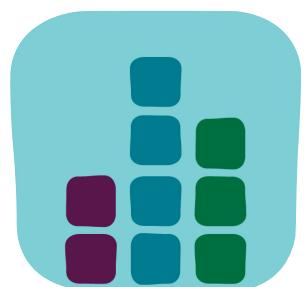
useful for  
architectural analysis



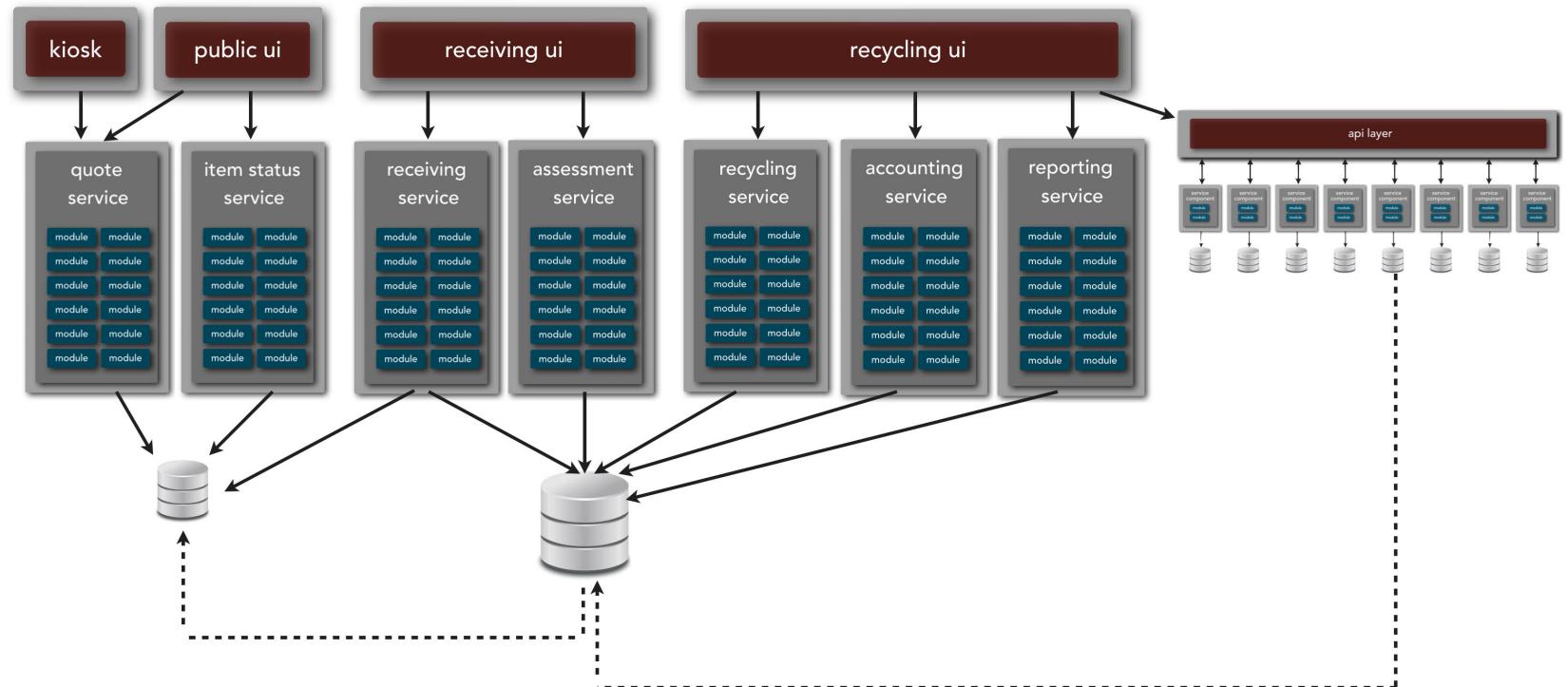
# Why Quantum?



useful for  
architectural analysis

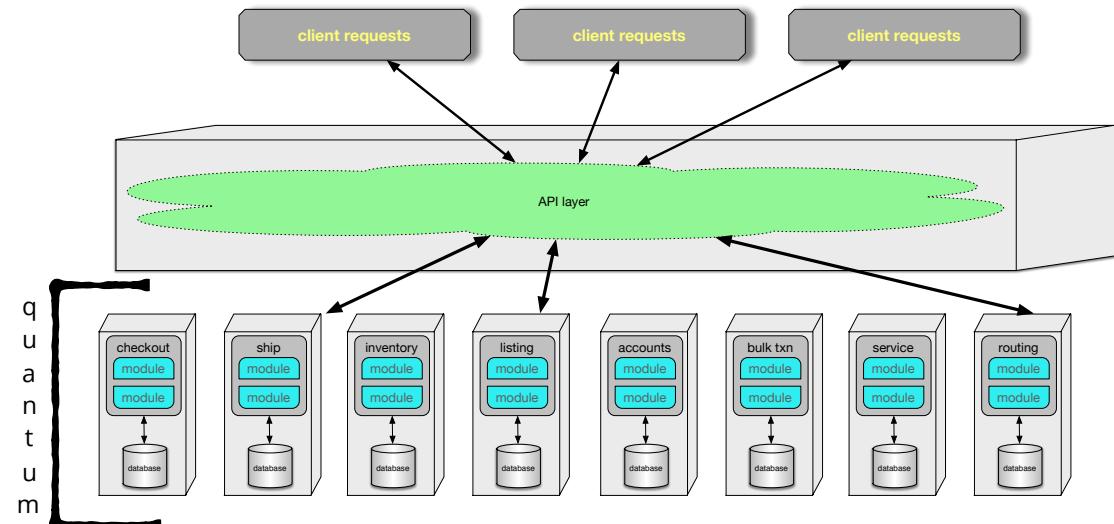
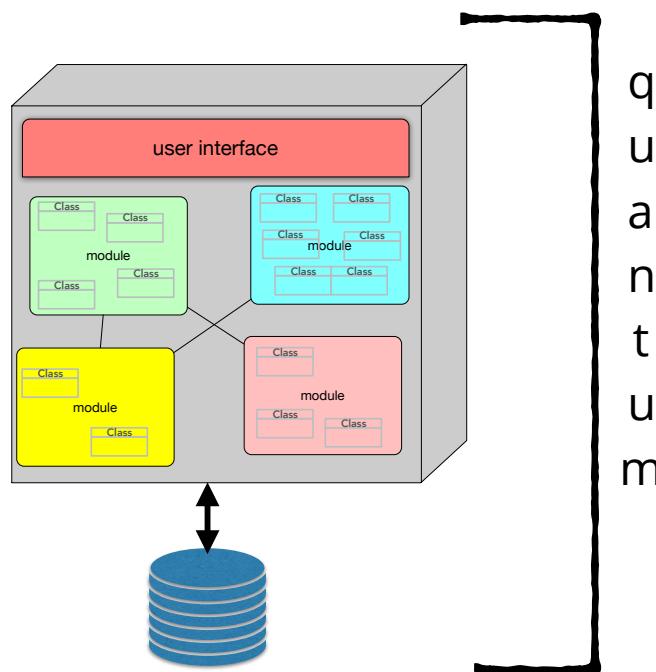


helps analyze  
coupling

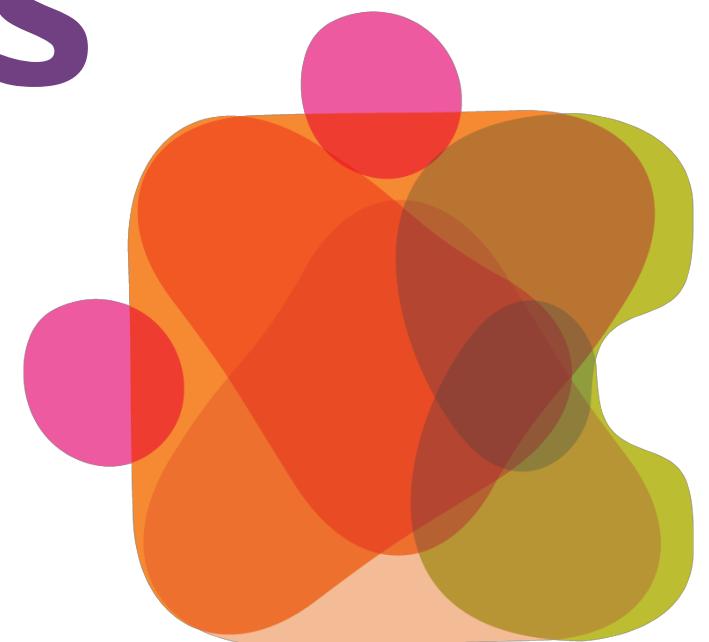


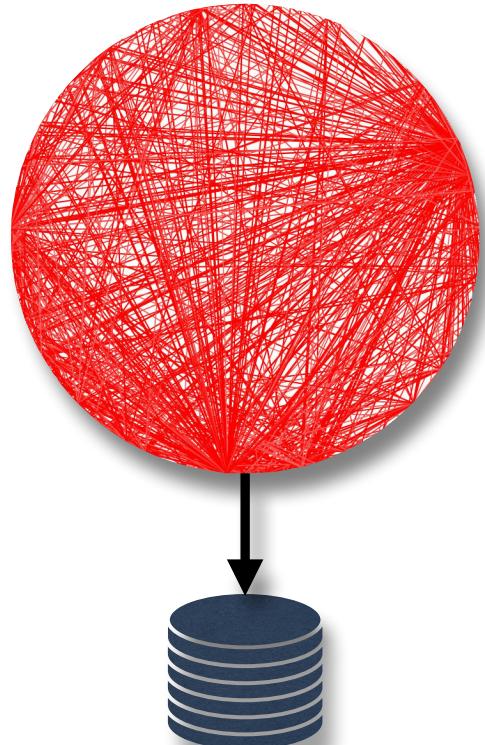
# Why Quantum?

*The quantum is where architectural characteristics live.*



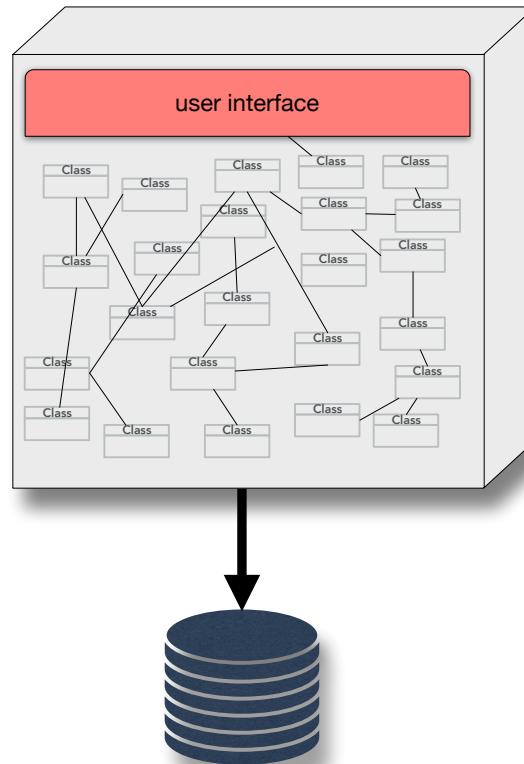
# Monoliths





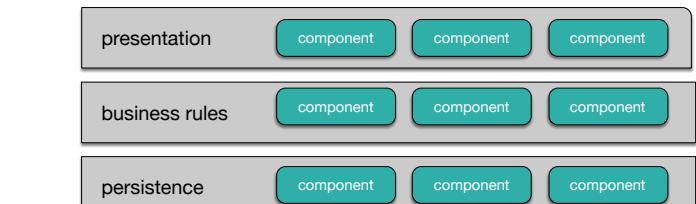
“Big Ball of Mud”

[https://en.wikipedia.org/wiki/Big\\_ball\\_of\\_mud](https://en.wikipedia.org/wiki/Big_ball_of_mud)

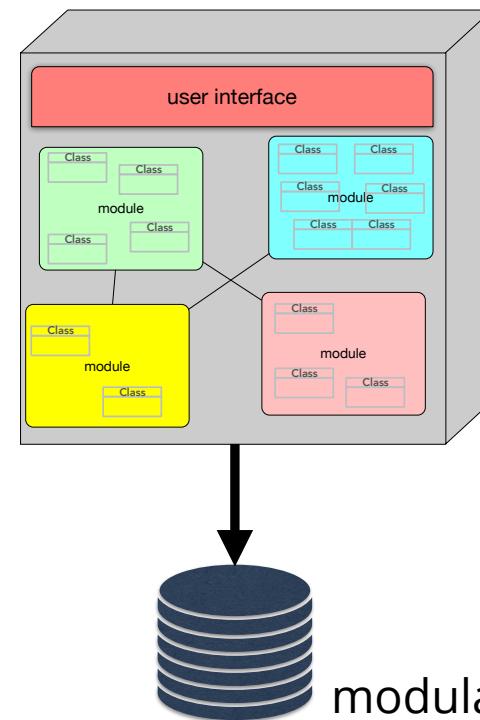


unstructured  
monolith

# Monoliths

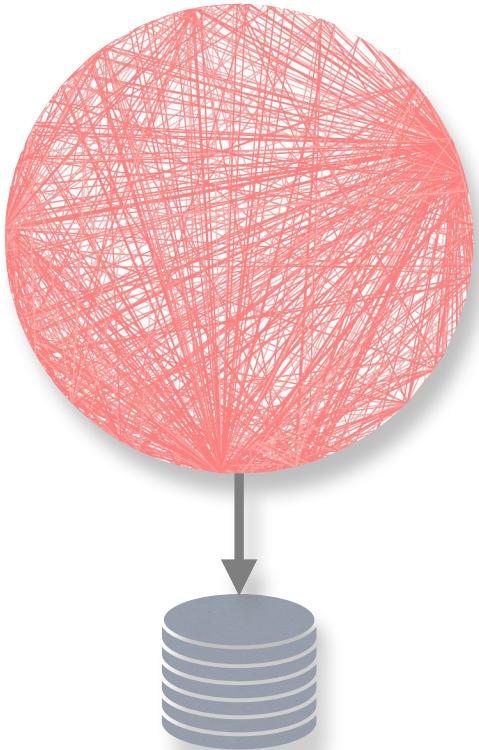


layered  
architecture



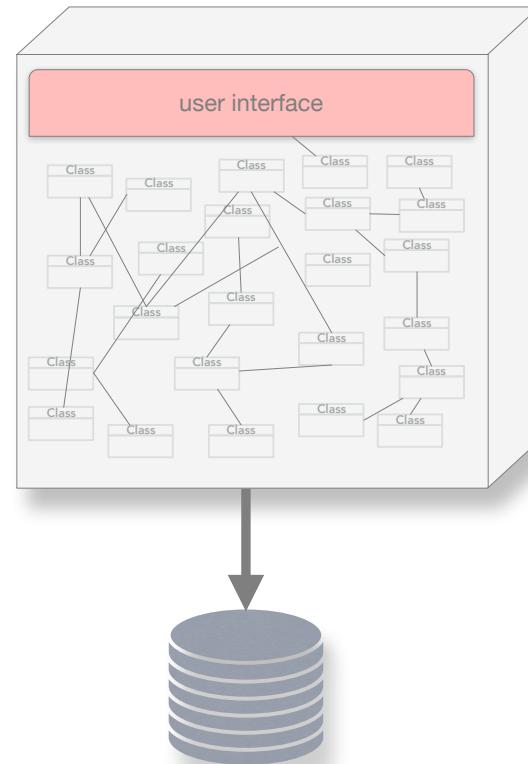
modular monolith

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



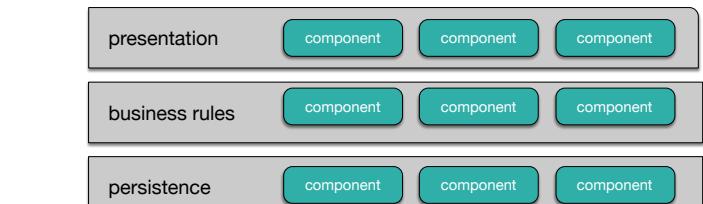
**“Big Ball of Mud”**

[https://en.wikipedia.org/wiki/Big\\_ball\\_of\\_mud](https://en.wikipedia.org/wiki/Big_ball_of_mud)

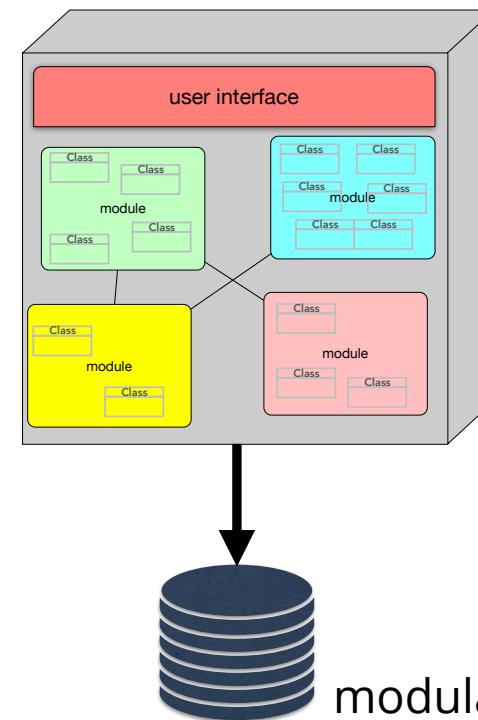


unstructured  
monolith

# Monoliths



layered  
architecture

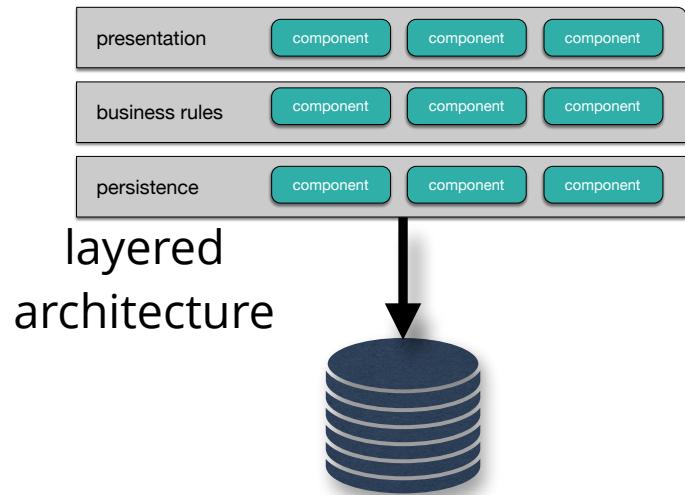


modular monolith

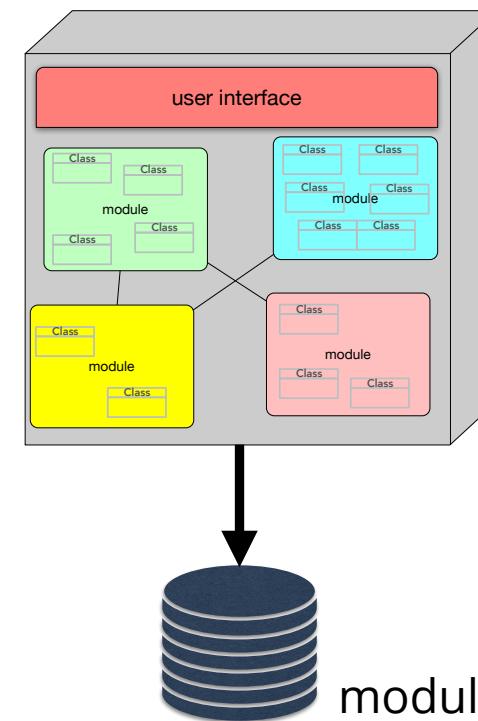
<http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths>

# Monoliths

technical partitioning

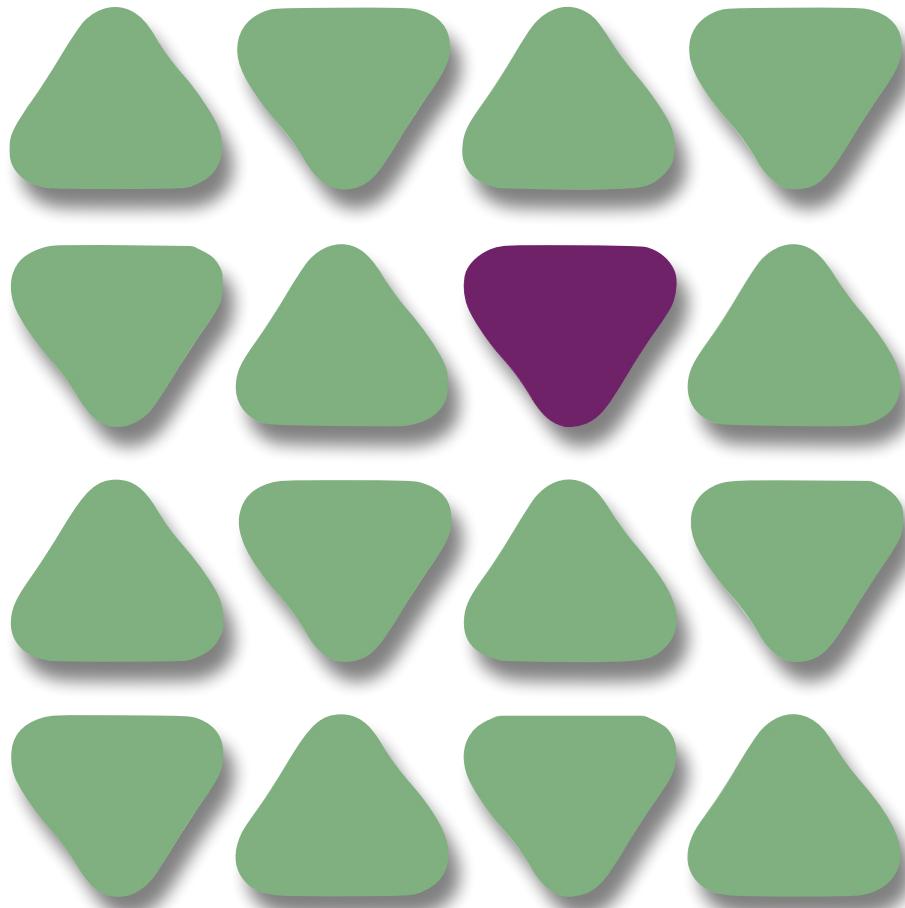


domain partitioning



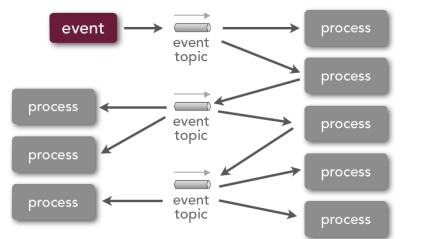
modular monolith

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

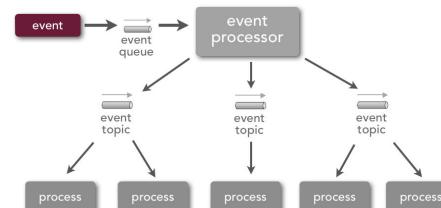


distributed systems

# Event-driven Architectures

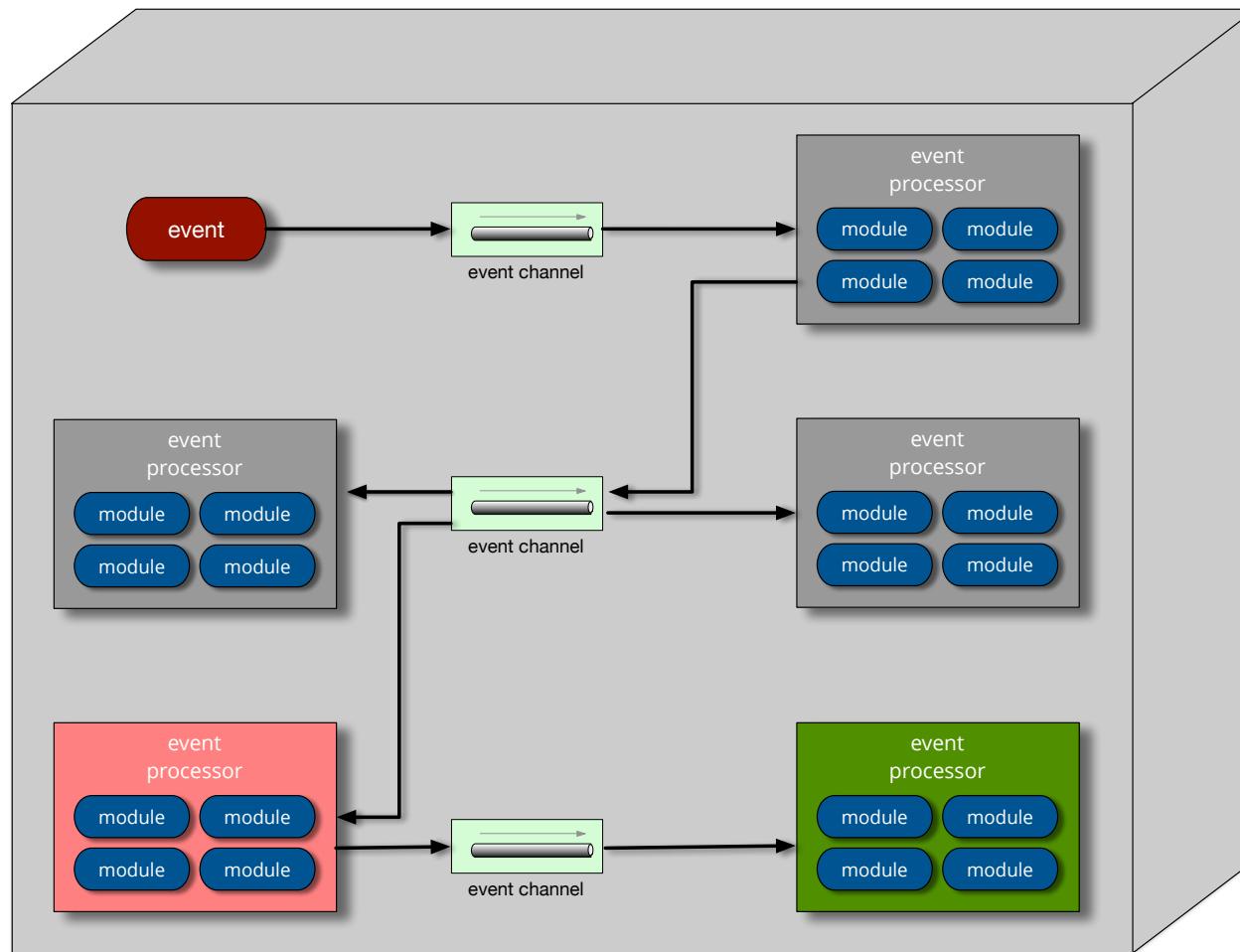


broker topology

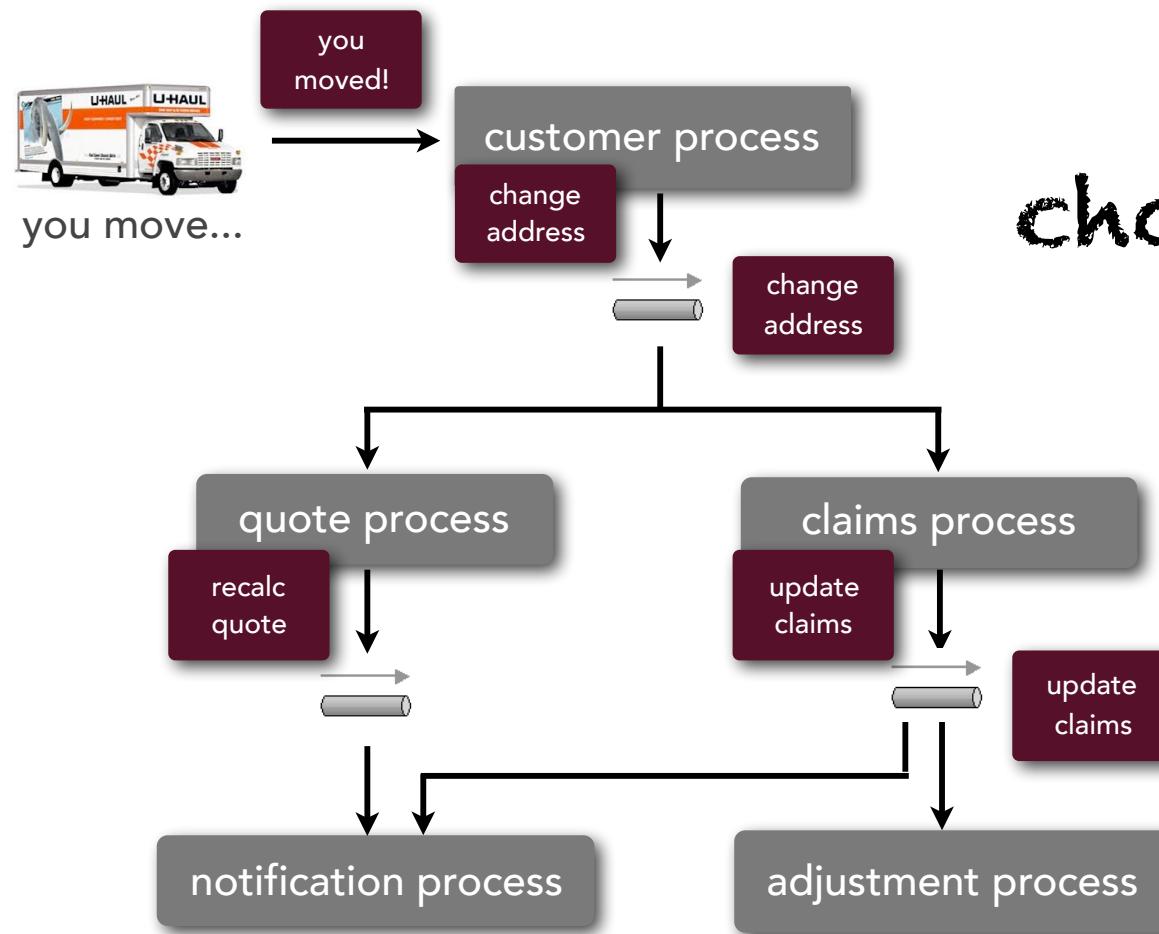


mediator topology

# Broker Base Architecture

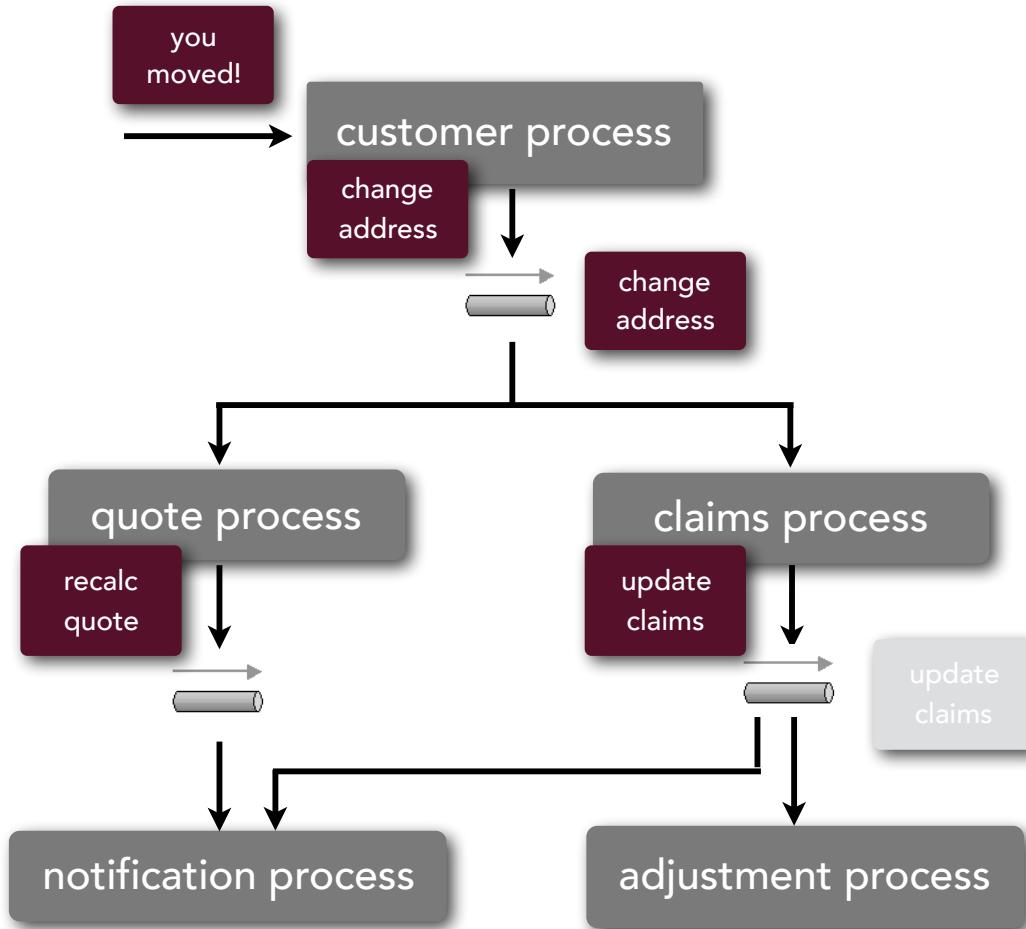


# Broker Message Passing



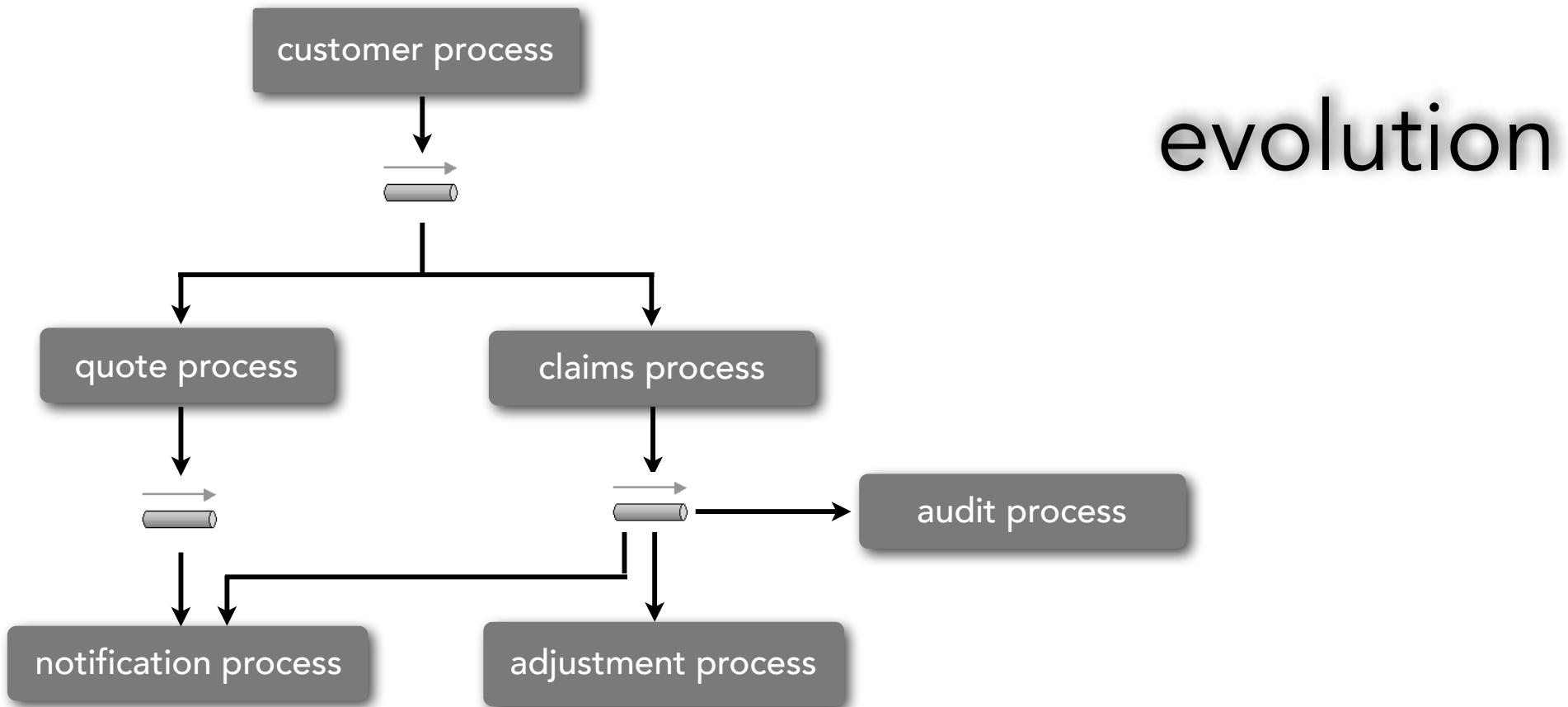
choreography

# Broker Message Passing

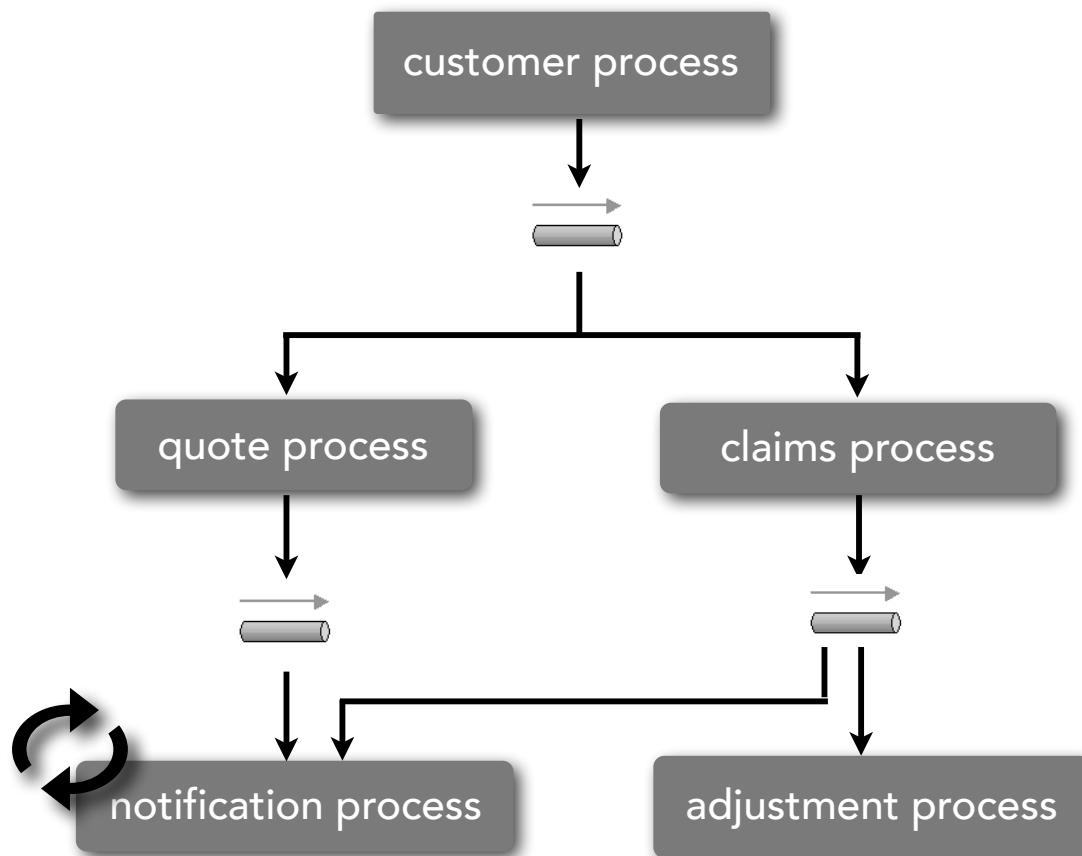


Error handling?

# Broker Message Passing



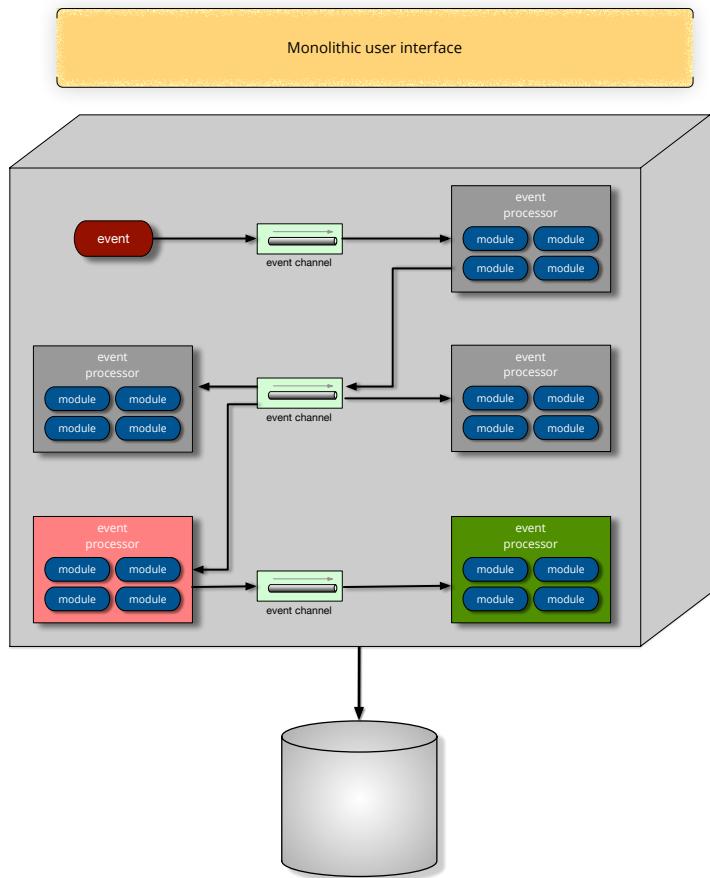
# Broker Message Passing



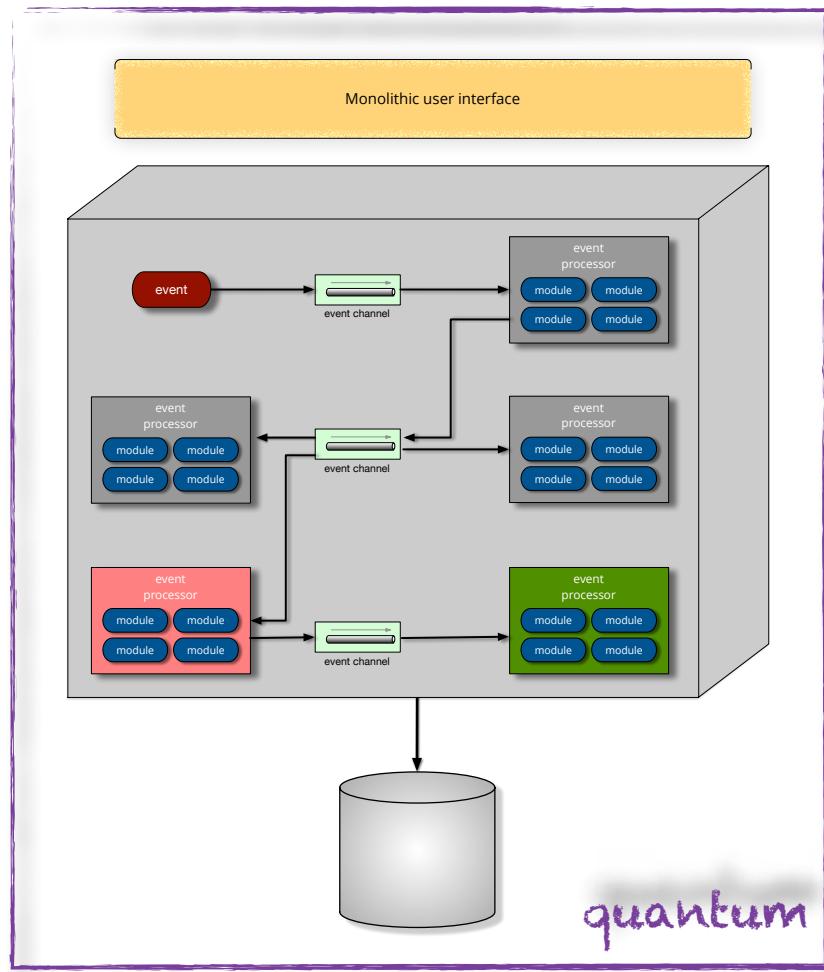
Unified notification  
message to customer?

In asynchronous systems,  
coordination happens at the end

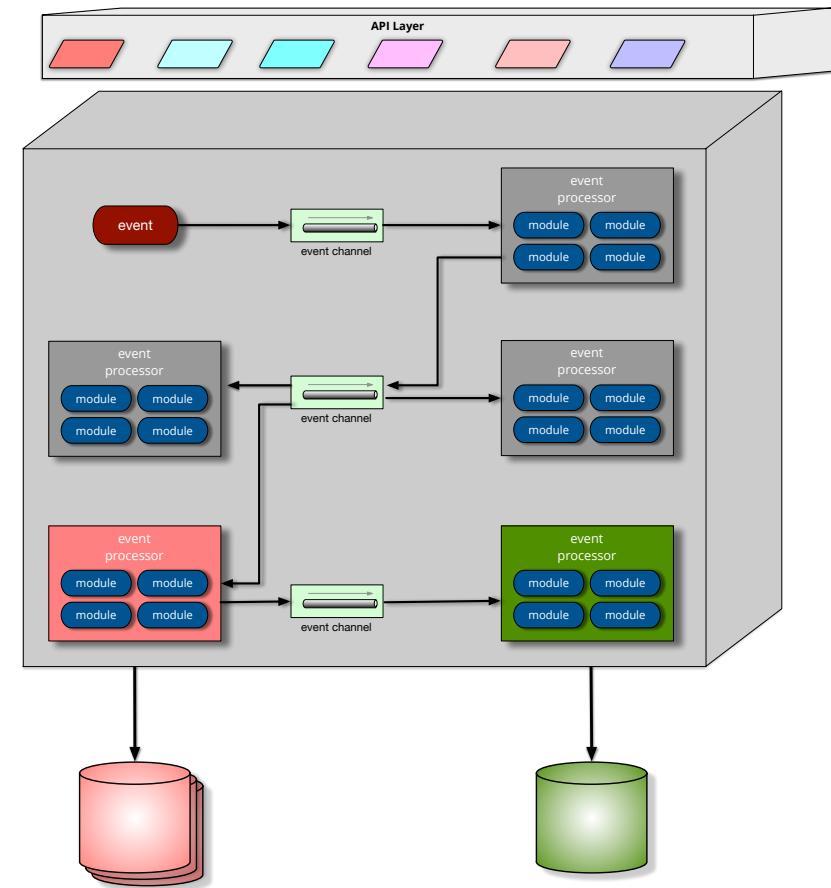
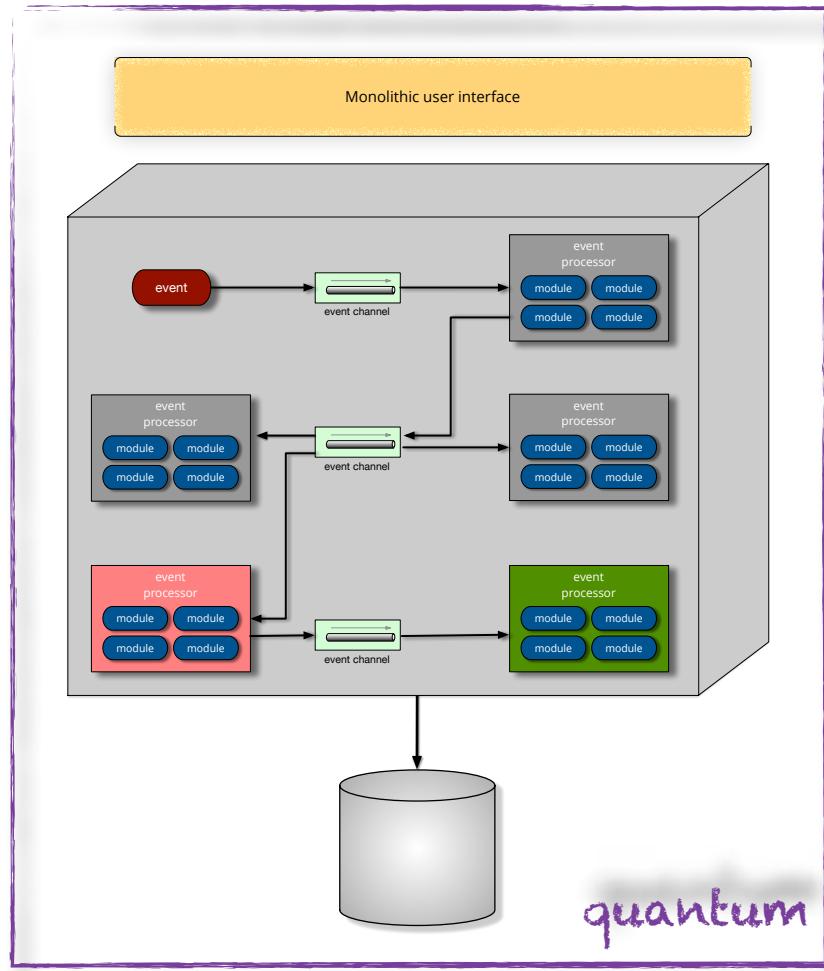
# Broker: Quanta



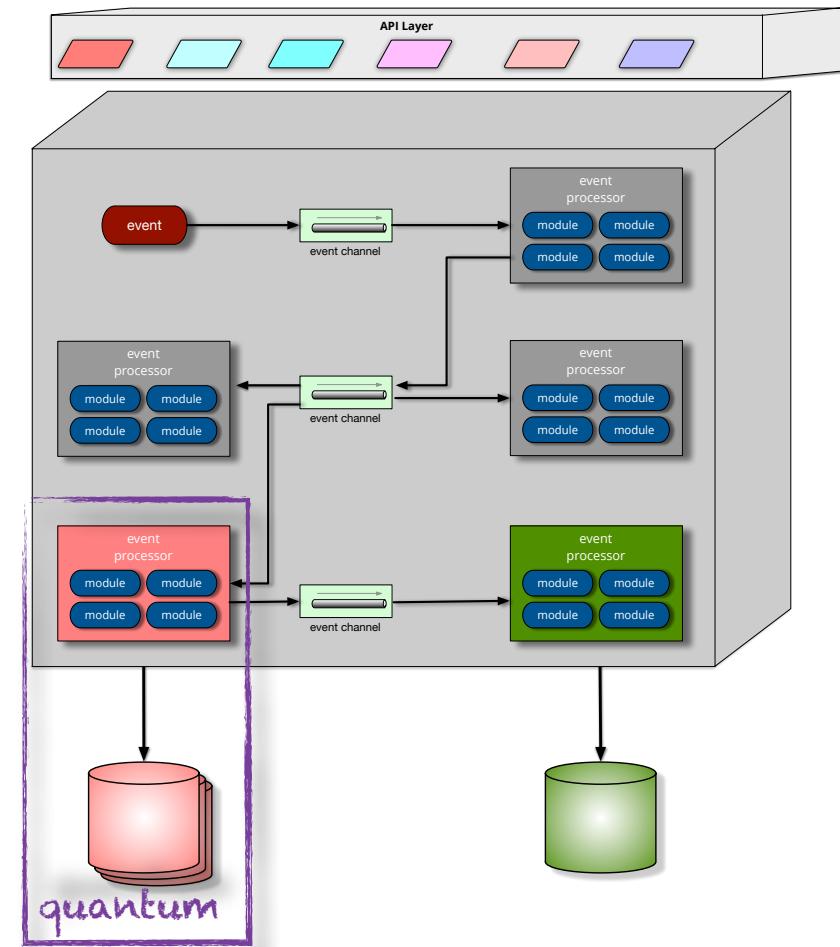
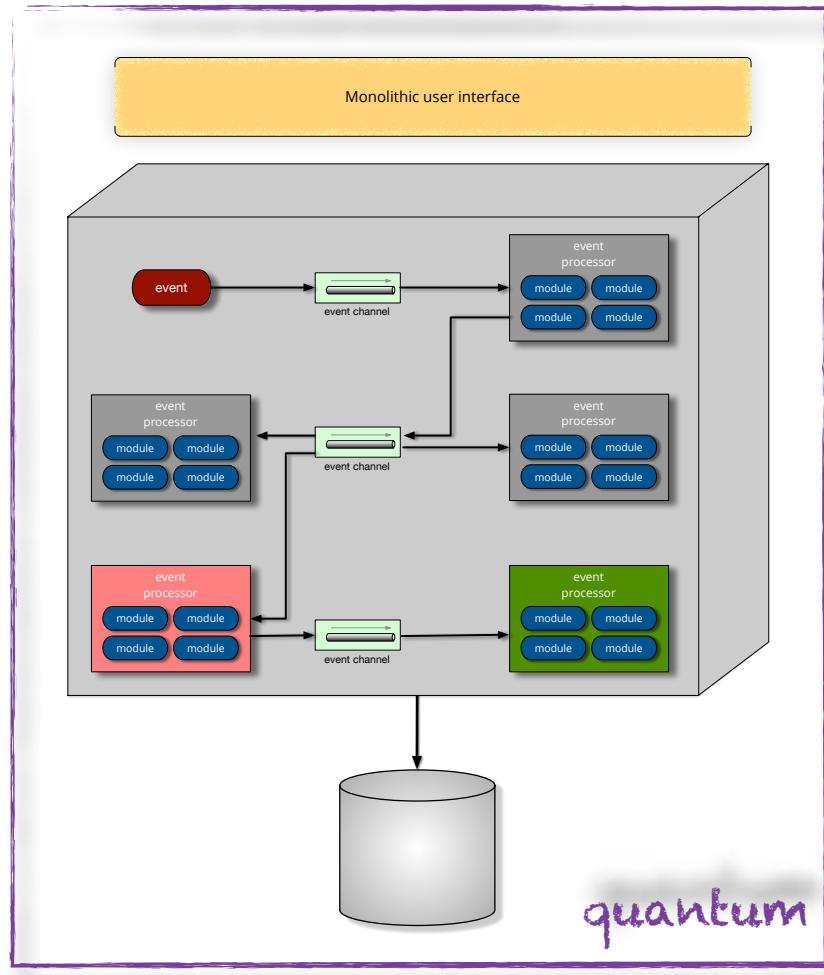
# Broker: Quanta



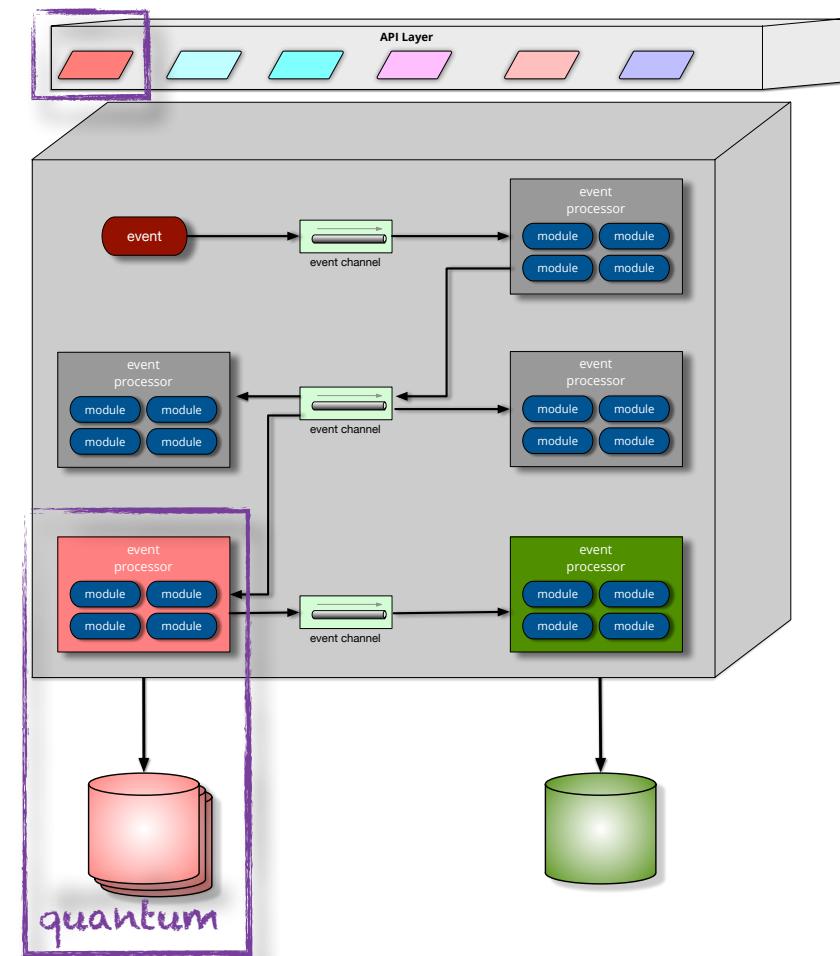
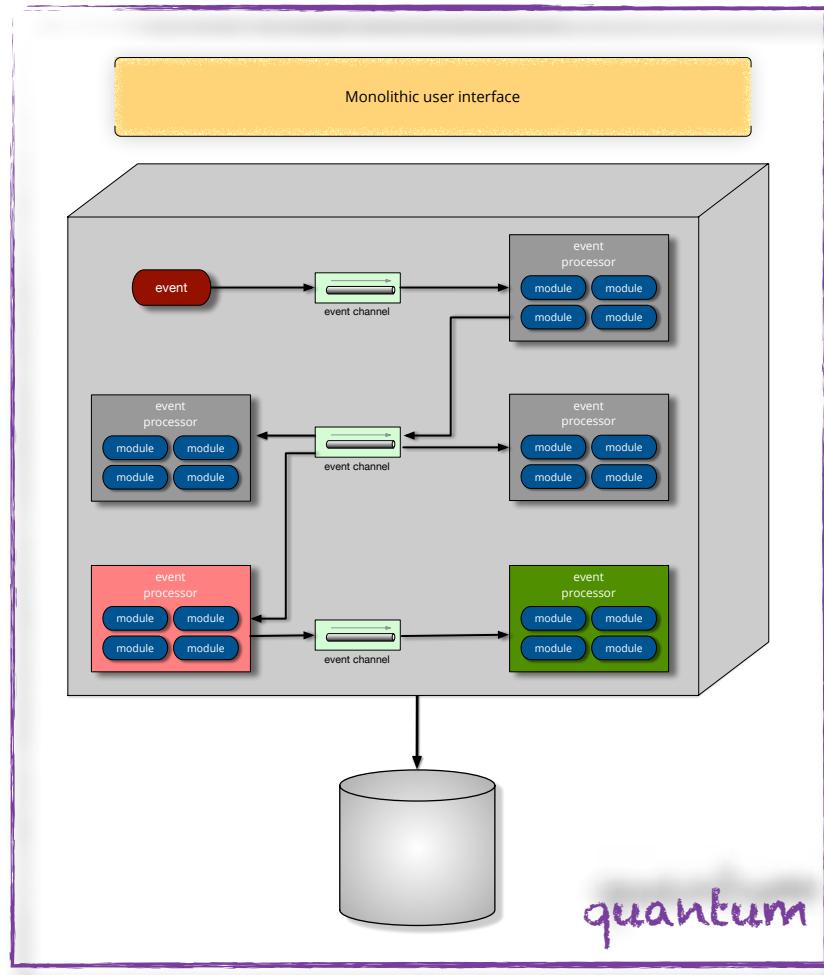
# Broker: Quanta



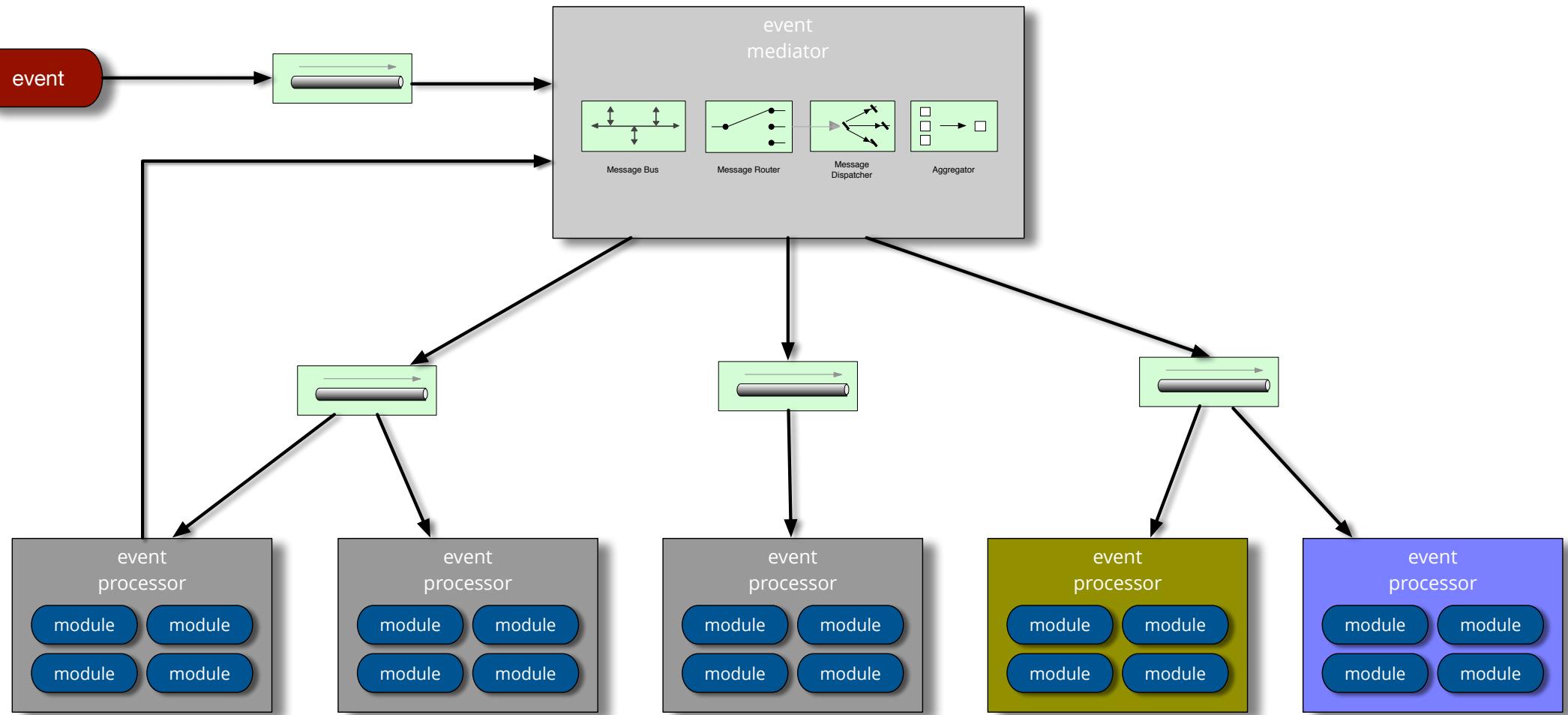
# Broker: Quanta



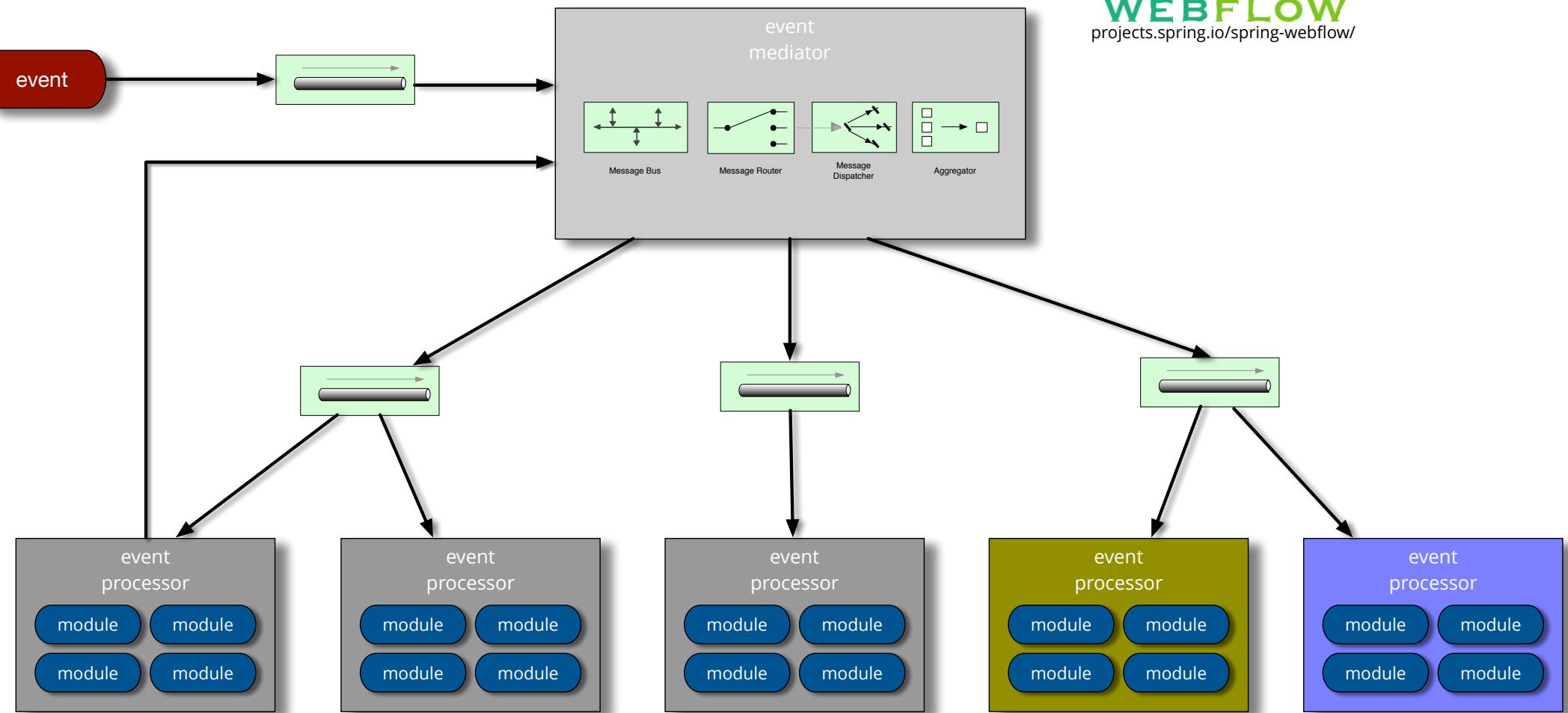
# Broker: Quanta



# Mediator EDA



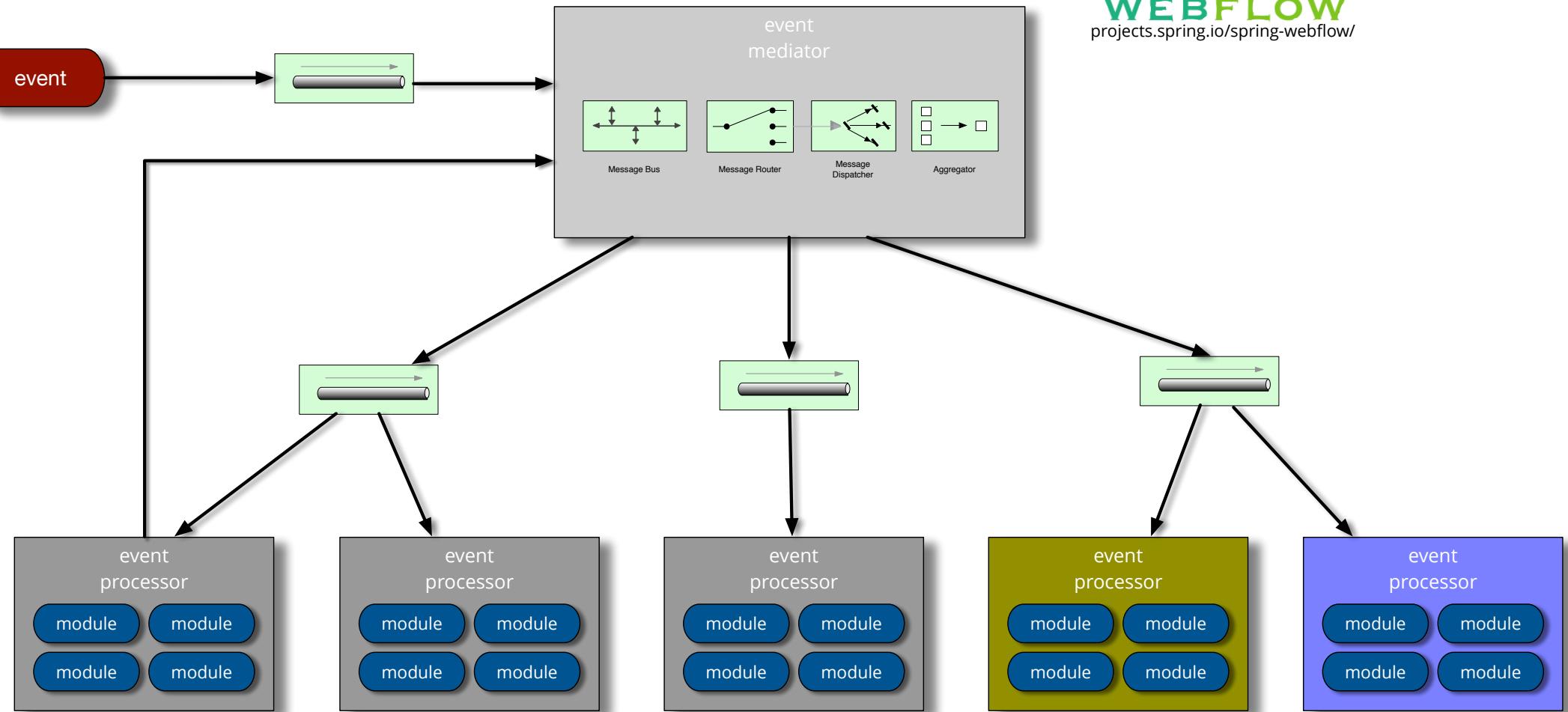
# Mediator EDA



# Mediator EDA



WEBFLOW  
[projects.spring.io/spring-webflow/](http://projects.spring.io/spring-webflow/)



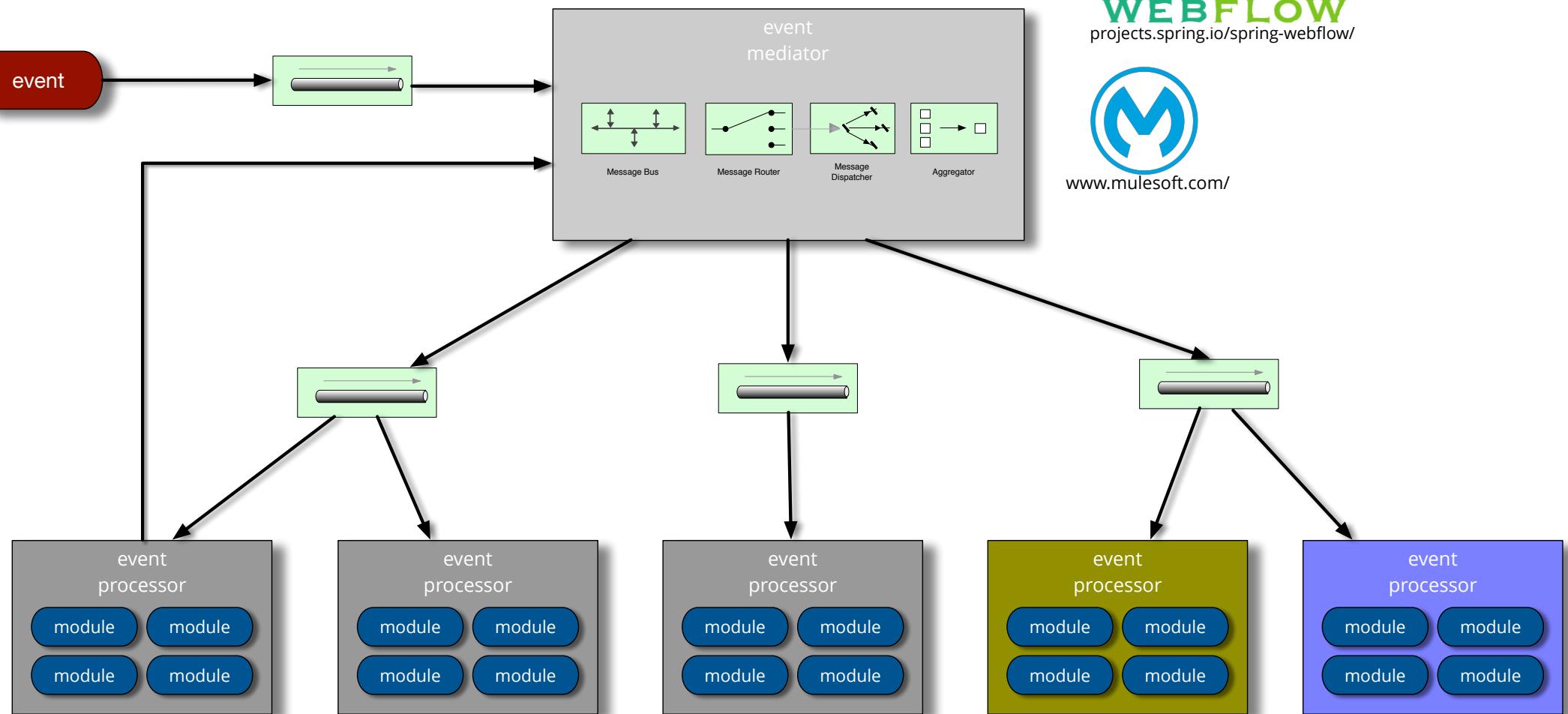
# Mediator EDA



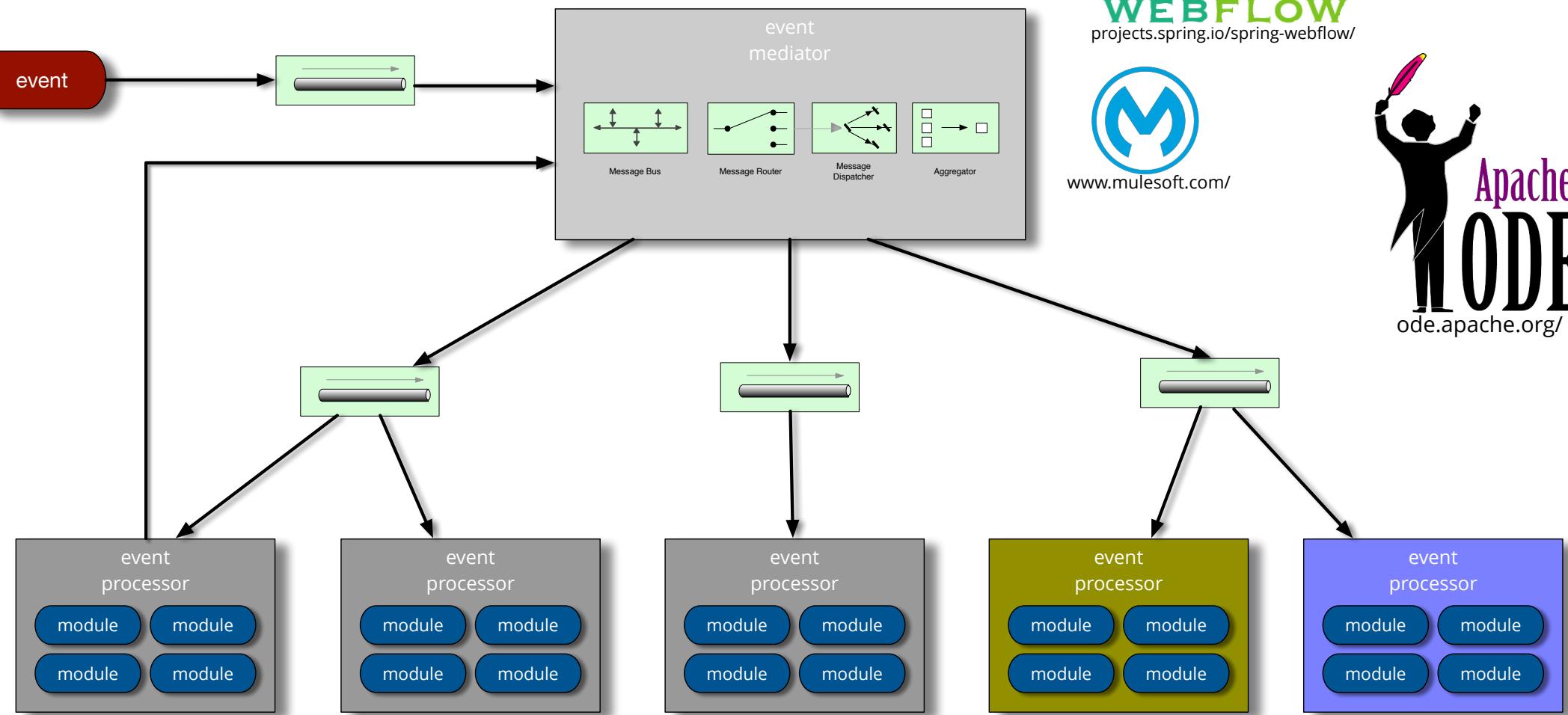
**WEBFLOW**  
[projects.spring.io/spring-webflow/](http://projects.spring.io/spring-webflow/)



[www.mulesoft.com/](http://www.mulesoft.com/)



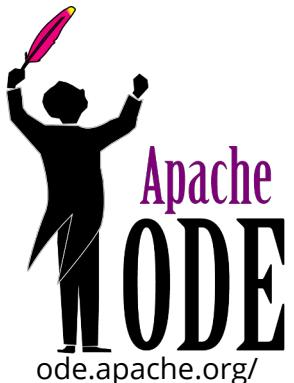
# Mediator EDA



**WEBFLOW**  
projects.spring.io/spring-webflow/



www.mulesoft.com/





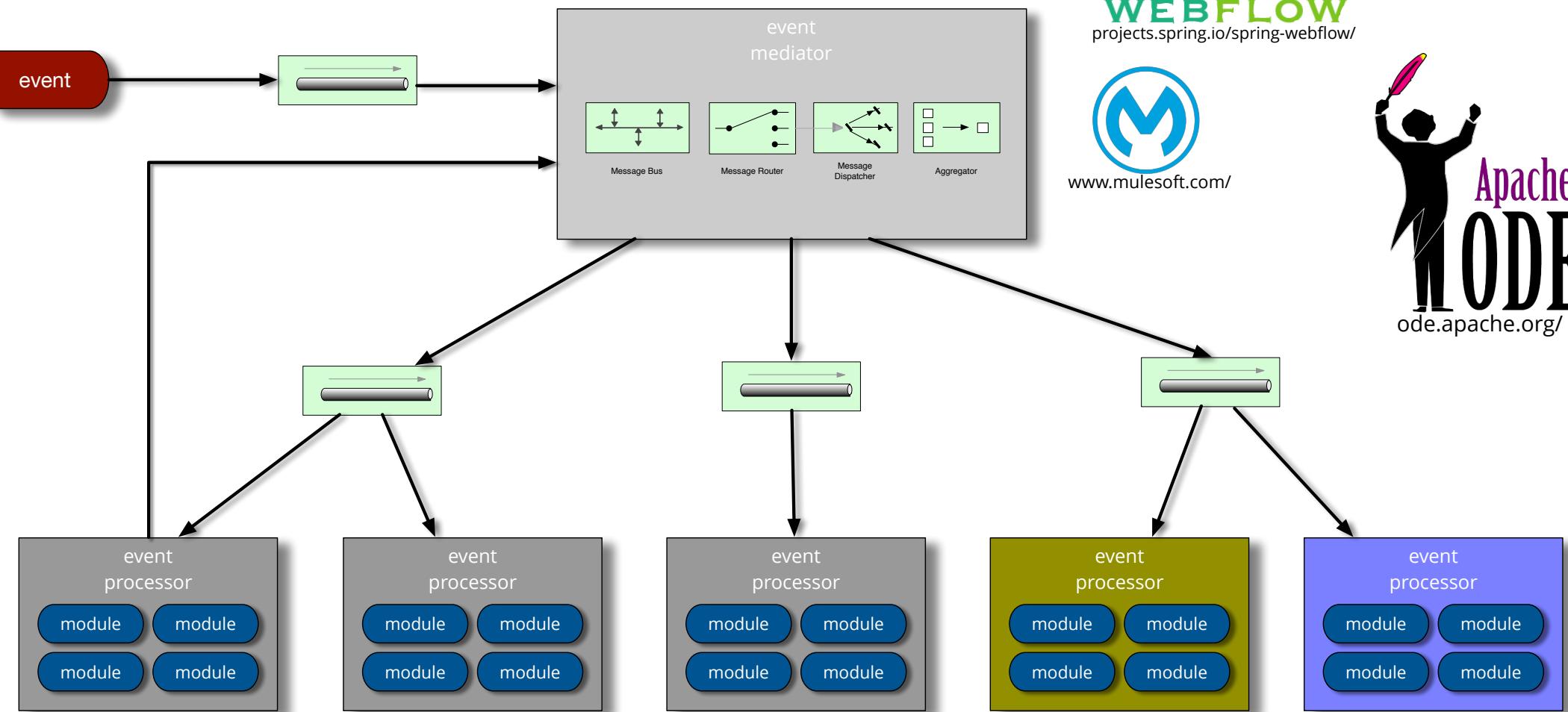








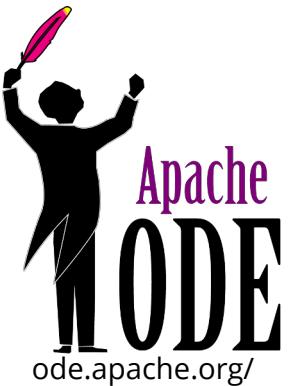
# Mediator EDA



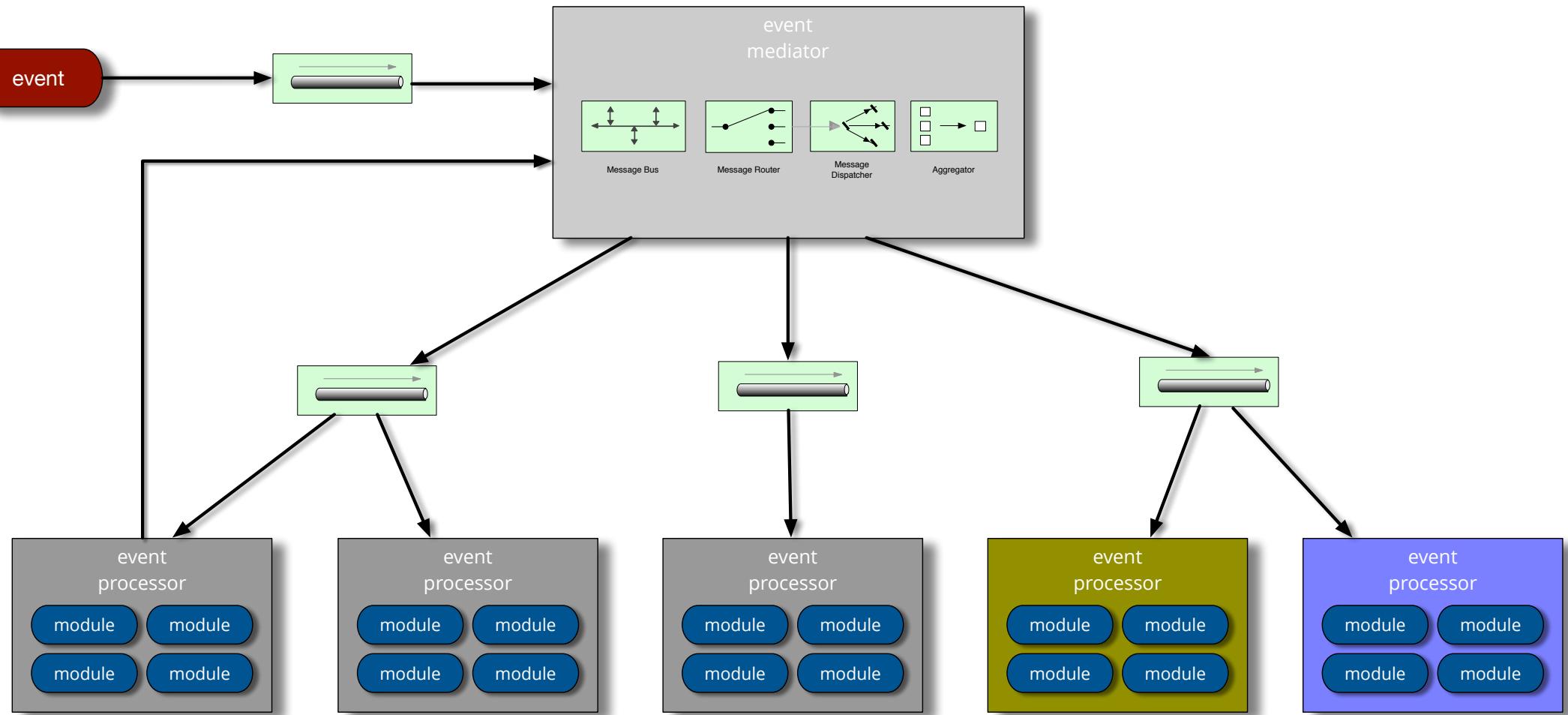
**WEBFLOW**  
projects.spring.io/spring-webflow/



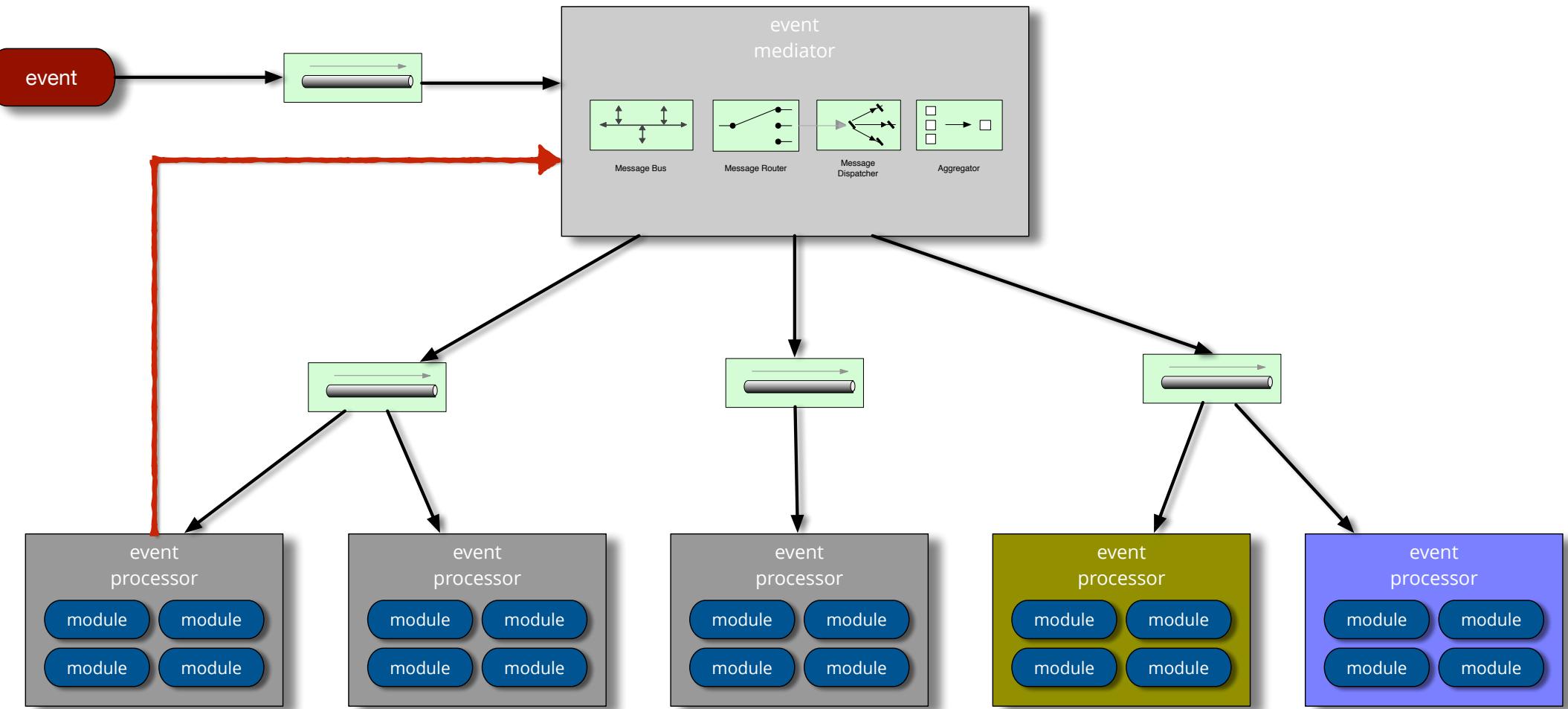
www.mulesoft.com/



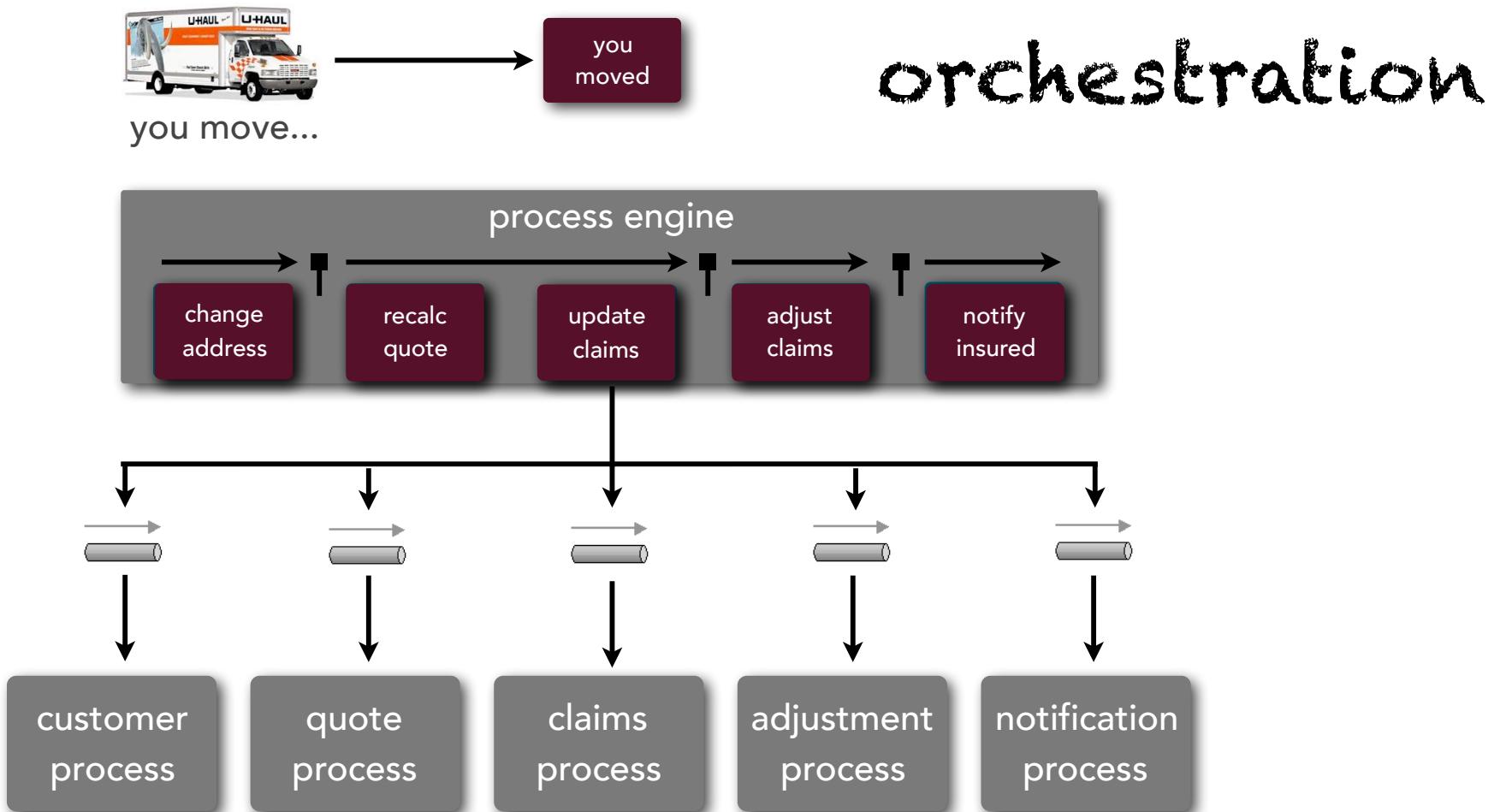
# Mediator EDA



# Mediator EDA



# Mediator Message Flow





you  
moved

you move...

process engine

change  
address

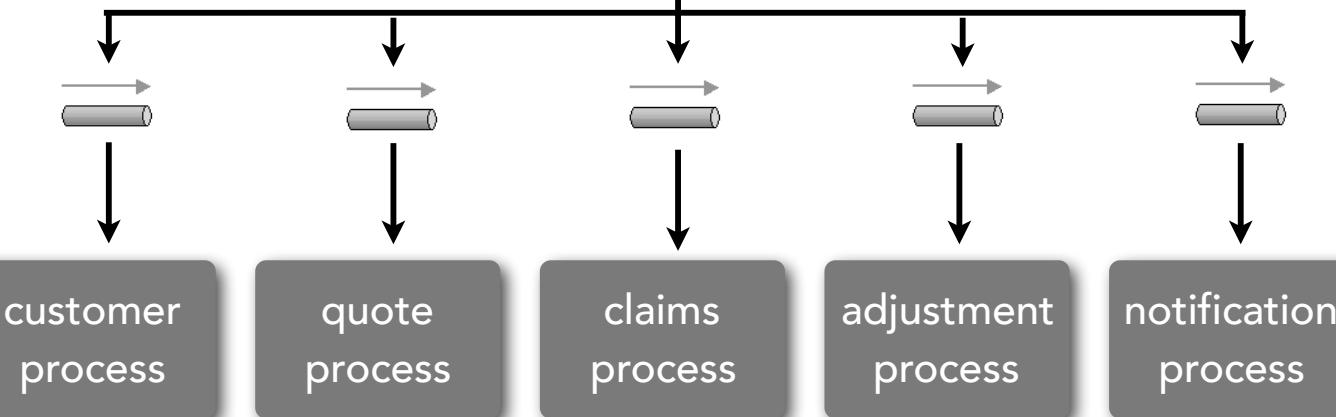
recalc  
quote

update  
claims

adjust  
claims

notify  
insured

Error handling?





you  
moved

you move...

process engine

change  
address

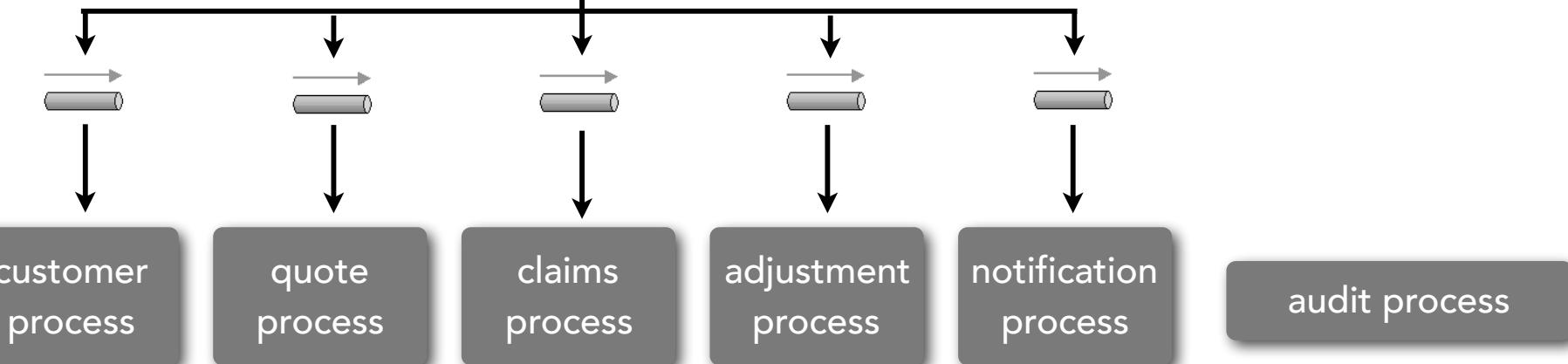
recalc  
quote

update  
claims

adjust  
claims

notify  
insured

evolution

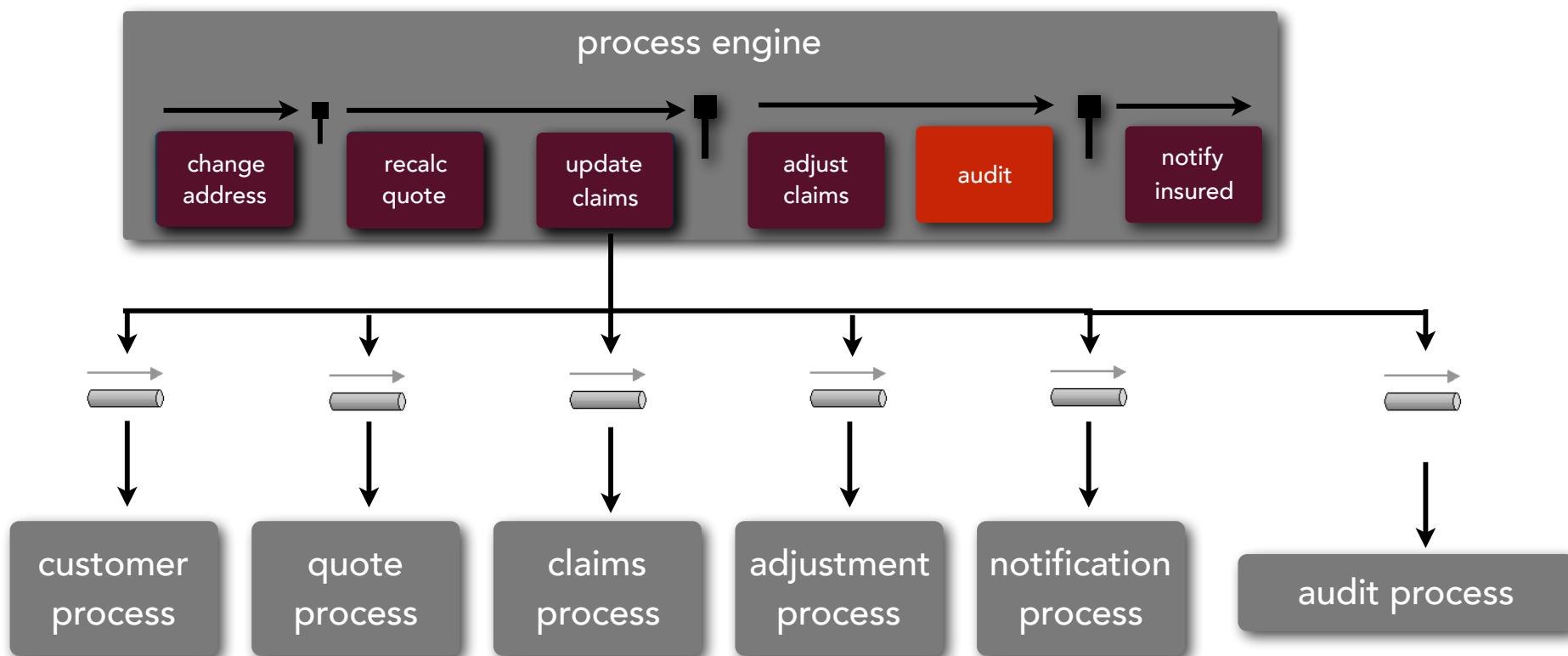




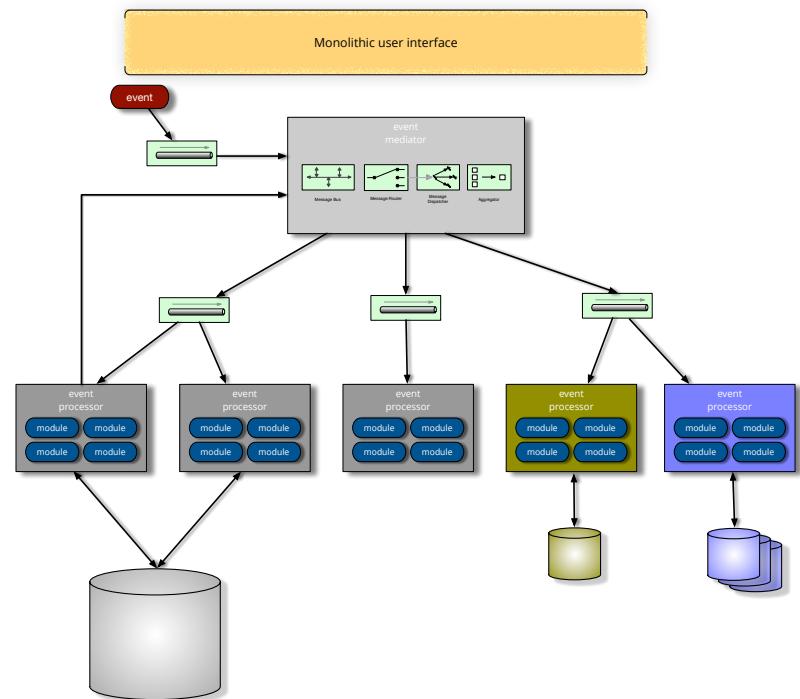
you  
moved

you move...

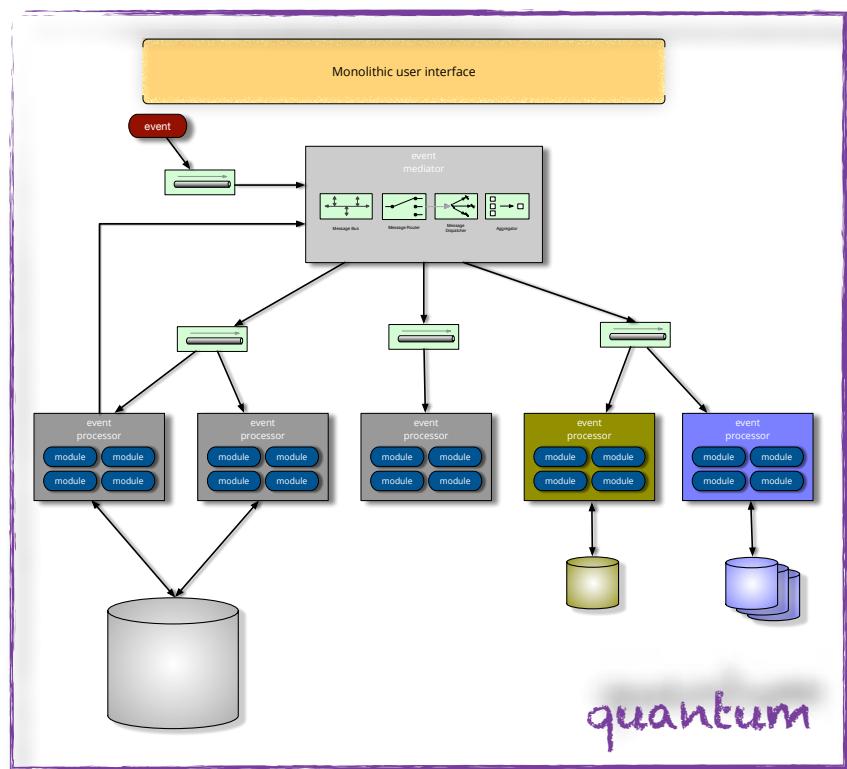
# evolution



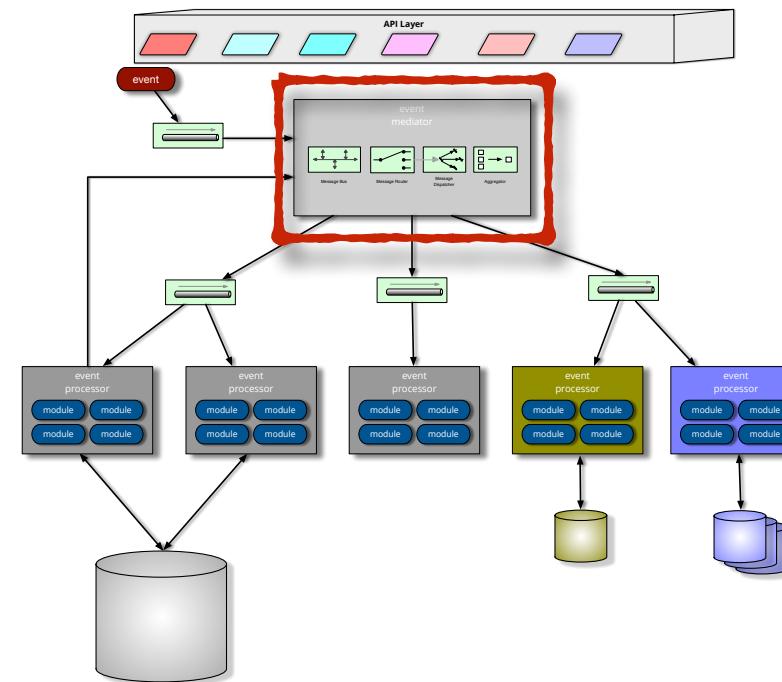
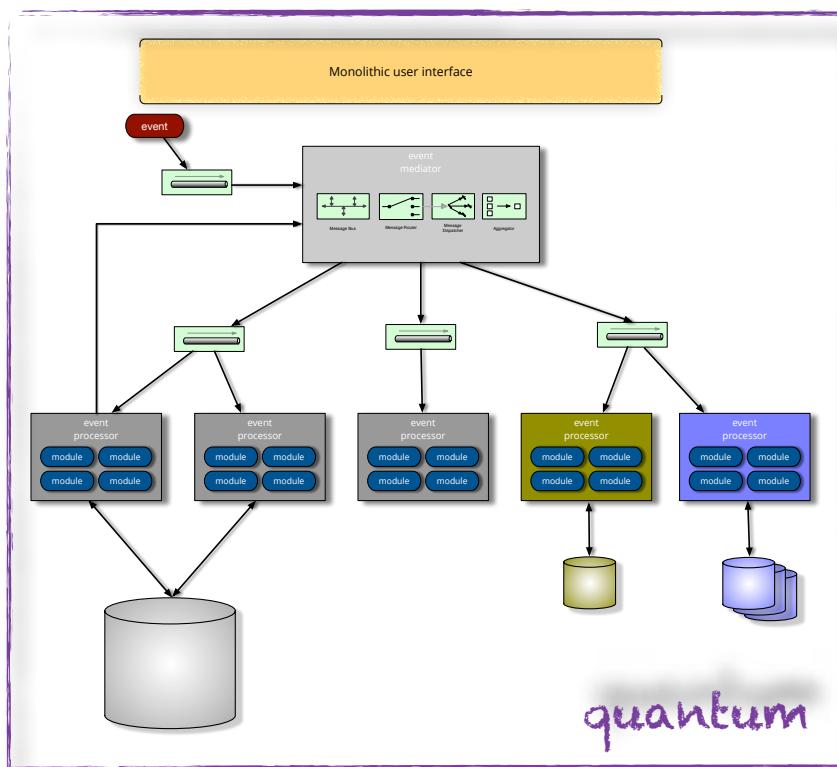
# Mediator Quanta



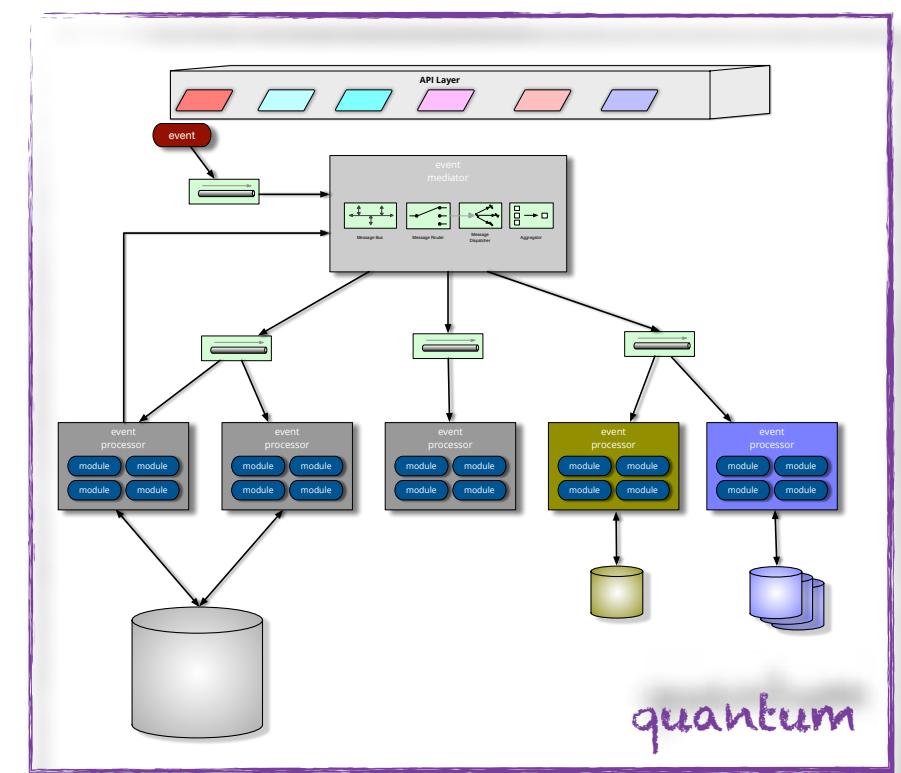
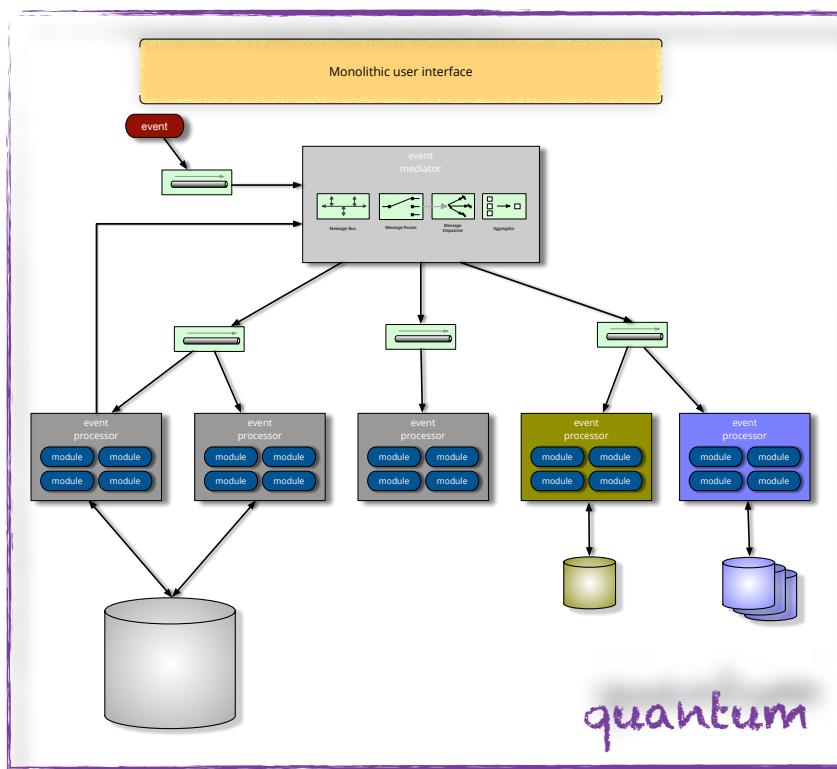
# Mediator Quanta



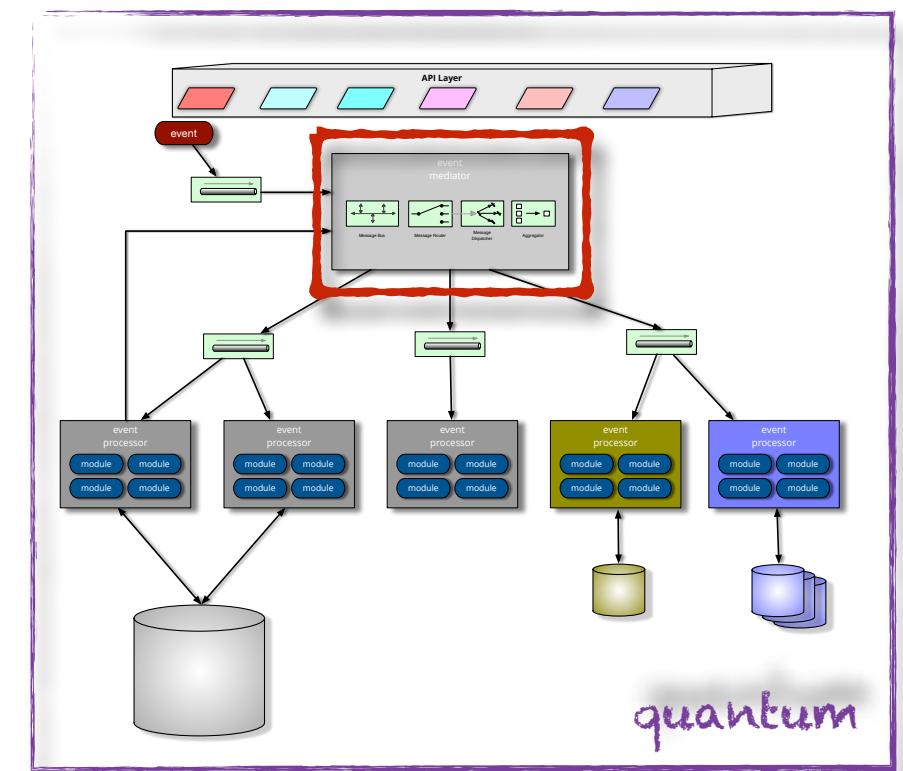
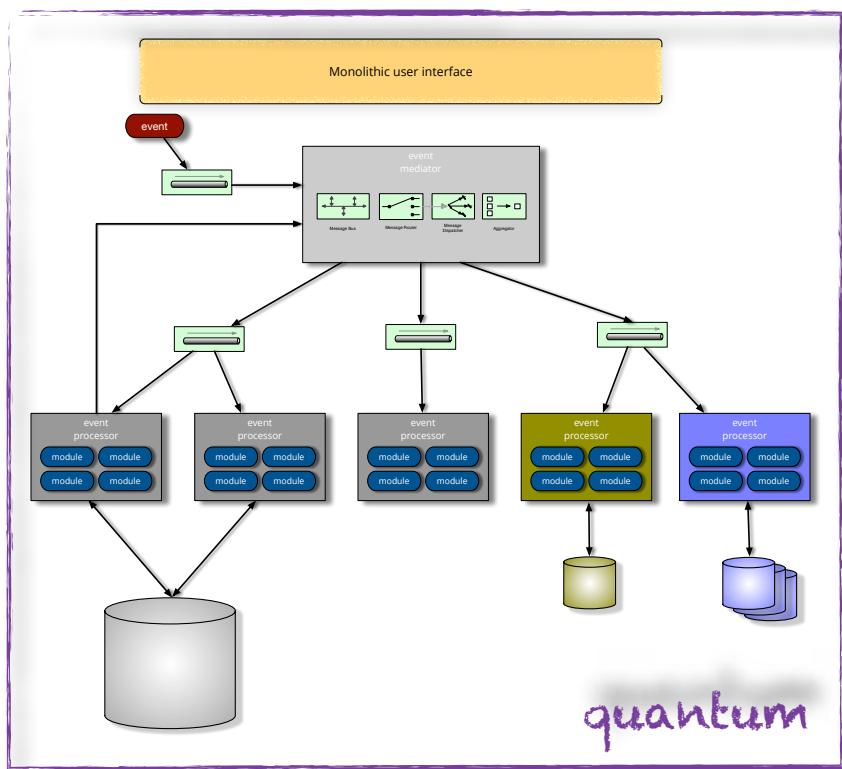
# Mediator Quanta



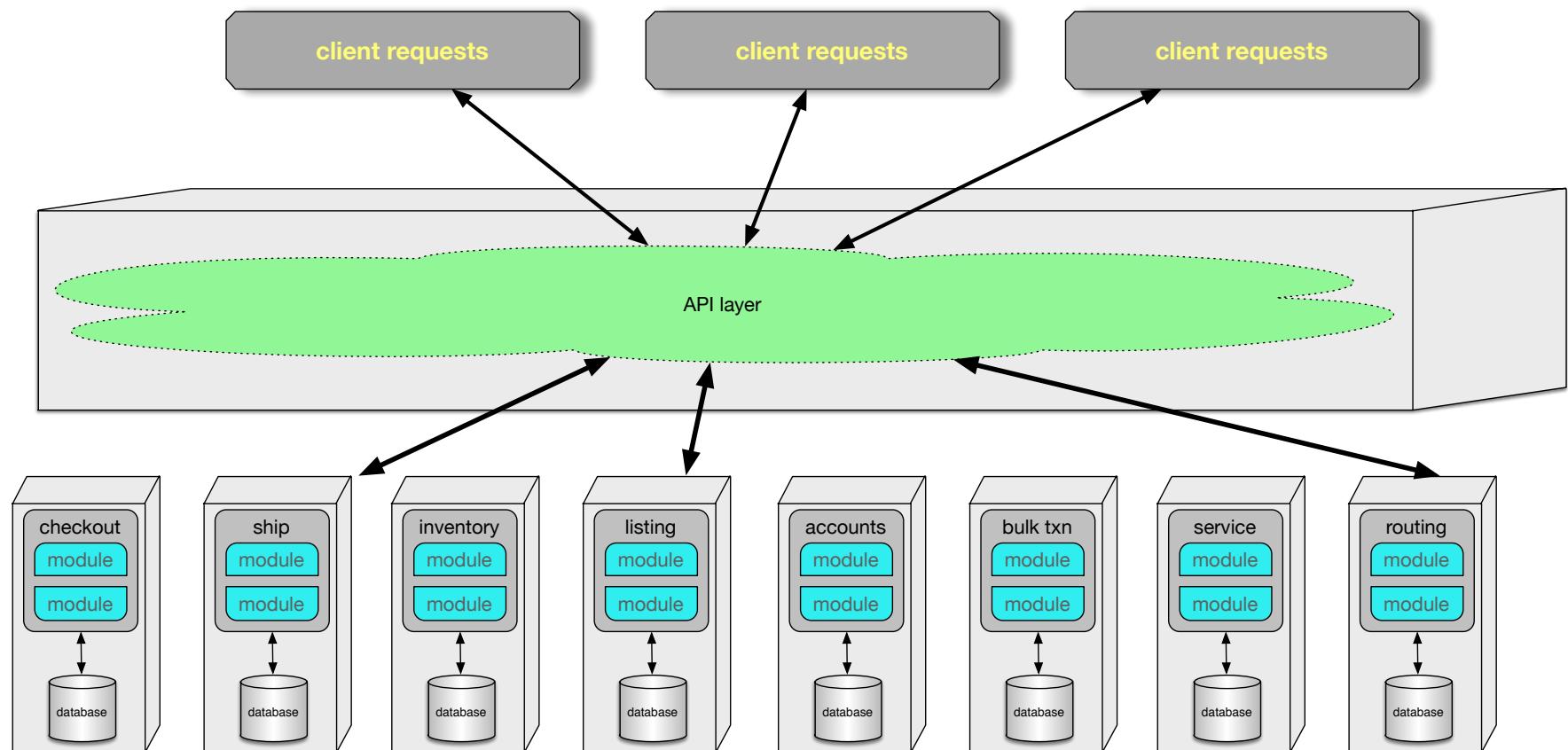
# Mediator Quanta



# Mediator Quanta



# Microservices



# What makes microservices so evolvable?

*extremely loose coupling*

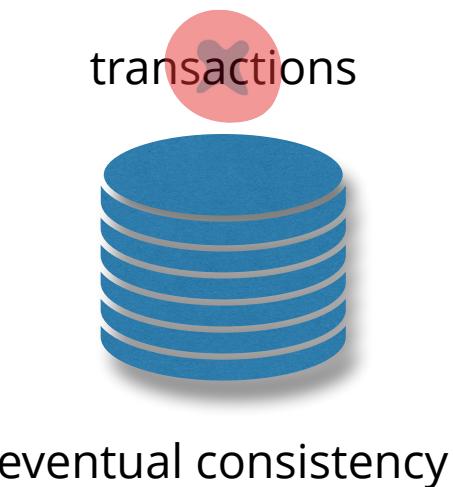
# What makes microservices so evolvable?

*extremely loose coupling*



# What makes microservices so evolvable?

*extremely loose coupling*



# What makes microservices so evolvable?

*extremely loose coupling*

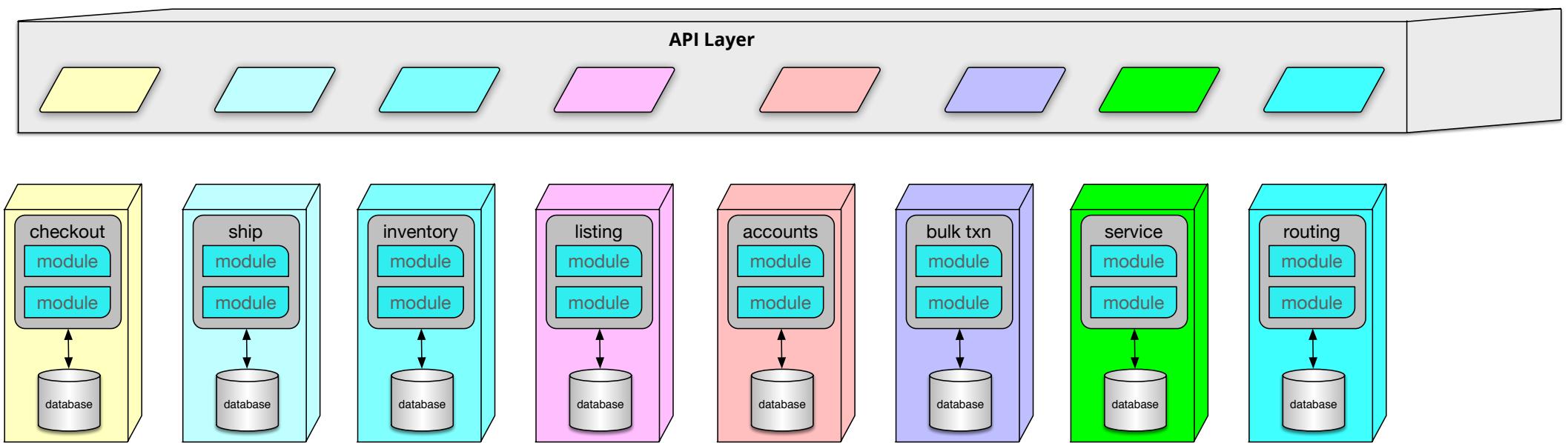


# What makes microservices so evolvable?

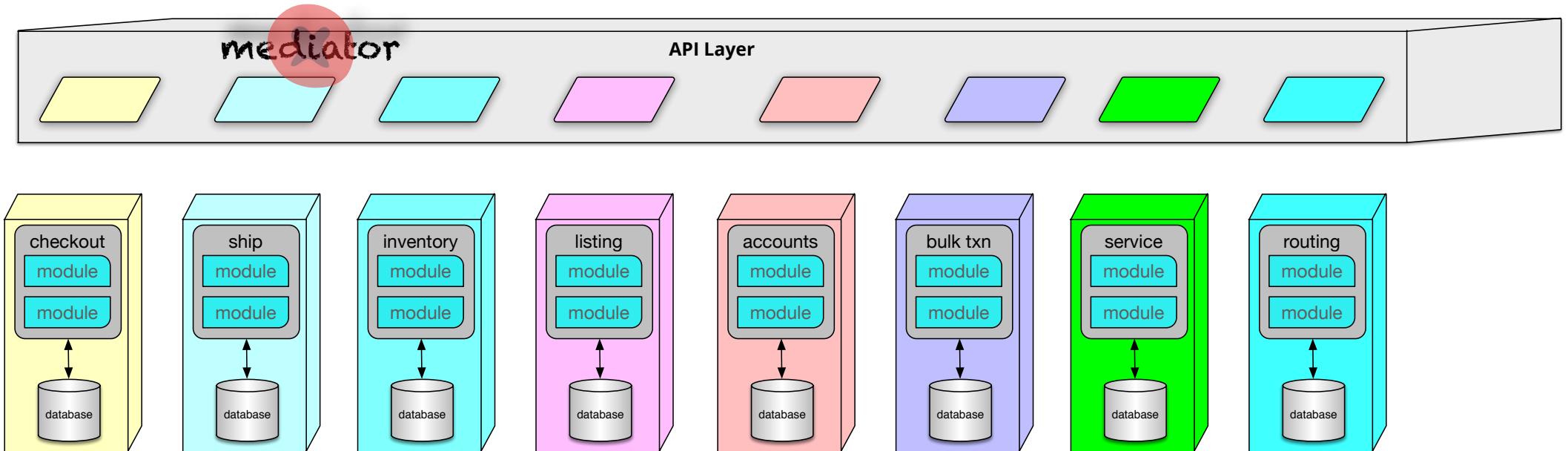
*extremely loose coupling*



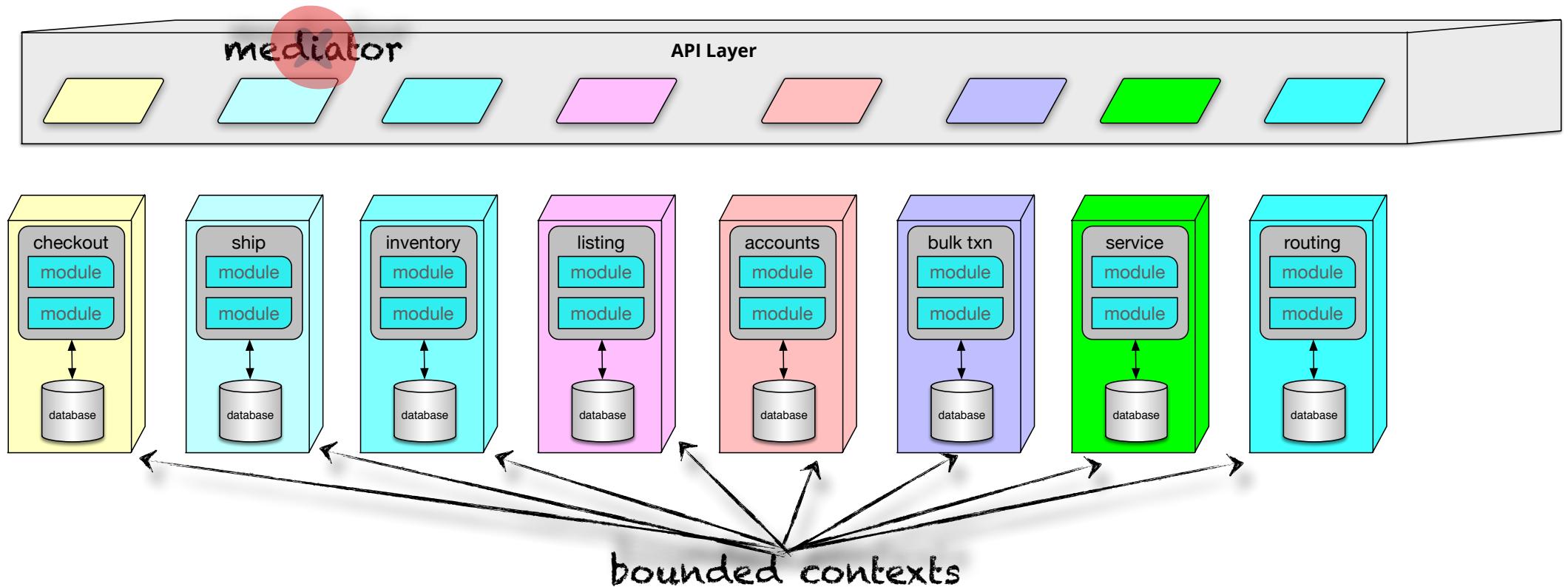
# Microservices



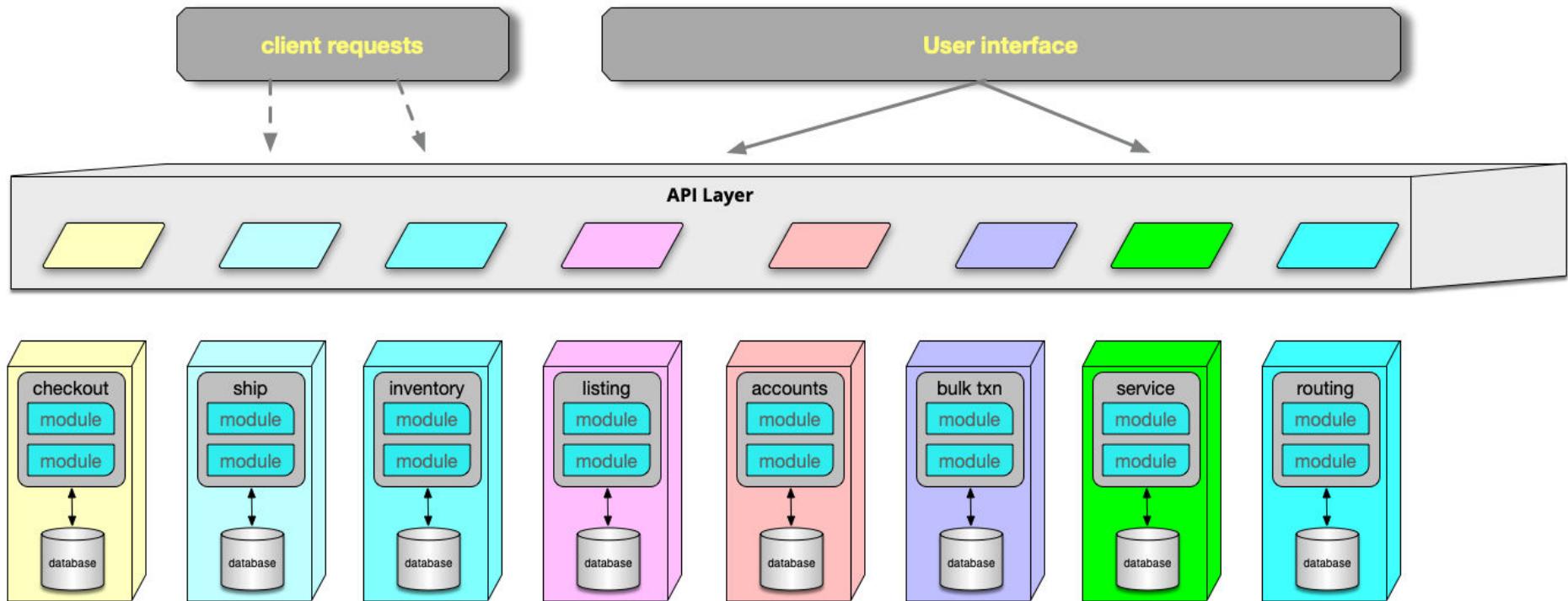
# Microservices



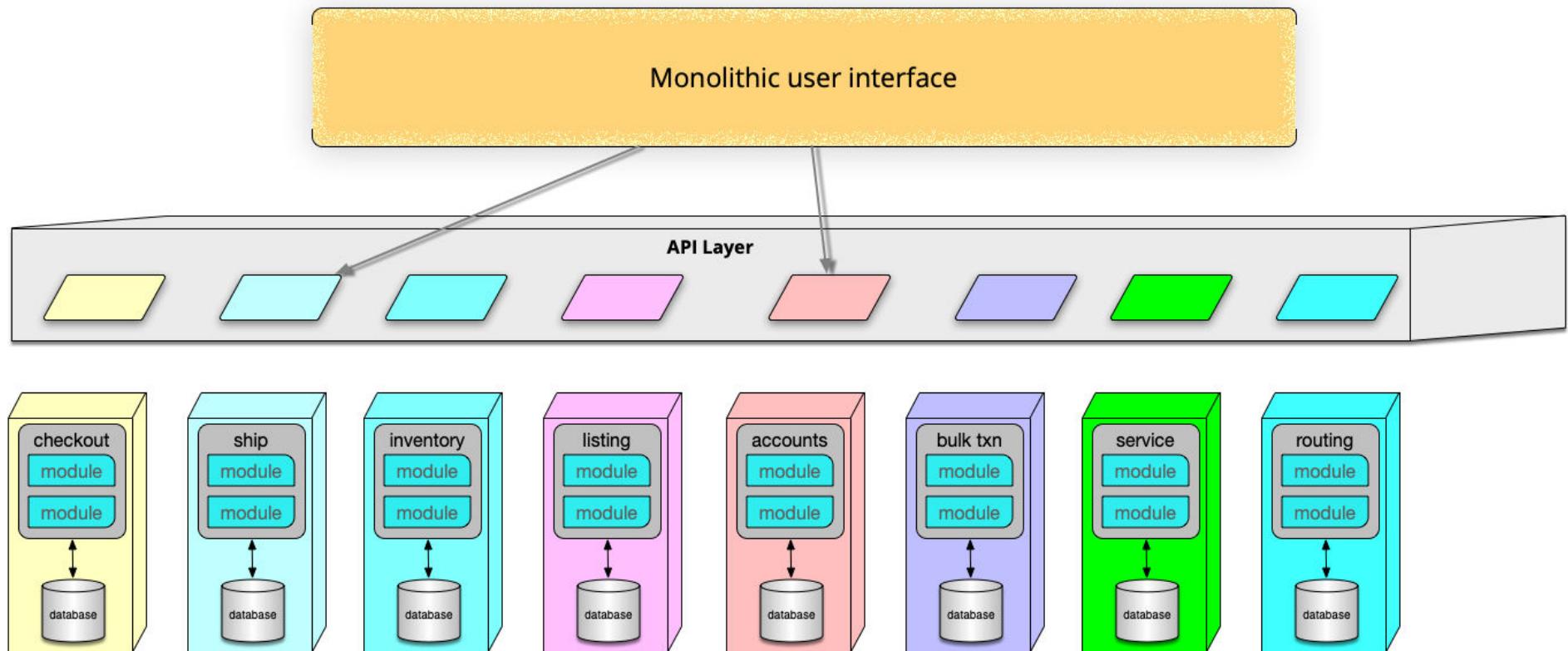
# Microservices



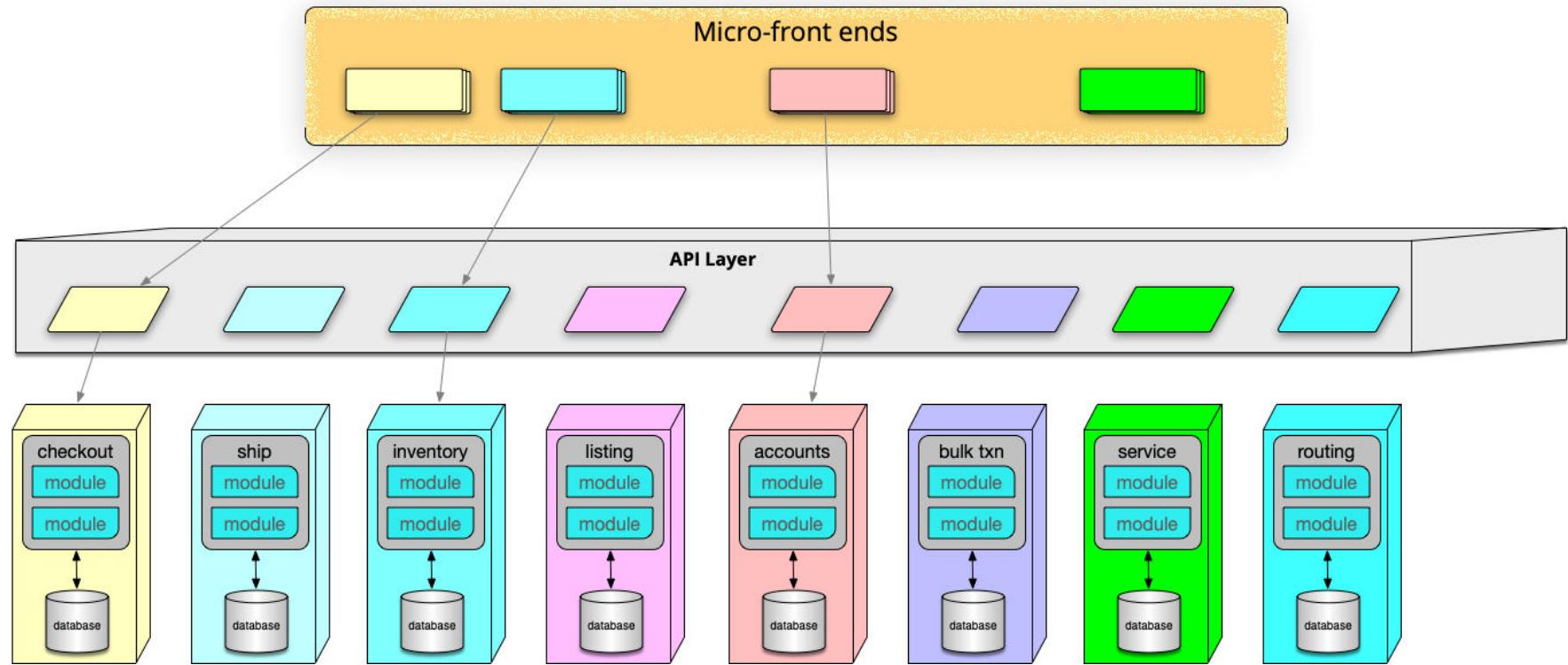
# Microservices



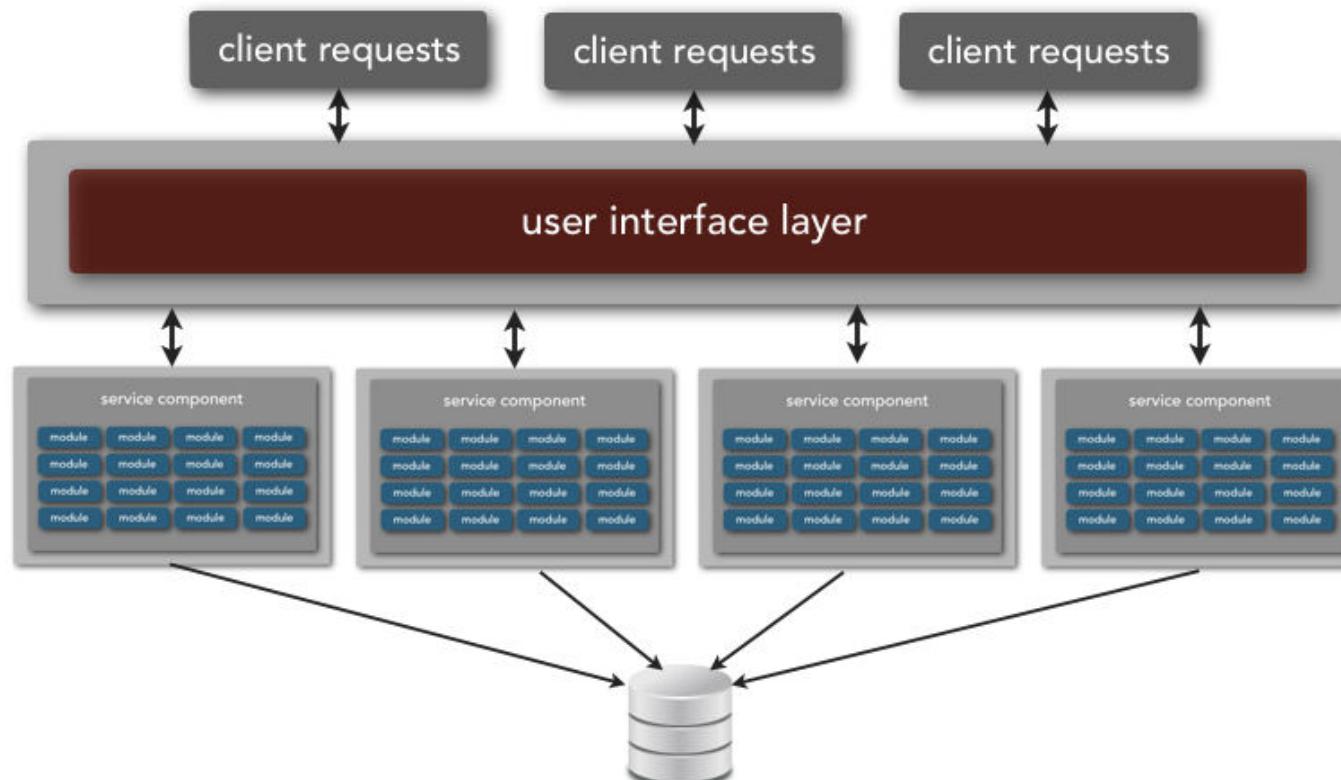
# Microservices: Monolithic UI



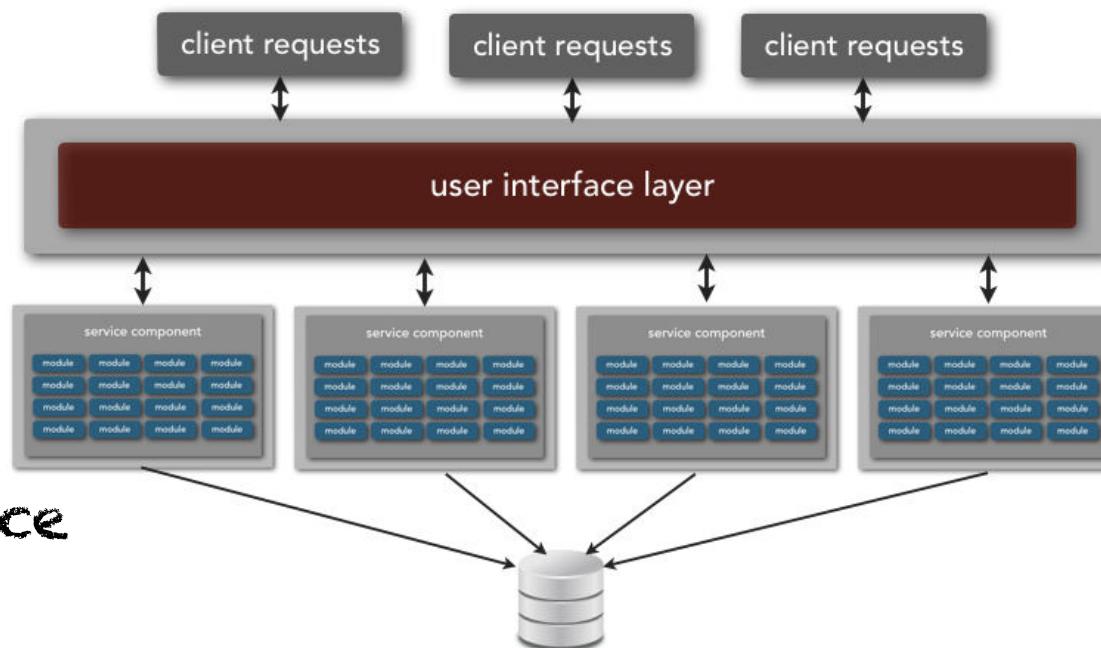
# Microservices



# Service-based Architectures

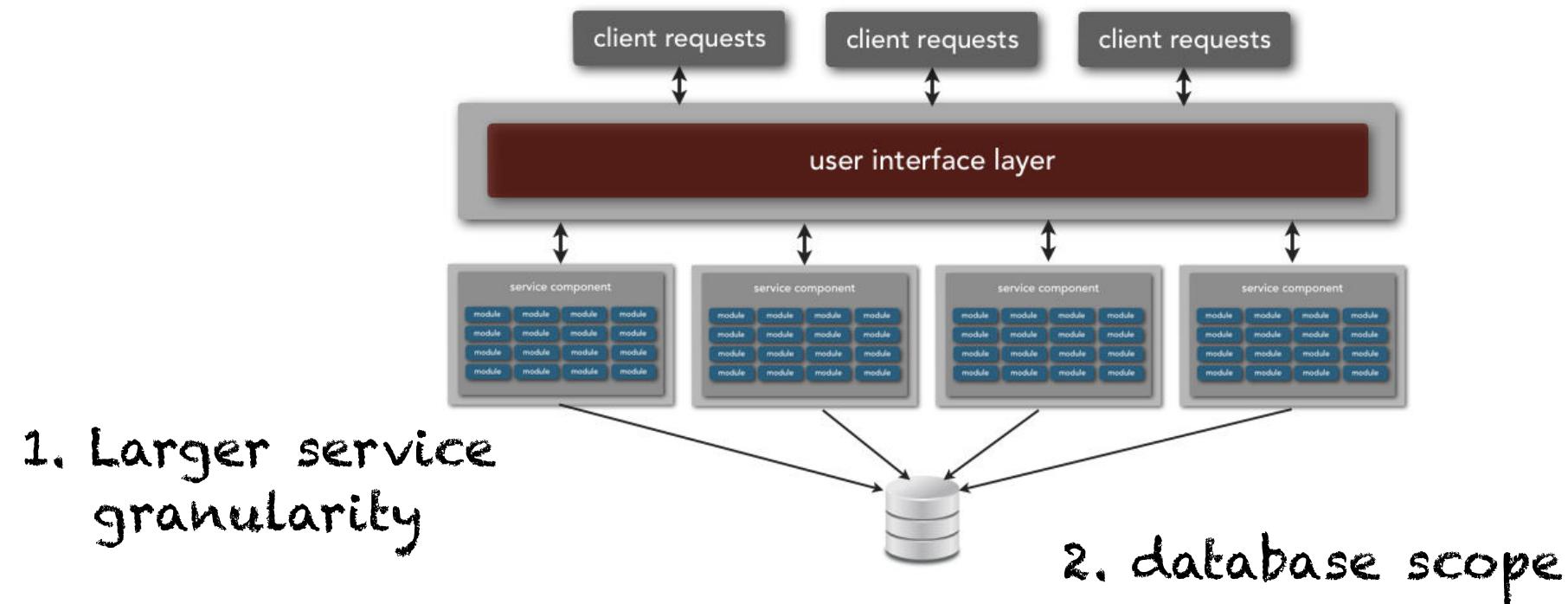


# Service-based Architectures



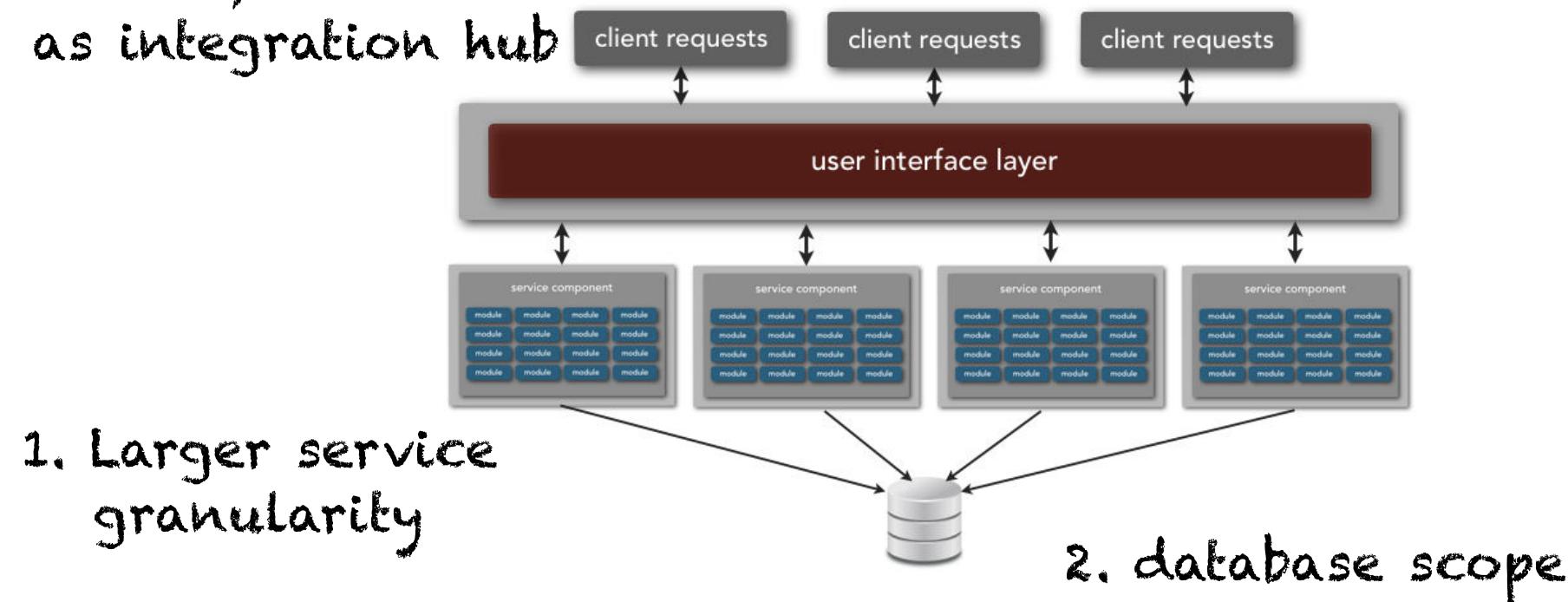
1. Larger service granularity

# Service-based Architectures



# Service-based Architectures

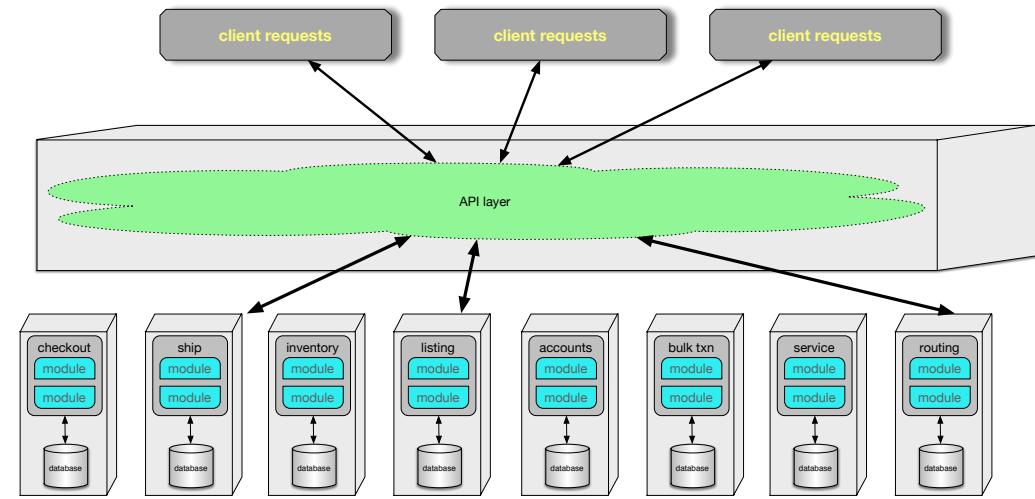
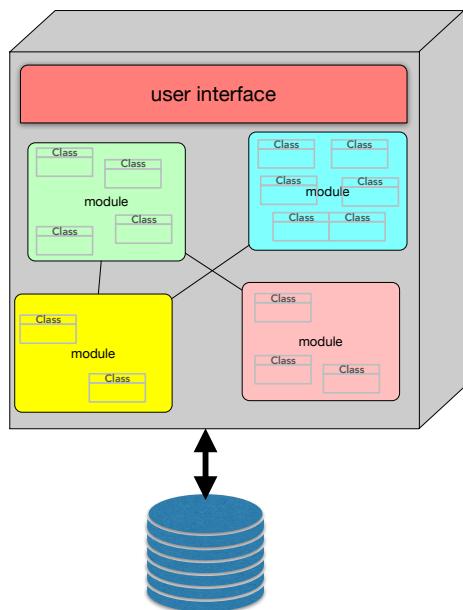
3. use of service bus  
as integration hub

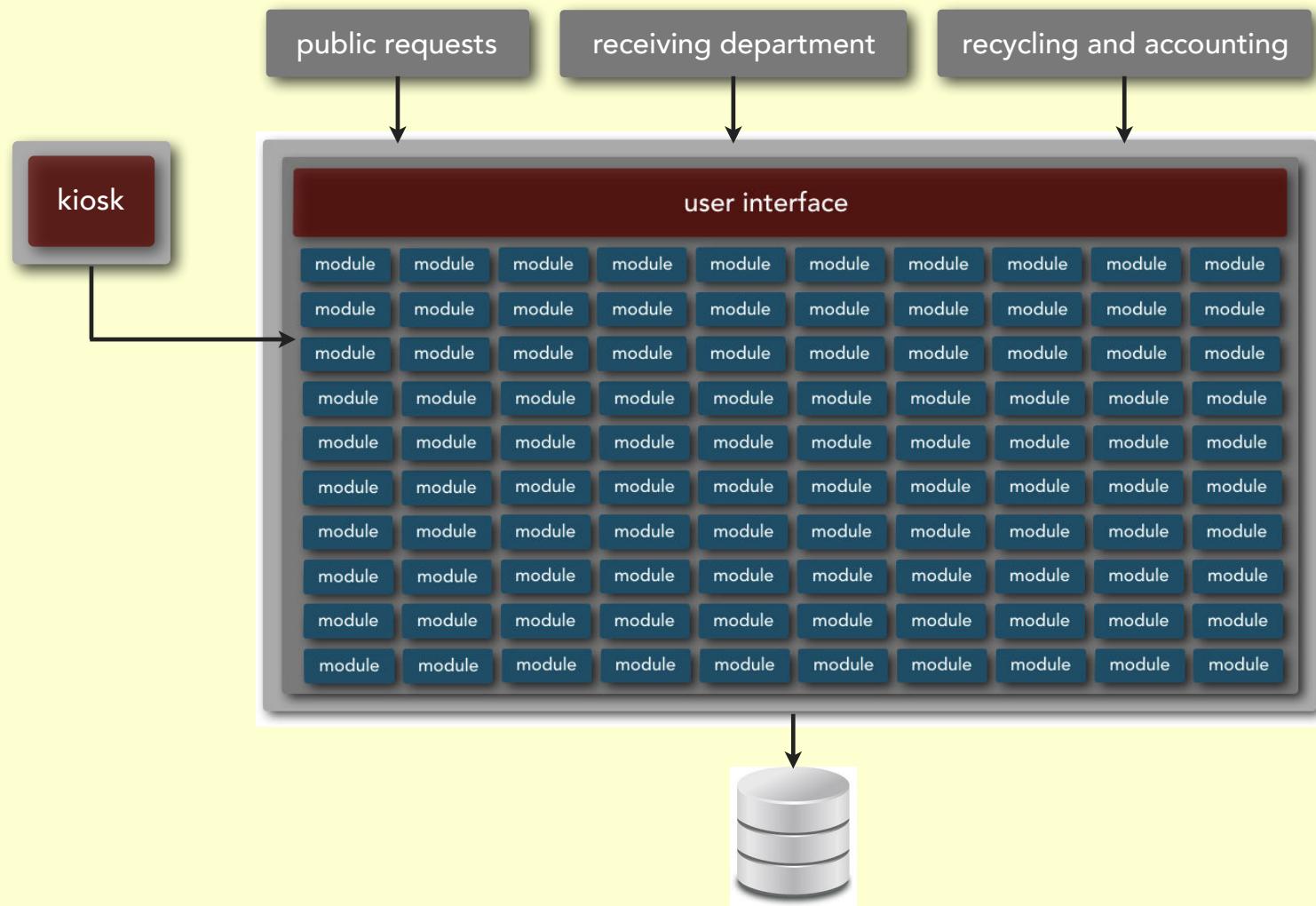


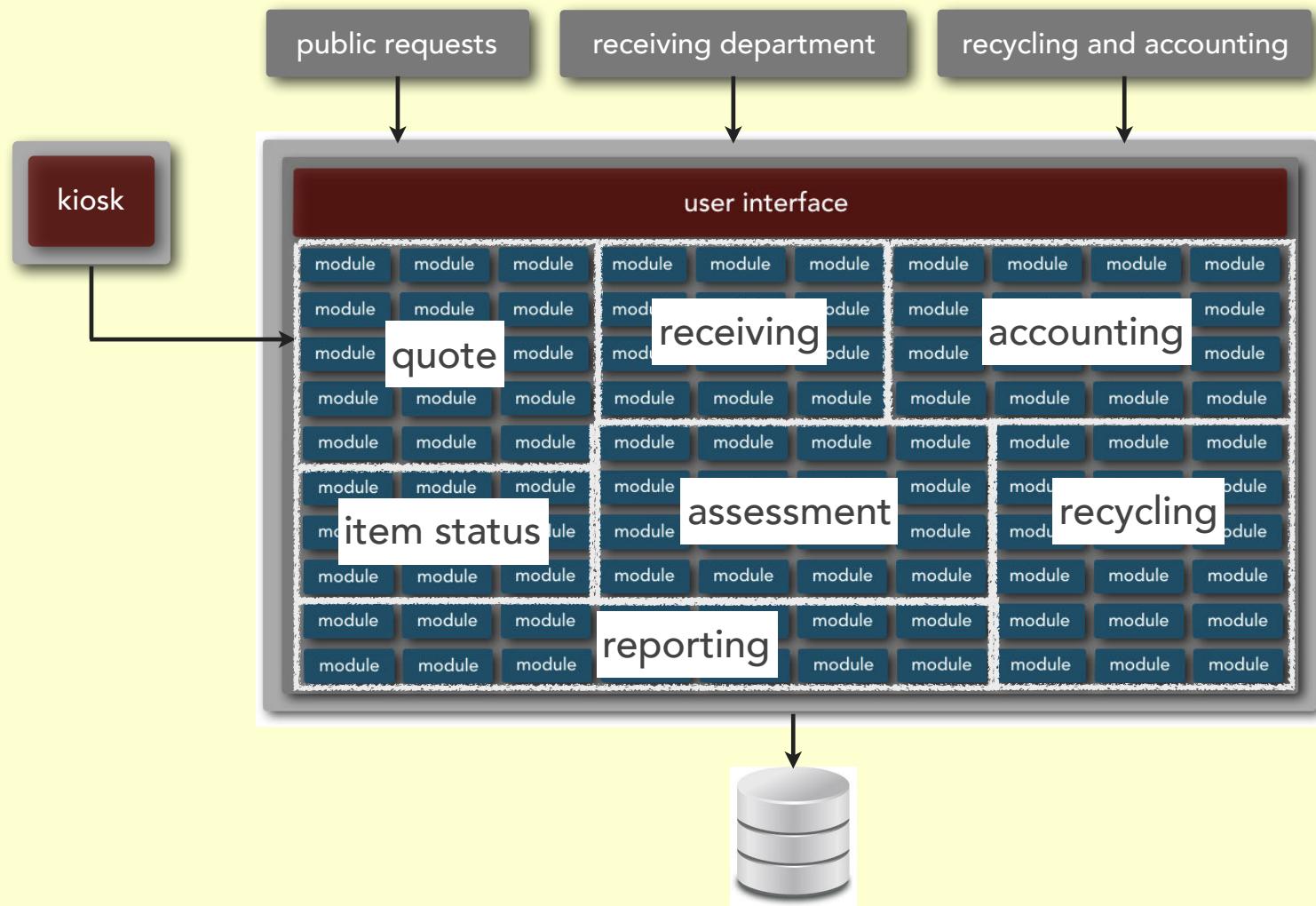
# Smaller Quanta Size

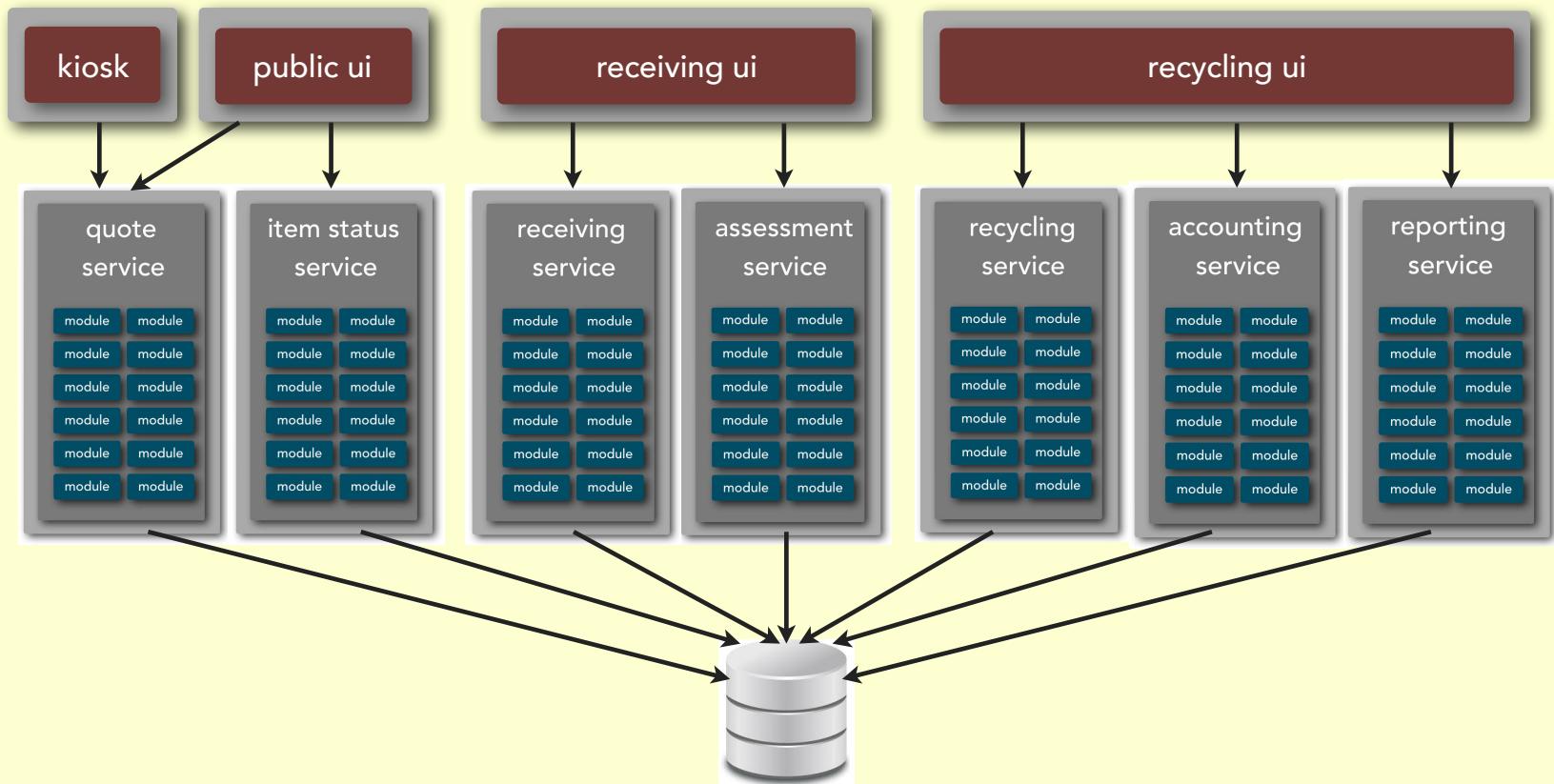
△  
≡

## Evolutionary

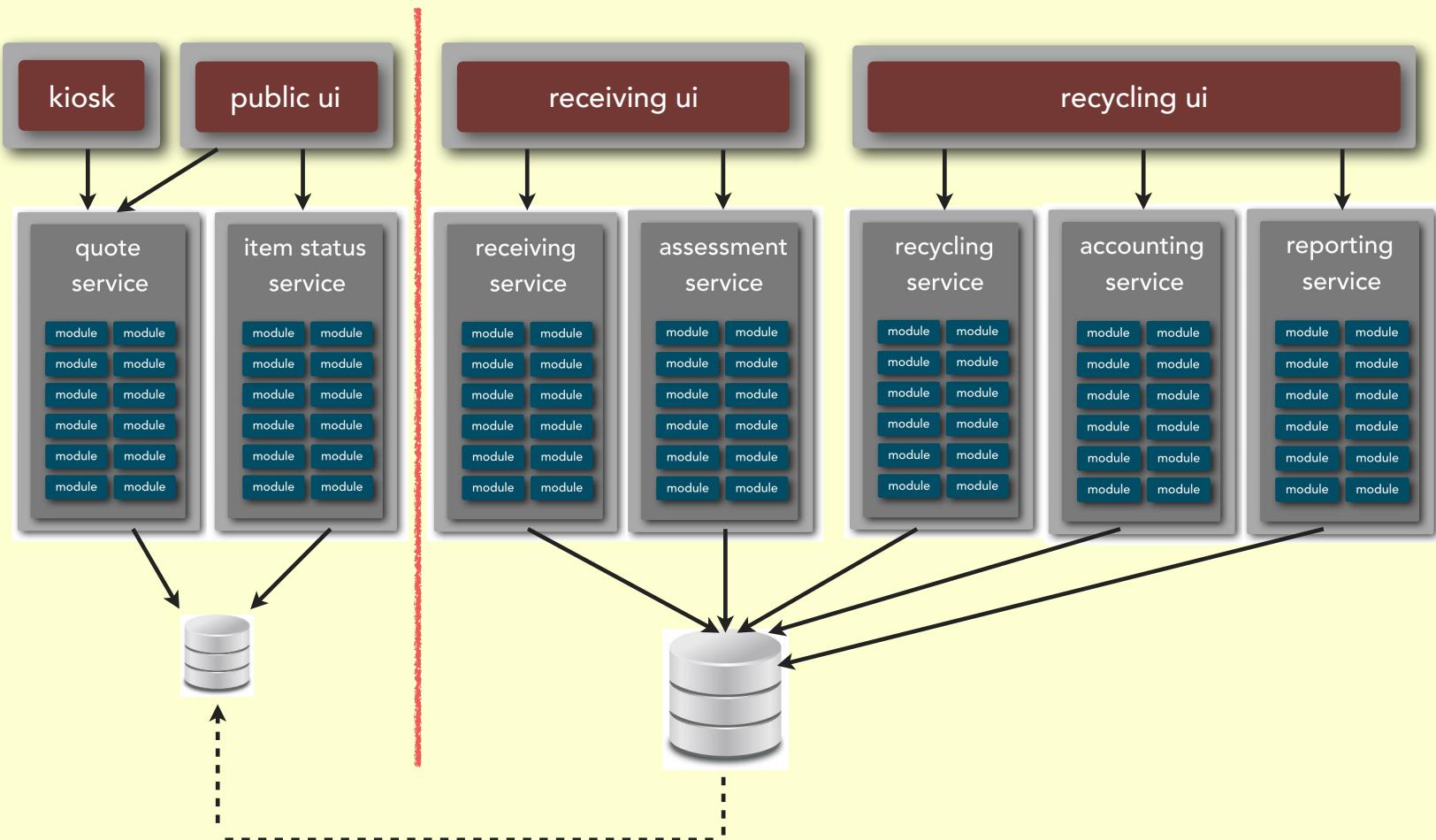








service-based architecture



Your Architectural Kata is...

# Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

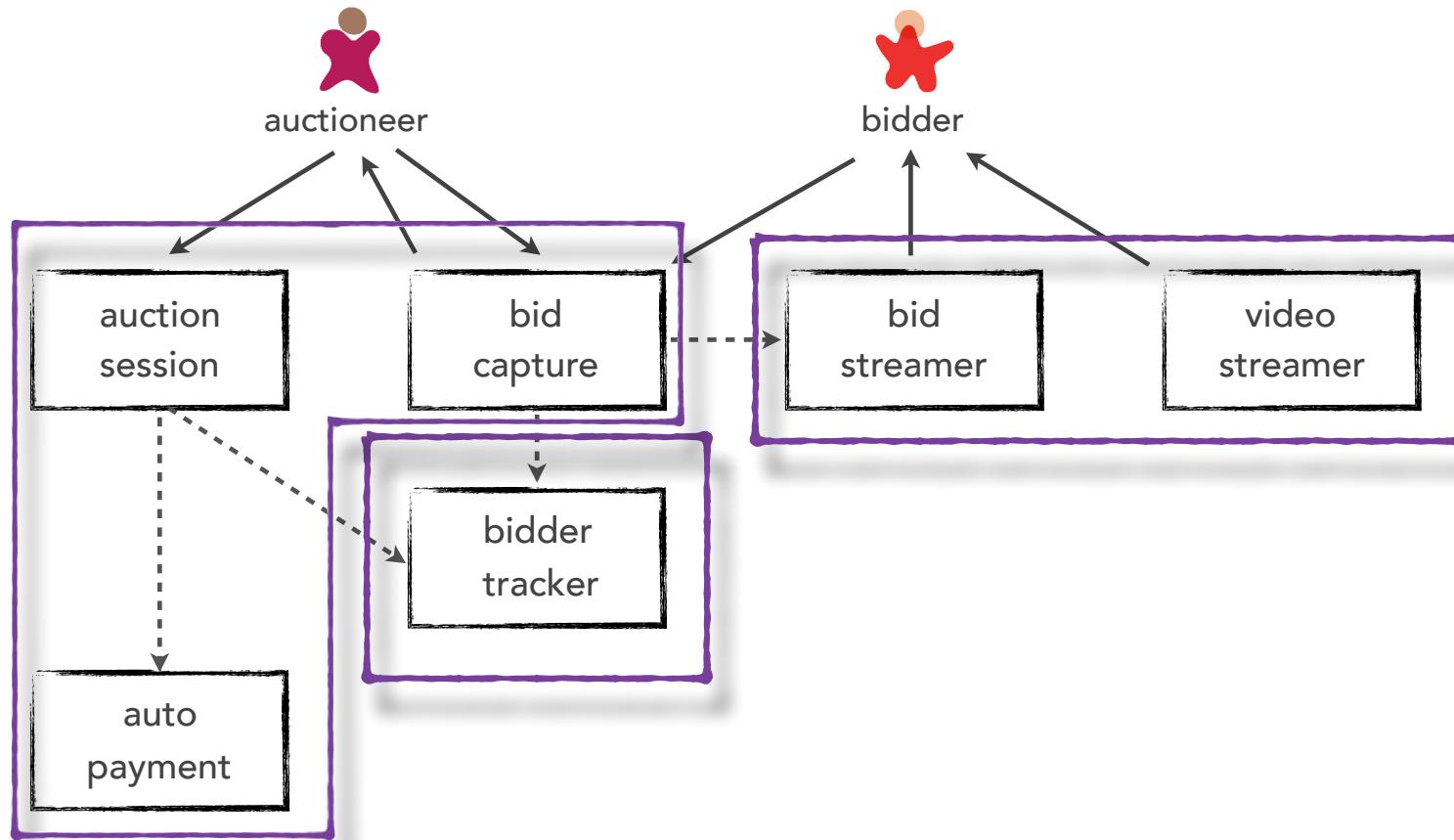
- **Users:** scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements:**
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability elasticity  
(security)

Your Architectural Kata is...

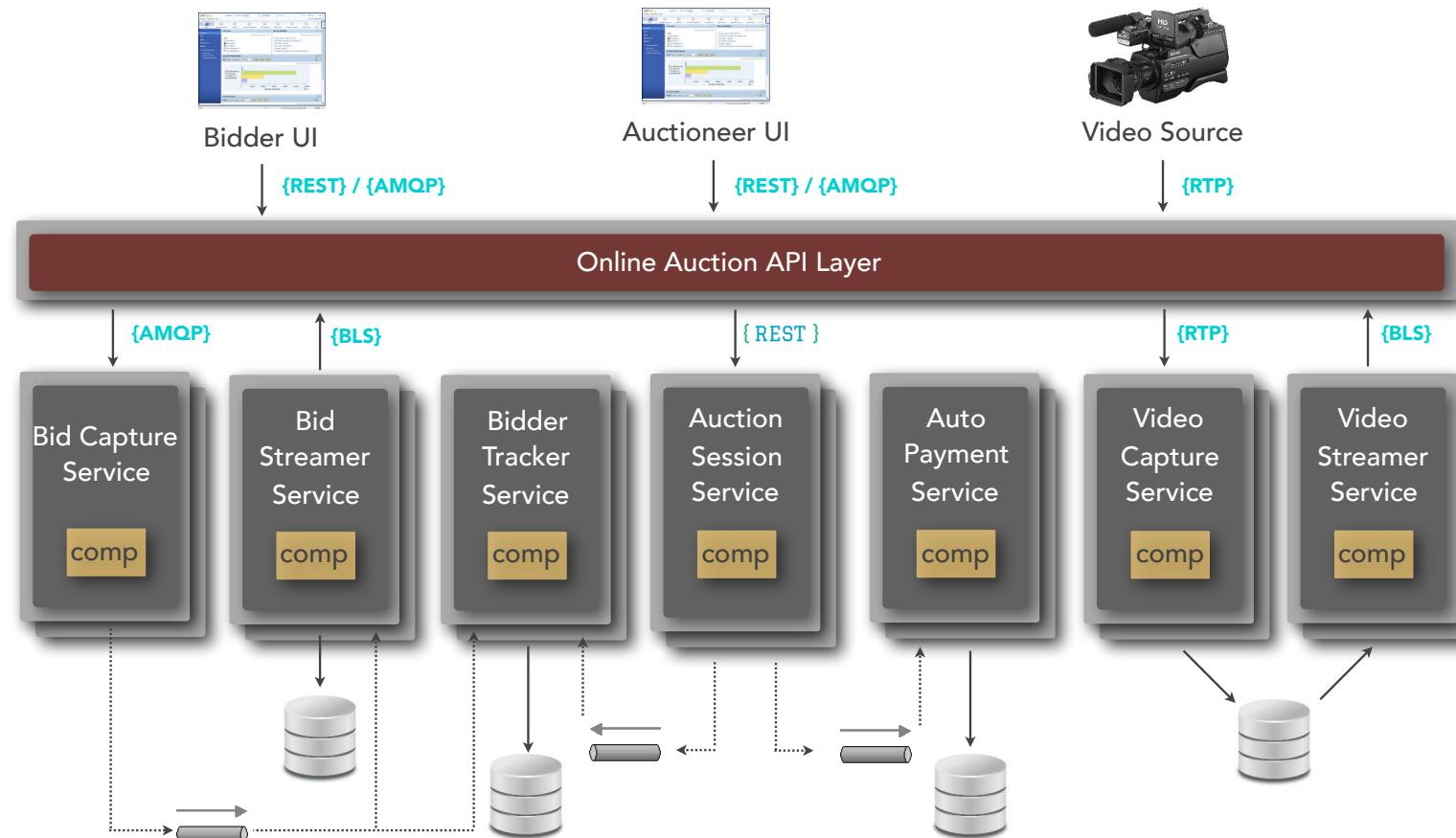
# Going Going Gone!

quanta



Your Architectural Kata is...

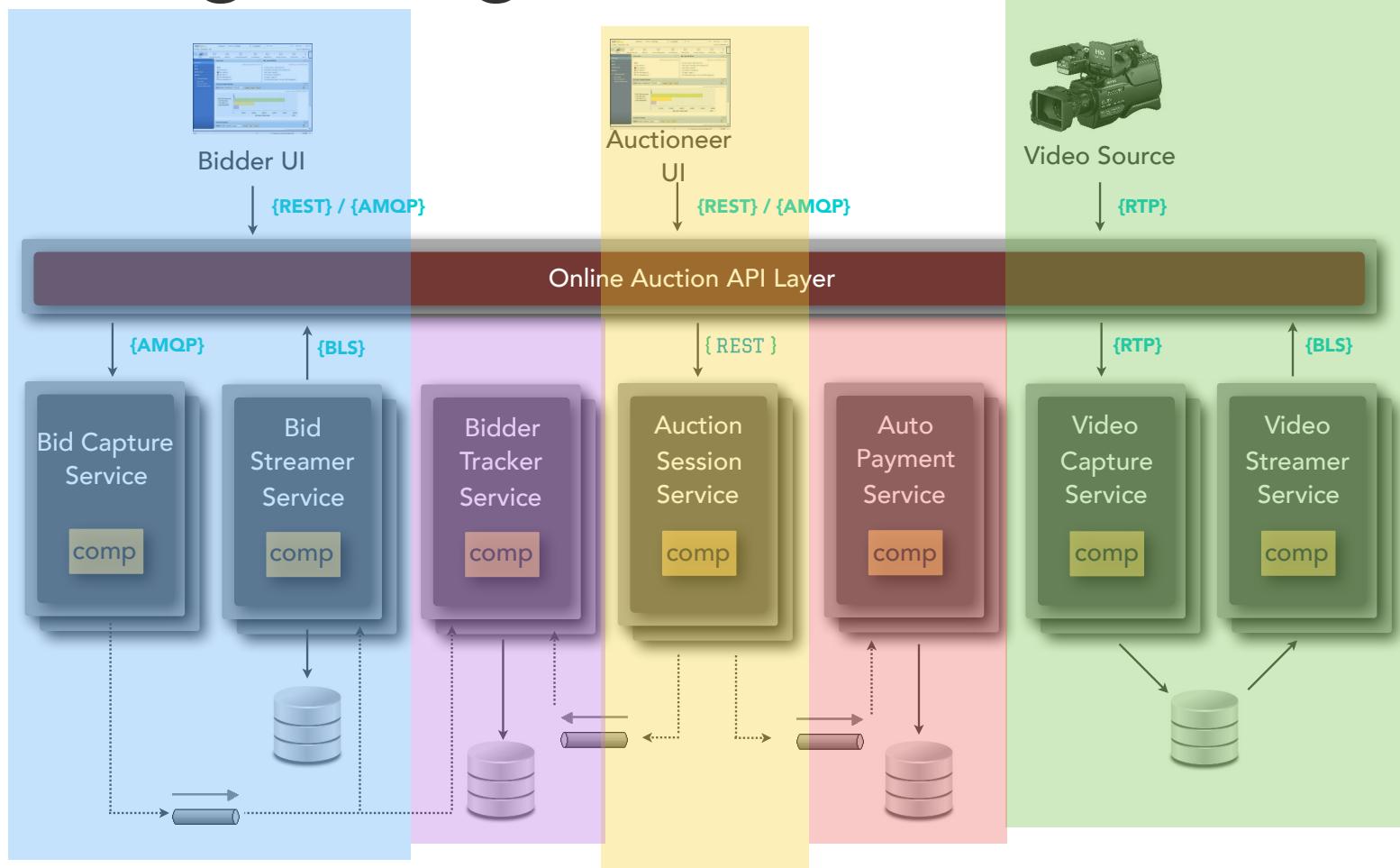
# Going Going Gone!



Your Architectural Kata is...

quanta

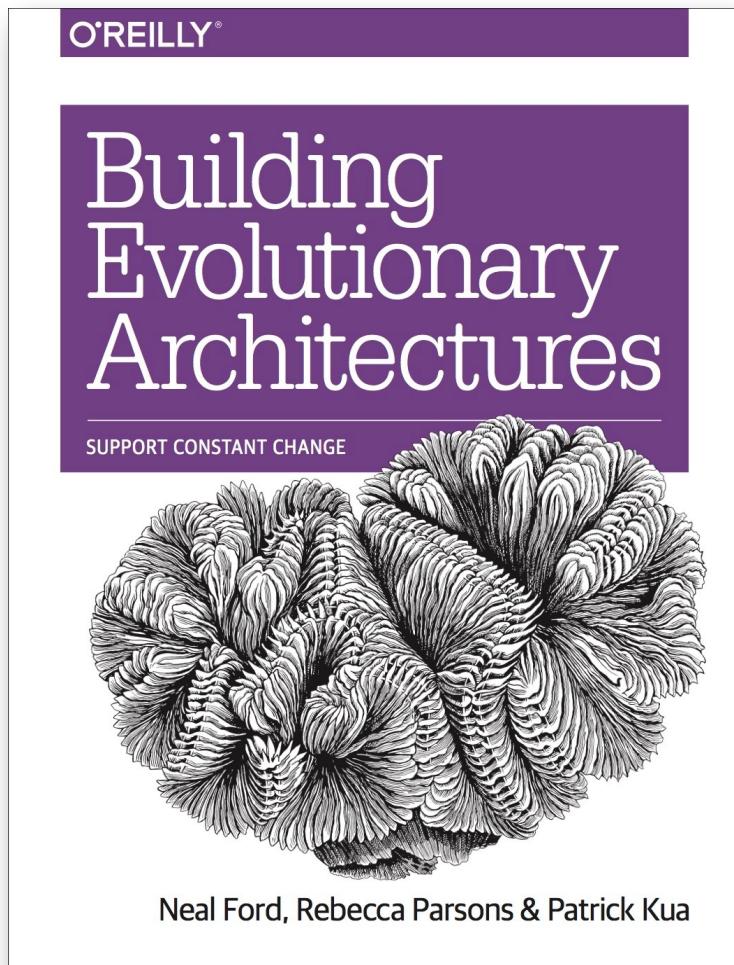
# Going Going Gone!



# 4. Identify Number of Quanta

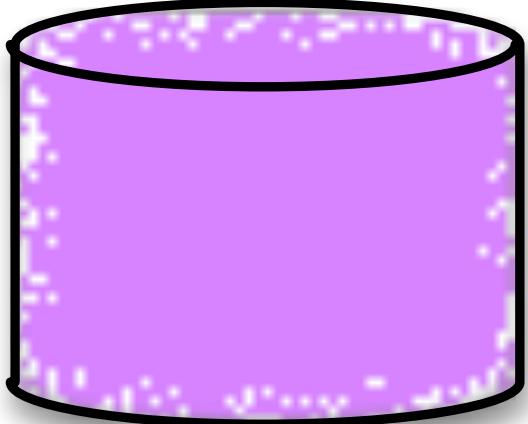


# Building Evolutionary Architectures



## EVOLUTIONARY DATA





```
class Class1 {
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main(string[] args) {
        // Logon
        BTULicenseManager licenseManager = new BTULicenseManager();

        // put your valid license here
        string networkLicense = "00000000-0000-0000-0000-000000000000-00000000";
        string password;

        password = "";
        licenseManager.Logon<( networkLicense, password >

        Console.WriteLine<"Logged on.">;
        string fullName = @"C:\Documents and Settings\rkuo.SNAPSTREAM\My Documents\My Videos\South Park-(Freak Strike)-2004-08-17-0.mpg";
        BTULibrary library = new BTULibrary<>

        // Get properties
        PUSPropertyBag bag = library.GetMediaByFullName<( fullName >

        // Print properties to the console
        Console.WriteLine<"Properties of <0>", fullName >;
        foreach< PUSProperty prop in bag.Properties > {
            Console.WriteLine<"Property: <0>, <1>", prop.Name, prop.Value >
        }

        // Put the PUSPropertyBag into a more friendly collection class.
        // It's a good idea for you to write a friendlier wrapper class that
        // would allow you to add and remove properties and cast back to
        // the PUSPropertyBag type on the fly.
        ArrayList aProperties = new ArrayList<( bag.Properties >

        // Change the "EpisodeDescription" property
        foreach< PUSProperty prop in aProperties > {
            if< prop.Name == "EpisodeDescription" > {
                prop.Value = "The boys compete to appear on a talk show. <Edited by Beyond TU Framework>";
            }
        }

        // Create a new PUSPropertyBag with the edited property
        PUSPropertyBag newBag = new PUSPropertyBag<>;
        newBag.Properties = (PUSPropertyID)aProperties.ToArray<( typeof(PUSProperty) >

        // This method will edit the recording
        library.EditMedia<( fullName, newBag >

        // Print properties to the console and verify the change
        Console.WriteLine<"Edited properties of <0>", fullName >;
        foreach< PUSProperty prop in bag.Properties > {
            Console.WriteLine<"Property: <0>, <1>", prop.Name, prop.Value >
        }

        // Pause so you can see the output, hit enter to continue
        Console.WriteLine<"Press any key to exit...">;
        Console.ReadLine();
        return;
    }
}
```

# Data & code are both abstractions based on the real world.



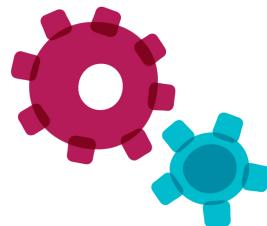
The image shows a 3D cylinder containing a screenshot of a code editor. The code editor displays a Java class named `Class1`. The code is as follows:

```
class Class1 {
    /**
     * Entry point for the application.
     */
    @ThreadLocal
    static void Main(String[] args) {
        // Logon
        BTULicenseManager licenseManager = new BTULicenseManager();
        // put more valid ticket here
        string networkLicense = "00000000-0000-0000-0000-000000000000";
        string password;
        password = "";
        licenseManager.Logon(networkLicense, password);
        Console.WriteLine("Logged on.");
        string fullName = @"C:\Documents and Settings\ruke.SNAPSTERM\My Documents\My Videos\South Park - Freak Strike>-2004-08-17-0.mpg";
        BTULibrary library = new BTULibrary();
        //Get properties
        PUSPropertyBag bag = library.GetMediaByFullName(fullName);
        //Print properties to the console
        Console.WriteLine("Properties of <{0}>, {1}", fullName);
        foreach(PUSProperty prop in bag.Properties) {
            Console.WriteLine("Property: <{0}>, {1}", prop.Name, prop.Value);
        }
        // Put the PUSPropertyBag into a more friendly collection class.
        // It's a good idea for you to write a friendlier orange class that
        // would allow you to add and remove properties and cast back to
        // the PUSPropertyBag type on the fly.
        foreach(PUSProperty prop in bag.Properties) {
            prop.Name = "EpisodeDescription";
            foreach(PUSProperty prop in bag.Properties) {
                if(prop.Name == "EpisodeDescription") {
                    prop.Value = "The boys compete to appear on a talk show. <Edited by Beyond TU Framework>";
                }
            }
            bag.Add(prop);
        }
        // Create a new PUSPropertyBag with the edited property
        PUSPropertyBag newBag = new PUSPropertyBag();
        newBag.Properties = (PUSProperty[])bag.Properties.ToArray(typeof(PUSProperty));
        // This method will edit the recording
        library.EditMedia(fullName, newBag);
        // Print properties to the console and verify the change
        Console.WriteLine("Edited properties of <{0}>, {1}", fullName);
        foreach(PUSProperty prop in bag.Properties) {
            Console.WriteLine("Property: <{0}>, {1}", prop.Name, prop.Value);
        }
        // Once we are done see the output, hit enter to continue
        Console.WriteLine("Press any key to exit...");
        Console.ReadLine();
    }
}
```

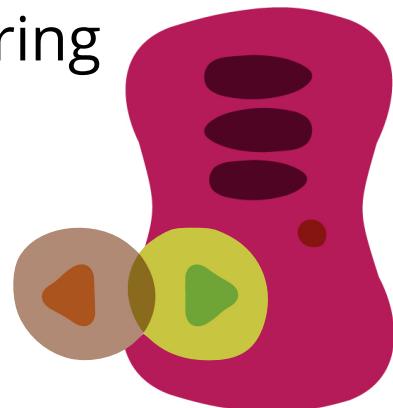
Data & code are symbiotic.

# DB Evolution & Deployment

scripting all db changes incrementally

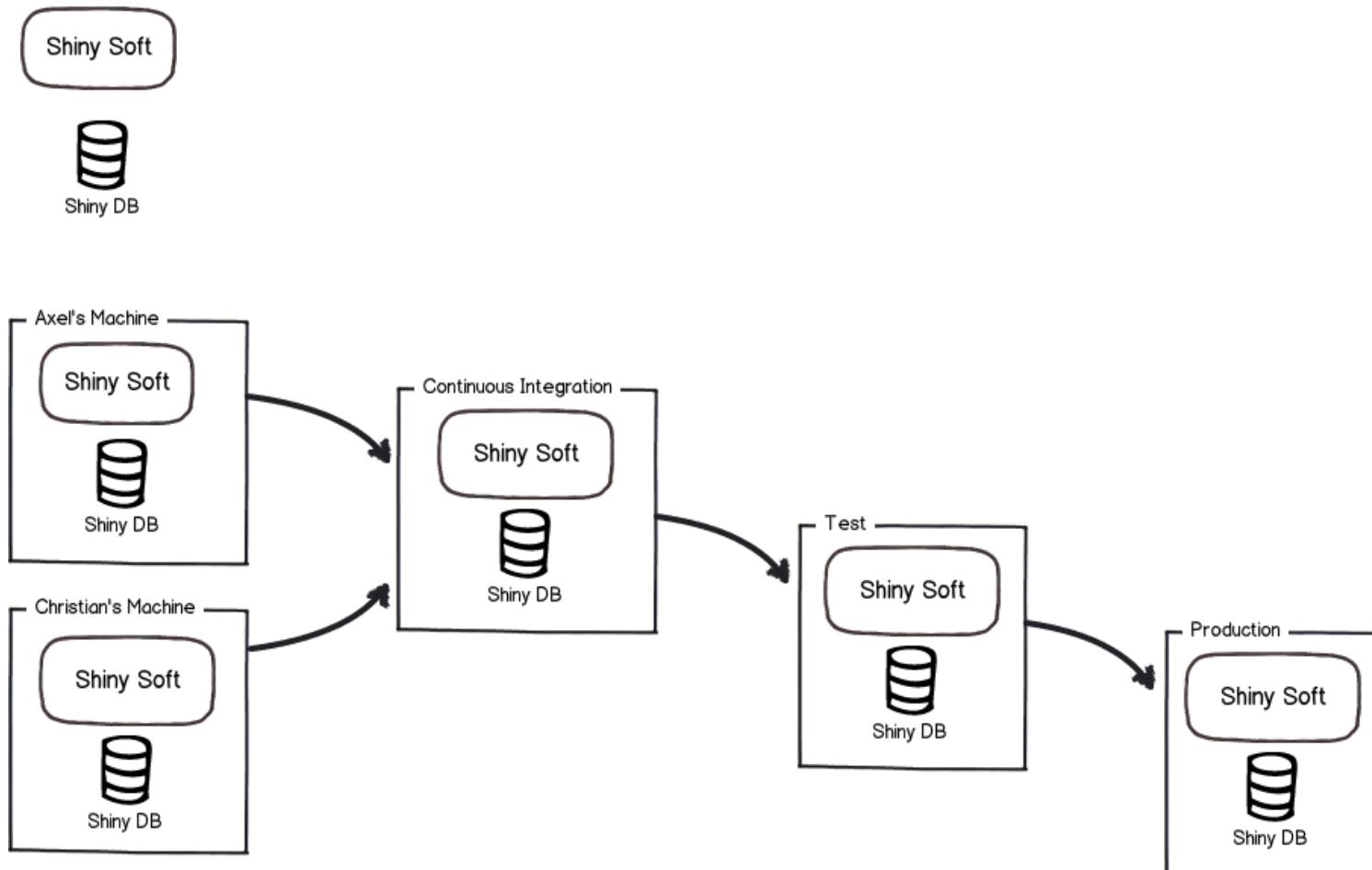


db refactoring



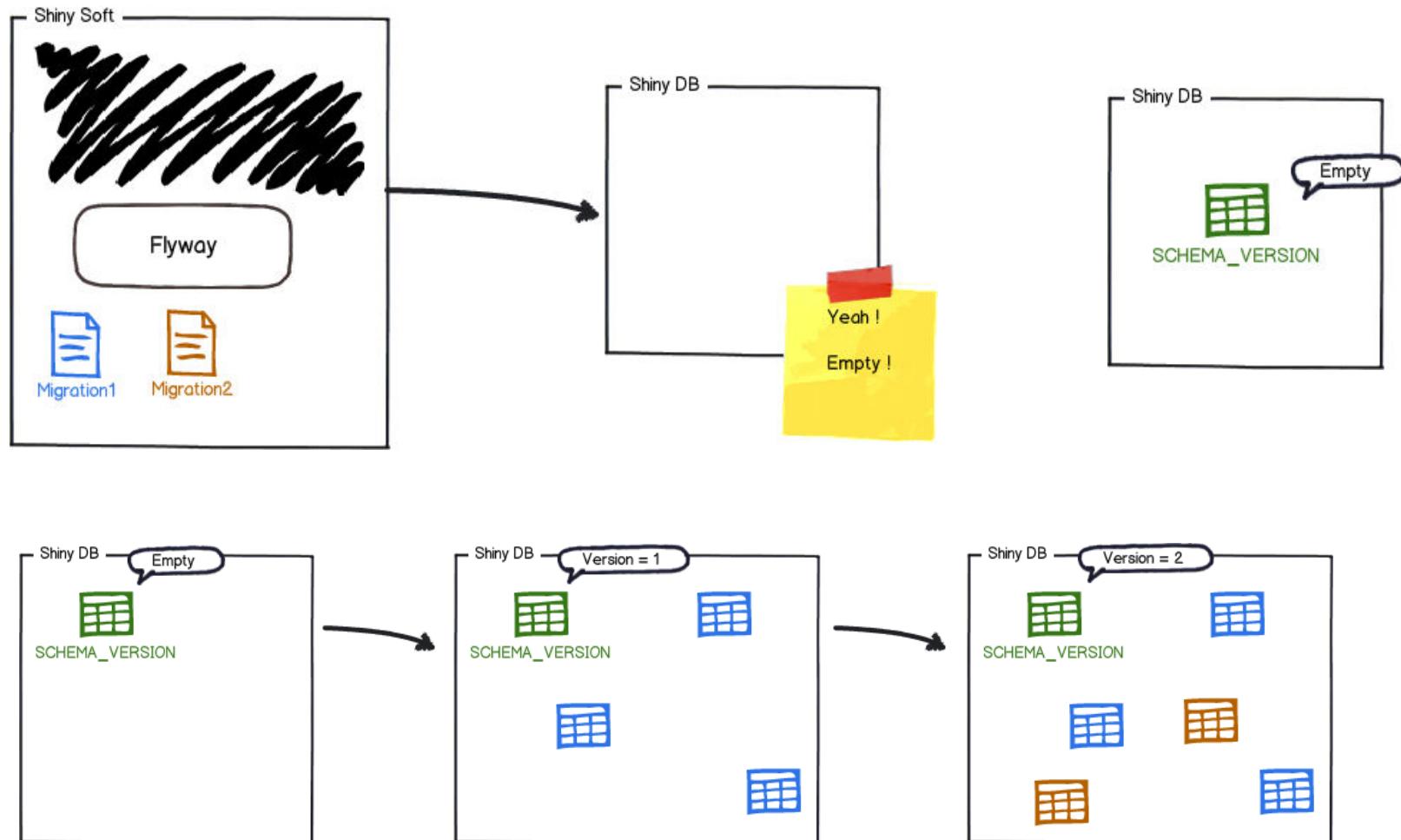
decouple db migration  
from app migration

# DbDeploy Pattern

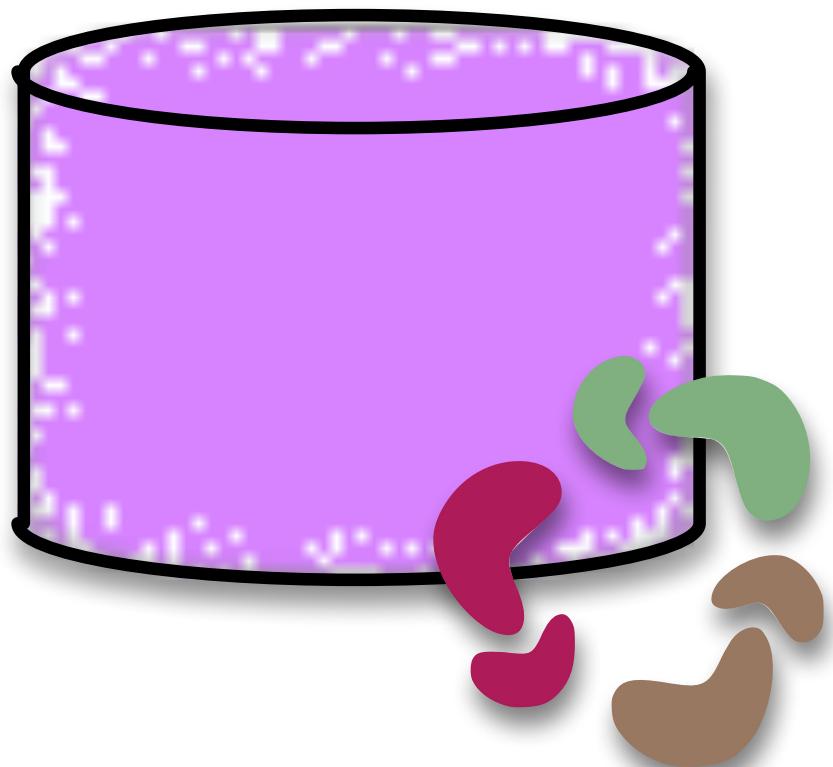


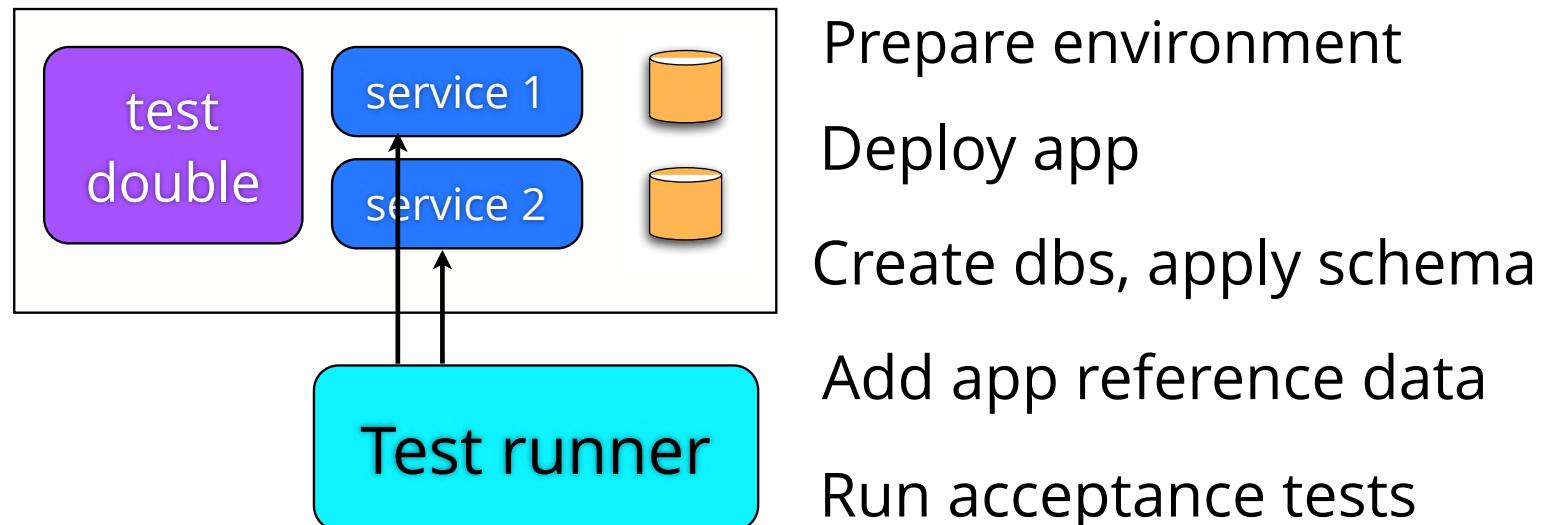
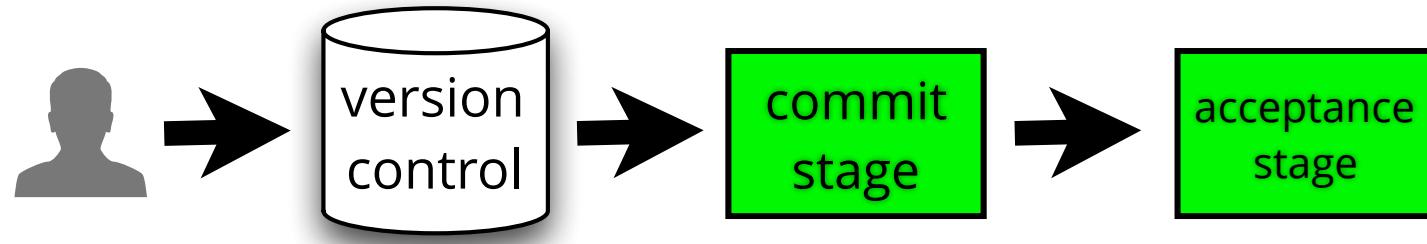


<http://flywaydb.org/>



# Continuous Integration for Databases





# For DB CI We Need To:

start with a clean database

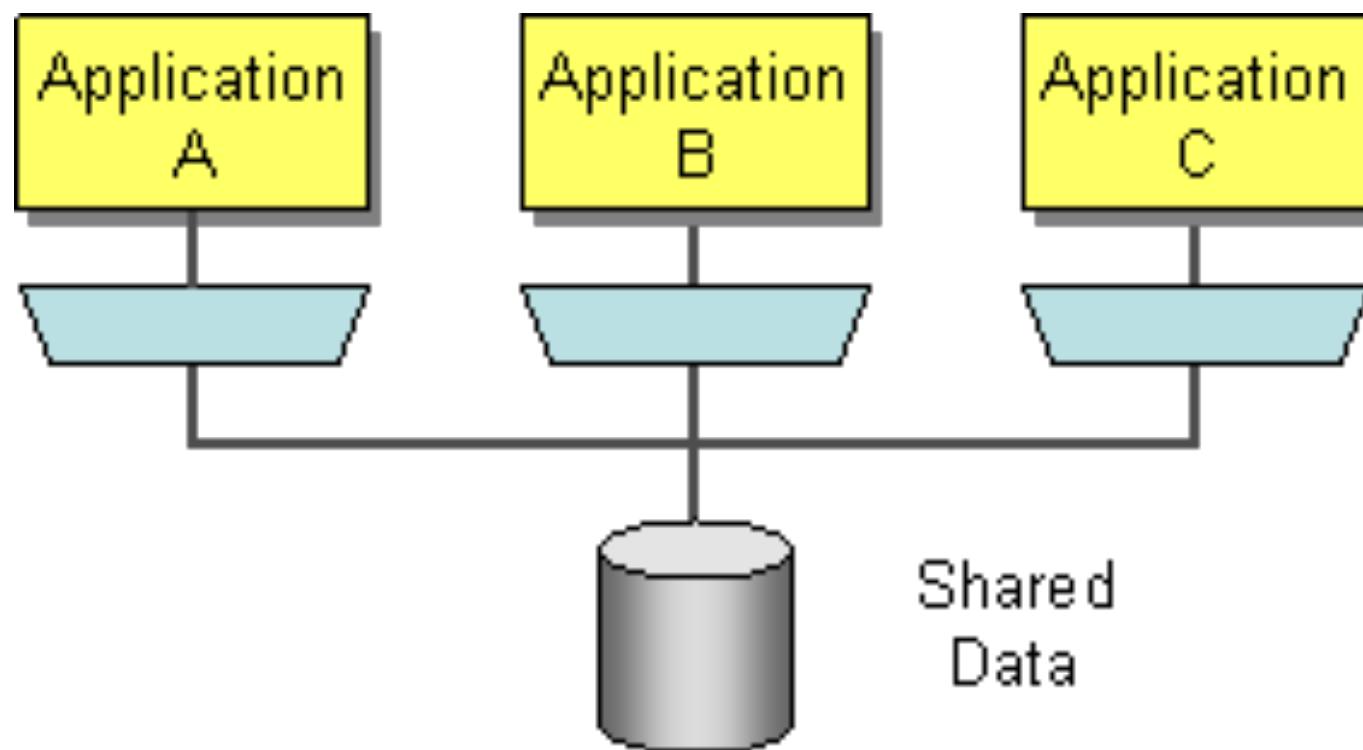


apply changes incrementally

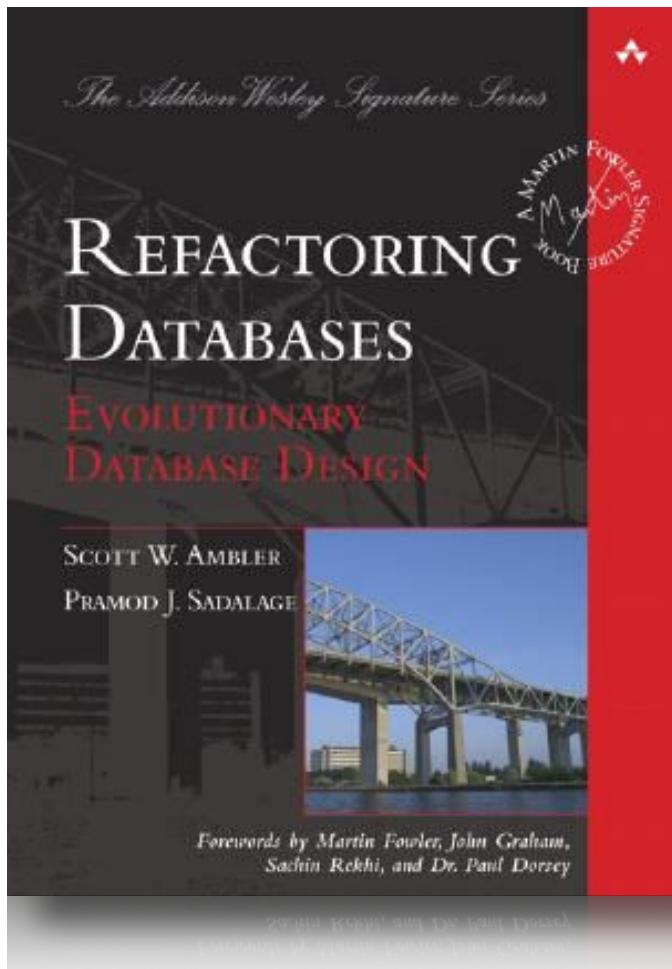
use the same process everywhere

be comprehensive in change management

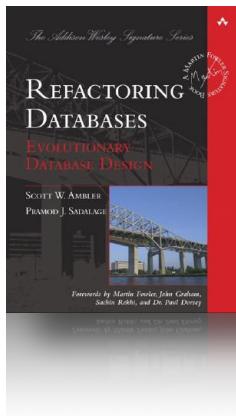
# Shared-database Integration



# Refactoring Databases

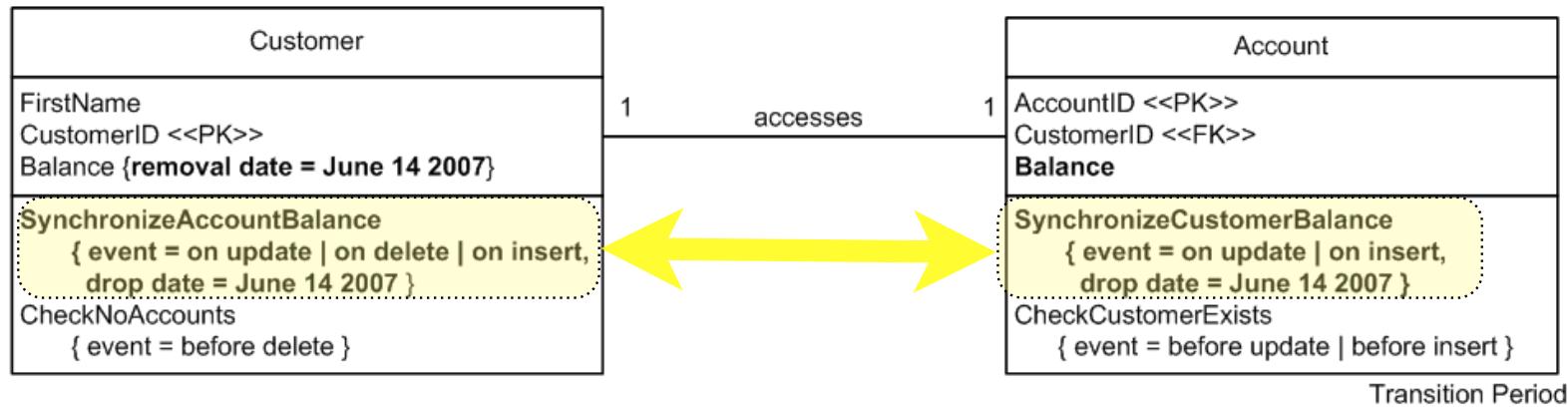
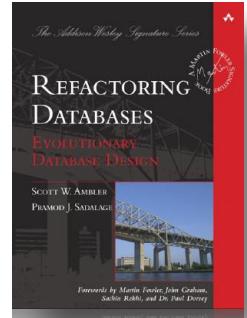


# Move Column Refactoring



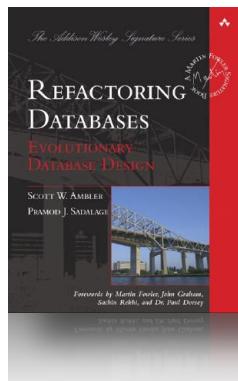
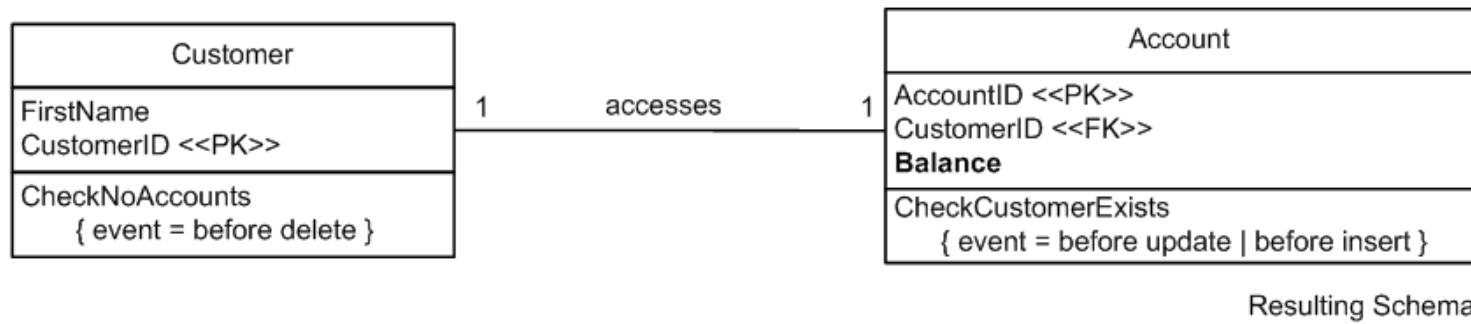
Original Schema

# Transition Period

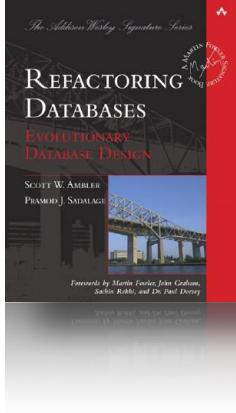


## Move Column Refactoring

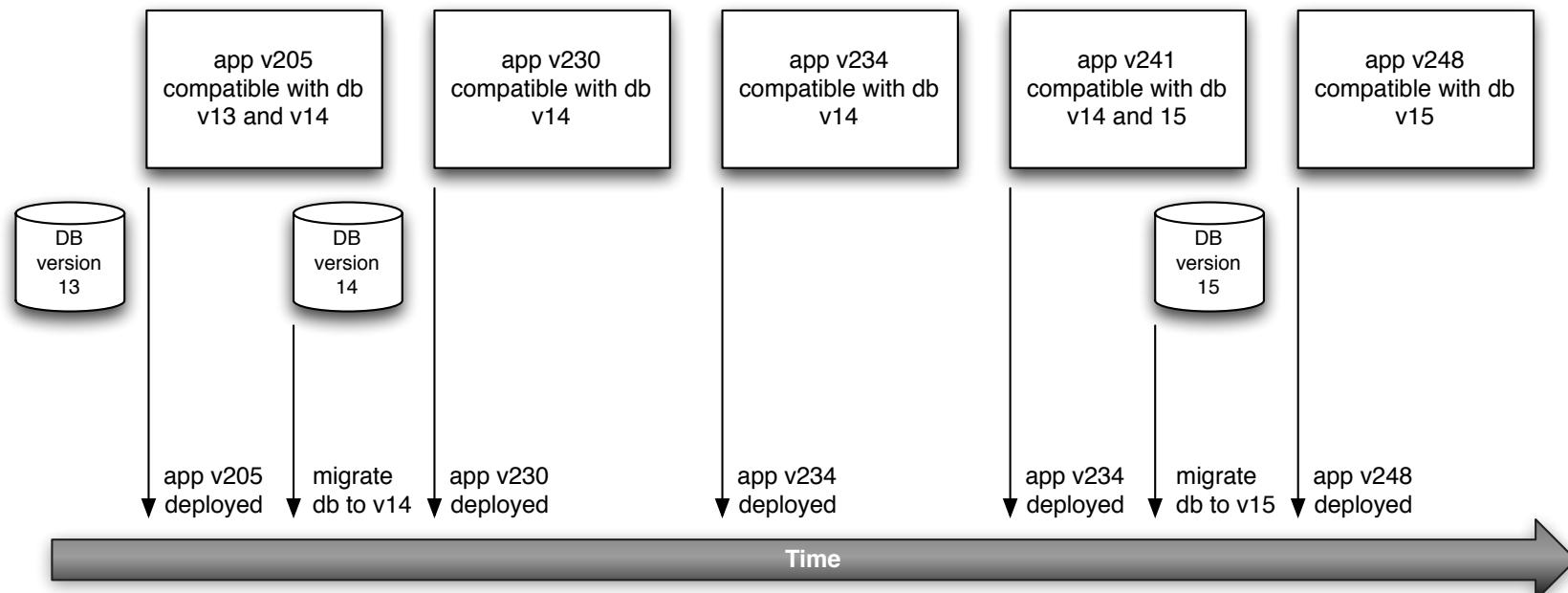
# Ending Schema



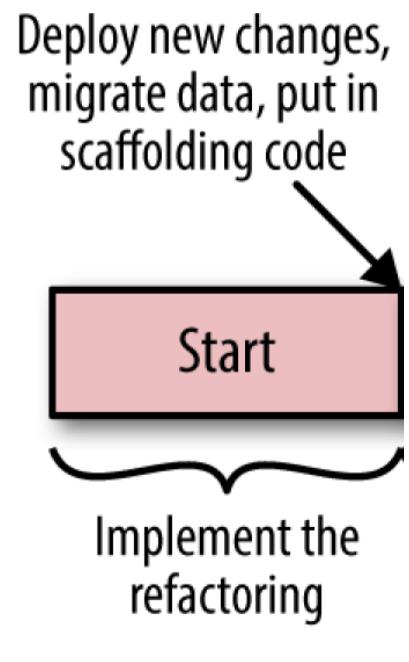
# Move Column Refactoring



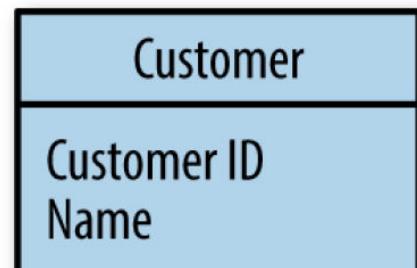
# Decouple DB Updates: the Expand/contract Pattern



# Expand/Contract Pattern



# Change 'name' to 'firstname' & 'lastname'

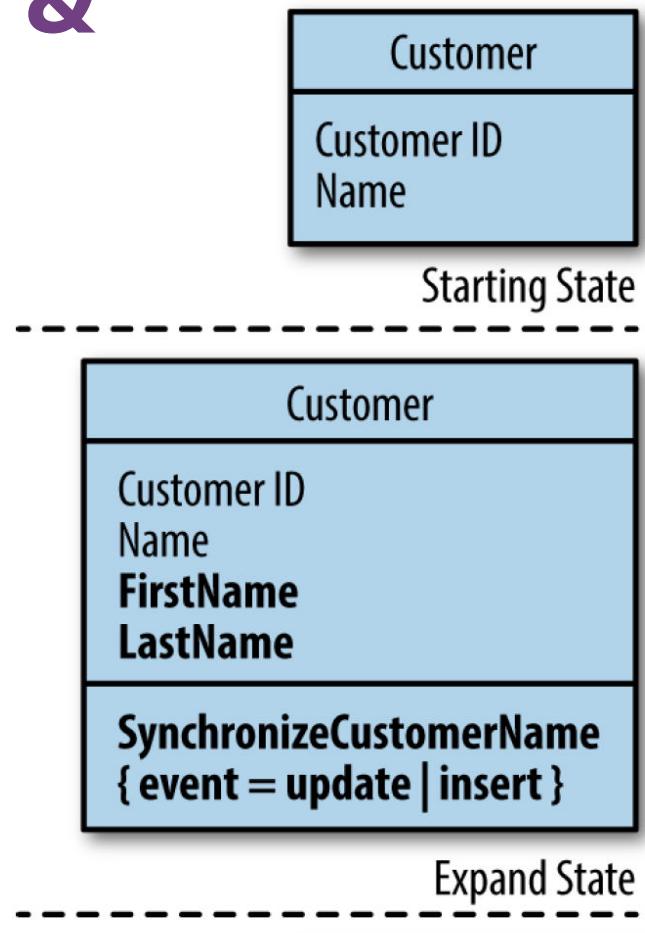


Starting State

Start

name = "Pramod Sadalage"

# Change 'name' to 'firstname' & 'lastname'



**Start**

name = "Pramod Sadalage"

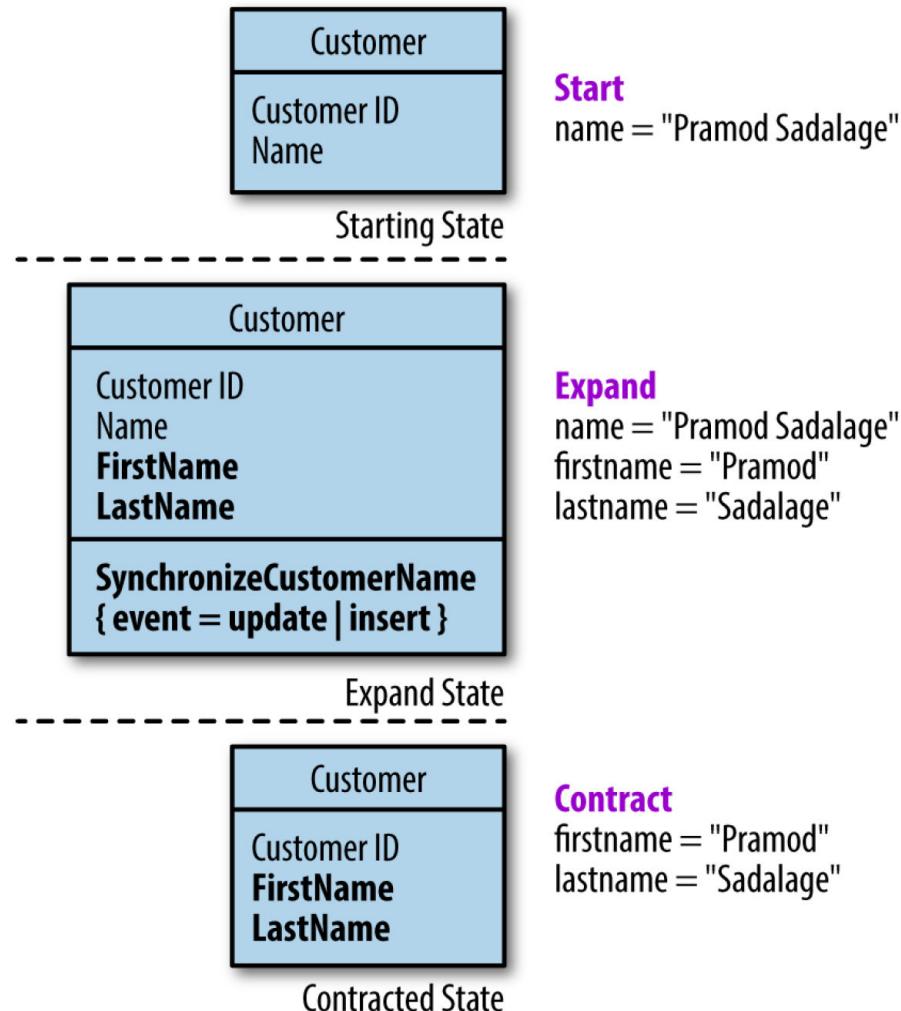
**Expand**

name = "Pramod Sadalage"

firstname = "Pramod"

lastname = "Sadlage"

# Change 'name' to 'firstname' & 'lastname'



# #1: integration points, legacy data

```
ALTER TABLE customer ADD firstname VARCHAR2(60);
ALTER TABLE customer ADD lastname VARCHAR2(60);
ALTER TABLE customer DROP COLUMN name;
```

## #2: integration points, legacy data

```
ALTER TABLE Customer ADD firstname VARCHAR2(60);
ALTER TABLE Customer ADD lastname VARCHAR2(60);
UPDATE Customer set firstname = extractfirstname (name);
UPDATE Customer set lastname = extractlastname (name);
ALTER TABLE customer DROP COLUMN name;
```

# #3: integration points, legacy data

```
ALTER TABLE Customer ADD firstname VARCHAR2(60);
ALTER TABLE Customer ADD lastname VARCHAR2(60);

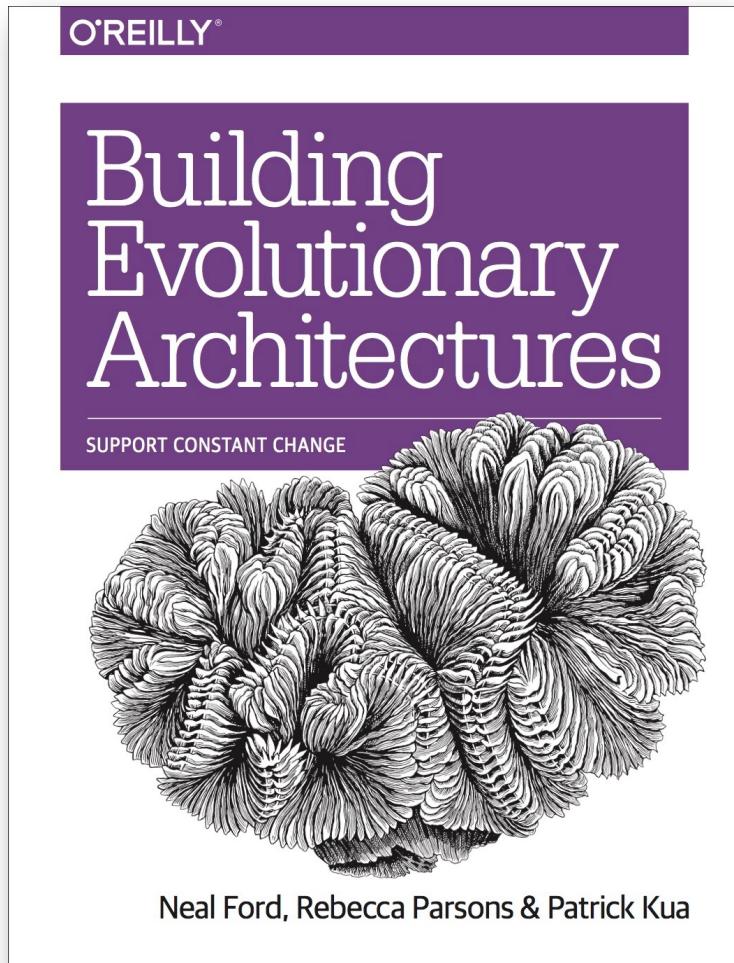
UPDATE Customer set firstname = extractfirstname (name);
UPDATE Customer set lastname = extractlastname (name);

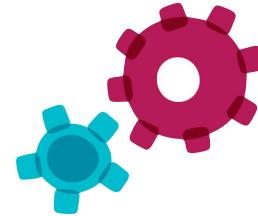
CREATE OR REPLACE TRIGGER SynchronizeName
BEFORE INSERT OR UPDATE
ON Customer
REFERENCING OLD AS OLD NEW AS NEW
FOR EACH ROW
BEGIN

IF :NEW.Name IS NULL THEN
    :NEW.Name := :NEW.firstname||' '||:NEW.lastname;
END IF;
IF :NEW.name IS NOT NULL THEN
    :NEW.firstname := extractfirstname(:NEW.name);
    :NEW.lastname := extractlastname(:NEW.name);
END IF;
END;
```

# Building Evolutionary Architectures

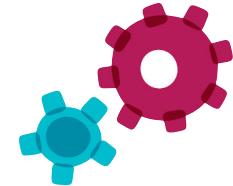
BUILDING EVOLVABLE  
ARCHITECTURES



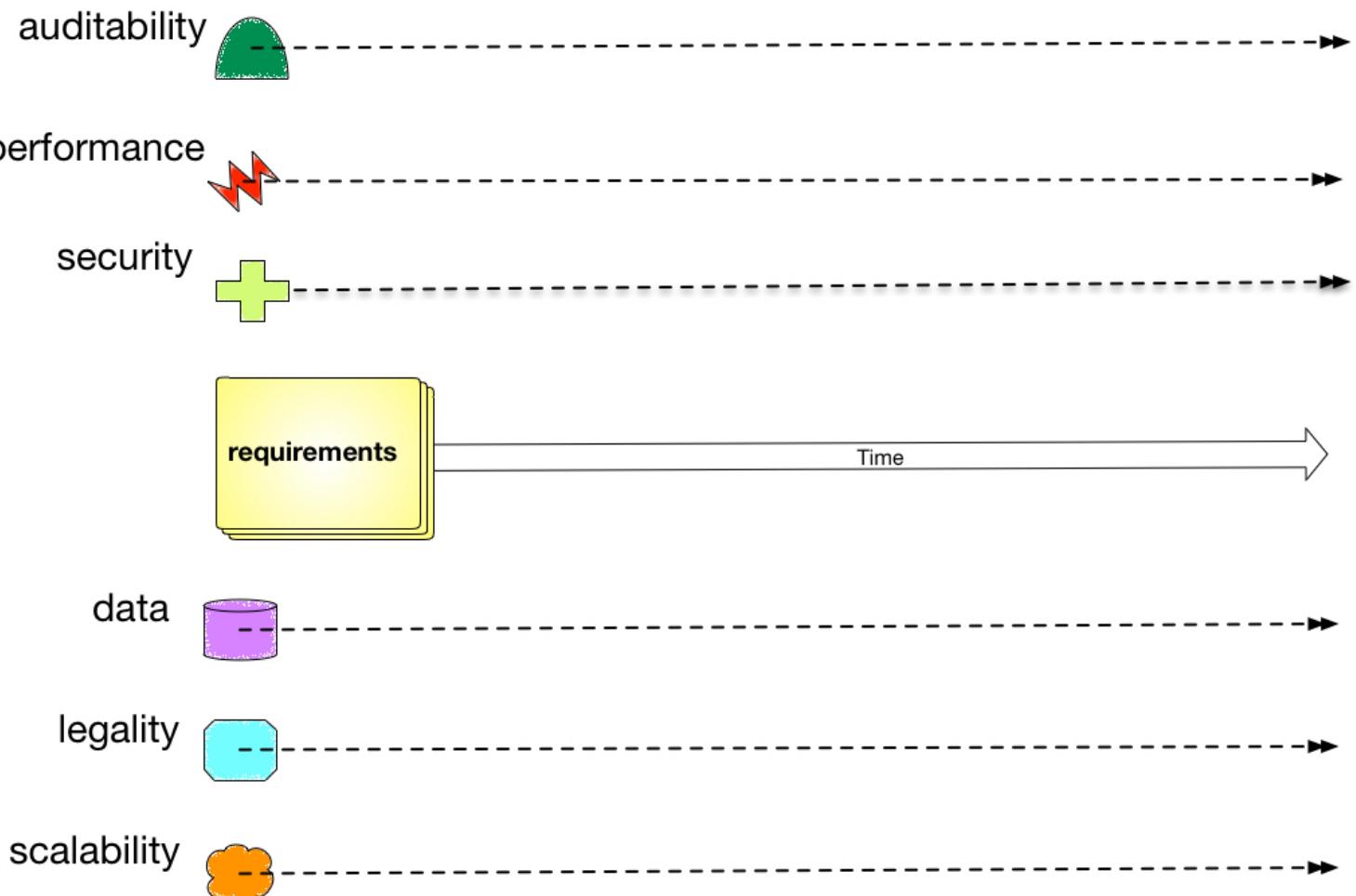


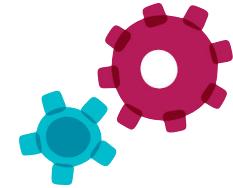
# Mechanics

1. Identify dimensions



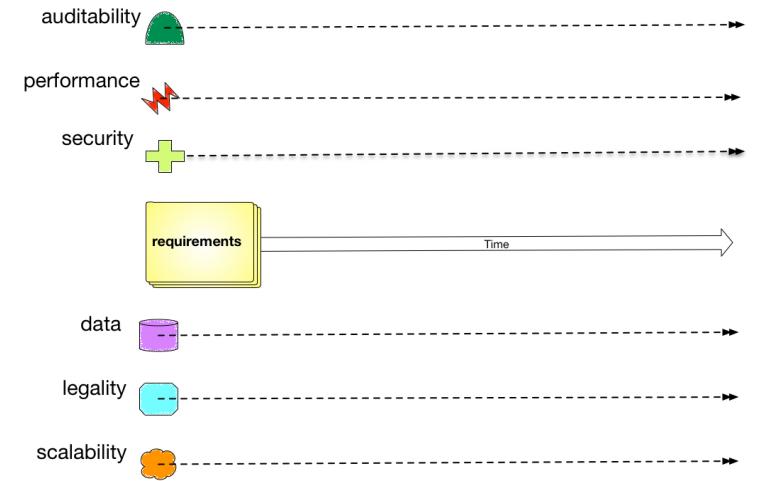
# 1. Identify dimensions

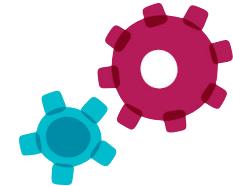




# prioritization

## 1. Identify dimensions

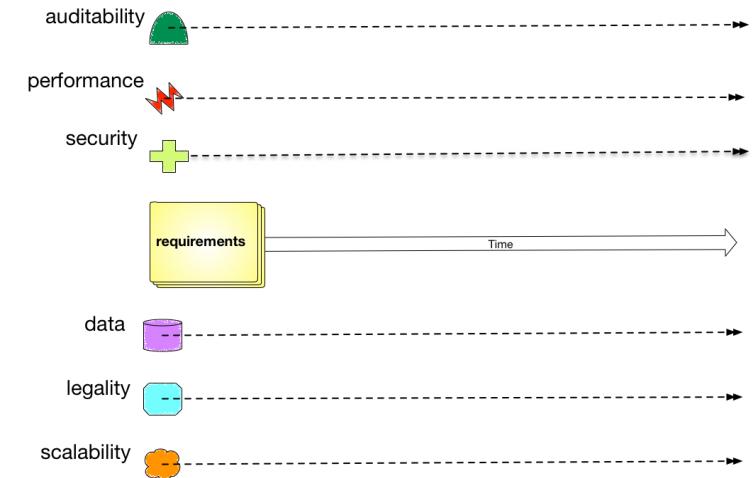


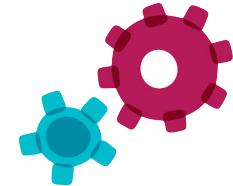


# 1. Identify dimensions

prioritization

cost to implement & maintain



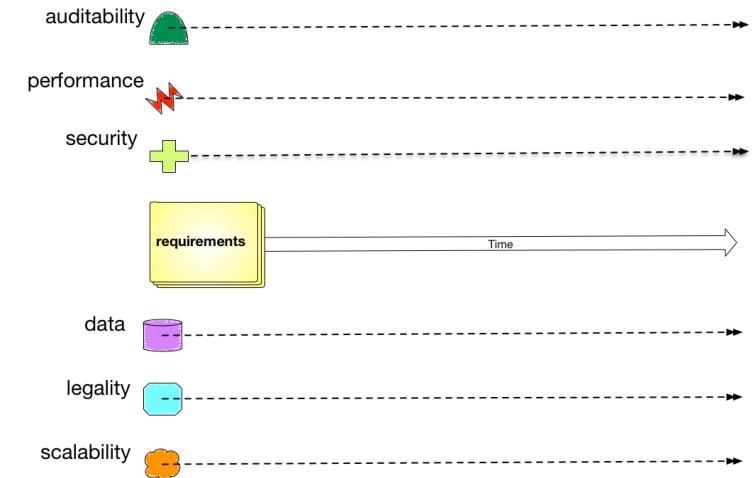


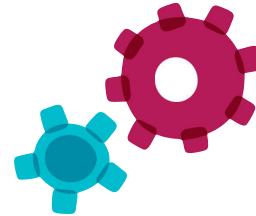
# 1. Identify dimensions

prioritization

cost to implement & maintain

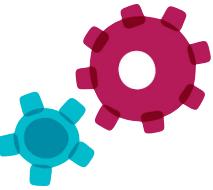
change variable “fit it” cost to constant  
maintenance cost



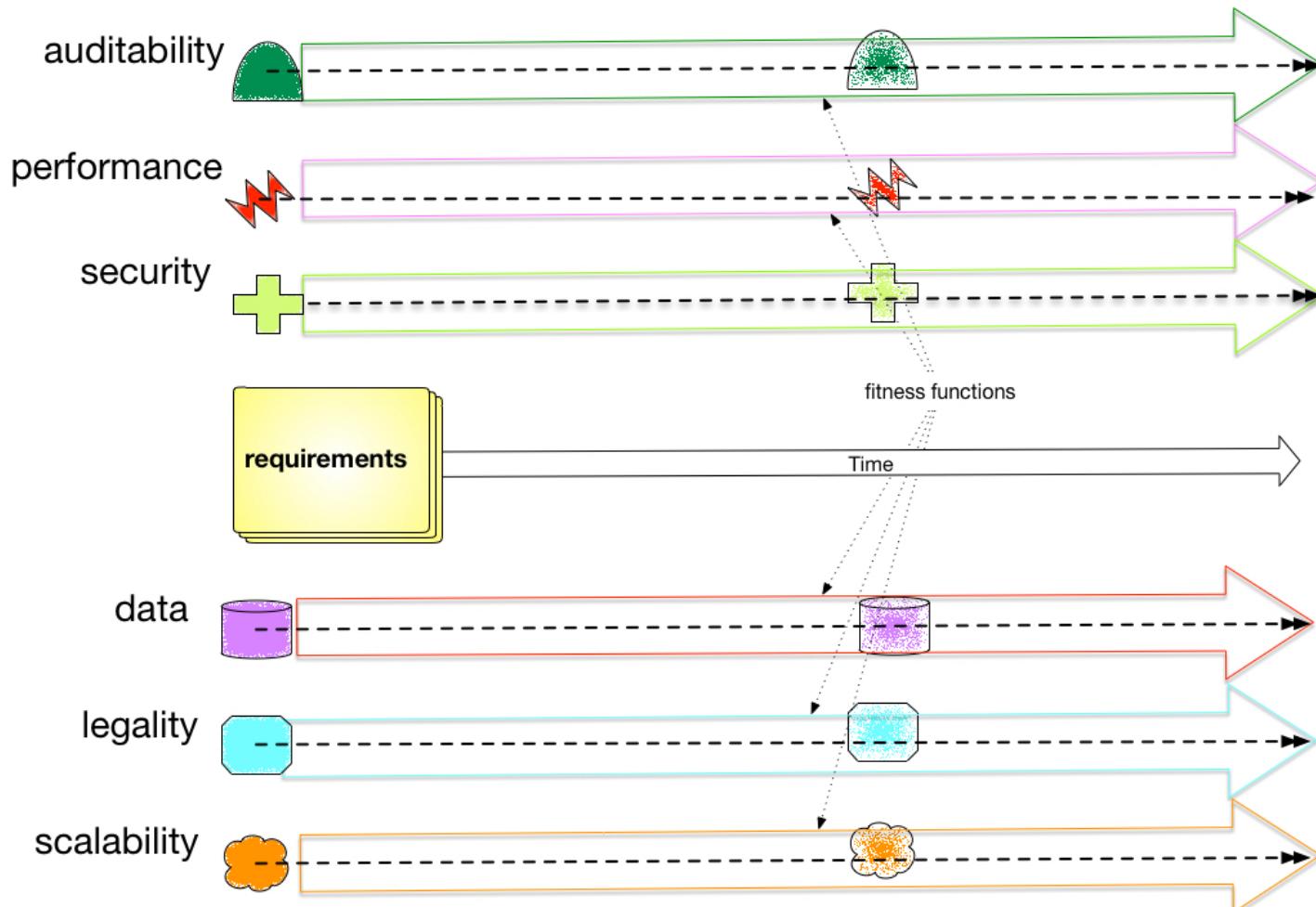


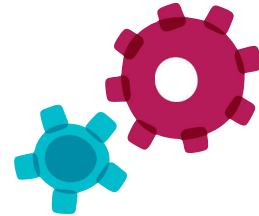
# Mechanics

1. Identify dimensions
2. Define fitness function(s)



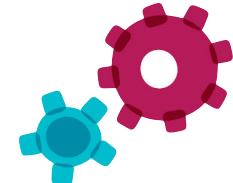
## 2. Define Fitness Function(s)



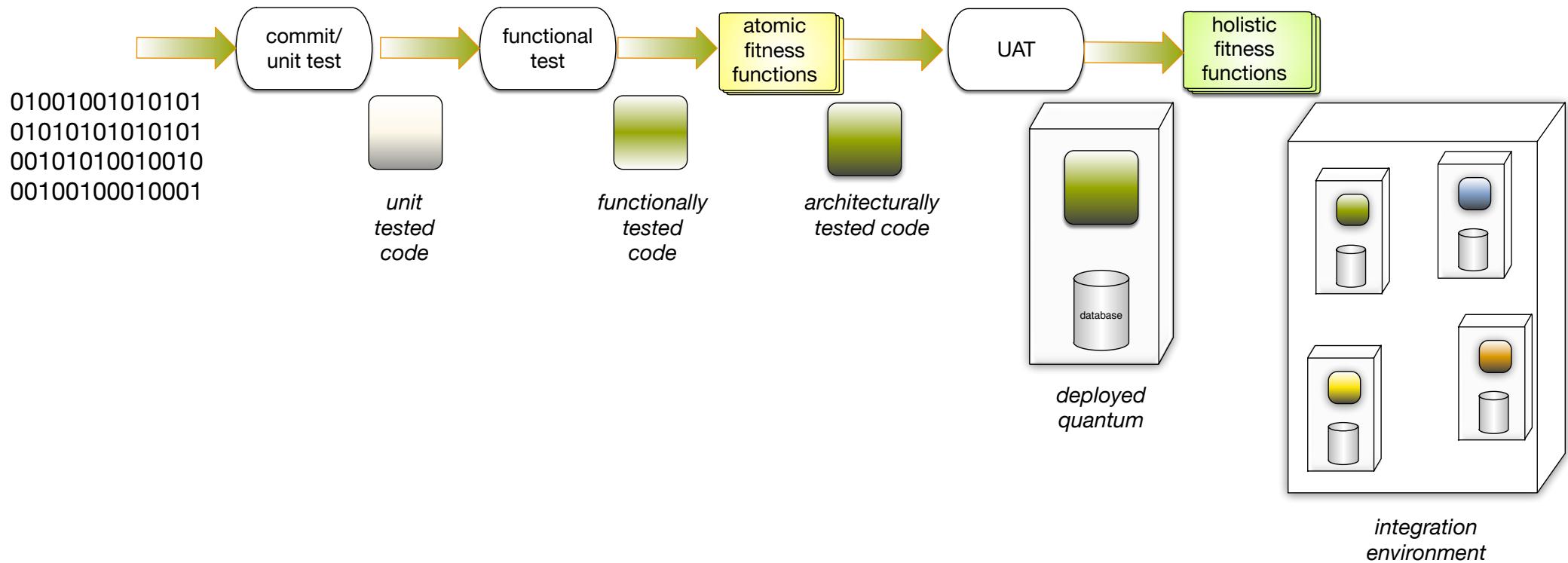


# Mechanics

1. Identify dimensions
2. Define fitness function(s)
3. Use deployment pipelines and/or continuous fitness functions

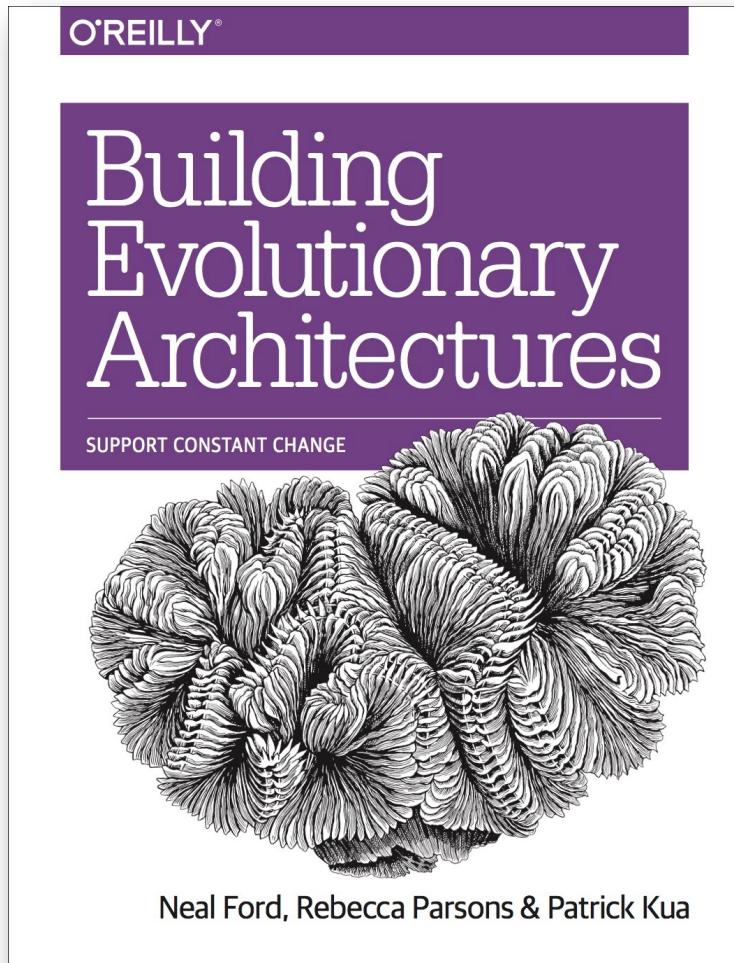


### 3. Use deployment pipelines and/or continuous fitness functions

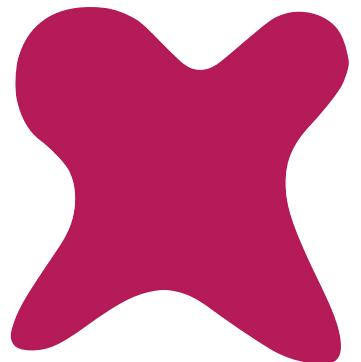


# Building Evolutionary Architectures

## GUIDELINES FOR EVOLUTIONARY ARCHITECTURES



# Remove Needless Variables



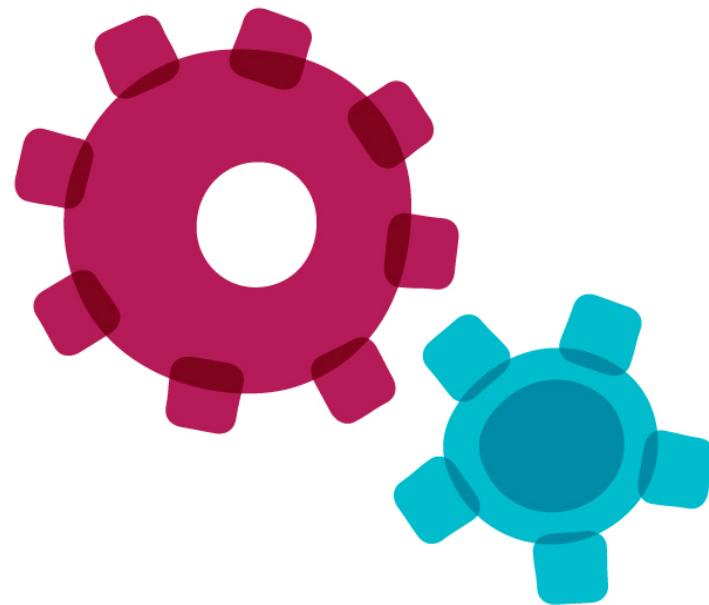
# Remove Needless Variables





# Remove Needless Variables

# Remove Needless Variables



Knight Capital

DOUG SEVEN

Something can be learned in the course of observing things

HOME IOT WORKSHOP PRODUCT MANAGEMENT ABOUT

POSTS COMMENTS

You are here: Home / DevOps / Knightmare: A DevOps Cautionary Tale

## Knightmare: A DevOps Cautionary Tale

APRIL 17, 2014 BY D7 37 COMMENTS

★★★★★ 70 Votes

I was speaking at a conference last year on the topics of DevOps, Configuration as Code, and Continuous Delivery and used the following story to demonstrate the importance making deployments fully automated and repeatable as part of a DevOps/Continuous Delivery initiative. Since that conference I have been asked by several people to share the story through my blog. This story is true – this really happened. This is my telling of the story based on what I have read (I was not involved in this).

This is the story of how a company with nearly \$400 million in assets went bankrupt in 45-minutes because of a failed deployment.

### Background

Knight Capital Group is an American global financial services firm engaging in [market making](#), electronic execution, and institutional sales and trading. In 2012 Knight was the largest trader in US equities with market share of around 17% on each the NYSE and NASDAQ. Knight's Electronic

DOUG SEVEN



SEARCH

Search this website... Search

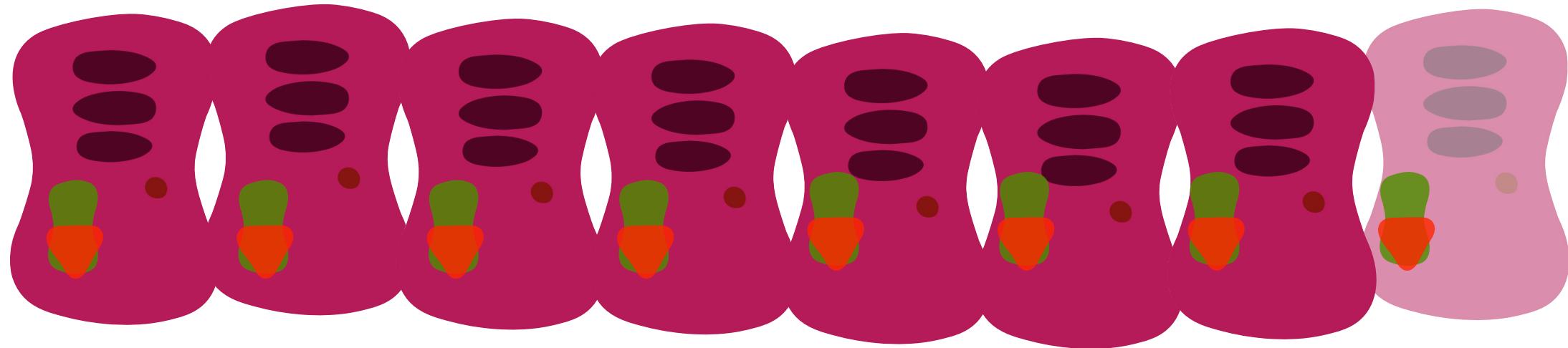
RECENT POSTS:

- IoT Workshop: Lesson 3 – Input Controls Output
- Arduino: Reading Analog Voltage
- Arduino 'Hello, World!' – Sending Digital Output

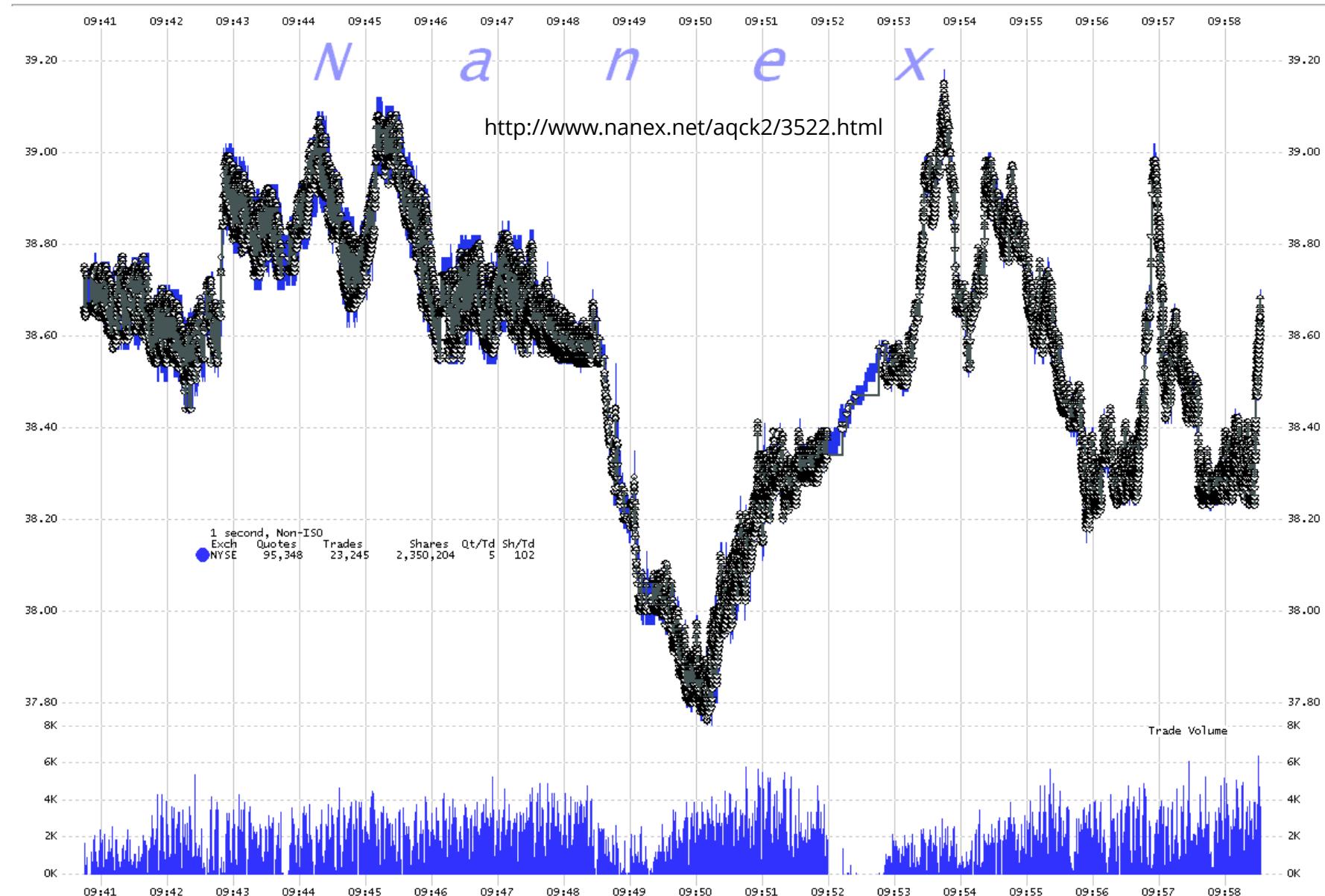
"bankrupt in 45 minutes"

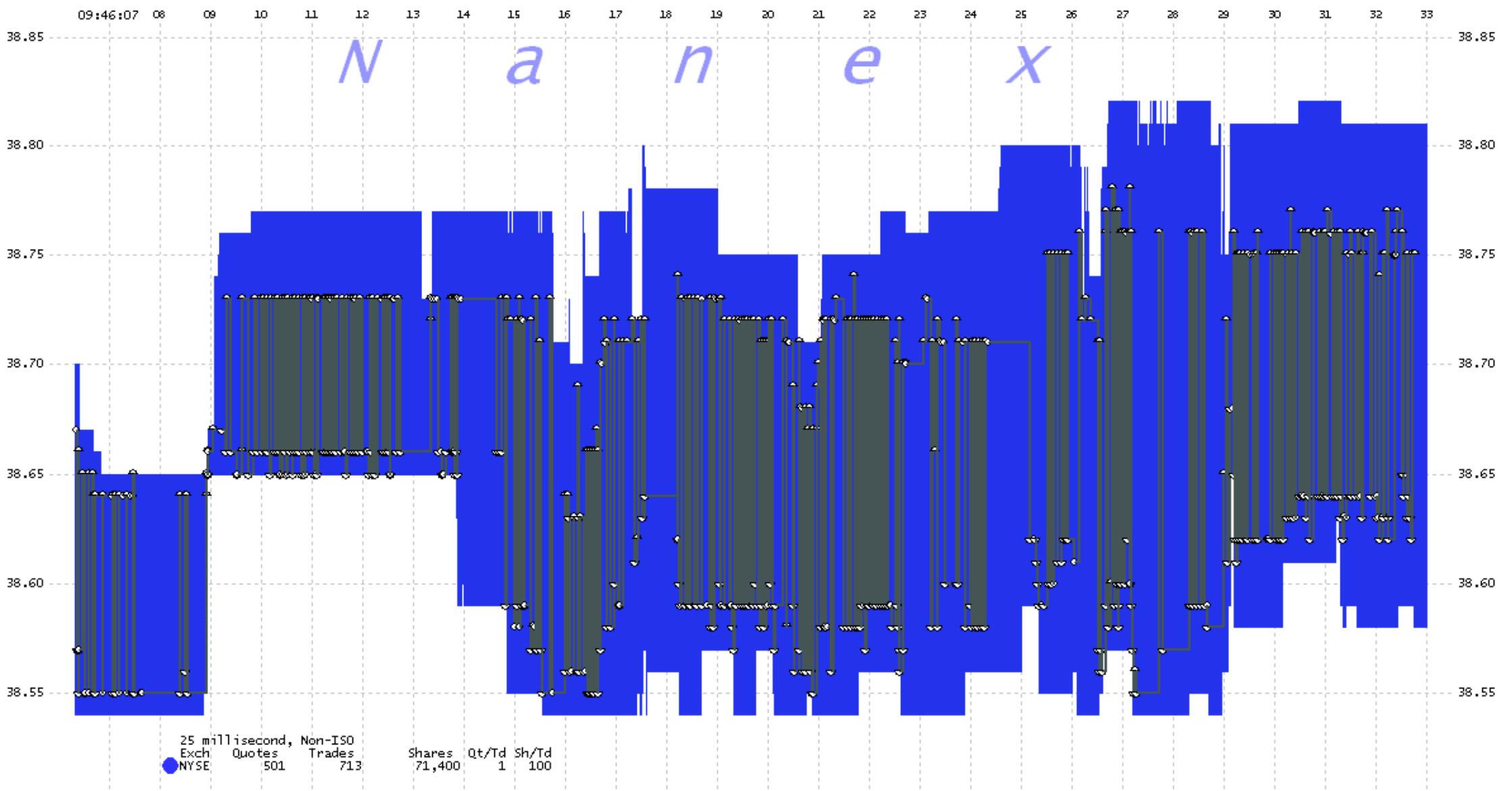
# SMARS

PowerPeg



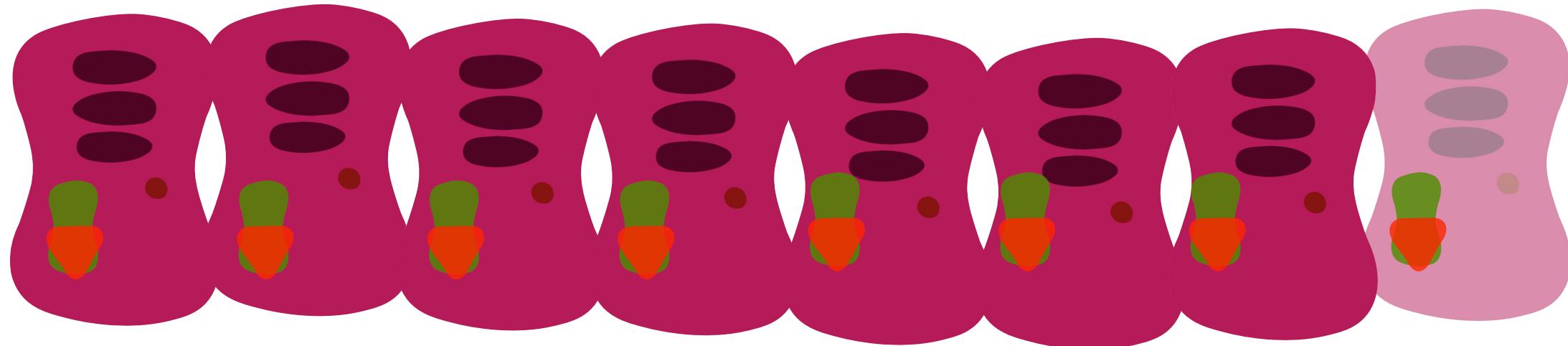
**1. EXC One second interval chart. Circles are trades, the blue coloring is the NYSE bid and ask which is mostly covered by gray lines that connect the trades.**





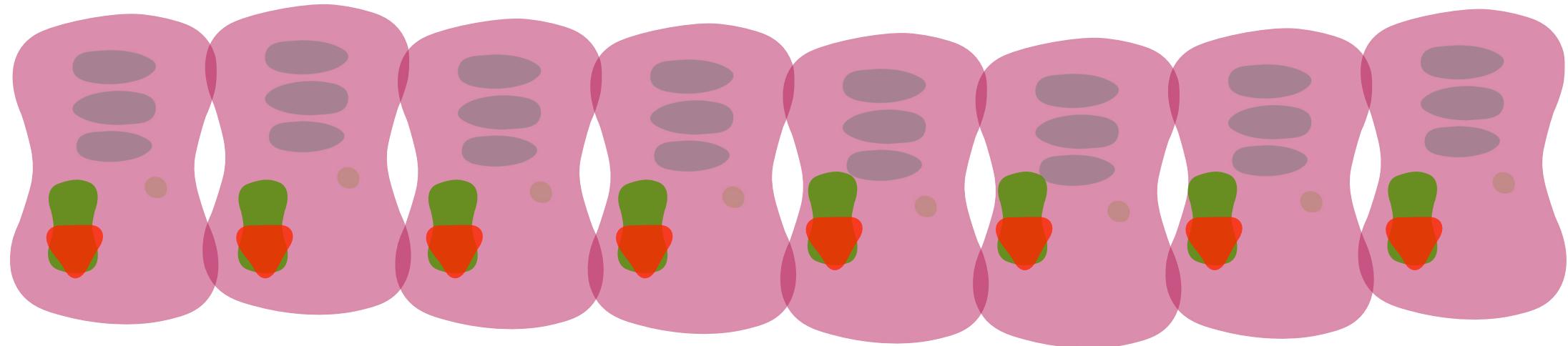
# SMARS

PowerPeg



# SMARS

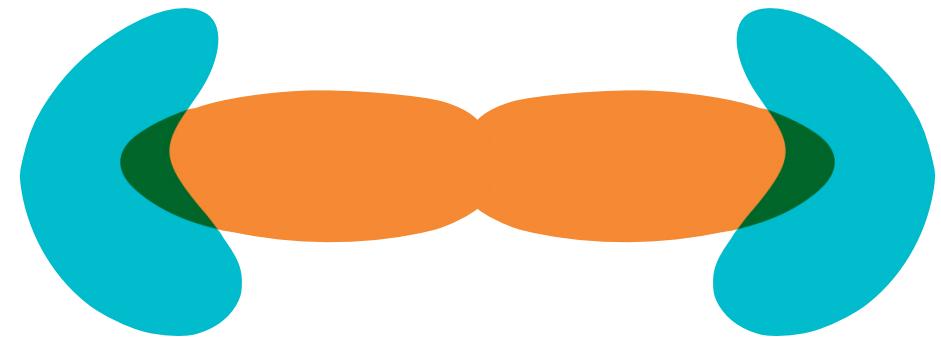
PowerPeg



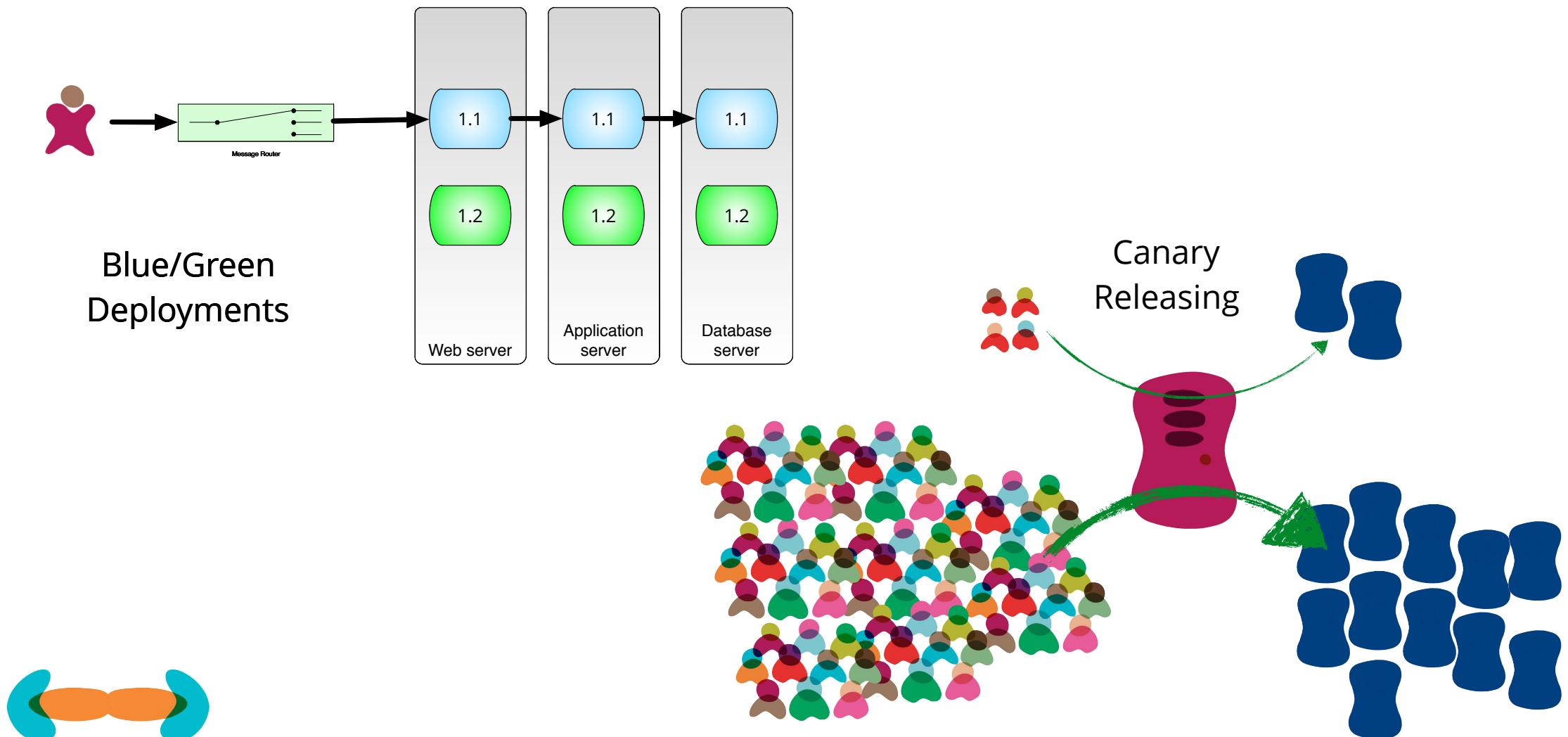


**Identify and remove  
needless variability.**

# Make Decisions Reversible

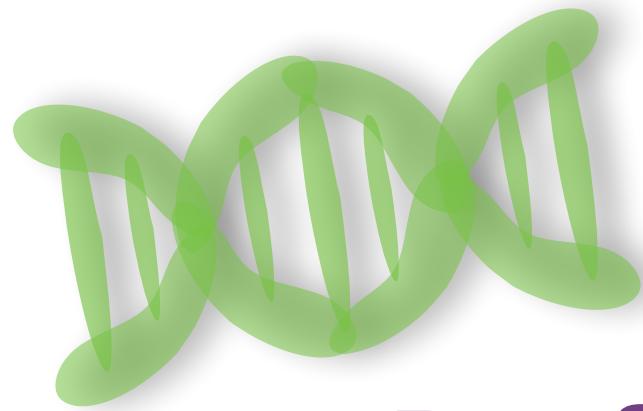


# Make Decisions Reversible

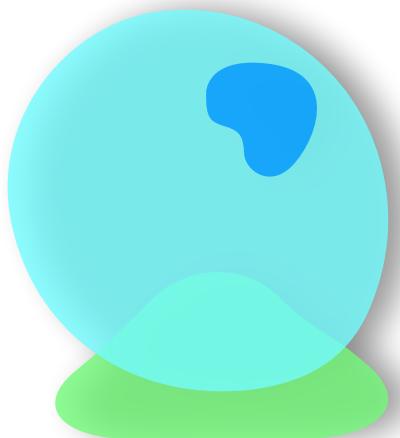




**Make as many decisions as  
possible reversible  
(without overengineering).**



**Prefer Evolvable over  
Predictable**



*...because as we know, there are known knowns; there are things we know we know.*

*We also know there are known unknowns; that is to say we know there are some things we do not know.*

*But there are also unknown unknowns—the ones we don't know we don't know.*

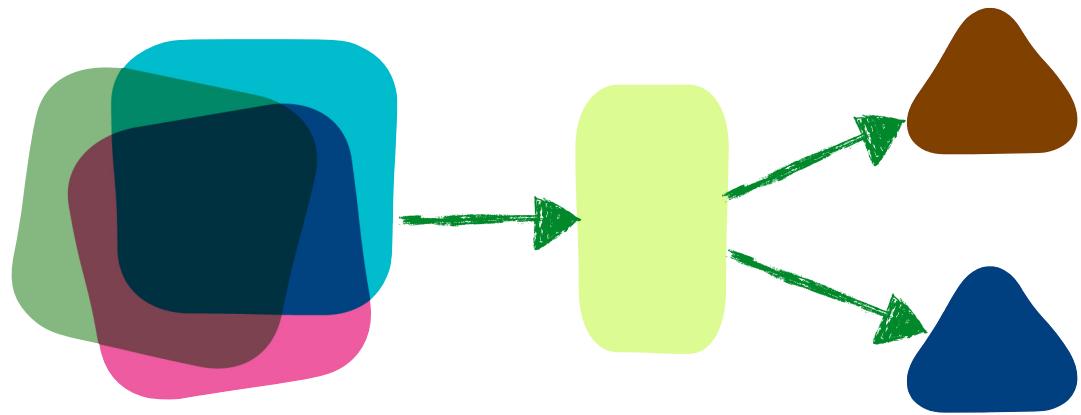
-former US Secretary of Defense Donald Rumsfeld

# unknown unknowns

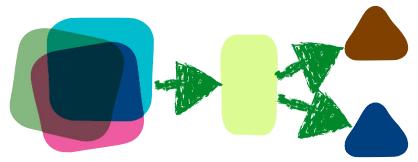
*All architectures become iterative because of unknown unknowns; agile just recognizes this and does it sooner.*

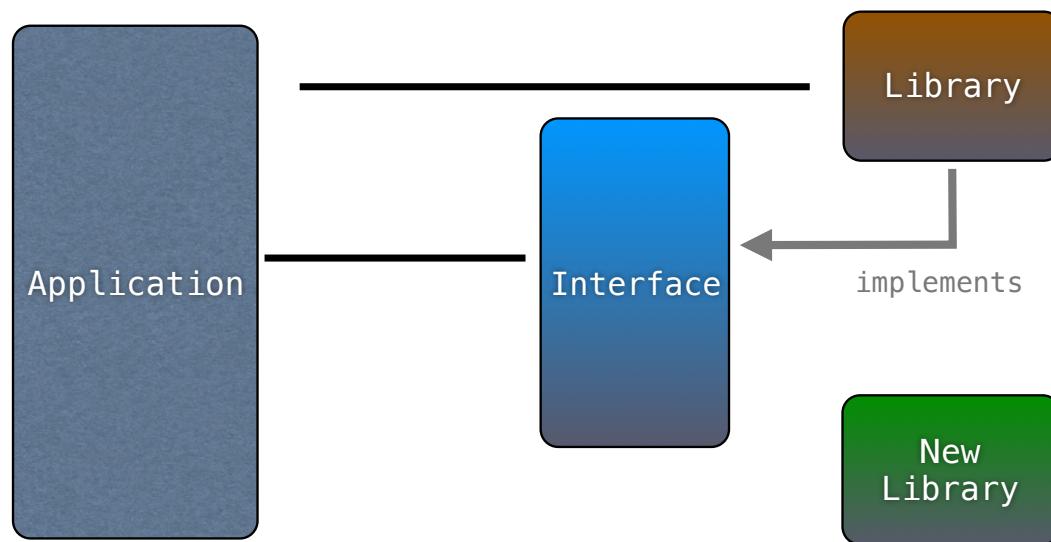
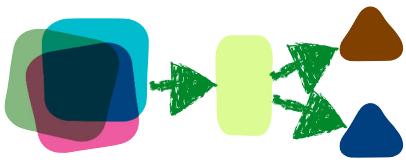
-Mark Richards

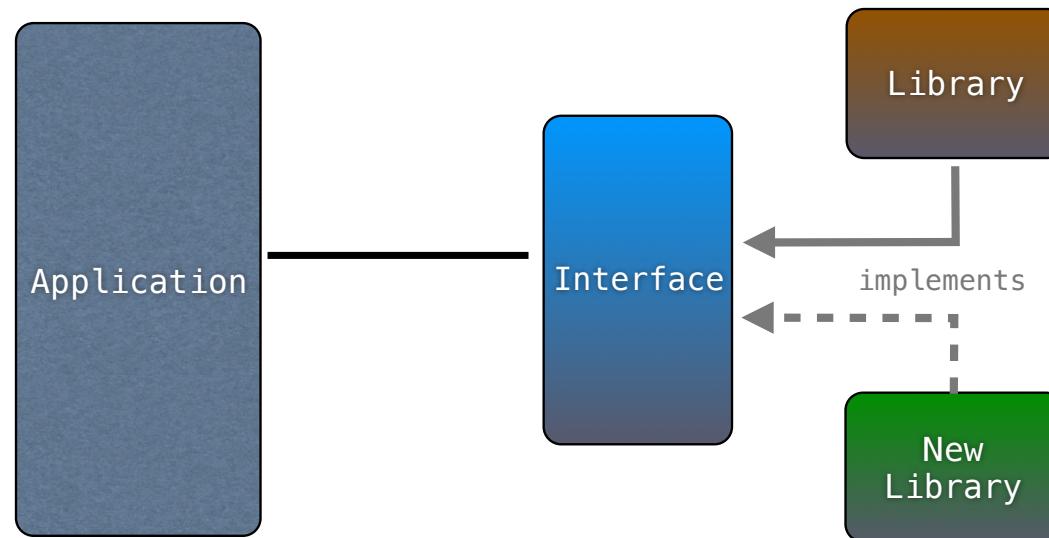


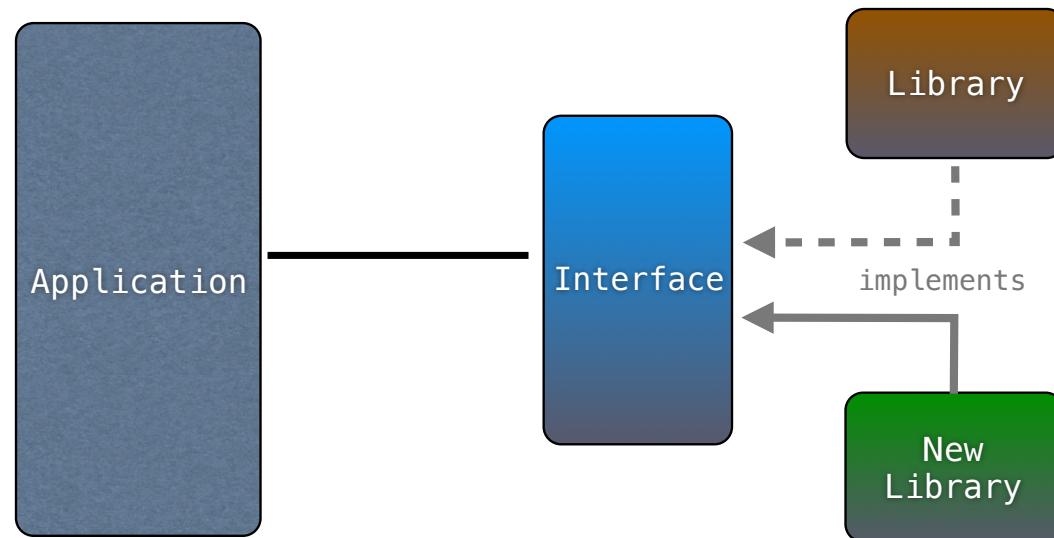
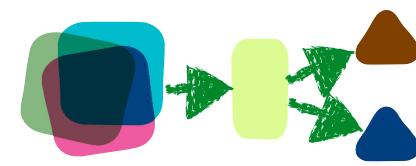


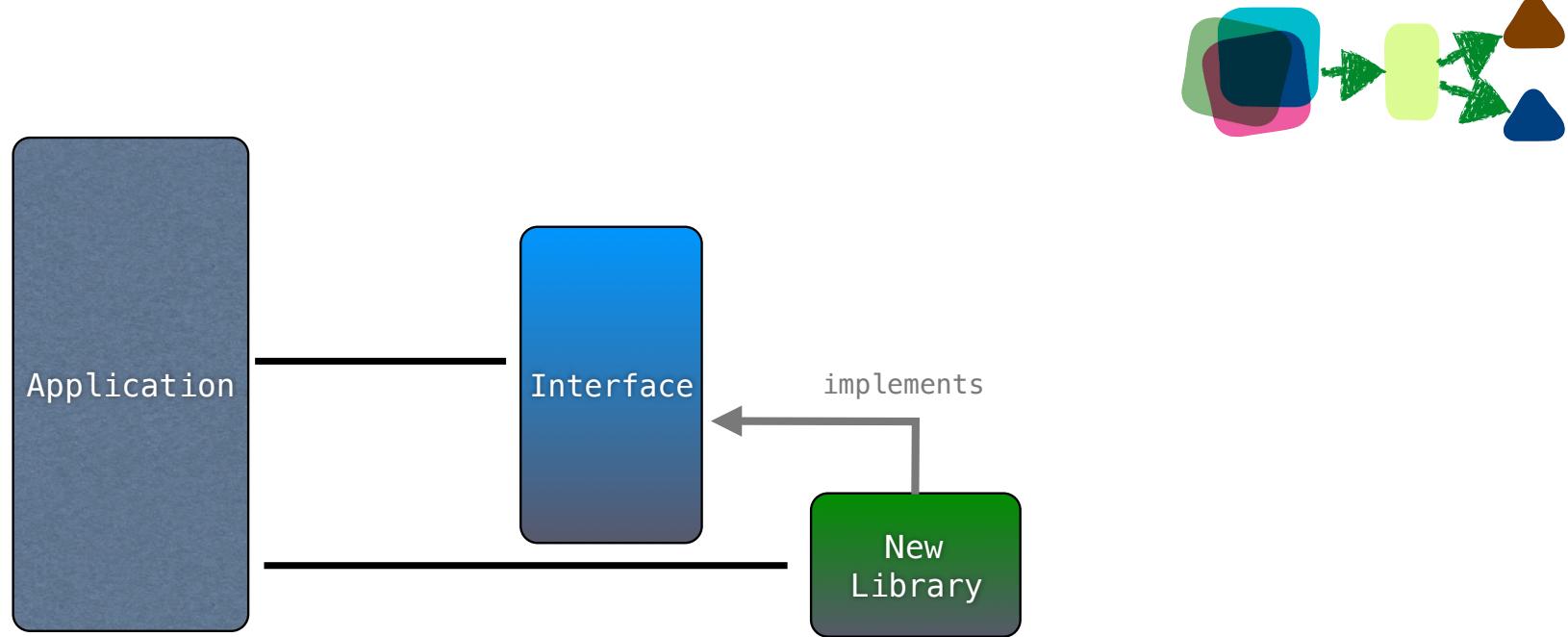
# Build Anti-corruption Layers



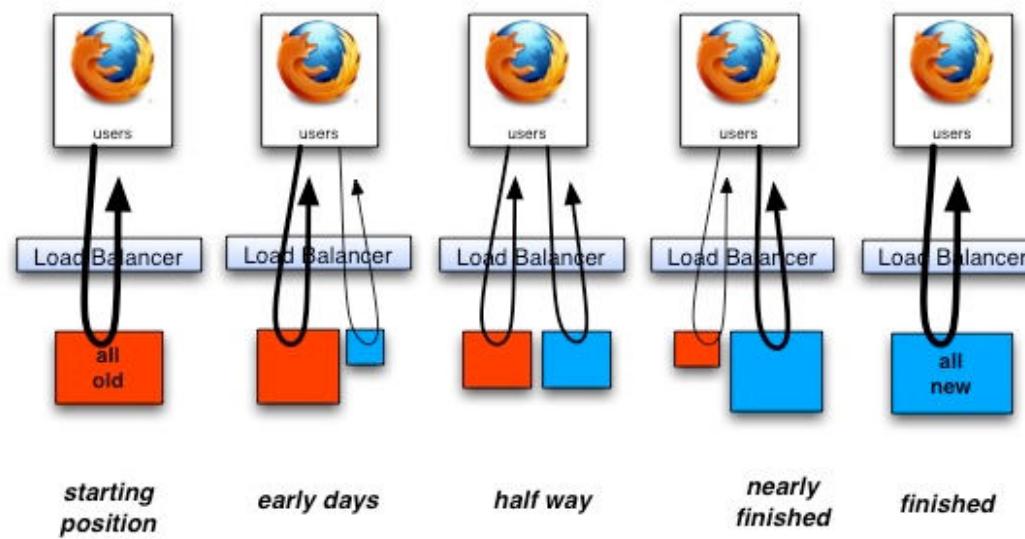








# Strangler Pattern

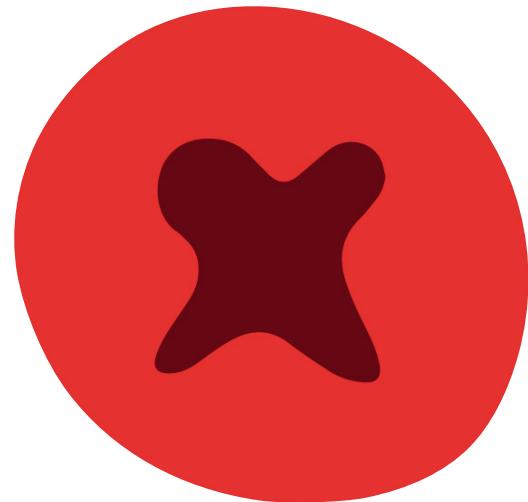


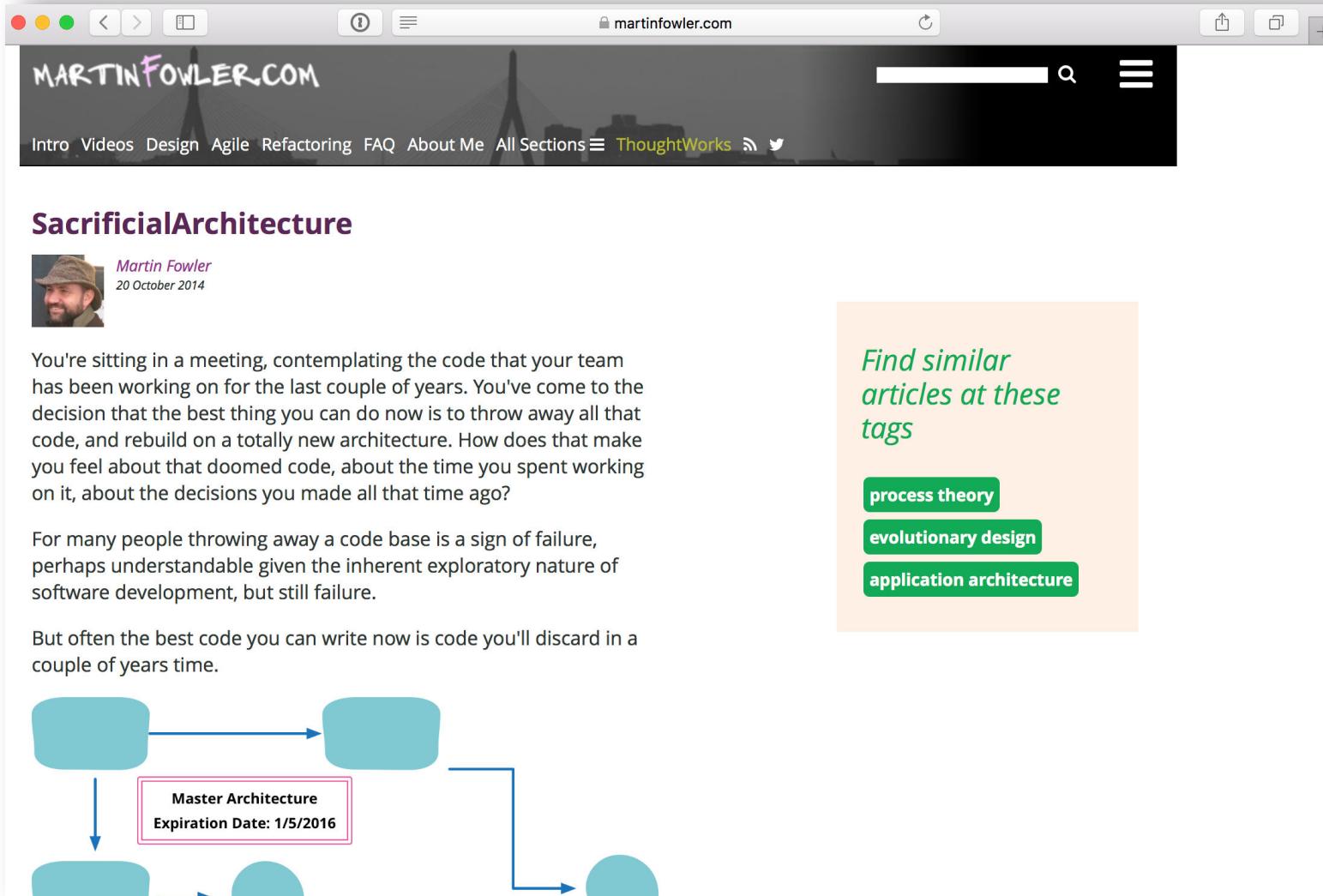
make something new that obsoletes a small percentage of something old

put them live together

rinse, repeat

# Build Sacrificial Architectures



A screenshot of a web browser displaying the Martin Fowler website at [martinfowler.com](https://martinfowler.com). The page title is "SacrificialArchitecture". The author is Martin Fowler, with a photo and the date 20 October 2014. The text discusses the decision to throw away old code and rebuild. A sidebar on the right suggests similar articles based on tags: process theory, evolutionary design, and application architecture. Below the main content is a diagram illustrating a flow from a master architecture to a new one.

**MARTINFOWLER.COM**

Intro Videos Design Agile Refactoring FAQ About Me All Sections **ThoughtWorks**  

## SacrificialArchitecture

 **Martin Fowler**  
20 October 2014

You're sitting in a meeting, contemplating the code that your team has been working on for the last couple of years. You've come to the decision that the best thing you can do now is to throw away all that code, and rebuild on a totally new architecture. How does that make you feel about that doomed code, about the time you spent working on it, about the decisions you made all that time ago?

For many people throwing away a code base is a sign of failure, perhaps understandable given the inherent exploratory nature of software development, but still failure.

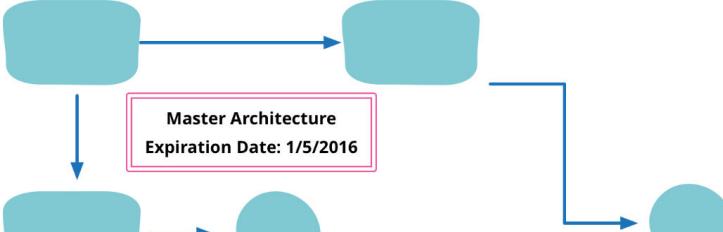
But often the best code you can write now is code you'll discard in a couple of years time.

**Find similar articles at these tags**

**process theory**

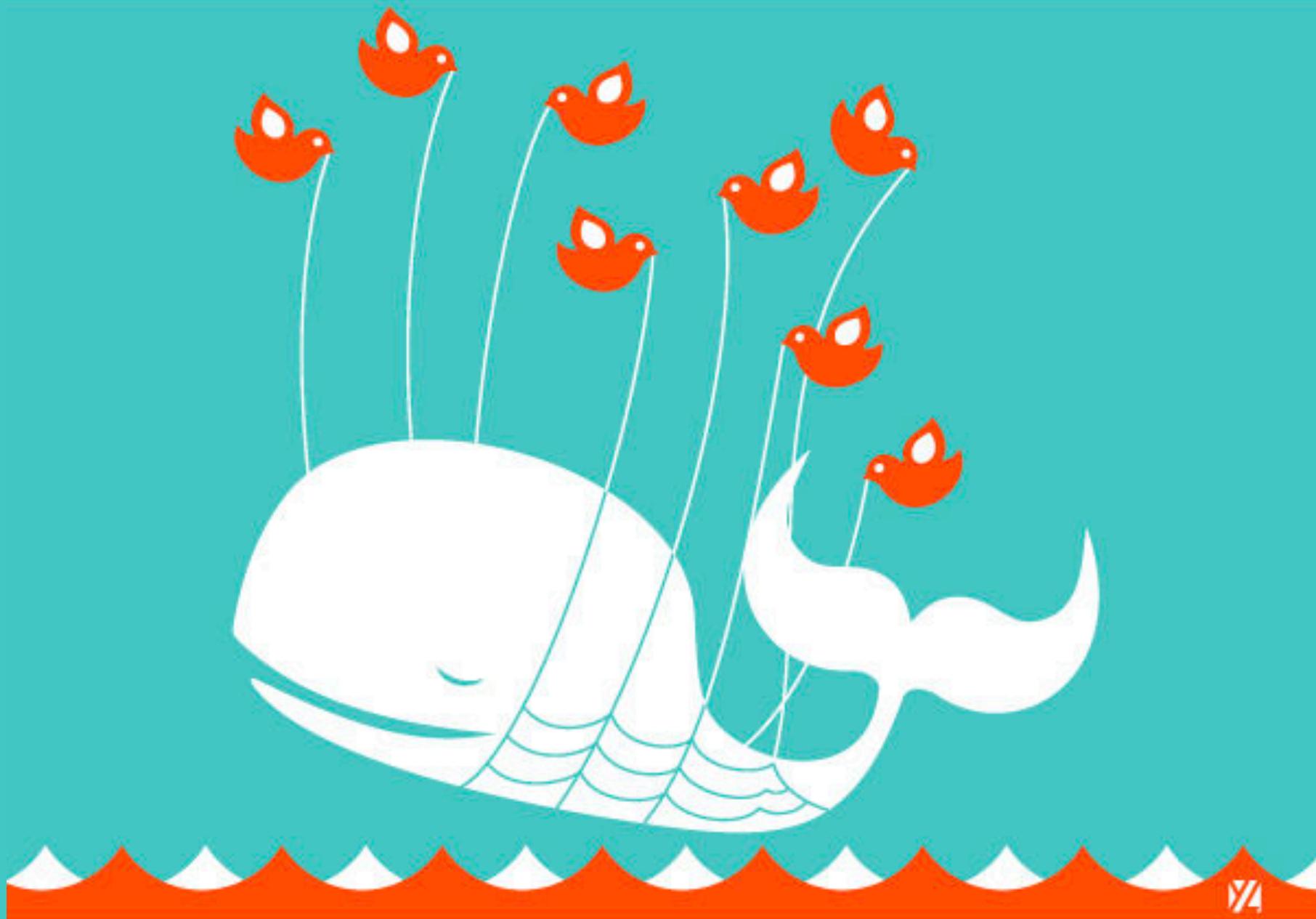
**evolutionary design**

**application architecture**



```
graph LR; MA[Master Architecture  
Expiration Date: 1/5/2016] --> N1(( )); N1 --> N2(( )); N1 --> N3(( )); N2 --> N4(( )); N3 --> N5(( )); N4 --> N6(( ));
```

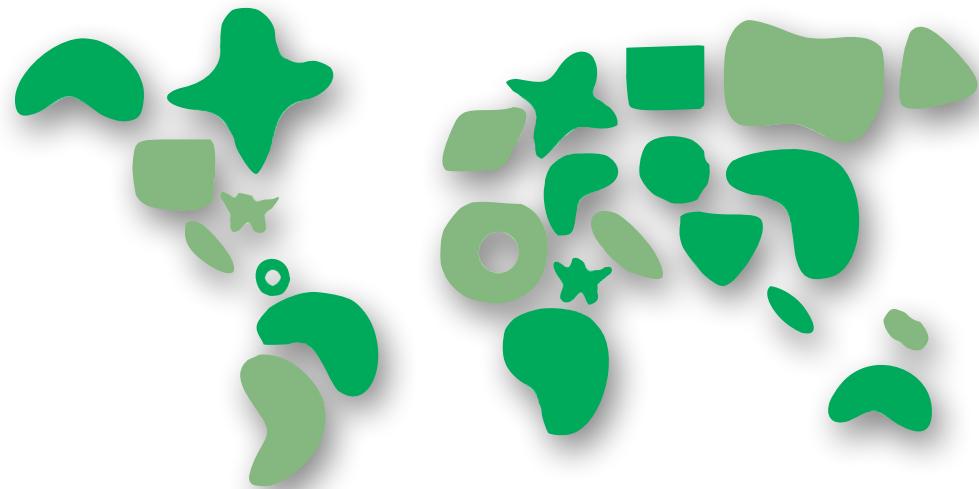
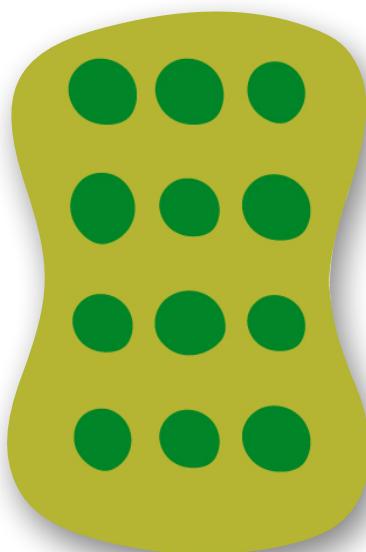
<https://martinfowler.com/bliki/SacrificialArchitecture.html>





**Build Sacrificial  
Architectures**

# The Business Case



# Hypothesis Driven UX

The screenshot shows a Medium article page. At the top, there's a navigation bar with icons for back, forward, search, and user profile. The title 'Hypothesis Driven UX' is displayed prominently. Below the title, a bio for 'Maximilian Wambach' is shown, along with the date 'Jan 11, 2015' and a note indicating a '5 min read'. The main content features a large, close-up photograph of a bottle of wine lying on its side, surrounded by many corks. The wine bottle label is visible, showing the words 'HYPOTHESIS' and 'SCHLOSS SAUTERN'. The headline 'Hypothesis driven UX design' is overlaid on the image. A subtitle below it reads: 'The value of design changes when you enable whole teams to learn instead of just looking at pretty mockups'. The text of the article begins with a paragraph about a project at mobile.de where a product manager had an idea to redesign the iPhone app iteratively to address usability issues.

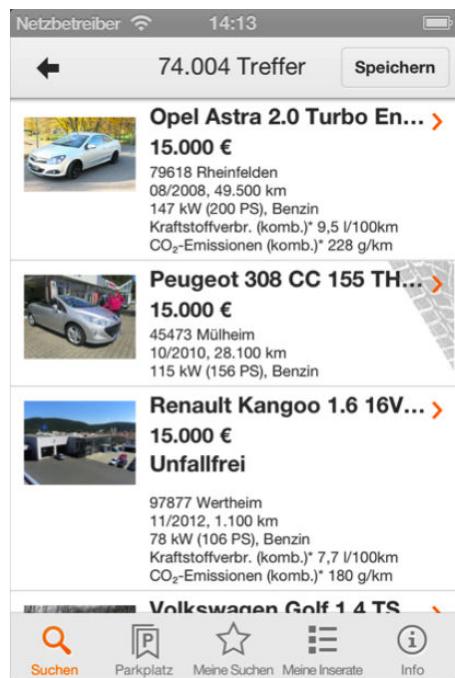
Shortly before I joined mobile.de (an ebay owned automotive platform in Germany with millions of visits per month) a smart product manager had an idea: The iPhone app should be redesigned not only to match current visual design guidelines and trends but also to get rid of usability issues that were caused by the current design approach.

Shortly after my start at the company I was handed the project. We tried to tackle the project with an iterative approach. Knowing that people are mostly reluctant to change we changed only parts of the app, released and when everything went well we took on the next elements of the app. Everything went pretty well: numbers stayed stable, no "redesign drop" and feedback from users was overall positive.

**Curious case of a result view**

<https://medium.com/@mwambach1/hypotheses-driven-ux-design-c75fbf3ce7cc#.gk3dpip81>

# Hypothesis Driven UX



# Three Hypotheses

## **More Listings**

If we provide more listings on the screen then we can provide better comparability and offer more diversity on our platform because users like to compare a lot of listings on the result page

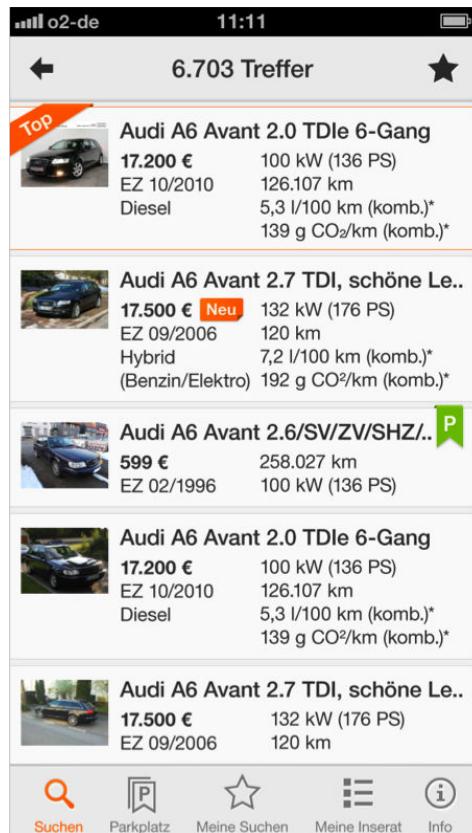
## **Better Structure**

If we provide more structure to our listings then we achieve a better scanability because the user is able to scan the relevant information quicker

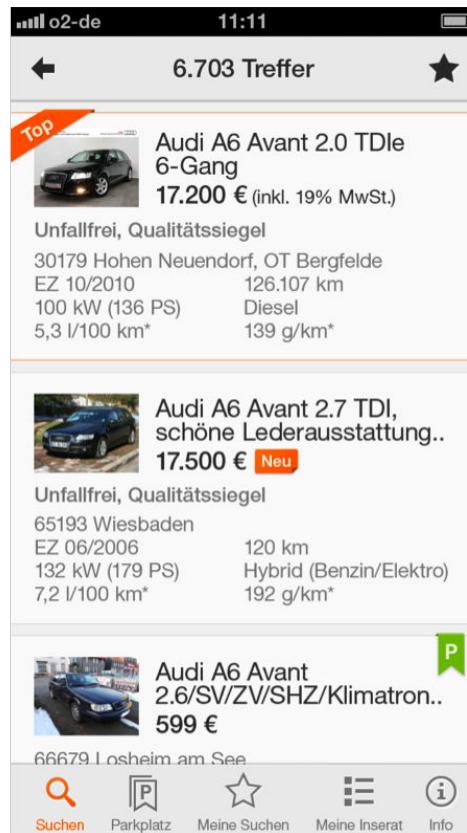
## **Better Prioritization**

If we prioritize information according to user needs then we achieve better guidance because the user can see all relevant information at a glance

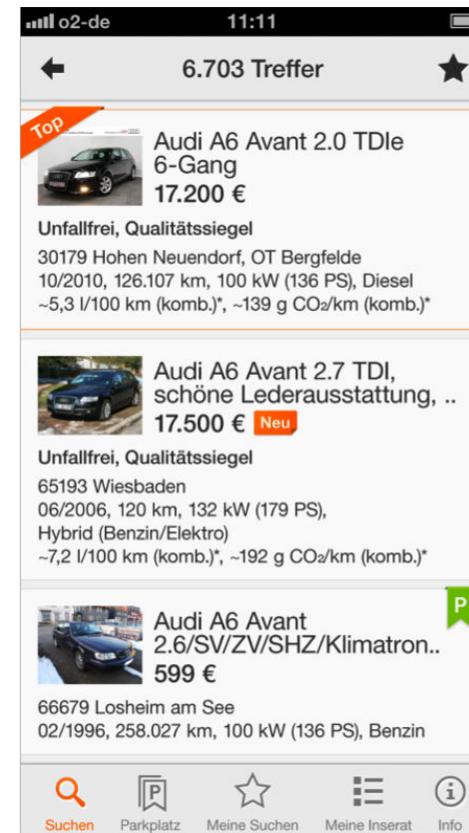
# Experiments to Perform



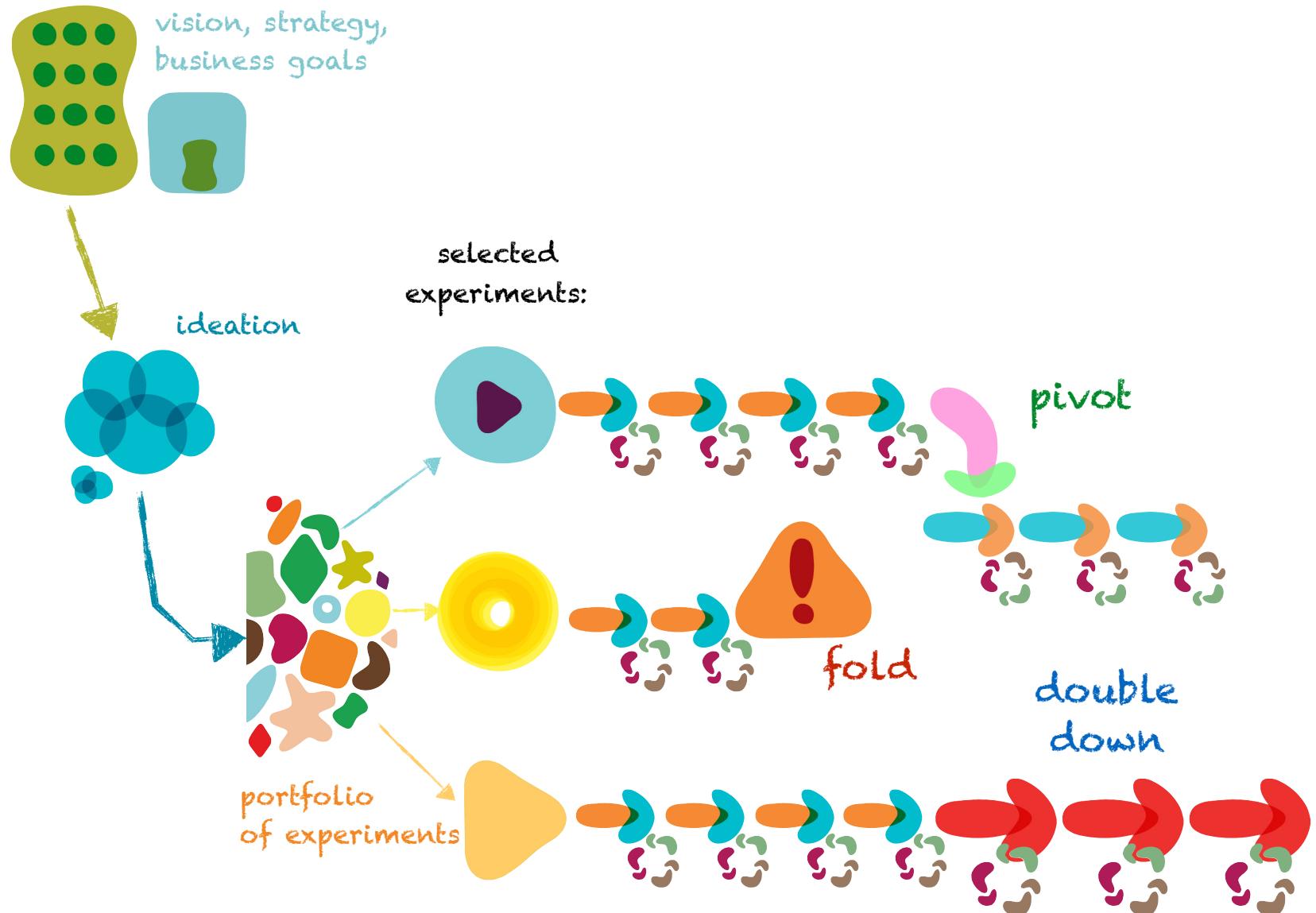
More Listings

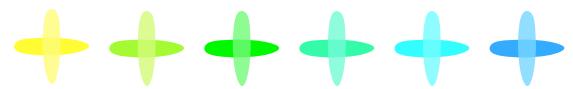


Better Structure

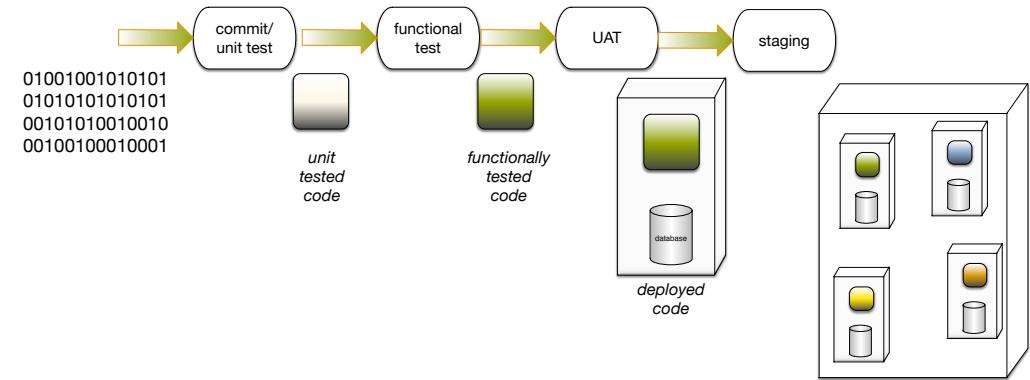
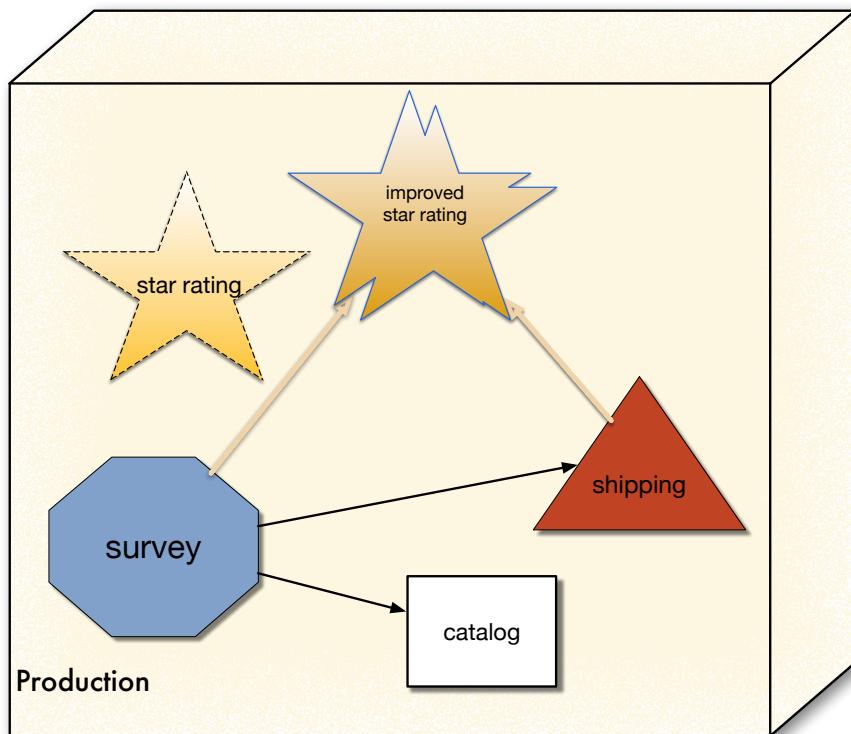


Better Prioritization

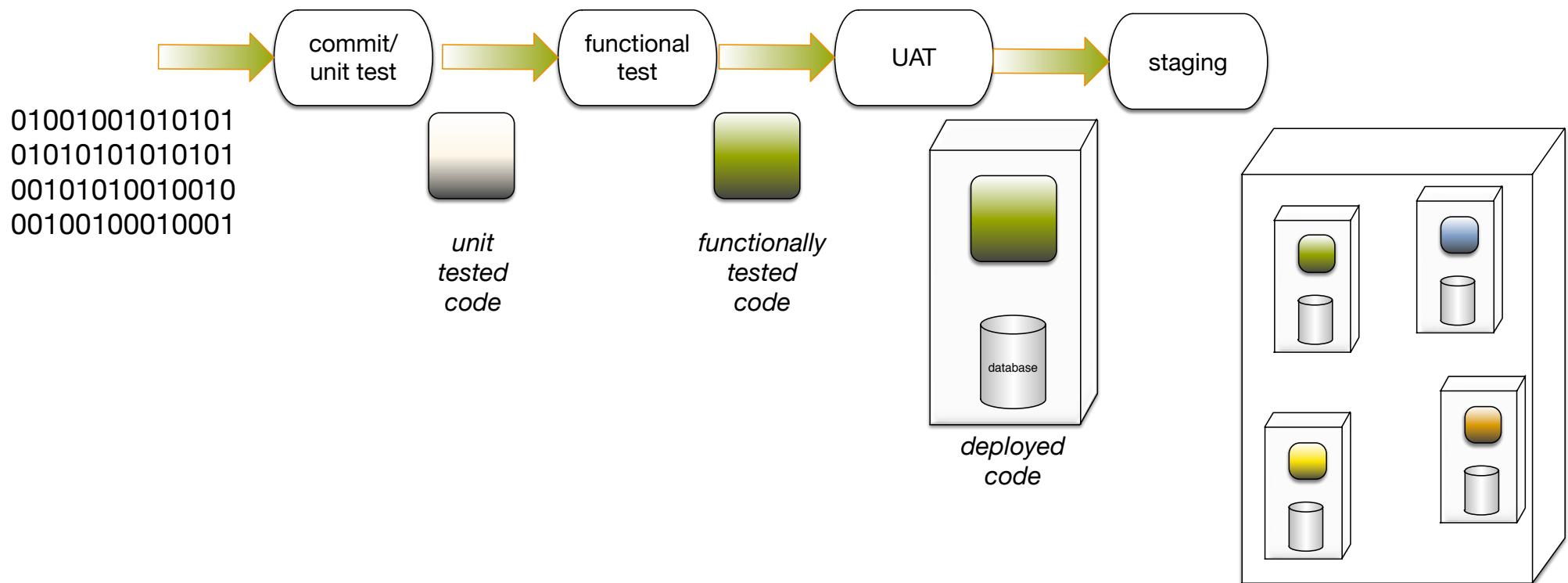




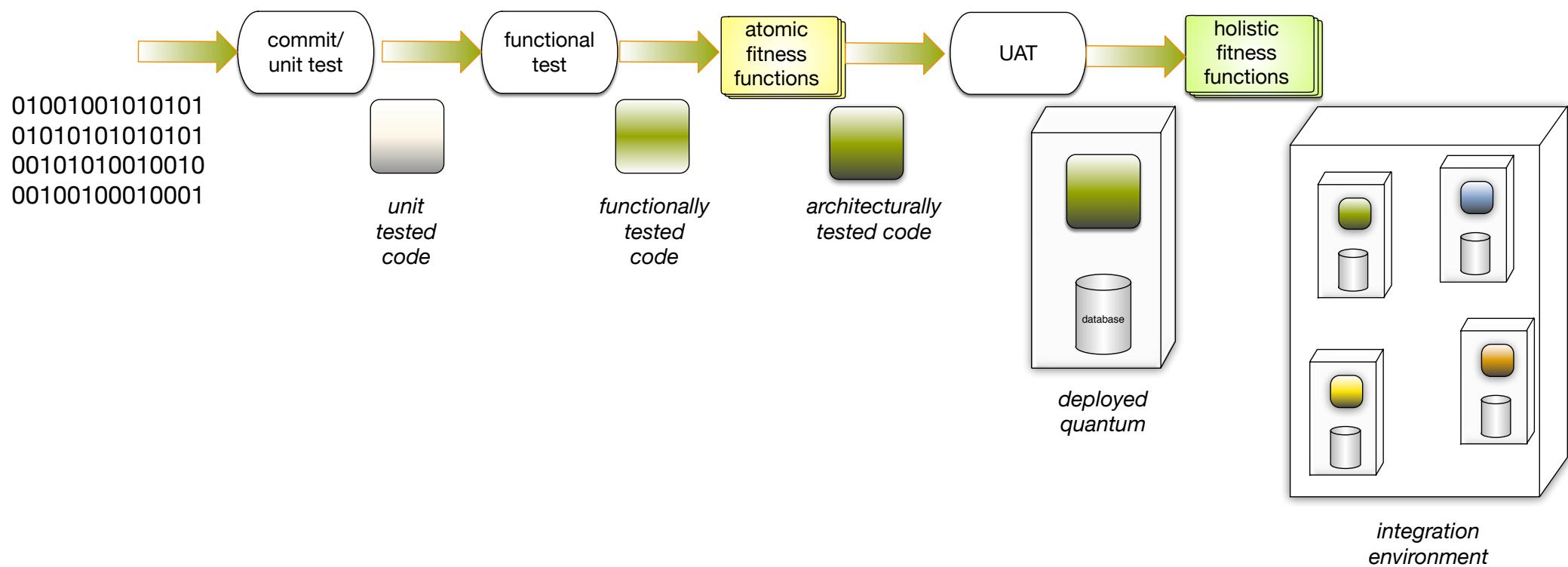
# incremental



# Deployment Pipeline

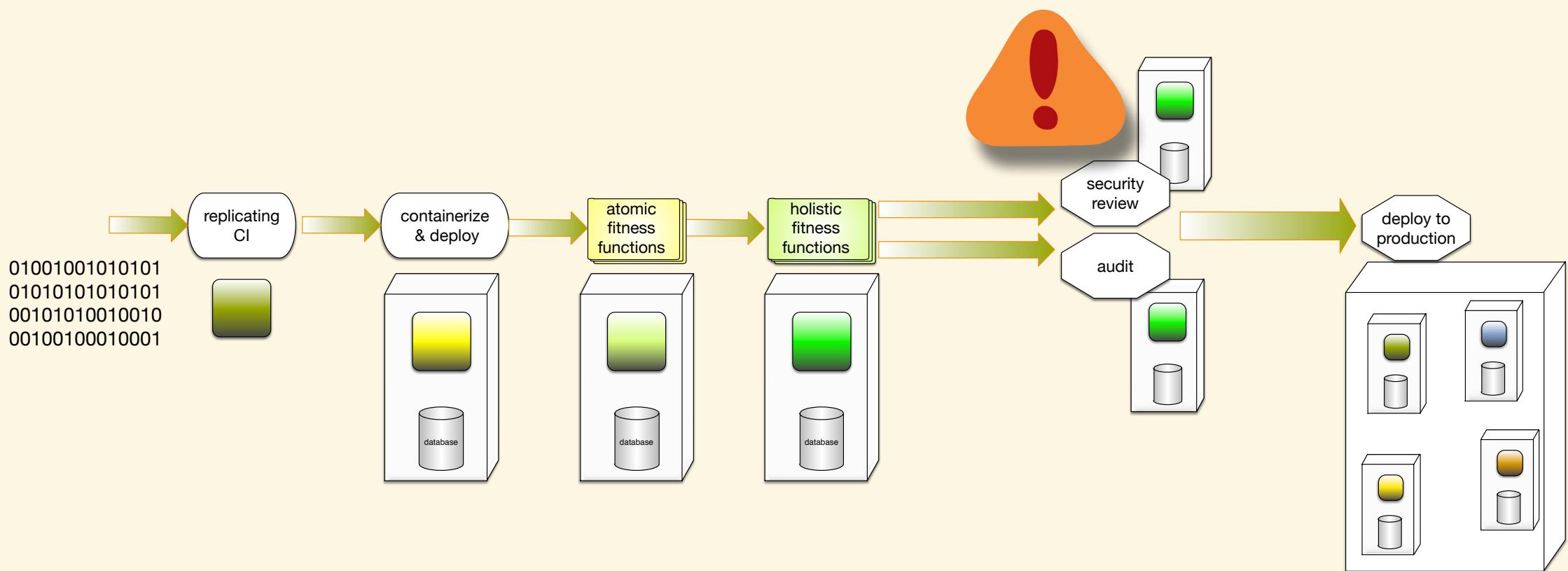


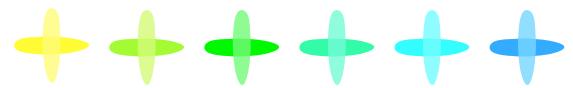
# Deployment Pipeline + Fitness Functions



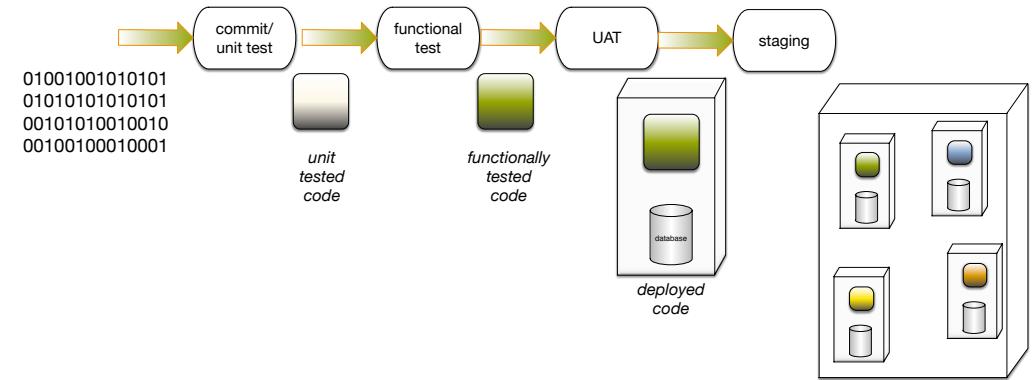
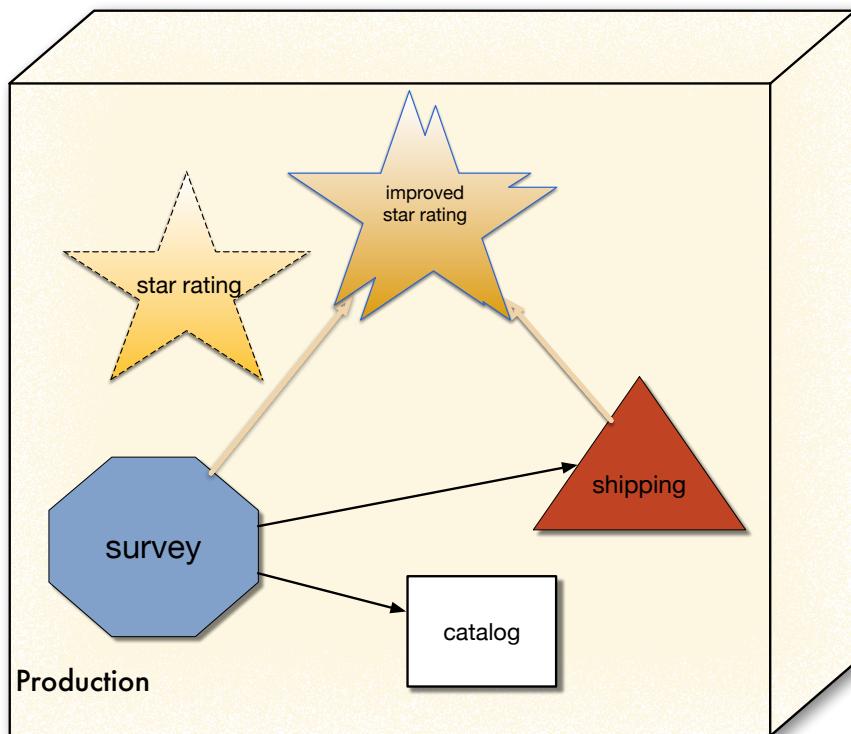
Penultimate  
↑  
3

# Deployment Pipeline

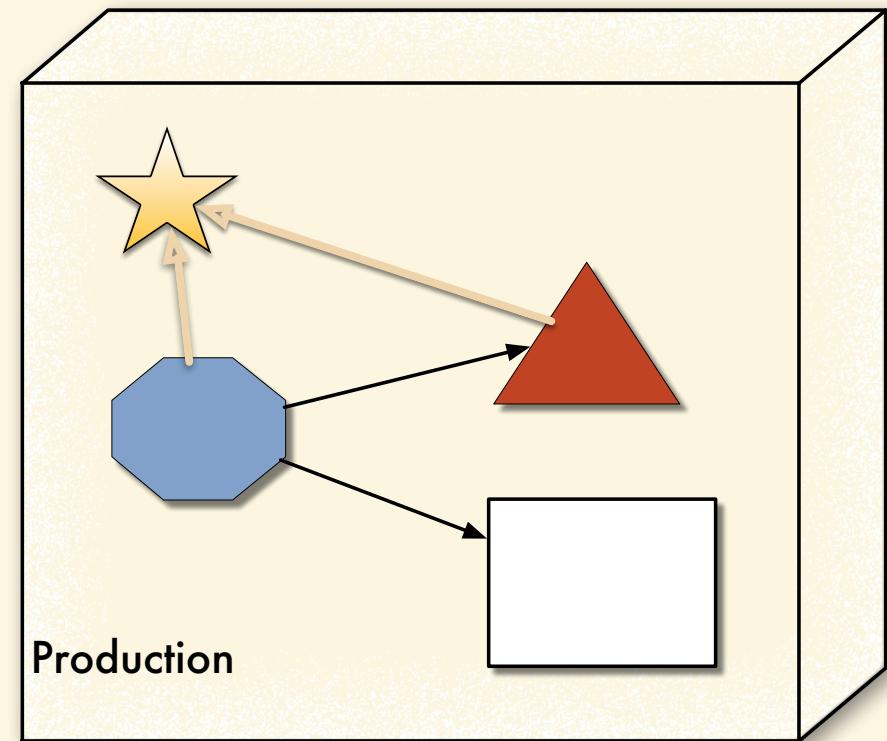
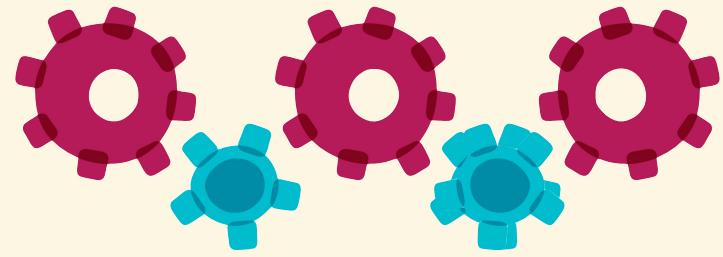




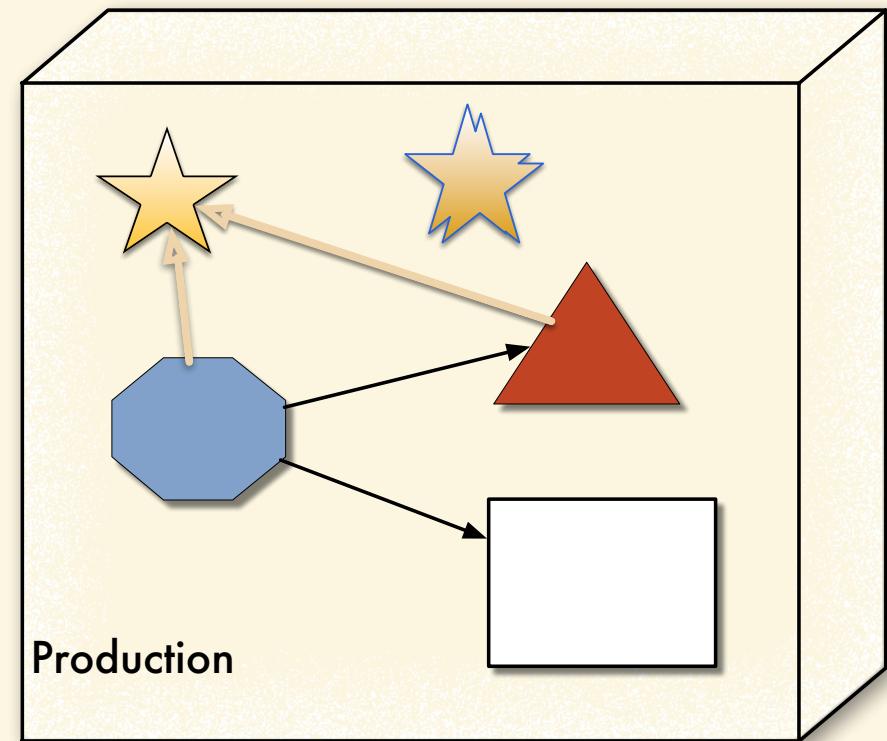
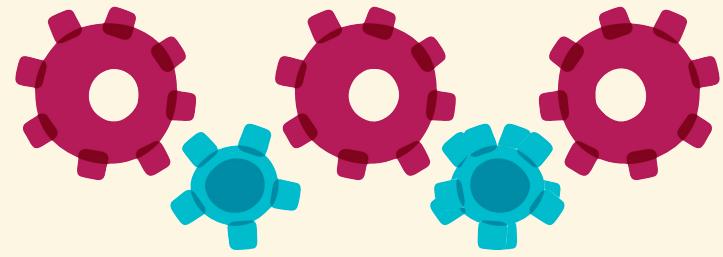
# incremental



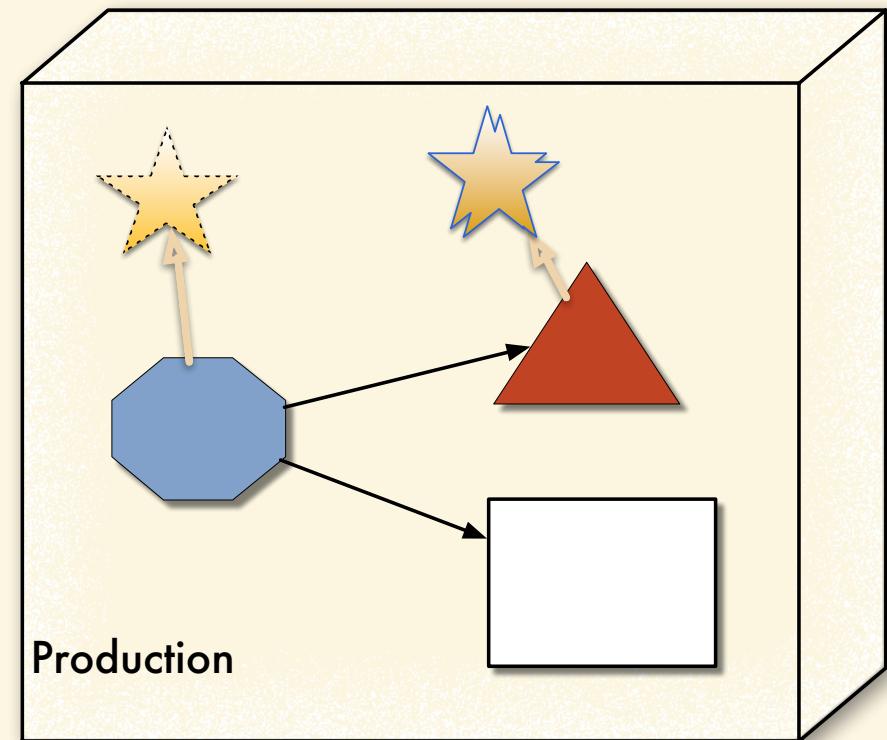
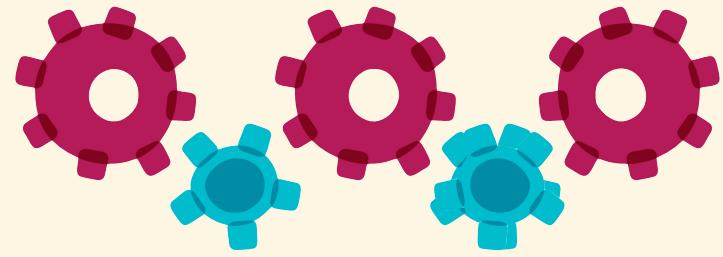
# Penultima↑e



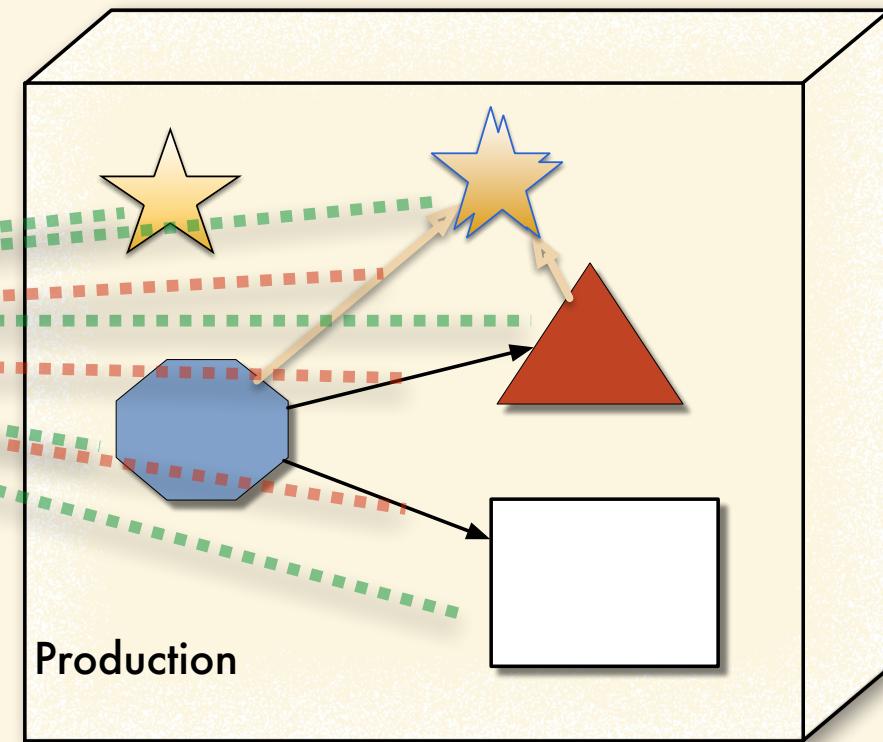
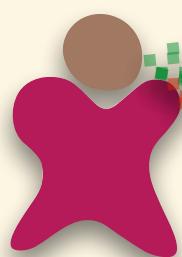
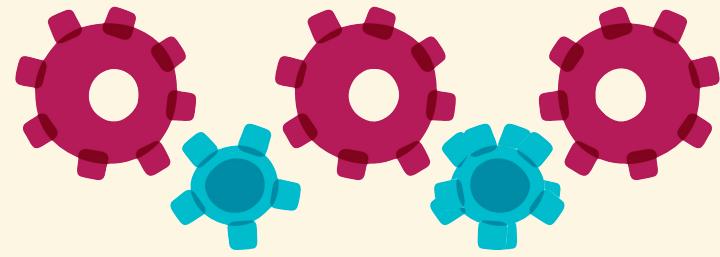
# Penultima↑e



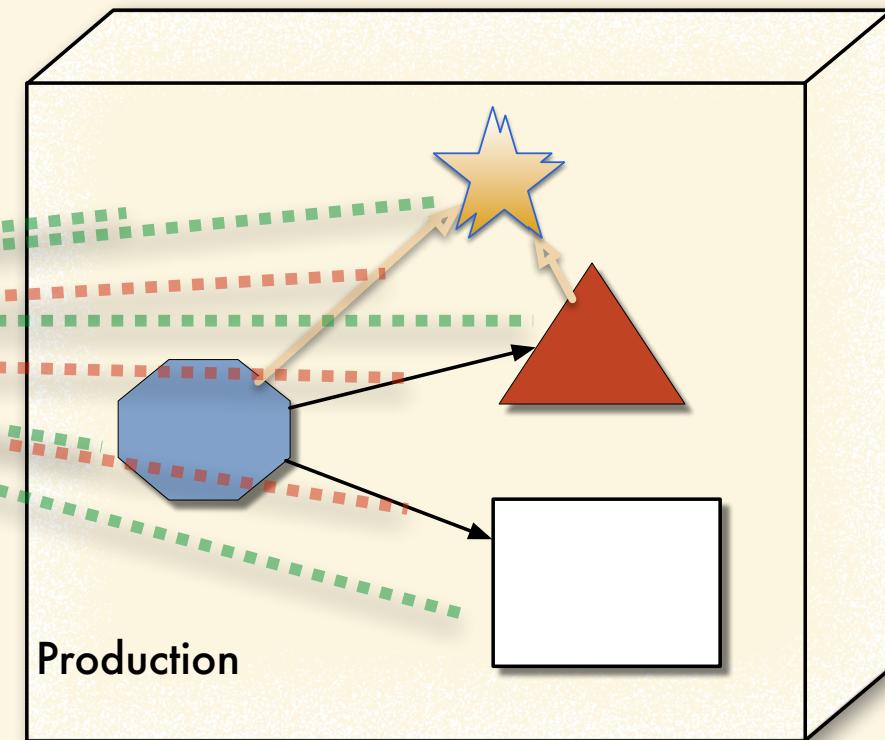
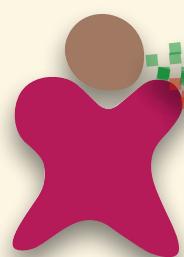
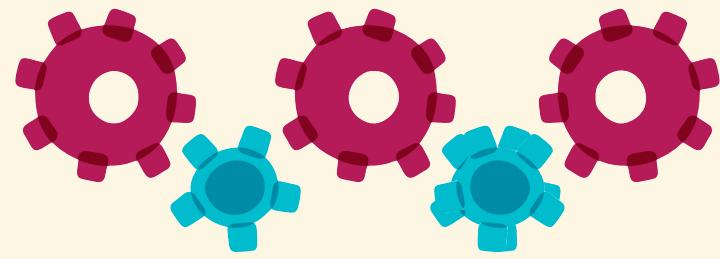
Penultima ↑ e



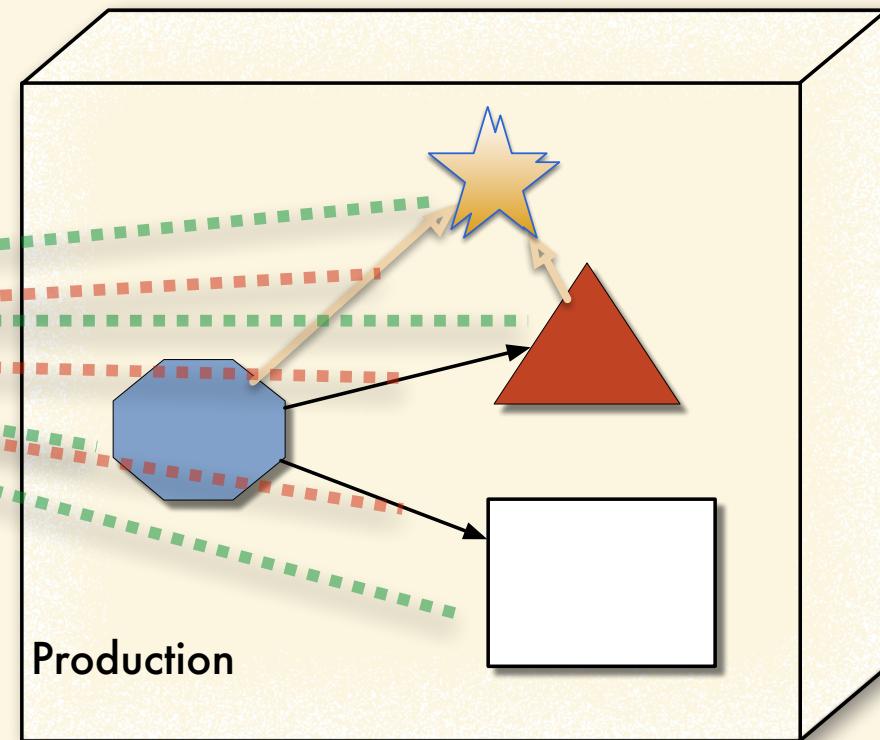
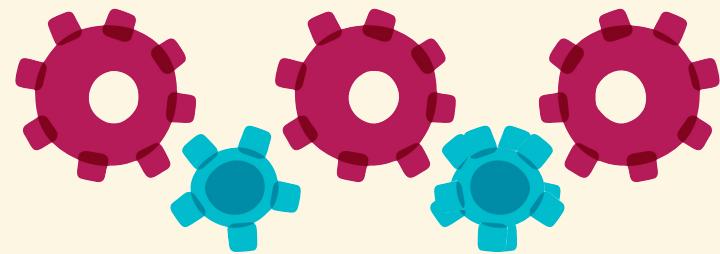
# Penultima↑e



# Penultima↑e



# Penultima↑e



The screenshot shows a web browser displaying a blog post from GitHub Engineering. The title of the post is "Move Fast and Fix Things". The author is listed as "vmg" and the date is "December 15, 2015". The post discusses technical debt and how GitHub handles it. It includes a section titled "Merges in Git" which explains GitHub's storage model and merge strategy. The post concludes with a note about a shell script used for temporary working directories.

Anyone who has worked on a large enough codebase knows that technical debt is an inescapable reality: The more rapidly an application grows in size and complexity, the more technical debt is accrued. With GitHub's growth over the last 7 years, we have found plenty of nooks and crannies in our codebase that are inevitably below our very best engineering standards. But we've also found effective and efficient ways of paying down that technical debt, even in the most active parts of our systems.

At GitHub we try not to brag about the "shortcuts" we've taken over the years to scale our web application to more than 12 million users. In fact, we do quite the opposite: we make a conscious effort to study our codebase looking for systems that can be rewritten to be cleaner, simpler and more efficient, and we develop tools and workflows that allow us to perform these rewrites efficiently and reliably.

As an example, two weeks ago we replaced one of the most critical code paths in our infrastructure: the code that performs merges when you press the Merge Button in a Pull Request. Although we routinely perform these kind of refactorings throughout our web app, the importance of the merge code makes it an interesting story to demonstrate our workflow.

## Merges in Git

We've [talked at length in the past](#) about the storage model that GitHub uses for repositories in our platform and our Enterprise offerings. There are many implementation details that make this model efficient in both performance and disk usage, but the most relevant one here is the fact that repositories are always stored "bare".

This means that the actual files in the repository (the ones that you would see on your working directory when you clone the repository) are not actually available on disk in our infrastructure: they are compressed and delta'ed inside [packfiles](#).

Because of this, performing a merge in a production environment is a nontrivial endeavour. Git knows several [merge strategies](#), but the recursive merge strategy that you'd get by default when using `git merge` to merge two branches in a local repository assumes the existence of a working tree for the repository, with all the files checked out on it.

The workaround we developed in the early days of GitHub for this limitation is effective, but not particularly elegant: instead of using the default [`git-merge--recursive`](#) strategy, we wrote our own merge strategy based on the original one that Git used back in the day: [`git-merge--resolve`](#). With some tweaking, the old strategy can be adapted to not require an actual checkout of the files on disk.

To accomplish this, we wrote a shell script that sets up a [temporary working directory](#), in

Move  
Fast  
&  
Fix  
Things



```
def create_merge_commit(base, head, author, commit_message)
  base = resolve_commit(base)
  head = resolve_commit(head)
  commit_message = Rugged.prettyify_message(commit_message)

  merge_base = rugged.merge_base(base, head)
  return [nil, "already_merged"] if merge_base == head.oid

  ancestor_tree = merge_base && Rugged::Commit.lookup(rugged, merge_base).tree
  merge_options = {
    :fail_on_conflict => true,
    :skip_recur => true,
    :no_recursive => true,
  }
  index = base.tree.merge(head.tree, ancestor_tree, merge_options)
  return [nil, "merge_conflict"] if (index.nil? || index.conflicts?)

  options = {
    :message => commit_message,
    :committer => author,
    :author => author,
    :parents => [base, head],
    :tree => index.write_tree(rugged)
  }

  [Rugged::Commit.create(rugged, options), nil]
end
```

```
def create_merge_commit(author, base, head, options = {})
  commit_message = options[:commit_message] || "Merge #{head} into #{base}"
  now = Time.current

  science "create_merge_commit" do |e|
    e.context :base => base.to_s, :head => head.to_s, :repo => repository.repo
    e.use { create_merge_commit_git(author, now, base, head, commit_message) }
    e.try { create_merge_commit_rugged(author, now, base, head, commit_message) }
  end
end
```

A screenshot of a GitHub repository page for the 'scientist' library. The repository has 71 commits, 1 branch, 6 releases, and 16 contributors. The README.md file contains a section titled "Scientist!" which describes the library as "A Ruby library for carefully refactoring critical paths." It includes a "How do I science?" section with code examples for changing permission handling in a web application. A note at the bottom of the README explains how to wrap existing behavior with a 'use' block and new behavior with a 'try' block.

<https://github.com/github/scientist>



```
require "scientist"

class MyWidget
  def allows?(user)
    experiment = Scientist::Default.new "widget-permissions"
    experiment.use { model.check_user?(user).valid? } # old way
    experiment.try { user.can?(:read, model) } # new way

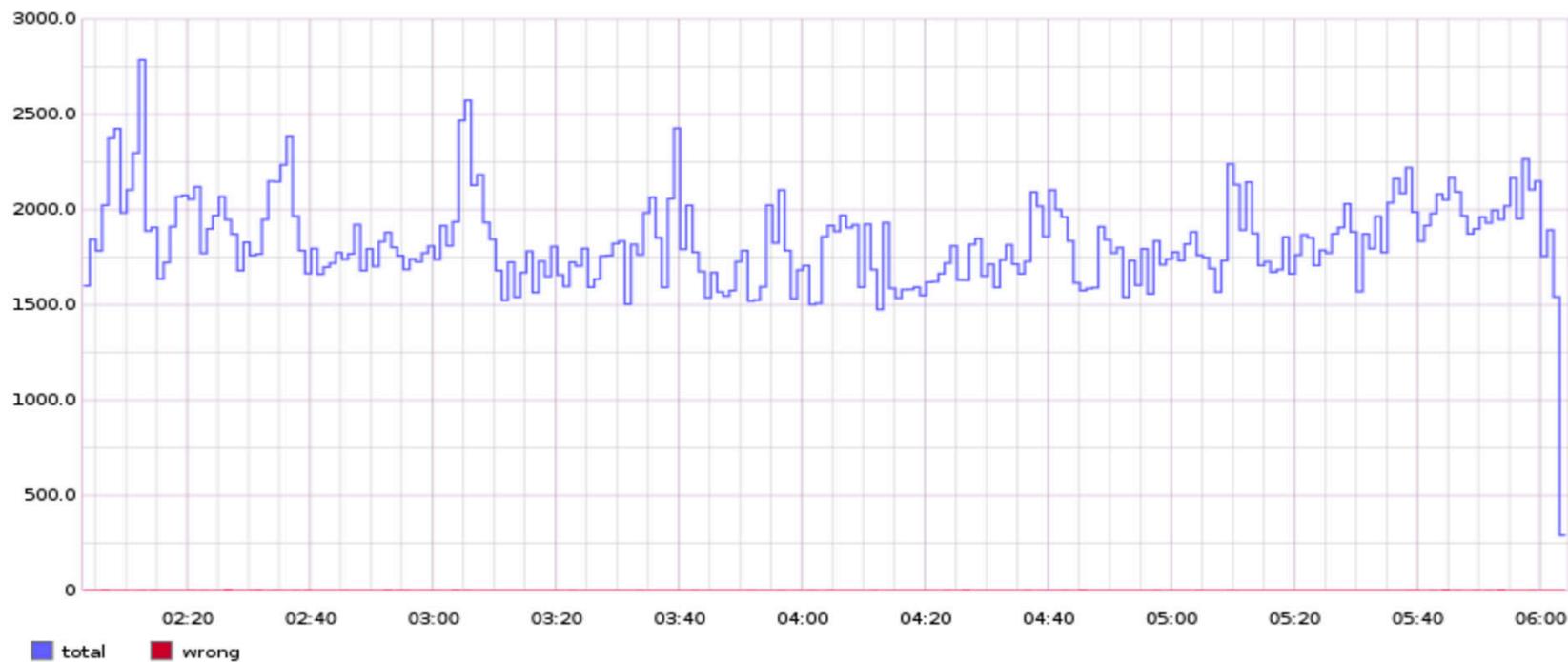
    experiment.run
  end
end
```

- It decides whether or not to run the try block,
- Randomizes the order in which use and try blocks are run,
- Measures the durations of all behaviors,
- Compares the result of try to the result of use,
- Swallows (but records) any exceptions raised in the try block
- Publishes all this information.



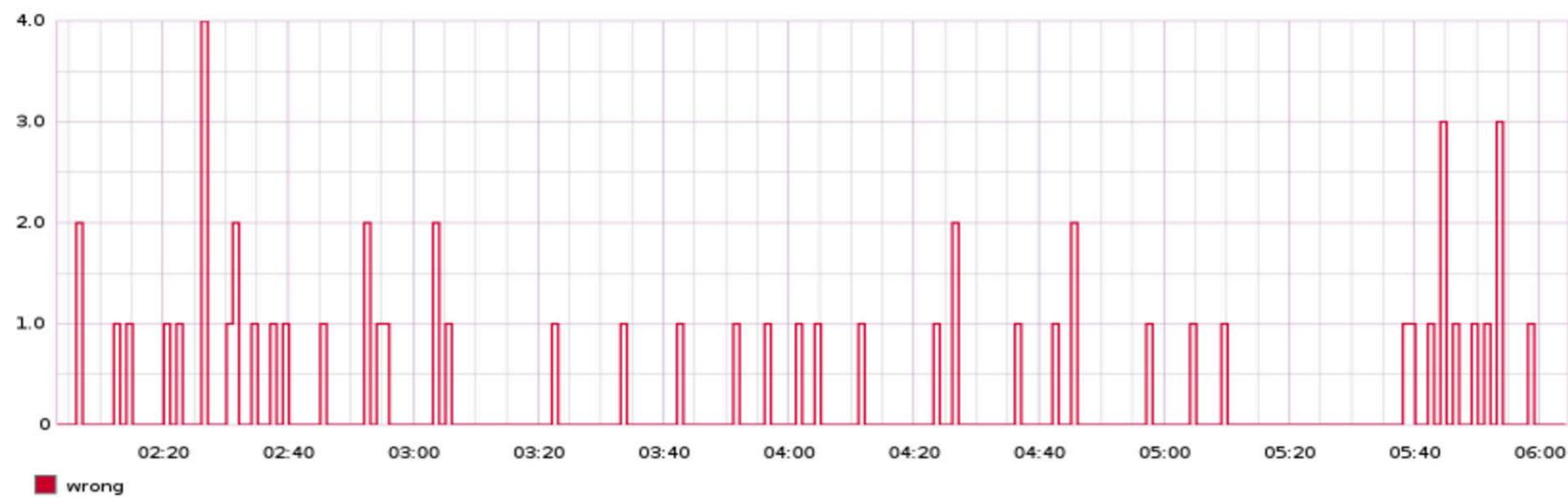
## Accuracy

The number of times that the candidate and the control agree or disagree. [View mismatches](#)



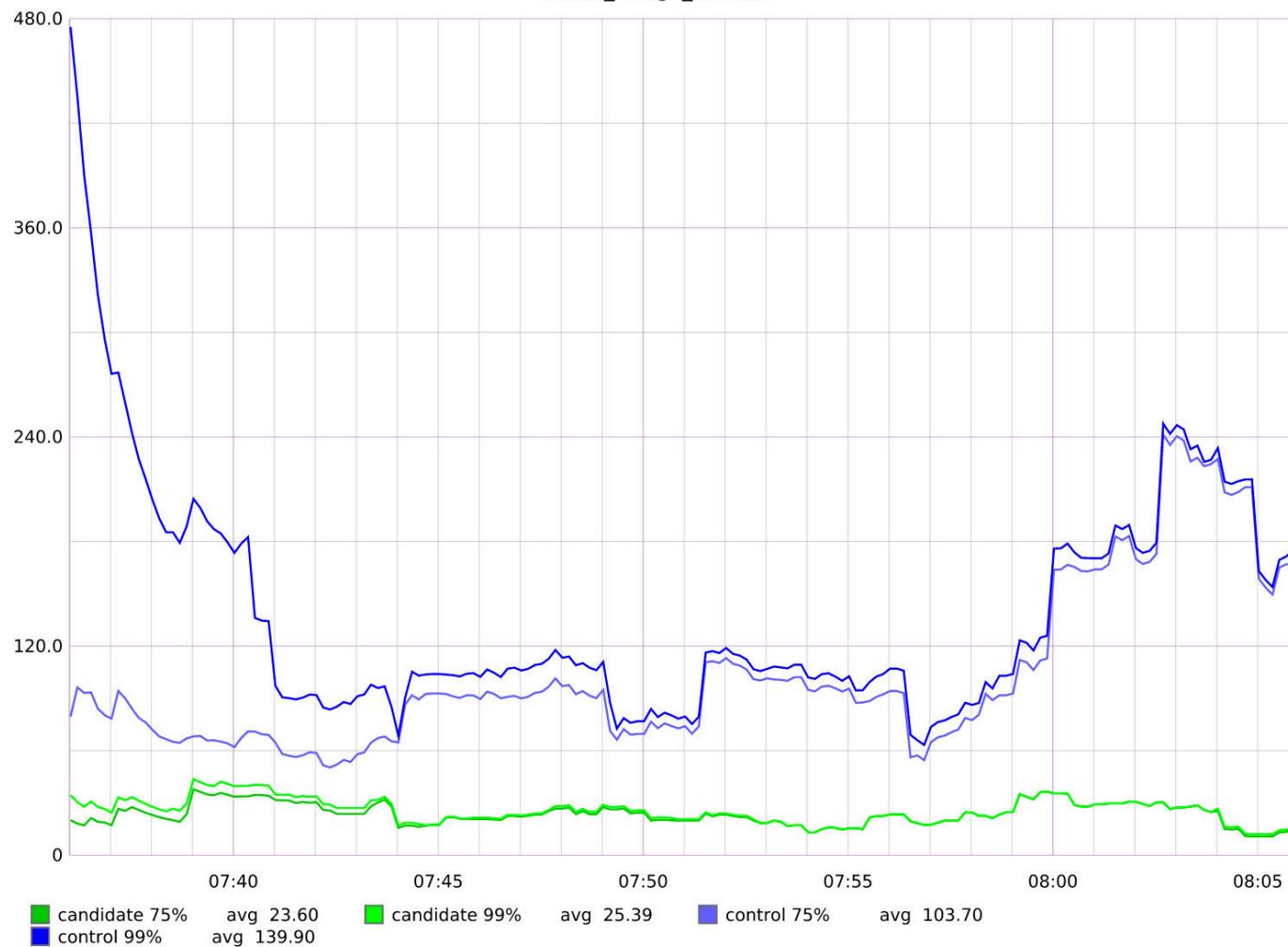


The number of incorrect/ignored only.





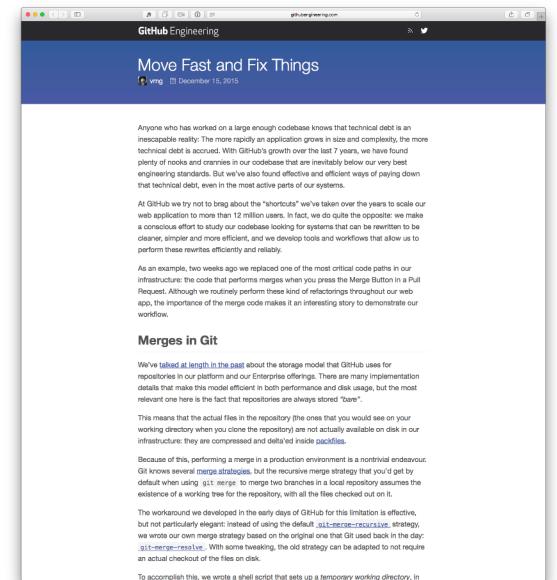
create\_merge\_commit



4 days

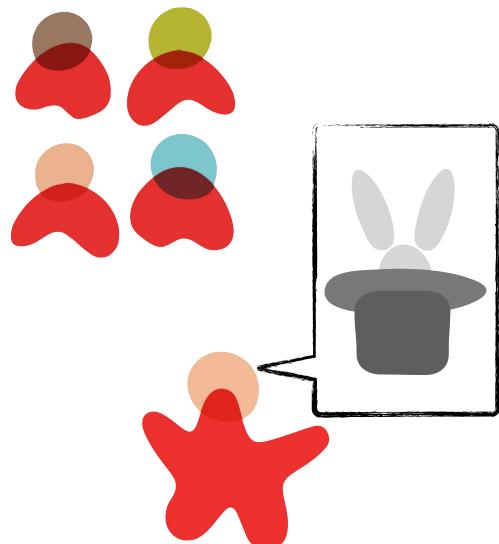
24 hours/  
no mismatches or slow cases

> 10,000,000  
comparisons

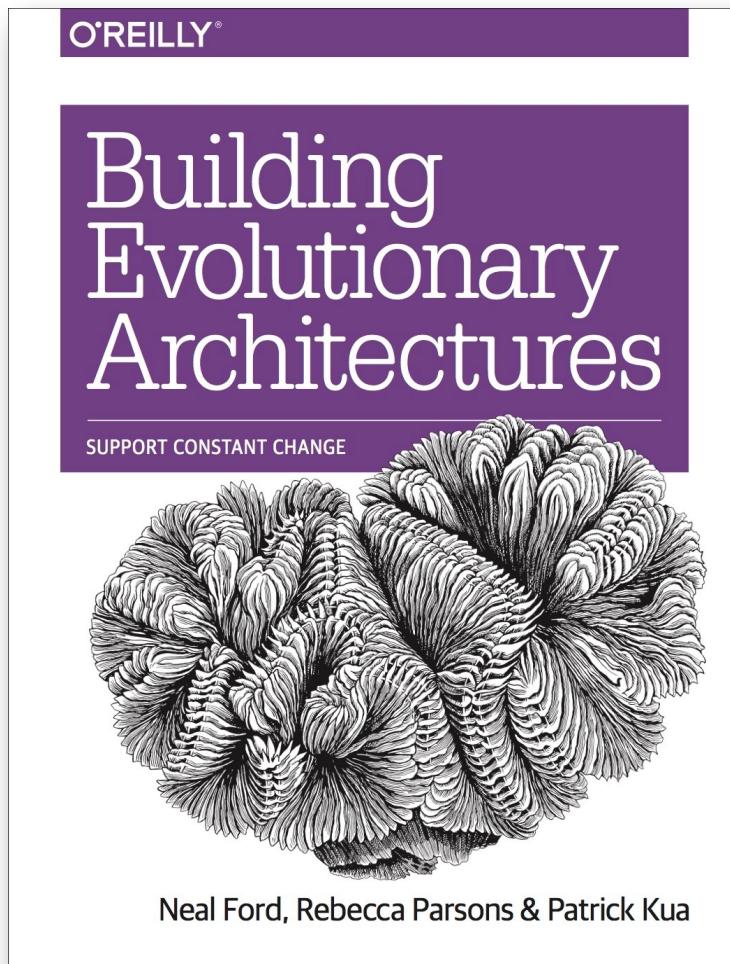


# Case Study: Consulting Judo

Demonstration defeats discussion!



# Building Evolutionary Architectures



For more information:

A screenshot of a website page for 'Building Evolutionary Architectures'. The top navigation bar includes links for Home, Blog, Talks, and The Book. The main content area features a large image of green grass. Below the image is a button labeled 'Learn More'. At the bottom of the page, there are three circular icons with accompanying text: 'Incremental' (describing building parts at different points), 'Fitness Functions' (describing optimization for the environment), and 'Multiple Dimensions' (describing both technical and domain changes).

<http://evolutionaryarchitecture.com>