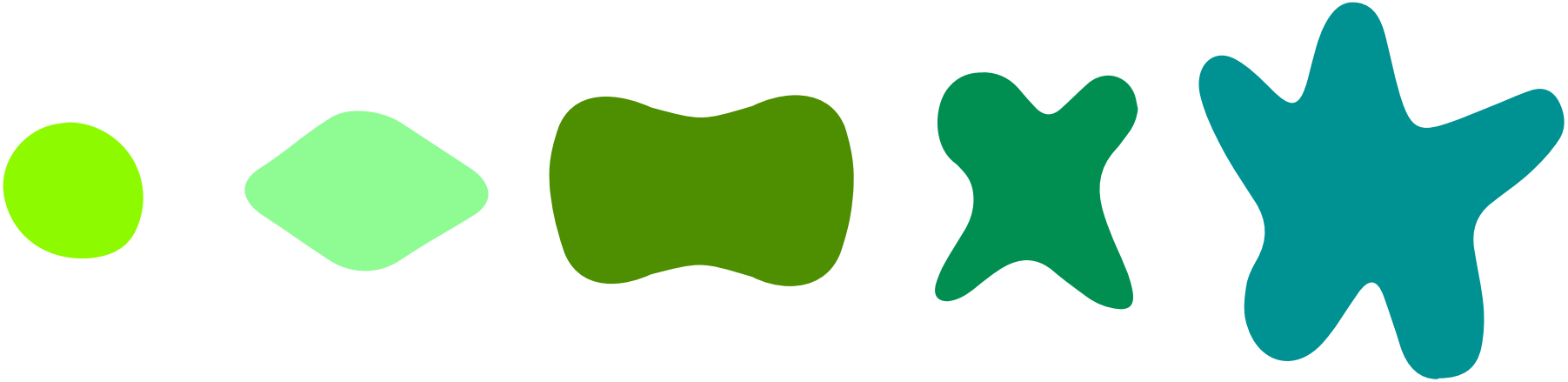


Evolutionary Architecture

ThoughtWorks®

NEAL FORD

Director / Software Architect / Meme Wrangler

@neal4d

nealford.com

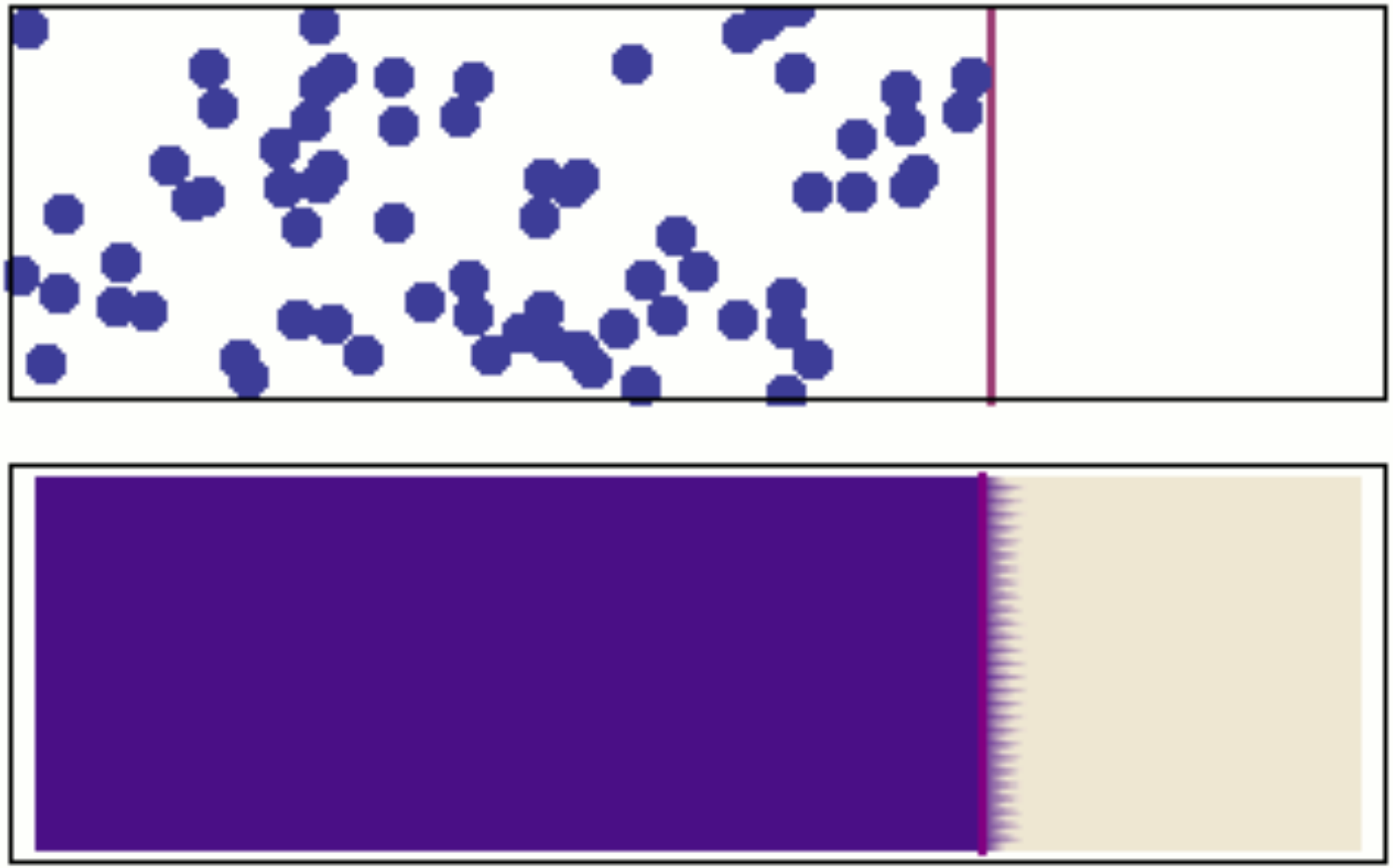
Rebecca Parsons

Pat Kua

Neal Ford

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

Dynamic Equilibrium



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

Dynamic Equilibrium

Docker

Definition :

An evolutionary architecture supports continual and incremental change as a first principle along multiple dimensions.

Types of Architecture:

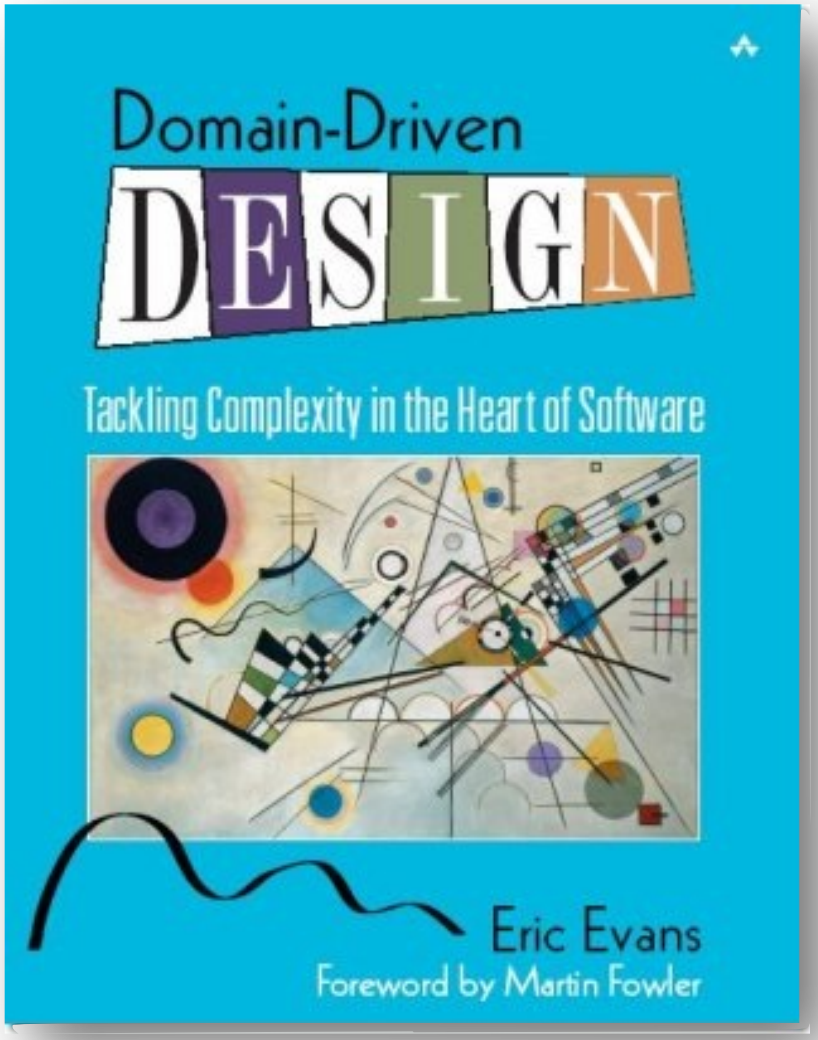
Technical: The implementation parts of the architecture

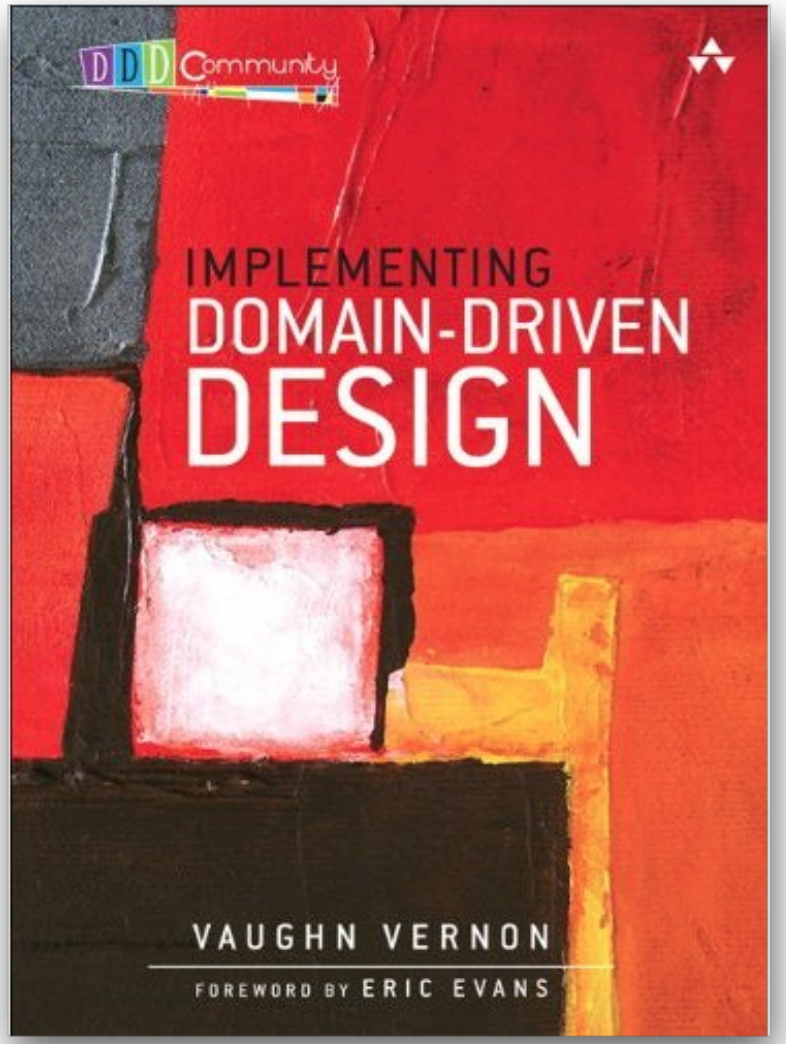
Data: Database schemas, table layouts, optimization planning, etc.

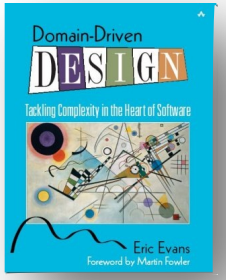
Security: Defines security policies, guidelines, and specifies tools to help uncover deficiencies.

Domain:

Domain Driven Design

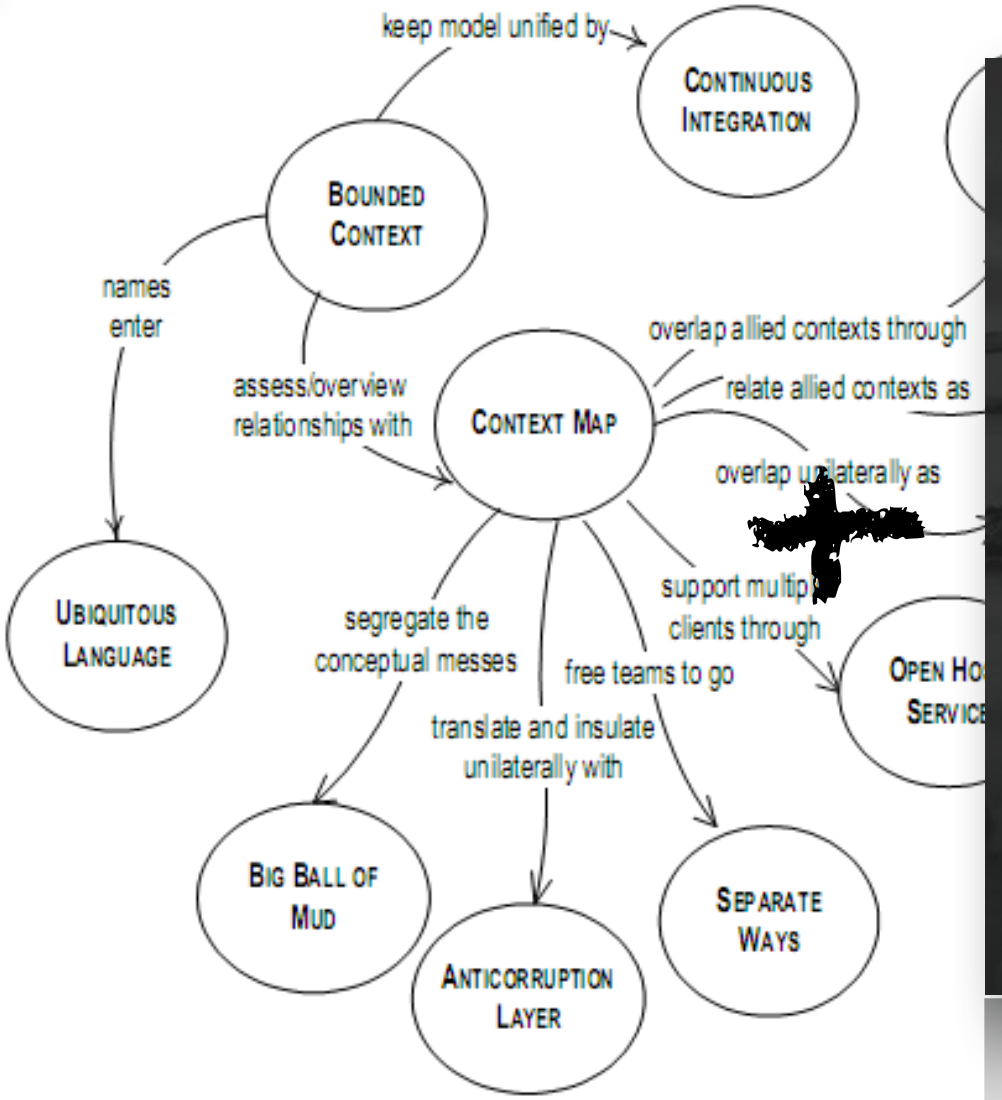


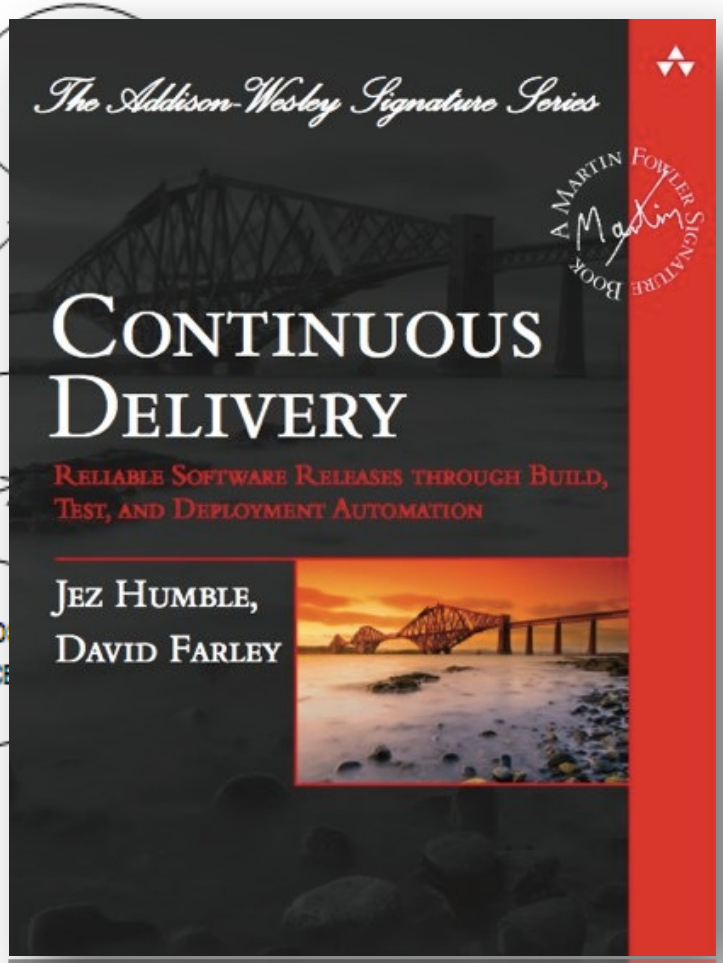




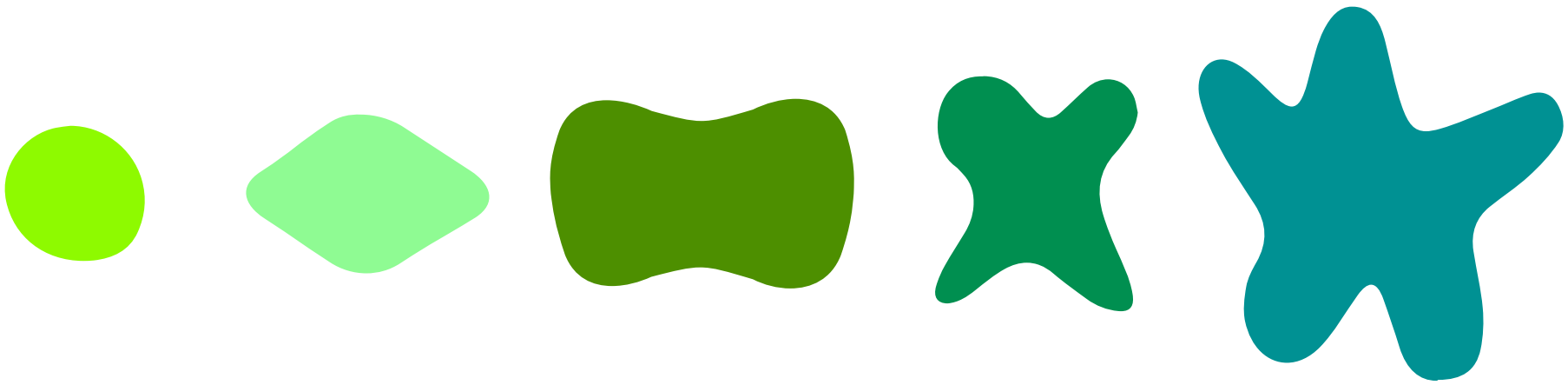
Bounded Context

Maintaining Model Integrity

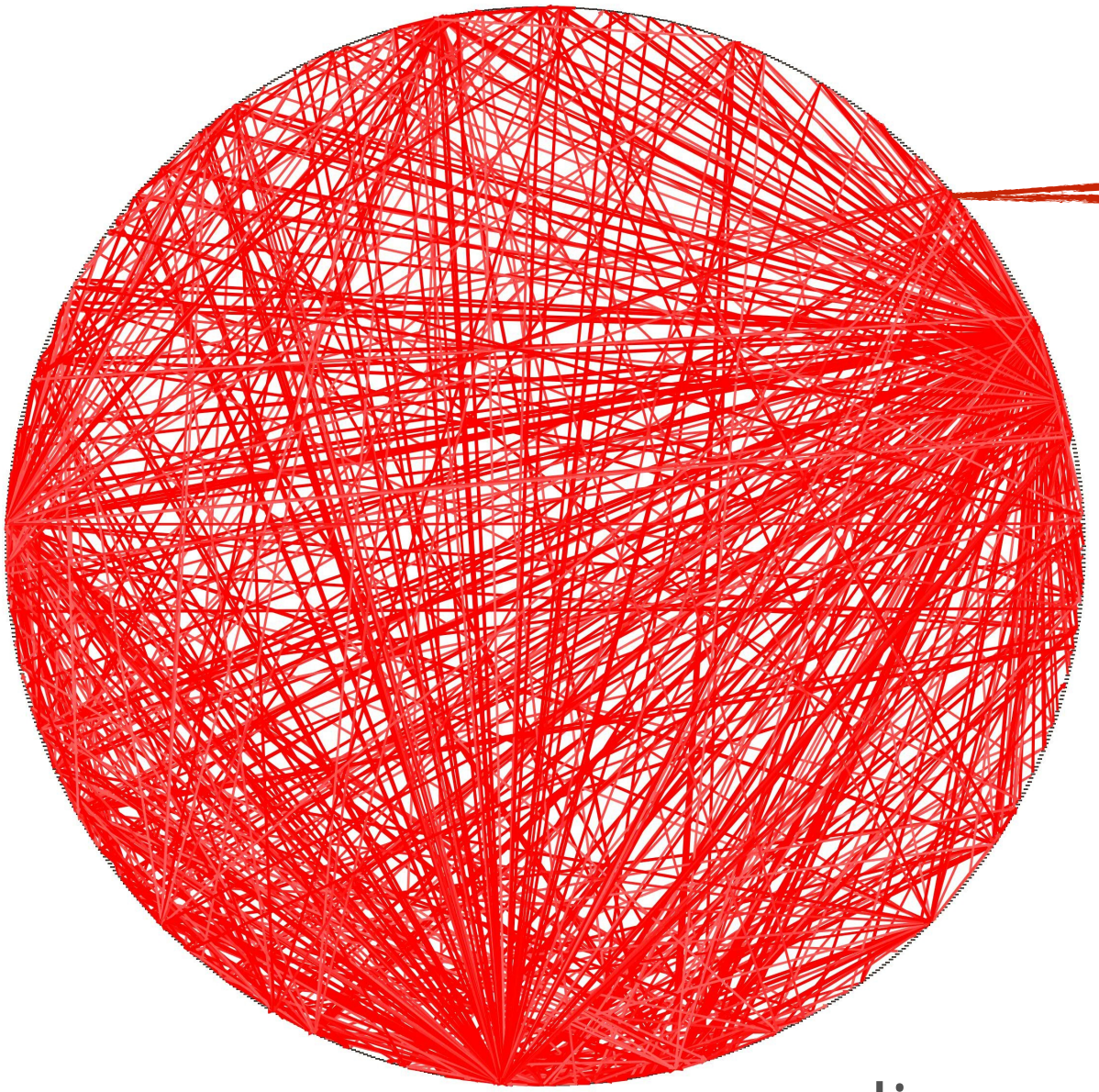




Evolvability of Architectures



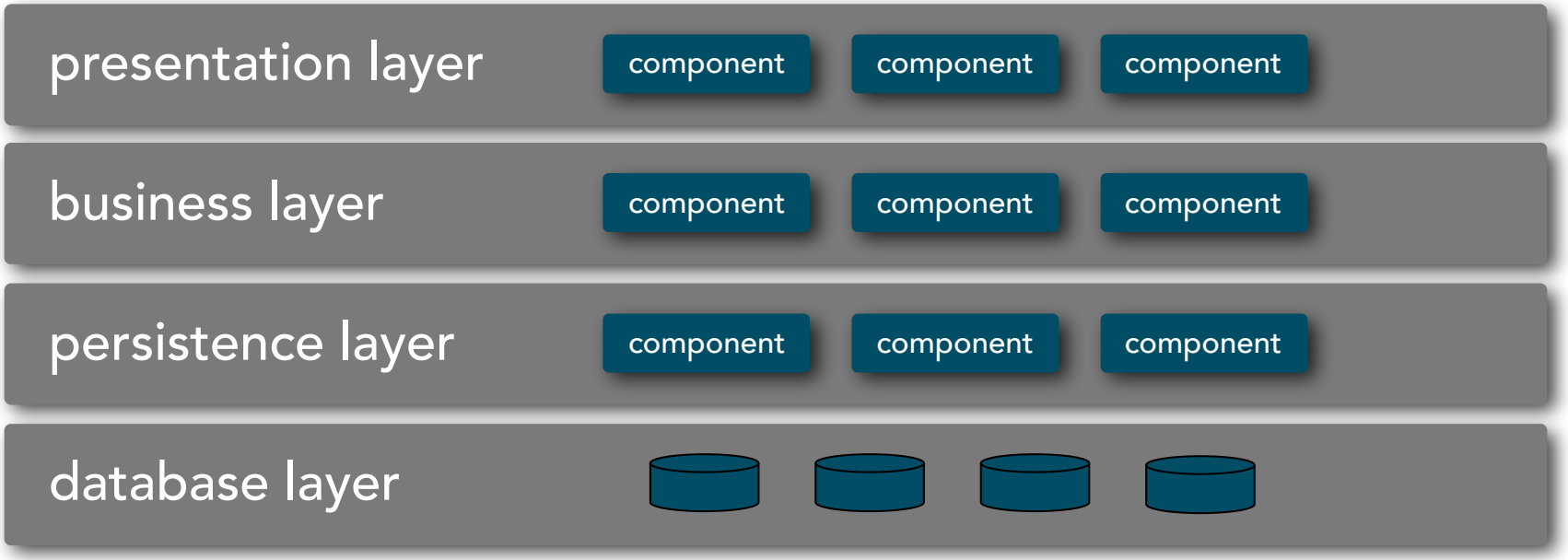
Big Ball of Mud



coupling connections

dimensions :

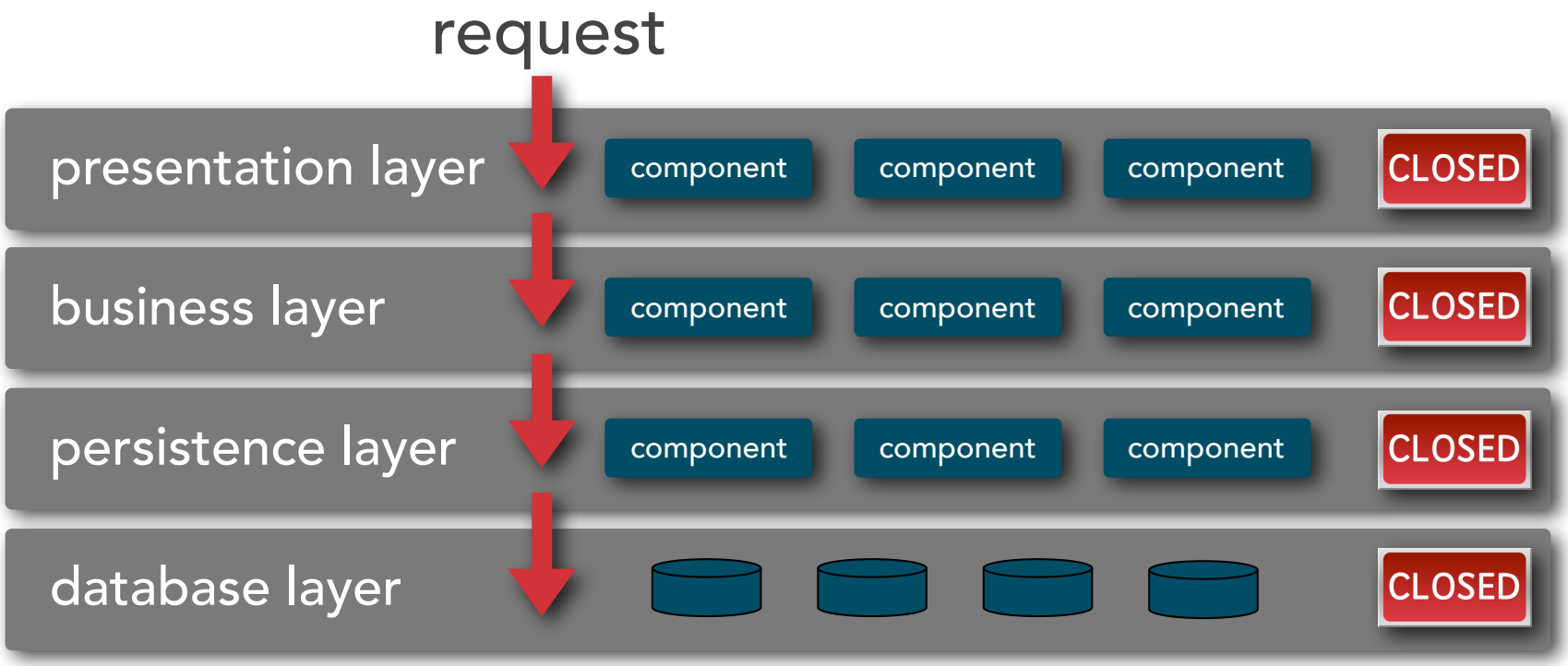
Layered Architecture



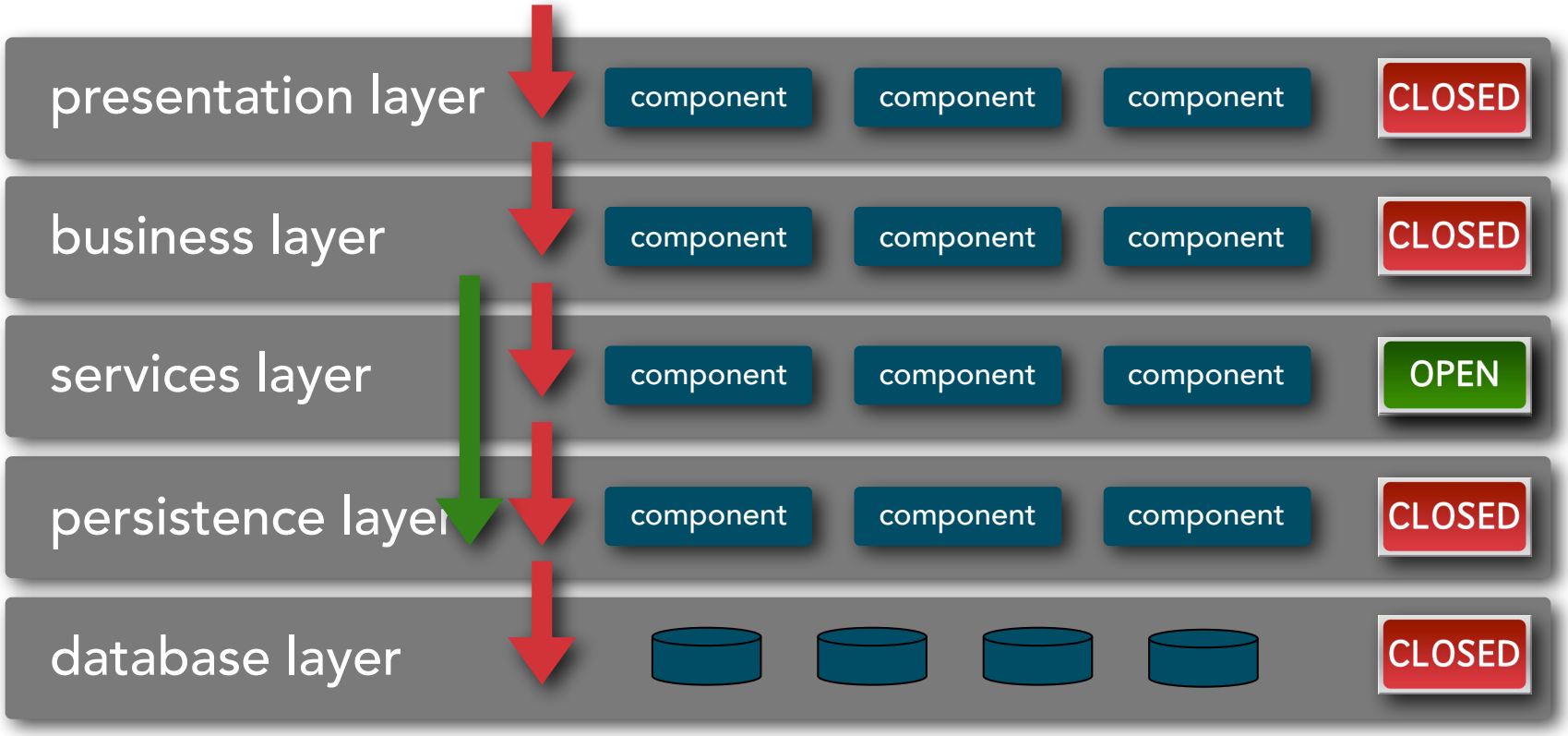
opportunities: 4

dimensions : 1

Layered Architecture



Layered Architecture



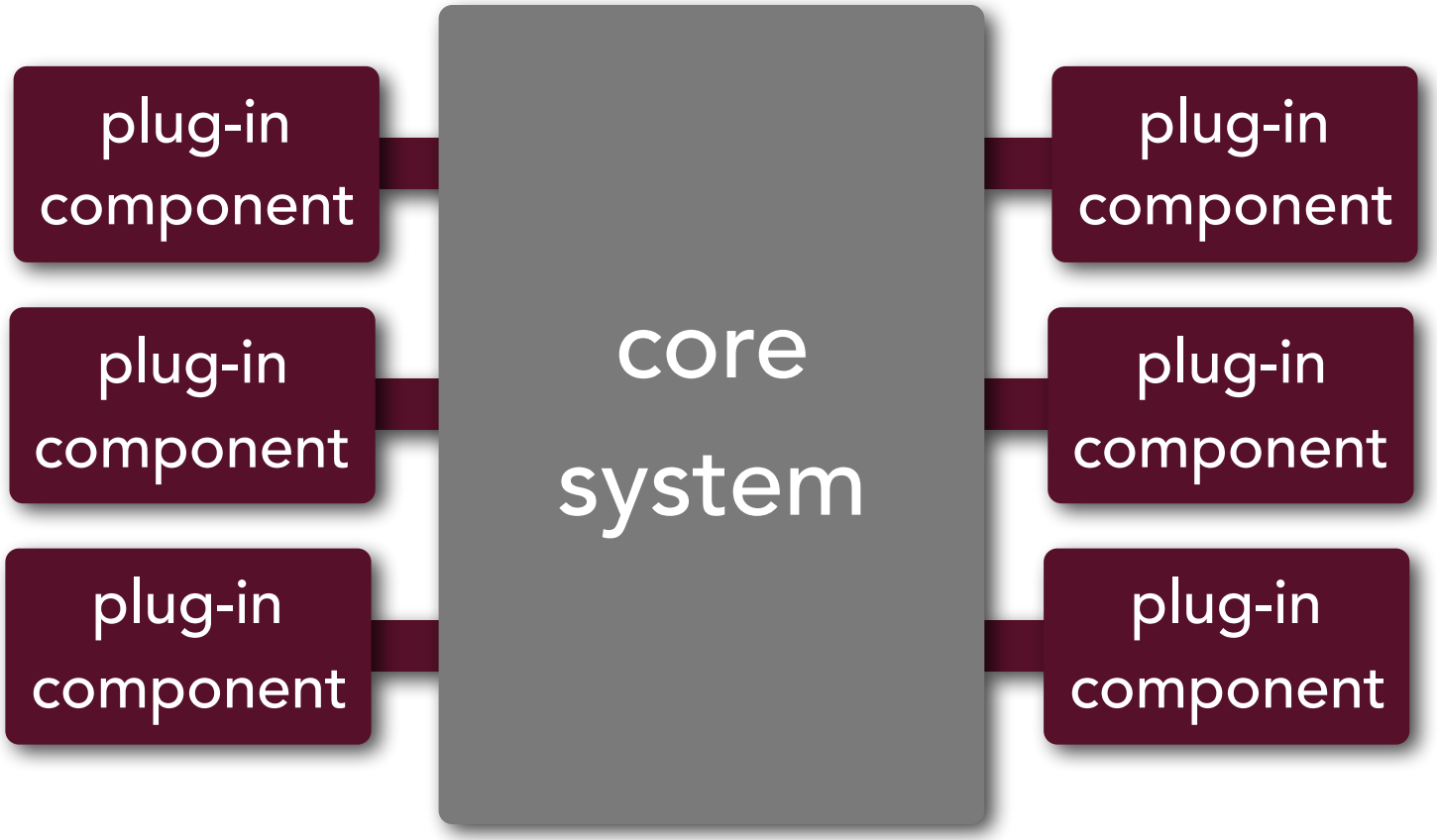
opportunities for evolution = L - (2 x L^o)

L : # of layers

L^o : # of open layers

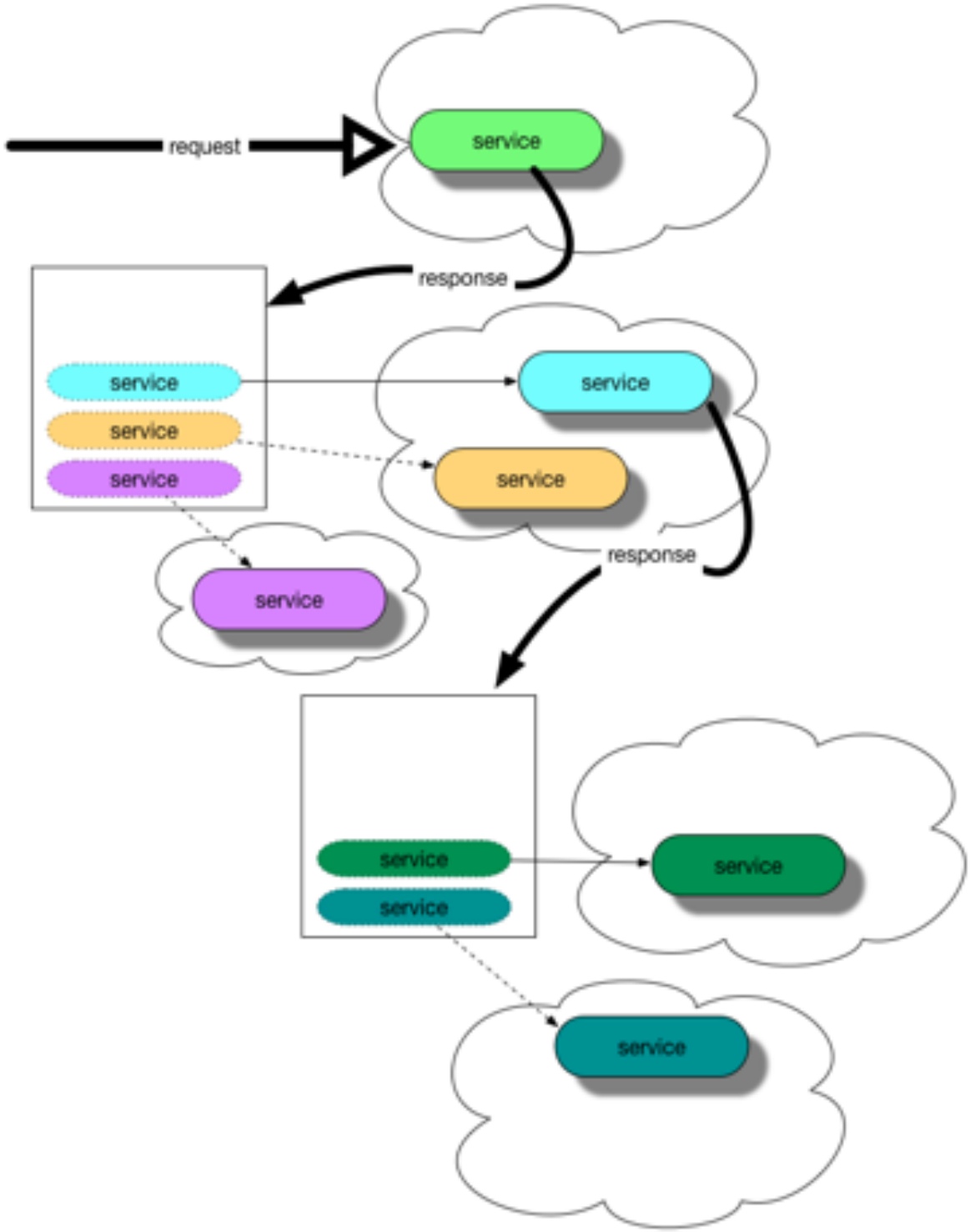
dimensions : 1

Microkernel



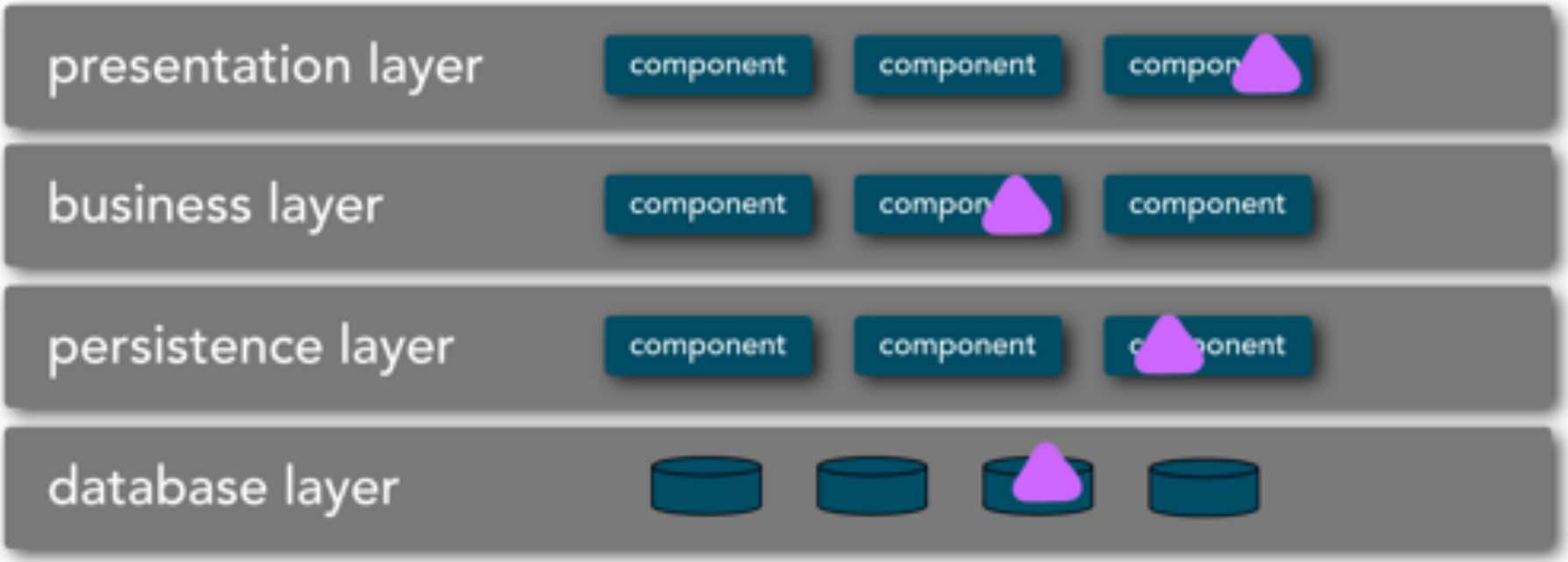
dimensions : 1

REST + HATEOS



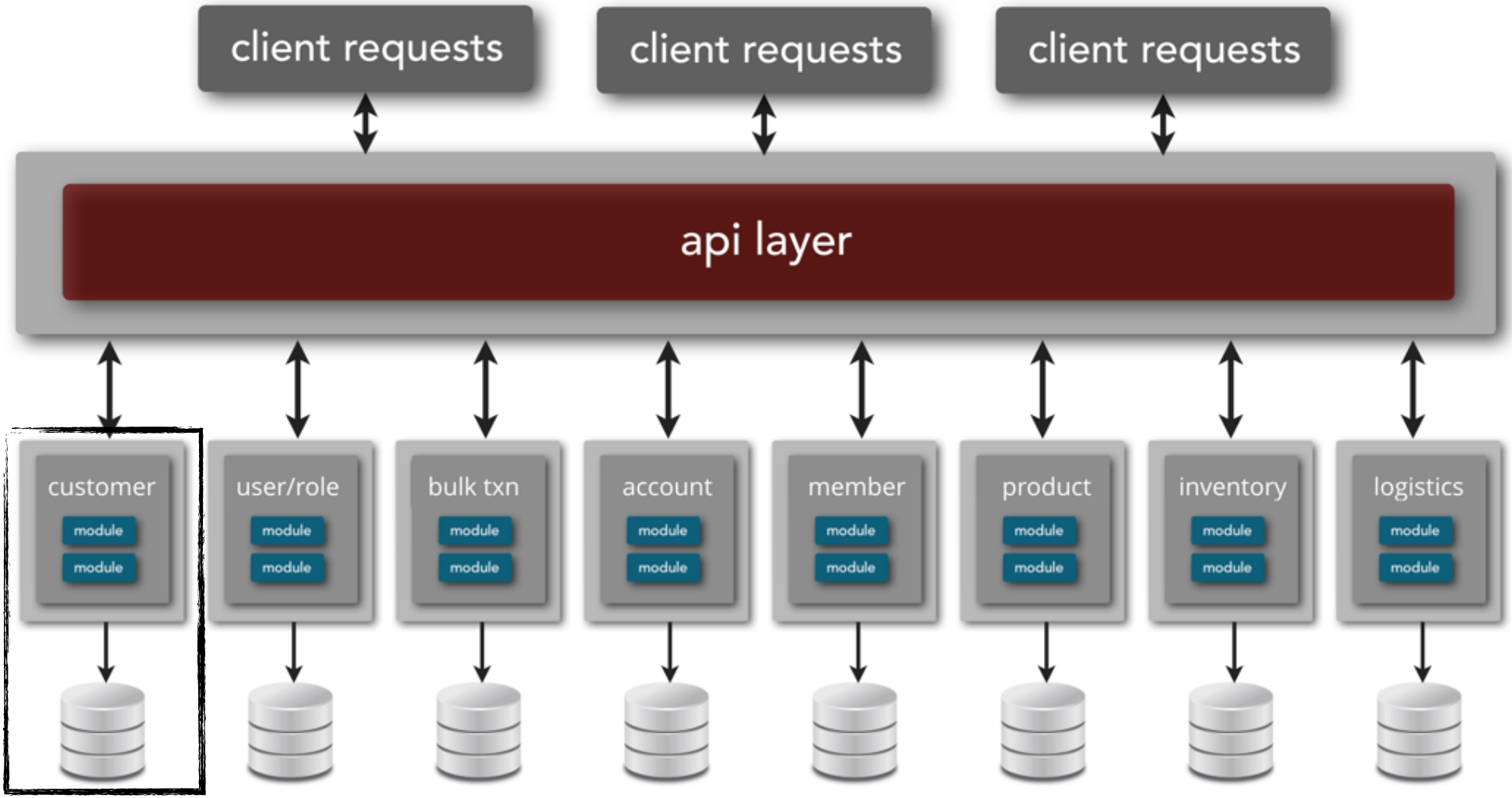
dimensions : 1

Domain Shift



domain dimensions :

Microservices



evolutionary architecture dimensions :

Definition :

evolutionary architecture

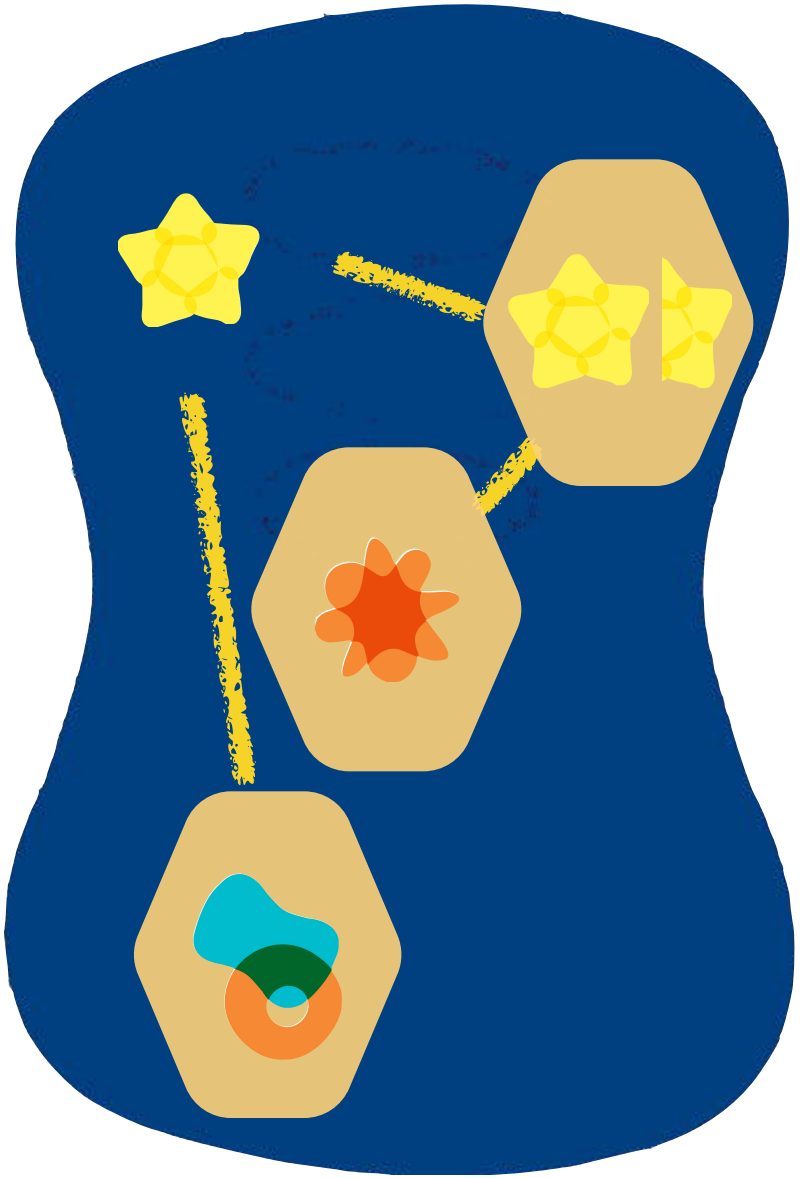
An evolutionary architecture supports continual and incremental change as a first principle along multiple dimensions.

Incremental Change

Components are *deployed*.

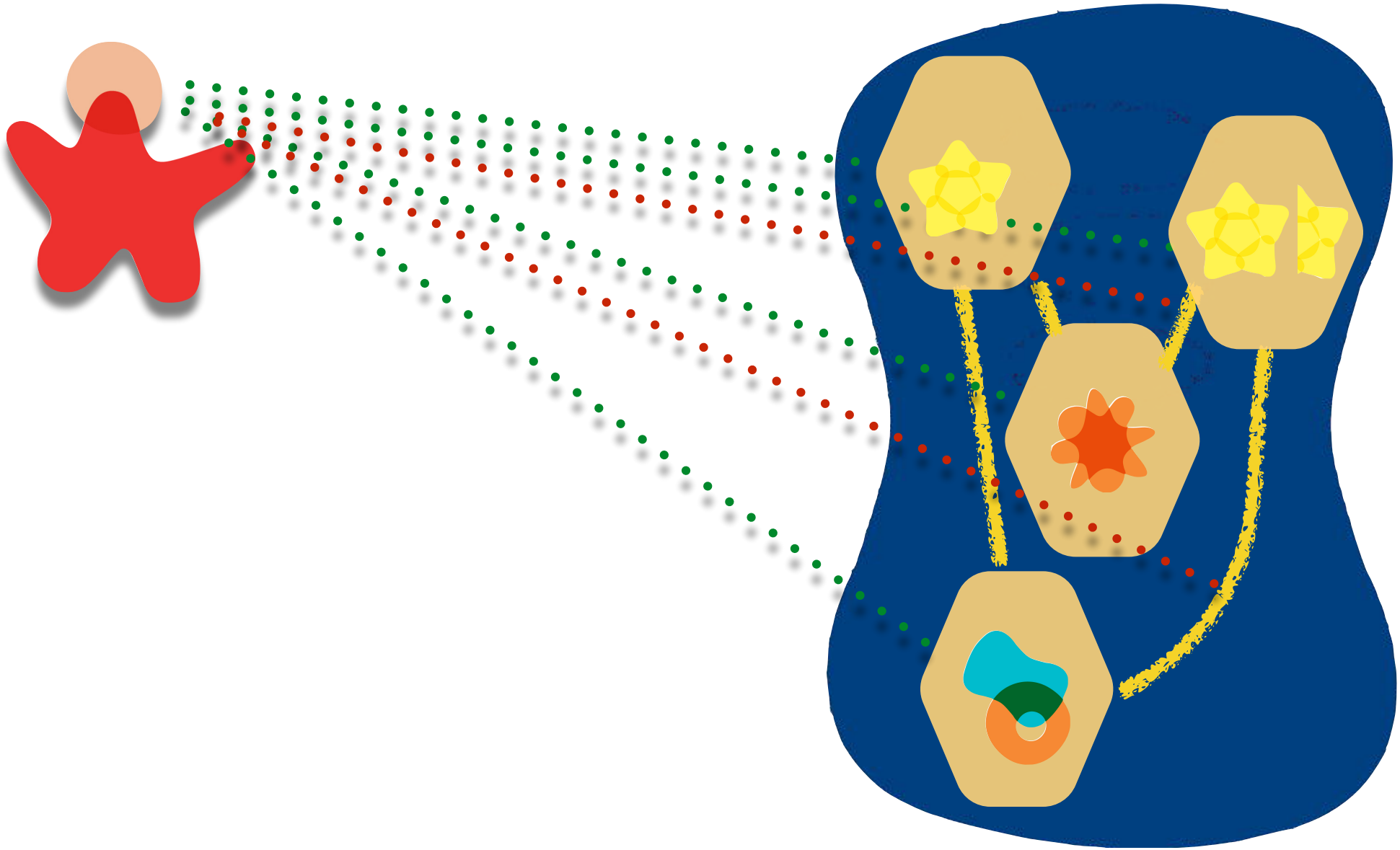
Features are *released*.

Applications consist of *routing*.



production

Incremental Change



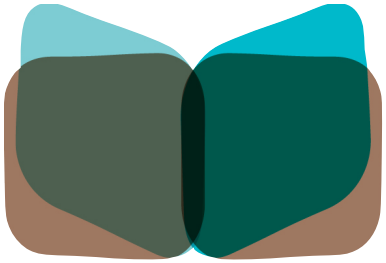
production

Definition :

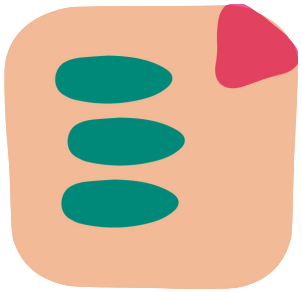
evolutionary architecture

An evolutionary architecture supports continual and incremental change as a first principle along multiple dimensions.

Agenda

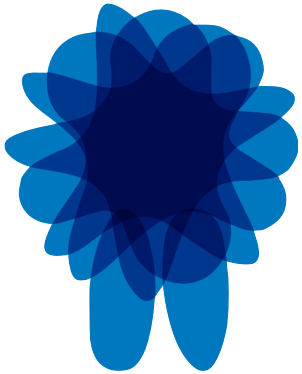


definition

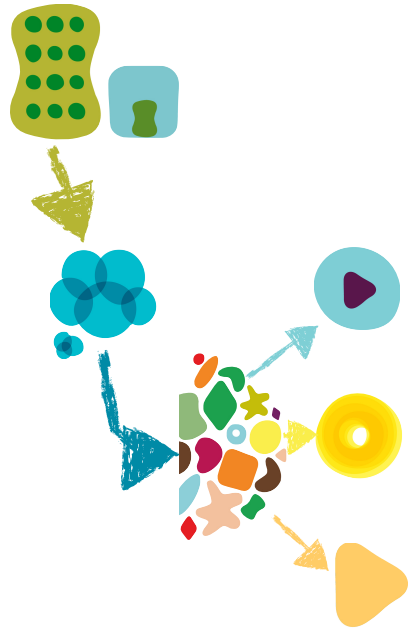


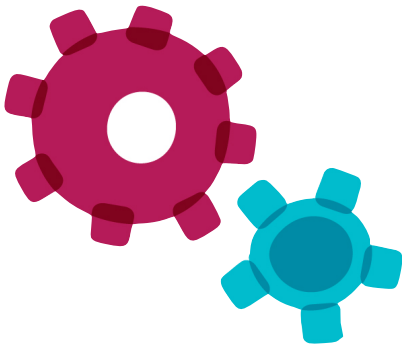
characteristics

principles

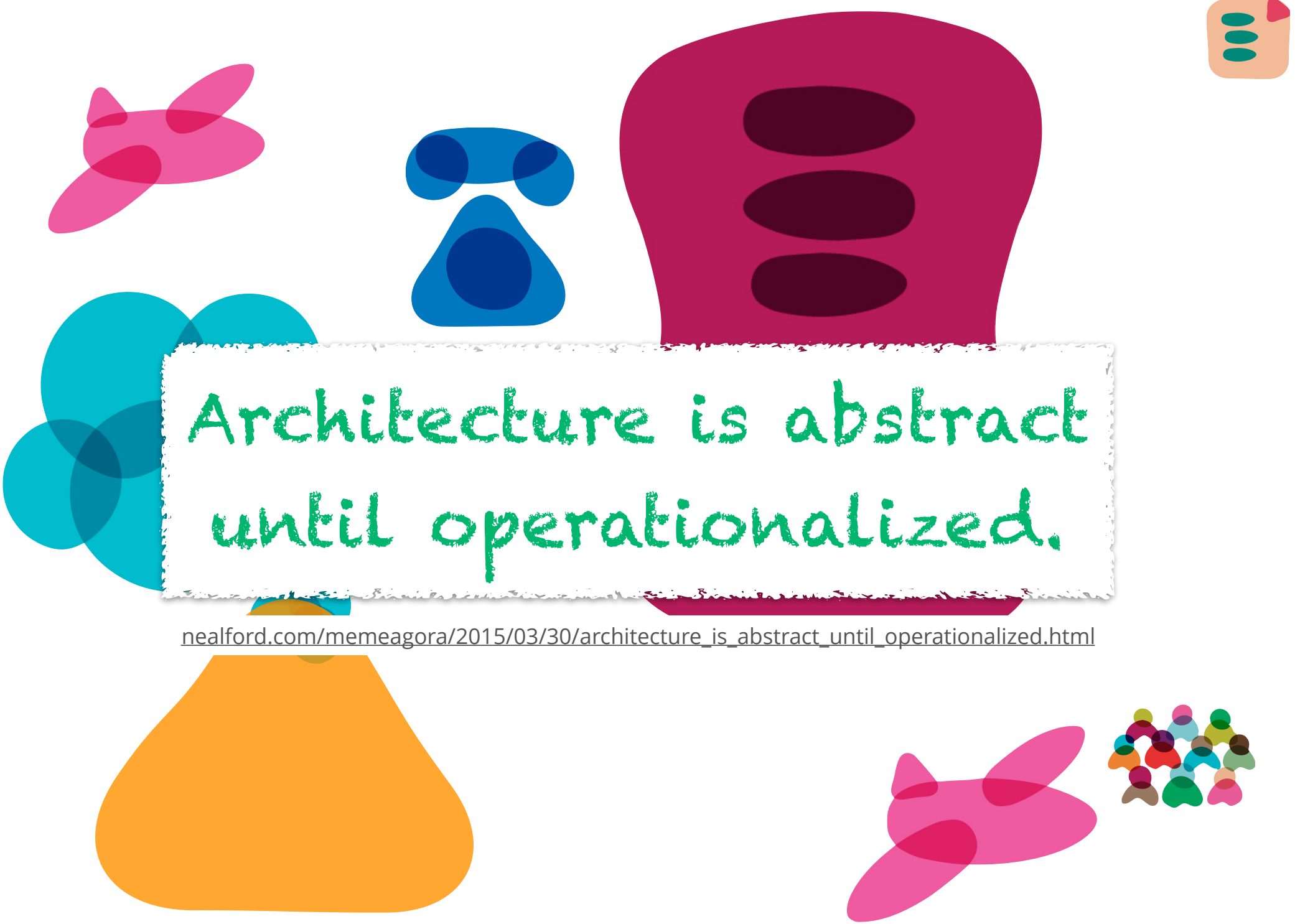


conclusion



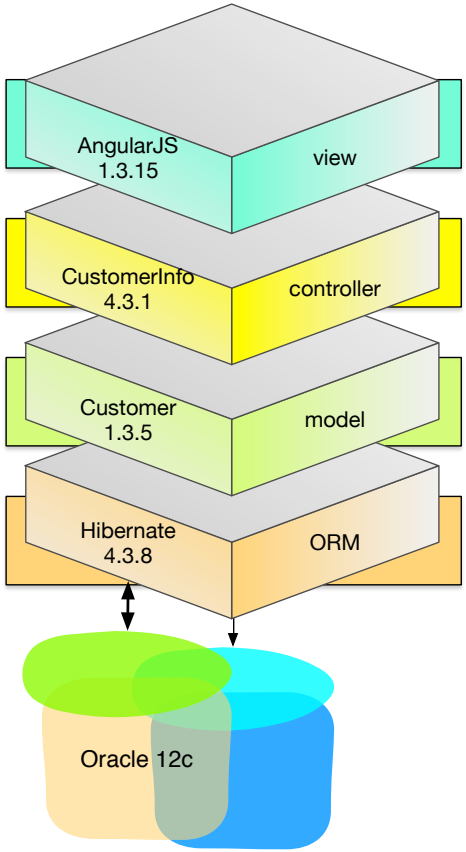


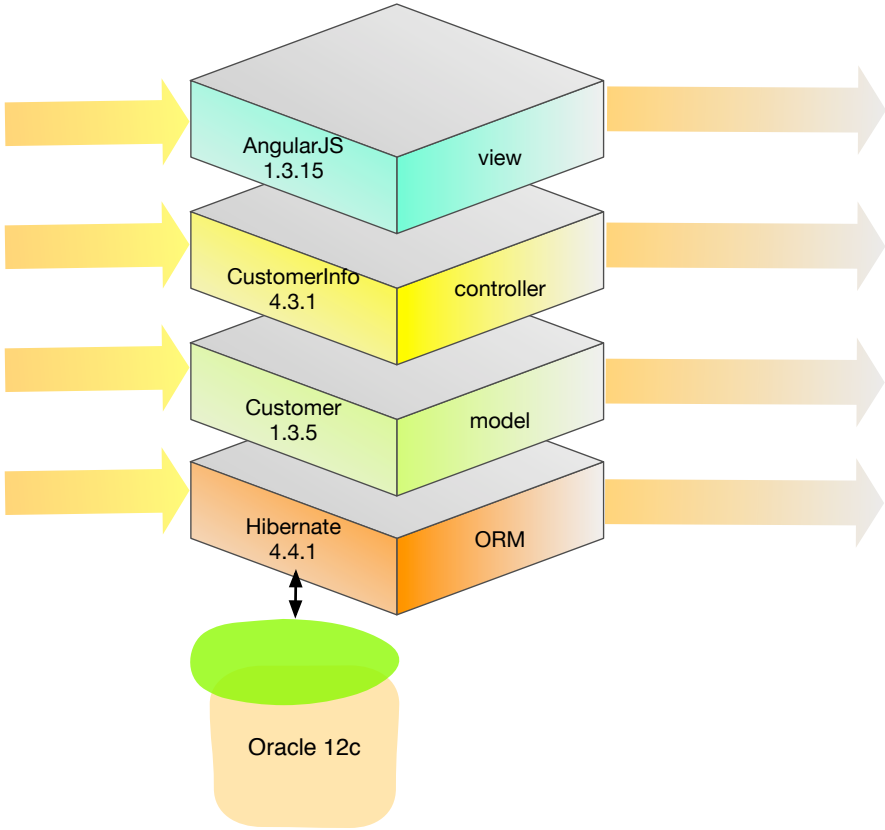
practices



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

Architecture is abstract until operationalized.

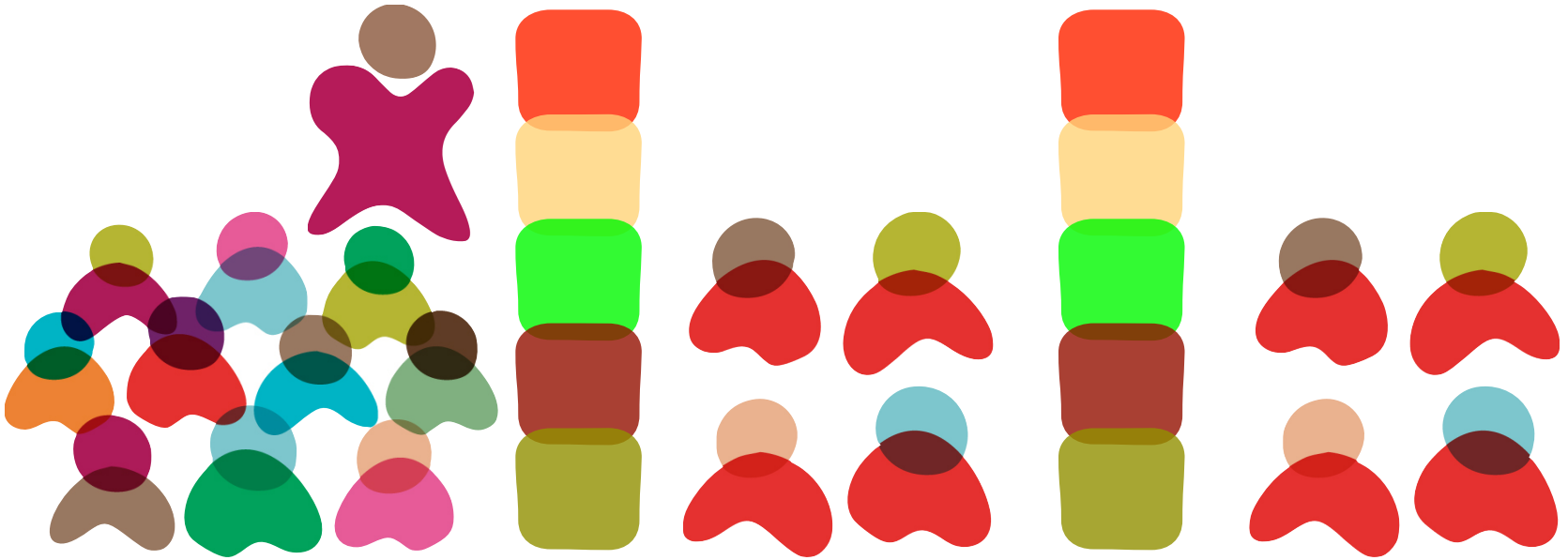




2D 3D 4D

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

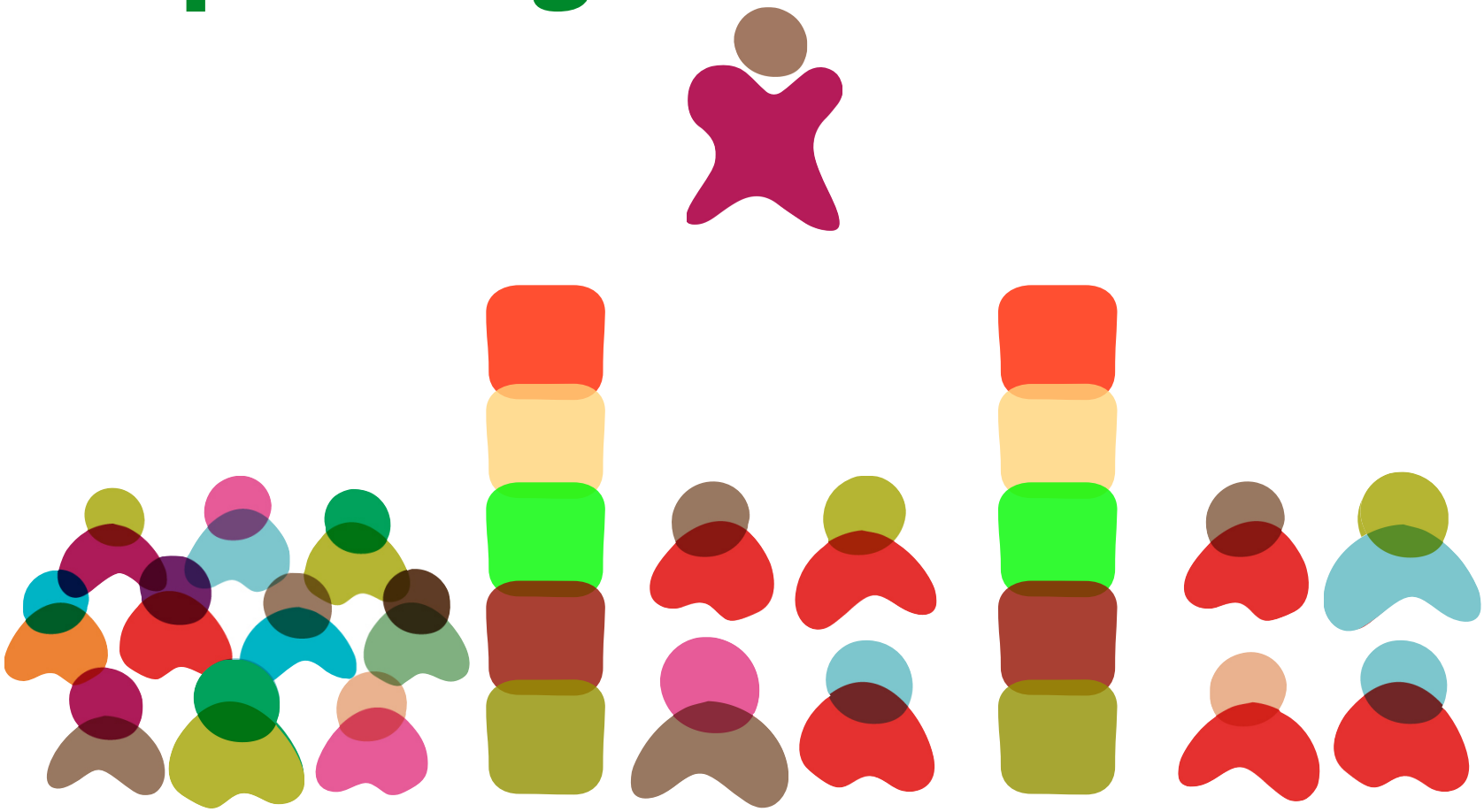
Expanding Role of Architect



engineering DBAs operations

pre-CD

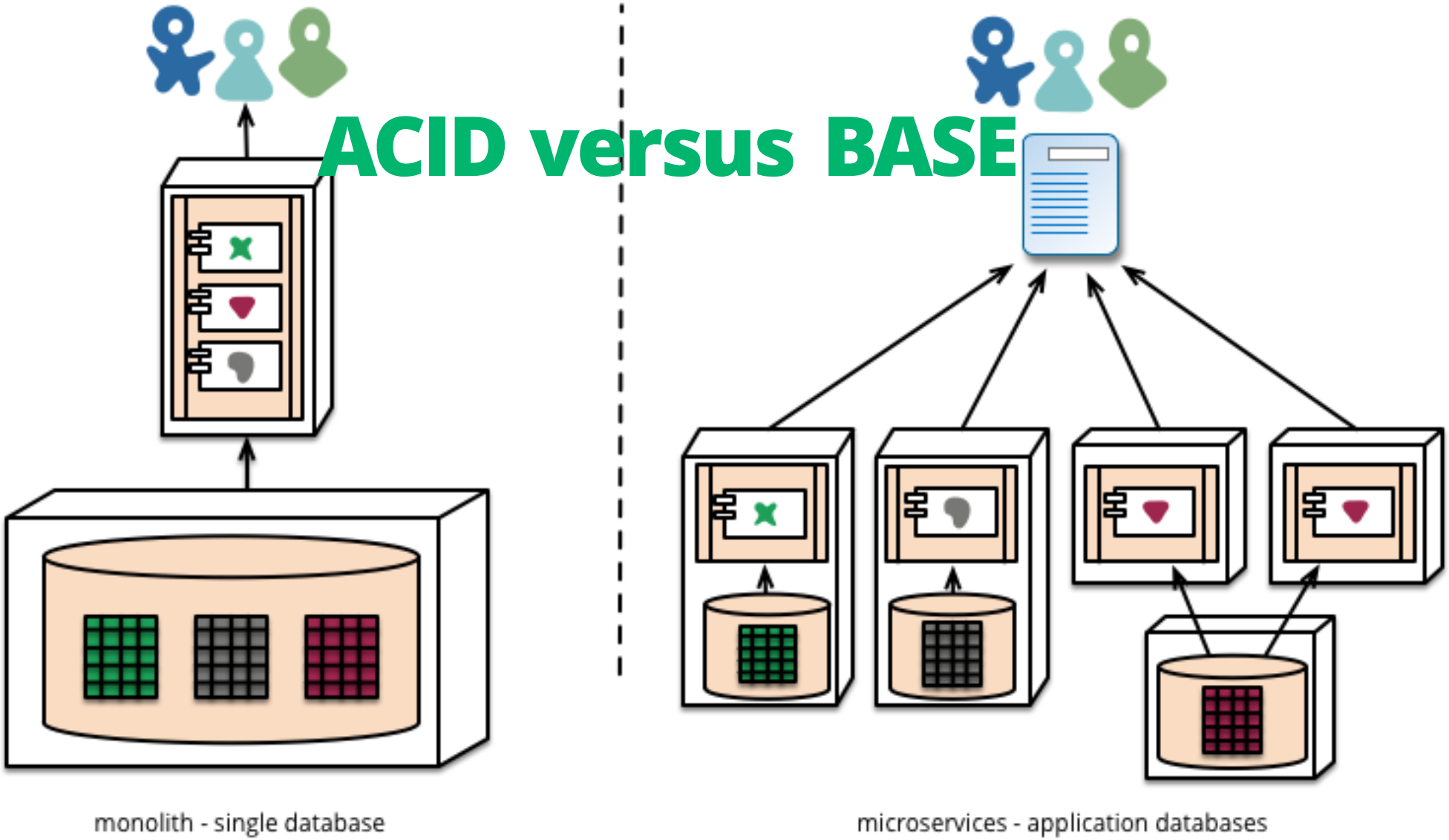
Expanding Role of Architect



post-CD

Decentralized Data Management

ACID versus BASE



monolith - single database

microservices - application databases

ACID versus BASE

ACID

- **A**tomic
- **C**onsistent
- **I**solated
- **D**urable

BASE

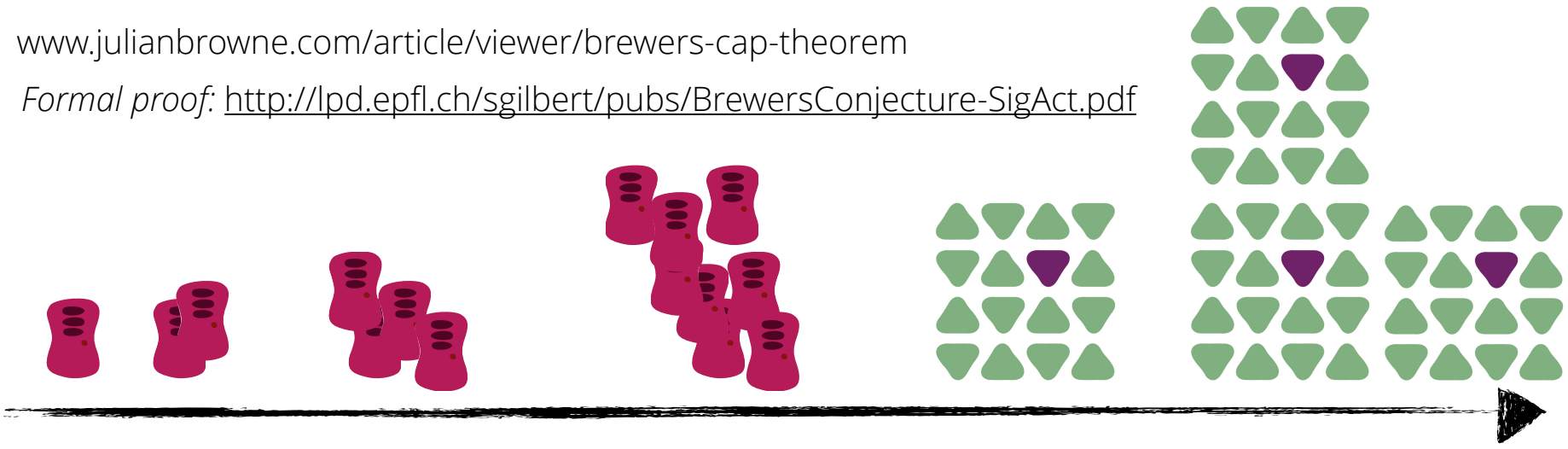
- **B**asic **A**vailability
- **S**oft-state
- **E**ventual Consistency

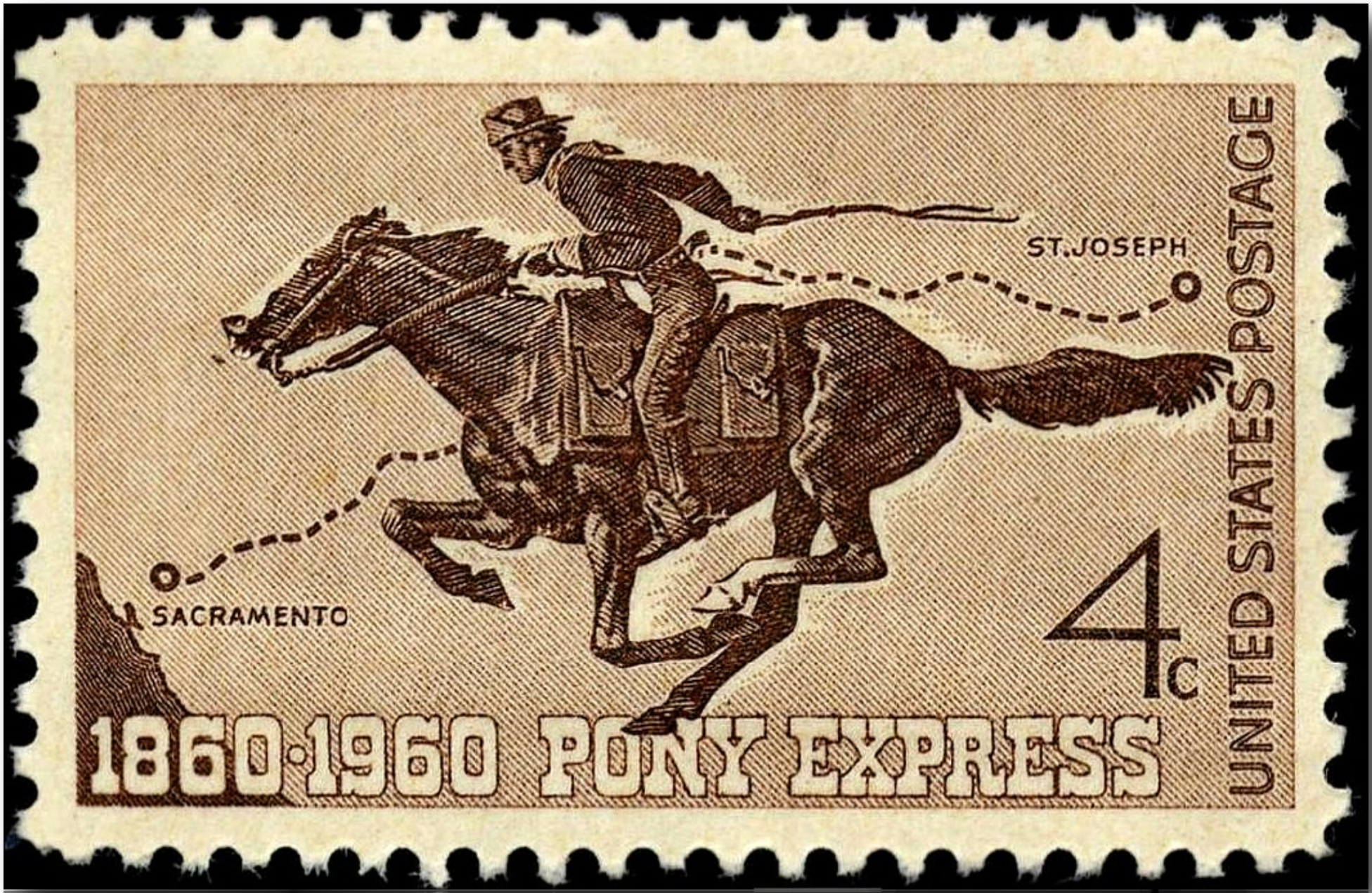
As your system becomes more distributed, prefer BASE to ACID...

...because CAP Theorem

www.julianbrowne.com/article/viewer/brewers-cap-theorem

Formal proof: http://lpd.epfl.ch/sgilbert/pubs/BrewersConjecture-SigAct.pdf

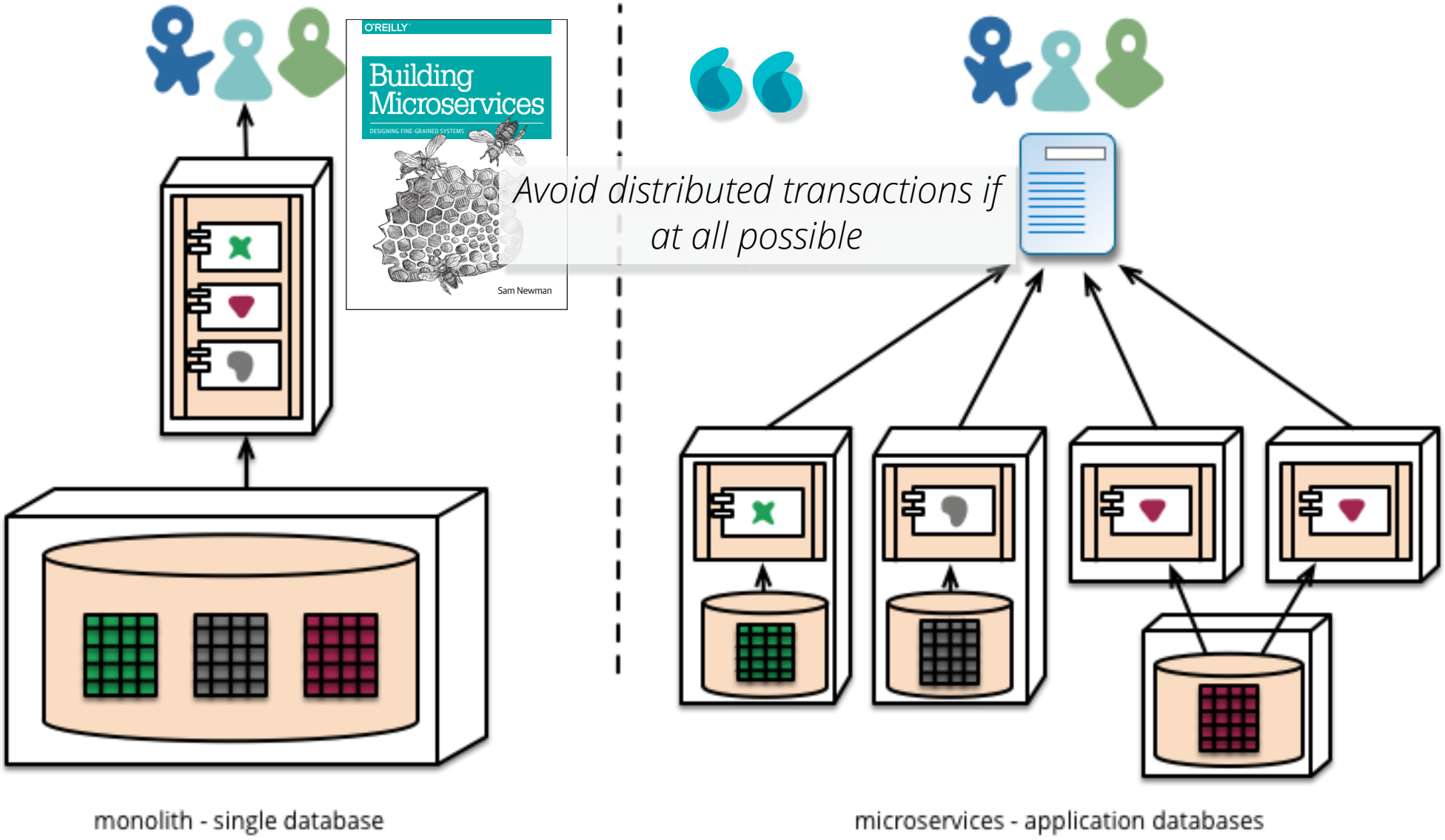




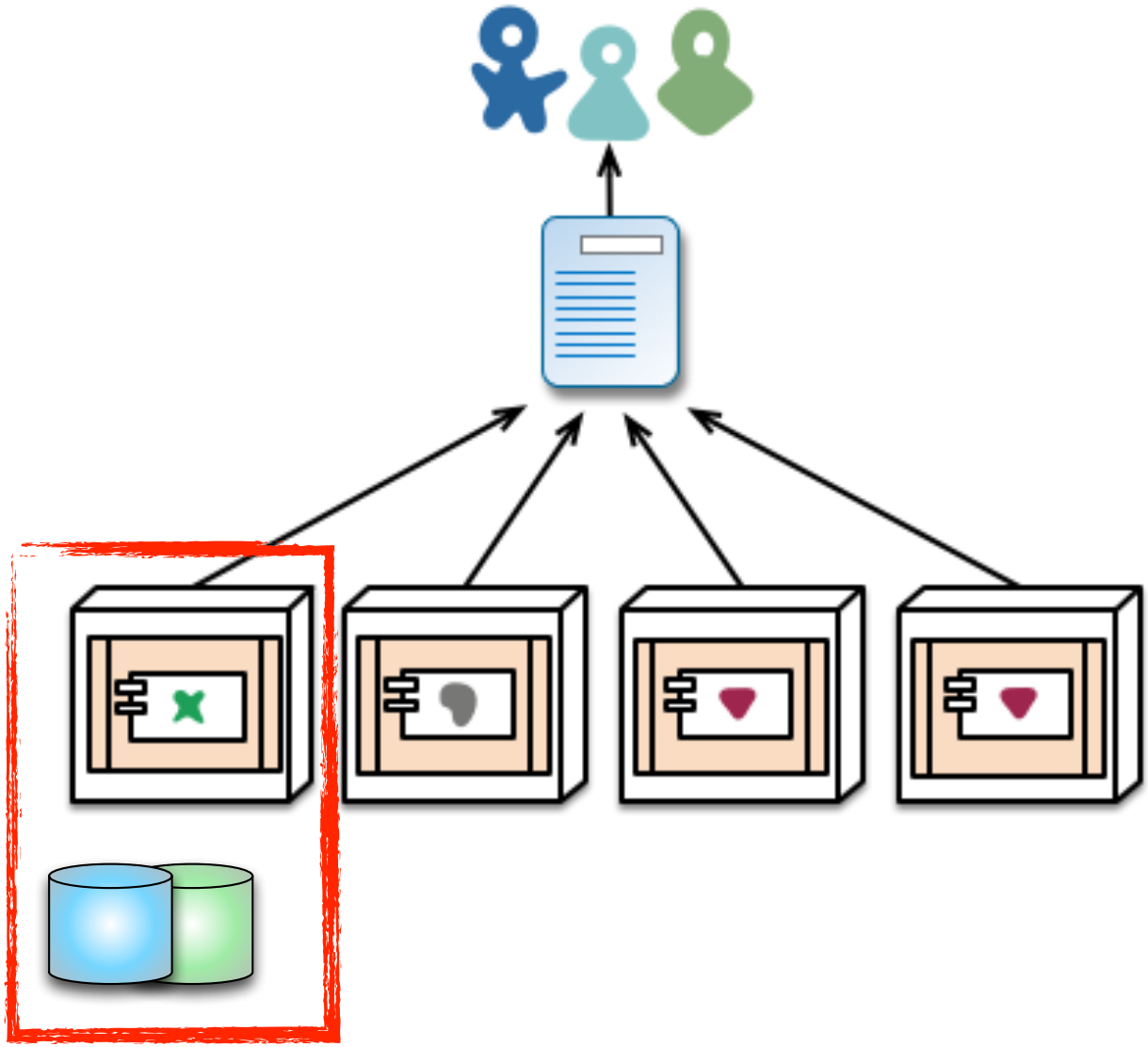
http://highscalability.com/blog/2013/5/1/myth-eric-brewer-on-why-banks-are-base-not-acid-availability.html

http://www.eaipatterns.com/docs/IEEE_Software_Design_2PC.pdf

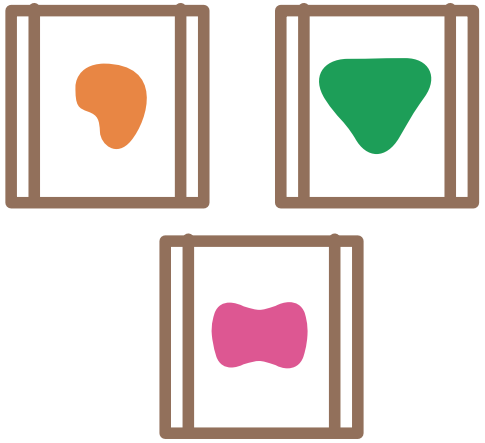
Decentralized Data Management

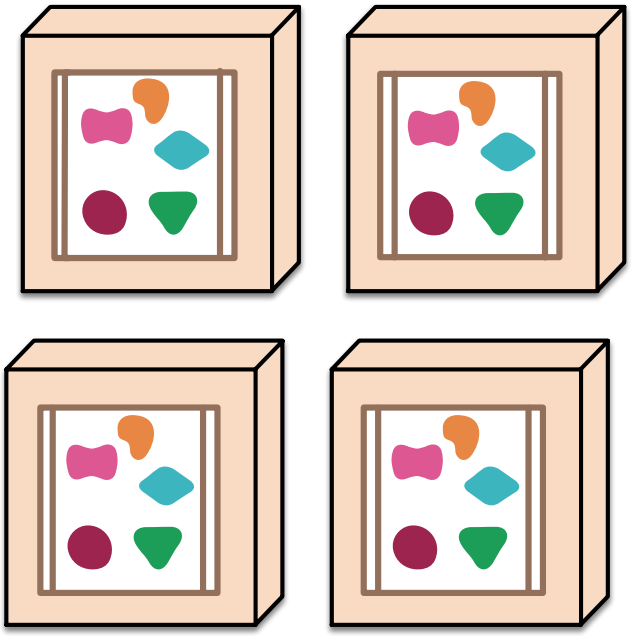


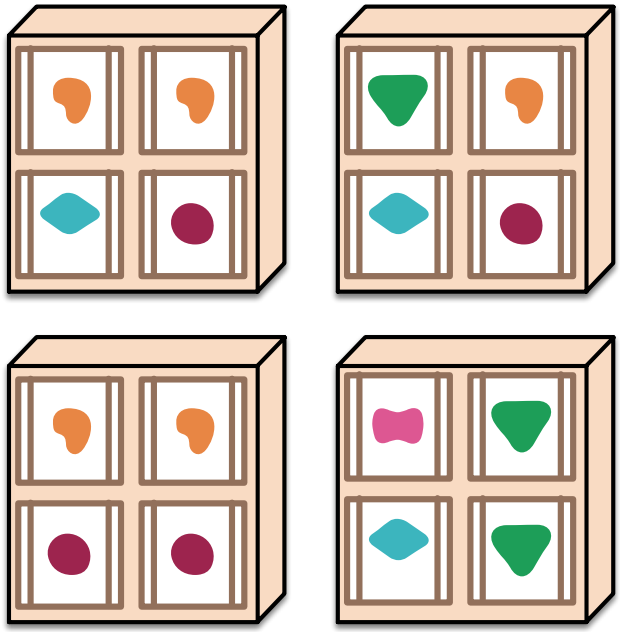
Decentralized Governance



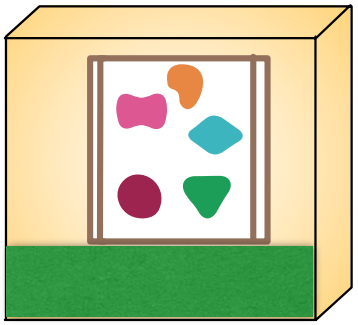
Decentralized Governance

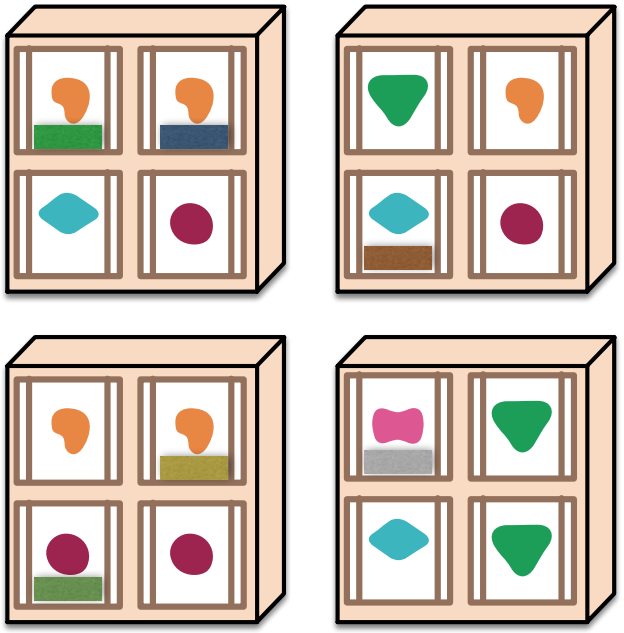






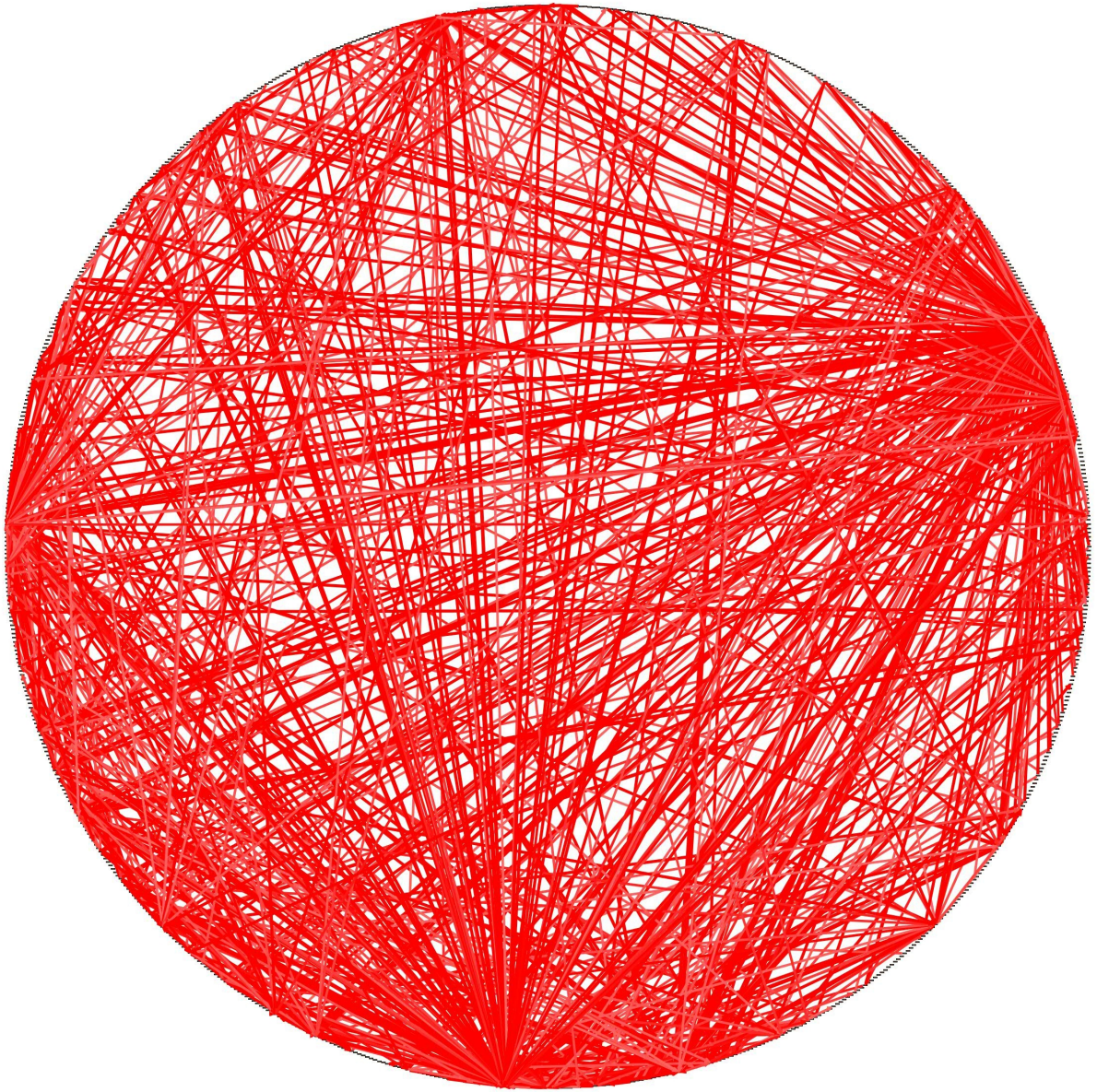
Decentralized Governance



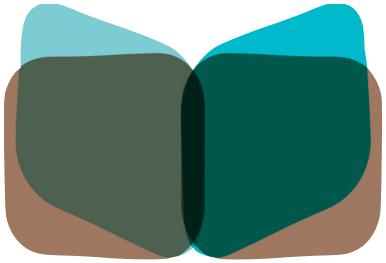


Enterprise architects suffer from less pressure to make the correct choice(s) in microservice architectures.

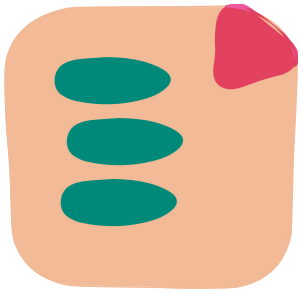
Coupling



Agenda

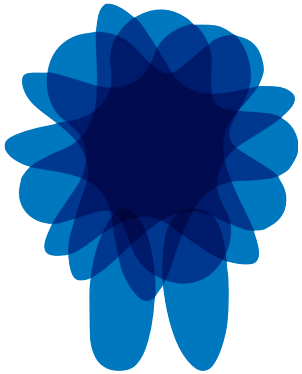


definition

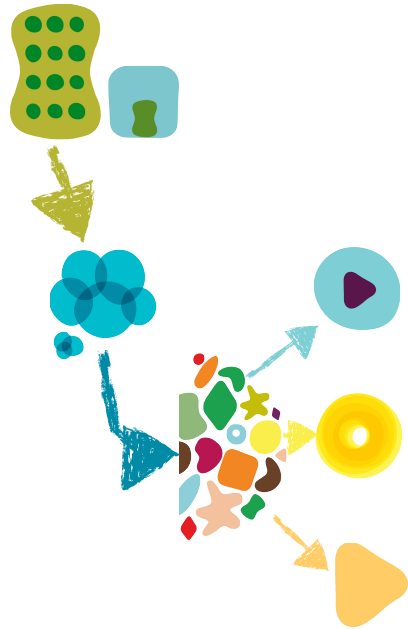


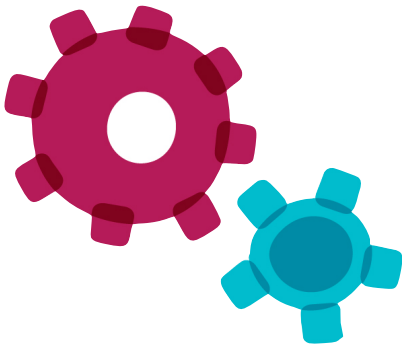
characteristics

principles



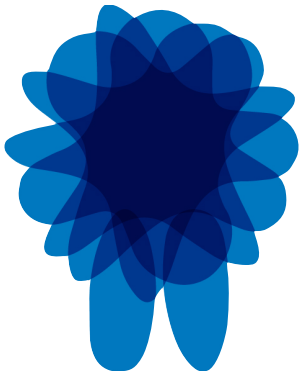
conclusion



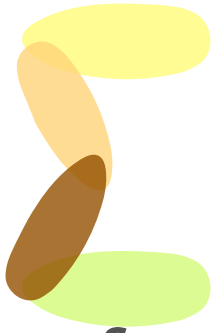


practices

principles



architecture

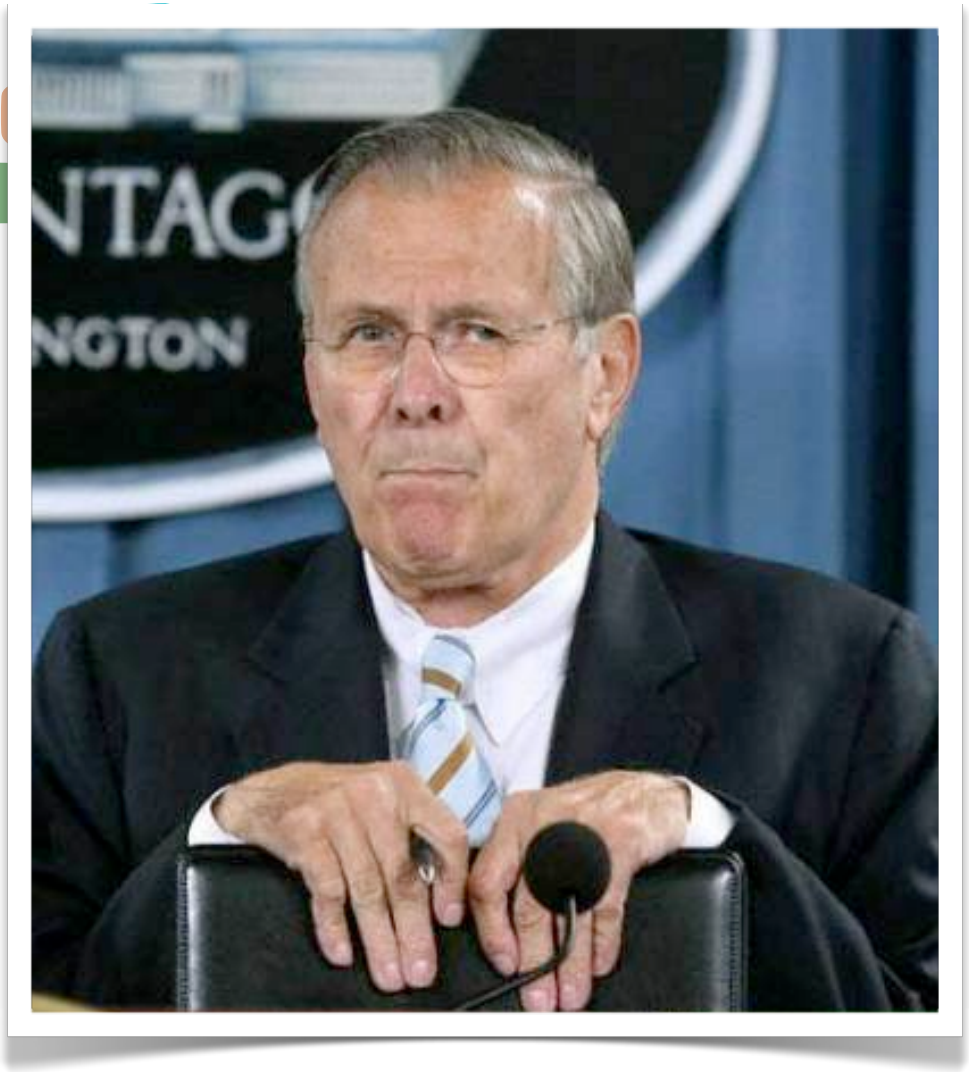


fitness functions

people

process

anti-principles



"There are known unknowns.

That is to say there are things that we now know we don't know.

But there are also unknown unknowns.

There are things we do not know we don't know."

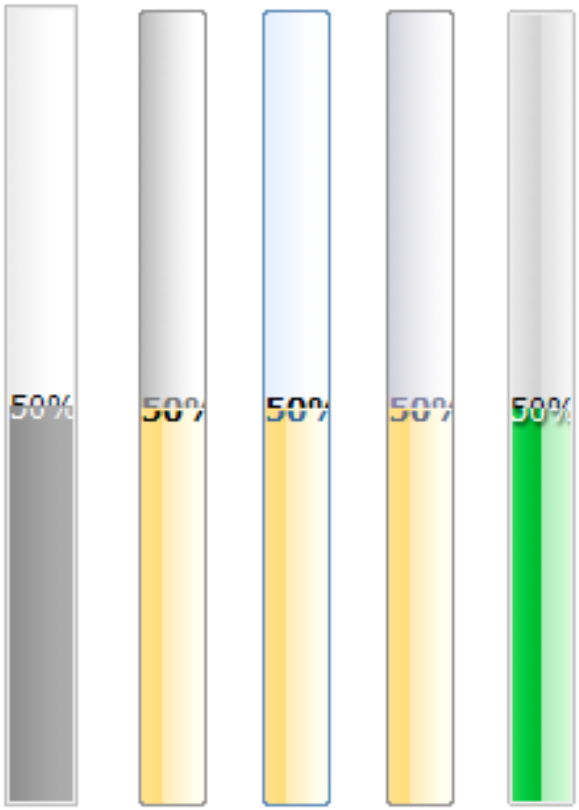
There are things we do not know we don't know."

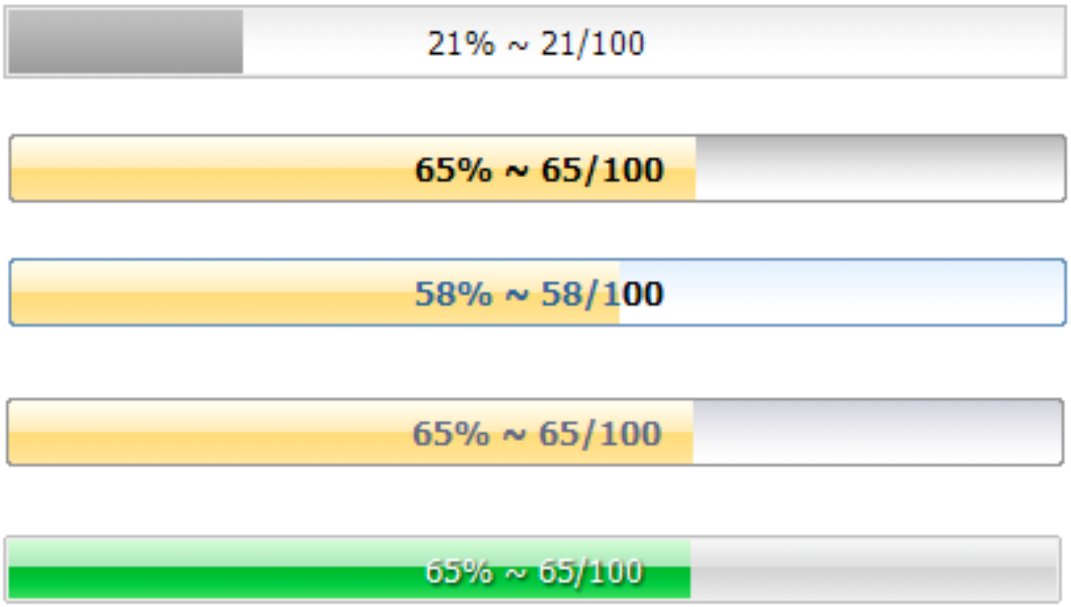
Unknown unknowns

prefer pro/reactive to predictive

evolution of asynchronous messaging

Progress Bars & Async Upload

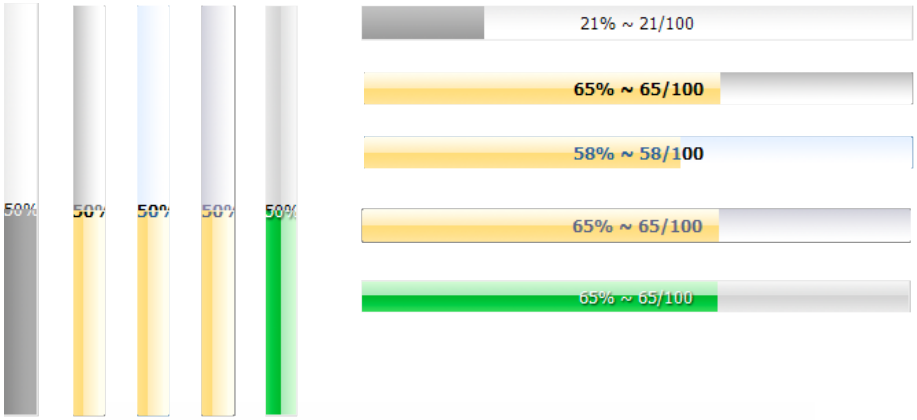


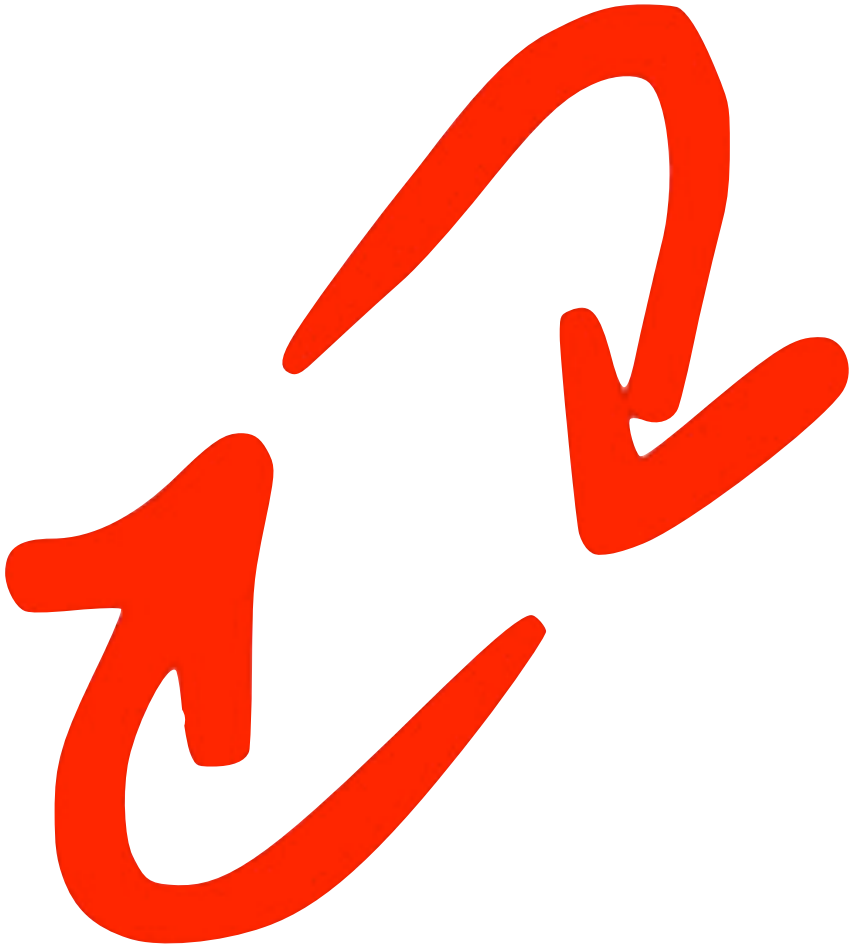


backgrounDrb

http://backgroundrb.rubyforge.org/

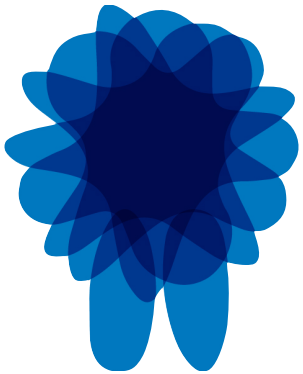
3 Kinds:



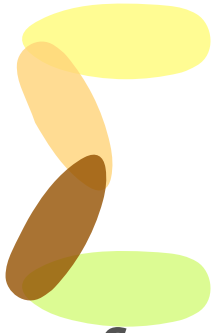



(starling)
switch to a real
messaging queue

principles



architecture



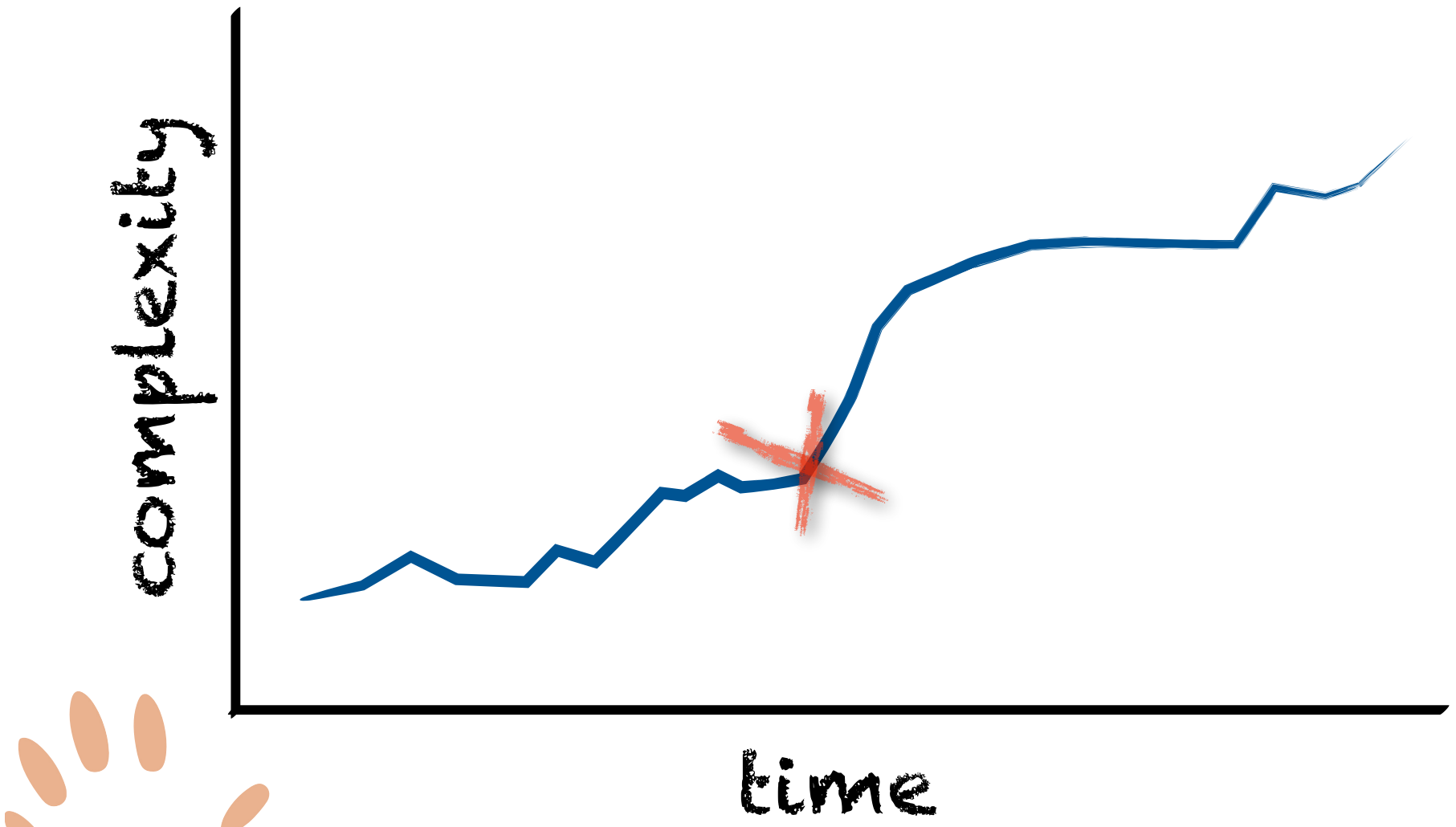
fitness functions

people

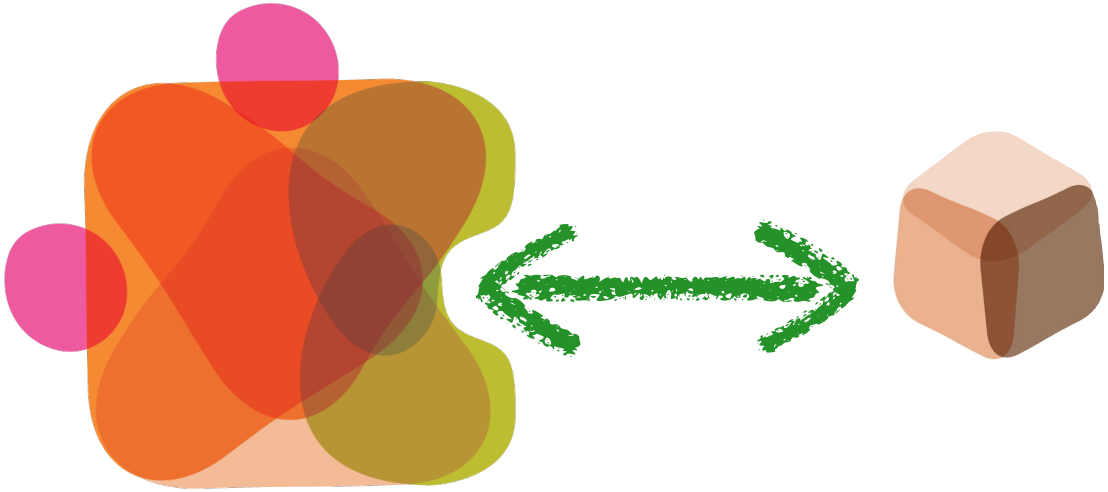
process

anti-principles

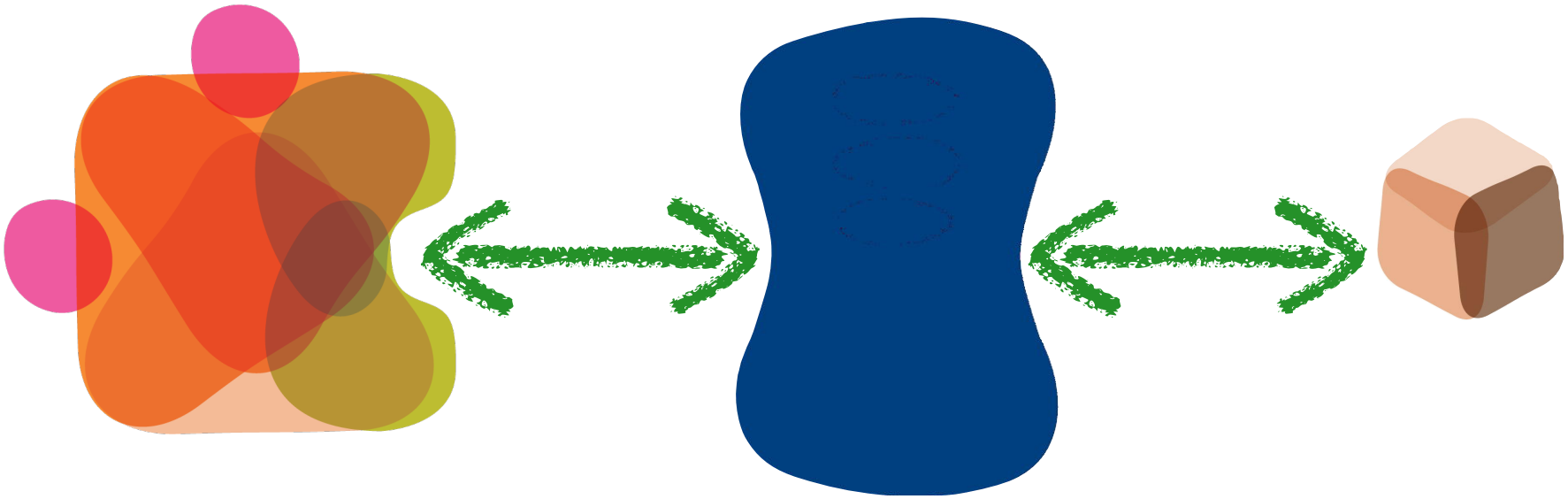
Last Responsible Moment



Last Responsible Moment



Last Responsible Moment

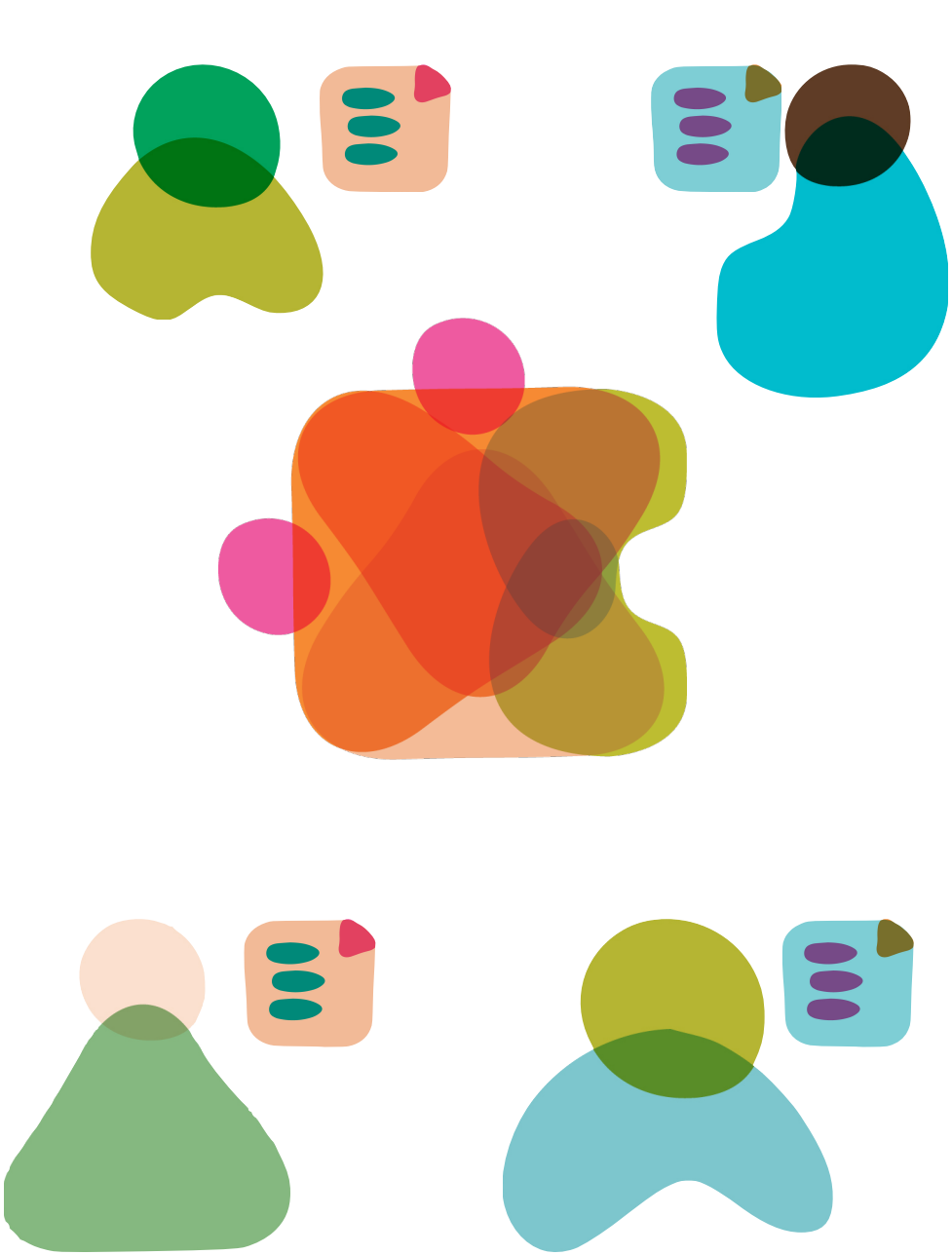


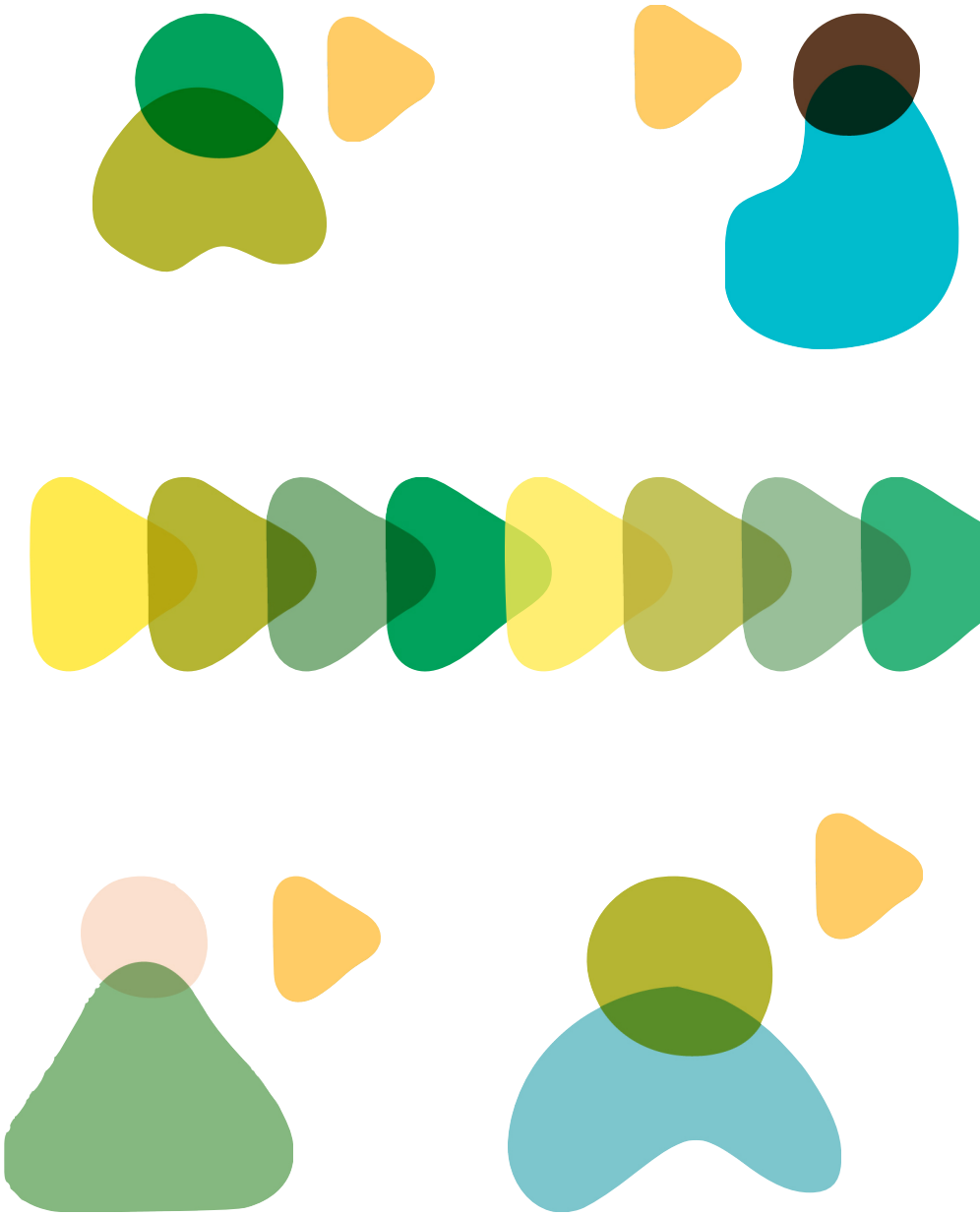
DDD's "Anti-corruption Layer"

this is not an excuse to abstract all the things!

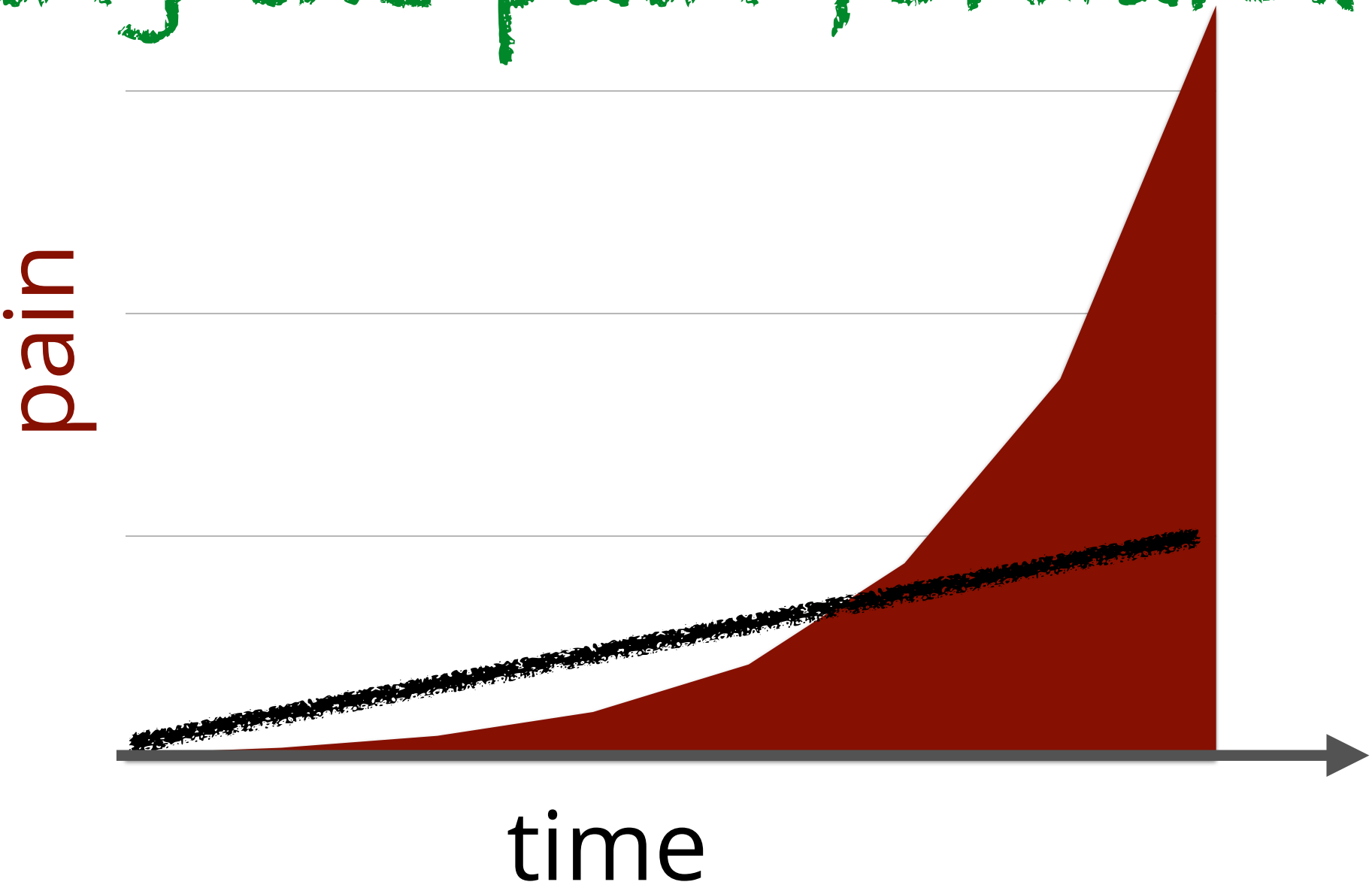
prefer pro/reactive to predictive

Bring the Pain Forward

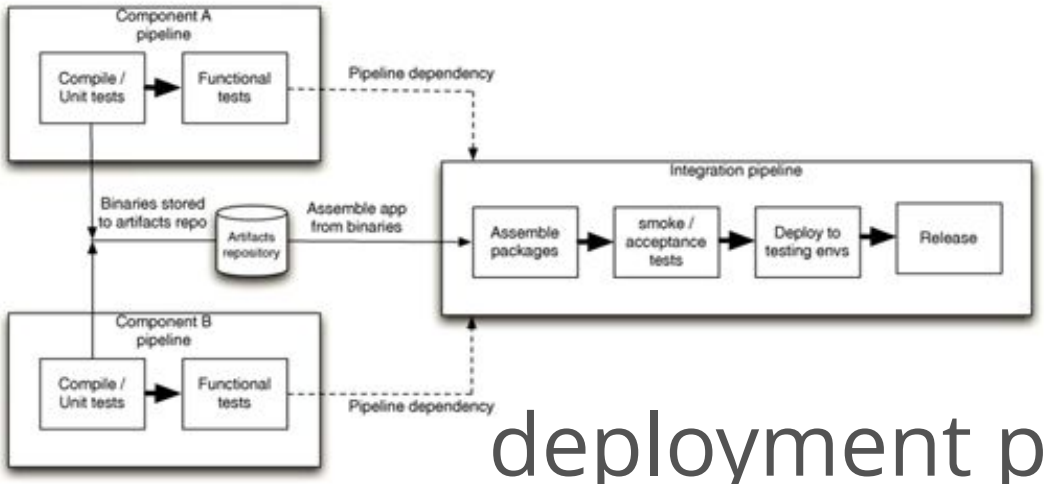




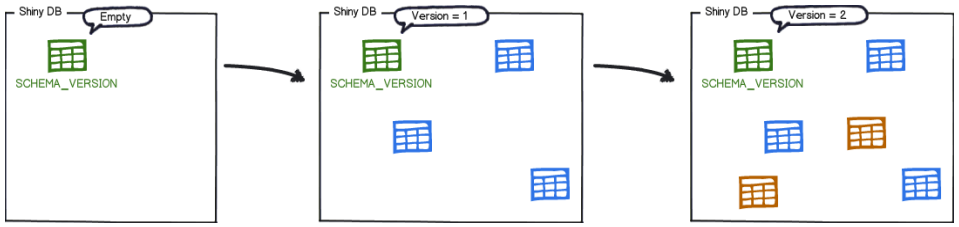
Bring the pain forward.



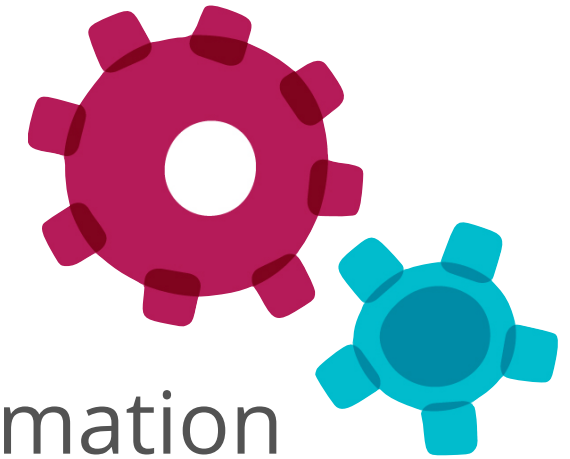
Bring the Pain Forward



deployment pipelines

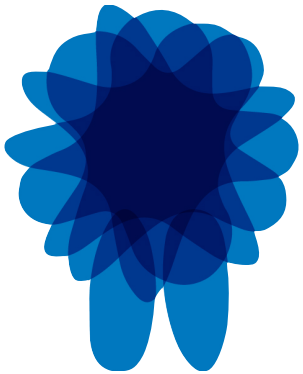


database migrations/ refactoring

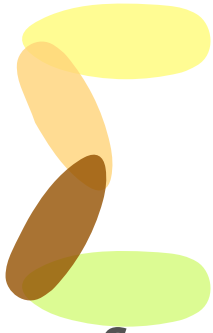


automation

principles



architecture



fitness functions

people

process

anti-principles

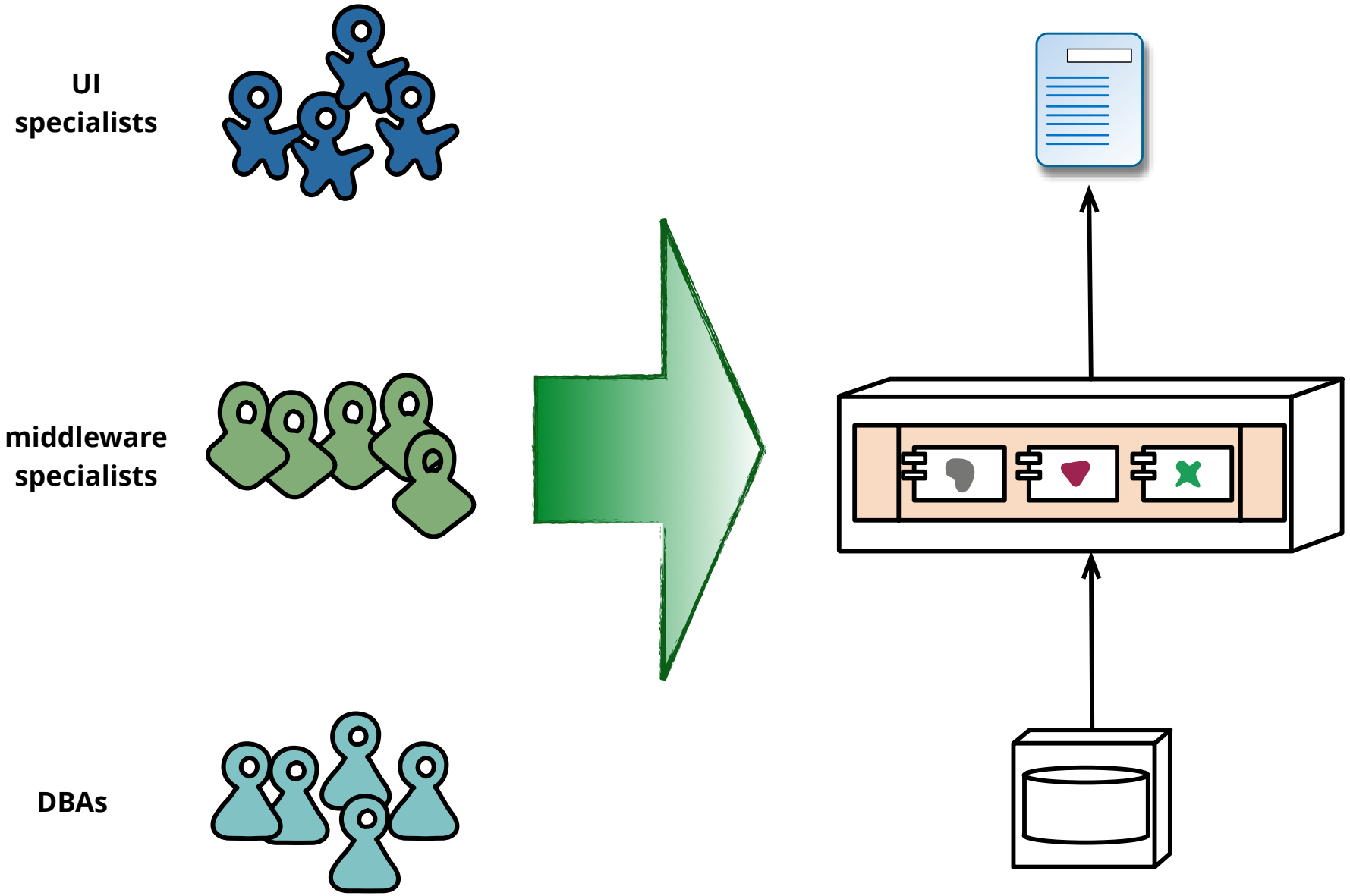
Conway's Law

"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations"

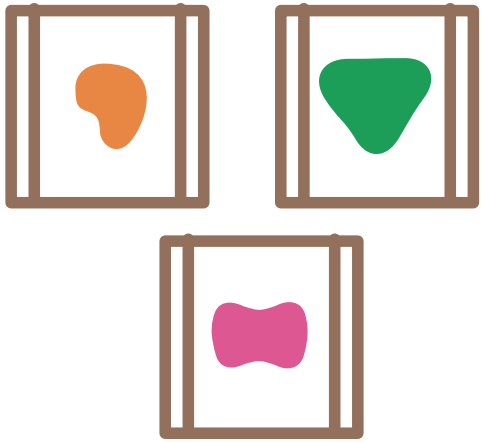
Melvin Conway, 1968

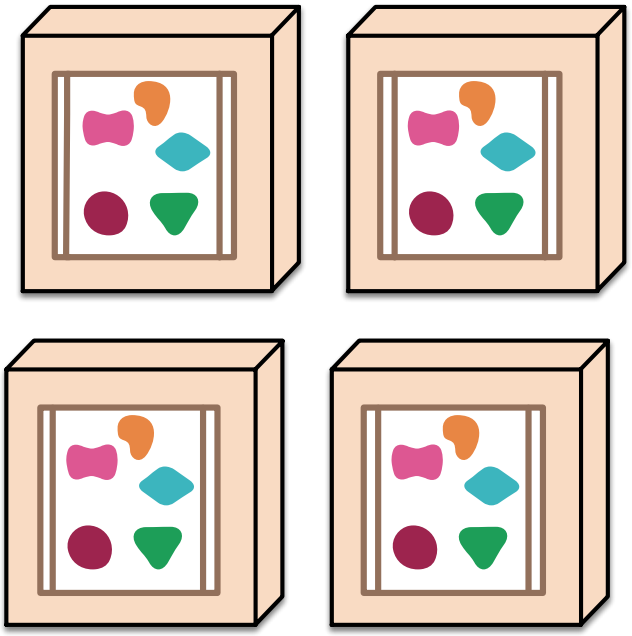
en.wikipedia.org/wiki/Conway%27s_law

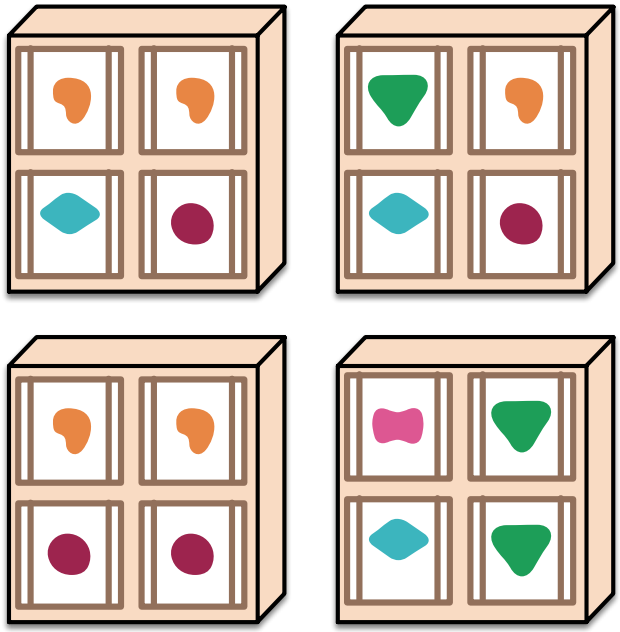
Conway's Law Side Effect

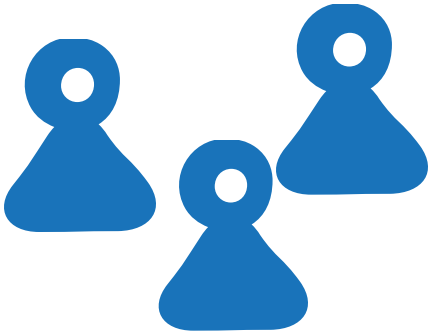


Monoliths vs. Microservices

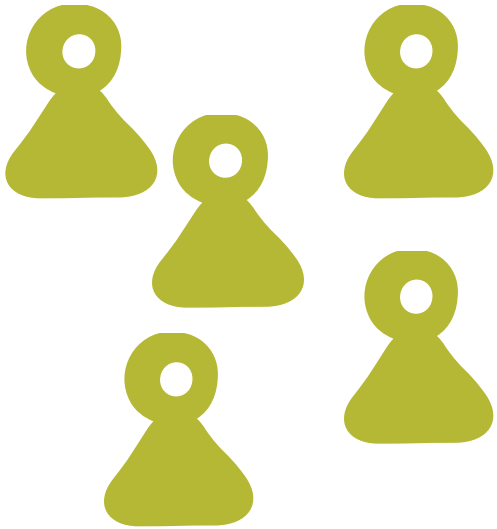




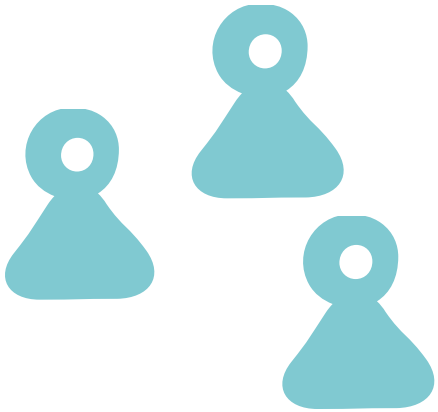




user interface



server-side

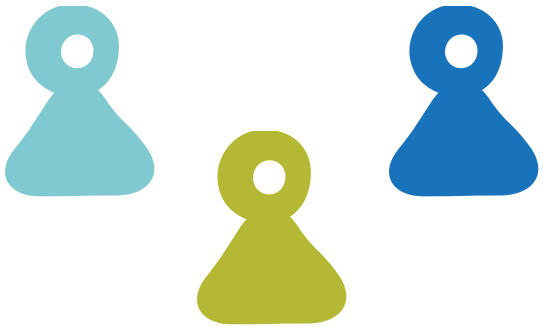


DBA

Orders

Shipping

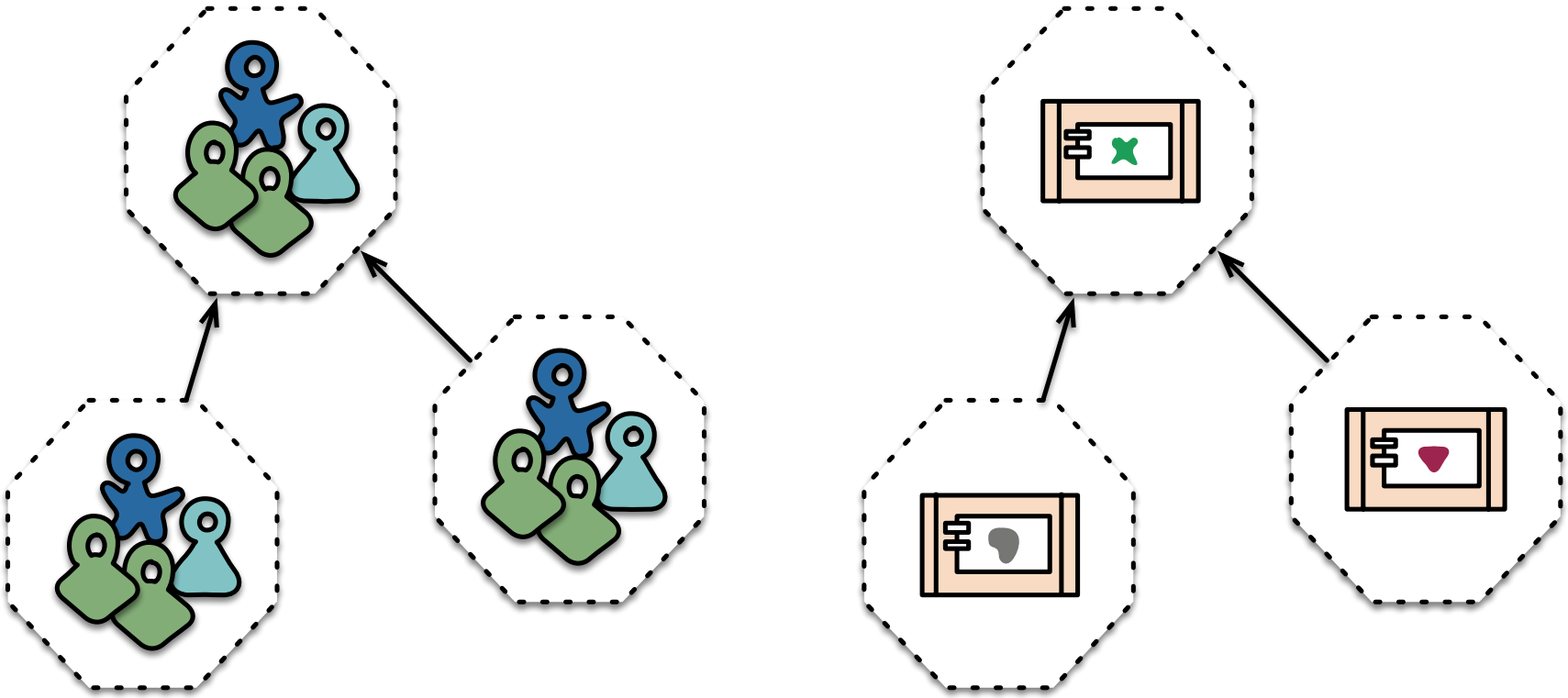
Catalog



Inverse Conway Maneuver

Domain/Architecture Isomorphism

Inverse Conway Maneuver

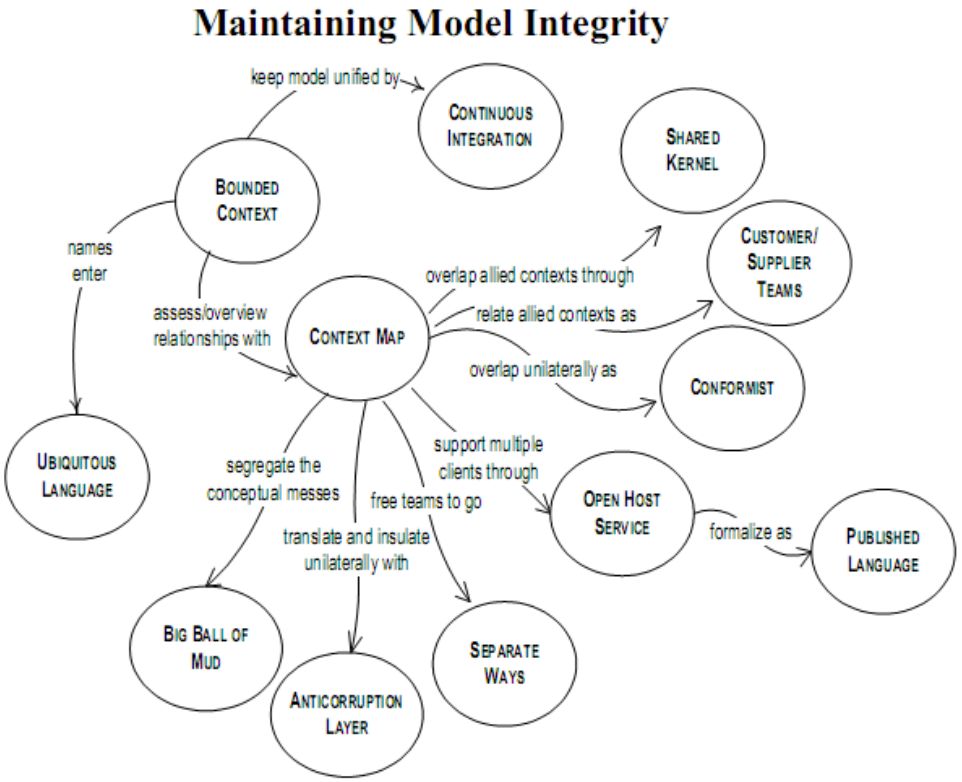


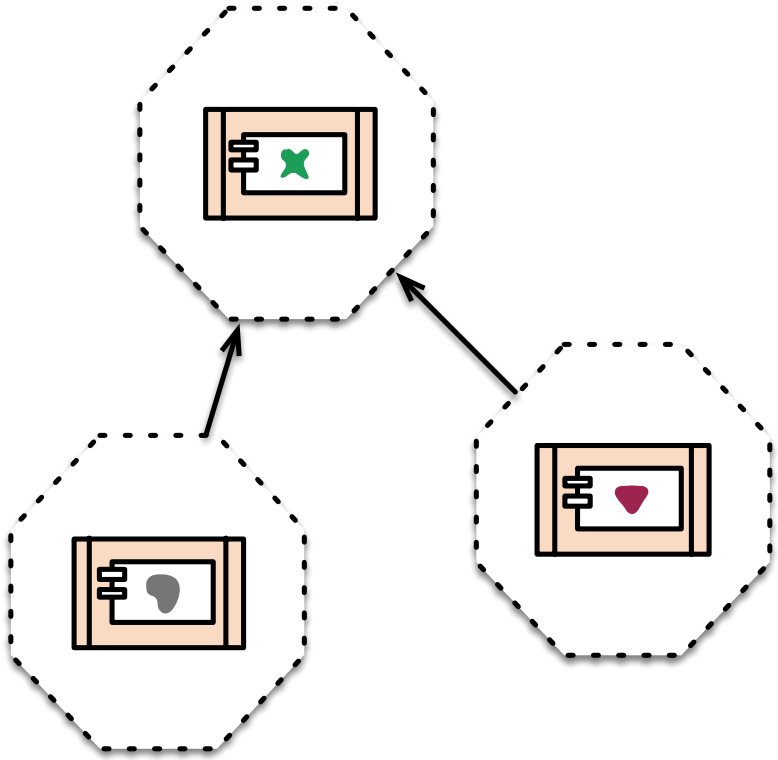
cross-functional teams…

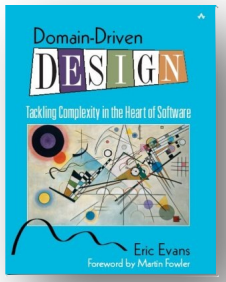
…organized around business capabilities

Because Conway's Law!

DDD's "bounded context"...



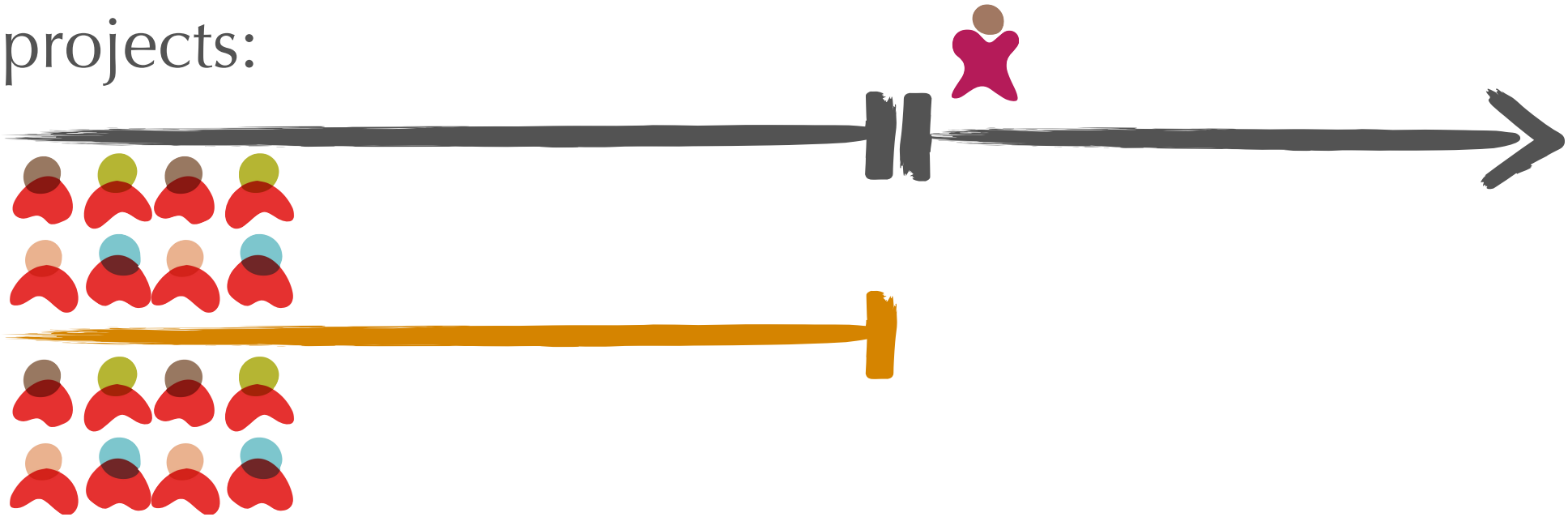




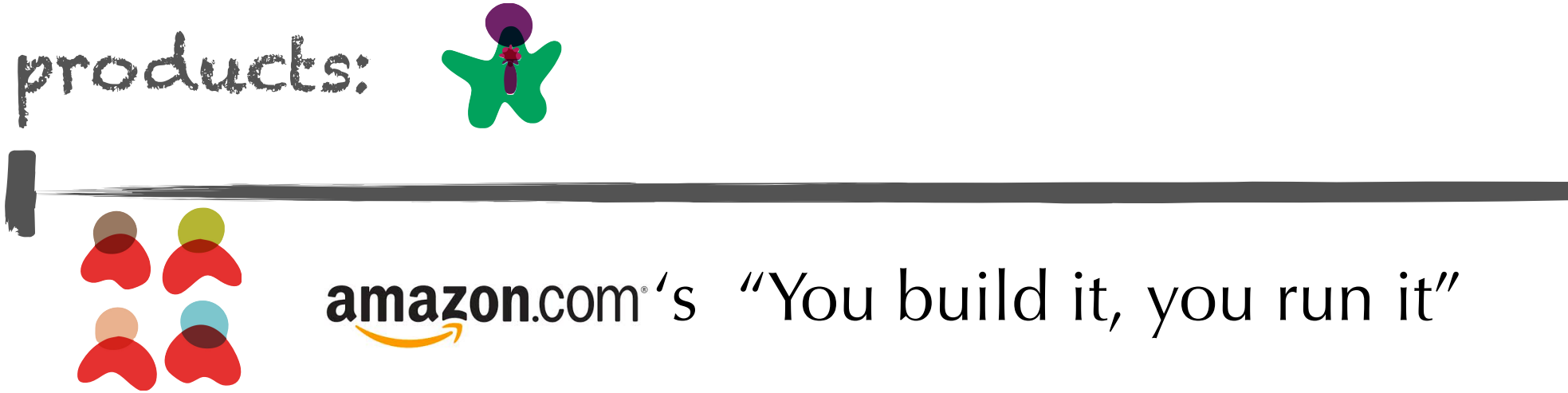
...physically realized.

Products, not Projects

projects:

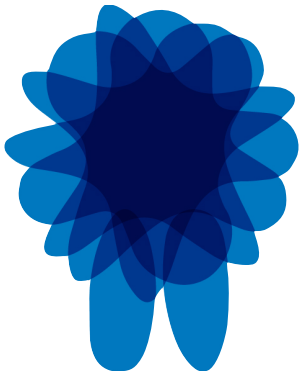


products:

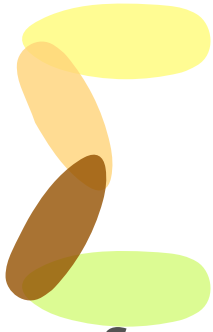


amazon.com's "You build it, you run it"

principles



architecture



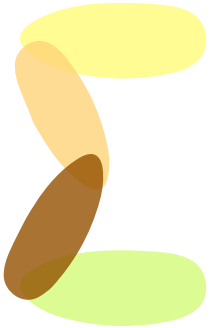
fitness functions

people

process

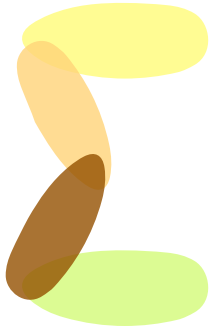
anti-principles

Fitness Function



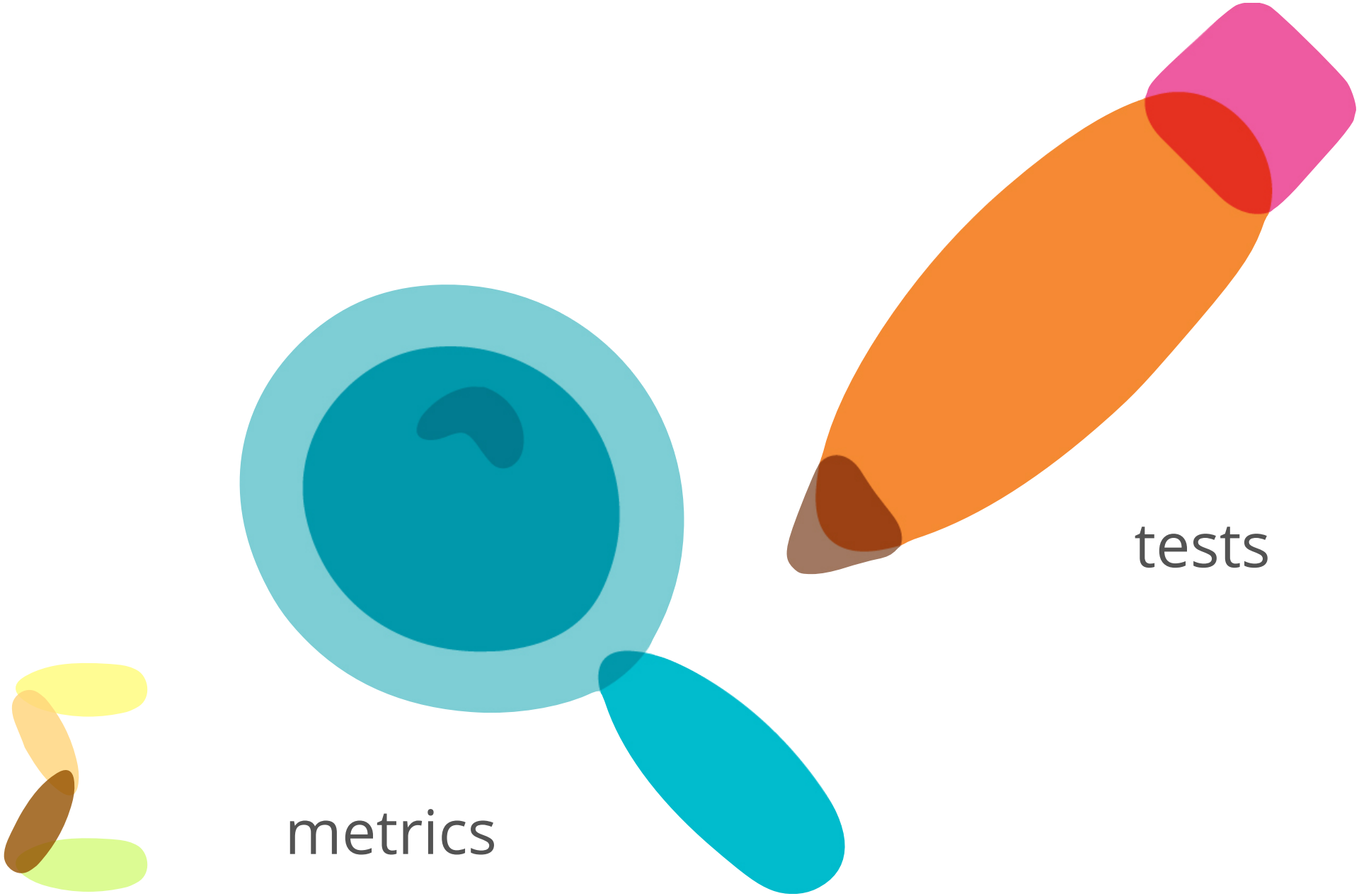
Architecture Fitness Function

metrics

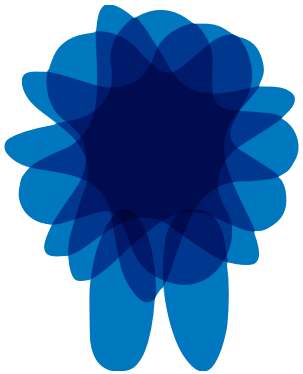


tests

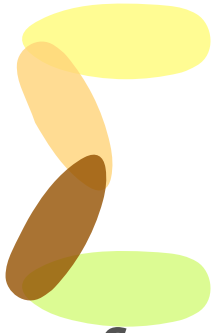
Architecture Fitness Function



principles



architecture



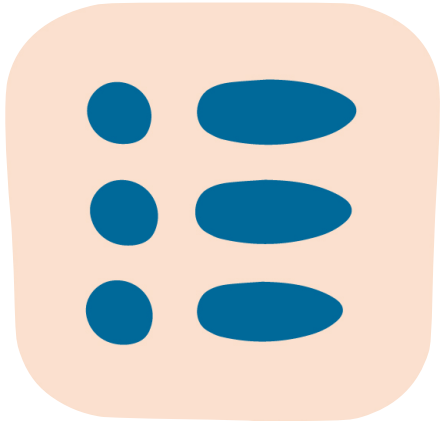
fitness functions

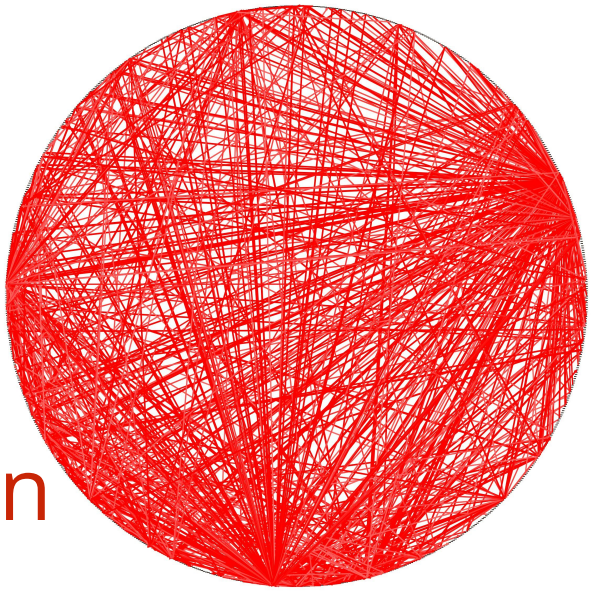
people

process

anti-principles

Anti-principles

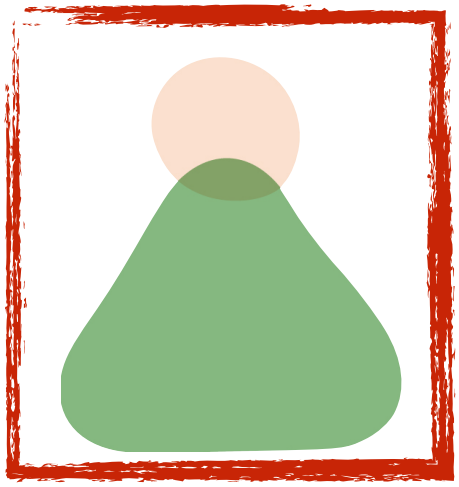




coding via configuration

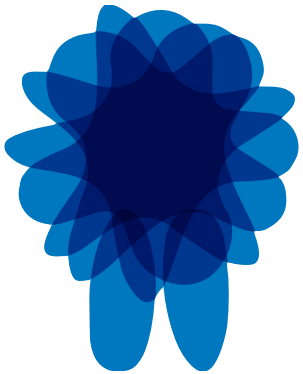
transactions

exuberant coupling

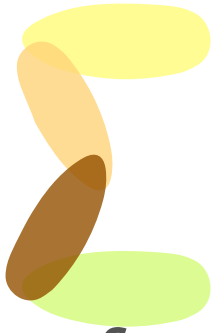


canonical governance

principles



architecture



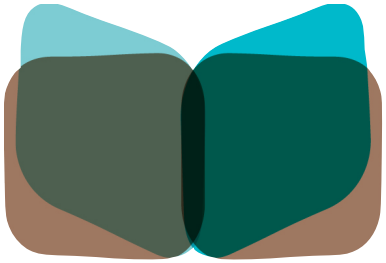
fitness functions

people

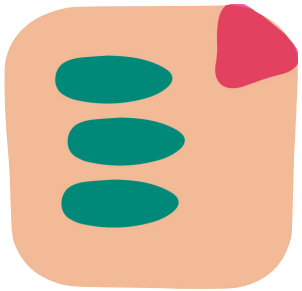
process

anti-principles

Agenda

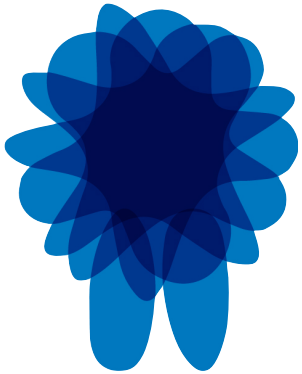


definition

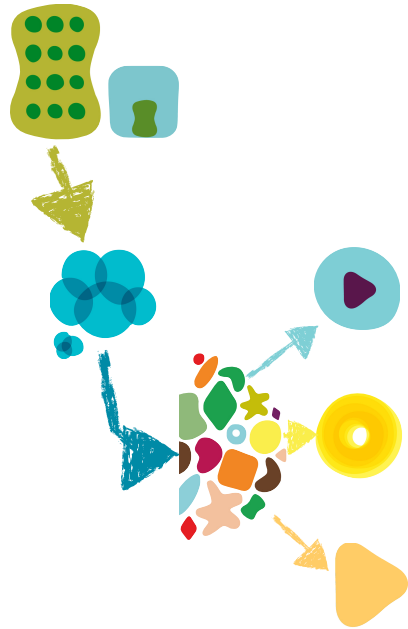


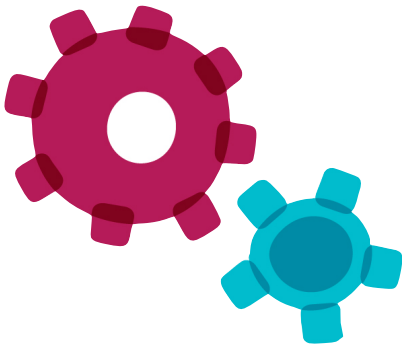
characteristics

principles



conclusion

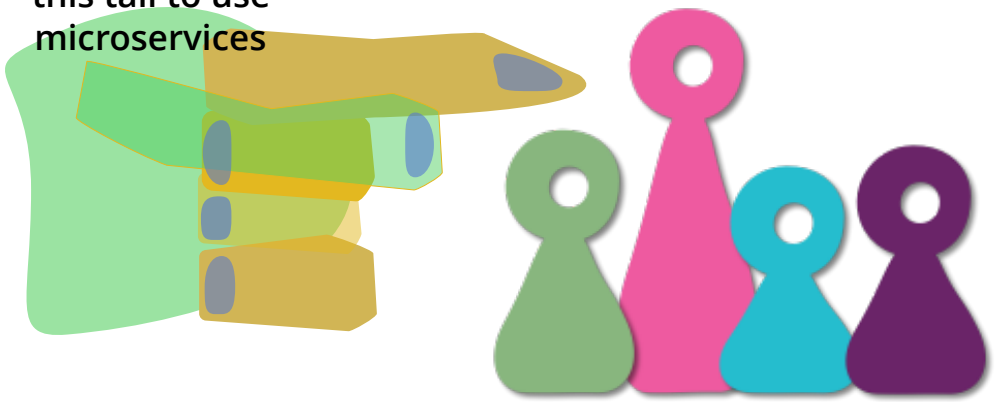


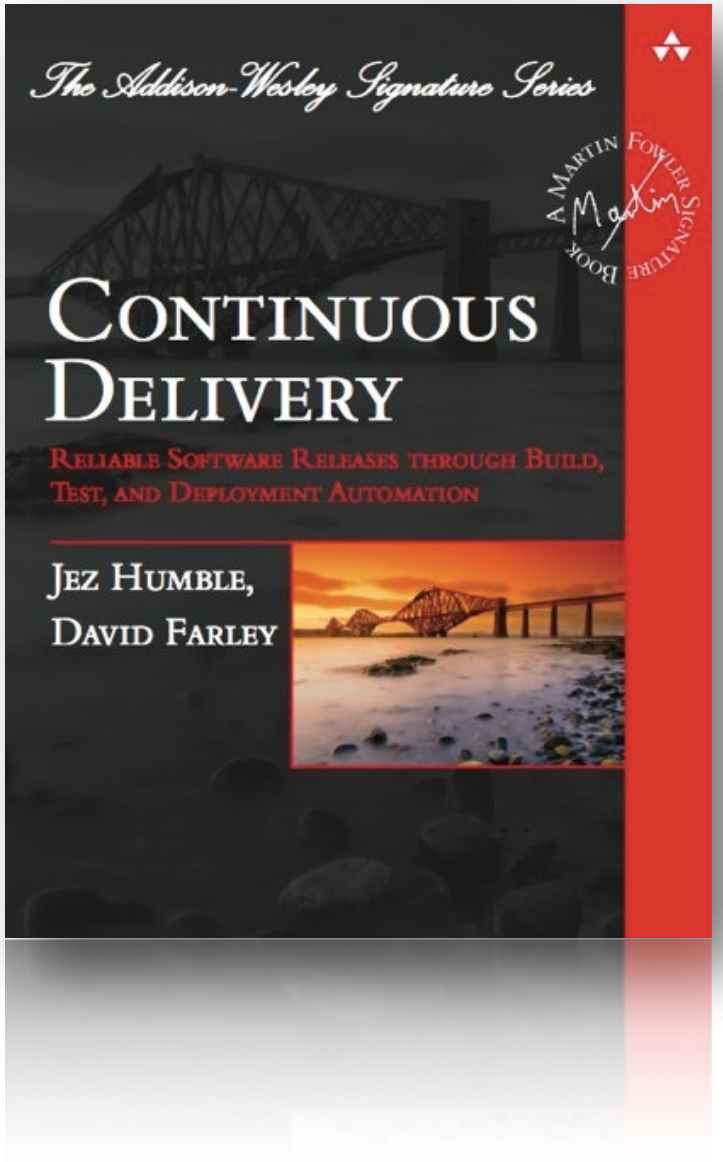


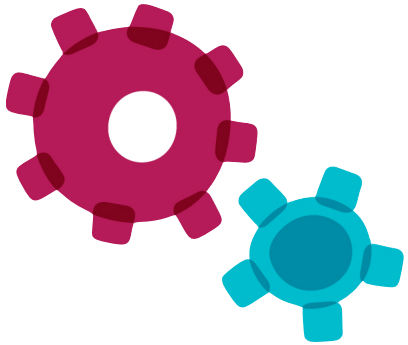
practices

Continuous Delivery

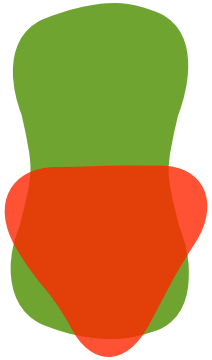
You must be this tall to use microservices



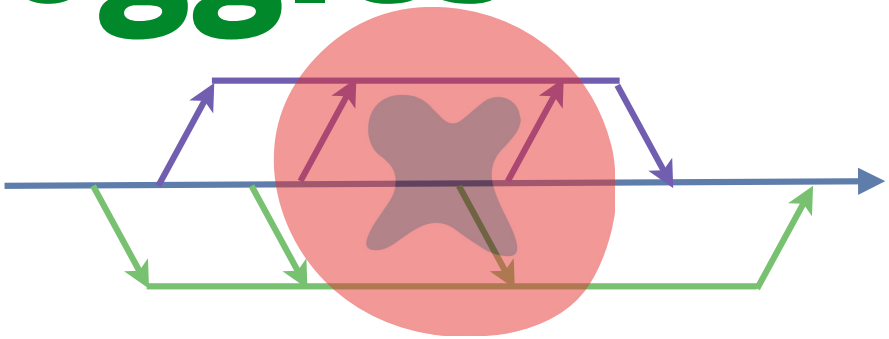


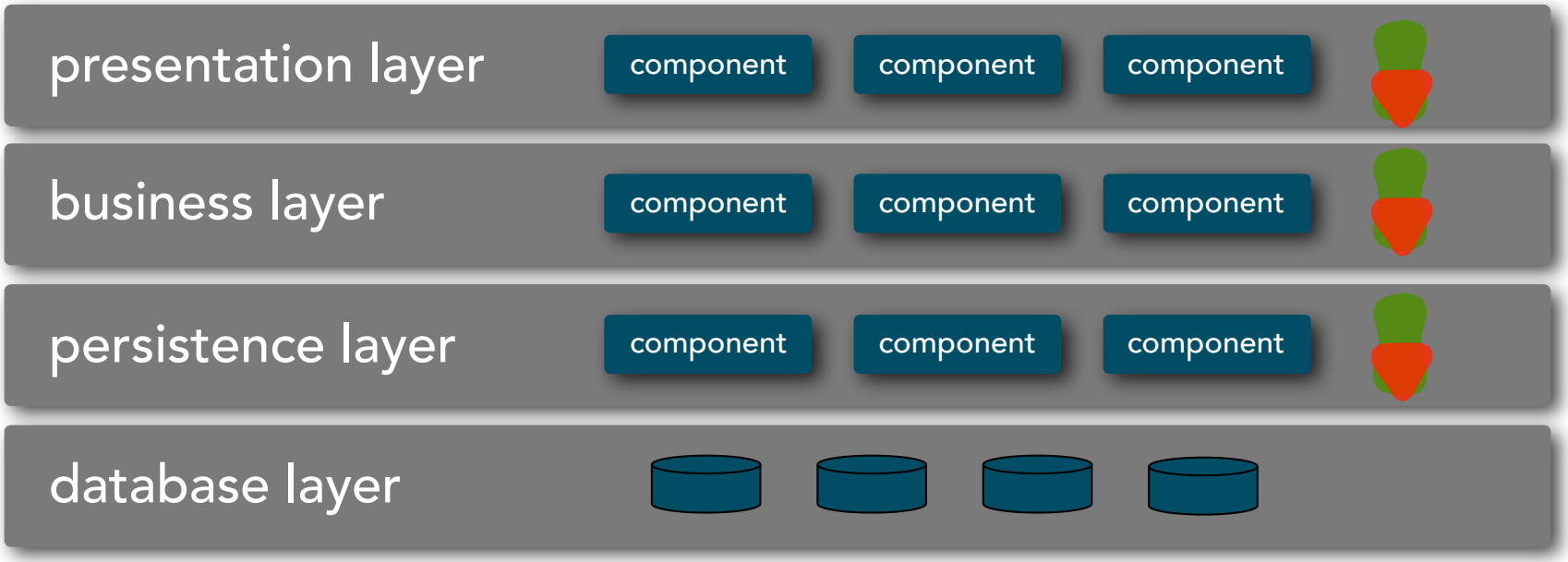


practices

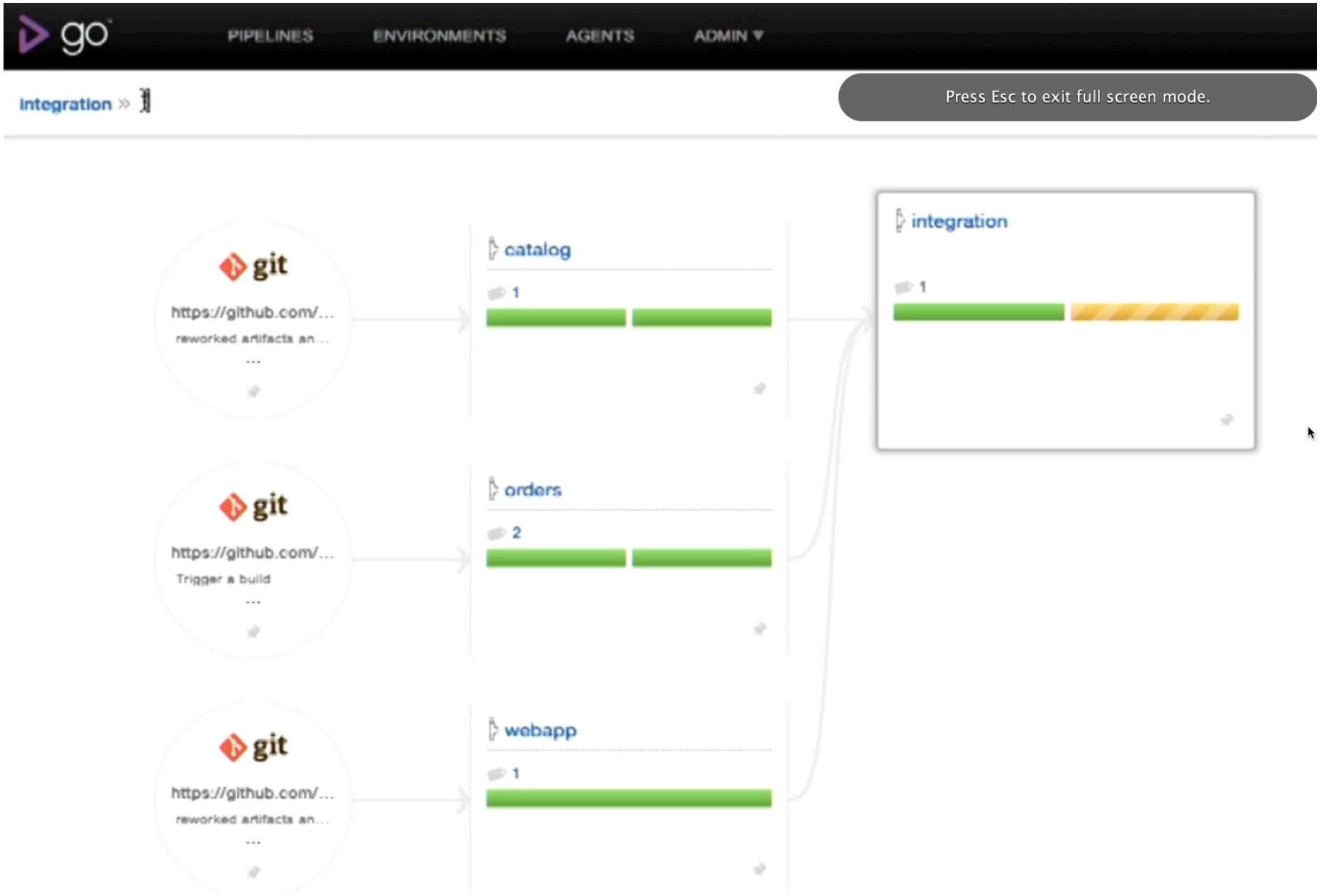


Feature Toggles

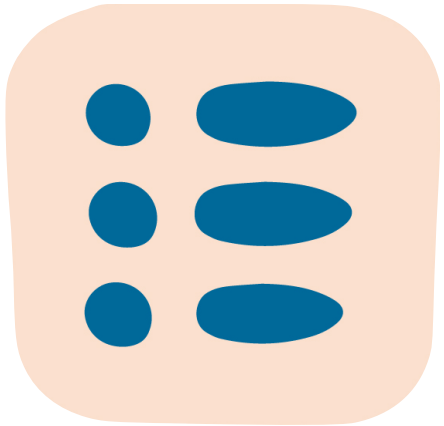




Deployment Pipelines

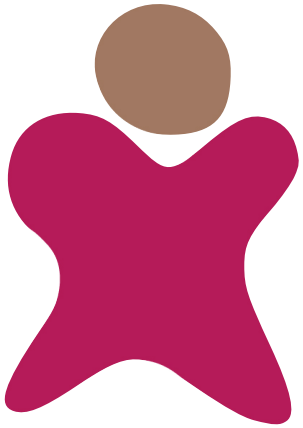


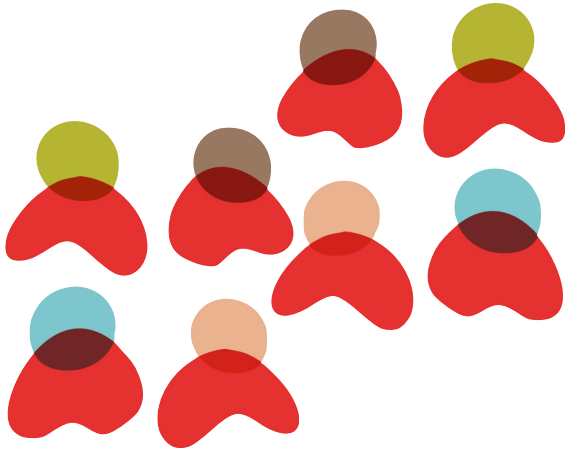
Team Practices



articulated principles

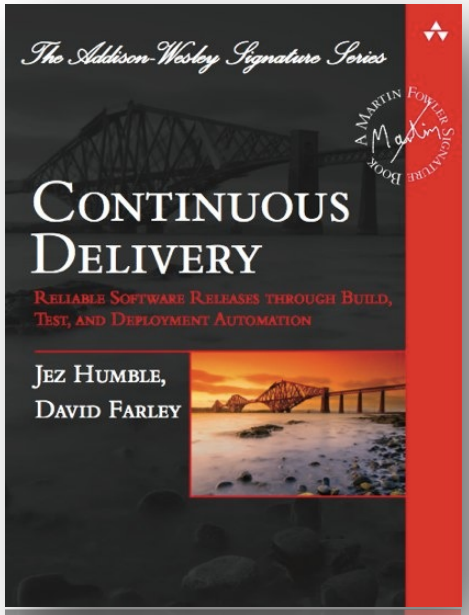
cross-functional teams

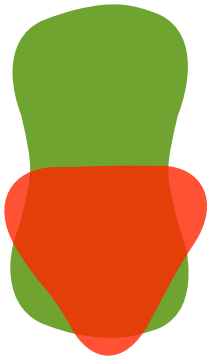




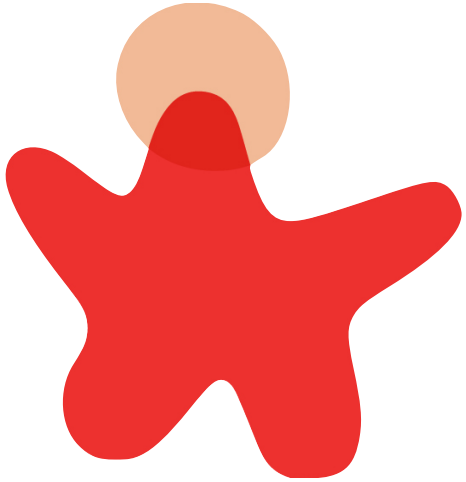
architectural briefings

Experimentation Enablers





feature toggles



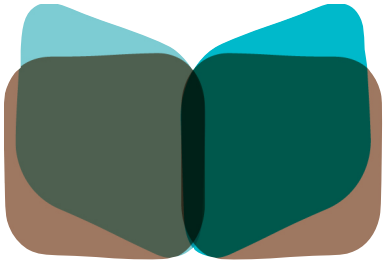
trusted user(s)

metrics

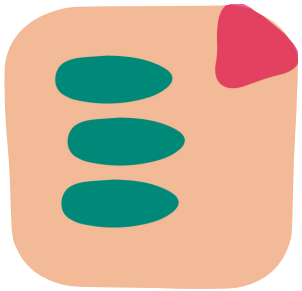
fitness functions

spikes

Agenda

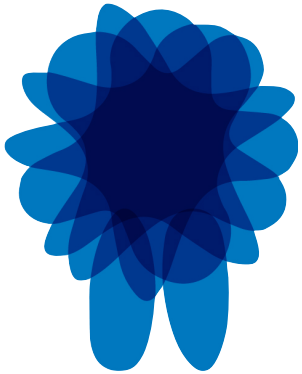


definition

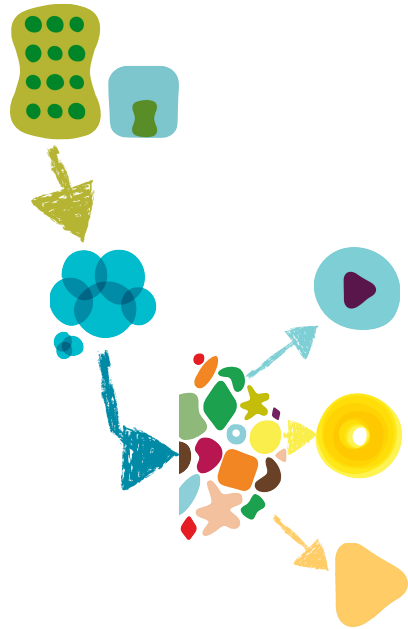


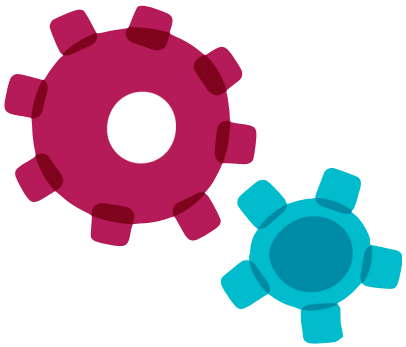
characteristics

principles

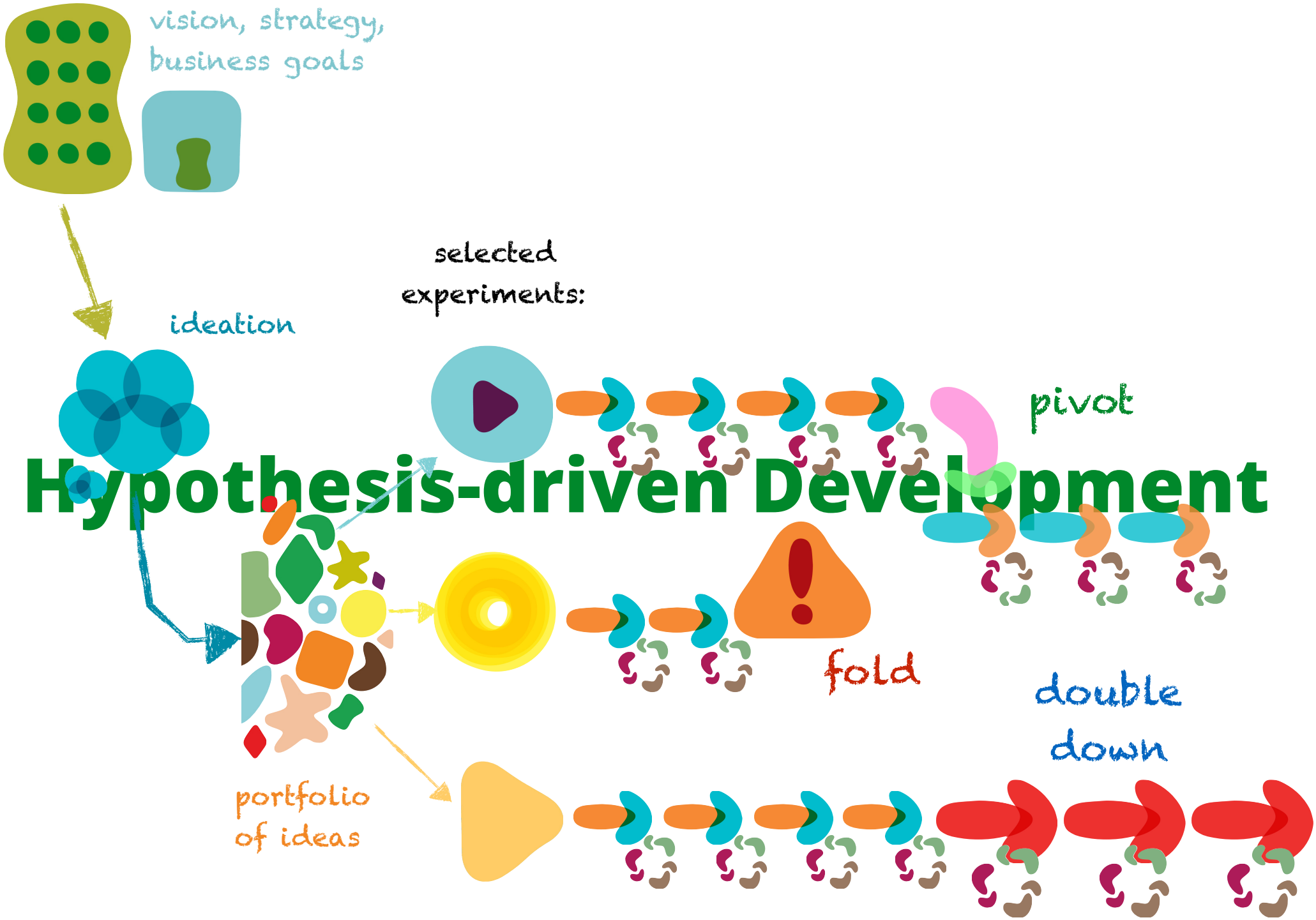


conclusion





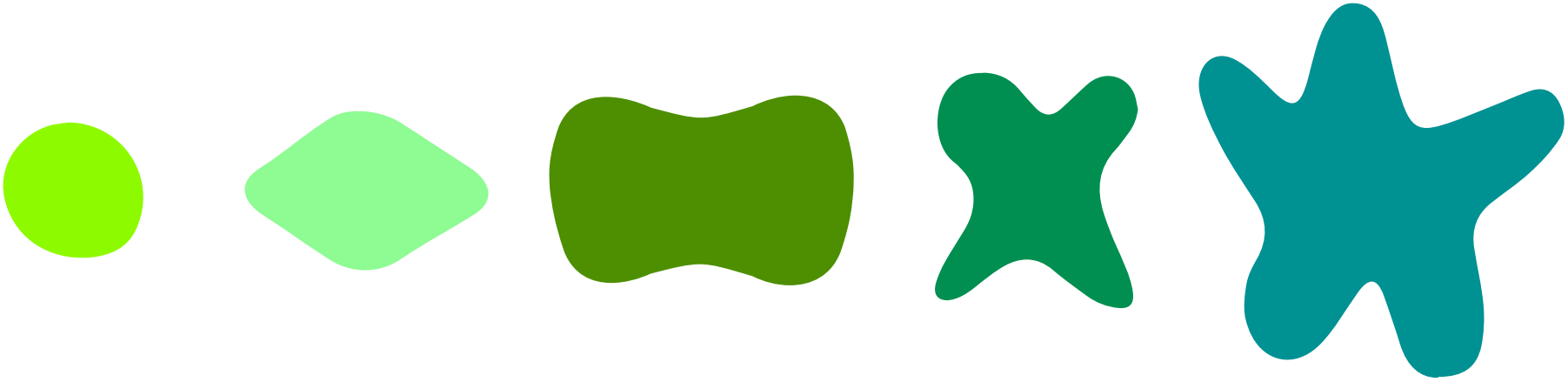
practices

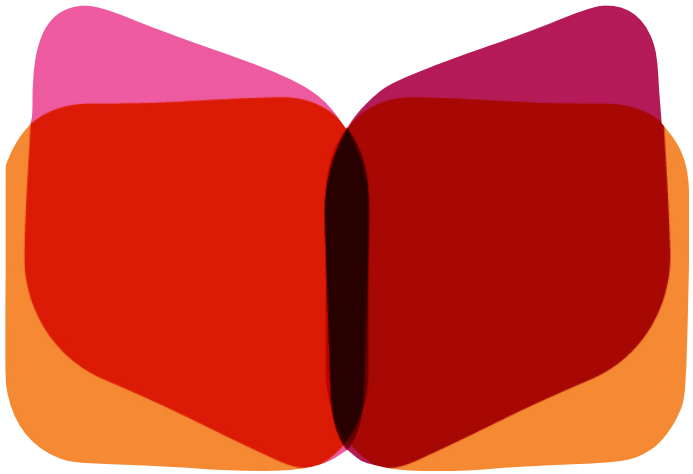

vision, strategy,
business goals
ideation
selected
experiments:
pivot
Hypothesis-driven Development
fold
double
down
portfolio
of ideas

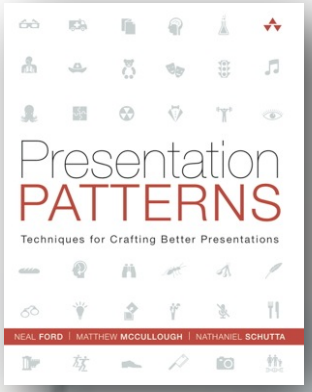
Dynamic Equilibrium

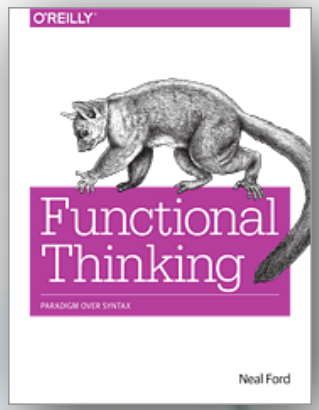
<insert new technology here>

Stay Tuned!









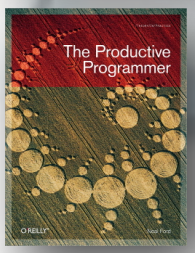
nealford.com

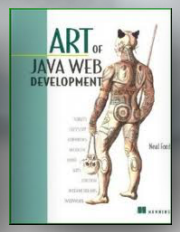
@neal4d

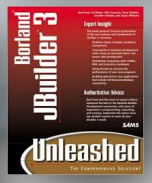
ThoughtWorks®

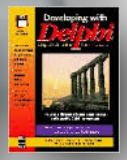
NEAL FORD

Director / Software Architect / Meme Wrangler



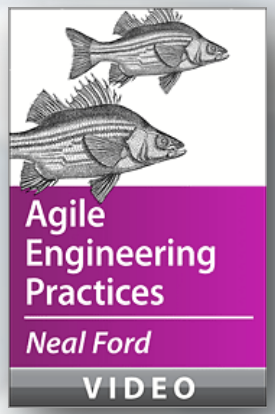


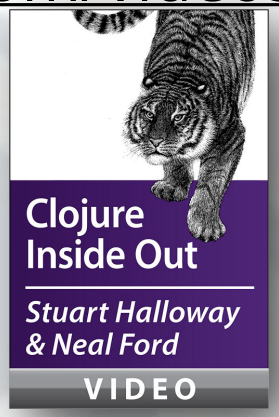


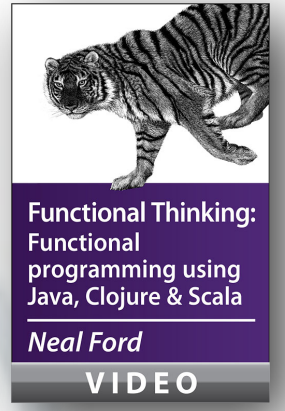


nealford.com/books

nealford.com/videos



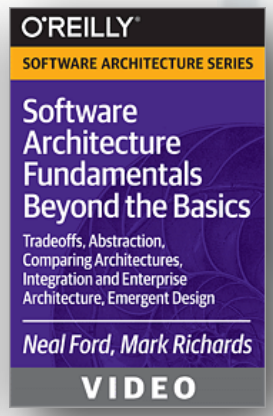


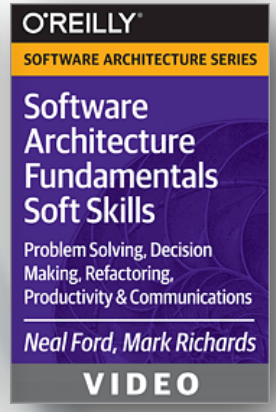


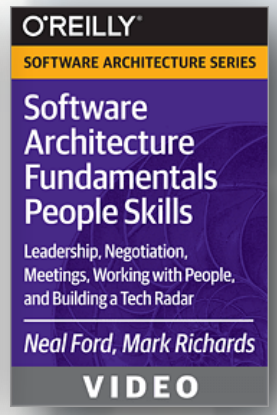
O'REILLY®

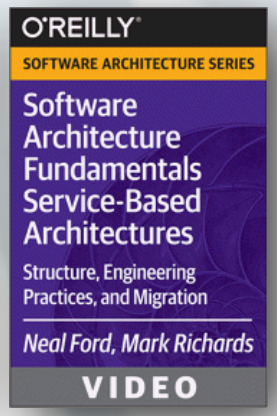
SOFTWARE ARCHITECTURE SERIES

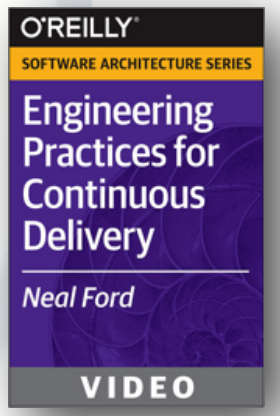
www.oreilly.com/software-architecture-video-training-series.html



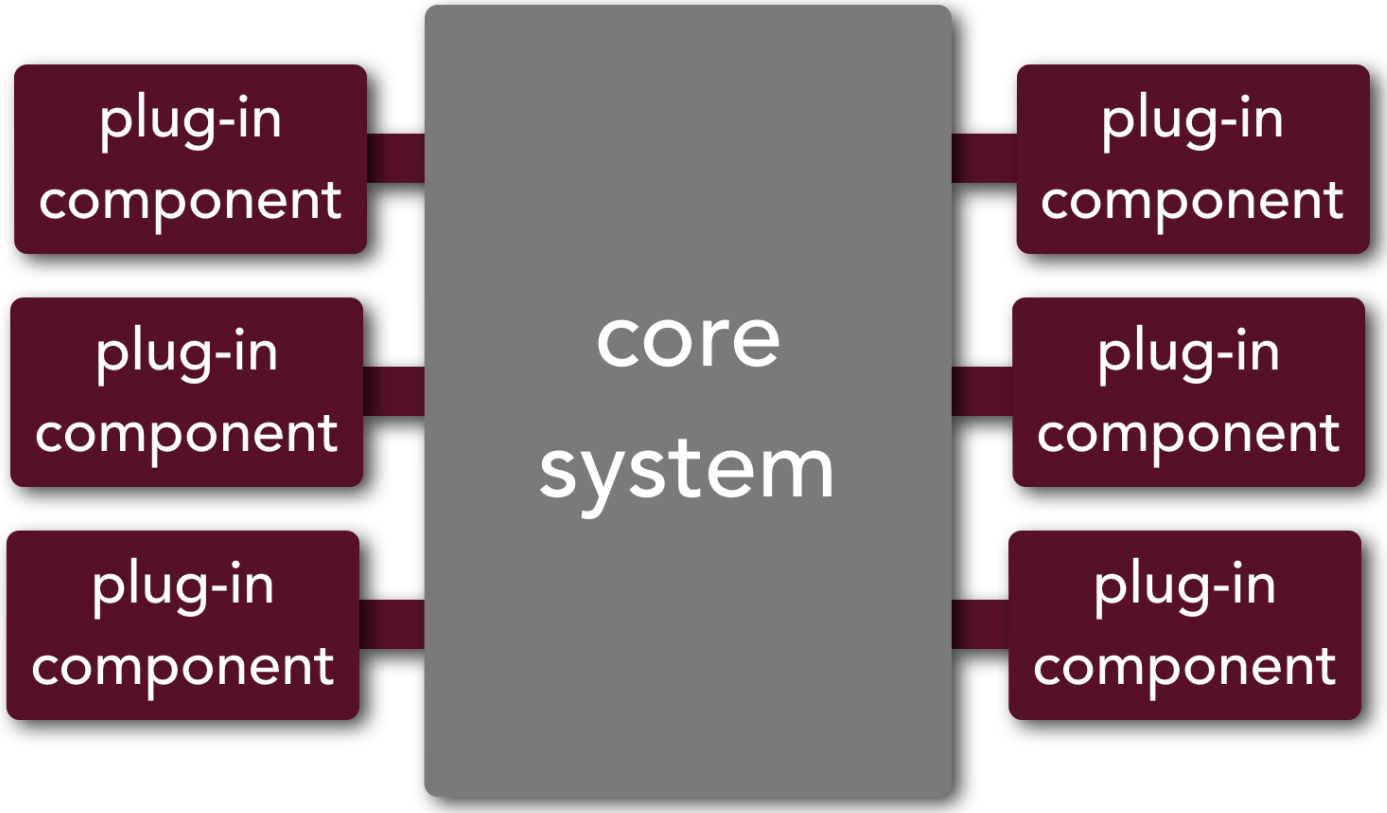




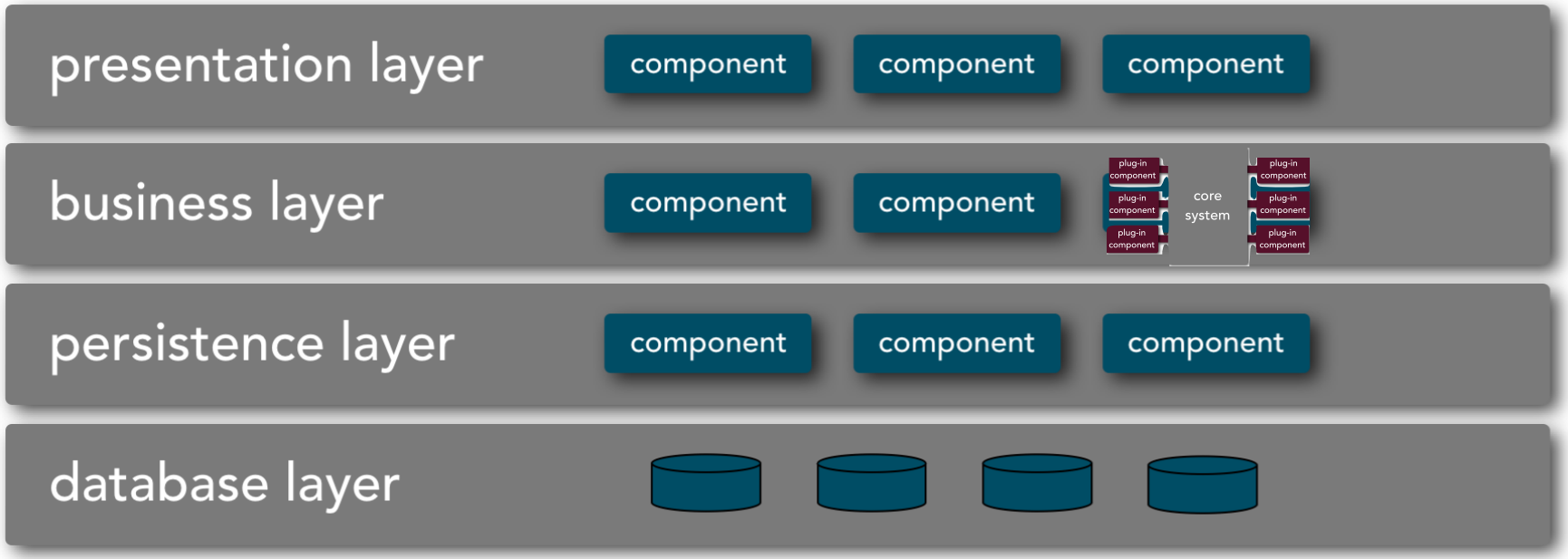




Composability



Composability



Composability