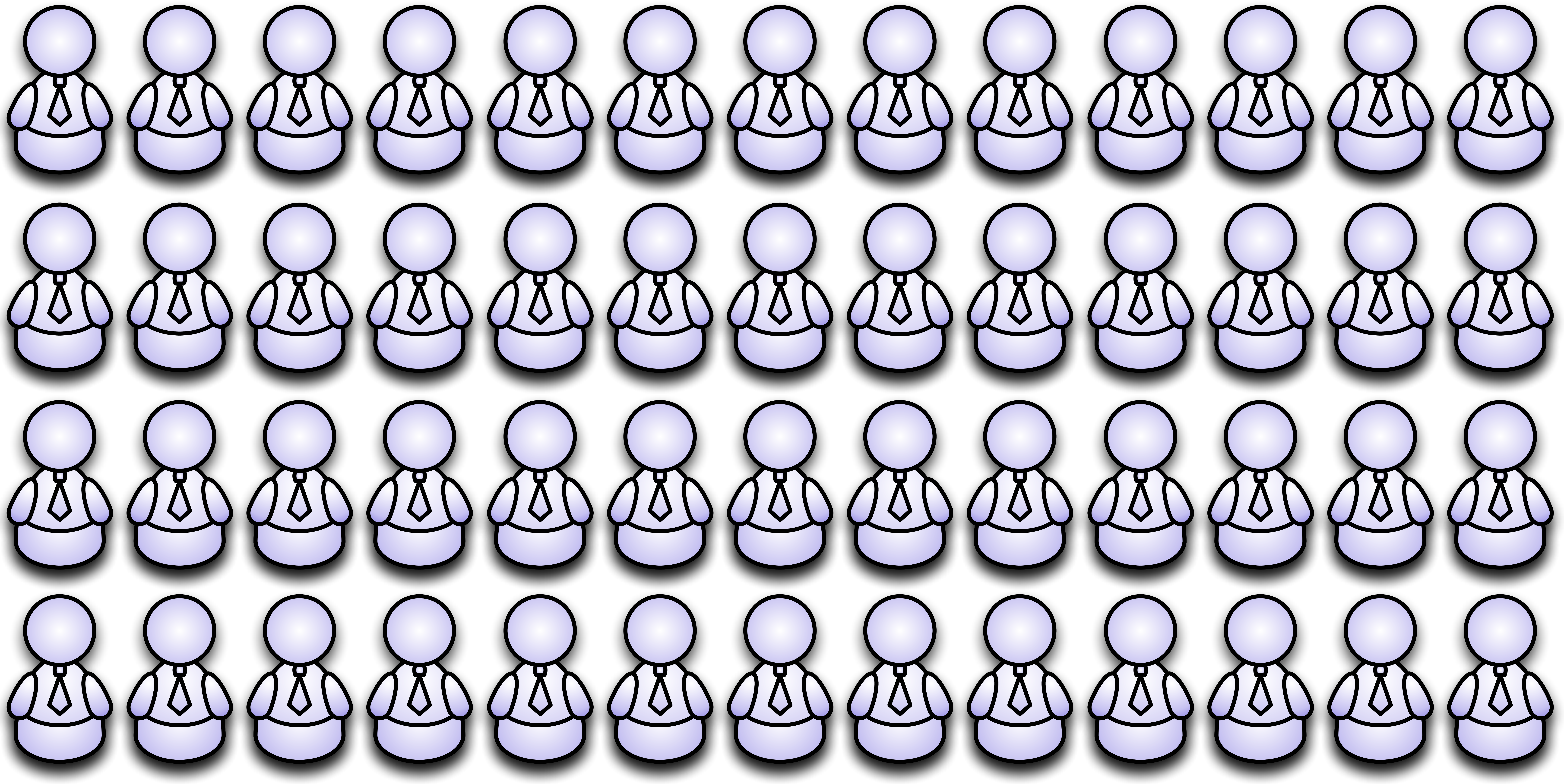
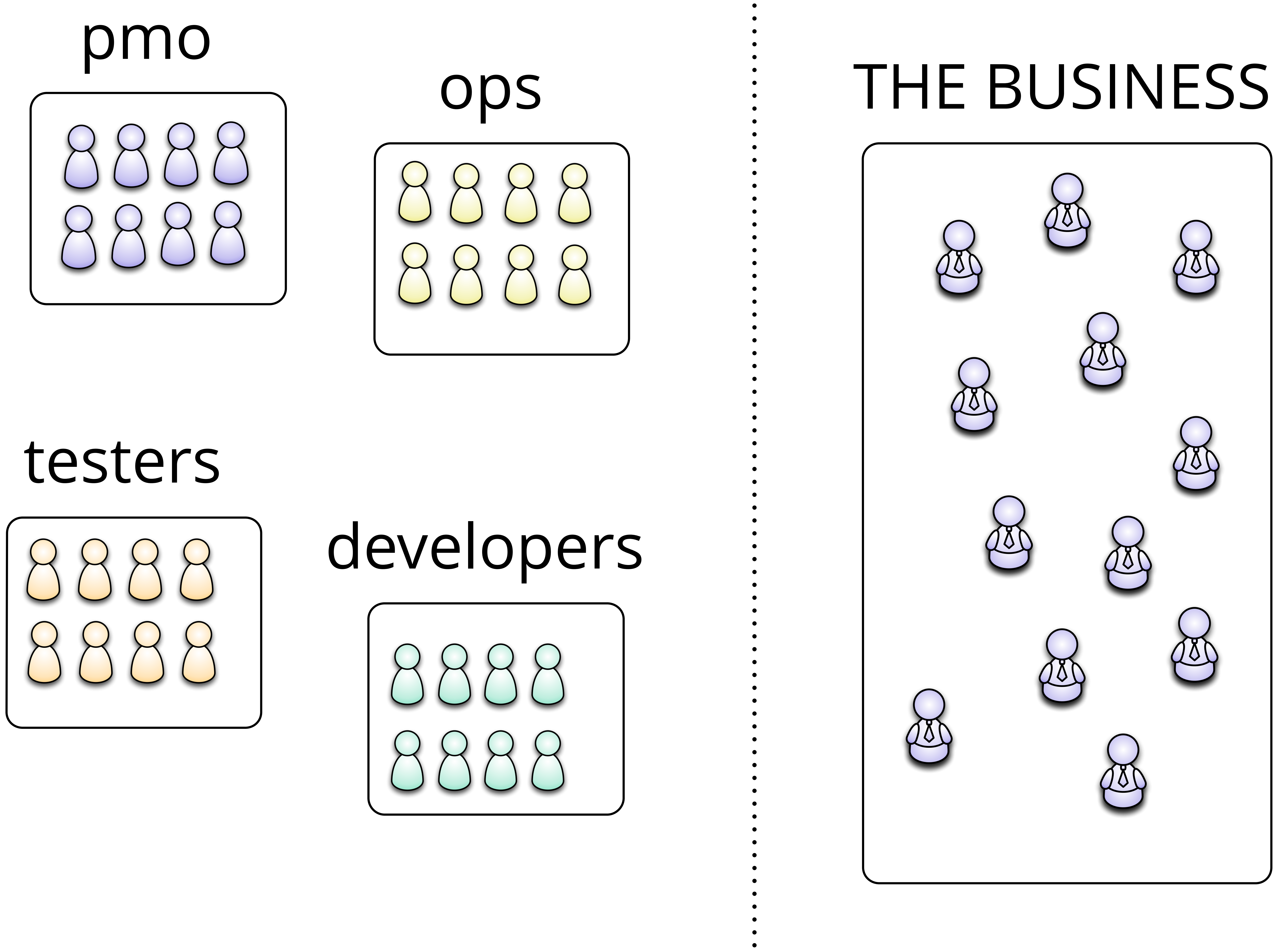
ThoughtWorks®

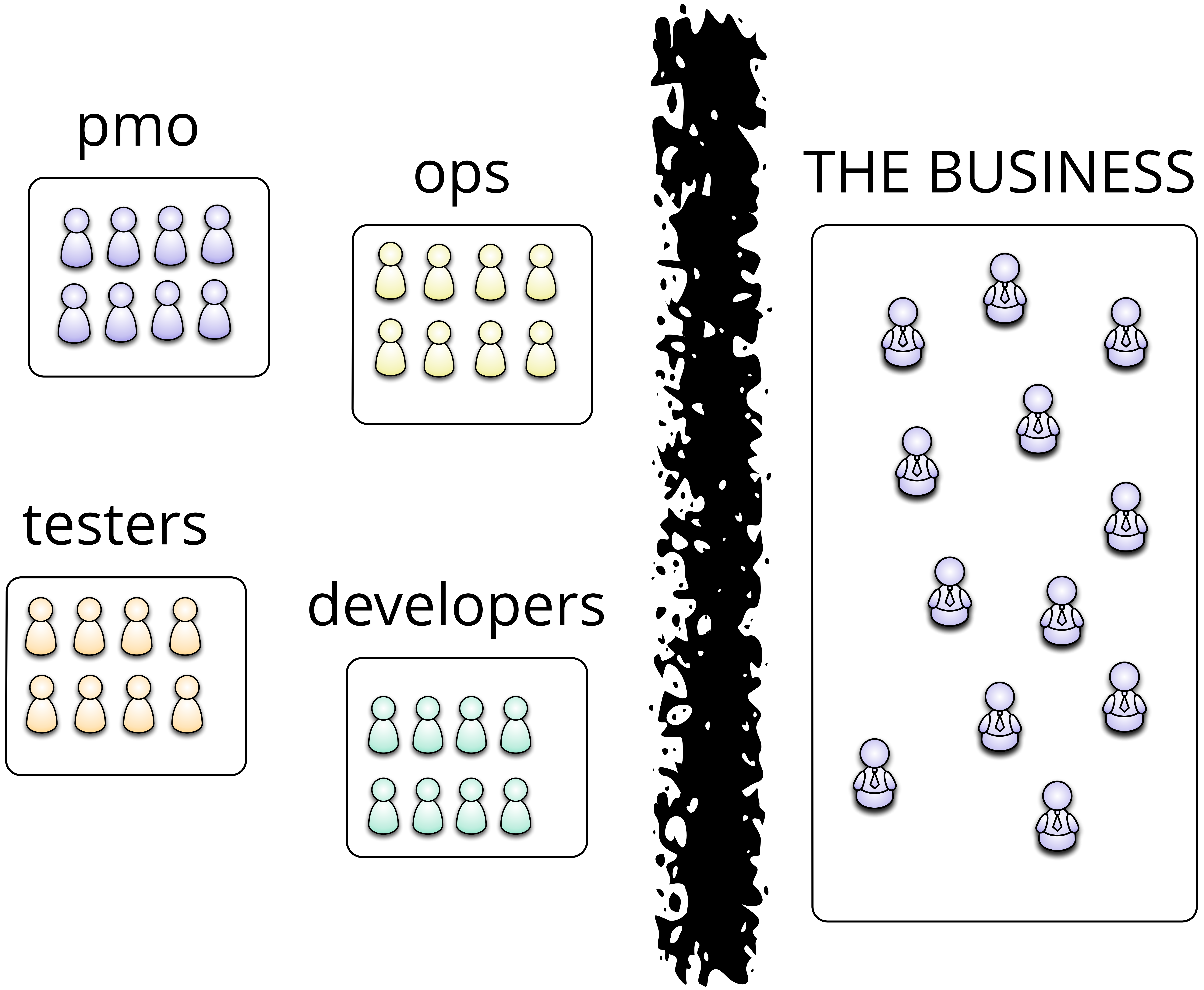
LESSONS FROM A POLYGLOT PORTFOLIO

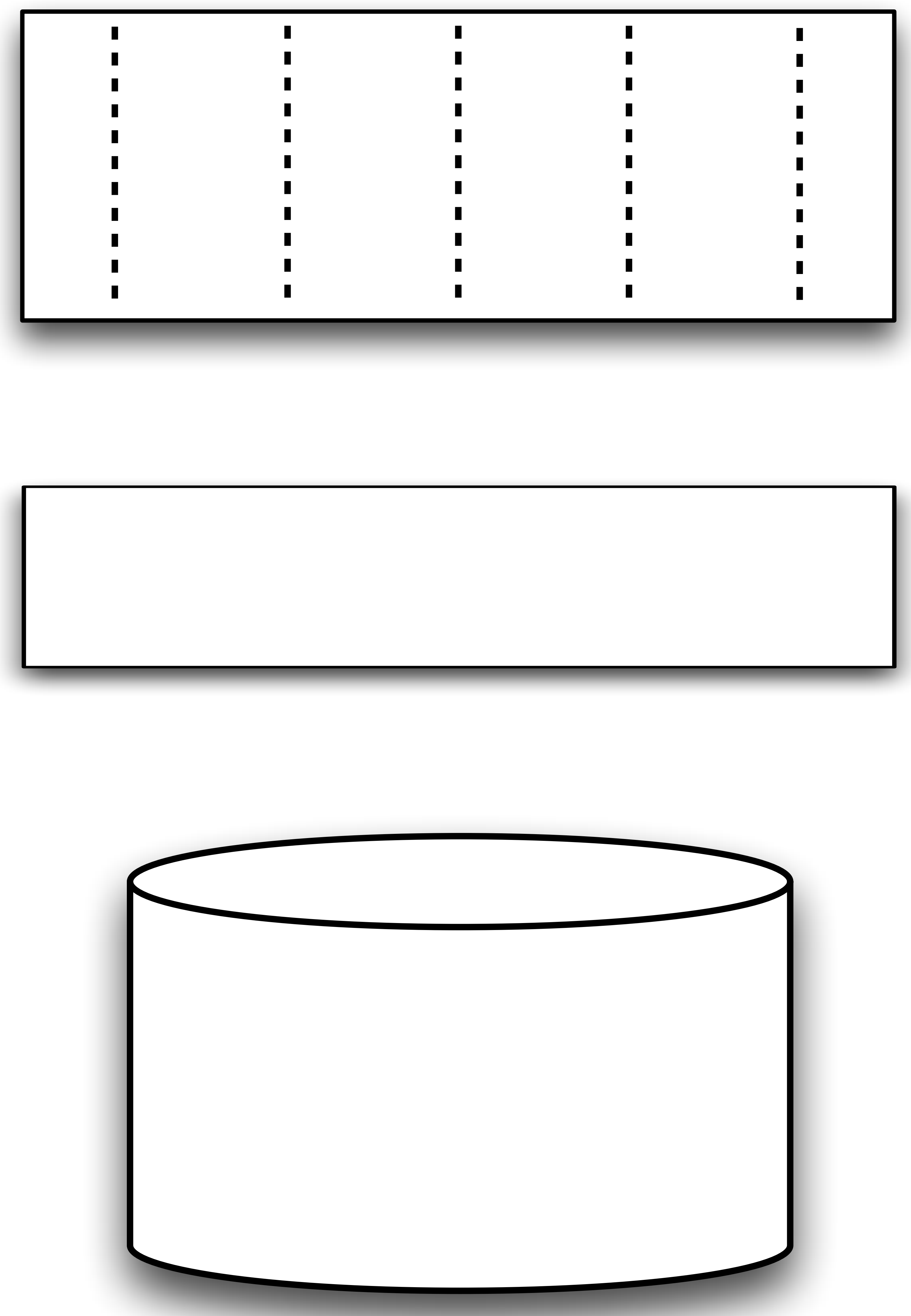
james lewis

jalewis@thoughtworks.com
@boicy



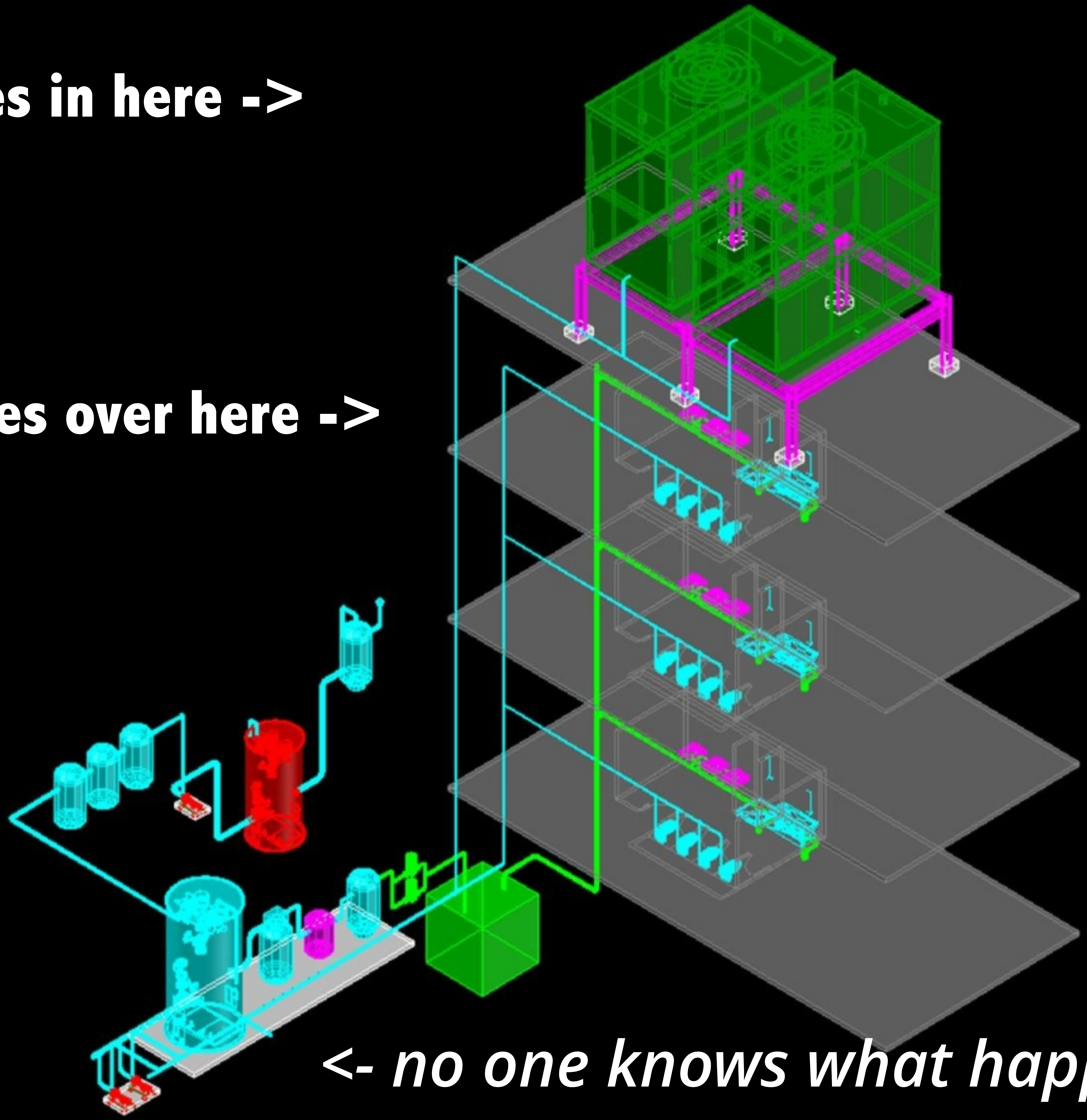

pmo
ops
testers
developers
THE BUSINESS


pmo
ops
THE BUSINESS
testers
developers

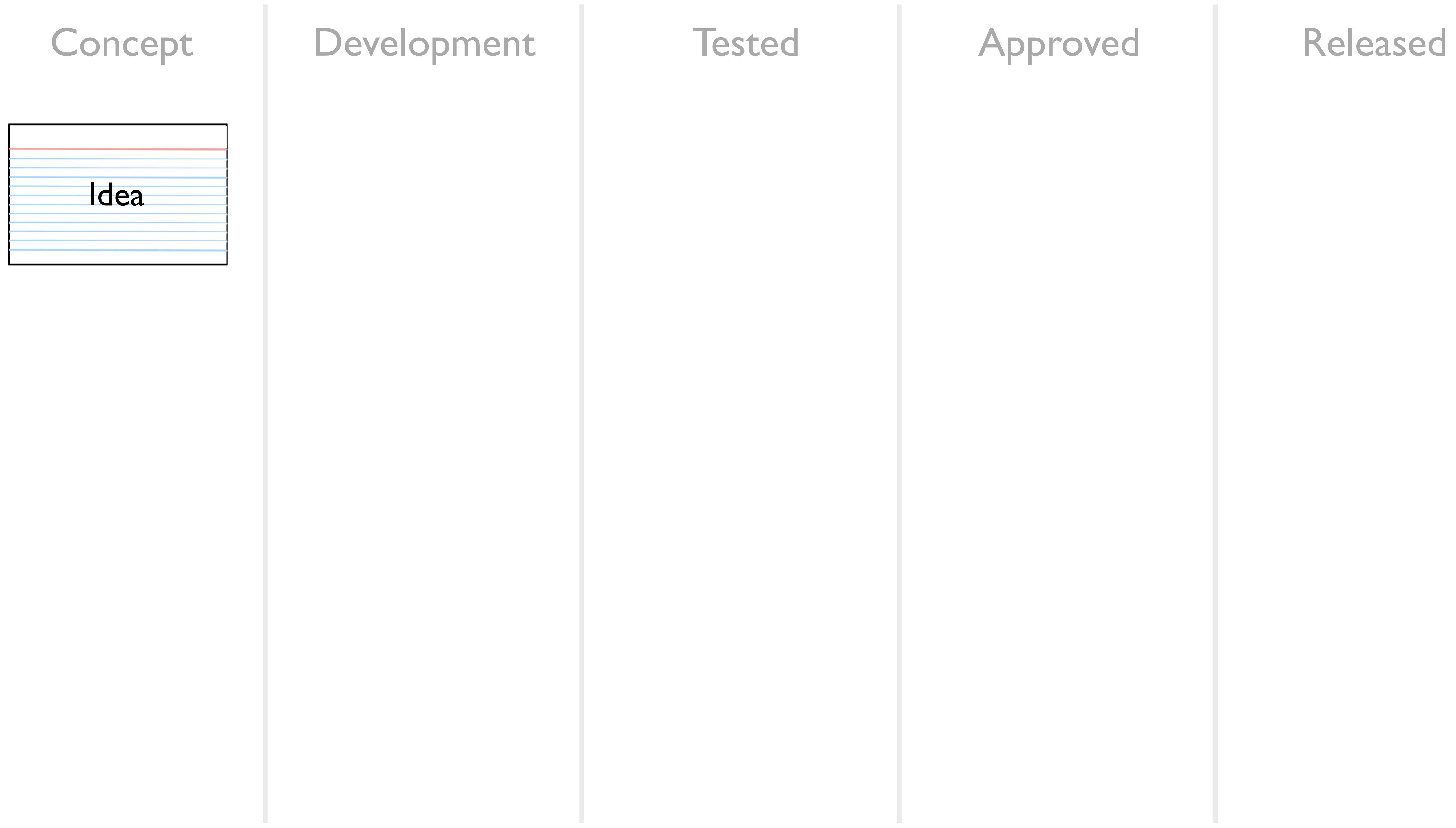


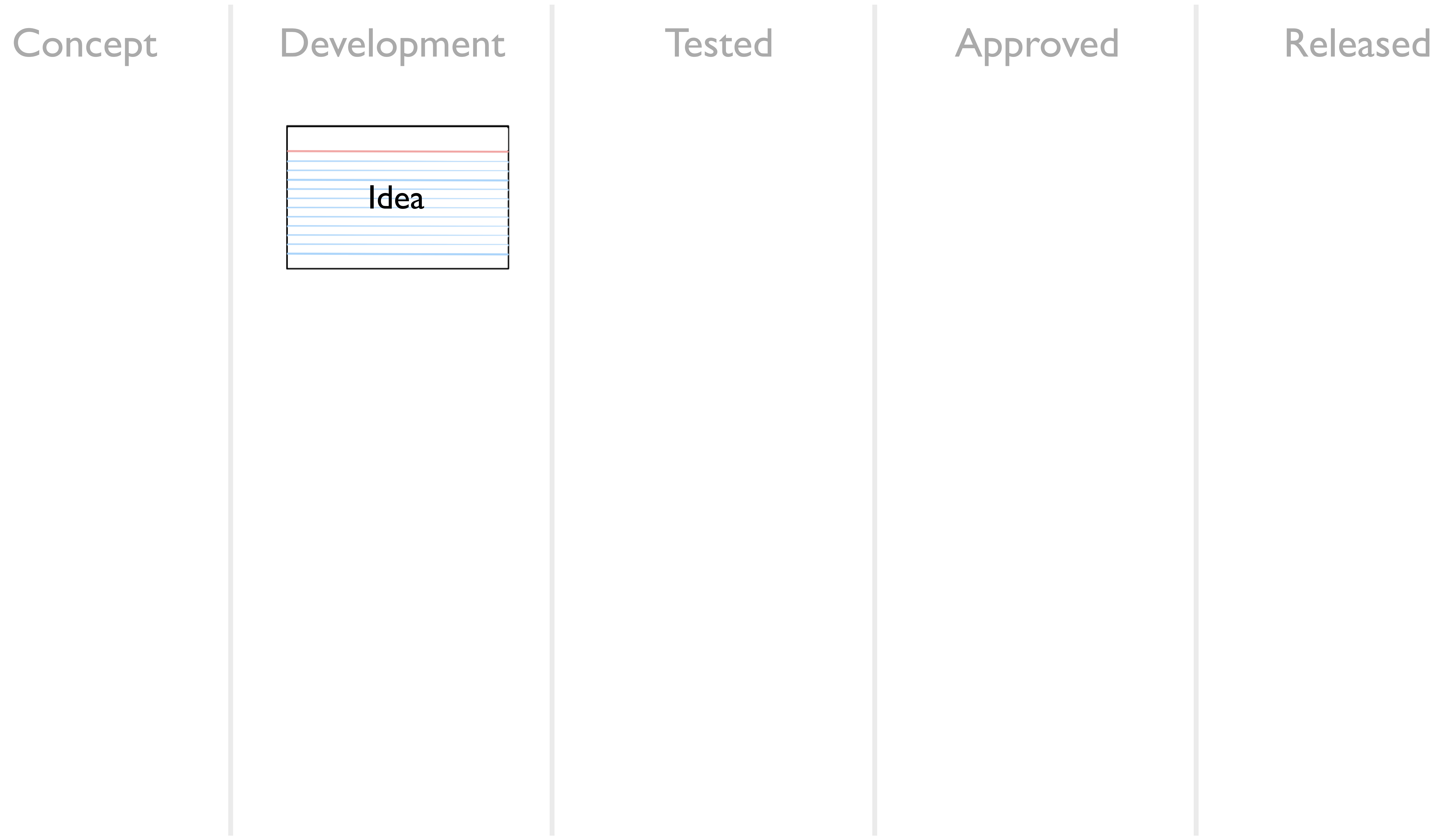
HOHNER

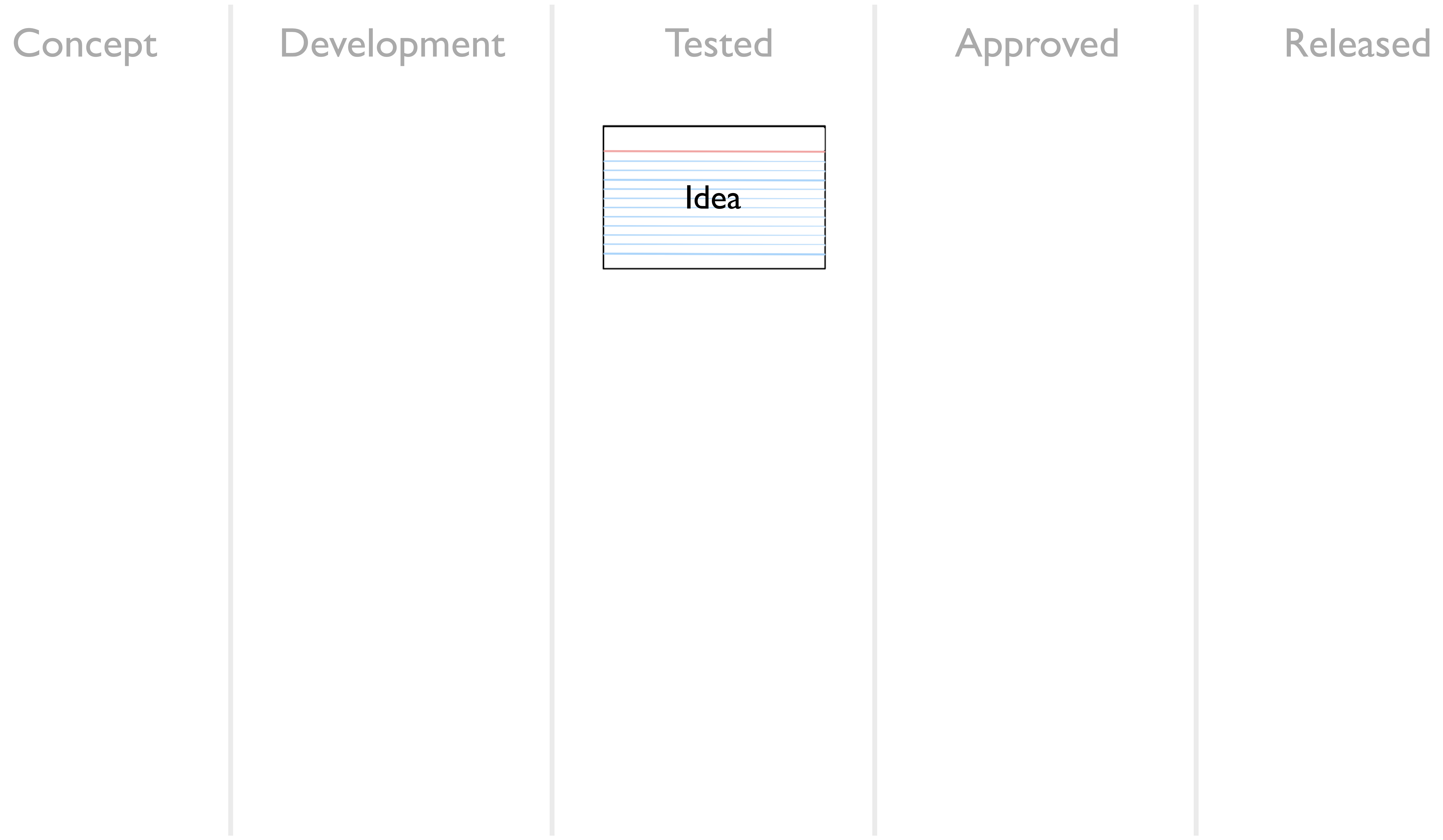
USA
Columbia
YOUNG · CRIPPEN

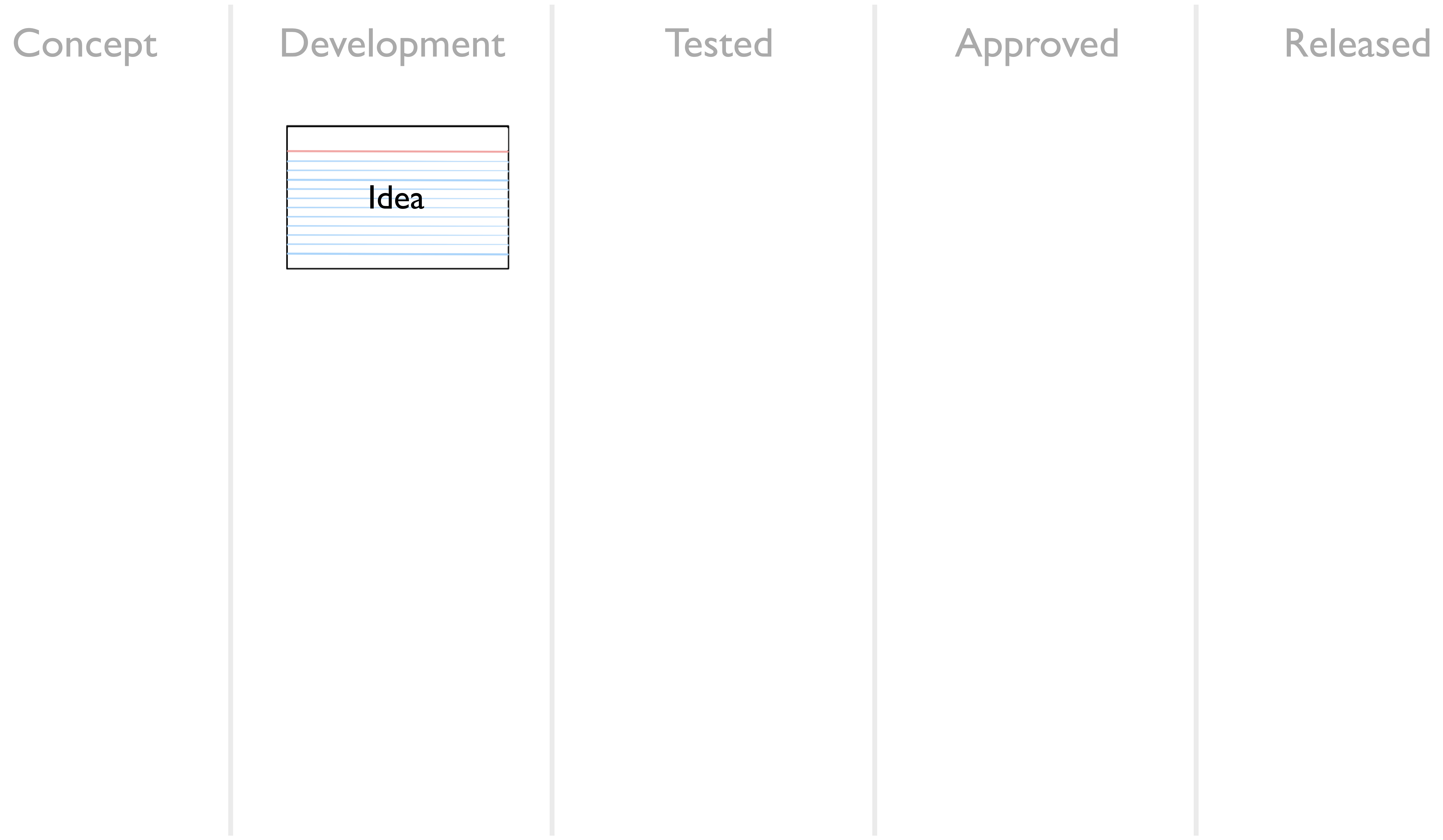

data goes in here ->
ETL takes over here ->
<- no one knows what happens here

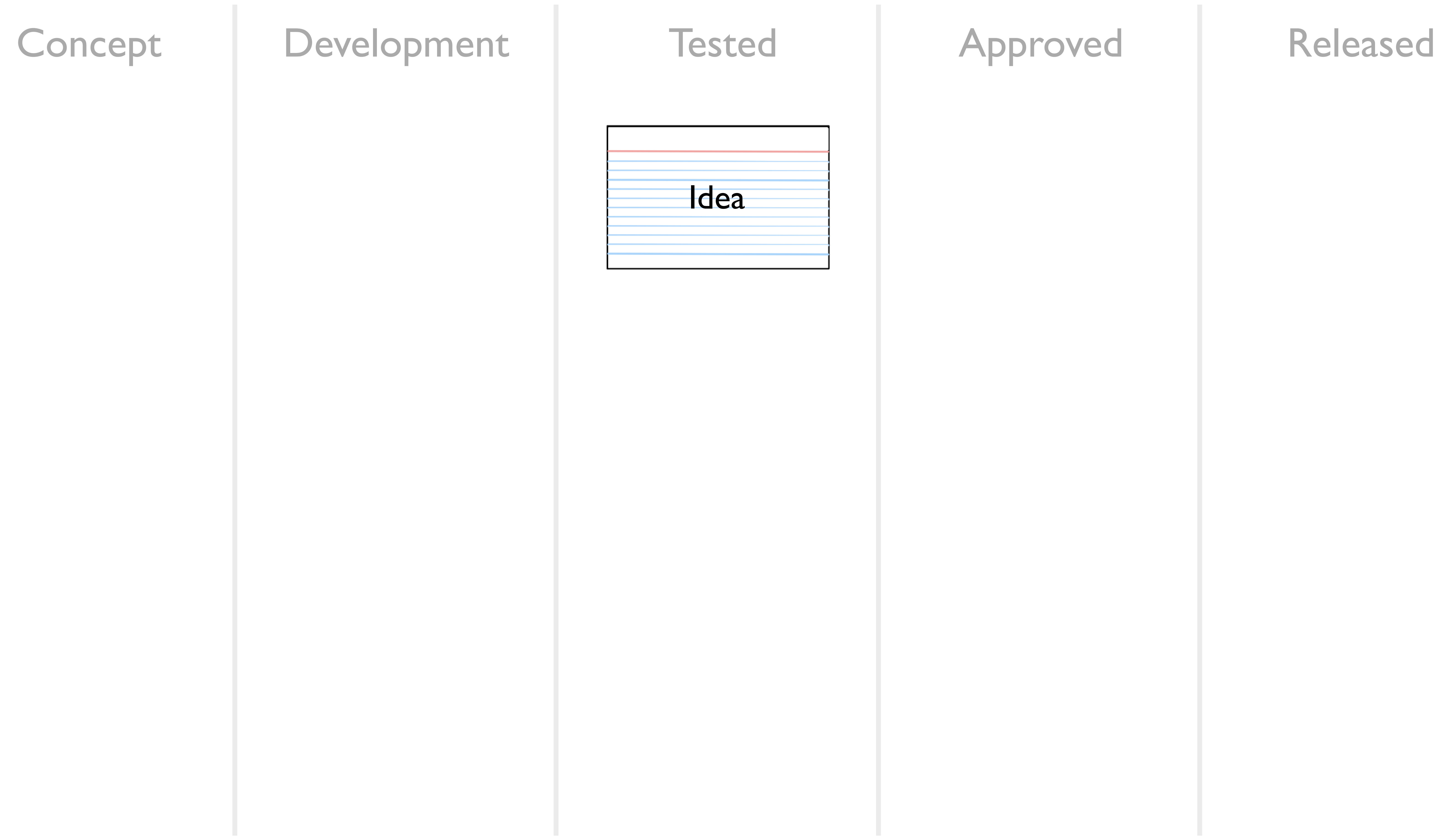
start

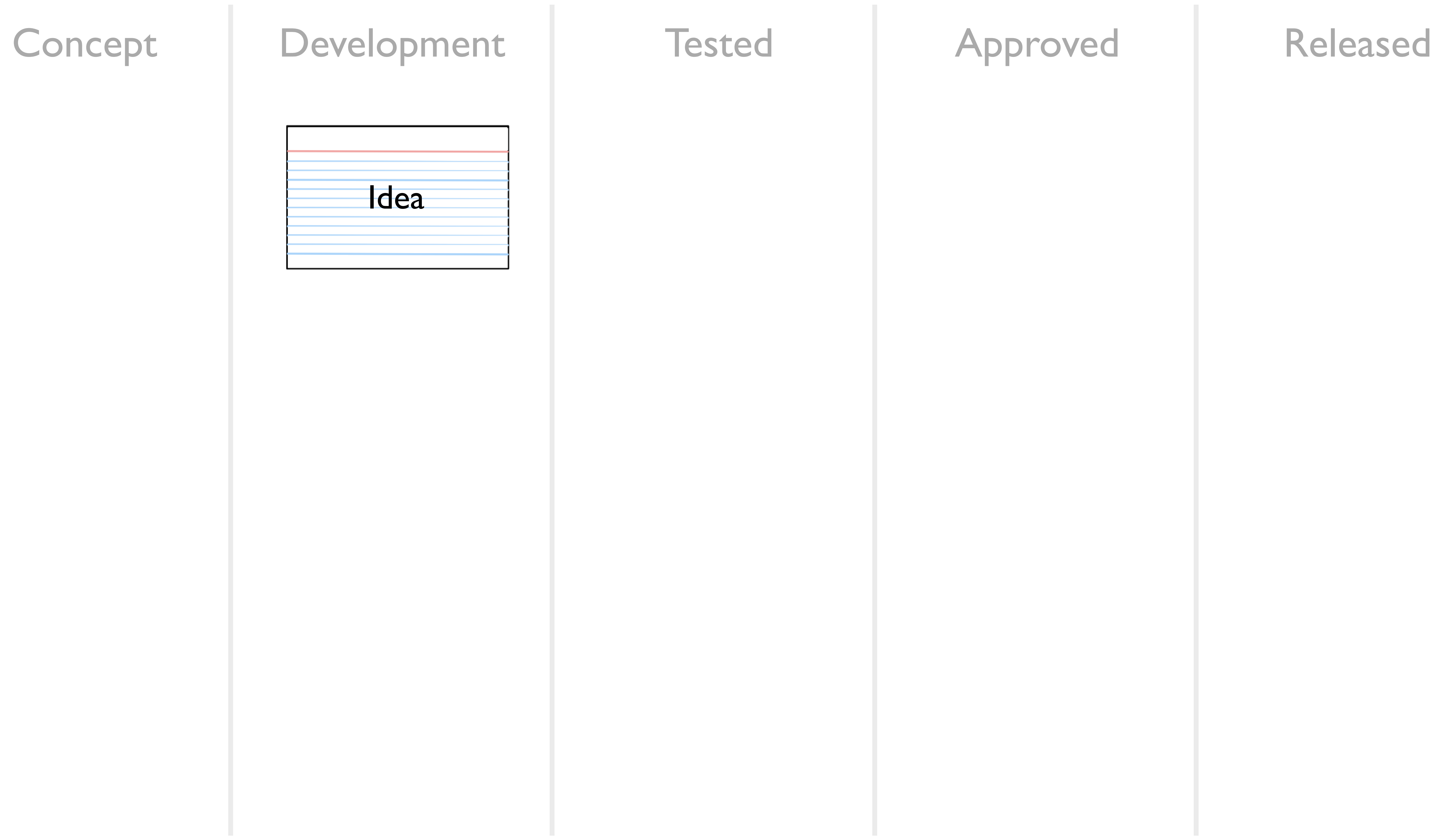

Concept
Development
Tested
Approved
Released
Idea


Concept
Development
Tested
Approved
Released
Idea

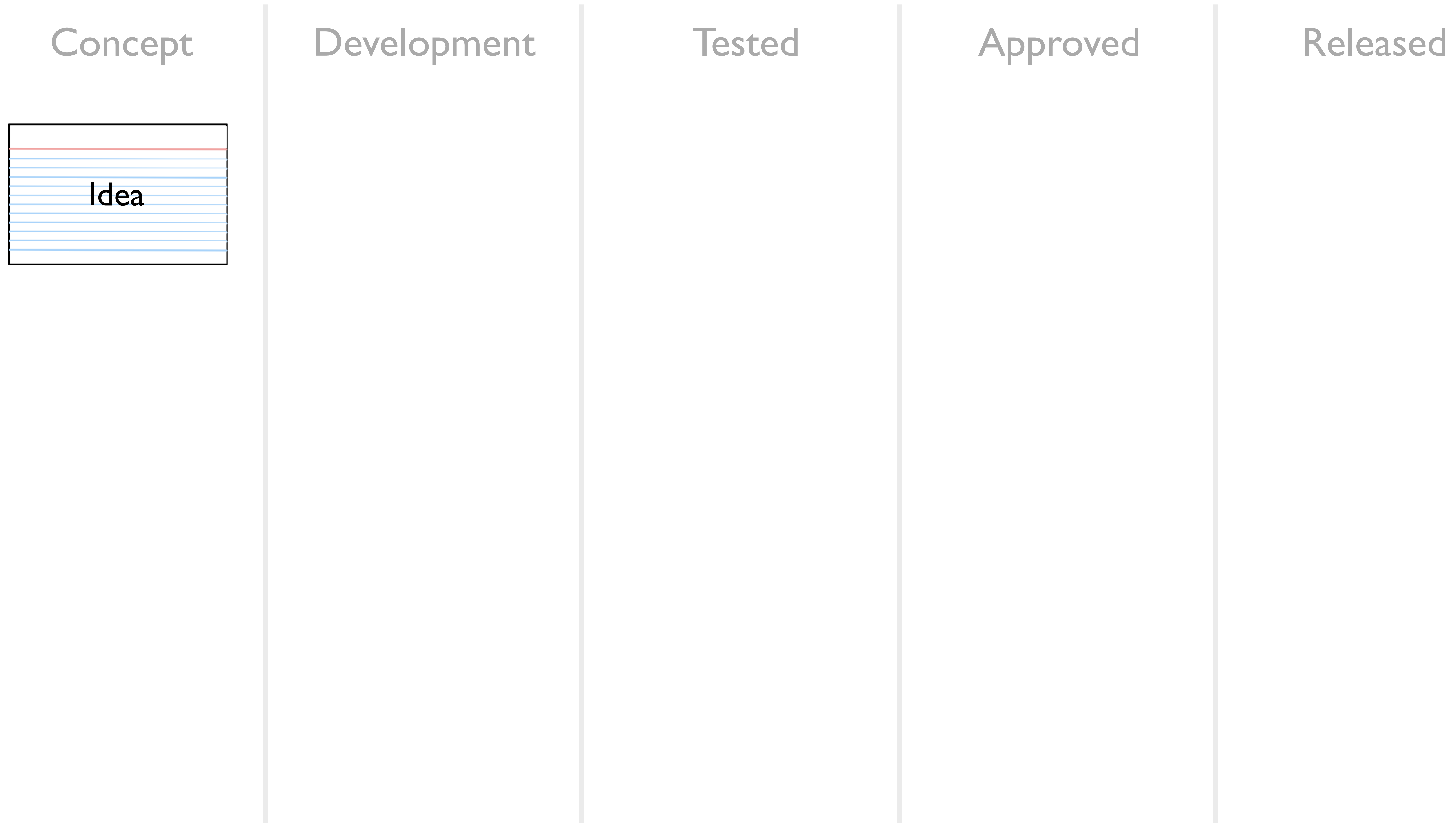

Concept
Development
Tested
Approved
Released
Idea

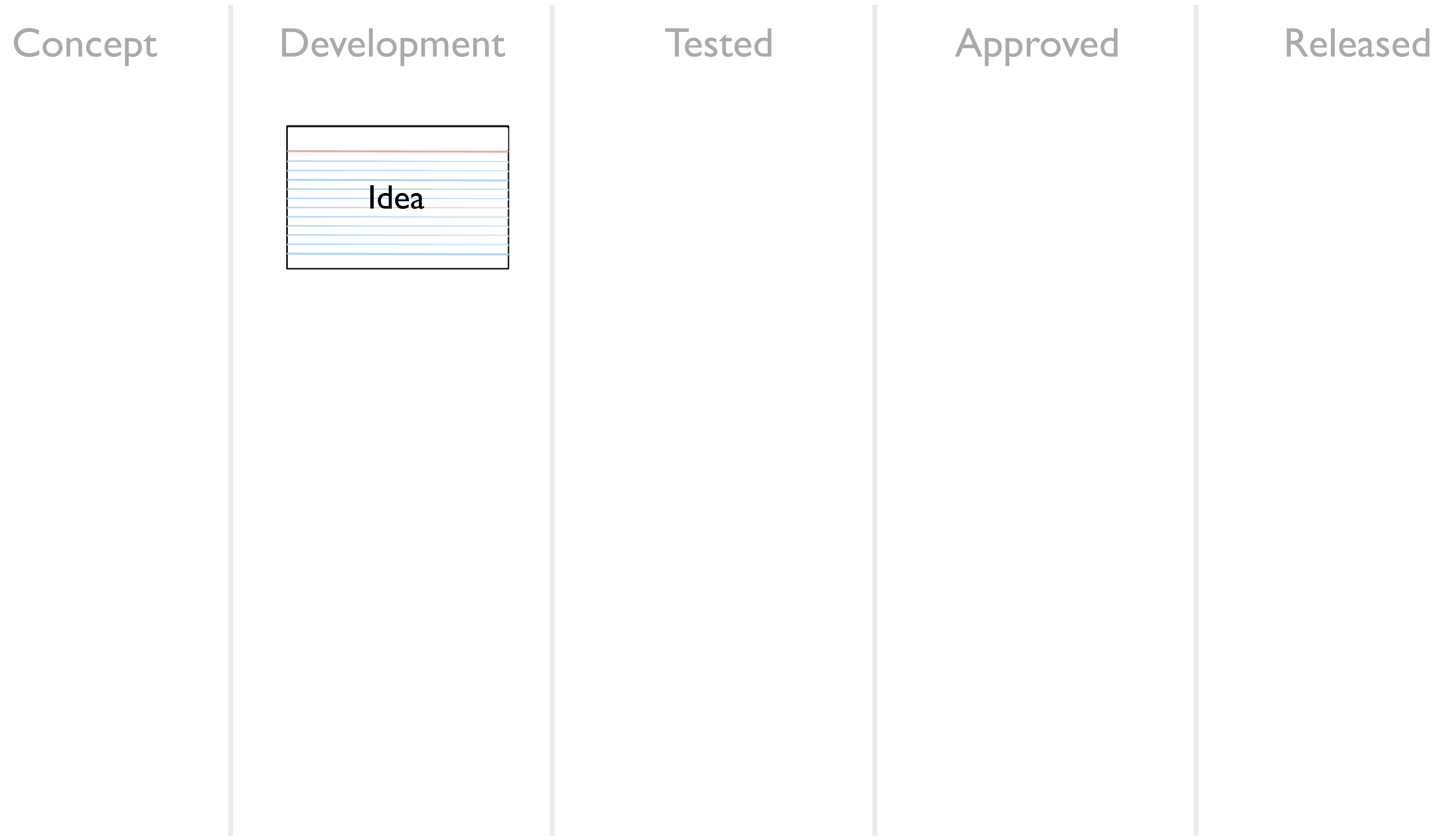

Concept
Development
Tested
Approved
Released
Idea

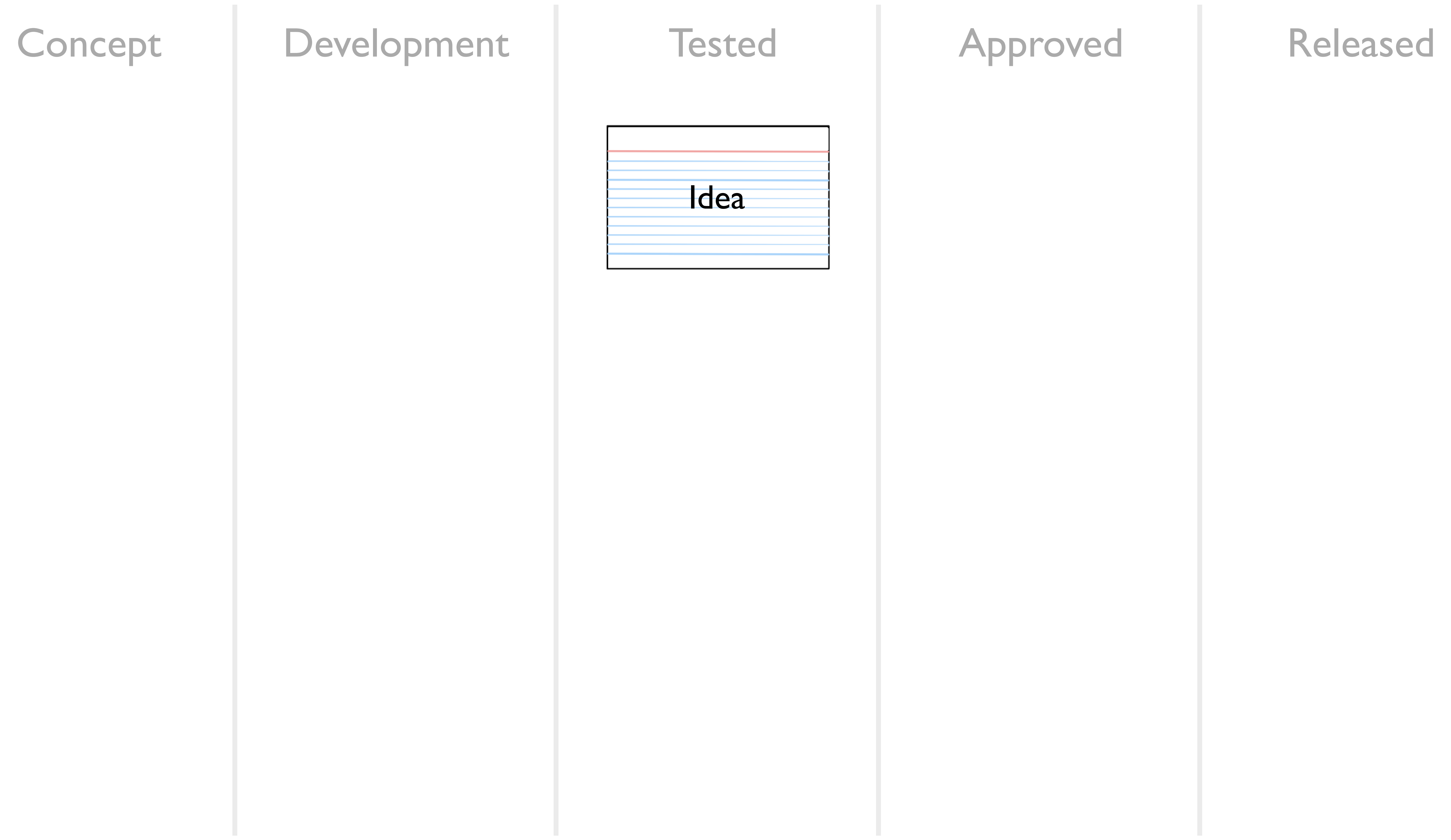

Concept
Development
Tested
Approved
Released
Idea

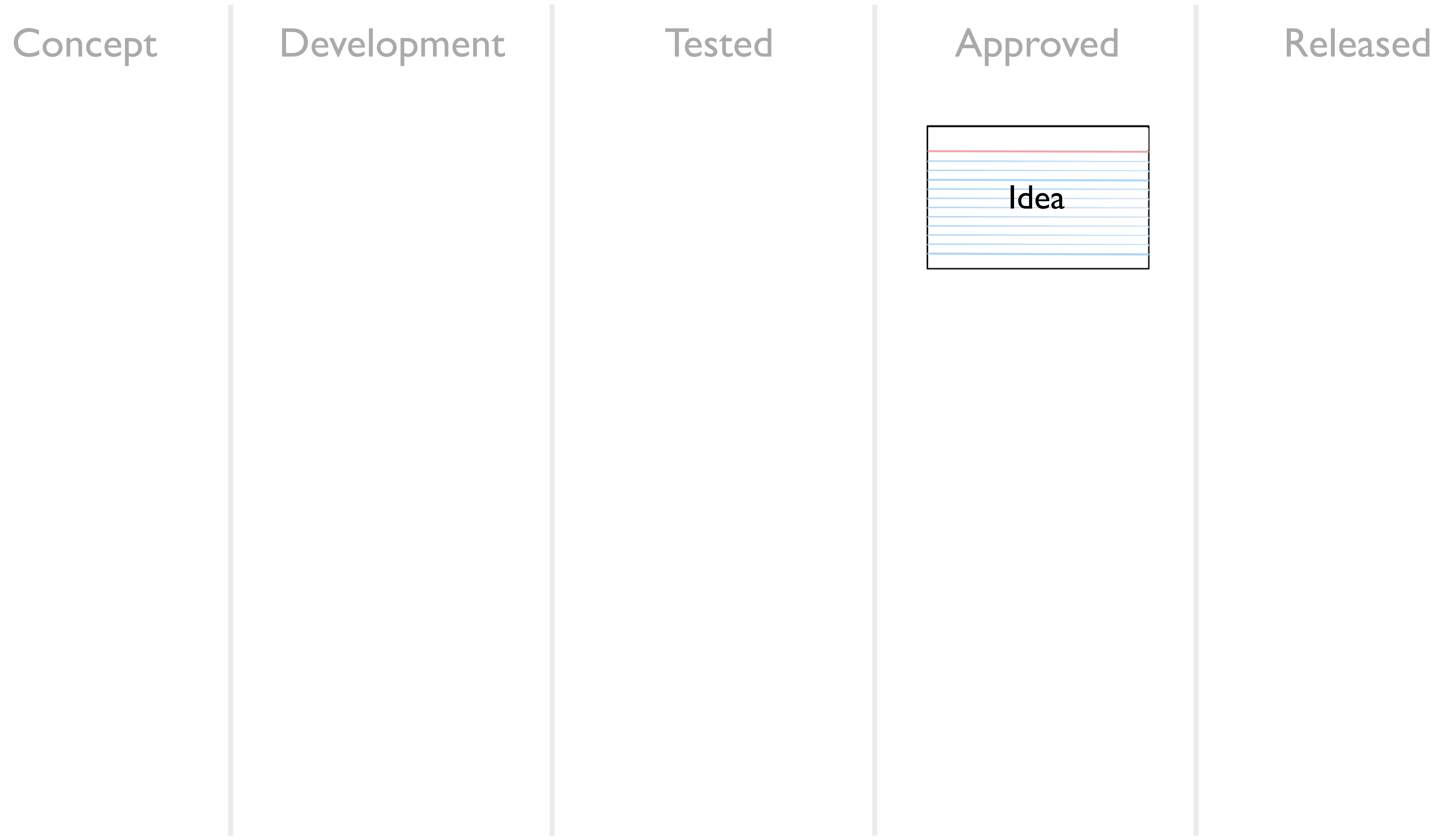

Concept
Development
Tested
Approved
Released
Idea

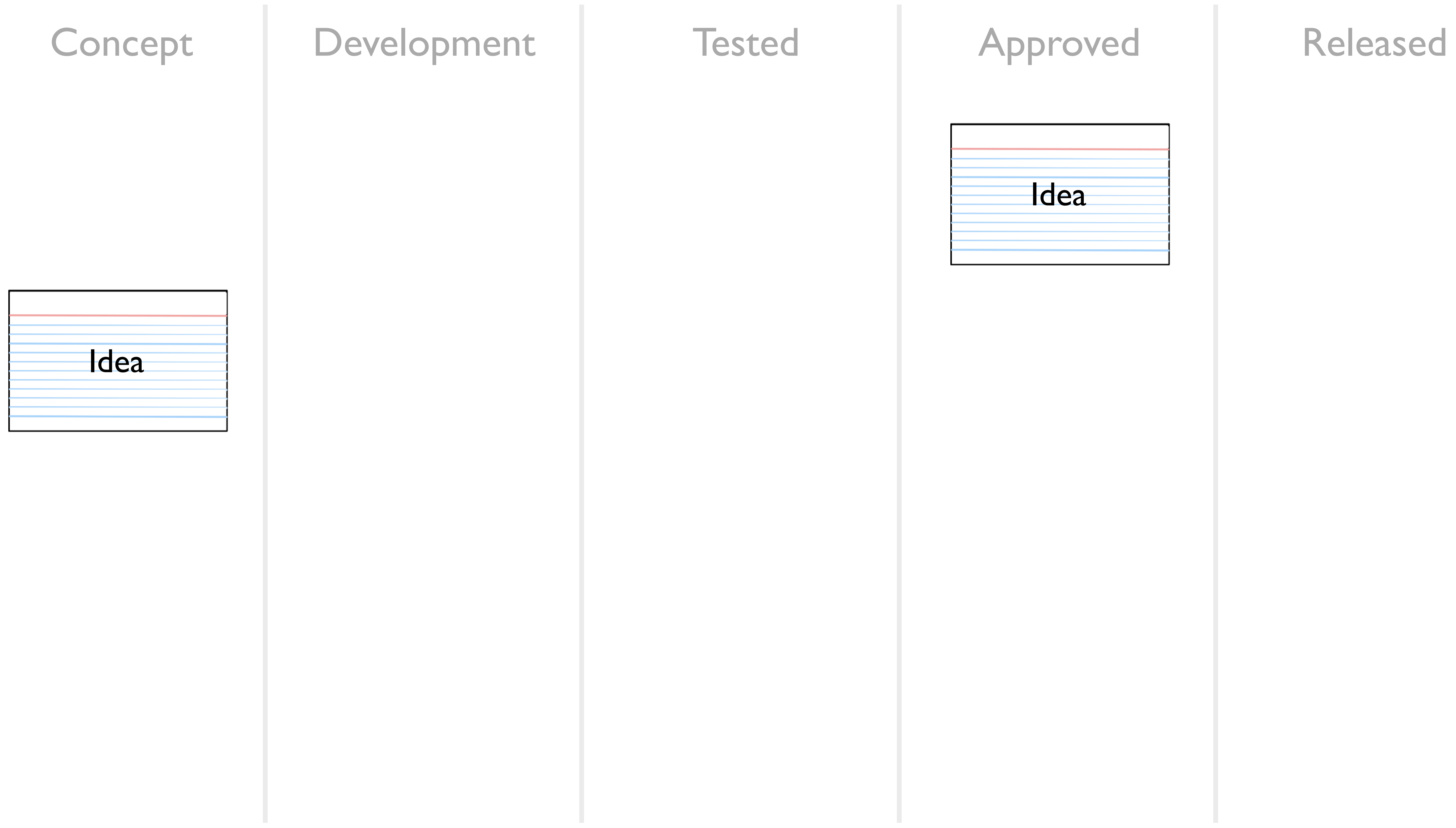
Concept	Development	Tested	Approved	Released
		Idea		

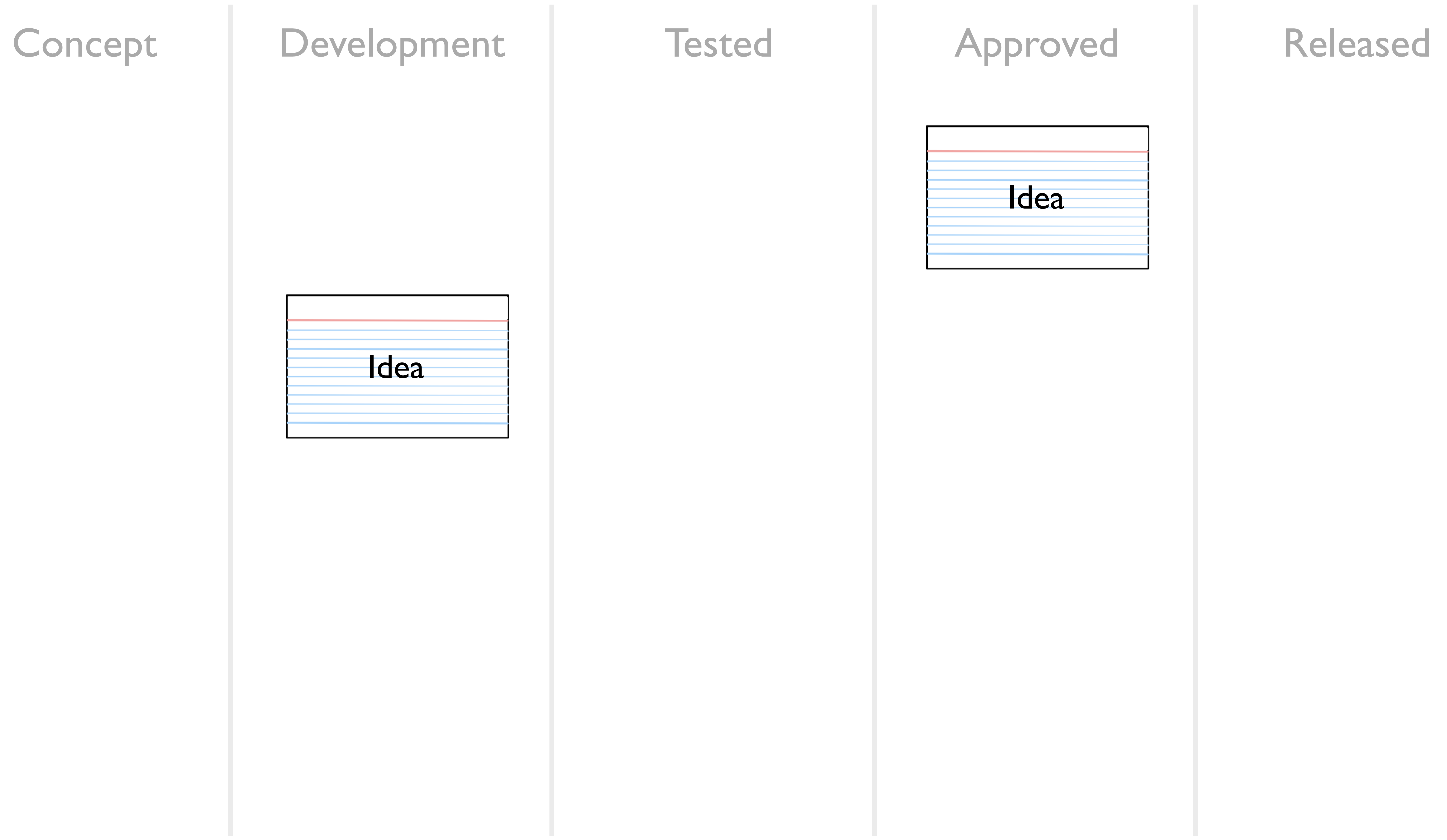

Concept
Development
Tested
Approved
Released
Idea

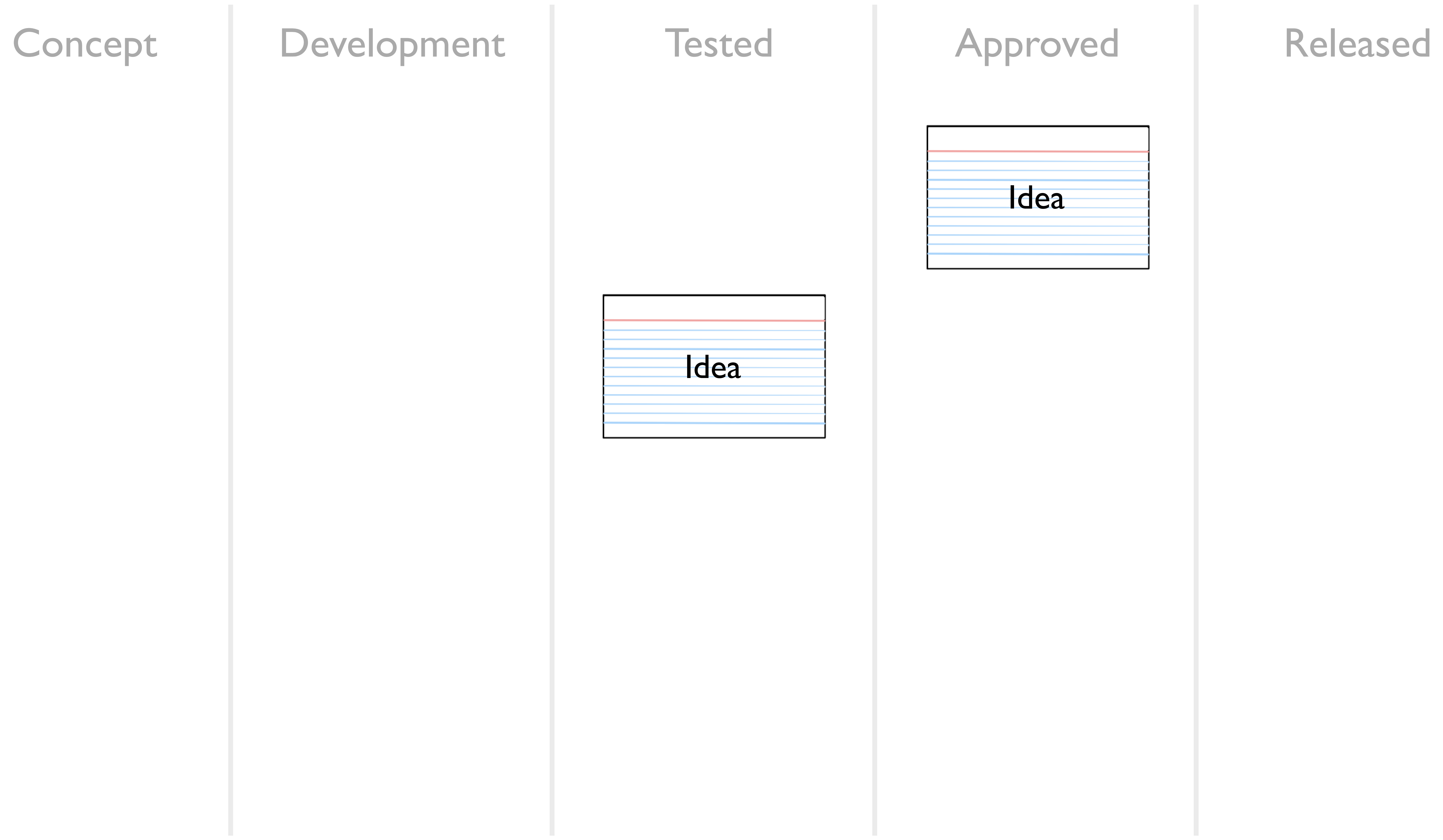

Concept
Development
Tested
Approved
Released
Idea

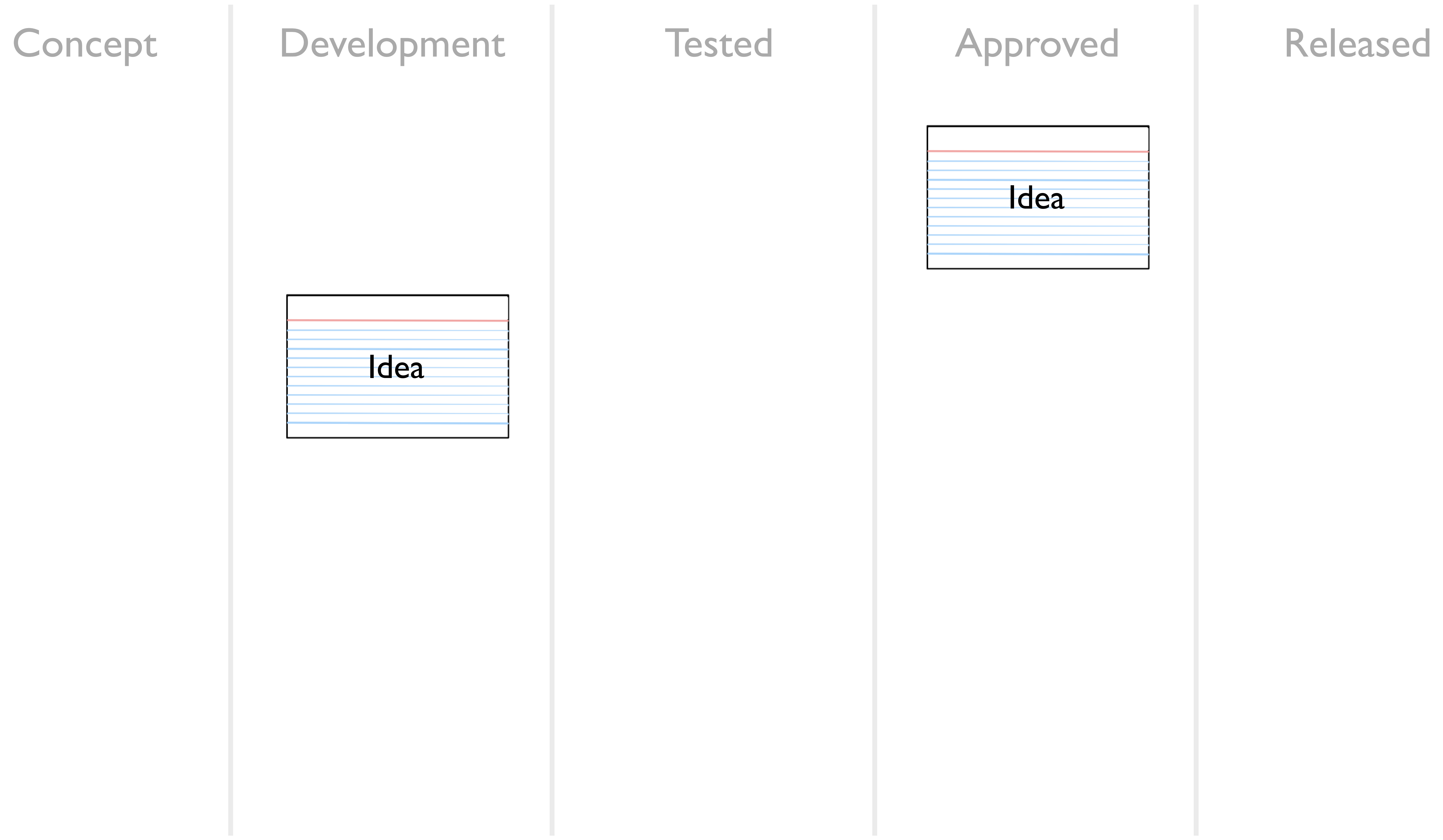

Concept
Development
Tested
Approved
Released
Idea

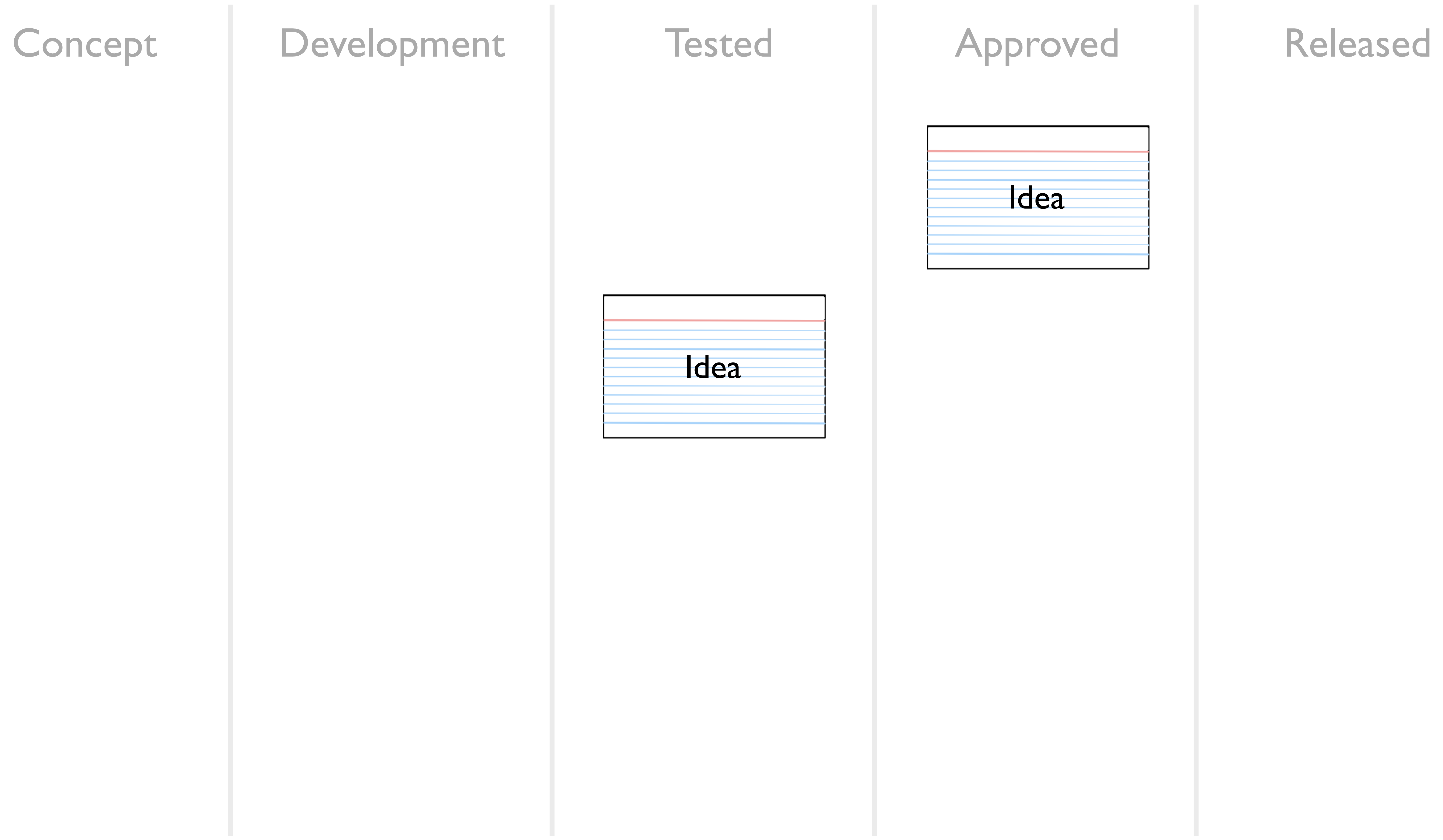

Concept
Development
Tested
Approved
Released
Idea

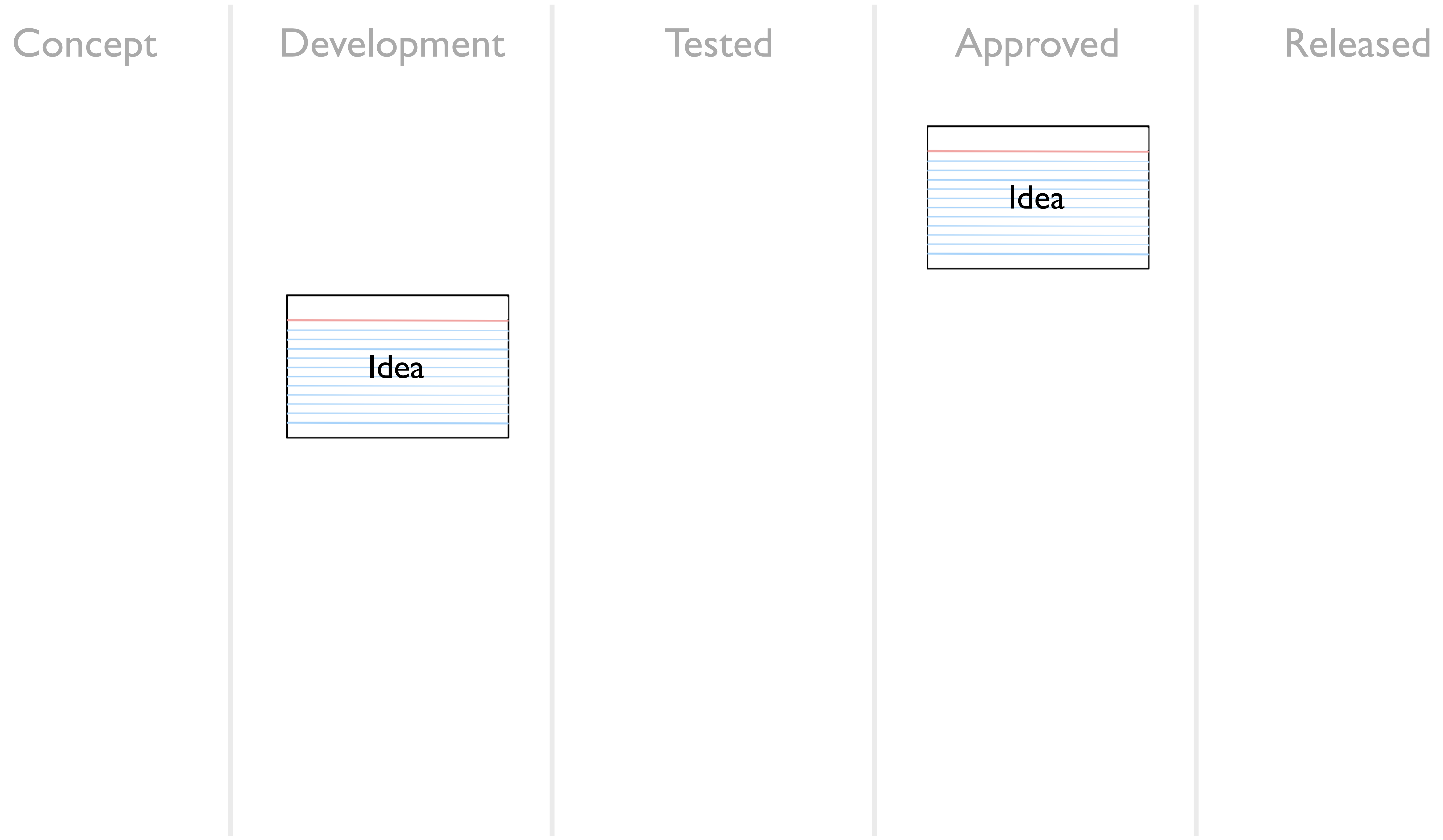

Concept
Development
Tested
Approved
Released
Idea
Idea

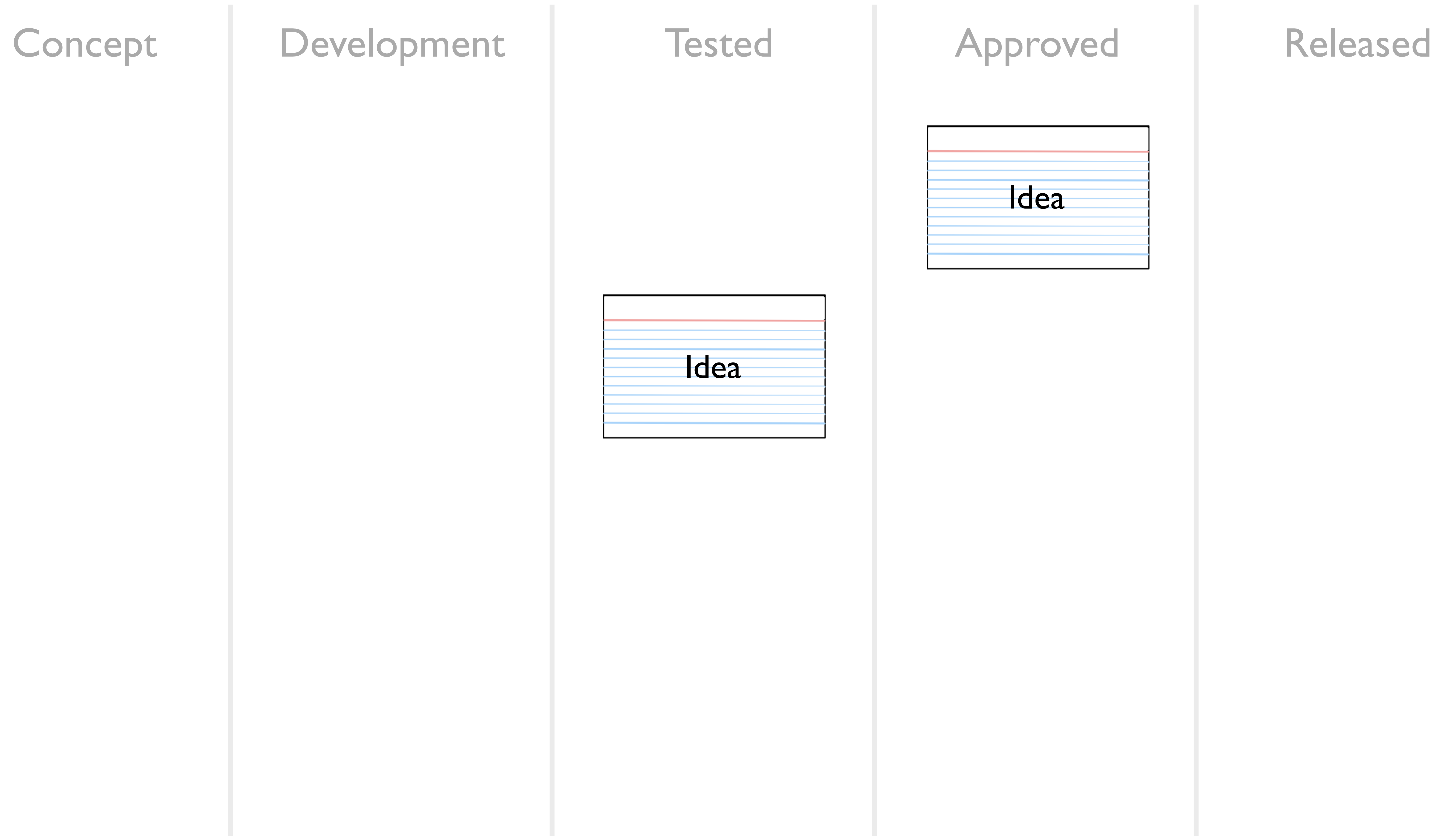

Concept
Development
Tested
Approved
Released
Idea
Idea

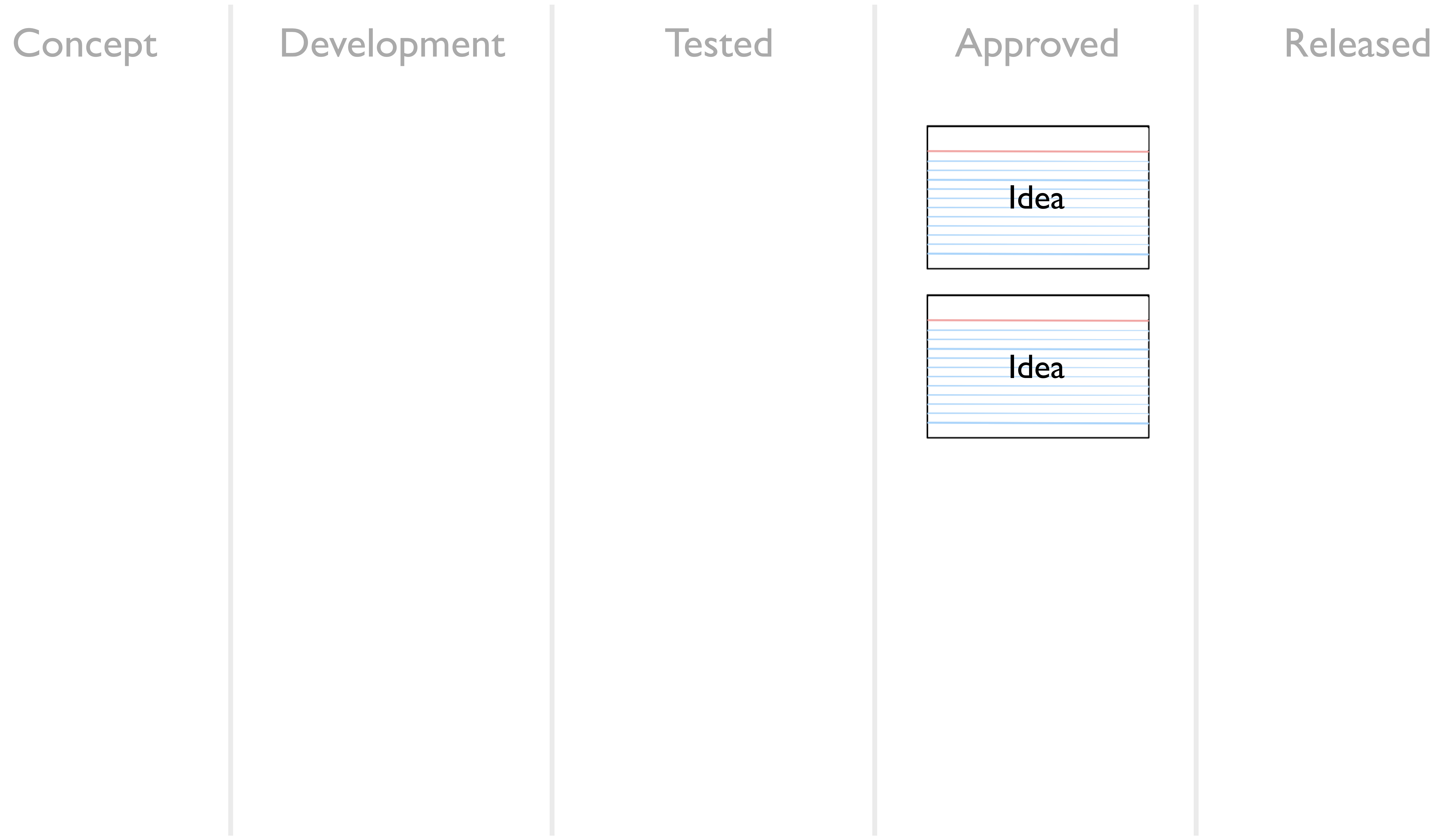

Concept
Development
Tested
Approved
Released
Idea
Idea

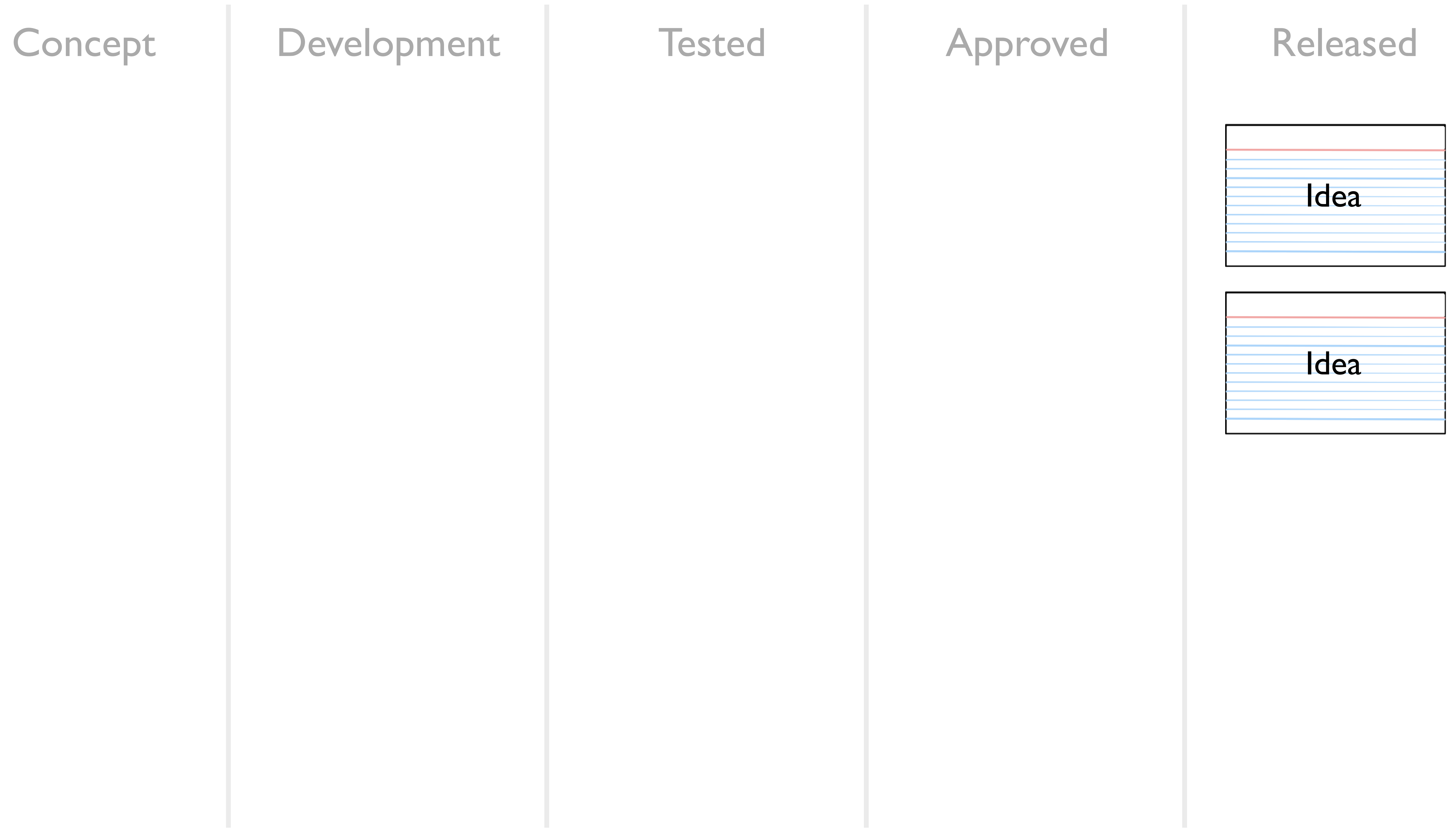

Concept
Development
Tested
Approved
Released
Idea
Idea


Concept
Development
Tested
Approved
Released
Idea
Idea


Concept
Development
Tested
Approved
Released
Idea
Idea


Concept
Development
Tested
Approved
Released
Idea
Idea


Concept
Development
Tested
Approved
Released
Idea
Idea

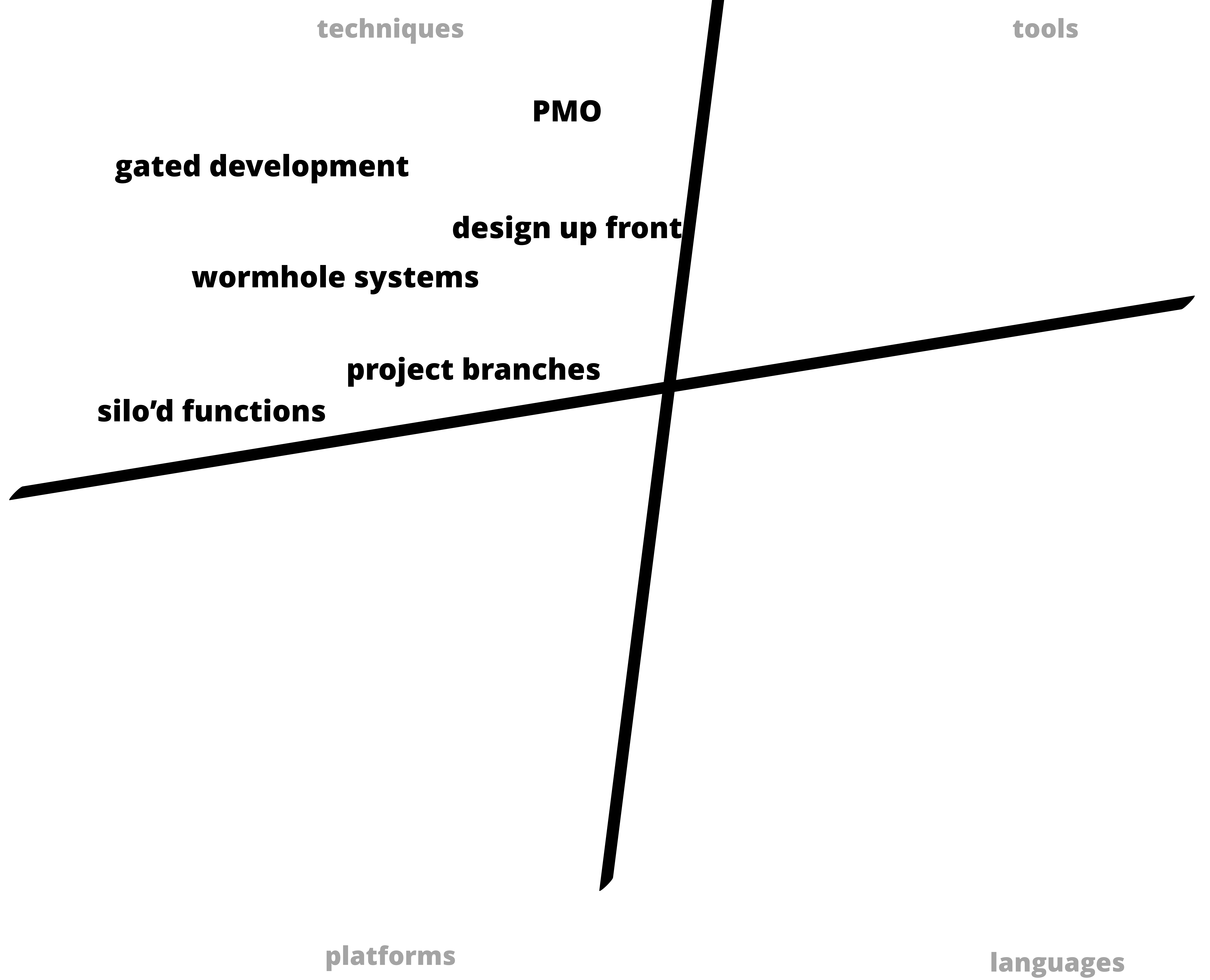

Concept
Development
Tested
Approved
Released
Idea
Idea

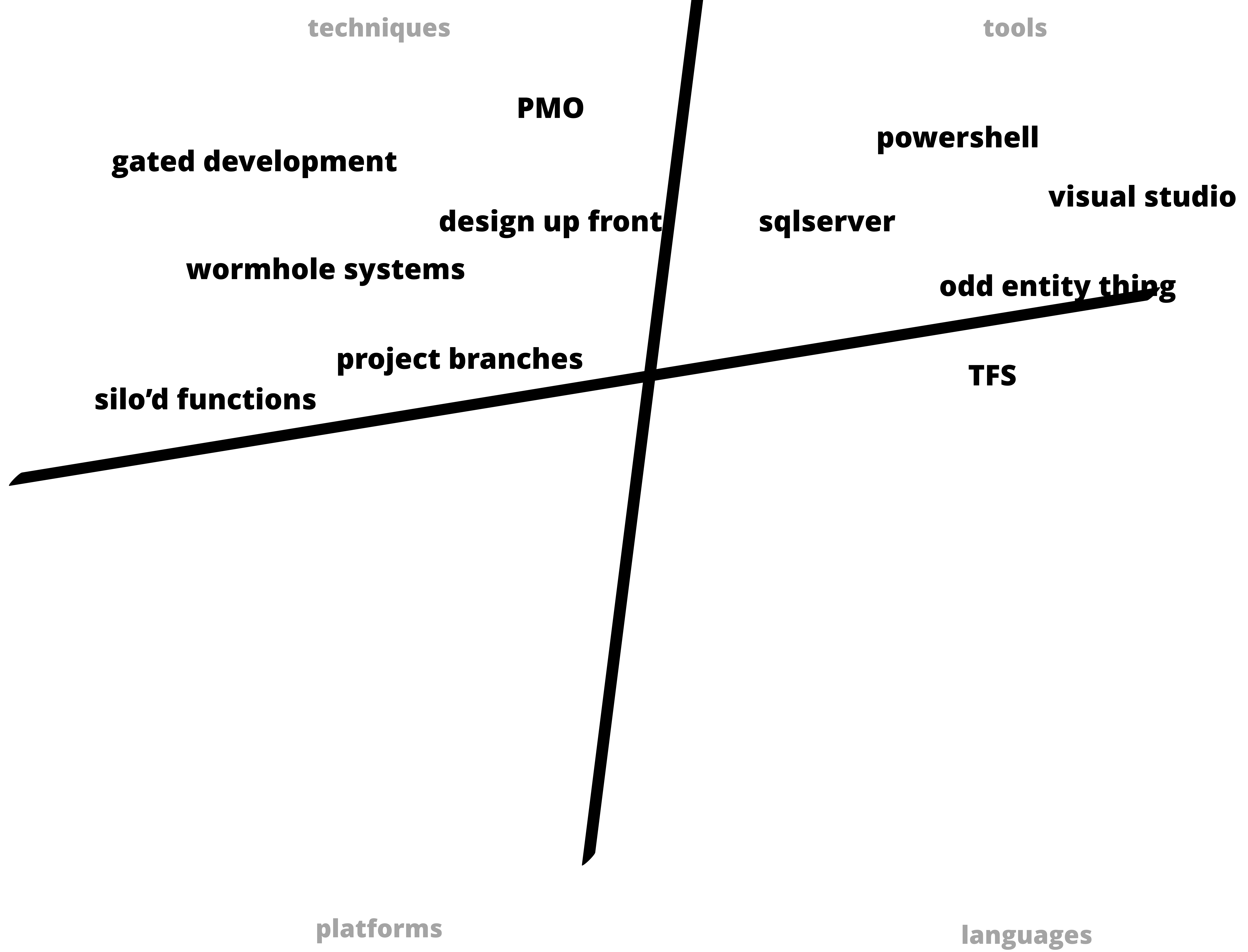
techniques

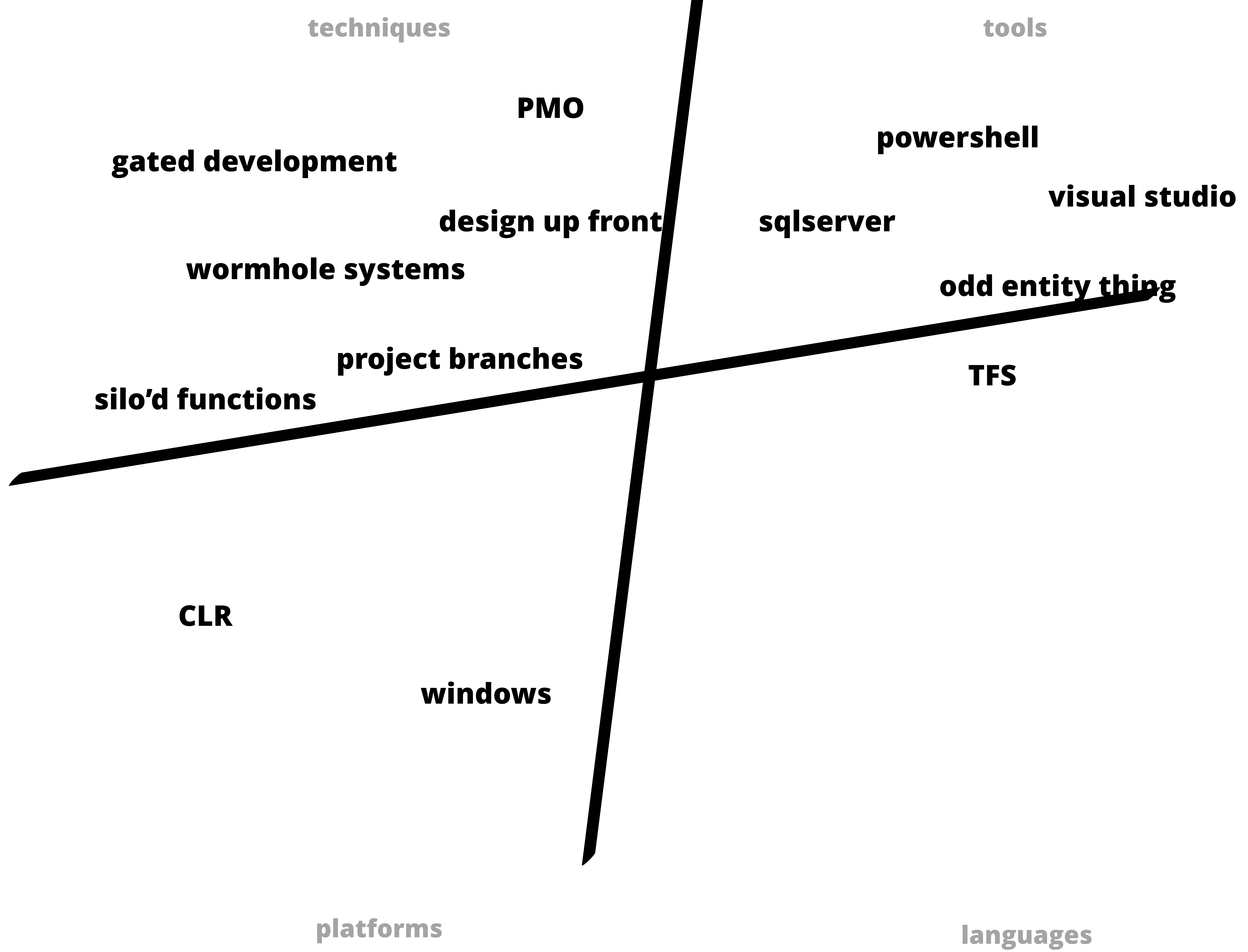
tools

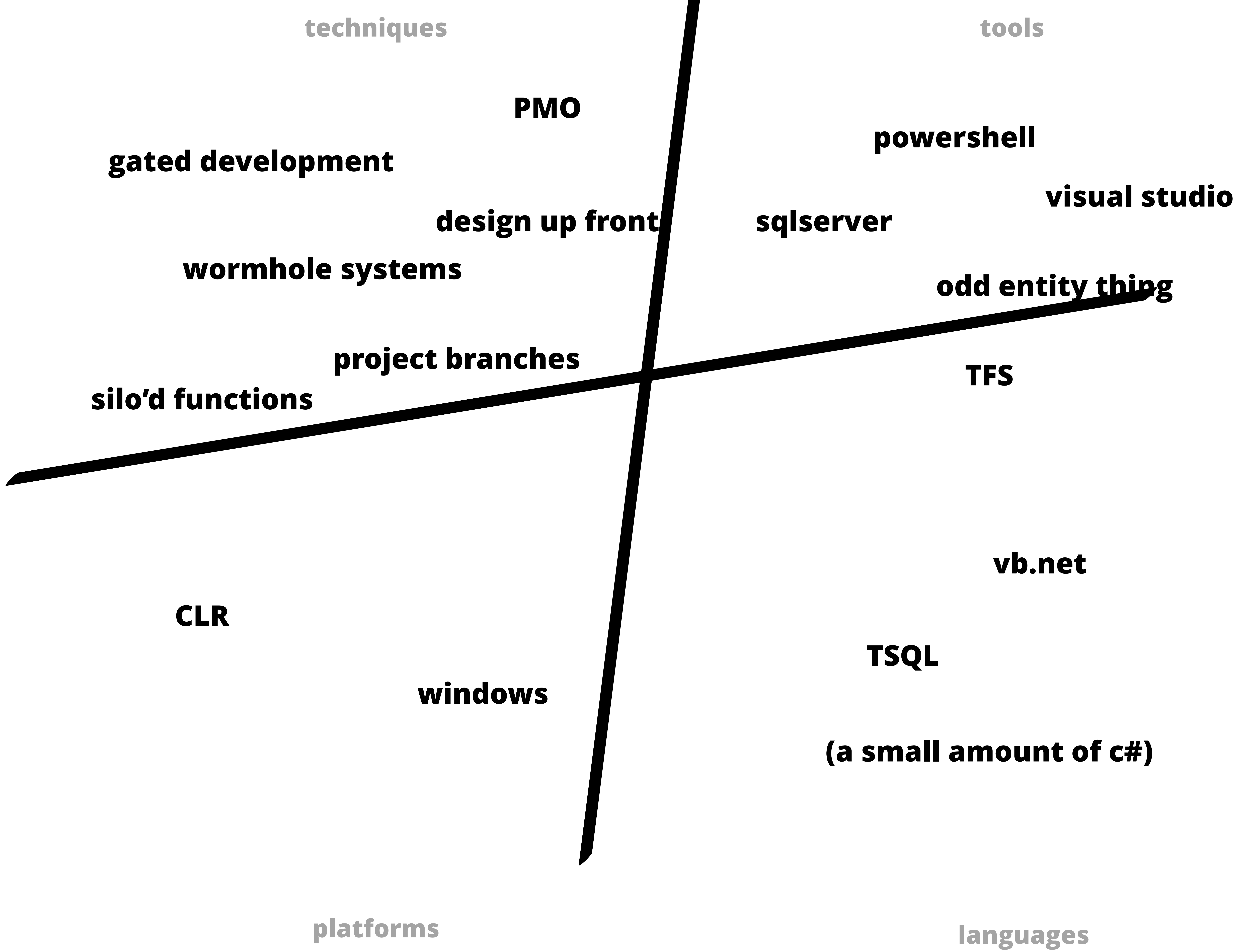
platforms

languages


techniques
tools
PMO
gated development
design up front
wormhole systems
project branches
silo’d functions
platforms
languages


techniques
tools
PMO
powershell
gated development
visual studio
design up front
sqlserver
wormhole systems
odd entity thing
project branches
TFS
silo’d functions
platforms
languages


techniques
tools
PMO
powershell
gated development
visual studio
design up front
sqlserver
wormhole systems
odd entity thing
project branches
TFS
silo'd functions
CLR
windows
platforms
languages


techniques
PMO
gated development
design up front
wormhole systems
project branches
silo'd functions
tools
powershell
visual studio
sqlserver
odd entity thing
TFS
platforms
CLR
windows
languages
vb.net
TSQL
(a small amount of c#)

Y U NO
USE
OTHER STUFF?
quickmeme.com

our heroes start their journey

CONSULTANTS!
HELL YEAH!

NASA FACTS

An Educational Services Publication of the National Aeronautics and Space Administration

(G–12–62)
Supplement

A REPORT ON THE FIRST RELAY COMMUNICATIONS SATELLITE

January 14, 1963

The first Relay satellite, launched December 13, 1962, could not at first function properly because of an abnormal power drain on its storage batteries. The problems relative to the satellite had been partially resolved by January 3, 1963, making possible the beginning of experiments in transatlantic communication.

On Relay's fifth orbit, some 14 hours after launch, the ground test station at Nutley, New Jersey, checked the satellite's condition. The satellite's voltage was indicated at 22.5, which is below the lower limit of 24 volts required for operation of the communications equipment with-
damage to the battery.

The trouble was traced to the voltage regulator in the No. 1 transponder—the receiving, amplifying, and transmitting apparatus in Relay. Relay is equipped with two identical transponders, each with its own voltage regulator. The voltage regulator channels power to the transponder at a proper voltage and acts as an on-off switch for the transponder.

Telemetry showed that the regulator was conducting power to the transponder even though it was nominally off. As a result, it was partially powering the transponder, and draining the batteries. Extensive tests and analyses indicated that the main power transistor for the voltage regulator had temperature characteristics that could account for the equipment's malfunction

tor No. 1. Because of difficulties experienced with the command system of transponder No. 1, project managers decided to employ the other transponder. On January 3, 1963, they activated transponder No. 2 and carried out a continuing sequence of transatlantic television, telephone, and teletype communication tests.

Some difficulty has been encountered with the command system of the satellite. However, techniques have been developed for satisfactorily commanding the equipment.

Encouraged by the success of the tests, NASA, in cooperation with broadcasting companies of the United States and Europe, scheduled a public telecast via Relay. On January 9, 1963, Relay carried a television program from the National Gallery of Art in Washington, D.C., to stations in France and Great Britain. British and French viewers saw the unveiling of the Mona Lisa painting in the National Gallery and President Kennedy and others who were present at the unveiling ceremony. (Leonardo da Vinci's Mona Lisa was loaned to the United States by France.) Reception in Europe was excellent.

The television pictures from the gallery were transmitted conventionally to the Relay ground station at Andover, Maine, from which they were beamed to the satellite for retransmittal to Europe.

A second Relay launch is scheduled in the sec-

“the measure of a body's resistance to changes in velocity”

inertia

organisational inertia

technical inertia

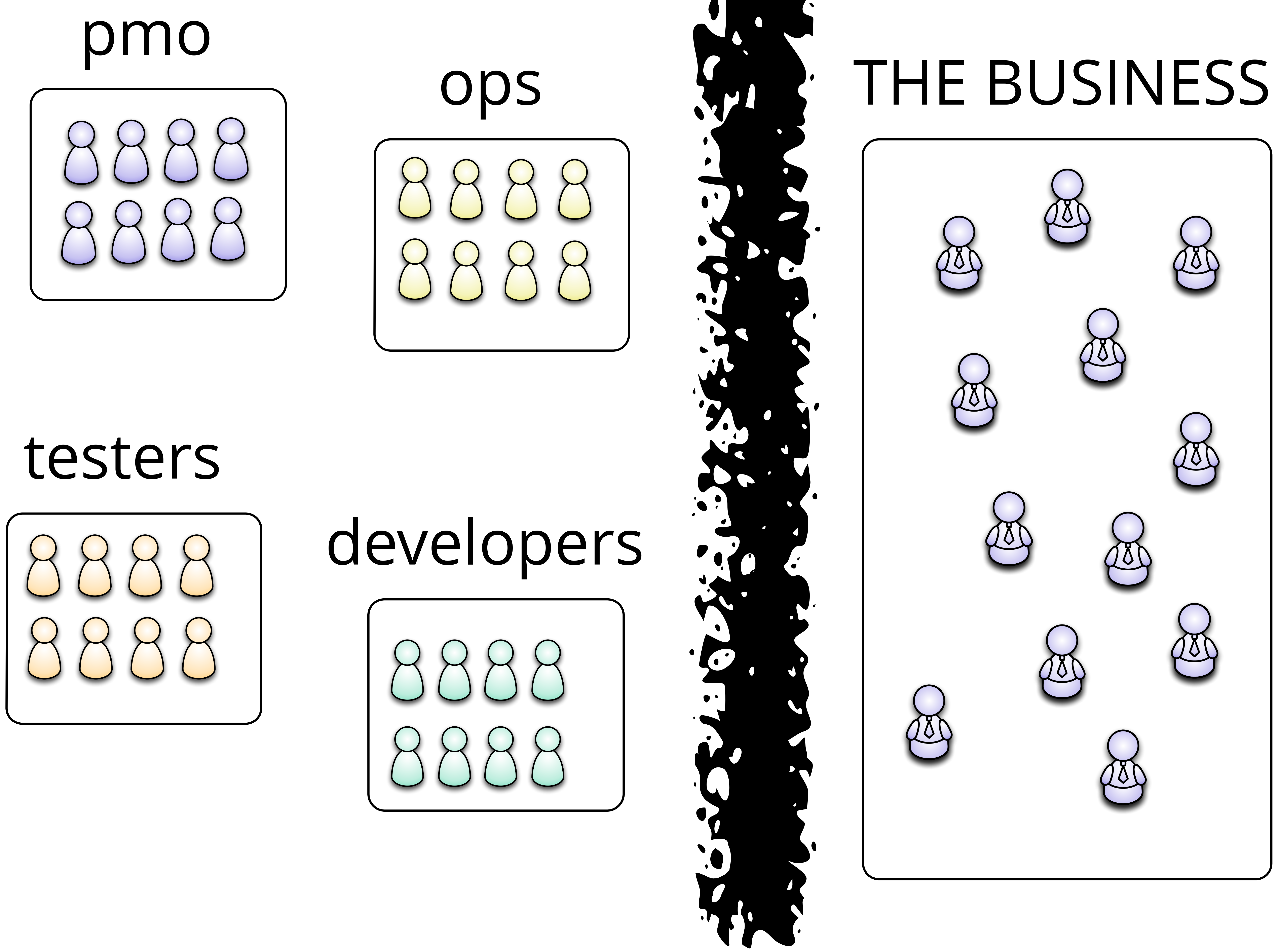
product teams

agile software development

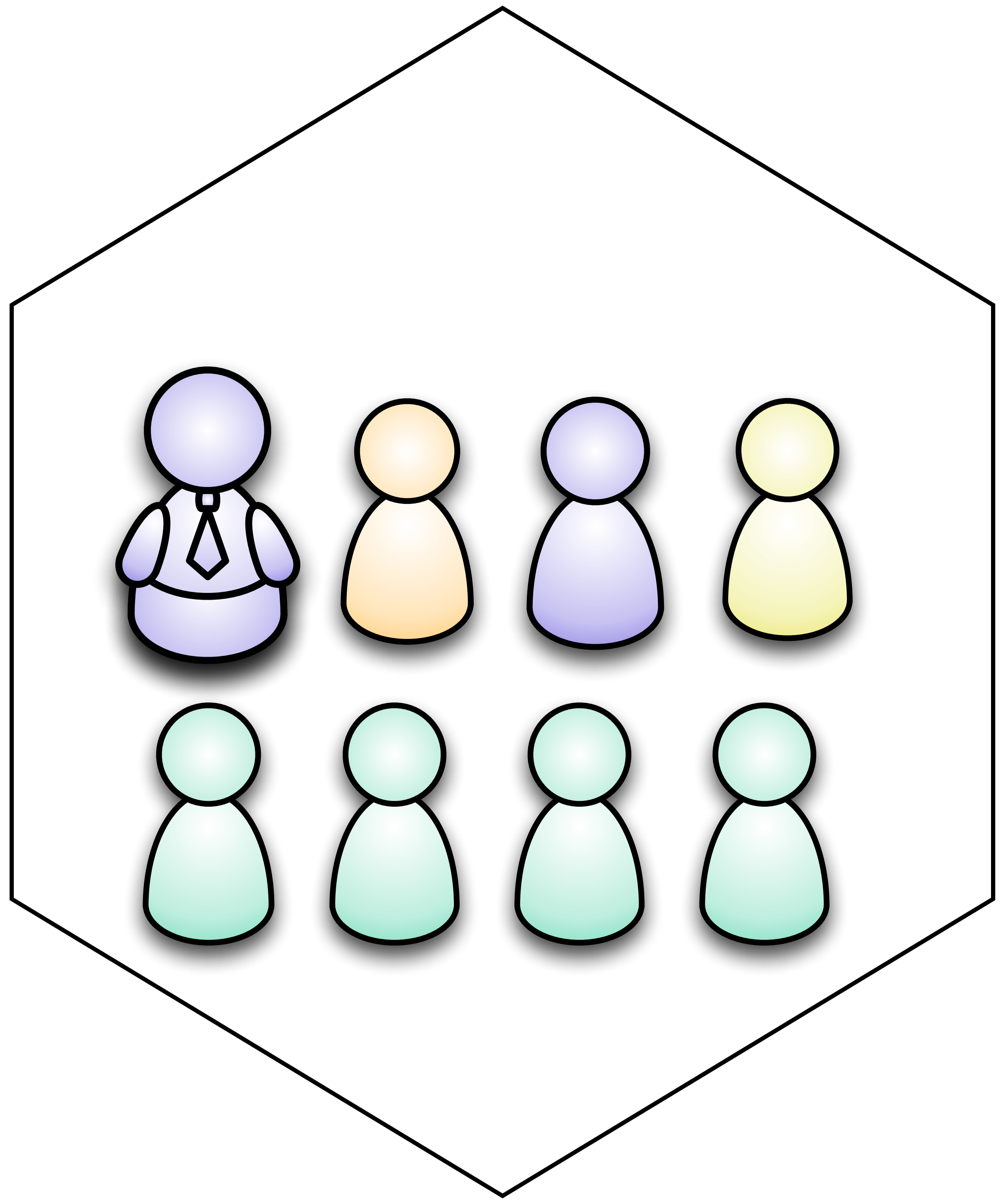
*Start small, low risk to **build trust***

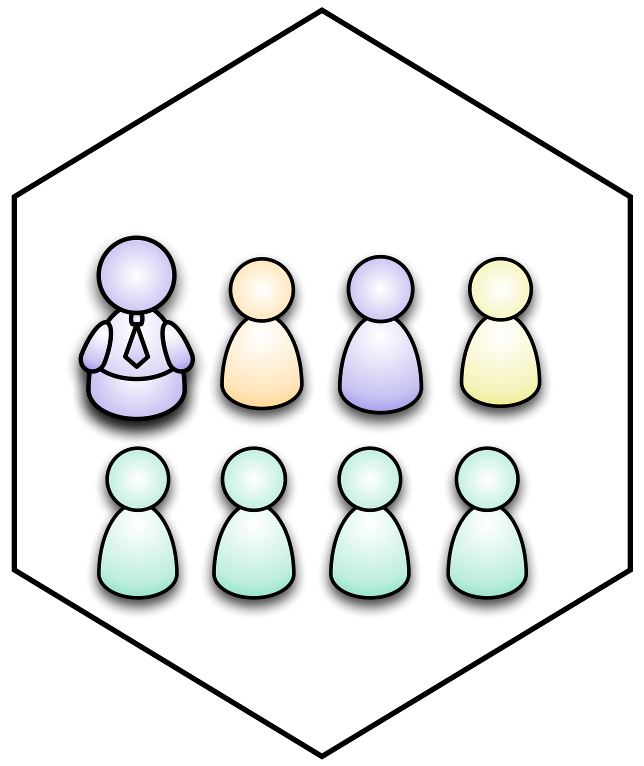
service oriented architecture

product teams



product teams



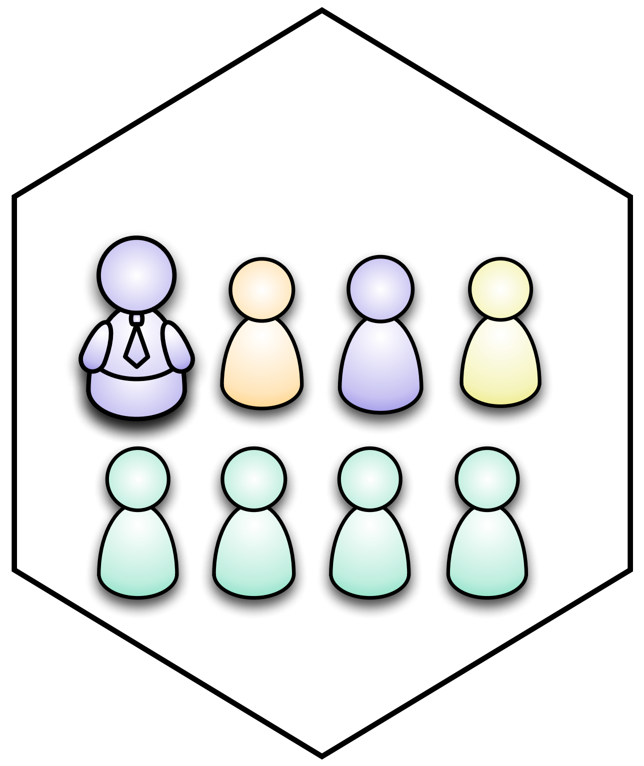


product teams

agile software development

Start small, low risk to ***build trust***

service oriented architecture



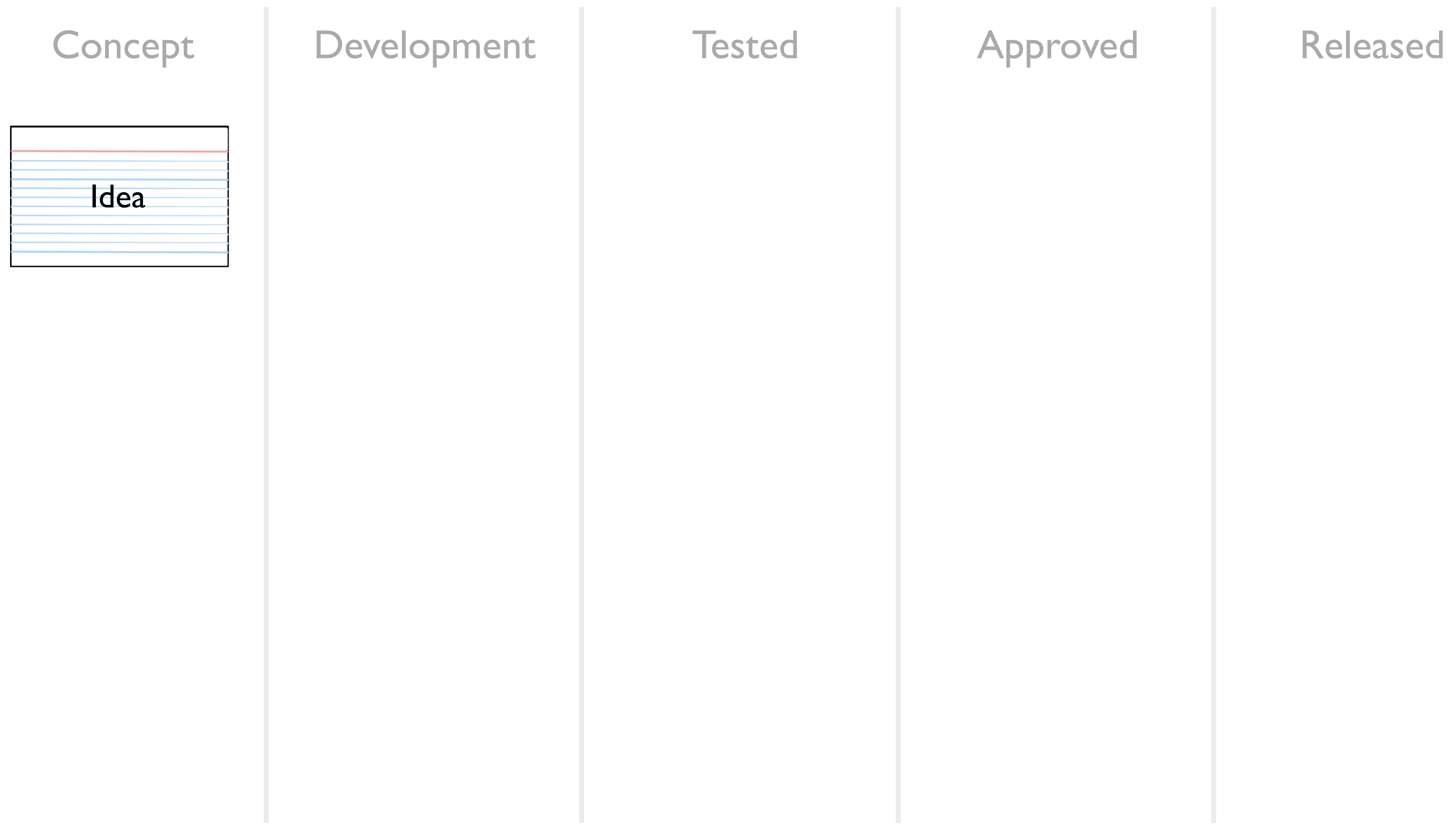
product teams

agile software development

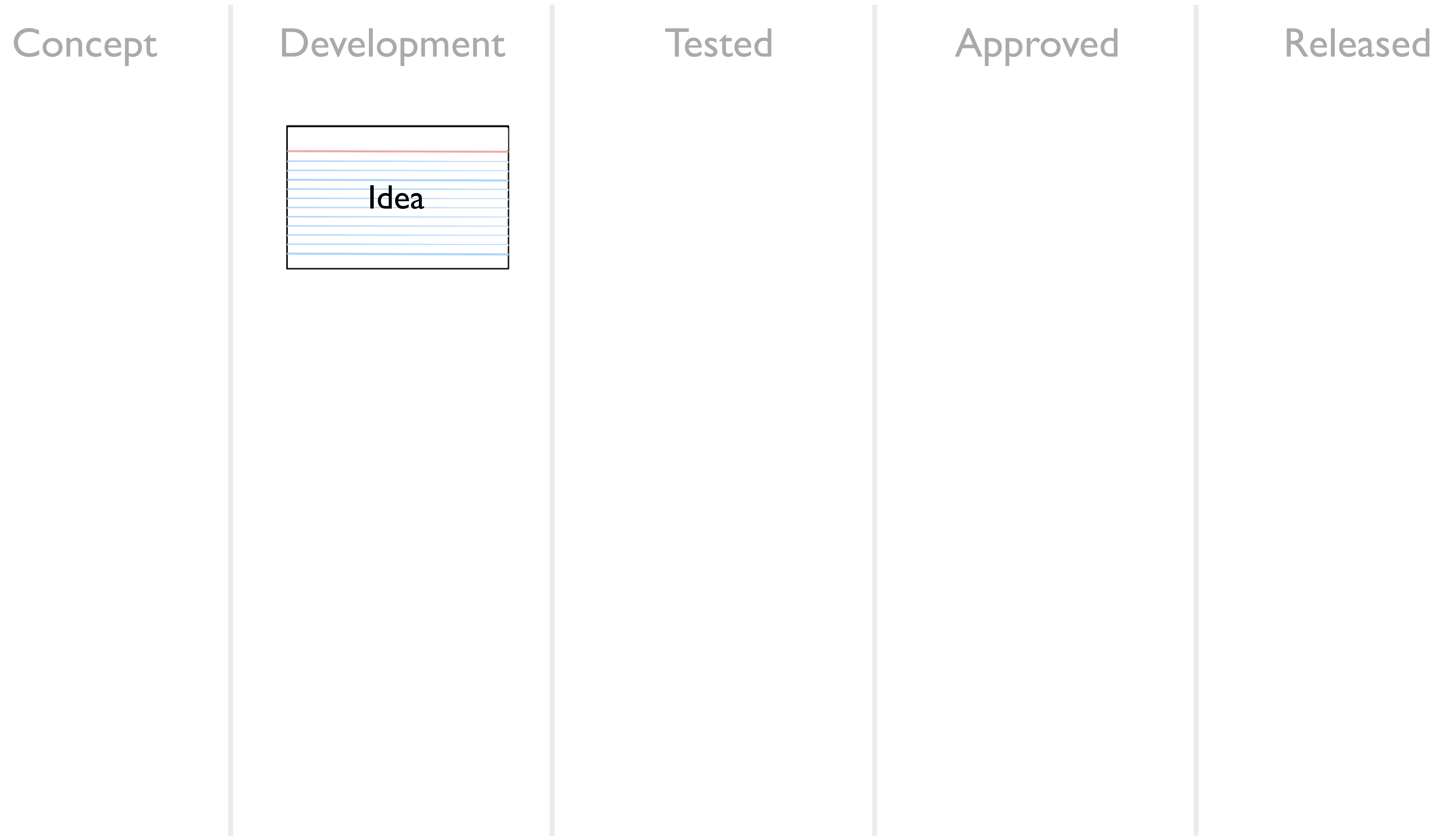
*Start small, low risk to **build trust***

service oriented architecture

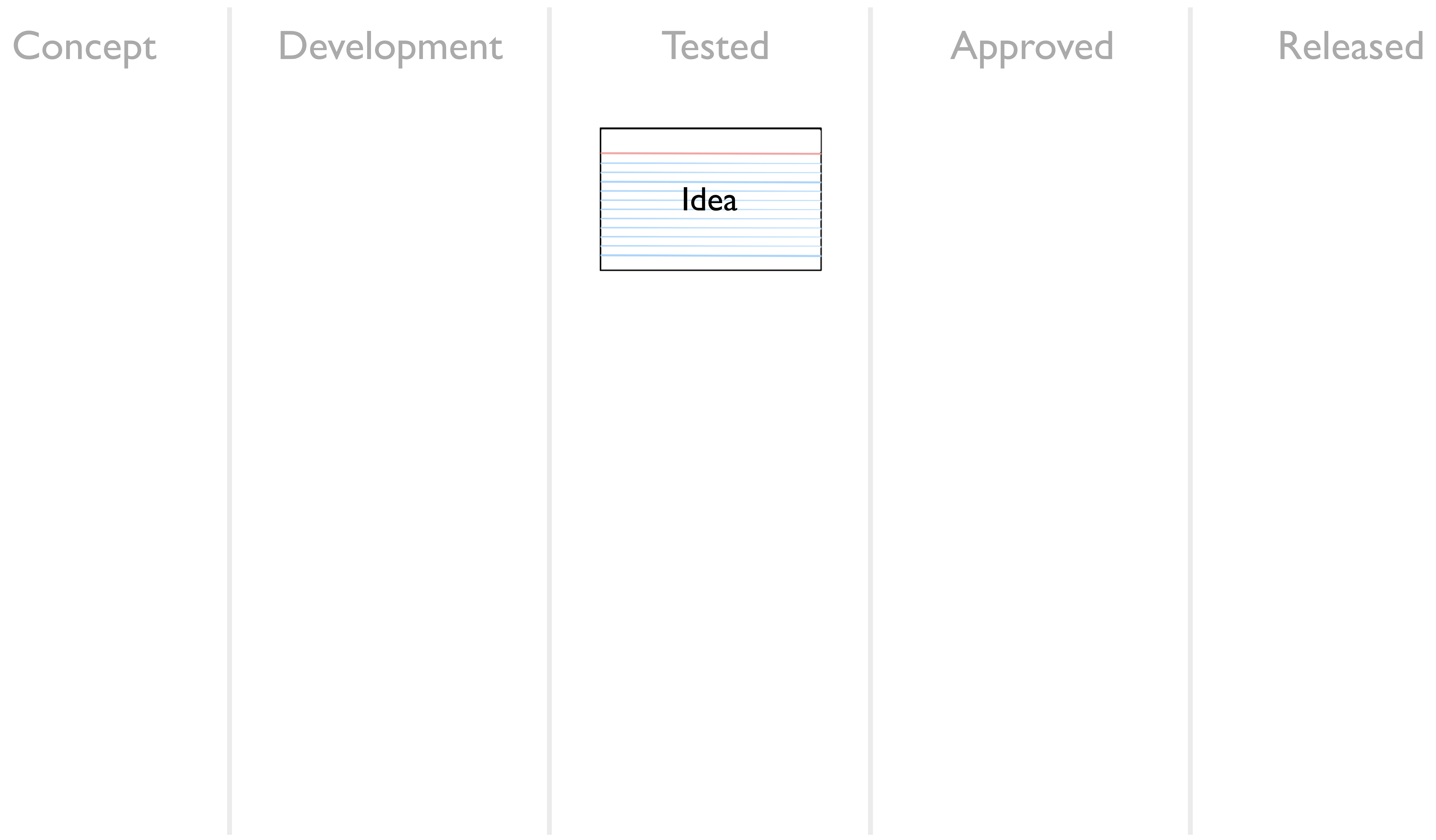
agile software development



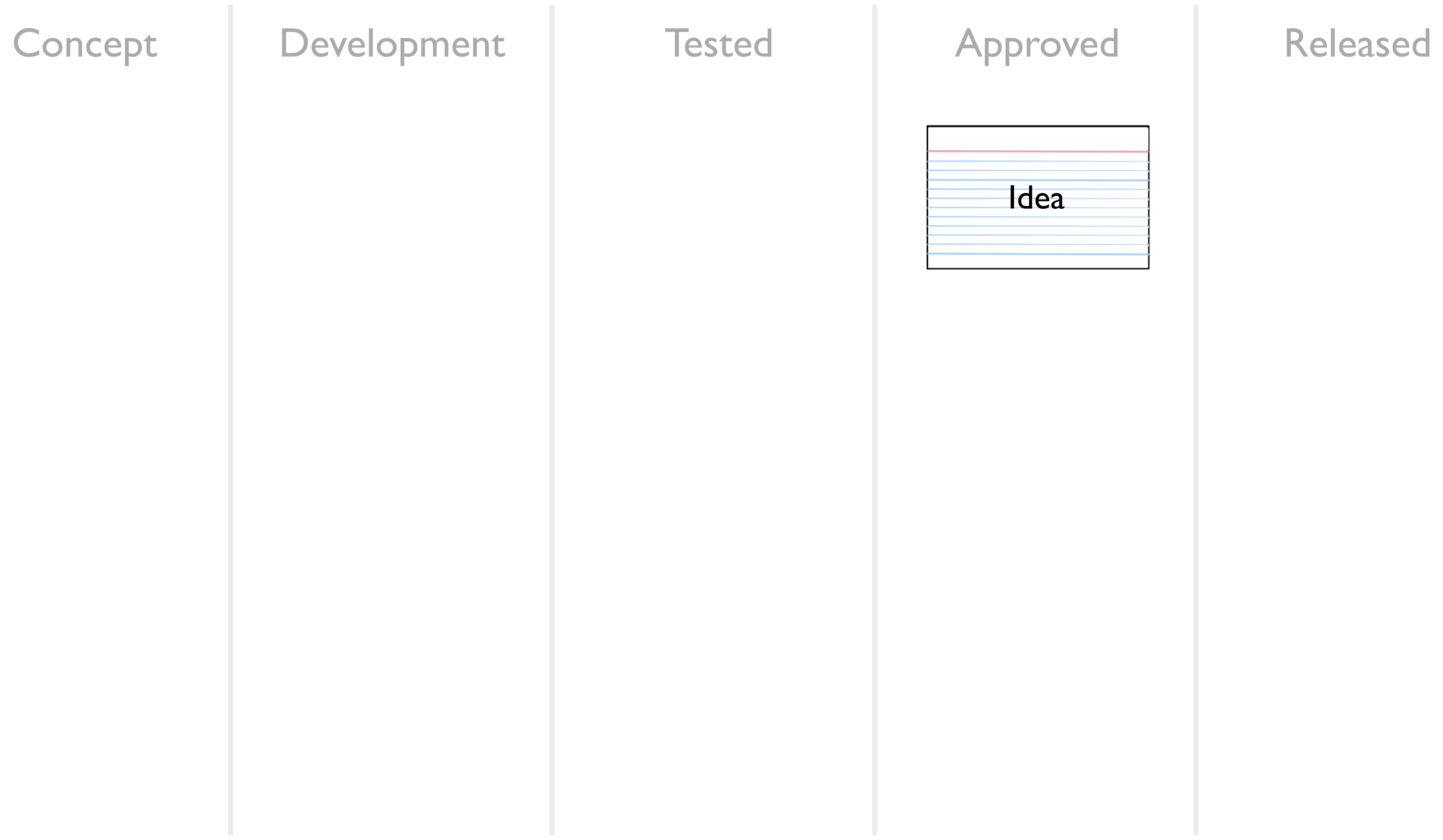
agile software development



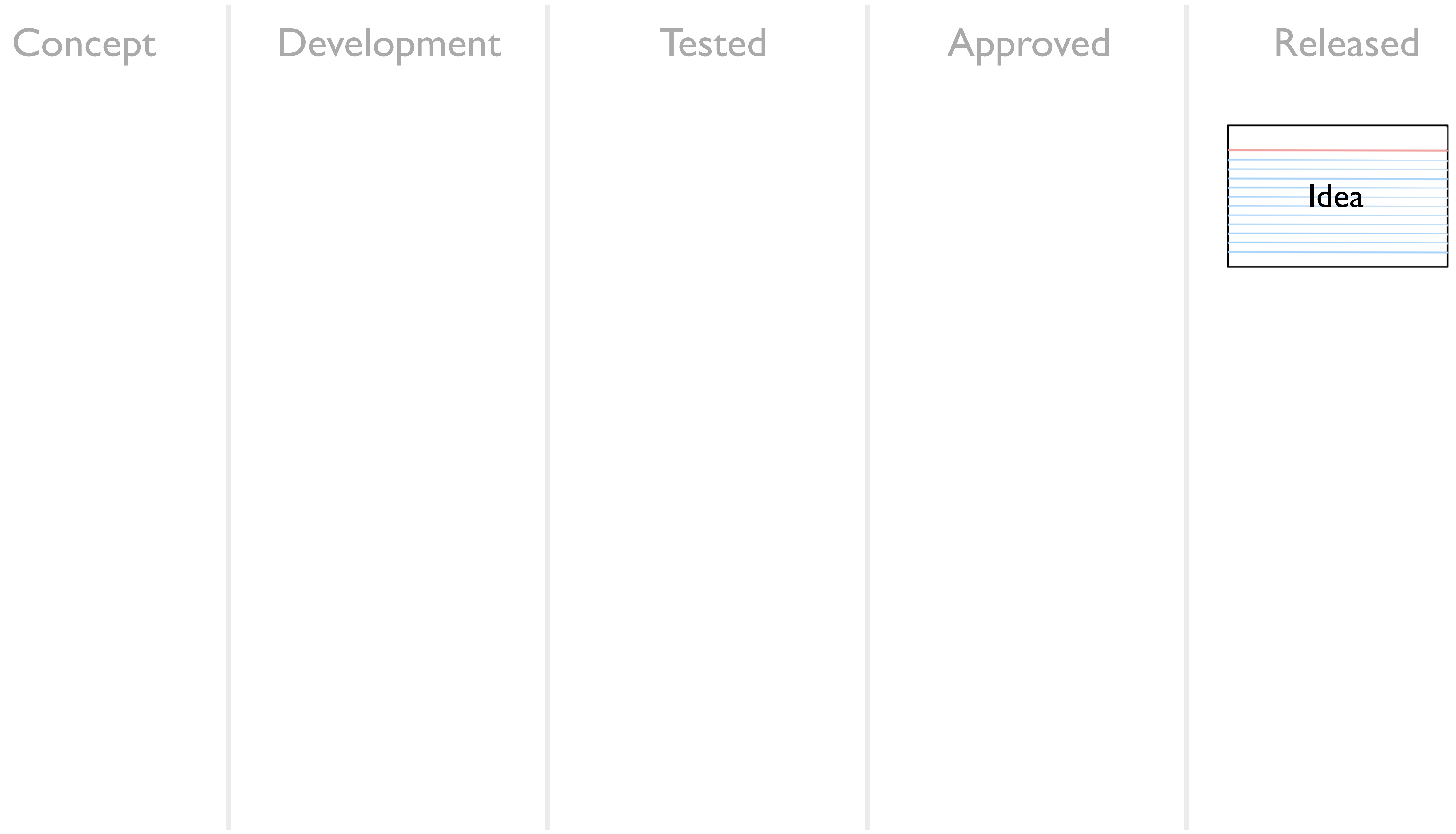
agile software development



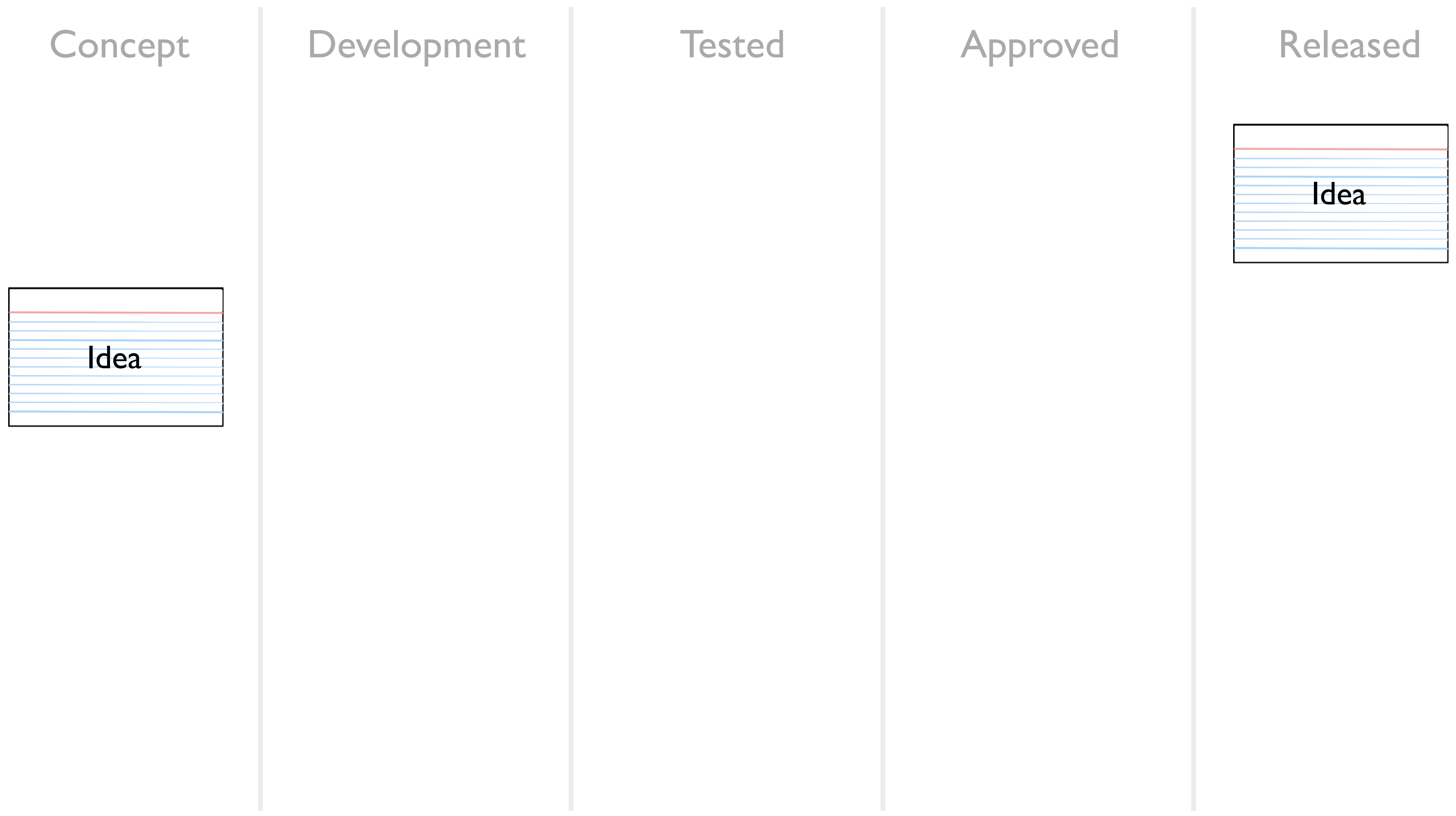
agile software development



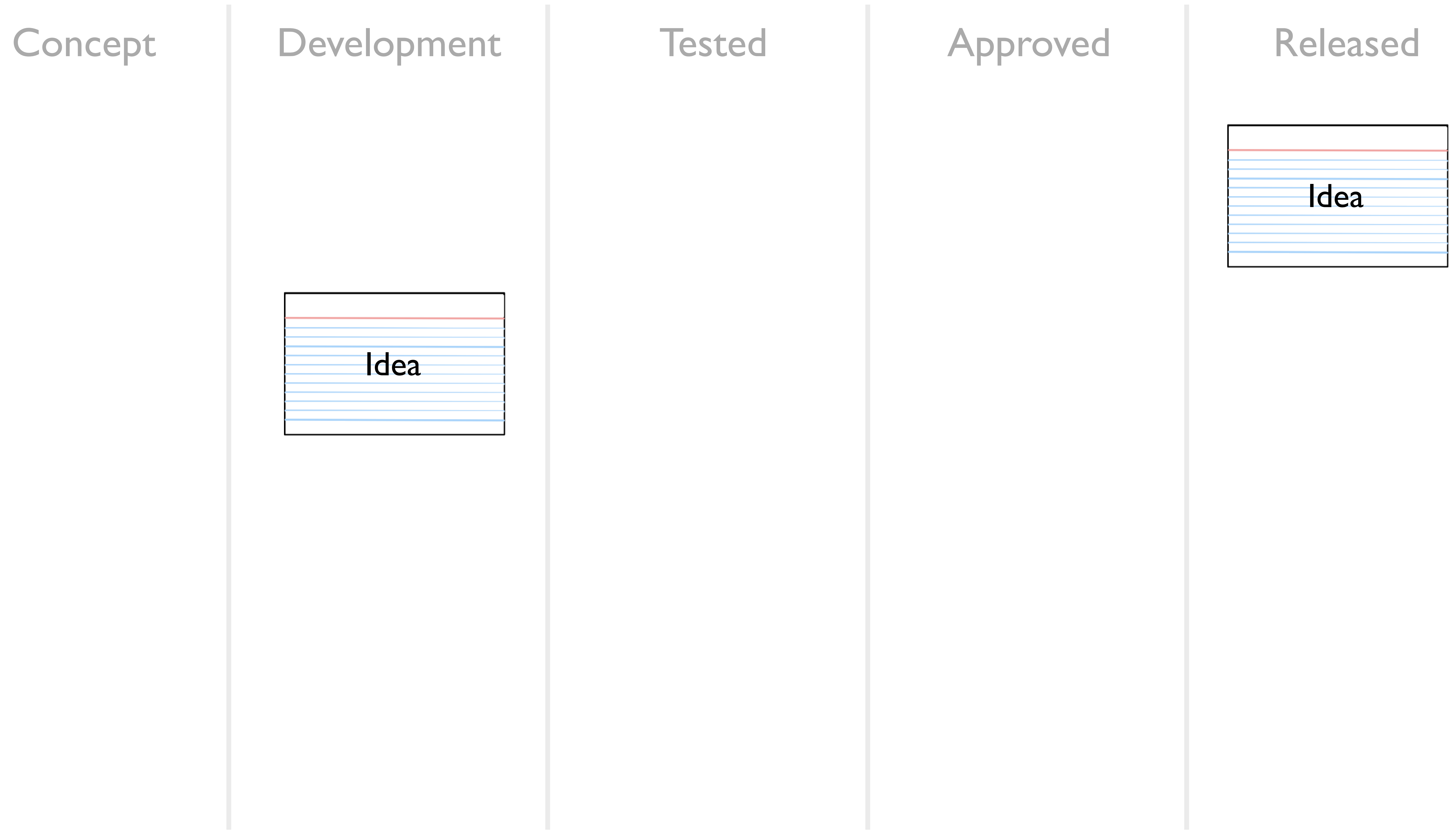
agile software development



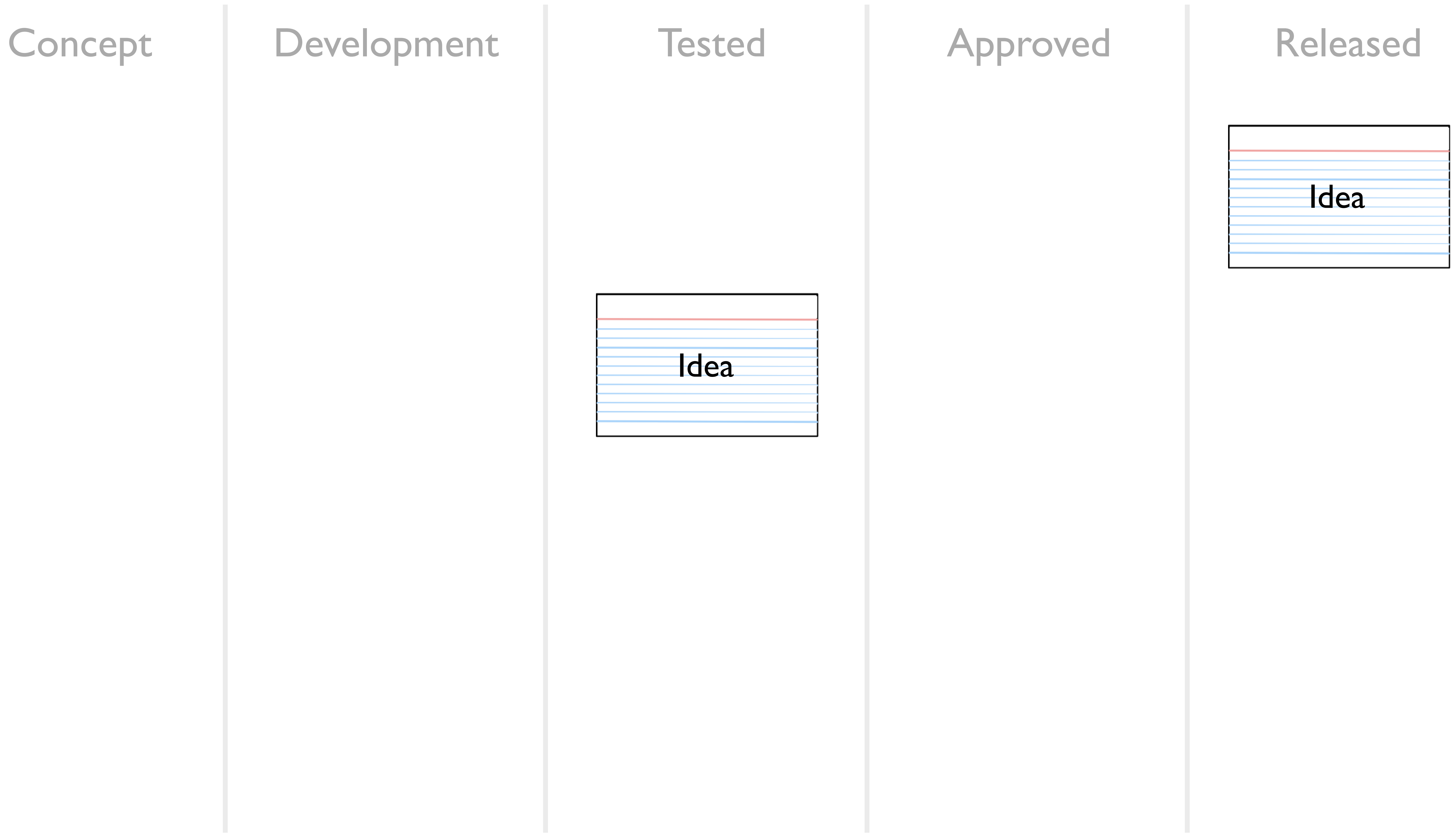
agile software development



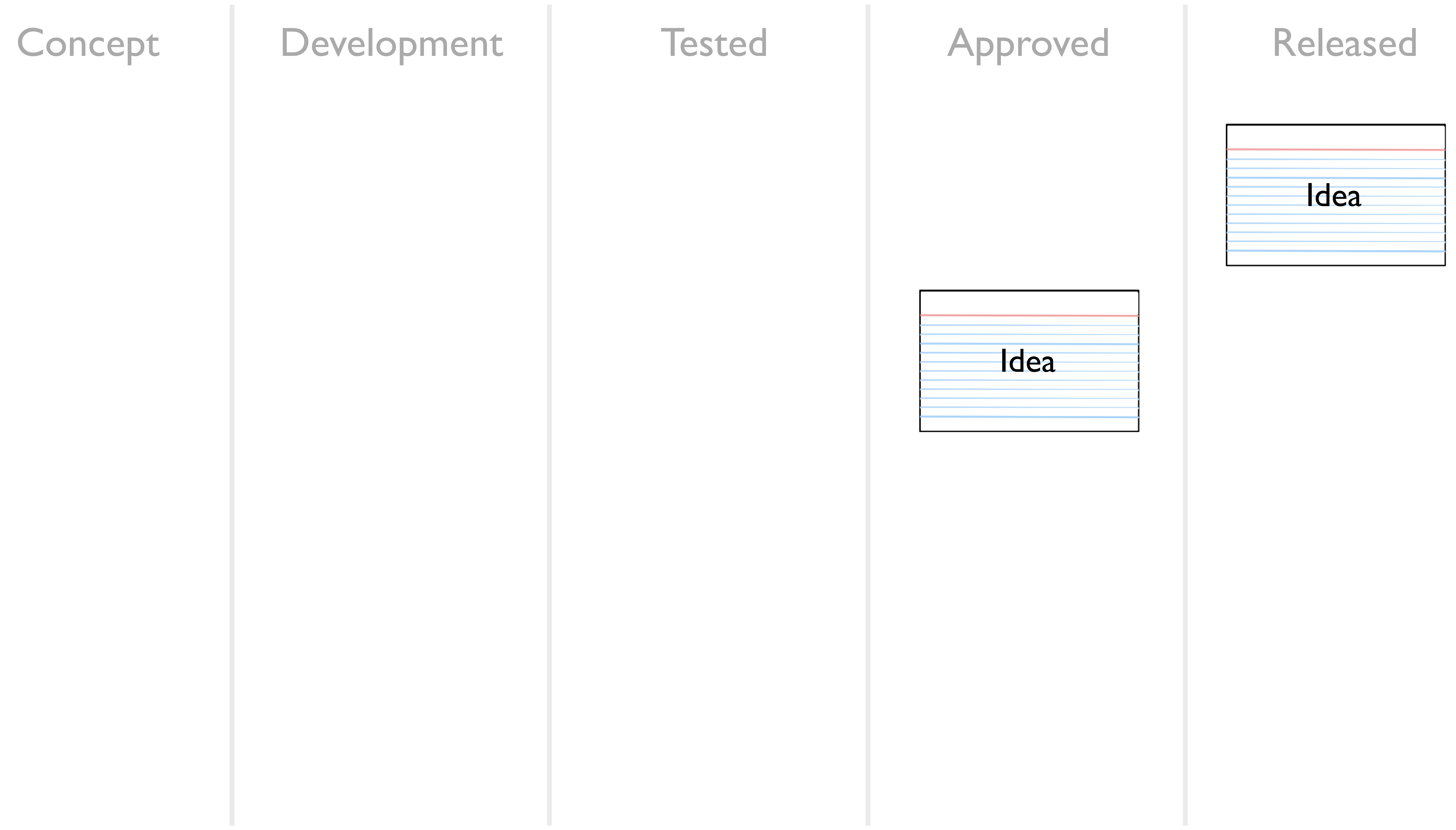
agile software development



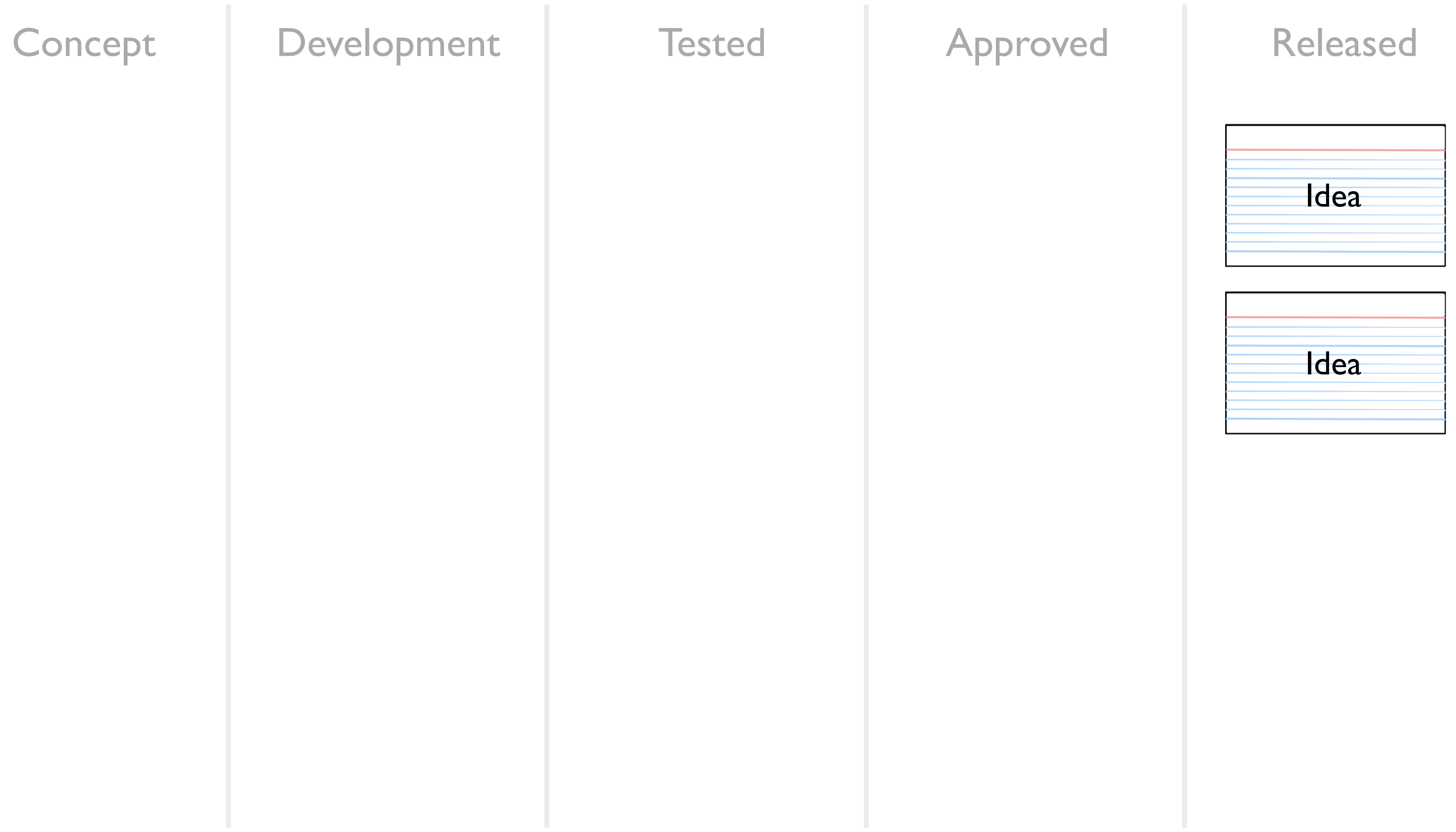
agile software development

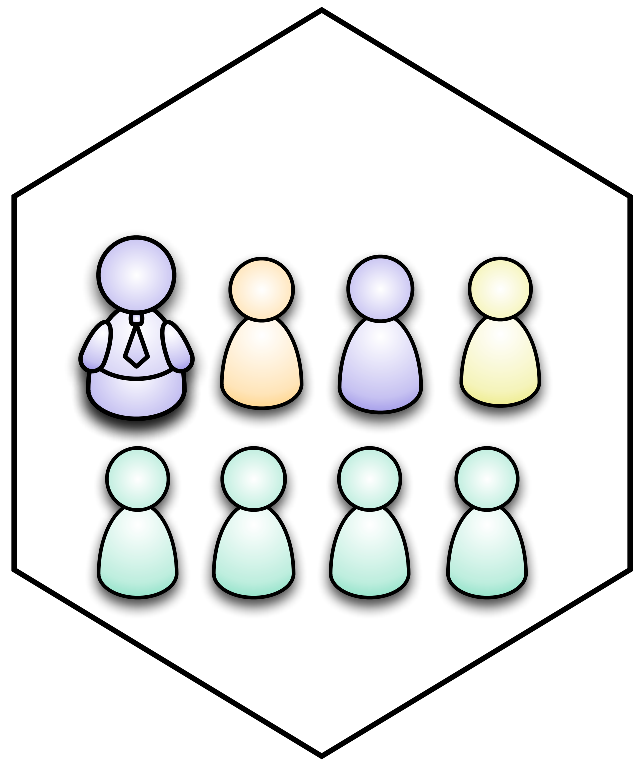


agile software development



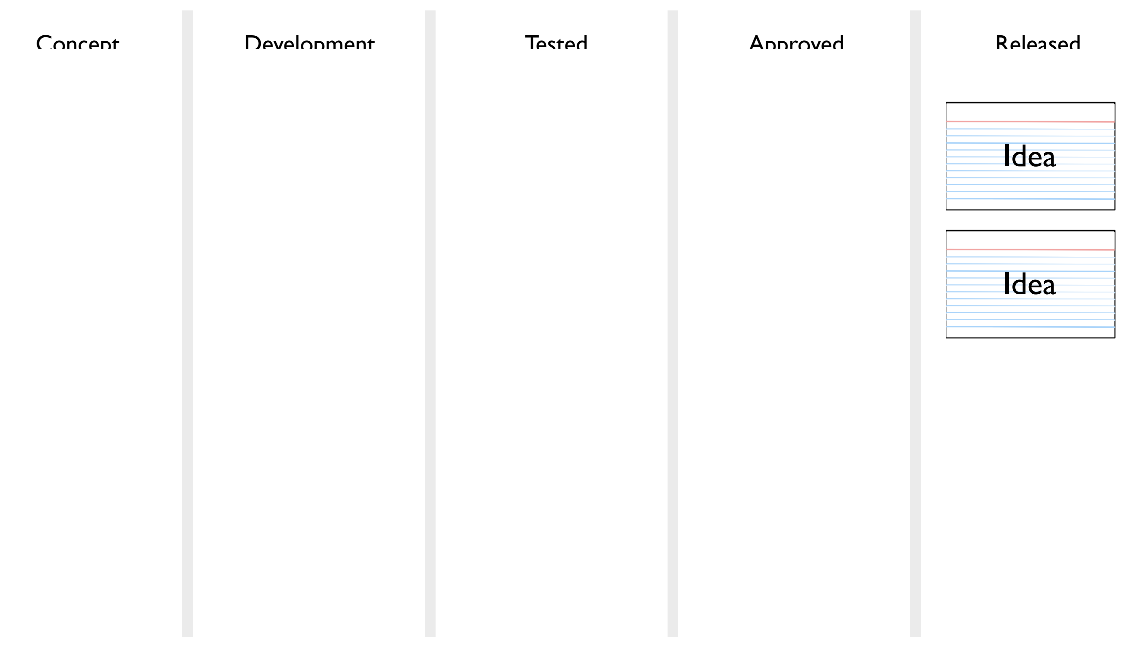
agile software development





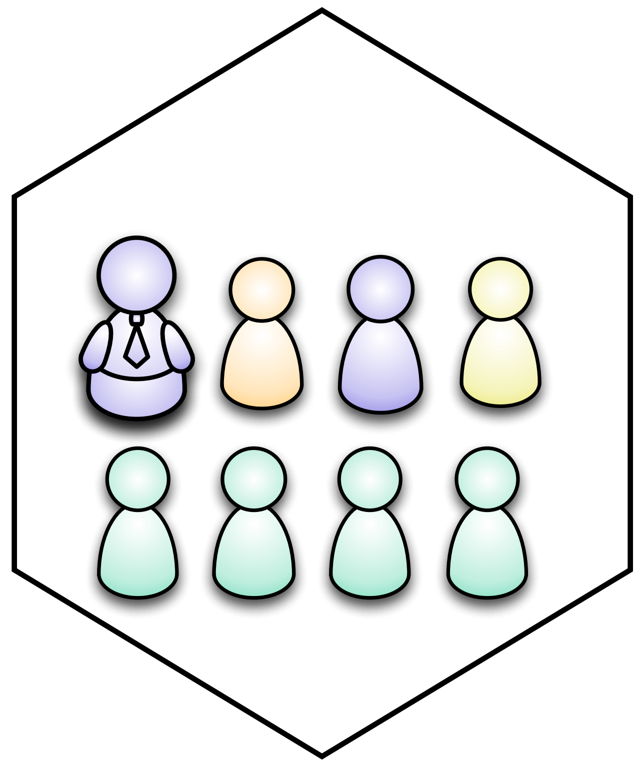
product teams

agile software development

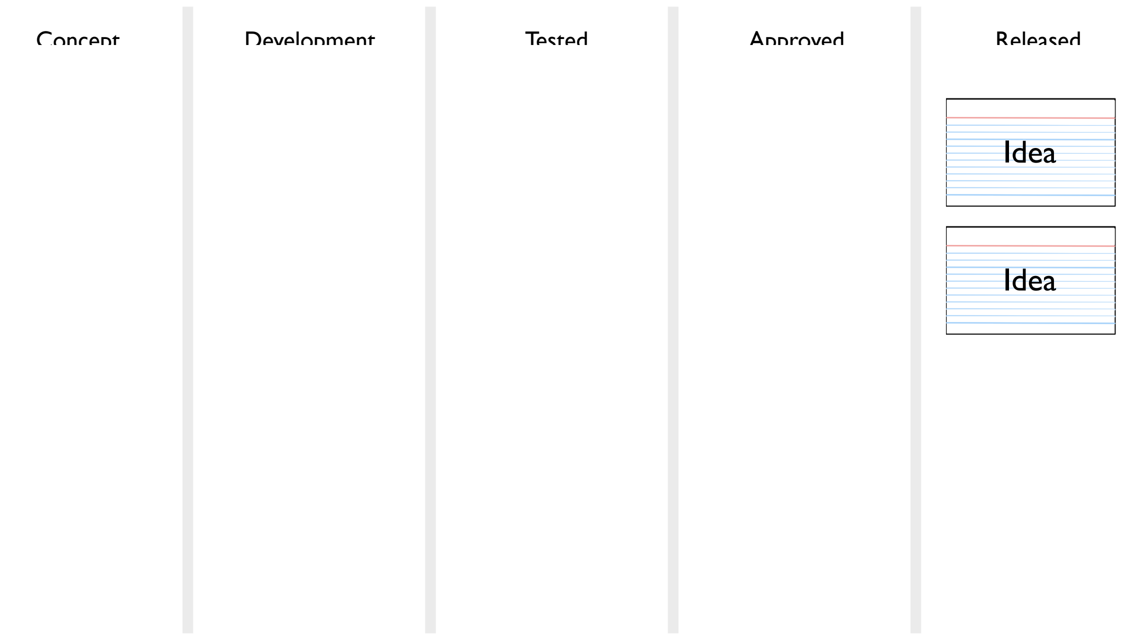


*Start small, low risk to **build trust***

service oriented architecture



product teams

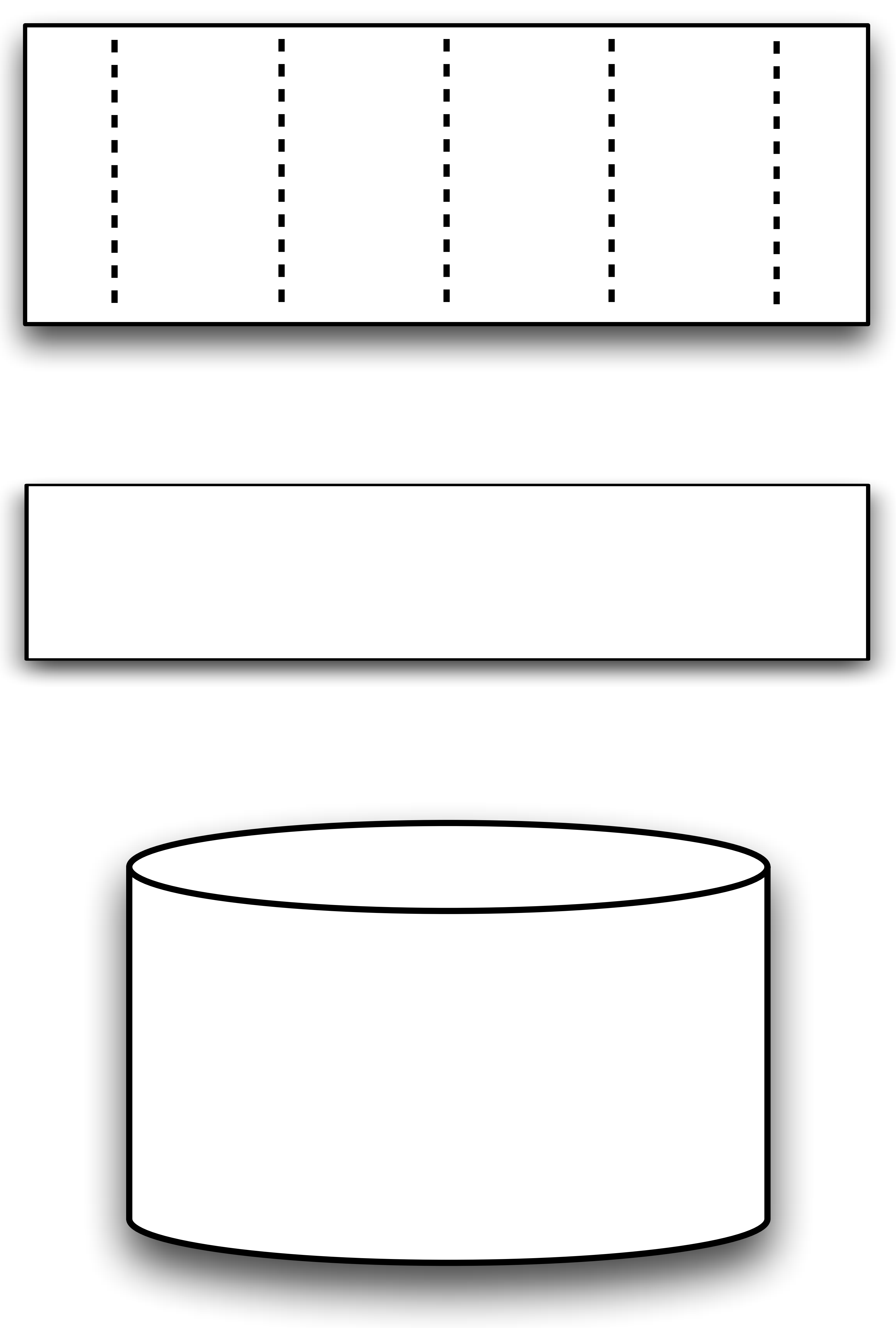


agile software development

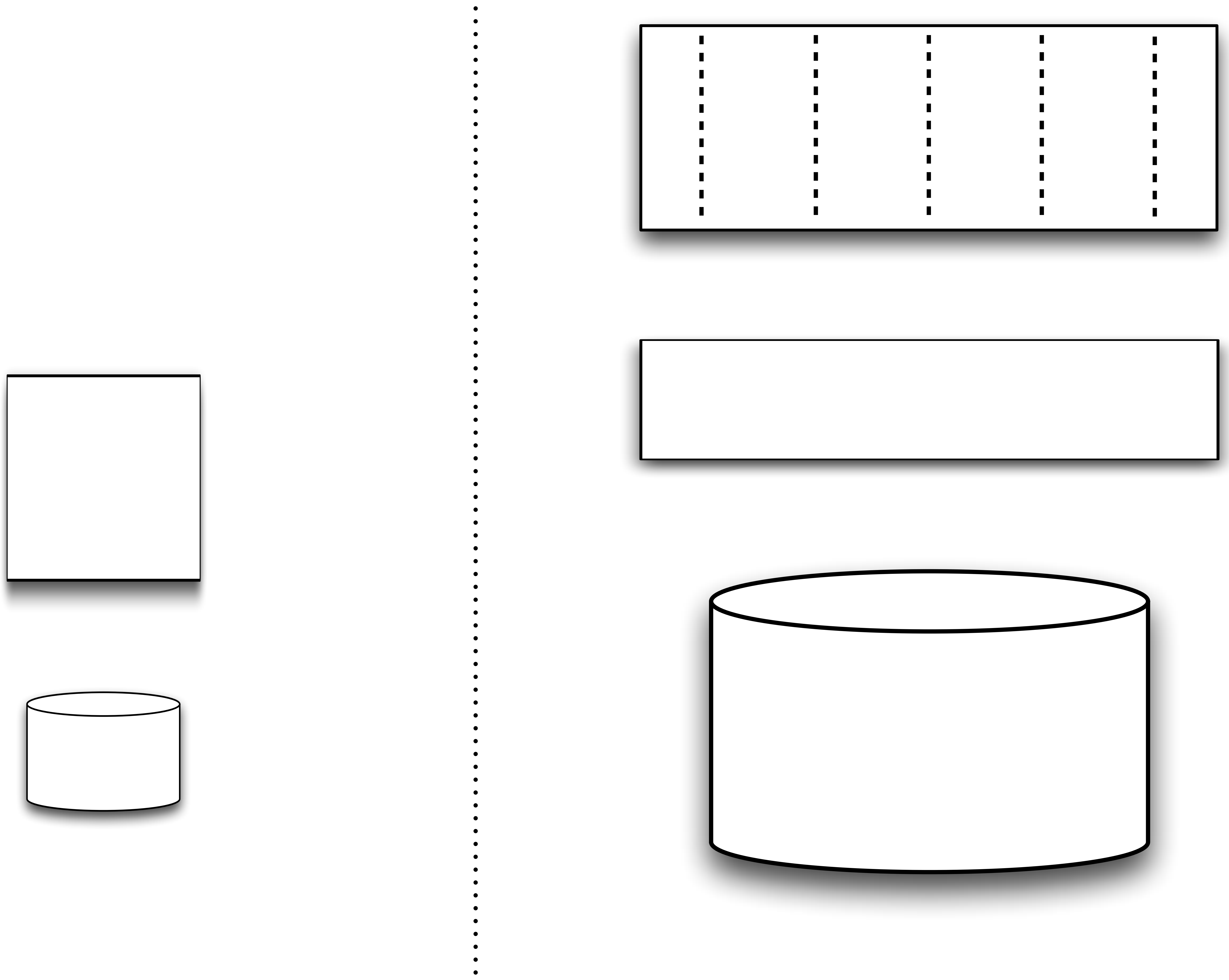
Start small, low risk to build trust

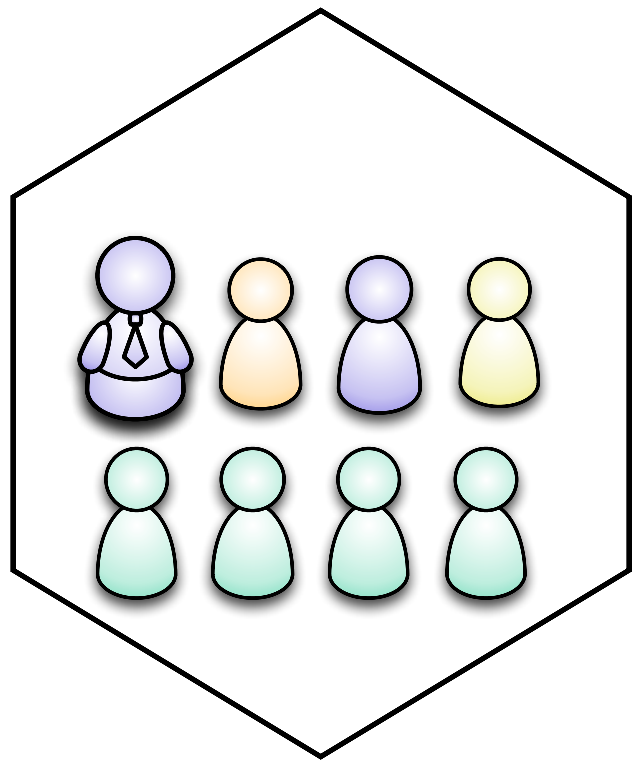
service oriented architecture

Start small, low risk to ***build trust***

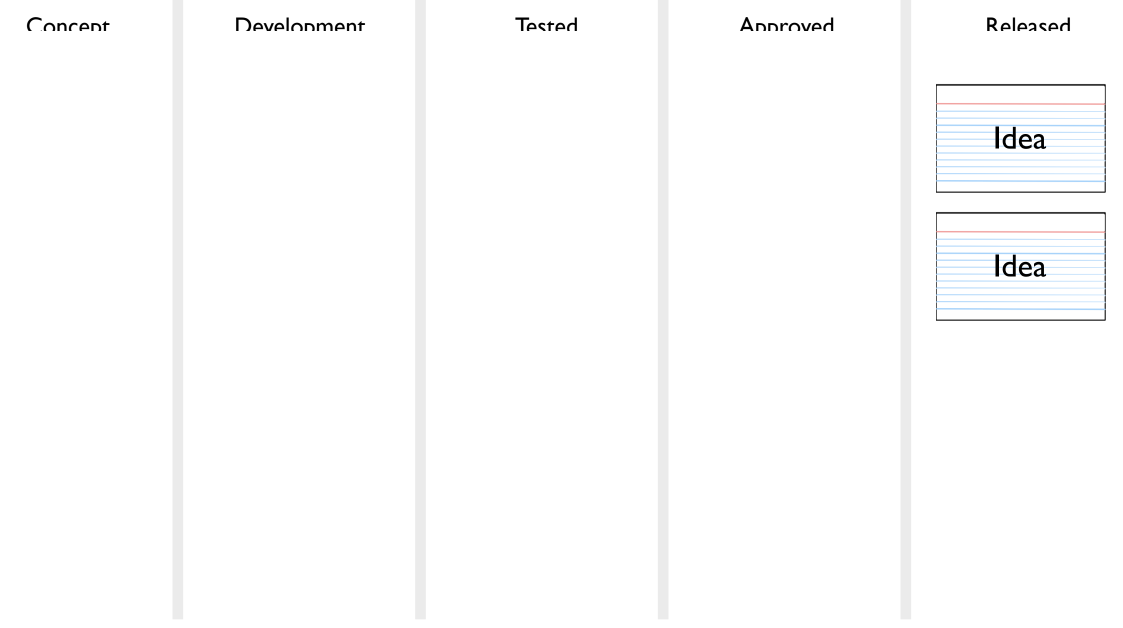


*Start small, low risk to **build trust***

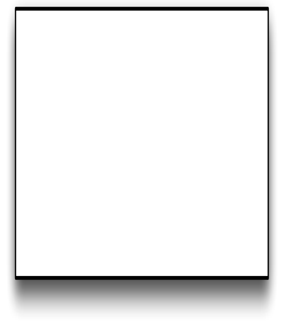




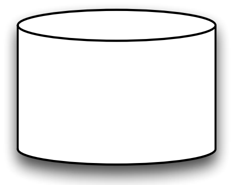
product teams



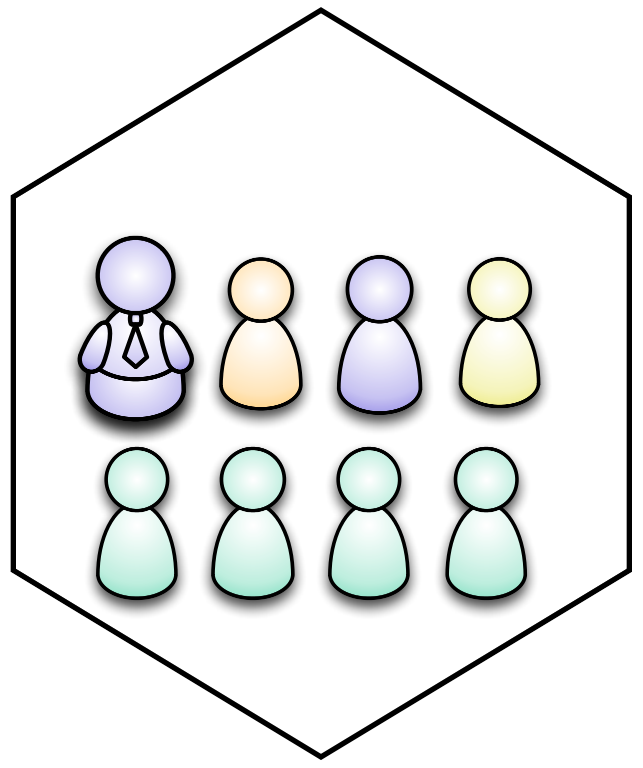
agile software development



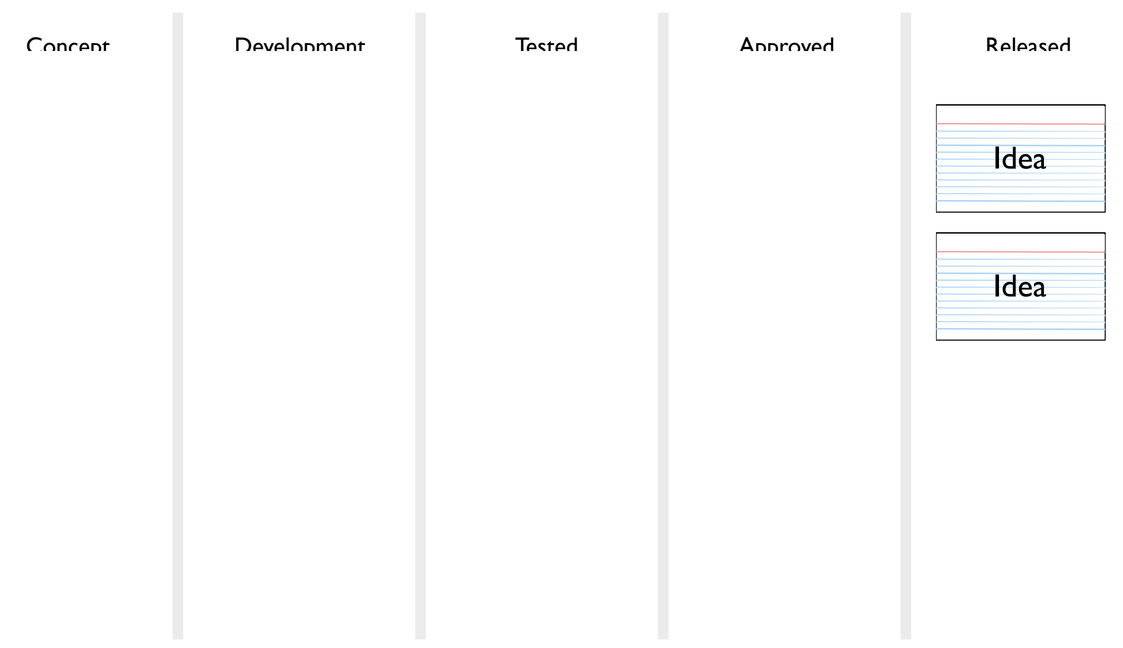
Start small, low risk to build trust



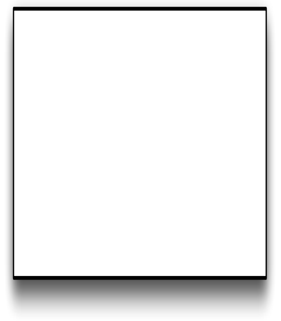
service oriented architecture



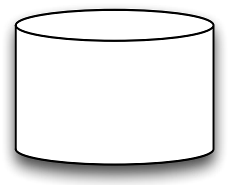
product teams



agile software development

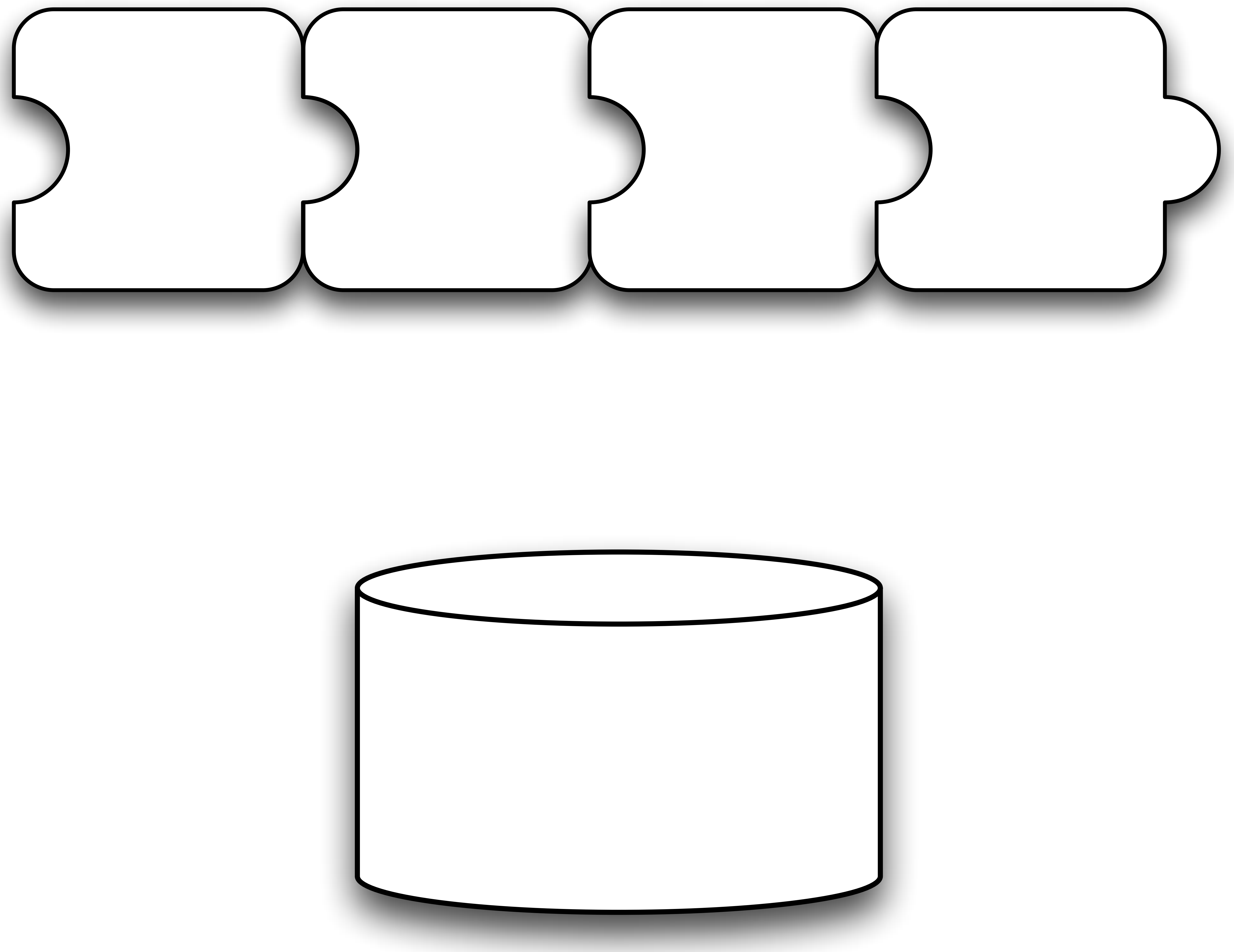


*Start small, low risk to **build trust***

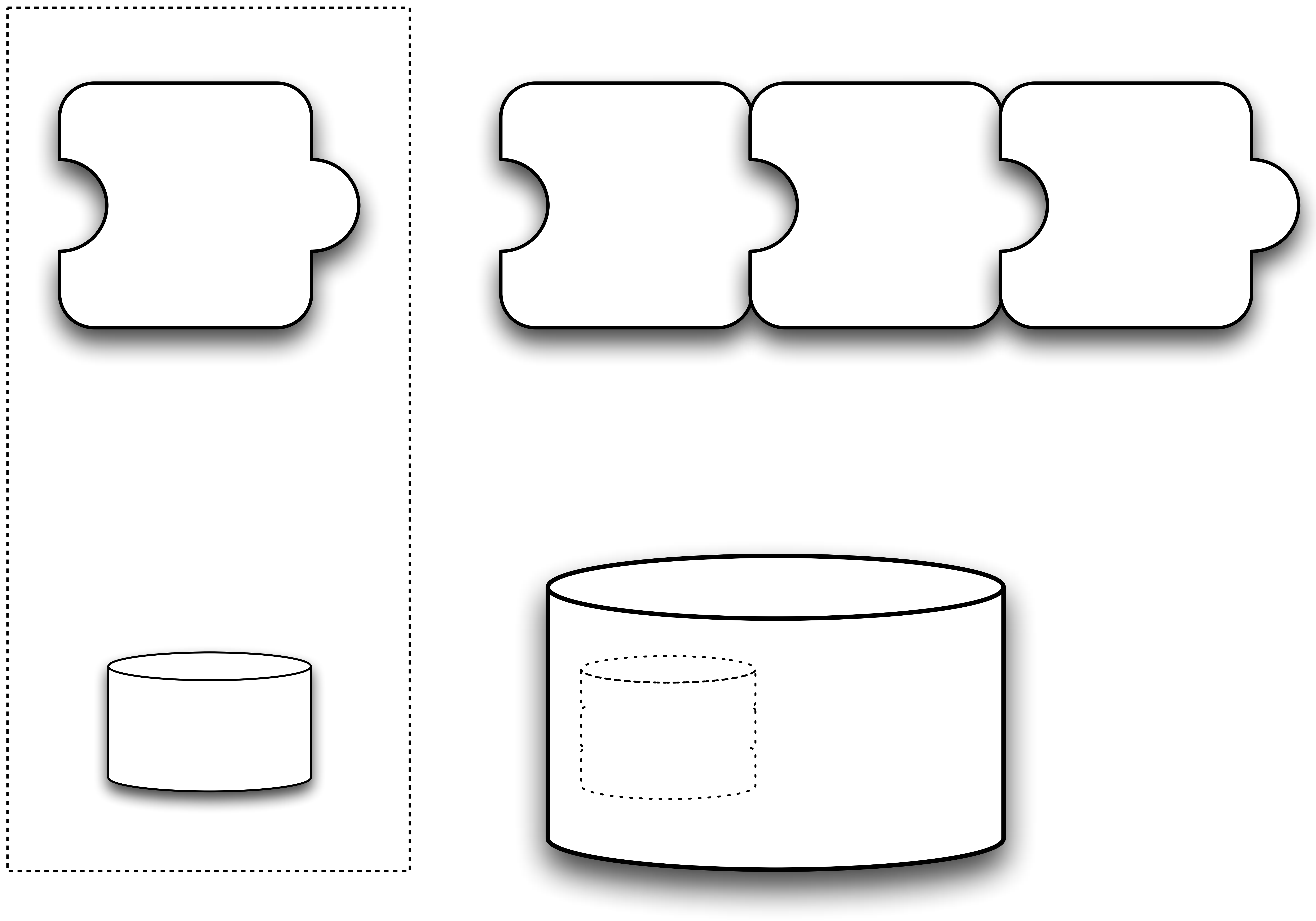


service oriented architecture

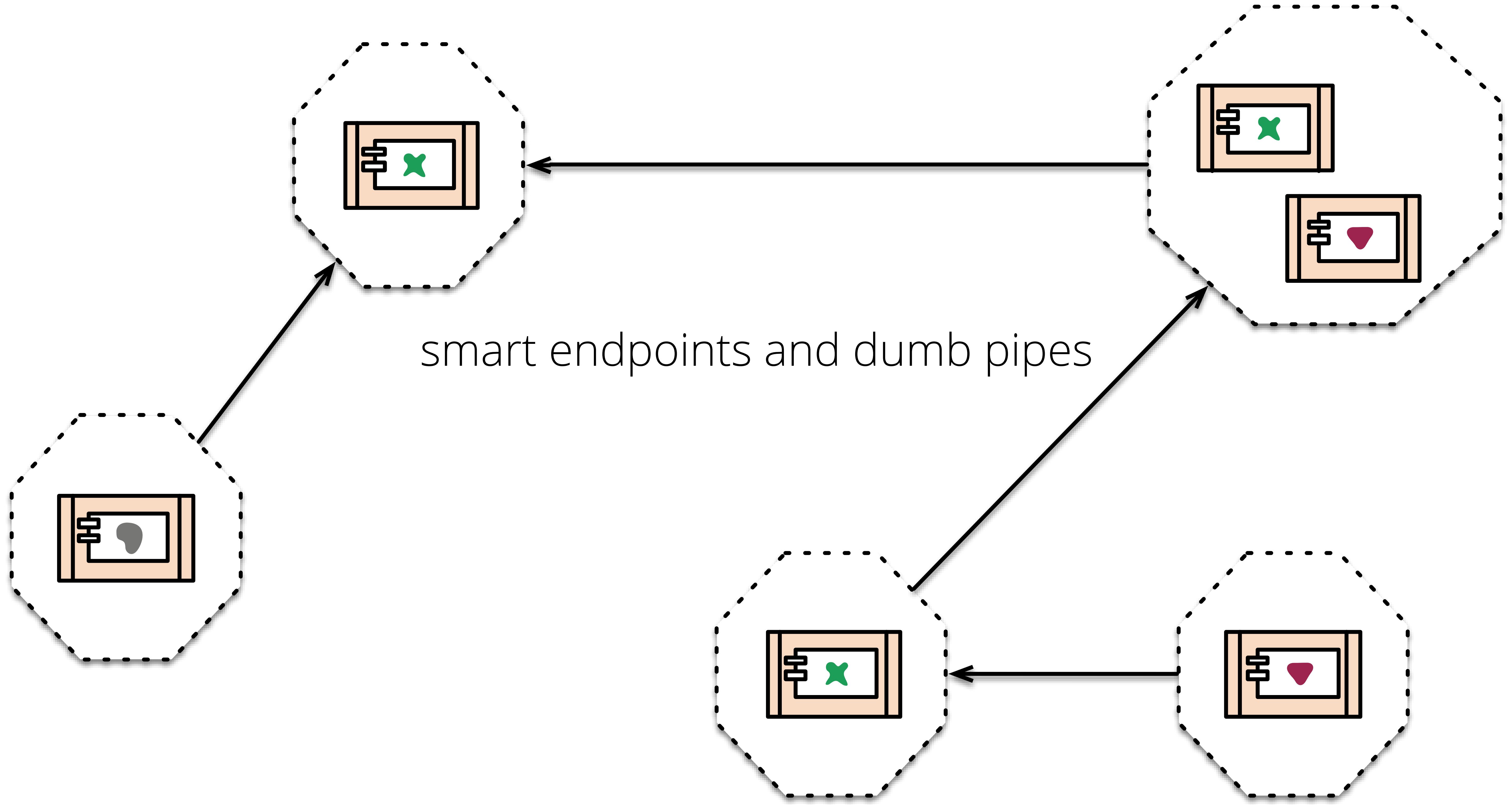
service oriented architecture

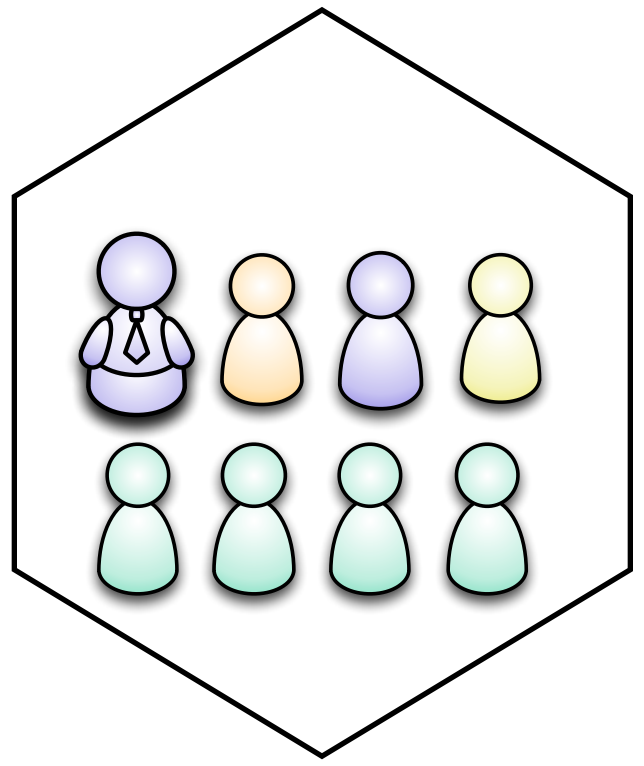


service oriented architecture

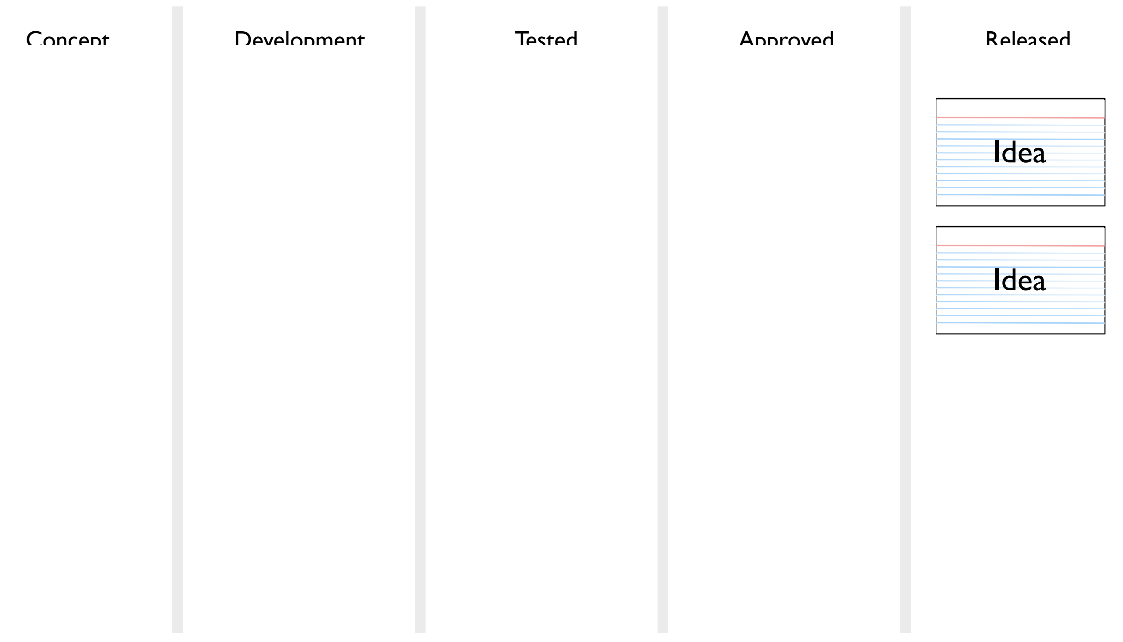


service oriented architecture

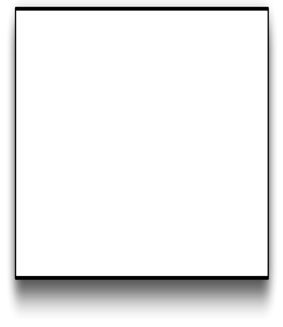




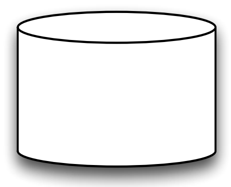
product teams



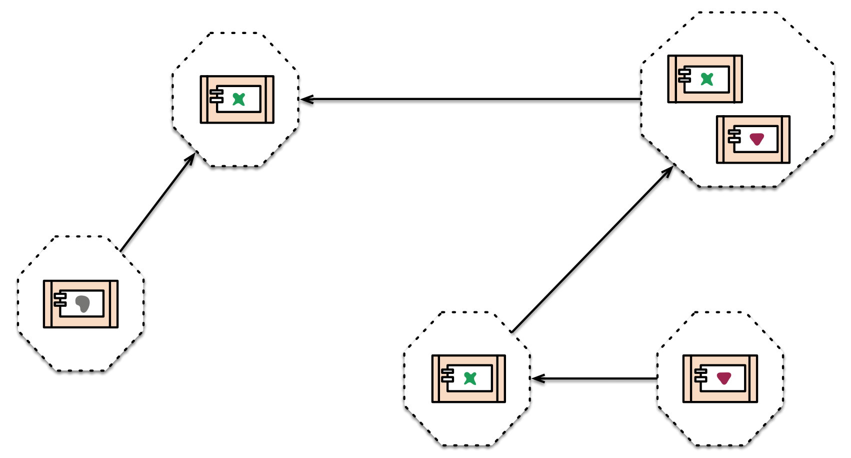
agile software development



Start small, low risk to ***build trust***



service oriented architecture



techniques

product team

PMO

gated development

design up front

wormhole systems

TDD

agile

project branches

silo'd functions

tools

powershell

visual studio

sqlserver

odd entity thing

TFS

CLR

windows

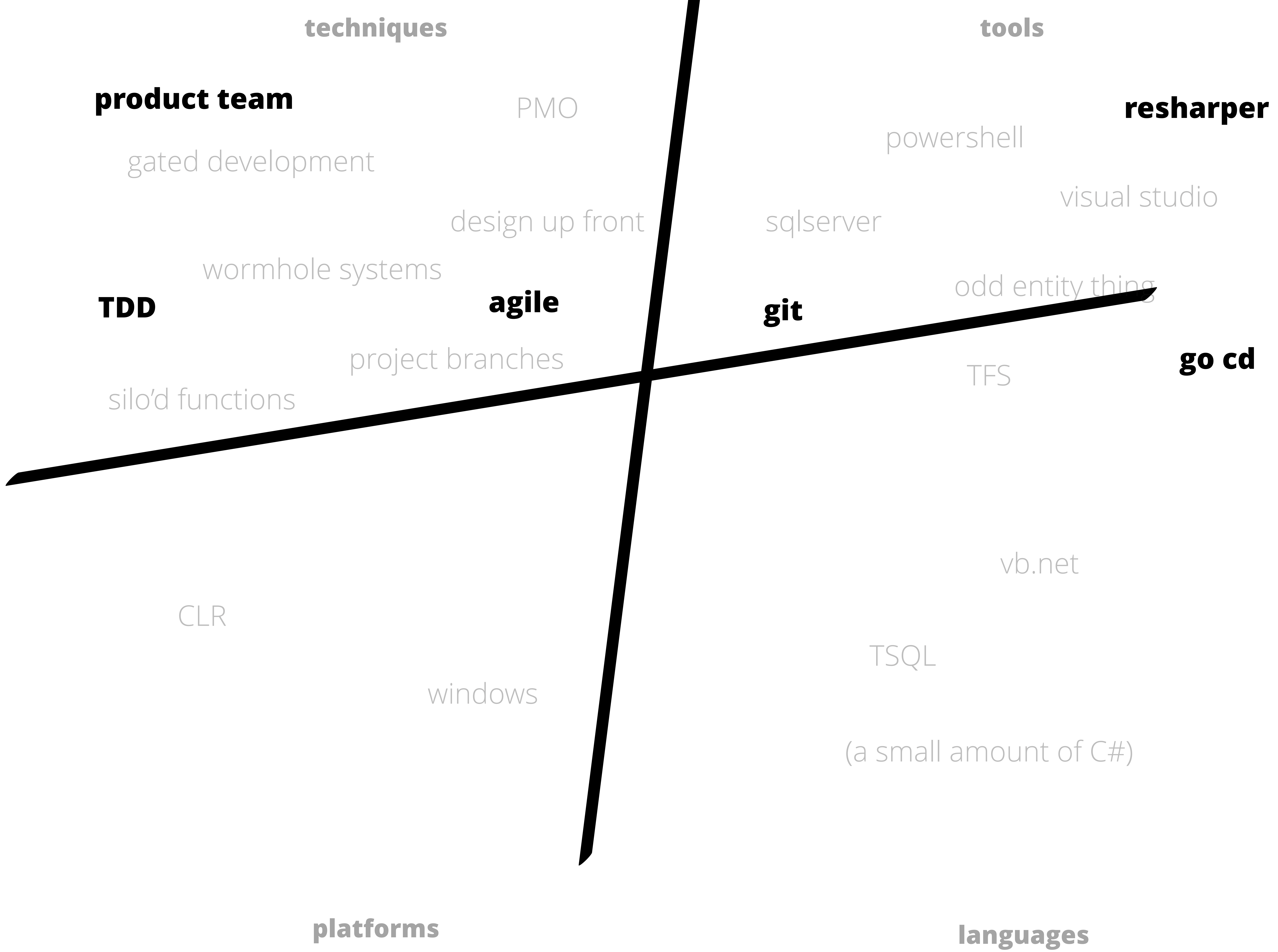
platforms

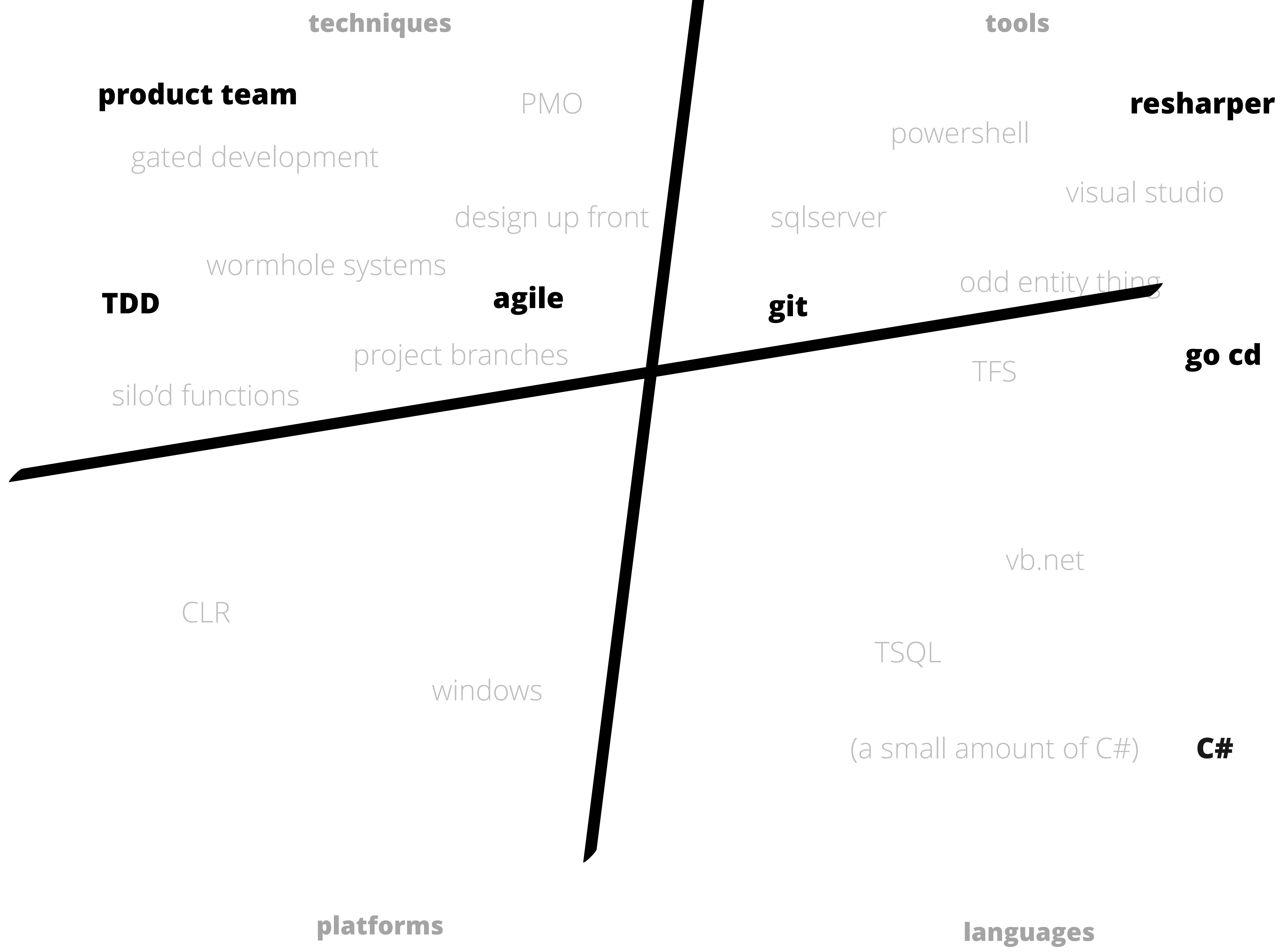
vb.net

TSQL

(a small amount of c#)

languages


techniques
tools
platforms
languages
product team
PMO
gated development
design up front
wormhole systems
TDD
agile
project branches
silo'd functions
resharper
powershell
visual studio
sqlserver
odd entity thing
git
go cd
TFS
CLR
windows
vb.net
TSQL
(a small amount of C#)


techniques
tools
product team
PMO
resharper
powershell
gated development
visual studio
design up front
sqlserver
wormhole systems
odd entity thing
TDD
agile
git
go cd
project branches
TFS
silo'd functions
vb.net
CLR
TSQL
windows
(a small amount of C#)
C#
platforms
languages

separate products

conway’s law

microservices

event sourcing

monitoring

right tool for the job

separate products

conway’s law

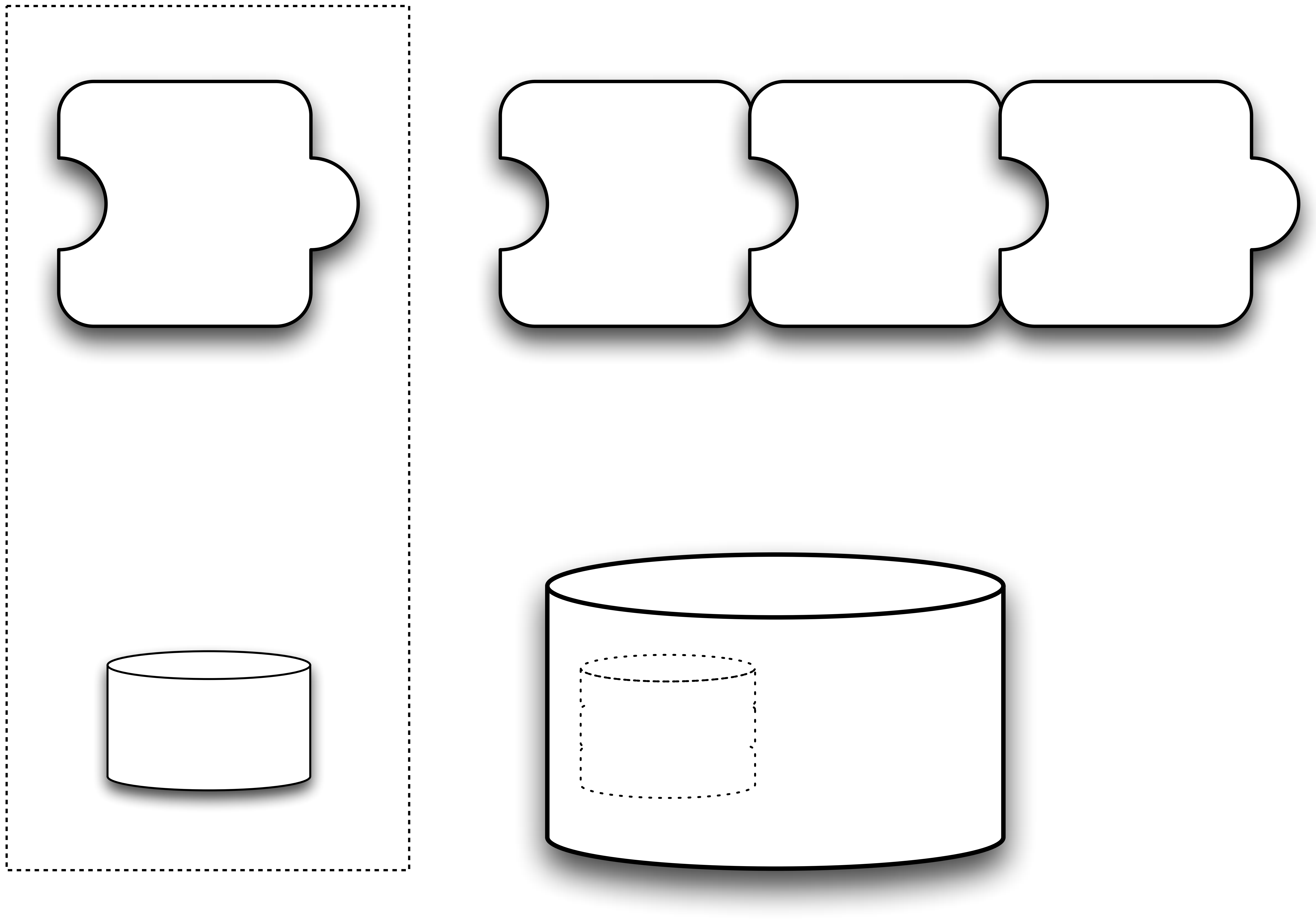
microservices

event sourcing

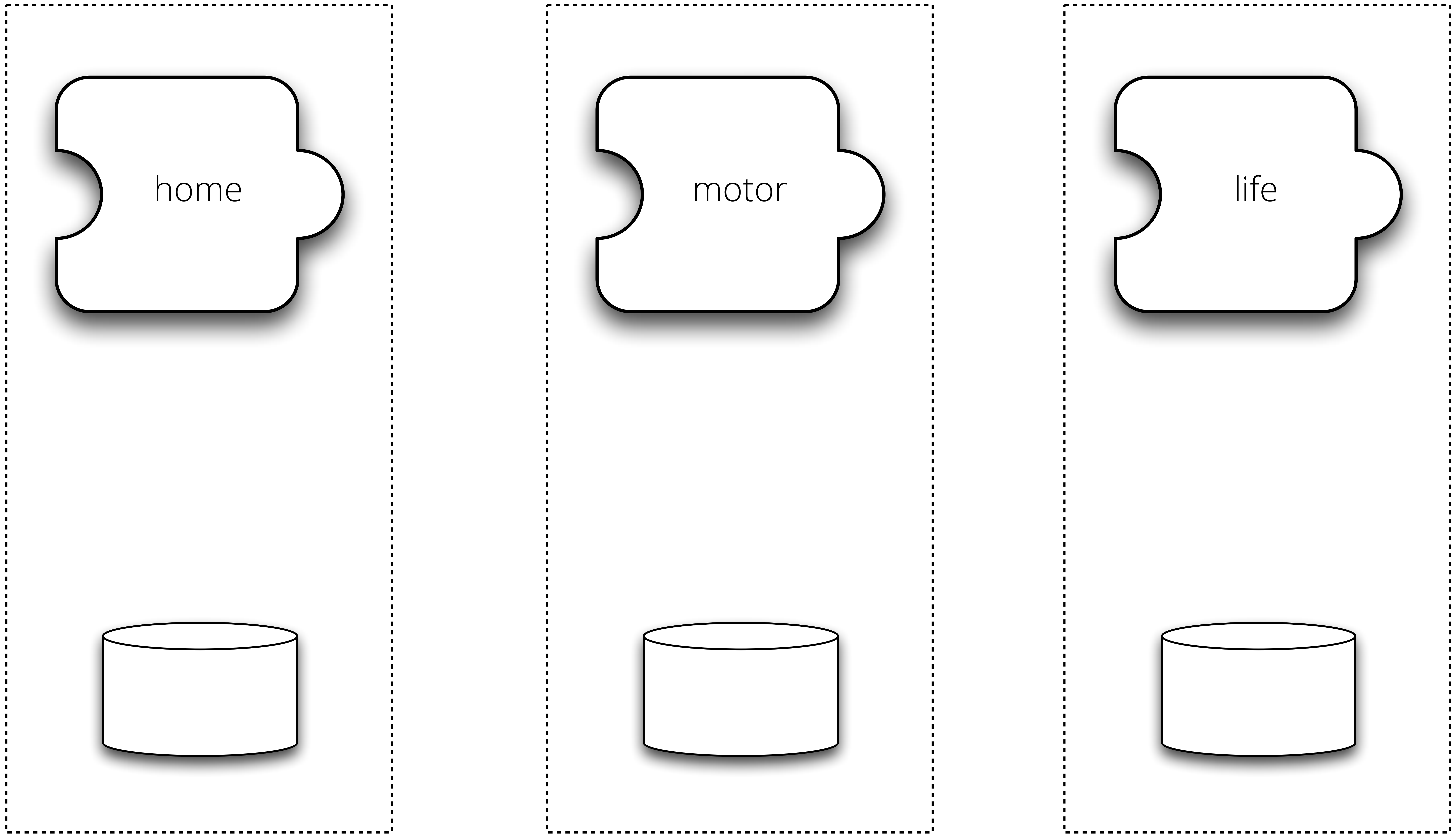
monitoring

right tool for the job

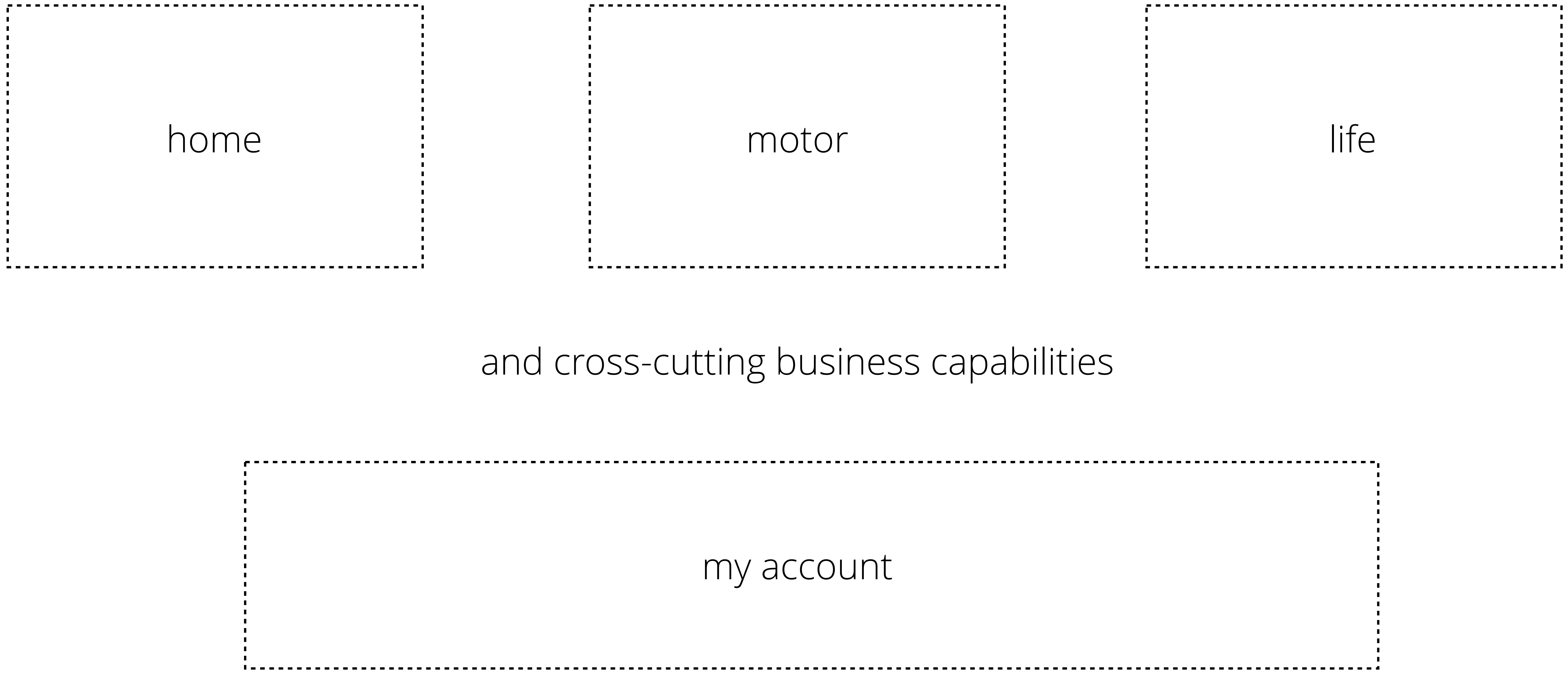
separate products

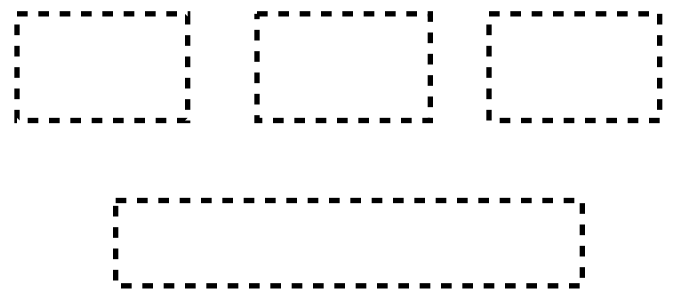


separate products



separate products





separate products

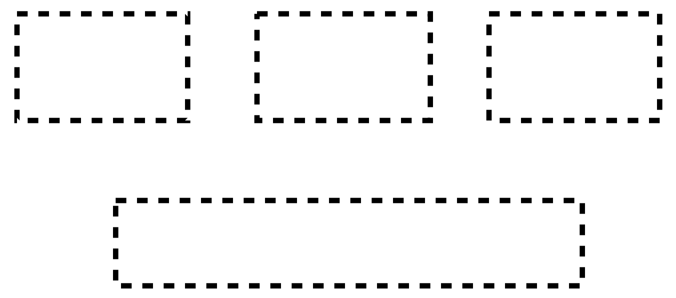
conway's law

microservices

event sourcing

monitoring

right tool for the job



separate products

conway’s law

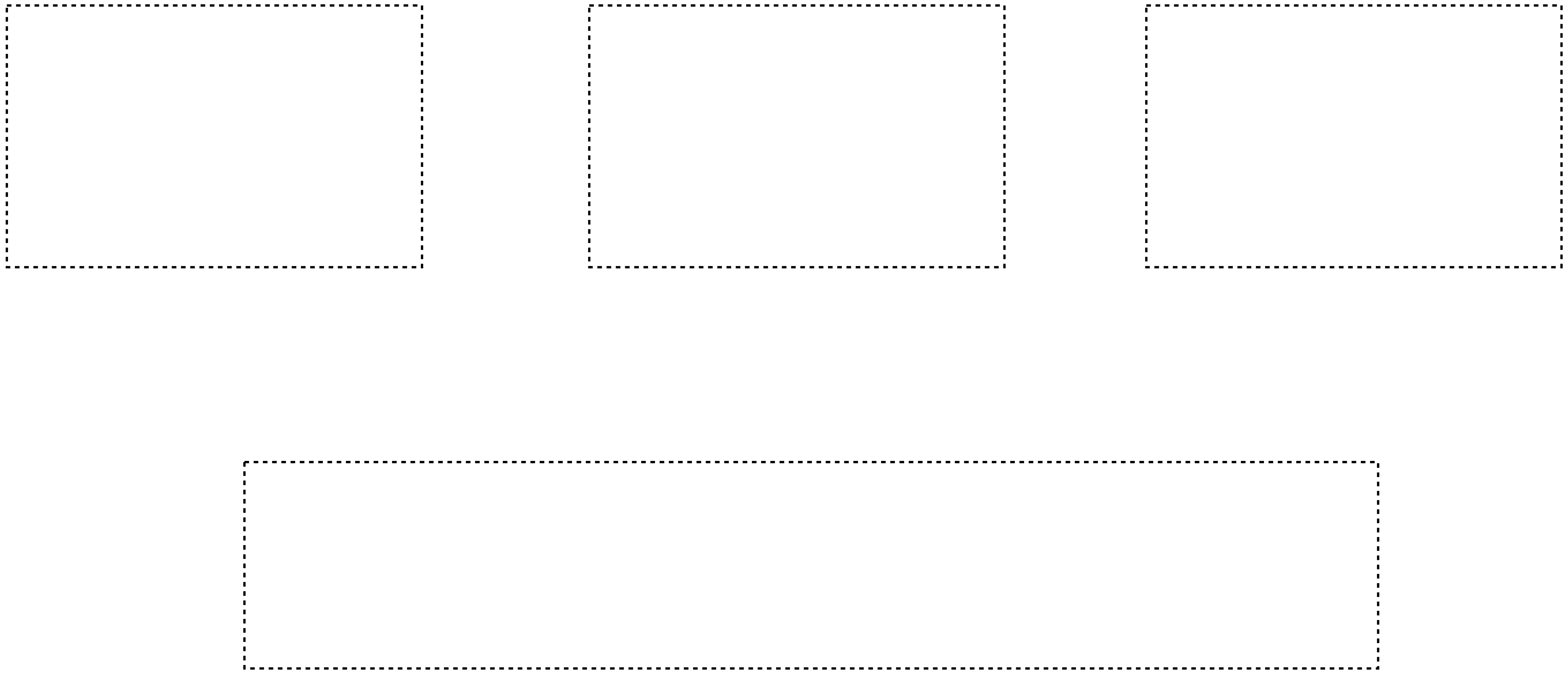
microservices

event sourcing

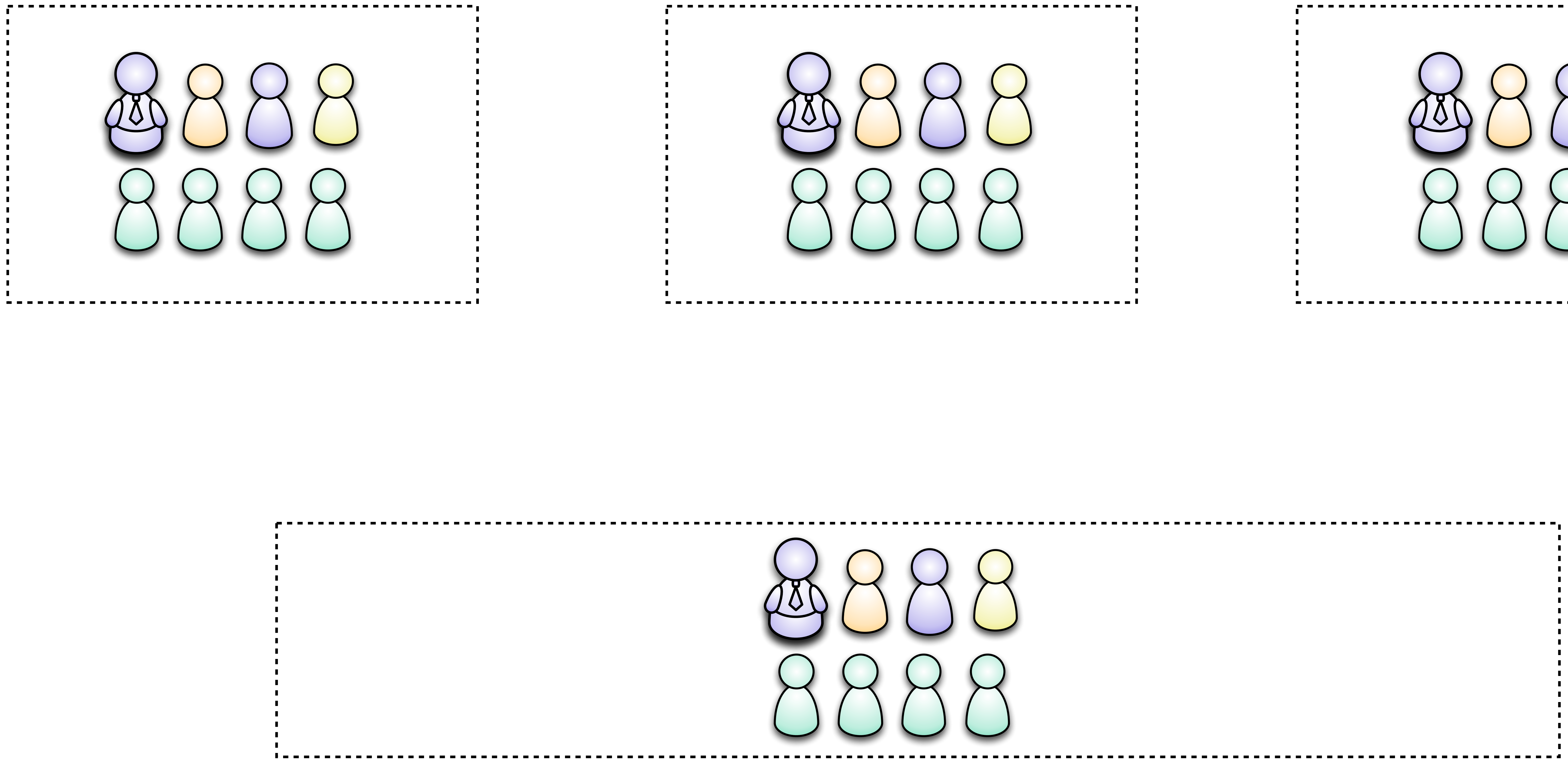
monitoring

right tool for the job

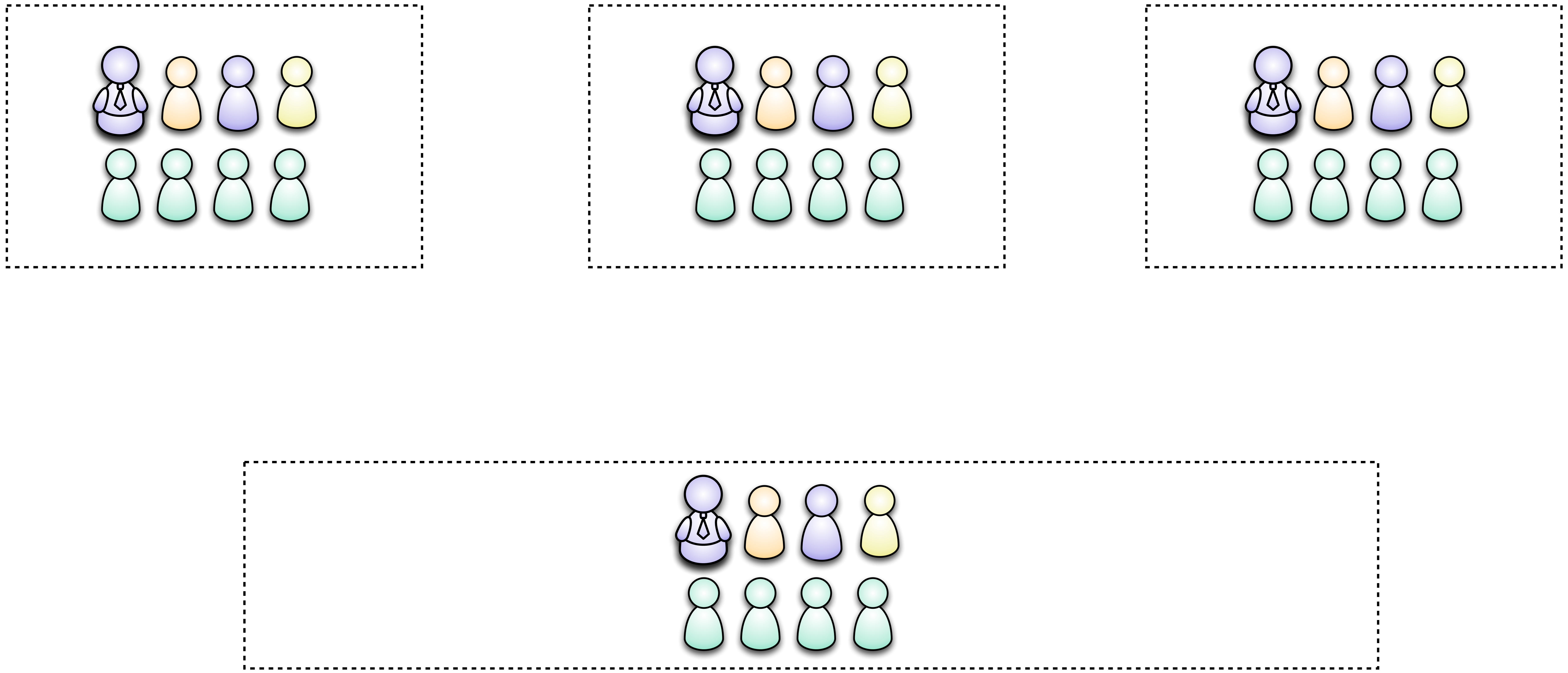
conway's law



conway's law



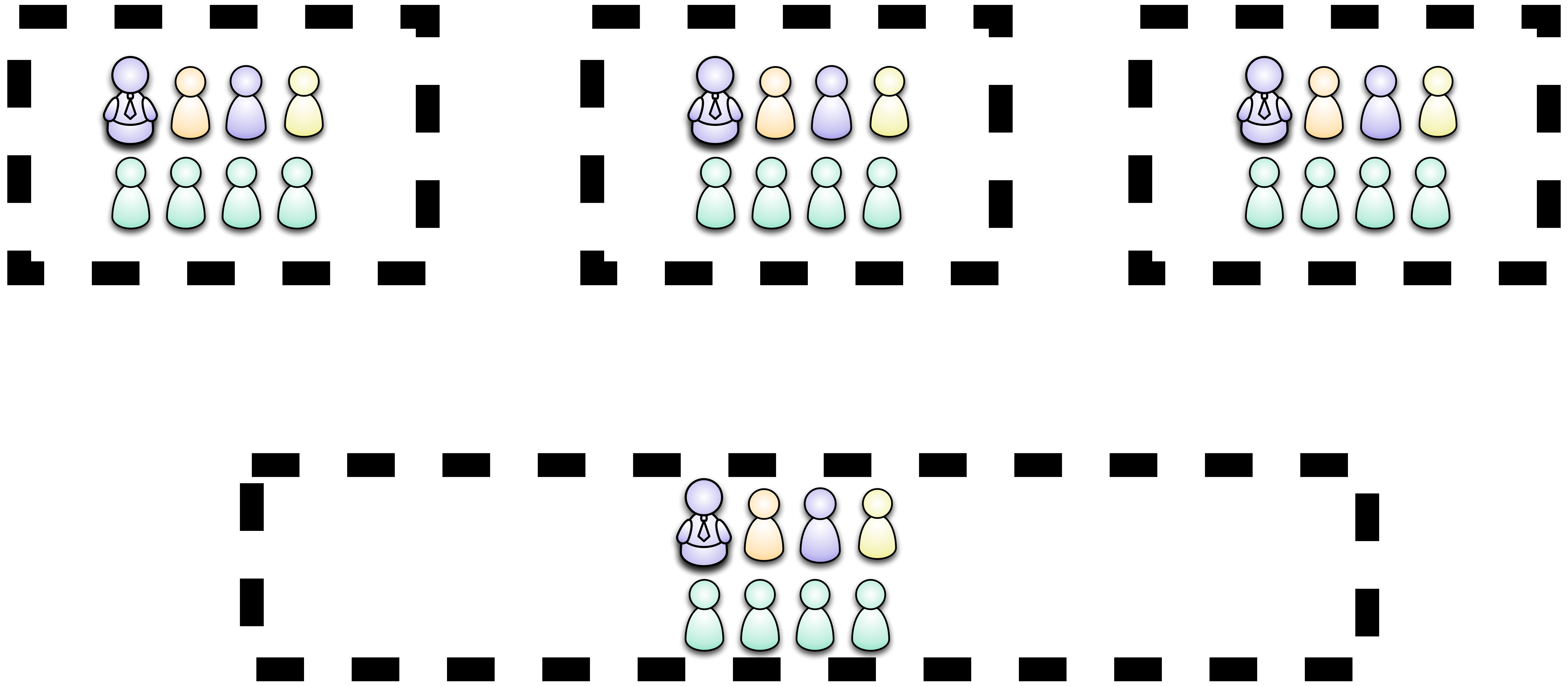
conway's law

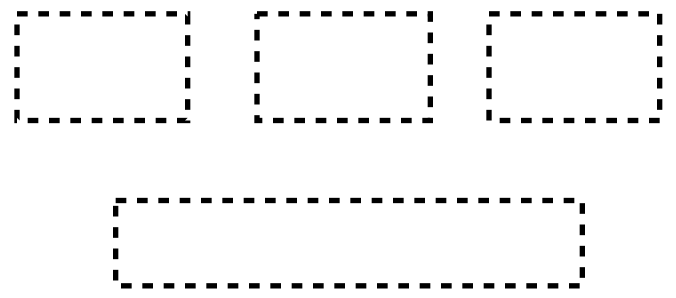


“...organizations which design systems ... are constrained to produce designs which are ***copies of the communication structure*** of those organizations”

Melvin Conway, 1968

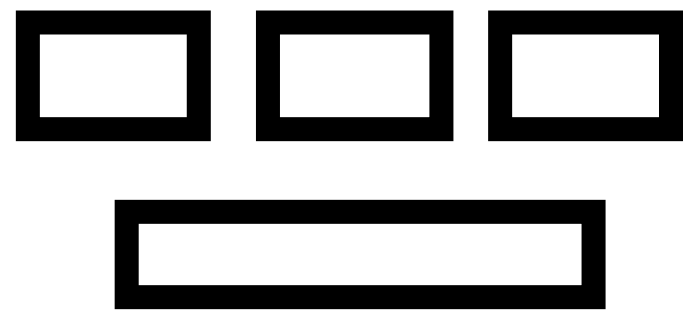
conway's law





separate products

conway's law



microservices

event sourcing

monitoring

right tool for the job

separate products

conway’s law

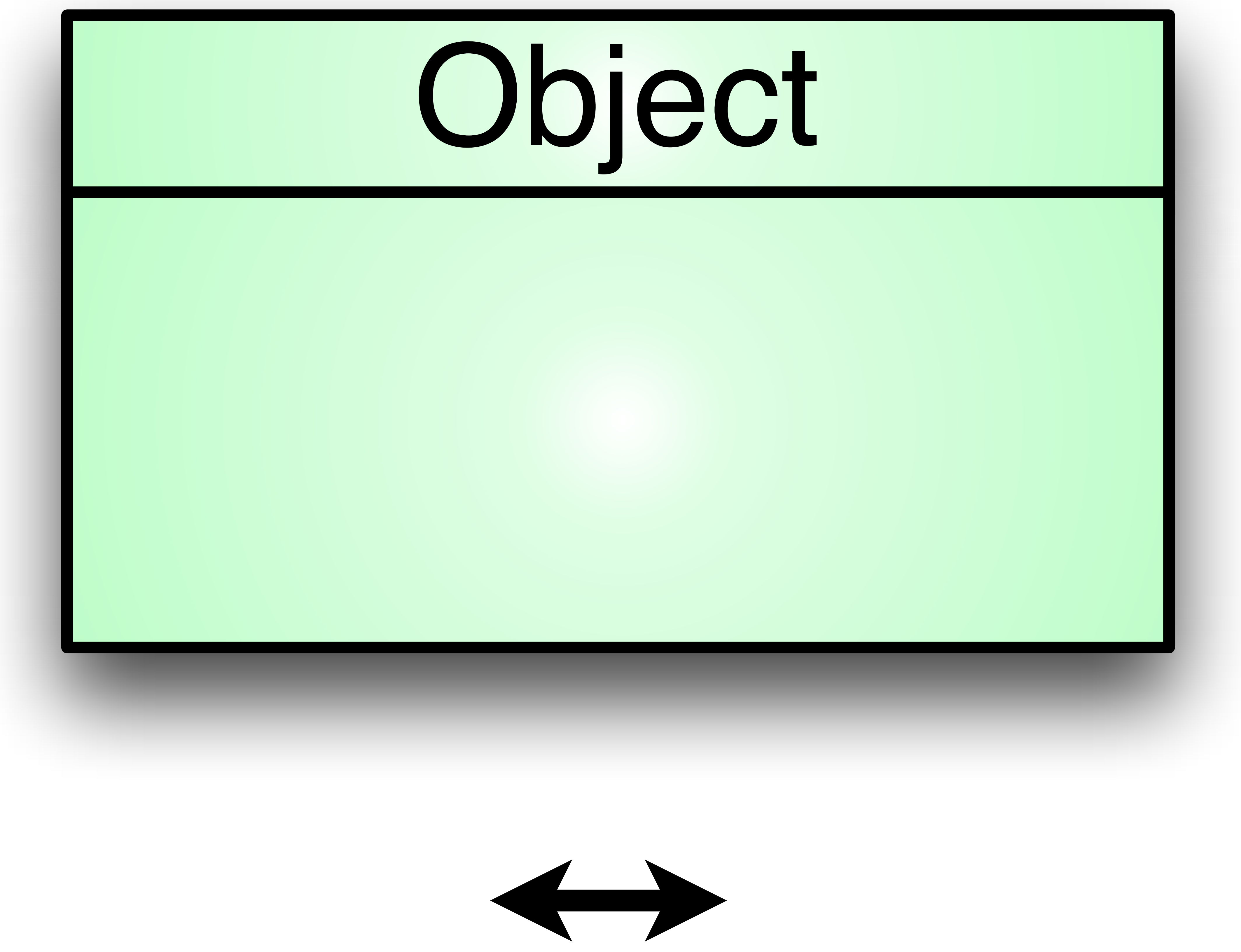
microservices

event sourcing

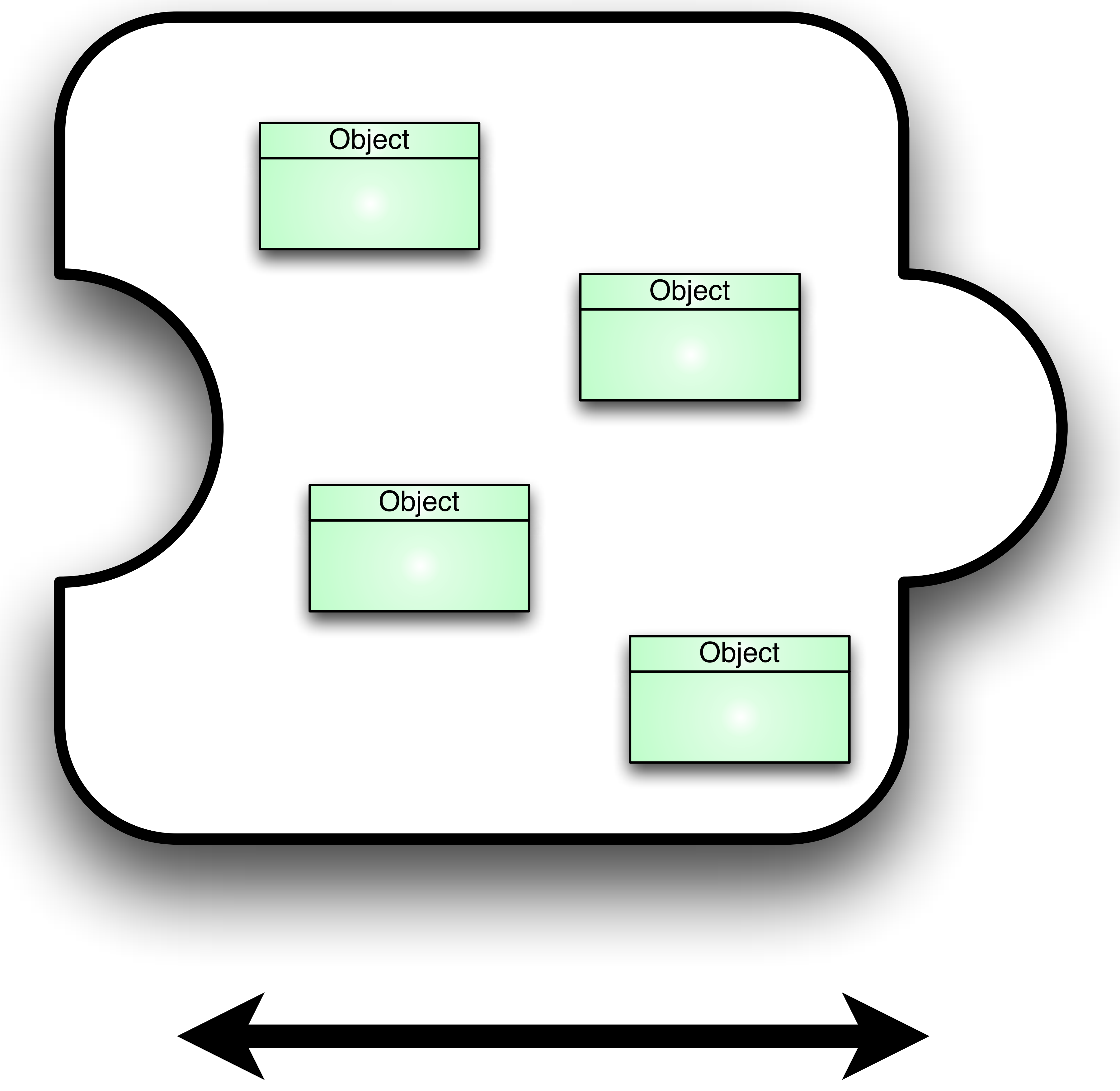
monitoring

right tool for the job

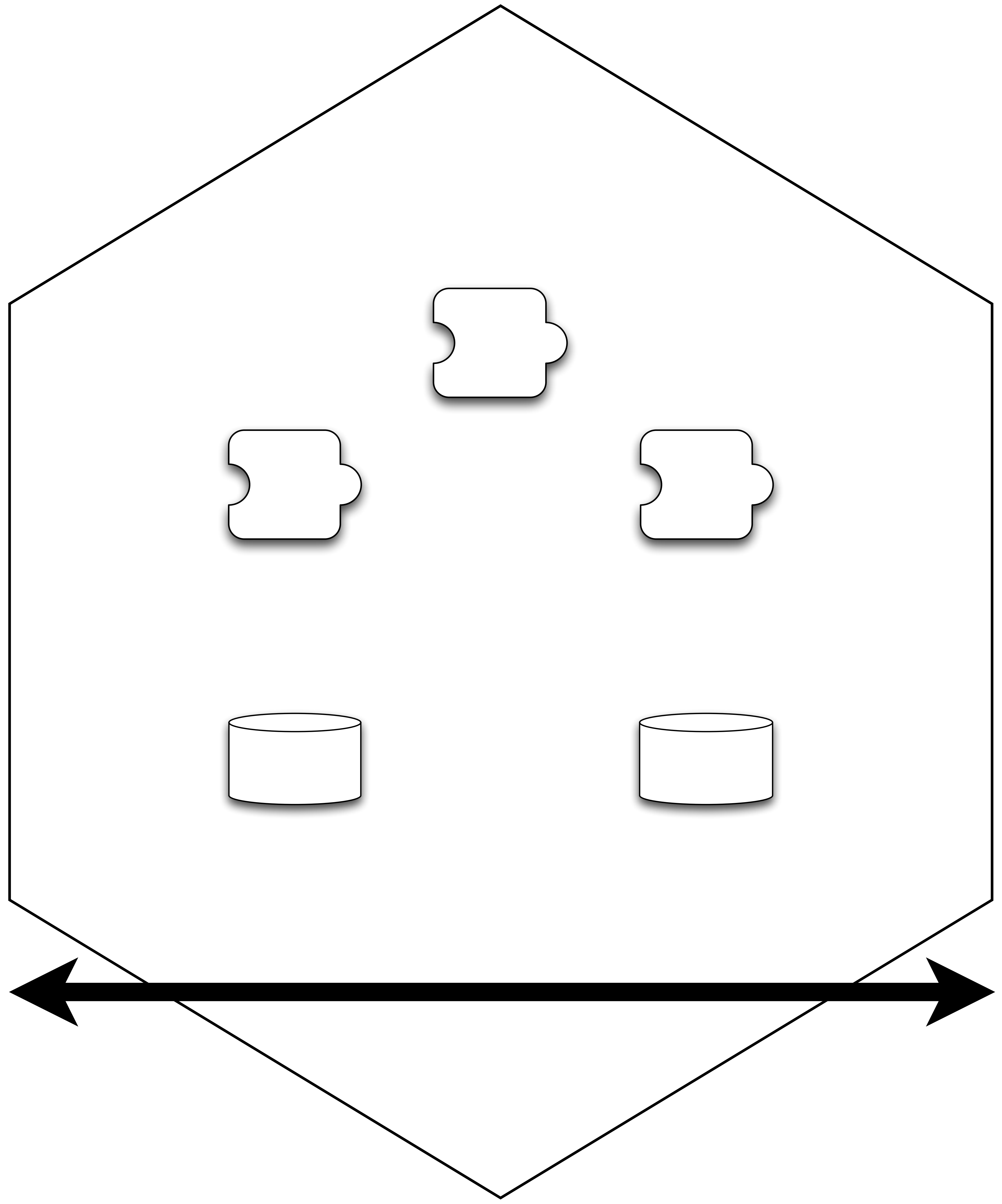
microservices



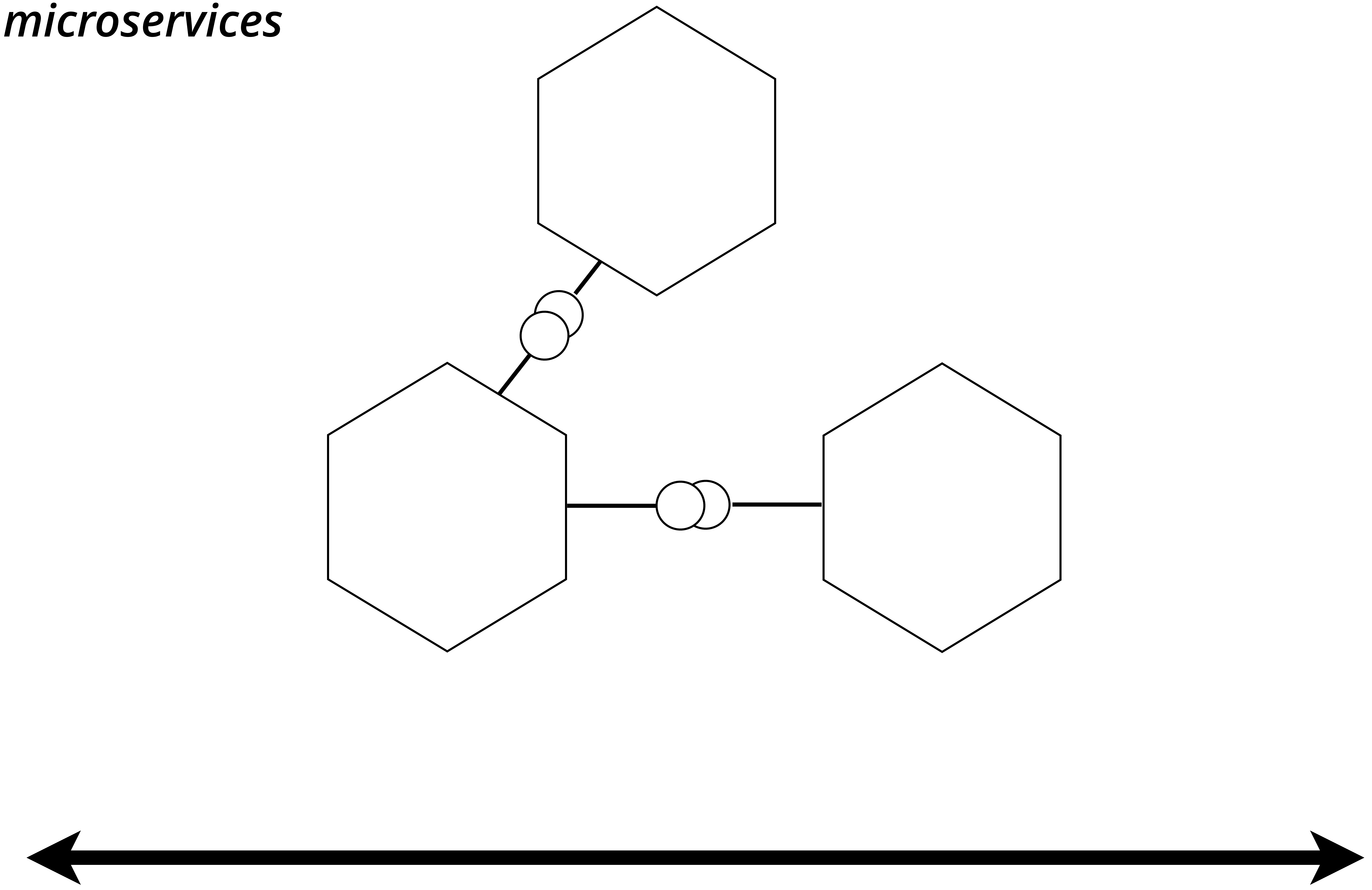
microservices



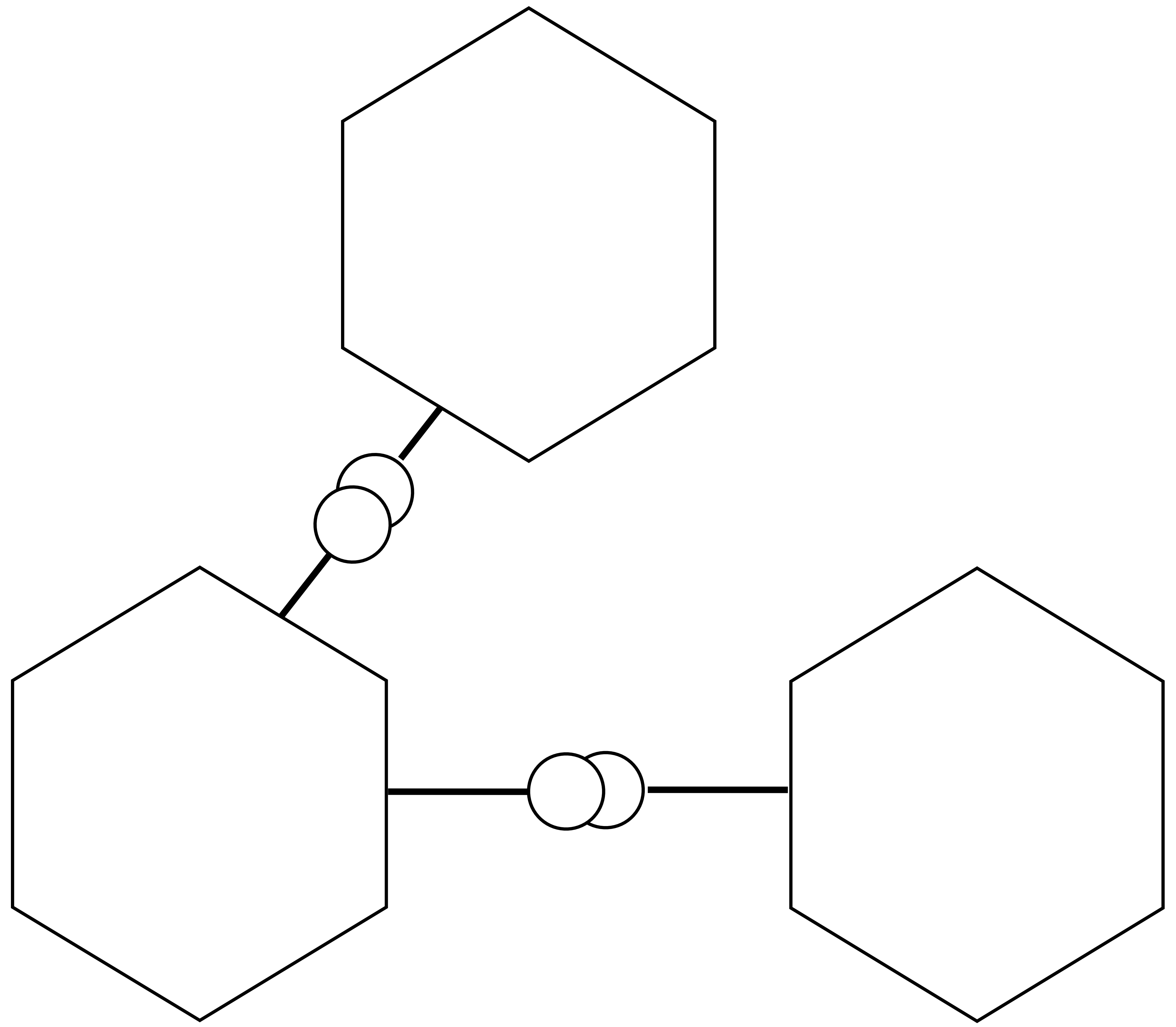
microservices



microservices

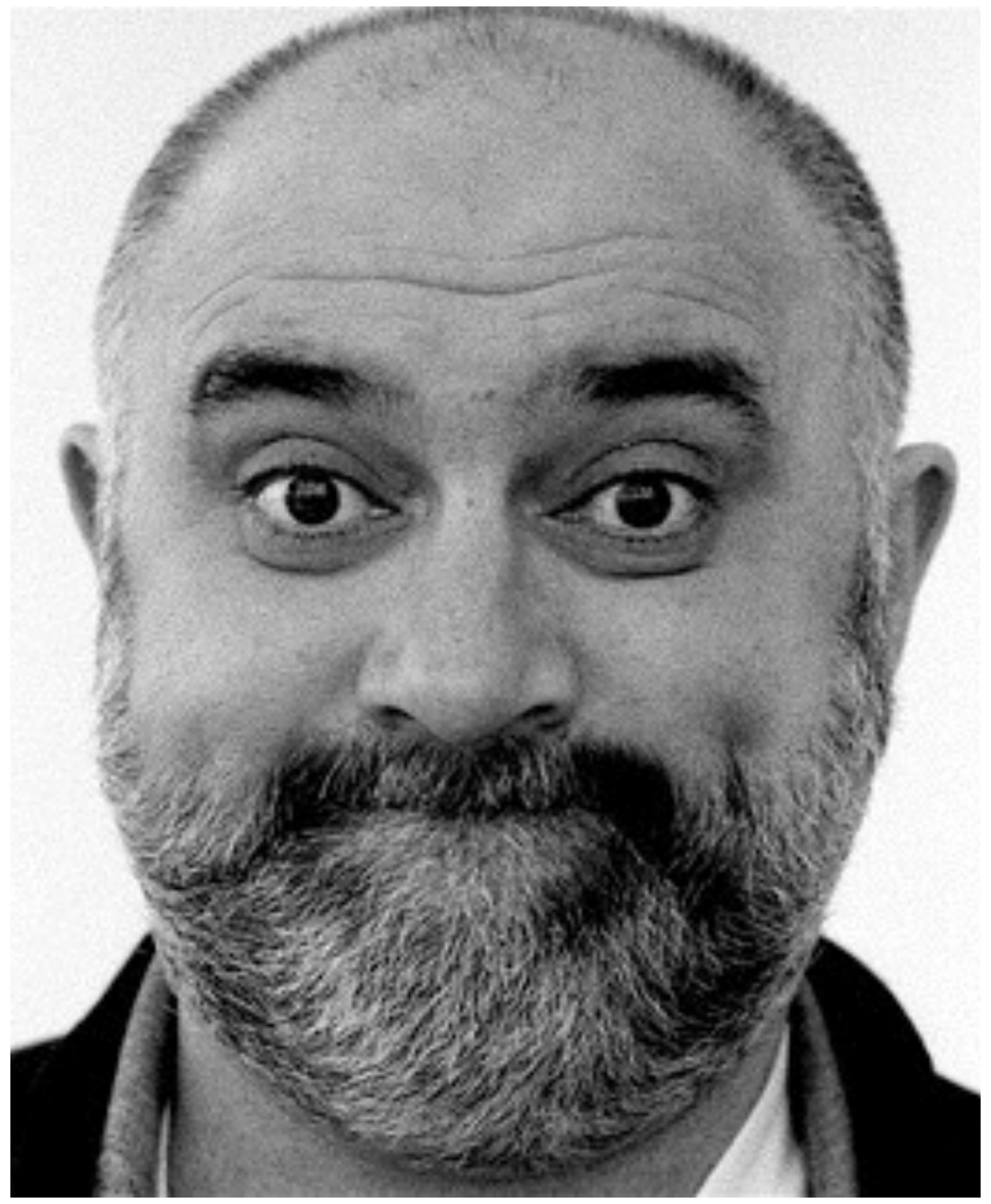


microservices



AS WE CHUNK UP DOMAINS, EACH DOMAIN SHOULD BE SMALL ENOUGH TO FIT IN MY HEAD

microservices



AND WHILE I HAVE A GIANT HEAD, ITS NOT FULL OF MUCH STUFF SO THATS OK...

Microservices

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

---

25 March 2014

James Lewis

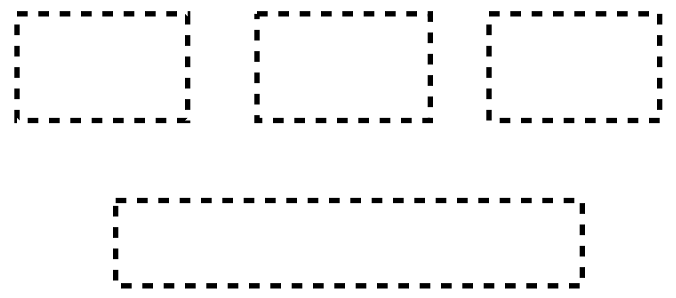
James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of systems using microservices and has been an active participant in the growing community for a couple of years.

Martin Fowler

Martin Fowler is an author, speaker, and general loud-mouth on software development. He's long been puzzled by the problem of how to componentize

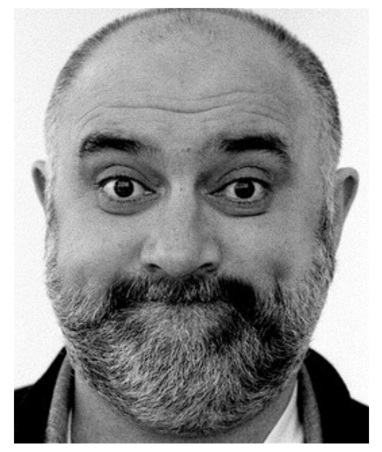
Contents

Sidebars

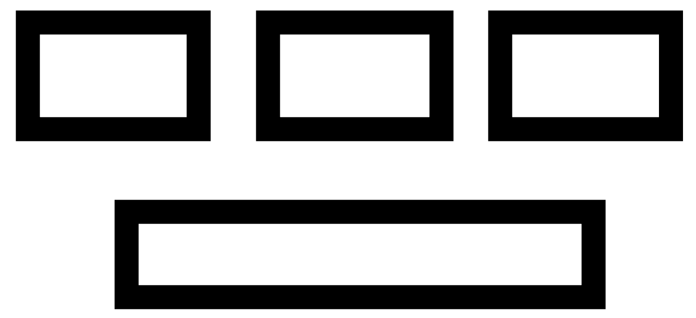


separate products

conway’s law



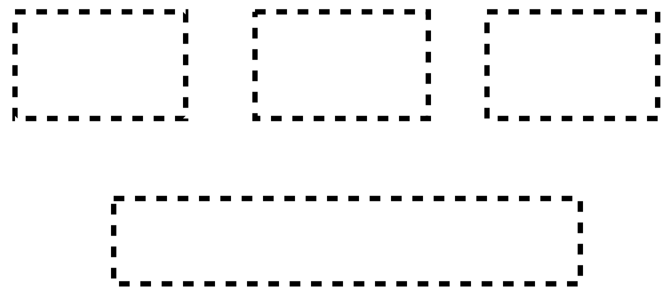
microservices

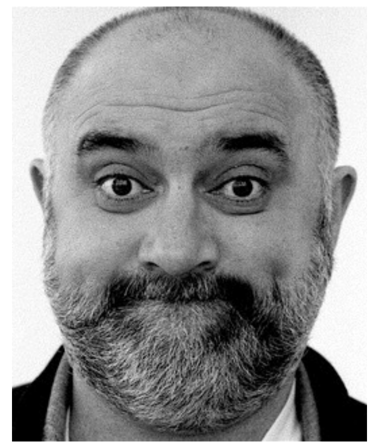


event sourcing

monitoring

right tool for the job





separate products

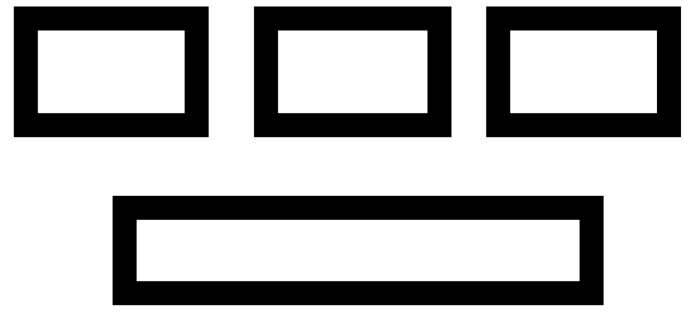
conway’s law

microservices

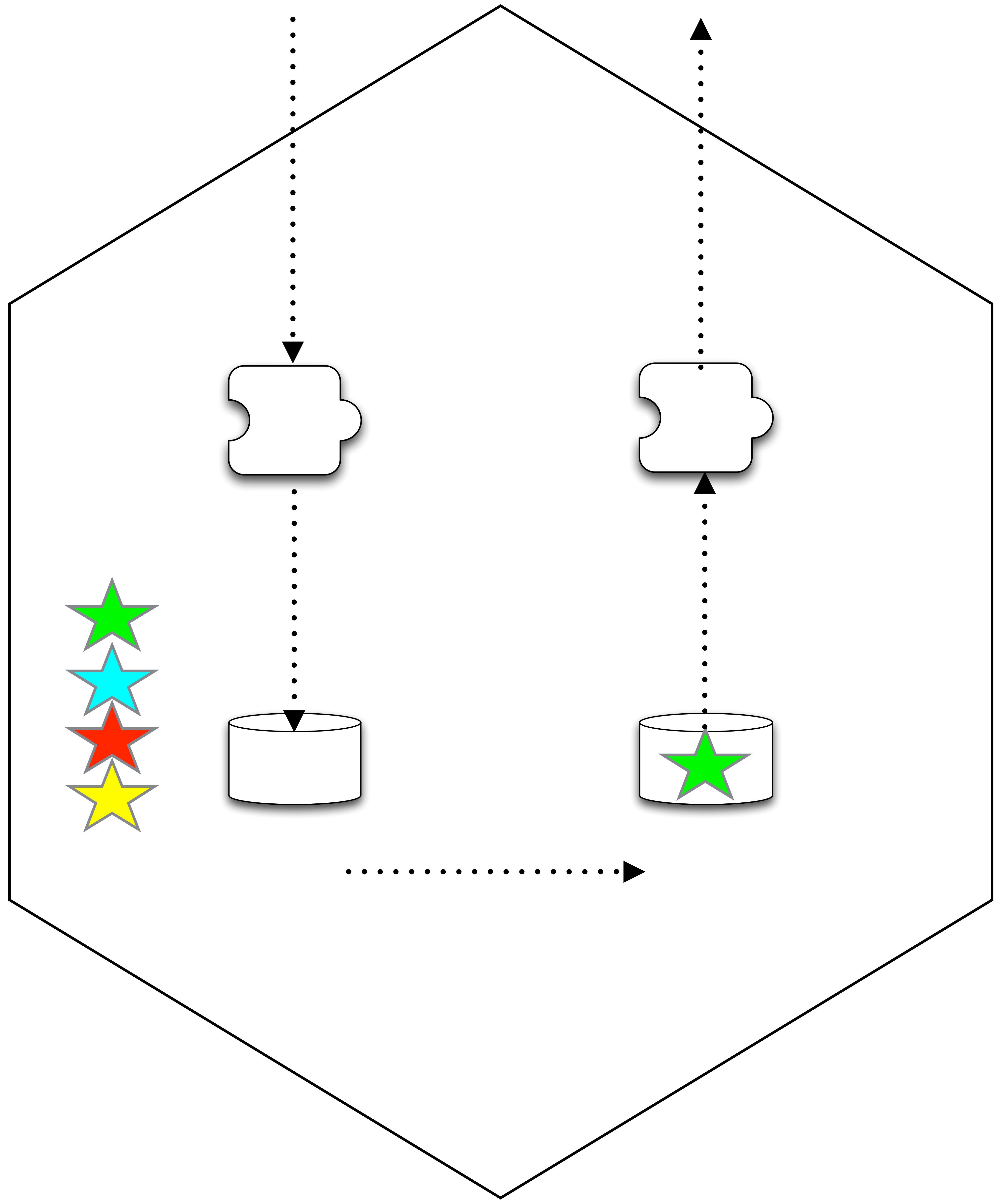
event sourcing

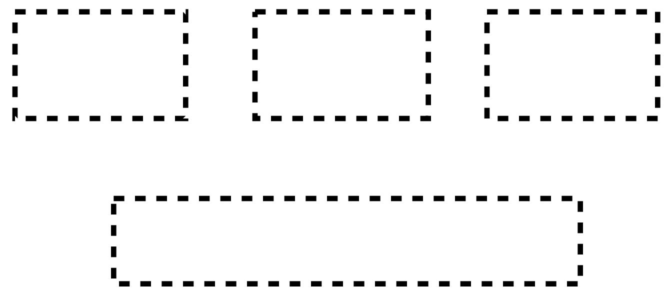
monitoring

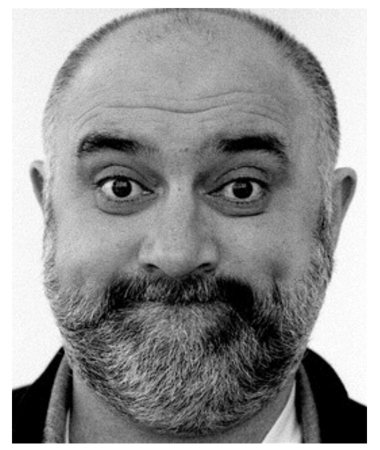
right tool for the job



event sourcing







separate products

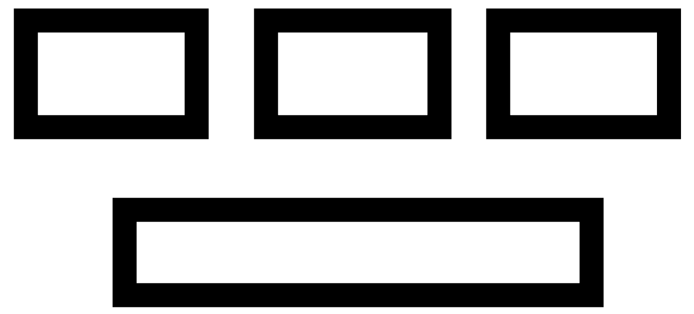
conway's law

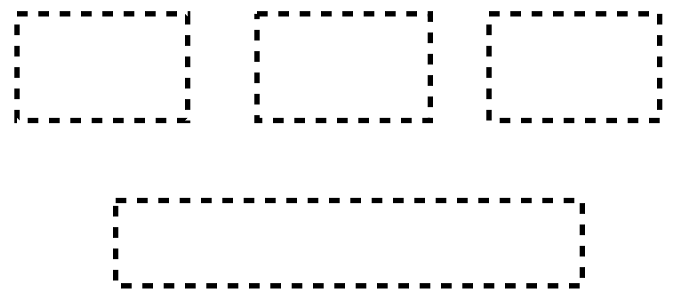
microservices

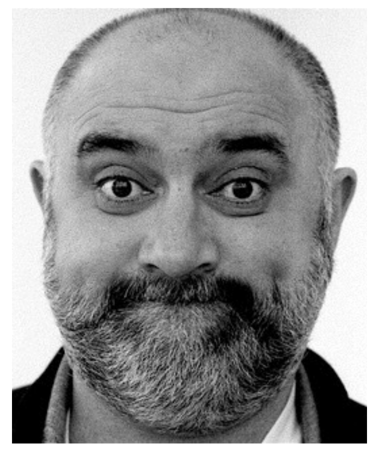
event sourcing

monitoring

right tool for the job







separate products

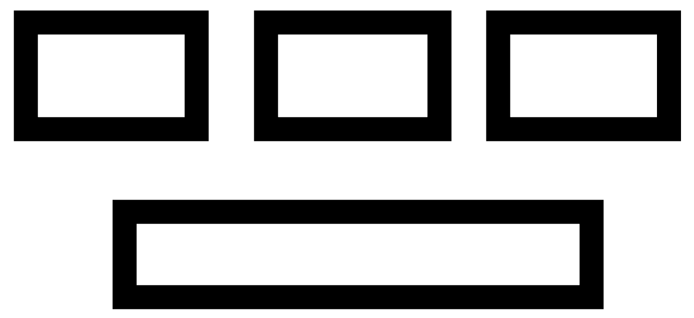
conway's law

microservices

event sourcing

monitoring

right tool for the job



monitoring

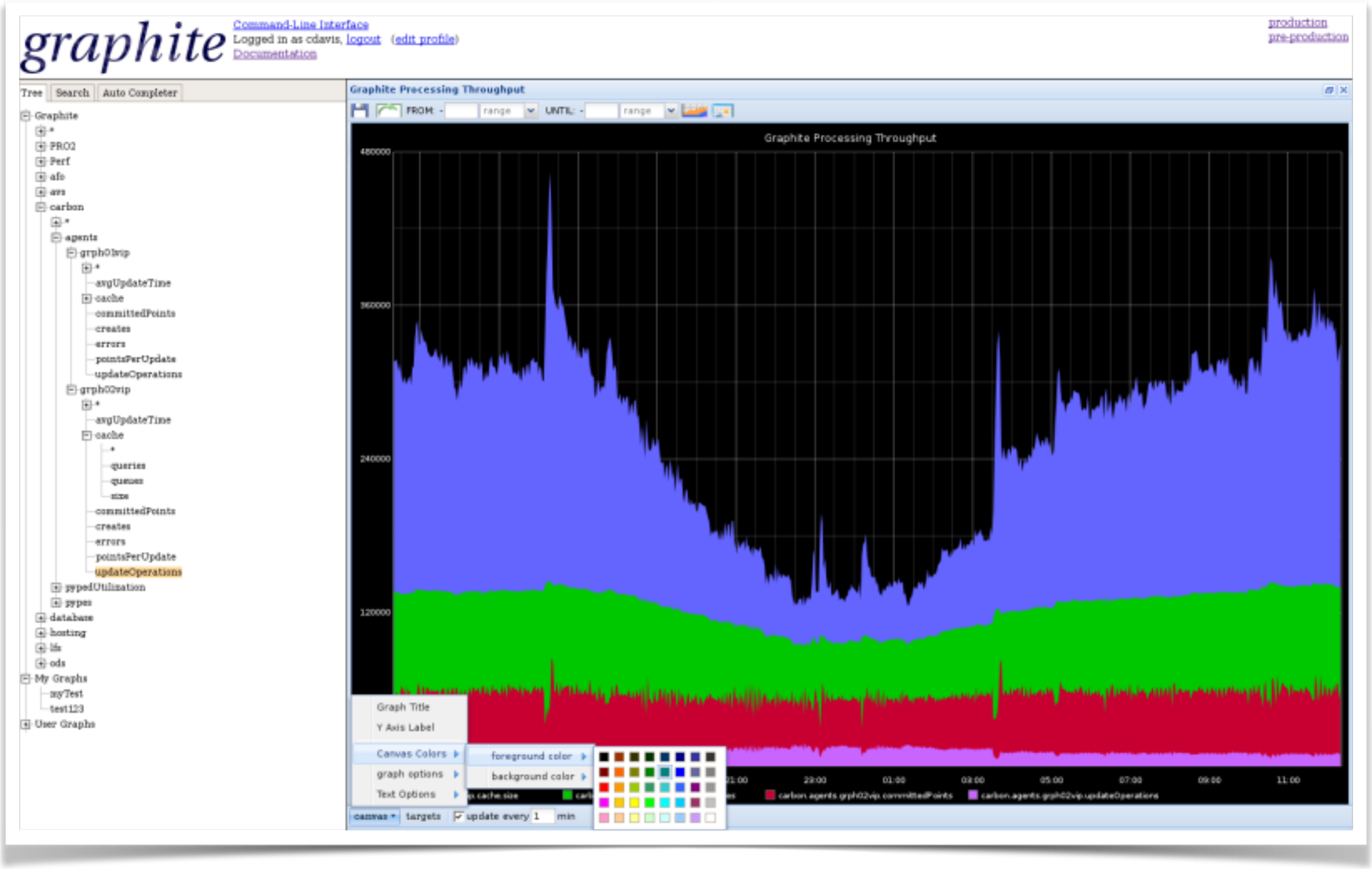
“the understanding of a specific cause

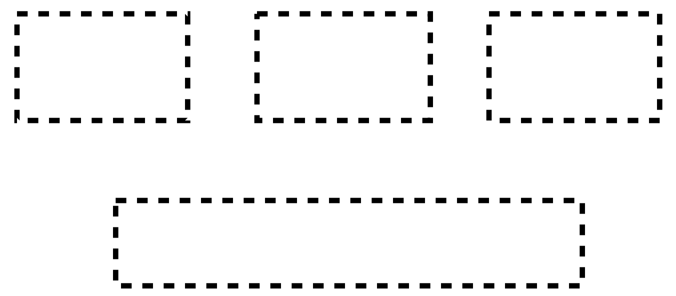
and effect in a specific context”

insight

operational insight

business insight


graphite
Command-Line Interface
Logged in as cdavis, logout (edit profile)
Documentation
production
pre-production
Tree
Search
Auto Completer
Graphite
*
PRO2
Perf
afo
avs
carbon
*
agents
grph01vip
*
avgUpdateTime
cache
committedPoints
creates
errors
pointsPerUpdate
updateOperations
grph02vip
*
avgUpdateTime
cache
*
queries
queues
size
committedPoints
creates
errors
pointsPerUpdate
updateOperations
pypedUtilization
pypes
database
hosting
lfs
ods
My Graphs
myTest
test123
User Graphs
Graphite Processing Throughput
FROM
range
UNTIL
range
Graphite Processing Throughput
480000
360000
240000
120000
21:00
23:00
01:00
03:00
05:00
07:00
09:00
11:00
p.cache.size
carbon.agents.grph02vip.committedPoints
carbon.agents.grph02vip.updateOperations
Graph Title
Y Axis Label
Canvas Colors
graph options
Text Options
foreground color
background color
canvas
targets
update every 1 min



separate products

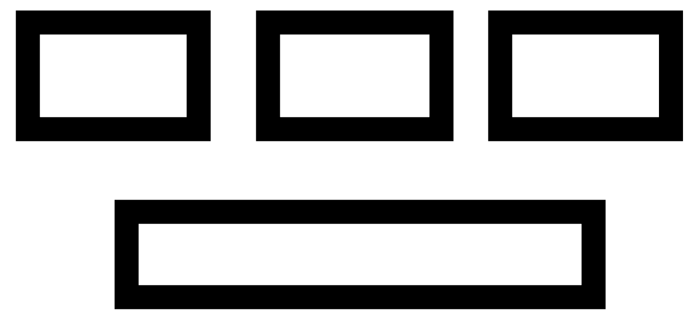
conway's law

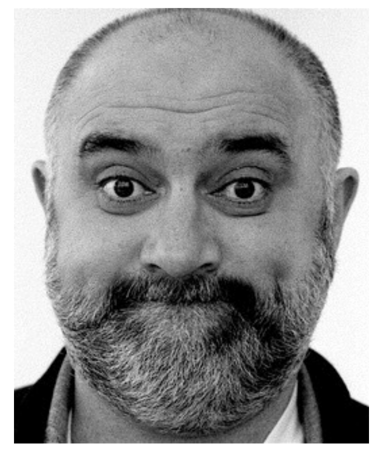
microservices

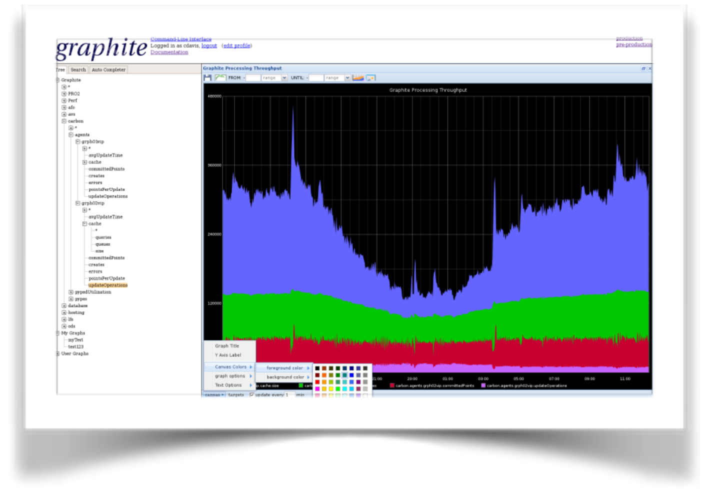
event sourcing

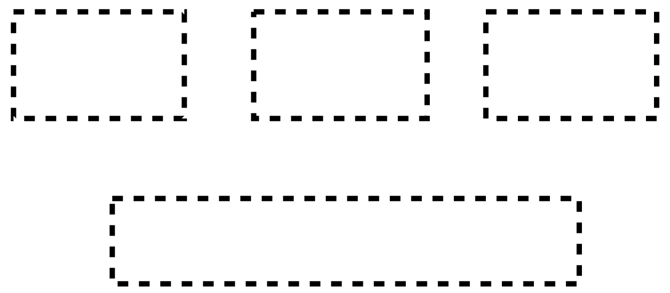
monitoring

right tool for the job









separate products

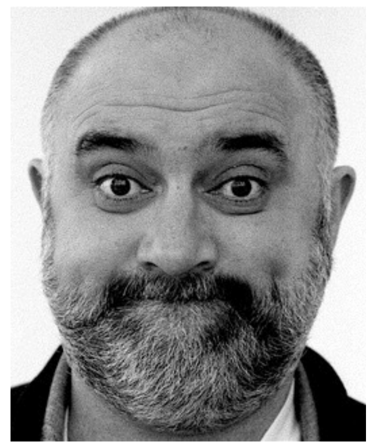
conway's law

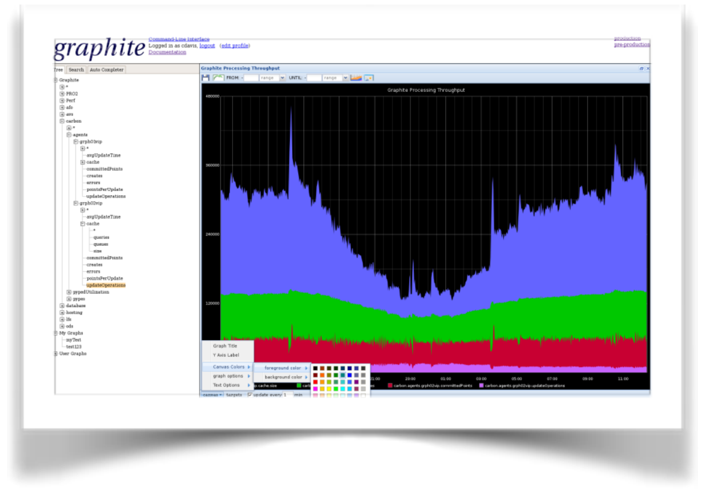
microservices

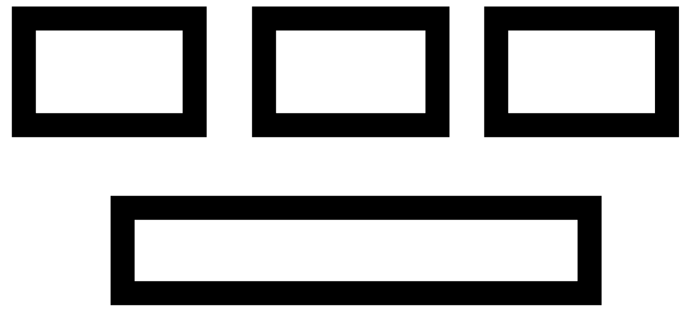
event sourcing

monitoring

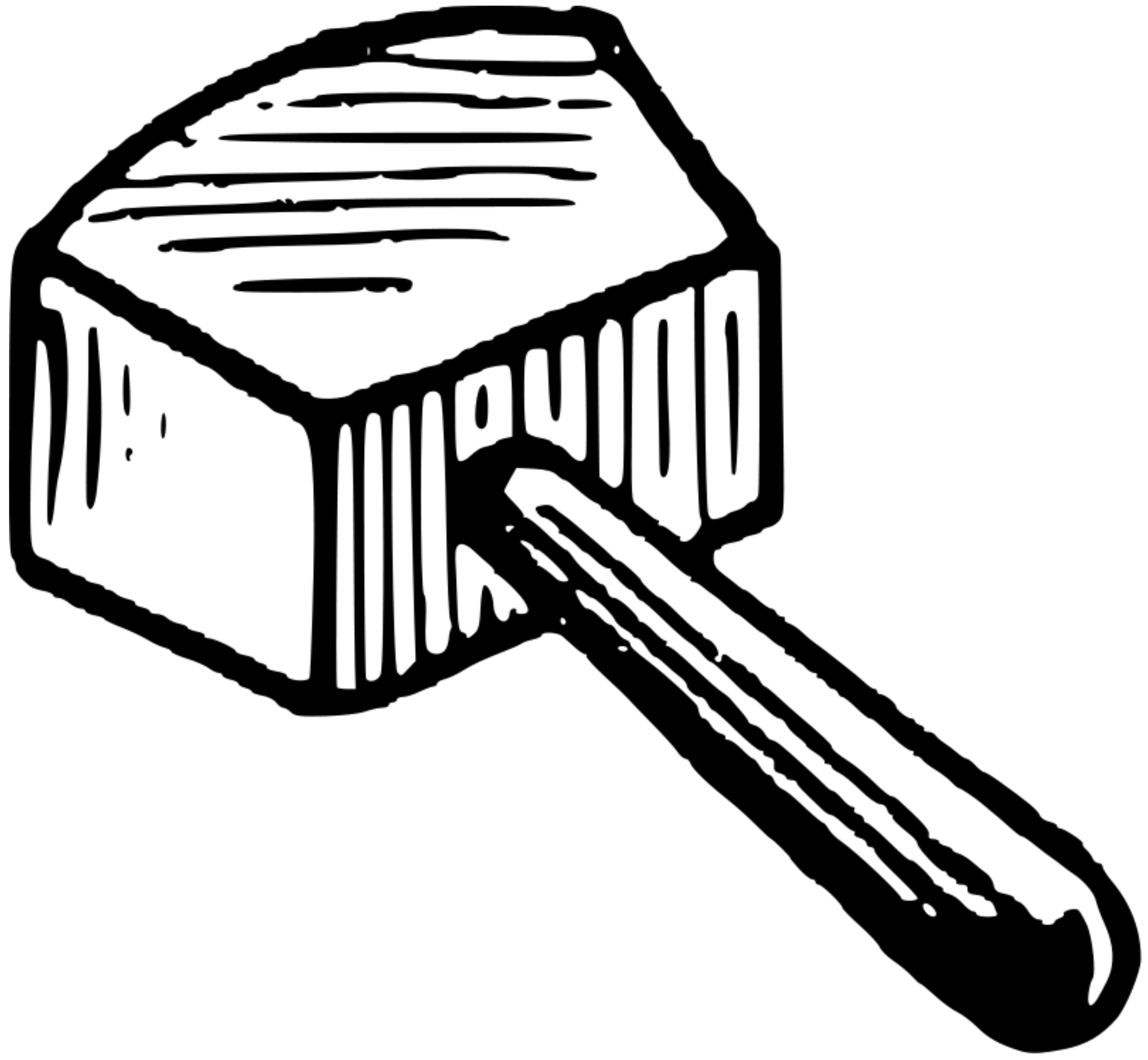
right tool for the job



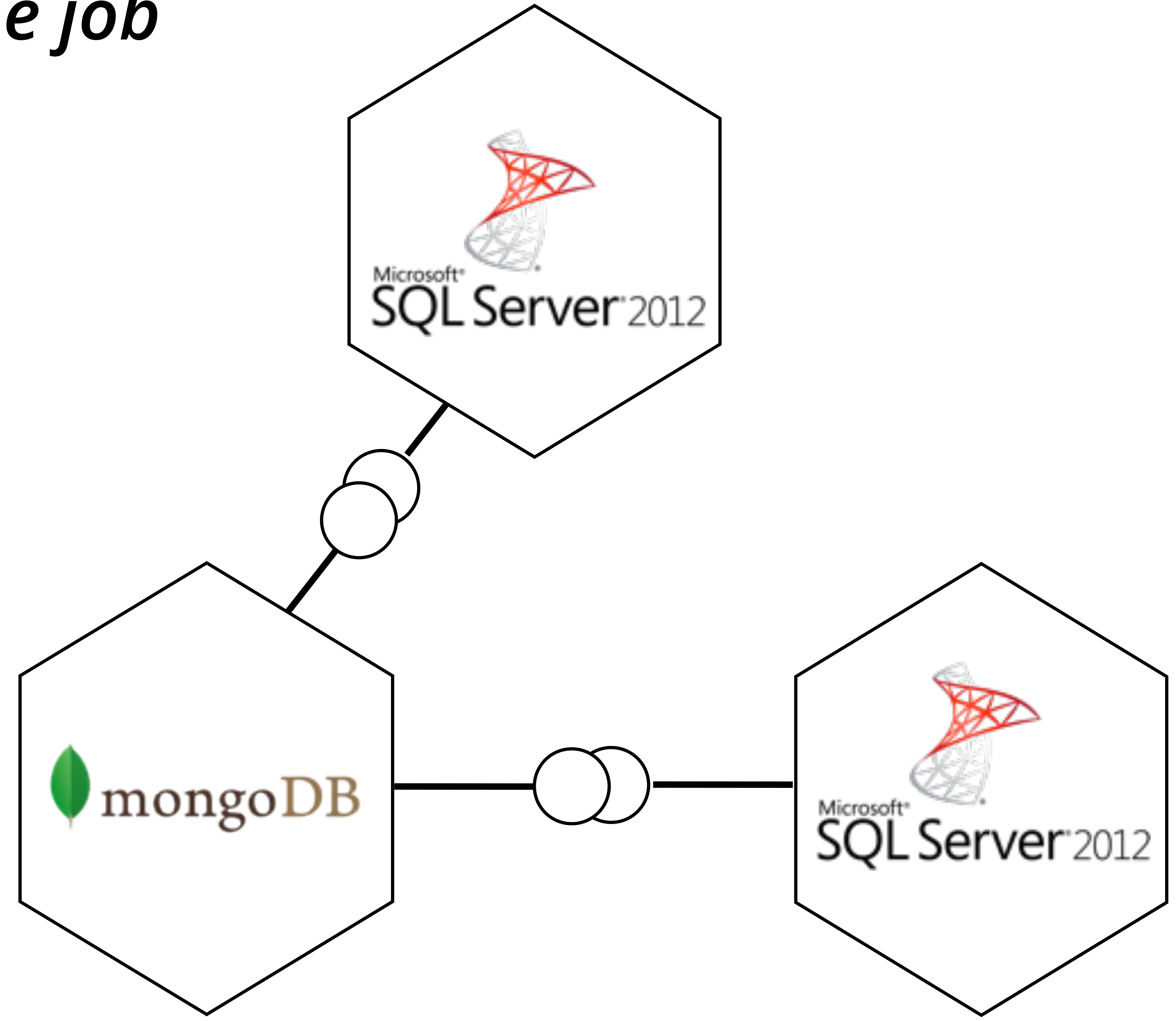




right tool for the job



right tool for the job

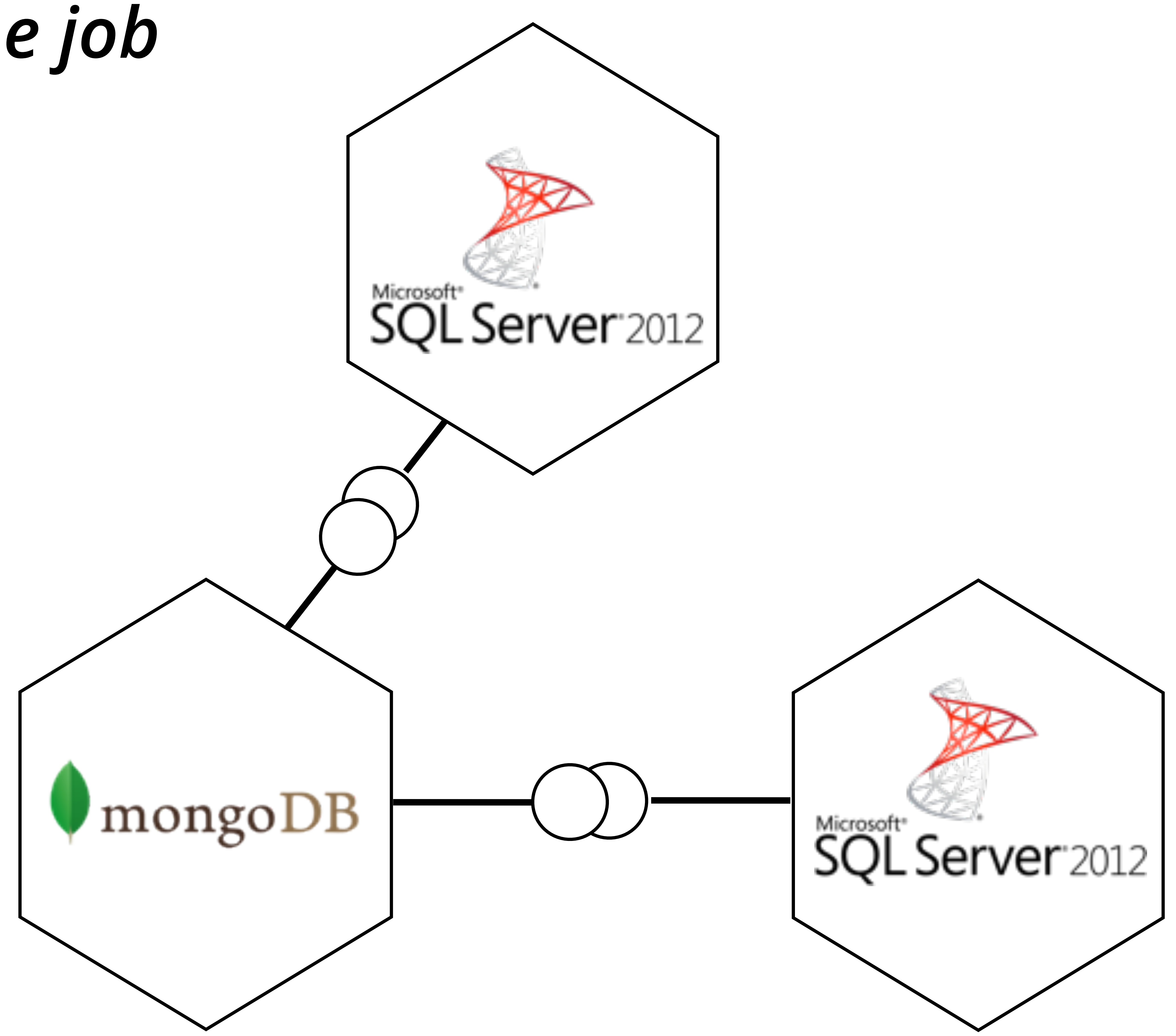


right tool for the job

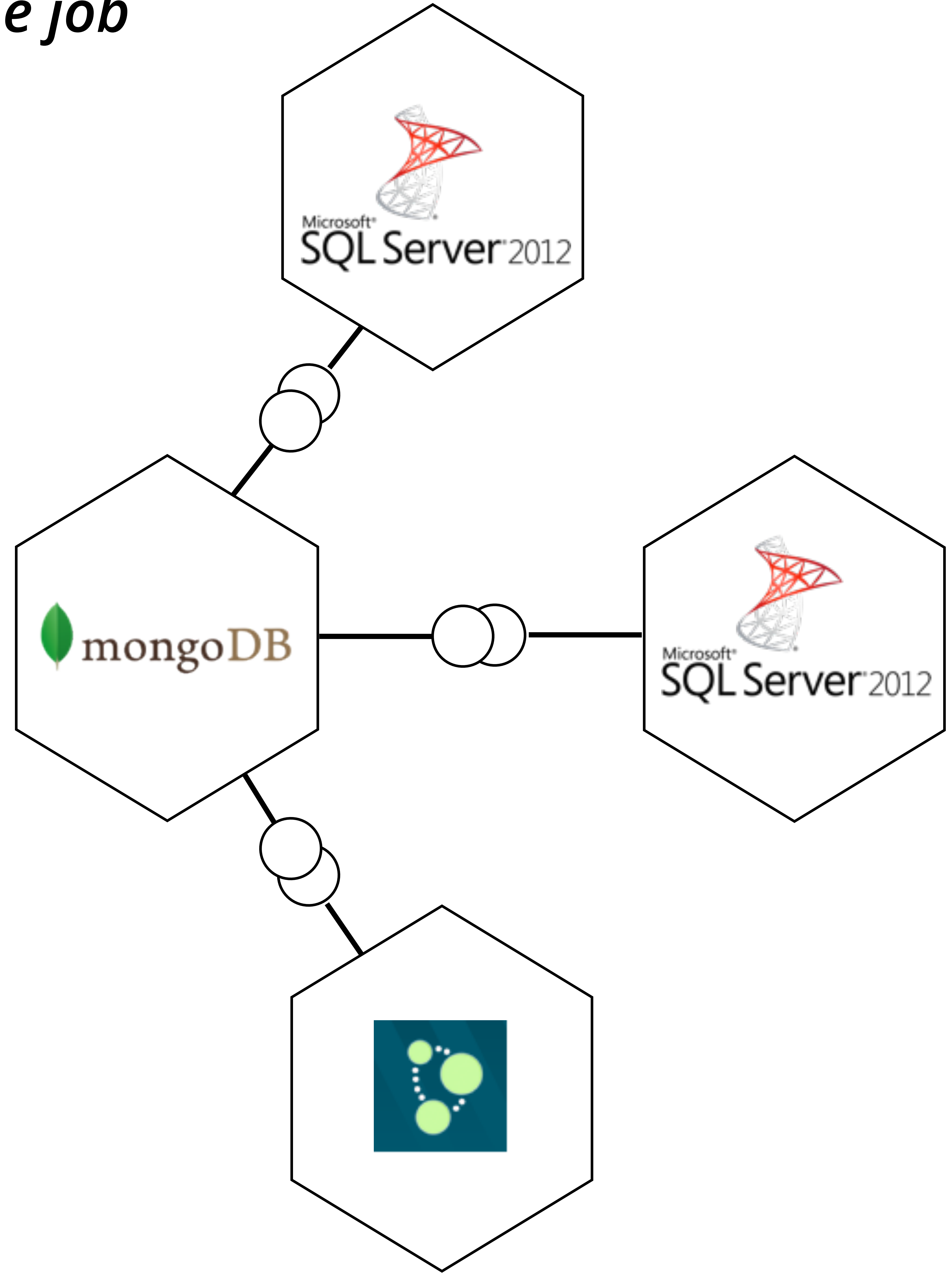
“Replaceable Component Architectures”

Dan North

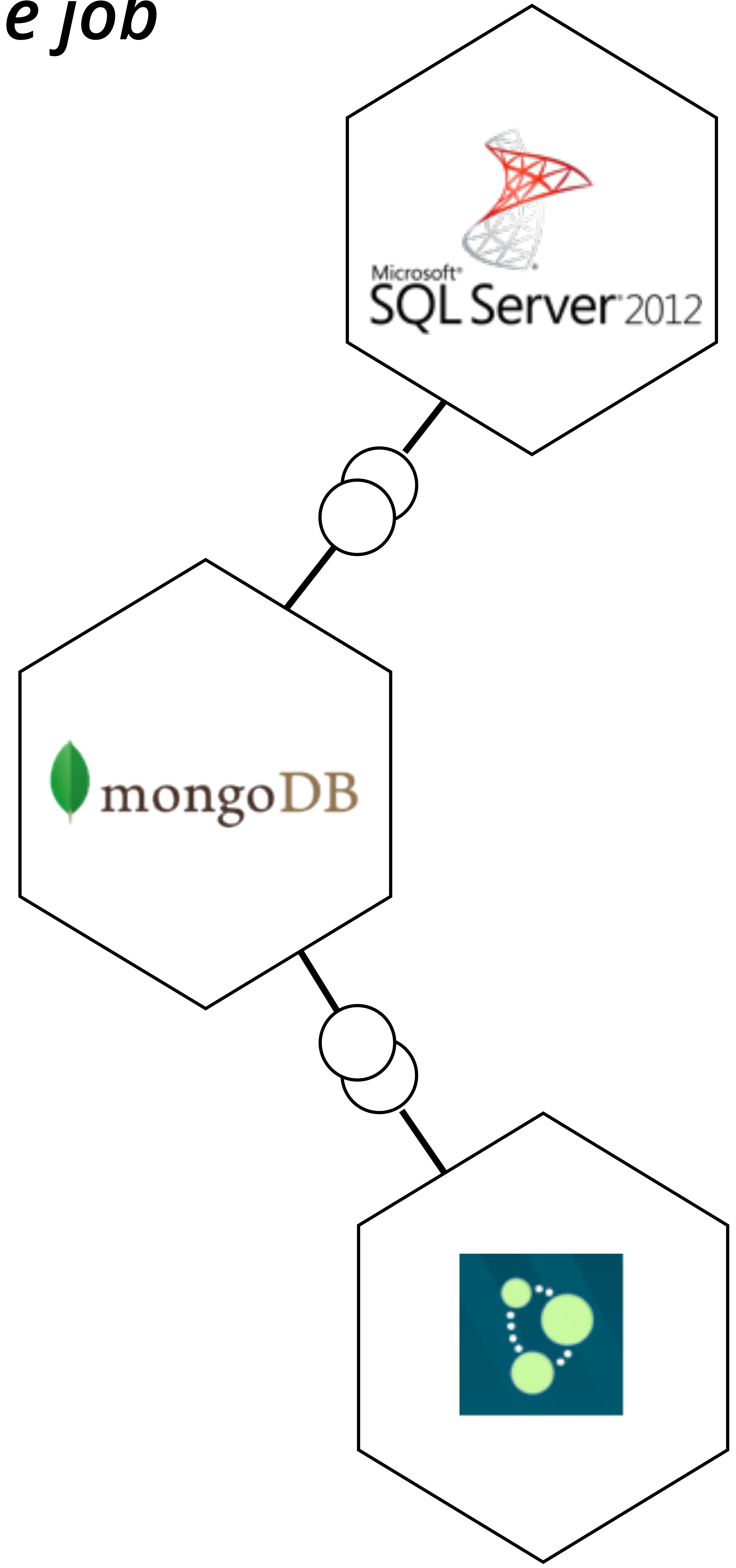
right tool for the job

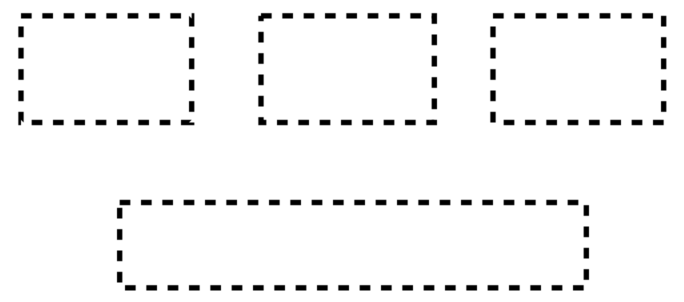


right tool for the job

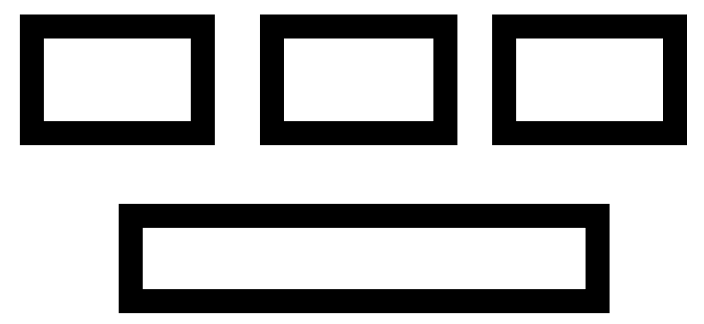


right tool for the job

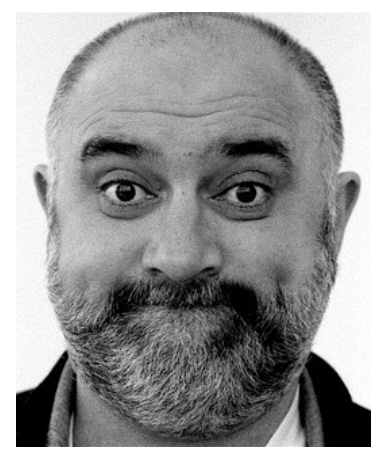




separate products

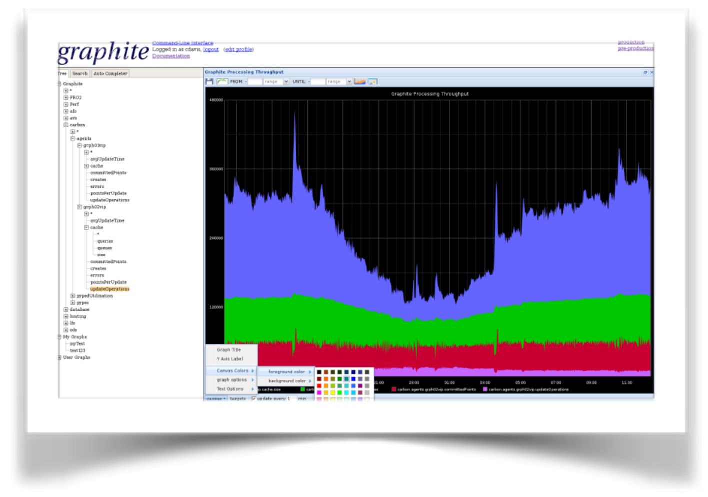


conway's law



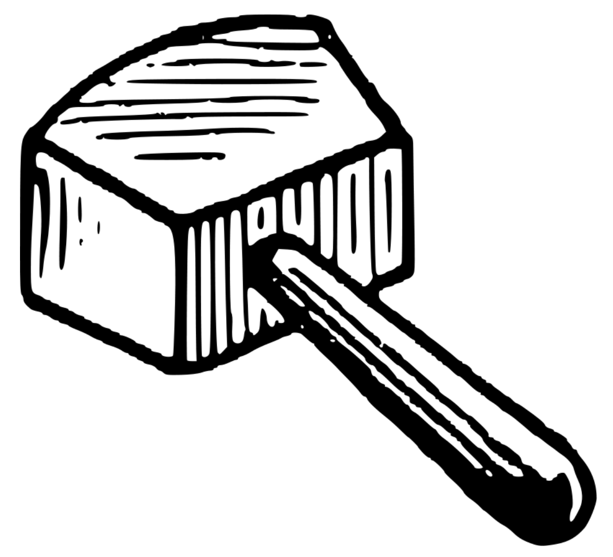
microservices

event sourcing

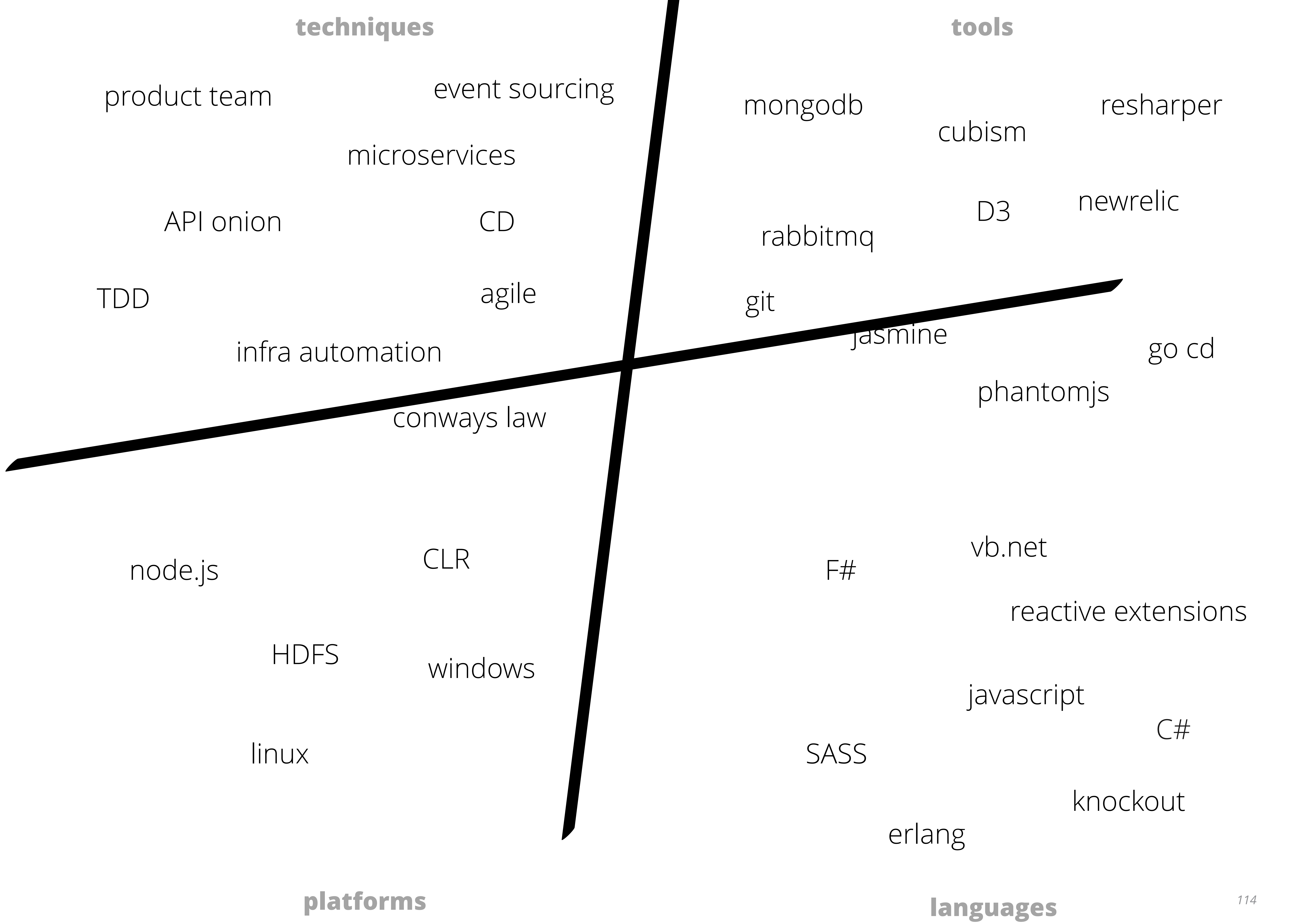


monitoring

right tool for the job



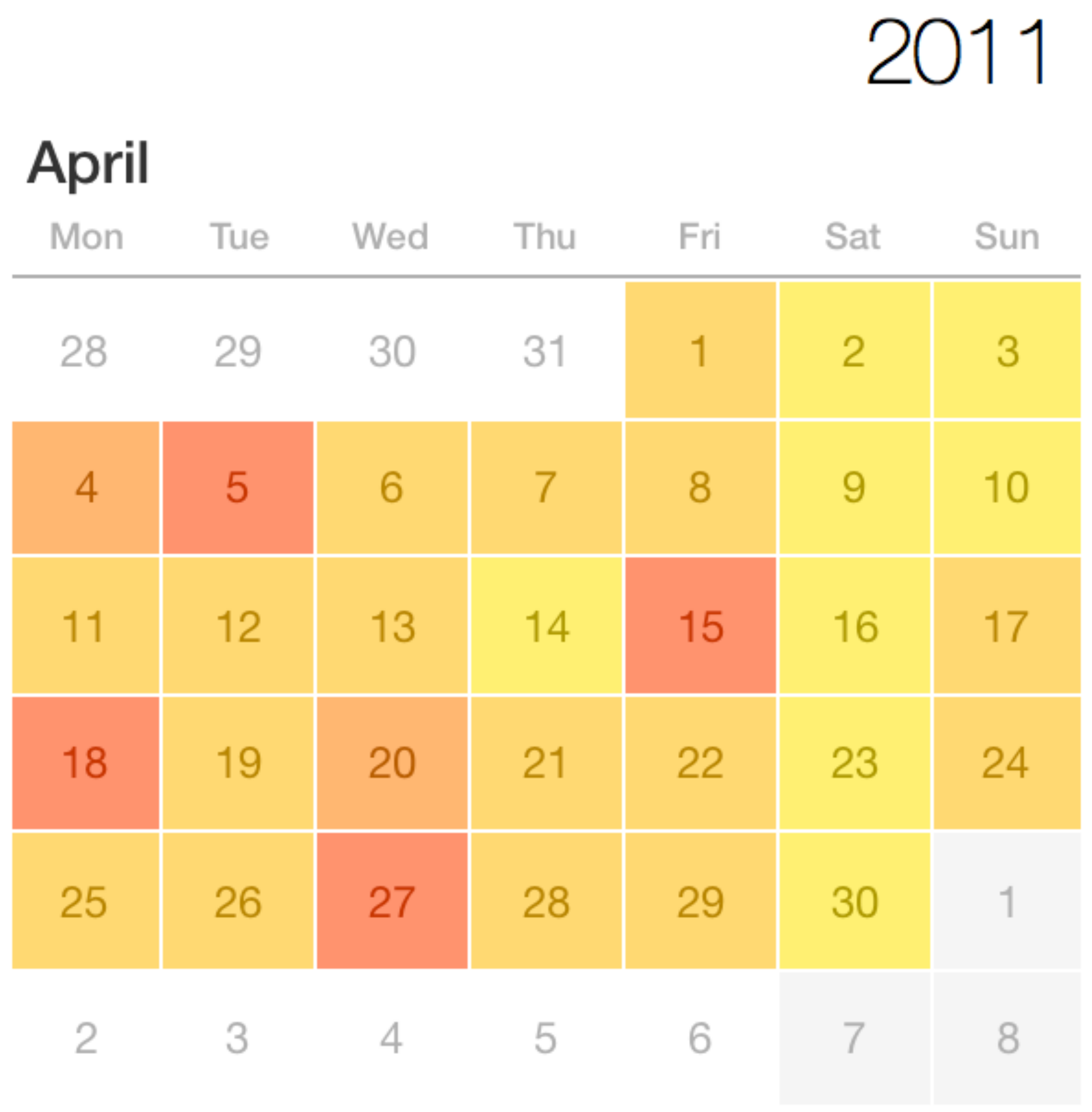
in summary


techniques
tools
product team
event sourcing
microservices
API onion
CD
TDD
agile
infra automation
conways law
mongodb
cubism
resharper
D3
newrelic
rabbitmq
git
jasmine
go cd
phantomjs
node.js
CLR
HDFS
windows
linux
F#
vb.net
reactive extensions
javascript
C#
SASS
knockout
erlang
platforms
languages

along the way we all learned some things

- ☐ events are awesome
- ☐ events suck
- ☐ taking on new languages is scary, but exciting
- ☐ because javascript (WAT?)
- ☐ API design is really really important
- ☐ asynchronicity, wait(10.seconds()), is tedious
- ☐ silo'd metrics suck
- ☐ just saying "devops" doesn't make it devops (and it's really scary to devs and ops)

change takes longer than you think



#neverdone

ThoughtWorks®

THANKS

james lewis

jalewis@thoughtworks.com
@boicy