WTF in Clichés

Balajee Mohan

@balajeetm

TESCO Technology

What's The Fun in Clichés

Balajee Mohan

@balajeetm

TESCO Technology

KISS

K.I.S.S

Keep It Short & Simple

What's The Fun in Clichés

Balajee Mohan

@balajeetm

TESCO Technology

What's The Fun in Clichés

Balajee Mohan

@balajeetm

TESCO Technology

What's The Fun in Clichés

Agile, Microservices, DevOps

Balajee Mohan

@balajeetm

TESCO Technology

Balajee Mohan

TESCO Technology

@balajeetm

http://blog.balajeetm.com/

Balajee Mohan

TESCO Technology

Transform the way we build software

@balajeetm

http://blog.balajeetm.com/

Balajee Mohan

TESCO Technology

Transform the way
we build software

@balajeetm

http://blog.balajeetm.com/

Balajee Mohan

TESCO Technology

Transform the way we build software

@balajeetm

http://blog.balajeetm.com/

SOFTWARE ENGINEERS

!=

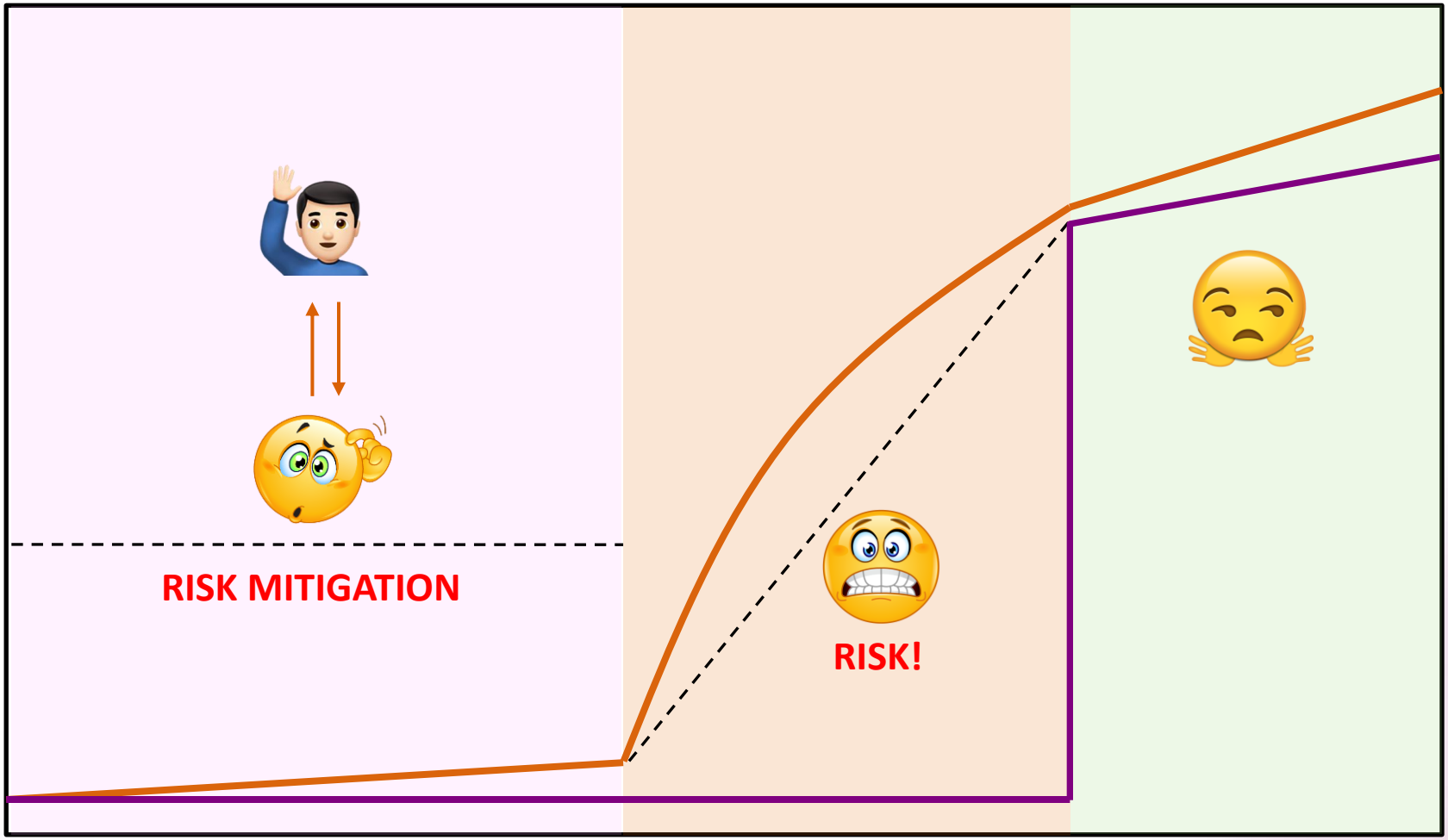
ENGINEERS

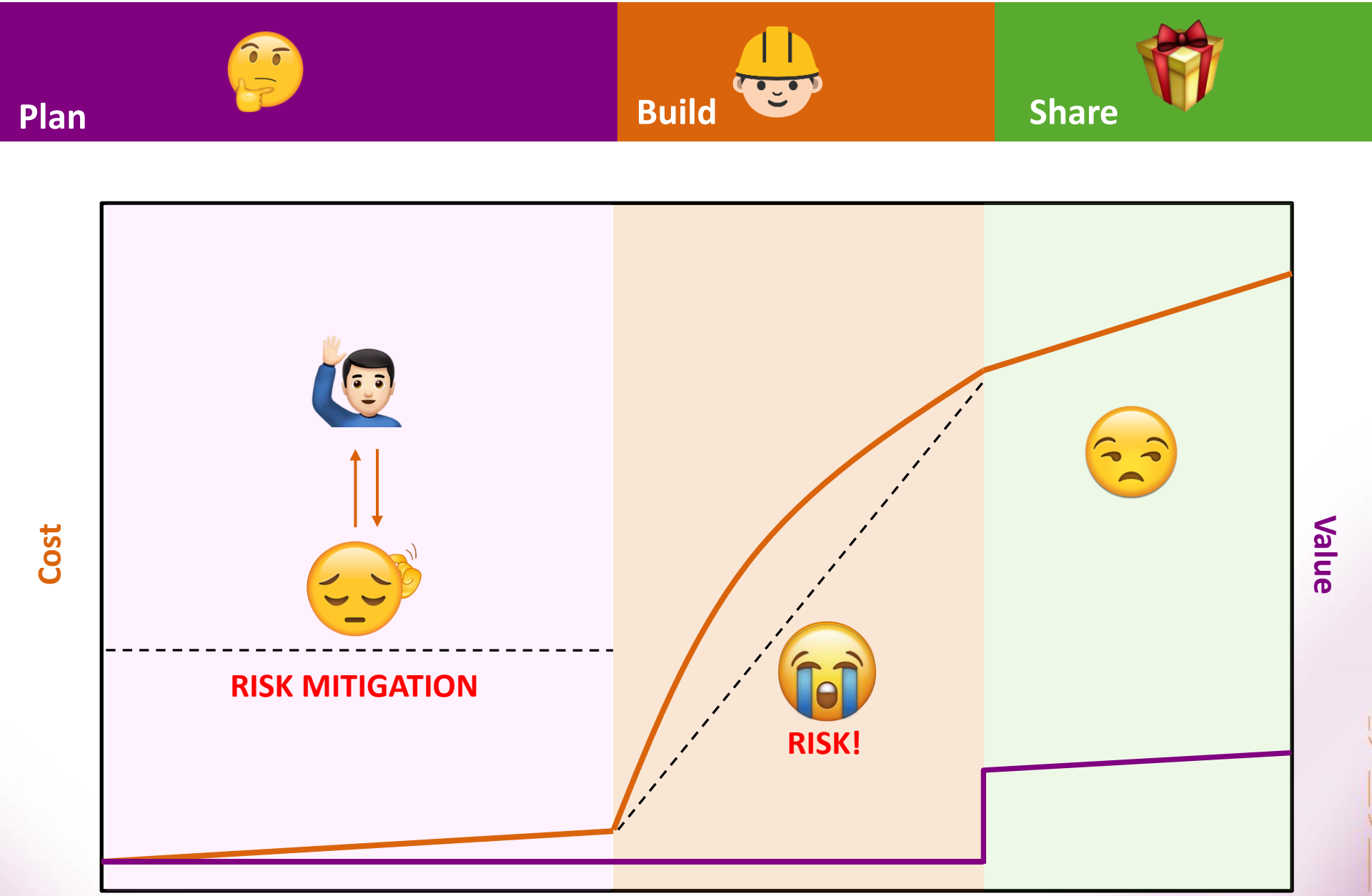
Plan 5 years | Build 4 years | Share

Plan

Build

Share




Plan
Build
Share
Cost
Value
RISK MITIGATION
RISK!
Time

SOFTWARE

!=

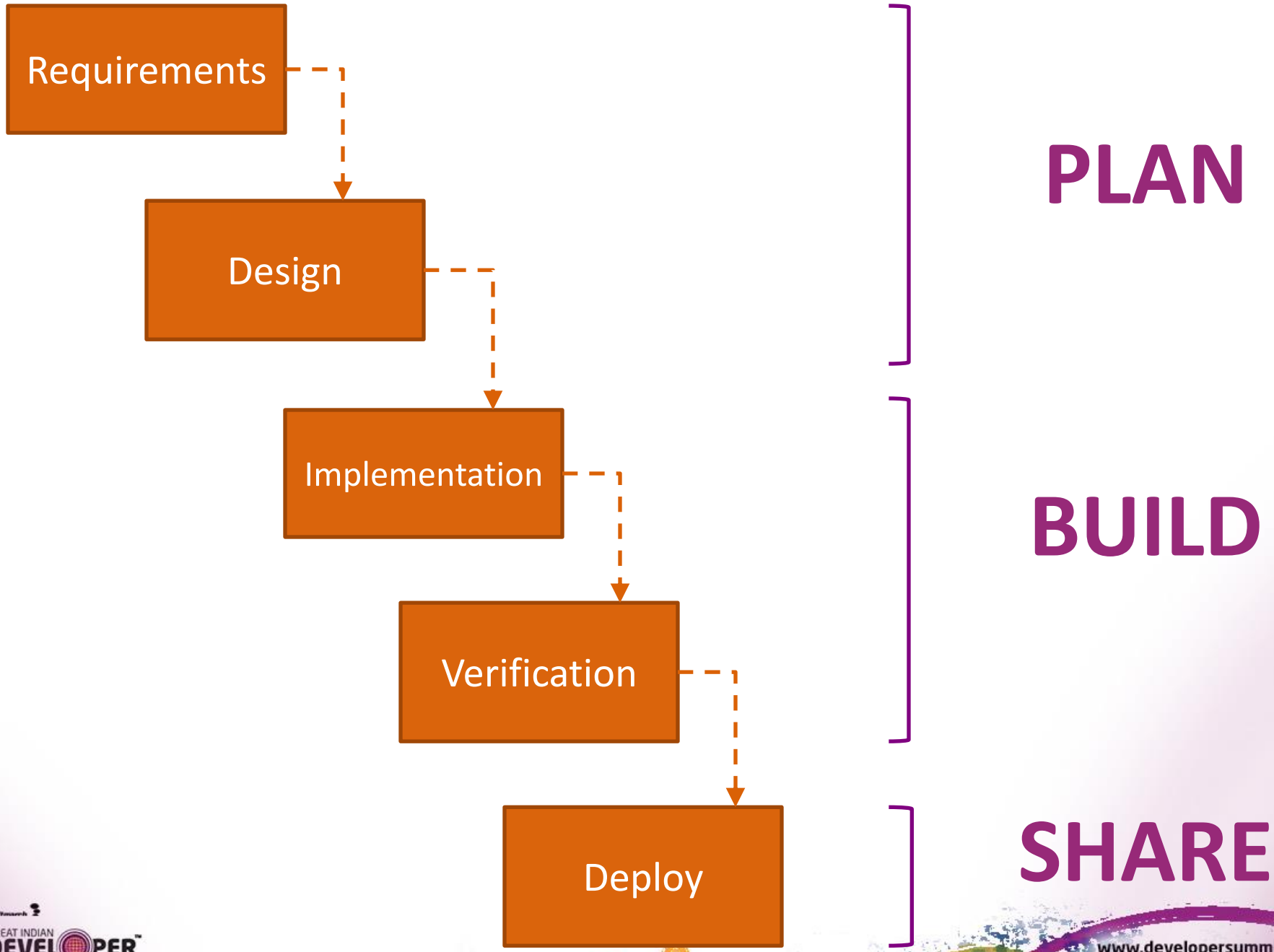
BRIDGE BUILDING

Low Cost to Change

BUT

\+ Low Cost to Distribute

Express Iteration


Requirements
Design
Implementation
Verification
Deploy
PLAN
BUILD
SHARE

PLAN	BUILD	SHARE

BAD MITIGATION = GREATER RISK

PLAN

BUILD

SHARE

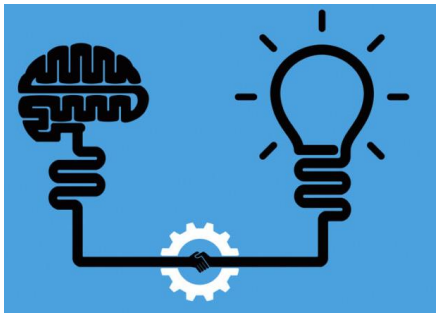
RISK MITIGATION ∝ FEEDBACK

INTERNET ERA

FASTER

FEEDBACK

INNOVATE



DELIVER

VELOCITY

Culture

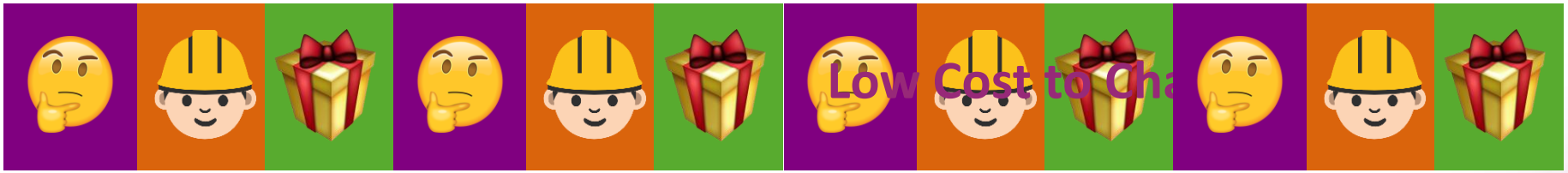
$$\vec{V} = |V| \cdot \hat{V}$$

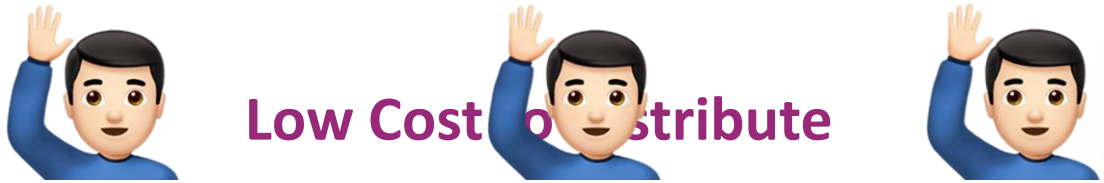
Culture Velocity

Build Expertise

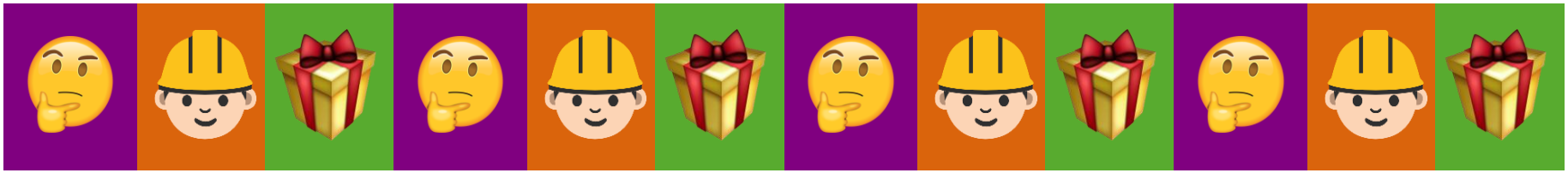
Learn & Unlearn

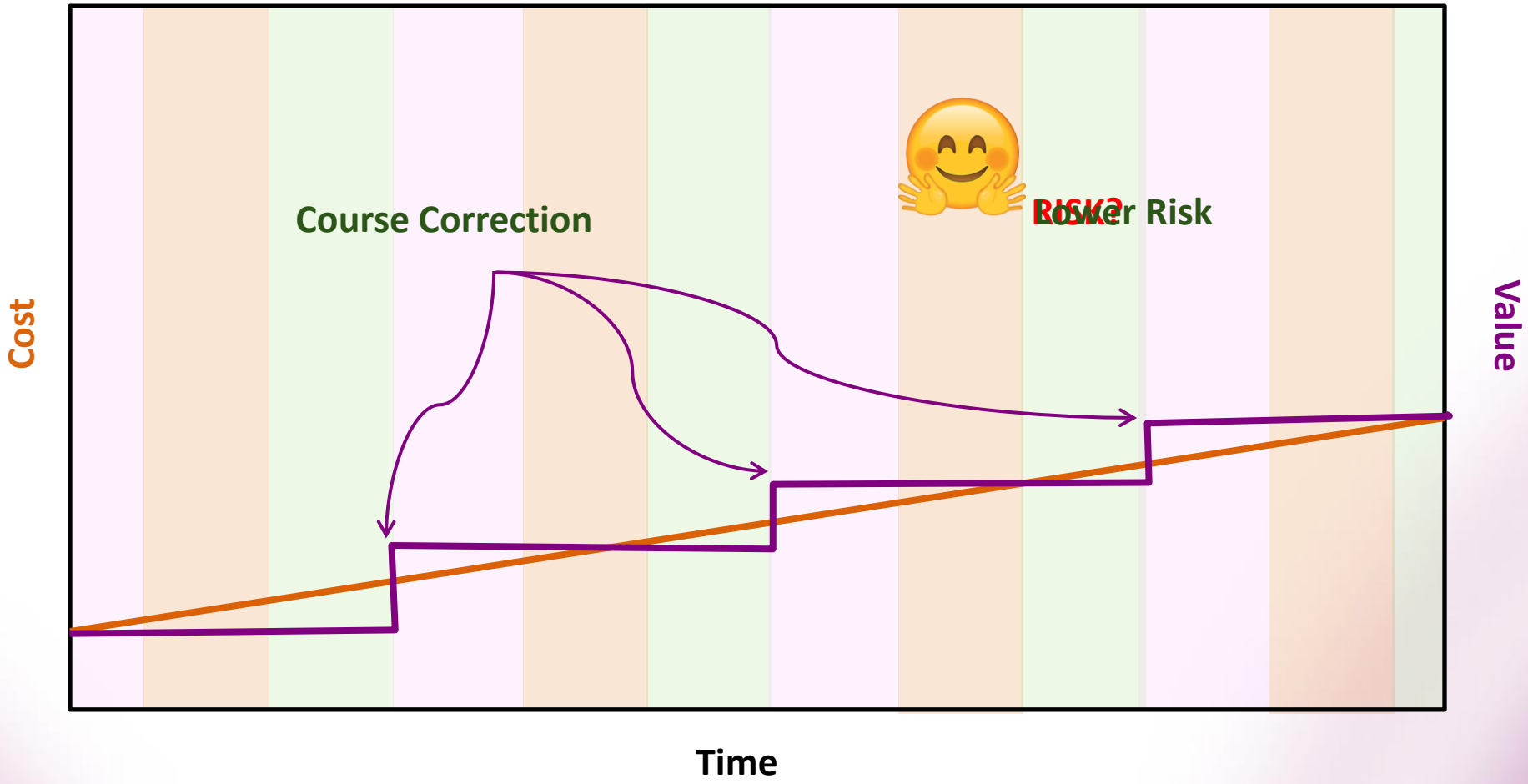
PLAN	BUILD	SHARE



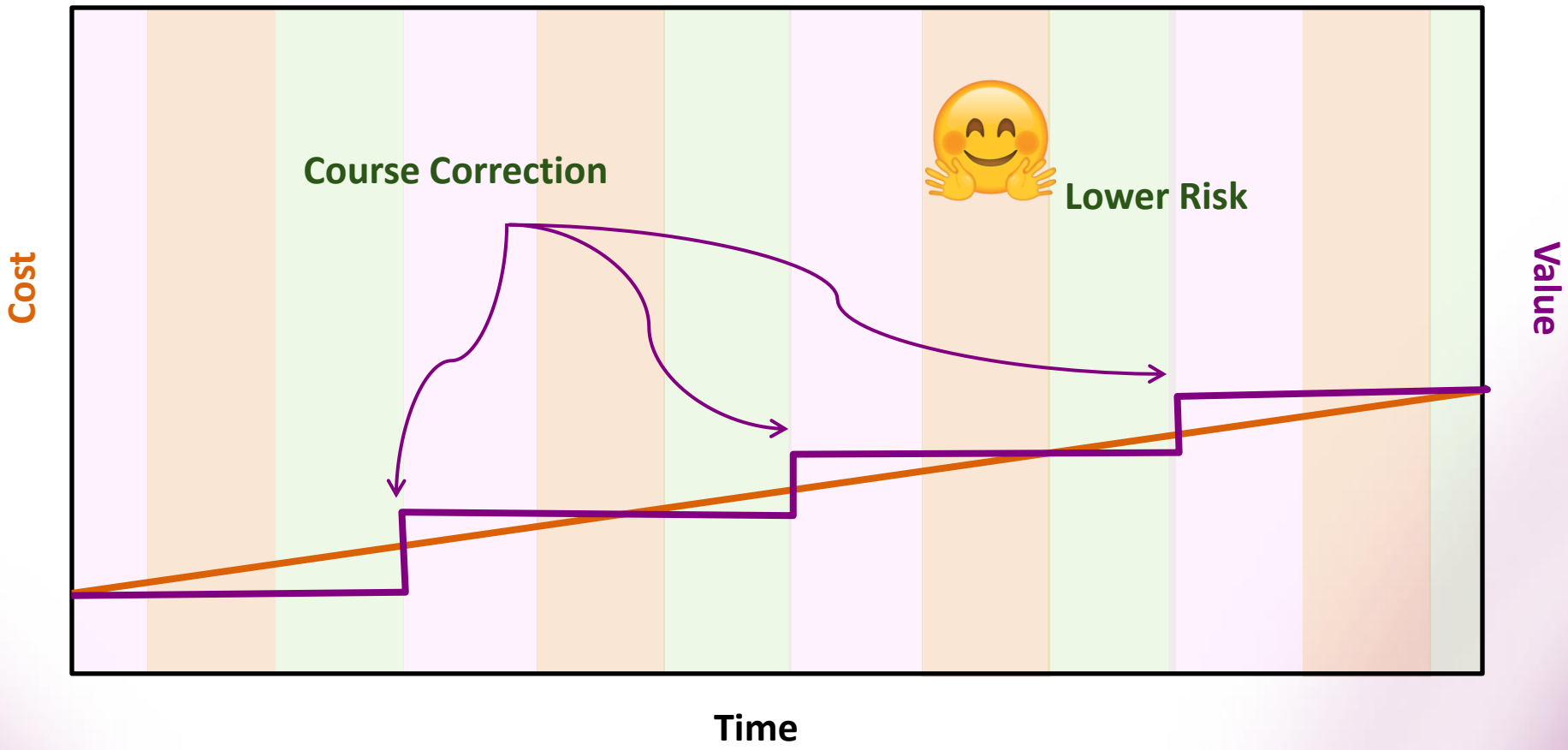


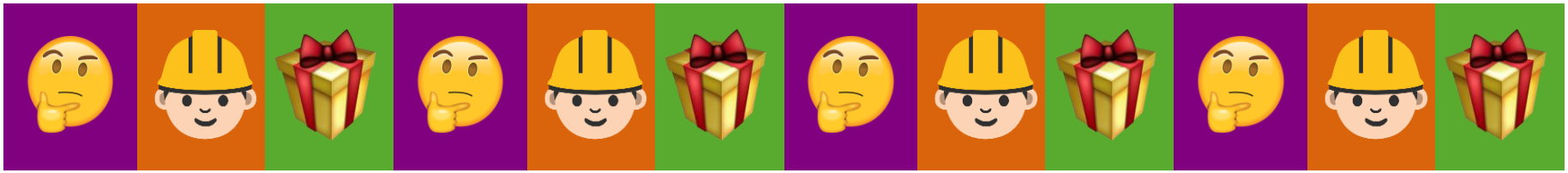
Low Cost to Distribute




Course Correction
Lower Risk
Cost
Value
Time

AGILE & EXPRESS ITERATION





AGILE & EXPRESS ITERATION

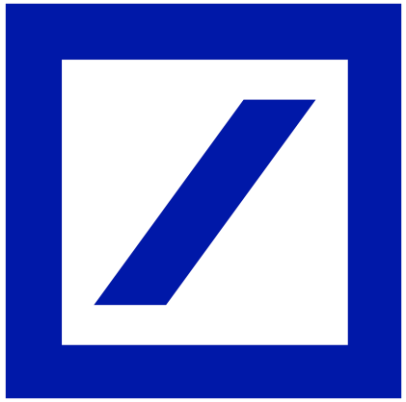
are the way to succeed in a competitive & unpredictable software economy

What is AGILE ?

Is a philosophy

What is AGILE ?

Is a philosophy



Deutsche Bank

https://www.db.com/company/en/media/logo-history.pdf

To achieve velocity by being flexible to change

Low Cost to Change

Low Cost to Distribute

Low Cost to Change ?

Low Cost to Distribute

Low Cost to Change

Low Cost to Change

Low Cost to Change

Low Cost to Change

Low Cost to Change

Low Cost to Change

Low Cost to Change

Low Cost to Change

Low Cost to Change

Low Cost to Change

COMPLEXITY

MONOLITH

MISCOMMUNICATION

CONFLICTING DIRECTIONS

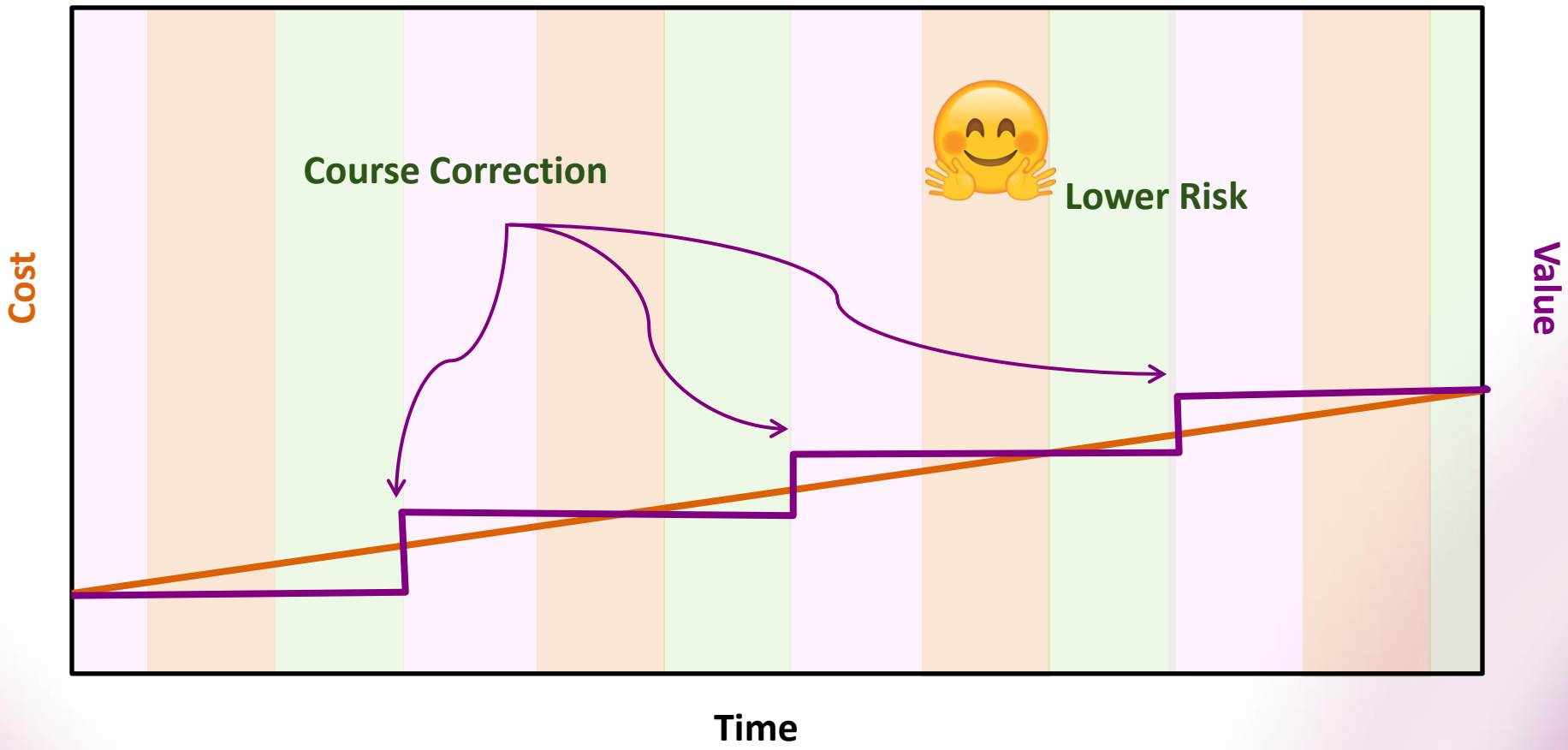
CONFLICT

INCONSISTENCY

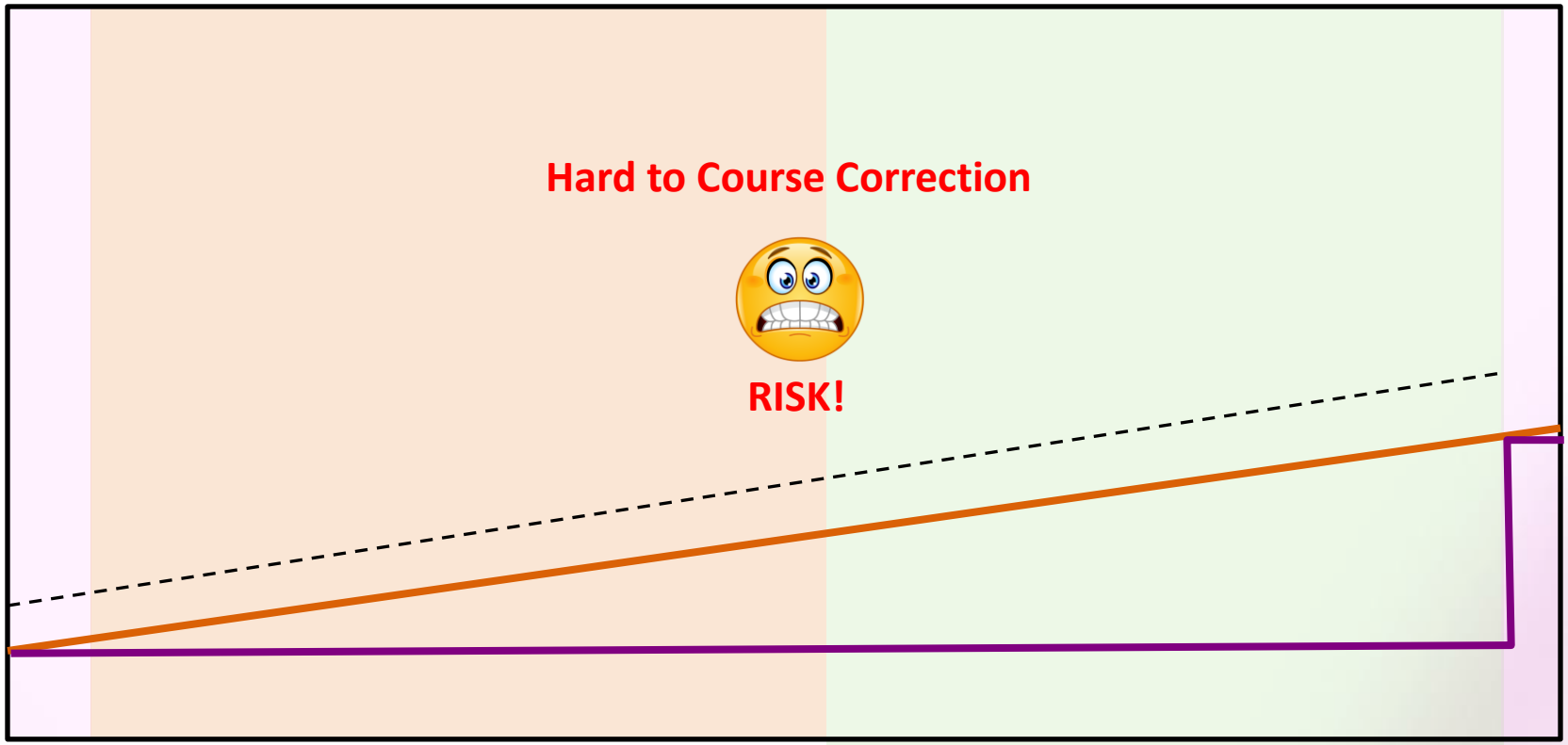
Low Cost to Change

Low Cost to Change

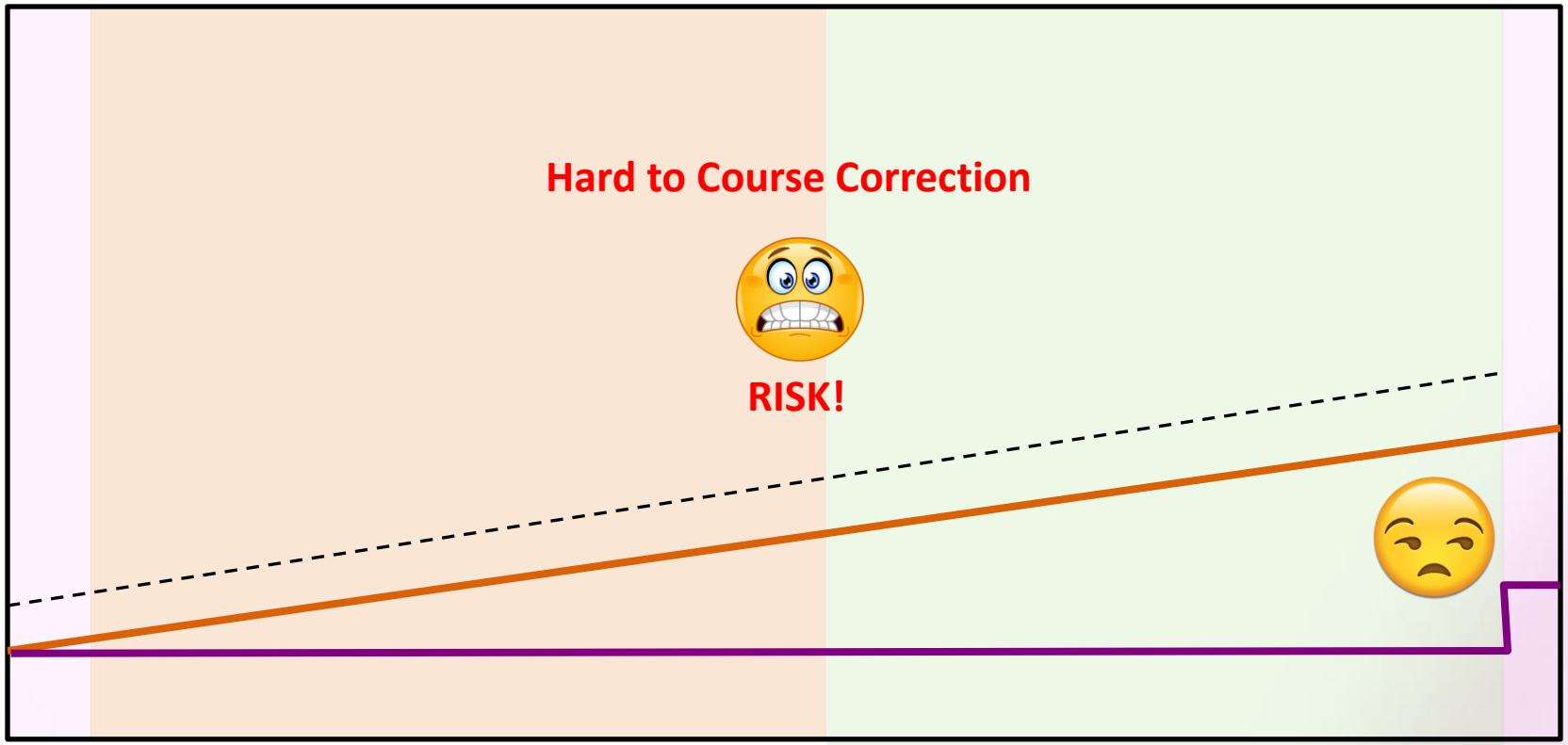
AGILE & EXPRESS ITERATION



PAINFULLY SLOW ITERATION



PAINFULLY SLOW ITERATION



SOFTWARE

!=

BRIDGE BUILDING

VELOCITY

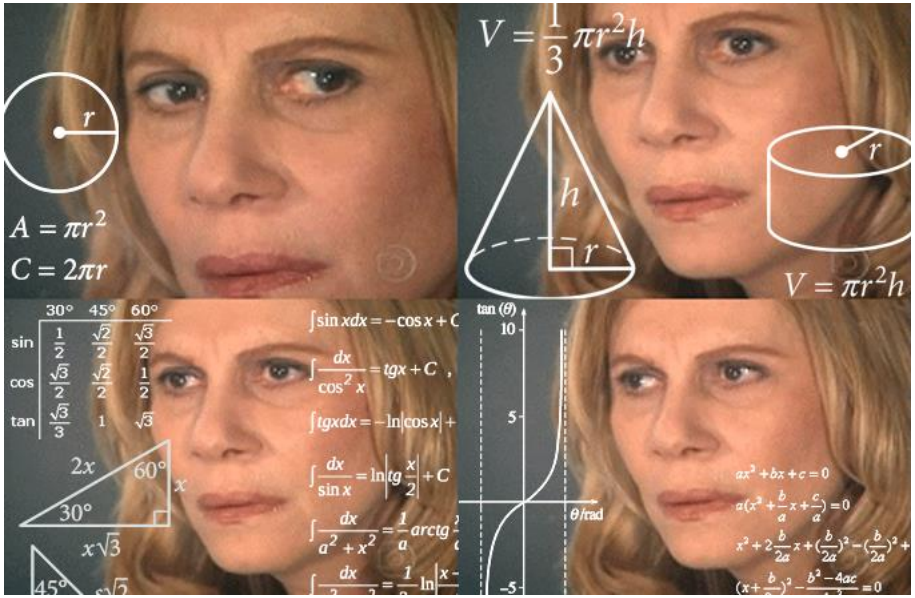
Development

$$\vec{V} = |V| \cdot \hat{V}$$

Development Velocity

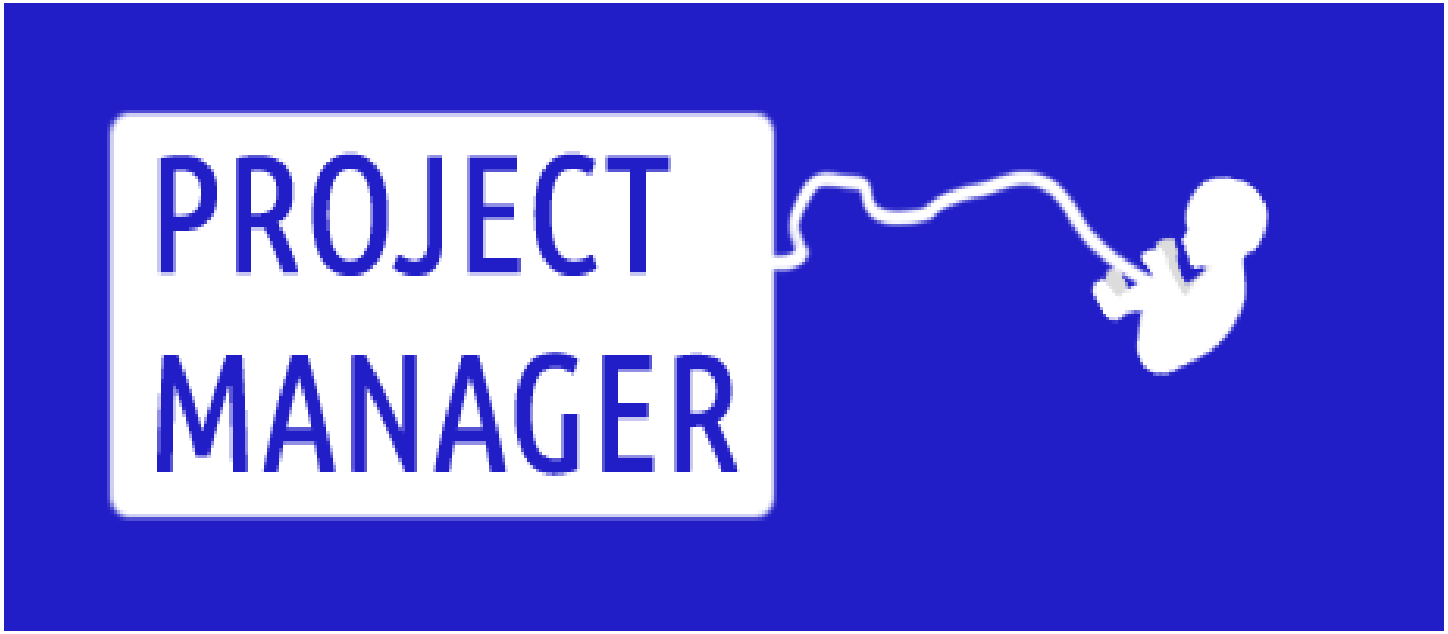
How many features are Delivered

How soon are they delivered



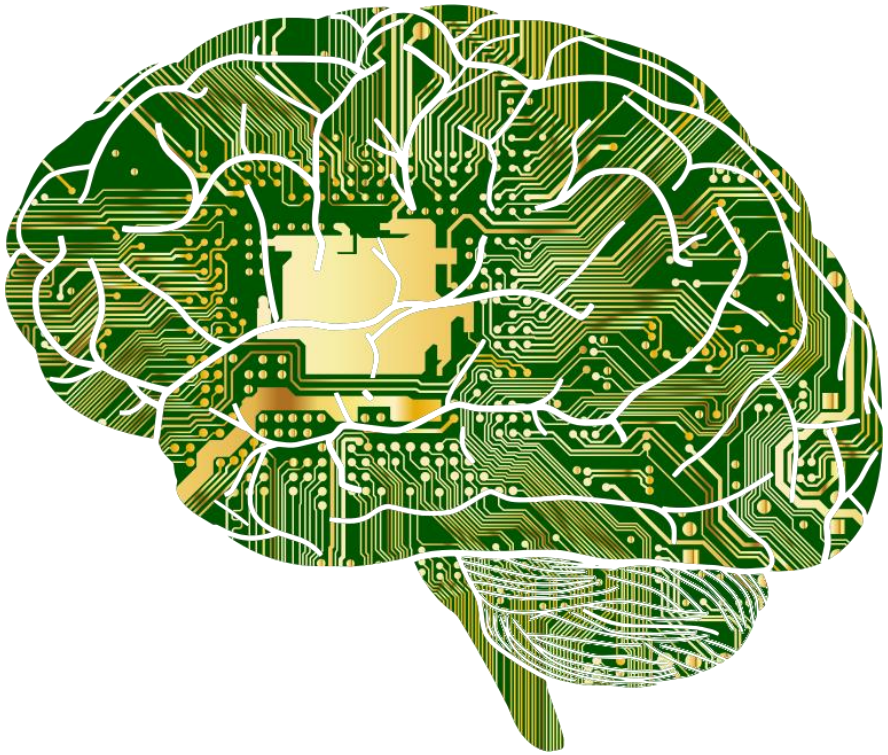
ISN'T IT SIMPLE MATH?

More the people = More the features



Thinks nine women can deliver a baby in one month

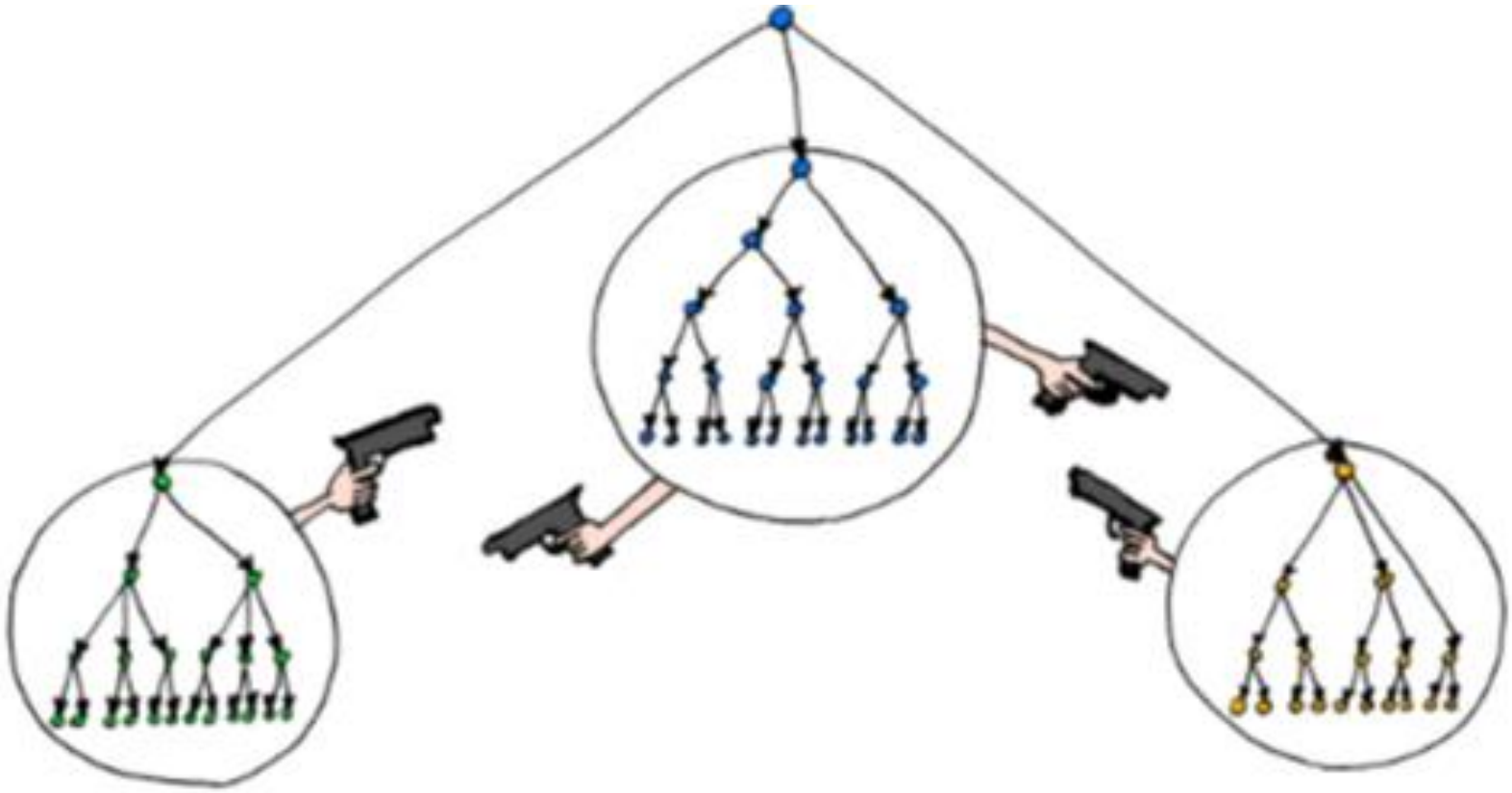
Why art thou a MONOLITH?

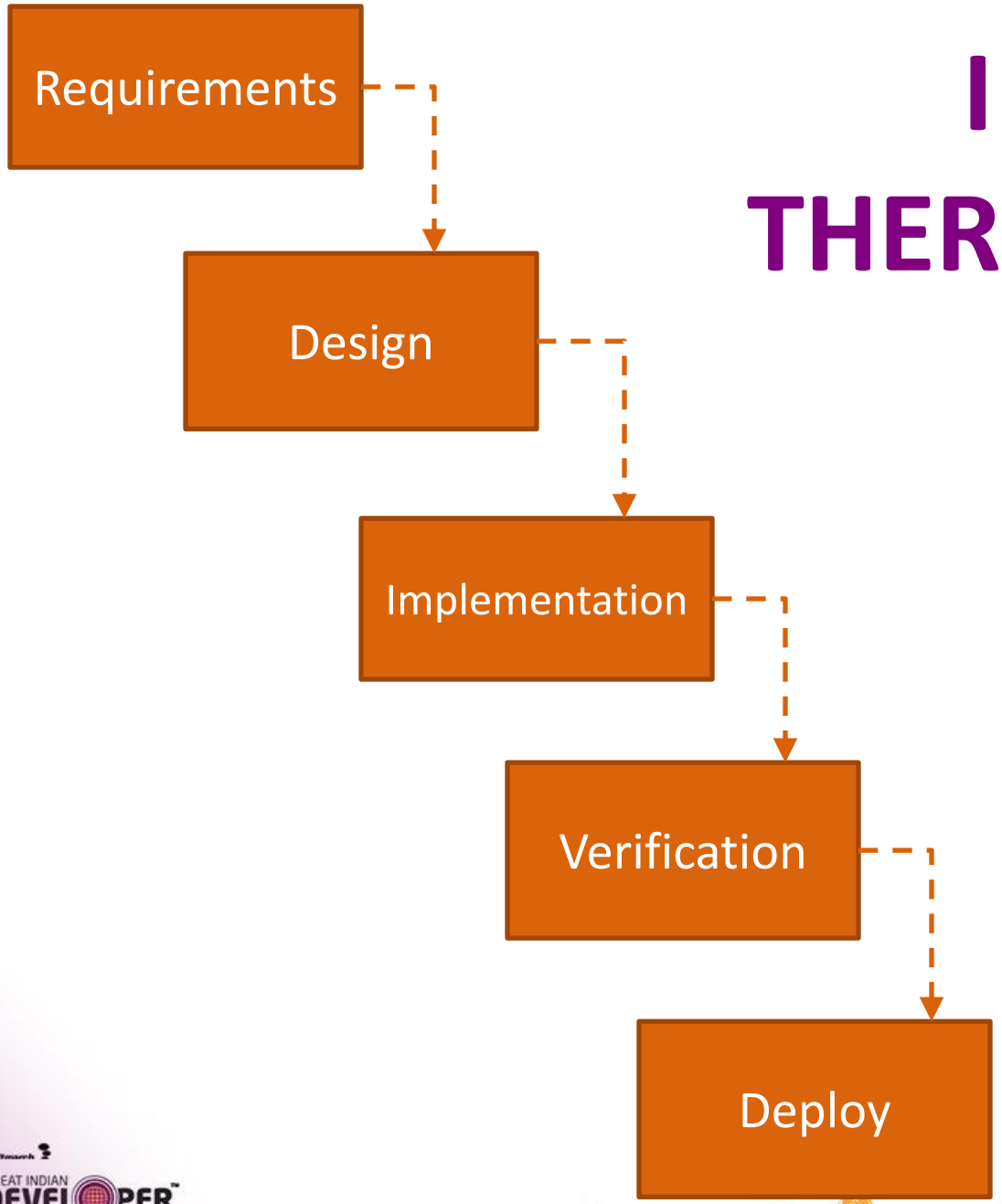


YOU ONLY HAVE ONE BRAIN


ORG STRUCTURE
CONWAY'S LAW
SOFTWARE ARCHITECTURE

CONWAY'S LAW





I THINK, THEREFORE I AM

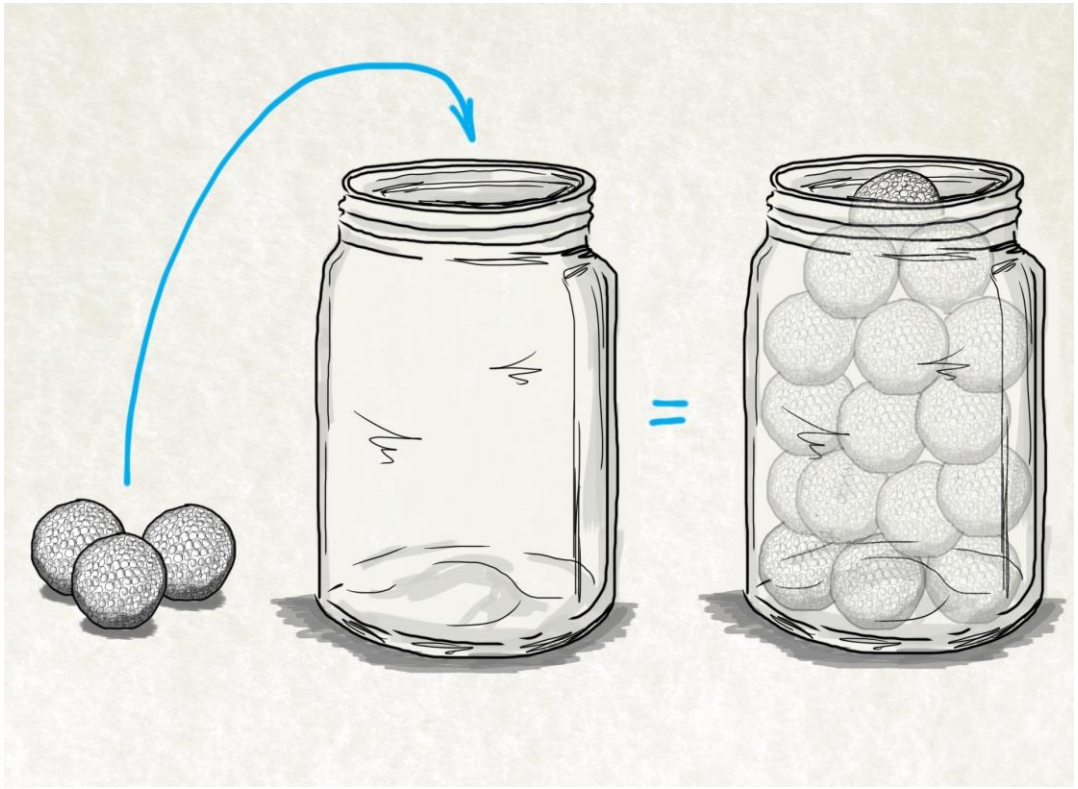
Design Patterns to the rescue!


Design Patterns
DRY
S.O.L.I.D
YAGNI


CAME UP WITH THE
BEST DESIGN EVER
REQUIREMENTS CHANGED
imgflip.com

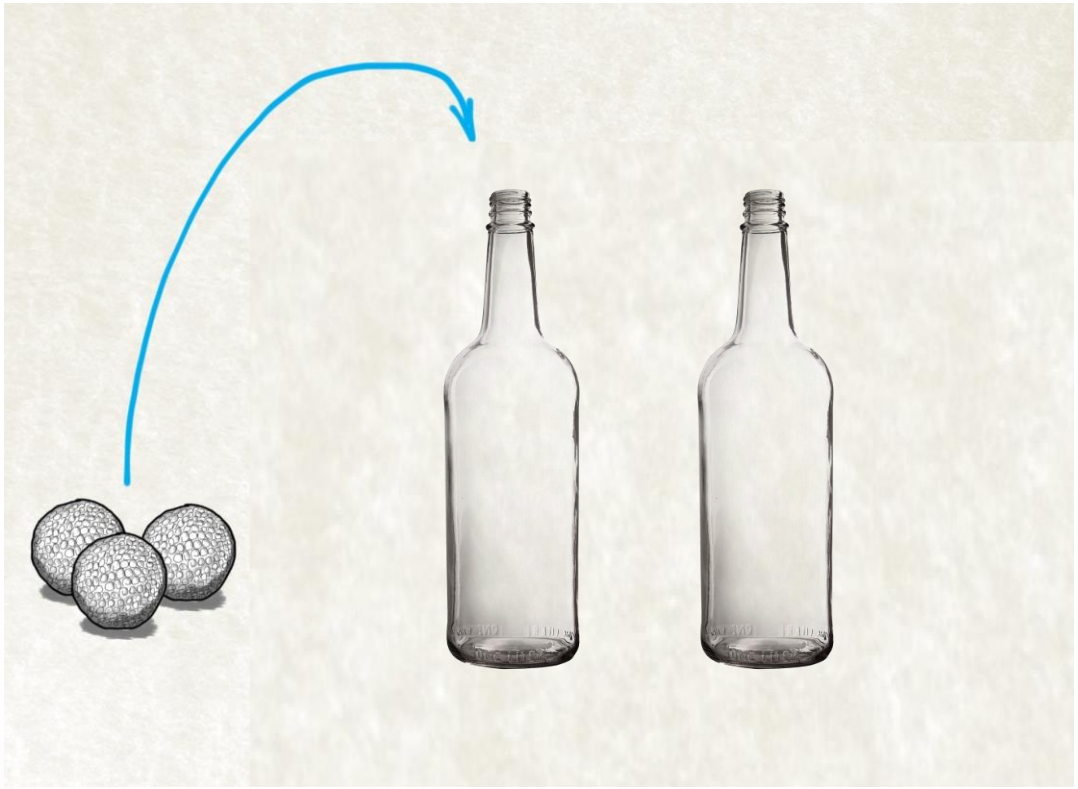
BAD LUCK DEV

BAD LUCK DEV




CAME UP WITH THE BEST DESIGN EVER
REQUIREMENTS CHANGED
imgflip.com

BAD LUCK DEV




CAME UP WITH THE BEST DESIGN EVER
REQUIREMENTS CHANGED
imgflip.com

THERE GOES MY DESIGN !

BAD LUCK DEV

CONSEQUENCES

CONSEQUENCES

Invasive Code Changes

- Functionality
- NFRs
- Design

ISOLATION

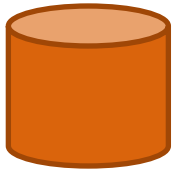
DESIGN

THOUGHT

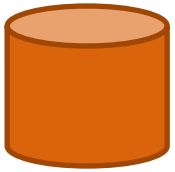
IMPLEMENTATION

ISOLATION

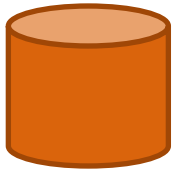
DOMAIN



THOUGHT DESIGN IMPLEMENTATION



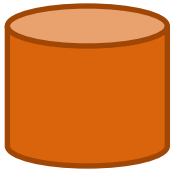
THOUGHT DESIGN IMPLEMENTATION



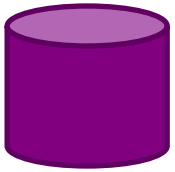
THOUGHT DESIGN IMPLEMENTATION

ISOLATION

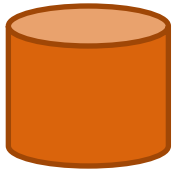
DOMAIN



THOUGHT DESIGN IMPLEMENTATION

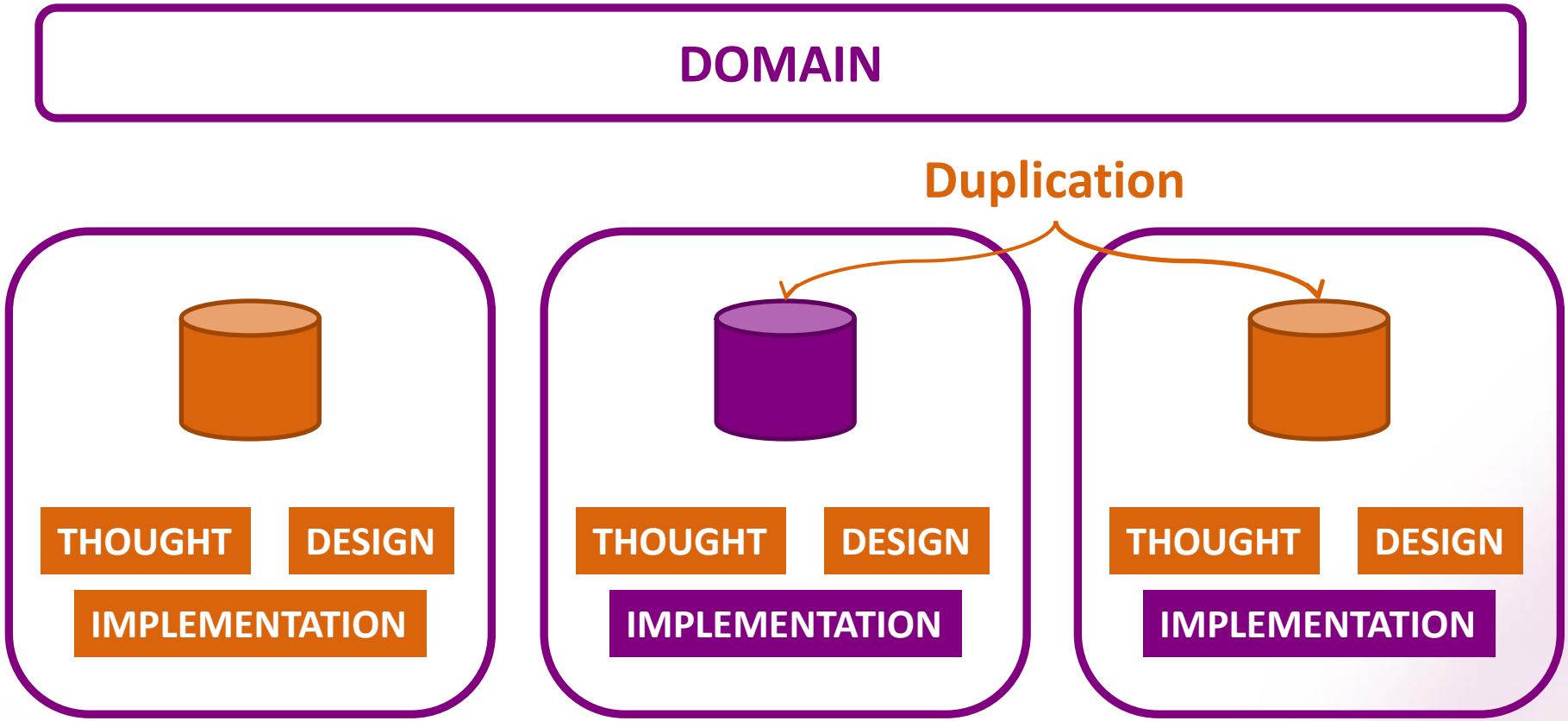


THOUGHT DESIGN IMPLEMENTATION



THOUGHT DESIGN IMPLEMENTATION

ISOLATION


DOMAIN
Duplication
THOUGHT
DESIGN
IMPLEMENTATION
THOUGHT
DESIGN
IMPLEMENTATION
THOUGHT
DESIGN
IMPLEMENTATION

DRY ?

DO NOT REPEAT YOURSELF

DRY ?

DO REPEAT YOURSELF

Lambda

Project Jigsaw

Functional Programming

JAVA IS DEAD,
LONG LIVE JAVA

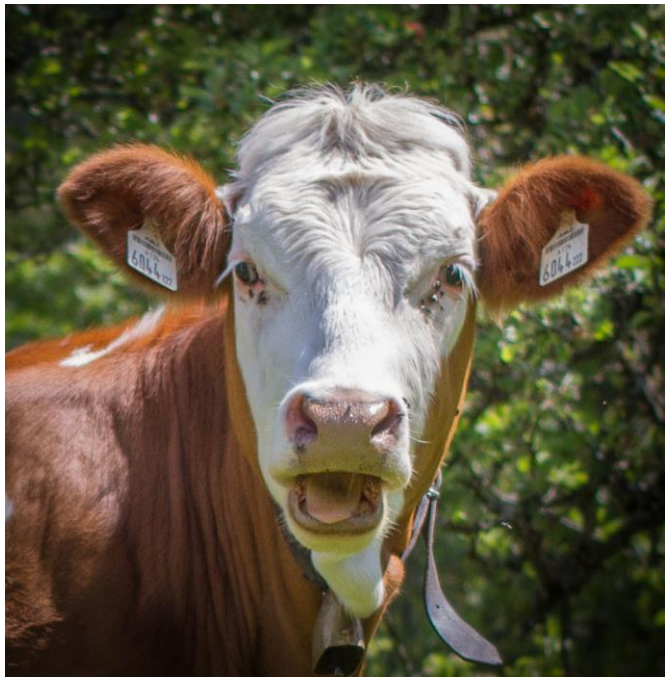
Thread Mutex Asynchronous

Synchronized Non-Blocking

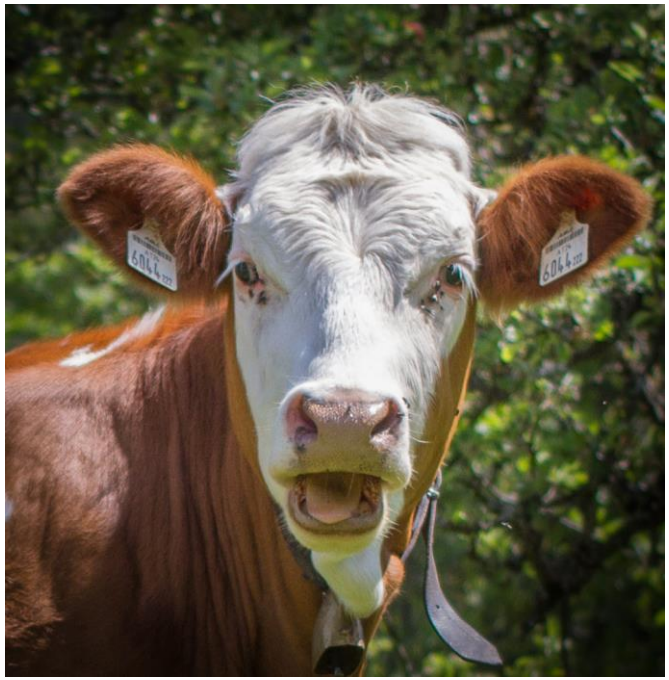
Joins & Waits Immutable

Reactive

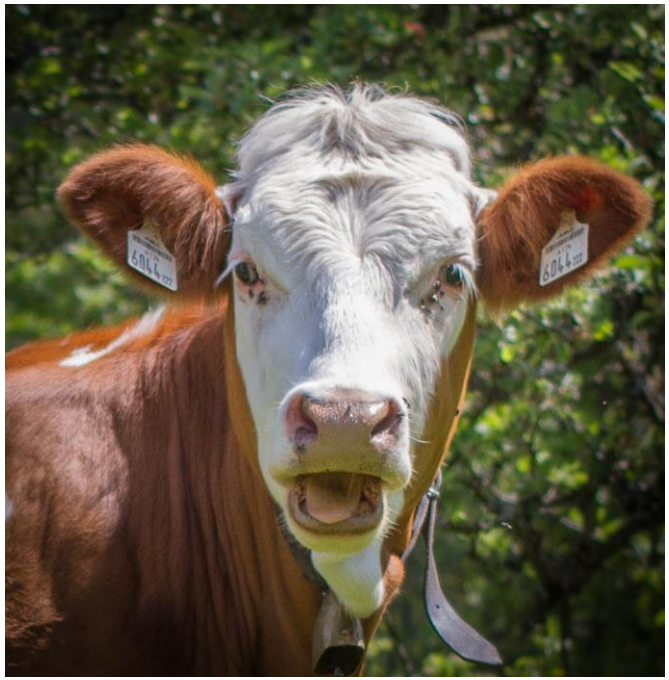
We can build enterprise apps without OOPs



We SHOULD build enterprise apps without OOPs



We can build enterprise apps without OOPs



SOFTWARE ENGINEERS

==

ENGINEERS

EOL OF RIGID STRUCTURES

SQL -> NOSQL

NORMALIZATION -> DENORMALIZATION

OOPS -> FUNCTIONAL

DRY -> DRY

MONOLITH -> MICROSERVICES

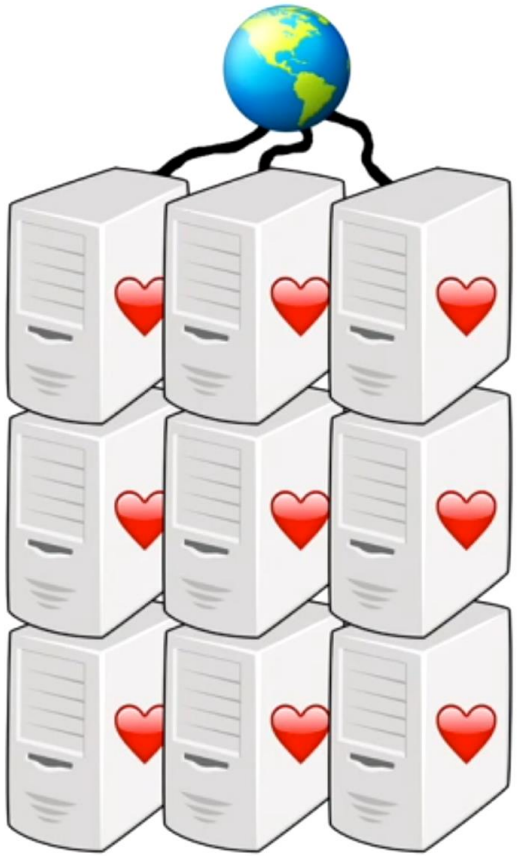
WATERFALL -> AGILE

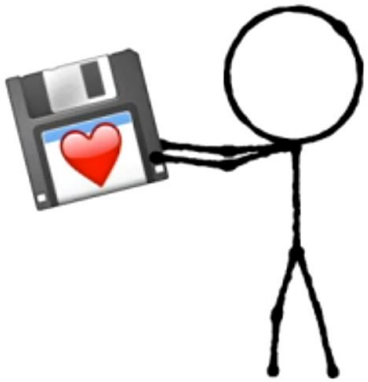
Low Cost to Change

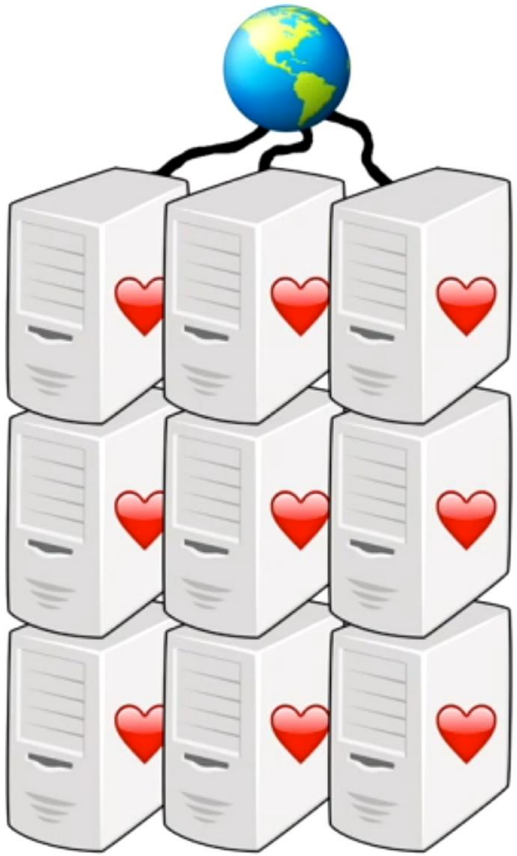
Low Cost to Distribute

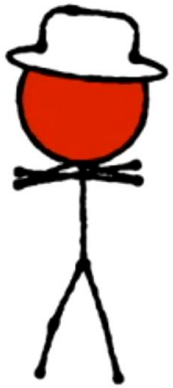
Low Cost to Change

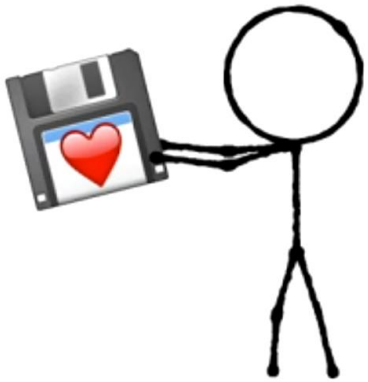
Low Cost to Distribute?

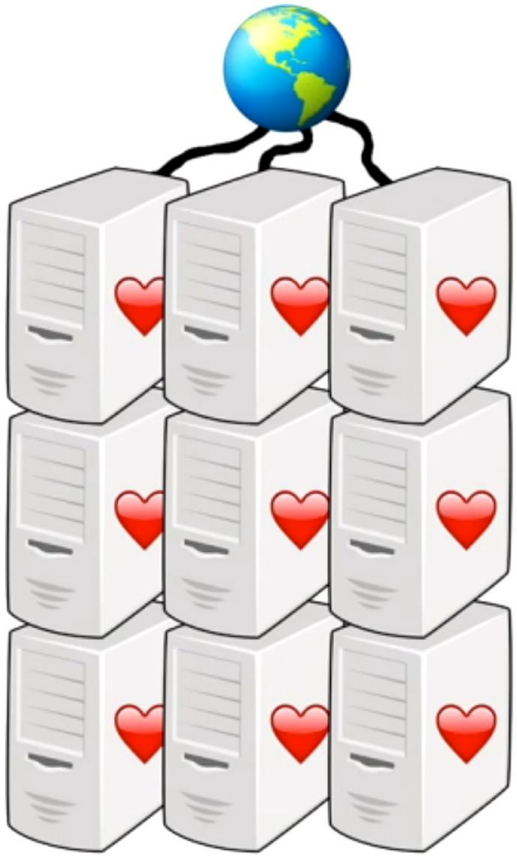


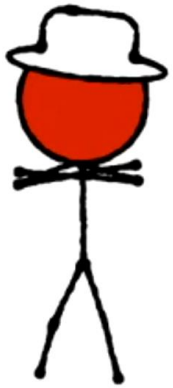


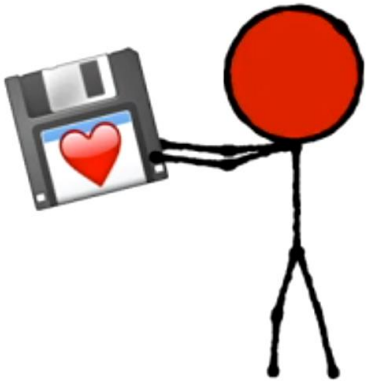


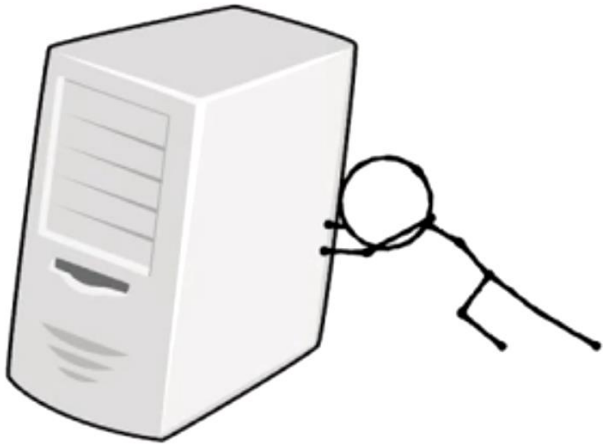


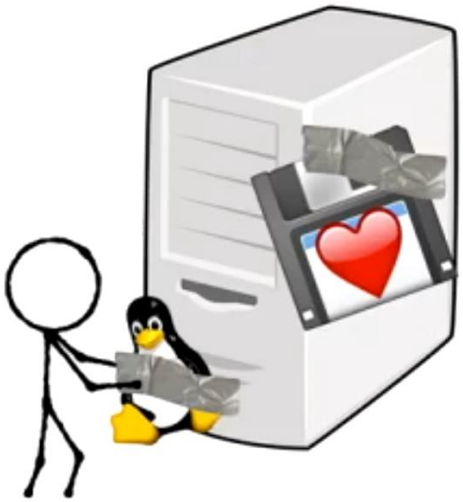


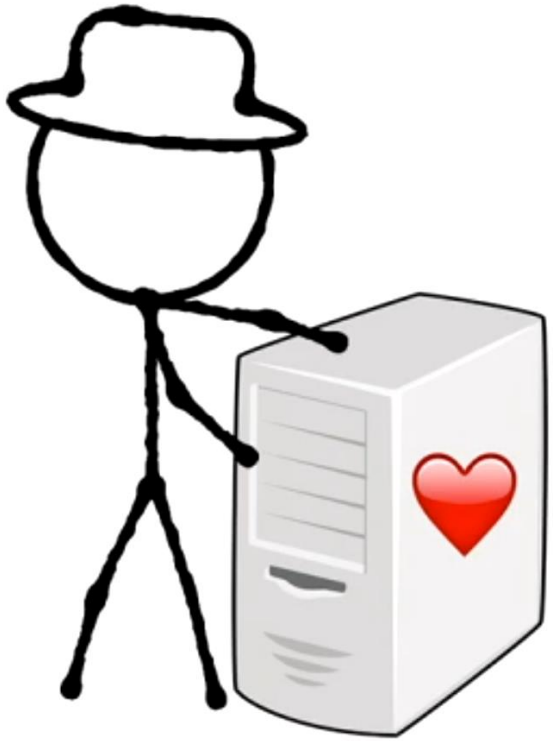












VELOCITY

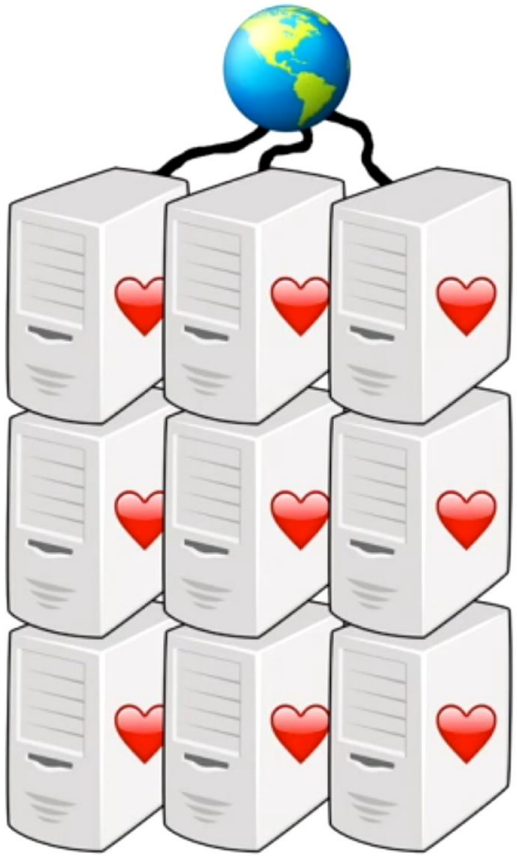
Deployment

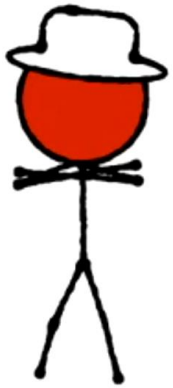
$$\vec{V} = |V| \cdot \hat{V}$$

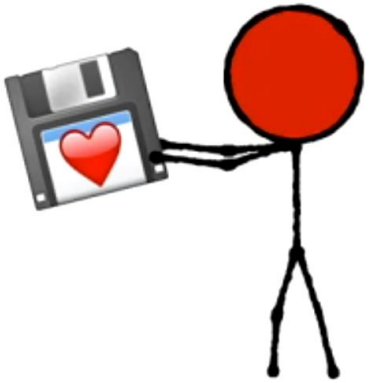
Deployment Velocity

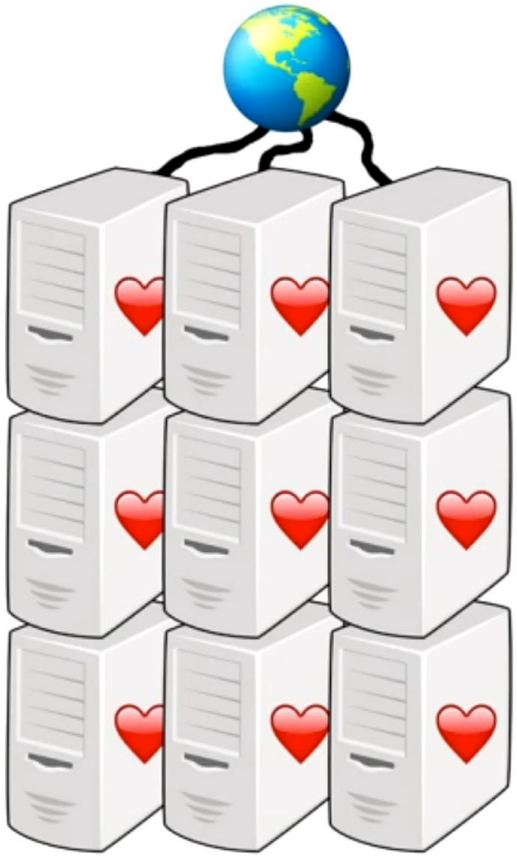
Deploy software without downtime

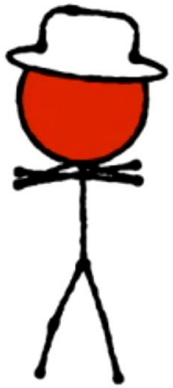
Again and again and again…









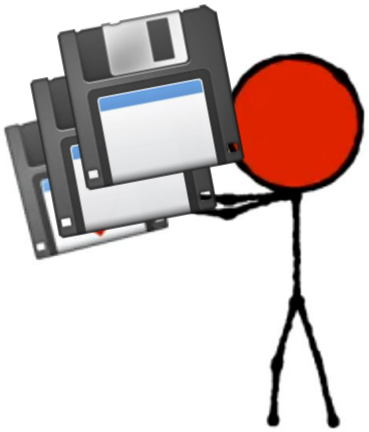


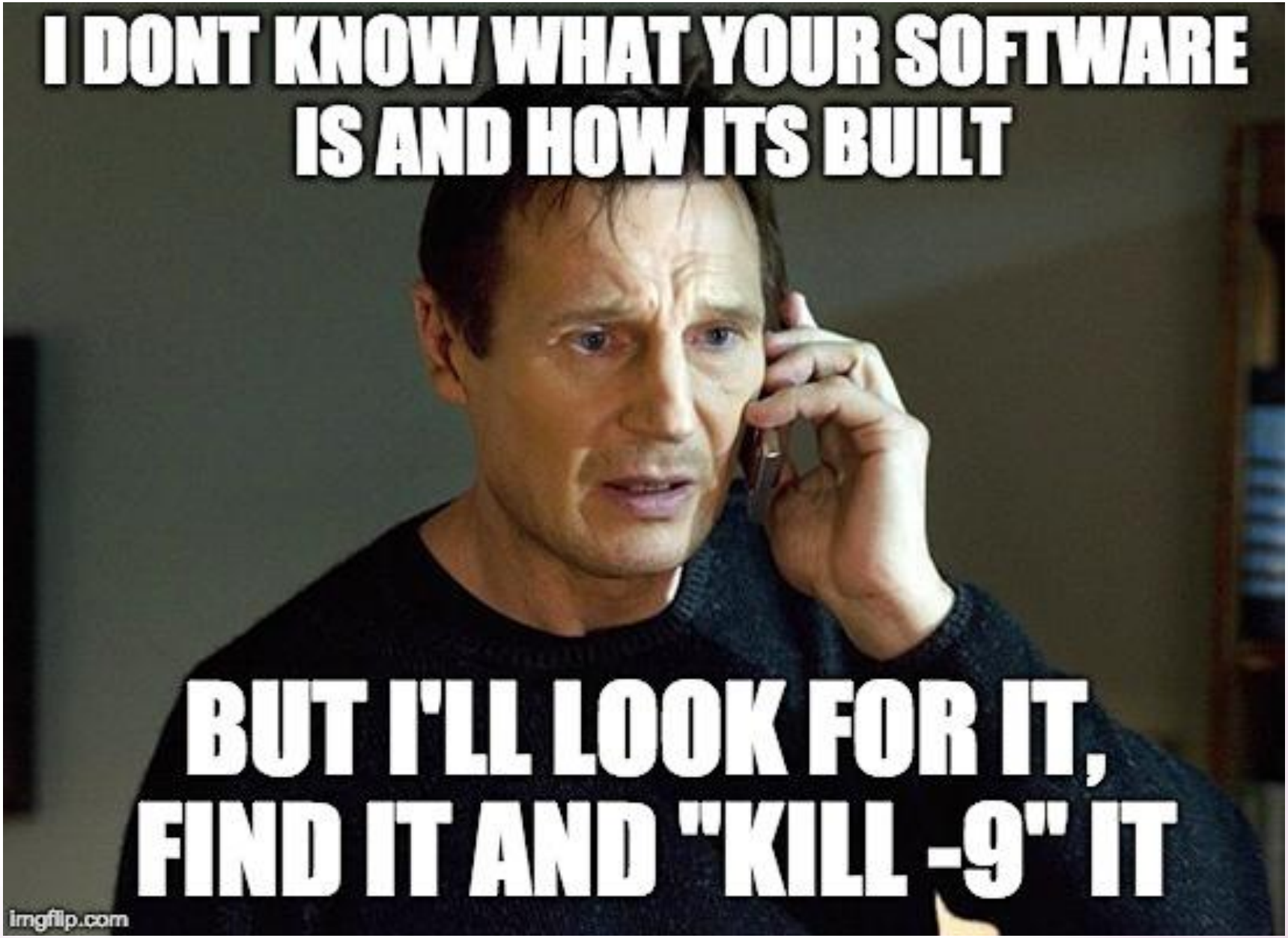
debian

Couchbase

redis

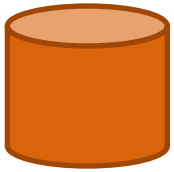
elasticsearch



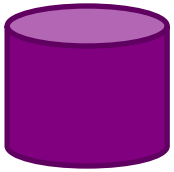

I DONT KNOW WHAT YOUR SOFTWARE IS AND HOW ITS BUILT
BUT I'LL LOOK FOR IT, FIND IT AND "KILL -9" IT
imgflip.com

ISOLATION

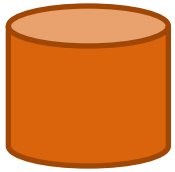
DOMAIN



THOUGHT DESIGN IMPLEMENTATION



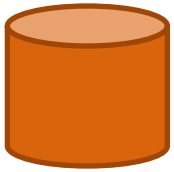
THOUGHT DESIGN IMPLEMENTATION



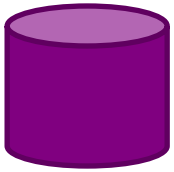
THOUGHT DESIGN IMPLEMENTATION

ISOLATION

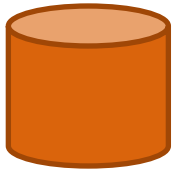
DOMAIN



THOUGHT DESIGN IMPLEMENTATION



THOUGHT DESIGN IMPLEMENTATION



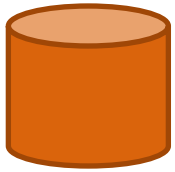
THOUGHT DESIGN IMPLEMENTATION

Give up heterogenous tech stacks

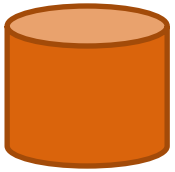
ISOLATION

DOMAIN

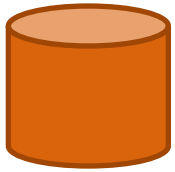
Forsake freedom of choice



THOUGHT DESIGN IMPLEMENTATION



THOUGHT DESIGN IMPLEMENTATION



THOUGHT DESIGN IMPLEMENTATION

Use similar data stores

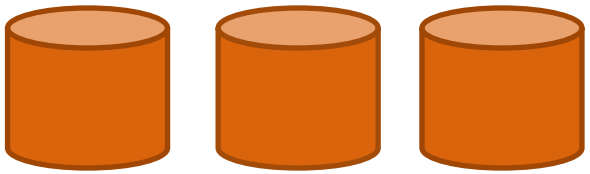
Give up heterogenous tech stacks

ISOLATION

DOMAIN

Forsake freedom of choice

Structures EMERGE



THOUGHT

DESIGN

IMPLEMENTATION

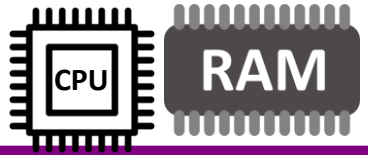
Use similar data stores

Give up heterogenous tech stacks

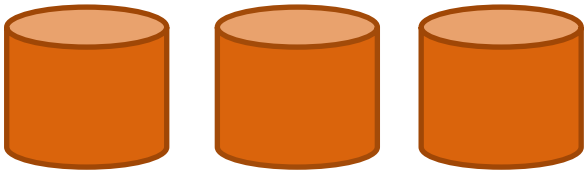
ISOLATION

INSTANCE 1

Forsake freedom of choice

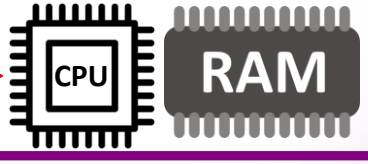


Structures EMERGE



RESOURCE CONTENTION

INSTANCE 2



THOUGHT DESIGN IMPLEMENTATION

THOUGHT DESIGN IMPLEMENTATION

THOUGHT DESIGN IMPLEMENTATION

Use similar data stores

Give up heterogenous tech stacks

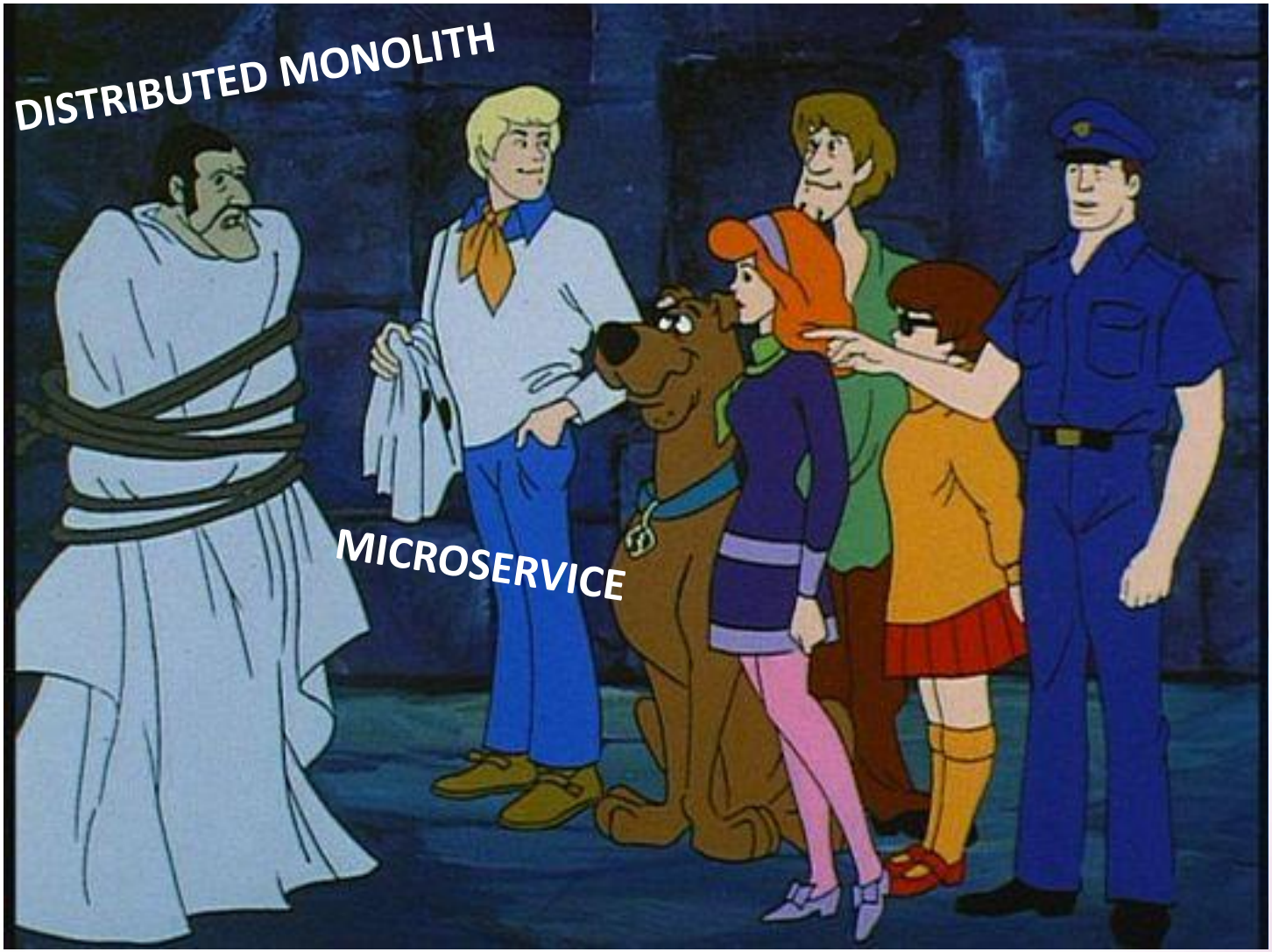
ISOLATION FAILED @ DEPLOYMENT

MICROSERVICES DO NOT WORK

ITS WAY TOO COMPLICATED

ITS JUST LIKE BEFORE

IT WAS YOU ALL ALONG



APP SOFTWARE

Spawn app resources

Scale on need

Continuously monitored

Improved on constant feedback

DEVELOPMENT

INFRASTRUCTURE

Spawn infra resources

Scale on need

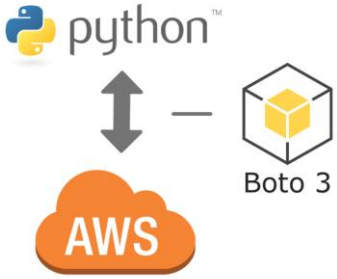
Continuously monitored

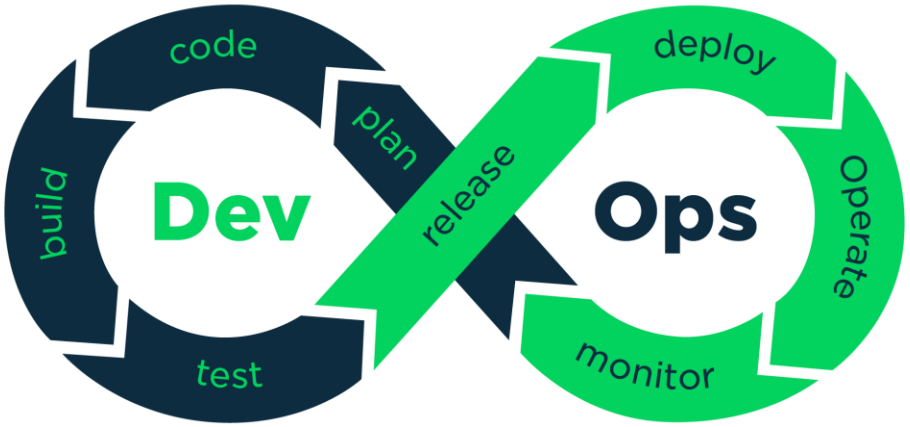
Improved on constant feedback

OPERATIONS

DEVELOPMENT  OPERATIONS

DEVOPS





CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY

CONTINUOUS DEPLOYMENT

Being flexible to change
is the way to succeed in a competitive & unpredictable software economy

THE MARRIAGE OF AGILE, MICROSERVICES & DEVOPS

$$\vec{V} = |V| \cdot \hat{V}$$

Culture Velocity — Build Expertise — Learn & Unlearn

$$\vec{V} = |V| \cdot \hat{V}$$

Development Velocity — How many features are Delivered — How soon are they delivered

$$\vec{V} = |V| \cdot \hat{V}$$

Deployment Velocity — Deploy to live — Again and again

All of us are capable of being a walking software factory

BDD

BRUCE-LEE DRIVEN DEVELOPMENT

BRUCE-LEE DRIVEN DEVELOPMENT

**The stiffest tree is most easily cracked,
while the bamboo ,
survives by bending with the wind.**

~ Bruce Lee

Moving away from rigidity

BRUCE-LEE DRIVEN DEVELOPMENT

Use only that which works, and take it from any place you can find it.

~ Bruce Lee

Picking heterogenous tools

BRUCE-LEE DRIVEN DEVELOPMENT

I fear not the man who has practiced
10,000 kicks once,
but I fear the man who has practiced
one kick 10,000 times.

~ Bruce Lee

Single Responsibility


amc

EMPTY YOUR MIND.

BE FORMLESS. SHAPELESS. LIKE WATER.

YOU PUT WATER INTO A CUP.
IT BECOMES THE CUP.

YOU PUT WATER INTO A BOTTLE.
IT BECOMES THE BOTTLE.

YOU PUT WATER INTO A TEAPOT.
IT BECOMES THE TEAPOT.

WATER CAN FLOW
OR IT CAN CRASH.

BE WATER, MY FRIEND.


As you think,
so you shall be
saltmarch
GREAT INDIAN
DEVELOPER
SUMMIT
2018
References - https://www.youtube.com/watch?v=xdw_9dADM-4


GREAT INDIAN
DEVELOPER™
SUMMIT 2019
Conference : April 23-26, Bangalore
12th EDITION
#gids19
Register early and get the best discounts!
www.developersummit.com
@greatindiandev
bit.ly/gidslinkedin
facebook.com/gids19
bit.ly/saltmarchyoutube
flickr.com/photos/saltmarch/