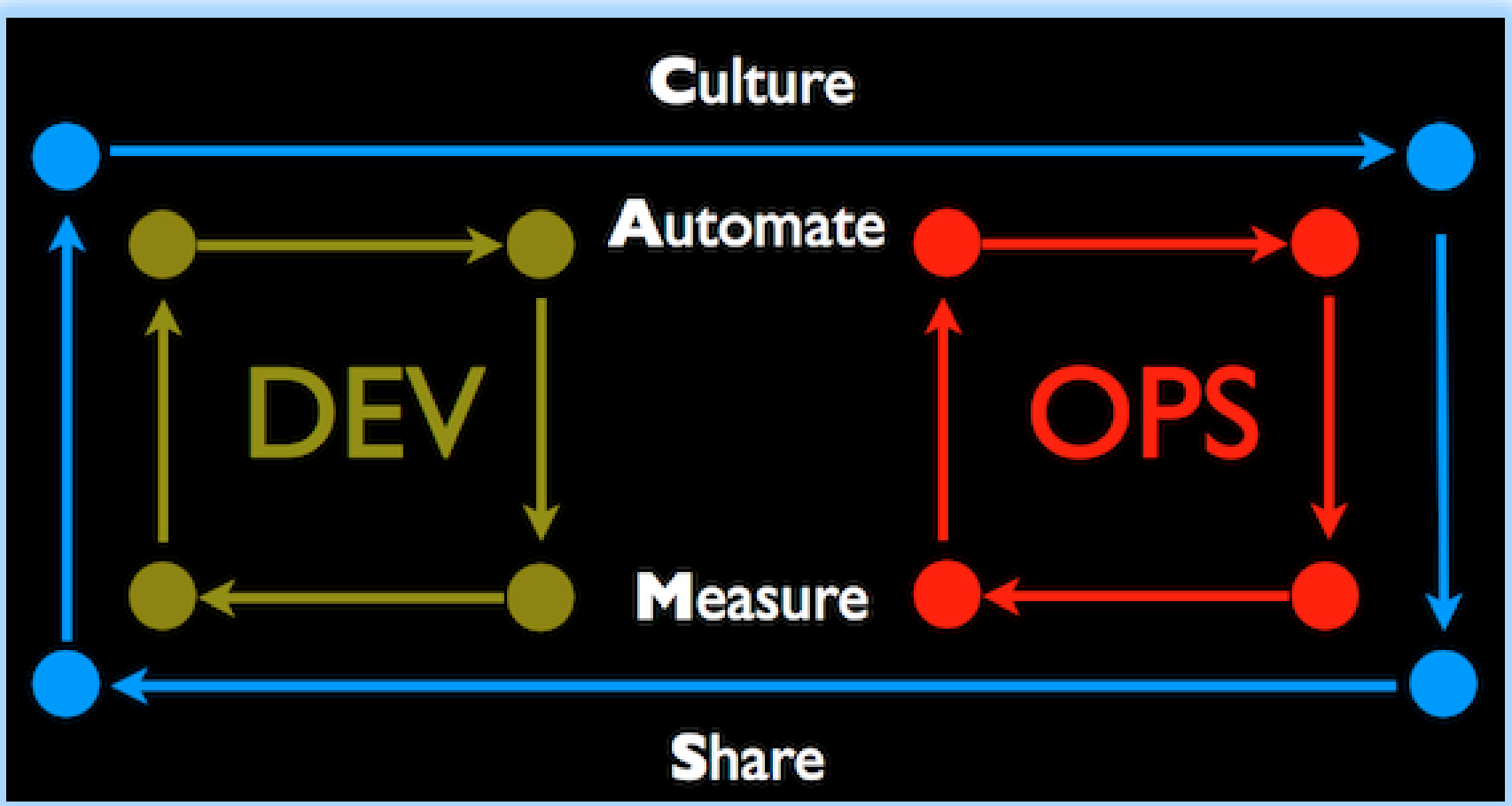
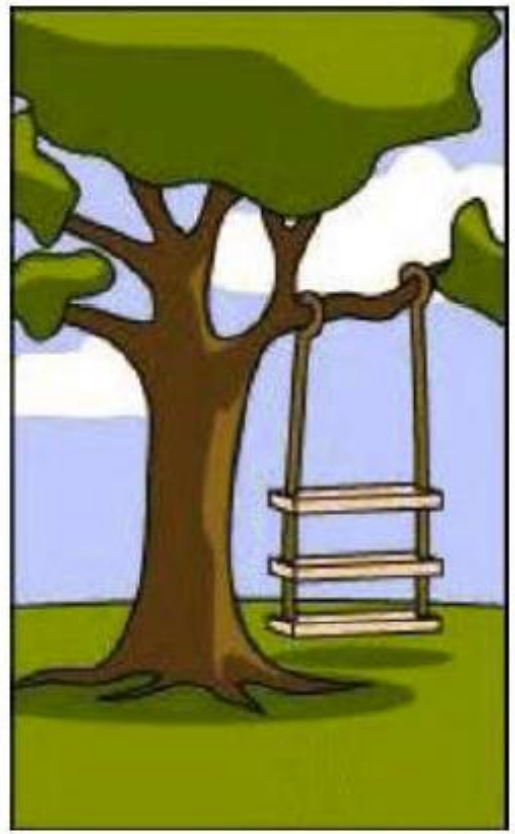
CS 816 – Software Production Engineering



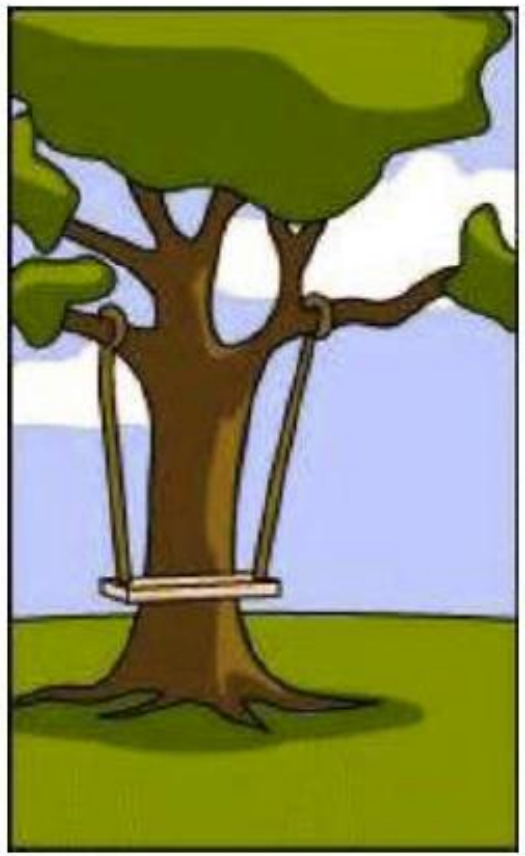
http://www.opscode.com/blog/2010/07/16/what-devops-means-to-me/

A typical software project



How the customer explained it

A typical software project

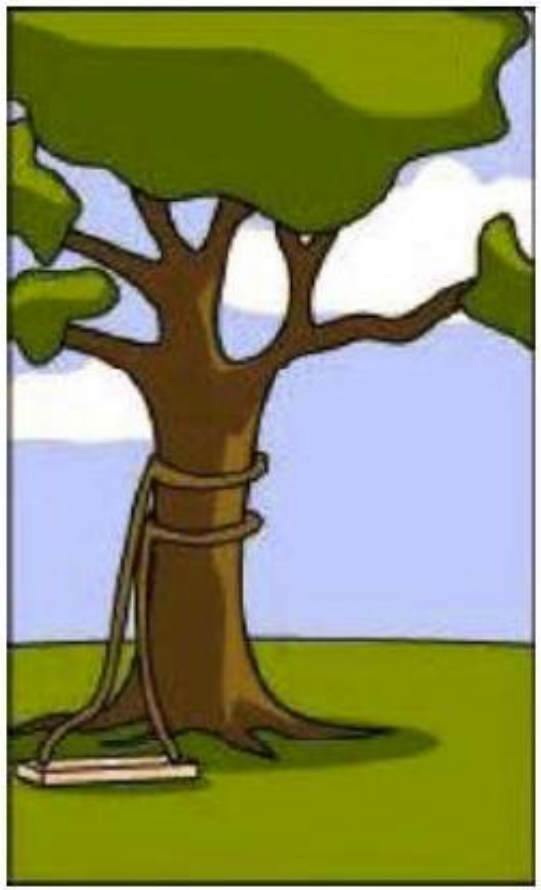


How the project leader understood it

A typical software project

How the analyst designed it

A typical software project

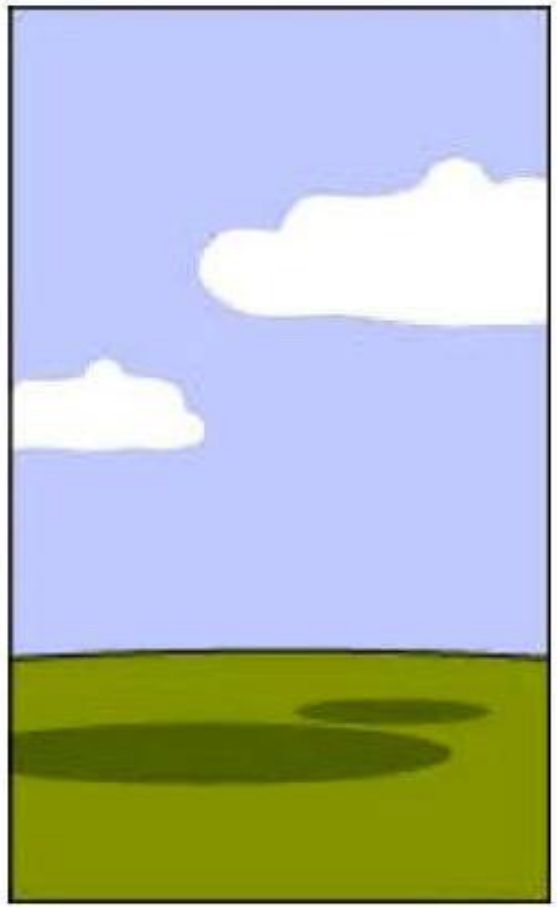


How the programmer wrote it

A typical software project

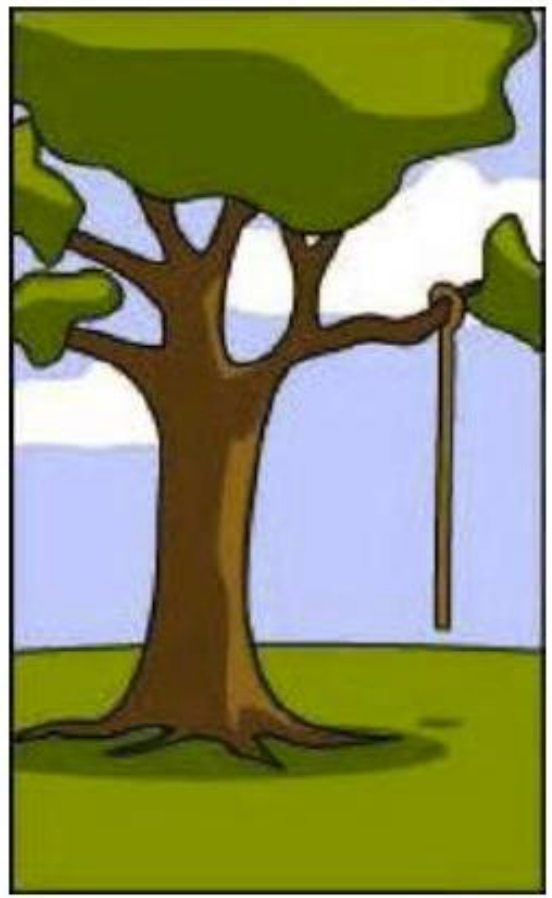
How the business consultant described it

A typical software project



How the project was documented

A typical software project

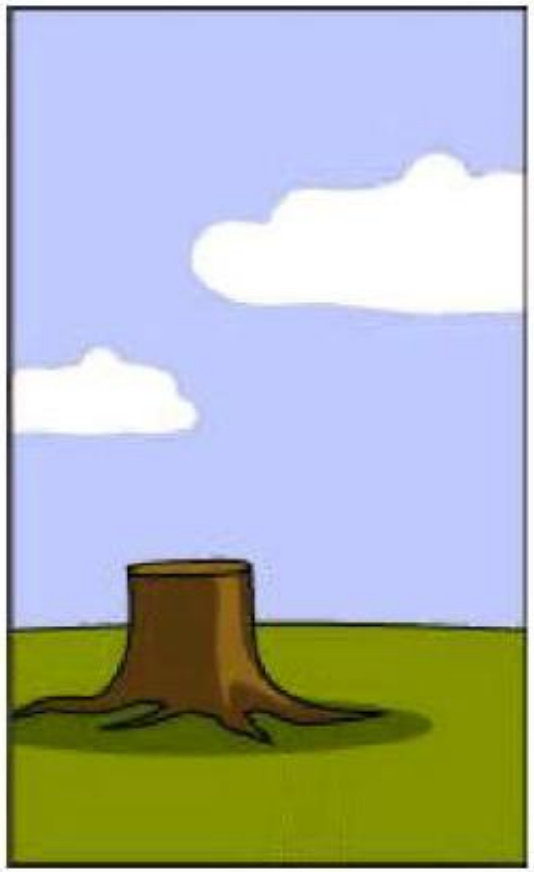


What operations installed

A typical software project

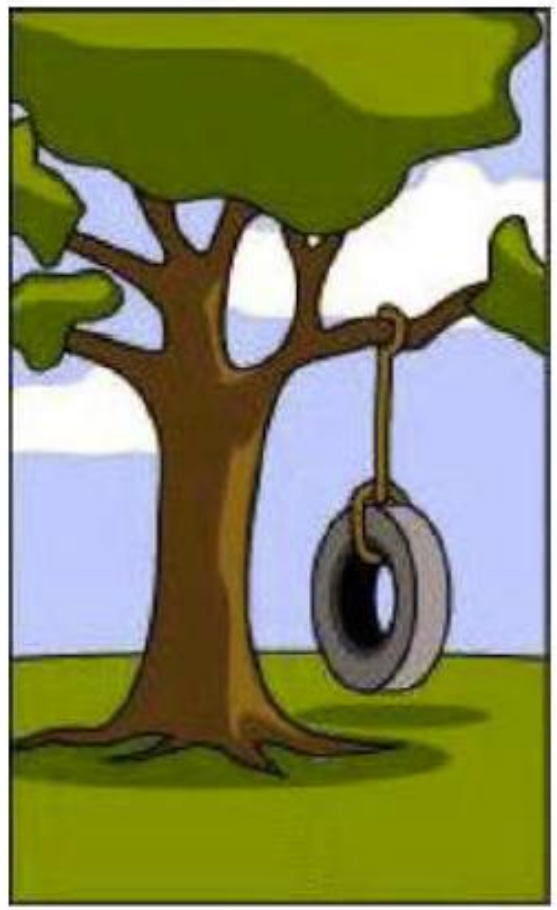
How the customer was billed

A typical software project



How it was supported

A typical software project



Finally, what the customer really needed

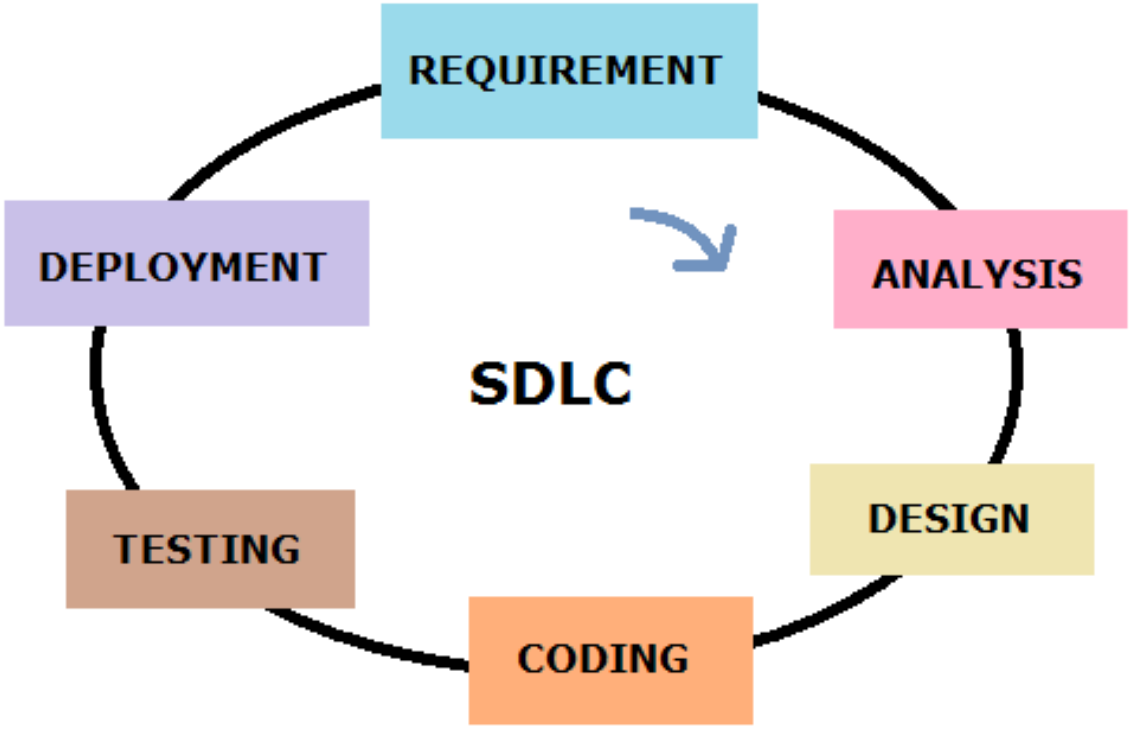
Software Engineering

- Software Engineering - Software engineering encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high quality computer software.

- Domains -Business, science and engineering

- Types of Software - System Software (BSP, OS), OSS, Application Software, Engineering or Scientific Software, Embedded Software, Web Application and AI Software.

- SDLC



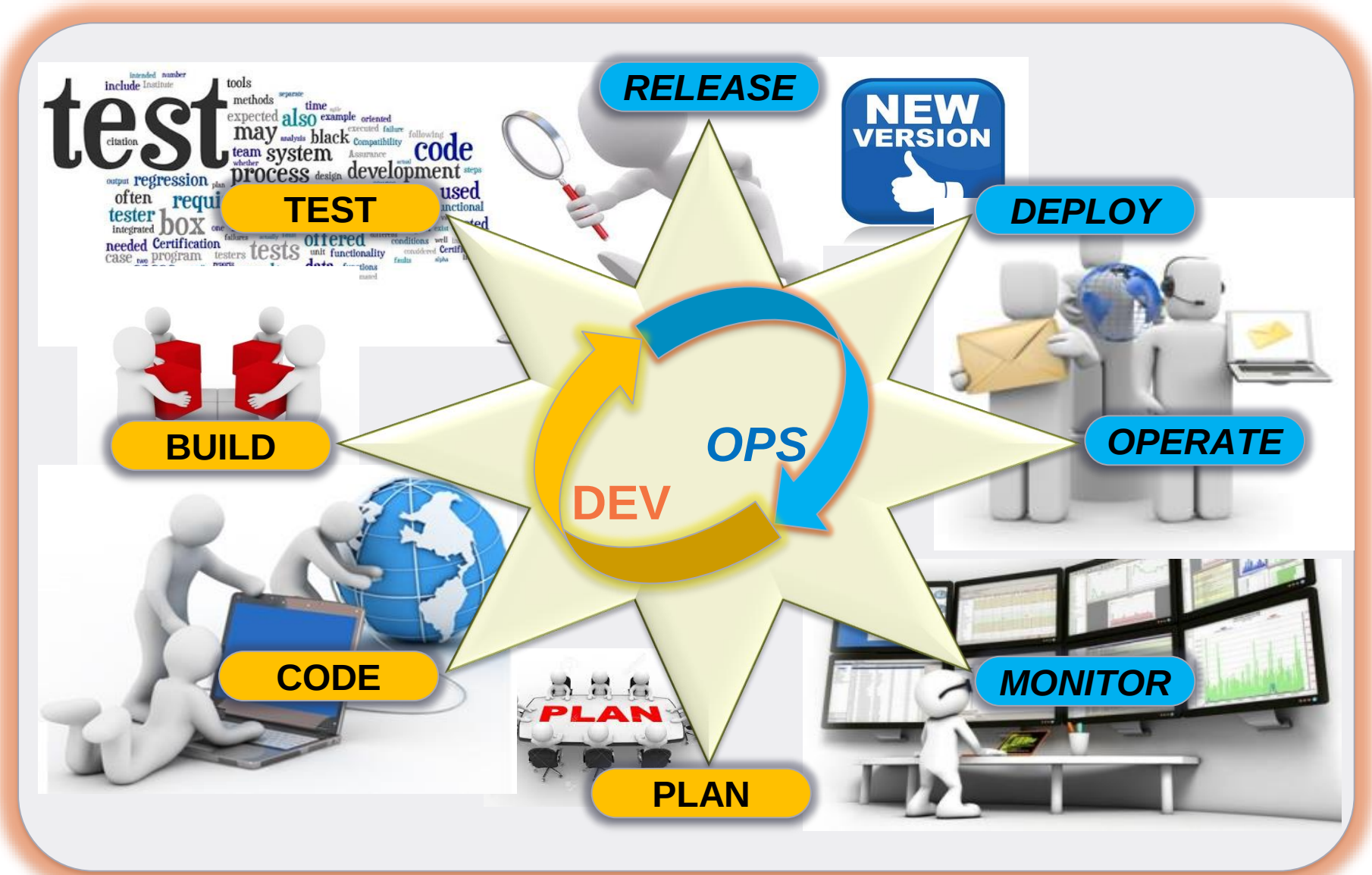
Project Execution Methodologies – The Change

Waterfall

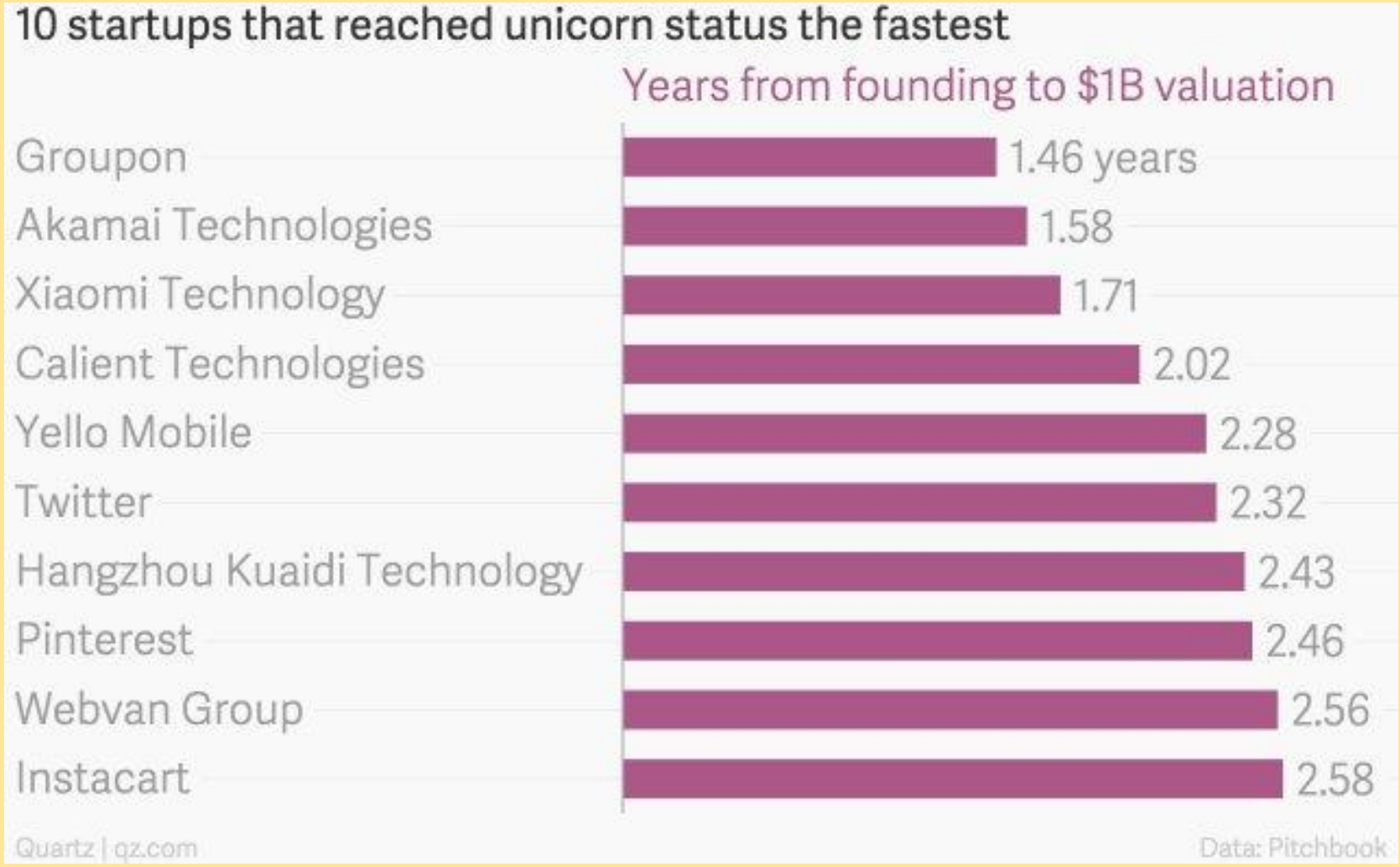
Agile

DevOps

Introduction



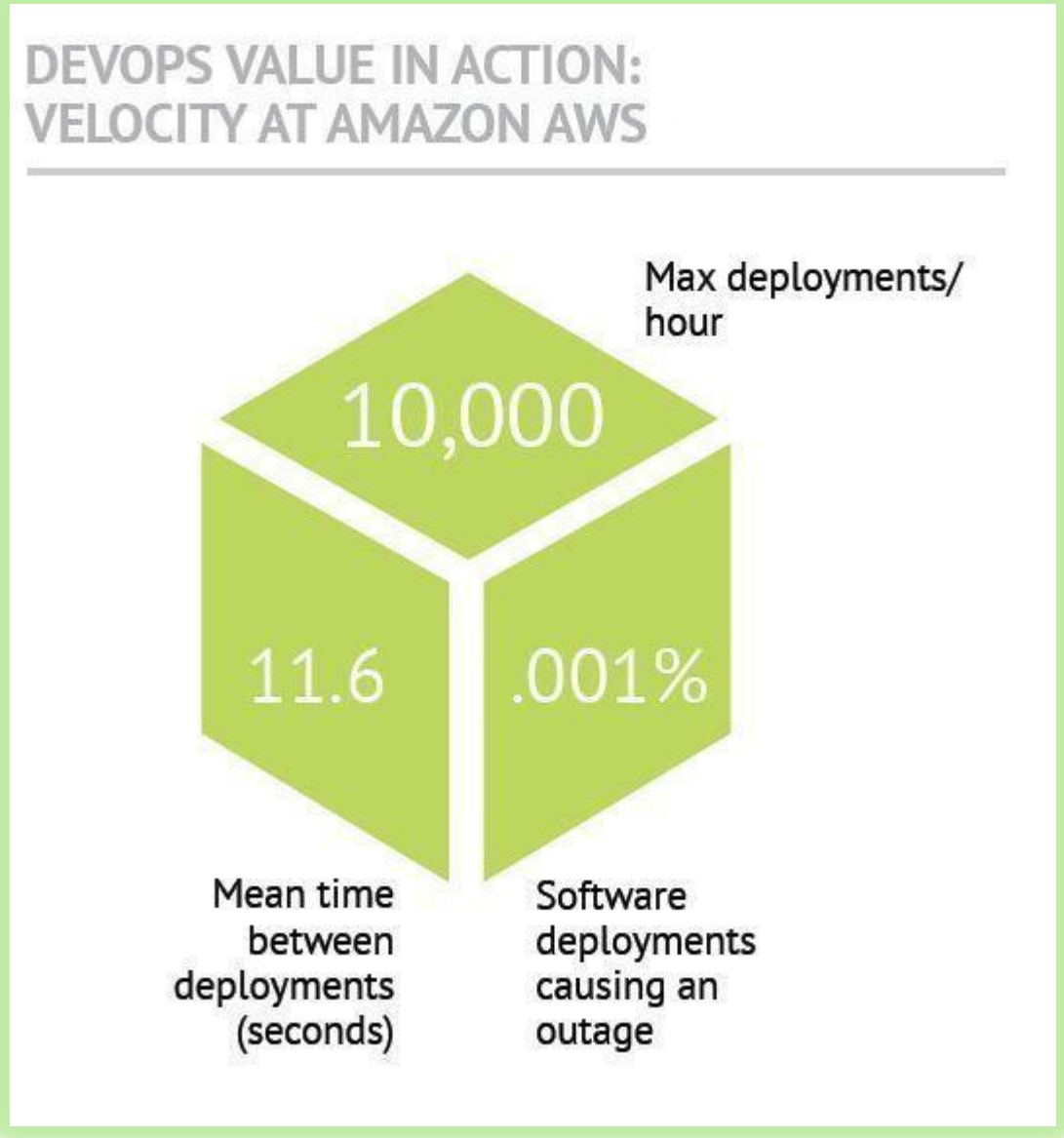
Business Agility



- A **unicorn** is a startup company valued at over $1 billion.
- A **decacorn** > $10 b
- A **hectocorn** > $100 b

DevOps in Action

Company	Deploy Frequency	Deploy Lead Time	Reliability	Customer Responsiveness
Amazon	23,000 / day	minutes	high	high
Google	5,500 / day	minutes	high	high
Netflix	500 / day	minutes	high	high
Facebook	1 / day	hours	high	high
Twitter[2]	3 / week	hours	high	high
typical enterprise	once every 9 months	months or quarters	low/medium	low/medium



Corporate Culture

Development wants to add new features quickly

Operations want stability

Done! In two weeks the development is complete!!

You Wish!! We will take four weeks to get servers and deploy

The biggest business problem in the new fully digitalized world is not the cost of IT operations or the DevOps team – it is the lost opportunity on not executing your business service delivery with enough quality and speed compared with the new breed of competitors attacking your business segment.

Business Agility – Typical Scenario

Businesses are under competitive pressure, narrow GTM window

Customer's are loving our competitor's apps for those new features, I need the same on our app by next 30 days

Give me cloud, give me chef, give me knife, give me hammer, and throw Ops team out

Challenges to IT Agility

1. Everything needs software.
2 Software runs on a server to become a service
3. Delivering a service from inception to its users is too slow and error-prone
4. There are internal friction points.
5. Defects released into production.
6. Inability to diagnose production issues quickly
7. Problems appear in some environments only; Blame shifting or finger pointing
9. Long delays while dev, QA, ops team waits on resource/response from other teams.
10. Releases slip or fail - Delay causes money loss.
11. Traditional Thinking: Conflicting Perspectives and Lesser Collaboration

Dev's job is to add new features; Ops' job is to keep the site stable and fast.

Brief History of DevOps

- In 2009, at the O'Reilly Velocity Conference, two Flickr employees—John Allspaw, senior vice president of technical operations, and Paul Hammond, director of engineering—deliver a seminal talk known as "10+ Deploys per Day: Dev and Ops Cooperation at Flickr." (Flickr is an image- and video-hosting website and web services suite)
- The talk is an energetic presentation in which John and Paul act out the classic "finger point" problem of Dev versus Ops. Whey they **brought together to a roomful of developers and operations teams. The usual blame was "It's not my code, it's your machines!" or it is not my machine, it is your code created the problem.**
- They explained that the only sensible way to build, test and deploy workable new software is to make development and operations transparent and integrated.
- The talk becomes widely credited with showing the world what development-operations collaboration can achieve.
- Viewing the presentation from Belgium via streamed video, Debois who is an independent IT consultant inspired to organize his own conference, called Devopsdays in Belgium.

Ref: https://www.ca.com/us/rewrite/articles/devops/a-short-history-of-devops.html

Brief History of DevOps

- DevOps emerged from an effort by businesses to respond more rapidly to market changes.
- It formed out of a fundamental need and is based on a simple philosophy—business works best, when efforts are coordinated and collaborative—so its growth has been rapid.
- The DevOps approach was designed to ensure that high-quality updated software gets into the hands of users more quickly. Though the DevOps is only a few years old, the DevOps movement is so widespread.

DevOps Basics

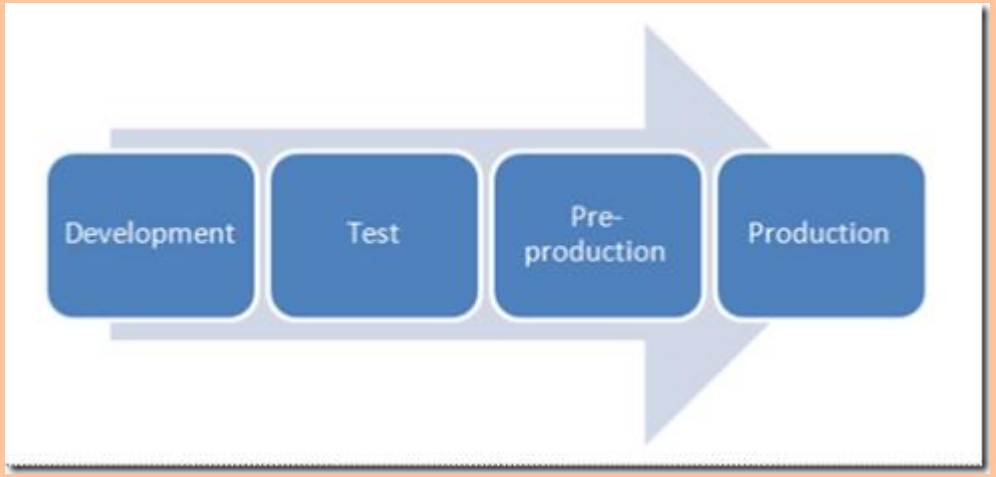
- **Dev** – People involved in developing product.
- **Ops** – System engineers, administrators, operations staff, release engineers, DBAs, network engineers and Security professionals.
- **Agile** Software Development – collaboration of customers, product management, developers and QA to fill in the gaps and rapidly iterate towards a better product.
- **DevOps** – extending Agile principles beyond the boundaries of “the code” to the entire delivered service.
- **Goal** of DevOps – span the entire delivery pipeline.
  - Improved deployment frequency
  - faster time to market
  - lower failure rate of new release
  - shortened lead time between fixes
  - faster mean time to recovery in the event of a new release crashing.

lead time - the time elapsed between the identification of a requirement and its fulfillment.

Mean time to failure is the length of time a system or application is expected to last in operation.

SDLC Journey

- **Goal is to make this journey smooth, quick and without hiccups.**
- **What is constant and variable in this SDLC Journey?**



- As the development team reaches code complete, then deploy the code to the test cluster where the QA team goes through a test pass.
- After meeting the test pass exit criteria, roll the code into an intermediary staging environment, which is called “Pre-production”.
- This staging environment matches our production hardware as close as possible and use it to do final acceptance testing.
- This model, helps ensure the highest quality release when finally deploy to Production.

SDLC

Each department defines its goals based on the division of work. Development and operations are two distinct departments. Often, these departments act like silos because they are independent of each other. The development department may be measured by its speed in creating new features, whereas the operations department may be judged by server uptime and application response time.

Development team struggle for change, whereas operations teams struggle for stability. The conflict between development and operations is caused by a combination of conflicting motivations, processes and tooling, as a result, isolated silos evolve.

In a nutshell, the conflict between development and operations is :

1. **Need for change**: Development produces changes (e.g: new features, bug fixes and work based on change requests). They want their changes rolled out to production.
2. **Fear of change**: Once the software is delivered, the operations department wants to avoid making changes to the software to ensure stable conditions for the production systems.

Agile Methodology

- Developers primarily focus on accelerating the creation of new features by, for instance, adopting Agile methodologies. The Agile movement has brought together programmers, testers, and business representatives.
- Conversely, operations teams are isolated groups that maintain stability and enhance performance by applying practices such as the Information Technology Infrastructure Library (ITIL).
- The principles of Agile methods are focused on defining, building and constructing software.
- DevOps helps development teams increase the frequency of application updates. Agile adoption traditionally omits operations, obstructing the delivery pace.
- DevOps removes this obstruction by applying agile values to the deployment, environment configuration, monitoring and maintenance tasks.

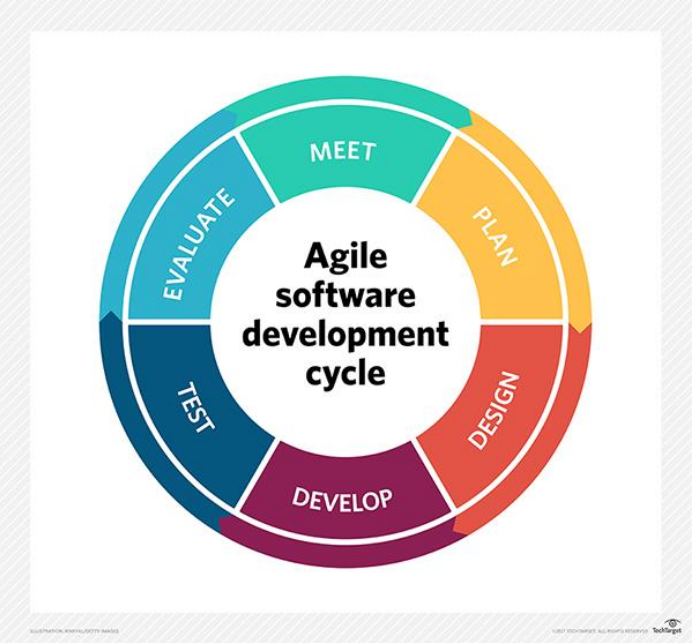
ITIL

The Information Technology Infrastructure Library (ITIL) is a set of detailed practices for IT activities such as IT service management and IT asset management that focus on aligning IT services with the needs of business.

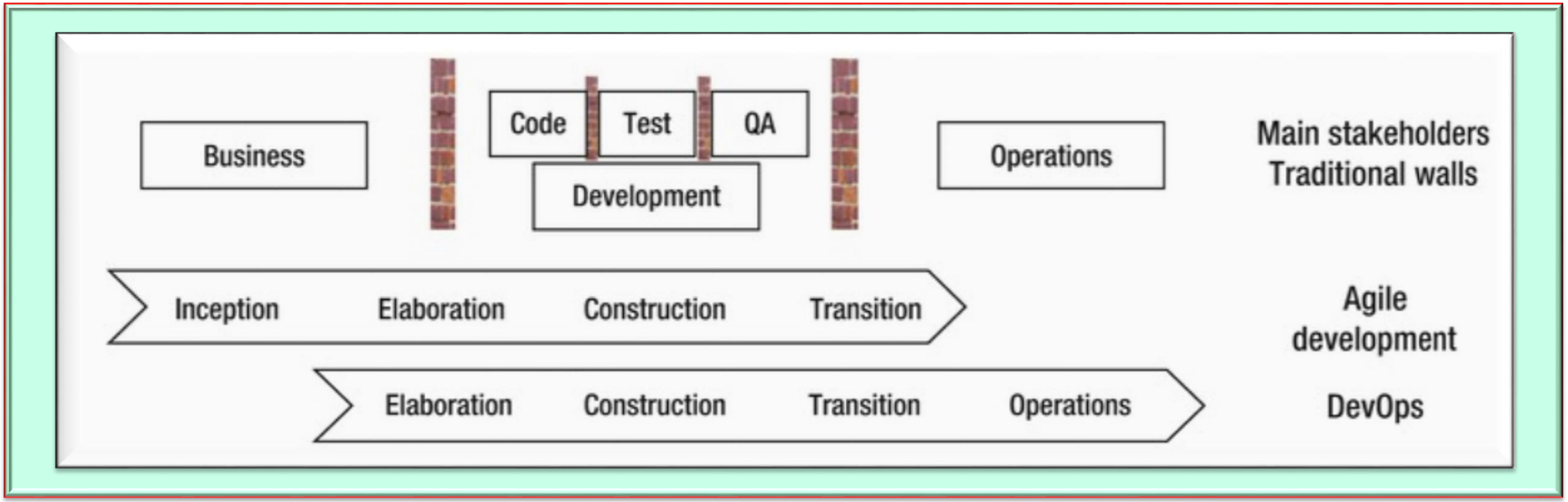
Management Practice	Purpose
Availability management	To ensure that services deliver agreed levels of availability to meet the needs of customers and users
Capacity and performance management	To ensure that services achieve agreed and expected performance, satisfying current and future demand in a cost-effective way
Change control	To maximize the number of successful changes by ensuring that risks have been properly assessed through to managing the change schedule
Continual improvement	To align practices and services with changing business needs through the ongoing improvement of products, services, and practices
Deployment management	To move new or changed hardware, software, documentation, processes, or any other component to live environments
Incident management	To minimize the negative impact of incidents by restoring normal service operation as quickly as possible
Knowledge management	To maintain and improve the effective, efficient, and convenient use of information and knowledge across the organization
Monitoring and event management	To systematically observe services and service components as well as record and report selected changes of state identified as events
Portfolio management	To ensure that the organization has the right mix of programs, projects, products, and services to execute the organization's strategy
Problem management	To reduce the likelihood and impact of incidents by identifying causes of incidents as well as managing workarounds and known errors
Release management	To make new and changed services and features available for use
Service catalog management	To provide a single source of consistent information on all services and service offerings and to ensure that it is available to the relevant audience
Service configuration management	To ensure that accurate and reliable information about the configuration of services, and the configuration items that support them, is available
Service desk	To capture demand for incident resolution and service requests. It should also be the entry point and single point of contact for users
Service financial management	To support the organization's strategies and plans for service management by ensuring that financial resources and investments are used effectively
Service level management	To set clear business-based targets for service levels and to ensure that delivery of services is properly monitored and managed against them
Service request management	To support the agreed quality of a service by handling all pre-defined, user-initiated service requests in an effective and user-friendly manner

Agile Methodology

1. **Inception**: In this interval, the vision of the system is developed, a project scope is defined, and the business case is justified.
2. **Elaboration**: In this interval, requirements are gathered and defined, risk factors are identified, and a system architecture is initialized.
3. **Construction**: In this interval, the software is delivered to the user.
4. **Operations**: In this interval, the software is available to the user and is maintained by the operations team.
5. **Transition** phase is from development to operations.

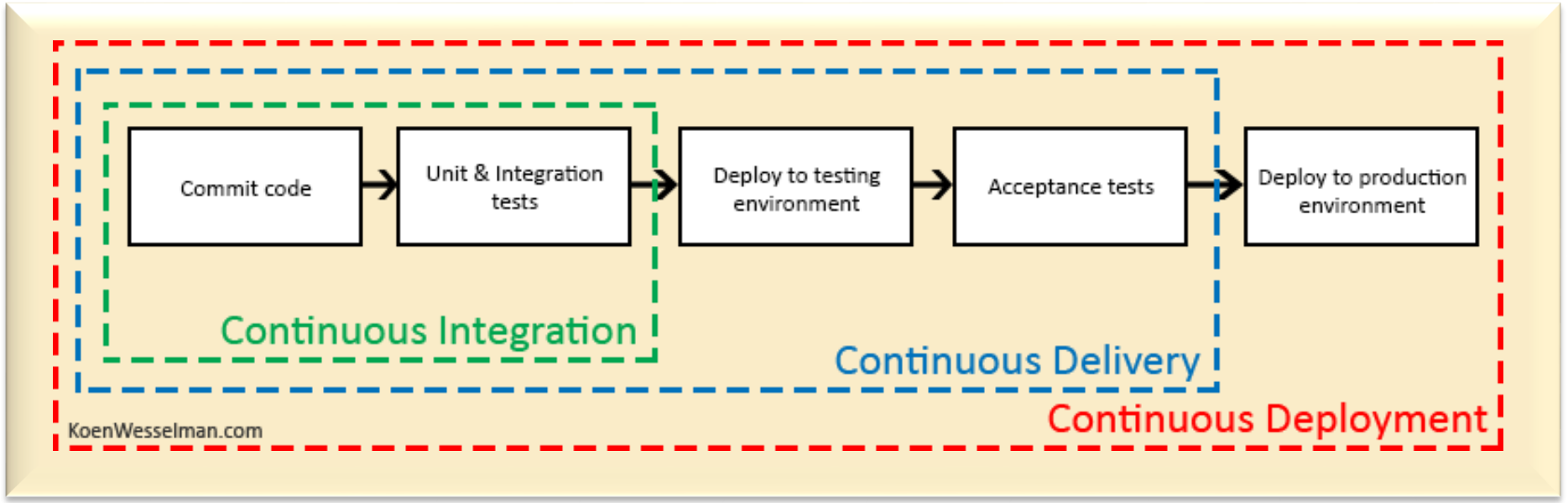


Traditional, Agile and DevOps – A comparison



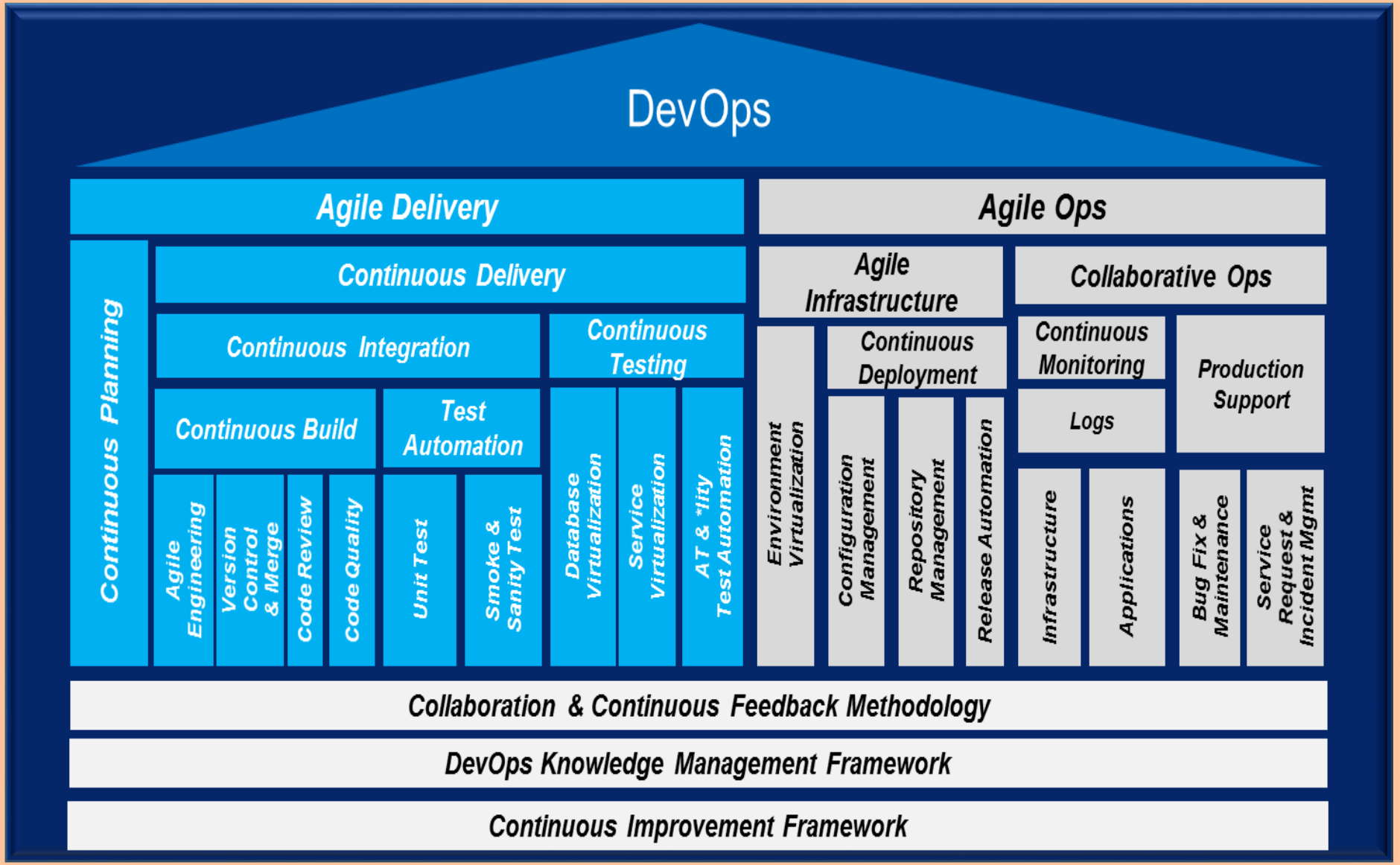
- Agile breaks the wall between Business and Development team.
- DevOps breaks the wall between Development and Operations team.
- DevOps centers on the concept of sharing: sharing ideas, issues, processes, tools and goals.

Continuous Delivery and Deployment



- In Continuous Delivery, the full software delivery life cycle automated till the last environment before production, so that you are ready at any time to deploy automatically to production.
- So, continuous Delivery doesn't mean every change is deployed to production ASAP. It means every change is proven to be deployable at any time
- In Continuous Deployment, you go one step further; you actually automatically deploy to production. The difference is really just whether there is *an automatic trigger or a manual trigger*.

DevOps Architecture

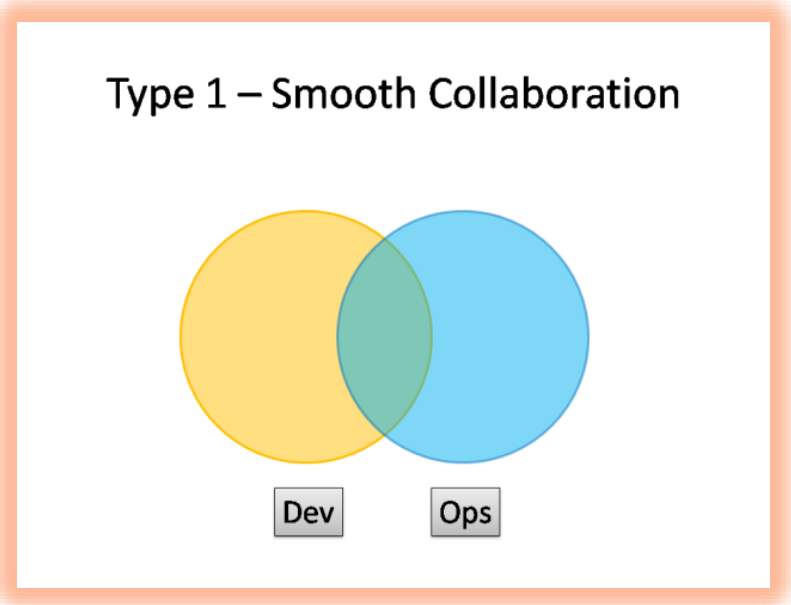


DevOps Models

- The primary goal of any DevOps setup within an organization is to **improve the delivery of value for customers and the business**.
- So, different companies might need different team structures for effective Dev and Ops collaboration.
- So what is the appropriate team structure for DevOps? Clearly, there is no magic conformation or team topology which will suit every organization.
- However, it is useful to characterize a small number of different models for team structures, some of which suit certain organizations better than others.
- Models:
  - 1. Smooth Collaboration
  - 2. Fully Embedded
  - 3. Infrastructure as a Service
  - 4. DevOps as a Service
  - 5. Temporary DevOps Team

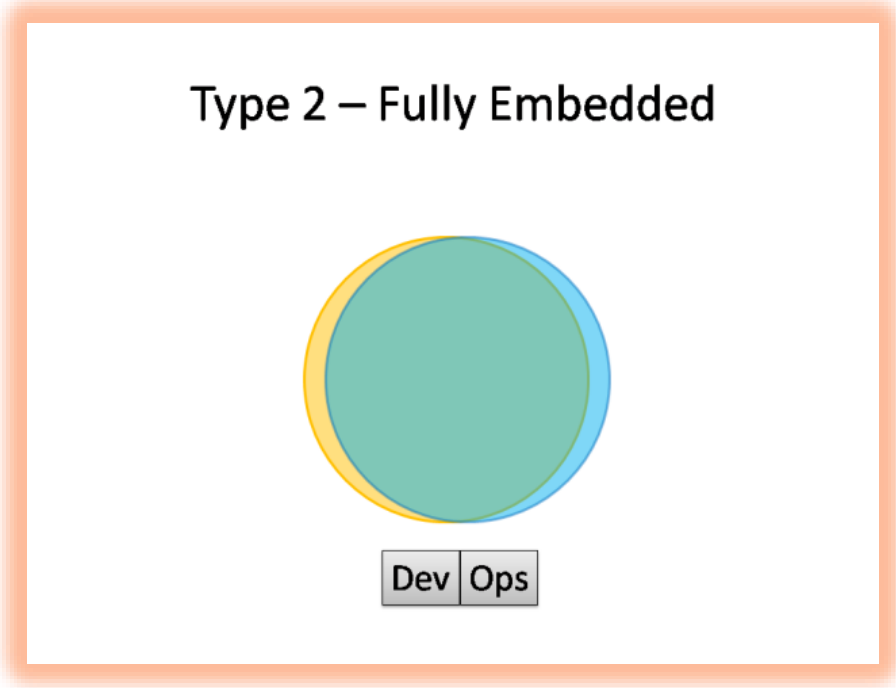
Dev and Ops Smooth Collaboration

- This type has smooth collaboration between Dev teams and Ops teams, each specializing where needed, and also sharing where needed.
- Dev and Ops must have effective shared goal that may be ‘Delivering Reliable, or Frequent Changes’.
- Ops team must be comfortable working with Devs and get to grips with test-driven coding and Git, and Devs must take operational features seriously and seek out Ops people for input into logging implementations, and so on.
- Type 1 is suitable for an organization with strong technical leadership and the potential effectiveness is high.



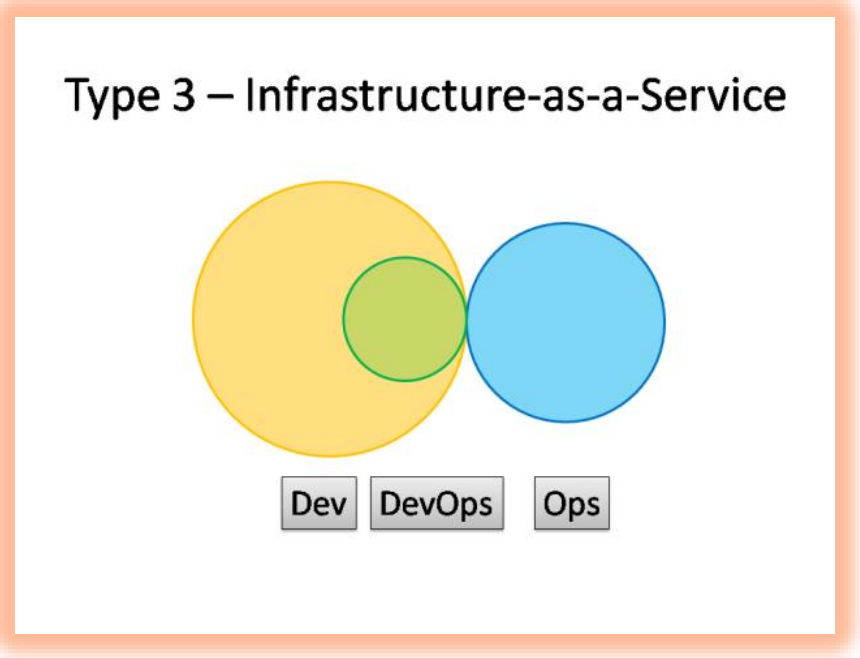
Dev and Ops Fully Embedded

- Where operations people have been fully embedded within product development teams in Type 2 topology.
- There is so little separation between Dev and Ops that all people are highly focused on a shared purpose.
- The *Fully Embedded* topology might also be called '**NoOps**', as there is no distinct or visible Operations team.
- This type is used by companies like Netflix and Facebook with a single web-based product have achieved.
- It is suitable for companies with a single main web-based product or service and it's potential effectiveness is also high.



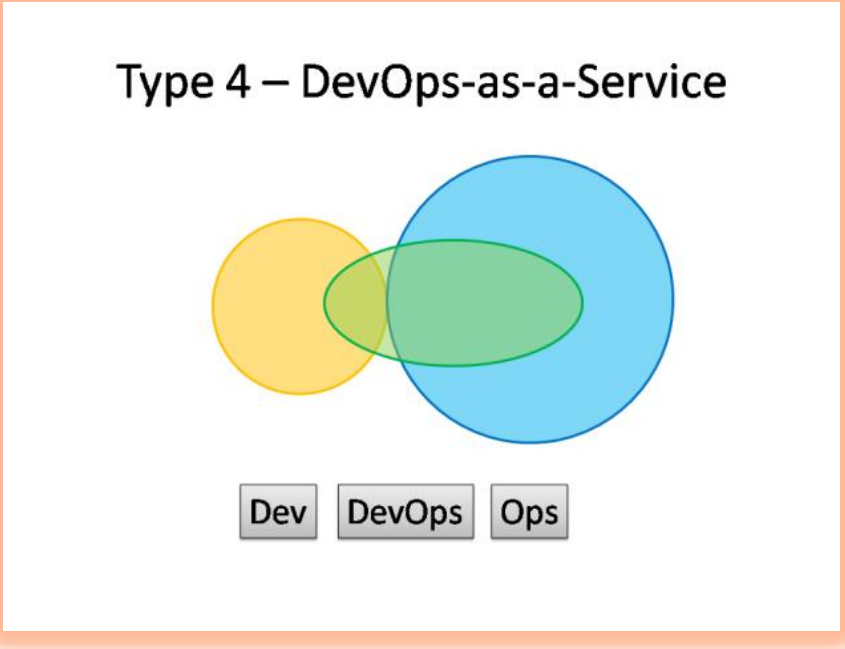
DevOps - IaaS

- This may be useful for Companies with a traditional IT Operations department which will not change rapidly. Or the companies run all their applications in the public cloud (Amazon EC2, OpenStack, Azure, etc.).
- This type helps to treat Operations as a team who simply provide the elastic infrastructure on which applications are deployed and run; the internal Ops team is thus directly equivalent to Amazon EC2, or Infrastructure-as-a-Service.
- A team (perhaps a virtual team) within Dev then acts as a source of expertise about operational features, metrics, monitoring, server provisioning, etc., and probably does most of the communication with the IaaS team. This team is still a Dev team.
- This type is suitable for Companies with several different products and services, with a traditional Ops department or the companies whose applications run entirely in the public cloud. The Potential effectiveness of this type is MEDIUM.



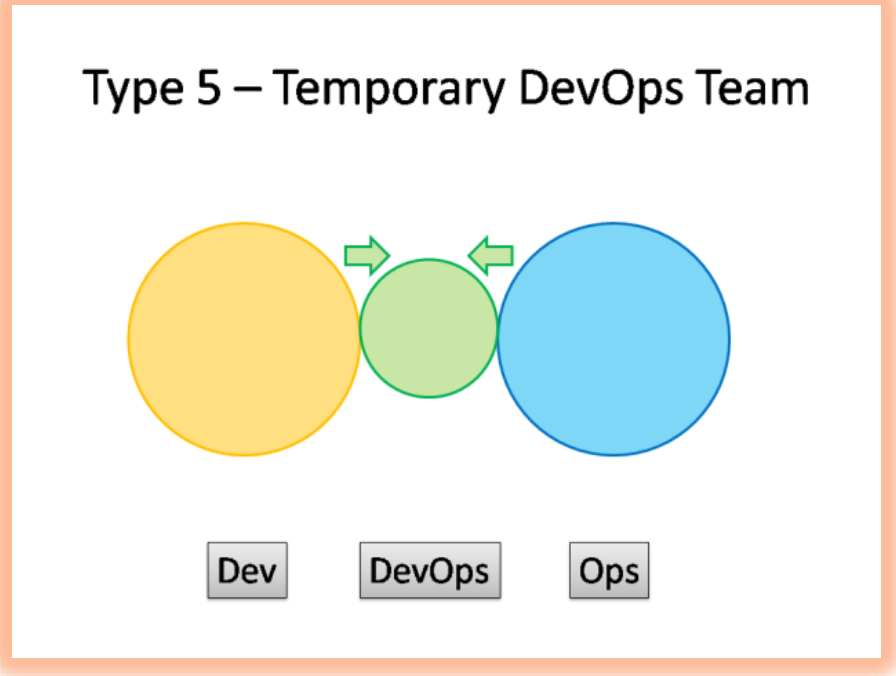
DevOps as a Service

- This type is suitable for some organizations, particularly smaller ones, may not have the budget, experience, or employees to take a lead on the operational aspects of the software they produce.
- The Dev team might then reach out to a service provider like OpenStack (Cloud Computing Company) to help them build test environments and automate their infrastructure and monitoring, and advise them on the kinds of operational features to implement during the software development cycles.
- So, *DevOps-as-a-Service* topology could be a useful and realistic way for a small organization or a team in a project to learn about automation, monitoring, and configuration management, and later move towards a Type 1 for *Smooth Collaboration* model when they grow and add more team members for operational focus.
- This type is good for smaller teams or organizations with limited experience of operational issues and it's Potential effectiveness is MEDIUM.



Temporary DevOps Team

- The members of the temporary team will ‘translate’ between Dev-speak and Ops-speak. If you start to see the value of bringing Dev and Ops together, then the temporary team has a real chance of achieving its aim.
- This type can be used as a precursor of type 1 (smooth collaboration) and it’s potential effectiveness is initially low and once it’s transformed to Type 1 then its potential effectiveness is high.



- Exactly which DevOps team structure or topology will suit an organization depends on several things. The discussed topologies so far, are meant as a reference guide or experimental for assessing which patterns might be appropriate in your project.
- But in reality, a combination of more than one pattern, or one pattern transforming into another, will be the best approach.

Components of Software Delivery

$$T_{DELIVERY} = \frac{T_{PLAN} + T_{DESIGN} + T_{DEVELOP} + T_{BUILD} + T_{DEPLOY} + T_{TEST} + T_{FIX} + T_{RELEASE} + T_{EVALUATE}}{\%\ TRUST\ (0\text{-}1)}$$

- $T_{DELIVERY}$ is optimized by minimizing the time for each of the tasks required to complete the delivery. They estimate the time to do each part and then the total is the sum of the parts.
- *Productivity is inversely proportional to the number of dependencies in a release.*
- ***Trust is inversely proportional to the number of dependencies in a release.***
- The important theme of DevOps is that the entire development-to-operations lifecycle must be viewed as one end-to-end process.

Traditional way – Impact on Roles

Dev

- Manual release process
- Different build servers
- Deployment done by development teams

QA

- Manual deploys
- Different puppet repos to production
- Manual testing

Ops

- Manual deploys
- Cheat sheets
- Have to learn complexities of each application through experience

DevOps Principles Involves….

1. DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

2. DevOps is also characterized by operations staff making use many of the same techniques/tools as developers for their systems work.

3. DevOps principles involves CALMS:

culture - people -> process -> tools

Automation - Infrastructure as code

Lean – Small batch sizes, focus on value for end user

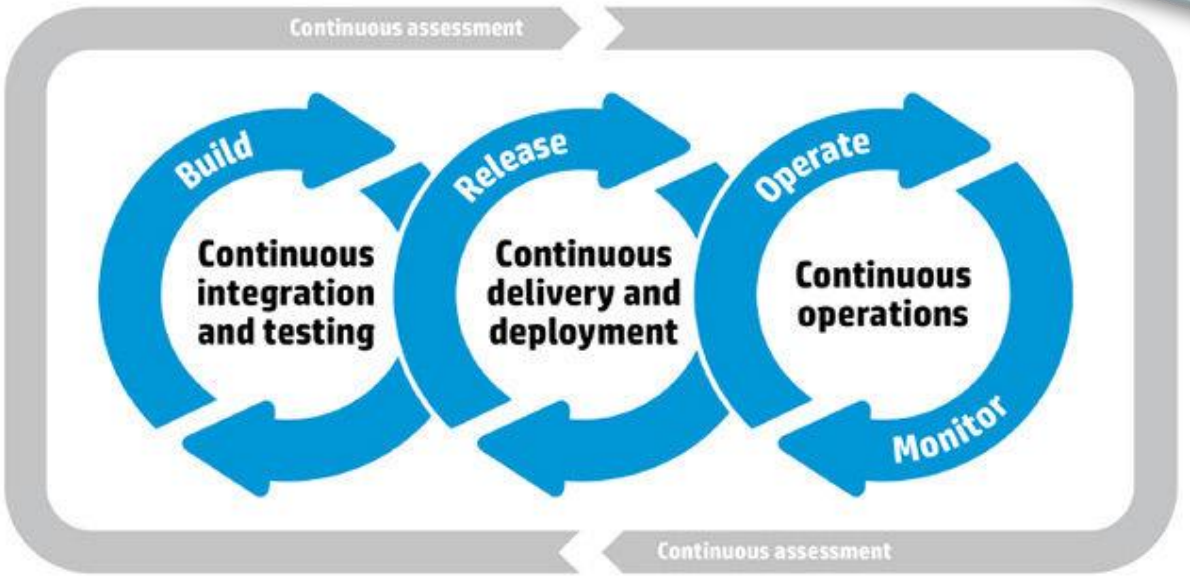
Measure – Measure everything; Show improvement

Sharing - Collaboration between Dev and Ops.

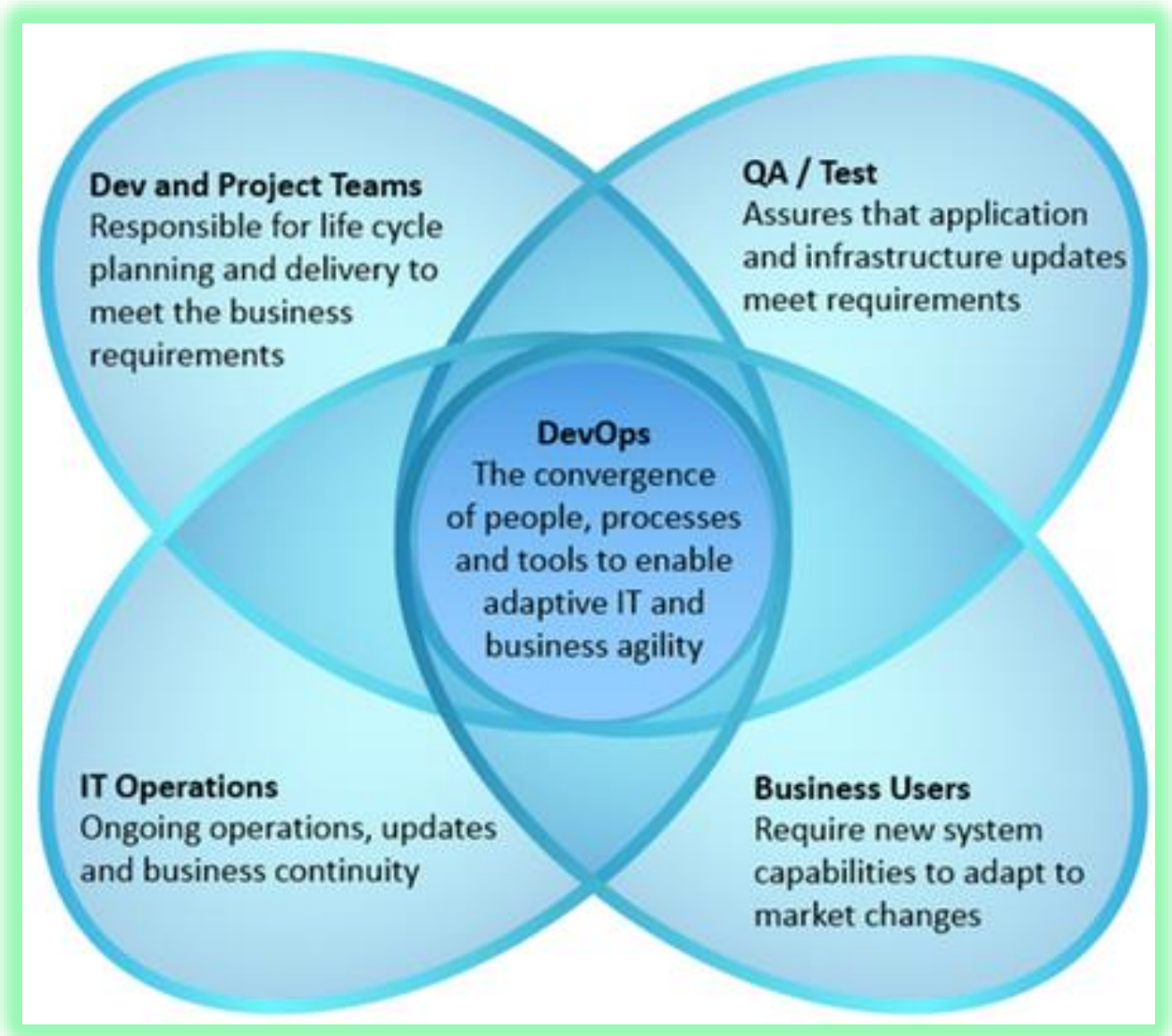
DevOps Practices Involves

1. Version control for all
2. Automated testing
3. Proactive monitoring and metrics
4. Configuration management.

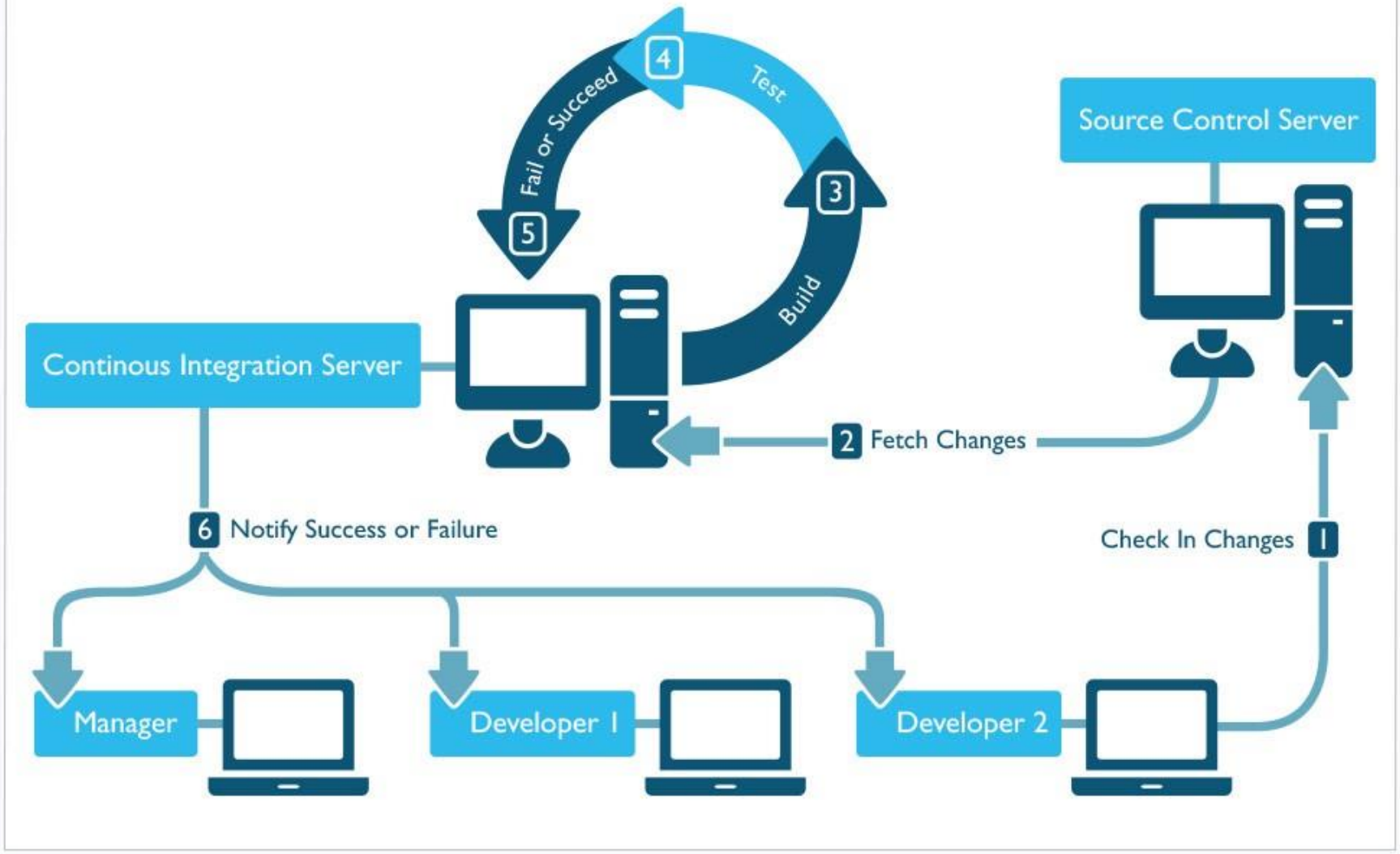
5. Continuous integration /deployment /delivery
6. virtualization/cloud/containers
7. Toolchain Approach



Roles and Responsibilities



Software Development Life Cycle - Automation



DevOps Way

Dev

- Deploy product like environment in every commit
- Test each and every changes
- Automate release

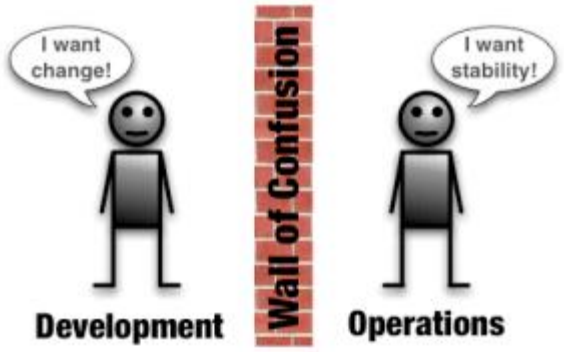
QA

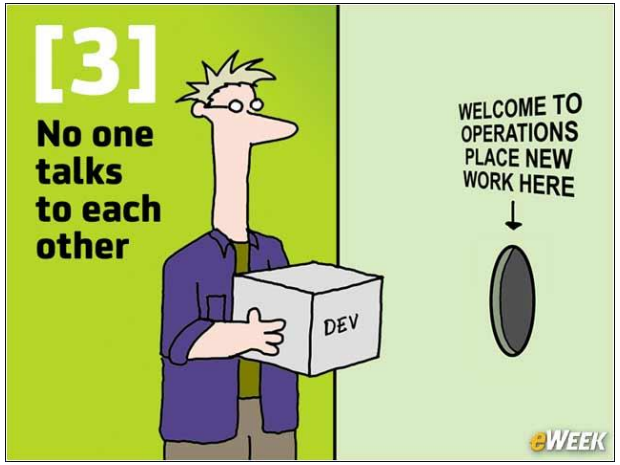
- Automated self service deploy
- Test environment same as production via shared puppet repo
- Automated acceptance tests

Ops

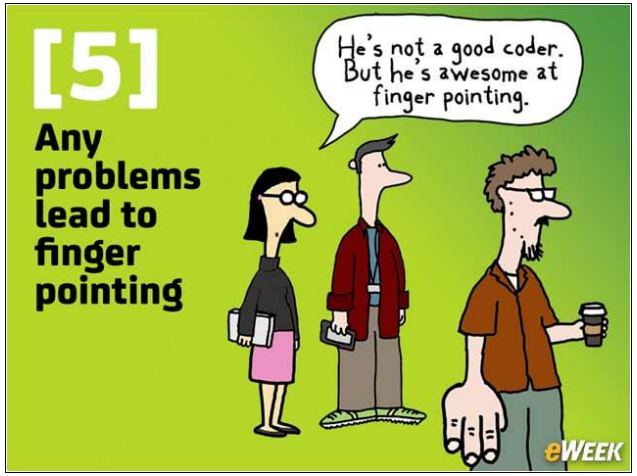
- Application tested thoroughly
- Have confidence in rejecting an application if it fails
- Minimal knowledge required to supporting application

DevOps Barriers





❖ This cultural change is made especially difficult because of the conflicting nature of departmental roles.



- ➢ Operations seeks organizational stability
- ➢ Developers seek change
- ➢ Testers seek risk reduction.

❖ Getting these people and opposing viewpoints to work cohesively is a critical challenge in enterprise DevOps adoption.

Remove DevOps Barriers

Culture: Respect -if there is only one thing you do

- Don't stereotype
- Respect other people's expertise, opinions and responsibilities.
- Don't just say "No"
- ***Don't hide things***

Developers: Talk to ops about the impact of your code:

- what metrics will change, and how?
- what are the risks?
- what are the signs that something is going wrong?

This means Dev needs to work this out before talking to ops.

Remove DevOps Barriers

Trust:

- Ops needs to trust dev to involve them on feature discussions.
- Dev needs to trust ops to discuss infrastructure changes.
- Everyone needs to trust that everyone else is doing their best for the business.

Ops: be transparent, give devs access to systems.

- Healthy attitude about failure:
- Avoiding Blame - No fingerpointing
  - Ops who think like devs - Devs who think like ops
  - Through Automation one step build and deploy.

- **Key Benefits**
  - Faster release of apps with automation of integrated build, test and deployment process
  - Increase developer and operational efficiency by managing your infrastructure as code
  - Improve customer experience with immediate feedback loops and continuous improvement

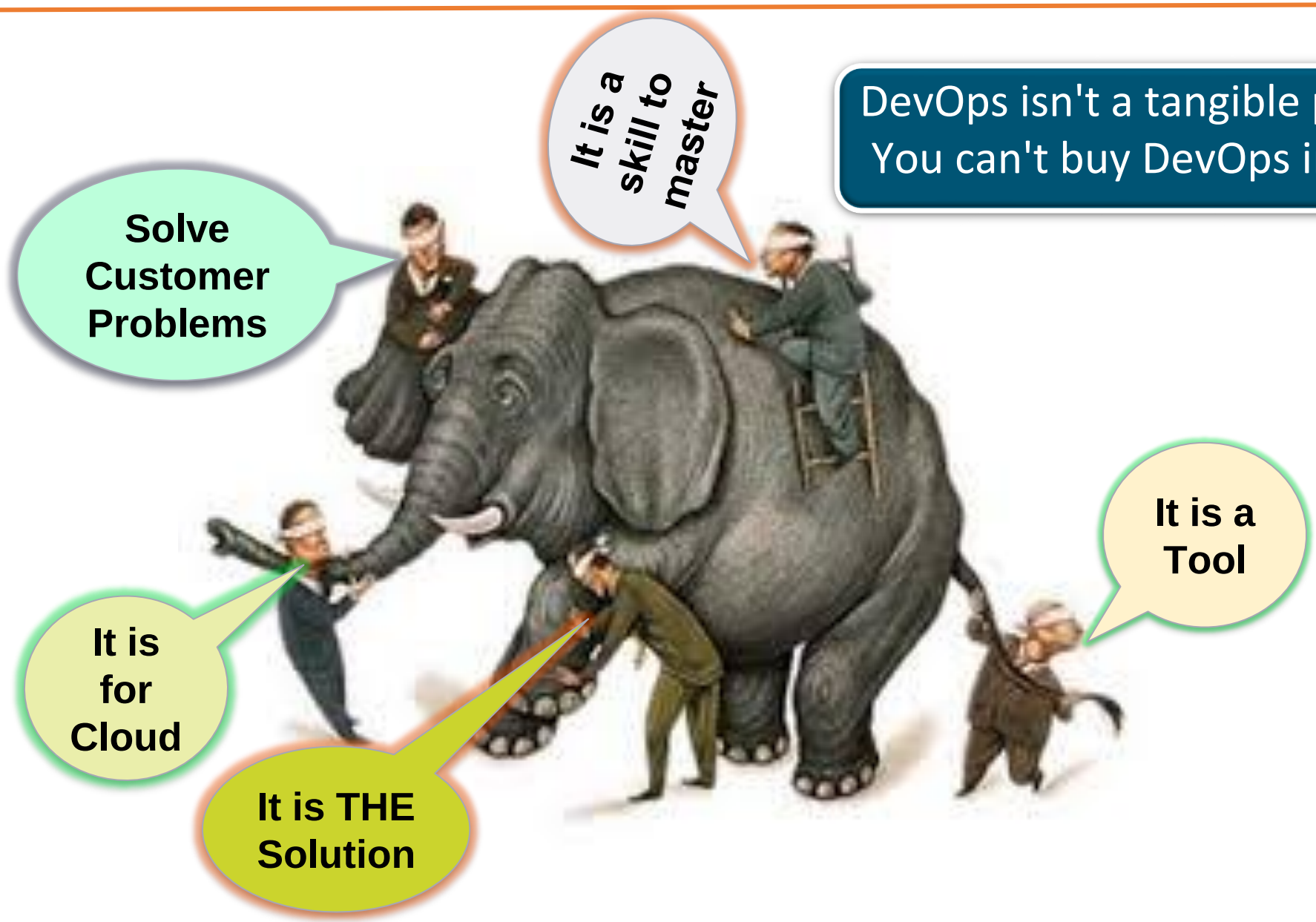
Impact for the Development Team

- Develop application code. (BAU)
- Develop Infrastructure as code by using the infrastructure APIs. (New)
- Check in application code into code repository. (BAU)

- Check in Infrastructure code into code repository. (New)
- Excel in Application Development. (BAU)
- Learn DevOps concepts – eg. Culture; Automation Tools for CI, CT, CD, CD, CM; Scripts for Provisioning, Profiling, Modeling your infra. (New)

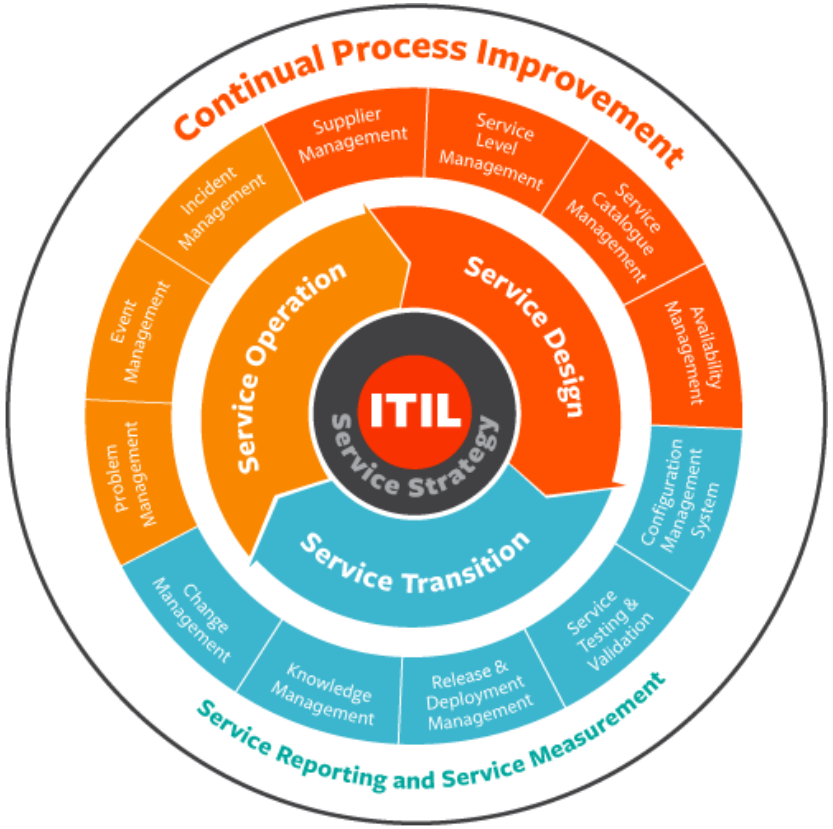
- Work closely with the Ops team. Can't blame them any more for infrastructure issues. You have more control now. (New)

Popular Myths / Misconceptions



Popular Myths / Misconceptions

- **DevOps is Development + Operations**
  - ✓ Modern application lifecycle has many other teams involved like Business , Senior Management, partners, Testers , Compliance etc.
- **DevOps is all about using tools & Automation**
  - ✓ Surprise !!! There is No DevOps tool !!! You have tools for CI/CD and Operations. Tools and automation are part of DevOps culture.
- **DevOps is a Skill - DevOps Engineer , DevOps Manager**
  - ✓ Replace the word “DevOps” with Agile . Does it make sense ? Like Agile , you can have DevOps Team not an DevOps Engineer or Manager
- **DevOps disrupts existing processes like ITIL**
  - ✓ It complements & enhances these processes. DevOps still uses the same principles but with an enhanced scope.

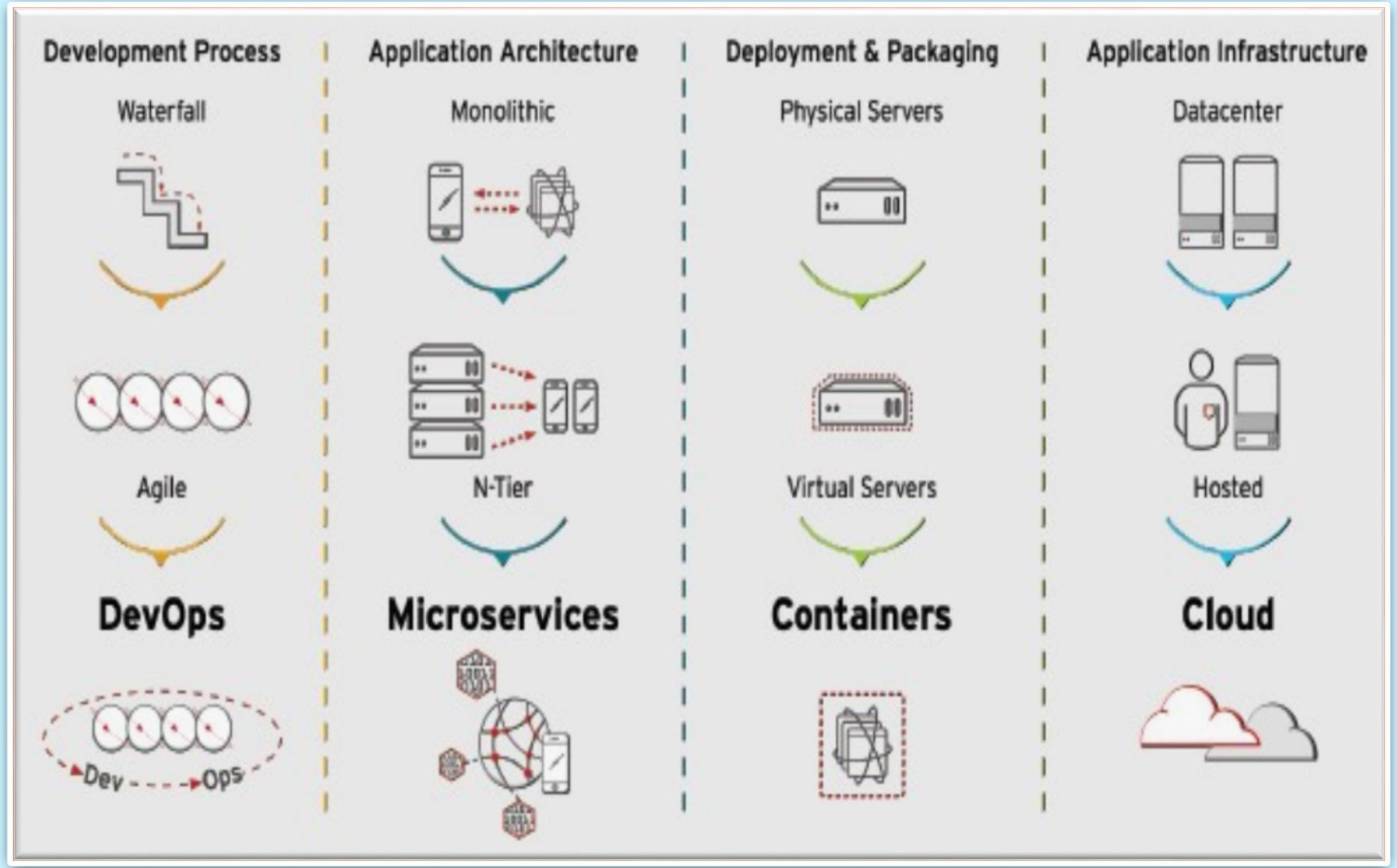


ITIL, an acronym for Information Technology Infrastructure Library, is a set of detailed practices for IT service management that focuses on aligning IT services with the needs of business

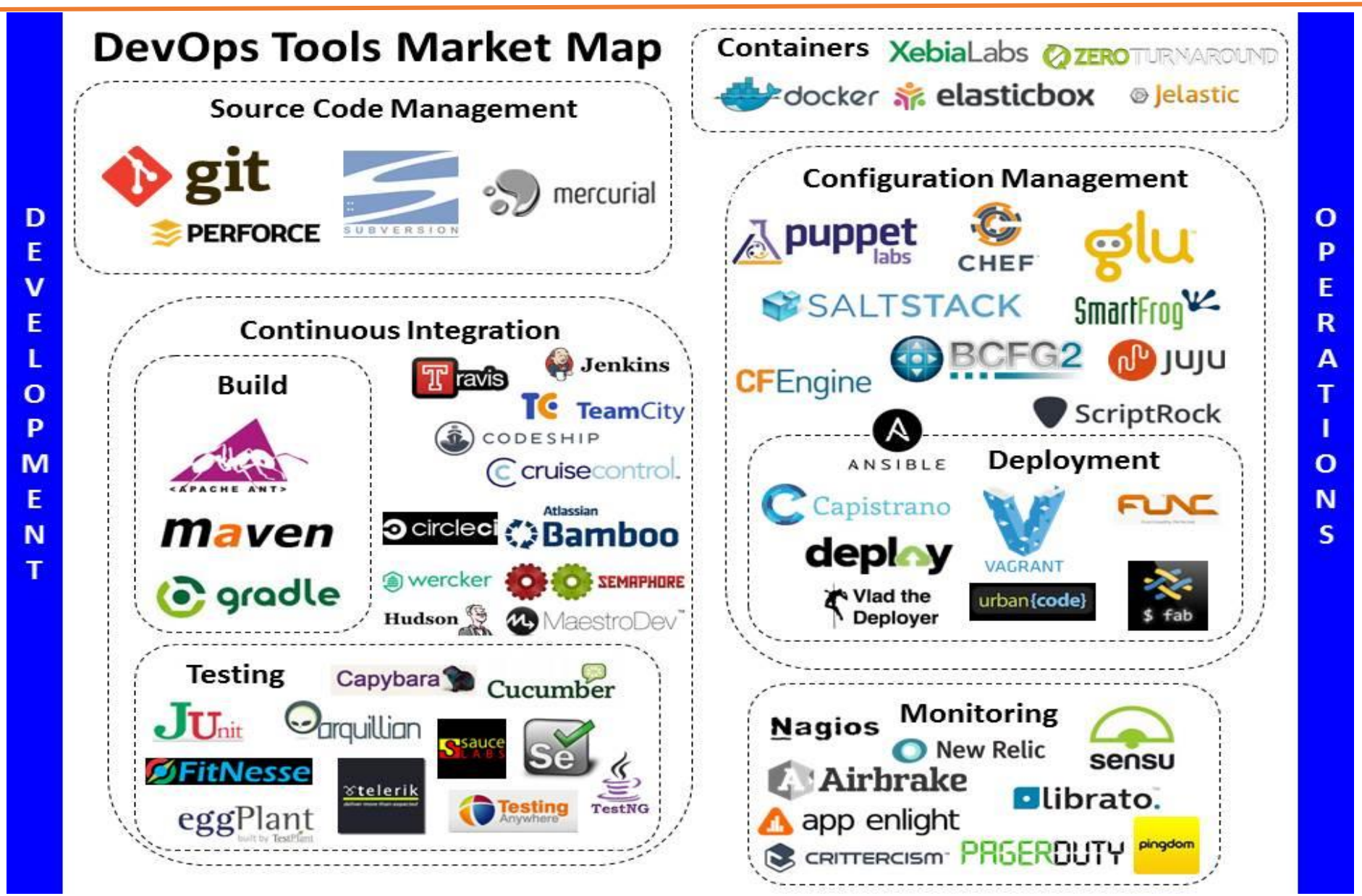
Popular Myths / Misconceptions

- **Everybody can / should do 10 deploys a day (or any other exotic number) with DevOps**
  - ✓ No , only if there is a business need and supporting eco system.
- **Is for the cloud or web shops only**
  - ✓ A DevOps approach is applicable to any infrastructure but a cloud infra gives you an additional benefit
- **It is THE solution – a silver bullet**
  - ✓ Thinking a DevOps strategy will solve all issues in simple manner. It's a collaborative & Iterative approach to solving issues pertaining to people , process and technology to achieve streamlined Flow
- **Finally, Only Agile Projects can do DevOps**
  - ✓ It is applicable to any development model.

Enterprise Applications - Evolving Themes

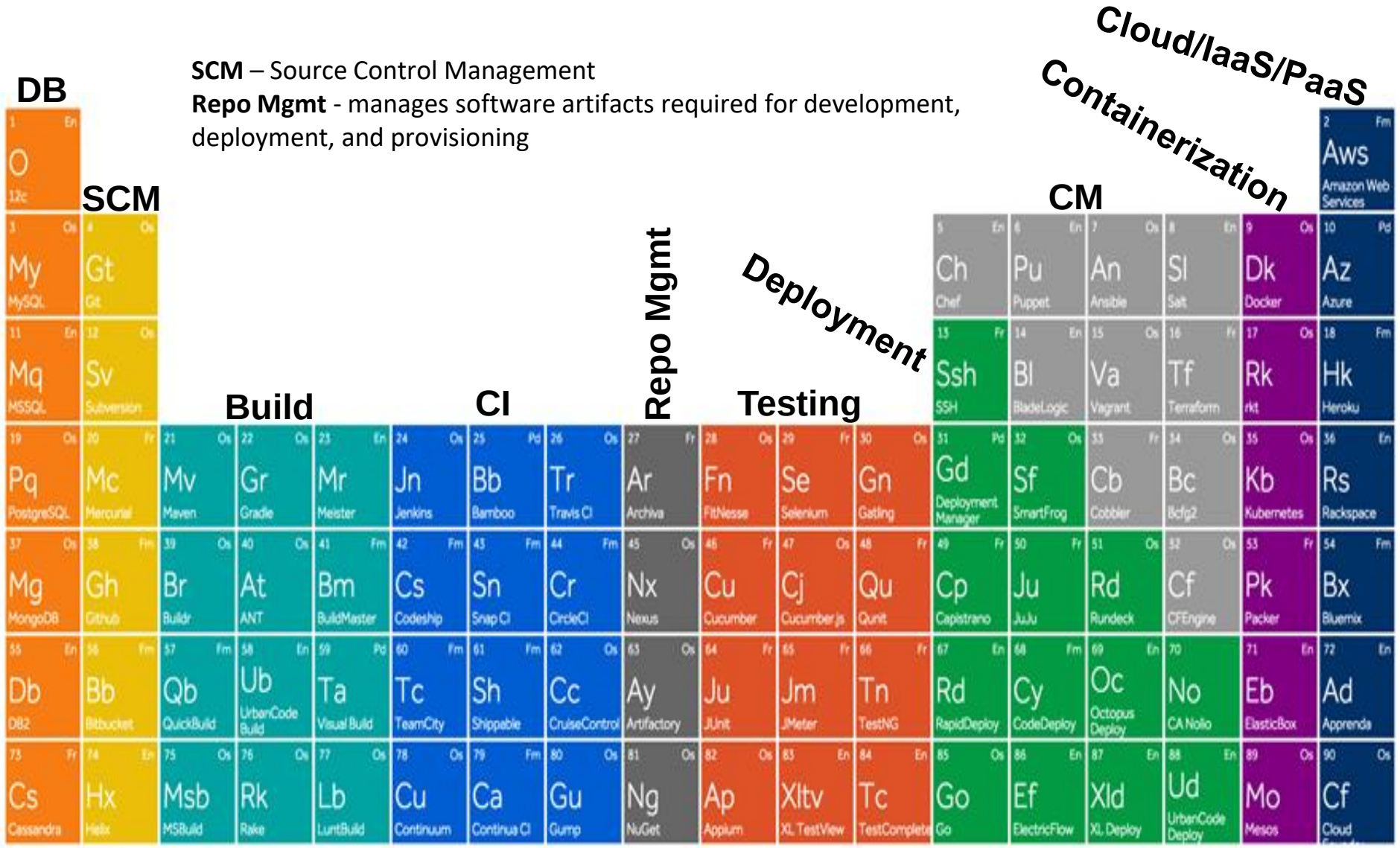


DevOps Tools



Periodic Table of DevOps Tools

SCM – Source Control Management
Repo Mgmt - manages software artifacts required for development, deployment, and provisioning



References

1. https://www.ca.com/us/rewrite/articles/devops/a-short-history-of-devops.html
2. SDLC - http://trevinchow.com/blog/2007/09/12/the-perils-of-no-pre-production-testing/
3. DevOps for Developers by Michael Huttermann
4. https://blog.matthewskelton.net/2013/10/22/what-team-structure-is-right-for-devops-to-flourish/