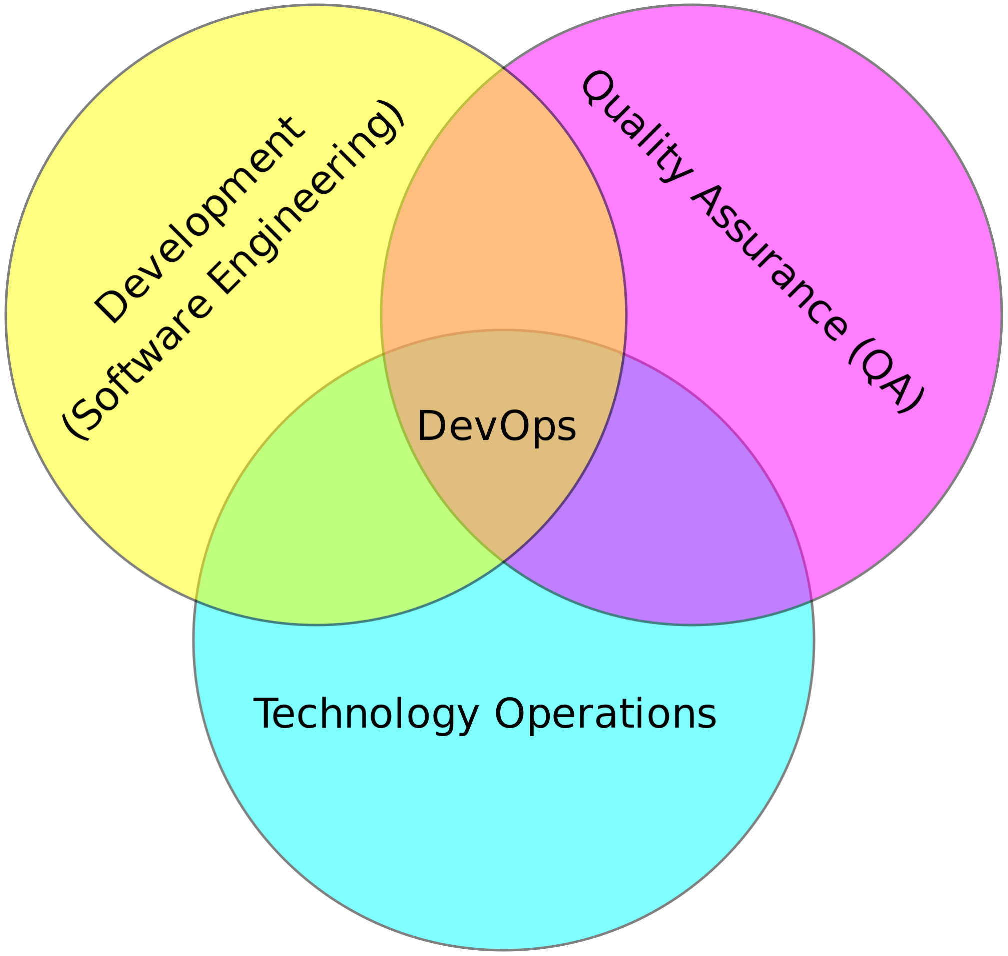
DevOps and Architecture

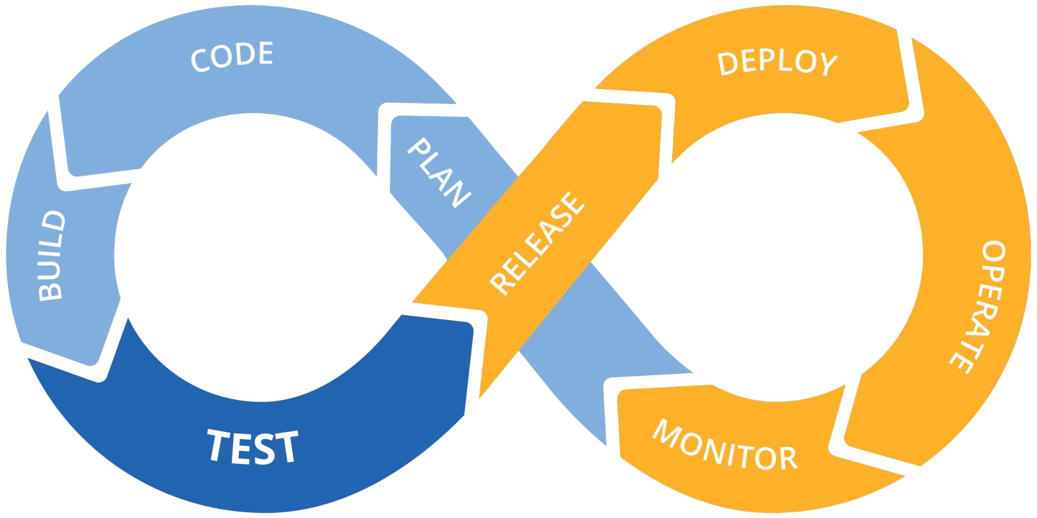
Where does DevOps Live?



Key Issues

- Quality is the key – normally operations staff guard that but development are responsible for designing in the Quality Attributes that ensure quality.
- Operations have the direct experience of use of the system – monitoring that use is a way of empirically verifying quality – operations have the data that is used to regulate operations and is essential information for development.
- Development is responsible for building in the right monitoring to ensure operations can operate effectively

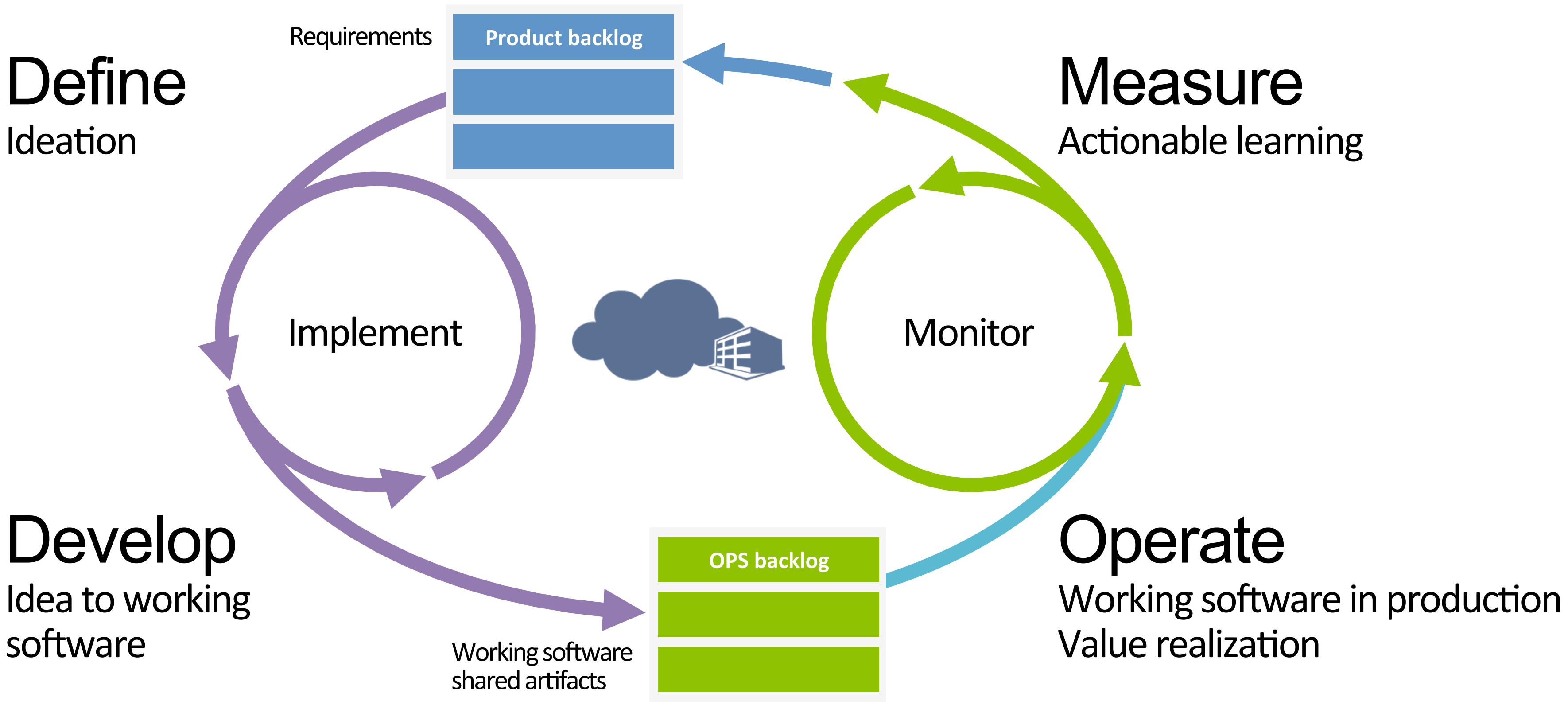
DevOps Sees Development and operations as a unified entity



The Essence of DevOps

- DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal operation while ensuring necessary quality.
- Movement Comes from Open Source (OSLC: http://open-services.net)
- When Development and Operations Synergize
- Covers the *entire* Application LifeCycle

The Whole Application LifeCycle



Why OSLC and Standardizing Matters

Software runs the world

But it is heterogeneous and disjoint

And it needs to be integrated

Traditional approaches to software integration require custom software

Custom software is expensive to maintain and a limit on future choice

Custom integrations drive software TCO higher and limit choice

Open standards are key enablers for broad and large-scale integration

OSLC standards simplify lifecycle integration leading to cost savings and increased flexibility

OSLC is helping the world run more efficiently

Standardizing OSLC will increase adoption and acceptance of OSLC

The world will benefit from standardized integration through OSLC

Aspirations for OSLC

- Foundational technology for all integration
- The household name for integrations
- Natural choice for standardizing loosely-coupled integrations in new domains

The Technical Vision for OSLC

Users can work seamlessly across their tools

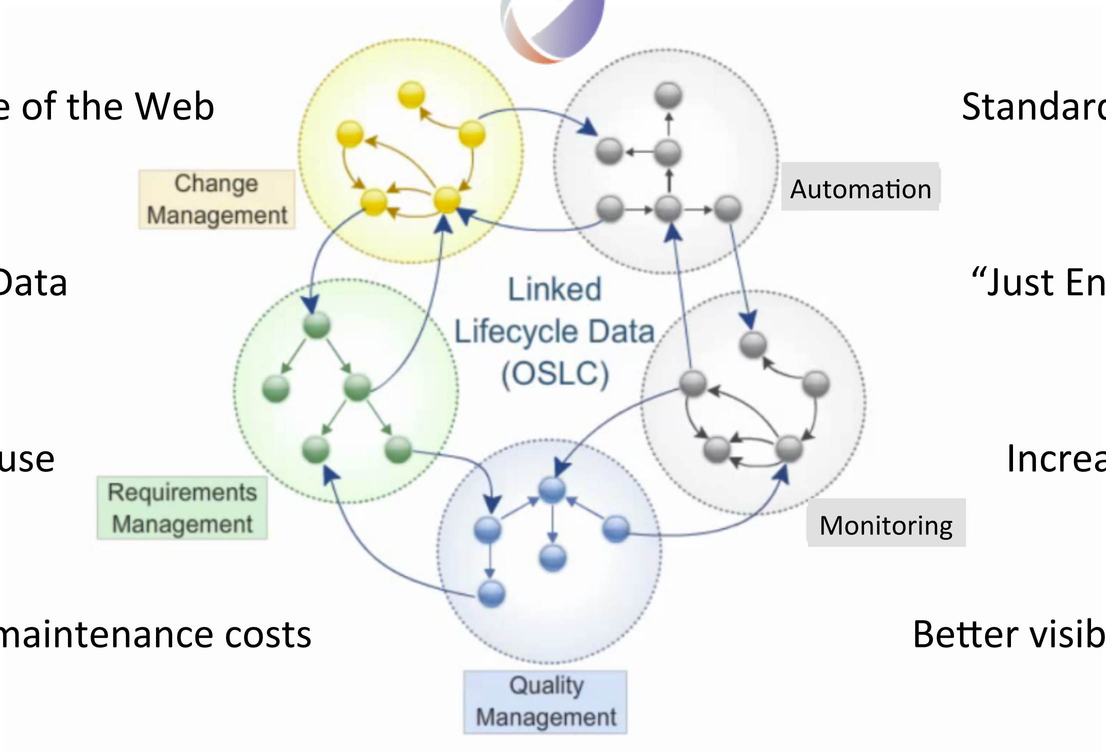
(complex and fragile synchronization schemes not required)

Architecture of the Web

Linked Data

Increased reuse

Decreased maintenance costs



Standard Interfaces

"Just Enough" integration

Increased traceability

Better visibility

OSLC is an <u>open</u> and <u>scalable</u> approach to lifecycle integration.
It <u>simplifies</u> key integration scenarios across <u>heterogeneous</u> tools

Key issues for DevOps

- Traditionally we use test as the way of delivering quality change but we can “shepherd” committed change into use by controlling quantities of change, users experiencing change, results of monitoring than this may offer a better way.
- Delivery mechanism needs to be high quality: reliable, repeatable, available,

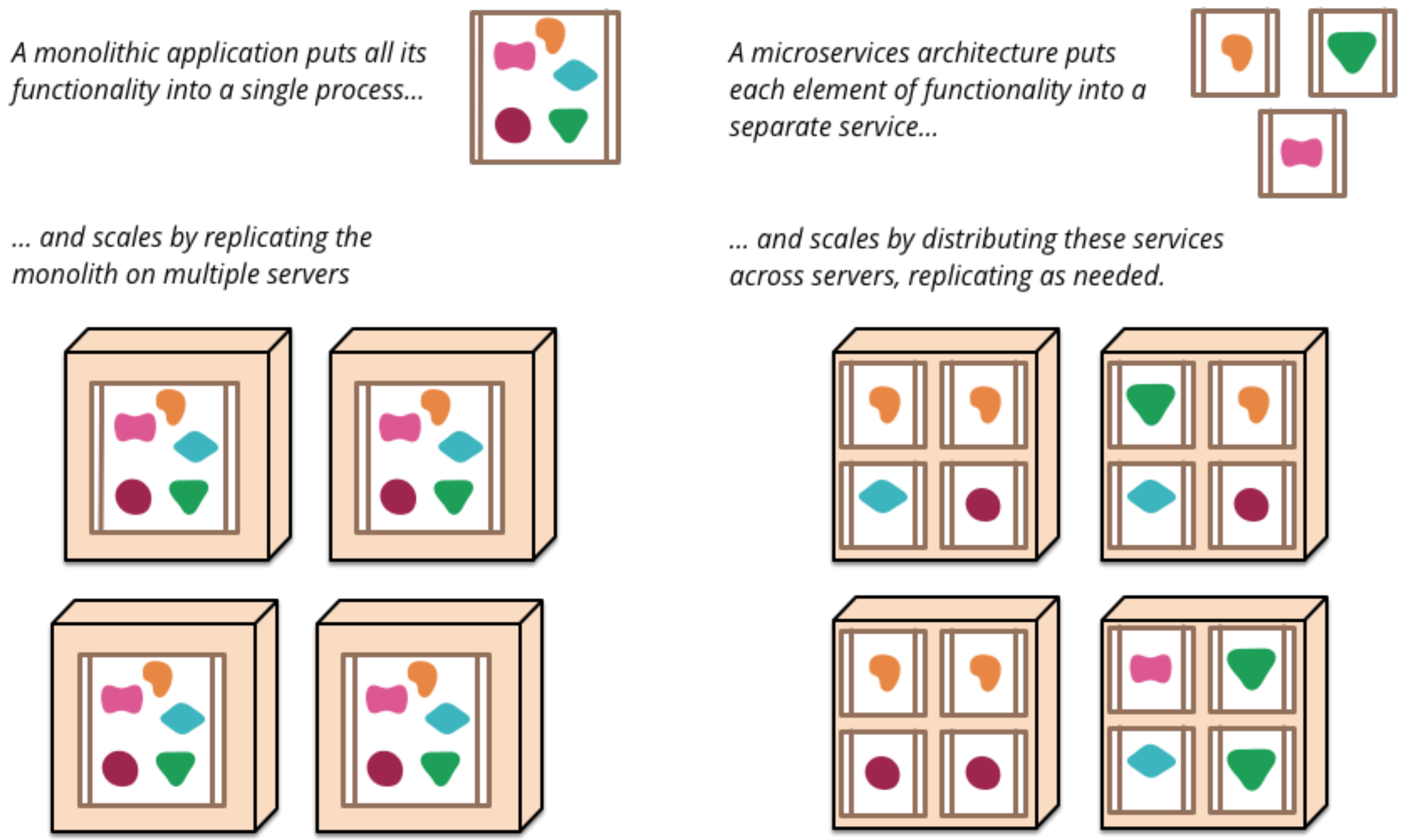
Critical points

- Making the decision to commit the code to be introduced into the system.
- Transitioning from being under consideration into part of the production deployment that will be used by all users.
- Issues is how to have enough confidence to make each of these transitions. Monitoring is critical.
- The question is how to ensure the transitions are as reliable as possible.

The extent of the lifecycle

- Involves all people involved in the delivery of the service/application
- Operations and development people are in continuous interaction.
- We need architecture to achieve this.
- Microservices architectural pattern is often used.

Microservices


A monolithic application puts all its functionality into a single process...

... and scales by replicating the monolith on multiple servers

A microservices architecture puts each element of functionality into a separate service...

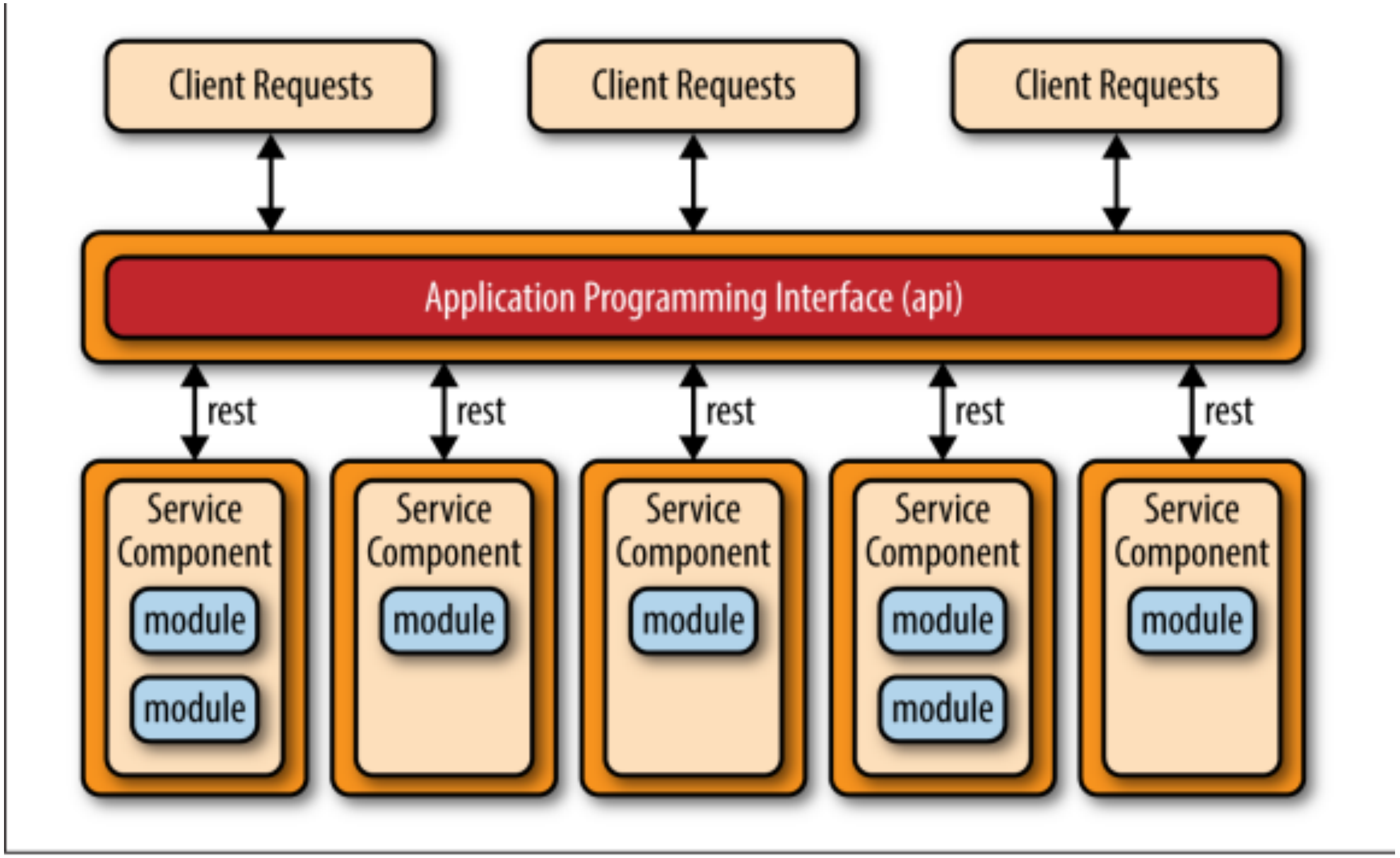
... and scales by distributing these services across servers, replicating as needed.

http://martinfowler.com/articles/microservices.html

Attributes of Microservice Architecture

- Separately deployed units
- Very small service components
- Single purpose function or an independent portion of functionality
- Distributed
- Loosely coupled
- Multiple versions are acceptable
- Asynchronous
- No Orchestration

Example: Providing and API



Thanks to Assaf Gannon

A Practical Example

"Do painful things more frequently, so you can make it less painful…"
https://www.eiseverywhere.com/file_uploads/c6b285b64a18cc2742c0fb20061d6530_OBC_BreakoutSession_AdrianCockcroft_1pm.pdf

Adrian Cockcroft, Architect, NetFlix

The First Way – Systems Thinking

- Understand the entire flow of work
- Seek to increase the flow of work
- Stop problems early and often
  - Don’t let them flow downstream
- Keep everyone thinking globally
- Deeply understand your systems

Richa

Defining Work and Make It Visible

- Business Projects
  - The new inventory system
- Internal IT Projects
  - Deployment Automation
- Changes
  - Database Performance Tuning
- Unplanned Work
  - Web Site Outage

One Step Environment Creation

- Need a common environment build process
  - For development, qa and production
- The environment will evolve as development proceeds
- The longer you wait to have a common environment build process, the harder it is to create one

The First Way Goals

- One source of the truth
  - Code, environment and configuration in one place
- Consistent release process
  - Automation is essential (one click)
- Decreasing cycle times, Faster release cadence

The Second Way – Feedback Loops

- Understanding and responding to the needs of all customers (internal and external)
- Shorten and amplify all feedback loops
- With feedback comes quality

Crossteam Connections

- Development is embedded in the Ops incident escalation process
- Dev and Ops collaborate on post-mortems and root cause analysis
- Monitoring and metrics become essential

A Word about Metrics

- Avoid vanity metrics
  - Giving you numbers that make happy noises

- Real metrics are actionable
  - What do you do when it goes up?
  - What do you do when it goes down?

- Real metrics reflect business, not technology

Automating Feedback Loops

- Capture as much data as possible at the incident
- Avoid interpretation
- The issue becomes the data, not the people who gathered it

Second Way Goals

- Defects and performance issues fixed faster
- Ops and InfoSec user stories appear as part of the application
- Everyone is communicating better
- More work getting done

The Third Way – Synergy

- Consistent process and effective feedback result in agility
- Now use that agility to experiment
- You only learn from failure
  - So fail often, but recover quickly

Break Things Before Production

- Consistency in code, environments and configuration
- ASSERTs to catch misconfigurations and inconsistencies
- Static code analysis, and testing become part of the continuous integration and deployment

Battling Technical Debt

- Allocate 20% of cycles to technical debt reduction
  - Before you end up allocating 100%
- Remember they’re still visible stories with measurable metrics

Fighting Against One Right Way

- Rapid cycling encourages experimentation
  - Every feature can be split-tested
  - Use Metric Driven Development
  - If you can’t tell which test is better, what’s the point?

Third Way Goals

- Ability to anticipate, even define new business needs through visibility in the systems
- Ability to test and optimize new business opportunities in the system while managing risk

Resources

- Visible Ops Handbook (Gene Kim)
- The Phoenix Project (Gene Kim)
- Web Operations (Allspaw/Robbins)
- Continuous Delivery (Humble/Farley)
- Lean Startup (Eric Reis)