Practical DevOps

Getting Beyond the Basics

Introduction

- Understanding the DevOps Movement and how it changes some common paradigms
- A practical way to understand and break down the challenges associated with applying DevOps techniques
- 5 simple steps to get started

Background

THE DEVOPS MOVEMENT

Where did DevOps Come From?

- Much debate
  - Next step in Agile?
  - Confluence of things and problems from realizing value of Agile?
  - Another word for Continuous Delivery?
  - A side-effect of the explosion of virtualization?
  - A byproduct of commodity cloud services?
  - Some Belgian dude* hallucinating after one too many waffle, fried potato, and chocolate beers?
  - Something deeper?

* Patrick Debois – long credited with coining the term “DevOps” is Belgian

DevOps Can Be Hard to Define

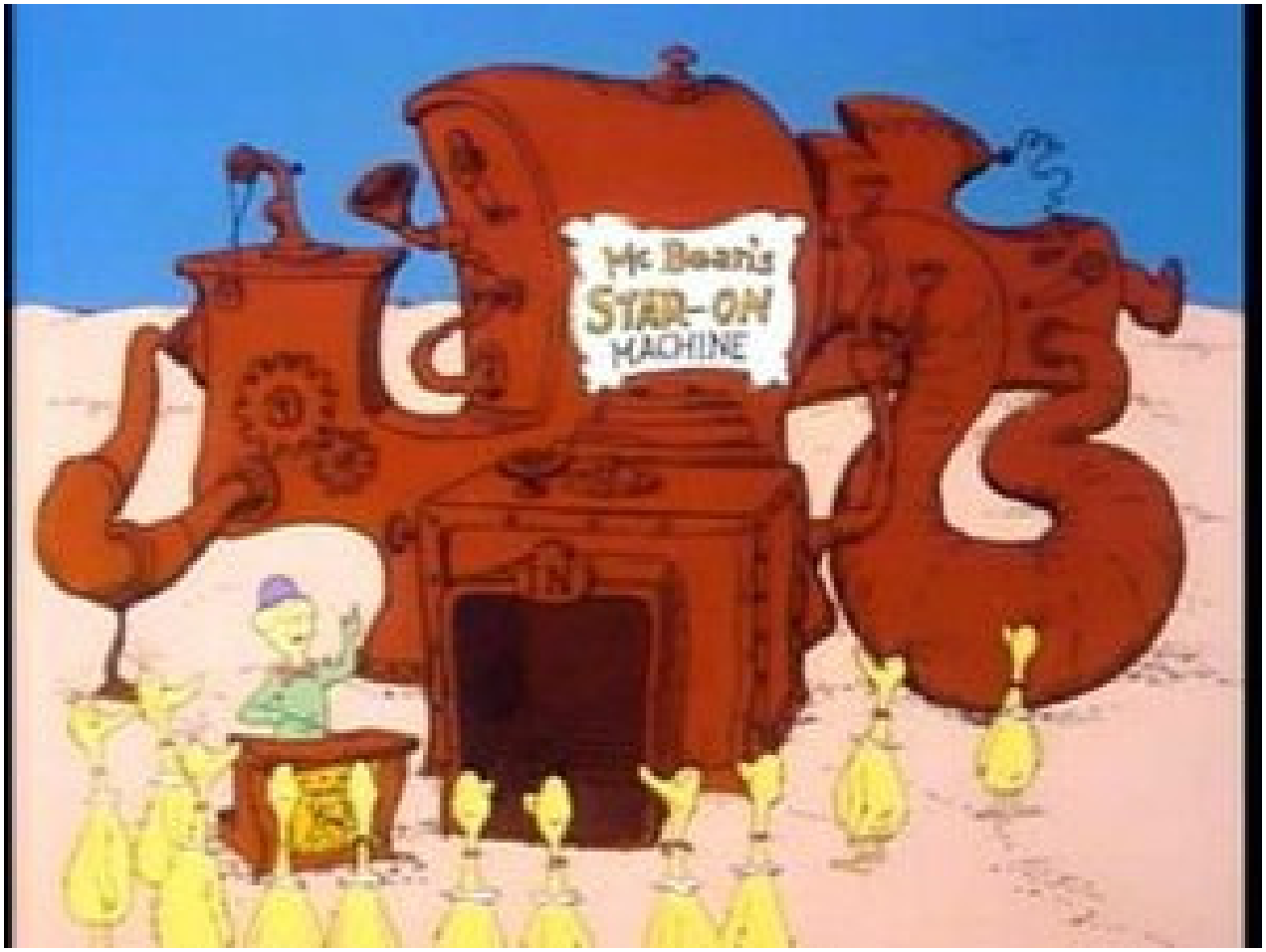
[DevOps is] “an IT service delivery approach rooted in agile philosophy, with an emphasis on business outcomes, not process orthodoxy” --Gartner

- Movement to unify execution across Dev, Test, and Operations
- Based on lessons of Agile and Lean
- Seeks to eliminate cultural, organizational, and tool barriers that create friction and cost in Release Management
- Highly associated with cloud methodologies and technologies

WTF it Means in Practical Terms

- Ability to deliver changes in my application system to any environment(s) in my shop (including production) in a completely predictable way, a minimal timeframe, and with minimal human effort.
- The ability to observe the impact of those changes and use the observations to inform the next set of changes.

Change Applied As Needed



Wrapping Your Head Around It

BREAKING DOWN THE PROBLEM

Development or Delivery

Engineering or Economics

Software Development: Engineering Driven	Software Delivery: Economics Driven
Distinct development phases	Continuously evolving systems
Distinct handoff from development team to maintenance team	Common process, platform, and team for development and maintenance
Distinct and sequential activities from requirements to design to code to test to operations	Sequence of usable capabilities with ever-increasing value
Phase and role specific processes and tools	Collaborative platform of integrated tools and process enactment
Collocated teams	Geographically distributed, Web-based collaboration
Early precision in complete plans and requirements	Evolving precision as uncertainties are resolved
Governance through measurement of artifact production and activity completion	Governance through measurement of incremental outcomes, and progress/quality trends
Engineering discipline: precisely define requirements/plans completely, then track progress against static plans and scope	Economic discipline: reduce uncertainties, manage variance, measure trends, adapt and steer with continuous negotiation of targets

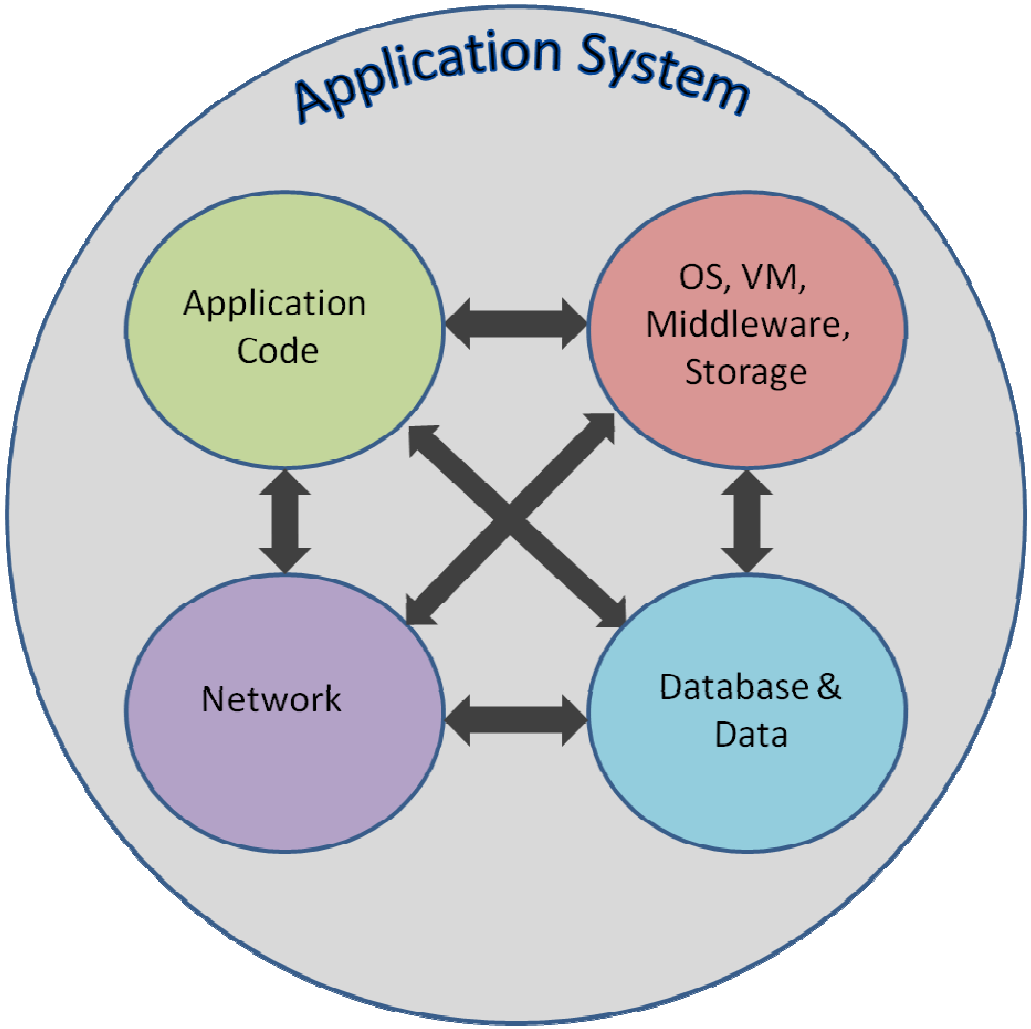
Source: Whitepaper: Improving Software Economics, Walker Royce, VP Services – Rational Software, 2009

Allow Ops to be Risk-Value Driven

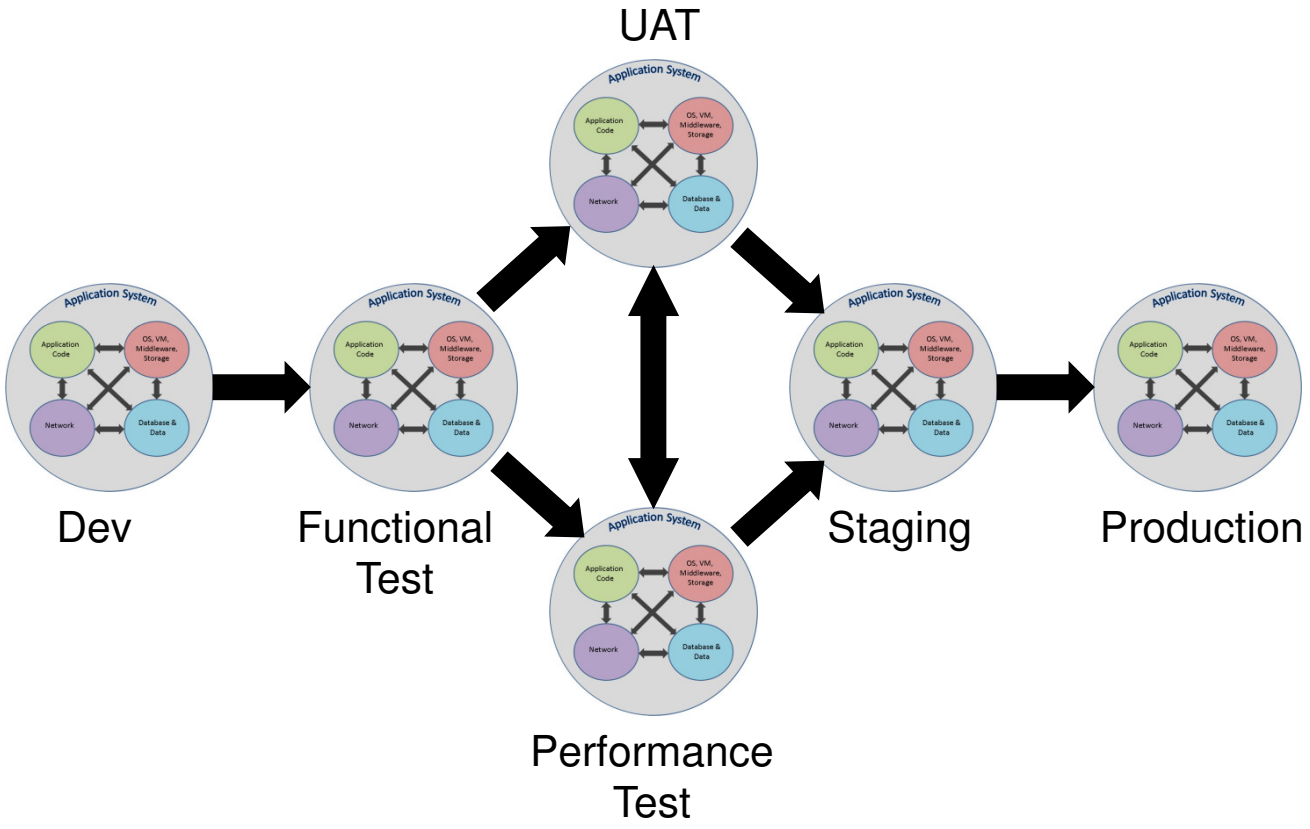
- Address common project risks, for example:
  - Stakeholder consensus around vision
  - Proving the architecture early
  - Align with enterprise direction
  - Work on things that promote learning early in the lifecycle

- Value Driven
  - Work on the most valuable things first
  - Continued assessment of project viability and business value
  - Determining when sufficient functionality has been produced
  - Potentially consumable solutions throughout the lifecycle
  - Continually assessing new work against the vision

It Is All One System



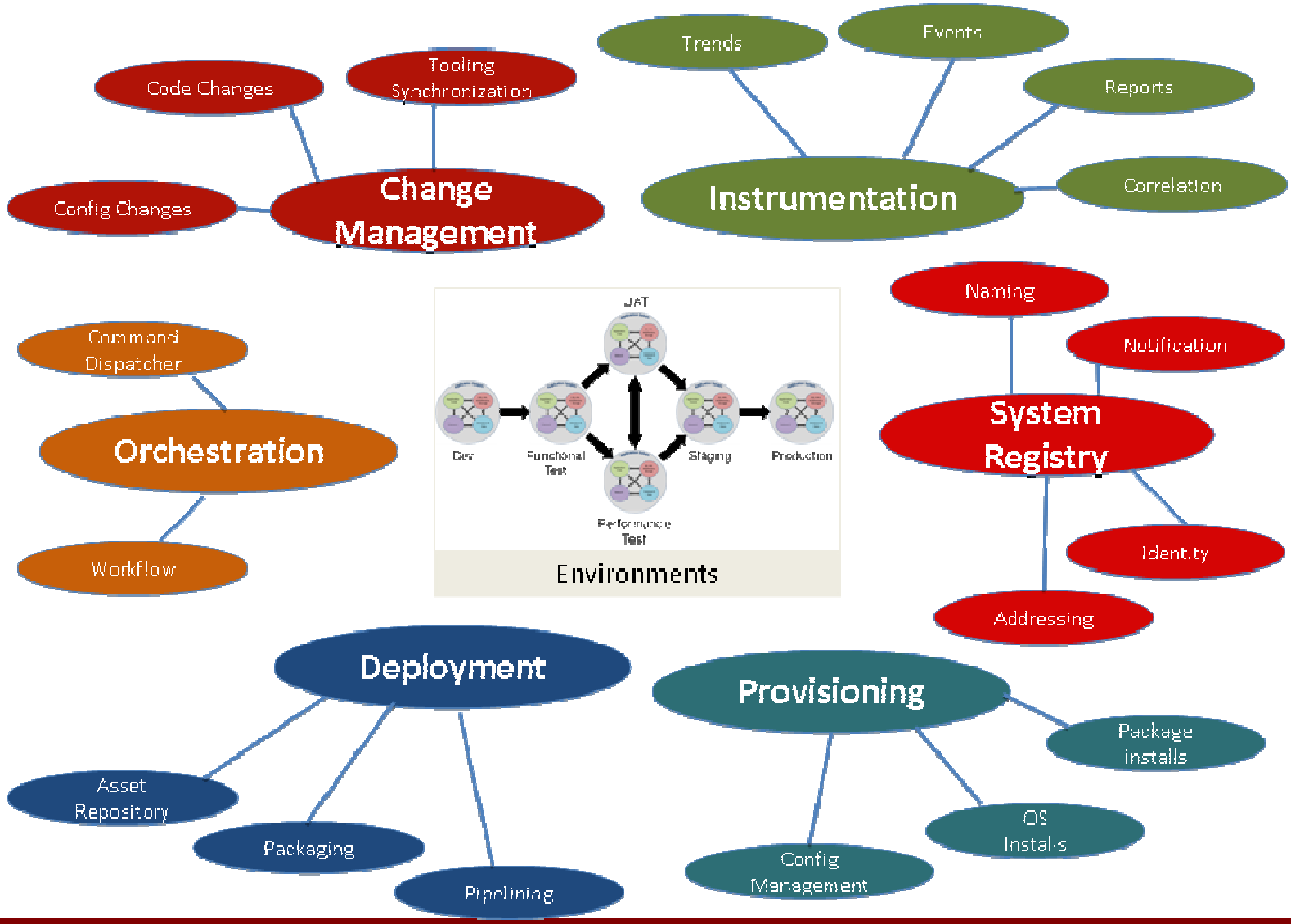
There Are Potentially Many Instances of the System



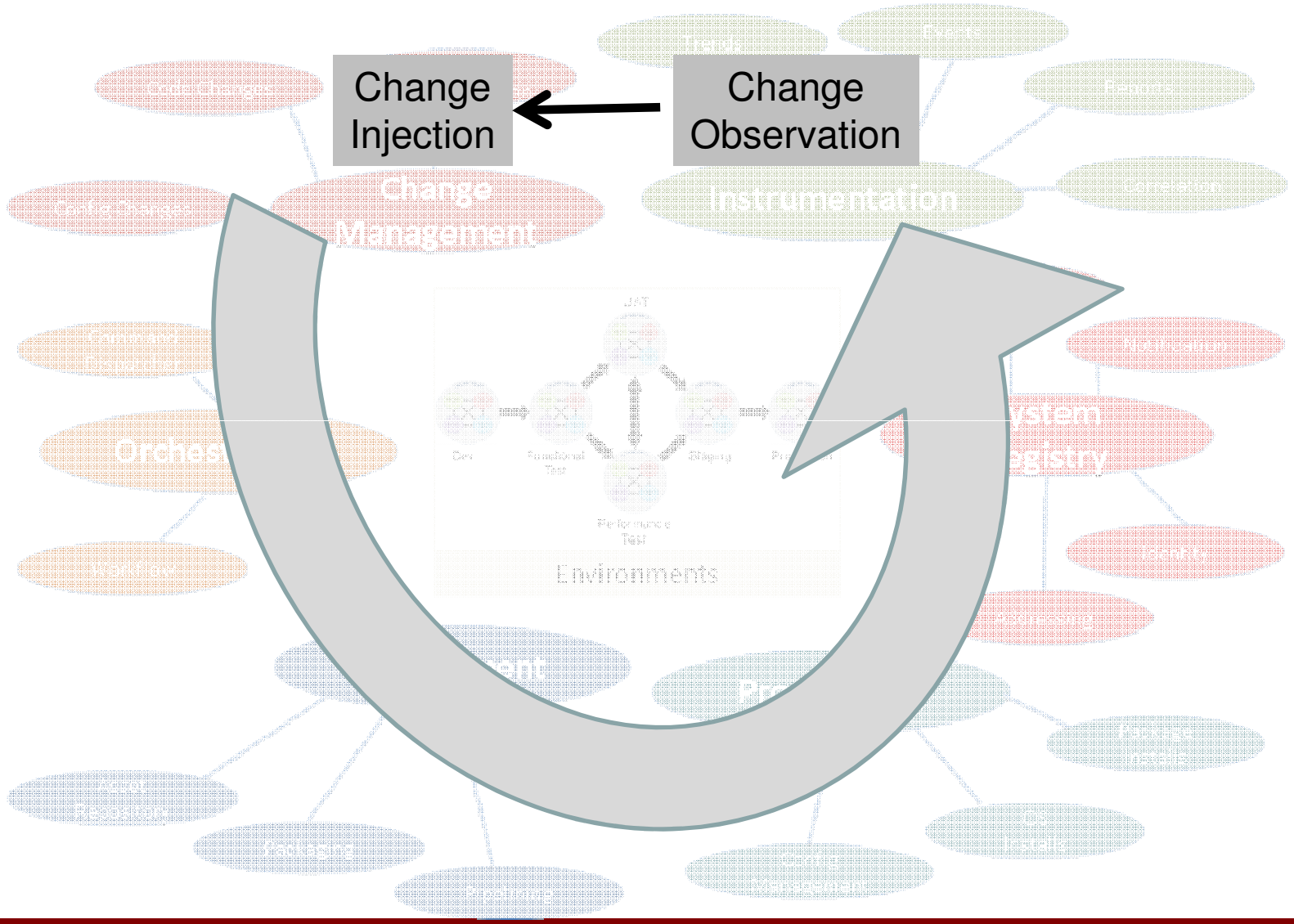
The Case for a System for Changing the System

- Application systems are complex all by themselves
- Having multiple instances with different groups of changes increases dramatically
- Many changes to those instances done frequently
- The changes originate from multiple input points
- You must ask minimal effort from the folks submitting the changes

Capabilities in a System for Changing Systems



Change Application / Review Cycle



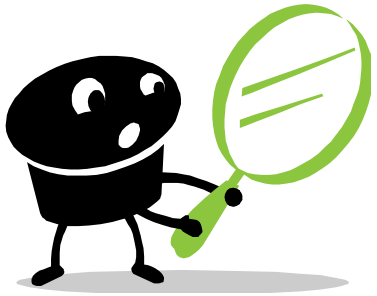
What Next?

5 STEPS TO MOVING FORWARD

0 - A Reminder

GOOD	BAD
READY	READY
AIM	FIRE
FIRE	AIM

1 - Understand Your Changes

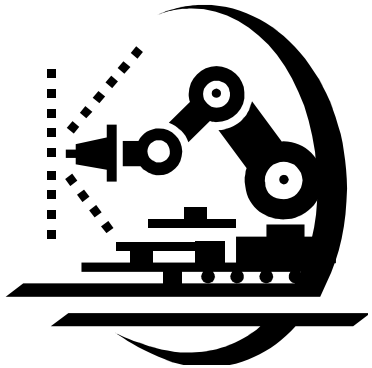


- Look at the changes independently of how they are applied
  - What they are & what they look like
  - Where they come from
- Discuss how you measure the impact of the changes on your user-facing instance of the application system

2 - Know Thyself

- Look at your process from end-to-end
  - All components
  - All environments
- Make sure that everyone understands it the same way – no matter how imperfect that understanding
- Based on that understanding figure out the most problematic area based on:
  - It takes a long time
  - It breaks a lot

3 - Start Automating



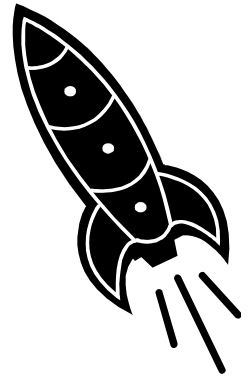
- Automate – starting with the slow broken stuff
  - Automation forces understanding and fixing of processes
  - Will reveal underlying issues

- Provides a solid demonstrable win
  - Obvious value to all
  - Helps break down barriers & build support for DevOps efforts

4 - Architect for Maintainability

- Call it resiliency if you prefer
- Application System evolution often reflects their environment's weaknesses
  - Can lock you to a way of doing things
  - Perpetuates negative behaviors
- Expect to evolve architecture
  - Build in the instrumentation
  - Feature switches
  - Loose coupling
- As always - prioritize based on value

5 - Measure the Benefit



- How much money did you save the organization?
- How much faster are you able to put a value producing feature into the hands of the users?
  - Is the delay between deployment and availability technical or a marketing decision?
- Never forget:

NO BUCKS – NO BUCK ROGERS

FINAL THOUGHTS

Some Good DevOps Behaviors

- Focus on delivering software all the way through
- Realize it is one system and plan/act accordingly
- Remember the team is a lot bigger than your little group
- Standardize how those changes are applied – EVERYWHERE
- Understand your context and your changes – ALL OF THEM
- Invest the effort in keeping the team's view unified
- Watch what the system does when changed
- Learn from the system's behavior and adapt your process
- Instrument your process to know what is most valuable to automate next at any given point in time

“To add speed, add lightness”

- Colin Chapman (1928 – 1982), founder of Lotus Cars, PLC

Additional Resources and Upcoming Events

- Agile Austin DevOps SIG (2/27 – next meeting)
  - http://agileaustin.org
- DevOps Days Austin (4/30 – 5/1)
  - http://devopsdays-austin-2013-es2001.eventbrite.com/
  - Some tickets still available
- Austin Cloud User Group (2/19)
  - http://acug.cloudug.org
- DevOps Weekly mailing list
  - http://devopsweekly.com
- InfoQ has a DevOps section

THANK YOU