DevOps from Within

The Art of Automated Testing

About Tom Gilmore

Tom Gilmore (www.linkedin.com/in/gilmoretom) is a coach who specializes in Agile/Automation/DevOps coaching. Tom is able to help guide organizations through all aspects of their journey to Agile development, automated testing or DevOps.

With over 20 years of experience in quality assurance, Agile development and automated testing. Tom is able to help organizations focus on developing solutions to improve quality and decrease release time through DevOps, Agile development and automated testing.

He has coached and trained a multitude of organizations. The short list includes, Forever Living, Evaluate to Win, Able Engineering, Glynlyon, Choice Hotels International, Discount Tire, Concord Servicing, First Solar, Edgenuity and Agworks. Tom enjoys his work, and does everything to ensure he shares his knowledge with others who seek it, by founding groups such as the Arizona DevOps Selenium Group.

Do you Recognize These?

- Kaizen?
- Kanban?
- JIT (Just in Time)?
- Gemba?
- Value Stream Mapping?
- WIP (Work in Progress)?
- Jikoda?
- Pull Flow?
- Continual Improvement?
- Takt Time?
- Cycle Time?
- Andon Cord?
- Theory of Constraints?
- Heijunka?
- PDCA?

Father of Quality and TQM

- W. Edward Deming is often referred to as the father of quality and his ideas laid much of the groundwork for what is today called Six Sigma. His unique fusion of science and human engineering have left a profound impact on modern management.

- Deming's theories about variation and knowledge are absolutely relevant to where DevOps should be today and where it might be going in the future

- TQM is a management approach to long–term success through customer satisfaction.

- In a TQM effort, all members of an organization participate in improving processes, products, services, and the culture in which they work.

Lean Manufacturing's Importance

- **Lean manufacturing -** is a systematic method for the elimination of waste ("Muda") within a manufacturing system. Lean also takes into account waste created through overburden ("Muri") and waste created through unevenness in work loads ("Mura").

- In order to understand the affect DevOps is having on the workplace, we need to look at the Lean transformation to manufacturing in the 1980's
  - Lean brought about a revolution to traditional manufacturing
  - Toyota was at the forefront of this revolution
    - Four times the productivity with twice the output
    - Lean adoption brought unprecedented increase in worker productivity, better delivery performance, reduced inventories and higher customer satisfaction and employee happiness

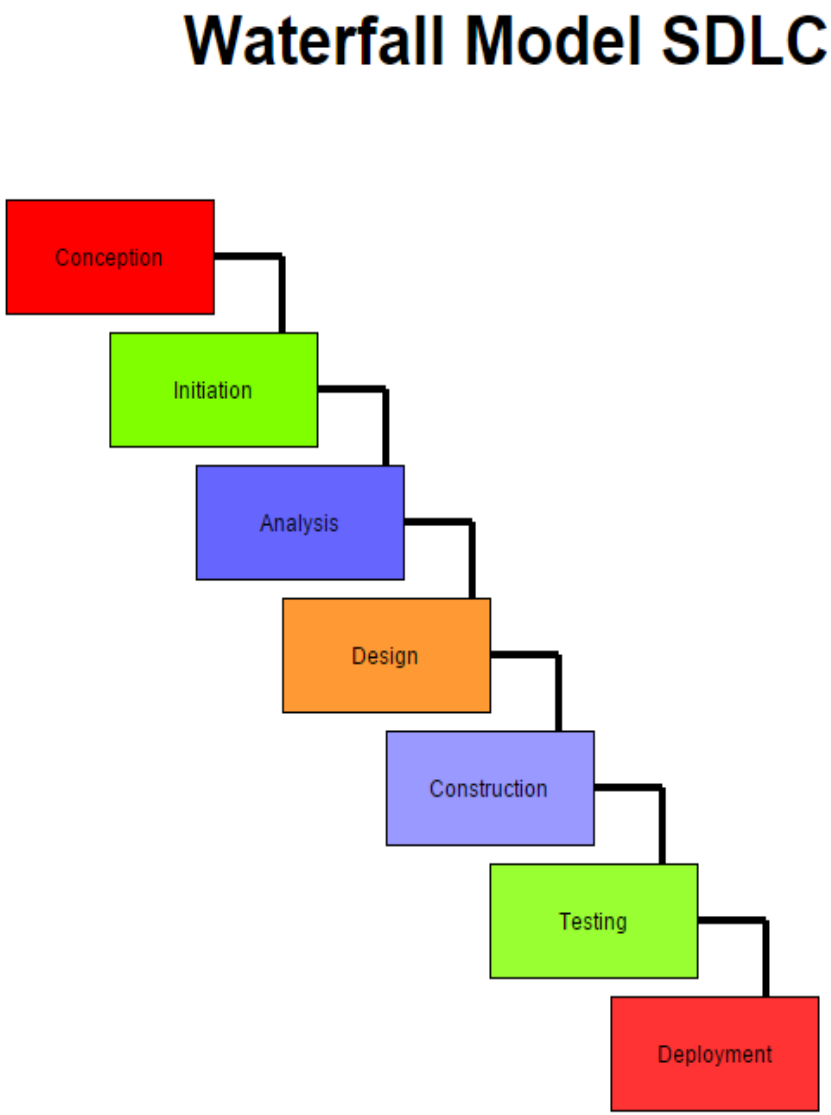
Lean Manufacturing's Importance

- In order to reap the benefits of DevOps, requires organizations to change their culture and practices. In large complex organizations this is seen as being too difficult
- The challenges faced by the Lean revolution
  - The Lean revolution was also extremely difficult
  - Required changing incentives, overcoming work center silos, training/re-skilling employees and focusing on flows instead of units produced
  - Organizations that did not adopt Lean became irrelevant
- DevOps could possibly have a bigger impact on business value then the manufacturing revolution
  - IT is the new factory floor for almost every type of company
  - 95% of all capital projects have an IT component and 50% of all capital spending is technology related.
  - IT is increasingly how businesses acquire new customer and deliver value to them

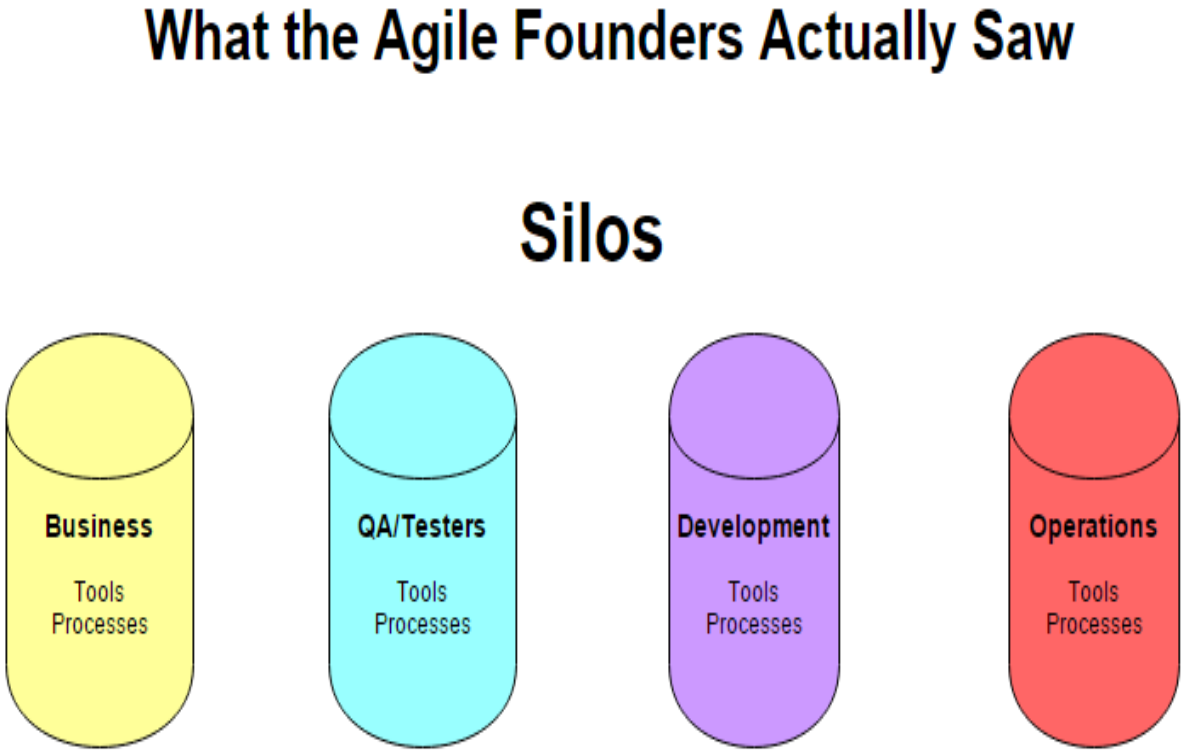
Lean Manufacturing's Importance

- DevOps is even more difficult than the Lean Manufacturing revolution
  - IT work is often invisible, making it more difficult determine WIP
  - Most organizations are trapped in low trust command and control cultures that thrive on fear. Fear prevents experimentation and innovation
  - This culture has led to an "order taking" mentality in IT
  - IT has come to view itself as a service provider to the organization, instead of a critical part of delivery value to the customer
- Organizations that trivialize the difficulties facing a DevOps adoption as simply "technical problems" are putting their organizations at grave risk
- Organizations adopting DevOps will likely see productivity surge at a level as high or higher than the Lean manufacturing revolution
- Just like the Lean manufacturing revolution, there will be winners and losers. Borrowing from Dr. W. Edwards Deming – "**Learning is not compulsory, but neither is survival**"

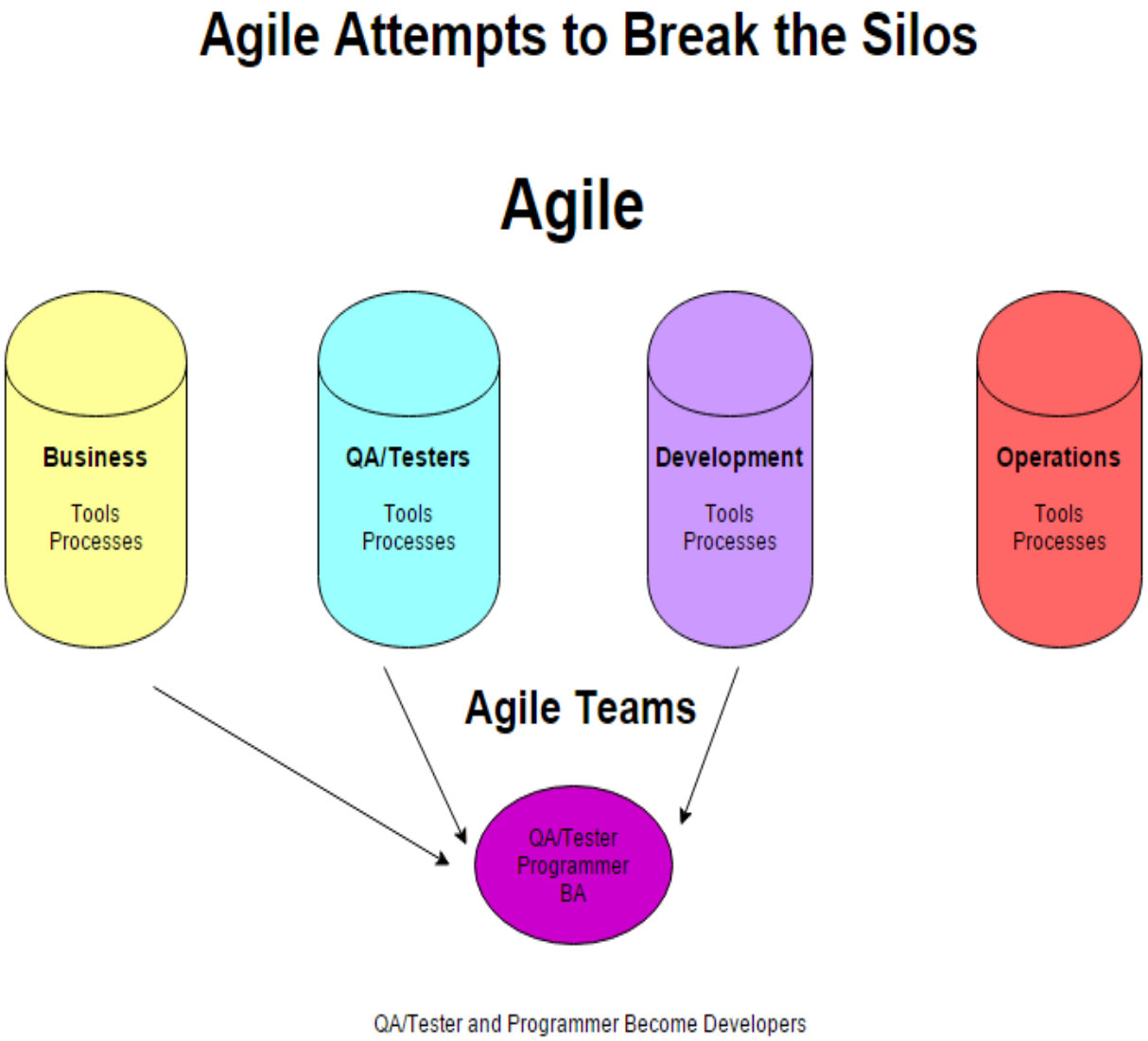
How We Got Here



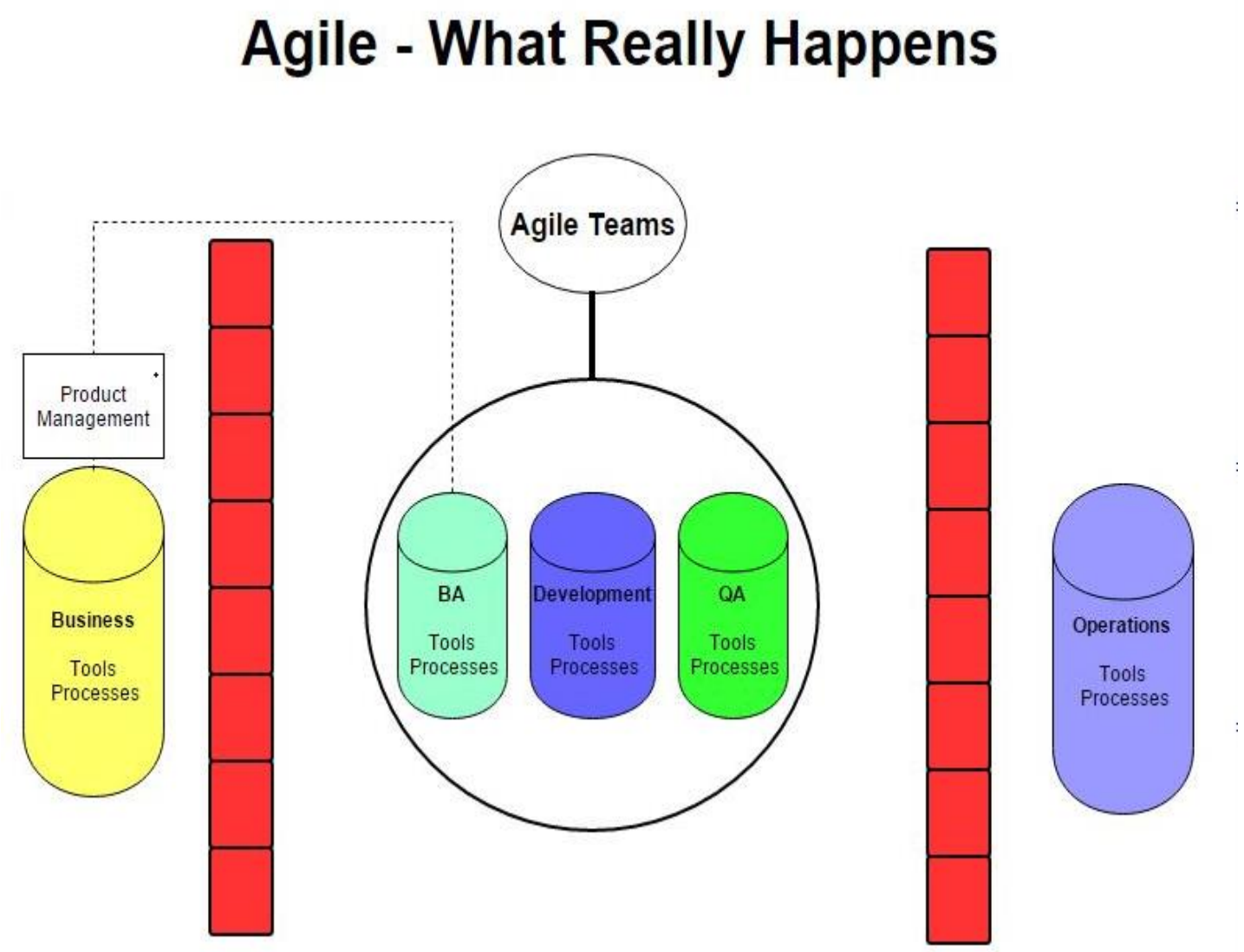
How We Got Here



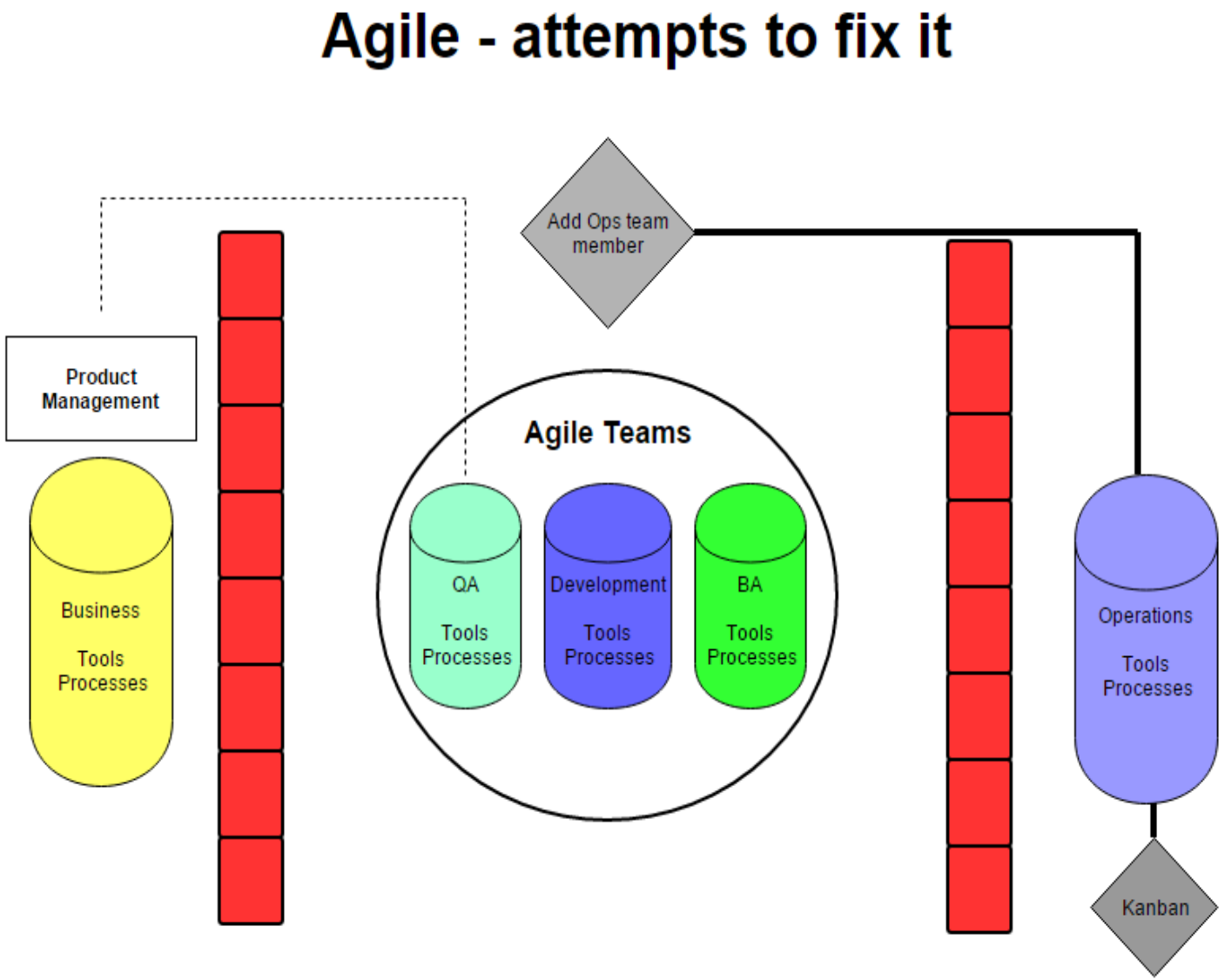
How We Got Here



How We Got Here

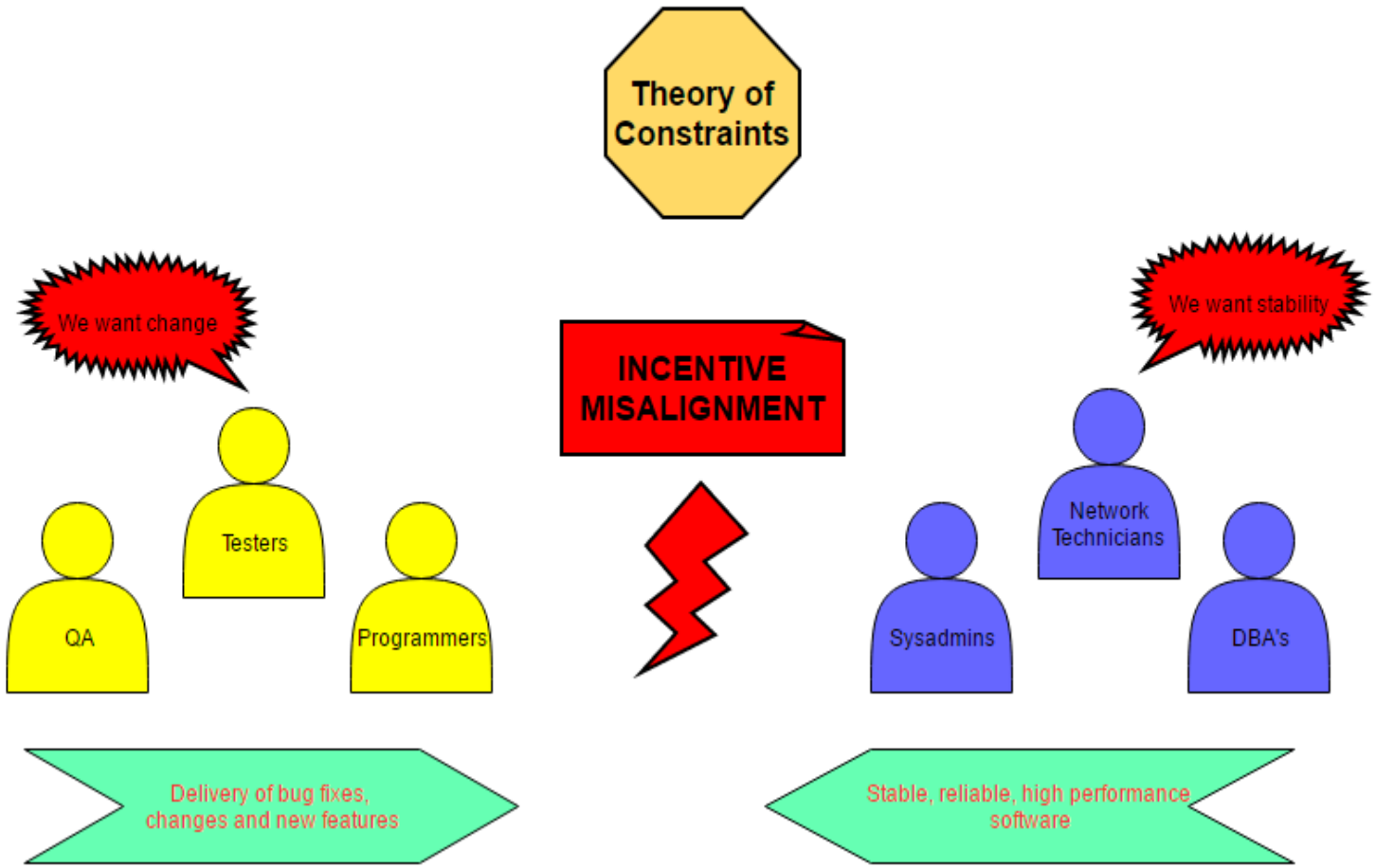


How We Got Here



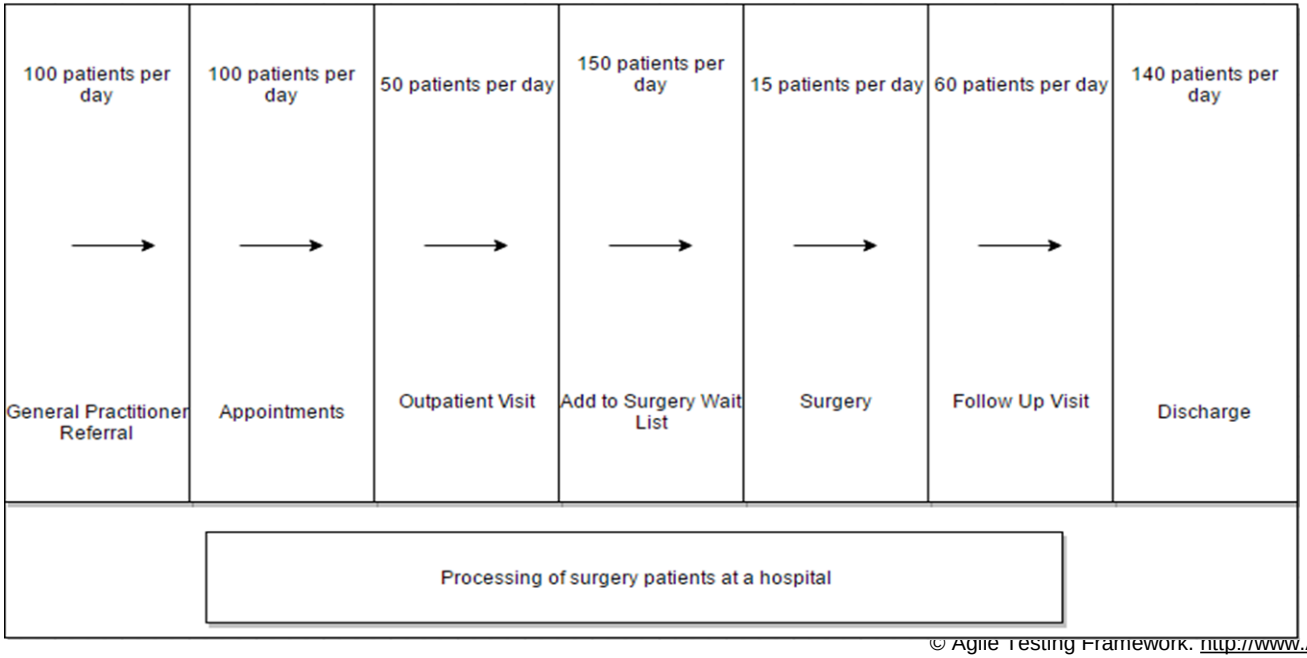
How We Got Here

Why the attempts fail



Theory of Constraints

- The Theory of Constraints (TOC) is a management philosophy introduced by Eliyahu M. Goldratt in 1984.
- It has become a bedrock methodology used in Lean Manufacturing and numerous other industries.
- The **Theory of Constraints** is a methodology for identifying the most important limiting factor (i.e. constraint) that stands in the way of achieving a goal and then systematically improving that constraint until it is no longer the limiting factor.



DevOps History

2008 - Patrick Debois coins the term DevOps while organizing a DevOpsDays conference in Belgium.
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."

NOTE: It is important to remember that the term DevOps is widely used in the IT industry these days, and many different types of connotations are associated with it.

CIO survey in 2015 – 79% of companies in the US and 66% of companies worldwide had plans to start adopting DevOps **practices!!!!!!**

The reality is that about 1% of companies worldwide actually do DevOps.

DevOps?

DevOps is an approach to close the gaps between development and operations by aligning incentives and providing shared approaches for processes and tools. In DevOps we also broaden the usage of Agile practices to operations in order to foster collaboration and streamline the entire software delivery process in a holistic way.

DevOps Is Not!

- DevOps is not a buzz term.
- DevOps is not a new department or silo.
- DevOps is also not a new role profile or a new job title.
- There is no one-size-fits-all solution, and no “DevOps-by-the-book” approach
- There are a number of vendors advertising a one size fits all solution, its simply fools gold.

DevOps Is!

- Lightweight tools
- Brings Agile approaches to all parts of the streamlined software delivery process
- DevOps does not necessarily introduce new tools.
- People and processes are more important than the tools.

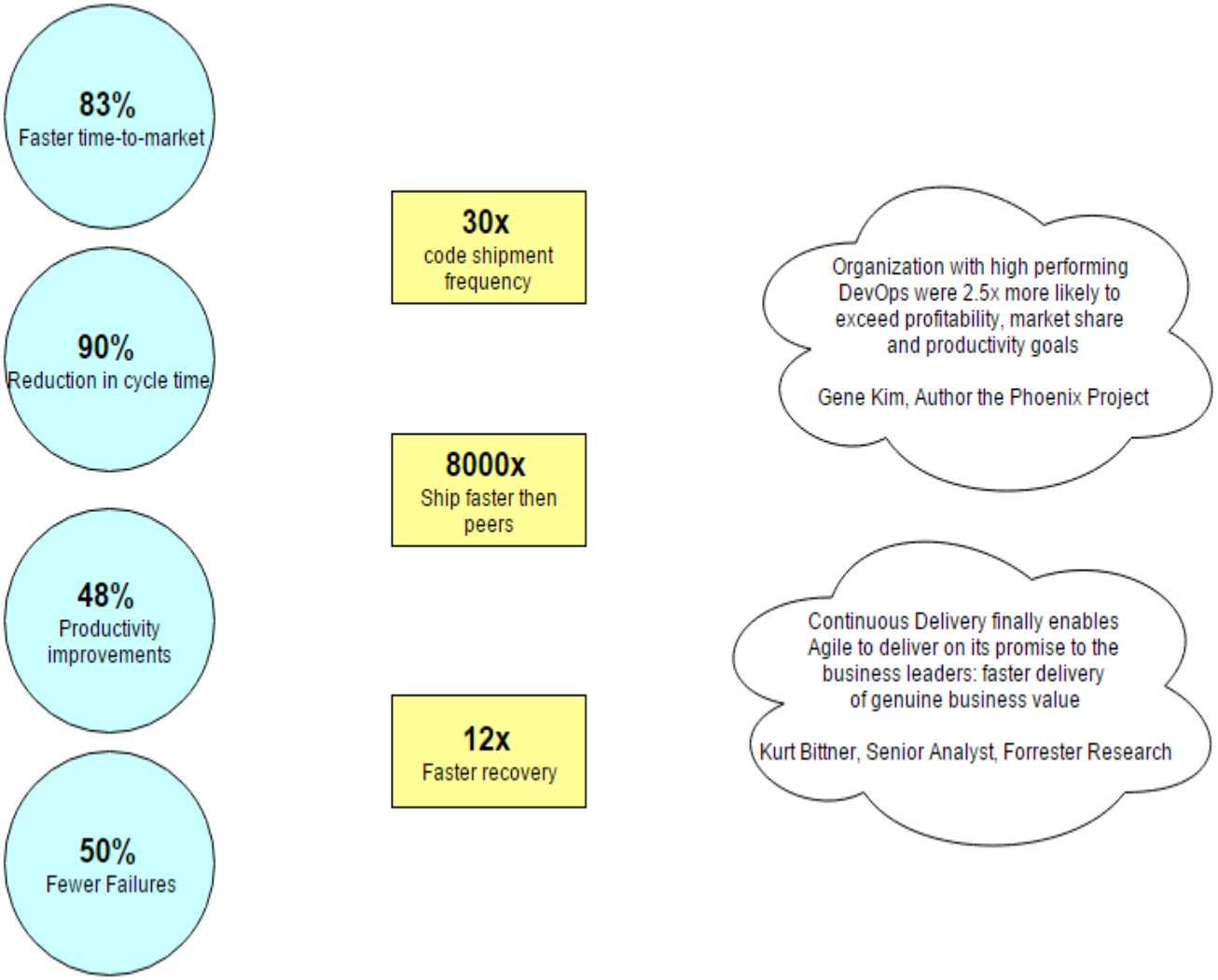
DevOps Companies and Success Stories

Internet Based - Google, Facebook, SalesForce, Linkedin, Etsy, Spotify, Netflix, Twitter, Amazon, Paychex, Intuit
Insurance – State Farm, Nationwide, Blue Cross of California, USAA
Retailing – The GAP, Nordstrom, Macys, Williams Sonoma, Sherman-Williams
Financial Institutions – BNY Mellon, Bank of America, ING, GE Capital, E-Trade, Quicken Loans, Western Union, Capital One, Loyds, Rabobank
Government/Education - Kansas State University, UK Government, Department of Homeland Security
Engineering/Software - Dynatrace, CSC, SAP, CA, Red Hat, Raytheon
Entertainment – Disney, LEGO, Sony, Orbitz

Success Stories

- Capital One, 6x increase in delivery frequency
- Bank of America, 25x increase in delivery frequency
- Bank of America, 6x production defect reduction in an environment where a single high severity fix can cost up to $75k
- CSG International, leveraged test automation to replace legacy components, achieving an estimated savings of 9 Million dollars

Advantages of DevOps



Traditional Project Settings

Traditional Project Attributes

- Hero Cult
- Emphasis on Titles
- Shadow Responsibilities
- Favor a Plan over Planning

Leads to Organizational and Cultural Barriers

- Separate Teams
- No Common Language
- Fear

Blame Game – Dev vs Ops (conflict during deploy, after deploy and performance)

Operations is seen as the bottleneck (by management and business)

DevOps Project Settings

Selecting DevOps Team Members

- Innovators
- Followers
- Laggards

DevOps Team Fundamentals

- Transparency (no skunkworks)
- Traceability
- Cross Functional teams with end-to-end ownership of services
- Evangelize
- Openness and Honesty
- Common Language (Specification by Example)
- Moving from a “ticket culture” to an “ownership culture”

Kaizen - A strategy where employees work together proactively to achieve regular, incremental improvements in the process.

DevOps Kaizen Culture

Starts Within

The DevOps transformation journey has shown a clear evolution pattern, what started as essentially bottom-up, grassroots movements, often starting with more tractable low hanging fruit such as automation and tool adoption, have now shifted their focus to disseminating their learnings and effectively scaling DevOps (first raising awareness, then practices) across teams, departments and even geographies.

- Local vs. Global Thinking
- Respect
- Trust
- No Victims
- The Smell Test
- Shaman
- Slaying the Dragon
- Fearless Behavior
- Piercing the IT Veil
- No Assholes

Building Blocks of DevOps

Measurement and Metrics

- "Change" serves as a shared unit of measurement.
- Cycle Time
- Definition of Done
- SonarQube (Required to be part of CI, quality gates)
- Measure DevOps Progress (Specification by Example)
- Measure Cultural Change (surveys)

Improve and Accelerate Delivery

- Incremental releases
- Versioning and Baselines
- SCM Required
- Applies to App, DB and Infrastructure code
- By reducing batch size, we can deploy more frequently because reducing batch size drives down cycle time.

Heijunka (Level Scheduling) - Reduces lead times (since each product or variant is manufactured more frequently) and inventory (since batches are smaller).

Building Blocks of DevOps

Automatic Releasing

- The key goal of automatic releasing is to reduce the risk of releasing.
- Should you automate everything? NO
  Law of Marginal Costs
  Noun/Verb Mistake. Testing is an activity (i.e., it's not a noun).
  Understanding testing as a noun is suboptimal and may
  lead to a focus on meaningless artifact types (e.g., test cases)
  instead of concretely and effectively testing the application.
- Required to flip the testing pyramid

Jidoka - After Jidoka, workers can frequently monitor multiple stations and many quality issues can be detected immediately (improving quality).

Building Blocks of DevOps

Apply Releases Incrementally and Iteratively

- An iteration is a mini-project that may result in an increment of the software. Iterating starts with an idea of what is wanted, and the code is refined to get the desired result. An increment is a small unit of functionality. Incrementing allows you to build a better understanding of what you need, assembling the software piece by piece.
- Upgrading all of the components in one big-bang release is the highest-risk way of rolling out new functionality.
- Instead, deploy the components independently in a side-by side configuration wherever possible.

Continuous Flow – Development process where work-in-process smoothly flows through production with minimal (or no) buffers between steps of the development process.

Building Blocks of DevOps

Monitoring

- Automatic releasing must be accompanied by monitoring.
- Monitoring is the activity of continuously collecting and storing data about the state of the application, middleware, and infrastructure and making this state visible to the whole team.
- Monitoring is used to detect (or even prevent) production incidents and to minimize MTTR and MTTD. (Mean Time to Resolution/Detection)
- Monitoring Driven Development – The Holy Grail – Health Dashboard
- Running Automated Tests in Production – Yes
- Smoke Tests – There is a reason we do this
- DevOps fundamental - Never break the customer (internal or external)
- Feedback Loops

Poka-Yoke - Design error detection and prevention into the production processes with the goal of achieving zero defects.

Building Blocks of DevOps

Decoupled Deployment and Release

- Decoupling deployment and release improves and accelerates delivery, which is one building block of DevOps.
- We no longer use traditional branching strategies.

Branch by Abstraction – Paul Hammant Thoughtworks

- Create an abstraction over the part of the system that you need to change.

Feature Toggles

- The concept of a feature toggle is to deliver the complete code to production but use data-driven switches to decide which feature is made available during runtime.

Dark Launching – Facebook

- Dark launching is the strategy of deploying first versions of functionality into production before releasing the functionality to all users.

Blue-Green Deployment

- The core of this strategy is that we deploy the new version of the application side by side with the old version.

Building Blocks of DevOps

Specification by Example

- Behavior Driven Development/Acceptance Driven Development (BDD/ATDD)
- DevOps requires a common language (Release example)
- Living/executable documentation
- It provides the traceability for everything your doing
- Provides a common language

Visual Factory - Makes the state and condition of manufacturing processes easily accessible and very clear – to everyone.

Infrastructure as Code

- We treat the infrastructure the exact same as application code.
- Lives in an SCM
- Versioned
- Baselined
- Is tested just like application code (TDD)
- Is monitored constantly just like application code

Benefits and Leading Practices

Benefits

- Security
- A/B Testing
- Performance and Load Testing
- Chaos Monkey (Andon Cord)
- Mobile (treat just like binary files)

Lead Practices

- Dojos
- Andon Cord
- Internal DevOps Days or software engineering conferences
- Engineering blogs
- Communities of practice
- DevOps/IT academies
- Hackathons

Starting DevOps From Within

- **Find** - a senior level management sponsor.
- **Gemba** - Talk with as many people across the business as possible.
- **Look** - for people involved in the process of getting software from conception to the consumer and supporting it when it's live.
- **Talking** - Provides an opportunity to evangelize and form a wider network of like minded people.
- **Minimum Viable Product –** Deliver the smallest viable product.
- **Dedicated Teams –** Form dedicated, cross-functional teams, avoid over-specialization and ensure they are not interrupted by side projects.
- **Loosely Coupled Architecture –** Use loose architectural coupling within and between applications.
- **Minimize Hand-offs, Maximize Flow –** Eliminate unnecessary steps, delays and friction between steps to increase the flow of work.
- **Deliver in Small Batches –** Deliver in smaller batches helps to expose the most uncertain issues first and enables feedback on the most valuable usage models earlier.
- **Transparency –** (ATDD/BDD)
- **Eliminate Overhead –**Eliminate periodic and manual status reports and status meetings.
- **Automate Testing using APIs –** (Smoke Suites for quick ROI)

Any questions...

Learn more at www.AgileTestingFramework.com

Learn more at www.torak.com

This presentation was inspired by the work of many people and we have done our very best to attribute all authors of texts and images, and recognize any copyrights. If you think that anything in this presentation should be changed, added or removed, please contact us.