CONTINUOUS TESTING: THE NEW NORMAL

TOM WISSINK, NDIA T&E CONFERENCE, MARCH 2017

SESSION ABSTRACT

Continuous testing (CT) is a term being used more and more in the Commercial software development arena. While some have mastered CT, most of us (including in the Government software arena) struggle with how to transform our current testing approaches to CT approaches and align them with evolving development methodologies. This presentation will review current examples of CT implementations across different software development methodologies (agile, waterfall, incremental) and describes where CT type testing yields the best benefits. Arguably the most challenging methodology that demands CT testing is DevOps. DevOps requires all phases of testing to be done quickly and in parallel with the development process and some contend that testing continues into actual operations. Leave this session with a better understanding of CT, and how this approach can be best leveraged in your software development environment for Government systems (including Weapons Systems).

AGENDA

- INTRODUCTION
- TEST TYPES DEFINED
- DEVELOPMENT METHODOLOGIES
- CONTINUOUS TESTING (CT) APPROACHES
- “THE TOOLCHAIN”
- SUCCESSES WITH CT
- WRAP-UP
- FINAL COMMENT

TEST TYPES DEFINED

Goal: High test coverage for each test type or identify risk for reduced coverage

Unit Testing (White-Box)

Statement, Condition, Decision & Path

Metric is Logic Coverage – tools available

Integration Testing (Interface Testing: this is generally very weak)

Identify threads through system

Threads part of incremental/build plans

Early look at Technical Performance Measures – TPM's (stability, performance, RMA, etc)

Metric is Interface coverage – external & internal interface

TEST TYPES DEFINED (CONT'D)

Functional Testing (Black-Box)

Boundary Value Analysis, Output Forcing, Equivalence Class Partitioning, Cause & Effect Graphing, Combinatorial, etc.

Metric is Function coverage – SRS requirements, etc

System Testing (Official Sell-Off)

Types - Scenario-Based, Risk-Based, Exploratory, Model-Based, etc

Includes several categories of requirements – HMI, Non-Functional (Performance, RMA, Configuration, Installation), etc.

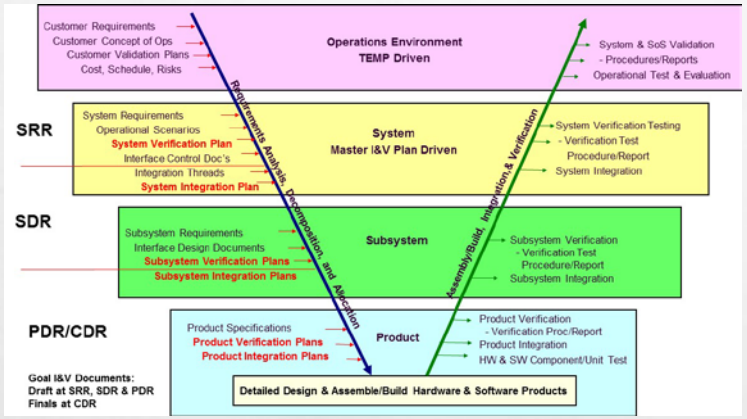
Metric is Requirements coverage – System Spec, Ops Concept/Scenarios, etc

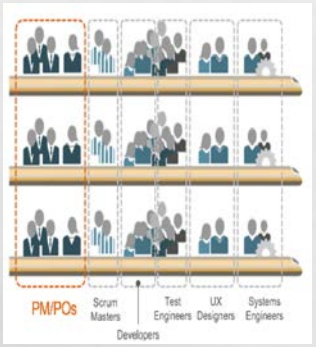
Acceptance Testing (Customer Testing)

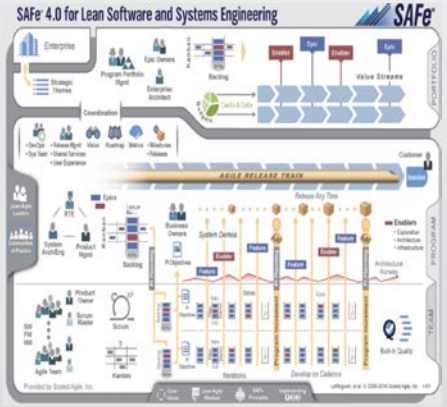
Operational Test & Evaluation (OT&E), Beta, etc

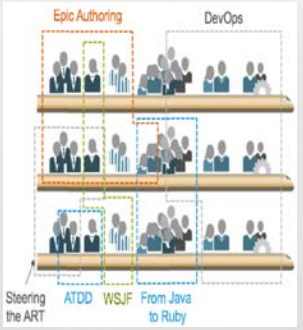
Metric typically owned by Customer but good to know

DEVELOPMENT METHODOLOGIES

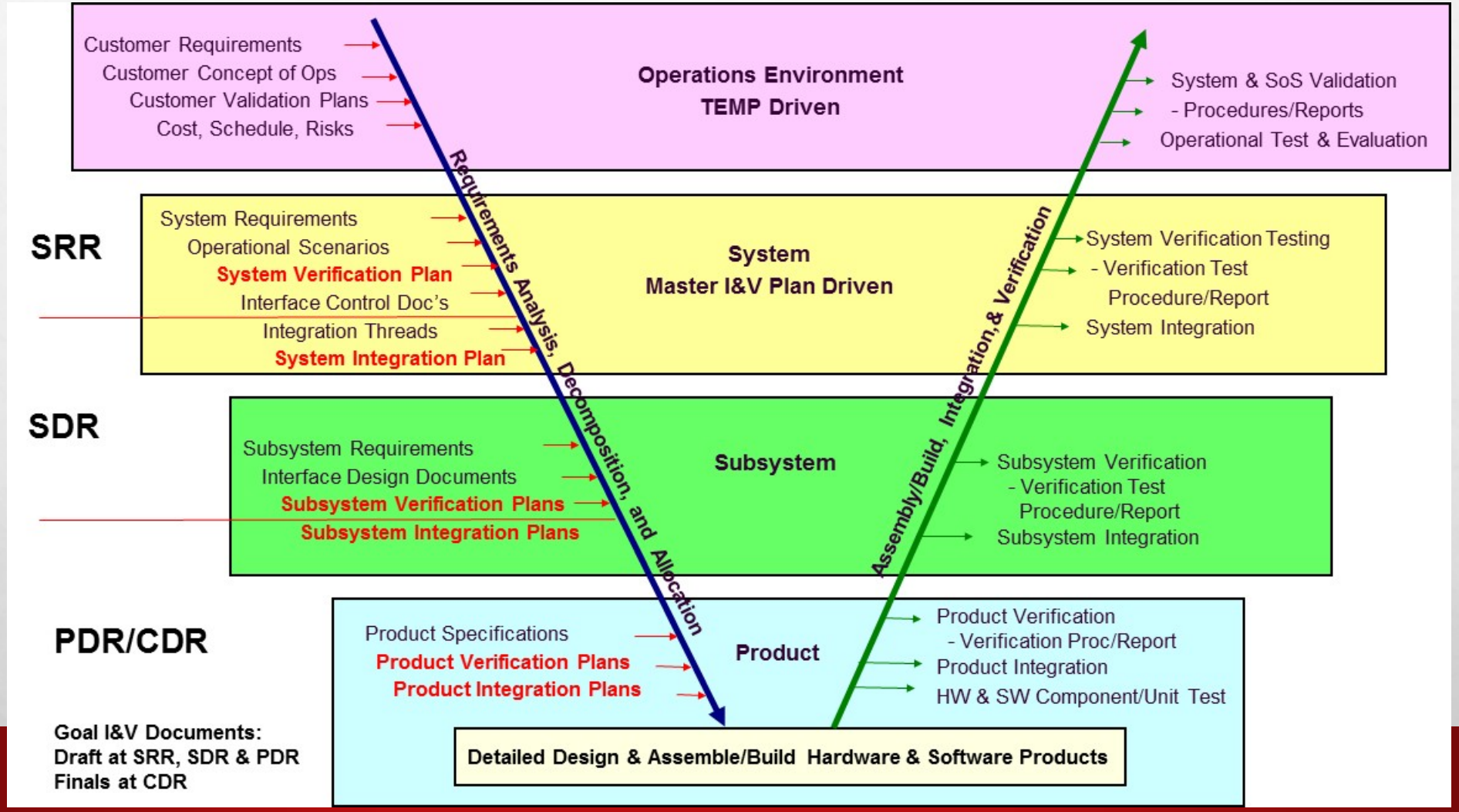






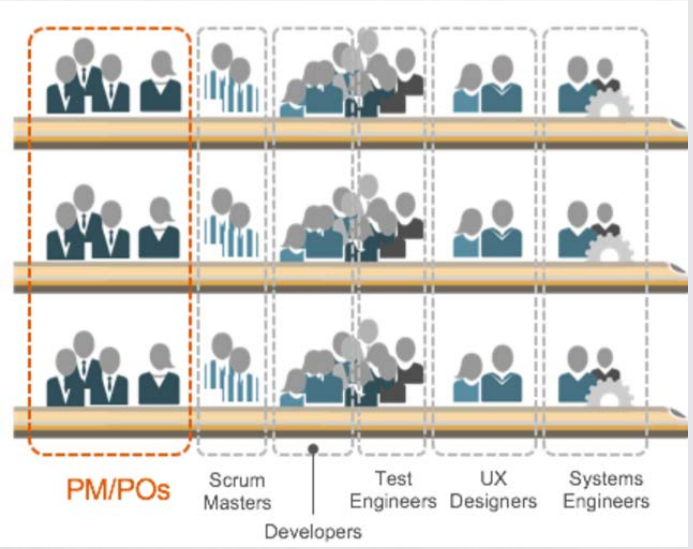


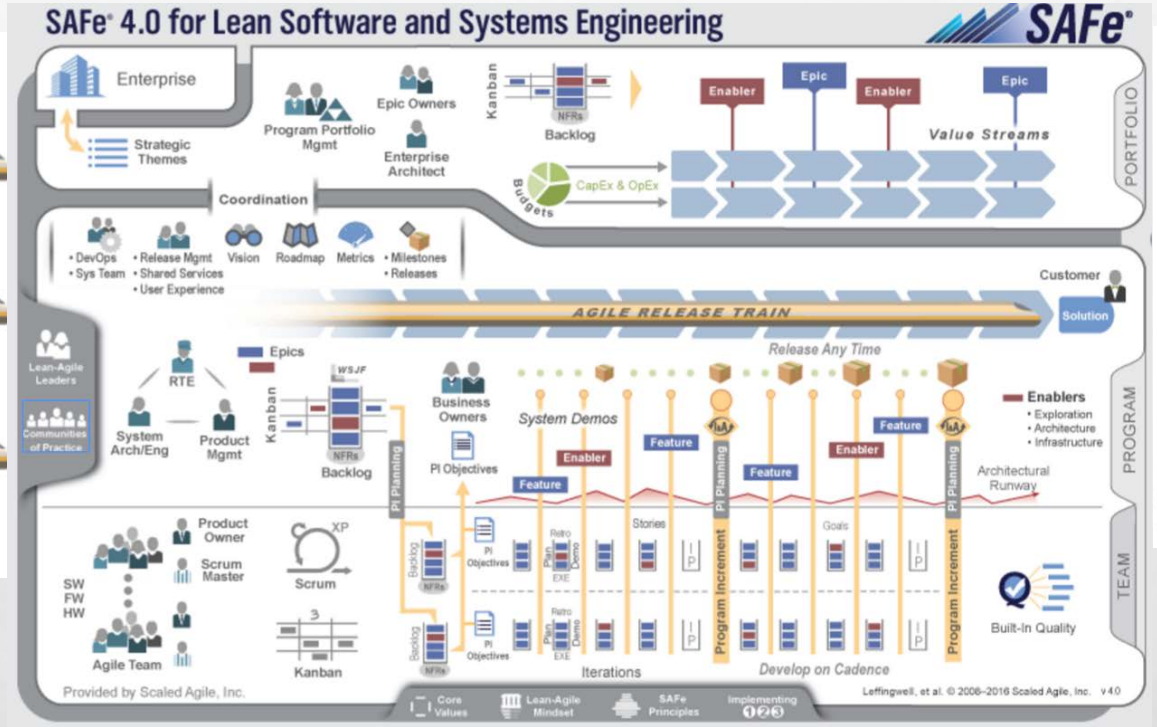
Effective and Efficient Test Lifecycle in any Software Development Methodology (WF, Incremental, etc.)

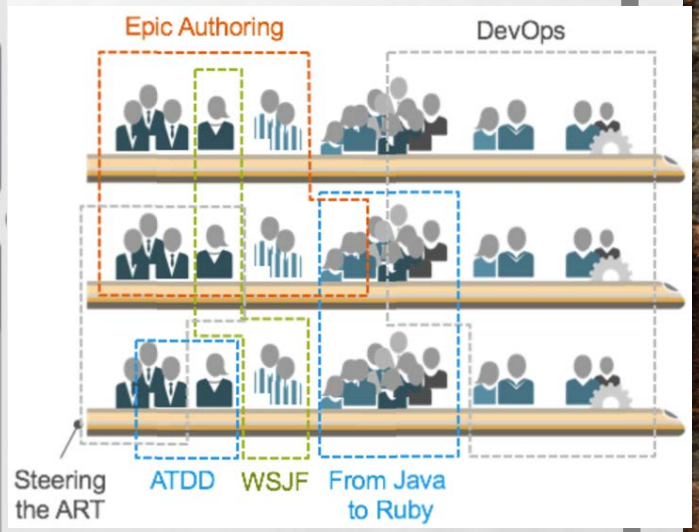


Effective and Efficient Test Lifecycle in any Software Development Methodology (Agile, DevOps, etc.)

Effective and Efficient Test Lifecycle in any Software Development Methodology (Agile, DevOps, etc.)



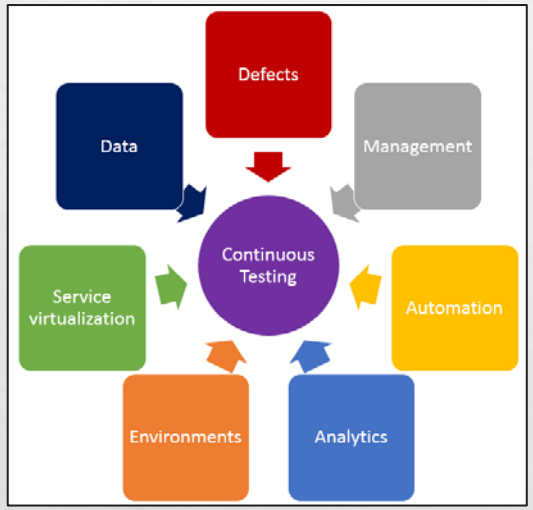


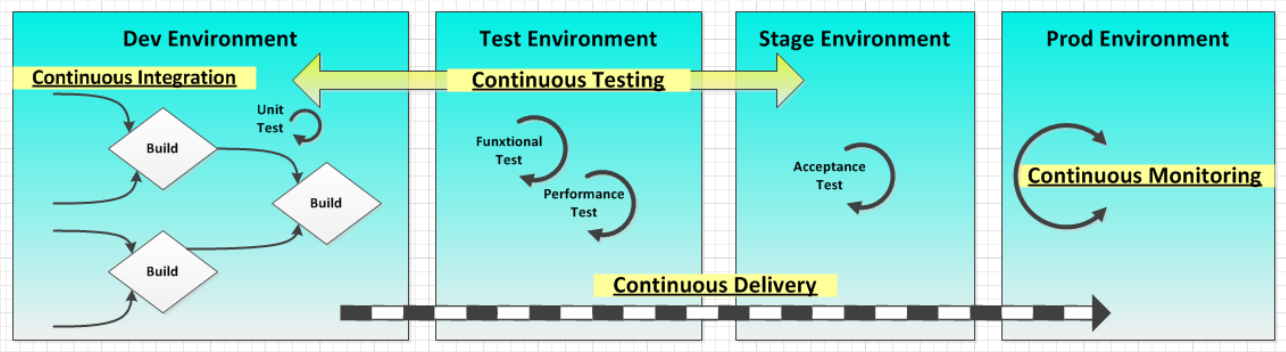


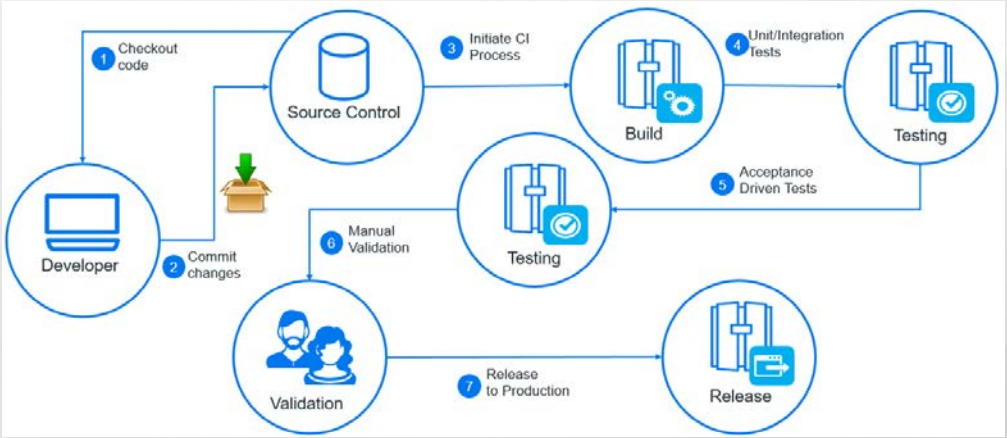
See SAFe at www.scaledagileframework.com

ART – Agile Release Trains
ATDD – Acceptance Test Driven Development
WSJF – Weighted Shortest Job First

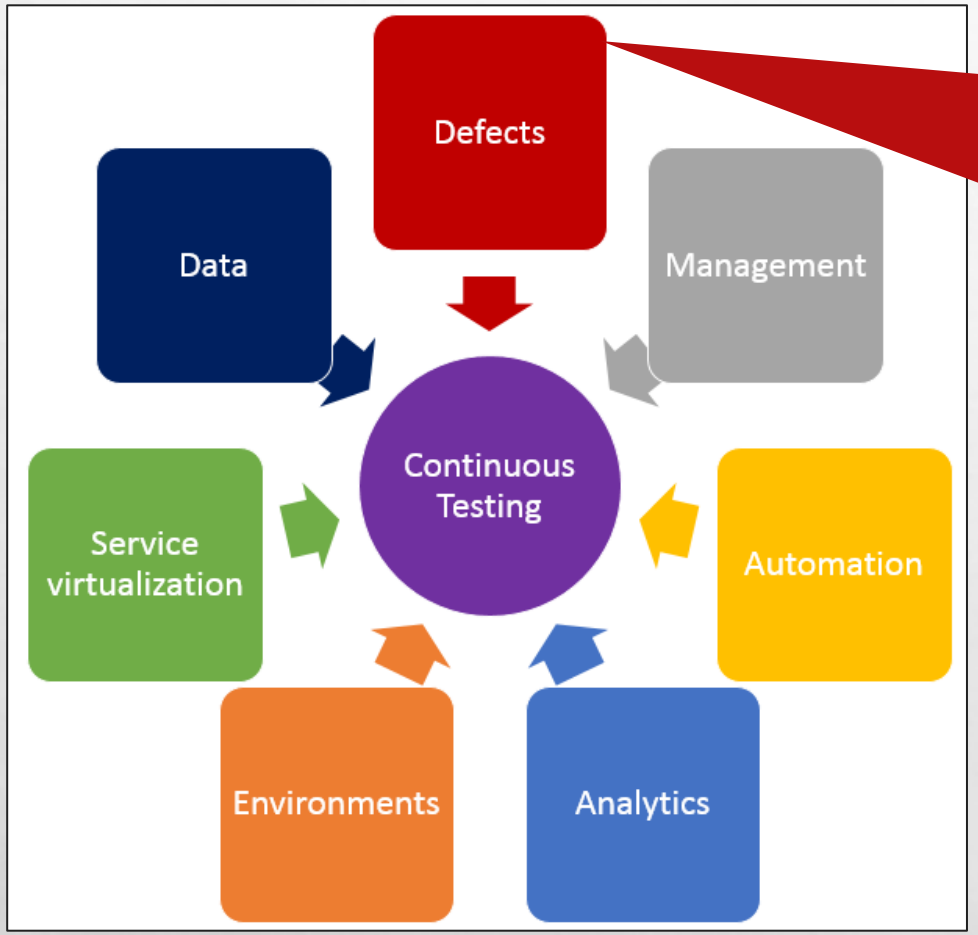
CONTINUOUS TESTING APPROACHES





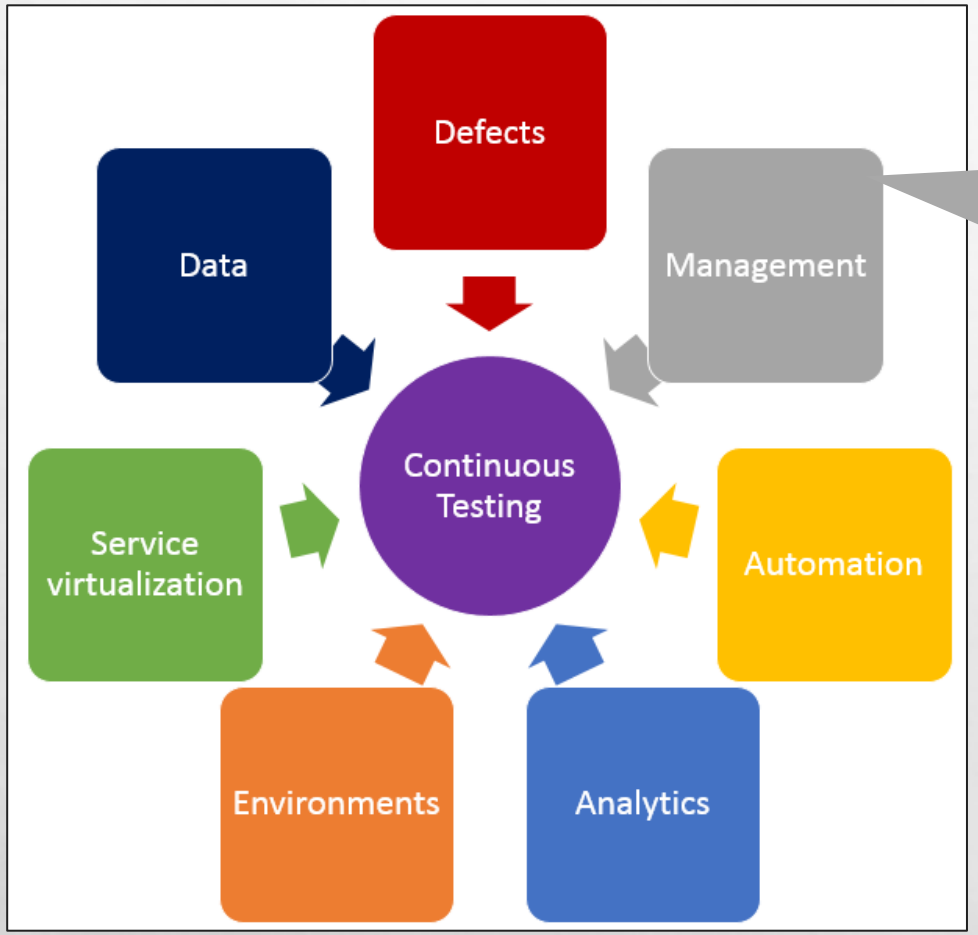


“Continuous Testing: Shift Left and Find the Right Balance”



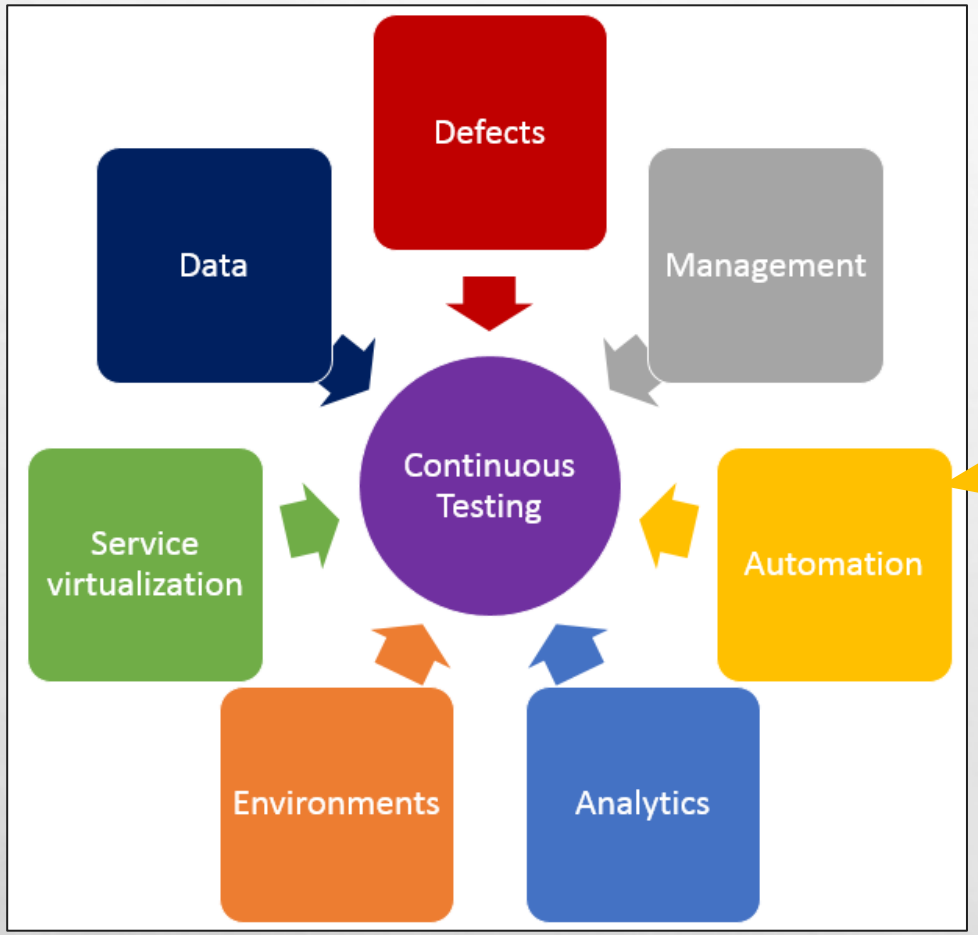
Defects – Are teams spending too much time logging, triaging or analyzing defects? What about time spent on defects that aren’t “real” defects—where there is a misunderstanding between the test and the code? Or what if they could prevent entire schools of defects from ever being created in the first place?

“Continuous Testing: Shift Left and Find the Right Balance”



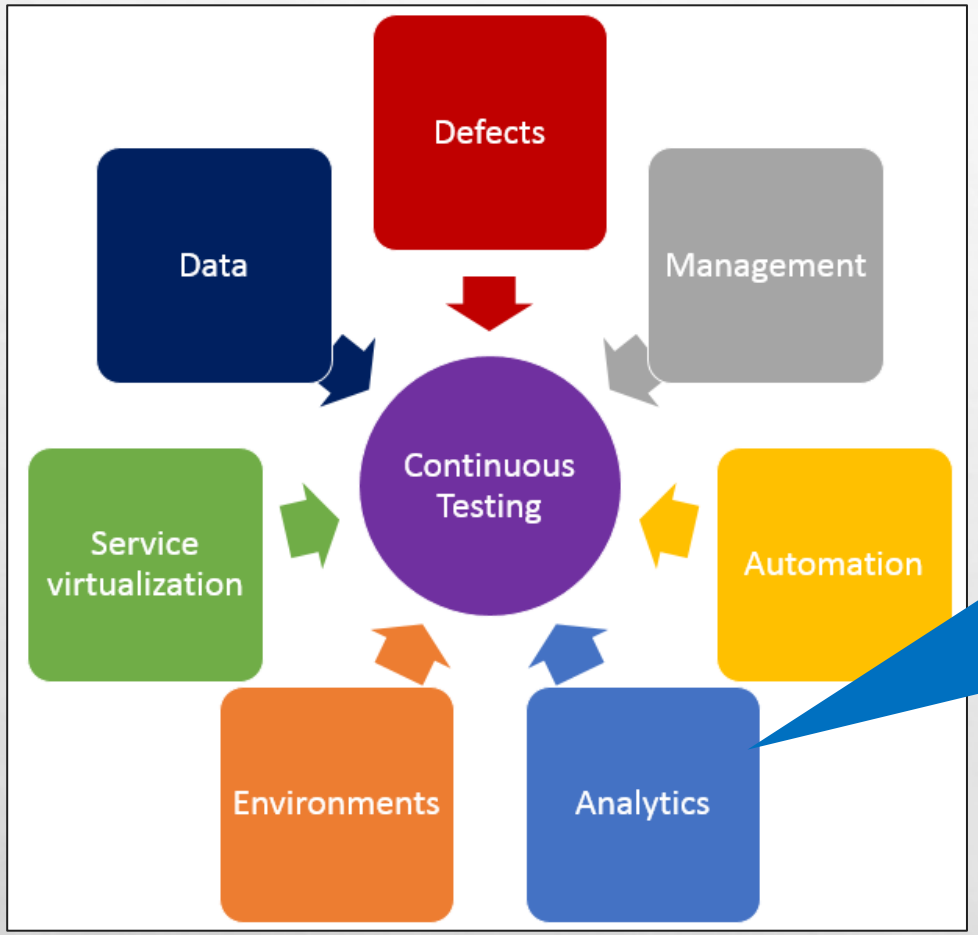
Test Management – Are teams spending time manually crafting status reports and rolling up test execution results? Or do they have a tool that provides that information in real time and allows stakeholders to drill down as needed? How do teams know if they are on schedule with their test effort, behind schedule or even ahead of schedule?

“Continuous Testing: Shift Left and Find the Right Balance”



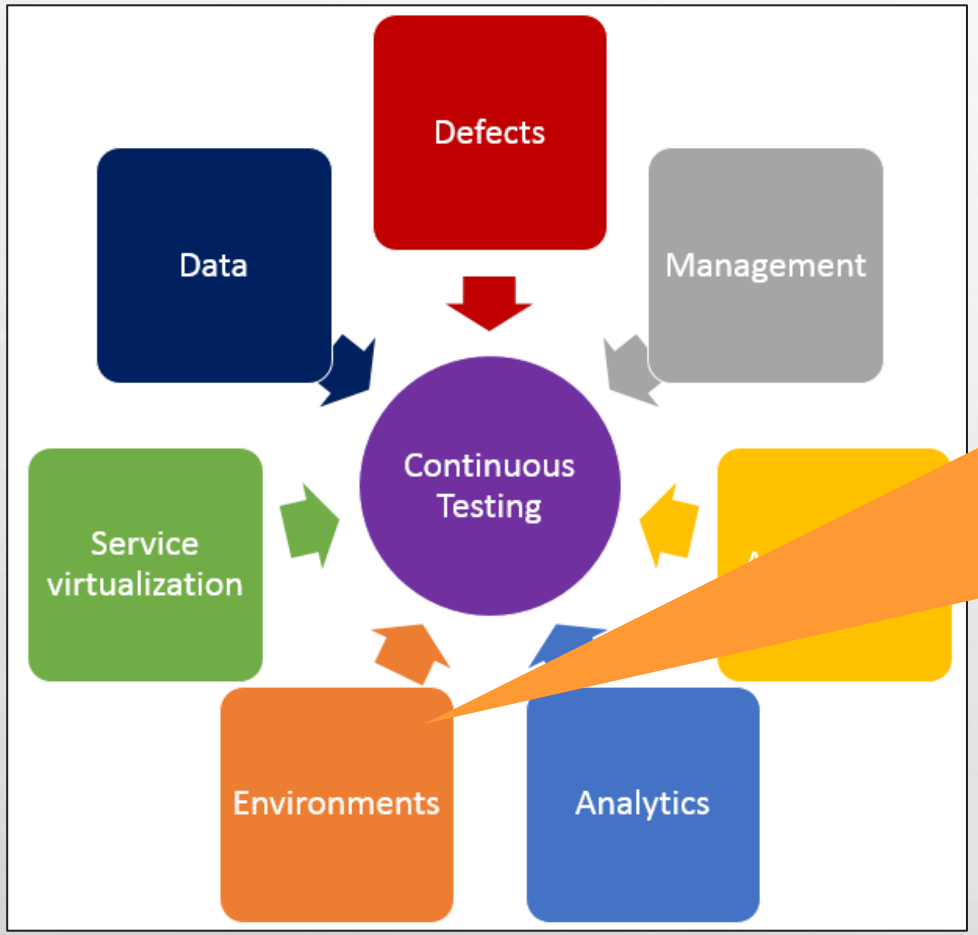
Test Automation – How efficient are teams at re-executing existing tests? Do they run most or even all of them manually? If they’ve automated tests, are they focused only on functional tests at the user interface layer, or are they running functional API-layer tests, performance tests and even security tests? Do they have a robust and maintainable test automation framework?

“Continuous Testing: Shift Left and Find the Right Balance”



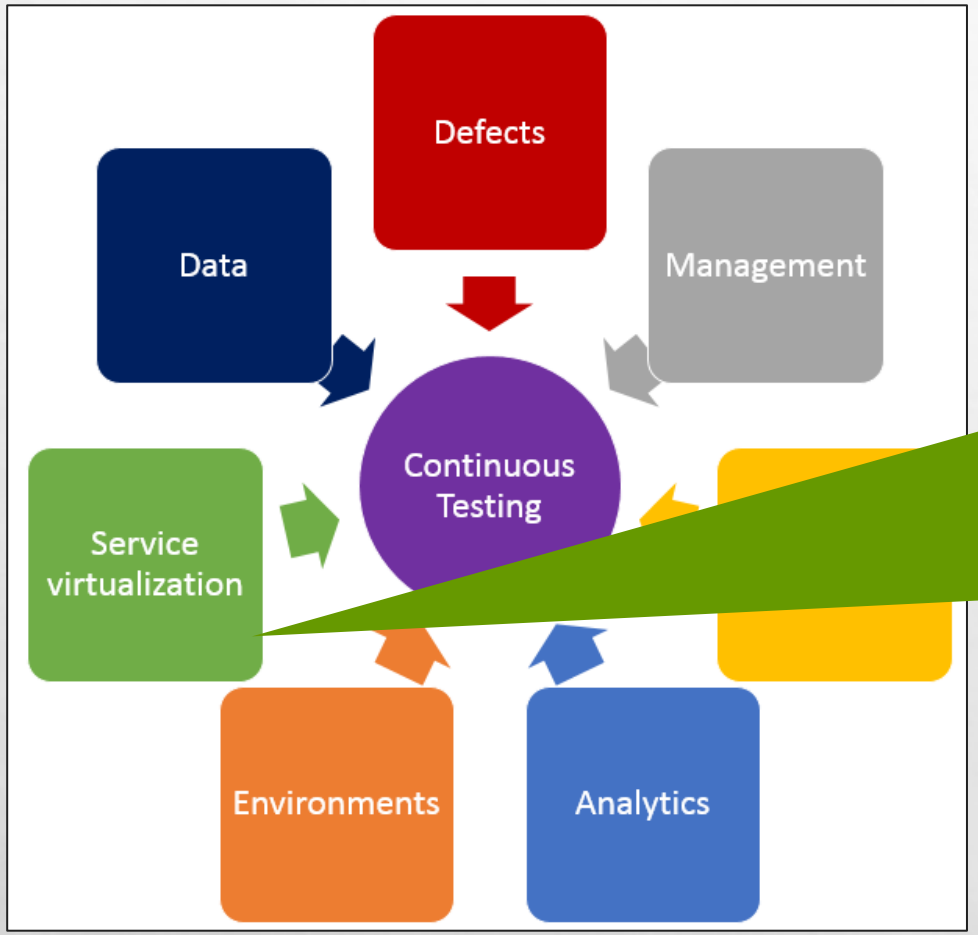
Analytics – How do teams know which tests they should run, when and even why they are running those tests at those times? How good is their test effectiveness, are they running the fewest number of tests that find the largest number of problems? Impact analysis is critical in selecting the right sets of tests to execute whenever they get a new build.

“Continuous Testing: Shift Left and Find the Right Balance”



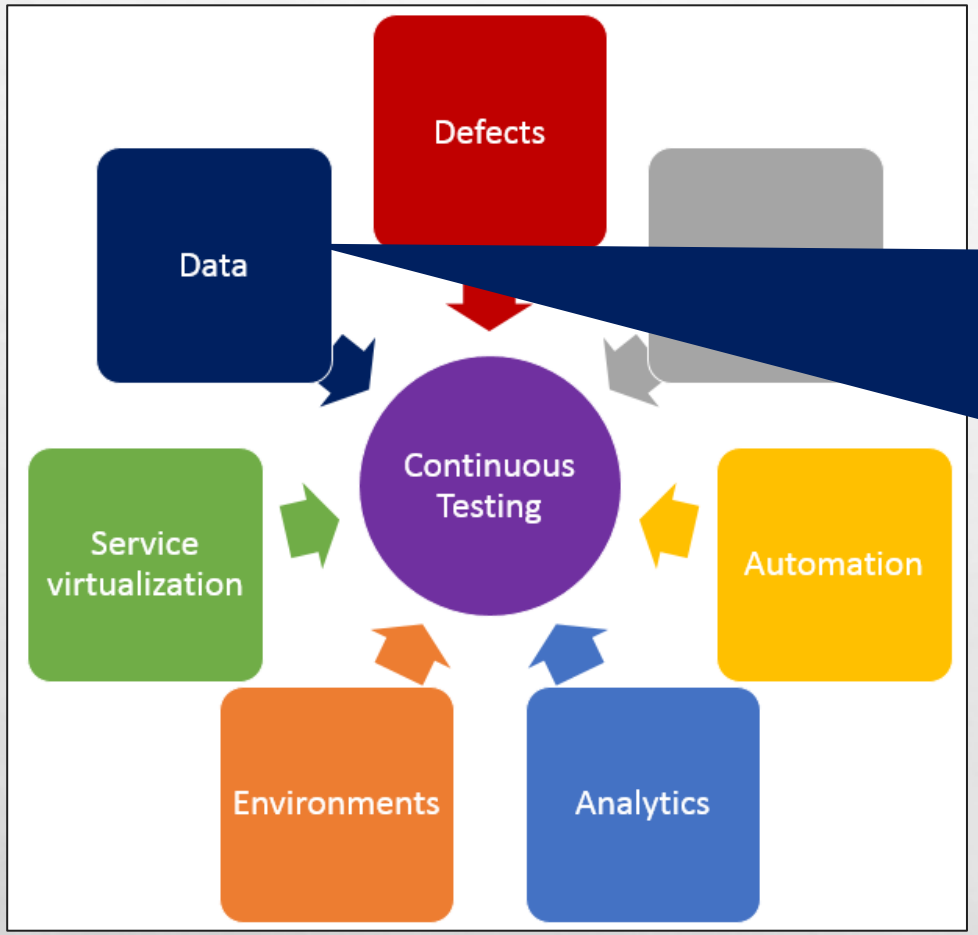
Test Environments – Are teams constantly waiting on test environments to be provisioned and configured properly? Do they run tests and discover after the fact that the test environment wasn’t “right,” so they have to fix the environment and then re-run all the tests again? Do they hear from developers, “It works on my machine!” but it doesn’t work in the test environment?

“Continuous Testing: Shift Left and Find the Right Balance”

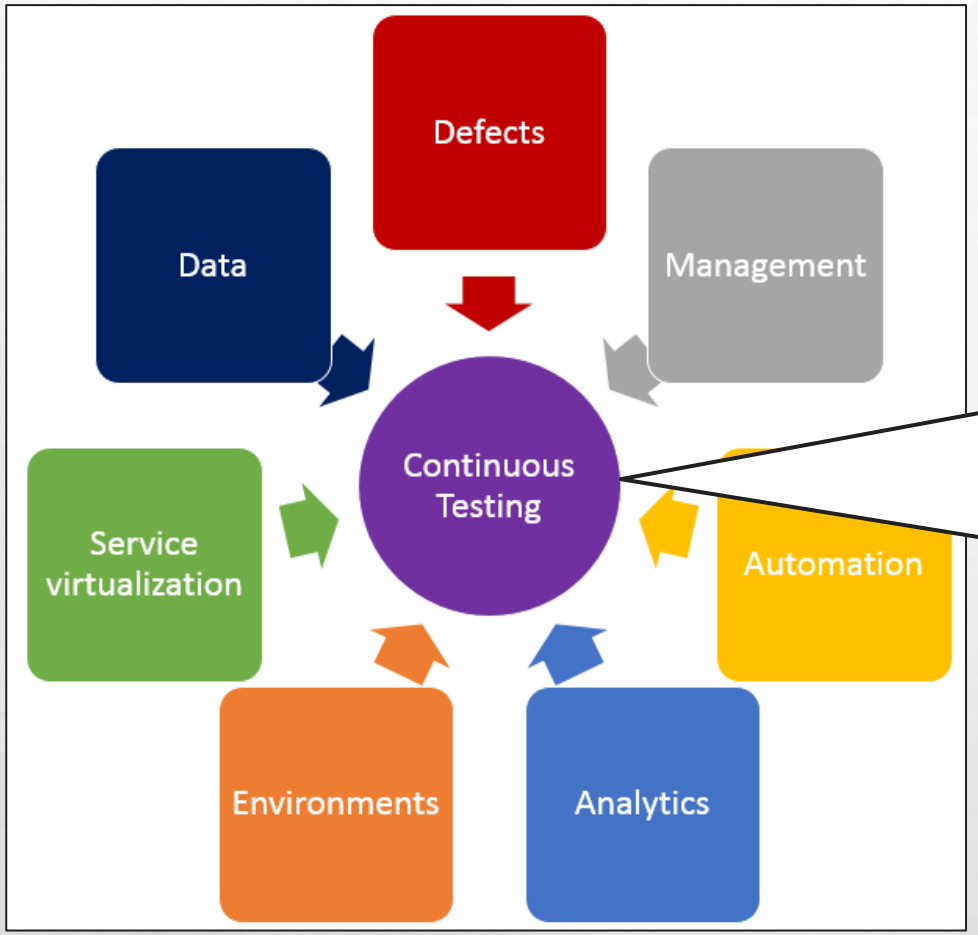


Service Virtualization – Are teams waiting for dependent systems to become available before they can “really” test? Are they using a “big bang” approach to conduct end-to-end system testing, where they throw all the components together and hope they work and interact properly? Can teams test exception and error scenarios before going to production? Are they testing the easiest parts first just because they are available, and delaying the high-risk areas for the end of the testing effort?

“Continuous Testing: Shift Left and Find the Right Balance”



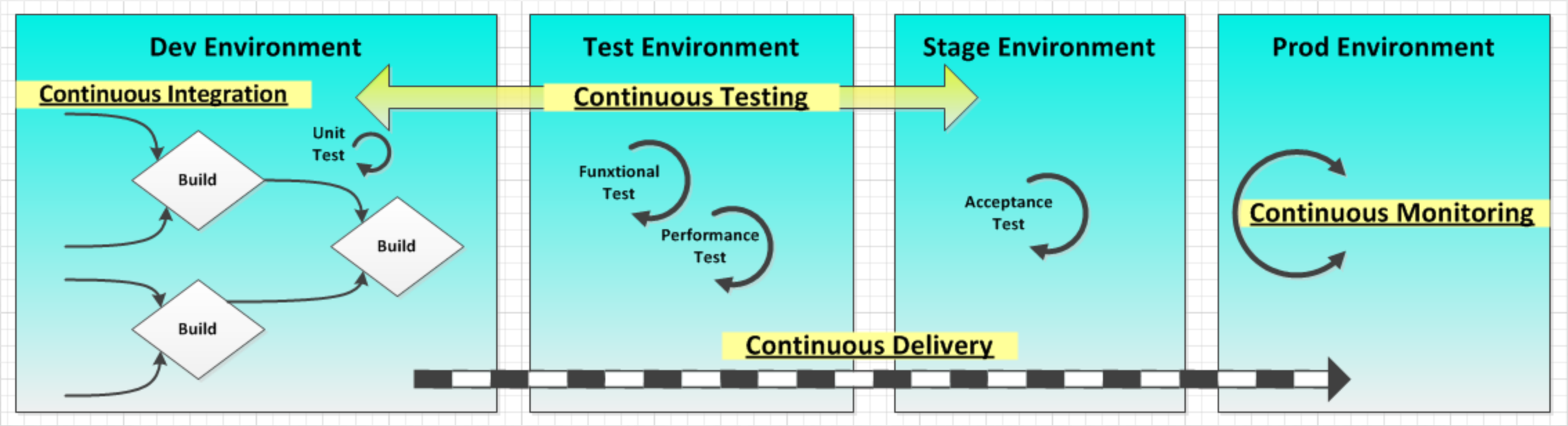
Test Data – Do teams have the needed sets of production-like test data to ensure they are covering the right test scenarios? Are there exception and error scenarios that we can’t execute because they don’t have the right sets of test data?



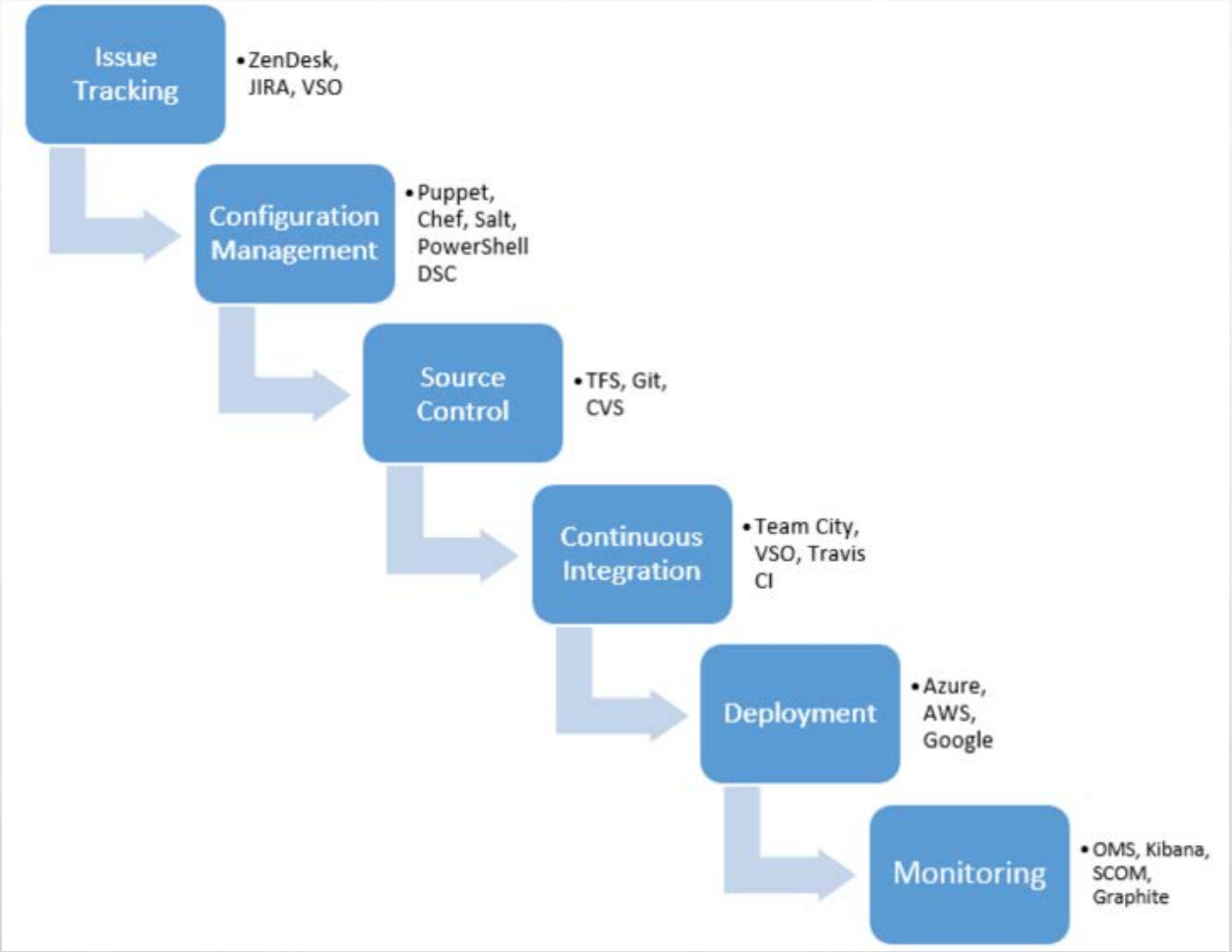
"Continuous Testing: Shift Left and Find the Right Balance"

by Marianne Hollier on DEVOPS.com

“Understanding DevOps – Part 4: Continuous Testing and Continuous Monitoring” by Sanjeev Sharma on sdarchitect.wordpress.com



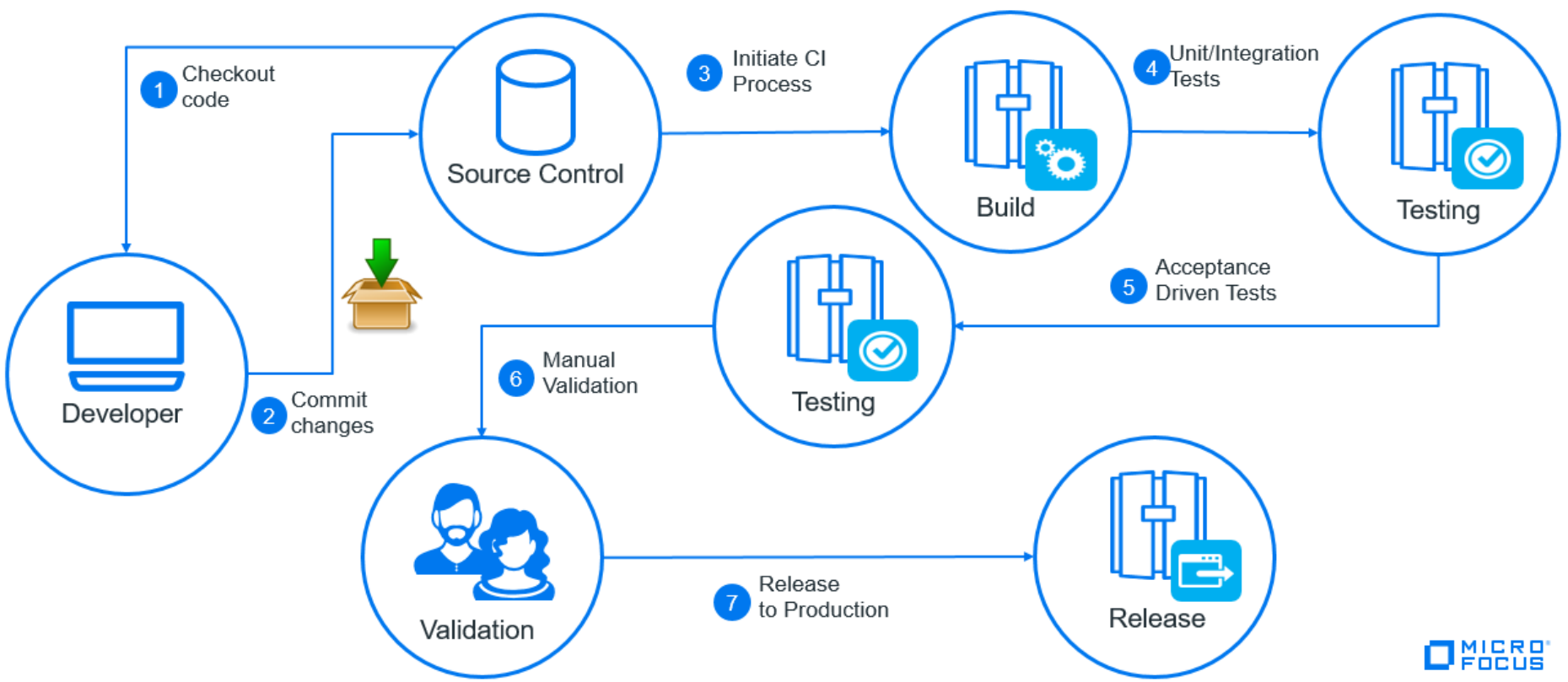
SOME OPEN-SOURCE CATEGORIES/TOOLS IN A TOOLCHAIN



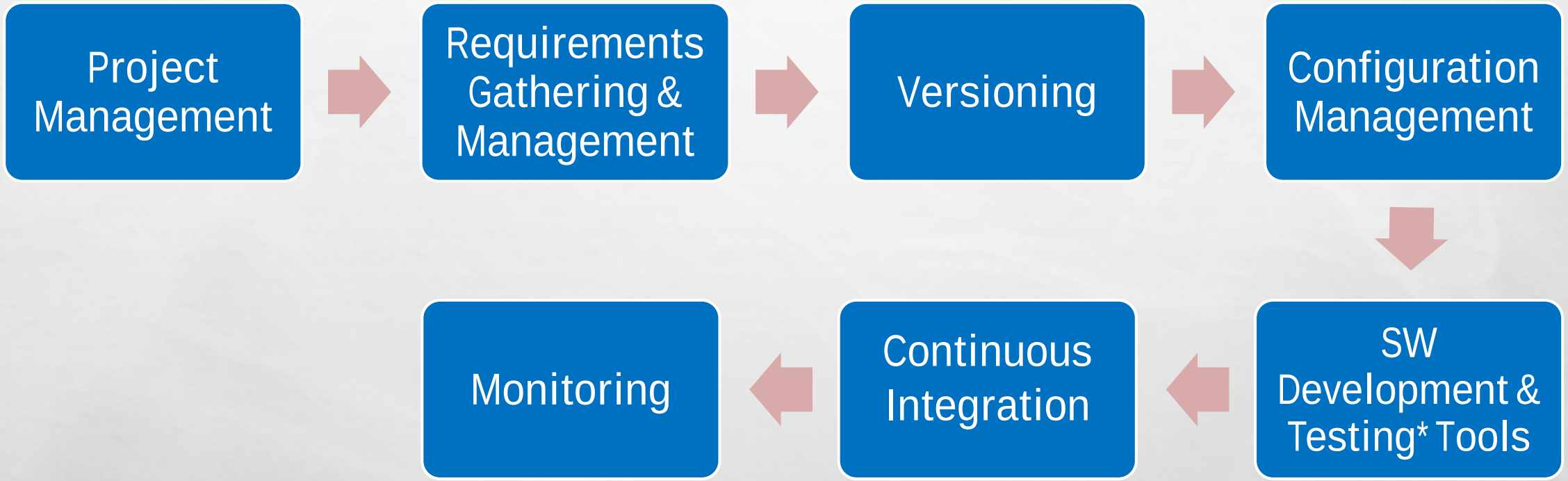
A VENDORS VIEW OF CD WITH CT

Continuous delivery (CD): a SW engineering approach in which teams produce software in short cycles…

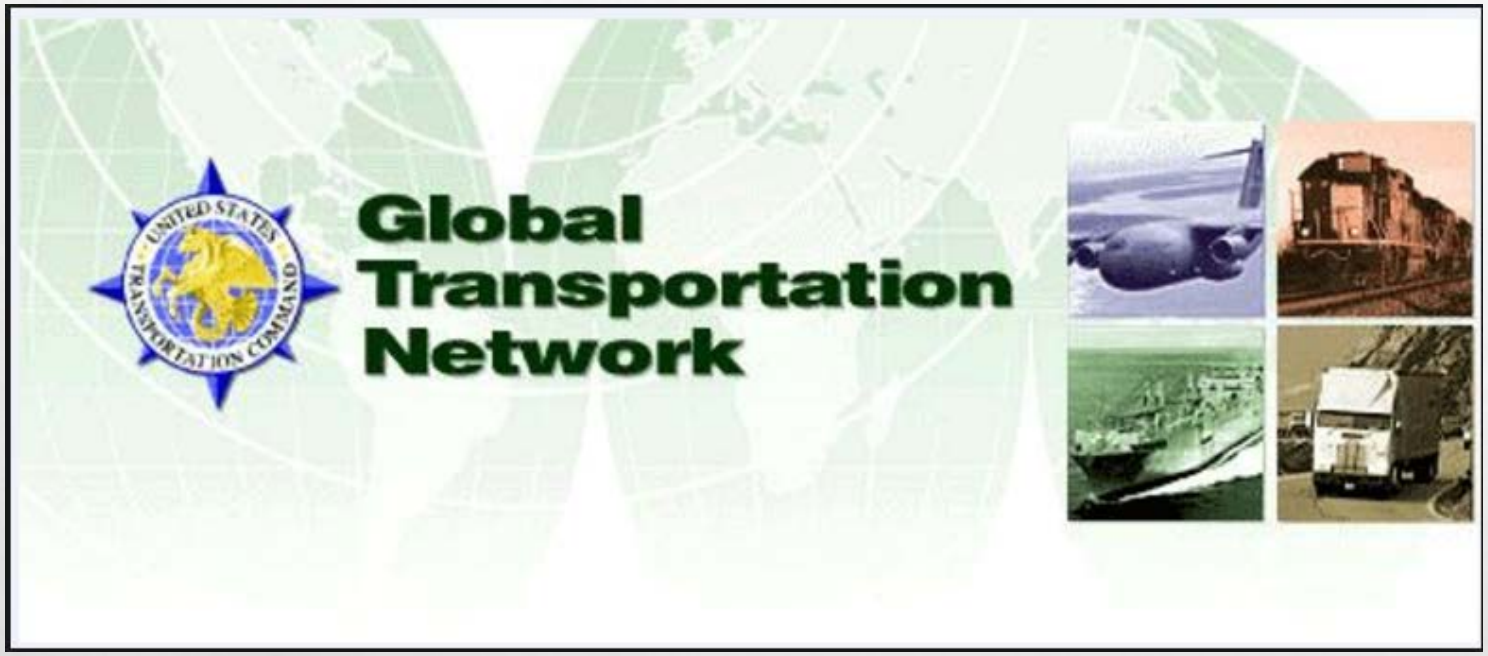
- o Continuous Delivery is not Continuous Deployment
- o Relies on 3 foundations:
  - Configuration management,
  - Continuous integration,
  - ***Continuous Testing***

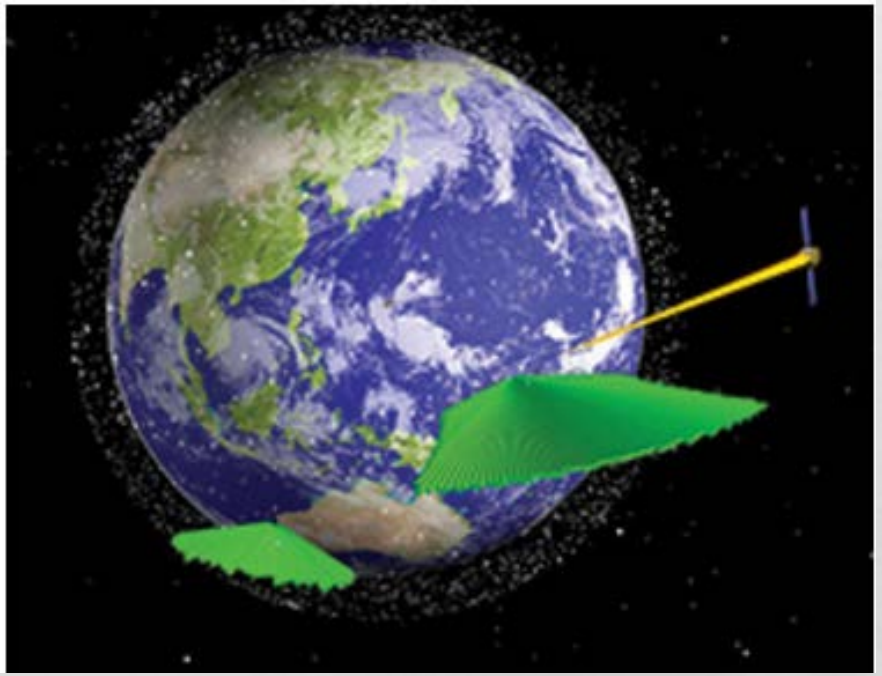


MY SET OF TOOL CATEGORIES IN A TOOLCHAIN



SUCCESS WITH CT IN WATERFALL AND AGILE



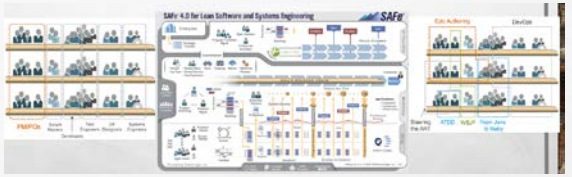


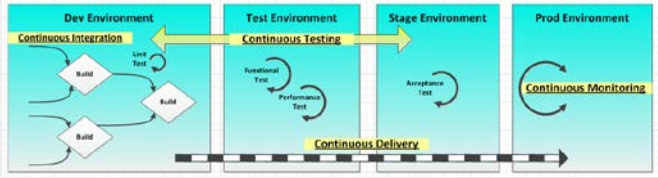
10 COMPANIES KILLING IT AT DEVOPS FROM TECHBEACON.COM

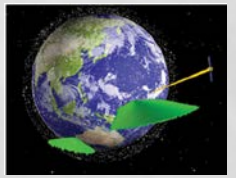
- 1. AMAZON
- 2. NETFLIX
- 3. TARGET
- 4. WALMART
- 5. NORDSTROM
- 6. FACEBOOK
- 7. ETSY
- 8. ADOBE
- 9. SONY PICTURES ENTERTAINMENT
- 10. FIDELITY WORLDWIDE INVESTMENT

WRAP-UP

- CT can be used in any development methodology and provide value
- A robust toolchain is crucial to successful CD and CT – Should be in Proposal/TMRR phase
- Good testing practices are still important and maybe more so in CT as in any other development methodology
- DevOps demands CT to achieve desired outcomes of faster delivery with high quality







FINAL COMMENT:

THE WARFIGHTER DESERVES QUICKER DELIVERIES WITH HIGHER QUALITY!