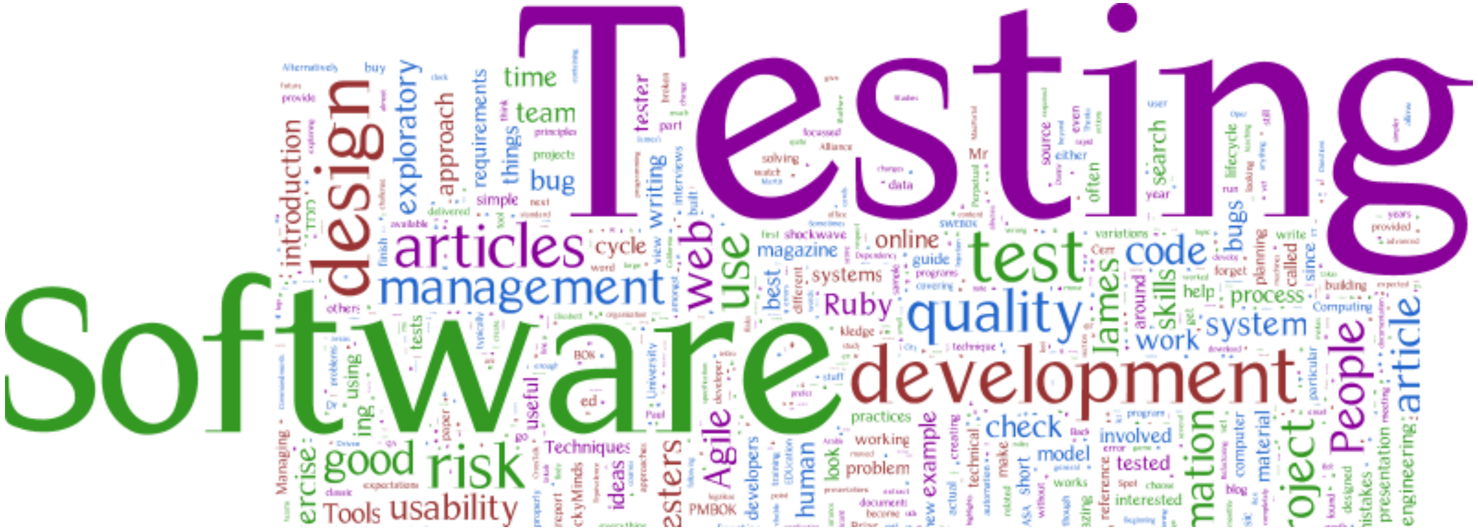


Software Testing for Continuous Delivery

Seminar 2: Intro Cont'd & Developer Testing

Dr. Byron J. Williams
August 23, 2019

UF Herbert Wertheim College of Engineering
Department of Computer & Information Science & Engineering
UNIVERSITY *of* FLORIDA

FICS Research

“It is the one who does the work that does the learning”

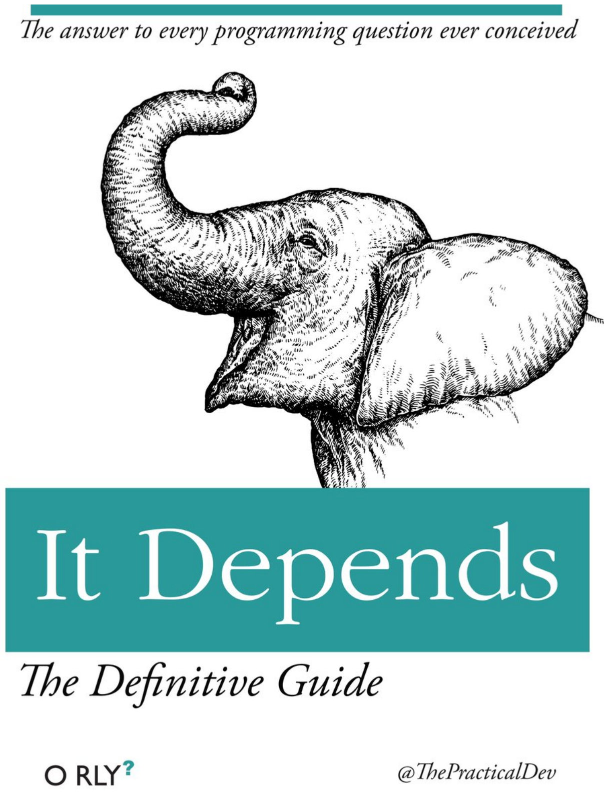
–*Terry Doyle*

UF Herbert Wertheim College of Engineering
Department of Computer & Information Science & Engineering
UNIVERSITY *of* FLORIDA

Points to Remember

NO

- Different vocabularies
- "It depends"
    - A "science of the artificial"
- Author's panacea of anecdotal evidence or narrow research
- Seemingly contradictory claims - ISQTB / Agile / Others
- Peer review - provides confidence



RECAP

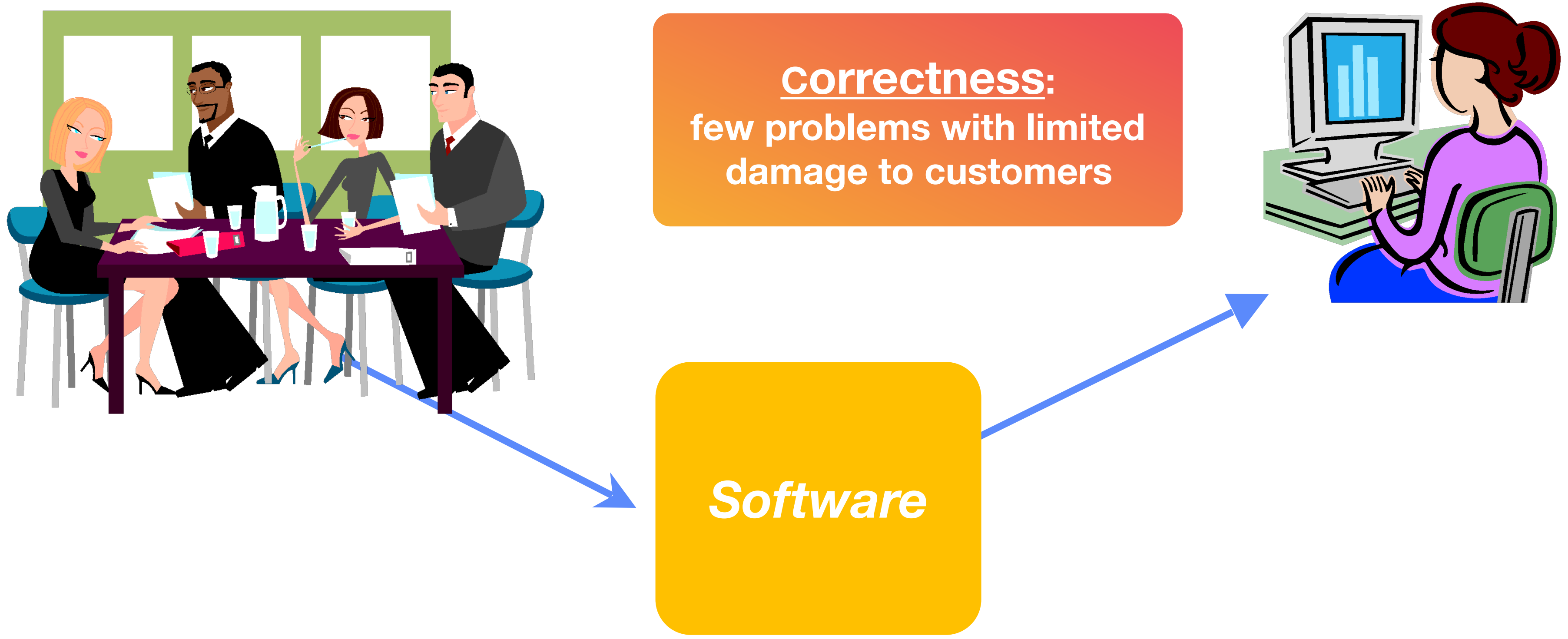
- What is your definition of high **“quality”**?
- Webster’s Dictionary defines “quality” as: ***a degree of excellence***

Software Quality:
The totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs.

- *Deliver software system that …*
    - does what it is supposed to do
    - does the things in a desired way
    - show/demonstrate/prove the above two points

What is software quality?

RECAP



Quality Assurance

To ensure that few, if any, defects remain in the software when it is delivered to its customers or released to the market

RECAP

The Agile Manifesto–a statement of values

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

Source: www.agilemanifesto.org

Herbert Wertheim College of Engineering
UNIVERSITY *of* FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

DEPARTMENT OF COMPUTER & INFORMATION SCIENCE & ENGINEERING
FLORIDA INSTITUTE FOR CYBERSECURITY RESEARCH (FICS)
Dr. Byron J. Williams

Software Testing for Continuous Delivery

CIS 4930 - Computer & Information Science & Engineering

Testing (operational) Definition

- Defined: the **execution of software and the observation of its behavior…** as an outcome demonstrated using controlled experiments
- ***Purpose***:
  - To demonstrate quality / proper behavior
  - To detect problems that need to be fixed
- General Process Steps:
  1. Test Planning & Prep
  2. Test Execution
  3. Analysis & Follow-up

Another Definition:

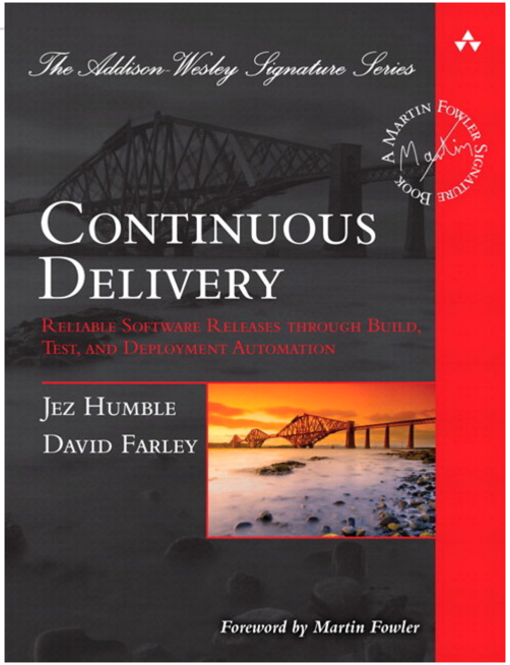
"Testing is the process of evaluating a product by learning about it through exploration and experimentation, which includes to some degree: questioning, study, modeling, observation, inference, etc."

(Bach 2013).

RECAP

Continuous Delivery

- The ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—**into production**, or into the hands of users, **safely** and **quickly** in a **sustainable** way.
- ensuring our code is **always** in a deployable state, even in the face of teams of **thousands** of developers making changes on a daily basis

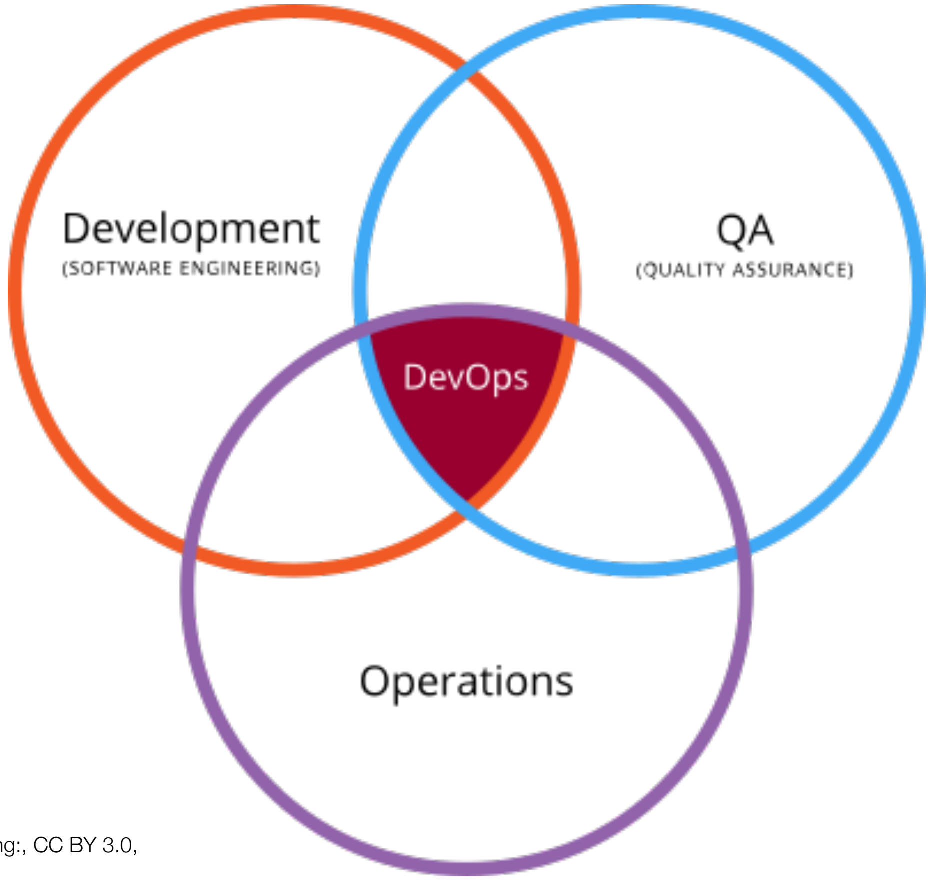

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

Source: https://continuousdelivery.com

RECAP

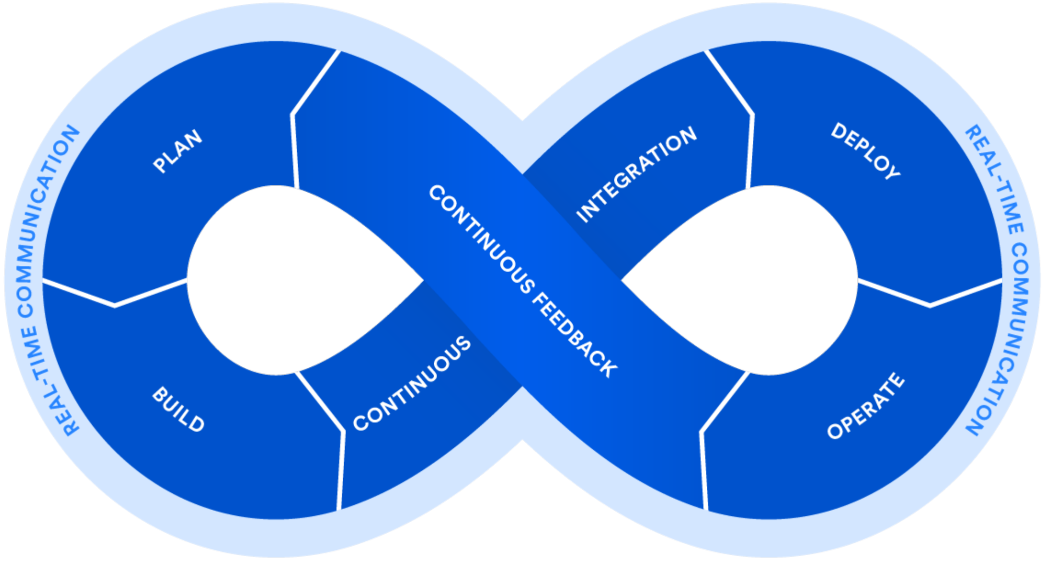
Software Development Operations (DevOps)

Venn diagram

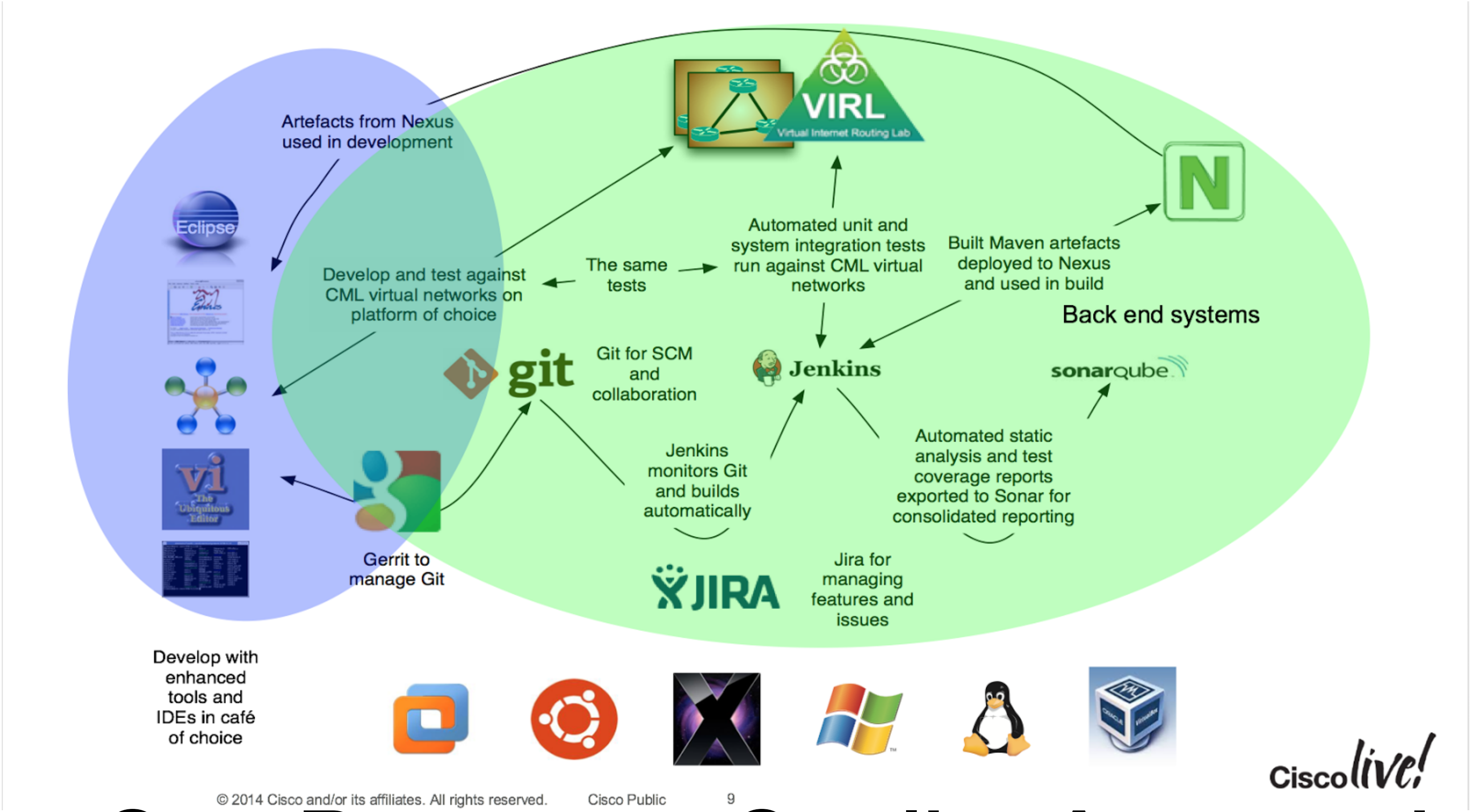


DevOps

- Set of practices that **automates** the processes between **software development** and **IT teams**, in order that they can **build**, **test**, and **release** software **faster** and more **reliably**
- Establishing a **culture of collaboration** between teams that historically functioned in relative silos
- Increased trust, faster software releases, ability to solve critical issues quickly, and better manage unplanned work

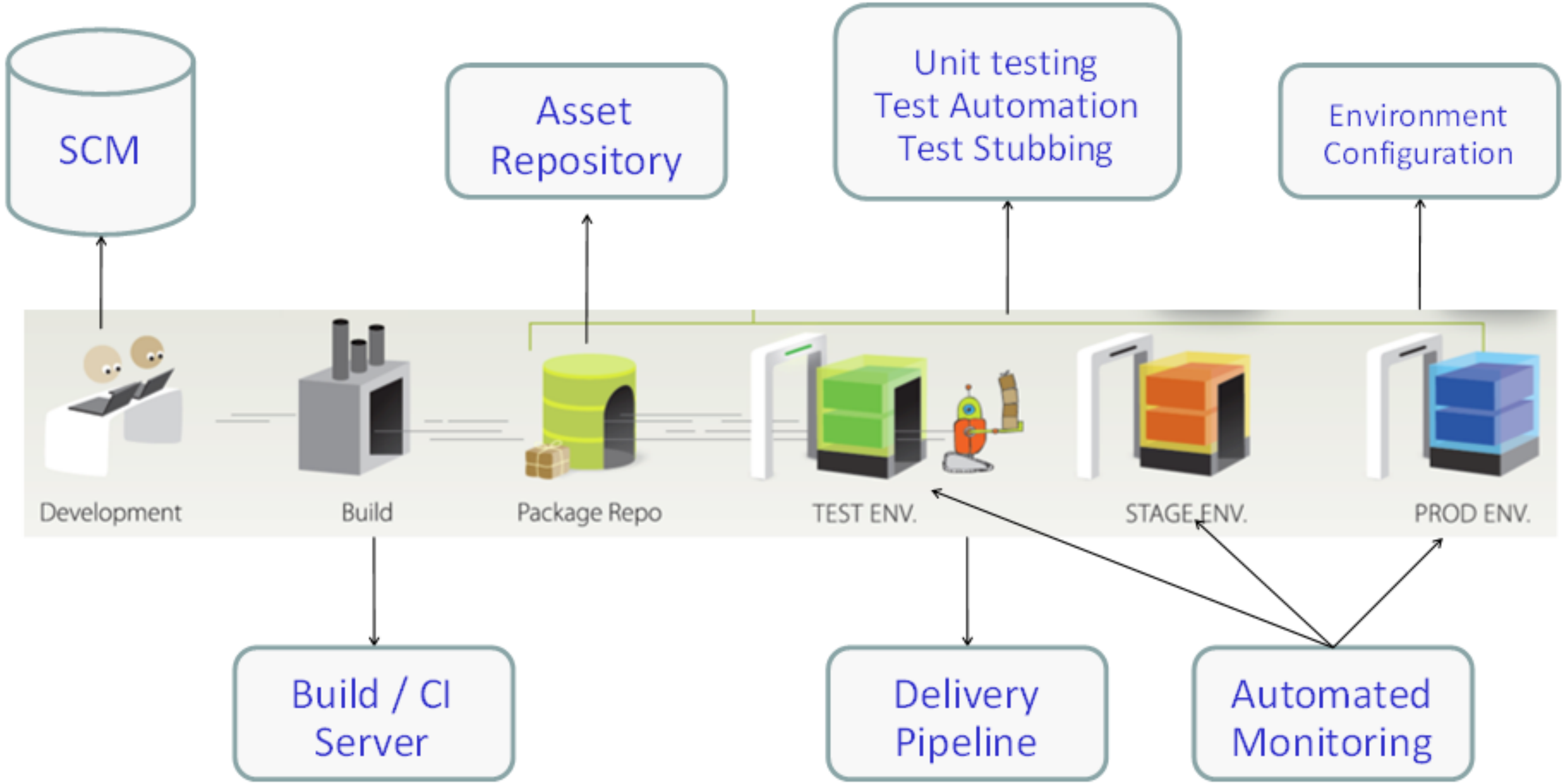


https://www.atlassian.com/devops


Artefacts from Nexus used in development
Eclipse
vi The Ubiquitous Editor
Develop and test against CML virtual networks on platform of choice
The same tests
VIRL
Virtual Internet Routing Lab
Automated unit and system integration tests run against CML virtual networks
Built Maven artefacts deployed to Nexus and used in build
N
Back end systems
git
Git for SCM and collaboration
Jenkins
sonarqube
Gerrit to manage Git
Jenkins monitors Git and builds automatically
Automated static analysis and test coverage reports exported to Sonar for consolidated reporting
JIRA
Jira for managing features and issues
Develop with enhanced tools and IDEs in café of choice
Cisco live!
© 2014 Cisco and/or its affiliates. All rights reserved.
Cisco Public
9

DevOps Process - Quality Automation

Build, Deploy, Provision, Release Toolchain

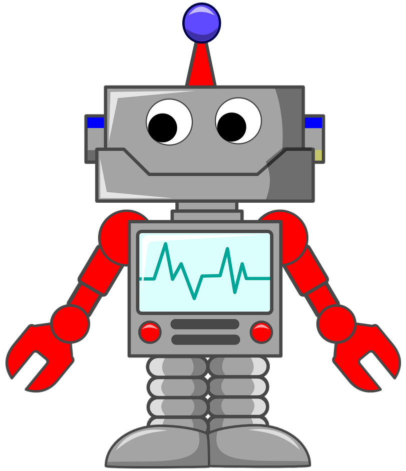


Sample DevOps Deployment Pipeline

https://www.ibm.com/developerworks/community/blogs/c914709e-8097-4537-92ef-8982fc416138/entry/devops_best_practice_-_establishing_a_%E2%80%9Csingle_source_of_truth%E2%80%9D?lang=en

Test Automation

- The use of software to **control the execution of tests**, the **comparison** of actual **outcomes** to predicted outcomes, the setting up of test **preconditions**, and other test control and test **reporting** functions
- Steps:
    1. Control execution
    2. Compare actual vs. predicted outcomes
    3. Setting preconditions
    4. Other reporting & control functions



Test Automation

- **Test Case**
  - The test case values, prefix values, postfix values, and expected results necessary for a complete execution and evaluation of the software under test
  - Test Set
    - A set of test cases / test suite
- **Executable Test Script**
  - A test case that is prepared in a form to be executed automatically on the test software and produce a report
- **Test Automation Framework**
  - A set of assumptions, concepts, and tools that support test automation (e.g., JUnit, xUnit, go test, many others)

Continuous Integration (CI)

- A software engineering process where isolated **code changes are immediately tested and reported** on as they are added to a larger codebase
- Developers incorporate their progress and code changes to the codebase daily (or more frequently)
- The goal is to **provide rapid feedback** to identify and **correct defects** as soon as they are introduced
- **Enables automation**
- Requires dedicated tools and automated tests to be written for the system

Continuous Integration

- Agile methods work best when the current version of the software can be **run against all tests at any time**

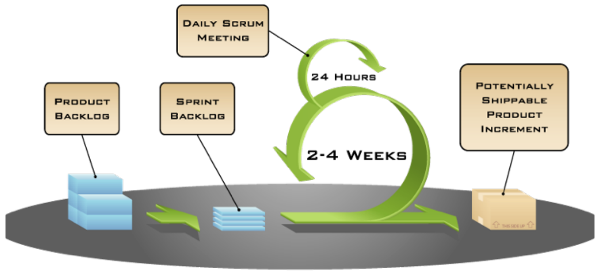
A *continuous integration server* rebuilds the system, returns, and reverifies tests whenever *any* update is checked into the repository

- Mistakes are caught earlier
- Other developers are aware of changes early
- The rebuild and reverify must happen as soon as possible
  - Thus, tests need to execute quickly

A *continuous integration server* doesn't just run tests, it decides if a modified system is still correct

Why Be Agile?

- Agile methods start by recognizing that developers (people) are **bad at predicting** the future (good at approximating)
  - Software engineers are generally **not good at developing requirements** (can blame customers / end-users)
  - We **do not anticipate** many changes
  - Changes we do anticipate are **not needed**
- Requirements (and other "non-executable artifacts") tend to **go out of date** very quickly
  - We seldom take time to update them
  - Many current software projects **change continuously**
- Agile methods expect software to **start small and evolve over time**
  - **Embraces software evolution** instead of fighting it



A Limited View of Correctness

- In traditional methods, we try to define all correct behavior completely, at the beginning
  - What is correctness?
  - Does "correctness" mean anything in large engineering products?
  - People are VERY BAD at completely defining correctness
- In agile methods, **one way** to redefine correctness to be relative to a specific set of tests
  - If the software behaves correctly on the tests, it is "correct"
  - Instead of defining all behaviors, we demonstrate some **behaviors**
  - Mathematicians may be disappointed at the lack of completeness

Correctness: few problems with limited damage to customers **demonstrated by observed (tested) behavior**

Testing / QA / DevOps Terms

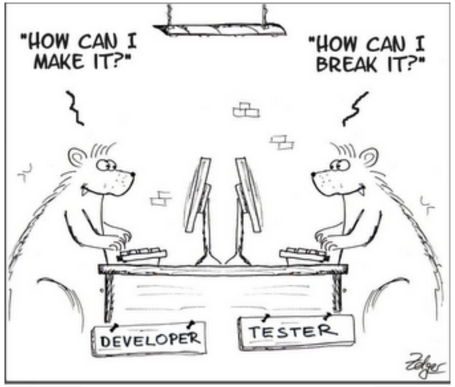
Smoke testing	Defect	Mutation testing
Code review	Load (stability) tests	Performance testing
Software quality	Usability testing	Defect density
Penetration testing	Software inspection	Beta testing
Test runner	User interface testing	System testing
Integration testing	Exploratory testing	Error
Operational profile	Defect containment	Quality Engineering
Unit testing	Partition testing	Fault
Regression testing	Boundary value analysis	Quality Assurance
Fuzz testing	Validation	Failure
Equivalence class	Acceptance testing	Severity levels (defects)
Verification	Gray box testing	Continuous integration
Continuous delivery	Automation	Correctness

Herbert Wertheim College of Engineering
UNIVERSITY *of* FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

DEPARTMENT OF COMPUTER & INFORMATION SCIENCE & ENGINEERING
FLORIDA INSTITUTE FOR CYBERSECURITY RESEARCH (FICS)
Dr. Byron J. Williams

Developer Testing



They are not so much different,
but they have different path for the same goal,
to improve quality!!

Testing Perspectives

"developer" **-** person whose primary responsibility is to write source code - the output of the developers should be working software, not just something that compiles

Developer Testing (developer mind-set)

taking ownership of the quality of the produced code, instead of expecting that someone else will test it

Developer testing is an umbrella term for all test-related activities a developer engages in

To build - **Testing to Support**

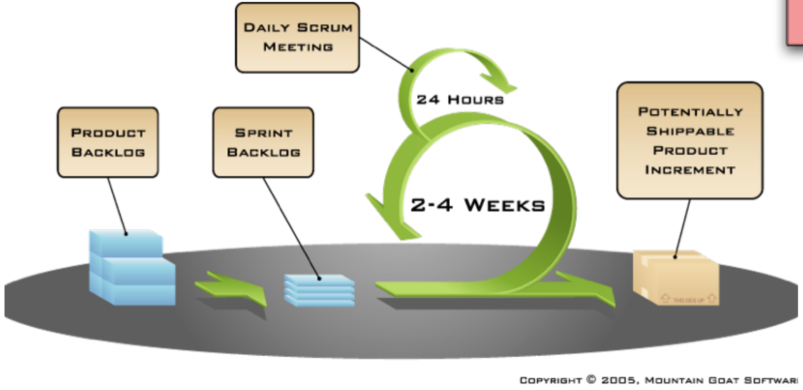
Software Testing (tester mind-set)

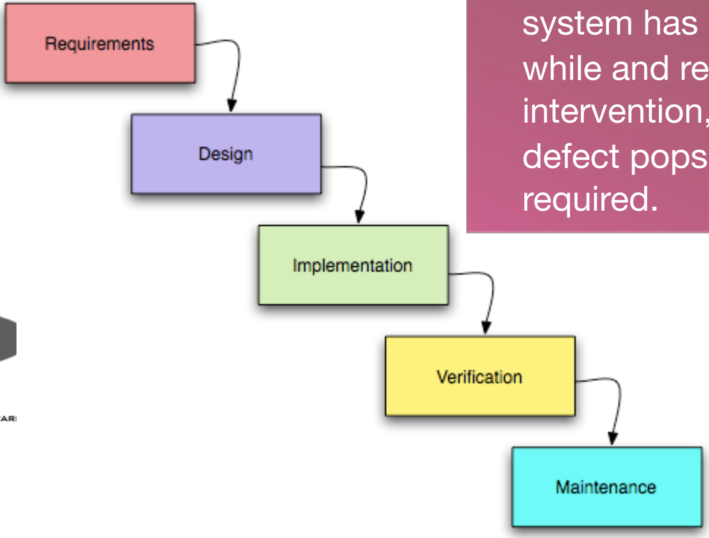
investigating how the product might fail

To break - **Testing to Critique**

Developer Testing

Developer testing as such is quite independent of the development process. Waterfall, ad hoc, agile—regardless of how the software is being developed, applying developer testing practices will result in better software.


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software



Maintenance (2 parts):

Maintenance of a system under development—The system is already running in production while new features are being added to it.

Patching and bug fixing—The system has been stable for quite a while and requires relatively little intervention, but once in a while a defect pops up and a bug fix is required.

Points to Remember

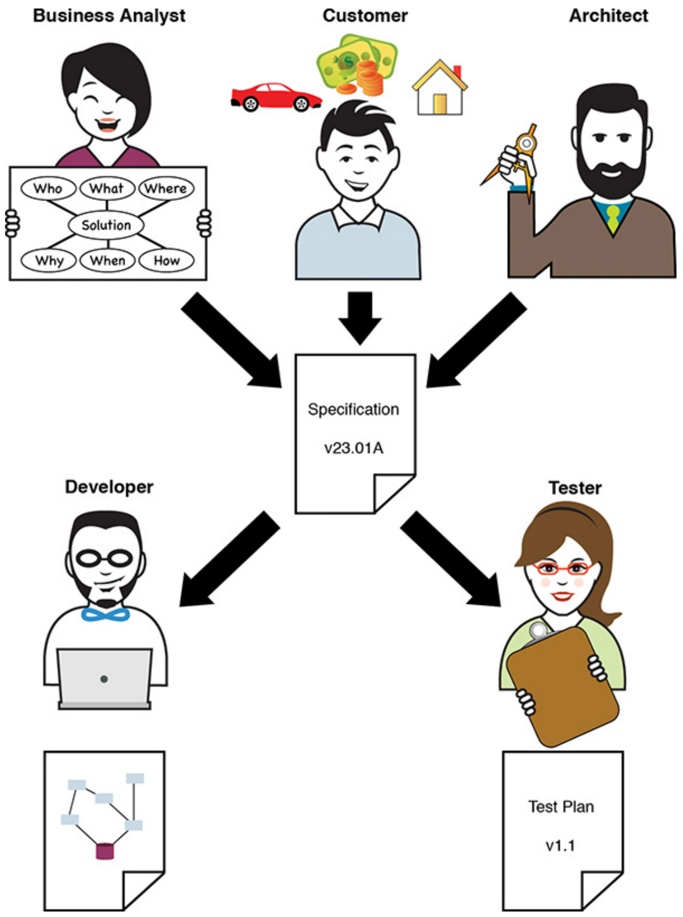
- Quality cannot be tested in, it has to be built in
- Tests and testers are needed because developers suffer from author bias i.e., the inability to see faults in one's own creation
- All written code ***should*** have tests - the opposite: code that turns all attempts to change it into a mixture of one part guessing game and one part nightmare—**legacy code**
  - legacy code is code without tests

Traditional Testing

Thought of as a **verification phase** occurring after a construction phase. First something gets built and then it's verified to make sure that it works tends to decouple testing from development

Because of the clear division of labor, employing traditional testing may create an environment where **developers** and **testers** develop quite an **adversarial view** of each other

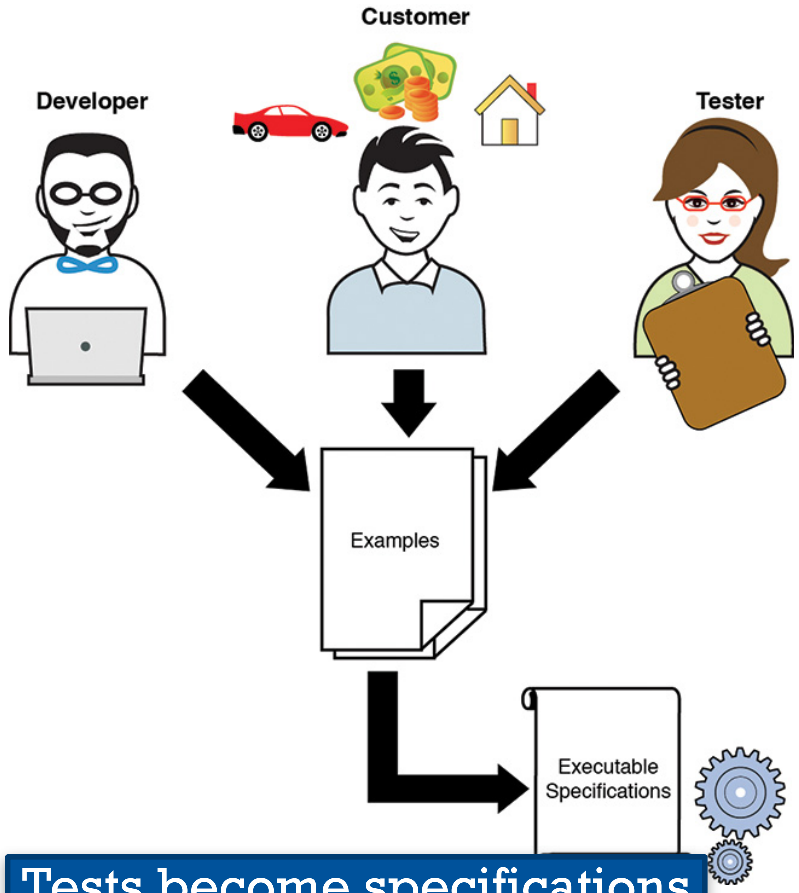
developer's point of view, results in written defect reports or tickets in a bug-tracking tool



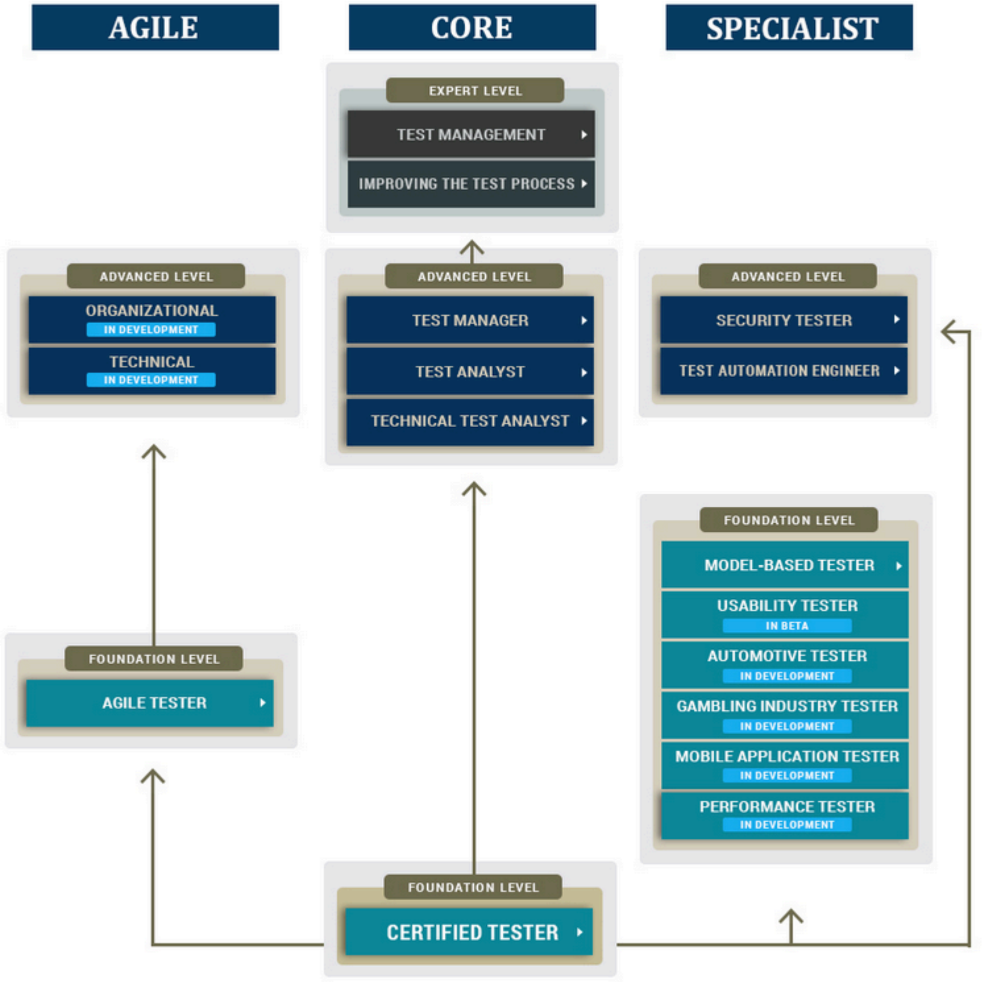
Agile Testing

Enables agile development - the role of the tester is shifted from reactive to proactive - helping the customer or product owner to specify desired functionality, by making sure that testing activities are taken into account during planning and estimation meetings, by educating and assisting the developers in test design and test automation

***Everyone** on an agile team is responsible for turning the functionality requested by the customer into software*



Tests become specifications



ISTQB®
International Software Testing Qualifications Board