DevOps for Defense

Test-Driven Mindset

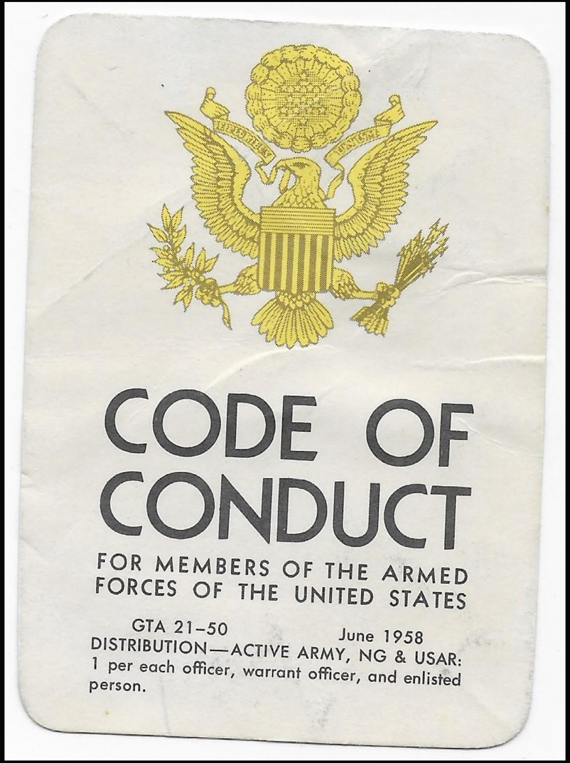
JD Black
June 2021

https://devopsfordefense.org
https://www.meetup.com/DevOps-for-Defense/
https://github.com/jondavid-black/DevOpsForDefense
devopsfordefense@gmail.com
https://twitter.com/devops4defense

Sponsored by: TEKsystems Own change

DevOps for Defense Meetup: Code of Conduct

- ➢ UNCLASSIFIED ONLY!!!!
- ➢ Treat each other with respect and professionalism.
- ➢ Do not talk about private, sensitive, or proprietary work.
- ➢ Do talk about your experiences, needs, desires to improve work in our domain.
- ➢ Do share your thoughts.
- ➢ Do learn from others.
- ➢ If you're attending virtually, do mute yourself while others are speaking!



CODE OF CONDUCT

FOR MEMBERS OF THE ARMED FORCES OF THE UNITED STATES

GTA 21-50 June 1958

DISTRIBUTION—ACTIVE ARMY, NG & USAR:
1 per each officer, warrant officer, and enlisted person.

Hybrid Meetup: Not As Easy As I Thought

Last Month:

- Very Meh Audio Quality
- Couldn't Hear Virtual Participants
- Ad-Hoc Post-Presentation Networking / Lean-Coffee

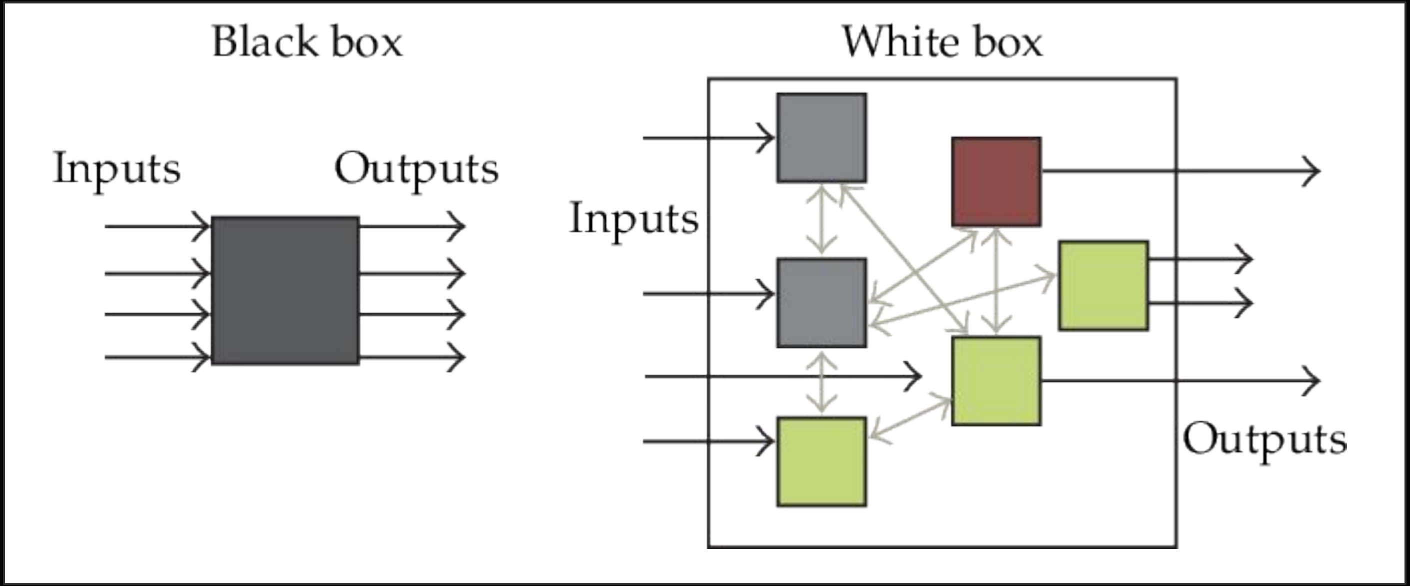
This Month:

- Improved Audio Setup
- Improved Video Setup
- Presenter Leads Online Lean-Coffee

+

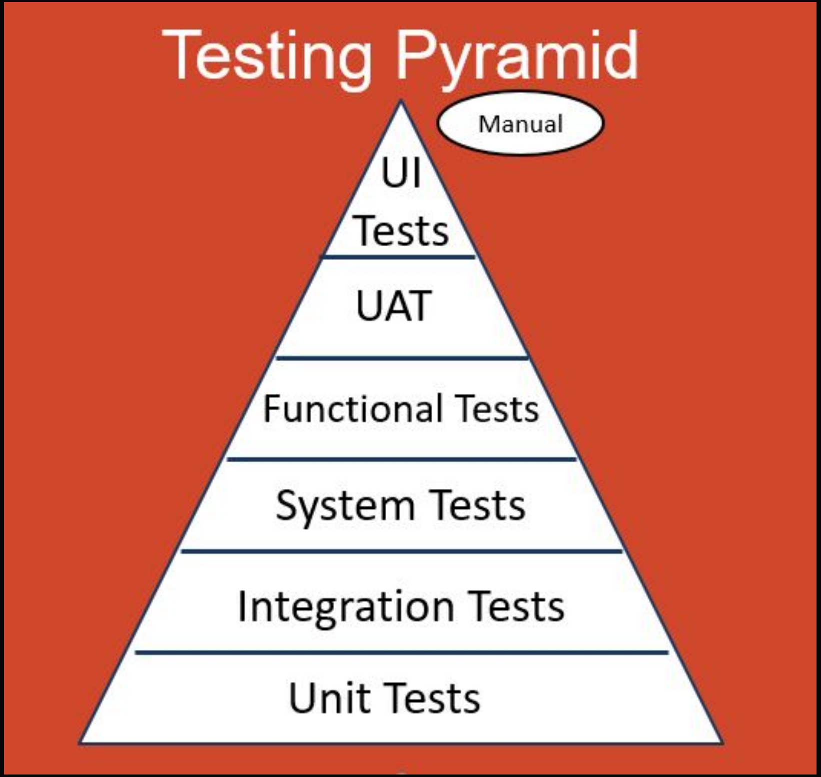
How Do You Test When Failure is Not an Option?

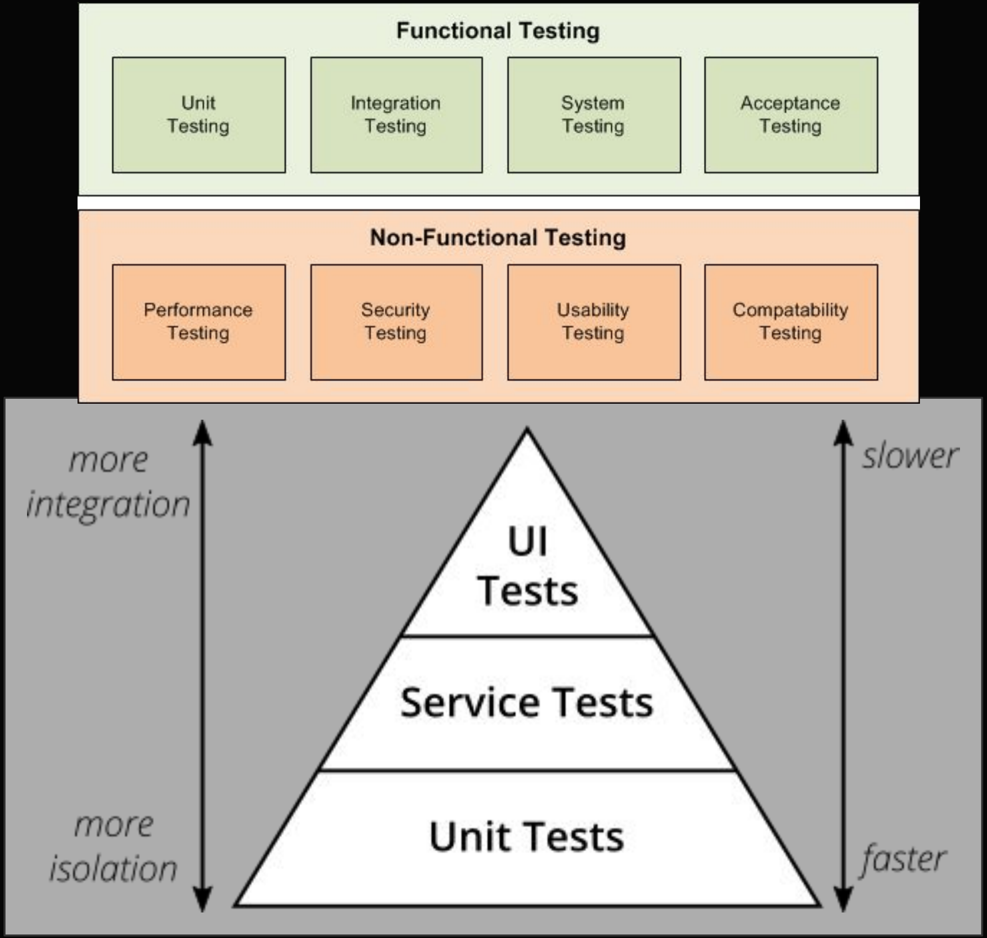
Definitions: White vs Black Box



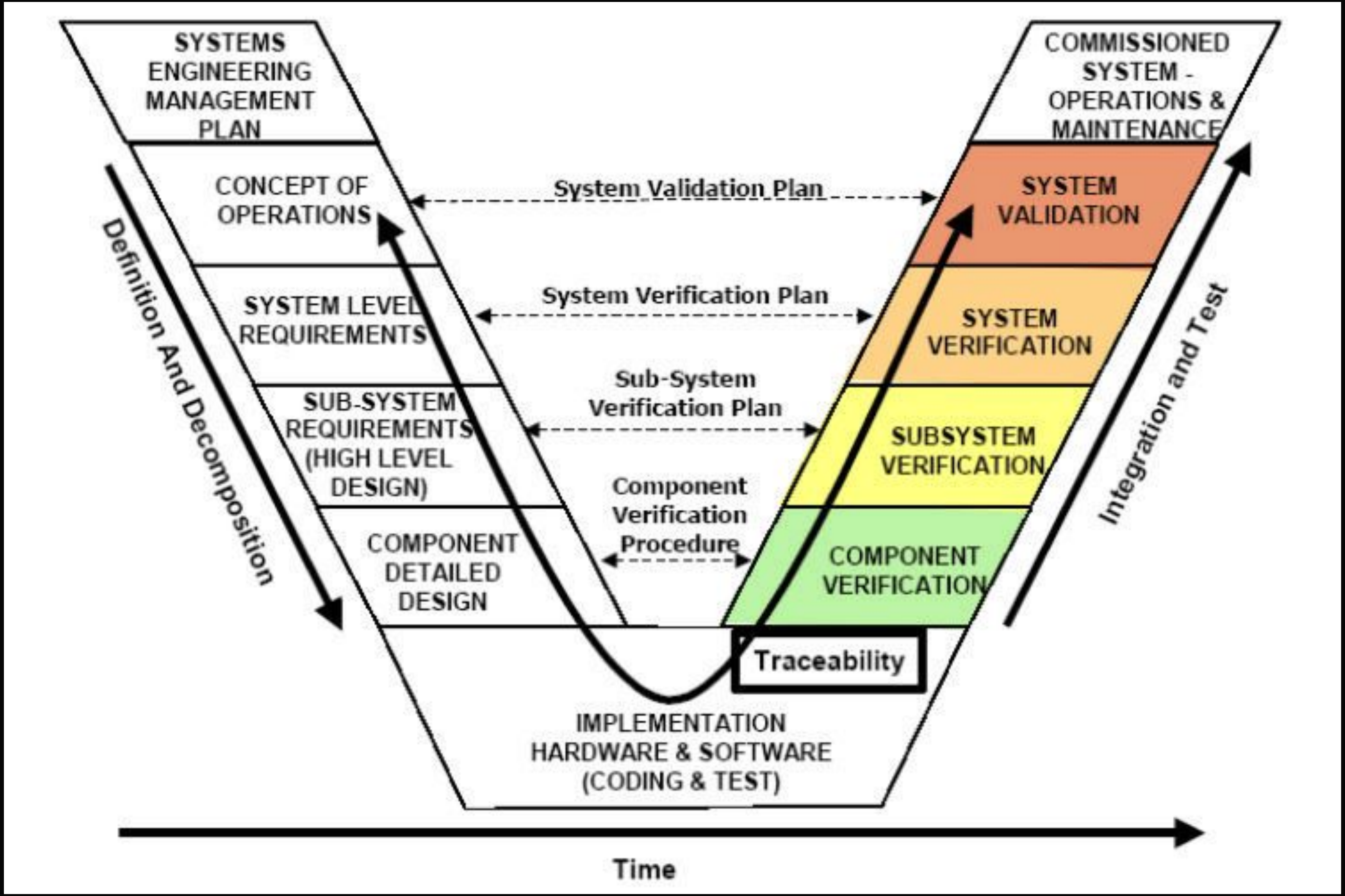
Test as a User	Test as a Developer
Interface Driven	Design Driven
Requirements Coverage	Path / Branch Coverage
Nominal & Off-Nominal	Isolation of Unit Under Test

Definitions: Test Pyramid

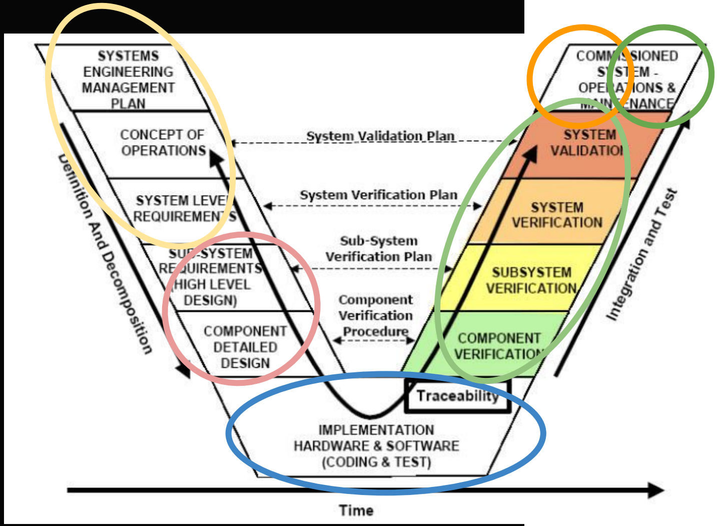


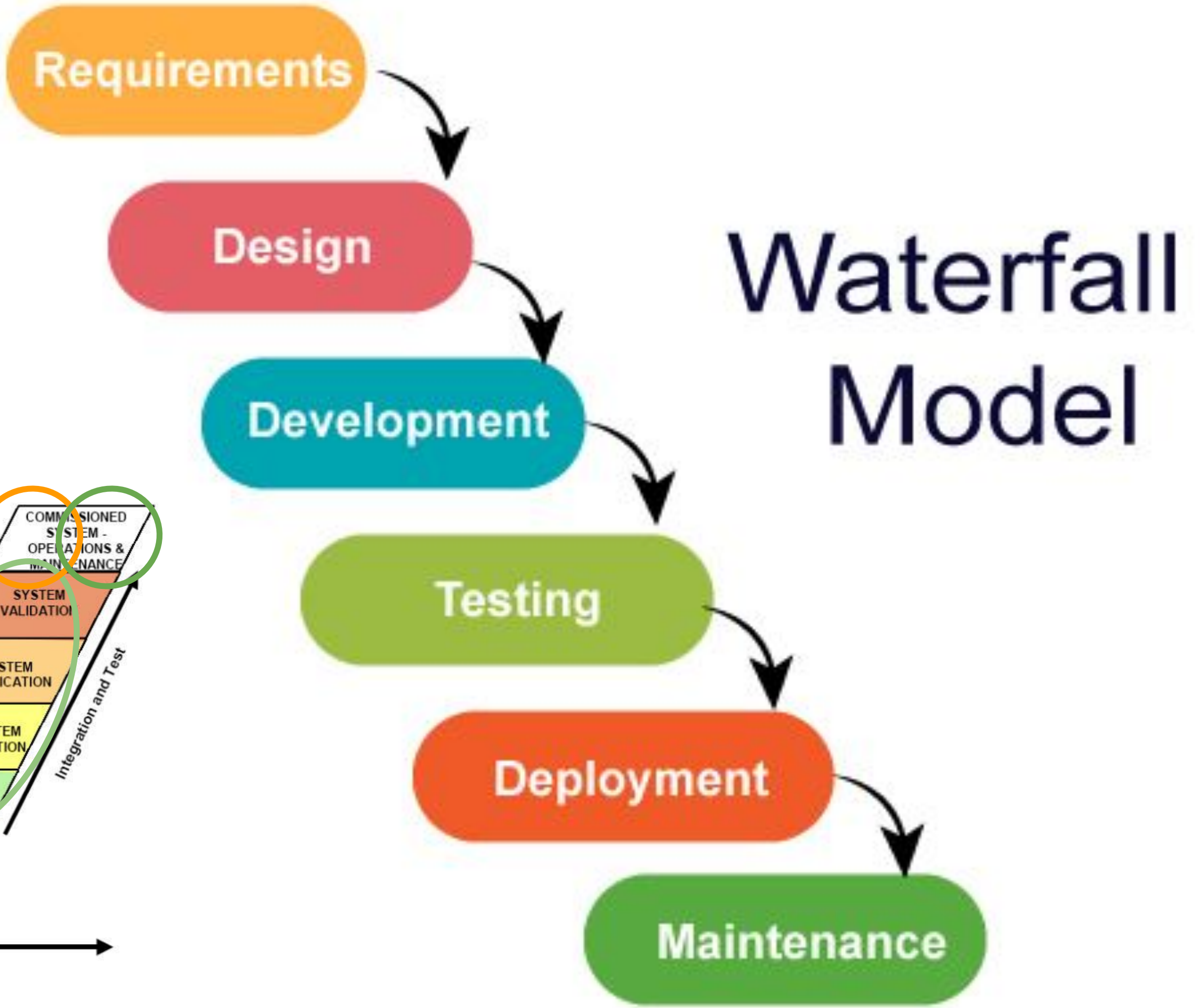


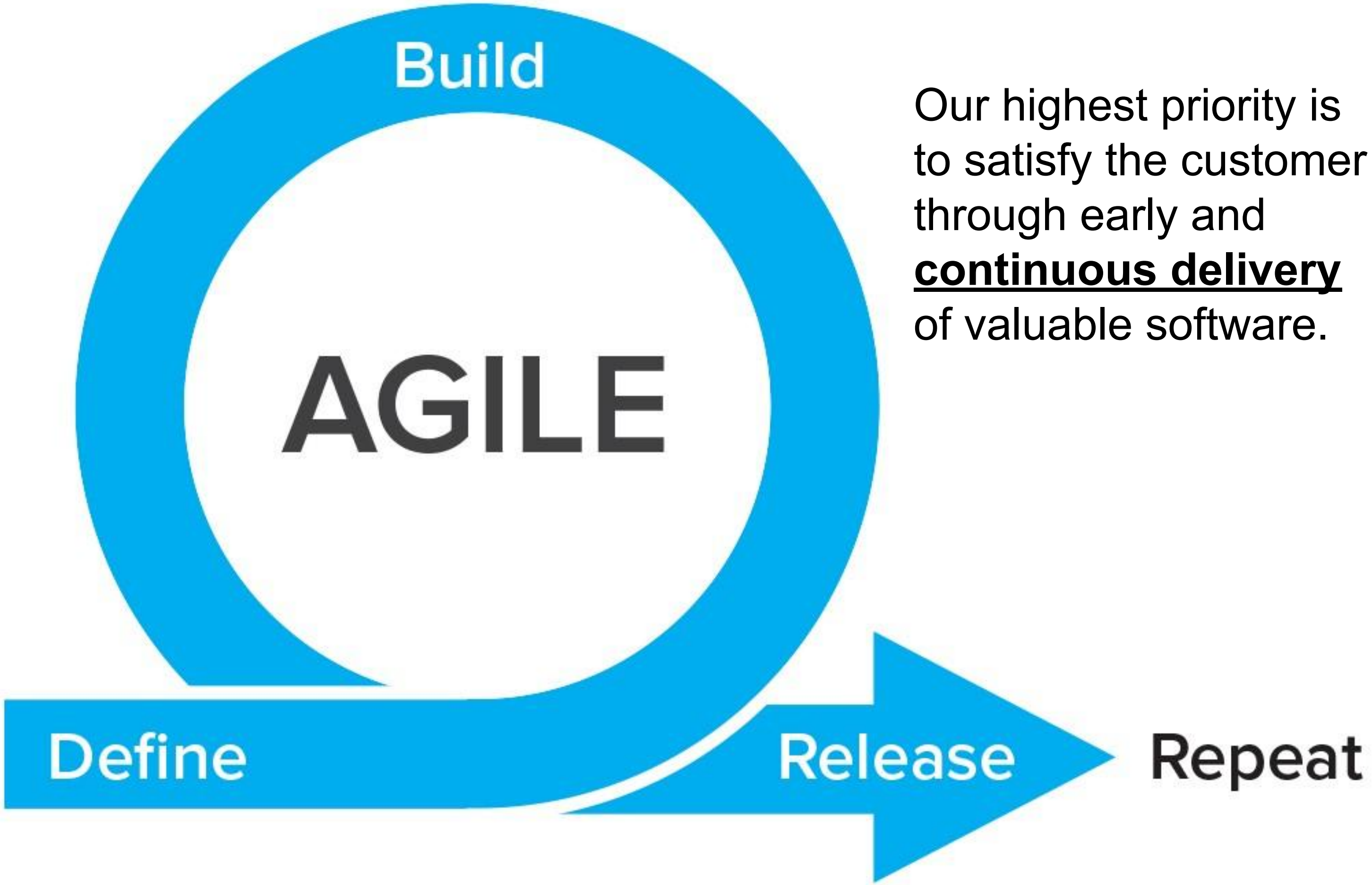
Traditional System Life Cycle


SYSTEMS ENGINEERING MANAGEMENT PLAN
CONCEPT OF OPERATIONS
SYSTEM LEVEL REQUIREMENTS
SUB-SYSTEM REQUIREMENTS (HIGH LEVEL DESIGN)
COMPONENT DETAILED DESIGN
IMPLEMENTATION HARDWARE & SOFTWARE (CODING & TEST)
COMPONENT VERIFICATION
SUBSYSTEM VERIFICATION
SYSTEM VERIFICATION
SYSTEM VALIDATION
COMMISSIONED SYSTEM - OPERATIONS & MAINTENANCE
System Validation Plan
System Verification Plan
Sub-System Verification Plan
Component Verification Procedure
Traceability
Definition And Decomposition
Integration and Test
Time

Traditional “Engineering V” aligns well to Waterfall


SYSTEMS ENGINEERING MANAGEMENT PLAN
CONCEPT OF OPERATIONS
SYSTEM LEVEL REQUIREMENTS
SUB-SYSTEM REQUIREMENTS (HIGH LEVEL DESIGN)
COMPONENT DETAILED DESIGN
IMPLEMENTATION HARDWARE & SOFTWARE (CODING & TEST)
COMPONENT VERIFICATION
SUBSYSTEM VERIFICATION
SYSTEM VERIFICATION
SYSTEM VALIDATION
COMMISSIONED SYSTEM - OPERATIONS & MAINTENANCE
System Validation Plan
System Verification Plan
Sub-System Verification Plan
Component Verification Procedure
Traceability
Definition And Decomposition
Integration and Test
Time

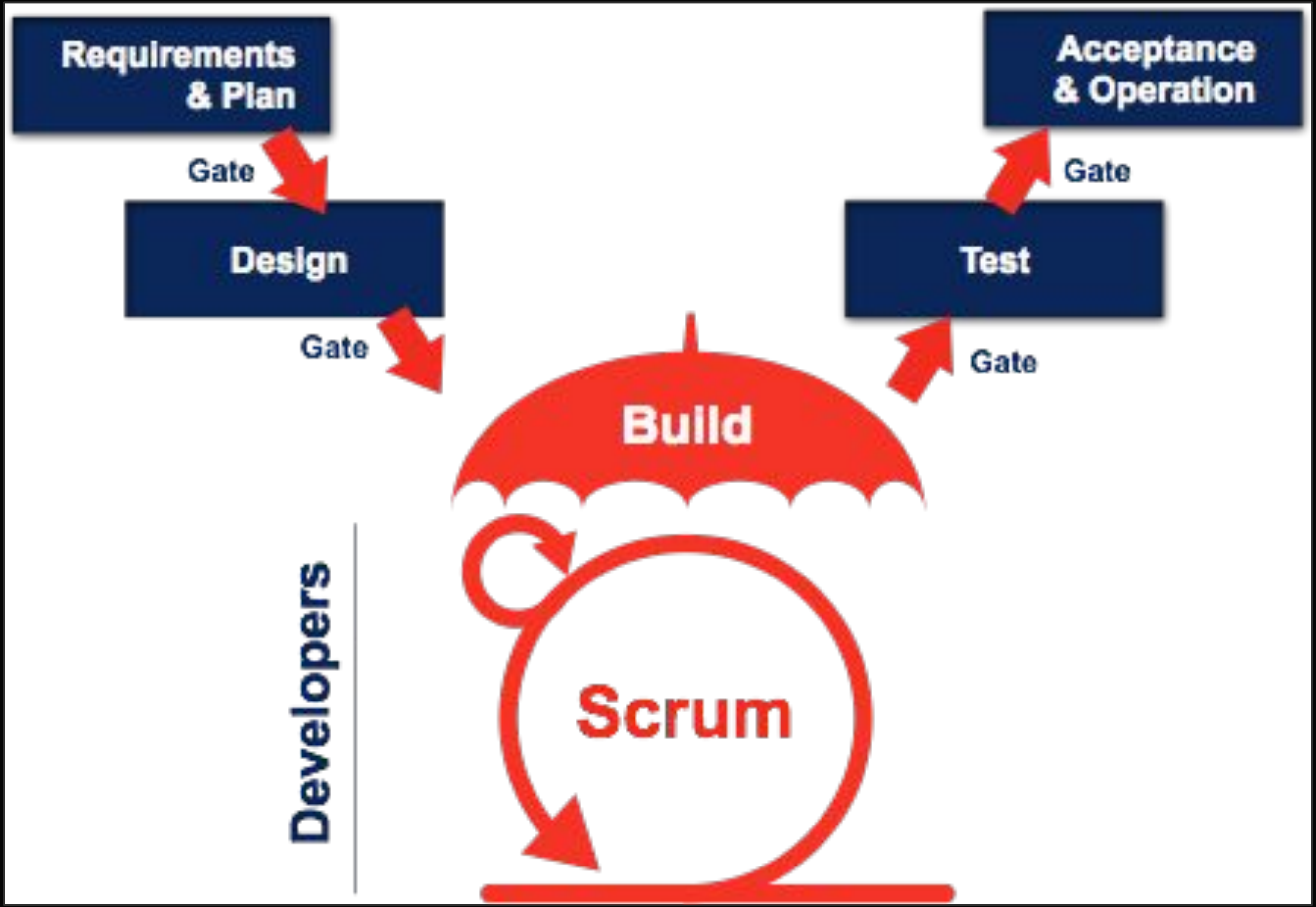




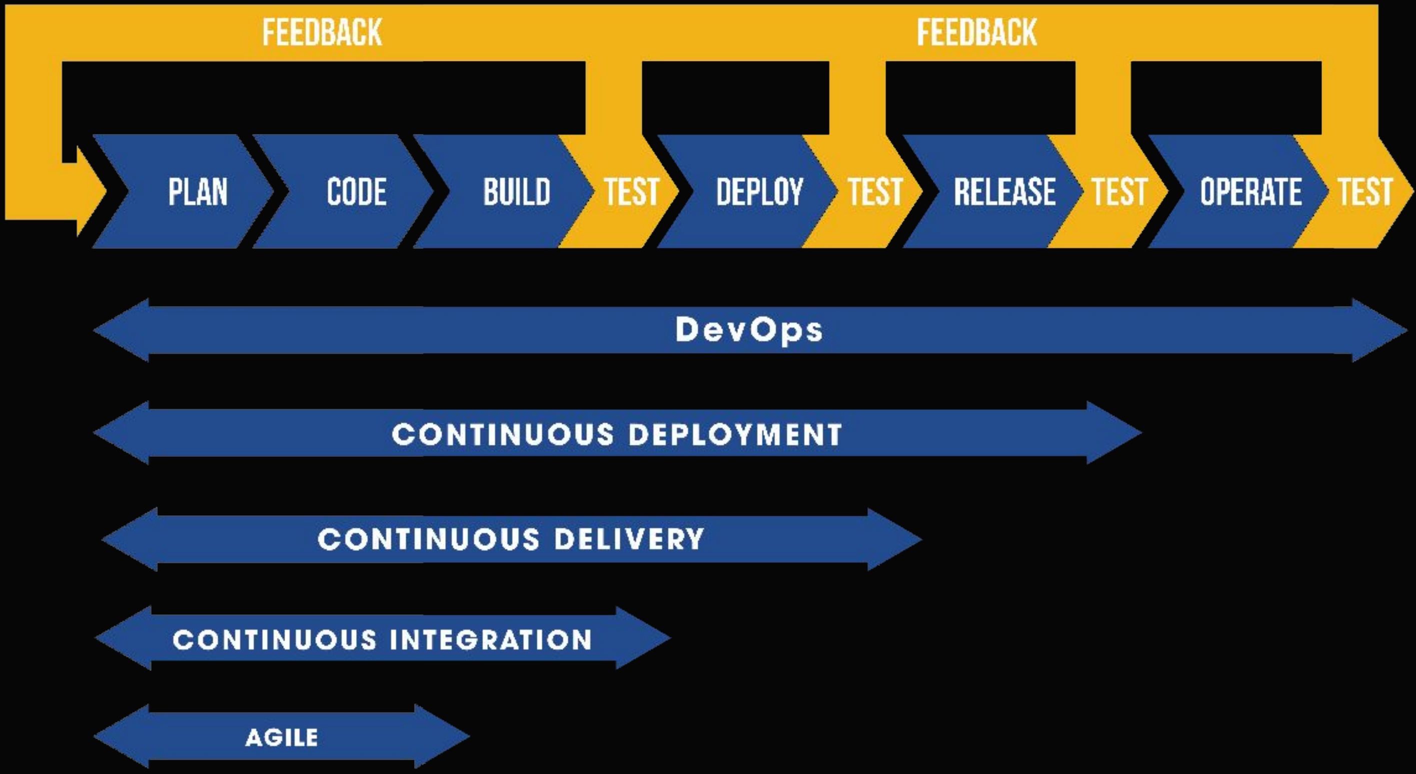
Our highest priority is to satisfy the customer through early and **continuous delivery** of valuable software.

So How Does Integration & Test Fit into Agile/DevSecOps?

Water-Scrum-Fall (aka FrAgile) is the Wrong Answer!



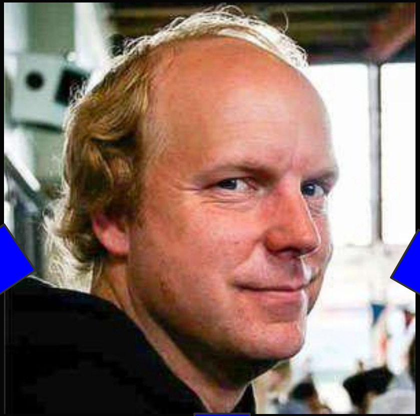
More Testing! (Not Less!)



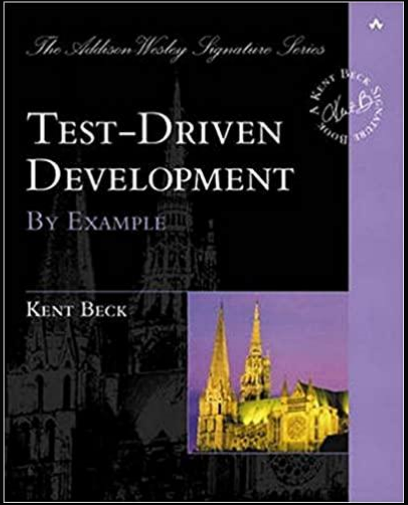
“Continuous is much more often than you think.” - Mike Roberts, Thoughtworks

Test Driven Development - Background

Kent Beck



2002



The Addison Wesley Signature Series

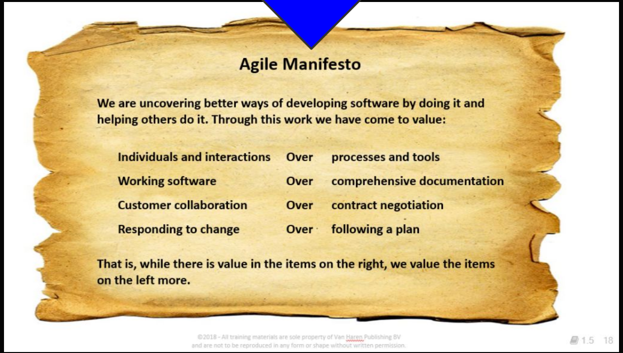
A Kent Beck Signature Book

TEST-DRIVEN DEVELOPMENT

BY EXAMPLE

KENT BECK

2001



Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions	Over	processes and tools
Working software	Over	comprehensive documentation
Customer collaboration	Over	contract negotiation
Responding to change	Over	following a plan

That is, while there is value in the items on the right, we value the items on the left more.

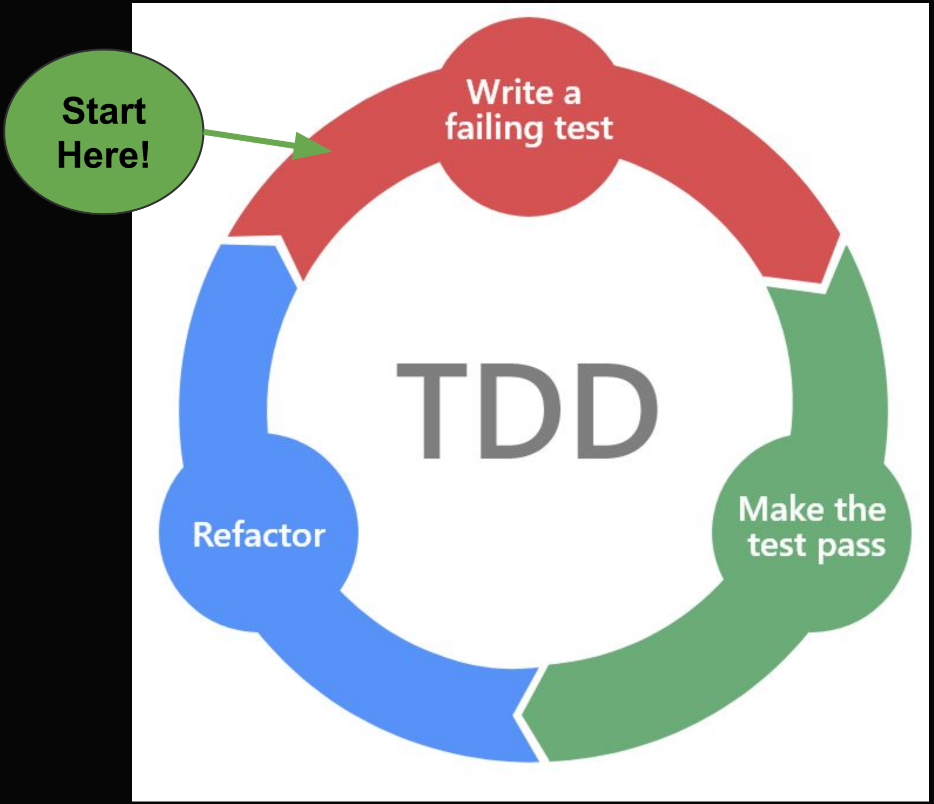
1.5 18

2002

JUnit

Test Driven Development in Practice (FYI...This is Hard!)

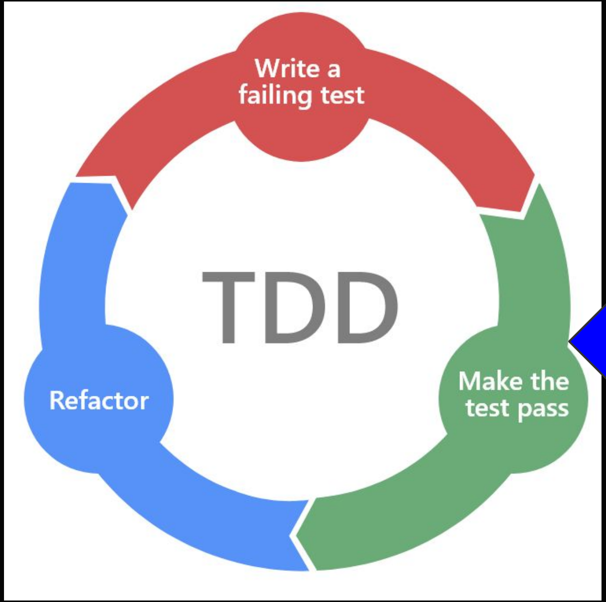
- Red
  - Understand the problem (i.e. requirements) and the expected outcome (i.e. success criteria)
  - Write the test to demonstrate the expected outcome
  - Ensure the test fails
    - Or claim success!
- Green
  - Do the minimum work necessary to make the test pass
  - Demonstrate with objective evidence
- Refactor
  - Ensure implementation:
    - Meets Standards
    - Passes Quality Checks
    - Is Adequately Documented
    - Etc.



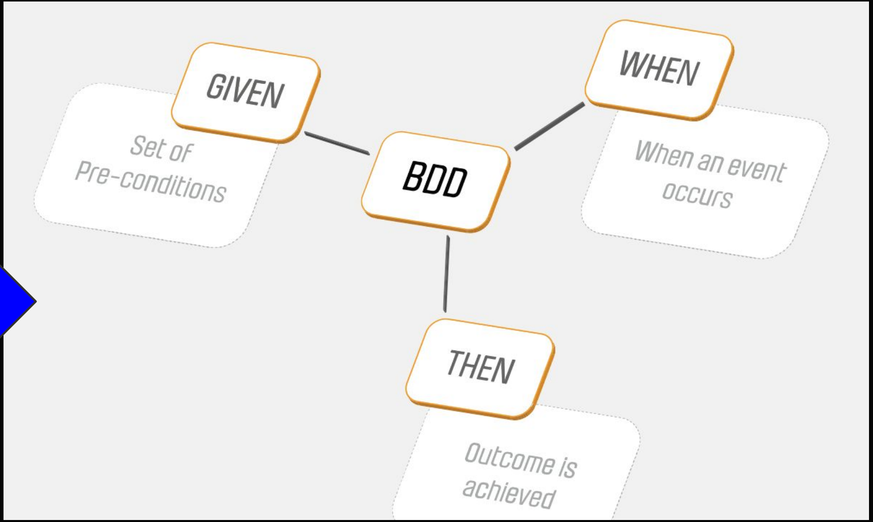
I&T "Shifts Left" and Aligns More Closely with System Engineering

Behavior Driven Development (TDD evolved)

Dan North



2006 https://dannorth.net/introducing-bdd/

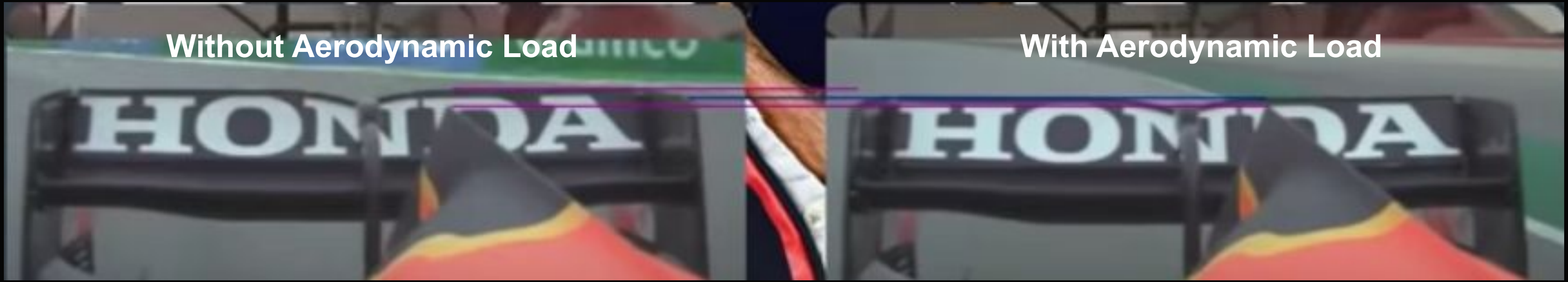


BDD follows TDD Cycle but Elevates to Focus on Value/Outcomes

Don't Just Test an Implementation,

Ensure the Implementation Delivers the Value the Customer Needs

Current Events Example - Formula 1 “Flexi-Wings”



Requirement:

any specific part of the car influencing its aerodynamic performance must remain immobile in relation to the sprung part of the car

Right to Revise Test:

the FIA reserves the right to introduce further load/deflection tests on any part of the bodywork which appears to be moving while the car is in motion

Test Specification:

FLEXIBILITY TESTS

Put the car under specific loads and measure how much it deforms in particular directions

Bodywork may not move more than 1° horizontally when subject to rearward force of 1000 N (at defined points)

Bodywork may not move more than 3 mm vertically when subject to downward force of 500 N (at defined points)

How does TDD/BDD fit into a DevSecOps SW Factory?

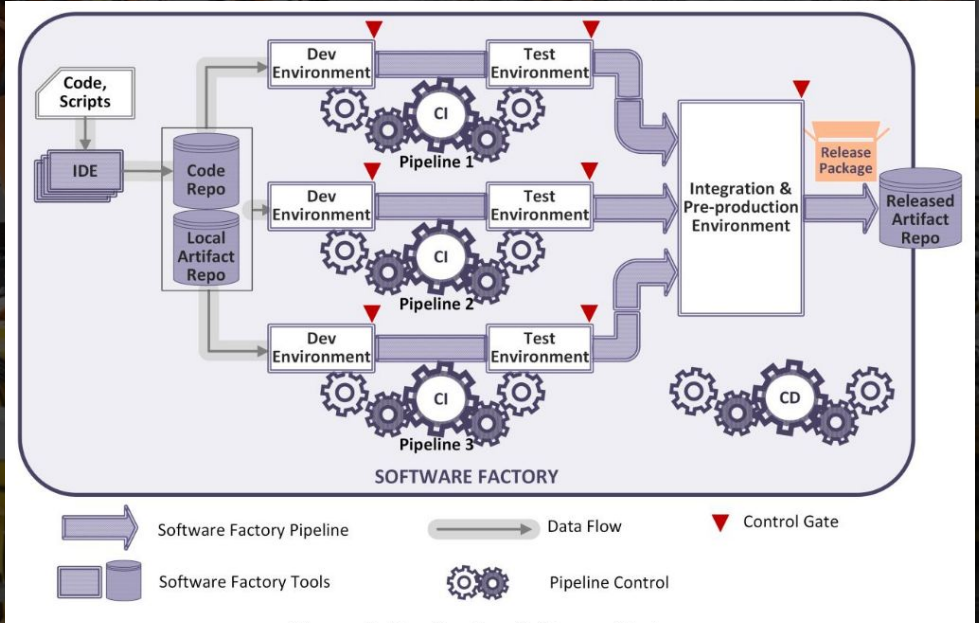


Figure 9: DevSecOps Software Factory

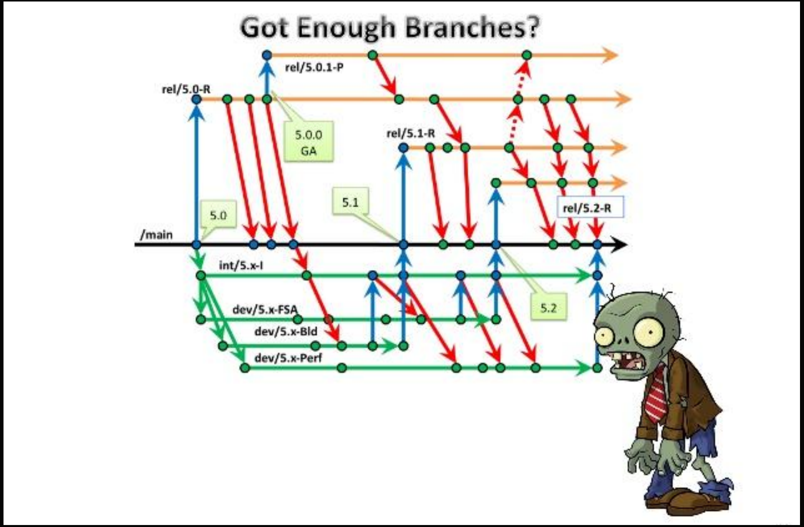
Good Testing

1. Enables Speed
2. Ensures Quality
3. Reduced Rework

Must be

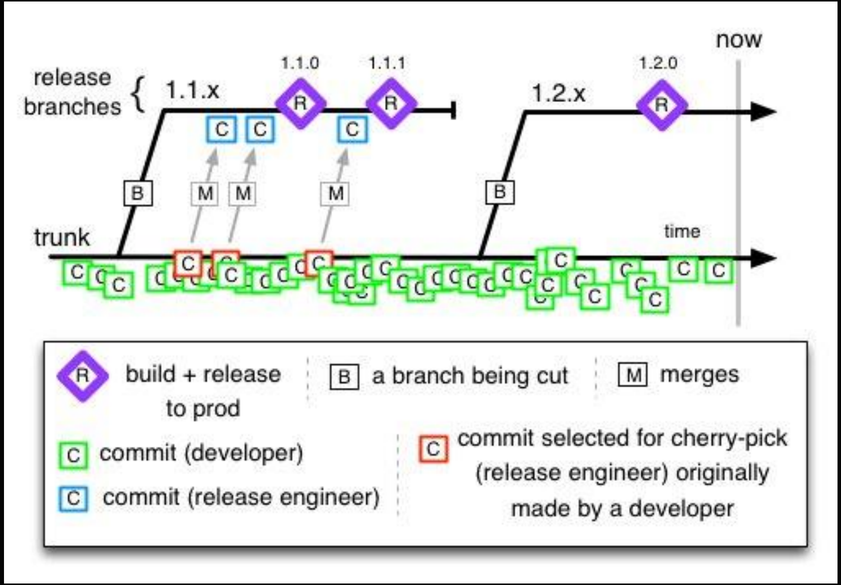
1. Fast
2. Affordable
3. Repeatable
4. Reliable

SW Factory CI/CD Prerequisite: Good CM



"CM Smells"

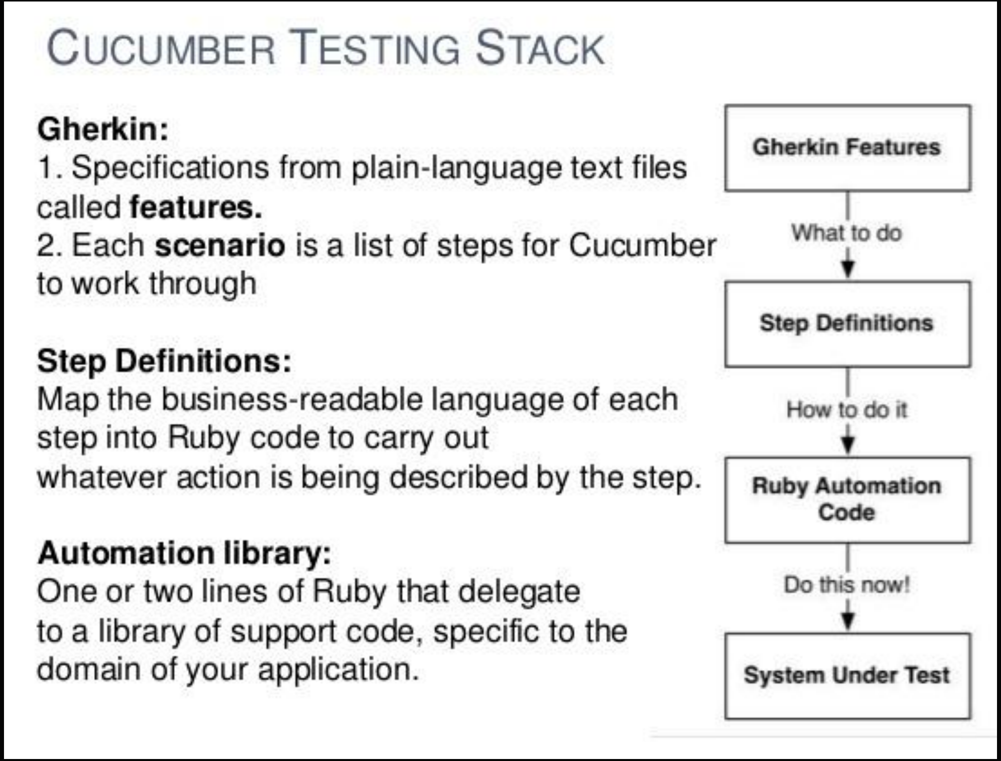
- Lots of branches
- Long lived branches
- Large complex merges take a long time
- Deep branch hierarchy (e.g. based on org)
- You know what a config spec is and have mastered manipulating it


now
release branches
1.1.x
1.1.0
1.1.1
1.2.x
1.2.0
R
C
B
M
trunk
time
R build + release to prod
B a branch being cut
M merges
C commit (developer)
C commit (release engineer)
C commit selected for cherry-pick (release engineer) originally made by a developer

Trunk-Based Development

- Recognize that branches defer integration
- Commit to trunk / master is the ideal
- Use short lived feature branches (aka Pull Requests) if that works best for your team
- Monitor branch lifespan & kill them off

Automated Acceptance Testing...the BDD Way

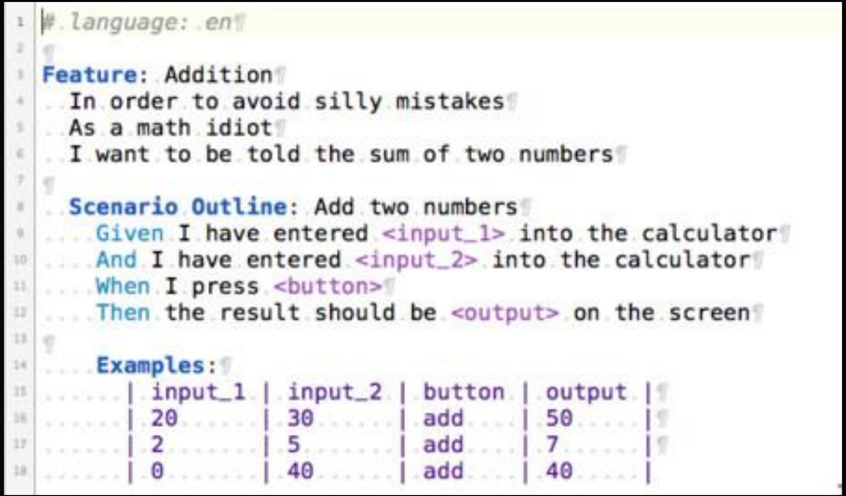


CUCUMBER TESTING STACK

Gherkin:
1. Specifications from plain-language text files called **features.**
2. Each **scenario** is a list of steps for Cucumber to work through

Step Definitions:
Map the business-readable language of each step into Ruby code to carry out whatever action is being described by the step.

Automation library:
One or two lines of Ruby that delegate to a library of support code, specific to the domain of your application.

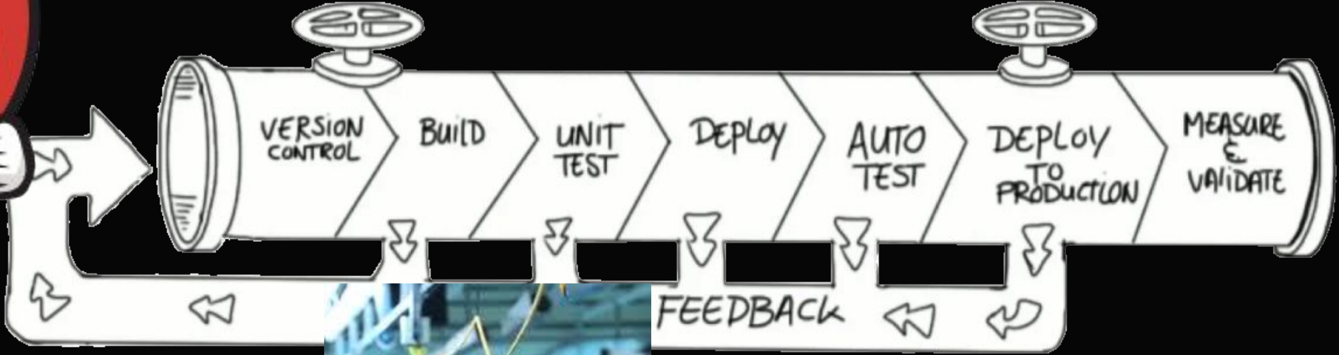


```
# language: en

Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario Outline: Add two numbers
    Given I have entered <input_1> into the calculator
    And I have entered <input_2> into the calculator
    When I press <button>
    Then the result should be <output> on the screen

    Examples:
      | input_1 | input_2 | button | output |
      | 20      | 30      | add    | 50     |
      | 2       | 5       | add    | 7      |
      | 0       | 40      | add    | 40     |
```

*Supports multiple implementation languages...including Java

The CI/CD Fundamentals

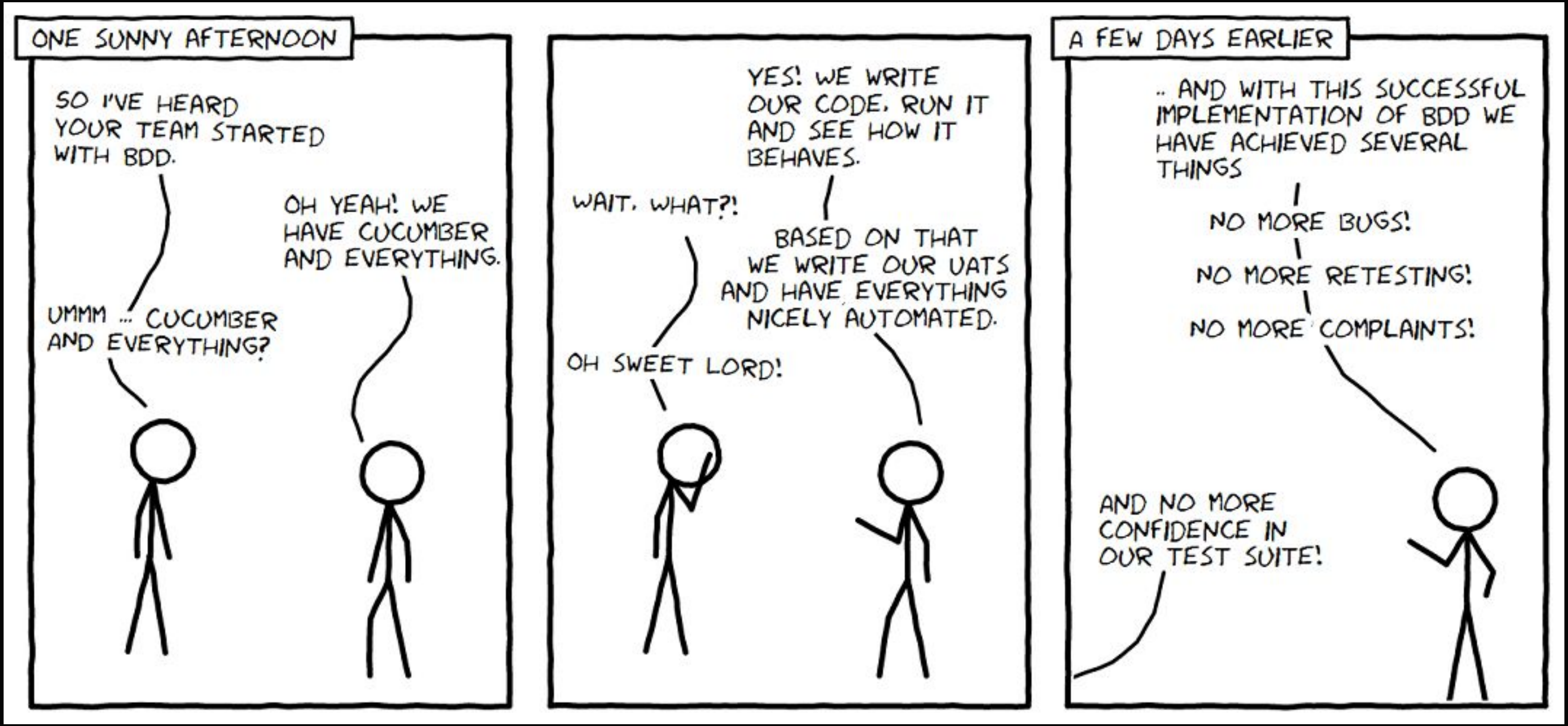
2) Your pipeline steps delegate to an automate build tool (e.g. Gradle)



1) Use a pipeline tool to automate execution (e.g. Jenkins)

3) When something fails in your pipeline, stop and fix it <u>immediately</u> (e.g. Andon Cord)

Proceed with Caution…..


ONE SUNNY AFTERNOON
SO I'VE HEARD YOUR TEAM STARTED WITH BDD.
OH YEAH! WE HAVE CUCUMBER AND EVERYTHING.
UMMM ... CUCUMBER AND EVERYTHING?
YES! WE WRITE OUR CODE, RUN IT AND SEE HOW IT BEHAVES.
WAIT, WHAT?!
BASED ON THAT WE WRITE OUR UATS AND HAVE EVERYTHING NICELY AUTOMATED.
OH SWEET LORD!
A FEW DAYS EARLIER
.. AND WITH THIS SUCCESSFUL IMPLEMENTATION OF BDD WE HAVE ACHIEVED SEVERAL THINGS
NO MORE BUGS!
NO MORE RETESTING!
NO MORE COMPLAINTS!
AND NO MORE CONFIDENCE IN OUR TEST SUITE!

BDD Demonstration Repo

1. Test an application I didn’t write
2. Strict Black Box / Acceptance Testing
3. Fully Automated Example

https://github.com/jondavid-black/BDD-Introduction

Gradle

cucumber

Group Exercise: Virtual Lean Coffee

1. The facilitator creates a new board on http://agile.coffee and shares the link in the Zoom chat with everyone.
2. The facilitator has a short introduction.
3. Everyone makes cards with questions or topics for discussion on the subject.
4. Everyone votes on each question or topic by clicking the thumbs up button on the corner of the card. 5 votes per person.
5. Cards with most votes goes first. Set a timer for 5 minutes and discuss.
6. After 5 minutes, either vote (thumbs up/down in your Zoom window) to keep going or move on to the next card.

Suggested Topics: "Test Driven Mindset", "Test Automation Tools"

DevOps Resources

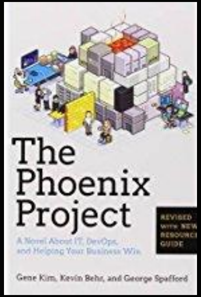
https://devopsfordefense.org/resources/

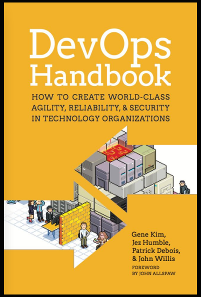
https://www.meetup.com/DevOps-for-Defense/
https://github.com/jondavid-black/DevOpsForDefense
devopsfordefense@gmail.com

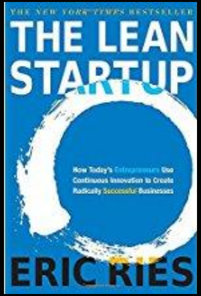
Books / Publications:

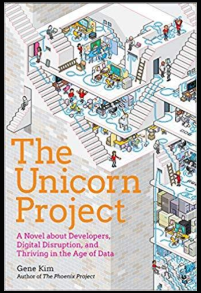
Conference Presentations (YouTube):

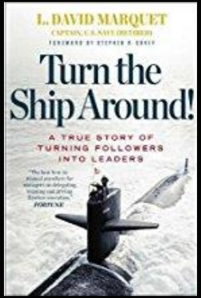
- DevOps Enterprise Summit (DOES)
- IT Revolution
- Velocity
- GoTo

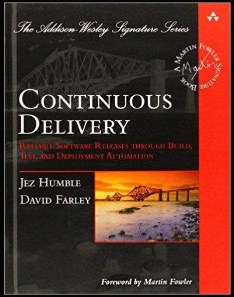

The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford

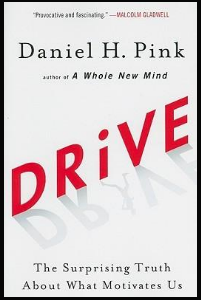

DevOps Handbook
How to Create World-Class Agility, Reliability, & Security in Technology Organizations
Gene Kim, Jez Humble, Patrick Debois, & John Willis

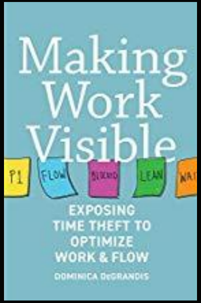


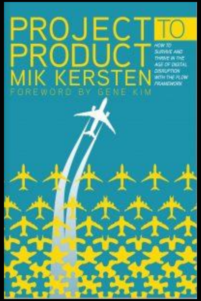

The Unicorn Project
A Novel about Developers, Digital Disruption, and Thriving in the Age of Data
Gene Kim

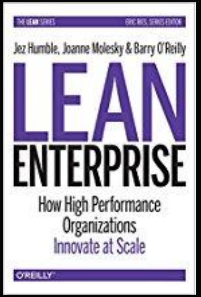


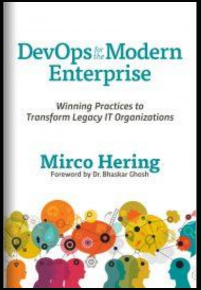


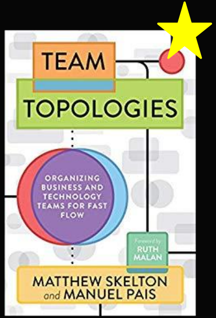







Jez Humble, Joanne Molesky & Barry O'Reilly
Lean Enterprise
How High Performance Organizations Innovate at Scale


DevOps for the Modern Enterprise
Winning Practices to Transform Legacy IT Organizations
Mirco Hering


Team Topologies
Organizing Business and Technology Teams for Fast Flow
Matthew Skelton and Manuel Pais