Grand Valley State University
www.gvsu.edu

Software Engineering Deployment Management

CIS641

Erik Fredericks // fredericks@oakland.edu

Adapted from materials provided by Gregory Schymik


- CHANGE IS GOOD.
- YEAH, BUT IT'S NOT EASY.

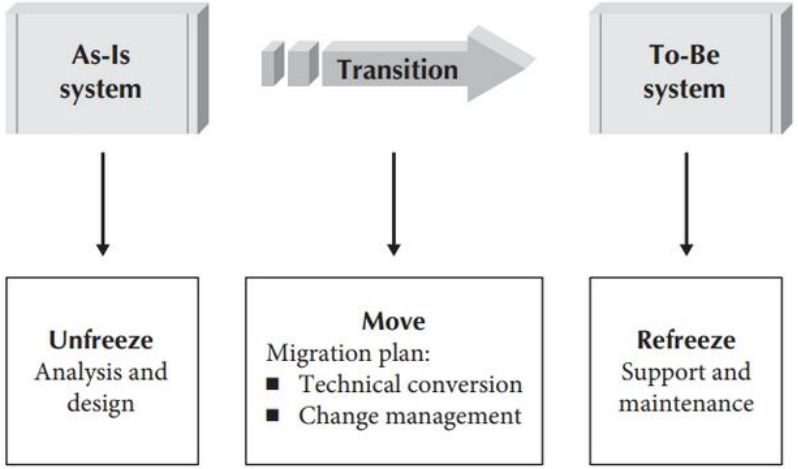

Change
Management
risk
opportunity
process
complex
customer
plan
planning
analysis
retention
data
assessment
impact
project
cost
evaluation
treatment
strategy
research
resources
context
performance
important
scope

Plans for today

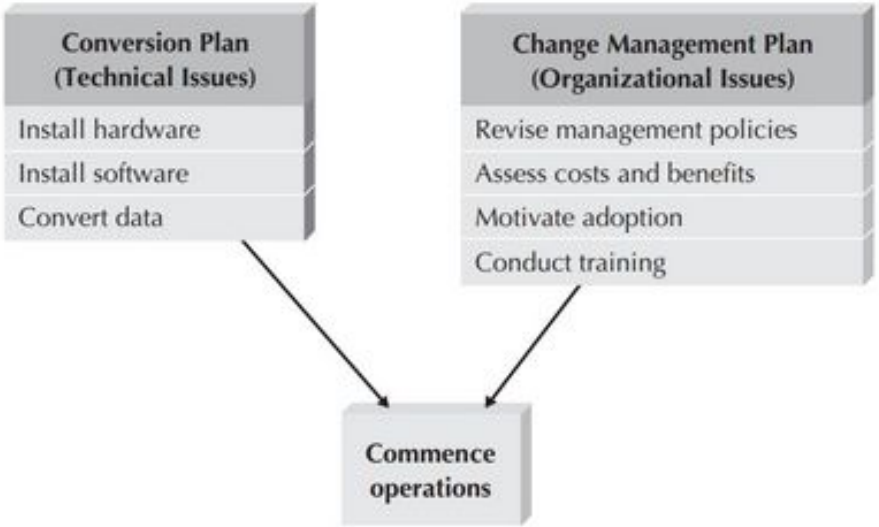
Research paper

Change and Application Deployment

Group planning time



Often overlooked!!
...but why?



Organizational change agents

Nine Reasons Why People Resist Change (Hill, 2009, p. 47):

1. They believe the change is unnecessary or will make things worse.
2. They don't trust the people leading the change effort.
3. They don't like the way the change was introduced.
4. They are not confident the change will succeed.
5. They did not have any input or in planning and implementing the change effort.
6. They feel that change will mean personal loss — of security, money, status, or friends.
7. They believe in the status quo.
8. They've already experienced a lot of change and can't handle any more disruption.
9. They're afraid they don't have the skills to do their work in new ways required by the change.

2/01/reasons-why-

Six Reasons Why People Support Change (Hill, 2009, p. 47):

1. They believe the change makes sense and that it is the right course of action.
2. They respect the people leading the change effort.
3. They anticipate new opportunities and challenges that come from the change.
4. They were involved in planning and implementing the change effort.
5. They believe the change will lead to personal gain.
6. They like and enjoy the excitement of change.

Change strategies

Rational - Empirical Approach

- Paint a picture - what's in it for them?

Normative - Re-education Approach

- Fix the "That's not how we do things around here" culture

Power - Coercive Approach

- Reward/punish

Environmental - Adaptive Approach

- Monday morning

Group discussion (~10 minutes)

Teamwork!
Your teams will be deploying an application that will change. the. world.
Or it will disrupt things, whatever the latest buzzword is.

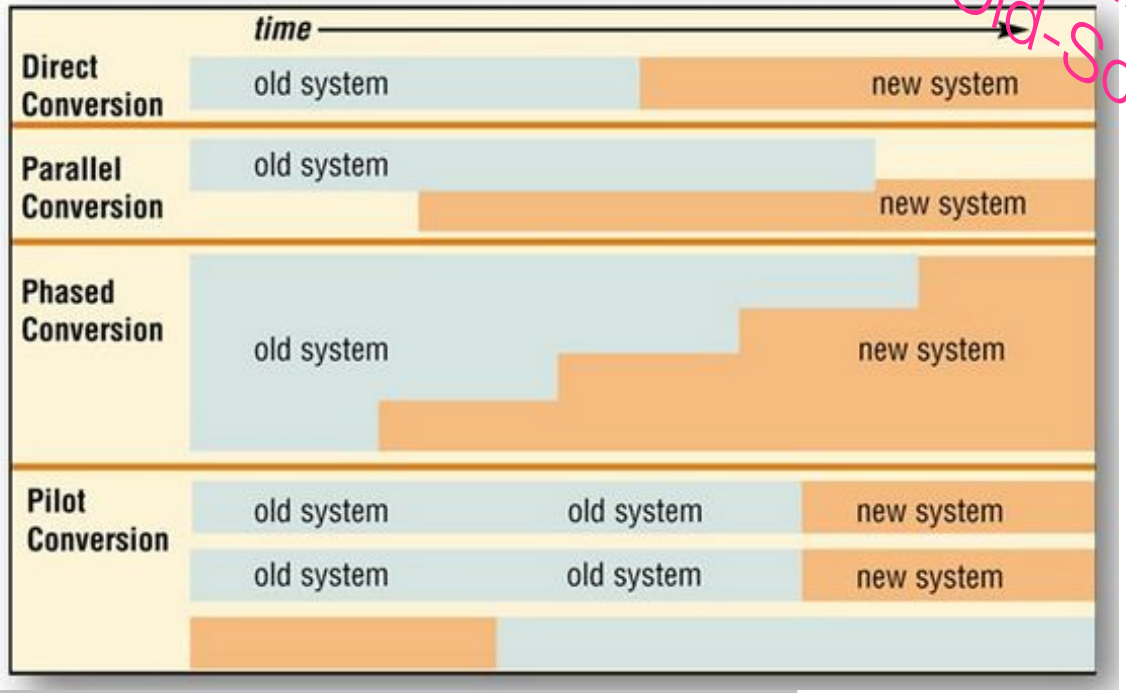
Takeaway:

> Come up with a **change process** to transition a customer from an **old product** to a **new product**

Consider:

- Employees will resist change, how will you convince them?
- Managers will fall back on their old ways, how will you change their minds?
- Other organizational issues you may think of?

Application Deployment



This is Old-School!!!

	Conversion Style		Conversion Location			Conversion Modules	
Characteristic	Direct Conversion	Parallel Conversion	Pilot Conversion	Phased Conversion	Simultaneous Conversion	Whole-System Conversion	Modular Conversion
Risk	High	Low	Low	Medium	High	High	Medium
Cost	Low	High	Medium	Medium	High	Medium	High
Time	Short	Long	Medium	Long	Short	Short	Long

Application deployment

Guidelines

- Simple - minimal
- Consistent
- Organized
- Secure
- Plan B
- Be Agile
- Checklist
- Continuous
- Automate
- Top Tools

BUILD

DEPLOY

MANAGE

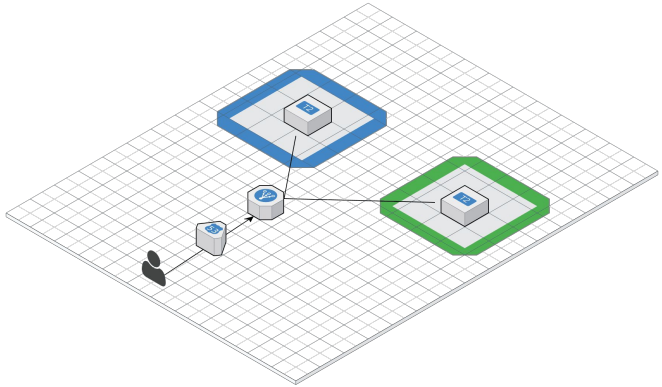
Why Automate?

- Easier to repeat
- More time for dev
- No need for dedicated team
- Higher frequency

Modern deployment schemes

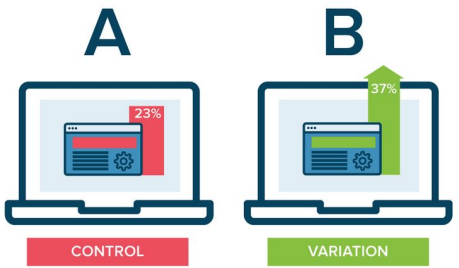
Recreate
- Switch off old, start new

Ramped
- Slowly replace instances of old with new



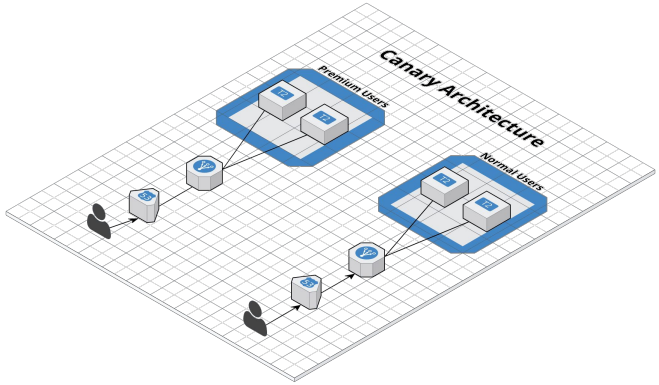
Blue-Green

A/B Testing

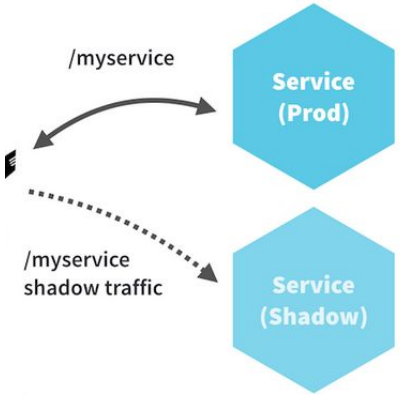


Modern deployment schemes

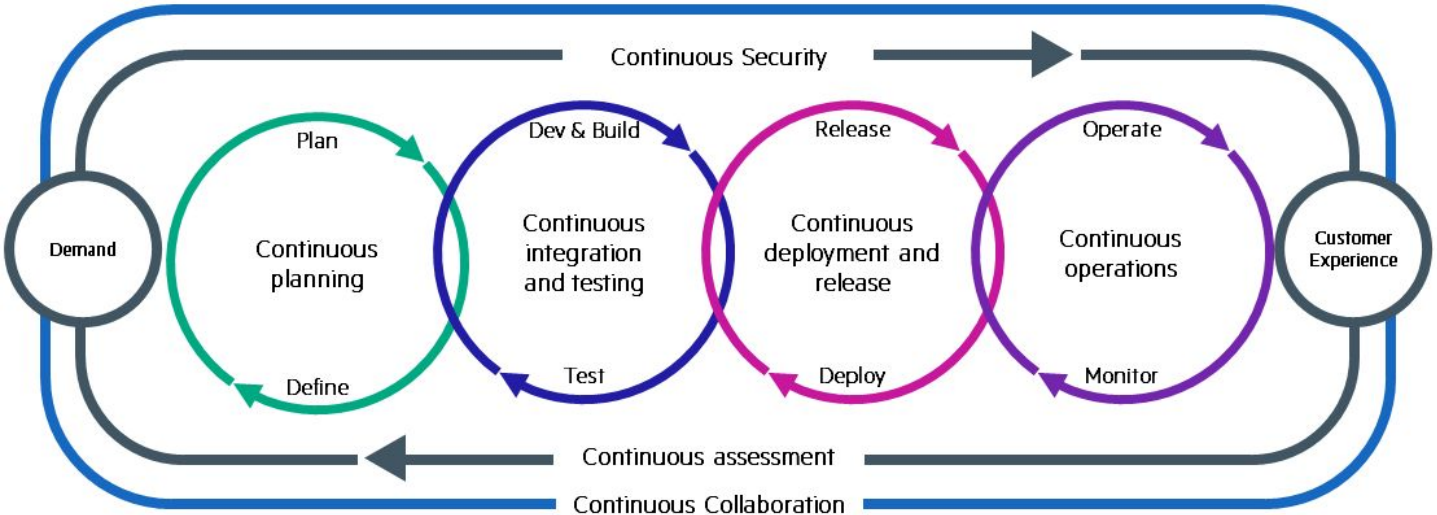
Canary



Shadow



DevOps



https://software.microfocus.com/en-us/solutions/devops-solutions

...

DevOps is not...

A product ➜ philosophy, approach, method

An interchangeable buzzword ("Agile", "Cloud", "Automation")

A separate team ➜ no new silos!!

A staff **consolidation** technique

New ➜ a new term but collaboration has existed for a long time

Only about dev and ops ➜ it is a culture

- A set of processes and practices that optimize service delivery

DevOps requires...

A set of processes and practices that optimize service delivery

- Culture
- Automation
- Measurement
- Sharing

https://www.youtube.com/watch?v=aFWi8ToAjpU (2 minutes)
https://www.youtube.com/watch?v=_I94-tJIovg (7 minutes ➔ watch if you want)

DevOps pipeline

Build

Unit Test

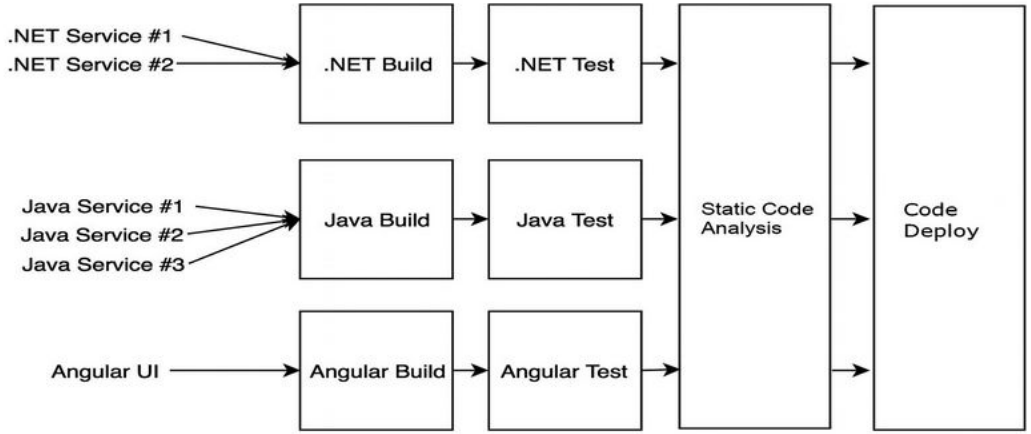
Static Code Scan/Security Scan

Packaging Publishing of Artifacts

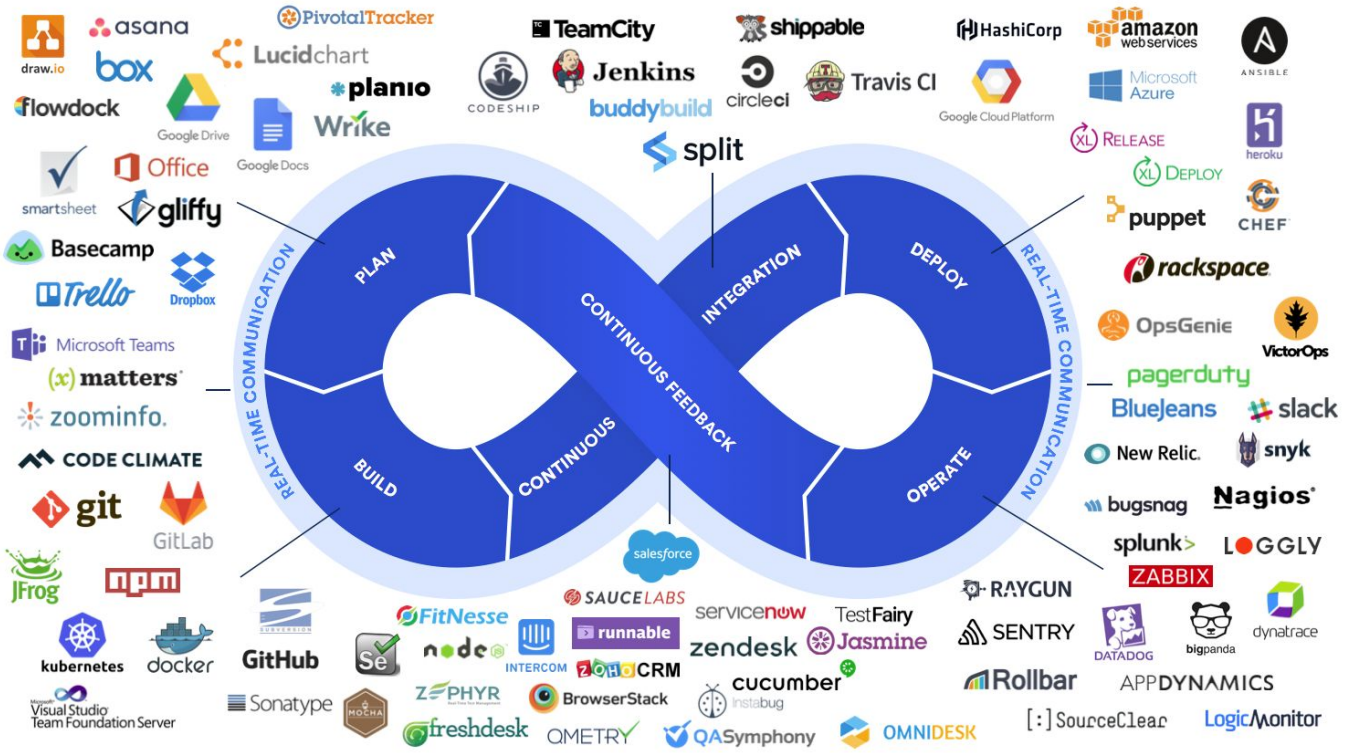
Deploying

End to End Tests

Performance Tests



https://www.youtube.com/watch?v=Me3ea4nUt0U
(10 minutes in)


PLAN
CODE
BUILD
CONTINUOUS
INTEGRATION
DEPLOY
OPERATE
CONTINUOUS FEEDBACK
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
box
Flowdock
Google Drive
PivotalTracker
Lucidchart
planio
Google Docs
Wrike
smartsheet
Office
gliffy
Basecamp
Trello
Dropbox
Microsoft Teams
(x) matters
zoominfo.
CODE CLIMATE
git
GitLab
JFrog
npm
kubernetes
docker
Visual Studio Team Foundation Server
GitHub
Sonatype
CODESHIP
TeamCity
Jenkins
buddybuild
shippable
circleci
Travis CI
split
salesforce
SAUCELABS
FitNesse
node
INTERCOM
runnable
ZOHO CRM
ZEPHYR
BrowserStack
freshdesk
QMETRY
servicenow
TestFairy
zendesk
Jasmine
cucumber
Instabug
QASymphony
OMNIDESK
HashiCorp
Google Cloud Platform
amazon web services
Microsoft Azure
RELEASE
DEPLOY
ANSIBLE
heroku
puppet
CHEF
rackspace
OpsGenie
VictorOps
pagerduty
BlueJeans
slack
New Relic.
snyk
bugsnag
Nagios
splunk>
LOGGLY
ZABBIX
RAYGUN
SENTRY
DATADOG
bigpanda
dynatrace
Rollbar
APPDYNAMICS
[:] SourceClear
LogicMonitor

SHARE

BUSINESS | JOURNAL REPORTS: TECHNOLOGY

It's Time to Get Rid of the IT Department

It made sense in a bygone era, when technology was separate from the business. Now it just hurts both.

No man is an island. And the IT department shouldn't be one, either.

Despite their mission, which often talks about driving corporatewide innovation and digital transformation, chief information officers, as heads of these departments, are frequently reduced to running a metaphorical island. Just look at any organization's structure, and you are very likely to see a rectangular box labeled IT, with its own management hierarchy and budget.

Let's all have a kneejerk reaction from the headline!

Thoughts on removing the IT department?

Thoughts on having each dept. have their own IT dept.?

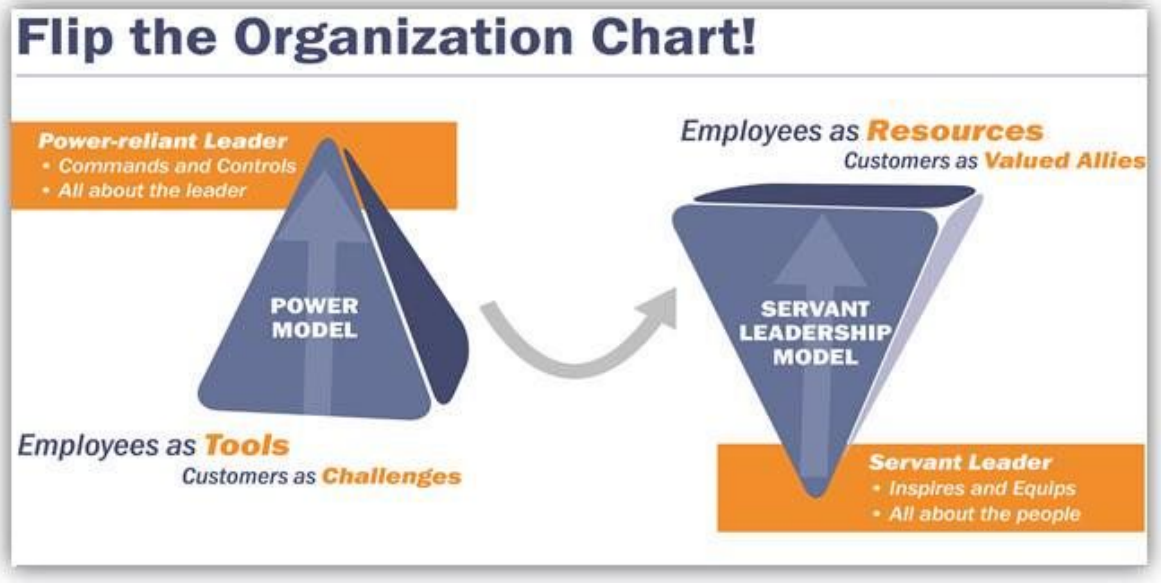
Managing technical people

The great irony of management is that the higher up you go, the less actual control you have.

Your job, as manager, is to make others more productive

Unfortunately, as you probably already know, people suck. People flake, screw up, ignore you, pester you, disappear, quit, and/or lose faith in you, frequently for no good reason. Guess what? Now that you're a manager, their screwups are your screwups, and their problems are your problems. Get used to it. You need to develop an early-warning sense of people problems and have contingency plans in place.....**once people learn that they can trust you, then you can usually trust them back**

Servant leadership



SERVANT LEADERSHIP CHECKLIST

PM

- ☐ Self-Aware
- ☐ Foresight
- ☐ Collaborative
- ☐ Humble
- ☐ Listen
- ☐ Trusting
- ☐ Integrity
- ☐ Doesn't Abuse Authority
- ☐ Coach
- ☐ Result-Oriented
- ☐ Intellectual Authority
- ☐ Resolve Conflict

THE 11 PILLARS
OF SERVANT LEADERSHIP

1 CALLING

YOU ARE COMPELLED TO LEAD OTHERS BECAUSE OF A BELIEF IN SOMETHING THAT IS LARGER THAN YOURSELF.

2 LISTENING

YOU BELIEVE THAT THE BEST WAY TO UNDERSTAND AND HELP OTHERS IS TO LISTEN TO THEM.

3 EMPATHY

YOU UNDERSTAND THAT EVERYONE HAS THEIR OWN PERSPECTIVE AND YOU TRY TO SEE THE WORLD THROUGH THEIR LENS.

4 HEALING

YOU RECOGNIZE THAT AS A LEADER OF OTHERS, YOU HAVE THE ABILITY TO CHANGE THE NARRATIVE OF THEIR STORIES.

5 AWARENESS

YOU RECOGNIZE THE NEED TO BE AWARE OF YOURSELF AND YOUR SURROUNDINGS, AND CHALLENGE WHAT DOESN'T FEEL RIGHT.

6 PERSUASION

YOUR ROLE IS NOT TO DIRECT OTHERS, BUT TO ENCOURAGE THEM TO MOVE IN A DIRECTION THAT IS BEST FOR THEM.

7 FORESIGHT

YOU HAVE THE ABILITY TO PREDICT AND UNDERSTAND THE IMPACT OF THE ACTIONS, AND HELP NAVIGATE A BETTER COURSE.

8 CONCEPTUALIZATION

AS A LEADER, YOU ARE ABLE TO SHARE THE VISION AND ARTICULATE THE OUTCOME SO THAT YOUR TEAM CAN DETERMINE HOW TO GET THERE.

9 STEWARDSHIP

YOUR ACCOUNTABILITY AND COMMITMENT TO LEAD EXTENDS FAR BEYOND YOUR PEOPLE AND YOUR COMPANY TO COMMUNITY AND PLANET.

10 GROWTH

YOUR SINGLE GREATEST SUCCESS AND ACCOMPLISHMENT AS A SERVANT LEADER IS TO GROW AND DEVELOP YOUR PEOPLE.

11 COMMUNITY

YOUR WORKPLACE CULTURE IS A PLACE WHERE ALL ARE WELCOME AND ALL MATTER.

SERVANT LEADERSHIP
CENTRE OF CANADA

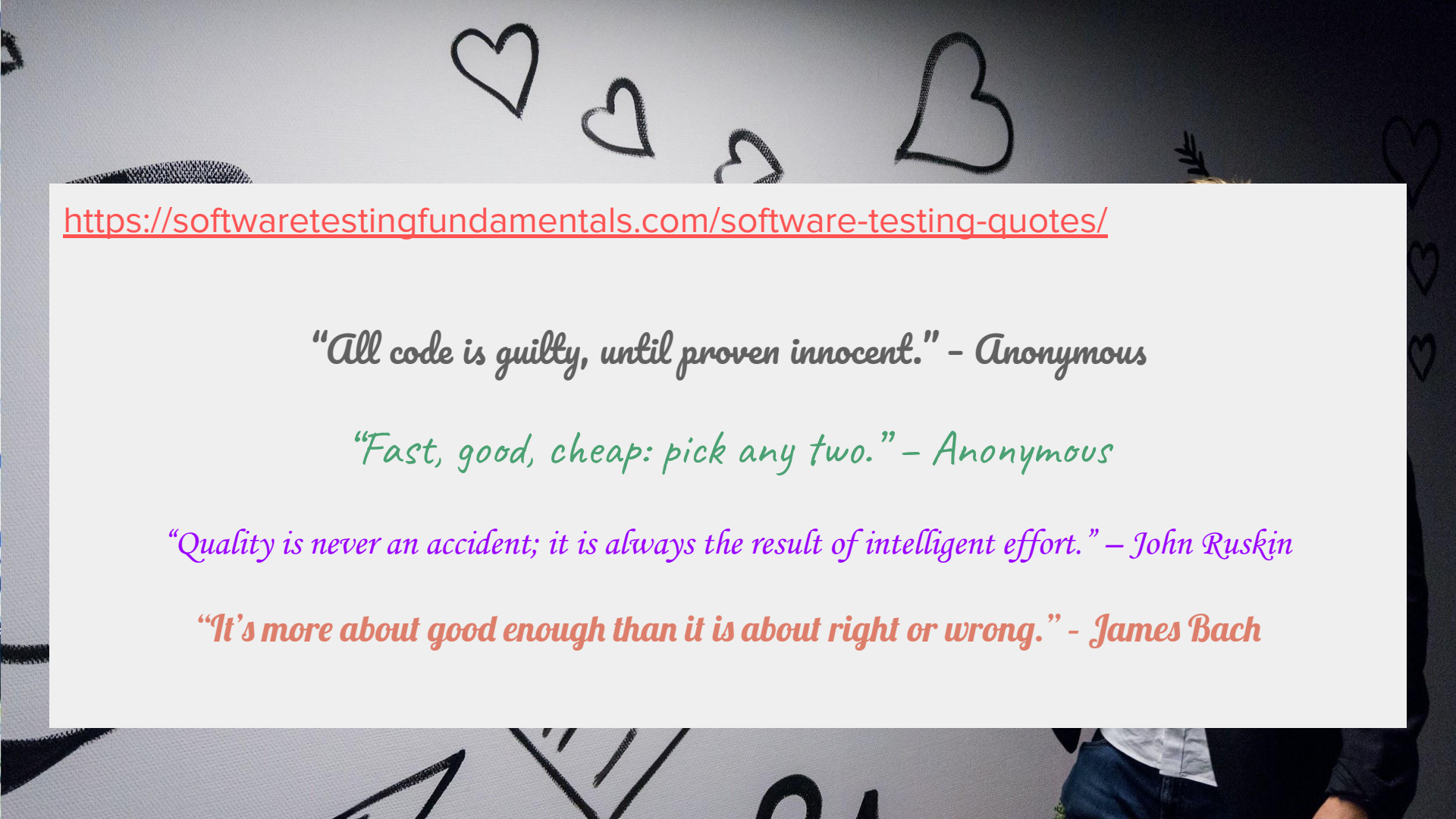
Quality management

Software Quality Control:
"The function of software quality that checks that the project follows its standards, processes, and procedures, and that the project produces the required internal and external (deliverable) products"

Software Quality Assurance:
"The function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented"

Simply put, Quality Assurance focuses on the process of quality, while Quality Control focuses on the quality of output.

“Quality is free, but only to those who are willing to pay heavily for it.”— DeMarco and Lister



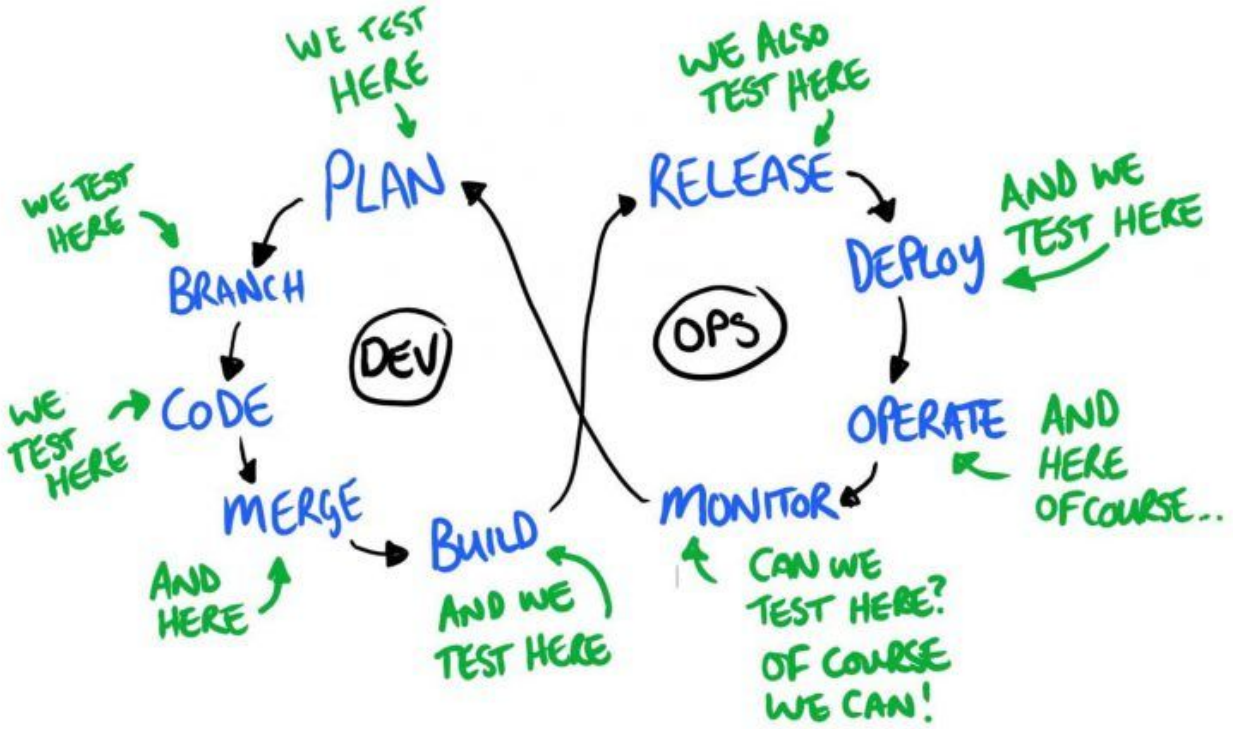
https://softwaretestingfundamentals.com/software-testing-quotes/

"All code is guilty, until proven innocent." – Anonymous

"Fast, good, cheap: pick any two." – Anonymous

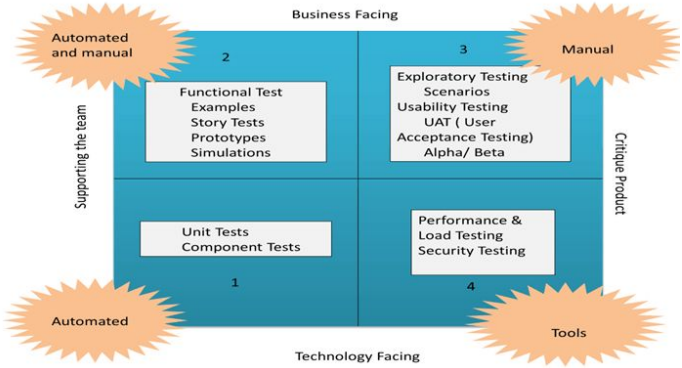
"Quality is never an accident; it is always the result of intelligent effort." – John Ruskin

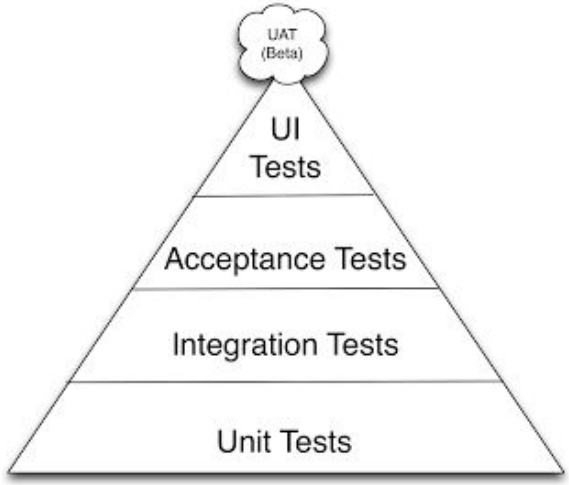
"It's more about good enough than it is about right or wrong." - James Bach



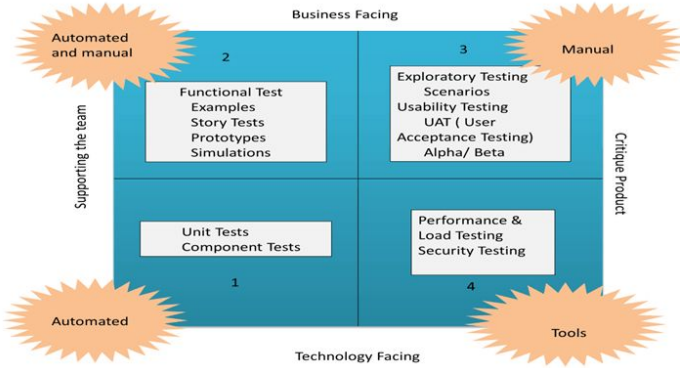
You can **always** test "it"!

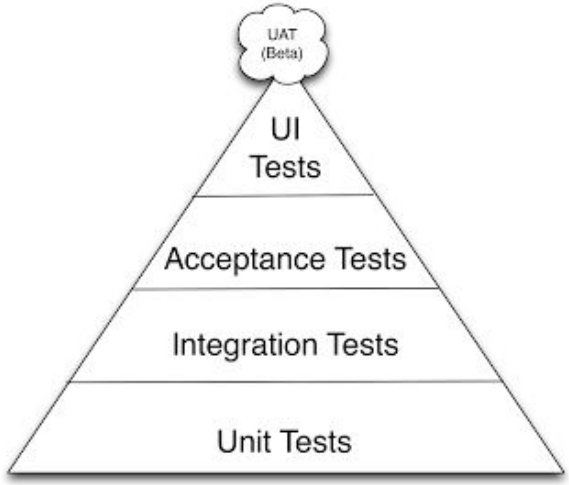
- **Unit tests** are narrow in scope and typically verify the behaviour of individual methods or functions.
- **Integration tests** make sure that multiple components behave correctly together. This can involve several classes as well as testing the integration with other services.
- **Acceptance tests** are similar to the integration tests but they focus on the business cases rather than the components themselves.
- **UI tests** will make sure that the application functions correctly from a user perspective.





- Start writing tests for the critical parts of your codebase.
- Get a CI service to run those tests automatically on every push to the main repository.
- Make sure that your team integrates their changes everyday.
- Fix the build as soon as it’s broken.
- Write tests for every new story that you implement.





What...is...TDD?

TDD is a style of development where the following **two simple rules** are observed:

1. Write new code only if an automated test has failed, and
2. Eliminate duplication and other **code smells**.

Other Names:

- Test-Driven Design
- Test-First Development

TDD processes

The Test-First Stoplight cycle by William Wake:

Stop light progresses through Green, Yellow, and Red repeatedly.
https://xp123.com/articles/the-test-first-stoplight/

Test-first stoplight

Prepare a test list
Start the following process:

- Start (**Green Light**)
- Write a test
- Code may fail to compile (**Yellow Light**)
- Implement just enough (a stub) to compile
- Run the test and ensure it fails (**Red Light**)
- Implement just enough to make the test pass (**Green Light**)
- Improve design by refactoring
- Repeat all existing tests to ensure they are passing (**Green Light**)
- Repeat

The goal is to decrease the interval between writing tests and production code to a matter of a few minutes.

example!

Expect <name>MyName</name>

Green light

1) Create test

Yellow light

2) Method doesn't exist
   a) Add stub to solve

Red light

3) Method fails!

Green light

4) Fix method

```
public class Person {
    String name;
    int favorite = -1;

    public Person (String name, int favorite) {
        this.name = name;
        this.favorite = favorite;
    }
}
```

```
Person p = new Person("MyName", -1);
@Test
public void test_Person() {
  assertEquals("<name>MyName</name>", p.asXml());
}
```

```
(in Person)
public String asXml() {
  return null;
}
```

GREEN LIGHT GREEN LIGHT

```
public String asXml() {
  return "<name>" + name + "</name>";
}
```

Benefits of TDD

Tests actually get written!

- Repeatable verification of working code

Flexibility

- Alleviates fear when cleaning and/or improving code
- High-coverage test suite enables flexible designs

Documentation

- Each unit test explains how some part of the overall system works
- Tests are unambiguous documents that are executable and won't get out of sync with the application
- Tests are low-level design documents

Benefits of TDD

Minimal Debugging

- You know where the bug is, because you just added it
- Small tests covering minimal functional code = small area to search

Better Design

- Most of the code, by definition, is testable
- TDD enforces decoupling
- Result is a highly modular and decoupled software structure

Professionalism

- Clean code
- Flexible code
- Code that works
- On time

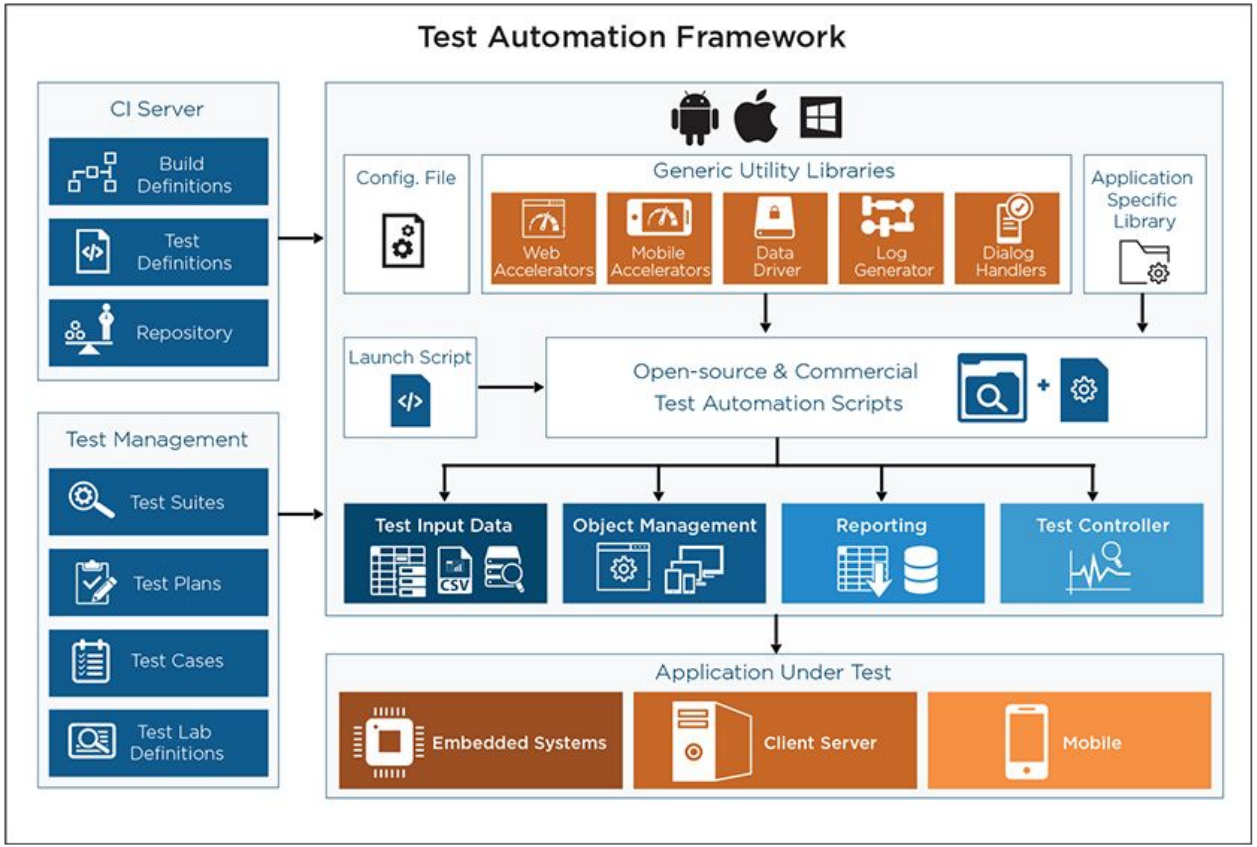
"Quality at speed"

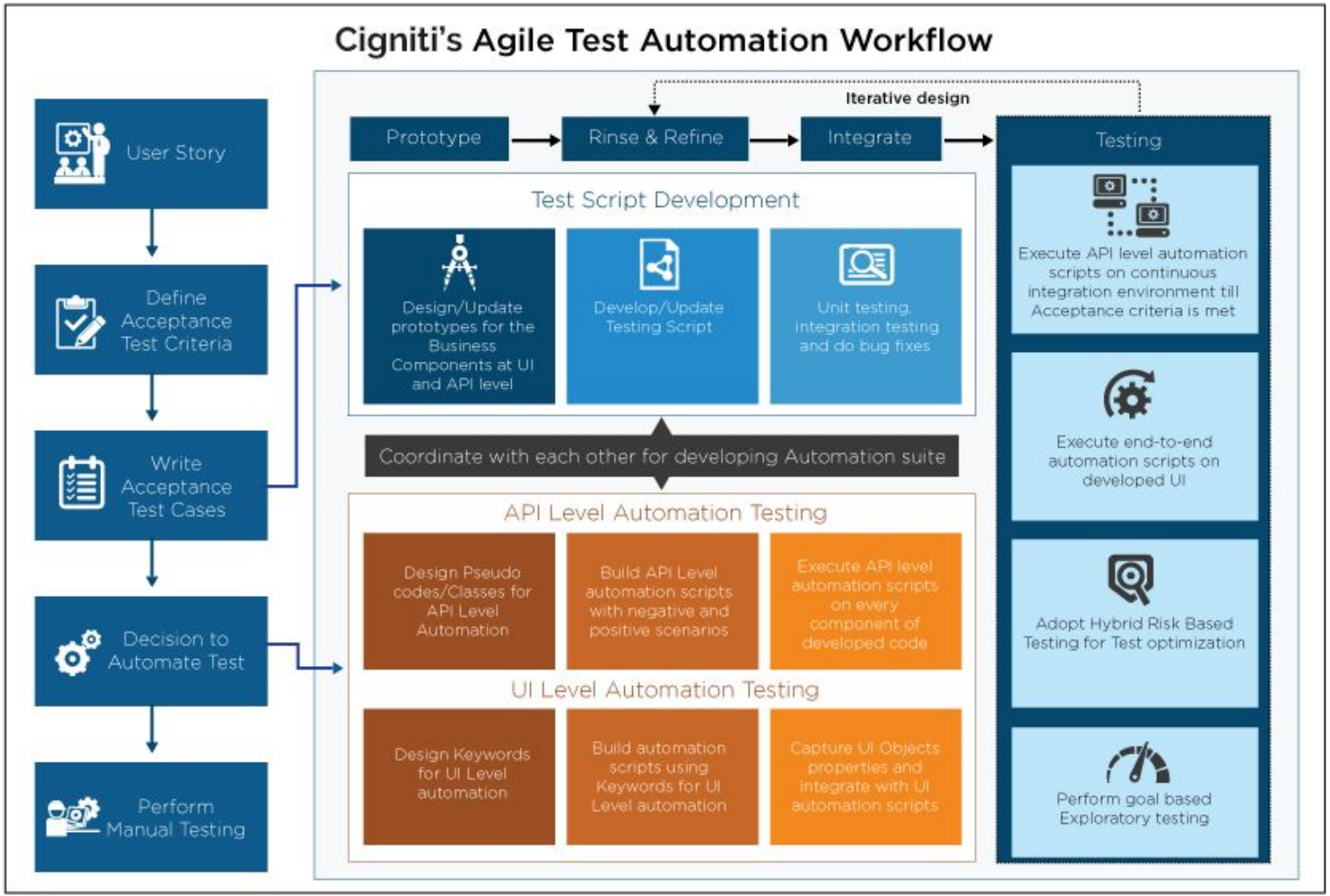
Automated Testing

- ...conducting specific tests via automation as opposed to conducting them manually

Test Automation

- ...automating the tracking and managing of all testing needs, including how much of the system different tests cover and what other types of testing might be required to cover all moving parts.


Test Automation Framework
CI Server
Build Definitions
Test Definitions
Repository
Test Management
Test Suites
Test Plans
Test Cases
Test Lab Definitions
Config. File
Generic Utility Libraries
Web Accelerators
Mobile Accelerators
Data Driver
Log Generator
Dialog Handlers
Application Specific Library
Launch Script
Open-source & Commercial Test Automation Scripts
Test Input Data
CSV
Object Management
Reporting
Test Controller
Application Under Test
Embedded Systems
Client Server
Mobile


Cigniti's Agile Test Automation Workflow
User Story
Define Acceptance Test Criteria
Write Acceptance Test Cases
Decision to Automate Test
Perform Manual Testing
Iterative design
Prototype
Rinse & Refine
Integrate
Test Script Development
Design/Update prototypes for the Business Components at UI and API level
Develop/Update Testing Script
Unit testing, integration testing and do bug fixes
Coordinate with each other for developing Automation suite
API Level Automation Testing
Design Pseudo codes/Classes for API Level Automation
Build API Level automation scripts with negative and positive scenarios
Execute API level automation scripts on every component of developed code
UI Level Automation Testing
Design Keywords for UI Level automation
Build automation scripts using Keywords for UI Level automation
Capture UI Objects properties and integrate with UI automation scripts
Testing
Execute API level automation scripts on continuous integration environment till Acceptance criteria is met
Execute end-to-end automation scripts on developed UI
Adopt Hybrid Risk Based Testing for Test optimization
Perform goal based Exploratory testing

Continuous integration?

TRAVIS DEMO TIME WOO

ONE MORE WEEK OF CLASS!

- It'll be evolutionary

The remainder of class!

- Group planning / discussions for how you're going to manage your deliverables
- Free group work time
- Get professorly help (from me)
    - Free glares!