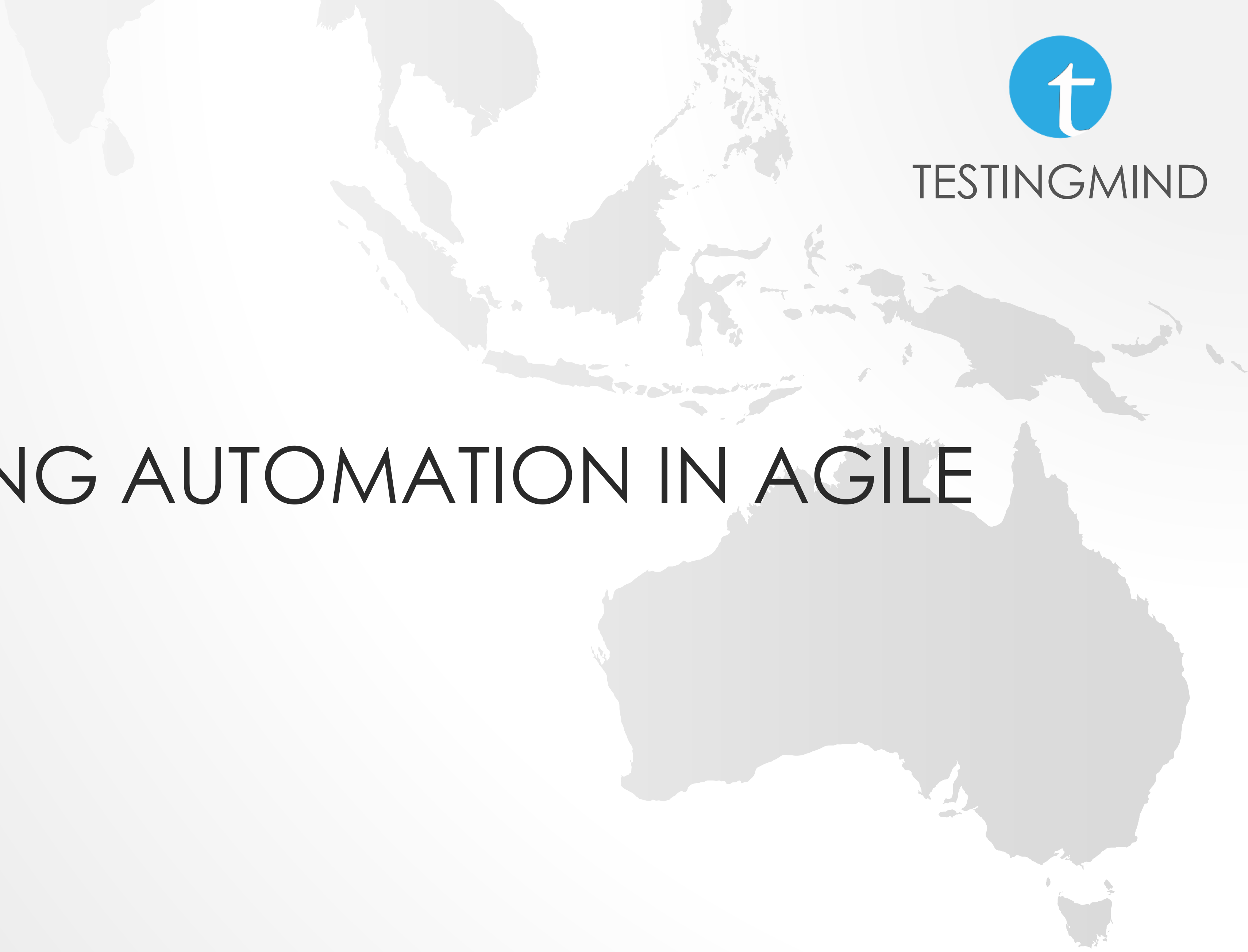


TESTINGMIND

STRATEGIZING AUTOMATION IN AGILE

SPEAKER NAME

location
date , year

Quality Engineering Core Principles

Building software with quality is a complex challenge; our approach to quality is built on these principles.

We believe that quality engineering is **more than testing**

We believe that **close proximity** between quality engineering and development is critical

We believe that the **whole team** must be accountable for quality

We believe that **automation is critical** for high-velocity, high-quality delivery

Automation is Critical

Automation is critical to enable iterative software delivery. However, test automation is challenging and when done incorrectly can consume more time and effort that it saves.

Test Automation should align to the Test **Automation Pyramid**

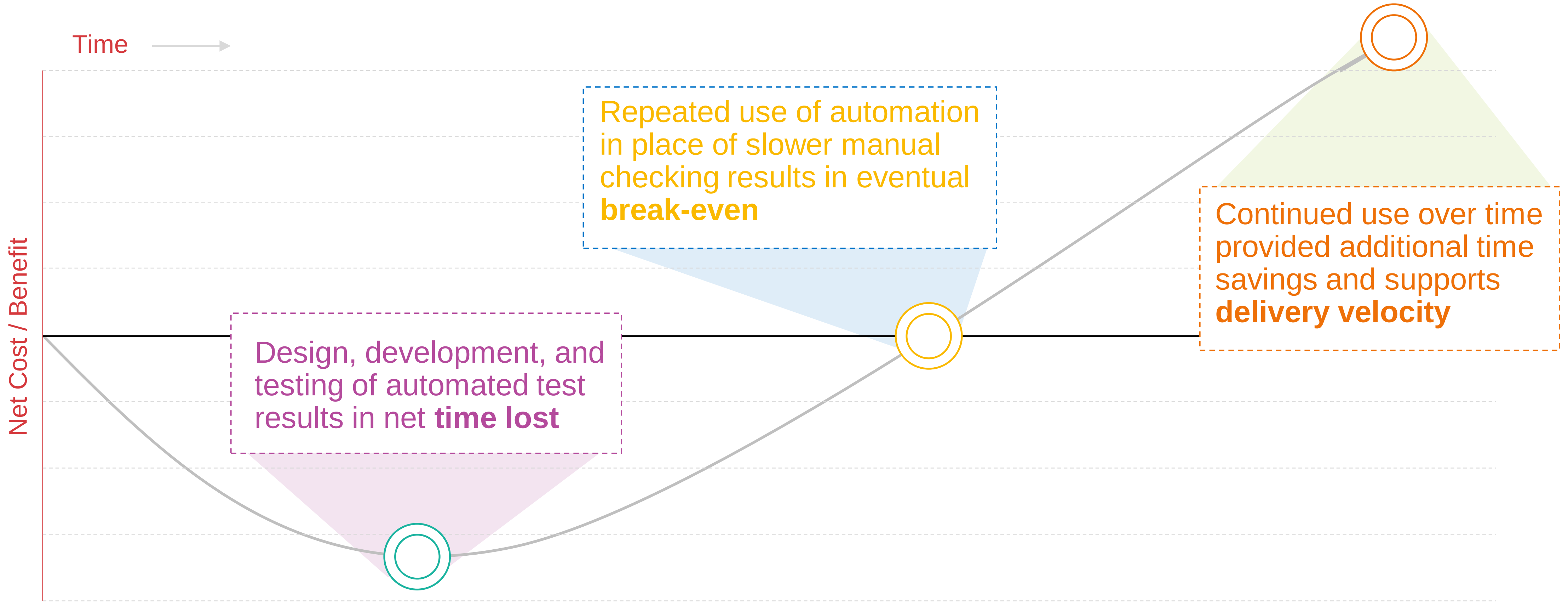
Without Automation, iterative delivery hits the **Regression Death Spiral**

Automation is an investment, and if done poorly, this investment can be wasteful

Automation is a **first class deliverable**, done with the same rigor as all software development

Automation is an Investment

All automation has upfront and ongoing costs. Understanding these costs and being realistic about future benefits is critical to ensuring a successful investment.


Time
Net Cost / Benefit
Repeated use of automation in place of slower manual checking results in eventual **break-even**
Continued use over time provided additional time savings and supports **delivery velocity**
Design, development, and testing of automated test results in net **time lost**

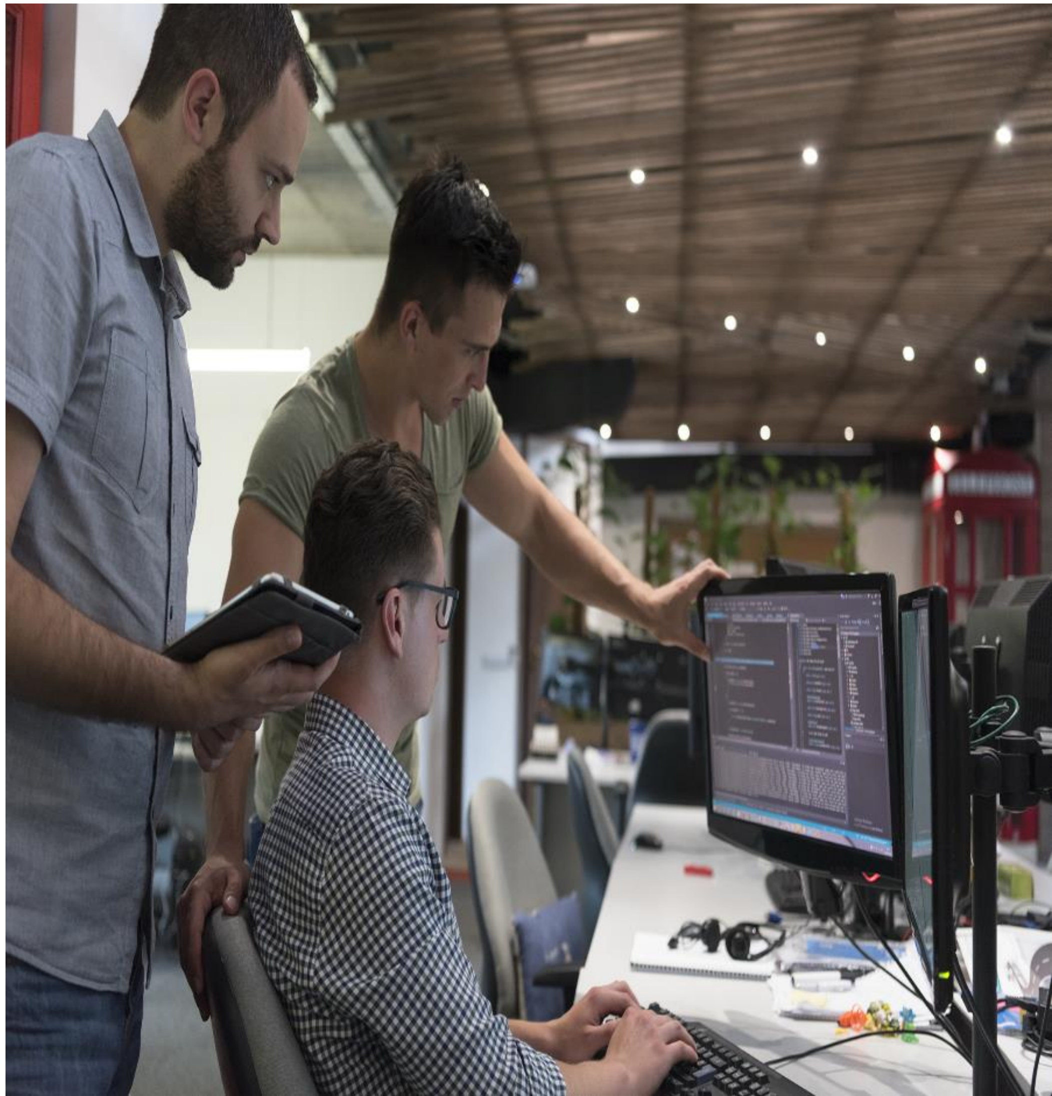
Making Test Automation effective in Agile

- Whole-team approach
- Keep it simple
- Create an automation project plan
  - What is your definition of automated testing
  - What problem are you trying to solve with automation
  - What are the organizations objectives for the automation project
  - How do your test automation objectives support your overall testing goals
  - How does manual testing fit in with your automation testing plan
  - What is your expectation of test coverage with automation?
- Allocate enough time. Take time to do it right.
- Iterative feedback
- Get the “right” automation tool
- Continuous Improvement
- Encourage developers to create application code that is testable.

Core Automated Test Strategy Elements

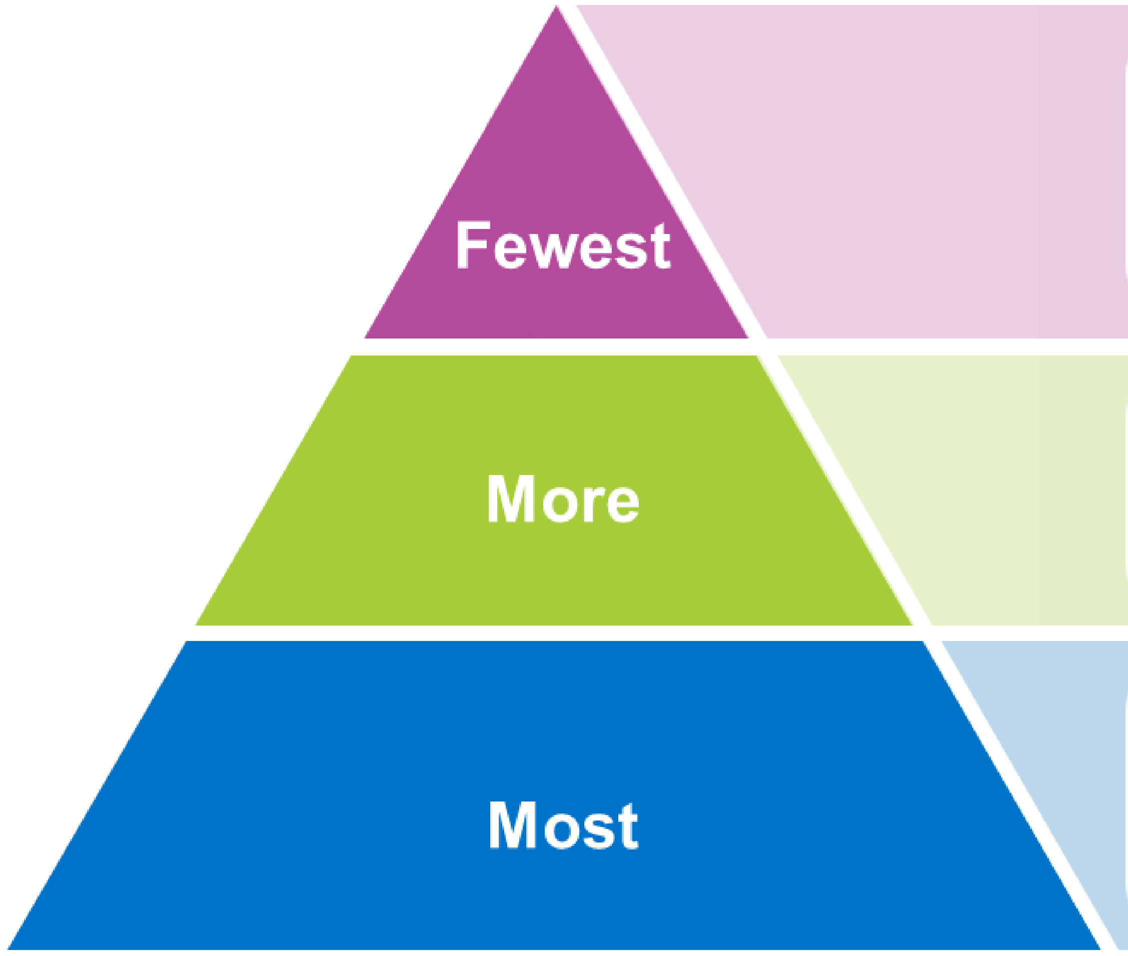
A successful test automation strategy is implemented at multiple levels of the solution stack. Test automation should be reliable, easily and frequently run, and failures easily diagnosed and resolved.

1. A shared automation test framework and tools across the enterprise is better than product specific frameworks
2. Treat automation code just like product code
3. Integrate tests into the DevOps toolchain
4. Smaller, fast executing tests are easier to diagnose and maintain
5. Automation should not simply map manual test cases into automated test cases
6. Frequent test execution and analysis
7. BDD approach to test definition as appropriate
8. Test Results are transparent and accessible
9. Appropriately leverage emulators, physical devices, and device cloud solutions for mobile
10. Application architecture should keep testability in mind



Automation is aligned to the Pyramid

The Automation Test Pyramid forwards an opinion on the relative abundance of different automation types



End-to-End Tests
Exercise the full application environment

- Harder to Automate
- Expensive to maintain
- Slow
- Flakey
- Representative of User Experience

API / Boundary / Integration Tests
Exercise pieces of the application

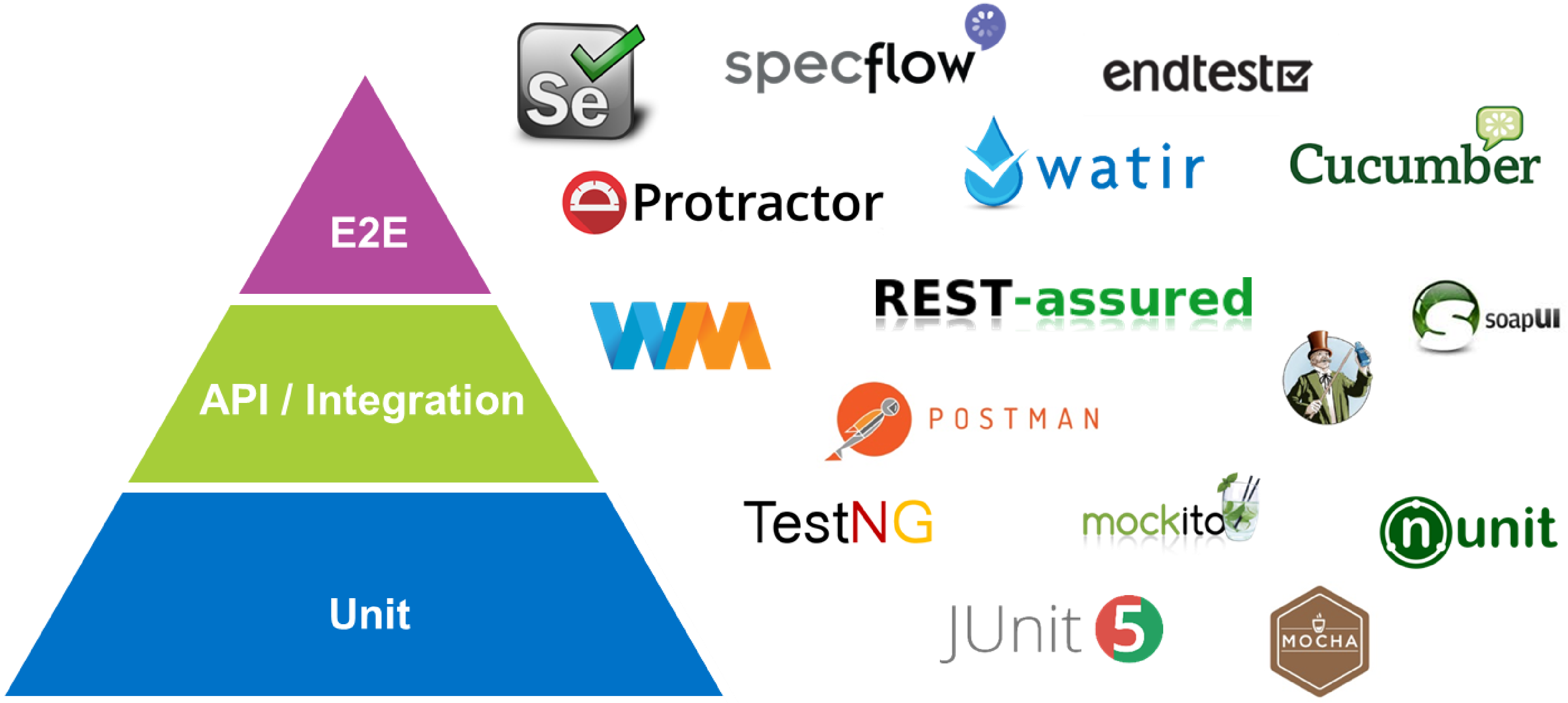
- Easier to Automate
- Faster
- Cheaper
- Less Flaky

Unit Tests
Exercise small chunks of code

- Easiest to Automate
- Fastest
- Cheapest
- Foundation of Pyramid
- Cannot prove system behavior

Automation Tool Selection and Integration is Key

The spectrum of automation frameworks and tools is ever expanding. Usually a combination of technologies, smartly architected and integrated together, is required to adequately address the full automation pyramid,



What is an Automation Framework?

An automation framework is not the same thing as a suite of automated scripts. Automation within a sophisticated framework is what enables reusable, scalable, and maintainable test automation

A collection of tools and libraries, **combined into a cohesive solution,** to support the automation requirements of a project, team, or company.

Automation frameworks empower engineers to **easily add or extend automated tests.**

Example Framework

Gradle
Build and dependency mgmt

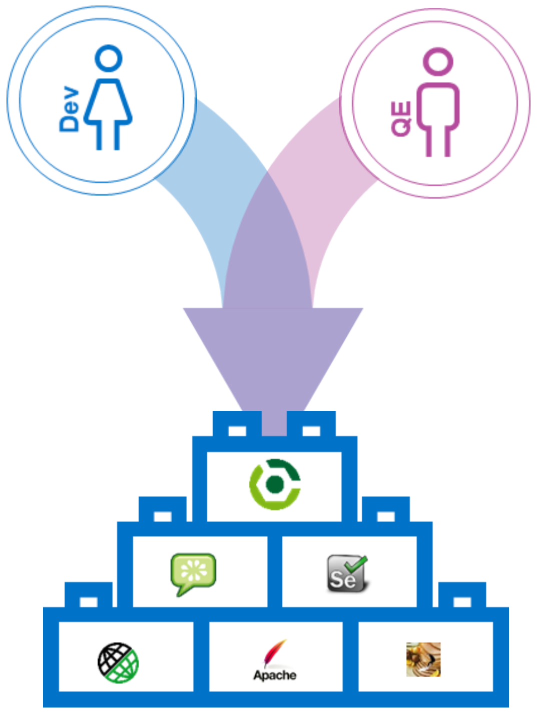
Cucumber-JVM
BDD framework

Selenium WebDriver
Browser API

RestAssured
REST API client library

Apache CommonConfig
Configuration mgmt

Hamcrest
Fluent assertions



Choosing test Cases best for Automation

1

Mission Critical tests that covers functionality that must absolutely work

- Test cases that are repetitive
- Error prone when executed manually,
- Relatively easy to automate
- With no dependency to other systems.

2

Test cases that don't hit functional areas with constant updates

- Areas of the application with refactor or active development.

3

Where there is maximum potential ROI

- What test cases whose failure may cause the most damage for a defect found in production.
- Cross-platform testing with multiple field validations

4

Test cases that need to executed with

- Different sets of Data
- Different browsers
- Different Environments
- Different set of users

What is difficult to automate

Higher costs in terms of time and effort to automate

- Mixed Technology Tests
- Dynamic Content
  - Unknow starting state
- Waiting for events
- Handling alerts/ pop ups
- Complex workflows
- Certain aspects of web application
  - Ex: Recognizing UI elements with dynamic ID's
  - Switching between multiple windows and automating iFrames

Inserting test automation into agile delivery

You have the tool, buy in from the team and a strategy in place...What's next??

Identify E to E Scenarios

- Audit your application and identify end to end workflows

Break it down by priority

- Categorize these E to E tests and group them into buckets based on priority.

Automate Smoke test

- Automate the "sanity" check tests
- Integrate and execute them with CI/ CD

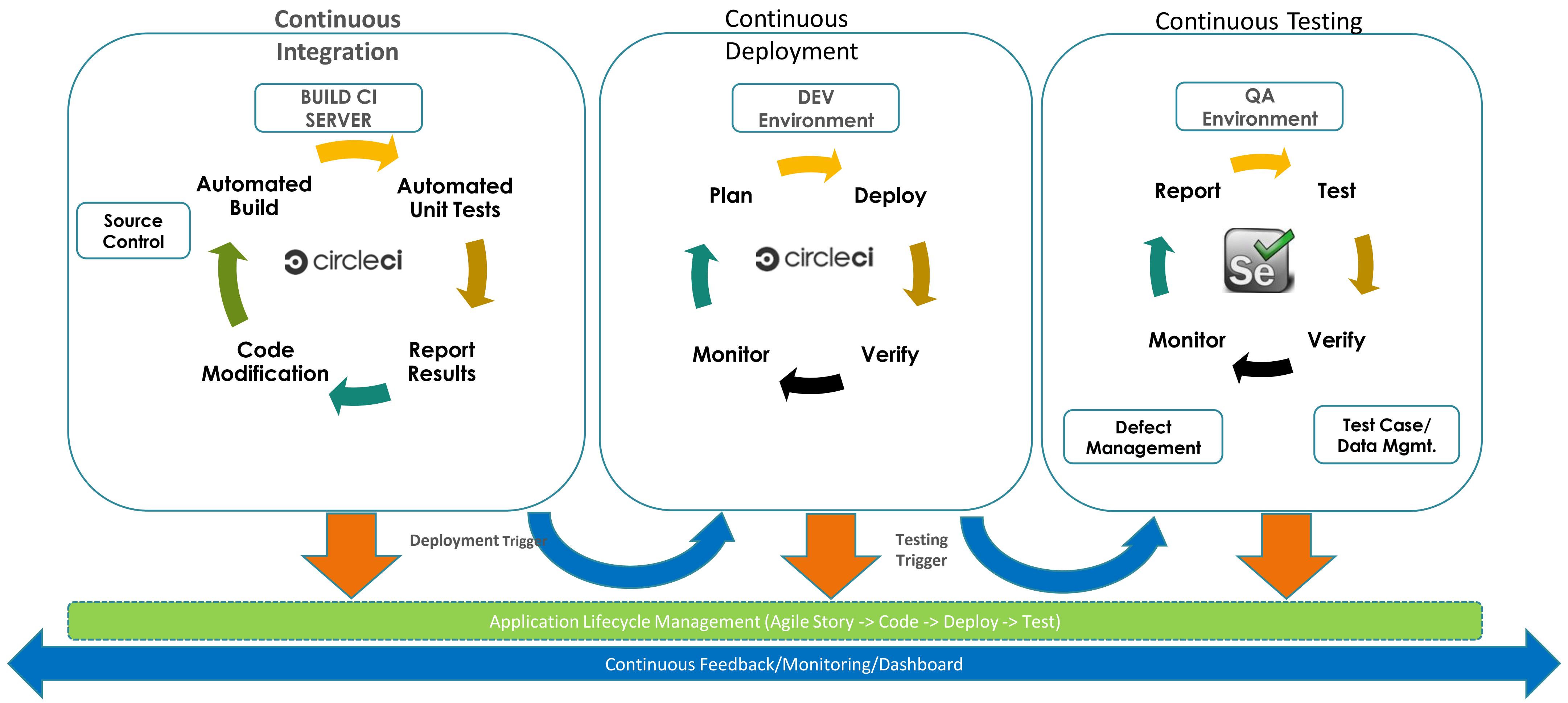
Automate priority 1 tests

- Can be run "on-demand"
- Have a policy in place to review results.

Audit and iterate as needed

- Continually "sell" the benefits of automation
- Never standstill with your automation. Refactor, reorganize as you see fit.

CI + CD = CT (CONTINUOUS TESTING)



CONTACT

- **Email:**
- **LinkedIn:**