



What DevOps Means for Testers and Testing

Jeffery Payne | Chief Executive Officer | jeff.payne@coveros.com

Jeffery Payne

Jeffery Payne is CEO and founder of Coveros, Inc., a company that helps organizations accelerate software delivery using agile methods. Prior to founding Coveros, he was the co-founder of application security company Cigital, where he served as CEO for 16 years.

Jeffery is a recognized software expert and popular keynote speaker at both business and technology conferences on a variety of software quality, security, DevOps, and agile topics. He has testified in front of congress on issues such as digital rights mgmt., software quality, and software research.

Jeffery is the technical editor of the AgileConnection community (www.agileconnection.com)





WHO WE ARE

Coveros helps organizations accelerate the delivery of secure, reliable software.

We provide consulting, coaching, and learning opportunities to enterprises, teams, and individuals.

Our Products + Services

Consulting Services

Agile software development
Agile testing and automation
DevSecOps engineering
Application security

Immersive Learning

50+ hands-on training classes
taught in both public and private
settings
Enterprise, team, individual
coaching to reinforce concepts
taught in class

Community Support

Software Conferences

STAR
EAST

STAR
WEST

Agile +
DevOps
WEST

Agile +
DevOps
EAST

Software Communities

AGILECONNECTION™

STICKYMINDS™

TECHWELLHUB
A SLACK COMMUNITY

Coveros customers receive discounted or free conference passes based on volume of business

Agenda

What IS DevOps?

- Definitions
- Problems it solves
- Measuring progress
- Terminology
- DevOps vs. CI/CD
- Value streams

Agile Testing

- Agile Testing Quadrants
- Agile Test Auto Pyramid
- Automation vs. Manual
- Automation stats

Test Auto in DevOps

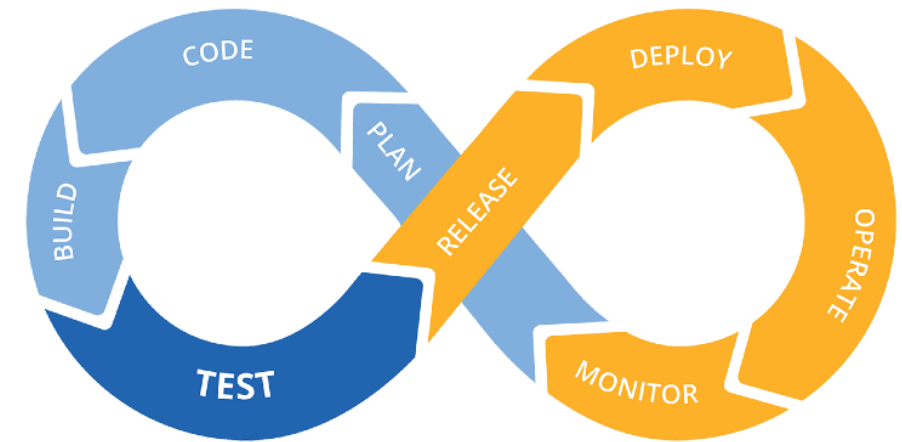
- Testing focus during DevOps
- Quality gates
- Balancing act
- Effective automated tests
- Automation approaches

What IS DevOps?

DevOps is a software development method that stresses communication, collaboration and integration between everyone in the software lifecycle

DevOps extends the principles of Agile into Operations

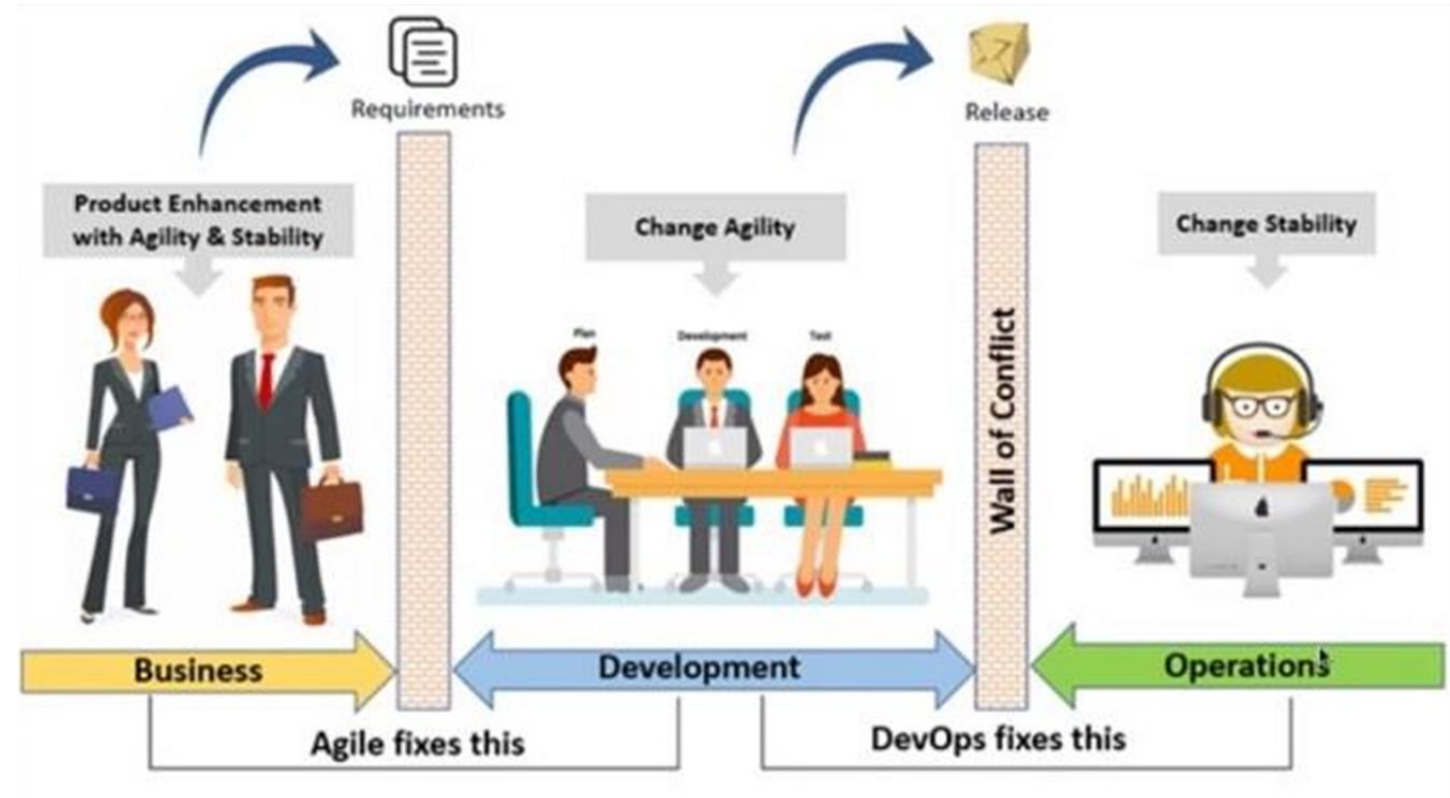
DevOps is NOT a tool or a particular process



Why DevOps? Wall of Conflict

Agile seeks to align customer needs with what is implemented

DevOps seeks to align continuous change with stability



DevOps Solves Common Problems

Challenges

DevOps Solutions

It compiles, therefore it works!



Developer test what they build

Big bang integration



Continuous integration of changes

Late lifecycle QA and security analysis



Shift assurance activities left

“Well, it worked on MY machine”



Common platforms

Delays in getting environments stood up



Self-service and automated setup of environments

Midnight deployments that always fail



Continuous delivery/deployment

DevOps Terminology

Lean - a way of thinking about creating needed value with fewer resources and less waste

Value Stream Mapping - a Lean management method that allows you to visualize, analyze and improve all the steps in a product delivery process

Shift Left –take a task that's traditionally done at a later stage of the software process and perform that task at earlier steps

Shift Right –the practice of performing testing, quality, and performance evaluation in production under real-world conditions

DevOps Terminology

Continuous Integration – integrate and test software on a regular basis to thwart downstream issues

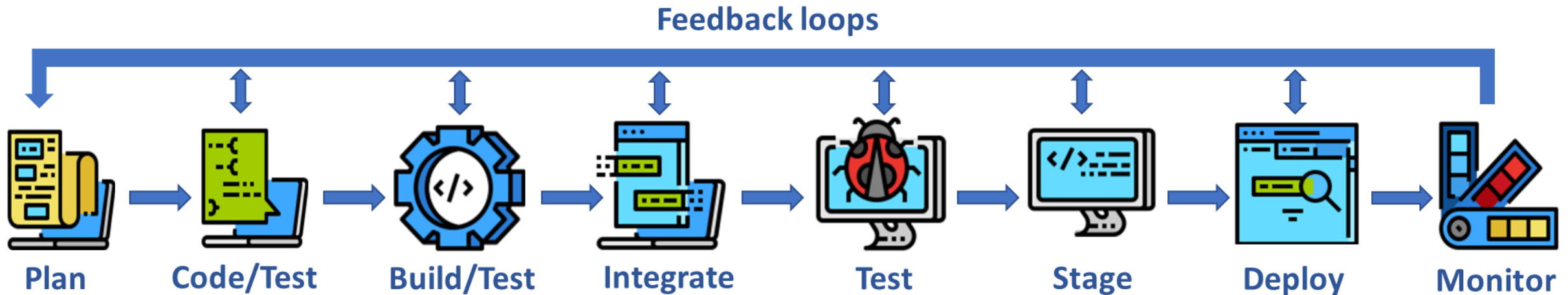
Continuous Delivery – practices to assure code can be rapidly and safely deployed to downstream testing and production environments

Continuous Deployment – the ability for code changes committed to a repository to be automatically tested and deployed into production without human intervention

Continuous Monitoring – monitoring software in product to thwart issues and collect usage information

DevOps Pipelines

A DevOps pipeline is a set of practices that the development (Dev) and operations (Ops) teams implement to build, test, and deploy software faster and easier. One of the primary purposes of a pipeline is to keep the software development process organized and focused.



DevOps vs. CI/CD

DevOps

All about effective communication, collaboration within an end-to-end software development lifecycle

Focuses on processes, culture, roles, change, and organizational structure

Business oriented metrics

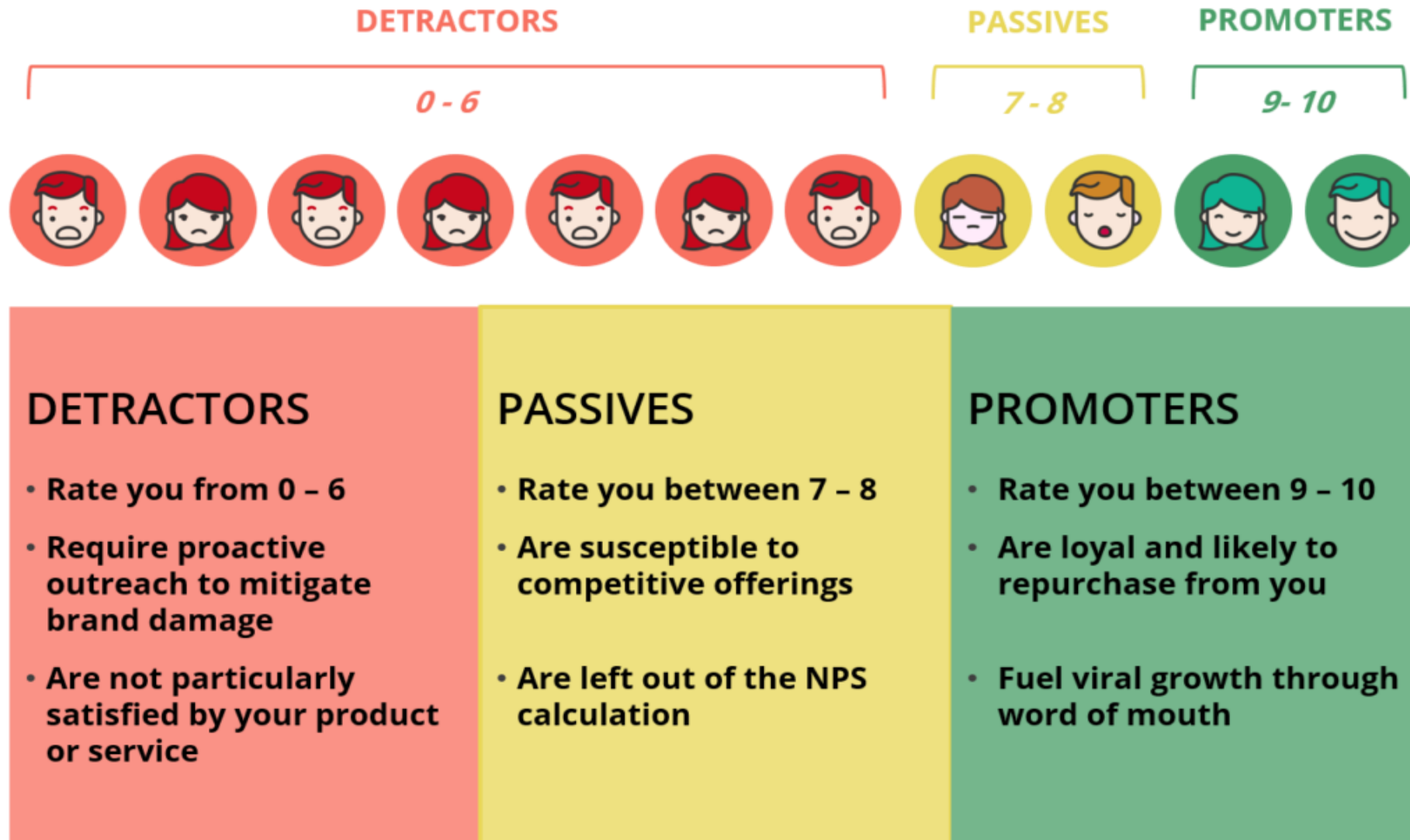
CI/CD

All about how to effectively automate the process to accelerate delivery and customer value





Focuses on tooling to build, test, integrate, secure, deliver, deploy, and monitor software applications

Technology oriented metrics

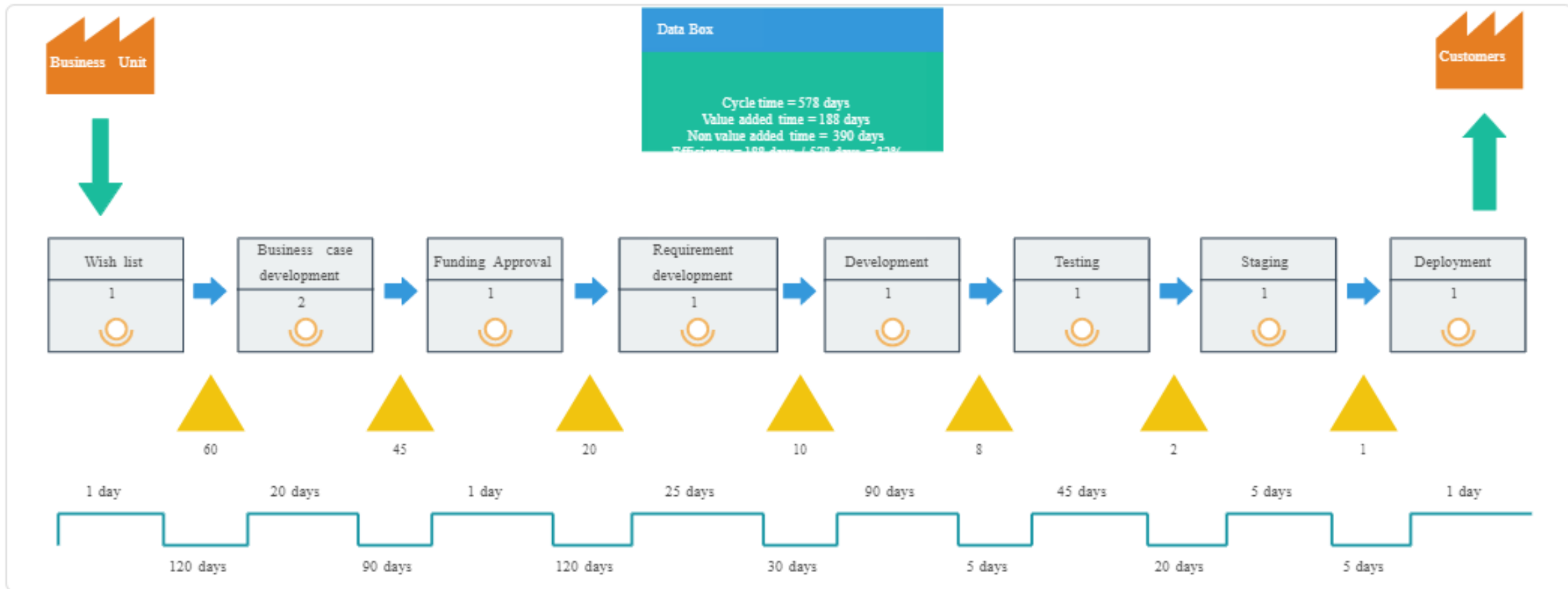
Measuring Customer Value



Common Delivery Metrics

Software delivery performance metric	Elite	High	Medium	Low
 Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
 Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
 Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
 Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Value Streams and Process Improvement



DevOps vs. DevSecOps

DevSecOps means different things to different people

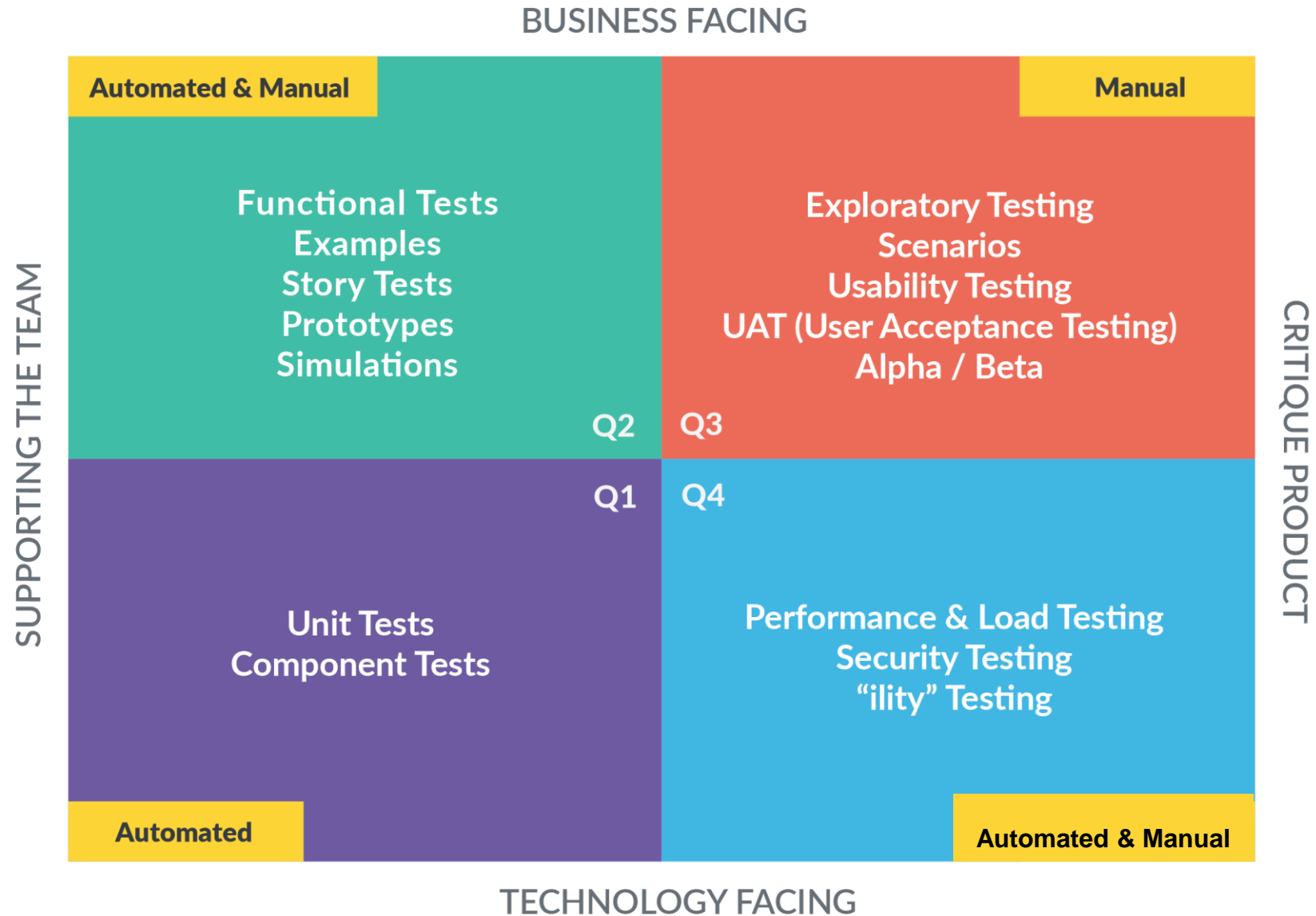
Regardless our goal is to shift security activities left

Build security in

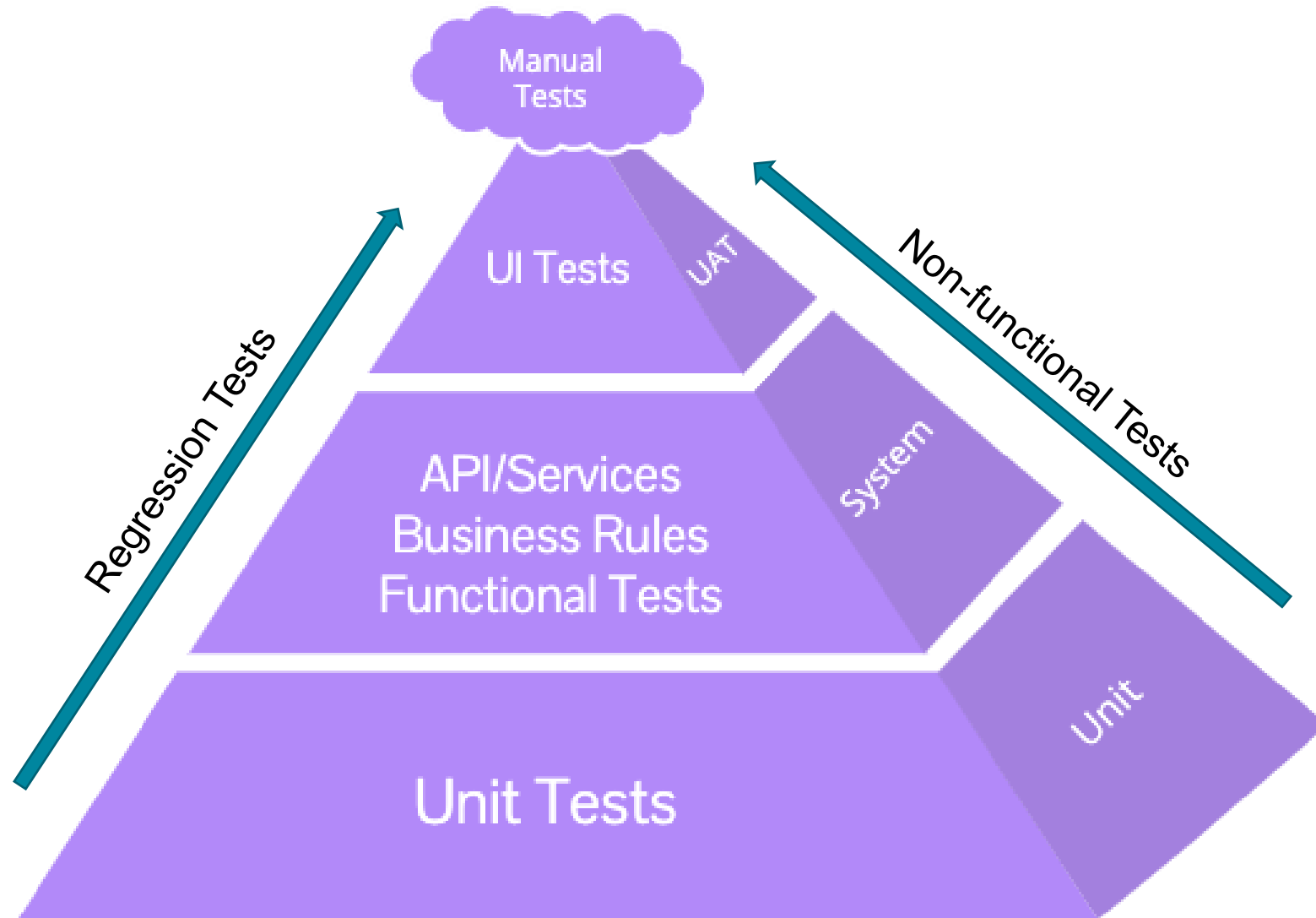
Test security earlier (often with tools)



Agile Testing



Agile Test Automation Pyramid



How Much Often Gets Automated?

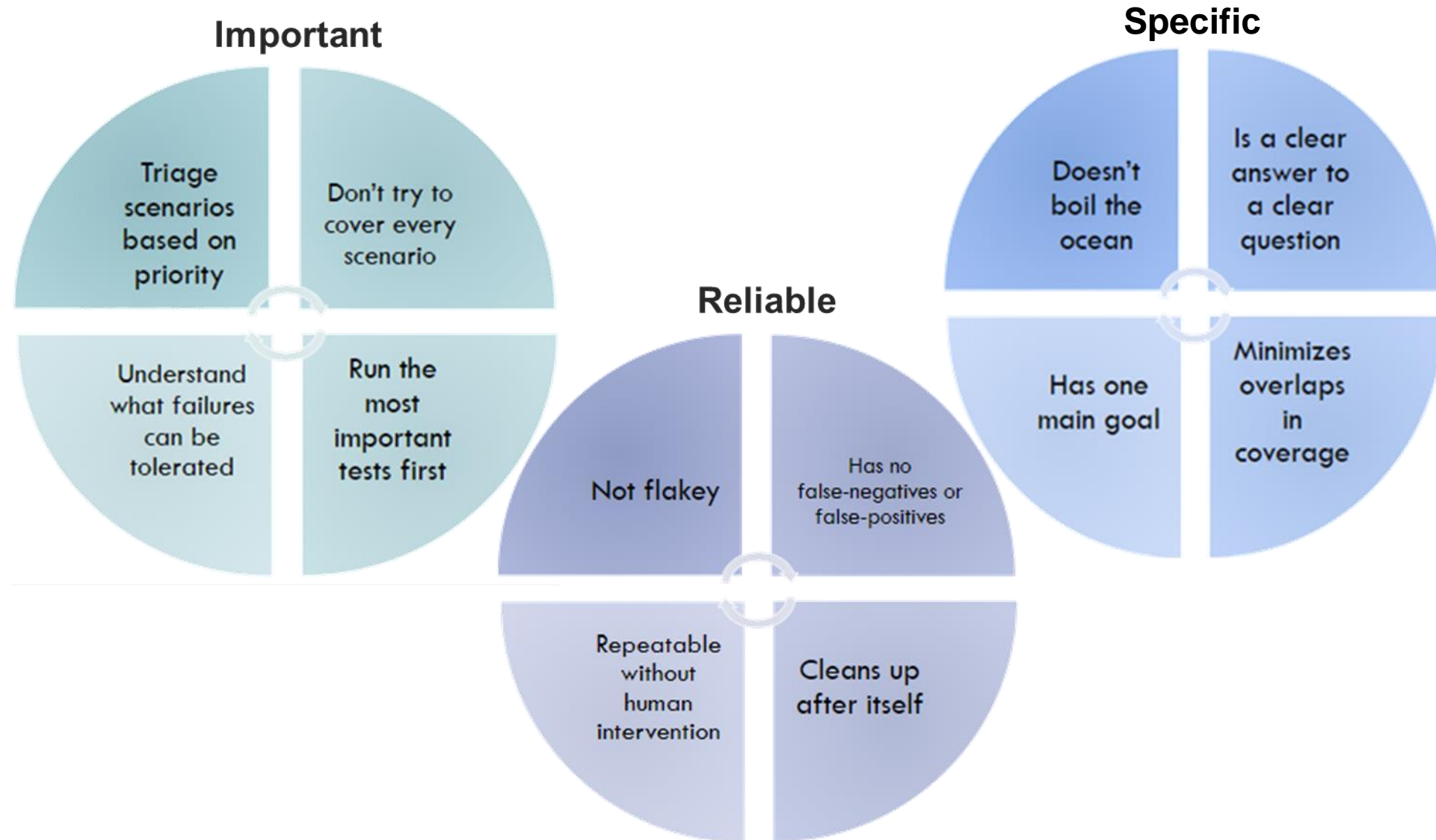
Studies show that automation is a sound investment

If the right things are automated, it frees up dev and test to do more important activities

But there is effort necessary to maintain automation and if the wrong things are automated, that effort can be substantial

	Low	Medium	High	Elite
Automated build	64%	81%	91%	92%
Automated unit tests	57%	66%	84%	87%
Automated acceptance tests	28%	38%	48%	58%
Automated performance tests	18%	23%	18%	28%
Automated security tests	15%	28%	25%	31%
Automated provisioning and deployment to testing environments	39%	54%	68%	72%
Automated deployment to production	17%	38%	60%	69%
Integration with chatbots / Slack	29%	33%	24%	69%
Integration with production monitoring and observability tools	13%	23%	41%	57%
None of the above	9%	14%	5%	4%

What Makes a Good DevOps Pipeline Test?

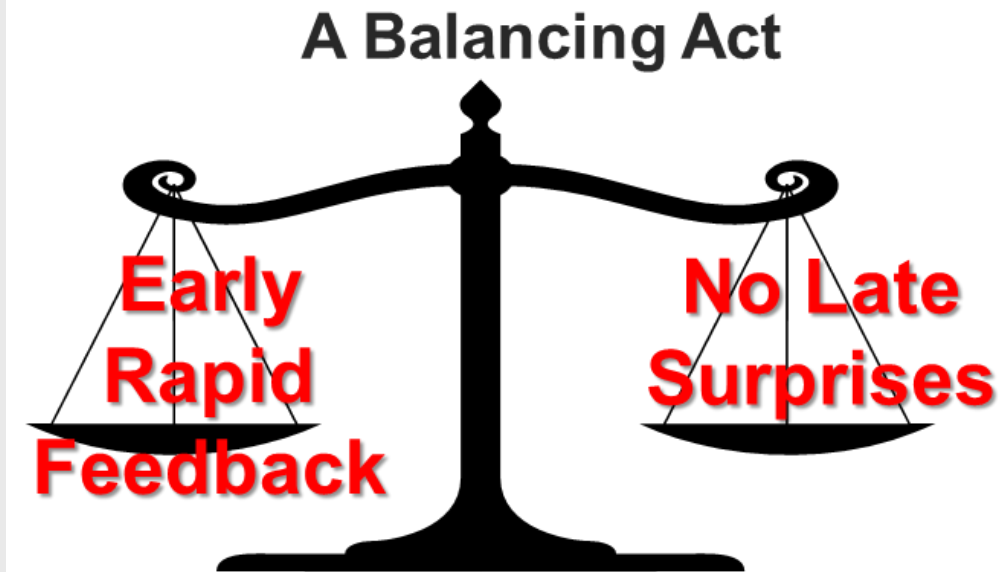


DevOps Testing Balancing Act

We want rapid early feedback so we *fail fast* (or learn quickly).

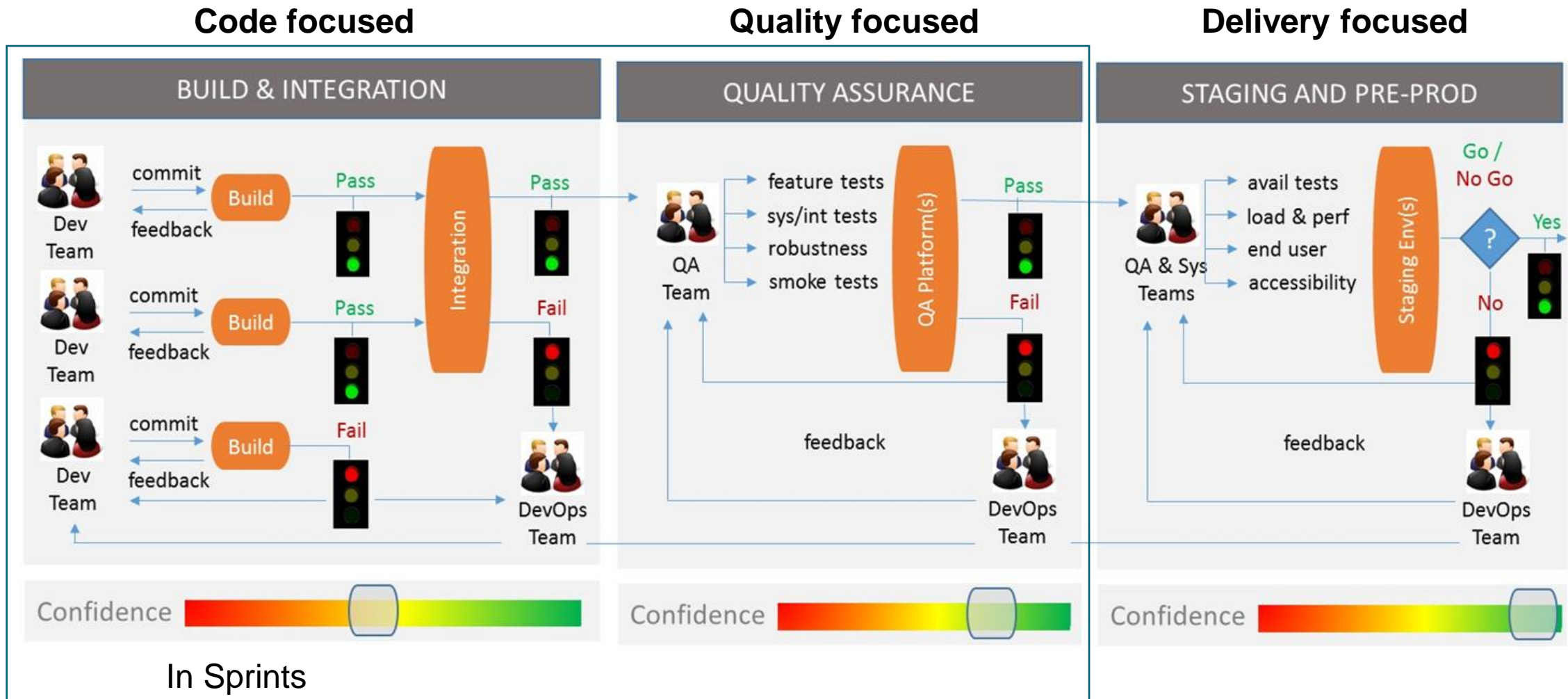
But if what we do early misses critical quality issues, we get late surprises that slow down deployment (or worse).

This balancing act is the key to creating a successful DevOps testing process



Where Testing is Often Performed

Testing here is:



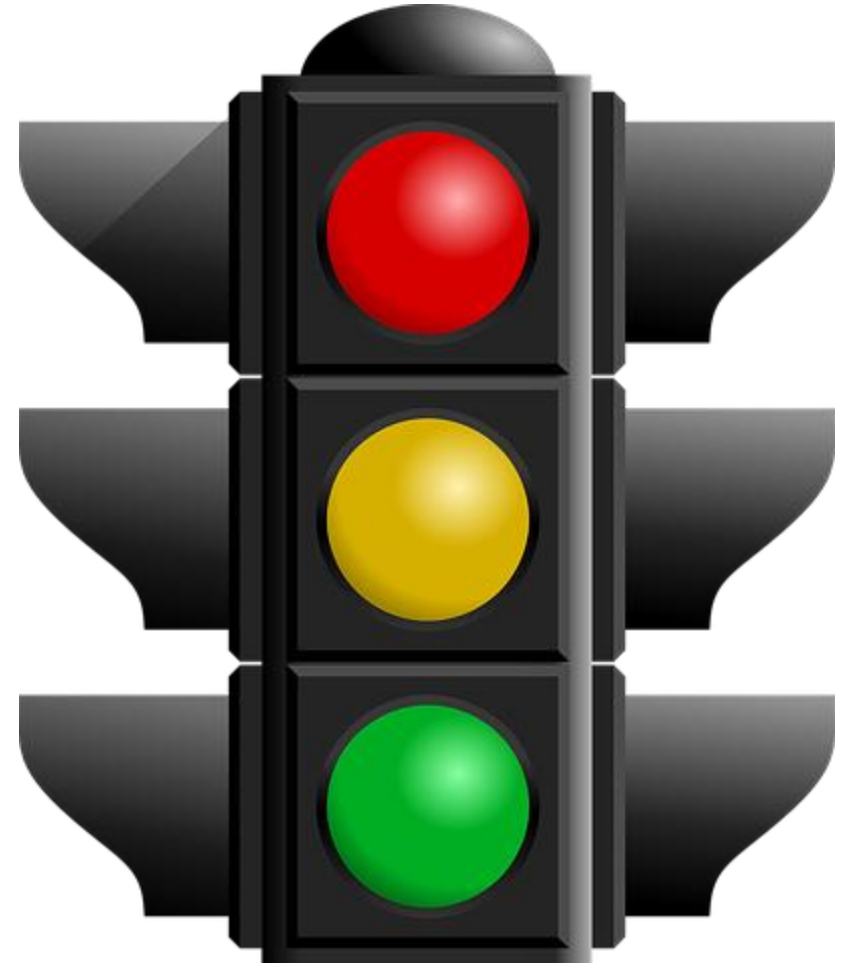
Quality Gates

Quality gates determine whether testing has been successful enough for a change to move forward in a DevOps Pipeline

Common quality gates for testing activities:

- Unit testing ➡ code coverage
- User story tests ➡ story acceptance criteria
- API tests ➡ coverage of API spec
- Integration tests ➡ coverage of interface pts
- System tests ➡ workflows
- Smoke tests ➡ happy paths work

Don't forget that blockers and critical defects are often gates too!



Code Focused Test Automation

Static Code Analysis

Evaluates the quality and security of your code using static analysis

Ideally, run first by developers in their IDE / in their development env

Reasons to also run during CI:

- Trust but verify
- Gather trending data
- Address blockers & gate

Unit Testing

Tests the smallest testable part of an application individually and independently

100% automated (lives in the code)

Fast enough to run all unit tests during CI

Code coverage is often a gate

The feedback loop for your continuous integration process should not take more than 10-15 minutes

Establishing CI in 4 (not so easy) Steps

Get your developers and testers in the habit of writing appropriate tests before and during code development.

Setup a CI server that automatically builds and runs defined test suites each time a code change is pushed to your main repository (trunk)

Make sure changes are pushed to the trunk (or short-lived branch) daily at a minimum

Fix the build as soon as it is broken

There is no one size fits all CI process you should follow

Quality Focused Test Automation

Functional Testing

Testing user stories against their acceptance criteria

Integration testing of features, stories, components, services

End to end workflow tests (aka acceptance tests)

Smoke testing

Non-functional Testing

Types of NFT depend on the quality attributes that are important for your application and market

- Load, Performance, Stress
- Security
- Availability
- Reliability

Some NFT must be performed on production-like platforms

Shift left any testing defined here that can be performed during CI and its into your CI feedback cycle

Optimizing Quality Focused Test Auto

Identify and test at testable interfaces below the user interface as much as possible!

Testable interfaces in modern software applications

- Services (for service-oriented and microservice architectures)
- Application tier APIs (for n-tier architectures)

Run some amount of automated regression tests at least once per day

Parallelize automation if using the cloud or have test environment capacity

There is no one size fits all CI process you should follow

Where Does Regression Testing Fit?

Fit automated regression testing wherever your feedback cycles allow it

Automated regression testing at some level should be performed at least daily to align with CI/CD goals

Some organizations are parallelizing their large automated regression test suites to run periodically throughout the day in the cloud

Split large automated regression test suites up to shift them left and/or run some amount of testing earlier / faster in your process

Regression testing is essential in Agile and DevOps as it support fearless refactoring and iterative development

Delivery Focused Test Automation

Delivery-focused testing is necessary testing we haven't been able to shift left

Often testing activities that:

- must be run on production-like platforms that are not accessible by Dev and Test
- support external compliance or regulatory needs
- internal acceptance testing to support a sign-off to go into production
- testing of monitoring capabilities and roll back/forward capabilities
- smoke testing in production to make sure our deployment works!

Most of these activities are semi-automated at best

Low Hanging Fruit for Automation

Unit tests (in the code)

User story acceptance tests (good tooling)

API tests (well specified and good tooling)

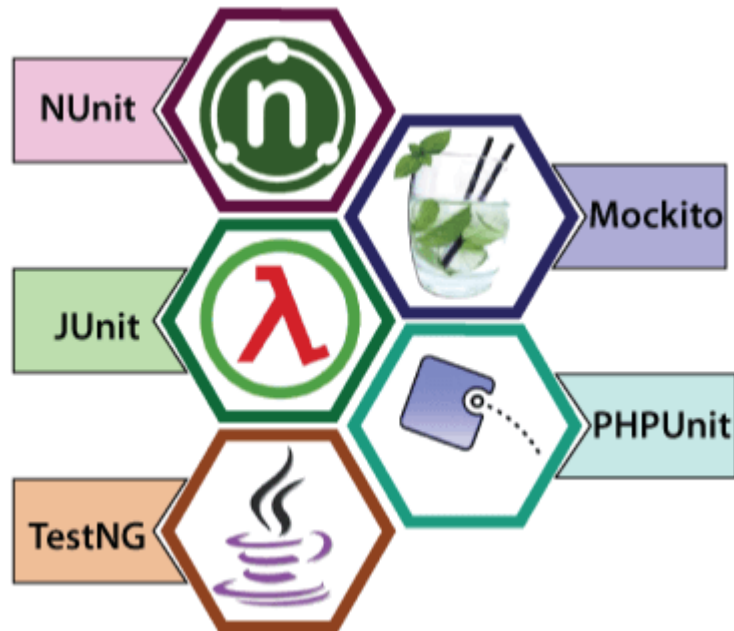
Smoke tests (simple)

Performance tests (you don't have a choice)

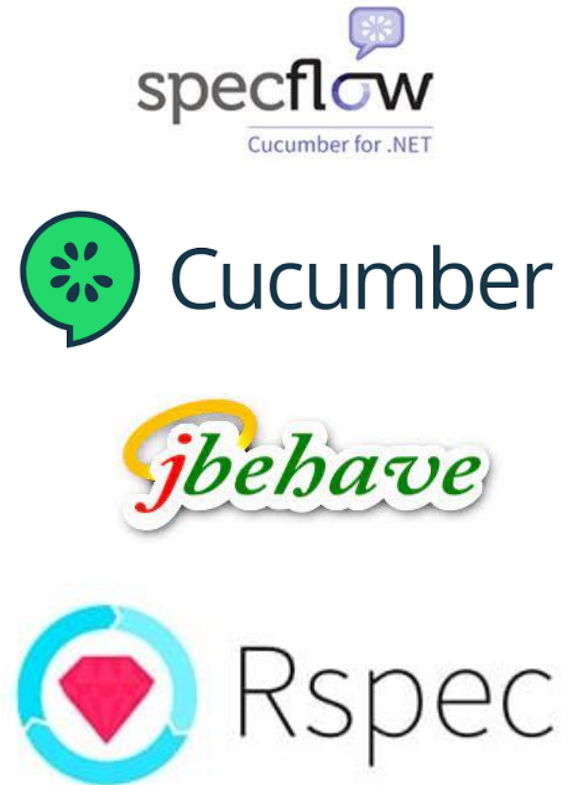


Sampling of Tools

Unit testing



User story acceptance



API testing



Options for Creating Automated Tests

Centralized automation team

Developers do all of the testings and automate as they go

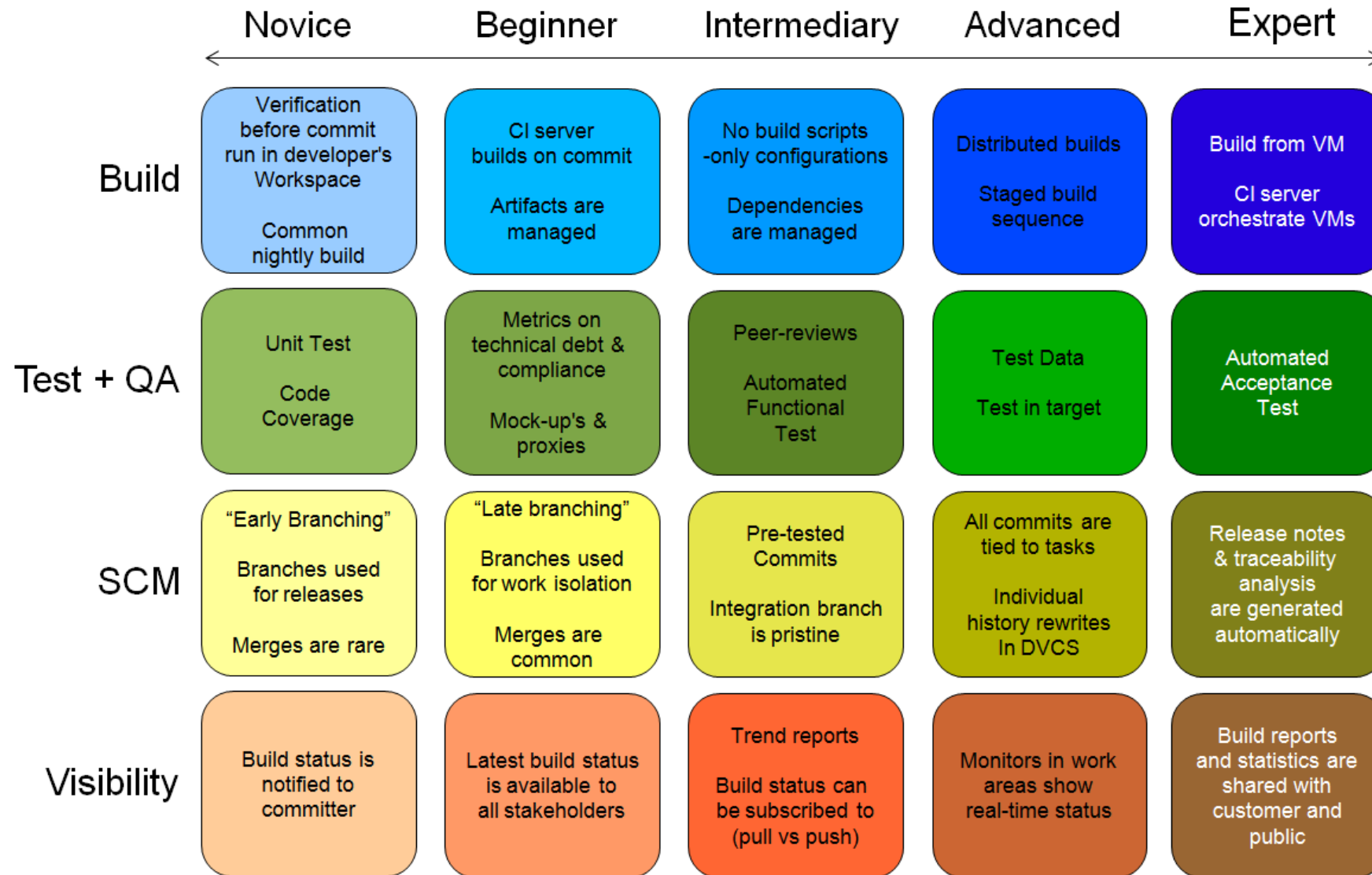
SDET model to support in Sprint automation

Hire test auto specialists for your teams

Retrain current manual testers



Continuous Delivery Maturity Model





QUESTIONS?

Thank you!

