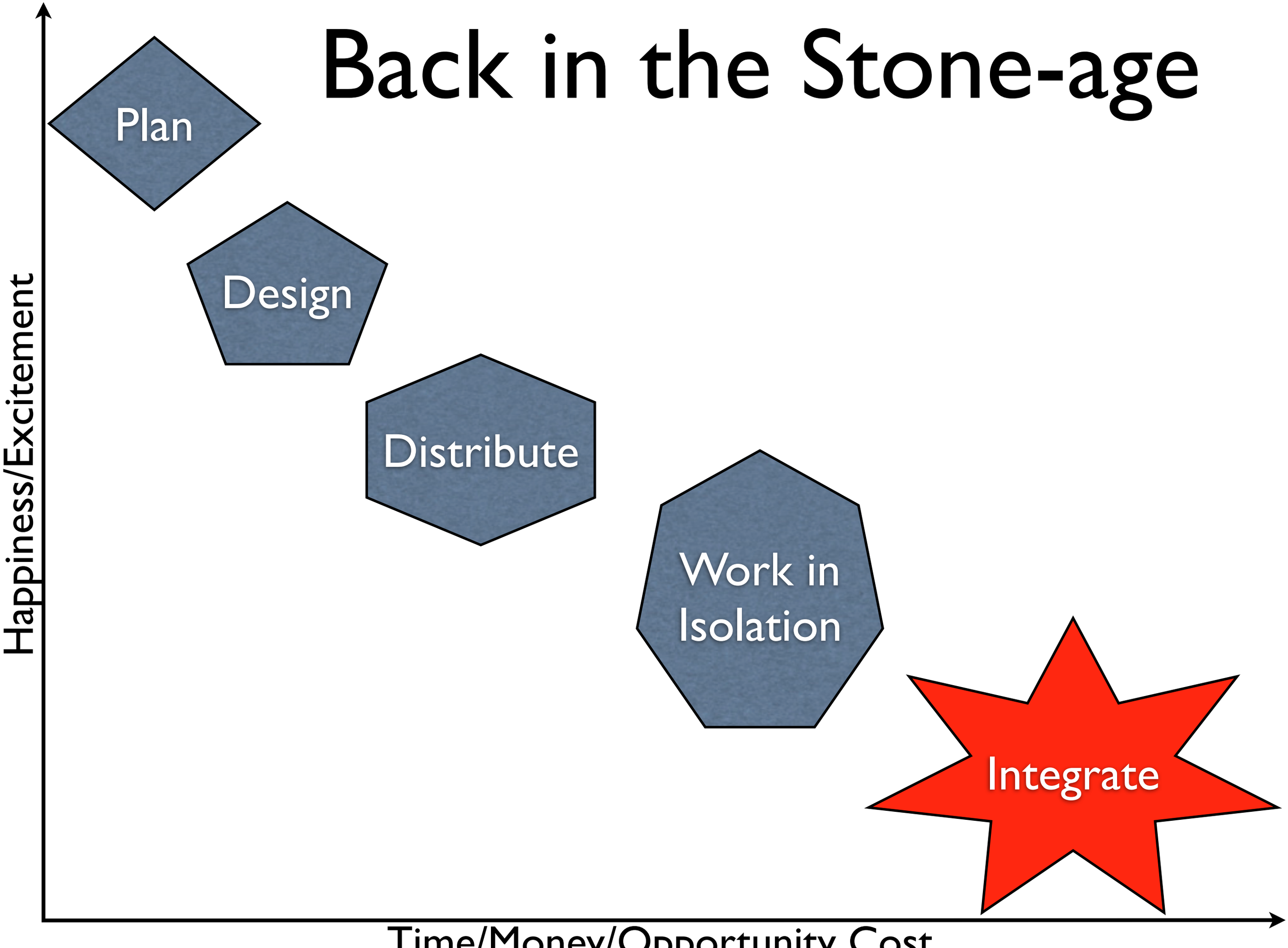
agile faqs

Inverting the Pyramid

Naresh Jain
naresh@agilefaqs.com
twitter @nashjain
http://nareshjain.com

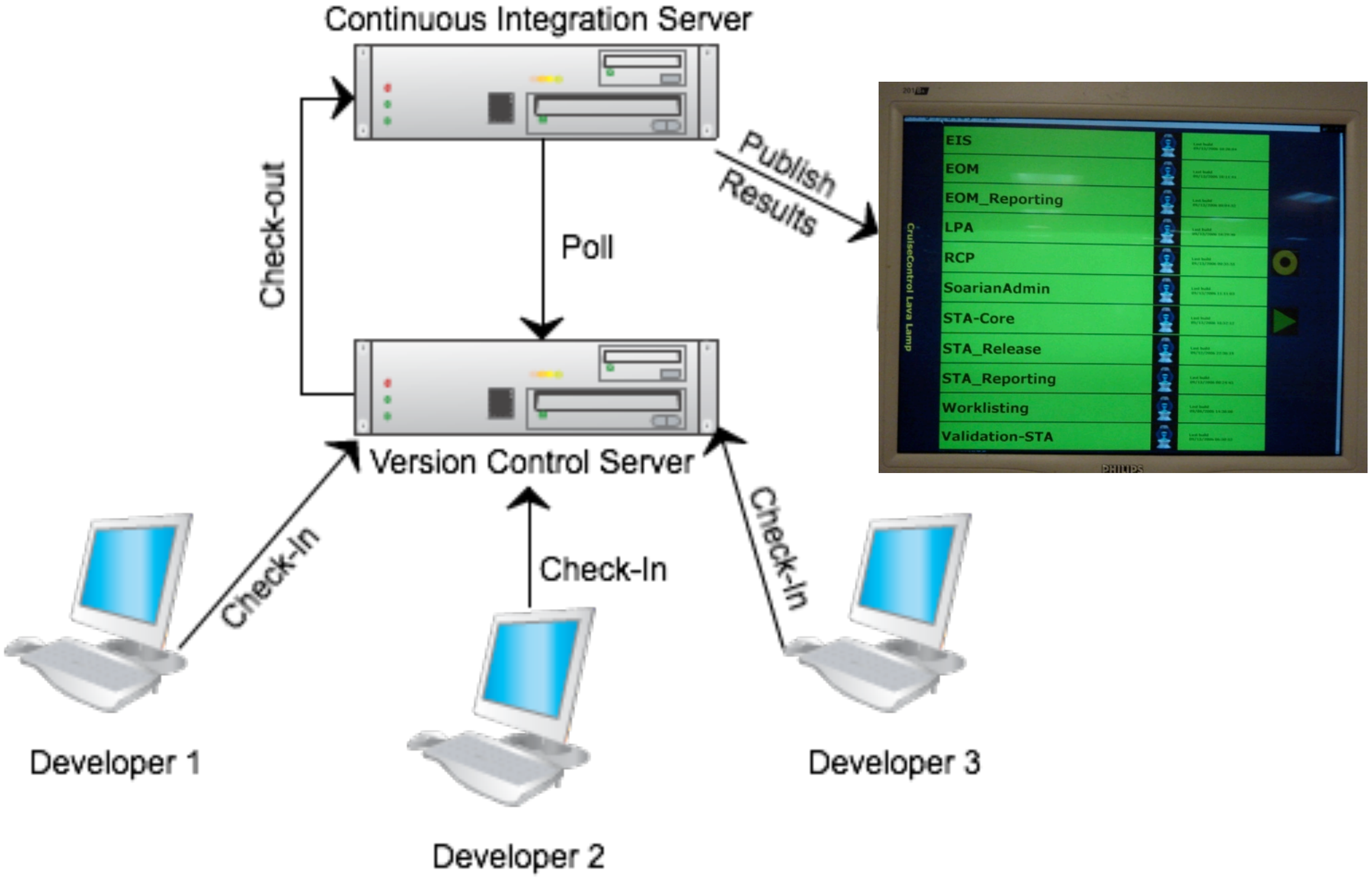
Back in the Stone-age



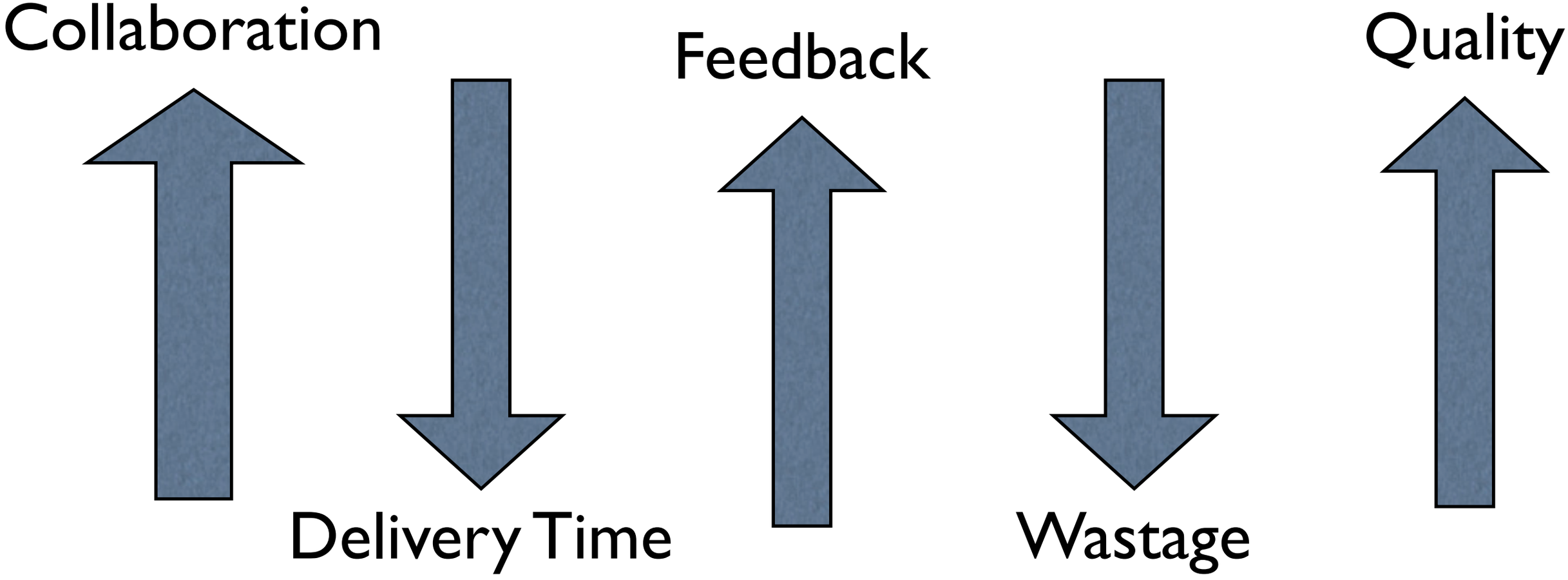
Last Minute Integration Surprises

BAD things were visible too Late...

Birth of CI


Continuous Integration Server
Check-out
Poll
Publish
Results
Version Control Server
Check-In
Check-In
Check-In
Developer 1
Developer 2
Developer 3
CruiseControl Lava Lamp
EIS
EOM
EOM_Reporting
LPA
RCP
SoarianAdmin
STA-Core
STA_Release
STA_Reporting
Worklisting
Validation-STA

CI Helped Us Learn That... Life can Suck a lot Less!

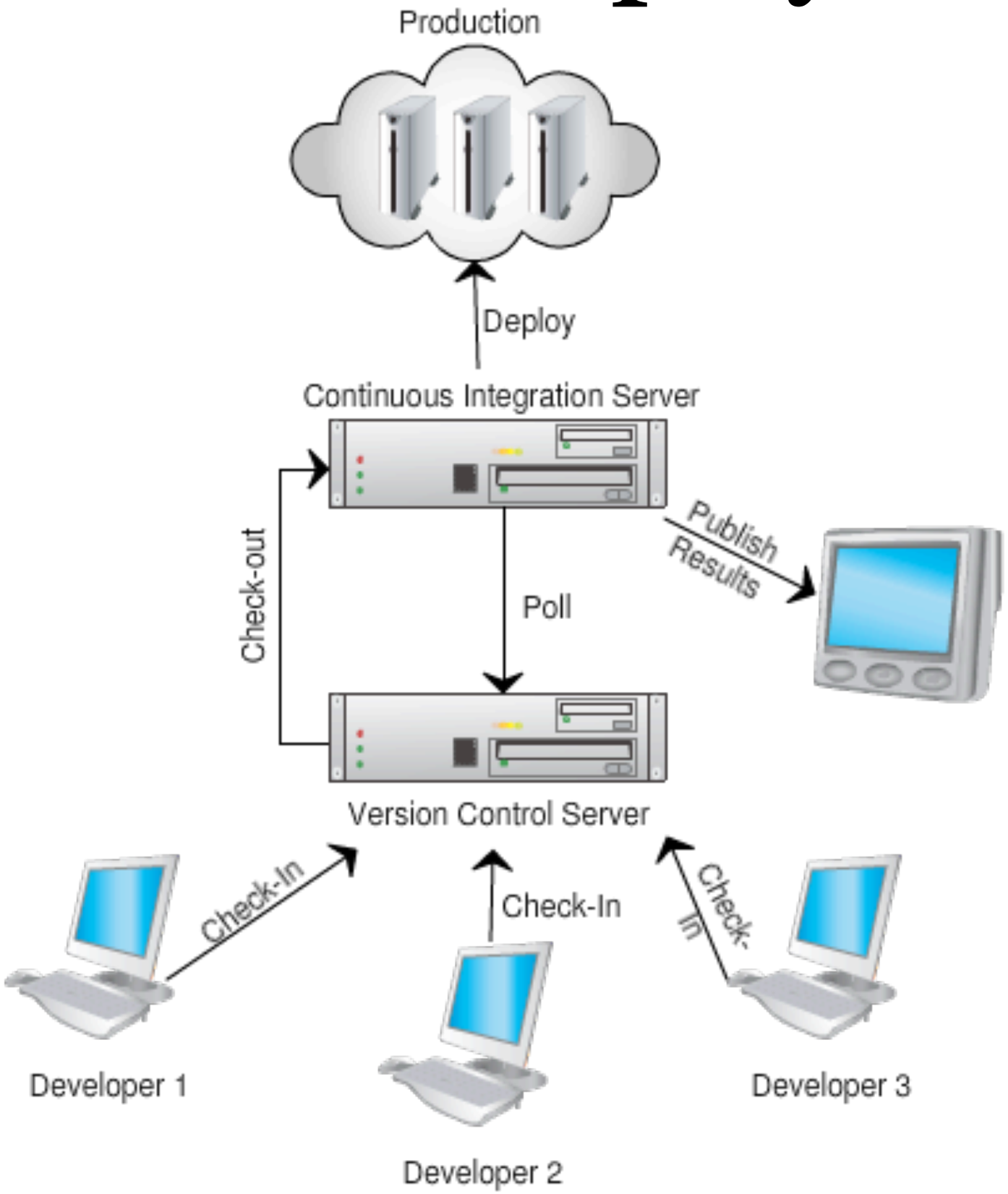


Lean-Start-up Community

Tried something quite disruptive...

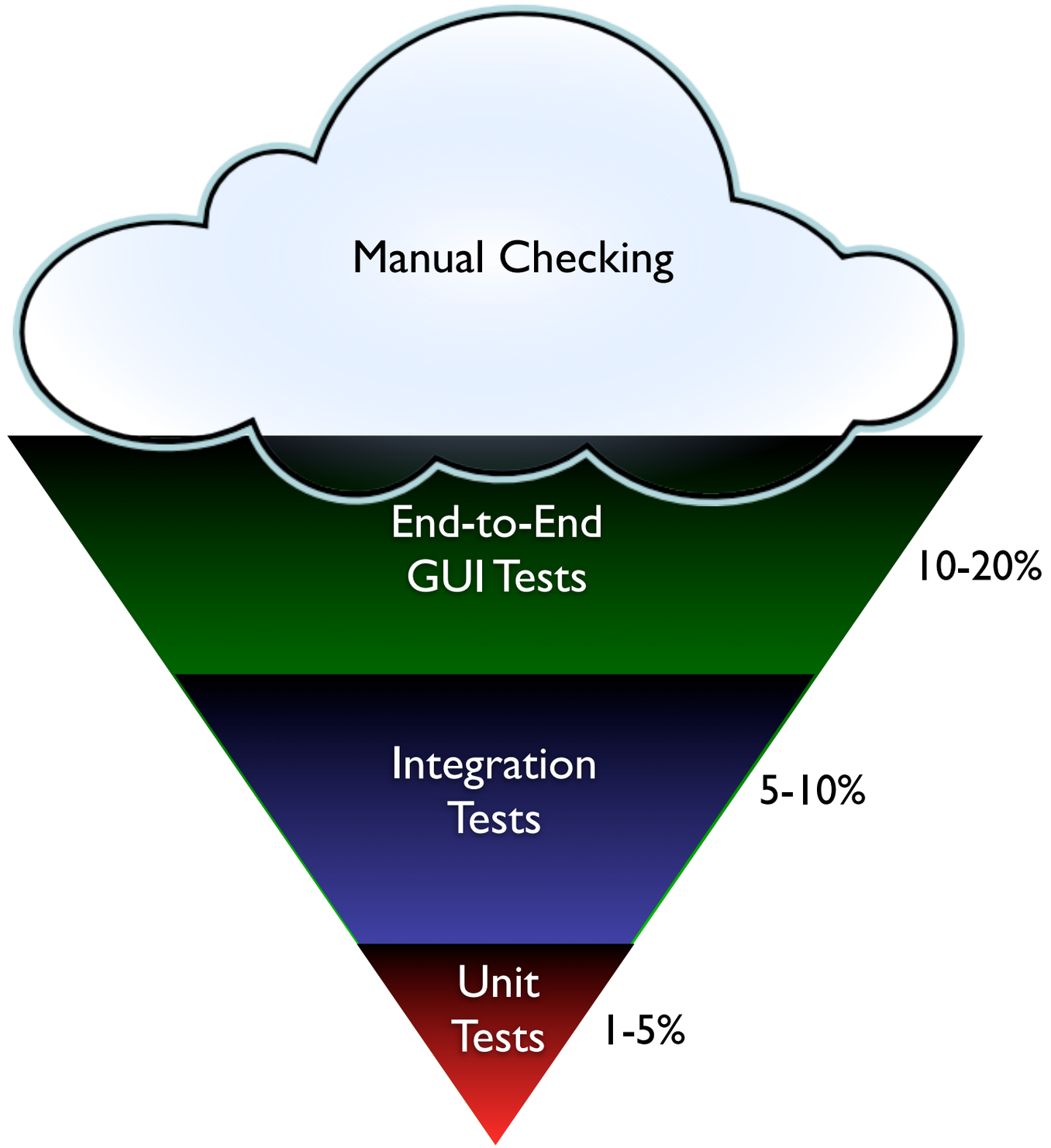
Continuous Deployment

agile faqs



But...

Software Testing Ice-cream Cone



Commercial Break!


ELEVATION 12,005 FEET
2.3 MILES
ABOVE SEA LEVEL

Mumbai

Simple Design and Testing Conference

AgileCoachCamp

Agile India 2005

Conference on XP and Agile Methodologies. 4-5 march 2005

ASCI

Agile Software Community of India

{ techcamp }

Agile Philly

CODECHEF BETA

Tech Talks!


IDEA
ENTREPRENEUR


ENTREPRENEUR
inside

hp

invent

BARCO

intuit

YAHOO!

RAVA

industrial logic

amazon

Schlumberger

redgate

ingeniously simple tools

EMC²

ThoughtWorks

RBS

The Royal Bank of Scotland

Google

SIEMENS

medical

DELL

ISRO

Indian Space Research Organisation

MISYS

Directi

CSC

Adobe

JOHN DEERE

iND

iN DEMAND

imagination at work

VOLVO

Testing VS. Checking

Testing is explorative, probing and learning oriented.

Checking is confirmative (verification and validation of what we already know). The outcome of a check is simply a pass or fail result; the outcome doesn't require human interpretation. Hence checking should be the first target for automation.

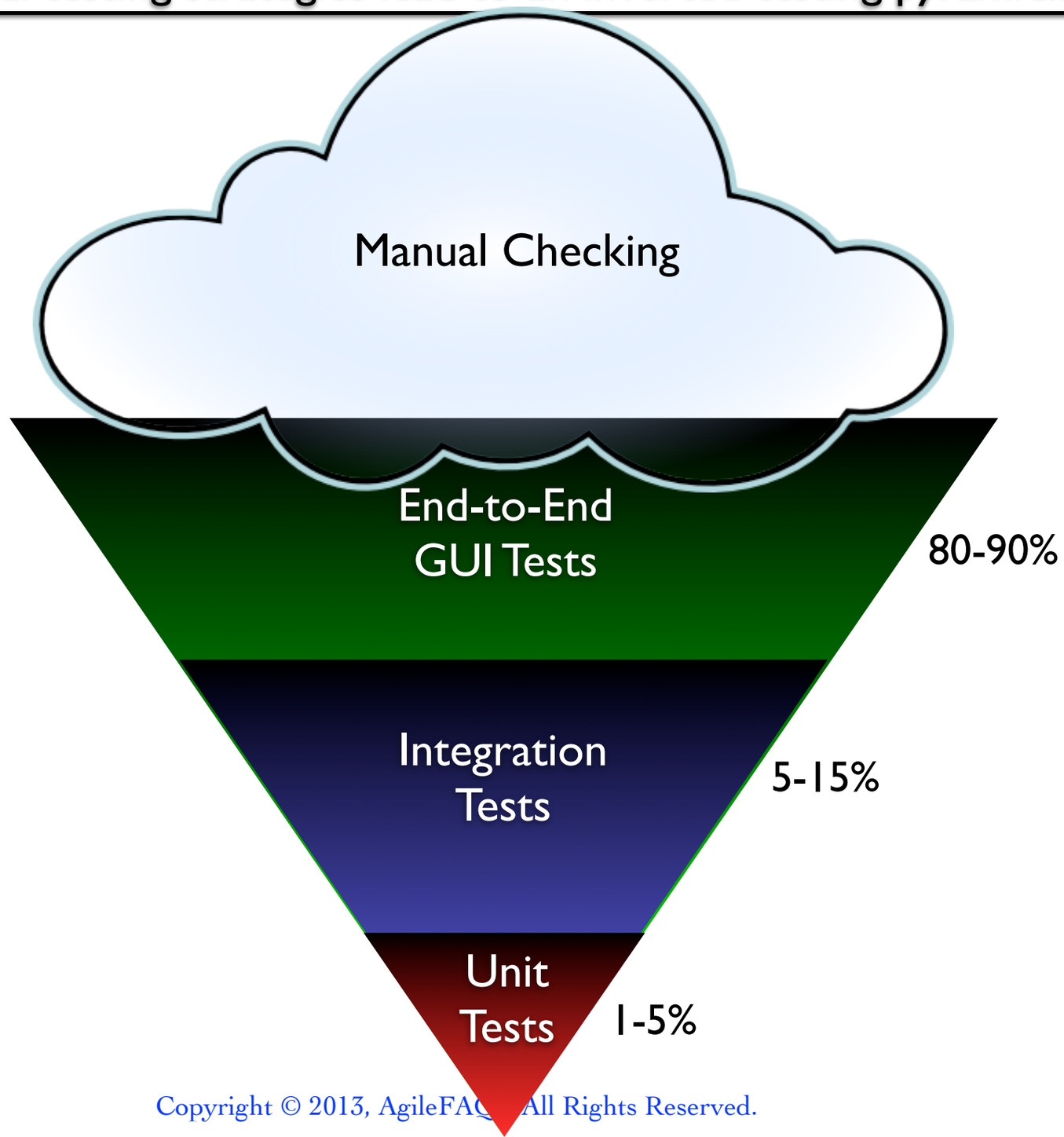
James Bach points out that checking does require some element of testing; to create a check requires an act of test design, and to act upon the result of a check requires test result interpretation and learning. But its important to distinguish between the two because when people refer to Testing they really mean Checking.

Why is this distinction important?

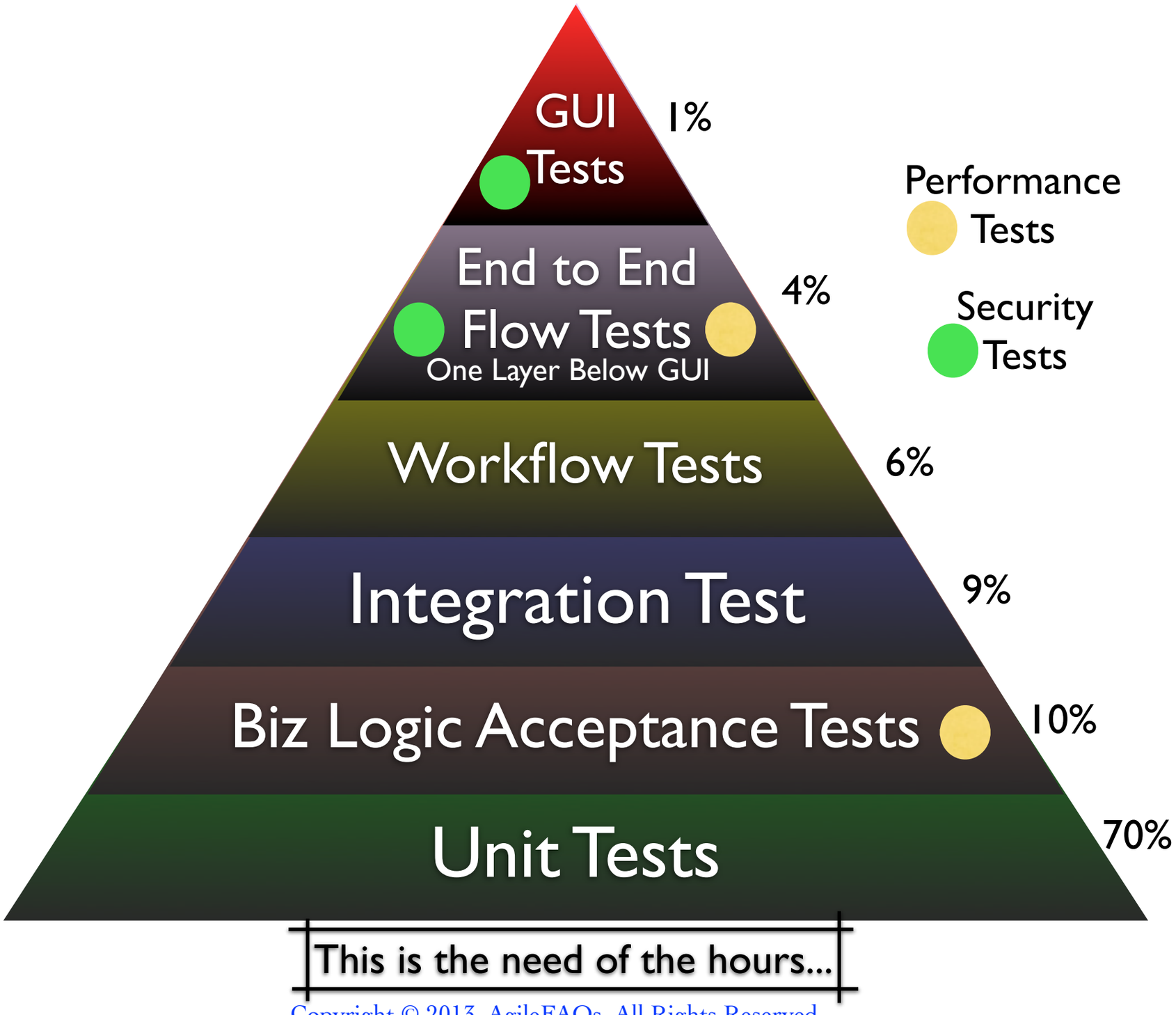
Michael Bolton explains it really well: "A development strategy that emphasizes checking at the expense of testing is one in which we'll emphasize confirmation of existing knowledge over discovery of new knowledge. That might be okay. A few checks might constitute sufficient testing for some purpose; no testing and no checking at all might even be sufficient for some purposes. But the more we emphasize checking over testing, and the less testing we do generally, the more we leave ourselves vulnerable to the Black Swan."

Inverting the Testing Pyramid

Typical testing strategies lead to an inverted testing pyramid...

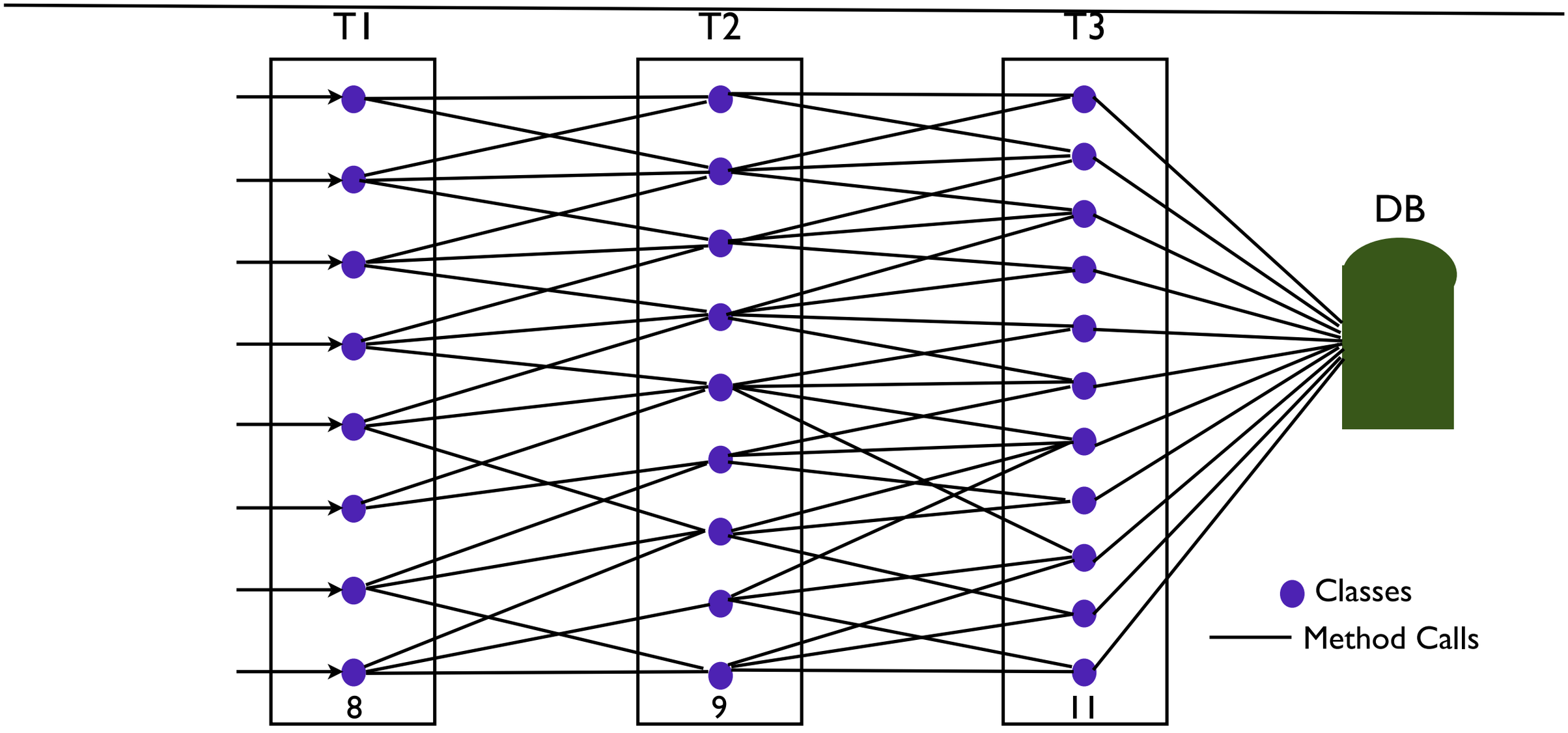


Inverting the Testing Pyramid



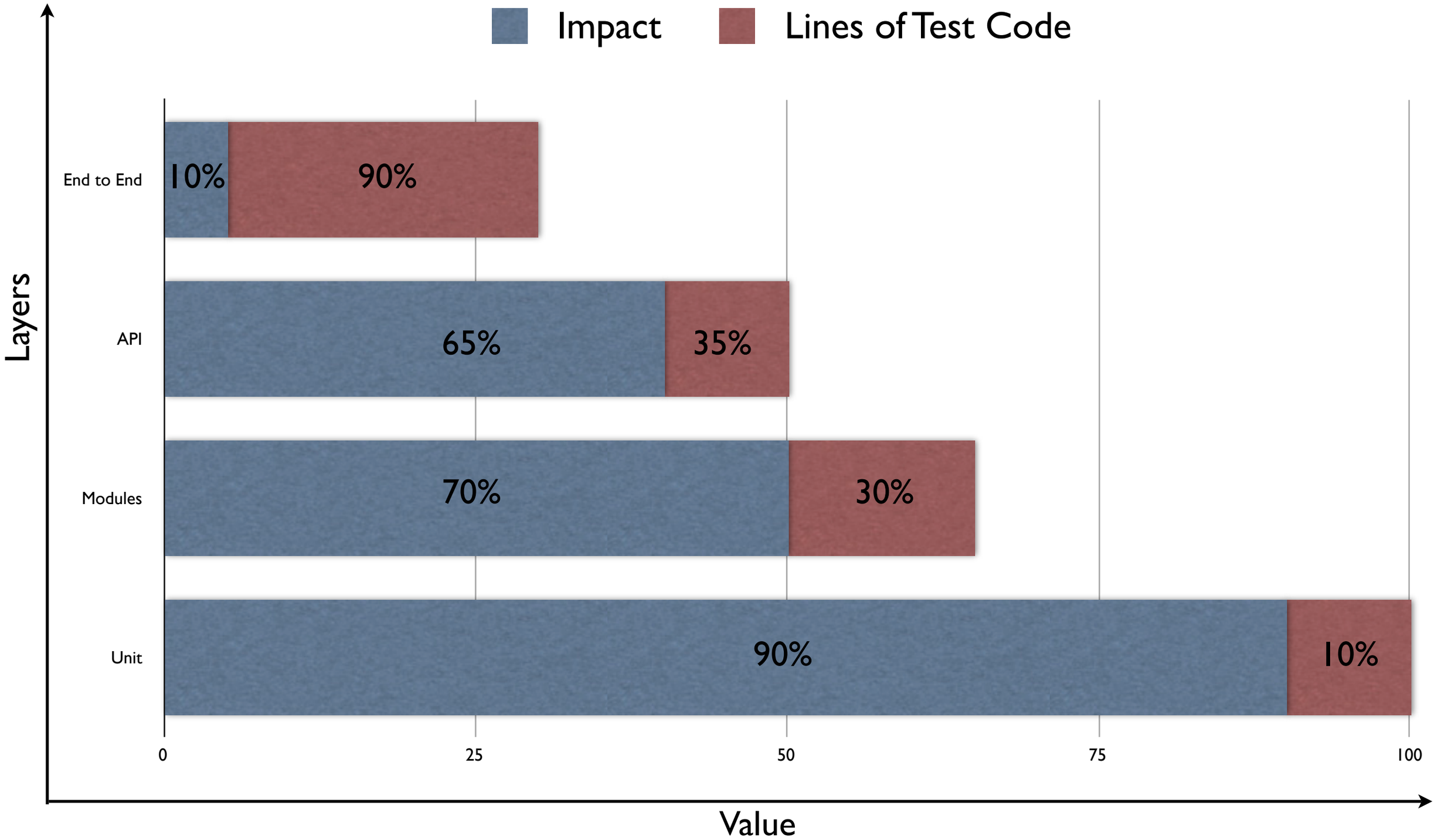

INVERTING THE
PYRAMID
THE HISTORY OF FOOTBALL TACTICS

Number of End-to-End Tests



70 Complex, Fragile End-to-End Tests
vs.
61 Unit tests + 8 End-to-End Tests

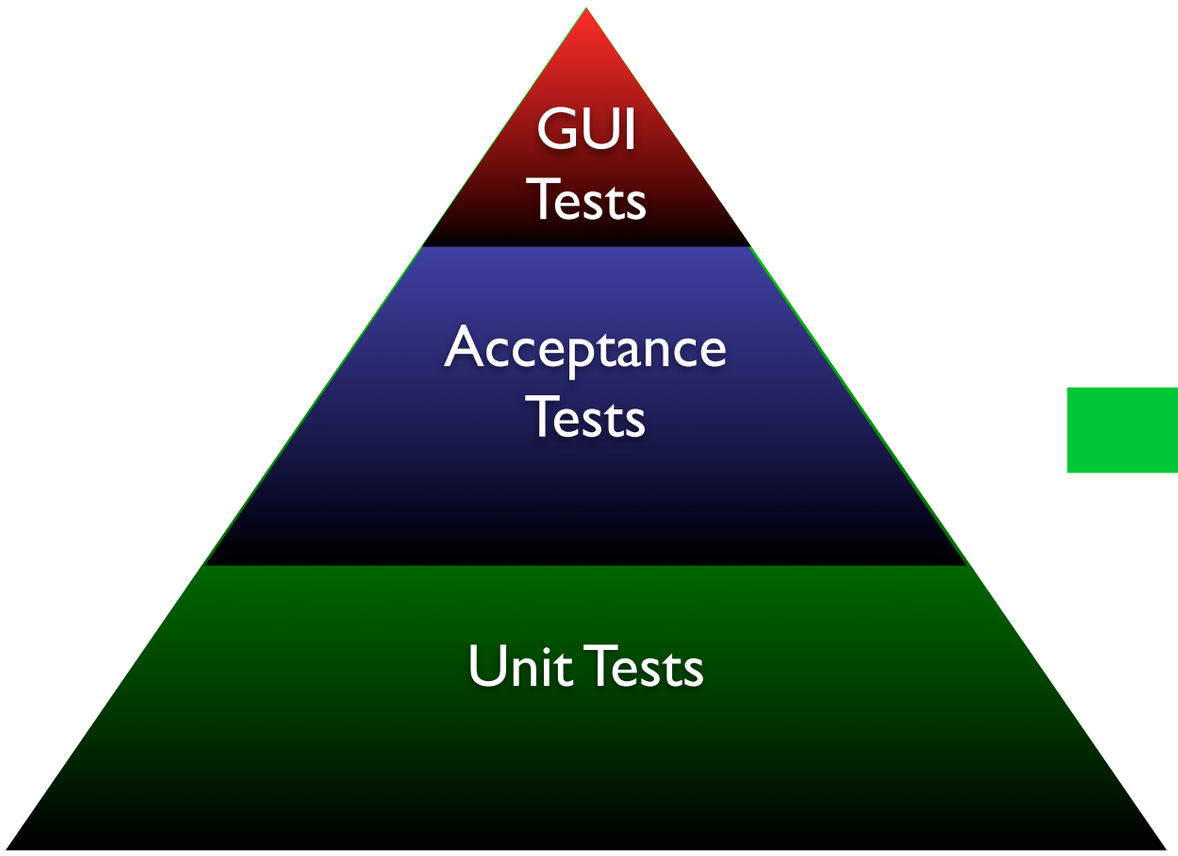
Best ROI for Testing



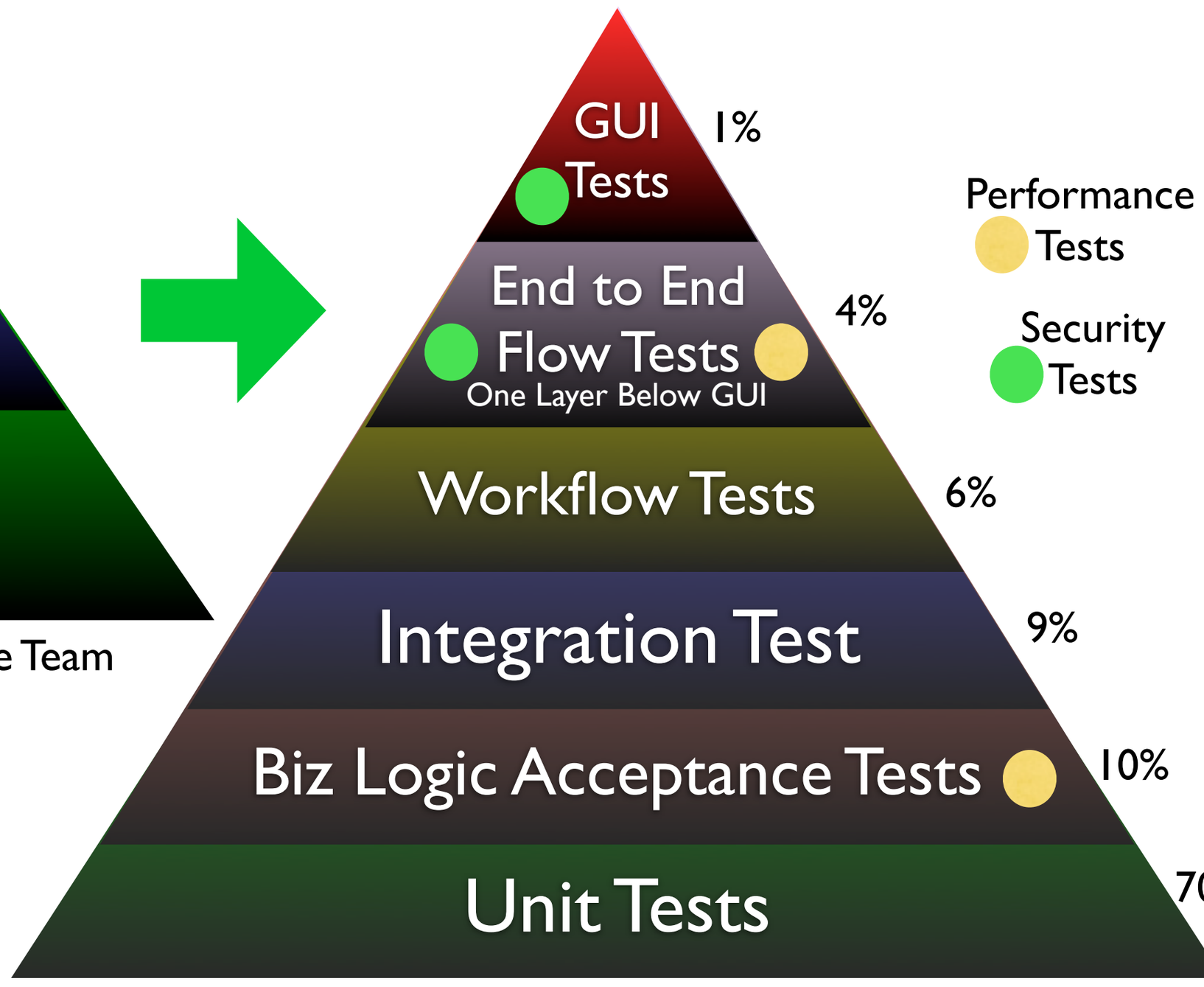
Inverting the Pyramid

How to…?

Types of Tests in the Testing Pyramid



Small Agile Projects with Single Team



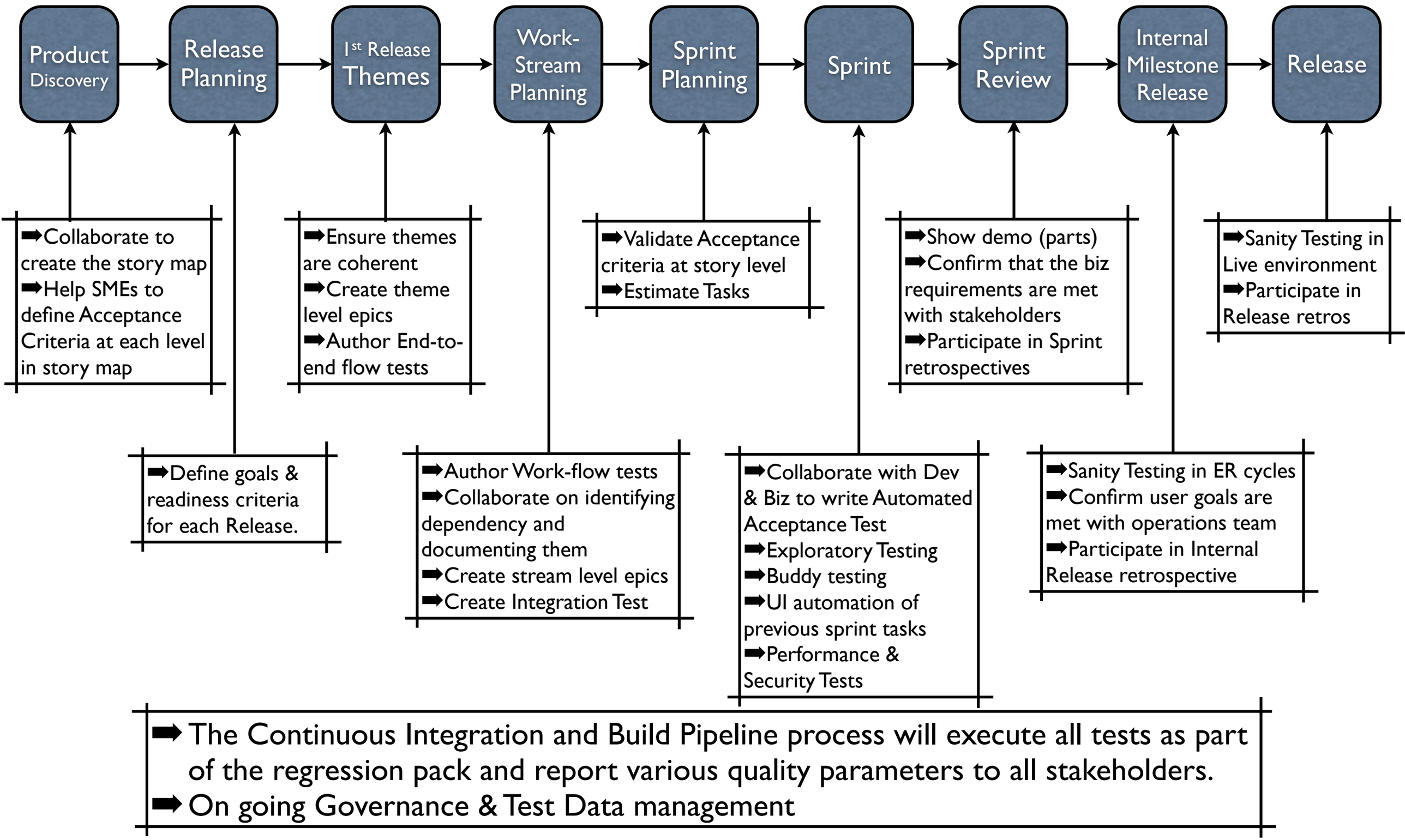
Large Scale Agile Projects with Global Teams

Types of Tests for Large Agile Project

Type	Tool	Who	When	Level	Objective
Unit Tests	xUnit, Mockito/ Moq, Jasmine, HSqlDB, Jumble	Developers	During the sprint, while coding	Class	Each class functions correctly in isolation. The UI components (including JS) can also be unit tested in isolation.
Biz Logic Acceptance Tests	Cucumber (some may need Lisa) JMeter/LoadRunner	TechBAs + Developers + Testers (QA)	Acceptance Criteria are defined before sprint starts. Acceptance Tests are written during sprint, before coding	Work-steam specific business functionality (At component level)	From a business logic point of view, the functionality is implemented correctly. Works in isolation with other work-streams/modules. Also good point to do some basic performance testing.
Integration Test	xUnit + Lisa	Developers + Testers (QA) + TechBAs	Contracts are defined before sprint, tests are implemented during the sprint	Between dependencies (work-streams, upstream/down-stream systems, 3rd Party)	2 Components can communicate with each other correctly. Best suited for negative path testing. Should not test any functional aspects of the application.
Workflow Tests	Cucumber + Lisa	SMEs + TechBAs + UX + Developers + Arch + Testers (QA + UAT)	When a theme is picked (functional group of sprints)	Across a few work-streams (Issuance/ Acceptance)	Inside a trade work flow, a particular step (issuance or acceptance workflow) works correctly from the business point of view. Upstream/Downstream dependencies stubbed.
End to End Flow Tests	Cucumber JMeter/LoadRunner	SMEs + TechBAs + UX + Developers + Arch + Testers (QA + UAT)	When an Internal Milestone is decided, the E2E flow tests are defined & authored. As & when the complete workflow is implemented these tests are enabled.	End to end Trade flow, one layer below the UI	Ensure a specific type of trade can go thru all the steps successfully. Very important to do detailed performance testing at this stage.
GUI Tests	Selenium/Sahi/ Watir/White/ SWTBot	TechBAs + UX + Testers (QA)	Usually its takes 3-4 sprints to complete one whole workflow including the UI. After that the UI tests will be authored and executed.	End to end Trade flow with GUI	Ensure few basic types of trades can go thru all the steps successfully via a UI.

agile faqs

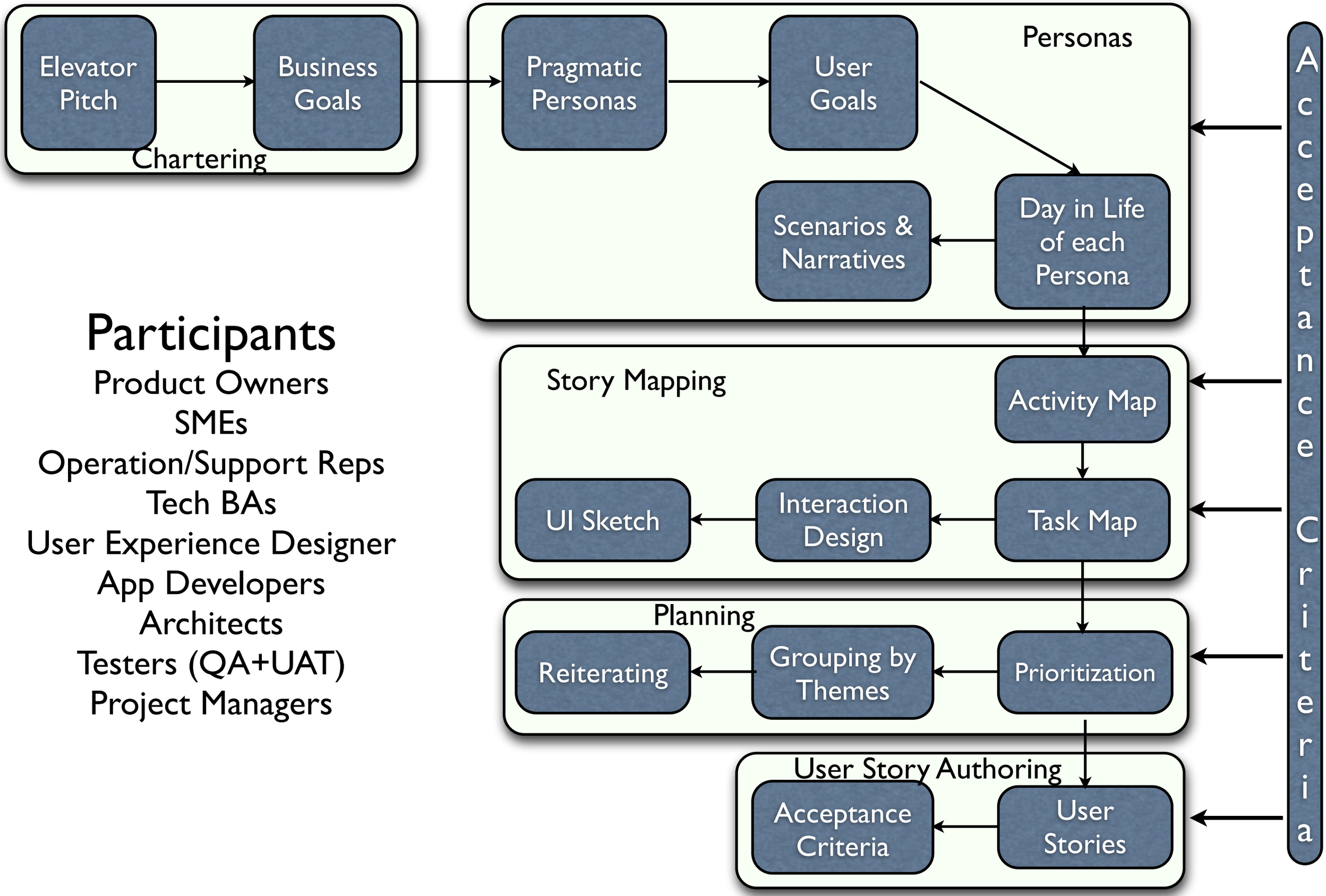
"Testing" Touch Points during Project Lifecycle



➡ The Continuous Integration and Build Pipeline process will execute all tests as part of the regression pack and report various quality parameters to all stakeholders.

➡ On going Governance & Test Data management

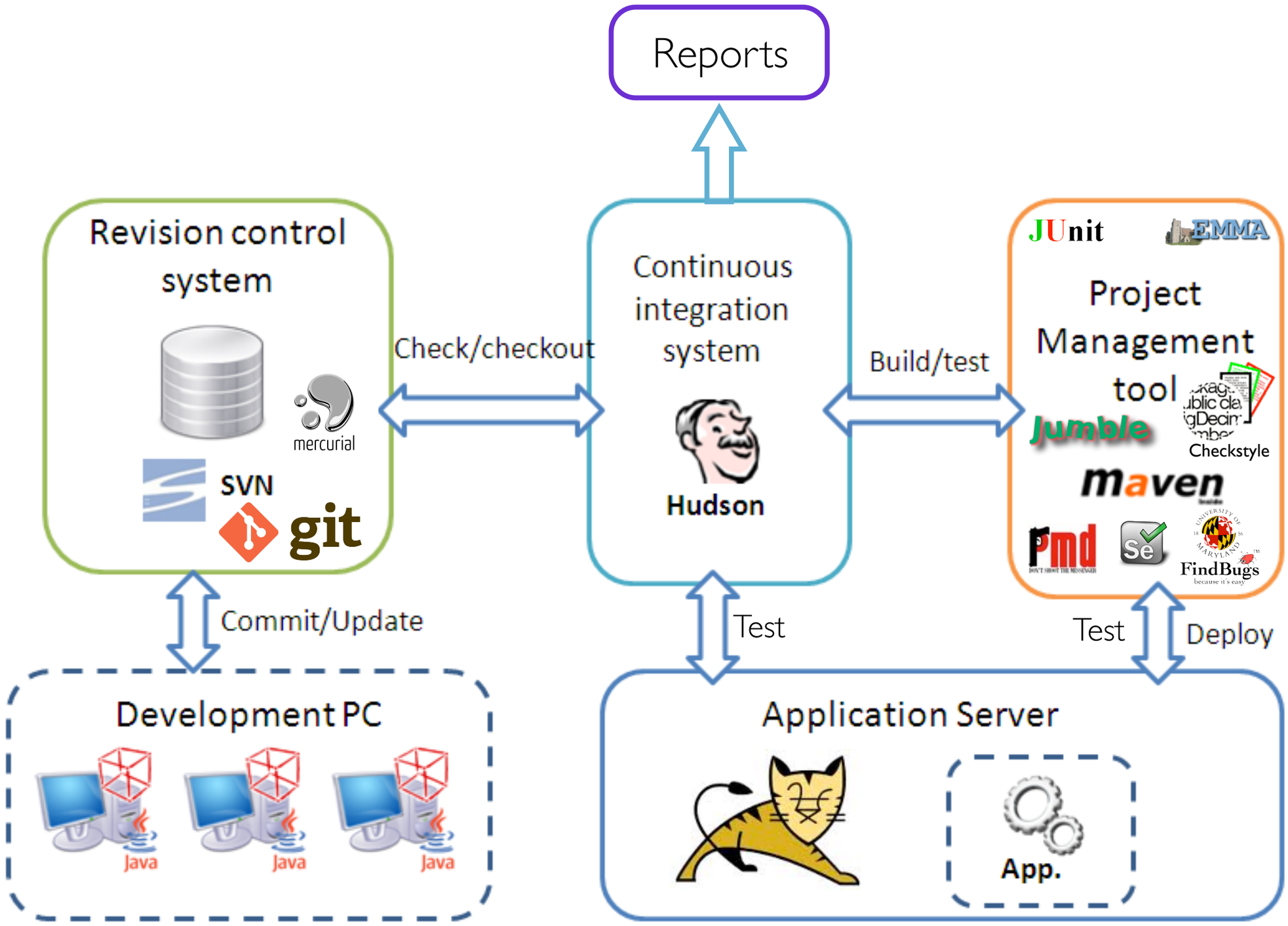
Collaborative Product Discovery Steps



Participants

Product Owners
SMEs
Operation/Support Reps
Tech BAs
User Experience Designer
App Developers
Architects
Testers (QA+UAT)
Project Managers

Typical Continuous Integration Process



agile faqs

Continuous Integration Build Pipeline

Dev Box

Local Dev Build ~5 Min

Update

Checkin

Version Control

Checkout

Checkout

Checkout

Checkout

Checkout

Team CI Server

Smoke Build ~10 Min

Team CI Server

Functional Build ~30 Min

Dev Env

Cross Stream Build ~60 Min

SIT Env

Theme Build ~60 Min

QA Env

Product Build ~60 Min

UAT Env

UAT Build ~30 Min

Staging Env

Staging Build ~30 Min

Prod Env

Live Build ~30 Min

Each build will update Task statues

Project/Task Management Tool

Retrieve Artifacts

Publish Artifacts (jars)

Retrieve Artifacts

Retrieve Published Artifacts

Retrieve Published Artifacts

Publish/Promote Dependency Artifacts

Retrieve Published Artifacts

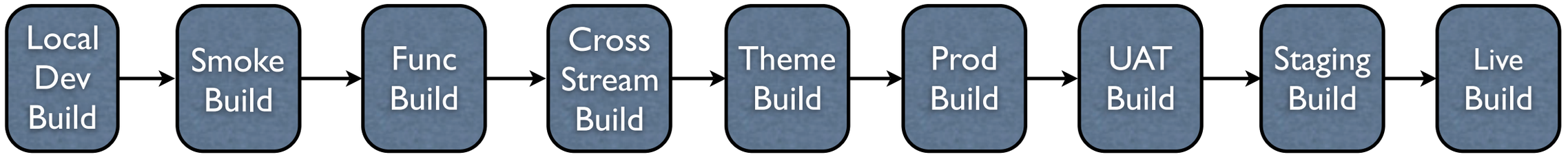
Retrieve Published Artifacts

Dependency Management

SCM

Each build will broadcast build stats and other reports to all relevant stakeholders at each stage of the build pipeline

agile faqs



Objective	Ensure basic sanity before the developer checks in code.	Ensure the developer has checked in all files, the code compiles, all unit tests work and basic code quality related stats are reported to the devs.	Ensure each story's business logic is implemented correctly and satisfies the doneness criteria at the story level	Ensure each story integrates with other streams and business logic continues to work as expected. Promote new produced jars to dependent teams.	Ensure each implemented, complete trade workflow works as expected. Interact with as many sub-systems & interfaces as possible.	Ensure each implemented end-to-end feature works as expected including various performance and security related SLAs are met. Also test out the migration scripts	Ensure Operations can execute an end to end workflow and it meets their goals.	Ensure the product as as a whole can work with other systems in the enterprise. Make Go/No-Go decision.	Deploy the product to the live environment.
Job	Retains the environment. Only compiles and tests locally changed code (incremental).	Compiles, Unit test and static/dynamic code analysis on an incremental environment	Clean environment, setup database, team specific biz acceptance tests. Stubs/Mocks out other functional modules or sub-systems.	Integrate promoted code with other modules, runs cross-team workflow & integration tests in an incremental env. Stubs/Mocks other sub-systems. Promote jars to dependent stream.	Clean environment, setup dependent systems, execute theme-level acceptance tests. As few stubs/ mocks as possible.	Migrate existing system to the latest version, setup dependent systems, execute product-level acceptance and UI tests. Almost no stubs/ mocks.	Migrate existing system to the latest version, setup dependent systems, let the Operations team use the system thoroughly.	Migrate existing system to the latest version, test the system with other systems during the Enterprise Release cycle.	Migrate existing system to the latest version and party!
Duration	~ 5 mins.	~ 10 mins.	~ 30 mins	~ 60 mins	~ 60 mins	~ 60 mins	~ 30 mins	~ 30 mins	~ 30 mins
Stake-holders	Developer writing the code	Developers of a given team	Team members of a given team + Teach BA	All functionally affected Teams + SMEs	All teams + SMEs	All teams + SMEs	SCM + SMEs + Operations Team	Enterprise Wide	World-wide
Trigger/ Frequency	Before checking in code. At least every 30 mins.	Every checkin. At least every 30 mins.	On successful Smoke build. At least every 2 hours	On successful Functional build. At least twice each day	On successful Cross-Stream build. At least once each day	On successful Theme build. At least once each week	Each Sprint	Each Enterprise Release Cycle	On Successful Staging Build (Manually triggered for now)
Where	On developer's workstation	Team CI Server	Team CI Server	Dev Environment	SIT Environment	QA Environment	UAT Environment	Stating Environment	Production Env
Artifacts	Code+Tests +Config	Team produced Jars, static code analysis reports	Team produced Jars/ Wars. Static+ Dynamic Code Analysis reports	Integrated jars pushed to dependent teams	Fully functional app with workflow level reports	Configured & packaged app (war) with migration scripts & traceability reports.	Configured & packaged app (war) with migration scripts	Configured & packaged app (war) with migration scripts	$$

➡ DevOps Roles (currently missing) & Test Data strategy is very important to get this working.
➡ Overall governance model will also be required to make sure build failures are addressed immediately.

agile faqs

Example Jenkins Build Pipeline Plugin's View

Jenkins

search

log in | sign up

Jenkins » phoenix-build-pipeline

DISABLE AUTO REFRESH

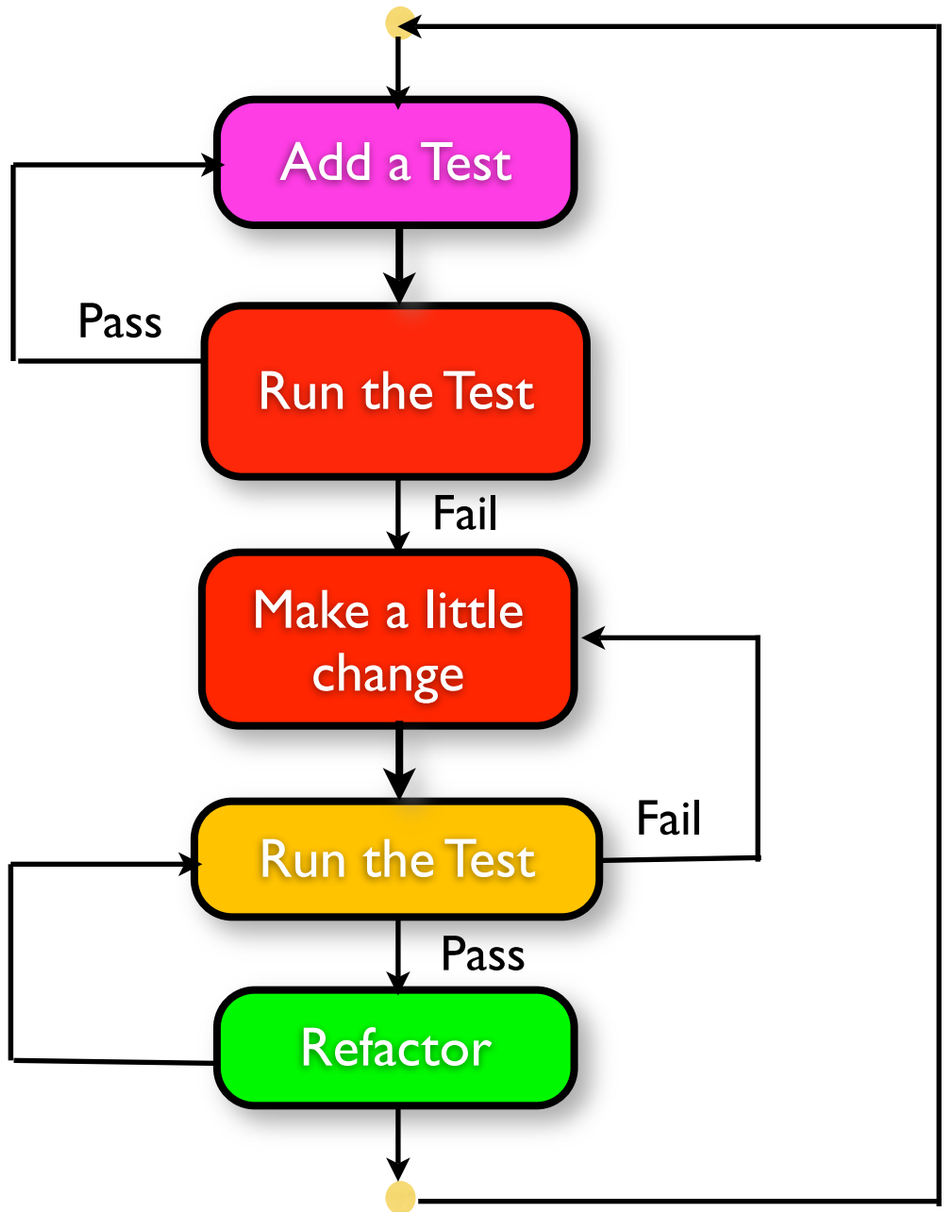
View/Hide Build Pipeline Icon Legend

Project Status	Build Stability	Build Status
Success	More than 80%	SUCCESS
Unstable	60-79%	UNSTABLE
Failure	40-59%	FAILURE
Abort	20-39%	ABORT
Building	Less than 20%	BUILDING

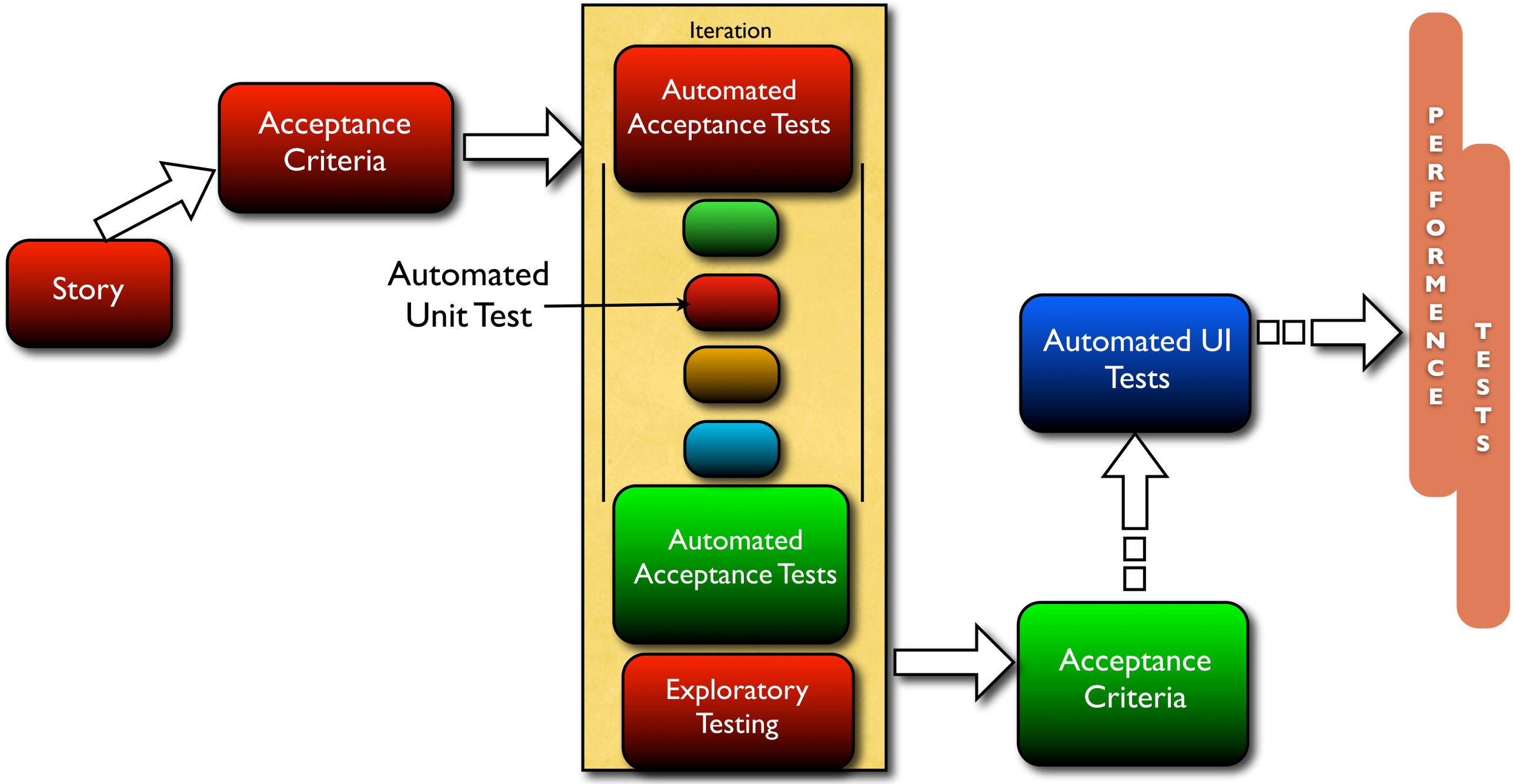
	phoenix-default	phoenix-integration-tests	phoenix-web-tests	phoenix-test-deploy	phoenix-uat-deploy	phoenix-production-deploy
revision :	phoenix-default #31 Duration: 17 sec	phoenix-integration-tests #29 Duration: 29 sec	phoenix-web-tests #31 Duration: 0.14 sec	phoenix-test-deploy #6 Duration: 90 ms	phoenix-uat-deploy #4 Duration: 0.1 sec	Manual Execution
revision :	phoenix-default #30 Duration: 16 sec	phoenix-integration-tests #28 Duration: 33 sec	phoenix-web-tests #30 Duration: 0.15 sec	phoenix-test-deploy #5 Duration: 0.74 sec	phoenix-uat-deploy #2 Duration: 0.34 sec	phoenix-production-deploy #2 Duration: 97 ms
revision :	phoenix-default #29 Duration: 17 sec	phoenix-integration-tests #27 Duration: 31 sec	phoenix-web-tests #29 Duration: 1.4 sec			
revision :	phoenix-default #28 Duration: 15 sec	phoenix-integration-tests #26 Duration: 34 sec	phoenix-web-tests #28 Duration: 1.4 sec			
revision :	phoenix-default #27 Duration: 16 sec	phoenix-integration-tests #25 Duration: 32 sec	phoenix-web-tests #27 Duration: 1 min 3 sec	phoenix-test-deploy #4 Duration: 0.12 sec	Manual Execution	

Test Driven Development

TDD Rhythm - Test, Code, Refactor



Acceptance Test Driven Development

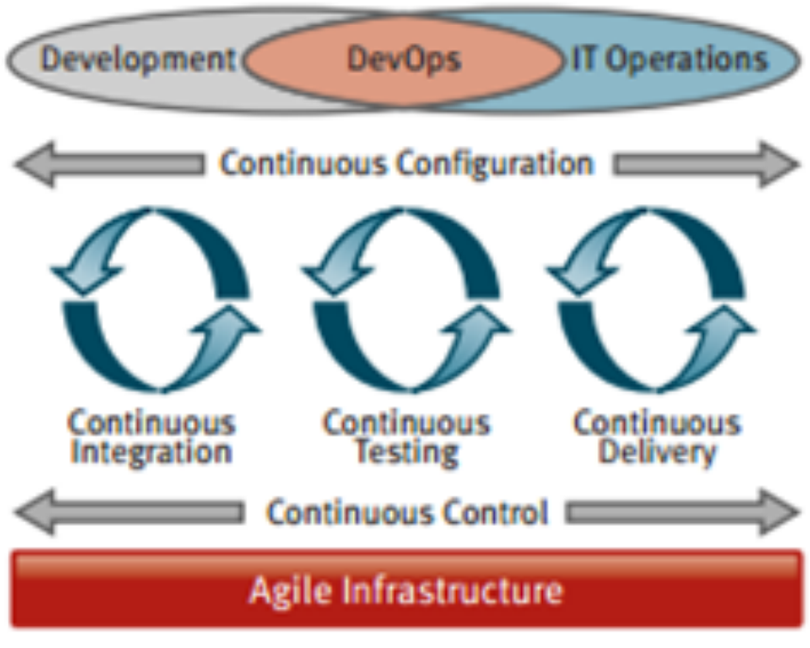


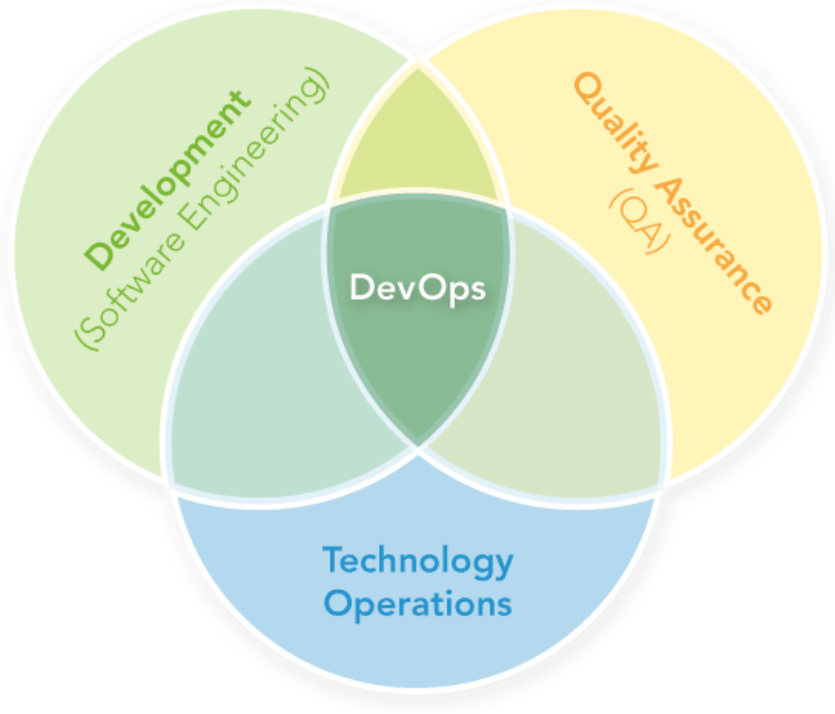
DEV OPS

is a verb

Lifecycle Management
Continuous Delivery
Repeatable Processes

Devops is application lifecycle management with the goal of continuous delivery achieved through the discovery, refinement and optimization of repeatable processes.





Thank you

Naresh Jain
naresh@agilefaqs.com