Welcome

Agile Testing

DevelopIntelligence

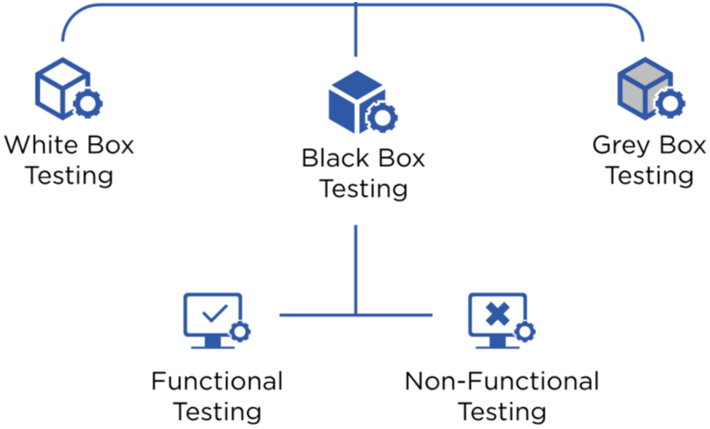
A PLURALSIGHT COMPANY

What is software testing?

Software testing is the act of:

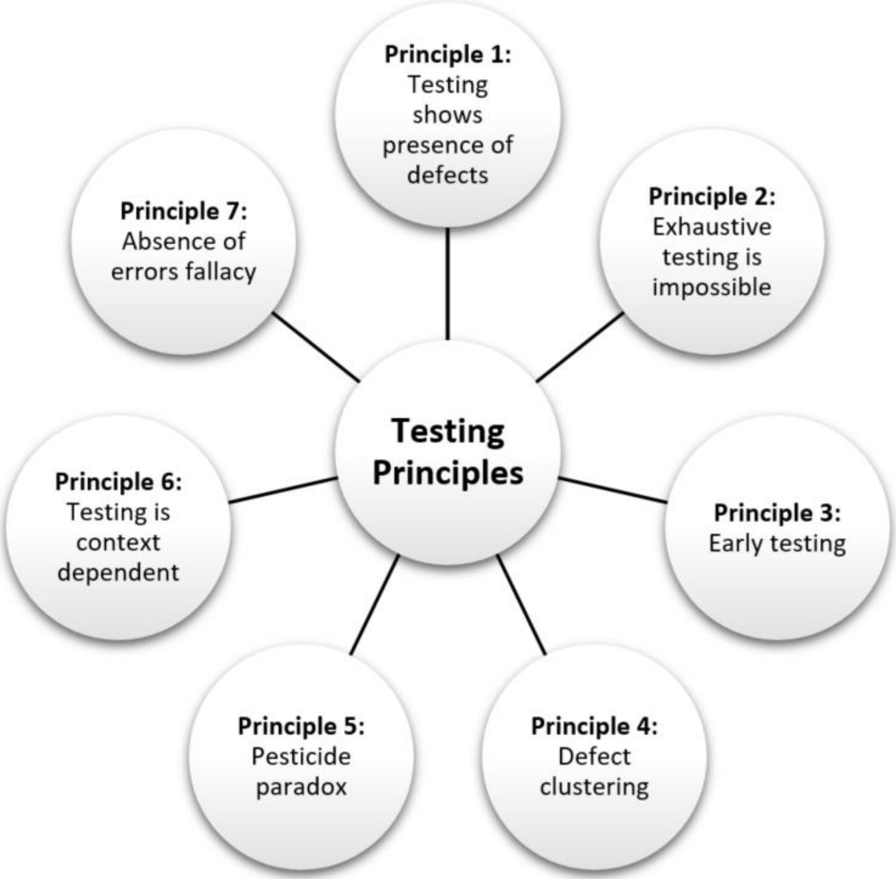
- Examining the artifacts and the behavior of the software under test by validation and verification.
- Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

- Wikipedia

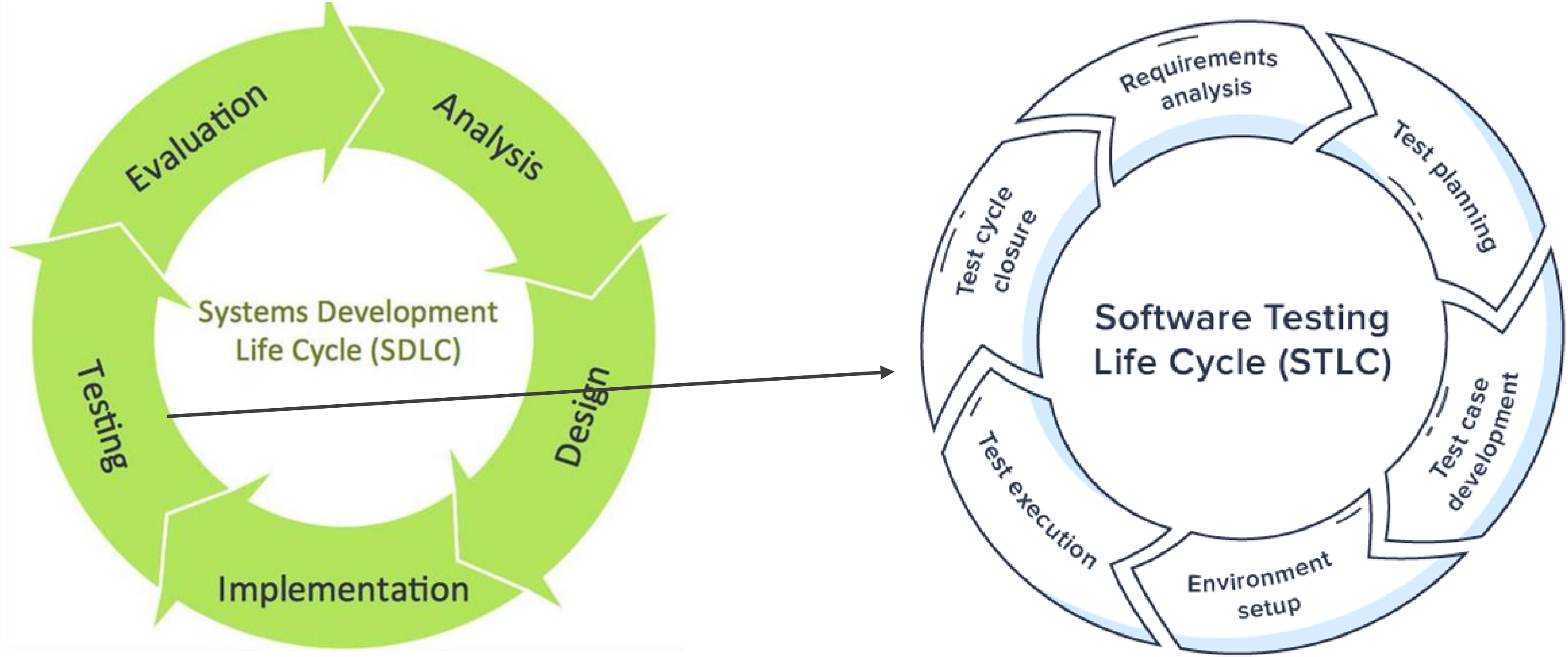


Software testing principles

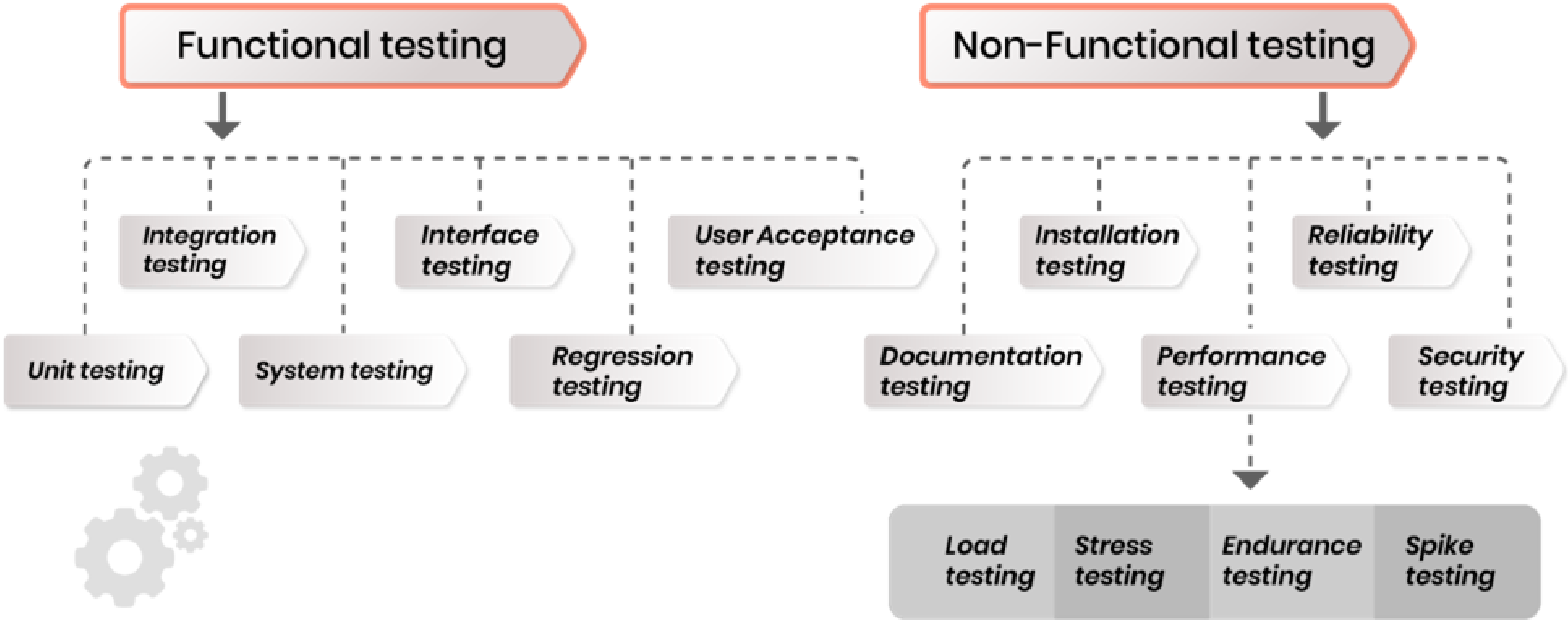
Myth: Principles are just for reference. I will not use them in practice.


Testing Principles
Principle 1: Testing shows presence of defects
Principle 2: Exhaustive testing is impossible
Principle 3: Early testing
Principle 4: Defect clustering
Principle 5: Pesticide paradox
Principle 6: Testing is context dependent
Principle 7: Absence of errors fallacy

Software Testing Lifecycle



Types of software testing



Functional testing

Unit testing

Integration testing

System testing

Interface testing

Regression testing

User Acceptance testing

Non-Functional testing

Documentation testing

Installation testing

Performance testing

Reliability testing

Security testing

Load testing

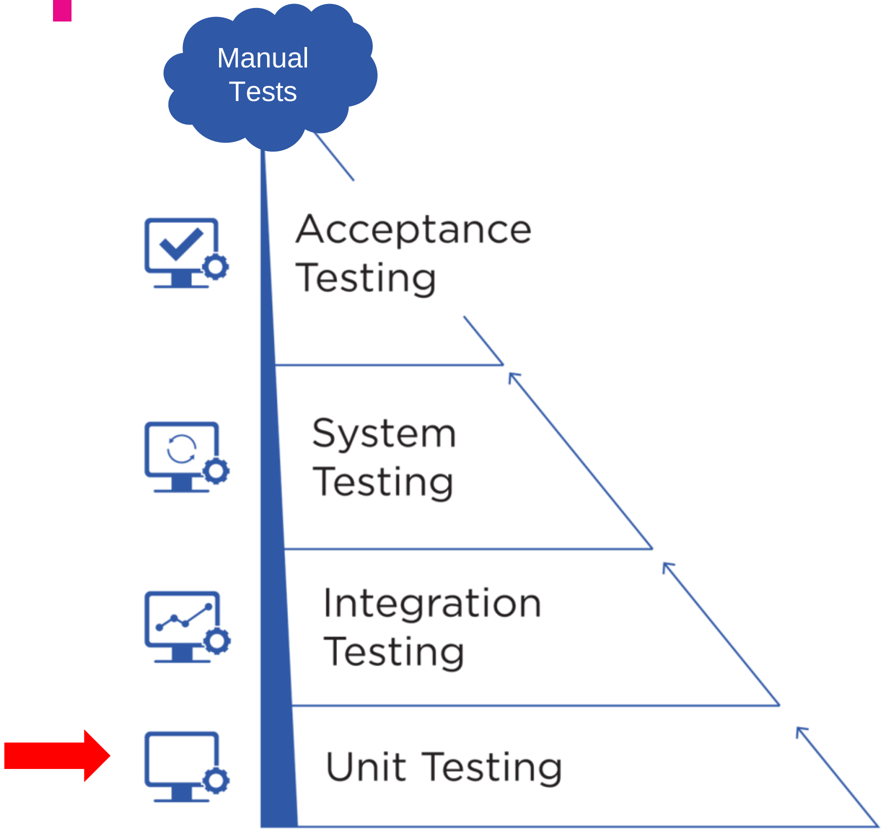
Stress testing

Endurance testing

Spike testing

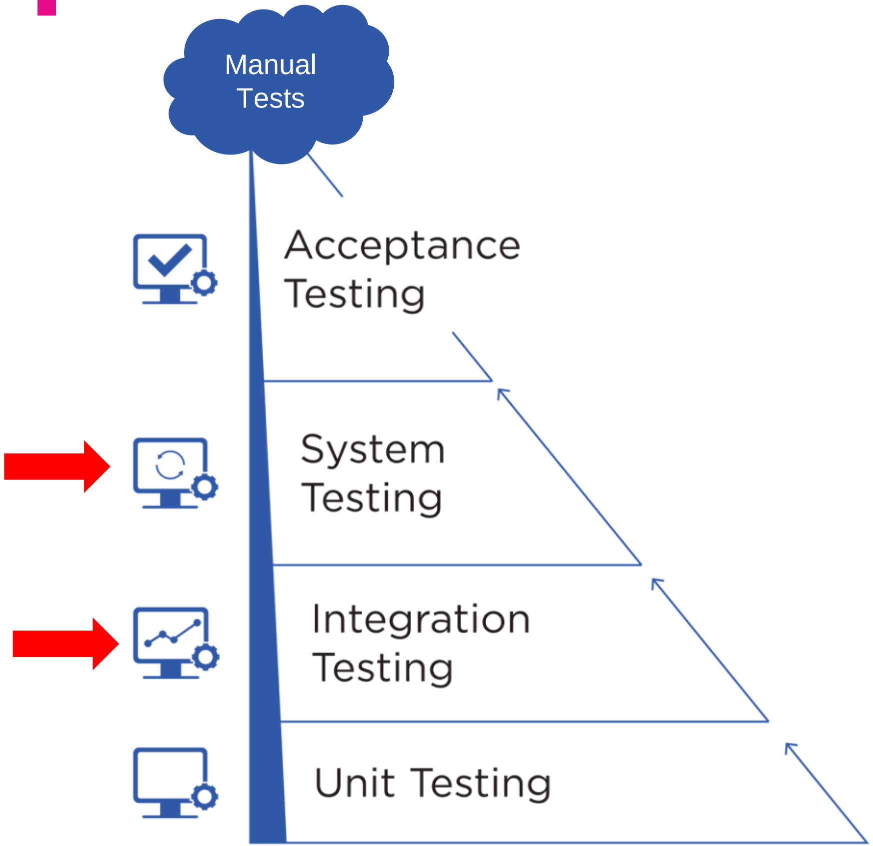
Types of Tests

Unit Testing



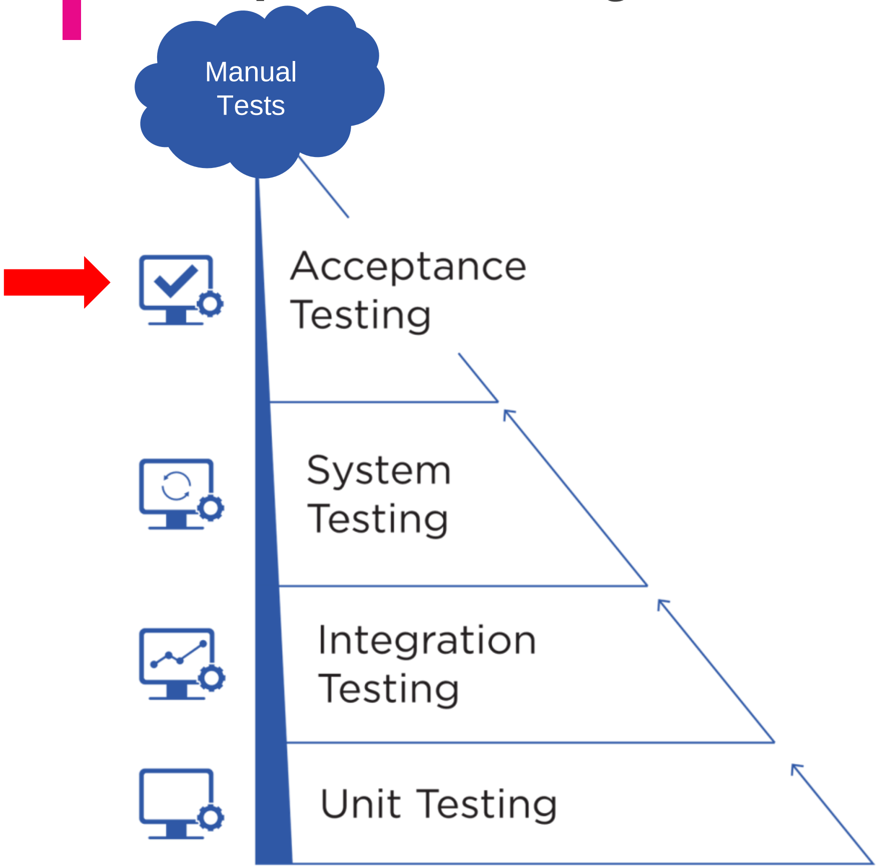
- Usually written by developers
- Uses Assertions
- Testing smallest part of the system in isolation
- Best Performance – execute early, often and within few seconds
- Smallest scope - easier to locate and understand errors
- Frequency – Every code check-in or pull request
- Use of test doubles to replace dependencies – dummies, stubs, spikes, mocks
- Increases confidence in changing/maintaining code

Middle Layer Testing



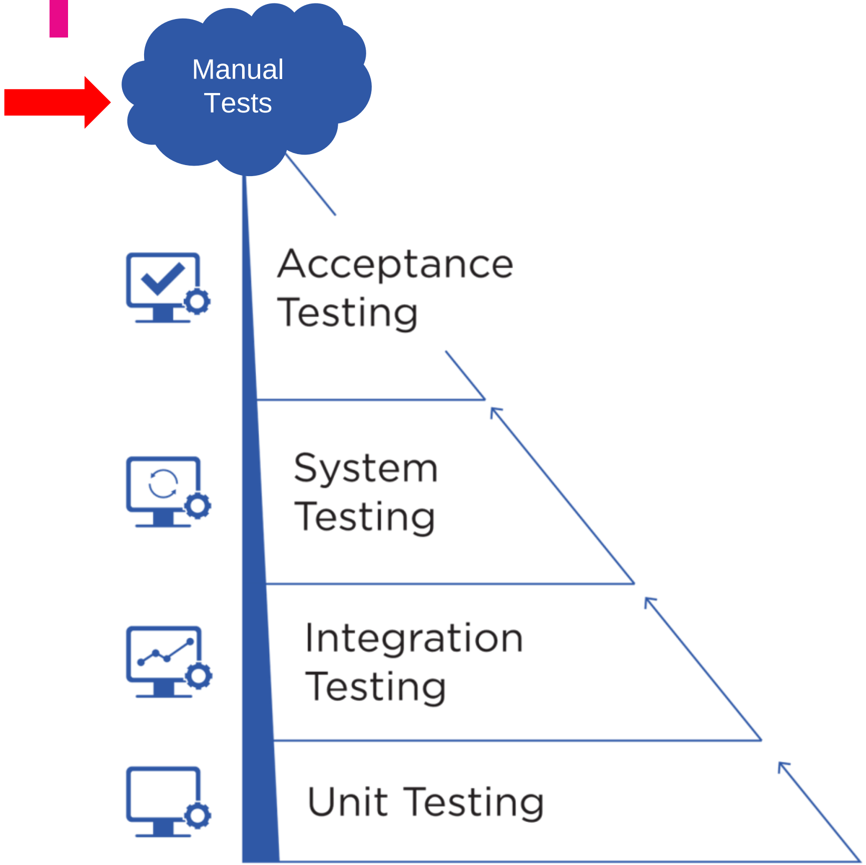
- Together called as service/API layer tests, also termed as “middleware layer”.
- Component Tests: Look at individual components. Validates the functionality is working as expected with other components.
- Integration Tests: Targets modules/features that integrate directly with other dependencies outside the application. Can be used as gating/staging from preprod to prod etc.
- System Integration Tests: Large scale integration suite testing end-to-end workflows. Often coordinated across teams and unique to your application or system.

Acceptance Testing



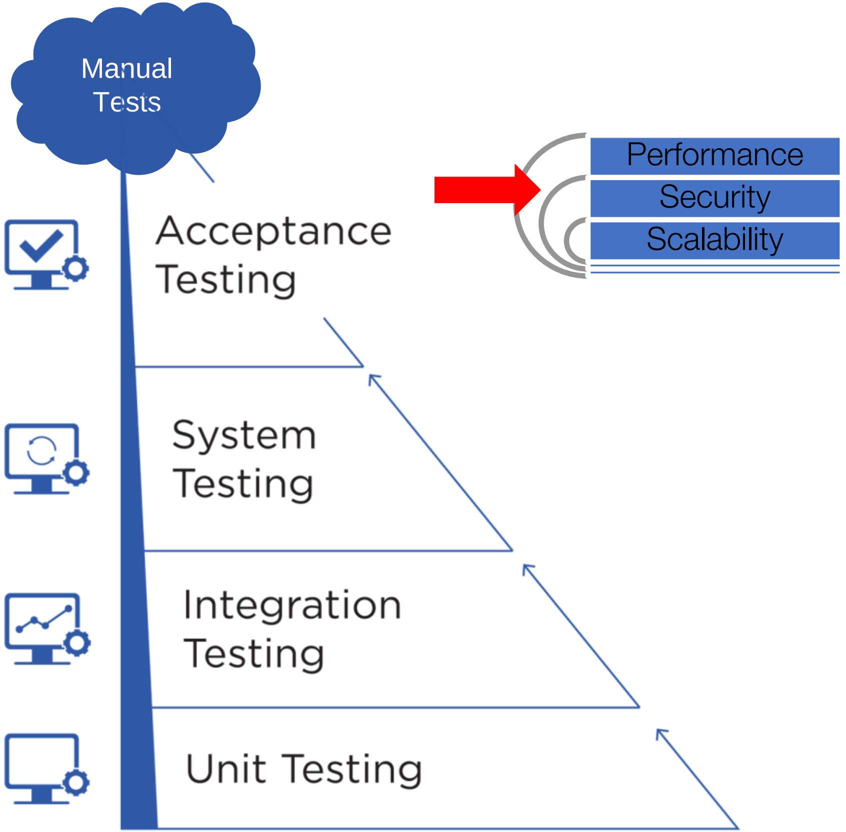
- Similar to integration tests in the sense that they test different parts of the application
- End-to-end testing
- Ensure high-level functionality works as expected or described and delivers business value
- Maximum scope such as - logics, UI workflows, navigation, transitions, calculations, buttons, layouts etc.
- Can be brittle or flaky and a lot of work to maintain
- Test critical workflows

Manual Testing



- Not regression testing
- Experience + creativity
- Learn about the system, discover defects & improve automated testing
- Can be based on missions/test charter/persona
- Covers scenarios which can’t be automated or are too complex to automate
- Edge Scenarios for mission critical applications to prevent failure

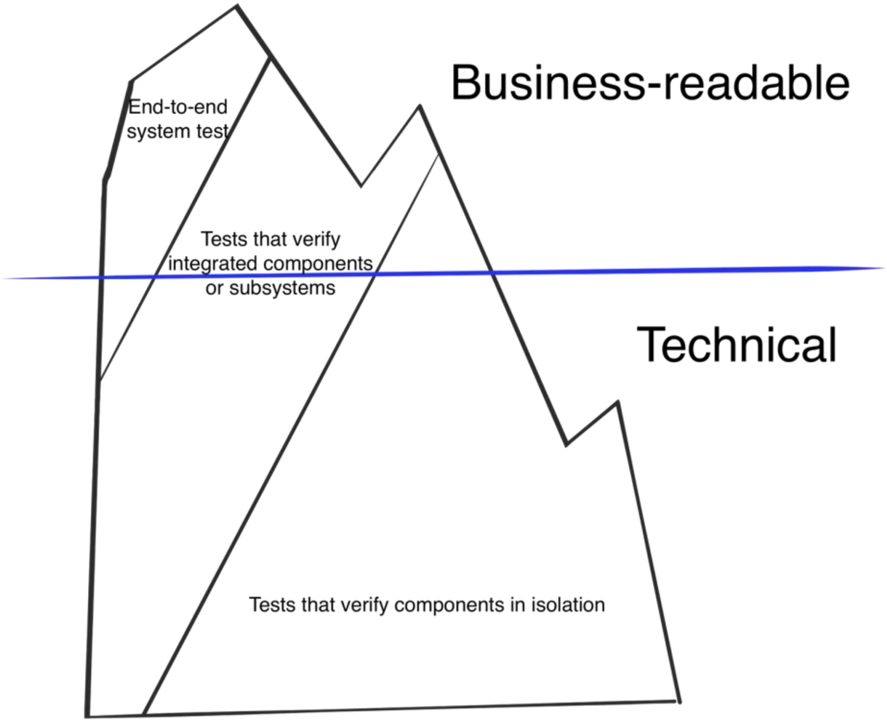
Performance, Security, Scalability



- **Performance Testing -** Generating data or metrics on the performance of your application. E.g. – A tipping point where your application can’t take any more requests.
- **Security Testing -** To check for vulnerability and security loopholes resulting in loss or unauthorized access of data. E.g.- PID (Personal Identification Information) of application users.
- **Scalability Testing -** Testing of a software application to measure its capability to scale up or scale out in terms of any of its non-functional capability.

Unit Testing

The Testing Iceberg

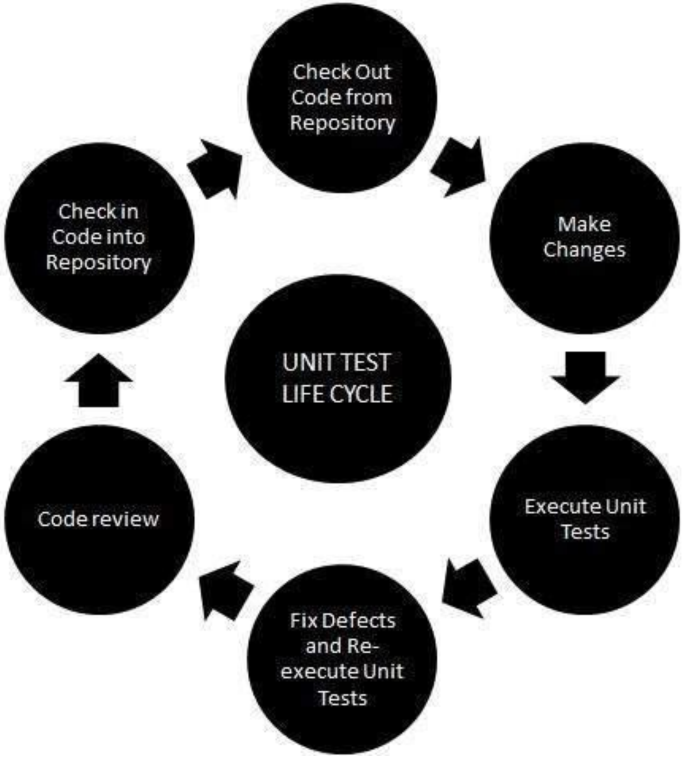


Business-readable
End-to-end system test
Tests that verify integrated components or subsystems
Technical
Tests that verify components in isolation

What is a unit test?

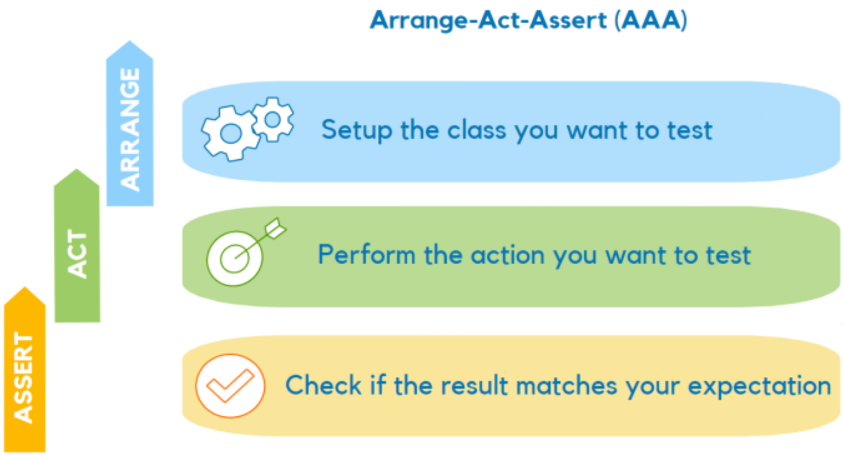
A unit is the tiniest testable component/functionality(unit) of the software. At this level, individual units of the software are tested.

- Concerned with functional correctness & completeness of individual units of code
- Usually has a few input(s) and a single output. One function may have multiple unit tests according to usage
- E.g. – Classes, functions, procedures, interfaces
- Primarily written and executed by the same person who writes code for that piece or component.
- Ensures :
  - Code meets expectation & specification
  - Code continues to meet expectations over time: Avoiding regression
- White box testing



UNIT TEST LIFE CYCLE
Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Unit Tests
Code review
Check in Code into Repository

Unit test structure - AAA

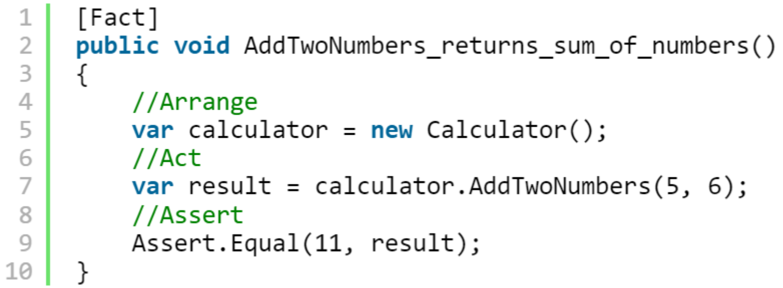


Arrange-Act-Assert (AAA)

ARRANGE — Setup the class you want to test

ACT — Perform the action you want to test

ASSERT — Check if the result matches your expectation



```
[Fact]
public void AddTwoNumbers_returns_sum_of_numbers()
{
    //Arrange
    var calculator = new Calculator();
    //Act
    var result = calculator.AddTwoNumbers(5, 6);
    //Assert
    Assert.Equal(11, result);
}
```

10 minutes

Demo – Our first unit test

Instructions:

Check if the title of book is – “All the Light We Cannot See”

Steps:

- “Arrange“ an object of class “Book”
- “Act” by calling getTitle() method
- “Assert” the returned result

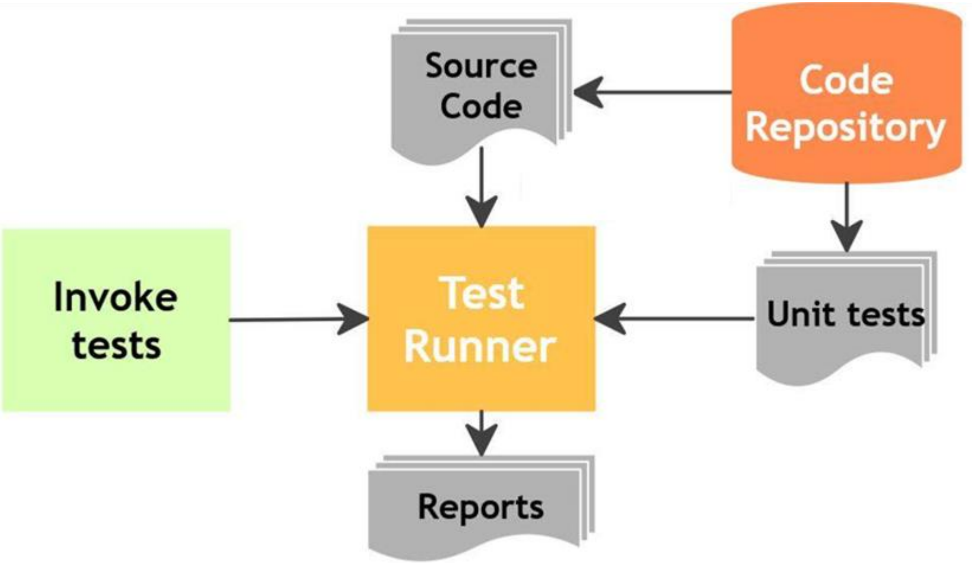
How was the first-code?

What did we just do?

What did you observe?

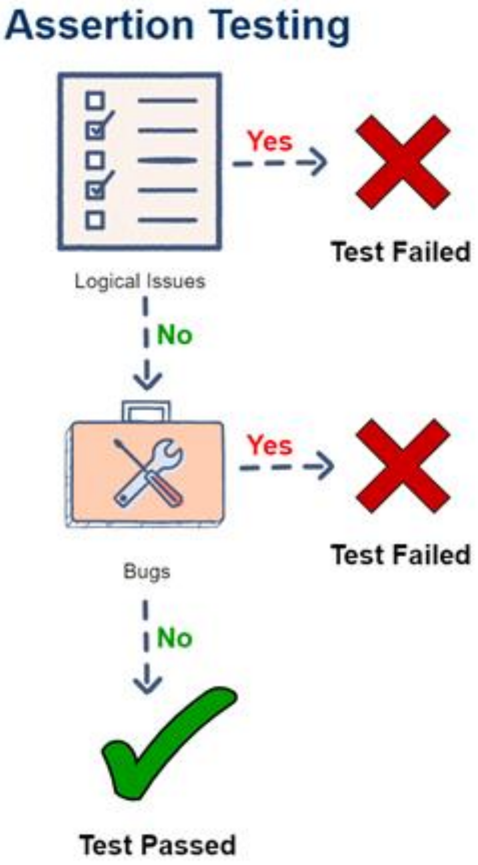
Did you understand the code?

What was new for you?



Assertion

- Used to test a logical expression
- At a binary level, assertion is just a Boolean expression – containing a true or false
- Assertions are used to declare the expected behavior or outcome of a test
- An assertion is true if the logical expression that is being tested is true and there are no bugs in the program.



Why unit test?

- Faster Debugging
- Faster Development
- Better Design
- Excellent Regression Tool
- Reduce future costs such as – bug fixes, maintenance, new additions etc.
- Integration testing become much easier
- Provides a living document of the system

Guidelines/Good Practices

- Keep them small and fast (able to execute within seconds)
- Should be fully automated
- Simple to run
- Measure the tests – code coverage
- Fix failing tests immediately
- Keep testing at unit level
- Keep tests independent
- Test the trivial cases too

Unit test tools

TestNG
JUnit
nunit
Jasmine
KARMA
MOCHA
chai
Gulp
Qu
Jest
istanbul
SINON.JS
jsdom

And many more…

15-20 minutes

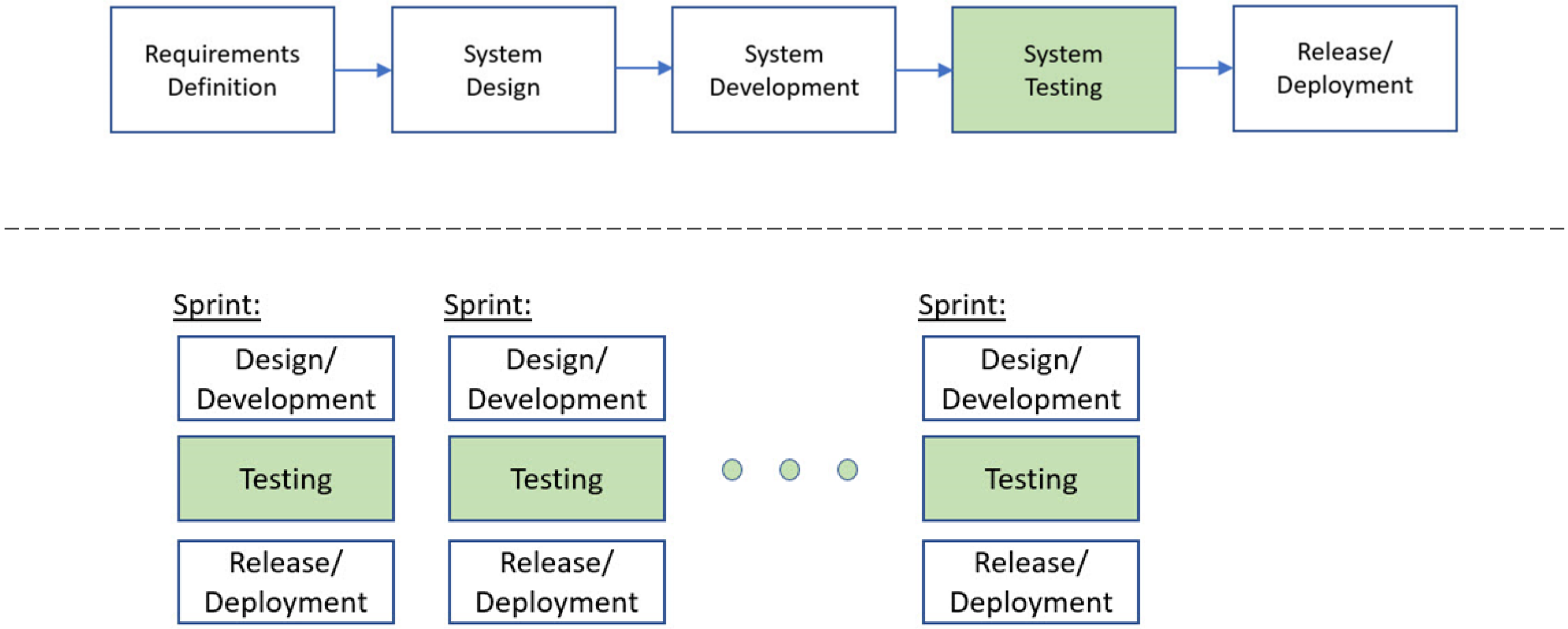
Activity – Write & execute few more unit tests

Instructions:

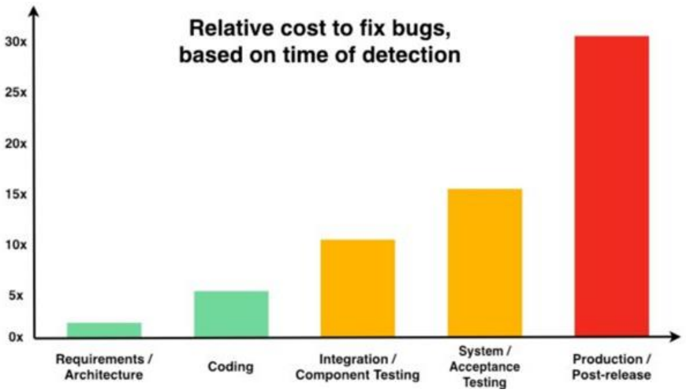
- Check ISBN of the book is – 1501173219
- Check publisher name is - Celadon Books
- Check the cost is $23.00

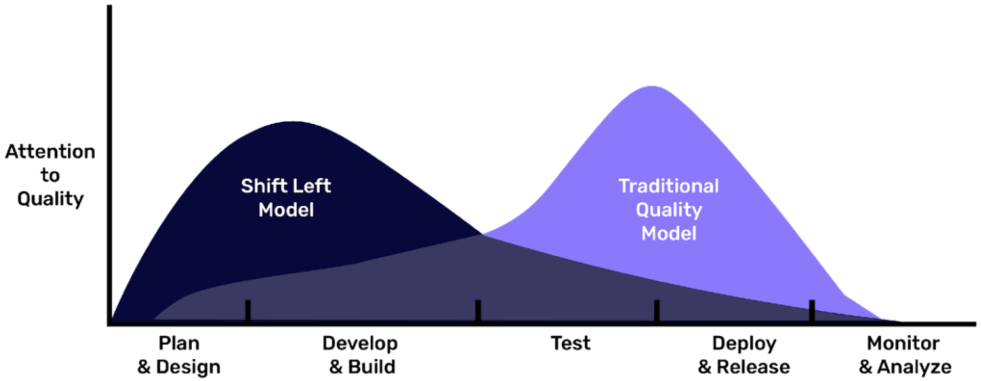
Testing in Agile – An Introduction

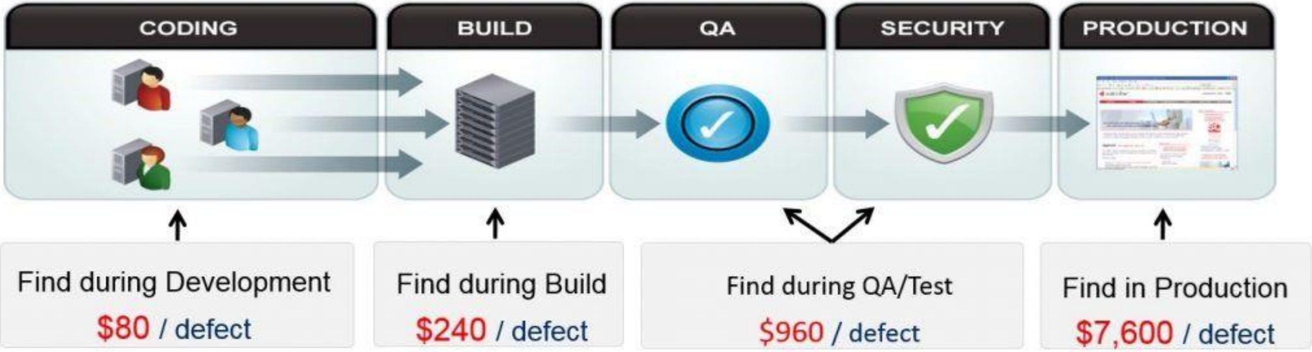
Agile Testing vs Traditional Testing



Shift left – why?

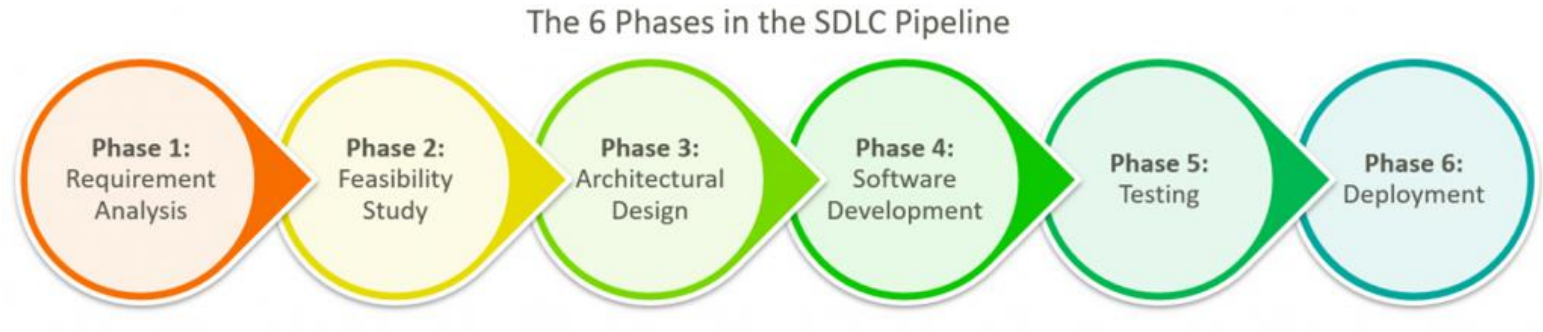


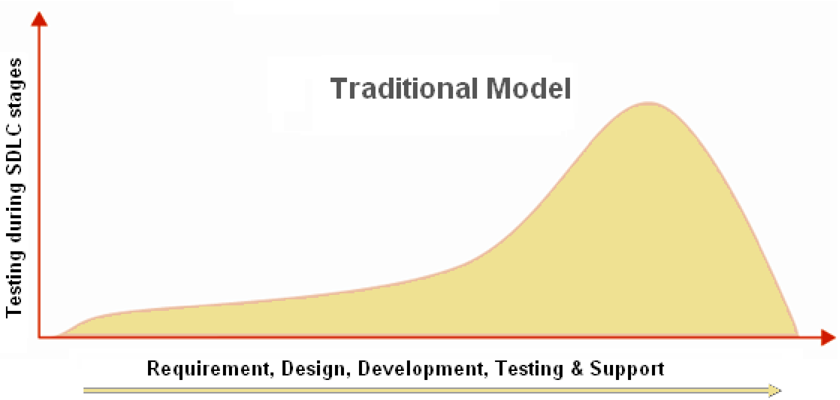


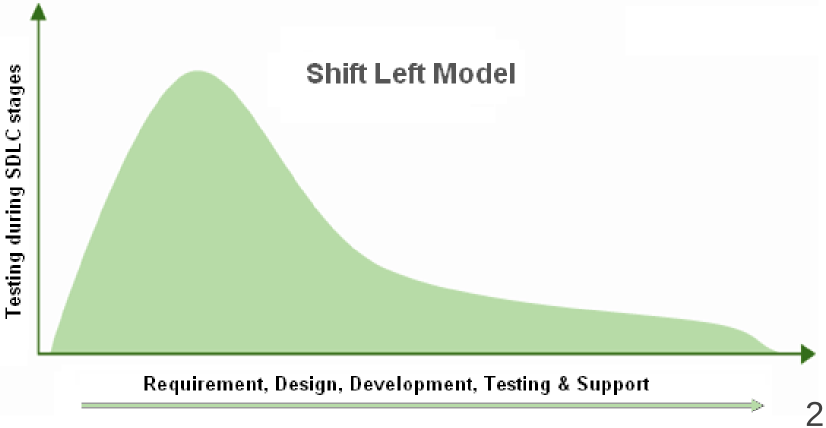


A research by Ponemon Institute, in 2017

Shift left – what?

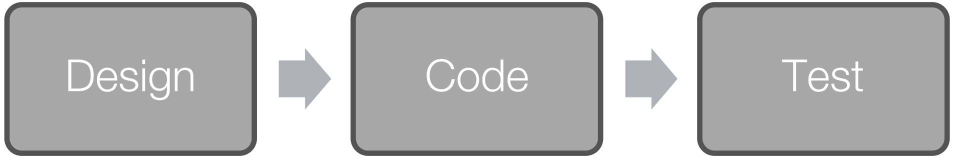

The 6 Phases in the SDLC Pipeline
Phase 1: Requirement Analysis
Phase 2: Feasibility Study
Phase 3: Architectural Design
Phase 4: Software Development
Phase 5: Testing
Phase 6: Deployment



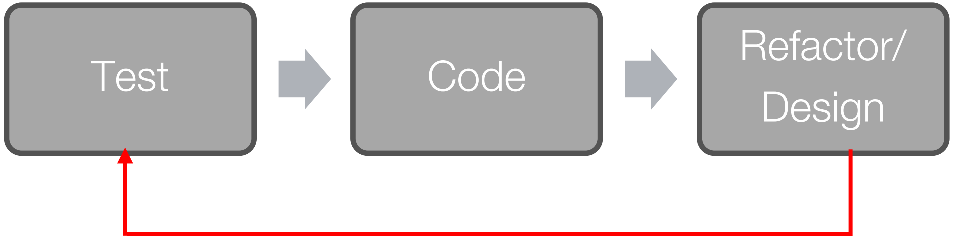


When do We Test?

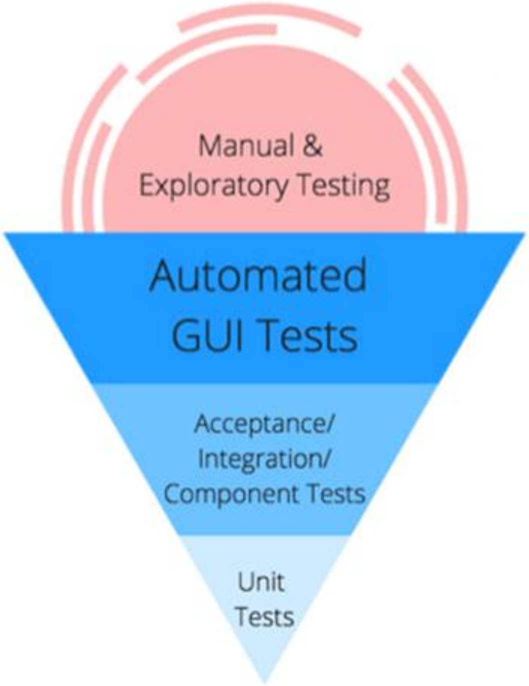
Traditional Approach



Agile Testing Approach

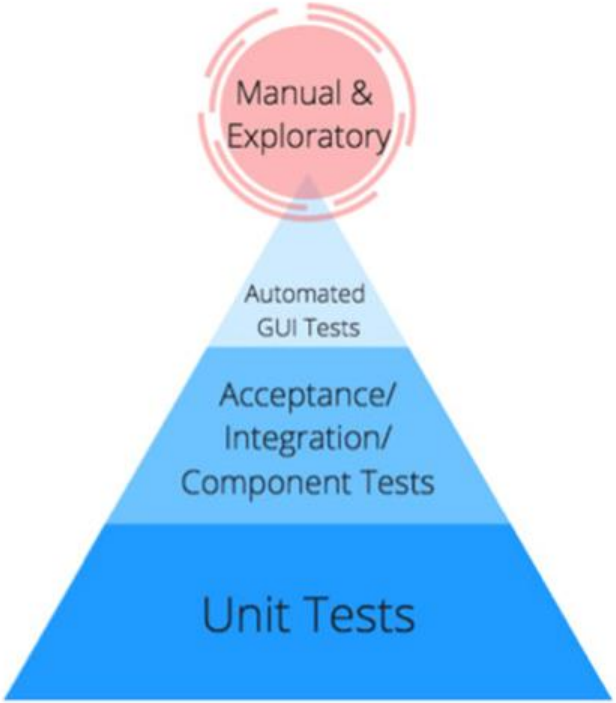


Testing Pyramids

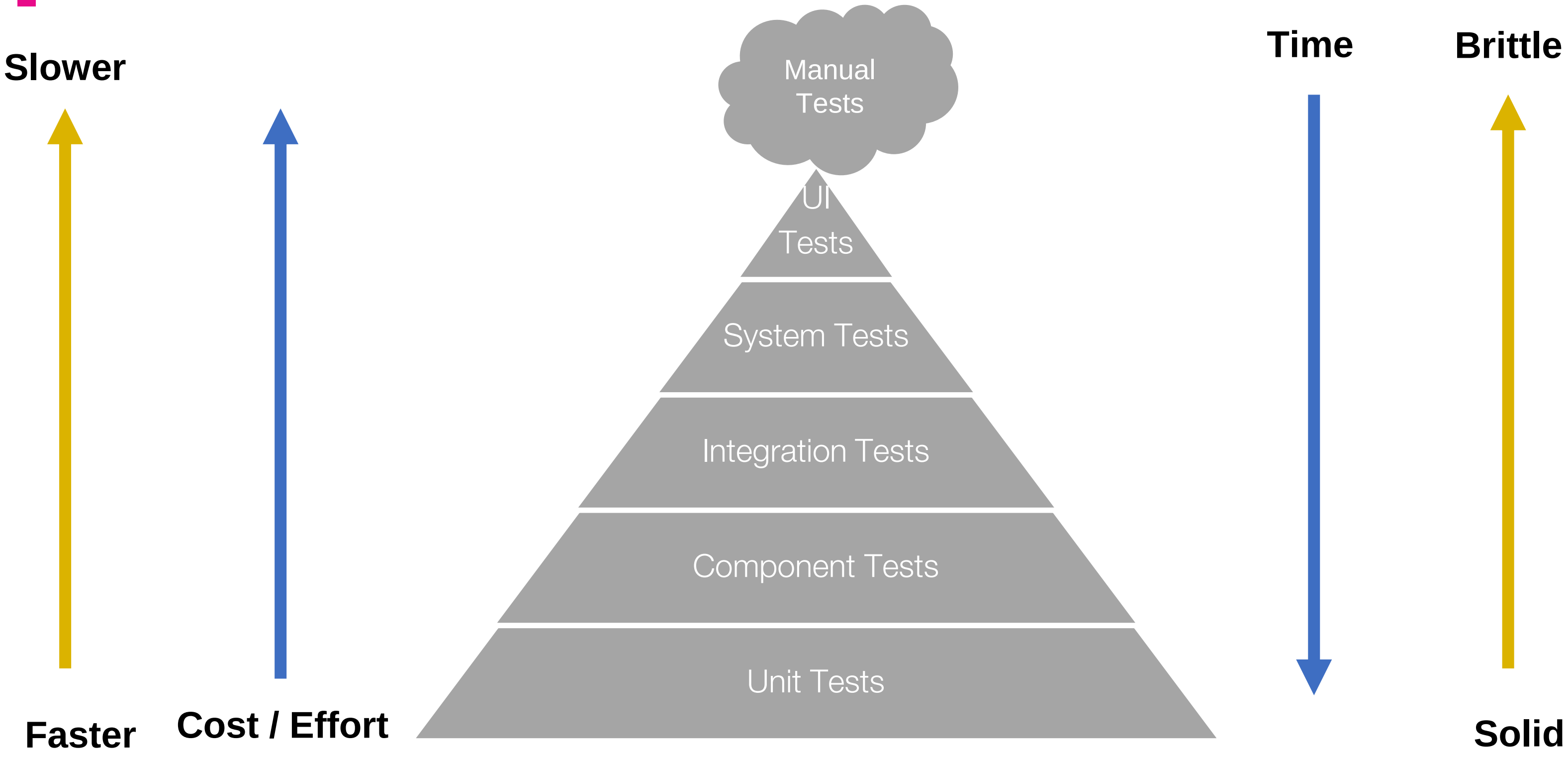


More Time & Effort

Higher ROI



Agile Testing Pyramid



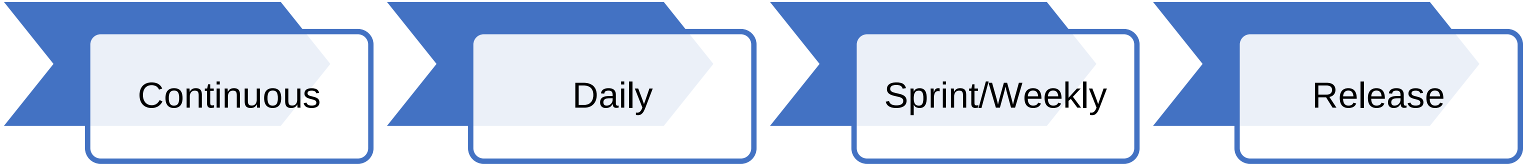
Agile Testing Strategy

- Every build
- Unit test
- Static, dynamic analysis
- Smoke tests

- Regression
- Functional test

- Long running tests
- Load/Performance
- Security

- UI tests
- User Acceptance Test (UAT)



Continuous

Daily

Sprint/Weekly

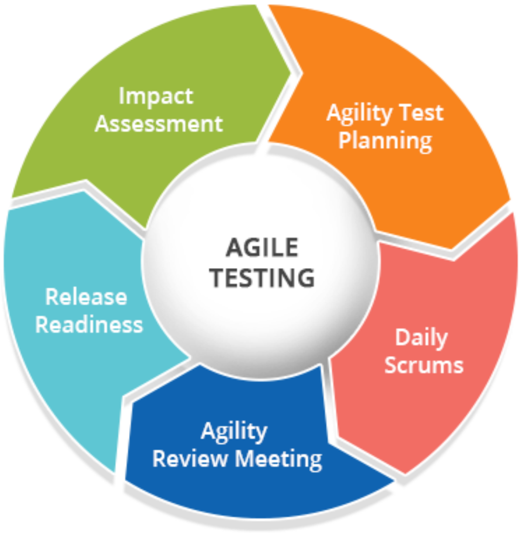
Release

Agile Testing Life Cycle

- Impact assessment
- Agile Testing Planning
- Release Readiness
- Daily Scrums
- Test Agility Review

Impact Assessment
Gather inputs from stake holders and users, this will act as feedback for next deployment cycle.

Approval Meetings for Deployment
At this stage we review the features that have been developed/ implemented are ready to go live or not.

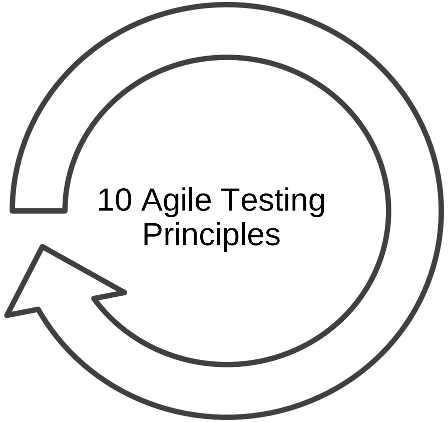


Start- Test Plan Meet Up
All stakeholders come together to plan the schedule of testing process, meeting frequency and deliverables.

Daily Standup Meetings
This includes the everyday standup morning meeting to catch up on the status of testing and set the goals for whole day.

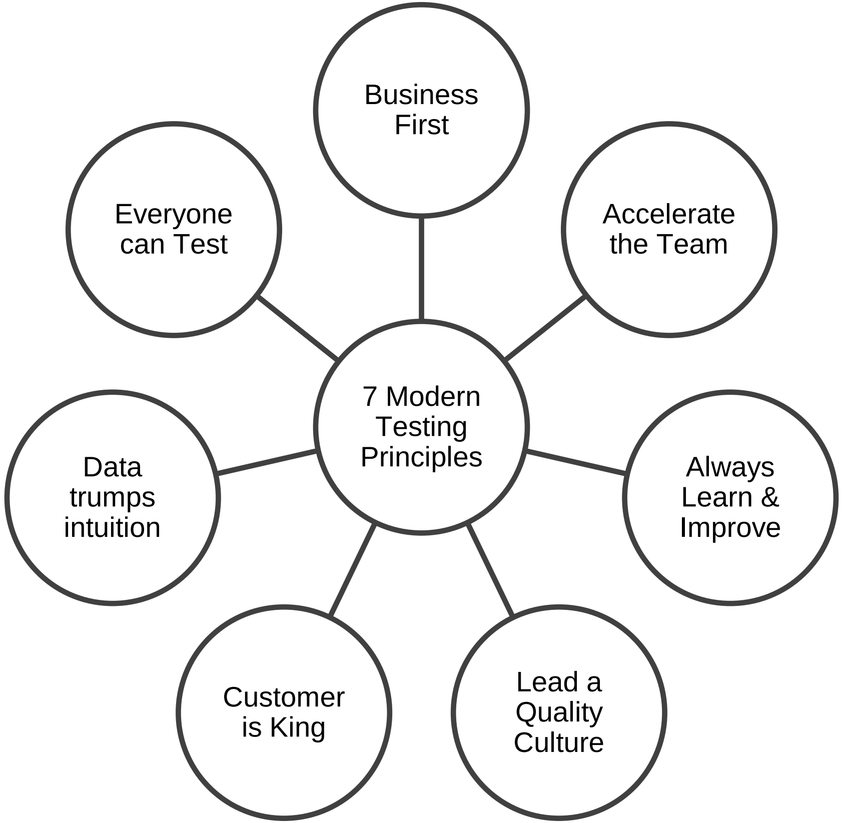
Agility Review Meeting
Weekly review meetings with stakeholders meet to review and assess the progress against milestones.

Agile Testing Principles



10 Agile Testing Principles

- Provide Continuous Feedback
- Deliver Value to Customer
- Enable face-to-face communication
- Have Courage
- Keep it Simple
- Practice Continuous Improvement
- Respond to Change
- Self-organize
- Focus on people
- Enjoy



7 Modern Testing Principles

Business First

Accelerate the Team

Always Learn & Improve

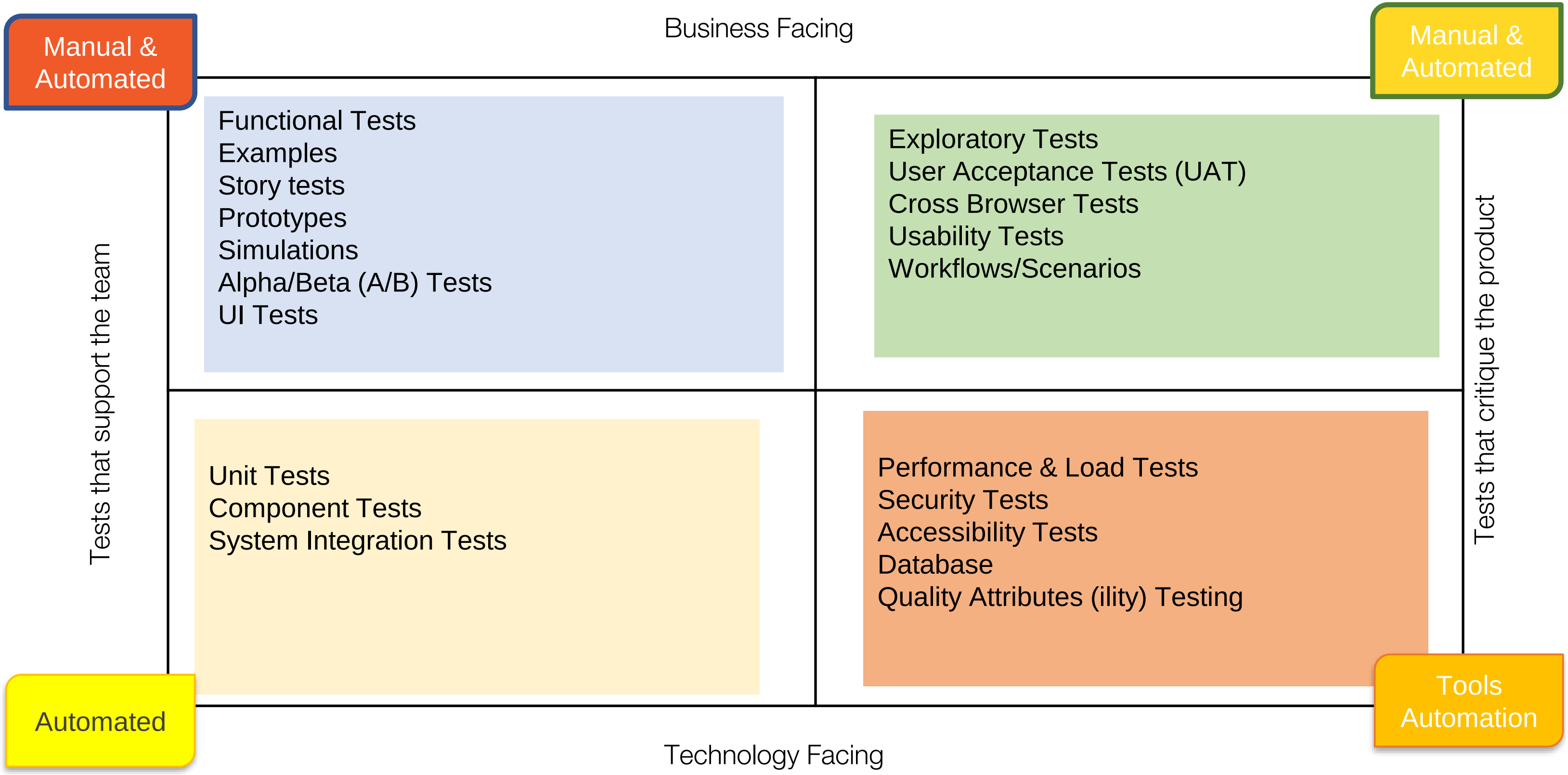
Lead a Quality Culture

Customer is King

Data trumps intuition

Everyone can Test

Agile Testing Quadrants



Business Facing

Manual & Automated

Functional Tests
Examples
Story tests
Prototypes
Simulations
Alpha/Beta (A/B) Tests
UI Tests

Manual & Automated

Exploratory Tests
User Acceptance Tests (UAT)
Cross Browser Tests
Usability Tests
Workflows/Scenarios

Tests that support the team

Tests that critique the product

Unit Tests
Component Tests
System Integration Tests

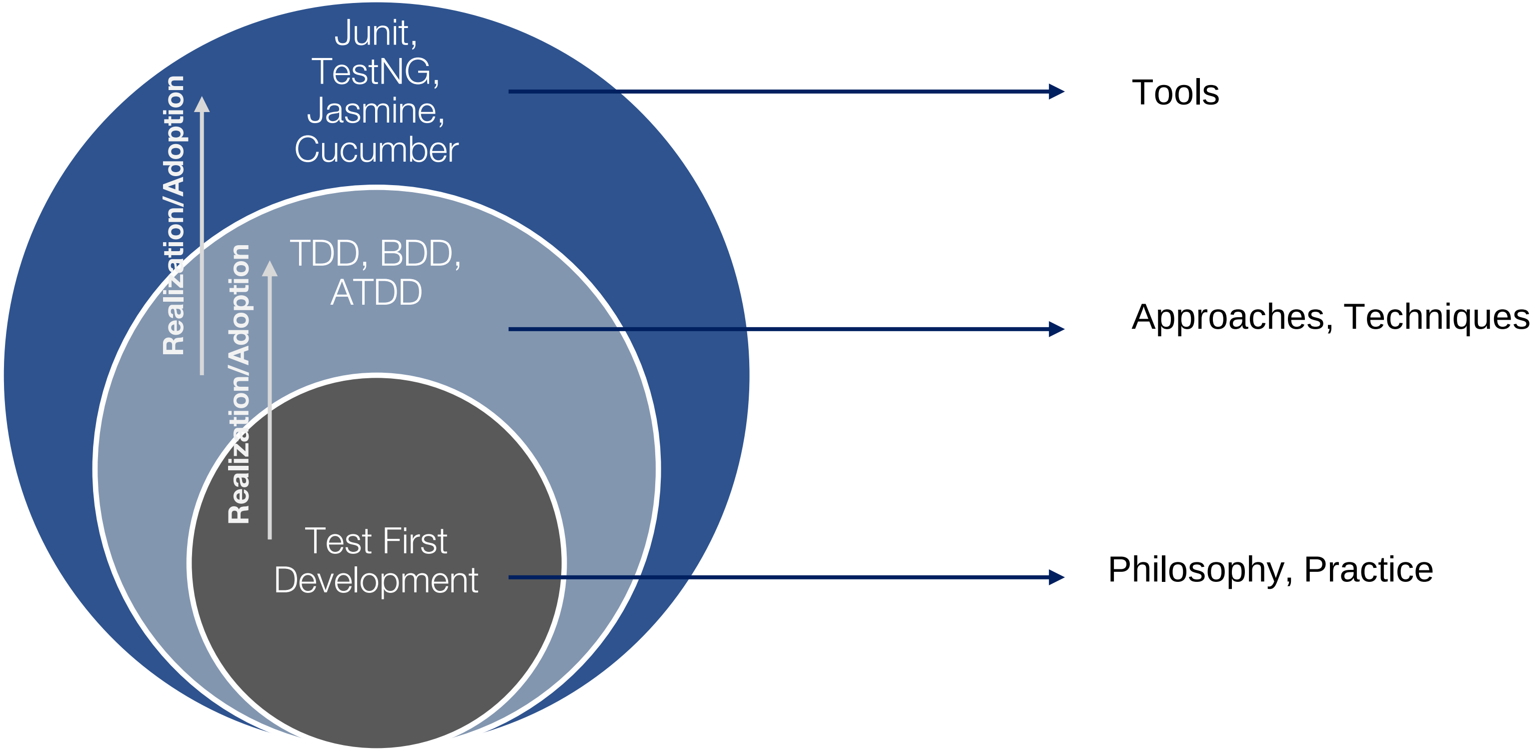
Automated

Performance & Load Tests
Security Tests
Accessibility Tests
Database
Quality Attributes (ility) Testing

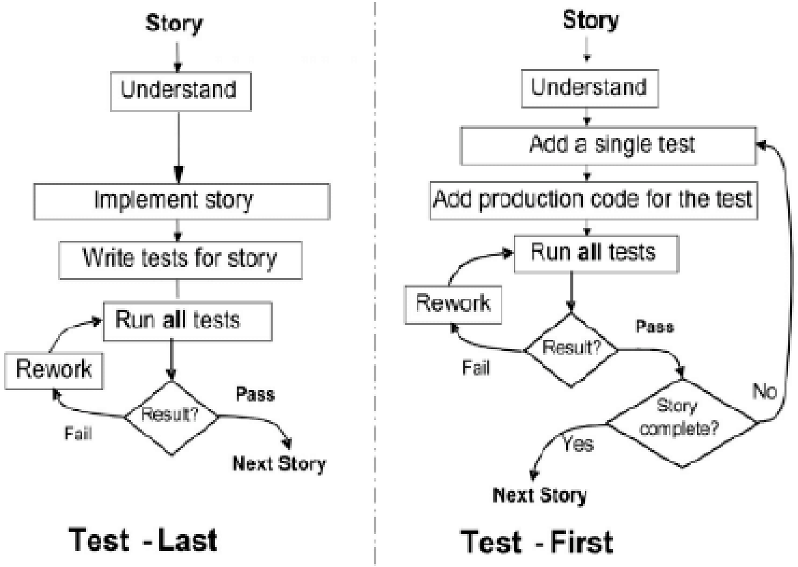
Tools Automation

Technology Facing

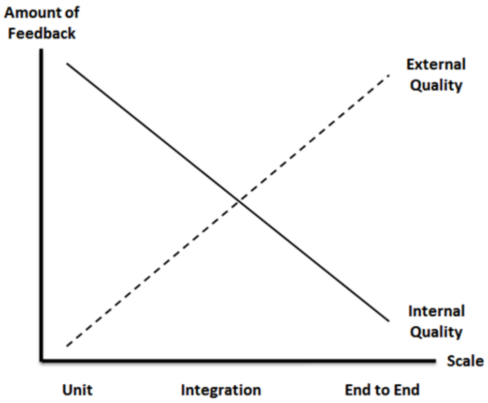
Agile Testing - Test First Development



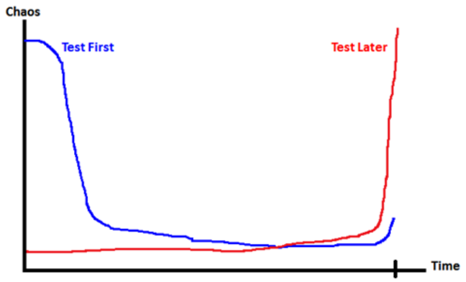
Test First vs Test Last



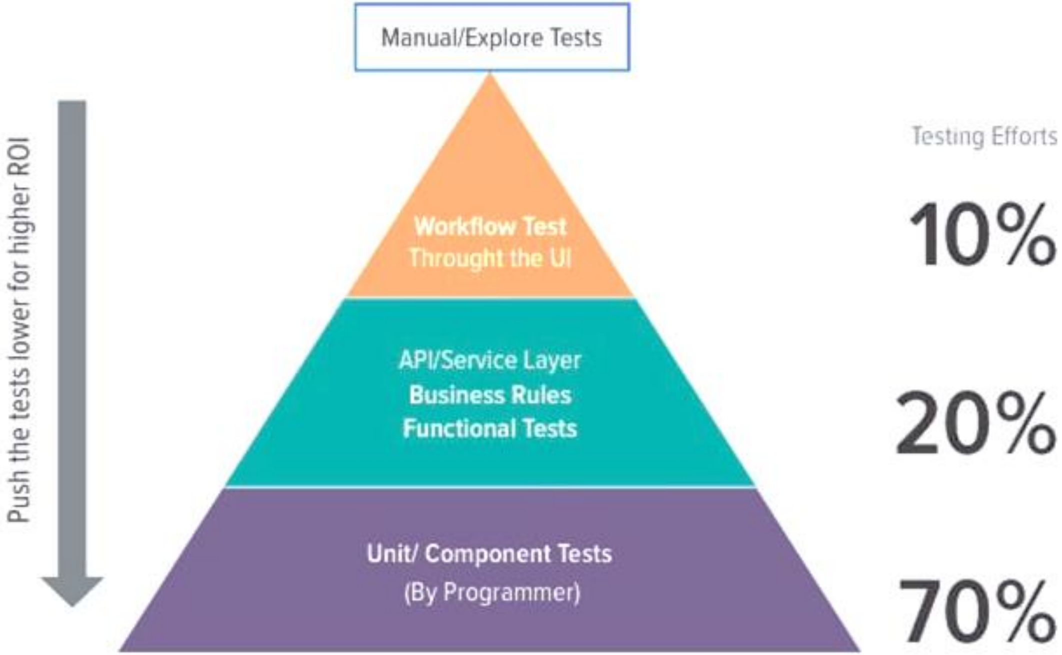
Feedback from Tests



Visible Uncertainty in Test-First and Test-Later Projects

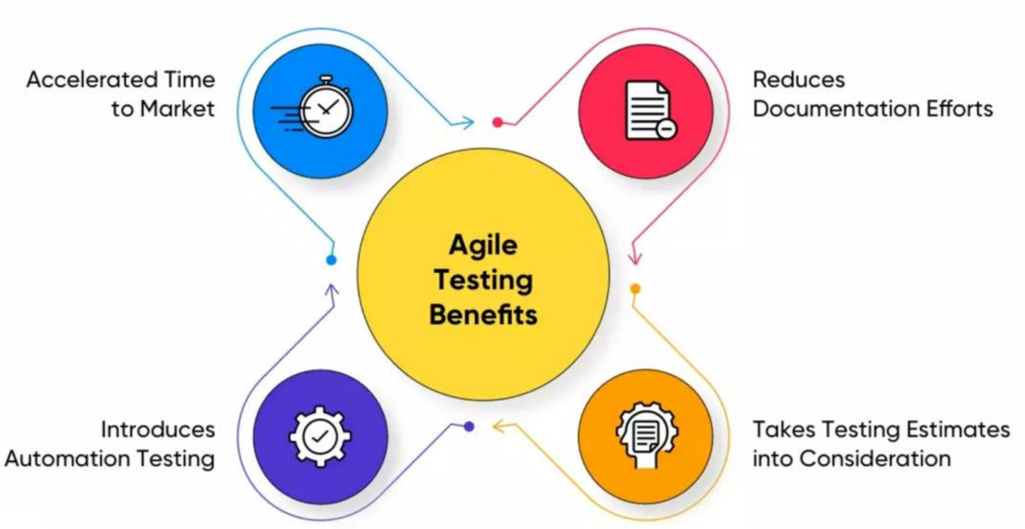


Agile Testing Efforts



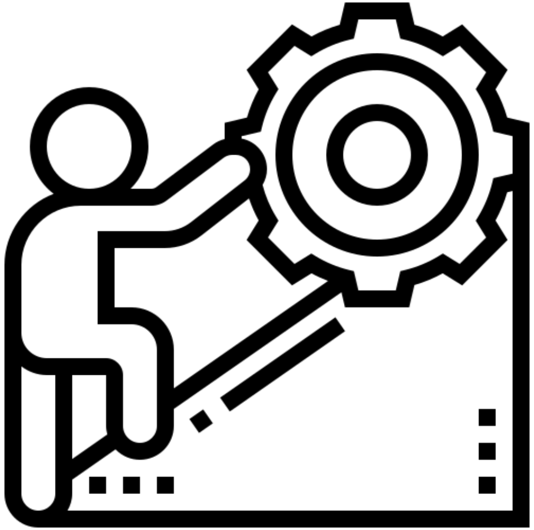
Agile Testing Benefits

- Early testing form initial phases
- Cohesion between developers and testers means that quality is at the forefront of everyone’s minds
- Lightweight documentation - focus is not on writing multiple test plans, test cases, test reports, bug reports, but on making sure that the solution is of high quality


Accelerated Time to Market
Reduces Documentation Efforts
Agile Testing Benefits
Introduces Automation Testing
Takes Testing Estimates into Consideration

Agile Testing Challenges

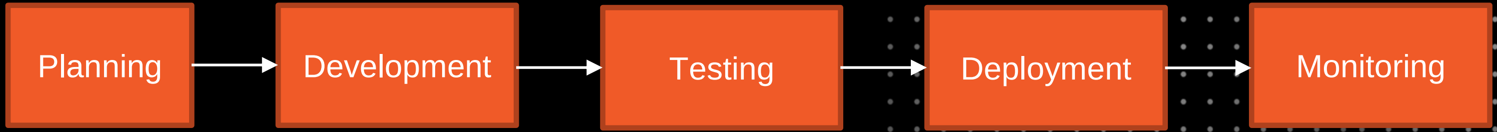
- Insignificant/Incomplete Information
- Continuous Testing
- Slow Feedback Loop
- Frequently Changing Requirements
- Unclear Quality Measurement Criteria
- Performance Bottlenecks



20 minutes

Activity – Change your process; change your quality

Part 1: Visualize your current software development lifecycle (SDLC) using a pen and paper or diagramming tool. Try to capture each of the activities in your SDLC and how they flow into one another. Here is an example diagram to help you get started:



Planning → Development → Testing → Deployment → Monitoring

Part 2 : Next, create a second diagram that would describe your 'dream' SDLC flow where the whole team has ownership of quality. You can add new activities, add responsibilities or re-arrange your SDLC process.

Part 3 : Comparing your two SDLC processes, write down three small experiments/ideas that you could try out as a team to help nudge you towards whole team ownership of quality.

Software Testing Tools and Techniques

Manual Testing

- In manual testing , testers manually execute test cases without using any automation tools.
- It requires a tester to play the role of an end user.
- Any new application must be manually tested before its testing can be automated.
- Manual testing requires more effort, but is necessary to check automation feasibility.
- Manual testing does not require knowledge of any testing tool.
- One key software testing principle is ***100% Automation is not possible*** making manual testing imperative and important.

Advantages of Manual Testing

- It is preferable for products with short life cycles.
- It is preferable for products that have GUIs that constantly change.
- It requires less time and expense to begin productive manual testing.
- Automation can not replace human intuition, inference, and inductive reasoning- *yet*.
- Automation can not change course in the middle of a test run to examine something that had not been previously considered.
- Automation tests are more easily fooled than human testers - *yet*.

Limitations of Manual Testing

- Requires more time or more resources, sometimes both - especially for long running projects.
- Performance testing is impractical with manual testing.
- Less Accurate.
- Executing same tests again and again is time consuming and tedious.
- Not Suitable for Large scale projects and time critical projects.
- Batch Testing is not possible as for every test execution human user interaction is mandatory.
- Scope of manual test cases is very limited.
- Comparing large amounts of data is impractical.
- Checking relevance of search of operation is difficult.
- Processing change requests during software maintenance takes more time.

Why Automated Testing?

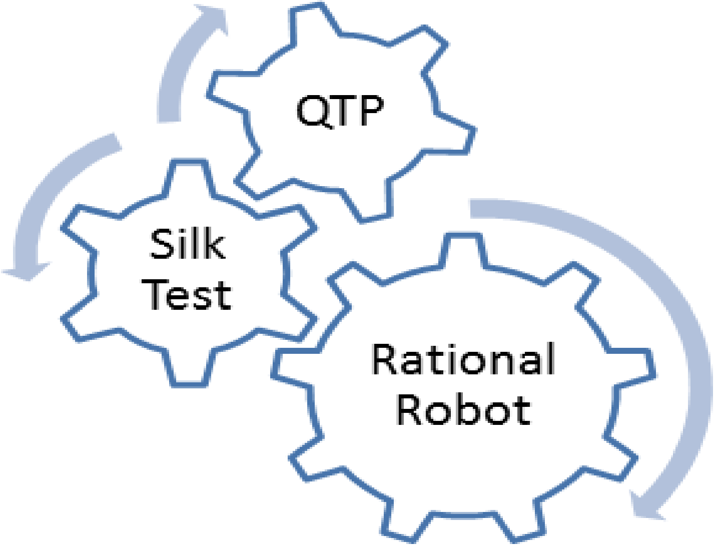
- Challenging to test multilingual sites/interfaces manually.
- Does not require human intervention.
- Increases speed of test execution.
- Increase test coverage exponentially.
- Manual testing can become boring for testers and hence is error prone.

Automation Testing vs. Manual Testing

Automation Testing	Manual Testing
Perform **same operation** each time	Test execution is **not accurate all the time,** hence not reliable
Useful to execute a set of test cases **frequently**	Useful when the test case only needs to run **once or twice**
Fewer testers required to run test cases	**Large number** of testers required
Platform **independent**	Platform **dependent**
Testers can **fetch complicated information** from code	Does **not involve** any programming task to **fetch hidden information**
Faster	**Slower**
Not helpful in testing UI	Helpful in testing UI
High cost	**Less** than automation

Automation tools

Some of the most popular test tools



QTP (Quick Test Professional)

- HP's Quick Test Professional (now known as HP Functional Test) is the *market leader* in Functional Testing Tool space.
- It supports keyword driven testing.
- Suitable for both client/server and web-based applications.
- QTP has better error handling mechanism than most tools.
- It offers excellent data driven testing features.

Rational Robot

- Rational Robot is a complete set of components for automating the testing of Microsoft Windows client/server and internet applications.
- The main component lets you start recording tests in as few as two mouse clicks.
- After recording, Robot can play back (run) the tests in a fraction of the time it would take to repeat the actions manually.
- Enables defect detection and includes test cases and test management.
- It supports multiple UI technologies.

Selenium

- It is an open-source portable software testing framework for web applications.
- It provides a record/playback tool for authoring tests without learning a test scripting language using **Selenium IDE**.
- It supports writing tests in a test domain-specific language - including in C#, Groovy, Java, Perl, PHP, Python, and Ruby.
- The tests can then be run against most modern web browsers.
- It can be deployed on all major platforms - Windows, Linux, and Macintosh.

SilkTest

- SilkTest is an automated function and regression testing tool for enterprise applications.
- It is used for testing e-business applications.
- It offers test planning and management, direct database access and validation support.
- It supports extensions for .NET, Java (Swing and SWT), DOM, IE, Firefox, and SAP Windows GUI.

Benefits of Automated Testing

- **Reliable:** Tests perform precisely the same operations each time they are run, thereby eliminating human error.
- **Repeatable:** You can test how the software reacts under repeated execution of the same operations.
- **Programmable**: You can program sophisticated tests that bring out hidden information from the application.
- **Comprehensive:** You can build a suite of tests that covers every feature in your application.
- **Reusable:** You can reuse tests on different versions of an application, even if the user interface changes.
- **Fast:** Automated tools run tests significantly faster than human users - approx. 70% faster than manual testing!
- **Better Quality Software:** Because you can run more tests in less time with fewer resources
- **Cost Reduction:** As the number of resources for regression test are reduced.

Automated test tool features

- Essential
- Highly Desirable
- Nice to Have

Essential

- The ability to divide the script into a small number (3 or 4) of repeatable modules.
- Supports the use of multiple data sheets/tables (at least 8).
- The ability to store objects names in data tables and refer to them in the script.
- The ability to treat the contents of different cells in data sheets as input data, output data, windows, objects, functions, commands, URLs, executable paths, commands, etc.
- The ability to access any data sheet from any module.

Highly Desirable

- The ability to recover from Severity 1 (fatal) errors and still continue to the end.
- The ability for the users to create custom functions.
- The ability to write directly into the results report.
- The ability to write into the data table (using a script).

Nice to Have:

- Direct interface to the test management system (bi-directional).
- The ability to add comments to data table (rows).
- The ability to restrict output to the results file .

Static Test Tools

- Static analysis tools are generally used by developers as part of the development and component testing process.
- They can help developers to understand the structure of the code.
- They can also be used to enforce coding standards.
- These tools do not involve actual input and output.
- Static analysis tools are an extension of compiler technology.

Features for Selecting Static Test Tools

- Assessment of the organization’s maturity (e.g. readiness for change).
- Identification of the areas where tool support will help improve testing processes.
- Evaluation of tools against clear requirements and objective criteria.
- Evaluation of the vendor (training, support and other commercial aspects) or open-source network of support.
- Identifying and planning internal implementation team (including coaching and mentoring for those new to the use of the tool).

Dynamic analysis tools

- ***Dynamic*** because they require the code to be in a **running state**.
- They analyze what is happening ***behind the scenes*** - that is in the process and its runtime env. while the software is running.
- These tools are used by developers in component testing and component integration testing.
- E.g.: Testing of middleware and security policies, or when looking for robustness defects.

Features of Dynamic Test Tools

- Detect memory leaks.
- Identify pointer arithmetic errors such as null pointers.
- Identify time dependencies.
- Test the software system with 'live' data.

What is a Testing Framework?

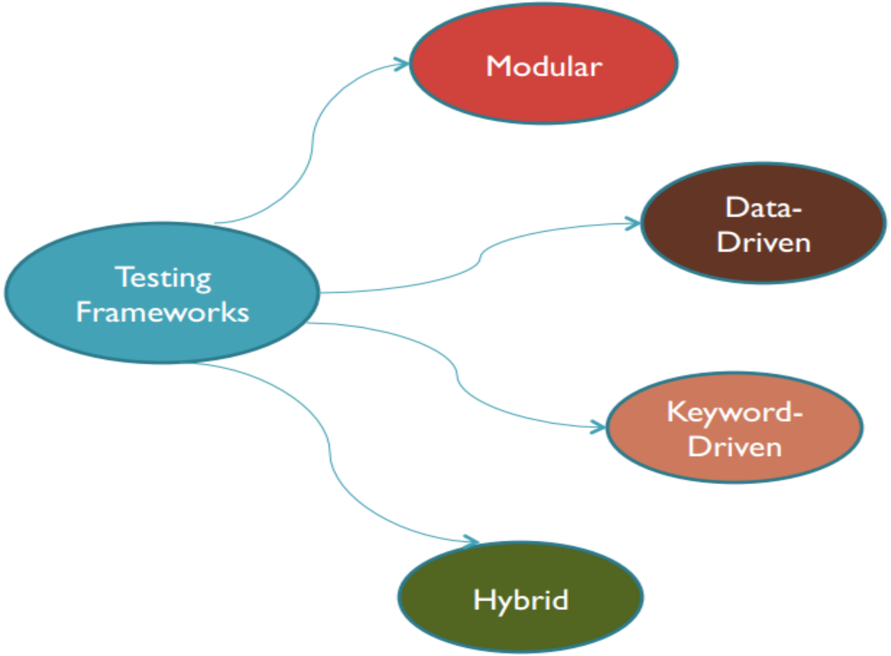
- Testing automation framework is an execution environment for automated tests.
- It is the overall system in which the tests will be automated.
- It is defined as the set of assumptions, concepts, and practices that constitute a work platform or support for automated testing.
- It is application independent.
- It is easy to expand, maintain and perpetuate.

Why we need a Testing Framework

Suppose if each project implements a unique strategy then the time needed for a tester to become productive in the new environment will take long. To handle this we cannot make changes to the automation environment for each new application that comes along.

- A testing framework can be of particular help here as they are application independent and have the capability to expand with the requirements of each application.
- A testing framework helps avoid duplication of automated test cases across the application.
- They also help teams organize their test suites and improve the testing efficiency.

Types Of Testing Frameworks



Testing Frameworks

Modular

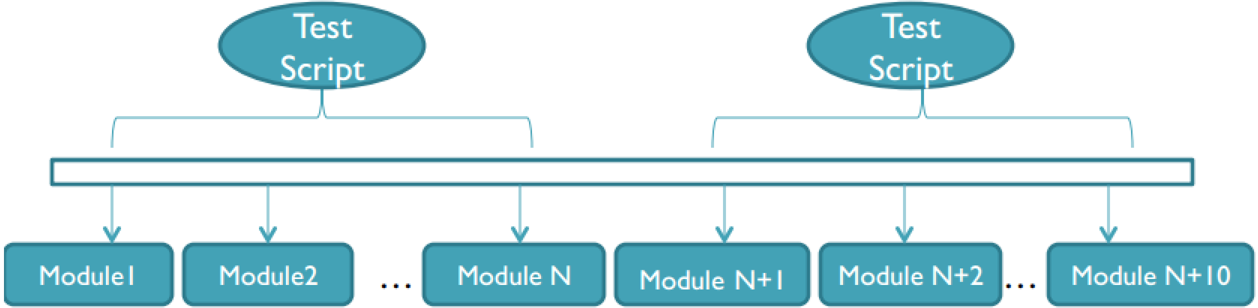
Data-Driven

Keyword-Driven

Hybrid

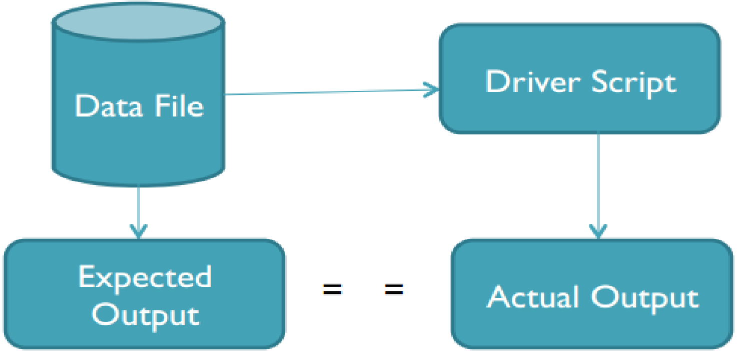
Modular Testing Framework

- The Modular testing framework is built on the concept of abstraction.
- Creates independent scripts representing the modules of the application under test.



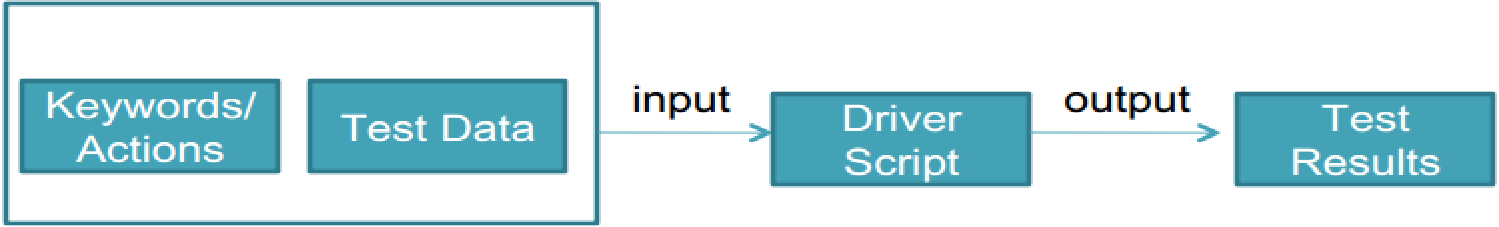
Data-Driven Testing Framework

- Data driven testing is where the test input and the expected output results are stored in a separate data file (normally in a tabular format) so that a single driver script can execute all the test cases with multiple sets of data.
- The driver script contains navigation through the program, reading of the data files and logging of the test status information.



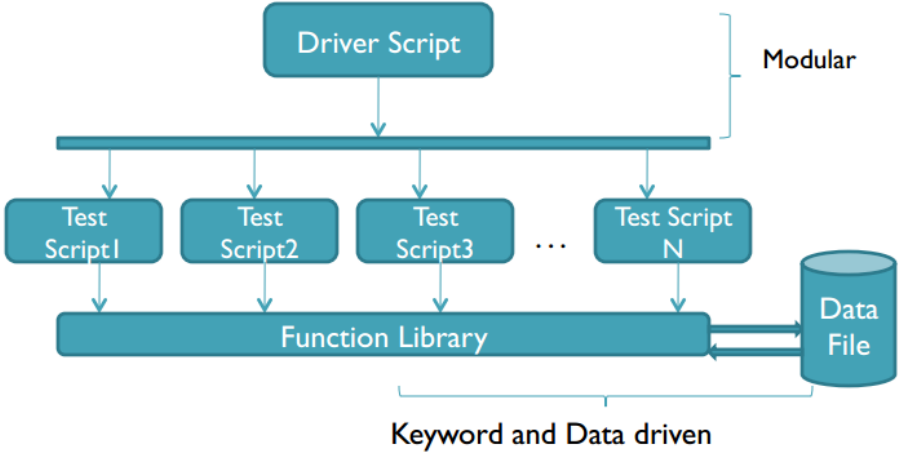
Keyword- Driven Testing Framework

- Keyword driven testing is an application independent framework utilizing data tables and self explanatory keywords to explain the actions to be performed on the application under test.
- The test data kept in an external file as are the directive test scripts.
- These directives are called keywords.



Hybrid Testing Framework

- Hybrid testing framework is the combination of modular, data-driven, and keyword-driven testing frameworks.
- It helps the data driven scripts take advantage of the libraries mostly used in keyword driven testing.



Thank you!

DevelopIntelligence
A PLURALSIGHT COMPANY