Cognizant

Wirearchy Session

7th May 2019

Cognizant

Welcome to QEA Wirearchy Session - 2

Thanks for joining us

Cognizant

Zero touch QA automation Framework

D&B

Varadharajan Srinivasan

Suhardhini Kannapiran

Abstract

With the industry transformation to Digital Trend, there is always a need for an efficient streamlined Continuous Delivery. We would talk about Zero Touch QA Automation Platform in this session. It deals with the Solution on Integrated DevOps & Quality Approach. This would consist of Artificial Intelligence in Reporting and Data Visualization. The Complete QA Cycle is automated from Test Design to Test Closure with Continuous Monitoring, involving no manual intervention. This would also give insights on Business benefits/outcome on adopting this approach with streamlined Governance.

Top 5 Challenges Faced

1. Need simplified automation approach to enable functional testers with test automation
2. Need manual effort reduction in end-to-end test process (test design, test results update, defect creation, end to end traceability, summary report etc.)
3. Need ability to automatically analyze the test failures
4. Need ability to focus on 360 degree Quality Assurance
5. Need solution to measure Code coverage

Cognizant

Challenge 1: Need to simplify Test Automation

AGILE ADOPTION

- Expected to have more frequent releases
- Current Team have less automation expertise
- Go beyond UI test to API and data testing
- Need to automate regression / progression test cases

AND NO COMPROMISE ON PRODUCT QUALITY

CHALLENGE

- Stress endures handling knowledge upskilling and project deadlines in parallel
- Automation Enablement for entire team is time taking process
- Testing needs to go away from only E2E to Component, API to reduce cycle time

Agile Transformation demands

Layered | High adoption | Simplified

Challenge 2 - Need to automate the E2E QA Ecosystem

Skip QA Process…?

Impacts

- No E2E Traceability
- lack of documentation
- loss of knowhow
- Lack of Quality Control
- Low Defect Containment efficiency
- No view on project health

Spend hours on the QA Process…?

Impacts

- Considerable manual effort
- Prolonged Time to Market
- Not suitable in Agile delivery models

Impacts Outweigh

Requirement Analysis
↓
Test Design
↓
ALM Update
↓
Test Scripts Development
↓
Test Execution
↓
Test Logs
↓
Test Reporting
↓
Test Closure

Focus not just on Test Execution Automation

Efficient and Contin Testing Agility possible through Fully automated QA Process

Cognizant

Challenge 3 - Simplify Test Analysis & Reporting

Challenges encountered without an efficient Failure Analysis & Reporting Mechanism

Huge Effort spent!

Multiple test runs a day & continuous investigation of failures leading to considerable effort spend

Delayed Feedback!

Unproductive days due to bad builds and test failures caused by environmental issues

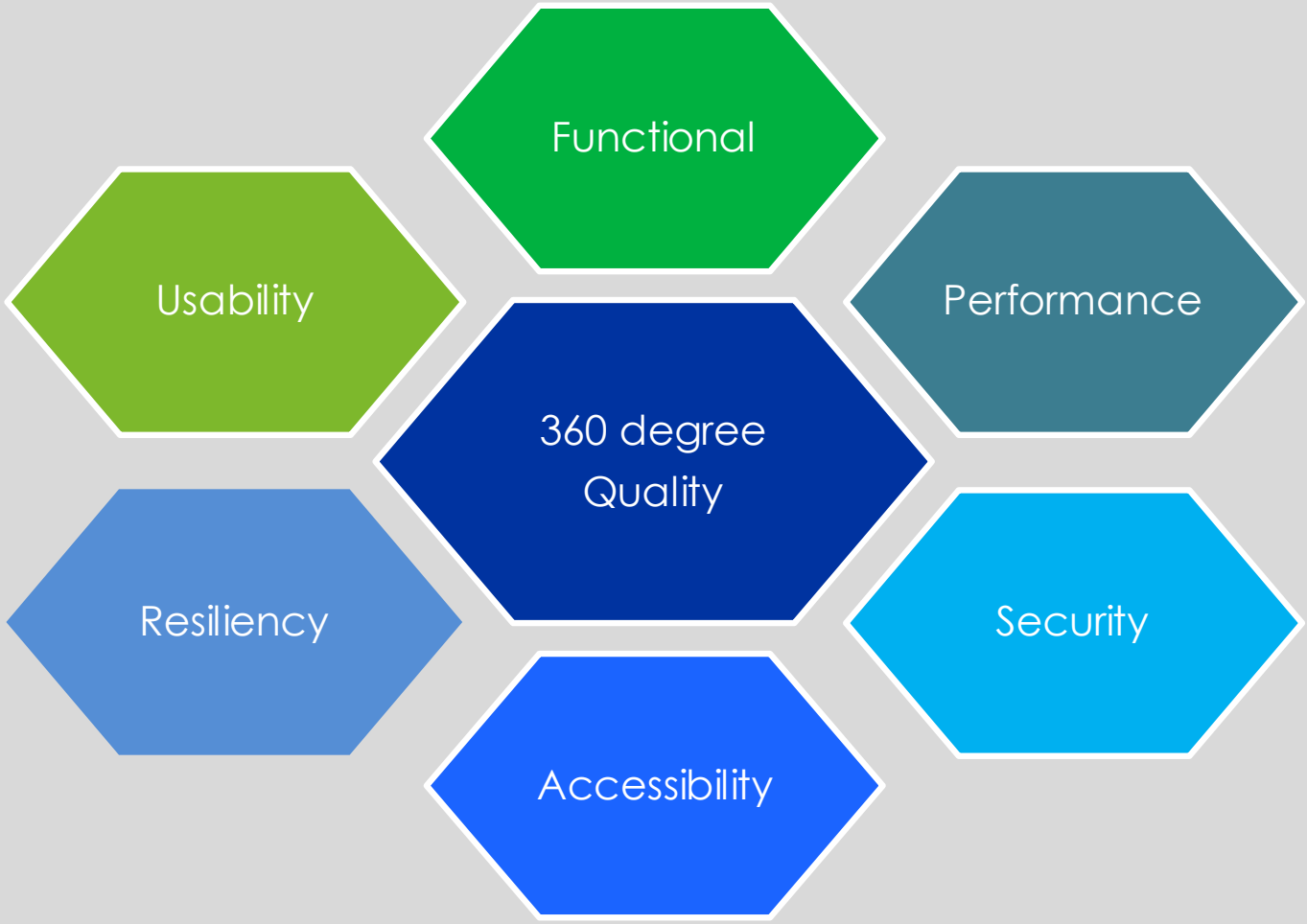
Costly & Complicated Adoption!

Challenge in adapting to complicated tools amidst project deadlines

No Transparency on Project Health

Cumbersome to monitor and maintain KPIs in multiple forum

Challenge 4 – 360 degree Quality Assurance



Challenge

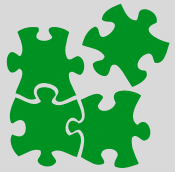
Imperative to assure on all these factors for improved quality and enriched customer experience

Important to have an early feedback and deliver at speed in a digitally assured business

Challenge 5 - Need solution to measure Code coverage

Facilitating faster Stabilization of final release ?

Focusing on high value tests?



Enough testing in place ?

Conventional requirements-based testing of code tends to result in considerable over-testing with no guarantee of complete coverage

Redundant & non-productive	**There are too many test cases which test the same code path**
Incomplete & un-tested	**A tester is not aware of the areas that have not been tested**

Impact *- This typically results in additional QA cost and extended testing timelines which in turn impact project success*

Deep Dive

ZERO TOUCH QA – A Vision for Digital IT Industry

Adopt next-generation *testing* practices to test early, often, automatically and continuously

Zero Touch QA Platform

JUnit
Code Commit Status update

PMD

Junit Coverage

JIRA

sonarqube

Sonar, JUnit (Static Analysis, Code Coverage)

Requirements

Eclipse

Version Control GIT/Stash

Build
Jenkins, Maven, Artifact

Deployment
Jenkins

Jenkins

CI

Commit code

Continuous Testing / Continuous Analysis / Continuous Reporting

- When the code is committed in GIT, the development Jenkins job gets automatically triggered
- Development Jenkins job triggers the following test in QA
  - Functional Test
  - Performance test
  - Sea light Test
  - Automatic creation of test cases

Auto QA Work Flow

Sealights

FORTIFY

APACHE JMeter™

1 Karate

Production Release

UAT/Stakeholder's Demo

Quality Intelligence Test

Automatic Trigger of Security Test

Automatic Trigger of QA Performance Test

Automatic Trigger of QA Functional Test

Automated validation

5

4

3

2

Automatic upload of results in Confluence

AUTO Analysis of Failures

Automatic creation of Test Cases/Defects in JIRA

Confluence

ZEPHYR for JIRA

Cognizant

1 Automated Functional Testing: Simplification/ Script less Automation

Challenge

01 Getting the existing team to upskill in programming language to automate

02 Meeting the current project deliverables and upskilling puts stress on the testers

03 To retain the domain knowledge that enables quality product delivery

Solution

Simplify the various types of automation using a BDD construct

Example: Karate for API test

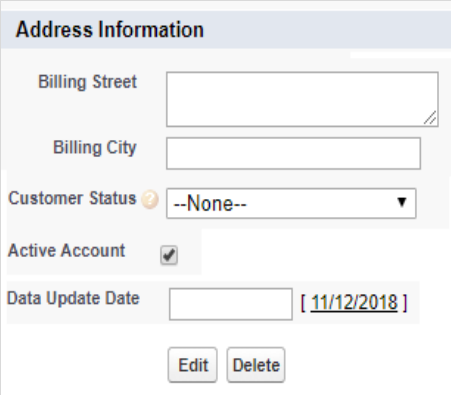
Benefits

Enabled 90% of existing team for progression and regression automation

Reduced test automation effort by 60%

1 Automated Functional Testing: UI Test Simplification

1. Identify Patterns → 2. Review and Align with UI Developers → 3. Abstract with BDD Construct → 4. Publish Language of Automation



3. Abstract with BDD Construct (Scenario)

Then I enter the text {103 John F Kennedy Pkwy, STE 234} into {Billing Street} Text Area

Then I enter the text {New York} into {Billing City} Text Box

Then I select the drop down value as {Available} in {Customer Status}

Then I select the {Active Account} checkbox

Then I select the {Current Date} date for {Data Update Date} field

Then I click on the {Edit} Button

Control	1. Identify Patten (Generic XPATH)
Text Area	//label[text()="'+ LABEL +'"]/following::td[1]/textarea
Text Box	//label[text()="'+ LABEL +'"]/following::input[1]
Check Box	//label[text()="'+ LABEL +'"]/following::input[1]
Date Field	//label[text()="'+ LABEL +'"]/following::td[1]//span[@class='dateFormat']
Button	//input[@value="'+ LABEL +'"]
Drop Down	//label[text()="'+ LABEL +'"]//following::td[1]//select

Cognizant

1 Automated Functional Testing: API Test Simplification

1. Identify Patterns → 2. Review and Align with UI Developers → 3. Abstract with BDD Construct → 4. Publish Language of Automation

Karate Framework Integration with BDD

```
Scenario: create and retrieve a cat

Given url 'http://myhost.com/v1/cats'
And request { name: 'Billie' }
When method post
Then status 201
And match response == { id: '#notnull', name: 'Billie' }

Given path response.id
When method get
Then status 200
```

JSON is 'native' to the syntax

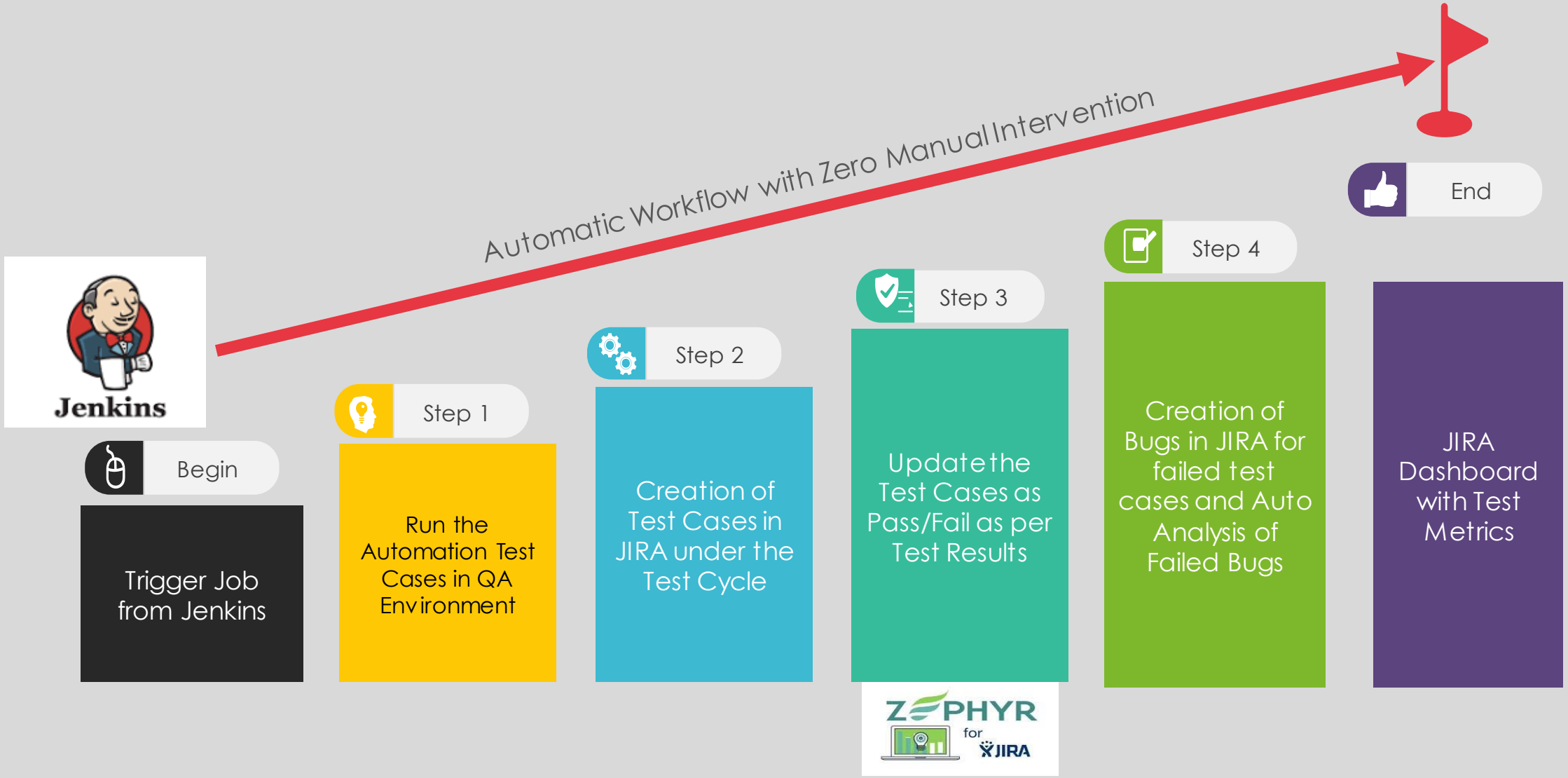
Intuitive DSL for HTTP

Payload assertion in one line

Second HTTP call using response data

Cognizant

2 Automatic creation of Test Cases/Defects: Automate E2E QA Lifecycle Process

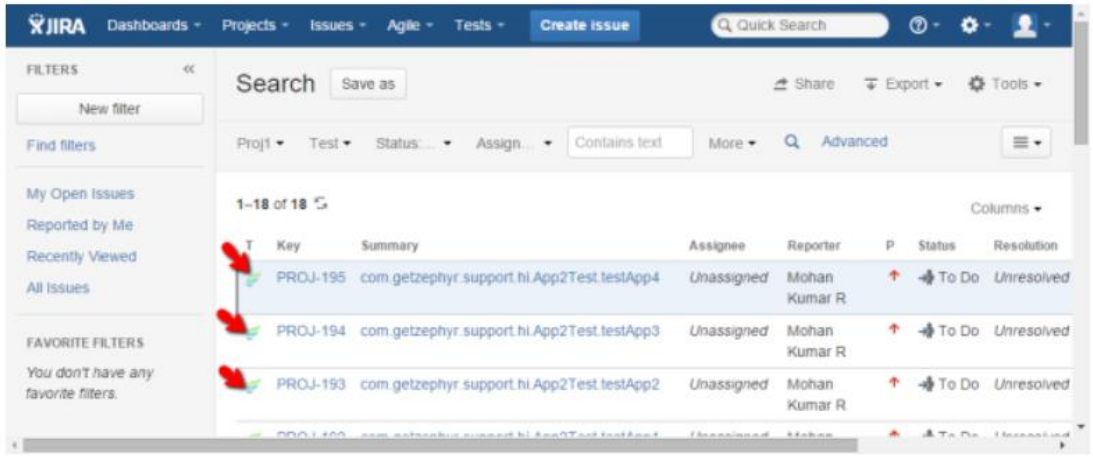


Cognizant

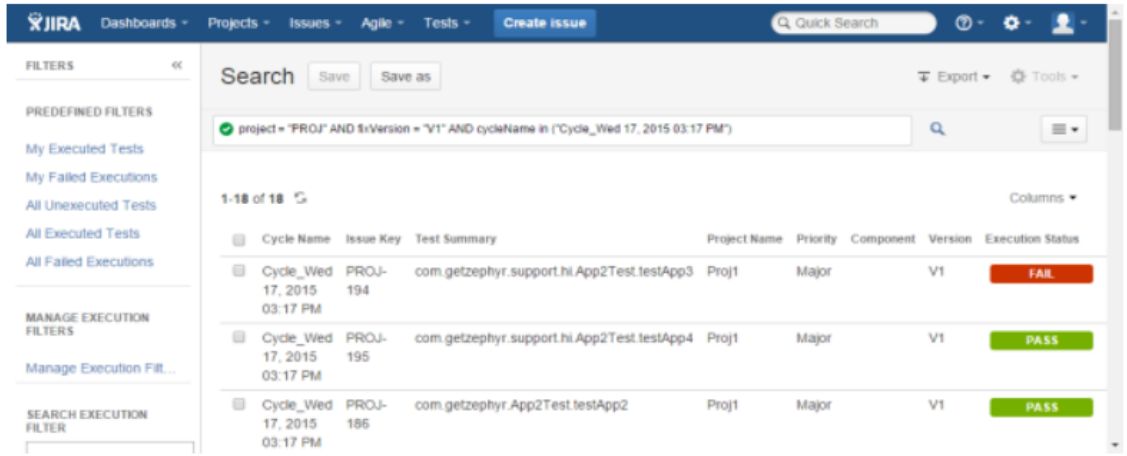
2 Automatic creation of Test Cases/Defects: Zephyr JIRA Integration

AUTOMATED TEST CASE

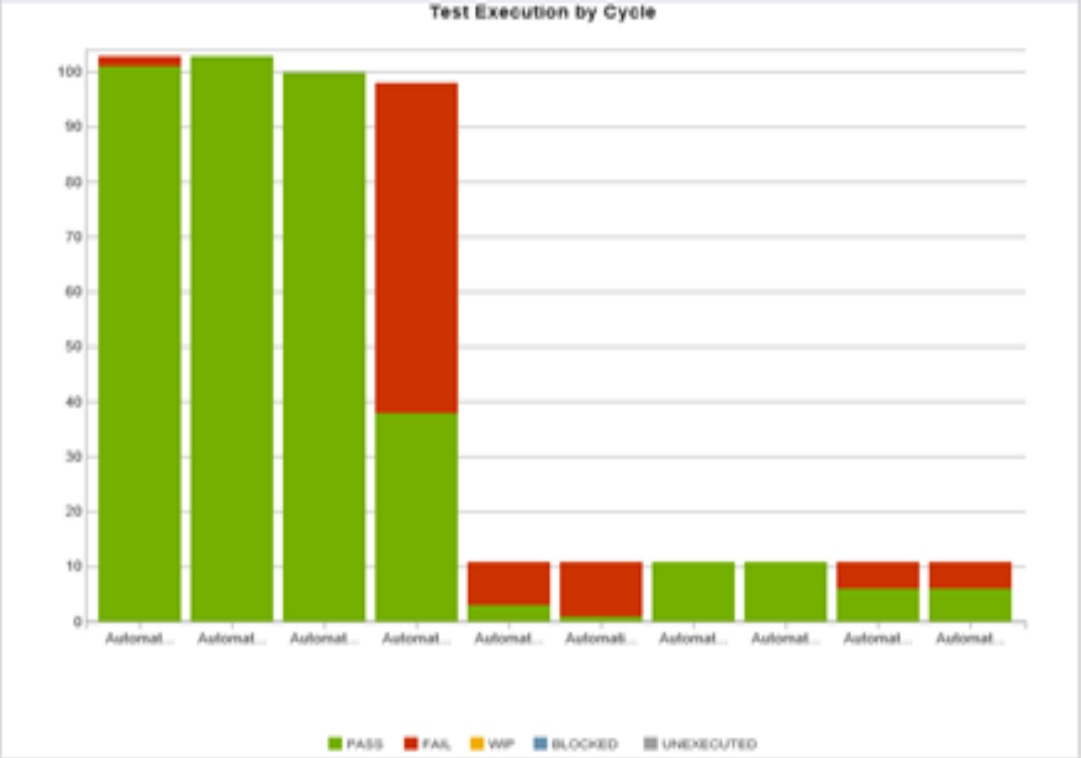
Jenkins creates the test cases in the selected JIRA Project.



Finally, Jenkins assigns these test cases to the selected cycle and executes all the tests.



AUTOMATED STATUS AND DEFECTS



2 Automatic creation of Test Cases/Defects: JIRA Serenity Integration

Activity

All | **Comments** | Work Log | History | Activity

Bruce Wayne added a comment - 12 hours ago - edited

Thucydides Test Results

Test report

- Display flight details to featured destinations: SUCCESS :
- Display flight details with lookahead: ERROR :
- Display flight durations: IGNORED :
- Display flight for all available flights: ERROR :
- Display international flight details by flight number: SUCCESS :
- Display русский flight details by flight number: SUCCESS :
- Display русский international flight details by flight number: SUCCESS :
- Should display flight durations: SUCCESS :
- Should display flights to featured destinations: SUCCESS :

JIRA report with Serenity comments

3 Auto Analysis of failures: AI powered Test Automation Dashboard with Real-Time Analytics

Manage all your automation results and reports in one place

Make automation results analysis actionable & collaborative

Establish fast traceability with defect management

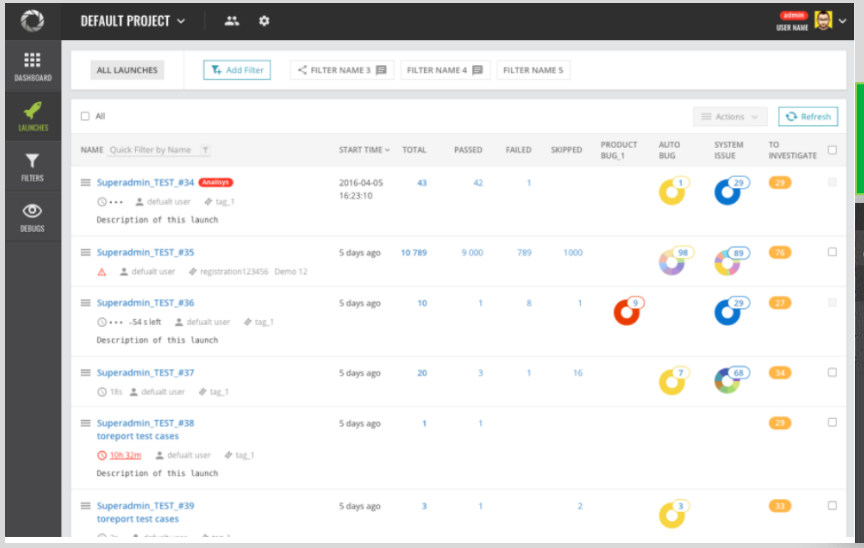
Accelerate routine results analysis

Visualize metrics and analytics

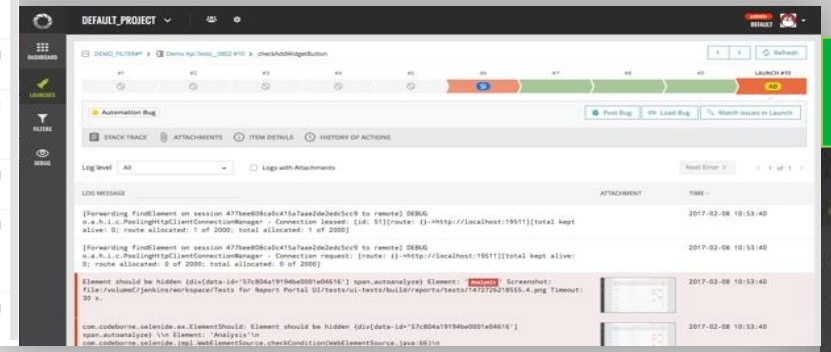
Make smarter decisions together

3 Auto Analysis of failures: AI powered Test Automation Dashboard with Real-Time Analytics

AL BASED ANALYSIS



FAILURE TRACKING / MANAGEMENT



CENTRAL REPOSITORY FOR INTEGRATED DASHBOARD AND REPORTS

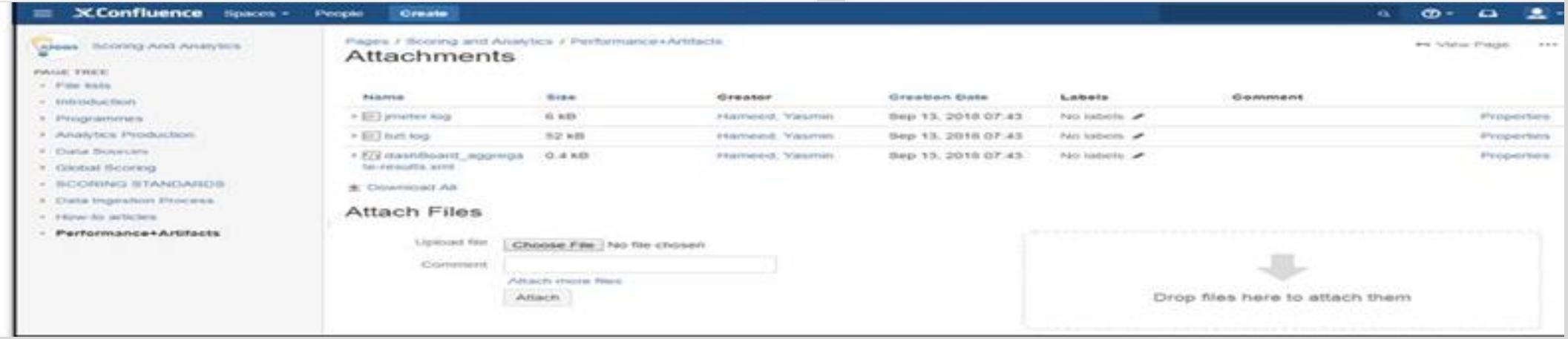
4 Automatic trigger of NFT and auto upload of test results: 360 degree Quality Assurance

Features:

- Perform Performance Testing using Jmeter, integrated with Jenkins
- Perform Security Testing using Fortify, integrated with Jenkins
- Upload the Results in Confluence automatically

Benefits:

- Transparent visibility of test results
- 70% of effort savings in Test Closure activities



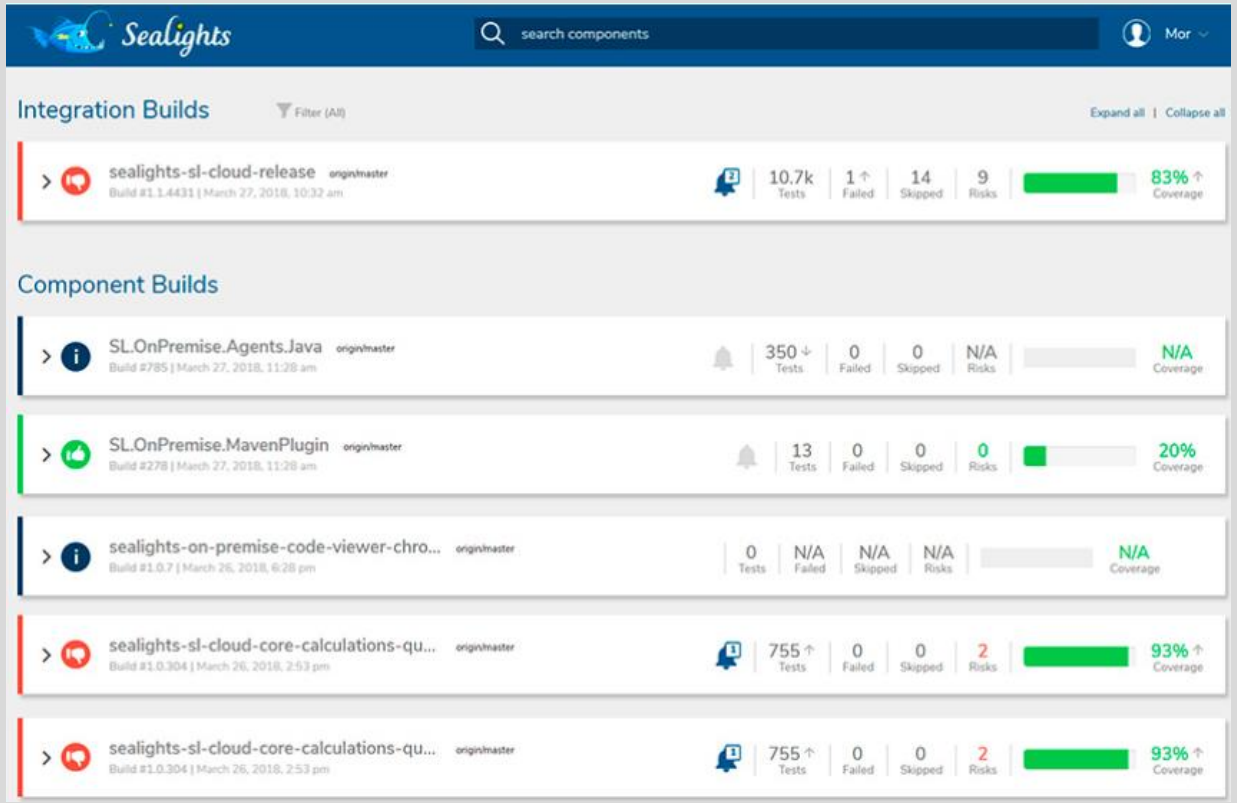
5 Quality Intelligence Test with Sealights tool: Automatic Code Coverage Analysis

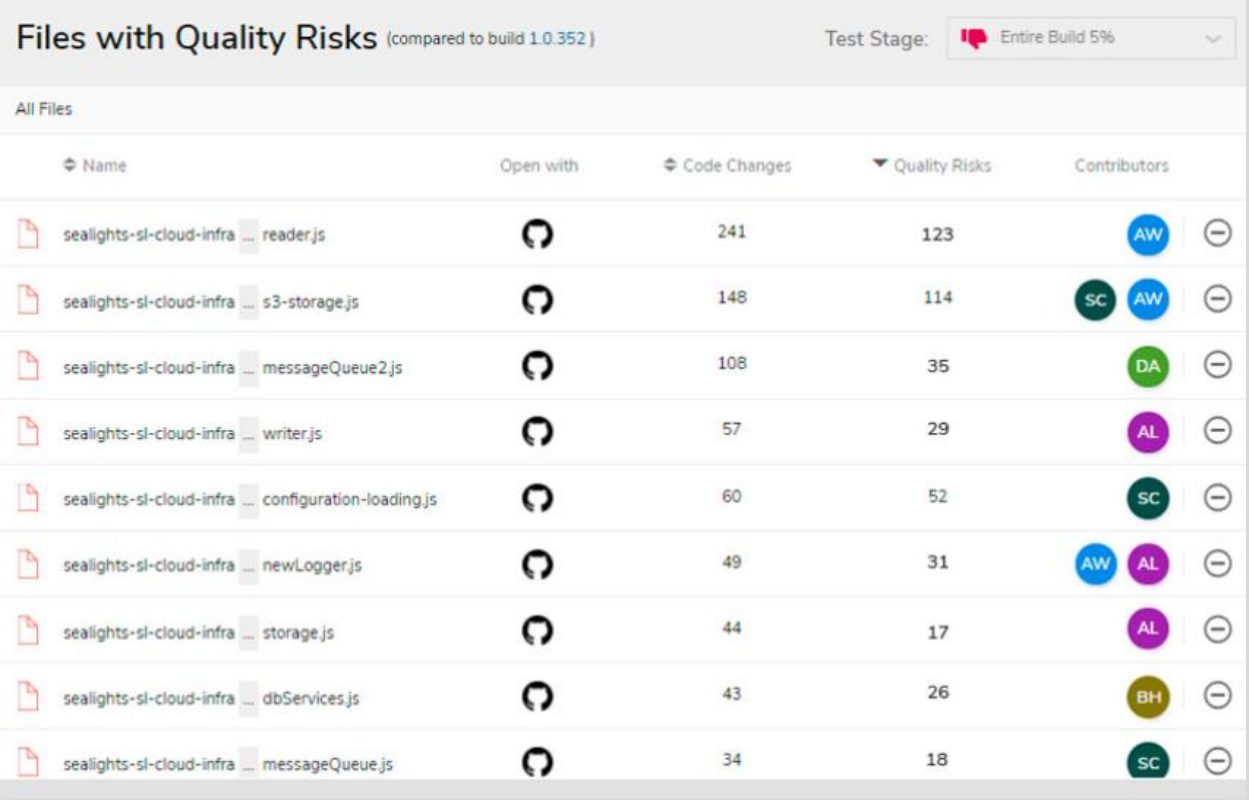
Features:

- Test Gap Analytics
- Release Quality Analytics
- Block Untested Code Changes
- Utilize Test Impact Analytics to Test Smarter

Benefits:

- 100% Improvement in Code Coverage for Test Cases
- Test Quality Analytics to Test More with "Less Tests Approach"





Cognizant

For further updates and discussions, please contact

Varadharajan Srinivasan (108294)

Suhardhini Kannapiran (120933)

Cognizant

Cognizant

Automation in BDD Way

AIG

Sandip Agarwala

Jishan Ali Mondal

Cognizant

Automation in BDD Way

API Framework KARATE

UI Framework Gherkin

Presenter Sandip Agarwala
Jishan Ali Mondal

DATE 05/07/2019

Agenda

GHERKIN AUTOMATION APPROACH
Details on the gherkin approach and areas of implementation

DEVELOPMENT USING RTEEe
Gherkin based feature files, Step definition, POM

KARATE FRAMEWORK
Framework features, examples

BENEFITS
Details on the Achievements & Benefits

Cognizant

Gherkin Automation Approach

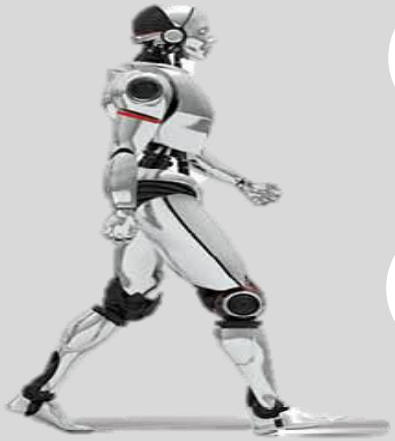
Conventional Testing (No Automation)

Selenium Automation

BDD + Gherkin UI framework

Gherkin UI + KARATE Framework

CICD + DevOps

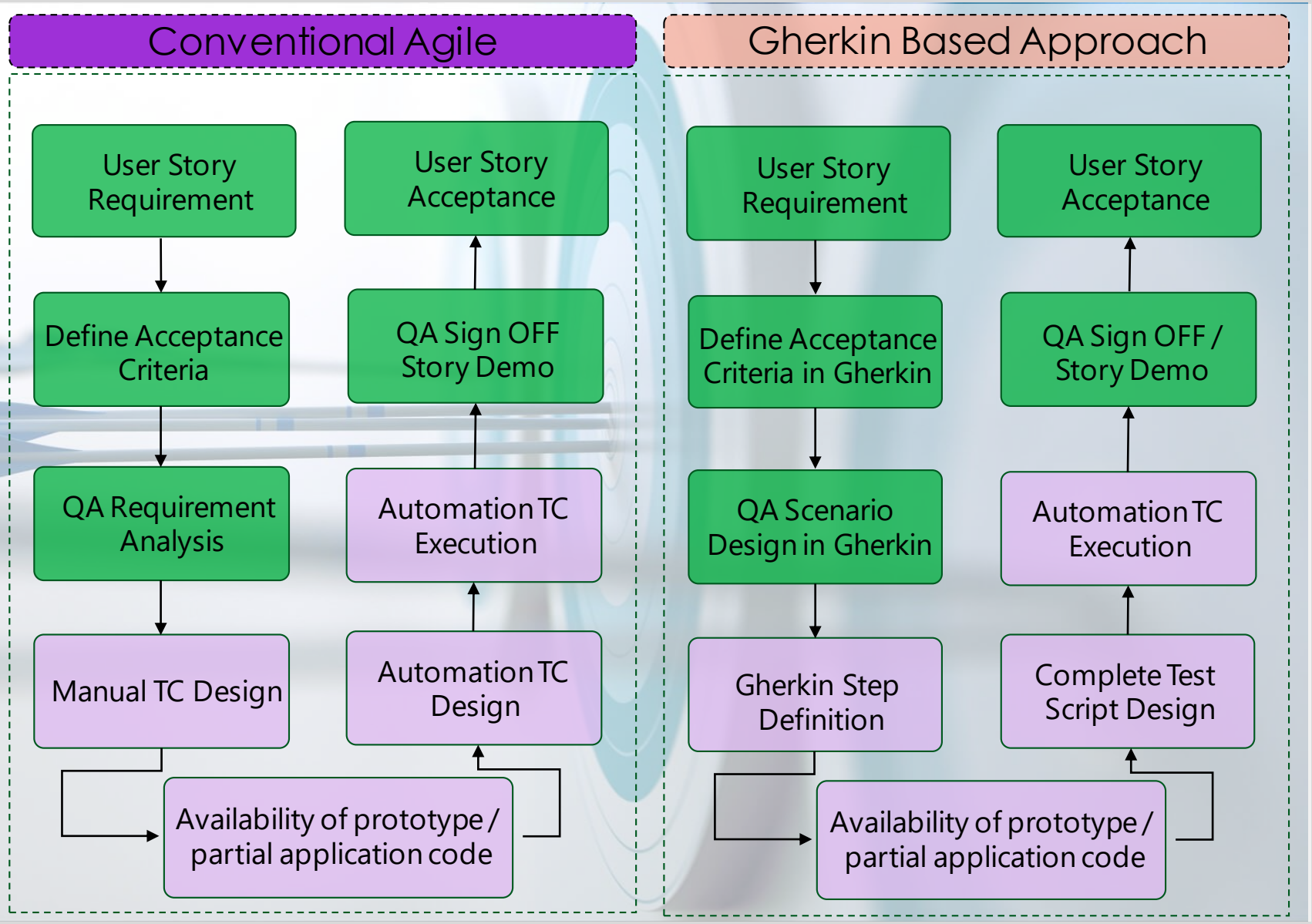


UI Regression Automation	UI In-sprint Automation	API Automation	API End-to-End	Accessibility Test Automation	Performance Automation
Optimization of existing regression test cases following gherkin based automation	Enablement of in-sprint automation by implementing gherkin based approach throughout SDLC	API regression and in-sprint automation following KARATE framework	Chained End-to-end journey scenario [Ex: client login using multi-factor authentication]	Enhancement of existing framework for automated accessibility testing using AXE libraries	Gatling Integration with KARATE for API performance Testing.

Gherkin Automation Approach - Continued

FEATURES

- Participation of three Amigos (BA, DEV, Tester) while defining acceptance criteria
- Minimum effort in QA scenario design as the high level scenarios are already defined
- Same acceptance criteria can be extended while designing QA scenarios
- Zero effort in converting manual steps into automation
- Having lesser dependency on prototype as the steps can be defined separately and will be automatically executed
- Zero rejection during demo as the same acceptance criteria being used while designing tests
- Better responsiveness to CR's as step level changes automatically gets applied for all scenarios



Cognizant

Gherkin Framework - Overview

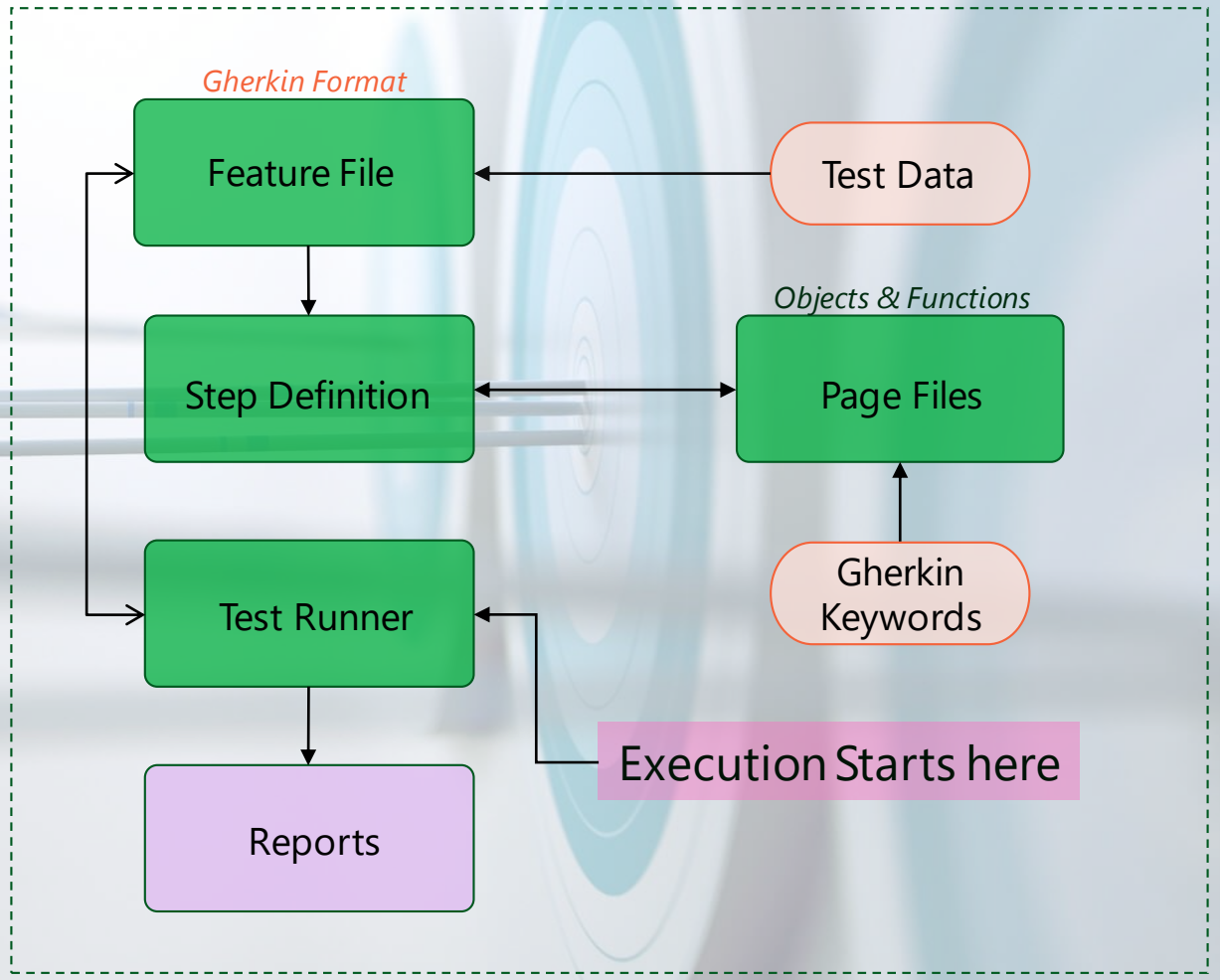
PRE-REQUISITES

- Java 8 or higher
- IntelliJ IDE
- Cucumber Plugin
- Git.exe
- Maven

FEATURES

- Scenario design in Gherkin Format
- Test data maintained in scenarios
- Ready keyword library for script development
- Page object model for better script handling
- Reporting using Extent Reports

ARCHITECTURE



Cognizant

Development using Gherkin

WHAT IS GWT

Given
<Preconditions and/or Inputs>
When
<The action under test>
Then
<The expected outcome or verification>

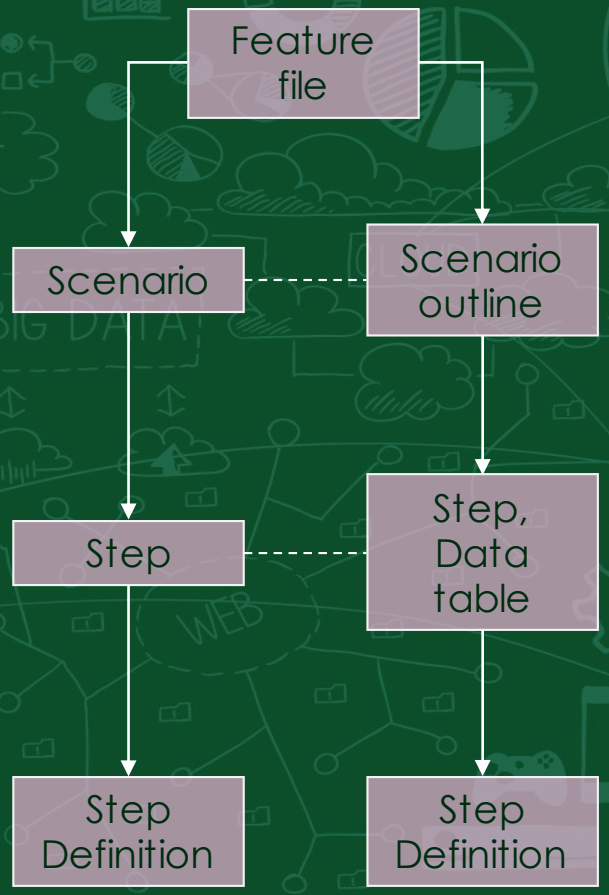
COMPONENTS

- Background
- Scenario
- Scenario Outline
- Step
- Step Definition
- Tagging

PARAMETERIZATION

- Parameterization can be done at scenario level as well as step level

STRUCTURE OF FEATURE FILE




```
Feature: Instant messanger feature
    As a online user
    I want to chat using Instant messanger
    So that I can chat with my friends

Scenario: Verify friends are online in Instant messenger (IM)
    Given IM is available in ABC website
    When I log into ABC social website
    And I open up the IM tab
    Then I should see my online friends

Scenario Outline: Verify friends availability status
    Given IM is available in ABC website
    And <User> exists and logged in
    When I log into ABC social website
    And I open up the IM tab
    Then I should see <User> online
    Examples:
    | User  |
    | Rakesh |
    | Sam   |
    | Sushil |
    | Vijay  |
```


Cognizant

Development using Gherkin – Continued

Built in - RTEEe KEYWORDS

Keywords are reusable methods designed to perform actions on elements present in the application, like –

- Click
- Select dropdown value
- Enter text
- Check checkbox
- Validate text
- Validate element's presence
- etc.

Keywords are used in sequence to form the body of the step definition directly or to create page method.

FORMATION OF STEP DEFINITION/PAGE METHOD USING KEYWORDS

Step

```
When I log into ABC social website
```

Step Definition

```
@When("I log into ABC social website")
public void
ILogIntoABCSocialWebsite(String URL,
String UserName, String Password)
{
  inVokeApp(app url: "URL");
  setValue(object: UserName,value:
"user");
  setValue(object: Password,value:
"efsdffgf");
  clickElement(object: LoginButton);
}
```

KEYWORDS FORMING STEP DEFINITION

KEYWORDS FORMING PAGE METHOD

Step

```
When I log into ABC social website
```

Step Definition

```
@When("I log into ABC social website")
public void ILogIntoABCSocialWebsite()
{
  inject.getPage(LoginPage.class)
              . LogIntoABCSocialWebsite(URL,"user","efsdffgf")
}
```

Page Method

```
public class LoginPage() {
 public void LogIntoABCSocialWebsite(String URL, String UserName,
String Password)
 {
  inVokeApp(app url: "URL");
  setValue(object: UserName,value: "asdsd");
  setValue(object: Password,value: "efsdffgf");
  clickElement(object: LoginButton);
 }
}
```

Development using Gherkin – Continued

PAGE OBJECT MODEL CONTAINS

- Element locators like
  - Id
  - Name
  - Class
  - Link text
  - Partial link text
  - Tag name
  - Xpath
- Methods performing operations on elements

LOCATOR FORMAT

<Access modifier> By <Variable name> = By.<locator type>("<Value>")

METHOD FORMAT

<Access modifier> <Return type> <Method name> (<Variable Type> <Value>) {

<Method body goes here>

}

SAMPLE PAGE OBJECT MODEL

```
package [illegible] test.cucumber.Repository.Pages;

import [illegible] test.core.element.custom.customElementsFunctions;
import [illegible] test.cucumber.stepDefinition.Injection;
import org.openqa.selenium.By;

public class ValicSelectClientPage extends customElementsFunctions {
    //    ************************************************************Objects*************************************************
    private By SSN_Textbox = By.id("[illegible]");
    private By Continue_Button= By.id("[illegible]");
    //    ************************************************************************************************************************
    Injection inject;
    public [illegible]SelectClientPage(Injection inject) { super(inject); }

    public [illegible]SelectClientPage clickedOnLinkForAndEntered(String strText,String strType,String strSSN) throws Throwable {
        if (strType.equalsIgnoreCase( anotherString: "csp")) {
            enteredSSNAndContinued(strSSN);
        } else if (strType.equalsIgnoreCase( anotherString: "fa")) {
            enteredSSNAndContinued(strSSN);
        }
        return this;
    }

    public [illegible]WorkspacePage enteredSSNAndContinued(String strSSN) throws Throwable {
        waitForElement(SSN_Textbox,  intWaitTime: 6);
        setValue(SSN_Textbox,strSSN,  intWaitTime: 5);
        clickElement(Continue_Button,  intWaitTime: 5);
        return new [illegible]WorkspacePage(inject);
    }
}
```

KARATE Features and development

Karate

Web-Services Testing Made `Simple.`

maven-central v0.9.2 | build passing | release v0.9.2 | support slack | Follow 632

Hello World

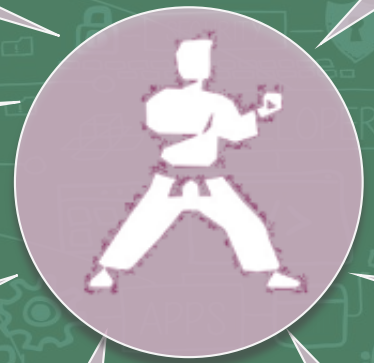
```
Scenario: create and retrieve a cat

Given url 'http://myhost.com/v1/cats'
And request { name: 'Billie' }
When method post
Then status 201
And match response == { id: '#notnull', name: 'Billie' }

Given path response.id
When method get
Then status 200
```

- JSON is 'native' to the syntax
- Intuitive DSL for HTTP
- Payload assertion in one line
- Second HTTP call using response data

Index

Start	Maven | Gradle | Quickstart | Standalone Executable | Naming Conventions | Script Structure
Run	JUnit 4 | JUnit 5 | Command Line | IDE Support | Tags / Grouping | Parallel Execution | Java API
Report	Configuration | Environment Switching | Reports | JUnit HTML Report | Logging
Types	JSON | XML | JavaScript Functions | Reading Files | Type / String Conversion | Floats and Integers | Embedded Expressions | JsonPath | XPath | Karate Expressions
Variables	`def` | `text` | `table` | `yaml` | `string` | `json` | `xml` | `xmlstring` | `bytes` | `copy`
Actions	`assert` | `print` | `replace` | `get` | `set` | `remove` | `configure` | `call` | `callonce` | `eval` | `read()` | `karate` API
HTTP	`url` | `path` | `request` | `method` | `status` | `soap action` | `retry until`
Request	`param` | `header` | `cookie` | `form field` | `multipart file` | `multipart field` | `multipart entity` | `params` | `headers` | `cookies` | `form fields` | `multipart files` | `multipart fields`
Response	`response` | `responseBytes` | `responseStatus` | `responseHeaders` | `responseCookies` | `responseTime` | `requestTimeStamp`
Assert	`match ==` | `match !=` | `match contains` | `match contains only` | `match contains any` | `match !contains` | `match each` | `match header` | Fuzzy Matching | Schema Validation | `contains` short-cuts
Re-Use	Calling Other `*.feature` Files | Data Driven Features | Calling JavaScript Functions | Calling Java Code | Commonly Needed Utilities | Data Driven Scenarios
Advanced	Polling | Conditional Logic | Before / After Hooks | JSON Transforms | HTTP Basic Auth | Header Manipulation | GraphQL | Websockets / Async

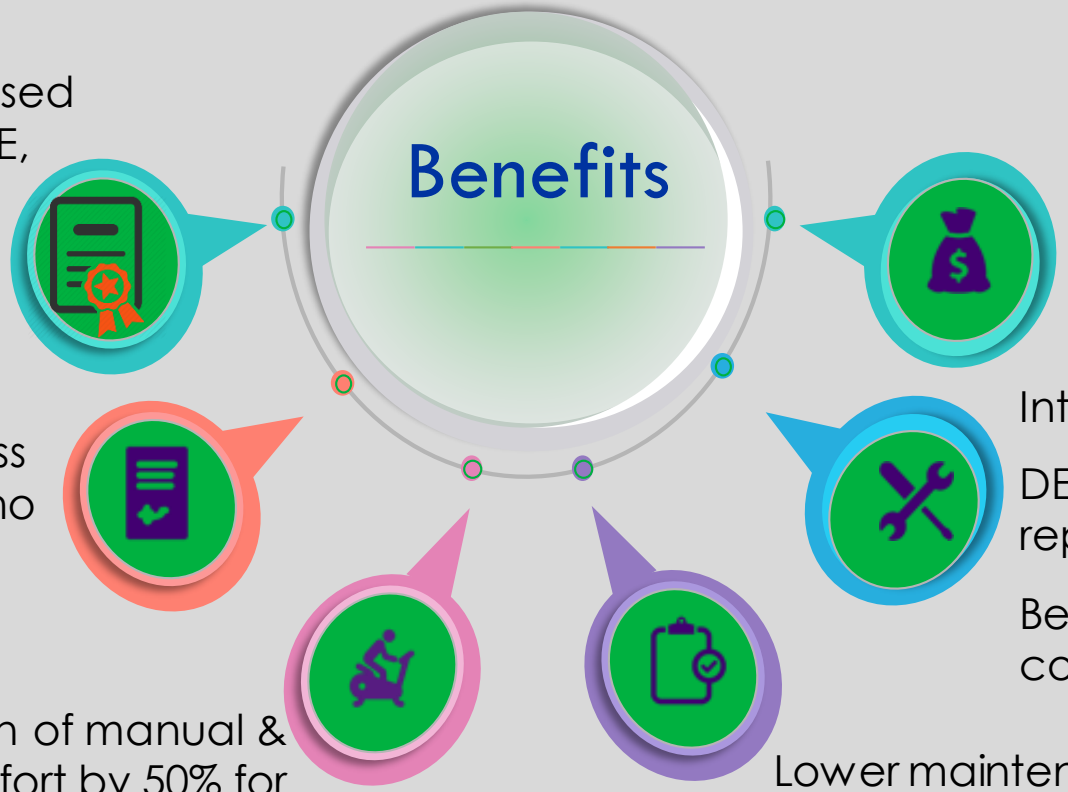
KARATE FEATURES



- No JAVA knowledge required
- Less code, less maintenance
- Pipeline Integration
- Built in keywords
- JUNIT, TESTNG
- Cucumber based Gherkin Support
- Data driven testing
- JSON schema validation support
- Easy debugging

Cognizant

Benefits



Zero Licensing Cost – Based on Selenium and KARATE, which are open source

User friendly: The frameworks are very easy to use. Successfully cross trained 80% manual resources who are delivering quality scripts

Reduction of manual & automation effort by 50% for both in-sprint and regression

Lower maintenance due to implementation of page object model, Utilization of KARATE

Effort Savings due to automation execution **$49896**

Integration with DevOps Pipeline

DEV and QA team using same repository

Better Dev-QA-Business collaboration

For further updates and discussions, please contact

Sandip Agarwala (119625)

Jishan Ali Mondal (261179)

Cognizant

Cognizant

Thank You