Daimler ICertis Assessment - Quality Engineering Part

Executive Summary

The assessment objective was to understand and identify the opportunities in the current Agile Test Practices as part of the NPS solution to achieve the following goals -

- Accelerate deployment and time to market with high quality index
- Improve product quality and reduce issues by adopting shift-left best practices
- Increase visibility, agility and workflow between parties

The critical observation made during the assessment in terms of Quality Engineering process are -

- Few Anti Patterns of Agile Test Practices were observed, the primary one is the current planning and grooming process forces extensive Functional Acceptance and Integration Testing getting executed in Daimler Testing phase; due to short window for testing in Current Sprint Window. Shift Left is one of the foundational pattern of Agile Test Practice which is deviated. There are stories sized more than 13 story points which have limited or no Functional and Exploratory testing in Sprint cycle. This results in finding more bugs in Daimler Testing and UAT Testing phase.
- Lack of automation limits the possibilities to get quick feedback on the changes in code/environment/infrastructure through integrated CI/CD. The current manual Sanity Certification takes 2 to 3 hours with 3 Engineers and the sanity suite consists of 13 End to End test cases. No TDD, ATDD, BDD methodologies is implemented in the development life cycle which leads to quality issues. Unit Testing is addressed as part of minimal manual functional testing done in Developer’s environment by Test Engineer to sign of the user stories during the sprint phase.
- Lean Documentation practice is too lean here. Functional Test cases don’t have proper or no test steps and they are mostly one liner scenarios in the Test Repository. The test execution happens with tribal knowledge within the team. Test Repository is not intact with updates relevant to the changes in the functionality; repository management is aligns to release scope and not functional business requirements. No Test Plans or Test Summary Report to strategy and sign off testing for the EPICs.
- Quality Environment strategy do not incorporate best practices and hence most of the Functional Testing from ICertis is executed in Engineer's Local environment and Dev environment which are not stable ones. These environments are not isolated and hardened for QA validations; the testing on these environments are not advised for certifying features or verifying bugs. Zero automation limits environments validation for sanity as a pre-requisite to any testing activities.
- No Standard Process to defined on Non Functional Testing activities. The Process to identify, define, analyse and monitor performance, scalability and security aspects of the implementation is not clear. Static Code Analysis is not in place to assess performance, scalability and security governance of coding practices. A mature process should allow us to continuously assess and monitor these aspect of the implementation and provides better code quality up in the stream.

Current State

In the current delivery framework model, there are few major gaps on Agile Test Practices due to the current implications with planning, grooming and execution aspect. There is very minimal testing effort spent in a sprint for the developed user stories. Most of the functional and integration testing is pushed to Daimler Testing Phase than the Realisation phase. This challenge is posed to the testers because of the large sized stories which gets code delivered at mere end of the sprint and the deployment frequency to the QA environment is forced twice in a two weeks sprint cycle due to deployment challenges with P2P Migration process. Though there is module based testing done on Developer's Local environment (Laptop/Desktop) which should be address through automated Unit Tests. There is likely very minimal chances that these test are repeated in Integrated QA environment before handing over to Daimler Testing Phase. Hence the deliverable to Daimler Testing Phase has potential critical functional bugs and quality issues which is exhibited.
With current Non-Production environment model of Single Dev, QA, UAT, HotFix Environments; the programmers and testers don't have their own sandboxes/playgrounds to work and test in the features during sprints. We recognise that not all applications lend themselves to this, but you need to know what build is under test. Testing in Development environment due Sprint (Realisation phase) doesn't work well and the environment is never stable enough for effective testing.

Parallel environment preparation is challenging due to time consuming deployment model with P2P process; though the infrastructure preparation an one time activity is automated, the deployment for code changes and business configuration changes is complex and take 2 to 3 hours depends on the amount of configuration changes. The code deployment is straight forward and not time consuming activity.
Sanity Certification of environment is done manually and it takes 2 to 3 hours to complete. There is a plan to start Sanity suite automation in upcoming releases. Missing of Automation Approach and Automated Regression Suites limits the opportunity to establish continuous integration and deployment. Rapidly changing code (UI) for Daimler customisation and legacy product code are the primary barriers observed to automation.
Interaction between the customer and developer exhibited during multiple phase like Release Planning phase (2 Months) and Refinement Phase (2 - 2 Weeks Sprints); but major chuck of quality issues surface in Post Integration phase are due to 'Requirement Gaps'. The understanding of the feature requirement from the Product Owners through the Professional Services team (Functional Consultants) which involves both or one type of changes 'code changes and configuration changes' done with deep collaboration between ICertis and Daimler Technical and Business teams and recorded in Jira and TFS. Changes beyond the information gathered as outcome of these planning and grooming discussion; should be considered as change request and taken as user stories for subsequent sprint/releases.

There is a standard practice of Performance and Security Testing for Core Application and not for the Daimler specific customisation code base. The performance testing are focused upon web rendering aspect and not specific to reliability, resiliency and scalability aspect of application. No static code analysis is available to capture security gaps and code quality aspects. With lack of no Unit and Component Test; continuous and quality delivery to customer is undue challenge without iterative feedback early and often. We need automation to provide a safety net, provide us with essential feedback, and help drive coding.

Current Testing Activities involves

- Functional Acceptance Testing for PBI during Realisation Phase prepared and executed on Dev and QA environment by iCertis Team; on completion the status of the PBI will be marked as "Testing in Progress Testing Completed" state in TFS and "Ready for Daimler Testing" in Jira for respective User Story. Due to limited timeline there are observation of limited documentation on test artefacts in MTM; one liner test cases without any proper test steps which leads to ambiguity and transition of knowledge; lack of test plan and test completion summary report for a release pose challenge to traceability; limited testing in sprint cycle results on finding bugs during later stages which is expensive to debug and fix.
- Daimler Testing involves Functional Acceptance Testing on QA and UAT environment based on the possible integration available to test the use cases for E2E certification and mark status to "Done" in Jira; provided there aren't any Must, and Should Priority Bugs. There is no standard process to the criteria, there is always exceptions to override this criteria based on the agreement different stakeholders; this should be driven by release management team.
- UAT Testing is done by Professional Services and Business Users with integrated components to validate the End to End aspect of the Business flows and behaviour. There is no insight of what is being tested and the artefacts to iCertis Team. This is completely driven by Daimler team; when bugs are filed in JIRA, subsequent bug items are filed in TFS to support. UAT Testing at Sprint 5 and 6; is very late to identify issues and resolve them.
- Production Testing is done by set of suppliers and business users to validate the new features and sign off. There is no involvement from iCertis team, they would be standby for any support.

Findings Summary

Item	Findings	Recommendations
1.	Lack of sufficient testing coverage during the testing phase in the sprint leads to quality issues found in UAT and production. Large stories with more than 13 story points limits testing activities within the sprint cycle. Such stories are either not completely validated or spilled over to next sprint which prevents faster feedback to the change. This leads to quality issues in Daimler testing phase. Around 68% of stories delivered in .9 Release in December are large stories with 13+ story points, 6 of them medium size (8 Story points) and 8 were small (5 or 3 Story pointers). This explains why 16 bugs were only found in QA while 62 bugs found in Daimler testing.	Large and Complex Stories needs to broken down to independent and small stories. The Best Practice is to make sure the User Story is of proper quality and adheres to INVEST criteria (Independent, Negotiable, Valuable, Estimable, Small and Testable). Story Mapping Technique can be used to organise the requirements to better prioritise and plan. This will allow to decompose the complex user stories to small and independent stories with top down approach to achieve the end to end requirement. Sprint duration should be extended to 2+1 Weeks to accommodate Daimler Testing (User Acceptance Testing) in the same sprint and sign off the User Story for DoD. UAT and Production Testing will be as part of Release Window.

2.	The current environment strategy is highly risk as the Functional Testing is executed in Developer's laptop and Dev environment which are not stable and reliable. Same environment (QA) is utilised for acceptance testing along with integration testing. There is no on-demand environment for continuous deployment for certifications.	Build Deployment and Environment strategy to have automation which enables developers and testers to deploy the build on-demand. The Non-Production environment should be having both managed and non-managed environments. This avoids the risk of missing validations in local or dev environments due to chances of instability in these environments. The following deployment steps for both code and configuration changes should be automated and stitched together as part of this deployment process - Backup P2P Code Deploy ES Sync Sanity Test Restore. Increased deployment frequency will help to test as and when the code gets committed.
3.	Lack of automated Unit and Component Testing in the current practice will not allow quality to be baked in as part of every build and this limits the fail-fast approach in the scrum.	To start investing on the Unit and Component testing, the approach is to address the frequently changed modules/components and start adding automated Unit and Component Test with a goal of 25% code coverage as phase 1 target and 60% coverage for phase 2 target and 80% for phase 3 target. The automated Unit test cases will be added to CI Pipeline jobs as when it is developed.
4.	Sanity, Integration and Regression Testing are manual and takes long pool to complete. Rapidly changing code (UI) for Daimler customisation and legacy product code are the primary barriers observed to automation. Lack of automation limits the possibilities to get quick feedback on the changes in code/environment/infrastructure through integrated CI/CD. Hence there is much of manual intervention in the delivery process and quality issues slipping to Post Integration and Production.	The test phase focuses on automation tools for providing comprehensive testing capabilities related to functional, integration, performance, system, acceptance, regression, and non-functional testing activities. One key objective of Test phase is to enable shift-left concept – focusing on behavioural driven development incorporating test driven development practices in the development phases enabling development teams access to implement and measure unit test, code quality, static analysis, static application security testing, automated functional testing, integration testing, and regression testing. These capabilities should be enabled in the development workspace – via dedicated development environment, and development integration environment.
5.	Performance and Security are done at core product level but not for the application changes introduced for Daimler features. No standard practice for Non Functional Testing; performance and security testing missing standard criteria to identify the candidate change to go through the relevant certification.	There should be standard practice defined to identify and execute Non Functional Test as part of the implementation of New Features. The practice of identifying the candidates for non functional testing should be discussed during the refinement phase and subsequent stories are created to address the need.
6.	Lack of Quality Practice and Governance, test artefacts like Lean Test Plan, Test Cases, Lean Test Summary Report for Functional and Non Functional Testing completion are missing or incomplete. Functional Test Cases don't have proper test steps and mostly one liners. Regression Test Cases needs updates to sync up with current functionality.	Quality Practices and Governance should be introduced to ensure some of the manual activities are track and managed. As part of the automation of Regression suite, the test cases to be updated and the team has to work with Product team to update and review the test cases before backlog automation scripting activity starts. Without Test Automation, it is critical to keep the test cases upto date and have sufficient information to avoid any gaps during validations.

Target State

Agile Test Practice -

The product development of NPS solutions involves both code changes and business configuration changes which constitutes dynamic UI and DB Changes to helps to introduce business use case changes. With typical business configuration application and modern agile practices, the test approaches would be to follow the agile quadrants principles to address technical level and business use case validations with automation and manual testing approaches respectively. As part of the typical Agile Practice, the user stories sizing and sprint planning should accommodate develop, test and business certify addressing true DoD of a User Story. The Best Practice is to make sure the User Story is of proper quality and adheres to INVEST criteria (Independent, Negotiable, Valuable, Estimable, Small and Testable). Story Mapping Technique can be used to organise the requirements to better prioritise and plan. This will allow to decompose the complex user stories to small and independent stories with top down approach to achieve the end to end requirement. The development, testing and product acceptance testing of a user story should be done in one iteration.

Test Driven Development should be deployed to enable the culture to automate first and develop code after to emphasis the shared responsibilities of product quality. While improving and extending automation practices both Dev and QE engineers contribute to build the backlogs on Unit Testing, Integration Testing, Regression Testing and User Acceptance Testing. Over few dedicated sprints with dedicated or shared team, the Test Driven Development practices can be followed on regular sprint cycles delivering business features. Team is required to trained on TDD approach, Toolset and Mindset.

Acceptance Test Cases and Acceptance Criteria should be arrived during grooming the user stories. The completion of User Stories should be driven by these acceptance test and criteria. Lightweight Documentation should be maintained for the traceability aspect. Simplified Test Plan for every feature set or epic is required to maintain the checklist to suggest test, focus on the essence of the test, capture test ideas for exploratory and non-functional test. Similarly Test Summary Report for a Release is maintained to track the completion of test planned and record the findings as part of exploratory and non functional testing.

Test Engineering Practice -

Test Automation is a core activity of any agile development methodology which enables DevOps strategy to Continuously Delivery to Production. Test automation becomes ever more important due to the quick feedback response that it provides to the development team about the health of the components, system and applications.
Automation coverage should be increased by the number of unit tests, component tests, integration test and API tests. The low-level tests will provide a safety net to ensure the code is working as intended and helps prevent defects escaping in other layers of testing.
We recommend 90% to 95% of Unit Test Coverage and 85% of functional coverage of component, API and integration test. These regression test are more frequent and aligned with the process of Continuous Integration.

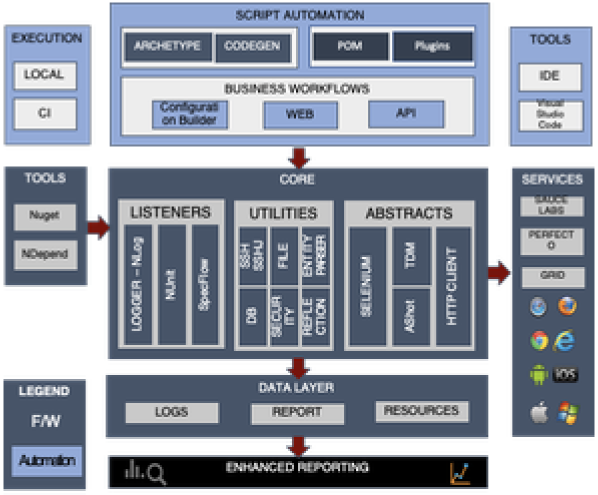
This can be achieved with adaptive implementation of Automation for both Functional Validation for the Sprint and Regression coverage. With Automation in place, the test execution can be seamless between iCertis and Daimler Testing Phase. Automation Test Scripts can be extended for both function and regression suites between these two entities provided they collaborate and plan the strategy of automation effort.

The automation testing approach should address two key aspects, the product platform/components' and business validations; the target approach should deploy both **TDD (Test Driven Development) and BDD (Behavioural Driven Development)** models for automation which drives both unit & component automation validates engineering changes for Platform and End to End business behaviour automation validates the Business configuration changes.
To make it more customer and developer driven approach, **ATDD (Acceptance Test Driven Development)** should be introduced to which focuses on capturing requirements in acceptance tests and uses them to drive the development. The approach will be similar to BDD (GWT Approach - Given When Then) and closer to TDD model. Acceptance tests are a part of an overall testing strategy are the customer tests that demonstrate the business intent of a system which can reused for UAT testing.
Component tests are technical acceptance tests developed by an architect that specify the behaviour of modules or components pertaining to focused engineering platform/application solution through Microservices or Web Services. Unit tests are created by the developer to drive easy-to-maintain code. *Cross-functional testing includes usability testing, exploratory testing, and non-functional testing (Performance, Scalability, Resiliency and Security).*
One of the main issues with our current state was frequent changes in requirements and scope creep. With the ATDD approach, it will be imperative that once the developer started working on a story, the requirements becomes locked. Any changes must be prioritised alongside the other upcoming stories and added to the upcoming sprints. This would reduce the workload of both developers and testers to improve the productivity, and prevented the stakeholders from having unrealistic expectations of completed features. As well this process would help continuous automation practices in both sprint and release cycles.

Role and Responsibilities -
Scrum QE Team will understand the user stories and features through the workshop with PO, BA, UI/UX and Backend Developers. Test Scenarios are identified, reviewed, and automated for product output as part of the Sprint deliverables.
Scrum QE Team will verify and validate at component and functional level to demonstrate the functionality to the PO to meet the acceptance criteria to accomplish the DoD. They will execute and validate the predefined acceptance test before marking the story as done.
Unit Test, Component Test will be automated and maintained by Scrum team (Dev and QE).
Integration QE Team Aka Daimler Testing Team will understand the user stories and features through the workshop with PO, BA, UI/UX and Backend Developers. Integrated or E2E Acceptance Test Scenarios are identified, reviewed and automated for product output as part of the Sprint deliverables.
Integration QE Team will verify and validate at Integration and End to End aspects. Integration team will also help in running and maintaining the regression testing to ensure the current sprint deliverables are not breaking the existing functionalities of the application. Integration, E2E tests will be automated and maintained by the Integration QE Team. This will be in BDD approach.
Integration QE team will be supporting the CI Pipeline configuration with DevOps engineers to engage the automation suite. This is will be joint team effort of both iCertis and Daimler Testing Team.
Performance QE Team will understand the performance goals of the applications both web service calls and UI rendering on web platform from the BA and Architect. They will device the workload and build scripts to automate the performance test execution with technical profiling sessions with Programmers and Scrum QE Team. They will run the scripts and analyse the results before sharing to the stakeholders and be responsible to answer questions pertaining to the results. On agreement for additional value-added service, the Performance engineer will be allocated to support configure and maintain the early Performance Testing toolset to provide continuous monitoring on the Performance aspect of the product/application using ELK or alternate Stack.
PO (Product Owner) should run User Acceptance Tests or Business Acceptance Tests to confirm the built product is what was expected and that it meets user's expectations. **Integration QE Team** will help in facilitating the User Acceptance Testing and the PO/BA with collaboration with Professional Services team will be perform the UAT Testing during the UAT window (1 Sprint).

Automation Framework -

To follow the ATDD approach, the automation framework should be designed to support keyword or Gherkin based solutions (Cucumber/SpecFlow) with critical features of TDD solutions to enable component test automation using NUnit.



The automation framework should have following critical features to address the need for this project -

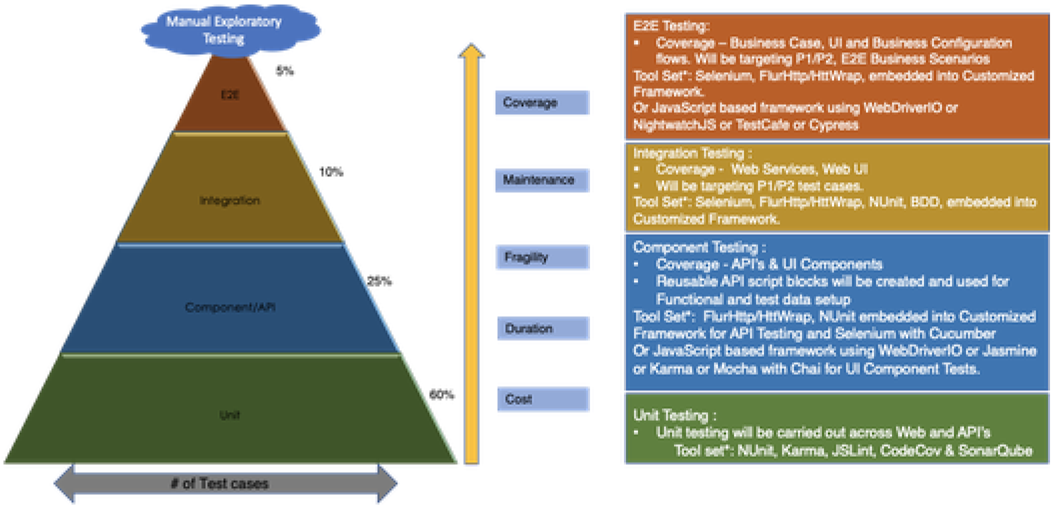
- Archetype - For Easy Adoption and Onboarding. Self Learning and Training can be addressed using the samples and docs as part of the Archetype feature.
- Runner Client - To Initiate the Test Run through command line and abstract the methods with embedded listener to drive the test run and log/record each actions to control and manage the test run. Support both NUnit and SpecFlow approach to give hybrid approach of TDD and BDD.
- Page Factory or POM - Dynamic Page Factory or Object Model to enable Web UI Element and Actions instanced through JSON through ONDemand approach. The testing involves certifying both code changes and business configuration changes involving dynamic UI components; strategising the UI automation should effectively use JSON based approach to add and manage the UI Objects.
- Page Object Recorded - The Utility would help in record and update the changes on the UI quickly on click of a button. This can be easily built as Chrome plugin. To address dynamic UI as part of the Business Configuration changes, a browser based XPI/Plugin is required for quick and easy
- Logger Utility - Logging and Snapshot capabilities to record and present the Test Results with Evidences
- GWT Wrapper - SpecFlow/Gherkin Library with a Wrapper to handle the way of writing the specification steps using Domain Specific Language with drivers to manage the hooks (Scenario's Interceptor) and transformations (Parameter Interceptor)
- Extended Reporting or Dashboard Feature - Critical for Rapid Development Environment (Always seen as good to have feature)
- Test Repo Connector - Use TFS API to connect with MTM and Update the Run status for Automated Test Cases

Technical Stack - Framework (Free Open Source Software Stack)

- IDE - Visual Studio Code
- Programming Language - C# [Preferable for MS shop; Otherwise JavaScript/Ruby/Python are most suitable to have neutral language; I would prefer Ruby which seamlessly simplifies to embed both UI and Component Driven Testing for Web Services)]
- Web Drivers - Selenium Webdriver 3.14.x
- HTTP Libraries - Fluently HTTPClient (Lightweights like FlurlHttp or HttWrap can be used)
- Test API - NUnit for TDD and SpecFlow for BDD approach (Hybrid approach to build ATDD based solution)
- Helper Components -
  - ExcelHelper - Excel Data Reader - 3.0.0 (To Support Data Driven or Hybrid automation approach)
  - LogHelper - C# Logger or NLog (Customised Logging Library on standard open libraries)
  - HTMLHelper - Generic helper library which will be used to perform operation on the HTML table on a Web Page
  - DatabaseExtensionHelper - Helper Method or Library to establish connection with database which allows retrieval/storage of informations
  - ConfigurationHandler - Usage of different Namespaces or classical way of App.config methods
- Database - CosmoDB or DocumentDB to Store the Execution results for Automation Reporting and Quality Dashboard
- Dashboard - GraphQL or Kibana to define and develop the Quality Dashboards to monitor and govern the Test Automation execution during Sprint, Integration and UAT phases.

Automation Approach

Automation approach is to address the automation backlog and new feature automation. For new feature automation, Unit and Component Automation will be addressed by Developers and Integration and Acceptance E2E Test will be automated by Testing team.



E2E Testing:
- Coverage – Business Case, UI and Business Configuration flows. Will be targeting P1/P2, E2E Business Scenarios

Tool Set*: Selenium, FlurHttp/HttWrap, embedded into Customized Framework.
Or JavaScript based framework using WebDriverIO or NightwatchJS or TestCafe or Cypress

Integration Testing :
- Coverage - Web Services, Web UI
- Will be targeting P1/P2 test cases.

Tool Set*: Selenium, FlurHttp/HttWrap, NUnit, BDD, embedded into Customized Framework.

Component Testing :
- Coverage - API's & UI Components
- Reusable API script blocks will be created and used for Functional and test data setup

Tool Set*: FlurHttp/HttWrap, NUnit embedded into Customized Framework for API Testing and Selenium with Cucumber
Or JavaScript based framework using WebDriverIO or Jasmine or Karma or Mocha with Chai for UI Component Tests.

Unit Testing :
- Unit testing will be carried out across Web and API's
  Tool set*: NUnit, Karma, JSLint, CodeCov & SonarQube

Sanity Suite: To Sanity check that the application's existing Functional intact. Smoke Test pack should catch the most obvious issues and it should last no longer than 5 to 15 minutes to give quick feedback.

· Environment & Infrastructure Verification Pack - 10 to 15 Test to validate environment and infrastructure components and services for health check of the environment

· Sanity Verification Pack - Heart Beat Pack - E2E Tests which can be run on Production and Non Production environment to Validate the Critical Business Flows exercised by the Primary Customers or Product Users

Component Functional Regression Packs – should be no more than half hour with multiple isolated chunks

This pack contains the full regression suite of tests and the goal is to get quick feedback with a larger set of tests. If the feedback takes more than 1 hour, it is considered slow. Either the number of tests have to be reduced by using pairwise test technique, (or) by creating test packs based on relevant Regression Suite approach.

End to End Regression Pack – should be no more than 2 hours

This pack contains tests to cover the application as a whole. The aim of these tests is to ensure that various parts of the application which connect to various databases and integrated components like Approval Box, etc work properly.

The End to End tests are not meant to test all the functionalities as those are already tested in the functional regression packs. However, these tests are "light-weight" which just check the transitions from one state to another and a handful of the most important scenarios or user journeys in production site.

These tests are mainly executed through the GUI, as they check how users would use the system.

All automation must be architected to reduce maintenance through code re-use. Any given function that repeats throughout the applications should be modularised for easy maintenance.

Component Test, Integration Test and E2E Test will verify the functionalities pertaining to the Platform, Component and Web/Mobile UI implementations.

Back Log Automation Strategy

Since there is no automation available and there is a plan for Sanity Automation; good amount of engineering efforts is required from both Dev and Test Engineering teams. This can be accounted as part of the Tech Debt effort and considered as top priority to achieve the DevOps Strategic Goals.

With given 2000+ manual regression test as backlog; the automation strategy should be well defined and tracked to closure of all back log items before next development iteration begins. The approach has to be planned as part of regular sprints with both Dev and QE Engineers, Leads and Architects to accomplish in building the critical Sanity and P1 automation suites to enable & setup the continuous deployment pipeline.

- Back Log Automation Strategy for this engagement will have the following critical milestones or epics to track and closure -
- Review and Cleanup current Manual Regression Suite for Context and Content Updates - Business Analyst, Daimler Test Leads, iCertis Leads
- Requirement Traceability Matrix Fixes/Corrections in Test Management Repo - Product Owner, Business Analyst, Daimler Test Leads, iCertis Leads
- Priority Categorisation - Test priority for Business, Functional and Platform Criteria - Product Owner, Business Analyst, Daimler Test Leads, iCertis Leads
- Identification of Automation Candidates - Product Owner, Business Analyst, Daimler Test Leads, iCertis Leads
- Review the Framework capabilities and workshop with stakeholders - Automation Architects & Leads, Development Leads/Architects
- Plan and Design the Automation Implementation - Automation Architects & Leads, Development Leads/Architects
- Modify the Framework capabilities to address Automation implementation plan & design - Automation Architects & Leads, Development Leads/Architects
- Framework Capabilities and Sample Automation Training and Workshop - Dev Engineers, Automation Engineers and Automation Architect & Lead
- Build, Validate and Deliver Sanity Suites [EVP, IVP, BVP, and SVP] - Dev Engineers, Automation Engineers and Automation Architect & Lead
- Co-ordinate and Integrated with Build/CI Pipeline
- Build, Validate and Deliver P1 Regression Suites - Dev Engineers, Automation Engineers and Automation Architect & Lead
- Co-ordinate and Integrated with Build/CI Pipeline
- Review and Update the POM/Page Factory JSON; Isolate the components to separate code or database repo - Dev Lead and Automation Architects/Champions
- Build, Validate and Deliver P2 Regression Suites - Automation Engineers and Automation Architect & Lead
- Co-ordinate and Integrated with Build/CI Pipeline
- Build, Validate and Deliver P3 Regression Suites - Automation Engineers, Test Engineers and Automation Lead

Dedicated Lean Automation Tools Team is going to be critical for first two quarters or till the framework is stabilised and major scope of automation backlog is completed. Later the team can join the sprint team to help on continuous automation effort or join DevOps team to enable the other automation tasks.

Automation Runs and Deployment Pipelines

In the target state, testing should happen at different times in the development lifecycle - throughout instead of at the end (which is likely happening in current state - most of the testing is happening in Daimler Testing and UAT Testing phases which mere end of the Delivery cycle). We should heavily rely on automation (primarily unit tests and component tests) and our pipeline to give us a sense of the quality of the system.

Testers should work closely with the developers to ensure that quality is maintained and that defects almost never occur. Testers typically focus on the "User Tests" from the pyramid above. They can also pair with developers to help them acquire and improve their testing skills. Testers also have a role to play in the definition of the build pipeline and support for continuous improvement of the pipeline till attains a mature state.

In many ways, the pipeline is the closest thing we have to a "testing phase". This is where all of our tests should run each time we check in code and ultimately deploy our code to the production environment. Although all pipelines are different, there are some core patterns that is common from team to team. Here is an example of the stages that the team should do:

1. Build and run the unit tests
2. Run static code analysis and fail if quality rules are not met
3. Deploy with all work in progress feature toggles on and run only the tests that focus on un-released behaviour. Any external or internal backend system outside of our control should be mocked.
4. Deploy with all work in progress feature toggles off and run all remaining tests to provide regression. The tests need to run fast so they should run in parallel. Also, any external or internal backend system outside of our control should be mocked.
5. Deploy with toggles off and run a small subset of tests across multiple browsers and devices. Backends should be mocked.
6. Deploy with toggles off and run a very relevant subset of integration tests without mocks. This ensures full integration works properly.
7. Run static and dynamic security tests.
8. Run load and performance tests.
9. Run compliance tests and address any additional pre-production compliance tasks.
10. Run UAT Testing with Supplier or Buyers
11. Deploy to production.

Transformational Summary

Queries to ask?

S.No	Queries/Follow-up with iCertis Team	Response
1	The Current Regression Suite Size and review the test repositories for functional mapping and traceability. Understand the prep and execution activities for release activity; will take previous release and deep dive the flow by sprint over sprint activities	

2.	Review the Test Plan for feature set which includes Non Functional Test Requirement and Test Result/Completion Summary Report	
3.	Review and understand the artefacts for Performance Testing activities	
4.	Review the Bug Report and Sample Bugs with Priorities Must, Could, Should in Non Prod and Prod	
5.	Run down a Recent Penetration Testing Analysis and Report	