

**DFT 1**

**UFT Test Automation Strategy**

Version 1.2

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version Number | Release  Date | Prepared By | Reviewed By | Approved By | Summary of Changes |
| 1.0 | Apr-16-2020 | Ravi Sindri |  |  | Initial Version |
| 1.1 | Apr-23-2020 | Venu Vallapaneni |  |  | Draft Version |
| 1.2 | Apr-24-2020 |  | Ravi Sindri |  | WIP Version |
|  |  |  |  |  |  |

Contents

[1.0 Introduction 1](#_Toc38908689)

[1.1 Purpose 1](#_Toc38908690)

[1.2 Target Audience 1](#_Toc38908691)

[1.3 Related Documents 1](#_Toc38908692)

[1.4 Definitions and Acronyms 1](#_Toc38908693)

[2.0 Test Automation Scope and Objectives 2](#_Toc38908694)

[2.1 High-level Script Development Plan 2](#_Toc38908695)

[2.2 Applications Under Test 3](#_Toc38908696)

[2.3 Testing Types 4](#_Toc38908697)

[3.0 Test Automation Process 5](#_Toc38908698)

[3.1 Testing Tools 5](#_Toc38908699)

[3.2 Test Environment 5](#_Toc38908700)

[3.3 Test Cases Selection Process 5](#_Toc38908701)

[3.4 Test Data Preparation 6](#_Toc38908702)

[3.5 Test Script Development & Review Process 7](#_Toc38908703)

[3.6 Test Script Development Standards 8](#_Toc38908704)

[3.7 Test Script Execution & Reports 8](#_Toc38908705)

[4.0 Test Automation Framework 8](#_Toc38908706)

[4.1 Framework Implementation & Maintenance 13](#_Toc38908707)

[4.2 Test Automation Farm Setup 13](#_Toc38908708)

[5.0 QA Metrics & Measurement Framework 14](#_Toc38908709)

[5.1 Verita QA Dashboard 14](#_Toc38908710)

[5.2 QA Metrics 15](#_Toc38908711)

[6.0 Reporting and Communication 16](#_Toc38908712)

[6.1 Status Reporting 16](#_Toc38908713)

[6.2 Meetings 16](#_Toc38908714)

[6.3 Escalation Procedure and Resolution Mechanism 17](#_Toc38908715)

# Introduction

## Purpose

The purpose of this test automation strategy is to provide the readers a comprehensive view of the test automation approach taken for building an automated regression suite covering the functional areas across various workstreams for Autodesk DFT Phase 1 as part of Apollo program.

## Target Audience

| Audience | Role |
| --- | --- |
| Program Managers | Contributors |
| Teat Automation Architects | Contributors |
| Test Automation Leads | Contributors |
| Test Automation Engineers | Reference |
| Business Users | Reference |

## Related Documents

This document makes reference to the following documents in a direct or indirect method:

| Reference Links |
| --- |
| Apollo Program  <https://wiki.autodesk.com/display/DBP/Apollo> |
| DFT - Phase 1 UAT Validation Milestone  <https://adskdft.testrail.io/index.php?/milestones/view/13> |
| QA Automation  <https://wiki.autodesk.com/x/52_GJQ> |

## Definitions and Acronyms

| Acronyms | Definition |
| --- | --- |
| CTAF | Cigniti Test Automation Framework |
| UFT | Unified Functional Testing |
| AUT | Application Under Test |
| SUT | System Under Test |
| UAT | User Acceptance Testing |
| SIT | System Integration Testing |
| OR | Object Repository |
| TRTM | Test Requirement Traceability Matrix |
| DSR | Daily Status Report |
| WSR | Weekly Status Report |
| MSR | Monthly Status Report |

# Test Automation Scope and Objectives

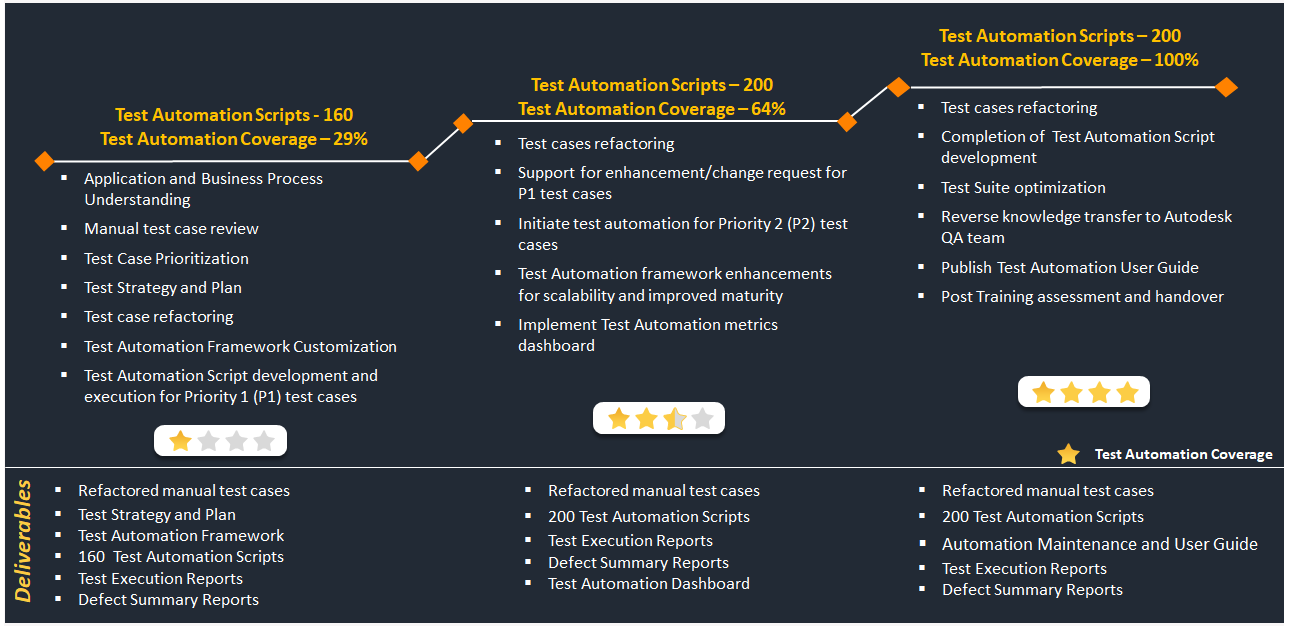
The strategy is to build test automation suite covering the functional testing requirements of various workstreams. Microfocus UFT and CTAF will be used as the test automation tool to build the test automation suite. The same test automation solution can be extended to various other types of testing such as Smoke/Build Verification, System, Integration and End-to-End testing.

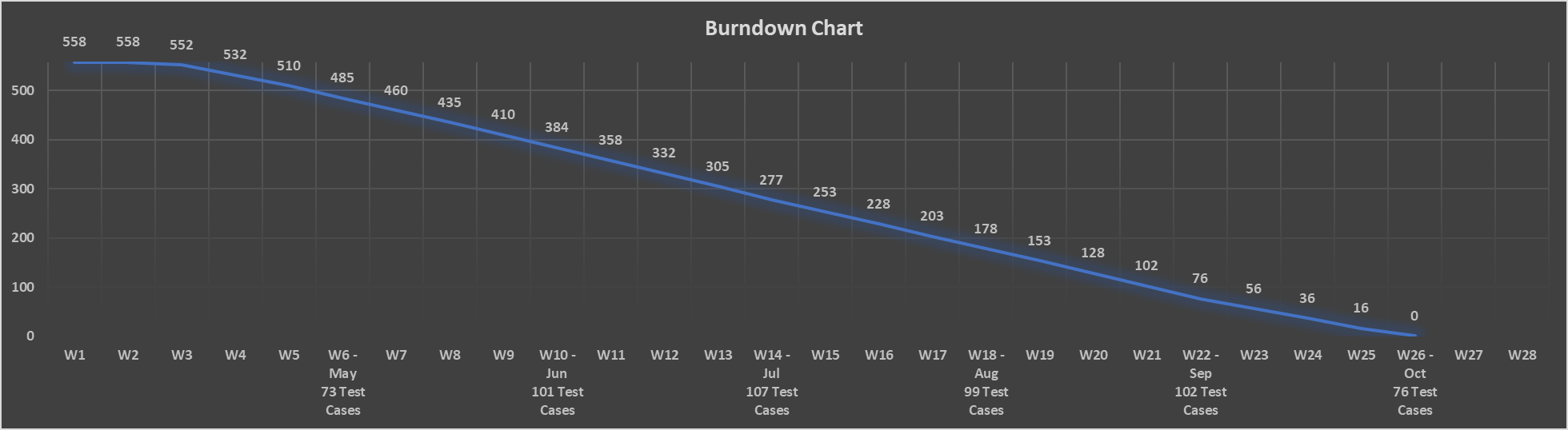
Since the objective and scope of this automation is to support the fictional testing needs of the applications, it doesn’t cover the non-functional testing requirements of the applications.

## High-level Script Development Plan

This plan is derived based on the DFT 1 UAT test cases covering the 6 workstreams and considering Oct 2020 timeline.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Workstream ->** | **BlackLine** | **FA** | **GR** | **PA** | **PTP** | **RTR** | **Total** |
| Total No. of Test Cases | 284 | 552 | 116 | 397 | 678 | 245 | **2272** |
| Total No. of Unique  Test Cases | 23 | 51 | 116 | 25 | 292 | 51 | **558** |





## Applications Under Test

| Applications | Description | Application Type |
| --- | --- | --- |
| SAP Fiori, SAP S4/HANA |  | SAP Web |
| SAP GUI |  | SAP Desktop |
| SAP Ariba |  | SAP Web |
| SAP BRIM |  |  |
| SAP Hybris SOM |  |  |
| Workday |  | Web |
| Concur |  | Web |
| Salesforce CPQ |  | Web |
| OpenText |  | Web |

## Testing Types

|  |  |  |  |
| --- | --- | --- | --- |
| S. No. | Testing Type | In Scope  (Yes/No) | Objective |
| 1 | Smoke Testing | Yes | To decide if a build is stable enough to proceed with further testing, perform non-exhaustive set of tests that aim at ensuring that the most important functions work fine |
| 2 | System Integration Testing | No | N/A |
| 3 | System Testing | Yes | To evaluate the system’s compliance with the specified requirements it covers all the combined parts of a system. |
| 4 | Regression Testing | Yes | To ensure that previously developed and tested software still performs after a change |
| 5 | Acceptance Testing | Yes | To evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery |
| 6 | Mobile Testing | No | N/A |
| 7 | Performance Testing | No | N/A |
| 8 | Load/Stress Testing | No | N/A |
| 9 | Compatibility Testing | No | N/A |
| 10 | Security Testing | No | N/A |

# Test Automation Process

## Testing Tools

|  |  |
| --- | --- |
| **Testing Tools** | **Purpose** |
| UFT v15.0 | Test Automation Script Development & Execution |
| TestRail | Manual Test Case Authoring |
| JIRA | Project Management & Defects |
| Confluence | Wiki |
| GitHub | Test Automation Repository |

## Test Environment

|  |  |
| --- | --- |
| **Environment** | **Purpose** |
| QA Environment | Script Development and Dry Run |
|  | Script Execution (Dev. Final Run) |
|  | Script Execution (Release Run) |
|  |  |
|  |  |

## Test Cases Selection Process

The total no. of test cases available OR planned for automation as part of DFT Phase 1 are 2272 which are distributed across 6 workstreams. These test cases are analyzed considering the two factors listed below to derive 558 unique automated scripts that will directly or indirectly cover the 2272 test cases in TestRail. A requirement traceability matrix will be established the mapping between the manual test cases and automated script to track and measure the coverage.

* We have considered the “Title” column in the test cases sheet as the unique test cases
* “Run” is a test data variation and will have the exactly same test steps with slight variation of validation points

|  |  |  |  |
| --- | --- | --- | --- |
| **Workstream** | **Total No. of Test Cases** | **Total No. of Unique Test Cases** | **"Run"** |
| **BlackLine** | 284 | 23 | UAT - BlackLine - AMER TAX (Jacqueline), UAT - BlackLine - AMER TAX (Julia),UAT - BlackLine - AMER TAX (Tiffany), UAT - BlackLine - AP (Elly), UAT - BlackLine - AP (Esther),UAT - BlackLine - AP (Pei Fen),UAT - BlackLine - APAC ACCTG (Shuqi) |
| **FA** | 552 | 51 | UAT - FA & PA - 1000, UAT - FA & PA - 1100, UAT - FA & PA - General - 1000, UAT - FA & PA - General - 2000, UAT - FA & PA - General - Global |
| **GR** | 116 | 116 | UAT - Global Reporting - APAC, UAT - Global Reporting - EMEA - EUR, UAT - Global Reporting - EMEA - GBP,UAT - Global Reporting - Global Consolidations ,UAT - Global Reporting - Global Consolidations - Custom Reports |
| **PA** | 397 | 25 | UAT - FA & PA - 1000, UAT - FA & PA - General - Global, UAT Äì FA & PA Äì External Customer Project Äì 2000, UAT Äì FA & PA Äì External Customer Project Äì 2100 |
| **PTP** | 678 | 292 | UAT - PTP - 1000, UAT - PTP - 1320, UAT - PTP - Ariba - 1200, UAT - PTP - Ariba - 2006, UAT - PTP - Banking Payments for AMER LATAM, UAT - PTP - EBS, UAT - PTP - Mass Payment Run - 1000, UAT - PTP - Mass Payment Run - 1100 |
| **RTR** | 245 | 51 | UAT - RTR - 1000, UAT - RTR - 2040, UAT - RTR - COA, UAT - RTR - ECC-S4 Interface,UAT - RTR - FX Rate from S4 to Workday, UAT - RTR - Sales Commission |
| **Total** | 2272 | 558 |  |

## Test Data Preparation

Test data are inputs used while executing test cases to verify given function produces expected results. The respective workstreams are responsible for providing appropriate test data that is required for the test cases to be executed successfully.

Each test case will have an associated test input data file (excel sheet). The template of the test input data file might differ based on the test cases (and based on the preference set by respective workstreams). However, it is compulsory to follow certain standards while preparing the test input data files.

|  |  |
| --- | --- |
| **Standards** | **Example** |
| Test Input Data File Naming Standards | The name should be same as the corresponding automated script name. The name should follow proper case with no spaces in between. For Ex. : <AutomatedScriptName>.xls |
| Column Names | The column name should be a logical name (preferably, same as the label of the corresponding field on the application) with prefix “R” for reading values from the column to input into the application and “W” for writing the output generated by the application. Follow proper case with no spaces in between.  For reading column Ex.: UserName  For writing into columns Ex.: Output\_OrderNo |

## Test Script Development & Review Process

Before starting the script development, it is required to perform one cycle of manual execution of the test cases to ensure the pre-requisites are met. This includes test environment availability, appropriate user credentials, test input data availability etc. The other aspect is to understand the navigation flow, application behavior and expected results etc. This would help the test automation team to orchestrate the framework accordingly and implement the required handlers to avoid script errors.

TestRail will be used as the single source for the manual test cases. Mapping will be created between the manual test cases and the automated scripts to establish traceability and thereby to track test automation coverage.

Test scripts will be checked-in to Git repository on a daily basis with execution reports and to obtain sign-off from the engineering team.

The following process will be followed by the test automation team during the test automation script development:

## Test Script Development Standards

TBD

## Test Script Execution & Test Results

It is recommended to ensure that the prerequisites are met before planning the script execution. The scripts can be executed in various ways:

* Standalone – Run from UFT
* Standalone – Executing the “ExecutionBatchFile” utility
* Scheduled – From Jenkins Job(s)
* Scheduled – From Windows Schedular using the “ExecutionBatchFile” utility
* Planned – Planned in TestRail and executing the “ExecutionBatchFile” utility

The framework provides the ability to list all the test scripts in “TestScriptsList” excel with “Run” column value set to “Yes”/”No”. When the execution is performed using the “ExecutionBatchFile” utility, it basically reads all the rows that are marked as “Yes” from TestScriptsList excel, loads the respective test scripts and performs the script execution. Hence it is recommended to use the “ExecutionBatchFile” utility while planning for a batch execution using the TestScriptsList excel. Broadly, there will be two states of script execution:

* Successful Execution: In case of successful execution, the framework continues with the next test script in a sequence as planned and updates the status Pass/Fail against each test case
* Unsuccessful Execution: If the script is not able to proceed further due to unavailability of prerequisites OR in case of unexpected exceptions, it will be redirected to the recovery scenario block to gracefully handle the failure. The recovery scenario aborts the current script execution, writes exception details into a log file and continues with the next script execution

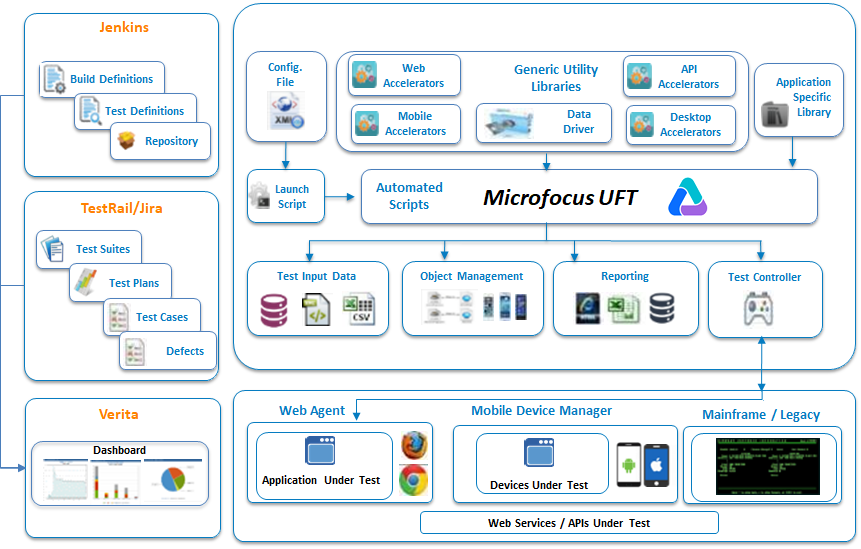
**Test Results:** Once the script execution is completed, the framework generates an execution report file in the “TestResults“ folder of the respective project. It is also possible to push the execution report files to a specified location for archiving. This can be configured using the Config settings.

# Test Automation Framework

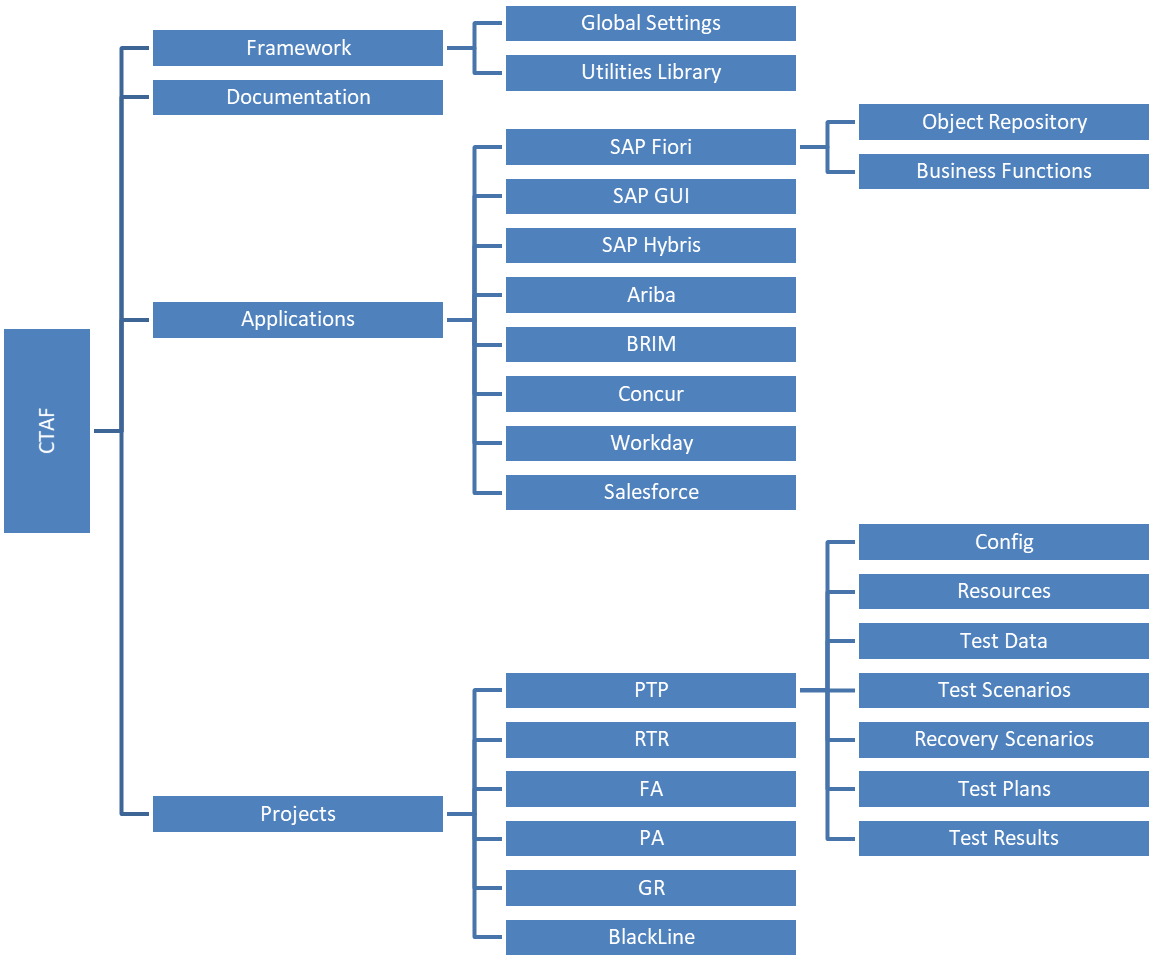
CTAF (Cigniti Test Automation Framework) is an open architecture framework built using Microfocus UFT and VB Script. The framework is developed keeping the functional test automation needs of Desktop, Web, API and Mobile applications. Since framework acts as the base for any test automation, it is important to follow the best design concepts considering various factors. The following are the key factors that are considered to define CTAF and to be followed while building test automation scripts.

| Key Factors | Description |
| --- | --- |
| Maintainability | Orchestrate the framework components such a way that, it enables ease-of-maintenance of various test automation artefacts. Implementation of single-point maintenance, reusability, modular design concepts etc. |
| Modularity/Functional Decomposition | Avoid linear scripting. Scripts are divided into smaller units of code to the best possible extent. So that the scripts are easy to read and debug. |
| Reusability | A thorough analysis needs to be done as to how to build the scripting components so that they are generic by nature and thereby allowing high degree of reusability. |
| Datadriven | Providing the capabilities to drive the test scripts based on the variants of test input data by avoiding hardcoded values in the script |
| Extendability | Providing the capabilities to extend the framework to support test automation for various application covering Web, Desktop, API and Mobile |

The framework is pro-agile and comes with the capabilities to integrated with CI/CD server such as Jenkins, Test Management tool such as TestRail and Jira, and with Cigniti’ s IP Verita (Quality Engineering Platform and Dashboard). The following is the conceptual view of the framework with the default/recommended folder structure.



**Project Structure**:



|  |  |
| --- | --- |
| Component | Description |
| Global Settings/Config | This folder contains the global environment settings which are accessible to the automation suite to run effectively.  Generic Config.xml: This file contains variables related to browsers and configurations.  SAPFiori/Ariba/ConcurConfig.xml: This file contains variables such as SAP Applications URL’s, App Usernames and Passwords etc.  Global.vbs: This file contains all folder path related to the suite (I.e. Test Data file Path, Results folder path) |
| Utilities Library/Business Functions | Contains the Common Functions and Application(s) related common functions.  Generic/Common Functions: Generic Functions are those functions which can be used across all the applications in Autodesk which are independent of applications by passing desired parameters for Ex:   * Kill Process: Kills various applications that are running like Excel, MS Access, etc. * Click on Button/Link/Image: Clicks the Button/Link/Image which is displayed in the application by passing parameters * Set Value in text field: Enter data in the application in a particular field e.g. textbox, etc. * Excel: To read/write/compare data from/to Excel file(s) and input the data to the application * Reporting: This Library file contains all HTML Report related functions are like ‘fCreateResultsSummaryFile’ and ‘fDetailedFailure, etc.   Business Functions: Business functions library is used to perform all the operations like Control validations, screen navigations on each application screens according to the steps mentioned in the test data file. Individual files are maintained for all the applications such as:   * SAPFIoriBusinessFunctions.vbs * SAPAribaBusinessFunctions.vbs * SAPConcurBusinessFunctions.vbs * SAPGUIBusinessFunctions.vbs |
| Object Repository (OR) | The physical description of application specific objects is stored in a shared repository file. Regular expressions will be effectively used to edit object properties in order to make the same object compatible across different releases.  Regular expressions are used to increase the flexibility and adaptability of the tests which enables UFT to identify objects and text strings with varying values. It is a string that specifies a complex search phrase. By using special characters, we define the conditions of the search.  All objects related to various applications are captured and stored in the respective shared object repository. Following are few examples of the object repository:   * SAPFiori.tsr: OR for SAP Fiori Application * SAPAriba.tsr: OR for SAP Ariba Application * SAPConcur.tsr: OR for SAP Concur Application * SAPGUI.tsr: OR for SAP GUI Application |
| Test Scenario | This folder contains the test scripts that are automated in line with the manual test cases/flows etc. |
| Recovery Scenarios | Recovery Scenarios instruct UFT to recover from unexpected events and errors that occur in your Testing environment during a run session. Recovery scenario becomes crucial for large tests, which run unattended and are paused until recovery operation is performed increasing the test execution time.  UFT enables us to create recovery scenarios and associate them with specific tests and the same will activate specific recovery operations when trigger events occur.  The Recovery Scenario Manager provides a wizard that guides us through the process of defining a recovery scenario, which includes a definition of an unexpected event and the operation(s) necessary to recover the run session.  A recovery scenario consists of the following:  Trigger Event: The event that interrupts your run session, for example, a window that may pop up on screen, or a Quick Test run error.  Recovery Operation(s): The operation(s) to perform to enable Quick Test to continue running the test after the trigger event interrupts the run session. For example, clicking an OK button in a pop-up window, or restarting Microsoft Windows.  Post-Recovery Test Run Option: The instructions on how Quick Test should proceed after the recovery operations have been performed, and from which point in the test Quick Test should continue, if at all. For example, we may want to restart a test from the beginning, or skip a step entirely and continue with the next step in the test |
| Test Data | This folder contains the relevant test data files created for each test case. Please refer to 3.4 – Test data Preparation section for more details. |
| Test Results | Contains the test execution result files generated by the tools at the end of each execution. Subfolders with Date/Time Stamp will be created for each execution. The name of the folder/file is customizable and can be set through the config settings.  These reports will have two level of details, 1. Test Execution Summary and 2. Test Execution Details Report. Summary Report will have test environment details printed on the footer and the list of test cases executed with hyperlink and a pie chart of pass/fail counts.  The hyperlink takes to the next level i.e. detailed report which basically captures step wise execution details with Test Step, Details, Status printed on the report with a link to a screenshot of the application screen. |
| Resources | This folder contains templates, images such as JPG/PNG files etc. that are used mainly in the HTML reports. |
| Documentation | This folder contains various documents such as Framework User Guide, Coding Standards Guidelines and Common Functions Help Reference etc. |

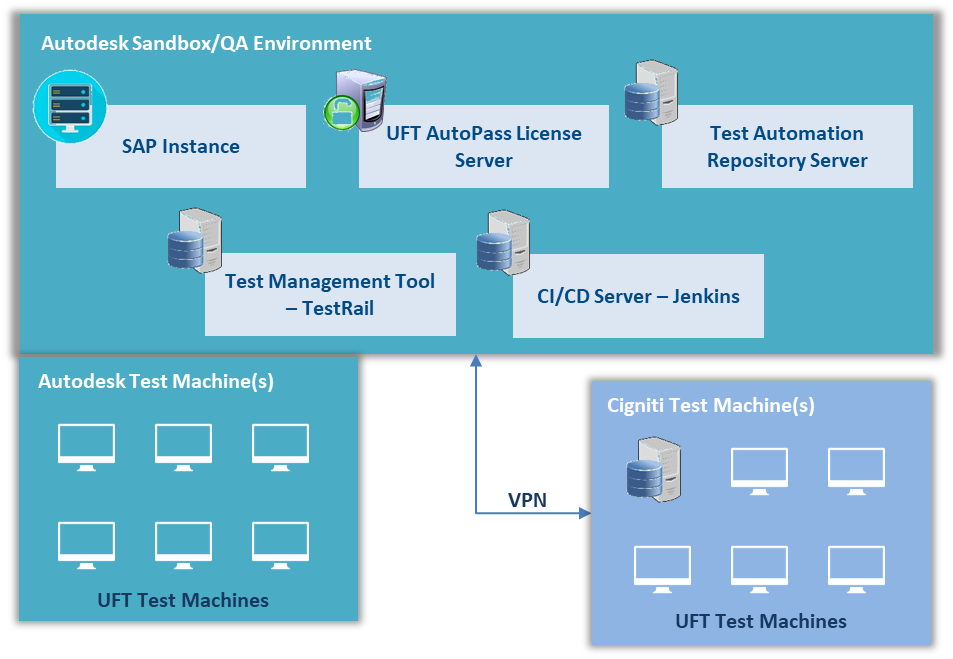
## Framework Implementation & Maintenance

A centralized repository with the replica of the above folder structure will be created in Git with appropriate access controls. Framework Folder will have access restricted to Architects team, any change request to framework components will be routed through this team. Script development team will have full access to their respective projects. Execution team will have Read Only access to the complete project.

| Team | Members |
| --- | --- |
| Architects Team | Ramesh, Santhosh and Balaji |
| Script Development/Maintenance Team | Cigniti Team & Autodesk Team |
| Script Execution Team | Automated Agent, Project Teams and Business Teams |

During the framework implementation phase, Architects Team will work with project/business teams to understand the testing requirements for their respective applications, analyze the test validations, identify the pre-requisites/dependencies and customize the framework accordingly. Post which, the team will identify few critical scenarios for each application to do feasibility and demonstrate the proof of concept to the respective teams. Gather suggestions/ review comments, analyze the possibilities to implement the change, enhance and rollout the framework across project teams.

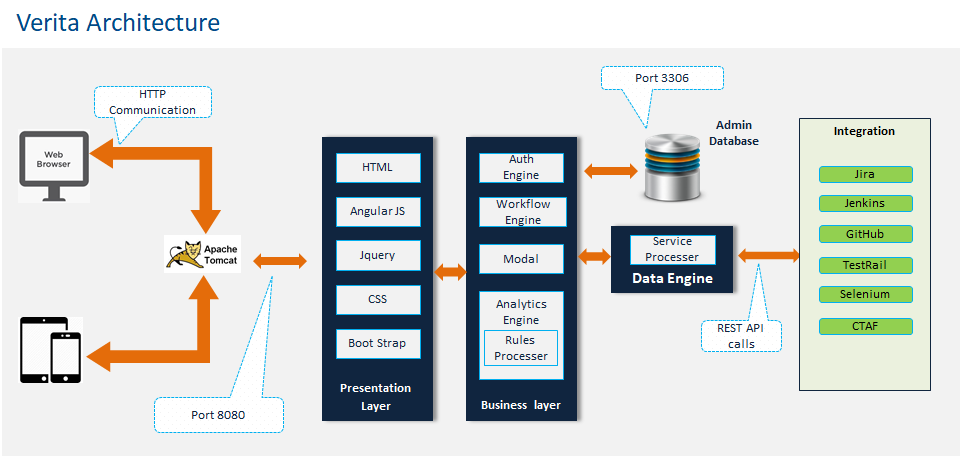
## Test Automation Farm Setup



# QA Metrics & Measurement Framework

## Verita Dashboard

Verita (Cigniti IP) comes with the ability to Quality Metrics from heterogeneous and disparate tools in the ALM ecosystem. Verita Dashboard is implemented and customized to represent the QA Metrics in form of various widgets. It is a role-based system, catering to the needs of diverse stakeholders inside the enterprise with a holistic view of Product Quality.



## QA Metrics

| Metrics | Description | Data Source |
| --- | --- | --- |
| Test Case Distribution By Application Functional Area | This metric shows the total no. of test cases distributed by application functionality | TestRail/Jira |
| Test Case Distribution By Testing Type | This metric shows the total no. of test cases distributed by testing type such as Smoke, Functional, System, Regression testing | TestRail/Jira |
| Test Case Authoring Productivity | This metric shows the total no. of test cases authored per hr. to track the rate at which test cases are authored | TestRail/Jira |
| Test Case Authoring Progress | This metric shows the total no. of test cases authored vs. total no. of test cases planned for the sprint | TestRail/Jira |
| Test Case Execution Progress | This metric shows the total no. of test cases Planned, Executed, Passed, Failed, Blocked for the sprint – iteration | TestRail/Jira |
| Production Defect Leakage | This metric shows the total no. of defects slippage to production after QA release | TestRail/Jira |
| Defect Distribution By Application Area | This metric shows how the total no. of defects distributed by Application/Module/Functionality | TestRail/Jira |
| Defect Distribution By Root Cause | This metric shows how the total no. of defects distributed by various reasons such as Requirement, Design, Configuration, Implementation, Functionality related defects | TestRail/Jira |
| Defect Distribution By Origin | This metric shows how the total no. of defects distributed by various phases of development lifecycle | TestRail/Jira |
| Defect Distribution By Class | This metric shows how the total no. of defects distributed by Severity and Priority | TestRail/Jira |
| Defect Distribution By Status | This metric shows how the total no. of defects distributed by various status such as New, Accepted, Rejected, Assigned, Open, Fixed, Retested, and Closed | TestRail/Jira |
| Test Automation Coverage By Application Area | This metric shows the total no. of automated scripts distributed by application functionality | TestRail/Jira |
| Test Automation Progress | This metric shows the total no. of automated scripts developed vs. total no. of test cases identified for automation | TestRail/Jira |
| Test Script Authoring Productivity | This metric shows the total no. of automated scripts developed per hr. to track the rate at which test cases are automated | TestRail/Jira |
| Test Automation Maintenance Effort | This metric shows the no. of hrs. spent to keep the test automated scripts up to date due to changes introduced in the application release on release | TestRail/Jira |
| Test Automation Execution Progress | This metric shows the no. of automated scripts Planned, Executed, Passed, Failed, Blocked for the sprint iteration | TestRail/Jira |

# Reporting and Communication

## Status Reporting

* Below is the list of reports to be published
  + Daily status report
  + Weekly status report
  + Monthly status report
  + Sprint end status report.

## Meetings

Below is the list of suggested meetings to communicate with required stakeholders

* Daily scrum, weekly status call, Mid Sprint, Sprint End Review meetings (Biweekly), Sprint Planning, Sprint Retrospective, Backlog Grooming
* Service Delivery Review
* Quarterly Business Review
* Annual Business Review

## Escalation Procedure and Resolution Mechanism

<Describe the escalation procedure that would be followed internally and externally in the project or at account level along with the mechanism for resolution of issues.>