TBTAF DISCOVERER

SOFTWARE QUALITY ASSURANCE PLAN

VERSION 1

NOVEMBER 25, 2015

Discoverer Team

This page intentionally left blank.

TBTAF DISCOVERER

SOFTWARE QUALITY ASSURANCE PLAN

VERSION 1

NOVEMBER 25, 2015

SQA Plan Approvals:

_____	_____
SQA Manager	**Date**
_____	_____
Project Manager	**Date**
_____	_____
Program Manager	**Date**

SECTION 1. PURPOSE

The purpose of this plan is to define the TBTAF Discoverer Software Quality Assurance (SQA) organization, SQA tasks and responsibilities; and to provide an insight on the quality assurance process of the module. This document defines a plan to assure the correct functioning and integration of the TBTAF Discoverer module. TBTAF stands for Tag Based Test Based Automation.

1.1 SCOPE

This plan will be used to apply tests with files where the Discoverer module intervenes as part of the framework, specifically searching for and extracting metadata within the .py files located in the input path or paths.

1.2 SYSTEM OVERVIEW

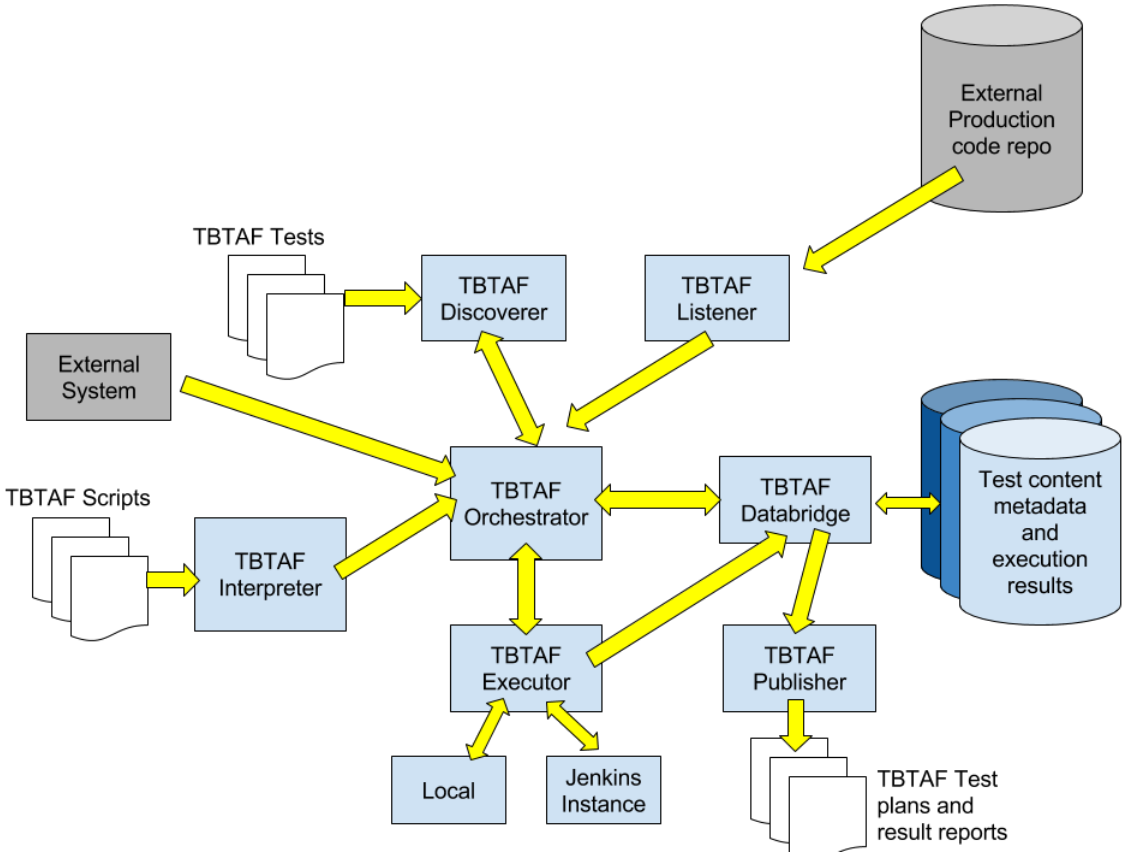
The TBTAF Discoverer is a module that is in charge of parsing test cases and extracting their metadata. This information is then passed on to the Orchestrator for further processing.

SECTION 2. MANAGEMENT

This section describes each major element of the organization that influences the quality of the software.

Development of the TBTAF System has been divide in groups of MCC students where each of them is in charge of the development and integration of each module.

Please refer the follow image of the integration of the components.



The following chart defines the SQA roles and responsibilities of the members of the Discoverer team and their function

Role	**Name**
SQA Mgr	Jose Gerardo Padilla
Prog Mgr	Pablo Sainz

Proj Mgr	Jose Gerardo Padilla
SW Dev1.	Fernando Ramirez Garibay
Sw Dev2.	Alberto Velasco Vasquez.
Sw Dev3.	Paola Mariana Vicencio Hernandez
Sw Test1.	Fernando Ramirez Garibay
Sw Test2.	Alberto Velasco Vasquez.
Sw Test3.	Paola Mariana Vicencio Hernandez

SECTION 3. SQA TASKS

Describe the portion of the software life cycle covered by this SQA Plan, the tasks to be performed with special emphasis on SQA activities, and relationship between these tasks and the planned major checkpoints. The sequence of the tasks should be indicated. Tailor this section to reflect those tasks being verified that relate to the project's current/projected activities.

The SQA tasks are described in the following tables:

SQA Plan	**SQA Mgr**	**Prog Mgr**	**SW Dev**	**SW Test**
Develop/Document SQA Plan	X			
Review SQA Plan	X	X	X	X
Approve SQA Plan	X	X		
Review products	X	X	X	X
Approve product		X		
Develop/Document SDP and other project plans (Test Plan, Training Plan, Computer Resource Life Cycle Management Plan (CRLCMP))		X		
Review plans	X	X	X	X
Approve plans		X		
Evaluate PPT&O Process	X			
Resolve Audit Findings		X		
Develop/document Sys Rqmts		x		
Review Sys Rqmts	X	X	X	X
Approve Sys Rqmts		X		
Evaluate/report Sys Rqmts Analysis Process	X			
Resolve Audit Findings		X		

Software Design Process	**SQA Mgr**	**Prog Mgr**	**Proj Mgr**	**SW Dev**	**SW Test**
Develop/document SW Design				X	X
CM SW Design					
Review SW Design	X	X	X	X	X
Approve SW Design		X	X		
Maintain SDL and SDFs				X	
Evaluate/report SW Design Process	X				
Resolve Audit Findings		X	X		
Software Implementation & Unit Testing Process	**SQA Mgr**	**Prog Mgr**	**Proj Mgr**	**SW Dev**	**SW Test**
Develop/fix code				X	
Code review	X			X	X
Unit Test				X	X
Unit Integration and Testing Process	**SQA Mgr**	**Prog Mgr**	**Proj Mgr**	**SW Dev**	**SW Test**
Integrate SW				X	
Test Integrated SW					X
Fix errors				X	
CI Qualification Testing Process	**SQA Mgr**	**Prog Mgr**	**Proj Mgr**	**SW Dev**	**SW Test**
Performance Test					X
Fix errors				X	
Maintain SDL and SDFs				X	X

SECTION 4. STANDARDS, PRACTICES, CONVENTIONS AND METRICS

Identify the standards (mandatory requirements) to be applied. State how compliance with these items is to be monitored and assured.

- Document standards – Wiki Portfolio
- Coding standards – Python as main language for development
- Coding Documents standards – Python Documentation
- Test Standards – IEEE Standard for software test documentation

4.1 METRICS

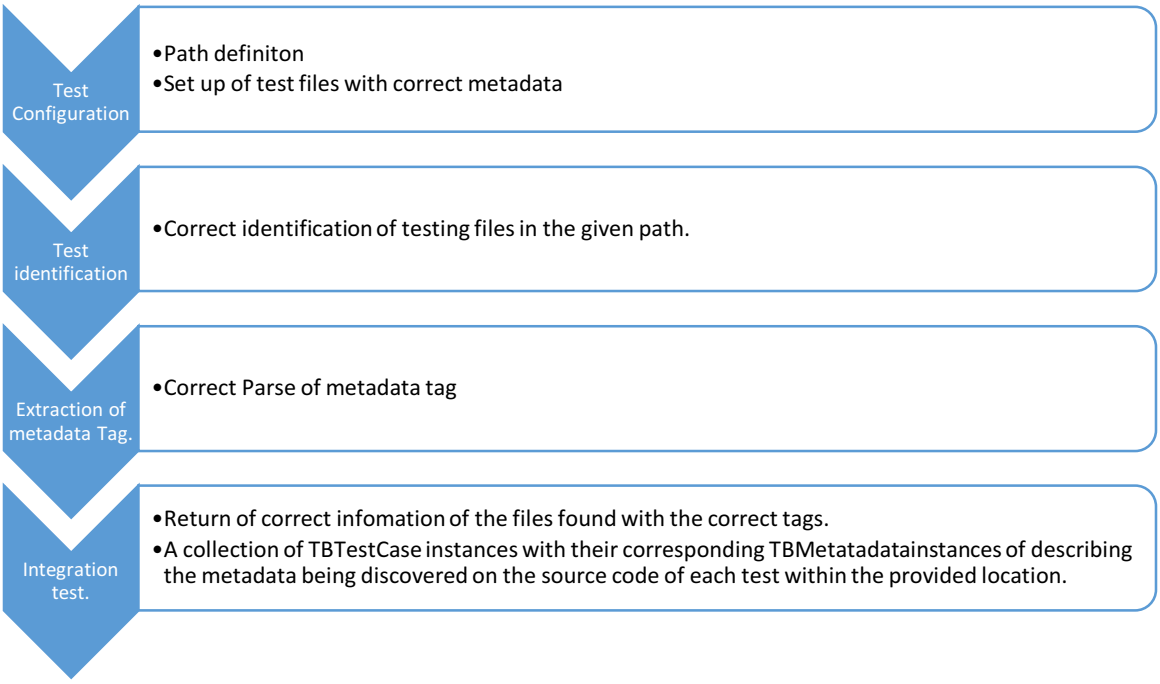
- LOC - lines of code is used to measure the size of the software
- Unit tests per module.
- Number of aserttions.

SECTION 5. TEST

Identify all other tests not included in verification and validation and state the methods used. Describe any testing techniques or methods that can be used to detect errors, to develop sets of test data, and to monitor computer system resources.

TBTAF test includes integration for each module.

For the Discoverer testing please refer to the follow image and the layers it will cover.



SECTION 6. TOOLS, TECHNIQUES, AND METHODOLOGIES

Identify the special software tools, techniques, and methodologies that support SQA, state their purposes, and describe their use.

Tools - SQA software tools include, but are not limited to, operating system utilities, debugging aids, documentation aids, checklists, structuring preprocessors, file comparators, structure analyzers, code analyzers, standards auditors, simulators, execution analyzers, performance monitors, statistical analysis packages, software development folder/files, software traceability matrices, test drivers, test case generators, static or dynamic test tools, and information engineering CASE tools.

Techniques - techniques include review of the use of standards, software inspections, requirements tracing, requirements and design verification, reliability measurements and assessments, and rigorous or formal logic analysis.

Methodologies - methodologies are an integrated set of the above tools and techniques. The methodologies should be well documented for accomplishing the task or activity and provide a description of the process to be used.

Four main tools were used for the development of discoverer:

- Git: tool responsible for the version control part of the project.
- Python 2.7: programming language used to develop the module.
- Eclipse: development environment used to develop the Discoverer module and test its functionality.
- PyDev: plugin for the Eclipse IDE that allows writing code in Python.

SECTION 7. CODE CONTROL

The purpose of this section is to define the methods and facilities used to maintain, store, secure and document controlled versions of the identified software during all phases of the software life cycle whose appropriate use will be verified by SQA. This may be implemented in conjunction with a computer program library. This may be provided as a part of the SCM Plan. If so, an appropriate reference should be made.

TBTAF Discoverer uses the project's GitHub repository for code control. All code changes are committed to this location and stored for use of the other team members in charge of the development of this module. To use the repository, the following process was followed:

- Perform a pull from the Git repository before starting to work on the code.
- After finishing coding, check the Git repository for the last check in date to make sure no additional commits have been done while developing.
- If a commit was done, check the changes and merge them.
- If no commit was done, commit the new code.

This ensured that all developers were working with the latest version of the module that had already been downloaded. SQA activities for this purpose require an audit on the pull – commit process to make sure it is followed according.

SECTION 8. SUPPLIER CONTROL

The purpose of this section is to state the provisions by which SQA assures that software provided by suppliers meets established requirements.

In some cases, projects do not foresee purchasing software. If that's the case, the following paragraphs may apply.

All development software has been acquired with external suppliers. It was selected based on preferences of the developers. Although it was not specifically tested for the project, it is widely known that these tools are sufficient enough to develop the project.

SECTION 9. QA CHALLENGES

The purpose of this section is to describe the possible challenges that will be faced during the quality assurance process of the module.

The main challenge for the quality assurance of the discoverer module lies on the integration part of the project. Since the development teams for all different components of the TBTAF framework are spread out over all course members, coordination is both crucial and difficult to achieve. Because of the heterogeneity of the development groups the development of the different modules was asynchronous, which leads to some potential problems arising from miscommunication or the lack thereof.

Another challenge that is present when trying to assure the quality is the expectations of the module. Due to the nature of the project and the collaborative development, delivering a robust, quality module does not suffice; the type and form of the results thrown by the module are essential to the success of the project. Even if we can make sure the Discoverer module delivers a certain output with quality, it would not matter if it is not what the "client" components are expecting. This is why communication between module-development teams is essential, as the module has to be able to deliver a quality output in both type and form.

The last challenge is how to provide prompt feedback to other development teams if the other modules are causing quality problems to other modules. Communication should be fast and concrete to make sure any problems are addressed on time for the delivery of the final product

SECTION 10. QA ASSUMPTIONS

The purpose of this section is to describe the assumptions taken for the quality assurance work of the Discoverer module.

There are a few assumptions regarding the quality assurance process of the Discoverer module. First, it is assumed that the information on the TBTAF Wiki is followed closely by all development teams. This ensures that both the inputs and outputs defined by the specification are followed, reducing the risk of incompatibilities between modules.

Another assumption is that all modules of the TBTAF framework will be written using the Python programming language. All files prepared to test the new modules will be used by the developed components, creating common ground on how the modules should work together.

SECTION 11. QA TACTICS

The purpose of this section is to describe the tactics that will be followed to assure quality on the Discoverer module.

There are a number of activities to be performed in order to ensure that quality assurance takes an important place in the development of the Discoverer module. These activities are:

- Record and analyze the results of all testing done. This includes unit, module, and integration testing.
- All module tests must be performed by each of the development team members (all three of them)to make sure results are more viable.
- Communicate the results of quality assurance measures to developers of the other modules. This is especially important with the Orchestrator module development team, as they are directly affected by the quality of the Discoverer module.

SECTION 12. INTEGRATION TACTICS

The purpose of this section is to describe the tactics to be used during the integration of the Discoverer module as part of the TBTAF framework.

A plan was developed in order to complete the integration part of the module. This plan consists in completing a series of tasks that will lead to a full integration of the discoverer module with the rest of the components of the TBTAF framework.

The steps of the plan are:

- Analyze and study the Wiki description of the Discoverer module, gathering as much information as possible on the inputs and outputs of the component.
- Organize meeting with Program Manager to discuss any questions regarding the design and implementation of the Discoverer module.
- Discuss internally within the module development team on the best approach to developing the module.
- Meet with the Orchestrator development team to discuss and agree on the type and form of the outputs to be delivered to their module by the Discoverer.
- Perform all development activities.
- Perform all testing of the Discoverer module.
- Meet with the Orchestrator development team to perform integration tests.

Integration tests will include:

- Testing Orchestrator behavior with correct inputs.
- Testing Orchestrator behavior with incorrect inputs.
    1. Incomplete inputs
    2. Wrongly-formatted inputs
    3. Inputs of the wrong type
- Testing Orchestrator with mistimed inputs
- Stress testing