Chapter 2

JSystem Automation Framework General Description

➢ **In this chapter…**

2.1 JSystem Automation Framework General Overview

The JSystem Model comprises the, System Under Test (SUT), Automation Layers and Framework Services.

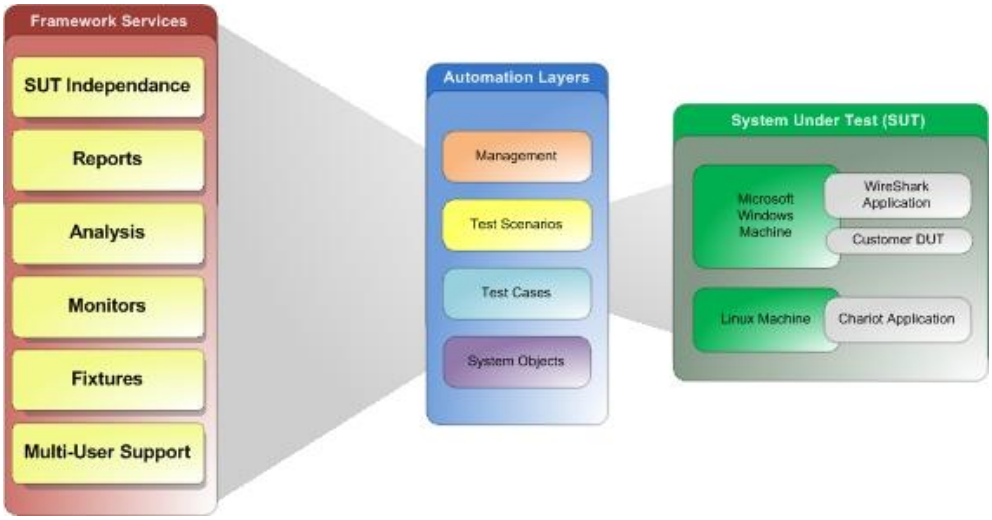

Figure 1: JSystem Model

2.1.1 System Under Test (SUT)

The system under test (SUT) is the user testing environment. When planning an automation project one of the first steps required is to define the SUT. The SUT is the device or software that is being tested, as well as the equipment and applications that participate in the setup.

The test environment can include the following items:

- Device and or software being tested.
- Workstations or PC's that are part of the setup and require management.
- Traffic generator.
- Tools used to interface with the device and or software that are being tested, such as database utilities.

2.2 The Four Layers of Automation

The JSystem Automation Framework refers to automation layers; these layers are the component elements within the architectural framework of a system. System testing is a complicated process that requires a robust and intelligent solution.

JSystem divides the testing process into four layers that simplifies and streamlines the testing process; these layers are defined as follows:

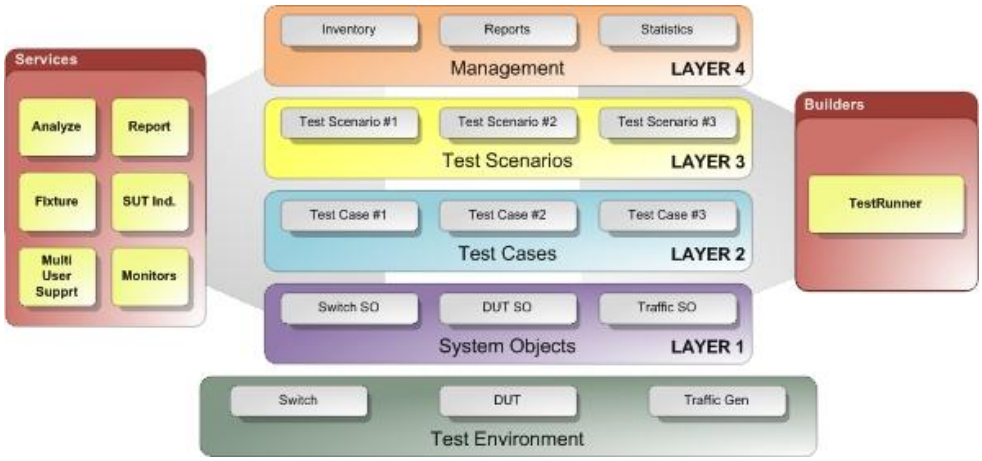

Figure 2: The Four Layers of Automation Diagram

The diagram visually displays the interaction between the four automation layers:

1. The SystemObject layer.
2. The Test Case layer.
3. The Test Scenario Layer.
4. The Management Layer.

2.2.1 SystemObject Layer 1

The first layer of JSystem is the SystemObject (or Driver) layer. For every managed object there is a SystemObject. The SystemObjects are written in Java and form the management layer that controls the SUT.

2.2.1.1 The SystemObject

Hides Complexity - This layer acts as a mediator that exposes the SUT to the Tests layer applying a simple user friendly interface that "**speaks**" in the language of the QA engineer. All the connectivity issues are hidden from the person who writes the tests.

Reporting - Every SystemObject operation (method) leaves a remark, reporting its action. In some cases it collects information for analysis.

Management Error Handling - If a step fails the test framework receives an exception directly circumventing the tests themselves.

Analyzes Results - The SystemObject layer enables a "**unite**" mechanism to analyze results.

2.2.2 Tests Cases Layer 2

The Tests layer in JSystem is based on and extends the JUnit, that is, a JSystem test is a JUnit test with extensions and or additions. Once the SystemObjects have been written, preparing the tests is a simple process, usually performed by a QA engineer with a programming orientation.

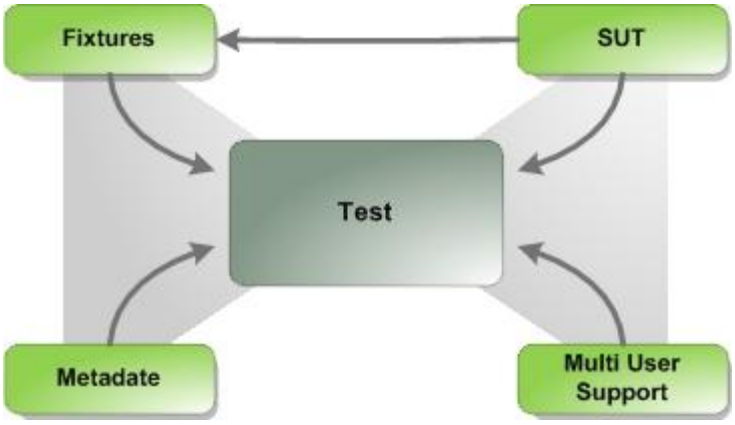

Figure 3: Second Layer Test Illustration

Fixtures are used to configure the SUT. Fixtures are Java classes that are responsible for bringing the system to the state required by the test. By using fixtures, the tests author can reuse configuration code and separate it from testing code.

Note: Once the SystemObjects have been written preparing and writing tests is a simple process

> ***Reference: In order to maintain a streamlined writing process the QA Engineer should refer to the JSystem recommended guide lines,*** *Error! Reference source not found.**, on page** Error! Bookmark not defined..*

The guidelines for writing tests are as follows:

- **Keep it simple -** Do not complicate the test logic. Each test should be designed for a specific SUT. Do not design tests to run on a variety of SUT's.
- **Keep it short -** It is preferable to have a large number of shorter tests rather than a small number of longer, complicated tests.
- **Use fixtures -** Use fixtures to configure your SUT, and share the fixtures between tests.
- **Use Reports and Documentation -** If the tests are well documented, analyzing scenario execution results will be faster. It is crucial that you plan your project in such a way that it can scale to hundreds and thousands of tests.
- **Start with an Easy Test -** Start with simple and stable tests. Do not start to automate the most complex part of your system first.

2.2.3 Test Scenarios Layer 3

Tests are grouped together in a hierarchical manner within a scenario. These JSystem scenarios are written as Ant scripts in addition to grouping the tests; the user can parameterize a test, control the Java Virtual Machine (JVM) on which the test are activated, manage scenarios, execute scenarios, analyze scenario execution results and publish the results to a central reporting server.

2.2.4 Management – Layer 4

The Scenario manager is an application that can be run from any desktop computer. The management application is used to group tests into scenarios, and manage these scenarios. In addition, the user can also set parameters for tests, execute the scenarios and then analyze the scenario execution results. The results can then be published to a central reporting server.

2.2.4.1 Full Centralized Report System

The complete JSystem Automation Framework solution includes a centralize reports server. The report server is a J2EE based application that includes a database.

The report application enables the user to access the scenario execution results from any workstation as well as view the historical execution results, compare between scenario executions and run a wide range of reports that produce graphs and charts.

2.3 Introduction to Framework Services

The JSystem framework services are divided into six building blocks that provide the simple and exact functionality required to perform all testing operations required by QA engineers, they are:

1. **SUT Independence** – SUT Independence is the ability to run the same test on different setups without changing the test.
   - **SystemObject lifecycle** - This service includes the ability to control the creation, initiation and destruction of system objects.
2. **Reports** - The reporting mechanism enables the user to quickly analyze the cause of any test execution failure.
3. **Analysis** – Analyzers provide an easy way to verify SystemObjects operations. The SystemObject operations are used to manage and or receive information from the SUT.
4. **Monitors** – Monitors are processes that are also referred to as threads; these threads are defined per test.
5. **Fixtures** - The fixture is a special type of procedure that brings the SUT to a specific configuration state. The fixture enables the tests to share system configurations.
6. **Multi User Support** – The JSystem Automation Framework supports the ability to define tests according to parameter.

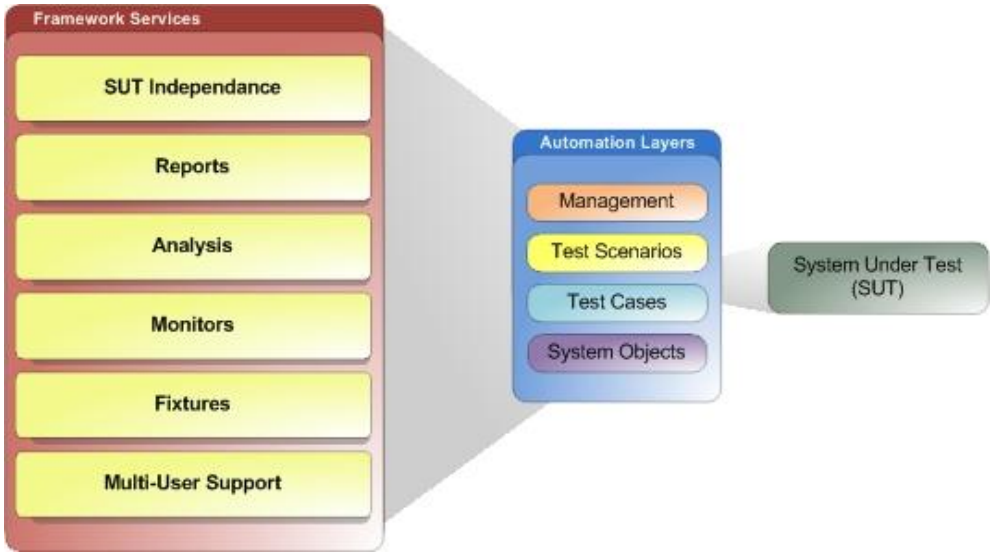

Figure 4: Framework Services

Reference: For a further in-depth understanding of the framework services, refer to "Chapter 4 JSystem Framework Services"