Chapter 12

JSystem Automation Framework Automation Testing Guidelines

➢ **In this chapter…**

Automation Testing Guidelines Overview	*Page 2*
The Golden Rules of Testing	*Page 2*
User Profiles	*Page 3*
Basic Test Design Considerations	*Page 4*
The Automation Life Cycle	*Page 5*
Test Plan Environment	*Page 6*
Integration	*Page 6*
Scalability and Maintenance	*Page 7*

12.1 Automation Testing Guidelines Overview

The following section details the guidelines and procedure required to design and build automated testing environments.

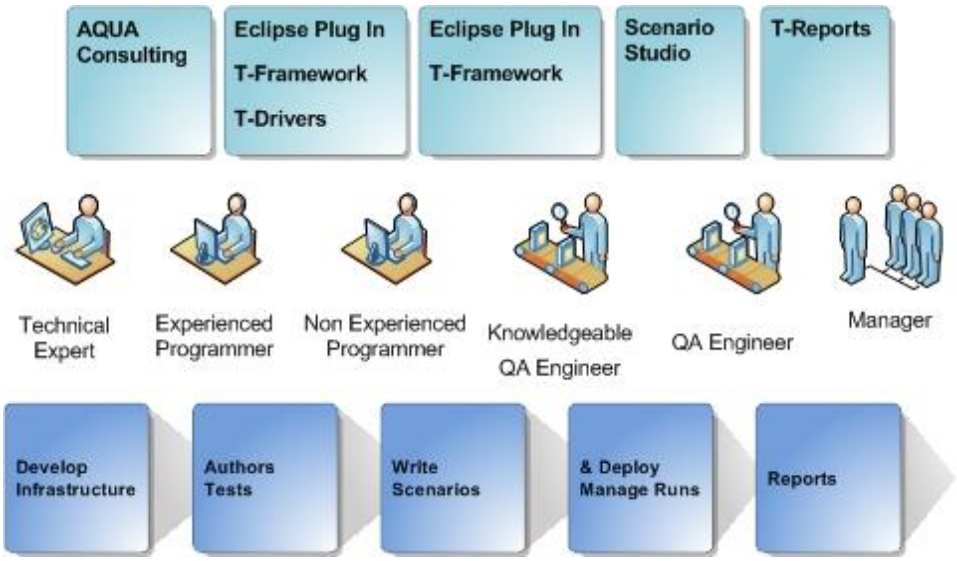

Figure 1: Automation Test Basic Life Cycle

The diagram illustrates the JSystem automation workflow of a life cycle test. The illustration also details the user profiles task division and thought logic behind the development and test authoring stages and the test environment.

12.1.1 The Golden Rules of Testing

The following rules are a set of guide lines designed to streamline the automation process:

- **Design, don't Over Design** – Make sure to keep the design simple so that the programming process does not become too long causing it to become obsolete.
- **Compatibility** – Always consider backward compatibility.
- **Welcome Change** – Accept and integrate changes in project requirements at any stage of the development process.
- **Considerations** – Do not forget the inter-test / inter-setup relationship.
- Keep it Readable and Simple
- Stay Synchronized
- Integrate Constantly
- Be Flexible

12.1.2 User Profiles

JSystem Automation Framework refers to user profile definitions to clearly define the design and task requirements of a standard testing environment. These user profiles define the logical breakdown of the JSystem Automation Framework structure, and clearly define the automation life cycle.

- **Technical Expert** – an experienced engineer with a background in automation and QA testing. The technical expert deals with system QA, and assists in the design of the testing framework. Writes the tasks and defines what will be checked, the checking order and the tests that are performed. In a typical work environment this position is filled by the CTO of a company or an experienced QA engineer.
- **Experienced Programmer** – Experienced in java programming, the main task is to write the **SystemObjects** (Drivers) that will be used to interact with the system under test (SUT)
- **Non Experienced Programmer** – requiring a minimal understanding of the test environment and java programming, the main task is to write the java tests.
- **Knowledgeable QA Engineer** – requiring a robust understanding of the test environment, the main task is to assemble the building blocks of java code and drivers into scenarios. These scenarios are built in the JRunner application.
- **QA Engineer** – operates the JRunner application by selecting scenarios and running them.
- **Manager** –the main task is to track the test deployment and monitor the automation configuration of the test environment structure.

12.1.3 Basic Test Design Considerations

The basis for the design and extraction requirements of the test environment is first calculated by ascertaining the set up needs of the system. The Programmer then calculates whether run-time setup-swapping is required.

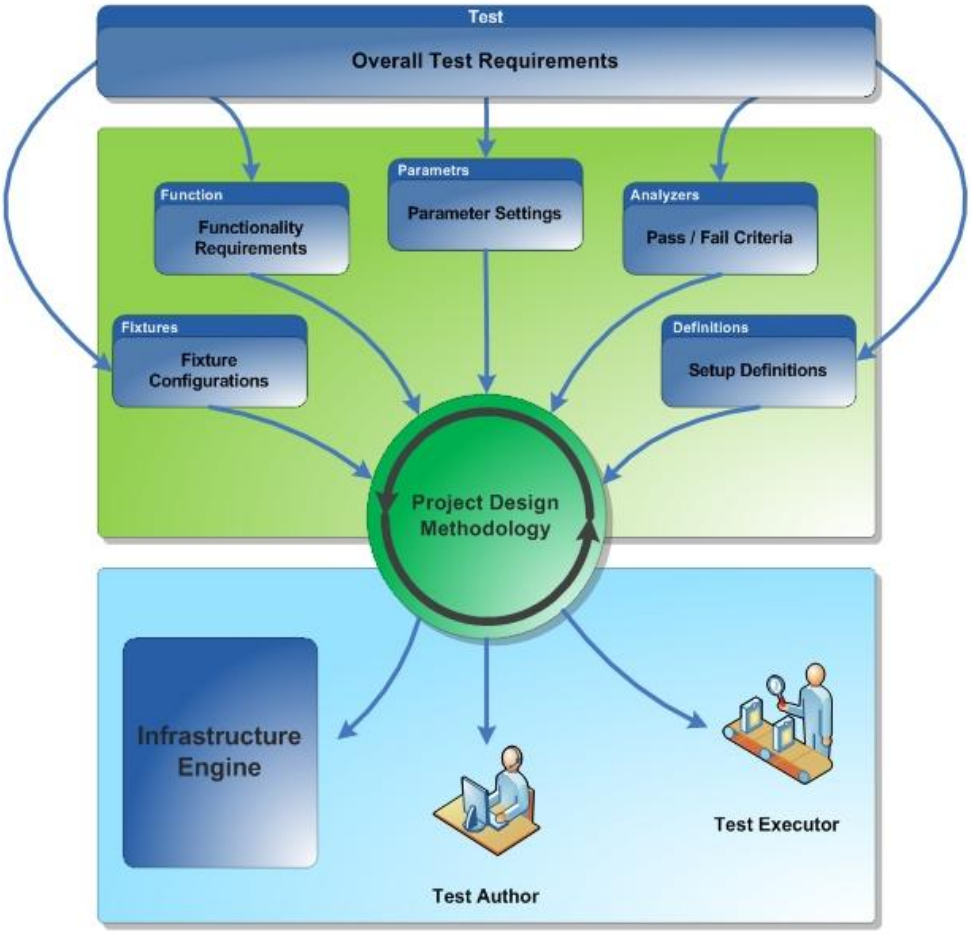

Figure 2: Basic test Design Considerations

After the preliminary design requirements have been decided upon, the programmer then configures the fixtures. The system objects (drivers) functionality requirements are then defined, and the degree of user freedom is set via the parameter settings.

Certain allowances must be made when designing tests to also take into consideration the fact that the test must be able to run on different devices and different CLI versions. The analyzer definitions are then set.

Note: When setting the test criteria, at least one or more clear pass or fail criteria must be included.

When designing a test the inter-test relationship must be taken into consideration. This includes pre-configuration and configuration cleaning (roll back).

12.1.4 The Automation Life Cycle

The automation life cycle can be divided into four main stages of automation project development, Module, Assembly, Deploy and Manage.

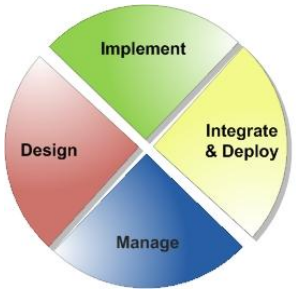

Figure 3: Automation Stages

The illustration below details the four development stages of an automation cycle. It shows the chronological order of operations required to develop the module, showing the different interactive relationships of certain elements within the development process.

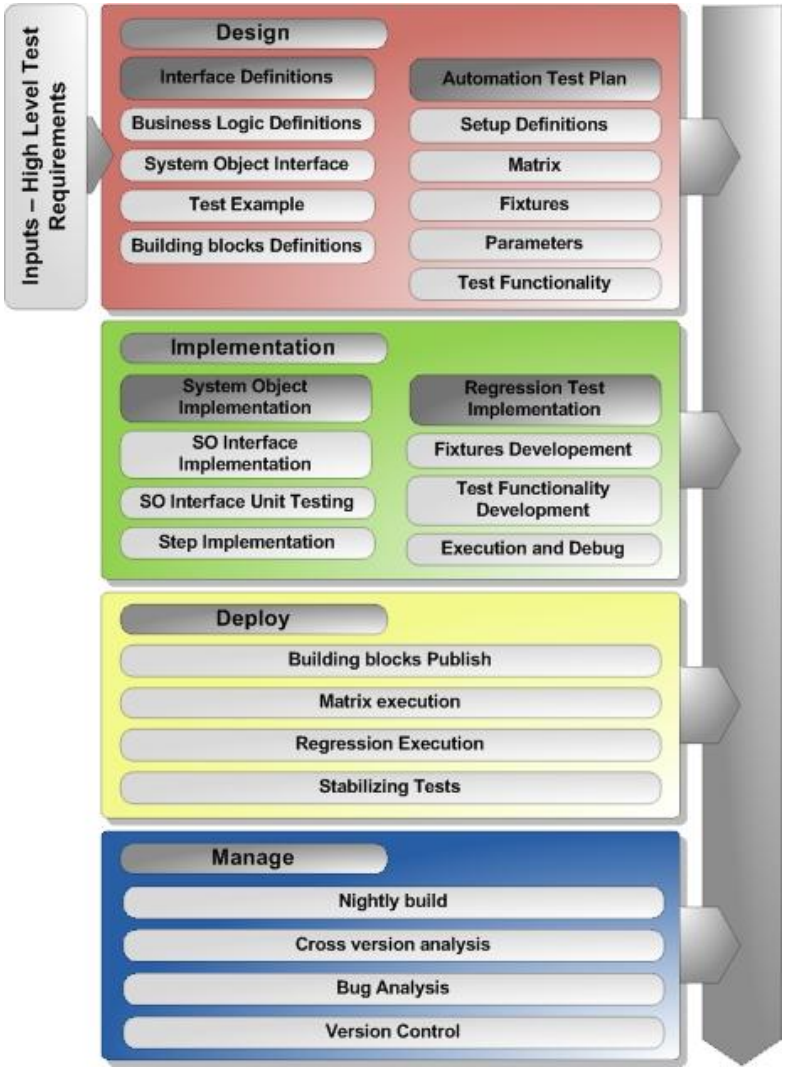

Figure 4: Detailed Automation Life Cycle

12.1.5 Test Plan Environment

The test plan is the basis for all the design structure and architectural elements within the automation project. When designing an automation project certain project framework aspects should be kept in mind.

- **Do not share the automation set up –** Sharing the automation setup on more than one machine can cause a system failure because manual overrides and or changes made on one machine can affect tests being automatically run via another machine.
- **Upgrading Setup** - When a set up is upgraded always consider backward compatibility.
- **The Mutual Aspects between the Test and the Setup** – Automatic tests are not executed in a vacuum, the programmer must define the setup presets for example, return to factory default presets. Then define the root fixture maintaining consistency and predictability and making sure that the setup status is known at all times. Finally, consider what test should or should not the change in the set up.

12.1.6 Integration

Once the product has been developed and the tests have been written for the product in the development environment, the product is ready for testing. The product and tests now enter the QA lab environment, the QA engineer then begins the process of product setup, whereby the product and the relevant tests are setup. This process is referred to as product integration. The QA Testing engineer should also keep the following in mind when preparing the integration of the product and the tests:

- **Leave time for unpredictable aspects:**
  - Moving from a manual to an automatic test run is not trivial issue.
  - When applying an unattended run, no human logic can be applied.
- **Start running overnight scenario as soon as possible** – Applying this simple principle will save a lot of integration time.
- **Source Control:**
  - Always stay synchronized between the **Ignis JSystem** Automation Framework environment and the internal environment.
  - Always format the code before commitment, making sure to check file compatibility awareness.

12.1.7 Scalability and Maintenance

When creating tests the user should ensure that the tests are short, well documented, use consistent naming conventions and that the code is neatly written and conventionally formatted. It is preferable to write a larger amount of short atomic tests rather than large elaborate tests.

Further considerations when writing tests is to maintain a linear flow of steps, making sure that the test follows a logic progression. The following points should be adhered to in order to streamline the writing process.

- Do your best to avoid meta data usage
- Fractional debug analysis
- Very complicated re-construction processes
- Version control enemy