Automating Software Builds

Advantages of Automating software build application:

* A necessary pre-condition for continuous integration and [continuous testing](https://en.wikipedia.org/wiki/Continuous_testing)
* Improve product quality
* Accelerate the compile and link processing
* Eliminate redundant tasks
* Minimize "bad builds"
* Eliminate dependencies on key personnel
* Have history of builds and releases in order to investigate issues
* Save time and money - because of the reasons listed above.[5]

**Automation Testing?**

Automation testing can be defined as a way to run a set of tests over and over again without having to execute them manually. Introducing automation tests in your test strategy is a way to save money and time.

### Automation Based on the Type of Testing

**Automation of Functional Tests:**

Functional tests are written to test the business logic behind an application. Automating these mean writing scripts to validate the business logic and the functionality expected from the application.

**Automation of Non-Functional Tests:**

Non-functional tests define the non-business requirements of the application. These are the requirements related to performance, security, databases, etc. These requirements can remain constant or can be scaled as per the size of the software.

### Automation Based on the Phase of Testing

**Automation of Unit Tests:**

These tests are run during the development phase itself, ideally by the dev after the completion of development and before handing over the system to the testers for testing.

**Automation of API Tests:**

API tests are run during the integration phase. These may be run by the development or testing team and can be run before or after the UI layer is built for the application. These tests target the testing based on the request and response on which the application is built.

**Automation of UI based tests:**

UI Based tests are run during the test execution phase. These are specifically run by the testers and are run only once before the UI of the application is handed over to them. These test the functionality and business logic of the application from the front end of the application.

### Automation Based on the Type of Tests

**Unit Tests:**

Unit Tests are the tests that are built to test the code of an application and are usually built into the code itself. They target the coding standards like how the methods and functions are written.

These tests are more often written by the developers themselves, however, in today’s world, automation testers may also be asked to write them.

Executing these tests and getting no bugs from them will mean that your code will compile and run without any code issues. These tests usually do not target the functional aspects of the application and as they target code, it is more appropriate to automate them so that they can be run as and when required by the developer.

**Smoke Tests:**

The smoke test is a famous test performed in the test life cycle. These are post-build tests, they are executed immediately after any build is given out of the application to ensure that the application is still functioning after the build is done.

This is a small test suite and is something that will be executed multiple times and thereby it makes sense to automate it. These tests will usually be of a functional nature and depending on the type of application a tool can be picked for them.

**API tests:**

API testing has become very famous in the past few years. Applications built on the API architecture can perform this testing.

In API testing, the testers validate the business layer of the application by checking the request-response combinations for the various API’s on which the application is built. API Tests can also be done as a part of the integration tests below.

**Integration Tests:**

Integration test as the name itself suggests means testing the application by integrating all the modules and checking the functionality of the application.

Integration testing can be done through API testing or can be done through the UI layer of the application.

**UI tests:**

UI tests are done from the UI layer or the frontend of the application. These may target testing the functionality or simply test the UI elements of an application.

Automating the UI to test the functionality is a common practice. However, automating the GUI features is one of the more complicated automation.

**Regression tests:**

One of the most commonly automated test suites is the regression test suite. Regression, as you may already know, is the test that is done at the end of testing a new module to ensure that none of the existing modules have been affected by it.

It is repeated after each new iteration of testing and the main test cases stay fixed with usually a few new additions after a new iteration. As it is frequently run almost all the test teams try to automate this pack.

**Automation as Continuous integration:**

Continuous Integration may again be running on the automated regression tests itself, however, in achieving CI, we enable the regression or identified test suite to be run every time when a new deployment is done.

**Security Tests:**

Security testing can be both functional as well as a non-functional type of testing which involves testing the application for vulnerabilities. Functional tests will compose of tests related to authorization etc., whereas non-functional requirements maybe test for SQL injection, cross-site scripting, etc.

**Performance Tests and Quality control:**

Performance tests are non-functional tests which target the requirements like testing of load, stress, scalability of the application.

**Acceptance tests:**

Acceptance tests again fall under functional tests which are usually done to ensure if the acceptance criteria given by the client has been fulfilled.

So far, we have described the type of tests that can be automated and various classifications of the same, all classifications eventually will lead to the same end results of a test suite being automated. As we said earlier a little understanding is required on how these are different from frameworks.

Once you have identified the tests that you want to automate from the above classification, then you will need to design your logic in a manner to execute these tests smoothly, without much manual intervention. This design of a manual test suite into an automated test suite is where the frameworks come in.

**Now we will explore Top 3 Test Automation Types**

1. Unit Testing
2. API Testing
3. GUI Testing

**#1)** **Automated Unit Tests**

[Automated Unit tests](https://www.softwaretestinghelp.com/unit-testing/) are written to test the code level. Bugs are identified in the functions, methods, and routines written by the developers.

Some companies ask the developers to do the unit testing themselves and some hire specialized test automation resources. These resources have access to source code and they write unit tests to break the production code.

Due to the presence of unit tests, whenever the code compiles, all unit tests run and tell us the result that if all the functionality are working. If any unit test fails, then it means that there is now a bug present in the production code.

Some of the most popular tools present in the market include [NUnit](http://www.nunit.org/" \o "NUnit) and [JUnit](http://junit.org/). Microsoft also provides its own framework for unit testing called [MSTest](http://msdn.microsoft.com/en-us/library/ms243176.aspx" \o "MSTest). Go through the websites of these tools and they will provide you more examples and tutorials on how to write unit tests.

**#2)** **Automated Web Service / API Tests**

An Application Programming Interface (API) makes it possible for the software to talk to other software applications. Just like any other software, APIs need to be tested. In this type of testing, GUI is usually not involved.

What we test here is usually the functionality, compliance and security issues. In web applications, we can test the Request and Response of our application that if they are secure and encrypted or not.

This is one of the examples where we can use API Testing. The most popular tool for API testing is [SOAPUI](http://smartbear.com/products/qa-tools/web-service-testing-tool/) which has both free and paid versions. There are other tools as well, which you can use according to your need.

**#3)** **Automated GUI Tests.**

This type of automated testing is the toughest form of automation as it involves testing of a User interface of the application.

It is tough as the GUI’s are highly subject to change. But this type of testing is also closest to what the users will do with our application. As the user will use the mouse and keyboard, automated GUI tests also mimic the same behavior by making use of mouse and keyboard to click or write to objects present on the user interface.

Due to this, we can find bugs early and it can be used in many scenarios such as regression testing or filling up forms which takes too much time.

The most popular GUI testing tools include [Micro Focus Unified Functional Testing (UFT)](https://www.microfocus.com/en-us/products/unified-functional-automated-testing/overview), [Selenium](http://docs.seleniumhq.org/), [Test Complete](http://smartbear.com/products/qa-tools/automated-testing-tools/) and [Microsoft Coded UI](http://msdn.microsoft.com/en-us/library/dd286726(v=vs.100).aspx) (which is a part of Visual Studio ultimate and premium editions).

Just like the types of automation tests, there are multiple types of frameworks as well.

## What Is Front end Testing?

Front-end Testing is testing Graphical User Interface (GUI), functionality and usability of website or application. The main aim of Frontend testing is to ensure frontend is defect free with successive updates. Apart from this Frontend testing is conducted for:

* CSS Regression Testing: Minor CSS changes that break the frontend layout
* Changes to JS files that make the frontend non-functional
* Performance Check