##### REGRESSION TESTING :

*1st build –* Customer gives requirements – development team start developing features – testing team start writing test cases – testing team write about 1000 test cases for the 1st release of the product and after execution of the test cases – the product is released – customer does acceptance testing – and the product is moved to production.

*2nd build –* now, customer asks for 2extra features to be added and gives the requirements for the extra features – development team start building the extra features – testing team start writing test cases for the extra features – about 200extra test cases are written – thus a total of 1200 test cases are written for both the releases – now testing team – start testing the new features using the 200 new test cases – once that‟s done, then start testing the old features using the old 1000 test cases to check if adding new features has broken the old features. Testing old features is called regression testing. Once everything has been tested, now the product is given to the customer who does acceptance testing and then moves the product to production.

*3rd build –* after the 2nd release, the customer wants to remove one of the features (say Loans) – he removes all the *Loans* related test cases (about 100) – and then tests all the other features to check if all the other features are working fine. This is called regression testing.

###### Talk 1

Testing the unchanged features to make sure that it is not broken because of the changes (changes means – addition, modification, deletion or defect fixing)

***Talk 2***

Re-execution of same test cases in different builds or releases to make sure that changes (addition, modification, deletion or defect fixing) are not introducing defects in unchanged features.

###### When the development team gives a build, chances are there they would have done some changes. That change might affect unchanged features. So, Testing the unchanged features to make sure that it is not broken because of the changes is called Regression Testing.

Majority of time spent in testing is on regression testing.

Based on changes, we should do **different types of regression testing,**

* Unit Regression Testing
* Regional Regression Testing
* Full Regression Testing

###### Unit Regression Testing (URT)

Here, we are going to test only the changes.

In Build B01, a bug is found and a report is sent to the developer. The developer fixes the bug and also sends along some new features developed in the 2nd build B02. The TE tests only if the bug is fixed.

##### For ex,

**CREATE USER**

**Name**

**Address**

**Telephone Number Email Id**

**….**

**….**

**….**

**….**

**CANCEL**

**SUBMIT**

When developer gives the above application for testing in the 1st build – the TE finds that clicking on the **submit** button goes to a blank page – this is a bug and is sent to the developer for defect fixing – when the new build comes in along with the defect fixes – the TE tests only the **submit** button. Here we are not going to test the other features of the 1st build and move to test the new features sent in the 2nd build. We are sure that fixing the **submit** button is not going to affect other features – so we test only the fixed defect.

###### Testing only the modified features is called Unit Regression Testing.

Let us consider another **example,**

**CANCEL**

**SEARCH**

**(Search field) 1 – 20 characters**

###### Build 1 – B01 Build 2 – B02

For the above application, in the 1st build – the developers develop a “**search”** field which accepts *1-20 characters.* The TE test the search field using test case design techniques.

Now, the customer makes some changes in the requirements and requests that the “**search**” field be able to accept *1-40 characters*. The TE tests only the search field to see if it accepts 1-40 characters and doesn‟t test for any other feature of the 1st build.

###### Regional Regression Testing (RRT)

Testing the changes and impact regions is called Regional Regression Testing.

##### Build 1 – B01

**Impact Areas Changes in B02**

The module 1,2,3,4 is given by developers for testing during the 1st build. The TE finds a defect in Module 4. The defect report is sent to the developers and the development team fixes the bug and sends the 2nd build

in which the bug is fixed. Now, the TE realizes that defect fixing in module 4 has impacted some features in module 1 and 3. So, the TE first tests module 4 where the bug has been fixed and then tests the impact areas i.e, module 1 and module 3. This is known as *regional regression testing*.

###### Story 1

After the 1st build, the customer sends some changes in requirement and also to add new features to be added to the product. The requirements are sent to both development team and testing team.

The development team starts making the changes and also building the new features as per the requirements.

Now, the test lead sends a mail to the customer asking – which and all are the **impact areas** that will be affected after the necessary changes are made – so that he will get an idea as to which and all features needed to be tested again. He also sends a mail to the development team to know which and all areas in the application will be affected as a result of the modifications and additions of features. And similarly he sends a mail to his testing team for a list of impact areas. Thus he gathers **impact list** from the customer, development team and also the testing team.

This **impact list** is sent to the all testing engineers who look at the list and check if their features are modified and if yes they then they do regional regression testing. The **impact areas** and changed areas are all tested by the respective engineers for whom the features are allotted. Each TE tests only his features which could have been affected as a result of the changes and modifications.

The problem with the above method is that the test lead may not get the full idea of the impact areas because the customer and development team may not have so much time to respond to his emails.

###### Story 2

To solve the above problem (story 1), we do the following.

Whenever a new build comes in along with new features and defect fixes. The testing team will have a standing meeting – they discuss if their features are affected by the above changes and thus they themselves do impact analysis and come up with the impact list where maximum possible impact areas are covered and chances of bugs creeping up is less.

Whenever the new build comes, the testing team does the following,

 Smoke testing (check basic functionality)

 Test new features

 Test the modified features

 Retesting the bugs

 Regional regression testing (checking the impact areas)

The below graph shows that *increase in testing effort will not lead to catching more bugs*,

Thus, we can see that the initial effort spent on regional regression testing will lead to catching more number of bugs. But with the effort spent on full regression testing will diminish the number of bugs we catch. Thus, we can conclude increase in testing effort will not lead to catching more bugs.

***Full Regression Testing***

After 2 releases of the product, during the 3rd release – customer asks for adding 2new features, deleting 1feature and modifying 1feature. Also some bugs needed to be fixed. The testing team after doing impact analysis find out that making all the above changes will lead to testing the entire product.

###### Thus, Testing the changes and all the remaining features is called Full Regression Testing.

*When do we do Full Regression Testing ?*

 When changes are more

 Whenever the changes are done in the root of the product. **For ex,** JVM is the root of Java application. Whenever any changes are made in JVM, the entire Java application is tested.

Regional Regression Testing is the most preferred method of regression testing. But the problem is, we may miss a lot of bugs doing Regional Regression Testing.

We can solve this problem by the following method – when a product is given for testing, for the 1st ten cycles, we do regional regression testing, then for the 11th cycle, we do FRT. Again, for the next 10 cycles, we do RRT and for the 21st cycle we do FRT. Thus we continue like this, for the last ten cycles of the release – we do **only** FRT. Thus, following the above method – we can catch a lot of bugs.

### Interview Questions

###### What is Regression Testing ?

***Ans) 1st tell – definition of Regression Testing Then continue with,***

###### Based on the changes, we test only the changes OR the changes and impact areas OR the changes and the entire product.

***Thus, we have different types of Regression Testing, namely,***

######  Unit Regression Testing – test only the changes

 ***Regional Regression Testing – test only the changes and impact areas***

######  Full Regression Testing – test all the changes and the entire product

* 1. ***Difference between Re-testing and Regression Testing.***

###### Ans) Re-Testing – developer fixes the bug(or makes some changes) and gives the product for testing. We are testing only the fixed bug(or changed areas) i.e, we are testing only the defect fixes. We are re- validating the defect

***Regression testing – we are testing the fixed (or changed areas) and also testing the other areas to check if they are broken.***

**Disadvantages** of doing regression testing manually again and again,

* + - Monotonous job
    - Efficiency drops down
    - Test execution time is more
    - No consistency in test execution

Thus, we go for Automation to solve this problem. When we have more cycles of Regression testing – we go for Automation.

**Automation :**

Customer gives requirements – we start writing test cases – about 1000test cases are written for the entire product – development team gives the 1st build – we convert about 600 test cases to QTP scripts and the remaining 400 test cases are not converted – we executed the converted QTP test cases using QTP tool. Remaining 400, we test manually – thus 60% of time is saved by automating our testing.

**QTP** stands for **Quick Test Professional**