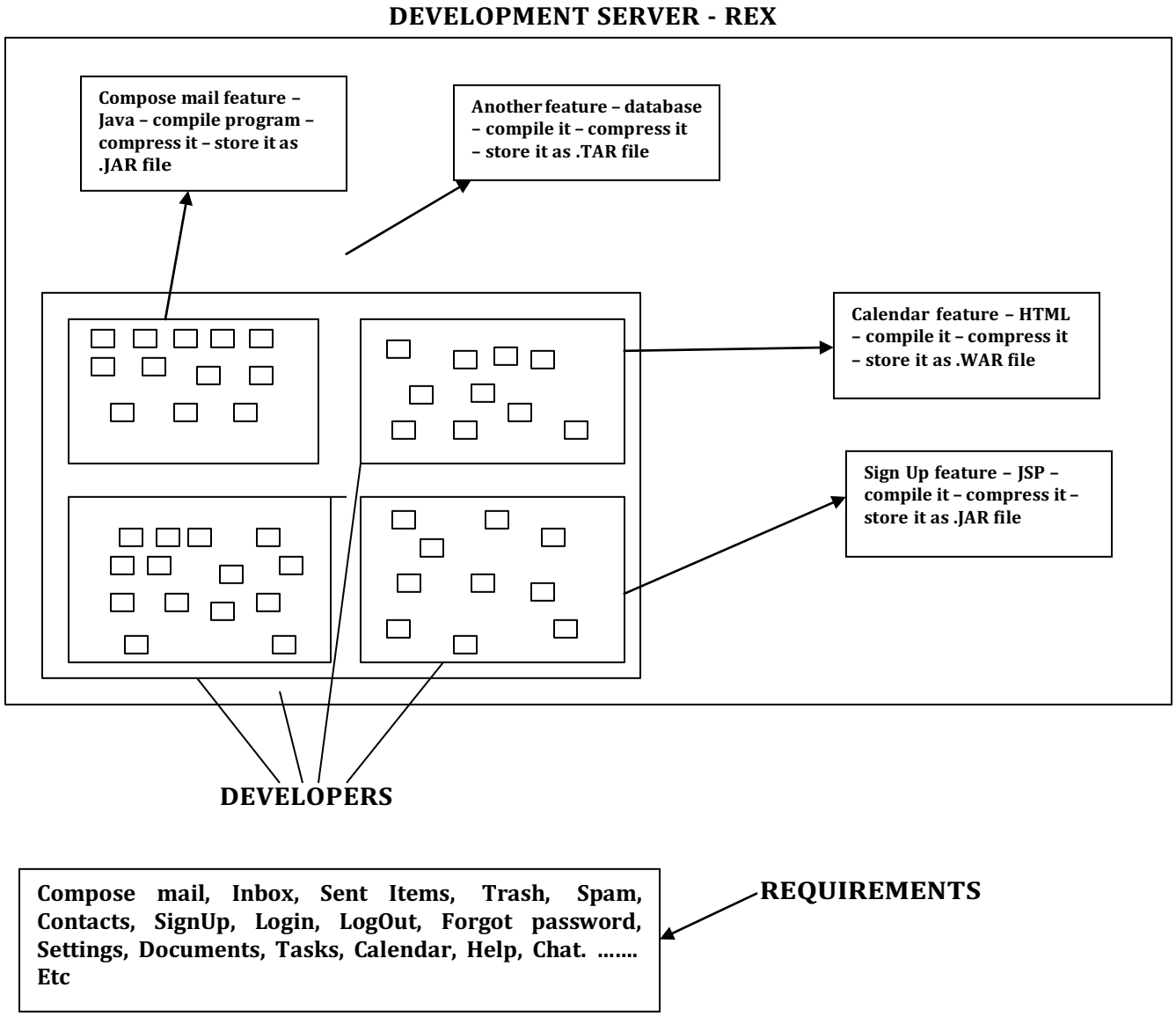
When we get a job, **we will be working on 3 types of projects**,

- **Stand-alone application** – software installed in one computer and used by only one person. **For ex –** Installing s/w of a Calculator, Adobe Photoshop, MS Office, AutoCad
- **Web Application** – any application software accessed through browser is called web application. **For ex –** yahoo.com, gmail.com
- **Client – Server application** – here, we are installing both client and server software to access the application.

Testing WEB APPLICATION



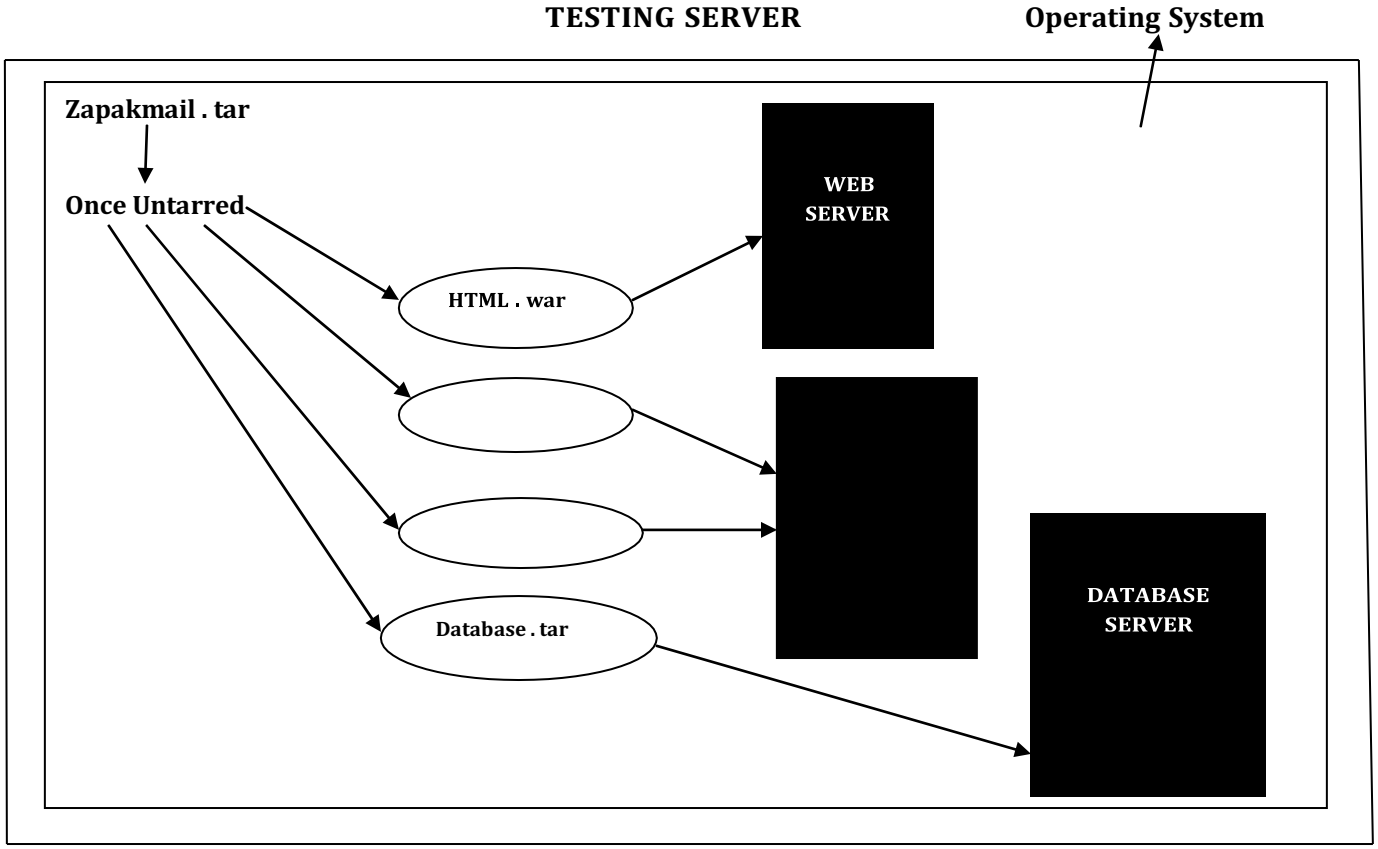
The customer sends the requirements to both development team and testing team. Now, the test lead will understand the requirements and logically assign the work to other team members.

The developers write the programs to build the features. They have to build the various features given by the requirements of the customers. The developer uses *different languages* and compiles and compresses the programs.

The developer compiles the Java program and then compresses it to **.Jar** file.

The developer compiles the JSP program and then compresses it to **.Jar** file.
The developer will compile the database and then compress it to **.Tar** file.
The developers will not compile the HTML directly and compress it to **.War** file.
Now again he will compress all the compressed file into **.tar** and puts it into 1 file and stores it in **D:** drive as, **D : // Builds / B01 zapakmail . tar** and sends a mail to the testing team.

In the test environment 1st he will install the OS, Web Server, Application Server, Database Server. This is done before installing the product software. After installing, he will copy from **D :** drive and paste the **Tar** build. He will then **untar,** then we get **.jar, .tar, .war, .jar** files.
All JSPs and Web HTML programs should be moved into WebServer. All Java Programs should be moved into ApplicationServer and all DataBase programs goes to DataBaseServer. **All this is illustrated in the figure below,**



For Web Application, the WebServer, ApplicationServer, DatabaseServer should be installed. Once the application is ready for testing, the test lead will send a mail to his team with the following **URL,**

http : // QA.zapakmail.com

with all the above required features like – compose mail, inbox, sent items etc.
The test engineer will open the browser and copies the **url** sent to him along with the username and password.

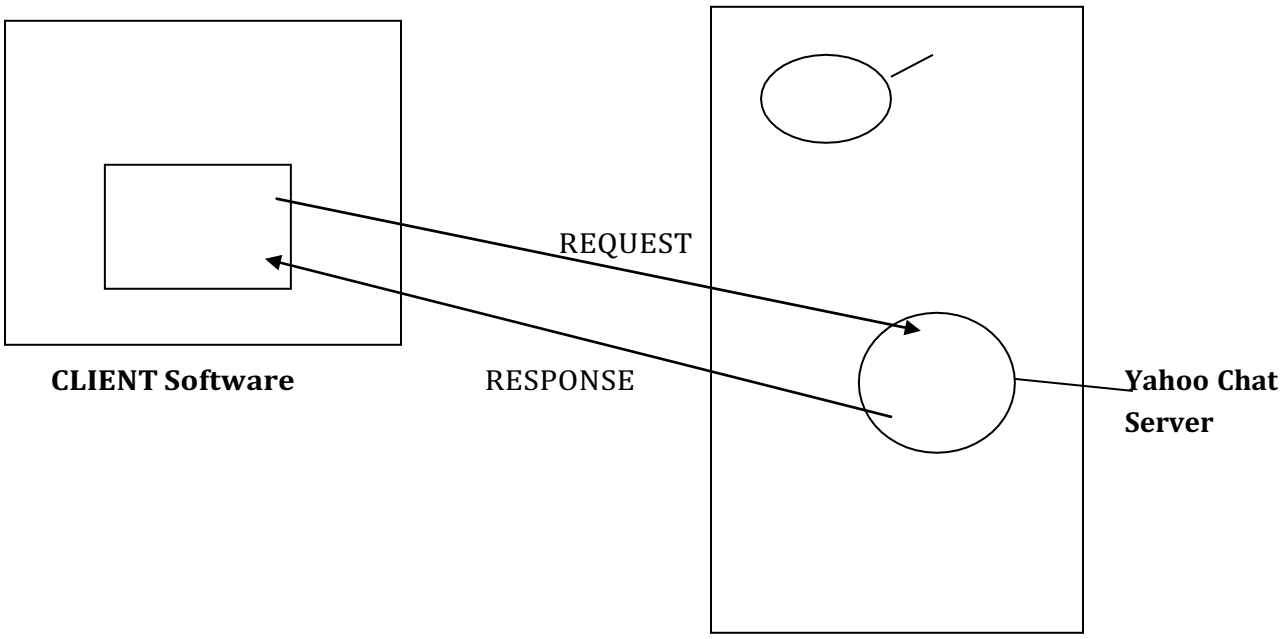
***

Whenever new build comes,

- Test Lead removes all programs in Web, Application and Database Servers
- Once again copy, untar and move new files to their specific server
- Once again start the server

Testing CLIENT – SERVER application :-
Ex – Yahoo messenger, Gtalk, ATM

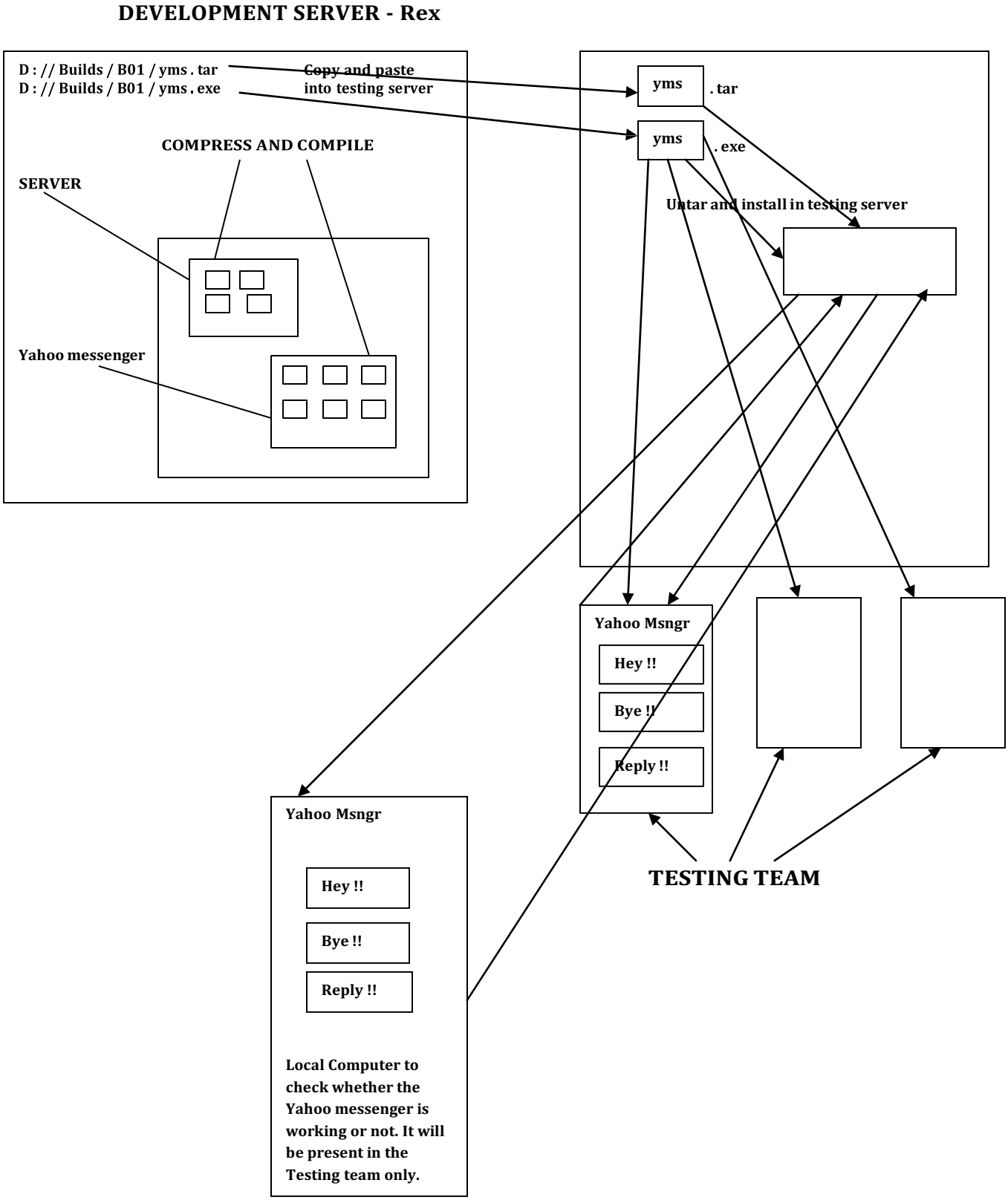
Here, we **consider the example of YahooMessenger :-**
It requires 2 software – client software and server software.



Here, Yahoo messenger s/w is the client s/w and Yahoo server is the server s/w.
The main use of server s/w is,

- Server is used to communicate the information
- To store the information

Here, we 1st download yahoo messenger into the client software, the s/w is installed from the server. Yahoo messenger is installed in the client server. It then displays the username and password page (Login page). Once the client enters the username and password and submits, the request will be sent to the server, it will send a response to display the Homepage. The response is coming from the client server itself. The client displays the next window.
When we enter the information and submit, it goes to the Yahoo Server and stores information and sends a response. The client server displays the next window because of the executable Yahoo Messenger. Exe file on the client‟s PC.
The calendar, contacts etc and all other features of the YahooMessenger will open from the client s/w only because he has already downloaded and installed the s/w before only.
Whenever we double-click on Contacts, the page/window will be displayed once we enter the details and submit, the request goes to the server and stores the information and response is sent back.


DEVELOPMENT SERVER - Rex
D : // Builds / B01 / yms . tar
D : // Builds / B01 / yms . exe
Copy and paste
into testing server
COMPRESS AND COMPILE
SERVER
Yahoo messenger
yms
. tar
yms
. exe
Untar and install in testing server
Yahoo Msngr
Hey !!
Bye !!
Reply !!
Yahoo Msngr
Hey !!
Bye !!
Reply !!
Local Computer to check whether the Yahoo messenger is working or not. It will be present in the Testing team only.
TESTING TEAM

Once the developer writes the program for YahooMessenger and server, he compiles and compresses,

- Server s/w will be named as yms.tar
- Client s/w will be named as ym.exe

This is saved in the folder as,

D : // Builds / B01 / yms.tar

D : // Builds / B01 / ym.exe

This is copied to QA (test environment) from Rex (development environment) and pastes both the build. Yms.tar is untarred and install it in QA server

Then test engineer downloads and installs ym.exe in their computer as well as the local computer. Now start testing all the features.

When a bug is detected, it can wither be in client s/w or server s/w. changes can be done in server s/w or client s/w.

There are cases where we can predict or may not predict where the defect is.

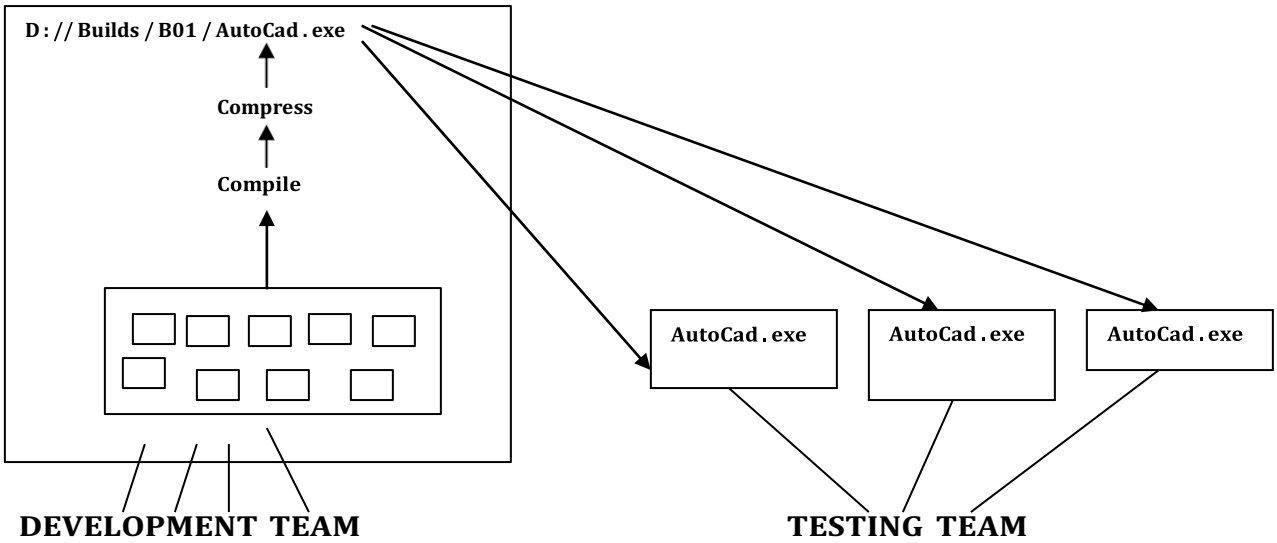
When the Test engineer is sending bugs to the developers, the developers will fix the bugs and add new features in either server s/w or client s/w or both.

When the new s/w build is sent to the Testing team. They will uninstall the server and installs the new server s/w. They will also uninstall the YM in their computer and install new YM because by adding new features, the previous YM may not work properly.

Whatever OS is used by end-users, the s/w has to be tested in all Os(s). Ex – XP, Vista, Windows 7 etc

Testing the s/w in different OS(es) is called **compatibility test.**

Testing STAND – ALONE application :



Examples of stand-alone applications are – Photoshop editor, Nero, MS-word, etc

Here, we are installing only 1 s/w which is client s/w.

The developers write the programs, they compile and compress and save it in,

D : // Builds / B01 / AutoCad.exe

Development lead sends a mail to Test lead saying that the product is ready.

Now, the Testing Team should copy, paste and install the s/w in his own computer or local computer and test the allotted features.
Whenever the developer sends a new Build, the Test engineer will uninstall the old build and install the new build. Before releasing the product, we should also test for compatibility. Always we should work on latest build.

RELIABILITY TESTING

Testing the functionality of an application continuously for a particular period of time.
For ex – let us consider our cellphones / mobiles. The s/w may not work continuously. In mobile, after a week (or) ten days, the phone may hang up because whatever feature is present in the phone, whenever it is used it continuously creates an object in the RAM of the phone. Once the RAM is completely filled with objects, if we get any call or message – we will be unable to press the call button and the phone hangs up. Thus we make use of a **clean-up software.** When we switch off the phone and switch it on again, all the objects in the RAM gets deleted.
For doing Reliability testing, we write an automated program or script and click on Run. We do Reliability testing using ready-made tools and not manually. The test engineer will run the programs using Automated tools.

RECOVERY TESTING

Testing the application to check how well it recovers from crashes or disasters.
The steps involved in Recovery Testing are,

1. Introduce defect and crash the application – Somebody will guide us as to how and when will the software crash. OR. By experience after few months of experience on working the project, we can get to know how and when the s/w can and will crash.
2. Whenever s/w crashes, it should not disappear but should write error log message (or) crash log message where in reason for crashing should be specified. **Ex –** ***C : // Program Files /QTP /crash.log***
3. It should kill its own process before it disappears. **For ex –** In Windows, we have TaskManager to show which process will be running.
4. Re-open the application. The application must be reopened with previous settings.

For ex – when using Mozilla FireFox, if the power goes off. When we switch on the PC and re-open Mozilla FireFox, we get a message asking whether we want to **start a new session** or **restore previous session**.
For any product developed, the developers write a recovery mechanism which explains – why the s/w is crashing, whether crashlog messages are written or not, etc.

ACCEPTANCE TESTING

Acceptance testing is done by end users. Here, they use the s/w for the business for a particular period of time and check whether the s/w can handle all kinds of real-time business scenarios / situations.

For Acceptance testing, let us consider the example shown below.

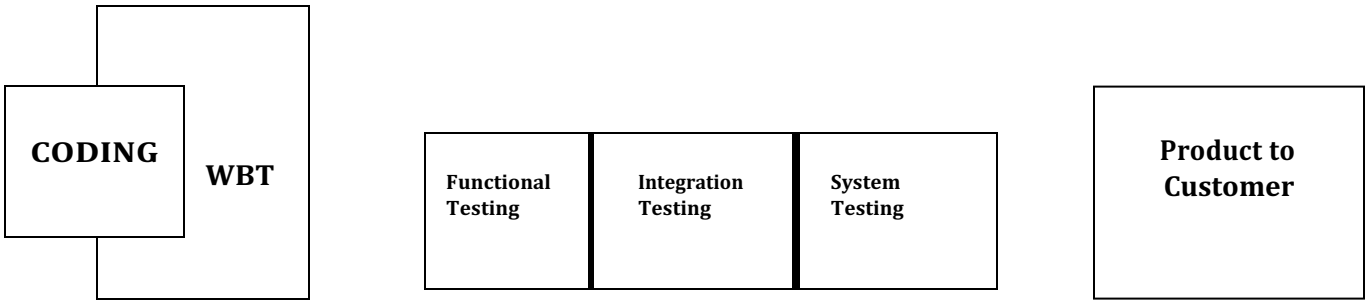
WIPRO **FED - EX**

Fed-ex with its requirements asks Wipro to develop the s/w and Wipro agrees to give the s/w in 2 releases like below,

Jan 2010 —— **25crores** —— **Sept 2010** **Sept 2010** —— **18crores** —— **Feb 2011**

On September 8th, test manager tells the project manager that there is a critical bug in the application which will take another 5days to fix it.
But the project manager says you just deliver the application and by the time they implement in Fed-ex, it takes another 25days so we can fix the bugs or otherwise we will have to pay the penalty for each day after the said release day. **Is this the real scenario ? – No**. Then what happens, we will see now in 3 cases which really and who really does the acceptance testing.

CASE 1 :- here, we will discuss how the acceptance testing is done or how the test engineer testing becomes the acceptance testing here.



Usually, the actual flow of testing will be like above. But, here a small difference we see where the system testing or end-to-end testing becomes the acceptance testing. To understand this, follow the sequence below,
Fed-ex gives the requirements and Wipro develops the s/w and do all testing and gives it to Fed-ex
Are the Fed-ex going to use the s/w as soon as they get from Wipro ? – NO, certainly not. Then what do they do ? – Observe,
Fed-ex, they have some group of Test Engineers and after they get the s/w, this team starts testing it. So, now we can understand that though the test engineer do the testing but it is done at customer level. This end-to-end testing is called ACCEPTANCE TESTING.

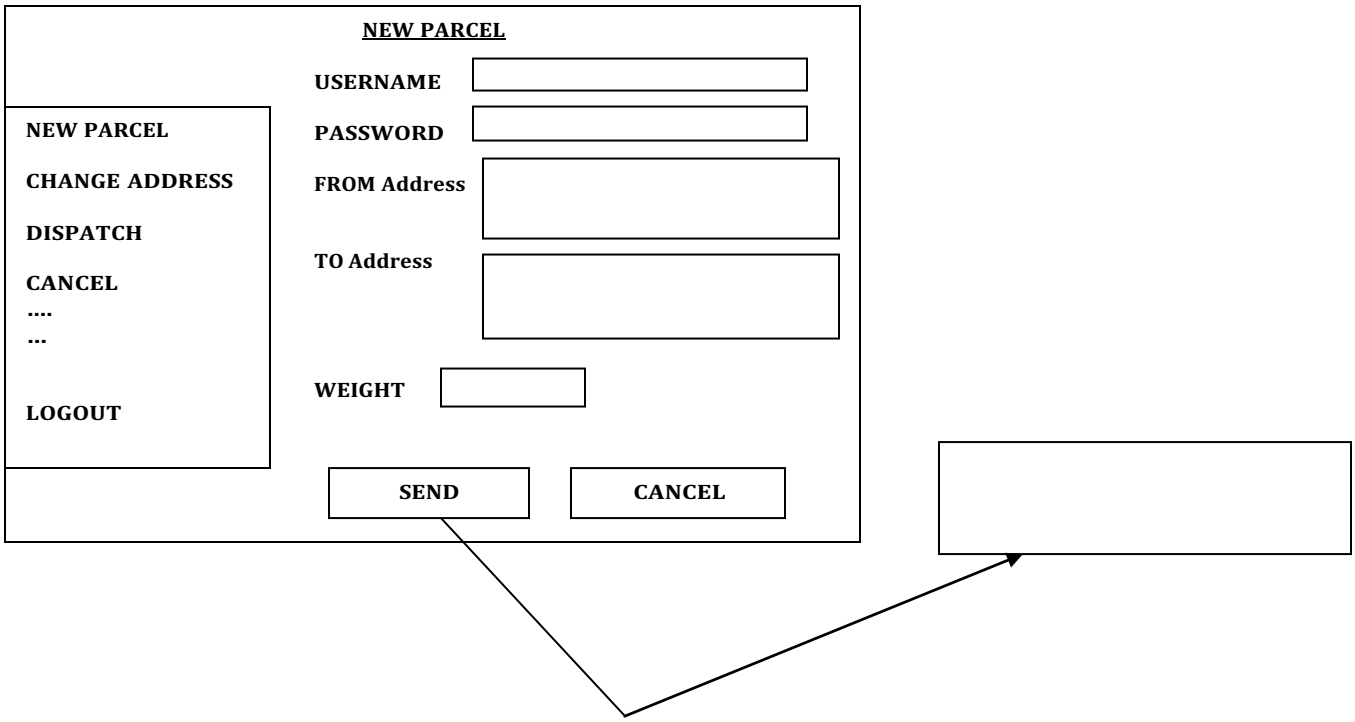
The difference between Wipro test engineers and Fed-ex test engineers are,

- The Wipro testing do Functional Testing, Integration Testing and System testing. But at Fed-ex, the testing team do only end-to-end testing / system testing.

The difference between end-to-end testing of Wipro and Fed-ex is,

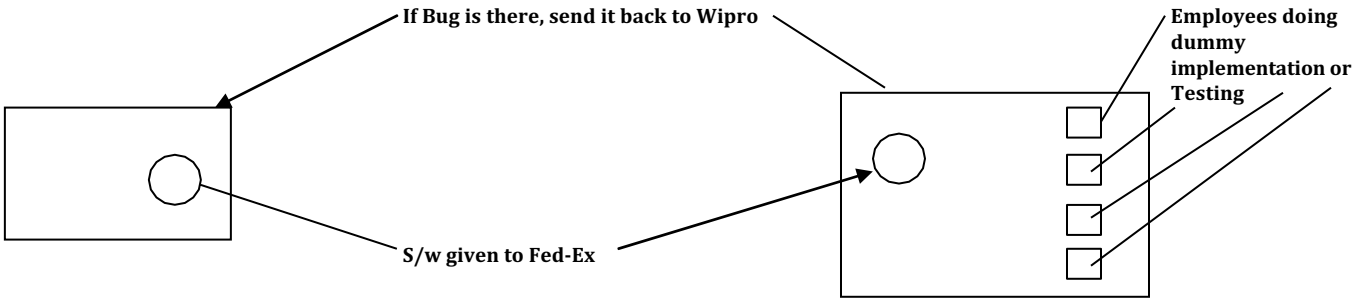
- Fed-ex engineer is a domain expert
- Fed-ex engineer understands the business well
- Fed-ex engineer tests for real time data
- Fed-ex engineer is the one who gave the requirements.

To understand this, we see the example below. If the application format is like below,



In the above example, after the product is given to Fed-Ex Test Engineers, they do testing and they know after the application has been filled above, it should produce an message saying "Parcel 1 Docket ID Produced". If this is not happening, they give back the application for fixing bugs. Now, the Fed-Ex checks whether this feature is there or not in the requirement. If it is there and Wipro have not done fix it, then Penalty Counts for Wipro from that day, whereas the TE at Wipro will not be knowing this and thus arises the difference in testing at Wipro and Fed-Ex.
Thus, the TE become END-USERS here and this testing is known as Acceptance Testing.

CASE 2 :-
In this case, we see how the employees are becoming end-users and do acceptance testing.



The s/w is developed and tested at Wipro‟s place and then sent to Fed-ex. At Fed-Ex, they have less TEs and so it is not possible for them to do Acceptance testing. So, out of 400 employees of Fed-ex, Fed-ex gives the s/w to 40 employees and installs the product at their systems and asks them to start using the s/w and come up with bugs or issues.
Now, the 40 employees, they do dummy implementation (i.e, they implement the data into the application and also have the data written manually). Now, the employee here becomes the end-users and come up with bugs and issues when using the s/w.
These issues are verified against requirements and now penalty is charged for Wipro (sometimes, penalty is charged on an hourly basis).
If the bug found is not as per requirement, then Fed-Ex can go for CR or RFE.
CR – Change Request – i.e, if the requirement has not been specified properly, then Fed-Ex gives the correct requirement and requests for change.
RFE – Request For Enhancement – if Fed-Ex feels that a particular module can be enhanced and developed in a better way, then they can send the CRS as RFE and Wipro goes on to make the necessary changes.

Thus, **Acceptance Testing can also be defined as –** ***end-to-end testing done by engineers sitting in customer's place. Here, they take real time scenarios and check whether the s/w works or not. Here also, we are able to take real time business scenarios because the end-users know how the business flow works.***

We are getting more and more builds for Acceptance Testing means,

- The product quality which is delivered to customers is not good. Development and testing both are not good.
- After receiving the s/w, customer is getting more and more ideas, so he is asking for more and more changes
- The requirement which was given in the beginning is not clear.

CASE 3 :-
Here, the Fed-ex customers become the end users.

Here, the s/w is developed and tested and implemented at Fed-ex production servers and thousands of users start using the s/w. This comprises the **1st release.** When using the s/w, Fed-ex comes up with more number of features and enhancements and sends the CRS to Wipro who make the additional changes and modules and give it to Fed-ex.
Thus, what is happening here is – the requirements are collected by Fed-ex from customers and end-users and then the s/w is developed.

The number of cycles depends on,

- Number of features
- Complexity of features
- How new features affect old features

Hot fix – in production environment, whenever the client finds critical bugs – developers fix the bugs – small team of TEs test it – reinstall the s/w – client starts using the new s/w. This entire process is known as **Hot fix**. It takes few hours to 1day.
For ex, if the login feature itself is not working at the production environment, then the client immediately sends it for fixing which is done asap.

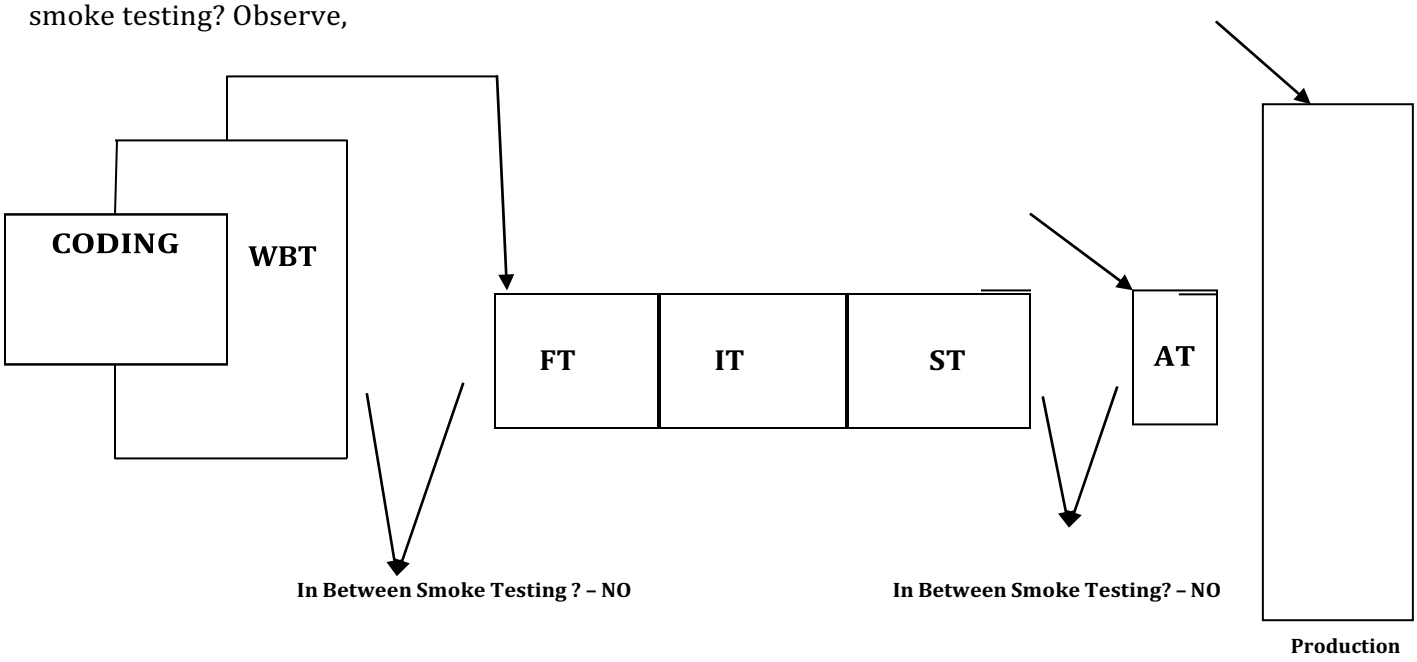
SLA – Service Level Agreement

Interim Release – (short release).
Between 2 major releases – there is a short release of enhancements – this comes up when the client requires a small bunch of features very urgently. Out of 70developers, around 10come out and out of 30 TEs, around 3 come out – they develop and test the s/w – client does 1 short round of Acceptance testing – before adding it to the production environment – this interim could take just around 15days to 1month.

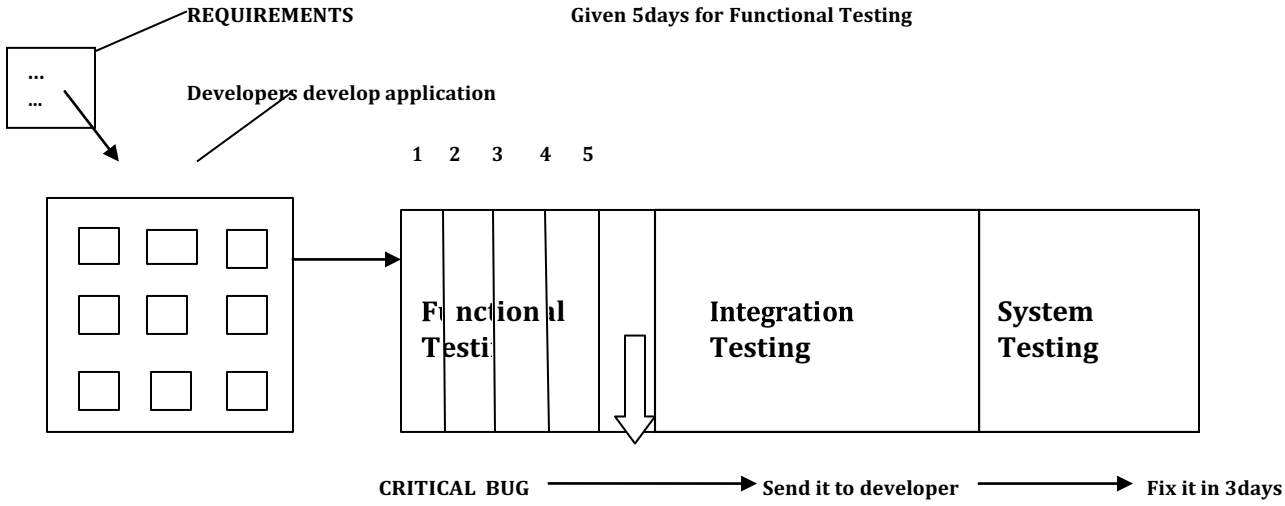
***** SMOKE TESTING or SANITY TESTING or DRY RUN or SKIM TESTING or BUILD VERIFICATION TESTING ***** (*Very very important interview question)*

Testing the basic or critical features of an application before doing thorough testing or rigorous testing is called as smoke testing.
It is also called Build Verification Testing – because we check whether the build is broken or not.
Whenever a new build comes in, we always start with smoke testing, because for every new build – there might be some changes which might have broken a major feature (fixing the bug or adding a new feature could have affected a major portion of the original software).
In smoke testing, we do only positive testing – i.e, we enter only valid data and not invalid data.

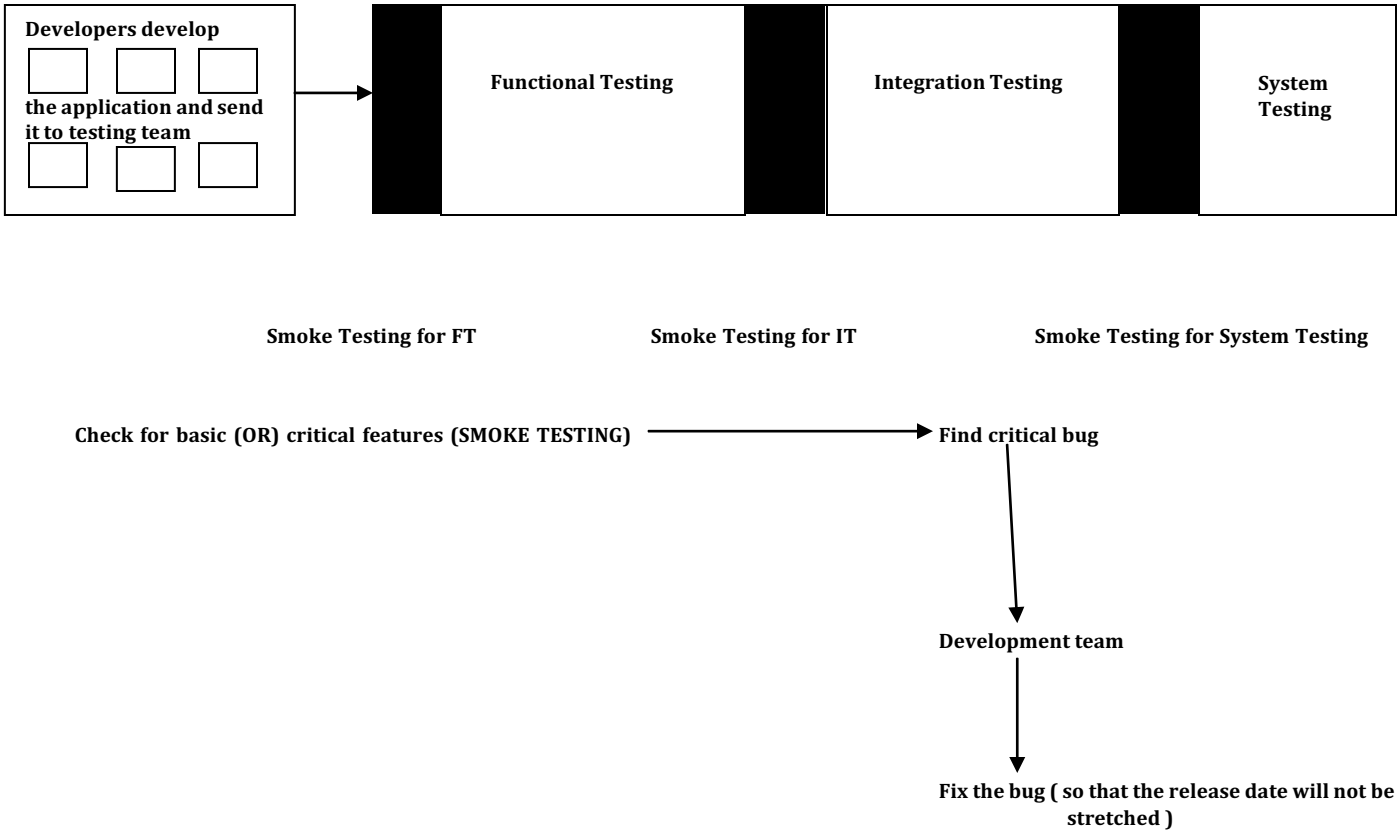
Do we have separate testing (or) do we have to do it in between FT, IT, ST ? Then, where actually do we do smoke testing? Observe,



From the above diagram, it may be confusing when we actually do smoke testing
Now, we have to understand that smoke testing is done in all testing before proceeding deep into the testing we do.
The below example will make us understand better when to do smoke testing,
Developers develop application and gives it for testing. The testing team will start with FT. suppose we assume that 5days we are given for FT. on the 1st day, we check one module and later 2nd day we go for another module. On the 5th day, we find a critical bug, when it is given to the developer – he says it will take another 3days to fix it. Then we have to stretch the release date to extra 3days.
Then how do we overcome this ? – Observe how smoke testing works here. In the above scenario, instead of testing module by module deeply and come up with critical bug at the end, it is better to do smoke testing before we go for deep testing i.e, in each module – we have to test for basic (or) critical feature and

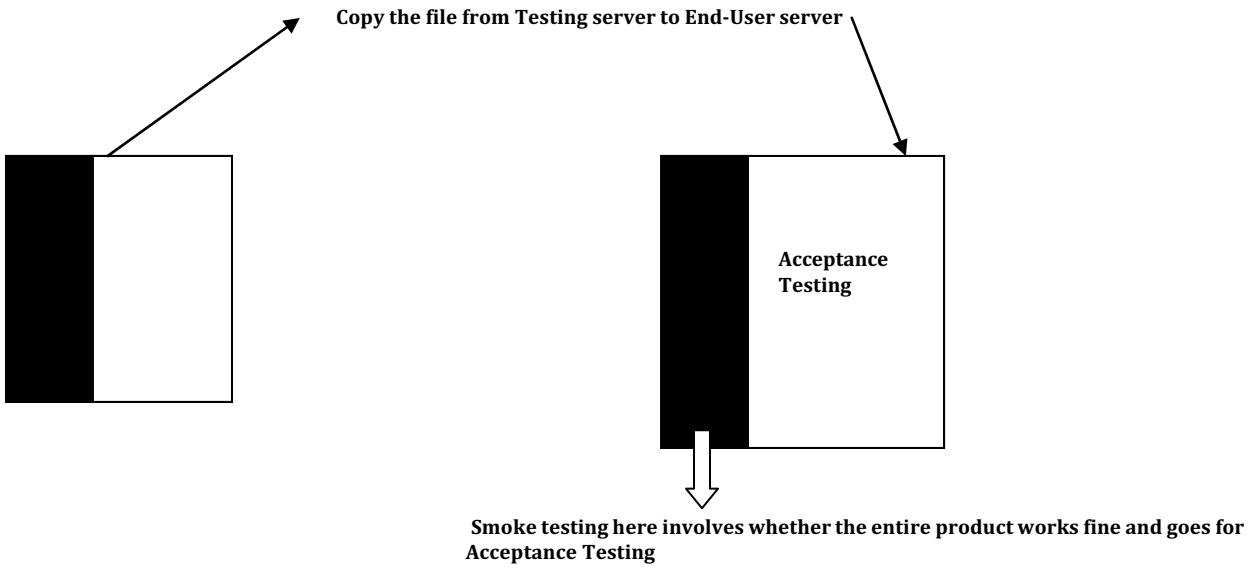


then proceed for deep testing. The scenario will be like this as shown in the figure below,



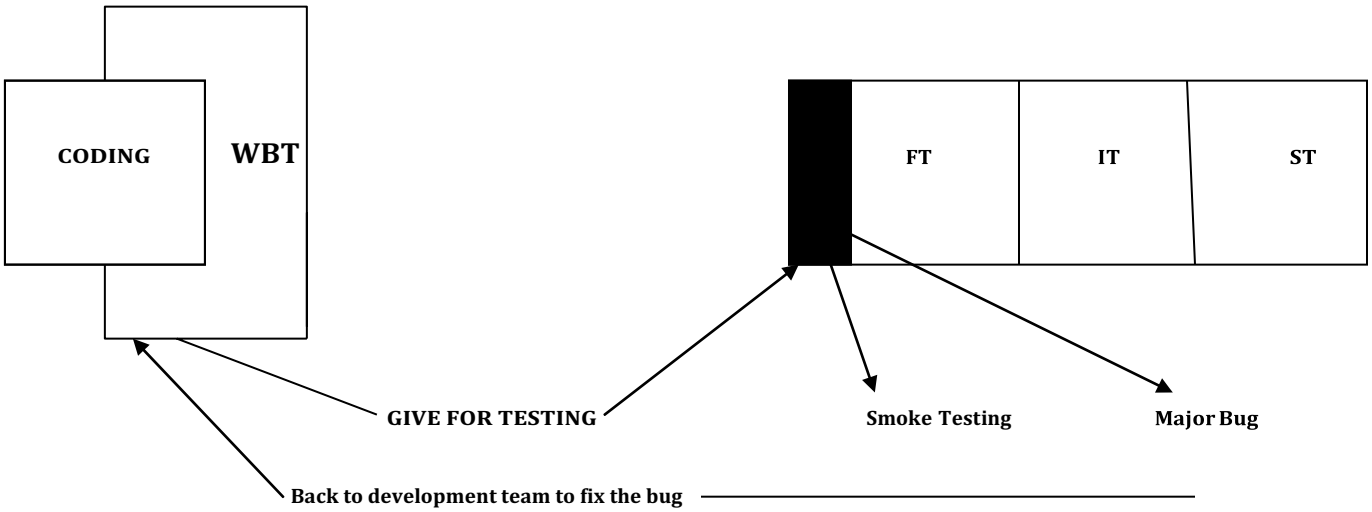
Question arises – how do we know which is the critical feature? – we will come to know which is the critical feature or basic feature when we proceed with the testing.

Smoke Testing in Acceptance Testing

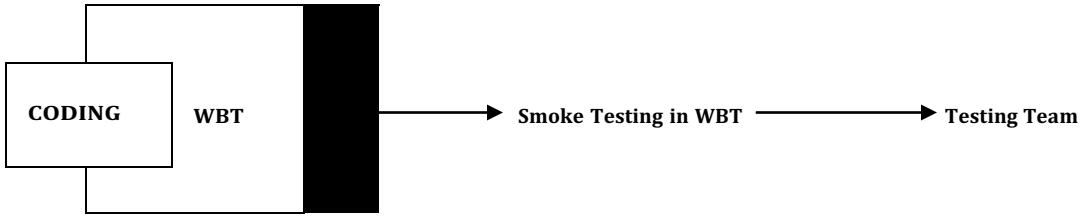


Let us now consider **3cases** where in which we see where we do smoke testing in different kinds of testing.

CASE 1

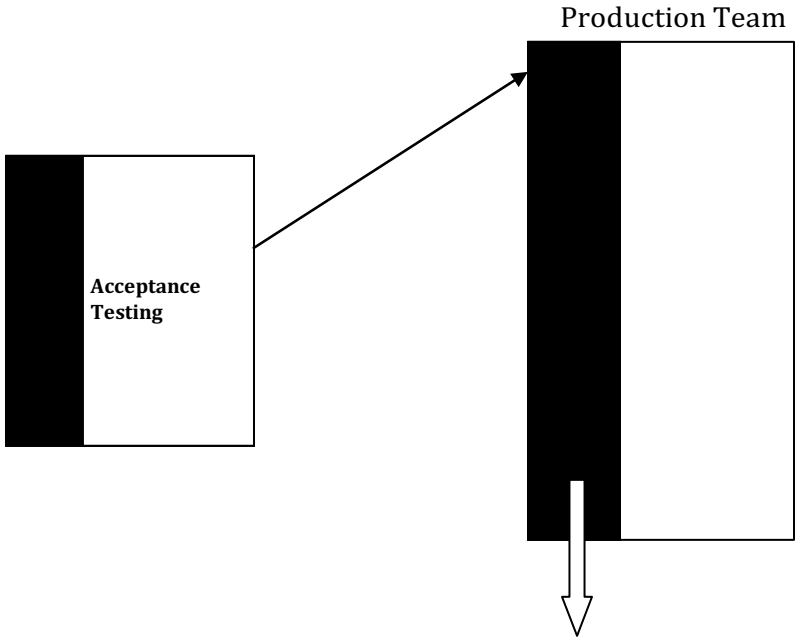


Major bug in the initial stages is an insult to the developer.
Sometimes, the below procedure is also followed,



If this is done, then the testing team need not do smoke testing as the bugs are already fixed in WBT. But depending upon the project or the organization, normally this is not followed.

CASE 2



SMOKE TESTING – Whether Installation is working fine or not

In the above scenario, smoke testing in production team means after we do Acceptance Testing, Smoke Testing involves whether the s/w developed is installed fine or not.

Important Points to Remember

- When we are doing smoke testing, we do only positive testing (only valid data is entered)
- Here, we test only basic or critical features
- Here, we take basic features and test for important scenarios
- Whenever the build comes to the customer, before the customer / client does Acceptance Testing, he also does Smoke Testing before doing Acceptance Testing
- When the product is installed in production, we do quick smoke testing to ensure product is installed properly.

Why we do Smoke testing ?

- Just to ensure that product is testable
- Do smoke testing in the beginning – catch bugs in basic features – send it to development team so that development team will have sufficient time to fix it.
- Just to ensure that product is installed properly.

In early stages of product development, doing smoke testing fetches more number of bugs. But, in later stages of product development, if you do smoke testing – the number of bugs that you are going to catch in smoke testing will be very less. Thus, gradually the effort spent on smoke testing is less.

Note :- *only for information purpose (NOT FOR STUDYING)*

Smoke testing is classified into two types,

- ***Formal Smoke Testing*** – the development team sends the s/w to the test lead. The test lead then instructs the testing team to do smoke testing and send report after smoke testing. Once, the testing team is done with smoke testing, they send the smoke testing report to the Test lead.
- ***Informal Smoke Testing*** – here, the test lead says the product is ready and to start testing. He does not specify to do smoke testing. But, still the testing team start testing the product by doing smoke testing.

INTERVIEW QUESTIONS

1) Difference between Smoke Testing and Sanity Testing and Dry Run

Ans) Sanity Testing

- ***Narrow and deep testing. scripted***
- ***Take some very very important features and do deep testing***
- ***It is manually done***

Smoke Testing

- ***UnScripted. Shallow and wide testing***
- ***Take all important features and do high-level testing***
- ***Build comes – write automation scripts and run the script. Thus test done automatically.***

Dry Run - A dry run is a testing process where the effects of a possible failure are intentionally mitigated. For example, an aerospace company may conduct a "dry run" of a takeoff using a new aircraft on a runway before the first test flight.