Manual_Assignment_4
Defect Management &
Automation Core Testing (Load Runner)

Submitted by Kuldeep Parmar

1. Mention what are the categories of defects?

Types of Defects :

Defects can be categorized into different types basing on the core issues they address as described below

Functionality Defects:

Defects directly related to functionalities i.e. features not working properly
e.g., Calculator has no '=' button for the calculation

Performance Defects:

Software doesn't meet the expected performance requirements as it should be
e.g., Website's loading time to open

User Interface Defects:

Difficult to operate for the users i.e., Not a user friendliness
e.g., Login page has no cancel button, Alignment problem etc

Compatibility Defects:

Software does not work correctly on different hardware and software configurations

e.g., Application not running on Android or Windows platform
Application interface is appeared differently in different browsers

Security Defects:

Software doesn't protect the user's data from malicious attack
e.g., Password entered in visible form

Authentication: Accepting an invalid username/password
Authorization: Accessibility to pages though permission not given

Documentation Defects:

Document is incorrect or inaccurate to use the features of the app
e.g., Test cases having wrong entries

Data Quality/Database Defects:

Deals with improper handling of data in the database
e.g., Values not deleted/inserted into the database properly Improper/wrong/null values inserted in place of the actual values

2. Bug Categories

A defect accepted by development team is bug
So, bug categories can be shown as

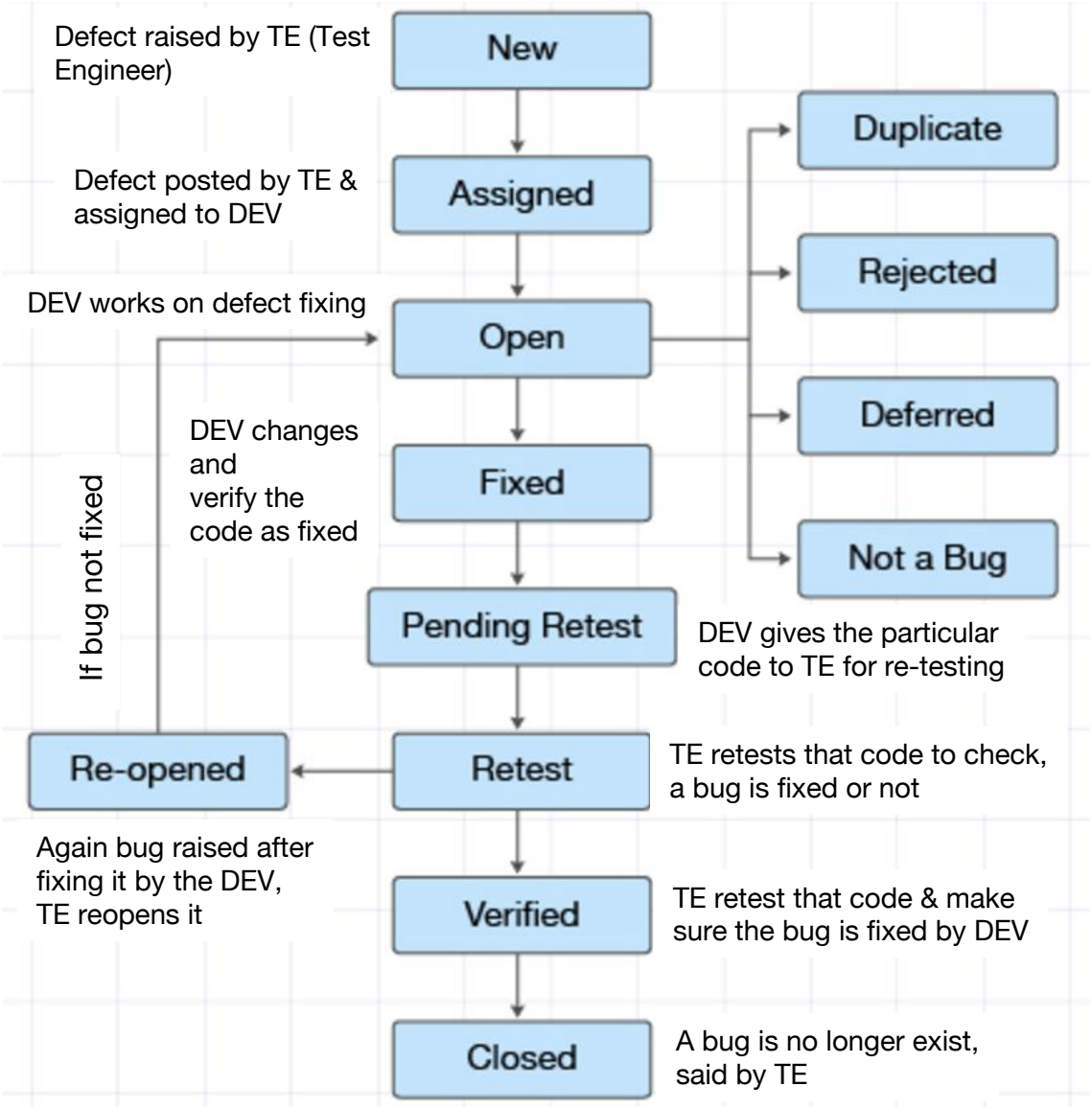
Functional Bug, Performance Bug, User Interface Bug, Compatibility Bug, Security Bug, Documentation Bug, Data Quality/Database Bug etc

3. Difference between Priority and Severity

Priority	Severity
It is the sequence in which developers need to fix bugs or defects (Show the level of importance)	The impact of Defect /bug on the customer business workflow is known as Severity (Denotes the degree of impact)
Business values and release time drives bug priority	The performance of software drives bug severity
Driven by business value and time frame	Driven by software performance/functionality
Categorized as: High (Fix it immediately) Medium (Fix it in next to next build) Low (Fix it in comming build)	Categorized as: Critical, High, Medium, Low, Cosmetic
Value is subjective	The criticality of the bug does not change with time
It indicates how early any bug will be fixed (Priority defines the order of bug fix)	It indicates the seriousness and impact of the bug, and hence, the fixing queue is determined (Determine how server it will be at the user end)
Bug priority is finalized by the manager in consultation with the client	Bug severity is determined by the QA engineer
It is based on customer requirements	As mentioned, it is based on the software's functionality and technical aspects

4. What is Bug Life Cycle?

The defect accepted by a developer is called a bug
The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as ‘Defect Life Cycle’ or ‘Bug Life Cycle’
When a bug is discovered, it goes through several states and eventually reaches one of the terminal states, where it becomes inactive and closed
The process by which the defect moves through the life cycle is depicted as below



New:
When a new defect is logged and posted for the first time. It is assigned a status as “New”

Assigned:
Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team

Open:
The developer starts analyzing and works on the defect fix

1) Duplicate:
If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “Duplicate”

2) Rejected:
If the developer feels the defect is not a genuine defect then it changes the defect to “Rejected”

3) Deferred:
If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs

4) Not a bug:
If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”

Fixed:
When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed”

Pending retest:
Once the defect is fixed the developer gives a particular code for retesting the code to the tester

Since the software testing remains pending from the testers end, the status assigned is “Pending Retest”

Retest:
Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test”

Verified:
The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “Verified”

Re-opened:
If the bug persists even after the developer has fixed the bug, the tester changes the status to “Re-opened”
Once again the bug goes through the life cycle

Closed:
If the bug is no longer exists then tester assigns the status “Closed”

5. Which components have you used in Load Runner?

LoadRunner is a performance testing tool that includes following components:

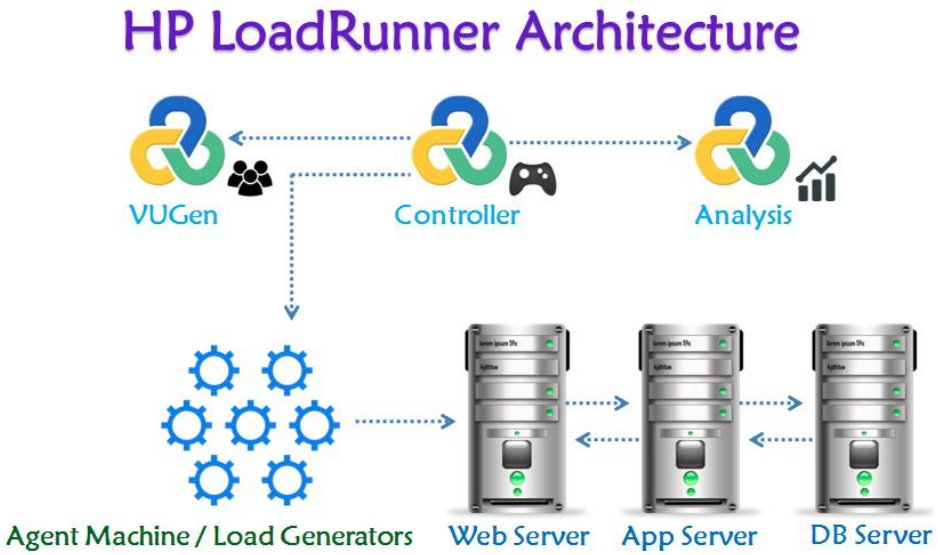
1. Virtual User / Vuser scripts: It will allow you to record and replay script. And allow you to enhance your script

2. Controller: It will allow you to design the scenarios and execute the test with multiple users

- Load Generators: It generates the load against the application
- Agent Process: It will establish the connection between Load Generator & Controller

3. Analysis: It will allow to analyze the statistics and results and finding out the bottlenecks

Such components are used to simulate user activity, measure performance, and analyze results



6. How can you set the number of Vusers in Load Runner?

In Load Runner, we can set the number of Vusers in the following ways:

- **Controller:** In the Controller, go to "Scenario" > "Vusers" and set the "Number of Vusers" field

- **Runtime Settings:** In the Vuser script, go to "Runtime Settings" > "General" and set the "Number of Vusers" field

- **Command Line:** Use the command line option "-n" followed by the number of Vusers, for example: "lr -n 10"

Note: The number of Vusers can also be controlled through the Load Runner API and through external automation tools

7. What is Correlation?

Correlation is the process of capturing and replacing dynamic values in a script with unique values for each virtual user (Vuser). This ensures that each Vuser interacts with the application independently, simulating real-user behavior

- Handle session IDs, tokens, and other unique identifiers
- Simulate unique user inputs, like usernames and passwords
- Capture and reuse dynamic data, such as timestamps or order numbers

By correlating dynamic values, you ensure that your load test accurately reflects real-world usage and avoids errors caused by duplicate or hardcoded values

8. What is the process for developing a Vuser Script?

The process for developing a Vuser script in LoadRunner is:

- **Record:** Capture user interactions with the application using LoadRunner's recording tool
- **Analyze:** Review the recorded script, identify dynamic values, and determine what needs to be correlated

- **Parameterize:** Replace dynamic values with parameters to enable unique data for each Vuser
- **Correlate:** Capture and replace dynamic values with correlated data
- **Script Enhancement:** Add think time, loops, conditional statements, and other logic to simulate realistic user behavior
- **Verification:** Validate the script's functionality and accuracy
- **Debug:** Test and debug the script to ensure it runs smoothly
- **Finalize:** Prepare the script for load testing by setting runtime settings and configuring logging

This process helps to create a robust Vuser script that accurately simulates real-user interactions with the application

9. How Load Runner interacts with the application?

Load Runner uses software agents called "protocols" to communicate with the application. These protocols mimic real-user interactions by sending requests to the application's servers and receiving responses. The protocols used depend on the application's architecture and technology stack
Here's how it works:

1. **application Selection:** LoadRunner selects the appropriate protocol (e.g., HTTP, Web Services, Citrix) based on the application's technology

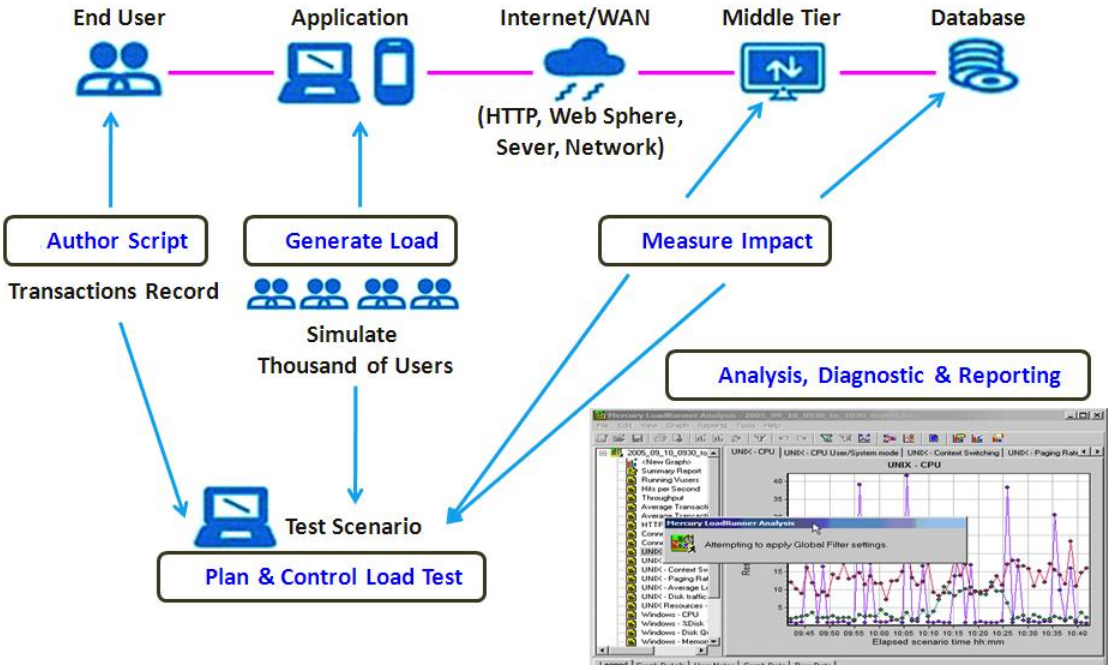
2. **Script Recording:** LoadRunner records user interactions with the application using the selected protocol, capturing requests and responses

3. **Script Replay:** During load testing, LoadRunner replays the recorded script, sending requests to the application's servers using the same protocol

4. **Request/Response:** The application processes the requests and sends responses back to LoadRunner, which analyzes the responses to verify functionality and performance

5. **Data Correlation:** LoadRunner correlates dynamic data, such as session IDs or tokens, to ensure each virtual user interacts with the application independently

6. **Generate Load:** LoadRunner generates a large volume of virtual users, each executing the script and interacting with the application, simulating real-world usage



10. How many VUsers are required for load testing?

The main purpose of VUsers is to simulate the live environment. Universal formula to calculate the arriving rate to the system is Little's Law

$N = Z * (R + T)$

where N – number of VUsers
Z – Transactions per Second (TPS)
R – Response Time in seconds
T – Think Time in seconds

If you get the following data from the stakeholders i.e. TPS, Response Time and Think Time, number of VUsers can be calculated

e.g., TPS is 100, R is 3 sec and T is 2 sec then N will be

$N = 100 * (3+2)$
$= 100 * 5$
$= 500$

Peak load will be 500 VUsers

The number of VUsers (Virtual Users) required for load testing depends on several factors, including:

1. Expected user load: Estimate the number of real users who will be using the application simultaneously

2. Application complexity: More complex applications may require more VUsers to simulate realistic usage

3. Test goals: Identify what you want to achieve with load testing (e.g., peak performance, stress testing)

4. Hardware resources: Ensure the load testing environment can handle the desired number of VUsers. Here are some general guidelines:

- **Low-load testing:** 10-50 VUsers (e.g., small applications, development testing)
- **Medium-load testing (Stress):** 50-200 VUsers (e.g., medium-sized applications, performance testing)
- **High-load testing (Scalability):** 200-1,000 VUsers (e.g., large applications, stress testing)
- **Extreme-load testing (flood):** 1,000+ VUsers (e.g., very large applications, extreme stress testing)

Here, the key is to simulate realistic user behavior and gradually increase the load to measure the application's performance and identify bottlenecks

11. What is the relationship between Response Time and Throughput?

- **Response Time:** The time taken by an application to respond to a user request or action, i.e. how long a user must wait for a response

- **Throughput:** The number of user requests or actions an application can handle within a given time period

e.g., transactions (or any action) per second

The relationship between Response Time and Throughput:

- Inverse correlation: As Throughput increases (more requests handled), Response Time typically decreases (faster responses)
- Trade-off: Improving one metric can impact the other For example, optimizing for faster Response Times might reduce Throughput, and vice versa

- Balance: Aim for a balance between Response Time and Throughput to ensure a good user experience and efficient system performance
- In load testing, analyzing both Response Time and Throughput helps identify performance bottlenecks and optimize application performance

12. Advantages of Bugzila

Bugzilla is a popular choice for bug tracking and management in software development teams

Advantages of bugzilla are given below,

- **Open-source:** Free to use and modify
- **Customizable:** Tailor it to your team's needs
- **Scalable:** Handles large numbers of bugs and users
- **User-friendly:** Intuitive interface for easy bug tracking
- **Searchable:** Powerful search functionality for quick bug lookup
- **Reporting:** Generate detailed reports on bug trends and metrics
- **Collaboration:** Assign, track, and discuss bugs with team members
- **Version control:** Track bugs across different product versions
- **Security:** Granular access control and security features
- **Community support:** Active community and extensive documentation
- **Integration:** Integrates with various tools and platforms
- **Stability:** Reliable and stable bug tracking system