**[Types of Load Testing](http://pureqastuff.blogspot.com/2010/01/types-of-load-testing.html)**

Q: What are Load, Stress, Volume, etc. testing?  
  
A: Volume testing is a way to test functionality. Stress testing is a way to test reliability.  
 Load testing is a way to test performance.  
  
Testing an application under heavy but expected loads is known as load testing.

It generally refers to the practice of modeling the expected usage of a software program by simulating multiple users accessing the system's services concurrently. As such, load testing is most relevant for a multi-user system, often one built using a client/server model, such as a web server tested under a range of loads to determine at what point the system's response time degrades or fails. Although you could perform a load test on a word processor by or graphics editor forcing it read in an extremely large document; on a financial package by forcing to generate a report based on several years' worth of data, etc.  
  
When the load placed on the system is accelerated beyond normal usage patterns, in order to test the system's response at unusually high or peak loads, it is known as stress testing. Stress testing is often incorrectly used interchangeably with load and performance testing.  
In a stress test, the load is usually so great that error conditions are the expected result, although there is a grey area between the two domains and no clear boundary exists where you could say that an activity ceases to be a load test and becomes a stress test.  
There is little agreement on what the specific goals of load testing are. The term is often used synonymously with performance testing, reliability testing, and volume testing.  
Performance testing is usually performed to determine how fast some aspect of a system performs under a particular workload. It can serve different purposes: to demonstrate that the system meets performance criteria; to compare two systems to find which performs better; to measure what parts of the system or workload cause the system to perform badly.  
In the diagnostic case, we use tools such as profilers to measure what parts of a device or software contribute most to the poor performance. In performance testing, it is often crucial (and often difficult to arrange) for the test conditions to be similar to the expected actual use.  
A reliability test determines how likely a piece of hardware (or sometimes software) is to fail. It is part of reliability theory, used originally as a tool to help nineteenth century insurance companies compute profitable rates to charge their customers. Statistical models appropriate for any of these are generically called "time-to-event" models. Death or failure is called an "event", and the goal is to project or forecast the rate of events for a given population or probability of an event for an individual.  
Volume testing is used to determine if the system under test can handle the required amounts of data, user requests, etc.  
Soak testing occurs when running a system at normal to high levels of load for prolonged periods of time. A soak test would normally execute several times more transactions in an entire day than would be expected in a busy day. This should identify any performance problems that appear after a large number of transactions have been executed.

[**Requirements Traceability**](http://pureqastuff.blogspot.com/2010/01/requirements-traceability.html)

Requirements traceability is an active area of research in system and software Engineering  
A well accepted definition of requirements traceability was given in 1994 by Gotel and Finkelstein: "Requirements Traceability refers to the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origin, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)”  
  
Now a days IT people are finding ways to reduce overall life cycle costs ,to deliver reliable and quality products. Every where we are finding fewer staff resources and higher quality expectations about the product/project in short term guidelines. In this kind of situation one needs some solution to keep in track the requirements, any change in requirements, and impact of changes in requirements on another module , and last and the most important one that whether our implemented system’s characteristics are matching with the requirements or not.  
Today, applying Requirements Traceability offers a high level of project control and assured quality that is difficult to achieve by any other means. Through the advent of Requirements Management tools, Requirements Traceability has matured to support and enhance Project  
  
Management, Impact Analysis and Change Management, Defect Management, process improvement and team communications.  
  
Tracing requirements involves documenting the links between the various requirements (i.e., Business requirements, User requirements, functional and Test requirements) and Work Products (i.e., Requirements Documents, Design specifications, Software code, Test plans, Test cases and other artifacts of the development process.)  
  
A Requirements Traceability matrix can simplify the process of tracing requirements. It serves as a graphical representation of traceable relationships between requirements and work products.  
  
With a traceability matrix, IT teams can easily track customer requirements through the software development cycle, diminishing the risk of missing stated or derived requirements, especially when developing large, complex systems.  
  
Along verifying system functionality, Requirements Traceability also tackles some other problems.  
  
1. Requirements Traceability minimizes Scopre creep. Scope creep occurs when any feature or functionality in a final product does not link directly with a customer requirement. When it happens, it costs software organizations time. Without knowing the customer requirement, developers are left to analyze or define functionality on their own or in an uncontrolled manner. This often leads to defects and can be a source of poor system performance or usability in the final product. Requirements traceability detects where functionality or features are missing requirements earlier in the process.  
2. Tracing requirements from development to testing checks that each functional  
requirement used in development matches those established for test cases. This lets IT  
teams double-check that all required system features are tested.  
3. A traceability matrix identifies all the requirements and components of the process—such as design specifications, code and test cases—that need modification to fulfill the change request. This helps ensure that all affected work products are modified to support the change request.  
  
Using a Requirements Management tool can offer additional information about customer requirements. IT teams across the development cycle can track, integrate and share key project information, or requirements attributes—such as priority, status, originator, cost, owner, release, assigned to, source, sponsor and notes.  
The Requirements Management tool will also provide so many solutions such as requirements priorities (which module should be done first according to its priority), functionality testing priorities (which critical functionality to be tested first), notification of change in requirements to assigned people, promotes team communications, reusability of test cases, plans and scripts (if necessary and suitable) within the project or in parallel and subsequent project.  
Many organizations recognize the importance of requirements traceability in implementing a quality software development program.  
  
 Yet, a requirements management tool today allows organizations to do more. Using Requirements management tool project development and test managers can check that all customer requirements are implemented and tested, ensuring quality, reliable system capabilities and characteristics.