**Simple Wikipedia definition of Bug is:** “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”

**What is the difference between a bug, a defect, and an error?**

According to the British norm BS 7925-1: bug--generic term for fault, failure, error, human action that produces an incorrect result.  
Robert Vander wall offers these formal definitions from IEEE 610.1. The  
sub-points are his own.  
  
**Mistake (an error)**: A human action that produces an incorrect result.  
- Mistake made in translation or interpretation.  
- Lots of taxonomies exist to describe errors. **Fault**: An incorrect step, process or data definition.  
- Manifestation of the error in implementation  
**Failure**: An incorrect result.  
**Bug:** An informal word describing any of the above. (Not IEEE)

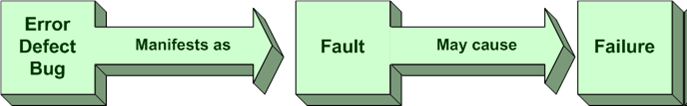
**Error:** Defined as the occurrence of an incorrect result produced by a system/software.

**Bug:** Software/Hardware error that caused the system to malfunction.

**Defect:** Non-confirmation to product specifications. Error and bug makes the defect to occur in system/software.

**Causes of software defects**

A human being can make an error (mistake), which produces a defect (fault, bug) in the code, in software or a system, or in a document. If a defect in code is executed, the system will fail to do what it should do (or do something it shouldn’t), causing a failure. Defects in software, systems or documents may result in failures, but not all defects do so.



**How much testing is enough?**

On what basis is the decision made to stop?

Stopping testing too early risks leaving severe errors in the system;

Continuing testing too long can delay live release resulting in:

• Loss of revenue

• Loss of business (which may or may not result in loss of revenue)

• Damage to corporate reputation (which may or may not result in both of the above)

**Testing and Risk:**

**So, when should testing stop?**

The answer is depends on the risk to the business.

The amount of testing performed depends on the risks involved. Risk must be used as the basis for allocating the test time i.e. available and for selecting what to test and where to place emphasis.

In fact, testing can continue after release of the program to the live environment – it doesn’t necessarily stop on final release.

Testing usually stops when management has enough information to make a reasoned decision about the benefits and risk of releasing the product and decides to either release the product or cancel the product.

**Software Testing Types:**

**Black box testing** - Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

**White box testing** - This testing is based on knowledge of the internal logic of an application’s code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

**Unit testing** - Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses.

**Incremental integration testing** - Bottom up approach for testing i.e. continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately done by programmers or by testers.

**Integration testing** - Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

**Functional testing** - This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.

**System testing** - Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

**End-to-end testing** - Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

**Sanity testing** - Testing to determine if a new software version is performing well enough to accept it for a major testing effort. If application is crashing for initial use then system is not stable enough for further testing and build or application is assigned to fix.

**Regression testing** - Testing the application as a whole for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.

**Acceptance testing** -Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer do this testing to determine whether to accept application.

**Load testing** - It’s a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system’s response time degrades or fails.

**Stress testing** - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

**Performance testing** - Term often used interchangeably with ’stress’ and ‘load’ testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

**Usability testing** - User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

**Install/uninstall testing** - Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment.

**Recovery testing** - Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

**Security testing** - Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

**Compatibility testing** - Testing how well software performs in a particular hardware/software/operating system/network environment and different combination s of above.

**Comparison testing** - Comparison of product strengths and weaknesses with previous versions or other similar products.

**Alpha testing** - In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

**Beta testing** - Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.