# Software Testing

Software Testing is the process of determining that a product or application does what it is supposed to do, in terms of functionality, usability and performance.

Software testing is best used throughout the development of applications so problems can be found early and fixed with a minimum impact to development.

Functional

* **Unit Testing**: This initial testing phase checks each individual component or unit of the system to ensure it operates correctly, allowing developers to fix bugs early in the development process. These components are often the smallest testable parts of an application, including functions, methods, or classes. Typically done during software development Unit Testing can done manually or automatically. Takes a Black-Box approach where the specific implementation is not considered, just the results of the operation.
* **Integration Testing**: This type of testing validates the interactions between different software units, ensuring they work together seamlessly to achieve the expected functionality. Takes place after unit testing and highlights issues in the communication between different units, data formats or API’s.
* **Regression Testing**: Whenever modifications are made to the software, regression testing verifies that these changes do not introduce new defects or disrupt existing functionality. Performed after introducing code changes (ie bug fixes or new features) regression testing confirms if the new code will introduce changes into the existing codebase. For example, did a bugfix in the transaction processing break the daily report?
* **User Acceptance Testing (UAT)**: Conducted by end users, UAT assesses whether the system meets their expectations and requirements, typically performed at the end of the software development lifecycle. UAT should be done as software is developed – providing your client with a large application that is nothing like what they expected does not give a good impression and will require a lot of work to correct. It can be performed internally (Alpha testing) or by a select group of the clients users (Beta testing) before the finished product is given for final Acceptance Testing.

Non-Functional

* **Performance**: This evaluates how the system performs under various workloads, assessing factors like speed, stability, and responsiveness to ensure it can handle multiple users accessing it simultaneously. Many different criteria can be inspected including CPU usage, Memory usage, transactions per second or graphics performance.
* **Security**: This involves identifying vulnerabilities within the application, simulating potential attacks to uncover weak points in the code that could be exploited by hackers.
* **Stress Testing**: Verifies the behaviour of the application when the it is put under a heavy load such as a large dataset for processing, multiple concurrent requests or many simultaneous users. In game development, for example, insufficient stress testing can lead to a disastrous game launch when players are unable to use the game.
* **System** **Testing** verifies that the complete application works correctly in a real-world environment. It helps identify integration or functionality issues before the product is released.  
  Also known as end-to-end testing, system testing validates the complete and integrated software system against requirements. It ensures all system components work as intended before the product goes live.  
  System testing is performed after integration testing and before user acceptance testing (UAT). It is done once all components have been integrated and the system is ready for end-to-end testing against the requirements.

The most performant type of testing is automatic testing. Manual testing is very intensive of both time and manpower, but automatic tests are designed to be quick and stress free and at the click of a button.

By automating a large amount of testing we can allow the automation of subsequent processes such as delivery and deployment. Code pipelines allow a series of actions to be taken as code is added to an codebase, from executing unit test, deployment to testing servers, execution of scripts to build databases and even publication of software packages to public repositories such as nuget.

Automated testing may even be used without the knowledge of developers. For example, the code storage site GitHub processes check-in code for dangerous vulnerabilities or sensitive information. It can warn the developers or even take action to sanitise the check-in code automatically.

With the continuing development of AI, its use in automated development has started to become common place where AI will examine the code for known vulnerabilities performing thousands of tests in a few seconds without a human having to write them.