Table of Contents

Software testing

Software testing is an essential component of the software development life cycle (SDLC) for ensuring that software programs satisfy the desired quality requirements. Software testing is an important procedure in ALM (Application Lifecycle Management) and DevOps that helps to ensure that software is built, tested, and deployed quickly and efficiently.

ALM and DevOps are concerned with continuous delivery and continuous testing, in which software is tested at every stage of development, from basic design to final deployment. This method assists in identifying and resolving difficulties early in the development process, lowering the chance of costly errors and delays.

One of the primary advantages of software testing in ALM and DevOps is that it helps to ensure that software applications are of high quality and satisfy the required specifications. Developers can identify and repair flaws before they become serious problems by testing the software at each stage of the development process, lowering the chance of costly errors and delays.

Another advantage of ALM and DevOps software testing is that it can help to increase the speed and efficiency of software development. Developers can save time and resources while reducing the chance of errors or faults in software by employing automated testing tools and techniques.

Furthermore, ALM and DevOps software testing can help to minimize the entire cost of software development and maintenance. Developers can save time and resources by identifying and addressing issues early in the development process, as well as reducing the possibility of costly errors or delays in the program.

Software testing is an important procedure in ALM and DevOps since it helps to verify that software applications are of high quality, efficient, and cost-effective. Developers can save time, cut expenses, and improve the overall quality and performance of software systems by leveraging automation technologies and processes.

https://en.wikipedia.org/wiki/Software_testing

- Defects management & Quality Attributes
- vs Quality Management / Assurance
- Software Quality
- Azure Test Plans
- DevOps Verify
- Code Coverage

What is software testing?

Software testing is the process of verifying and validating software to ensure that it meets the desired requirements and quality standards.

Why is software testing important?

Software testing is important because it helps to identify defects and errors in software, improve the

quality of software, and reduce the risk of software failure.

What are some common types of software testing?

Some common types of software testing include unit testing, integration testing, system testing, acceptance testing, and regression testing.

What is the difference between manual and automated testing?

Manual testing is the process of testing software manually, using human testers to execute test cases and identify defects. Automated testing is the process of testing software using automated scripts or tools to execute test cases and identify defects.

What are some common testing techniques used in software testing?

Common testing techniques used in software testing include black-box testing, white-box testing, gray-box testing, boundary value analysis, equivalence partitioning, and error guessing.

What is the purpose of test cases in software testing?

Test cases are used to define the steps and expected results of a particular test scenario in software testing. They help to ensure that testing is performed consistently and thoroughly, and that defects and errors are identified and addressed.

What is the role of a test plan in software testing?

A test plan is a document that outlines the scope, objectives, approach, and resources of a software testing project. It helps to ensure that testing is planned, organized, and executed effectively.

What is continuous testing?

Continuous testing is the practice of testing software continuously throughout the software development lifecycle, using automated testing tools and techniques. It helps to identify defects and errors early in the development process, and ensures that software is delivered with the desired quality and performance.

What is exploratory testing?

Exploratory testing is the process of testing software without predefined test cases or scripts, using the tester's intuition and experience to identify defects and errors. It can be used to complement other testing techniques, and is particularly useful for finding defects that are difficult to identify using

scripted testing.

What is the difference between functional and non-functional testing?

Functional testing is the process of testing software to ensure that it meets the functional requirements, or what the software is supposed to do. Non-functional testing is the process of testing software to ensure that it meets the non-functional requirements, or how well the software performs in terms of factors such as performance, security, usability, and reliability.

> Software Testing RSS

Snippet from Wikipedia: ***Software testing***

> **Software testing** is the act of examining the artifacts and the behavior of the software under test by verification and validation. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to:
>
> - analyzing the product requirements for completeness and correctness in various contexts like industry perspective, business perspective, feasibility and viability of implementation, usability, performance, security, infrastructure considerations, etc.
> - reviewing the product architecture and the overall design of the product
> - working with product developers on improvement in coding techniques, design patterns, tests that can be written as part of code based on various techniques like boundary conditions, etc.
> - executing a program or application with the intent of examining behavior
> - reviewing the deployment infrastructure and associated scripts and automation
> - taking part in production activities by using monitoring and observability techniques
>
> Software testing can provide objective, independent information about the quality of software and the risk of its failure to users or sponsors.
>
> Software testing can determine the correctness of software under the assumption of some specific hypotheses, but testing cannot identify all the failures within the software. Instead, it furnishes a *criticism* or *comparison* that compares the state and behavior of the product against test oracles — principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria.
>
> A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions, but only that it does not function properly under specific conditions. The scope of

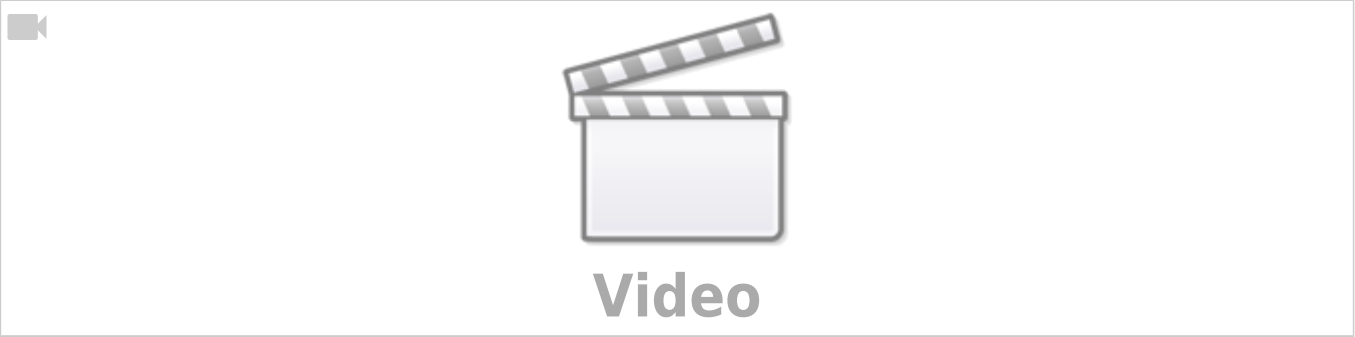
> software testing may include the examination of code as well as the execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.[:41–43]
>
> Every software product caters to a specific audience. For instance, the audience for video game software differs significantly from that of banking software. Therefore, when an organization develops or invests in a software product, it must assess whether the product aligns with the expectations of its end users, target audience, purchasers, and other stakeholders. Software testing plays a critical role in making this assessment.

GitHub Topics

- https://github.com/topics/testing

Testing is the practice of systematically testing software to make sure it works. Testing can be iterative, and happen multiple times.

Eliminate bugs and ship with more confidence by adding these tools to your workflow.



Source: YouTube

ISTQB

- https://www.istqb.org/

External links:

- https://www.guru99.com/test-management.html

Disciplines & Methodologies

- Feature toggle
- Git Flow
- Software deployment
- Software development process
- Software documentation
- Software quality assurance
- Software quality management
- SQALE
- Static program analysis
- Test-driven development
- Unit testing

Tools & Technologies

- Azure DevOps
- CMake
- codeBeamer
- Gauge
- Gerrit Code Review
- GitHub Learning Lab
- GitLab
- HP ALM
- Invoke-Build
- Jama
- Jenkins
- Microfocus
- Orcanos
- Postman
- Rally Software
- Rational solution for CLM
- Rational Team Concert
- ReQtest
- Selenium
- SoapUI
- SonarQube
- Team Foundation Server
- TeamForge
- Test automation
- Helix ALM (TestTrack)
- Visual Studio
- VSALM
- Visual Studio Team Services

Links

- GeeksforGeeks
- Good and Bad Technical Debt (and how TDD helps)
- Guru99
- List of tools for static code analysis

- Don't repeat yourself (DRY)
- EITBOK
- IEEE software life cycle
- Issue tracking system
- ISO/IEC 15504
- Technical Debt
- V-Model

governance, development

Related:

• Azure DevOps	tool, devops, devopsplan, devopscreate, devopsverify, devopspackaging, devopsrelease, requirements, programming, test, ci, projects, release
• CMake	tool, programming, test, devopscreate, devopsverify, devopspackaging
• Code Coverage	test
• codeBeamer	tool, test, release, devopsverify, devopsrelease
• Don't repeat yourself (DRY)	kb, programming, test, devopsverify
• EITBOK	kb, architecture, test, maintenance, release
• Feature toggle	method, architecture, programming, test, devopscreate
• Gauge	tool, test, devopsverify
• GeeksforGeeks	link, architecture, programming, test, maintenance, devopscreate, devopsverify, devopsconfigure
• Gerrit Code Review	tool, programming, test, devopscreate, release
• Git Flow	method, architecture, test, git
• GitHub Learning Lab	tool, git, programming, test, maintenance, change, devopscreate, learning
• GitLab	tool, requirements, devopscreate, git, architecture, programming, test, maintenance, ci, release
• Good and Bad Technical Debt (and how TDD helps)	link, test, devopsverify
• Guru99	link, architecture, programming, test, devopscreate, devopsverify
• Helix ALM (TestTrack)	tool, requirements, test, maintenance, change
• HP ALM	tool, requirements, architecture, programming, test, maintenance, change, ci, projects, release
• IEEE software life cycle	kb, requirements, architecture, test, maintenance, change, projects, release
• Invoke-Build	tool, test, devopsverify

• ISO/IEC 15504	kb, requirements, test, projects
• Issue tracking system	kb, test, maintenance, projects
• Jama	tool, requirements, devopsplan, test, projects
• Jenkins	tool, test, devopscreate, ci, release
• List of tools for static code analysis	link, architecture, test
• Microfocus	tool, requirements, test
• Orcanos	tool, requirements, test
• Postman	tool, architecture, programming, test
• Rally Software	tool, requirements, devopsplan, devopscreate, test, maintenance, ci, projects, release
• Rational solution for CLM	tool, requirements, architecture, programming, test, maintenance, change, ci, projects, release
• Rational Team Concert	tool, requirements, architecture, programming, test, maintenance, change, ci, projects, release
• ReQtest	tool, requirements, test, devopsverify
• Selenium	tool, programming, test, devopsverify, release
• SoapUI	tool, test, devopsverify
• Software deployment	method, programming, test, maintenance
• Software development process	method, requirements, architecture, programming, test, maintenance, change, ci, projects, release
• Software documentation	method, requirements, architecture, programming, test, maintenance, change, ci, projects, release
• Software quality assurance	method, test, devopsverify, projects
• Software quality management	method, test, devopsverify, projects
• SonarQube	tool, architecture, programming, test, maintenance, devopsplan, devopscreate, devopsverify
• SQALE	method, test
• Static program analysis	method, requirements, architecture, programming, test, maintenance, ci, projects, release
• Team Foundation Server	tool, requirements, devopscreate, architecture, programming, test, maintenance, change, ci, projects, release
• TeamForge	tool, requirements, architecture, programming, test, maintenance, change, ci, projects, release
• Technical Debt	kb, test, maintenance, change, projects, devopsverify
• Test automation	tool, programming, test, devopsverify, skill, automation
• Test-driven development	method, test, projects, devopscreate
• Unit testing	method, programming, test, devopsverify
• V-Model	kb, requirements, test, devopsverify, projects
• Visual Studio	tool, requirements, programming, test, devopscreate, devopsverify, ci, release
• Visual Studio Team Services	tool, requirements, architecture, programming, test, maintenance, change, ci, projects, release
• VSALM	tool, requirements, architecture, programming, test, maintenance, change, ci, projects, release

ToDo

- 

- Support Us... →
- Agile Testing
- API Testing Tools
- Appium
- Application Security Testing (AST)
- Application Testing Services
- AutoIt
- Behat
- BrowserStack
- Bugwolf
- Burp Proxy
- CA Test Data Manager
- CasperJS
- Cobertura
- Codacy
- CodeFactor
- CodeScene
- Concordion
- CrossBrowserTesting
- Cucumber
- Cucumber Testing
- Cucumber.js
- Cyara Velocity
- Cypress.io
- Defect Tracking
- Dynamic Application Security Testing (DAST)
- FitNesse
- Flood.io
- Functional Testing Tools
- Gatling
- Gauntlt
- GenRocket
- Google Test
- HttpMaster
- https://www.istqb.org/
- Inspec
- Jasmine
- JMeter
- JUnit
- Karate
- Karma
- Katalon Studio - Intelligent Test Automation
- Load Testing Tools
- Locust
- Mainframe Testing Tools
- Micro Focus LoadRunner
- Micro Focus Unified Functional Testing (UFT)

- Mobile App Testing Tools
- Mocha
- Nightwatch.js
- NUnit
- Parasoft API Test
- Parasoft Development Testing Platform
- Parasoft Environment Manager
- Parasoft SOATest
- Parasoft Virtualize
- Perfecto Continuous Quality Lab
- Performance Testing Tools
- Protractor
- pytest
- QF-Test
- Qualitia Automation Studio
- Qunit
- Ranorex
- Rational Integration Tester
- Rational Quality Manager
- Robot Framework
- Sahi
- Sauce Labs
- Screenster
- Selenium
- Service Virtualization
- Smartbear SoapUI
- SoapUI
- SpecFlow
- Squash TA
- Squash TM
- Test Architect
- Test Automation Tools
- Test Data Management
- Test Management Tools
- TestComplete
- TestNG
- TestObject
- TestRail
- Tricentis Tosca
- Watir
- xUnit.net

- Test Planning and Management
- Test Case Design and Execution
- Test Automation
- Functional Testing
- Non-Functional Testing
- Regression Testing
- Integration Testing
- System Testing
- Acceptance Testing

- Performance Testing
- Security Testing
- Usability Testing
- Exploratory Testing
- Test Reporting and Metrics
- Defect Management
- Test Environment Management
- Test Data Management
- Test Process Improvement
- Test Automation Frameworks
- Continuous Testing
- Shift-Left Testing
- Test-Driven Development (TDD)
- Behavior-Driven Development (BDD)
- Test Coverage
- Test Design Techniques
- Risk-Based Testing