TESTING & VALIDATION

Adnan Ibrahim Adnan Sawalha	0221536
Ahmad Hisham Sa'da Al-Hmouz	0228229
Basel Salah Mohammed-Saleh Qarout	0225322
Fares Mohammad-Saleem Hamdi Hatahet	0221413
Obada Riyad Mohammed Abu Shanab	0222730

Project Supervisor: Dr. Ahmad Kheraisat

University of Jordan

King Abdullah Second School of Information Technology (KASIT)

IT Project Management, 1904472

December 20, 2025

Introduction

Testing and validation are fundamental activities in IT project management that aim to ensure the successful delivery of systems that meet stakeholders' requirements and expectations. Although the two terms are often used together, they represent distinct yet complementary activities that contribute to project success. When applied throughout the project lifecycle, testing and validation support quality improvement, reduce the likelihood of failure, and help ensure that project outcomes align with organizational objectives. Overall, they play a key role in building trust and confidence among all project stakeholders.

Software **testing** is a set of activities that can be planned in advance and conducted systematically to detect errors and verify that the software meets customer requirements [1].

Validation is confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled [2].

The importance of testing and validation lies in reducing costs by detecting defects before they propagate to later, more expensive stages, managing schedule by controlling scope, avoiding delays, and ensuring high software quality, furthermore, they are essential in all IT domains to ensure reliability and security.

Role of Testing & Validation in IT Project Management

- **Relationship with requirements engineering:**
  Testing and validation ensure that requirements are correctly implemented and that the final system meets user needs and intended use.
- **Project manager responsibilities vs. testing team responsibilities:**
  The project manager is responsible for planning, scheduling, and allocating resources for testing and validation, while the testing team is responsible for designing, executing, and reporting test cases and defects [3].

Objectives of Testing & Validation

- **Defect detection:**
  Testing and validation aim to identify defects and failures to prevent faulty software from reaching end users.
- **Quality assurance:**
  Testing and validation ensure that the software meets defined quality standards and specified requirements.
- **Stakeholder confidence:**
  Effective testing and validation increase stakeholder confidence by demonstrating that the system is reliable and fit for purpose.
- **Regulatory / contractual compliance:**
  Testing and validation provide objective evidence that the system complies with regulatory standards and contractual requirements.

Testing & Validation in SDLC

Relation to SDLC Phases

Testing and validation are performed across all phases of the SDLC [4]:

- **Requirements phase**: checking the requirements. Make sure they are clear, complete, and reflect what the user wants.
- **Design phase**: testing here focuses on planning, how the system will be tested and whether the design will be tested properly.
- **Development phase**: unit and integration testing verify correct function and interaction of components.
- **Deployment phase**: system and acceptance testing confirm that the built system meets stakeholders' expectations in a real environment.
- **Maintenance phase**: regression testing ensures updates do not break existing functionality.

Waterfall vs Agile Testing

Waterfall Model

- Testing traditionally happens after development is finished.
- The process is sequential and phase-based, so development is completed before testing [5].
- Problems may be discovered later, which increases cost and risk.

Agile Model

- Testing is continuous and embedded within each iteration or sprint.
- Agile supports continuous validation and regular feedback help detect defects early [5]

Quality Assurance

Quality assurance provides the framework and processes that guide testing and validation activities to ensure the final product meets quality standards and user expectations.

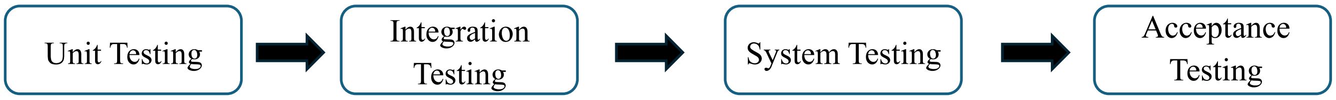
- **Static and Dynamic Testing**: Static testing involves reviews, inspections, and walkthroughs without executing the code, while dynamic testing executes the software to identify functional and performance issues [4].
- **Manual Testing**: Testing is performed by human testers without automation tools and is suitable for exploration, usability, and ad-hoc testing [6].
- **Automated Testing**: Test cases are executed using automation tools and scripts, making it effective for regression testing, performance testing, and repetitive tasks [7]

Levels of Testing and Software Testing Life Cycle (STLC)

Software testing is divided and categorized into two main classes:

- **Levels of Testing [8]**

Criteria	**Unit Testing**	**Integration Testing**	**System Testing**	**Acceptance Testing**
Purpose	To verify that individual units or components function correctly in isolation	To validate interactions and data flow between integrated modules	To ensure meeting specified functional and non-functional requirements	To confirm the system satisfies business requirements and is ready for delivery
Who Performs It	Developers	Developers & QA engineers	QA team	End users, clients, or business stakeholders



- **Logical Grouped Types of Testing [9], [10]**

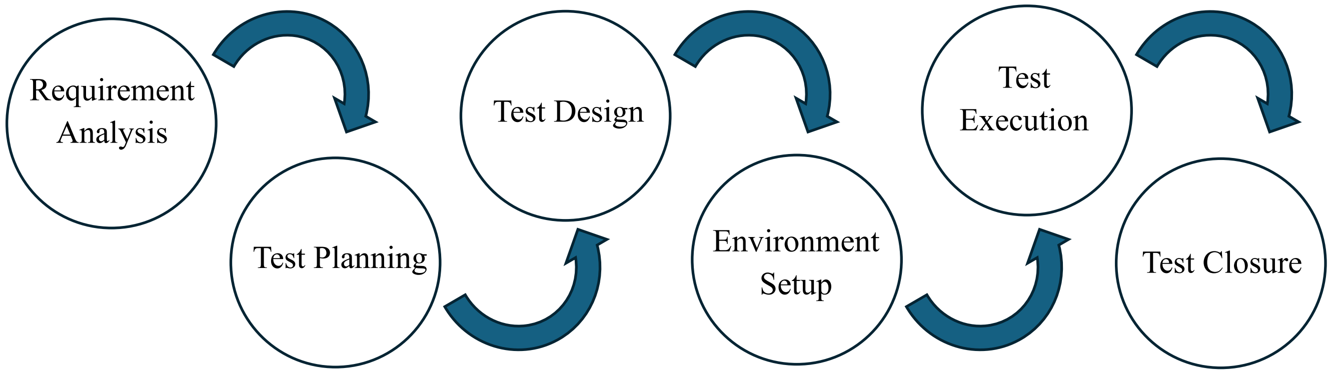
Testing types are logically grouped based on the evaluation aspect as shown in the following tables***Error! Reference source not found.***:

Testing Type	Purpose
Sanity Testing	A quick check to verify that specific functionality or bug fixes work after minor changes.
Regression Testing	Ensures recent changes do not break existing functionality
Smoke Testing	Confirms that core functionalities work in a new build, considered a quick testing technique

Testing Type	Purpose
Performance Testing	To assess system responsiveness and stability under expected workload conditions
Usability Testing	To evaluate how easily and efficiently users can interact with the system
Security Testing	To ensure protection of data through confidentiality, integrity, authentication, and authorization mechanisms

- **Software Testing Life Cycle (STLC) [11]**

STLC is a structured process that ensures systematic testing and validation of software quality.



Validation in Project Management

What validation means

Validation is the process of evaluating whether a software system fulfills its intended purpose and meets the real-world needs of its users and stakeholders. In project management terms, it answers the critical question: "Are we building the right system?" [12].

Validation techniques

- **Requirements Validation:** reviewing and analyzing the requirements document to ensure that they are clear, complete, consistent, and testable before development begins. Catching ambiguities or errors at this stage prevents costly rework later.
- **Prototyping:** Building early models or mock-ups to validate system behavior, usability, and functionality with users, ensuring the development team is on the right track.
- **Scenario-Based Validation:** This technique evaluates the system's behavior using real-world use cases, verifying that the system supports the users' daily tasks and meets business requirements before final deployment.
- **Stakeholder Reviews & Walkthroughs:** Involving stakeholders throughout all steps, including requirements, designs, or system outputs to confirm alignment with expectations.

When Validation Is Performed

- Starts early in projects stages.
- It continues during the design and development.
- Final validation occurs before system acceptance.

Why Validation Matters

- Ensures the system meets real user needs.
- Prevents building the wrong product.
- Reduces costly changes later.

Difference between testing and validation

- **Testing (Verification)** is a **product-focused** activity. It asks, **"Did we build the system right?"**. It focuses on identifying defects and ensuring the system works according to technical specifications.
- **Validation (Evaluation)** is a **user- and business-focused** activity. It asks, **"Did we build the right system?"**. It focuses on confirming that the system meets the user and business objectives.

Common challenges & Best Practices

Common Challenges:

- **Late testing involvement in the SDLC:** When testing is introduced late in the project, fixing problems becomes harder and more expensive. According to the IBM Systems Sciences Institute, fixing an error after product release can cost four to five times more than fixing it during design, and up to 100 times more than fixing it during maintenance. Delaying testing often causes time pressure, higher costs, and difficult rework. A well-known example is NASA's Mariner 1 mission, which was destroyed shortly after launch due to a minor coding mistake. 18 million dollars exploded just because of the lack of a hyphen in the code [13].
- **Frequent changes in requirements:** Poor communication and not involving users and stakeholders regularly often leads to misunderstanding of their real needs, causing frequent changes in requirements [14]. This makes testing difficult to manage and maintain, causing extra testing, and wasted time and money.
- **Time and budget constraints:** Testing is often limited by tight schedules and budgets. In large and complex projects, testing requires significant time and resources [15]. As a result, teams may focus only on critical features and ignore smaller details, which can reduce the overall quality of the software.
- **Inadequate test coverage and documentation:** Even experienced testers can miss a test case or two, sometimes hundreds. The number of possible bugs, especially those caused by user input, is very large and often unpredictable [15]. Because of this, unexpected errors should always be expected, and good documentation and coverage are essential.

Best Practices:

- **Early involvement of testing and validation:** Testing and validation should start early in the design phase and continue throughout the project [15]. Clear quality goals should be defined based on time and budget limits. Regular testing helps detect small issues before they become serious problems.
- **Clear and testable requirements:** Clear, measurable requirements are one of the most important best practices in software engineering [15]. They allow developers to measure system performance accurately. Unclear requirements lead to misunderstandings, communication problems, and unnecessary rework. Requirements should be clear from the start.
- **Risk-based testing:** Risk-based testing focuses on the most important parts of the system. Risk is defined by how likely a failure is and how severe its impact would be. This approach helps teams use limited time and budget efficiently by finding serious defects early instead of spending effort on low-impact areas [16].
- **Continuous stakeholder communication:** Stakeholders should be involved early and regularly. Continuous communication ensures that expectations are aligned, reduces unexpected changes late in the project, and lowers the cost of rework.

References

[1] R. S. . Pressman, *Software engineering : a practitioner's approach*. McGraw-Hill Higher Education, 2010.

[2] K. Esaki, "System Quality Requirement and Evaluation Importance of application of the ISO/IEC25000 series," 2013.

[3] Project Management Institute, *A guide to the project management body of knowledge (PMBOK guide)*. Project Management Institute, 2017.

[4] N. Honest, "Role of Testing in Software Development Life Cycle," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 5, pp. 886–889, May 2019, doi: 10.26438/ijcse/v7i5.886889.

[5] A. Sinha and P. Das, "Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry," in *2021 5th International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/IEMENTech53263.2021.9614779.

[6] G. J. . Myers, Corey. Sandler, and Tom. Badgett, *The art of software testing*. John Wiley & Sons, 2012.

[7] T. H. Tse, "'Book review: M. Fewster and D. Graham, Software Test Automation: Effective Use of Test Execution Tools, Addison Wesley, New York, NY, USA (1999)', ACM Computing Reviews," Dec. 1999.

[8] M. A. Umar, "Comprehensive study of software testing: Categories, levels, techniques, and types," Jun. 2020, doi: 10.36227/techrxiv.12578714.

[9] A. Mailewa, J. Herath, and S. Herath, "A Survey of Effective and Efficient Software Testing."

[10] H. S. Samra, "Study on Non Functional Software Testing," *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY*, vol. 4, no. 1, pp. 151–155, Feb. 2013, doi: 10.24297/ijct.v4i1c.3115.

[11] M. Parihar, "Role Of Software Testing Life Cycle(STLC) In Software Development Life Cycle (SDLC)," *International Journal of Research Available*, 2019, [Online]. Available: https://journals.pen2print.org/index.php/ijr/

[12] Ian. Sommerville, *Software engineering*. Pearson, 2011.

[13] QATestLab, "The True Cost of a Software Bug." Accessed: Dec. 19, 2025. [Online]. Available: https://qatestlab.medium.com/the-true-cost-of-a-software-bug-f8ee6a08b10b

[14] M. Kuštelega and R. Mekovec, "A Systematic Analysis of Requirements Elicitation Problems and Challenges."

[15] M. Tuteja and G. Dubey, “A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models,” 2012.

[16] M. Felderer and I. Schieferdecker, “A taxonomy of risk-based testing,” *International Journal on Software Tools for Technology Transfer*, vol. 16, no. 5, pp. 559–568, Dec. 2019, doi: 10.1007/s10009-014-0332-3.