**2-2 Journal: Dynamic and Static Testing**

**Static Testing**

Static testing involves evaluating software work products without executing the code. This form of testing includes reviews and static analysis of documents such as requirements, design specifications, and code. Static testing aims to identify defects early in the development lifecycle, making it easier and cheaper to correct them before the software is built and executed. Techniques used in static testing include formal and informal reviews, walkthroughs, and inspections.

**Dynamic Testing**

Dynamic testing, on the other hand, involves executing the code and evaluating the software's behavior during its execution. This type of testing is performed to validate the software's functionality and performance. Dynamic testing techniques include unit testing, integration testing, system testing, and acceptance testing. These tests are executed to identify runtime errors, logical errors, and performance issues.

**Static Testing versus Dynamic Testing**

The primary difference between static and dynamic testing is that static testing is conducted without executing the code, while dynamic testing requires code execution. Static testing focuses on reviewing and analyzing documentation and code for errors and inconsistencies. It can uncover defects early in the development process, such as ambiguities, omissions, and deviations from standards. Dynamic testing is concerned with validating the software's behavior during execution, identifying issues like runtime errors, performance bottlenecks, and incorrect functionality. Another difference lies in the methods used: static testing involves techniques such as inspections and walkthroughs, which are typically manual processes, while dynamic testing involves automated test scripts and manual test execution.

**Importance of utilizing both Static and Dynamic Testing**

Employing both static and dynamic testing is crucial for many reasons:

**Comprehensive Defect Detection**: Static testing can identify defects early in the development process, such as requirements and design issues, which dynamic testing might miss. Dynamic testing is essential for finding runtime/performance-related issues that static testing cannot detect.

**Cost-Effectiveness**: Detecting and fixing defects early through static testing is more cost-effective than addressing them later in the development cycle during dynamic testing. Early detection can save considerable time and resources.

**Improved Software Quality**: Using both testing techniques ensures a more thorough examination of the software, leading to higher quality and more reliable products. Static testing enhances the quality of documentation and code, while dynamic testing validates the software's functionality and performance.

**Risk Mitigation**: Combining static and dynamic testing helps mitigate risks by ensuring that both structural and behavioral aspects of the software are tested. This comprehensive approach reduces the likelihood of critical defects making it to production.

**Better Communication**: Static testing, especially through reviews and walkthroughs, promotes better communication among team members, fostering a collaborative environment where potential issues are discussed and resolved early.

**References**

Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). Software testing: An ISTQB-BCS certified tester foundation guide (4th ed.). BCS Learning & Development Limited.