Danny Forte

CS-320

Southern New Hampshire University

September 14, 2025

**CS 320 Module Two Journal: Static and Dynamic Testing Techniques**

Software testing is a critical component of the development lifecycle, ensuring that applications are reliable, secure, and meet user expectations. Two foundational approaches—static testing and dynamic testing—serve complementary roles in identifying and resolving defects. Understanding their distinctions and interplay is essential for building robust, trustworthy software systems.

Static testing is a verification technique that evaluates software artifacts without executing the code. It typically occurs early in the development process and focuses on reviewing documentation, design specifications, and source code. Techniques include code reviews, walkthroughs, inspections, and static analysis tools. These methods help uncover syntax errors, logic flaws, and security vulnerabilities before runtime, reducing the cost and complexity of fixing issues later in the cycle.

Dynamic testing, in contrast, is a validation technique that involves executing the software to observe its behavior. It assesses functionality, performance, and user experience under real or simulated conditions. Common dynamic testing methods include unit testing, integration testing, system testing, and user acceptance testing. This approach verifies that the software meets its requirements and performs reliably across environments.

The key difference between static and dynamic testing lies in execution. Static testing does not require the code to run, making it ideal for early detection of structural and logical issues. Dynamic testing, on the other hand, involves running the software to identify defects that only emerge during execution. Static testing is preventive, while dynamic testing is detective. Each targets different types of errors and contributes uniquely to software quality.

It is important to use both static and dynamic testing because they complement each other. Static testing catches issues early, often before code is compiled, which saves time and resources. Dynamic testing ensures that the software behaves correctly in real-world scenarios and meets user expectations. Together, they form a comprehensive strategy that improves defect detection, reduces rework, and enhances overall software reliability. For secure software development—especially in contexts involving cryptography, RESTful APIs, and certificate management—this dual approach is indispensable.