Emily Domoracki

Module Two: Journal

SNHU CS-320 | July 2023

Professor Johnathan Norman

Static testing is the practice of testing requirements and documentation by means of review and analysis, without executing code. Through static testing, we “find the cause of failures rather than the failure itself”. We can test functional and non-functional software requirements this way, as well as supporting and peripheral work products such as budget, schedules, and contracts. This style of testing illuminates redundancies, poor coding practices, overlooked specifications, and defects early on – contributing to reduced project fulfilment time and cost. Dynamic testing is typically performed after static testing. With this method, we are looking for defects during code execution. Different techniques are employed to test software according to predetermined test procedure. Both functional and non-functional requirements can be tested this way.

Both static and dynamic testing can be initiated formally and informally. Different contexts and work models require a nuanced approach. During static testing, we can use reviews to assess defects and inefficiencies in work products through checklists, dry-runs, and role or perspective-based assessments. The ascending levels of review are informal, walkthrough, technical review, and inspection. To inspect software code before execution, we can use tools to perform static analysis. This form of static testing checks for dead code, spelling errors, inefficient data structures, and deviance from best coding practices. Some of the methods we can use to test software dynamically – once the test procedures have been established – are black-box testing, white-box testing, and experienced-based testing. The black-box method uses test cases that directly address the work product specifications. This technique can utilize equivalence partitioning, boundary value analysis, decision table testing, and use case testing to check that the software adheres to the specified behaviors. White-box testing addresses software component structures. By reading and analyzing code, we can use automated tools and other techniques such as flow-charts, to ensure the program structure executes code successfully, securely, and efficiently. When specifications do not directly address a test basis, the tester may employ experience-based techniques to approach testing, such as error guessing or exploratory testing. Neither static testing nor dynamic testing alone are sufficient to examine a project for defects. It is important to include both in the SDLC, as illustrated above.