Static and dynamic testing, while different in application, are both critical to successfully testing software. “Static testing tests software and work products without executing code (Hambling et al., 2019, p. 17).” By analyzing documentation and reviewing code, the causes of errors and failures can be eliminated. External peer review of code and documents allows for an unbiased analysis and opinion, which can help identify issues early in the software development lifecycle (SDLC). Unlike static testing, dynamic testing “is carried out by executing actual code or the final software or hardware product (Hambling et al., 2019, p. 76).” Dynamic testing focuses on externally visible behaviors, such as what a system would display next or to a user. Since dynamic testing typically executes code or the entire software package, it can be tedious to identify issues. This is where static testing comes in, allowing developers to identify logical fallacies or inconsistencies prior to code development. Although static and dynamic testing are different, they are both necessary to test a system completely and accurately prior to deployment.

While both forms of testing, static and dynamic, have a similar objective to find defects and eliminate them, they accomplish this in a very different manner. As previously discussed, “static testing is carried out against work products without actually executing any code, whereas dynamic testing is carried out by executing actual code or the final software or hardware project (Hambling et al., 2019, p. 76).” Static testing is good at identifying and correcting requirements defects, design defects, coding defects, deviations from standards, incorrect interface specifications, security vulnerabilities, and gaps or inaccuracies in test basis traceability or coverage (Hambling et al., 2019, p. 77). By reviewing specification documentation and processes, static testing helps eliminate these “human-error” fallacies. Dynamic testing will exercise the code and can help determine if the code, logical conclusions within the code, and its design produce the expected output or result. From a static review standpoint, the code may be free of defects or logical inconsistencies. But when the code is put through dynamic testing, maybe the expected output isn’t necessarily what was expected or desired. This example helps illustrate a difficulty in testing software because the reviewed code may be correct technically but doesn’t give the desired or expected output. This can be confusing to developers because the code works the way it was written, but something else is wrong with a logical assumption or code execution. The code may be technically correct, but maybe it is technically correct but not in the manner that was intended. In my experience, this is when peer programming and peer review are critical because it allows for an unbiased opinion and analysis of the code which can help illuminate issues that may not have been seen.

References

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *SOFTWARE TESTING : an istqb-bcs certified tester foundation guide.* BCS Learning & Development Limited. https://ebookcentral-proquest-com.ezproxy.snhu.edu/lib/snhu-ebooks/detail.action?docID=5837074