**Static and Dynamic Testing in Software**

**CS-230 Module 2**

Damean Murphy-Short

September 4, 2023

In the realm of sotware testing, there are numerous ways to categorize testing methods, but one clear and useful distinction which can be made is between static and dynamic testing. Both static and dynamic testing have important roles to play throughout the software development life cycle (SDLC), but can be more useful than the other at different points. In this paper, each will be explored in detail, highlighting the differences between the two.

Static testing is the process of detecting defects without executing any pieces of code (Bellairs, R., 2023). This can be carried out in numerous ways, but most importantly, it can be carried out before any code has been written. By reviewing documentation, specifications, and written requirements before writing any code, a team can catch and remove a good deal of defects before fault-prone code is built upon a bad spec. It is also useful to go through “dry runs” once code, or a structure for the program, has been built, which allows one to logically ensure that everything on paper should function as expected according to previously set out guidelines. Static testing is a good way of ensuring simple defects like typos and ambiguities do not cause costly mistakes further into the SDLC.

Dynamic testing is the process of testing a software product by executing portions of its code. This ensures that the program, if tested against good test cases that adhere to a basis in the project requirements, actually delivers the functionality that has been laid out for it in the specifications that were reviewed in static testing. Some examples of dynamic testing are unit tests that verify a system produces the correct behaviour or output given any valid or invalid input.

As can be seen, while dynamic and static testing are mutually exclusive activities, they both play distinctly important roles in the SDLC. Without static testing to remove defects from the specification, dynamic tests may end up with false bases, causing both the codes and the tests to be rewritten. Similarly, one cannot leave a program statically tested without any dynamic testing, as the dynamic tests are what guarantee the code itself *does* adhere to functional requirements instead of postulating that it *will* adhere to requirements. Similarly, static testing may uncover defects in the actual product even when dynamic testing does not exhibit incorrect behaviour. For example, an experienced programmer looking at a line that allocates memory may see a potential memory leak that would not otherwise be tested for in dynamic testing.

References.

Bellairs, R. (2023). What is Static analysis? Static Code Analysis Overview. *Perforce Software*. https://www.perforce.com/blog/sca/what-static-analysis