

SOFTWARE TESTING (SE-C-04)

AIMS:

With the growing significance of computer systems within industry and wider society, techniques that assist in the production of reliable software are becoming increasingly important. The complexity of many computer systems requires the application of a battery of such techniques.

Two of the most promising approaches are formal methods and software testing. Traditionally formal methods and software testing have been seen as rivals. Thus, they largely failed to inform one another and there was very little interaction between the two communities. In recent years, however, a new consensus has developed. Under this consensus, these approaches are seen as complementary. This has led to work that explores ways in which these approaches complement.

The use of a formal specification or model eliminates ambiguity and thus reduces the chance of errors being introduced during software development. Where a formal specification exists, both the source code and the specification may be seen as formal objects that can be analysed and manipulated. The use of a formal specification introduces the possibility of the formal and, potentially, automatic analysis of the relationship between the specification and the source code. This is often assumed to take the form of a proof, but such a proof cannot guarantee operational correctness. For this reason, even where such a proof exists, it is important to apply dynamic testing.

Software testing is an important and, traditionally, extremely expensive part of the software development process. Studies suggest that testing often forms in the order of fifty percent of the total development cost. Where formal specifications and models exist, these may be used as the basis for automating parts of the testing process. This may lead to more efficient and effective testing. It may thus transpire that the automation of parts of the software testing process is one of the most significant benefits of using a formal specification language. The links between testing and formal methods do, however, go well beyond generating tests from a formal specification.

This module aims to introduce and critically analyse current techniques for software testing, in particular the importance of formal methods towards this goal.

LEARNING OUTCOMES:

Upon successful completion of this module, the student will be able to:

- critically evaluate the importance of software testing;
- appreciate the need and usefulness of formal methods for the testing process;
- develop an integrated approach for software testing and formal theories;
- critically evaluate testing tools.

SYLLABUS CONTENT:

- Introduction to testing fundamentals.
- Structural testing.
- Functional testing.
- Foundation for combining formal methods and testing.
- Assertion based testing.
- Model based formal methods.
- Finite state machine based testing.
- Testing from a process algebra.
- Testing with UML's dynamic models.
- Temporal logic, model checking and their role in testing.
- Tools for automatic testing and continuous integration.
- The management process of software testing.

PREREQUISITES: None

RECOMMENDED ASSESMENT: Coursework and unseen paper