Abstract

Testing has been the primary way for assuring the quality of a software product. The main challenge in testing is to generate effective test cases. Test case generation at the initial phase of SDLC has a greater impact than that made at a later phase. As the analysis and design stage is critical compared to other stages, detecting and correcting bugs at this stage is less costly compared to later stages of development. Designing metrics at this stage helps the test manager in making decisions for allocating time and resources. Unified Modeling Language (UML) is an added advantage to this which is a semi-formal visual language that supports the design and development of complex object-oriented systems. UML is a visual modeling language that can be used to specify, visualize, construct and document the artifacts of a software system. UML models are an important source of information for test case design. In this thesis, we propose techniques to generate, minimize and prioritize test cases for object-oriented software using behavioral UML 2.x models. The models used represent the relevant features of the systems under test (SUT).

First, we propose a cluster level testing methodology to test object-oriented software based on UML activity diagram. In this technique, an activity flow graph (AFG) is derived from an activity diagram. From the AFG, different control flow sequences are identified. Our technique achieves all activity path coverage criterion which undoubtedly increases the number of test cases and detects faults in loop, faults in decision and activity synchronization faults.

Next, we propose a novel approach to semi-automatically synthesize the integration of use case diagram and interaction overview diagram of a system. We have proposed an algorithm named Use Case Interaction Scenarios Generation (UCISc-Gen) for generation of test scenarios. The advantage of our test scenario generation technique is that it is helpful in detecting interaction and synchronization faults as the objects communicate with each other. We use message-activity path coverage criterion which is a stronger coverage criterion to achieve interaction faults when they communicate and activity synchronization faults when they execute. In our approach, we use UML 2.x diagrams instead of UML 1.x diagrams which contain more functionalities.

Subsequently, to deal with another challenging issue to minimize the test suite,
we present an algorithm, named *Test suite minimizer*, to minimize the generated test suites. Modified Condition / Decision Coverage (MC/DC) is a stronger coverage criterion than statement coverage and branch coverage which is used to generate minimal test cases. Here, we identify the number of predicates from the XMI file of an activity model of a ATM system under test. Then, by passing the predicates to a JAVA parser and by using dissimilarity matrix, we minimize the number of test cases. In our approach, we considered MC/DC coverage criterion and observe that the test-suite size is significantly minimized.

Finally, we present an integrated approach and a prioritization technique to generate cluster level test scenarios from an intermediate graph called Sequence Interaction Graph (*SIG*) constructed from the sequence and interaction overview diagrams, satisfying the *message-activity path* test adequacy criteria. The test sequences are a set of theoretical paths starting from initial node to end node and edges along the path, while taking conditions (pre-condition and post-condition) into consideration. Our approach achieves *message-activity path* coverage. We have named our approach *Test Case Prioritization using UML sequence and interaction overview diagrams (TCPN)*.

Besides the theoretical analysis, studies conducted by us, we have also implemented our algorithms to experimentally verify their *correctness* and *preciseness* with different case studies.

**Keywords:** Activity Diagram; activity flow graph; APFD; Backward slicing; Interaction Overview Diagram; Method-Activity Path Coverage; minimal test cases; Model dependency graph; Sequence Diagram; Sequence Interaction graph; UML 2.x diagrams; Test case; Use case Diagram.