Abstract

This thesis investigates the problems of generating system level test cases from activity and sequence diagrams. First, an approach for generating system level test cases using activity diagrams is presented. In order to interpret nested structures resulting from different combinations of control constructs such as loops, decision, concurrency and synchronization, a classification of control flow constructs is presented. On the basis of classification, transformation procedure for transforming an activity diagram to an intermediate representation known as *Intermediate Testable Model* (ITM) is defined. The flow graph analysis of procedural programs from the fields of code optimization and compiler technique (Hecht; 1977) provides the definition of regions (Aho et al.; 1986) which forms the foundation for the proposed transformation procedure. The control flow analysis reveal interesting insights into the structure of the activity diagram which can be exploited for test case synthesis. The transformation procedure also satisfies the properties like confluence, completeness and termination.

Next, the approach proposed for activity diagrams is extended to sequence diagrams. In order to generate scenarios from a sequence diagram containing different fragments and their nesting, a two-phase approach is proposed. In the first phase, control flow analysis of UML 2.0 sequence diagram is carried out. A directed graph representation of sequence diagram known as scenario graph is presented. In the second phase, ITM is generated from the scenario graph. It is shown that ITM can be applied in a similar manner as proposed in the first approach.

Finally, to automate the test execution, the synthesis of test data using UML model artifacts is presented. The test data synthesis approach is a domain based derivation procedure where the domain of a variable is a set of all values that the variable may take. The initial domain is derived from the *type* information associated to the variables of a corresponding method call in the class diagram. In addition, the OCL constraints associated to attributes of a class diagram is used to reduce the initial domain. The type and constraint information from class diagram and OCL constraints are integrated into scenario graph and test scenarios are then generated. For each test scenario, the approach finds a feasible domain of variables satisfying all the constraints along the scenario.

Keywords – Model-based testing, test case generation, test data synthesis, Unified Modeling Language, automatic system level testing