Abstract

Many researches to testing object-oriented systems (OOSs) have been proposed for the past decade. After all, almost all large OO software specifications still contain incompleteness, inconsistency, and ambiguity. The framework can be defined using any state-based specification notation and used to derive test cases from state-based specifications in this paper, it is demonstrated using the RSML notation. A state transition diagram (STD) derived from RSML specification provides a complete behavior of a given OOS. System testing is concerned with testing an entire system based on its specifications. In the context of object-oriented, UML development, this means that system test requirements are derived from UML analysis artifacts such as use cases, their corresponding sequence and collaboration diagrams, class diagrams. The goal is here to support the derivation of functional system test requirements, which will be transformed into test cases once we have detailed design information. In this paper, we describe a methodology in a practical way and illustrate it with an example. In this paper a framework that formally defines test data sets and their relation to the
Designing of Testing Framework through Finite State Automata for Object-Oriented Systems using UML

operations in a specification and to other test data sets, providing structure to the testing process.

References

   Proceedings Second International Conference on the Unified Modeling Language (UML'99),
   Fort Collins, CO. pp. 416-429.
   Computer Programming. vol. 8, pp. 231-274.
   Approach to Class-Level Testing of Object-Oriented Programs. ACM Transactions on Software
   Object-Oriented Software Testing and Metrics. International Journal of Software Engineering
   20, no. 9, pp. 684-707.
    Engineering. vol. 17, no.3 pp.241-258.
11. J. D. Ullman and J.E. Hopcroft. 1979. Introduction to Automata Theory, Languages and
    489-521.
    244-263.

Index Terms

Computer Science | Software Engineering
Keywords

Object-Oriented Program; software testing; software complexity; Finite automaton; UM