ABSTRACT

Specification-Based Testing offers many advantages in Software Testing. The specification of a software product can be used as a guide for designing functional tests for the product. The specification precisely defines fundamental aspects of the software, while more detailed and structural information is omitted. Thus the software tester can have the important information about the product's functionality without having to extract it from unnecessary details. The specification is an authoritative description of system behaviour and can be used to derive test data (Stocks et al 1993).

With so many advantages to be gained, the research into specification-based testing has recently received renewed attention. Various methods have been employed to derive test information from specification concentrating on generating test data. Many researches have been aimed at simultaneously achieving high efficiency and reduced cost of testing by selecting appropriate test cases (Noritaka Kobayashi et al 2001). Since the cost of testing is controlled by avoiding redundant and unnecessary tests (Hong Zhu et al 1997), the current research focuses on generating efficient test data.

Specification-Fault-Based Testing attempts to detect faults in the implementation that are derived from mis-interpreting the specification. This testing involves planting faults into the specification. In mutation analysis, a large number of mutants are generated. A mutant represents a possible fault.

Mutation analysis requires extensive computation resources to store and execute a large number of mutants. It also requires huge human resources to determine if live mutants are equivalent to the original specification. Reduction of testing expense to a practically acceptable level has been the focus of current research in this field.
Testing is fault based when its motive is to demonstrate the absence of pre-specified faults. Fault-based testing is adopted from mutation testing (Morell 1990). Fault based testing theory does not rely on the ability to generate and execute all alternatives, as is done in mutation testing. The development of cost-effective tests for larger systems may well require a move towards test stimuli targeting actual faults.

A number of methods have been proposed for generating test cases from predicates in a specification. These methods derive various test conditions from logic expressions with the aim of detecting a variety of faults. A fault based approach of generating test cases selects test cases that aim at detecting certain types of faults. This approach is similar to hardware testing where manufacturing flaws are hypothesized and then test sets are derived to detect these flaws. Given a particular specification, it is possible to compute the conditions under which the fault will cause failure.

There is a hierarchy of fault classes (Kuhn 1999) in specification-based software testing. The relationships among different types of faults for boolean expressions can be established as a hierarchy. Test cases that are generated to detect the faults at the lower level of the hierarchy will cover a larger set of faults. If a test causes $S \oplus S'$ to evaluate to true, then the fault will be detected, where $S$ and $S'$ denote a specification and its faulty implementation respectively. $S \oplus S'$ is referred to as the fault condition for that fault (Kuhn 1999, Tsuchiya et al 2002). Once a particular fault class is hypothesized, it is therefore possible to generate test cases that can detect all the faults in the class and the procedure for generating test cases has been proposed.

The fault condition proposed by Kuhn (1999) was applied to various types of faults and test sets were generated. A new fault class namely, Term Insertion Fault is introduced. For some faults, a simplified fault condition than the one proposed by Kuhn (1999) has been obtained.
The test sets of various fault classes are analyzed and it is observed that the test set for various faults can be derived from a single fault class. Literal Reference Fault is analyzed and expressions are derived from Literal Reference Fault to detect various fault classes. The hierarchy of fault classes proposed by Lau et al (2001) is analyzed and empirical results are proposed. The hierarchy is extended by adding the new fault, namely, Term Insertion Fault.

The Null Fault is defined and a method to detect it is proposed. During the analysis of hierarchy of fault classes, certain exceptions surfaced. These exceptions have been studied and analyzed. It is observed that whenever the faulty implementation of the specification is a different form of the original specification, the fault condition proposed by Kuhn (1999) does not generate any test case. In this case, the faulty implementation is regarded as the equivalent formula of the original specification. Further, to overcome the exceptions that occurred in the hierarchy of fault classes, test set generation for various fault classes from Literal Negation Fault is proposed.

The characteristics of various faults have been studied. The study helped to propose a new fault condition that is computationally efficient and it generated smaller test sets for some faults. The new fault condition is compared with the fault condition proposed by Kuhn (1999) and the results are tabulated.

The results have been tested with real life Traffic Alert and Collision Avoidance Subsystem (TCAS II) specifications. The results obtained are compared with the existing work, wherever applicable. The research work provides a good number of research avenues and may stimulate interest among other researchers in the field of fault-based testing.