IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITB10024

Chapter VIII

Regression Test Selection for Database Applications

Ramzi A. Haraty, Lebanese American University, Lebanon

Nashat Mansour, Labanese American University, Lebanon

Bassel A. Daou, University of Ottawa, Canada

ABSTRACT

Database applications features such as Structured Query Language programming, exception handling, integrity constraints, and table triggers pose difficulties for maintenance activities, especially for regression testing that follows modifying database applications. In this chapter, we address these difficulties and propose a two-phase regression testing methodology. In phase 1, we explore control flow and data flow analysis issues of database applications. Then, we propose an impact analysis technique that is based on dependencies that exist among the components of database applications. This analysis leads to selecting test cases from the initial test suite for regression testing the modified application. In phase 2, we propose two algorithms for reducing the number of regression test cases. The Graph Walk algorithm walks through the control flow graph of database modules and selects a safe set of test cases to retest. The Call Graph Firewall algorithm uses a firewall for the inter-procedural level. Our experience with this regression testing methodology shows that the impact analysis technique is adequate for selecting regression tests and that phase 2 techniques can be used for further reduction in the number of these tests.

INTRODUCTION

Software maintenance involves changing programs due to errors, alterations in user requirements or changes in the hardware/software environment. Regression testing is an important activity of software maintenance, which ensures that the modified software still satisfies its intended requirements (Hartmann & Robson, 1989). It attempts to revalidate modified software and ensure that new errors are not introduced into the previously tested code. Regression testing involves four issues: change impact identification, test suite maintenance, test strategy, and test case selection. Change impact identification involves locating all the modules and other program segments that are affected by the modification. Test suite maintenance attempts to keep the test suite status current and reusable for future revalidation. Test strategy involves finding a test sequence for retesting the software. Test case selection attempts to reduce the cost of regression testing by selecting a subset of the test suite that has been used during the application development. This subset of tests is then used to test modified programs (Rothermel & Harold, 1998).

In database applications a number of new features are supported, such as Structured Query Language (SQL) statements, table constraints, exception handling, and table triggers. These features introduce new difficulties that hinder regression test selection. In this work, we concentrate on impact analysis and test selection for SQL-based systems. Regression testing is necessary for assuring the quality of a system after modifying it. Ad hoc regression testing involves either rerunning all the test cases that are included in the test suite determined during the initial development of software (Select-All approach) or selecting a random subset of this initial test suite (Select-Random approach). But, the Select-Random approach is unreliable, since it might miss selecting test cases that reveal adverse effects of modifications. Hence, the Select-Random approach might compromise the quality of the modified system. On the other hand, the Select-All approach is expensive in terms of time and cost, since it usually includes many test cases that do not reveal the impact of the modification made to the system. Therefore, it is important to use regression testing methods that reduce the number of selected test cases in order to save time and money, especially for large software systems, while maintaining the quality of the system (Wong et al., 1997).

SQL, the standard query language, is a declarative language used for the manipulation of table data in database applications. It stands as the heart of database applications modules (ISO/IEC 9075, 1992). The usage of SQL in a procedural context has its implications. We categorize these implications into three categories: control dependencies, data flow dependencies, and component dependencies. The nature of SQL and the existence of table constraints lead to using exception handling techniques in database modules. Exception handling complicates control flow dependencies between statements in database modules. This complexity should be handled in the process of applying control flow-based regression testing techniques. Moreover, table triggers firings because of modifying SQL statements create implicit inter-modular control flow dependencies between modules. These dependencies should be explored for performing inter-module regression testing.

The manipulation of database tables by different modules, using SQL, leads to a state-based behavior of modules. It also creates data flow dependencies between the modules. The dynamic behavior of SQL, in which the exact table rows manipulated is not known until run-time, makes it very difficult to trace such data dependencies. Furthermore, SQL manipulates database components such as tables and views. These facts create component dependencies between the various components handled by SQL statements and the modules in which the statements are located. These component dependency relations are transitive. Whenever a change is made to one component, this transitivity introduces a ripple effect of change.

In this chapter, we propose a new two-phase methodology for regression testing SQL-based database applications. Phase 1 involves detecting modifications and performing change impact analysis. The impact analysis technique localizes the effects of change, identifies all the affected components, and selects a preliminary set of test cases that traverse modified