

Increasing Software Safety: Moving Test Support to the Runtime Image

Ronald Finkbine, Ph.D., Indiana University Southeast, rfinkbin@ius.edu

ABSTRACT

Traditional software development has maintained separate phases for software development and software testing has always been distinct and well removed from the binary image that resides and executes on production computer platforms. Active testing modifies this distinction by continuing to maintain testing as a distinct phase in the software development life cycle (SDLC), but also to provide a copy of test support (test scripts and test cases) into the production software system. Under active testing, computer software will still be fully tested prior to deployment but the in-production software also will allow intermittent or on-demand testing during the runtime of this system on the production hardware in the production environment. This increases software security and safety by allowing for run-time verification of software. This paper describes a project under development that uses active testing to allow for intermittent testing of software during run-time.

The following tools are part of this project and will be described.

Database: A database is necessary for storage of the test suite. The test suite can be maintained by the testing team both during development and during run-time.

Test-Case-Control-File-Set: Each file in this set contains the exact output for one successfully completed test case. Since the test suite should contain a good number of test cases, there could be many members in this set and it can change during the production phase of this system by the addition or deletion of test cases.

File-Compare-Utility: This would test the output from an executed test case against its associated control file. This byte-for-byte comparison is the same as the function computed by the Unix/Linux utility *diff* or the DOS utility *comp*.

Test-Case-Insertion-Function: Test case insertion is used for introduction of a test case into the production system. The insertion would require a test case description and the control file (expected output). This function can be restricted to supervisory personnel if the development is a high-integrity software project.

Test-Case-Removal-Function: Conversely, test case removal is available for a test case determined to be inappropriate or redundant. This function can also be restricted to supervisory personnel.

Test-Case-Generator: Test case generation can be used to expand the test suite. This is a proposed extension to the project and would require review of current research results in the field of test case generation.

Test-Suite-Driver: A test case script for one-step execution of the entire test suite implemented as a sequential testing of each test case is necessary. The output of each test case execution is put into a file and the file comparison utility described above is used to determine if the output file just produced matches the control file byte-for-byte. If all test cases are passed then a file attesting to this is generated. This summary file can be encrypted and emailed for software development in a high-integrity development environment.

This project is being implemented in the Java language using a MySQL database. Development is occurring on a Windows XP platform with testing and porting of the software to a Linux platform. In order to increase the production effort on this project, a grant application is being prepared.

AUTHOR INFORMATION

Dr. Finkbine is an Assistant Professor of Computer Science at Indiana University Southeast and has a Ph.D. in Computer Science from the New Mexico Institute of Mining and Technology.