A Comprehensive Study on Code Coverage Analysis for effective Test Development/Enhancement Methodology

Biswadeb Bandyopadhyay
Assistant Professor, Department of Computer Applications (CA)
University of Engineering and Management, Kolkata
biswadeb.bandyopadhyay@uem.edu.in

Abstract-The paper describes a tool developed by a team of testing practitioners dealing with a large collection of test suites used for testing Teradata which is large relational database system software. This team has studied the existing database software, available test suites, collected the code coverage data for a large number of available test suites for Teradata and developed an effective Web based Code coverage analysis tool to analyze these Code coverage performance of these test suite while running on Teradata to be able to understand code coverage of these test suites data in various possible dimensions. This activity was undertaken as part of a test engineering initiative to bring in place a set of innovative test engineering practices as potential business value adds.

1. Introduction

An effective way to measure the Quality of software product is the amount of code that has been tested (i.e. Code coverage). While this does not guarantee that the code is defect free, the risk of uncovering more defects from the customer's site is reduced considerably as more code is tested during the product test cycle. It should be realized that even 100% coverage does not guarantee a defect free code. Most Test engineer would agree that while one can never be sure of a bug free code, a significant milestone is achieved when "all the code has been tested." Code coverage can be a valuable measure, especially when time is taken to achieve a high coverage value.

The idea behind the code coverage is to improve the test cases by

- Identifying the uncovered code by the existing test suites, adding the new test cases and thereby improving the test suites.
- Identifying the redundant test cases in the existing test cases, thereby reducing the execution time of the test cases by removing them.

While working with these test suites, the team took an initiative to analyze the code coverage for all the available test suites to get some degree of confidence as to the existing level of code coverage. Code coverage provides a deep insight into the adequacy of the test cases and the need of or scope for improvement. The team undertook a comprehensive analysis of these test suites and collected code coverage data for a large number of such test suites. A Web based tool was developed where all these coverage data were stored, to order to do a comprehensive analysis of these code coverage data in various dimensions.

The Web based code coverage analysis tool developed by the team provided a convenient platform from where the user can obtain and analyze the code coverage data for various test suites. This proved to be an effective tool to quickly understand and analyze the test coverage scenarios.

The benefits of this tool were to be able to generate the following analysis reports

- Line level, function level and module level code coverage reports
- Annotated source code for function wise, module wise and test suite wise coverage data
- Annotated source code of a selected implementation file with lines hit, lines not hit, lines partially hit
- Analytical report of code coverage of a selected implementation file for various test suites
- To provide information on the most appropriate test suites to validate a bug fix/code

enhancements which will guarantee the maximum statement coverage of the file being added/modified

- To analyze any field reported problems, to identify whether the root cause of the failure was due to non-coverage of the code segment where the fix for the problem was found
- To identify the root cause of any regression problems due to any limitation of existing test suites used for regression testing

2. Why Rational's Pure Coverage:

There are lots of profiling tools in the market. But most of the tools do not support for server side, i.e. applications that will run continuously. They will generate coverage data only on graceful exit of the application. But Rational's Visual Purecoverage tool generates a detailed report of the code coverage, even the application terminates abnormally.

There two flavors of Purecoverage packages from Rational, one is on UNIX and another for Windows. On UNIX, it is necessary to instrument (adding Purecov option) at the time of compilation itself. So it would necessitate to re-build the package. But on NT we can instrument after the compilation, so there is no need to re-build the package. It is also possible to instrument any particular EXE/DLL that may be neede. But these should be built without any optimization options.

It is possible to save the coverage data of Purecoverage either in CFY format (only Purecoverage can open) or in ASCII Text format.

3. Implementation

Following section describes development of a code coverage analysis tool by a team of test practitioners who were involved in the development of test suites for testing a large relational database management system. The team utilized its experience of developing these test suites to extend it further to assess effectiveness of these test suites in terms of code coverage performance of these tests on the system under test. Essential idea was to develop such a code coverage analysis tool and utilize this tool in analyzing the test suites being developed, use that information for rationalizing the tests by eliminating the tests that are redundant, augmenting test suites with new test cases to improve the code coverage further, thereby enhancing the effectiveness of the test suites.

The test team developed a web based "Code Coverage Analysis tool" that provides a convenient platform from which the user can obtain and analyze the code coverage data for various Teradata Regression Test Suites. This data was collected using Rational's Purecoverage tool. The Code Coverage Analysis Tool provides all information such as percentage of DBS code coverage for different regression test suites, at module level, function level and line level. This provides complete information about overall code coverage performance of each individual test suites as a whole.

3.1 Minimum Cost – Maximum Coverage Model

To implement Minimum Cost - Maximum Coverage model for rationalizing Test suites and enhancing Code Coverage.

Cost parameters for optimization:

- Line Coverage (Ci)
- Number of Test Suites (Ti)
- Execution Time (Ri)

Min Cost –Max Coverage Model optimizes:
Min (Ti)=Max (Ci) & Min (Ri)

- Track the line coverage and minimize overlap across the tests
- Minimal test sets to cover Maximum source code

3.2 Tool is used by the Developers

- To identify the areas of code which are not covered or partially covered in order to improve the test cases
- To use the tool as a workbench for ensuring that the test cases they generate are adequate

3.3 Tool is used by the Testers

- To choose the Test Suites based on their percentage of coverage for a
- given source file
- To Correlate the Test Suites for their coverage for a given source file

3.4 Features of the Tool

Tool was built around Java Servlets, Java Server Pages, HTML, Apache Web Server, Jakarta-Tomcat, Carcase Version Control Software, Rational Pure Coverage Tool, Unix Shell scripts and Teradata RDBMS.

Feature includes:

- Data extraction
- Data loading
- Report generation with Code Coverage Analysis
- using HTML / Java Servlets / Apache Web Server / Jakarta-Tomcat / Teradata ODBC driver/ JDBC-ODBC bridge

Tool generates following reports:

- Correlating different Test Suites for Line-wise coverage for a given source file
- Function-wise coverage for different Test Suites for a given source file
- Pictorial presentation of Module-wise coverage for a given Test Suite

- Highlighting source file for coverage data with annotations

- Blue indicating covered lines
- Red indicating uncovered lines
- Pink indicating partially covered lines
- Showing list of Test Suites in ascending order of coverage for a given source file

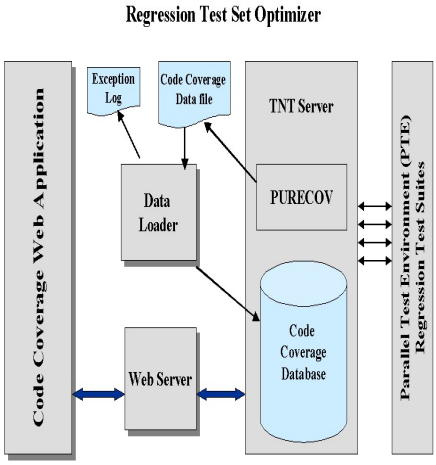

Figure 1 - Regression Test Set Optimizer

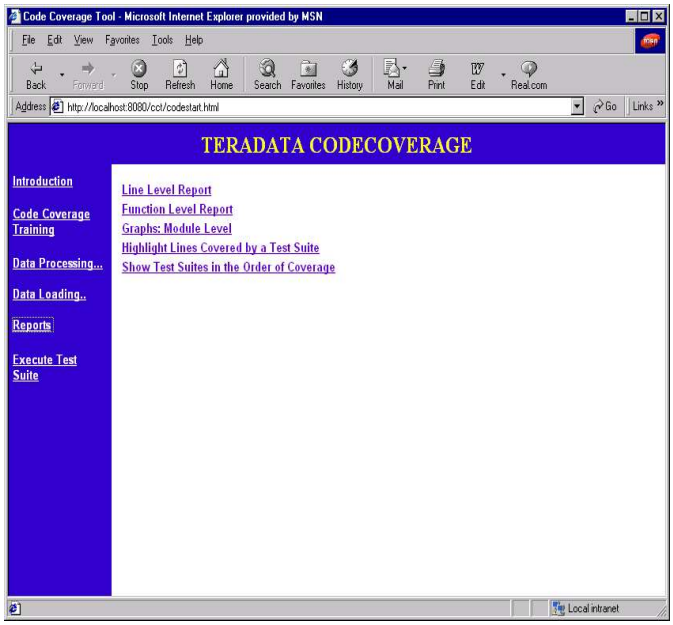

Figure 2 - Teradata Code Coverage Tool

Figure 3 - Line Wise Code Coverage Menu

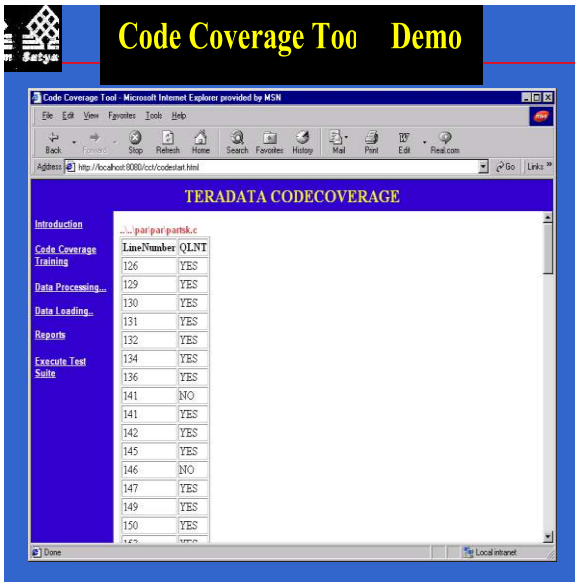

Figure 4 - Line Wise Code Coverage Output

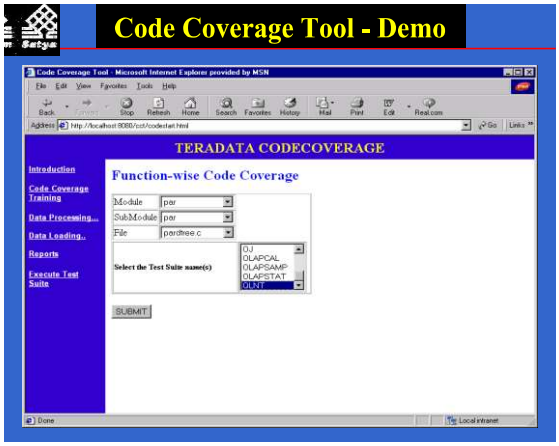

Figure 5 - Function Wise Code Coverage Menu

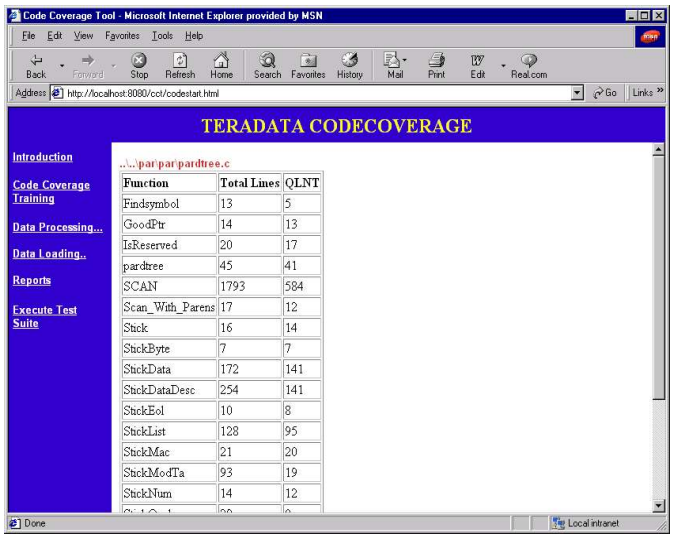

Figure 6 - Function Wise Code Coverage Output

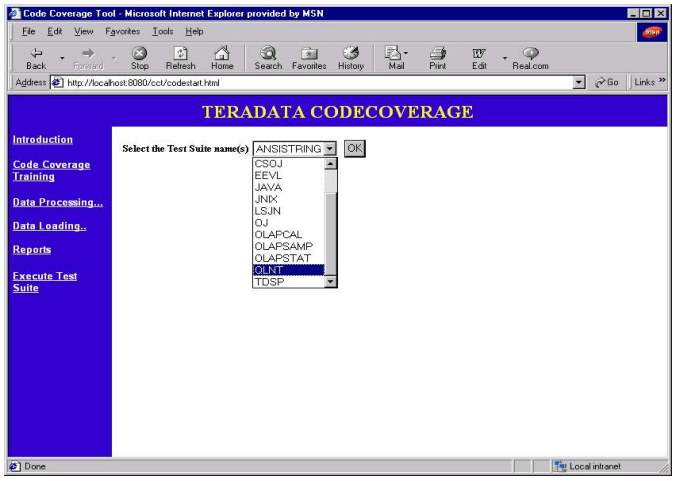

Figure 7 - Test Suite Wise Coverage Menu

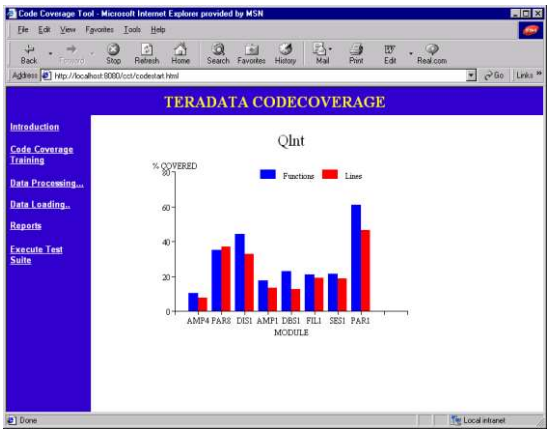

Figure 8 - Coverage Output of a particular Test Suite

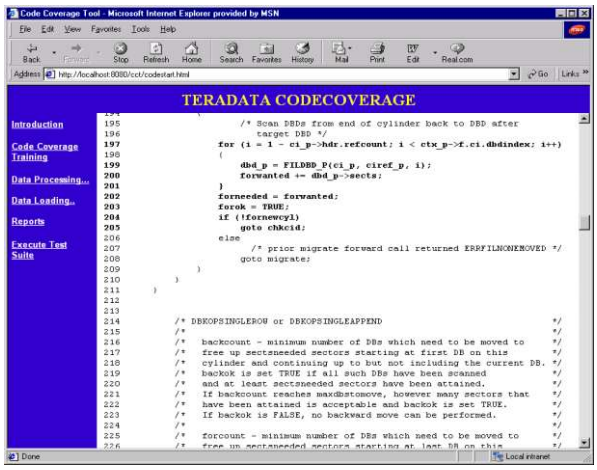

Figure 11 - Line Wise Coverage Output of a Source File

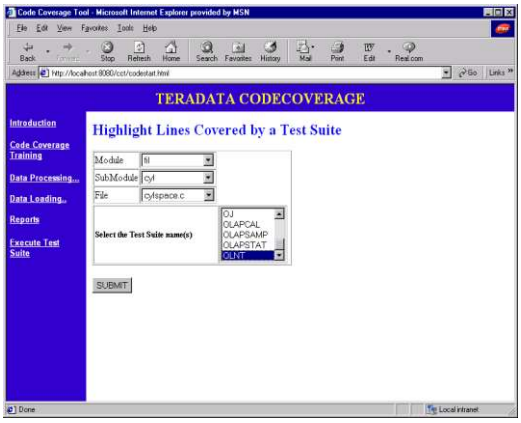

Figure 9 – Highlight Lines Covered by a Source File Menu

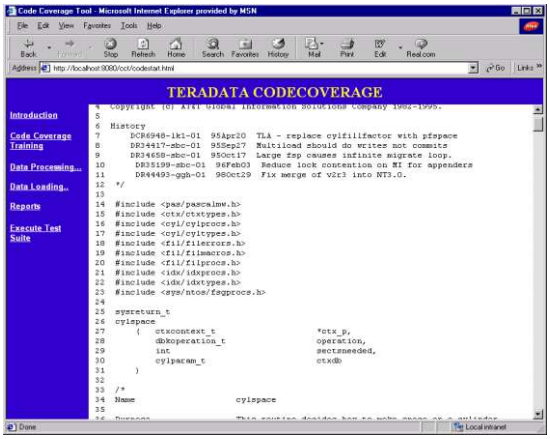

Figure 10 - Annotated Source Code of a File

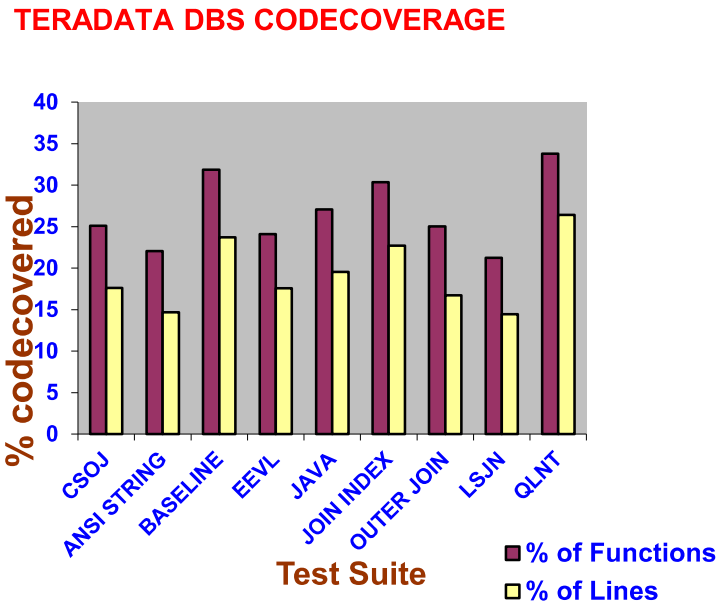

Figure 12 - Test Suite Wise Code Coverage Output

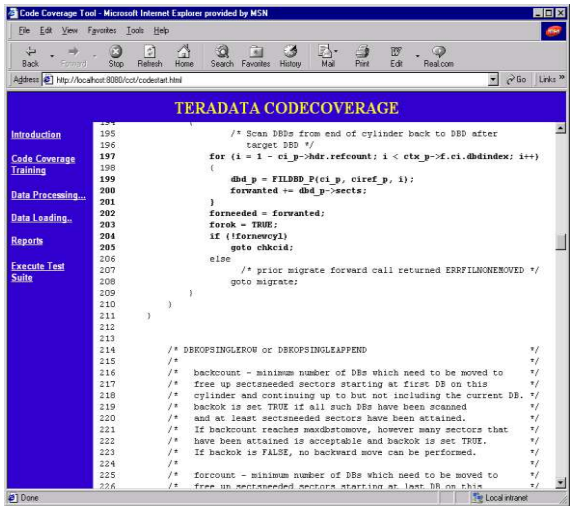

Figure 13 - Annotated Source Code of a File

4. Conclusion

Tool developed by the turned out to be quite effective in performing a number of activities. These included, Rationalizing the Test Suites, Generating Traceability Matrix, Improving Percentage of Code Coverage, Analysis of field encountered problems, to identify the root cause of any regression problems, due to any limitation of existing test suites used for regression testing, non-coverage of the code segment where the fix for a particular problem was found. The tool is unique in the sense that it focuses on optimizing the test suites from code coverage perspective of test suites, providing dual benefits simultaneously by improving code coverage and optimizing the test cases, which in turn results in reduction of test cycle time.

References

1. Effective Methods for Software Testing, William E. Perry, Willey Publication
2. Software Engineering: A Practitioner's Approach, Roger S. Pressman
3. Software Testing Concepts and Tools, Nageswara Rao Pusuluri
4. Software Testing in Real Worls, Edward Kit
5. Code Coverage vs Test Coverage: A Detailed Guide, Shreya Bose
6. Efficient use of code coverage in large-scale software development, Yong Woo Kim, CASCON '03: Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research
7. Code coverage analysis in practice for large systems, Yoram Adler; Noam Behar; Orna Raz; Onn Shehory; Nadav Steindler; Shmuel Ur; Aviad Zlotnick, 2011 33rd International Conference on Software Engineering (ICSE)
8. A REVIEW ON CODE COVERAGE ANALYSIS SARITA PATHY1 P.G. Dept. Of Computer Science and Application Jyoti Vihar, Sambalpur University, Burla, Sambalpur,Odisha,India