

Improving Combinatorial Methods to Optimise Efficient Test Suite Generation

ICT Centre Conference 2008

Poster ID:

158

Chris Stanbridge, John Ryan-Brown
The Australian E-Health Research Centre CSIRO

Introduction

Testing of all combinations of components and their variables in a system is dubbed “exhaustive testing” for a good reason. As cost constraints for system testing limit the extent of quality assurance, efficient methods of generating test suites for black-box testing become critical.

Combinatorial Testing

The efficiency of a test suite can be measured by the number of test cases necessary to uncover the defects in a system (providing the time to execute a test is approximately constant for all tests in that suite).

The number of test cases generated and requiring execution for exhaustive testing is:

$$\prod_{i=1}^N v_i$$

Where v_i is the number of variables each one of the N components can take.

This test strategy is the gold standard in black-box testing and (providing failures are discrete and identifiable) can be regarded as the method by which the greatest number of functional defects in the system will be detected. Inclusion of a new component in the system that can take two variations of input (say a radio button on a web page) will double the number of tests in the suite.

The other end of the test suite scale is single mode testing where the tests exercise each single component while disregarding the effects of combinations of inputs. This considers a component as a singleton and is normally executed as part of unit testing.

The hypothesis that an optimal combinatorial test strategy lies between these two extremes was investigated using a real system under test with known seeded defects and a tool developed in house to generate suites of tests based on an increasing combinatorial strength.

The System Under Test

The test system was a feature-set of a data linking application developed at AEHRC. Components were identified [Figure 1] with each component taking a number of compatible values as inputs to the system. Outputs based on every combination of input were pre-calculated as expected results to assist with the identification of defects.

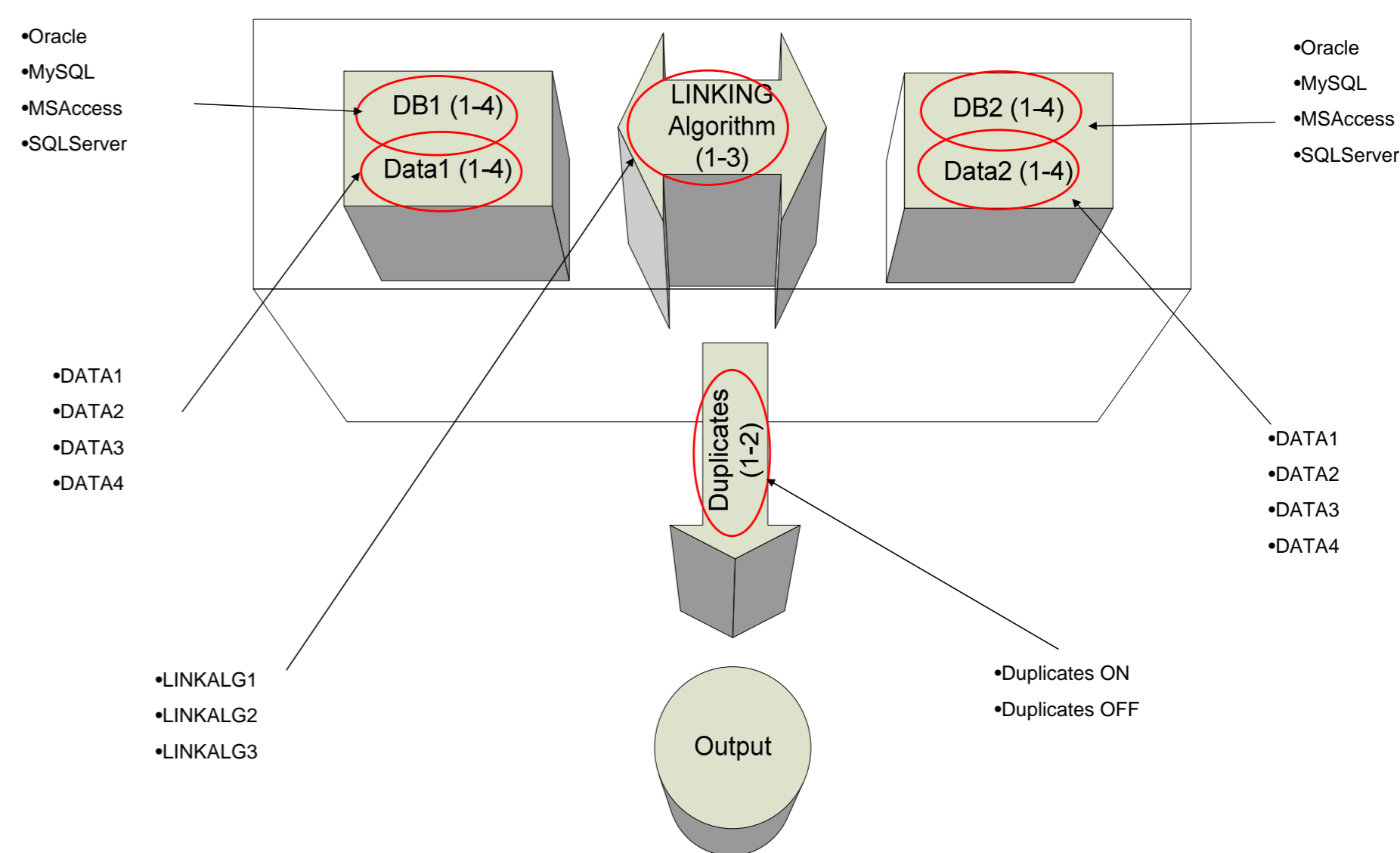


Figure 1: Analysis of the SUT (System Under Test) demonstrates six input components, each of the inputs taking several possible values.

Combo-Test Tool

Combo-Test was implemented as a command-line tool that generates test suites with a minimal set of test cases controllable by the SUT inputs (components and their values) and a strength input (the number of combinations of components that are to be exercised). The tool was developed using Java and utilises a JDBC connection to a pre-installed database. Combo-Test uses a thresholding technique to repeatedly parse through an exhaustive table finding matches against an array of n -sets. Several local minima are discovered for each execution of the tool, the one with the least number of tests is the one chosen.

Table 1: Combo-Test output defect rate. The varying suite strengths were run and the outputted suites were executed against the SUT. Redundant test cases indicates the tests that when run would trigger a defect that had already been detected, hence were of no value to the test suite.

Suite Strength	ComboTest Output	Defects Detected	Redundant Test Cases
1	4	0	N/A
2	20	5	1
3	64	5	15
4	281	6	83
5	768	6	237
6	1536	6	498

Conclusion

The experimental results showed that executing test suites generated by the ComboTest tool provided a comparable defect hit rate as running the exhaustive suite, but with a fraction of the execution effort.

The tool is now used as part of standard black-box system testing at the AEHRC, expanding the suite strength to best meet the time constraints put on the project.



References

- Mats Grindal et al 'Combination Testing Strategies: A Survey, ASWEC 2004.
- Mats Grindal et al 'An Evaluation of Combination Strategies for Test Case Selection', ASWEC 2006.
- Bertrand Meyer, 'Seven Principles of Software Testing'. IEEE Computer Aug. 2008



Further information

contact: Chris Stanbridge
phone: (07) 3253 3649
email: christopher.stanbridge@csiro.au
web: www.ict.csiro.au or www.aehrc.net
www.csiro.au