Automated Test Case Selection Based on a Similarity Function

Emanuela G. Cartaxo, Francisco G. O. Neto, Patrícia D. L. Machado

{emanuela,netojin,patricia}@dsc.ufcg.edu.br

Abstract:
A strategy for automatic test case selection based on the use of a similarity function is presented. Test case selection is a crucial activity to model-based testing since the number of automatically generated test cases is usually enormous and possibly unfeasible. Also, a considerable number of test cases are redundant, that is, they exercise similar features of the application and/or are capable of uncovering a similar set of faults. The strategy is aimed at selecting the less similar test cases while providing the best possible coverage of the functional model from which test cases are generated.

1 Introduction

Testing activity is crucial to the success of a software project, but this activity often requires about 50% of software development resources [Bei90]. Due to fierce dispute in applications market, such as the mobile phone one, customers are demanding cost reductions in their maintenance contracts [DJK+99]. Then, companies have been investing in research for approaches to decrease costs while keeping or increasing applications quality.

In this context, model-based testing appears as a promising approach to control software quality as well as to reduce the inherent costs of a test process, since test cases can be generated from the software specification, concomitantly to its development.

Functional testing requires full coverage, however, due to the costs of test execution (normally manual test execution), we might need to reduce the size of the test suite. Research has been continuously conducted towards strategies of test case selection in order to reduce the number of test cases [TB02]. However, in practice, the selection of which tests must be executed is generally a manual process that is error prone, and without sufficient guaranties that the system will be properly tested.

In this paper, we propose an automatic strategy for test case selection that is based on the use of a similarity function. The idea is that, whenever full coverage on model-based test case generation is unfeasible and/or test cases are mostly redundant, a similarity function can be automatically applied to discard similar test cases, keeping the most representative ones that still provides an adequate coverage of the functional model. We consider Labeled Transition Systems (LTSs) as the model from which test cases can be obtained. The reason is that LTSs are the underlying semantics of a number of formalisms and they can be easily obtained from functional specifications by using translation tools. Given an LTS and a

path coverage percentage as goal, the strategy reduces the size of the general test suite (according to the percentage), assuring that the remaining test cases are the less similar and covers the LTS transitions as much as possible.

This paper is structured as follows. In Section 2, the proposed test case selection strategy based on similarity is presented. Also, experiments are used in order to illustrate its application and obtained results are compared with random selection (Section 3). Section 4 presents the related works. Conclusions about this work are presented in Section 5. Finally, the acknowledgments are showed in Section 6.

2 Similarity-Based Test Case Selection

In this Section, we introduce the Similarity-Based Test Case Selection. This strategy reduces the number of redundant test cases according to a degree of similarity between them. This is inspired by the work of Jin-Cherng Lin and his colleagues [LY01].

For this selection, the tester must pass an intended path coverage percentage and an LTS behavioral model. LTS provides a global, monolithic description of the set of all possible behaviors of the system; a path on the LTS can be taken as a test sequence. Therefore, LTSs are highly testable models. An LTS [dVT00] is a 4-tuple $S = (Q, A, T, q_0)$, where Q is a finite, nonempty set of states; A is a finite, nonempty set of labels (denoting actions); T, the transition relation, is a subset of $Q \times A \times Q$ and q_0 is the initial state.

Here, we are considering that each LTS path (initial state until final or visited state) is a test case. In order to do this, we identify all paths from the LTS. A path can be obtained, using the Depth First Search method (DFS), by traversing an LTS starting from the initial state.

To reduce the number of test cases, we use a similarity criterion. Based on the similarity degree between each pair of test cases, the test cases to be eliminated are those that have the biggest similarity degree: one of them is eliminated. The choice is made by observing the test case that has the smallest path, i.e, the smallest number of functionalities.

From the LTS behavioral model, we assemble the similarity paths matrix. This matrix is:

- $n \times n$, where n is the number of paths and each n represents one path;
- $a_{ij} = SimilarityFunction(i, j)$; This function is calculated by observing the number of identical transitions (nit) - the "from" and "to" states and transition label are the same - and the average between paths length ($avg(|i|, |j|)$).

$$SimilarityFunction(i, j) = nit/(avg(|i|, |j|))$$

The number of identical transitions, between two test cases, discloses how much a case is similar to another one. Then, we divide that number per average between paths length to balance the similarities.

Figure 1 presents an LTS behavioral model (Figure 1(x)) and the four paths (Figure 1(a), (b), (c) and (d)). The similarity matrix is presented in Figure 2(x), the paths length in

Figure 2(y).

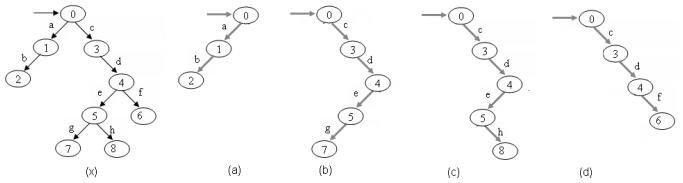

Figure 1: Possible paths from an LTS model.

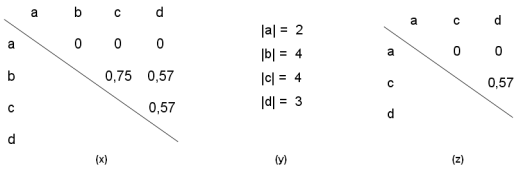

Figure 2: (x)Similarity Matrix, (y)Paths length, (z) Similarity Matrix after elimination.

Considering that the desired path coverage percentage is 50%, we must eliminate two test cases (out of four). Observing the matrix in Figure 2(x), we have that "b" and "c" have the biggest similarity, this way we have to eliminate one of them. As they have the same length (the paths length can be seen in Figure 2(y)), then we make a random choice, for example "b". In Figure 2(z), see the matrix after elimination of "b".

Observing the matrix in Figure 2(z), we have that "c" and "d" have now the biggest similarity. As $|c| > |d|$, then "d" is eliminated.

Note that the eliminated test cases are the most similar: "b" is very similar to "c", and "c" is very similar to "d". Of course, for this simple example, it is clearly possible to execute all test cases, but in an actual setting (with costs and time restrictions), 100% coverage may be unfeasible.

3 Evaluating the Similarity-Based Test Case Selection Strategy

Subjective impression from the use of this strategy is that it selects the most different test cases. In order to assess the validity of this impression, we compared our strategy with a random selection. Rather than deterministic choice, random selection has proven to be more effective when coverage and diversified choices are of concern [LT96]. For this, we have chosen three different reactive applications in the same application domain and we have specified their LTS behavioral models.

Our target is to measure the percentage of transitions coverage in the two strategies. For each application, we applied 100 times our strategy and the random strategy, considering that we have budget for executing only 50% of test cases for each application.

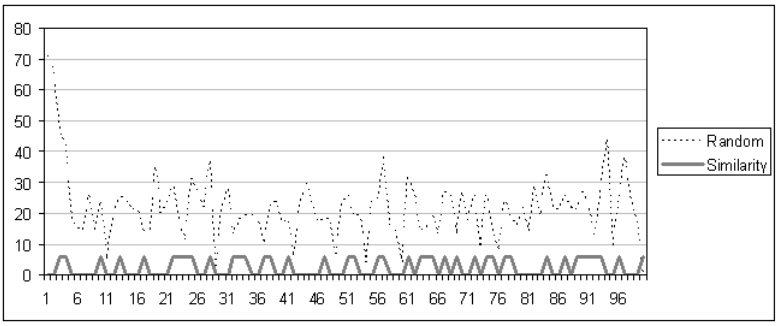

Figure 3: Application 1

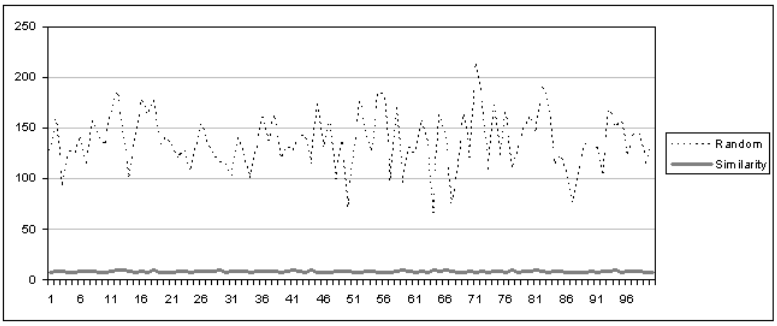

Figure 4: Application 2

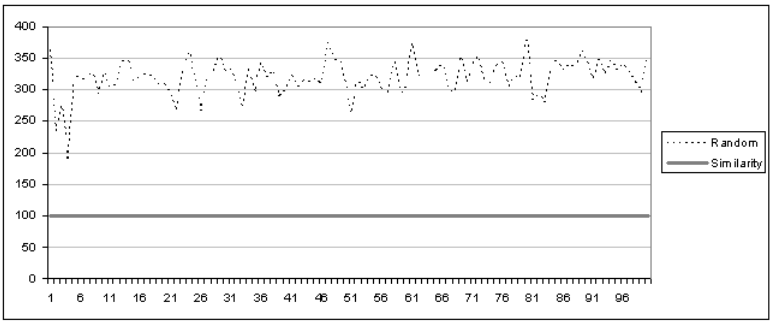

Figure 5: Application 3

The results are presented in Figures 3, 4 and 5. For each strategy and each of the 100 experiments (x-axis or abscissa), the number of excluded transitions (y-axis or ordinate) is presented.

Observing each experiment, we can see that the reduction of 50% of test cases with similarity strategy, in some cases, does not significantly reduce the transitions coverage. Ap-

plications 1 and 3 do clearly present redundant test cases, whereas Application 2 has a very few similar test cases. This is reflected in the final coverage obtained as expected: the strategy indeed eliminates redundancy as far as transition coverage is concerned.

Furthermore, the similarity strategy guaranties a more effective coverage than the random one. In the extreme cases, exemplified by Application 2, the similarity strategy is also more effective even though the percentage of eliminated transitions can be considered to be high for requirements. In the worst case, where there is no similarity at all between test cases, the transition coverage tends to be the same in both cases, since the similarity strategy applies a random choice when the level of similarity and the paths length is the same.

4 Related Works

This work is concerned with functional test case selection with specification coverage as the main goal. Therefore, code coverage based strategies are out of the scope of this paper since their goals are different from ours [MS01].

Test Generation with Verification technology (TGV) tool [JJ04] - a conformance test generator. This tool selects test cases from model. The test cases are selected from a test purpose, that is a specific objective that a tester would like to test, and can be seen as a specification of a test case. Even though a test purpose target the test at a particular functionality, reducing the final test suite size, the result can still be a huge exhaustive test suite that can be refined by using our strategy.

The Cow Suite tools derive test cases from UML sequence diagrams and use case diagrams [BBM02]. Their test generation algorithm is exhaustive. For each diagram a weight function indicating the functional importance is attributed. This way the test case selection strategy chooses the most important set of test cases.

SPACES [BLM$^+$07] is a tool for functional component testing. In this tool, weights are associated to the model's transitions. According to the weights the most important set of test cases are selected.

In [HGS93] a methodology for controlling the size of a test suite is presented. The input is a Requirements Traceability Matrix. This methodology guarantees 100% requirements coverage. This way the test case selection strategy chooses the minimal set of test cases disregarding the model and thus the transitions coverage.

5 Concluding Remarks

An automatic strategy to test case selection was presented. The strategy is based on similarity between test cases. The main goal of this strategy is, by observing the similarity between test cases, to minimize redundancy and assure an adequate transition coverage. was experimentally evaluated by three applications. The obtained results were compared

with random choice, and they showed that the similarity strategy guaranties a more effective coverage than the random one. As further work, other selection techniques need to be explored specially to handle requirements coverage. The strategy is currently supported by a tool.

6 Acknowledgments

This work has been developed in the context of a research cooperation between Motorola Inc., CIn-UFPE/Brazil and UFCG/Brazil. We thank the entire group for all the support, criticisms and suggestions throughout the development of this research. This work is also supported by FAPESQ/CNPq/Brazil.

References

[BBM02] F. Basanieri, A. Bertolino, and E. Marchetti. The Cow_Suite Approach to Planning and Deriving Test Suites in UML Projects. In *UML 2002 - The Unified Modeling Language. Model Engineering, Languages, Concepts, and Tools.*, LNCS. Springer, 2002.

[Bei90] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, second edition, 1990.

[BLM+07] D. L. Barbosa, H. S. Lima, P. D. L. Machado, J. C. A. Figueiredo, M. A. Juca, and W. L. Andrade. Automating Functional Testing of Components from UML Specifications. *Int. Journal of Software Eng. and Knowledge Engineering*, 2007.

[DJK+99] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton, and B. M. Horowitz. Model-Based Testing in Practice. In *International Conference on Software Engineering*, pages 285–294, 1999.

[dVT00] R. G. de Vries and J. Tretmans. On-the-fly conformance testing using SPIN. 2(4):382–393, March 2000.

[HGS93] M. J. Harrold, R. Gupta, and M. L. Soffa. A methodology for controlling the size of a test suite. *ACM Transactions on Software Engineering and Methodology*, 2(3), 1993.

[JJ04] C. Jard and T. Jéron. TGV: theory, principles and algorithms, A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems. *Software Tools for Technology Transfer (STTT)*, 6, 2004.

[LT96] R. C. Linger and C. J. Trammell. Cleanroom Software Engineering Reference Model. Technical report, CMU/SEI-96-TR-022, 1996.

[LY01] J. C. Lin and P. L. Yeh. Automatic test data generation for path testing using GAs. *Inf. Sci.*, 2001.

[MS01] J. D. McGregor and D. A. Sykes. *A Practical Guide to Testing Object-Oriented Software*. Addison-Wesley, 2001.

[TB02] J. Tretmans and E. Brinksma. Côte de Resyste – Automated Model Based Testing. In M. Schweizer, editor, *Progress 2002 – 3^{rd} Workshop on Embedded Systems*. STW Technology Foundation, 2002.