Research Journal of Applied Sciences, Engineering and Technology 7(14): 2981-2986, 2014
DOI:10.19026/rjaset.7.630
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: October 26, 2013 Accepted: November 12, 2013 Published: April 12, 2014

Research Article

Application of Search Algorithms for Model Based Regression Testing

Sidra Noureen and Sohail Asghar
University Institute of Information Technology, PMAS (AAUR), Rawalpindi, Pakistan

Abstract: UML models have gained their significance as reported in the literature. The use of a model to describe the behavior of a system is a proven and major advantage to test. With the help of Model Based Testing (MBT), it is possible to automatically generate test cases. When MBT is applied on large industrial systems, there is problem to sampling the test cases from the suit of entire test because it is difficult to execute the huge number of test cases being generated. The motivation of this study is to design a multi objective genetic algorithm based test case selection technique which can select the most appropriate subset of test cases. NSGA (Non-dominated Sorting Genetic Algorithm) is used as an optimization algorithm and its fitness function is improved for selecting test cases from the dataset. It is concluded that there is a room to improve the performance of NSGA algorithm by means of tailoring its respective fitness function.

Keywords: Metaheuristic, MBT, regression testing, test case selection

INTRODUCTION

Model based testing is getting more popular because of the fact that model emulates the entire functionality of the system at a glance as compared to a code. The most extensively used models for testing purpose are UML models (El-Far and Whittaker, 2001). El-Far (2001) reported that MBT is preferable because of the certain benefits it can provide. With the help of MBT, any changes which are introduced in the system can be easily accommodated while requiring less cost and effort. In contrast to this when a change is introduced in the code, several changes are needed to be carried out along with that particular change, as one part of the code is connected to some other part of the code. This eventually increases the cost. Previously code base techniques were employed for testing purpose, but those techniques had the problem that testing process started quite late, after the coding phase, but in MBT the testing process starts earlier which helps in early bug identification (Utting and Legeard, 2006). Recently search algorithms have been used with MBT. These algorithms are being applied on UML models. In order to generate optimal solutions to complex problems, Meta-Heuristic Search (MHS) algorithms are mostly employed.

MHS and local search algorithms are two categories of search algorithms. Local search algorithms include hill climbing and tabu search, while meta-heuristic search algorithms include simulated annealing and genetic algorithm. Many of the search algorithms such as simulated annealing and hill climbing have been used earlier (Ali *et al*., 2008). However, these algorithms get trapped in the local optimum problem (Coley, 1999). Meta-heuristic search algorithms are used widely nowadays as they do not get trapped in local optima (Ali *et al*., 2008). Among search algorithms, genetic algorithms are widely used as they can search the solution as well as evolve the new generation. Search algorithms have been applied to different models. In Hemmati *et al*. (2011) works test cases are selected which are similar to each other and genetic algorithm is used. This study is an enhancement of the technique carried out by Hemmati *et al*. (2011) with the help of MBT; we can automatically generate test cases. But the distinction of this study is that it aims to design a multi objective genetic algorithm based test case selection technique to select a small set of test cases.

Multi objective optimization techniques were being used since 1951. However, since 1970, such optimization approaches started getting more and more popularity till their practical and useful application was realized in 1967. The use of evolutionary algorithms for optimization of multi objective problems was first introduced by Rosenberg in 1960's. Multi objective optimization algorithms are categorized as: Aggregating approaches, VEGA, Lexicographic ordering, the ε-constraint method, Target vector approaches. All of these are non Pareto based approaches. These approaches have a limitation that they work with few objectives. Hence keeping in view of this limitation, Pareto based approaches were introduced which include: Pure Pareto ranking, MOGA, NSGA etc., (Coello, 2001).

Corresponding Author: Sidra Noureen, University Institute of Information Technology, PMAS (AAUR), Rawalpindi, Pakistan

In this study NSGA is implemented. Dias and De Vasconcelos (2002) defined multi objective problems in mathematical form as:

$$y = f(x) = \{f_1(x), f_2(x), \dots f_m(x)\}$$

Subject: to $g(x) = \{g_1(x), g_2(x), \dots g_j(x)\} \leq 0$

where, $h(x) = \{h_1(x), h_2(x), \dots h_k(x)\} = 0$

$$x = (x_1, x_2, \dots x_n) \in X$$
$$y = (y_1, y_2, \dots y_m) \in Y$$

Here x is decision vector. The solutions of such multi objective problems are expressed in terms of non-dominated individuals. We say that x^1 is said to be dominated than x^2 if none of the values of x^2 is less than x^1 and at least one value of x^2 is strictly greater than x^1 (Srinivas and Deb, 1994).

LITERATURE REVIEW

The techniques which were used in the past such as linear programming were static. But in software engineering such static algorithms are not quite useful, because in this filed the problems have such objectives that cannot be handled by linear techniques. SBST can handle such problems in a much better way because it involves fitness functions which are quite complex. The benefits of search based testing have been counted by scientists. With the help of experiments, it is found to be a wrathful and valuable area in the domain of software engineering.

Researchers have applied search algorithms on UML models. Generation of test data has been arisen as a major problem within model based testing. Ali *et al*. (2007) provide an empirical investigation of search based test case generation. They presented the results of a systematic, comprehensive review. They emphasized that the genetic algorithms perform better than random search in terms of structural coverage.

A large number of test cases are produced during the testing process and it is not viable to run all of them as it takes lots and lots of time. Hence selection of test cases, prioritization and minimization of test cases is also an essential problem in testing phase.

According to Hemmati *et al*. (2011) test cases are selected on similarity basis. The Author has used state machine diagram for test selection purpose and genetic algorithm is applied. The results point out that the approach has reduced cost of detecting faults up to 73% in one second of time. But, the results of the approach rely on one industrial case study; it should be replicated as many times as possible.

The limitation of GA for not embracing multiple objectives led to the need towards such a technique that could work with more than one objective which are opposing each other and takes the search process towards a better solution. Researchers started working on multi-objective types of Genetic Algorithm (GA).

Dias and De Vasconcelos (2002) have used NSGA (Non-Dominated Sorting Genetic Algorithm) to resolve the problems of optimization. It is described that it is not always necessary that there is only one solution to more than one objective. Hence some multi objective approach is required and NSGA is one of the multi objective algorithms. The Author has compared the results of NSGA with some other algorithms and has investigated the results. It was found that NSGA is better as compared to other multi objective genetic algorithms. The results have been shown with the help of the experiment, which is why results are reliable.

Iqbal *et al*. (2012) has proposed a methodology for testing real time systems. Two algorithms GA and (1+1) EA are employed and their ability to identify faults is detected. The approach when applied on an industrial case study shows consistent results. Author suggested combining (1+1) EA and random testing to achieve better results.

This study consists of three sections. The first section gives an introduction to the topic and related work in the field. The second section describes the proposed work and finally the last section describes the results and discussions.

RESEARCH METHODOLOGY

In the previous section, evolutionary and search based algorithms are discussed for model based testing. In proposed work, UML's state chart diagram and Multi-Objective Genetic Algorithm (MOGA) is used. This study is partial extension of Hemmati *et al*. (2011) work and the difference is that they used steady state GA; on the contrary Non-Dominated Sorting GA (NSGA) which is a multi-objective technique of GA is used in this study. GA makes the test case selection process easier (Deb *et al*., 2002). If fitness function is made with care, then search process can lead to the best possible solution. GA is a metaheuristic technique which works well for the complex problems. Selection of test cases is a complex problem and requires optimal solution; hence GA can be used for this problem. GA involves a function which selects the fit individuals from the population. This function is the fitness function. In our case GA selects test cases. Figure 1 depicts the flow of our work.

The similarity function which was defined by Hemmati *et al*. (2011) is used in this study which has been formulated as:

$$Sim\,Msr(s_n) = \sum_{tp_i, tp_j \in s_n \wedge i > j} SimFunc(tp_i, tp_j)$$

where, $SimFunc\ (tp_i, tp_j)$is the similarity between two test paths (tp_i, tp_j) which is calculated as:

- Identify the two triggers which are similar in two test paths.
- Find out the total transitions in the test path.

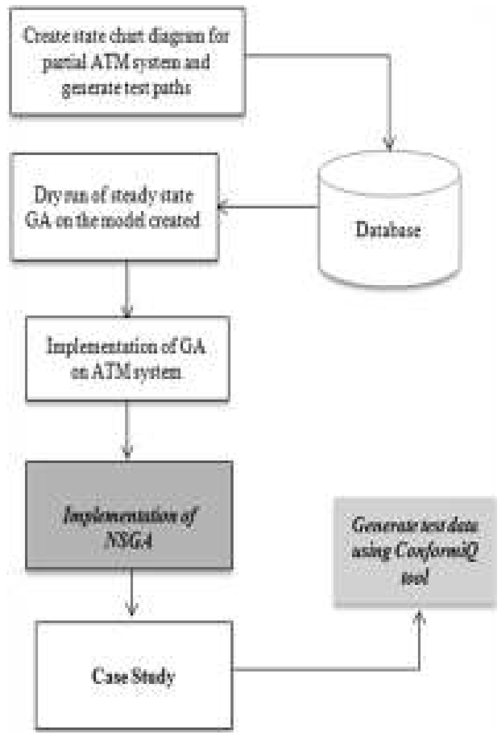
Create state chart diagram for partial ATM system and generate test paths
Database
Dry run of steady state GA on the model created
Implementation of GA on ATM system
Implementation of NSGA
Generate test data using ConformiQ tool
Case Study

Fig. 1: Flow of work (Implementation of NSGA and automatic generation of test data)

- Find the average number of transitions in the test path.
- Divide the number of similar triggers on the average number of transitions.

This similarity function which was named as Tb uses similar triggers. Tb was defined as:

> *"Tb(tpi, tpj) = Number of identical triggers in tpi and tpj divided by the average length (number of transitions in the test path) of tpi and tpj".*

According to Hemmati *et al*. (2011), in a state chart diagram, similar events are more likely to occur so in such cases Tb is quite useful. Tb selects one test path and leaves the other. This is applied on an ATM mode land the test data which can be generated is like: tp1 = (RC, "success", RP, "prs", CT, "tc", PT, "cf", EC, F), tp2 = (RC, "success", RP, "cp", EC, F) For this kind of test data, $Tb = (tp_1, tp_2) = 0.2$(one similar eent and total length 4).

The other fitness function which is used here and which was defined by Ma *et al*. (2005) is:

$$Fitness(test\ path) = {Cov\ (tp_i)}\Big/{Cost(tp_i)} \qquad (1)$$

where, Cov is "coverage". We have calculated coverage as follows:

$$Cov = \sum_{tp \in s_n} Size\ (tp_a, tp_b, \ldots tp_z) \qquad (2)$$

e.g., after crossover the obtained child are like: {tp1, tp2, tp5} and if supposed that size of tp1 is 4, size of tp2 is 5 and size of tp5 is 2. Then total coverage will be = 4+5+2 = 11.

On the other hand cost of test path is total size of generation. This can be formulated as:

$$Cost = \lim_{i=1 \to n}(tp_i) \qquad (3)$$

e.g., {tp1, tp2, tp5}. So Cost will be 3 because total elements in this generation are 3.

In this study the selection technique used involves following steps as depicted in Fig. 2 (Dias and De Vasconcelos, 2002).

In the first step of implementing this algorithm, the population is initialized. The input is given in the form of test paths (string type) like tp = {1, a, 2, b, 3}. In this the alphabet represents an event and the numbers show states. Such kinds of test paths make a population, where one test path is considered a gene.

The probability of individuals is calculated and individuals are assigned ranks on the basis of that probability. The probability is calculated using the following formula:

$$P_{tp} = 1/n$$

The calculated probability is used later for separating non-dominated individuals.

Non-dominated individuals are then identified from the population. It is supposed that these individuals make the first front. After this a sharing method is applied. We modified the existing formula of sharing the fitness value. Our formula for sharing fitness between individuals is:

$$Sharing = 1\ if\ {d_{ij}}\Big/{\theta_{shared}}$$

$$when\ < \theta_{shared}$$
$$0\ otherwise$$

where,

d_{ij}	= The difference between similarity values of two individuals
θ_{shared}	= Maximum similarity between two individuals due to which they can be the components of a niche

Individuals in the first front are then ignored and then remaining population is sorted according to the same procedure.

These steps are repeated until all the population is sorted. We define the genetic operators too which include selection, crossover and mutation.

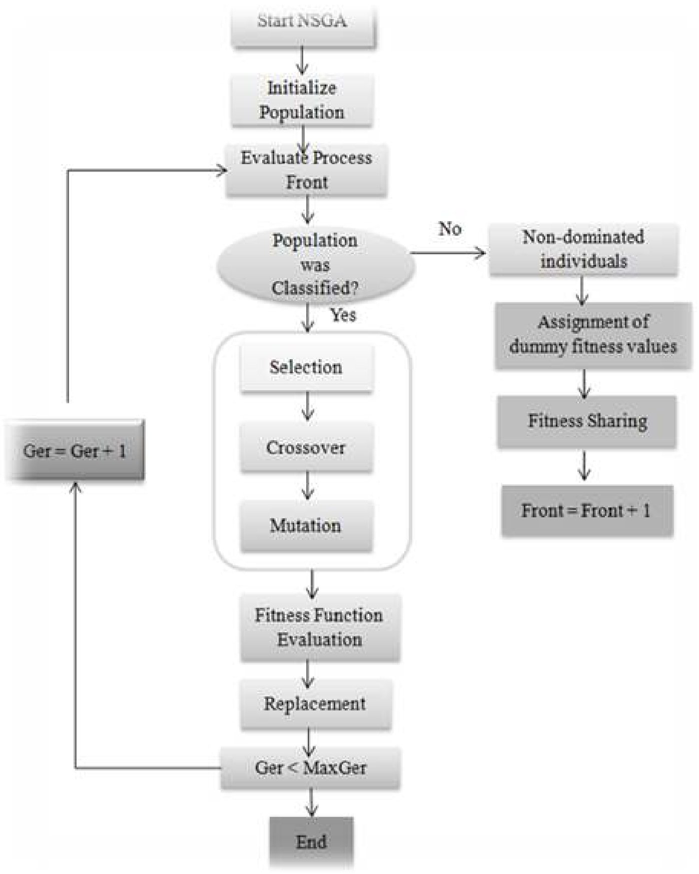

Fig. 2: Process of non-dominated sorting genetic algorithm

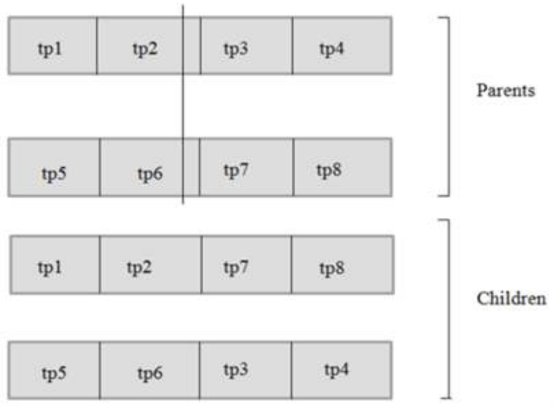

Fig. 3: Crossover between test paths

Once the non-dominated individuals are sorted, then selection is performed. Ranks are assigned. In rank selection we calculate sum of the assigned ranks and then calculate the probability values on basis of which individuals are selected.

Then crossover is performed. The parts of individuals which are selected are crossed over to combine the characteristics of the both the individual. Single point crossover is used. This is shown as in Fig. 3.

Then mutation operator is applied to replace any test path in the children which has low similarity value, to be assumed tp7 has low similarity value and in population there exists a test path tp11 which has high similarity value then tp7 is replaced with tp11.

Now, the fitness function is evaluated. If the test paths in the children are fit, then parents are replaced with children, otherwise the process is repeated.

Automatic generation of test cases: It is possible to automatically generate test cases from UML models. Until now, the test cases were developed manually and then test scripts were written for the test cases (Sieving and Öman, 2010). But now the trend is towards the use of MBT. The tools which are used in industry for automated testing use MBT because MBT provides this benefit of automation. Different tools are used for different UML models. For this particular work Conformi Q Qtronic tool is selected which generates test cases from state chart diagram. The model is built in Conformi Q modeler. It automatically generates test cases from the state chart diagram. Conformi Q Qtronic generates test cases in different forms like in TTCN-, junit, html etc., we have generated them in junit as well as in html. Moreover the tool provides several advantages like:

- Test cases are generated efficiently
- Quality tests are generated
- The chance of errors in the design as well as tests is minimized
- When design is improved, the design document which is prepared in start is automatically improved and there are no errors in that.
- There is no mess of creating test manually and no bugs are found at the end

Afterwards the model is loaded in the Qtronic IDE. Further there is a computation server given by Conformi Q, which runs at the backend and creates the test cases. Conformi Q Qtronic works with Eclipse SDK plug-in. It automatically generates test cases from state chart Diagram.

RESULTS AND DISCUSSION

The preceding section is about the Non-dominated Sorting Genetic algorithm that how it works for the problem of test case selection. This chapter tells about the results of the Non-dominated Sorting Genetic Algorithm (NSGA). It is discussed how our results are different from the existing work done to solve the problem of selecting a subset of test paths from a large number of test dataset. The algorithm is implemented in Java language and tool used is Net Beans 7.0. The robotic system case study is used for extracting test data.

Case study: The robot system is a real time system. The robot performs four functions. It can turn to right, to left, it can move forward and backward. These are the states of the system i.e., right, left, forward and backward. A timer is also set for the system after which it changes its direction. Several events occur on the system and whenever any event occurs, the transition takes place and the state of the system changes. This case study generates almost thirty test paths. Among these test paths, whenever two test paths are found to be same on the basis of this criterion, the similarity function discards one test path and keeps the other one.

Results and discussion for steady state GA: Figure 4 depicts the fitness function values for the test paths in the child generations which are selected by steady state GA. X-axis represent the fitness values and Y-axis represents the test paths. Every time we run the Genetic Algorithm, different generations are selected because GA is an evolutionary algorithm. The test paths are initially chosen randomly.

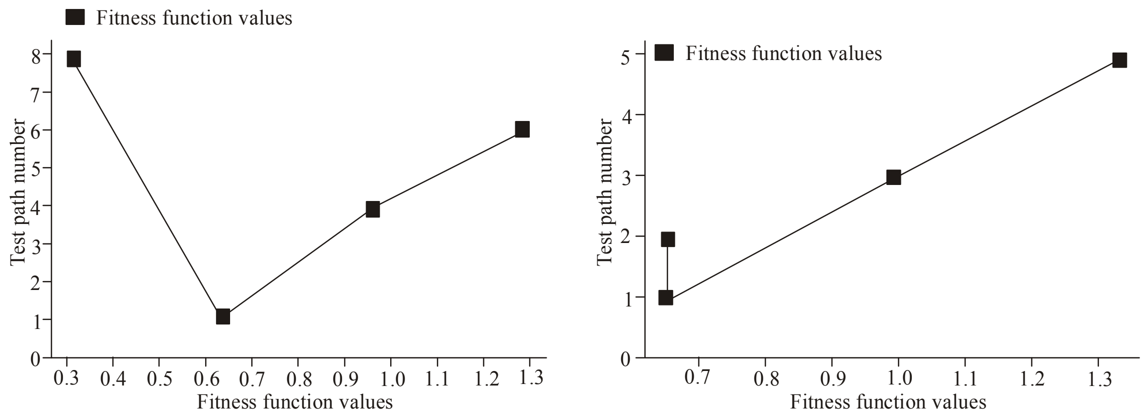

Fig. 4: Fitness function values for test paths selected by steady state GA

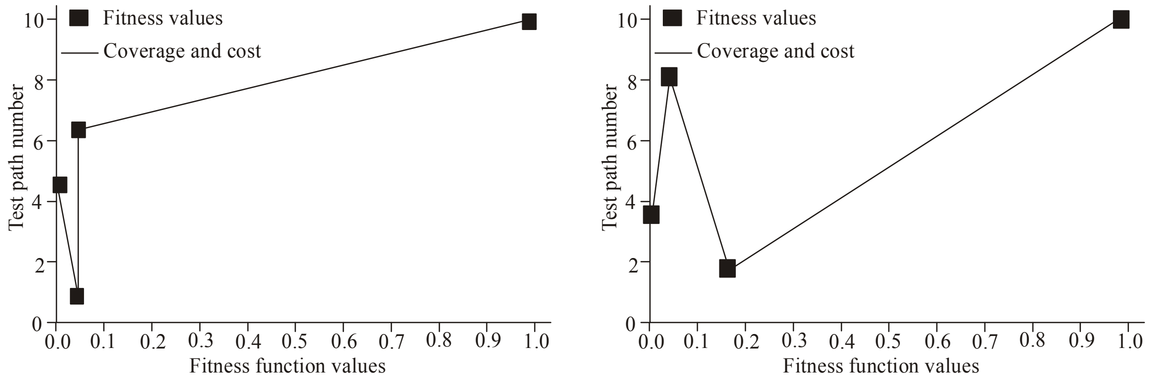

Fig. 5: Fitness function values for test paths selected by NSGA

Results and discussion for NSGA: Figure 5 shows the fitness function values of the test paths which are selected by Non-dominated sorting Genetic Algorithm. As shown by graphs of GA, here too X-axis shows fitness function values and on Y-axis the test paths are shown. NSGA also selects different generations on every run because of being an evolutionary algorithm.

CONCLUSION AND RECOMMENDATIONS

The results of the existing steady state GA and this approach (NSGA) show that two algorithms generate their own results according to the nature of the problem. We can conclude that as steady state GA works with a single objective while NSGA works with multiple objectives, so the results of the two approaches are different. The performance of the algorithm improves if the function that selects the test paths is improved. In the coming future, we need to implement the approach with other multi-objective algorithms and compare with that, further other evolutionary algorithms mentioned in the literature like ACO, PSO etc., are needed to be checked for their performance for the same problem.

ACKNOWLEDGMENT

The author gratefully acknowledges Babar Shabbir, Adia Khalid, Omer Bashir and all the other people who contributed and helped in this study.

REFERENCES

Ali, A., A. Nadeem, Z. Iqbal and M. Usman, 2007. Regression testing based on UML design models. Proceeding of the 13th IEEE International Symposium on Pacific Rim Dependable Computing, pp: 85-88.

Ali, S., L. Briand, H. Hemmati and R. Panesar-Walawege, 2008. A systematic review of the application and empirical investigation of search-based test case generation. IEEE T. Software Eng., 36(6): 742-762.

Coello, C.A.C.C., 2001. A Short Tutorial on Evolutionary Multiobjective Optimization. In: Zitzler, E. *et al.* (Eds.): EMO 2001. LNCS 1993, Springer-Verlag, Berlin, Heidelberg, pp: 21-40.

Coley, D.A., 1999. An Introduction to Genetic Algorithm for Scientists and Engineers. World Scientific Publishing Co. Pte. Ltd., Singapore.

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, 2002. A fast and elitist multi objective genetic algorithm: NSGA II. IEEE T. Evolut. Comput., 6(2): 182-197.

Dias, A.H.F. and J.A. De Vasconcelos, 2002. Multi objective genetic algorithms applied to solve optimization problems. IEEE T. Magn., 38(2): 1133-1136.

El-Far, I.K., 2001. Enjoying the perks of model-based testing. Proceeding of the Software Testing, Analysis and Review Conference (STARWEST 2001).

El-Far, I.K. and J.A. Whittaker, 2001. Model-based software testing. In: Marciniak, J.J. (Ed.), Encyclopedia of Software Engineering. Wiley, Chichester.

Hemmati, H., L. Briand, A. Arcuri and S. Ali, 2011. An enhanced test case selection approach for model-based testing: An industrial case study. Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp: 267-276.

Iqbal, M.Z., A. Arcuri and L. Briand, 2012. Empirical investigation of search algorithms for environment model-based testing of real-time embedded software. Proceedings of the 2012 International Symposium on Software Testing and Analysis (ISSTA 2012), pp: 199-209.

Ma, X., B. Sheng and C. Ye, 2005. Test-suite Reduction Using Genetic Algorithm. In: Cao, J., W. Nejdl and M. Xu (Eds.): APPT 2005. LNCS 3756, Springer-Verlag, Berlin, Heidelberg, pp: 253-262.

Sieving, R. and P. Öman, 2010. Pilot project for model based testing using conformiq qtronic. M.A. Thesis, Department of Computer Science and Electrical Engineering, Division of Media Technology, Lulea University of Technology, Arena.

Srinivas, N. and K. Deb, 1994. Multi objective optimization using non-dominated sorting enetic algorithm. J. Evolut. Comput., 2(3): 221-248.

Utting, M. and B. Legeard, 2006. Practical Model-based Testing: A Tools Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.