EuroSTAR’93, 25 - 28 October 1993, Edwardian Hotel, London, U.K.

Tool-Supported Test Case Design for Black-Box Testing by Means of the Classification-Tree Editor

Matthias Grochtmann
Klaus Grimm
Joachim Wegener
Daimler-Benz AG
Forschung und Technik
Alt-Moabit 91b
D-10559 Berlin, Germany
Tel: +49 30 39 982-229 / -226 / -232
Fax: +49 30 39 982-107

Abstract

The systematic test is an inevitable part of the verification and validation process for software. The most important prerequisite for a thorough software test is the design of relevant test cases, since they determine the kind and scope and hence the quality of the test. The graphical editor CTE (classification-tree editor) presented in this paper supports the systematic design of black-box test cases. The tool is based on the classification-tree method, an approach to partition testing which uses a descriptive tree-like notation and which is especially suited for automation. The tool has already been tried out successfully on actual examples in various divisions of the Daimler-Benz Group.

Prerequisite Key Words: none

Topic Descriptors: Test Automation, Testing Tool, Test Case Determination, Black-Box Testing, Functional Testing, Partition Testing

1. Introduction

The systematic test is an inevitable part of the verification and validation process for software. Testing is aimed at finding errors in the test object *and* giving confidence in its correct behaviour by executing the test object with selected input values.

The overall testing process can be structured into the following central test activities: During test case determination the input situations to be tested are defined. Concrete input values which fulfill the abstract test cases are determined during test data generation. For these test data the expected outputs are then predicted. The test object is run with the test data and thus the actual output values are produced. By comparing expected and actual values the test results are determined. Additionally, monitoring can give information on the behaviour of the test object during test execution.

The most important prerequisite for a thorough software test is the design of relevant test cases, since they determine the kind and scope of the test. The graphical editor CTE (classification-tree editor) presented in this paper supports the descriptive and systematic design of black-box test cases.

2. State of the Art

As experience shows, a tool support is extremely helpful in real-world test problems. Tools for white-box testing (i.e. testing based on the structure of the program itself) are widely used in practice. Typical examples are coverage analyzers for branch testing. Routine activities like regression testing are also widely automated. Surveys of testing tools can, for example, be found in DEMIL and GRAH.

However, there is a lack of tools for test case design using a black-box approach (i.e. testing based on the functional specification). This shortage is mainly caused by the lack of black-box testing methods which are well suited for automation and which are, at the same time, powerful and easy to use in practice. Therefore, the first step in the development of a black-box testing tool has to be the selection of an appropriate testing method.

3. The Classification-Tree Method

The tool CTE is based on the classification-tree method (GROCHT), a special approach to (black-box) partition testing. The classification-tree method is a self-development partly using and improving ideas from the category-partition method defined by Ostrand and Balcer (OSTR).

By means of the classification-tree method, the input domain of a test object is regarded under various aspects assessed as relevant for the test. For each aspect, disjoint and complete classifications are formed. Classes resulting from these classifications may be further classified – even recursively. Test cases are formed by combining classes of different classifications. The stepwise partition of the input domain by means of classifications is represented graphically in the form of a tree. This tree is subsequently used as the head of a combination table in which the test cases are marked. When using the classification-tree method, the most important source of information for the tester is the functional specification of the given test object. A major advantage of the classification-tree method is that it turns test case design into a process comprising several structured and systematized parts – making it easy to handle, understandable and also documentable.

The use of the classification-tree method will be explained using a simple example. The test object is a Computer Vision System which should determine the size of different objects (Figure 1). The possible inputs are various building blocks. Appropriate aspects in this particular case would be, for example, the size, colour and shape of a block (Figure 2).

The classification based on the aspect 'colour' leads, for example, to a partition of the input domain into red, green and blue blocks, the classification based on shape produces a partition into circular, triangular and square blocks. An additional aspect is introduced for the

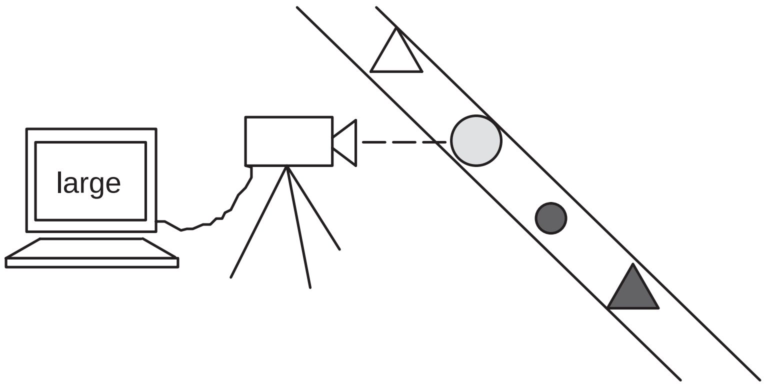

Figure 1: Computer Vision System

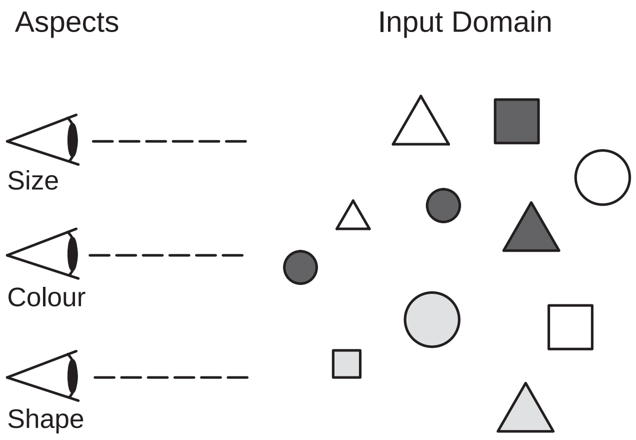

Figure 2: Aspects for Classification

triangle class: the shape of triangle. The various classifications and classes are noted as classification tree (Figure 3).

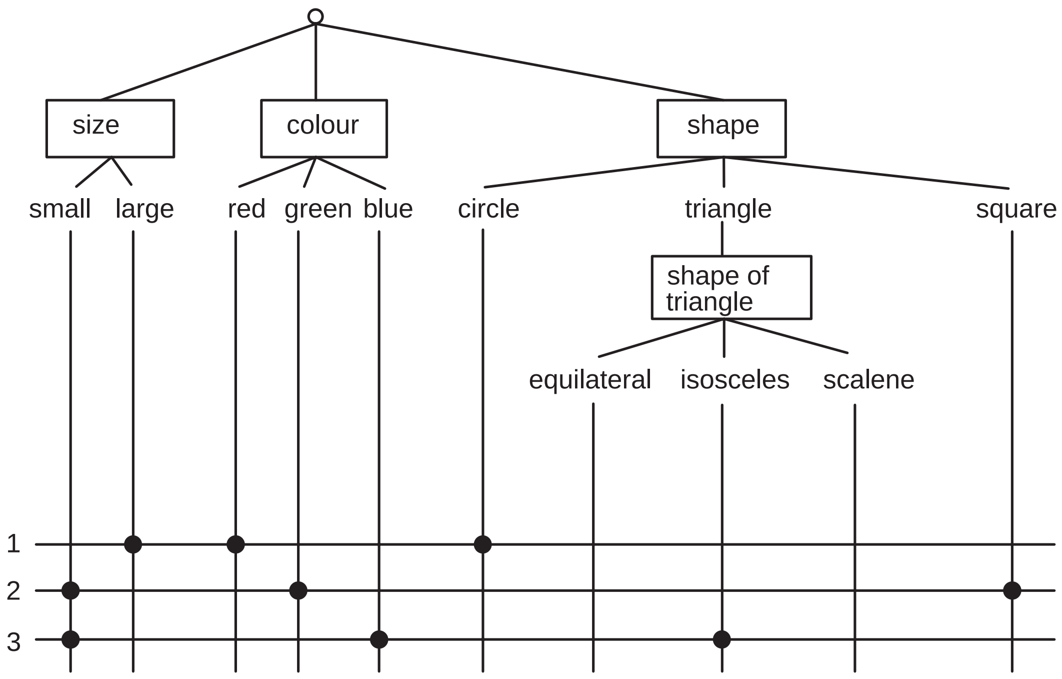

Figure 3: Classification Tree

In the combination table associated with the tree some possible test cases are marked as examples. Test case three, for instance, describes the test with a small blue isosceles triangle.

The classification-tree method is especially suited for automation since (a) it decomposes the test case design process into several steps which can be automated individually allowing the tool to appropriately guide the user and (b) it offers a graphical notation well suited for visualization in a modern graphical user interface.

4. The Classification-Tree Editor CTE

The two main phases of the classification-tree method – design of a classification tree and definition of test cases in the table – are both supported by the tool CTE. For each phase a suitable working area is provided. Thereby, CTE supports systematic and efficient test case determination for black-box testing.

The classification-tree editor CTE uses a separate window on the screen (Figure 4). In the upper part of the window there is a drawing area in which the user can build up a classification tree interactively (1). The lower part of the window depicts a corresponding table in which test cases can be marked interactively (2). Each test case is numbered (3). The menu bar (4) offers access to several pull-down menus which offer various commands e.g. for saving, editing and printing. The current state of the CTE is displayed in the status area (5). Pop-up menus are used to access element-specific commands in the working area (6).

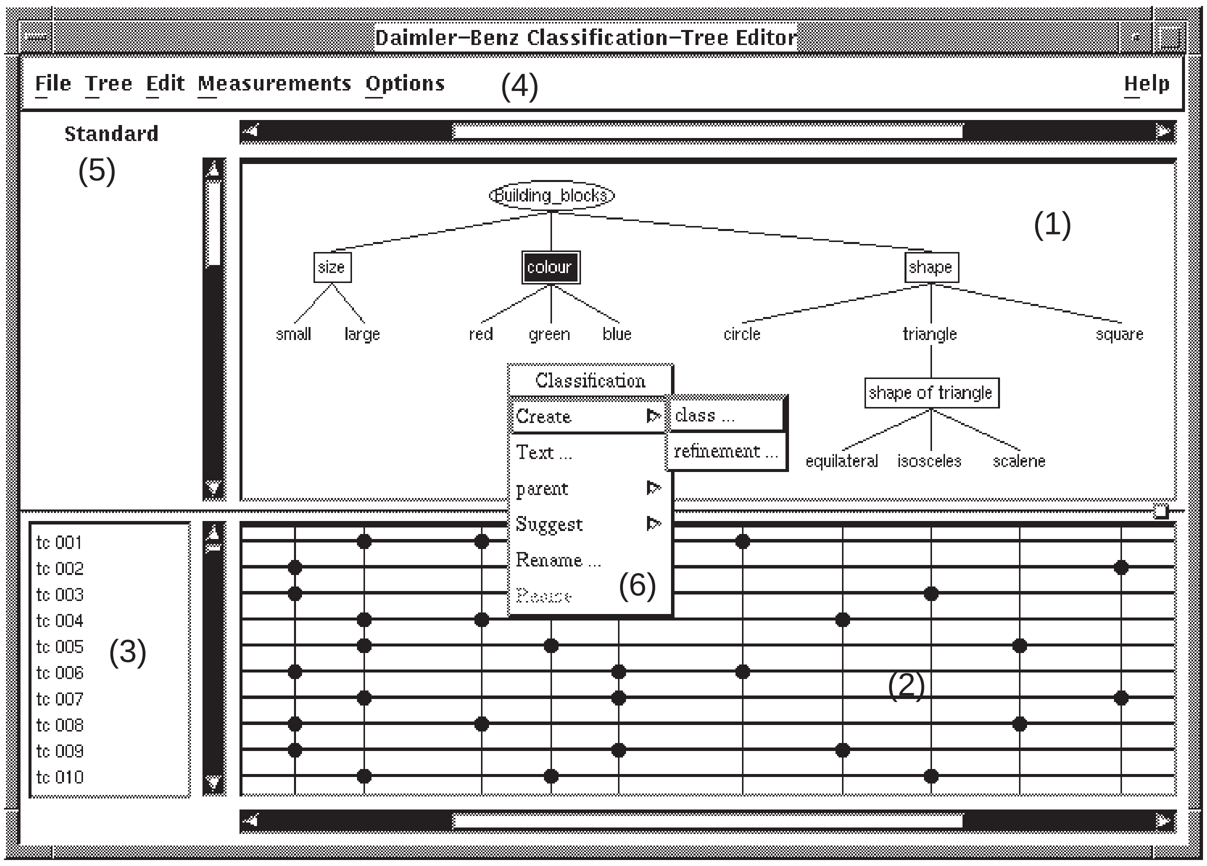

Figure 4: Classification-Tree Editor

To give the user optimal support editing is done syntax-directed and object-oriented. Several functions are performed automatically. This includes drawing of connections between tree elements, updating the combination table after changes in the tree and checking the syntactical consistency of table entries.

The CTE offers features which allow large-scale classification trees to be structured in order to support the test case design for large testing problems efficiently. This can be illustrated, for example, in Figure 5 where a screen dump of the CTE used for the test of a part of the CTE itself is shown. The test object is a procedure ‘is_line_covered_by_rectangle’ which should determine, whether a distinct line is covered by a given rectangle. This procedure is used in the CTE for determining the need of redrawing parts of the tree in case of layout changes.

The main window (in the background) shows that the input domain of the test object is distinguished for instance according to the existence and degree of coverage and according to the positions of the end points P1 and P2 of the line. Refinement symbols are used at various places, in order to make more detailed differentiations in separate windows. For example, the child window in the foreground shows that the case that P1 lays outside the rectangle is

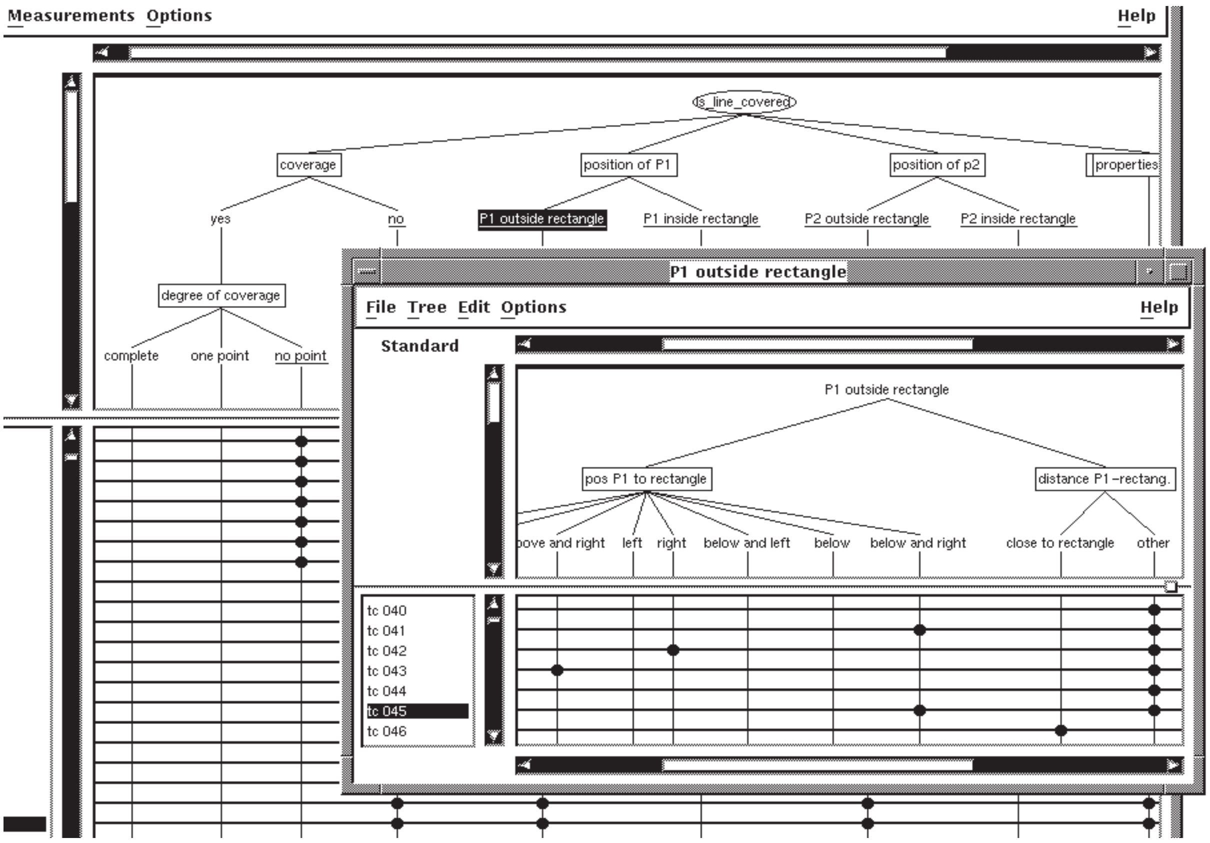

Figure 5: CTE used for large testing problem

further distinguished according to the position of P1 with respect to the rectangle and according to the distance of P1 from the rectangle. Other windows could be opened to show the complete tree and table. In this example the test case determination process led to 49 test cases and one error in the test object could be detected.

As test documentation plays an important role in systematic testing, the CTE offers suitable support for this activity. For example, the test case design can be documented easily by printing out the trees and tables. Furthermore, the tool can automatically generate text versions of the test cases, based on the test case definition in the table. For example, the text version of test case 45 of the example is given by:

Coverage: no
  Minimum distance line – rectangle: very small
Position of P1: P1 outside rectangle
  Position of P1 with respect to rectangle: below and right
  Distance of P1 from rectangle: other
Position of P2: P2 outside rectangle
  Position of P2 with respect to rectangle: left
  Distance of P2 from rectangle: other
Course of line: slanting
  Direction of line (P1 –> P2): bottom right –> top left
  Gradient of line: medium

These text versions serve on the one hand as documentation, on the other hand as a basis for the subsequent activities of software testing like the generation of concrete test data.

The CTE has been developed as an in-house tool on VAXstation under VMS and OSF/Motif in C. It is also available for Ultrix on DECstations. A version for SUN/Solaris is planned.

5. Practical Experience

The tool – in conjunction with the method – has already been tried out successfully on actual examples in various divisions of the Daimler-Benz Group. As a result, some divisions recently started to use the tool in larger projects.

Examples for such real-world applications are a control system for the airfield lighting of an international airport, an identification system for automatic mail sorting machines and an integrated ship management system.

During the trials, the test documentation generated by the CTE proved to be appropriate and useful. The fact that the CTE guides and supports testers but does not limit them was also positively judged by users.

Most important: A good error detection rate was observed. For example, in one module test for the identification system the number of test cases could be halved compared to an existing set of test cases and at the same time two errors were found. A detailed description of the results of the practical applications of the classification-tree method and the CTE can be found in GROCHT.

6. Conclusion

In the future, the experimental use of the CTE will focus on the test of large, distributed, parallel and real-time systems to receive more feedback on the applicability of the tool to these important areas of testing.

Since the CTE has been so well received in in-house practice, it is now planned to transform it into a commercial product. In this respect the trials have already provided some valuable information on ways of further improving the CTE, some of which have already been realized.

For example, additional editing functions like ‘copy/paste’ and ‘undo’ will enhance the usability of the CTE. An automatic generation of test cases in the table according to combination rules or even of complete classification trees from formal specifications is planned as well.

Furthermore, the CTE will be integrated within the overall computer-aided test system Tessy. Tessy is under development by Daimler-Benz Research in Berlin in cooperation with divisions of the Daimler-Benz Group. It will give testers suitable support not only for test case determination by means of the CTE but also for all other central activities of software testing.

References

DEMIL	DeMillo, R.A., McCracken, W.M., Martin, R.J., Passafiume, J.F., **Software Testing and Evaluation**, Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.
GRAH	Graham, D.R. (Ed.), **Computer-Aided Software Testing: The CAST Report**, Unicom Seminars Ltd., Middlesex, UK, 1991.
GROCHT	Grochtmann, M., Grimm, K., Classification Trees for Partition Testing, to appear in **Software Testing, Verification & Reliability**, Wiley.
OSTR	Ostrand, T., Balcer, M., The Category-Partition Method for Specifying and Generating Functional Tests. **Communications of the ACM**, Volume 31, Number 6, June 1988, pp. 676 - 686.