Quality Week 1995

Test Case Design Using Classification Trees and the Classification-Tree Editor CTE

Matthias Grochtmann
Joachim Wegener
Klaus Grimm

Daimler-Benz AG
Research and Technology
Alt-Moabit 96a
D-10559 Berlin, Germany
Tel: +49 30 39 982-229
Fax: +49 30 39 982-107
email: grochtm@dbresearch-berlin.de

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 classification-tree method and the graphical editor CTE (classification-tree editor) support the systematic design of black-box test cases. The classification-tree method is an approach to partition testing which uses a descriptive tree-like notation and which is especially suited for automation. Method and tool have already been tried out successfully on actual examples in various divisions of the Daimler-Benz Group. Since the CTE has been so well received in in-house practice, it is now transformed into a product version with a number of additional features found to be useful during the practical trials.

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 behavior 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 meet the test case conditions 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. The test results are determined by comparing expected and actual values. Additionally, monitoring can be used to obtain information about the behavior 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.

2. State of the Art

As experience shows, methods and tools are extremely helpful in real-world test problems (DeMillo et al., 1987; Graham, 1991). Methods and tools for white-box testing (i.e. testing based on the structure of the program itself) are widely used in practice. A typical white-box approach is branch testing which is supported by coverage analyzers.

However, there is a lack of methods and tools for test case design using a black-box approach (i.e. testing based on the functional specification). Thus the classification-tree method and the classification-tree editor were developed by Daimler-Benz Research to improve this situation.

3. The Classification-Tree Method

The classification-tree method (Grochtmann and Grimm, 1993) is a special approach to (black-box) partition testing partly using and improving ideas from the category-partition method defined by Ostrand and Balcer (1988).

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. The stepwise partition of the input domain by means of classifications is represented graphically in the form of a tree. Subsequently, test cases are formed by combining classes of different classifications. This is done by using the tree 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, the colour and the 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 the shape produces a partition into circular, triangular and square blocks. An additional aspect is introduced for the triangle class: the shape of triangle. The various classifications and classes are noted as classification tree (Figure 3). Some possible test cases are marked as examples in the combination table associated with the tree. 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.

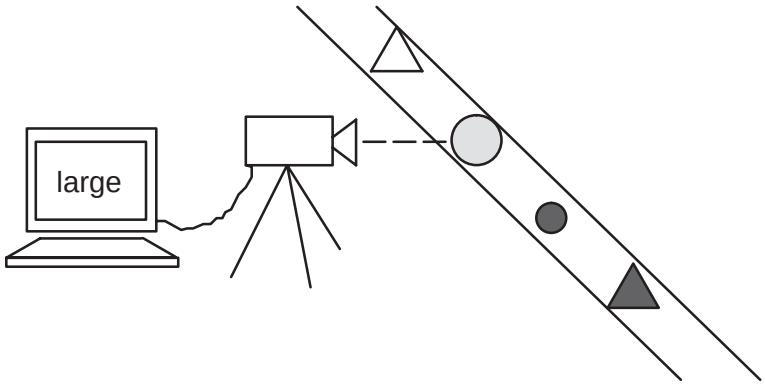

Figure 1: Computer Vision System

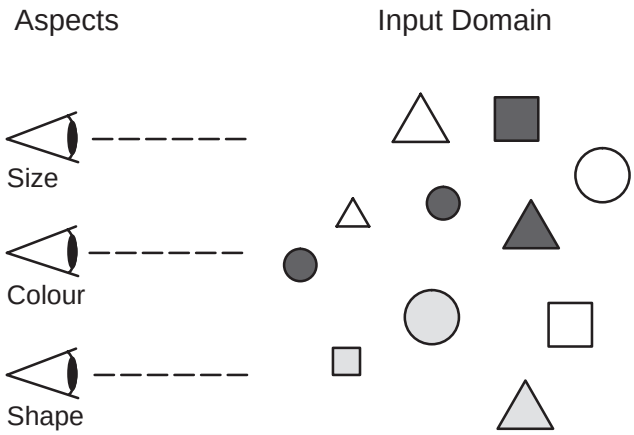

Figure 2: Aspects for Classification

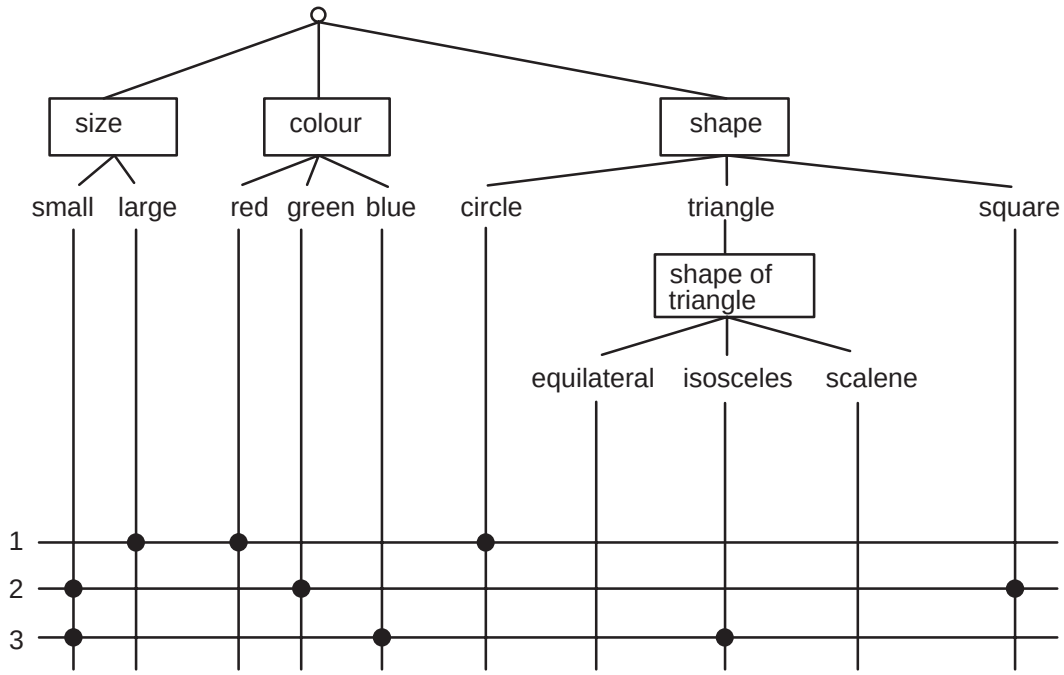

Figure 3: Classification Tree

4. The Classification-Tree Editor CTE

The classification-tree editor CTE is based on the classification-tree method and supports systematic and efficient test case determination for black-box testing (Grochtmann et al., 1993). 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. For each phase a suitable working area is provided.

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 row is numbered (3). The menu bar (4) offers access to several pull-down menus which provide various commands, e.g. for saving, editing and printing. The current working mode of the CTE is displayed in the status area (5). Pop-up menus are used to choose element-specific commands in the working area (6).

To give the user optimal support, editing is done in a syntax-directed and object-oriented way. Several functions are performed automatically. These include 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

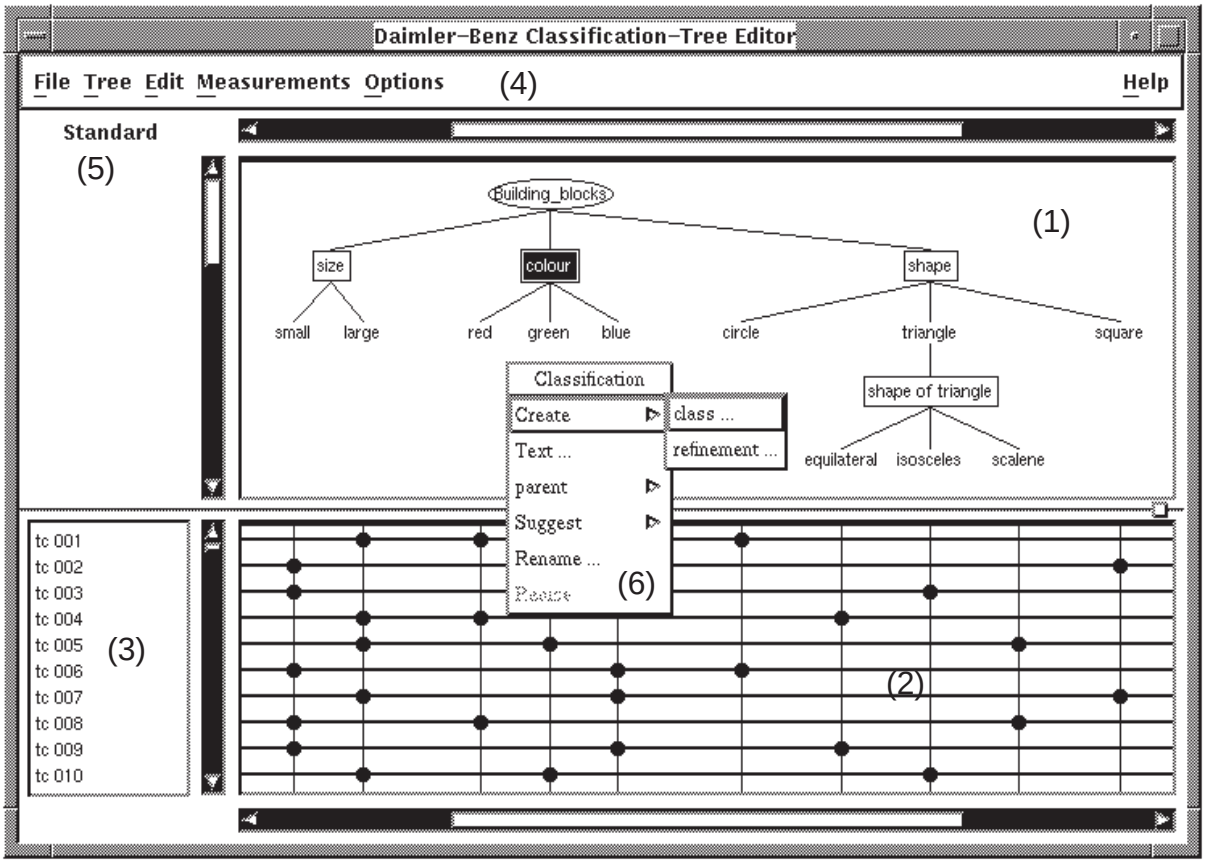

Figure 4: Classification-Tree Editor

is shown. The test object is a procedure ‘is_line_covered_by_rectangle’ which should determine whether or not a distinct line is covered by a given rectangle. This procedure is used in the CTE for determining the need for redrawing parts of the tree in case of window exposure events.

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 lies outside the rectangle is 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 can 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 shown as:

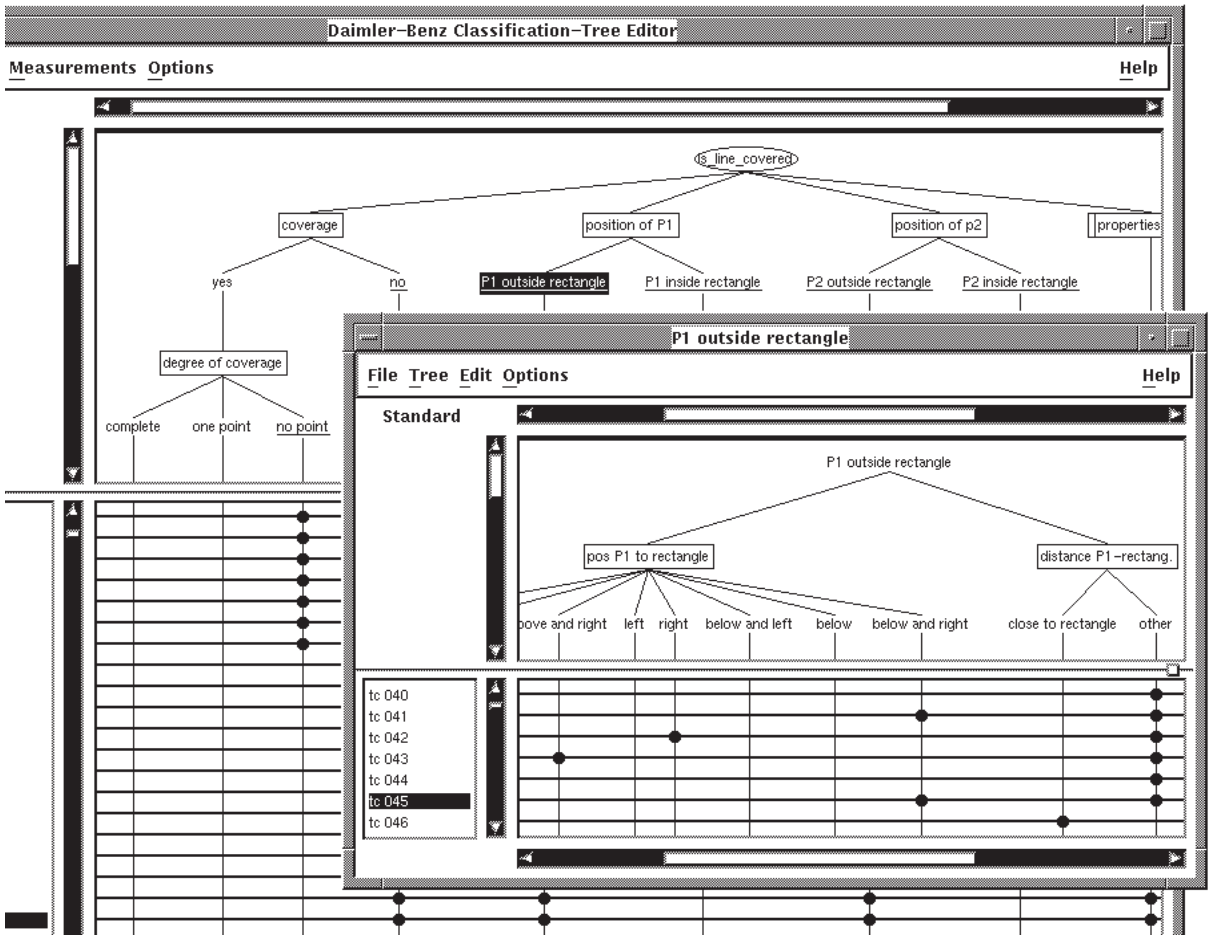

Figure 5: CTE Used for Large Testing Problem

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

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

The first version of the CTE was developed as an in-house tool on VAXstation under VMS and OSF/Motif written in C. It is also available for Ultrix, HP UX, SUN OS and Solaris.

5. Practical Experience

Until now 21 applications of method and tool have been performed. Most of them were carried out within actual projects of various divisions of the Daimler-Benz Group. 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. The results observed were promising and successful, and consequently, some divisions have now started to use the method and tool on a regular basis in larger projects. In some cases the mandatory use of the classification-tree method and the CTE was made part of division-specific development standards.

The most important feature of the classification-tree method was observed to be a good error detection rate. For example, in one module test for the identification system the number of test cases could be halved compared to a previously used set of test cases due to existing redundancy and at the same time new errors were found. A detailed description of some results of the practical applications of the classification-tree method and the CTE can be found in (Grochtmann and Grimm, 1993) as well as (Grimm and Grochtmann, 1994).

A first approach to a testing strategy aimed primarily at interactive business software giving some rules to apply method and tool efficiently in this field is described by Grochtmann (1994). It was developed from experience gained in the process of test case determination for parts of a management system for a large educational institution.

Method and tool are not only applicable to software – recently a pure hardware system was successfully tested using method and tool: Test cases for the culling oversize system of an automatic mail sorting machine were generated, with real letters and postcards forming the corresponding test data.

During the trials, the test documentation generated proved to be appropriate and useful. It was especially helpful for the following test activities like test data generation. The fact that the method guides and supports testers but does not limit them was also positively judged by users. Moreover, in most cases the specification of the system under test could be improved, too.

However, experience showed also that a large, real-world system cannot be tested reasonably with a single classification-tree, as such a tree would become too large to handle. Therefore, the functionality of the system under test has to be divided into several separate test objects. This has to be done in such a way that each of the resulting test objects can be tested individually by means of the classification-tree method and that by testing all test objects the complete system is tested thoroughly (Grochtmann, 1994).

In general, it seems possible to achieve savings of up to 50% when the classification-tree method and the CTE tool are used (Wegener et al., 1994).

6. The Upcoming CTE Product

Since the CTE was well received in in-house practice, it is now being transformed into a product version. In this respect the trials already provided some valuable information on ways of further improving the CTE. Some of them are now realized in the first product version.

Important new features in the CTE product are (see Figure 6):

– In addition to several editing commands for the tree a number of editing commands for the combination table are available (1). They allow, for example, to easily generate a number of related test cases.

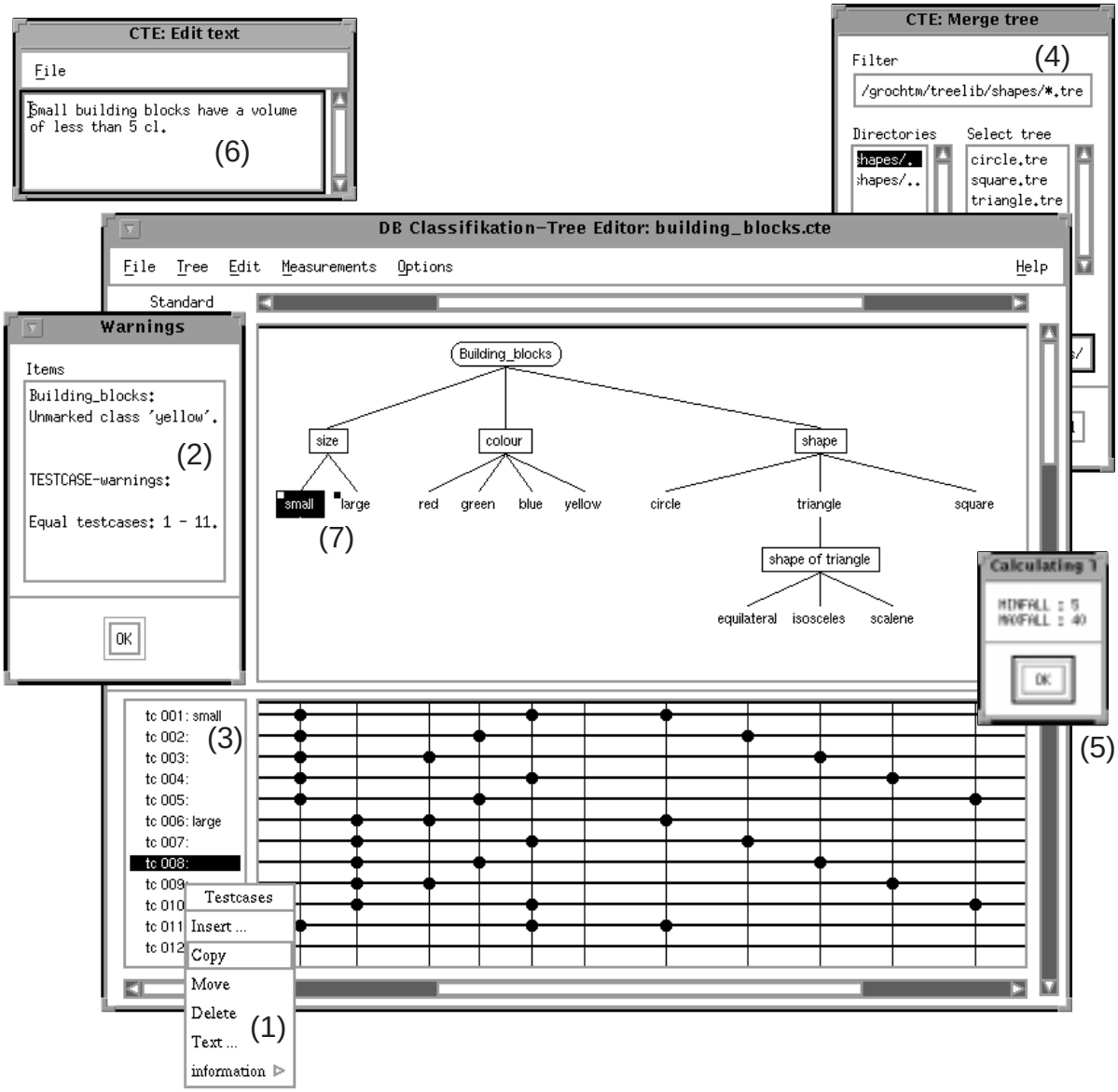

Figure 6: Features of the CTE Product Version

– Several consistency checks are available, e.g. checking for unused classes or for redundant test cases (2). In this way, some basic completeness and efficiency criteria for a thorough test can be guaranteed.

– In addition to the test case number the user can give each test case a descriptive name (3). During the practical trials it was found that the introduction of “meta test cases” which comprise a group of related test cases testing the same test idea is a powerful construction principle for the building of a large set of test cases. This can be documented readily by naming related test cases appropriately.

– A tree library can be used to save and later reuse parts of classification trees (4). An organization can use this feature to accumulate its testing knowledge gained during different projects and to reuse it later in new applications.

– Measures corresponding to the minimality and the maximality criterion are calculated automatically (5). The minimality criterion says that each class has to be used in at least one test case, the maximality criterion means that each possible combination of classes has to be used as a test case. The minimality criterion should always be fulfilled in a thorough test whereas, generally, there is no need to fulfil the maximality criterion as the number of necessary cases grows fast in real test problems and – as experience shows – normally only a limited number of combinations is required for a thorough test. The CTE calculates the minimum number of test cases which are necessary to fulfill the minimal criterion and the number of all different combinations of classes (maximality criterion). These measures help the tester to gain information on the complexity of the tree and thus the test problem.

– Tree and table elements can be annotated with explanatory texts providing additional documentation (6). Using these annotations the tester can, for example, express the reasoning behind the tree and table elements. Optionally, the user can request the CTE to display/print tree elements and test cases with special markings which show the existence of annotated texts or specifications (7).

The product version will be available in summer of 1995 for VMS, HP UX, SUN OS and Solaris. A PC/Windows version is planned for the end of 1995.

7. Conclusions

The classification-tree method and the CTE proved to be of high practical value for systematic test case design. A wider availability of these research results will be given by the CTE product version.

For the next versions of the CTE it is planned to enhance the functionality of the tool further. For example, an automatic generation of test cases in the table according to predefined combination rules or even the generation of complete classification trees from formal specifications is planned.

Furthermore, the CTE is now also an integral part of the overall computer-aided test system TESSY (Wegener and Pitschinetz, 1995). 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 such as test execution, monitoring and test evaluation.

In the future, research will focus on the test of large, distributed, parallel and real-time systems to further improve the support in these important areas of testing. Furthermore, regarding the growing importance of formal methods in the development of high-quality software, it is planned to combine the strengths of formal methods with systematic testing.

8. References

DeMillo, R.A., McCracken, W.M., Martin, R.J., Passafiume, J.F. (1987) Software Testing and Evaluation. Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.

Graham, D.R. (Ed.) (1991) Computer-Aided Software Testing: The CAST Report. Unicom Seminars Ltd., Middlesex, UK, 1991.

Grimm, K., Grochtmann, M. (1994) A New Approach to Systematic Testing of Safety-Related Computer Systems. International Conference on Computer Safety, Reliability and Security (SAFECOMP'94), 23-26 October 1994, Anaheim, California, USA.

Grochtmann, M., Grimm, K. (1993) Classification Trees for Partition Testing. Software Testing, Verification & Reliability, Volume 3, Number 2, June 1993, Wiley, pp. 63-82.

Grochtmann, M., Grimm, K., Wegener, J. (1993) Tool-Supported Test Case Design for Black-Box Testing by Means of the Classification-Tree Editor. EuroSTAR '93 – 1st European International Conference on Software Testing, Analysis and Review, 25 - 28 October 1993, London, UK, pp. 169-176.

Grochtmann, M. (1994) Test Case Design Using Classification Trees. Proceedings of STAR '94, 8-12 May 1994, Washington, DC, pp. 93-117.

Ostrand, T., Balcer, M. (1988) The Category-Partition Method for Specifying and Generating Functional Tests. Communications of the ACM, Volume 31, Number 6, June 1988, pp. 676-686.

Wegener, J., Pitschinetz, R. (1995) Tessy – An Overall Unit Testing Tool. Eighth International Software Quality Week (QW'95), 30 May - 2 June 1995, San Francisco, California, USA, in this volume.

Wegener, J., Pitschinetz, R., Grimm, K., Grochtmann, M. (1994) Tessy – Yet Another Computer-Aided Software Testing Tool? EuroSTAR '94 – 2nd European International Conference on Software Testing, Analysis and Review, 10 - 13 October 1994, Brussels, Belgium, pp. 36/1-36/13.