# Test Case Design on the Basis of Z Specifications

Harbhajan Singh, Mirko Conrad, and Sadegh Sadeghipour

DaimlerChrysler AG, Research and Technology, FT3/SM,
Alt-Moabit 96a, D-10559 Berlin, Germany
e-mail: {singh|conrad|sadegh}@daimlerchrysler.com

Software testing is the primary method through which the producer of software and the customer gain confidence that the software will work as intended or specified. A test which does not find errors can provide increased justification for confidence in the correctness of the test object. Since testing often consumes up to 50% of the overall costs for software, a cost-effective test process is decisive in industrial competition.

Especially in the area of safety-critical systems, the use of formal methods for the development of software is highly recommended by standards and legal requirements. These methods promise high product confidence because the precision of specifications written in a formal specification language forces specifiers to express requirements and design decisions in a clear and unambiguous manner. However, a post-developmental validation through testing is indispensable, and the use of formal methods can improve the software test process: The formal specification of the test object forms an ideal basis for a systematic design of test cases, test data generation, and test evaluation. Since it contains all the information necessary for the central test activities, these steps can be highly automated. The automation potential so created goes far beyond that which is possible by using natural language specifications. This large automation potential is essential to manage the complexity of large specifications and leads to a cost-effective software development.

Test case design is the most crucial activity for a trustworthy software test. A test case describes a certain input situation to be tested by comprising a set of input data. As formal specifications contain most of the information needed for black-box testing, a concept for automating test case design based on specifications written in the formal specification language Z [Spi92] has been recently developed [SCS97]. Any implemented operation, whose specification is given by a Z operation schema, can be considered as a test object. The concept is based on systematic partition testing, i.e, the input space of a test object is partitioned into equivalent classes with respect to its input specifications.

The main idea of the presented concept is an effective combination of the classification-tree method [GG93] with the disjunctive normal form approach [Gou83, DF93]. The classification-tree method is a special approach to partition testing and is being successfully used in various divisions of the Daimler-Benz Group. The disjunctive normal form approach is a widely used syntax-based approach for generating test cases on the basis of formal specifications. Both the methods are combined in the following way:

A classification tree is constructed from the formal specification, and the resulting high level test cases are transformed into refined test cases by generating a disjunctive normal form for them. The test cases obtained in this way cover all the specified aspects of the software explicitly. Furthermore, such test cases contain information for evaluating the test results. The combination contains advantages of both methods, overcomes most of their limitations, and can be supported by tools. An increased automation during testing in this way will result in an efficient and cost-effective software development.

In order to support this concept with powerful tools, the syntax-directed graphical classification-tree editor CTE, developed at Daimler-Benz Systems Technology Research in Berlin [GWG95], is being further extended. The further development of CTE proceeds

in two directions. The first direction concerns the semi-automatic classification-tree generation based on the formal specification. This involves mainly the following steps:

- Select the Z operation schema corresponding to the test object.
- Determine test relevant variables.These could be, for example, input and before-state variables, which are used in the predicate part of the schema under consideration.
- Note type information for these Variables for creating classification aspects and classes for them.
- Derive test relevant information for these variables by collecting their predicates individually by analysing schema predicates. Here we distinguish between partitioning and defining predicates.
- Offer this information for classification aspects and classes to the user interactively for constructing a classification tree.

The second direction of the extension of CTE concerns following aspects:

- Derivation of test case specifications in Z through conjunction of selected classes with the Z schema of the test object.
- Interface with the proof tool Isabelle for transforming a selected high level test case into DNF to get refined test cases.
- Interface to represent and handle refined test cases.

## References


DF93. J. Dick and A. Faivre. Automating the generation and sequencing of test cases from model-based specifications. In J.C.P. Woodcock and P.G. Larsen, editors, *Proceedings of Formal Methods Europe '93*, volume 670 of *LNCS*, pages 268–284, 1993.

GG93. M. Grochtmann and K. Grimm. Systematischer Test mit der Klassifikationsbaum-Methode. Technical report, Daimler-Benz AG, 1993.

Gou83. J. S. Gourlay. A mathematical framework for the investigation of testing. *IEEE Transactions on Software Engineering*, SE-9(6):686–709, November 1983.

GWG95. M. Grochtmann, J. Wegner, and K. Grimm. Test case design using classification trees and the classification-tree editor. In *Proceedings of 8th International Quality Week, San Francisco*, pages Paper 4–A–4, May 30-June 2 1995.

SCS97. H. Singh, M. Conrad, and S. Sadeghipour. Test case design based on z and the classification-tree method. In M.G. Hinchy and S. Liu, editors, *Proceedings of First IEEE Internationsl Conference on Formal Engineering Methods*, pages 81-90. IEEE Computer Society, 1997.

Spi92. J.M. Spivey. *The Z Notation. A Reference Manual.* Prentice Hall, New York, 2 edition, 1992.