BZ-Testing-Tools: Model-Based Test generation

Bruno Legeard
LIFC - CNRS - INRIA
, Université de Franche-Comté
16, route de Gray - 25030 Besançon, France
Email: legeard@lifc.univ-fcomte.fr

1 What is BZ-Testing-Tools

BZ-Testing-Tools – BZ-TT – is a tool-set for animation and test generation from B, Z and Statechart specifications. BZ-Testing-Tools provides several testing strategies (partition analysis, cause-effect testing, boundary-value testing and domain testing), and several test model coverage criteria (multiple condition coverage, boundary coverage and transition coverage).

BZ-TT takes as an input a formal model of the technical requirements of the system under test. In fact, the BZ-TT uses one internal form (BZP). The formal model are automatically translated into this form.

BZ-TT uses a customized constraint solver to symbolically execute the input formal model, both for model animation (Animator) and for test computation (Test Generator).

BZ-TT generates abstract test cases which are translated into executable test scripts in order to harness the test execution on the test bed (Test Reification).

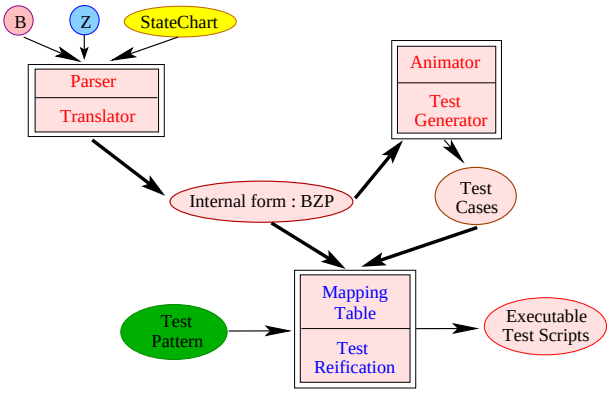


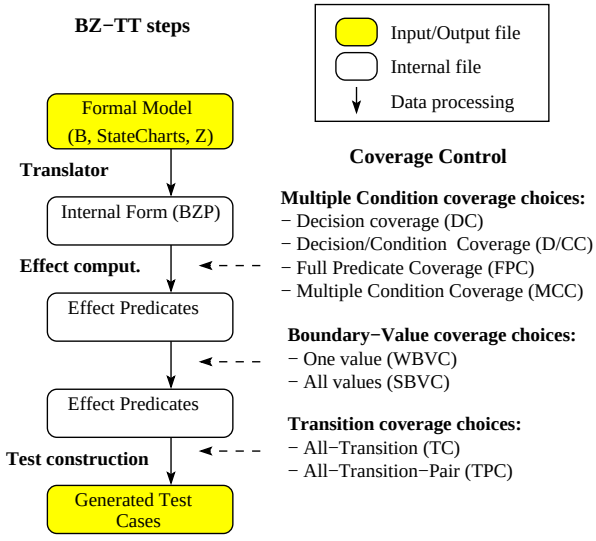
Figure 1. Architecture of BZ-Testing-Tools

2 BZ-TT test coverage criteria

The main idea of BZ-TT is to produce a test case for each behavior defined in the formal model and for each boundary value. Therefore, BZ-TT generates the test suite using three families of model coverage criteria:

- **Multiple condition coverage** – This family relates to multiple conditions in the decisions that appear in the model,
- **Boundary value coverage** – This family relates to the choice of values in equivalence classes,
- **Transition coverage** – This family relates to the coverage of transitions or transition-pairs in the model.

3 The BZ-TT test generation process



4 Unique features of BZ-TT

The unique features of the BZ-TT method are that it:

- takes B, Z and Statecharts specifications as input;
- automatically produces boundary-value test cases (both boundary states and boundary input values);
- produces both nominal and robustness test cases;
- is fully supported by tools: the BZ-TT environment;

- has been validated in several industry case studies(GSM 11-11 smart card software, Java Card Virtual Machine Transaction mechanism, a ticket validation algorithm in the transport industry and an automobile windscreen wiper controller);
- allows the validation engineer to drive the test generation process.

5 What is a generated test case

The BZ-TT method consists of testing the system when it is in a boundary state, which is a state where at least one state variable has a value at an extremum – minimum or maximum – of its sub-domains. At this boundary state, we want to test all the possible behaviours of the specification. That is, the goal is to invoke each operation (or external event) with extremum values of the sub-domains of the input parameters.

We divide the trace constituting the test case into four sub sequences:

- Preamble: this takes the system from its initial state to a boundary state.
- Body: this invokes one update operation with input boundary values.
- Identification: this is a sequence of observation operations to enable a pass/fail verdict to be assigned.
- Postamble: this takes the system back to the boundary state, or to an initial state. This enables test cases to be concatenated.

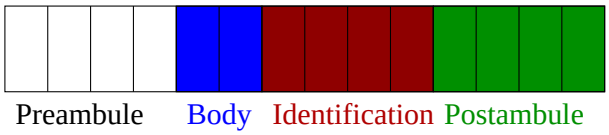


Figure 2. Test sequence

The test generation algorithm computes positive test cases with valid boundary input values at body invocations. A set of one or more test cases, concatenated together, defines a test sequence. For negative test cases, the body part is generated with invalid input boundary values, and no identification or postamble parts are generated, because the system arrives at an indeterminate state from the formal model point of view. Instead, the test engineer must manually define an oracle for negative test cases (typically something like the system terminates without crashing).

After positive and negative test cases are generated by this procedure, they are automatically translated into executable test scripts, using a test script pattern and a reification relation between the abstract and concrete operation names, inputs and outputs.

6 The validation engineer drives the test generation process

BZ-TT is a model-based test generator which helps to automate black-box testing for systems or components. But it is a tool for the validation engineer and it gives him/her all the facilities to drive the test generation. Firstly, the validation engineer adapts (or constructs) the formal model for the test purpose. This means that it is light-weight formal modeling taking in account the test objectives and the point of control and observation of the system under test to define the level of abstraction of the model. Secondly, the validation engineer uses the GUI provided by BZ-TT to choose the model coverage criteria (see the screenshot below).

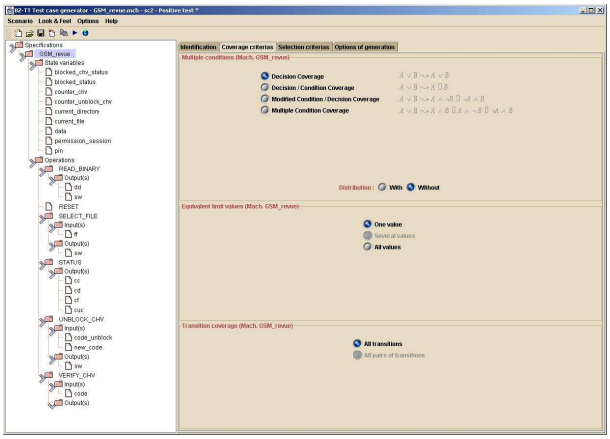


Figure 3. Test generation GUI

7 Maturity of the BZ-TT technology

The BZ-TT tool-set is developed at the Computer Science Laboratory of the University of Franche-Comté CNRS INRIA (France), in the Constraint group led by Professor Bruno Legeard, and in partnership with Dr. Mark Utting, from the University of Waikato (New Zealand).

The BZ-TT technology is currently in an industrialisation process in a new start-up company, LEIRIOS Technologies – www.leirios.com.

8 Some publications of the BZ-Testing-Tools project

- B. Legeard, F. Bouquet
  Reification of Executable Test Scripts in Formal Specification-Based Test Generation: The Java Card Transaction Mechanism Case Study
  In proc. of FME'03, Formal Methods Europe, LNCS n° xxxx, pages xx-xx, Springer, September 2003, Pisa, Italy.

- F. Ambert, F. Bouquet, S. Chemin, S. Guenaud, B. Legeard, F. Peureux, M. Utting, N. Vacelet
  BZ-Testing-Tools: A Tool-Set for Test Generation from Z and B using Constraint Logic Programming
  In proc. of FATES'02, Formal Approaches to Testing of Software, Workshop of CONCUR'02, Brnö, Czech Republic, August 2002, Technical Report, INRIA, pp 105-120.

- B. Legeard, F. Peureux, M. Utting
  Automated boundary testing from Z and B
  In proc. of FME'02, Formal Methods Europe, LNCS n° 2391, pages 21-40, Springer, July 2002, Copenhaguen, Denmark.

- F. Bouquet, B. Legeard, F. Peureux
  CLPS-B - A Constraint Solver for B
  In proc. of the conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'02, ETAPS, LNCS 2280, p.188-204, Springer, April 2002, Grenoble, France.

- B. Legeard, F. Peureux, M. Utting
  A comparison of the BTT and TTF/Z Test-Generation Methods
  In proc. of ZB'02, International Conference on Z and B Formal Methods, LNCS 2272, p. 309- 329, Springer, January 2002, Grenoble.

- B. Legeard, F. Peureux
  Generation of functional test sequences from B formal specifications presentation and industrial case study
  In proc. of ASE'01, International Conference on Automated Software Engineering, IEEE Computer Society Press, p. 377-381,November 2001, San Diego, USA.