**Coverage**

Each path in a test model from the start node to an end node is a potential test case. Therefore it is possible to define different coverage measures against the model. The user can introduce special information into his model which can be used to define specific coverage criteria. He can also mark special paths in the model as more relevant than others for which a coverage measure can be defined.

The model-based test design allows measuring the test coverage against different features of the model. A test case generator should be able to support at least the following coverage criteria.

**Model-based coverage definition**

Requirements for Model Coverage
Standard coverage criteria from white box testing of code may also be applied to test models. Test case generators shall support a least the following coverage criteria:

C0: Node Coverage
C1: Edge Coverage
C2: (Full) Path Coverage

Node coverage

Node coverage means every node is used at least in one test case.

*Node coverage shall be supported.*

Edge coverage

This means every edge is used at least in one test case.

*Edge coverage shall be supported.*

Full Path coverage

Full Path coverage means every possible path is used at least in one test case. Duplicate test cases should be avoided.

*Full path coverage must be possible at least without considering loops.*

For test models with loops, full path coverage (C2a) results in an infinite number of test cases. Therefore, it must be possible to limit the number of loops. At least one of the following possibilities shall be supported:

* + C2b: Boundary-Interior (=C2a with specific rules for loops)
	+ C2c: Structured (C2a with exactly n loops)

*Condition Coverage (C3) should be supported either through modeling of test data or by inclusion of decision tables / classification trees.*

Loop Definition

A test case generator must provide a definition for “loop”. There are at least three different variants:

* Node based – a loop is counted if the same node is reached the next time regardless of the entry edge
* Edge based- a loop is counted if the same node is reached by the same entry edge
* Path based – loop is counted if the same node is reached by using the same path

*Full Path coverage without loops shall be supported.*

*Full Path coverage considering loops should be supported*

User-defined coverage definition

* Requirement coverage
* Named Path coverage

Requirement coverage

Requirements can be assigned to diagrams or to single elements of a model. One requirement can be assigned to one or more elements. Requirement coverage can be defined in different ways.

1. Simple coverage: at least one test case for every requirement
2. Complete coverage: each path containing the requirement ( one or more TC for every requirement)
3. Single requirement coverage: all occurrences of a requirement are covered ( one or more TC for every requirement)

*At least the simple requirement coverage shall be supported.*

*The “complete” requirement coverage should be supported.*

*The single requirement coverage should be supported.*

Named Path coverage

A Named Path is a path through the model that has been explicitly assigned. It describes one single potential test case.

*One or more Named Paths in one model shall be supported.*

User-defined coverage

It is possible to introduce additional information into a model which can be used to define user-specific coverage criteria e.g. risk coverage.

*The user should be able to use this information as coverage criteria.*

General Requirements on Generation Strategies

Test models are often structured in different hierarchies to keep the diagrams simple and readable. To avoid test case explosion it is useful to use different coverage criteria in different diagrams of a model.

*It should be possible to define coverage criteria for sub-diagrams to avoid test case explosion.*

*It should be possible to exclude inconsistent paths from the generation (e.g. using guard conditions).*

**Filtering/Selection**

Generating test cases from models creates often too much test cases to be executed even for medium-sized models. Therefore it’s essential to be able to select or filter test cases to create reduced test sets or to create test sets for special test purposes.

*Selection of test sets from a generated test suite should be supported.*

To reduce the number of test cases the use of filters is also a very helpful method. Filters can be defined in respect to features of the model, e.g. nodes or edges, or to information the user annotates e.g. requirements or costs.

A common scenario for filter usage is for example to generate test cases using a full path coverage strategy and filter the test cases which are of interest for the current test situation. This could be test cases only for those requirements that already have been implemented.

*It should be possible to filter model features like edges, nodes, and path lengths.*

*It should be possible to filter user-specific features like requirements, priorities, costs, durations etc.*

A common scenario would be to select a test set which has an overall execution time not exceeding a fixed limit.

Often it is necessary to combine more than one filter to receive the needed result. Therefore:

*It should be possible to use a sequence of filters.*