

Model-based testing

Angelo Gargantini

Testing e Verifica del Software AA 2526

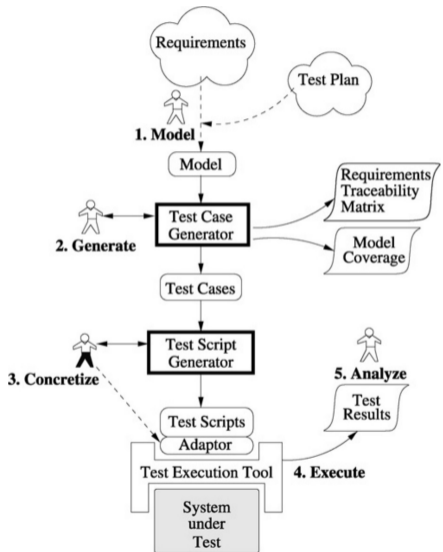
29 aprile 2026

What is model-based testing

Model-based testing has become a bit of a buzzword in recent years, and we have noticed that people are using the term for a wide variety of test generation techniques. The following are the four main approaches known as model-based testing.

- 1 Generation of test input data from a domain model
- 2 Generation of test cases from an environment model
- 3 Generation of test cases with oracles from a behavior model
- 4 Generation of test scripts from abstract tests

Model-Based testing process



steps for MBT

- 1 Model the SUT and/or its environment.
- 2 Generate abstract tests from the model.
- 3 Concretize the abstract tests to make them executable.
- 4 Execute the tests on the SUT and assign verdicts.
- 5 Analyze the test results.

1. Writing the model

- The first step of model-based testing is to write an abstract model of the system that we want to test.
 - ABSTRACT: much smaller and simpler than the SUT itself.
- check that the model is consistent and has the desired behavior.
 - for example using the simulator/ animator (and model checker)

- The second step of model-based testing is to generate abstract tests from the model.
- Use some selection criteria for “covering” the model

- The third step of model-based testing is to transform the abstract tests into executable concrete tests.
- abstract tests lack some of the detail needed by the SUT and are not directly executable.
 - spesso fatto a mano (time consuming)
 - This may be done by a transformation tool, which uses various templates and mappings to translate each abstract test case into an executable test script.
 - Or it may be done by writing some adaptor code that wraps around the SUT and implements each abstract operation in terms of the lower-level SUT facilities.

- The fourth step is to execute the concrete tests on the system under test.
 - online model-based testing, the tests will be executed as they are produced, so the model-based testing tool will manage the test execution process and record the results.
 - offline model-based testing, we have just generated a set of concrete test scripts in some existing language, so we can continue to use our existing test execution tools and practices.

How to model your system

- The first and most important step in modeling a system for testing is deciding on a good level of abstraction, that is, deciding which aspects of the SUT to include in your model and which aspects to omit.
 - Since the model is just for test generation purposes, it does not have to specify all the behavior of the system.
 - Several smaller partial models are often more useful than one huge and complex model.
- The following are some typical simplifications:
 - Focus primarily on the SUT
 - Include only those operations that you wish to test
 - Include only the data that are useful for modeling the behavior of the operations
 - Replace a complex data field, or a class, by a simple enumeration.

Two possible choices

- 1 Model only the inputs of your program
 - focus on what are the “critical” inputs of your system and use them
 - PARTITION/COMBINATORIAL TESTING
- 2 Model also the behavior
 - FSM and FSM-based testing