

Reactis Three-Page Product Documentation

Steve Sims, Reactive Systems, February 28, 2005

Reactis is an *embedded software design automation* tool suite --- a collection of applications supporting the efficient production of high-quality control software. Reactis currently consists of three components:

Reactis Tester generates comprehensive test suites from Simulink®/Stateflow® models. The test suites exercise large portions of the software under test while avoiding redundancy, thereby maximizing the probability of finding defects within the time available for testing. The generated tests store model-generated outputs as well as inputs so that test harnesses can automatically check the correctness of source code implementations of models.

Reactis Simulator enables users to visualize the results of executing tests produced by Reactis Tester. The tool also enables users to fine-tune Tester-generated test suites and to visualize data item values and coverage information during simulation.

Reactis Validator automatically searches models for violations of user-specified requirements.

Reactis Tester automatically generates test suites from models. The test suites provide comprehensive coverage of different test-quality metrics---including the *Modified Condition/Decision Coverage* (MC/DC) test coverage measure mandated by the US Federal Aviation Administration (FAA) in its DO-178/B guidelines---while at the same time minimizing redundancy in tests. Each test case in a test suite consists of a sequence of inputs fed into the model as well as the responses to those inputs generated by the model. These tests may then be used for a variety of purposes, including the following.

Implementation conformance.

The tests may be applied to source-code implementations of models to ensure conformance with model behavior.

Model testing and debugging.

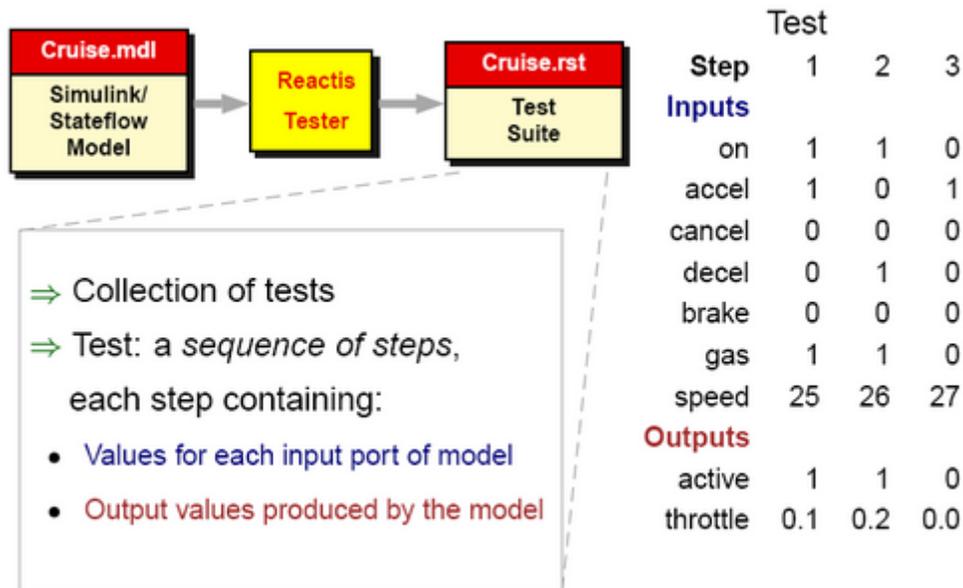
The tests may be run on the models themselves to study and revise model behavior.

Reverse engineering of models from source.

Tests may be generated from models derived from legacy code in order to check conformance between model and code.

Reactis Tester enables engineers to maximize the effectiveness of testing while reducing the time actually spent on testing.

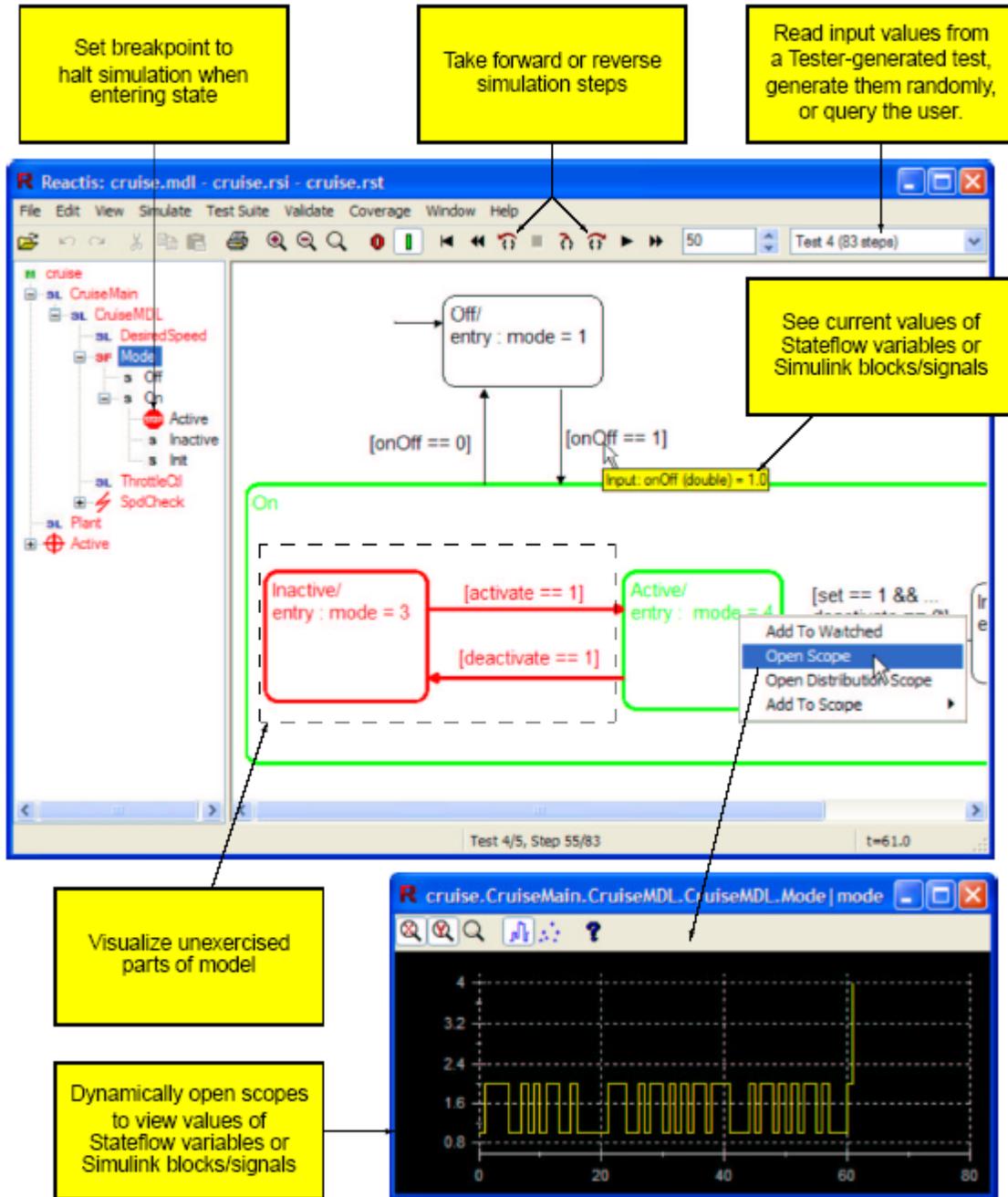
The structure of Tester-generated test suites is shown below. A test may be viewed as a matrix in which each row corresponds to either an inport or outport and each column represents a simulation step.



Test suites are constructed by simulating a model and recording the input and output values at each step. The model computes the outputs at each step, but several approaches are possible for selecting the input values to drive simulation. The input data could be captured during field testing or constructed manually by an engineer, but these are expensive tasks. Alternatively, the inputs could be generated randomly; however, this approach yields tests with poor coverage.

Reactis Tester employs a novel approach called *guided simulation* to generate quality input data automatically. The idea behind this approach is to use algorithms and heuristics to automatically generate inputs that cause *coverage targets* (i.e. model elements that the user wants to ensure are executed at least once) that have not yet been covered to be executed. Reactis currently allows users to track several different classes of targets, or *coverage criteria* such as exercising all branches, entering every Stateflow state at least once, and covering all *Modified Condition/Decision Coverage (MC/DC)* targets

Reactis Simulator enables users to visualize model execution. Simulator's user interface is similar to those of traditional debuggers from programming languages: it allows users to step through the execution of models by hand as well as set break points. Simulator also supports reverse execution, the replay of tests generated by Reactis Tester, the graphical display of different coverage metrics, and the capability to fine-tune Tester-generated test suites.



Reactis Validator performs automated searches of models for violations of user-specified requirements. If Validator finds a violation, it returns a test that leads to the problem. This test may then be executed in Reactis Simulator to gain an understanding of the sequence of events that leads to the problem. Validator enables the early detection of design errors and inconsistencies and reduces the effort required for design reviews.