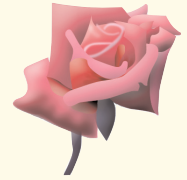


# SoBEIT

## Structural or Behavioural Execution Instrumentation Tool



**SoBEIT** supports development and assurance of quality software through automated specification-based testing. SoBEIT supports software testers with the following capabilities:

- persistent test artifact development and maintenance;
- fully automated testing, including monitored test execution;
- formal, automated test result checking via specification-based test oracles;
- functional and structural test adequacy definition and measurement.

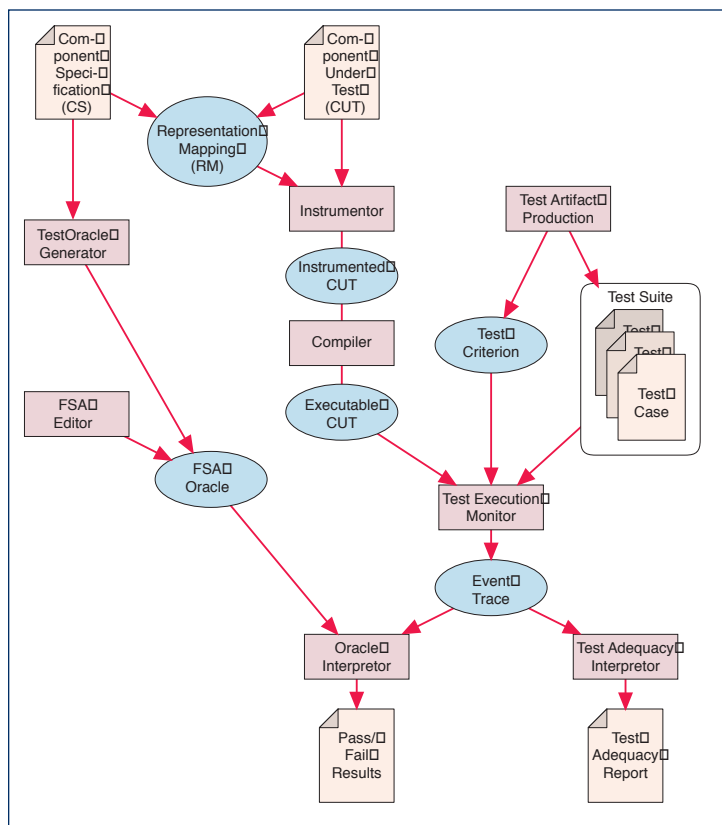


Figure 1 - SoBEIT Process Flow Diagram

As illustrated in Figure 1, the tester provides SoBEIT with the component-under-test (CUT) and corresponding component specification (CS) along with a persistent test suite (TS) and SoBEIT does all the rest:

- the Instrumentor takes a representation mapping relating events in CS to occurrences in CUT and instruments the CUT to collect event traces;
- the instrumented CUT is automatically executed on selected test cases specified in TS, based upon choices made by the tester (as illustrated in Figure 2);
- the test executions are monitored to collect the required event traces;
- an FSA representation of the specification CS is interpreted on the event traces, thus serving to compare the test execution results to the specification;
- pass/fail results are reported.

**Test Artifact Production.** SoBEIT enables the development, maintenance, and persistence of test process artifacts - such as test cases, test suites (sets of test cases), test oracles, and test criteria - as well as relationships between those artifacts. Test artifacts can be created by the tester, imported from external sources, or generated from other tools. The test artifacts are maintained in a persistent object repository to enable perpetual testing activities.

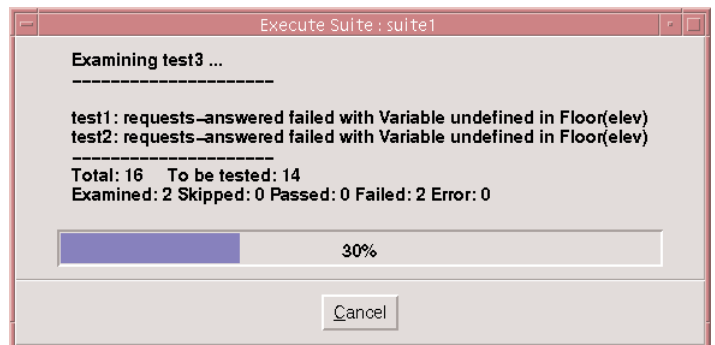


Figure 3 - SoBEIT Test Suite Execution Progress

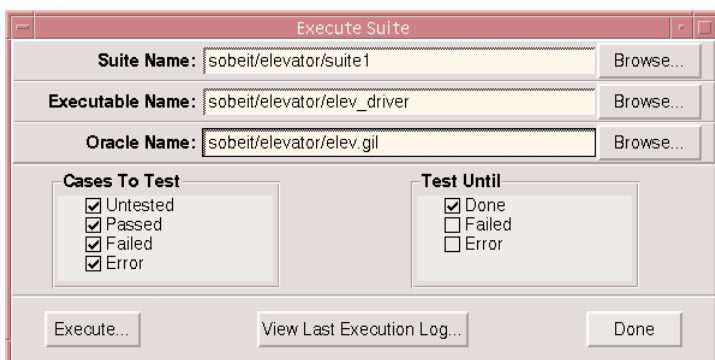


Figure 2 - SoBEIT Execution Suite

**Automated Test Execution.** SoBEIT provides the capability to automate test execution while monitoring for test adequacy measurement and test result checking. Monitoring is accomplished by instrumenting the program to collect relevant traces. After execution of a test suite, a summary report is generated which provides access to individual test case reports (as illustrated in figure 4).

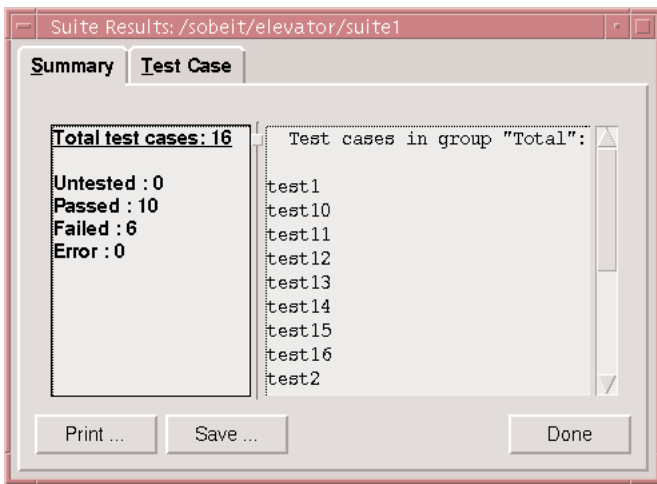


Figure 4 - SOBEIT Test Suite Results

**Formal Test Result Checking.** SOBEIT implements the EASOF (Execution-Time Analysis of Specification-based Oracle Failures) model, through which test oracles are derived from formal specifications and formal test result checking is supported. This facilitates a much more effective testing process through the use of formally defined test oracles – mechanisms for automatically determining if test execution behavior satisfies required properties – since manual test result checking, as is usually done, is one of the most ineffective, costly, and error-prone activities in the testing process. SOBEIT translates formal specifications (such as those developed in GIL, Graphical Interval Logic, illustrated in Figure 5) of required properties into finite state automata (FSA) (see Figure 6), which are then used for run-time test result checking. FSAs that specify required properties can also be developed directly via a FSA Editor.

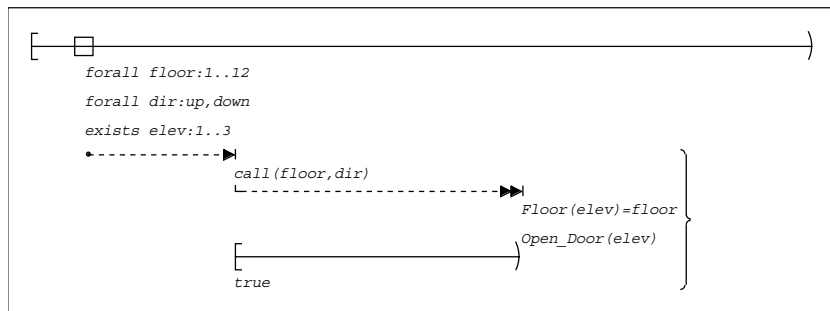
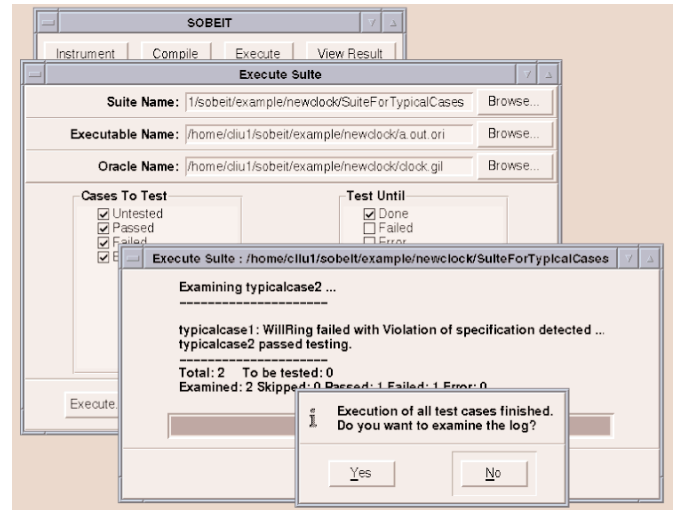


Figure 5 - GIL Specification

**Test Adequacy.** SOBEIT measures test adequacy based upon covering selected specification behaviors (based upon the FSA representation) or program structures. Thus, SOBEIT supports both functional (black box) and structural (white box) test criteria.



SOBEIT is written in Tcl/Tk but it integrates components written in a variety of languages. The Instrumentor currently works for Ada83, although instrumentation can be done manually. The OracleGenerator currently works for GIL, but FSAs can be provided directly via an FSA Editor. The rest of SOBEIT is language independent. SOBEIT is a scaled-down version of ROSATEA's TAOS environment serving to prototype structural and behavioral instrumentation capabilities.

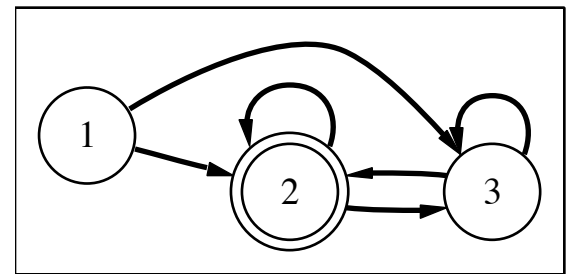
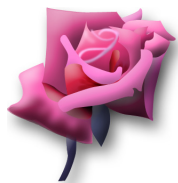
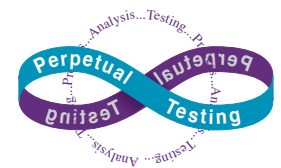


Figure 6 - FSA of GIL in Figure 5



The ROSATEA tools are research prototypes, some of which have been successfully transitioned into use on real projects, but not yet as commercial systems. These tools were developed by the Research Organization for Specification- and Architecture-based Testing E (&) Analysis at the University of California at Irvine. The work was done in conjunction with the Perpetual Testing Projects sponsored by the DARPA's EDCS program.



The SOBEIT research and development effort is sponsored in part by: the Air Force Materiel Command, Air Force Research Laboratory, and the Defense Advanced Research Projects Agency under agreement number #F30602-97-2-0033. The views and conclusions contained herein are those of the researchers and should not be interpreted as representing the official position or policy, either expressed or implied, of the U.S. Government, AFMC, Rome Laboratory, DARPA, or the University of California, and no official endorsement should be inferred.

**Freely Available Software**  
Information about UC Irvine's Research Organization for Specification- and Architecture-based Testing E (&) Analysis (ROSATEA), as well as its software, is available at:

<http://www.ics.uci.edu/pub/rosatea/>

**Contact Information**  
Professor Debra J. Richardson  
Information and Computer Science  
University of California  
Irvine, California 92697-3425

url: <http://www.ics.uci.edu/~djr>  
email: [djr@ics.uci.edu](mailto:djr@ics.uci.edu)  
voice: 949-824-7353  
fax: 949-824-1715