First International Workshop on Automated Program Analysis, Testing and Verification

Automated Testing of Real-Time Tasks

Joachim Wegener, Roman Pitschinetz, Harmen Sthamer
DaimlerChrysler AG, Research and Technology, Alt-Moabit 96a, D-10559 Berlin, Germany
Joachim.Wegener@daimlerchrysler.com
Roman.Pitschinetz@daimlerchrysler.com
Harmen.Sthamer@daimlerchrysler.com

The development of embedded systems is a crucial area of responsibility in industrial practice. Many embedded systems need to meet real-time requirements. This adds a new dimension to the testing of such systems – not only the logical behavior, but also the temporal behavior of these systems requires thorough testing. In comparison with conventional software systems, the testing of embedded systems is more complex due to several specific technical characteristics such as the development in host-target environments, the intense interaction of the systems with the real application environment, and the limited resources of the target system. In order to facilitate systematic and largely automated testing in defiance of the complexity of real-time systems powerful testing tools are required. Therefore, in this work the testing system TESSY has been extended in order to support the total testing life-cycle of real-time tasks. New components allow a thorough examination of the logical as well as the temporal behavior of the tasks. The logical behavior is tested by means of function-oriented and structure-oriented testing methods; the testing of temporal behavior is automated by evolutionary testing.

TESSY [1] concentrates mainly on test case design, test execution, monitoring, test evaluation, and test documentation. TESSY automates all test activities except the test case generation for examining logical program behavior. In order to automate the test execution, the required test drivers are generated, communication between host and target system is automatically built, the program code is instrumented and coverage analysis is performed, and the execution times on the target system are measured. Regression testing is also entirely automated by TESSY.

For the generation of functional test cases, TESSY uses the classification-tree method [2]. TESSY therefore, contains the classification-tree editor, CTE [3]. Branch testing is supported by the structure-oriented test method. It is possible to instrument the program code to record the branches executed during functional testing and to define the amount of branch coverage obtained. On the basis of this information, the functional test may be further improved or expanded by structure-oriented test cases. This test strategy guarantees an extensive test of the logical program behavior. The test can be run with or without instrumentation in order to exclude side-effects from the instrumentation. The results generated from the test object will then be automatically compared with each other and deviations documented in the generated test documentation.

The most important property, however, is the automation of testing temporal behavior by means of evolutionary testing. Errors in the temporal behavior of real-time systems usually result from a violation of specified timing constraints. The tester's task is to find input situations that result in the maximum execution times. If the execution times exceed the specified constraints, an error has been detected. In evolutionary testing the search for the longest execution time is considered a discontinuous, nonlinear optimization problem, with the input domain of the test object as search space, sets of test data as decision variables, and execution times as objective values. In order to solve this optimization problem, evolutionary algorithms are used to approximate the longest execution times of a test object within several generations. The application of evolutionary algorithms for test data generation is known as evolutionary testing. Previous works have shown that evolutionary testing is superior to random testing [4] and systematic testing [5] when it comes to examining the temporal behavior of real-time systems.

Logical and temporal behavior testing are combined through *seeding*. Test data collected by the tester for the functional test are integrated into the initial population of the evolutionary test. This means that the evolutionary test benefits from the tester's knowledge concerning the functions and internal structures of the test object. The search does not commence with a randomly generated population.

The first industrial application of TESSY with the set of properties described in this paper was initiated last year for testing an engine control system containing more than 20 different tasks. All tasks were tested for their logical program behavior with the classification-tree method and complete branch coverage for all the tasks was reached. Further, six time-critical tasks have been tested for their temporal behavior with evolutionary testing. To avoid probe effects (deviations from actual run-time behavior) instrumentation is turned off for the tasks.

The number of input parameters of these tasks varies from 9 to 18 with a number of program lines set between 39 and 119, the static program paths differ from 1 to 37 million and the cyclomatic complexity from 1 to 27. For each task evolutionary testing generated between 7,500 and 15,000 sets of test data. The target processor is the Siemens C167 with 1 Mbytes SRAM and with a speed of 20 MHz. The testing of one single task took approximately 1 hour and all tests were carried out on the target system that has been designated for future use in cars. The execution times were determined using hardware timers of the target environment with a resolution of 400 ns. The results of the evolutionary tests compared with the execution times determined by the developers' tests are shown in Table 1.

task	Longest execution time in µs		Lines of code	No. of parameters	Program paths	Cyclomatic Complexity
	Evolutionary test	Developer test				
1	69,6 µs	67,2 µs	41	18	224	10
2	120,8 µs	108,4 µs	119	18	37.748.736	27
3	112,0 µs	108,4 µs	98	17	1	1
4	68,8 µs	64,0 µs	81	32	60.480	23
5	59,6 µs	57,6 µs	39	14	408	11
6	58,4 µs	54,0 µs	56	9	36.864	18

Table 1: Maximum execution times of engine control tasks determined by evolutionary testing and developers' tests

These TESSY extensions described have proved to be highly applicable in practice for testing an engine control system. Both, the logical and the temporal behavior have been thoroughly tested. The deployment of the CTE methodology has been approved and utilized by the developers in order to generate systematic test cases that obtained 100% branch coverage. All other test activities are fully automatically executed on the target system, specifically the testing of the temporal behavior. For the 6 tasks testing the temporal behavior, longer execution times were found with the evolutionary test than with the developers' tests. This proved to be the case even though evolutionary testing treats the software as black boxes, whereas developers are familiar with the function and structure of the software and achieve 100% branch coverage. An explanation might be the use of system calls, linkage, and compiler optimization whose effects on temporal behavior can only be guessed with difficulty by the developers. However, it should be noted that the execution times determined did not exceed the specified timing constraints for any of the tasks. The intensive testing has certainly strengthened the developers' confidence in a correct temporal behavior of the system. With an average number of 7 regression tests for each task, TESSY's entirely automated execution of regression testing has proved extremely useful.

Future work on testing real-time systems will focus on how static analysis techniques could support evolutionary testing, e.g. for search space reduction, to find a selection of evolutionary algorithms for test use, and to obtain information on internal states of the test object that may influence its temporal behavior. Further, the combination of evolutionary testing methods with static analysis techniques for the estimation of worst case execution times is meant to facilitate a precise forecast of the actual longest execution times of tasks [6]. Future plans, include the expansion of TESSY for integration testing and the examination of the suitability of evolutionary testing for system testing.

References:

[1] Wegener, J. and Pitschinetz, R. (1994): *TESSY - Yet Another Computer-Aided Software Testing Tool?* Proceedings of the European International Conference on Software Testing, Analysis & Review EuroSTAR '94, Bruxelles, Belgium.

[2] Grochtmann, M. and Grimm, K. (1993): *Classification Trees for Partition Testing*. Software Testing, Verification & Reliability, vol. 3, no. 2, pp. 63-82, Wiley.

[3] Grochtmann, M. and Wegener, J. (1995): *Test Case Design Using Classification Trees and the Classification-Tree Editor CTE*. Proceedings of the Software Quality Week '95, San Francisco, USA.

[4] Wegener, J. and Grochtmann, M. (1998): *Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing*. Real-Time Systems, vol. 15, no. 3, pp. 275-298, Kluwer Academic Publishers.

[5] Mueller, F. and Wegener, J. (1998): *A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints.* Proceedings of the IEEE Real-Time Technology and Applications Symposium RTAS '98, pp. 144-154, Denver, USA.

[6] Wegener, J., Pohlheim, H., and Sthamer, H. (1999*): Testing the Temporal Behavior of Real-Time Tasks using Extended Evolutionary Algorithms.* Proceedings of the European International Conference on Software Testing, Analysis & Review EuroSTAR '99, Barcelona, Spain.