Evolutionary Testing in Component-Based Real-Time System Construction

Hans-Gerhard Groß

Fraunhofer Institute for Experimental Software Engineering

Sauerwiesen 6
D-67661 Kaiserslautern, Germany

grossh@iese.fhg.de

Nikolas Mayer

Fraunhofer Institute for Experimental Software Engineering

Sauerwiesen 6
D-67661 Kaiserslautern, Germany

mayern@iese.fhg.de

Abstract

Execution time analysis is an essential verification activity during real-time system development. This activity can be performed dynamically through optimization-based analysis techniques such as evolutionary testing, and it is already successfully used under the traditional procedural development paradigm. Our work represents an initial attempt in making evolutionary testing also applicable under the more recent object-oriented, component-based software development paradigm.

1 REAL-TIME REQUIREMENTS

Real-time requirements are not only affected by individual components in a component-based real-time system, but also by the entirety of all components that make up an application. Each individual component may have a well-defined timing behavior on a particular platform. However, this is completely changed if it is plugged to other components that collectively implement functionality of a real-time application. The timing behavior of each possible combination of a component's feasible usage profiles in respect to other components must therefore be verified when the compliance to the timing schedule of such a system is tested. Since the component developer cannot anticipate the usage profile of a component in a particular application, timing verification must be performed when components are assembled and put together into a new application.

2 EVOLUTIONARY TESTING

Evolutionary testing applies evolutionary algorithms to typical software testing problems. The target is to find optimal tests for a given test criterion. Dynamic timing analysis can be defined as software testing with the violation of the timing schedule as test criterion. This, in fact, represents a typical optimization problem that can be tackled by an evolutionary search process. The related cost function is represented by the time it takes to execute the operation under test.

The fact that object-oriented entities are inherently encapsulated, and they also represent state machines, creates a fundamental difficulty for the application of evolutionary testing. In order to generate worst-case timing behavior for some operation of an object, an evolutionary algorithm must not only optimize and provide the input parameter values for the operation, but additionally, it must optimize and provide appropriate initial states from which the event will be triggered. The execution time of an operation is defined through the values of the internal state variables plus the input values of the operation. The evolutionary algorithm must set the values for the internal state variables from outside the object's encapsulation boundary. However, most components will not exhibit their internal state attributes. Dynamic execution-time analysis that is applied to component-based real-time systems is therefore relying on a specific component architecture.

3 EXTENDED COMPONENTS

Such an extended component architecture may be realized by an additional testing interface of the component under test to enable the needed controllability for performing dynamic execution time assessment. An external client of the component can access this interface; for this instance it is the test software in form of the evolutionary testing algorithm. It uses the evolutionary testing interface to set the internal state variables of the tested component to the values that are generated within the evolutionary process. The normal functional interface is used to execute the tested operation and provide its input parameter sets generated by the evolutionary algorithm. The execution time of the operation is captured and interpreted as the fitness of the current test set by the evolutionary algorithm.