Using Application States in Software Testing

Chang Liu **Debra J. Richardson**
Information and Computer Science
University of California, Irvine
Irvine, CA 92697, USA
+1 949 824 7353
{liu,djr}@ics.uci.edu

Keywords
software testing, test automation, coverage analysis.

1 APPLICATION STATES

We propose to use application states in software testing to better associate test cases with coverage information and to enable test designers to adjust the emphasis of test coverage.

Application states are abstract states of application software characterized only by properties interesting to test designers in a particular situation. In a more formal way, application states can be defined as follows:

When a software application A is running, suppose domain D is the set of all internal or external events or entities that could affect the behavior of A. A *property p* is a function on domain D. Its range V is a set of all possible values of this property. The mapping between D and these values are determined by test designers. Domain D is very large in practice. It is not feasible to define P by assigning a value v for every single point in D. A piece of code that checks the status of D and returns a value is more viable way to determine the value of a property.

A *property set PS* is a set of properties. The value of one property may be dependent on the value of another property within the same property set.

A *property collection PC* is a subset of a property set. A property collection represents the set of properties that a test designer cares about under a specific circumstance.

An *application state s* for a property collection PC is a set of property/value pairs, where all the properties must be a member of PC and all values must be legally defined by the corresponding properties. We say that PC is the *context property collection* of s and that *s fits in PC.*

An application state does not necessarily have values for all properties in its context property collection. If a property p_j is missing from an application state s_i and p_j is an element of s_i's context property collection, we say that s_i *does not care about property* p_j. This means application state s_i allows p_j to have any possible value.

An *action* is a test step that a tester would perform to test the application-under-test. An *action list* is a sequence of actions. Actions or action lists can drive an application from one application state to another application state. An *application state diagram* is a state diagram that has all application states in the context of a particular property collection. These application states are nodes. Actions or action lists serve as transitions between nodes. Application state diagrams are the basis for application state generation, test case generation and coverage analysis.

Application-state-based testing is a software testing technique based on the concept of application states. It first determines all possible application states in the context of a property collection given by test designers. It then creates test cases using available actions to cover as many application states as possible. *Application-state-based coverage* is the number of application states covered by a test suite divided by the total number of possible application states in the context of a given property collection.

By adding or removing properties to or from a property collection, test designers can fine-tune the number of test cases and adjust the emphasis of coverage. A new application state diagram is generated from the new property collection. Test cases with actions relevant to these application states are automatically generated or selected to provide the best coverage.

2 SUMMARY

Application states can be used in software testing to break down test cases into manageable pieces, generate new test cases, and control the number of test cases in a test suite. They can also help test designers to conceptualize test design.