ConSequence - Model-Based Testing With State Machines and Concatenated Sequence Diagrams

Stephan Weißleder
Fraunhofer-Institute FIRST
Kekuléstraße 7, 12489 Berlin, Germany
stephan.weissleder@first.fraunhofer.de

Dehla Sokenou
GEBIT Solutions
Koenigsallee 75b, 14193 Berlin, Germany
dehla.sokenou@gebit.de

1 Motivation

Model-based testing (MBT) offers several advantages over traditional testing such as requirements formalization, which can lead to early detection of inconsistencies, automation of test design, which usually implies a significant decrease of test design costs, and reduced test maintenance costs.

In this paper, we deal with two open issues of MBT: First, coverage criteria are often used to measure test quality and to steer automatic test generation. This means that covering all specified structural elements is the main focus of the generated test suite. As a consequence, typical user behavior is seldom reflected in the test suite, and the intention of single test cases is often hard to understand for users. To solve this issue, we propose to define typical user behavior as sequences that are mapped on the behavioral model. The second issue is the poor traceability from behavioral system models to functional requirements. An intuitive approach to allow for traceability is to annotate model elements with references to requirements [1, page 131]. This approach, however, suffers from two draw-backs: The model becomes overloaded and the integration of requirements links in the system model has to be done manually. Instead, we propose to use separate models for requirements and system behavior and to link them automatically during test generation.

We present an approach to concatenate transition sequences of requirements that drives test generation by new requirements-based coverage criteria.

2 The ConSequence Approach

We focus on concatenating sequences that potentially represent functional requirements. Thus, we use the terms *typical interaction sequence* and *requirements* interchangeably for the rest of the paper.

We define relations of functional requirements and transition sequences of state machines: Each behavior specified by a functional requirement can be mapped to several state machine transition sequences depending, e.g., on the start state. If the transition sequences of different requirements overlap each other, then the corresponding requirements are also overlapping.

Mapping. The task of the mapping is to identify the contexts of the requirements in the behavioral model, i.e., to map each requirement to transition sequences in a state machine. State machine transitions are triggered by incoming events. Thus, only the incoming message events for the object described by the state machine are extracted from the sequence diagram and mapped to state machine transition sequences that are triggered by the same sequence of events.

Overlap Identification. We define overlapping transition sequences using the following terms for all transition sequences *ts1* and *ts2*: *Start state* of any ts1 is the source state of the first transition in ts1. *End state* of any ts1 is the target state of the last transition in ts1. ts1 is a *prefix/postfix* of ts2 iff ts2 starts/ends with the sequence of transitions defined in ts1 and optionally contains subsequent/preceding transitions. ts1 *is part of* ts2 iff the transitions in ts1 occur in ts2 at any place in the same order.

We define that two transition sequences *overlap* iff a) the start state of one transition sequence is equal to the end state of the other, b) there is a non-empty prefix of one transition sequence that is equal to a non-empty post-fix of the other, or c) one transition sequence is part of the other. The matching transitions that imply the overlap are called the *overlap*. The corresponding requirements are also considered overlapping if the mapped transition sequences do.

Concatenation and Test Generation. Based on the overlap relations between pairs of transition sequences, we create an *overlap graph* covering all mapped transition sequences. Two overlapping transition sequences are concatenated by creating a new sequence that first contains all transitions of the preceding transition sequence and then all transitions of the subsequent transition sequence without using the overlap of both transition sequences twice. Concatenating non-overlapping transition sequences is done via finding a path on the overlap graph from one transition sequence to the other and concatenating all contained overlapping transition sequences.

Test generation based on concatenating transition sequences consists of several steps: 1) Concatenat-

ing all required transition sequences in the desired order. 2) Concatenating the resulting transition sequence with the outgoing transition of the initial node and a transition sequence that starts at the target state of the initial state's outgoing transition. The resulting transition sequence is a connected path of the state machine that starts at the initial state and describes a possible system behavior that covers the desired transition sequences and requirements, respectively. 3) Validating the resulting path, i.e., checking that all constraints regarding guard conditions and effects on the contained transitions are valid.

2.1 Coverage Criteria

In this section, we describe coverage criteria for test generation with the ConSequence approach:

All-Requirements is satisfied if for each requirement, *at least one* of the mapped transition sequences is covered by a test case. **All-Sequences** is satisfied if for each requirement, *all* mapped transition sequences have been covered by test cases at least once. **All-Requirements-Pairs** is satisfied if for each ordered pair of requirements, *at least one* of the mapped transition sequences are covered in the order of the requirements. **All-Sequence-Pairs** is satisfied if for each ordered pair of requirements, *all* corresponding transition sequences are covered in the order of the requirements. **All-Sequence-Twice** is satisfied if for each requirement, *all* mapped transition sequences have been covered by test cases *at least twice*, which also comprises transition loops.

3 Case Study

The case study is based on models describing the behavior of an automated teller machine. One state machine describes the behavior of the system and 25 sequence diagrams describe single interaction sequences.

Results. We evaluate the effects of 1) the test suites generated with ConSequence, 2) structure-based coverage criteria with corresponding test suites generated by ParTeG [2], and 3) combining both test suites. Table 1 shows the test suite size measured as the number of test cases, the number of system calls, and the mutation score of the mutation analysis using the presented five requirements-based coverage criteria, the structure-based coverage criteria, and the combination of some of them.

As expected, All-Requirements performs worst of all coverage criteria and All-Sequences performs better than All-Requirements without much effort overhead. All-Requirements-Pairs and All-Sequence-Pairs perform even better than All-Sequences. Their execution costs, however, increase dramatically: more than 300 test cases with more than 5000 system calls for All-Requirements-Pairs and All-Sequence-Pairs calls compared to just 23 test cases with only 159 system calls for All-Sequences! The number of test goals for any of these two coverage criteria rises quadratically compared to the number of given requirements.

Coverage Criterion	Number of Test Cases	Number of System Calls	Mutation Score
All-Requirements	19	165	69/87
All-Sequences	23	159	72/87
All-Req.-Pairs	317	5411	73/87
All-Sequence-Pairs	369	6277	73/87
All-Sequence-Twice	23	440	73/87
All-States	3	15	39/87
All-Transitions	11	69	72/87
MCC	12	76	72/87
MCC + All-Sequences	35	235	74/87
MCC + All-Sequence-Pairs	381	6353	75/87
MCC + All-Sequence-Twice	35	516	75/87

Table 1: Results of mutation analysis.

We chose the coverage criteria All-States, All-Transitions, and Multiple Condition Coverage (MCC) for running test generation for the structure-oriented coverage criteria. As expected, All-States performs worst of all coverage criteria. The coverage criterion MCC detected 72 mutants but used only 12 test cases. Compared to All-Sequences, this is the same mutation score for half of the test cases.

Comparing both results, our first conclusion is that requirements-based coverage criteria can be able to reach a high mutation score and structural-based coverage criteria can achieve still good mutation scores with less effort. The next result is that there are benefits of applying both kinds of coverage criteria. For instance, combining MCC and All-Sequence-Twice detects 75 mutants with only 35 test cases. Given the mutation scores of the single test suites, we consider this improvement substantial.

4 Conclusion and Outlook

In this paper, we presented a new approach to traceability and to test generation in model-based testing based on concatenation of requirements.

There are several points to discuss. For instance, the quality of the presented test generation approach strongly depends on the quality of the given requirements. For instance, parts of the state machine may be not covered by them and, consequently, pairs of non-overlapping transition sequences cannot be concatenated using the overlap graph. In such cases, transition sequence concatenation has to be done by applying other techniques.

As future work, we plan to use the described connection of sequence diagrams and state machines for integration testing.

References

[1] M. Utting and B. Legeard. *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[2] S. Weißleder. ParTeG (Partition Test Generator). http://parteg.sourceforge.net.