Modbat: A model-based API tester for event-driven systems

C. Artho A. Biere M. Hagiya R. Potter
R. Ramler Y. Tanabe F. Weitl M. Yamamoto

Software testing executes a system under test by giving it a series of inputs and comparing the outputs to expected values. Model-based testing generates test executions from an abstract model that describes the system behavior.

Existing model-based approaches are not well-suited for event-driven or input/output-driven systems. In particular, there is a need to support non-blocking I/O operations, or operations throwing exceptions when communication is disrupted.

We present a new tool called “Modbat”, which is specialized for testing the application programming interface of systems where these issues are common.

1 Introduction

Software testing executes parts of a system under test (SUT). Various techniques have been developed to automate testing. In particular, *model-based testing* has emerged as a fast-developing field, where test cases are derived from an abstract model rather than implemented directly as code. A special model-based testing tool executes the SUT

Cyrille Artho, 産業技術総合研究所, AIST
Armin Biere, Johannes Kepler University
Masami Hagiya, 東京大学, The University of Tokyo
Richard Potter, 東京大学, The University of Tokyo
Rudolf Ramler, Softw. Competence Center Hagenberg
Yoshinori Tanabe, 国立情報学研究所, NII
Franz Weitl, 千葉大学, Chiba University
Mitsuharu Yamamoto, 千葉大学, Chiba University

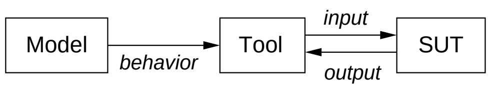

図 1 Model-based testing.

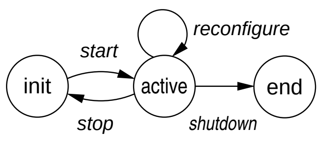

図 2 FSM modeling component behavior.

based on input/output sequences generated according to the model (see Fig. 1).

In previous work [1], we have shown that the notation used in existing tools [2] [3] [4] [7] is not well-suited to express the behavior of event-driven systems, which include databases, file systems, and cloud computing middleware. In fact, the latter is subject of an ongoing project co-funded by JSPS, which motivated and influenced this work.[*]

The systems mentioned above all depend on possibly unreliable hardware or communication links. To adequately test these systems, more support is needed to directly denote exceptions, and actions that depend on system events [1].

Our tool called *Modbat* addresses these problems. In Modbat, system behavior is described

[*] クラウドコンピューティングミドルウェアのソフトウェアモデル検査手法, JSPS kaken-hi 23240003.

using finite-state machines (see Fig. 2), which can be refined using a domain-specific language (DSL) provided by Modbat. Modbat generates *event sequences* from that model, which call the application programming interface (API) of the SUT. Results can be checked using assertions, or stored in model variables, to be used in subsequent calls.

2 Usage of Modbat

Modbat is specialized for API testing of program libraries or frameworks. It is written in Scala [5] and provides an embedded DSL [8] in which the model is specified. Any SUT that compiles to Java bytecode can be tested with Modbat. A tester uses Modbat as follows (see Fig. 3):

1. The tester defines a model using a finite-state machine that is expressed in our Scala-based DSL. The model is compiled against a library provided by Modbat. For example, a transition from Fig. 2 could be written as

```
"init" -> "active" :=
  { c = new Component; c.start }.
```

2. The tester runs Modbat against the compiled model. Modbat explores the model using a random search, executing the SUT in tandem. The sequence of transitions executed between the initial and final model states constitutes a *test run*. After each test run, the model and the SUT are reset to their initial state.
3. A failure is detected when a test run violates a property. When a failure is found, Modbat writes an error trace to a file, giving the necessary information to analyze the error. For debugging, a failed test can be replayed.

3 Conclusions and future work

Modbat is designed to test the API of state-based, event-driven systems. It addresses the needs of many projects including an ongoing project where we want to study the reliability of cloud

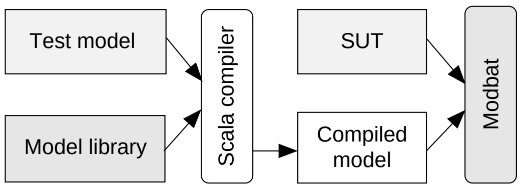

図 3 Architecture of Modbat.

computing middleware. Modbat uses a DSL based on the Scala language to succinctly express various event-driven properties.

Future work will examine the integration of tools that generate test data [3][4] with Modbat for generating an extended set of event sequences.

We also plan to implement the output of error traces in a JUnit-compatible format, so we can compare the results achieved with Modbat to the findings of a previous study comparing manually written tests to tool-based test case generation [6].

参考文献

[1] C. Artho. Separation of transitions, actions, and exceptions in model-based testing. *Post-proceedings of 12th Int. Conf. on Computer Aided Systems Theory (Eurocast 2009)*, 5717:279–286, 2009.

[2] K. Claessen and J. Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. *SIGPLAN Not.*, 35(9):268–279, 2000.

[3] T. Kitamura, T. Do, H. Ohsaki, L. Fang, and S. Yatabe. Test-case design by feature trees. In *Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2012).*, volume 7609 of *LNCS*, pages 458–473. Springer, 2012.

[4] R. Nils. Scalacheck, a powerful tool for automatic unit testing, 2012.

[5] M. Odersky, L. Spoon, and B. Venners. *Programming in Scala: A Comprehensive Step-by-step Guide.* Artima Inc., USA, 2nd edition, 2010.

[6] R. Ramler, D. Winkler, and M. Schmidt. Random test case generation and manual unit testing: Substitute or complement in retrofitting tests for legacy code? In *36th Conf. on Software Engineering and Advanced Applications*, pages 286–293. IEEE Computer Society, 2012.

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

[8] D. Wampler and A. Payne. *Programming Scala.* O'Reilly Series. O'Reilly Media, 2009.