MODEL-LEVEL AUTOMATIC TEST GENERATION FOR UML STATECHARTS*

András Tóth Dániel Varró András Pataricza
Department of Measurement and Information Systems,
Budapest University of Technology and Economics
H-1521 Budapest, Magyar tudósok körútja 2, Hungary
{varro,pataric}@mit.bme.hu

Abstract. *We present a framework for model-level testing of behavioral UML models. For automatic test generation, we use planner algorithms to deal with the complexity of UML models. Our approach is characterized by an automatic and metamodel-driven transformation from UML statecharts to a tool independent representation of planner algorithms from which the input language of concrete planner tools can be easily derived. As a result, UML based system models can be tested in an early phase of the design prior to implementation.*
Keywords: *testing, UML statecharts, planner algorithms, model transformation*

1 Introduction

Due to the complexity of IT applications there is an increasing need to ensure that the system under design will inevitably fulfill its specification. For this reason, a *verification and validation (V&V*) step based on the *use of formal methods* is indispensable still in an early phase of the design process. In V&V, we try to prove *mathematically* that all the requirements are satisfied for any possible execution of the system. Unfortunately, these techniques require the traversal of the entire state space, which is frequently impossible due to limitations of computation power.

Instead of verification, dominant design methodologies of systems engineering mainly include a phase of *testing* [1], where only a subset of all possible behavior of the system is investigated (thus we cannot guarantee that all the requirements will be met) but the automatic generation of test cases covering the most important functionality of the target application is highly reliable and automated. Despite the high-level of automation, a thorough testing phase is very expensive consuming more than 50 percent of the development costs, since many design errors sought early in the development process are captured by testing only after a complete implementation.

We present an automatic transformation from behavioral UML models to a tool independent representation of planner algorithms which automatically generate test cases for a certain test criteria afterwards. Building on the concepts reported in [2], we extend their formalism by

*This work was supported by the project OTKA T-038027 of the Hungarian Scientific Research Fund.

covering a richer subset of statechart, and introduce a metamodel of planners, which capture the mathematical structure of planners in standard way corresponding to best engineering practices.

2 A Model Driven Testing Framework

The *input* of our framework [3] is an UML statechart exported to tool-independent XMI format by a commercial UML tool. Our transformation program generates a text file as the *output,* conforming to the input language of a planner tool (we used Graphplan for our experiments).

In order to achieve independence of concrete back-end planner tools, we created a *planner metamodel*, which describes planners in the form of a UML class diagram in order to present the abstract mathematical definition in a visual way that can be easily understood by systems engineers as well. As the planner model is represented in such a tool independent format, our framework can be ported to various planner tools without rewriting the entire transformation.

For our transformation, we conceptually follow the guidelines of [2]. However, we extended the capabilities of the transformation for handling *shallow history* and *deep history states*. Our framework does not handle the automatic transformation of test criterions into initial and goal states, and the back-annotation of test vectors. In the future, we plan to exploit the use of standard UML extensions (profiles) to provide a suitable solution for the problem.

Finally, planner tools automatically generate test cases for a given test criterion by finding a feasible plan that leads from the initial state to the goal state (representing the test criterion). Typical test criterions include testing each transition, each state, or each execution path (up to a certain depth) at least once.

3 Conclusions

As a result, UML designs can be tested and design flaws can be detected in the modeling phase of the development process prior to any implementation activities saving a considerable amount of cost. Even though our framework is still in an early stage, our transformation program (written in Java) can already handle a rich subset of the UML statechart formalism by generating automatically a planner model for the *Graphplan* tool. In order to keep our methodology open for additional back-end planner tools, we constructed a planner metamodel to provide a high-level representation of the mathematical definition.

Future work will include (i) the handling of various UML diagrams in an integrated and consistent test environment (ii) the automatic derivation of test criteria (*initial* and *goal* states) from UML representations, (iii) the back-annotation of test results to UML models, and (iv) the exploitation of hierarchical planners as the back-end tools that provide a closer and optimized fit to the hierarchical nature of statecharts.

References

[1] Binder, R.: Testing Object-Oriented Systems. New Jersey (USA), Prentice Hall 1999.

[2] Fröhlich, P., Link, J.: Automated Test Case Generation from Dynamic Models. In: Bertino, E. (Ed.): Proceedings of the ECOOP 2000, 14th European Conference on Object-Oriented Programming, Springer-Verlag Berlin Heidelberg 2000, pp: 472-491

[3] Tóth, A., Varró, D., Pataricza A.: Model Level Automatic Test Generation for UML Statecharts. Technical report. Dept. of Measurement and Inf. Systems, Budapest Univ. of Technology and Economics, January 2003.