Optimized Framework for Model-Based Testing of Automotive Distributed Systems

Lukáš KREJČÍ[1],

[1] Dept. of Measurement, Czech Technical University, Technická 2, 166 27 Praha, Czech Republic

krejclu6@fel.cvut.cz

Abstract. *The paper presents an optimized framework for model-based testing of automotive distributed system.*

The proposed framework, envisioned for integration into existing, currently developed model-based testing tool, utilizes several test sequences generation strategies combined with automated testing priority assignment in order to reduce the testing procedure's mean time to error-detection.

It is shown, that the presented framework's advantages, such as the reduction of the testing time, as well as low requirements for the testing operators' insight, are valuable for the automotive distributed systems testing process.

Keywords

Model-based testing, automotive systems, timed automata, UPPAAL, classifiers.

1. Introduction

The Aim of this paper is to present new optimized framework for model-based testing of automotive distributed systems.

At the presence, the testing process used in the automotive industry is composed from three distinct parts. The compulsory standard tests, the specialized test-cases and integration testing. The compulsory standard tests are given by various international, national or organization safety standards and hence are unavoidable. The prearranged specialized test-cases are often based on the organization know-how and are therefore desirable, as they can uncover specific corner-case faults. Purpose of the integration testing is to test the automotive system as a whole in order to discover possible errors caused by distributed systems interconnection or improper user interaction. Currently, the integration testing is usually done manually by testing operators and thus can be automated using the model-based testing principles.

The model-based testing is a popular approach of automated testing, which utilizes model of a tested system in order to generate a testing sequence or drive the testing procedure in real time. Since the automotive systems are reactive and real-time, it was necessary to develop a suitable testing tool.

2. Background

The development of the model-based testing tool, called TASysTest, is described in [1]. Since automotive systems are real-time, this tool utilizes a modeling language based on the systems of timed automata, developed by UPPAAL team [3].

UPPAAL team used existing theory of timed automata, described in [2], with discrete variables and synchronization capabilities. As defined by UPPAAL, timed automata systems are a powerful tool for the model-checking verification of real-time systems. Such a language allows to describe the modeled system as a set of Finite State Machines bound by system of transition labels and automata variables and constants. The timed automata systems are stored as XML-formatted files with the standard schema defined by UPPAAL team.

Despite existence of various test sequence generation strategies, the TASysTest tools is currently only able to generate pseudo-random test sequences. Since this approach can be suboptimal in several cases, new optimized testing framework for this tool was designed in order to reduce the mean time required for the error-detection

3. Optimization Approaches

There are several promising approaches for the reduction of the error-detection mean time required by the model-based testing procedure. The optimization techniques can operate on the model level and their purpose is a reduction of model's state space. Alternative approaches operate directly on the sequence generation level.

3.1 Model-Level Optimizations

The test sequence generation procedure time complexity is undoubtedly dependent on the size of tested system model. Hence, the procedure can be possibly optimized by the model size reduction.

As the model typically describes a collection of parallel processes in the modelled system, it is common that some of them are identical. Thus, if the rest of the system is independent on the number of these identical processes and if these processes do not access any shared variable, they can be, apart from one, omitted from the procedure. Nevertheless, this reduction is not feasible, if these processes execute distinct operations on the transitions, as such processes might control different hardware devices and therefore ignoring them could possibly result in the error detection failure.

Furthermore, it is also possible to reduce the total size of model's state space by a technique called Partial Order Reduction, described i.e. in [4]. This reduction is frequently used in the model-checking formal verification and is based upon fact that in some cases the order of executed operations is irrelevant with respect to the verified property. Fig. 1 shows a simple example of such reduction.

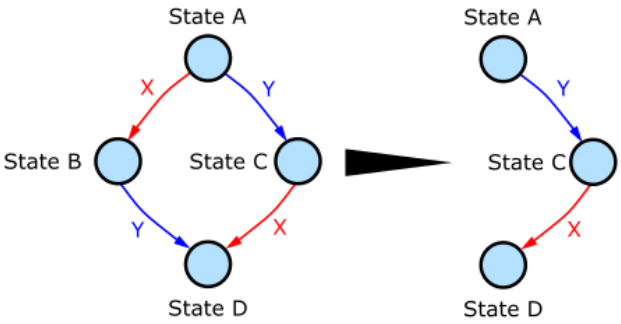

Fig. 1. The Partial Order Reduction example.

State B is in this example omitted from the reduced state space as the order of transitions X and Y, as well as state B itself, does not influence the verified property. However, in the case of model-testing, where no explicit property is being verified, usage of this approach could be problematic. If tested system contains a fault, which occurrence is invoked only by exact order of transitions, such reduction could result in detection failure. Because the overhead required to bypass described flaw would significantly increase overall time-complexity, this approach is not viable.

Since the model state-space reduction techniques can cause error detection failure or require significant overhead, the proposed framework does not utilize them and use sequence generation approaches as a more viable alternative.

3.2 Sequence Generation Approaches

As already mentioned, the test sequence generation can be driven according to various strategies. Examples of popular approaches are pseudo-random sequence generation, state or transition coverage maximization and selective prioritization. The proposed framework uses all of these approaches.

Since the developed tool is envisioned for testing of the automotive comfort systems, the inputs of the typical system under test are continuously affected by human users. Thus, the pseudo-random sequence generation is sufficient for simulation of random user-generated inputs and for that reason it is currently utilized by the tool. The proposed framework uses this strategy as a foundation and extends it by combining it with additional approaches.

As its name implies, the state (resp. transition) coverage maximization is a strategy that generates a sequence with complete state (resp. transition) coverage according to the system model. As the framework is designed for the automotive systems, where maximal state (resp. transition) coverage is appropriate due to the safety reasons, utilization of this strategy is highly desirable. However, this strategy is particularly hard to implement, as the state coverage maximization problem is equivalent to the Hamiltonian Path problem, which is NP-complete. In order to maintain reasonably low time complexity, this approach can be realized as an approximation heuristic. Because the resulting sequence should be pseudo-random, one of potential solutions is based on the modification of the transition probabilities used in the sequence generation in a way that lowers the selection probability of the previously selected states.

Last strategy used by the framework is the selective prioritization. In several scenarios, parts of tested system (i.e. states and transitions) may be labeled, possibly by the testing operator or model designer, with a priority of interest. If such labeling is available, it is used by this particular strategy in order to generate a test sequence with increased coverage of the prioritized parts of the tested system. This can be straightforwardly achieved by the Nearest-Neighbor heuristics. Though, the generated sequence should be pseudo-random. Consequently, one of possible solutions is based on the weighted random walk on the state space graph using the priority labels as edge weights.

Considering the scenario with the automotive systems, the states and transitions labeled by a high-priority labels will be in typical case such states and transitions, which failure would result in fatal safety violation or would negatively affect the overall user experience. Moreover, having the correct high-priority labels on states and transitions with higher error-rate can significantly reduce the time required for error detection.

Still, proper manual configuration of priorities of such states and transitions requires a significant insight to the tested system. Hence, the proposed framework encases a procedure called Automated Pinpointing that is able to automatically assign priorities to the states and transitions of the tested system according to the information given stored in the model.

3.3 Automated Pinpointing

As declared earlier, the Automated Pinpointing is a procedure, which obtains the priority labels for the selective prioritization strategy automatically according to the model. In order to accomplish this, this procedure

utilizes several classifiers in various forms (i.e. artificial neural network). However, regardless of the used classifiers types, each of them requires some supplemental information about the modeled system. With the aim of having these supplementary facts available, the modeling language utilized by the developed tool needs to be extended with an ability to store the state, transition and template extra-data.

The extra-data are stored in the comment sections of the state, transitions and template and for example can provide following additional:

- Safety index indicating, how severe failure can the state, transition or template functionality cause.
- User experience index indicating, how severe user experience impairment can the state, transition or template functionality cause.
- Vulnerability index indicating, how failure-vulnerable is the state, transition or template functionality.
- Functionalities correlation indicating, how intertwined are distinct functionalities of distinct states, transitions and templates.

The rationale behind the safety, user experience and vulnerability indexes is simple. The higher each index is, the more worthwhile is to test related state, transition or template promptly.

The functionalities correlation extra-data can be especially useful in scenario, where several functionalities are linked together (i.e. through usage of the same codebase). For example, let functionalities A and B share 90 % of the codebase. Then, if transitions linked to the functionality A are faulty and thus have high priority of testing, then the transitions linked to the functionality B should also have high priority of testing, as the error incidence in the source code of functionality A would probably invoke similar error in functionality B.

As already mentioned, the proposed framework is able to use multiple classifiers at once. These classifiers are separated into two following categories:

- Context-insensitive classifiers that work on the level of separate states and transitions. That means these classifiers take a single state or transition extra-data as an input.
- Context-sensitive classifiers that work on the level of a template or even whole model. That means these classifiers take all extra-data from the template or entire model, as well as the template or model structure as an input.

Results obtained from different classifiers are compared and their potential incongruence is reported to the testing operator, as it may indicate possible extra-data misconfiguration or occurrence of unspecific anomalous conditions within model.

The training data necessary for the proper operation of the context-insensitive classifiers can be effortlessly obtained by running multiple complete tests and labelling the model manually by the testing operator afterwards. From the other hand, acquisition of the training data for the context-sensitive classifiers can be problematic, since each sample from the training set must necessarily contain an entire template or model structure. Thus, each template or model has to have its own training data sets.

The framework makes use of the outputs of all utilized classifiers in order to label the model with priority labels. As already revealed, possible incongruence in the classifiers outputs is used to detect inconclusive results caused by the wrong extra-data or anomalous conditions and is therefore reported to the testing operator.

4. Framework Structure

The proposed framework utilizes mentioned principles in a form of a pipelined processing. Fig. 2 depicts its structure.

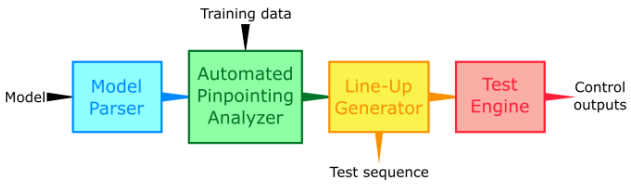

Fig. 2. The framework pipeline structure.

The framework's pipeline consists of following parts:

- The Model Parser, which loads the model from its XML-formatted file.
- The Automated Pinpointing Analyzer, which encases the previously described procedure of Automated Pinpointing. This block takes parsed model as an input and outputs the model enriched with priority and incongruence labels.
- The Line-Up Generator, which encases the test sequence generation strategies described in previous sections. This block takes enriched model as an input and outputs a script for the testing environment and the control outputs for the Test Processor (if it is present in the chain).
- The Test Engine, which is included in the chain only optionally (i.e. for the online testing). This block takes the control inputs and directly executes given operations on the tested system according to them.

Described pipelined modular architecture allows seamless integration of the proposed framework into the developed tool.

5. Conclusions and Future Work

In this paper, the optimized framework for the model-testing of an automotive distributed system is presented. The described framework combines the pseudo-random test sequences generation with a state and transition coverage maximization and selective prioritization strategies in order to generate such testing sequences, which reduce the required mean time of the error-detection. In addition, the presented framework utilizes a procedure for automatic priorities assignment, which is able to use information provided by the model for pinpointing the parts of the model, which are most worthwhile to be tested. Furthermore, the framework's pipelined structure allowing easy integration into testing systems was presented.

At the presence, the framework is being implemented into TASysTest tool that currently utilizes only simple pseudo-random test sequences generation. As this tool uses the modeling language based on the timed automata, the presented framework uses it as well. However, it is possible to adapt the framework for other graphical modeling languages.

The future research will primarily consist of finding the most suitable way of the extra-data and training data storage. Afterwards, the research will continue by analysis of various classifiers and sequence generation heuristics in order to find the most feasible ones, as well as the most reasonable training strategies. Additional research will be focused on the issues of automatic obtainment of the model extra-data.

The presented framework, as well as entire TASysTest tool will be tested, thanks to the co-operation with Škoda Auto a. s., on the real automotive systems developed and manufactured by this company.

Acknowledgements

Research described in the paper was supervised by doc. Ing. Jiří Novák, PhD. and supported by the Czech Student Grant Agency under grant no. SGS16/171/OHK3/2T/13.

References

[1] GRUS, T. Implementation of Integration Testing Test Cases Generation Tool. *Master's Thesis, CTU in Prague, FEE,* 2014.

[2] ALUR, R., DILL, D. L. Theory of timed automata. *Theoretical Computer Science 126*, 1994, p. 183-235.

[3] BEHRMANN, G., DAVID, A., LARSEN, K. G. A tutorial on UPPAAL. *In proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT'04)*.

[4] CLARKE, E. M., GRUMBERG, O., MINEA, M., PELED, D. State space reduction using partial order techniques. 1998.

About Authors...

Lukáš KREJČÍ was born in Prague in 1990. He received his bachelor's and master's degree from the CTU in Prague, FEE in 2012 and 2014 respectively. Since February 2015, he has been studying the branch of doctoral studies called Measurement and Instrumentation at the Department of Measurement on CTU in Prague, FEE.