An Energy-aware Mutation Testing Framework for EAST-ADL Architectural Models

Raluca Marinescu*, Predrag Filipovikj*, Eduard Enoiu*, Jonatan Larsson†, and Cristina Seceleanu*

*{first.last}@mdh.se, †jln13010@student.mdh.se
Mälardalen University, Västerås, Sweden.

Background and Motivation. Early design artifacts of embedded systems, such as architectural models, represent convenient abstractions for reasoning about a system's structure and functionality. One such example is the Electronic Architecture and Software Tools-Architecture Description Language (EAST-ADL) [3], a domain-specific architectural language that targets the automotive industry. EAST-ADL is used to represent both hardware and software elements, as well as related extra-functional information (e.g., timing properties, triggering information, resource consumption). Testing architectural models [1] is an important activity in engineering large-scale industrial systems, which sparks a growing research interest. Modern embedded systems, such as autonomous vehicles and robots, have low-energy computing demands, making testing for energy usage increasingly important. Nevertheless, testing resource-aware properties of architectural models has received less attention than the functional testing of such models. In our previous work [11], we have outlined a method for testing energy consumption in embedded systems using manually created faults based on statistical model checking of a priced formal system model. In this paper, we extend our previous work by showing how mutation testing [6] can be used to generate and select test cases based on the concept of energy-aware mutants– small syntactic modifications in the architectural model, intended to mimic real energy faults. Test cases that can distinguish a certain behavior from its mutations are sensitive to changes in the model, and hence considered to be good at detecting faults. The main contributions of this paper are: (i) an approach for creating energy-related mutants for EAST-ADL architectural models, (ii) a method for overcoming the equivalent mutant problem [9] (i.e., the problem of finding a test case which can distinguish the observable behavior of a mutant from the original one), (iii) a test generation approach based on UPPAAL Statistical Model Checker (SMC) [4], and (iv) a test selection criteria based on mutation analysis using our MATS tool[1] [8].

Proposed Framework. In this section, we describe our mutation testing framework that uses energy consumption goals to automatically select test suites based on random system simulations. The framework is enabled by transforming the EAST-ADL model into a network of priced timed automata (PTA) [10]. It is composed of several steps, mirrored in Figure 1:

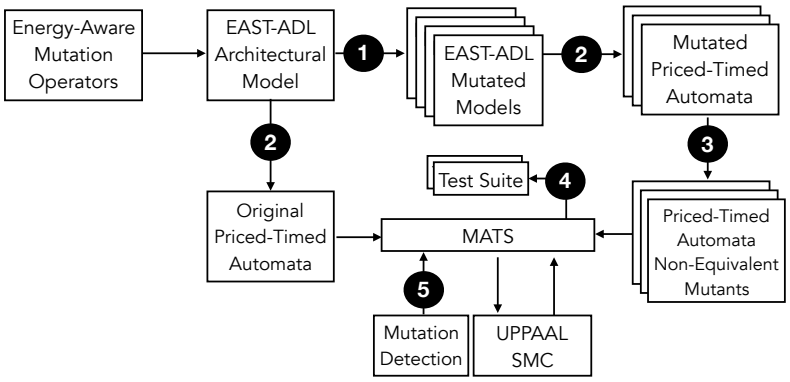

Figure 1: Overview of the energy-aware mutation testing framework.

[1] MATS is an open source software and is available at https://github.com/JLN93/MATS-Tool

(1) Energy-Aware Mutant Generation. The consumption of a resource r for an East-adl component represents the accumulated resource usage up to some point in time. Based on this assumption, resources can be classified as continuous or discrete [13] . In this paper, we focus on energy consumption, which is a continuous resource assumed to evolve linearly in time ($r(t) = n \times t$, where $n \in \mathbb{N}$ and t is the elapsed time). Since in East-adl the resource usage annotation is provided at component level, we define the total energy consumption of the system as $r_{total}(t) = \sum_{i=1}^{m} r_i(t)$, where m is the number of functional components. In mutation testing, faults are injected based on a predefined set of mutation operators. Ideally, such mutants should represent commonly-occurring faults, but to the best of our knowledge, there is no previous work on identification of resource-related faults at the architectural level. Given this, we propose a set of mutation operators applied on: (i) the East-adl resource annotation (Energy Consumption Replacement Operator (ero)), (ii) the timing behavior of an East-adl component (Period Replacement Operator (pro), Execution Time Replacement Operator (ero)), and (iii) the structure of the functional architecture (Component Removal Operator (cro), Component Insertion Operator (cio), and Triggering pattern Replacement Operator (tro)). These mutation operators are systematically applied to the entire East-adl model, resulting in a set of energy-aware mutants, each simulating one syntactic model change.

(2) East-adl to PTA. In order to use Uppaal SMC for test case generation, we transform the East-adl model (with energy consumption annotations) into a PTA model. Each East-adl component is automatically transformed into a network of two PTA: an *interface* automaton, which encodes the interface of the component, and a *behavior* automaton, used to model the component's internal behavior. The triggering of each component, timing information, as well as the resource annotations, are included in the interface PTA. The energy consumption starts at the moment data is read from the input ports until the component writes the data to the output ports. This means that the energy consumed by each component increases with the execution time, modeled as a cost "c" in PTA ($c(t) = n_c \times t$, where $n_c \in \mathbb{N}$ is the rate of consumption over time t), but not when the component is idle ($c'(t) = 0$). A monitor automaton is added to compute the energy used by the system based on the energy consumed by each component. For more details, we refer the reader to our previous work [10].

(3) Detection of Equivalent Mutants. Let O_n be the original PTA model and M_m be a mutant of the former obtained by applying a predefined mutant operator. We say that models O_n and M_m are *equivalent* if there is no input parameter for which the difference in energy consumption of the models exceeds some predefined threshold within some bounded time limit. Otherwise, there is a valuation of the input parameters for which the mutant can be detected. From the above, it is obvious that the *mutant equivalence check* can be reduced to a *satisfiability problem* [5]. Let $\Phi = \{\varphi_1, \varphi_2, ..., \varphi_k\}$ and $\Psi = \{\psi_1, \psi_2, ..., \psi_l\}$ denote the set of constraints for the energy consumption of c_n and c_m in O_n and M_m, respectively. The mutant M_m is not equivalent to O_n if the following conjunction evaluates to true: $\exists l_k . \bigwedge_{i=1}^{k} \varphi_i(l_k) \wedge \bigwedge_{j=1}^{l} \psi_j(l_k) \wedge (|c_n - c_m| \geq threshold)$, where l_k is an arbitrary input parameter. Reducing the mutant equivalence checking to a satisfiability problem has been considered in other frameworks. Brillout et. al [2] exploits a similar technique for functional testing of Simulink models. In comparison, our framework is specifically tailored for resource-aware mutation testing of architectural models. The HiLiTe tool [12] is another example of using an SMT solver for improving test case generation for large-scale complex and constrained models. Given the fact that the energy consumption is of continuous nature, we have to resort to a specialized type of SMT-solving suitable for hybrid systems [7].

(4) Test Suite Generation. We create executable test cases using the Mats tool [8], by extracting the input parameters and the energy values at predefined time points from the simulation traces produced by Uppaal SMC. Each test input is a vector of signals where the time-dependent behavior of the model is executed using an ordered sequence of signals. Mats uses Uppaal SMC for obtaining simulation traces over a predefined number of runs of the system model. A simulation can be formulated as the property: *simulate* $n[bound]\{E_1, .., E_k\}$ in Uppaal SMC, where n is the number of simulations to be performed, *bound* is the time bound on the simulations, and $E_1, .., E_k$ are the monitored expressions. Each test case is executed on both the original model and its mutated counterpart. In order to minimize the final set of test cases we remove the test cases not contributing to the mutation score [8].

(5) Mutant Detection Criteria. We show how to detect energy mutants using the Mats tool. A

mutant is detected by a test suite if the energy signal diverges drastically at certain time points from the expected values (e.g, substantial energy deviations). To measure the mutant-revealing ability of a test suite, we use a quantitative measure of a mutant detection oracle. Let a test case T be generated for a mutated model M, and let $E_M = E_{M1}, ..., E_{MN}$ be the set of energy signals obtained by running M for the test inputs in T and sampled at N time points. Let $E_O = E_{O1}, ..., E_{ON}$ be the corresponding expected energy signals. We use a threshold to check if the distance between each value of E_O and E_M at each time point is larger than this threshold. If there is at least one energy value in E_M for which the distance is larger than the expected threshold then we consider the mutant M detected.

Conclusions and Future Work. In this paper we have outlined a framework for energy-aware mutation testing of EAST-ADL architectural models. Given the large number of energy mutations we aim to reduce the number of equivalent mutants by employing an SMT-solver. In addition, this framework selects test suites contributing to the overall mutation score using UPPAAL SMC and MATS. Future work aims to apply this framework on an industrial case to expose its strengths as well as limitations both in terms of test efficiency and effectiveness.

Acknowledgements. The authors of this work are supported by the following projects: Swedish Governmental Agency for Innovation Systems (VINNOVA) and ECSEL (EU's Horizon 2020) under grant agreement No 737494, VINNOVA VeriSpec project 2013-01299, Swedish Research Council (VR) project "Adequacy-based testing of extra functional properties of embedded systems" and the Swedish Knowledge Foundation (KKS) project DPAC – "Dependable Platforms for Autonomous systems and Control".

References

[1] Antonia Bertolino, Paola Inverardi, and Henry Muccini. Software architecture-based analysis and testing: a look into achievements and future challenges. *Computing*, 95(8):633–648, 2013.

[2] Angelo Brillout, Nannan He, Michele Mazzucchi, Daniel Kroening, Mitra Purandare, Philipp Rümmer, and Georg Weissenbacher. Mutation-based test case generation for Simulink models. In *Formal Methods for Components and Objects*, pages 208–227. Springer, 2010.

[3] Philippe Cuenot, Patrick Frey, Rolf Johansson, Henrik Lönn, Papadopoulos, et al. The EAST-ADL architecture description language for automotive embedded software. In *Model-based engineering of embedded real-time systems*, pages 297–307. Springer, 2010.

[4] Alexandre David, Kim Larsen, Axel Legay, Marius Mikučionis, Danny Poulsen, Jonas Van Vliet, and Zheng Wang. Statistical model checking for networks of priced timed automata. *Formal Modeling and Analysis of Timed Systems*, pages 80–96, 2011.

[5] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.

[6] Richard A DeMillo, Richard J Lipton, and Frederick G Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41, 1978.

[7] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dreach: δ-reachability analysis for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015.

[8] Jonatan Larsson. Automatic Test Generation and Mutation Analysis using UPPAAL SMC. In *Bachelor of Science Thesis Report*. MDH Diva, 2017.

[9] Lech Madeyski, Wojciech Orzeszyna, Richard Torkar, and Mariusz Jozala. Overcoming the equivalent mutant problem: A systematic literature review and a comparative experiment of second order mutation. *IEEE Transactions on Software Engineering*, 40(1):23–42, 2014.

[10] Raluca Marinescu, Eduard Enoiu, and Cristina Seceleanu. Statistical Analysis of Resource Usage of Embedded Systems Modeled in EAST-ADL. In *VLSI Symposium*, pages 380–385. IEEE, 2015.

[11] Raluca Marinescu, Eduard Enoiu, Cristina Seceleanu, and Daniel Sundmark. Automatic Test Generation for Energy Consumption of Embedded Systems Modeled in EAST-ADL. In *International Conference on Software Testing, Verification and Validation Workshops*, pages 69–76. IEEE, 2017.

[12] Hao Ren, Devesh Bhatt, and Jan Hvozdovic. Improving an Industrial Test Generation Tool Using SMT Solver. In *NASA Formal Methods Symposium*, pages 100–106. Springer, 2016.

[13] Cristina Seceleanu, Aneta Vulgarakis, and Paul Pettersson. REMES: A Resource Model for Embedded Systems. In *International Conference on Engineering of Complex Computer Systems*. IEEE, 2009.