Statistical Analysis of Resource Usage of Embedded Systems Modeled in EAST-ADL

Raluca Marinescu*, Eduard Paul Enoiu*, Cristina Seceleanu*
*Mälardalen University, Västerås, Sweden, <firstname.lastname>@mdh.se

***Abstract*—The growing complexity of modern automotive embedded systems requires new techniques for model-based design that take into consideration both software and hardware constraints, and enable verification at early stages of development. In this context, EAST-ADL has been developed as a domain-specific language dedicated to modeling functional-, software-, and hardware- architecture of automotive systems. This language offers convenient abstractions that support modeling of function, as well as relevant extra-functional properties, like timing and resource usage. These features make it a suitable framework for reasoning about the system's behavior. By providing formal semantics to the EAST-ADL language, as a network of priced timed automata, it becomes possible to reason about feasibility and worst-case resource consumption of the embedded components. In this paper, we show how to analyze such embedded systems modeled in EAST-ADL by using statistical model-checking. We report our experience from applying this approach to an industrial Brake-by-Wire system prototype.**

***Index Terms*—embedded systems, EAST-ADL, statistical model-checking, resource usage.**

I. Introduction

As modern embedded systems contain an increasing number of software and hardware features, significant problems [1] arise in the integration of new sub-systems due to the emergent behavior of the whole system. It is not enough to have correct sub-systems, they must also be properly integrated in a system that meets both its functional and extra-functional requirements, such as real-time performance and resource consumption.

Integration and analysis of extra-functional requirements have been tackled in several domains [2], [3], [4]. Consider, for instance, the automotive domain where a change to a software sub-system or the substitution of a hardware component can affect the system's latency, energy and memory utilization. The modus operandi in industrial practice [1] shows that analyzing resource usage attributes is often done ad-hoc by industrial engineers, by using disjoint specifications. This trend could result in a costly and risky integration phase during the development of embedded systems.

In this context, architectural models that can be introduced earlier in the development process provide a holistic system description that captures the structure of the system, as well as related extra-functional information, e.g., timing properties, triggering annotations, and resource annotations. EAST-ADL [5] is a domain-specific language for modeling software and hardware specifications in the automotive domain. The approach supports modeling of resource usage, which enables the early analysis of extra-functional requirements like feasibility and worst-case resource consumption of components.

In this paper, we show how efficient verification techniques, like statistical model-checking, can be applied on high-level design artifacts, such as EAST-ADL models, to provide early information on the resource consumption of an automotive embedded system. To achieve this, we propose a methodology that uses our existing tool VITAL [6], in which it is possible to obtain formal models in form of networks of timed automata [7] from EAST-ADL architectural models. Here, we extend such networks with resource annotations based on the information provided in the architectural model, thus creating networks of priced timed automata, which are extensions of timed automata with costs. Consequently, we can employ UPPAAL SMC [8], the UPPAAL extension for statistical model-checking, to analyze the resource usage of automotive systems described in EAST-ADL. The results of the analysis, which we apply on a Brake-by-Wire industrial prototype, provide valuable information on the system's resource-usage prior to the actual implementation.

This paper is structured as follows. We start by briefly presenting the EAST-ADL architectural language and the tools involved in the verification process (i.e., the VITAL tool and the UPPAAL SMC model-checker) in Section II. Our resource usage analysis methodology is described in Section III. In Section IV we show how EAST-ADL functions can be analyzed in terms of resource-usage. Next, in Section V we introduce our case-study, the Brake-by-Wire (BBW) system, and we show how the proposed methodology is applied on this industrial system. We end the paper by discussing related works in Section VI, and by presenting our conclusions in Section VII.

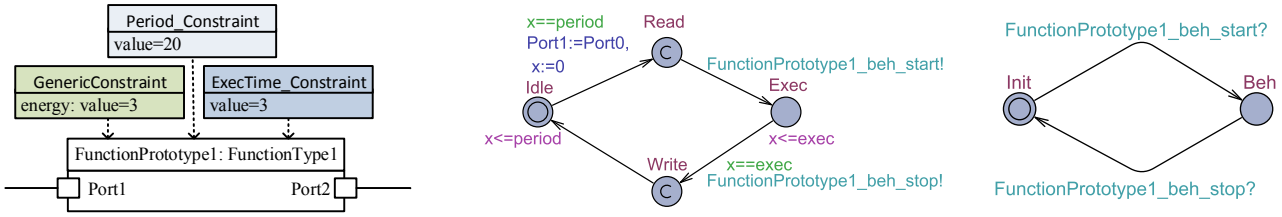
II. Preliminaries

In this section, we present the tools and frameworks used in our analysis methodology.

A. EAST-ADL Language

EAST-ADL [5] is an AUTOSAR[1][9] compatible architectural description language dedicated to the development of automotive embedded systems. The functionality of the system is defined at four levels of abstraction, as follows: (i) the *Vehicle Level*, the highest level of abstraction, describes the electronic features as they are perceived externally, (ii) the

[1]AUTOSAR stands for AUTomotive Open System ARchitecture developed by manufactures as a standard in the automotive domain.



(a) A generic EAST-ADL *FunctionPrototype*.

(b) The interface timed automaton.

(c) Behavior template.

Fig. 1: The ViTAL transformation.

Analysis Level provides an abstract functional representation of the architecture, (iii) the *Design Level* provides a detailed functional representation of the architecture, together with the allocation of these elements onto the hardware platform, and (iv) the *Implementation Level* provides the implementation of the system using AUTOSAR elements.

At each abstraction level, the system model uses components, each a *FunctionType*, which describe the functional elements of the system. The *FunctionType* has: (i) ports that receive and provide data, respectively, (ii) a trigger, either time-based or event-based, and (iii) an associated behavior. Each of these components is instantiated as one or more of type *FunctionPrototype*, which are connected to provide the system model. The execution of each *FunctionPrototype* is based on the "*read-execute-write*" semantics, and the associated behavior can be defined using different notations and tools (e.g., Simulink or UPPAAL PORT timed automata [10]). The model can be extended with a *GenericConstraint* annotation, which allows the system designer to specify various extra-functional properties, such as energy consumption or memory utilization.

B. The ViTAL tool

VITAL [6] is an analysis tool that enables formal verification of EAST-ADL models based on model-checking techniques. To achieve this, the tool provides a transformation from an architectural model to: (i) UPPAAL PORT timed automata (for component-based verification with the UPPAAL PORT model-checker) [10], and (ii) UPPAAL timed automata (for verification with the UPPAAL model-checker) [6]. In this paper, we will use the latter formalism. Further information on timed automata theory can be found elsewhere [7].

To formally verify that the EAST-ADL model meets its requirements, we represent the architectural elements as UPPAAL timed automata, by an automatic transformation within VITAL. Each *FunctionPrototype* (see Figure 1a) is transformed into a network of two synchronized automata: (i) an interface timed automaton (Figure 1b), dedicated to the elements of the EAST-ADL component interface, and (ii) a behavior timed automaton (Figure 1c), dedicated to the behavior of the EAST-ADL component. For more details we refer the reader to our previous work [6].

The result of this transformation is a network of timed automata that can be analyzed with the UPPAAL model-checker. In this paper, we extend this transformation to also include resource annotations.

C. UPPAAL SMC

UPPAAL SMC [8] is an extension of UPPAAL that enables the analysis of different performance properties of networks of priced timed automata with stochastic semantics. Statistical model-checking generates stochastic simulations to estimate probabilities and probability distributions over time with given confidence levels, so the technique scales better than exact symbolic model-checking.

Priced timed automata (PTA) are extensions of timed automata with cost variables that can evolve at integer rates (also $\neq 1$) and are used in this paper to capture the resource usage. The resource usage is modeled via a function $P : (L \cup E) \rightarrow \mathbb{N}$, where L is a finite set of locations and E is the set of edges of the PTA model, which assigns costs to both locations and edges. A network of PTA (NPTA) can be expressed as a composition of n PTA over clocks and actions; the PTA synchronize on send-receive actions (i.e., *send* b! is complementary to *receive* b?) and can use shared variables in guards (boolean conditions that enable the execution of edges).

UPPAAL SMC uses an extension of WMTL [11] to verify properties like:

- *Hypothesis testing*: check if the probability to reach state ϕ within cost $x \leq C$ is greater or equal to a certain threshold p ($Pr(\Diamond_{x \leq C} \phi) \geq p$),
- *Probability evaluation*: calculate the probability $Pr(\Diamond_{x \leq C} \phi)$ for a given NPTA,
- *Probability comparison*: is $P(\Diamond_{x \leq C} \phi_1) > P(\Diamond_{y \leq D} \phi_2)$?

III. METHODOLOGY OVERVIEW OF RESOURCE ANALYSIS

In this section, we propose an approach to resource usage analysis for EAST-ADL with UPPAAL SMC. The methodology presented in this paper is tailored to EAST-ADL, and it contains the following steps (also mirrored in Figure 2):

1) *Model Transformation.* To analyze the EAST-ADL model, we map it to a finite-state formal model suitable for model-checking. For this, we use VITAL to automatically obtain a network of timed automata (see Section II-B).
2) *Resource Annotation.* We annotate the resulting model such that the resource usage information in the EAST-ADL model (as *GenericConstraint*) is expressed in the PTA formalism.
3) *Requirement Formalization.* We formalize the extra-functional properties such that they can be verified with the UPPAAL SMC model-checker.

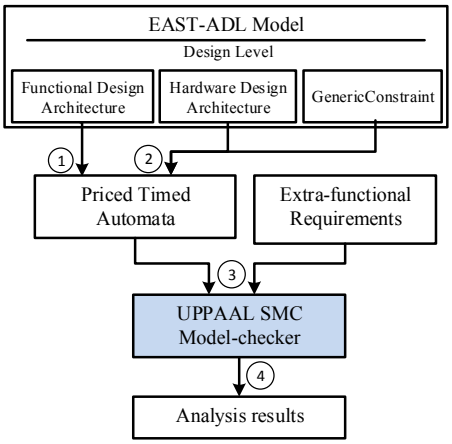


Fig. 2: Methodology overview.

4) *Analysis Results Generation.* We use UPPAAL SMC to generate analysis results. This model-checker requires as input the resulting model of Step 1 and 2, and the properties expressed in Step 3. Currently, UPPAAL SMC supports queries related to the stochastic interpretation of PTA including the visualization of expressions along the simulated runs.

While UPPAAL SMC is a viable tool for statistical model-checking, it is not directly tailored to analyzing resource usage in EAST-ADL. We demonstrate our solution to this, by discussing the above steps in the following section.

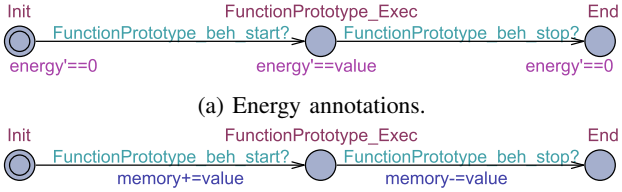
IV. Analyzing Resource-usage of EAST-ADL Functions

A *GenericConstraint* allows the EAST-ADL model to be annotated with different types of resources, like *memoryConsumption*, *powerConsumption*, *weight*, *developmentCost*, etc. Resources have different properties that impact the model and the resource analysis. The consumption of a resource c is the accumulated resource usage up to some point in time, calculated based on the rate of consumption over time c'. Based on this, resources can be classified [3] as: (i) continuous ($c' = n$, with $n \in \mathbb{Z}-\{\infty\}$), and (ii) discrete ($c' = 0$ or $c' = \infty$).

In this paper we focus on two types of EAST-ADL resource usage: *energyConsumption*, of a continuous resource, and *memoryConsumption*, of a discrete resource.

A. The Priced Timed Automata Model

To be able to analyze the resource usage of continuous and discrete resources of systems modeled in EAST-ADL, one needs to create the corresponding formal model as a network of PTA. For this, we use the VITAL tool, which provides automatic transformation from the EAST-ADL model to a timed automata model. We extend the resulting formal model with the corresponding resource annotations, to obtain the PTA model that can be used by UPPAAL SMC. Concretely, we extend the timed automata network with a monitor automaton that contains all the resource annotations of the EAST-ADL model, including energy consumption and memory usage.



(a) Energy annotations.

(b) Memory annotations.

Fig. 3: The resource annotations.

The monitor is a loop-free PTA that follows the execution of the system, which is achieved through the already existing synchronization channels *FunctionPrototype_beh_start* and *FunctionPrototype_beh_stop*. Assuming an architectural model consisting of only one *FunctionPrototype*, we annotate the monitor as follows:

- With a continuous resource, that is, energy, whose consumption is increasing with time and, in our case, is consumed in the *FunctionPrototype_Exec* location. The rate of consumption is $energy' == value$ (see Figure 3a), where $value$ is provided in the EAST-ADL model;
- With a discrete resource, that is, static memory, which is used instantaneously, so its usage does not depend on time. We encode the memory allocation via the edge assignment *memory+=value*, and memory deallocation via the edge assignment *memory-=value* (see Figure 3b).

B. Resource Analysis

In the following section, we present four types of resource analysis performed on EAST-ADL, using our methodology.

1) Simulation: This technique provides graphical visualization of behavior over a predefined number of runs of the system model. Simulation can be formulated as the property:

$$simulate\ n[bound]\{E_1, .., E_k\}$$

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 expressions to be monitored.

2) Resource Feasibility Analysis: Feasibility analysis is used to verify if a certain resource usage stays within the available resource amounts provided by the platform. The verification is achieved by examining the probability distribution of a particular resource, which is formulated as a probability evaluation, as follows:

$$Pr[bound](\psi)$$

where $bound$ defines the time bound for the runs, and ψ is either of the form <> q (eventually q), or [] q (always q), where q is a state predicate.

3) Worst-case Resource Consumption Analysis: The worst-case resource consumption analysis returns a path that will eventually reach a certain behavior at a maximal cost. This problem is reduced to maximizing the resource cost function such that the following property is satisfied:

$$E[bound; n](max : expr)$$

where *bound* is the time bound, n gives the number of runs, and $expr$ is the expression to be evaluated.

4) Resource Leakage: A resource leak, in our case a memory leak, occurs when the system uses a resource that is unable to be released back for later usage (even though it should do so). To this end, we perform a probability evaluation using UPPAAL SMC to detect possible memory leakage.

V. APPLYING SMC ON THE BRAKE-BY-WIRE SYSTEM

In this section, we apply our methodology to provide resource analysis for an industrial prototype system used previously in our research [6], [10], namely the Brake-by-Wire system.

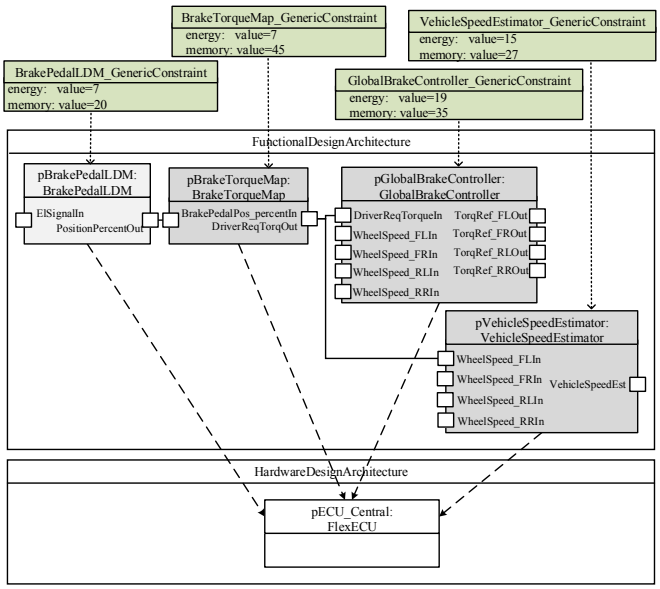


Fig. 4: The EAST-ADL model of the BBW system.

A. Brake-by-Wire System

The Brake-by-Wire (BBW) system is a braking system equipped with an Anti-Lock Braking (ABS) function, and without any mechanical connectors between the brake pedal and the brake actuators. A sensor attached to the brake pedal reads its position, which is used to compute the desired global brake torque. At each wheel, a sensor measures the speed of the wheel, which is used by the ABS algorithm together with the brake torque and the estimated vehicle speed to compute the actual brake torque that will be sent to the actuator. The ABS algorithm computes the slip rate s based on the following equation:

$$s = (v - w \times R)/v$$

where v is the speed of the vehicle, w is the speed of the wheel, and R is the radius of the wheel. The friction coefficient has a nonlinear relationship with the slip rate: when s starts increasing, the friction coefficient also increases, and its value reaches the peak when s is around 0.2. After that, further increase in s reduces the friction coefficient of the wheel. For this reason, if s is greater than 0.2 the brake actuator is released and no brake is applied, otherwise the requested brake torque is used. Figure 4 presents a part of the BBW system modeled in EAST-ADL, at Design Level, which is allocated to the pedal ECU. This model is extended with annotations for energy and memory consumptions, respectively, as a *GenericConstraint* each.

B. Applying ViTAL on the Brake-By-Wire System

In this section, we apply the proposed methodology and its tool support on the BBW system. We use the VITAL tool to automatically transform the EAST-ADL model of the BBW system into a network of timed automata. The timed automaton modeling the interface behavior of one of the BBW components, the pBrakePedalLDM *FunctionPrototype*, is depicted in Figure 5.

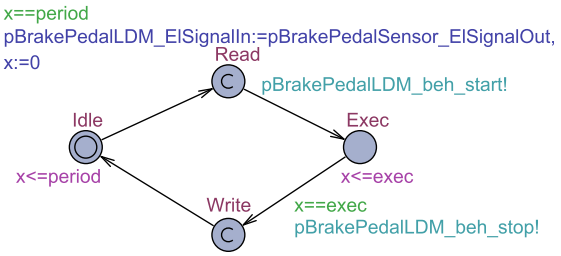


Fig. 5: The interface automaton of the pBrakePedalLDM.

C. The Monitor

Being equipped with the formal model of the BBW system, we can focus on the hardware elements of the platform, and their available resources, as they are represented in EAST-ADL. For the BBW system, we are interested in the *pECU_Central* component shown in the *Hardware Design Architecture*. This ECU is dedicated to the following pedal computational elements (depicted in Figure 4): *pBrakePedalLDM*, *pBrakeTorqueMap*, *pGlobalBrakeController*, and *pVehicleSpeedEstimator*, which are allocated to this ECU.

To analyze the resource consumption of the *pECU_Central* component (actually of the four allocated components), we have implemented a resource monitor depicted in Figure 6. The energy is consumed during the execution of the ECU. For the *pBrakePedalLDM* component, the model is annotated with energy'==6.3+random(1.4), which represents the consumption rate inherited from EAST-ADL $\pm$ 10% tolerance. In a similar manner, the monitor is annotated with its memory usage. Since memory is a static, discrete resource, it is allocated before the component is executed, and it is deallocated at the end of the execution.

D. Analysis of Energy Consumption

Figure 7 depicts the simulation of the energy consumption of the four components allocated on the *pECU_Central* hardware component. Concretely, the plot shows one stochastic simulation carried out for 50 time units, obtained by using the following query on the model: simulate 1[<=50]{energy, memory, t}.

We are also interested in the mean energy consumption and its distribution over runs bounded by a certain value. To obtain

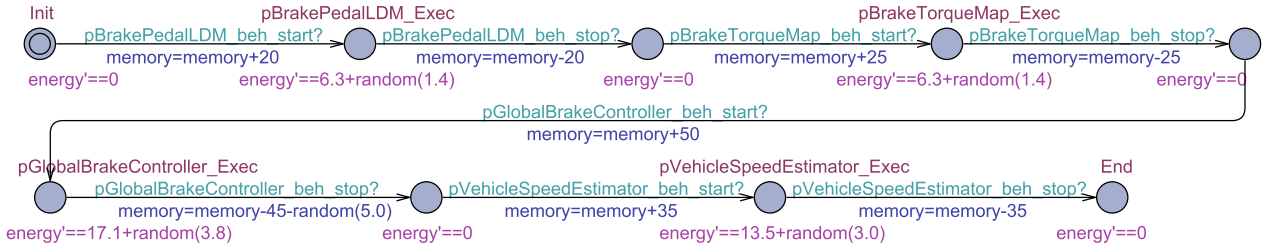


Fig. 6: Allocation of the software elements on the hardware platform.

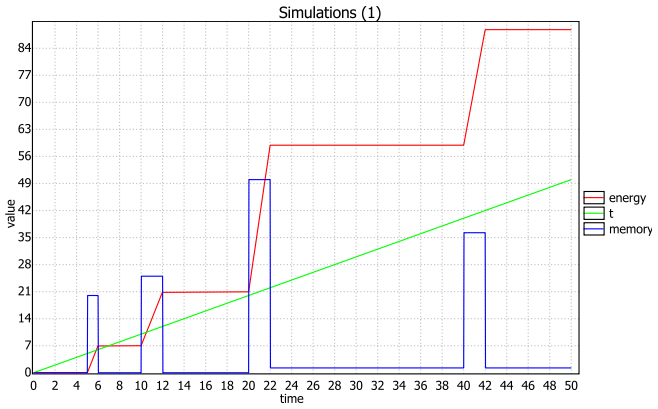


Fig. 7: Simulation of the energy consumption and memory utilization on pECU_Central.

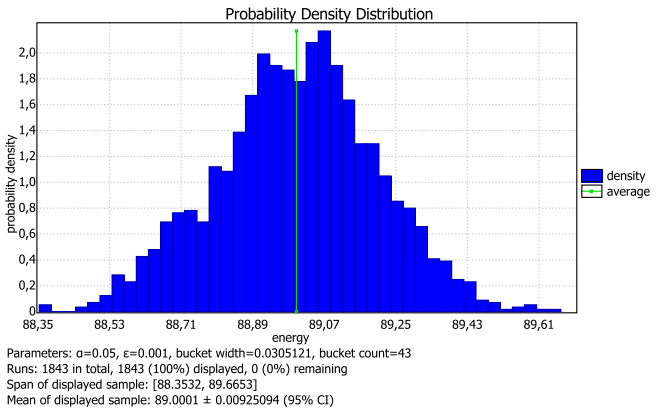


Fig. 8: Estimated energy probability distribution.

this, we check the query Pr[energy<=100](<> Monitor.End), where we assume that the bound of the actual energy is a realistic value that covers the reachable range for all runs. This value is based on the previous simulation of the model. Using UPPAAL SMC, we record the distribution of the energy consumption over 1843 runs, as shown in Figure 8. For increased accuracy, the energy consumption is checked with $\alpha = 0.05$ (the default value) and $\epsilon = 0.001$, which are parameters that improve the precision of the assessed runs. The mean value of the energy consumption is estimated at about 89 energy units, with the minimum value of the energy being approximated at 88.3532 and the maximum value of the energy being 89.6653 for the *pECU_Central* component.

In addition, we evaluate the maximum expected value for the energy. For this, we simulate the system over 2000 runs, trying to maximize the energy consumption, with the query E[t<=50, 2000](**max** : energy). The mean value provided by UPPAAL SMC for the maximum consumption is 89.0019. We note here that 89.0001 is the mean value of the energy distribution.

E. Analysis of Memory Usage

In addition to the energy analysis, in Figure 7 we depict the simulation of the memory usage of the four components allocated onto *pECU_Central*. As observed on the plot, not all the memory is deallocated by the *pGlobalBrakeController*, meaning that we encounter a memory leak. In order to check this with UPPAAL SMC, we use the following query against the model: Pr[t<=50] (<>Monitor.End **and** memory==0.0). The verifier returns the probability that the system will not suffer from a memory leak, in our case this being between 0 and 0.0019.

F. Discussion

In Table I, we present the overall results of our resource analysis methodology applied to the BBW system. This table lists–for each resource, analysis type and property to be checked–the number of states explored during model-checking, together with the time and memory used, as well as the number of simulation runs. Regarding energy analysis, UPPAAL SMC is able to find a solution by exploring 5068 states in 31 ms, using 27164 KB, within a single simulation of the model. We can observe that energy feasibility analysis and worst-case energy consumption analysis are computationally very expensive compared to the other properties. Nevertheless, this shows how capable UPPAAL SMC is in analyzing a realistic industrial system. Our observations could allow an industrial designer to gain deeper understanding in the system's resource behavior, and consequently adjust and optimize both software and hardware designs accordingly.

VI. RELATED WORK

Recently, there has been a growing interest in developing analysis and testing techniques to enhance the adoption of architectural description languages into industrial practice. Related work has also been carried out with respect to the analysis of embedded resources [12], using UML notations intended to complement architecture description languages. For instance, the work of Mallet et al. [13] can been seen

Resource	Analysis Type	Property	States Explored	Time (ms)	Memory (KiB)	Runs
Energy	*Simulation*	simulate 1[<=50]{energy}	5068	31	27164	1
	Feasibility Analysis	Pr[energy<=100](<> Monitor.End)	7841965	51995	27372	1843
	Worst-Case Energy Consumption	E[t<=50,2000](**max** : energy)	10136000	71277	27364	2000
Memory	*Simulation*	simulate n[<=50]{memory}	5068	47	27064	1
	Memory Leak	Pr[t<=50](<>Monitor.End **and** memory==0.0)	182448	1170	27356	36

TABLE I: Overall Results for Resource Analysis

as one of the major efforts in modeling of embedded systems and their resource usage. Targeting a similar goal, we have already implemented a methodology for formal analysis and verification of EAST-ADL [10]. In addition, in a recent paper we have proposed an extension of EAST-ADL for modeling and analysis of a system's resource-usage [14], which we have analyzed exhaustively with UPPAAL. In this paper, we have improved our previous work by proposing a method that employs statistical model checking for resource usage analysis of EAST-ADL models, in an attempt to increase the verification's scalability. To illustrate our approach, we have applied the proposed methodology on an industrial Brake-By-Wire system.

Other researchers have tried to address the problem of establishing a generic formal foundation for modeling and analysis of resources in embedded systems [15], from low-level code resource estimates [16] to higher level UML and formal approaches [17]. To continue this trend, we have focused on using abstract resource usage information at architectural levels of EAST-ADL, in order to provide to industrial systems designers analysis means for simulating and optimizing the system's overall resource usage.

VII. Conclusions and Future Work

In this paper, we have proposed a methodology for analyzing resource usage for EAST-ADL, using the UPPAAL SMC model-checker. We presented a case-study on which our methodology is applied to transform and analyze an industrial Brake-By-Wire system prototype. We have shown how the initial system can be transformed and annotated using the priced-timed automata formalism, and how the resource-wise behavior is semantically translated and statistically model-checked using UPPAAL SMC.

Regarding the analysis, we have shown how to analyze the usage of resources expressed in EAST-ADL within the priced-timed automata framework. In this setting, we have simulated the energy and memory usage of various BBW components, and have also performed resource feasibility analysis, as well as derive analysis results for worst-case energy consumption and memory leakage. Such results can help a system designer to gain a deeper understanding of the resource behavior of embedded systems modeled in EAST-ADL. As future work, we plan to implement this methodology in the VITAL tool and apply it to other real-world embedded systems.

References

[1] B. Graaf, M. Lormans, and H. Toetenel, "Embedded Software Engineering: The State Of The Practice," *IEEE Software*, vol. 20, no. 6, 2003.

[2] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach," *IEEE Transactions on Software Engineering*, vol. 18, no. 6, 1992.

[3] C. Seceleanu, A. Vulgarakis, and P. Pettersson, "Remes: A Resource Model for Embedded Systems," in *International Conference on Engineering of Complex Computer Systems*. IEEE, 2009.

[4] T. Šimunić, L. Benini, and G. De Micheli, "Cycle-accurate Simulation of Energy Consumption in Embedded Systems," in *Proceedings of the Design Automation Conference*. ACM, 1999.

[5] H. Blom, H. Lönn, F. Hagl, Y. Papadopoulos, M.-O. Reiser, C.-J. Sjöstedt, D.-J. Chen, F. Tagliabò, S. Torchiaro, and S. Tucci, "EAST-ADL: An Architecture Description Language for Automotive Software-Intensive Systems," *EAST-ADL White Paper*, vol. 1, 2013.

[6] R. Marinescu, H. Kaijser, M. Mikuèionis, C. Seceleanu, H. Lönn, and A. David, "Analyzing Industrial Architectural Models by Simulation and Model-Checking," in *Third International Workshop on Formal Techniques for Safety-Critical Systems*. Springer, 2014.

[7] R. Alur, "Timed Automata," in *International Conference on Computer Aided Verification*. Springer, 1999.

[8] P. Bulychev, A. David, K. G. Larsen, M. Mikučionis, D. B. Poulsen, A. Legay, and Z. Wang, "UPPAAL-SMC: Statistical Model Checking for Priced Timed Automata," *Workshop on Quantitative Aspects of Programming Languages and Systems*, 2012.

[9] (2014) The AUTomotive Open System ARchitecture (AUTOSAR) Standard. Available from http://www.autosar.org/.

[10] E.-Y. Kang, E. P. Enoiu, R. Marinescu, C. Seceleanu, P.-Y. Schobbens, and P. Pettersson, "A Methodology for Formal Analysis and Verification of EAST-ADL Models," *Reliability Engineering and System Safety*, vol. 120, 2013.

[11] P. Bulychev, A. David, K. G. Larsen, A. Legay, G. Li, and D. B. Poulsen, "Rewrite-Based Statistical Model Checking of WMTL," in *Runtime Verification*. Springer, 2013.

[12] H. H. Ammar, V. Cortellessa, and A. Ibrahim, "Modeling Resources in a UML-Based Simulative Environment," in *International Conference on Computer Systems and Applications*. IEEE, 2001.

[13] F. Mallet, M.-A. Peraldi-Frati, and C. André, "Marte CCSL to execute East-ADL Timing Requirements," in *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*. IEEE, 2009, pp. 249–253.

[14] R. Marinescu and E. P. Enoiu, "Extending EAST-ADL for Modeling and Analysis of System's Resource-Usage," in *Computer Software and Applications Conference Workshops*. IEEE, 2012.

[15] A. Vulgarakis and C. Seceleanu, "Embedded Systems Resources: Views on Modeling and Analysis," in *International Workshop On Component-Based Design Of Resource-Constrained Systems*, July 2008.

[16] J. Muskens and M. Chaudron, "Prediction of Run-Time Resource Consumption in Multi-task Component-Based Software Systems," in *Component-Based Software Engineering*, 2004.

[17] S. Becker, H. Koziolek, and R. Reussnerl, "The Palladio Component Model for Model-Driven Performance Prediction," *Journal of Systems and Software*, 2009.