# ViTAL : A Verification Tool for EAST-ADL Models using UPPAAL PORT

Eduard Paul Enoiu, Raluca Marinescu, Cristina Seceleanu, and Paul Pettersson
*Mälardalen Real-Time Research Centre (MRTC)*
*Mälardalen University*
*Västerås, Sweden*
*Email: {eduard.paul.enoiu, raluca.marinescu, cristina.seceleanu, paul.pettersson}@mdh.se*

***Abstract*—The influence of the systems architecture on the functions and other properties of embedded systems makes its high level analysis and verification very desirable. EAST-ADL is an architecture description language dedicated to automotive embedded system design with focus on structural and functional modeling. The behavioral description is not integrated within the execution semantics, which makes it harder to transform, analyze, and verify EAST-ADL models. Model-based techniques help to address this issue by enabling automated transformation between different design models, and providing means for simulation and verification. We present a way of integrating architectural models and verification techniques, which has been implemented in a tool called ViTAL. Consequently, ViTAL provides the possibility to express the functional EAST-ADL behavior as timed automata models, which have precise semantics and can be formally verified. The ViTAL tool enables the transformation of EAST-ADL functional models to the UPPAAL PORT tool for model checking. This method improves the verification of functional and timing requirements in EAST-ADL, and makes it possible to identify dependencies and potential conflicts between different vehicle functions before the actual AUTOSAR implementation.**

***Keywords*-model-based techniques; verification; analysis; UPPAAL PORT; EAST-ADL; Model transformation;**

## I. INTRODUCTION

The current trend is to use Model-driven Development (MDD) for automotive embedded systems and provide a basis for a systematic design at multiple abstraction levels [8]. EAST-ADL [9], [13] is an architecture description language for modeling and development of automotive embedded systems, covering the specification of requirements, system environment, vehicle functions, software and hardware resources, behavior, timing constraints, and other related information [23]. The EAST-ADL language provides an integrated modeling framework that uses concepts from MDD and component-based development [12].

EAST-ADL focuses on functional specifications [10] with support for structural definition. The behavior is defined only on the EAST-ADL component abstraction level, in terms of functional blocks. The functional behavior of a component is described using external notations such as Simulink or UML [24], and therefore the possibility to construct, verify, and transform EAST-ADL models using formal methods is restricted [11]. Also, many automotive functions are real-time, so formal verification of both functional and timely behavior is necessary, to ensure the real-time requirements at the architectural level. For instance, in automotive control applications one could verify that input data from sensors, the actual control computations, and output data to actuators are behaving accordingly to constraints of the controlled environment. Therefore, the verification of EAST-ADL models becomes truly challenging, especially when multiple software components are involved.

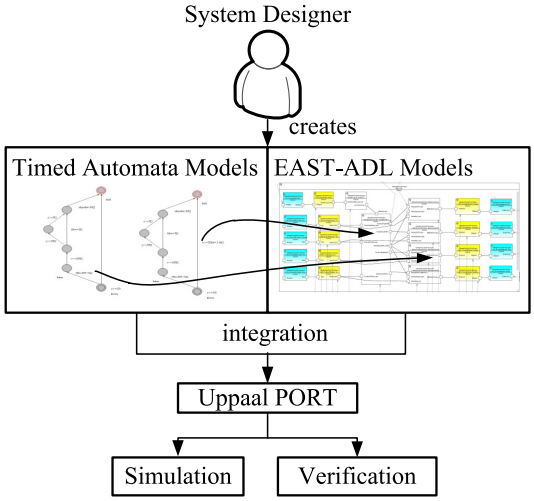


Figure 1. The workflow of the integrated simulation and verification tool

This paper proposes a method for integrating architectural description languages and verification techniques, tailored for EAST-ADL models and implemented in the tool ViTAL [1](A **Ver**ification **T**ool for EAST-**ADL** Models using UPPAAL PORT). ViTAL provides model-checking of EAST-ADL descriptions with respect to timing and functional behavioral requirements. As depicted in Fig. 1, the system designer creates the EAST-ADL models and the execution behavior using timed automata framework [3], and check whether a given requirement is satisfied. To achieve this, we implement

[1]ViTAL is available at http://www.idt.mdh.se/personal/eep/vital

978-2-9541-8100-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ICECCS.2012.42

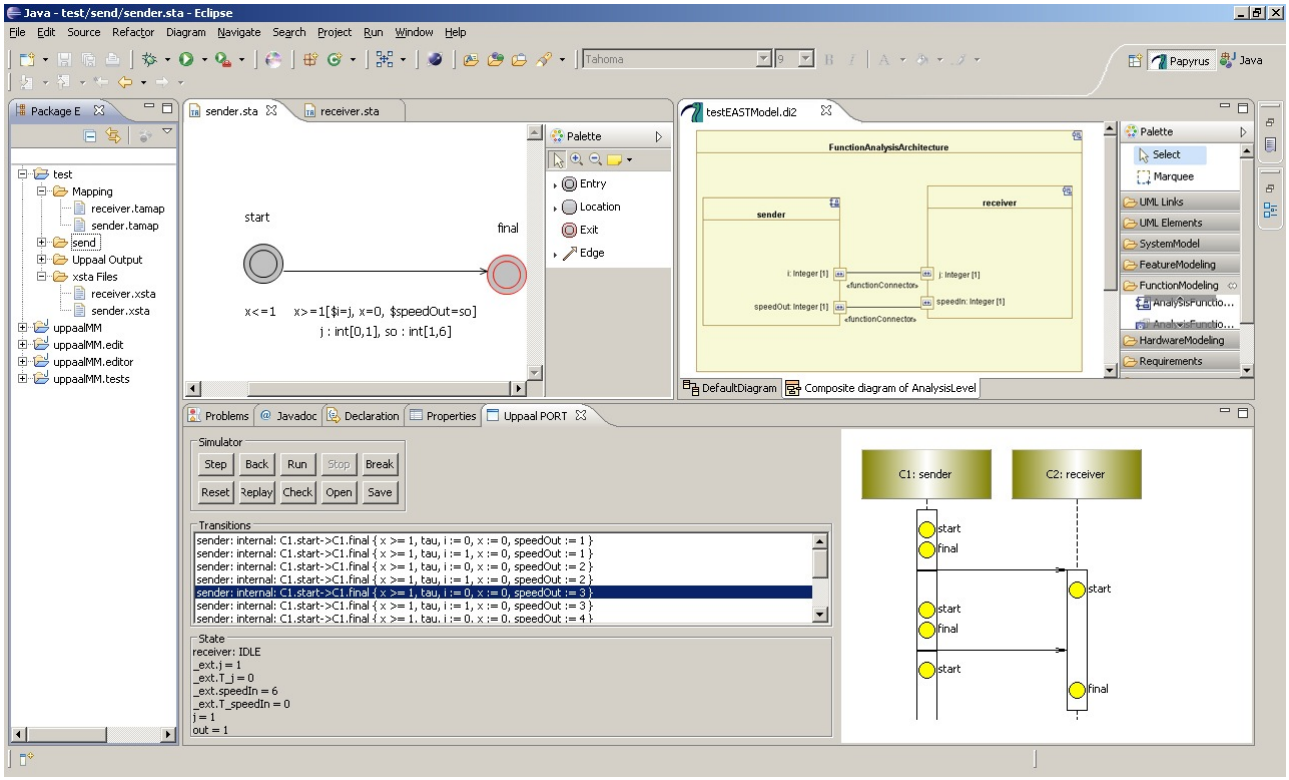


Figure 2. Integrated EAST-ADL platform editor, timed automata editor (upper view) and UPPAAL PORT simulator (lower view)

an automatic model transformation to UPPAAL PORT model-checker [17], which enables UPPAAL PORT to handle EAST-ADL models as input, and provides functional and timing behavior of functional blocks using timed automata semantics [3]. To increase user friendliness and alignment with the implementation of the EAST-ADL profile, we propose an integrated environment based on Eclipse plug-ins, as can be observed in Fig. 2. Our modeling and verification tool contains the following: an editor for timed automata visual description of the functional and timing behavior of EAST-ADL functional blocks, automated transformation of EAST-ADL models to UPPAAL PORT input model, support for mapping external timed automata variables to external ports, a simulator that can be used to validate the behavior of an EAST-ADL modeled system, and support for verifying reachability and liveness properties formalized in a subset of Timed Computation Tree Logic (TCTL).

In our approach, we combine powerful model checking techniques with the formal semantics of EAST-ADL models and a user-friendly graphical interface. The main features provided by ViTAL are:

- Support for formal verification of the execution behavior and timing using EAST-ADL language.
- The hierarchical structure of EAST-ADL ("read-execute-write" component semantics) is exploited in our approach by using UPPAAL PORT for efficient model-checking.

The paper is organized as follows. Section II briefly overviews EAST-ADL and UPPAAL PORT. Section III introduces the modeling approach for functional specification in EAST-ADL. Section IV describes our method and tool environment used for capturing the behavior inside each functional block and the transformation scheme to UPPAAL PORT. Next, we apply ViTAL on the Brake-By-Wire case study in Section V. In Section VI we compare to related work, before concluding the paper and presenting future works in Section VII.

## II. BACKGROUND

### A. EAST-ADL

EAST-ADL is an architecture description language specified through a meta-model and implemented as a UML2 profile [10]. It is structured into different abstraction layers representing different stages of an engineering process: vehicle level, analysis level, design level, and implementation

level. These levels are supported by complete traceability between them, reflecting the amount of details in an electronic system from a higher to a lower abstraction layer, as can be observed in Fig. 3.

The vehicle features (e.g. breaks) of an electronic system are modeled at the vehicle level, the highest level of abstraction. These features are refined at the analysis and design level by abstract elements representing software or device functions such as sensors and actuators. The implementation level is the lowest level of abstraction and is defined by using the AUTOSAR standard [22].

`Vehicle Level` is intended for elaboration of electronic features. A complete representation of the abstract functional definition of features in system context is modeled in the `Analysis Level`. This representation captures the main interfaces and behavior of the vehicle subsystems and allows validation and verification of the integrated system or its subsystems on a high level of abstraction.The `Analyisis Level` forms a natural constraint for developers when dealing with refined features and requirements, so the application software development needs to be iterated and performed together with the other levels of abstraction. The details for functional definition of software, including elementary decomposition, are introduced at the `Design Level`. The `Implementation Level` describes reusable code and AUTOSAR compliant software and system configuration for hardware deployment [10]. However, traceability is supported from `Implementation Level` elements to `Vehicle Level` elements [22].

In addition, this structural organization of EAST-ADL has modeling constructs for behavior, requirements, timing, variability, and safety aspects. EAST-ADL captures structural components that refer to external or internal behavior, as Simulink models.

### B. UPPAAL PORT

UPPAAL PORT is an extension of the UPPAAL tool, which supports simulation and model-checking of component-based systems, without the usual conversion or flattening to the model of network of timed automata. This is complemented by the Partial Order Reduction Technique (PORT) [5], [17] that UPPAAL PORT uses, to improve the efficiency of the model-checking analysis. This technique has been suggested [15] to reduce the state-space explosion caused by interleavings, with the main idea of exploring only a relevant subset of the state-space when model-checking. More precisely, UPPAAL PORT only explores properties which preserve a subset of the full model based on independence of transitions, instead of examining all possible sequences, which would add to the model-checking complexity, unnecessary. Since, the synchronization of global time is restricting the independence of transitions in the timed automata

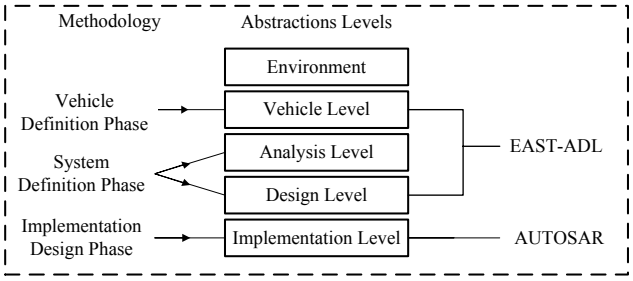


Figure 3. Overview of the EAST-ADL architecture

framework, UPPAAL PORT uses local time semantics to increase independence, therefore allowing the analysis of components in isolation followed by synchronization to a shared state whenever one writes to another.

UPPAAL PORT uses a guided PORT, which is based on the structure of the component-based system under analysis. This means that the employed component model has transitions independent of actions in other components. The performed experiments [17] suggest the use of UPPAAL PORT when dealing with a "read-execute-write" component model semantics.

## III. MODELING APPROACH AND INTEGRATION BRIDGE

The main purpose of this section is to introduce the ViTAL modeling approach for EAST-ADL system models.

Nowadays, a vehicle may be composed of more than 2,000 software and hardware based functions. Usually, the requirements engineer decides which functions are needed and how they should be structured in terms of interactions. EAST-ADL describes the whole vehicle system from several abstraction layers. As this paper only discusses the abstract functional models of a system, we employ the EAST-ADL functional abstraction, as the modeling language to specify the structure of a system. Specifically, on the analysis level, the system is described by a Functional Analysis Architecture (FAA). The FAA is composed of a number of interconnected Function Prototypes ($f_p$), where each prototype is an instantiation of Function Type ($f_t$) [9].

The challenge of the FAA is to target different functions for concepts like allocation of requirements, specifying, analyzing, and verifying functional requirements before implementation. In order to support unambiguous modeling and analysis, we have used the fact that there are no dependencies between the FAA and the other EAST-ADL levels [10]. This means that FAA can be defined separately from the other levels. For simplicity, we assume that, from an FAA-level point of view, the description is complete with respect to the dependencies between different functionalities.

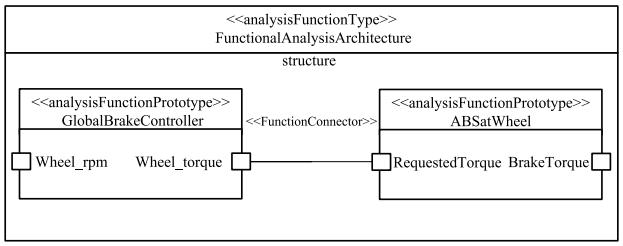


Figure 4. An EAST-ADL diagram modeling the functionality of a simplified system

### A. *The* EAST-ADL *Functional Model*

The goal of using EAST-ADL functional model with respect to behavior is to handle how model components are related to each other on capturing behavior and algorithms of the system as well as the environment.

In EAST-ADL functional modeling, systems in FAA are built from interconnected function blocks with well-defined interfaces consisting of a set of input- and output function ports (elements of flow ports to represent data transfer). The $f_p$ can be hierarchical, but the composing sub-functions have synchronous execution semantics. These functional blocks are time-triggered, or triggered based on data arrival on their flow ports. An $f_p$ follows the "read-execute-write" semantics, which ensures that once a function is triggered, it reads all input flow ports, executes the computation, and then writes to its output flow ports, all without interruption. For the presented work, the architectural specifications are used from structure, behavioral, and timing EAST-ADL packages [10], [11]. This simplifies the definition of semantics, and makes it easily extensible. As depicted in Fig. 4, the core intermediate model consists of three modeling elements: composite `Analysis Function Type`, basic `Analysis Function Prototype`, connectors `Function Connector`, and behavioral description named `Model`. The first three elements are translated from the EAST-ADL structure package. The behavioral description complemented with timing formalisms is complying with the component-based approach that enables early formal analysis of relevant concerns.

### B. *Timed Behavior*

We define the timed behavior of a functional block as a Timed Automaton (TA), extended with data variables and a final location. An $f_p$ in our setting is defined by its interface in terms of ports and a specified timed behavior [17]. The TA model is another abstraction of the $f_p$ behavior, where the assumed and desired properties of the system components are captured. To support analysis and verification of EAST-ADL FAA models in UPPAAL PORT, it

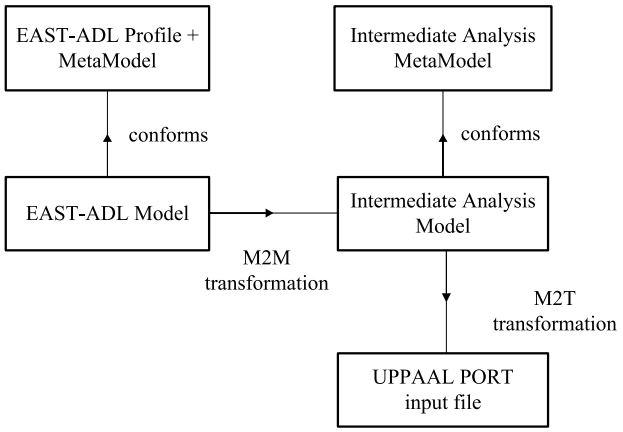


Figure 5. Model Export from EAST-ADL to UPPAAL PORT

is required that each functional block is associated with a behavioral model consisting of a TA, and a mapping between ports and automata variables. To provide system developers with concrete support in modeling the timed behavior of a functional block, an integration bridge is necessary. Hence, our next step has been to provide a direct mapping between flow ports and TA variables as additional model parameters. Consequently, this model association constitutes the bridge between EAST-ADL and TA models.

## IV. MODEL TRANSFORMATION TO UPPAAL PORT

Before proceeding further, we have to mention a few assumptions that we have made, for simplicity. For this work, no $f_t$ can have instances of other $f_t$ with the exception of FAA. The FAA $f_t$ is available and required for the transformation. With these assumptions, we have developed the model transformation shown in Fig. 5. We specify the input models for UPPAAL PORT as described in the metamodel that uses a subset of the EAST-ADL language constructs. In our approach, we focus on a certain subset of the tool data that we consider relevant in the context of integration. Irrelevant constructs, such as requirements models, variability models, are left out because they are outside the scope of this verification tool towards operational and timing requirements at EAST-ADL analysis level.

### A. *ViTAL and Model Integration*

We define a minimal structural integration inside ViTAL, an intermediate model, from which we can derive the constructs of EAST-ADL language. This simplifies the definition of semantics, and makes it easily extensible. The core intermediate model consists of three modeling elements: composite $f_t$, basic $f_p$, and connections. Using these, we can

describe all constructs in our assumed EAST-ADL model on FAA. A simple one-to-one mapping rule between structural entities is not sufficient though. Several parameters need to be handled in the integration process.

Each modeling element, except for the FAA $f_t$, has a set of flow ports, through which it can interact. Each flow port is represented as an input or an output port that has an associated type. A flow port is associated with the same type of data as the associated variable. Similar to the EAST-ADL language itself, connections define how data can be transferred between two $f_p s$. We assume no knowledge about the time that it takes for the data to be transmitted over a connection or if data can be lost. This assumption is acceptable when modeling the abstract functional system in EAST-ADL at analysis level, and therefore most implementation details are hidden. Nevertheless, the transmission over a connection, the execution, and communication resources are modeled in EAST-ADL at design level. Other structural EAST-ADL constructs are not represented directly by any modeling element, hence they are not influencing the transformation.

For the presented integration in ViTAL, the architectural information related to structure and timing are partially derived from the EAST-ADL model. Every $f_p$ is annotated in the intermediate model with an *event function* that submits to a *periodic constraint*. An event function is a trigger generator annotated with a parameter T for period. A new period starts every T time units, and the event function generates a trigger after each period elapses.

The EAST-ADL language imposes some restrictions on the $f_p$ behavior that should be addressed in the intermediate model as well. For example, the *run-to-completion* semantics mentions that input flow ports may only be accessed at the beginning of each triggering, and output flow ports are only written at the end of the computation. Therefore, TA($f_p$) denotes its behavior augmented with an interface. The interface of an $f_p$ consists of flow ports and the annotated trigger information. An input flow port has an associated variable holding the current data flow. A basic $f_p$ corresponds to a basic intermediate functional block with an automaton that can capture the behavior of the associated $f_t$ and maybe some other information like execution time. The internal computation of an $f_p$ starts with reading all input flow ports. These internal input data is used together with other functional information during the $f_p$ execution, before writing the variables to the output flow ports.

With the mentioned assumptions, an intermediate metamodel has been developed, which is described in this section. The modeling elements of the intermediate model, used in the integration, are described in Fig. 6 where the core elements represents structure at analysis level of abstraction, model behavior, and timing information. The intermediate model provides function modeling concepts which are mapped to concepts of component based design. `Element` can be mapped with a `FunctionType` and `EventFunction` elements. There are connectors like `FunctionConnection` and ports such `FlowPort`. Ports are typed so a `FlowPort` is typed by an `Analysis Data Type`. Furthermore an `Element` can have a behavioral description as a `Model` element.

The intermediate model obtained after the transformation represents the execution behavior, and can include triggering and timing information, but also some assumed functionality. Therefore, ViTAL provides means to extend the internal behavior of $f_p$ not only in terms of timing, but also content.

### B. Implemented Integration

The EAST-ADL language is implemented in a UML2 profile with the purpose of providing the ability to describe EAST-ADL-compliant models using this profile. The Papyrus UML tool implements all the properties and stereotypes as defined in EAST-ADL specification and may be applied on any kind of tool-supported UML models. The model transformation and modeling environment is based on Eclipse[2], which ensures a seamless integration with the UML Editor in Papyrus, needed for developing EAST-ADL models. It provides an intuitive and user friendly graphical environment.

The proposed ViTAL tool is a collection of Eclipse IDE plug-ins. Eclipse IDE has become a popular development platform, in particular within the open source community. Our analysis and verification tool is build upon an MDD set of Eclipse plug-ins: Eclipse Modeling Framework [3], Graphical Modeling Framework, Graphical Editing Framework, ATL, and Acceleo. Fig. 7 shows the EAST-ADL model transformation architecture. The EAST-ADL editor plugin uses Papyrus UML to create the analysis functions and interconnections among them. Papyrus saves its models in two files, one using a ".uml" extension and the other using a ".di" extension. The former file contains the actual model information, whereas the latter file contains all the graphical information.

The Papyrus UML Editor produces compatible EMF models that serve as a basis for our combined structural and behavioral mapping to UPPAAL PORT. As shown in Fig. 5, we introduce an intermediate model that serves as the interface between the EAST-ADL model and UPPAAL PORT input model[4]. The intermediate model conforms to the EAST-ADL metamodel that is aligned with the EAST-ADL profile and UML metamodel. The structural mapping transforms an EAST-ADL model that was created in the

[2]Eclipse is a multi language software development environment, benefiting from an extensible plug-in system.

[3]Eclipse Modeling Framework (EMF) is a modeling framework that has code generation capabilities for enabling viewing and editing of models.

[4]The input language employed is used to determine the structure of the modeled system.

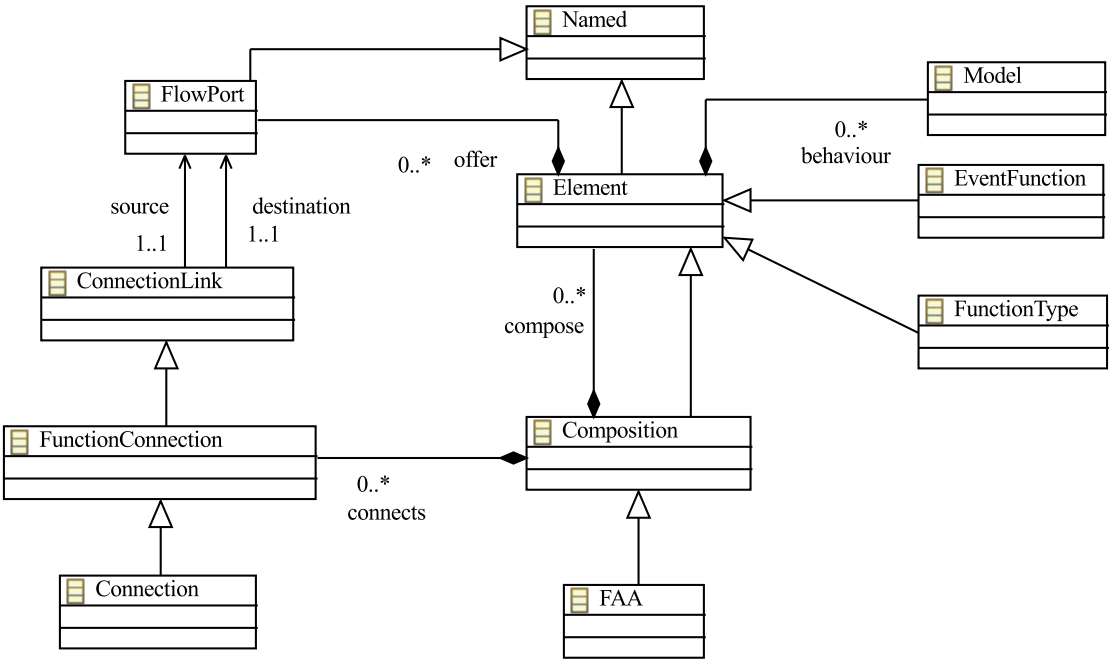


Figure 6. Simplified diagram representing the meta-model elements for the intermediate model

Papyrus UML modeling environment, into an intermediate model. The transformation is called M2M transformation in Fig. 5. The structure of the intermediate model resembles the UPPAAL PORT input model [5], so it is close to the structure of the desired output. This step of the transformation achieves an integration between the domain of EAST-ADL and that of UPPAAL PORT. We use the ATL M2M Component to convert models from one side to the other, due to its simplicity and integration within the Eclipse platform. We have implemented the mapping rules presented in Section IV-A as an ATL description of the transformation logic. For the transformation, a few modifications of both metamodels have been made. In principle, these changes are in fact ways to preserve the semantics of the original model. For instance, EAST-ADL uses the type - prototype constructions in which the declaration is in $f_t$, and the actual usage is in $f_p$. In this case, the structural transformation has a pointer between $f_t$ and the contained TA($f_p$).

In addition to the mentioned automated structural transformation, a manual TA integration needs to be carried out. In order to model the timing and behavior of an $f_p$, we integrate a TA editor. The behavior can be represented in a graphical notation by the system designer. These models differ from UPPAAL TA models as follows: (i) the timed behavior is extended with a final location out of which no edges are leaving, and (ii) synchronization channels are not allowed, because of the semantics employed by EAST-ADL models. For more details on the specifics of the TA employed by UPPAAL PORT, we refer the reader to the work of Håkansson and Pettersson [17].

With this information at hand, we need to bind TA variables to the flow ports of the EAST-ADL functions, next. This is needed in order to use the structural information contained in the intermediate model. We provide a variable to the port mapping plug-in. In the current version of ViTAL, the mapping is using the name of the timed automaton file to automatically generate the parameters to be used.

Once the previous steps have been completed, the TA and the intermediate model can be merged into the output of this process, which is compiled to an XML-format accepted by UPPAAL PORT tool [6]. This transformation is carried out using the Acceleo code generator for transforming the intermediate model into code. The ViTAL tool architecture is shown in Fig 7. The user interface integrates an editor for EAST-ADL models in the Eclipse framework, as well as a TA editor to model the timing and behavior of EAST-ADL functional blocks. UPPAAL PORT introduces support for simulation and verification, using a client-server architecture [16]. The UPPAAL PORT model-checker consists of two

[5]We refer the reader to the SaveCCM language reference manual for more details [2].

[6]The XML syntax describing the element definitions from the Document Type Definition is available in the SaveCCM language reference manual [2].

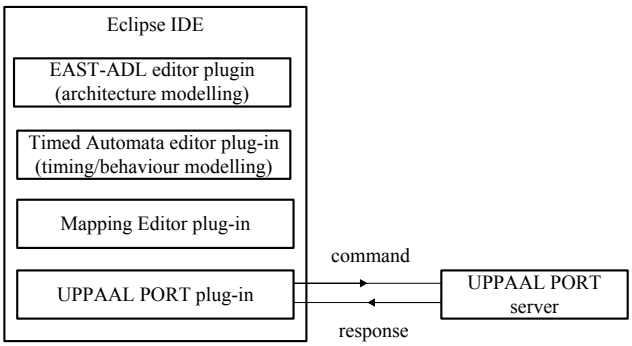


Figure 7. Overview of the ViTAL tool architecture

modules: the Eclipse plug-in used as the graphical simulator, and the server executing the verification.

Using the integrated simulator it is possible to validate the behavior and timing of an EAST-ADL functional model, prior to design and implementation. The simulator allows stepping forward and backwards in the state space, while selecting possible transitions. In a simulation trace, a data transfer is defined by the exchange of data between EAST-ADL $f_p$s (annotated with the timed automata locations) via their respective ports. Also, by using the verifier interface, it is possible to establish, by model checking the described behavior, whether the system model satisfies the functional and timing requirements specified in a subset of TCTL.

## V. Example: A Brake-by-Wire Control System

In order to check the applicability of ViTAL, we have performed a case study in which a Brake-By-Wire (BBW) system is modeled in EAST-ADL. The case study is based on a use case provided by Volvo Technology within the MBAT project [21]. In Fig. 8 one can see a simplified schematic illustration of the BBW system with Anti-lock Braking System (ABS) function, where no mechanical connection exists between the brake pedal and the brake actuators applied to the four wheels. The system is composed of five Electronic Control Units (ECU) connected by a network bus. The central ECU has three components:

- Brake Pedal Sensor (BPS) - reads the pedals position percentage.
- Brake Torque Calculator (BTC) - computes the desired global torque.
- Global Brake Controller (GBC) - calculates the torque required for each wheel.

The intended functionality of the BBW system is the following: when the driver brakes, it uses the pedal, and the brake actuators are applying a force that relates with the angle of the pressed pedal. The system in composed of a `Brake Pedal Sensor` that reads the pedal position percentage used by the `Brake Torque Calculator` to compute the desired Global Torque used by the `Global Brake Controller` to calculate the torque required for each wheel. The `Wheel Sensor` measures wheel speed for the `Global Brake Controller`, the `ABS` controls the braking to prevent locking the wheel, based on the slip rate.

The other four ECUs are connected to the four wheels, respectively. At each wheel, the Wheel Sensor measures the wheel speed and sends a signal to the GBC component. The ABS controls the wheel braking in order to prevent locking the wheel, based on the slip value. The slip value is calculated by the equation:

$$s = (v - w \times r)/v,$$

where $v$ is the vehicle speed, $w$ the wheel speed, and $r$ the wheel radius. The friction coefficient of the wheel has a nonlinear relationship with $s$: when $s$ increases from zero, the friction coefficient also increases and the value reaches the peak when $s$ is around 0.2. After that, further increase in $s$ reduces the friction coefficient. For this reason, if $s$ is greater than 0.2 the brake actuator is released and no brake is applied, else the requested brake torque is used.

We have modeled, simulated, and verified the BBW system in our tool ViTAL. The system has been modeled in Papyrus UML Editor, where a UML profile is used for architectural description. As illustrated in Fig 8, we use only the structural and timing specifications. The architecture of the system is encapsulated in one FAA $f_t$ that contains six interconnected $f_p$s modeled using the TA editor. Each TA($f_p$) defines the actual functional and timing behavior of the $f_p$.

The slip rate calculation is controlled by variable `slipRate`. For instance, from location `calculateSlipRate`, based on the current vehicle speed `vSpeed`, the wheel speed `wSpeed`, and the wheel radius `wRadius`, the `TorqueCmd` controls the wheel braking in order to prevent locking the wheel. Consequently, the ABS enters location `BrakeTorque`, and jumps back to location `Start`, provided that `slipRate` is greater than 20, the brake actuator is released and no brake is applied, else the requested brake torque is used.

A set of properties concerning the safety and liveness of the BBW system have been verified. We discuss a few representative properties. The property of deadlock freedom is satisfied for all execution paths of the state-space. As our approach and also the underlying tool supports simulation and verification of architectural properties including functional and timing properties, the following CTL specification is an example of such property, which ensures the brake reaction delay specified in the BBW model:

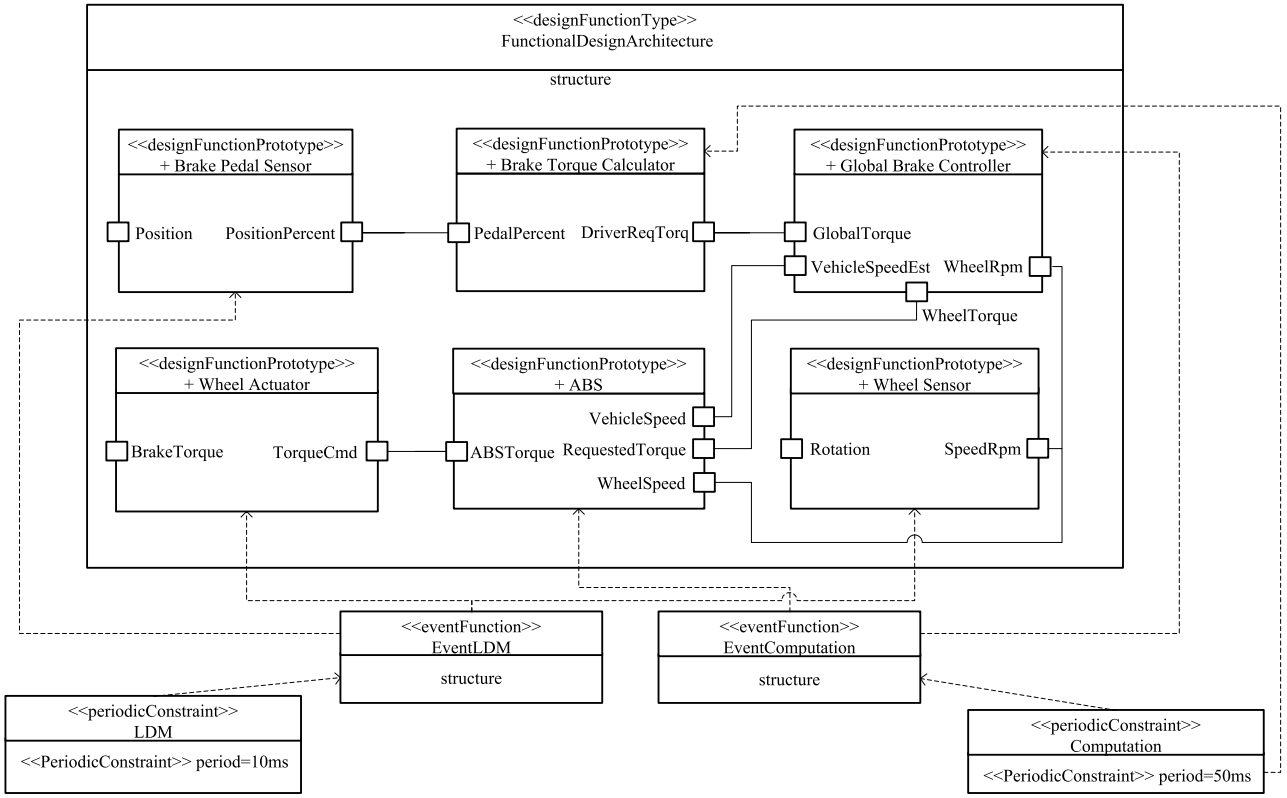


Figure 8. Brake by Wire control system

$$A[\,](BBW.reaction\ imply\ (BBW.clock < 200))$$

One of the functional requirements of the system is related to the slip rate $s$. With ViTAL, we can verify the following functionality: in case the slip rate variable exceeds $0.2$, the brake actuator is released and no brake is applied:

$$A[\,](BTC.s > 0.2\ imply\ (ABS.brake = 0))$$

We note that ViTAL is indirectly using the delay constraints information from EAST-ADL models during verification. To handle automated integration during verification of TCTL properties of this type, this delay constraint information should be considered as a transformation parameter and then checked automatically in UPPAAL PORT.

## VI. RELATED WORK

A lot of work has been done to allow the formal analysis and verification of Architecture Description Languages (ADLs ) for real-time embedded systems, and specifically on the Architecture Analysis and Design Language (AADL) in which an external behavior specification is used [1], [18]; all this work deals with formal analysis of behavior specified outside AADL by using communicating timed automata [1] or C/Ada source code [18]. Others are focusing on AADL enriched with its behavioral annex [6], [7]. Specifically, Berthomieu et al. [6] maps AADL and its behavior annex into the Fiacre language towards behavioral verification with TINA tool. The main difference between this work and ours is that we give a graphical behavior specification with support for model checking, system simulation, timing analysis, and an integrated tool based on Eclipse.

Several methods have been developed for the formal analysis and verification of EAST-ADL models. A lot of effort has been carried out to allow the use of UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) [4] together with EAST-ADL for timing analysis [20]. Feng et al. use the SPIN model checker for formal verification of EAST-ADL functional models [14]. The work is based on UML2 activity diagrams, and in contrast to our work, it does not allow the integration of timing constraints in the behavioral model. Qureshi et al. describe an integration effort towards formal verification of

EAST-ADL models based on timing constraints [22]. This allows a manual transformation from EAST-ADL models to UPPAAL models in order to achieve verification of constraints with respect to triggering and timing. Even though it offers prototype support for model-checking reachability and safety properties corresponding to the timing constraints, it does not support model-checking of functional constraints, or improves verification of complex system models, with respect to space and time, via the PORT model-checking technique, as the ViTAL tool does. For more information on advantages of using a PORT technique we refer the reader to the following experimental benchmark of Håkansson and Pettersson [17]. Kang et al. [19] performed a pre-study towards verification of EAST-ADL models using UPPAAL PORT with the aim of identifying integration needs. The results were considered in support towards our approach.

## VII. CONCLUSION

The analysis and verification of EAST-ADL models requires a consistent and integrated environment that brings together model-driven development and formal analysis. In our case, the employed formalism is the timed automata framework that facilities capturing the execution flow inside each functional block and the complex interactions between components. In this paper, we have described a method and transformation environment towards the integration of EAST-ADL and UPPAAL PORT. The main goal of our integration work has been twofold: (i) to provide an unambiguous behavioral description of EAST-ADL function blocks, and (ii) to bring formal verification capabilities to the EAST-ADL models. Both desiderata have been fulfilled within the same modeling and verification tool , which we call ViTAL. ViTAL is enhancing the behavioral definition of the EAST-ADL language and allows formal modeling, simulation, and verification of functional and timing requirements. The prerequisite artifacts for the system's formal analysis are the EAST-ADL architectural model, and the TA behavioral model that the system designer creates. Within ViTAL, we have integrated such models, in order to be able to simulate and check whether a given requirement is satisfied, by model-checking the TA description with UPPAAL PORT. In particular, the independence introduced by the "run-to-completion" semantics, employed by the EAST-ADL functional modeling, is exploited by UPPAAL PORT, in order to reduce time and space requirements for model checking.

Out of the possible future continuations of this work, we select the following, as our nearest research targets: (i) richer transformation constructs in order to automatically check delay and synchronization constraints, and (ii) the integration of UML2 activity diagrams in the employed transformation formalism, to capture the execution flow inside each functional block directly from EAST-ADL.

## ACKNOWLEDGMENT

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n$^o$ 269335 and from VINNOVA, the Swedish Governmental Agency for Innovation Systems. Henrik Lönn, Thomas Söderqvist, Daniel Karlsson from Volvo Technology, Lei Feng from KTH Royal Institute of Technology, as well as the paper reviewers, are gratefully acknowledged for valuable suggestions and insights on the topic of this paper.

## REFERENCES

[1] T. Abdoul, J. Champeau, P.T. Dhaussy, P.-Y. Pillain, and J.-C. Roger. Aadl execution semantics transformation for formal verification. In *Engineering of Complex Computer Systems, 2008. ICECCS 2008. 13th IEEE International Conference on*, pages 263 –268, 31 2008-april 3 2008.

[2] Mikael Åkerholm, Jan Carlson, John Håkansson, Hans Hansson, Mikael Nolin, Thomas Nolte, and Paul Pettersson. The saveccm language reference manual. Technical Report ISSN 1404-3041 ISRN MDH-MRTC-207/2007-1-SE, Målardalen University, January 2007.

[3] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235, 1994.

[4] Charles André, Frédéric Mallet, and Robert De Simone. Modeling of Immediate vs. Delayed Data Communications: from AADL to UML MARTE. In *ECSI Forum on specification & Design Languages (FDL)*, pages 249–254, Barcelona, Espagne, 2007. ECSI, ECSI.

[5] Johan Bengtsson, Bengt Jonsson, Johan Lilius, and Wang Yi. Partial order reductions for timed systems. In Davide Sangiorgi and Robert de Simone, editors, *CONCUR'98 Concurrency Theory*, volume 1466 of *Lecture Notes in Computer Science*, pages 485–500. Springer Berlin / Heidelberg, 1998.

[6] Bernard Berthomieu, Jean-Paul Bodeveix, Silvano Dal Zilio, Pierre Dissaux, Mamoun Filali, Sbastien Heim, Pierre Gaufillet, and Franois Vernadat. Formal Verification of AADL models with Fiacre and Tina. In *ERTSS 2010 – 5th International Congress and Exhibition on Embedded Real-Time Software and Systems*, May 2010.

[7] Stefan Björnander, Cristina Seceleanu, Kristina Lundqvist, and Paul Pettersson. Abv a verifier for the architecture analysis and design language (aadl). In *Sixth IEEE International Workshop UML and AADL at ICECCS 2011*, April 2011.

[8] Manfred Broy. Challenges in automotive software engineering. In *Proceedings of the 28th international conference on Software engineering*, pages 33–42, 2006.

[9] MAENAD Consortium. East-adl domain model specification: http://www.maenad.eu/. http://www.maenad.eu/, 2011.

[10] The ATESST2 Consortium. East-adl profile specification: www.atesst.org/home/. www.atesst.org, 2010.

[11] The ATESST2 Consortium. Evaluation report east-adl2 behavior support: http://www.atesst.org/home/. www.atesst.org, 2010.

[12] P. Cuenot, De Jiu Chen, S. Gerard, H. Lönn, M.-O. Reiser, D. Servat, C.-J. Sjostedt, R.T. Kolagari, M. Torngren, and M. Weber. Managing complexity of automotive electronics using the east-adl. In *Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on*, july 2007.

[13] Philippe Cuenot, Patrick Frey, Rolf Johansson, Henrik Lönn, Yiannis Papadopoulos, Mark-Oliver Reiser, Anders Sandberg, David Servat, Ramin Tavakoli Kolagari, Martin Törngren, and Matthias Weber. 11 the east-adl architecture description language for automotive embedded software. In *Model-Based Engineering of Embedded Real-Time Systems*, Lecture Notes in Computer Science, pages 297–307. Springer, 2011.

[14] Lei Feng, DeJiu Chen, H. Lönn, and M. Torngren. Verifying system behaviors in east-adl2 with the spin model checker. In *Mechatronics and Automation (ICMA), 2010 International Conference on*, pages 144 –149, aug. 2010.

[15] Patrice Godefroid and Pierre Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. In Kim Larsen and Arne Skou, editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 332–342. Springer Berlin / Heidelberg, 1992.

[16] John Hakansson, Jan Carlson, Aurelien Monot, and Paul Pettersson. Component-based design and analysis of embedded systems with uppaal port. In *6th International Symposium on Automated Technology for Verification and Analysis*, pages 252–257. Springer-Verlag, October 2008.

[17] John Hakansson and Paul Pettersson. Partial order reduction for verification of real-time components. In *Proceedings of the 5th international conference on Formal modeling and analysis of timed systems*, pages 211–226. Springer-Verlag, 2007.

[18] Jerome Hugues, Bechir Zalila, Laurent Pautet, and Fabrice Kordon. From the prototype to the final embedded system using the ocarina aadl tool suite. *ACM Trans. Embed. Comput. Syst.*, 7(4):42:1–42:25, August 2008.

[19] Eun-Young Kang, Pierre Yves Schnobbens, and Paul Pettersson. Verifying functional behaviors of automotive products in east-adl2 using uppaal-port. In *Proceedings of the 30th International Conference on Computer Safety, Reliability and Security (SAFECOMP'11)*. Springer-Verlag, September 2011.

[20] F. Mallet, M.-A. Peraldi-Frati, and C. Andre. Marte ccsl to execute east-adl timing requirements. In *Object/Component/Service-Oriented Real-Time Distributed Computing, 2009. ISORC '09. IEEE International Symposium on*, pages 249 –253, march 2009.

[21] ARTEMIS MBAT. Consortium website: http://www.mbat-artemis.eu/, November 2011.

[22] Tahir Naseer Qureshi, DeJiu Chen, Henrik Lönn, and Martin Törngren. From east-adl to autosar software architecture: a mapping scheme. In *Proceedings of the 5th European conference on Software architecture*, ECSA'11, pages 328–335, Berlin, Heidelberg, 2011. Springer-Verlag.

[23] Anders Sandberg, DeJiu Chen, Henrik Lönn, Rolf Johansson, Lei Feng, Martin Törngren, Sandra Torchiaro, Ramin Tavakoli-Kolagari, and Andreas Abele. Model-based safety engineering of interdependent functions in automotive vehicles using east-adl2. In *Proceedings of the 29th international conference on Computer safety, reliability, and security*, pages 332–346. Springer-Verlag, 2010.

[24] Carl-Johan Sjöstedt, Jianlin Shi, Martin Törngren, David Servat, Dejiu Chen, Viktor Ahlsten, and Henrik Lönn. Mapping simulink to uml in the design of embedded systems: Investigating scenarios and transformations. *CiteSeerX - Scientific Literature Digital Library*, 2009.