Framework for Transforming Compact Surface Languages into Augmented EAST-ADL Models

Imad Berrouyne*, Alessio Bucaioni*, Federico Ciccozzi*, Muhammad Waseem Anwar*, Henrik Lönn†
* Mälardalen University (Sweden), *name.surname@mdu.se*
† Volvo Group Trucks Technology (Sweden), *henrik.lonn@volvo.com*

***Abstract*—Automotive systems require rigorous methods to address their complexity. EAST-ADL has emerged as a promising architecture description language for specifying requirements, from the vehicle level down to software implementation. However, due to the inherent versatility of EAST-ADL, defining specific and fine-grained architectural features of automotive systems becomes challenging without addressing its intricacies. These challenges often result in several important aspects being overlooked. In this paper, we propose a two-step process consisting of: (1) separately specifying specific architectural features using domain-specific languages, and (2) injecting these features into an EAST-ADL core model through model transformation. Our conclusions demonstrate that domain-specific languages provide a succinct means of targeting specific modeling aspects while maintaining equivalence to the implementation in EAST-ADL.**

***Index Terms*—Automotive Systems, Timing Analysis, Variability, Model Transformation, EAST-ADL, Domain-Specific Language**

I. Introduction

The EAST-ADL is an architecture description language that enables the modeling of automotive systems across four levels of abstraction, spanning from the vehicle level down to the implementation level. It provides concepts to describe various concerns such as variability, timing, and safety, to name a few [1]. It complements AUTOSAR [2] and provides standardized concepts for describing automotive systems from a high-level engineering perspective.

The potential of EAST-ADL is restricted by its own versatility and flexibility. In fact, in conjunction with AUTOSAR, it offers enough concepts for a comprehensive description of vehicle complexity, be it software or hardware [3]. However, such versatility creates an intertwining of concerns. Considering some of these concerns during the engineering process becomes challenging due to the induced complexity of the automotive system model. Managing the complexity of an automotive system is multifaceted. While the specification of core concepts dealing with functional and hardware architecture is generally well supported, specific aspects such as timing, variability, and safety are often overlooked because of the intricacies of EAST-ADL [4].

Numerous studies have used the versatility of EAST-ADL to cover a variety of concerns (e.g., verification of a system, application aware of energy) in the automotive industry [5], [4], [6], [7], [8]. These concerns often represent crucial aspects for meeting customer needs. Hence, it is imperative to establish a method that distinctly separates them, specifically isolating their concepts from the foundational abstractions of EAST-ADL, to ensure that their specification is both precise and unambiguous.

The advantage of using EAST-ADL over existing methods (e.g., UML, SysML) [9], [10] lies in its reliance on well-established automotive abstractions, which were specifically designed by automotive experts to address the diverse needs and requirements of automotive systems. Thus, we can leverage its versatility, which aligns with standardized concepts, while maintaining a lightweight, separate layer to address specific needs.

This paper is a follow-up to our previous paper, in which we presented an approach to separate these concerns and emphasized the need for a transformation approach to bridge such separation [11]. The contribution of this paper consists of a model-to-model transformation framework addressing this facet of our work. The framework allows for the generation of an augmented EAST-ADL model from compact surface languages. The latter consists of a DSL with small and specific constructs addressing limited concepts. Indeed, the proposed framework separates these aspects from the core EAST-ADL model and uses model-to-model transformation rules to map them back into their equivalents in the core EAST-ADL model. We provide an assessment of our framework on two aspects: timing and variability.

The remainder of this paper is structured as follows. Section II provides an overview of the related work. Section III explains how we design our research. Section IV presents a running example describing a Windscreen Wiper. Section V provides the proposed transformation framework consisting of a brief description of our previous work and the transformation procedure. Section VI presents an application of the framework to our simplified case studies. Section VII discusses the implications of our approach. Finally, Section VIII presents the conclusion and future work.

II. Related Work

EAST-ADL is an architecture description language for automotive embedded systems, supporting multiple abstraction levels and various concerns throughout the system lifecycle [12]. Many approaches use the versatility of EAST-ADL to address certain aspects in automotive systems in a formal manner [5], [13], [6], [14], [8]. However, the literature lacks a framework that leverages this versatility in a structured manner.

Other complementary modeling languages such as AUTOSAR, RCM [15] and SysML support multiple aspects, including variability, beyond core concepts. AUTOSAR focuses on standardizing software interfaces and communication between components but lacks a dedicated, compact syntax for capturing variability, limiting it to software-related elements [16]. SysML, an extension of UML for system engineering, expresses variability through constructs like Block Definition Diagrams and Parametric Diagrams. However, it lacks standardized concepts. There are many studies that use model transformation to create models conforming to AUTOSAR and SysML in order to meet specific modeling aspects [17], [18], [19]. However, these approaches are developed in silos and lack a systematic framework for performing such transformations. The benefits of our approach is that it relies on EAST-ADL abstractions which were designed by automotive experts and are meant to cover the various needs and requirements of an automotive system.

Other studies have explored the capabilities of modeling languages using new abstractions, akin to surface languages. Holtmann et al. [20] introduced EATXT, a DSL for specifying EAST-ADL models in textual format, which describes entire models but does not provide means for examining specific aspects using new abstractions. It also lacks a transformation procedure that maps its concepts back into the EAST-ADL. Grönninger et al. [16] proposed using model views to address the complexity of representing SysML variability, focusing on specific aspects of the model.

Other works employ model-driven techniques to enhance the management of various aspects of software product lines, such as test script generation [21] and configuration files [22].

III. Research design

We aim to make the implementation of specific modeling aspects more accessible using a modular architecture. In that respect, we conduct our study within the scope of two commonly used modeling aspects: variability and timing. We selected these two aspects because they are natively supported by EAST-ADL, allowing their concepts to be seamlessly isolated into a DSL with minimal effort, making them ideal for a proof of concept. The following research questions aim to answer how the proposed framework can be beneficial in implementing them.

Our methodology initially permits testing our framework using small syntax with concrete real-world use cases to validate the feasibility of creating equivalent models using different abstractions.

We assess the feasibility of our approach by addressing a series of research questions and offering contributions that enhance the specification and implementation of specific modeling aspects within our framework.

A. Research questions

This paper seeks to answer the following Research Questions (RQs):

- **RQ1:** How can variability and timing be addressed more effectively using specific modeling aspects compared to existing EAST-ADL mechanisms?
- **RQ2:** How can the proposed solution be adapted or scaled to work seamlessly with EAST-ADL?

These RQs shape our investigation toward improving the separation of concerns within the EAST-ADL development process, focusing on concrete simplified case studies: variability and timing, empirical validation, and potential real-world applications in the automotive industry.

B. Contributions

We tackle the above-mentioned RQs by proposing a framework that leads to the following Research Contributions (RCs):

- **RC1:** A framework that separates the definition of specific modeling aspects from core concepts.
- **RC2:** Transformation mappings of two simplified case studies demonstrating the feasibility of the approach.

IV. Running Example

Figure 1 depicts our running example, which consists of the windscreen wiper functionality of an automotive system. This example targets variability aspects (upper part), timing aspects (lower part), and shows how they can be separated from the core concepts of EAST-ADL (middle part). For instance, in this example, specifying variability and timing using EAST-ADL constructs would require the EAST-ADL classes shown on the left-hand side in a single model. This representation mixes several levels of abstraction and requires three stakeholders: one responsible for specifying the structural features of the wiper, another responsible for ensuring the correct timing constraints, and another specifying the decisions with respect to variability.

In fact, while the structural features of an automotive system are generally fixed, the decision on what variants should be resolved and what timing constraints should be considered are dynamic, meaning that we expect to change them according to business and quality requirements. We propose to separate these three overlapping concerns using one core model, i.e., the EAST-ADL model in the middle, and two separate text-based layers for specific modeling aspects.

The example includes three windscreen wiper variants: *WiperCtrlStd*, *WiperCtrlAutoReturn*, and *WiperCtrlAutomatic*, specified in a core EAST-ADL model. On the one hand, the variability layer describes the constraints they must satisfy during variability resolution. It targets three frequently used types of constraints: ModelYear, Brand, and Class. For instance, the *WiperCtrlStd* can only be resolved if ModelYear is *2016*, Brand is *X*, and Class is *Heavy Duty*. On the other hand, the timing layer defines the constraints required to ensure the timing guarantees essential for the proper operation of the wipers.

Our running example provides a concrete illustration of the separation of variability and timing aspects and how they can affect a core EAST-ADL model by referencing its elements. Although these aspects can be specified separately in their

respective layers, it is desirable to include them in an EAST-ADL model to conform with the EAST-ADL standardized concepts. The goal of the presented framework is to enable the inclusion of these aspects after they have been specified separately.

Throughout this paper, we will rely on this running example to vividly illustrate the various facets of the proposed framework. Specifically, Section VI presents the outcome of applying this framework to one of the above-mentioned aspects.

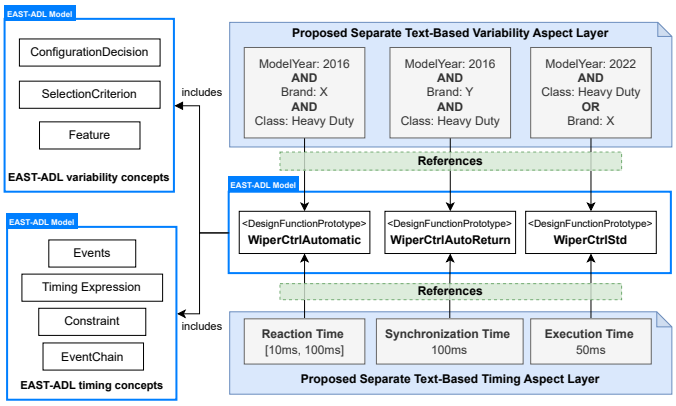

Fig. 1. Windscreen Wiper Running Example: the running example can be defined using the EAST-ADL concepts in the blue box labeled "EAST-ADL Model"; the EAST-ADL variability and timing concepts can be replaced by the proposed aspect layers that reference the core EAST-ADL model in the middle.

V. Framework for Separating and Implementing Specific Modeling Aspects

The proposed framework consists of two main parts. The first part is the specification of specific modeling aspects in a compact surface language. This part was detailed in our previous work [11]. To maintain continuity, we will provide a brief summary of that paper in the following section. The second part consists of the transformation that generates the augmented EAST-ADL model based on the information specified using such a compact surface language. We argue that this transformation enables the separation of concerns, i.e., separating the specific modeling aspects from the foundational concepts of EAST-ADL while adhering to the EAST-ADL standardized concepts through model transformation.

A. Specific Modeling Aspects

The first part of this approach consists of specifying the specific modeling aspects using a compact surface language. Practically, we describe these modeling aspects using a Domain-Specific Language (DSL). This DSL offers a straightforward and decoupled manner to express the engineer's concerns. For instance, in the running example, we target the specific cases where one wants to ensure that the final model meets certain criteria in terms of variability or timing. This DSL provides an editor permitting to validate the model instances, offer quick fixes and autocompletion.

1) Variability Aspects: In the case of our running example, we choose a specification for variability that relies on constraints. In this context, a constraint defines the conditions under which a given variability feature can be included. As a proof of concept, we propose the code snippet in Listing 1 as a potential syntax for its surface language. This code snippet reflects the concerns expressed in Figure 1 in a textual format and is intended to be written by an engineer focusing on the variability aspects in the final EAST-ADL model.

In this example, we create three **INCLUDE** instructions. The first applies to *WiperCtrlAutomatic*, the second to *WiperCtrlAutoReturn*, and the third to *WiperCtrlStd*. For instance, the first **INCLUDE** targets the WiperCtrlAutomatic contained in the WindscreenWipersPackage from the core model. It has three constraints, all retrieved from the VariabilityConstraints model: ModelYear="2016", Brand="X", and Class="Heavy Duty".

2) Timing Aspects: The timing aspects rely on a DSL that permits the specification of the timing constraints required for a given function. Listing 2 shows the corresponding potential syntax for each of the timing constraints presented in the running example.

In this example, we provide three **function** definitions. The first encompasses the timing constraint for *WiperCtrlAutomatic*, the second for *WiperCtrlAutoReturn*, and the third for *WiperCtrlStd*. For instance, the WiperCtrlAutomaticTiming function defines the constraint of WiperCtrlAutomatic, ensuring that the reaction time is within 10 to 50 ms. It contains two events, WiperCtrlOn and WiperCtrlRunning, which are used in the specification of ReactionConstraint. The **timing** definition consists of the **target**, the **type** of constraint, the **stimulus**, which is the event that triggers the constraint, the **response**, which is the event that happens after the stimulus, and the **constraint**, containing the parameters of the constraint.
The output of this part is a document containing a specification conforming to the syntax of the above-mentioned DSL. This specification is a model that serves as an input to the transformation procedure detailed in the next section. It is worth mentioning that such a DSL is designed to be compact, which limits its ability to address variability from a broader perspective. Thus, it is not intended to replace, for instance, the built-in feature modeling concepts of EAST-ADL.

B. Transformation Procedure

The second part consists of transforming the specification into an augmented EAST-ADL model. In this context, we propose the framework shown in Figure 3. It consists of five defining components. The first component is the EAST-ADL core model, where we reference the foundational concepts from the specification using specific abstractions. The second component is the grammar of the surface language to which the compact surface language must conform. The third component is the set of transformation mappings that create the link between the surface language and the EAST-ADL standardized concepts. The fourth component is the augmented EAST-ADL model generated by the application of these mappings. The fifth component is the set of applications that

```
INCLUDE WindscreenWipersPackage/WiperCtrlAutomatic IF VariabilityConstraints/ModelYear="2016"
                                                    AND VariabilityConstraints/Brand="X" AND VariabilityConstraints/Class="Heavy Duty"
INCLUDE WindscreenWipersPackage/WiperCtrlAutoReturn IF VariabilityConstraints/ModelYear="2016"
                                                    AND VariabilityConstraints/Brand="Y" AND VariabilityConstraints/Class="Heavy Duty"
INCLUDE WindscreenWipersPackage/WiperCtrlStd IF VariabilityConstraints/ModelYear="2022"
                                                    AND VariabilityConstraints/Class="Heavy Duty" OR VariabilityConstraints/Brand="X"
```

Listing 1. Specification of the Running Example Variability Aspects

```
function WiperCtrlAutomaticTiming {
  event WiperCtrlOn
  event WiperCtrlRunning
  timing ReactionTime {
    target: WindscreenWipersPackage/
            WiperCtrlAutomatic
    type: ReactionConstraint
    stimulus: WiperCtrlOn
    response: WiperCtrlRunning
    constraint: [10ms, 100ms]
  }
}
```

```
function WiperCtrlAutoReturnTiming {
  event LeftWiperParkPosition
  event RightWiperParkPosition
  timing SynchronizationTime {
    target: WindscreenWipersPackage/
            WiperCtrlAutoReturn
    type: SynchronizationConstraint
    events: (LeftWiperParkPosition,
             RightWiperParkPosition)
    constraint: 50ms
  }
}
```

```
function WiperCtrlStdTiming {
  event WiperCtrlOn
  event WiperCtrlRunning
  timing ExecutionTime {
    target: WindscreenWipersPackage/
            WiperCtrlStd
    type: ExecutionTimeConstraint
    start: WiperCtrlOn
    stop: WiperCtrlRunning
    constraint: 50ms
  }
}
```

Listing 2. Specification of the Running Example Timing Aspects

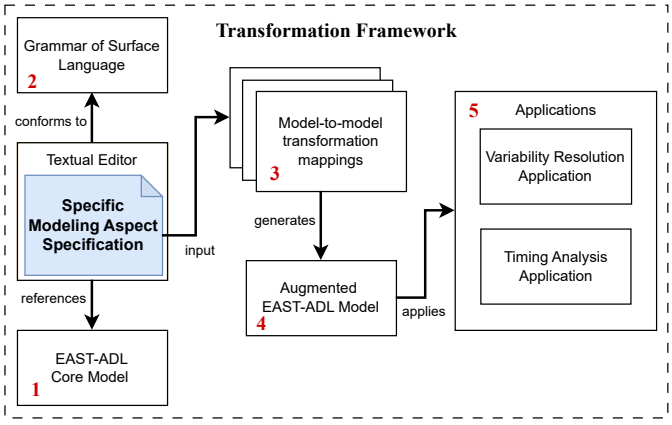

Fig. 3. Architecture of the Proposed Transformation Framework

can leverage this augmented model (e.g., variability resolution, timing analysis) [23].

The framework relies on a model-to-model transformation that takes as input a surface language model, i.e., the blue artifact in the figure provided by the textual editor, which conforms to the grammar and references the core EAST-ADL model elements to address the concerns at hand. Essentially, this transformation weaves the information specified in the surface language with the core EAST-ADL model to create an augmented EAST-ADL model that satisfies the target concerns. In the case of our running example, this procedure reflects the constraints in the generated artifact.

The implementation of the present framework is feasible using existing model-based tools. In this context, the Eclipse Modeling Framework (EMF) [24] constitutes the basis of the architecture, permitting a seamless integration of all components. The Eclipse editor hosts the DSL, enabling the specification of specific modeling aspects. The DSL grammar is defined using Xtext [25]. This grammar supports cross-referencing to the core EAST-ADL model, which is serialized as an EAXML file. EAXML is the standard format for serializing EAST-ADL models. The specification document generated through the Eclipse editor serves as the input for a model-to-model transformation. The implementation of this transformation relies on the Atlas Transformation Language (ATL) [26]. It consists of a set of model-to-model transformation rules that implement the mappings between the DSL and the augmented EAST-ADL model. This procedure results in the generation of an EAST-ADL model serialized as an EAXML file. This final artifact can be applied to various tasks, such as variability resolution or timing analysis.

C. Usage process

The usage process follows a waterfall approach, where the first phase involves specifying the core concepts, and the second phase focuses on defining the more dynamic and specific aspects. Since the structural features at the core—such as vehicle, analysis, design, and implementation levels—are generally static and require minimal change, their specification is typically performed once during the initial stages. Therefore, relying on the EAST-ADL core model, despite its rigidity at this phase, is reasonable. This step often involves architectural thinking and must be conducted by car experts equipped with EAST-ADL tooling.

In contrast, other dynamic aspects, designed using compact surface languages, require more frequent modifications and build upon the established core structural features. These aspects should be separated from the core model and designed to require minimal setup, ensuring they are not overlooked.

VI. Simplified Case Studies

In this section, we examine two examples of these aspects: variability and timing. We highlight the challenges of addressing them within the existing EAST-ADL tools and the benefits of specifying them using the proposed framework.

A. Variability

We apply the transformation procedure to variability within the proposed framework. The presented variability language provides concepts similar to those available in EAST-ADL. The variability concepts in EAST-ADL are distributed across multiple packages as an integral component of the overarching EAST-ADL architecture. Consequently, enabling the specification of variability requires navigating and integrating several interconnected concepts and abstractions.

Figure 4 depicts an example of the mapping between the surface language model and the augmented EAST-ADL model performed by transformation rules. It applies the transformation procedure to variability aspects of the running example. The left-hand side consists of the source surface language model, while the right-hand side consists of the target augmented EAST-ADL model. The red arrows indicate the mappings performed by the transformation. In this example, an **INCLUDE** class is mapped into EAST-ADL using four mappings: Mapping M1, Mapping M2, Mapping M3, and Mapping M4. First, Mapping M1 creates a ConfigurationDecision , which is an essential concept in EAST-ADL for selecting the various features in an automotive system. Then, Mapping M2 creates the composition of a SelectionCriteria, which is also an important concept for defining the criteria of this selection. Next, Mapping M3 sets the source EAElement to the SelectionCriteria. Finally, Mapping M4 associates a Feature as a target to the ConfigurationDecision. All these mappings ensure the conformance of the generated elements with the EAST-ADL standardized concepts.

For the sake of simplicity, we show one instance of the mapping that occurs for the first **INCLUDE** instruction in Listing 1, specifically how to transform the constraint stating that ModelYear should be 2016 in the WiperCtrlAutomatic. Thus, the Feature class is instantiated with ModelYear=2016, while the EAElement is instantiated with WiperCtrlAutomatic as a DesignFunctionPrototype, which inherits from an EAElement in the EAST-ADL standardized concepts. This procedure applies to all the constraints specified in each rule of Listing 1 in a similar manner.

B. Timing

Timing specification is crucial in automotive systems, requiring precise alignment with the specifier's objectives. While EAST-ADL supports the definition of timing constraints, the process can be intricate due to the need to navigate the complexities of the EAST-ADL architecture. The isolation of these aspects enables their systematic consideration, thereby ensuring safety through precise timing management.

Figure 5 details an example of the mappings required to transform the concepts of the surface language model into

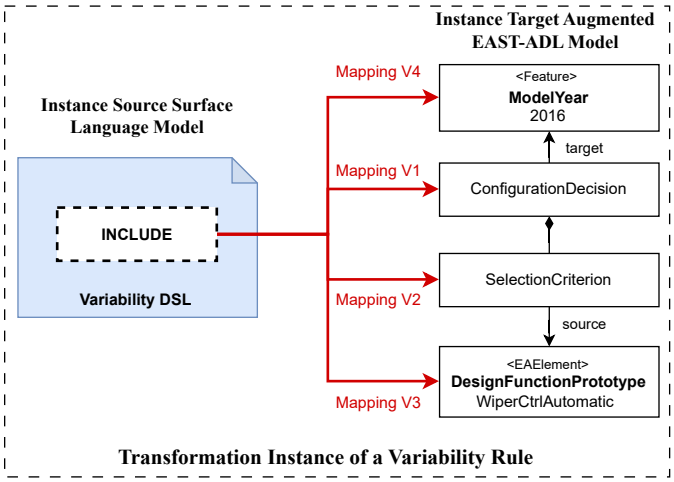

Fig. 4. Example of an Instance of the Transformation Mapping on a Variability Rule in the Running Example: the transformation rule selects the **INCLUDE** construct as an entry point to instantiate the mappings; the target augmented model is instantiated with the information specified within this construct.

their equivalents in the EAST-ADL model. The left-hand side consists of the source surface language model, showing that the **function** construct is mapped into EAST-ADL using four mappings. First, Mapping M1 creates a ReactionConstraint, which is a type of timing constraint defined in EAST-ADL for specifying the duration within which a corresponding response must occur following the occurrence of a stimulus. Then, Mapping M2 creates the composition of a TimingExpression , which specifies the minimum *10ms* for the reaction time, while Mapping M3 sets the maximum of *100ms* using another TimingExpression. Finally, Mapping M4 associates the ReactionConstraint with an EventChain, another EAST-ADL concept, allowing the specification of the stimulus Event and the response Event.
This transformation procedure is applied to each construct of the variability and timing DSLs. The transformation rules aim to reproduce the same outcomes as EAST-ADL while using a separate DSL. The DSL benefits from a lightweight syntax and a clear separation of concerns.

VII. Discussion

The proposed framework offers a structured and systematic approach to separating concerns within EAST-ADL. Designed by leading automotive experts, EAST-ADL is highly versatile and tailored to address the broad requirements of automotive systems, making it well-suited for designing their structural features. On the other hand, DSLs enable a more concise and streamlined specification of variability and timing by isolating these concepts into a distinct layer, thereby avoiding the complexity of EAST-ADL's heavy tooling.

This framework provides an opportunity to explore EAST-ADL through multiple abstractions while relying on model transformation to ensure consistency with EAST-ADL's standardized concepts. While we present a proof of concept for the specific case outlined in the running example, further

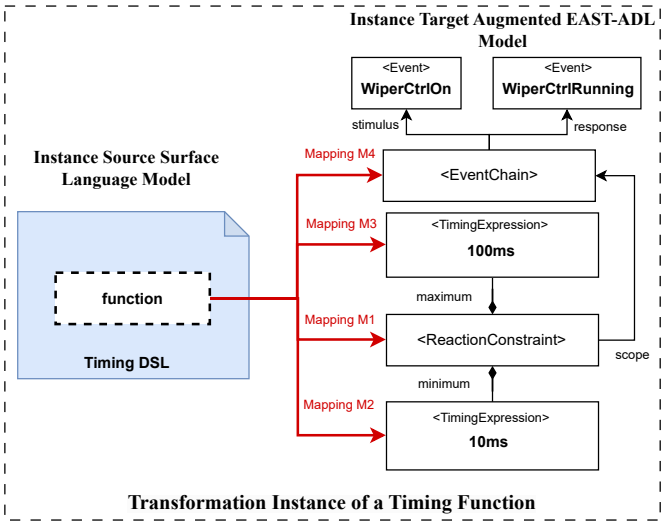

Fig. 5. Example of an Instance of the Transformation Mapping on a Timing Function in the Running Example: the transformation rule selects the **function** construct as an entry point to instantiate the mappings; the target augmented model is populated using the information specified in the function.

applications to other scenarios are necessary to comprehensively evaluate its benefits and implications in production environments.

The main challenge posed by this framework is its integration with the existing toolchain. Indeed, several tools (e.g., Eatxt [20], EATOP [27], GefEditor [28]) have been developed recently to support the EAST-ADL ecosystem. These tools address particular concerns within the ecosystem. Eatxt offers a textual representation of EAST-ADL, EATOP provides a tree view of the EAST-ADL model, and GefEditor delivers a graphical view. However, their lack of maturity leads to instability, complicating seamless integration and interoperability with the proposed framework. Changing the EAST-ADL model means creating new abstractions.

We consider the proposed framework as an initial solution for separating concerns of specific modeling aspects in EAST-ADL. It identifies the essential components necessary for this separation and offers concrete guidance that will pave the way for an evaluation of the approach. In the ongoing implementation, we are focused on defining and testing a set of separate Xtext DSLs that address specific modeling aspects, enabling seamless integration with the current EAST-ADL tooling and providing mechanisms for applications such as variability resolution.

VIII. Conclusion

We presented a framework for transforming compact surface languages into augmented EAST-ADL models. The framework relies on a model-to-model transformation procedure that maps the concepts of the surface language model into their equivalents in EAST-ADL. Using an instance of a transformation, we demonstrated how this mapping can be performed based on a running example. Model transformation bridges the gap created by the separation of these aspects from the original core model. The primary benefit of the proposed framework is the separation of concerns. Its implementation can be achieved using existing model-based tools such as EMF, Xtext, and ATL.

In future work, we plan to apply the transformation to other aspects such as safety, explore more consistent transformation rules to ensure full equivalence between DSLs and EAST-ADL, and implement a comprehensive and modular transformation framework.

Acknowledgments

This research work has been funded by the Swedish Knowledge Foundation through the MoDEV project (20200234), by Vinnova through the iSecure (202301899) and AIDA (202402068) projects, and by the KDT Joint Undertaking through the MATISSE project (101140216).

References

[1] H. Lonn, "Far EAST: modeling an automotive software architecture using the EAST ADL," in *"Software Engineering for Automotive Systems" Workshop W14S - 26th International Conference on Software Engineering*. IEE, 2004.

[2] S. Fürst, J. Mössinger, S. Bunzel, T. Weber, F. Kirschke-Biller, P. Heitkämper, G. Kinkelin, K. Nishikawa, and K. Lange, "Autosar–a worldwide standard is on the road," in *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*, vol. 62, no. 5. Citeseer, 2009.

[3] H. Blom, D.-J. Chen, H. Kaijser, H. Lönn, Y. Papadopoulos, M.-O. Reiser, R. T. Kolagari, and S. Tucci, "East-adl: An architecture description language for automotive software-intensive systems in the light of recent use and research," *International Journal of System Dynamics Applications (IJSDA)*, vol. 5, no. 3, pp. 1–20, 2016.

[4] A. Bucaioni, S. Mubeen, A. Cicchetti, and M. Sjödin, "Exploring timing model extractions at east-adl design-level using model transformations," in *2015 12th international conference on information technology-new generations*. IEEE, 2015, pp. 595–600.

[5] Dejiu Chen, "Towards the Integration of UPPAAL for Formal Verification of EAST-ADL Timing Constraint Specification," 2011.

[6] E. Kang, "A Formal Verification Technique for Architecture-based Embedded Systems in EAST-ADL," *arXiv.org*, 2019.

[7] R. Marinescu, M. Saadatmand, A. Bucaioni, C. Seceleanu, and P. Pettersson, "A model-based testing framework for automotive embedded systems," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2014, pp. 38–47.

[8] E.-Y. Kang, G. Perrouin, and P.-Y. Schobbens, "Model-based verification of energy-aware real-time automotive systems," in *2013 18th International Conference on Engineering of Complex Computer Systems*. IEEE, 2013, pp. 135–144.

[9] A. Schwarz *et al.*, "Uml-based modeling approach for automotive system development," in *Proceedings of the International Conference on Software Engineering*. IEEE, 2006. [Online]. Available: https://ieeexplore.ieee.org/document/1600680

[10] R. Maschotta, A. Wichmann, A. Zimmermann, and K. Gruber, "Integrated automotive requirements engineering with a sysml-based domain-specific language," in *2019 IEEE International Conference on Mechatronics (ICM)*, vol. 1. IEEE, 2019, pp. 402–409.

[11] I. Berrouyne, A. Bucaioni, F. Ciccozzi, and H. Lönn, "Towards compact surface languages for specific modelling aspects in east-adl," in *12th Embedded Real-Time Systems Congress*, June 2024. [Online]. Available: http://www.ipr.mdu.se/publications/6901-

[12] T. N. Qureshi, D.-J. Chen, M. Persson, and M. Törngren, "Towards the integration of uppaal for formal verification of east-adl timing constraint specification," in *TiMoBD workshop*, 2011.

[13] T. Qureshi, Dejiu Chen, Magnus Persson, and Martin Törngren, "Towards the Integration of EAST-ADL and UPPAAL for FormalVerification of EAST-ADL Timing Constraint Specification," 2011.

[14] Eduard Paul Enoiu, R. Marinescu, C. Seceleanu, and P. Pettersson, "Vital: A Verification Tool for EAST-ADL Models Using UPPAAL PORT," *IEEE International Conference on Engineering of Complex Computer Systems*, 2012.

[15] A. Bucaioni, A. Cicchetti, and M. Sjödin, "Towards a metamodel for the rubus component model." in *ModComp@ MoDELS*. Citeseer, 2014, pp. 46–56.

[16] AUTOSAR Consortium, *AUTomotive Open System ARchitecture (AUTOSAR) Specification*, AUTOSAR Consortium, 2022, release 19-11. [Online]. Available: https://www.autosar.org/standards/

[17] R. Flores, D. Rosa, and M. Murugesan, "Using model transformation/-code generation technology to migrate legacy software assets to autosar," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 4, no. 2011-01-1264, pp. 10–16, 2011.

[18] H. Giese, S. Hildebrandt, and S. Neumann, "Model synchronization at work: keeping sysml and autosar models consistent," *Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*, pp. 555–579, 2010.

[19] A. Daghsen, K. Chaaban, S. Saudrais, and M. Shawky, "Model transformation and scheduling analysis of an autosar system," in *The 2011 International Conference on Embedded Systems and Applications (ESA 2011)*, 2011, pp. 54–59.

[20] J. Holtmann, J.-P. Steghöfer, and W. Zhang, "Exploiting meta-model structures in the generation of xtext editors." in *MODELSWARD*, 2023, pp. 218–225.

[21] A. Bucaioni, F. Di Silvestro, I. Singh, M. Saadatmand, H. Muccini, and T. Jochumsson, "Model-based automation of test script generation across product variants: a railway perspective," in *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE, 2021, pp. 20–29.

[22] E. Ferko, A. Bucaioni, J. Carlson, and Z. Haider, "Automatic generation of configuration files: an experience report from the railway domain." *J. Object Technol.*, vol. 20, no. 3, pp. 4–1, 2021.

[23] M. W. Anwar, F. Ciccozzi, and A. Bucaioni, "Enabling blended modelling of timing and variability in east-adl," in *Proceedings of the 16th ACM SIGPLAN International Conference on Software Language Engineering*, ser. SLE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 169–180. [Online]. Available: https://doi.org/10.1145/3623476.3623518

[24] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: eclipse modeling framework*. Pearson Education, 2008.

[25] M. Eysholdt and H. Behrens, "Xtext: implement your language faster than the quick and dirty way," in *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, 2010, pp. 307–309.

[26] F. Jouault and I. Kurtev, "Transforming models with atl," in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2005, pp. 128–138.

[27] S. Baci, H. Kaijser, H. Lönn, M. Tichy, and W. Yuan, "Tool assisted model based multi objective analyses of automotive embedded systems," *safety*, vol. 10, no. 11, p. 12, 2015.

[28] D. Rubel, J. Wren, and E. Clayberg, *The Eclipse Graphical Editing Framework (GEF)*. Addison-Wesley Professional, 2012.