Extending EAST-ADL for Modeling and Analysis of Partitions on Functional Architectures

Christoph Etzel and Bernhard Bauer
Institute of Computer Science, University of Augsburg, Germany

Keywords: System Architecture, Model-driven Systems Engineering, Automotive Systems Engineering, EAST-ADL.

Abstract: The complexity in automotive systems engineering is increasing over the last decade. In particular, new comfort functions as well as functions towards autonomous driving are reasons for this complexity. A new dimension is introduced by the usage of multi-core processors since there is a shift from sequential to parallel thinking in the different development phases. Therefore, in this paper we present an approach for supporting the development process of distributed systems aligned with the EAST-ADL approach, by using partitioning. We present an extension to EAST-ADL for partitioning and show ways how an automatic partitioning on different levels of abstraction can be achieved. These partitions can support system designers during the design process of functional architectures, by giving a first insight how well the functional components can be distributed on hardware in later stages of the development process.

1 INTRODUCTION

In recent years, automotive systems are containing an increasing part of hard- and software systems resulting in huge distributed system. Already in 2007 an BMW 7 contained 67 embedded devices providing 270 functions interacting with the user (Pretschner et al., 2007). Up to 100 Electronic Control Units (ECUs) have been placed in premium vehicles in 2013 (Lukasiewycz et al., 2013). With the development of autonomous vehicles, the complexity will further increase, assuring a safe and comfort driving experience. To lower the system complexity in vehicles, multi-core systems each replacing several single-core ECUs are an intensive discussed topic and first vehicles are on the road using multi-core architectures (Arbeitskreis Multicore, BICCnet Innovationszirkel Embedded Systems, 2011), (Macher et al., 2015). The upcoming multi-core systems in the embedded domain need a "parallel thinking" already from the beginning of the system development. The hardware and software for such systems have to be highly optimized, to be competitive. This makes it more important to have a proper design on early, more abstract, design steps. To design embedded systems many different models and stakeholders are necessary and active during the design process (Gajski et al., 2009). Concerning multi-core system development, most projects use a bottom-up approach with the shortcomings of understanding the big picture of the system to allow a detailed analysis of the whole system (Macher et al., 2015). Moreover, in such a development process several stakeholders with different views and concerns may be involved.

Using a model based approach to provide all stakeholders with customized views during the development process helps them in the course of achieving their engineering and optimization tasks. This can be supported by using different abstraction levels, starting with a high level view in the early stage to a more detailed on a technical view in a later stage, to manage complexity. The architecture description language EAST-ADL supports such an approach by providing modeling means to express requirements, features, functional components, timing and safety constraints and other engineering related information.

Our approach tries to support the system designer during his architectural design decisions with a focus on "parallel thinking", starting from the requirements, analyzing and parallelization of the logical and component architectures to achieve partitions, based on data dependencies, cycles etc. The partitioning shall give the system designer a closer insight on how well the architecture can be distributed, e.g., without having a high communication overhead. This will be supported by calculating key figures of partitions. In addition, the set of components in a partition may be executed independently from components in other par-

Etzel, C. and Bauer, B.
Extending EAST-ADL for Modeling and Analysis of Partitions on Functional Architectures.
DOI: 10.5220/0007688301670176
In *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2019)*, pages 167-176
ISBN: 978-989-758-358-2

titions. This allows the system designer to choose a suitable hardware architecture for the developed logical architecture. At current the EAST-ADL release has no modeling notations to express partitions independent of the functional composition modeling. Therefore, we introduce a possible extension of the EAST-ADL and present two partitioning algorithms.

This paper is structured as follows. Section 2 gives an introduction to EAST-ADL and the used partitioning algorithms to obtain parallel systems. In Section 3 we present our approach. This is followed (Section 4) by the extension of EAST-ADL by meta-model elements to handle partitioning information. Section 5 shows how the algorithms are applied to our different target abstraction levels to find partitions and which parameters can be extracted from the partitions. The approach is evaluated in a case study showing a brake-by-wire system example (Section 6). The paper completes with the conclusion and giving an outlook for further research.

2 PRELIMINARIES

This section introduces already existing languages, methods and algorithms used in this paper.

2.1 EAST-ADL

The EAST-ADL (Blom et al., 2016) is an Architecture Description Language with the focuses on the automotive embedded electronic system engineering problem domain, currently developed by the EAST-ADL Association (EAST-ADL Association, 2018). It is aligned with the established AUTomotive Open System ARchitecture (AUTOSAR) standard (AUTOSAR, 2018) and extends it with higher levels of abstraction. While AUTOSAR's most abstract concept is the software architecture, the EAST-ADL provides means to model the system architecture and capture essential engineering information on this stage.

For this purpose the current release of the EAST-ADL2 (EAST-ADL Association, 2013) has four levels describing the system on different abstraction levels and viewpoints (see left hand side of Figure 1. The **Vehicle Level** includes a Technical Feature Model of the electric and electronic system. It is a software product line architecture by using decomposition and variability to allow different feature configurations. The **Analysis Level** includes the Functional Analysis Architecture (FAA). Features of the Vehicle Level are realized by abstract functions and connected through devices (e.g., sensors or actuators) to the vehicle environment. The **Design Level** includes the Functional Design Architecture (FDA) and the Hardware Design Architecture (HDA). On these level the abstract functions of the FAA are decomposed with implementation-oriented aspects including middleware and hardware abstraction. The hardware design defines physical resources and their connection. Functions from the FDA are allocated on entities of the HDA. The **Implementation Level** is the connection to the AUTOSAR system model. The elements of the Design Level are mapped to AUTOSAR entities (Qureshi et al., 2011). This supports traceability over the whole development process.

Beside these abstraction levels, EAST-ADL is extended by several cross-cutting concern extensions, which span over these layers. Examples for these extensions are Environment, Requirements, Variability and Timing. Figure 1 shows the abstraction levels with their models horizontal and the extensions (cross-cutting concerns) are vertically aligned over all levels. The red circled extension *Partitioning* and methods on the right side of the figure is our contribution, which is subject of this paper. For this reason, a short description how the Functional Analysis Architecture (FAA) and the Functional Design Architecture (FDA) are structured is provided. Both architectures contain a component based architecture model to describe the system. *FunctionType*s are abstract function component types to structure the system. An instance of a *FunctionType* is a *FunctionPrototype*. A *FunctionType* contains *FunctionPort*s, which can be connected together using *FunctionConnector*s. To enable hierarchical architectures, the specializations of *FunctionType* on the Analysis and Design Level (*FunctionAnalysisType* and *FunctionDesignType* can own parts in form of *FunctionAnalysisPrototype*s respectively *FunctionDesignPrototype*s. The *FunctionConnector*s between owned prototypes are called assembly connection, while a connection between a port of the type itself and a prototype is called delegation connection.

The Design Level includes information of the hardware system in the Hardware Design Architecture (HDA). The HDA defines the connectivity, capabilities and basic safety characteristics of technical architectures, e.g., the number of processors, cores and the communication matrix. These information can be expressed using the following elements from the EAST-ADL modeling language: An execution unit (ECU) is a *HardwareComponentType*, typed as a *Node*. Cores are modeled as contained *HardwareComponentType*s of type *Node* inside an ECU. The bus system type and its speed is defined using ports (*HardwarePortConnector*s) on every ECU. The port connectors are linked together using *HardwareCon-*

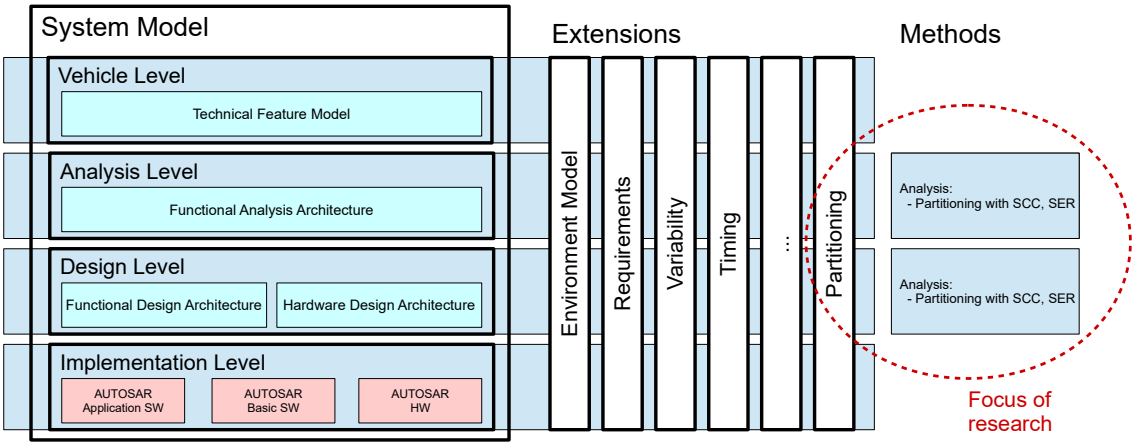

Figure 1: EAST-ADL abstraction levels with the containing models and cross cutting extensions. On the right hand side the partitioning extension and methods provided in this research.

*nector*s forming the bus system.

The EAST-ADL supports a linking between elements of different abstraction levels using the *Realization* relationship. This allows a full traceability over all abstraction levels.

In addition, it should be noted that the levels of the EAST-ADL can be seen as the equivalents to usual developing phases. E.g., the correspondence to the V-Modell XT (Weit e.V., 2018) is as follows: the Vehicle Level, including its feature models, is part of the systems requirements analysis; the Analysis and Design Level are used in the system analysis, system architecture and system design phase; the Implementation Level belongs to the software architecture phase. This last phase is not part of the paper, but listed to have a complete and sound overview. The ATESST2 project released a methodology guideline (The ATESST2 Consortium, 2010) for the development with EAST-ADL2. It defines a top-down development process and we embed our partitioning analysis into it.

2.2 Partitioning Algorithms

To partition the architectures, two different algorithms have been chosen. The strongly connected components algorithm is a classical algorithm from graph theory, while the extended single entry region algorithm is a more recent publication dedicated to AUTOSAR systems.

2.2.1 Strongly Connected Components

An algorithm to find highly interconnected nodes in directed graph is called strongly connected components (SCC) analysis. A (sub-)graph is called strongly connected if there exists a path between each pair of nodes. A set of strongly connected components of a graph form a partition representing a strongly connected subgraph. Tarjan (Tarjan, 1972) presents definitions for strong connectivity in graphs and an algorithm for computing the strongly connected components. An application of the SCC algorithm on system of systems to support architectural decision making is shown by (Potts et al., 2017). Since feedback loops are a common pattern in designing control systems, we see potential in applying the SCC on the EAST-ADL architectures.

2.2.2 Single Entry Region

The Single Entry Region (SER) analysis is introduced by Kienberger et al. in (Kienberger et al., 2014) and further refined in (Kienberger et al., 2016). It is based on work of (Ottenstein and Ottenstein, 1984), (Johnson et al., 1994), (Tip, 1995) and (Gotz et al., 2009). The idea is to identify regions having a loose coupling to other parts of the system and therefore be kind of isolated. Kienberger et al. describe SER as follows for data dependencies between nodes:

- The number of nodes is greater or equal to two.
- All input dependencies from nodes outside the SER are routed over a single “entry node”.
- Every node inside an SER is reachable from the entry node, either direct or transitive.

It is performed on an AUTOSAR system description model, namely on the component based architecture formed by “Runnable Entities” (AUTOSAR’s atomic executable and schedulable units) and their data dependencies. From our point of view, the analysis can be used for every appropriate type of dependency in a graph. Since the AUTOSAR system description model is derived from the FDA on EAST-ADL’s

Design Level and the FDA and FAA are component based architectures with data dependencies, we adopt the SER analysis to EAST-ADL.

3 APPROACH

Our goal is to support the system designer during his architectural design decisions in order to have an architectural model, that is well suited for further fine-grained development. Partitioning, in our context, is the process of grouping the system under development (SUD) into different parts without changing its functional component based architecture. Therefore, modeling elements are defined and used to structure the system model elements based on the EAST-ADL modeling concepts. The partitioning provides additional views on the SUD, depending on which criteria it is performed. The main focus is on partitioning of the Analysis Level as well as the Design Level.

Using the EAST-ADL, the goal is to keep the EAST-ADL meta-model unchanged and extend it only minimal invasive. This is achieved by introducing a new extension called "Partitioning" (see Figure 2), where meta-model elements are defined to express partitions and store additional information helpful for an analysis of the SUD, while not modifying the FAA on the Analysis Level or the FDA on the Design Level. For example, on the Analysis Level, the SUD is described by the FAA by using functional devices and analysis functions. The functional devices are the connection to the environment; using sensors to get data from the environment and actuators to interact with it. Typically *AnalysisFunction*s link sensors to actuators, by performing calculations on the sensors data and react accordingly through the actuators. The connection between the devices and functions is modeled by ports to provide and receive data, which are linked together with function connectors. This component based description of the architecture together with additional data defined in the extension is used to determine partitions.

The extension of EAST-ADL and the analysis are implemented in our tool. It is based on the Eclipse Modeling Framework[1], the Model Analysis Framework[2], EATOP[3] and Artop[4].

[1]Eclipse Modeling Framework (EMF) https://www.eclipse.org/modeling/emf/

[2]Model Analysis Framework - Data-flow based model analysis (MAF) https://www.informatik.uni-augsburg.de/en/chairs/swt/ds/projects/mde/maf/

[3]EATOP Project https://www.eclipse.org/eatop/

[4]AUTOSAR Tool Platform (Artop) https://www.artop.org/

4 EXTENDING EAST-ADL WITH PARTITIONING MODELING

Our approach has a strong focus on partitioning of the different abstraction levels provided by EAST-ADL. Since EAST-ADL structures the system model into different abstraction levels we introduce distinct partitioning models for all levels of abstraction. While these models are different on every abstraction level, the meta-model elements are shared to support a common handling of partitioning in every use case. The newly introduced elements are derived from already specified elements in the EAST-ADL to be compatible with it. In the following definitions, most elements from the EAST-ADL meta-model can be identified by the prefix "*EA*", for example, *EAElement* and *EANumericalValue* from the EAST-ADL infrastructure package. If an element has no prefix, it will be indicated in the text. Building up on basic elements of the EAST-ADL, allows the use of EAST-ADL realization links to connect elements across different abstraction levels to achieve full traceability.

The developed meta-model elements to capture partitions are visualized in Figure 2. The root of the new elements is *PartitionModel* which links the architectural description of one EAST-ADL level to the partitioning architecture. It is derived from *EAElement*, which is an abstract meta class of the EAST-ADL meta-model, defining an identifiable and named element of the domain model. It can be identified by using a global unique identifier called UUID, has an expressive name and can contain comments for additional descriptions. The partition model contains two associations the *targetLevel* and *partitionArchitecture*. The *targetLevel* is used to link the partition model to the level it partitions; i.e., the analysis or design level object which are of the EAST-ADL meta-model type *SystemModel*. The association *partitionArchitecture* points to the root package of the partition architecture. Since the partitioning is done independently on every abstraction level, only elements which are part of the target level are allowed to be linked in the partition architecture and its children.

Name: PartitionModel
Description: The *PartitionModel* is used to organize the partition architecture of an abstraction level.
Generalizations: EAElement
Attributes: No additional attributes.
Associations
- targetLevel: SystemModel [1]
- partitionArchitecture: PartitionPackage [1]

Constraints: All (nested) referenced elements in the *partitionArchitecture* shall be part of the refer-

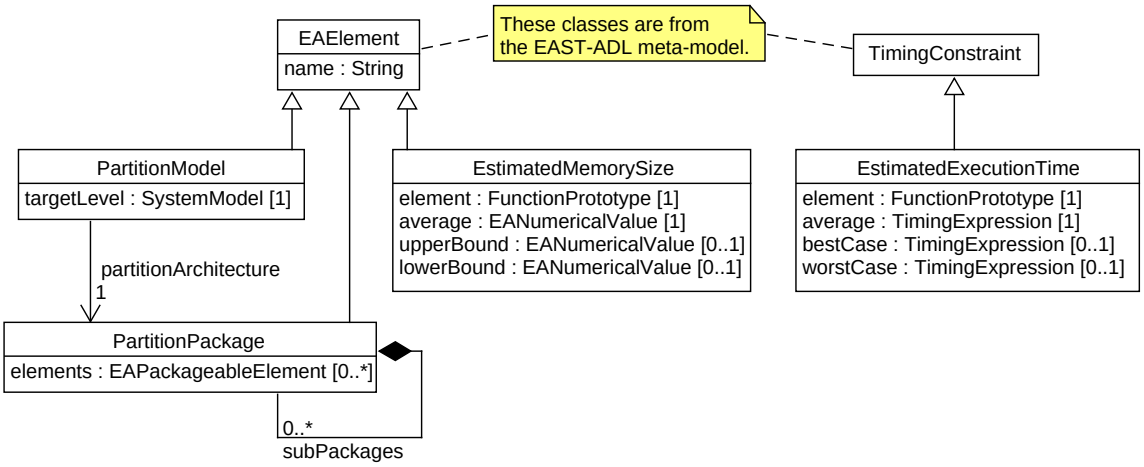

Figure 2: The "Partitioning" meta-model extension.

enced *targetLevel*.

Semantics: *PartitionModel* is the representation of a nested set of partitions for a specific system design level.

*PartitionPackage*s are used to assemble elements into partitions. Even if it looks very similar to the description of the EAST-ADL meta-model element *EAPackage*, it cannot be derived from it. The reason is that an *EAPackage* uses a composition to aggregate the containing elements, while a *PartitionPackage* shall only provide an association to the elements in the architecture. The *subPackages* association contains sub partitions and is realized using a composition. A *PartitionPackage* can contain multiple elements and packages to achieve a hierarchical partition architecture.

Name: PartitionPackage

Description: The PartitionPackage is used to form partitions of elements.

Generalizations: EAElement

Attributes: No additional attributes.

Associations

- elements: EAPackageableElement [0..*]
- subPackages: PartitionPackage [0..*] {comp.}

Constraints: No additional constraints

Semantics: *PartitionPackage*s can be used to organize *EAPackageableElement*s which form a partition. The packages can be structured hierarchically, where each level may contain variable number of *EAPackageableElement*s and sub packages forming sub partitions.

In the EAST-ADL, the timing extension for example, defines constraints and other modeling elements to specify important requirements in the architecture. Additionally to the structural description of the partitioning, modeling elements are introduced to enrich the functional architecture with information helpful for analyzing in respect to partitioning. To allow a more accurate partitioning of an architecture, two additional elements are defined, describing estimated values of memory and execution time.

The estimated memory size can be captured by using the newly introduced meta-model element *EstimatedMemorySize*. It has an association to an element in the system model and three values describing its estimated average memory size in bytes and optional upper/lower bound values to define a spectrum the memory size varies.

Name: EstimatedMemorySize

Description: The estimated size of memory used by the function in bytes.

Generalizations: EAElement

Attributes: No additional attributes.

Associations:

- element: FunctionPrototype [1]
- average: EANumericalValue [1]
- upperBound: EANumericalValue [0..1]
- lowerBound: EANumericalValue [0..1]

Constraints: If set, the values shall comply to $lowerBound \leq average \leq upperBound$.

Semantics: The *EstimatedMemorySize* stores the estimated or measured average memory size in bytes and optional an upper/lower bound.

The EAST-ADL timing extension describes an execution time constraint specifying the upper and lower bound run-time of an event. We introduce an *EstimatedExecutionTime* element, storing estimated or measured average execution time of a function and optionally a best and worst case value. It is derived from the element *TimingConstraint* and uses *TimingExpression*s to express the containing values. Both elements are from the EAST-ADL timing package.

Name: EstimatedExecutionTime

Description: The estimated execution time of the function.

Generalizations: TimingConstraint
Attributes: No additional attributes.
Associations

- element: FunctionPrototype [1]
- average: TimingExpression [1]
- bestCase: TimingExpression [0..1]
- worstCase: TimingExpression [0..1]

Constraints: If set, the values shall comply to $bestCase \leq average \leq worstCase$.
Semantics: The *EstimatedExecutionTime* stores the estimated or measured values of the average execution time and optional a best/worst case value.

In a SUD all three values have be in line with already defined execution time constraints.

5 ANALYZING EAST-ADL MODELS FOR PARTITIONS

In this section, we describe how the SCC and SER algorithms are used to automatic search for partitions on the architectures of the analysis and design level. Partitions are formed by sets of functional components and analysis is done independently on the analysis and design level. It is also possible to model partitions manually, by using the former described extension.

5.1 Parameters Influencing the Analysis

The analysis of the architectures can depend on more than considering data dependencies. We identified two different kinds of relevant clustering parameters in our use cases: communication between functions and resource usage of functions. Examples for the communication perspective are the coupling between functions or the utilization of connections. On the resource side execution time, execution frequency and memory consumption are values of interest. The newly introduced meta-model elements and already available elements in the EAST-ADL enable to consider these parameters in the analysis.

The **Data Flow Weight** describes the weight of the data exchanged on one connection between two nodes. Using EAST-ADL modeling language the size of the transferred data can be provided via *EADatatype* and the repetition of this transfer via function triggering stored in a *FunctionTrigger*. The **Function Computational Time Weight** combines our introduced *EstimatedExecutionTime* element to estimate the computing time in conjunction with function triggering to get an idea how a processor is utilized by a function. Using the newly specified *EstimatedMemorySize*, the **Function Memory Weight** (either the binary size or the resource usage during run-time including temporary memory usage) can be used.

These parameters can either be used in partition searching algorithms or to calculate key figures of a partition. E.g., partitions can be rated by their memory footprint summing up the **Function Memory Weight** of every component, or by their **Function Computational Time Weight**, if it is assumed that the set of one partition is executed sequentially. These key figures are indicators for the system designer to judge about the architecture and possibly perform a refactoring.

5.2 Analysis Level

The analysis level includes an abstract functional representation of the architecture captured in the FAA. This architecture is designed very early in the development process during the system analysis phase (The ATESST2 Consortium, 2010). From a methodology point of view, the partitioning shall be placed in the development process after the task to specify the analysis function details. The result of partitioning analysis can then be used to further refine the architecture in an iterative way.

Before starting the analysis on the FAA, we have implemented multiple model pre-checks in our tool, such as if all directions of the ports and the binding to the function connectors are reasonable. For example, if two functions are connected via “IN” ports a warning is raised. The same applies to “OUT” ports. Additionally, it should be noted that a client-server connection in the model is interpreted as a bi-directional connection between the components.

5.2.1 SCC Analysis

The first analysis implemented is the strongly connected component search. The directed graph consists of the analysis function prototypes as the vertices and the function connectors as the directed edges between the vertices. Since the SCC algorithm analyzes paths between the vertices only the communication between the functions is used to form partitions.

The results of the strongly connected component search is transferred into a partitioning model, where a set of strongly connected functions forms a partition. For every detected set with more than one component a *PartitionPackage* is created referring to the containing functions. An example with three graphs can be seen in Figure 3. The sets of strongly connected components enclosing more than one element are visualized with the same color. In the graph on

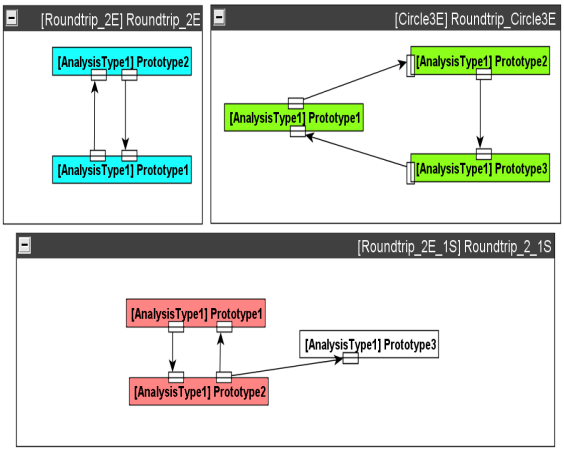

Figure 3: Three examples of graphs with strongly connected components.

the bottom of the figure is a single element "Prototype3" not colored (white background), since it forms a strongly connected set containing only itself and sets with just one element do not need a distinct color.

5.2.2 SER Analysis

Another implemented algorithm is the Single Entry Region (SER) analysis, which was developed for AUTOSAR system description models (Kienberger et al., 2016). A brief general description can be found in Section 2.2.2. We adapt the algorithm to fit to the EAST-ADL analysis level. For this purpose, every *AnalysisFunctionPrototype* contained in the FAA represents a node. The dependencies between the nodes are formed by the function connectors between the prototypes. The dependency weights are calculated by using the introduced **Data Flow Weight** parameter and summing it for every connection between a pair of nodes. The output of the algorithm are regions containing sets of *AnalysisFunctionPrototype*s. This gets transferred into the partitioning model such that every calculated region forms one partition.

5.3 Design Level

The design level includes an implementation-oriented functional model of the architecture captured in the FDA. Looking into the design process, the FDA is specified during the design phase in parallel with the HDA (The ATESST2 Consortium, 2010). This newly introduced partitioning step, shall be placed in the development process after the task to specify the design details, but before the allocation the functions to the HDA. The result of partitioning analysis can then be used to further refine the architecture in an iterative way and as an input artifact to the HDA allocation task.

Since the elements to model the FDA are very similar to the ones used for the FAA, the SCC and SER analysis are analogous to the analysis on the FAA. The graphs are formed by function prototypes and function connectors. Even the pre-checks and the handling of client-server connections are identical.

By using a partition model of our analysis an engineer can allocate functions to elements of the HDA. Elements grouped into one partition by these two algorithms are candidates to be allocated on one node, because they communicate with each other. Placing them on one node or closely connected nodes can reduce the communication overhead.

6 CASE STUDY - BRAKE-BY-WIRE SYSTEM EXAMPLE

To evaluate the proposed approach a case study on an example architecture is carried out. Since space is limited, only the SER analysis on the Analysis and Design Level is shown in detail. It should be noted, that the SCC analysis would not find partitions with more than one component in this particular example. Nevertheless, we picked this model, because it illustrates the SER analysis and the partition transition during the development process very clearly.

The "brake-by-wire for four wheel vehicles" model is originally from the EAST-ADL Association and published on their website[5].

The analysis architecture (see Figure 4) consists of 16 components and 26 connections between these. The main function is a *pGlobalBrakeController*, which gets data from 4 wheel speed sensors, the vehicle speed and the requested brake force. The vehicle speed is calculated by the *pVehSpeedEstimator* getting data from the wheel speed sensors. The vehicle speed is provided to the *pGlobalBrakeController* and the four ABS controllers. The brake force is calculated by the *pBrakeTorqueMap* with data from the *pBrakePedalSensor*. The four ABS controllers are sending data to each brake actuator.

The colored components in Figure 4 are proposed partitions of the SER algorithm. The upper green colored partition consists of two components (*pBrakePedalSensor* and *pBrakeTorqueMap*), the lower partitions are each formed by the ABS and the brake actuator of one wheel.

[5]Brake-by-Wire System II (http://www.east-adl.info/Resources.html) (accessed January 16, 2019)

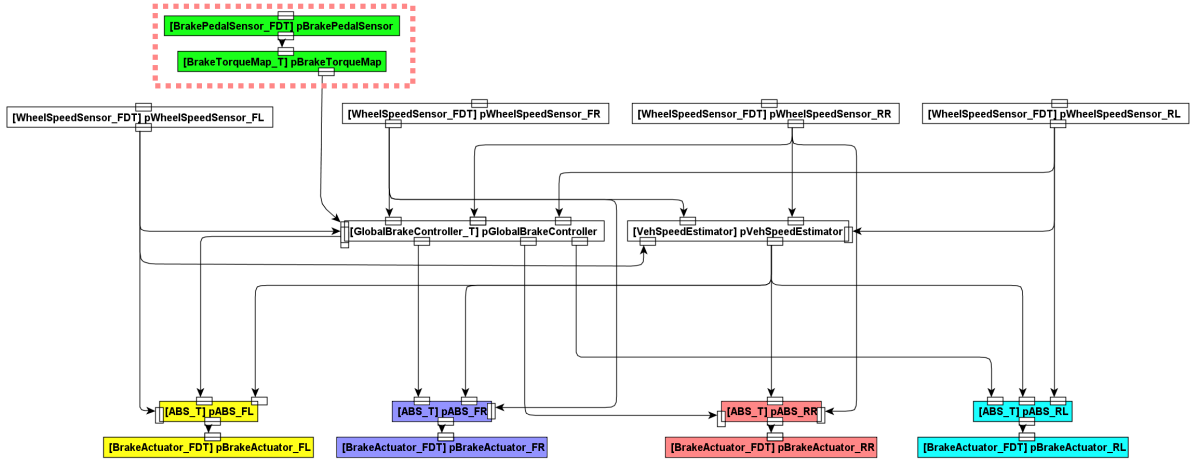

Figure 4: Functional Analysis Architecture of Brake-by-Wire Example with SER partitions of maximum size (colored elements).

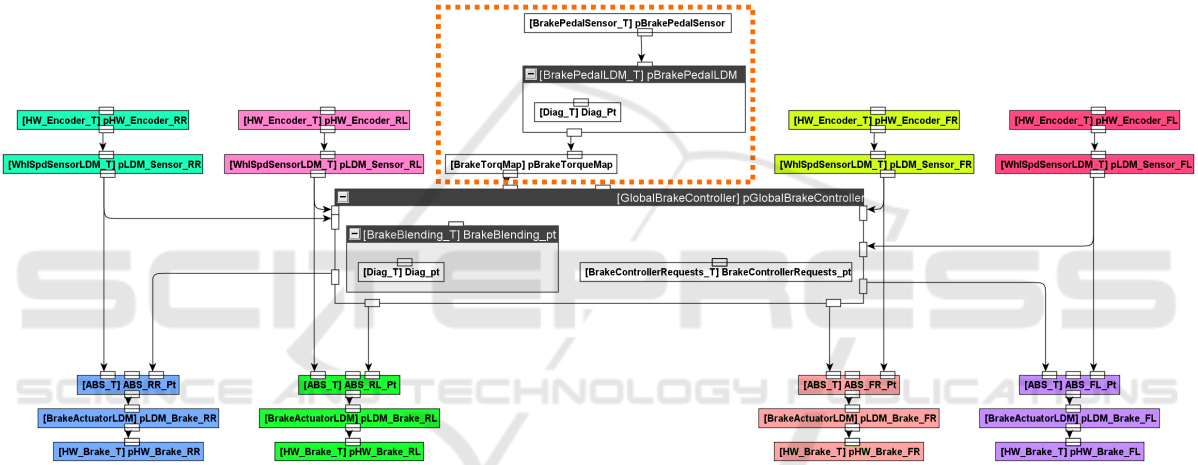

Figure 5: Functional Design Architecture of Brake-by-Wire Example with SER partitions of maximum size (colored elements).

The design architecture (see Figure 5) is derived from the functional analysis architecture. It contains 28 components and 27 connections (some components are for diagnoses, their connections outside the scope of this model example have been omitted). For example, a wheel speed sensor from the functional analysis architecture is now more detailed by using two components. One is a hardware encoder providing the digital hardware signal and the other is a local device manager (LDM) encapsulating the hardware device specific parts. On the actuator side, a similar detailing is performed by using a LDM and a hardware function component for the realization. Two components for diagnose tasks are also embedded in the example. One is a diagnose component in the *pBrakePedalLDM* and the other one in the *pGlobalBrakeController*.

The partitions found using the SER algorithm are very similar to the ones on the analysis architecture. On the bottom, every ABS component together with a LDM and the actuator form a partition. Four new partitions are originated from the decomposition of the wheel speed sensors into hardware encoders and LDMs. A difference can be seen looking at the former partition of the *pBrakePedalSensor* and *pBrakeTorqueMap*, which is for a better recognition marked with a square of orange dots in both figures. Because a diagnose component (*Diag_Pt*), which provides data to other components not visible in this figure, is embedded in the *pBrakePedalLDM*, it is not marked as a potential partition on this level. An option to in-/excluding diagnose components in the analysis is part of our framework. Turning it off, would clearly mark it as a partition containing the components *pBrakePedalSensor*, *pBrakePedalLDM* and *pBrakeTorqueMap*. Since there is no flag in the EAST-ADL

meta-model to identify diagnose components, we are using a naming schema (the prefix "*Diag_*") to recognize these components. Using our analysis results an engineer can check if the transition from the analysis architecture to the design architecture is sound (e.g., having a closer look, why one partition is now missing) and link the partitioned elements to elements of the HDA.

7 RELATED WORK

Marinescu et al. (Marinescu and Enoiu, 2012) propose a modeling extension for EAST-ADL and model analysis with the focus on resource-usage. The analysis are applied on the FDA using a priced timed automata to predict resource usage and optimizing resource utilization. In contrast to our approach, theirs is focusing on resource usage and allocation, while ours is proposing a general extension to describe partitions and algorithms focusing on the analysis of data dependencies. A mapping of parts of our extension to theirs is possible, e.g., *EstimatedMemorySize* (ours) to *MemoryConstraint* (theirs). In the development process, their resource-usage analysis is placed after ours during the development of the design level elements.

Automation of certain design processes steps enables system designers to focus on the challenging parts. Walker et al. (Walker et al., 2013) have developed a multi-objective optimization approach for EAST-ADL system architectures. Our algorithms could be connected into their framework by an *Analysis Wrapper* and provide results to the optimization engine. However, there has to be done further research how to derive quantitative criteria for the optimizer from the partitioning models.

8 CONCLUSION AND FURTHER RESEARCH

In this paper we presented an approach to support system designers during the development process by doing partitioning on functional architectures. Therefore, we extended the EAST-ADL meta-model minimal invasive with an extension to capture partitions. Beside the structural partitioning model, the extension introduces two elements to improve the search and rating of partitions. To support automated analysis, we presented the SCC and SER algorithm to search for partitions on the FAA and FDA. Both algorithms may be used without the partitioning extension, if no persistence of the partitions is needed to perform further analysis. Moreover, we applied the new approach to a small case study from the EAST-ADL consortium.

The first results concerning our approach are very promising and in the next steps we will evaluate it with additional scenarios. Moreover, we will develop more sophisticated algorithms for partitioning of functional models on these levels of abstraction. We think the proposed approach is not limited to the EAST-ADL modeling language and can be transferred to similar concepts even outside the automotive domain. Examples for other languages are SysML[6] and AADL[7], both strongly influenced the EAST-ADL specification (Blom et al., 2016).

ACKNOWLEDGEMENTS

This work was partially funded within the project ARAMiS II by the German Federal Ministry for Education and Research with the funding ID 01IS16025. The responsibility for the content remains with the authors.

REFERENCES

Arbeitskreis Multicore, BICCnet Innovationszirkel Embedded Systems (2011). Relevanz eines Multicore-Ökosystems für künftige Embedded Systems: Positionspapier zur Bedeutung, Bestandsaufnahme und Potentialermittlung der Multicore-Technologie für den Industrie- und Forschungsstandort Deutschland. URL: https://www.bicc-net.de/workspace/uploads/subfeatures/downloads/positionspapier_multicore_oeko-53077aea8dd0c.pdf.

AUTOSAR (2018). AUTOSAR website. https://www.autosar.org/ (accessed January 16, 2019).

Blom, H., De-Jiu, C., Kaijser, H., LÃ¶nn, H., Papadopoulos, Y., Reiser, M.-O., Kolagari, R. T., and Tucci, S. (2016). 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)*, 5(3):1–20.

EAST-ADL Association (2013). EAST-ADL Domain Model Specification. Version V2.1.12.

EAST-ADL Association (2018). EAST-ADL website. http://www.east-adl.info/ (accessed January 16, 2019).

Gajski, D. D., Abdi, S., Gerstlauer, A., and Schirner, G. (2009). *Embedded system design: Modeling, synthe-*

[6]Issued by the OMG, http://www.omgsysml.org/

[7]Issued by the SAE International, http://www.aadl.info/

sis and verification. Springer, Dordrecht and New York.

Gotz, M., Roser, S., Lautenbacher, F., and Bauer, B. (2009). Token analysis of graph-oriented process models. In *13th Enterprise Distributed Object Computing Conference Workshops*, pages 15–24.

Johnson, R., Pearson, D., and Pingali, K. (1994). Program structure tree: Computing control regions in linear time. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 171–185. ACM.

Kienberger, J., Minnerup, P., Kuntz, S., and Bauer, B. (2014). Analysis and validation of AUTOSAR models. In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, MODELSWARD 2014, pages 274–281, Portugal. SCITEPRESS - Science and Technology Publications, Lda.

Kienberger, J., Saad, C., Kuntz, S., and Bauer, B. (2016). Efficient parallelization of complex automotive systems. In Balaji, P. and Leung, K.-C., editors, *Proceedings of the 7th International Workshop on Programming Models and Applications for Multicores and Manycores*, pages 40–49. ACM.

Lukasiewycz, M., Steinhorst, S., Andalam, S., Sagstetter, F., Waszecki, P., Wanli Chang, Kauer, M., Mundhenk, P., Shanker, S., Fahmy, S., and Chakraborty, S. (2013). System architecture and software design for electric vehicles. In IEEE, editor, *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6.

Macher, G., Höller, A., Armengaud, E., and Kreiner, C. (2015). Automotive embedded software: Migration challenges to multi-core computing platforms. In *IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1386–1393.

Marinescu, R. and Enoiu, E. P. (2012). Extending EAST-ADL for modeling and analysis of system's resource-usage. In *IEEE 36th Annual Computer Software and Applications Conference Workshops*, pages 532–537.

Ottenstein, K. J. and Ottenstein, L. M. (1984). The program dependence graph in a software development environment. *SIGPLAN Not.*, 19(5):177–184.

Potts, M., Sartor, P., Johnson, A., and Bullock, S. (2017). Hidden structures: using graph theory to explore complex system of systems architectures. *International Conference on Complex Systems Design & Management. CSD & M*.

Pretschner, A., Broy, M., Kruger, I. H., and Stauner, T. (2007). Software engineering for automotive systems: A roadmap. In *Future of Software Engineering*, pages 55–71.

Qureshi, T. N., Chen, D., Lönn, H., and Törngren, M. (2011). From EAST-ADL to AUTOSAR software architecture: A mapping scheme. In Crnkovic, I., Gruhn, V., and Book, M., editors, *Software Architecture*, pages 328–335, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160.

The ATESST2 Consortium (2010). Methodology guideline when using EAST-ADL2. Deliverable D5.1.1 V1.1.

Tip, F. (1995). A survey of program slicing techniques. *Journal of Programming Languages*, 3:121–189.

Walker, M., Reiser, M.-O., Tucci-Piergiovanni, S., Papadopoulos, Y., Lönn, H., Mraidha, C., Parker, D., Chen, D., and Servat, D. (2013). Automatic optimisation of system architectures using EAST-ADL. *Journal of Systems and Software*, 86(10):2467–2487.

Weit e.V. (2018). V-Modell XT: Das deutsche Referenzmodell für Systementwicklungsprojekte Version 2.2.