FAR EAST: Modeling an Automotive Software Architecture Using the EAST ADL

Henrik Lönn and Tripti Saxena
Electronics and Software, Volvo Technology Corp., Gothenburg, Sweden

Martin Törngren
*Department of Machine Design
Royal Institute of Technology, Sweden*

Mikael Nolin,
*Mälardalen Real-Time Research Center
Mälardalen University, Sweden*

Abstract

We present a case study where a concept vehicle is re-modeled using a new systems modeling approach, the EAST ADL (Architecture Description Language). EAST ADL is a language for modeling and development of software based systems. The application domain is automotive software-based systems. The language has been developed within the project EAST-EEA by representatives of European automotive industries and academic research sites.

EAST ADL supports modeling during all stages of development of vehicle functions; from function selection, through function specification, to implementation of a running system. The language further supports modeling of aspects orthogonal to software structure, such as requirements, behavior, validation, and verification.

The FAR vehicle has been previously developed, using model based development. Within this paper we re-model the FAR vehicle using the EAST ADL. The new model ties together the various models, code and documentation in a consistent structure with clear relationships between entities.

1. Introduction

Electronics and software are firmly established in automotive systems, ever since the introduction of electronic engine management systems in the eighties. An increasing part of value, characteristics and new functionality is based on software. This trend has brought several challenges for the development of software based automotive systems.

Development of software and electronics for an automotive system is a large scale effort. A typical project can involve 10 first tier suppliers, require integration of 30 ECUs, and may consist of up to 300000 lines of code. The total effort is in the range of 50 person-years.

The integration of different systems is becoming tighter, and is particularly difficult when different suppliers are involved. The number of computers (ECU, electronic control unit) increases because most systems have their own set of ECUs. Each ECU introduces cost, fault-prone electrical connections and requires space. The increased complexity, integration and criticality of software based functions calls for a new approach to embedded software. This challenge is amplified by the necessity to share ECUs between systems.

The EAST-EEA [1] project addresses this challenge by investigating architecture and development aspects to support hardware independent software development under the new needs mentioned above. Most European car manufacturers, major suppliers and several research institutes are involved in the project. EAST-EEA is an ITEA project running for 3 years, ending 2004.

One of the aspects addressed by the project is the modeling of software based systems. It was recognized that adequate models are necessary to manage complexity, perform analysis and verification, make good allocation decisions, as well as support the supplier-manufacturer relationship. To this end, the EAST ADL (Architecture Description Language) has been designed to provide a unified language for the necessary models.

This paper gives an introduction to the EAST ADL and demonstrates its use to model the software and electronics architecture of the concept vehicle FAR, a by-wire scale car developed using state of the art development tools [2]. The modeling work was done as part of a thesis project at Volvo Technology Corp., Gothenburg.

2. Background

A key challenge in developing electronics and software for automotive systems is the provision of adequate modeling techniques and tools. Such techniques need to address the inherent product complexity in terms of a number the constituent entities (e.g. functions, tasks, ECU's and networks) and different relations (e.g. communication, refinement, variants and allocation) between them. In addition, such techniques need to explicitly support multidisciplinary development providing adequate abstractions and views for engineers using a diverse set of models, methods and tools. Moreover, such techniques need to consider both the description of functional requirements and non-functional constraints.

Model based development is well developed in mechanical engineering, however, much less so when it comes to embedded software. Several attempts have been made over the last decades to define suitable modeling languages and descriptions. A recent survey indicates that there still remains lots of work and research for modeling approaches [3]. Limitations include the lack of coverage of certain design stages (e.g. no functional description) and lack of support for certain design activities and analysis techniques (e.g. typically little support for non-functional constraints).

Important in any modeling approach is the explicit definition of the purpose of the models, the stakeholders, and consideration of requirements imposed by relevant design and analysis techniques.

Related work in the area of systems modeling is the SysML (Systems Modeling Language) by the OMG [4]. SysML is a domain-independent language for the higher abstraction levels of systems. Currently, the SysML does not specifically cover the implementation aspects of software, but relies on the recent UML2 standard. SysML is meant to be instantiated for particular domains to capture specific needs.

Another approach is the AADL, Avionics ADL [4]. AADL is a further development of the MetaH language. Compared to the EAST-ADL it focuses on the software architecture and single processor implementation related models. MetaH/AADL provides good support for real-time and reliability analysis compared to many other approaches [3].

3. The FAR Project

The concept car used in this work was developed in the FAR project. The FAR project was carried out in cooperation between Volvo Car Corporation and the Department of Machine Design at the Royal Institute of Technology (KTH). The project was implemented as part of a final-year project-course part of the Master of Science specialization in mechatronics at KTH. In the project, engineers and senior researchers at KTH and Volvo Cars supported a group of 10 students. The overall purpose of the FAR project was to develop a suitable tool-chain environment that supports model based development (MBD) of embedded control systems and in

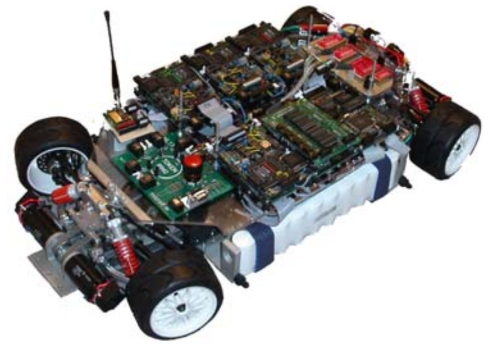
Figure 1. The FAR Vehicle.

particular Function and ARchitecture (FAR) integration. This tool chain should allow for rapid prototyping of automotive functionality with support for evaluation of electrical design concepts in distributed computer systems.

To demonstrate the usefulness of the FAR tool-chain, and to support spectacular demonstrations, a physical model car with four-wheel steering, individual wheel braking and four-wheel drive was also developed. Figure 1 shows the car.

Supporting distributed systems was one central aspect since tools for distributed control systems development are only very recently appearing.

Because the project was carried out as a more or less complete prototype development project, the results include everything from requirements specifications and function descriptions, to the software and hardware architectures and the mechanics of the vehicle itself.

UML Use cases [2] were used to model requirements in early design phases. It describes the functionality that the system should provide to the user. In Figure 3, the Use case diagram that was given as an input requirement to the project is shown. It details three main users and their purposes in the project.

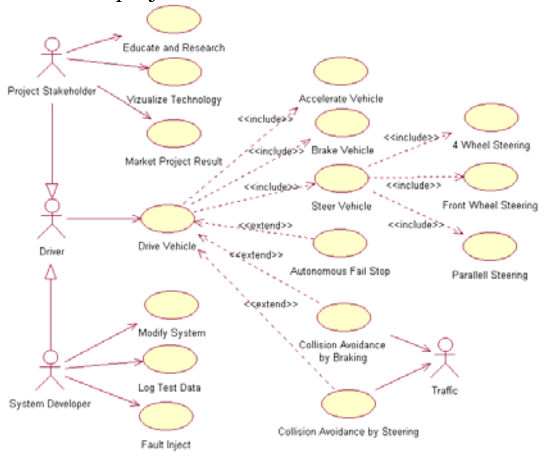

Figure 2. The Use Case Diagram for the project.

3.1 Functional View

Figure 3 shows the overall structure of the functionality. The HMI and actuators are physical instances, while global and local control are functionality implemented in software. Global control handles the vehicle as a complete system, such as collision avoidance functionality, while local control handles individual actuators. This separation of global and local control supports a high degree of reusability and separation of concerns.

3.2 Hardware View

To separate the driver's HMI and the car, two radio links are used. The HMI consists of one node, one video receiver, and a joystick or steering wheel with pedals. On the car there are six nodes connected through a TT-CAN network. One of the nodes is the central node, connected to the driver's HMI. Another node is the radar node used to measure distance to adjacent objects. The four wheel nodes control the local actuators at each wheel.

3.3 Software View and Tool chain

The software platform was developed using the Rubus real-time operating system, Rubus RTOS [6], and Time-triggered CAN (TTCAN). The application functionality was developed in the Matlab tools Simulink and Stateflow from Mathworks. Stateflow models mode logic for global control and Simulink models actuator coordination and local control, as shown in Figure 4. C-code is automatically generated from Simulink and Stateflow using the Targetlink code generation tool from dSPACE.

4. The EAST ADL

The goal of this work is to model the FAR vehicle in EAST ADL (Architecture Description Language) [7].

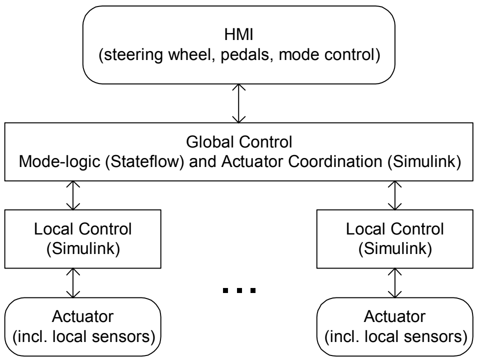

Figure 3. Functional view of the application.

The purpose of this language is to capture the software and electronics architecture with enough detail to allow modeling for documentation, design, analysis and synthesis. These activities require system descriptions on several abstraction levels, from top level user features down to tasks and communication frames in CPUs and communication links. Moreover, the activities also need to express non-structural aspects of the system under development; this includes aspects like requirements, behavior and validation & verification. The EAST ADL does not prescribe a process or methodology. Instead, the defined set of design artifacts may be developed using company specific processes.

Please note that the EAST ADL is work in progress. The approach may thus be adjusted before the project finalization in 2004.

4.1 Structure

The EAST ADL is structured into 7 layers, see Figure 5. The motivation for this layered structure is that the abstraction layers may evolve independently of each other with only a loose coupling through requirements entities and associations. In certain cases, entire abstraction layers may be omitted from a model, for example when legacy systems are re-modeled.

The EAST ADL abstraction layers from top to bottom are the following:

- *Vehicle View* describing user visible features such as anti-lock braking or windscreen wipers.
- *Functional Analysis Architecture* capturing the behavior and algorithms of the Vehicle View functions. There is an n-to-m mapping between Vehicle View entities and Functional Analysis Architecture entities, i.e. one or several functions may realize one or several features.
- *Functional Design Architecture* representing a decomposition of functionality in the Functional Analysis Architecture to meet constraints regarding allocation, efficiency, re-use, supplier concerns, etc.

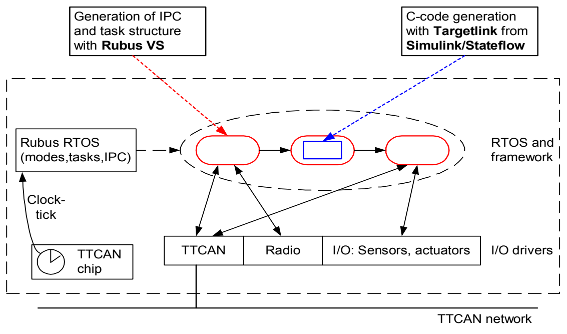
Generation of IPC and task structure with Rubus VS
C-code generation with Targetlink from Simulink/Stateflow
Rubus RTOS (modes,tasks,IPC)
RTOS and framework
Clock-tick
TTCAN chip
TTCAN
Radio
I/O: Sensors, actuators
I/O drivers
TTCAN network

Figure 4. Software view for one computer node illustrating the tools used and how functionality is integrated in the tasking structure.

Again, there is an n-to-m mapping between entities on Functional Design Architecture and Functional Analysis Architecture.

- *Function Instance Model* where the class representation of the Functional Design Architecture has been instantiated to a flat software structure suitable for allocation. The function Instance model contains the leaf functions of the Functional Design Architecture. The Function Instance Model could thus be auto-generated in many cases.

In parallel to the application functionality, the execution environment is modeled from three views:

- The *Hardware Architecture* contains ECUs, communication links, sensors and actuators and their connections.
- The *Platform Model* models the operating system or Middleware API and the services provided.
- The *Allocation Model* contains the tasks and frames that carry application code and –communication as well as the configuration information needed for the mapping of the application software to hardware.

Figure 5 illustrates the relations between layers.

4.2 Requirements, Behavior and V&V

Besides the structural decomposition, which is typical for any software development/modeling approach, the EAST ADL also has support for modeling cross-cutting concerns (i.e. concerns that are independent, or existing on a multitude, of different abstraction levels) like requirements, behavioral descriptions and validation & verification activities.

Requirements can be modeled either as textual descriptions or using formal descriptions. Requirements are not isolated in a view of their own, instead

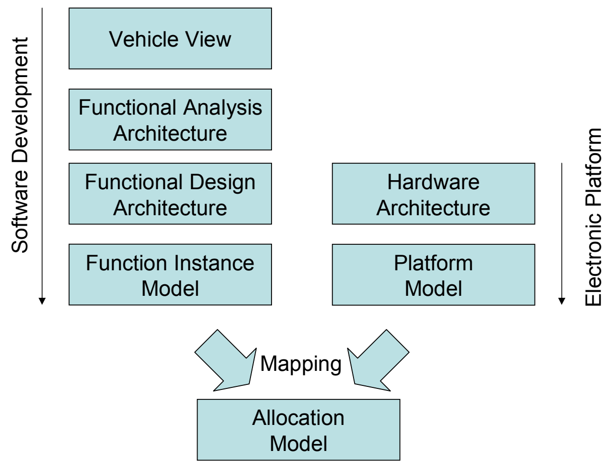

Figure 5. The layers of the EAST ADL

requirements can be attached to any ADL object. For instance, in the Vehicle View it may be natural to express end-user visible requirements on the functions specified in that view. Whereas in the function instance model, detailed requirements on, e.g., timing behavior such as signal delays can be naturally expressed. The requirements in EAST ADL are extension to the requirements package in SysML [4]. Requirements can have traceable dependencies amongst each other and can be decomposed into sub-requirements. The EAST ADL also has mechanism to handle variants and dependencies between variants. This can be used, e.g., to specify dependencies such that an automatic gearbox can only exist together with a certain set of engines.

In EAST ADL several types of behaviors can be modeled. For instance, behavior modeled in external tools like Simulink [9], Statemate [10] or Ascet-MD [11] can be attached to EAST ADL objects. This supports modeling, analysis and code synthesis of behavior in the way that it is currently used within the automotive industry. Also, UML 2.0 can be used to describe the behavior of an object. In this case, UML-statecharts can be used. From behavioral models code can be automatically generated. In addition, C code can be associated as a behavior description; in this case, however, no analysis of the behavior can be made, only mapping of the code to the ADL object is performed. The EAST ADL separates the concept of "specifying behavior" and "implementation behavior" in the sense that a software function can have a simple (incomplete) model as its specifying behavior, whereas the functions implementation behavior specifies its actual (and complete) behavior.

The final cross-cutting issue handled by the EAST ADL is validation & verification (V&V). The ADL has entities to specify what V&V methods should be used for different ADL objects. Also, the results of the V&V activities are stored within the ADL. This facilitates a holistic approach to system development, where requirements, system description and V&V are integrated in one language. Hence, it is easy to make sure that all requirements have been verified and what objects need to be accounted for when performing a particular verification task. The ADL also has entities to support automatic verification of key-properties. For instance, special requirements entities exists to allow expression of timing requirements, and attributes on software functions allow specification of timing behavior. Together, these requirements and attributes can be used to perform an automatic timing (schedulability) analysis. Also, for instance, using behavior specifications of objects, automatic verification of system behavior or system properties (such as absence of dangerous states) can be made. The EAST ADL does not specify which V&V activities that are to be automated, nor does it provide

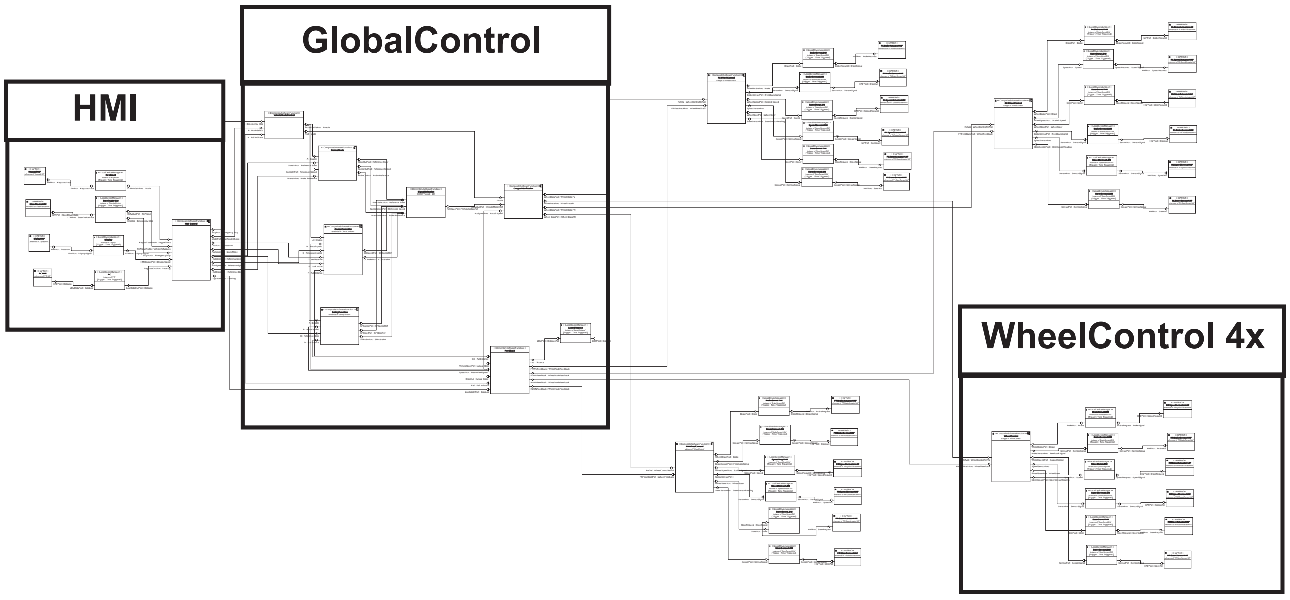

Figure 6. The Functional Design Architecture.

tools or methods for such automatic V&V. The ADL only provides the necessary information to perform V&V activities; how this information is used within an organization is unspecified.

5. The FAR Vehicle Model

The EAST ADL was applied to the software and electronics architecture of the FAR vehicle in order to explore the different concepts of the language. Explaining the models using a middle out approach, we start by describing the Functional Design Architecture, FDA. Figure 6 shows the overall structure of the FDA. The software structure of the FAR vehicle is shown here. Four CompositeSoftwareFunctions (hierarchical building blocks of the FDA) perform the local control of steer, brake and propulsion. These are instances of the same wheel controller. The global control is done by several CompositeSoftwareFunctions, such as ModeSelector, Feedback, CruiseControl, etc. The HMI interpret driver commands and feedback and distribute setpoint values to the control system.

Below, a selection of additional aspects from the FAR model will be described. We start from the top level with the vehicle type hierarchy.

5.1 Vehicle Types, Features and requirements

Vehicles generally have a complex variant structure. A small vehicle type hierarchy was identified for FAR, see Figure 7. This hierarchy is used to define the feature content of each vehicle type.

The various requirements on the vehicle are modelled on several abstraction levels. Figure 8 shows two examples of functional requirements associated with the Braking feature. Other examples of requirements are qualitative requirements like safety and performance or timing requirements, see Section 5.7.

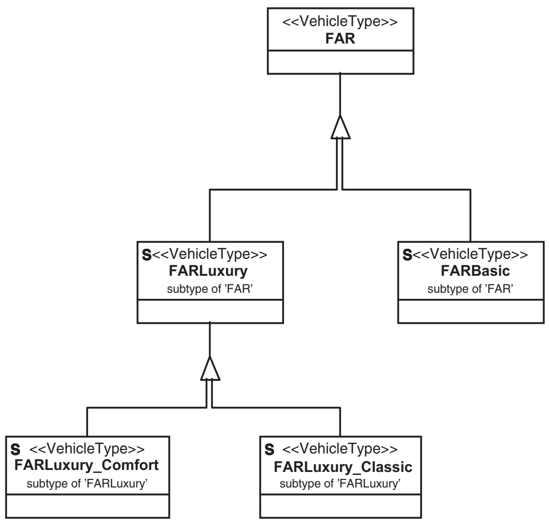

Figure 7. The VehicleType hierarchy (VV Layer)

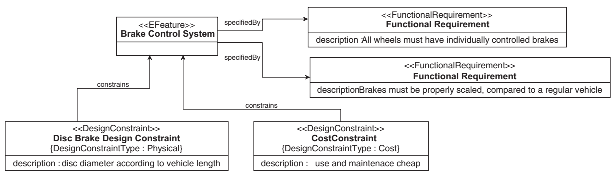
<<EFeature>>
Brake Control System
specifiedBy
<<FunctionalRequirement>>
Functional Requirement
description All wheels must have individually controlled brakes
specifiedBy
<<FunctionalRequirement>>
Functional Requirement
descriptionBrakes must be properly scaled, compared to a regular vehicle
constrains
constrains
<<DesignConstraint>>
Disc Brake Design Constraint
{DesignConstraintType : Physical}
description : disc diameter according to vehicle length
<<DesignConstraint>>
CostConstraint
{DesignConstraintType : Cost}
description : use and maintenace cheap

Figure 8. Functional requirements on braking

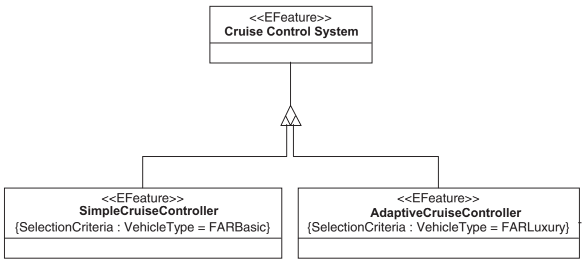

Figure 9. Variability illustrated by the Cruise Controller

5.2 Variability

A vehicle project in EAST ADL has a number of features. Which features that are included can be chosen based on the vehicle type. For example, two kinds of cruise controller can be used, simple CC or Autonomous CC, see Figure 9.

The CruiseController entity is a *variation point* that is replaced by one of its *variants* Simple CC and ACC. Which variant is chosen is decided by the *SelectonCriterion* expression. This is an OCL expression, but simplified here for readability.

The hardware and software entities that implement the electronic features will also vary, and the same variability concept is applied here, see Figure 10.

5.3 Legacy Tools and Code

The FAR system is developed using existing tools like Simulink/Stateflow and Targetlink. In order to link the EAST ADL model to the legacy tools, the ElementarySoftwareFunctions (the atomic building blocks of the FDA) of the model have references to these, see Figure 11. The interfaces defined in the model are met by the code and external model. The model structure is thus the master description tying together the various tools in use.

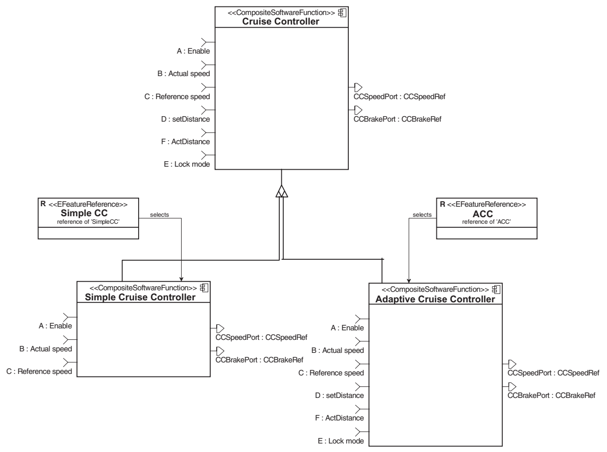

Figure 10. Variability in software: The cruise functionality in the Functional Design Architecture

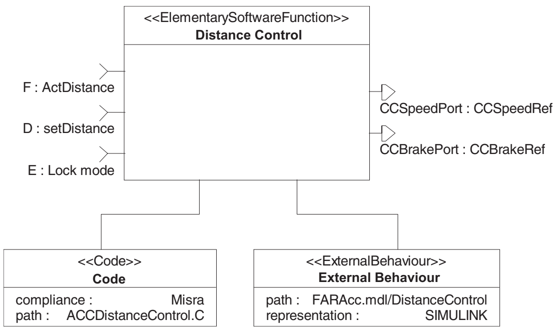

Figure 11. Link to legacy tools (FDA layer)

5.4 Device Interfacing

The link to sensors and actuators go through the LocalDeviceManagers which take care of the functional interfaces. Low level aspects like setting up A/D converters and accessing the right I/O pins are handled by HardwareAbstractionFunctions, HAFs. The HAF is in turn associated with a peripheral, a hardware entity representing the A/D converter or I/O device. Figure 12 shows the involved entities.

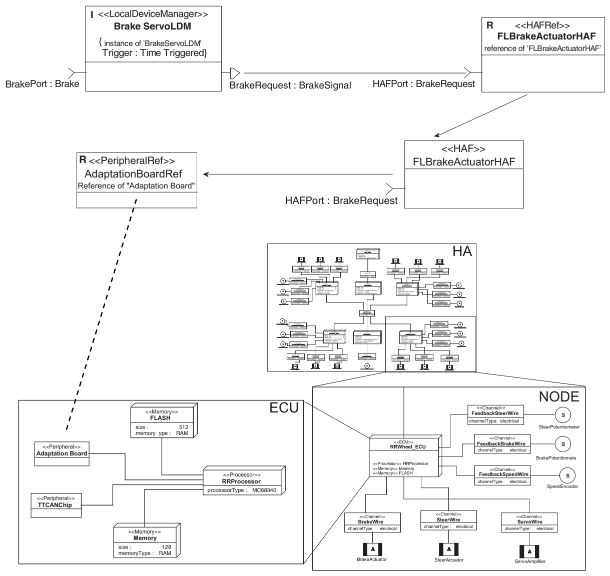

Figure 12. Device interfacing involving software entities from the FDA and PA as well as hardware entities.

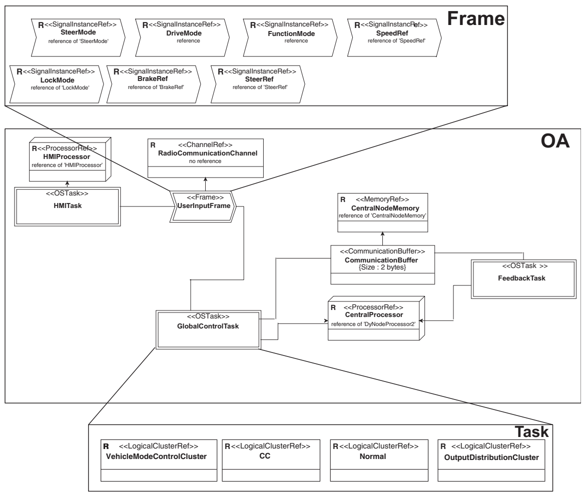
Frame
R <<SignalInstanceRef>> SteerMode reference of 'SteerMode'
R <<SignalInstanceRef>> DriveMode reference
R <<SignalInstanceRef>> FunctionMode reference
R <<SignalInstanceRef>> SpeedRef reference of 'SpeedRef'
R <<SignalInstanceRef>> LockMode reference of 'LockMode'
R <<SignalInstanceRef>> BrakeRef reference of 'BrakeRef'
R <<SignalInstanceRef>> SteerRef reference of 'SteerRef'
OA
R <<ProcessorRef>> HMIProcessor reference of 'HMIProcessor'
R <<ChannelRef>> RadioCommunicationChannel no reference
<<OSTask>> HMITask
<<Frame>> UserInputFrame
R <<MemoryRef>> CentralNodeMemory reference of 'CentralNodeMemory'
<<CommunicationBuffer>> CommunicationBuffer {Size : 2 bytes}
<<OSTask >> FeedbackTask
R <<ProcessorRef>> CentralProcessor reference of 'DyNodeProcessor2'
<<OSTask>> GlobalControlTask
Task
R <<LogicalClusterRef>> VehicleModeControlCluster
R <<LogicalClusterRef>> CC
R <<LogicalClusterRef>> Normal
R <<LogicalClusterRef>> OutputDistributionCluster

Figure 14. Allocation Model

5.5 Allocation

The software and message allocation is modeled in the *Allocation Model*. Figure 13 shows a part of the AM containing the brake setpoint value frame with four SignalInstances corresponding to brake values for the four wheels. The picture also show an OSTask with the brake control software in the form of FunctionInstances grouped to a LogicalCluster. The allocated entities all belong to the function instance model while the allocation targets are hardware architecture entities.

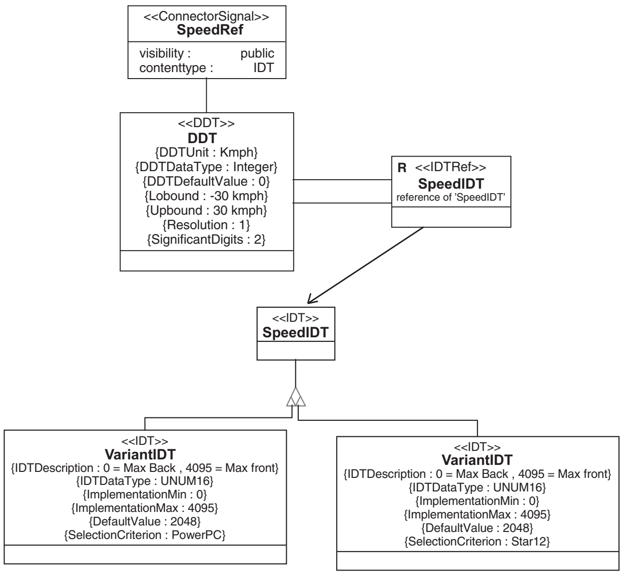

Figure 13. Design Data Type with Two Implementation Data Types

5.6 Data Types

Two kinds of data types are defined for signals and parameters in the model: The Design Data Type is the hardware independent abstract data type that is used to specify and then verify the design. For example, the resolution, range and unit can be investigated and verified by simulation and analysis. To match the chosen hardware, an implementation data type is defined that matches the DDT in resolution and range. To match more than one processor target, two implementation data types are defined here using the variability concept.

5.7 Timing

Timing aspects are defined both in terms of requirements and actual design. The FAR vehicle uses a time triggered bus, and the entire architecture is therefore periodic and time triggered. The EAST ADL model of this vehicle specifies timing in terms of a period and offset (relative to the period). The global time in this system means that this timing definition can be met in practice.

Timing requirements can be defined using, e.g. end-to-end deadlines, see Figure 15.

6. Conclusions

The need to enhance software integration in automotive software is increasingly evident. Integration concerns software from different suppliers, across domains and from previous projects The Enabling technologies for automotive software integration is currently emerging from several sources, for example industry initiatives like AUTOSAR [12] and EU 6th Framework programmes such as EASIS [13]. Gaining experience with modeling approaches that tie together all aspects of electronic systems development is increasingly important in this context. The application of EAST ADL to a concept vehicle as presented in this paper is an example of this.

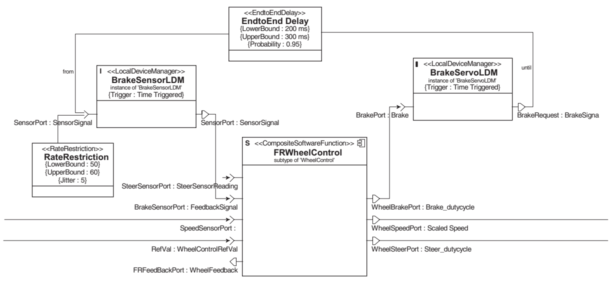
<<EndtoEndDelay>> EndtoEnd Delay
{LowerBound : 200 ms}
{UpperBound : 300 ms}
{Probability : 0.95}
from
until
I <<LocalDeviceManager>> BrakeSensorLDM instance of 'BrakeSensorLDM' {Trigger : Time Triggered}
I <<LocalDeviceManager>> BrakeServoLDM instance of 'BrakeServoLDM' {Trigger : Time Triggered}
SensorPort : SensorSignal
SensorPort : SensorSignal
BrakePort : Brake
BrakeRequest : BrakeSigna
<<RateRestriction>> RateRestriction
{LowerBound : 50}
{UpperBound : 60}
{Jitter : 5}
S <<CompositeSoftwareFunction>> FRWheelControl subtype of 'WheelControl'
SteerSensorPort : SteerSensorReading
BrakeSensorPort : FeedbackSignal
SpeedSensorPort :
RefVal : WheelControlRefVal
FRFeedBackPort : WheelFeedback
WheelBrakePort : Brake_dutycycle
WheelSpeedPort : Scaled Speed
WheelSteerPort : Steer_dutycycle

Figure 15. End to end timing requirement

This paper has outlined the modeling concepts of EAST ADL, as applied during the re-modeling of an existing by-wire model vehicle, FAR. The focus of the thesis project has been to model the concrete parts of the system, i.e. software, hardware and the vehicle configuration. The EAST ADL supports additional aspects such as requirements, process and verification and validation aspects. A continuation of this work would thus be to complete the model in these respects too. Such extension would illustrate the progress from requirements to implementation, and the use of requirements associations to relate entities of different abstraction layers.

7. References

[1] Embedded Architecture and Software Tools, the EAST-EEA project. *http://www.east-eea.net*

[2] Martin Törngren, Niklas Adamsson· Per Johannessen. Lessons Learned from Model Based Development of a Distributed Embedded Automotive Control System. To appear at the SAE World Congress, Detroit 2004. SAE Paper no: 2004-01-0713

[3] El-khoury Jad, DeJiu Chen and Martin Törngren. A Survey of Modeling Approaches for Embedded Computer Control Systems. Technical Report. TRITA - MMK 2003:36, ISSN 1400 –1179, ISRN KTH/MMK/R-03/11-SE. Department of Machine Design, KTH 2003

[4] OMG: SysML, the Systems Modeling Language. *http://www.sysml.org*

[5] AADL, Avionics Architecture Description Language. SAE standardization, Aerospace Avionic Systems Division, AS-2C. *http://www.sae.org/technicalcommittees/aasd.htm*

[6] Arcticus AB: The Rubus Operating System. http://www.arcticus.se

[7] Ulrich Freund, Orazio Gurrieri, Jochen Küster, Henrik Lonn, Jörn Migge, Mark-Oliver Reiser, Thomas Wierczoch and Matthias Weber: An Architecture Description Language for developing Automotive ECU-Software. *To Appear:* INCOSE 2004

[8] The Generic Modeling Environment, *http://www.isis.vanderbilt.edu/Projects/gme/*

[9] The Mathworks: Simulink. *http://www.mathworks.com/*

[10] Statemate. *http://www.ilogix.com*

[11] ETAS GmbH : ASCET-MD. *www.etasgroup.com*

[12] AUTOSAR: Automotive Open Systems Architecture. *http://www.autosar.org*

[13] EASIS: Electronic Architecture and System Engineering for Integrated Safety Systems. EU 6th framework project 507690

8. Acknowledgements:

The EAST ADL definition is a joint effort in the EAST project. We would especially like to acknowledge Ulrich Freund, Orazio Gurrieri, Jörn Migge, Mark-Oliver Reiser, and Thomas Wierczoch for their efforts. Thanks are also due to Vilgot Claesson for feedback on this work.