



Grant Agreement 224442

Advancing Traffic Efficiency and Safety through Software Technology phase 2 (ATESST2)

Report type	D3.1 Appendix A3.4
Report name	Update Suggestions for Behavioral Support
Dissemination level	PU
Status	Final
Version number	1.1
Date of preparation	2010-06-09

Authors**Editor**

DeJiu Chen

E-mail

chen@md.kth.se

Authors

DeJiu Chen

E-mail

chen@md.kth.se

Anders.Sandberg

anders.sandberg@mecel.se

Lei Feng

leifeng@md.kth.se

Carl-Johan Sjöstedt

carlj@md.kth.se

The Consortium

Volvo Technology Corporation (S)	VW/Carmeq (D)	Centro Ricerche Fiat (I)
Continental Automotive (D)	Delphi/Mecel (S)	
Mentor Graphics Hungary (H)	CEA LIST (F)	
Kungliga Tekniska Högskolan (S)	Technische Universität Berlin (D)	University of Hull (GB)

Revision chart and history log

Version	Date	Reason
1.0	2010-05-31	Public release
1.1	2010-06-09	Extended public release with a behavior constraint annex integrated with the latest language definition and samples from a case study.

Table of contents

Advancing Traffic Efficiency and Safety through Software Technology phase 2 (ATESST2).....	1
Authors	2
Revision chart and history log.....	3
Table of contents	4
Executive Summary.....	5
List of Figures	6
List of Tables	7
1 Introduction	8
2 A domain model update proposal for extended behavior modeling support	10
2.1 Motivation and Concept.....	10
2.2 Behavior Constraints.....	11
2.2.1 <i>Parameter Constraint</i>	13
2.2.2 <i>State Machine Constraint</i>	13
2.2.3 <i>Computation Constraint</i>	13
2.2.4 <i>Support for Requirement and Vehicle Feature Modeling</i>	14
2.3 Example Case.....	15
3 A domain model update proposal for physical interactions	18
3.1 FunctionPowerPort.....	18
3.2 FunctionPowerPort semantics.....	19
4 Conclusions	20

Executive Summary

This appendix summarizes the strategies and suggestions on updating the EAST-ADL2 support for behavior modeling.

List of Figures

Figure 1. An overview of the current EAST-ADL2 Behavior Modeling Package.	8
Figure 2. The proposed behavior constraint annex and EAST-ADL2 domain model.....	9
Figure 3. An overview the domain model update proposal for extended behavior modeling support.	10
Figure 4. Key behavior modeling constructs and their extensions.	11
Figure 5. Proposed behavior constraints and the relations to the existing EAST-ADL2 constructs.	12
Figure 6. A top-level view of the EAST-ADL2 model.....	15
Figure 7. Specifying requirements and the operation situations.	15
Figure 8. Specifying the behavior constraints of environmental situations.....	16
Figure 9. Specifying the function behaviors of a system function.	16
Figure 10. Specifying the state machine constraints for a function behavior.	17
Figure 11. The FunctionPowerPort.	18

List of Tables

Table 1: Semantical alignment of the FunctionPowerPort 19

1 Introduction

Current EAST-ADL2 provides good support for the structuring and lifecycle management of automotive E/E system. The behavior modeling support covers the declarations of system modes, computation/transfer functions, execution dynamics, and related error behaviors. Figure 1 provides an overview of current EAST-ADL2 behavior modeling support, which allows the assignments of function behaviors and their triggering dynamics to system functions. The definitions for behaviors is however textual or based on the referenced external models (*FunctionBehavior.path:String*). No additional behavioral attributes are directly given in EAST-ADL2. Previous work in ATESS2 has performed a detailed evaluation of the current EAST-ADL2 behavior modeling support and related concepts (documented in the deliverable ATESS2_Deliverable_D2.1_A3.4).

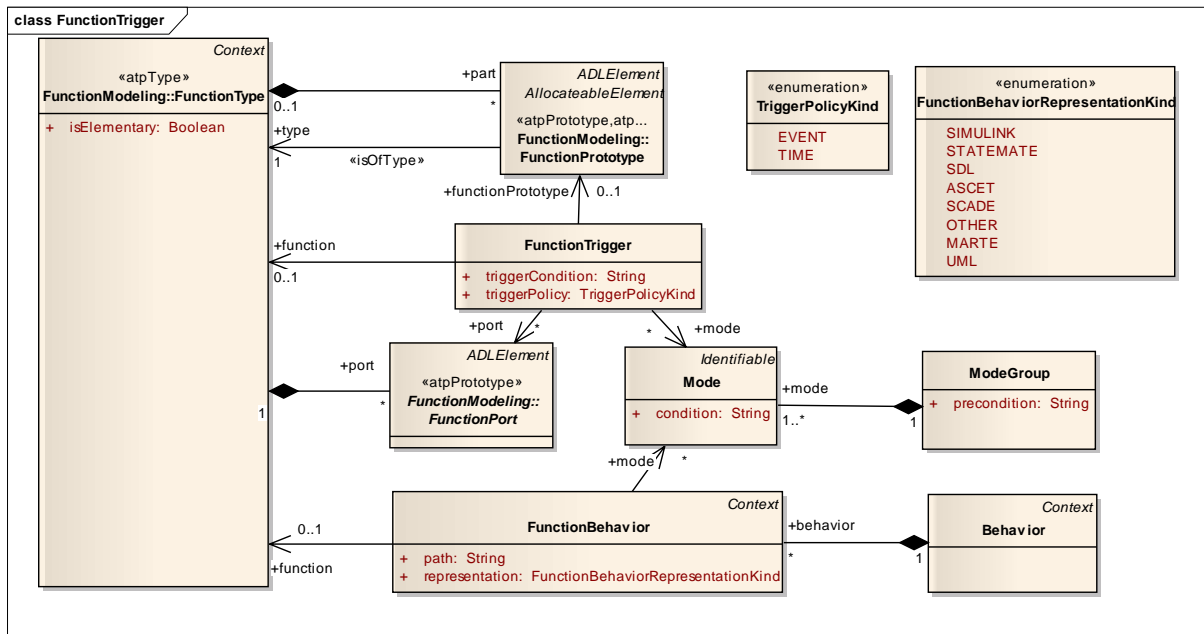


Figure 1. An overview of the current EAST-ADL2 Behavior Modeling Package.

As the actual definitions of behaviors are either given textually or relying on external models and formalisms (e.g., Simulink/Matlab), the language provides only a rather weak support for analyzing and managing related behavioral information in requirements, vehicle features (VFM), external behavior models, fault models, and V&V cases. The lack of more fine-grained behavioral attributes in EAST-ADL2 makes it difficult to specify the behavioral agreements/contracts of a system function to its environment such as in terms of pre-&post-conditions, operation states, etc.

A model exchange support from Simulink models to and from EAST-ADL2 models is highly desired from an industrial point of view. With current EAST-ADL2 language, the transformation can only be delimited to model-configuration (i.e., focusing on the transformations of function composition, connection topology, model execution dynamics). For example, it is impossible to exchange any further details about the function logics specified by the requirements and vehicle features in EAST-ADL2 and implemented by external tools (e.g., in regards to internal parameters, expected states, computation flow, etc.). Similarly, current EAST-ADL2 support for error logics is based on external fault modeling languages. There is no explicit language means for relating error definitions to nominal behaviors e.g., the transitions between nominal states and errors.

An earlier proposed strategy was to adopt the UML behavior modeling support for the specification of behavior in EAST-ADL2. However, this would require a tight coupling between

EAST-ADL2 and UML and is therefore considered to be too restrictive from an automotive point of view.

This deliverable describes a concrete behavior modeling update proposal that constitutes the basis for the EAST-ADL2 behavior constraint annex (See Figure 1). The proposal extends current EAST-ADL2 language with additional constructs for a more fine-grained specification of some key behavior attributes, targeting system requirements, vehicle features, system functions, and error models. In addition, a description of the introduced function power port for physical interactions is also included.

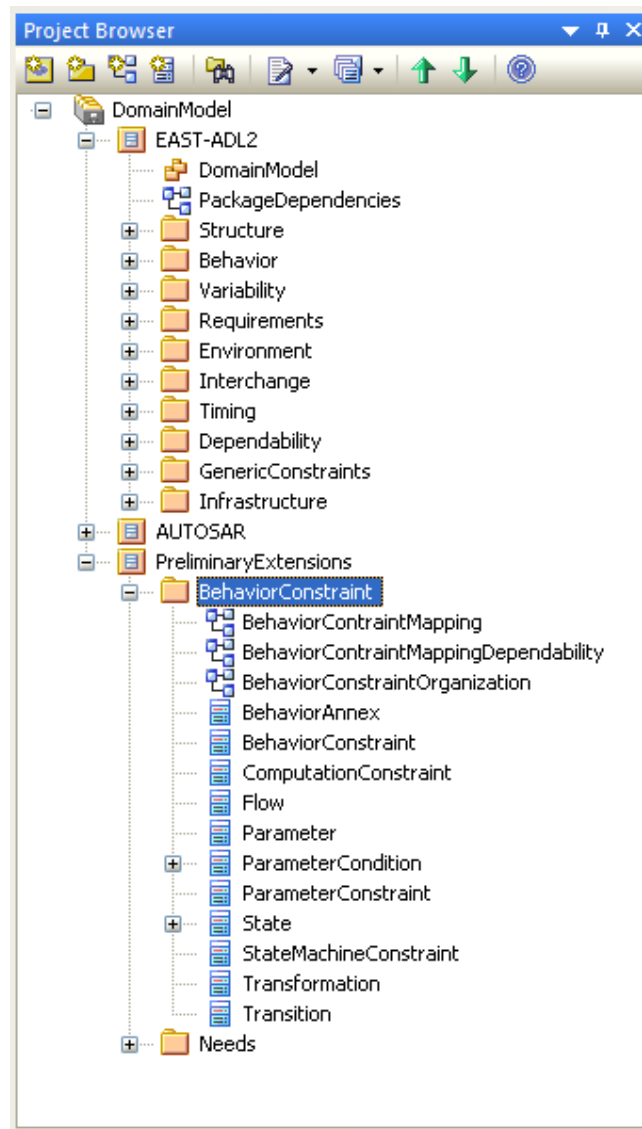


Figure 2. The proposed behavior constraint annex and EAST-ADL2 domain model.

2 A domain model update proposal for extended behavior modeling support

2.1 Motivation and Concept

One aim of this update proposal is to allow the declarations of key behavior properties with EAST-ADL2 and thereby to constrain behavior definitions in external models and formalisms (e.g., Simulink, UML, etc). The provision of such behavioral declarations in EAST-ADL2 is also considered necessary to support the formalization of textual requirements as well as for the reasoning of operational behaviors of vehicle features and environmental situations in early development stages. The proposed behavior modeling extension is given as a set of *behavior constraints*. See Figure 2 for an overview of these proposed constructs and its relations to other EAST-ADL2 definitions.

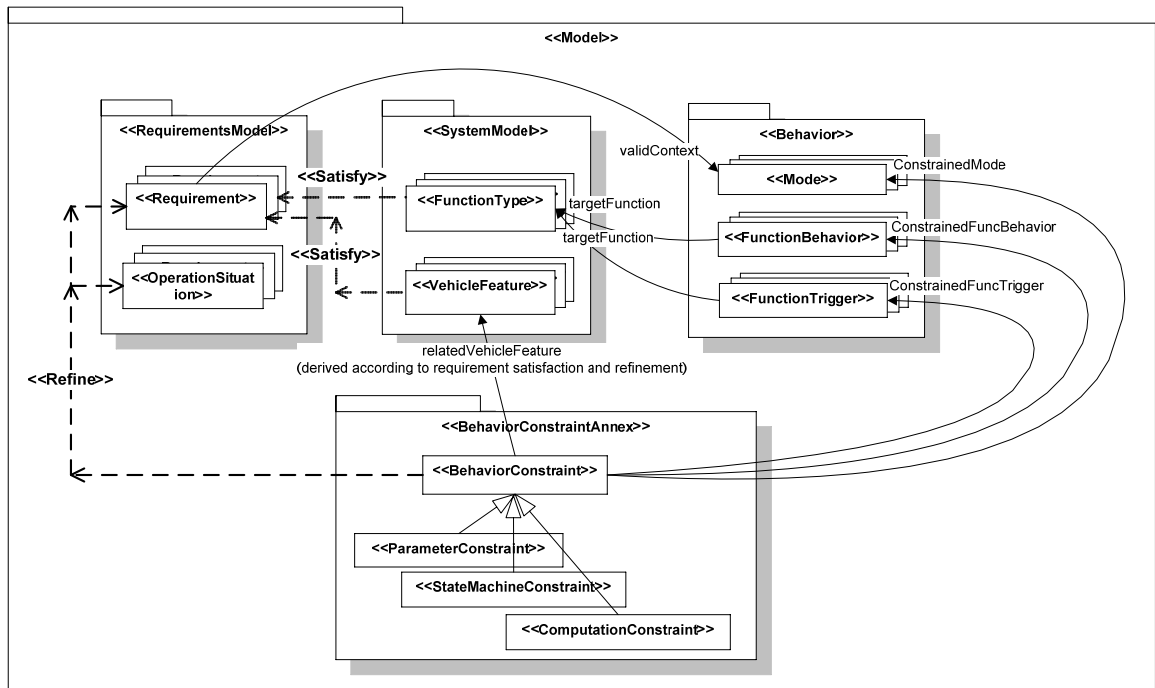


Figure 3. An overview the domain model update proposal for extended behavior modeling support.

The proposed behavior constraints are managed within the behavior constraint annex module. They are integrated into the EAST-ADL2 meta-model in a similar way as timing, dependability, and other non-functional constraints. This means that the behavior constraints target directly *Behavior::Mode*, *Behavior::FunctionTrigger*, *Behavior::FunctionBehavior*, and thereby any external behavior models linked by these EAST-ADL2 definitions. Through the references from function behaviors and function triggers to their targets, the system artefacts under constraint can be derived. Depending on the system artefacts under constraint, a behavior constraint can be applied either for *FAA* (Functional Analysis Architecture), or for *FDA* (Functional Design Architecture), or for environment specification.

For functional requirements and operation situations, the introduction of proposed behavior constraints allows the related textual descriptions in the requirement model to be refined through the existing requirement *refine* relationship in EAST-ADL2 as shown in Figure 2. The refinements in terms of behavior constraints provide declarations of behavior elements, such as the expected parameters, states and transitions, of one or multiple functional requirements. This facilitates the analysis and validation of requirements in regards to the semantics and

consistency of textual descriptions, as well as the reuse and management of requirements and their implied behaviors.

In EAST-ADL2, the definitions of system end-to-end functionalities are supported by the *VehicleFeature* constructs. Such features *satisfy* some functional requirements and are in turn *realized* by some system artefacts in *FAA* (Functional Analysis Architecture). The definitions of such end-to-end functionalities are currently textual. There are no links from such functionality definitions to the definitions of modes and function behaviors. With the proposed behavior constraints, the behavioral restrictions assigned to such end-to-end functionalities are derived through the *Satisfy* and *Refine* links. In other words, a vehicle feature satisfying a particular functional requirement normally also poses all behaviors refining that functional requirement.

Current EAST-ADL2 supports the definitions of error behaviors and anomaly through external logics. The proposed behavior constraints can be used to provide explicit language support for defining the semantics of anomaly (i.e., faulty conditions), and a fine-grained integration error behaviors to nominal state machine and computation behaviors.

2.2 Behavior Constraints

An overview of the proposed behavior constraints is given in Figure 4 and Figure 5. These constraints are given in three categories: 1. *Parameter Constraints*; 2. *State Machine Constraints*; and 3. *Computation Constraints*.

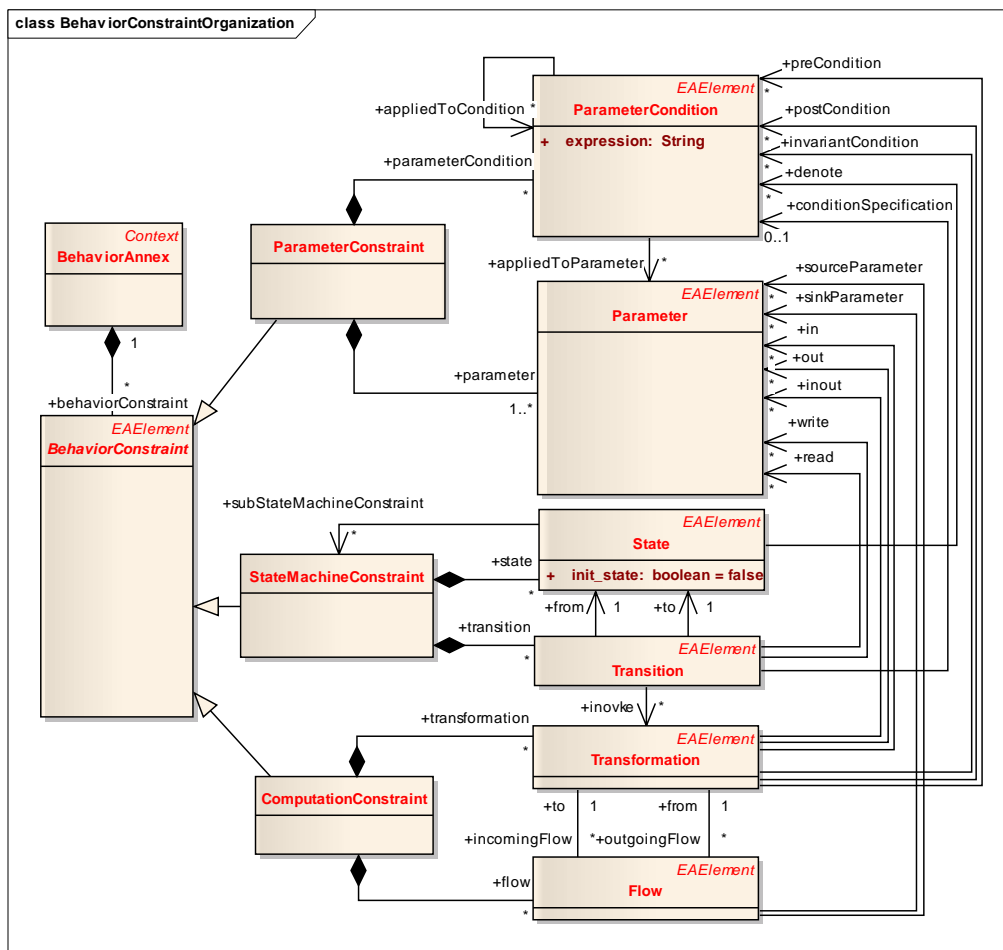


Figure 4. Key behavior modeling constructs and their extensions.

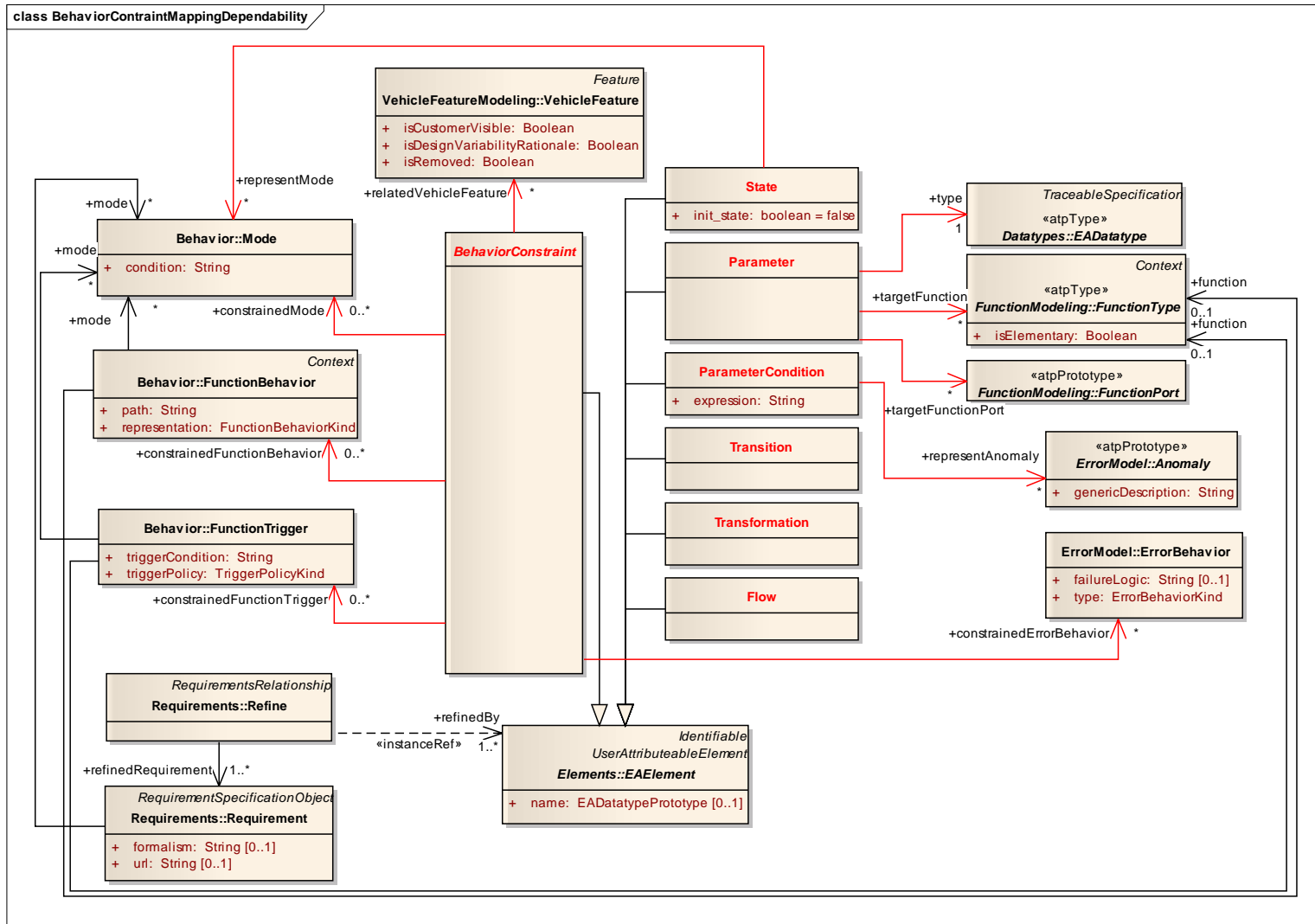


Figure 5. Proposed behavior constraints and the relations to the existing EAST-ADL2 constructs.

2.2.1 Parameter Constraint

The parameter constraints denote the quantities owned or processed by a mode, a function behavior, or a function trigger, as well as the conditions under which each of these quantities is used. They are used for the definitions of operation conditions, states, as well as the desired mapping of in- and output variables. For support parameter constraints, the following language constructs are introduced:

- **BehaviorConstraint::ParameterConstraint** – a placeholder for statements of the parameters of a behavior and their conditions.
- **BehaviorConstraint::Parameter** – statements of in-, out-, or local variables to be processed or the attributes owned by a behavior. A parameter can indicate an environmental variable such as the monitored or controlled variables of an E/E system, or a piece of application or event data being processed within the E/E system. It can also be used to denote event data carries information about the occurrence of particular events. Each parameter is typed by a *DataType* for specifying the related meta-information like unit, valid range, required accuracy, etc. While input/output parameters target the I/O ports of system functions, internal parameters target directly system functions..
- **BehaviorConstraint::ParameterCondition** – statements of the conditions of individual parameters in nominal and erroneous operating situations. For example, a parameter condition may state that a monitored environmental variable must fall within a particular segment in its total value range for a specific vehicle control. For a computation behavior, parameter conditions can be used to specify its pre-/post-conditions and invariants that must be true before, after, and during the execution. Parameter conditions can also be used as a basis for defining the states of parameters during computation as well as the relations of two sets of parameters (e.g., input and output mapping, event to output mapping).

2.2.2 State Machine Constraint

The state machine constraints denote the states and state transitions of modes, function behaviors, or function triggers. To this end, the following language constructs are introduced:

- **BehaviorConstraint::StateMachineConstraint** – a placeholder for statements of states and transitions.
- **BehaviorConstraint::State** – statements of the states representing the value conditions of one or multiple parameters of a behavior. Within each state, there can be subordinate state machines.
- **BehaviorConstraint::Transition** – statements of state transitions in response to certain event data given as the *read* parameters and when certain guard conditions are met. During a state transition, a transformation can be invoked.

2.2.3 Computation Constraint

The computation constraints define the logical transformations of data in a function behavior. The declarations are supported by the following language constructs:

- **BehaviorConstraint::ComputationBehavior** – a placeholder for statements of the computational transformations and flows.
- **BehaviorConstraint::Transformation** – statements of the data processing activities mapping two sets of parameters. A computation transformation has input and output parameters and is further characterized by some *pre-*, *post-*, and *invariant* conditions.

- **BehaviorConstraint::Flow** – statements of the control-flow of transformations and data flows of transformation parameters.

2.2.4 Support for Requirement and Vehicle Feature Modeling

A requirement expresses a condition or capability that must be met or possessed by a system or system component. Requirements can be introduced in different phases of the development process for different reasons. With EAST-ADL2, a requirement can have *Satisfy* associations to the artifacts proving the implementation, *DeriveReq* associations to the derived requirements or constraints, and *Verify* associations to the specifications of related verification and validation efforts.

The proposed behavior constraints provide explicit language support for a more precise definition of the semantics of behaviors as well as the interdependences, e.g., allowing the differentiation of functional requirements that share the same input parameters but produces different outputs. Based on the *Refine* links from textual requirements to behavior constraints or elements (e.g., parameters, states, and flows), the following behavioral information can be defined with EAST-ADL2:

- *Implied operation conditions* – statement of the environmental conditions e.g., in terms of environmental parameters and parameter conditions in which a requirement is valid.
- *Implied operation behaviors* – statement of the required operational states and state transitions, transformations and flows of a requirement.
- *Implied operation parameters* – statement of the parameters implied by a requirement (e.g., the monitored and controlled environmental variables, expected inputs and outputs) and the related parameter conditions.

In EAST-ADL2, a technical feature model captures the agreed system end-to-end functionalities in terms of *vehicle features* and their dependencies such as shared requirements, operational modes, and environment assumptions. Other information referenced by the vehicle level's technical features includes the satisfied vehicle level requirements, derived hazards and safety goals. With the proposed behavior constraints, the behavior constraints related to a vehicle feature, such as the operational modes of features, and the environmental conditions where the feature is valid, can be derived according to the requirement *satisfy* and *refine* links.

2.3 Example Case

To investigate the engineering needs and evaluate related language support, an industrial case study based on a vehicle longitudinal control has been performed. Here, we highlight some key features of the behavior constraint support. The modules of the EAST-ADL2 model for the case system is shown in Figure 6. The behavior constraint module manages the proposed behavior constraints that provide additional behavior information for the requirements and system behaviors.

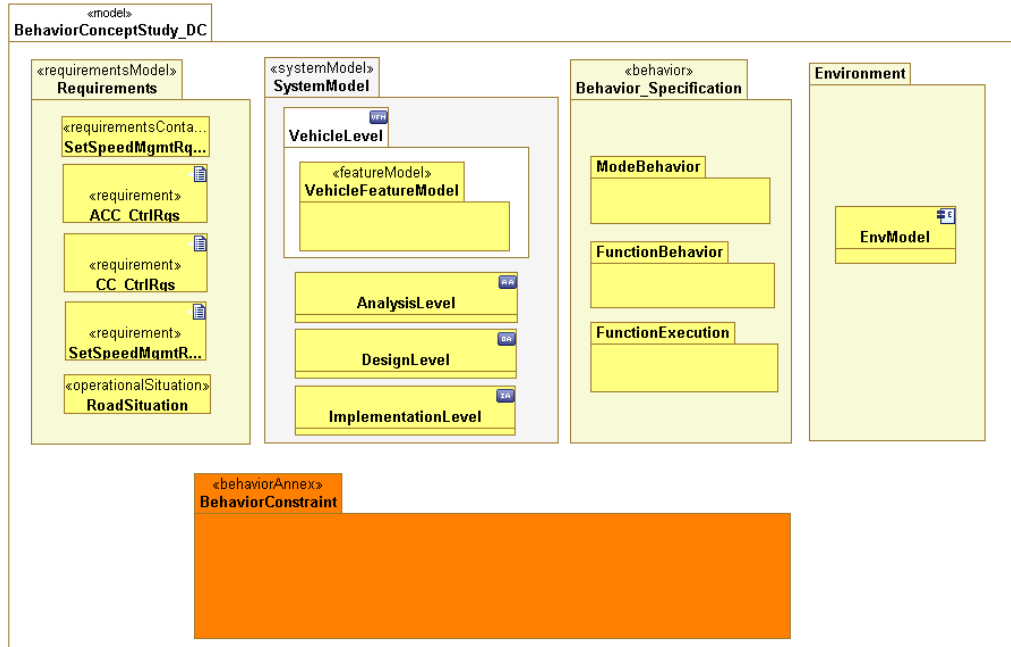


Figure 6. A top-level view of the EAST-ADL2 model.

Within the requirement model, the functional requirements and their implied operation context and environmental situations are first specified. This information is however captured textually without direct EAST-ADL2 support. See Figure 7.

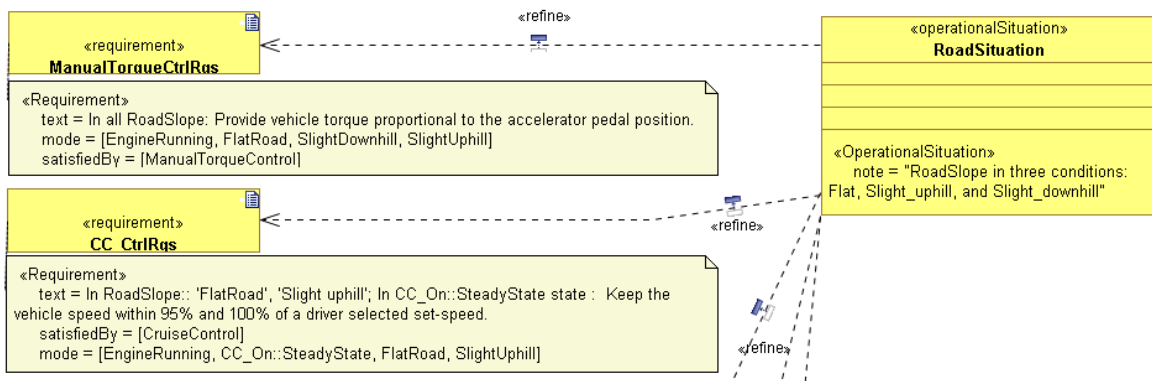


Figure 7. Specifying requirements and the operation situations.

The refinement of the textual requirements and operation situations provides more precise information about the parameters, parameter conditions, and state machine and computation assumptions. This is illustrated in Figure 8 for the road situation. Refinement of other functional behavior can be performed with the same pattern.

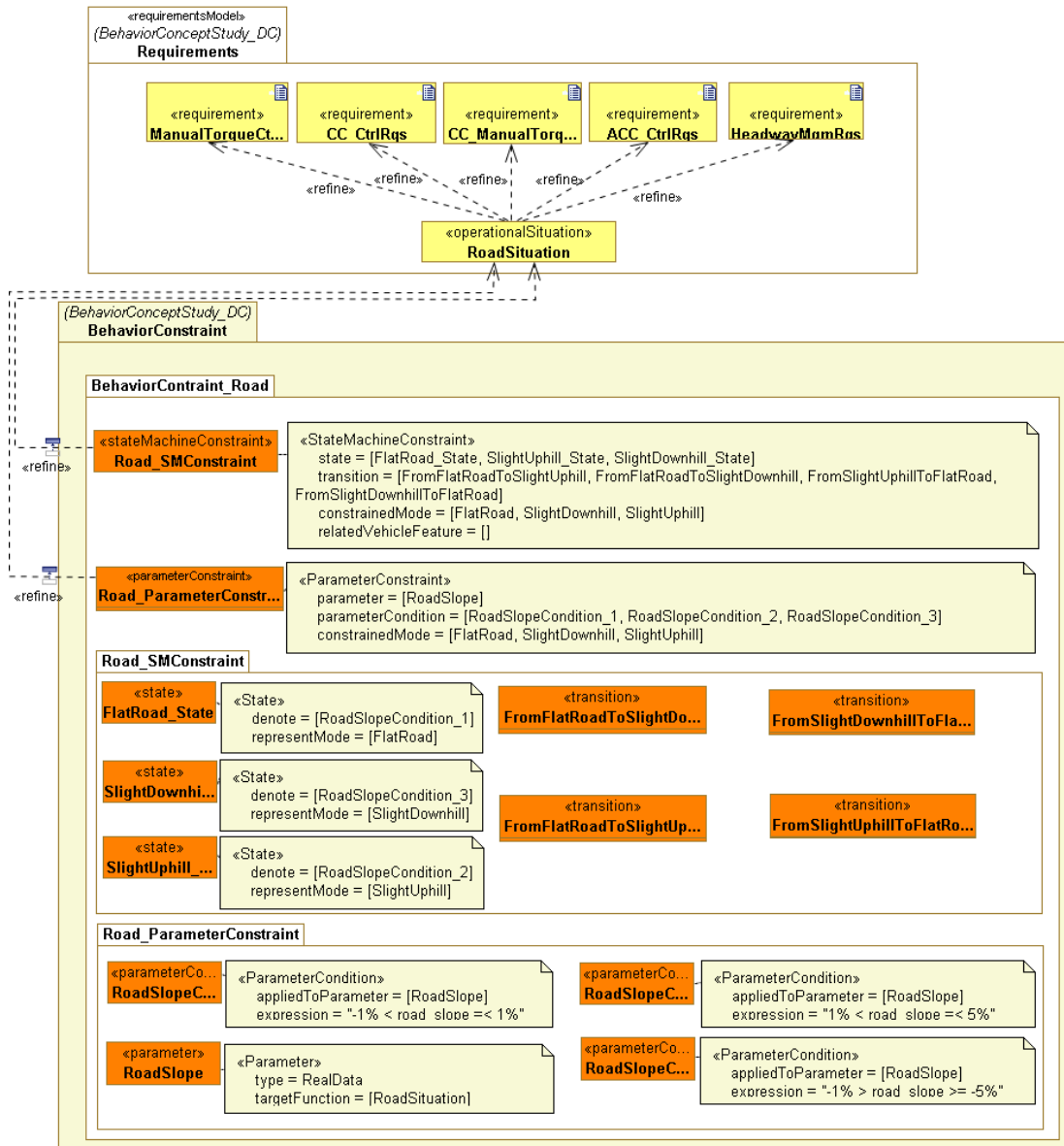


Figure 8. Specifying the behavior constraints of environmental situations.

Figure 10 shows the assignment of state machine constraints to the function behavior *CruiseControlBehavior*, which contains the behavior model of the *CruiseControl* function. This state machine behavior constraint are related to the vehicle features *CruiseControl*, *AdaptiveCruiseControl*, and *BasicCruiseControl* as all these end-to-end functionalities satisfy the functional requirements that are refined by these state machine constraints.

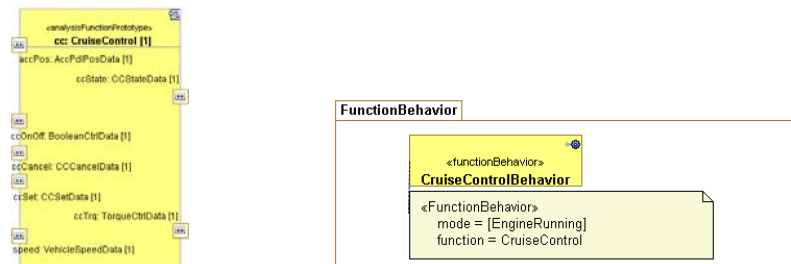


Figure 9. Specifying the function behaviors of a system function.

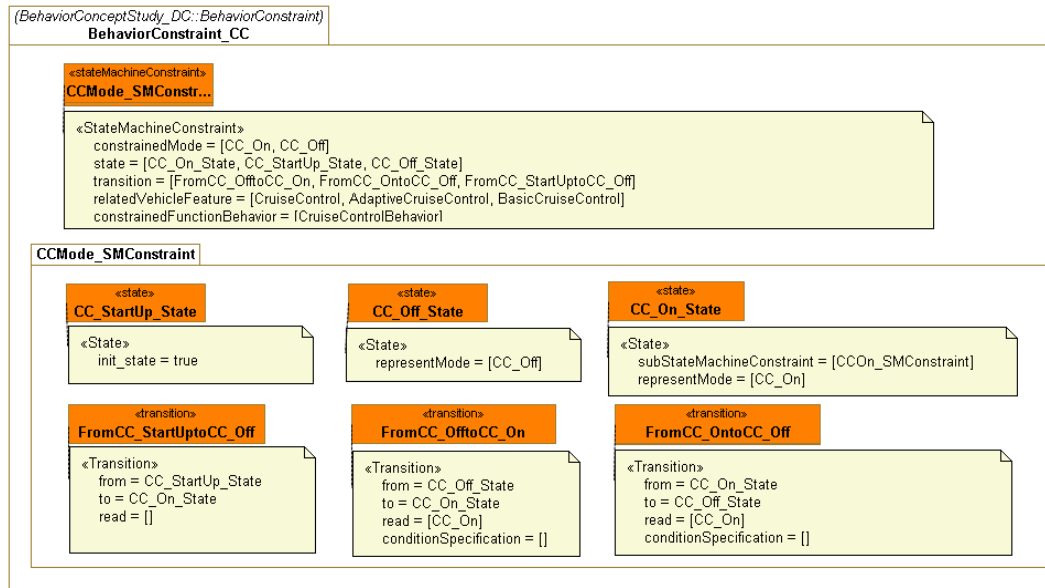


Figure 10. Specifying the state machine constraints for a function behavior.

3 A domain model update proposal for physical interactions

3.1 FunctionPowerPort

The rationale behind the *FunctionPowerPort* is to be found in D2.1 A3.4. The use of the ordinary *FunctionFlowPort*, with additional constraints was not considered to be a satisfying solution, due to the characteristics of a power port, where the direction is not specified. Hence, a new element, the *FunctionPowerPort*, was created, which inherits from *FunctionPort*. This link is needed to allow connectors. The statement “FunctionPorts are single buffer overwrite and nonconsumable” was removed from the *FunctionPort*, and it is now in the *FunctionFlowPort* and *FunctionClientServerPort* descriptions.

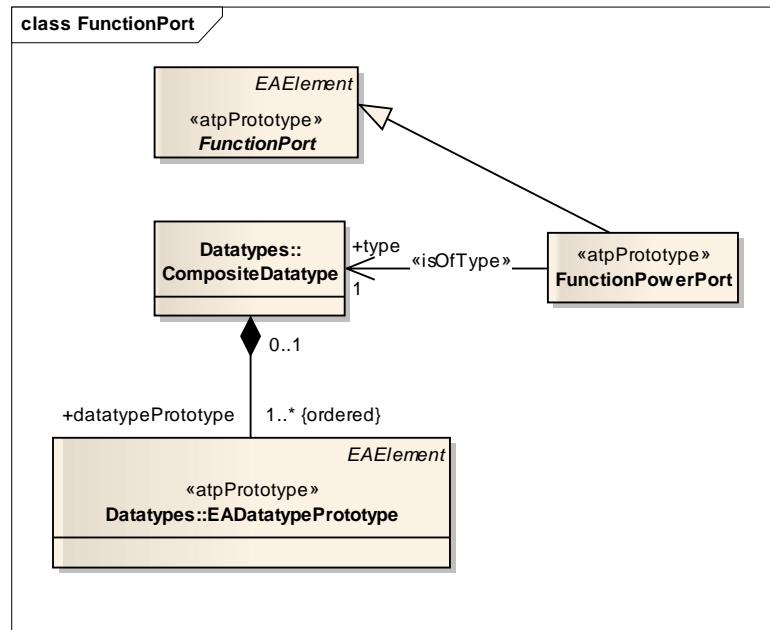


Figure 11. The FunctionPowerPort.

A *FunctionPowerPort* has exactly two conjugate variables, named *across* and *through*, which is the dominant way to name them in literature. This is not directly visible in the metamodel, since a *FunctionPort* only can have one datatype. Instead, a *CompositeDatatype* containing the *Across* and *Through* variables must be used.

The following constraints apply on a *FunctionPowerPort*

1. The owner of a *FunctionPowerPort* is either a *FunctionalDevice*, a *HardwareFunctionType*, or a *FunctionType* for environment
2. Two connected *FunctionPowerPort* must have the same *Datatype*.
3. The typing *Datatype* shall have two *datatypePrototypes* called *Across* and *Through*, with *Datatypes* that are consistent and representing the variables of the *PowerPort*.

3.2 FunctionPowerPort semantics

The typing *Datatype* owns two *datatypePrototypes* called *Across* and *Through*, representing the exchanged physical variables of the *FunctionPowerPort*. In two or more directly connected function power ports, the *Across* variables always get the same value and the *Through* variables always sum up to zero. This should be equivalent to the MATLAB/Simscape definition (from the MATLAB help):

- Two directly connected Conserving ports must have the same values for all their *Across* variables (such as pressure or angular velocity).
- You can branch Physical connection lines. When you do so, components directly connected with one another continue to share the same *Across* variables. Any *Through* variable (such as flow rate or torque) transferred along the Physical connection line is divided among the multiple components connected by the branches. How the *Through* variable is divided is determined by the system dynamics. For each *Through* variable, the sum of all its values flowing into a branch point equals the sum of all its values flowing out.

This should also be equivalent to a Modelica connector:

When drawing connection lines between ports, the meaning is that corresponding connector variables without the "flow" prefix are identical (here: "v") and that corresponding connector variables with the "flow" prefix (here: "i") are defined by a zero-sum equation (the sum of all corresponding "flow" variables is zero). The motivation is to automatically fulfill the relevant balance equations at the infinitesimal small connection point.

VHDL-AMS has a similar concept, called Terminal Port. In Table 1, an overview of the equivalence between the concepts. Note the name of this port type varies between all the languages, which is unfortunate. The PowerPort is also used in literature for these types of ports.

Table 1: Semantical alignment of the FunctionPowerPort

EAST-ADL2	Modelica	MATLAB/Simscape	VHDL-AMS
FunctionPowerPort	Connector	Physical Conserving port	Terminal Port
Across	(Potential)	Across	Across
Through	Flow	Through	Through

4 Conclusions

In this document, we have presented a concrete proposal for the behavior modeling update for a more precise behavior modeling support in regards to requirement specification and system behavior specification. Providing the behavior constraints in EAST-ADL2 is considered important for managing requirements, features, and external behavior models. The suggested solutions can be implemented in different ways with a UML profile. We have illustrated the possibility of supporting the declaration of behavior constraints through UML::Class. It is however also possible to treat these behavior constraints as the extensions UML behavior elements (e.g., UML::State and UML::Transition). This will allow the reuse of UML representation of such behavior constraints.

It is also possible to transform the behavior constraints to external tools. An example implementation of the Simulink exchange plugin has been developed as a proof-of-concept for such an external behavior definition. The inclusion of the *FunctionPowerPort* allows interchange of physical modeling models, which enables behavior definition in such languages, most notably MATLAB/Simscape and Modelica.