

ATESST

Your digital version of the ATESST2 newsletter #5 2010.



[Click here to download the UML Modelling tool Papyrus and the EAST-ADL profile](#)

Dear Colleague,

in this ATESST2 newsletter, we will present how cooperative systems are handled in EAST-ADL and how the environment is modelled. Some guidelines for the structure of system modelling are presented to show how cooperative systems are supported.

Further information about the ATESST2 project and the EAST-ADL language can be found at <http://www.atesst.org/>.

IMPORTANT: An intermediate version of the EAST-ADL domain model specification is [here](#).

In This Issue

[Final Open Workshop of the ATESST2 Project, 21 June 2010, Frankfurt](#)
[Active Safety System and Cooperative Active Safety System Definition](#)
[Cooperative System Modelling](#)
[Overview of Modelling Environment Models in Simulink Combined with EAST-ADL](#)
[Introducing FunctionPowerPorts](#)
[The Simulink to EAST-ADL Plug-in](#)
[The ClampConnector](#)

Final Open Workshop of the ATESST2 Project, 21 June 2010, Frankfurt

Agenda

08:30 Welcome coffee

09:00 Seminar session

12:00 Lunch

13:00 Break-out sessions: Small presentations and demonstrations on selected topics:

17:00 End of Day

Venue

Mercure, FRANKFURT ESCHBORN OST
Helfmann-Park 6
D - 65760 Eschborn
Tel +49(0)6196-901-0
Fax +49(0)6196-901-900
h0491@accor.com

www.mercure-frankfurteschborn-ost.com

Topics in the seminar session include EAST-ADL Overview, relation to AUTOSAR, methodology, tooling, modelling concepts for variability, safety, requirements and V&V, cooperative systems and environment.



EAST-ADL Spotlight

ATESST2 defines EAST-ADL as a domain-specific language using meta-modelling constructs such as classes, attributes, and relationships.

The project also implements a UML2 profile which is used in UML2 tools for user modelling.

The EAST-ADL definition also serves as the specification for implementation in domain-specific tools.

EAST-ADL2

Active Safety System and Cooperative Active Safety System Definition

In general, cooperative systems form a broad class of systems where the function is allocated over several cooperating entities. In the ATESSST2 project the focus area has been Cooperative Active Safety Systems, a term that needs a definition:

Active Safety Systems are automotive EE systems with the aim to avoid accidents. In case of an unavoidable accident it seeks to reduce the severity. Sensing and actuation resides on the same vehicle and is not distributed between vehicles.

Cooperative Active Safety System is an Active Safety System where sensing and actuation is distributed over several vehicles and communicated between vehicles. Hence, it represents a system of systems. This definition implies that we are dealing with functions that have real-time control aspects as well as them being safety critical in their implementation.

Previous newsletters have discussed the abstraction levels and their content, see www.atesst.org → [Newsletter](#).

[Click here to download a presentation with an overview of EAST-ADL cooperative system modelling.](#)

Cooperative System Modelling

The modelling of cooperative active safety systems inherits most of its content from the modelling of active safety systems, where only in-vehicle communication is considered. The differences come when the environment and vehicle-to-vehicle connections are modelled.

Modelling cooperative systems requires no dedicated language concepts and this has been the design principle when the language support has been developed. There are minor adaptations, mainly on multiplicity and 'legal' use of connectors but in general the same concepts are used for conventional and cooperative systems. The support is based on some basic language use principles (guidelines) that apply:

- Each vehicle is modelled within one SystemModel element in the EAST-ADL model.
- The environment is modelled using the following pattern:
 - For each vehicle there is a Local environment that contains the vehicle control, dynamics and other properties that are vehicle specific and may be unique when several vehicles are simulated.
 - All vehicles move in a Global environment. This model contains properties relating to vehicle position and other data on how a vehicle appears with respect to other vehicles. Further decomposition into a "Near" and "Far" environment is also possible.

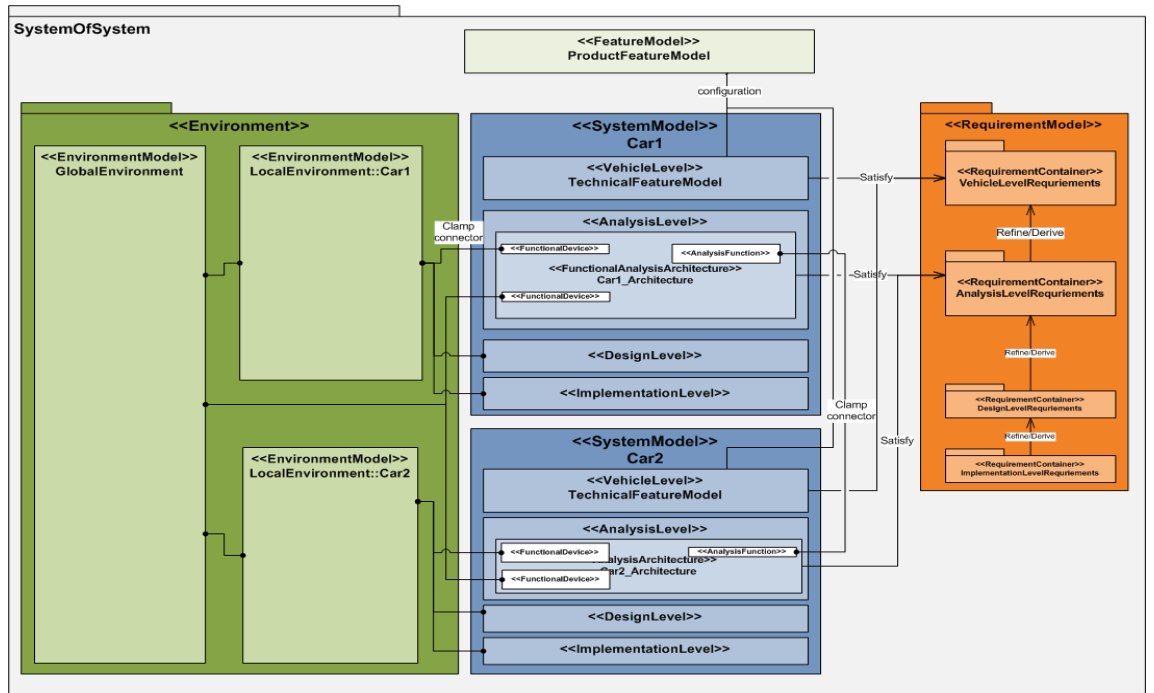


Figure 1: EAST-ADL model structure for a cooperative function.

Each system model entity can be an instance of the same 'type' of vehicle, but with individual configurations and properties. Each vehicle has its own SystemModel entity that contains the containers for the different abstraction levels. The real time saving work comes from re-using the FunctionalAnalysisArchitecture class, as a 'prototype' in the new vehicle. This modelling pattern with 'type' elements that define structure and 'prototypes' used to instantiate types that are individual in nature is commonly used in the language.

The same 'type-prototype' concept applies to the LocalEnvironment model that for the same reason needs to be independent even if they share the same meaning and thus type definition.

The model structure also shows the distinction between how Clamp connectors and normal Flow ports are used in the model. A clamp connector is used when the signal transferred is a mapping between the physical world, Environment, and a sensor in the logical world. A clamp is also allowed when connecting elements on the same abstraction level, i.e., AnalysisLevel, on different System models. Flow connectors are used when connecting elements inside one System model or Environment model.

The figure tries to show that requirements on both vehicle and analysis level can be associated with modelling elements on several vehicles, or system models. This is necessary for many requirements where timing and distributed behaviour is concerned. In many cases a requirement for a cooperative system is only applicable to the system that resides in a sole vehicle though.

equation-based modeling, or “Modelica-like” modelling. The main idea is to model physical connectors and ports like they their physical equivalence, e.g. an electrical port is modelled as an electrical pin, and two connected mechanical objects behaves as if their flanges would be mated.

By nature, such a physical connection can transfer signals in two directions, e.g. an engine connected to a drive-line can both produce torque and receive torque, and so by introducing a directed torque signal using a flowport could be error-prone. Instead of letting the modeller set the direction, it is taken care of by the tool at the time of simulation, so the flow of torque can be in both directions.

This kind of port has different names in different languages and tools: in Simscape they are called “Physical Conserving port”, in VHDL-AMS “Terminal Port”, and in Modelica a “Connector”. In EAST-ADL, we chose to use the name FunctionPowerPort, since it in some literature it is called power port, referring to the two conjugate variables that are represented in the port, e.g. torque and velocity of which the product has the unit power.

Some of the main advantages of this technique include:

- Reusable components. Using causal techniques, the same component must be modelled in different ways depending on how it fits into the system.
 - Cleaner modelling, fewer connections between subsystems are needed
 - More engineering work and modelling is dealt with by tools, rather than the engineer, which (at least in theory) is less error-prone, and more efficient.
-

The Simulink to EAST-ADL Plug-in

A plug-in for exchange with MATLAB/Simulink was developed already in the former ATESSST project, however, in ATESSST2, it has been completely rewritten. One new feature is the use of the Simulink library mechanism for EAST-ADL functions, and that there is now a MATLAB plug-in as well as a Papyrus plug-in. The idea is that the modeller partitions his model in Simulink subsystems, which will correspond to FunctionPrototypes. By placing the block in the EAST-ADL Simulink library, it will be mapped to a FunctionType. When a block from the library is used in the model, the user cannot change the structure of the block. This conforms well to the Type-Prototype mechanism in EAST-ADL, (it is actually called Prototype – Type in Simulink, but with opposite meaning!). There are many reasons why this solution was chosen, the most important is that it allows bi-directional transformation and synchronization, without having to know all the underlying details of the blocks in Simulink.

This could be a typical workflow for working with Simulink models and EAST-ADL:

- Develop the control algorithm in Simulink
- Partition the model in MATLAB/Simulink into subsystems, and register the subsystems in the Simulink EAST-ADL library.
- Save the model file as a .Simulink file (added option in the Simulink menu)
- Convert it to EAST-ADL using the transformation plug-in, where the transformations are implemented using ATL, Atlas Transformation Language
- In EAST-ADL, the model is linked to other modelling artefacts, e.g. features, requirements, use-cases. Perhaps the component needs to have an additional port.
- By transforming it back, the Simulink model will have the extra port defined in EAST-ADL, and the models will stay synchronized.

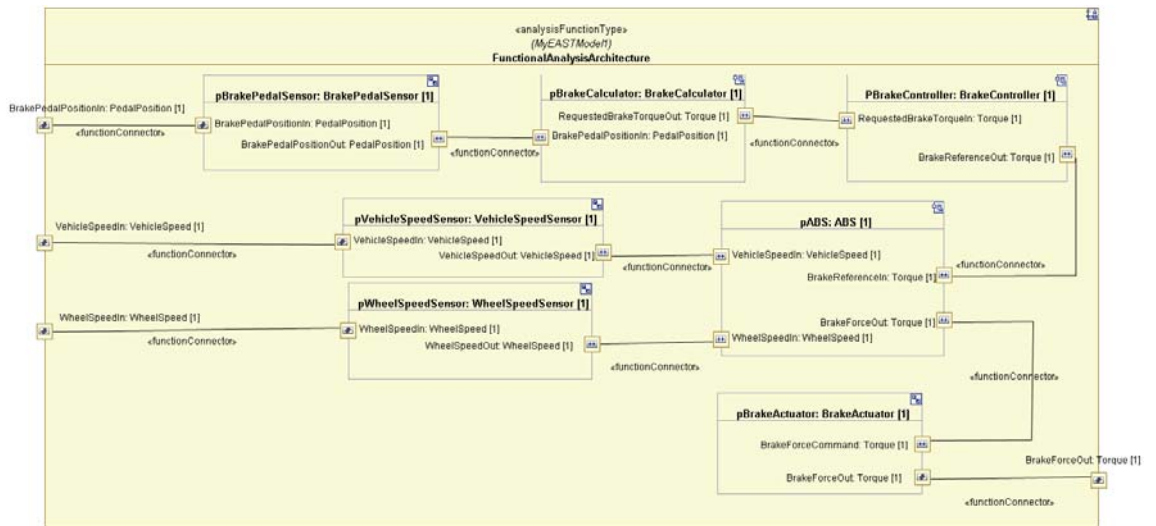


Figure 3: The FAA model corresponding to the Simulink model above.

Note that the environment model representing, e.g., the vehicle dynamics is modelled separately (not shown here)

The ClampConnector

To simplify the connection between a FunctionType and the environment, the clamp connector is introduced in the EAST-ADL language. Before this connector was available, a connector from the output of a FunctionalDevice, which could be buried deep down in the modelling hierarchy, had to be connected using delegation connectors between all the blocks in the hierarchy.

The clamp connector connects ports across function boundaries and containment hierarchies. It is used to connect from an EnvironmentModel to the FunctionalAnalysisArchitecture, the FunctionalDesignArchitecture, the autosarSystem or another EnvironmentModel. Typically, the EnvironmentModel contains physical ports, which restrict the valid ports in the FunctionalAnalysisArchitecture to those on FunctionalDevices and in the FunctionalDesignArchitecture to those on HardwareFunctions.



If you choose to receive/not to receive future ATESS2 newsletters, please inform owner-sig-adl@vtec.volvo.se.

The ATESS2 consortium

