MAENAD

Grant Agreement 260057

Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles

Report type — Deliverable D6.1.3

Report name — Case study analysis and safety assessment

Dissemination level	**PU**
Status	**Final (third iteration)**
Version number	**3.1**
Date of preparation	**2014-02-21**

Authors

Editor	E-mail
Stefano Cerchio	stefano.cerchio@crf.it

Authors	E-mail
Carlo LaTorre	carlo.latorre@4sgroup.it
DeJiu Chen	chen@md.kth.se
Frank Hagl	Frank.Hagl@continental-corp
Henrik Lönn	henrik.lonn@volvo.com
Luis P Azevedo	L.P.Azevedo@2012.hull.ac.uk
Mark – Oliver Reiser	Mark-Oliver.Reiser@tu-berlin.de
Martin Walker	martin.walker@hull.ac.uk
Renato Librino	renato.librino@4sgroup.it
Sandra Torchiaro	sandra.torchiaro@crf.it
Sara Tucci-Piergiovanni	sara.tucci@cea.fr
Tahir Naseer Qureshi	tnqu@md.kth.se

Reviewers	E-mail
Henrik Lönn	henrik.lonn@volvo.com
Renato Librino	renato.librino@4sgroup.it

The Consortium

Volvo Technology Corporation (S) Centro Ricerche Fiat (I)

Continental Automotive (D) Delphi/Mecel (S) 4S Group (I)

ArcCore AB (S) MetaCase (Fi) Systemite (SE) CEA LIST (F)

Kungliga Tekniska Högskolan (S) Technische Universität Berlin (D) University of Hull (GB)

Revision chart and history log

Version	Date	Reason
0.1	2011-03-08	Document Outline
0.5	2011-05-02	Mid-Term
1.0	2011-08-30	First year Version
2.0	2012-08-31	Intermediate (second iteration)
2.1	2013-06-30	3rd release
2.2	2013-07-10	Merge all contrib. from partners and send for review.
3.0	2013-08	Final version based on review comments from Volvo and 4SG
3.1 prel	2014-02-18	Final version reflecting M42 status for review
3.1	2014-02-21	Final version

Approval	Date
Henrik Lönn	20140221

List of abbreviations

AADL	**Architecture and Analysis Design Language**
ASIL	**Automotive Safety Integrity Level**
ETA	**Event Tree Analysis**
FEV	**Fully Electric Vehicle(s)**
FMEA	**Failure Modes and Effects Analysis**
FMU	**Functional Mockup Interface**
FTA	**Fault Tree Analysis**
HAZOP	**Hazard & Operability Analysis**
HiP-HOPS	**Hierarchically-Performed Hazard Origin & Propagation Studies**
HRC	**Heteorogenous Rich Components**
MAW	**MAENAD Analysis Workbench**
MMW	**MAENAD Modelling Workbench**
SRA	**Safety Requirement Allocation**
SRD	**Safety Requirement Derivation**
UML	**Unified Modeling Language**
XSLT	**Extensible Style sheet Language Transformations**

Table of contents

Authors 2
Revision chart and history log 3
List of abbreviations....4
Table of contents 5
1 Executive Summary....7
2 Introduction....8
2.1 Purpose and target group....8
2.2 Relation to other project activities....8
2.3 Analysis Approach....8
3 Verification....11
3.1 Verification Activities results....11
4 Trials....12
4.1 Evaluation of ability to support ISO 26262....13
4.1.1 Trial 1:HiP-HOPS Gateway....13
4.2 Evaluation of performance analyses....16
4.2.1 Trial 2: Timing Analysis plug-in....16
4.2.2 Trial 3: Simulink Gateway....17
4.2.3 Trial 4: MODELISAR FMU import....18
4.2.4 Trial 5: EATOP Analyzer....18
4.2.5 Trial 6: Formal Verification....19
4.3 Evaluation of optimization approaches....21
4.3.1 Trial 7: OptiPAL....21
4.4 Modelling infrastructure....23
4.4.1 Trial 8: Model Exchange....23
4.4.2 Trial 9: AUTOSAR Gateway....24
5 MAENAD Analysis Workbench Evaluation....26
5.1 Dependability Modelling and Safety Analysis with HiP-HOPS....27
5.2 ASIL allocation with HiP-HOPS....32
5.3 Timing Analysis with Qompass....35
5.4 Simulink Gateway....38
5.5 Functional Mock-up Unit Import....43
5.6 EATOP Analyzer....47
5.7 OptiPAL....49
5.8 Model Exchange....56
5.9 AUTOSAR Gateway....60
5.10 Model-checking Gateway....63

6 Verification and Validation 67
6.1 Fault Injection 67
7 Conclusion 72
8 References 73

1 Executive Summary

The goal of the MAENAD project is the development of a methodology and the related tools aimed to an optimal design of electric vehicles, providing support for the management and transfer of information both during the product development lifecycle and during the concept phase to identify optimal architecture based on objective evaluation criteria. MAENAD also includes the development of tools to support the design of embedded systems.

The methodology heavily relies on the paradigm of Model-based design, and is implemented by:

- extending the capabilities of the EAST-ADL architecture description language in order to describe and capture the distinctive characteristics of FEV(s);
- extending the capabilities of tools that rely on the language, to support the design of automotive embedded systems based on the new engineering scenario and design needs;
- developing dedicated tools that, starting from the language, extract the necessary information and provide support for the optimal choice of architectures based on criteria of quality and cost, using the automatic exploration of possible architectural variants.

In the context of the MAENAD project, Work Package 6 is responsible for assessing the effectiveness of the methods and tools developed during the project.The assessment is performed using and applying these methods and tools to case studies, representative of the engineering scenarios from which the MAENAD project arises.

In particular, the purpose of Work Task 6.1 is concerned with modelling aspects related to the E\E architecture of FEV(s) through the use of the EAST-ADL language and modelling tools, and to exercise the analysis tools toward the project objectives, of which this report reflects and formalizes the activities. The results of the analysis have a direct impact on WT6.1 regarding modelling aspects, for refining of the case studies if they fail to cover the various aspects related to the design of a FEV(s) (Gap Analysis), and constitute an input to WT6.2, which aims to reflect these results on the methodology and language, interfacing with other WP and coordinating appropriate changes accordingly.

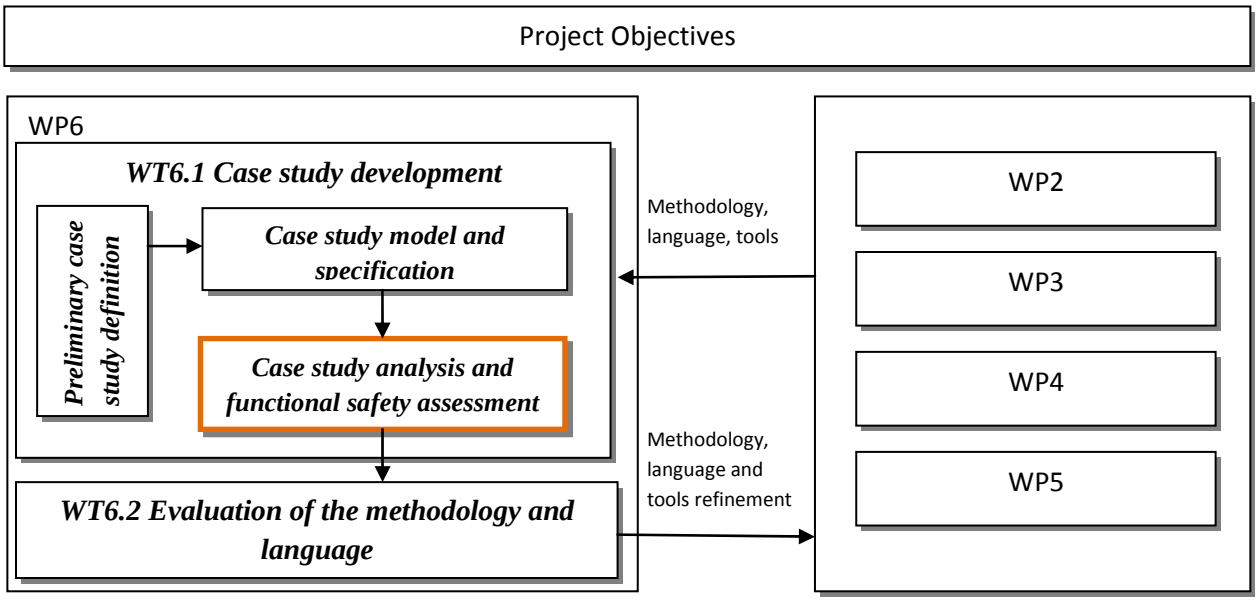


Figure 1-1: WP6 structure and relationship

2 Introduction

2.1 Purpose and target group

The purpose of this deliverable is to:

1. Establish an evaluation framework for the MAENAD project, providing the main trials, steps and analysis approach to be carried out by the partners for measuring the achieved quality of the project outcomes.
2. Summarize the evaluation activities performed by the various working groups performed on the MAENAD Analysis Workbenches and related tools.
3. Report the results of the evaluation activities.

The deliverable is structured as follows:

- Trials: describes the evaluation methodology for each tools/plug-in step by step, with the evaluation criteria.
- MAENAD Analysis Workbench: report the results of the evaluation activities, the related tools and procedures.

2.2 Relation to other project activities

As evaluation is relevant to almost all of the work packages (WP), and, in addition, requirements gathered for the MAENAD project by WP2 have been committed and established across all the WPs, almost all the MAENAD deliverables have an impact on the evaluation activities.

2.3 Analysis Approach

As previously described, the MAENAD project heavily relies on model-based design and associated techniques to meet the project goals, that is, to provide a comprehensive development framework for the design and development of FEV(s).

The core of the project relies on the enhancement of the EAST-ADL modelling language. The language EAST-ADL is a Domain Specific Language, expressly designed with the focus on the automotive electronic systems. The language offers an effective means to capture and transfer information related to an automotive electronic system across the different layers of the supply chain, as well between application engineers, providing a structured set of viewpoints on the overall architecture and the capability for generating others.

The language itself doesn't provide capability for simulation and analysis, but provide means to reference models coming from external tools. Furthermore, a set of plugins and tools built on top of the language will be enhanced or developed as extension points to provide support for the analysis of complex systems, or to provide models transformation capability to create a link with external analysis and simulation tools.

The solution of the project for the main goal is focused on different fields of intervention:

- Develop capabilities for modelling and analysis support, following ISO 26262, with the main purpose to extend the capability of the EAST-ADL language to support ISO 26262 development process and analysis (FMEA, FTA, risk analysis, ASIL allocation,…).
- Develop capabilities for prediction of dependability & performance.

- Develop capabilities for design optimization. This includes the support for the selection of optimal FEV architectures, starting from attributes like performance and cost.

The objectives of WP6 are derived from the main project objectives, and this report will summarize the activities toward their fulfilment:

- O4-1: Evaluation of ability to support ISO 26262 and other standards influencing FEV.
- O4-2: Evaluation of dependability & performance analyses.
- O4-3: Evaluation of optimization approaches.

The fourth objective, “O4-4: Evaluation of suitability of overall methodology for FEV design”, is implicitly satisfied by addressing the above three objectives, and will be a joint work between WT6.1 and WT6.2, in which WT6.1 will exercise MAENAD methods and tools through case studies, collect analysis and experimental results and transfer these results to WT6.2.

The latter WT, starting from those data, will evaluate the fitness of purpose from OEM(s) and stakeholders perspective.

MAENAD can be seen as a project addressing multiple concerns with the goals to provide methods and tools to support development of FEV. A standard "V-cycle" development model can be applied to the project for the delivery of intended outcomes. User requirements are collected by WP2, technical requirements are derived from them and SW development activities and language refinements are performed.

Starting from those concerns and from the project objectives, the analysis of the outputs of the project follow a **verification/validation** approach.

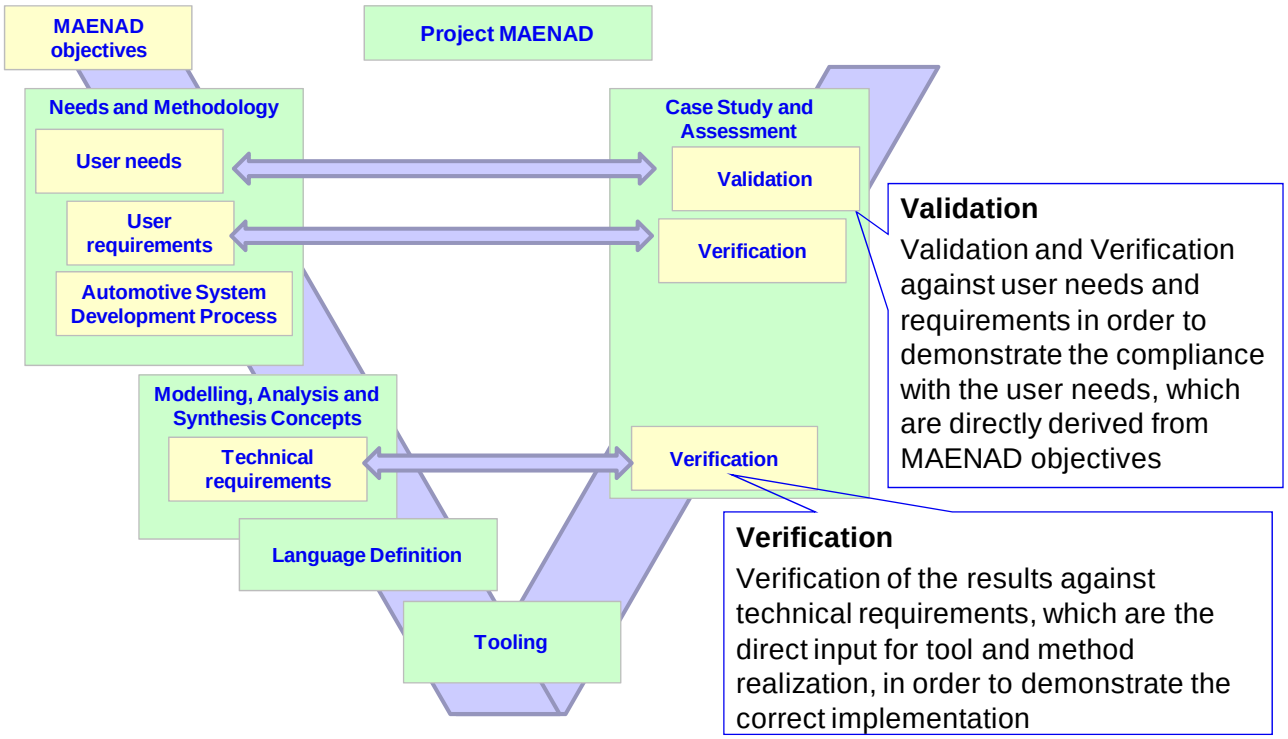


Figure 2-1: assessment concept

During the verification phase, the correct implementation of the technical requirements and specification is evaluated.

During the validation phase, it will be evaluated whether the project outputs satisfy the user needs (e.g. the User requirements). This phase will focuses on the investigation of the effectiveness of the developed tools to support the analysis and decision making during the design of a FEV. The work will report the results to WT6.2, as a basis to provide inputs for WP2 “Requirements & Methodology”, WP4 “Language” and WP5 “Tooling”. Part of the activities will be also on the evaluation about the completeness of the analysis that the WP6 will perform on the project outputs. This analysis will serve as a basis for the successive refinements of the use cases inside WP6, in order to guarantees that all the project outcomes will be evaluated.

3 Verification

During the first part of the project, requirements, needs and use cases have been collected and assigned to work packages. Those requirements cover aspects related to engineering needs and scenario relevant for the design of FEV(s), and include traits derived from an exhaustive analysis on FEV standards and regulations as well as best practices and state of the art methods for the development of complex automotive systems.

The requirements collected have been implemented as new properties and attributes on the modelling elements of the language or as guidelines and modelling patterns using existing constructs.

The verification of the requirements concerning analysis activities consist of a review of the project outputs (language and modelling patterns, tools…) in relation to the requirements collected by WP2. A "coverage criteria" is adopted to assess progress.

The table below show an example of matrix adopted for the analysis

High level user requirement		Medium level user requirement (the text may be more extensive than shown in the box)							Tools	Validation criteria			Validator		
Ref.		Subject	Language/Modelling	Req. ref.	Analysis	Req. ref.	Methodology	Req. ref.		Implem	Complet	Easy of	Demo 1	Demo 2	Demo 3
4SG 7	EV safety standards/ ISO 6469-1	Insulation	- Insulation symbols - Insulation attributes (withstand		Insulation analysis (overall resistance,		- Deployment of insulation resistance - Addressing insulation monitoring								
		Heath generation					Designing a monitoring system to prevent dangerous effects to persons,								
		RESS over-current interruption	Modelling of an over-current interruption device		RESS short circuit analysis (current		- Designing an overcurrent interruption device								
4SG 8	EV safety standards/ ISO 6469-2	Connection of the vehicle to an off-board electric					Designing a means to make impossible to move the vehicle when connected to								
		Indication of reduced power					Designing a warning to signal to the driver that the propulsion power is								
		Driving backwards					Designing means to prevent unintentional switching in reverse								
		Parking					Designing a warning to indicate whether propulsion is in the								
		Protection against failures					In functional safety development, include unintended acceleration,								

Figure 3-1: Requirements based analysis matrix example.

3.1 Verification Activities results

At present, requirements have been collected and consolidated by WP2. Whiting the project, WP6 supported the WP2 activities providing feedback on requirements coverage.

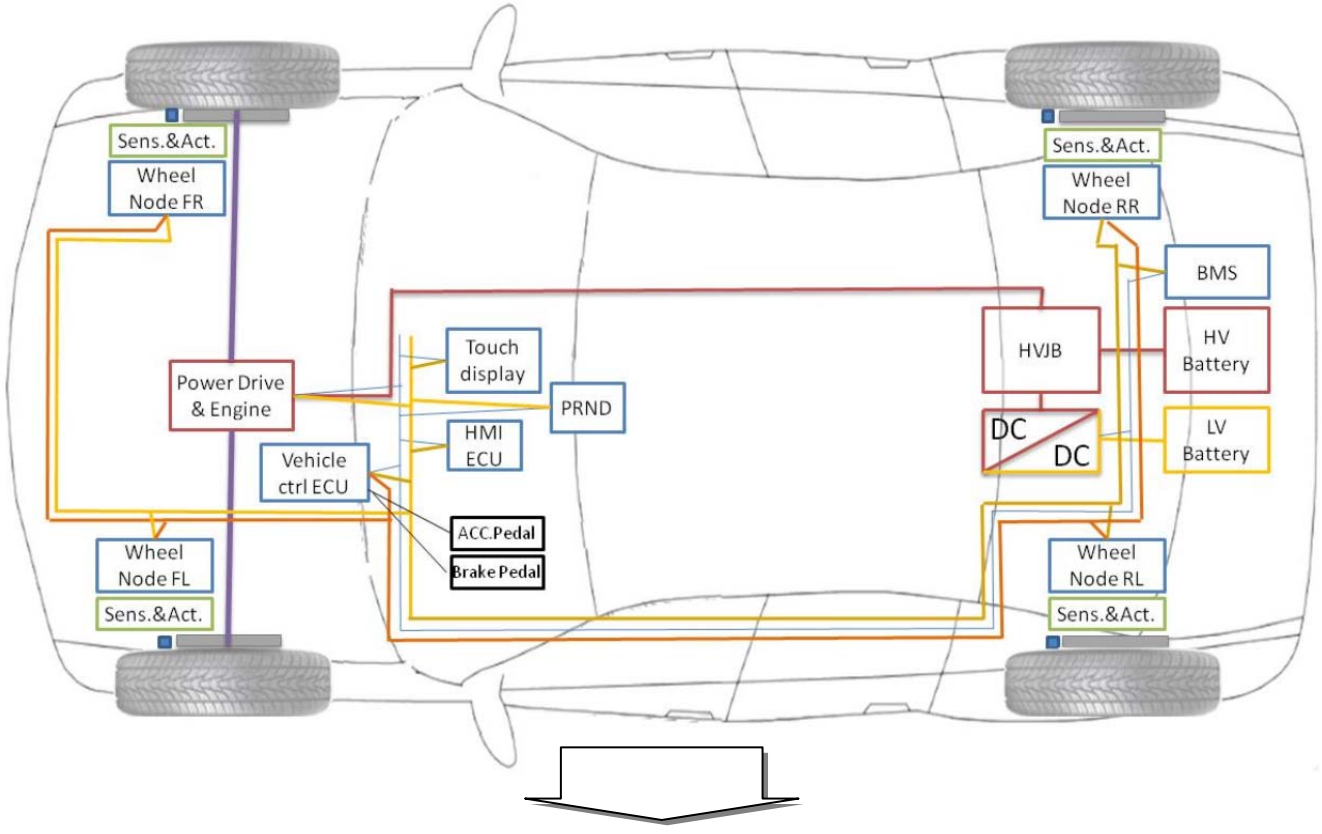
4 Trials

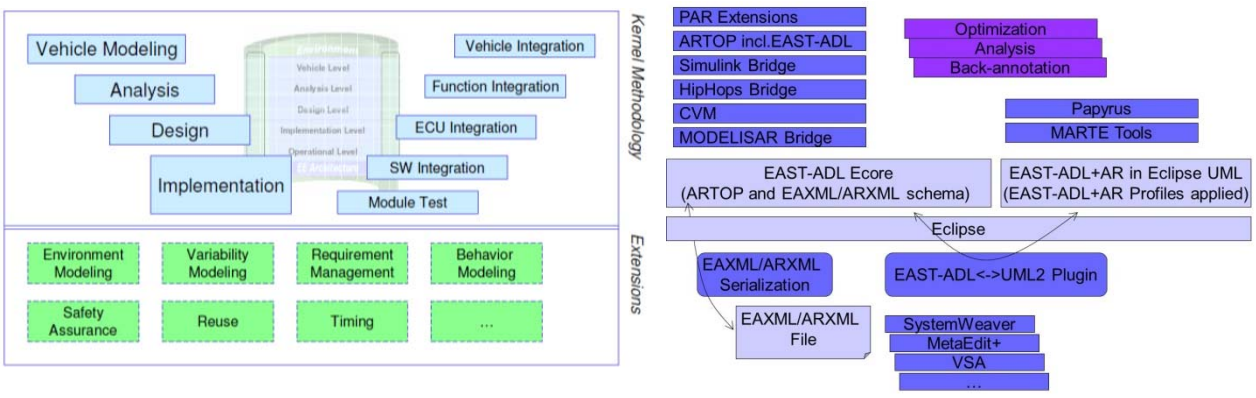
The purpose of this section is to specify analysis activities to experiment the MAENAD Analysis Workbench, to assess the progress toward the project objectives, with the main focus on the Maenad Analysis Workbench

The analysis is conducted by exercising the tools with three different case studies, representative of FEV(s) subsystems:

- Propulsion and power distribution subset with the associated interlock functionality for safety features, and the driving mode selector management
- Regenerative braking systems based on an innovative brake by wire distributed architecture.
- Driving mode management for electric vehicle, with energy supervision algorithm to support the driver in critical range situation, as well the related HMI to interact with the driver

The picture below provide an overview of a simplified E\E architecture which combine the three subsystems, and that will be used to exercise the MAENAD concepts,





Each case study address different aspects of the MAENAD methodology (EV development process, ISO 26262 safety analysis, language capability, modelling aspects, dependability methods, optimization algorithm).

The topics addressed by the analysis are listed below.

- ISO 26262 safety analysis activities
- V&V (Fault injection)
- Language capabilities
- Timing analysis
- Dependability & Performances analysis
- Optimization
- Formal verification

The following list summarize the main artefacts of the Analysis Platform that are subject of the evaluation:

- AUTOSAR Gateway
- Timing Analysis
- Simulink Gateway
- HiP-HOPS Gateway
- Architecture optimization and configuration (Optipal)
- Plugin for EAST-ADL exchange format EAXML
- MODELISAR FMU Import

Analysis tools have been divided into categories according to the related engineering and technical domain that they address.

The trials specify, for each tool that belongs to the Analysis Platform, the evaluation step and procedure to be carried out to evaluate the technology, and the expected results. This section doesn't include the results of the evaluation activities; these are presented later in this deliverable.

It also doesn't include the evaluation itself, only the methodologies, process, procedures, templates etc.

4.1 Evaluation of ability to support ISO 26262

This chapter introduces the analysis that will be carried out to evaluate the capabilities of the project outputs to support the ISO 26262 automotive safety standard.

At the present, the EAST-ADL language already includes semantic features to support the development of safety related embedded control systems. The language provides constructs, at different abstraction levels, to capture and transfer information across the supply chain in line with ISO26262. The support from the language covers the concept phase of safety lifecycle, providing semantic for the definition of the Item, traceability of requirements, Hazard identification and risk analysis (definition of hazard, hazardous event, safety goal, ASIL, severity, …) , functional safety concept (safe state, fault tolerant time interval,…) as well during the definition of the technical safety concept.

In addition, the language also provides support for the error modelling, enabling the specification of the behaviour of entities in the presence of errors. Analysis support is provided by external tools.

4.1.1 Trial 1:HiP-HOPS Gateway

Support for safety analysis (FMEA, FTA, ASIL decomposition) is provided externally by the HiP-HOPS analysis tool. The link between the modelling environment and the analysis tool is provided by a dedicated gateway plugin that enables the export of EAST-ADL models (annotated with the

necessary error modelling information) to the HiP-HOPS format. This can then be read in by HiP-HOPS to perform the analyses.

The input to the HiP-HOPS plugin is an EAST-ADL error model annotated with HiP-HOPS-compatible failure propagation logic. The error model provides the structural information about the system while the logic provides the information about the failure behaviour of each system entity, which HiP-HOPS uses to model the propagation of failures through the system. The propagation logic (provided in ErrorBehaviours) takes the form of Boolean expressions that relate output FaultFailures of an ErrorModelType (representing e.g. components, functions etc.) to a combination of input FaultFailures and internal failure modes. The model is then transformed by the plugin so that HiP-HOPS can read the information and perform automatic FMEA and FTA.

For ASIL decomposition, the exported error model must also contain relevant ASILs assigned to system-level failures (i.e., those which cause hazards). ASILs are provided in EAST-ADL using SafetyConstraints assigned to the FaultFailures (input/output faults) and optionally InternalFaults (component/function failure modes) of the error model. HiP-HOPS uses its internal model of the system failure propagation to determine which low-level failure modes contribute to which system-level failures (and thus which system-level ASILs) and therefore determines which combinations of ASILs can be assigned to the failure modes and input/output faults of the system.

At present, EAST-ADL and HiP-HOPS provides a good infrastructure for the support of automatic FMEA/FTA analysis. The support for automatic ASIL decomposition is still being developed but prototype support exists; the intention is for the ASIL decomposition capability to become fully integrated, as with FMEA/FTA support. Future enhancements will include the import of the analysis results back into the model and improvements to the exchange of information between the modelling environment and analysis tool, as well as more scalable and effective ASIL decomposition algorithms.

Key points for the analysis

- Availability of modelling support for safety oriented analysis in MAENAD Modelling Workbenches
- Effectiveness of translation between EAST-ADL models and HiP-HOPS environment through model transformation (HiP-HOPS gateway).
- Effectiveness/correctness of the results of the automated analysis.

Analysis strategies

ITEM	**Evaluation Steps**
Modelling support for safety oriented analysis in Papyrus environment	1.a Annotation of an Existing Use Case model with the safety related attributes in the different abstraction levels of the language in the development environment. Evaluation: Capability of the tools to support the modelling needs and constructs.
Automatic translation between EAST-ADL and HiP-HOPS models	1.a Translation of an annotated use case model to an intermediate format through HiP-HOPS Gateway and import of the file into HiP-HOPS. Evaluation: Level of automation of the gateway. Correctness of the information exchanged between the tools. Review of the correct import of the safety analysis results in the original model.
Effectiveness/correctness of the Automated FMEA analysis	Analysis will be performed automatically by HiP-HOPS tool.

Effectiveness/correctness of the Automated FTA analysis	Evaluation: Review of the results by safety experts through safety reviews. Evaluation metrics will be the correctness of results (FMEA, FTA, ASIL decomposition) and "usability" of the results (ASIL decomposition, for the support to identify a minimal set of optimal solutions among all the possibilities). Validation of the safety goals related to the technical/functional safety requirements implemented through physical fault injection and test bench.
Effectiveness/correctness of the Automated ASIL decomposition	

4.2 Evaluation of performance analyses

This chapter is all about the analysis that will be carried out to evaluate the capabilities of the MAENAD Analysis Workbenches for performance analyses.

Support for performance analysis will be provided through dedicated plug-in for link with Timing analysis engines and state of the art simulation environments.

4.2.1 Trial 2: Timing Analysis plug-in

In the Modeling Analysis Workbench, a dedicated plugin will be provided to perform timing analysis. It will be based on an adaptation of an existing plugin called Qompass, developed by CEA List. This plugin is intended for MARTE models. It takes as input: 1) a description of a software component architecture, each component features operations that can be executed and depending on their allocation to hardware elements require various communication mechanisms to interact with one another – inter-variable communication is assumed when two components are on the same node for instance; 2) software allocation to an hardware architecture (Nodes and buses) whose performance are quantified with some parameters such as processor speed; 3) a list of execution scenarios describing the flow of execution across parts of the software component architecture – each computation or communication steps feature worst-case execution time, each scenario features a timing constraint, typically a deadline for the completion of the flow; 4) a scheduling policy chosen and an allocation of computation and communication steps to tasks on each node – one can use a one-to-one mapping or different schemes, resulting in different loads for each task and hence for each node. The plugin provides an RMA schedulability analysis, which tells whether the list of scenarios as described are schedulable, i.e. deadlines are met or not – a parameter showing how much the system is over- or under-loaded is provided.

To analyse an EAST-ADL model, additional information will then have to be added manually by a user, at least in the first version of the plugin provided, typically the software allocation which is different from the function allocation, the tasks at the level of nodes and the scenarios. In later versions, manual intervention is intended to be reduced as much as possible.

During P2 the plugin has been considerably enhanced by adding:

- An automatic transformation that translates EAST-ADL schedulability relevant information to the MARTE schedulability model
- Enhanced schedulability analysis able to analyse functional allocation without the need of adding implementation -related information

During P3 has been enhanced to analyze the Brake-By-Wire case study model.

Key points for the analysis

- Availability of modelling support for time analysis in MAENAD Modelling Workbenches
- Effectiveness of automatic transformation
- Effectiveness/correctness of the results of the automated analysis.
- Suitability for performance analysis in line with ISO26262 (predictability of fault tolerant time interval, fault reaction time, diagnostic test interval)

Analysis strategies

ITEM	Evaluation Steps

Modelling support for timing oriented analysis	1.a Annotation of an Existing Use Case model with the timing related attributes in the different abstraction level of the language. Evaluation: Capability of the tools to support the modelling needs and construct. Formalization of timing requirements and properties in relation to structural model elements
Effectiveness of automatic generation mechanism for the missing modelling elements	Evaluation: level of automation to complement an initial EAST-AD model so that it is analysable: i.e. generation of a MARTE model based on End-to-end flow descriptions, allocation from function to hardware allocation
Effectiveness/correctness of the results of the automated analysis	Evaluation: comparison of the simulation results with experimental results performed through a physical test bench, fault injection, network time analyses.

4.2.2 **Trial 3: Simulink Gateway**

The Simulink Gateway is a set of tools\plugins providing a mechanism for bidirectional transformation between EAST-ADL models and Simulink equivalent models. The gateway is intended to provide support for simulation-based analysis to EAST-ADL conformant models, and to enrich to some extent the MAENAD Analysis Workbenches with the features that Matlab/Simulink can provide.

The Simulink gateway is composed of a User Interface to the Matlab/Simulink environment that supports the rule-based design of Simulink models conformant to EAST-ADL constraints, and a transformation engine that provides model to model transformation capability between models through intermediate representations.

Key points for the analysis

- Effectiveness of translation automation from EAST-ADL conformant models to Simulink behavioural models
- Suitability of the Simulink Gateway for Verification and Validation activities

Analysis strategies

ITEM	**Evaluation Steps**
Effectiveness of translation automation from EAST-ADL conformant models to Simulink behavioural models	1.a Export of an existing Use Case model developed with MMW with the Simulink gateway, import the equivalent model in Simulink environment Evaluation: Correctness of the transformation: • data flow, • data types • architecture elements mapping • hierarchical organization of the equivalent model • missing elements and completeness of transformation • Possibility to execute the equivalent model and effort to make it executable • Number of error/warning raised by the Simulink model

	checker
Suitability of the Simulink Gateway for Verification and Validation activities	Evaluation: • Possibility to export part of an EAST-ADL model (set up of a test bench, simulation of the environment, plant or non-available elements of an EE architecture) • Suitability of the model to be used in conjunction with Simulink design verifier, steps toward the completion of the model and time saved compared to a design without the export features.

4.2.3 Trial 4: MODELISAR FMU import

The MODELISAR ITEA2 project addressed the design of systems and of embedded software in vehicles. The main output of the project was a specification of an open standardized interface to be used between simulation environments to exchange models and for co-simulation, FMI (www.fmi-standard.org). A Function Mockup Unit (FMU) contains an executable and a Function Mockup Interface (FMI). Tools can use the Function Mockup Interface to properly execute several Function Mockup Units, and thus simulate several components without having access to its internal specification.

Providing the capability to exchange EAST-ADL models with the FMI format paves the road for model exchange and simulation of EAST-ADL models.

Key points for the analysis

The FMU import represents model synthesis from external tools, as it makes it possible to reuse the structure defined elsewhere, and to export to external formats. With additional tool support, it would also be possible to synthesize complete FMUs out of an EAST-ADL model. It would mean that components or subsystems of an EAST-ADL model could be simulated based on the MODELISAR concepts. It would require references to executable components from within the EAST-ADL model. At this point, however, the benefit is the consistent definition of component interface between an external component in MODELISAR FMU and its EAST-ADL model.

Analysis strategies

ITEM	**Evaluation Steps**
Evaluation of the capability to import MODELISAR FMI into an EAST-ADL model	Define an MODELISAR Function Mockup Interface specification file for a FMU. Run MODELISAR FMU import plugin Assess imported information transformed to EAST-ADL elements

4.2.4 Trial 5: EATOP Analyzer

EAST-ADL supports annotation of non-functional properties such as cost, power consumption or cable length. By linking requirements to such annotations, and specifying properties using the same annotations, there is enough information to compute total cost or power for a composition, and compare with the requirement. Fore mode-varying properties, it is also possible to define requirements and properties per mode.

Key points for the analysis

The analysis serves to validate and demonstrate how EAST-ADL supports multi-aspect annotations on a FEV-relevant example. The analysis will

- Identify relevant requirements and the corresponding refining annotations on a top element in an EAST-ADL hierarchy
- Compute the total cost/cable length/etc. for an EAST-ADL hierarchy based on the types' annotations.
- Compare the analysis result with the requirement
- Update the V&V elements of the model with analysis results

Analysis strategies

ITEM	Evaluation Steps
Evaluation of the analysis capability based on Requirements, V&V elements, GenericConsstraints and modes.	Define an EAST-ADL system hierarchy, Add requirements, modes and GenericConstraints Select Requirements to analyse Choose mode to analyse

4.2.5 Trial 6: Formal Verification

As a system architecture description language, EAST-ADL plays an important role for consolidating various kinds of behaviour concerns in the engineering of automotive EE systems. In MAENAD, an investigation of the EAST-ADL support for formal verification of behaviour centric system properties, based on the regenerative braking system case, will be carried out.

The aim is to validate the EAST-ADL support for formalizing various temporal concerns, such as during requirements engineering, function and execution design, safety engineering, etc. By aligning the EAST-ADL semantics with existing mature formalisms, one can then allow formal verification of such concerns through the corresponding external analysis engines. One advantage is that the EAST-ADL users will then obtain analysis leverage by model-checking. Compared to those standalone analytical models in external tools, EAST-ADL models complement with detailed architecture information and facilitate the integration of many related architectural aspects for the purpose of architecture design, safety engineering, reuse and change management.

Key points for the analysis

The most important objective of this case study is to validate the EAST-ADL support for temporal constraints as well as the claimed advantages to be brought in by EAST-ADL. This will be achieved through two existing mature formalisms: UPPAAL and SPIN. Both UPPAAL and SPIN allow exhaustive reasoning of the compositional consequence of behaviours. They are considered as two representative technologies in the area of formal verification.

- UPPAAL is a timed model checker for formal verification of real-time embedded systems (http://www.uppaal.com/). Based on timed-automata theory, UPPAAL provides support for modelling and simulating system behaviours in the form of compositional automata. The tool has been used in several industrial cases and is recently commercialized.
- SPIN is a model checker for formal verification of distributed and concurrent systems (http://spinroot.com). Compared to UPPAAL, the SPIN approach emphasizes the logical aspects of temporal behaviours. It deliberatively avoids the quantitative notion of time, but focuses on the interaction and synchronization of asynchronous processes. This simplification allows SPIN to verify the functional or logical properties of more complex system than timed model checkers usually do.

The intended language validation through UPPAAL and SPIN will be performed in the context of FEV development. By incorporating the analysis engines of UPPAAL and SPIN, the case study will prove and demonstrate the following features of EAST-ADL in regard to behaviour specification and formal verification:

- Supporting precise definitions of temporal characteristics for the definition and analysis of safety constraints (required by 4SG#0050, 4SG#0057, 4SG#0058, 4SG#0059)
- Supporting assessment of completeness and correctness of the safety requirements (required by 4SG#0048)
- Supporting the descriptions of driving profiles (required by CON#2001), physical dynamics (required by CRF#0006b, CRF#0007b), power management procedures (required by CRF#0010b, CRF#0011b, CRF#0013b, CRF#0014b, CRF#0015b), fault tolerance design (required by CRF#0017b, CRF#0018b)
- Supporting the generation and precise definition of test cases (4SG#0049a, 4SG#0050)
- Supporting the integration with external formalisms (CON#0017, CON#0018, CON#0019)

Analysis strategies

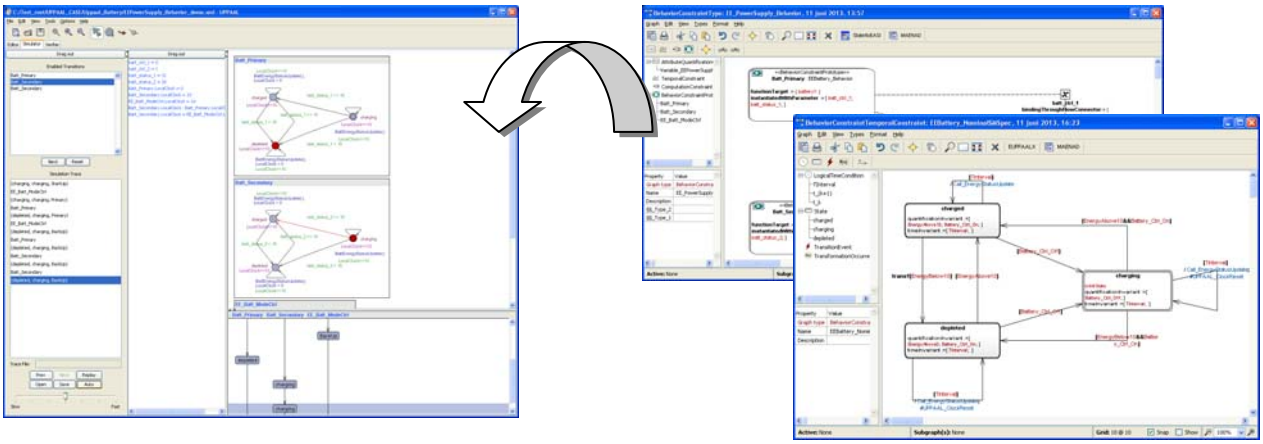
ITEM	Evaluation Steps
Validation of the EAST-ADL semantics for temporal constraint specification	Analyse the FEV requirements, and then elicit and specify the constraints on boundary conditions, modes and control logics, (end-to-end) timing. Formalize the temporal constraints, their targets and other traceable artefacts in EAST-ADL.
Validation the provision of analysis leverage through the incorporation of the UPPAAL&SPIN engines.	Formalize and assess the conceptual mappings between EAST-ADL and UPPAAL&SPIN models. Develop transformation algorithms and proof-of-concept tool support.

Analysis results

This case study will be based on the regenerative braking case system. An extension with emergency braking assistant function may also be used for the study of some detailed temporal characteristics. Main results from the case study, as proofs for the intended modelling features mentioned above, include:

- Report on the validation of EAST-ADL modelling support for temporal constraints in regard to mature formalisms represented by UPPAAL&SPIN.
- Example EAST-ADL models that provide precise definitions of temporal characteristics for the definition and analysis of safety constraints.
- Example EAST-ADL models and corresponding external analytical models in UPPAAL&SPIN for assessing the completeness and correctness of safety requirements.
- Example EAST-ADL models that provide precise descriptions of driving profiles, physical dynamics, power management procedures, fault tolerance design.
- Example EAST-ADL models that provide support for the generation and precise definition of test cases.
- Proof-of-concept solutions for the mappings and integrations of EAST-ADL with UPPAAL&SPIN.

See Figure 4-1 for an illustration of the intended mapping from EAST-ADL declaration to UPPAAL model for the verification of temporal properties.



UPPAAL Analytical Model **EAST-ADL Model**

Figure 4-1. EAST-ADL temporal constraint model and its UPPAAL correspondence.

4.3 Evaluation of optimization approaches

The current concept for optimization in EAST-ADL is to develop an optimization architecture built around a central engine based on genetic algorithms that can leverage existing variability constructs in EAST-ADL and previous variability development work as well as linking with the various external analysis tools (for safety, timing, behaviour analysis etc.).

4.3.1 Trial 7: OptiPAL

As planned, the input to the process will be an EAST-ADL model that has been augmented with variability elements to define the potential design space (by establishing possible alternative architectural choices to be experimented with) as well as all data necessary for the analyses/evaluation of each object (e.g. an error model for safety analysis, etc.). An initial function called the Optimization Space Definition Module (OSDM) will take this model and determine the possible design space. The Central Optimization Engine (COE) then uses this as the basis of the optimization process, experimenting with different possibilities. Each design candidate is passed to the Variability Resolution Mechanism (VRM), which generates a new model variant with all optimization variability resolved for that given design candidate. This model variant can then be analysed by the external analysis tools and evaluated on the basis of their results. The COE then tries new possibilities, keeping good candidates and discarding poor candidates, until it is told to stop. It then presents the list of good candidates it has kept so far, which form a Pareto front (a set representing optimal trade-offs between the various multiple optimization objectives - e.g. minimize cost, maximize safety etc.). Each 'optimal' candidate can be generated on demand from the original model by the VRM, so the optimization is not creating and storing a huge set of variant models.

Key points for the analysis

- Availability of language construct for the definition of cost constraints
- Ensure the generated 'optimal' candidate models are coherent and plausible

Analysis strategies

ITEM	Evaluation Steps
Modelling support for optimization oriented analysis	1.a Annotation of model elements of an existing use cases with attributes necessary for optimization analyses (cost function, error, timing,...) Evaluation: support provided by MMW for the definition of attributes and constraints related to optimization analysis
Coherent and plausible generation of optimal candidates models	1.a Extension of an existing Use case to include different vehicle\architecture variants. 1.b Annotation of the modelling elements of each architecture variants with the necessary attributes to perform optimization analysis. 1.c Run the plug-in Evaluation: Assessment of the results based on engineering experience

4.4 Modelling infrastructure

4.4.1 Trial 8: Model Exchange

The MAENAD Workbench heavily relies on tools and plugin from different tool vendors and institutions. Support for a neutral data exchange format is recommended to grant interoperability between tools and achieve a higher acceptance of the project outputs.

An XML-based exchange format (EAXML), allows tools to exchange EAST-ADL models between UML2 tools and an AUTOSAR-compliant XML format. The schema for the EAXML exchange format is automatically generated from the EAST-ADL meta-model.

At the present, import export support capabilities based on EAXML exchange format has been implemented between MetaEdit+ and SystemWeaver development environment.

Import/export features are currently limited to the Hardware Design Architecture, to investigate the feasibility of the concept.

Key points for the analysis

- Export capabilities of models built on “X” development environment to EAXML exchange format.
- Import capabilities of models on “X” development environment from EAXML source
- Bidirectional and consistent exchange of information between tools that belong to different tools provider.

Analysis strategies

ITEM	Evaluation Steps
Model import/export	1.: Export of an existing Use Case model to EAXML exchange format using tool “X”. 2:Import of the EAXML file generated in step 1 using tool “X” Evaluation: consistency check of generated EAXML output through file review, Models generated through EAXML import shall match the original
Model Exchange	1.a: Export of an existing Use Case model to EAXML using tool “X” 1.b: import the EAXML file using tool “Y”. 1.c: Changes of some trait of the imported model with tool “Y” environment 1.d: Export of the modified model to EAXML from tool “Y” 1.e:import the EAXML file in tool “X” Evaluation: consistency check between the models in the two different development environment, absence of loss information, changes on the model shall be correctly reflected in the two environments

4.4.2 Trial 9: AUTOSAR Gateway

AUTOSAR is an open and standardized automotive software architecture, jointly developed by manufacturers, suppliers and tool developers.

EAST-ADL complements AUTOSAR in several ways. The EAST-ADL metamodel is specified according to the same rules of the AUTOSAR metamodel, and AUTOSAR elements can be referenced by EAST-ADL elements. The result is a compositional metamodel where functional architecture aspects and software architecture aspects coexist, and are capable of enriching AUTOSAR with timing constraints, safety constraints and support for safety analysis, support for requirements engineering, variability management.

The goal of the AUTOSAR gateway is to provide an effective means for automatic model transformation from an EAST-ADL design architecture to an AUTOSAR software architecture, supporting the engineers during the design phases.

More precisely, the AUTOSAR gateway takes an EAST-ADL model as an input, i.e. an EAST-ADL design function architecture and hardware description architecture. From this, an AUTOSAR architecture is generated which tries to export as much information as possible. Essentially the core transformation is to map EAST-ADL each elementary function to a runnable entity. Runnables are regrouped into the software components, based on the grouping of elementary functions within non-elementary ones. Also the allocation of elementary functions on the hardware nodes is taken into the account while regrouping runnables into the software components. This prevents regrouping into the same software components those runnable entities that correspond to the elementary functions, mapped to the different nodes. Apart from the software architecture and internal behaviour of each software component, AUTOSAR gateway generates communication, ports, and their interfaces. It also proceeds with the generation of a hardware topology and finally, allocation of software components into the ECUs.

It is significant that now the generation is a twofold process. In the first step, user can generate the AUTOSAR model specified with the created AUTOSAR UML profile. This enables to perform the changes on the generated AUTOSAR architecture within the modelling workbench. The second step of the transformation will take as an input an AUTOSAR model expressed with the AUTOSAR UML profile and will generate the AUTOSAR compliant, XML file – ARXML.

Key points for the analysis

Generation of an AUTOSAR architecture modelled with the AUTOSAR UML profile.

- Comparison between an expected and an obtained result of a generation.

Generation of an AUTOSAR xml (ARXML) file from the model stereotyped with the AUTOSAR UML profile.

- Validation if all the elements expressed with the AUTOSAR UML profile were transferred into an ARXML file.
- Correctness of a generated ARXML file in terms of a compatibility with the standard.

Analysis strategies

ITEM	Evaluation Steps
Comparison between an expected and an obtained result of a model to model generation (EAST-ADL model to AUTOSAR model)	Evaluation: Define the expected results of a generation, taking into account implemented strategy for the generation of a software architecture, and compare it with the actual result.

Validation if all the elements expressed with the AUTOSAR UML profile were transferred into an ARXML file.	Evaluation: Generated from an AUTOSAR model (expressed with the AUTOSAR UML profile) the ARXML file should be imported/opened with an AR compliant design tool. Assessment will be done based on: • The effective capability to import the model in the new design environment • Consistency of the AR SW architecture with the AUTOSAR model expressed with the AUTOSAR UML profile • Plausibility of the auto-generated communication mechanism between AR runnables based on original partitioning constraints

5 MAENAD Analysis Workbench Evaluation

The purpose of this section is to describe the analysis activities performed to experiment the MAENAD Analysis Workbench.

5.1 Dependability Modelling and Safety Analysis with HiP-HOPS

Through its Dependability package, EAST-ADL allows a model-based approach to ISO26262 by structuring and formalizing the related concerns. Based on this, a wide range of user support has been developed, including safety analysis, traceability and consistency management.

Procedures

Dependability Modelling with ME+

One key step in safety engineering is about risk assessment, for which the definitions of items, hazards events and safety goals are of particular concern. With EAST-ADL dependability modeling support, the safety engineers capture these concerns as well as their dependencies according to ISO26262. See Figure 5-1 for an example case.

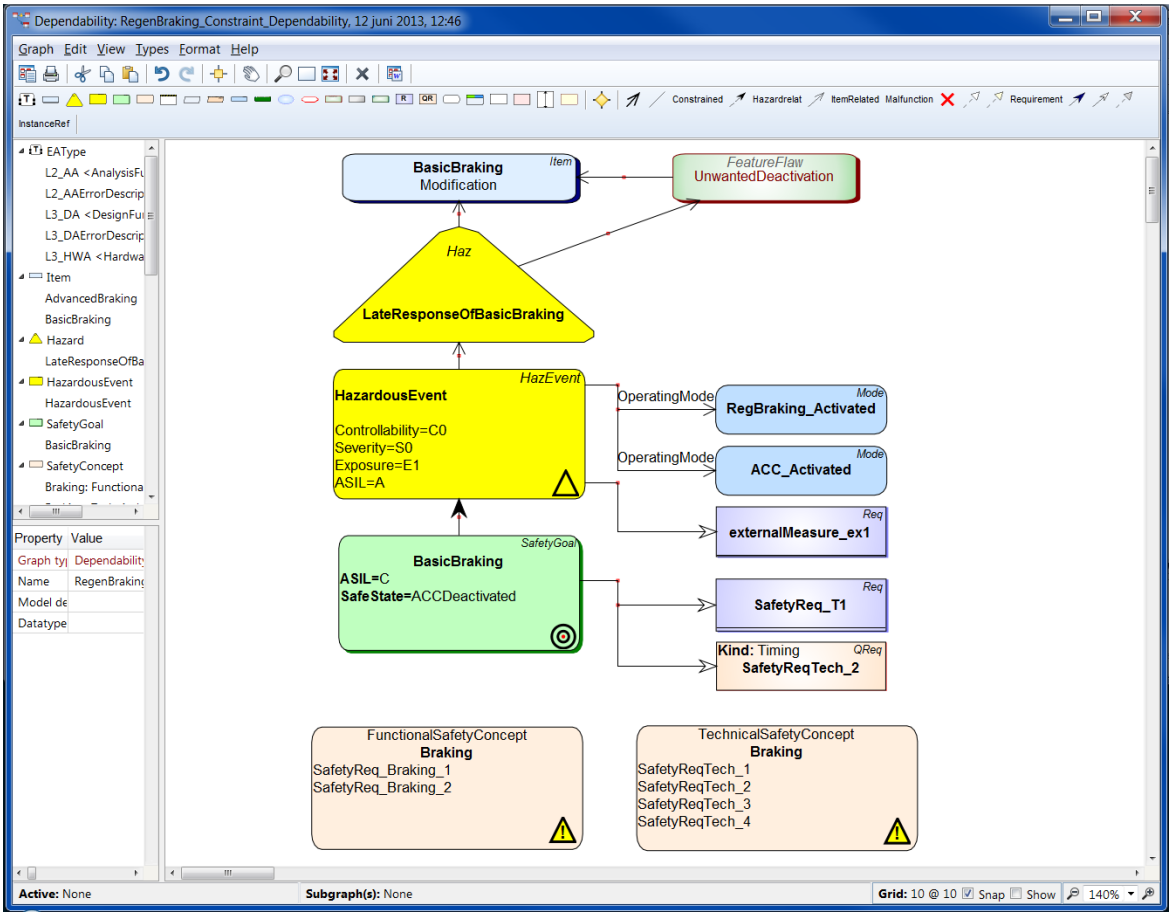


Figure 5-1: Dependability model structuring information related to a safety goal.

Based on the language, a well-maintained traceability from the safety concerns to related requirements and artefacts in system specification is achieved. The figure below shows one result of the ME+ tool support for automatically deriving the related nominal architecture solutions that an safety item (BasicBraking) has due to its FeatureFlaw declaration (UnwantedDeactivatoin).

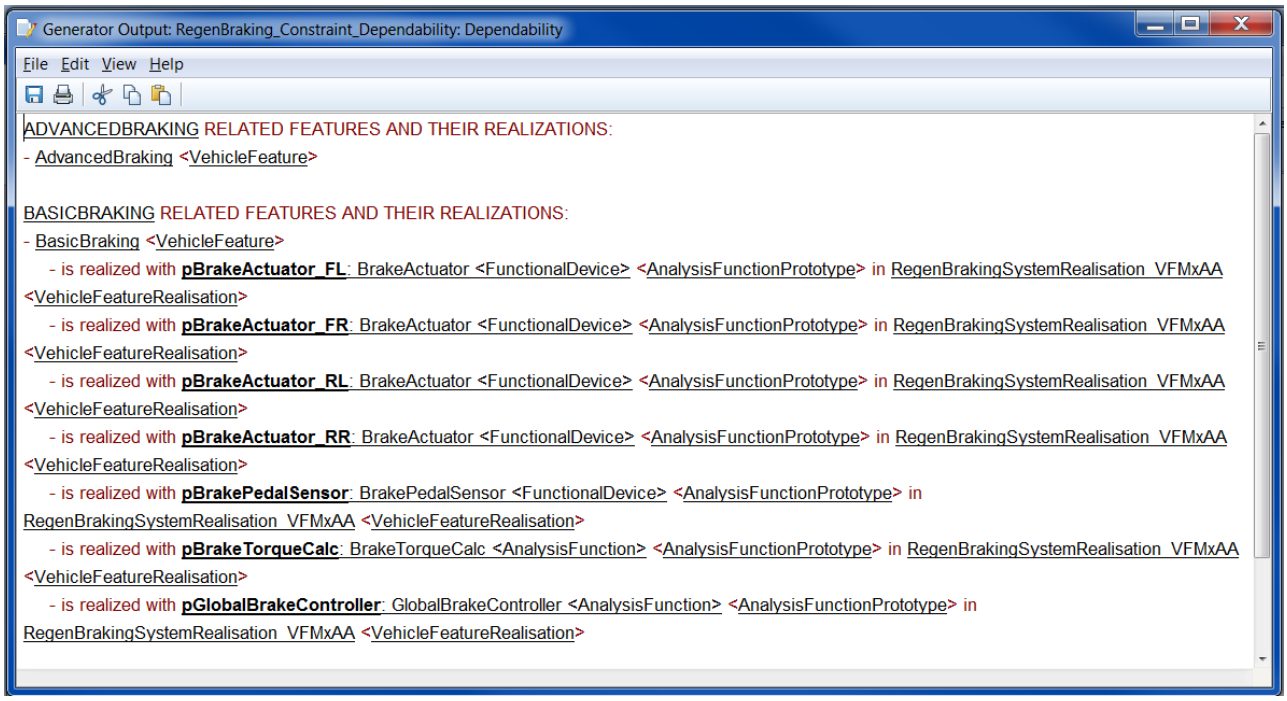


Figure 5-2.Error model defining the faults and error propagations of target system hardware and function.

Error Modelling with ME+

To derive more detailed functional and technical safety requirements, error modeling and safety analysis in terms FTA&FMEA become necessary. To this end, EAST-ADL allows the engineers to precisely declare the plausible error behaviors of a system and thereby to synthesize the fault trees automatically. Figure 5-3 below shows the error model of a braking system, where the blocks represent the error models of individual functional units and the links represent the error propagation channels due to communication links or allocation relations in the design.

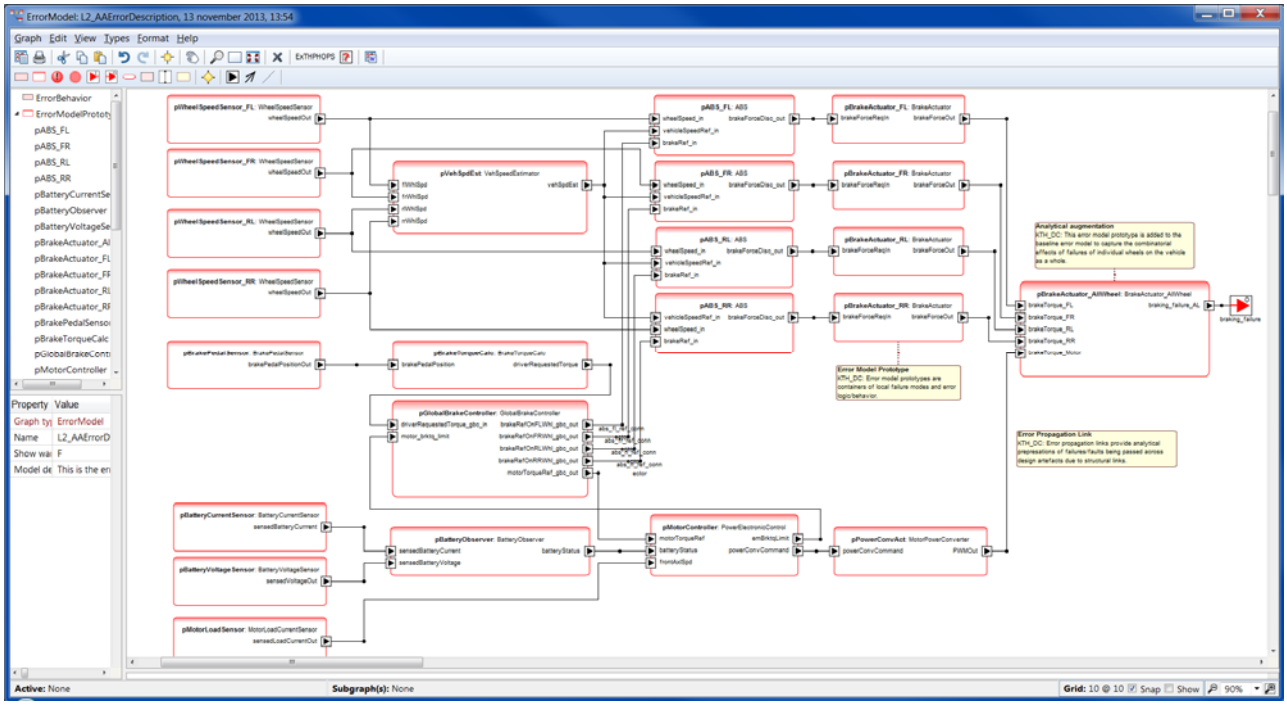


Figure 5-3.Error model defining the faults and error propagations within a braking system.

For the reason of usability, the ME+ tool can automatically generate an initial error model that contains the error model blocks and error propagation links according to the structure of

corresponding architecture model. Within each error model block, there are declarations of error behaviors capturing the logics relating some failures to the faults causing their occurrences. Figure 5-4 shows the error behaviors declared within the error model pBrakeActuator_AllWheel (shown as the rightmost block in Figure 5-3) for capturing the combinational effects of individual wheel braking failures. Currently, the declarations error behaviors need to be added manually by the engineers.

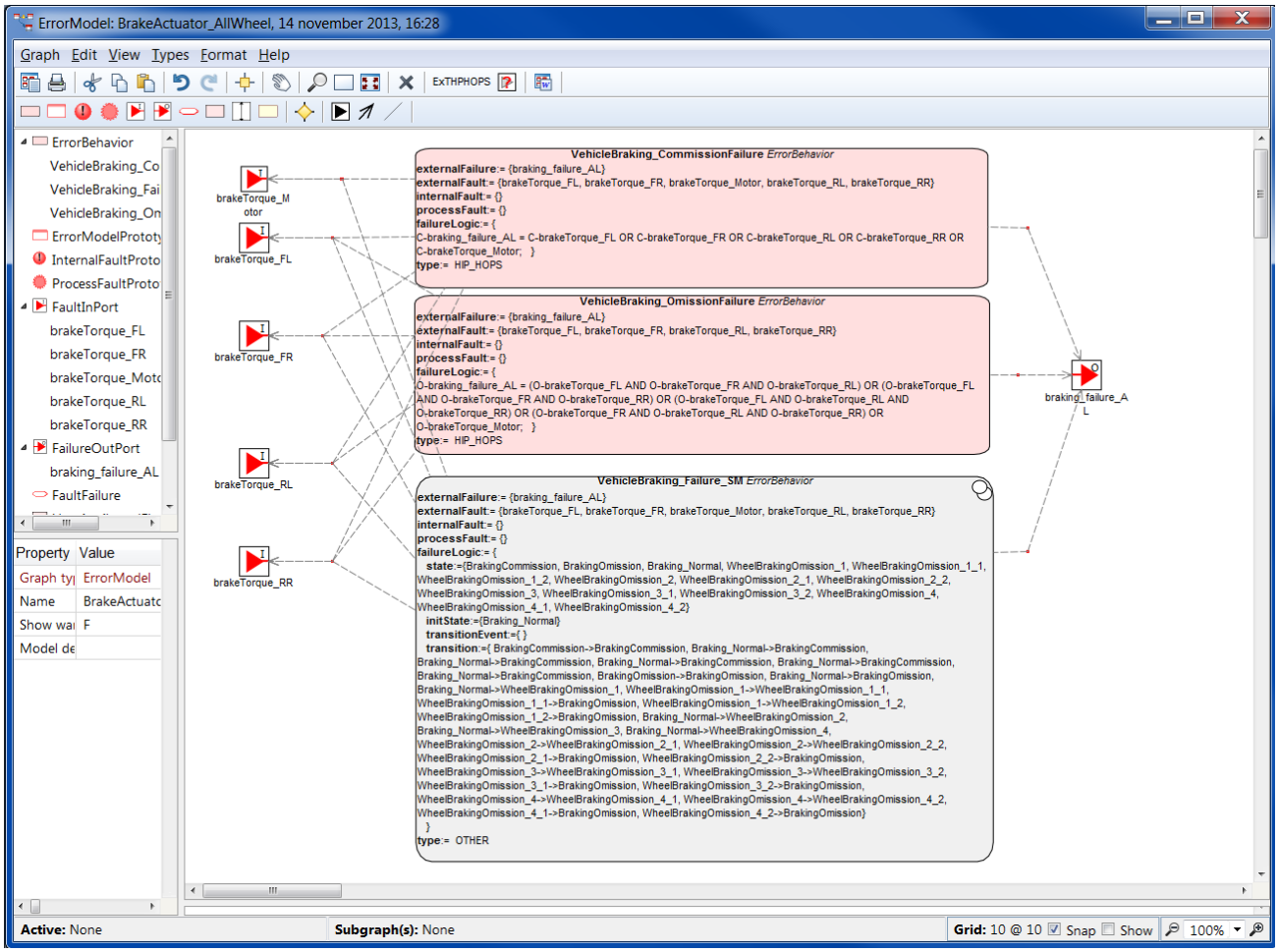


Figure 5-4. Error behavior declarations for the error logic of a system function. The upper two error behavior declaration blocks contain HiP-HOPS Boolean logic expressions. The bottom one contains a SM based specification.

The exact formalism for a failure logic description is chosen according to the analysis methods of interest as well as the complexity of error logic. The current ME+ implementation allows Boolean logic expressions according to HiP-HOPS and state-machine descriptions according to EAST-ADL Behavior Constraint Annex. This latter variant is shown by the bottommost error behavior block in Figure 5-4, with its failure logic definition given by the SM model shown in Figure 5-5.

With the specification of error model and its error behaviors, the HiP-HOPS plugin of ME+ parses the model and creates a HiP-HOPS input file (.hip) for static safety analysis. See Figure 5-6. For a state-machine (SM) based specification, an additional parsing step is performed to convert the SM specifications to HiP-HOS native expressions (this additional step is supported by a parser developed by Univ. of Hull).The final analysis results for the case system by the HiP-HOPS tool is shown in Figure 5-7 and Figure 5-8.

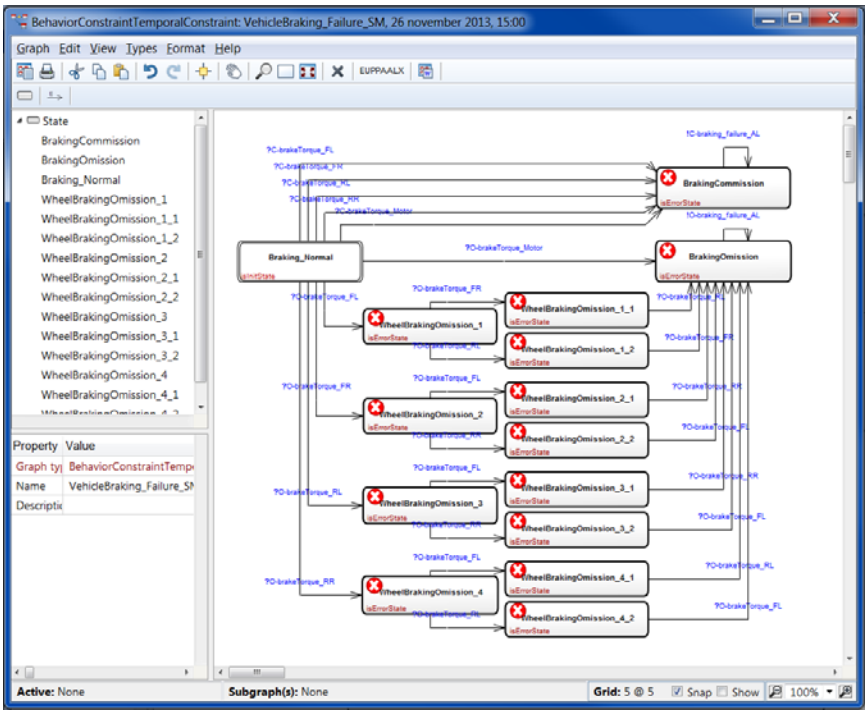


Figure 5-5. A SM based specification of top-level error behaviors of vehicle braking.

```
Model {
        Name "<<ErrorModel>>L2_AAErrorDescription"
        Description"This is the error model for the analysis architecture"
        System {
                ...
                Block {
                        Blocktype "<<ErrorModelPrototype>>"
                        Name "pABS_FL"
                        Port {
                                Porttype "<<FailureOutPort>>"
                                Name "brakeForceDisc_out"
                                Description ""
                                }
                        ...

                        FData {
                          Implementations {
                              Implementation {
                                Current
                                ...
                                Description "extracted error behavior descriptions"
                                ...
                                Basic Events {
                                        Basic Event {
                                        Name "ABS_VariableFailToUpdate"
                                        Description ""
                                        ...                     }
                                        ...
                                        }
                                        Output Deviations {
                                        ...
                                        Output Deviation {
                                        Name "O-brakeForceDisc_out"
                                        ...
                                        "O-brakeRef_in OR O-vehicleSpeedRef_in OR O-wheelSpeed_in OR ABS_VariableFailToUpdate"}
                                        }
                              }
                            }
                          H_BlockType "Not Refined"
                        }
                }
                ...

                Line
                {
                        Default OR
                        Linetype Directed
                        Name ""
                        Description ""
                        Connections {
                                pBrakeActuator_FL.brakeForceOut =OUTPUT,
                                pBrakeActuator_AllWheel.brakeTorque_FL=DEFAULT
                                }
                }

                ...

                Line
                {
                        Default OR
                        Linetype Directed
                        Name ""
                        Description ""
                        Connections {
                                pBrakeActuator_RR.brakeForceOut =OUTPUT,
                                pBrakeActuator_AllWheel.brakeTorque_RR=DEFAULT
                                }
                }
```

Figure 5-6. An excerpt of the HiP-HOPS file (.hip) generated by the HiP-HOPS plugin of ME+ for the EAST-ADL error model of the case study system.

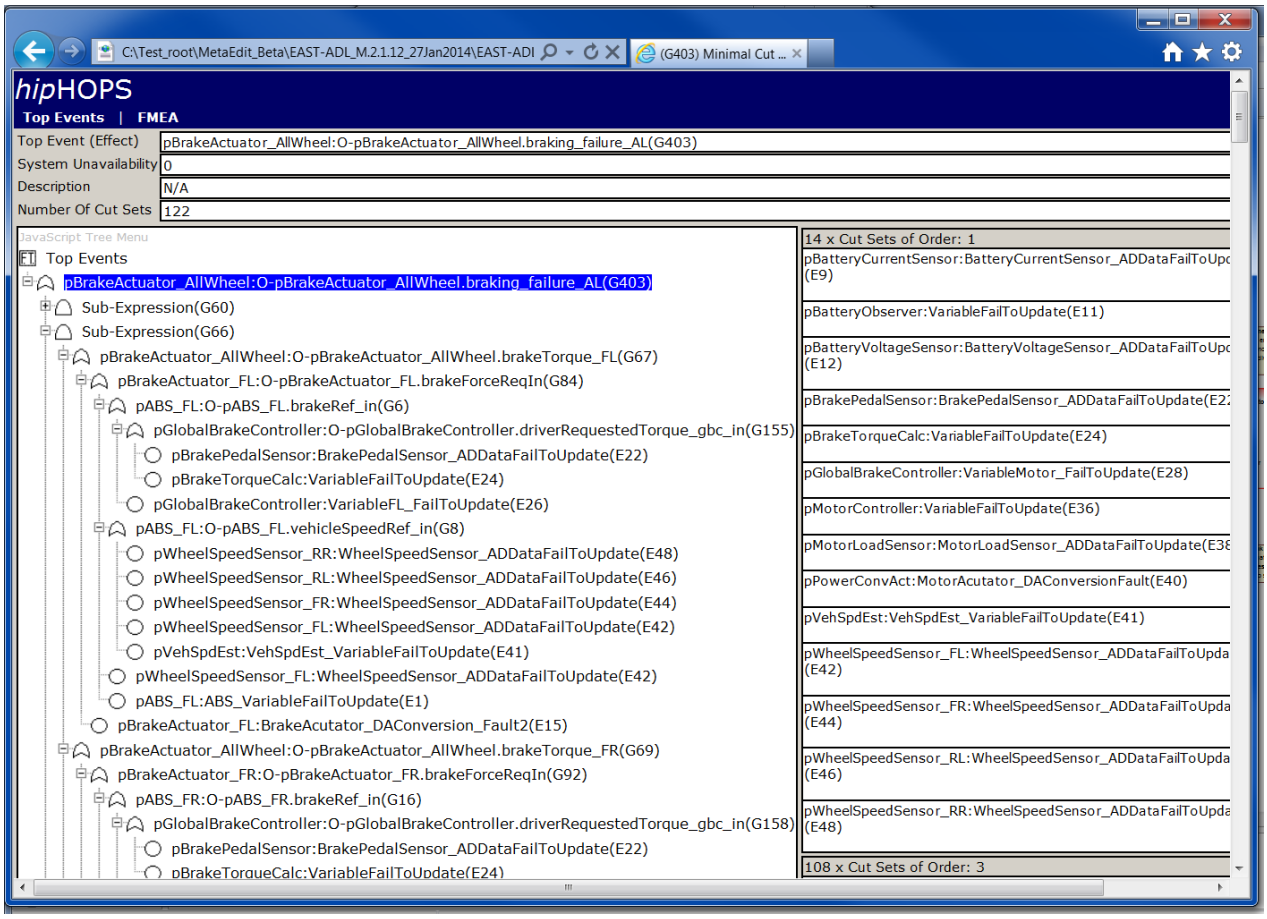
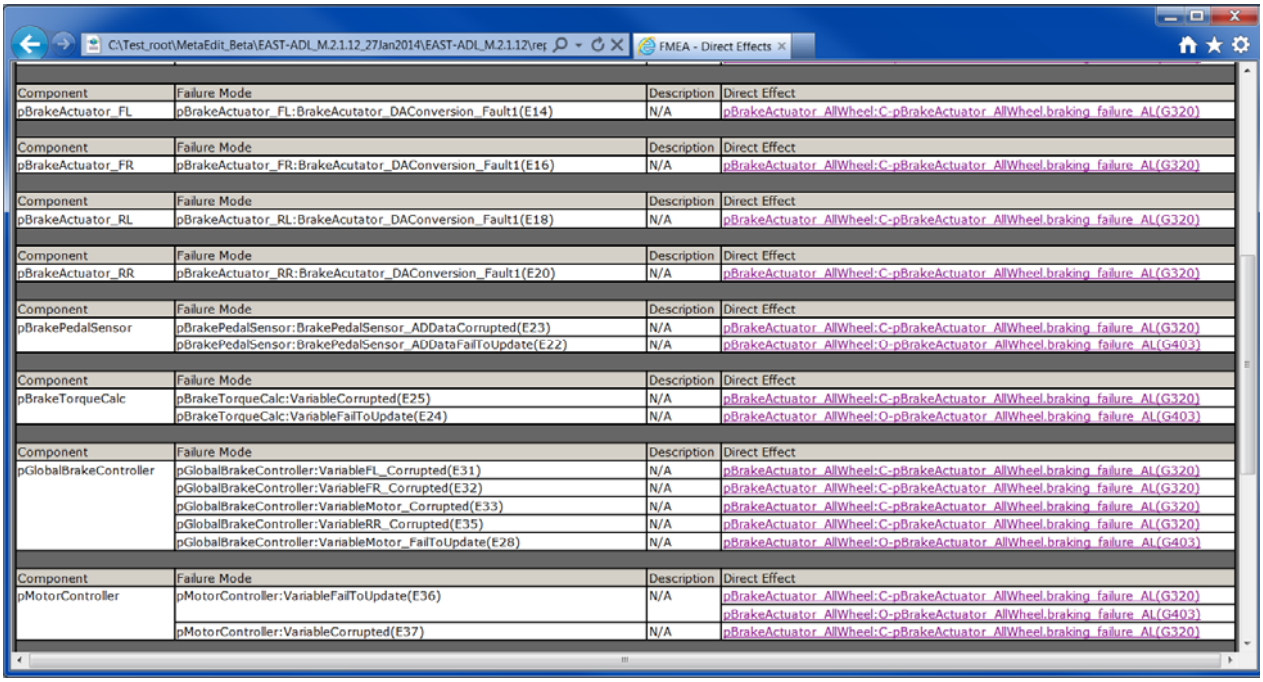


Figure 5-7. One of the fault tree and some cut sets generated by HiP-HOPS according to the EAST-ADL error specifications.



Component	Failure Mode	Description	Direct Effect
pBrakeActuator_FL	pBrakeActuator_FL:BrakeAcutator_DAConversion_Fault1(E14)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
Component	Failure Mode	Description	Direct Effect
pBrakeActuator_FR	pBrakeActuator_FR:BrakeAcutator_DAConversion_Fault1(E16)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
Component	Failure Mode	Description	Direct Effect
pBrakeActuator_RL	pBrakeActuator_RL:BrakeAcutator_DAConversion_Fault1(E18)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
Component	Failure Mode	Description	Direct Effect
pBrakeActuator_RR	pBrakeActuator_RR:BrakeAcutator_DAConversion_Fault1(E20)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
Component	Failure Mode	Description	Direct Effect
pBrakePedalSensor	pBrakePedalSensor:BrakePedalSensor_ADDataCorrupted(E23)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
	pBrakePedalSensor:BrakePedalSensor_ADDataFailToUpdate(E22)	N/A	pBrakeActuator_AllWheel:O-pBrakeActuator_AllWheel.braking_failure_AL(G403)
Component	Failure Mode	Description	Direct Effect
pBrakeTorqueCalc	pBrakeTorqueCalc:VariableCorrupted(E25)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
	pBrakeTorqueCalc:VariableFailToUpdate(E24)	N/A	pBrakeActuator_AllWheel:O-pBrakeActuator_AllWheel.braking_failure_AL(G403)
Component	Failure Mode	Description	Direct Effect
pGlobalBrakeController	pGlobalBrakeController:VariableFL_Corrupted(E31)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
	pGlobalBrakeController:VariableFR_Corrupted(E32)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
	pGlobalBrakeController:VariableMotor_Corrupted(E33)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
	pGlobalBrakeController:VariableRR_Corrupted(E35)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
	pGlobalBrakeController:VariableMotor_FailToUpdate(E28)	N/A	pBrakeActuator_AllWheel:O-pBrakeActuator_AllWheel.braking_failure_AL(G403)
Component	Failure Mode	Description	Direct Effect
pMotorController	pMotorController:VariableFailToUpdate(E36)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)
			pBrakeActuator_AllWheel:O-pBrakeActuator_AllWheel.braking_failure_AL(G403)
	pMotorController:VariableCorrupted(E37)	N/A	pBrakeActuator_AllWheel:C-pBrakeActuator_AllWheel.braking_failure_AL(G320)

Figure 5-8. FMEA table generated by HiP-HOPS according to the EAST-ADL error specifications.

Evaluation Results

Through the case studies that have been performed, it shows that the current combination of EAST-ADL/ME+ and HiP-HOPS can together provide a very effective environment for dependability modelling and safety engineering. The ME+ implementation of EAST-ADL provides a user-friendly approach to the modelling and management of safety concerns according to ISO26262. The translation to external analytical models as well as the analysis by HiPHOPS is automatic and efficient. For the baking case system model, which contains 23 error model instances, more than 75 error behaviour descriptions, and 35 links of error propagations, the generation of corresponding HiPHOPS input file by ME+ takes less than one second in a PC with Windows 7, Intel Core i7-4800MQ and 16 GB memory. No support for feeding back the results from safety analysis to the dependability model, such as in terms of derived safety requirements, is currently provided.

5.2 ASIL allocation with HiP-HOPS

Automatic ASIL allocation and decomposition allows for the efficient allocation of ASILs across a system without violating the safety requirements. It is made possible via an extension to HiP-HOPS and requires very little additional information from an EAST-ADL model other than a normal error model and the definition of hazards (together with their associated ASILs). The approach uses the existing fault propagation information to determine which system elements contribute to which hazards. If there is redundancy between system elements leading to the same hazard, then the ASIL associated with that hazard's safety goal may be decomposed across them; otherwise, each element must be allocated a sufficient ASIL to meet the safety goal for that hazard.

Procedures

Experimentation with the ASIL allocation process using EAST-ADL models is performed using the HiP-HOPS export available as an extension for the EPM modelling tool developed by TUB, enabling analysis of models created in EPM via HiP-HOPS.

Modelling for ASIL Decomposition with EPM

ASIL allocation and decomposition follows the top-down design of a safety-related system. ASILs are initially assigned to top-level safety requirements and as the system architecture and respective requirements are being refined they are allocated to subsystems and components. The rationale for determining the ASILs for each component is based on the component's contribution to risk, i.e., on the effects of its failure behaviour on top level safety requirements. In order to be aware of these failure relationships, and afterwards perform ASIL decomposition, HiP-HOPS needs to receive an architectural model that has been annotated with the failure behaviour of each component and the definition of top-level system function failures expressed through logical combinations of component output errors. Finally the tool needs to be informed of the ASILs assigned to the system top level failures at the risk assessment stage. During years 2 and 3 of MAENAD, EPM has been extended to enables all of this.

An example for annotating components with failure data, concerning system-level failure behaviour, is shown in Figure 5-9. The notation to link a system failure with a component output error incorporates information about the structure of the model designed in EPM - for example: Failure-BBW_FDA::pBrakeActuators.BrakeForceOutFL relates to the failure of the front left wheel output (BrakeForceFL) which belongs to the subsystem pBrakeActuators that in turn is within the system Functional Design Analysis view. If two or more outputs are involved in a top-level failure then their dependencies are expressed through capitalized logical operators (e.g. AND, OR). Finally a text input field allows to assign an ASIL to the system failure in the form of a number ranging between 0 to 4 (relating to ASILs QM, A, B, C and D in this order).

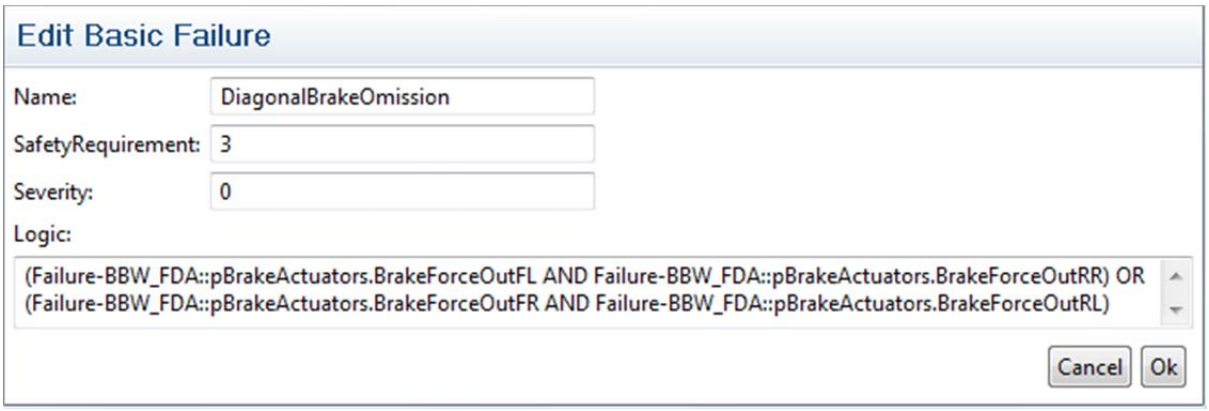
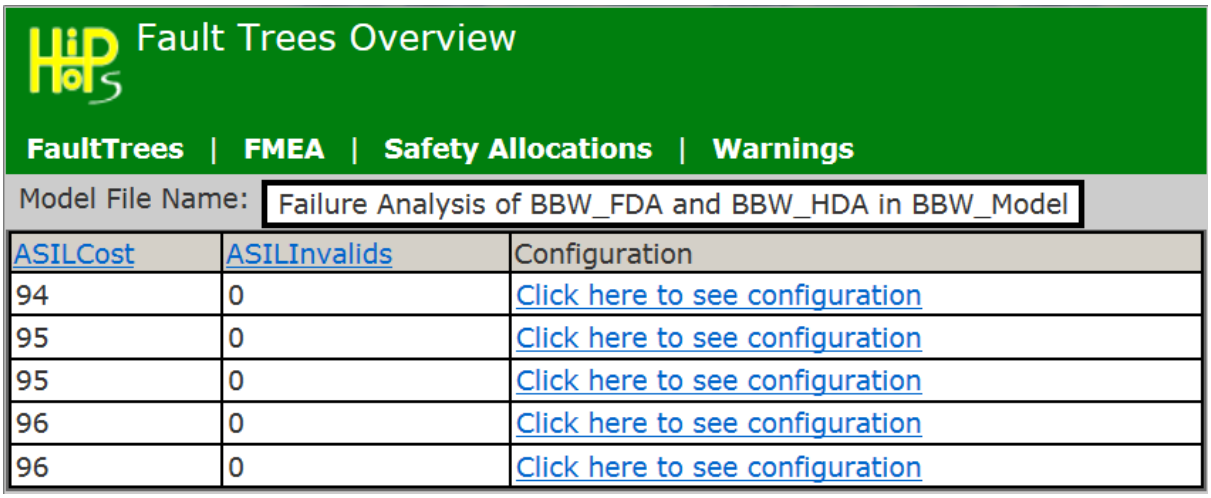


Figure 5-9. EPM - Identifying system level failures and assigning ASILs to them.

ASIL Decomposition in HiP-HOPS

HiP-HOPS reads the error model exported from EPM and automatically synthetises fault trees that have the high level failures identified for the system as top events. Analysis of these fault trees yields their minimal cut sets, i.e., the minimal combinations of components failures that result in system failures. Through this information HiP-HOPS is able to define ASIL Decomposition restrictions for the entire system; it then makes use of its optimization approach, based on a Tabu Search technique, to find the most efficient ASIL assignments. Detailed information on these processes is present in document 3.2.1.



ASILCost	ASILInvalids	Configuration
94	0	Click here to see configuration
95	0	Click here to see configuration
95	0	Click here to see configuration
96	0	Click here to see configuration
96	0	Click here to see configuration

Figure 5-10.HiP-HOPS – Best ASIL allocation solutions found by the solver in each run.

Finally, HiP-HOPS produces an HTML report with the set of best solutions obtained through multiple runs of the optimization algorithm. For the BBW system, the report produced is shown below.

The solutions can be ordered by *fitness* (*ASILCost*). In this case the optimizer tried to find the solutions that minimize the sum of ASILs assigned throughout the system architecture. It should be noted however that other approaches are supported by the technique, such as the minimization of ASIL dependant implementation costs for components.

The user can choose a solution and then inspect the ASILs assigned to individual components. The graphical interface respects the hierarchical structure of the system model created in EPM. *IndividualID* concerns the optimizer run in which the solution was found.

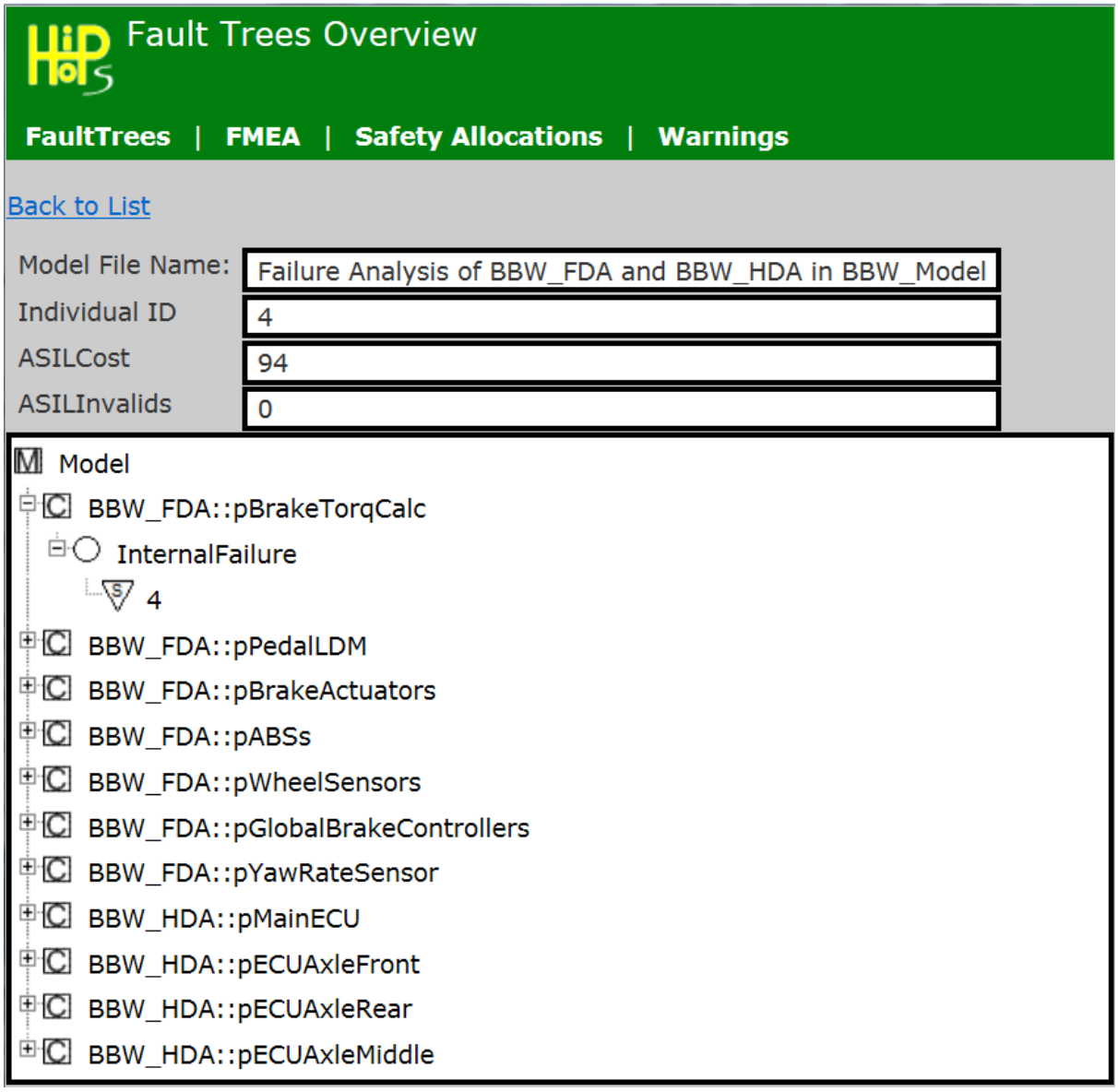


Figure 5-11.HiP-HOPS - Inspecting of one the ASIL allocation solutions found.

Evaluation Results

The original implementation of the algorithm was relatively primitive and suffered from scalability problems. For even small systems with only a handful of elements and a small number of faults, there can still be hundreds or even thousands of possible allocations, and for more realistic systems (e.g. with >50 faults), the algorithm fails to complete on standard computing hardware. Analysis of the BBW model helped to further establish where the precise limits of the original algorithm were. Apart from performance challenges, already initial experience on the basis of smaller experimental tests showed that the technique has a lot of potential.

Therefore additional work has been undertaken by UOH in year 3 of the MAENAD project to find more efficient approaches for ASIL allocation using heuristic approaches, e.g. optimisation. This work is expected to continue after the end of the MAENAD project.

The main example model being used as the main case study is a development of the Brake-by-Wire (BBW) system, initially provided by Volvo. This model has been extended by UOH (with assistance from VTEC) with experimental ASIL allocation and hazard information for the purposes of the evaluation of the decomposition technique. In our tests, 10 seconds have passed from exporting the model in EPM to the results in the HTML. However, the BBW serves the purpose of

an illustrative system, and perhaps is short in demonstrating the scalability of the technique we present here. However, we have applied the technique to a larger system. The architecture in question included 186 failures modes, leading to a total ASIL allocation solution space of 5^{186}= 1.01 x 10^{130}. Our previous exhaustive algorithms had failed to complete this test model after three months of running time; however through the Tabu Search based optimization technique, the best known solutions for this model have been found in under 5 minutes. Further information on the current algorithm can be found in D3.2.1.

5.3 Timing Analysis with Qompass

The Qompass plug-in developed originally for schedulability analysis of MARTE models, has been adapted and extended its application on EAST-ADL models. MARTE models, originally input of the Qompass tool, contained a set of tasks and scheduling information (priorities, shared resources protocols, etc) where functions are mapped into tasks and signals are mapped into messages. The Qompass tool is able, on these tasks models, to perform response-time analysis. During MAENAD, however, it was immediately clear that the most detailed information provided by EAST-ADL did not cover the task set definition. In order to make the Qompass tool relevant for EAST-ADL models we worked on two different axes: (i) extend Qompass with so-called early-stage schedulability analysis, i.e. a schedulability assessment able to detect allocation errors preventing schedulability of the system and (ii) extend Qompass with an automatic transformation of EAST-ADL model to MARTE functional models (without task set and scheduling information), containing end-to-end chains of function activations, estimated execution times for each function, end-to-end deadlines, and functional allocation to hardware nodes and buses.

Procedures

Relevant EAST-ADL information for timing analysis includes design the chains of function activations that compose a system response and subject to a deadline; the allocation of functions to nodes; the estimation of the resource demand of each function, the maximal utilization capacity of resources.

Figure 5-12 shows an excerpt of the timing information annotating the design functional graph. Activation semantics is added to that functional graph through events definition and timing constraints among these events. This gives a timed partial order of execution for functions and end-to-end timing constraints, which are needed to make timing analysis.

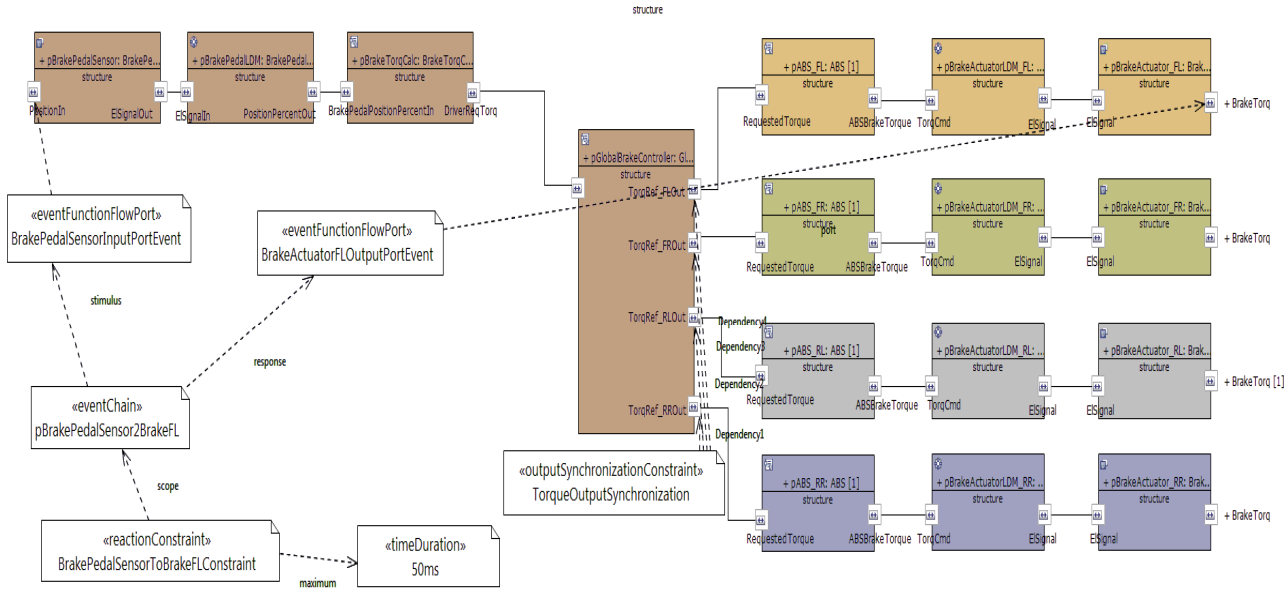


Figure 5-12.Excerpt of the timing model for the BBW

EAST-ADL relevant information is then transformed in a UML/MARTE model as shown in Figure Figure 5-13 and Figure 5-14.The MARTE model is represented by an UML Activity, corresponding to the EAST-ADL functional graph, annotated with MARTE stereotypes, containing timing and allocation information coming from the EAST-ADL model.

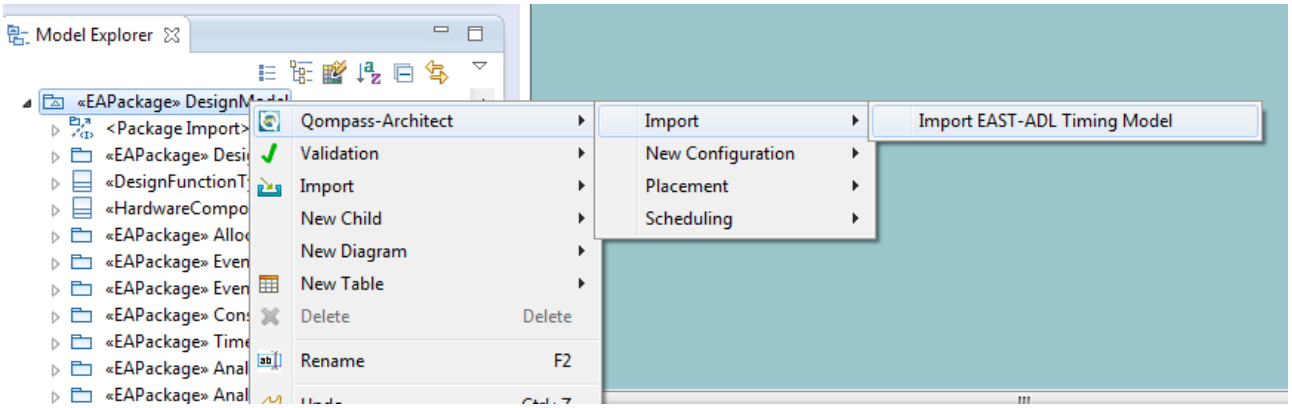


Figure 5-13. BBW EAST-ADL Timing Model Transformation.

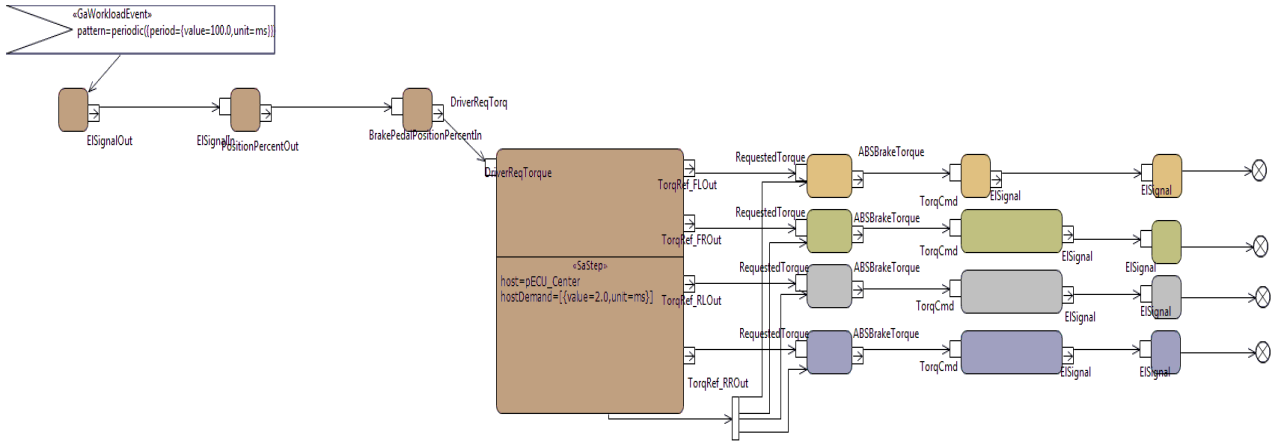


Figure 5-14. BBW MARTE model.

On this MARTE model it is now possible to launch the analysis. Various indicators for schedulability are calculated by the analysis as the node utilization Figure 5-15 and bus utilization Figure 5-16.

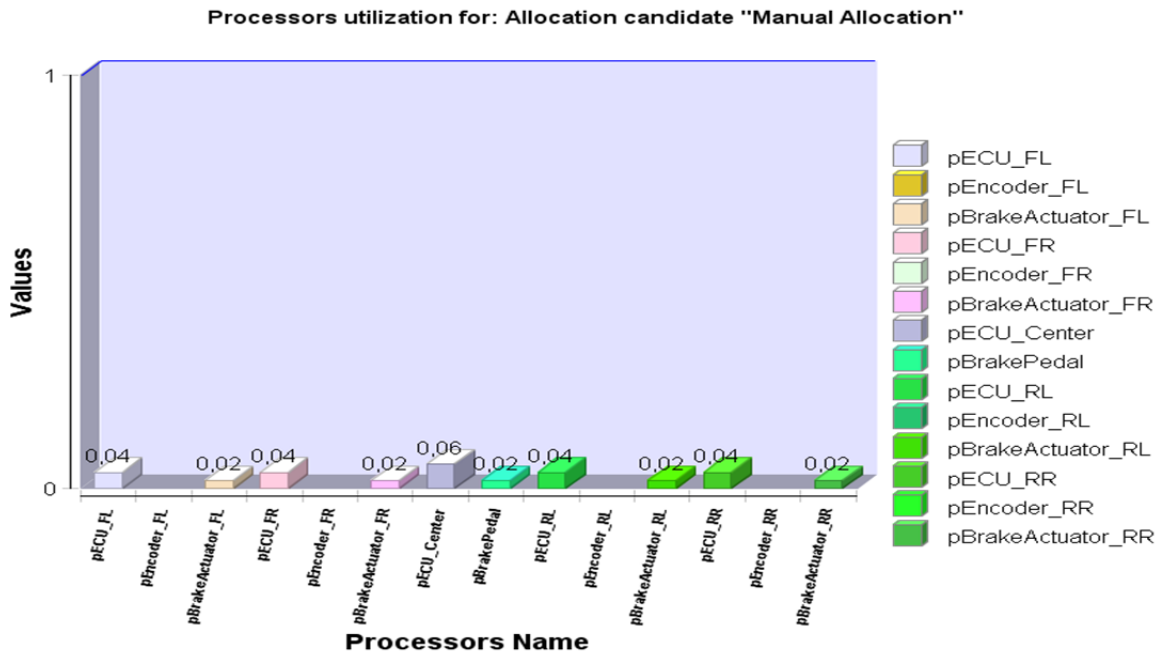


Figure 5-15. Node utilization.

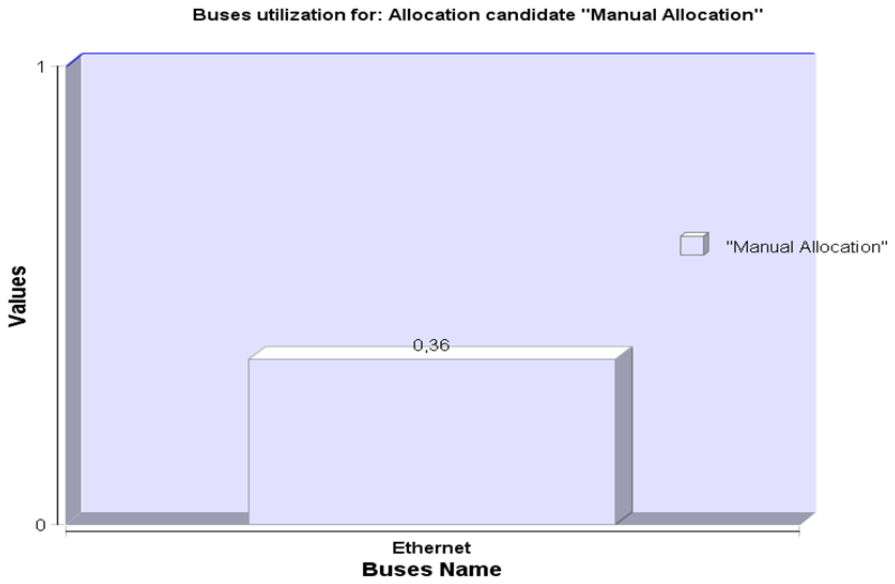


Figure 5-16. Bus utilization.

Bus and processor utilization are key indicators for system schedulability, however the respect of maximal resource utilization capacities does not imply that the system is schedulable. System schedulability can be verified only computing end-to-end latency of each path from sensors to actuators and comparing latencies against end-to-end deadlines. So-called response-time analysis is usually employed to compute end-to-end deadlines. Response time analysis, however, considered tasks chains instead of function chains. Functions and tasks, however, are not the same thing: a task is the software resource that executes functions. Response time is very sensitive to the way in which functions are partitioned into tasks. For this reason response-time analysis is usually reserved to the implementation phase. On the other hand, if the MARTE model is completed by defining as well a task set, i.e., how functions are partitioned into tasks and scheduling parameters, end-to-end latencies can be computed as shown in Figure 5-17.

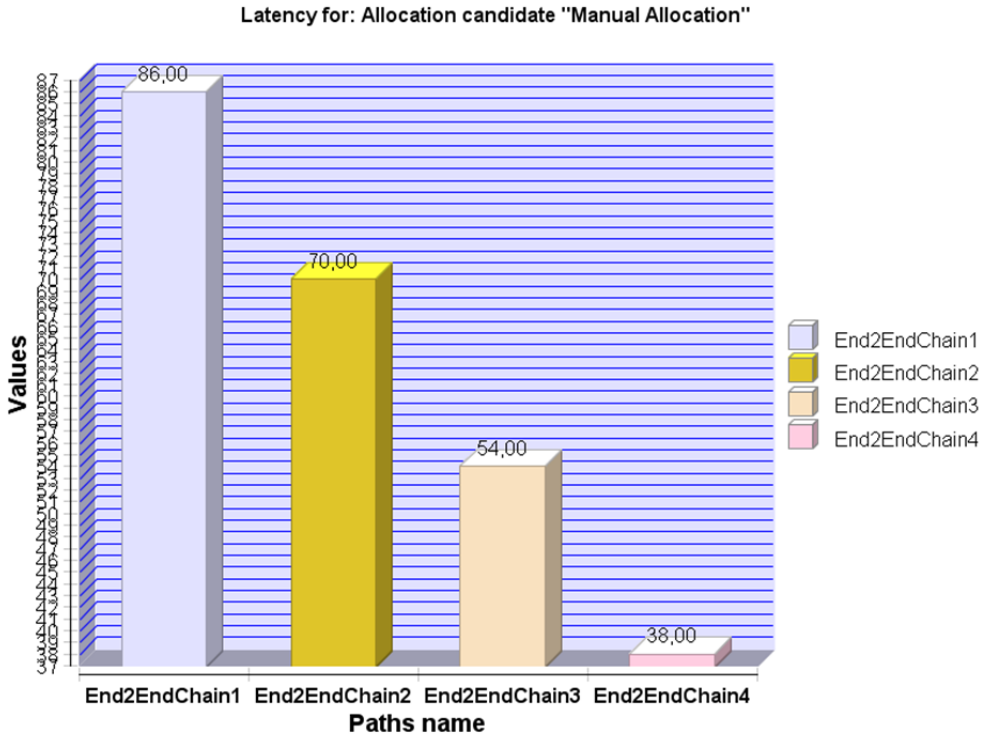


Figure 5-17. Latency.

Evaluation Results

The initial algorithm for timing analysis provided by Qompass was not able to compute response times and other schedulability-relevant parameters (node and bus utilization) on general functional graphs, i.e. graphs including forks and joins of different data flows. On the other hand this ability was required to analyze the BBW model. For this reason Y3 work focuses on supporting the BBW case study, by extending the analysis to generic graphs. Moreover, results visualization has been improved to user-friendly charts generated with Birt as shown in the previous paragraph.

5.4 Simulink Gateway

Two alternative mapping strategies are applied for the mapping from EAST-ADL functional architecture descriptions to Simulink models. The first approach is based on Matlab API, where the Simlink Gateway generates a script file (.m) consisting of Matlab commands for Simulink model creation. The second approach is based on Simulink file format, where the Simulink Gateway generates a Simulink file (.mdl) defining the expected model. Both approaches are support by the MetaEdit+ implementation of EAST-ADL by using the through the script language MERL.

Procedures

Translation of EAST-ADL model through Matlab API

Experimental model transformation with the regenerative Brake-by-Wire (BBW) system is performed using the ME+ (MetaEdit+) plugins by KTH and MetaCase. Figure 5-19 shows the Matlab code generated by the Simulink Gateway for the creation of the Simulink model iteratively according to the decomposition hierarchy in the EAST-ADL model shown in Figure 5-18. The resulted Simulink Model is shown in Figure 5-20.

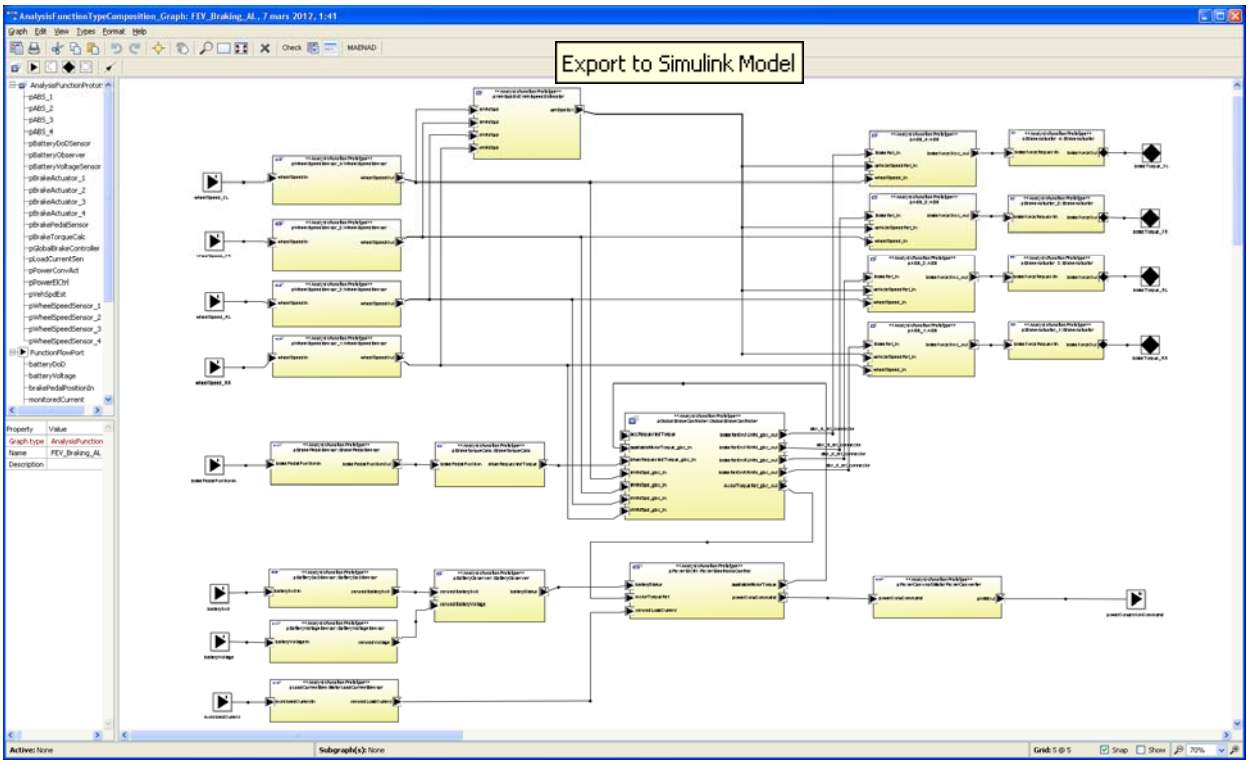


Figure 5-18. The functional architecture description of the regenerative Brake-by-Wire (BBW) system in EAST-ADL.

```
% A demonstration of the export of EAST-ADL Analysis Architecture model in ME+ to Simulink model
% Author: Mechatronics Lab, KTH
% Date: 2012-10-20

%Specify the name of the model to create
fname ='FEV_Braking_AL';

% Check if the file already exists and delete it if it does
 if exist(fname,'file') == 4
   % If it does then check whether it is open
   if bdIsLoaded(fname)
     % If it is then close it (without saving!)
       close_system(fname,0)
   end
   % delete the file
   delete([fname,'.mdl']);
end
% Create the system
new_system(fname);
% Creating the blocks and connections recursively
add_block('built-in/SubSystem','FEV_Braking_AL/pABS_1','Position',[10 1 220 70],'BackgroundColor','yellow');
add_block('built-in/outport','FEV_Braking_AL/pABS_1/brakeForceDisc_out','BackgroundColor','green');
...
add_block('built-in/inport','FEV_Braking_AL/pABS_1/wheelSpeed_in','BackgroundColor','green');

add_block('built-in/SubSystem','FEV_Braking_AL/pABS_2','Position',[10 1 220 70],'BackgroundColor','yellow');
....
add_block('built-in/inport','FEV_Braking_AL/pABS_4/BrakeRef_in','BackgroundColor','green');
```

```
add_block('built-in/inport','FEV_Braking_AL/pABS_4/vehicleSpeedRef_in','BackgroundColor','green');
add_block('built-in/inport','FEV_Braking_AL/pABS_4/wheelSpeed_in','BackgroundColor','green');
....
add_block('built-in/SubSystem','FEV_Braking_AL/pBrakeActuator_2','Position',[10 1 220 70],'BackgroundColor','yellow');
add_block('built-in/inport','FEV_Braking_AL/pBrakeActuator_2/brakeForceRequestIn','BackgroundColor','green');
add_block('built-in/outport','FEV_Braking_AL/pBrakeActuator_2/brakeForceOut','BackgroundColor','red');
....
add_block('built-in/SubSystem','FEV_Braking_AL/pGlobalBrakeController','Position',[10 1 220 70],'BackgroundColor','yellow');
add_block('built-in/inport','FEV_Braking_AL/pGlobalBrakeController/accRequestedTorque','BackgroundColor','green');
add_block('built-in/inport','FEV_Braking_AL/pGlobalBrakeController/availableMotorTorque_gbc_in','BackgroundColor','green');
....
add_line('FEV_Braking_AL','pWheelSpeedSensor_4/1','pGlobalBrakeController/4','autorouting','on');
add_line('FEV_Braking_AL','pWheelSpeedSensor_2/1','pVehSpdEst/3','autorouting','on');
add_line('FEV_Braking_AL','wheelSpeed_RL/1','pWheelSpeedSensor_2/1','autorouting','on');
.....
add_line('FEV_Braking_AL','pGlobalBrakeController/1','pABS_4/1','autorouting','on');
add_line('FEV_Braking_AL','pGlobalBrakeController/2','pABS_3/1','autorouting','on');
add_line('FEV_Braking_AL','pGlobalBrakeController/4','pABS_1/1','autorouting','on');
add_line('FEV_Braking_AL','pGlobalBrakeController/3','pABS_2/1','autorouting','on');


% Save the model
save_system(fname);
```

Figure 5-19. An excerpt of the Matlab code generated by the Simulink Gateway in ME+ for the creation of the regenerative Brake-by-Wire (BBW) system in Simulink.

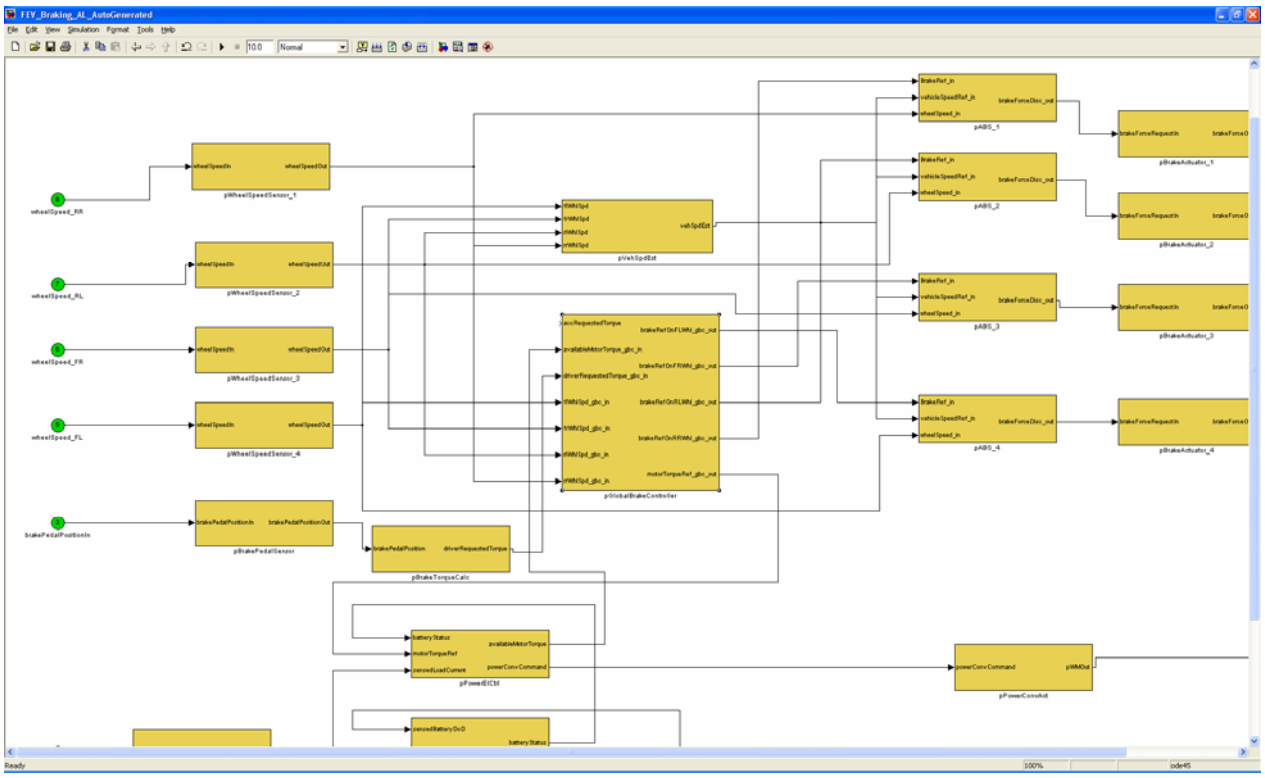


Figure 5-20. The Simulink model generated by the Simulink Gateway in ME+ for the EAST-ADL functional architecture description of the regenerative Brake-by-Wire (BBW) system.

To support the transformation of behaviour specification, this Simulink plugin also searches the behaviour constraint annotated for each functional block and thereby creates the Matlab API instructions for the setting up of corresponding StateFlow model. This is shown in Figure 5-21.

```
sflib %Open Stateflow Library first
open('Advanced_EEPowerSupply');

%%%%%%%%%%%%%%%%%%%%%%%%add stateflow chart within the structural target according to the protoype name%%%%%%%%%%%%%%%%%%%%%%%%
add_block('sflib/Chart','Advanced_EEPowerSupply/battery1/Batt_Primary');
set_param('Advanced_EEPowerSupply/battery1/Batt_Primary','Position', [120 10 350 150]);

rt = sfroot;
chartHandle=rt.find('-isa','Stateflow.Chart','Name','Batt_Primary');

%%%%%%%%%%add states%%%%%%%%%%%%%%%%%%%%%%

charged= Stateflow.State(chartHandle);
charged.Name = 'charged';
charged.Position = [60 100 90 35];

charging= Stateflow.State(chartHandle);
charging.Name = 'charging';
charging.Position = [60 200 90 35];

depleted= Stateflow.State(chartHandle);
depleted.Name = 'depleted';
depleted.Position = [60 300 90 35];

%%%%%%%%%%add transitions%%%%%%%%%%%%%%%%%%

trans_b1=Stateflow.Transition(chartHandle);
trans_b1.Source=charging;
trans_b1.Destination=charged;
trans_b1.SourceOClock = 6;
trans_b1.DestinationOClock = 1;
trans_b1.LabelString='[EnergyStatus >= 10&&Battery_Ctrl == 1]'

trans_b2=Stateflow.Transition(chartHandle);
trans_b2.Source=charging;
trans_b2.Destination=depleted;
trans_b2.SourceOClock = 7;
trans_b2.DestinationOClock = 2;
trans_b2.LabelString='[EnergyStatus < 10&&Battery_Ctrl == 1]'

trans_b3=Stateflow.Transition(chartHandle);
trans_b3.Source=charged;
trans_b3.Destination=charged;
trans_b3.SourceOClock = 8;
trans_b3.DestinationOClock = 3;
trans_b3.LabelString='{Call_EnergyStatusUpdate(SoC, mode)}'

trans_b4=Stateflow.Transition(chartHandle);
trans_b4.Source=charged;
trans_b4.Destination=depleted;
trans_b4.SourceOClock = 9;
trans_b4.DestinationOClock = 4;
trans_b4.LabelString='[EnergyStatus < 10]'

trans_b5=Stateflow.Transition(chartHandle);
trans_b5.Source=depleted;
trans_b5.Destination=charged;
trans_b5.SourceOClock = 10;
trans_b5.DestinationOClock = 5;
trans_b5.LabelString='[EnergyStatus >= 10]'

trans_b6=Stateflow.Transition(chartHandle);
trans_b6.Source=depleted;
trans_b6.Destination=depleted;
trans_b6.SourceOClock = 11;
trans_b6.DestinationOClock = 6;
trans_b6.LabelString='{Call_EnergyStatusUpdate(SoC, mode)}'

trans_b7=Stateflow.Transition(chartHandle);
trans_b7.Source=depleted;
trans_b7.Destination=charging;
trans_b7.SourceOClock = 12;
trans_b7.DestinationOClock = 7;
trans_b7.LabelString='[Battery_Ctrl == 0]'

trans_b8=Stateflow.Transition(chartHandle);
trans_b8.Source=charged;
trans_b8.Destination=charging;
trans_b8.SourceOClock = 13;
trans_b8.DestinationOClock = 8;
trans_b8.LabelString='[Battery_Ctrl == 0]'

trans_b9=Stateflow.Transition(chartHandle);
trans_b9.Source=charging;
trans_b9.Destination=charging;
trans_b9.SourceOClock = 14;
trans_b9.DestinationOClock = 9;
trans_b9.LabelString='{Call_EnergyStatusUpdate(SoC, mode)}'
DefTran=Stateflow.Transition(chartHandle);
DefTran.Destination = charging;
DefTranA.DestinationOClock = 0;
```

Figure 5-21. An excerpt of the Matlab code generated by the Simulink Gateway in ME+ for the creation of StateFlow model for the battery power management behaviors.

The Plugin have been exercised on the “Propulsion and power distribution” subsystem. Currently, the prototype plug-in is available on Metaedit+ development environment and can be activated through a button located on the main command bar of the “Functional Design Architecture” graph viewer.

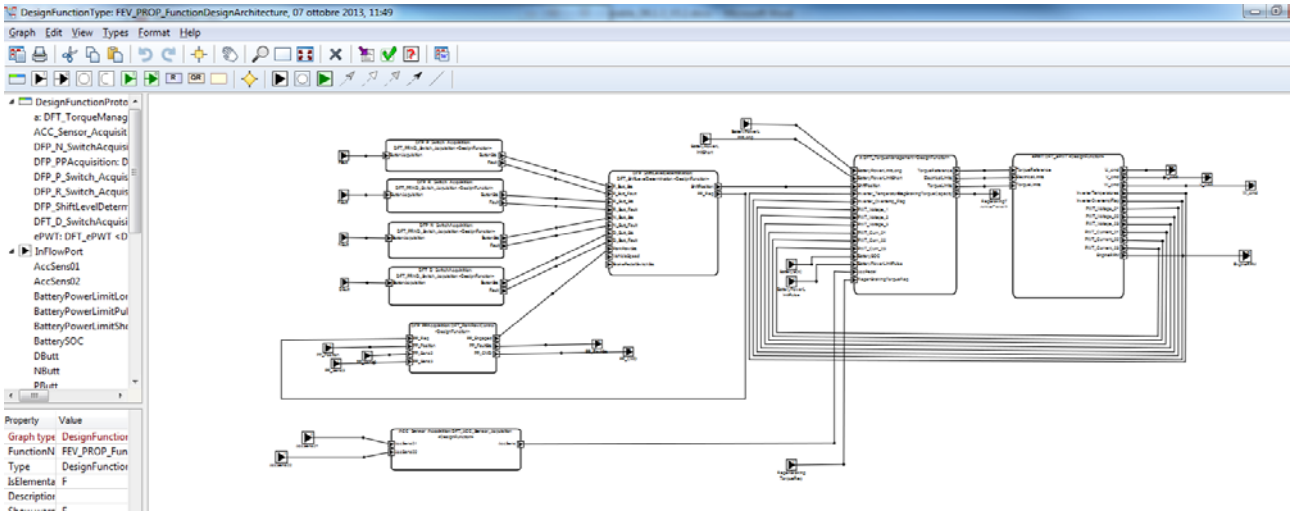


Figure 5-22. The functional architecture description of the Propulsion and power distribution system in EAST-ADL.

The tool generate a Simulink model for each EAST-ADL Design Function. The model hierarchy is respected; each Design Function instance of an EAST_ADL diagram that is part of another Design Function is translated in the target Simulink model using the “model reference” technique.

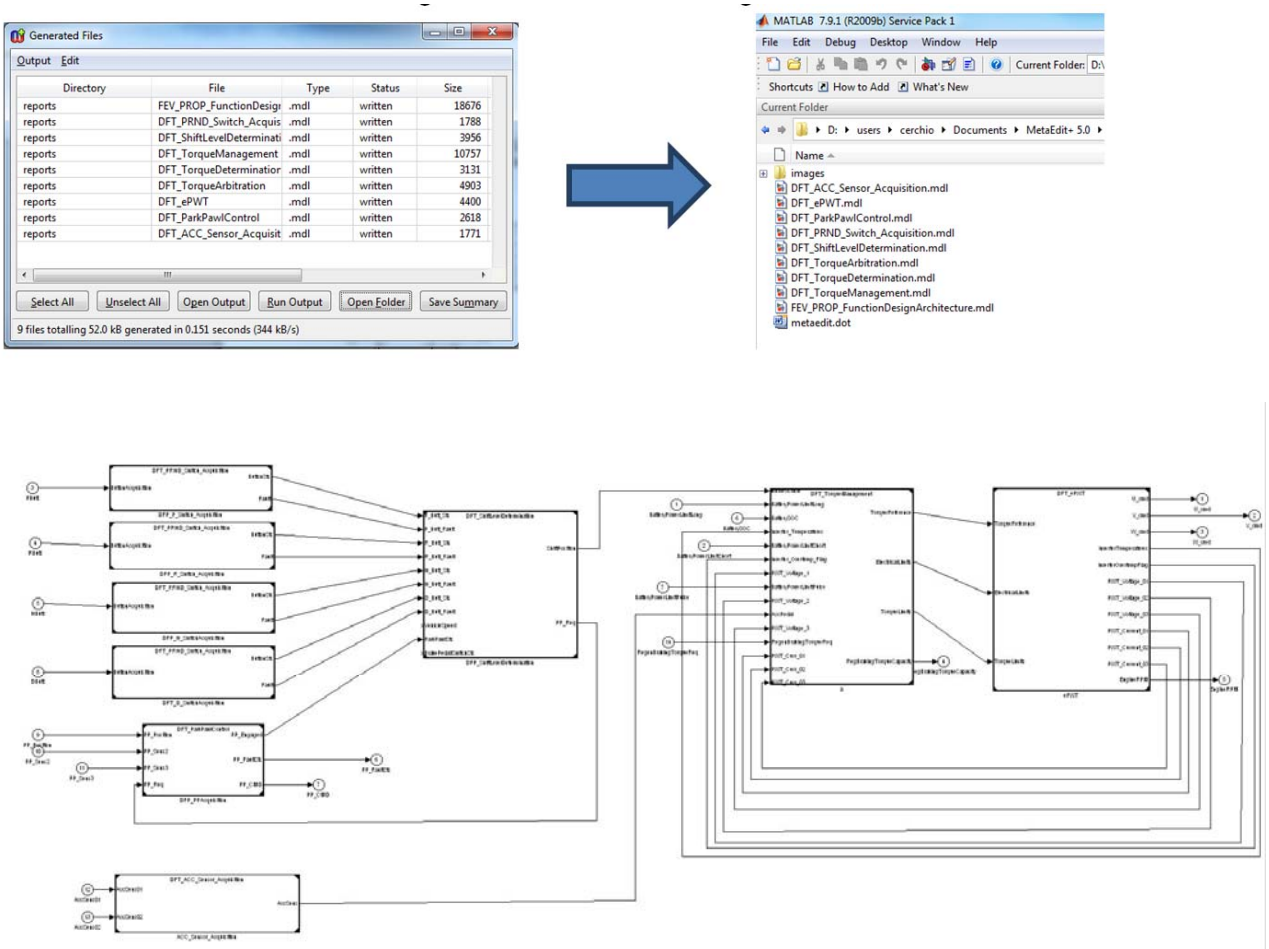


Figure 5-23. The corresponding Simulink model generated by the Simulink Gateway in ME+.

The following code snapshot illustrate the Simulink model created by the Simulink Gateway opened with a text editor.

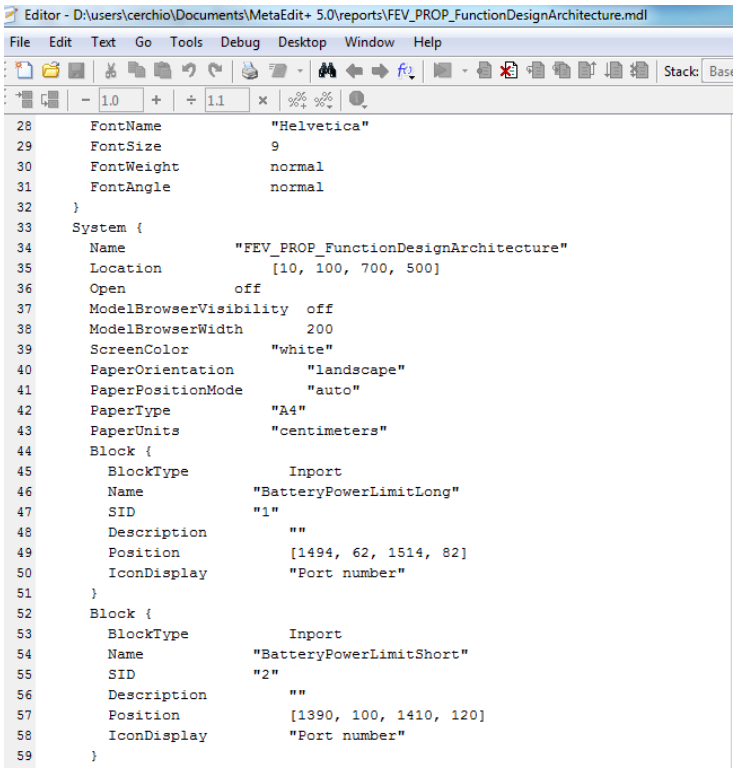


Figure 5-24. A snapshot of the Simulink code generated by the Simulink Gateway in ME+ for the creation of the "Propulsion and power distribution" subsystem.

Evaluation Results

The case studies show that with ME+ both transformation approaches are very efficient in regard to the creation of Simulink models using EAST-ADL architecture models as the inputs. Both approaches provide a full coverage of the data, communications and compositions. For the BBW case, each of these two transformation plugins takes less than one second in a PC with Windows 7, Intel Core i7-4800MQ and 16 GB memory. The Matlab API based approach also demonstrates that EAST-ADL models can also be transformed to executable Simulink models through the language support for behavior constraints annotations. Currently, this requires that the SM behaviors to be captured with the EAST-ADL temporal constraints and lower level computations to be expressed in C or Matlab Script. The generated Simulink model can be used for design verification/validation through simulation. Integration with the Matlab Design Verifier is currently out of scope due to the license issue.

5.5 Functional Mock-up Unit Import

The Functional Mock-up Interface (FMI) [6] is a standard that many simulation tools use for interchange of models, and co-simulation. The Functional Mock-up Interface defines the input and output variables of each unit and also the data types of these variables. This is similar to EAST-ADL AnalysisFunctionTypes, so a transformation is possible, simplifying creation of architectural models from simulation models.

Procedures

An input file representing a fictitious environment model was defined in Modelica and exported to an FMU including an FMI file see Figure 5-25.

```
<?xml version="1.0" encoding="UTF-8"?>
<fmiModelDescription
  fmiVersion="1.0"
  modelName="EnvironmentModelica"
  modelIdentifier="EnvironmentModelica"
  guid="{9db12878-b7ee-450c-9efa-7eb345bb0a9b}"
  generationTool="OpenModelica Compiler 1.9.0 beta4 (r15030)"
  generationDateAndTime="2013-08-15T07:23:49Z"
  variableNamingConvention="structured"
  numberOfContinuousStates="0"
  numberOfEventIndicators="0">
  <ModelVariables>
  <ScalarVariable
    name="WheelSpeed_RR"
    valueReference="0"
    variability="continuous"
    causality="output"
    alias="noAlias">
    <Real />
  </ScalarVariable>
  <ScalarVariable
    name="WheelSpeed_RL"
    valueReference="1"
    variability="continuous"
    causality="output"
    alias="noAlias">
    <Real />
  </ScalarVariable>
  <ScalarVariable
    name="WheelSpeed_FR"
    valueReference="2"
    variability="continuous"
    causality="output"
    alias="noAlias">
    <Real />
  </ScalarVariable>
  <ScalarVariable
    name="WheelSpeed_FL"
    valueReference="3"
    variability="continuous"
    causality="output"
    alias="noAlias">
    <Real />
  </ScalarVariable>
  <ScalarVariable
    name="BrakeForce_FL"
    valueReference="4"
    variability="continuous"
    causality="input"
    alias="noAlias">
    <Real />
  </ScalarVariable>
  <ScalarVariable
    name="BrakeForce_FR"
    valueReference="5"
    variability="continuous"
    causality="input"
    alias="noAlias">
    <Real />
  </ScalarVariable>
  <ScalarVariable
    name="BrakeForce_RR"
    valueReference="6"
    variability="continuous"
    causality="input"
    alias="noAlias">
    <Real />
  </ScalarVariable>
  <ScalarVariable
    name="BrakeForce_RL"
    valueReference="7"
    variability="continuous"
    causality="input"
```

```
    alias="noAlias">
    <Real />
  </ScalarVariable>
  </ModelVariables>
</fmiModelDescription>
```

Figure 5-25. FMI file generated by OpenModelica tool.

The FMI plugin is invoked from the package where the imported function type and its port datatypes should be located. This is illustrated in Figure 5-26 and Figure 5-27.

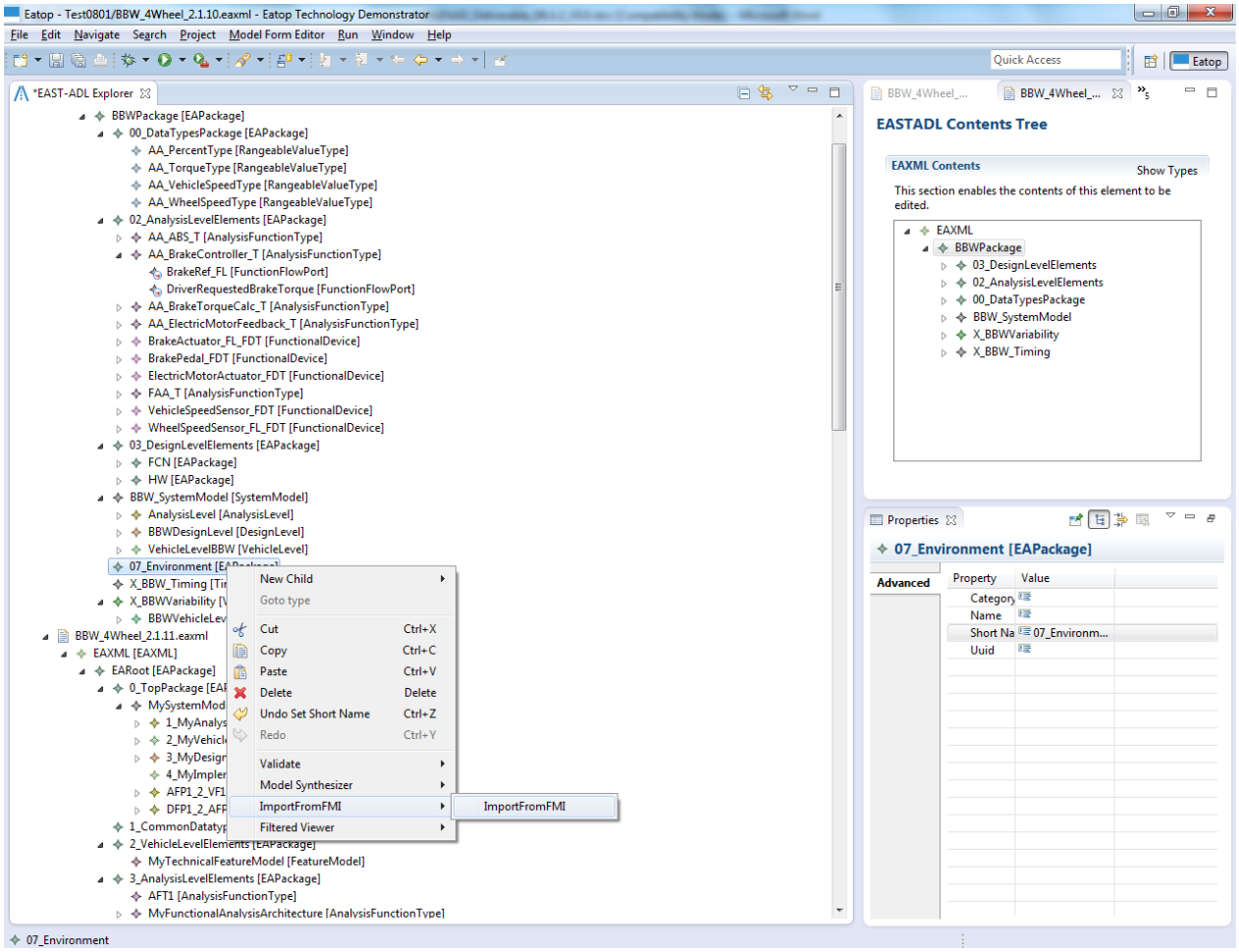


Figure 5-26. FMI Import plugin is invoked by right-clicking the target package in the tree browser.

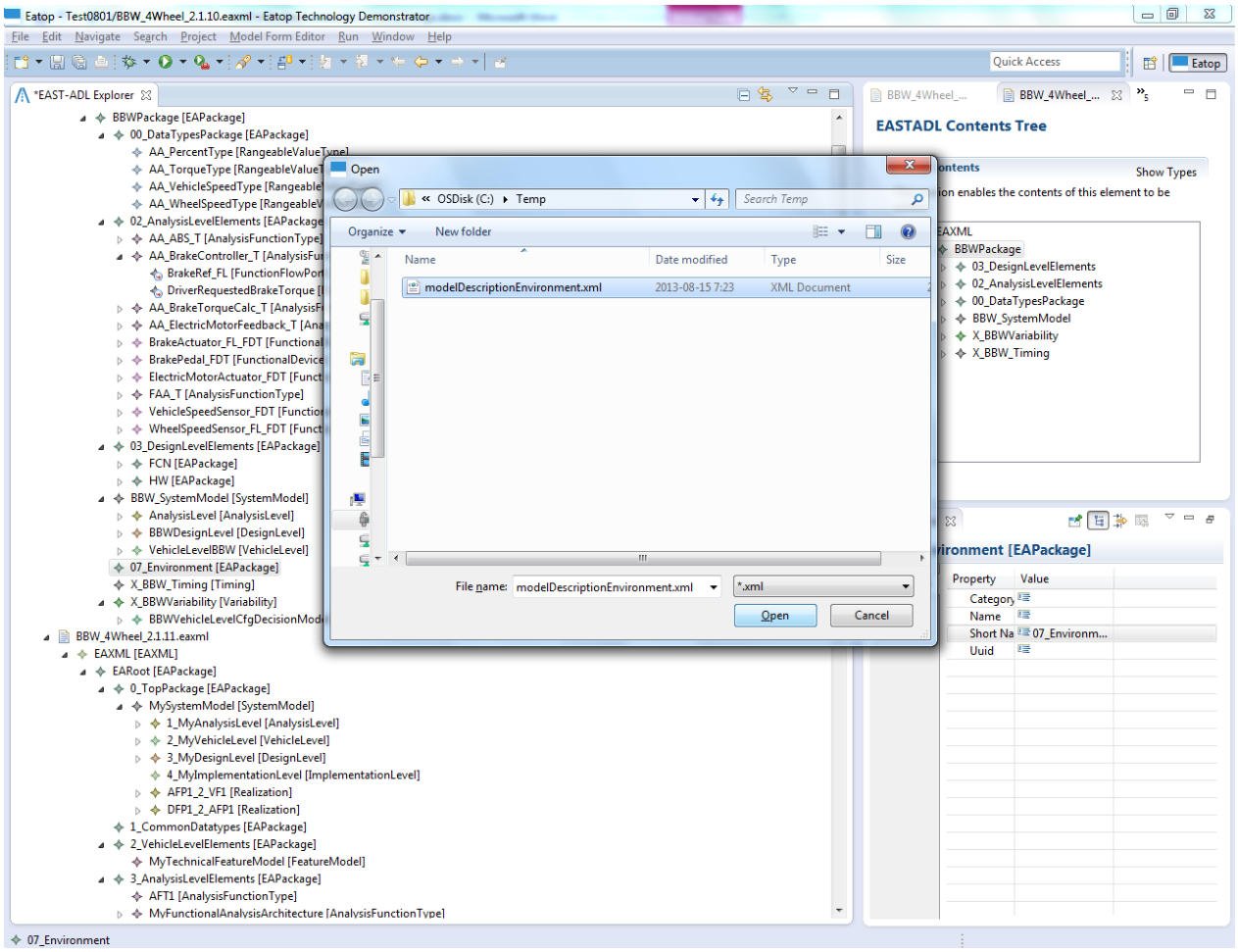


Figure 5-27. The FMI definition is an xml file and selected from the plugin dialogue.

Figure 5-28 shows how the model is updated with new datatypes and a new Analysis Function type in the selected package.

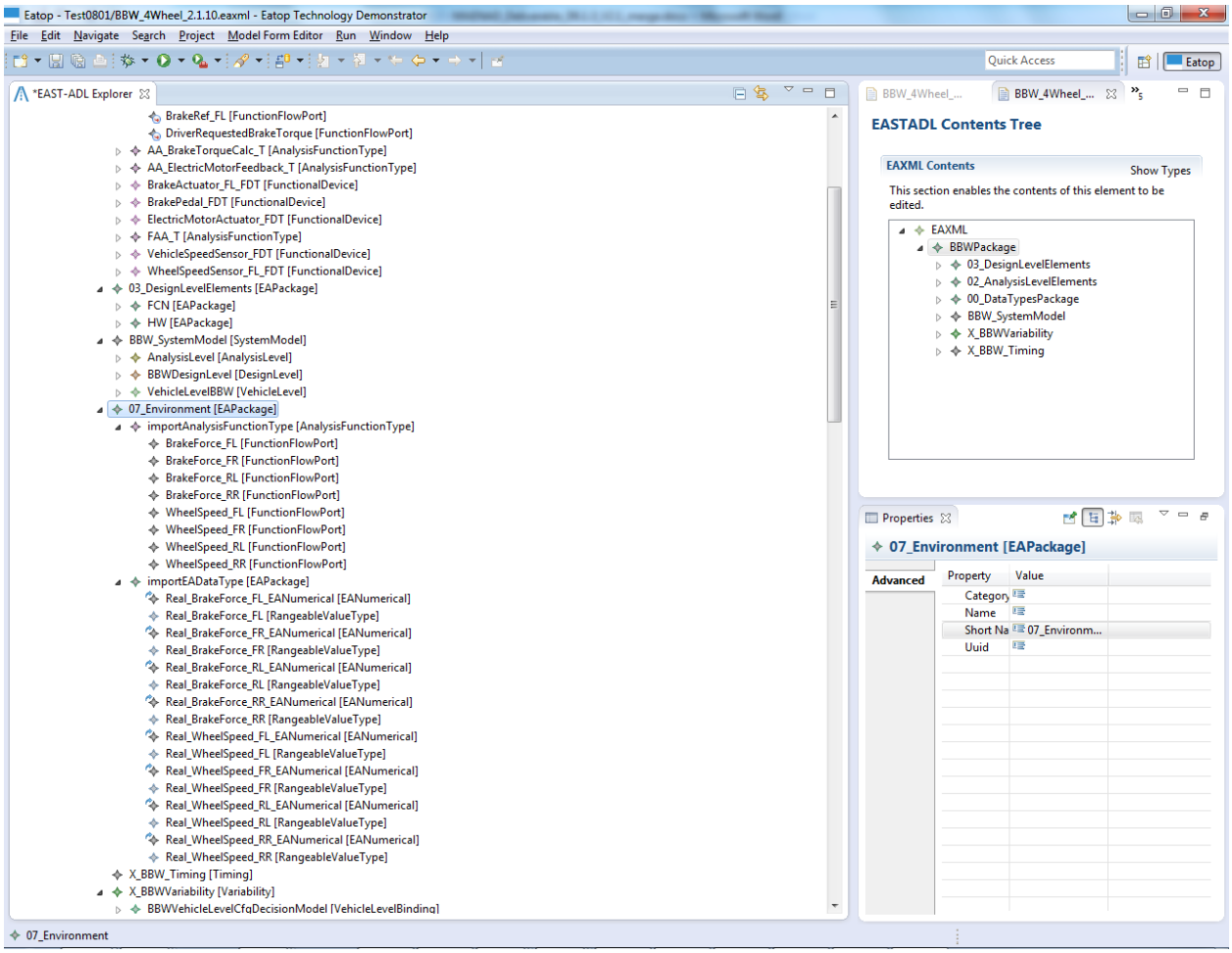


Figure 5-28. The resulting AnalysisFile with its ports and datatypes are inserted in the selected package.

Evaluation Results

The FMU:s defined and exported in an external tool could be successfully imported to the EAST-ADL model.

5.6 EATOP Analyzer

The Analyzer plugin for EATOP is a tool for evaluating additive GenericConstraint annotations. It respects modes and compares required values with calculated values for the respective kind of GenericConstraint.

Procedures

An EAST-ADL model with GenericConstraints annotating the architecture (functional or hardware) is prepared in EATOP or another EAST-ADL modelling tool with EAXML export capability. A requirement is defined together with a Refine relation from a GenericConstraint formalizing the required sum, e.g. the total power. Each contributor to the power consumption is annotated with a corresponding GenericConstraint. In case the annotations are relevant only in specific modes, this can be included by adding mode references from the constraints.

Analysis is initiated by right-clicking the requirement model containing the requirements to be checked, see Figure 5-29.

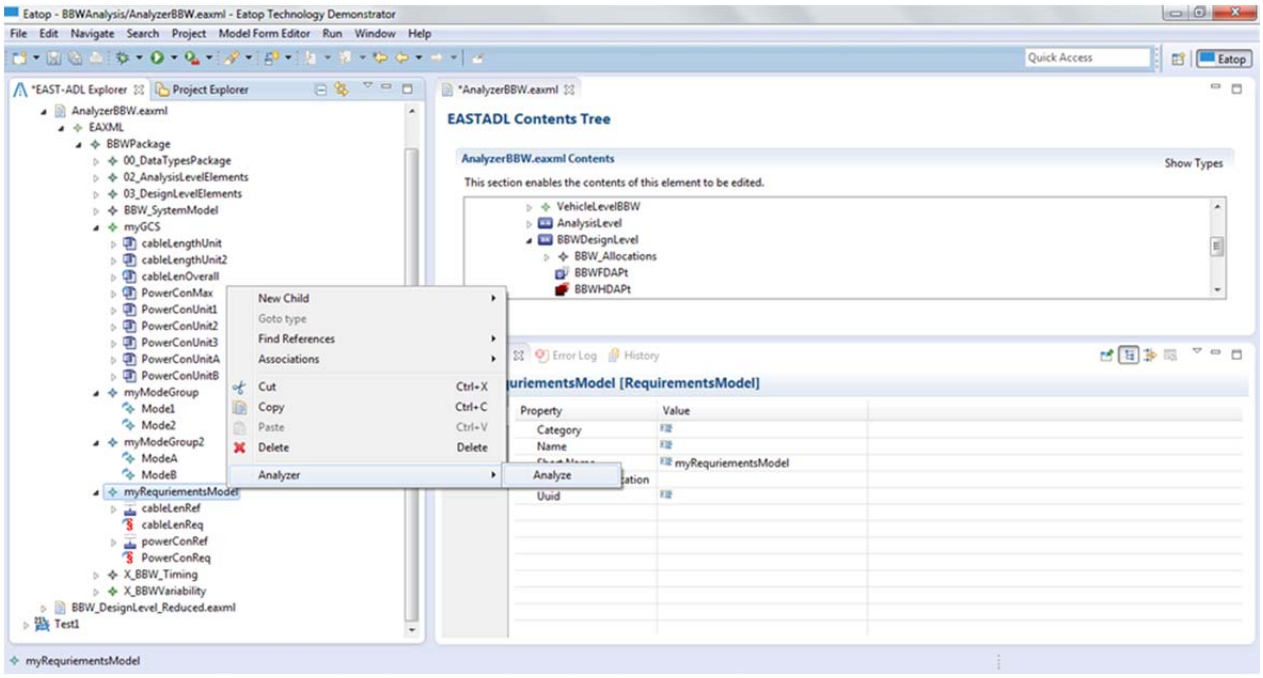


Figure 5-29. Analyzer is invoked by right-clicking the Requirement or Requirement Model.

On invoking the analyzer, the user chooses which requirements to evaluate. Following that, the applicable modes (if any) are shown and the user needs to decide for each mode group which mode is to be analyzed, see Figure 5-30.

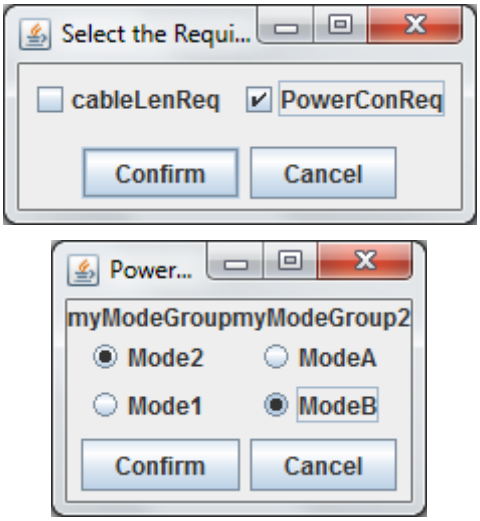


Figure 5-30. The user need to choose which requirements to check and for which modes.

After configuring the analysis, the sum of the properties is calculated. This is done by traversing the architecture hierarchy to which the requirement is related, and adding the appropriate value (appropriate mode and GenericConstraintKind)) for each prototype (as defined by its type). The computed sum and the required sum is shown after analysis, see Figure 5-31.

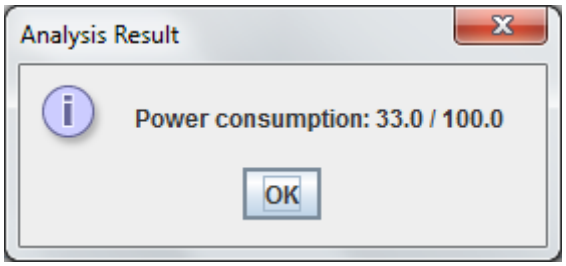


Figure 5-31. Analysis result and required value is shown after analysis.

Evaluation Results

The Analyzer plugin, and the annotation capability of EAST-ADL have been evaluated using some FEV-relevant requirements. It was concluded that the cocncept for annotating properties and defining requirements is appropriate also for mode-based properties.

5.7 OptiPAL

This section describes experimentation with the MAENAD optimization prototype, also known as OptiPAL, that has been built as an extension to the EPM modeling tool. EPM and OptiPAL are both based on the Eclipse platform, a major Java application framework that gained popularity in automotive industry in recent years. Based on example modeling and experimentation with OptiPAL, the MAENAD optimization architecture was refined and extended.

The basic goal of the MAENAD optimization architecture is to use a standard EAST-ADL model for design space exploration in order to find the “best” system architecture among a set of architecture alternatives defined by the modeler. This is based on the assumption that the set of viable architecture alternatives is so large that a manual assessment of the individual candidates is not feasible and therefore an automatic optimization approach is required. Even with automatic evaluation available, an extensive evaluation of the set of candidates in not possible and therefore more sophisticated optimization algorithms are required. As an additional challenge, individual candidates are not only evaluated with respect to a single objective (e.g. cost) but several, often conflicting, goals (e.g. cost vs. dependability vs. power consumption), where the is not single “best” solution but the solution has to be chosen by making a trade-off between the conflicting goals. In the literature, such optimization problems are referred to as multi-objective optimization.

The MAENAD optimization architecture an the OptiPAL prototype tool implementing this reference architecture follow this approach:

1. a standard EAST-ADL model is being used as the sole input for optimization.
2. in that model, the standard EAST-ADL variability modeling concepts are employed to define the design space, i.e. the set of architecture alternatives or *candidates* among which to find the “best” solution during optimization. The model is to be defined such that each valid configuration of the model’s variability provides such an architecture candidate.
3. the model is complemented with meta-information that will later allow analysis tools to assess the quality of an individual architecture alternative with respect to a particular design goal. In principle, there are two cases for defining this meta information:
    a. standard EAST-ADL modeling concepts are applied (e.g. EAST-ADL timing and behavior specifications or GenericConstraints and Requirements), or
    b. EAST-ADL user attributes are applied if the meta-information is not covered by the native EAST-ADL modeling entities.

4. configuration parameters are provided to OptiPAL to define various aspects of the optimization scenario: which objectives to include, which external analysis tools to use for evaluation of each objective, how many optimization cycles to perform, etc.
5. OptiPAL will then automatically create a fully resolved model (i.e. no remaining variability in the model) from the variant-rich base model from step 1 and send it, for each objective, to an external analysis tool in order to obtain a fitness value for this resolved model. This will provide a fitness value for each objective and for each resolved model, i.e. architecture alternative. This step constitutes the main optimization cycle.
6. While repeating the previous step over and over, the “best” – or to be more precise: the pareto-optimal – architecture candidates are retained and presented to the user as the main result of the optimization (including the fitness values).

This brief overview of the MAENAD optimization architecture shall only provide enough information to allow an understanding of the remainder of this section. In particular, the important challenge of distinguishing design space variability from product line variability is not covered in further detail here. More details on this and other aspects of the MAENAD optimization architecture on a conceptual level can be found in D3.2.1, details on the internal architecture of the prototype implementation OptiPAL are given in D5.2.1.

Procedures

The OptiPAL case study was based on the same Brake-by-Wire (BBW) system employed in many other evaluation tasks in MAENAD. The initial model was provided by VTEC. As a first step it was transferred to EPM (also by VTEC) and handed over to UoH where the model was complemented by a failure propagation model, providing meta-information for dependability as a first optimization objective. Then, the variability in the model was refined and extended by TUB to (1) have a sufficiently complex design space and (2) include non-trivial product line variability to allow an evaluation of this aspect of the MAENAD optimization architecture. Finally, as a joint effort by TUB, UoH, VTEC and other MAENAD partners, the model was complemented with meta-information for two more optimization objectives: cost and power consumption. This information was provided in the form of user attributes because EAST-ADL does not cover these aspects natively.

In summary, we have met information in the BBW model to support the following optimization objectives:

1. cost,
2. dependability,
3. power consumption.

As explained above, for each optimization objective an analysis tool must be in place in order to actually analyze the individual optimization candidates to obtain a fitness. In case of dependability we have HipHops, in the other two cases we have used a simplified, generic resource consumption analysis tool.

The following figure shows the top-level feature model of the BBW EAST-ADL model we have used for experimenting with OptiPAL.

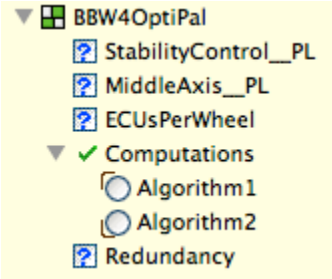


Figure 5-32. the top-level feature model of the BBW EAST-ADL model for experimenting with OptiPAL.

Features with a rectangular check box (with a question mark inside) are optional features, those with a round check box are mandatory alternative features. A suffix of "__PL" indicates that the corresponding feature represents product line variability, whereas all other features represent design space variability. The above top-level feature model is the entry point for OptiPAL to explore the design space, i.e. to generate fully-resolved configurations of the model that represent architecture alternatives to be evaluated during optimization. The feature model defines 8 design space configurations and 4 product line configurations leading to a total of 32 overall system configurations.

As usual with EAST-ADL's variability modeling, the variability defined in the above feature model is then linked to variability on the architecture level, i.e. optional function prototypes, ports, connectors in the FAA, FDA, HDA. The following examples are intended to given an impression of how this is defined, not a complete description of the BBW model.

The following figure shows how two mandatory alternative algorithms for the GlobalBrakeController are defined:

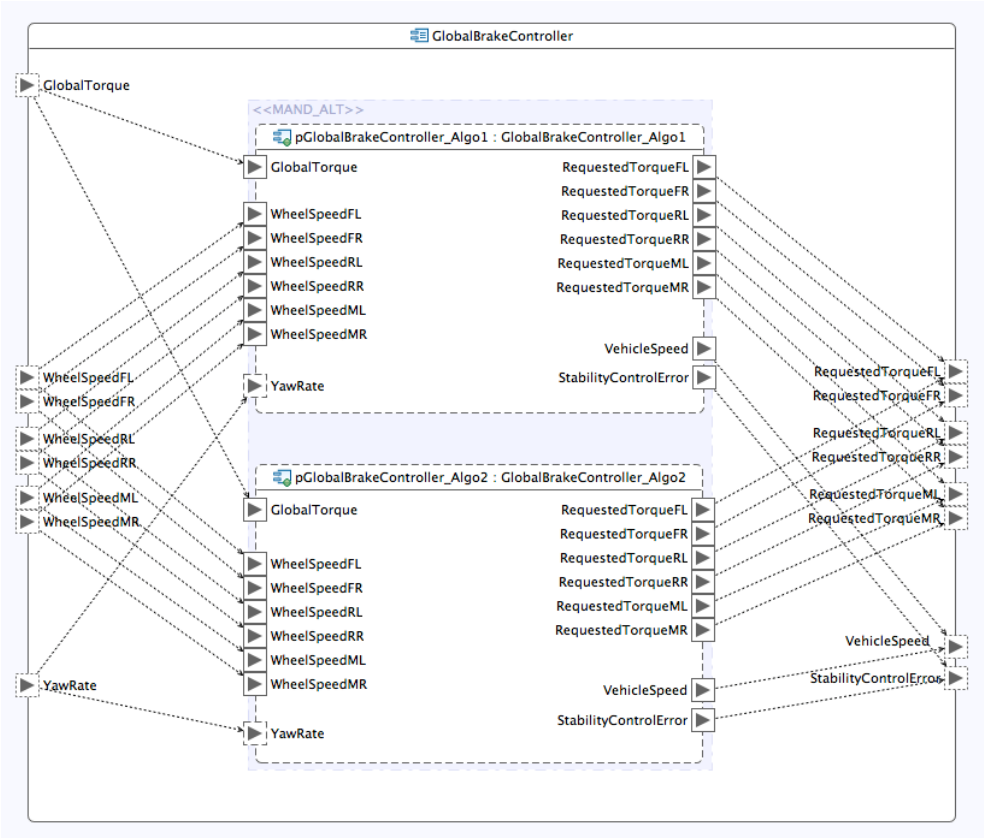


Figure 5-33. Defining two mandatory alternative algorithms for the GlobalBrakeController.

Dashed border of the two function prototypes in the center indicate that they are optional. Their selection / deselection is driven by the feature "Algorithm1" and "Algorithm2" of the top-level feature model presented above.

The next figure shows another very common pattern of design space variability in the BBW model for optimization, namely redundancy. We have a “WheelSensor” that might be implemented alone or might come with a redundant “WheelSensorSecondary” to increase dependability:

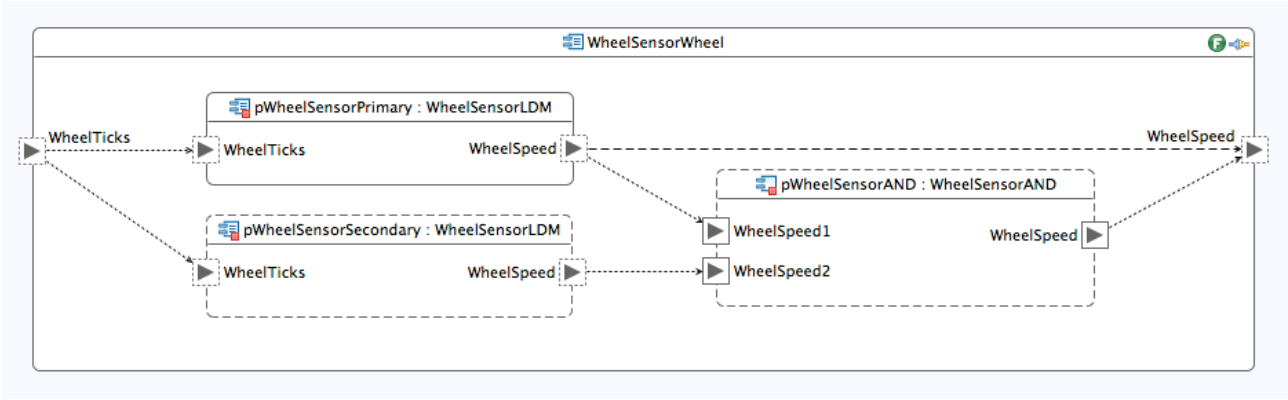
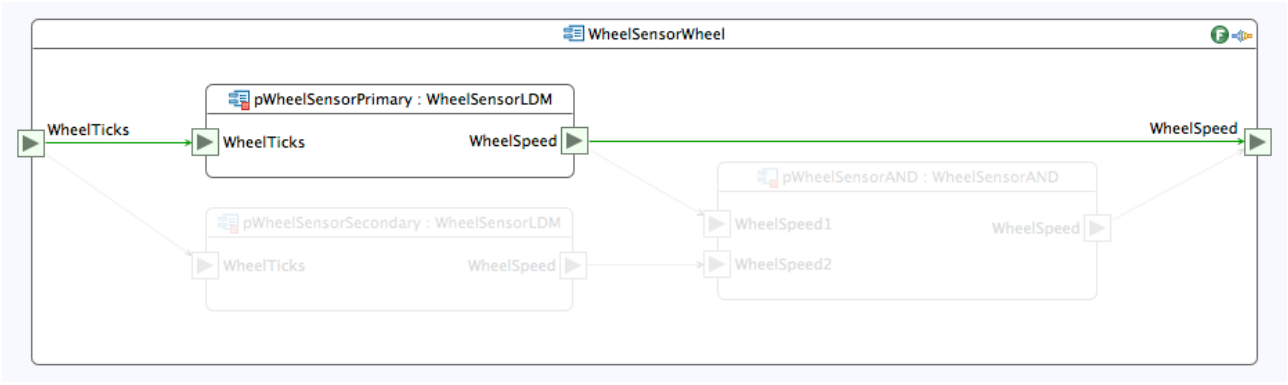


Figure 5-34. Defining a redundant design.

Here are the two valid configurations for the above figure (selection of “pWheelSensorSecondary” and “pWheelSensorAND” is driven by feature “Redundancy” in the top-level feature model):



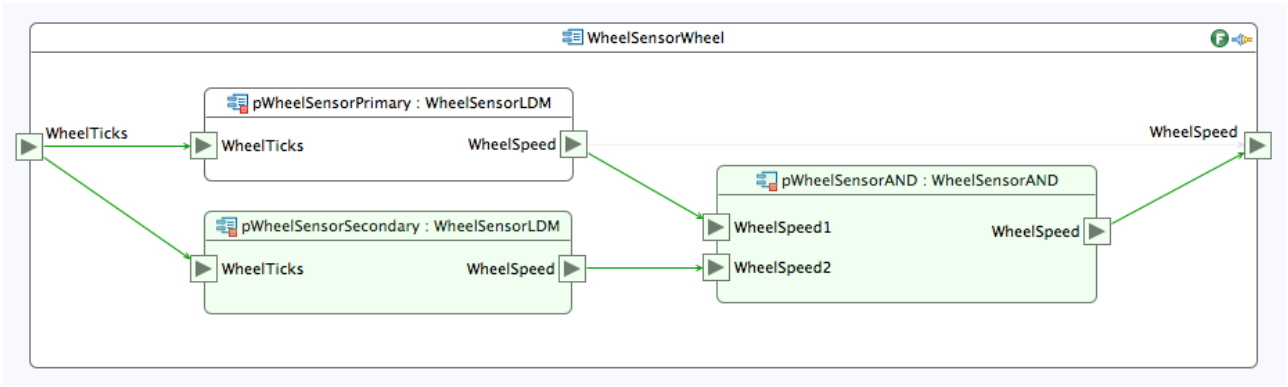


Figure 5-35. The two valid configurations for a redundant design.

As a final example, here is the HAD of the BBW model (the combination of the two features “MiddleAxis__PL” and “ECUsPerWheel” determine the selection / deselection of the optional ECUs for the front, middle, and rear axle):

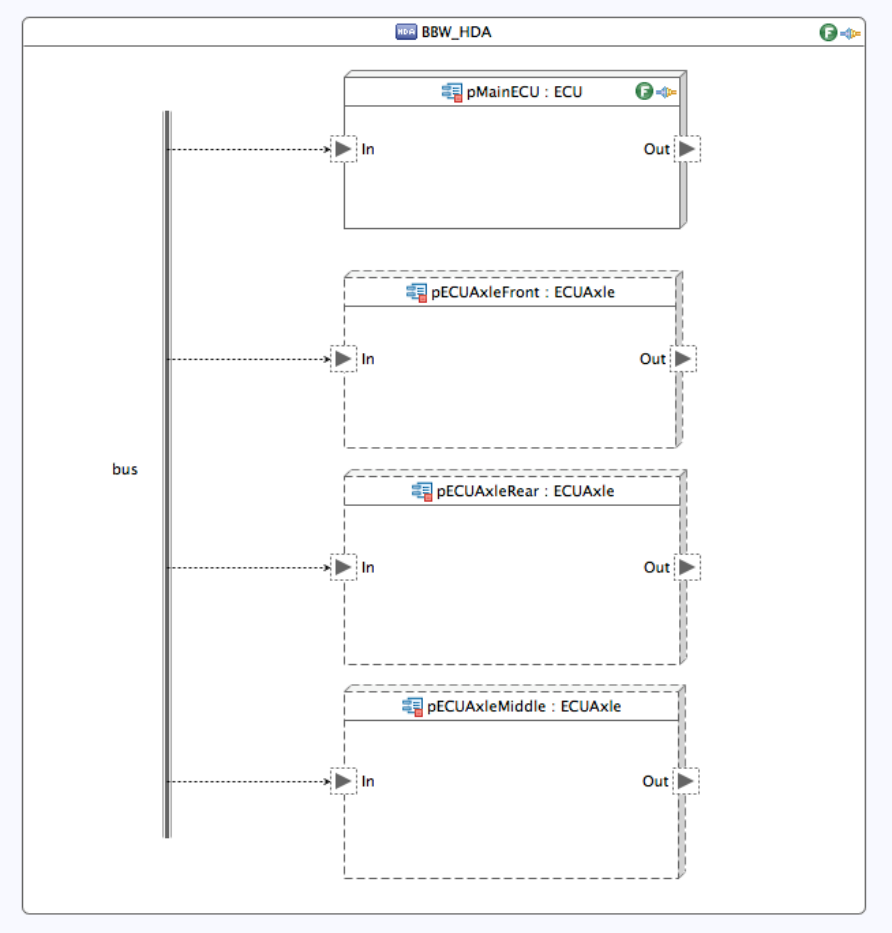


Figure 5-36. The hardware architecture of the BBW model.

As stated above, the top-level feature model defines 32 overall system configurations. Our BBW model contains much more complex variability on the architecture level, but for the purpose of experimenting with OptiPAL we have reduced the configurations to 32 in order to be able to manually review the validity of the optimization result, i.e. the output produced by OptiPAL.

Overall, the BBW model contains 60 function prototypes, 55 variation points and 43 features (including features in public features models on the architecture level!).

Evaluation Results

The BBW model with its 32 overall configurations (8 candidates with 4 representatives each) produced a pareto front with 2 pareto-optimal candidates that dominated the other 6 candidates. The following figures show the graphical representation of the pareto-optimal and dominated candidates as produced by OptiPAL.

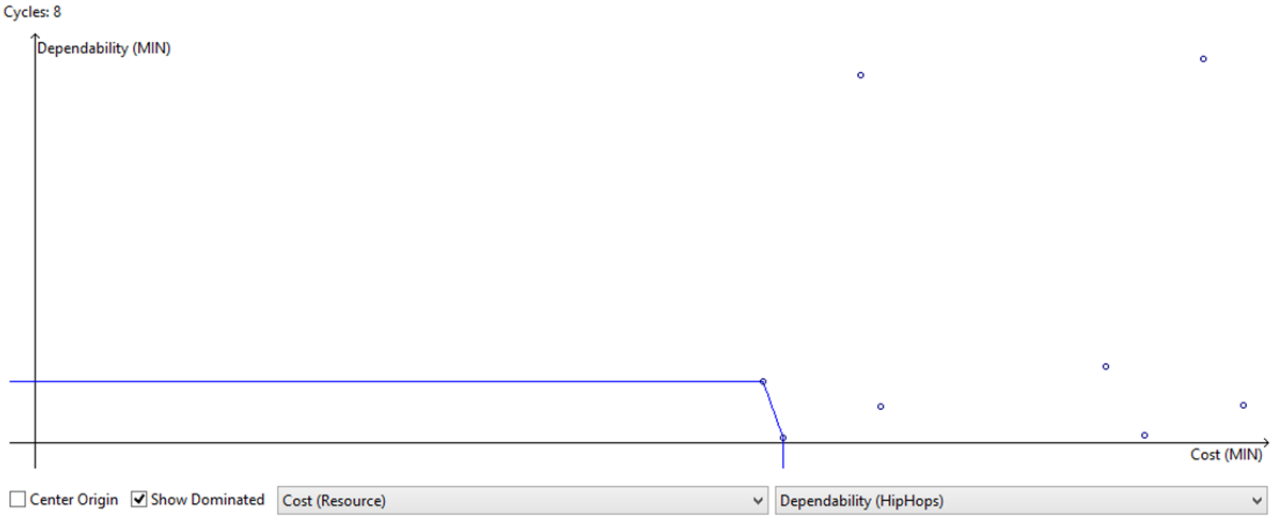


Figure 5-37. Cost vs. Dependability.

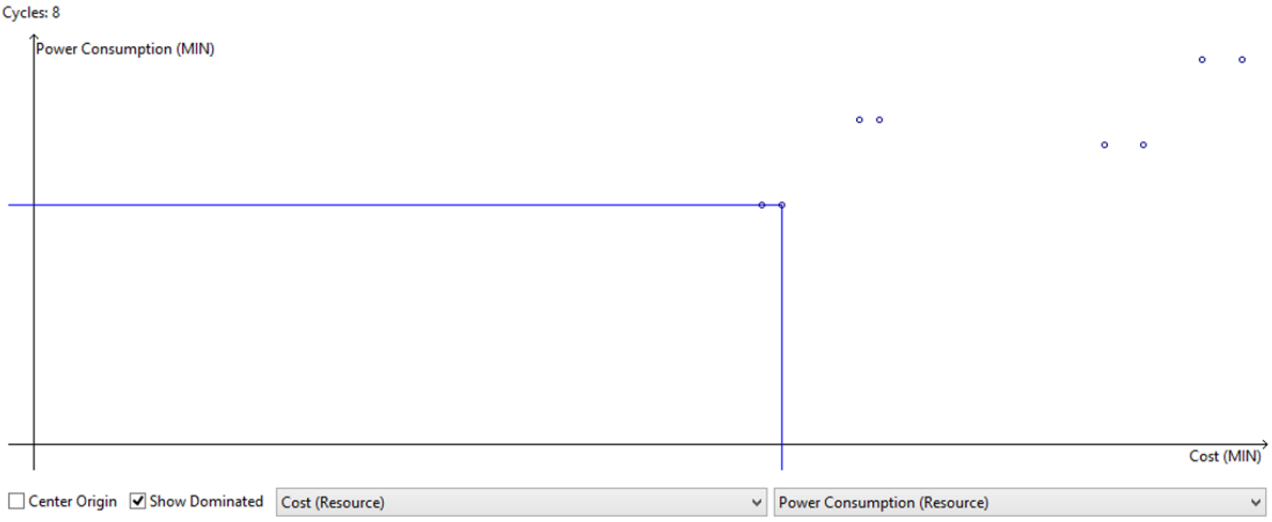


Figure 5-38. Cost vs. Power Consumption.

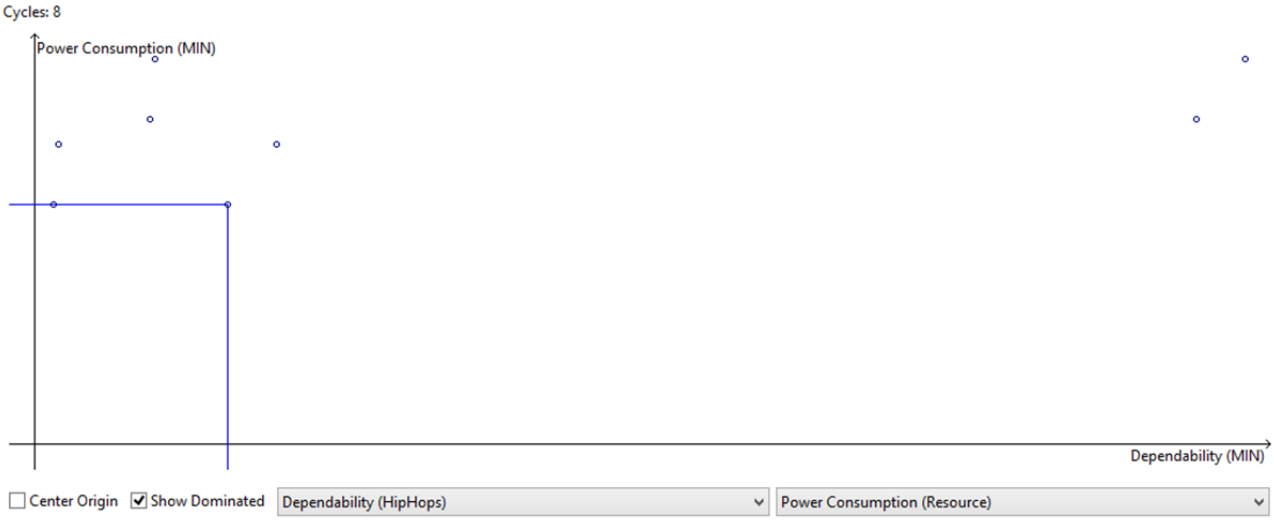


Figure 5-39. Dependability vs. Power Consumption.

The evaluation of OptiPal based on the BBW model and also other experimental models has provided insights in several areas, as detailed in the remained of this section.

Overall Usability and Stability

The overall usability and stability was satisfying for a prototype intended for experimentation purposes. More important for MAENAD than the actual usability and stability of the tool, however, is the observation that all remaining usability and stability issues are purely related to implementation limitations and not related to the basic idea behind the MAENAD optimization architecture or its concepts and not related to EAST-ADL.

Linking External Analysis Tools to OptiPAL via Generic Interface

One main aim of the MAENAD optimization architecture is to integrate several different optimization objectives while allowing for a heterogeneous tool environment in which each objective is assessed by a different external analysis tool. The approach of OptiPAL to provide a single consistent interface for all analysis means (referred to as AnalysisWrapper in D3.2.1 and D5.2.1) was successful during our experiments. The use of the Eclipse extension point mechanism even allows to add additional analysis tool without the need of changing or recompiling OptiPAL.

However, if the external analysis tool exists outside the Eclipse/Java environment, a significant run-time overhead has to be expected for exporting the model for each architecture candidate to be assessed and for importing the analysis results into OptiPAL (but see conclusion below). In our experiments we covered this case for the dependability objective, because HipHops is a native Windows application.

Feasibility of EAST-ADL's Variability Modeling Concepts

During the MAENAD optimization case study, the EAST-ADL language elements for modeling variability provided an effective means to define all variability encountered in our model. On the one hand, this shows that EAST-ADL has effective language constructs for modeling non-trivial system variability in general, on the other hand (and more important to MAENAD) this shows that the EAST-ADL variability modeling mechanisms can also applied to *design space* variability, i.e. for defining the set of architecture alternatives to be considered during an architecture optimization (definition of the optimization space). Before the MAENAD project, this was not an intended purpose of EAST-ADL variability modeling, but our experiments indicate that it can be used for that.

Relation of Design Space and Product Line Variability Within a Single EAST-ADL Model

When modeling the optimization space with variability (design space variability) in addition to the product line variability for which EAST-ADL variability modeling was intended, originally, then the two forms of variability have to be distinguished and properly handled during optimization. This was one of the major research challenges to be addressed by the MAENAD optimization architecture. The experiments show that design space and product line variability can be integrated within a single system model and that a multi-objective optimization can be performed based on such an integrated model. The notions of candidates and representatives and a flexible configuration of representative selection and the candidate fitness combination function (explained in D3.2.1, D5.2.1) allow the user to influence and precisely define how to deal with product line variability in the concrete optimization scenario at hand. Remaining challenges for further research include finding more strategies for representative selection and obtaining more insight into the strengths and weaknesses of each strategy, identifying more strategies of candidate fitness combination and again obtaining a better understanding of the pros and cons of each strategy. The main result of MAENAD in the area of optimization, however, the notion of candidates vs. representatives, representative selection and candidate fitness combination, and the overall MAENAD optimization reference architecture are well consolidated.

Conclusion

In summary, the only major obstacle we have encountered during this case study that was not solved during the project is the overall performance and run time. The run time of a full evaluation of all 32 configurations with three objectives was around 27 seconds (Windows 8 desk top computer with an i5-4430 3GHz processor, 8GB RAM and an SSD mass storage drive). In part,

this can be explained by the prototypical nature of the OptiPAL implementation and therefore we can assume that an implementation intended for immediate application in industrial practice would not suffer from this limitation. On the other hand, this is caused by the fact that we aim to integrate different external analysis tools, requiring a transformation of the single source model into the analysis tool's proprietary input format for each individual architecture alternative to be evaluated. As long as analysis tools are focused on a single or only a few aspects of a system (as is industrial practice today), true multi-objective optimization as intended in MAENAD will always have to accept this factor and its impact on run-time. But our experiments during MAENAD show that the approach can still be applied for meaningful, non-trivial use cases from industrial practice with an acceptable runtime (even with our current, non-optimized prototypical implementation).

5.8 Model Exchange

Procedures

During the second year of the project, support for models exchange between different tools has been developed by the tool supplier partners of the MAENAD project.

The developed features provide means to transform a model described in a proprietary XML format to a "standardized" XML format (EAXML) and re-import it back.

Each tool supplier adopts a different solution, based on intermediate transformation steps, and provide user friendly interface to hide the complexities of the process.

EAST-ADL XML file can be generated natively by the tools starting from any Hardware Architecture model. The generated EAXML file reflects the original structure preserving hierarchy, package structure and the type-prototype pattern.

One solution adopted involves the use of XLST language and the related processor.

The XSLT is a XML-based language used for the transformation of XML documents.

The XSLT processor takes one or more XML source document and one or more XSLT style sheet modules. The XSLT processor apply the rules, instructions and directives provided through the XSLT style sheet modules, and generate a new output document based on the contents of the existing ones, leaving the original files unchanged.

In this context, the EAXML import is achieved through an external tool (EAXML processor) that takes as inputs the EAXML file and an XSLT transformation script (EAST_ADL_import.xslt) and generate a file format natively supported by the tool.

The following picture explain the export/import transformation path

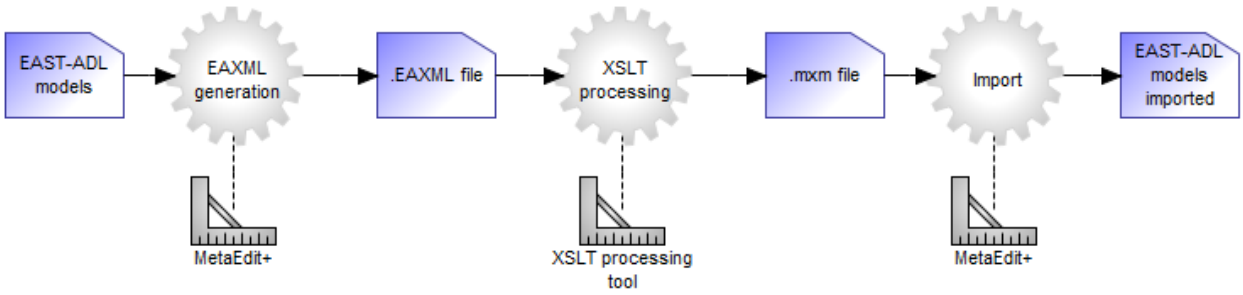


Figure 5-40. export/import transformation path for EAXML file through XSLT processor.

A second approach relies on the use of ATL (Atlas Model Transformation language).

The source model, expressed using a proprietary XML format, is processed together with the source meta-model and the target meta-model by the ATL processor.

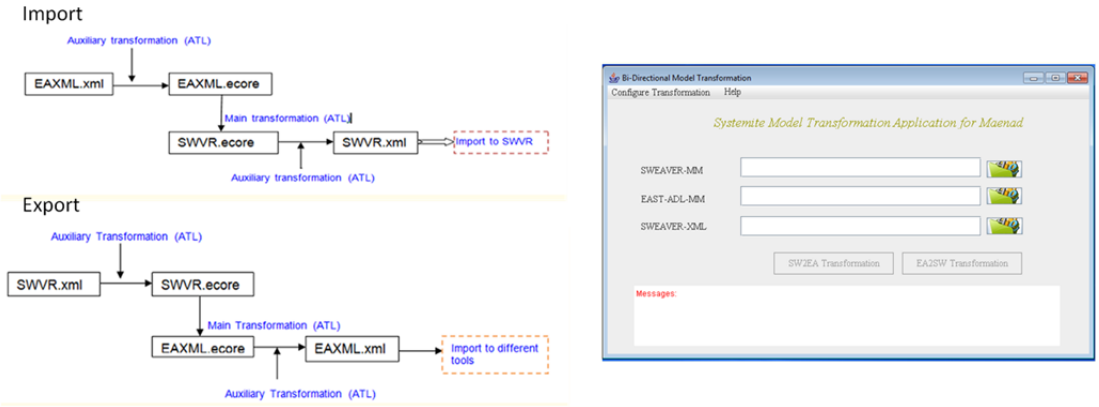


Figure 5-41. import/export transformation path through ATL.

Evaluation Results

Import/export functionalities have been tested on a subset of the “Propulsion” subsystem.

The EAXML generator has been activated using the launcher button on the main MetaEdit+ toolbar

As result, the embedded EAXML generator has created the EAXML file starting from the input model.

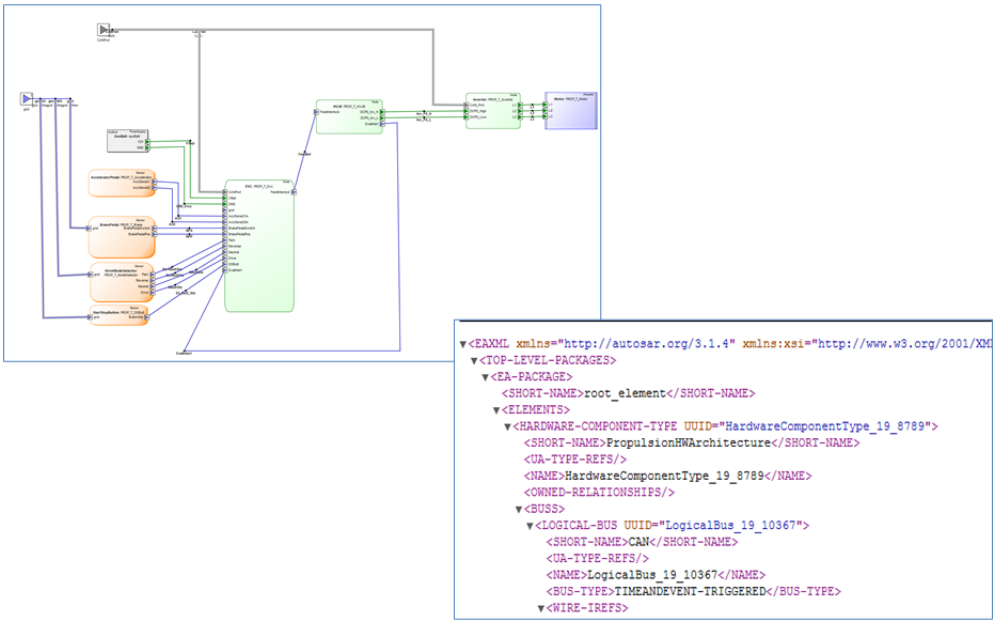


Figure 5-42. Transformation of HDA propulsion subsystem to EAXML format.

The resulting EAXML file adhere to the “EAST-ADL_M2.1.9.20110830.xsd” XML schema.

The generated EAXML file has been checked through a free XML validator tool (http://www.xmlvalidation.com) against the reference schema.

At the time of writing, no deviation between the generated output file and the reference schema has been detected.

The resulting EAXML output file has been re-imported as a new model in MetaEdit. The development environment provides user command to automatically run the XSLT processor that transforms the EAXML file to native format.

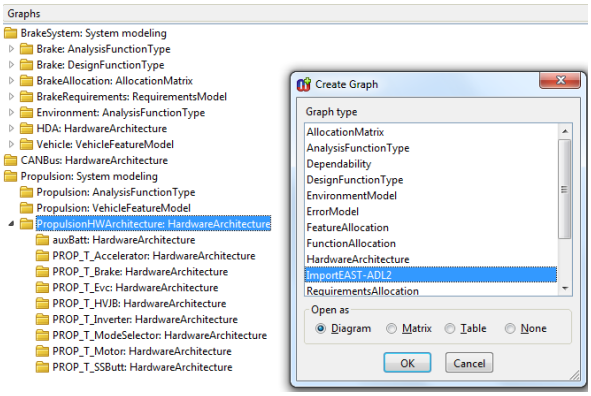


Figure 5-43. Import procedure of EAXML file in MetaEdit+.

The EAXML file contains information about data types, relationship and hierarchy of the original model, but not information about the representation data. The tool create automatically a representation of the model, the original layout should be restored manually.

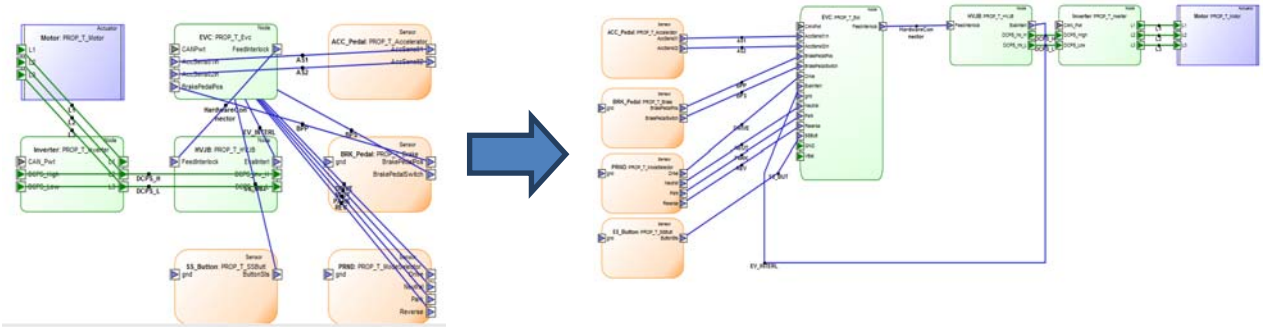


Figure 5-44. . Imported model.

The models exchange between two development environments has been tested on a small exemplary subsystem. The trials demonstrate the validity of the approach. Further experiments with complex systems are planned to proof the concept.

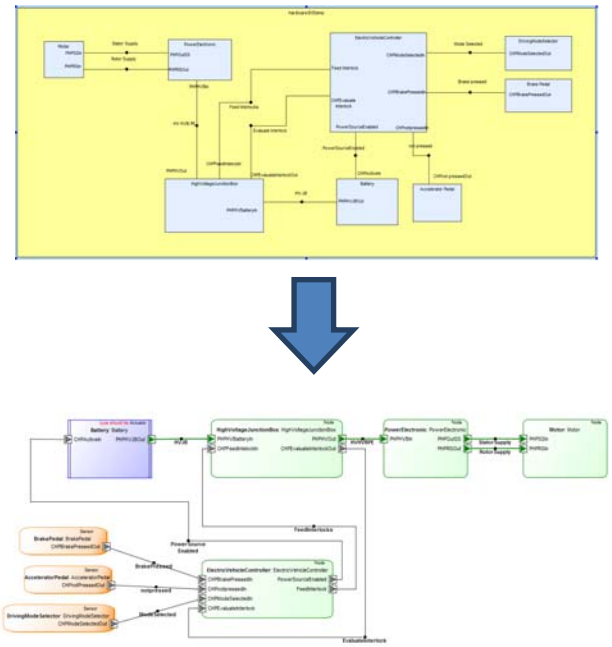


Figure 5-45. Model exchange between different development environments

5.9 AUTOSAR Gateway

Procedures

AUTOSAR Gateway (ARGateway) is designed for the generation of an AUTOSAR compliant architecture from an EAST-ADL2 model. The generation process is twofold, namely from EAST-ADL2 model generation of an AUTOSAR model takes place and from this model, ARXML file can be generated. This is shown on the Figure 5-46. These two processes can be triggered separately by the user. Naturally, an AUTOSAR model first has to be generated. In order to represent the EAST-ADL2 specification, UML profile for the EAST-ADL2 is used. The same solution holds for the modelling of the AUTOSAR architectures. Therefore the first transformation is a transformation between the model expressed with the EAST-ADL2 UML profile and the model expressed with the AUTOSAR UML profile. Generation of a model based on the AUTOSAR UML profile first, instead of direct generation of an ARXML file from the EAST-ADL2 model (which was the approach used in the previous implementation of an AUTOSAR Gateway) has a significant advantage. Namely, the designer can manually change the final design of an implementation level working directly on the models.

Current version of the AUTOSAR Gateway features with a predefined approach for the generation of a software architecture. In fact the generation of a software architecture, i.e. software component prototypes and their internal behaviour, poses the main question. In our predefined approach, software architecture is mostly generated based on the compositional structure of a functional specification. Non-elementary functions are transformed into composite software components. Each elementary function is transformed into the runnable entity and runnable entities are grouped into the software components in a way in which elementary functions were grouped into the non-elementary functions. Of course if two elementary functions were grouped into the same non-elementary function, but allocated to different nodes, their corresponding runnable entities will not state a part of the same internal behaviour. Previous implementation of an AUTOSAR Gateway was following simpler approach where for each elementary function, one atomic software component with one runnable entity have been generated.

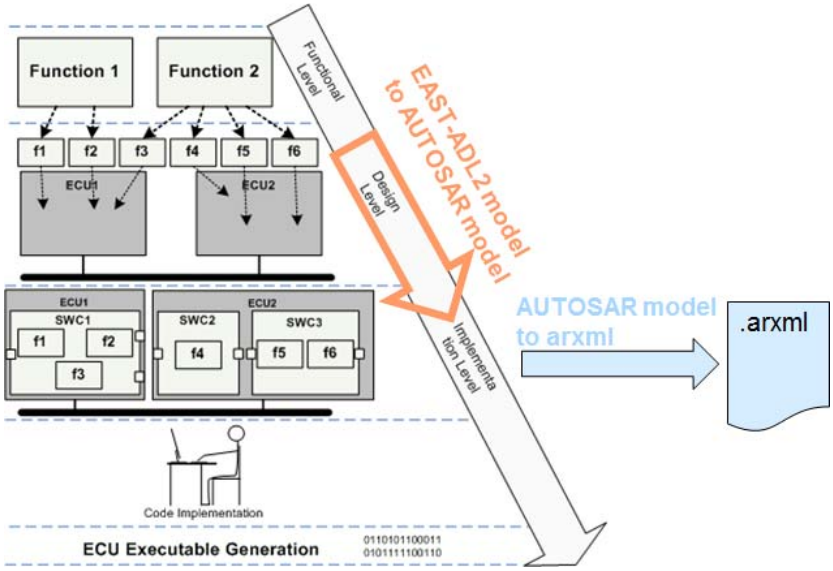


Figure 5-46. AUTOSAR Gateway twofold process.

Evaluation Results

AUTOSAR Gateway has been tested on the EAST-ADL2 model representing the Brake-by-Wire. The results are shown on a set of below figures. Result of a first step of a transformation this is a

UML model stereotyped with an AUTOSAR UML profile. This model is packaged into the *Technical View* package. This package contains three sub-packages: *Application View*, *Topology View* and *Mapping View*. The first contains specification of Software Component Types and Prototypes, their ports, interfaces, connectors and internal behaviour (see Figure 5-48 and Figure 5-49). Topology View contains specification of ECUs, their instances (ECUInstance), sensors and actuators and physical channels (see Figure 5-50). In the Mapping View package there is a specification of software components prototypes mapping into the execution nodes.

In the second step of a transformation, ARXML file is generated. The result of a transformation can be viewed in the Artop tool (seeFigure 5-51).

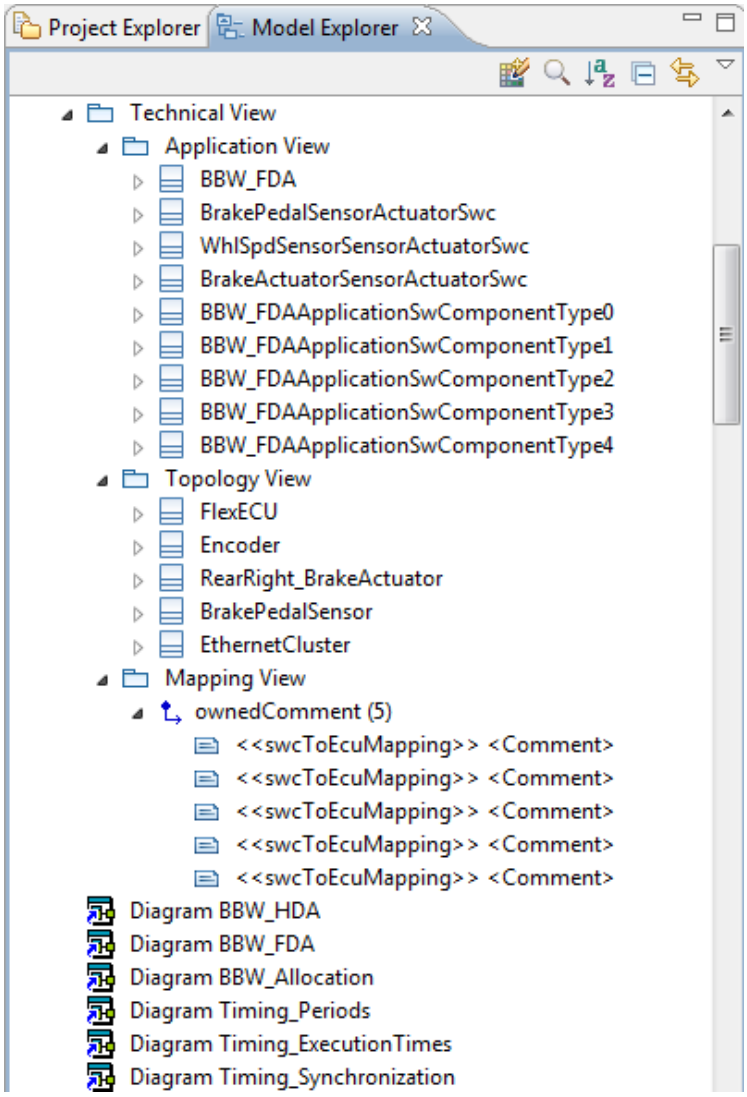


Figure 5-47. View on the generated model in the Model Explorer.

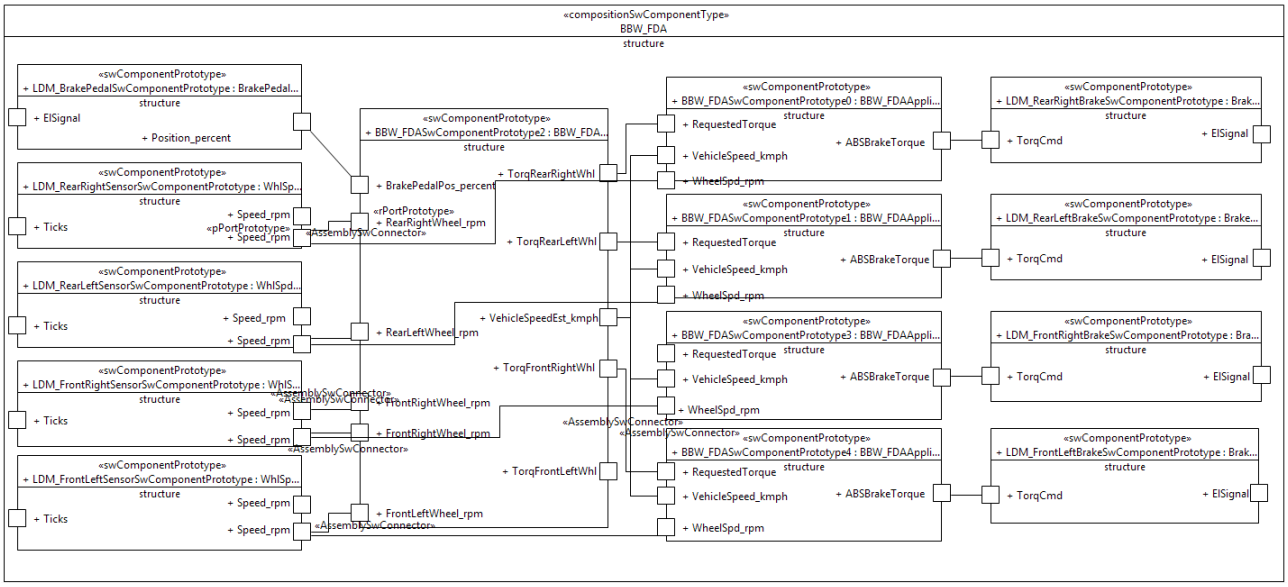


Figure 5-48. Composite diagram depicting Software Architecture.

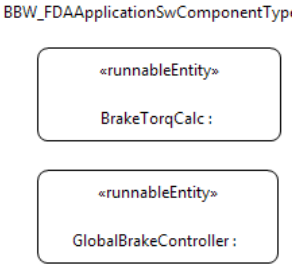


Figure 5-49. Runnable Entities.

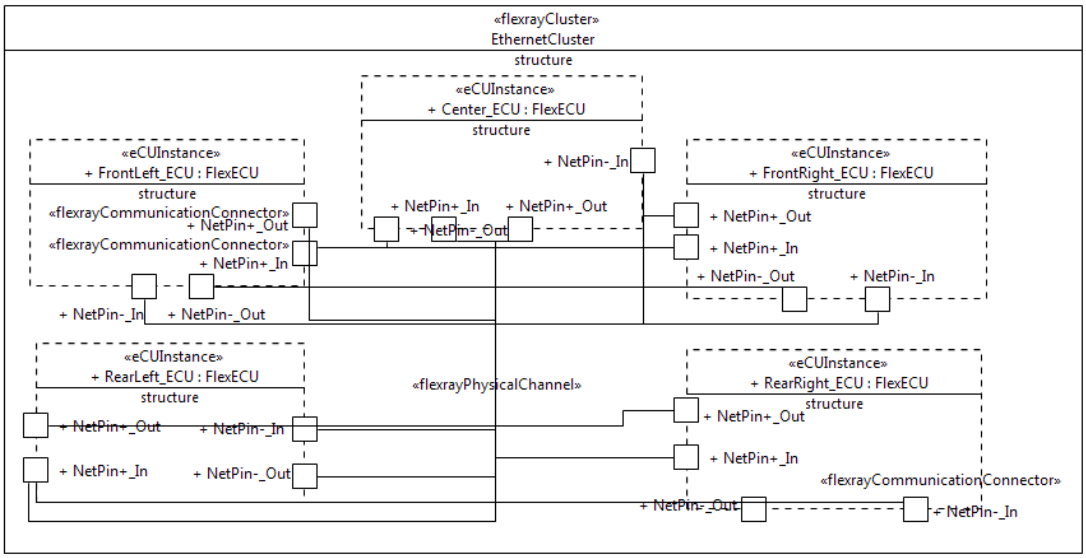


Figure 5-50. Flexray Cluster.

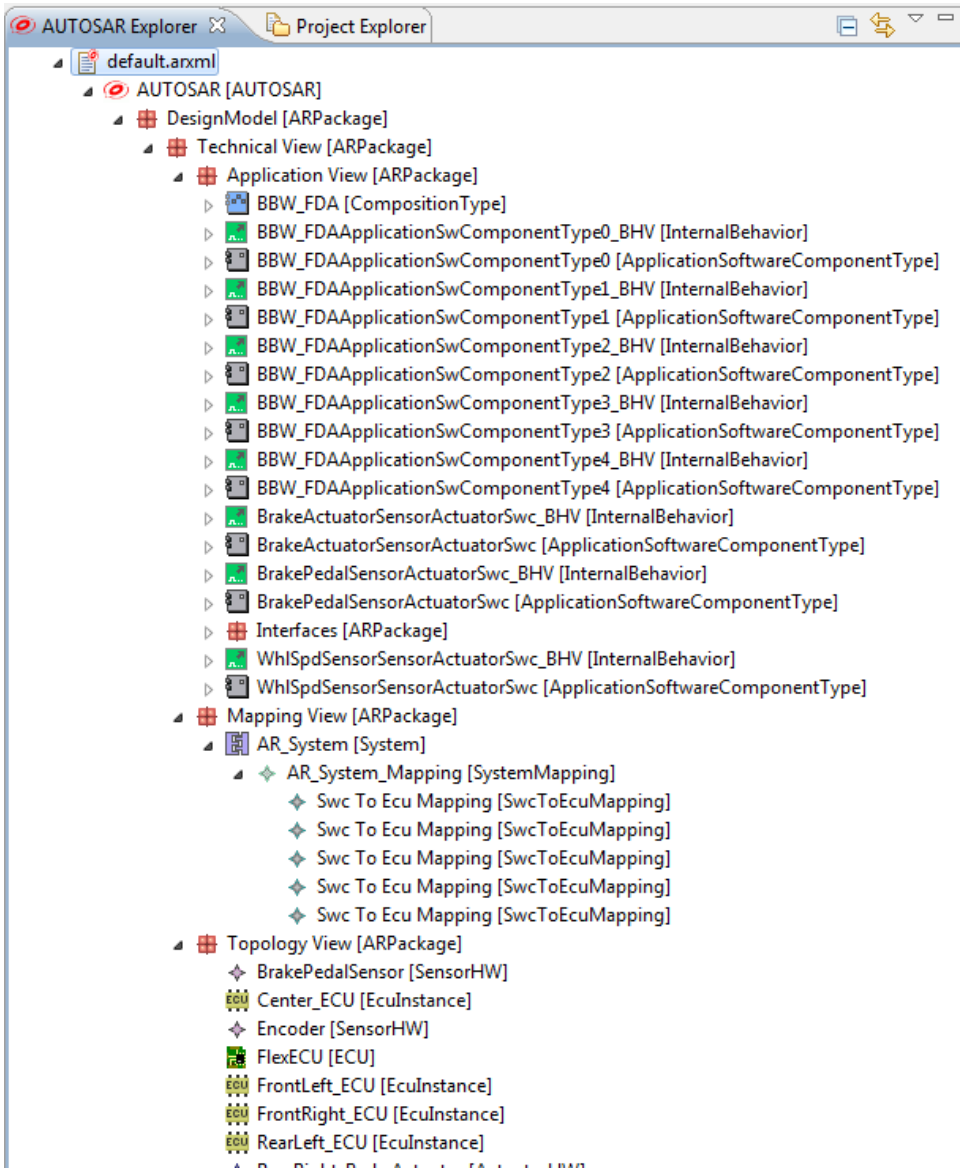


Figure 5-51. View on the generated ARXML file in the Artop AUTOSAR Explorer.

5.10 Model-checking Gateway

The newly developed language package Behavior Constraint Description Annex provides a formal semantics (i.e. Hybrid-System Model) for behavior annotations in EAST-ADL, such as requirement refinement, design contract specification, and the definition of error logics. Due to this formal semantics, model-checking through some tools (e.g. SPIN, UPPAAL, and Matlab/Simulink) becomes possible.

Procedures

In EAST-ADL, a behavior annotation, referred to as behavior constraint, can get different roles depending on the declared target associations. For example, such a behavior constraint can be used to capture the bounds of the acceptable behaviors of a system function. A behavior constraint can also be used to refine the textual statements of requirements including assumed system operational situations. There are three categories of such constraints: 1. Attribute Quantification Constraint for the declarations of value attributes (e.g. parameters and variables) and the related acausal quantifications; 2. Temporal Constraints for the declarations of of state-machine constraints where the history of behaviors on a timeline is taken into consideration; and 3.

Computation Constraints for the declarations of cause-effect dependencies of data in terms of logical transformations (for data assignments) and logical paths.

Figure 5-52 shows the declaration of a quantification constraint in regard to the slip rate estimation for ABS function. According to the constraint description, the estimated slip rate (SlipRate) should follow the slip rate quantification (SlipRateQuantification) with the expression:

SlipRate=(VehicleSpeedIn - WheelSpeedIn*WheelRadius)/VehicleSpeedIn.

Here, the VehicleSpeedIn and WheelSpeedIn are two variables received through the functional ports vehicleSpeedRef_in and wheelSpeed_in respectively. The WheelRadius is a constant with the value of maximum allowed wheel radius. The EAST-ADL Behavior Constraint Description Annex uses an abstract notion of time, referred to as logical time condition, as the time basis for quantifying physical dynamics by means of continuous- and discrete-time model, or for defining the timed guard conditions and invariants of state-machines or computations.

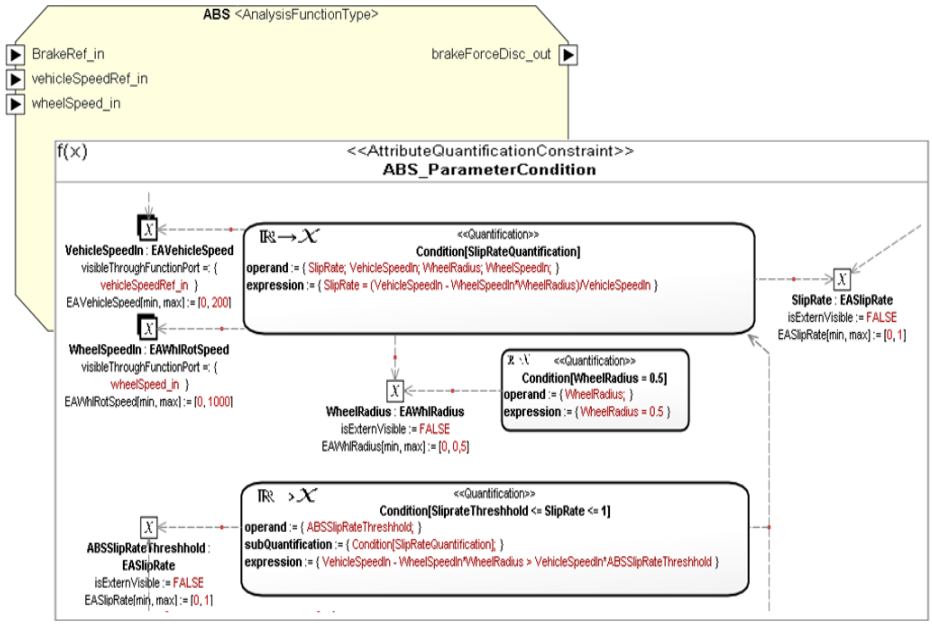


Figure 5-52. An excerpt of the attribute quantification constraint description for an ABS function

Figure 5-53 shows the declaration of temporal constraint description in state machine (SM) for the same system function. The state invariants and transition guards are precisely defined by some attribute quantification specifications.

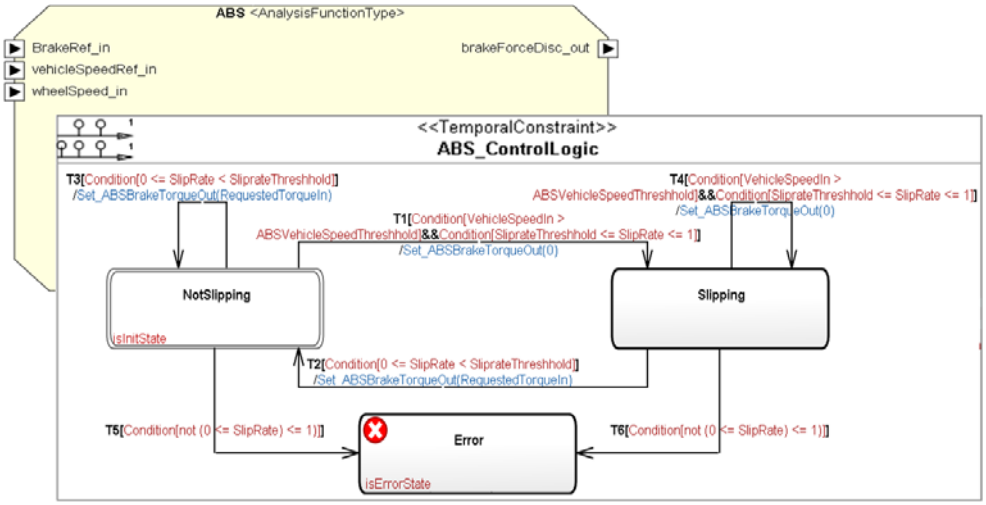


Figure 5-53. The temporal constraint description for an ABS function.

In Figure 5-54, the specification of computation constraint declares two valid invocations to a transformation Set_ABSBrakeTorqueOut that calculates the ABS brake torque request. The acual content of the logical transformation is currently given in C language or Matlab Script.

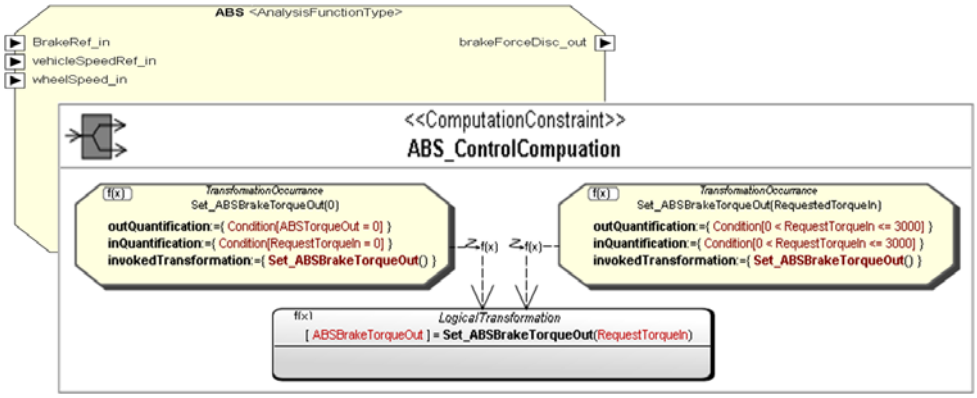


Figure 5-54. The computation constraint description for an ABS function.

A composition of such EAST-ADL behavior constraint description can be then transformed to the UPPAAL/SPIN tool for the analysis support. See Figure 5-55. The transformation result to UPPAAL is shown in Figure 5-56.

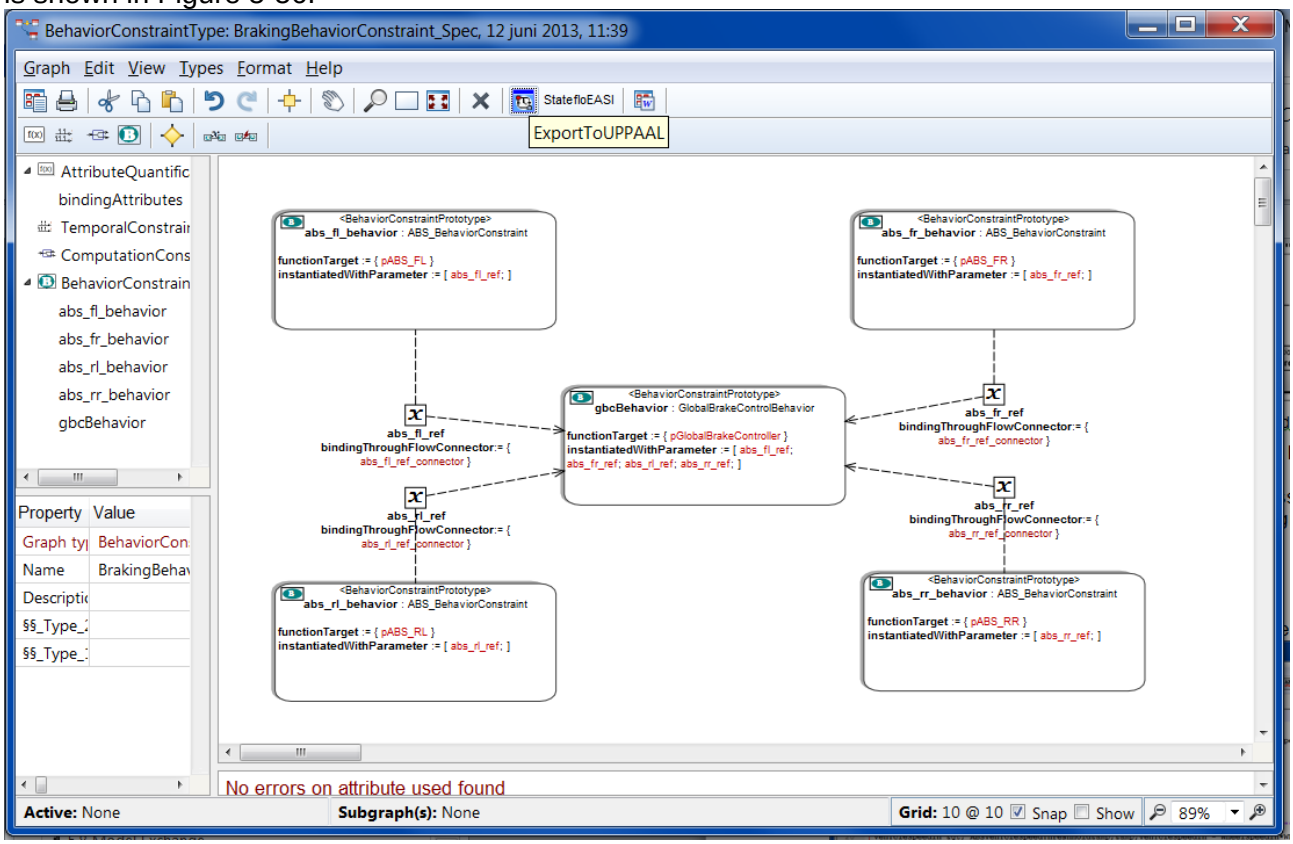


Figure 5-55. The ME+ tool support for generating UPPAAL model from an EAST-ADL composition of five temporal constraints in state-machines for four ABS functions and one vehicle function.

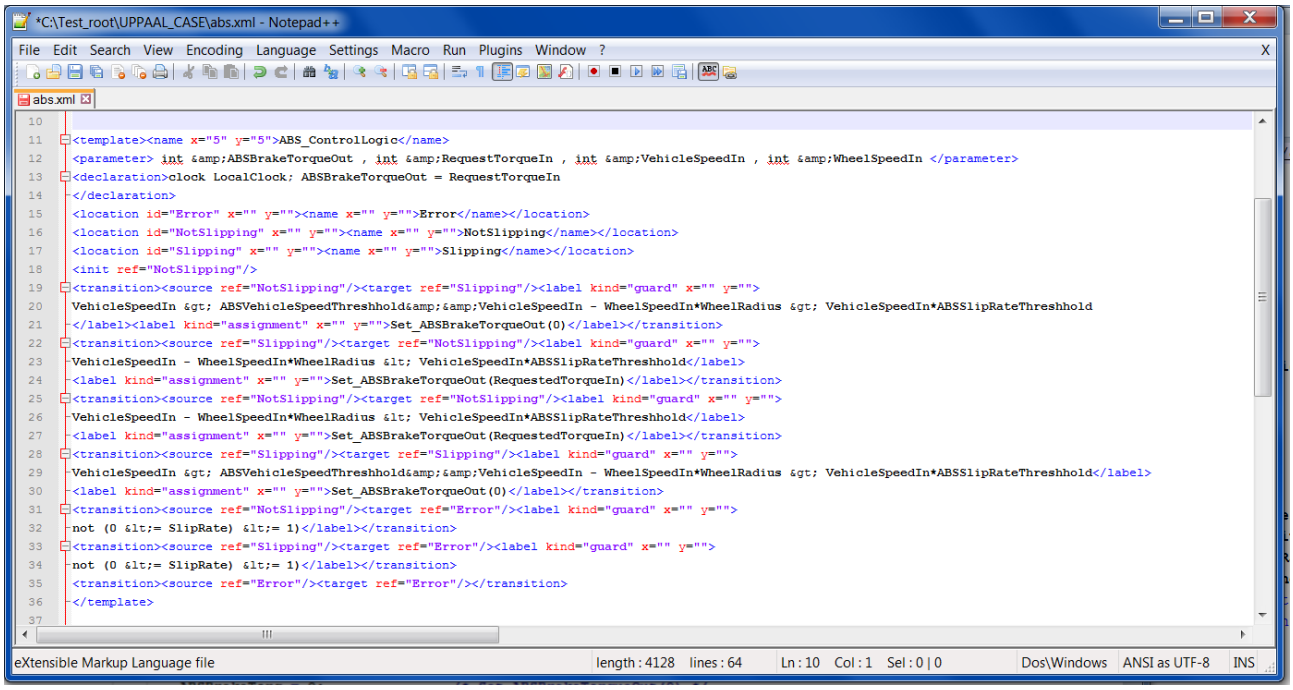


Figure 5-56. The UPPAAL Code of the ABS Function.

Evaluation Results

The model transformations to UPPAAL and SPIN have been tested with the BBW case system. Due to the formal semantics of EAST-ADL behavior constraint, the transformations are well defined. It is shown this language and tool support allows the users to exhaustively verifying the model against requirements, which need to be given in temporal logic statements (CTL). The test case generation has been investigated and but is currently conducted in collaboration with the AREMIS MBAT project due to the availability of testing tool.

6 Verification and Validation

Fault injection is a fundamental technique required by ISO26262 to provide evidence that the obtained product complies with the safety requirements. The main goal of this technique is to support the assessment of:

- the correct implementation of functional safety and technical safety requirements during:
  - HW/SW integration, system integration and vehicle integration phases, to verify that the integrated elements interact correctly.
  - HW development phase.
  - SW unit development phase and during SW unit integration on a SW architecture
- the effectiveness of a safety mechanism's diagnostic coverage at the HW/SW development phase and failure coverage at system/vehicle level

One of the main goals of the MAENAD project is to develop capabilities for modelling and analysis support, following ISO 26262. In this context, MAENAD language and related tools could provide support for experimental V&V activities based on fault injection technique with different scope: during the design phase of fault injection experiments, where models of the systems are used to transfer information useful for the design of test experiments according with the methods expressed by the norm, and during the execution of fault injection experiments.

Across the whole WP6 analysis activities, a test bench has been used as a basis to evaluate, to some extent, the suitability of MAENAD modelling language to support design of V&V activities and their execution using on a real subsystem.

6.1 Fault Injection

The test bench is designed to provide support for Fault injection experiment, especially those related to Integration phases (system level and vehicle level) of the safety lifecycle. Focus is on a subsystem of a real vehicle, interaction of this subsystem with the rest of the plant is emulated through dedicated HIL technologies.

The following picture provide an overview on the physical demonstrator, green boxes highlight the real component of the physical test bench setup. Interaction of the physical components with the emulated subsystem is done through the communication network.

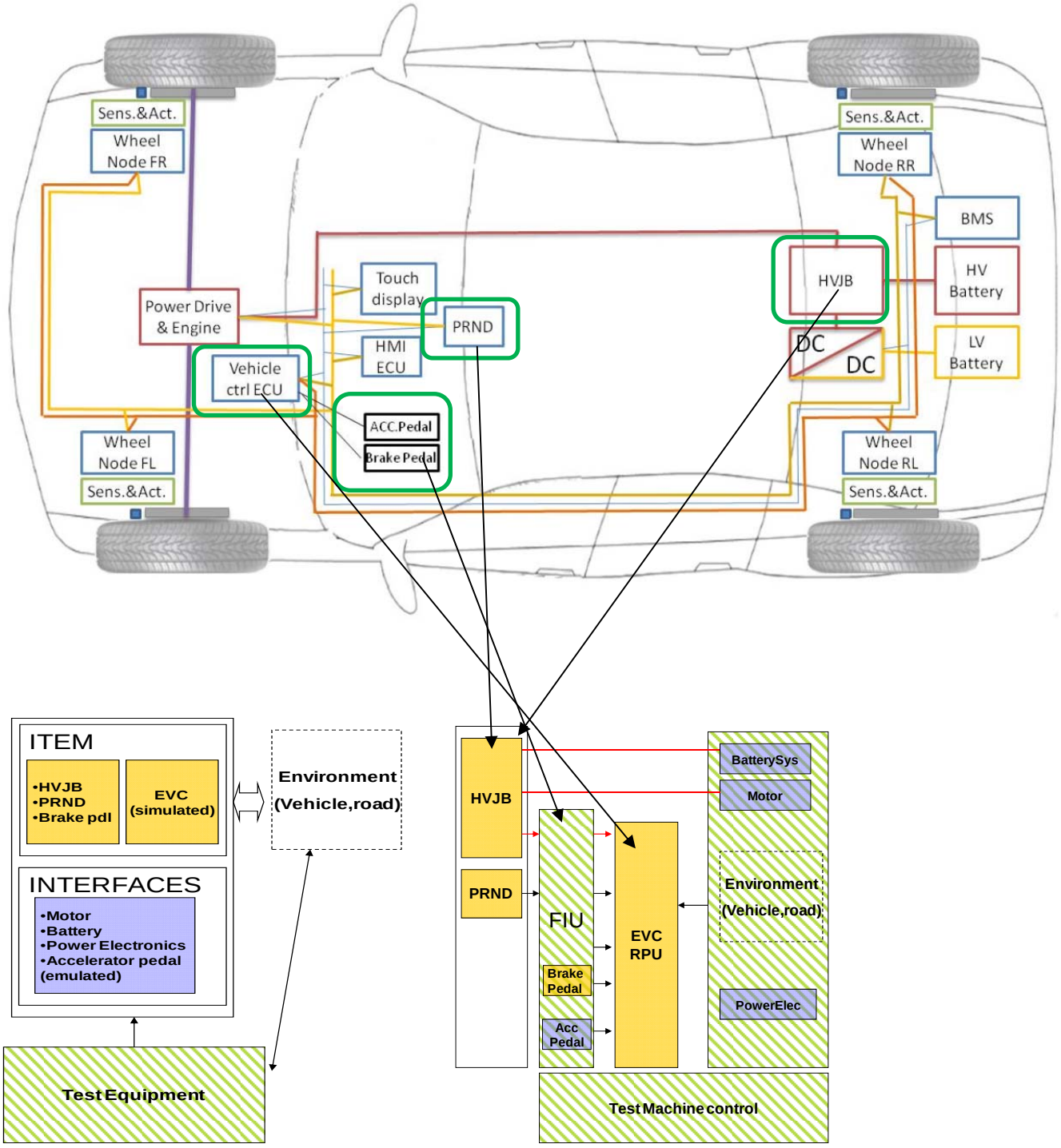
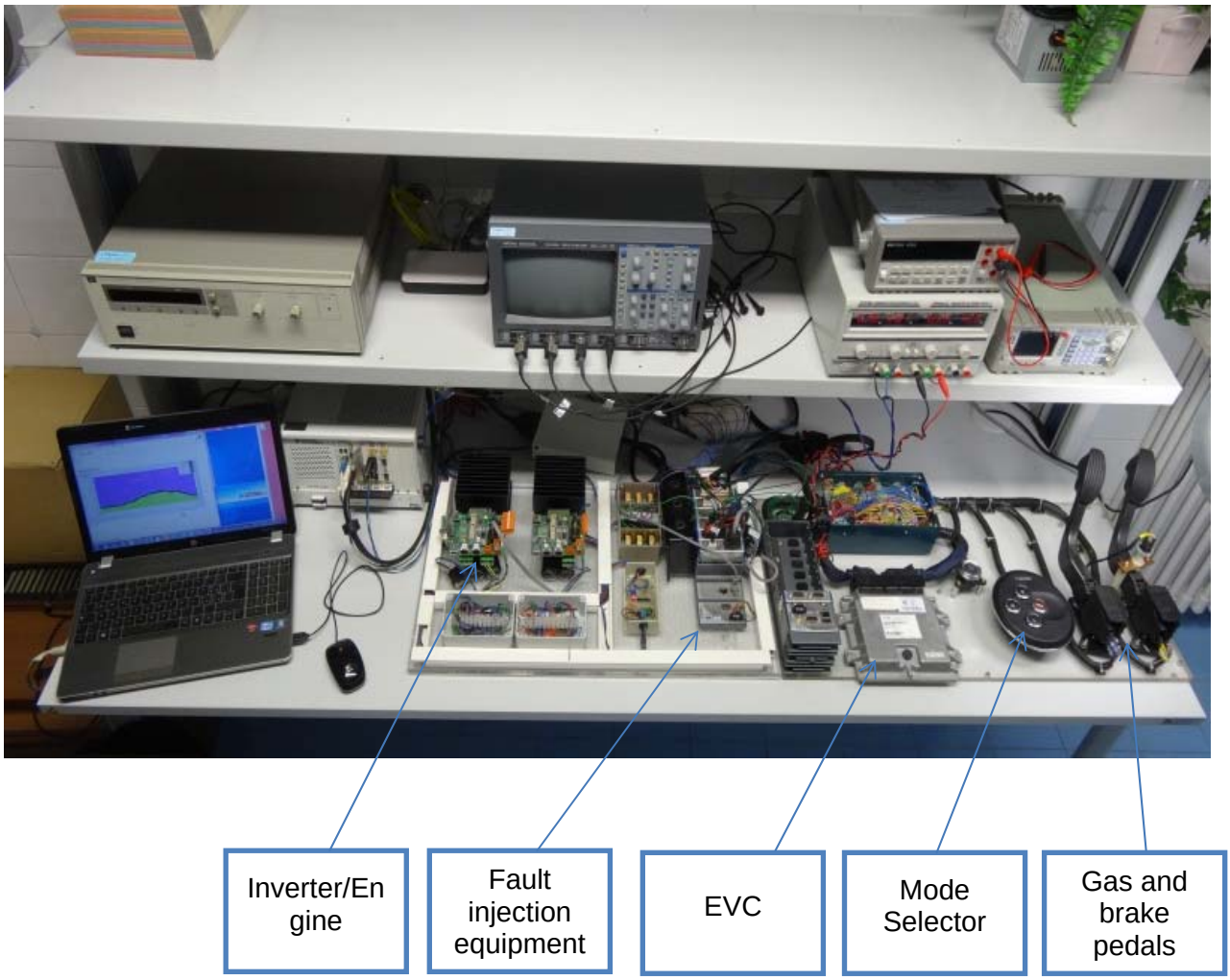


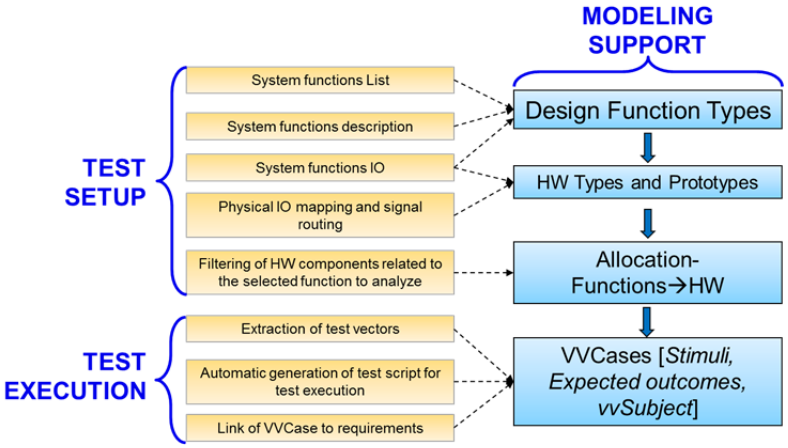
Figure 6-1:Test bench demonstrator concept and design diagram

The following picture show the physical prototype of the selected subsystem



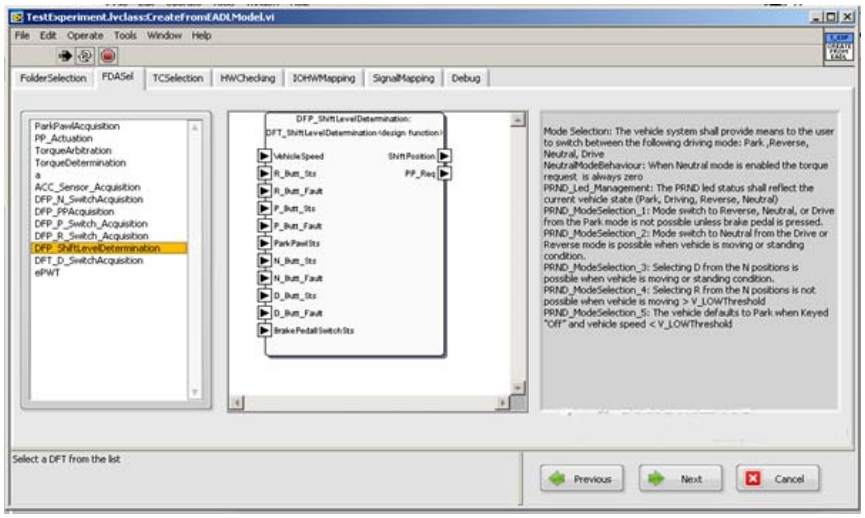
A tool has been built to support the V&V activities. It takes as an input the file generated from an EAST ADL model and guide the user through the steps of Test setup/selection and test executions

The following picture highlight the support from the EAST ADL language and the V&V activities

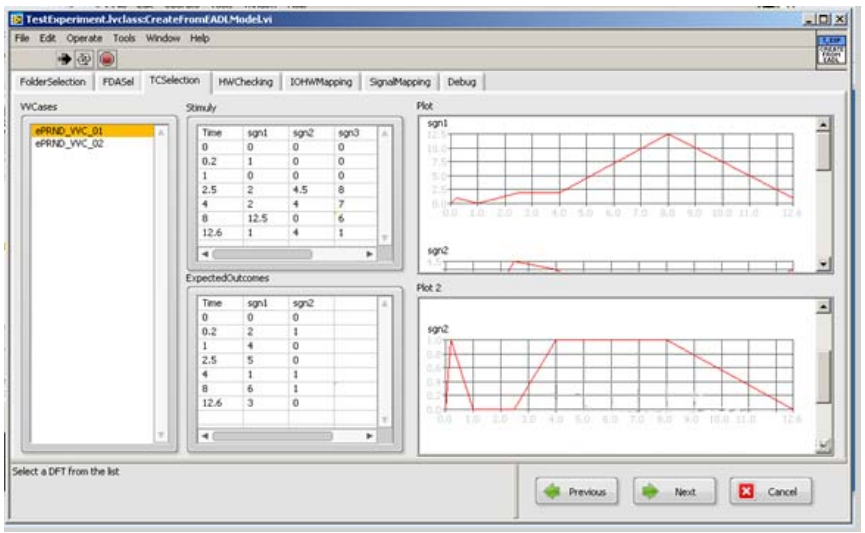


The tools import the EAXML file generated from an EAST-ADL model. Through the parsing of the EAXML file, it extract all the Design function types and prototypes, the HW prototypes, the list of requirements and the association to model elements, the allocation of functions to HW elements, the description of Verification and Validation related items and the association of such elements to design functions.

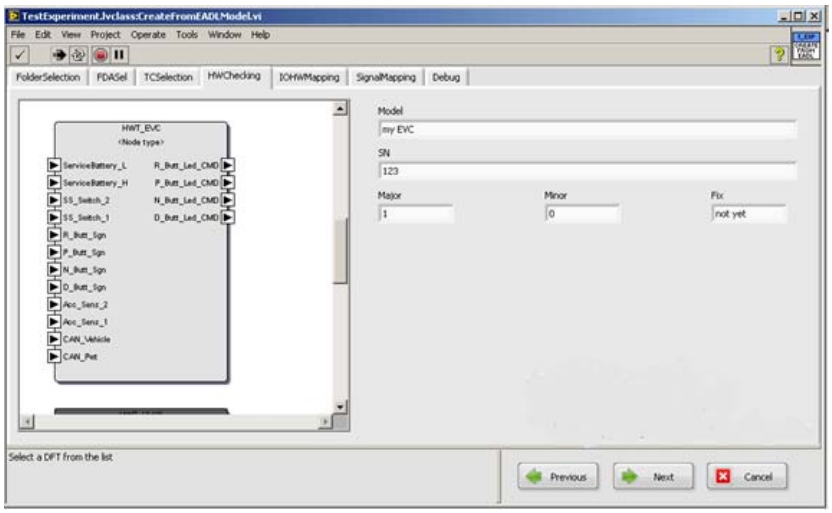
After the import, the user interface present to the user a list of the available functions on the subsystem and provide means to select one of them for a successive evaluation. From the link of requirements to design functions, the tool automatically extract the requirements related to the selected function.



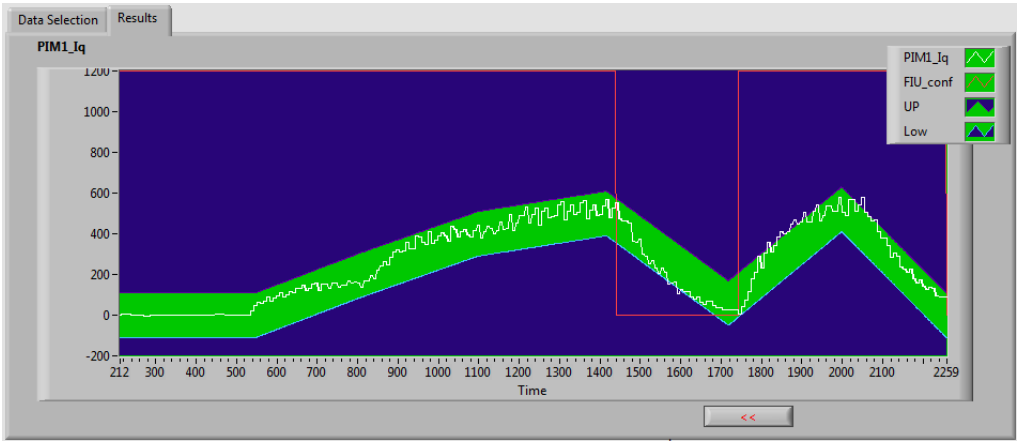
After the selection of the function, the tool filter and extracts automatically the associated V&VCases. and provide a means for the user to select one of them. Test Stimuli and expected outcomes are extracted automatically from the model, where the test procedure is described and references external file



From the information about the functions to HW mapping, the tool extract the HW items associated to the selected function. In this context, the tools provide means to include additional information about the SUT like serial number.



The tool automatically uploads on the test equipment the Test stimuli file, run the execution acquiring and logging the test experiment, and for each signal, compare the expected outcomes with the actual outcomes. The comparison is performed generating two bounding mask from the expected time profile and checking if the actual signal outcome remains within the upper and lower limits



7 Conclusion

This deliverable summarizes the progress and the activities related to the analysis of the MAENAD methodology and of the related tools through the application of tools and methodology on demonstrators.

8 References

[1] www.atesst.org
[2] MAENAD: Deliverable D3.1.1_V1.0
[3] MAENAD: Deliverable D3.2.1_V1.0
[4] MAENAD: Deliverable D5.2.1_V1.0
[5] MAENAD: Deliverable D6.2.1_V1.0
[6] Functional Mock-up Interface, FMI: https://www.fmi-standard.org