MDX and AUTOSAR Standards for Model Sharing to leverage Tier1 - OEM cooperation in the ECU software development

Stéphane Louvet[1], Dr. Mouham Tanimou[2]

[1] Robert Bosch (France) SAS, Diesel Gasoline Systems, Electronic Control, 32 avenue Michelet, 93404 Saint-Ouen, France
[2] Robert Bosch GmbH, Diesel Gasoline Systems, Electronic Control, Postfach 30 02 20, 70442 Stuttgart, Germany

Keywords : Engine Control, embedded software, Model Based Design, Software Sharing, Model Sharing, MDX, AUTOSAR, automatic code generation, ASCET, Simulink®

Abstract / Motivation

Software Sharing in automotive embedded software development has continuously grown over the last decades and is still getting more importance and attention. Main advantages for Software Sharing, as seen by the OEM, are acceleration of development time and cycle as well as a full flexibility to customize the final product, with introduction of its own software.
Although classic software sharing has shown its capabilities in practice, it shows some limitations and need to be extended in order to cover additional use cases. The concept of model sharing can help to address those use cases, the following paper outlines the use of ASAM-MDX and AUTOSAR standards with the model sharing to enhance customer cooperation.

1. Introduction

An increasing number of car manufacturers develop their own functionalities and/or their own software platform in order to communize functions across several control units supplied by different Tier 1 companies. This happens in general as stub-software - already compiled - to be integrated in the final Software. This proven process in practice needs however to be improved and can thus be enhanced to increase development efficiency.
In order to achieve such goals, OEMs need the ability to test enhanced or newly developed functions in a very early stage of development, those function checks need to be performed continuously along the V-Cycle development up to the final software. Model Based Design is a general accepted mean to do this and is becoming widespread in the automotive industry.
Within the automotive industry, modeling-tools like Simulink® (together with Embedded Coder or TargetLink® as code generators) and ASCET are widely used by OEMs to verify their functions in an early phase using virtual and/or rapid prototyping methodologies. The so verified model together with the achieved data specification can then be exchanged with suppliers as executable specification for functions to be integrated in the ECU; this is one side of the medal of what is called “Model Sharing” at Bosch. The other side of the same medal consists of provision of a development environment for simulation as well as for software build, together with a set of common modeling guidelines and libraries and common data description standard for further development.
This approach has a high degree of flexibility, since it provides a framework for OEMs starting from the scratch, but also allow to adapt the derived Process, Method and Tools to car manufacturers having their own approaches. Bosch has established with this approach a five-level classification for model Sharing between Tier1 and OEM, which starts from "Use of ECU virtualization environment” and goes up to “Joint Development”.
Use of Standards for the data description (e.g. ASAM with -MDX, CDF …- and/or AUTOSAR with service libraries, methodology …) as well as standardized interfaces specifications (e.g. as published with AUTOSAR R4.x) can significantly increase the effectiveness of this approach; e.g. by easing labels data management and exchange between development partners through a database (e.g. Visu IT!-ADD), as well as exchange of models.
The use of standard for data description with the approach described above raises following central questions that are subject to further analysis in this paper:

- How can standards be used with ECU Software development tools?
- What are the constraints on modelling pattern?
- How can migration of models be handled/managed between different standards?

In this paper we will present a rough description of benefits coming with the standards ASAM-MDX and AUTOSAR with respect to mutual development cooperation. Furthermore, we will focus also on their deployment with modeling tools; the technical aspect to ensure a seamless migration from MDX to AUTOSAR standard will be also described. Some of the results as well as benefits achieved with these standard in customer cooperation within series projects will also be presented.

2. Model-Sharing

Model-Sharing is a way of exchanging models and model artefacts in order to ease Software development and cooperation between two or several partners; it is part of our Model Based Software Development framework in the ECU development domain. It is also a way to allow rapid prototyping as well as simulation.
Basically, motivations for Model Sharing in embedded SW development process are:

- OEM focuses on physical modeling, simulation and rapid prototyping. The OEM can also, via model exchanges, take benefit from supplier physical modeling and simulation know-how.
- The supplier is in charge of industrialization of provided models as well as code generation from these models.

Provided models (In general in form of ASCET, Simulink® or TargetLink® models) by the OEM can be used as executable specifications for Software development by the supplier. Using a common tool with it a common language, and standardized files by the development partners for modeling leads to reduction of the amount of to be exchanged documents, and thus enhance the understanding of OEM requirements by the suppliers. This can significantly lead also to reduction of development effort and development time as well as delays in the development.

Typical use cases for Model Sharing are described in the sections below.

Use case 1: OEM describes the expected functionalities in customer specific functions developed by the supplier

- the desired changes can be implemented directly in the model
- Simulation/Rapid Prototyping can be carried out w/o waiting for an “official” SW release

Use case 2: OEM develops its owns model and the supplier is in charge of the “so-called” industrialization of the code

- This corresponds to the so-called “Commissioned Development” in Bosch process development
- The code implementation is simplified if both partners are working with the same modeling environment (examples : ASCET, Simulink®, TargetLink®)
- Functionalities can be tested with Rapid/virtual Prototyping before delivery to the supplier
- corrected (enhanced) model can be delivered back to the OEM: This helps to avoid additional traceability documents and minimize risk of forgotten improvements in own model

Cooperation levels

Since each customer has its own dedicated development process, it has therefore specific requirements for development cooperation with suppliers. On the other hand, suppliers need to classify the requests of OEM to find a common approach. Following this line of thought a five-level classification for model Sharing between Tier1 and OEM has been performed at Bosch: from "Use of ECU virtualization environment” up to “Joint Development” [3]. This build a base for discussions with OEMs before starting a new project and it provides a framework for development methodology and contracting. The cooperation usually contains aspects of different types of contracts like service or licensing contract. However, these aspects are typically covered by one agreement describing the complete scope of the cooperation.

Cooperation Levels	Description
Level 0	• Processing of Bosch model • Build simulation environment at customer
Level 1	• Non functional adaptation and processing of customer model • No delivery of model artefacts to customer
Level 2	• Non functional enhancement and processing of customer model • Delivery of enhanced model and model artefacts to customer
Level 3a	• Functional modification of customer model • Delivery of modified model to the customer
Level 3b	• Functional modification of Bosch model by customer • Delivery of modified model to Bosch
Level 4	• Joint development • Comparable level of contribution from Bosch and customer • Sharing of created Intellectual Property

Figure 1 - Different cooperation levels for model sharing

The picture below illustrates those cooperation levels in a contracting and process flow view

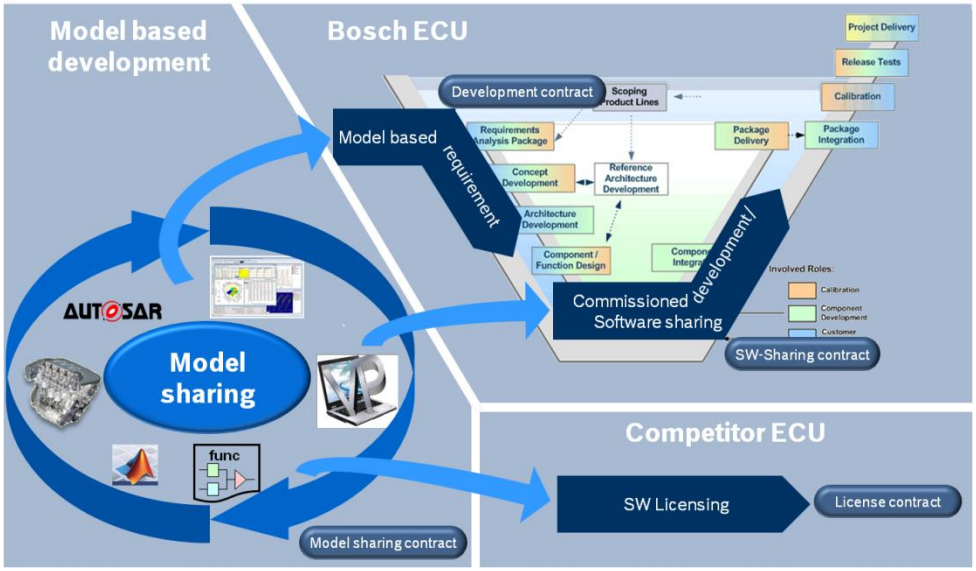

Figure 2 - Workflow for Model Sharing and contracting

Some of the cooperation models will be shortly described hereafter to better understand the use cases which are addressed in this paper.

Level 0: Virtual Prototyping

Since several years, it can be observed that the trend for ECU software engineering is moving towards PC simulation in order to

1) Save costs by increased development productivity and
2) Manage complexity and apply holistic engineering approaches.
3) Allow different development speed between HW, mechanics and SW

And OEM are requesting from Tier1 supplier a stringent support of development cycle oriented use-cases at OEM in a virtual manner. This requires to build a virtual test bench, which contains virtual engine ECU-Software (as it is running in the real ECU) as well as representative plant models in a virtual environment, which are real time capable. For the virtual test bench, PC-executable ECU-functions in a powerful simulation environment are then needed [1] [2].

Level 1: Processing of OEM Model

This level of model based cooperation is the so called “Commissionedd Development” within the development process at Bosch; the OEM delivers a (simulatable) model that has to be processed by the Tier1 supplier to generate production code without any change on the model. Alternatively, the OEM

can also deliver a specification (e.g. PDF-document) and the supplier creates a model according to this specification. This model will then be delivered to the OEM for further development.
Using same modeling tool by the OEM and the Tier1 supplier helps to avoid converting the model to another modeling environment.

Level 2: OEM Model Enhancement and Processing
This model sharing cooperation level is also considered as part of the development process "Commissioned Development" at Bosch.
The OEM delivers a model and the Tier1 supplier can introduce non-functional necessary modifications in the model for several purposes, such as:

- optimisation for production code generation
- model corrections in case of mistakes found during model implementation
- specific interfaces e.g. for diagnostic

In comparison to cooperation Level 1, the final model contains added value introduced by the Tier1 supplier to the original the OEM model. Furthermore the development efficiency is getting enhanced, since the modified model by the supplier could be redelivered to the OEM and thus modifications done by the supplier are then available to the OEM and so risks of model discrepancies are reduced since the corrections are done at a single site.

Level 3:
Level 3a: Use of OEM Models in Cooperation
The OEM owns a model and delivers it to the supplier who is then charged to industrialize the model and generate production code generation out of the model. Different to cooperation Level 2, the supplier can introduce functional modifications in the OEM model. The kind of Know-How added by the supplier can be for example specific calculations mechanisms such as special filters (e.g. replacement of PT1 filter by a Butterworth filter to achieve certain goals). The modified models are delivered back to the OEM for further development. In general, for all cooperation levels, a CIL (Customer Interface Layer) is needed to combine OEM functions with Bosch functions.

Level 3b: Use of Bosch Models in Cooperation
The supplier provides its own models to the OEM who can then enhance the model by its own functionalities and Know-How. The OEM delivers back the modified model to the supplier who is then in charge of industrialisation and production code generation. The final responsibility for the model (physical Behaviour as well as code out of the model) remains with the supplier who should review the OEM functional modification for approval. The development time of OEM specific functions can be drastically reduced with this cooperation level since there is no need for the OEM to deliver any specification or schematic that would have to be analysed by the supplier and then interpreted in the model.

Although Know-How/functionality is appended to the original model with this cooperation level, the ownership of the model remain with the original provider of the model; i.e. Ownership of model remain with OEM in case of cooperation level 3A and ownership of model remain with Bosch with cooperation level 3B.

Level 4: Joint development
The joint development deals with a common ownership of the models. The base model can be created at the start of the partnership or can be provided by one of the party. The level of contribution from the supplier and the OEM is comparable and the created Intellectual Properties are shared. It leads to accelerated development: a unique model can be continuously exchanged between the supplier and the OEM or modified by one party with the contribution of the other party. As a constraint, it is necessary to define clear responsibilities between the supplier and the OEM.

3. Standards with Model-Sharing cooperation
ASAM-MDX standard and AUTOSAR standards are being considered in this paper.
The ASAM-MDX standard is an ASAM (Association for Standardisation of Automation and Measuring systems: http://www.asam.net) standard, and the signification of MDX is "Meta Data Exchange Format". This format is often used with another ASAM standard called CDF (Calibration Data Format).
AUTOSAR stands for "AUTomotive Open System ARchitecture": http://www.autosar.org. This standard has been released by a consortium of car makers, software suppliers and tools suppliers. This standard describes also data specification for development collaboration.

Both standards (ASAM-MDX as well as AUTOSAR) describe the interfaces and data definition used in the corresponding models within XML files. The AUTOSAR standard is more complete since it describes also the Basic Software as well as the interfaces between the Application Software and the Basic Software. Both standards can not be applied at the same time on a model: the code can be either generated with a MDX target or with an AUTOSAR target.
The most evident advantage of such standard remains in the fact that the data descriptions can be easily shared between OEM and suppliers at the same time as the models exchange. Since the format used is XML-language, these files can be processed more easily than Excel-like formats and are less sensitive to characters issues. Another advantage of these standards is that it helps to avoid developing specific tools or scripts as it is the case when using specific OEM or supplier formats. Development activities can then be performed with standard tools from the market without any deep customization.

As generally known, a function design can be performed by using different modelling styles. Hence standards may have an impact on the modelling pattern. The cooperation between OEM and suppliers with models exchanges can be eased if both partners are using the same standard for their Model Based Development tool chain. Non-use of same standard can really impede model exchange, especially with regard to library blocks intended for simulation as well as for code generation; some library blocks may include memory states like for filters or flip-flop, and thus the memory states can be described with workspace elements in form of Simulink® objects or Matlab objects.

With R2012B onwards, MathWorks has introduced with a PSP (= Pilot Support Package) in Simulink® a set of MDX objects (derived from Simulink® objects) which have been developed in cooperation with Bosch, intended for use in cooperation projects between Bosch and its customers. Together with this PSP, Bosch has developed a Simulink® library containing AUTOSAR services and other specific functions (e.g. BSW specific functions). In case of Simulink® model exchange containing these library blocks, a simulation can be performed since these objects are based on Simulink® objects; code generation will however require the use of the MDX package. If however the data specification on customer site are based on Matlab objects, simulation will be performed with default values but not with the specified version; a conversion will therefore be needed to have appropriate simulation. Furthermore, working with a standard for the data description avoids also to create its own specific data description rules. It then avoids to create its own scripts to generated the data description files during the code generation phase.

From a modelling point of view, a standard like AUTOSAR provides also standardized interfaces with the Basic Software like diagnostic interfaces, and thus permits to connect easier Application Software functions to the Basic Software services.

From an integration point of view, if the OEM provides Software Sharing code which has been generated with standardized interface, the integration of the Software Components in the supplier Software is made easier since the files can be processed with the standard supplier tool chain, thus avoiding usage of specific tools to generate adapter layers with specific interfaces mechanisms.
One of the main objective of AUTOSAR as standard is to ensure portability of functions towards different platforms and thus ease integration of Software Components from different development partners. Therefore data descriptions are defined in an XML file, which can be processed by a tool during the software build phase to generate automatically the appropriate header and source code files for data declaration and definition.

When sharing code with a supplier, the alignment of standard used at OEM side with standard used by the integrating supplier allows an enhanced efficiency in the continuous integration process; thus necessary software adaptations can be automatized more easily.

Data consistency is a crucial element in the automotive embedded software development (especially when using pre-emptive and cooperative tasks in a single- or multicore context). Data specification information within standard files (MDX and AUTOSAR files) can be used to ensure the data consistency and thus these files can be directly be processed during the software build phase to take care on data consistency mechanism in the final software (hex-file). For MDX file, the so called tool MCOP (Message Copy OPtimizer) is used whereas for AUTOSAR files, the RTE generator ensures the data consistency mechanisms especially for Sender-Receiver implicit communication.
The configuration of MDX objects for a Simulink® model is performed using the MDX explorer (derived from standard Simulink® model explorer) and is part of the data dictionary to be shared between development partners and the configuration of AUTOSAR objects is performed in the workspace (to create AUTOSAR objects) as well as with the AUTOSAR Mapping Editor for Simulink® to prepare the

model for RTE configuration. Working with the same standard between OEM and supplier can reduce the effort of the model preparation at the supplier side for the code generation in case of Model Sharing Level 2 (or so-called Commissioned Development).

With the Model Sharing level 0, the MDX or AUTOSAR standards are supported by tools used at Bosch to generate the artefacts for Virtual Prototyping, done with INTECRIO or EVE [1] [2]. Some tests have been done with Software Sharing code which are not using MDX or AUTOSAR. A non-negligible effort had been spent to adapt the tool and scripts to the code and some information were missing and the way to extract is not as reliable as with an MDX or ARXML file.

4. The ASAM-MDX standard

The ASAM-MDX standard is supported by the tool chains at Bosch DGS-EC for years and the MDX target is part of the Simulink® tool chain. The name of the MDX files is in form of *_mdx.xml. The structure of the MDX files is organized into several parts:

- Data dictionary for the variables: it describes the implementation information such as the Datatype, the Compu Method, the Data Constraints, the Memory Location, the type of communication
- Data dictionary for the parameters: it describes the implementation information of each variable and parameters such as the Datatype, the Compu Method, the Data Constraints, the Memory Location, the Record Layout
- Description of the Compu Method
- Description of the Data Constraints
- The services: for each runnable, it describes the direction for each interface (READ, WRITE or READWRITE) and the type of access. The information provided in this part is very important for the tools which ensure the data consistency
- A list of internal variables, interfaces, parameters and runnable defined in the Software Component
- A list of interfaces and parameters which are exported by the Software Component
- A list of interfaces and parameters which are imported by the Software Component

The definition of the Compu Methods and the Data constraints can be done in self-containment or centrally defined.
In addition to the MDX files, a CDF file can be used to define the default values for the parameters.

The MDX and CDF files are generated by Simulink® with the AddOn MDX Exporter from MathWorks. It is necessary to create MDX elements in the Workspace such as *MDX.Variable* and *MDX.CalPrm*. Some information, which are not present in these Workspace elements like memory location or scope (import, local, export), are defined in the MDX Explorer. It is accessible from the Simulink® model and its content is embedded in the model.
A primary role of the MDX files is to generate automatically the headers files for the data declaration during the software build phase. The headers are enriched with other information. It avoids to generate them during the code generation phase. Hence it permits to have a flexible configuration depending on each project.
Another primary role is to be able to generate automatically an A2l file. The A2l is a standard file from ASAM association which contains the complete data description for the calibration tools like INCA. Without XML files like MDX file, it would be impossible to generate the A2l file with the c code and header file only since they do not contains the Compu Method description for example.

Describing all the data and the scope of each interface in an XML file for every Software Components offers many other advantages, especially with self-containment. During the software build phase, it is indeed possible to do a check if there are open interfaces. The software build can be interrupted in an earlier phase than without this functionality and a log file that provides a list of the issues is generated. It is also possible the check the data definition incoherencies for the interfaces: for example a data type difference between an imported interface and the exported interface.

With the new generation of microcontrollers with Multi-Core technology, the MDX standard provides further advantages for the integrating suppliers regarding the data consistency constraints. With Multi-Core microcontroller, the runnable in the Application Software part can be distributed over several cores. The MDX standard introduces the concept of import and export messages. The data consistency between inputs and outputs communicating between two different cores has to be ensured. In the Bosch DGS-EC software build tool chain, this is done with the tool MCOP. It is a tool included in the DGS-EC

software build tool chain and it uses the READ/WRITE access defined in the MDX files as input to determine which data has to be protected by doing a data consistency analysis. By using MCOP solution with MDX files, there is no need to implement data consistency features in the c code, since the consistency is implemented during the software build process. The consistency implementation is adapted to every software configuration (task distribution and functions scheduling).
It is a mechanism similar to the implicit communication with the AUTOSAR sender-receiver interfaces. The main difference remains in the fact that the implementation of the data protection with MCOP and MDX files is done in the object code and the ELF file with the VARED mechanism during the compilation phase. With AUTOSAR, the data protection is implemented during the RTE generation, in an earlier phase of the software build process.

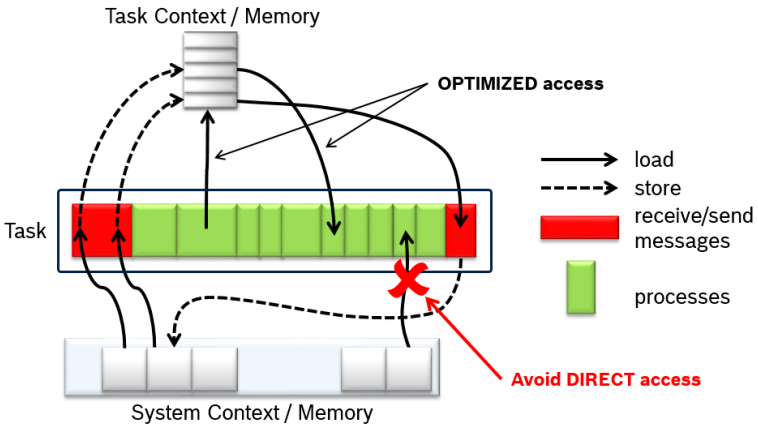

Figure 3 – Bosch MCOP concept

In a Model Sharing context, the support of the MDX standard is depending on the modelling tools and their configurations. With ASCET tool, the Bosch DGS-EC target includes a feature which permits the code generator to generate an XML file called PaVast which has a format and a semantic very similar to the MDX format. Both format are supported by the same tool chain during the software build process. If an OEM wants to exchange ASCET models with Bosch more easily, it is recommended to use a compatible target to avoid the use of a conversion tool.
With the Simulink® tool, the MDX files can be generated with the MDX Exporter from MathWorks. It can be either configured in the OEM Simulink® environment or it is included in the Bosch DGS-EC Add-On which offers a complete code generation environment compatible with the Bosch DGS-EC code development rules. This Add-On can be delivered to any OEM in the framework of a cooperation contract. The cooperation between OEM and supplier when exchanging models can be eased when both development partners are using the same MDX standard for their Model Based Development tool chain. It can be a blocker when exchanging library blocks for the code generation. Some library blocks, like filters or flip-flop, may include memory states. For example, the memory states can be described with the Workspace elements *MDX.Variable*. If the user of a Simulink® model who need this library does not have the MathWorks MDX Exporter, he shall do with a script a conversion of MDX Workspace elements into Simulink® elements in order to be able to run a simulation.
With a Model Sharing Level 1 or 2, if he OEM delivers Simulink® models without any MDX support, the MDX configuration can be added by the supplier. The adaptation is done in two steps. The first step is dealing with a conversion of the OEM data dictionary into MDX elements in the Workspace. The conversion task can be eased if the OEM is able to provide a data dictionary in MDX format. It can be more easily converted into MDX elements since the XML format can be processed more easily than Excel format and is less sensitive to issues with bad fields' content with bad characters.

The second step is dealing with information which are not present in these Workspace elements like memory location or scope (import, local, export). They shall be defined in the MDX Explorer which is accessible from the Simulink® model and these data are embedded in the model. The filling of the MDX Explorer can be done more easily if the OEM is able to provide a data dictionary in MDX format containing the useful information like the scope (import, local, export). The information can also be completed with Matlab scripts through specific API.

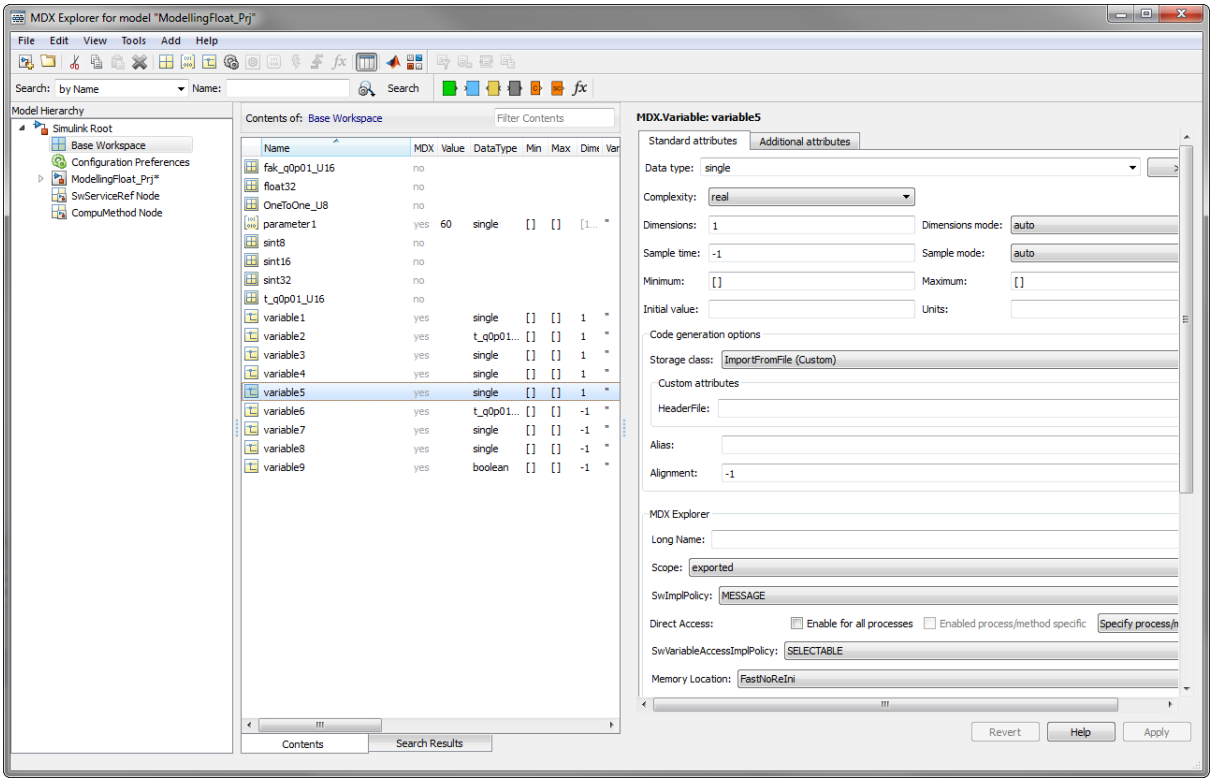

Figure 4 – Simulink MDX Exporter

With the use of XML files for the support of a standard like MDX, it is strongly recommended to use an automatic code generator. The MDX files contain in general hundreds of lines of code with tags.

5. The AUTOSAR standard

Bosch DGS-EC is working on an efficient use of the AUTOSAR standard in a scope of development for series. It is a more complete standard than the MDX standard since the Basic Software is standardized and clearly decoupled from the Application Software. It describes the concept of Virtual Functional Bus which is supported by the Run Time Environment (or RTE). In the same way as the MDX standard, the data are described in a XML file, it is called ARXML (AUTOSAR XML). It offers a much more complete semantic and is based on AR Packages and AUTOSAR element references. Contrary to the MDX standard, the infrastructure interfaces such as diagnostic, NVM, DCM interface are supported as standardized Basic Software interfaces. It is a main advantage in comparison with MDX since the BSW interfaces can be implemented in the models and described in the ARXML files. With Software Sharing context, it avoids using implementing Sender-Receiver interfaces in the Software Components and developing Customer Interface Layers to adapt these Sender-Receiver interfaces to the integrating supplier BSW specific interfaces.
Another advantage of the AUTOSAR standard is dealing with standardized AUTOSAR service libraries. These service libraries have standardized names and argument names. Their algorithms are also precisely described so that they can be implemented by any integrating supplier. AUTOSAR service routines do not use AUTOSAR mechanisms: they can be used for developments with MDX standard as well as AUTOSAR.
With the Simulink® tool chain, Bosch DGS-EC is able to deliver Simulink® library blocks supporting the AUTOSAR service libraries, since they are not all available with the native MathWorks blocks.

AUTOSAR is supported by many tools from the market, this is an important advantage since the in-house development of specific software development tools can be avoided. If an OEM is working with AUTOSAR, then there is no need for its integrating suppliers to develop specific tool. It would be very costly if every OEM and supplier had their own standard. AUTOSAR has changed the landscape of the manufacturer - supplier relationship. With this standard, it can much easier than ever to integrate the code of an OEM or a Third Party into an integrating supplier software. It is a mean to ease the Software Sharing and open new cooperation frameworks. The structure of the ARXML file is however not fixed as for MDX: the structure ordering is depending on the AR-Package structure and AR-Packages name. The AR-Packages can be spread over several ARXML files and a Software Component can be

described by several ARXML files, for example one ARXML file for each AR-Package. The elements properties are not defined only by short name, but by reference to the AR-Packages.

The main automotive modeling tools ASCET, TargetLink® or Simulink® are supporting AUTOSAR. With ASCET, the concept of AUTOSAR Model has been introduced in the tool. It is very similar to the non-AUTOSAR models, especially regarding the modelling of the functionalities. From an architecture point of view, an atomic Software Component is represented by one ASCET module. A composition can be represented in the ASCET Project by an assembly of ASCET modules.

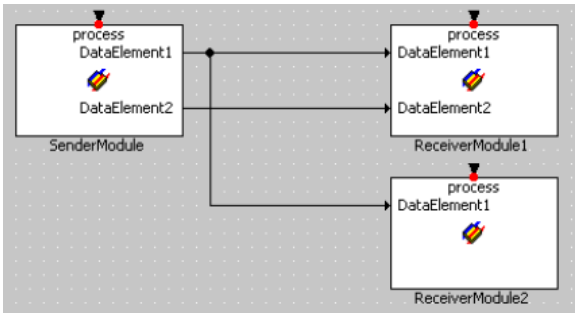

Figure 5 – Composition in ASCET

Instead of Import / export message blocks, there is a specific block available for the Sender/Receiver interfaces and the type of communication (explicit VS implicit) can be selected.

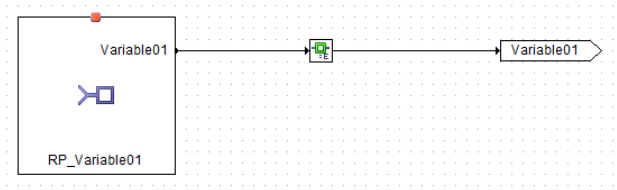

Figure 6 – AUTOSAR ports modelling in ASCET

An additional tab has been introduced to configure the runnable to events mapping. The naming convention for the AR-Package and many AUTOSAR elements are configured in an XML configuration file.

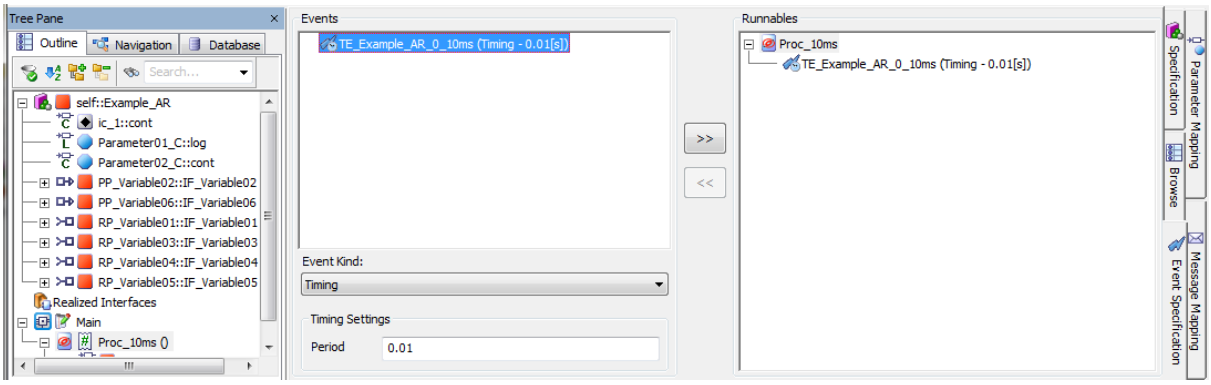

Figure 7 – ASCET runnable to event mapping

For the development of AUTOSAR Software Components, a cooperation between MathWorks and Bosch has been established within the framework of a strategic partnership in order to support AUTOSAR with Simulink® in an efficient way. At first glance, a Simulink® model for AUTOSAR looks like a model for MDX. There are three main differences. The first one affects the Workspace elements: it is necessary to create AUTOSAR elements such as *AUTOSAR.Signal* and *AUTOSAR.Parameter*. ARXML files can be imported through an API to load for example the Internal Behavior or central elements for the central definition of AUTOSAR elements such as base types or Compu Method.

The second difference is dealing with the AUTOSAR Mapping Editor which contains two sheets: the ports, the interfaces (Sender/Receiver, Client/Server), the IRV and the runnables are defined in the first sheet. Their mappings to the models inports, outports and the function call is ensured in the second sheet. The content of the AUTOSAR Mapping Editor can be filled automatically with a script thanks to

APIs. The script shall be configured to fit to the naming convention applied in the environment development. For example, a Receive Port could have a prefix “R_” or “RP_”. This automation is possible if the ports, interfaces and data elements share the same name root. This is a rule established at Bosch DGS-EC and it allows to do also an automation of the AUTOSAR composition, called “Autocomposition”.

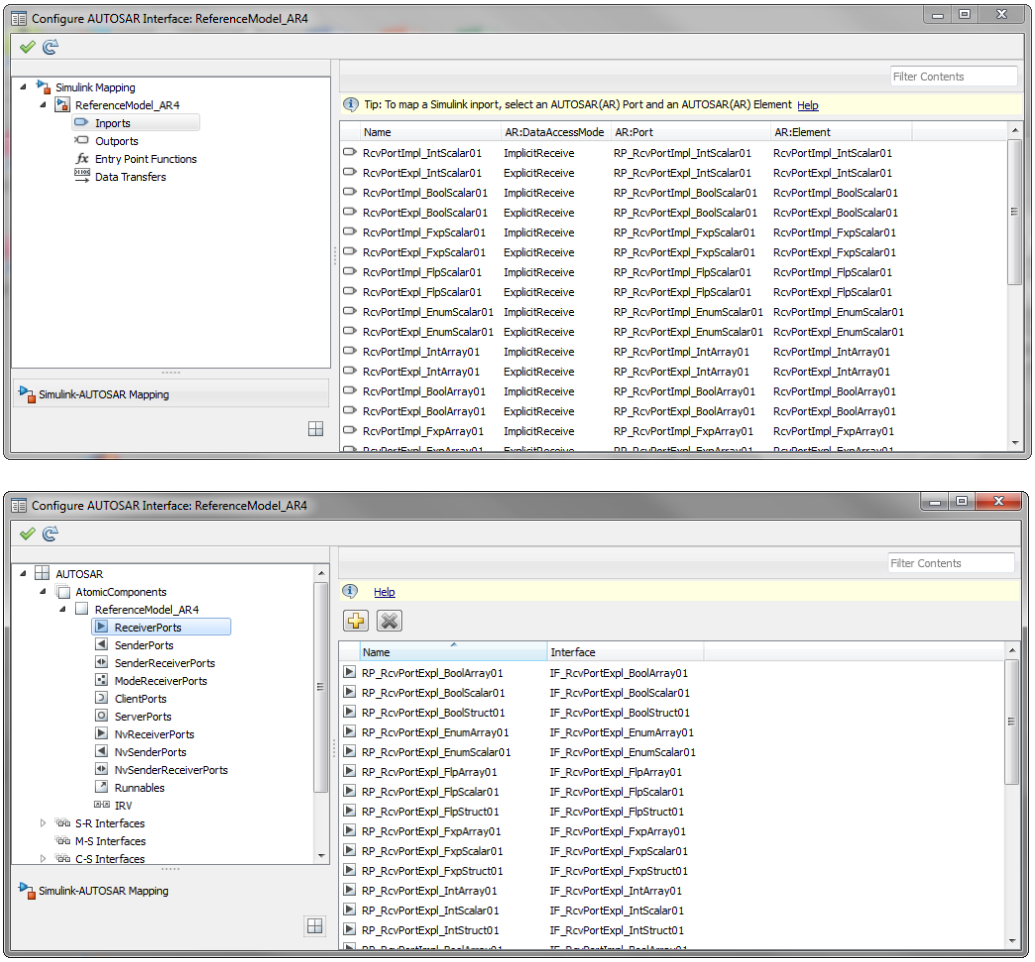

Figure 8 – Simulink AUTOSAR Mapping Editor

The third difference is dealing with the accesses to the infrastructure: there are described with Client-Server interfaces and many of them have standardized names. “Simulink® Function” and “Simulink® Caller” blocks can be used.

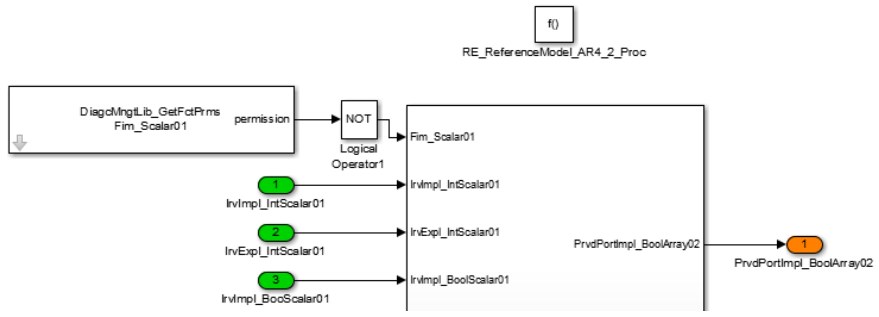

Figure 9 – AUTOSAR FiM interface modelling in Simulink

The software build process has to be adapted to an AUTOSAR project: in comparison with non-AUTOSAR project, a tool called RTE Generator shall be invoked. It generates c and h files from the ARXML abstraction layer. It generate the RTE which ensure the communication. In most of the projects; there is a mix-mode with a part of the software in MDX or any other specific OEM standard. The implementation of Customer Interface Layers is required and is a drawback to a smooth transition to

AUTOSAR in comparison to a “big bang” introduction. It requires then additional development effort and additional RAM consumption. There is also an impact on the OS scheduler: a pure AUTOSAR scheduler can not be implemented. The legacy OS scheduler references RTE task container tasks which are described in a configuration ARXML.

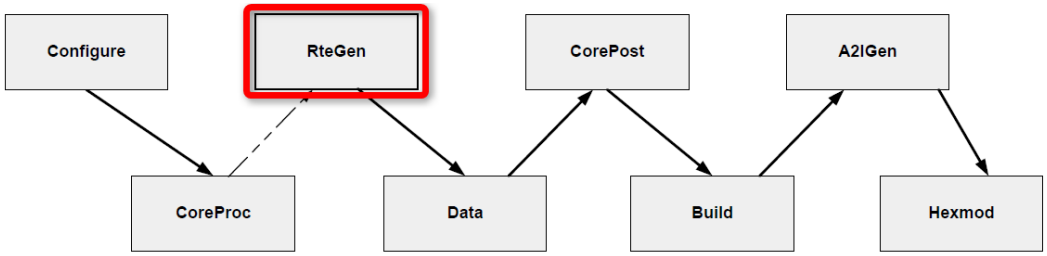

Figure 10 – Bosch Software build Process

The implicit Sender-Receiver communication is recommended to ensure the data consistency for the communication between two different cores in a Multi-Core microcontroller or between two different tasks with a pre-emptive Operating System (OS). Buffers are created during the RTE generation in the c and h files. This is done indeed before the compilation phase and it does not require specific tools as for MDX standard, since the protection mechanisms are described in the standard and supported by the RTE Generators.

In a Model Sharing context, the tools ASCET, TargetLink® or Simulink® offer the possibility to develop AUTOSAR Software Components without spending so much time on the target configuration. With ASCET tool, an AUTOSAR target configuration is available and the AUTOSAR model can be quite easily exchanged between OEM and suppliers. It is however very important to keep in mind that some AUTOSAR features availability is depending on the AUTOSAR release. All the partners in a cooperation framework shall agree on the supported release before starting the development to avoid model incompatibilities. In the same way, it is very important for an OEM or a third party to agree on the AUTOSAR Release supported by the RTE Generator. They have to be sure that the required AUTOSAR features in the models are supported by the RTE Generation version in use for the project.
With the Simulink® tool, the same precautions have to be taken regarding the AUTOSAR release compatibilities. As an example, the PRPortPrototype feature has been introduced AUTOSAR 4.1.1. If the development partners are working with different Simulink® versions, this feature may not be available in an older version and there would be an issue in the AUTOSAR Mapping Editor Content. With the same example, this feature could be generated in the ARXML but it is not sure whether the RTE Generator version in use at the integrating supplier side would support this feature. Developing an AUTOSAR Software Component has an impact on the modelling patterns. For example, multi-triggering on runnable level is not possible, since one runnable shall be mapped with only one event. The standardization of the BSW interfaces is a great enabler for Model Sharing. An obvious example are the DEM / FIM diagnostic interfaces. In comparison with MDX, every development partners can use the same service access and there is no need to develop customer specific services on the integrating supplier side. As a drawback it reduce the possibilities to bring innovations in these services and the OEM and suppliers have less levers to differentiate their technology from their competitors.

Shared software development for electronic control units becoming more and more important. In order to cope with the complexity of the Software development for Engine Control, a strong and highly flexible Software architecture is required. The answer of Bosch to this challenge is the creation of the Software architecture VeMotionSAR™ that strengthens the cooperation with the OEMs and that has been published in Internet (http://www.bosch-vemotionsar.com). One of the main benefits of this architecture is the modularity of the function description and the definition of the interfaces. It permits to describe individually the various specifications of the corresponding development partners’ functionalities in a highly modular architecture domain according to their resources / tools.

6. Migration from legacy standard to MDX

MDX standard is mature and in use at Bosch DGS-EC for years for series project. With the new Multi-Core microcontrollers and the need of data consistency mechanisms, it can be a good alternative for the integrating suppliers who are using specific standards which are not supporting Multi-Core. It is therefore interesting to analyze the impacts and the constraints on Model Sharing cooperation’s when

doing a migration to the MDX standard. AUTOSAR is a promising standard in the automotive embedded software and more and more control units have an AUTOSAR RTE with a partly or complete AUTOSAR Application Software. The description of migrations uses cases are focused on Simulink® tool in this chapter.

For the conversion of Simulink® models to MDX, the main challenges are oriented to the migration of the Workspace elements and the configuration of the MDX Explorer. The Workspace elements can be easily converted to MDX elements with Matlab scripts. Some information contained in the Workspace elements could be lost if the original models are using Custom Storage Class. Depending on the features, they can either be taken into account in the MDX Explorer or may lead to some changes in the model. For example the attribute to know if a variable shall be stored in the Non Volatile Memory can be set in a Custom Storage Class. During the migration, this attribute shall be taken into account in the MDX Explorer by setting the memory location to a Non Volatile Memory location. This example demonstrates that the MDX Explorer content shall be updated at the same time as the Workspace elements. Since all the MDX Explorer attribute can be accessed with APIs, a m file for the MDX Explorer configuration can be created during the migration and running it will fill the MDX Explorer content. This is the solution chosen at Bosch for converter scripts.
The migration to the MDX standard may have also an impact on the modeling pattern in a Simulink® model. A frequent issue is faced for inport which may uses the value of the outports in the same runnable. It is not possible to have the same name for the inport and the outport, the inport shall be removed and the signal shall be replaced by a Unit Delay block connected to the outport. It is more complicated to handle it if the same outport is also initialized in a separate initialization runnable. For this case, a specific UnitDelay block, which includes a memory state value defined in the Workspace as a *MDX.Variable,* has been created at Bosch DGS-EC. Another impact on the modelling patterns is dealing with the Multi-Core constraints: it is not possible to keep the multi-triggering on runnable level. The functionality need to be refactored or the runnable content needs to be duplicated. The duplication may be complicated in case of state machine or any functions containing state values.

7. Migration from legacy standard or MDX to AUTOSAR

For the conversion of Simulink® models from legacy or MDX to AUTOSAR, the main challenges are oriented to the migration of the Workspace elements and the configuration of the AUTOSAR Mapping Editor. The most important challenge for developments with AUTOSAR is to be as efficient as the standard it is supposed to replace in a defined tool chain. For ECU Software development with its complexity, it is reasonable to consider that it would not lead to short-term cost reduction. However, it could avoid spending more efforts in long-term by reducing the development and maintenance effort of specific tools n case an OEM or a supplier is using its own standards . A migration to AUTOSAR makes sense if the tool chain is ready for efficient development with AUTOSAR from an economic point of view.

The experience shows that it is necessary to write some scripts to reduce the manual effort and the risk of error. The advantage of Model Based Development lies in the fact that the AUTOSAR adaptation can be done directly in the model used for the code generation. Prior to the migration task, many clarifications with the OEM are necessary such as:

- Architecture: one model = one atomic SW-C, or one model = one Runnable ?
- AR-Package structure
- Naming conventions for the AR elements short names
- Restriction on the naming rules for the ports, interface and data elements to allow an autocomposition

The Workspace elements can be converted into AUTOSAR elements with Matlab scripts. At Bosch DGS-EC, the measurement points are defined as Per Instance Memory (PIM) whereas they are *Simulink.Signal* or *MDX.Variable* with other targets where the inputs and outputs are also *Simulink.Signal* or *MDX.Variable*. The script shall detect the inputs and the outputs in the models in order to identify which *Simulink.Signal* or *MDX.Variable* shall be converted into *AUTOSAR.Signal* and the other into PIM. The AUTOSAR Mapping Editor can filled automatically with a script.
There are also impacts on the modelling patterns:

- The multi-triggering on runnable level is not compatible with AUTOSAR: with a Simulink® model, it is not possible to have a multi-triggering on the Function Call subsystems representing the runnable since one runnable shall be mapped to ne event.
- The messages exchanged between runnable shall be converted as Inter Runnable Variables (IRV).

- With Simulink, the measurement points are defined in MDX as global variable with a resolve signal to Workspace: in AUTOSAR they shall be defined as static memory or Per Instance Memory (PIM).
- At Bosch DGS-EC, many legacy services can not be mapped one to one to AUTOSAR standardized client-server interfaces. In case of software sharing, these interfaces shall be replaced by Sender-Receiver interface and the integrating supplier shall map them to its equivalent services via an adapter.
- the physical system constants are not supported by AUTOSAR, a work-around is necessary

In order to prepare a seamless migration to AUTOSAR for mid or long term, some measures can be taken in the modelling pattern rules for the legacy models to ensure a compatibility with AUTOSAR. The modeling pattern shall define so that a model could converted automatically into AUTOSAR without any rework.

- For the interpolation, it necessary to have blocks which support MDX and AUTOSAR at the same time. The Simulink® look-up table will support soon both MDX and AUTOSAR standard with AUTOSAR R4.x libraries Ifx and Ifl.
- For the Dem and FiM interface in AUTOSAR, if similar services are available in legacy environment, it could be possible to create Simulink® blocks with some an automatic configuration inside, depending on the choice between the legacy or AUTOSAR target.
- Legacy services which are not identical with the AUTOSAR similar service require to define modeling pattern adaptations.

Summary

Model Sharing is currently applied at Bosch for several serial projects and is leading to clear improvements in development efficiency. Use of Standards (e.g. ASAM (MDX, CDF …) AUTOSAR (service libraries, methodology …) as well as standardized interfaces specifications (e.g. as published with AUTOSAR R4.x) can significantly ease exchange of models.

MDX standard is supported by the code generation tool chains for years at Bosch and permits to perform interfaces coherency check during the Software Build process as well as to provide information to build the A2l file. With the Multi-Core microcontrollers, the MDX files are used by the tool MCOP to ensure the data consistency.

The AUTOSAR standard clearly defined a separation between the Basic Software and the Application Software and provides standardized interfaces with standardized mechanism. In comparison with MDX, the infrastructure interfaces with the Basic Software are standardized which is a lever to improve the model exchanges. For engine control unit software, the Process, Method and Tool for AUTOSAR developments are not yet as efficient as with MDX development. Bosch is working actively with the tool editors to reduce the gaps. On one side, Tier 1 need to know if the OEM want to base their future development with the AUTOSAR standard. On the other side, OEM want to know when the Tier 1 will be able to support AUTOSAR in an efficient way. The approach at Bosch has been to edit internal AUTOSAR coding guidelines. The current tasks are focused on the identification and the reduction of gaps between those guidelines and what is supported by the tools.

It is possible to migrate a legacy Simulink model to MDX or AUTOSAR. Many migration tasks can be automatized with scripts. The modelling patterns are also impacted and if the migration to AUTOSAR is a long-term strategy, it is recommended to take into account this modelling constraints in the legacy models in short term.

Bibliography:

1. Dr. M. Tanimou, Dr. M. Münzenmay, K. Zimmermann, Robert Bosch GmbH; Dr. B. Lumpp, Dr. E. Trapel, E. Bouillon, Dr. M. McMackin, MAN Truck & Bus AG: Software-in-the-Loop durch Co-Simulation von EDC-Modell und GT-Modell; Mainz, 14th MTZ-Conference 2014-01-0189 - virtual powertrain creation, 2013
2. Lumpp, B., Tanimou, M., McMackin, M., Bouillon, E. et al., "Desktop Simulation and Calibration of Diesel Engine ECU Software using Software-in-the-Loop Methodology," SAE Technical Paper 2014-01-0189, 2014, doi:10.4271/2014-01-0189
3. S. Louvet, Robert Bosch (France) SAS, Dr. U. Niebling, Dr. M. Tanimou, Robert Bosch GmbH : Model Sharing to leverage customer cooperation in the ECU software development; Toulouse, ERTS 2014-Conference, 2014