Improving the Interoperability of Automotive Tools by Raising the Abstraction from Legacy XML Formats to Standardized Metamodels

Mark Brörkens and Matthias Köster

Carmeq GmbH, Carnotstr. 4, 10587 Berlin, Germany
{mark.broerkens,matthias.koester}@carmeq.com

Abstract. Automotive system design demands frequent exchange of data between different parties and tools. In order to improve the interoperability, standardization bodies and partnerships have put high effort in defining XML based languages for system descriptions.

However, the mere existence of a standardized XML based data exchange format doesn't guarantee seamless interoperability. The validation possibilities given by XML DTD or Schema are not sufficient. Additionally, the maintenance of XML formats for the growing complexity of today's systems is an increasing challenge.

This paper describes the experiences with the model-driven approach taken by the automotive initiative AUTOSAR. It illustrates the limitations of designing data exchange formats in XML and shows how a higher level of abstraction increases the interoperability between tools. A powerful concept for mapping a metamodel to XML schema allows for integrating legacy XML formats.

Furthermore, current activities on improving interoperability by automatically generating a tool framework for AUTOSAR and other automotive tools are explained.

1 Introduction

The development of an automotive electric/electronic system is distributed over many parties. Whenever information is exchanged, the involved parties need to agree on common syntax and semantics. While exchanging data via informal documents often was sufficient in the past, the complexity of todays automotive systems [1][2] requires a more formal machine readable exchange of information.

An important step for improving the interoperability between different parties is to standardize a common domain-specific language including its syntax and semantics. Since all involved tools must be able to read and write this language, using XML (eXtensible Markup Language)[3][4] is a good idea: Tool vendors can use high quality off-the-shelf XML libraries for XML processing and validation.

The automotive initiative AUTOSAR (AUTomotive Open System ARchitecture) [5] has started to standardize a data exchange format that covers several steps in automotive system development including abstract description of

D.H. Akehurst, R. Vogel, and R.F. Paige (Eds.): ECMDA-FA 2007, LNCS 4530, pp. 59–67, 2007.

software components as well as fine-grained mapping of bits to frames on communication networks. Section 2 describes the (meta)model based approach for designing and generating the AUTOSAR XML schema.

For a seamless interoperability between tools, additional consistency checks that exceed the expressive power of W3C XML schema [4] and alternative XML schema languages such as W3C DTD [3] or RELAX NG [6] are required. These constraints are often defined informally and therefore are likely to be interpreted differently by various tool vendors.

Section 3 describes the layered architecture for tool interoperability defined by AUTOSAR. This architecture defines a hierarchy of abstraction levels and their impact on tool interoperability.

The metamodel developed by AUTOSAR can not only be used to generate the XML schema or documentation, it can also be used as a source for the generation of a tool platform. Since AUTOSAR itself is not developing tools, some AUTOSAR members have formed the informal OTF for Automotive [1] initiative. This initiative plans to develop an open platform that can be used by any tool vendor. The current status of that initiative is presented in section 4.

2 Improving Interoperability by Standardization of Data Exchange Formats

2.1 Standardisation on XML Level

Standardization bodies and partnerships have put high effort in defining XML based data exchange formats for specific use cases. Some examples are:

- ASAM FIBEX [7] (Field Bus Exchange Format) focuses on message-oriented bus communication systems. It allows to describe bus configuration, parameterization, design, monitoring and simulation.
- MSR SW [8] focuses on the description of automotive software

Those data exchange formats usually are defined by a formal XML DTD or XML Schema and informal descriptions of constraints which exceed the expressive power of generic XML validation techniques.

The XML DTDs or XML schema specified in these standards are created manually or are assembled semi-automatically out of several XML fragments for better maintainability. The availability of these XML formats highly improves the interoperability and reduces development costs of tools: Generic off-the-shelf XML libraries can be used to parse and validate XML descriptions. However, this manual approach of creating a XML-based data exchange format is not sufficient for the needs within the AUTOSAR development partnership:

- Complex interlinked structures are very *hard to understand and maintain* at the XML level.

[1] Open Tool Framework for Automotive.

- Conceptual discussions on the AUTOSAR language are often interrupted by *discussions on how to implement the concepts in a XML schema language.*
- The *expressive power* of XML schema languages such as XML DTD or XML schema is *not sufficient* for seamless tool interoperability. Additional constraints need to be defined separately and are (if not defined using a formal unambiguous language) a potential risk for misinterpretations and therefore are a risk for limited interoperability.

For reasons mentioned before, AUTOSAR decided not to create the XML-based exchange format by directly editing XML schema descriptions. Instead, the implementation of the language in XML was de-coupled from discussions on the actual content. This was achieved by raising the abstraction from XML to a metamodel.

2.2 Standardization on Metamodel Level

AUTOSAR started modeling the AUTOSAR language using class diagrams. The increasing number of developers and growing complexity of the model required a modeling guide. As described in "Definition and Generation of Data Exchange Formats in AUTOSAR" [9] , the focus of that modeling guide is to support the automotive experts in modeling their domain knowledge in the AUTOSAR metamodel. Individual developers do not need to delve into UML modeling techniques. This approach has the following advantages:

- The graphical representation allows for quickly getting a good overview, even over highly interlinked information.
- The experts can concentrate on their domain knowledge and do not need to care about the implementation in an XML schema language
- The replica mechanism of the UML tool Enterprise Architect [10] allowed for distributed development of several experts on the same model.

AUTOSAR delegated the implementation of its XML schema to a small group of modeling and XML experts. This group defined powerful rules for mapping the AUTOSAR metamodel to XML schema. These mapping rules had to fulfill the following requirements:

- Allow for strong validation using standard XML parsers
- Ability to reproduce existing XML structures and patterns that are well established in the automotive domain. (See for example [11]).
- Configuration of mapping rules based on tagged values in the metamodel

Even though the AUTOSAR language is modeled using UML class diagrams, some constraints can't be expressed: A more powerful constraint language is required.

Example for a Model Constraint. In the AUTOSAR metamodel, the class `ElectricalRange` represents an electrical range for different applications. This class has attributes `minVoltage` and `maxVoltage` that specify the range in voltage (see figure 1 for details). It also has the attribute `typicalVoltage`, which must be a voltage in the specified range. This constraint can't be expressed with XML schema. But it can be expressed easily with an OCL constraint (Object Constraint Language) [12]:

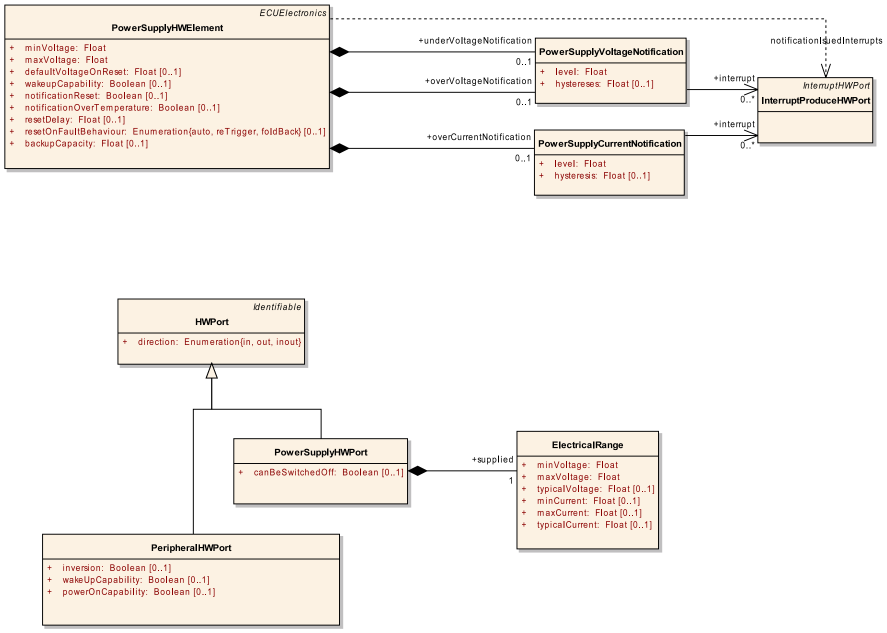
ECUElectronics
PowerSupplyHWElement
+ minVoltage: Float
+ maxVoltage: Float
+ defaultVoltageOnReset: Float [0..1]
+ wakeupCapability: Boolean [0..1]
+ notificationReset: Boolean [0..1]
+ notificationOverTemperature: Boolean [0..1]
+ resetDelay: Float [0..1]
+ resetOnFaultBehaviour: Enumeration{auto, reTrigger, foldBack} [0..1]
+ backupCapacity: Float [0..1]
+underVoltageNotification
0..1
+overVoltageNotification
0..1
+overCurrentNotification
0..1
PowerSupplyVoltageNotification
+ level: Float
+ hystereses: Float [0..1]
PowerSupplyCurrentNotification
+ level: Float
+ hysteresis: Float [0..1]
notificationIsuedInterrupts
+interrupt
0..*
+interrupt
0..*
InterruptHWPort
InterruptProduceHWPort
Identifiable
HWPort
+ direction: Enumeration{in, out, inout}
PowerSupplyHWPort
+ canBeSwitchedOff: Boolean [0..1]
+supplied
1
ElectricalRange
+ minVoltage: Float
+ maxVoltage: Float
+ typicalVoltage: Float [0..1]
+ minCurrent: Float [0..1]
+ maxCurrent: Float [0..1]
+ typicalCurrent: Float [0..1]
PeripheralHWPort
+ inversion: Boolean [0..1]
+ wakeUpCapability: Boolean [0..1]
+ powerOnCapability: Boolean [0..1]

Fig. 1. AUTOSAR 2.0 metamodel for classes `ElectricalRange` and `PowerSupplyHWPort`

```
context ElectricalRange

inv self.minVoltage <= self.typicalVoltage and
  self.typicalVoltage <= self.maxVoltage
```

Elaborating the example above, the class `PowerSupplyHWPort` describes power supplying and consuming hardware ports. If the `direction` attribute is set to `in`, the port is a power supplier. If `direction` is set to `out`, the port consumes power. A `PowerSupplyHWPort` has a pin of type `HWPin`, which can be connected via a `PinHWConnection` with another port. Connecting two such pins requires that an out and an in pin are connected and that both have a compatible electrical range. This can be checked with the following OCL constraint:

```
context PowerSupplyHWPort

let out =
 PinHWConnection.allInstances()
   ->select(conn | conn.connectedPin->includes(self.pin))
    .connectedPin->excluding(self.pin).port

inv self.pin.direction = 'in' implies
 out.direction = 'out'
 and
 self.electricalRange.minVoltage = out.electricalRange.minVoltage
 and
 self.electricalRange.maxVoltage = out.electricalRange.maxVoltage
```

Breaking these constraints can cause severe damage, thus it's especially important for an OEM (Original Equipment Manufacture) to ensure that the AUTOSAR XML descriptions from different suppliers describe a consistent system. This can be reached by adding these constraints to the metamodel and validating them when assembling the system description from different suppliers.

Although AUTOSAR has started using OCL for formal descriptions of constraints in the metamodel most constraints are still described informally. A more extensive use of OCL instead of informal descriptions would help to increase the interoperability.

3 Layered Architecture for Tool Interoperabilty

Figure 2 shows a layered architecture for tool interoperability. On the first layer, files are used to exchange data between different tools. On the next layer the data format is specified. Using XML as the file format allows easy exchange of data, but doesn't ensure the referential integrity of the data. On the next layer the content can be accessed as interlinked data structures and referential integrity can be reached, but the semantic consistency of the data can't be easily expressed or checked. On top of the content layer is the semantic layer, where the data consistency can be checked by validating constraints on the content. Applications built on top of this layer can assume that the data, on which they operate, is consistent. Regardless of the intended use all tools must implement the lower four layers.

The upper-most levels focus on how tools present information to the users and how the business logic is implemented. Those levels are highly dependent on the intended use and therefore are only mentioned for the sake of completeness.

The base for any AUTOSAR tool is its ability to persist data as XML files. Working with XML makes it possible to use generic XML validation techniques.

Representing the data as instances of metamodels (content layer) supports the use of standard model APIs (Application Programming Interface) to access and query the data. Rather than working with XML and textual links between

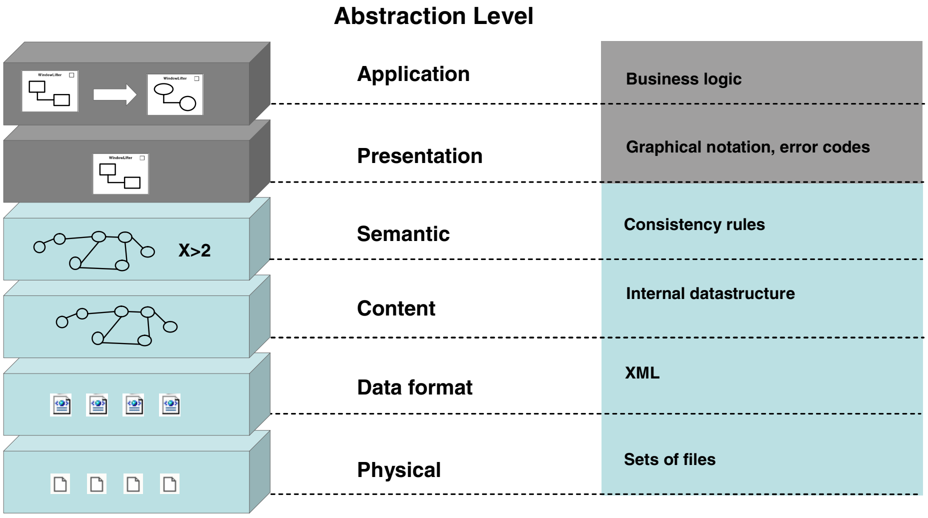

Fig. 2. Layered architecture for tool interoperability

XML elements, we are now working with references and links between classes and objects. This makes it much easier to ensure the referential integrity of the data. A higher level of abstraction is reached, making it easier to access and manipulate the data.

Using a reflective standard model API makes it is possible to build a generic validation layer on top of the model layer. This enables us to leverage the full power of model based validation techniques (e.g. OCL[12]).

4 OTF – a Layered Framework for Tool Interoperability

The Open Tool Framework (OTF) is an implementation of the layered architecture as described in the previous section. OTF is realized as a set of Eclipse plug-ins and uses the Eclipse Modeling Framework (EMF) [13] as its foundation. EMF is an open source implementation of the OMG EMOF 2.0 standard [14] and defines a mapping from EMOF to Java. This mapping can be used to generate a metamodel specific API, but EMF also provides a reflective model API. Other frameworks can use the reflective model API to provide generic services on top of EMF. A transaction layer and several OCL validation engines are available as open source.

EMF also provides an extensible framework for loading and saving models. This framework is based on the concept of resources, which consist of a set of model elements that should be stored at the same physical location. To specify the physical location of a resource, a Universal Resource Identifier (URI) is used. This rather abstract and generic approach allows support for a lot of different storage system. EMF provides persistence to XMI (XML Metadata Interchange)

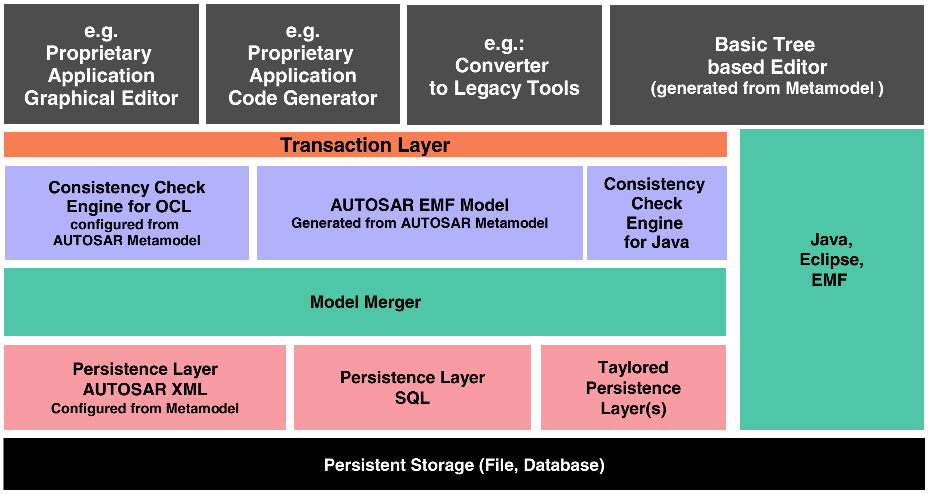

Fig. 3. OTF: a layered framework for tool interoperability

[15] and XML file formats, but other open source frameworks are available that provide persistence to RDBMS (Relational Database Management System).

EMF provides extension points to add custom resource implementations. This mechanism allows for integrating custom resources for the AUTOSAR XML format. The rules for mapping between AUTOSAR XML and the AUTOSAR metamodel are defined in the AUTOSAR "Model Persistence Rules for XML" [5] specification and are implemented by the OTF persistence layer. Since we extended the standard EMF resource mechanism, all available frameworks based on EMF and its resource system can be leveraged.

Figure 3 shows the architecture of the OTF framework. The lower three levels of tool interoperability are realized by the persistence layers and the model merger. The persistence layers transform AUTOSAR data from various representations into an instance of the AUTOSAR metamodel. The model merger merges data from several sources and detects redundant or conflicting information. Consistency checks are supported using the EMF validation mechanism. Constraints can either be defined as OCL expressions which are interpreted by the EMF OCL validation engine or by Java classes which navigate over the AUTOSAR EMF model. Read and write requests from application plugins (e.g. graphical editors, code generators, converters) are coordinated by the transaction layer.

5 Summary and Outlook

The model-driven approach for designing and generating the AUTOSAR XML schema was very successful. More than 30 domain experts have been actively working on the AUTOSAR metamodel in the last 3 years. Today the metamodel contains more than 600 classes and more than 1300 structural features. This complexity would not have been possible by manually creating a XML schema. Other automotive standardization organizations are already adopting this model-based approach. Future versions of the MSR Software [8] are likely to use the AUTOSAR concept. Additionally the Requirements Interchange Format specified by the Hersteller Initiative Software [16] is using the AUTOSAR approach.

Today the AUTOSAR metamodel is not only used as the source for generating the AUTOSAR XML schema, additionally it is used as the single source for a substantial amount of documentation.

Furthermore, the OTF initiative has shown that the metamodel can be used to generate an implementation of a tool platform that conforms to the designed domain specific language. This generated platform provides features such as reading, writing, merging and validating automotive system descriptions. Tool vendors can build their tools on top of the OTF.

We hope that more standardisation bodies will use the approach outlined in this paper, since this can improve the interoperability of tools dramatically.

References

1. AUTOSAR: Media release. (2006), http://www.autosar.org/download/AUTOSAR_long_en.pdf
2. Fennel, H., et al.: Achievements and exploitation of the AUTOSAR development partnership. (2006), http://www.autosar.org/download/AUTOSAR_Paper_Convergence_2006.pdf
3. W3C: Extensible markup language (xml) 1.1 (2 edn.). (September 2006), http://www.w3.org/TR/2006/REC-xml11-20060816/
4. W3C: Xml schema part1: Structures second edition. (October 2004), http://www.w3.org/TR/xmlschema-1/
5. AUTOSAR: Official website of the autosar development partnership. (2006), http://www.autosar.org
6. ISO: ISO/IEC FDIS 19757-2 document schema definition language (dsdl) – part 2: Regular-grammar-based validation – relax ng. (December 2002), http://www.relaxng.org/#specs
7. ASAM: FBX, FIBEX - field bus exchange format, version 2.0. (June 2006), http://www.asam.net/03_standards_06.php
8. ASAM: MSRSW, MSR software, version 2.3. (June 2005), http://www.asam.net/03_standards_10.php
9. Pagel, M., Brörkens, M.: Definition and generation of data exchange formats definition and generation of data exchange formats in autosar. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, pp. 52–65. Springer, Heidelberg (2006)
10. Sparx Systems: Enterprise architect product page. (2006), http://www.sparxsystems.com.au, http://www.sparxsystems.com.au/
11. ASAM: HDO, harmonized data objects, version 1.0. (July 2004), http://www.asam.net/03_standards_09.php
12. OMG: UML OCL specification version 2.0. (June 2005), http://www.omg.org/cgi-bin/doc?ptc/2005-06-06
13. Eclipse Foundation: Website of the EMF eclipse project. (2006), http://www.eclipse.org/emf

14. OMG: Meta Object Facility (MOF) specification version 2.0. (January 2006), http://www.omg.org/cgi-bin/doc?formal/2006-01-01
15. OMG: XML Metadata Interchange (XMI) specification version 2.1 (September 2005)
16. Automotive HIS: Requirements interchange format, version 1.0a. (November 2005), http://www.automotive-his.de/simutool.htm