Integrating the AUTOSAR tool chain with Eclipse based model transformations

Andreas Graf, Markus Völter

Integrating the AUTOSAR tool chain with Eclipse based model transformations

Andreas Graf[1],Markus Völter[2]

1: itemis GmbH, Blücherstr. 32, 75177 Pforzheim, Germany
2: independent/itemis GmbH, Hohnerstr. 25, 70469 Stuttgart, Germany

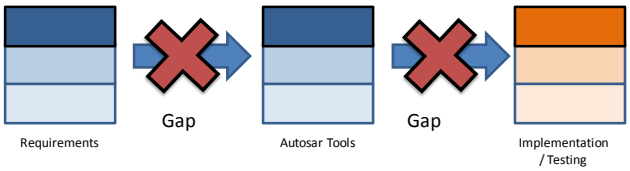
Extended Abstract for Application:

Keywords: AUTOSAR, Eclipse, Open Source, model transformation, Artop, Model Driven Engineering, AUTOSAR tool chain, software development process, AUTOSAR migration, automotive software, standardization, Engineering tools integration and interoperability, Open source, open systems.

1. Introduction

AUTOSAR is establishing itself as a prominent standard in automotive systems and is expected to significantly improve software architecture and the software development processes. However, the introduction of AUTOSAR also poses some challenges.

AUTOSAR only defines part of the development process, it does not specify how to interface tools in preceding process steps (such as functional design or requirements engineering) and to subsequent steps (such as testing or the actual implementation of the software).



In addition, AUTOSAR currently does not cover all aspects of automotive software. AUTOSAR software needs to communicate with non-AUTOSAR software, for example in the infotainment domain or even with off-board software.

In addition to these conceptual challenges, the availability of tools can be a challenge, too. Similar to the early phase of UML, AUTOSAR tools are rather new and immature, and experience with those tools is not widespread. Tool features might be lacking and the integration of AUTOSAR tools into a company's overall tool chain will definitely not be provided out of the box by an off-the-shelf tool. It needs customization.

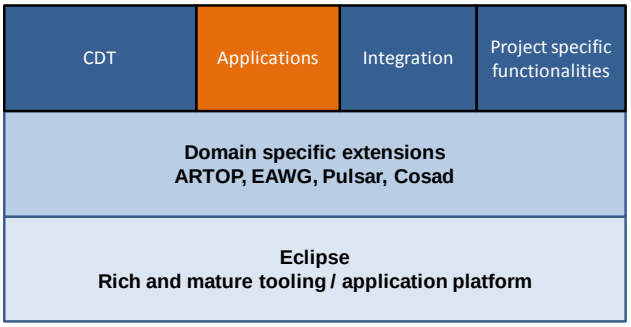
The different steps in the AUTOSAR methodology might be addressed by different AUTOSAR tools from different vendors. Those tools need to interact with each other, increasing increases the need for an open approach that allows for customization and extension.

2. Scope of the paper

In this paper, we discuss a number of examples of the challenges above. The paper shows how these challenges can be alleviated by Eclipse based tools.

Eclipse has established itself as an industrial strength development tool in industries that develop complex systems, such as aerospace and automotive. OEMs, first-tiers and suppliers already successfully use it for software development and as a rich and mature tooling platform. It can be seen as a platform for domain specific extensions that can be used out-of-the box (e.g. the C programming IDE "CDT") as well as a platform for additional tools.

The open nature of the Eclipse platform allows for the integration of applications and project specific extensions.



With Eclipse's latest Galileo release, several mature tools for model-driven software development (MDSD) have become available. These provide a flexible and low-cost set of tools for bridging the gaps in existing AUTOSAR tool chains.

The paper is structured as follows: We first introduce the AUTOSAR tool platform Artop and a number of open source Eclipse Modeling tools and explain their role in model driven development. We then show example challenges in the integration of an AUTOSAR tool chain and sketch solutions for these challenges based on the Eclipse based tools introduced.

3. Tools

In this paper, we look at Eclipse-based tools for model-driven software development. Eclipse is an open source project started by IBM a couple of years ago. Initially Eclipse was a Java development environment. However, from the start, Eclipse has been architected to be extendable in significant ways. Over time, a number of additional projects have been initiated as part of the Eclipse platform; among them the Eclipse Modeling project.

The Eclipse Modeling project is a collection of frameworks and tools for model driven development. In sum, they provide a wide range of solutions for various aspects of model driven development, from language definition to editor construction to code generation as well as model verification and validation.

This section introduces some of the tools from Eclipse Modeling. Specifically, we suggest those tools as a basis for integrating an AUTOSAR tool chain.

EMF

EMF, the Eclipse Modeling Framework, forms the basis for all Eclipse Modeling tools. At its core, it provides the Ecore meta meta model, a formalism for defining the abstract syntax of modeling languages. EMF comes with a set of related frameworks and tools for validation, query, persistence, and model transactions.

Xpand

Xpand is a framework for code generation (in fact, you can generate any kind of textual artifact such as configuration files or documentation as well). It processes models based on languages defined with EMF. The Xpand language itself is quite sophisticated in the sense that it contains a powerful expression language for querying and traversing models. It also supports polymorphism and aspects for code generation templates, making sure that nontrivial code generators remain maintainable.

Xtend

Xtend is used for transforming models into other models, for modifying existing models before they are fed into a code generator, as well as for implementing simple helper functions that are used as part of code generation templates.

Check

Check is used for validating models. Constraints are expressed in the constraints Check language (which embeds the same OCL-like expression language used in Xpand and Xtend). The constraints are then evaluated either in real time in the editors when creating models or as part of model transformation and code generation workflows.

Xtext

Xtext is a framework for defining textual domain specific languages. Based on a grammar definition, the Xtext tooling generates a parser, an EMF meta-model, as well as an editor for the language defined by the grammar. The editor provides all the features known from current IDE's: code completion, syntax highlighting, folding, customizable outline views, go-to definition and find references, etc. Using Xtext, the effort of defining languages and providing editor support on a level that makes using that language productive and comfortable becomes significantly simpler.

GMF

GMF supports the construction of graphical editors (box and line style) for existing EMF meta-models. Using this framework, you can define graphical notations for your own languages.

Integration of the tools

Using the above mentioned tools together results in a complete toolset for model driven development. You define your meta-models using EMF, then you add a graphical or textual notation using GMF or Xtext, you use check for validating models, and finally, you can use Xtend and Xpand to transform the models created with your graphical or textual editors into other models or text.

An additional project, the modeling workflow engine, is used to orchestrate the processing of models. It provides facilities for loading models from files and supplying them to validation, transformation, and code generation components.

Artop

One of the fundamental motivations of AUTOSAR is to increase the reuse of software within the car. Transferring this motivation to the AUTOSAR authoring tools is a logical step. This motivation drives the Artop initiative.

The AUTOSAR Tool Platform (Artop) is an implementation of common base functionality for AUTOSAR development tools. Artop uses many of the above mentioned frameworks to provide specific base functionalities for creating AUTOSAR tools. This includes a complete implementation of the AUTOSAR meta-model in the form of an Ecore model, an extensible validation engine for checking AUTOSAR specific and even project-specific constraints, a sophisticated workspace management

that takes into account AUTOSAR's way of storing models and lots more.

Artop is developed by the Artop User Group, a group of AUTOSAR members and partners and uses Eclipse not only as its technical platform but also uses a similar community approach to develop the software. Artop is open to any AUTOSAR member or partner.

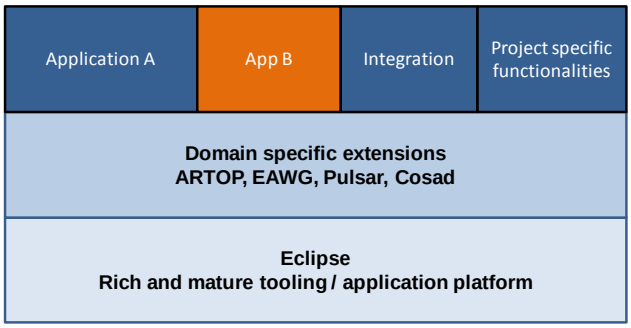
Sphinx

The Sphinx project proposal [Sph2010] is a proposal for a new Eclipse Model Development Tools (MDT) subproject.
While developing Artop a lot of functionalities were developed, that had their origins in the use of AUTOSAR, but were not specific to AUTOSAR at all. Similar functionalities were also required in other industries than the automotive domain, like avionics.
The goal of the Sphinx proposal is to remove these features from the Artop platform and rather develop them in a larger community within the Sphinx project.

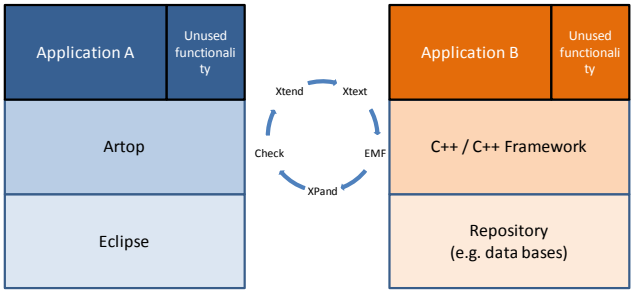
4. Bridging the gaps

Based on these tools, there are two basic types of tool integration: A tight integration on the Eclipse platform and a looser coupling through data exchange formats.

The tight integration allows tools that conform to the Eclipse architecture to be integrated within one application. The user does not have to switch between different applications, because he sees an integrated development environment.



However, existing tools used by projects might not be based on Eclipse. By making use of the Eclipse modeling tools, a loose integration is still possible through data import and export.



5. Challenges and Solutions

In this section we show how to use the aforementioned tools to address a number of challenges in the integration of an AUTOSAR tool chain.

To describe the relevant AUTOSAR models, we will introduce a textual description of AUTOSAR models called ARText. ARText is based on Xtext and an early implementation is available through the Artop distribution. This language is used to illustrate AUTOSAR models used in the examples, since it is much more concise than a graphical notation. Here is an example of a definition of a sensor component with an interface in ARText format:

```
interface senderReceiver  cardata {
      data int32 speed
}

component atomic Sensor {
      ports {
            sender X provides cardata
      }
}
```

Model-to-Model-Transformations and Model-to-Text-Transformations are shown as examples in the solutions to the challenges. The code fragments of the transformation are for illustration only. Real transformations might include additional code.

Adding AUTOSAR Import / Export to existing tools

Challenge: All tools within the AUTOSAR tool chain have to be able to read / write data in the AUTOSAR XML format. The AUTOSAR XML-format is a complex data exchange format. Adding reading / writing functionality to an existing application requires significant effort. If every tool re-implements the XML functionality, the end user basically pays several times for the development of a non-differentiating feature. The cost and time can be significantly reused by making use of Artop and the Eclipse modeling tools.

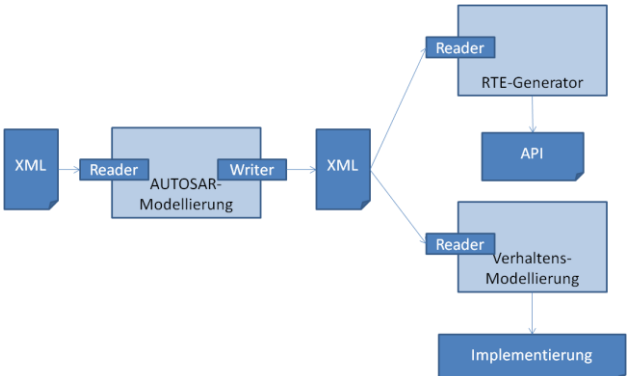


Figure 1 – XML Readers and Writers are a redundant functionality in the tool chain.

Solution: Artop provides capabilities to read and write the AUTOSAR XML formats to and from the Eclipse EMF. Using model transformations, AUTOSAR models can be converted to the native format of existing tools.

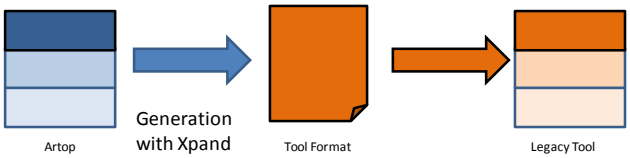


Figure 2 – Integrating non-AUTOSAR tools by model transformation

For the example, a transformation of the AUTOSAR model into a CSV (Comma-Separated-Value)-file is given. CSV is a common file format that can be read by various spread sheet and database tools. Each line of the output should contain a software component name and information about the ports of that software component. A Xpand template could be similar to the template shown in Fragment 1.

```
«DEFINE Start FOR infrastructure::autosar::AUTOSAR»
«FILE "specific"+counterInc("arpackage")+".csv"»
    «FOREACH this.eAllContents.typeSelect(
    components::ApplicationSoftwareComponentType) AS
    swc»

      «swc.shortName»;«FOREACH swc.ports AS
      p»«p.shortName»;

      «EXPAND portInfo FOR p»;«ENDFOREACH-»

    «ENDFOREACH»
«ENDFILE»
«ENDDEFINE»

«DEFINE portInfo FOR components::PPortPrototype»
        provided;«providedInterface.shortName»;
«ENDDEFINE»

«DEFINE portInfo FOR components::RPortPrototype»
        required;«requiredInterface.shortName»;
«ENDDEFINE»
```

Code Fragment 1 – M2T of AUTOSAR into .csv

```
Sensor;X;provided;cardata;
swc2;in;required;cardata;
```

Code Fragment 2 – CSV Result

The first line of the output file contains the component of the short example given in the introduction with the specification of the provided part. The second line shows an additional software component to illustrate the structure of the output file. Any output file in a textual format can be generated by Xpand.

Integration of Behavioral Models

Challenge: AUTOSAR explicitly provides no means for the specification of the behavior (program logic) of software. In model bases software development, a number of modeling approaches are well established (such as state machines or dataflow modeling). However, AUTOSAR does not specify a mapping to behavioral modeling paradigms. The linking and synchronization between with artifacts in other modeling tools is not defined.

Solution: Use model transformations to synchronize the AUTOSAR model with a behavioral model. This helps to avoid redundant modeling of elements in more than one tool and manual synchronization. The exemplary solution links AUTOSAR models with UML state machines.

The meta-model elements of AUTOSAR and the state machine have to be mapped. A state machine needs to react to data being received. For this solution, we map AUTOSAR data elements to state machine triggers. Based on this mapping, a model-to-model-transformation generates the skeleton of a state machine and infers the required events / triggers from the AUTOSAR specification.

```
create uml::StateMachine
transform(components::ApplicationSoftwareComponentT
ype m ) :
        this.ownedTrigger.addAll(m.ports.typeSelect
(components::PPortPrototype).eContents.typeSelect(p
ortinterface::DataElementPrototype).de2trigger())
;

create uml::Trigger
de2trigger(portinterface::DataElementPrototype d) :
        this.name = d.shortName
        ;
```

Code Fragment 3 – M2M of AUTOSAR into UML state machines

The code fragment generates a state machine for every software component and to keep the models synchronized, adds a trigger for each data element of required interfaces (i.e. incoming data).

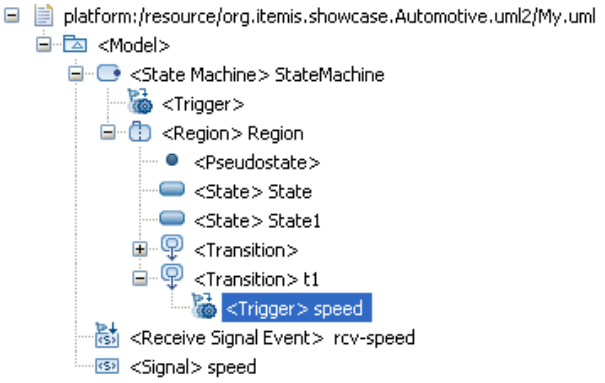


Figure 3 - AUTOSAR Data Element as triggers in state machines

Comfortable Editing of Large models

Challenge: Creating complex models with graphical modeling tools often is a time-consuming task, since the creation of new model elements involve a lot of mouse movements and switching between keyboard and mouse.

Solution: Textual domain specific languages are an established approach to create a user-friendly modeling environment. A comfortable environment with syntax highlighting, code completion and navigation functionality allows the user to quickly create models without the need for time-consuming context switches between mouse and keyboard [ARText2009]. Additionally, textual formats successfully scale to large projects and are also suited to distributed development environments. This is due to the format taking advantage of the well established textual tools and infrastructure supporting this environment. For example, configuration, source, change and version control. Support for these functions is integral for an effective distributed development environment.
With ARText the Artop community provides an environment for textual languages based on AUTOSAR. It defines and implements a textual language to describe AUTOSAR models. The framework is based upon Xtext which gives access to rich features for textual languages like syntax-highlighting and code-completion. The textual notation used in this document is the ARText notation.

Developers who are more inclined to textual formats will benefit from a less steep learning curve when developing AUTOSAR models with ARText.

Project Specific Code generation

Challenge: Existing tools might not be prepared for AUTOSAR-conforming code generation. Manual post processing of the code is necessary.

Solution: Create AUTOSAR conforming code by using customized generation templates based on Xpand.

For the exemplary integration of open-source state machine modeling / simulation, the Yakindu tool generation has been made AUTOSAR conforming.

The state machine for terminal control

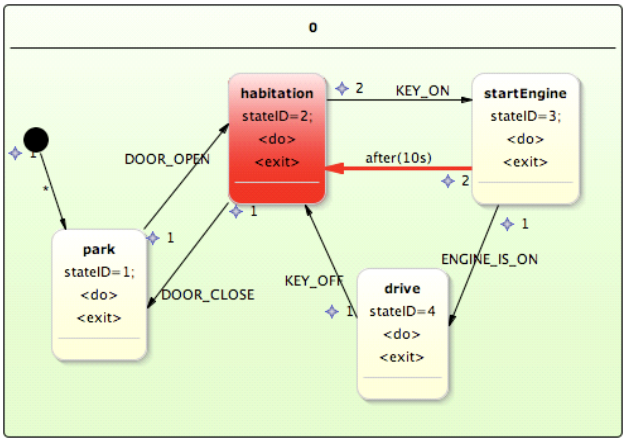


Figure 4 - State Machine as input to the generation.

can be transformed into AUTOSAR compliant code that uses the RTE as an interface to get notified of incoming transition triggers and reacts by sending data through the RTE.

Migrating Legacy Models to AUTOSAR

Challenge: Migrate existing component models to AUTOSAR. Before the introduction of AUTOSAR, projects often modeled the software architecture with other notations (e.g. UML). These models have to be migrated to AUTOSAR.

Solution: Load the models into Eclipse and transform them to AUTOSAR. Most UML2 should have a XMI-conforming export. These models are loaded into a UML2-EMF model and subsequently transformed to Artop models directly by Xtend or to textual representations like ARText.

Suppose that every class in a UML model should be transformed into an AUTOSAR software component. The following Xtend transformation will perform this transformation.

```
create infrastructure::autosar::AUTOSAR
transform(uml::Model x) :
        this.topLevelPackages.add(p(x) )
        ;

create infrastructure::autosar::ARPackage
p(uml::Model x ) :
        this.elements.addAll(x.eAllContents.typeSel
ect(uml::Class).transform())
;

create components::ApplicationSoftwareComponentType
transform(uml::Class x) :
        this.setShortName(x.name)
        ;
```

Code Fragment 4 – M2M of UML to AUTOSAR

The second create function will create an AUTOSAR package in the destination model and call the transformation for every class in the UML model. The semantics of the class to component transformation is specified in the third create statement. Here it is a simply copying of the name.

Generating RTE for non-AUTOSAR Basic SW

Challenge: Implement an AUTOSAR RTE on platforms that have no AUTOSAR Basic Software available. One of the suggested migration strategies to AUTOSAR ECUs is the use of legacy BSW with AUTOSAR conforming application layer as a first step [AR2010]. However, COTS RTE generators cannot be used, since they usually do not support project specific legacy basic software.

Solution: Use Xpand to generate a RTE that conforms the API for the application software and interfaces to the platform specific basic software. Supposing, that a communication call in the legacy software would have the signature "void Legacy_SendCall(…)", a RTE that makes use of this generation can be generated by a code fragment like:

```
«DEFINE Rte FOR infrastructure::autosar::AUTOSAR»
«FILE "rte"+counterInc("arpackage")+".c"»
«FOREACH
this.eAllContents.typeSelect(components::Applicatio
nSoftwareComponentType) AS swc»
«FOREACH
swc.ports.typeSelect(components::PPortPrototype) AS
p»
«FOREACH
p.providedInterface.eContents.typeSelect(portinterf
ace::DataElementPrototype) AS di»
void
Rte_Send_«swc.shortName»_«p.shortName»_«di.shortNam
e»(«di.type» «di.shortName» )
{
        Legacy_SendCall(«di.shortName»);
}
«ENDFOREACH»
«ENDFOREACH-»
«ENDFOREACH»
«ENDFILE»
«ENDDEFINE»
```

Code Fragment 5 – M2M of UML to AUTOSAR

6. Conclusion

In the paper we showed some of today's recurrent challenges in tools and described the need for more integrated tool chains. Challenges in the integration of an AUTOSAR tool chain can be solved by the use of inexpensive Eclipse based tooling. An open platform is the precondition for an integrated tool chain. The solutions shown in this paper have been applied in various projects.

4. References:

[Sph2010] Sphinx project proposal, http://www.eclipse.org/proposals/sphinx/

[ARText2009] S. Benz, D. Wong. "ARText - Driving Developments with Xtext", http://www.slideshare.net/sebastianbenz/artext-driving-developments-with-xtext

[AR2010] S. Fürst et al., "AUTOSAR – Migration and Advantages of an Industry-Wide Standard". Proceedings of the VDA technical Congress 2010