Built-in Interoperability and Scalability of an Eclipse-based AUTOSAR Tool Platform

S Eberle, E Fourgeau

Built-in Interoperability and Scalability of an Eclipse-based AUTOSAR Tool Platform

S. Eberle[1], E. Fourgeau[2]

1: Geensys, 242 boulevard Jean Jaurès, 92100 Boulogne-Billancourt
2: Geensys, 242 boulevard Jean Jaurès, 92100 Boulogne-Billancourt

Abstract: The automotive industry is experiencing a major paradigm shift by introducing their next generation embedded software engineering standard AUTOSAR. This does not only create a need for new tool environments supporting AUTOSAR-based electrical & electronics (EE) system design but also requires these tools to be open, customizable, and highly interoperable. Eclipse is a promising platform for realizing such engineering environments and delivers many of the necessary basic building blocks.

Keywords: AUTOSAR, Eclipse, Tool Platform

1. Introduction

The worldwide value creation in electrical & electronics (EE) systems in automotive industry reached estimated €127 billions in 2002 and should amount €316 billions in 2015 according to a Mercer study. In less than 2 years from now, software will make up an estimated 40 percent of this value creation. The considerable and increasing complexity of automotive software systems, their huge economic relevance as well as the increasingly stringent requirements on software dominant innovations in relation to comfort, safety, fuel economy, emission reduction, and onboard diagnostics have been pushing the automotive industry toward novel engineering solutions for EE architectures, such as the AUTOSAR software standardization initiative, with major consequences for EE processes and tools and new ways of sharing development and exploitation tasks.

AUTOSAR, with contributions coming from the entire automotive development chain (OEMs, suppliers, tool manufacturers, semiconductor manufacturers, software houses), has set itself the goal of developing an open, standardized EE architecture concept for the automotive industry, with some distinctive principles, such as the scalability and transferability of functions, leading to standardized software interfaces and descriptions in terms of XML file formats (see figure 1).

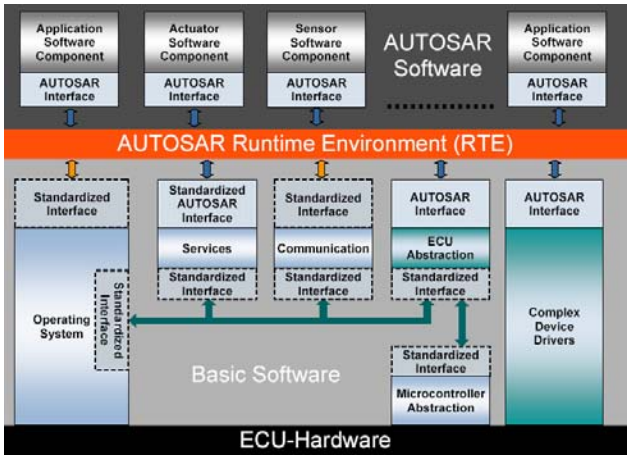

Figure 1: The AUTOSAR Architecture

However, the design of automotive systems at large, and moreover of built-in AUTOSAR systems, demands continuous exchange of data between different parties. In addition, appropriate tools are required supporting each of the interlaced development steps which are implemented by OEMs together with their suppliers and integration partners. The mere existence of a standardized XML-based data exchange format doesn't guarantee seamless interoperability which is required to fulfil the needs of architecture-oriented development processes (see figure 2).

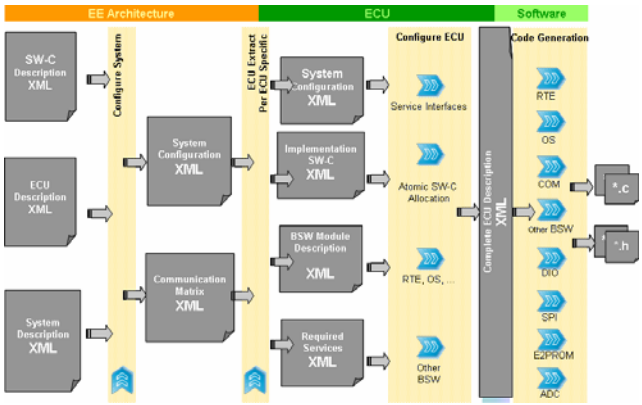

Figure 2: The AUTOSAR Process

These processes ultimately couple SYSTEM level and ECU level design tools in a single PRODUCT-centric development flow, with distributed responsibilities across the various development stakeholders. This poses the crucial demand for the

conceptual work on a tool framework which would serve as a tool hub enabling tools from different origins to smoothly cooperate with each other, and cross-linked data analysis and reports to be easily established along the various steps of the development cycle. By creating a versatile and open tool platform, a growing number of contributors including OEMs, suppliers, and tool vendors can actively increase the dynamics of system engineering tools exploitation.

In addition to the interoperability of commercial off-the-shelf tools, a prominent compelling reason for such an open platform is the need for customers to develop dedicated, i.e. use case, workflow and company-centric tools that are tightly linked to the custom processes used by each stakeholder in EE system development cycles (OEMs and suppliers alike). An open platform enabling close integration of off-the-shelf and custom tools will ease the setup of such dedicated tool environments and help to cover specific requirements along the development cycle.

2. Need for Openness and Ease of Use

Even though AUTOSAR defines common concepts for EE architectures and parts of the underlying development processes, there are many aspects of the development cycle which are not covered. On the one hand there are design activities which are not addressed by AUTOSAR standard such as the concrete function and behaviour of components, dependencies between configuration parameters of AUTOSAR basic software, etc. On the other hand AUTOSAR is more or less limited to EE system design and doesn't provide much advice on the development steps before and after design, e.g. requirements management and tracing, incorporation of timing and safety constraints, simulation, test, etc. So, it's up to each OEM and supplier to come up with appropriate concepts and processes for those aspects. And as usual when things are done off-standard, the resulting concepts and processes appear to be fairly different in each organization and therefore can be considered domain-specific.

These facts have an important impact on the question how to provide an appropriate tool support for AUTOSAR-based EE system engineering. It becomes clear that the tool environments which are limited to purely AUTOSAR-based aspects are not of much value. What OEMs and suppliers need are integrated tool solutions which provide seamless integration and interoperation of standardized AUTOSAR and domain-specific non-AUTOSAR development steps and design activities.

When thinking of how to make a good tool environment for AUTOSAR according to these considerations, looking at other embedded system engineering tools such as MATLAB and Simulink can be helpful. Providing comprehensive math and graphics functions, they are powerful computation and modeling/simulation environments. The interesting fact is however that the versatility and success of MATLAB and Simulink does not arise from their completeness. It is because they are based on an architecture which clearly separates the generic engineering environment core from problem-near and typically domain-specific behavior. The latter is realized using built-in scripting languages which are easy to learn and can be used without requiring in-depth knowledge of the engineering environment core. This enables users who are usually domain experts rather than tool experts to adapt the engineering environment to domain-specific contexts by conveniently modifying existing or adding new behaviors.

This approach is an interesting starting point for AUTOSAR-based EE system engineering environments. The different non-AUTOSAR development steps and design activities normally require very dedicated kinds of skills and expertise and may even involve specific languages, formalisms or meta models. Due to the complexity and specificity of these different development methodologies and given the fact that they are moving on and evolving quickly, it is not likely that all these needs can be satisfied by a single case tool from a single tool vendor. Therefore, a clear separation in two regards has to be made. First, to have on the one hand an open AUTOSAR tool platform and on the other hand custom extensions which are built on top of this platform. Second, there are on the one hand the tool suppliers who are true experts from a tool technology point of view and are the best to provide such an AUTOSAR tool platform. On the other hand there are the tool users who are experts of the methodologies behind the respective development steps and design activities. They are consequently the best to know how to create the extensions required for integrating AUTOSAR with these methodologies into a continuous development flow. The idea is to involve the respective experts into tool development without requiring that any of the parties has to know everything.

A promising approach for providing integrated tool support covering AUTOSAR and non-AUTOSAR aspects of EE system development is therefore to have an open AUTOSAR tool platform providing standardized AUTOSAR tooling services plus extension mechanisms which are easy enough to be used by different kinds of EE domain experts for

realizing and adding in their own tools or coupling the platform with third-party or legacy tools.

3. Eclipse-based AUTOSAR Tool Development Kit

Leveraging on AUTOSAR standard description files, structured development process, scalability and exchangeability principles, Geensys has been striving to develop such an open and easy to extend tool framework, capable of hosting various AUTOSAR-compliant design and verification tools, and providing mechanisms for being extended by custom tools or being integrated with third-party or legacy tools. It is called AUTOSAR Tool Development Kit (TDK), and represents an intrinsic part of Geensys' AUTOSAR Builder tool suite. Alike the AUTOSAR Builder tool suite, the TDK has been built upon open source Eclipse technology. Eclipse is a versatile and technology-rich platform with an open and innovative ecosystem for Eclipse-based add-ons and solutions. Given that Eclipse is based on the well-known Java programming language and its openness and extensibility it is an ideal basis for AUTOSAR TDK.

At the bottom line, AUTOSAR TDK provides the most important basic building blocks which are typically required for realizing any kind of AUTOSAR-related tool. Instead of reinventing the wheel with each AUTOSAR tool to be realized, tool developers and users can reuse this infrastructure and base their own solutions upon it. AUTOSAR TDK therefore encompasses EMF-based implementations of AUTOSAR meta model releases 2.0, 2.1, and 3.0, and a number of related services including AUTOSAR XSD conform serialization, rule-based validation, tree-based viewers, form-based and graphical editing, and template-based target code and documentation generation (see figure 3). All these models and services are accessible via on open APIs. They are at the same time the common basis for the off-the-shelf AUTOSAR design tools realized by Geensys (see figure 4) and simplify the task of developing custom tools or connectors to third-party or legacy tools for AUTOSAR TDK adopters outside Geensys.

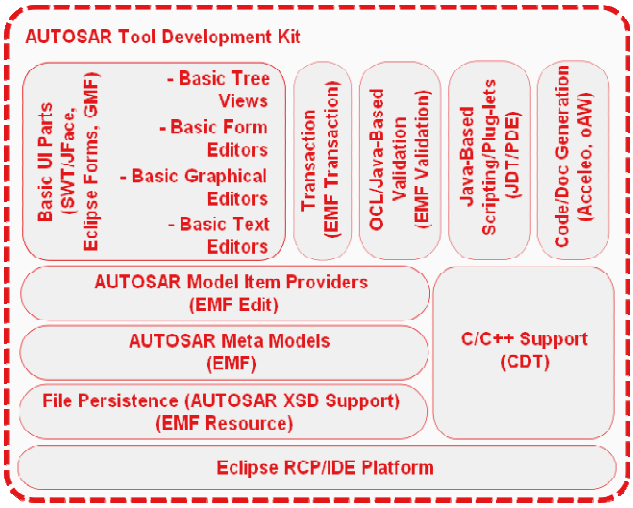

Figure 3: Meta Models, Services and APIs offered by AUTOSAR TDK

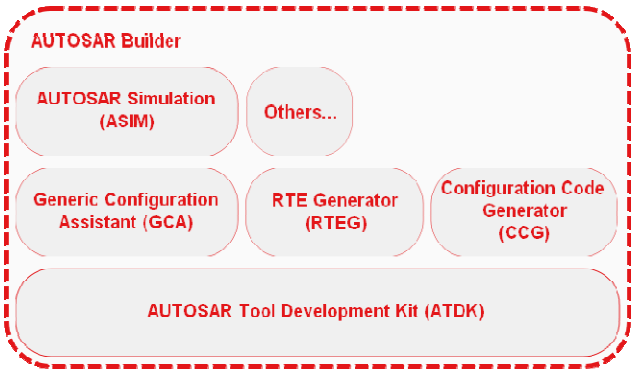

Figure 4: Off-the-shelf AUTOSAR design tools on top of AUTOSAR TDK

In order to ensure that the features of AUTOSAR TDK are not only exploitable by tool development experts with in-depth Eclipse knowledge but also by EE domain experts having only general programming skills, an easy to use scripting and plug-let environment has been added to AUTOSAR TDK. However, rather than relying on some existing or, even worse, inventing a new dedicated scripting language and having all AUTOSAR TDK adopters to learn it, the broadly known and well-understood Java programming language is reused for that purpose. Eclipse's built-in Java-development environment JDT which very is intuitive and comfortable to use has been extended and made available as script development and debugging environment. Equinox and PDE have been leveraged for contributing Java-based scripts in the form of dynamic plug-lets. As a result, scripts become compiled and deployed on the fly after each modification and are immediately available for execution. Manual export and contribution to the underlying Eclipse installation followed by a time-consuming restart of the Eclipse environment are no longer necessary. Users can develop, debug, and run scripts in short cycles and

directly within the environment where they intend to use them (see figure 5).

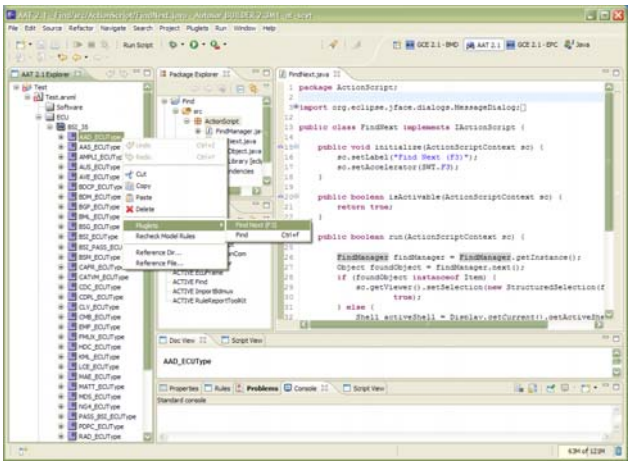

Figure 5: Development and running scripts inside AUTOSAR Builder

Given these features AUTOSAR TDK enables a wide range of application scenarios. Existing AUTOSAR tool components can be enhanced or adapted to fit domain or even project-specific needs. This is typically necessary when it comes to guided or automatic configuration of AUTOSAR basic software modules. Dedicated tool environments can be setup to support different process roles and constraints. Connectors can be created to integrate AUTOSAR with non-AUTOSAR tools, like e.g. design tools for vehicle functions and behaviors. Legacy artifacts such as company-specific vehicle message matrixes can be imported and seamlessly integrated into the AUTOSAR design flow.

5. Conclusion

AUTOSAR is the new open standard for automotive EE architectures and represents one of the biggest standardization activities in this sector. AUTOSAR is accompanied by a highly innovative system development methodology where the successive stages have to be supported by specific tools which are in turn open, inter-communicating, and collaborative. Eclipse is an ideal platform for realizing an AUTOSAR development tool kit which complies with these three fundamental characteristics.

AUTOSAR is nevertheless merely a design standard and does not cover the complete EE system development cycle. Even within the scope of AUTOSAR there are many activities like e.g. automation of AUTOSAR basic software configuration which remain domain-specific. Tool environments for AUTOSAR-based development of EE systems can therefore hardly be complete out of the box. Consequently, AUTOSAR tools must not only be ready to use in terms of AUTOSAR-based design functions. They also must be extensible by custom tools which cover domain-specific design activities or connectors to third-party or legacy tools for integration with non-AUTOSAR development steps.

Geensys responds to this demand by building AUTOSAR tools upon an open and easy to extend tool platform: AUTOSAR TDK. It is based on the open source Eclipse framework and provides common meta models, APIs and services which are typically required by all kinds of AUTOSAR tools. It also provides a built-in scripting support enabling automation-like extension and customization of existing AUTOSAR tools as well as creation of new AUTOSAR tools. Scripts are written in fairly well-known Java programming language, dynamically compiled, and packaged into plug-lets and deployed at runtime. This enables EE domain experts who typically have general programming skills but no dedicated Eclipse knowledge to instantly start developing the very specific kind of tools and connectors they need and to run and use them within the same environment.

8. Glossary

AUTOSAR:	AUTomotive Open System Architecture
EE:	*Electrical & Electronics*
TDK:	*Tool Development Kit*
EMF:	*Eclipse Modeling Framework*
XSD:	*XML Schema Definition*
XML:	*eXtensible Markup Language*
JDT:	*Java Development Tooling*
PDE:	*Plug-in Development Environment*