Diagnostic Development with AUTOSAR – the Benefits for Vehicle Manufacturers and Suppliers

OEMs and Tier 1 suppliers utilize many different processes for the development of vehicle diagnostics. Various exchange formats are used and the implemented tools are usually adapted to each company's specific process. At the very latest, problems start occurring when diagnostic descriptions have to be exchanged with development partners and used in their tool chains. This is always a time-consuming undertaking – if indeed it is possible at all to exchange data without any loss of information. The AUTOSAR Diagnostic Extract Template (DEXT) offers a completely new range of possibilities for diagnostic development. The basic software modules that are relevant for diagnostics can be configured uniformly across enterprise boundaries, for example OEMs and Tier 1 suppliers, or in collaborations between OEMs. Tasks that were previously performed by the integrator at the Tier 1 supplier can now be realized in a decentralized way using DEXT.

DEXT Compared to Other Diagnostic Data Formats

DEXT was initially published with AUTOSAR 4.2.1. In the early stage, the description of the data transported using UDS and the UDS fault memory were standardized. AUTOSAR 4.3.0 added corresponding extensions for OBD-II, WWH-OBD, FIM (AUTOSAR Function Inhibition Manager) and SAE J1939. Consequently, at this level of content, DEXT covers the configuration of all basic software modules for diagnostics that are supported in AUTOSAR. With DEXT, it is possible to describe not only the data transported using the respective protocol, but also the origin of the data in the ECU's application software. Only if both types of information are available it is possible to fully configure the diagnostic basic software. The diagnostic protocol and, in particular, the description of the diagnostic services and data transfer on the network, are not described. Here, AUTOSAR refers to the requirements from the UDS and OBD-II standards.

On the other hand, ODX has established itself as the data format for generic diagnostic testers. ODX describes the diagnostic protocol as well as the data transported between the ECU and the tester, together with the way this data is interpreted in a diagnostic tester. The source of the data in the ECU is of no importance for its processing in the tester and is therefore not specified in ODX. Despite this, ODX might be used as initial configuration input, primarily describing the existence and structure of the diagnostic data on the network. However, the diagnostic data must be linked to the ECU application either manually, or by means of special process steps. The information gap between ODX and ECU software is at its greatest when it comes to the description of the fault memory. For example, the debouncing or displacement algorithms used for error detection are of fundamental importance for the basic software, whereas this information is completely lacking in ODX. To some extent, the huge differences between the vehicle manufacturers' ODX authoring guidelines further complicate the possibility of exchanging ECU configurations.

In practice, processes that use the AUTOSAR-ECUC format for data exchange and initial data in the basic software rarely reach their goal. The ECUC format changes frequently and is adapted with every new version of the standard. Primarily it is designed as an input format for the embedded software code generators. ECUC also permits manufacturer-specific extensions, thus making it unsuitable as a neutral exchange format.

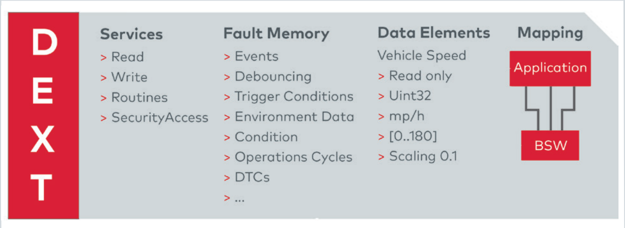

Figure 1: Elements of the AUTOSAR Diagnostic Extract

AUTOSAR DEXT has been specially designed to meet the requirements regarding input data of the basic software. The main elements are (Figure 1):

> Selection of the diagnostic services and associated subservices for UDS, OBD, WWH-OBD and J1939
> Fault path and fault memory
> Diagnostic data elements and the associated packaging
> Mapping of the diagnostic data elements to the application software
> Function-inhibition (FiM)

The example below illustrates the advantages of the AUTOSAR Diagnostic Extract based on a Data Identifier (DID). The AUTOSAR metamodel formally defines how a DID should be modeled. Unlike in ODX, there is no freedom of interpretation here and therefore also no risk of misunderstandings during data exchange between tools. The existence of a DID is specified by the instance of a DiagnosticDataIdentifier. This instance contains the 16-bit number of the DID which is required for UDS. In addition, this instance aggregates one or more data elements, with the name and type of the DID data being defined. For the purposes of data type modeling, AUTOSAR reuses the pre-existing system template metamodel. The DID itself can be used in a diagnostic service by the service instances DiagnosticReadDataByIdentifier, DiagnosticWriteDataByIdentifier or DiagnosticIOControl by means of a reference to the DiagnosticDataIdentifier. This is all what is needed to define the configuration of a DID in the diagnostic basic software (BSW).

However, the basic software must still interact with the application software in order to read, write, or overwrite the DID. That is why DEXT contains an additional element: diagnostic mapping. It describes the connections between the diagnostic elements in the basic software, such as routines, DID data, events, and the software components (SWCs) of the application layer. To this end, the interfaces of the software components must be suitably modeled in a way that adheres to a modeling pattern defined in AUTOSAR. Various communication patterns exist to access a function call on a client/server interface or read/write data via a sender/receiver interface. In the past, integrators had to manually configure thousands of connections of this type between the ports in the basic software and the application software. When DEXT is used, this operation can be automated and performed at the OEM/Tier 1 supplier. Integrators take over the mapping rather than having to generate it manually themselves for a large number of different connections. This reduces the probability of error, while simultaneously saving time and increasing quality.

Scenarios and Roles in Distributed Diagnostic Development

Currently, there are considerable differences between the diagnostic development processes that are used today. In addition to the tools and their exchange formats, there are major differences in the contribution of OEM and TIER1. In practice, all the possible variants can be found: from end-to-end design by the OEM through combined diagnostic development by the OEM and Tier 1 supplier and on to complete development performed exclusively by the Tier 1 supplier. Table 1 provides an overview of typical diagnostic processes.

Scenario 1 Complete design of diagnostics by the OEM	**Scenario 2** TIER1 extends the OEM diagnostic specification	**Scenario 2** TIER1 creates the diagnostic specification
> OEM creates system design (incl. SWCs with diagnostic interfaces) > OEM provides the ECU diagnostic specification > TIER1 takes over the ECU diagnostic specification from the OEM	> OEM creates system design (incl. SWCs with diagnostic interfaces) > OEM provides the ECU diagnostic specification > TIER1 extends the ECU diagnostic design > OEM updates his diagnostic specification with the modifications of the TIER1	> TIER1 creates SWCs with diagnostic interfaces > TIER1 creates the diagnostic specification suitable to SWCs > OEM takes over the diagnostic specification of the TIER1 in his specification

Table 1: Diagnostic processes between the OEM and Tier 1 supplier

Thanks to the use of DEXT, it is possible to support each of the three process variants. As in all AUTOSAR arxml formats, almost all the elements are also optional in DEXT. In the individual process steps, DEXT can either be initially created and enriched or can be extended at the end of the process. The created data is always inherently valid and can be exchanged. It is of no importance which data is added in which process step and this is defined simply by the applied process.

Requirements Relating to a Tool Chain that Supports these Processes

It must be possible to adapt tool chains used in the diagnostic development process to the above-mentioned scenarios. At the start of the process, it is necessary to define the diagnostic requirements in the Requirements Management System (RMS). From the diagnostic requirements, it is then possible to derive requirements for the application software and requirements for the associated diagnostic services. In general, the two types of requirements are implemented by different roles – in the form of software components as well as in the form of a suitable basic software configuration. They are then combined by the integrator during ECU integration. This is exactly where the aforementioned diagnostic mapping in DEXT comes into play. The overall process is illustrated in Figure 2.

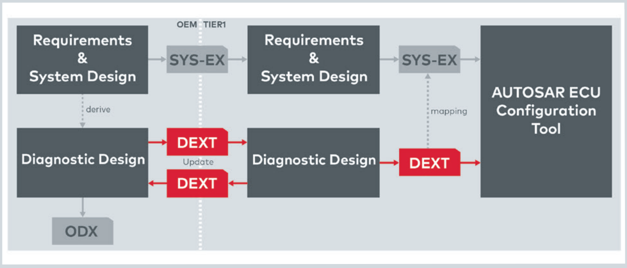

Figure 2: Diagnostic process between the OEM and Tier 1 supplier

In a top-down process, the diagnostic application software is first developed, or existing software is re-used. The diagnostic input data is derived from the requirements and interface descriptions for the application software. This represents a major advantage compared to many existing processes in which the diagnostic data is adapted to the requirements and application software. This is often a very time-consuming manual harmonization task.
At the same time the DEXT format is created, it is also necessary to create the ODX data. Generating ODX and DEXT data from a common source ensures that the diagnostic tester behavior matches the diagnostic software in the ECU.

Example Based on a Tool Chain

Figure 3 illustrates the example of a tool chain for diagnostic development that meets these requirements using tools from Vector. It consists of the tools PREEvision for requirements and system design, CANdelaStudio as the diagnostic authoring tool and DaVinci Configurator Pro for configuring the basic software. A diagnostic requirement is defined in PREEvision at an early development stage. For example, this could be a requirement to provide an oil temperature sensor. For this sensor, the diagnostics should contain a service for reading the oil temperature, a service for overwriting the sensor value by means of I/O control as well as support for one or more possible DTCs that indicate a temperature sensor malfunction.
The software component interfaces of the System Extract are derived from these requirements in arxml format. The software component interfaces also define the parameters for the diagnostic objects. In the example of the oil temperature sensor, a software component port provides the current temperature value and the port's interface defines whether the measured value is a 16 or 32-bit value, together with the conversion formula and unit used. A software component synchronization functionality specially developed for this workflow in CANdelaStudio then automatically uses this information to generate the appropriate diagnostic data for the following diagnostic elements:

> Diagnostic Data Identifiers (DIDs) for Read, Write and I/O-Control
> Routine Control Identifiers (RIDs)
> Events and DTCs

In our example, the diagnostic expert creates a "ReadDataByIdentifier" diagnostic service in CANdelaStudio. This contains a "Temperature" data element with an unsigned 16-bit value and a resolution of 0.1 Kelvin, a I/O-Control service with the same data element and a DTC indicating a sensor defect. The diagnostic expert also defines the identifier that is used to access the oil temperature in the diagnostic communication.
Additionally, CANdelaStudio stores the port at which the software component supplies the temperature and the diagnostic service that reads the data from this port. Using this information, CANdelaStudio is able to export the DEXT format together with the diagnostic mapping. DaVinci Configurator Pro reads in the DEXT format and derives the configuration of the diagnostic basic software modules from it. DaVinci Configurator Pro then generates the software component interface for the diagnostic basic software modules and connects it to the ports of the application software components – in a way which is compatible with the diagnostic mappings in the AUTOSAR DEXT.

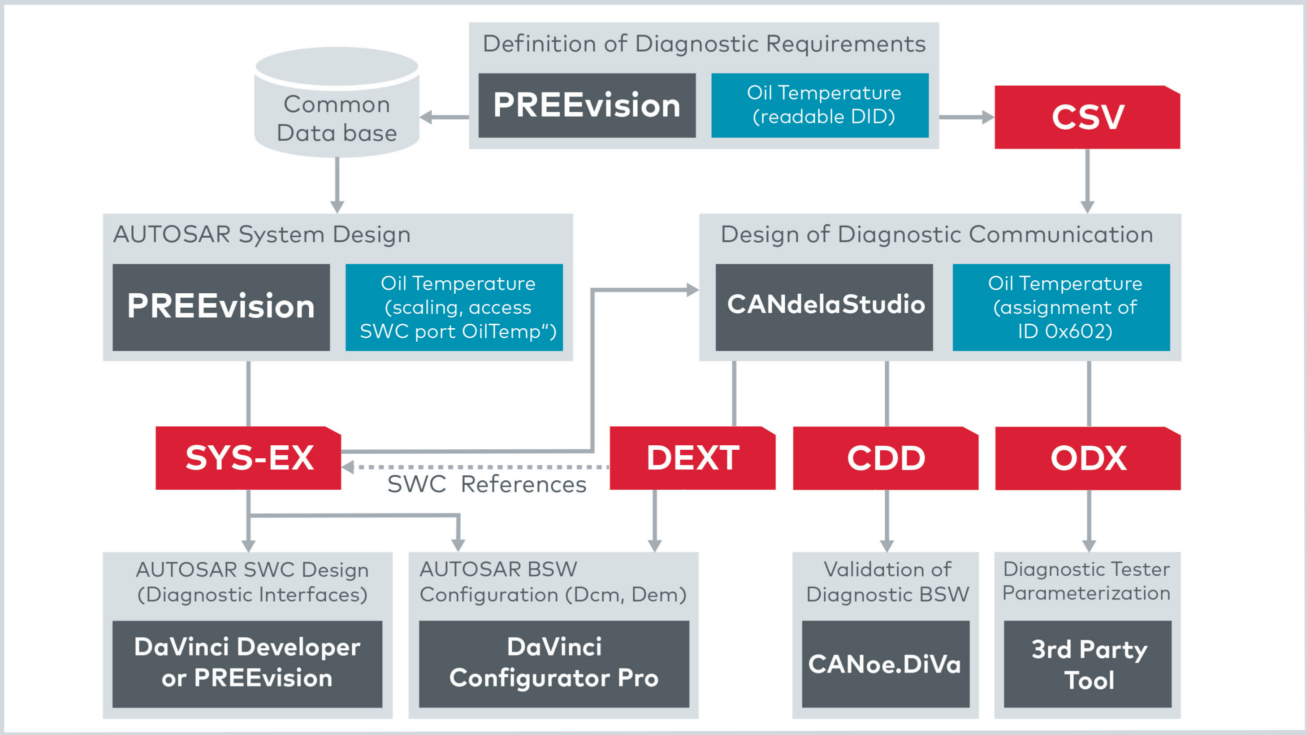
Definition of Diagnostic Requirements
Common Data base
PREEvision
Oil Temperature (readable DID)
CSV
AUTOSAR System Design
PREEvision
Oil Temperature (scaling, access SWC port OilTemp")
Design of Diagnostic Communication
CANdelaStudio
Oil Temperature (assignment of ID 0x602)
SYS-EX
SWC References
DEXT
CDD
ODX
AUTOSAR SWC Design (Diagnostic Interfaces)
DaVinci Developer or PREEvision
AUTOSAR BSW Configuration (Dcm, Dem)
DaVinci Configurator Pro
Validation of Diagnostic BSW
CANoe.DiVa
Diagnostic Tester Parameterization
3rd Party Tool

Figure 2: Diagnostic workflow based on the example of a Vector tool chain

Conclusions and Outlook

The AUTOSAR Diagnostic Extract opens new possibilities in the field of diagnostic development. From the precise description of the data, including the derivation of the basic software configuration, through the distributed development of diagnostics at the OEM and Tier 1 supplier through to a top-down approach to the automatic integration of the diagnostic functions in the ECU. The Vector tool chain uses these capabilities and simultaneously ensures the synchronization of the ODX and DEXT data.
AUTOSAR DEXT is used in both AUTOSAR platforms: initially developed for AUTOSAR Classic, it is also the only diagnostic description format in AUTOSAR Adaptive. The presented method can therefore also be used for the AUTOSAR Adaptive platform based development.

Author
Dipl.-Inf. Wigbert Knape is Product Manager for the Embedded Software and Systems product line at Vector in Stuttgart.

Author
Dipl.-Ing. (FH) Matthias Wernicke is Product Manager for AUTOSAR Tools in the Embedded Software and Systems product line at Vector in Stuttgart.

Author
Dr. Klaus Beiter heads a development team in the Diagnostic product line at Vector in Stuttgart.

Translation of a German publication in Hanser Automotive magazine, issue 9/2018