The development of modern electric/electronic (E/E) architectures is a challenge today more than ever. Numerous development criteria must be taken into account, and classic vehicle domains need to be linked to new functions in the areas of driver assistance and autonomous driving. Completely new functions that are no longer limited to just the vehicle, but are also provided as services in the "IT backend" outside the vehicle, are emerging here. The introduction of service-oriented architectures and powerful domain computers, Ethernet to on-board communication, connectivity gateways and, last but not least, increasing safety and security requirements represent profound and radical changes for every development organization. To master these complex development tasks successfully as a team, a development platform and E/E database, which can be implemented in different ways, are required.

Many functions provided by the E/E system of modern vehicles can be associated with the classic powertrain, chassis, body and multimedia domains (Figure 1), as before. As a result of the use of new technologies both inside and outside the vehicle, however, issues such as driver assistance, automated driving and connectivity (Car2x/V2x) are gaining tremendously in importance.

A comprehensive function and ECU network such as this can only be controlled efficiently if the development organization can depend on a computer-aided development environment with a centralized E/E database. Without this development environment, the expenditure required for internal coordination and the management of redundant and potentially inconsistent data increases.

This results in higher development costs, more time being required and, in the worst case scenario, errors that are first noticed out in the field.

**Controlling the Function and ECU Network**
Most vehicle functions provided by the E/E system concern open-loop/closed-loop control, monitoring and diagnostic functions; they interact with the mechanical vehicle components via sensors and actuators. As the sensors and actuators are installed at different geometric locations in the vehicle, E/E systems are naturally geometrically distributed systems. Many functions are active within one domain, but can also operate with one another across domain boundaries - on a function network. Driver assistance functions, in particular, work closely together with the functions of the powertrain, steering and brake systems. This can be illustrated using control chains of the required sensors, ECU functions and actuators required for implementing a vehicle function (Figure 2).
The definition of the function and ECU network and the distribution of the functions on the ECU network represents a high degree of freedom in the design and is thus an important architecture decision. Nearly all evaluation criteria of an architecture, such as total cost, weight, installation space, bus-load, safety and security are influenced by this. For the serial development process, this means that a differentiation must be consistently made between the two layers, i.e. function network and ECU network. In addition, these layers are not just developed for one special design variant.

Development always covers a greater number of vehicles designed based on a common platform. Typically, many people are involved and several variants (generally more than 20) are developed together. This means that the complexity of the task is enormous. For an organization the only way this can be overcome, is to keep an overview of all development artifacts and their interaction through common storage of the artifacts, visualization, computer-aided evaluation and design automation.

**Process Control in the Partitioning and Integration of the E/E System**

This differentiation between the function network and ECU network represents a decisive prerequisite for the successful partitioning and integration of the E/E system and the cooperation between vehicle manufacturers and suppliers. Building upon this, the tasks, roles and responsibilities for partitioning, integration of the function and ECU network can be precisely defined, planned and tracked. An overview of the entire process in E/E development is presented in Figure 3 [1].
The development process must be supported by suitable IT systems. For this purpose, specific development tools are used for development disciplines such as mechanical construction, E/E development, and software and hardware development (Figure 4). All the development results arising there are supplied to a Product Life Cycle Management System (PLM system) that supports the subsequent production and service processes.

**Exchange Standards as the Key to Success**

In E/E development, particularly at the interface between vehicle manufacturers and suppliers, the advantages of standardized exchange formats for E/E artifacts have already been recognized for a while.

By now, the required standards are available, consolidated, widely accepted and applied for years:

- ReqIF or RIF is used for the exchange of requirements, function descriptions and test cases.
- AUTOSAR provides a base software platform for ECUs that can be configured using comprehensive description formats. The most important format here is the **AUTOSAR system description**, which fully covers the areas of software architecture, the ECU network and communication for CAN, LIN, FlexRay and Ethernet with regard to the data to be exchanged between vehicle manufacturers and ECU suppliers.
> KBL is widely used for the exchange of wiring harness and geometry information, and it is expected that this standard will be supplemented with VEC over the next few years and replaced with it in the medium term.

This means that substantial volumes are covered for the E/E data exchange between the supplier and manufacturer. Text-based specification documents are increasingly being supplemented or even replaced by the exchange of “digital” specifications in these formats.

**Team Platform for E/E Development**

The creation and maintenance of these E/E specification elements is always distributed across teams and departments, often even beyond company boundaries. This team-based work must be enabled by the development tools used. This immediately raises the question of a suitable team and database platform on which the following basic requirements are placed:

> Multi-user operation: Simultaneous, parallel processing of artifacts by multiple users with as little potential for conflict as possible
> Traceability of the requirements over the course of implementation up to the test cases (especially with safety-related functions required by ISO 26262)
> Archiving, versioning and reuse concept for the arising E/E artifacts on a granular level
> Well-organized change and release management with integrated release and project planning, as well as transparent project tracking
> Controllable maintenance expenditure for the E/E artifacts based on a single-source principle for data storage

> Granular concept for authorization and maturity states for the support of different roles
> Support for large organizations with many users, large data quantities and various different development locations
> Integration of customers and suppliers
> Justifiable administration and operating expenditure, including backup support
> Future-proof thanks to consistent further development of the E/E development environment and migration of the available data
> Openness thanks to flexible integration into the existing IT, process and tool landscape

There are a variety of alternatives for the implementation of an E/E development environment, which are different in principles:

1. **Document-based method of working:**
   Use of different authoring tools and storage of the created file artifacts in a centralized database (Figure 5a)

The advantages of using authoring tools tried and tested in use with the corresponding dataset are balanced out by the disadvantage of teamwork only supported on coarse granularity and traceability only available on the files created here. Close cross-team cooperation on the level of detailed E/E artifacts using the single-source principle is thus not feasible. Consistent development beyond the boundaries of the individual authoring tools is difficult and requires additional manual effort.
The use of tried-and-tested authoring tools by the team is advantageous. Limited integration continues to have a negative effect.

From the operational standpoint, multiple databases and authoring tools must be maintained and possibly even integrated. Point-to-point connections are the result. Integration is possible using standards such as OSLC, and the integrated artifacts are saved in one location based on the single-source principle, but are visible and even editable using different authoring tools.

Relationships beyond database boundaries that arise in this way are weak and cannot be reliably versioned or archived, however. Additionally, such relationships may be destroyed during data migration on a tool version update (assuming that data migration is even possible), or be impossible to create retrospectively if a new tool version introduces a new relationship possibility.

As a result, overlapping, redundant data management with proxy objects is often accepted as well; relationships must be ensured using cyclical export/import and update processes. Other disadvantages include (as before) the lack of an integrated data model, the always noticeable tool boundaries, various different architectures (e.g. rich client or web client) and different release, versioning, backup and migration concepts. This results in high integration and operational expenditures.
3. Model-based and integrated method of working
Use of a consistent authoring tool and storage of the created model artifacts in a centralized database (Figure 5c)

The weak points of a model-based, but distributed, method of working can be resolved using a consistent authoring tool and centralized storage in a database. Wherever the advantages of a consistent E/E engineering data model with centralized data storage have been identified, proprietary developments (frequently databases) often arose as the solution in the past, as a commercial product solution was simply not available.

Proprietary Solutions: High Effort and Expensive
The advantages and disadvantages are obvious. With a proprietary solution, the data model and process can be supported at the organization in a highly targeted way. Proprietary solutions lead to high development, maintenance and testing expenditures, however. The user interface is often not state of the art as well.

Product Solution with Broad Configuration Options: Feasible and Economical
A product solution with broad configuration options is easily justified through efficiency savings and greatly reduced integration efforts. PREEvision follows precisely this approach. A consistent and complete automotive E/E data model could not be designed until acceptance of the standards described previously was achieved in the automotive industry.

The PREEvision E/E data model is the foundation for a consistent model-based method of working from the requirements to all the implementation steps through to testing.
All the required aspects of an E/E architecture are modeled in an integrated approach:

- Functions, features and requirements
- Network aspects and function distribution
- Software and communication aspects
- Hardware, wiring harness and geometry aspects

In adopting this approach, PREEvision follows the three proven systems engineering principles of abstraction, decomposition and reuse and supports the modeling of product lines and product variants with various model layers (Figure 6).

Considering the data model layers (Figure 6), the abstraction from the geometry layer with installation locations and routing paths, the wiring harness layer and electric circuit layer to the ECU network layer is represented in the vertical direction. The software and communication details are modeled in parallel with that. Hardware and software aspects are described abstractly in the logical function architecture layer. An even greater degree of abstraction occurs in the layers for the requirements and customer features. The horizontal direction of the PREEvision data model representation supports decomposition. Hierarchy concepts that can be used bottom up or top down are available in each layer. The third direction is orthogonal to the other two directions and enables reuse and variant concepts in each model layer and hierarchy level.

An additional benefit arises from the logical function architecture abstracted from the implementation in hardware and software. The control chains for vehicle functions, which are often stable for many years, can be described here, whereas the hardware, software and communication technologies for their implementation change from one vehicle generation to the next. This applies for functions of both control and monitoring natures in the domains of powertrain, chassis, driver assistance and body, as well as for the functions of the multimedia systems. All these layers are interconnected via so-called mappings.

Figure 7: Architecture of the PREEvision collaboration platform
The storage of files created using tools for the behavior modeling of functions for simulation, rapid prototyping and code generation is possible in many locations in this model context.

Import and export interfaces for the creation and processing of the specification formats RIF, ReqIF, AUTOSAR and KBL are available. The PREEvision data model aggregates the data to be exchanged here into easy-to-understand, easy-to-edit and reusable engineering artifacts, such as ECUs, plugs, pins, software components, signals and messages and provides the necessary relationships between them. Powerful engineering tools, graphical diagrams and table-based editors are available for editing.

The integrated model-based approach supports the single-source principle, through providing cross-domain visibility of items to be analyzed. Any sub-section or the whole of the model can be checked for consistency and completeness with configurable validations. Particular benefits may be realized when considering safety aspects; the process for functional safety is supported by tools for well-known methods, such as hazard analysis and risk assessment, FMEA and fault-tree analysis, which are carried out using the existing product line data.

By using a feature model according to Feature-Oriented Domain Analysis (or “FODA”) principles in the customer feature layer, a valid selection of features can be defined for a vehicle variant [2]. The features can be linked with the corresponding artifacts of all other layers. As a result, an E/E architecture can be derived, edited, checked and exported for a selected variant.

PREEvision supports teamwork with a collaboration platform in a three-tier architecture (Figure 7). A team of users can work in parallel on one or multiple product lines. All E/E artifacts are subject to fine-granular version management and are connected with tickets for coordinated change and release processes.

Comprehensive configuration options come along with the mature, tried-and-tested and disclosed data model and many development tools in the product scope: roles and rights, additional application-specific attributes, maturity states, reports, table editors and views, scripts, validations and specific, even automatable, import and export interfaces.

This enables the specific needs of the respective development organization to be flexibly mapped and PREEvision to be integrated into the existing process and tool landscape. In addition, so-called perspectives with filter and editor pre-settings are able to be used to set up a user-friendly operating interface for dedicated use cases (Figure 8).
Summary and Outlook

Last but not least, future process support is ensured thanks to intensive maintenance and development of PREEvision by Vector to keep pace with the rapid innovation seen in the automotive industry. A migration mechanism for the inventory data in the database is provided as part of the delivery of a new version. This enables further development of the data model and the switch to new PREEvision versions with justifiable expenditure, a demanding task whose meaning is often not realized during proprietary development until it has become acute.

The PREEvision approach has been applied in productive use for years and supports the entire E/E development process in a consistent, model-based and future-proof way, from requirement management to architectural design, function-oriented system and component design, software and communication design and wiring harness development through to testing and release for serial production.

Literature References


Authors

Dr. Thomas Beck
Vector Informatik GmbH
Managing Director

Dr.-Ing. Clemens Reichmann
Vector Informatik GmbH
Technical Director PREEvision

Dipl.-Ing. Jörg Schäuffele
Vector Informatik GmbH
Product Manager PREEvision

Originally published in "ATZ elektronik" magazine
Issue 06 – December 2016