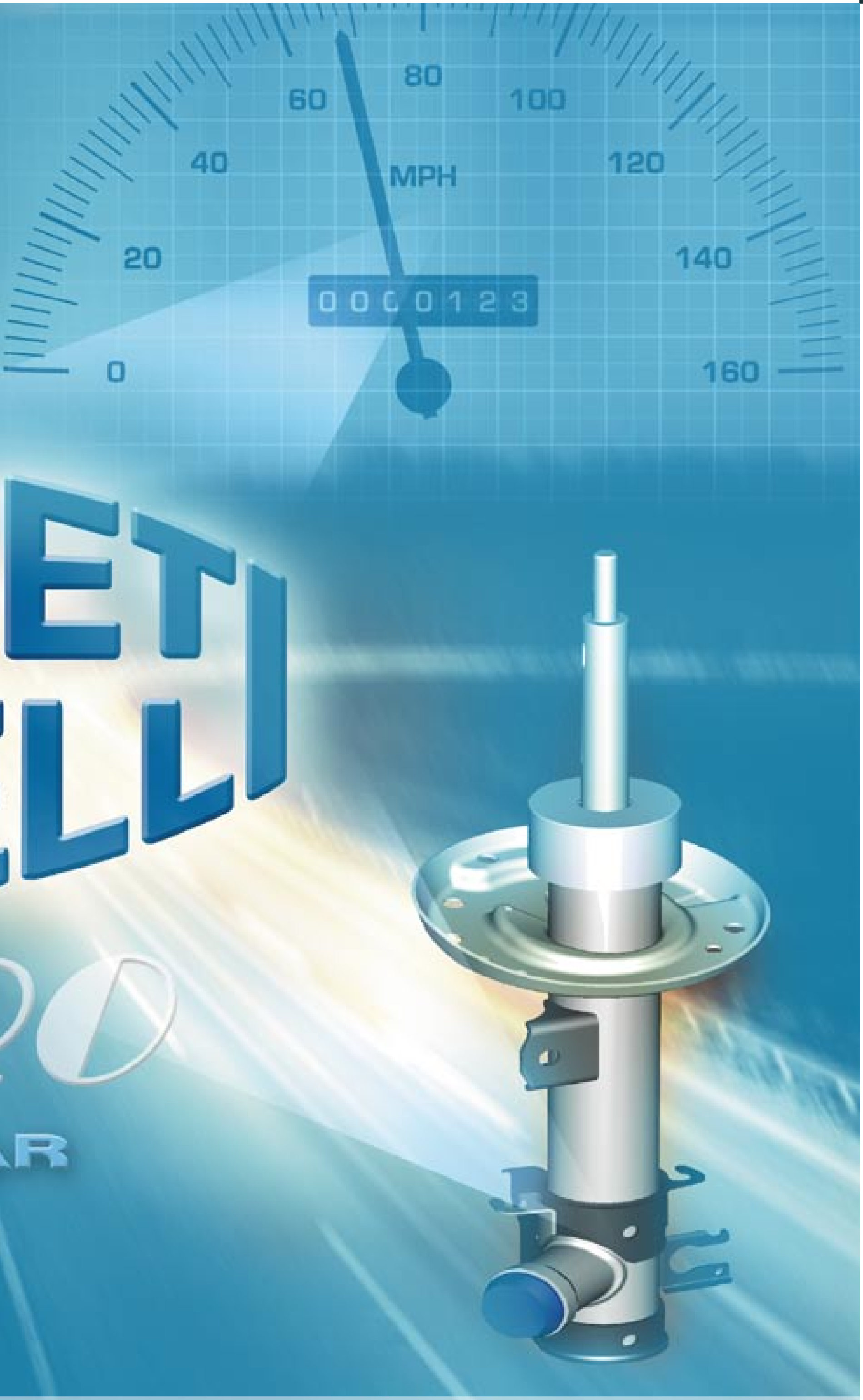
Engine Management AUTOSAR Way

the

Magneti Marelli migrates ECU software to the AUTOSAR standard



AUTOSAR is a hot topic in the development departments of automotive suppliers and OEMs. The standard is frequently accused of having too much overhead in implementing complex systems. In an engine ECU project, Magneti Marelli has set out to prove otherwise.

„Magneti Marelli AUTOSAR Cross-Project X-PRO“ is a special project in which we are developing a purpose-built demonstrator to verify the feasibility of AUTOSAR-compliant software architectures – even for complex engine management systems. In concrete terms, our task is to migrate the entire software of an existing engine ECU to AUTOSAR and then reimplement it on that ECU. Engine control software is usually enormous in size, making it vital to be able to reuse modules simply. That is reason enough for our interest in software development the AUTOSAR way. Working together with dSPACE, we chose an approach in which the new automation feature of SystemDesk, the system design software, plays a major part. Because this demonstrator project is concerned with feasibility and with obtaining information on performance and resource consumption, we did not design a new architecture top-down, but largely retained the existing ECU software design. To do so, from the existing ECU data we extracted all the information needed for reconstructing the software architecture and scheduling, and transferred it to SystemDesk by means of a script (fig. 1). This procedure quasi-automatically produces an AUTOSAR-compliant architecture that requires only a few manual adjustments in SystemDesk.

Preparing Data and Information

Before AUTOSAR, there was no standard available for describing all the implementation information, so it is not possible to reuse the individual software modules systematically. Thus, our first task was to collect data on the ECU’s configuration, parameterization and implementation from a diversity of data sources and bring it together at a

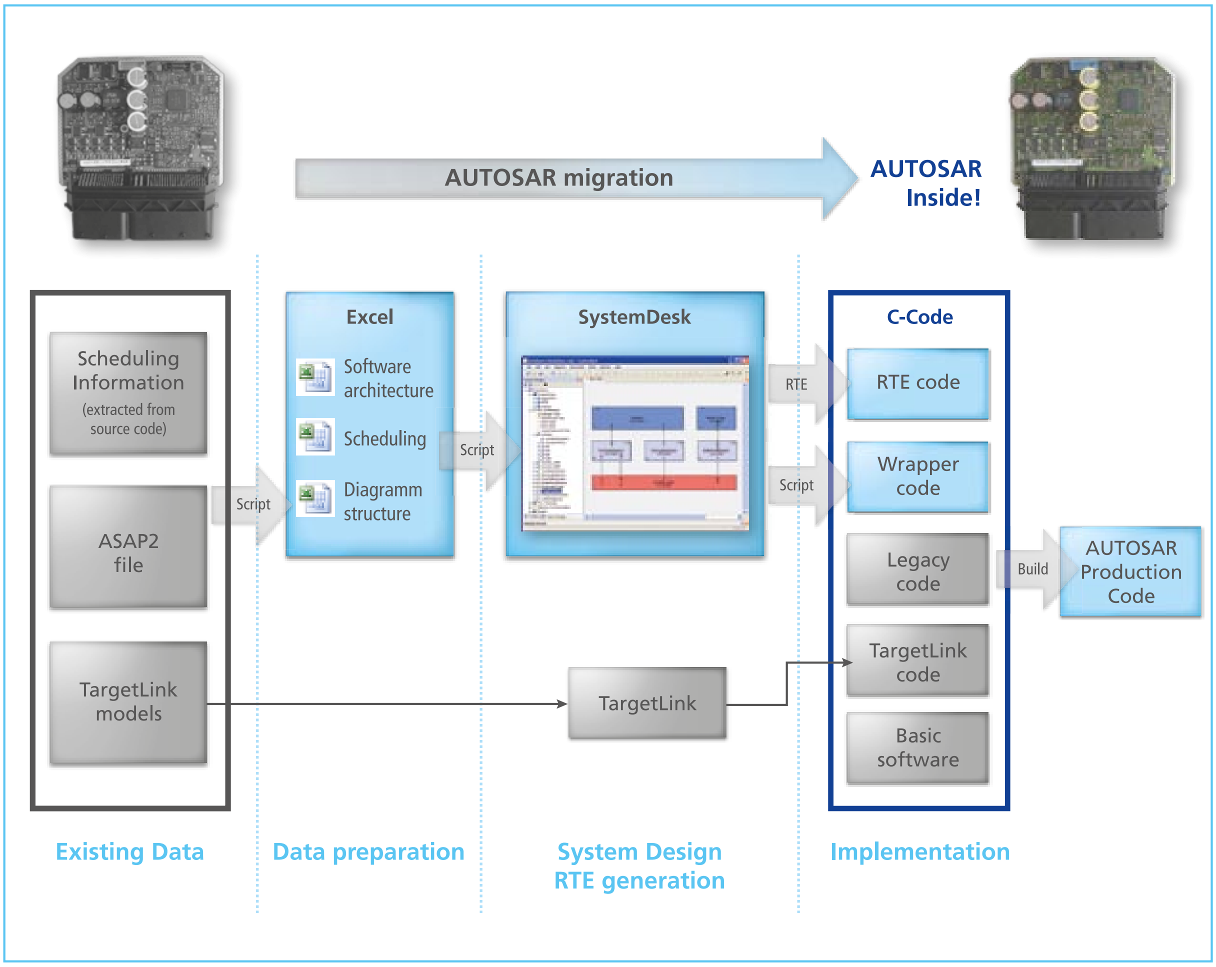


Fig. 1: Overall workflow for AUTOSAR migration.

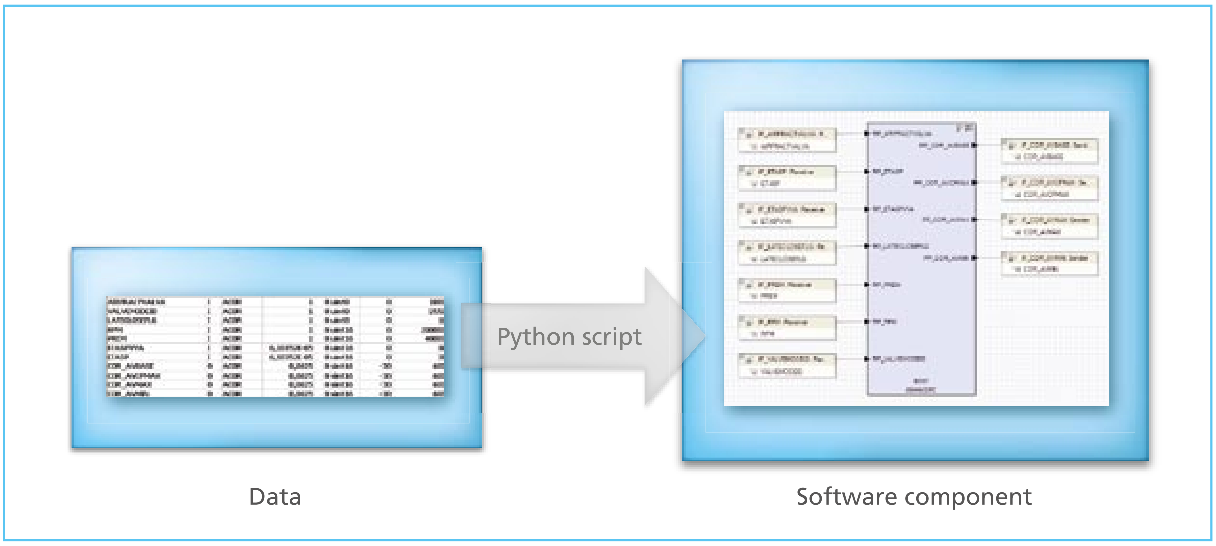


Fig. 2: All the variables of a software component (SWC) in Excel on the left, with corresponding SWCs in SystemDesk on the right.

central point for further processing. For example, we used a script to extract the architecture information from existing TargetLink function models. Description files such as the ASAP2 file were also used to obtain the data types, min/max values and scalings of the implemented variables. Since the configuration of the old ECU's operating system also had to be reused, the entire function schedule was extracted from the old source files by a script and transferred to Excel. Standards like AUTOSAR will obviously make it much easier to reutilize existing software modules and implementation data in the future.

Building the Software Architecture

Data from an Excel sheet now had to be migrated to the system world to produce an AUTOSAR-compliant system architecture. Via SystemDesk's automation interface, a Python script was used to create all the required software components fully automatically, including sender/receiver interfaces and data

"Variables are accessed via the SystemDesk-generated RTE with the same performance as in classic implementation."

Luigi Romagnoli, Magneti Marelli

elements (fig. 2). With an import volume of over 20,000 elements, the occasional error is obviously unavoidable. However, SystemDesk is able to detect and display all the inconsistencies in the input data. For example, if a data type did not match the specified scaling information and min/max values, this error was indicated while the software architecture was still being created.
To import the variables, we used a naming scheme that allowed all the connections in the software architecture to be built automatically. With more than 170 software components and several thousand input and output signals, this saves an inestimable amount of time. With such enormous quantities of data, it hardly makes sense to represent the entire architecture in a single diagram. SystemDesk therefore allows subfunctionalities to be brought together in composition diagrams, each providing a different view of the software architecture. This means that even in large-scale projects everyone has a clear view of what they are doing, and the relevant information is always available for discussing aspects that concern several teams.

Importing the OS Schedule

The old operating system schedule also had to be used for the AUTOSAR ECU. First an Excel sheet was used to assign runnables to software components, and then all that was left to do was to specify the execution sequence within the operating system tasks. After that the schedule was transferred to SystemDesk completely automatically.

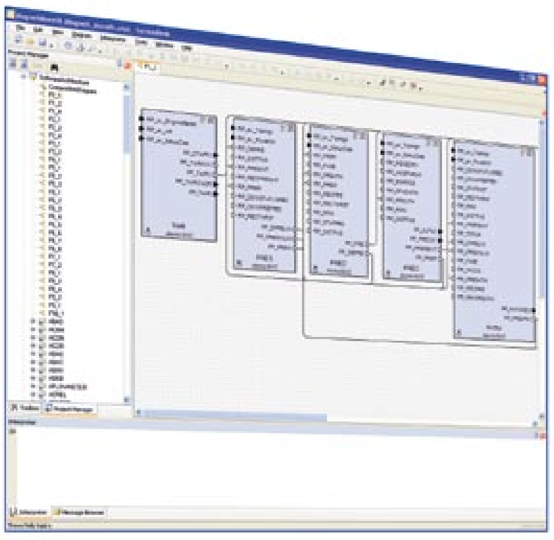


Fig. 3: Subset of the overall model shown in an individual composition diagram.

"SystemDesk's automation feature not only speeds up processes, it also ensures data integrity."

Alessandro Palma, Magneti Marelli

Reducing Graphical Complexity

Engine ECUs are among the most complex ECUs in a vehicle. Around 10,000 variables, 600 runnables, and 16 tasks have to be defined and optimally managed. Any means of reducing complexity helps avoid errors and speeds up the development process. Composition diagrams showing individual aspects of the software, as described above, are not the only way in which SystemDesk reduces complexity. There are numerous other methods (see also Fig. 3). For example, it takes only three clicks to hide all the (connected) ports of a software component or to display a dialog showing all the as yet unconnected ports of an entire software architecture. What we particularly liked about working with SystemDesk's graphical interface was the ability to represent complex systems in a simple, reduced form. This is the only way to run successful reviews and explain new ideas quickly on screen.

RTE Generation

In migrating legacy code to the AUTOSAR standard, an important intermediate goal is to perform RTE generation for the entire software architecture, including the OS schedule. The run-time environment (RTE) is slim middleware that embodies the communication between AUTOSAR software components in software form. In some circumstances, this intermediate layer might require extra resources compared with our previous software implementation. It was therefore important for us to estimate the memory and execution time requirement of the RTE code generated by SystemDesk. To obtain useful information on this, no AUTOSAR-compliant application code or basic software is needed at this point in time.
The analysis of the RTE code generated by SystemDesk revealed that memory consumption was virtually negligible. Access to the variables was implemented almost completely by define statements, so there was hardly any impairment to performance as compared to classic implementation.

Integrating the Application Code

Having established that RTE generation delivers sufficiently efficient code, we are currently working on developing the AUTOSAR application layer. Not only the handcoded legacy code will be reused for this, but also the entire C code that was formerly generated directly from TargetLink models. To make the code AUTOSAR-compliant, the approach we are using in this project phase is to embed both legacy and TargetLink code in AUTOSAR-compliant wrappers, using the SystemDesk automation feature, among others. Another reason for choosing this procedure is that the legacy code and the TargetLink code both have the same interface, so the same kind of

> "The numerous options provided for graphically reducing complexity make working with large-scale system models easier and faster."
>
> *Luigi Romagnoli, Magneti Marelli*

wrapper can be used. However, in the future we plan to generate both the AUTOSAR-compliant code and the software component descriptions directly from TargetLink models using TargetLink's special AUTOSAR support. We expect that this will improve resource consumption by eliminating the need for AUTOSAR-compliant wrappers, and also simplify the workflow in the interaction with SystemDesk, making the overall development process more efficient.

Conclusions and Outlook

The development tasks we have performed show that the specifications and description files for an existing engine ECU can be migrated to AUTOSAR with a reasonable amount of effort. With support from automatable tools, even complex systems and large data volumes can be handled reliably.

Initial performance measurements show that the AUTOSAR standard does not necessarily lead to increased execution time and memory consumption if the development tools are designed for the best possible efficiency. Work on the AUTOSAR-compliant architecture and the wrapper software has been completed. We are now in the implementation phase and are integrating the existing application code and the operating system. We will be presenting the completely migrated ECU in the last quarter of the year.

Alessandro Palma
Luigi Romagnoli
Walter Nesci
Manager AUTOSAR Cross-Project X-PRO
walter.nesci@magnetimarelli.com
Magneti Marelli
Italy

Glossary

ASAP2 – An ASAP2 description file containing all the information on the relevant data objects in the ECU, such as characteristics like parameters, maps, and look-up tables.

Python – Script language defined for maximum simplicity and usability.

Runnables – An executable element in an AUTOSAR SWC, comparable to a function.

Runtime Environment (RTE) – Intermediate layer that connects the software components in the application code and the basic software in AUTOSAR designs.

Scheduling – Definition of the timing of process and task execution.

Sender/receiver interfaces – Data interfaces for AUTOSAR SWCs.

Software component (SWC) – Structural element of AUTOSAR used for creating reusable software modules.

Task – A process that runs in a system.

Current Model Data

- **Software components: 172**
- **Data elements: 2650**
- **Data accesses: about 5000**
- **Tasks: 16**
- **Runnables: 624**

Summary

- An existing ECU successfully migrated to AUTOSAR
- Efficient RTE code straight from SystemDesk
- The complexity of engine management models can be handled with SystemDesk