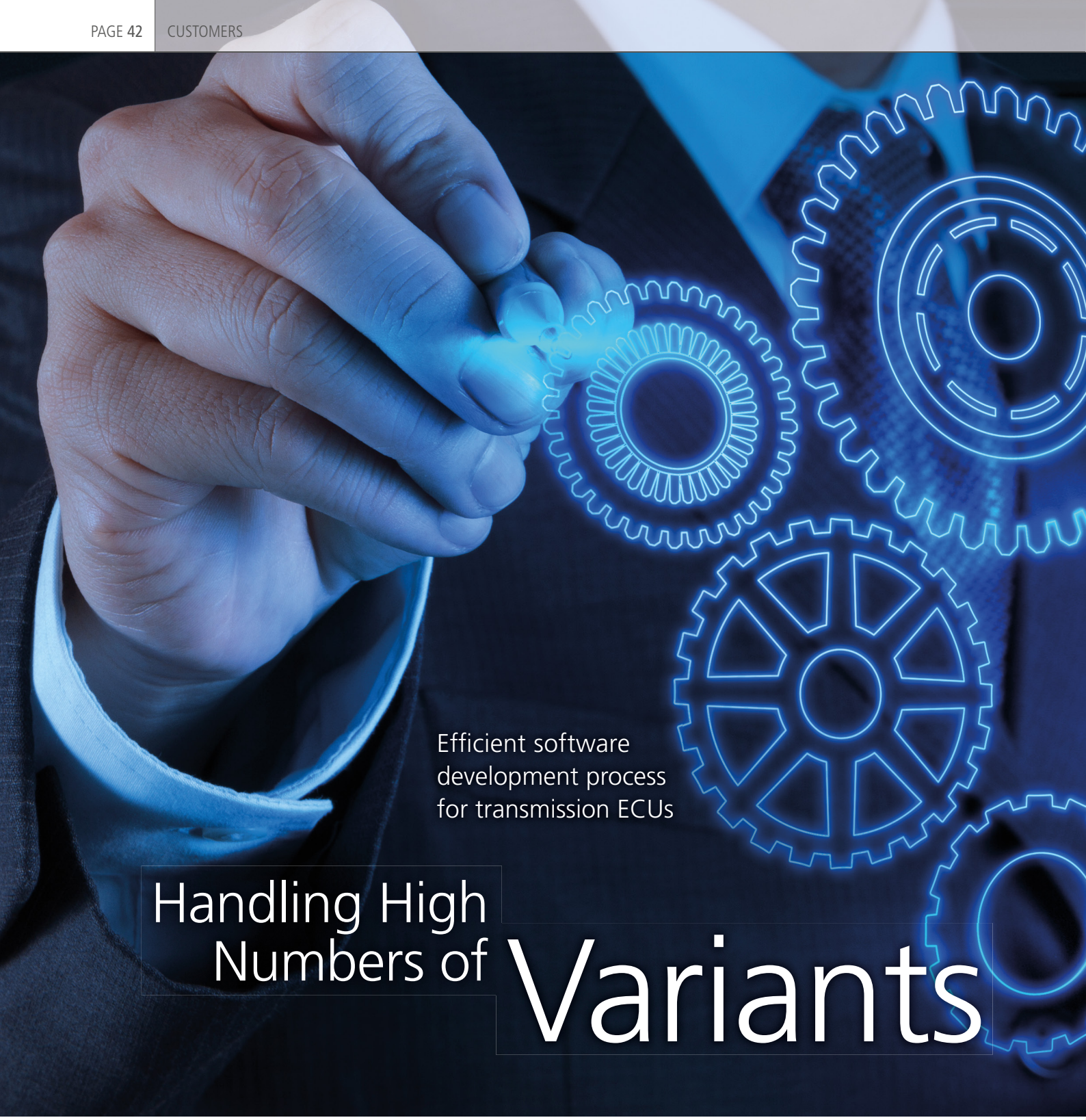


Efficient software development process for transmission ECUs

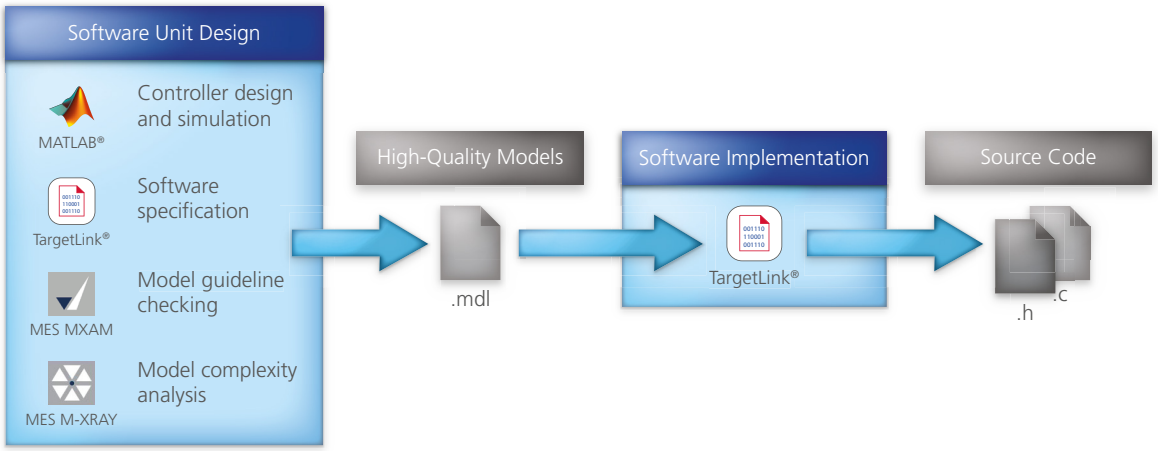
Handling High Numbers of Variants

The high number of variants is one of the many challenges that have to be tackled during the development of electronic control unit (ECU) software. That is why automotive suppliers such as ZF Friedrichshafen AG use methods that involve tool support for crucial development steps. One such tool is dSPACE's production code generator, TargetLink.

The continuous pressure for innovation in the automotive industry constantly requires new functionalities, which have to be brought to the market at ever shorter intervals. The wide range of versions and variants of current vehicle generations is therefore increasing. This trend does not stop at transmission development, where the tried-and-tested automatic transmissions with a torque converter provide a growing number of gears and cover a wide range of applications up to hybrid drives. This variety of versions and the new transmission functions create new challenges for software development. The many functions of hybrid controls, which are often networked and sometimes safety-relevant, have to be integrated in existing software environments and implemented on ECUs with limited resources in terms of memory and computing power. Handling the transmission variants and complying with ISO 26262 and other safety-relevant standards places important requirements on development methods and the development process. In addition, solutions to new functional requirements have to be included in series production fast and efficiently. Here, quality is a decisive factor.

Development Methods and the Development Process

To meet all of the requirements and make software available as soon as possible, the software development for embedded systems often uses model-based development methods. This especially holds true for the automotive industry. Here, the development method influences the entire development process, from requirements definition to software release. To fully benefit from the advantages of model-based development, there has to be an overall development concept that uses this method in all phases. There are three main goals that can be reached with model-based development: Improvements in quality, short development times, and comprehensive automation. Safety-relevant functions should also be programmed on the basis of models. This entails strict requirements for tools and processes, because they have to comply with many stringent standards and achieve a very low error rate. Efficiency is increased by model reuse, a joint tool chain, and purposeful variant management. The goal is to frontload more tasks to detect errors early on and thus shorten development times. >>



Tool chain and workflow for model-based development.

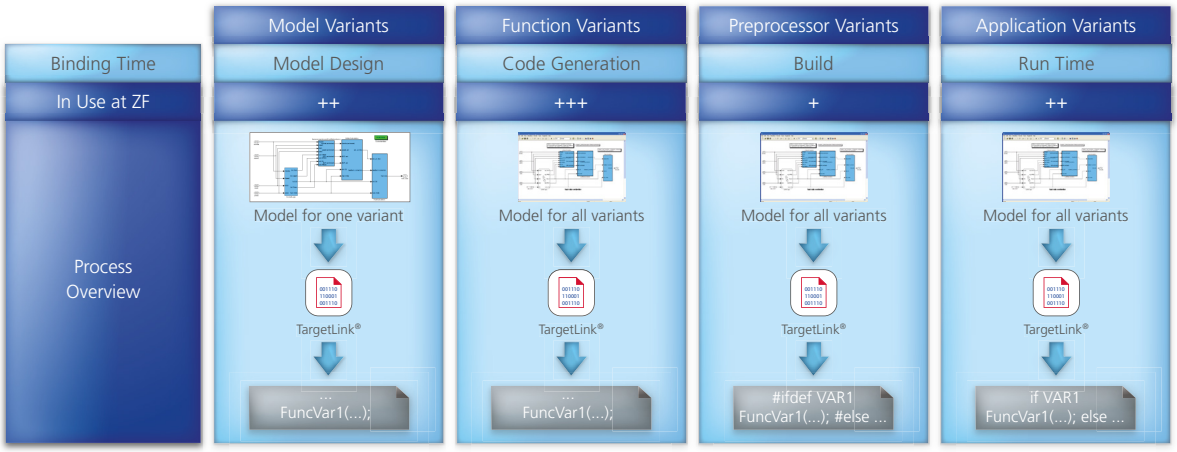
The Tool Chain

The most important tools for model-based software development are the modeling platform and the production code generator. ZF uses MATLAB®/Simulink® by MathWorks® as the modeling platform in almost all of their projects. dSPACE TargetLink® is used as the code generator. The module tests are executed with the TargetLink-integrated simulation concept and the MTest tool from Model Engineering Solutions GmbH (MES). The tools MXAM and MXRAY, also from MES, are used to perform automatic checks of compliance with modeling guidelines and to analyze the model complexity. Together, they ensure the high quality of a model. Other tools support the process during model analysis and testing as well as in the configuration and variant management. The tool chain is used throughout the company, and production code generated by TargetLink has been used in many products since the tool's introduction in 2008.

Developing Safety-Relevant Functions

Typically, QM functions (validation with traditional tools) are not the only ones to be created on the basis of models. The same applies to safety-relevant functions. One major advantage is that the production code generator TargetLink was certified according to ISO 26262 by TÜV SÜD (German international certification authority) and is therefore approved for the development of safety-relevant systems up to ASIL-D. In practice, this means that no tedious code reviews are needed, for example. Furthermore, dSPACE provides a reference workflow for the model-based software development of safety-relevant systems that ZF integrated into their

The various ways of variant handling.



standard process. To establish project-independent as well as project-specific methods, ZF use their own modeling guidelines, which are based on the prevalent standards. This ensures that the methods and recommendations of ISO 26262 are already considered in the modeling phase.

Various Ways to Handle Variants

To be able to use as many carry-over parts as possible for several variants, ZF uses various kinds of variant handling. The capabilities of the code generator, TargetLink, are very similar to those of traditional programming. However, in model-based development, these are extended by additional methods. One way to handle variants is using model variants that code is generated for selectively. The basis is always formed by an overall model that consists of multiple submodules. Most of the modules are the same for the different variants. This means, each module exists only once. However, some modules are variant-dependent. In this case, there is one module for each variant. When the overall model is built, the relevant modules are used according to the selected variant. This type of variant handling is used when the overall model is set up. Once the overall model is complete, it is no longer possible to use a different variant. Other ways to handle variants include using function variant switches and preprocessor switches. When function variant switches are used, TargetLink generates variant-dependent code parts. Software parts that are irrelevant for the selected variant are not generated. Preprocessor switches, on the other hand, work as you would expect from traditional software development. The production code generated by TargetLink contains all variants, and the variant-dependent evaluation is executed when the code is being compiled. TargetLink can also handle data variants that are controlled dynamically by the software application.

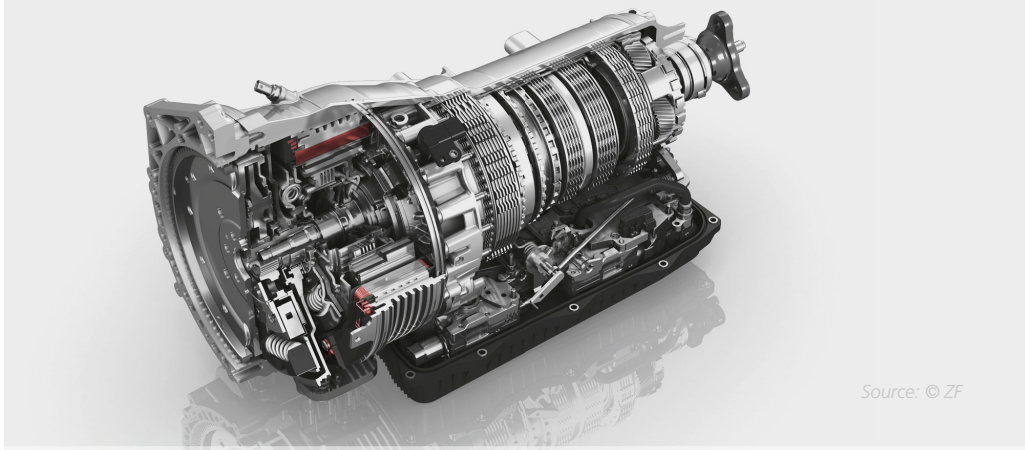
SYNECT is data management and collaboration software with a special focus on model-based development. SYNECT helps engineers handle models, signals and parameters, their dependencies, versions and variants, and the underlying requirements.

Outlook: Using Feature Management

To manage the growing complexity of diverse customer and function variants even better, many companies are thinking about using feature management systems in the future. They take various feature models to generate sets of parameter settings and use them in the TargetLink model or code. This requires a multistage approach that uses feature models for both the entire project and individual components. Because this approach is expected to be very complex and requires high traceability, many development departments are researching the possibility to integrate data management systems, such as dSPACE SYNECT®, into their tool chains and processes by coupling it to requirements management.

With the kind permission of the ZF Friedrichshafen AG.

The 8-gear plug-in hybrid drive is one variant of the 8 HP series from ZF for which production code is generated with TargetLink.



Source: © ZF