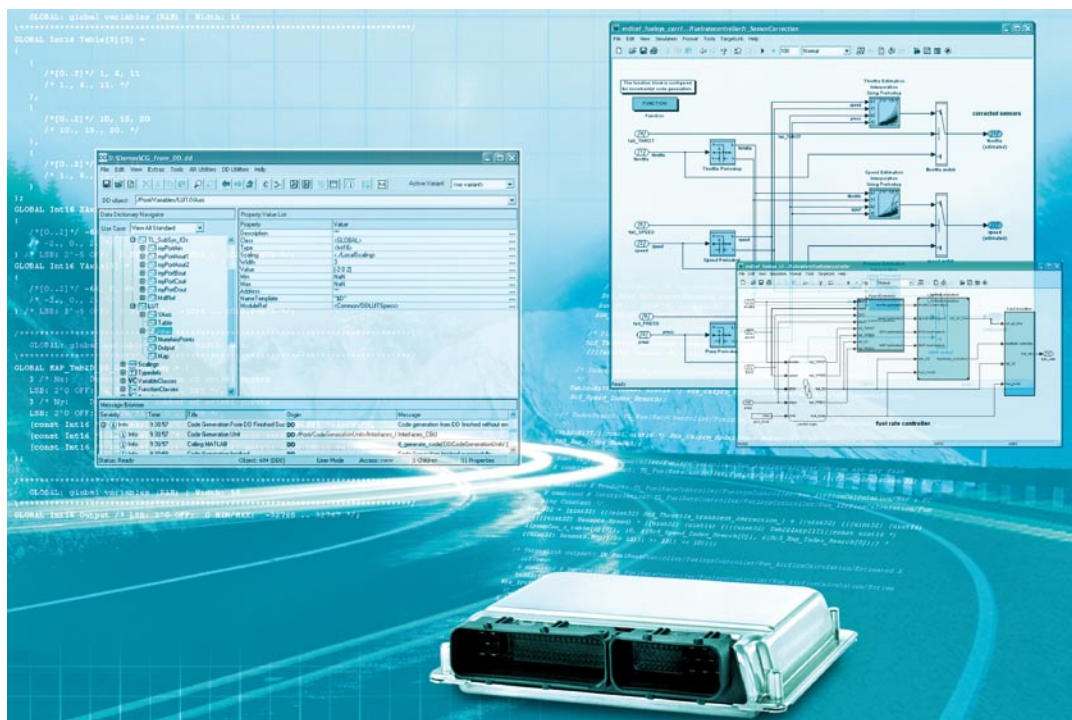


From Single Modules to Application Software



Dr. Ulrich Eisemann, dSPACE GmbH, Germany

Translation of "Module by Module – Unterstützung einer modularen Vorgehensweise bei der modell-basierten Software-Entwicklung"
Published at: Hanser Automotive 05/2011

Support for a modular approach to model-based software development

Model-based development and automatic production code generation are now widely used and increasingly popular in the automotive industry. Because large teams are using these methods more often to develop ever more extensive functionalities, the focus is shifting towards requirements for modular development in order to utilize the efficiency and quality gains on a grand scale. This article therefore shows the possibilities provided today by modern development tools to support modular, component-based development. A tool chain based on Simulink®/TargetLink is used as an example, as it is in widespread use in the automotive field.

The overall procedure in the widest sense can be broken down into the individual development steps and roles sketched in Fig. 1. An administrator or software architect defines a set of modules or components and their interfaces in order to split the overall functionality into smaller, modular units that can be

reused later in another context. These modules must then be modeled by developers, implemented by automatic code generation, and verified in a module test. Software integration can also be model-based, for example, as a higher-level integration model that is designed, implemented and finally verified by similar methods to those used for the individual modules. In a development process according to Fig. 1, the aspects of particular importance are consistent data storage, the separation of data and algorithms, module-independent and incremental code generation, and efficient testing. This will be described in detail below.

Consistent Data Management

For efficient development in large teams, data consistency and support for the single source principle are immensely important. Data with global project relevance, such as data types, scaling formulas, and also interface variables

and application parameters, have to be made accessible to several developers, and data consistency must be ensured, see Fig. 1. At the same time, each module developer must naturally handle only those variables that are relevant to the module that he/she is developing. In Simulink/TargetLink tool chains, models are always employed in conjunction with the dSPACE Data Dictionary, which holds all the data that TargetLink users need and can be filled in different ways. If the global data dictionary shown in Fig. 1 is a department-wide database, XML imports or the dSPACE Data Dictionary's API can be used to import all the relevant data consistently. If the dSPACE Data Dictionary itself is used as a global data dictionary, an additional, powerful Include mechanism is available to partition the data into developer/integrator data dictionaries. The dSPACE Data Dictionary also supports a Diff&Merge mechanism for applying changes to data transparently and safely, plus the

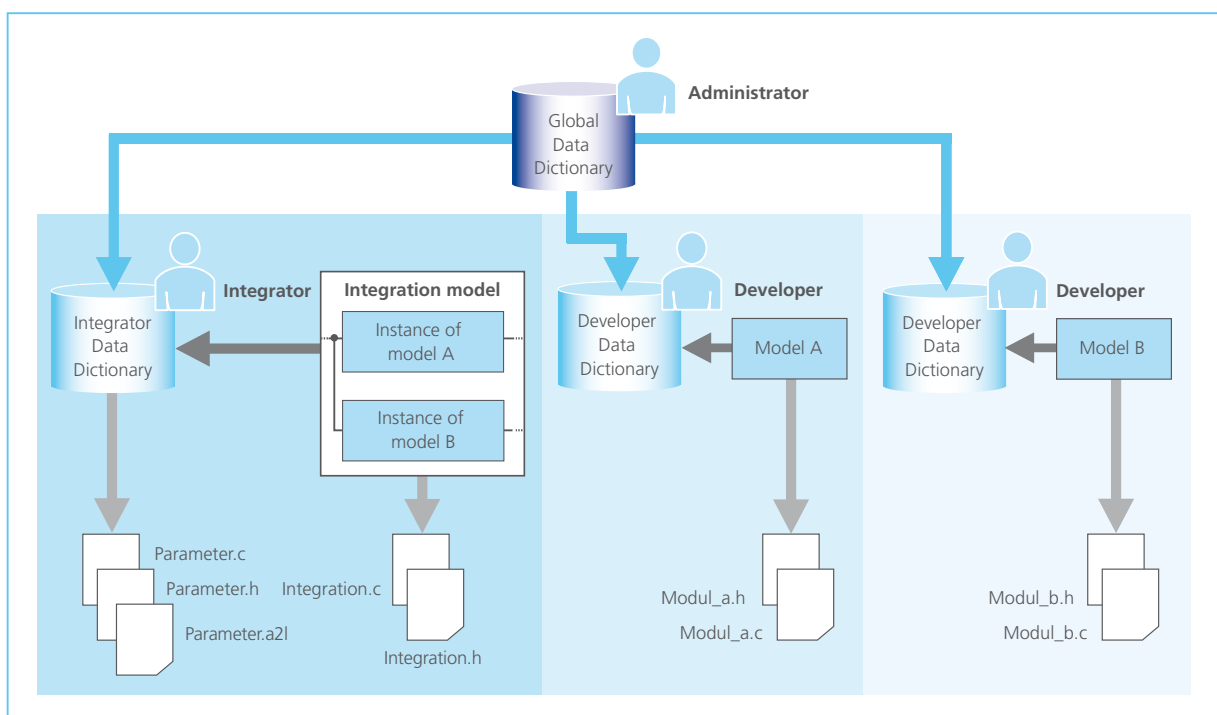


Fig. 1: Schematic of a modular, model-based development process in which individual developers implement modular units that are then integrated into the overall model.

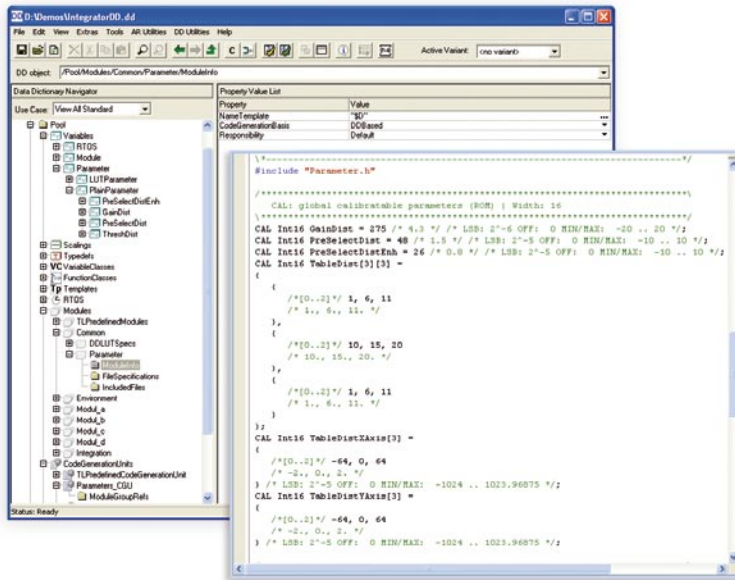


Figure 2: All the application parameters and interface variables in an ECU project can be generated directly from the data dictionary independently of any specific model.

assignment of read and write privileges for use as protection against undesired modifications.

Data Dictionary-Based Code Generation

For objects of global project relevance such as application parameters and interface variables, it is recommended to use not only separate administration, but also higher-level generation that is independent of specific models to avoid problems in software integration and software integration testing. The dSPACE Data Dictionary provides the ability to generate code and A2L files independently of individual models in accordance with the global project nature of the data. Figure 2 shows an example of how application parameters, including their data types and scaling formulas, are specified and assigned to individual modules in the data dictionary. The integrator (see Fig. 1) then generates these modules directly from the data dictionary together with the A2L files that are required for calibration. This procedure also allows all the parameters in an ECU project to be

partitioned off in a single code and A2L file, which greatly simplifies handling. The dSPACE Data Dictionary accordingly not only holds the application parameters of the model-based parts, but also enables developers to handle legacy parameters and generate them from the data dictionary.

Designing, Implementing and Testing Models and Modules

The model-based development of individual modules, which Fig. 1 outlines as an activity of individual developers, has become an established method. Nevertheless, some points need to be noted on reusing the modules and integrating them into a modular development process. One is that the design must of course be based on the specifications that are imported into the developer data dictionary. One particular result of this is clear separation between the actual algorithms of the TargetLink model and the associated parameterizations in the developer data dictionary that are referenced by the model. In addition, to simplify later software integration and reuse the functionalities,

it is recommended either to organize model parts in Simulink libraries or to design models so that they can be easily referenced later by higher-level models. TargetLink supports both of these options in conjunction with incremental code generation. TargetLink also enables module tests to be executed with different sets of application parameters without regenerating code or recompiling. This not only saves a substantial amount of time in testing, but also results in greater confidence in the quality of the module, because all the tests were performed with the same code.

Model-Based Software Integration

Despite being error-prone, purely manual software integration is still frequently used in practice. Yet in a model-based development process, this step can be automated in numerous ways, see Fig. 1. First an integration model is created with the subfunctionalities already developed, using either Simulink model libraries or so-called referenced models for the individual subfunctionalities in order to comply with the single source principle. Software integration is then performed by incremental code generation in the form of "glue code" that merely calls or links the existing functions for the submodels. This ensures that code that was already thoroughly tested in the module test phase can be reused, at either source code or object code level. Moreover, high-level generation of interface variables, parameters and other shared items from the data dictionary also avoids problems in the build process, because the allocation and assignment of the

items is clearly defined in the integrator data dictionary.

Component-Based Development According to AUTOSAR

The principles of modular development are naturally completely implemented with a standard like AUTOSAR. In this case the individual developers implement AUTOSAR software components, i.e., reusable elements that can be developed independently of one another, with well-defined interfaces and connection specifications. The techniques of model-based design for developing AUTOSAR software have recently proved to be of great value in practice. In an AUTOSAR workflow, each individual developer creates software components from the model-based design. The components are designed separately and implemented by automatic code generation, and subsequently undergo component testing. In this case, the global data dictionary shown in Fig. 1 is typically represented in an AUTOSAR architecture tool such as dSPACE SystemDesk, where a software architecture is initially created that consists of numerous components with well-defined interfaces. The architecture tool uses the XML format standardized in AUTOSAR to exchange data with TargetLink, thereby ensuring that the interfaces of the individual components are compatible. When the necessary configuration settings have been made for the Run-Time Environment (RTE), such as defining the tasks and mapping the software components to single ECUs, software integration is then performed largely automatically by the RTE generator.

Regardless of whether a project is to be conventional or AUTOSAR-compliant, model-based design and automatic code generation can be used not only at the level of single modules and small functionalities, but also on a grand scale in large development teams.



Dr. Ulrich Eisemann

studied Electrical Engineering at the RWTH Aachen University in Germany, where he subsequently received his doctorate in the field of electronic color image processing. In 2004 he switched to the automotive field by joining Product Management at dSPACE GmbH in Paderborn, Germany. He is now Product Manager for the TargetLink production code generator at dSPACE concerned with issues of model-based design, automatic code generation and AUTOSAR.

**Company Headquarters
in Germany**

dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Tel.: +49 5251 1638-0
Fax: +49 5251 16198-0
info@dspace.de

China

dSPACE Mechatronic Control
Technology (Shanghai) Co., Ltd.
Jinling Haixin Building Unit B, 25F/L
Fuzhou Road 666
200001 Shanghai
Tel.: +86 21 6391 7666
Fax: +86 21 6391 7445
infochina@dspace.com

United Kingdom

dSPACE Ltd.
Unit B7 · Beech House
Melbourn Science Park
Melbourn
Hertfordshire · SG8 6HB
Tel.: +44 1763 269 020
Fax: +44 1763 269 021
info@dspace.co.uk

Japan

dSPACE Japan K.K.
10F Gotenyama Trust Tower
4-7-35 Kitashinagawa
Shinagawa-ku
Tokyo 140-0001
Tel.: +81 3 5798 5460
Fax: +81 3 5798 5464
info@dspace.jp

France

dSPACE SARL
7 Parc Burospace
Route de Gisy
91573 Bièvres Cedex
Tel.: +33 169 355 060
Fax: +33 169 355 061
info@dspace.fr

USA and Canada

dSPACE Inc.
50131 Pontiac Trail
Wixom · MI 48393-2020
Tel.: +1 248 295 4700
Fax: +1 248 295 2950
info@dspaceinc.com