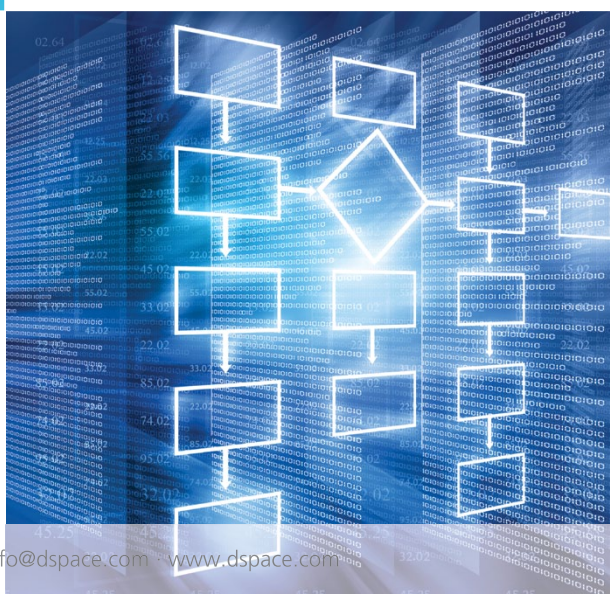
Solving the Complexity Puzzle

More and more extensive functionalities are being developed in work groups that consist of a large number of software developers from various development partners – a great challenge for the development process. A reliable tool chain for efficient, model-based software development is therefore crucial. Simulink/TargetLink and the tools by Model Engineering Solutions provide a tailor-made solution.

How to beat complexity and build consistency – even in large-scale distributed development



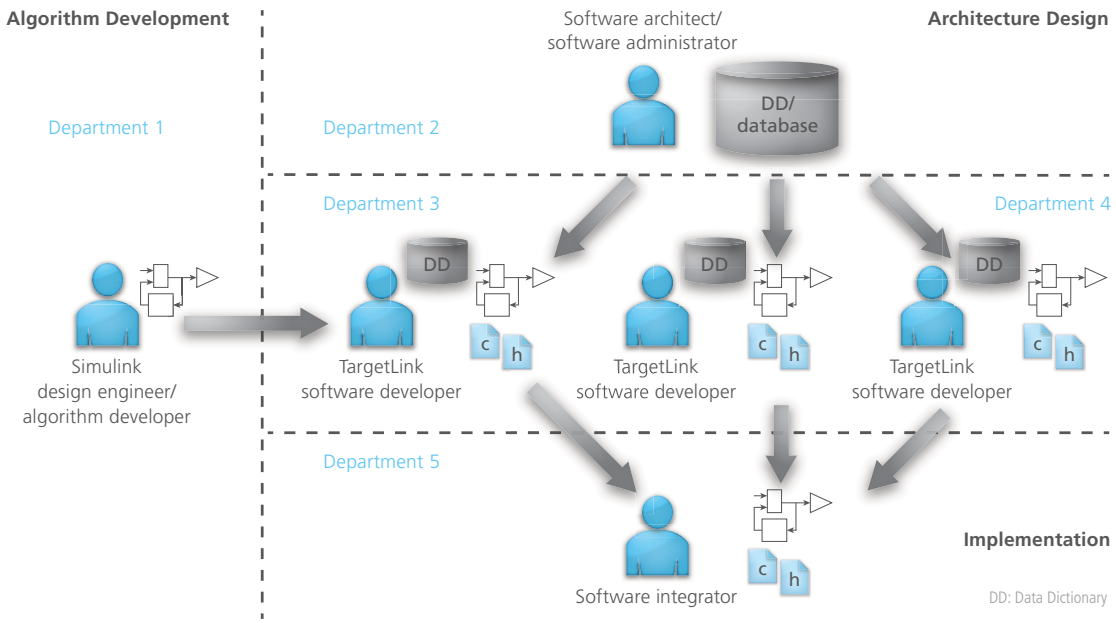


Figure 1: Distributed development in large teams. The model is defined in the domains Architecture Design and Algorithm Development and is further enriched by the development of subfunctions. The results are then aggregated, validated and implemented on an ECU.

Challenge: Distributed Development

The Simulink®/TargetLink® models, the executable specifications of the software functions, generate not only code but also other artifacts, such as A2L files, AUTOSAR XML files and software documentation. If design and automatic production code generation apply only to individual software components and functions, developers do not detect inconsistencies until they integrate the components. Often, mechanisms for testing previous development steps do not exist. This problem is becoming more and more critical because vehicle functions are increasingly complex and require the development environment to be distributed across many work groups. To make modular, distributed development of extensive functionalities efficient, developers need to adapt development mechanisms and modify a tool chain tailored to Simulink/TargetLink.

Method	Benefit
Modeling guidelines	■ Consistency ■ Lower susceptibility to errors ■ Less rework
Reuse (libraries, referenced models)	■ Modular development ■ Clarity due to model organization and hierarchy ■ Reduced development effort by reusing the same models
Single source specifications	■ Easier exchange between development team members due to software and interface specifications in the Data Dictionary
Incremental code generation	■ Quicker reviews ■ Faster code generation ■ Easier software integration and testing
Code generation from the Data Dictionary	■ Generation of shared variables in one file
Diff&Merge mechanisms via TargetLink Data Dictionary and Model Compare	■ Traceability of changes to interface definitions and the model
Complexity analysis with M-XRAY	■ Indication of appropriate model partitioning

Modeling Guidelines Improve Consistency and Mitigate Susceptibility to Errors

Simulink/Stateflow® provide many modeling possibilities, but not all of them can be used for efficient production code generation. Modeling guidelines lowering the risk of faulty models are especially important when many developers work on the same software. Adhering to these guidelines minimizes the amount of reworking needed, harmonizes modeling styles, simplifies testing and serves

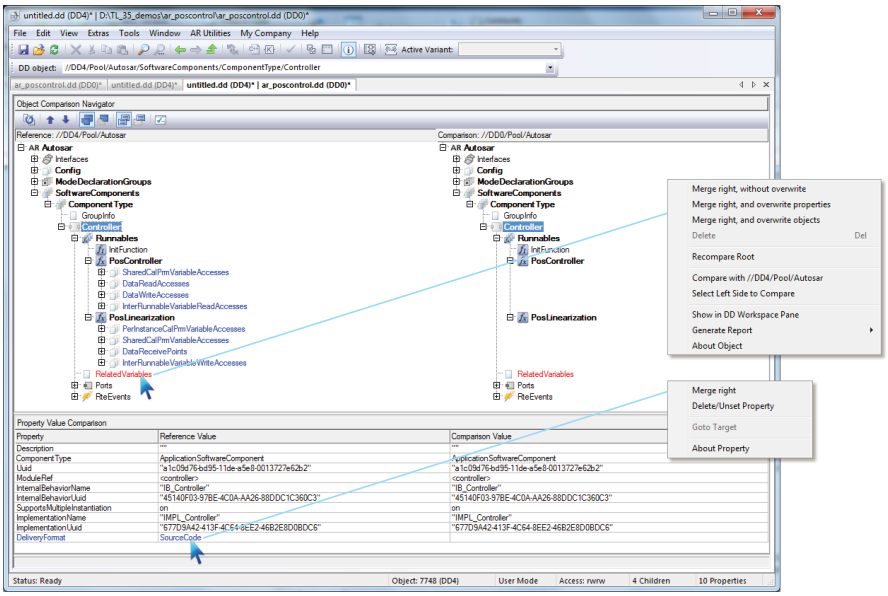


Figure 2: Comparison of different software specifications.

as a reference for reviews. It is also easier for development teams to exchange models and functionalities. Tools for automated guideline checks, such as the MES Model Examiner®, check for guideline violations and correct them.

Partitioning and Reusing Models

The single source principle is an integral part of the distributed development process. 'Single source' means that the same model is used in different development phases, from design to closed-loop control to integration. Simulink/TargetLink realize these mechanisms by:

- Using Simulink library mechanisms to reuse multi-instantiable model parts or
- Using model referencing mechanisms to integrate models into other models

Simple Exchange and Administration

In large development teams, tasks such as function development, software architecture and administration, software development and integration are rarely carried out by only one person. Rather, a large number of team members access the same information (figure 1). Since design engineers predominantly exchange, edit, and save specifications, these have to be consistent. TargetLink offers a specialized tool, the TargetLink Data Dictionary (TL-DD), which by default supports various exchange formats, such as XML or AUTOSAR XML. The data objects in the model and in the Data Dictionary are linked, so that the algorithm is separated from the data, and the

data in the model and in the Data Dictionary are synchronous.

Powerful Incremental Code Generation

Incremental code generation is another core method of distributed, model-based development. Code is generated incrementally for the individual software units. The repercussions that changes to a small function have on the overall software functions are kept to a minimum because the software units are independent from each other. Code only has to be generated for the unit that has been modified, while the rest remains unchanged.
Manual reviews are therefore less time-consuming and code generation time is held to a minimum. This makes it possible to develop large functionalities more efficiently and faster.

Generating Code from the Data Dictionary

Code is generated directly from the Data Dictionary, independently of the model, and global or shared variables from the Data Dictionary are generated to a file. This method is used when:

- A file is created that contains all global variables, such as interface variables, and specifies the access rights to them.
- All calibration parameters of different functions are generated in a single calibration parameter file
- Variables that are used in automatically or manually generated legacy code are generated in a file.

Efficient Diff&Merge Mechanisms

When developing new software, design engineers have to be able to identify changes, especially when different departments and suppliers are involved. Design engineers and integrators exchange the modified software artifacts.
A reliable tool chain that identifies modifications is therefore indispensable. TargetLink Data Dictionary has the mechanisms needed to compare different versions and display changes (figure 2). These modifications can then be traced back to the model to see their effects on it. dSPACE's Model Compare, for instance, provides convenient and comprehensive functionalities to compare models.
DD mechanisms update interface definitions automatically to ensure

Figure 3: Analysis report of M-XRAY.

Structural Overview: Main Part

Name	Complexity	*	Name	Complexity	*	Name	Complexity	*	Name	Complexity
Fuelsys	96		EGO sensor	33						
			MAP sensor	33						
			Engine speed	33						
			Engine gas dynamics	33		Mixing & Combustion	59		System lag	9
						Throttle & Manifold	42		Intake manifold	67
									Throttle	350
			Fuel rate controller	58		Airflow calculation	780			
						Fuel calculation	75		Switchable compensation	200

consistency when changes are made (figure 2).

Validating Model Architecture

When TargetLink models are used for distributed development, they have to be divided into subfunctions and subsystems. The complexity of the individual subsystems must be kept to a minimum to reduce the number of possible errors and to ensure subsystems are readable and maintainable. At the same time, this approach fulfills the requirements of safety standards, such as ISO 26262, which call for low complexity. Model complexity can be checked automatically by the MES Model Examiner AddOn M-XRAY. M-XRAY calculates and evaluates the complexity of the overall model and the individual subsystems.

In addition, it collates all the model metrics relevant for a qualitative evaluation of TargetLink models (see Excursion on model metrics). M-XRAY generates an analysis report (figure 3) that gives an overview of the model hierarchy and the complexity of each subsystem. This makes it easy to evaluate a model's complexity and to identify particularly complex subsystems. ■

Reference values for local complexity

Reference value	Evaluation
MV ≥ 750	High
MV < 750	Medium
MV < 300	Low

Summary

There are many efficient methods to control complexity and consistency in distributed development which can also be used by large development teams. Model partitioning, incremental code generation, and tools supporting change tracking are the ingredients for success. Design engineers can use measurements and metrics to evaluate the partitioning. With this approach, design engineers can develop extensive functionalities more easily and exchange the developed subfunctions more efficiently and with less errors.

Excursion

Model Metrics

Using metrics, developers can compare TargetLink models and evaluate their complexity and quality. Safety standards, such as ISO 26262, stipulate that the complexity of safety-critical models must be evaluated (see ISO 26262-6, §5.4.7, table 1). Model metrics can also be used to estimate the effort required for testing and reviews. By capturing metrics values for different development stages, developers can also monitor a model's development and identify particularly complex and error-prone model parts very early on.

Metrics such as the number of blocks, *modeling depth*, interface width, or cyclomatic complexity are also used to measure model complexity. However, these metrics are based on programming concepts and are not often suited for evaluating models. An evaluation of the cyclomatic complexity of models, for example, is not very informative due to the data flow orientation in Simulink.

The measurement of *model volume* (MV), derived from Halstead complexity measures, is establishing itself in the industry as an important way to evaluate model complexity. This measurement allows developers to evaluate model complexity because it includes not only model blocks but also the links between blocks, their weights and their own complexities, and the signals used to link blocks.

The MES Model Examiner® and the M-XRAY AddOn can be used to analyze and evaluate TargetLink models with model metrics. M-XRAY analyzes the models and calculates their volume and all relevant metrics values. It then presents the results in a compact, structured table. This tool therefore makes it possible to efficiently calculate complexity distribution in a model and keep it to a minimum. ■

Literature:
Stürmer, I., Pohlheim, H., Rogier, T.: "Calculation and Visualization of Model Complexity in Model-based Design of Safety-related Software", (in German) in Keller, B. et. al., Automotive - Safety & Security, Shaker, pp. 69-82, 2010.