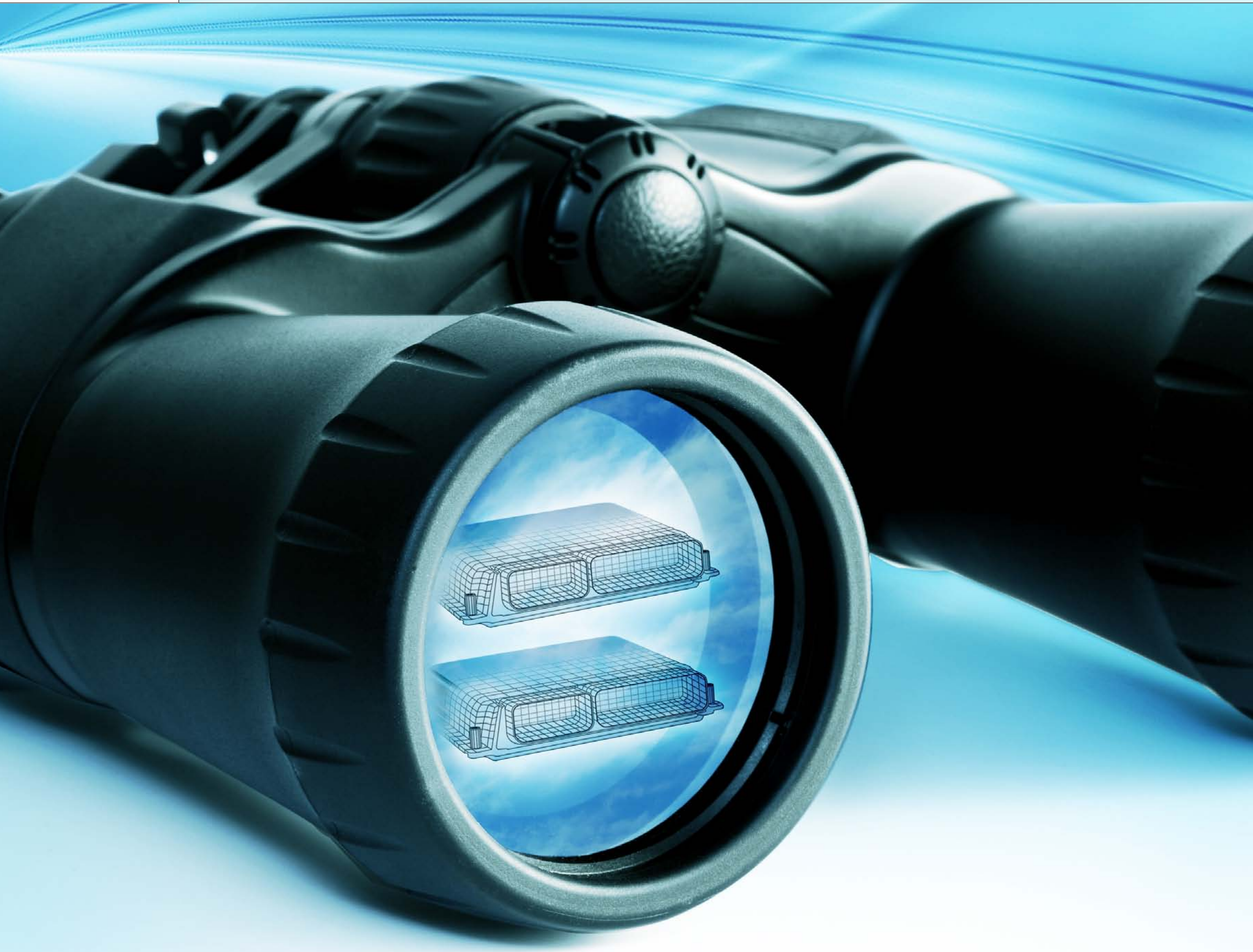
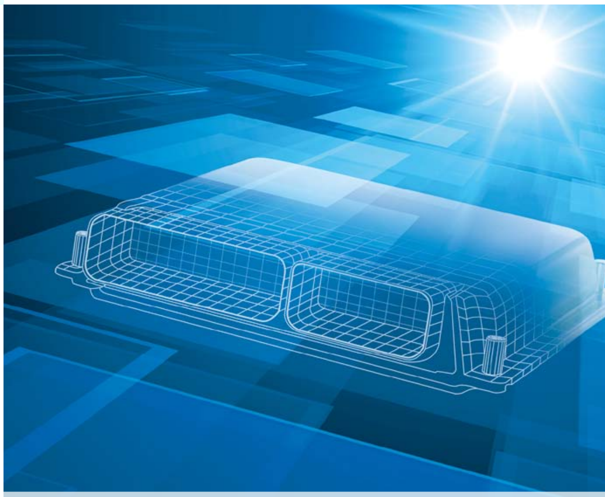


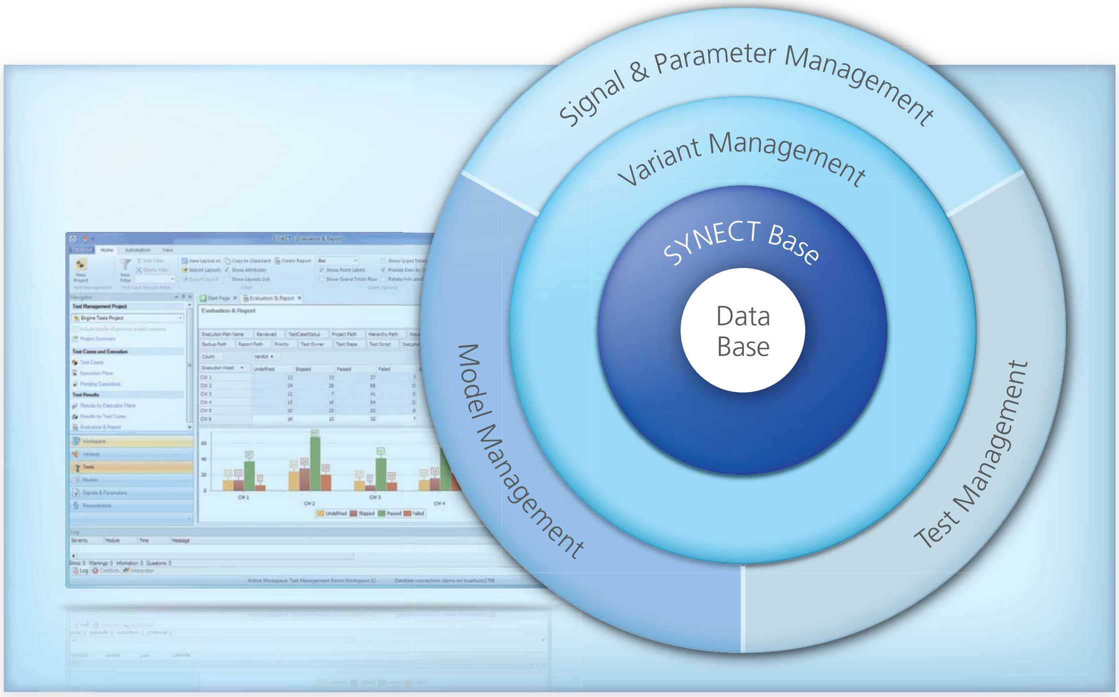
Forward

Model-based development with automatic production code generation and X-in-the-loop test methods have boosted the productivity of the automotive software development process for many years now. Development work and quality assurance for the rapidly growing number of new, software-intensive vehicle functions would be inconceivable without sophisticated tool support. The question now is: How will the development processes evolve in the future?

Ways to achieve greater productivity in the development process

Thinking





Modular data management concept with dSPACE SYNECT.

Status Quo

Today's development tools are typically not stand-alone, isolated tools used by individual developers, but are part of an extensive tool-chain, deeply integrated into the development process and networked with other tools across multiple departments and teams all over the world. Tool workflows and interactions have improved considerably over the last few years, in part because of standards such as ASAM and AUTOSAR. Given this sophistication in the development process, can productivity rise even higher? And if so, what preconditions need to be fulfilled?

First, a general comment: Higher productivity can be understood to mean "developing more functions in the same time". However, we must not forget that as the complexity of automotive systems continues to grow, it will take extra effort just to maintain the current level of productivity. Two particularly tough development challenges are: future advanced driver assistance systems (ADAS), which will prepare the way for semi-automated and fully automated driving, and electromobility. These features are especially challenging, in view of their additional requirements for functional safety. An increase in productivity, therefore, appears to be absolutely essential, even if it "only" means taking the same time to develop the same number of functions to production level as before.

Concepts for Mastering Complexity

When asked what needs to be done to cope with the complexity and costs of software development, development engineers and executives in the automotive industry usually recommend one or more of the following courses of action:

- Simulate more, i.e., include more details and more complex systems, and systematically reuse existing simulation models.
- Continuously generate ECU software versions and test them on their own and in networks.
- Begin system validation early, move more tests from the road to the laboratory, and set up a virtual validation strategy.
- Introduce strategies for reusing models, tests, software components and other data across multiple development phases and teams.

Companies are pursuing two promising approaches to implement these recommendations:

- Setting up active management for the fast-growing volume of models, tests and other data objects in model-based development (MBD). The key issues here are variant management, documentation and retrieval of models and tests, usability criteria (which model and which

Complexity can be mastered safely by active central data management.

test can be used for what), and traceability.

- Establishing a process for virtual validation, that begins with validating functions and software by PC-based simulation very early in development, and that allows models and tests to be reused later in hardware-in-the-loop (HIL) simulation.

Data Management: A Necessity

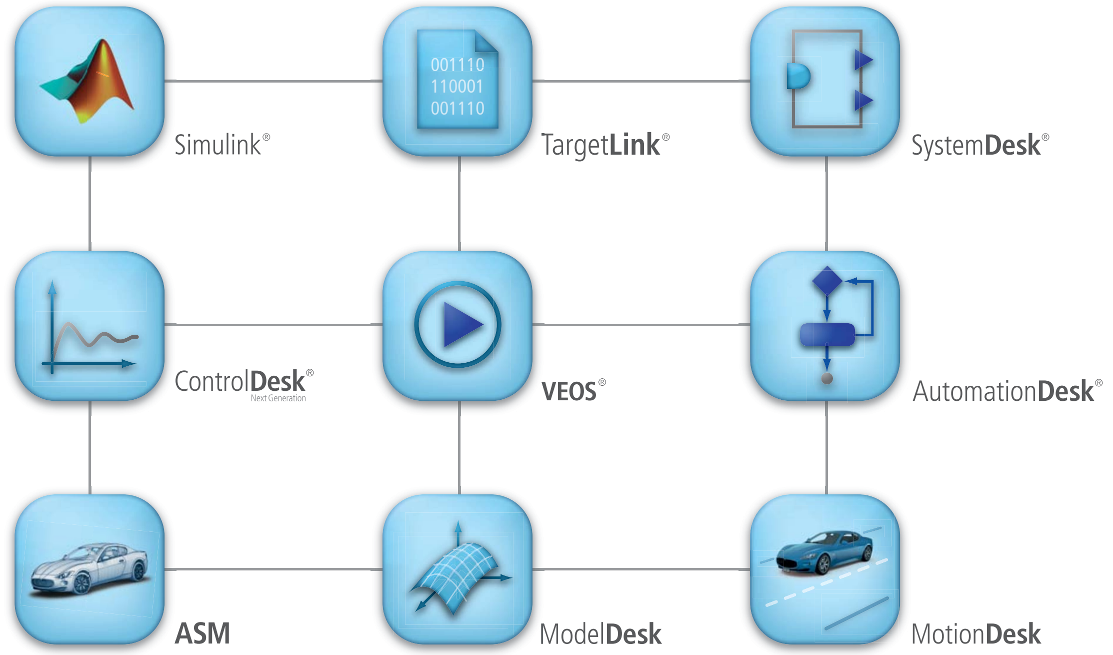
At present, data backbones for model-based development and ECU testing are an exception, rather than a norm in the IT infrastructures of OEMs and suppliers. The pressure to find a solution is revealed by what they are saying:

- "There's a risk that our engineers will sometimes lose track of things in the flood of data from different development phases."
- "We're having to work harder and harder to cope with the exploding number of software variants."
- "We need complete traceability to set up a safety-oriented development process according to ISO 26262."
- "My team spends too much time transferring data from one tool to another."
- "How can we store and retrieve specific data objects efficiently in model-based development?"
- "How can I find out which tests suit my ECU variant, so I don't waste time trying out unsuitable tests on the HIL test bench?"

Activities to find a solution to these challenges are underway in numerous companies. In many cases, this meant (and means) developing solutions in-house, often with only modest success. In-house developments very soon reveal their limitations when it comes to long-term maintenance, and when they have to be extended to meet new requirements. Existing tools for product life cycle management (PLM) and application life cycle management (ALM) cannot represent the artifacts of model-based development and the relationships between them with the necessary granularity. Function models, plant models (with varying modeling depths), source and object code, parameter sets, signal descriptions, topology/architecture descriptions, AUTOSAR objects, test scenarios, test scripts, test results and stimuli are all typical data objects of the MBD process that have to be managed with their contextual semantics.

Example of a Requirement: Data Management for Models

Simulation models are a good example of an MBD object to present requirements for a data management solution. Model properties, such as interfaces, parameters, and variant validity, as well as user-specific data, have to be stored with the necessary fine granularity so that models can be assembled for simulation, ECU testing or software builds. It will not be



Tool chain for virtual validation.

possible to reuse a model across all levels, from the integration model at vehicle level down to individual blocks in a library, unless we stop treating it as a black box and instead use modularization and a hierarchy to represent its insides. Traceability requires that a model part or a signal on any hierarchy level can be linked through to the requirement that its specification is based on.
Similarly, for test automation, it might be necessary to make the usability of a test variant conditional on the value of a specific model parameter. This goes far beyond the file-based storage of models and tests used by current configuration management systems.

Data Management in Model-Based Development

SYNECT® is a new dSPACE product currently evolving as a solution for these requirements. With integrated variant management, it supports the management of tests, models, and other entities. Engineering tools for tasks such as creating and editing models, autocoding, and test development are connected to SYNECT so that defined, consistent data versions are available for daily work and can be fed back to the data base in a controlled manner. SYNECT can also be integrated into an existing IT infrastructure. Data exchange between SYNECT and tools such as ALM/PLM will be established via interfaces like Open Services for Lifecycle Collaboration (OSLC). Bidirectional traceability can be implemented as fine-grained as necessary. For example, the relationships between requirements and items derived from requirements such as function models, tests and test results can be completely documented according to ISO 26262. The result is considerably improved data consistency, more efficient collaboration, and easier data reuse throughout the entire development process.

Validation Strategies with Virtual ECUs

To achieve a high level of validation in the virtual world, there must be

early, continuous software integration of the ECUs involved, allowing them to be tested individually or in a network as "virtual ECUs" in a PC simulation with realistic environment models. This will give engineers a feel for the performance of complex, multi-ECU functions at a very early stage. Errors in the control strategy or software implementation will be detected early (saving time and money). Because the simulation does not have to run under real-time conditions, more detailed environment models and more complex simulation processes can be used than on a HIL test bench, so functions can be optimized with maximum realism. Models are already configured, parameterized and validated on the PC. All the models, tests and data can then be reused on a HIL test bench with the same tools, so that the HIL system is no longer tied up with "unproductive" tasks such as test development.

Seamless Transition to Testing Real ECUs

Virtual ECUs can run alongside real ECUs on the HIL test bench as substitutes for real ECUs that are unavailable, or even as devices under test. All the virtual ECUs in the ECU network can gradually be replaced by real ECUs, making the transition to testing the real ECU network

Virtual validation brings time and cost benefits.

The VEOS Simulation Platform

For virtual validation, dSPACE offers VEOS®, a PC simulation platform for virtual ECUs, distributed functions and environment models. Virtual ECUs are typically generated from software components, according to the AUTOSAR standard, but can also be created directly from Simulink®/TargetLink® function models. Basic software modules such as services, the operating system, and communication stacks can be added in order to represent ECU behavior realistically. Environment models from different modeling tools are integrated via the new Functional Mock-up Interface (FMI) standard. What makes PC simulation particularly efficient and powerful is that all the test and experiment tools that are available on a HIL test bench can also be used in conjunction with VEOS. Extensive simulation runs and tests are developed in the usual tool environment and implemented and executed on a PC.

extremely smooth. This kind of integrated tool chain for testing both virtual and real ECUs enables OEMs and ECU suppliers to define and roll out new validation strategies. The potential benefit to automotive software development of this new addition to test methods has already been demonstrated in initial projects carried out with vehicle manufacturers. ■

Dr. Rainer Otterbach, dSPACE

Conclusion

With the enormous growth in the number of high-quality, software-based vehicle functions, new approaches to mastering complexity are needed in order to maintain or even increase the productivity of automotive software development in the future. Active data management and virtual validation strategies are two effective approaches that build on current state-of-the-art model-based development and have great potential for further optimizing development processes. At dSPACE, we look forward to setting out on new roads to innovation with our customers and to introducing requirements-driven solutions based on our new products SYNECT and VEOS.

Dr. Rainer Otterbach
Dr. Rainer Otterbach is Director of Product Management at dSPACE GmbH.