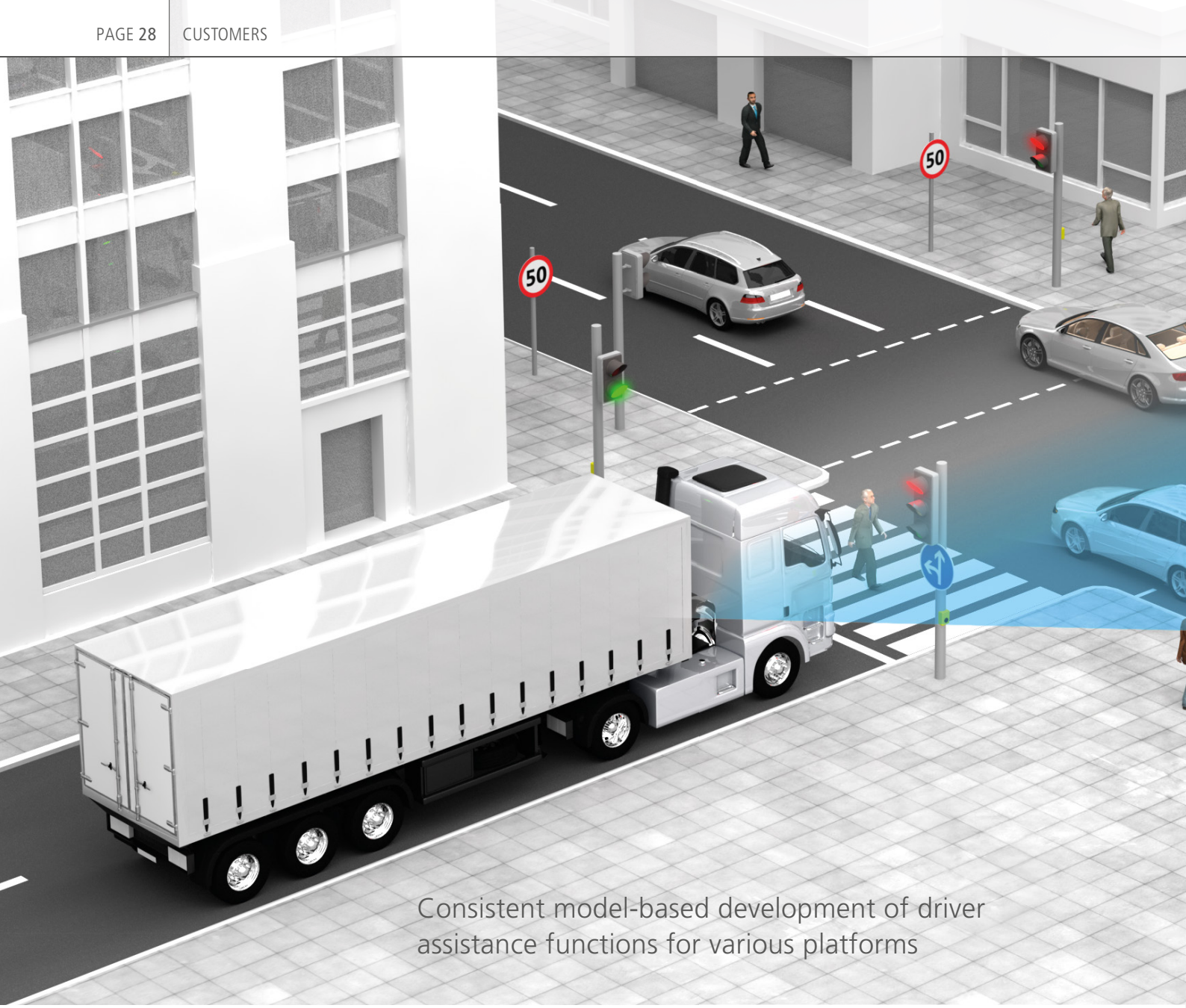
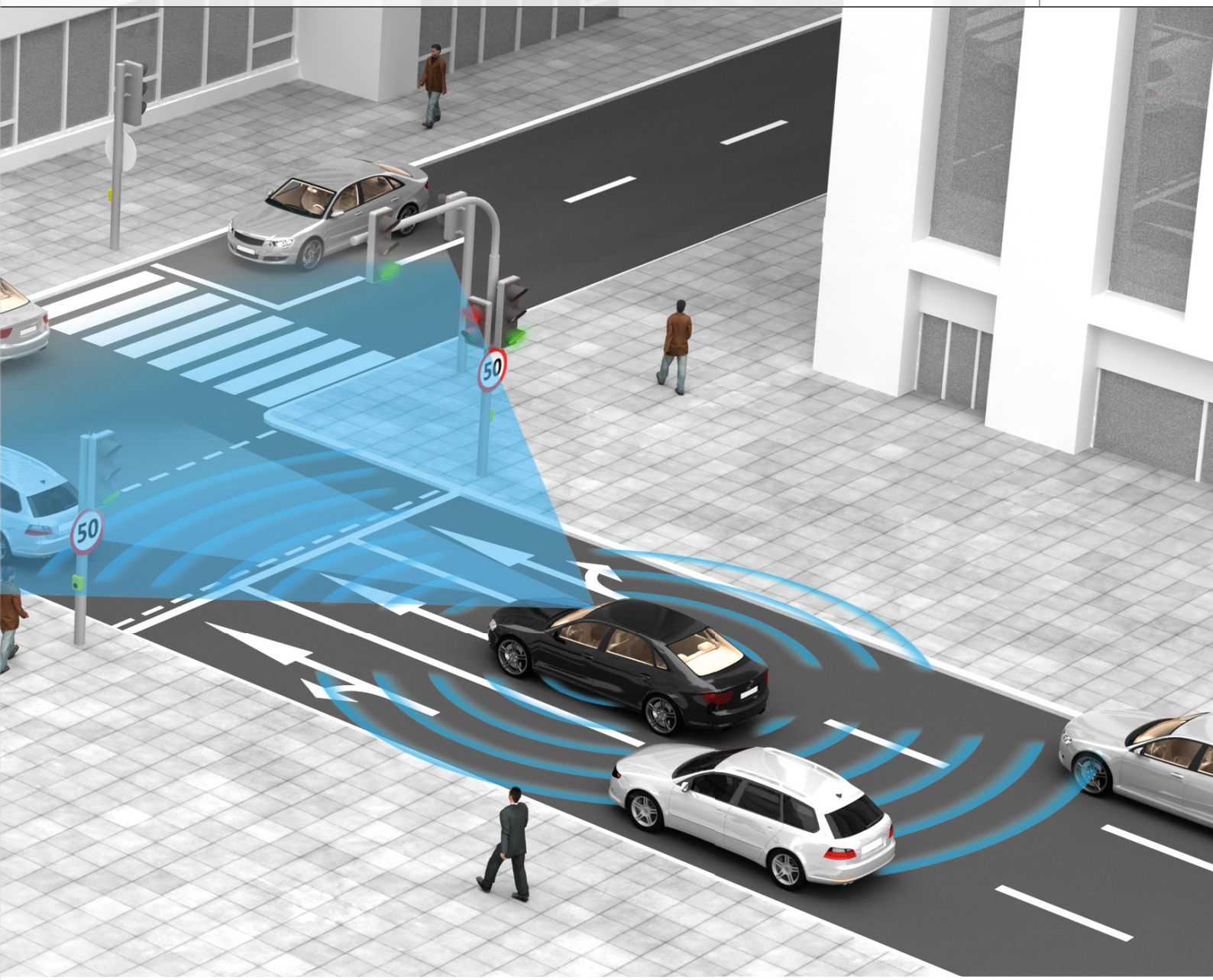


Consistent model-based development of driver assistance functions for various platforms

Developing Intelligent Assistants

In an advanced development project for sensor-based driver assistance systems at Automotive Safety Technologies GmbH, the goal is to handle and analyze complex data reliably. The production code generator TargetLink supports this task and enables a seamless, efficient workflow.

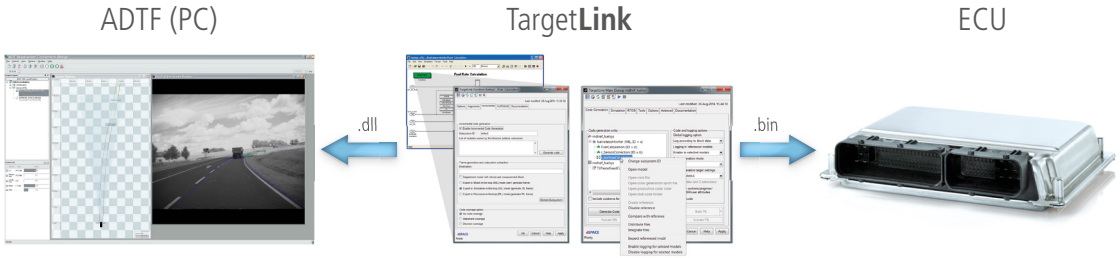


The field of sensor-based driver assistance, with its multitude of data fusions from many different sensors, generates complex algorithms and data structures, as well as large amounts of data. Advanced development projects that focus on cross traffic and intersection traffic are all about the efficient processing of:

- Data fusions from many different sensors (camera, radar, laser)
- Building objects on the basis of raw laser data
- Camera-based multi-hypothesis object tracking (of vehicles, pedestrians, etc.)

To ensure that diagnoses of the complex functions can be performed during model-based development, a high volume of function-internal information, i.e., information that is not directly accessible, must be made available for validation and testing. These bits of detailed information require additional memory and an increased run time, which is only acceptable during the development phase. After the function has been delivered, this information is not needed any longer. Instead, the focus is now on minimal memory requirements and an optimal run time. This culminates in the need for a software component (SWC) that can be created scalably in terms of its non-functional scope and compiled flexibly to match the specific application.

During the development phase, suitable target platforms are often >>



Simplified representation of integrating a software component (SWC) generated from a function model: While in the control unit the SWC is integrated as a binary file (.bin), the ADTF includes the SWC as an executable library (.dll, Windows).

not available yet and it is important to implement prototype functions quickly, without any relevance to specific platforms, especially during preliminary development projects The development environment ADTF (Automotive Data and Time-Triggered Framework) is a fitting tool for this, making it possible to run functions on a PC and connect them with other components.

Development Environment for Sensor Data

ADTF is a development environment used for synchronously recording sensor data online in the vehicle and supplying the relevant function.

ADTF is also used for playing the recorded data offline in order to let the functions run independently of the recording time. In this environment, components can be self-programmed, or existing components can be controlled on the ECU. Thus, in principle it is possible to use the same model for generating the ECU code and to create a Windows®/Linux library. Another requirement, as mentioned above, is that during the development phase a maximum amount of development-relevant, function-internal information must be accessible so that the functionality can be analyzed and understood, especially if the functionality behaves incorrectly. To achieve accessibility, though, this internal data must already be provided by the model. The component that calls the compiled model within the development environment can then analyze and visualize this data or forward it to other components.

Specifications in the TargetLink Data Dictionary to provide the desired flexibility in code generation for debug variables.

Property	Values
Description	"Variable class"
Storage	default
Scope	global
ArgClass	< >
Volatile	off
Const	off
Macro	off
Alias	off
InitAtDefinition	off
RestartFunctionName	"InitPredictionVariables" Initialization function
SectionName	"
TypePrefix	"UAS_API" Compiler switch
DeclarationStatements	{ }
UseName	on

Requirements for Technical Implementation

To implement the functions in executable ECU software, the production code generator TargetLink® by dSPACE comes into play. With this tool, code for a SWC is generated from the function model and integrated into both the controller and the development environment. It must be possible to trace and visualize a behavioral analysis of the SWC at run time for these two platforms. This happens in both run time environments (the ECU and the development environment) but differently. On the ECU, the SWC makes the predefined runtime variables accessible to an external tool via a measurement and calibration protocol (XCP). In comparison, in the development environment any SWC variables can be transferred to other programs and visualized if desired.

Since a PC-based environment does not require any memory from the SWC, unlike an ECU, the SWC can be extended by any number of further debug variables to fully exploit the debugging capabilities. The chal-

> "With TargetLink, we are implementing an automated, seamless workflow for the efficient development of high-performance driver assistance functions that supports a switchable code generation for both the development platform and the target processor."
>
> *Matthias Issbruecker, Automotive Safety Technologies GmbH*

lenge lies in leaving the ECU code largely unchanged on the one hand, and taking account of the requirements for integrating the SWC on the other, such as creating as many variables for debugging as possible and making these variables available. Another challenge is the high level of automation for integration in each target platform (Linux or Windows) to reduce the level of manual interventions during the workflow. The solution is based on modifications made in the SWC model, in the SWC database, and in appropriate scripts.

Automated Implementation with TargetLink

To achieve this automation and support the integration of TargetLink in the development process, TargetLink's callback mechanism was used. TargetLink-specific hook functions support customer-specific modifications during the compilation process. Customer-defined instructions are executed automatically when the hook functions are called, making additional manual adjustments unnecessary.
With TargetLink, it was possible to implement the additional debug variables as arrays that were either created (development environment) or omitted (ECU) during the compilation process via a compiler switch. Compilation variants can be used to develop the same code for the different development platforms and to reduce the code to the desired form during the compilation process. Since these debug variable arrays can have different widths and data types, a reusable generic solution was implemented in the form of a library. In addition, with preprocessor instructions and automatically inserted prefixes, it was possible to generate identical C code for different run-time environments (ECU, development environment for Windows or Linux), depending on the compiler switch. ■

Matthias Issbruecker, Automotive Safety Technologies GmbH,
Mohinder Pandey

Conclusion

To meet the requirements for efficient integrated development, Automotive Safety Technologies uses the production code generator TargetLink to generate target-platform-oriented code. A compiler switch is used to generate platform-specific code automatically from a function model. The code is either equipped with additional debugging variables or optimized to run on ECUs. The advantage is that the development environment ADTF is fed from the exact same function model in order to develop driver assistance functions comfortably, to generate the code for the ECUs, and to test the code under real run-time conditions.

Matthias Issbruecker
Matthias Issbruecker works on the development of intersection assistants at Automotive Safety Technologies GmbH in Gaimersheim, Germany.

Mohinder Pandey
Mohinder Pandey worked on the preliminary development of intersection assistants at Automotive Safety Technologies GmbH in Gaimersheim, Germany.