Using Automatic Code Generation for Safety-Critical System Development

By Michael Jungmann, MTU Aero Engines and Michael Beine, dSPACE GmbH

The number of safety-critical systems in vehicles is rapidly increasing. A few years ago, the failure of a computer system in a vehicle would in the worst case mean the loss of a function, but in the systems of the future, the wrong reaction to a fault may be a safety hazard for the vehicle's occupants and other road users.

To minimize the dangers of such systems, special development standards and processes have been designed for use in safety-critical applications. The established standard in automotive electronics is IEC61508. This is a generic safety standard that requires the definition of more detailed standards for specific industries and projects. Software engineering studies have shown that the RTCA DO-178B software development standard, originally defined for the aviation industry, is also a suitable detailed standard that corresponds to the IEC61508 safety standard .The software development process according to RTCA DO-178B is organized according to the well-known V-cycle (see fig. 1).

The left side of the V-cycle describes the implementation path, starting out from high-level requirements and becoming more detailed at every step through to the creation of actual production code, while the right side represents the verification path, in which each verification phase is shown opposite its corresponding implementation phase. In any study of automatic code generation, the main focus is on the coding and unit testing phases.

The unit or module test phase is for verification of the production code and of the smallest functional blocks in the executable software. The verification activities in this phase include static analysis and dynamic testing of functionality. The activities to be performed in each phase differ only very slightly in the individual standards. However, all standards have one thing in common: The objective of safety can be achieved only by systematically performing all the activities. Taken on its own, no individual step within this process allows the quality of the developed software to be assessed.

The automotive industry is increasingly using automatic code generators for software development. In contrast, they are hardly ever used in safety-critical systems. Firstly, very special requirements are imposed on the code for safety-critical systems. And secondly, many software suppliers are only just beginning to apply appropriate development standards and cannot tackle the introduction of automatic code generation at the same time.
The initial impetus for extending automatic code generation to safety-critical systems will come from software producers who have detailed experience of using the appropriate software development processes in other fields.

Through close cooperation with its parent company MTU Aero Engines GmbH, ATENA Engineering has decades of experience in developing safety-critical systems in the aviation sector to fall back on. For example, MTU Aero Engines GmbH has developed and produced the engine controllers for a number of European aviation projects, and is still actively involved in such work. The aircraft engine controllers are multichannel electronic control units (ECUs) with between 4 and 10 processors. The response times needed for control are in a range of 2 ms. RTCA DO178, along with its predecessors and project-specific derivations, has been the standard for developing such systems for many years. ATENA itself offers engineering services both to the aviation industry and to automobile manufacturers and their suppliers. This constellation means that ATENA is already in a position to apply software development standards comprehensively to safety-critical systems in automobiles.

ATENA is currently developing vehicle systems that are classified according to IEC61508 SIL3 and whose software components are up to 25,000 code lines in size. There are several reasons for using an automatic code generator in the development of such systems. Firstly, a high rate of change is anticipated for current projects, and secondly, the software design is already available as an executable specification that an automatic code generator can convert with a lower implementation error rate.

IEC61508 requires that any tools used must be either certified or proven in use, and this requirement must be addressed before using a code generator. Certification, involves undergoing a procedure under a national or international standard, which is not specified in greater detail. One applicable standard is the one mentioned above, RTCA DO-178B. This gives users two alternatives. One is to have the code generator itself certified according to RTCA DO-178B, which might enable developers to cut down on the volume of verification activities. However, such cuts would apply only to individual activities in the unit or module testing phase and are project-specific. General omission of a verification phase is not normally possible.

Certifying the code generator presupposes that a certifiable code generator exists, i.e., the development documentation for the code generator must have been created and supplied by the tool vendor according to RTCA DO-178B. Moreover, the code generator requires project-specific certification, in which the interaction between the code generator, compilation system and target processor is tested. Consequently this form of certification is configuration-specific.

Certification of the code generator must be according to the safety level that applies to the application software. This all involves an enormous workload, which is reflected in the cost of the certifiable code generator and of project-specific certification. The price of a certifiable code generator is between five and ten times higher than that of an uncertified generator. Project-specific certification involves costs of between 50,000 and 100,000. Moreover, the certification procedure for a code generator is also very time-consuming, and the maintenance cycle for such a generator is currently around two years, while approx. 3 months are sufficient to produce a corrected version for an uncertified generator. The alternative way is to subject the application software to all the prescribed verification activities as if no code generator had been used in its creation. The verification process used for this is also able to find errors that might be introduced by an automatic code generator.

ATENA decided not to use certified code generators for safety-critical systems. Instead complete verification of the application software is performed, and the verification process itself is automated as much as possible. Nevertheless, great importance is placed on the quality and reliability of the code generator that is used. Quality characteristics that do not involve the disadvantages described above, such as a development process that complies with ISO/IEC15504 (SPiCE), are regarded as important. A thorough evaluation of the available code generation tools led to the decision to use TargetLink in a SIL 3 project. The main reasons for this decision were the technical features as well as the high product quality. TargetLink, the code generator from dSPACE, is fully integrated into MATLAB® and allows reliable conversion into C code of software designs that are available as Simulink®/Stateflow® models.

The correctness of the conversion can be tested by means of simulation and comparison with results from reference simulations, frequently referred to as model-in-the-loop simulations (MIL). This verification can be performed stepwise:

First offline simulation is performed on the host PC; this is also known as software-in-the-loop simulation (SIL).
Then simulation is performed in real time on the target processor with the actual compiler integrated; this is also called processor-in-the-loop simulation (PIL). The user has full control over file and function partitioning. This allows code for specific model parts to be generated in separate C functions and files. These model parts can also be implemented and tested incrementally. This has the advantage that when changes are made, it is not necessary to regenerate and test the code for model parts and software modules that have not changed.

SIL and PIL simulation are complemented by code coverage analysis. The coverage types currently supported are statement and decision coverage. All this means that extensive support is given to the unit or module testing phase, including model parts that are generated incrementally. All in all, TargetLink offers comprehensive configuration options to meet the requirements for code in safety-critical systems. Special attention is paid to quality assurance in TargetLink product development.

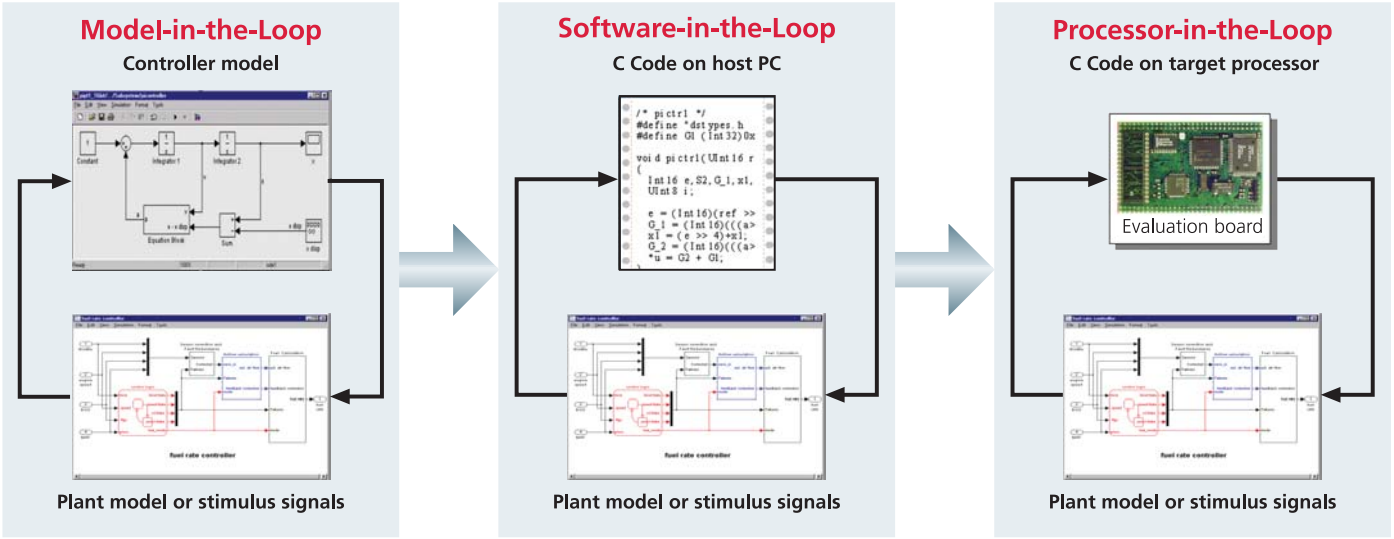
For maximum reliability and quality, comprehensive tests are performed before each release of a TargetLink version.

- Automatic test engine run: Several hundred thousand code patterns are tested and several million test points are run through. These tests are repeated for every processor that is supported separately.

- Automatic test suite run: This consists of far above a thousand models. Different input values are used and parameters are varied to produce more than a hundred thousand test cases, whose results are compared with expected values.

- Semiautomatic system test: This is to check the installation and correct functioning in different PC configurations and in interaction with different MATLAB and compiler versions.

- Manual testing with customer models: Tests are performed using large, real ECU models and evaluated manually.



In addition, before official product release a beta test phase is carried out with selected customers. When TargetLink is finally delivered to customers, it is already to a certain extent 'proven in use'.
Furthermore particular emphasis is placed on a mature development process for developing TargetLink. TargetLink is developed according to ISO/IEC15504 (SPiCE), as required by HIS, the German manufacturers' initiative for software. Compliance is monitored by an independent auditor. The extent of auditing is also specified by HIS. After all, both the development process and the release tests are supervised by an independent in-house software quality management team.

In order to integrate TargetLink into the software development process, ATENA made several adaptations to the code generation process. These were aimed at further automating the generation process and achieving the necessary software quality for safety-critical applications. They were supported by the TargetLink API, which allows all the processes to be automated, while at the same time providing options for intervention in the individual process phases and access to all TargetLink properties and settings.

The software development process, whose implementation phase is being given decisive support by dSPACE's TargetLink, has now been in use at ATENA since November 2002. Automatic code generation plays a major role, as the company has succeeded in generating 80% of the entire production code from Simulink models. The code generator is embedded in a project-specific tool chain. This guarantees compliance with the quality criteria for safety-critical applications. ATENA and dSPACE are cooperating closely to extend integration into the tool chain in future versions of TargetLink and to reinforce the support given to safety-critical aspects of code generation.