Code Generation for Safety-Critical Systems

Michael Beine
dSPACE GmbH, Paderborn, Germany

Michael Jungmann
MTU Aero Engines GmbH, Munich, Germany

Introduction

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 cope with rising demands, such as the growing number of electronic systems in a vehicle, increasing complexity and shorter time-to-market, the automotive industry is increasingly adopting model-based design methods and using automatic code generators for software development.

In contrast, automatic code generators are hardly ever used for the development of safety-critical systems. Firstly, very special requirements are imposed on the code for safety-critical systems. Secondly, many software suppliers are only just beginning to apply appropriate development standards, and so they cannot tackle the introduction of automatic code generation at the same time.

However especially the high complexity and functional requirements of safety-critical systems demand the use of modern tools for developing, designing, implementing, verifying and validating such systems.

It is natural to rely on experiences from the aviation industry when developing safety-critical systems for automotive applications. In the aviation industry for several decades programmable systems have been used for flight control, aircraft engine control, landing gear control etc.. The safety and reliability requirements of these systems are comparable with steer-by-wire or brake-by-wire systems which are currently under development in the automotive industry.

Safety Standards for Safety-Critical Systems

To minimize the dangers of safety-critical systems, special development standards and processes have been designed for use in such applications.

All safety standards define a set of activities that have to be carried out in order to achieve a desired safety level. Those activities can be generally grouped into three categories: selecting development methods and tools, implementing the system, and verifying and validating the system.

The standards differ regarding their perspective on the system to be developed. Some standards cover the development of the overall system. Among them are IEC 61508 and ARINC 653. Other standards like RTCA DO-178B and DoD-2167A only deal with the development of the software, but are more detailed.

The established standard in automotive electronics is IEC 61508 [1]. 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 [2] software development standard, originally defined for the aviation industry, is also a suitable detailed standard that corresponds to the IEC 61508 safety standard [3 et al].

A Suitable Software Development Process

Most of these standards follow the well-known V-cycle, since the number of listed requirements barely leaves room for choice. Figure 1 shows the software development process for safety-critical systems organized according to the V-cycle.

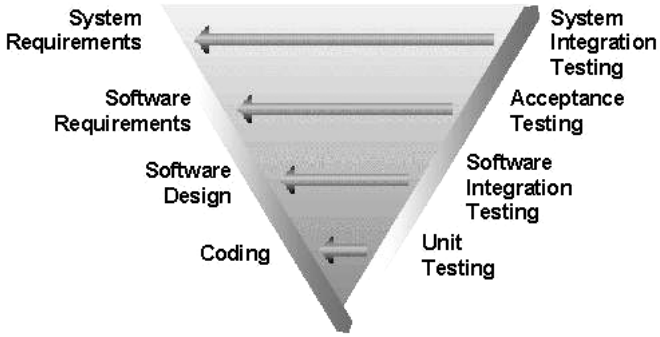

Figure 1: The V-Model for software development

The left side of this V-cycle describes the implementation path. It starts out from high-level requirements and becomes more detailed at every step through to the creation of actual production code. The right side represents the verification path, in which each verification phase is shown opposite its corresponding implementation phase.

Model-Based Design

Using model-based design methods and tools for implementation has tangible benefits. System requirements analysis is the first step on the implementation path. All functional and non-functional requirements for the system are expressed. Using model-based design methods and tools for implementation results in an executable specification, which has many advantages over a purely textual specification. A model-based specification meets the requirement of IEC 61508 to use a semi-formal method for specification, which is required because it leaves less room for interpretation than a textual one, thus reducing the possibility of misinterpretation and error. It allows a seamless continuation of the implementation path and naturally connects its different phases. With model-based design the system behaviour can be assessed and tested in every phase of the implementation, allowing to find and fix undesired behaviour and errors early on.

Verification Activities

The first phase of the verification path is the unit or module test phase to verify the production code and the smallest functional blocks in the executable software. The activities in this phase include static analysis and dynamic testing of functionality. Software integration testing, acceptance testing and system integration testing complete the verification path.

The activities to be performed in each phase differ only 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.

Role and Benefits of Automatic Code Generation

In any study of automatic code generation the focus is on the coding and the unit testing phase. Coding marks a cornerstone in the software development process; automatic code generation is the key technology in this development phase. There are several reasons for using an automatic code generator in the development of safety-critical systems.

Using automatic code generation is natural once the software design is already available as an executable specification. A code generator can convert this executable specification with a lower implementation error rate than a human programmer. Every manual translation step is prone to errors and is time consuming. The potential for introducing errors is high, especially when going through iterations and making successive changes, and consistency between the software design specification and the code is often lost. This is not the case when using automatic code generation. A code generator translates the model and changes made to the model to code reliably, reproducibly, and constantly day after day. It ensures that the generated code is always consistent with the software design. Furthermore, since the documentation can typically also be generated along with the code, it is easily kept up to date as well. Thus with automatic code generation the software design, the implementation and the documentation are automatically kept in sync.

Using a code generator is no guarantee for getting error-free software. There is no formal proof that there will never be a coding error, since there is an infinite number of combinations of modeling constructs and parameters. On the other hand, a well designed and tested code generator produces significantly less programming errors than human software programmers do. While software programmers might have good and bad days, the automatic code generator performs its task with the same level of quality every day. The code generator learns with every improvement made and from every bug that has been fixed and never forgets.

Quality and Reliability Aspects

The quality and reliability of any tools used for the development of safety-critical systems are of special importance.

Pros and Cons of Tool Certification

IEC 61508 requires that any tools used must be either certified, or proven in use, or that a justification for their use must be given. Certification involves undergoing a procedure under a national or international standard, which is not specified in greater detail. One applicable standard is the above mentioned 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 at the same level that is applicable for the application software. Moreover, the code generator requires a system-specific certification, in which the interaction between the code generator, compilation system and target processor is tested.

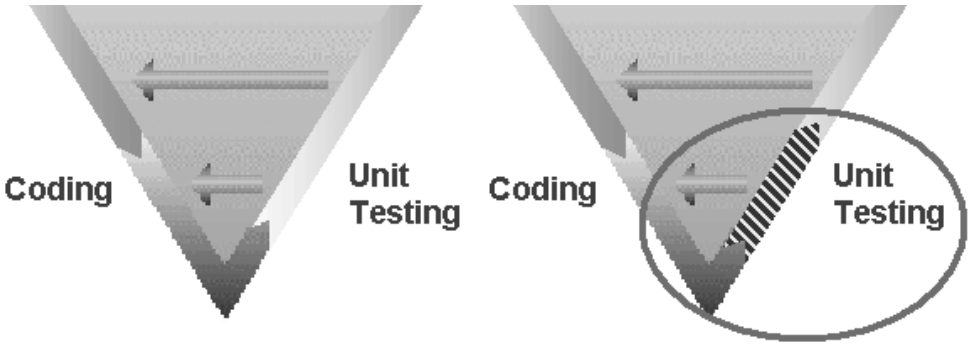

Figure 3: Work reduction potential when using a certified code generator

This all involves an enormous workload, which is reflected in the cost of a 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 one. Project-specific certification involves extra costs, which can be 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 approximately three months are sufficient to produce a corrected version for an uncertified code 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 can also find errors that might be introduced by an automatic code generator. This approach does not involve the disadvantages described above, and is also more cost effective for most projects.

Conformance to Software Quality Standards

Systematic development and system testing of ECU software is of paramount importance. The “Hersteller Initiative Software” (HIS), a German automotive OEM initiative consisting of Audi, BMW, DaimlerChrysler, Porsche, and Volkswagen, demands that ECU development and the development of major software tools comply with ISO/IEC 15504, also known as SPICE (“Software Process Improvement Capability Determination”) [4], [5].

SPICE has been defined by a working group of ISO/IEC on the basis of existing quality standards and combines ideas from ISO 9001 and CMM. SPICE requires that formal process assessments be performed by an independent organization on a regular basis. The assessment results are then used to derive process improvement activities. When the standard is followed and a high level of compliance is achieved there is a good chance that the software produced will be of high quality.

Since a code generation tool is a complex piece of software and has a substantial impact on ECU code, this demand is valid also for a code generator.

Code Generator Requirements

Using automatic code generation for the development of safety-related software imposes several requirements on the code generator.

Code Requirements

For example multiple and very special requirements are imposed on the source code for safety-critical systems. Among them are

- the restriction to a subset of the programming language used which is deemed safe,
- restricting control and dataflow structures to precisely specified variants,
- following accurately specified rules regarding the scope of functions and data,
- following predefined complexity measures, and
- the readability, maintainability, and testability of the source code.

Such demands on the code quality are defined in the MISRA C guidelines [6]. This is a generally accessible standard for the use of C in ECU projects up to SIL 3 produced by the British „Motor Industry Software Reliability Association“. Its title is “Guidelines for the Use of the C Language in Vehicle-Based Software” but it is commonly referred to as “MISRA C”. Depending on the certification process chosen, these requirements also have to be met by the automatically generated code.

Furthermore there is the demand that the generated code must not use significantly more memory and execution time than hand code. Missing efficiency for a long time has been the reason why automatic code generation has not been applied in production.

Process Integration

To fully enjoy the benefits of automatic code generation, the code generator has to be optimally integrated into the development process and tool chain in use. Seamless integration and interaction with the model-based design tool is self-explanatory. Furthermore open interfaces and the possibility for tool automation are useful

to connect with other tools, automate work steps, and to prepare and support later development phases, e.g. the succeeding unit and module test.

The Production Code Generator TargetLink

In the following the production code generator TargetLink is used as an example.

TargetLink for Safety-Critical Systems Development

The design and simulation tools most widely used in the automotive industry are MATLAB®, Simulink® and Stateflow®. Data flow models are described in Simulink, while Stateflow is used for the control flow parts. Simulink and Stateflow are integrated, and allow a mix of both types in one model. TargetLink, the production code generator from dSPACE, is seamlessly integrated into MATLAB and allows reliable conversion into C code of software designs that are available as Simulink/Stateflow models.

A major advantage of using model-based design methods and tools is the capability of early verification by means of simulation. TargetLink supports different simulation modes which allow the correctness of the implementation, i.e. the code generated by TargetLink, to be tested directly. This is done by comparing simulation results with results from reference simulations, frequently referred to as model-in-the-loop simulations (MIL). This verification can be performed stepwise:

- First, the generated code is compiled with a host compiler and executed on the host PC; this is also known as software-in-the-loop simulation (SIL).
- Then simulation is performed on an evaluation board equipped with the target processor. Therefore the generated code is compiled with the actual target compiler; this simulation mode is called processor-in-the-loop simulation (PIL).

In all simulation modes TargetLink allows signal traces of block outputs to be logged. These signal traces can be saved and plotted on top of each other, thus providing direct visual feedback and allowing further analysis.

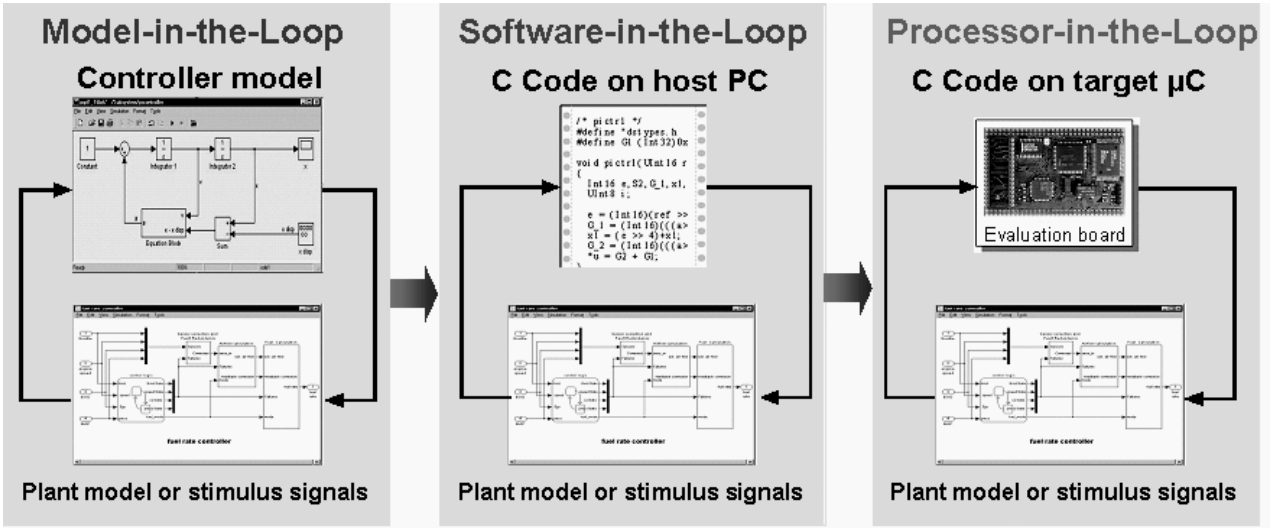

Figure 4: Model-, Software- and Processor-in-the-Loop simulation for software verification

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.

The code generated by TargetLink can well match human programmers efficiency in terms of memory consumption and execution speed since TargetLink has been specifically designed for production coding. Numerous benchmarks and user experience reports show that in most cases TargetLink is within a narrow range of what the human programmers produce.

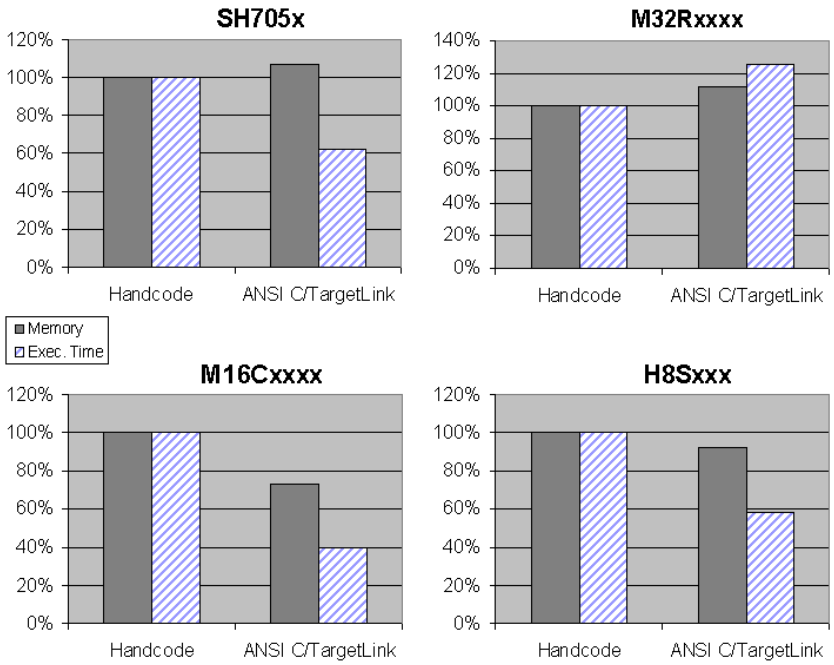

Figure 5: Customer benchmark comparing TargetLink code with handwritten code on different target processors

In addition TargetLink code is easily readable and has many comments, enabling the user to go back from the code to the model. Unnecessary macros, function calls, cryptic naming etc. are avoided. Comprehensive configuration options give the user full control over variable, function and file naming, as well as the flexibility of partitioning the code into functions and files to keep the structure logical and manageable. Thus TargetLink code can be subjected to code inspections and reviews.

Furthermore it complies with most of the 127 MISRA rules. MISRA expressly permits deviations from the standard if they are technically justified and documented. Such a compliance document that describes deviations from the MISRA standard is published by dSPACE [8].

A comprehensive and fully documented API grants access to all TargetLink properties and settings. It allows all processes to be automated, while at the same time providing options for intervention in the individual process phases. For example, so called “hook functions” allow user tasks to be performed at all stages of the build process.

Software Quality Assurance for TargetLink

Special attention is paid to quality assurance during 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 test is run on several 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.

A beta test phase is carried out with selected customers before official product release. When TargetLink is delivered to customers, it is already to a certain extent 'proven in use'.

Particular emphasis is placed on a mature development process for developing TargetLink. To secure, improve and certify systematic development TargetLink is developed according to SPICE, as required by HIS, the German manufacturers' initiative for software. Compliance is monitored by an independent auditor. The scope of auditing is also specified by HIS.

Additionally, both the development process and the release tests are supervised by an independent in-house software quality management team.

User Experiences

TargetLink has been used for many control applications worldwide. TargetLink code is in production in many applications, some of which are safety-related. For example TargetLink has been used to develop a cabin pressure control system. Therefore TargetLink generated code was certified according to RTCA DO-178B Level A, the highest safety level, meaning that a failure would have catastrophic consequences for the aircraft [7].

In the following the use of TargetLink at ATENA Engineering for the development of safety-critical software is described.

Experiences with Safety-Critical Systems and Automatic Code Generation at ATENA

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 multi-channel electronic control units (ECUs) with between 4 and 10 processors. The response times needed for control are in a range of 2 ms. RTCA DO-178A [9], along with its predecessors and project-specific derivations, has been the standard for developing such systems for many years. ATENA as a subsidiary of MTU Aero Engines offers engineering services both to the aviation industry and to automobile manufacturers and their suppliers. This constellation means that ATENA is in a position to apply software development standards comprehensively to safety-critical systems in automobiles.

MTU Aero Engines as well as ATENA have had comprehensive experiences with automatic code generation. Both have performed detailed evaluations of different code generators in the past. The focus was on the applicability for the development of safety-critical systems according to RTCA DO-178B Level A and IEC 61508 SIL3 respectively. Central points of the evaluations were:

- Integration into the overall development process
- Quality and efficiency of the generated code
- Flexible and comprehensive configuration options of the code generator

Application of TargetLink at ATENA

Because of the disadvantages described earlier, 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 used. Quality characteristics that do not involve the disadvantages de-

scribed above, such as a development process that complies with ISO/IEC15504 (SPiCE), are regarded as important.

At ATENA model-based design using the MathWorks tools MATLAB, Simulink, Stateflow was already established. Therefore a search has been made for a code generator that integrates well with this modeling tool suite.

A thorough evaluation of the available code generation tools led to the decision to use TargetLink for the software implementation of a safety-critical system according to IEC 61508 SIL3. It is an automotive application related to alternative fuel concepts. The volume of the related Simulink model is approximately 120,000 subsystems. The software components are up to 25,000 code lines in size. The main reasons for this decision were TargetLink's technical features as well as its high product quality.

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. The adaptations included the greatly reduced use of pointers and interrupts, compliance with various complexity criteria and replacement of "function-like" macros and functions from the standard libraries by user-defined functions. Work was also performed on enabling such complex systems to be broken down into subsystems and translated separately, and on allowing generation of standardized description files for calibration tools for the subsystems. The test frame software for testing the individual modules was also automatically generated for preparing the unit testing phase. The adaptations were supported by the TargetLink API. It allows all the processes to be automated, while at the same time provides 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 up to 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.

Recently, ATENA delivered the final product for the first development phase for integration tests at the OEM's facilities. From this first development phase the following experience was gained:

- The model-based requirement analysis and design phase led to an executable specification which allowed early system verification. This resulted in a mature specification. Later software tests revealed only few errors that originated from the specification.
- The automatic code generation allowed the implementation of the software design with a low error rate and with high efficiency especially for minor modifications.
- The source code generated by ATENA's software development process with dSPACE's TargetLink as a key item is of acceptable quality for safety-critical applications.

Based on the evaluation results also made at MTU Aero Engines and the project experiences made at ATENA, it is planned to use TargetLink also for future avionics projects at MTU Aero Engines.

Summary

Automatic code generation complementing model-based design provides many benefits for the user and can be applied for the development of safety-critical systems. It is suitable for software that has to comply with safety standards when embedded into an adequate development process. There are several requirements imposed on the code generator relating to quality and technical features that have to be met.

TargetLink, the production code generator from dSPACE, meets these requirements and therefore is also a suitable code generator for the development of safety-critical systems.

ATENA Engineering opted for TargetLink due to its technical features and quality level. At the same time ATENA decided to use a code generator with fewer qualification options, but to perform complete qualification

on application software. This alternative is considered to be more cost effective than using a code generator which is fully qualified.

References

1. IEC 61508 Functional Safety of electrical/electronic/programmable electronic safety related systems, IEC 1998
2. RTCA/DO-178B Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc., 1st Dec 1992
3. Bauer,C.; Plawecki,D.: IEC 61508, Part 3 vs. RTCA/DO-178B A comparative Study. Konferenz Anwendung des internationalen Standards IEC 61508 in der Praxis, Januar 2003
4. ISO/IEC TR 15504:1998, Information Technology - Software Process Assessment
5. Wagner, Merkle, Bortolazzi, Marx, Lange: Hersteller Initiative Software, Automotive Electronics 1/2003
6. MISRA Guidelines for the Use of the C Language in Vehicle Based Systems, April 1998
7. Aloui, Andreas: C Code Reaches New Heights at Nord-Micro, dSPACE Newsletter, 1/2002
8. Thomsen, T.: Integration of International Standards for Production Code Generation, SAE Technical Paper 03AE-32, 2003
9. RTCA/DO-178A Software Considerations in Airborne Systems and Equipment Certification, RTCA Inc., 22nd Mar 1985

Authors

Michael Beine
dSPACE GmbH – Product Manager TargetLink
Technologiepark 25
33100 Paderborn
Germany
mailto:mbeine@dspace.de

Michael Jungmann
MTU Aero Engines GmbH
Engine Systems Design/Software Development
Dachauer Straße 665
80995 Munich
Germany
mailto:Michael.Jungmann@muc.mtu.de