

Automated Checking of MISRA TargetLink and AUTOSAR Guidelines

Ingo Stürmer

Model Engineering Solutions GmbH

Stamat Stamatov

dSPACE, Inc.

Ulrich Eisemann

dSPACE GmbH

Copyright © 2009 SAE International

ABSTRACT

Model-based development of embedded automotive control software is characterized by the use of executable models throughout the entire development process. Modeling and simulation tools that are frequently used in this context include, for example, Simulink and Stateflow from The MathWorks. Code generators such as TargetLink make it possible to automatically generate efficient C code directly from these models. The quality of the models used for code generation has a direct influence on the quality of the generated C code. This is why it is vital that specific quality criteria be fulfilled when developing safety-relevant systems, for example, conformity with modeling guidelines starting at the model level. MISRA modeling guidelines exist for use with the TargetLink code generator. Conformity with these guidelines ensures a safe language subset and significantly improves the safety of generated code, for example, by avoiding ambiguous semantics.

This paper describes the requirements that MISRA TargetLink Guidelines impose on modeling TargetLink models, and the corresponding automated guideline checking provided by a new generation of checking tools.

INTRODUCTION

Model-based development of embedded automotive software using modeling and simulation tools such as Simulink and Stateflow from The MathWorks has been the established approach in the industry for many years now. These modeling languages make it possible to specify the functionality to be implemented graphically, and use simulation to validate the models at an early stage. Accordingly, these models are also considered executable specifications. Code generators such as TargetLink [2] make it possible to automatically generate efficient C code directly from these models (see [4]). This allows for a significant productivity gain in the software implementation phase.

Model-based development has some crucial advantages compared to traditional, code-based development. One key advantage is that the functionality to be developed can be simulated and evaluated at an early stage, before implementation occurs or the code even exists. One disadvantage, on the other hand, is that it is not possible to directly use the tried and tested methods (best practices) of traditional programming, such as guidelines relating to documentation, structuring, design patterns, etc.

Best practices for programming C code for safety-relevant systems, such as the MISRA C Guidelines [6], have been available for several years. There are also tools available that check conformity with the MISRA C Guidelines at the C code level. Conformity with such guidelines in developing software for automotive control devices is in most cases prescribed by the automotive manufacturers themselves. As a result, the MISRA C Guidelines are now considered almost a binding standard (e.g. ISO C). Since November 2007, MISRA modeling guidelines for generating production code from TargetLink models¹ using the TargetLink code generator from dSPACE have also been available [7].

A tool providing automated checking of guidelines at the model level as well as automated correction of guideline violations has recently been introduced. The tool is called *Model Examiner* [13] from Model Engineering Solutions (MES). MES and dSPACE are cooperating in developing model checks for MISRA TargetLink and AUTOSAR checks for TargetLink.

Now we will give a brief overview of the remainder of this paper. Section 2 discusses the importance of modeling guidelines in software development for the automotive sector. In this context, general, company-specific, and tool-specific guidelines will be examined and illustrated by means of examples. The focus will be on an examination of the MISRA TargetLink Guidelines for production code generation of safety-critical software in the automotive sector. The availability of the guidelines spurred the need for a checking tool that can be used for automatic recognition and correction of guideline violations. Section 3 describes the tool environment provided by the *Model Examiner* and informs the reader of which user scenarios in model-based development result for everyday practice. Section 4 summarizes the findings of this paper and provides an outlook for other relevant analysis and repair possibilities for models in the context of production code generation of automotive software.

2 GUIDELINES FOR MODELING OF EMBEDDED SOFTWARE IN THE AUTOMOBILE

The direct result of increasing software complexity in the vehicle is increasing model complexity. The modeling tools that are typically used in this field offer only limited mechanisms for helping the developer deal with this ever greater complexity. Furthermore, the use of alternative, semantically equivalent solutions for modeling the same functionality aggravates functional, resource, and maintenance-related issues. As a result, it has become generally accepted that conformity to modeling guidelines can greatly improve the quality of models in terms of readability, serviceability, testability, reutilization, exchange between OEMs and suppliers, etc.

¹ TargetLink models are based on Simulink models that contain an additional TargetLink blockset.

2.1 MAAB GUIDELINES FOR MODELING WITH SIMULINK AND STATEFLOW IN THE AUTOMOTIVE SECTOR - Guidelines that help meet the aforementioned quality criteria have been available for Simulink and Stateflow models for quite some time. These include the MathWorks Automotive Advisory Board (MAAB) Guidelines [5], whose use in the field of automotive software modeling is generally accepted. Examples of MAAB rules include those relating to the alignment of blocks, the representation of signal flow, guidelines for generally accepted design patterns, as well as the application of signals, vectors, and buses.

Example of a MAAB guideline - Fig. 1 shows a model that conforms to the na_0005 MAAB guideline 'Port block name visibility in Simulink models'. This guideline stipulates that the signal line label, which emanates from an input/output block, must be (1) assigned, (2) visible, and (3) identical to the name of the port (as illustrated by the signal *EngRPM_LP*). Moreover, the name of the signal must be identical to the port of the subsystem into which the signal leads.

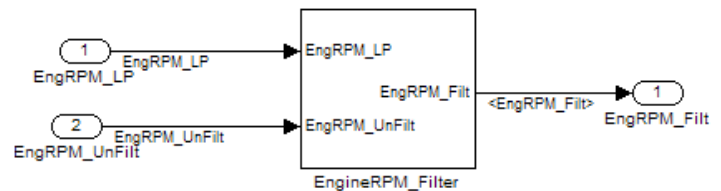


Fig. 1: MAAB guideline na_0005: 'Port block name visibility in Simulink models'

2.2 COMPANY-SPECIFIC MODELING GUIDELINES -

The MAAB Guidelines offer only limited support for code generation. They provide little information about which patterns and model or block configurations might have a negative effect on the code generated from the model. As a result, many automotive manufacturers have adapted or supplemented the MAAB guidelines with company and project-specific guidelines that reflect their own modeling and code generation requirements. These additions also address modeling constructs that refer to ambiguous behavior or problematic modeling constructs. This can best be illustrated by an example taken from the Daimler modeling guidelines [1].

Example of a company-specific modeling guideline -

Fig. 2 shows an example of a model illustrating a Daimler modeling guideline, which refers to the correct use of flowchart semantics [11]. Flowcharts are a language subset of Stateflow semantics and are usually used to describe control structures such as *if-then-else* or *switch-case* graphically with the help of directional, annotated edges (transitions with labels) and branching points (junctions). The evaluation order of transitions depends on what is known as the 'implicit execution order'. This states that the order of a transition is determined by its label (annotation with event, condition, and action part), and its graphical position (12 o'clock rule). In the top section of Fig. 2, first the right transition

is checked for whether *Flag1* is equal to zero. If this is not the case, *Flag1* is tested for *one* or *Flag 2* for *zero*. If this condition also does not evaluate to *true*, then the third path with the statement *action3* is executed.

The problem here is that the execution order automatically changes if the event *Event1* is introduced at the second transition, as this gives the label a greater weight than the label of the first transition (see bottom section of Fig. 2). In other words, the flowchart semantics or its execution order changes. As expected, this can easily lead to errors and unintended behavior. It therefore makes sense to establish a guideline at this point, which provides an explicit order in place of the implicit order. The former can be directly applied via an option in the Stateflow chart.

2.3 TOOL MANUFACTURER-SPECIFIC MODELING GUIDELINES - The company dSPACE issues its own modeling guidelines for Simulink and Stateflow for its TargetLink code generator [3]. These guidelines aid the correct use of the TargetLink code generator (block configurations, modeling patterns, etc.). They also refer

to Simulink and Stateflow patterns and block configurations, which can generally have a negative effect on the code, regardless of whether the models are used as a basis for manual implementation or code generation.

Example of a dSPACE TargetLink modeling guideline - The dSPACE TargetLink 'General Naming Conventions' guideline states that model names for code generation (i.e. port, subsystem, parameter names) may only consist of valid characters ([A-Za-z_0-9]) and must have a maximum length of 31 characters. The underlying reason for this is that longer identifiers are treated differently by different compilers. Furthermore, words that are reserved words in the target C language may not be used (e.g. *auto*, *extern*, *function*, *while*, etc). This guarantees that valid C identifiers can be generated by the code generator from the names given in the model.

This guideline was created for the TargetLink code generator, but could be applied to other code generators as well.

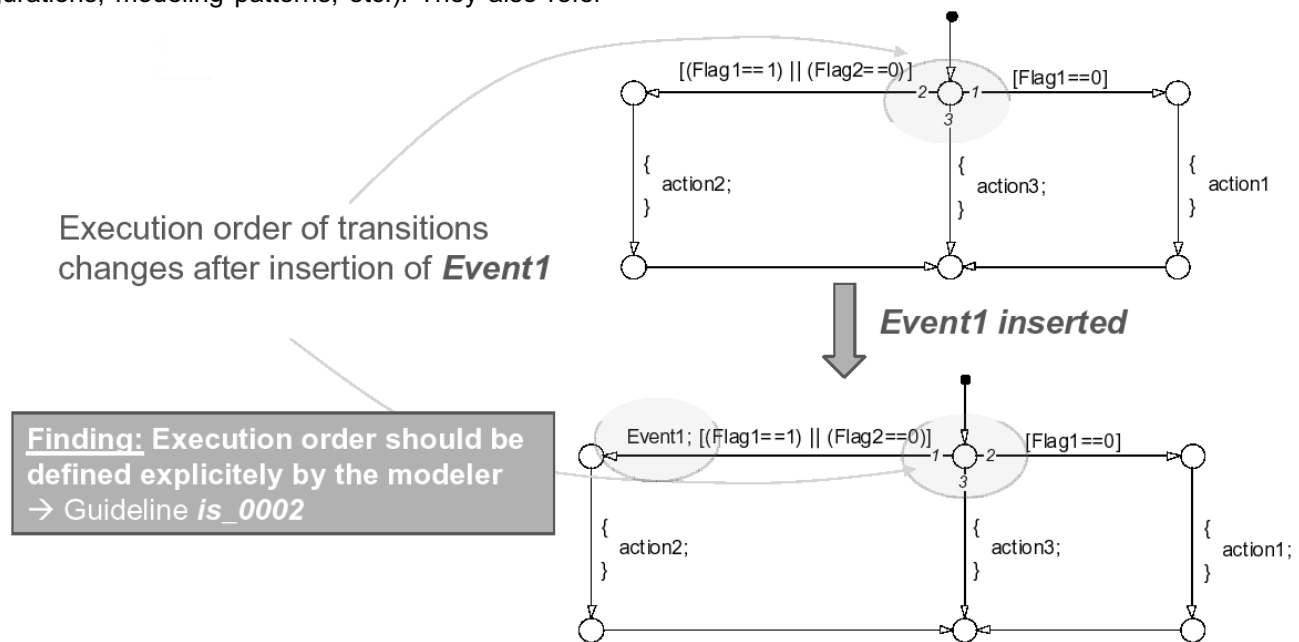


Fig.2: Example illustrating a Daimler modeling guideline [11]

2.4 THE MISRA TARGETLINK GUIDELINES - The Motor Industry Software Reliability Association (MISRA²) is a consortium of automotive manufacturers, automotive industry suppliers, and freelance engineers. The goal of the consortium is to disseminate best practices for the development of safety-relevant electronic systems in the vehicle and secure their correct application.

The MISRA TargetLink Guidelines [7] contain modeling specifications relating to functional safety, which are particularly relevant to safety-critical projects. The new

guidelines reflect the greatly increased importance of model-based design and automatic series code generation. The guidelines address the following issues in particular:

1. Avoidance of particular modeling elements and styles in MATLAB, Simulink, and Stateflow under the aspect of functional safety.
2. Modeling elements that support a transparent model structure and thus simplify the model review process.
3. Guidelines for specifying implementation information in the model in order to adhere to a maximum number of MISRA C:2004 guidelines in the generated code.

² MISRA and MISRA C are registered trademarks of MIRA Limited, held on behalf of the MISRA Consortium. No endorsement by MISRA is claimed or implied for any product.