FLORE
Repository istituzionale dell'Università degli Studi di Firenze

Formal modeling for railway signaling using commercial tools

Questa è la Versione finale referata (Post print/Accepted manuscript) della seguente pubblicazione:

Original Citation:

Formal modeling for railway signaling using commercial tools / A. Ferrari; A. Fantechi; S. Bacherini; N. Zingoni. - ELETTRONICO. - (2009), pp. 166-170. (Intervento presentato al convegno The First NASA Formal Methods Symposium tenutosi a NASA Ames Conference Center, Moffett Field, California, USA nel April 2009).

Availability:

This version is available at: 2158/386562 since:

Terms of use:

Open Access

La pubblicazione è resa disponibile sotto le norme e i termini della licenza di deposito, secondo quanto stabilito dalla Policy per l'accesso aperto dell'Università degli Studi di Firenze (https://www.sba.unifi.it/upload/policy-oa-2016-1.pdf)

Publisher copyright claim:

(Article begins on next page)

24 April 2024

Modeling Guidelines for Code Generation in the Railway Signaling Context

Alessio Ferrari
General Electric Transportation Systems
Florence, Italy
alessio.ferrari@ge.com

Alessandro Fantechi
University of Florence
D.S.I., Florence, Italy
fantechi@dsi.unifi.it

Stefano Bacherini
General Electric Transportation Systems
Florence, Italy
stefano.bacherini@ge.com

Niccoló Zingoni
General Electric Transportation Systems
Florence, Italy
niccolo.zingoni@ge.com

Abstract

Modeling guidelines constitute one of the fundamental cornerstones for Model Based Development. Their relevance is essential when dealing with code generation in the safety-critical domain. This article presents the experience of a railway signaling systems manufacturer on this issue.

1 Introduction

Introduction of Model-Based Development (MBD) and code generation in the industrial safety-critical sector created a crucial paradigm shift in the development process of dependable systems. While traditional software development focuses on the code, with MBD practices the focus shifts to model abstractions. The change has fundamental implications for safety-critical systems, which still need to guarantee a high degree of confidence also at code level. Usage of the Simulink/Stateflow platform for modeling, which is a *de facto* standard in control software development, does not ensure by itself production of high-quality dependable code. This issue has been addressed by companies through the definition of modeling rules imposing restrictions on the usage of design tools components, in order to enable production of qualified code.

The MAAB Control Algorithm Modeling Guidelines (MathWorks Automotive Advisory Board)[3] is a well established set of publicly available rules for modeling with Simulink/Stateflow. This set of recommendations has been developed by a group of OEMs and suppliers of the automotive sector with the objective of enforcing and easing the usage of the MathWorks tools within the automotive industry. The guidelines have been published in 2001 and afterwords revisited in 2007 in order to integrate some additional rules developed by the Japanese division of MAAB [5].

The scope of the current edition of the guidelines ranges from model maintainability and readability to code generation issues. The rules are conceived as a reference baseline and therefore they need to be tailored to comply with the characteristics of each industrial context. Customization of these recommendations has been performed for the automotive control systems domain in order to enforce code generation [7]. The MAAB guidelines have been found profitable also in the aerospace/avionics sector [1] and they have been adopted by the MathWorks Aerospace Leadership Council (MALC).

General Electric Transportation Systems (GETS) is a well known railway signaling systems manufacturer leading in Automatic Train Protection (ATP) systems technology. Inside an effort of adopting formal methods within its own development process, GETS decided to introduce system modeling by means of the MathWorks tools [2], and in 2008 chose to move to code generation. This article reports the experience performed by GETS in developing its own modeling standard through customizing the MAAB rules for the railway signaling domain and shows the result of this experience with a successful product development story.

2 Models for Railway Signaling Systems Rationale

MAAB guidelines have been developed to address the need of the automotive industries for a common language in MBD using the MathWorks tools. GETS decided to adopt these rules as a baseline for defining its own modeling standard according to the needs of the railway signaling systems domain.
Railway signaling systems software, and in particular the code of Automatic Train Protection (ATP) systems, is characterized by the extensive usage of control modes logic and message analysis algorithms. These kind of features can be easily modeled through Stateflow state machines, while Simulink blocks are more suitable for numerical algorithms [3]. This observation suggested the possibility of adopting Stateflow as the only tool for modeling the software, and use Simulink as a framework for allowing Stateflow Charts integration and to simulate the external environment.
Any railway signaling application software developed for Europe shall comply with the CENELEC standard [4]. The norm does not cover code generation issues, but relies just on source code quality, requiring structuring, readability, traceability and testability of the code. Two code generators have been evaluated: Real Time Workshop Embedded Coder (RTW) and Stateflow Coder. The first one allows code production for complete Simulink systems, while the second one generates code for only Stateflow Charts. Stateflow Coder has been chosen as the code generation tool since the code produced through RTW Embedded Coder was barely readable, not compliant with CENELEC quality requirements, and the development was focused on Stateflow modeling. With Stateflow Coder, integration among different generated software units was easily addressed by hand.
The MAAB guidelines are mainly composed by Naming Conventions, Model Architecture, Simulink and Stateflow recommendations. The same structure has been followed by GETS while defining its own guidelines. The majority of the work has been focused on Model Architecture and Stateflow rules. GETS already provided its own Naming Conventions for software and these conventions have been translated into modeling rules according to specific analysis of the generated code. Since only Stateflow blocks were used for the purpose of code generation, only MAAB Simulink rules dealing with model structure have been adopted and included in the Model Architecture rules set.
MAAB recommendations for Model Architecture and Stateflow have been analyzed and enhanced to produce the currently used GETS modeling standard.

3 Model Architecture Guidelines

Guidelines on Model Architecture concern the hierarchical development of Simulink/Stateflow models. The MAAB recommendations belonging to this set are quite general and do not give much support in developing the architecture of a system and moreover do not cover issues related to code generation. More detailed architectural guidelines have been developed to fit the railway signaling domain and address the code generation problem. The most important ones are reported below.

Rule GE_A_1: View Partitioning *The system shall be partitioned into three hierarchical view: Interface View, Architecture View and Design View.*
The Interface View describes the interaction with external systems. It is implemented through Simulink and it is linked to the Architecture View through a Model Reference block. The Architecture View describes the interaction between functional units. It is implemented in a Simulink model and it is constituted by interacting Stateflow Charts. The Design View represents the single functional units and it is implemented through Stateflow.

Rule GE_A_2: Interface View *The overall system shall define its Interface View in terms of just variables and buses. No direct connection with custom driver code is allowed at any level of the model.*

This rule enables the possibility of generating the code from the system and afterwords connecting the input variables with the sensor drivers (e.g., odometer, operator dashboard) and the output variables with the actuators drivers (e.g., braking system, operator display), facilitating the integration phase.

Rule GE_A_3: Design View *Each functional unit shall define input and output data and shall be implemented through a single Stateflow Chart.*
This rule helps concurrent development, since developers can implement Stateflow Charts separately, once the interface with the other components is given. At code generation level this enables easy unit testing: each Stateflow Chart is translated into a single file with a unique interface function having the same input and outputs of the Stateflow chart from which it has been generated.

4 Stateflow Guidelines

MAAB recommendations given for Stateflow modeling concern Stateflow constructs usage and patterns implementation. Each one of these rules is rated with a priority label (Recommended (R), Strongly Recommended (SR) and Mandatory (M)), issuing the level of importance of the recommendation. Even though the guidelines are rather detailed, they need some priority restriction and modification to be suitable for the railway signaling context. GETS, as many other companies, has its own coding standard addressing the quality requirements of the CENELEC norm. A strong effort has been devoted to identify Stateflow modeling rules ensuring compliance of the generated code with the existing software coding standard. Table 1 summarizes the rules that have been selected from the Stateflow recommendation set of MAAB and the priority restrictions or possible modifications issued.

Rule ID	Title	Priority	Priority Restriction
db_0129	Stateflow transition appearance	SR	Modified
db_0133	Use of patterns for Flowcharts	SR	M
db_0132	Transitions in Flowcharts	SR	M
jc_501	Format of entries in a State block	R	SR
jc_0521	Use of the return value from graphical functions	R	SR
na_0001	Bitwise Stateflow operators	SR	M
jc_0451	Use of unary minus on unsigned integers in Stateflow	R	M
na_0013	Comparison operation in Stateflow	R	M
db_0122	Stateflow and Simulink interface signals and parameters	SR	SR
db_0125	Scope of internal signals and local auxiliary variables	SR	M
jc_0491	Reuse of variables within a single Stateflow scope	R	M
jm_0011	Pointers in Stateflow	SR	M
db_0151	State machine patterns for transition actions	SR	Modified
db_0148	Flowchart patterns for conditions	SR	M
db_0149	Flowchart patterns for condition actions	SR	M
db_0134	Flowchart patterns for If constructs	SR	M
db_0159	Flowchart patterns for case constructs	SR	M
db_0135	Flowchart patterns for loop constructs	R	M

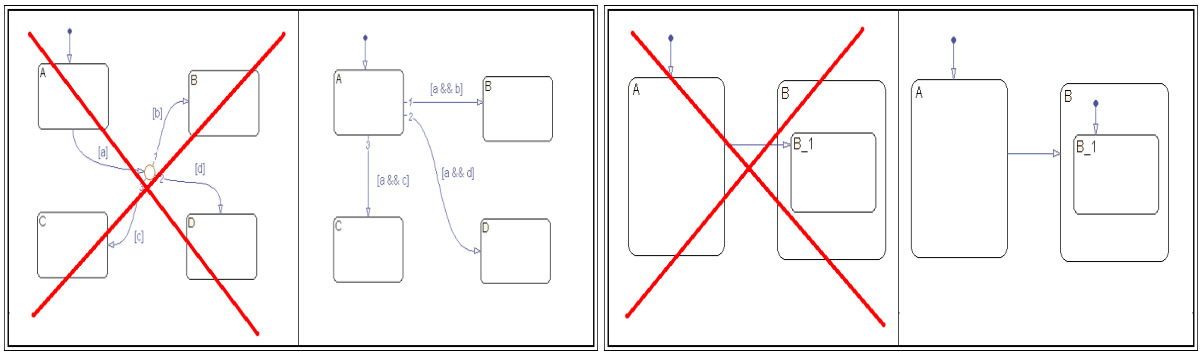
Table 1: MAAB Guidelines Restrictions

Priority restrictions are in general given for those rules having some impact on code generation. For example, rule db_0125 states that every local data in a Stateflow Chart shall be defined at Chart level

and not at Machine level. Since every local data that is defined at Machine level is generated as global data by Stateflow Coder, this rule has to be set as mandatory: global data is forbidden by the GETS software coding standard. Another example is the rule jc_0451 which regulates the usage of unary minus on unsigned integers. Unary minus shall be forbidden for unsigned integers to avoid possible overflow errors even if no warning is issued by the code generator.
Some rules needed to be replaced or modified when considered in the GETS context. For example rule db_0132 states that transitions entering states are allowed if they are directed to sub-states. This is not permitted by GETS modeling rules (see GE_S_04 in figure 1) since allowing a transition between states at different hierarchical level reduces the separation of concerns and moreover the generated code results in a *switch/case* statement without a *default* condition, violating one of the pre-existing GETS coding rules.
Other GETS specific guidelines have been added to address the need for GETS coding rules compliance. These additional recommendations range from limitation in the number of states for each Stateflow Chart to restriction in Function objects usage. Examples of additional rules are reported below.

Rule GE_S_01: States and Junctions *States and Junctions shall not be used jointly*
Rule GE_S_02: Path Connections *Avoid path connections in Function objects*
Rule GE_S_03: States Number *Each state machine shall be composed of maximum 10 states having the same hierarchical level*

Violation of the first two rules generates code with additional local variables with names depending on the code generator, decreasing the readability degree of the software. The third rule is used to limit the maximum number of *case* statements for each *switch/case* block.



(a) GE_S_01 (b) GE_S_04

Figure 1: Two examples of the GETS modeling rules

5 SCMT/SSC Baseline 3 Story

GETS experience with Stateflow modeling started in 2002 with the specification of the SSC ATP system [2], a platform developed by GETS for the Railway Italian Network (RFI) and currently deployed over 2000 kilometers of Italian secondary lines. The model of SSC was developed during the requirements definition phase, to support interaction with the costumer and without considering the issues related to code generation. At the end of 2007 GETS started refactoring the previously existing SSC model to enable code generation. The refactoring experience led to the definition of a preliminary version of the GETS modeling rules, bound to modify and enhance the MAAB guidelines.
In 2008 GETS was involved in a bid for the development of a new platform, called SCMT/SSC Baseline

3. The system was deemed to integrate into a single on-board platform the functionality of SSC and SCMT, the ATP system deployed over the main Italian lines. In order to speed-up development time and to deal with the complexity of the new platform, GETS decided to build a model of the SCMT system targeted for code generation, according to the previously defined rules. A third model, the SCMT/SSC Manager, was developed to let the two systems cooperate in an integrated environment. After code generation the software units of the three models have been integrated with the driver code of the system and software/hardware integration testing has been successfully performed. Along with the development of the new platform the GETS modeling rules have been updated and formalized to address new issues related to the large scale of the project.
The final on-board platform is composed by about 150K LOC of generated application level code and 40K LOC of hand-crafted driver and operating system code. This means that the majority of the software can be directly simulated and tested through Simulink. Development of the entire system, including testing activities, took about one year with a team of 15 people. Since SSC/SCMT Baseline 3 was a larger scale project compared to previous ones, it is not possible to quantify the productivity enhancement achieved. What can be stated is that the adoption of the modeling tools together with the MAAB rules tailoring allowed GETS to manage a complex platform in a structured yet flexible manner without increasing the number of developers. This would have been rather hard to address through the traditional GETS process based on hand-crafted code.

6 Conclusion

In this article we presented the successful experience of a railway signaling manufacturer in developing its own modeling guidelines for code generation, through proper tailoring of the MAAB recommendations. The experience demonstrated how automatic production of qualified code can be addressed through a rigorous set of rules, and showed how the peculiarities of the railway signaling context can be handled through suitable modeling choices. GETS is currently considering customizable tools that can provide automatic verification of these rules, such as MATE [6] and Simulink Model Advisor.

References

[1] National Aeronautics and Space Administration. *Report on the Utility of the MAAB Style Guide for V&V/IV&V of NASA Simulink/Stateflow Models*. N.A.S.A, May 1st, 2004. `http://sarpresults.ivv.nasa.gov/`.

[2] S. Bacherini, A. Fantechi, M. Tempestini, and N. Zingoni. *A Story about Formal Methods Adoption by a Railway Signaling Manufacturer*. 2006. FM 2006, Hamilton, Canada, Lecture Notes in Computer Science, 4025.

[3] Mathworks Automotive Advisory Board. *Control Algorithm Modeling Guidelines Using Matlab, Simulink and Stateflow, Version 2.0*. July 27th, 2007. `http://www.mathworks.com/industries/auto/maab.html`.

[4] European Committee for Electrotechnical Standardization. *CENELEC EN 50128, Railway Applications, Software for Railway Control and Protection Systems*. June, 1997.

[5] Japan MathWorks Automotive Advisory Board (J-MAAB). *Simulink StyleGuide*. March, 2003. `http://www.embeddedcmmi.at/fileadmin/docs/reports/J-MAAB_SimulinkStyleGuide_Eng.pdf`.

[6] D. Sturmer and I. Travkin. *Automated Transformation of MATLAB Simulink and Stateflow Models*. 2007. Proceedings of 4th Workshop on Object-oriented Modeling of Embedded Real-time Systems, pp 57-62.

[7] dSPACE GmbH TargetLink Product Management. *Modeling Guidelines for MATLAB/Simulink/Stateflow and TargetLink*. May 21st, 2008. `https://www.e-guidelines.de`.