Using SCADE to Develop Mission-critical High-quality Radar Application Software

Bernard Dion and Aubanel Monnier
Esterel Technologies, Parc Euclide – 8 rue Blaise Pascal, F-78990, Elancourt, France
bernard.dion@esterel-technologies.com
aubanel.monnier@esterel-technologies.com

Abstract:
The development of embedded software for radar applications may require software design processes that comply with the DO-178B standard or related processes. This raises issues about cost and productivity associated with the development of such mission-critical applications. In this paper, we introduce the value of the formal design, verification, and implementation technologies offered by model-based development with SCADE® to develop radar applications and comply with DO-178B. In essence, we look at the problems developers face when confronted with compliance to demanding quality standards and how the SCADE Suite® technology helps them increase their productivity while maintaining the highest level of availability and reliability in their applications. We also discuss the new version standard in development, DO-178C.

Key Words: radar software, model-based design, certification, DO-178B, DO-178C

I   INTRODUCTION

Safety and mission-critical software development requires rigorous processes, methods, and tools in order to ensure high quality and reliability of the developed product. In the aeronautics industry, authorities, such as FAA or EASA, certify the produced software according to the DO-178B standard [3]. Certification is the process verifying that the entire development life cycle conforms to the requirements of the standard.

In DO-178B, the software development process is decomposed in phases with precise objectives. Then, verification that the output of each phase conforms to the requirements the phase receives as inputs is required. The overall objective of these verification activities is to make sure that a given phase does not introduce errors during its realization.

The challenge is to master the cost of these verification activities, which in a certification context represent the highest part of the overall cost, while ensuring a high reliability and quality of the resulting software.

In this paper, we show how the SCADE Suite approach promoted by Esterel Technologies [1] provides solutions for this challenge when applied to mission-critical embedded software for radar applications. The approach is based on model-based development supported by a formal software development language, which has the desired characteristics for safe modeling. The core ideas of the approach are:
- Minimizing the risk of error introduction while modeling;
- Increasing the detection of errors and ambiguities in the early requirement specification coming from the system level activities; and
- Enabling the automatic code generation with a certified code generator producing high-quality and reliable software.

The paper is organized as follows:
- Section II presents the model-based design method of SCADE Suite, showing its benefits regarding the challenge.
- Section III presents the model-based verification activities enabled in the SCADE Suite framework.
- Section IV presents the certified automatic code generation showing the properties of the resulting code and its advantages.

II   MODEL-BASED DESIGN WITH SCADE

System requirements for safety and mission-critical embedded software are made of low-level parts in charge of handling the I/O data coming from or going to the hardware environment parts in charge of administrating the I/O data to transform them from their hardware representation, such as bus frames, into software data-structures and parts corresponding to the software core functions exchanging information with the hardware environment through the low-level parts.

In this context, the system requirements for the software controller functions express both the
mathematical framework that has to be implemented and the logical aspects (modes, control logic in the algorithms). For complex systems handling a high number of input data and logical conditions, capturing the requirements to build a software specification is very challenging and error-prone. Indeed, especially for the logical aspects, completeness is hard to establish, ambiguities are often present, and as it is easy to miss corner cases.

The SCADE model-based development approach promotes the use of a rigorous language to capture both the mathematical and logical system requirements using a user-friendly graphical notation: bloc diagrams for the modeling of the mathematical aspects as data-flow equations and state machines for modeling of the logical aspects. The power of SCADE allows for both formalisms within the same modeling environment and the ability to enable nesting of data-flow bloc diagrams inside states, as well as the nesting of state machines inside bloc diagrams, with no limitation [2]. This increases readability and compactness of specifications in a very significant manner.

SCADE is used to model the reactive part of the software controller. This is the part that must be called in a cyclic way in a real-time environment; it calls the SCADE function based on a regular frequency or based on events. At this level, hand code may be needed to complete the portion covered by SCADE to develop the functions elaborating and administrating the data coming from and going to the low-level layers and hardware environment, see Figure 1.

![Figure 1: SCADE part in the embedded system](image)

Figure 2 shows an example of a SCADE model specifying a radar application.

![Figure 2: the Radar top level view in SCADE](image)

The Scade language relies on a formal semantics ensuring deterministic behavior. Moreover, modeling constructs are safe by forcing the user to correctly handle initialization of data under all conditions, by ensuring bounded iterative schemes preventing from infinite loops, and by forcing the explicit data typing of all variables and ensuring type consistency of the model. Model errors such as absence of initialization, type mismatch, incomplete connections, and unused variables are automatically checked and detected at model level.

The benefit of the Scade language is clearly to capture the requirements in a straightforward way by expressing explicitly the interplay between logical and computation aspects (see Figures 3 and 4). The Scade modeling activity tends to reveal systematically the presence of ambiguities in the requirements. Typically, at model level, a case where a data is used while it was not specified how that data is initialized is detected.
As this is a strong need for all radar applications, the Scade language provides powerful constructs to handle arrays and matrix computation using a vector data type. Constructs are provided to uniformly apply a function over all of the elements of an array in one operation and in one execution cycle. Figure 5 shows the example of the pair-wise sum of two vectors of integers performed using the SCADE Map operator: if the two input vectors are $A = [a_1, \ldots, a_n]$ and $B = [b_1, \ldots, b_n]$, then this model computes $C = [a_1 + b_1, \ldots, a_n + b_n]$.

As this is a strong need for all radar applications, the Scade language provides powerful constructs to handle arrays and matrix computation using a vector data type. Constructs are provided to uniformly apply a function over all of the elements of an array in one operation and in one execution cycle. Figure 5 shows the example of the pair-wise sum of two vectors of integers performed using the SCADE Map operator: if the two input vectors are $A = [a_1, \ldots, a_n]$ and $B = [b_1, \ldots, b_n]$, then this model computes $C = [a_1 + b_1, \ldots, a_n + b_n]$.

The key property of SCADE is that a SCADE model is executable. Therefore, in addition to model review, functional verification can be operated through testing using the SCADE Suite Simulator. In addition, SCADE provides features to measure automatically the structural coverage of a SCADE model with respect to a suite of requirement-based tests. Coverage analysis may reveal uncovered modeling parts, which may be due to test incompleteness, dead model parts, etc. The module in charge of performing the functional verification and coverage measurement is SCADE Suite Model Test Coverage.

In SCADE Suite, features are available for conducting user-friendly simulation sessions by building a graphical environment in which inputs can be efficiently elaborated and outputs...
comfortably visualized. This is enabled by the coupling with SCADE Display®, which is a flexible graphics design and code generation tool suite for the development of safety-critical embedded display systems. With native support of the OpenGL® SC standard, SCADE Display is the new generation display framework, spanning prototyping, display design, simulation, verification and validation, certified code generation, and smooth integration with other applications. Figure 6 shows a typical simulation session of a radar SCADE model where the input and outputs are controlled from and displayed to a SCADE Display graphical model.

The benefit of enabling a complete verification of the software requirement specification is that it allows for discovering errors and shortcomings at the early stage of the software development life cycle. Discovering these errors late in the process would be much more costly.

IV CERTIFICATION WITH SCADE

In a traditional software development flow, the software requirement specification phase is followed by the coding phase in order to implement the software. Then, the verification of the coding phase requires ensuring that the source code complies with the software requirements specification through review and traceability, that the code is functionally correct, and that the tests cover structurally the code in a sufficient manner.

In SCADE, the coding phase is fully automated by the certified C code generator SCADE Suite KCG (KCG). The generated C code has the desired properties for safety critical applications. The code is traceable with respect to the SCADE model and is human readable. The code is ANSI-C compliant, platform independent, and portable. A sample of code generated from a small operator is shown in Figure 7, where the code generated for the state machine portion is very close to a human approach for state machines coding, and the code corresponding to the sum over vectors is exactly the way a C coder would have written it.

The behavior is guaranteed to be deterministic; the code has no dynamic memory allocation, no pointer arithmetic, statically bounded loops, safe typing, and safe array access preventing from outbound access. Regarding performance, SCADE users have reported that the code conforms to the constraints of critical software for both size and speed.

Being certified, the resulting code from KCG can be trusted, and verification activities at the code level simply eliminated. Traditionally these activities represent a significant part of the overall cost.
CONCLUSION

The model-based approach of SCADE for the development of mission and safety-critical radar application software comes with complete methodology compliant with the most stringent requirements of safety standards including DO-178B. The aim of this approach is to enable the development of high quality and reliable applications while reducing the cost of their development and verification compared to traditional flows based on manual coding. The key idea is to switch from a process whose main phases are Design/Code/Verify to a process whose main phases are Design/Verify/Generate. By doing so, the main verification activities are conducted earlier and rely on trusted automatic code generation. The characteristics and benefits of each phase are as follows:

- **Design**: the Scade language includes a graphical notation to capture both the mathematical computation aspects and the control logic. The notation allows for mixing of bloc diagrams for data-flow computations and state machines for the state-based logic, and nesting them with no limitation. By being unambiguous and formal, the notation can reveal errors in the system requirements.

- **Verify**: a SCADE model is executable, enabling verification of the compliance of the model with respect to the input requirements, in addition to model review. Moreover, the SCADE Suite Model Test Coverage module allows for verification of the structural coverage of the SCADE model in order to ensure the model is doing no more and no less than capturing the input requirements.

- **Generate**: the SCADE Suite KCG code generator produces safe, target independent, readable, and traceable ANSI-C code. KCG is certified with respect to DO-178B (Level A) and as such, the verification activities on the C code can be removed. The code has the desired properties for safety-critical applications (no dynamic memory, no pointer arithmetic, statically bounded loops, variables initialized before used, etc.). The code performances are thus highly predictable.

The SCADE users feedback have reported a 30 to 50 percent savings in the overall cost of the software lifecycle compared to flows based on manual coding.

This approach has been further refined within the new version of the standard (DO-178C), and its technology supplements including “Model-based Development and Verification” and “Formal Method”. The SCADE environment is now ready for the new standard.

REFERENCES

2. Colaço, Jean-Louis; Pagano, Bruno; Pouzet, Marc. A Conservative Extension of Synchronous Dataflow with State Machines. s.l.: EMSOFT, 2005
3. DO-178B Software Considerations in Airborne Systems and Equipment Certification, RCTA and EUROCAE, 1992
4. DO-178B Methodological Handbook, Esterel Technologies Web Site

BIO DATA OF AUTHORS

Bernard Dion is co-founder and Chief Technical Officer of Esterel Technologies since 1999. He is the Secretary of the tools qualification group of the SC205/WG71 committee (DO-178C). In 1978, he received a Ph.D. in Computer Sciences from the University of Wisconsin at Madison (USA).

Aubanel Monnier is a Senior Consultant at Esterel Technologies since 2002. He is a member of the ARINC 661 standardization committee for Cockpit Display Systems. In 2001, he received an Engineering degree from ESI in Nice-Sophia Antipolis (France).