Adding Formal Requirements Modeling to SysML

Mark R. Blackburn
www.markblackburn.com

Abstract. This paper seeks to raise awareness on the SCR extensions derived from industry use, and discusses how an integration of Software Cost Reduction (SCR) modeling concepts address current limitations in the System Modeling Language (SysML) standard. The paper describes an integration of a formal requirement modeling approach based the SCR method with the SysML structure mechanisms providing an alternative means for formalizing requirement behavior using a table-based approach that is supported by tools that extend the original SCR method.

Keywords: Requirement modeling, SysML, system engineering

1 Introduction

Model Driven Engineering (MDE) is fundamentally based on relevant abstractions and complementary tool automation. The emergence of domain specific modeling languages (DSML) demonstrates the need for modeling based on relevant abstractions that are simplified to suit the users. The System Modeling Language (SysML) is a DSML that extends the Unified Modeling Language (UML) for system engineers. The SysML requirement blocks rely on text-based statements that lack formal behavioral semantics, which limits tool analysis and simulation that can help in identifying defects or anatomies in requirements.

The Software Cost Reduction (SCR) method is a requirement modeling method with various types of tool support. SCR has been applied to large-scale industrial applications dating back nearly 30 years [1], [2] while demonstrating measurable benefits for helping to reduce requirement defects resulting in reduced program cost [3]. SCR modeling is fundamentally based on the use of tables that represent condition and event functions, and mode classes. The behavior formalism provides precise syntax and semantics that leverage tool-based analysis and simulation supporting defect identification in large and complex systems. Driven by industrial need, extensive tool support has been developed to support model transformation, simulation, analysis, property provers, and test vector generation. Users' needs have driven extensions to the original SCR such as support for structure and array data types, parameterized functions, and quantifiers. Recently a DSML and tool have been created that leverages some new SCR-based tool capabilities while demonstrating the general applicability of the fundamental modeling concepts underlying SCR. The DSML leverages the model analysis, which not only provides a means for verifying the model, but supports automated verification of the resulting implementation, and

provides an efficient means for verifying the evolution of the domain specific modeling language, modeling tool and tool transformations through SCR-based tools.

An objective of this paper is to raise awareness of the SCR extensions derived from industry use, and discuss how the integration of SCR concepts address current limitations in the SysML standard especially as it applies to requirement modeling. This paper discusses the model integration of SysML with an extension based on the SCR method for requirement modeling. While working with many companies on programs applications in domains such as avionic, aerospace, defense, automotive, medical, it is clear that system engineers develop requirements from a functional point of view. SCR inherently supports this functional point of view, and can semantically integrate formalized behavior with the structural interface mechanisms of SysML.

Section 2 discusses usage of SCR on large scale industrial projects and discusses the practical importance of providing system engineers with a functional-based approach to requirement modeling. Section 3 discusses the conceptual integration of SysML with SCR and tool support. Section 4 briefly discusses alternative methods that formalize requirements, and summarizes why SCR may be the best approach for system engineers while addressing a limitation in SysML.

2 Industrial Examples of SCR

The SCR method has been used on many types of system in various domains [4]. The Systems and Software Consortium (SSCI) has worked with members using many modeling approaches and tools including the SCR method and tools. This section discusses a few perspectives gained from working with members adopting SCR or in doing assessments of program engineering and modeling practices.

A typical process for system engineers begins with customer requirements and proceeds through functional analysis, where design tradeoffs are made, resulting in derived requirements that are allocated to subsystems. The use of models at the system levels for industrial systems that include hardware, electrical, hydraulic, and other mechanical controls with significant software content is too often semi-formal combining some models with text or diagrams or spreadsheets. Semi-formal models are inadequate for understanding dependencies across subsystems as well as impact analysis resulting from changes at the systems level. Many systems today are often part of a product family (e.g., aircraft variants, ground vehicles) and there are many opportunities for efficiency that can be gained by models that are semantically related across the various subsystems in different variants.

Example 1: Lockheed Martin documented their use of SCR on the C130J program back in 1994 [2]. They have recently upgraded the tools and helped drive extensions to the SCR method such as parameterized function tables, strings, arrays and quantification. They are applying the method to a product family derived from the original C130J and have converted thousands of tables from a legacy tool to newer model-based tools. They have established a process that uses a baseline of several thousand SCR tables (requirements) for different avionics functions of the C130J aircraft. They are able to include elements from the baseline and either override or extend with new custom-specific requirements models in SCR. They have provided

automatic document generation from the requirements that provide functional views of the customer requirements that are then allocated to the various subsystems.

Example 2: the Lockheed Martin C-5M program recently documented the result of applying the SCR method [3]. The program developed several thousand requirements modeled as SCR tables spanning numerous versions and releases. Requirement modeling helped to develop better requirements and interface information to support the design and implementation process. The systems engineers used requirement simulations of the models to validate the correctness and consistency of the requirements. The requirement modeling and simulation processes uncovered a large number of requirement defects prior to software implementation. Measurement data substantiates the claimed process improvements and program benefits. Measurement data supports the conclusion that the C-5M program process detected defects earlier, had about ½ the total number of defects, and on average corrected the defects twice as fast as on another related program, the C-5 AMP.

Example 3: System engineers and their customers view the system from the perspectives of functions that satisfy customer needed capabilities. During a recent program assessment, the projects used UML-based modeling at the system level, but stated that the functional view is lost in what ends up being a more object-oriented view of the system. The object-oriented view tends to result in a more bottom-up view focused more on design. In contrast SCR provides a functional view, which might be more applicable to system engineers working top down from functional analysis of customer requirements.

Example 4: as discussed in Section 1, another SSCI member has been working on a research project to demonstrate the development of a DSML and tool chain. The DSML model is transformed into a SCR model and tools support model analysis, consistency checking (e.g., disjointness, race conditions), and test vector and test driver generation for verifying the implementation associated with the models. The key point is that the underlying SCR modeling concepts are sufficiently general to support the DSML.

Examples 1 and 2 illustrate two successful uses of SCR. However, in both cases, the SCR lacks an explicit context of how the functions are allocated to the system blocks. This gap provides the opportunity for leveraging the SysML structure modeling capabilities when integrating with SCR behavior modeling.

3 Model Integration of SCR and SysML

SCR models represent required functionality of a system or component using tables to relate monitored variables (inputs) to controlled variables (outputs), as reflected in Fig. 1. There are three basic types of tables: 1) Condition Table, 2) Event Table, and 3) Mode Transition Table. A mode transition table is a type of state machine, where related system states are called system modes, and the transitions of the state machine are characterized by events. An event occurs when any system entity changes value. A condition is a predicate characterizing a system state. A term is any function defined in terms of input variables, modes, or other terms. The SCR tables can be combined to specify complex relationships between monitored and controlled variables using mode or terms variables. This allows common conditions, events, and modes to be defined once, and referenced multiple times. For mapping SCR into

SysML, there are at least two ways: 1) the SCR modeling constructs can be used to represent requirements at the use case level; the actors may relate to monitored or controlled variables, and the use case function is modeled as SCR tables; 2) SCR could be used to model the requirements or derived requirements of any block, where the interfaces of the block are mapped to SCR inputs and outputs. As reflected in Fig. 1. a new SCR feature includes parameterized function tables. A function can be referenced by one more condition, event, model or function tables.

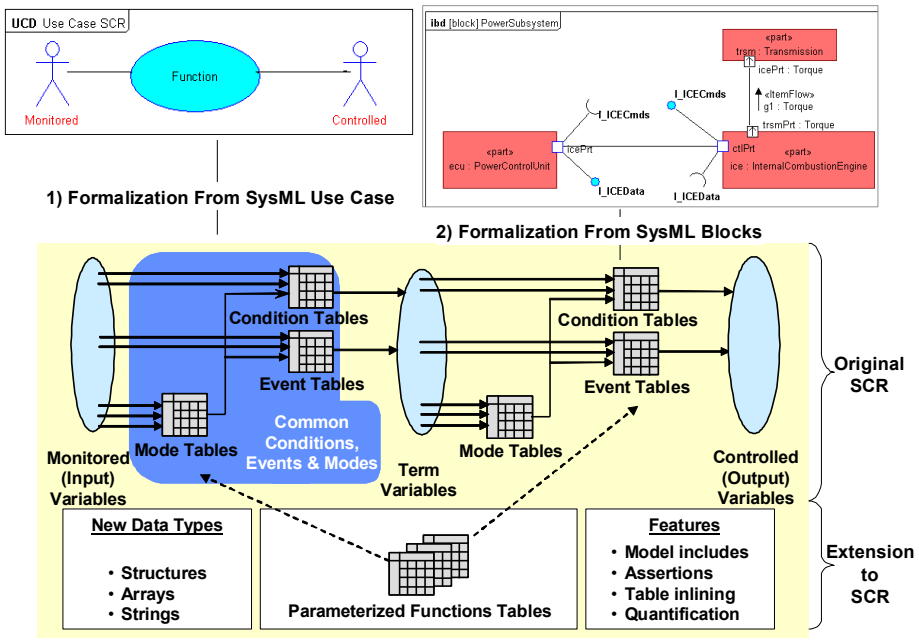

Fig. 1. One or more SCR condition, event, or mode tables are used to relate monitored (input) variables to controlled (output) variables. Parameterized function tables are a new extension that can be referenced by one or more tables.

Heitmeyer provides a historical perspective on SCR, and tool summary [5]. Tool-based automation for SCR includes functions for model creation, consistency checking, model checking, simulation, invariant generation, test vector generation, as well as integration with requirement management tools and theorem provers that can be used to evaluate security and safety properties. New SCR extensions include data types for strings, structures, and arrays, parameterized function tables, modeling constructs for quantification, assertions, inlining and model libraries. These extensions provide greater requirement formality with tool automation to help system engineers in early validation of the requirements with automatically generated requirement-based test artifacts to support verification, and requirement-to-test traceability.

4 Conclusions

This paper discusses how the integration of SCR concepts addresses current limitations in the SysML standard especially as it applies to formal requirement modeling. It also discusses some SCR extensions developed to meet the needs of industry users. There are other formal requirements modeling methods and associated tools such as such as RSML and its variants SpecTRM [7] or the KAOS [8] goal-driven methodology that could provide greater rigor for SysML. The simplicity and generality of the SCR modeling method and extensive tool support may make it easily applicable to SysML. As discussed in Section 3, the integration is relatively straightforward, and provides formal requirement modeling that might be easy to use by system engineers that do not have extensive programming background. This could address a problem stated by a representative of one of the leading commercial modeling tool suppliers. Their tool, like many of the competitors provides extensive support for UML and SysML, however, the process steps required for constructing a model of the system that can be used for analysis and simulation is complex. To enable the models to support simulation, users must be able to complete aspects of sequence or state machines using some form of programming language. The effort involved to gain proficiency is often a barrier to entry for system engineers. Finally, Lutz reported that the primary cause of safety-related faults was errors in functional and interface requirements [6]. The cases cited in Section 2 have shown that modeling helps identify defects, but if the modeling tools and methods are too complex to use, the lack of use of tools on complex systems increases the likelihood that defects will go undetected.

References

1. Heninger, K., Parnas, D.L., Shore, J.E., Kallander, J.W.: Software requirements for the A-7E aircraft. Technical Report 3876, Naval Research Lab., Wash., DC, (1978)
2. Faulk, S., Finneran, L., Kirby, J., Shah, S., Sutton, J.: Experience Applying the CoRE Method to the Lockheed C130J, Proceedings of the Ninth Annual Conference on Computer Assurance, IEEE 94CH34157, Gaithersburg, Maryland, (1994)
3. Allen, S.D. Hall. M.B., Mansfield, M.D., Kelly, V., Blackburn, M.R.: Requirement Modeling for the C-5 Modernization. CrossTalk, February, (2009)
4. Blackburn, M.R., Busser, R.D., Nauman, A.M., Morgan, T.R.: Life cycle integration use of model-based testing tools. Digital Avionics Systems Conference, (2005)
5. Heitmeyer, C., Jeffords, R.D.: Applying a Formal Requirements Method to Three NASA Systems: Lessons Learned, IEEE Aerospace Conference, (2007)
6. Lutz, R.R.: Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems", IEEE Proceedings RE'93, San Diego (1993)
7. Zimmerman, M., Rodriguez, M., Ingram, B., Katahira, M., de Villepin, M., Leveson, N.: Making formal methods practical. Digital Avionics System Conference, (2000)
8. van Lamsweerde, A.: Building Formal Requirements Models for Reliable Software. Invited Paper Ada-Europe 2001, Leuven, LNCS, Vol. 2043, Springer-Verlag (2001)