Combining SysML and formals methods for safety requirements verification

Dominique Evrot, Jean-François Pétin, Gérard Morel

Combining SysML and Formals Methods for Safety Requirements Verification

Dominique Evrot, dominique.evrot@incose.org; Jean-François Pétin, jeanfrancois.petin@incose.org; and Gérard Morel, gerard.morel@incose.org

This work, developed at Nancy Research Centre of Automatic Control (CRAN from Nancy University) jointly with INRS (Institut National de Recherche et de Sécurité) deals with the verification of safety requirements for the design of complex control systems involving software, mechanical, or electrical components for industrial safety-critical applications such as power plant control or embedded control systems.

The dynamic of such control systems results from an interaction network between all the system components that can introduce undesired behaviour with critical impact on safety. Consequently, safety properties of the control system cannot be verified by only proving local properties of each component, but need to be studied through the emerging behaviour that issues from the interaction network. The main difficulties of this last verification are caused by the heterogeneity of the control components different technology domains that use proper design formalisms and tools.

During the last years, tools allowing multiple-skill approaches and a high abstraction level have been developed. SysML is a UML2.0 profile that has met system engineering community expectations about requirements modelling, behavioural and architectural designs, and the abstract representation of systems. However, this toolbox does not provide verification supports, so there is still a need for verification tools that fir with a multiple-skill approach. On the other hand, dedicated tools for modelling and verifying real-time reactive system are available using the discrete event system theory and model-checking technology.

However, there is a growing interest in methods and tools that facilitate the validation of software-intensive automation systems. This interest becomes a legal requirement when dealing with safety-critical systems: the safety-related standard *IEC 61508* strongly recommends that formal verification methods be applied in the certification process by the suppliers, integrators, or independent external authorities, but does not define how they can be applied. The main idea is to develop an integrated methodology that combines both approaches: a system-oriented approach that identifies and formalizes safety requirements using SysML, and a skill-oriented approach for components design and verification using the formal methods of the discrete event system theory.

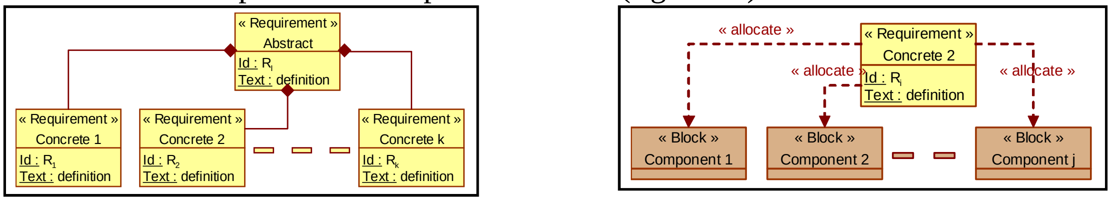
An engineer's central problem is to ensure that the system under control to be developed (denoted *S*), is compliant with the end-user requirements (denoted *R*). This assertion can be noted as $S \vDash R$. We must first remember that(1) requirements *R* are usually captured in multiple levels of abstraction and broken down into sub-requirements at different levels of abstraction, denoted $R = \{R_1, R_2, \dots R_n\}$; and that(2) the system under control is often composed of heterogeneous subsystems (such as mechanical, software, or electronic) that cooperate to achieve the systems goal, denoted $S = \{S_1, S_2, \dots, S_m\}$. These two facts complicatethe initial problem by introducing a non-bijective relationship between the set of requirements and the set of system components (or subsystems). In other words, it means that a given component S_i may satisfy a subset of requirements, such that $S_i \vDash \{E_k\}_{k \in [1,n]}$; and it means that a given requirement R_j may be satisfied by a subset of system components, such that $\{S_k\}_{k \in [1,m]} \vDash R_i$. It is our objective to identify and prove this relationship is preserved, especially for the safety requirements, through the whole systems engineering process.

Two ways of thinking about this topic can be found in the scientific literature. One way is based on the properties assertion method that uses an *a posteriori* verification process that aims to prove that a given system specification, design, or implementation fulfills a given model of safety requirements. An important technique in this type of work is model checking (Clarke, Grumberg, and Peled 2000). Limits are linked to the size of the models due to explosion phenomena and to the identification and formalisation of the properties to be proved. The other way of thinking is based on a refinement method that uses an *a priori* verification process that aims to progressively enrich an initial set of abstract requirements to produce a concrete model of the system that satisfies, by construction, the set of requirements (Pétin, Morel, and Panetto 2006). Formal refinement mechanisms such as those supported by the "B method" (Abrial 1996) belong to this kind of approach.

In this way, we have proposed an integrated method that aims to formally identify and refine the safety requirements of a complex control system by using the SysML requirements diagram, to allocate the requirements on a given set of system components using a meta-model of requirements traceability concepts, to define the impact of this allocation in terms of safety properties to be satisfied by each of the system components, and then to provide the existing model-checkers with this correct model of component properties.

Requirements Refinement, Allocation, and Projection with SysML

To design a component's architecture that fulfills the system's safety requirements, we need a set of more detailed specifications related to the safety control functions or the material and immaterial barriers to be used. We based our approach on refinement rationale in the same idea than the "B" formal refinement mechanism. This mechanism substitutes a set of more concrete requirements for an abstract requirement. In the SysML Requirements diagram, this operation is represented by implication or composition links, meaning that if the set of concrete requirements is fulfilled, then the abstract requirement is also fulfilled (figure 1a). A requirement can refine two upper requirements. Requirements are modelled with a class stereotype that includes an open list of attributes like the requirement text definition, source, identification, and so on. This set of requirements must then be allocated onto system functions or components. Functions are modelled using the Activity SysML diagram while system architecture and components are modeled using the Block and Block Definition SysML diagrams. These diagrams allow the engineer to trace the requirements that system functions and components are expected to fulfil (figure 1b).



(a) requirements refinement (b) requirements allocation

Figure 1. Requirements modelling using SysML

In order to provide proofs about requirements refinement and allocation, we need a more formal description of the different safety properties, associated to each abstract and concrete requirements, denoted respectively *Paj* and *Pci*, In this way, we have modified the SysML meta-model to introduce logic properties that formalize the textual description of requirements (see figure 2). This class stereotype allows the engineer to describe the properties in terms of logic predicates or numerical parameters. This extension enables an engineer (1) to make the link between system modelling and trade-oriented design with the objective to prove that behaviour of a given component is compliant with local expected properties; in this way, concrete properties *Pi* of each components can be reused as logic or temporal predicates by model checking tools, such as UPPAAL[1], or can be reused as post-conditions by simulation and testing tools; and (2) to formally demonstrate that the combination of all of the components' concrete properties *Pci* establishes the system's abstract properties *Paj*, using a theorem prover, such as COQ (Huet, Khan, and Paulin-Mohring 1995).

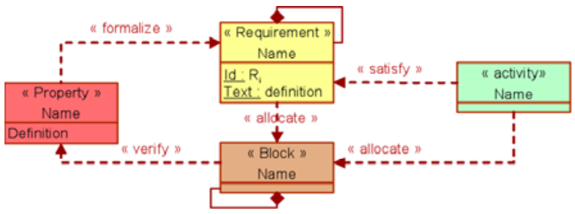


Figure 2: SysML Requirements traceability

This approach has been implemented (figure 2b) through the definition of an XML format that enables the exchange of data and models between different commercial tools, such as those designed for system specification with SysML (Magic Draw), for control system design and simulation using trade-oriented languages (ControlBuild), and for proving safety properties (UPPAAL model checker) using XSLT transformations.

On-going work

We have proposed a system modeling approach that combines non-formal methods, which are necessary to capture and formalize system requirements, and formal methods, which are required by safety-related standard such as *IEC61508*, to prove that the local behaviour of each system component contributes to satisfying system properties. Even if the end result is clearly related to software control, the approach does not focus on this point of view but tries to deal with all system technical components in a systems engineering context. This point is very helpful for the development of COTS-based (commercial off-the-shelf) control, where subcontractors' requirements must be clearly identified. Although these interdisciplinary exchanges between computer science and system engineering approaches demonstrate that they contribute to verify the highest Safety Integrity Levels (IEC 61508), common experiments on laboratory-scale and industrial-scale case-studies emphasize that effort must still be employed to make the proposed engineering framework effective in practice.

References

Abrial, J.R. 1996. *The B-book: Assigning programs to meanings*. Cambridge, U.K.: Cambridge University Press.

Clarke, E. M., O. Grumberg, and D. A. Peled. 2000. *Model checking*. Cambridge, MA: MIT Press.

Huet, G., G. Khan, and C. Paulin-Mohring. 1995. *The Coq proof assistant: A tutorial.* INRIA (French National Institute for Research in Computer Science and Control) Technical Report 178. Rocquencourt, France: INRIA.

Pétin, J.-F., G. Morel, and H. Panetto. 2006. Formal specification method for production systems automation. *European Journal of Control* 12 (2): 115-130

[1] Magic Draw is a product of No Magic, Inc.; ControlBuild is a product of Geensys; UPPAAL is developed by the Information Technology Department of Uppsala University (Sweden) and the Computer Science Department of Aalborg University (Denmark)