Combining SysML and formal methods for safety requirements verification

Jean-François Pétin, Dominique Evrot, Gérard Morel, Pascal Lamy

▶ **To cite this version:**

Jean-François Pétin, Dominique Evrot, Gérard Morel, Pascal Lamy. Combining SysML and formal methods for safety requirements verification. 22nd International Conference on Software & Systems Engineering and their Applications, Dec 2010, Paris, France. pp.CDROM. hal-00533311

HAL Id: hal-00533311
https://hal.science/hal-00533311

Submitted on 5 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Combining SysML and formal models for safety requirements verification

Jean-François Pétin[1], Dominique Evrot[1+2], Gérard Morel[1], Pascal Lamy[2]
(1) CRAN, UMR 7039 Nancy Université - CNRS, France, (2) INRS, France

Abstract. Industrial safety-related standards strongly recommend the use of formal methods to control the complexity of software-intensive automation. This paper deals with the verification of safety requirements for the design of an industrial machinery control system embedding safety-critical software applications. It combines SysML semi-formal modelling approach to capture and structure safety requirements and model-checking techniques for the formal verification purposes.

Key-words : System specification, SysML, formal verification, Discrete Event Systems, Safety requirements

1 INTRODUCTION

This paper deals with the verification of safety requirements for the design of complex systems involving software, mechanical, electrical, or pneumatic components for industrial safety-critical machinery. The control of such complex 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 and technology domains that use proper design formalisms and tools.
In the development of industrial automation, standards, such as the safety-related IEC 61508[1] or its application in the machinery domain (IEC 62061[2]), strongly recommend the use of formal methods to control the complexity of software-intensive applications. In this way, the role of INRS (Institut National de Recherche et de Sécurité), acting as a French independent institute, is to anticipate future risk by acquiring new knowledge and converting current knowledge into practical know-how available to professional in charge of prevention.
This paper presents a combination of SysML semi-formal modelling approach, to identify, specify and refine properties and architectures of an INRS case study, with formal models for the design of its control system and for the verification of its dynamic properties. Section 2 introduces the problem statement by showing complementarities between system-oriented approaches and formal models. This section ends by the description of INRS case study which is developed using our proposed methodology in section 3. The final section provides some conclusions and open issues.

2 PROBLEM STATEMENT

Automation engineering has to ensure that the system under control to be developed, noted S, is compliant with the end-user requirements, noted R. Taking into account the fact that:

- requirements R are usually captured in multiple levels of abstraction and broken down into sub-requirements at different levels of abstraction, noted $R = \{R_1, R_2, \dots R_n\}$,
- the system under control is often composed of heterogeneous components (mechanical, software, electronic sub-systems) that cooperate to achieve the systems goal, noted $S = \{S_1, S_2, \dots, S_m\}$,

the initial problem becomes a non bijective relationship between the set of requirements and the set of system components. Fusaoka [1] and Supervisory Control Theory [2] have formalised this problem focusing on Discrete Event System dynamics point of view. This formulation can be extended [3] by taking into account system features (dynamics but also functions, structure, information, etc) according to the predicate $S_i \wedge S_p \Rightarrow$ Goal (where S_c means Control system and S_p means physical system) that has to be preserved through the automation engineering processes. More generally speaking, it means that:

- a given system component S_i may satisfy a sub-set of requirements such that $S_i \Rightarrow \{R_k\}_{k\in[1,n]}$,
- a given requirement R_j may be satisfied by a subset of system components such that $\{S_k\}_{k\in[1,m]} \Rightarrow R_i$.

Identifying and proving the preservation of these relationships for the safety requirements, all along the system engineering process, is our main objective. Two ways of thinking can be found in the scientific literature to partially cover this issue: formal methods & models and system engineering approaches.

2.1 Formal models

Formal modelling process relies on mathematical foundations that may ensure that a given system design and/or implementation fulfils a given model of requirements. Conceptual and practical approaches have been widely explored by academic bodies related to computer sciences and automatic control: examples are software verification by theorem

[1] IEC 61508, Functional safety of electrical/electronic/ programmable electronic (E/E/PE) safety-related systems
[2] IEC 62061, Safety of machinery- Functional safety of electrical, electronic and programmable electronic control systems

proving [4], model checking [5], or automatic synthesis [6]. More precisely in the area of automation, these approaches are applied:

- during the design phase, to prove control model properties, using analysis techniques and model checking based on the building of the reachable states space of the control software,
- during the implementation phase, to prove PLC (Programmable Logic Controller) programs properties by applying formal techniques to check the properties satisfied by controllers implemented using IEC 61131-3 programming languages [7].

Due to the combinatory explosion phenomena induced by the exploration of the whole state space, these techniques are efficient for small-sized components. For larger scale applications, system global requirements must be transformed into multi-component local properties that can be processed by a model-checker. Bridging the gap from global system properties to a composition of local properties remains difficult [8].
Second difficulty is linked to the underlying mathematical representation which is limited to the modelling of system dynamics, and hardly covers the description of other system features. More precisely, the decomposition of a complex system in several sub-systems, involving software, hardware and man-machine interface components, does not facilitate the traceability and the refinement of global system properties through the sub-system decomposition, design, verification and validation. This aspect requires method that enables model refinement to capture very abstract requirements including safety properties and to project them into all system components [9]. Following the consensus that early phases of a system definition are the most important in ensuring that the target system will satisfy the user's requirements, many systems and automation engineering practitioners [3] [10] [11] consider that time is ripe to formalize the earlier phases of system specification. This challenge justifies the use of system-oriented approaches [12] in automation engineering.

2.2 System-oriented models

Standards define system engineering as an "interdisciplinary collaborative approach to derive, evolve and verify a life cycle balanced system solution which satisfies customer expectations and meets public acceptability" (IEEE 1220[3], ISO 15288[4]). System engineering is usually involving two complementary and inter-correlated processes related to system definition that aims to identify and analyse the system needs and requirements and to system development that focus on the system design in accordance with identified requirements.
This definition meets our goal that is to identify safety system requirements issued from an interdisciplinary approach and to design and verify a system control that contributes to satisfy the identified requirements.
Models and tools promoted by system engineering are most of the time based on a wide set of graphical formalism that help to capture the various system features as well for its definition as for its development (requirements engineering, functional, structural and behavioural modelling, etc). Some of them, based on the Unified Modelling Language (UML[5]) and its extensions, cover the various aspect of system modelling, while other ones focus on a specific part such as requirements expression (DOORS[6]) or documentation traceability (Reqtify[7]). UML and its extension UML2.0, is a system modelling toolbox that should address use cases identification and architectural, behavioural design of the system. These tools allow a multi-trades approach due to their high level of abstraction and its graphic notation. However, it seems that its origin and primary application being software-oriented, this language cannot be adequate for all aspects of systems engineering (hardware, software, man-machine interface…). The new profile SysML has been developed to meet system engineering community expectations. SysML toolbox contains new diagrams, like the requirement diagram, that allows a most efficient system analysis and design.
Using UML/SysML based models for automation engineering has been identified as an efficient way to handle the complexity of current control applications [13]. OOONEIDA [14], TORERO [15] and CORFU projects [16] as well as the definition of UML profile for process automation [17] can be mentioned in this way. However, the wide notation toolbox provided by UML/SysML suffers from a lack of formal semantics for validation and verification issues. Efforts have been made towards UML formalization but they are often focused on static equivalence between diagrams in a Model Driven Engineering perspective: executive semantics of the dynamics diagrams are formalized by providing equivalence to DES models such as Petri Nets.

Our objective is to bridge the gap between a necessary semi-formal approach, such as SysML and formals model that are efficient for validation and verification of safety properties. In this way, an integrated method has been proposed to identify the safety requirements of a complex control system using the SysML requirements diagram, to allocate them on a given set of system components, to define the impact of this allocation in terms of safety properties to be satisfied by each system component, and then to provide the model-checkers with this correct model of component properties.

[3] Standard for application and management of systems engineering
[4] Systems and software engineering, System life cycle processes
[5] OMG, Object Management Group, www.omg.org
[6] Telelogic, www.telelogic.com
[7] Geensys, www.geensys.com

2.3 Case study

We consider a mechanical press that aims at stamping metal products with a tool in vertical movement. A crankshaft equipped with two sensors for top and bottom dead centres gives this vertical movement. A clutch, which is actuated by a pneumatic valve, ensures the transmission between motor and crankshaft. Operator protection is supplied thanks to a two hands command. An emergency stop enable or not the stamping action. We focus on the press function that is in charge of clutching and braking the press motor in the normal operating mode (motor is considered as always working). In this mode, the press is initially stopped in up position and waits for a two hands command. Then, the tool is falling until the bottom dead centre is reached, then is rising back, and finally stopped when in up position. Moreover, if the operator releases the hand command during the falling phase, or if emergency stop occurs, the press is stopped.

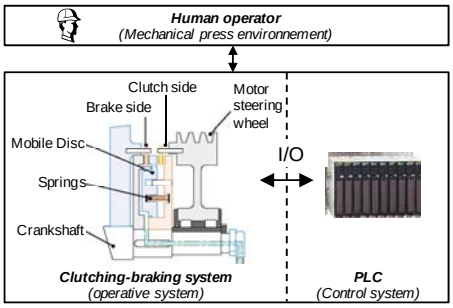


Fig. 1. Mechanical press case study

Usually, control of such safety control system is performed by wired technology. Due to economic constraints (control application reusability, functions complexity), control of industrial machinery ensures more and more safety functions using Safe Programmable Logic Controller (Safe-PLC). Integration of safety functions into the software parts of programmable systems can be envisaged only if the safety level of programmable technology set is at least equal to the wired technology. One role of INRS is to study methods for the development of safety machines according to recommendations listed in the annex IV of the machinery directive [18] and IEC 61508. In particular, the higher level SIL 4 recommends the use of formal methods to control the quality of software intensive applications.
In this way, INRS has studied the B method [4] and ordered a case study development using this method [19]. This study has clearly shown the benefit of using B for proved code generation. Nevertheless, modelling methodology is not explicit and relies on the expert practice. In the context of formal system engineering and taking into account the need for traceability of safety properties, this approach must be enrich by modelling guidelines:

- to better detail how the safety requirements are taken into account when modelling,
- to enlarge software engineering to automation and system engineering including software, hardware, man-machine interface and physical systems.

In other words, starting the study of a complex system using formal models appears to be hardly achievable in practice. It justifies the use of less formal model such as SysML [20] to cover the first phases of specification using a graphical language that promotes the knowledge sharing between the involved engineers.

3 SYSML-BASED METHODOLOGY TOWARDS REQUIREMENT VERIFICATION

Requirements are the formalization of the user needs for one part and the formalization of provider limits or standards prescriptions for other ones. Requirements can be expressed at a high level of abstraction when dealing with system global features but can also be expressed at a very low level of abstraction when depending on technical local choices for system component (for example material and immaterial barriers to be used). Structuring this set of requirements is based on an incremental modelling, in the same idea than the B formal refinement, which consists in substituting an abstract requirement by a set of more concrete ones. Using SysML *Requirements diagram*, requirements are modelled with a class stereotype including an open list of attributes such as text definition, source, id, and so on. Structuring the requirements is done using the composition link provided by SysML, which means that the composed requirement is realized if and only if all the components requirements are realized. Figure 2 illustrates the decomposition of one abstract requirement into several more concrete ones. Benefit of such a diagram is to facilitate the elicitation and structuring of requirements that are usually expressed by the end-users and/or the standards in natural language [21].
This set of requirements must then be allocated (or mapped) onto system functions and/or components. Functions are modelled using classical Activity and/or Use Case diagrams. System components are modelled using SysML *Block* defined as class stereotypes. Structure of the control system is modelled using Block Definition Diagram (BDD) while detailed definition of control components can be modelled using Internal Block Diagram (IBD). IBD is an internal description of a component, similar to synchronous DES models, and which can be exported to widely used tools for control design (Simulink[8], Scade[9], ControlBuild[10], etc).

[8] Simulink is a MathWorks, Inc. product

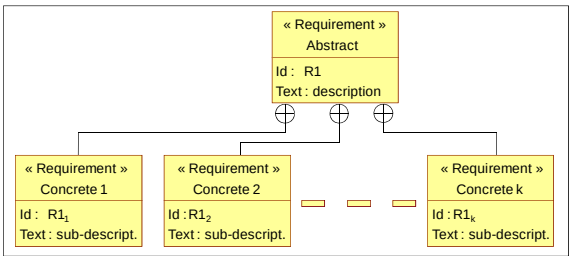


Fig 2. Requirements refinement

Allocation is made using the following SysML links: *satisfy* for the mapping between requirements and blocks (figure 3) and *refines* for the mapping between requirements and functions (figure 4). Block and activity relationship is modelled with *allocate* link: it means that an activity having a *refines* link with a requirement must be implemented by the block that *satisfy* this given requirement (figure 4). Note that requirement allocation is preserved by the block composition link. It means that if a requirement is *satisfied* by a component, then it is also *satisfied* by all its sub-components.

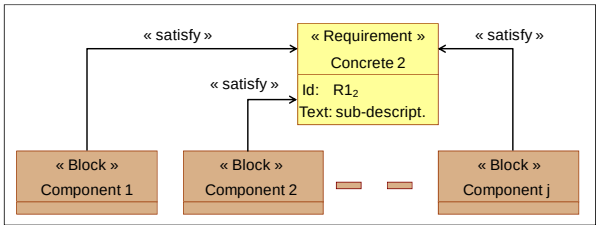
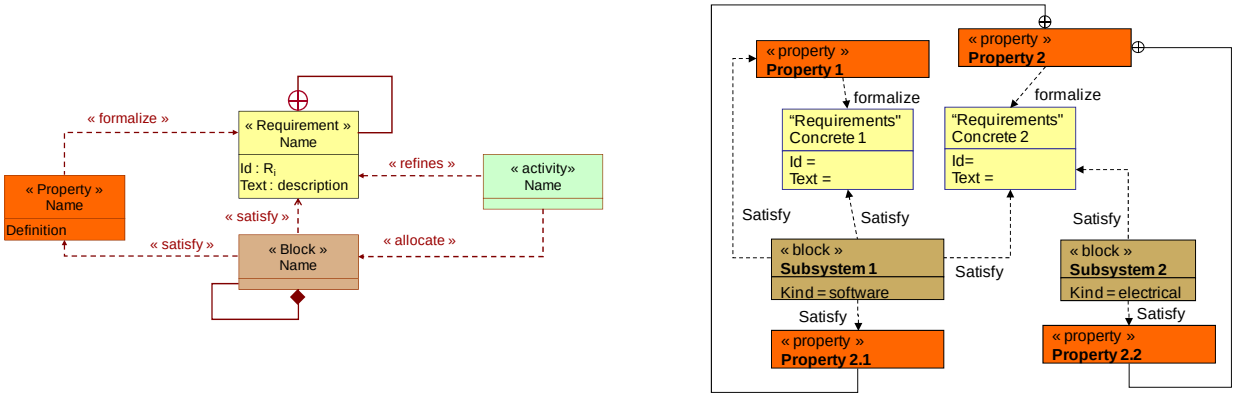


Fig 3. Requirements allocation

In order to provide proofs about requirements composition and allocation, we need a more formal description of the different safety properties, associated to each abstract and concrete requirement, respectively noted Pa(i) and Pc(j). In this way, we have extended the SysML meta-model to introduce *property* concept.



a) Requirements, property, activity and blocs b) Properties refinement

Fig 4. Requirements traceability

Property is a requirement stereotype that *formalizes* requirements textual description in terms of logic predicates and/or numerical parameters (figure 4a). It proposes a list of attributes involving language type, regular expression, kind, and state machine describing some sequential properties. This extension enables:

- 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[11] or can be reused as post-conditions by simulation and testing tools,
- to formally demonstrate that a logic combination of component's concrete properties Pc(j) establishes the system abstract properties Pa(i), using a theorem prover, such as COQ[12] (in figure 4a, P2 is composed of P2.1 and P2.2).

[9] SCADE is a Lustre-based Esterel Technologies product

[10] ControlBuild is IEC61131-3-based Geensys product

[11] UPPAAL is developed by the Information Technology Department of the Uppsala Univ. (Sweden) and the Computer Science Department of the Aalborg Univ. (Denmark).

[12] Huet G., Khan G. and Paulin-Mohring C, 1995. The Coq proof assistant, *Technical Report 178*, Inria.

Integration of tools for system specification with SysML, for control system design & simulation using trade-oriented languages (ControlBuild) and for proving safety properties (UPPAAL model checker), have been impleneted using XML and XSLT transformations based on a meta-model of requirements, properties and blocks shared concepts (Figure 4a). Following sub-sections illustrates all the steps of the proposed methodology using the INRS case study.

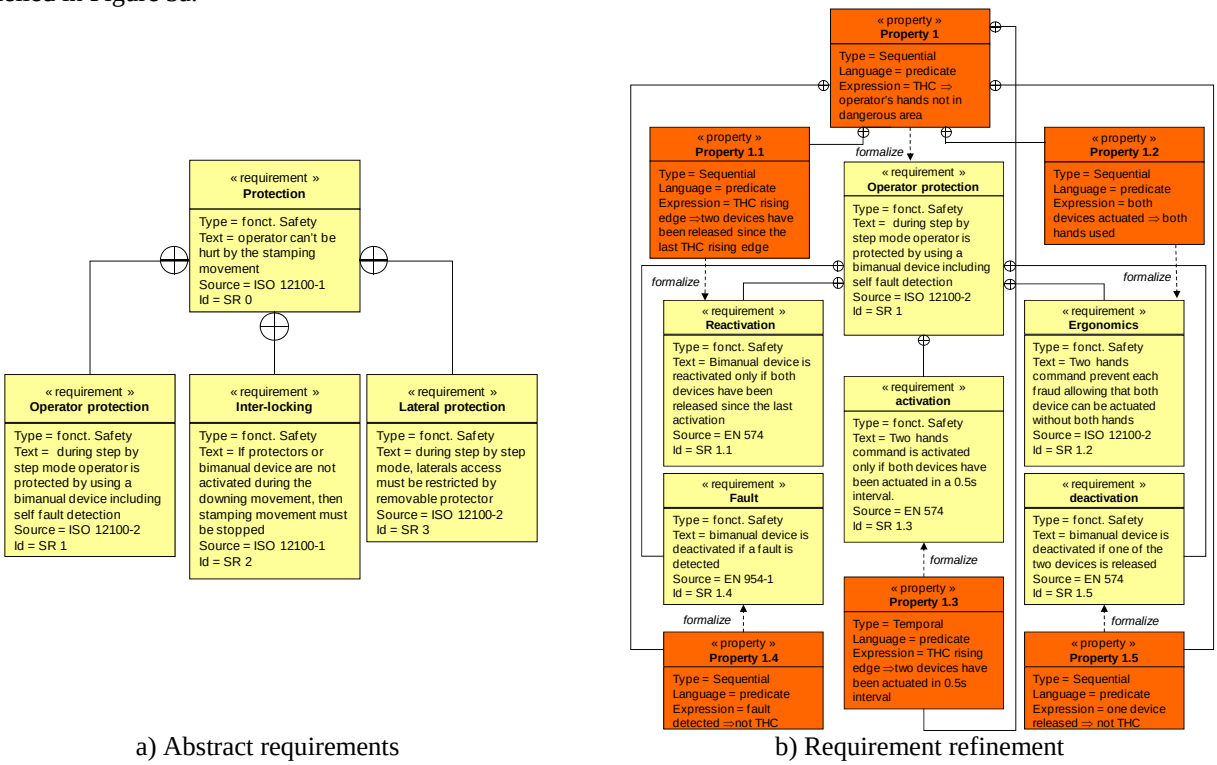
4 APPLICATION ON THE INRS CASE STUDY

4.2 Safety requirements modelling

Safety requirements for the mechanical press are issued from the ISO 12100-1 risk analysis including:

- a hazard identification and risk assesment based on risk analysis methods like fault tree analysis or Failures Modes Effects Criticality Analysis (FMECA).
- a risk reduction based on prevention measures, integration of safety devices and information notices.

Safety analysis shows that the most important risk is a crash risk of operator's hands during the stamping movement. It identifies two access types to the dangerous area, a front access and two lateral accesses. Because of the crankshaft position in the press architecture, lateral access are potentially damageable not only during the downing phase (when the tool stamp the iron sheet) but also during the rising phase (when the tool return to its initial position). So lateral access must be avoided during all the press movement (requirement SR3) and front access must only be avoided during the downing phase (requirement SR1).
According to the ISO 12100-2 protector choice algorithm, we choose removable protectors for the lateral access (SR3) and a two hands command (THC) for the front access (SR1). THC role is to require that the operator use his two hands to press on the THC two devices (kind of pushing buttons) if he wants to activate the stamping movement. All protection devices are interlocked with the press stamping movement. It means that if a protection is not activated during the stamping movement then the movement is stopped (Requirement SR2). These first set of abstract requirements are modelled in Figure 5a.



a) Abstract requirements b) Requirement refinement

Fig 5. Requirements modelling

Our study focuses onto the two hands command. Requirement SR1 is split according to EN954-1 general standard and to EN 574 two hands commands dedicated standard into a set of five detailed requirements related to:

- ergonomics of the two hands command (requirement SR1.2 prevent that THC can be activated with only one hand),
- self fault detection (requirement SR1.4 says that THC must be desactivated if a fault is detected),
- two hands command activation and deactivation (requirements SR1.1 means that THC is reactivated only if both devices have been released, SR1.3. means that THC is activated only if both devices have been actuated in a 0,5 s interval and SR1.5 means that THC is deactivated if one device is released.

These new sub-requirements are modelled by enriching the previous SysML Requirement Diagram and by associating semi-formal predicates associated to each requirement (figure 5b) using property stereotype. It means that satisfying

property P1 is equivalent to fully satisfy properties P1.1. to P1.5. Proving this equivalence between first order logic predicates can be done manually or using a theorem prover such as COQ. Note that a same approach must be used to refine requirement SR2 and SR3.
Next phase consist identifying the system architecture including software and hardware components in order to allocate each requirement on a supporting device.

4.3 Safety requirements allocation

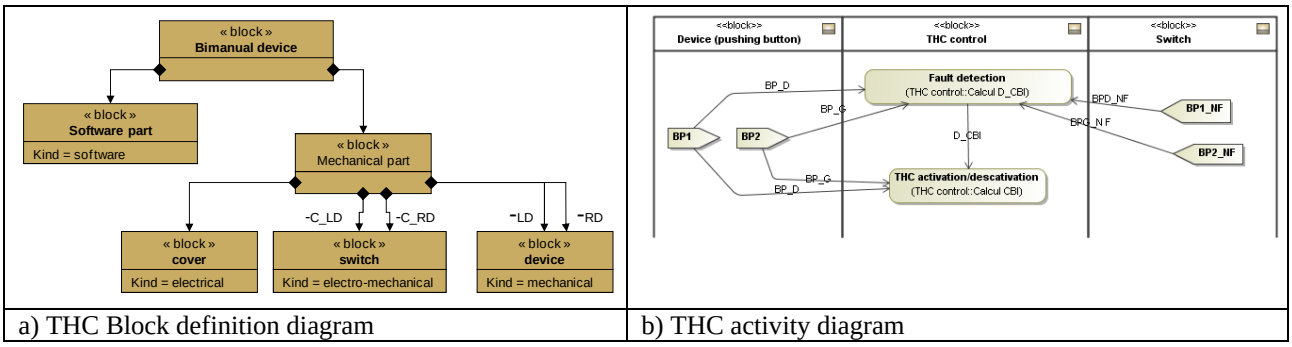
Requirements allocation using “satisfy” link enable to link one or more blocks (system component) to one or more requirements. Two hands command requirements allocation is given in table 1.

Requirements	Components
SR 1	All
SR 1.1	Software / Device
SR 1.2	Cover / Switch
SR 1.3	Software / Device
SR 1.4	Software / Device
SR 1.5	Software / Device

Table 1. Requirements allocation

Description of structural and functional views of the press-system is done using respectively:

- block diagrams that define the system component architecture. For example classical THC has been architecture involve devices (pushing buttons), a cover, two switches for faults detections, and software component (Figure 6a).
- activity diagrams that define operation calls and data exchange. For example, Figure 6b describes operations that are supported by the tow hand command.



a) THC Block definition diagram | b) THC activity diagram

Fig 6. Block definition and activity diagrams

Requirements allocation is modelled using “satisfy” link between requirements and blocks. For example, THC requirements allocation leads to the figure 7 SysML diagram.

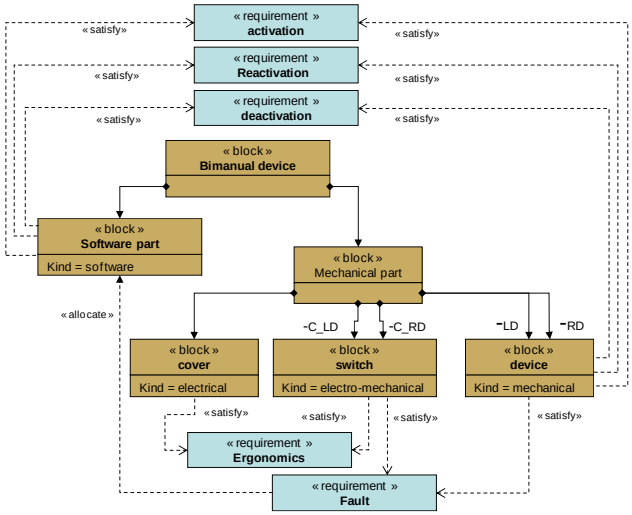


Fig 7. THC requirements allocation

Properties associated to requirements become the properties that each component must satisfy. This operation can lead:

- to decompose a property into several sub-properties when a requirement and its associated property is allocated to several component; this is the case for properties 1.1, 1.3, 1.5 which are projected on *software* component (properties 1.1.a, 1.3.a, 1.5.a) and on *device* component (1.1.b, 1.3.b) and for 1.4 property which is split into *software* (1.4a), *device* (1.3b/1.4c) and *switch* blocks (1.4b).
- to possibly merge properties when a same component satisfies several properties; this is the case of the *device* block which satisfies part of 1.3 and 1.4 properties that can be merged into 1.3b/1.4c property. Merging operation between predicates results from a logical AND between elementary predicates.

Correctness of these decomposition and merging operations can be established and demonstrated using manual or a theorem prover. It leads to the final THC requirements allocation given by the figure 11 where LD and RD represent respectively the activation of left and right devices or pushing buttons, C_LD and C_RD represent respectively the activation of fault switch associated to left and right pushing devices while HTC represent the activation of two hand command.

Next phase consists in a detailed design of the identified components (and more particularly the behavioural description of the system), and in proving that the system design is compliant with the identified properties.

4.4 Safety requirements verification

The behavioural design of the THC software control part has to satisfy the properties 1.1a, 1.3a, 1.4a and 1.5a identified in figure 8. This design must also be compliant with the functional architecture shown in figure 8. It can be done using *internal block diagrams* (IBD) and *state diagrams* provided by SysML. Note that the synchronous data flow structure provided by IBD facilitates its transcription into dedicated control design tool based on synchronous DES models such as Simulink, Scade or ControlBuild (figure 9).

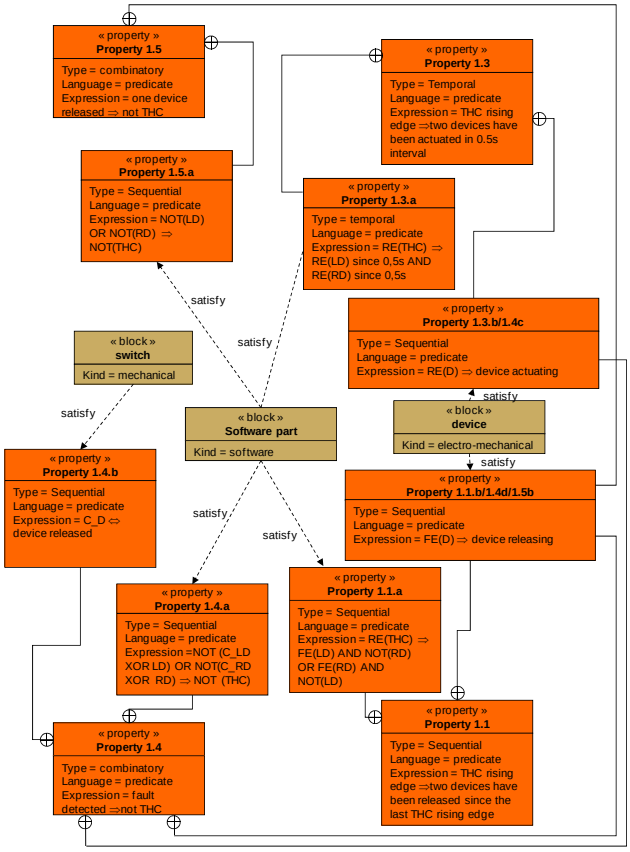


Fig 8. THC properties

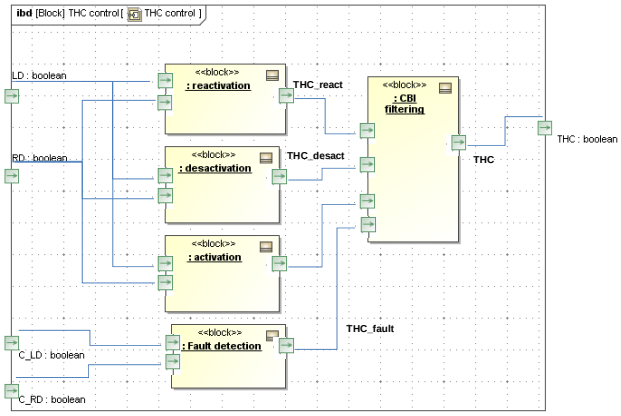


Fig 9. THC Intrenal Block Diagram

Verifying that all behavioural descriptions of the blocks satisfy the local properties identified by figure 8 requires the use of a model checker and the translation of the behavioural models (internal block diagrams and state diagrams) into a dedicated language supported by the model checker. For this study, UPPAAL model checker has been chosen. State-based machine are used for the description of the behavioural model and temporal logic for the description of the properties to be proved. Translation of the behavioural models from SysML to UPPAAL is quite obvious due to the use of State machines both in SysML and UPPAAL modelling. For the properties translation, three kinds of properties can be mentioned:

- combinatory properties (P1.5.a ; P1.3.a)
- sequential properties (P1.4.a)
- temporal properties (P1.1.a).

Translation of the combinatory properties is the simpler one. Indeed, they can all be expressed as "it is always true that …" and can be modelled using AG operator in UPPAAL :

- AG (NOT(LD) OR NOT(RD) ⇒ NOT(HTC) [P1.5a]
- AG (NOT(LD XOR C_LD) OR NOT(RD XOR C_RD) ⇒ NOT(THC) *[P1.3a]*

Translation of sequential properties is more difficult. It requires the use of an observer; i.e. a state machine that registers the system evolution and that introduces a faulty state in which the system must never go. For example, P1.4a is represented by such an observer which involves 3 states (figure 10): the first one with THC active, the second one with THC inactive and both devices released and a third one, halfway, with THC inactive and only one device released. The property P1.4a formalizes that activating THC is only possible if both devices have been released since the previous activation. So when state 3 is active (THC have just been deactivated), it is possible to reach state 1 only by activating state 2. Consequently, the faulty state, corresponding to the evolution that must be avoided, can be reached from state 3 if THC is activated. Verification of the properties is done by proving using UPPAAL that the faulty state can never be reached.

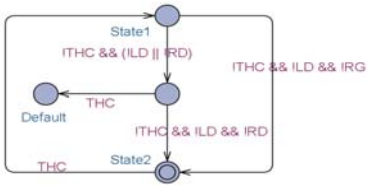


Fig 10. Observer for property P1.4a

All these translations have been implemented using model transformation as recommended by Model Driven Engineering. It is based on the definition of a XML common format using XSLT transformations to enable the exchange of data and models between different commercial tools, such as those used for the INRS case study: SysML system specification (Magic Draw), control system design and simulation using trade-oriented languages (ControlBuild), model-checker for proving the safety properties (UPPAAL).

4.5 Discussion

In this study, only safety properties have been focused on, the functional properties have not been considered. This is justified by the chosen implementation that follows the recommendations provided by the standards related to manufacturing machines, i.e. to separate safety related components from nominal components. Consequently, the chosen implementation in Safe-PLC is based on a safety filter that is placed between control system and physical system and that

enables or disables the control system outputs [22]. In other words, it means that this filter ensures that the safety requirements are always satisfied whatever the outputs of the control system are (figure 11).

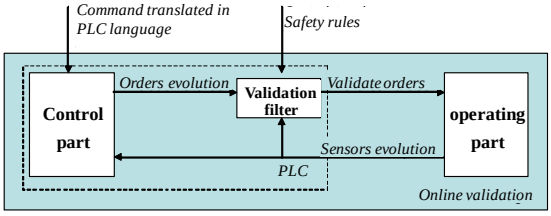


Fig 11. Safety filter [22]

Benefit is that functional components of the machine control can be developed according classical approaches while more costly methods needed to reach level 2, 3 or 4 of SIL (Safety Integrity Level) are limited to safety components that are involved in safety filter. Note that splitting functional and safety requirements is not always an obvious task: if a component realizes both safety and nominal functions then it must be consider entirely as safety related.

5 CONCLUSION

There is a growing interest in formal 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 IEC 61508 safety-related standard strongly recommends the use of formal verification methods to be applied in the certification process by the suppliers, integrators, or independent external authorities, but without defining how they can be applied. On the other way, system engineering semi-formal approaches appears to be essential for capturing and structuring the requirements of a complex system and for proposing high level system functional architectures that cope with the identified requirements.

Trying to match theses apparently opposite way of thinking, we have proposed a system modelling approach that combines non-formal methods based on SysML requirements and block diagrams, and formal methods such as model-checking to prove that the local behaviour of each system component contributes to satisfy system requirements. This work is based on the mapping between structured models of the requirements to be satisfied, the functions to be realized and the component to be implemented.
One limit must be mentioned: composition of properties is assumed to be described by a simple logical expression (i.e. satisfying a property is assumed to be equivalent to satisfy a logical combination of sub-properties) but it is not always true especially when manipulating temporal expression involving operators such as *before*, *after* or *until*. Further work is currently developed in this way. Benefit relies on the requirements traceability all along the system engineering process and on the associated verification capabilities. 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 safety integrity highest levels, 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.

Acknowledgment

This work has been partially funded by INRS in the framework of Dominique Evrot PhD thesis (*Contribution à la vérification d'exigences de sécurité: application au domaine de la machine industrielle, Thèse de l'Université H. Poincaré, juillet 2008, in French*). We would address our acknowledgement to our INRS partners who participated to this project.

6 REFERENCES

[1] Fusaoka A., Seki H., Takahashi K. (1983). A description and reasoning of plant controllers in temporal logic. *Proceedings of the 8th Int. Joint Conference on Artificial Intelligence*, pp. 405-408, 1983, Karlsruhe, Germany,.

[2] Ramadge P.J., Wonham W.M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, Vol. 25, n° 1.

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

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

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

[6] Cassandras C.G., Lafortune S. (1999). *Introduction to Discrete Event Systems*. Kluwer Academic Publisher, ISBN 0-7923-8609-4, 1999.

[7] Roussel J.M., Faure J.M. (2002). An algebraic approach for PLC programs verification. *6th International Workshop on Discrete Event Systems (WODES'02)*, pp. 303-308, Zaragoza, Spain, , 2-4 October, 2002

[8] Barragan S., Roth M., Faure J. M. (2005). Obtaining temporal and timed properties of logic controllers from fault tree analysis, in proc of*12th IFAC INCOM*, vol 1, pp 241-246, 17-19/05, St-Etienne, France.

[9] Evrot D., Pétin J-F., Mery D. (2005). Formal specification of safe manufacturing machines using the B method: application to a mechanical press, *in proc. 12th IFAC INCOM*, vol1, pp 277-282, 17-19/05, St-Etienne, France.

[10] Achatz R. (2006). Requirements engineering: A key success factor, *Automation Technology in Practice, Issue 3, November 2006.*

[11] Johnson T. L. (2007). Improving automation software dependability: a role for formal methods ? *Control Engineering Practice, Volume 15, Issue 11*, November 2007, Pages 1403-1415.

[12] Sheard S. (2006). Complex Systems Science and its Effects on Systems Engineering. *European Systems Engineering Conference*, 18-20 September 2006, Edinburgh, UK.

[13] Bonfé M., Fantuzzi C. (2005). Object-Oriented modeling of logic control systems for industrial applications. *Automation Technology in Practice, Issue 2, October 2005.*

[14] Auinger F., Brennan R., Christensen J., Lastra L.M., Vyatkin V. (2005). Requirements and solutions to Software encapsulation and engineering in next generation manufacturing systems: OOONEIDA approach. *International Journal of Computer Integrated Manufacturing, Vol. 18(7), p572-585.*

[15] Ferrarini L, Veber C., Schwab C., Tangermann M., Prayati A. (2005). Control functions development for distributed automation systems using the Torero approach. *16th IFAC World Congress*, July 4-8, 2005, Prague (Czech Republic).

[16] Tranoris C., Thramboulidis K. (2006). A tool supported engineering process for developing control applications, *Computers in Industry, Vol. 57(5), June 2006, Pages 462-472.*

[17] Katzke U., Vogel-Heuser B. (2005). UML-PA as an Engineering Model for Distributed Process Automation. *Proceedings of the 16th IFAC world Conference*, 2005. Prag, Czech Republic.

[19] Abrial J.R (2005), Case study of a complete reactive system in Event-B: a mechanical press controller, *tutorial 3 of ZB'2005*, Guildford, UK, 13-15/04/2005

[20] Evrot D., Pétin J-F., Morel G., Lamy P., 2007. Using SysML for identification and refinement of machinery safety properties. *Proceedings of IFAC Workshop on Dependable Control of Discretes Systems*, June 2007, Cachan, France.

[21] Blaise, J. C., Lhoste, P., Ciccotelli, J.(2003). Formalisation of normative knowledge for safe design. In : *Safety Science, 41, 241-261*.

[22] P. Marangé, F. Gellot, B Riera (2009). Proposition of plant model for the verification of system safety. *Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing* (INCOM'09), June 3-5 2009, Moscow, Russia.