HAL
open science

Traçabilité et vérification d'exigences de sécurité

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

To cite this version:

Jean-François Pétin, Dominique Evrot, Gérard Morel, Pascal Lamy. Traçabilité et vérification d'exigences de sécurité. Génie logiciel : le magazine de l'ingénierie du logiciel et des systèmes, 2010, 95, pp.27-37. hal-00549136

HAL Id: hal-00549136

https://hal.science/hal-00549136v1

Submitted on 21 Dec 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.

Traçabilité et vérification d'exigences de sécurité

JEAN-FRANÇOIS PETIN, DOMINIQUE EVROT, GERARD MOREL ET PASCAL LAMY

Résumé : Les normes industrielles relatives à la conception de systèmes programmables intégrant des fonctions à la sécurité, telles que la norme IEC 61508, recommandent fortement l'utilisation de méthodes formelles pour maîtriser la complexité du cycle de développement et en particulier la traçabilité des exigences de sécurité. Cet article traite de la vérification des exigences en matière de sécurité pour la conception de la commande programmée de systèmes de sécurité. Il combine une approche qualitative de spécification fonctionnelle et structurelle du système à concevoir sous la forme de diagrammes SysML et de techniques quantitatives de vérification formelle, telle que le *model-checking*, permettant de garantir que le comportement du système préserve les propriétés de sécurité identifiées dans les modèles SysML.

Mots clés : SysML, ingénierie système, spécification, exigences de sécurité, vérification formelle

1 INTRODUCTION

Cet article traite de la vérification des exigences de sécurité pour la conception de Systèmes E/E/PE (Electrical/Electronic/Programmable Electronic) dits de sécurité. Ces systèmes incluent les Systèmes Instrumentés de Sécurité (SIS), qui sont des systèmes dédiés à la sécurité et qui visent à réduire les risques à des niveaux acceptables, et plus généralement des systèmes non dédiés à la sécurité mais dont la défaillance peut provoquer des accidents graves vis-à-vis des biens, des personnes ou de l'environnement.

Pour les premiers, les travaux présentés dans cet article font l'objet d'une expérimentation sur un cas d'étude proposé par l'INRS (Institut National de Recherche et de Sécurité) qui concerne la commande discrète d'une machine industrielle équipée de dispositifs de protection des opérateurs tels que barrière immatérielle, arrêt d'urgence, commande bimanuelle… Pour les seconds, l'approche proposée est actuellement évaluée dans le cadre du développement de commandes critiques à base de COTS (Component Off The Shelf) dans le domaine du ferroviaire. Pour des raisons de taille et de confidentialité des résultats, cet article se focalise sur le cas d'étude proposé par l'INRS.

D'un point de vue industriel, le développement de tels systèmes est fortement encadré par un ensemble de normes parmi lesquelles nous pouvons citer la norme IEC 61508 et ses déclinaisons sectorielles (IEC 62061 dans le domaine de la machine, IEC 61511 pour les processus industriels, IEC 61513 dans le nucléaire, EN 50126, 50128 et 50129 dans le ferroviaire, IEC 26262 dans l'automobile…). Ces normes reposent toutes sur le concept de niveaux d'intégrité de sécurité (SIL pour Safety Integrity Level) proposant une classification des objectifs de sécurité imposés par les clients ou les autorités de certification en fonction notamment de la fréquence d'occurrence et de la gravité des événements redoutés. Compte tenu du niveau de SIL visé pour un système donné, les normes :

- fixent des objectifs quantitatifs à atteindre en termes de probabilité d'occurrence de la défaillance dans le cas de modes de sollicitation continue (PFH – *Probability of Failure per Hour*) ou occasionnelle (PFD – *Probability of Failure on Demand*),
- proposent des recommandations qualitatives concernant le cycle de développement du système de sécurité, notamment en termes de traçabilité des exigences.

Pour répondre aux objectifs quantitatifs relatifs aux valeurs des PFH et PFD, les ingénieurs en charge de la sûreté de fonctionnement des systèmes de sécurité disposent de modèles stochastiques permettant l'évaluation d'une fiabilité

prévisionnelle comme, par exemple, les arbres de défaillances, les chaines de Markov ou encore les réseaux de Petri stochastiques. Cette évaluation concerne essentiellement les parties matérielles des systèmes de sécurité dans la mesure où les erreurs systématiques des logiciels sont difficilement appréciables par ce type de techniques. Pour atteindre les niveaux de fiabilité requis, ces études conduisent à la définition d'architectures matérielles incluant des redondances ou de la diversité fonctionnelle. À titre d'exemple, dans le cas d'étude proposé par l'INRS, la commande de la machine industrielle et de ses dispositifs de protection est implantée sur un Automate Programmable Industriel de Sécurité (APIdS) dont l'architecture comporte plusieurs voies redondantes éventuellement votées. Dans le cadre d'une commande embarquée à base de COTS, la solution d'implantation pourrait reposer sur la mise en œuvre d'une carte électronique à base de FPGA redondés.

Néanmoins, contrairement aux applications de commande basées sur des technologies câblées, ces précautions architecturales ne seront pas suffisantes pour prendre en compte les éventuelles erreurs systématiques liées à la mise en œuvre d'une commande programmable et à sa complexité inhérente. C'est pourquoi, les normes de la famille IEC 61508 mettent l'accent sur un deuxième point concernant la maîtrise des cycles de développement. Elles recommandent notamment l'utilisation de modèles et de méthodes, plus ou moins formels, permettant d'assurer la traçabilité des exigences ou encore la vérification des propriétés de sécurité (par exemple pour démontrer l'absence d'événements redoutés dus à un défaut logiciel).

Cet article a pour objectif de répondre à cette exigence de maîtrise du cycle de développement en proposant une démarche méthodologique basée sur l'utilisation conjointe du langage SysML et de langages spécifiques à un métier (DSL pour Domain Specific Language). Le premier propose un formalisme de spécification unifié autour de trois piliers relatifs à la modélisation des exigences, de la structure et du comportement d'un système, tandis que les seconds s'avèrent indispensables pour la conception détaillée de la commande et la vérification formelle de ses propriétés dynamiques.

La section 2 définit le contexte de l'étude et formalise le problème. La section 3 montre les complémentarités entre les approches axées sur l'Ingénierie Système et sur les modèles comportementaux de la commande des Systèmes à Événements Discrets (D.E.S). La section 4 présente la démarche méthodologique proposée assurant la traçabilité et la vérification des exigences de sécurité. La section 5 illustre la proposition sur le cas d'étude INRS concernant la commande d'une presse industrielle incluant des fonctions de sécurité relative à la protection des opérateurs. La dernière section fournit quelques conclusions et ouvre des perspectives pour passer de l'étude de faisabilité proposée dans cet article à une formalisation de la démarche nécessaire pour son application dans le contexte industriel des systèmes critiques.

2 DEFINITION DU PROBLEME

Le problème posé peut s'énoncer simplement : il s'agit de s'assurer qu'un système de sécurité, noté S, satisfait un ensemble d'exigences, noté E.

$$S \models E \quad (1)$$

Les systèmes de sécurité étudiés intègrent de nombreux composants de nature hétérogène dans les domaines logiciels, mécaniques, électriques ou pneumatiques. Si l'on considère que le réseau d'interaction entre composants peut provoquer, notamment aux interfaces, des comportements émergents indésirables, il en résulte que les propriétés de sécurité caractérisant globalement le système ne peuvent se réduire à l'assemblage des propriétés élémentaires de chacun de ses composants. Il s'agit donc de s'assurer que la composition des n composants d'un système, notés $\{S_1, \ldots, S_n\}$, satisfait l'ensemble des exigences.

$$S_1 \times \ldots \times S_n \models E \quad (2)$$

Dans le domaine des systèmes à événements discrets, Fusaoka [1] et la théorie de la commande supervisée [2] ont abordé ce problème au travers de la synthèse, dans le formalisme des automates, d'un modèle de superviseur S (non connu) à partir d'un modèle de procédé, noté G, et d'une spécification K, tels que la composition synchrone de S et G soit le sous-langage maximal contrôlable de K par rapport à G.

D'autre part, les exigences sont habituellement modélisées selon plusieurs niveaux d'abstraction ou de décomposition selon les parties prenantes qui les expriment ou encore les niveaux de finesse avec lesquels elles sont décrites. Le problème initialement posé consiste donc à s'assurer que la composition des composants d'un système satisfont un ensemble d'exigences, noté $E = \{E_1, E_2, \ldots E_m\}$, avec :

- $S_i \models \{E_k\}_{k \in [1, m]}$, ce qui signifie qu'un constituant S_i peut satisfaire plusieurs exigences appartenant à l'ensemble E,
- $S_j \times \ldots \times S_l \models E_i$, avec $j \leq l \leq n$, ce qui signifie qu'une exigence E_i peut être satisfaite par la composition de plusieurs composants appartenant à l'ensemble S.

Enfin, si l'on considère qu'un processus classique de développement en ingénierie système [3] est basé sur l'expression du besoin, la formalisation des exigences ainsi que la définition d'architectures fonctionnelle et structurelle, assurer la traçabilité des exigences de sécurité peut s'entendre au travers de la définition :

- d'une architecture fonctionnelle, au sens classique ou SADT du terme, identifiant un ensemble F de fonctions et d'une application *stfby* (*satisfied by*) de E $\rightarrow$ F, permettant d'associer à chaque exigence identifiée une ou plusieurs fonctions permettant d'y répondre,
- d'une architecture structurelle identifiant un ensemble S de composants et d'une application *reaby* (*realised by*) de F $\rightarrow$ S, permettant d'associer à chaque fonction un ou plusieurs composants permettant sa réalisation,
- et enfin d'une application permettant d'assurer la traçabilité entre exigences et composants au travers de la composition des fonctions *stfby* et *reaby*.

Deux sous-objectifs complémentaires découlent donc de cette analyse du problème : il s'agit, d'une part, d'assurer la traçabilité des exigences de sécurité tout au long du cycle de développement et, d'autre part, de prouver que les exigences de sécurité sont satisfaites (2). La littérature scientifique couvre séparément ces deux questions au travers des approches en ingénierie des systèmes et des techniques de validation & vérification formelle.

3 ETAT DE L'ART

3.1 APPROCHES EN INGENIERIE SYSTEME

Selon l'AFIS (Association Française d'Ingénierie Système), l'ingénierie système peut se définir comme « *une démarche méthodologique interdisciplinaire qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante aux besoins d'un client tout en satisfaisant l'ensemble des parties prenantes* ». Cette démarche intègre de nombreux processus standardisés par les normes IEEE 1220 ou ISO 15288 parmi lesquels nous retrouvons la définition et l'analyse des exigences, la définition des architectures fonctionnelle et structurelle et le processus de validation & vérification.

Les modèles et outils utilisés en ingénierie système restent, pour certains, très spécifiques à un processus particulier : par exemple Doors[1] pour la modélisation des exigences ou Reqtify[2] pour la gestion de la documentation et de la traçabilité. Mais pour la plupart, ils proposent un vaste ensemble de formalismes graphiques qui permettent d'appréhender les différents points de vue et processus (exigences, architectures fonctionnelle, structurelle, informationnelle, comportementale…). En particulier, les langages de modélisation tels que ceux proposés par UML2 et son profil SysML pour l'ingénierie système [4], standardisé par l'OMG en collaboration avec l'INCOSE[3] ou encore par KAOS [5] manipulent des diagrammes où les exigences peuvent être identifiées, structurées et associées à des architectures fonctionnelles et structurelles (Figure 1).

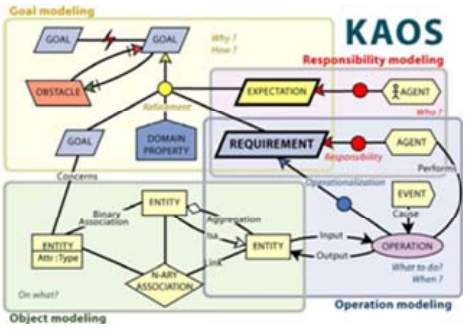


Figure 1 : KAOS

Cependant, tous ces langages ne restent bien souvent que des boîtes à outils sans véritable démarche méthodologique associée et n'offrent pas le caractère formel qui permettrait d'avoir une évaluation quantitative des résultats d'ingénierie.

Sur le plan méthodologique, des efforts ont été accomplis en ce sens dans le domaine de l'Automatique, notamment autour des modèles UML et de la norme IEC 61499 dans le cadre des projets Oooneida [6], Torero [7] ou Corfu [8] ou encore du profil UML-PA (UML for Process Automation) [9]. En ce qui concerne la formalisation, des correspondances entre diagrammes comportementaux UML ou SysML et des modèles dynamiques formels tels que les réseaux de Petri ont été établis, d'autres approches préférant l'utilisation directe de langages formels tels que le langage B [10] pour assister la phase de spécification [11]. La démarche proposée dans cet article est une contribution à cet effort méthodologique.

3.2 TECHNIQUES DE VERIFICATION FORMELLE

En automatique des Systèmes à Événements Discrets, deux grandes familles de méthodes peuvent être utilisées pour apporter une preuve formelle qu'un modèle comportemental de système vérifie une propriété.

La première famille de méthodes, dites par assertion, repose sur l'analyse d'un modèle comportemental donné vis-à-vis d'une propriété énoncée sous la forme d'un prédicat logique ou temporel. La vérification proposée est donc réalisée *a pos-*

teriori. Parmi ces techniques, la plus classique est le *model-checking* [12], qui consiste à construire et à explorer un espace d'états afin de vérifier le respect d'un prédicat LTL, CTL ou CTL* pour chacun des états atteignables.

La deuxième famille de méthodes, dites par raffinement, consiste à élaborer un modèle *a priori* correct par construction. Parmi ces techniques, nous pouvons citer la synthèse de la commande basée sur la théorie de la commande supervisée, sur la composition d'automates à états finis et l'élimination d'états interdits [13].

Plus précisément dans le domaine du développement de systèmes de commande, ces approches sont appliquées :

- pendant la phase de conception, pour prouver les propriétés de modèle de commande,
- pendant la phase d'implantation, pour prouver que le code produit vérifie bien certaines propriétés [14] ; ce code peut être sous forme d'un des langages de la norme IEC 61131-3 en cas d'implantation sur un automate programmable industriel (PLC) ou sous forme VHDL pour une implantation sur FPGA.

En raison du phénomène d'explosion combinatoire induit par l'exploration de l'espace d'état ou la composition d'automates, ces techniques s'avèrent très efficaces pour les composants de petite taille. De plus, l'identification et la formalisation des propriétés restent des tâches difficiles qui conditionnent l'ensemble du processus de vérification ou de synthèse. Une mauvaise formalisation entraînera évidemment un mauvais diagnostic quant au respect d'une propriété ou à une synthèse de commande trop ou pas assez restrictive selon la nature de l'erreur de spécification.

Ces méthodes devront donc être appliquées de postérieurement à une analyse « système » permettant d'isoler puis de composer des composants de taille réduite munis de leurs propriétés locales [15-16]. En ce sens, notre approche sera basée sur l'utilisation de SysML en tant qu'outil de modélisation « système » aboutissant à une vision consensuelle des exigences et des architectures et qui retarde, en quelque sorte, l'utilisation de langages plus spécifiques (DSL) couvrant la modélisation de la dynamique de la commande et la vérification par *model-checking* de ses propriétés.

4 TRAÇABILITE ET VERIFICATION DES EXIGENCES DE SECURITE

4.1 MODELISATION DES EXIGENCES

Selon l'AFIS, *« une exigence est un énoncé qui prescrit une fonction, une aptitude ou une caractéristique auxquelles doit satisfaire un produit ou un système dans un contexte donné »*. Elle s'exprime en termes de missions ou de performances (exigences fonctionnelles) mais également en termes d'aptitudes opérationnelles comme la sécurité ou la fiabilité, de contraintes environnementales, normatives ou de développement (exigences non fonctionnelles).

La modélisation des exigences repose sur un processus incrémental basé sur des mécanismes de raffinement et de décomposition permettant d'appréhender différents niveaux d'abstraction :

- le raffinement consiste à transformer une exigence en une nouvelle exigence contenant une information plus précise et des niveaux de détails supplémentaires ; à titre d'exemple, l'exigence « la commande bimanuelle est activée par un appui simultané des deux mains sur les interfaces » peut être raffinée en l'exigence « la commande bimanuelle est activée par les appuis des deux mains sur l'interface, les deux appuis étant séparés au maximum de 500 ms »,
- la décomposition consiste à répartir une exigences sur plusieurs sous-exigences, ce qui signifie que l'exigence composée est satisfaite si et seulement si toutes les sous-exigences sont satisfaites ; à titre d'exemple, l'exigence « protection opérateur » peut être décomposée en « protection par commande bimanuelle » et « protection latérale ».

En SysML, le formalisme utilisé pour la modélisation et la structuration des exigences est le diagramme des exigences. Il définit une exigence comme un objet possédant des attributs propres (ID, texte, source de l'exigence, type, niveau de priorité, méthode de vérification, etc.). Ces objets peuvent être en relation avec d'autres exigences (liens de composition et de dérivation pour représenter respectivement les mécanismes de décomposition et de raffinement) ou d'autres objets supports (activités, composants, etc.). La Figure 2 illustre la décomposition d'une exigence abstraite en plusieurs exigences plus concrètes.

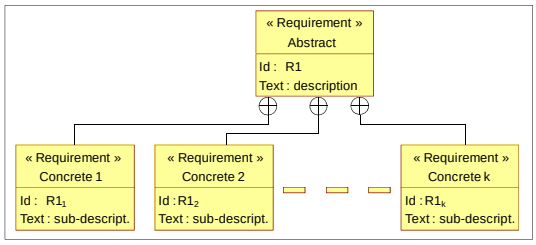


Figure 2 : Décomposition des exigences

4.2 DEFINITION DES ARCHITECTURES

L'architecture fonctionnelle décrit et hiérarchise l'ensemble des fonctions qu'offre le système à développer. Cette analyse fonctionnelle, équiva-

lente à ce que l'on peut réaliser avec un formalisme de type SADT, peut être réalisée de trois manières différentes en SysML [17] :

- par l'intermédiaire d'un cas d'utilisation, même si ces diagrammes offrent un pouvoir d'expression limité, notamment en termes de typage des flux et de hiérarchisation,
- par l'intermédiaire d'un diagramme interne de blocs ; ces diagrammes étant plutôt réservés à la description des composants du système, la définition de deux stéréotypes de bloc (un pour les fonctions, un pour les composants) s'avère nécessaire pour éviter toute confusion sémantique.
- par l'intermédiaire d'un diagramme d'activité avec une restriction sur l'utilisation des flots de contrôle. En effet, ces diagrammes étant dédiés à la modélisation comportementale, aucun flot de contrôle ne sera employé dans le cadre d'une analyse fonctionnelle.

Compte tenu de la nécessité de faire apparaître les relations fonctions / composants, nous avons choisi de différencier les diagrammes et donc d'adopter le diagramme d'activité pour la représentation de l'architecture fonctionnelle.

L'architecture structurelle décrit et hiérarchise les composants du système à développer. En SysML, le composant se décrit à l'aide du concept de bloc qui est un stéréotype de classe et au travers de deux types de diagrammes :

- le diagramme de définition de bloc (BDD pour Block Definition Diagram) permet de représenter la structure du système au travers de relations composés/composants et similaires à ce que l'on peut réaliser à l'aide d'un diagramme de classes,
- le diagramme interne de bloc (IBD pour Internal Block Diagram) est un diagramme de type flot de données qui permet la description détaillée d'un composé, dans l'esprit des modèles flot de données offerts par des outils de conception de systèmes de commande tels que Simulink[4], Scade[5] ou ControlBuild[6] [18].

4.3 Mecanismes de traçabilite

Trois relations SysML nous permettent d'établir des correspondances entre fonctions, exigences et composants (Figure 3).

La relation SysML « *Allocate* » associe une fonction à un ou plusieurs composants (modélisés sous forme de bloc SysML) sur lesquels repose sa réalisation. Elle correspond à l'application *reaby*.

La relation SysML « *Safisfy* » associe une fonction à une ou plusieurs exigences qu'elle est sensée satisfaire. Elle correspond à l'application *stfby*. Par extension, la composition *reaby* o *stfby* qui établit un lien entre un ou plusieurs composant et une ou plusieurs exigences sera aussi représentée par le lien SysML « *satisfy* ».

Enfin, la relation SysML « *Refine* » associe une propriété décrite sous la forme d'une contrainte SysML à une exigence. Cette propriété correspond à une formalisation de l'exigence sous la forme de prédicat.

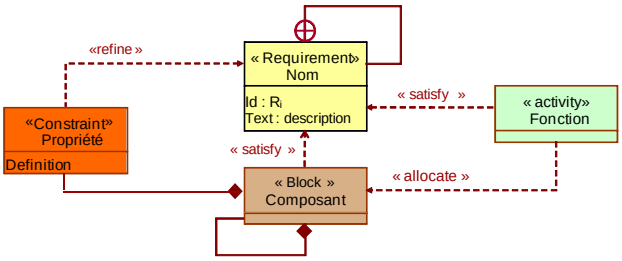
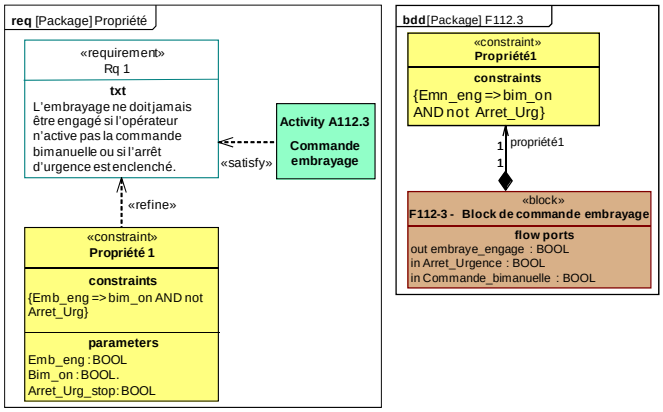


Figure 3 : Meta-modèle pour la traçabilité

Enfin, il convient d'établir les relations entre les variables manipulées par la contrainte – c'est-à-dire les variables intervenant dans le prédicat caractérisant la propriété – et les variables manipulées par les composants – c'est-à-dire les ports des blocs. La solution proposée repose sur l'utilisation du diagramme paramétrique. Les hypothèses sont les suivantes :

- les liens « *refine* », « *satisfy* » et « *allocate* » ont respectivement permis d'établir les relations entre une propriété Pi et une exigence Ei, entre une exigence Ei et une fonction Fi et entre une fonction et un composant Si (Figure 4a),
- par composition, il existe donc une relation entre un composant S_i et une propriété P_i si *satisfy*(F_i) = E_i et *Refine*(E_i) = P_i (Figure 4b).



a) Exigence / Propriété / Fonction b) Propriété / Composant

Figure 4 : Traçabilité

La relation entre variables de la contrainte et variables des blocs (flowport) est alors représentée dans un diagramme paramétrique tel que celui de la Figure 5.

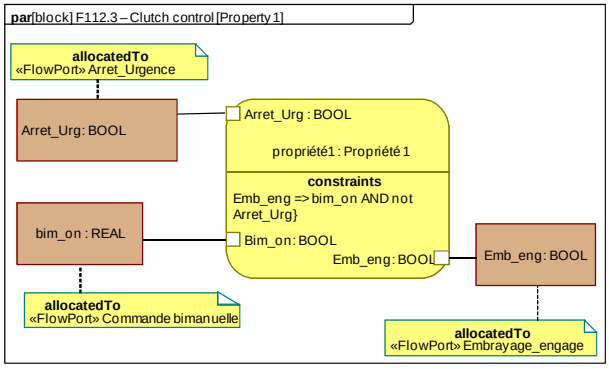


Figure 5 : Diagramme paramétrique

Il faut également souligner que ces mécanismes de traçabilité doivent opérer pour tous les niveaux d'abstraction. En d'autres termes, cela signifie que les relations d'allocation et de satisfaction doivent être préservées quel que soit le niveau de décomposition ou de raffinement. À titre d'exemple, la Figure 6 présente une propriété P1 associée à une exigence Rq1 satisfaite par un seul composant S1. En revanche, la propriété P2 est associée à une exigence Rq2 satisfaite grâce à deux composants S1 et S2. Ces deux composants doivent alors être associés à deux sous-propriétés disctinctes P2.1 et P2.2, elles-mêmes issues de la décomposition de P2.

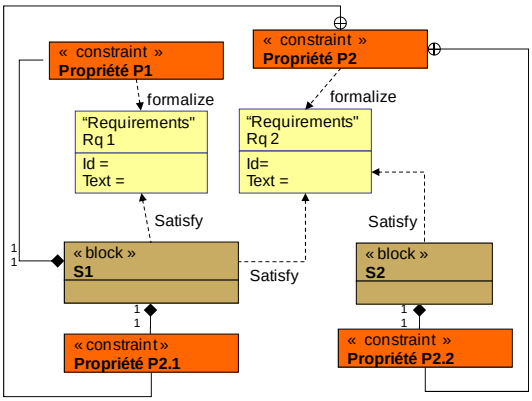


Figure 6 : Décomposition de propriétés

Démontrer la cohérence de la décomposition (P2.1 P2.2 P2) est assez trivial lorsque les opérateurs de composition des propriétés sont limités à des opérateurs logiques ET, OU mais peut s'avérer beaucoup plus complexe si les propriétés sont décrites à l'aide d'opérateurs temporels.

4.4 Verification des exigences

La conception détaillée de la dynamique de la commande et la vérification de ses propriétés de sécurité requièrent l'emploi de modèles et d'outils spécifiques (DSL). Les principaux objets SysML réutilisés en entrée des DSL sont :

- les diagrammes IBD qui fournissent l'architecture logique de la commande, interprétable dans les outils classiques flots de données tels que Simulink, Scade ou ControlBuild comme un réseau interconnecté de composants (ou modules ou nœuds selon les outils utilisés) muni de ports d'entrée et de sortie typés.
- les contraintes SysML qui permettent de caractériser la ou les propriétés logiques ou temporelles que devra vérifier chacun des composants et composés
- les fonctions SysML qui permettent de caractériser la ou les fonctions que devra remplir chacun des composants et composés.

Sur la base de cette spécification, les ingénieurs spécialisés en conception de systèmes de commande pourront proposer des modèles dynamiques satisfaisant ces spécifications, tandis que la vérification pourra se faire en exploitant les contraintes SysML comme les propriétés à vérifier dans *model-checker* ou sous la forme de pré et post conditions à préserver au cours d'un scénario de test. Afin de démontrer la faisabilité de l'approche, les transformations de modèles ont été simplement implémentées à l'aide de procédures d'import/export basées sur la technologie XML et XSLT ainsi que sur un méta-modèle des concepts partagés dont la Figure 3 présente un extrait.

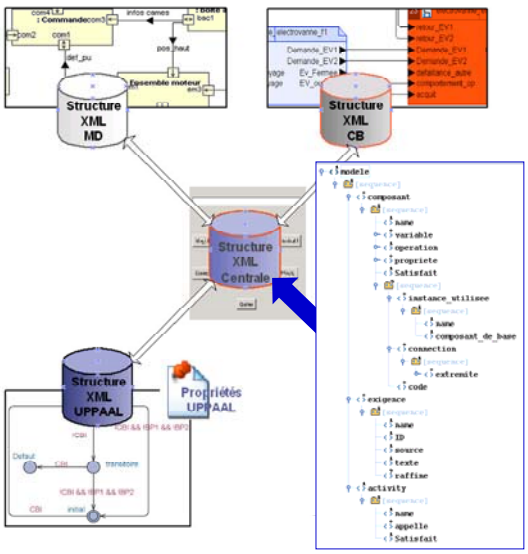


Figure 7 : Implantation de la démarche

Les outils connectés sont Magic Draw et Artisan Studio pour la modélisation système en SysML, ControlBuild pour la conception et la simulation de la commande et UPPAAL[7] pour la vérification de propriétés.

5 APPLICATION A L'ETUDE DE CAS INRS

5.1 ÉTUDE DE CAS : PRESSE MECANIQUE

Nous considérons une presse mécanique (Figure 8), simplifiée par rapport à la machine réelle, qui permet de poinçonner des produits en métal avec un outil en mouvement vertical. Un vilebrequin équipé de deux capteurs « point mort » haut et bas génère ce mouvement vertical. Un dispositif d'embrayage, qui est actionné par une vanne pneumatique, assure la transmission entre un moteur et le vilebrequin.

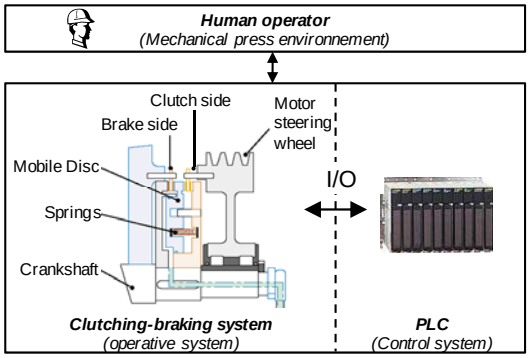


Figure 8 : Cas d'étude de la presse mécanique

Nous nous concentrons sur la fonction de commande de l'embrayage et du moteur en mode de fonctionnement coup par coup. Dans ce mode, la presse est initialement arrêtée en position haute et attend une demande de l'opérateur pour démarrer l'emboutissage. L'outil descend alors jusqu'au point mort bas puis remonte et s'arrête à sa position initiale haute.

Dans ce contexte, nous nous focalisons sur deux fonctions de sécurité :

- la commande du dispositif de commande bimanuelle qui impose à l'opérateur d'avoir les deux mains posées sur l'interface pour pouvoir déclencher une demande d'emboutissage. Si l'opérateur retire la pression d'une main sur le dispositif pendant la phase de descente, la presse s'arrête immédiatement.
- un dispositif d'arrêt d'urgence provoque l'arrêt du mouvement.

Habituellement, la commande de ces dispositifs est réalisée en technologie câblée. En raison de contraintes économiques, la commande des machines industrielles assure de plus en plus de fonctions de sécurité à l'aide de technologies programmées comme les Automates Programmable Industriel dédiés à la Sécurité (APIdS). L'intégration des fonctions de sécurité dans des systèmes programmés ne peut bien sûr être envisagée que si le niveau de sécurité de la technologie programmable est au moins égal à celui de la technologie filaire. Un des rôles de l'INRS est d'acquérir de nouvelles connaissances comme de nouveaux modèles, méthodes et outils pour le développement de machines sûres afin de déterminer sous quelles conditions elles peuvent être transférées aux personnes en charge de la prévention. En particulier, la norme IEC 61508 recommande, pour les niveaux les plus élevés de SIL, l'utilisation de méthodes formelles pour contrôler la qualité des applications de commande à base de logiciel.

En ce sens, l'INRS a mené une étude sur l'utilisation de la méthode B [19-20] dans le cas de la presse mécanique. Cette étude a clairement mis en évidence l'efficacité de la méthode pour la génération de code prouvé. En revanche, la méthode de modélisation formelle n'est pas explicite, notamment en ce qui concerne les choix de structuration et d'interprétation du cahier des charges, et s'appuie sur la pratique des experts pour générer des modèles abstraits et formels dès les premières phases du projet. Cette nécessité d'une expertise implicite peut être vue comme un frein vers la traçabilité des exigences qui requiert au contraire que tous les choix soient justifiés et documentés. D'autre part, commencer l'étude d'un système complexe en utilisant d'emblée des modèles formels semble être difficilement réalisable dans la pratique. Tous ces constats justifient l'utilisation d'un modèle moins formel tel que SysML [21] pour couvrir les premières phases de spécification à l'aide d'un langage graphique qui favorise le partage entre les diverses parties prenantes.

5.2 MODELISATION DES EXIGENCES DE SECURITE

Les exigences fonctionnelles de la presse concernent sa mission, ses performances, son coût, son intégration dans une chaîne de production. Un premier modèle, qui nécessite bien sûr d'autres décompositions, est présenté en Figure 9. Cet article portant sur la modélisation des exigences de sécurité, nous ne détaillerons pas d'avantage ces exigences.

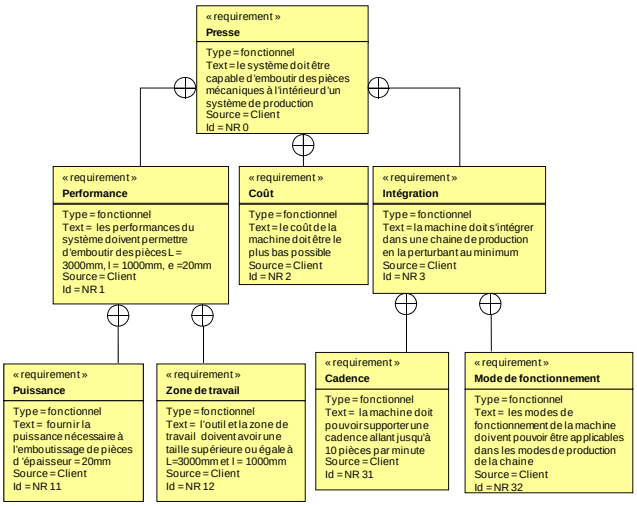


Figure 9 : Exigences fonctionnelles

Les exigences de sécurité sont déduites d'un processus d'analyse de risque, conforme aux recommandations de l'ISO 12100-1 et 13849-1, et comprenant :

- une identification des dangers et une évaluation prévisionnelle des risques basées sur des méthodes telles que les arbres de défaillances ou l'AMDEC (Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité).
- une réduction des risques reposant sur la mise en œuvre de dispositifs de prévention, visant à minimiser l'occurrence d'événements redoutés, ou de protection visant à minimiser leurs conséquences.

Concernant la presse mécanique étudiée, l'analyse de la sécurité identifie deux risques principaux (Figure 10) :

- écrasement des mains de l'opérateur durant le mouvement d'emboutissage.
- éjection d'une pièce durant l'emboutissage.

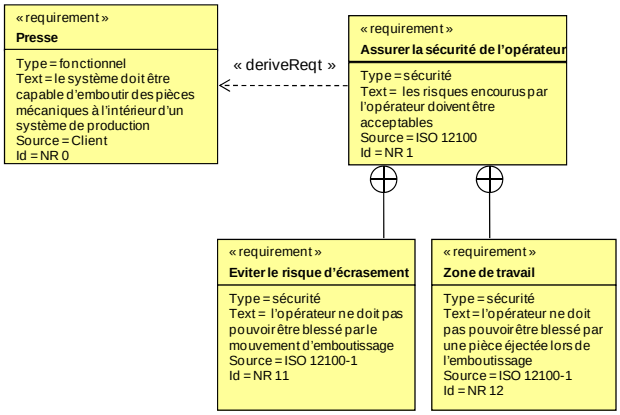


Figure 10 : Exigences de sécurité (1)

Dans la suite de l'article, nous focalisons sur le risque d'écrasement. Deux types d'accès à la zone dangereuse sont identifiés : un accès par l'avant de la machine et un accès latéral. En raison de la position du vilebrequin dans l'architecture de la presse, les accès latéraux sont potentiellement dommageables non seulement en phase de descente de l'outil (emboutissage) mais également dans la phase de remontée en position haute de la presse. Il faut éviter les accès latéraux quel que soit le mouvement de la presse (exigence SR3) et l'accès par l'avant uniquement en phase de descente (exigence SR1).

Conformément à la norme ISO 12100, les dispositifs choisis pour se prémunir de ces risques sont, d'une part, des protecteurs amovibles pour l'accès latéral (SR3) et, d'autre part, une commande bimanuelle (CB) pour l'accès avant (SR1). Le rôle de la commande bimanuelle est d'imposer à l'opérateur d'appliquer ses deux mains sur le dispositif durant la totalité du mouvement de descente. Tous les dispositifs de protection sont interverrouillés et conditionnés par le mouvement de la presse : si une protection n'est pas activée pendant le mouvement d'emboutissage, le mouvement est arrêté (exigence SR2).

Ces exigences sont modélisées dans le diagramme de la Figure 11.

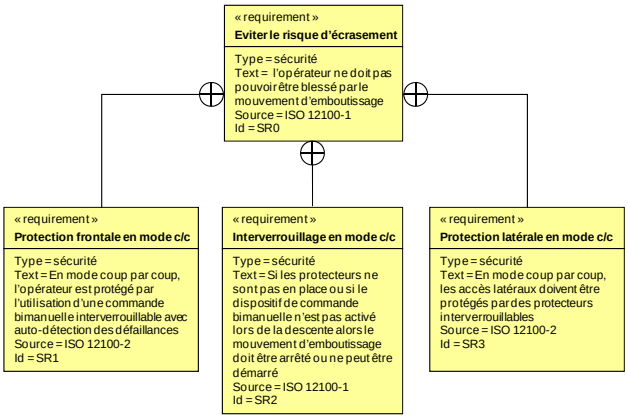


Figure 11 : Exigences de sécurité (2)

Considérons l'exigence de sécurité relative à la commande bimanuelle. Cette exigence SR1 peut être décomposée selon les normes ISO 13849-1, EN574 et ISO12100-2 en cinq sous-exigences relatives à :

- l'ergonomie du dispositif, notamment pour empêcher que la commande bimanuelle puisse être activée avec une seule main grâce à un capot de séparation (exigence SR1.2),
- auto-détection de défaillance (l'exigence SR1.4 prescrit que la commande bimanuelle doit être désactivée en cas de détection de défaillance),
- activation, désactivation et réactivation de la commande bimanuelle (l'exigence SR1.1 prescrit que la commande bimanuelle (CB) ne doit être réactivée que si les deux mains ont été retirées, SR1.3. prescrit que la CB est activée si les deux mains exercent une pression sur le dispositif dans un intervalle de temps inférieur à 0,5s et SR1.5 signifie que la CB est désactivée si une des deux mains est retirée.

Ces nouvelles exigences sont modélisées par le diagramme SysML de la Figure 12 : Exigences de sécurité (3). Dans ce diagramme, figurent également les contraintes SysML permettant de raffiner les exigences sous la forme de propriétés exprimées dans une logique à prédicat.

Les liens entre les variables présentes dans les contraintes seront ultérieurement associés aux variables d'entrée et de sortie des composants, selon la méthode proposée au paragraphe 4.3.

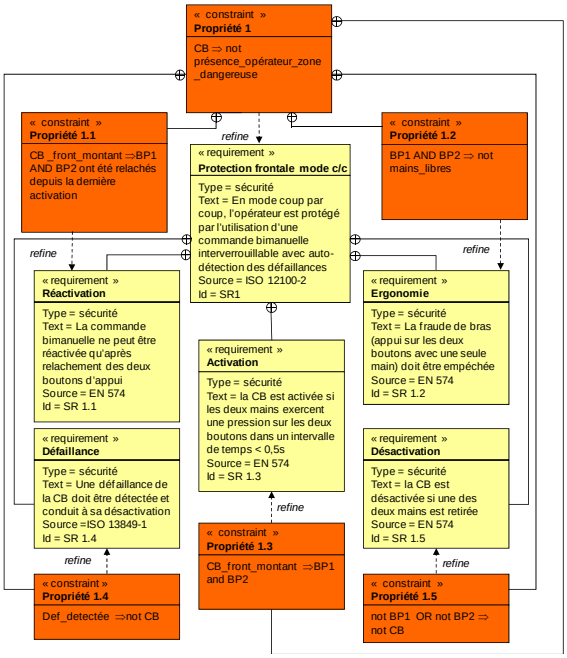


Figure 12 : Exigences de sécurité (3)

5.3 Definition des architectures

La description des architectures fonctionnelles et structurelles de la presse est réalisée en utilisant respectivement les diagrammes d'activité et les diagrammes de définition de bloc.

Les diagrammes d'activité définissent les principales fonctions de la presse, et en particulier les fonctions de sécurité, ainsi que les flux d'informations ou de matière qu'elles échangent. À titre d'exemple, la Figure 13 présente deux fonctions de sécurité associée à la gestion de la commande bimanuelle (activation /désactivation, détection de défaillance).

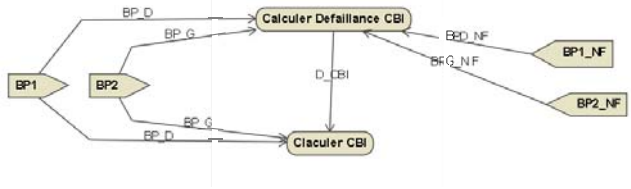


Figure 13 : Architecture fonctionnelle de la commande bimanuelle (extrait)

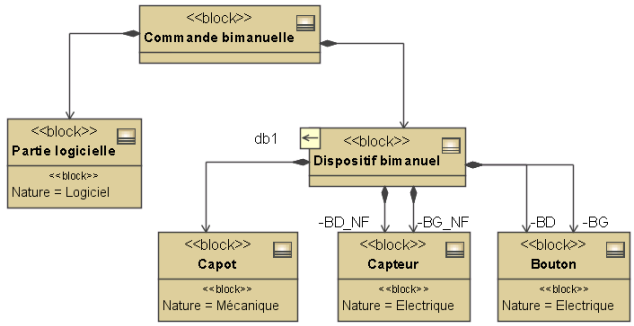


Figure 14 : Achitecture structurelle de la commande bimanuelle (extrait)

Les diagrammes de définition de blocs (BDD) décrivent l'architecture de la presse en termes de composants mécaniques, électriques, logiciels. À titre d'exemple, la Figure 14 présente l'architecture de la commande bimanuelle et l'ensemble de ses composants.

Enfin, les BDD peuvent être complétés par des IBD (Internal Block Diagram) permettant de représenter l'architecture interne des composants et, en particulier, un réseau interconnecté de composants logiciels de sécurité tels que celui de la Figure 15 relatif aux composants logiciels de la commande bimanuelle.

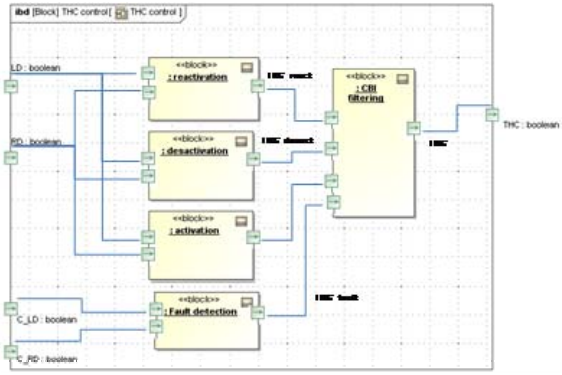
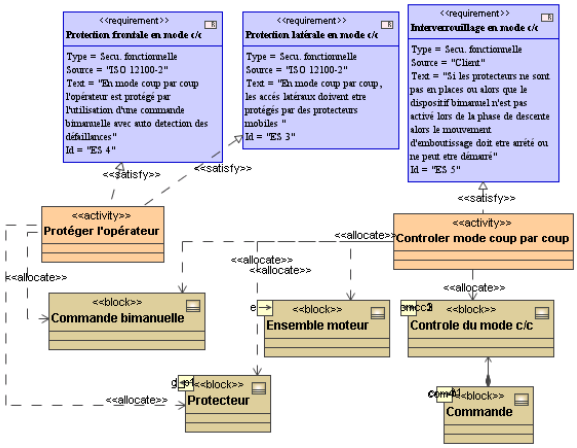


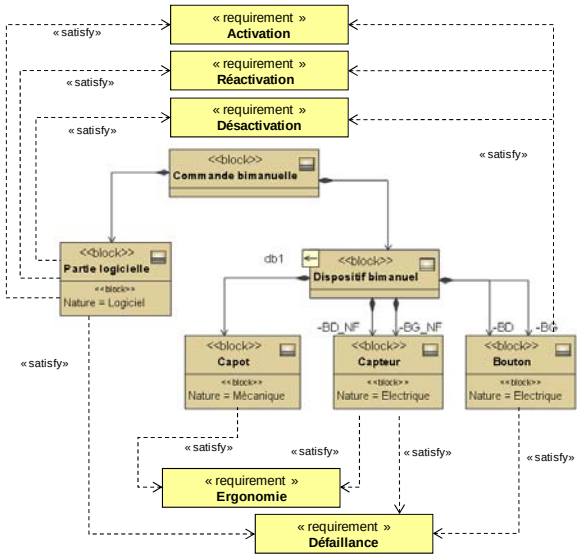
Figure 15 : Diagramme Interne de Bloc pour la commande bimanuelle (extrait)

5.4 Traçabilite des exigences de securite

L'utilisation des mécanismes *satisfy* et *allocate* permettent respectivement d'associer les fonctions aux exigences de sécurité et d'allouer ces mêmes fonctions sur les composants de la presse. La Figure 16a présente un extrait de ces relations exigences/fonctions/composants pour les exigences générales de sécurité de la presse ; la Figure 16b présente un extrait relatif aux exigences de sécurité de la commande bimanuelle.



a) Exigences de sécurité de la presse (extrait)



b) Exigences de sécurité de la commande bimanuelle

Figure 16 : Traçabilité des exigences de sécurité

Il est également possible de représenter ces relations entre exigences, fonctions et composants sous la forme d'une matrice de traçabilité (tableau 1).

Exigences	Composants
SR 1	Tous
SR 1.1	Logiciels et périphériques
SR 1.2	Capot et capteur
SR 1.3	Logiciels et périphériques
SR 1.4	Logiciels et périphériques
SR 1.5	Logiciels et périphériques

Tableau 1 : Matrice de traçabilité des exigences de sécurité pour la commande bimanuelle

Enfin, chaque exigence étant associée à une ou plusieurs propriétés exprimées sous la forme de contraintes SysML (Figure 12), les composants sont donc associés à un ensemble de propriétés qu'ils doivent respecter.

5.5 VERIFICATION DES EXIGENCES DE SECURITE

L'ensemble de ces spécifications SysML fournissent :

- les composants de la presse, et en particulier les composants logiciels de la commande bimanuelle (diagrammes BDD et IBD),
- les fonctions de sécurité que chacun d'entre eux doit assurer (relations *allocate*),
- les propriétés que chacun d'entre eux doit respecter (relations *satisfy* et *contraintes*).

À partir de ces informations, il est alors possible de passer à la réalisation de chacun des composants. Selon la nature de ces derniers, des modèles et outils spécifiques seront utilisés (DSL). En ce qui concerne la réalisation des composants logiques assurant les fonctions de sécurité de la commande bimanuelle, notre choix s'est orienté vers les modèles flots de données de Control-Build dont l'importation depuis des IBD est facilitée par leur similitude avec les IBD.

La vérification des composants réalisés est effectuée à l'aide du *model-checker* UPPAAL. Les propriétés des composants sont déduites de l'étude SysML modulo une transformation des prédicats contenus dans les contraintes sous la forme :

- d'une propriété CTL* dans le cas de propriétés logiques ne nécessitant pas de mémorisation d'état,
- d'un observateur mémorisant les évolutions d'état du système et représentant une séquence interdite sous la forme d'un état bloquant.

À titre d'exemple, la propriété P1.5 a été traduite dans UPPAAL sous la forme « il est toujours vrai que… » et l'opérateur de l'AG :

AG (NOT(BP1) OR NOT(BP2) ⇒ NOT(CB))

Considérons maintenant la propriété P1.1 qui stipule que la commande bimanuelle ne peut être réactivée que si les deux boutons ont préalablement été relâchés. Cette propriété est représentée par un observateur qui comprend quatre états : CB activée, CB désactivée avec les deux boutons libérés, CB désactivée avec un seul des deux boutons libéré et enfin un état erreur correspond à la séquence interdite (). Il convient alors de vérifier à l'aide du *model-checker* que l'état erreur ne peut jamais être atteint à l'aide de la propriété AG (not erreur).

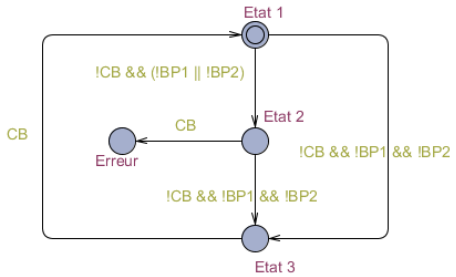


Figure 17 : Observateur sur UPPAAL (P1.1)

6 CONCLUSION

Dans le domaine des applications critiques à forte composante logicielle, l'intérêt croissant pour les méthodes formelles, soutenu par la norme IEC 61508 et toutes ses déclinaisons sectorielles, peut se justifier par leurs capacités à apporter la preuve que des modèles de système satisfont bien un ensemble donné de propriétés. Si l'on considère que l'écriture de ces propriétés, en lien avec les exigences exprimées par les parties prenantes, n'est pas une tâche aisée, et si l'on considère, en outre, qu'il est difficile

d'appréhender un système complexe et son fonctionnement uniquement à l'aide de méthodes formelles, l'utilisation de modèles qualitatifs utilisés en ingénierie système tel que SysML s'avère un préalable indispensable.

Notre proposition méthodologique a pour objectif de démontrer la faisabilité d'une approche intégrant ces deux points de vue dans le domaine de systèmes de sécurité. Elle contribue à rapprocher la modélisation système souvent non formelle et les langages formels spécifiques à un métier donné. Plus particulièrement, elle repose sur l'utilisation de SysML pour structurer les exigences de sécurité et assurer leur traçabilité lors de la définition d'architecture fonctionnelle et structurelle, et sur des modèles et outils spécifiques pour la conception et la vérification de composants logiciels de sécurité.

Cette étude de faisabilité doit être maintenant complétée par :

- une formalisation des mécanismes de composition des exigences de sécurité, notamment lorsque leur description nécessite l'utilisation d'opérateurs temporels ;
- une formalisation, par exemple à l'aide de langages tels que ATL, des transformations de modèles mises en œuvre entre diagrammes SysML et modèles de conception et de vérification, en lieu et place des procédures d'import/export définies actuellement entre outils.

Enfin, bien que la collaboration entre outils du génie informatique et outils de l'ingénierie système ait montré un intérêt pour la maîtrise de la sécurité fonctionnelle, l'effort doit être poursuivi pour passer à l'échelle sur des systèmes de taille industrielle. En ce sens, cette approche est actuellement expérimentée dans le cadre du développement d'une commande à base de COTS dans le domaine du ferroviaire. Plus particulièrement, l'application étudiée concerne la gestion des accès voyageur qui présente des risques pour la sécurité (ouverture intempestive, enfermement, etc). Dans ce contexte, où la notion de COTS induit la notion de sous-traitant, la spécification SysML des exigences et leur allocation sur une architecture de fonctions et de composants s'avère très précieuse.

7 REFERENCES

[1] A. Fusaoka, H. Seki et K. Takahashi : 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, Allemagne.

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

[3] S. Sheard : Complex Systems Science and its Effects on Systems Engineering ; *European Systems Engineering Conference*, 18-20 septembre 2006, Edimbourg, Grande-Bretagne.

[4] B. Willard : UML for Systems Engineering ; Computer Standards and Interfaces, Vol 29, pp 69-81, 2007

[5] W. Heaven et A. Finkelstein : A UML Profile to Support Requirements Engineering with KAOS ; *IEE Proceedings - Software*, Vol. 151, pp. 10-27, 2004

[6] F. Auinger, R. Brennan, J. Christensen, L. M. Lastra et V. Vyatkin : Requirements and solutions to software encapsulation and engineering in next generation manufacturing systems ; OOONEIDA approach ; *International Journal of Computer Integrated Manufacturing*, Vol. 18(7), pp. 572-585, 2005.

[7] L. Ferrarini, C. Veber, C. Schwab, M. Tangermann et A. Prayati : Control functions development for distributed automation systems using the Torero approach ; *16th IFAC World Congress*, 4-8 juillet 2005, Prague, République Tchèque.

[8] C. Tranoris et K. Thramboulidis : A tool-supported engineering process for developing control applications ; *Computers in Industry*, Vol. 57(5), juin 2006, pages 462-472.

[9] U. Katzke et B. Vogel-Heuse : UML-PA as an engineering model for distributed process automation ; *Proceedings of the 16th IFAC world Conference*, 2005. Prague.

[10] J.-R. Abrial : *The B-book : Assigning programs to meanings ;* Cambridge University Press, Grande-Bretagne, 1996.

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

[12] E. M. Clarke, O. Grumberg et D. A. Peled : *Model-checking* ; Cambridge, MA: MIT Press, 2000.

[13] C. G. Cassandras et S. Lafortune : *Introduction to Discrete Event Systems* ; Kluwer Academic Publisher, ISBN 0-7923-8609-4, 1999.

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

[15] R. Achatz : Requirements engineering: A key success factor : *Automation Technology in Practice*, Issue 3, novembre 2006.

[-16] T. L. Johnson : Improving automation software dependability: a role for formal methods? ; *Control Engineering Practice*, Volume 15, Issue 11, novembre 2007, pp. 1403-1415.

[17] P. Roques : From SADT to SysML: formulation of an embedding ; *Proceedings of the 22nd Int. Conference on Software & Systems Engineering and their Applications*, ICSSEA'10, Paris, 7-9 décembre 2010.

[18] H. Panetto, P. Lhoste, G. Morel et M. Roesch : SPEX : du génie logiciel pour le génie automatique ; *Génie Logiciel et Systèmes Experts*, n° 25, pages 22-28, 1991

[19] J.-R. Abrial : Case study of a complete reactive system in Event-B: a mechanical press controller ; *tutorial 3 of ZB'2005*, Guildford, Grande-Bretagne, 13-15 avril 2005

[-20] D. Evrot, J.-F. Pétin et D. Mery : Formal specification of safe manufacturing machines using the B method: application to a mechanical press ; *Proceedings of the 12th IFAC INCOM*, vol. 1, pp 277-282, 17-19 mai 2005, Saint-Étienne.

[21] D. Evrot, J.-F. Pétin, G. Morel et P. Lamy : Using SysML for identification and refinement of machinery safety properties ; *Proceedings of IFAC Workshop on Dependable Control of Discrete Systems*, juin 2007, Cachan, France.

NOTES

1 Doors est un produit d'IBM Rational (à l'origine, Telelogic), *ibm.com/software/rational*

2 Reqtify est un produit de Dassault Systèmes, *www.geensoft.com*

3 INCOSE : International Council on System Engineering dont l'AFIS (Association Française d'Ingénierie Système) est le chapitre français

4 Simulink est un produit MathWorks

5 Scade est un produit d'Esterel Technologies ; voir Génie Logiciel, n° 92, pp. 7-15, mars 2010

6 ControlBuild est un produit GeenSoft / Dassault Systèmes

7 UPPAAL est un outil développé par les universités d'Uppsala (Suède) et d'Aalborg (Danemark).