Modélisation UML/B pour la validation des exigences de sécurité des règles d'exploitation ferroviaires

Rahma Yangui

N^o d'ordre : 291

Centrale Lille

THÈSE

présentée en vue d'obtenir le grade de

DOCTEUR

en

Spécialité : Automatique, Génie Informatique, Traitement du Signal et Images

par

Rahma YANGUI ép BEN AYED

DOCTORAT DELIVRÉ PAR CENTRALE LILLE

Titre de la thèse :

Modélisation UML/B pour la validation des exigences de sécurité des règles d'exploitation ferroviaires

Soutenue le 19 février 2016 devant le jury d'examen :

Président	Pr, Yves LEDRU	LIG, Grenoble
Rapporteur	Pr, Catherine DUBOIS	ENSIIE, Evry
Rapporteur	MCF-HDR, Mohamed SALLAK	UTC, Compiègne
Membre	Directeur R&D, Thierry LECOMTE	Clearsy, Aix en Provence
Membre	Dr, Agnès LANUSSE	CEA-List, Saclay
Membre	Pr, Pascal YIM	ECL, Lille
Directeur de thèse	DR, Simon COLLART-DUTILLEUL	IFSTTAR, Lille
Encadrant	CR, Philippe BON	IFSTTAR, Lille

Thèse préparée dans le Laboratoire d'Évaluation des Systèmes de Transports Automatisés et de leur Sécurité
IFSTTAR, COSYS/ESTAS, Villeneuve d'Ascq

École Doctorale SPI 072 (EC Lille)

Remerciements

Les travaux de thèse présentés dans ce manuscrit ont été réalisés à l'institut français des sciences et technologies des transports, de l'aménagement et des réseaux (IFSTTAR) à Villeneuve d'Ascq. Je tiens tout d'abord à remercier IFSTTAR de m'avoir offert cette expérience magnifique. La recherche scientifique est un métier magique et c'est un bonheur de l'avoir commencé ici.

C'est avec toute ma profonde gratitude que j'exprime mes sincères remerciements à mon directeur de thèse *Simon Collart-Dutilleul* d'avoir diriger ce travail de thèse. Sans sa disponibilité malgré sa charge de travail, sa patience, les conseils qu'il m'a prodigués tout au long de ces trois ans, ses mots d'encouragement, ce travail n'aurait pas pu, sans doute, être mené à son terme. Merci.

J'exprime ma profonde reconnaissance et remerciement à mon encadrant *Philippe Bon* d'avoir encadré ce travail de thèse, pour son soutien, ses conseils et sa participation dans l'accomplissement de ce travail de thèse. Merci.

Je remercie chaleureusement Messieurs *Yves Ledru* et *Akram Idani*, qui ont suivi de près ce travail, sans qui cette partie n'existerait pas sous cette forme. Leurs observations judicieuses et leurs remarques ont été des plus déterminantes dans la conduite de ces travaux. Merci.

Mes vifs remerciements vont aussi à Madame *Catherine Dubois*, Professeur de l'École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise à Evry et Monsieur *Mohamed Sallak*, Maître de conférence HDR de l'université de Technologie de Compiègne, pour avoir accepté d'examiner ce travail en qualité de rapporteurs. Je tiens également à exprimer ma gratitude à Messieurs *Yves Ledru*, Professeur de l'université Joseph Fourier à Grenoble, et *Pascal Yim*, Professeur de l'École centrale de Lille, et Madame *Agnès Lanusse*, Ingénieur-Chercheur du CEA-List à Saclay, pour avoir accepté de faire partie du jury en qualité d'examinateurs, ainsi que Monsieur *Thierry Lecomte* en qualité d'invité.

Mes remerciements vont également à tous les personnels du centre IFSTTAR-Villeneuve d'Ascq pour la gentillesse et la convivialité dont ils ont fait preuve et qui ont rendu mon séjour très agréable parmi eux.

Je remercie chaleureusement mes amies *Hajer Sassi*, *Imen Chakroun*, *Sana Cherif*, *Hana Krichen*, *Chiraz Trabelsi*, *Karima Boufaras*, *Amira Bradai*, *Mouna Walha*, *Fatiha Azmani*, *Manel Hmimida*, *Amira Arous* pour leurs soutiens et leurs encouragements.

J'ai une pensée toute particulière à ma famille pour leur soutien et leur amour. Tout ça n'aurait jamais été possible sans le soutien inconditionnel de mes parents *Sondess* et *Anouar*, qui ont toujours cru en moi. Merci pour avoir fait de moi ce que je suis à présent. Merci à mes chères sœurs *Ameni* et *Nour* pour leurs conseils et leurs encouragements dans les moments difficiles.

Cette étape de ma vie n'aurait pas été la même sans la présence et l'amour de mon cher mari *Mohamed*. *Mohamed*, je te remercie de tout mon cœur d'être patient et souriant, de me soutenir et de m'encourager à aller plus loin et à franchir les difficultés tout en étant fier de moi. Je remercie également mon cher fils *Rayen* qui apporte toujours la joie de vivre au sein de ma petite famille. Par cette occasion, je remercie ma belle famille : mes beaux parents *Sarra* et *Abdelmajid* et ma belle sœur *Nadia* pour leur soutien et leurs encouragements.

Je dédie tout ce travail à mon cher fils *Rayen* et à ma futur fille ..., les plus chers à mon cœur.

En dernier, je remercie tous ceux qui m'ont aidé de près ou de loin à réaliser cette thèse.

Table des matières

Table des figures

Liste des tableaux

Glossaire

AFIS Association Française d'Ingénierie Système. 21

B La méthode B. i, vi–x, 4, 14, 15, 44–74, 77, 78, 85–87, 90, 94–98, 101–104, 112, 117, 127, 129–132, 160

CIM Computation Independent Model. 38, 39, 87

DAC Discretionary Access Control. 25

ERTMS European Rail Trafic Management System. i, v, ix–xi, 3, 5–8, 13, 15, 40, 42, 43, 71–73, 78, 79, 89, 101, 102, 105, 108–112, 115, 116, 119–122, 127, 129, 130, 132, 160

ETCS European Train Control System. 5, 7, 8, 10, 15, 40, 42, 78–81, 83, 89–91, 93, 98, 99, 101, 102, 105, 108, 109, 115, 129, 157, 158

FS Full Supervision. Mode technique dans lequel le train est supervisé en vitesse et déplacement. Marche normale attribuée au train. 82, 89, 90, 158

GSM-R Global System for Mobile communications - Railways. 5, 6, 82, 157, 158

IDM Ingénierie dirigée par les modèles. v, vi, ix, 19, 20, 27–30, 32, 34, 35, 37, 38, 42–44, 55, 59, 86, 87

MA Movement Authority : scénario nominal d'autorisation de mouvement. vii, ix–xi, 15, 77, 80–83, 89–99, 109, 129, 157, 158

MAC Mandatory Access Control. 26

MDA Model Driven Architecture. v, ix, 19, 29–32, 34, 35, 37–39, 56

Or-Bac Organization Based Access Control. i, vii, viii, x, 13–15, 26, 36, 103–110, 112–117, 125, 127, 130, 131

OS On Sight. (À vue). Mode qui autorise le conducteur à s'avancer en marche à vue. 82, 90, 157

Override EOA Scénario exceptionnel de franchissement d'un arrêt ETCS ou un EOA. vii–ix, xi, 15, 77, 78, 81, 83, 90–92, 98, 99, 103, 107–110, 129

PERFECT Performing Enhanced Rail Formal Engineering Constraints Traceability : Vers la formalisation des exigences ferroviaires et leur traçabilité. v, vii, 3, 5, 8, 9, 14, 22, 45, 46, 58, 71, 129

PIM Platform Independent Model. 31–33, 38, 44, 87, 112

PSM Platform Specific Model. 31–33, 38, 39, 87

RBAC Role Based Access Control. i, vii–x, 13–15, 26, 36, 77, 79, 83–88, 92, 93, 100–127, 130, 151–155, 160

SR Staff Responsible. Mode de déplacement utilisé dans les situations dégradées. Il est utilisable sous la responsabilité respective de l'agent-circulation et du conducteur. 80, 81, 83, 90

SysML System Modeling Language. 42, 71, 72

UML Unified Modeling Language. i, vi–ix, 4, 12, 14, 15, 30, 34, 36, 38, 42–74, 78, 85, 86, 92–96, 98, 101–104, 112, 114, 127, 129–131, 160

V&V Vérification et validation. vi, 11, 12, 38, 39, 41, 42, 50, 51, 55, 56, 74

XMI XML Metadata Interchange. 30

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte scientifique

La sécurité, l'un des attributs de sûreté de fonctionnement, est un enjeu majeur des systèmes critiques étant données la complexité et les conséquences graves pouvant découler d'erreur de conception. Afin de limiter ces erreurs, les architectures logicielles de ces systèmes sont soumises à un processus de développement ayant recours à des techniques d'ingénierie adaptées pour la spécification et la validation. Dans l'essor du génie logiciel, ces techniques d'ingénierie sont vues comme des méthodologies à forte valeur ajoutée sur tout ou partie du processus de développement. En effet, l'ingénierie des exigences, l'une des disciplines de l'ingénierie système, permet une meilleure conduite du développement des exigences, notamment des exigences de sécurité. Une autre forme d'ingénierie dirigée par les modèles apporte des solutions intéressantes pour automatiser le processus de développement en se basant sur les concepts de modélisation et de transformation de modèles.

Dans le processus de développement des systèmes critiques, nous partons du constat que les activités de spécification, de vérification et de validation constituent des tâches d'envergure nécessitant des mécanismes rigoureux. Ce faisant, un éventail de méthodes formelles existe dans l'optique de mener rigoureusement ces activités. Fondées sur des bases mathématiques, ces méthodes peuvent aider à maîtriser la complexité et réduire l'ambiguïté des spécificités des systèmes critiques, dès lors qu'elles permettent la spécification et le développement de systèmes dans un processus de développement formel, ainsi que la vérification des exigences de sécurité. En effet, les méthodes formelles ont montré des résultats probants et font preuve d'une efficacité incontestable depuis des dizaines d'années.

Toutefois, ces mécanismes présentent des difficultés considérables non seulement pour les non-experts en raison de leur complexité mais aussi pour les experts lors d'une formalisation directe des spécifications informelles. Cette difficulté scientifique s'atténue en alliant des notations formelles avec des notations semi-formelles. Ces dernières permettent de bénéficier d'un support de communication compréhensible aussi bien par les experts du domaine que par les experts des méthodes formelles. La problématique de couplage semi-formel/formel existe depuis des années. Cependant, notre choix des notations UML/B est motivé par des critères liés notamment à l'aptitude de ces deux notations à se coupler, au support d'outils disponibles ainsi qu'au domaine d'application. Le domaine d'application nous incite à aborder le contexte industriel au sein duquel sont menés nos travaux de recherche et sont appliquées nos contributions.

1.2 Contexte industriel

Les travaux présentés dans ce mémoire se placent dans le contexte de l'industrie ferroviaire. La problématique émane de l'augmentation de la criticité de ces systèmes au regard des exigences de sécurité, une des propriétés à garantir de la sûreté de fonctionnement. Dans le cadre de cette thèse, ces exigences de sécurité constituent une préoccupation centrale dans le processus de développement.

Ce travail a été réalisé dans le cadre du projet *PERFECT* (2012-2016) de l'Agence Nationale de Recherche (ANR-TDM), abréviation de « Performing Enhanced Rail Formal Engineering Constraints Traceability » : Vers la formalisation des exigences ferroviaires et leur traçabilité. Ce projet aborde la problématique d'une solution ERTMS de sécurité sur le sol français. Les partenaires du projet sont le CEA, l'École centrale de Lille, l'UTC/Heudiasyc, la PME ERSA et l'industriel Ansaldo-STS ainsi que l'IFSTTAR qui est le coordinateur du projet.

Dans cette section, nous nous intéressons au contexte industriel de nos travaux de recherche. Nous commençons par introduire le système ERTMS et ensuite nous poursuivons par la présentation du projet *PERFECT* dont nos travaux font partie.

1.2.1 Le système ERTMS

Le système ERTMS (European Rail Trafic Management System) [1] est une initiative de la commission européenne lancée au début des années 1990 et implémenté par huit membres de l'UNIFE [2] en Europe. Il est créé suite à des projets antérieurs pilotes internationaux qui visent à créer une standardisation de la signalisation en cabine permettant le passage des frontières nationales sans la nécessité d'une multiplicité des systèmes nationaux implantés à bord [Booth, 2006]. L'objectif majeur de ce projet est ainsi d'harmoniser la signalisation ferroviaire en Europe tout en garantissant la sécurité des circulations.

Le système ERTMS est composé du système européen de contrôle des trains (European Train Control System, ETCS) qui est le système de contrôle/commande, ainsi que du système de communication radio GSM-R (Global System for Mobile communications - Railways) pour la transmission de données entre l'ETCS embarqué (le sous-système à bord du train) et l'ETCS sol (le sous-système au sol). Le sous-système à bord comprend essentiellement l'équipement à bord et éventuellement une partie du système radio GSM-R selon le niveau d'ETCS (décrit ci-dessous). L'équipement à bord est un système embarqué qui supervise le mouvement du train auquel il appartient sur la base des informations échangées avec le sous-système au sol. Ce dernier peut être composé des éléments suivants :

1. European Rail Traffic Management System : http ://www.ertms.net
2. European Rail Industry : http ://www.unife.org

— **Balise** qui est un dispositif permettant la transmission ponctuelle des télégrammes au sous-système à bord.
— **Lineside Electronic Unit (LEU)** qui est un dispositif électronique de commande des balises permettant de générer les télégrammes à envoyer par les balises basés sur les informations provenant des systèmes au sol externes.
— **Euroloop et Radio infill** qui sont des sous-systèmes permettant de fournir des informations de signalisation anticipées concernant le prochain signal principal dans le sens de circulation du train.
— **Radio Block Center (RBC)** qui est l'équipement sol permettant la génération des messages destinés aux trains et le traitement des messages provenant des trains transmis via la radio.
— **La partie sol de GSM-R** qui est utilisée pour l'échange bidirectionnel, sol-train et train-sol, des messages.

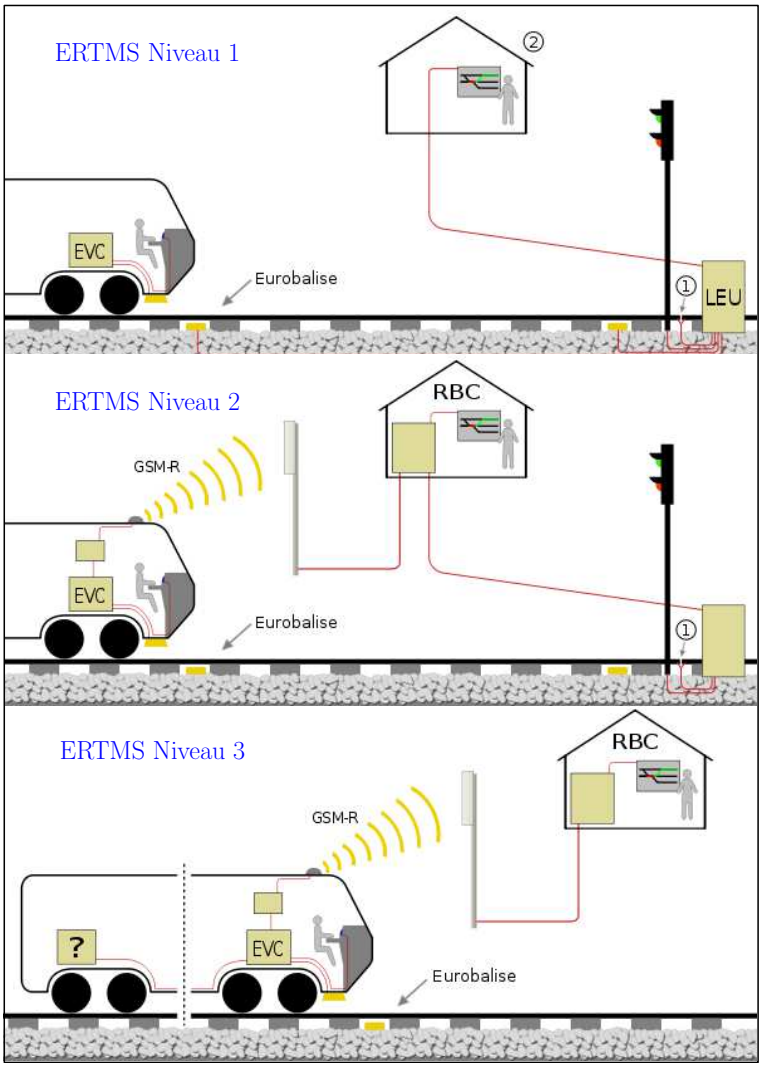

Figure 1.1 – Les différents niveaux d'ERTMS (voir note de bas de page 3)

Selon ces moyens de transmission et d'autres considérations, trois niveaux d'ERTMS/ETCS ont été définis [Schön *et al.*, 2014], ils sont schématisés sur la figure 1.1 [3] :

— **Niveau 1**, dans lequel la transmission des informations est réalisée uniquement par des balises ; dans ce niveau, la détection et la vérification de l'intégrité du train [4] sont réalisées par l'équipement au sol du système de signalisation.

— **Niveau 2**, dans lequel la transmission des informations est réalisée par la radio ; les balises sont utilisées pour assurer le recalage du train ; dans ce niveau, la détection du train et la vérification de l'intégrité du train sont aussi réalisées par l'équipement au sol du système de signalisation.

— **Niveau 3**, dans lequel la transmission est semblable au niveau 2 ; dans ce niveau, la détection et la vérification de l'intégrité du train sont réalisées par une information de localisation transmise par le train au sol, via la radio.

L'un des enjeux majeurs d'ERTMS/ETCS est l'interopérabilité qui vise à rendre opérationnels les différents systèmes techniques ferroviaires existant en Europe en facilitant les interconnexions entre les différents réseaux.

Pour ce faire, des spécifications techniques d'interopérabilité (STI) ont été définies. L'objectif de ces spécifications est l'intégration et l'harmonisation des normes techniques des systèmes ferroviaires facilitant la mobilité des personnes et des marchandises en favorisant l'utilisation du rail comme mode de transport en Europe. En particulier, des spécifications techniques d'interopérabilité pour l'exploitation et la gestion du trafic à grande vitesse ont été proposées par la directive 96/48/CE du Conseil relative à l'interopérabilité du système ferroviaire transeuropéen à grande vitesse. Cette directive a été révisée en 2004 et puis en 2007 par les directives 2004/50/CE et 2007/32/CE. Ces dernières modifient la directive 96/48/CE ainsi que la directive 2001/16/CE du Parlement européen et du Conseil relative à l'interopérabilité du système ferroviaire transeuropéen conventionnel [5].

Dans la directive 2001/16/CE, l'interopérabilité est définie comme ≪ *l'aptitude du système ferroviaire transeuropéen conventionnel à permettre la circulation sûre et sans rupture de trains en accomplissant les performances requises pour ces lignes. Cette aptitude repose sur l'ensemble des conditions réglementaires, techniques et opérationnelles qui doivent être remplies pour satisfaire aux exigences essentielles* ≫.

3. Cette figure est extraite de l'article wikipédia ≪ Système européen de contrôle des trains ≫ disponible sur l'adresse https ://fr.wikipedia.org/wiki/Syst%C3%A8me_europ%C3%A9en_de_contr%C3%B4le_des_trains

4. La vérification de l'intégrité du train consiste à contrôler que le train n'a pas perdu de wagon.

5. Les détails sur les différentes directives existent sur le site d'Eur-lex (Access to European Union law) http ://eur-lex.europa.eu/legal-content/FR/TXT/ ?uri=uriserv :l24229

Les différentes directives susmentionnées ont pour but de favoriser l'interconnexion et l'interopérabilité des réseaux nationaux de trains à grande vitesse ainsi que l'accès à ces réseaux. En effet, tous les États membres y sont invités à harmoniser leurs systèmes ferroviaires à grande vitesse pour permettre l'interopérabilité du réseau européen et ainsi renforcer l'utilisation du rail comme mode de transport en Europe. Cette tâche d'harmonisation et d'interopérabilité constitue une priorité pour l'Union Internationale des Chemins de fer (UIC). Le processus d'intégration européenne vise à établir des systèmes de transport performants opérant sans difficultés à l'échelle du continent tout entier. Le défi ambitieux du transport ferroviaire est alors de lutter contre les obstacles qui peuvent encore freiner son développement international. En effet, chaque pays possède des réglementations et des normes techniques nationales fruit d'évolutions techniques différentes dans le passé. Les différences entre ces réglementations et ces normes techniques nationales constituent un obstacle à la standardisation des matériels ferroviaires et une cause de ralentissement de la circulation des trains aux frontières. Des solutions sont alors recherchées, au niveau européen, afin de lever ces obstacles et de renforcer la position concurrentielle du rail [Union Internationale des Chemins de fer, 2000].

1.2.2 Le projet *PERFECT*

Comme évoqué ci-dessus, il existe une volonté européenne forte d'harmonisation du système ferroviaire européen via ERTMS. En revanche, la gestion de la signalisation dans le système ERTMS est régie par les règles nationales propres à chaque pays. L'analyse de la sécurité des systèmes doit donc pouvoir prendre en compte ces différents aspects et leur impact à plusieurs niveaux d'analyse. Le projet *PERFECT* contribue ainsi à l'évaluation du système en termes d'exigences de sécurité tout en assurant la cohérence et la consistance des règles d'exploitation au regard des règles nationales et européennes. En effet, les règles d'exploitation d'une ligne utilisant ERTMS/ETCS doivent simultanément respecter la spécification européenne, en faveur de l'interopérabilité, et les règles nationales.

Concrètement, ERTMS ne définit que l'environnement technique et opérationnel des matériels à bord des trains. Le comportement des matériels au sol est pour sa part essentiellement régi par les lois nationales. En effet, la partie bord du système de contrôle/commande des trains ETCS est interopérable. La partie sol peut être différente selon les pays, tout en gardant les mêmes objectifs de fonctionnalité [RFF, 2012]. Ce dernier aspect produit une certaine complexité dans l'analyse de la sécurité du système global. La proposition principale du projet *PERFECT* est d'étudier la consistance des différentes spécifications entre elles en utilisant des outils formels permettant l'évaluation du système en termes de sécurité. Cette problématique, centrale du projet, n'est abordée dans la littérature que de manière très parcellaire. La contribution des outils formels est ainsi située au niveau de la vérification de la consistance entre les différents documents au travers des règles d'exploi-

tation. Ces dernières sont définies afin de respecter les impératifs de sécurité et d'assurer la bonne gestion des circulations. Elles constituent l'interface de réglage entre le niveau européen et le niveau national des règlements (Figure 1.2).

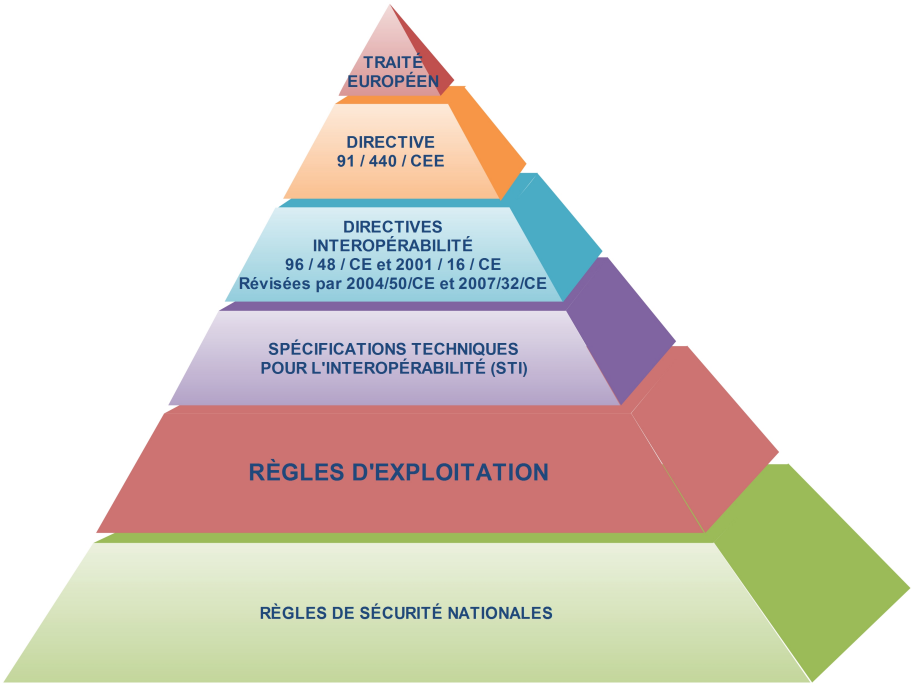

Figure 1.2 – Positionnement des règles d'exploitation dans les couches règlementaires [Collart-Dutilleul *et al.*, 2013]

Les travaux menés dans le cadre du projet *PERFECT* s'orientent vers des pistes méthodologiques offrant des solutions à différentes problématiques en liaison avec le système de contrôle/commande de la partie bord et la gestion de la signalisation de la partie sol. Ces pistes apportent des contributions à l'analyse de la réglementation et la modélisation de la gestion des enclenchements et de la gestion des itinéraires avec plusieurs technologies formelles, en vue de la vérification des propriétés de sécurité.

1.3 Problématique de la thèse

Dans le cadre de cette thèse, notre travail concerne l'analyse de la réglementation. La piste qui concerne la modélisation de la gestion des enclenchements et de la gestion des itinéraires a fait l'objet d'une autre thèse [Sun, 2015]. L'objectif principal étant l'analyse de la réglementation, nous mettons l'accent sur les deux principales problématiques suivantes :

— La spécification de la réglementation,
— La vérification et la validation des spécifications avec des méthodes formelles.

1.3.1 Spécification de la réglementation

La spécification est une phase du cycle de développement permettant de définir les exigences du système à partir d'un cahier des charges. Elle commence par l'analyse des besoins du cahier des charges dans le but de proposer une spécification fonctionnelle utile pour la conception globale et détaillée du système.

Dans ce contexte industriel, le cahier des charges est constitué des documents de spécifications européennes [UNISIG, 2006] et des règles d'exploitation ferroviaires sur une ligne nationale [RFF, 2012] :

- [UNISIG, 2006] est un document qui précise le futur standard unifié d'ETCS d'un point de vue technique. Cette spécification offre souvent des solutions multiples sur la façon de mettre en œuvre une fonction spécifique.
- [RFF, 2012] est un document créé dans le but d'une mise en service du système ETCS de niveau 2 sur la ligne à grande vitesse Est Européenne. Il reprend les principes et les dispositions relatifs à l'utilisation de ce système issus de la spécification technique d'interopérabilité (STI) et apporte des précisions utiles pour leurs mises en œuvre sur cette ligne.

Face à ces deux documents, la phase de spécification doit comprendre :

1. L'analyse des spécifications européennes face aux règles ferroviaires nationales via les règles d'exploitation,
2. La formalisation et la modélisation des scénarios issus des règles d'exploitation ferroviaires.

Les exigences accrues en sûreté de fonctionnement d'une façon générale ou en sécurité d'une façon particulière rendent incontournable l'utilisation des méthodes formelles. Nous devons alors tenir compte des exigences de sécurité dans le processus de développement et l'intégrer dans les phases de spécification et de conception du système. L'utilisation des techniques formelles pour de telles exigences critiques conduit à une lecture détaillée et une analyse fine et nous amène à poser des questions auxquelles personne n'avait pensé auparavant [OFTA, 1997]. La figure 1.3 montre le cycle de développement avec l'utilisation des techniques formelles qui considèrent les exigences de sécurité au centre de la spécification.

Le passage de la spécification informelle à la spécification avec une méthode formelle est une tâche ardue en raison du besoin d'une expertise pour la formalisation mathématique des spécifications en langage naturel. Il est indéniable que ce passage n'est pas du tout aisé et qu'un appel à d'autres langages semi-formels peut combler le fossé entre les langages informels et les langages formels.

Cette phase de spécification est une phase primordiale pour la vérification des exigences de sécurité et la validation du système. Le choix d'une approche basée sur le couplage semi-

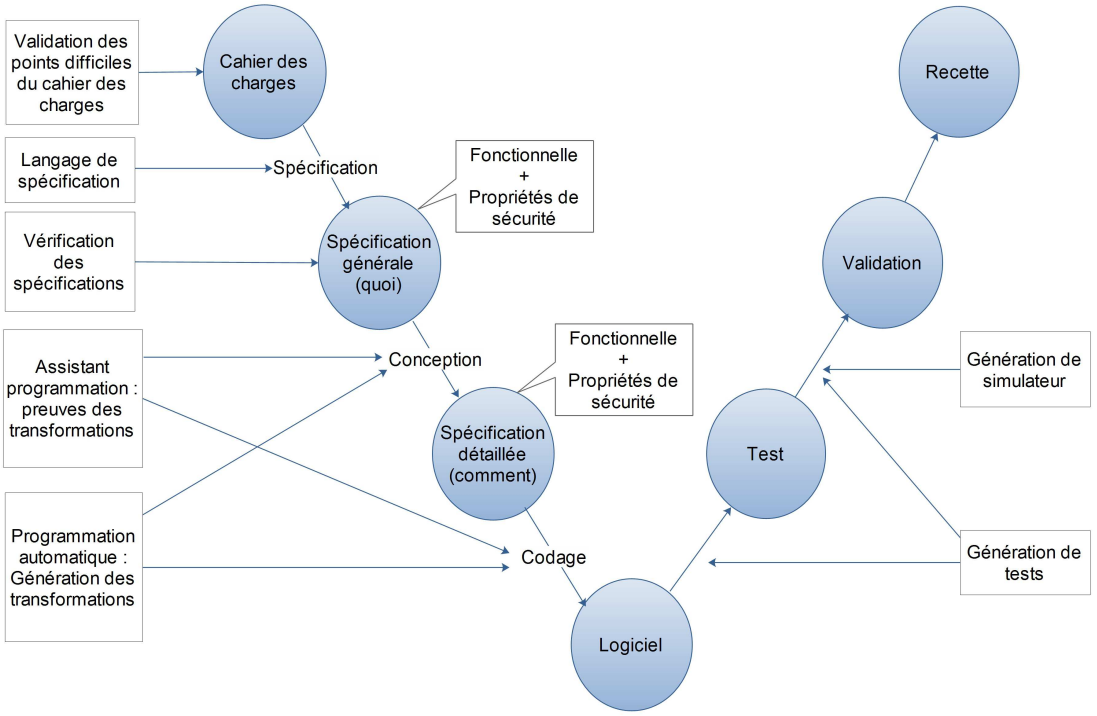

Figure 1.3 – Cycle de développement avec l'utilisation des techniques formelles inspiré de [OFTA, 1997]

formel/formel facilite ainsi la formalisation des exigences et permet d'entamer les activités de vérification et de validation.

1.3.2 La vérification et la validation formelles

L'utilisation d'un langage formel est considérée comme fondamentale pour mener rigoureusement les activités de vérification et de validation. Les techniques de vérification/ validation (V&V) sont nombreuses dans la littérature. Cependant, le choix d'une ou plusieurs parmi l'ensemble des techniques existantes repose sur leurs avantages et leurs limitations lors de leur mise en œuvre. D'une part, il existe des techniques telles que le test, la simulation et/ou l'animation qui ne sont pas exhaustives en termes de détection d'erreurs du système lors de son exécution. Par ailleurs, la preuve et le *model checking* sont des techniques de V&V exhaustives et rigoureuses, mais nécessitent des compétences poussées pour vérifier les exigences par démonstration de théorèmes (preuve) ou par exploration de l'espace d'états (*model checking*).

La figure 1.4 présente ces deux problématiques sous forme d'un processus à trois étapes. Ce processus établit la formalisation des spécifications informelles en des spécifications semi-formelles et la formalisation des spécifications semi-formelles en des spécification for-

melles qui seront l'entrée pour des activités de V&V.

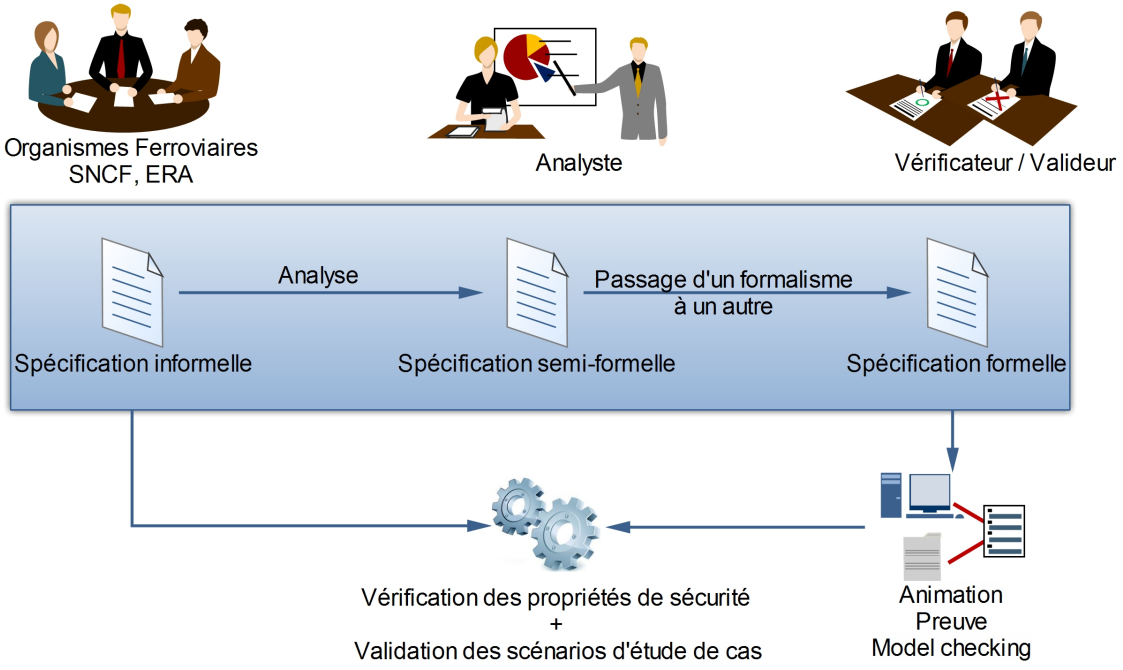

Figure 1.4 – Processus à trois étapes

1.4 Motivations et contributions

L'étude des règles d'exploitation ferroviaires est très peu abordée dans la littérature. La plupart des travaux de recherche dans le contexte ferroviaire se sont focalisés sur la modélisation de la partie sol dans le but de la gestion de la signalisation, la gestion des enclenchements et la gestion des itinéraires.

Notre étude préliminaire sur la problématique de cette thèse a permis de choisir les langages semi-formel et formel qui conviennent au besoin de spécification des règles d'exploitation. En ce qui concerne le langage semi-formel, nous avons choisi UML. Il s'agit d'une référence incontournable dans le génie logiciel avec la multitude de diagrammes et de vues à disposition pour une modélisation à la fois structurelle et comportementale. Dans l'optique de l'ingénierie dirigée par les modèles, le mécanisme d'extension par des profils UML permet de se focaliser sur les spécificités et les contraintes du domaine d'étude. En ce qui concerne le langage formel, la méthode B est considérée comme une méthode formelle complète et bien outillée qui couvre tout le processus de développement de la spécification à la génération de code exécutable. Les approches dans la littérature ont montré que la méthode B se prête bien pour un couplage avec UML. En outre, elle a été utilisée avec succès dans des applications industrielles ferroviaires depuis des années [Dollé *et al.*, 2003, Behm *et al.*, 1999].

Le contexte scientifique que nous avons défini pour cette thèse nous a mené à une étude de l'existant concernant les approches dirigées par les modèles permettant une proposition méthodologique outillée pour la modélisation des règles d'exploitation ferroviaires. En effet, cette étude doit tenir compte du fait que le système étudié inclut des acteurs humains et informatiques et gère des autorisations de déplacement des trains sur des lignes ferroviaires équipées d'ERTMS en France. Il s'est avéré alors que deux approches existantes dans le domaine des systèmes d'information peuvent être employées dans notre contexte industriel :

— Role Based Access Control (RBAC) est un modèle de contrôle d'accès basé sur les rôles. Un utilisateur se voit attribuer des permissions en fonction des rôles qu'il joue.
— Organization Based Access control (Or-Bac) est un modèle de contrôle d'accès basé sur les organisations qui vise à définir une politique de sécurité de façon plus riche et plus modulaire. Il étend RBAC avec les notions de contexte et de hiérarchie des organisations.

L'utilisation de ces deux approches constitue la contribution centrale de cette thèse. Elle met en exergue l'exploitation des approches de modélisation et de validation des systèmes d'information basées sur la « sécurité-confidentialité » au profit de la modélisation et la validation des systèmes ferroviaires basées sur la « sécurité-innocuité ». Ceci nous a permis de maîtriser la complexité des règles ferroviaires en garantissant une séparation des préoccupations : des aspects fonctionnels du système et des aspects de sécurité basés sur les responsabilités et les autorisations attribuées aux acteurs du système. Nous avons alors étudié la structuration du système ferroviaire en deux niveaux : un niveau fonctionnel, qui gère les déplacements des trains, et un niveau « sécuritaire » qui gère les autorisations de déplacement et les opérations effectuées par les différents acteurs du système (humains et logiciels). Nous avons utilisé le modèle RBAC et son extension en Or-Bac pour la spécification du modèle sécuritaire. Nos travaux ont notamment accompli les rapprochements suivants :

— Les acteurs correspondent aux utilisateurs d'un système sécuritaire ;
— Les responsabilités accordées aux acteurs correspondent aux rôles accordés aux utilisateurs ;
— La notion d'autorisation correspond à la notion sécuritaire de permission ;
— La variété des équipements et des réglementations nationales peut s'exprimer en introduisant une hiérarchie des organisations ;
— Les contraintes qui permettent de délivrer une autorisation peuvent s'appuyer sur divers types de contextes.

Les principales contributions de la thèse sont présentées de manière sommaire dans les

points suivants :

- L'adaptation d'une première approche existante pour le contrôle d'accès des systèmes d'information basée sur les rôles (Role Based Access Control) qui répond à nos besoins de spécification en UML/B de la réglementation.
- La vérification et la validation des exigences de sécurité ferroviaire ajoutées sur les spécifications formelles résultantes de cette approche.
- L'adaptation d'une deuxième approche existante pour le contrôle d'accès des systèmes d'information basée sur les organisations qui enrichit la première par des notions de contexte et de hiérarchie des organisations.

Nous avons présenté le contexte de cette thèse, la problématique et les contributions. Maintenant, nous détaillons l'organisation de ce mémoire.

1.5 Organisation du mémoire

Ce manuscrit est structuré en deux parties contenant chacune deux chapitres :

La première Partie est consacrée à la présentation du contexte et de l'état de l'art.

Le chapitre 2 dresse le contexte scientifique de nos travaux. Nous commençons, dans un premier temps, par un aperçu sur l'ingénierie système, puis nous introduisons l'ingénierie des exigences, l'un des processus de l'ingénierie système. Ensuite, nous nous focalisons sur deux types d'attributs de la sûreté de fonctionnement, à savoir : la « sécurité-innocuité » sur laquelle se base notre approche de modélisation et de validation des systèmes ferroviaires et la « sécurité-confidentialité » sur laquelle se basent les approches existantes des systèmes d'information. Dans un deuxième temps, nous abordons l'ingénierie dirigée par les modèles, ses concepts et ses verrous scientifiques. Ensuite, nous poursuivons en présentant l'intérêt de cette forme d'ingénierie dans l'industrie des systèmes critiques et en particulier pour les systèmes ferroviaires.

Le chapitre 3 est consacré au couplage des notations semi-formelles UML et des notations formelles B. Tout d'abord, nous mettons l'accent sur la phase de spécification et nous passons en revue les différents types de langages de spécification. Puis, nous présentons les phases de vérification et de validation ainsi que leurs différentes techniques. Une fois ces phases définies, nous passons au couplage semi-formel/formel. Nous donnons un aperçu sur les principes de couplage des notations. Puis, nous détaillons le choix des notations UML et B motivé par des critères que nous avons fixé. Ensuite, nous citons des travaux de la littérature qui sont liés aux approches de couplage UML/B. Enfin, nous présentons notre approche basée sur les notations UML et B utilisées dans le cadre du projet *PERFECT*.

La deuxième partie est bâtie sur les approches existantes RBAC et Or-Bac que nous avons utilisées pour la modélisation et la validation des règles d'exploitation ferroviaires.

Le chapitre 4 est consacré à la présentation de l'approche RBAC pour la modélisation et la validation des règles d'exploitation ferroviaires. Nous commençons par l'analyse de ces règles d'exploitation qui montre que celles-ci peuvent être modélisées d'une façon analogue à la modélisation des politiques de sécurité pour les systèmes d'information. Par la suite, nous présentons le modèle RBAC basé sur la notion de rôle, le méta-modèle SecureUML ainsi que la plate-forme B4MSecure permettant une modélisation conjointe UML/B des politiques de contrôle d'accès. Nous illustrons l'idée de l'utilisation du modèle RBAC par une étude de cas ferroviaire fondée sur un scénario nominal d'autorisation de mouvement « MA » et un scénario exceptionnel de franchissement d'un arrêt ETCS « Override EOA ». À partir de cette étude de cas, nous élaborons la modélisation UML/B avec B4MSecure permettant la séparation des aspects fonctionnels et des aspects de sécurité. Les spécifications résultantes en B seront ensuite soumises à la vérification et la validation formelles avec les outils *Atelier B* et *ProB* en y ajoutant des exigences de sécurité ferroviaire.

Le chapitre 5 aborde la deuxième approche Or-Bac pour les politiques de sécurité des systèmes d'information. Dans cette approche, des nouvelles notions sont définies face aux limitations du modèle RBAC. Nous commençons donc ce chapitre par une comparaison entre le modèle Or-Bac et le modèle RBAC en citant les avantages du premier face aux limitations du deuxième. Puis, nous démontrons l'intérêt de l'approche Or-Bac sur les systèmes ferroviaires. En effet, cette approche propose la notion d'organisation, comme l'élément central, et la notion de contexte permettant de structurer les contraintes concrètes de ces systèmes. Cette approche permet également de mettre en valeur l'interopérabilité ferroviaire, l'un des enjeux majeurs d'ERTMS, grâce à la hiérarchie des organisations. Ensuite, nous présentons le méta-modèle Or-Bac, extension du modèle RBAC, le profil UML correspondant ainsi que la classification des organisations dans une optique d'interopérabilité. Cette classification sera formalisée en B selon une stratégie d'architecture pour une implémentation des différentes organisations. Enfin, nous illustrons notre contribution par un scénario d'accident « Saint-Romain-en-Gier » qui s'appuie sur la notion de contexte.

Ce manuscrit s'achève sur une conclusion générale en donnant un bilan de nos contributions ainsi qu'un ensemble de perspectives à moyen et à long terme.

Publications issues de ce travail de thèse

Revues nationales

- Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Ledru, Y. et Idani, A. (2015). ≪ Formalismes basés sur les rôles pour la modélisation et la validation des règles d'exploitation ferroviaires ≫. *In Technique et Science Informatiques*, France. Lavoisier. Extension AFADL 2014.

Conférences nationales

- Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Ledru, Y. et Idani, A. (2014). ≪ Modélisation et validation formelle des règles d'exploitation ferroviaires ≫. *In Approches Formelles dans l'Assistance au Développement de Logiciels*, pages p1-15, France.

Conférences internationales

- Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Idani, A. et Ledru, Y. (2014). ≪ B Formal Validation of ERTMS/ETCS Railway Operating Rules ≫. *In 4th International ABZ Conference*, pages p124-129, France.
- Ben Ayed, R., Bon, P. et Collart-Dutilleul, S. (2014). Checking the European Railways Traffic Management System (ERTMS) operating rules using UML and B method. *In 14th International conference on Railway Engineering Design and Optimization*, pages p139-149, Italy.

Livrables

- Lanusse, A., Ferlin, A., Ayed, R. B., Sun, P., Boudi, Z., Perin, M. et Faivre, A. (2014). Livrable D2.1 du projet PERFECT. ≪ Propositions méthodologiques préliminaires pour le support à la certification à l'aide de méthodes formelles selon une approche orientée modèles ≫.

Présentations orales

- Ben Ayed, R. ≪ Modélisation UML et validation B de règles d'exploitation ferroviaires ≫. *Journée MFDL/MTV2*, Grenoble, 13-14 Janvier 2014.

Première partie

CONTEXTE ET ÉTAT DE L'ART

Chapitre 2

Contexte scientifique

Sommaire

2.1 Introduction

Les systèmes critiques présentent une part importante des secteurs industriels tels que le domaine du ferroviaire, de l'automobile, de l'aéronautique, du nucléaire, etc. La conception et le développement de ces systèmes doivent tenir compte non seulement des aspects fonctionnels mais aussi des aspects non-fonctionnels tels que les propriétés de sécurité et de la sûreté de fonctionnement, les contraintes temporelles, la fiabilité, la qualité de service, la flexibilité, l'évolutivité, la maintenabilité, la performance, ... Dès lors, la communauté scientifique a amélioré significativement le développement des systèmes informatiques sur la base de ces exigences et a approfondi ses recherches sur les méthodologies orientées modèles faisant face à la complexité des systèmes informatiques telles que l'ingénierie dirigée par les modèles.

Ce chapitre a pour objectif de dresser, en premier lieu, un aperçu sur l'ingénierie système. Parmi les différents processus de l'ingénierie système, nous introduisons l'ingénierie des exigences. La sûreté de fonctionnement (SdF) et plus précisément la sécurité-innocuité (« *safety* » en anglais), et la sécurité-confidentialité (« *security* » en anglais) sont les exigences sur lesquelles nous nous sommes focalisés. De telles exigences sont soumises à des normes qui exigent des impératifs à respecter.

En deuxième lieu, un état de l'art sur l'ingénierie dirigée par les modèles (IDM), en anglais *Model Driven Engineering (MDE)*, est abordé. Partant du principe « tout est modèle », nouvelle orientation du principe « tout est objet » de l'approche objet des années 80, nous donnons un aperçu sur les concepts de base de l'IDM à savoir la modélisation, la méta-modélisation et la transformation de modèles. Nous présentons par la suite quelques verrous scientifiques susceptibles d'être pris en considération lors du développement d'un système.

Nous entamons la dernière section par la mise en application de l'IDM dans plusieurs secteurs industriels. Faisant l'objet de notre étude, nous mettons à cet effet l'accent sur la mise en application de l'IDM pour les systèmes ferroviaires. Ceci expose l'intérêt de l'IDM vis-à-vis des préconisations des systèmes ferroviaires et étudie les projets existants dans la littérature.

2.2 Ingénierie Système

Face à la complexité des systèmes d'aujourd'hui, la disposition d'un ensemble d'activités permettant la conception et le développement du système d'une manière structurée devient nécessaire. Cette nécessité s'est traduite par l'apparition d'un nouveau domaine in-

titulé l'*ingénierie système* ou *ingénierie de systèmes* [1]. Cette nouvelle science est le résultat d'un retour d'expérience de la part de grandes entreprises industrielles impliquées dans le développement des systèmes complexes, à savoir Airbus, Thales, Alstom, Altran, EDF, etc. L'Association Française d'Ingénierie Système [2] (AFIS) regroupe ces entreprises parmi d'autres. Elle est affiliée à l'association internationale d'Ingénierie Système INCOSE [3] (International Council on Systems Engineering) en tant que « chapitre français de l'INCOSE ». L'objectif de cette association est de profiter de la synergie des compétences de ces entreprises pour enrichir et améliorer les bonnes pratiques de l'ingénierie système.

L'AFIS définit l'Ingénierie Système comme « *une démarche méthodologique générale 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* ». Les pratiques de cette démarche sont aujourd'hui répertoriées dans des normes, qui décrivent les pratiques métier en termes de processus réalisées à l'aide de méthodes supportées par des outils [Groupe de Travail Ingénierie Système de l'AFIS, 2009].

2.2.1 Ingénierie des exigences

L'une des disciplines de l'ingénierie système est l'ingénierie des exigences à laquelle se consacre une partie de notre travail. Elle occupe une part importante de l'ingénierie système. Pour mieux appréhender cette démarche, nous devons définir tout d'abord ce qu'est une exigence. Selon [Groupe de Travail Ingénierie Système de l'AFIS, 2009], « *une exigence prescrit une propriété dont l'obtention est jugée nécessaire. Son énoncé peut être une fonction, une aptitude, une caractéristique ou une limitation à laquelle doit satisfaire un système, un produit ou un processus* ».

Ces exigences peuvent exprimer un besoin en fonctionnalité ou en qualité : Il existe des exigences fonctionnelles et non-fonctionnelles. Les exigences non-fonctionnelles peuvent être exprimées par un ensemble de propriétés, définies en termes de contraintes temporelles, de qualité de service, de sûreté de fonctionnement, de sécurité informatique, etc.

Au cœur des processus techniques d'ingénierie d'un système, l'ingénierie des exigences vise essentiellement à développer les exigences et à les gérer. En effet, le développement des exigences a comme objectif d'obtenir un référentiel d'exigences validé par les parties prenantes, alors que l'objectif de la gestion des exigences est de maintenir ce référentiel stable dans le temps [Fanmuy *et al.*, 2001]. D'où, selon l'AFIS [4], « *l'ingénierie des exigences*

1. http ://www.afis.fr/nm-is/Pages/Ingénierie Système/Ingénierie Système.aspx
2. AFIS : https ://www.afis.fr/pages/accueil.aspx
3. INCOSE : http ://www.incose.org
4. https ://www.afis.fr/nm-is/Pages/Ingénierie des exigences.aspx

consiste, au travers de méthodes, règles et processus, à établir et maintenir un référentiel unique qui s'affine et se complète au cours du développement et qui est maintenu tout au long de la vie du système ≫.

La discipline de l'ingénierie des exigences contient deux éléments de base : les exigences et les besoins. Telles qu'elles sont déjà définies, les exigences représentent une vision du système du point de vue des concepteurs. Elles formalisent l'expression des besoins qui représentent une vision du système d'un point de vue utilisateur. Le passage des besoins vers les exigences étant critique, l'enjeu majeur de l'ingénierie des exigences est d'assurer un processus de transformation des besoins exprimés par les clients en exigences systèmes qui seront techniquement exploitables [Konaté, 2009]. Concrètement, le découpage du processus d'ingénierie des exigences varie selon les auteurs. Selon [Nuseibeh et Easterbrook, 2000], ce processus comprend les phases d'élicitation, de modélisation, d'analyse de spécification et de validation. D'autres auteurs tels que [Kotonya et Sommerville, 1998] proposent un découpage en trois phases qui intègre la modélisation dans la phase d'analyse, comparé avec le découpage de Nuseibeh et Easterbrook. Dans notre étude, nous nous intéressons aux phases de modélisation (intégrant la phase d'analyse de spécification) et de validation des exigences et plus précisément de sécurité. Ce qui nous incite à poursuivre cet état de l'art par les définitions de sûreté de fonctionnement et de sécurité dans la section 2.2.2.

La sécurité, l'une des propriétés de la sûreté de fonctionnement, est un mot clé directeur de notre travail de thèse. Nous distinguons la sécurité-innocuité de la sécurité-confidentialité (ou la sécurité des systèmes d'information). L'un de nos centres d'intérêt, dans le cadre du projet *PERFECT*, est la validation des propriétés de sécurité-innocuité sur des systèmes ferroviaires. Pour atteindre cet objectif, nous avons employé des techniques inhérentes à la sécurité-confidentialité. C'est la raison pour laquelle nous décrivons, dans ce qui suit, les exigences de sûreté de fonctionnement ainsi que leurs cadres normatifs. Nous donnons aussi un aperçu sur les politiques de contrôle d'accès qui nous semble être approprié pour la représentation de notre problématique.

2.2.2 Exigences de Sûreté de Fonctionnement et cadre normatif

La sûreté de fonctionnement (SdF) est une préoccupation récurrente de plusieurs domaines industriels. Elle est définie dans [Laprie *et al.*, 1995] comme étant ≪ *la propriété qui permet aux utilisateurs d'un système de placer une confiance justifiée dans le service qu'il leur délivre* ≫.

La sûreté de fonctionnement est caractérisée par un ensemble d'attributs selon les applications auxquelles le système est destiné [Kaâniche, 1999]. Ces attributs sont vus comme des propriétés différentes mais complémentaires :

— *la continuité du service conduit à la fiabilité,*
— *l'aptitude aux réparations et aux évolutions conduit à la maintenabilité,*
— *le fait d'être prêt à l'utilisation conduit à la disponibilité,*
— *l'absence d'altérations inappropriées du système conduit à l'intégrité,*
— *l'absence de conséquences catastrophiques pour l'environnement conduit à la sécurité-innocuité (au sens « safety » en anglais),*
— *l'absence de divulgations non-autorisées de l'information conduit à la confidentialité.*

L'ensemble des activités d'évaluation de la fiabilité, de la maintenabilité, de la disponibilité et de la sécurité forme ce qu'on appelle **FMDS** d'un système.

Les objectifs en termes de sûreté de fonctionnement varient en fonction du domaine industriel considéré. Par exemple, dans les transports, la *fiabilité* et la *sécurité-innocuité* sont privilégiés tandis que pour les systèmes d'information, la *confidentialité*, l'*intégrité* et la *disponibilité* sont mises en avant. Ces trois dernières propriétés se rapportent à l'information et définissent la *sécurité-confidentialité* au sens « *security* » en anglais. La figure 2.1, extraite de [Laprie *et al.*, 1995], schématise les attributs de la sûreté de fonctionnement et illustre la sécurité-innocuité et la sécurité-confidentialité.

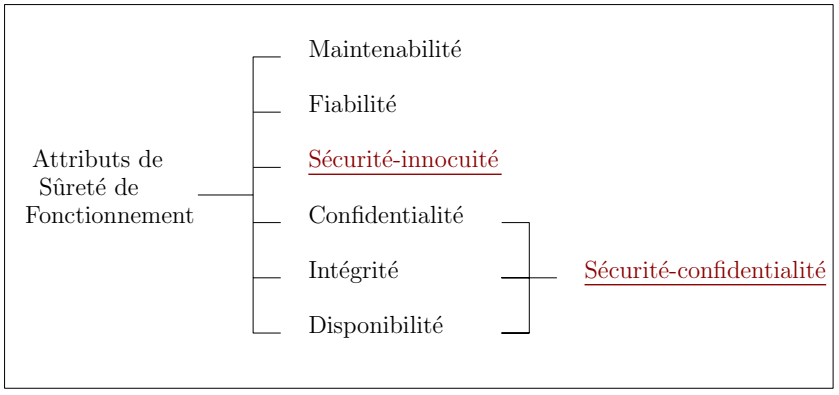

Figure 2.1 – Les attributs de sûreté de fonctionnement

Les précautions à prendre en compte dans le développement des systèmes critiques sont généralement fixées par une norme et dépendent du domaine d'application et surtout du degré de criticité du système. Ces systèmes critiques sont, généralement, soumis à des autorités de certification qui vérifient que les impératifs prévus par la norme ont été respectés. Le processus de certification consiste en l'attestation, par un organisme reconnu et habilité par les pouvoirs publics de l'État, de la conformité du produit certifié à un référentiel composé généralement de normes et de textes législatifs ou réglementaires [Sallak et Schön, 2013].

L'une des principales préoccupations de la sûreté de fonctionnement pour les industriels

est la sécurité. L'ensemble des notions générales de la sûreté de fonctionnement étant défini, nous ne présentons dans la suite que les concepts de la sécurité : au sens sécurité-innocuité (≪ *safety* ≫) et au sens de sécurité des systèmes d'information (≪ *security* ≫).

2.2.2.1 Sécurité-innocuité

La sécurité-innocuité exprime que des conséquences catastrophiques ne se produisent jamais en cours d'exécution. Elle définit les propriétés selon lesquelles un système est dit ≪ sûr ≫. Celle-ci a été normalisée par un référentiel réglementaire pour la sécurité : la norme IEC 61508. Cette norme présente une approche générique de toutes activités liées au cycle de vie des systèmes électriques-électroniques-programmables et une approche globalisée de la sécurité au sens sécurité-innocuité [IEC, IEC 61508, Norme Internationale, 1998]. Plusieurs déclinaisons de cette norme ont vu le jour définissant chacune les normes qui rendent applicable l'EN 61508 pour différents secteurs concernés. Nous pouvons citer la norme IEC 61513 pour le secteur nucléaire, les normes EN 50126/EN 50128/EN 50129 pour le secteur ferroviaire ou encore la norme ISO 26262 pour le secteur de l'automobile (Figure 2.2).

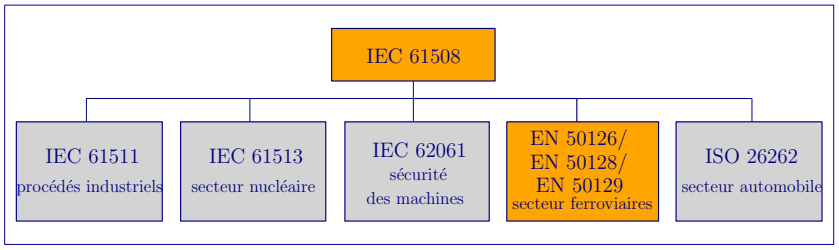

Figure 2.2 – La norme IEC 61508 et ses déclinaisons

Ces normes fixent des niveaux de sécurité ≪ SIL ≫ (Safety Integrity Level) pour l'IEC 61508 [Ingrey *et al.*, 2007]. Il existe quatre niveaux pour le référentiel SIL : SIL1, SIL2, SIL3 et SIL4, dans le sens croissant du niveau de sécurité. Le SIL permet de quantifier le niveau de sûreté de fonctionnement d'un dispositif de sécurité. Il prend en compte sa fiabilité, sa tolérance par rapport à ses propres défaillances ainsi que la capacité du fabricant à éviter de générer des défauts de conception matérielle ou logicielle pouvant réduire le niveau de sécurité. Des techniques et des méthodes sont mises en œuvre par les normes afin de respecter le niveau SIL objectif et démontrer que cet objectif est atteint. L'intérêt de cette démarche est alors de donner confiance aux clients, à travers une certification opérée par un organisme compétent.

Sécurité des systèmes ferroviaires :

La démonstration de la sécurité pour les systèmes ferroviaires est toujours une activité critique. Le référentiel normatif CENELEC pour les systèmes ferroviaires contient les normes :

— EN 50126 qui concerne la spécification et la démonstration de la fiabilité, de la disponibilité, de la maintenabilité et de la sécurité (FDMS) [CENELEC, NF EN 50126, 2000],
— EN 50128 qui concerne les systèmes de signalisation, de télécommunication et de traitement et plus précisément les logiciels pour systèmes de commande et de protection ferroviaire [CENELEC, NF EN 50128, 2001],
— EN 50129 qui concerne également les systèmes de signalisation, de télécommunication et de traitement et plus précisément les systèmes électroniques de sécurité pour la signalisation [CENELEC, NF EN 50129, 2003].

Nous pouvons citer, comme premier exemple de propriété de sécurité, l'absence de collision entre les trains.

2.2.2.2 Sécurité-confidentialité

Comme nous l'avons déjà introduit ci-dessus, la sécurité-confidentialité est définie par trois attributs : la confidentialité, l'intégrité et la disponibilité. Elle permet de garantir la sécurité des systèmes d'information.

Sécurité des systèmes d'information :

Afin de faire face aux vulnérabilités et aux attaques accidentelles ou malintentionnées des systèmes d'information, plusieurs techniques peuvent être employées telles que le contrôle d'accès et la cryptographie. Nous nous intéressons au contrôle d'accès qui est un processus protégeant les tentatives d'accès aux ressources du système contre un accès inapproprié ou non désiré tout en déterminant les activités autorisées des utilisateurs légitimes et en garantissant les propriétés de *sécurité-confidentialité* [Hu *et al.*, 2006].

Dans le but de renforcer les mécanismes de contrôle d'accès, des politiques de sécurité sont définies. Le standard pour la sécurité des systèmes d'information (ITSEC) [ITSEC, 1991], qui a été supplanté par la norme internationale ISO 15408 des critères communs pour l'évaluation de la sécurité des technologies d'information [ISO/IEC 15408-1, Norme Internationale, 1999], définit une politique de contrôle d'accès comme étant « *l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique* ». Une politique de contrôle d'accès définit alors les autorisations, éventuellement les interdictions, ainsi que les personnes habilitées à modifier cette politique. Une autorisation a pour but de ne permettre que les actions légitimes, et ainsi d'empêcher un utilisateur n'ayant pas les droits d'accès d'exécuter des opérations qui ne devraient pas lui être permises.

Les principales politiques proposées sont :

- Les politiques discrétionnaires DAC [Lampson, 1971] : des permissions sont ac-

cordées aux sujets en fonction de leur identité uniquement,

- Les politiques obligatoires MAC [Bell et LaPadula, 1976] : des permissions sont accordées en fonction du niveau de confiance des sujets,
- Les politiques à base de rôles telles que RBAC [Sandhu *et al.*, 2000] et Or-Bac [Abou EL Kalam, 2003]. Les politiques basées sur les rôles sont les plus récentes. À chaque rôle, des permissions ou privilèges, ensemble de droits correspondant aux tâches qui peuvent être réalisées, sont associées. Contrairement aux modèles précédents, les permissions ne sont plus associées d'une façon directe aux sujets, mais à travers des rôles. Les droits d'accès sont accordés aux rôles, ils sont ainsi attribués aux sujets en fonction des rôles qu'ils jouent.

Nous avons maintenant défini les différents aspects de la sécurité. Cette sécurité est primordiale pour les systèmes industriels critiques et pour lesquels les approches basées sur les modèles sont vues comme des méthodologies à forte valeur ajoutée. L'ingénierie dirigée par les modèles munie de ses concepts de modélisation, de méta-modélisation et de transformation de modèles offre une solution intéressante à l'ingénierie des exigences. En effet, la formalisation à travers le cadre structurel qu'offre la notion de méta-modélisation et l'automatisation des activités, via la transformation de modèles, représente un vrai apport pour la traçabilité entre les différentes étapes d'ingénierie des exigences. Après un tour rapide sur l'ingénierie système, l'ingénierie des exigences, ainsi que les exigences de sûreté de fonctionnement et principalement celles qui concernent la sécurité, nous ébauchons un état de l'art sur l'ingénierie dirigée par les modèles.

2.3 Ingénierie dirigée par les modèles

2.3.1 Du « tout est objet » vers le « tout est modèle »

Les langages à objets ont vu le jour au début des années 1970 avec le langage *SIMULA* [Dahl, 2004]. Ils sont sortis des laboratoires de recherche au milieu des années 1980, et depuis le début des années 1990 ils ont pris une place prépondérante et sont utilisés à grande échelle dans des domaines de l'informatique. Depuis lors, ces langages orientés objets ainsi que les architectures logicielles et les techniques de conception qui les supportent n'ont pas cessé d'évoluer.

Cette discipline de programmation orientée objet a apporté des réponses à nombreuses questions liées principalement à l'efficacité de la production des logiciels de qualité. Cette discipline met au premier plan la notion d'objet désignant une entité informatique appartenant au monde réel. Il possède deux caractéristiques : un état et un comportement. Il maintient son état via des variables appelées attributs et implémente son comportement à l'aide de méthodes.

Après cette vision basée sur les objets des années 80 et de son principe qui en découle « tout est objet », une nouvelle orientation d'ingénierie, appelée ingénierie dirigée par les modèles (IDM) est apparue avec le principe « tout est modèle ». L'IDM est le résultat de l'évolution du « génie logiciel » dans le but de maîtriser la complexité des systèmes informatiques qui ne cessent de s'accroître. L'IDM place la notion de modèle (plutôt que le code) au centre du cycle de développement et elle propose de modéliser les applications à un haut niveau d'abstraction sans penser à sa technologie cible. Dans ce paradigme, le code source n'est plus considéré comme l'élément central d'un logiciel, mais plutôt comme un élément dérivé des éléments de modélisation.
Le modèle est ainsi défini dans [Bézivin *et al.*, 2001] comme suit :

Définition 1 (Model) *A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system.*

Cette définition est traduite dans [Combemale, 2008b] :

Définition 2 (Modèle) *Un modèle est une abstraction d'un système, modélisé sous la forme d'un ensemble de faits construits dans une intention particulière. Un modèle doit pouvoir être utilisé pour répondre à des questions sur le système modélisé.*

[Combemale, 2008b] définit alors l'IDM comme « *une forme d'ingénierie générative dans laquelle tout ou partie d'une application est engendrée à partir de modèles* ».

[Bézivin, 2004] considère l'IDM comme une continuité des travaux précédents de la technologie objet, mais aussi une rupture. En effet, cette technologie des objets est considérée comme un point de départ vers de nouveaux chemins de migrations technologiques telles que l'ingénierie des modèles. D'ailleurs, les solutions proposées par l'IDM sont considérées plutôt complémentaires et non antagonistes avec celles de la technologie des objets du fait que l'IDM ne contredit pas les apports de la programmation par objets [Bézivin, 2004]. Toutefois, ce passage peut être considéré aussi comme une rupture du fait des perspectives et des différentes dimensions à considérer de l'une ou de l'autre. Autant dire qu'il est nécessaire de distinguer les relations de base de la technologie des objets, essentiellement héritage et instanciation, de celles de l'ingénierie des modèles, essentiellement représentation et conformité. La figure 2.3 montre les différentes dimensions permettant le passage de la technologie des objets à l'ingénierie des modèles.

La tendance de la technologie objet des années 80 a mis en avant les relations d'héritage et d'instanciation qui peuvent être exprimées par le fait qu'un objet peut être une instance d'une classe et une classe peut hériter d'une autre classe (Figure 2.3 (a)). La nouvelle tendance de l'ingénierie des modèles met en avant le modèle qui représente une vue particulière du système et considère que le modèle doit être écrit conformément au langage de son méta-modèle [Bézivin, 2005] (Figure 2.3 (b)) défini dans la section 2.3.2.1. Par

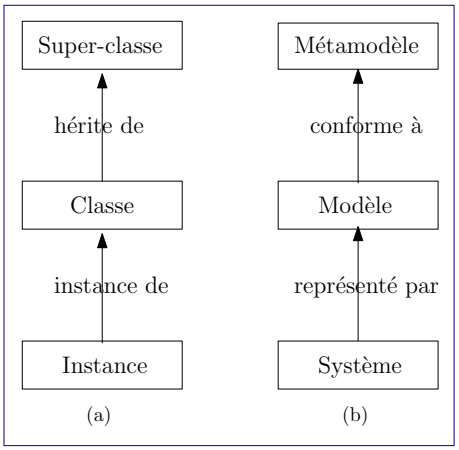

Figure 2.3 – Le passage de la technologie des objets à l'ingénierie des modèles [Bézivin, 2005]

conséquent, le passage de la technologie objet à l'ingénierie des modèles est justifié par l'utilisation de techniques similaires liées à l'abstraction.

Le modèle permet l'abstraction de la complexité d'un système en se focalisant sur une préoccupation bien particulière. Cette abstraction fournit non seulement une meilleure maîtrise de la complexité des systèmes mais aussi l'augmentation de la productivité et de la réutilisabilité ainsi que l'amélioration de la qualité logicielle. Un modèle est considéré utile s'il permet d'acquérir une meilleure compréhension du système. Dans un contexte d'ingénierie, un modèle est utile s'il permet de produire les mesures appropriées qui doivent être prises en compte pour atteindre et maintenir l'objectif du système. C'est sur ce principe de base que l'IDM s'appuie fondamentalement pour définir les deux principaux enjeux : la méta-modélisation et la transformation de modèles.

2.3.2 Concepts de base de l'IDM

L'objectif majeur de l'IDM est de définir un niveau d'abstraction plus élevé du système et d'automatiser le processus de développement. Il existe trois concepts de base de l'IDM : le modèle, le méta-modèle et le méta-méta-modèle.

Le modèle étant déjà défini dans la section précédente, nous nous focalisons sur la définition du méta-modèle, du méta-méta-modèle et de leurs langages.

2.3.2.1 Méta-modèle et langage de modélisation

Pour être en mesure de manipuler des modèles, leur langage doit être spécifié comme un modèle de ces modèles, appelé méta-modèle. D'où, un méta-modèle est un modèle

d'un langage de modélisation [Sprinkle *et al.*, 2014]. La définition de l'OMG [5] (Object Management Group) du méta-modèle est la suivante :

Définition 3 (Meta-model) *A meta-model is a model that defines the language for expressing a model*

Un modèle est une abstraction d'un système réel tandis qu'un méta-modèle est une abstraction d'ordre supérieur mettant en évidence les concepts utilisés pour définir le modèle [Barais, 2007]. D'où la relation, déjà évoquée dans la section 2.3.1, un modèle est « conforme à » son méta-modèle. Par analogie aux langages de programmation, un méta-modèle est l'équivalent de la syntaxe d'un langage. La différence est qu'un modèle se concentre sur la syntaxe abstraite alors qu'un programme se concentre sur la syntaxe concrète [Combemale *et al.*, 2006]. En ce qui concerne la sémantique, elle constitue le vocabulaire choisi pour nommer les concepts fondamentaux du domaine défini par la syntaxe abstraite. Bien que les deux concepts de langage et méta-modèle soient proches et ils sont souvent confondus dans la littérature, ils sont néanmoins différents [Bézivin *et al.*, 2004]. Selon le rapport de synthèse de l'AS CNRS [6] sur le MDA [Bézivin *et al.*, 2004] : *un langage est un système abstrait alors qu'un méta-modèle est une définition explicite de ce langage*. En d'autres termes, le langage joue le rôle de système, tandis que le (ou les) méta-modèle(s) joue(nt) le rôle de modèle(s) de ce langage.

2.3.2.2 Méta-méta-modèle et langage de méta-modélisation

À l'instar du méta-modèle qui définit le langage de modélisation et qui interprète un modèle, le méta-méta-modèle dispose d'une description du langage dans lequel est écrit le méta-modèle. C'est un modèle pour les méta-modèles [Hammoudi, 2010]. Dans le but de limiter le nombre de niveaux d'abstraction et de contourner la possibilité de définir un méta-méta-méta-modèle et ainsi de suite, ces langages de méta-modélisation sont dotés d'une propriété de méta-circularité. C'est la capacité de se décrire lui-même. Le caractère réflexif du méta-méta-modèle est illustré par le niveau *M3* dans la figure 2.4 qui représente les quatre niveaux de l'architecture de l'IDM. Cette architecture est également introduite par l'OMG en tant qu'*architecture 3+1* [Bézivin, 2005]. Le niveau *M0* représente le système réel. Au niveau *M1*, le modèle représente le système. Ce modèle est conforme à son méta-modèle au niveau *M2*. De même, ce méta-modèle est conforme à son méta-méta-modèle au niveau *M3*.

L'OMG a proposé un langage standardisé MOF (Meta Object Facility) pour la définition et la spécification des méta-modèles [OMG, 2015]. Le but de modélisation au travers MOF est de décrire les modèles dans ces différentes couches de la figure 2.4 en utilisant des

5. Object Management Group : www.omg.org
6. Actions Spécifiques du CNRS

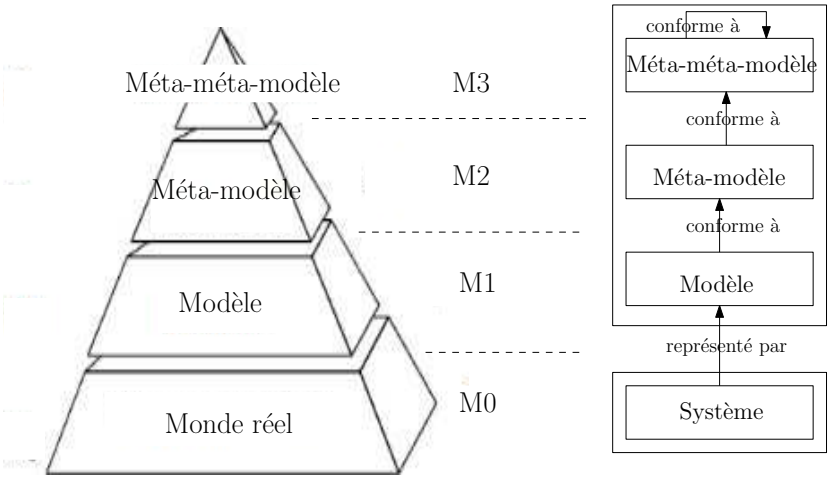

Figure 2.4 – Les quatre niveaux de l'architecture de l'IDM

abstractions de modélisation communes. Cela permet un accès homogène aux modèles dans les différents niveaux (ou différentes couches) grâce à la réflexion. En outre, cela permet de standardiser l'accès aux outils à travers une API (Application Programming Interface) commune et de sérialiser les modèles grâce au standard XMI (XML Metadata Interchange) [Sprinkle *et al.*, 2014]. Le MOF a contribué de manière significative aux principes fondamentaux de l'architecture dirigée par les modèles MDA de l'OMG.

2.3.3 L'approche MDA

L'architecture dirigée par les modèles ou MDA (en anglais Model Driven Architecture) est une approche de l'IDM proposée par l'OMG en 2000 (Figure 2.5). Cette approche vise la promulgation des bonnes pratiques de modélisation en exploitant les avantages de considérer le modèle au centre du développement logiciel. Elle se base sur la séparation des préoccupations entre la logique métier des systèmes informatiques et les plate-formes utilisées. Cette séparation est conçue, entre autres, dans le but d'éviter de mêler les spécifications fonctionnelles d'un système avec des spécifications techniques de la phase de mise en production sur une plate-forme donnée.

Plusieurs formalismes standards de modélisation sont définis par MDA, notamment UML (Unified Modeling Language), MOF et XMI, afin de promouvoir les qualités intrinsèques des modèles telles que pérennité, productivité et prise en compte des plate-formes d'exécution [Combemale, 2008b]. Le MDA place le modèle au centre du cycle de développement. En effet, en s'appuyant sur le standard UML, le principe clé du MDA consiste en l'utilisation des modèles séparément aux différentes phases de développement d'une application : de la phase d'élicitation des exigences jusqu'au codage (Figure 2.6). À cet effet, le MDA préconise l'élaboration des modèles suivants [Combemale, 2008b] :

— **CIM (Computation Independant Model)** qui est un modèle d'exigences indépen-

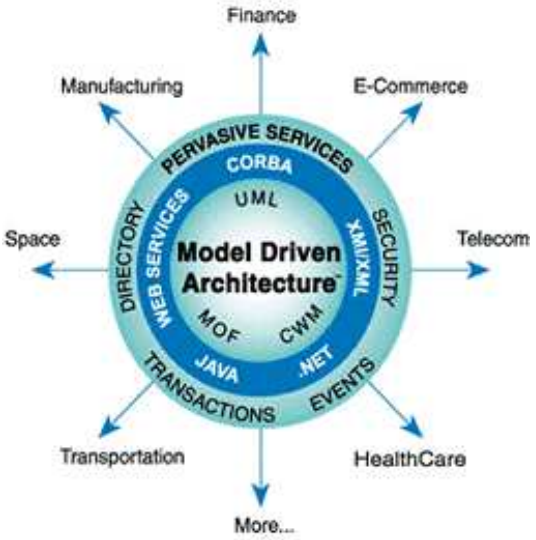

Figure 2.5 – MDA par l'OMG

dant de l'information,

— **PIM (Platform Independant Model)** qui est un modèle d'analyse et de conception indépendant de tout détail technique de la plate-forme,
— **PSM (Platform Specific Model)** qui est un modèle de code, combinant les spécifications du PIM avec les détails propres de la plate-forme, et
— **PDM (Platform Description Model)** qui est un modèle de description de la plate-forme.

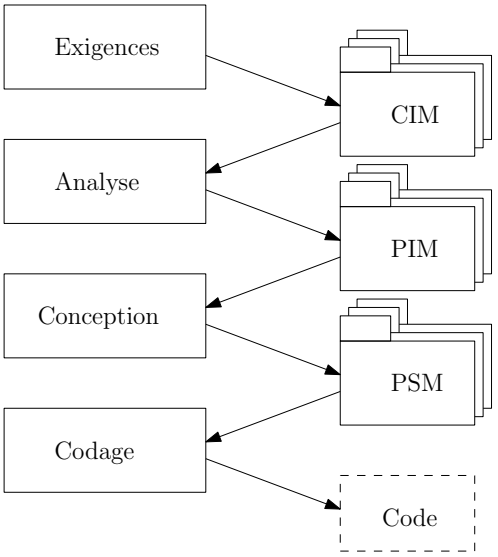

Figure 2.6 – Le cycle de développement d'une application en MDA

Le cycle de développement d'un logiciel selon l'approche MDA prend la forme de « Y » (Figure 2.7). Ce cycle, propre au MDD (Model Driven Development), met l'accent sur les différents niveaux des modèles présentés ci-dessus. Le PSM représente la plate-forme

de mise en œuvre permettant de générer un modèle spécifique à partir de PIM. Il peut être raffiné jusqu'à l'obtention d'une implémentation exécutable. Le passage de PIM à PSM nécessite des mécanismes de transformations qui garantissent la pérennité des modèles et leur productivité. Nous détaillons ces mécanismes dans la section suivante.

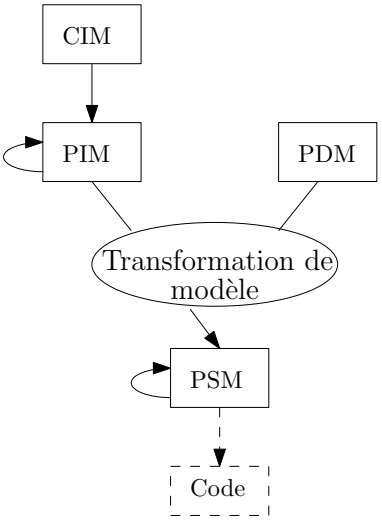

Figure 2.7 – Le processus de développement en Y

2.3.4 La transformation de modèles

Afin de rendre les modèles utilisables et opérationnels, la transformation de modèles est l'une des problématiques clés de l'IDM. Elle se situe au centre de l'approche MDA [Combemale, 2008b].

Le processus de transformation de modèles en MDA repose sur la génération d'un ou plusieurs modèles cibles à partir d'un ou plusieurs modèles sources. Cette transformation s'effectue par l'intermédiaire de règles qui décrivent la correspondance entre une ou plusieurs constructions du modèle source et une ou plusieurs constructions du modèle cible. Les modèles source et cible de la transformation doivent être conformes à leurs méta-modèles source et cible. Ceci est réalisé par un moteur de transformation (voir figure 2.8).

Selon la nature des méta-modèles source et cible et le changement du niveau d'abstraction, nous pouvons distinguer quatre types de transformations à savoir les transformations endogènes et exogènes et les transformations horizontales et verticales :

— Une transformation est endogène si les modèles source et cible ont le même méta-modèle,
— Une transformation est exogène si les modèles source et cible sont conformes à des méta-modèles différents,
— Une transformation est verticale si le niveau d'abstraction change comme par exemple dans le cas de passage de PIM à PSM et inversement,

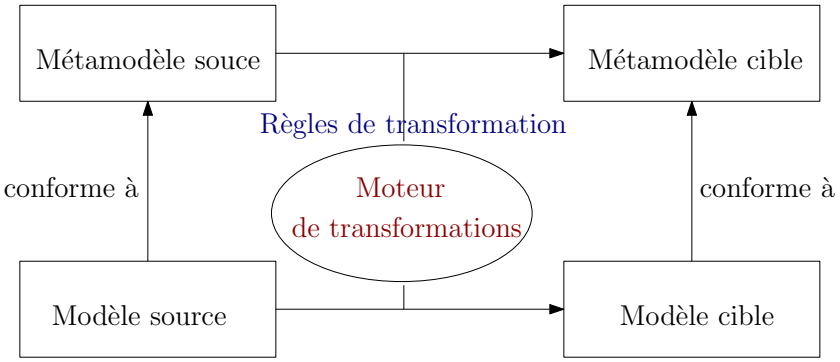

Figure 2.8 – Principe de transformation de modèles

— Une transformation est horizontale si le niveau d'abstraction ne change pas comme par exemple dans le cas de transformation de PIM à PIM ou de PSM à PSM.

La figure 2.9, issue de [Combemale, 2008a], donne des exemples de la combinaison des différents types de transformations. La restructuration, la normalisation et l'intégration de patrons sont des transformations endogènes horizontales ; tandis que la migration de logiciel et la fusion de modèles sont des transformations exogènes horizontales. Dans le cas de passage de PIM vers PSM pour le raffinement, la transformation est considérée endogène verticale ; alors que la génération de code et la rétro-conception sont des exemples de transformation exogène verticale.

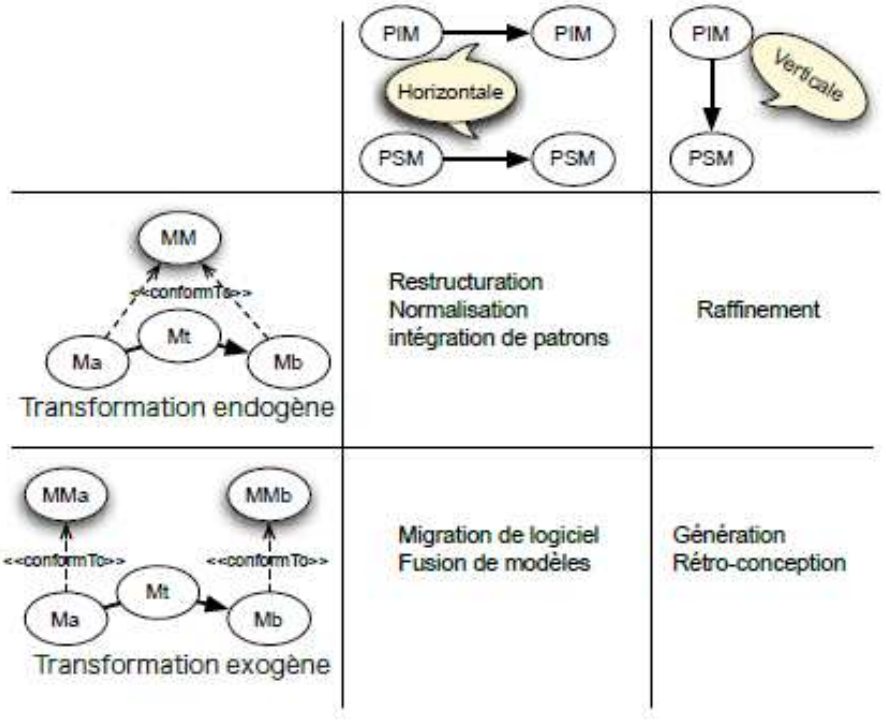

Figure 2.9 – Taxonomie des transformations de modèles [Combemale, 2008a]

Une autre répartition des approches de transformation de modèles peut être prise en compte, celle qui se scinde en approche de modèle-à-modèle (M2M) et approche de modèle-à-texte (M2T). L'approche M2M permet de transformer des modèles sources, conformes à ses méta-modèles, à d'autres modèles cibles, conformes à d'autres (ou les mêmes) méta-modèles. L'approche M2T est généralement utilisée pour la génération de code. Cette approche peut être considérée comme un cas particulier de l'approche M2M, du moment où on peut fournir un méta-modèle du langage de programmation cible [Czarnecki et Helsen, 2003].

2.3.5 Verrous scientifiques en IDM

Dans cette section, nous présentons quelques verrous scientifiques et technologiques qui semblent être liés à notre problématique de recherche. Le premier est celui de la construction de modèles selon un point de vue particulier (cf. section 2.3.5.1). Dans le cadre de ce travail de thèse, notre point de vue concerne la modélisation des règles d'exploitation ferroviaires en respectant les contraintes non-fonctionnelles de sécurité. Cette construction de modèles nécessite parfois une utilisation complémentaire de concepts d'autres domaines. Par conséquent, nous sommes confrontés à des problèmes d'hétérogénéité des modèles ou méta-modèles issus de chacun des domaines (cf. section 2.3.5.2). Face à cette hétérogénéité, nous nous contentons de la réutilisation et de l'adaptation des méta-modèles et des profils UML existants. Ces derniers, comme les DSLs (Domain Specific Languages) sont des solutions efficaces pour les problèmes liés à la modélisation et à la définition des méta-modèles (cf. section 2.3.5.3). Une fois le méta-modèle bien défini, son évolution par extension vers un autre méta-modèle peut influencer les modèles qui lui sont conformes (cf. section 2.3.5.4).

Remarque 2.1 *La plupart de ces verrous émanent du rapport de synthèse de l'AS CNRS sur le MDA [Bézivin* et al.*, 2004].*

2.3.5.1 La construction des modèles

L'ingénierie dirigée par les modèles propose une abstraction sous forme d'un ou plusieurs modèles d'un système réel pour éviter la complexité de sa réalité. Cette abstraction à travers un modèle peut être restreinte en fonction d'un point de vue particulier. Des contraintes non-fonctionnelles liées à un système peuvent alors s'ajouter pour définir des aspects de performance, de qualité, de sécurité, etc. Elles pourront faire l'objet de modèles spécifiques. L'objectif de modélisation devient ainsi un point central afin de pouvoir modéliser selon ce point de vue particulier. L'une des difficultés rencontrées est la construction des modèles : Quels sont les aspects à modéliser de cette réalité et jusqu'à quel niveau de détail désirons-nous modéliser cette réalité ? Cela nécessite le recours à plusieurs techniques complémentaires pour proposer des méthodologies selon la complexité du système ainsi que la finalité de cette modélisation [Bézivin *et al.*, 2004]. C'est notamment

le cas de nos travaux de recherche qui portent sur la sécurité des systèmes critiques ferroviaires et qui font appel aux techniques inhérentes à la sécurité des systèmes informatiques.

L'intégration de ces techniques est une tâche ardue nécessitant une approche pluridisciplinaire. La fédération des modèles dans une chaîne outillée permettant une modélisation multi-niveaux et multi-paradigmes est l'une des solutions proposées dans la littérature [Koudri *et al.*, 2013]. Cette dernière approche vise à fournir la possibilité de spécifier et d'analyser des systèmes complexes au travers de modèles hétérogènes issus de différents outils et vus comme un ensemble cohérent de points de vue. Ceci nous incite à attaquer un autre verrou de l'IDM qui est l'hétérogénéité des modèles et des méta-modèles.

2.3.5.2 L'hétérogénéité des modèles et des méta-modèles

La prolifération des méta-modèles pour décrire des modèles est à ce jour considérable [Bézivin *et al.*, 2004]. Plus la complexité du système et la complexité de la modélisation, parfois due à la diversité technologique, augmentent, plus l'hétérogénéité des méta-modèles et des modèles devient remarquable. Dans la littérature, des techniques rigoureuses face à l'hétérogénéité ont fait l'objet de certaines études de recherche. L'application des approches d'interopérabilité basées sur le MDA afin de faire fonctionner des modèles hétérogènes est l'une des techniques utilisées. [Chettaoui, 2008] propose une méthodologie de synchronisation pour l'interopérabilité entre outils de conception de produit basée sur le principe de l'IDM. D'autres travaux se sont focalisés sur l'approche de composition de méta-modèles ou modèles tels que l'approche ModelHel'X de composition de modèles hétérogènes pour la modélisation multi-paradigmes [Hardebolle, 2008].

Dans ce que nous avons réalisé, nous ne sommes pas allés jusqu'à la proposition d'une modélisation multi-paradigmes ou des mécanismes de composition faisant face à l'hétérogénéité. Nous avons procédé par réutilisation et adaptation des techniques existantes pour mener à bien l'activité de vérification et de validation des propriétés de sécurité dans le cadre de l'ingénierie dirigée par les modèles.

2.3.5.3 La définition des méta-modèles

Pour un même point de vue, plusieurs formes de modèles peuvent coexister : des modèles graphiques, des modèles textuels, des modèles formels, etc. Cette variété de modèles requiert la définition des concepts principaux de la problématique sous-jacente par des méta-modèles. Ces méta-modèles ont une influence sur la perception que possède le modeleur vis-à-vis du système qu'il doit décrire. Selon la finalité de la modélisation et la complexité du système à modéliser, le choix du méta-modèle varie. Pour cette raison, il doit être en adéquation avec celles-ci. Au delà du choix du méta-modèle, le choix du langage est important. Ceci dépend d'autres aspects de modélisation tels que la validation

de la conformité des modèles, le raisonnement sur les modèles et la possibilité d'exécution des modèles [Bézivin *et al.*, 2004]. Les bases théoriques du langage choisi peuvent parfois ne pas répondre à ces tâches ni à la finalité de la modélisation. Par exemple, un manque de rigueur dans la définition de ce langage peut mener à une définition partielle de la sémantique d'un modèle, ce qui est souvent le cas des notations graphiques. Ceci nous incite à utiliser, en plus des notations graphiques UML considérées comme semi-formelles, les notations formelles rigoureuses de la méthode B. Le chapitre suivant prend en charge la présentation de ces différentes notations ainsi que leur couplage.

Afin de rendre les méta-modèles plus riches et rigoureux, des DSMLs (Domain Specific Modeling Languages) apportent une solution ciblée et efficace pour des problèmes liés à la modélisation ou à la programmation, en séparant la sémantique du langage de sa syntaxe. Plutôt que d'être concentrés sur un problème technologique particulier tel que la programmation et l'échange de données ou de configuration, les DSLs peuvent représenter plus directement le domaine du problème qui est abordé [Cook, 2004].

Dans ce même contexte, lorsqu'il s'agit d'une modélisation UML, la modélisation des propriétés non-fonctionnelles d'un système ou métier est parfois délicate à réaliser. Le mécanisme d'extension, appelé profil UML a été proposé pour pallier sa rigidité. Un profil UML, standardisé par l'OMG, est un ensemble de techniques et de mécanismes permettant de définir des extensions d'UML afin de l'adapter à un domaine particulier. Cette adaptation est possible sur n'importe quel modèle UML sans remettre en cause le méta-modèle UML. Cette technique a été utilisée pour définir les politiques de contrôle d'accès RBAC et Or-Bac afin d'orienter la modélisation UML vers une modélisation basée sur des rôles, des permissions, etc (cf. section 2.2.2.2).

2.3.5.4 L'évolution des méta-modèles

Un autre défi important est lié à la préservation de la structure, des contraintes et de la sémantique. Suite à une mise à jour des outils vers une nouvelle version, ou bien un enrichissement de la sémantique du méta-modèle, du modèle qui lui était conforme par d'autres éléments de modélisation qui concernent un point de vue particulier du système, le méta-modèle évolue. Par conséquent, les modèles construits à partir de ce méta-modèle ne seront plus conformes à cette évolution. Dans ce cas, l'évolution de ces modèles est nécessaire afin d'être conformes au nouveau méta-modèle. [Sprinkle *et al.*, 2014] propose de représenter intuitivement les transformations évolutives. L'extension proposée est l'automatisation de ces transformations basée sur les changements du méta-modèle dans son évolution [Levendovszky *et al.*, 2014]. Un verrou de ce type est présent dans ce travail de thèse. On utilise un méta-modèle de RBAC (cf. chapitre 4), puis on l'étend vers un méta-modèle Or-Bac (cf. chapitre 5). Cette évolution, n'étant pas encore implémentée,

peut être rapportée aux perspectives de ces travaux de recherche.

2.4 Mise en application de l'IDM

Le modèle représente un point de vue particulier du système modélisé qui répond à un certain nombre de questions particulières. Ce principe existe depuis des dizaines d'années, avant même l'apparition des deux principes « tout est objet » et « tout est modèle ». En effet, selon [Minsky, 1965], « *pour un observateur A, M est un modèle d'un objet O, si M aide A à répondre aux questions qu'il se pose sur O* ». Par conséquent, afin de répondre aux questions qu'on se pose sur le système étudié, la modélisation nous semble indispensable. Dans le cas des systèmes critiques, la présence des aspects non-fonctionnels en plus des aspects fonctionnels complique la tâche de modélisation.

Plusieurs travaux de recherche académiques et industriels ont été entrepris avec succès au cours des dernières décennies autour de l'IDM. Dans ce qui suit, nous exposons quelques projets qui soulignent l'apport de l'IDM, aussi bien de point de vue théorique que pratique, dans l'industrie des systèmes critiques d'une façon générale et pour les systèmes ferroviaires en particulier.

2.4.1 IDM dans l'industrie des systèmes critiques

Les techniques issues de l'IDM font partie intégrante de la panoplie des approches à la disposition des industriels et influencent leurs pratiques en génie logiciel. Nous citons, dans cette section, quelques travaux de recherche qui exploitent l'IDM et les adaptent selon leurs besoins industriels afin de profiter des bonnes pratiques de développement logiciel d'envergure et de la synergie entre les travaux issus des différents domaines.

Dans le secteur aéronautique, les approches de MDA et MDT (Model Driven Testing) ont été intégrées et adaptées dans le processus du modèle en V pour le système de contrôle de trafic aérien [Carrozza *et al.*, 2013, Carrozza *et al.*, 2012]. Dans le secteur automobile, le projet VETESS [7] [Fondement *et al.*, 2010] s'inscrit dans le cadre de l'IDM pour la mise en œuvre automatique d'une série de tests de vérification concernant les composants des systèmes embarqués lors de la conception d'un véhicule. Quant au domaine médical, [Onisto *et al.*, 2014] montre l'intérêt de l'utilisation de l'IDM en termes de flexibilité et de modularité pour le développement d'une plate-forme d'échographie en temps réel. Les systèmes nucléaires, comme les autres systèmes critiques, bénéficient de l'intégration de l'IDM. Les travaux de thèse de Sannier [Sannier, 2013, Sannier et Baudry, 2012], dans

7. Vérification de systèmes embarqués VEhicules par génération automatique de TESts à partir des Spécifications

le cadre du projet CONNEXION [8], utilise l'IDM pour la formalisation du domaine par la définition d'un méta-modèle permettant une capitalisation et une vue globale à haut niveau d'un référentiel d'exigences.

Nous avons présenté quelques projets dans des domaines divers de l'industrie des systèmes critiques. Maintenant, nous nous focalisons sur les systèmes ferroviaires.

2.4.2 IDM pour les systèmes ferroviaires

La mise en application de l'IDM pour le développement dans un véritable contexte industriel tel que le ferroviaire a été soutenue par plusieurs projets. Dans cette section, nous détaillons le cycle de vie en V des deux projets récents CECRIS et OpenETCS dont les contributions dans le cadre de l'IDM sont intéressantes. Par la suite, nous exposons brièvement d'autres projets qui se focalisent sur l'utilisation des DSLs et des profils UML.

2.4.2.1 Le projet CECRIS

Le retour d'expérience du projet pilote mené dans Prolan Control Co, entreprise hongroise de fabrication de processus de contrôle et des systèmes de signalisation ferroviaire, a été introduit dans le partenariat industriel-académique au sein du projet « CErtification of CRItical Systems » (CECRIS). Ce dernier vise à introduire l'innovation dans le processus de développement de Prolan Block, un système de sécurité critique pour l'enclenchement ferroviaire certifié CENELEC EN50126, EN50128 et EN50129 SIL 4. Adapté au domaine d'application ferroviaire, le processus décrit dans [Scippacercola *et al.*, 2015] propose une méthodologie complète qui supporte la vérification et la validation (V&V) suivant le modèle conventionnel en V, tel qu'il est suggéré par les normes CENELEC. La partie gauche du modèle en V de la figure 2.10 concerne le processus de développement : la définition des exigences fonctionnelles et non-fonctionnelles, la conception du système, la conception des composants et l'implémentation. La partie droite, quant à elle, concerne les activités de V&V. Elle contient une phase de validation effectuée suite à la phase de vérification des composants et à la phase de vérification de l'intégration du système afin d'évaluer la conformité du produit par rapport aux exigences du système. Ce modèle est ainsi adapté aux principes de MDA par l'introduction des trois points de vues :

— CIM pour la spécification des exigences du système,
— PIM pour la conception du système et de ses composants, et
— PSM pour la phase d'implémentation.

Des phases intermédiaires, permettant le passage des différentes phases de développement aux activités de V&V, sont schématisées au milieu de la figure 2.10 :

8. COntrôle commande Nucléaire Numérique pour l'EXport et la rénovatION : https ://www.cluster-connexion.fr

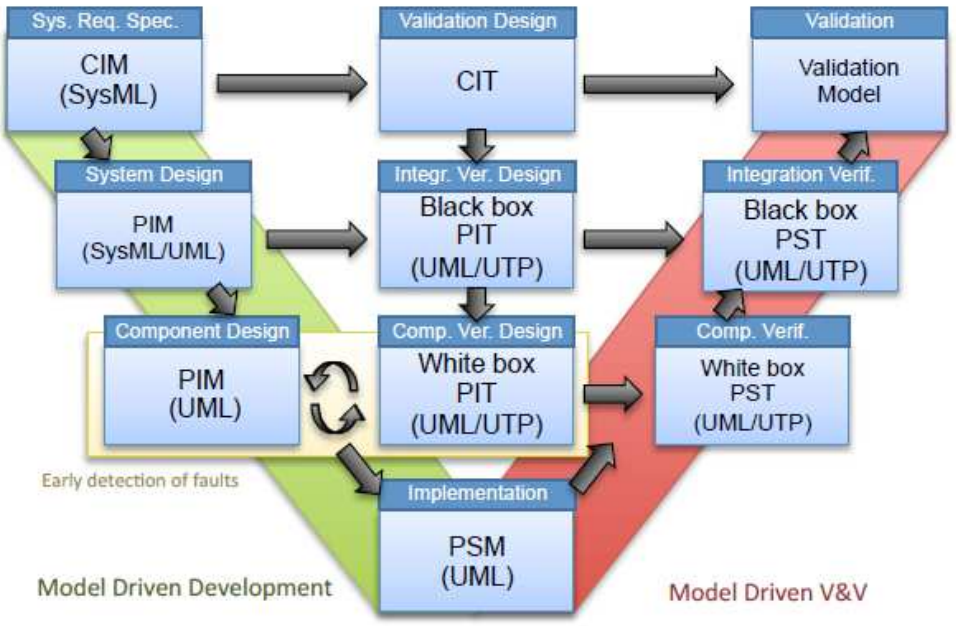

Figure 2.10 – Modèle en V proposé par [Scippacercola *et al.*, 2015]

— La conception de la validation définit un modèle du comportement des acteurs et de l'environnement CIT (Computation Independent Testing) qui exploite le modèle CIM.
— La conception de la vérification d'intégration définit un modèle de boîte noire BB-PIT (Black Box Platform Independent Testing) qui fournit des vues statiques et dynamiques des composants du système.
— La conception de la vérification des composants raffine la boîte noire par une boîte blanche WB-PIT (White Box Platform Independent Testing).

Les activités de V&V raffinent les modèles CIT et PITs qui envisagent de nouveaux détails découlant de la plate-forme cible, à partir de l'implémentation et du modèle PSM. En effet, les modèles BB-PIT et WB-PIT sont ensuite raffinés au cours de ces activités en exploitant de nouvelles connaissances de la plate-forme cible pour définir une nouvelle suite de test. Les nouveaux modèles spécifiques obtenus sont BB-PST (Black Box Platform Specific Testing) et WB-PST (White Box Platform Specific Testing). Les cas de test et les artefacts de V&V sont ensuite dérivés des PSTs par des transformations automatiques.

Ce projet propose un cycle en V adapté aux systèmes de signalisation ferroviaire et en particulier au niveau du système d'enclenchement ferroviaire, faisant partie des équipements « sol ». Cependant, les exigences de sécurité, motivation centrale de nos travaux de recherche, ne sont pas au centre de ce processus de développement basé sur les principes de MDA.

Le projet OpenETCS, quant à lui, se consacre à la partie « bord » des systèmes ferroviaires et intègre la sécurité dans son processus de développement.

2.4.2.2 Le projet OpenETCS

Il est aussi important de citer le projet OpenETCS (Open proofs for the European Train Control onboard System) qui a pour objectif de développer une chaîne d'outils complète de l'ensemble du processus de développement du système ERTMS/ETCS embarqué à bord, allant de la spécification et la modélisation jusqu'à la validation formelle et la génération de code. Pour ce faire, des standards ouverts ont été privilégiés sur tous les niveaux : les spécifications logicielle et matérielle, la définition des interfaces, les outils de conception, les procédures de vérification et de validation, etc. Le processus d'OpenETCS est subdivisé en deux sous-processus [Petit-Doche et Güdemann, 2013] : un processus de sécurité contenant toutes les contraintes logicielles et un processus qui ne tient pas compte de la sécurité contenant les activités de formalisation. Ce dernier est conçu pour obtenir un modèle exécutable en utilisant un générateur de code rapidement. Bien évidemment le code obtenu n'est ni sûr ni normalisé. En ce qui concerne le processus de sécurité d'OpenETCS, l'activité de formalisation est une phase incontournable pour fournir un modèle formel produisant un logiciel exécutable sécurisé. Sur la figure 2.11, les éléments d'entrée, qui sont l'ensemble des spécifications, sont en jaune, les activités de spécification et de conception sont en bleu, les activités de vérification et de validation sont en rouge et les activités de sécurité sont en vert. Le processus est subdivisé en quatre étapes dont les deux premières font partie de la phase système et les deux dernières de la phase logicielle :

— Une première étape (Step 1) d'analyse du système fournit un sous-système de spécification des exigences (SSRS - Sub-System Requirement Specification) qui définit le cadre de conception du système et sa structure (1a), complété par un API (Application Programming Interface) donnant les principaux interfaces du système et son interaction avec les artéfacts logiciels et matériels (1b). Dans la partie droite du processus un sous-système d'analyse des risques (SSHA - Sub-System Hazard Analyses) contient les analyses de sécurité de SSRS et permet la définition des propriétés de sécurité (1d).

— Une deuxième étape (Step 2) de modélisation semi-formelle décrit l'architecture du sous-système et les principales fonctions et attribue les exigences (2a). Le modèle fourni est complété par un modèle formel qui se focalise sur un sous-ensemble de fonctions ou de propriétés (2b).

— Une troisième étape (Step 3) de conception logicielle engendre deux approches en parallèle à partir du même modèle du sous-système : une approche permettant de compléter le modèle semi-formel constituant une branche fonctionnelle (3a) qui couvre autant que possible les exigences des SSRS ; et une approche se basant sur des outils et des méthodes permettant de compléter le modèle formel constituant

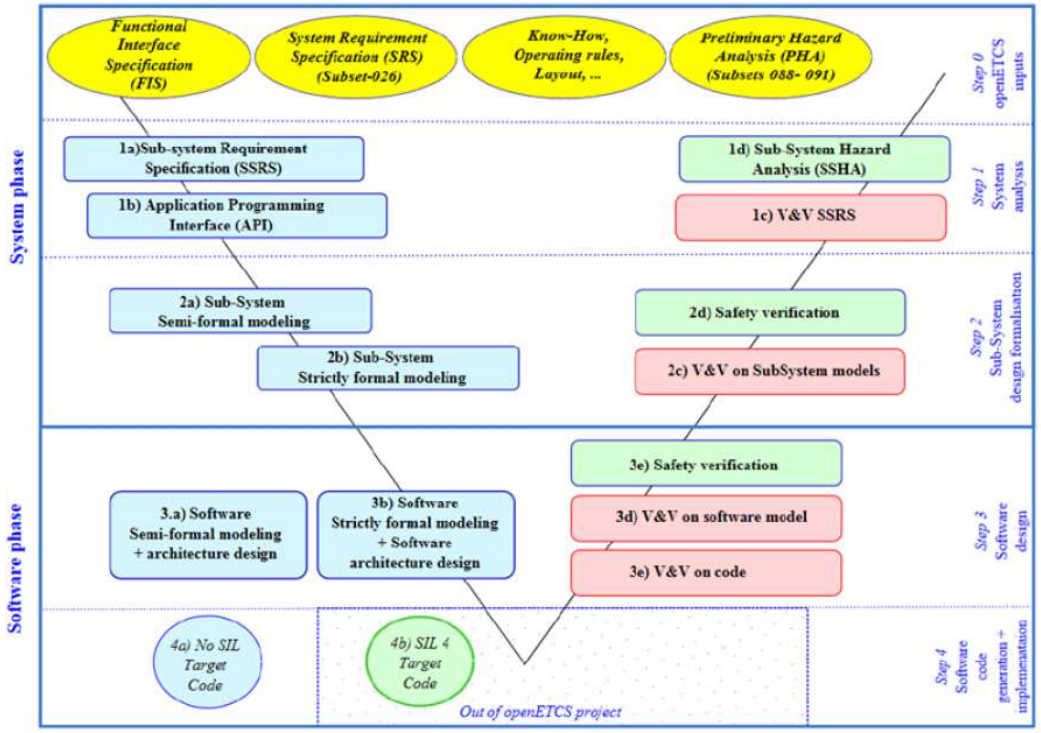

Figure 2.11 – Le processus du projet OpenETCS décrit dans [Lanusse *et al.*, 2014]

une branche fonctionnelle de sécurité (3b) qui doit être évalué sur un sous-ensemble des exigences des SSRS.

— Une quatrième étape (Step 4) de génération de code aboutit à deux types de code selon l'approche utilisée : un code exécutable non SIL (4a) généré à partir du modèle fonctionnel qui ne contient pas les contraintes spécifiques sur les moyens et les outils employés ; et un code SIL4 (4b) généré à partir du modèle fonctionnel de sécurité. Ce dernier n'est pas fourni par le projet OpenETCS parce que les activités de sécurité n'ont pas été menées dans l'ensemble du champ d'application du sous-système de l'unité à bord. En outre, les éléments de la plate-forme cible ne sont pas fournis.

Dans le cadre de ce projet, les travaux de [Peleska *et al.*, 2012] introduisent un approche OpenETCS dirigée par les modèles. Dans ce contexte, un DSL est proposé dans le but d'assurer le confinement des risques de sécurité.

2.4.2.3 Autres projets

Des approches dirigées par les modèles pour les activités de V&V ont été proposées aussi bien dans la communauté de recherche que par les industriels. Les techniques dirigées par les modèles ainsi utilisées sont principalement les profils et les DSLs. Le processus proposé par [Marrone *et al.*, 2014], dans le cadre du projet européen CRYSTAL [9], se base sur

9. CRitical sYSTem engineering AcceLeration

une approche de transformation de modèles en utilisant un profil UML dans le cadre des activités de V&V. Ce profil dépend des profils MARTE (Modeling and Analysis of Real-Time and Embedded Systems), DAM (Dependability Analysis and modeling) et UTP (UML Testing Profile). Il est ensuite spécialisé par des concepts et des caractéristiques spécifiques au domaine ferroviaire (le profil UML d'ERTMS/ETCS).

D'autres travaux de recherche se sont focalisés sur l'utilisation des DSLs pour les systèmes ferroviaires. Dans le cadre du projet danois RobustRailS [10] qui vise à établir une méthode holistique pour le développement et la vérification des nouveaux systèmes d'enclenchement sous ETCS niveau 2 [Vu *et al.*, 2014], un DSL est conçu pour la description des systèmes d'enclenchement compatibles avec l'ETCS niveau 2. La syntaxe abstraite et la sémantique statique de ce DSL sont définies avec le langage de spécification RAISE (RSL, RAISE Specification Language).

L'objectif de l'utilisation de ces techniques par les projets susmentionnés est la définition d'un processus unifié qui permet la génération automatique des modèles destinés aux activités de V&V. C'est le cas notamment pour le projet européen INESS [11], financé par le programme FP7 de l'union européenne dont la finalité est de fournir un système commun et intégré de signalisation ferroviaire en Europe [Dos Santos *et al.*, 2010]. L'intégration, la facilité d'utilisation et la réduction des activités manuelles par des transformations de modèle (semi-)automatiques sont les principaux avantages de l'IDM pour de tels projets.

Sur le plan national, des projets nationaux de l'agence nationale de recherche (ANR) ont été réalisés autour de l'IDM. Le projet IMOFIS [12] introduit un langage de modélisation SOPHIA [Cancila *et al.*, 2009b] et son méta-modèle dans une approche basée sur les modèles pour l'ingénierie de la sécurité des systèmes ferroviaires. Les concepts fondamentaux de ce langage sont basés sur une ontologie d'Alstom et définit sous forme de profil UML pour la sécurité qui peut être intégré avec le profil SysML (System Modeling Language) [13]. Dans le cadre du même projet, les auteurs [Cancila *et al.*, 2009a] ont abordé une étude de sûreté de fonctionnement dans le cadre de l'IDM et ont proposé une piste vers la construction d'un profil UML SdF à partir de la composition de certains paquetages précis de MARTE, SysML et d'autres profils métier dédiés.

10. Robustness in Railway OperationS
11. INtegrated European Signalling System
12. Ingénierie des MOdèles de FonctIons Sécuritaires
13. SysML est un langage de modélisation spécifique au domaine de l'ingénierie système. SysML se définit comme une extension d'un sous ensemble d'UML via l'utilisation du mécanisme de profil UML.

2.4.3 Synthèse sur la mise en application de l'IDM

Les travaux de la littérature relatifs à l'IDM dans différents domaines et particulièrement pour les systèmes de transport ferroviaire révèlent l'intérêt de l'intégration de l'IDM dans la définition du processus de développement logiciel. D'une part, les concepts de modélisation, de méta-modélisation et de transformation de modèles de l'IDM offrent un cadre méthodologique et technologique unificateur envers les différentes activités du cycle de développement. Ceci est fait en choisissant l'espace technique et le formalisme adéquat à chacune d'elles grâce à l'utilisation intensive des modèles et des transformations entre les modèles [Favre *et al.*, 2006]. D'autre part, l'utilisation des langages et modèles spécifiques aux domaines DSL (ou encore DSML) et des profils UML permettent de prendre en considération les spécificités et les contraintes extra-fonctionnelles qui varient d'un domaine à un autre.

De point de vue scientifique, certains travaux de recherche, parmi ceux qui sont cités, considèrent l'étude de sûreté de fonctionnement et de la sécurité au cœur du processus de développement. Sur le plan européen, le projet OpenETCS fournit deux sous-processus : l'un fonctionnel et l'autre fonctionnel de sécurité. Tandis que le projet IMOFIS, sur le plan national, intègre l'étude de la sûreté de fonctionnement dans l'ingénierie des systèmes ferroviaires. Dans le but de faciliter l'activité de modélisation de ces systèmes, des DSLs et des profils tels que ceux qui sont liés à la sécurité, au test (UTP), à la modélisation et l'analyse des systèmes temps-réel embarqués (MARTE), sont appliqués. Dans ce contexte scientifique, nous nous focalisons sur l'étude de la sécurité dans un processus de développement centré sur les modèles qui respecte les principes de l'IDM. Nous avons alors utilisé un profil UML de sécurité permettant de souligner les spécificités du système.

De point de vue industriel, la plupart de ces travaux ne se focalisent que sur une partie ou bien un sous-système du système ferroviaire : le système de signalisation et d'enclenchement pour la partie « sol » et le système embarqué à bord pour la partie « bord ». En outre, les systèmes ferroviaires étudiés sont soit des systèmes nationaux soit des systèmes européens sous les normes d'ERTMS. Dans ce contexte industriel, nous nous sommes intéressés, à la cohérence de la spécification et des règles d'exploitation au regard des exigences de sécurité. Cette problématique, pourtant cruciale pour un fonctionnement performant et en toute sécurité, est très peu abordée dans la littérature scientifique. En effet, la gestion de la signalisation dans le système ERTMS est régie par les règles nationales propres à chaque pays et non par des règles globales. Ceci rend très difficile l'évaluation du système en termes d'exigences de sécurité. Notre objectif est de fournir une analyse de ce que produit la rencontre entre la spécification européenne et les règles opérationnelles nationales.

2.5 Synthèse

Dans nos travaux de recherche, nous considérons les exigences de sécurité au cœur du processus de développement des systèmes industriels critiques ferroviaires. C'est la raison pour laquelle nous avons abordé, dans une première partie de notre étude de l'art, l'ingénierie des exigences faisant partie de l'ingénierie système. Nous nous focalisons sur les phases de modélisation et de validation des exigences de sécurité. Afin de profiter des concepts de modélisation, de méta-modélisation et de transformation de modèles, l'ingénierie dirigée par les modèles vient compléter l'ingénierie des exigences. En effet, les approches basées sur les modèles sont vues comme des méthodologies à forte valeur ajoutée permettant de définir un niveau d'abstraction plus élevé du système et d'automatiser le processus de développement. L'ingénierie dirigée par les modèles est alors abordée dans une deuxième partie de cet état de l'art. Nous avons alors présenté les concepts de base de l'IDM, ses verrous scientifiques liés à notre problématique de recherche et sa mise en application pour l'industrie des systèmes critiques d'une façon générale et pour les systèmes ferroviaires en particulier.

Dans le présent travail de thèse, nous nous intéressons à la modélisation des règles d'exploitation ferroviaires, en tenant compte des exigences de sécurité ferroviaire. Notre approche se situe au niveau PIM selon les principes de l'IDM. Plusieurs notations, dans la littérature, sont utilisées pour la modélisation. Ces notations diffèrent selon leur niveau de précision, d'ambiguïté et de facilité d'utilisation. Nous entamons le chapitre suivant par la description des différentes notations : informelles, semi-formelles et formelles. Dans l'optique de mener à bien la phase de spécification et les activités de validation et de vérification, nous avons opté pour une approche de couplage de notations semi-formelles et formelles UML/B afin de tirer profit de chacune d'entre elles.

Chapitre 3

Couplage de spécifications en UML et B

Sommaire

3.1 Introduction

Le processus de développement logiciel d'un système critique tel qu'un système ferroviaire repose généralement sur des documents informels contenant des exigences, plusieurs centaines de pages en langage naturel, sous forme de cahier des charges. Force est de constater que les langages informels qui décrivent ces documents ne disposent pas (ou peu) de moyen pour la formalisation et surtout la gestion des exigences. Il s'avère alors nécessaire de comprendre, d'interpréter et de formaliser ces documents afin de vérifier et de valider les exigences de départ. Le succès de toutes ces étapes dépend en grande partie de la phase de spécification. Cette phase a pour vocation de spécifier d'une manière claire, précise et non-ambiguë les caractéristiques du système à développer. Cependant, les notations informelles sont sujettes à des interprétations multiples. Ce constat illustre la nécessité de passer des notations informelles aux notations semi-formelles ou encore formelles en tirant profit des avantages de chacune d'elles. En effet, les notations semi-formelles, par leurs aspects graphiques, permettent une vue synthétique et intuitive du système. Quant aux notations formelles, la précision et la concision sont leurs principaux points forts. Ces dernières permettent la vérification et la validation formelles par des techniques formelles telles que la preuve, le model checking et l'animation. Cette vérification est cruciale, non seulement du point de vue industriel, mais surtout du point de vue de la sécurité. La sécurité est l'une des exigences primordiales pour les systèmes ferroviaires (décrite dans le chapitre 2). À cet égard, le couplage des notations semi-formelles et formelles est l'objet d'étude de plusieurs travaux de recherche dans l'optique d'exploiter leur complémentarité.

Pour commencer, nous jugeons indispensable d'identifier, dans le processus de développement, les deux étapes suivantes : la spécification d'une part, et les activités de validation et de vérification d'autre part. La section 3.2 définit la spécification, présente ses objectifs et explicite les différents types de langage de spécifications : naturels, semi-formels et formels. La section 3.3, quant à elle, définit la validation et la vérification et passe en revue les techniques formelles de validation et de vérification que nous avons utilisées dans nos contributions appliquées aux règles d'exploitation ferroviaires. Une fois la spécification, la validation et la vérification formelles définies, nous passons au couplage des notations semi-formelles et des notations formelles. En particulier, les notations UML et B sont choisies parmi les approches existantes dans la littérature. La section 3.5 définit UML, sa sémantique, ses vues et ses diagrammes, tandis que la section 3.6 présente la méthode B. Enfin, nous achevons ce chapitre par notre approche basée sur les notations UML et B dans le cadre du projet *PERFECT*.

3.2 Spécification

La vérification d'un cahier des charges, souvent complexe et difficile à comprendre et à contrôler, est une tâche difficile. La difficulté majeure se manifeste surtout au mo-

ment de la compréhension du problème. Cette difficulté se reflète notamment au stade du test du logiciel final si un choix qui aurait dû être réglé dans la phase d'analyse n'a pas été bien défini. Ceci peut perturber, voire mettre en péril la réalisation du processus de développement logiciel. La formalisation d'un choix fait typiquement partie de la tâche de spécification.

3.2.1 Définition et objectifs

La spécification, en génie logiciel, est l'étape qui consiste à décrire ce que le système doit faire. En d'autres termes, elle consiste à définir les exigences du système. Elle génère un ensemble de documents qui expriment les propriétés du système dans un *langage de spécification*. Ces documents peuvent être utilisés dans l'optique d'une modélisation, d'une vérification et/ou d'une validation :

— **Spécifier pour modéliser :** Étant donné qu'un modèle est une abstraction du système concret, la spécification permet d'énoncer les propriétés sur le système avant même sa conception et son développement et d'argumenter sur la cohérence de chaque composant du système.

— **Spécifier pour valider :** La spécification permet de poser les bonnes questions afin de s'assurer que le bon produit a été construit et que les composants du système sont conformes au cahier des charges.

— **Spécifier pour vérifier :** La spécification permet de s'assurer qu'un produit a été construit correctement et de vérifier à chaque étape du développement que la réalisation du système respecte bien les attentes initiales.

L'objectif de la spécification dépend également du type (ou classe) du langage de spécification détaillé dans la section suivante.

3.2.2 Différents types de langage de spécification

Avant de détailler les différents types de langage de spécification, nous exposons les définitions des termes *informel*, *semi-formel* et *formel*, issues de [ISO/IEC 15408-1, Norme Internationale, 1999] :

— ***Informel*** *(Informal) — Qui est exprimé à l'aide d'un langage naturel.*

— ***Semi-formel*** *(Semi-formal) — Qui est exprimé à l'aide d'un langage syntaxique restreint muni d'une sémantique définie.*

— ***Formel*** *(Formal) — Qui est exprimé dans un langage syntaxique restreint doté d'une sémantique définie basée sur des concepts mathématiques bien établis.*

Comme un langage informel est basé sur un langage naturel, nous considérons dans ce qui suit trois types de langage de spécification : les langages naturels, les langages semi-formels et les langages formels.

3.2.2.1 Langages naturels

Le langage naturel est le langage le plus facilement utilisable et compréhensible par les humains. Il représente le moyen le plus simple et direct de communication et d'échange entre les différents acteurs, qui sont dans notre contexte les experts du domaine, les concepteurs, les développeurs, . . . , pour exprimer ses besoins ou la perception d'un problème [Sadoun, 2014]. Néanmoins, ni sa syntaxe, ni sa sémantique ne sont parfaitement définies. En effet, d'une même idée plusieurs formulations sont possibles et d'une même formulation plusieurs sens peuvent résulter. Le fait que plusieurs interprétations peuvent découler d'une même formulation va engendrer des problèmes de compréhension.

Les spécifications écrites en langage naturel sont donc ambiguës et non précises à cause de ces multiples interprétations. En outre, dans certains cas, des informations pertinentes sont implicitement transmises nécessitant des connaissances extérieures. Ces dernières dépendent souvent du domaine et du contexte particulier et demandent un effort d'interprétation. Malgré cet effort, les spécifications en langage naturel peuvent aboutir à une mauvaise interprétation [Sadoun, 2014]. Par conséquent, ces spécifications ne sont pas adaptées pour des vérifications automatiques d'une manière directe.

3.2.2.2 Langages semi-formels

Les langages semi-formels sont généralement basés sur des notations graphiques et/ou textuelles standardisées qui ont une syntaxe précise, mais une sémantique assez faible.

Les langages semi-formels favorisent l'expressivité au cours de l'activité de modélisation tout en réduisant l'ambiguïté du langage naturel. [Héon *et al.*, 2010] définit un langage semi-formel comme « *un langage artificiel dont la sémantique comporte des éléments d'ambiguïté* ».

Les notations graphiques des langages semi-formels représentent un bon vecteur de communication non seulement entre concepteurs et utilisateurs, mais aussi entre concepteurs et développeurs. Elles permettent une vue synthétique, structurante et intuitive du système [Dupuy, 2000a]. En revanche, le manque d'une sémantique précise limite l'emploi des techniques de validation et de vérification. Il n'est pas possible de raisonner ou d'analyser formellement sur le système étudié.

Parmi les langages semi-formels les plus connus, nous pouvons citer UML (Unified Modeling Language) que nous aborderons dans la section 3.5, et son extension SysML (System Modeling Language). Ce sont des langages appartenant à la classe des méthodes orientées objet. D'autres langages semi-formels existent tels que le langage SADT (Structured Analysis and design Technics) et le langage SA/SD (Structured Analysis, Structured Design) appartenant à la classe des méthodes fonctionnelles.

3.2.2.3 Langages formels et méthodes formelles

Un langage formel dispose d'une syntaxe et d'une sémantique bien définies, par opposition au langage naturel (informel) qui peut donner lieu à plusieurs interprétations et au langage semi-formel qui dispose d'une syntaxe précise mais pas d'une sémantique bien définie. Aucune divergence d'interprétation n'est envisagée en utilisant un langage formel.

Il permet par ailleurs de mettre en évidence des problèmes dès le début. En effet, il permet de prouver formellement des propriétés sur le système dès sa spécification. Il n'est pas nécessaire d'attendre la phase de conception où les problèmes d'optimisation peuvent être le point d'intérêt des concepteurs, ni la phase de l'implémentation pour réaliser des tests exhaustifs.

Le choix d'une technique formelle dépend de l'adéquation de cette dernière au besoin, au domaine d'application et à la nature du système développé. Le rapport de synthèse du Groupe de travail « Méthodes formelles en logiciel » [OFTA, 1997] offre un ensemble de critères à prendre en compte pour le choix des langages de spécification. Le degré de généralité du langage est l'un de ces critères. En effet, il existe des langages avec un vocabulaire spécifique destiné à un domaine particulier, et d'autres qui sont généraux avec un nombre réduit de notions, mais assez abstraits pour traiter un grand nombre d'applications. CCS (Calculus of Communicating Systems) et CSP (Communicating Sequential Processes) sont des exemples de langages généraux qui sont basés sur les concepts de processus et d'opération de composition. Deux autres types de langages ont été évoqués dans ce rapport de synthèse à savoir les langages orientés modèles et les langages de description de comportements :

- « *les langages orientés modèles, tels que Z, VDM, B, Coq, Unity et TLA, qui sont fondés sur l'utilisation de calculs formels : calcul classique des prédicats pour les uns, logiques plus spécifiques, principalement logiques temporelles, pour d'autres. Ces langages permettent de décrire des comportements et des propriétés générales telles que la terminaison, l'absence de blocage, l'équité ou l'exclusion mutuelle.*
- *les langages de description de comportements ; ils permettent de définir le comportement d'un système par composition des comportements de ses constituants. On distingue parmi eux les algèbres de processus, dont les plus connus sont CCS et CSP, les langages de description de protocoles tels que SDL, Lotos et Estelle, les formalismes à base d'automates comme Statecharts, Grafcet et les réseaux de Petri, ainsi que les langages synchrones tels que Lustre et Esterel. Leur sémantique s'exprime souvent en termes de systèmes de transitions* ».

Remarque 3.1 *Dans la section 3.4.2, nous donnerons les critères de choix que nous avons adopté pour sélectionner le langage formel le plus approprié afin de répondre à nos besoins de vérification et de validation des règles d'exploitation ferroviaire au regard des exigences de sécurité.*

Une méthode formelle se caractérise par un langage de modélisation et une démarche de développement du modèle. Le langage de la méthode formelle spécifie, par ses notations mathématiques, une syntaxe rigoureusement définie par une sémantique précise. Cette sémantique permet d'interpréter ce que signifie chaque symbole ou notation de la syntaxe utilisée. Autrement dit, la sémantique est la définition de ce que chaque expression possède comme signification mathématique précise de ce qui est décrit par la syntaxe. La démarche, quant à elle, définit le processus de construction et de validation du modèle [OFTA, 1997]. D'après [Hinchey et Bowen, 1995], une méthode formelle est « *un ensemble d'outils et de notations (avec une sémantique formelle), utilisés pour spécifier de manière non ambiguë les exigences d'un système informatique et supportant les preuves de propriétés sur cette spécification et les preuves de corrections d'une implémentation éventuelle en ce qui concerne la spécification* ». De là, deux activités fondamentales permettent la mise en œuvre d'une méthode formelle, à savoir, la spécification formelle et la vérification formelle [Mariano, 1997] :

*La **spécification formelle** est l'expression, au moyen d'un langage formel, d'un résultat attendu.*

*La **vérification formelle** est la production d'une preuve démontrant que le produit respecte effectivement la spécification formelle dont il « prétend » être la réalisation.*

Les méthodes formelles, introduites dans certaines parties du cycle de vie classique du logiciel, apportent non seulement des réponses aux problèmes de qualité et de fiabilité, mais aussi aux problèmes de productivité et de maintenance [OFTA, 1997]. Lorsqu'il s'agit des logiciels critiques et de sécurité, son utilisation est alors incontournable en raison des exigences accrues en qualité de certains systèmes dont la garantie de la sûreté est l'une des propriétés primordiales. Ceci ne peut être révélé que par les activités de vérification et de validation formelles, détaillées dans la section suivante.

3.3 Vérification et validation

La phase de spécification est une phase primordiale, sans elle, les activités de vérification/validation (V&V) ne peuvent pas avoir lieu. Le choix des techniques de V&V est alors inhérent au choix du langage de spécification. Dans cette section, nous commençons par une définition de chacune de ces deux activités et nous poursuivons par une description des techniques choisies parmi les techniques existantes.

3.3.1 Définitions

Selon les normes ISO9000 :2005, la vérification est « *la confirmation par des preuves tangibles que les exigences spécifiées ont été satisfaites* ». Alors que la validation est « *la confirmation par des preuves tangibles que les exigences pour une utilisation spécifique ou une application prévue ont été satisfaites* ». En d'autres termes, la vérification est

une réponse à la question suivante : « Avons-nous bien construit le produit ? », tandis que la validation est une réponse à la question suivante : « Avons-nous construit le bon produit ? »

3.3.2 Techniques de vérification et de validation

Les phases de vérification et de validation (V&V) formelles sont cruciales pour les systèmes critiques. Les méthodes formelles peuvent apporter leur contribution par le prototypage rapide, par la preuve de propriétés, ou encore par les tests [OFTA, 1997] :

— Le prototypage rapide, via la simulation ou l'animation, permet de valider la spécification ou le prototype vis à vis d'un comportement attendu.

— Les techniques de preuve, par ce qu'on appelle le prouveur ou le contrôleur de théorèmes, ou les techniques de vérification de modèles à travers les approches par *model checking* sont des techniques de vérification formelle qui ont donné des résultats probants sur des applications industrielles significatives telles que les applications ferroviaires.

— Le test permet de révéler la présence d'erreurs dans l'exécution du système à l'aide d'un jeux de tests générés (automatiquement) à partir de la spécification formelle.

Dans le cadre de nos travaux, les techniques de V&V qui nous intéressent sont l'animation, la preuve et le model checking. L'utilisation de scénarios de tests est souvent un sujet de critique dans la littérature car généralement c'est une technique non exhaustive qui peut provoquer une détection tardive des erreurs et ainsi un coût souvent élevé pour la correction des erreurs détectées tardivement. Cette technique est souvent utilisée en conjonction avec les techniques de *model checking* et/ou la preuve. [Abrial, 2003] fait une comparaison entre la preuve et le test comme méthodes de validation de programme et montre comment l'utilisation de chacune de ces techniques influence les démarches de validation du système par l'analyste. Comme le test, il est recommandé d'utiliser la technique de validation par animation en conjonction avec d'autres techniques plus performantes de *model checking* et/ou de la preuve. En effet, l'animation n'est pas considérée comme une alternative pour les techniques de validation formelle « totale » [Vermesan et Coenen, 1999]. Dans cette thèse, nous avons utilisé cette technique en conjonction avec la preuve. Le *model checking*, quant à lui, est utilisé sans pouvoir aboutir aux résultats souhaitables. Ceci est justifié dans la section 4.6.1 du chapitre 4.

Nous passons en revue les trois techniques de *model checking*, de preuve et d'animation dans les sous-sections suivantes.

3.3.2.1 *Model checking*

Le model checking est destiné à la vérification formelle de modèles comportementaux de systèmes modélisés par rapport aux propriétés qui sont attendues sur ces modèles. Cette technique se fonde sur la construction d'un modèle (automate à états ou machine à états)

généralement fini qui décrit les états possibles, un état initial et les transitions d'état. Il s'établit par énumération exhaustive des états possibles (visités) à partir de l'état initial.

On pourra par exemple vérifier sur le système des propriétés de sécurité (voir chapitre 2) ou des propriétés de vivacité. Ces dernières expriment que « *quelque chose de bon se produit immanquablement au cours de l'exécution* » [Boulanger, 2012].

Étant donné une propriété du système à vérifier sur une machine, la technique de model checking explore l'ensemble des états atteignables par cette machine afin de vérifier que cette propriété est bien satisfaite. Deux cas se présentent : soit la propriété est vérifiée et maintenue par le modèle, soit une séquence de transitions d'état menant à la violation de la propriété est générée en guise de contre-exemple. Ceci montre que la propriété n'est pas maintenue par le modèle.

3.3.2.2 Preuve

La technique de vérification et de validation par preuve met en œuvre un système d'inférence. En effet, elle s'appuie sur un ensemble bien défini et formalisé de règles de raisonnement (modus ponens, récurrence, substitution, réécriture, etc.).

Il existe trois types d'outils support pour la preuve formelle, à savoir le vérificateur de preuves (*proof checker*), le prouveur de théorème interactif (*interactive theorem prover*) et le prouveur de théorème automatique (*automatic theorem prover*) [Haxthausen, 2010] :

- Le vérificateur de théorèmes permet de vérifier automatiquement si une preuve donnée est en fait une preuve correcte d'un théorème donné.
- Le prouveur de théorème interactif permet de construire interactivement une preuve correcte.
- Le prouveur de théorème automatique permet de chercher automatiquement une preuve pour un théorème donné.

Les outils de raisonnement automatiques pour les méthodes formelles combinent généralement la technique de *theorem proving* avec la technique de *proof checking* en fournissant des prouveurs de théorème semi-automatiques ou partiellement interactifs. Par conséquent, le terme prouveur de théorème (*theorem prover*) est généralement utilisé pour couvrir toute forme de déduction (partiellement) automatique [Vermesan et Coenen, 1999]. En guise d'exemple, l'*Atelier B* possède un prouveur de théorème automatique et un prouveur de théorème interactif.

L'objectif du prouveur de théorème est de construire une preuve mathématique pour un énoncé mathématique dans le but de démontrer qu'il est vrai. Si cet énoncé est démontré par la preuve, l'énoncé est alors vrai et il est considéré comme un théorème. Sinon, si une preuve est introuvable, on ne peut pas conclure que l'énoncé est faux. En effet, ce pourrait être faux, comme ce pourrait être vrai, mais la personne ou l'outil de preuve utilisé n'aboutit pas à trouver la preuve convenable pour démontrer que l'énoncé est vrai. Une fois le théorème démontré, il vaut pour la totalité du modèle.

Les preuves mathématiques peuvent être classées, selon leur rigueur, en preuves semi-formelles et preuves formelles. Les preuves semi-formelles sont construites d'un mélange de formules mathématiques et de langage naturel, utilisé le plus souvent dans les livres mathématiques. Les preuves formelles, celles qui nous intéressent, n'utilisent pas le langage naturel. Elles sont exprimées dans un langage symbolique ayant une syntaxe précise. En effet, une preuve mathématique dans une certaine logique mathématique consiste en une séquence d'étapes pour l'argumentation. Une étape de l'argumentation contient certaines prémisses (énoncés connus pour être vrais) et une conclusion (un nouvel énoncé) qui peut être tirée à partir de ces prémisses. Cette conclusion peut être utilisée ensuite comme prémisse à l'étape suivante et ainsi de suite, jusqu'à l'obtention d'une conclusion qui est l'énoncé à prouver [Haxthausen, 2010].

En vue d'une comparaison entre le model checking et la preuve, d'une part, le model checking est plus facile et plus rapide que la preuve de théorème. D'autre part, le prouveur de théorème ne se repose pas forcément sur des systèmes finis et décidables, contrairement au *model checker*. En effet, le prouveur de théorème est applicable dans certains cas où le model checking ne peut pas l'être à cause du problème d'explosion de l'espace d'état ou bien dans le cas où le système étudié ne peut pas être formalisé en modèle à états fini [Haxthausen, 2010].

3.3.2.3 Animation

L'animation est une technique de validation pour un prototypage rapide. Elle permet de visualiser certains scénarios formels du système et ainsi de valider le comportement dynamique du système. En effet, elle permet de démontrer, de manière précoce, l'absence (ou la présence) d'un comportement indésirable dans les scénarios utilisés. Dans l'absolu, elle ne garantit pas la correction du système, néanmoins elle augmente la confiance des utilisateurs envers leurs spécifications formelles.

L'animation n'est pas une alternative à la validation formelle et ne peut remplacer la preuve. Ces deux techniques ont un rôle important complémentaire [Vermesan et Coenen, 1999] et peuvent être utilisées en conjonction [Hinchey et Bowen, 1995]. En fait, l'animation peut être utilisée en tant que méthode de validation fournissant une vue rapide de l'exécution du modèle, complétée par la preuve pour une vérification approfondie. Elle permet de détecter et de localiser les éventuels problèmes dans un modèle. Certains outils d'animation tels que ProB (voir la section 3.6.6) supportent l'analyse des propriétés de vivacité et détectent les problèmes de blocage (*deadlock*) [Hallerstede *et al.*, 2013].

Vérifier et valider des systèmes critiques nécessite de raisonner sur les scénarios d'utilisation du système et les exigences qui leurs sont relatives. Ces scénarios doivent être modélisés d'une façon structurée et intuitive afin d'offrir une représentation facilitant la compréhension et la communication entre les différentes parties prenantes. Les notations semi-formelles, et surtout graphiques, ont la vocation de garantir une telle représentation.

Cependant, l'utilisation d'une notation semi-formelle dans une optique de vérification et de validation est depuis longtemps critiquée. En effet, on reproche aux notations semi-formelles leur ambiguïté et leur sémantique trop faible pour un raisonnement formel. Par conséquent, un couplage des spécifications semi-formelles et des spécifications formelles semble être la solution la plus adaptée pour profiter des avantages de la modélisation semi-formelle telle que celle d'UML et du raisonnement formel par vérification et validation formelles tel que celui de la méthode B. Les principes de couplage des notations semi-formelles et des notations formelles d'une façon générale et d'UML et B en particulier sont abordés dans la section suivante.

3.4 Du semi-formel au formel

La problématique d'intégration des spécifications est l'intérêt de nombreuses recherches depuis des dizaines d'années. La première tentative d'intégration des spécifications de [Fraser *et al.*, 1991] avait la vocation de combler le fossé entre les spécifications informelles SA (Sructured Analysis) et les spécifications formelles VDM (Vienna Development Method).

3.4.1 Principes de couplage des notations

Le couplage des spécifications semi-formelle et formelle est motivé par la complémentarité entre la structuration et la précision dans un processus de développement logiciel. En effet, les modèles semi-formels et les modèles formels se côtoient souvent dans la même approche grâce à cette complémentarité. Chacun de ces deux types de modèles peut pallier les inconvénients de l'autre.

Les modèles semi-formels sont connus par leur représentation intuitive, structurante et synthétique. Ils sont aussi connus par leur facilité de compréhension par les concepteurs ainsi que leur facilité de communication avec les développeurs et les utilisateurs. En revanche, ces modèles peuvent conduire à l'ambiguïté par leur sémantique qui n'est pas parfaitement définie.

Les modèles formels, quant à eux, ont une sémantique rigoureusement définie et ne laissent aucune possibilité de divergence dans leur interprétation. La précision et la concision sont facilement atteintes grâce à ses notations formelles. En outre, ils facilitent l'utilisation des techniques de vérification et de validation formelles (souvent d'une façon automatique). Néanmoins, ces modèles demandent souvent un effort supplémentaire pour une expertise plus poussée et nécessitent un bon niveau principalement en mathématique.

3.4.2 Choix des notations

Dans le cadre d'un processus de modélisation, le choix du langage UML s'impose de lui même. En effet, c'est un langage de modélisation universel qui est devenu une référence incontournable pour le domaine de génie logiciel après avoir été standardisé par l'OMG.

UML, par ses différentes vues et ses différents diagrammes, permet de modéliser à la fois la structure et le comportement du système. Par ailleurs, ce standard contribue notamment à l'IDM et définit des mécanismes d'extensions par les profils afin de personnaliser la modélisation à des fins spécifiques. Les langages fonctionnels, quant à eux, ne permettent ni la modélisation de l'aspect dynamique du système ni les mécanismes d'extensions. La section 3.5 présente UML en détail.

La sélection d'un langage formel n'est pas une tâche aisée compte tenu de la possibilité de classification des langages formels selon plusieurs critères qui dépendent du besoin, des concepts fondamentaux du langage, du support d'outils disponibles, du domaine d'application et de la nature du système développé. Par conséquent, nous avons cerné notre étude principalement sur les critères suivants (décrits dans les sous-sections suivantes) : la possibilité de couplage avec UML, le support d'outils disponibles pour les activités de V&V et l'adéquation par rapport au domaine d'application ferroviaire.

3.4.2.1 La possibilité de couplage avec UML

Pour ce premier critère, le langage OCL (Object Constraint Language) semble être le langage le plus approprié. En effet, c'est un langage très répandu dans la communauté de l'IDM qui permet l'expression de propriétés sur des modèles UML. UML n'était pas assez précis pour décrire les liaisons entre les éléments d'un modèle. Il s'est avéré nécessaire d'enrichir UML par des contraintes additionnelles pour faire face à l'ambiguïté, le manque de rigueur, l'incohérence, etc. Le langage OCL [OMG, 2014] a alors été développé pour contourner cette difficulté et a été standardisé par l'OMG en tant que langage formel utilisé pour spécifier des expressions sur des modèles UML, ainsi que pour exprimer les règles de bonne-formation du méta-modèle UML.

OCL est basé sur la théorie des ensembles et la logique des prédicats [Lonchamp, 2015]. C'est un langage d'expressions qui propose un formalisme utilisé en UML pour exprimer des invariants, des gardes, des contraintes, des pré- et des post-conditions, etc [OMG, 2014]. Il peut aussi être utilisé pour analyser un modèle via des requêtes. Néanmoins, plusieurs travaux dans la littérature recommandent le passage du modèle UML enrichi par des contraintes OCL vers d'autres formalismes et d'autres techniques d'analyse formelle plus mathématiques pour une question de performance [de Roquemaurel *et al.*, 2011b]. Bien qu'OCL soit considéré comme un langage formel simple, il souffre d'un manque d'une sémantique formelle complète [Duarte *et al.*, 2003]. C'est pourquoi, il est nécessaire d'aller vers un formalisme plus rigoureux. On trouve dans la littérature de nombreux travaux qui portent sur des transformations de UML avec des contraintes OCL vers d'autres notations formelles comme Z [Dupuy, 2000b], Object-Z [Roe *et al.*, 2002], B [Truong et Souquières, 2004, Ledang et Souquières, 2002], Alloy [Cunha *et al.*, 2015, Anastasakis *et al.*, 2010, Anastasakis *et al.*, 2007] et CSP [de Roquemaurel *et al.*, 2011a].

Alloy [Jackson, 2002] est un langage déclaratif permettant d'exprimer des propriétés

structurelles. Il se base sur la logique du premier ordre et dispose d'une syntaxe de déclaration compatible avec les modèles graphiques orientés objet. Sa syntaxe est donc largement compatible avec UML. Plusieurs travaux de couplage UML/Alloy existent dans la littérature. [Cunha *et al.*, 2015] propose une transformation bidirectionnelle de UML+OCL vers Alloy et d'Alloy vers UML+OCL, dans le contexte de l'ingénierie dirigée par les modèles. La transformation de UML+OCL vers Alloy permet la vérification et la validation, tandis que la transformation d'Alloy à UML+OCL permet de partager les modèles avec les parties prenantes et de les raffiner, à l'aide des outils de MDA, jusqu'à l'obtention d'un code. La transformation dans le sens UML+OCL vers Alloy a été déjà étudiée dans [Anastasakis *et al.*, 2010]. Ce papier propose, en plus de la transformation UML+OCL vers Alloy étendue par rapport aux précédents travaux [Anastasakis *et al.*, 2007], un profil UML pour Alloy pour une meilleure représentation des concepts d'Alloy en UML.

La méthode B est aussi l'une des méthodes formelles qui a été couplée avec UML dans plusieurs travaux de recherche. Nous consacrons la section 3.4.3 pour étudier les différentes approches de couplage UML/B dans la littérature.

Comme la méthode B, Z fait partie des méthodes orientées modèles qui représentent les logiciels par des données, caractérisées par leurs propriétés invariantes et des services qui manipulent ces données [OFTA, 1997]. Des travaux de la littérature ont proposé des approches de formalisation d'UML en notations Z [El-Miloudi et Ettouhami, 2015, El Miloudi *et al.*, 2013]. D'autres travaux ont étendu le langage Z vers Object-Z pour supporter les caractéristiques orientées-objet de l'UML [Miao *et al.*, 2002]. Cependant, la méthode B, qui a été inspirée de Z, est une méthode complète qui couvre tout le processus de développement depuis l'analyse jusqu'à l'implémentation du système comparée à Z [Gervais, 2004].

Le langage CSP est un langage de description de comportements de systèmes de processus communicants. C'est une algèbre de processus qui repose sur l'observation du comportement des processus communicants [Gervais, 2004]. Ce langage appartient alors à une classe de langages formels différente de celle de B et Z utilisée essentiellement pour la formalisation des systèmes concurrents. Le troisième critère étudie le choix du langage formel par rapport au domaine d'application et alors par rapport à l'objectif de son utilisation.

3.4.2.2 Le support d'outils disponibles pour les activités de V&V

La méthode B est une méthode bien outillée comparée à plusieurs langages formels OCL, Alloy, Z, etc. En effet, selon [de Roquemaurel *et al.*, 2011a], les outils actuels d'OCL ne sont pas efficaces pour une vérification des propriétés métier sur les modèles industriels de taille considérable. Alloy se prête toutefois mieux à une analyse sémantique totalement automatique comparé à OCL et B. Il dispose de deux types d'analyse : la simulation et la vérification de propriétés, associé à un outil appelé Alloy Analyzer [Ledru *et al.*, 2011]. Cependant, ni OCL ni Alloy ne disposent d'assistants de preuve ou de prouveurs de

théorèmes. En outre, les prouveurs de B supportent des espaces d'état plus considérables que ceux d'Alloy [Ledang et Dubois, 2010].

Les spécifications B peuvent être analysées en utilisant des outils de *model checking* et d'animation tels que *ProB* et des outils de preuve tels que l'*Atelier B* ou encore la plate-forme *Rodin*. Ces outils ont montré leur efficacité dans plusieurs applications industrielles. Chacun de ces outils de preuve possède un prouveur automatique et un prouveur interactif proposé à l'utilisateur dans le cas où le prouveur automatique n'arrive pas à prouver toutes les obligations de preuve. De plus, l'utilisateur a la possibilité d'enrichir la base de règles manuellement, de manière à fournir au prouveur des règles complémentaires permettant de simplifier ou généraliser des preuves complexes.

3.4.2.3 L'adéquation du langage par rapport au domaine d'application ferroviaire

Les domaines d'application pour les langages formels sont variés. Ils sont utilisés dans les systèmes d'information, les systèmes distribués, les réseaux, les systèmes critiques ferroviaires, avioniques, etc. Toutefois, les applications pour lesquelles les méthodes formelles ont été utilisées dans l'industrie sont encore limitées à des domaines bien particuliers. L'emploi d'une méthode inappropriée au domaine d'application peut ajouter des difficultés notationnelles qui ne font souvent qu'obscurcir le problème.

OCL a montré son efficacité dans des travaux liés à la cohérence de la modélisation en UML. Les travaux de [Bazex *et al.*, 2003] se situent dans le cadre de l'aide au développement du logiciel en utilisant UML comme support de modélisation et propose d'utiliser OCL intégré à UML pour vérifier un ensemble de points de vue. D'autres travaux de [Cariou *et al.*, 2004, Cariou *et al.*, 2009] proposent d'utiliser des contrats OCL pour spécifier des transformations de modèle indépendamment des technologies de transformation. Dans ce contexte, la méthode B a aussi été utilisée pour formaliser des transformations de modèle. En effet, les travaux de [Ledang et Dubois, 2010] visent à utiliser la méthode B et ses outils pour analyser et prouver la correction des règles de transformations de modèles en respectant les méta-modèles et les invariants ainsi que la consistance des règles de transformations entre elles.

La méthode B a été utilisée pour des applications industrielles de taille considérable dans divers domaines tels que les systèmes d'information [Ledru *et al.*, 2011], les bases de données [Mammar et Laleau, 2006a] et essentiellement dans le domaine du ferroviaire depuis des années [OFTA, 1997]. Son extension Event-B [Abrial, 2010] est une évolution dédiée à la modélisation et la vérification des systèmes discrets, mais aussi pour d'autres systèmes tels que les systèmes distribués [Butler, 2009] et les systèmes ferroviaires. Alloy a été aussi utilisé pour les systèmes critiques ferroviaires [Svendsen *et al.*, 2012, Eriksson, 2006], mais la méthode B reste la méthode la plus utilisée dans ce domaine. Par ailleurs,

le retour d'expérience de Systerel [1], l'un des acteurs industriels dans le secteur ferroviaire, a montré, qu'après plus que 20 ans de mise en œuvre de la méthode B, elle représente une technologie mature (intégration à la norme 50128) et efficace couvrant le cycle de vie de manière consistante de la spécification du logiciel à la production du code. De plus, le potentiel d'abstraction de B permet de faciliter la maîtrise de la complexité du système. Toutefois, l'utilisation de la méthode B nécessite une maîtrise poussée des mathématiques pour les aspects de modélisation et de preuve. Il est également difficile de reprendre un modèle non formel. Le passage par la modélisation UML semi-formelle que nous proposons, comme intermédiaire entre la spécification informelle et la spécification formelle, semble être une solution adéquate pour un tel problème.

Dans le domaine ferroviaire, la méthode B a été aussi utilisée avec d'autres langages formels pour la vérification des systèmes ferroviaires. [Moller *et al.*, 2013] propose une approche de modélisation basée à la fois sur CSP et B, appelée CSP||B. Le choix de cette approche est motivé par la combinaison du raisonnement basé sur les états (B) et le raisonnement basé sur les évènements (CSP). Cependant, la vérification des systèmes ferroviaires via cette approche se base uniquement sur la technique de *model checking* avec ProB. L'un des défis de cette approche est de faire face au problème d'explosion de l'espace d'état [James *et al.*, 2014]. Toutefois, la technique de vérification par la preuve semble être une solution à ce problème. D'autres travaux de [Sun, 2015], au sein du projet *PERFECT*, se sont intéressés à la modélisation avec des réseaux de Petri colorés du système d'enclenchement ferroviaire et proposent une transformation des modèles de réseau de Petri en machines B dans le but de bénéficier des avantages de la preuve et de la génération de code sûr.

3.4.2.4 Synthèse

Nous avons retenu la méthode B comme la méthode formelle la plus adaptée pour notre approche. En effet, elle a été utilisée avec succès pour une application industrielle de sécurité ferroviaire [Behm *et al.*, 1999]. Elle a été considérée, d'ores et déjà, comme efficace tout en aboutissant à des résultats probants dans le domaine et continue de l'être. On peut également tenir compte de la richesse des outils associés qui permettent une meilleure conduite des activités de vérification et de validation. Les travaux de la littérature présentés dans la section suivante concernent l'intégration de UML et B et montrent l'intérêt du couplage de ces deux notations. Enfin, nous pouvons également mentionner quelques critères en faveur de ce choix :

— la consistance de la couverture du cycle de développement de la spécification à la production du code en passant par une suite de raffinement (voir section 3.6.3), et
— l'avantage de la modularité (voir section 3.6.4) au sein de l'architecture d'un projet

1. Le retour d'expérience de Systerel se trouve sur ce lien http ://projects.laas.fr/IFSE/FMF/J1/P05_FBustany.pdf

B permettant la maîtrise de la complexité du système.

3.4.3 Approches de couplage UML/B

Dans cette section, nous présentons les différentes approches de couplage UML/B abordées dans la littérature.

Selon [Truong et Souquières, 2004], il existe deux approches de couplage d'UML et B :

— *Couplage par adjonction : il s'agit d'une intégration partielle, dans laquelle une partie du modèle à objets UML est remplacée par une expression formelle B. Cette approche est utile lorsque l'on souhaite ajouter plus de rigueur au développement.*

— *Couplage par dérivation : il s'agit de la transformation systématique des diagrammes UML en B. Cela nous fournit une vue générale d'un système avec deux spécifications et avec la possibilité de détecter certaines inconsistances dans la spécification.*

L'approche par dérivation d'UML vers B est l'une des trois approches de couplage abordées dans les travaux de thèse [Idani, 2006]. Les deux autres approches concernent le développement conjoint UML/B et la dérivation de B vers UML.

L'approche de dérivation de UML vers B a été étudiée par plusieurs travaux de recherche. Nous pouvons citer les travaux de [Laleau, 2002, Mammar et Laleau, 2006a] appliqués principalement sur des bases de données. Ces travaux ont été concrétisés par un outil de traduction de UML vers B : UML2SQL [Mammar et Laleau, 2006b]. Les travaux de thèse de [Ledang, 2002], qui sont complémentaires aux travaux de thèse de [Meyer, 2001], proposent une démarche de traduction de plusieurs types de diagrammes UML : les diagrammes de classes, les diagrammes de cas d'utilisation et les diagrammes de collaborations. Ces travaux ont également été concrétisés par l'outil ArgoUML+B [Ledang *et al.*, 2003], une extension de l'outil ArgoUML dédié pour la modélisation et la conception des diagrammes UML. Les travaux de [Snook et Butler, 2006] proposent un profil UML, appelé UML-B, permettant de spécialiser UML avec des détails du langage B qui ne font pas partie du standard UML et supportant le mécanisme de raffinement. Un outil de traduction de UML vers B a été développé dans ce contexte [Snook et Butler, 2004].

Comme l'approche de dérivation de UML vers B, l'approche conjointe UML/B mène à deux vues d'un même système, l'une semi-formelle graphique et l'autre formelle. Néanmoins, cette approche conjointe permet de faire évoluer la spécification tout en maintenant la cohérence et la traçabilité entre les deux représentations. Cette évolution permet de se concentrer simultanément sur les deux vues lors de chaque incrément de spécification [Okalas Ossami *et al.*, 2004, Okalas Ossami *et al.*, 2005, Okalas Ossami *et al.*, 2006]. Ces travaux ont été intégrés dans l'outil ArgoUML+B [Ledang *et al.*, 2003]. Par ailleurs, les travaux de [Idani *et al.*, 2010] ont présenté une infrastructure de couplage de notations graphiques et formelles qui réalise les transformations d'UML vers B dans le cadre de l'IDM. Cette

infrastructure permet de combiner les approches existantes, de les étendre et de les personnaliser dans le but d'en tirer profit en dehors du cadre applicatif dans lequel elles ont été établies. L'approche proposée est intégrée dans la plate-forme *B4MSecure*. Cette dernière permet de maîtriser et d'automatiser le processus de traduction en B des modèles fonctionnel et de sécurité. Dans le chapitre suivant, nous détaillons cette approche ainsi que la plate-forme B4MSecure.

L'approche de dérivation de B vers UML proposée par [Idani, 2006] a comme objectif principal de faciliter la compréhension des notations mathématiques de la méthode B en les dérivant vers des vues graphiques UML plus accessibles à des personnes qui ne seraient pas forcément familiarisées avec ces notations mathématiques.

3.4.4 Discussion

Notre objectif est d'avoir deux vues distinctes, l'une graphique et l'autre formelle intégrant à la fois structuration et précision et menant aux activités de vérification et de validation formelles des exigences de sécurité. L'approche par adjonction ne semble pas être adéquate. En effet, nous avons besoin d'une traduction d'une partie des spécifications informelles en des spécifications formelles en passant par des modèles UML. Certes l'intégration partielle de la méthode B dans une partie du modèle UML ajoute plus du rigueur au développement, mais elle ne permet en aucun cas de mener à bien l'analyse et la vérification de la correction des spécifications formelles contenant des spécifications fonctionnelles et non-fonctionnelles de sécurité.

L'approche conjointe UML/B permet de favoriser la cohérence et la traçabilité entre les deux vues du système. En effet, tout incrément du côté de la spécification UML est dérivé simultanément du côté de la spécification B. Dans le cadre de nos travaux, les exigences de sécurité s'ajoutent aux spécifications fonctionnelles. Il est ainsi intéressant d'enrichir les spécifications UML par les exigences de sécurité qui seront dérivées et intégrées dans les spécifications B.

3.5 Unified Modeling Language (UML)

UML (Unified Modeling Language) est un langage de modélisation objet graphique universel. C'est le résultat de la réunion des trois méthodes les plus répandues en orienté objet qui sont la méthode Booch de Grady Booch, OOSE (Object-Oriented Software Engineering) d'Ivar Jacobson, et OMT (Object Modeling Technique) de James Rumbaugh [Booch *et al.*, 2005]. En vue de la normalisation d'un langage de modélisation, l'OMG a standardisé la notation UML. Depuis lors, UML a eu une place prépondérante dans le processus de développement logiciel. En effet, un éventail de domaines en informatique utilisent UML comme langage de modélisation de leurs applications.

3.5.1 Définition et Sémantique UML

Dans cette section, nous présentons quelques notations UML qui nous semblent utiles pour notre approche ainsi que leurs sémantiques associées.

UML est constitué d'une large collection de notations dont le but est de modéliser des systèmes logiciels dans tous les aspects : données, état, comportement, communication, etc. Les notations UML sont interreliées et interdépendantes, ce qui rend la définition de la sémantique d'UML très difficile [Lano, 2009]. Ces notations sont traduites par un ensemble de diagrammes UML graphiques, défini dans la figure 3.1.

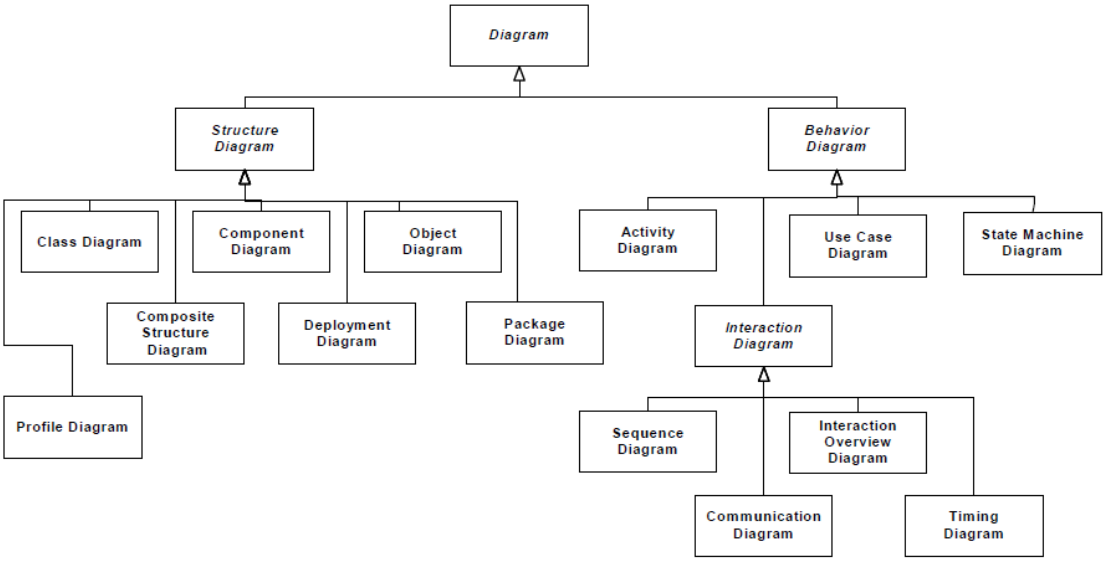

Figure 3.1 – Les différents diagrammes de UML2 [OMG, 2011]

Selon [Lano, 2009], une sémantique « *est une mise en correspondance précisément définie des éléments d'un langage dans un domaine de valeurs précisément défini* ».

Dans notre travail, nous nous sommes intéressés principalement au diagramme de classes. D'autres diagrammes UML, comme les diagrammes de séquence et les diagrammes d'activité, nous semblent intéressants, cependant ils restent des perspectives d'extension de ce présent travail de thèse. Ainsi, nous allons détailler le diagramme de classes et sa sémantique associée.

Le diagramme de classes est le plus utilisé pour la modélisation des logiciels. Il est principalement construit par des classes, des associations, des attributs et des opérations dans la structure interne d'une classe [OMG, 2011] :

- Une classe décrit un ensemble d'objets qui partagent les mêmes spécifications de caractéristiques, de contraintes et de sémantique. Le but d'une classe est de spécifier une classification des objets et de spécifier les caractéristiques qui caractérisent la structure et le comportement de ces objets.
- Une association décrit un ensemble de tuple dont les valeurs se réfèrent aux instances

typées. Une instance d'une association est appelée un lien. Un lien est un tuple avec une seule valeur pour chaque extrémité de l'association, où chaque valeur est une instance du type de l'extrémité. Une association permet alors de déclarer s'il peut y avoir des liens entre les instances des types associés.

- Les attributs d'une classe sont représentés par des instances de la propriété que possède cette classe. Certains de ces attributs peuvent représenter les extrémités navigables des associations binaires.
- Les opérations d'une classe peuvent être invoquées par un objet, compte tenu d'un ensemble particulier de substitutions pour les paramètres de l'opération. Une invocation d'opération peut provoquer des changements dans les valeurs des attributs de cet objet. Il peut également renvoyer une valeur de retour lorsqu'un type de retour de l'opération a été défini. Il est aussi possible d'associer des pré-conditions, des post-conditions et des body-conditions aux opérations d'une classe. En effet, les pré-conditions d'une classe définissent les conditions qui doivent être remplies lorsque l'opération est invoquée. Les post-conditions définissent les conditions qui seront vraies lorsque l'invocation de l'opération se termine avec succès en supposant que les pré-conditions ont été déjà satisfaites, tandis que les body-conditions contraignent le résultat de retour de l'opération. Ces dernières diffèrent des post-conditions par le fait qu'elles peuvent être remplacées lorsqu'une opération est redéfinie, alors que les post-conditions ne peuvent qu'être ajoutées lors de la redéfinition de l'opération.

3.5.2 Vues et diagrammes UML

Étant donné qu'une vue décrit un point de vue du système, [Kruchten, 1995] propose un modèle pour l'architecture de systèmes logiciels basé sur cinq vues, désignées par les « 4+1 » vues. L'utilisation de ces différentes vues permet de traiter séparément les préoccupations des différentes « parties prenantes » de l'architecture : les utilisateurs finaux, les développeurs, les ingénieurs systèmes, les chefs de projet, etc., et de gérer séparément les exigences fonctionnelles et non-fonctionnelles. Les cinq vues concernent :

— La vue logique qui décrit les aspects statiques et dynamiques d'un système en se basant sur la notion d'objet, classe, ...,
— La vue des processus qui capte les aspects de concurrence et de synchronisation pour la conception,
— La vue physique qui décrit la mise en correspondance du logiciel sur le matériel et reflète son aspect distribué
— La vue de développement qui décrit l'organisation statique du logiciel dans son environnement de développement,
— La vue de cas d'utilisation qui est redondante avec les autres vues (d'où le « +1 »). Néanmoins, elle permet de découvrir les éléments architecturaux lors de la conception et de valider cette architecture. En effet, elle assure la cohérence globale en

présentant des scénarios d'utilisation qui sont en quelque sorte une abstraction des exigences les plus importantes.

Comme le modèle de Kruchten est apparu avant UML, plusieurs tentatives d'application de l'architecture à « 4+1 » vues sur UML existent. Comme certains diagrammes UML peuvent être utilisés dans plusieurs vues, les propositions de classification de ces diagrammes diffèrent. Cependant, la modélisation du même type de diagramme UML change d'une vue à une autre. Nous faisons référence au travail de [Muchandi, 2007] qui fait correspondre chaque type de diagramme UML à une vue architecturale. La figure 3.2 illustre cette proposition, parmi d'autres comme celle d'IBM.

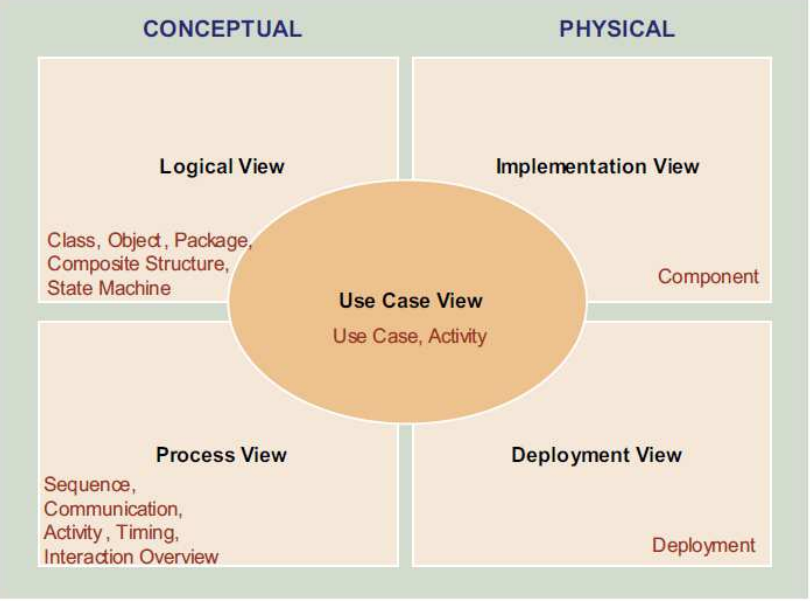

Figure 3.2 – L'application de l'architecture des « 4+1 » vues de Krutchen sur UML [Muchandi, 2007]

Les diagrammes UML peuvent également représenter les trois vues (aspects) suivantes : vue statique, vue dynamique et vue fonctionnelle. La vue fonctionnelle à travers le diagramme de cas d'utilisation permet d'appréhender les interactions entre les différents acteurs/utilisateurs du système et le système lui-même, en structurant les besoins des utilisateurs/acteurs et les objectifs correspondants du système. En ce qui concerne la vue statique, elle favorise la structuration des données du système et identifie les liaisons des unes par rapport aux autres. Le diagramme UML de classes, le diagramme d'objets, le diagramme de composants et le diagramme de déploiement mettent en exergue cette vue. Enfin, la vue dynamique, quant à elle, décrit l'évolution (ou la dynamique) du système par rapport à son cycle de vie. UML permet de spécifier cette vue à travers le diagramme d'états-transitions, le diagramme de séquence, le diagramme d'activité et le diagramme de communication, nouvelle appellation du diagramme de collaboration, en UML2.

3.6 La méthode B

La méthode B est une méthode qui couvre toutes les phases de développement formel. En effet, un cycle de développement classique se compose généralement de phases de spécification, de conception, d'implémentation et de test appuyées éventuellement par des activités de vérification et de validation. La méthode B apporte des modifications sur ce cycle par la formalisation d'un modèle abstrait aboutissant à la production de code exécutable par une suite de raffinements prouvés (voir figure 3.3). Le passage d'une phase à une autre dans un processus de développement en B se fait par raffinement (décrit dans la section 3.6.3) et implique la démonstration d'un ensemble d'obligations de preuve dans le but de validation. La méthode B a été aussi utilisée dans d'autres objectifs tels que la certification de propriétés ou de protocoles [Petit, 2003].

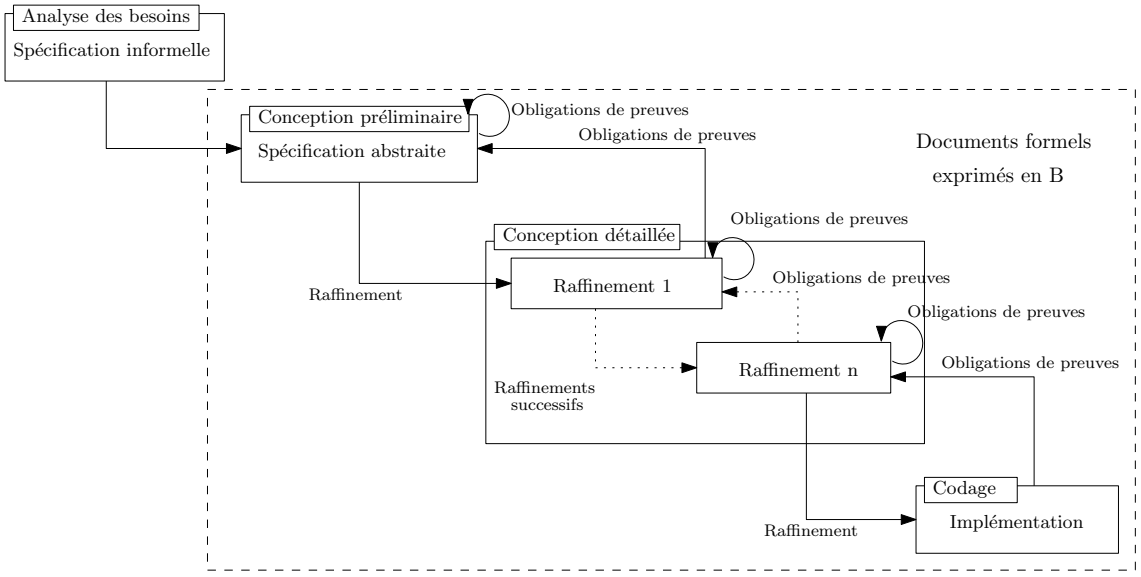

Figure 3.3 – Processus de développement en B [Idani, 2006]

Dans ce qui suit, nous présentons l'historique industriel de la méthode B et nous définissons ses fondements conceptuels et ses notations. Ensuite, nous poursuivons par la notion du raffinement et de la modularité en B. Nous achevons cette vue générale par la présentation de quelques outils de B que nous avons utilisés pour réaliser notre apport.

3.6.1 Historique

Plusieurs projets de métros automatiques sans conducteur, réalisés par Siemens, ont été développés sans méthode formelle [2] tels que le VAL à Lille et MAGGALY (Métro à Grand Gabarit de l'Agglomération LYonnaise). Suite à ces projets, des méthodes formelles ont été introduites pour réaliser le SACEM (Système d'Aide à la Conduite, à l'Exploitation

2. leader mondial des métros automatiques sans conducteur, des automatismes d'aide à la conduite et de l'automatisation des réseaux de métro traditionnels

et à la Maintenance), un automatisme ferroviaire destiné à la RATP [Dollé *et al.*, 2003].

Au milieu des années 80, J.-R. Abrial a conçu la méthode B [Abrial, 1996], en s'inspirant des méthodes formelles VDM et Z. Comme VDM et Z, la méthode B fait partie des méthodes formelles orientées modèle. Elle a été utilisée industriellement avec le projet Météor [Behm *et al.*, 1999], la ligne 14 du métro parisien entièrement automatique sans conducteur mise en service en 1998. Ce projet a été développé par *Matra Transport International* (maintenant Siemens) pour la RATP. Le pilotage de ce système a nécessité l'utilisation des logiciels de sécurité développés formellement avec la méthode B dont la preuve a permis de supprimer les tests unitaires et a donné un résultat remarquable. Comparé au projet SACEM, l'utilisation de la méthode B dans ce projet s'est effectuée plus en amont dès la phase de développement des logiciels de sécurité. En effet, comme la preuve de programme valide le code par rapport à la conception détaillée, il devrait de même pour la conception détaillée par rapport à la spécification. Dès lors, la méthode B a été généralisée à l'ensemble des logiciels pour les équipements SACEM [Dollé *et al.*, 2003].

3.6.2 Fondements et notations

Trois notations de base constituent les piliers du langage de modélisation de la méthode B, à savoir : la notation mathématique, la notation des machines abstraites et la notation des substitutions généralisées.

3.6.2.1 Notation mathématique

Les fondements mathématiques de la méthode B sont issus de la logique du premier ordre et de la théorie des ensembles.

Le langage B est un langage de données relevant de la théorie des ensembles permettant de modéliser les données et les propriétés des données. La méthode repose, toutefois, sur une théorie des ensembles typée. En effet, les éléments des ensembles en langage B ont tous la même structure fondamentale. D'ailleurs, un ensemble ne peut pas contenir, par exemple, des entiers et des couples d'entiers à la fois [OFTA, 1997]. Cette notion de typage repose sur la notion d'ensemble et la propriété de monotonie de l'inclusion [Clearsy System Engineering, 2009].

Définition 4 *Soient E une expression, S et T des ensembles tels que $S \subseteq T$. Si $E \in S$ alors $E \in T$. Le plus grand ensemble dans lequel E est contenu s'appelle le type de E.*

Dans le langage B, le typage se présente sous trois aspects [Clearsy System Engineering, 2009] :

— Les types du langage B qui sont les types de base ($\mathbb{Z}$, $\mathbb{N}$, $\mathbb{N}1$, NAT, BOOL, STRING) et les types construits à l'aide du produit cartésien, de l'ensemble des parties (ensemble d'ensembles) et des ensembles de records (la collection d'articles).

— Le typage des données qui est utilisé dans un prédicat ou une substitution (détaillée dans la section 3.6.2.3).
— La vérification de type qui s'effectue à travers des règles de typage relatives aux expressions, prédicats ou substitutions. La vérification de type d'un composant B est un pré-requis à la preuve du composant.

Dans le langage B, les prédicats du premier ordre sont utilisés pour exprimer des propriétés de données sous formes de prédicats construits avec les opérateurs propositionnels classiques (et ($\wedge$), ou ($\vee$), non ($\neg$), implique ($\Rightarrow$), équivaut ($\Leftrightarrow$)) ou des prédicats ayant des variables quantifiées universellement et existentiellement ($\forall$x.P et $\exists$x.P).

3.6.2.2 Notation des machines abstraites

La machine abstraite est la notion de base de la méthode B. Cette notion est similaire à des notions de programmation connues sous le nom d'un module, d'une classe ou d'un type de donnée abstrait [Abrial, 1996].

De façon générale, la machine abstraite se scinde en trois parties :

```
MACHINE
  PARTIE ENTÊTE
    Nom de la machine abstraite
    Paramètres de la machine
    Contraintes sur les paramètres
  PARTIE STATIQUE
    Déclaration d'ensembles
    Déclaration de constantes
    Propriétés des constantes
    Déclaration des variables (état)
    Invariant (caractérisation de l'état)
  PARTIE DYNAMIQUE
    Initialisation de l'état
    Opérations sur l'état
END
```

— La partie entête sert à identifier la machine abstraite : le nom de la machine dans la clause MACHINE, ses éventuels paramètres et la description de ses paramètres à l'aide de prédicats de typage dans la clause CONSTRAINTS.
— La partie statique contient la déclaration de la liste des ensembles abstraits et la définition des ensembles énumérés dans la clause SETS, ainsi que la déclaration des constantes dans la clause CONSTANTS et des variables dans la clause VARIABLES. Ces déclarations sont complétées par la définition du type et des propriétés de ces constantes et de ces variables respectivement dans les clauses PRO-

PERTIES et INVARIANT. Les propriétés de la clause INVARIANT doivent être toujours satisfaites par l'état de la machine.

— La partie dynamique définit l'évolution de l'état de la machine à travers une initialisation dans la clause INITIALISATION et les opérations de la machine dans la clause OPERATIONS. La clause OPERATIONS permet de déclarer les services offerts par une machine et de spécifier leur comportement en utilisant la notion de substituions généralisées. Les opérations décrites permettent, en effet, de modifier les données de la machine tout en préservant les propriétés des variables de la clause INVARIANT.

3.6.2.3 Notation de substitutions généralisées

La notation de substitutions généralisées a pour but de modéliser un ensemble de services qui définit le mécanisme d'évolution des données de la machine. Les substitutions sont des notations mathématiques permettant de modéliser la transformation de prédicats. En effet, elles produisent, à partir des prédicats qui définissent les données avant l'activation d'un service, les prédicats qui définissent les données après l'accomplissement de ce service [Abrial, 1996].

Les substitutions généralisées sont des extensions de la substitution élémentaire. Cette dernière est définie dans [Abrial, 1996] comme suit : ≪ *si x est une variable et e est une expression ensembliste, la substitution élémentaire x :=e, appliquée à un prédicat P, noté [x :=e]P, transforme P en un prédicat P', obtenu en remplaçant dans P toutes les occurrences libres*[3] *de x par e* ≫.

Selon [Abrial, 1996], les principales substitutions généralisées sont :

— *la substitution avec précondition, notée* ***PRE*** *P* ***THEN*** *S* ***END****, qui définit les conditions d'utilisation d'une autre substitution ;*

— *la substitution avec garde, notée* ***SELECT*** *P* ***THEN*** *S* ***END****, qui définit le contexte d'application d'une autre substitution ;*

— *la substitution de choix non-déterministe borné, notée* ***CHOICE*** *S1* ***OR*** *S2* ***END****, qui définit deux substitutions possibles ;*

— *la substitution de choix non-déterministe non-borné, notée* ***ANY*** *X* ***WHERE*** *P* ***THEN*** *S* ***END****, qui définit un nombre indéterminé de substitutions possibles, une pour chaque valeur des variables X qui satisfait au prédicat P ;*

— *la composition parallèle de substitutions, notée S1 || S2, qui effectue simultanément deux substitutions ;*

— *la substitution identité, notée* ***SKIP****, qui laisse inchangés les prédicats ;*

— *la substitution conditionnelle déterministe, notée* ***IF*** *C* ***THEN*** *S1* ***ELSE*** *S2* ***END*** *;*

— *la composition séquentielle de substitutions, notée S1 ; S2 ;*

3. Les occurrences libres d'une variable dans un prédicat sont les occurrences de cette variable qui ne sont pas introduites par un quantificateur.

— *et, finalement la substitution d'itération, notée* ***WHILE*** *C* ***DO*** *S* ***INVARIANT*** *I* ***VARIANT*** *V* ***END****.*

3.6.3 Raffinement

La méthode B propose une construction par approximations successives, appelée *raffinement*. La notion de raffinement est fondamentale en B. D'une part, elle permet d'introduire les détails du problème, qui n'étaient pas pris en compte auparavant, dans le processus de développement formel. Par conséquent, la traduction du problème initial est effectuée d'une façon progressive et non pas en une seule fois. D'autre part, le raffinement est également utilisé pour raffiner les machines abstraites afin d'obtenir un code exécutable. Le passage de la spécification abstraite vers le code nécessite des preuves de raffinement dans le but de s'assurer que le code final d'une machine satisfait la spécification initiale. Le dernier raffinement est désigné par *implantation*.

Sur le plan pratique, selon [Abrial, 1996], le raffinement d'une machine abstraite est mené par :

1. la suppression des éléments non-exécutables du pseudo-code des opérations de la machine abstraite (les pré-conditions et les choix (choice)),
2. l'introduction des structures de contrôle classique de la programmation (le séquencement et les boucles (loop)), et
3. la transformation des structures de données mathématiques (ensembles , relations, fonctions, séquences et arbres) en des structures programmables (des variables simples, des tableaux ou des fichiers).

Les opérations de la machine abstraite restent les mêmes dans le raffinement, mais leurs pseudo-codes seront modifiés selon le premier et le deuxième points.

3.6.4 Modularité en B

En partant du principe que la réalisation d'un système complexe ne peut s'accomplir en une seule fois, la méthode B propose une construction par plusieurs niveaux de raffinement et une décomposition en modules [Potet, 2002].

La décomposition en modules est une caractéristique importante de la méthode B car une architecture modulaire atténue la complexité perçue pour un système. Un module B permet de modéliser un sous-système faisant partie d'un projet B. Les modules sont constitués par des composants B. Un composant peut être une machine abstraite, un raffinement ou une implantation. En fonction des trois propriétés suivantes du module : (i) il comprend toujours une machine abstraite, (ii) il peut posséder une implantation et éventuellement des raffinements et (iii) il peut posséder un code associé, nous pouvons distinguer trois sortes de modules [Clearsy System Engineering, 2009] :

— des modules développés par raffinements successifs d'une machine abstraite qui satisfont les trois propriétés en ayant un code associé par traduction,
— des modules de base qui ne satisfont pas la deuxième propriété mais satisfont la troisième en ayant un code associé manuellement, et
— des modules abstraits qui ne satisfont ni la deuxième ni la troisième propriété.

Afin d'assurer la cohérence mathématique de ces composants (modules) et la correction d'un raffinement vis-à-vis de ses versions plus abstraites, les obligations de preuve sont ainsi générées par la méthode. La sous-section suivante décrit les obligations de preuve prescrites par la méthode.

3.6.5 Obligations de preuve

Selon [Clearsy System Engineering, 2011b], une obligation de preuve se définit comme suit :

Définition 5 *Une obligation de preuve est une formule mathématique à démontrer afin d'assurer qu'un composant B est correct. La théorie B indique quelles sont les obligations de preuve à démontrer pour assurer la correction d'un composant B donné. Dans cette optique, les obligations de preuve sont une aide au processus de vérification.*

Les obligations de preuve sont générées pour les machines abstraites, les raffinements et les implantations. Nous citons, dans ce qui suit, les obligations de preuve qui concernent la correction de l'initialisation et la correction des opérations pour les machines abstraites et les raffinements [Clearsy System Engineering, 2011b].

Les obligations de preuve relatives à une machine abstraite concernent, entre autres :
— la correction de l'initialisation : l'initialisation doit établir l'invariant de la machine (l'invariant de la machine doit être vrai après application de la substitution de l'initialisation) ;
— la correction des opérations : les opérations doivent préserver l'invariant (l'invariant de la machine doit être vrai après application de la substitution de l'opération, sachant que l'invariant était vrai avant) ; les opérations doivent établir leur post-condition.

Dans le cas du raffinement, la correction de l'initialisation et des opérations fait intervenir l'initialisation/l'opération abstraite afin de montrer la pertinence du raffinement. Les obligations de preuve relatives à un raffinement concernent, entre autres :
— la correction de l'initialisation : l'initialisation doit établir l'invariant du raffinement (propriétés des nouvelles variables et invariant de liaison) sans contredire l'initialisation spécifiée ;
— la correction des opérations : les opérations doivent préserver l'invariant du raffinement (propriétés des nouvelles variables et invariant de liaison) sans contredire l'opération abstraite spécifiée.

3.6.6 Outils existants de la méthode B

Un des points forts de B est la disposition d'un support d'outils performant pour la mise en œuvre industrielle. Les outils de preuve permettent de générer automatiquement des obligations de preuves à partir des composants du langage B. Pour qu'un composant soit correct, il faut ensuite démontrer ses obligations de preuve.

La méthode B dispose de nombreux outils, aussi bien industriels qu'académiques, destinés à la preuve formelle, le model checking et/ou l'animation. Les outils *Atelier B*[4] (développé par *ClearSy*) et *B-Toolkit* [Lano et Haughton, 1996] (développé par *B-Core*) sont deux outils industriels qui sont répandus en France et en Angleterre respectivement. Ces outils permettent de développer formellement un logiciel de sa spécification à la génération du code en assistant l'utilisateur dans les phases de preuves menées dans le but d'établir la sûreté de développement. En plus des outils industriels commercialisés, il existe des outils académiques ayant pour but de fournir des plate-formes d'expérimentations pour la méthode B tel que la plate-forme *BRILLANT* [Colin *et al.*, 2005]. L'outil *ProB*[5], quant à lui, est à la fois un model checker et un animateur pour la méthode B.

Dans ce qui suit, nous présentons l'*Atelier B* et *ProB* que nous avons utilisés comme outil de preuve et d'animation respectivement.

- L'*Atelier B* est un outil industriel développé par la société *ClearSy System Engineering* qui permet une utilisation opérationnelle de la méthode formelle B pour des développements de logiciels prouvés sans défaut. Il regroupe notamment un type-checker, un générateur d'obligations de preuve, un prouveur automatique et un prouveur interactif. Le prouveur automatique permet de démontrer la plupart des obligations de preuve vérifiables, tandis que le prouveur interactif permet de détecter des erreurs et de finaliser la preuve. Il supporte deux activités de vérification par la preuve : la vérification de la consistance utilisée pour montrer que les opérations de la machine préserve l'invariant, et la vérification du raffinement utilisée pour montrer qu'une machine est un raffinement valide d'une autre.
- *ProB* supporte plusieurs techniques de validation telles que l'animation et le model checking. L'animation de *ProB* [Leuschel et Butler, 2003] permet aux utilisateurs d'avoir confiance en leurs spécifications. C'est une animation étape-par-étape des machines B qui prend en charge les opérations, même non-déterministes. Elle permet de visualiser une description de l'état actuel de la machine, l'historique qui a conduit l'utilisateur à accéder à cet état actuel, et une liste de toutes les opérations permises. Selon [Leuschel et Butler, 2003], le point fort de l'animation de ProB est que la taille de la machine B et le nombre de variables de la machine B ne posent

4. Atelier B : http ://www.atelierb.eu/
5. https ://www3.hhu.de/stups/prob/index.php/Main_Page

pas un problème lors de l'animation. *ProB* est capable de gérer des machines B avec des centaines d'opérations et des milliers de variables. L'animation de *ProB* permet non seulement un retour immédiat sur l'état actuel de la machine et son historique, mais également la détection des problèmes de spécifications aussi bien pour des utilisateurs expérimentés que pour des utilisateurs débutants. En effet, les spécifications de grande taille sont très susceptibles de contenir des erreurs, dont certaines peuvent être identifiées au début du processus de développement par l'animation.

Nous avons maintenant défini les concepts fondamentaux d'UML et B. Nous entamons la dernière section qui comporte la description de notre approche.

3.7 Notre approche UML/B

3.7.1 Motivation et cadre du projet PERFECT

La gestion de la signalisation dans le système ERTMS est régie par les règles nationales propres à chaque pays et non par des règles globales. L'analyse de la sécurité des systèmes doit pouvoir prendre en considération ces différents aspects et leur impact à plusieurs niveaux d'analyse. Ceci n'est que peu abordé dans la littérature. Le projet *PERFECT*[6] a pour objectif la proposition d'une démarche méthodologique outillée pour aider à l'évaluation globale de la cohérence de la spécification et des règles d'exploitation au regard des exigences de sécurité. Cette démarche permet d'accompagner l'activité des organismes et des autorités de sûreté de fonctionnement dans les phases amont de définition des réglementations de sécurité relatives à la signalisation ferroviaire ou à leur déclinaison pour des lignes particulières.

Elle consiste en des pistes méthodologiques ciblées vers différentes problématiques : analyse de la réglementation, modélisation de la gestion des enclenchements et de la gestion des itinéraires avec plusieurs technologies formelles.

La motivation principale des actions menées dans le cadre de cette démarche part du constat que face à la multiplicité des facettes entrant en jeu dans l'analyse de la réglementation de sécurité et ses déclinaisons nationales et régionales, il n'est pas possible de proposer une approche unique de modélisation et d'analyse [Lanusse *et al.*, 2014]. La figure 3.4 montre les choix guidés par la volonté de transposer des modèles définis dans des formalismes (UML/SysML, Réseaux de Petri, etc) vers d'autres formalismes d'analyse externes appropriés à l'objectif d'évaluation [Lanusse *et al.*, 2014] :

- Pour les aspects d'évaluation de la réglementation à haut niveau de modélisation (la partie gauche de la figure 3.4), le couplage d'une modélisation semi-formelle en UML a pris en compte des aspects de sécurité et sa dérivation en spécifications formelles B [Ben Ayed *et al.*, 2015, Ben Ayed *et al.*, 2014b, Ben Ayed *et al.*, 2014c, Ben Ayed *et al.*, 2014a].

6. http ://perfect.ifsttar.fr/Site

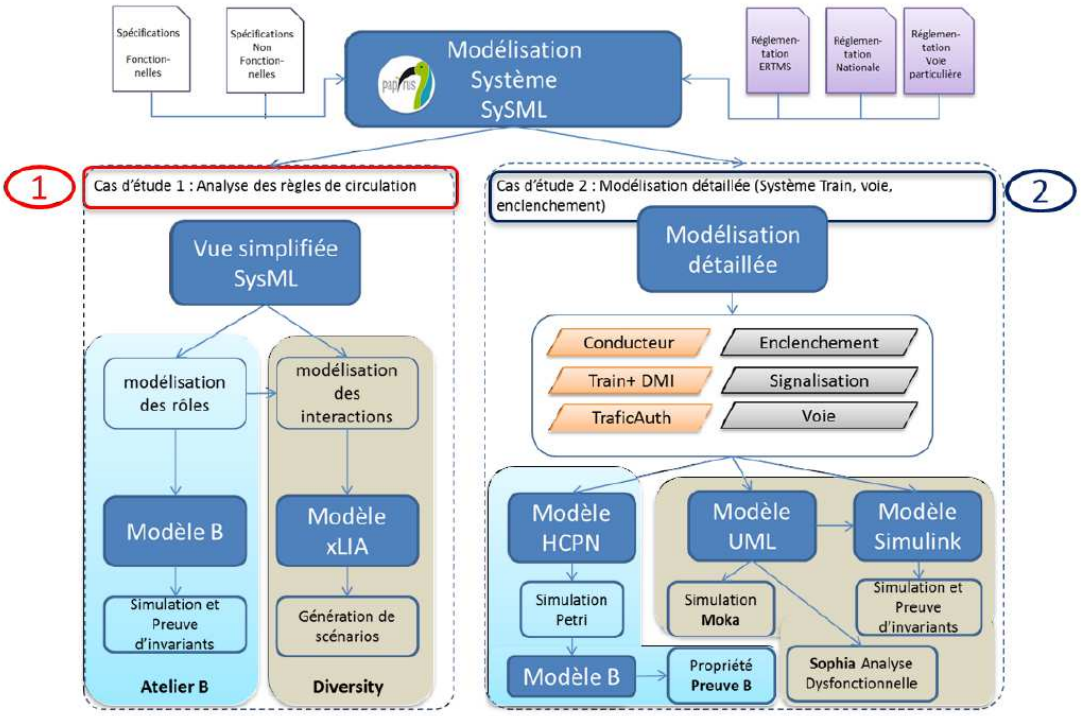

Figure 3.4 – Vue globale sur les travaux du projet *PERFECT* [Lanusse *et al.*, 2014]

- Pour les aspects d'évaluation de la mise en œuvre de la réglementation à un niveau détaillé (la partie droite de la figure 3.4), deux pistes méthodologiques ont été conduites :
  — la première porte sur une modélisation en réseau de Petri coloré hiérarchique d'un système d'enclenchement [El Amraoui et Mesghouni, 2014, Sun, 2015], le système ainsi modélisé peut être simulé et/ou traduit en B pour évaluation du respect d'invariants de sécurité [Sun, 2015].
  — la seconde porte sur une modélisation du comportement du même système d'enclenchement à l'aide de stateflow SIMULINK [Qiu *et al.*, 2014a, Qiu *et al.*, 2014b].

La partie gauche de la figure est la partie qui concerne nos travaux de recherche décrits dans le présent document. Dans cette partie, nous retrouvons notamment notre vision amont de l'analyse des procédures et des réglementations nationales au regard des règles européennes ERTMS basée sur le couplage UML/B. La modélisation SysML est utilisée pour capitaliser la formalisation des exigences, leur traçabilité et le lien avec les résultats de vérification. La partie droite définit une autre vision plus opérationnelle et détaillée des interactions entre systèmes : Train, Enclenchement, Signalisation, Voie, etc. Dans ce

contexte, les auteurs [Faivre *et al.*, 2015] ont proposé une étude comparative entre deux approches basées sur les modèles afin d'assurer la sûreté de fonctionnement du système ferroviaire dans le contexte d'ERTMS. Cette étude est effectuée au moyen des outils d'analyse avec différentes stratégies de vérification, à savoir *Matlab/Simulink* et *Diversity*.

Afin de bien cadrer notre approche par rapport au cycle de développement logiciel, nous présentons dans la sous-section suivante le cycle en V et le positionnement de notre approche.

3.7.2 Cycle de développement

Le cycle en V est le cycle recommandé par les différentes normes dont la 50128, comparé au cycle en cascade et le cycle en spirale connus pour la réalisation d'une application logicielle. Par sa branche descendante et sa branche remontante, le cycle en V apporte une amélioration par rapport au cycle en cascade. En effet, la branche descendante contient les activités de spécification, de conception et d'implémentation, les étapes nécessaires à la définition et la construction du produit. La branche remontante est destinée, quant à elle, aux activités de vérification et de validation. Ces activités font « écho » aux activités de la branche descendante. Ceci permet de détecter les défauts le plus tôt possible au niveau de chaque étape [Badreau et Boulanger, 2014].

La première phase de *spécification* commence par une analyse des besoins à partir du cahier de charges qui engendre les spécifications fonctionnelles. Ces spécifications seront l'entrée pour la phase de conception qui consiste à façonner et formaliser le système en répondant à l'ensemble des exigences du système (fonctionnelles, non-fonctionnelles, contraintes). Étant très proches, les activités de spécification et de conception sont souvent menées de façon parallèle et sont regroupées en une seule discipline [Badreau et Boulanger, 2014]. Dans notre cycle, nous ne séparons pas la phase de spécification, qui permet de regrouper l'ensemble des exigences du sous-système étudié, de la phase de conception. Cette phase permet une formalisation/modélisation semi-formelle en UML et formelle en B. Le modèle semi-formel qui répond aux spécifications fonctionnelles de départ sera enrichi par les exigences de sécurité ; et de même pour les spécifications formelles du modèle formel B [7]. Cette phase de conception est suivie souvent d'une phase de conception détaillée qui peut être un raffinement du modèle formel de la conception globale en suivant un processus de développement formel. Pendant la phase de conception détaillée, d'autres diagrammes UML peuvent être modélisés, traduits en des spécifications formelles B et ensuite intégrés dans l'architecture du système en B grâce à la modularité de la méthode B. La dernière phase est la phase de l'implantation qui peut être engendrée par le dernier raffinement de ce modèle selon le processus de développement formel de la méthode. Toutefois, nos travaux de recherche ne traitent ni la phase de conception détaillée ni la phase de l'implémentation.

7. Dans cette thèse, nous désignons souvent le modèle formel issu de la phase de formalisation/modélisation par des spécifications formelles B.

Les activités de la branche montante du cycle ont pour but d'améliorer l'anticipation sur la détection des défauts au niveau de chaque étape. Les activités de V&V qui correspondent aux activités de modélisation des spécifications fonctionnelles et non-fonctionnelles, sont alors faites sans attendre le passage aux phases de conception détaillée qui précède la phase d'implémentation.

Une étude de l'existant sur le couplage des notations semi-formelles et les notations formelles, ainsi que les différents langages de modélisation, a motivé notre choix de l'UML et B en tant que, respectivement, des langages semi-formel et formel de modélisation.

3.8 Synthèse

UML et B sont deux notations de spécifications qui sont répandues en génie logiciel. Le couplage entre ces deux types de notations est motivé par la complémentarité de leurs spécifications intégrant d'un côté la structuration et la facilité de compréhension et de communication par les parties prenantes et de l'autre côté la rigueur et la précision. L'utilisation des spécifications B traduites à partir des spécifications UML, par approche conjointe, permet de mener à bien les activités de vérification et de validation des règles d'exploitation ferroviaires vis-à-vis des exigences de sécurité.

L'idée de l'utilisation d'un profil UML permettant une meilleure modélisation de ces règles est intéressante. En effet, le profil UML permet de mettre en avant les concepts de la logique métier du domaine d'application. Dans la description de notre approche, nous n'avons pas détaillé nos contributions par rapport à l'utilisation d'un profil UML dédié pour le contrôle d'accès des systèmes d'information afin de répondre à notre problématique dans le domaine ferroviaire. Ceci fait l'objet du chapitre suivant.

Deuxième partie

CONTRIBUTIONS : DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION VERS LA SÉCURITÉ FERROVIAIRE

Chapitre 4

Approche basée sur Role Based Access Control (RBAC)

Sommaire

4.1 Introduction

La problématique de couplage des notations semi-formelle et formelle a déjà été abordée et le choix UML/B a déjà été fixé. Nous présentons donc dans ce chapitre nos contributions par rapport à la modélisation et la validation des règles d'exploitation ferroviaires. Ces règles ont la vocation de définir les autorisations de déplacement des trains sur des lignes ferroviaires équipées d'ERTMS en France. Ces autorisations sont spécifiées, en fonction des contraintes, aux acteurs humains et informatiques du système. En partant de ces concepts d'autorisations et d'acteurs, les travaux de recherche présentés dans ce chapitre proposent une approche originale fondée sur la réutilisation et l'adaptation des techniques de la sécurité des systèmes d'information pour des besoins de sécurité ferroviaire.

Notre étude de cas est extraite des principes et règles d'exploitation du système ERTMS/ETCS niveau 2 appliqués à la ligne à grande vitesse LGV-Est Européenne [1] [RFF, 2012] [2] et des spécifications européennes *SRS Subset 026* [UNISIG, 2006] (System Requirements Specification), disponibles sur le site de l'ERA [3]. Nous avons choisi deux scénarios pour notre étude, un scénario nominal d'autorisation de mouvement (en anglais *Movement Authority*, *MA*) et un scénario exceptionnel de franchissement d'un arrêt ETCS (en anglais *Override EOA*). Seule l'utilisation du système ETCS niveau 2 est concernée par la présente étude.

La modélisation de ces deux scénarios comprend un modèle fonctionnel, qui décrit les principaux concepts de ces mouvements de trains, et un modèle qui précise les responsabilités de chaque intervenant (conducteur de train, agent de circulation, etc.). Ce deuxième modèle, dit de sécurité, est exprimé comme un modèle de contrôle d'accès, en utilisant l'outil *B4MSecure* (détaillé dans la section 4.2.4).

Dans la section 4.2, nous montrons comment notre contribution scientifique permet de réutiliser les techniques de la sécurité des systèmes d'information et ses politiques de contrôle d'accès pour la sécurité ferroviaire. Ensuite, nous décrivons notre étude de cas comprenant deux scénarios de règles d'exploitation ferroviaires du système ERTMS/ETCS. Nous présentons dans la section 4.4 le modèle fonctionnel et les modèles de sécurité en RBAC (Role Based Access Control), ainsi que leur transformation en spécifications B à l'aide de la plate-forme *B4MSecure* dans la section 4.5. La section 4.6 montre l'animation et la preuve formelle des spécifications résultantes en utilisant l'animateur de *ProB* et le prouveur de l'*Atelier B*. La section 4.7 discute et compare nos travaux de recherche

1. http ://www.lgv-est.com/
2. http ://www.securite-ferroviaire.fr/reglementations/principes-et-regles-dexploitation-du-systeme-etcs-particularites-en-cas-de
3. European Railway Agency : http ://www.era.europa.eu

par rapport aux travaux existants dans la littérature. Enfin, la section 4.8 résume les contributions apportées dans ce chapitre.

4.2 De la sécurité des SI vers la sécurité ferroviaire

Dans cette section, nous commençons par les principes des règles d'exploitation ferroviaires et ces procédures. L'étude de ces règles vise à extraire les concepts nécessaires pour les définir. Ces concepts servent de base pour la modélisation de ces règles. Par la suite, nous montrons que le modèle RBAC pour la modélisation des politiques de contrôle d'accès des systèmes d'information permet de répondre à notre problématique de modélisation des règles d'exploitation. En effet, les concepts de RBAC à savoir *utilisateur, rôle, permission, opération et objet* permettent la modélisation des scénarios issus de ces règles.

4.2.1 Règles d'exploitation ferroviaires

Les règles d'exploitation, que nous décrivons dans cette section, émanent du document d'exploitation [RFF, 2012]. Ce document a la vocation de décrire l'utilisation du système ETCS de niveau 2 sur la ligne à grande vitesse LGV-Est Européenne. En effet, les normes ferroviaires européennes ERTMS/ETCS peuvent provoquer de nombreux changements lorsque celles-ci s'appliquent sur des lignes nationales. En particulier, ce document reprend les principes et les dispositions, relatifs à l'utilisation de ce système, et apporte des précisions utiles pour la mise en œuvre sur cette ligne. Il reprend alors les prescriptions que doivent respecter les exploitants. Pour plus de précision, ce document a pour objet :

— d'expliquer le fonctionnement du système ETCS niveau 2,
— de définir la signalisation complémentaire mise en œuvre sur la ligne à grande vitesse Est Européenne,
— d'édicter les principes et règles d'exploitation du système ETCS niveau 2.

Les règles d'exploitation ferroviaires, décrites dans ce document exploitent les spécifications techniques d'interopérabilité « Sous système exploitation et gestion du trafic à grande vitesse ». Le respect de ces spécifications techniques vise à garantir l'interopérabilité du matériel roulant conforme à la norme ERTMS.

Remarque 4.1 *Cet aspect d'interopérabilité lié à différents contextes nationaux de différentes lignes nationales sera développé dans le chapitre suivant.*

Les règles d'exploitation ont pour fonction principale de définir les conditions pour les mouvements des trains. Au centre de ces procédures, on trouve l'article 413 de ce document qui résume les différentes autorisations de mise en mouvement :

Article 413. Autorisation de mise en mouvement

L'autorisation de mise en mouvement peut être donnée par :

— *une MA* [a],

— *un des ordres écrits suivants :*

- *de se mettre en marche en SR* [b],
- *de franchir un EOA* [c],
- *de se remettre en marche après un ≪ train trip ≫* [d].

a. Movement Authority. Autorisation donnée à un train de circuler vers un point donné en tant que mouvement supervisé.

b. Staff Responsible. Mode de déplacement utilisé dans les situations dégradées. Il est utilisable sous la responsabilité respective de l'agent-circulation et du conducteur.

c. End of Authority (fin d'autorisation de mouvement). Point auquel la vitesse but indiquée sur le DMI est égale à zéro. Cette fin d'autorisation de mouvement peut correspondre à un repère d'arrêt ETCS.

d. La fonction ≪ train trip ≫ provoque un freinage d'urgence irréversible et l'allumage du symbole ≪ train trip ≫. Elle se déclenche notamment en cas de franchissement intempestif de l'EOA, du repère d'arrêt ETCS en mode SR ou de défaut de fonctionnement des lecteurs de balise.

Cette autorisation de mise en mouvement correspond donc à deux cas principaux : d'une part la réception d'une MA et d'autre part un ensemble de fonctionnements exceptionnels qui contient :

— la mise en marche en SR qui est un mode intermédiaire où le déplacement est effectué à vitesse réduite sous la responsabilité des agents. Ce mode est notamment utilisé au démarrage d'une mission lorsque la localisation du train n'est pas encore sûre,

— le franchissement d'un EOA qui est une mesure exceptionnelle correspondant à l'autorisation de franchir la limite de mouvement donnée par la dernière MA, et

— la remise en marche après un ≪ train trip ≫ qui est un mode de freinage d'urgence.

Pour les besoins de notre étude, nous avons choisi d'étudier le mode normal de réception d'une MA et le mode exceptionnel de franchissement d'un EOA. L'étude de cas de ces deux types de mouvement sera abordée dans la section 4.3.

Sur la base de ces prescriptions, notre objectif est de comprendre ce qu'est une règle d'exploitation et de la décortiquer afin de proposer la modélisation la plus convenable. En effet, nous réfléchissons sur le contenu de chaque règle et les relations entre les contenus. Certaines règles ont une portée très générale telle que l'article 413 ci-dessus, d'autres imposent un protocole très précis. Ce protocole constitue un scénario d'échange d'information entre les acteurs du système.

Dans ce qui suit, nous n'allons pas citer toutes les règles d'exploitation en langage naturel qui correspondent à ces deux cas. Les conditions d'obtention d'une MA sont décrites

dans l'article 103 de l'annexe B. Cet article explique comment l'infrastructure ferroviaire est découpée en sous-parties et comment ces sous-parties (cantons) sont manipulées pour éviter la collision ou le rattrapage des trains. Il décrit aussi la règle d'exploitation relative au début de mission où la MA est allouée en OS [4], en FS [5] ou en SR [6] selon des contraintes.

En ce qui concerne le mode exceptionnel de franchissement d'un EOA, nous présentons l'article 502 suivant :

> ***Article 502. Fonction Override EOA***
> *Cette fonction permet, lorsqu'elle est activée par le conducteur, de franchir un repère d'arrêt ETCS ou un EOA en absence de MA.*
> *Le conducteur ne doit pas franchir un EOA avant d'avoir reçu, de l'agent-circulation, un ordre écrit l'autorisant à le faire.*
> *La fonction Override EOA, lorsqu'elle est activée, provoque la transition vers le mode SR.*

Les prescriptions des règles d'exploitation dans l'annexe B et ci-dessus se basent sur un ensemble d'autorisations. Pour analyser l'ensemble des procédures prescrites par les règles d'exploitation, nous devons répondre aux questions suivantes : Qui fait quoi ? Quelles sont les autorisations ? Qui sont les responsables de ces autorisations ? Quelles sont les ressources concernées par ces autorisations ? Quelles sont les actions activées par ces autorisations ? Quelles sont les contraintes liées à ces autorisations ?

Nous pouvons alors extraire les concepts essentiels à la modélisation de ces règles qui sont :

— les acteurs du système,
— les autorisations,
— le responsable de chaque autorisation,
— les ressources concernées,
— les actions autorisées, et
— les contraintes associées.

Afin d'amener l'ensemble des actions autorisées à un niveau de sécurité acceptable, des règles de sécurité sont définies d'une façon directe ou indirecte dans ces prescriptions. Le tableau 4.1 ci-dessous comporte quelques réponses apportées à ces questions en se basant sur les concepts déduits des règles de mouvement avec l'allocation d'une MA et en précisant les règles de sécurité qui sont liées. Les concepts mentionnés dans ce tableau sont le résultat d'analyse de l'article 103.1 de l'annexe B, tandis que la règle de sécurité est le résultat d'analyse de l'article 103.2 de la même annexe.

4. On Sight. (À vue). Mode qui autorise le conducteur à s'avancer en marche à vue.

5. Full Supervision. Mode technique dans lequel le train est supervisé en vitesse et déplacement. Marche normale attribuée au train.

6. Staff Responsible. Mode de déplacement utilisé dans les situations dégradées. Il est utilisable sous la responsabilité respective de l'agent-circulation et du conducteur.

Concepts	**Valeurs**	**Sources**
Acteurs du système	un RBC, un conducteur	Article 103.1
Autorisations	RBC autorise une MA en OS, RBC alloue (crée) une MA en FS, le conducteur fait avancer le train en marche à vue	Article 103.1
Responsables des autorisations	RBC, Conducteur	Article 103.1
Ressources concernées	MA, Train	Article 103.1
Actions autorisées	créer_MA, avancer_Train	Article 103.1
Contraintes associées	Après connexion par GSM-R, si toutes les conditions sont réunies	Article 103.1
Règles de sécurité	Sur un canton de voie, au maximum un train peut circuler	Cette règle est décrite implicitement dans l'Article 103.2

Tableau 4.1 – Concepts liés à l'allocation de MA

De même, le tableau 4.2 ci-dessous comporte les concepts déduits des règles de franchissement d'un EOA. Les concepts mentionnés dans ce tableau sont le résultat d'analyse de l'article 502, tandis que la règle de sécurité est extraite des deux articles 103.1 de l'annexe B et 502 ci-dessus.

Concepts	**Valeurs**	**Sources**
Acteurs du système	un conducteur, un agent de circulation	Article 502
Autorisations	L'agent de circulation autorise le franchissement par un ordre écrit	Article 502
Responsables des autorisations	L'agent de circulation	Article 502
Ressources concernées	Ordre écrit	Article 502

Suite page suivante ...

Concepts	Valeurs	Sources
Actions autorisées	créer_Ordre_écrit, franchir_EOA	Article 502
Contraintes associées	Le conducteur ne doit pas franchir un EOA avant d'avoir reçu, de l'agent de circulation, un ordre écrit l'autorisant à le faire	Article 502
Règles de sécurité	Il est impossible d'avoir un ordre écrit et une MA au même temps	Article 103.1 : Si, pour diverses raisons, le train ne peut pas être localisé en sécurité en début de mission, le RBC ne peut pas allouer de MA au train qui part alors en mode SR, après avoir reçu de l'agent circulation l'ordre écrit correspondant, jusqu'au passage d'un PI, qui permet alors au train de se localiser. Article 502 : Cette fonction permet, lorsqu'elle est activée par le conducteur, de franchir un repère d'arrêt ETCS ou un EOA en absence de MA.

Tableau 4.2 – Concepts liés au franchissement d'EOA (« Override EOA »)

Dans ce qui suit, nous présentons le modèle RBAC, le méta-modèle SecureUML qui est basé sur RBAC ainsi que la plate-forme *B4MSecure* fondé sur ces deux modèles. Ces derniers donnent des réponses aux questions posées dans l'analyse des règles d'exploitation. Ceci sera l'objet de la discussion dans la section 4.2.5.

4.2.2 Le modèle RBAC

Comme nous l'avons déjà introduit brièvement dans le chapitre 2, le modèle RBAC (Role Based Access Control) est un modèle de contrôle d'accès basé sur les rôles visant la protection des ressources d'un système [Ferraiolo et Kuhn, 1992]. Ce modèle a été publié en tant que NIST RBAC [Sandhu *et al.*, 2000], puis standardisé par l'*ANSI (American National Standard Institute)*/INCITS en 2004 [ANSI, 2004] et révisé en 2012 [ANSI, 2012].

La figure 4.1 montre les différentes entités du modèle ainsi que les relations entre elles [Ferraiolo *et al.*, 2001, Akoka *et al.*, 2010] :

— *USERS* : un utilisateur représente un agent humain (être humain ou personne) ou logiciel (incluant des machines, des réseaux ou des agents autonomes intelligents) qui peut accéder aux ressources du système. L'accès aux ressources par les utilisateurs doit être contrôlé par les permissions.

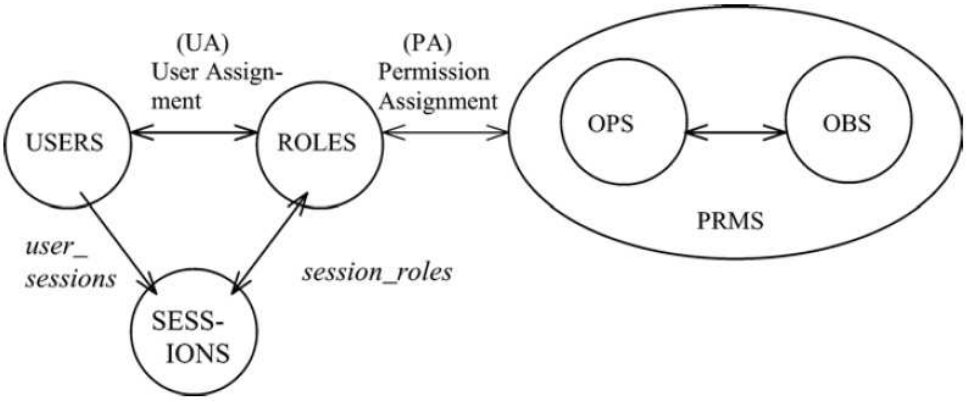

Figure 4.1 – Le modèle RBAC (ANSI-RBAC)

— *PERMISSIONS (PERMS)* : une permission est un privilège pour effectuer une opération sur un ou plusieurs objets RBAC protégés.
— *OBJECTS (OBS)* : un objet peut être une ressource du système soumis à un contrôle d'accès telle qu'un fichier, une imprimante, un terminal, un enregistrement de base de données, etc.
— *OPERATIONS (OPS)* : une opération est un type d'accès (action) qui peut être invoqué par un utilisateur du système sur une ressource. Une opération peut correspondre à toute action du système qui récupère ou communique des informations à un objet.
— *ROLES* : un rôle est une fonction d'une organisation avec une sémantique associée concernant l'autorité et la responsabilité. Un rôle RBAC permet de regrouper les permissions (privilèges) liées à cette fonction et de les accorder aux utilisateurs disposant de ce rôle. Le concept de rôle est central dans RBAC faisant le pont entre les utilisateurs et les permissions. En effet, deux relations sont définies : une relation (*PA : PermissionAssignment*) permettant d'accorder des permissions aux rôles et une relation (UA : UserAssignment) permettant d'attribuer ces rôles aux utilisateurs. Ces utilisateurs jouissent des privilèges selon les rôles qui leur sont accordés.
— *SESSIONS* : une session est une correspondance entre un utilisateur et un sous-ensemble actif de rôles qui lui sont affectés.

Vu la complexité croissante des systèmes d'information, plusieurs variantes du modèle RBAC ont été définies par *ANSI* [ANSI, 2004]. Chaque variante enrichit la précédente avec des concepts plus avancés. La première variante porte sur le noyau de RBAC (Core RBAC) avec les concepts *USERS, ROLES, SESSIONS, OPERATIONS, OBJECTS, PERMISSIONS, PermissionAssignment* et *UserAssignment*. Cette variante définit une collection minimale des éléments de RBAC, des ensembles d'éléments, et des relations afin d'atteindre un système complet de RBAC. La deuxième variante, appelée RBAC hiérarchique (Hierarchical RBAC), introduit la notion d'héritage de rôle (*Role Hierarchy*). L'héritage

des rôles est un moyen pour la structuration des rôles dans le but de mettre en œuvre les autorités et les responsabilités d'une organisation. L'introduction de cette notion permet ainsi de simplifier la définition des rôles et de faciliter l'héritage des permissions assignées aux rôles. Les deux autres variantes de RBAC sont celles qui permettent d'introduire les relations de séparation de devoirs, respectivement statique (Static Separation of Duty Relations) et dynamique (Dynamic Separation of Duty Relations). Ces deux dernières variantes permettent de gérer les conflits entre les rôles.

Remarque 4.2 *Le modèle de la figure 4.1 ci-dessus correspond au noyau du modèle RBAC (Core RBAC). Les différentes variantes de RBAC sont décrits formellement (définitions mathématiques) dans l'annexe A.*

Dans la littérature, des travaux de recherche ont proposé d'utiliser les méthodes formelles telles que Z et B pour la spécification du modèle RBAC. Les auteurs de [Abdallah et Khayat, 2006] ont proposé une formalisation des concepts de RBAC en Z afin d'éviter les ambiguïtés et les inconsistances d'interprétation. En revanche, les auteurs de [Huynh *et al.*, 2014] ont montré l'intérêt de valider les définitions mathématiques du standard RBAC par la méthode B et ont souligné un certain nombre d'erreurs techniques identifiés suite à l'animation avec ProB et la preuve avec Atelier B.

4.2.3 SecureUML

SecureUML est un langage de modélisation qui définit un vocabulaire pour annoter les modèles basés sur UML avec des informations pertinentes qui concernent le contrôle d'accès : utilisateur, rôle, permission, relation utilisateur-rôle et relation rôle-permission. Il est basé sur le modèle RBAC, décrit dans la section précédente, en y ajoutant des contraintes d'autorisation.

Comme le montre la figure 4.2, le méta-modèle de *SecureUML* est défini comme une extension du méta-modèle UML. Les concepts de RBAC sont représentés directement par des méta-classes *User*, *Role* et *Permission* et des associations entre ces méta-classes *RoleAssignment* et *PermissionAssignment*.

Dans le méta-modèle *SecureUML*, les ressources protégées sont représentées différemment. Elles ne sont pas définies avec une méta-classe dédiée comme les méta-classes *User, Role, Permission*. En effet, chaque élément de modèle UML, noté *ModelElement*, peut prendre le rôle d'une ressource protégée. La méta-classe *ResourceSet*, quant à elle, représente un ensemble d'éléments de modèle, défini par l'utilisateur, utilisé dans le but de définir les permissions et les contraintes d'autorisation. Une contrainte d'autorisation *Authorization-Constraint* est dérivée du type *Constraint* d'UML. Cette contrainte est attachée à la ressource protégée d'une façon directe ou indirecte via une permission. Une permission (*Permission*) est une relation d'association entre un rôle et *ModelElement* ou *ResourceSet*.

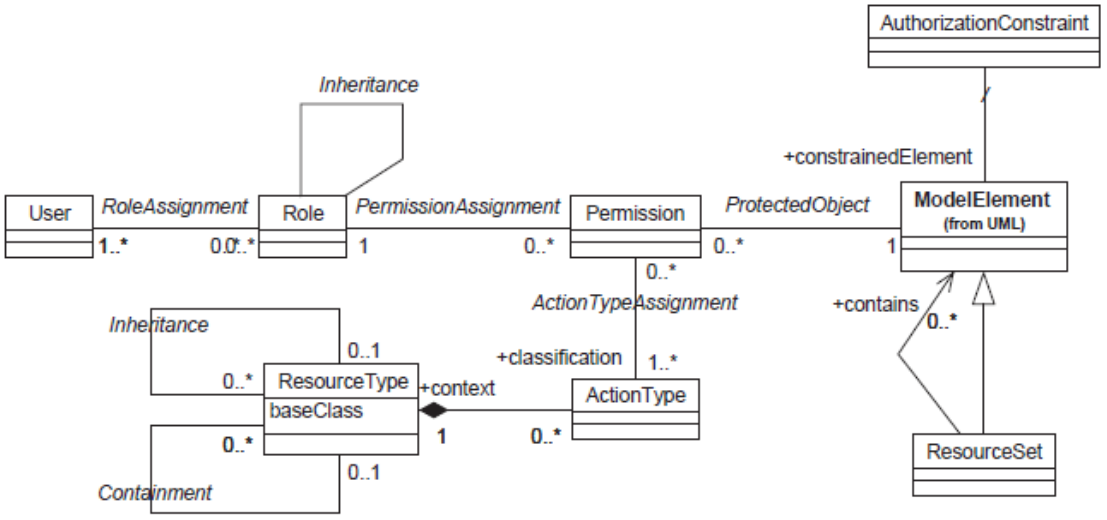

Figure 4.2 – Le méta-modèle de SecureUML [Lodderstedt *et al.*, 2002]

Afin de classifier la permission, sa sémantique est définie par un ou plusieurs types d'action (*ActionType*), via l'association *ActionTypeAssignment*. Chaque type d'action *ActionType* représente des opérations pertinentes définies sur un type particulier de ressources. Les actions exécuter, modifier, lire et écrire sont des exemples d'*ActionType*. L'ensemble des types d'action est défini en utilisant les types de ressources *ResourceType* [Lodderstedt *et al.*, 2002].

4.2.4 B4MSecure

B4MSecure[7] [Idani *et al.*, 2013] est une plate-forme Eclipse fondée sur l'approche IDM. Elle permet une modélisation conjointe en UML et B, basée sur le modèle RBAC, des aspects fonctionnels du système d'information ainsi que des politiques de contrôle d'accès. D'une part, la modélisation UML, fondée sur le profil *SecureUML*, offre des vues structurantes, compréhensibles et intuitives. D'autre part, la dérivation en B offre plus de rigueur et permet de tirer profit des outils de validation formelle. Du processus de dérivation de UML vers des spécifications formelles B, il en résulte deux modèles B [Idani *et al.*, 2014] :

— Un modèle fonctionnel B est issu du modèle fonctionnel UML, représenté par un diagramme de classes UML. Les spécifications formelles B du modèle fonctionnel peuvent être enrichies par des contraintes fonctionnelles et servent pour vérifier la correction du modèle fonctionnel.

— Un modèle, dit de sécurité, met en œuvre la politique de sécurité dans le but de contrôler l'accès aux diverses entités fonctionnelles. En UML, le modèle de sécurité est représenté sous forme de diagrammes de classes UML : un diagramme de classes pour les rôles, un diagramme de classes pour chaque entité fonctionnelle contenant

7. B4MSecure : http ://b4msecure.forge.imag.fr/

les différentes permissions de la politique de contrôle d'accès. Le modèle de sécurité en B inclut le modèle fonctionnel afin d'ajouter un filtre d'accès sur les opérations. En effet, chaque opération fonctionnelle (incontrôlée) est appelée par une opération de sécurité correspondante si l'utilisateur courant possède le droit (autorisation) de l'exécuter.

Les travaux de recherche menés dans le cadre du projet *SELKIS* montrent l'utilité et l'efficacité de cette plate-forme pour la validation formelle des scénarios dans le domaine d'un SI médical tout en détectant les séquences d'opérations malveillantes [Ledru *et al.*, 2011, Milhau *et al.*, 2011, Idani *et al.*, 2010].

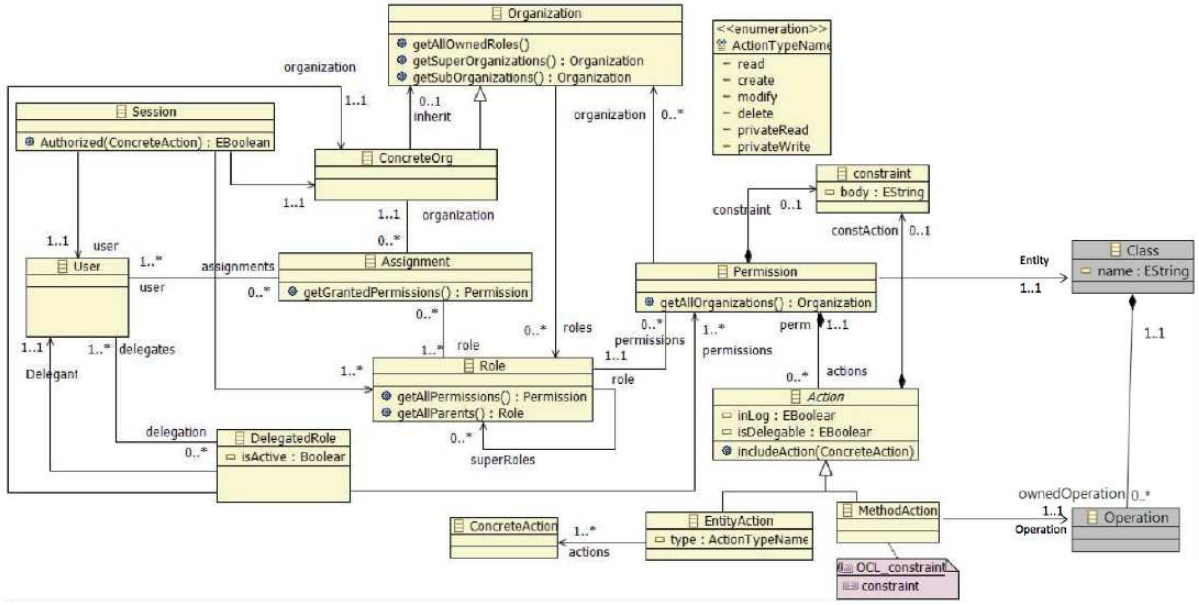

Figure 4.3 – Le méta-modèle de RBAC inspiré de SecureUML [Idani *et al.*, 2014]

La plate-forme *B4MSeure* a été développée dans le cadre du projet SELKIS [8]. L'objectif du projet *SELKIS* est de développer une méthode d'analyse et de conception de systèmes d'information sécurisés qui aborde les aspects fonctionnels et sécuritaires dès les premiers niveaux d'abstraction du développement. Compte tenu de cet objectif, cette méthode est fondée sur l'IDM employé auparavant qui permet de décrire un système d'information à trois niveaux d'abstraction : CIM, PIM et PSM [Akoka *et al.*, 2010].

Le méta-modèle de sécurité de la plate-forme *B4MSecure* (voir figure 4.3) s'inscrit dans le deuxième niveau d'abstraction (PIM). En effet, elle intègre les concepts de sécurité à un niveau indépendant des plate-formes (PIM). Ce méta-modèle comprend les concepts du modèle RBAC et est inspiré du méta-modèle SecureUML. Une permission est représentée par la méta-classe *Permission* et est associée à une classe fonctionnelle, issue du modèle

8. SEcure heaLth networKs Information Systems : une méthode de développement de systèmes d'information médicaux sécurisés de l'analyse des besoins à l'implémentation.

fonctionnel. Chaque permission peut contenir deux types d'actions :

— Une action de type *MethodAction* est rattachée à une seule opération de la classe fonctionnelle cible d'une permission autorisant l'invocation de cette opération.

— Les actions de type *EntityAction* permettent des types d'accès prédéfinis dans l'énumération *ActionTypeName* : *read, create, modify, delete, privateRead et privateModify*. Ce type d'action permet un accès aux attributs et aux associations d'une classe fonctionnelle en tant que constructeur (*create*), destructeur (*delete*), accesseur (*read, privateRead*) ou mutateur (*modify, privateModify*).

Le méta-modèle illustre les relations permission-rôle et utilisateur-rôle telles qu'elles sont définies dans le modèle RBAC. En effet, les utilisateurs (méta-classe *User*) sont affectés à des rôles (méta-classe *Role*) et ces rôles sont associés à des permissions (méta-classe *Permission*). D'où, un utilisateur, affecté à un rôle, est autorisé à effectuer une action via une permission à laquelle ce rôle est associé. Un autre concept défini dans le modèle RBAC, en particulier le modèle hiérarchique, concerne la hiérarchie des rôles. Ce concept est défini dans le méta-modèle par l'association réflexive *superRoles*. En outre, la méta-classe *Session* est aussi définie, permettant d'activer un ensemble de rôles parmi ceux qui sont affectés à l'utilisateur voulant se connecter [Idani *et al.*, 2014].

4.2.5 Discussion

De l'analyse des règles d'exploitation et de l'ensemble des questions posées pour accomplir leur modélisation, nous pouvons considérer l'interprétation suivante : une règle d'exploitation est vue comme une séquence d'autorisations d'exécuter certaines opérations (actions) par un ensemble d'utilisateurs (acteurs du système humains ou informatiques), en fonction des rôles (conducteur, agent de circulation, etc.) qui décrivent les responsabilités accordées aux utilisateurs.

L'étude du modèle RBAC et du méta-modèle SecureUML ainsi que l'approche de modélisation par *B4MSecure* permet de déduire les rapprochements suivants vis-à-vis des règles d'exploitation :

— Les acteurs correspondent aux utilisateurs ;

— Les responsabilités accordées aux acteurs correspondent aux rôles accordés aux utilisateurs ;

— La notion d'autorisation correspond à la notion sécuritaire de permission ;

— Les actions autorisées correspondent aux opérations ;

— Les ressources concernées correspondent aux objets ;

— Les contraintes qui permettent de délivrer une autorisation correspondent aux contraintes d'autorisation.

Notre objectif est de renforcer la sécurité des systèmes ferroviaires considérés critiques au sens *sécurité-innocuité* cherchant à éviter l'occurrence de défaillances catastrophiques en termes de vies humaines. D'où, l'adaptation d'une approche existante au domaine fer-

roviaire et en particulier pour les règles d'exploitation en ERTMS/ETCS est une solution méthodologique intéressante. En effet, nous illustrons par une étude de cas, dans ce qui suit, que la sécurité des systèmes d'information et les politiques de contrôle d'accès peuvent être utilisées pour des besoins de la sécurité des systèmes ferroviaires.

4.3 Étude de cas

Les règles d'exploitation définissent l'ensemble des interactions entre les systèmes «temps réels» embarqués et les opérateurs tels que le conducteur et l'agent de circulation, notamment dans les modes dégradés telles quelles sont introduites dans la section 4.2.1.

Dans la description des scénarios de notre cas d'étude, nous faisons référence au document [RFF, 2012] ainsi qu'aux spécifications européennes *SRS Subset 026* [UNISIG, 2006]. Ces dernières permettent d'ajouter quelques détails techniques manquants dans la description des règles d'exploitation du document [RFF, 2012].

4.3.1 Scénario nominal d'autorisation de mouvement (MA)

En ETCS niveau 2, le train reçoit une MA en « mode » nominal. L'obtention d'une MA permet d'utiliser la vitesse maximale de la ligne en mode FS (320km/h dans le cas de la ligne LGV-Est). Celle-ci est une autorisation donnée au train de circuler sur une distance donnée en tant que mouvement supervisé. La MA est mise à jour au fur et à mesure que le train avance.

La MA est caractérisée par (cf. paquet 15 de [UNISIG, 2006]) :

- **La section :** représente une distance par rapport au repère géographique du train. Elle est composée éventuellement de sous-sections. Une MA peut être appliquée sur une ou plusieurs sections. La dernière section est appelée la section de fin.

  Remarque 4.3 *La section est l'élément de découpage de la ligne du train, selon les spécifications SRS Subset 026. Une sous-section correspond au canton défini dans la section 4.2.1 par le document [RFF, 2012].*

- **La fin d'autorisation de mouvement :** (End Of Authority, EOA) est le « lieu » jusqu'où le train est autorisé à se mouvoir, en ce point la vitesse but indiquée sur l'interface conducteur-machine (Driver Machine Interface, DMI) est égale à zéro. Elle peut correspondre à un repère d'arrêt ETCS.

- **La vitesse cible à l'EOA :** est la vitesse autorisée à l'EOA. Lorsque la vitesse cible n'est pas nulle, l'EOA est appelé la limite de l'autorisation de mouvement (Limit Of Authority, LOA). Cette vitesse cible peut être limitée dans le temps.

- **Le point de danger :** est un point au-delà de l'EOA qui peut être atteint par l'extrémité avant du train sans risque d'une situation dangereuse.
- **Le délai d'attente :** est un délai qui peut être attaché à chaque section. Il provoque la révocation de l'itinéraire associé lorsque le train ne l'a pas encore emprunté.

Ce scénario, décrit dans [Ben Ayed *et al.*, 2014b, Ben Ayed *et al.*, 2014c], se déroule à l'aide d'interactions entre l'automatisme responsable de la gestion de sécurité à bord du train appelé *OnboardSafetyManagement*, l'automatisme responsable de la gestion de sécurité au sol appelé *TracksideSafetyManagement* et le conducteur qui suit les indications et les consignes via le DMI (voir la figure 4.4). Le DMI permet alors la communication entre le système à bord du train et le conducteur. Ce dernier doit respecter les indications et acquitter les consignes sur le DMI. Il peut aussi informer le système en entrant des informations.

MA.1 L'automatisme responsable de la gestion de sécurité à bord du train (*OnboardSafetyManagement*) demande une *MA* au sous-système au sol (*TracksideSystem*).

MA.2 L'automatisme responsable de la gestion de sécurité au sol (*TracksideSafetyManagement*) reçoit la demande de *MA* et propose une *MA* après l'avoir créée. Il peut également la modifier et/ou la supprimer.

MA.3 L'automatisme responsable de la gestion de sécurité à bord du train (*OnboardSafetyManagement*) reçoit la *MA* proposée du sous-système au sol (*TracksideSystem*), la valide et l'affiche sur le *Driver Machine Interface (DMI)*.

MA.4 Le conducteur (*Driver*) lit et suit les consignes relatives à la MA validée.

Remarque 4.4 *Quelques détails ne sont pas parfois pris en compte dans le but de simplifier la modélisation et de ne pas avoir plusieurs contraintes à gérer essentiellement lors de la traduction en spécifications B. Le mode ETCS (FS, OS, SR, etc.) n'est pas spécifié. De plus, dans la construction de l'ensemble des rôles, nous n'entrons pas dans le détail des différents composants constituant la partie bord (par exemple EVC*[9]*) et de la partie sol (par exemple RBC). D'où, le dénomination d'OnboardSafetyManagement et de TracksideSafetyManagement comme des rôles abstraits.*

Maintenant que nous avons présenté le scénario nominal de l'autorisation d'une MA, nous passons au scénario exceptionnel « Override EOA ».

4.3.2 Scénario exceptionnel « Override EOA »

« *Override EOA* » est un scénario déclenché par le conducteur dans des situations dégradées spécifiques en l'absence de MA. Lorsqu'il est activé par le conducteur, ce scénario

9. European Vital Computer (Ordinateur européen de sécurité) : calculateur de bord qui supervise la marche du train en fonction des données Sol et Bord.

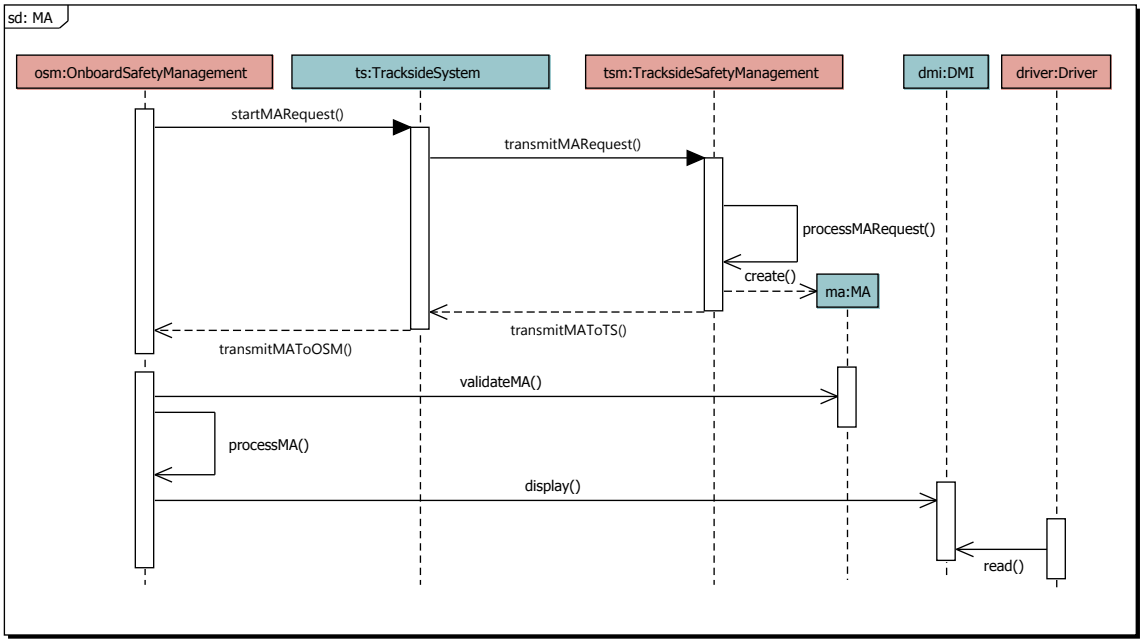

Figure 4.4 – Le diagramme de séquence du scénario nominal MA

permet à un train de franchir un repère d'arrêt ETCS ou un EOA après avoir reçu de l'agent de circulation une autorisation « Override EOA » permettant de désactiver certaines protections. Cette autorisation est donnée sous forme d'un ordre écrit. Ce dernier est un message de sécurité délivré par l'agent de circulation au conducteur dans le but de fournir des instructions. Il peut être délivré physiquement ou bien faire l'objet d'une transmission verbale par téléphone ou par radio sol train selon les modalités d'application de la réglementation technique de sécurité relative à la communication. Il existe plusieurs types d'ordres écrits d'ETCS01 à ETCS07. Nous pouvons citer par exemple l'ordre écrit ETCS01 d'autorisation de franchir un EOA traité dans notre étude de cas, l'ordre écrit ETCS03 d'obligation de rester à l'arrêt, l'ordre écrit ETCS04 d'annulation de l'ordre écrit ETCS03, etc. Un ordre écrit contient au minimum le type de l'autorisation, le numéro de l'autorisation, l'heure et la date de sa délivrance, le poste qui le délivre, à quel train il s'adresse, à quel endroit il s'applique et une indication claire, précise et sans ambiguïté des actions à effectuer. Le scénario de « Override EOA » est effectué comme suit (voir la figure 4.5) :

OverrideEOA.1 Le conducteur (*Driver*) demande un « Override EOA » à l'agent de circulation (*TrafficAgent*).

OverrideEOA.2 L'agent de circulation (*TrafficAgent*) accorde l'autorisation au conducteur (*Driver*) de franchir l'EOA via l'ordre écrit ETCS01. Il peut aussi le modifier et/ou le supprimer.

OverrideEOA.3 Le conducteur (*Driver*) appuie sur le bouton « Override EOA » et suit les indications et les consignes affichées sur le DMI suite au déclenchement de cette

procédure.

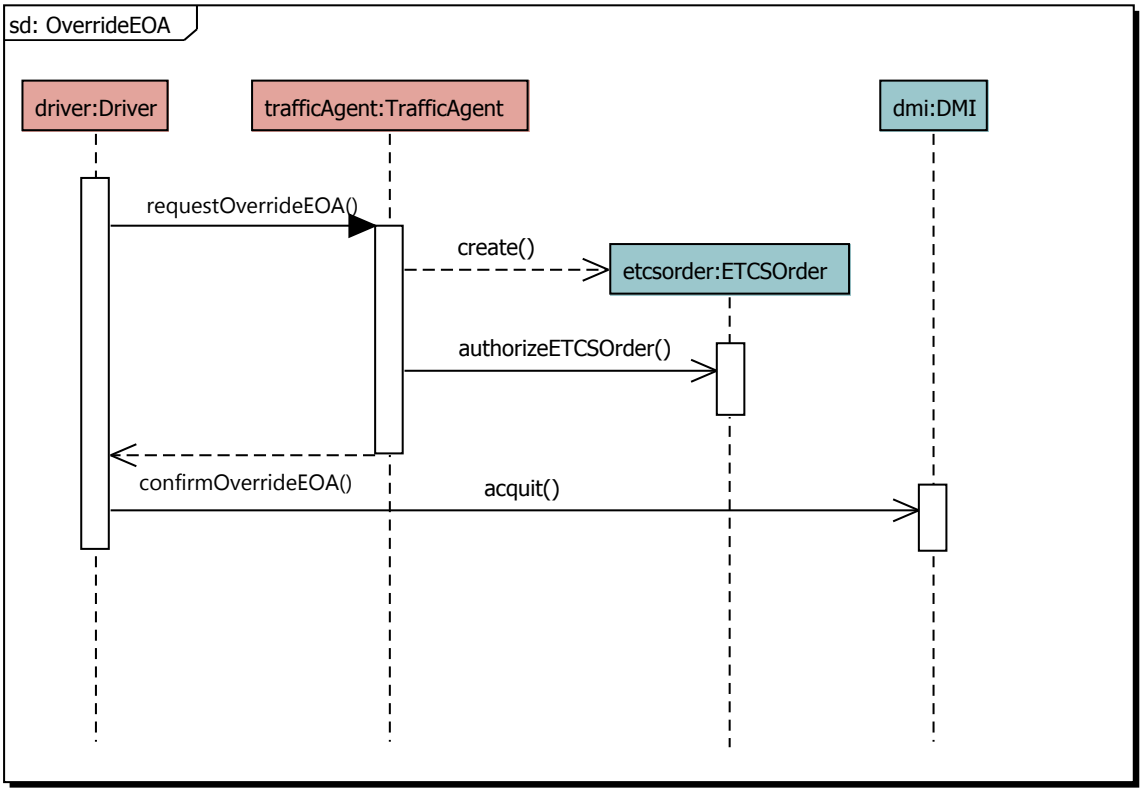

Figure 4.5 – Le diagramme de séquence du scénario Exceptionnel « Override EOA »

Les ordres et les instructions, notamment de MA et de « Override EOA », sont affichés sur le DMI sous forme de messages textuels ou de symboles.

Notre étude de cas basée sur ces deux scénarios révèle l'existence d'interactions entre le système et les agents agissant sur le système. Pour bien modéliser ces interactions, une séparation de la fonctionnalité du système et de la sécurité est nécessaire. Les notions de la sécurité-confidentialité (*Security*) et de la sécurité-innocuité (Safety) étant déjà introduites dans le chapitre 2, nous passons à l'intérêt de l'utilisation d'UML enrichi par le profil RBAC pour la vérification et la validation des règles d'exploitation ferroviaires.

4.4 Modélisation

Comme nous l'avons déjà évoqué dans le chapitre 3, le couplage des méthodes formelles, en l'occurrence la méthode B, et semi-formelles telles qu'UML est un sujet étudié depuis des années grâce à leurs complémentarités.

UML permet de modéliser une application logicielle selon une vision orientée objet. Il offre une modélisation d'un système au travers de plusieurs types de diagrammes

représentant autant de vues distinctes pour représenter des concepts particuliers du système. Les diagrammes UML utilisés dans nos travaux sont les diagrammes de classes permettant de décrire le système et ses fonctionnalités. L'aspect dynamique permettant la description du comportement du système n'est pas abordé et constitue une perspective de nos travaux.

4.4.1 Modèle fonctionnel

Le modèle fonctionnel permet de décrire les entités du système impliquées dans la réalisation des scénarios, les caractéristiques de chaque entité, les opérations effectuées sur chaque entité et les relations entre les entités. Les principes de modélisation des diagrammes de classes UML se basent sur les classes, attributs de classes, opérations de classes, les relations (agrégation, composition, généralisation/spécialisation), ainsi que sur les associations et les classes associatives qui permettent la modélisation fonctionnelle désirée. En effet, les classes sont utilisées pour la modélisation des entités. Les attributs de classes et les opérations de classes sont utilisés respectivement pour la modélisation des caractéristiques des entités et les opérations effectuées sur les entités. Les relations (agrégation, composition, etc.), les associations et les classes associatives sont utilisées pour la modélisation des relations entre les entités.

La figure 4.6 représente le système ETCS (la classe ETCSSystem) contenant le sous-système à bord (la classe OnboardSystem) qui fait partie du train ETCS (la classe TrainETCS) et le sous-système au sol (la classe TracksideSystem). L'autorisation de mouvement et l'ordre écrit sont modélisés par deux classes (les classes MA et ETCSOrder) contenant les attributs qui les caractérisent. Chacun est associé à un train ETCS et est affiché sous forme de consignes sur le DMI, qui fait partie du système à bord, après son traitement et sa validation. Comme décrit dans la section 4.3.1, une MA est transmise par le système au sol et elle est composée d'une ou plusieurs sections. Pour des raisons de sécurité, il ne peut y avoir qu'un seul train sur une section donnée.

4.4.2 Politique de sécurité

Des permissions sont accordées aux usagers du système selon les rôles qui leur sont attribués. Les concepts de permission et de rôle sont modélisés conformément au modèle RBAC, et ce, au travers d'une extension d'UML inspirée du profil SecureUML [Lodderstedt *et al.*, 2002]. En effet, une action n'est opérationnelle pour un rôle que si ce rôle a la permission de l'exécuter. Selon le profil RBAC, un rôle est représenté par une classe stéréotypée *Role* et une permission est représentée par une classe associative stéréotypée ≪ Permission ≫ entre le rôle et l'entité du système sur laquelle s'applique cette permission.

Les modèles de sécurité de notre étude de cas enrichissent le modèle fonctionnel par la modélisation de quatre rôles, à savoir le conducteur (Driver), l'agent de circulation

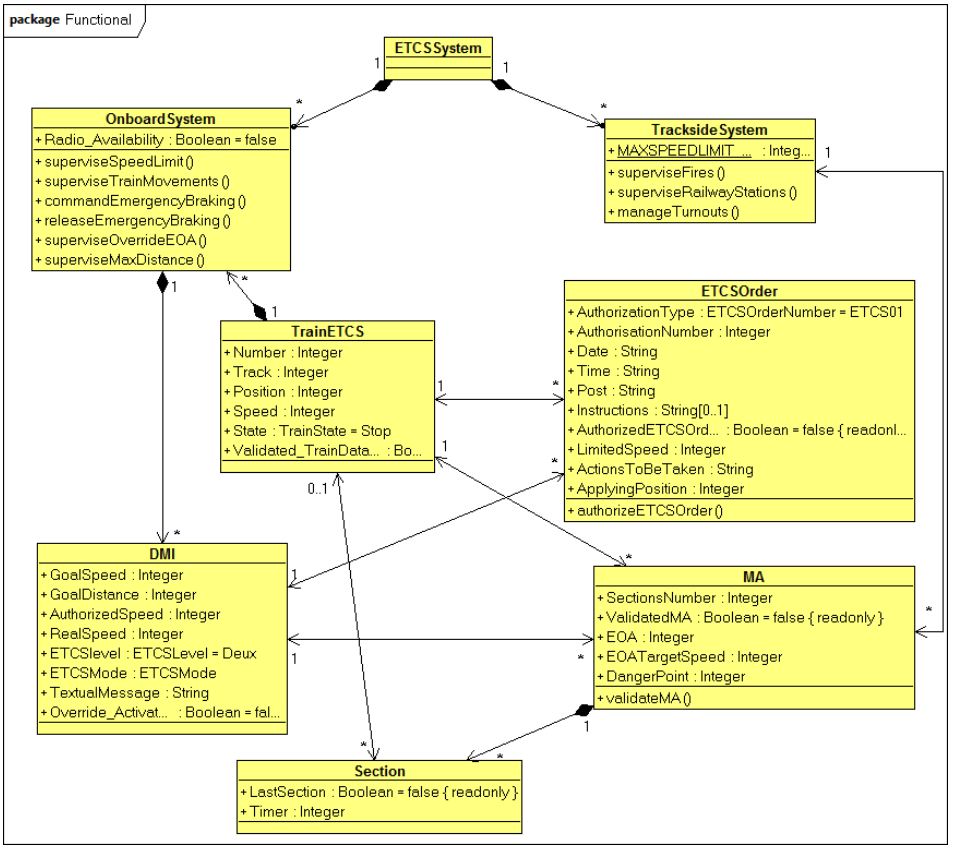

Figure 4.6 – Le modèle fonctionnel

(TrafficAgent), l'automatisme de gestion de sécurité à bord (*OnboardSafetyManagement*) et l'automatisme de gestion de sécurité au sol (*TracksideSafetyManagement*). La figure 4.7 est un modèle de sécurité décrivant les permissions accordées aux rôles sur une MA. En l'occurrence, le *TracksideSafetyManagement* peut créer une MA, la supprimer et/ou la modifier et l'*OnboardSafetyManagement* peut la valider une fois créée par *TracksideSafetyManagement*. De la même manière, nous décrivons pour chaque entité du modèle fonctionnel un modèle de sécurité contenant les permissions accordées aux rôles pouvant agir sur cette dernière.

4.5 Transformation en spécifications B

Dans le cadre du couplage des notations UML et B, la plate-forme dispose d'une base de règles de transformations de UML vers B. Dans cette section, nous illustrons la transformation du modèle fonctionnel ainsi que des modèles de sécurité en spécifications B.

La transformation automatique des modèles avec la plate-forme *B4MSecure* permet d'obtenir une unique machine B abstraite pour le modèle fonctionnel appelée « Functional » et une unique machine B pour tous les modèles de sécurité appelée « RBAC_Model ».

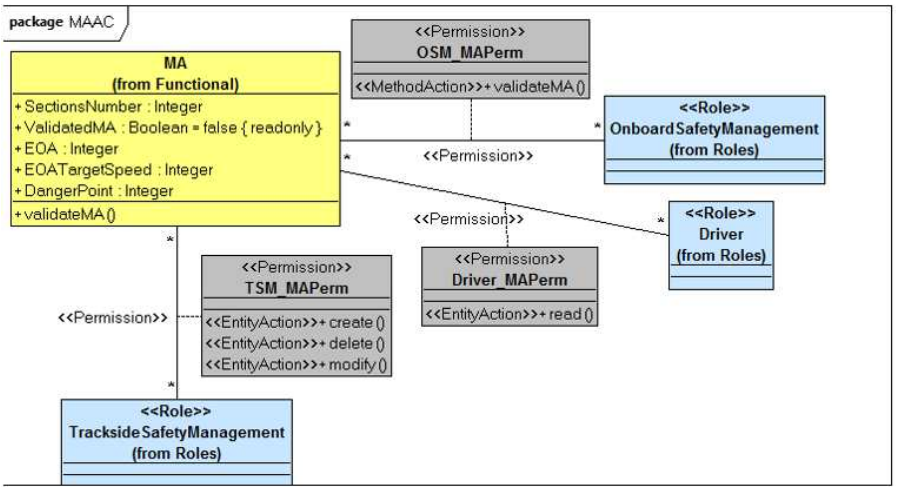

Figure 4.7 – Permissions associées à la classe MA

La traduction du modèle fonctionnel est inspirée des approches de transformation existantes de UML vers B [Idani *et al.*, 2010].

La spécification B décrit les ensembles (*SETS*), les variables abstraites (*ABSTRACT_VARIABLES*), les invariants (*INVARIANT*) et les opérations (*OPERATIONS*). La partie gauche de la figure 4.8 donne quelques extraits du modèle fonctionnel généré automatiquement par la plate-forme *B4MSecure*. Les classes *MA* et *ETCSOrder* sont transformées respectivement en un ensemble *MA_AS* et un ensemble *ETCSORDER* (déclarés dans la clause SETS) qui représentent les instances possibles de ces classes et les variables abstraites *MA* et *ETCSOrder* (déclarées dans la clause *ABSTRACT_VARIABLES*) qui représentent les instances existantes de ces classes. Les invariants des lignes 15 et 16, du modèle fonctionnel dans la partie gauche de la figure 4.8, montrent l'inclusion de *MA* dans *MA_AS* et de *ETCSOrder* dans *ETCSORDER*. Les attributs des classes (*MA__ValidatedMA* et *ETCSOrder__AuthorizedETCSOrder*) et les associations (*MAOfTrainETCS* entre la classe *MA* et *TrainETCS* et *ETCSOrderOfTrainETCS* entre la classe *ETCSOrder* et *TrainETCS*) sont représentés sous forme de variables (lignes 9, 10, 11 et 12 à gauche de la figure 4.8). Les invariants des lignes 17 et 18 sont relatifs aux attributs dans le but de spécifier leurs types (le type booléen « *BOOL* »). Les invariants des lignes 19 et 20 sont relatifs aux associations pour spécifier leurs multiplicités. Les méthodes des classes, ainsi que les constructeurs, les destructeurs, les accesseurs (getters) et les mutateurs (setters) sont transformés en opérations dans la partie dynamique du modèle B. Ces opérations respectent les invariants de types.

La spécification des opérations peut être enrichie par des pré-conditions et des substitutions, de même la machine peut se voir complétée par des invariants. Cette tâche peut être préalablement effectuée dans le modèle UML afin de générer automatiquement

des opérations bien spécifiées et des invariants de la machine autres que les invariants de types. En effet, la plate-forme *B4MSecure* permet d'ajouter des annotations en B sur le modèle UML pour spécifier des invariants et sur les méthodes des classes pour spécifier des pré-conditions et des substitutions d'opération. Les pré-conditions des lignes 31 et 36 et les substitutions d'opération des lignes 32 et 37, du modèle fonctionnel à gauche de la figure 4.8, peuvent être ajoutées dans le modèle UML sous forme d'annotations respectivement sur les méthodes *validateMA* et *authorizeETCSOrder* des classes *MA* et *ETCSOrder*.

```
1 Machine
2     Functional
3 SETS
4     MA_AS;
5     ETCSORDER; ...
6 ABSTRACT_VARIABLES
7     MA,
8     ETCSOrder,
9     MA__ValidatedMA,
10     ETCSOrder__AuthorizedETCSOrder,
11     MAOfTrainETCS,
12     ETCSOrderOfTrainETCS,
13     ...
14 INVARIANT
15     MA ⊆ MA_AS
16     ∧ ETCSOrder ⊆ ETCSORDER
17     ∧ MA__ValidatedMA ∈ MA → BOOL
18   ∧ ETCSOrder__AuthorizedETCSOrder ∈ ETCSOrder → BOOL
19     ∧ MAOfTrainETCS ∈ MA ⤖ TrainETCS
20     ∧ ETCSOrderOfTrainETCS ∈ ETCSOrder ⤖ TrainETCS
21     ∧ ...
22 INITIALISATION
23     MA := ∅
24     || ETCSOrder := ∅
25     || MA__ValidatedMA := ∅
26     || ETCSOrder__AuthorizedETCSOrder := ∅
27     ||...
28 OPERATIONS
29 MA__validateMA(Instance)=
30   PRE Instance ∈ MA
31     ∧ MA__ValidatedMA(Instance) = FALSE
32   THEN MA__ValidatedMA(Instance) := TRUE
33   END;
34 ETCSOrder__authorizeETCSOrder(Instance)=
35  PRE Instance ∈ ETCSOrder
36     ∧ ETCSOrder__AuthorizedETCSOrder(Instance) = FALSE
37  THEN ETCSOrder__AuthorizedETCSOrder(Instance) := TRUE
38  END; ...
39 END
```

```
1 Machine
2     RBAC_Model
3 INCLUDES
4     Functional, UserAssignments
5 SEES
6     ContextMachine
7 SETS
8     ENTITIES={MA_Label, ETCSOrder_Label, ...};
9  Attributes={MA__ValidatedMA_Label,ETCSOrder__AuthorizedETCSOrder_Label...};
10 Operations={MA__validateMA_Label, ETCSOrder__authorizeETCSOrder_Label...};
11 PERMISSIONS={OSM_MAPerm,Driver_MAPerm,TSM_MAPerm,TA_ETCSOrderPerm}
12 VARIABLES
13     PermissionAssignment, isPermitted, ...
14 INVARIANT
15     PermissionAssignment ∈ PERMISSIONS → (ROLES *ENTITIES)
16     ∧ isPermitted ∈ ROLES ↔ Operations ...
17 INITIALISATION
18     PermissionAssignment :=
19     {(OSM_MAPerm↦→(OnboardSafetyManagement↦→MA_Label)),
20     (Driver_MAPerm↦→(Driver↦→MA_Label)),
21     (TSM_MAPerm↦→(TracksideSafetyManagement↦→MA_Label)),
22     (TA_ETCSOrderPerm↦→(TrafficAgent↦→ETCSOrder_Label)),...}
23 OPERATIONS
24 secure_MA__validateMA(Instance)=
25   PRE Instance ∈ MA ∧ MA__ValidatedMA(Instance) = FALSE
26   THEN SELECT MA__validateMA_Label ∈ isPermitted[currentRole]
27     THEN MA__validateMA(Instance)
28     END
29   END;
30 secure_ETCSOrder__authorizeETCSOrder(Instance)=
31   PRE Instance ∈ ETCSOrder ∧
32     ETCSOrder__AuthorizedETCSOrder(Instance)= FALSE
33 THEN SELECT ETCSOrder__authorizeETCSOrder_Label∈isPermitted[currentRole]
34     THEN ETCSOrder__authorizeETCSOrder(Instance)
35     END
36   END; ...
37 END
```

Figure 4.8 – Machines ≪ *Functional* ≫ et ≪ *RBAC_Model* ≫

La machine associée au modèle de sécurité (à droite de la figure 4.8) inclut via un opérateur *INCLUDES* la machine associée au modèle fonctionnel pour vérifier les propriétés de sécurité. La clause *INCLUDES* permet de construire de manière modulaire des machines abstraites (ou des raffinements). Elle permet de regrouper dans une machine abstraite (ou un raffinement), les constituants (ensembles, constantes et variables) d'instances de machines ainsi que leurs propriétés (clause *PROPERTIES* et *INVARIANT*), afin de créer un composant enrichi à l'aide d'autres machines abstraites.

Cette machine appelée ≪ RBAC_Model ≫ ajoute des variables relatives aux permissions. Par exemple, *PermissionAssignment* est une fonction totale de l'ensemble *PERMISSIONS*

vers le produit cartésien *(ROLES * ENTITIES)*, *isPermitted* est une relation entre les deux ensembles *ROLES* et *Operations*. *PERMISSIONS*, *ENTITIES* et *Operations* sont des ensembles définis dans *RBAC_Model*, alors que *ROLES* est un ensemble défini dans la machine incluse *UserAssignments*.

Elle ajoute aussi des contraintes d'autorisation d'accès aux opérations du modèle fonctionnel. Elle associe une opération sécurisée correspondant à une opération fonctionnelle pour vérifier si l'utilisateur courant (le rôle) possède une permission d'effectuer cette opération. Afin de limiter l'accès à une opération fonctionnelle, un prédicat est ajouté dans l'opération sécurisée du modèle de sécurité qui vérifie si cette opération figure parmi les opérations permises au rôle de l'utilisateur courant *currentRole*. Ces prédicats apparaissent dans les lignes 26 et 33 à droite de la figure 4.8 pour les opérations d'autorisation de *MA* et d'*ETCSOrder*.

4.6 Vérification et validation des modèles formels

La vérification et la validation de modèles basées sur la méthode B ont pour objectif de garantir que les modèles respectent bien les exigences de départ. Nous nous focalisons sur la validation formelle des spécifications B générées automatiquement par la plate-forme *B4MSecure* au moyen des outils *ProB* et *Atelier B*.

4.6.1 *Atelier B* et *ProB*

Nous avons utilisé l'Atelier B pour prouver les spécifications B générées. Nous avons utilisé le prouveur automatique et un ensemble de commandes du prouveur interactif pour prouver les obligations de preuves restantes.

Le model checking est une technique de vérification formelle permettant de trouver, si elle existe, une séquence d'opérations qui, à partir d'un état initial, conduit à un état qui viole l'invariant (ou présente une autre erreur, comme l'interblocage). Cette séquence représente un contre-exemple de violation d'invariant. Le model checking permet également de garantir formellement l'absence d'erreurs lorsque tous les états ont été explorés sans trouver un contre-exemple. Ceci, bien évidemment, ne se produira que si l'espace d'état est fini et gérable par la machine. Dans le cas des espaces infinis, le model checking peut explorer l'espace d'état jusqu'à ce qu'il trouve une erreur ou génère un dépassement de mémoire. Ce qui est le cas des spécifications B générées automatiquement en utilisant la plate-forme *B4MSecure*. Le model checking de *ProB* sur le cas étudié ne parvient pas à explorer tout l'espace d'état et génère un message de dépassement de mémoire. Pour cette raison, nous avons utilisé *ProB* en tant qu'animateur pour tester nos scénarios. Une telle animation simule l'évolution de l'état du système. Le premier état est calculé à partir de l'initialisation. Les états suivants dépendent des opérations permises exécutées

par l'utilisateur et dont la pré-condition est vérifiée. Nous considérons l'animation comme une technique de validation complémentaire à la preuve par *Atelier B* du fait qu'elle permet de jouer étape-par-étape les scénarios préalablement définis par des documents de spécification. En partant de l'initialisation, nous pouvons tester l'accessibilité de l'opération suivante parmi les opérations permises en suivant la spécification du scénario. Par ailleurs, après avoir utilisé le prouveur automatique comme une première étape de preuve pour la génération des obligations de preuves démontrées automatiquement, nous pouvons aussi utiliser l'animation dans le but de détecter les états du système qui violent l'invariant et ainsi faciliter la poursuite de la deuxième étape de la preuve dite interactive.

4.6.2 Validation formelle

La transformation des modèles UML en spécifications B génère des opérations B qui respectent les invariants de types (les types des attributs de classes, la multiplicité des associations entre les classes, etc.) dans le modèle fonctionnel et la politique de sécurité (les rôles, les permissions, etc.) dans le modèle de sécurité. L'animation avec ProB permet de valider, suivant les droits de chaque rôle, certaines opérations du modèle fonctionnel et de vérifier la séquence d'opérations qui construit les deux scénarios. Pour la validation de nos exigences de sécurité ferroviaire, il est judicieux d'ajouter des contraintes tant pour le modèle fonctionnel que pour le modèle de sécurité afin d'assurer la conformité des modèles B enrichis par ces exigences. L'animation de la séquence d'opérations avec ProB révèle le besoin d'ajouter des contraintes de sécurité ferroviaire à certaines opérations sous forme de pré-conditions ou bien à la spécification en sa globalité sous forme d'invariants. Ces contraintes peuvent être exprimées sous forme d'annotations en B dans les modèles UML afin qu'elles soient transformées automatiquement par la plate-forme *B4MSecure* dans les spécifications générées en B.

Comme mentionné dans la section précédente, le modèle de sécurité inclut le modèle fonctionnel et chaque opération sécurisée du modèle de sécurité est associée à une opération du modèle fonctionnel sur laquelle s'ajoutent des contraintes d'autorisation d'accès. De ce fait, l'opération du modèle de sécurité appelante doit contenir les mêmes pré-conditions que l'opération du modèle fonctionnel appelée.

Nous rappelons que la fonction « Override EOA » permet, lorsqu'elle est activée par le conducteur, de franchir un repère d'arrêt ETCS ou un EOA en l'absence de MA. Le conducteur ne doit pas franchir un EOA avant d'avoir reçu, de l'agent de circulation, un ordre écrit l'autorisant à le faire. Nous rappelons aussi qu'à l'EOA la vitesse but indiquée sur l'interface conducteur-machine DMI est égale à zéro. Elle peut correspondre à un repère d'arrêt ETCS, c'est-à-dire le train est en état d'arrêt. Autrement dit, le train ne peut avancer que s'il possède une MA validée (par *OnboardSafetyManagement*), ou bien un

ordre écrit de franchissement d'arrêt ETCS ou un EOA, autorisé par l'agent de circulation et activé par le conducteur, en l'absence de MA. Ceci peut être exprimé par l'invariant de sécurité encadré dans la clause INVARIANT de la figure 4.9 :

```
SETS
   TrainState={Advance,Stop}
INVARIANT
   MAOfTrainETCS ∈ MA ⤖ TrainETCS
   ∧ MA__ValidatedMA ∈ MA → BOOL
   ∧ ETCSOrderOfTrainETCS ∈ ETCSOrder ⤖ TrainETCS
   ∧ ETCSOrder__AuthorizedETCSOrder ∈ ETCSOrder → BOOL
   ∧ TrainETCS__State ∈ TrainETCS → TrainState
   ∧ DMI__Override_Activated ∈ DMI → BOOL
   ∧ OnboardSystem_in_TrainETCS ∈ OnboardSystem ⤖ TrainETCS
   ∧ DMI_in_OnboardSystem ∈ DMI ⤖ OnboardSystem
∧ ∀ (tr).(tr∈TrainETCS ∧ ((MAOfTrainETCS^-1[{tr}]≠∅ ∧
MA__ValidatedMA(MAOfTrainETCS^-1(tr))=TRUE) ∨ (MAOfTrainETCS^-1[{tr}]=∅ ∧
ETCSOrderOfTrainETCS^-1[{tr}]≠∅ ∧
ETCSOrder__AuthorizedETCSOrder(ETCSOrderOfTrainETCS^-1(tr))=TRUE ∧
DMI__Override_Activated(DMI_in_OnboardSystem^-1(OnboardSystem_in_TrainETCS^-1(tr)))
=TRUE)) ⇒ TrainETCS__State(tr)=Advance)
```

Figure 4.9 – Propriété de sécurité ajoutée sous forme d'un invariant

L'ajout de cette propriété de sécurité sous forme d'un invariant avec une quantification universelle sur l'ensemble des trains nécessite la modification de plusieurs opérations afin de le respecter. L'opération « DMI__acquit » (Figure 4.10) permet d'activer « Override EOA » en l'absence de MA et en présence d'un ordre de franchissement autorisé, tandis que l'opération « DMI__moving » (Figure 4.11) permet de changer l'état de train (en marche) s'il possède une MA validée, ou bien un ordre écrit de franchissement autorisé et activé, en l'absence de MA.

La preuve par *Atelier B* parvient à démontrer automatiquement 393 obligations de preuve parmi 441 obligations de preuve générées. Les 48 obligations de preuve restantes ont été démontrées par le prouveur interactif d'*Atelier B* [Clearsy System Engineering, 2011a] en utilisant quelques commandes interactives telles que **dc** (DoCases) pour la preuve par cas qui sépare le cas général du cas spécifié dans l'opération à cause de la quantification universelle sur l'ensemble des trains et **ph** (particularize hypothesis) pour particulariser les hypothèses de la partie droite de l'implication et démontrer sa partie gauche. Les obligations de preuve, relatives aux invariants structurels ainsi que les invariants additionnels de sécurité ferroviaire, sont alors prouvées à 100%.

```
DMI__acquit(Instance, train) =
  PRE
    Instance∈DMI   ∧   train∈TrainETCS   ∧
    train=OnboardSystem_in_TrainETCS(DMI_in_OnboardSystem(Instance)) ∧
    MAOfTrainETCS^-1[{train}]=∅ ∧ ETCSOrderOfTrainETCS^-1[{train}]≠∅   ∧
    ETCSOrder__AuthorizedETCSOrder(ETCSOrderOfTrainETCS^-1(train)) = TRUE  ∧
    DMI__Override_Activated(Instance)=FALSE
 THEN
    DMI__Override_Activated(Instance) := TRUE
    || TrainETCS__State(train):=Advance
 END;
```

Figure 4.10 – Modification de l'opération « DMI__acquit » suite à l'ajout de l'invariant

```
DMI__moving(Instance, train) =
  PRE
    Instance∈DMI   ∧   train∈TrainETCS   ∧
    train=OnboardSystem_in_TrainETCS(DMI_in_OnboardSystem(Instance))   ∧
    ((MAOfTrainETCS^-1[{train}]≠∅ ∧ MA__ValidatedMA(MAOfTrainETCS^-1(train))=TRUE)
∨ (MAOfTrainETCS^-1[{train}]=∅ ∧ ETCSOrderOfTrainETCS^-1[{train}]≠∅ ∧
ETCSOrder__AuthorizedETCSOrder(ETCSOrderOfTrainETCS^-1(train)) = TRUE ∧
DMI__Override_Activated(DMI_in_OnboardSystem^-1(OnboardSystem_in_TrainETCS^-1(train)))
=TRUE))
 THEN
    TrainETCS__State(train):=Advance
 END;
```

Figure 4.11 – Modification de l'opération « DMI__moving » suite à l'ajout de l'invariant

Le modèle de sécurité contient un ensemble de définitions pour pouvoir extraire toutes les opérations permises accordées aux rôles, entre autres celles qui sont dues aux droits de création par les constructeurs, aux droits de destruction par les destructeurs, aux droits de lecture par les accesseurs (getters) et aux droits de modification par les mutateurs (setters). Le modèle de sécurité est considéré comme un filtre qui ne propose pas de nouvelles opérations et n'ajoute aucun comportement supplémentaire. Il permet donc de limiter l'accès à certaines opérations en fonction du rôle. Les principaux invariants à respecter sont ceux qui sont liés aux concepts de RBAC, ce qui est déjà réalisé par la transformation automatique à l'aide de l'outil *B4MSecure*.
L'animation avec *ProB* permet de construire correctement l'ensemble des permissions à partir des définitions. Une première animation permet de vérifier le comportement des scénarios tels qu'ils sont définis dans les spécifications. Puis des variantes de cette animation permettent de vérifier que les permissions accordées interdisent l'exécution des actions sécurisées par des rôles non autorisés.

4.7 Synthèse

Plusieurs travaux de recherche académiques et industriels ont été entrepris avec succès au cours des dernières décennies et ont permis de valider la pertinence de l'utilisation de nombreux formalismes et techniques associées (méthode B, vérification de propriétés, vérification déductive, model checking) pour le domaine ferroviaire et en particulier le système ERTMS. Nous pouvons citer, en guise d'exemple, le projet *OpenETCS* qui contient une contribution intéressante de Systerel [10] s'appuyant sur la maîtrise des méthodes formelles : B [Cavalli *et al.*, 2014] et SCADE [Petit-Doche *et al.*, 2015]. L'objectif de ce projet est de développer une chaîne d'outils holistique de l'ensemble du processus de développement du système ETCS à bord embarqué. En partant des spécifications informelles européennes du SRS Subset 026, le processus *OpenETCS* met en œuvre les principales phases allant de la spécification et la modélisation jusqu'à la validation formelle et la génération de code. En outre, le sous-projet *ERTMSFormalSpecs* [Ferier *et al.*, 2011], un sous-projet d'*OpenETCS*, définit un langage spécifique au domaine ce qu'on appelle DSL (Domain Specific Language), conçu pour exprimer les spécifications d'ERTMS du Subset 026 dans une représentation formelle concise et vérifiable, compréhensible par des spécialistes du domaine. Les travaux cités dans [Ferier *et al.*, 2011] n'incluent pas une phase de preuve des propriétés liées à la sécurité. En revanche, dans nos travaux de recherche, nous nous focalisons sur l'évaluation du système ERTMS en termes d'exigences de sécurité et nous intégrons également les règles opérationnelles nationales propres à chaque pays telles que celles de la ligne LGV-Est Européenne en France dans la phase de modélisation. Ces règles nationales complètent la norme européenne. Ainsi, la finalité de notre étude est aussi d'évaluer la cohérence de la spécification et des règles d'exploitation au regard des exigences de sécurité.

D'autres travaux de couplage de UML et B dans le domaine ferroviaire existent tels que les travaux dans le cadre du projet B-Rail [Boulanger, 2014] pour la modélisation des systèmes ferroviaires complexes. Le cas d'étude choisi est celui du « passage à niveau », un lieu de croisement entre un flux de circulation routière et un flux de circulation ferroviaire. La modélisation UML, dans ces travaux, est très générique et elle ne suit pas un modèle conceptuel permettant de guider le concepteur dans le processus de modélisation. Notre cas d'étude se focalise sur deux scénarios des règles d'exploitation ferroviaires fondés sur des autorisations données aux agents agissant sur les systèmes ferroviaires. Cette étude de cas nous a conduit à utiliser le profil UML existant pour le contrôle d'accès RBAC. Ce profil permet de faciliter la tâche de modélisation en UML et la traduction en B à l'aide de la plate-forme B4MSecure.

10. SYSTEREL : http ://www.systerel.fr/

4.8 Conclusion

Dans ce chapitre, nous avons commencé par une étude préliminaire des règles d'exploitation. L'objectif de cette étude est d'analyser ces règles et de définir les concepts nécessaires pour la modélisation de ces règles. Cette analyse nous a conduit à adapter l'approche de modélisation des politiques de sécurité des systèmes d'information pour la modélisation des règles d'exploitation par analogie des concepts.

Afin de mettre en œuvre notre contribution, nous nous sommes appuyés sur une étude de cas ferroviaire fondée sur deux scénarios extraits des règles d'exploitation ferroviaires ERTMS/ETCS appliquées sur la ligne à grande vitesse LGV Est-Européenne. Ces scénarios contiennent des aspects fonctionnels et des aspects de sécurité. Fondée sur l'ingénierie dirigée par les modèles, la plate-forme *B4MSecure* nous a permis, d'une part, de modéliser ces règles en diagrammes de classes UML renforcés par un profil UML pour la politique de contrôle d'accès RBAC inspirée de SecureUML. D'autre part, ces modèles sont transformés en des spécifications B. Les spécifications B résultantes sont enrichies par des propriétés de sécurité ferroviaire et soumises à des activités de validation et de vérification formelle. Pour cela, nous avons utilisé l'animateur *ProB* pour valider ces scénarios ainsi que les prouveurs automatique et interactive de l'*Atelier B* pour la vérification des propriétés ajoutées.

Chapitre 5

Approche basée sur Organization Based Access Control (Or-Bac) : Extension de RBAC

Sommaire

5.1 Introduction

Les politiques de contrôle d'accès ou d'autorisation permettent de renforcer la confiance en termes de sécurité du système. Le chapitre précédent a montré l'intérêt de l'utilisation de la politique de sécurité RBAC pour la sécurité ferroviaire en modélisant en UML les deux scénarios de notre étude de cas puis en les transformant en des spécifications B à l'aide de l'outil *B4MSecure*. Ces spécifications sont ensuite enrichies par des invariants de sécurité. Dans ce chapitre, nous montrons l'intérêt de l'utilisation de la politique de contrôle d'accès « Organization Based Access Control » (Or-Bac) pour les systèmes ferroviaires en réponse aux limitations de l'utilisation de RBAC. Ensuite, nous proposons une adaptation du modèle Or-Bac à notre contexte en tant qu'extension du modèle RBAC.

5.2 Or-Bac vs RBAC

Comme déjà défini dans le chapitre précédent, le modèle RBAC permet d'associer des privilèges (permissions) aux utilisateurs à travers des rôles. Néanmoins, le modèle RBAC ne permet pas de tenir compte du contexte et de la hiérarchie des organisations détaillés dans la section 5.2.2.

5.2.1 Le modèle Or-Bac

L'entité centrale du modèle Or-Bac est l'*Organisation*. Une organisation peut être considérée comme un groupe organisé de sujets jouant certains rôles. Notons qu'un groupe de sujets ne correspond pas nécessairement à une organisation. Plus précisément, le fait que chaque sujet joue un rôle dans une organisation correspond à un accord entre les sujets pour former une organisation. La notion d'organisation permet d'établir une politique de sécurité plus modulaire. En effet, le modèle Or-Bac définit des règles de sécurité spécifiques à l'organisation et une organisation peut être structurée en plusieurs sous-organisations qui ont chacune leur propre politique de sécurité [Cuppens et Miege, 2003, Cuppens et Miège, 2004].

Le modèle Or-Bac est élaboré sur deux niveaux : un niveau concret et un niveau abstrait. Cette architecture à deux niveaux permet de distinguer la rédaction de la politique de son implantation. Un niveau concret contient les entités *Sujet*, *Action* et *Objet* et un niveau abstrait contient respectivement les méta-entités *Rôle*, *Activité* et *Vue*. Ce modèle s'appuie alors non seulement sur l'ensemble des sujets mais également sur l'ensemble des objets et des actions et place la notion d'*Organisation* comme un concept central.

L'idée de ce modèle est d'accorder des privilèges à des sujets (entités actives) afin qu'ils puissent réaliser des actions sur des objets (entités non actives). La notion de rôle a été déjà introduite dans le modèle RBAC, cependant en Or-Bac, un sujet joue un rôle dans une organisation. Dès lors, les rôles associent des sujets qui remplissent les mêmes

fonctions dans une organisation. De plus, pour les nouvelles entités définies en Or-Bac, une activité est une abstraction d'un ensemble d'actions qui ont un même objectif dans une organisation. De même, une vue est une abstraction d'un ensemble d'objets qui satisfont une propriété commune dans une organisation ce qui se traduit par les relations suivantes :

— Relation sujet-rôle : Employ(Organisation, Sujet, Rôle)

— Relation action-activité : Consider(Organisation, Action, Activité)

— Relation objet-vue : Use(Organisation, Objet, Vue)

La notion de permission a déjà été introduite dans le modèle RBAC. Néanmoins, le modèle Or-Bac n'est pas restreint aux permissions mais permet également de définir des interdictions, des obligations et des recommandations. Une permission concrète, notée **IsPermitted** est dérivée d'une permission abstraite, notée **Permission**. La règle de dérivation est la suivante :

Si **Permission**(Organisation, Rôle, Activité, Vue) et
Employ(Organisation, Sujet, Rôle) et
Consider(Organisation, Action, Activité) et
Use(Organisation, Objet, Vue)
Alors **IsPermitted**(Sujet, Action, Objet)

Les règles de dérivation pour une interdiction, une obligation et une recommandation sont bâties sur le même schéma.

Le modèle Or-Bac, réalisé dans le cadre du projet RNRT MP6 et d'une thèse [Abou EL Kalam, 2003], offre également la possibilité d'exprimer le contexte d'une politique de sécurité. Ceci est défini, selon [Abou EL Kalam, 2003], comme « *toute information qui caractérise la situation d'une entité ou qui spécifie les circonstances concrètes dans lesquelles les organisations accordent des permissions à des rôles pour réaliser des activités sur des vues* ».

Dans les travaux de [Cuppens et Miege, 2003] sur la modélisation des contextes dans le modèle Or-Bac, plusieurs types de contexte ont été introduits tels que :

— *Le contexte temporel* régissant la durée de validité de privilèges,

— *Le contexte spatial* lié à l'appartenance à un réseau, ou la position géographique ou toute autre situation spatiale,

— *Le contexte déclaré par l'utilisateur* activé par un sujet (utilisateur) dans un cas exceptionnel, par exemple une situation d'urgence. L'utilisateur qui déclare le contexte est obligé en contre partie de faire un compte-rendu des opérations effectuées et si nécessaire des raisons qui l'ont motivé à déclarer ce contexte,

— *Le contexte prérequis* correspondant aux contraintes spécifiques au domaine d'application et dépendant de ses caractéristiques. Par exemple, en ferroviaire certaines autorisations changent selon le niveau d'application d'ERTMS/ETCS,

— *Le contexte provisionnel* permettant d'accorder des privilèges en fonction de l'historique des actions réalisées.

Ainsi, la règle de dérivation devient comme suit :

Si **Permission**(Organisation, Rôle, Activité, Vue, Contexte) et
Employ(Organisation, Sujet, Rôle) et
Consider(Organisation, Action, Activité) et
Use(Organisation, Objet, Vue) et
Define(Organisation, Sujet, Action, Objet, Contexte)
Alors **IsPermitted**(Sujet, Action, Objet)

La figure 5.1 montre une vue d'ensemble des concepts du modèle Or-Bac avec une structure en deux couches qui représente les différentes entités (par des rectangles) et les différentes relations (par des ovales).

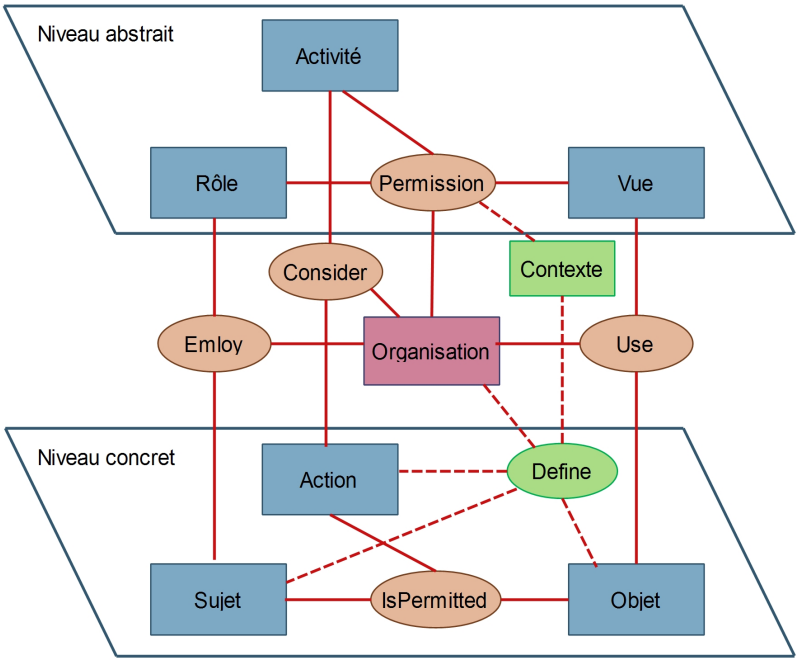

Figure 5.1 – Le modèle Or-Bac de [Cuppens et Miège, 2004]

5.2.2 Les limitations du modèle RBAC

Notre étude basée sur le modèle RBAC montre l'intérêt d'une telle politique de sécurité pour la modélisation des systèmes ferroviaires et la vérification formelle de ses propriétés de sécurité. Sur le plan réglementaire, les documents et les textes de spécifications sont exprimés en langage naturel. Ils ne sont pas assez formalisés. L'utilisation du modèle RBAC permet de modéliser et de formaliser les autorisations sous forme de permissions sur des entités actives (rôles) agissant sur des entités non actives (les entités du modèle fonctionnel). Cette approche favorise la séparation des préoccupations en séparant le modèle fonctionnel du modèle de sécurité.

Néanmoins, la discussion des politiques en vigueur permet de conclure que le modèle RBAC ne couvre pas toute la richesse et la complexité des systèmes ferroviaires européens. Sur le plan technique, la notion de rôle est prometteuse, cependant elle nécessite d'autres notions complémentaires telles que la notion de contexte, la hiérarchie des organisations et la prise en compte de l'interopérabilité.

Le contexte : Il permet de passer d'autorisations statiques à des autorisations dynamiques spécifiques à des circonstances concrètes dans lesquelles les systèmes accordent aux sujets (utilisateurs) des autorisations de réaliser des actions sur des objets. Cette notion existe également en RBAC/SecureUML, sous la forme de contraintes d'autorisation, mais elle ne prend pas en compte la variété de formes que propose Or-Bac à travers les différents types de contexte. En effet, le contexte permet d'ajouter un cadre applicatif des règles en précisant par exemple le lieu, le moment, le type d'intervention, l'état du système, etc. La sous-section 5.3.2 met en exergue l'intérêt de l'utilisation de cette notion dans le domaine ferroviaire.

La hiérarchie des organisations : Elle permet de structurer une organisation en sous-organisations ayant chacune une politique de sécurité. Une organisation, dite mère, peut alors avoir une politique de sécurité générique. Ses sous-organisations peuvent hériter de sa politique de sécurité mais aussi ajouter ou supprimer des autorisations afin de définir leurs propres politiques de sécurité. Cette notion semble être très intéressante car elle contribue clairement à l'interopérabilité. Cet aspect sera pris en compte dans l'extension du modèle RBAC par le modèle Or-Bac.

5.3 L'application d'Or-Bac sur les systèmes ferroviaires

Dans cette section, nous mettons l'accent sur l'intérêt du modèle Or-Bac pour les systèmes ferroviaires. Nous commençons par motiver l'application de ce modèle dans un contexte différent de celui pour lequel il a été conçu. Nous poursuivons par des exemples de contexte et de hiérarchie des organisations dans le contexte ferroviaire. Enfin, nous fournissons un exemple du scénario « Override EOA » défini avec le modèle Or-Bac.

5.3.1 Motivation

Les trois principales raisons de l'utilisation de ce modèle dans notre étude sur les systèmes ferroviaires sont les suivantes :

— *La spécification des autorisations contextuelles* qui dépendent des circonstances dans lesquelles ces autorisations sont accordées.

— *La définition d'une organisation et la hiérarchie de sous-organisations* qui ont chacune leurs règles de sécurité. Il est possible de spécifier des règles de sécurité

génériques au niveau d'une organisation mère ; ses sous-organisations peuvent alors hériter de ces règles et même ajouter ou supprimer des règles et ainsi définir leurs propres règles.

— *La spécification des différents types d'autorisation* à savoir les permissions, les interdictions, les obligations et les recommandations qui peuvent enrichir le modèle avec les différentes considérations qui en découlent.

Dans la section suivante, nous illustrons les deux premiers points ci-dessus par des exemples réels dans le domaine ferroviaire.

5.3.2 Le contexte

Le modèle Or-Bac offre la possibilité d'exprimer le contexte d'une règle de sécurité ferroviaire qui peut varier en fonction du temps, en fonction d'une localisation, en fonction de l'activation d'une action par un utilisateur ou en fonction de l'historique des actions. Un utilisateur peut aussi activer des permissions, interdictions, etc. selon son objectif et le travail qu'il est en train d'effectuer.

Dans le domaine ferroviaire, le contexte peut être utilisé également pour exprimer des notions sur lesquelles se basent les spécifications d'ERTMS/ETCS telles que les différents niveaux d'ERTMS, les situations dégradées, les conditions de passage d'une procédure à une autre, etc.

En se basant sur la procédure « Override EOA » décrite dans le chapitre précédent, nous pouvons citer quelques types de contexte avec des exemples concrets :

— *Le contexte prérequis* : les différents niveaux d'ERTMS/ETCS définissent les différentes utilisations d'ERTMS. En effet, au niveau 0, le franchissement d'un signal de danger, équivalent fonctionnel à un « Override EOA », est une procédure nationale. L'équipement ERTMS/ETCS embarqué n'est pas impliqué dans cette procédure, car il ne supervise pas les mouvements de trains et ainsi il n'y a pas de règles européennes à appliquer vis à vis de cette procédure. Cependant, dans les zones équipées d'ERTMS (niveau 1, 2 ou 3), l'équipement ERTMS/ETCS embarqué supervise les positions où le train doit s'arrêter. Le conducteur doit par la suite être en mesure d'inhiber cette supervision après avoir reçu un ordre de l'agent de circulation de franchir l'EOA. Par conséquent, la procédure « Override EOA » change selon le niveau d'ERTMS/ETCS qui est un prérequis pour les systèmes ferroviaires.

— *Le contexte déclaré par l'utilisateur* : la procédure « Override EOA » est déclenchée par le conducteur en cas de situation dégradée. Les situations dégradées sont des situations résultant d'un événement entravant le déroulement des scénarios nominaux ; par exemple la perte de la communication radio entre le système à bord et le système au sol. Suite à ces situations, le conducteur peut contacter l'agent de

circulation par téléphone et demander l'autorisation de franchissement afin qu'il puisse déclencher la procédure.

— *Le contexte temporel* : ce type de contexte peut concerner toute information liée à une contrainte de temps.

— *Le contexte spatial* : de même, ce contexte peut concerner toute information liée à une contrainte spatiale de distance. Nous citons un exemple à la fois pour le contexte temporel et pour le contexte spatial : une MA est obtenue pour une distance et pour une plage de temps données. En effet, au bout d'un certain temps si le train n'a pas parcouru une distance donnée, la MA sera perdue.

— *Le contexte provisionnel* : le conducteur ne doit déclencher la procédure « Override EOA » qu'après en avoir reçu l'autorisation de la part de l'agent de circulation. Par conséquent la permission de franchir un EOA est accordée seulement si l'agent de circulation a déjà autorisé le conducteur à le faire (à travers un ordre écrit ETCS01 pour l'organisation LGV-Est).

La structuration des types de contextes telle quelle est définie dans le modèle Or-Bac répond à notre logique métier. Néanmoins, dans le contexte ferroviaire, les contraintes liées au mouvement du train ne peuvent pas être classées dans l'un de ces types. Par exemple, des contraintes sur des vitesses se définissent comme étant des contraintes spatio-temporelles qui décrivent un mouvement du train. Nous définissons pour cela un contexte relatif au mouvement dynamique du train.

5.3.3 La hiérarchie des organisations

La hiérarchie des organisations permet de modéliser la structure des organisations réelles. Dans notre étude, ERTMS/ETCS peut être considéré comme une organisation mère. Toute ligne spécifique à un pays équipée d'ERTMS/ETCS constitue une sous-organisation qui peut hériter des règles de sécurité génériques de l'organisation mère, et également ajouter ou supprimer des règles et ainsi définir ses propres règles d'exploitation ferroviaires. En France par exemple, le document [RFF, 2012] sur la ligne à grande vitesse LGV-Est Européenne reprend les principes et les dispositions relatifs à l'utilisation du système ERTMS/ETCS en apportant des précisions utiles pour leur mise en œuvre. La définition d'une telle hiérarchie peut alors contribuer à mettre en avant l'interopérabilité des différents systèmes ERTMS/ETCS en Europe.

En ERTMS, le mode « Shunting », un mode de fonctionnement spécifique aux manœuvres, peut être sélectionné par le conducteur, seulement dans le cas où le train est à l'arrêt, ou s'il s'agit d'un ordre du système au sol. Toutefois, en France sur la ligne LGV-Est, ce mode n'est pas mis en œuvre. L'apparition du symbole indiquant le mode « Shunting » est à considérer comme une anomalie. Le mode « Reversing » permet au conducteur de changer la direction du mouvement du train tout en restant dans la même cabine, et ainsi

l'orientation du train reste inchangée. Ce mode n'est également pas mis en œuvre sur la ligne LGV-Est. On peut exprimer ces restrictions par l'inclusion d'une interdiction dans la politique de contrôle d'accès de la ligne LGV-Est.

5.3.4 Scénario « Override EOA » en Or-Bac

Le tableau ci-dessous décrit un aperçu du scénario « Override EOA » en spécifiant le comportement du système et en attribuant l'autorisation correspondante sous la forme **Permission**(Organisation, Rôle, Activité, Vue, **Contexte**).

Comportement Système (langage naturel)	**Documents**	**Autorisations**
Le conducteur demande la permission d'utiliser la procédure Override EOA.	SRS 5.8.1.4.1	**Permission**(ERTMS/ETCS, Driver, Requesting, Override_Request, **C1**)
L'agent de circulation accorde l'autorisation au conducteur de franchir l'EOA.	SRS 5.8.1.4.1	**Permission**(ERTMS/ETCS, TrafficAgent, Authorization, Override_Order, **C2**)
Pour la LGV Est, à l'aide d'un ordre écrit ETCS01.	RFN [RFF, 2012]	**Permission**(ERTMS_LGVEst, TrafficAgent, Authorization, ETCSOrder_ETCS01, **C2**)
Le conducteur appuie sur le bouton Override EOA.	SRS 5.8.2.3	**Permission**(ERTMS/ETCS, Driver, Acquittal, Button_OverrideEOA, **C3**)

Tableau 5.1 – La procédure « Override EOA » en ERTMS niveau 2

C1 est le contexte qui exprime que le bouton « Override EOA » est disponible parce que la vitesse du train actuelle est inférieure à la vitesse maximale autorisée pour franchir l'EOA (valeur nationale). C'est un contexte dynamique de mouvement du train.

C2 exprime que toutes les conditions d'établissement de l'itinéraire sont remplies conformément aux règles nationales. Nous ne parvenons pas à assigner un type à ce contexte car il est générique. Selon les documents à notre disposition, nous n'avons pas les détails permettant d'exprimer ces conditions.

C3 signifie que cette permission est accordée seulement si la permission de franchir l'EOA est déjà accordée par l'agent de circulation. C'est un contexte provisionnel qui s'appuie sur l'historique des actions précédemment réalisées.

Pour illustrer les notions de contexte et d'organisation du modèle Or-Bac vis-à-vis du modèle RBAC, nous abordons dans la section suivante notre vision en vue d'une interopérabilité ferroviaire telle quelle est définie dans la section 1.2.1 de l'introduction générale.

5.4 Notre vision en vue de l'interopérabilité

Les deux notions de contexte et de hiérarchie des organisations permettent la mise en avant de l'interopérabilité en tenant compte des règles nationales qui régissent les lignes nationales exploitant ERTMS. En effet, la notion de hiérarchie des organisations permet de spécifier les spécifications européennes ERTMS comme une organisation mère et les spécifications propres à chaque pays, au niveau national, y intégrant les règles nationales comme organisations filles. Ces organisations filles héritent des spécifications européennes en y ajoutant et/ou supprimant des règles qui leur sont propres. La notion de contexte, quant à elle, définit les circonstances concrètes correspondant à l'application d'une règle.

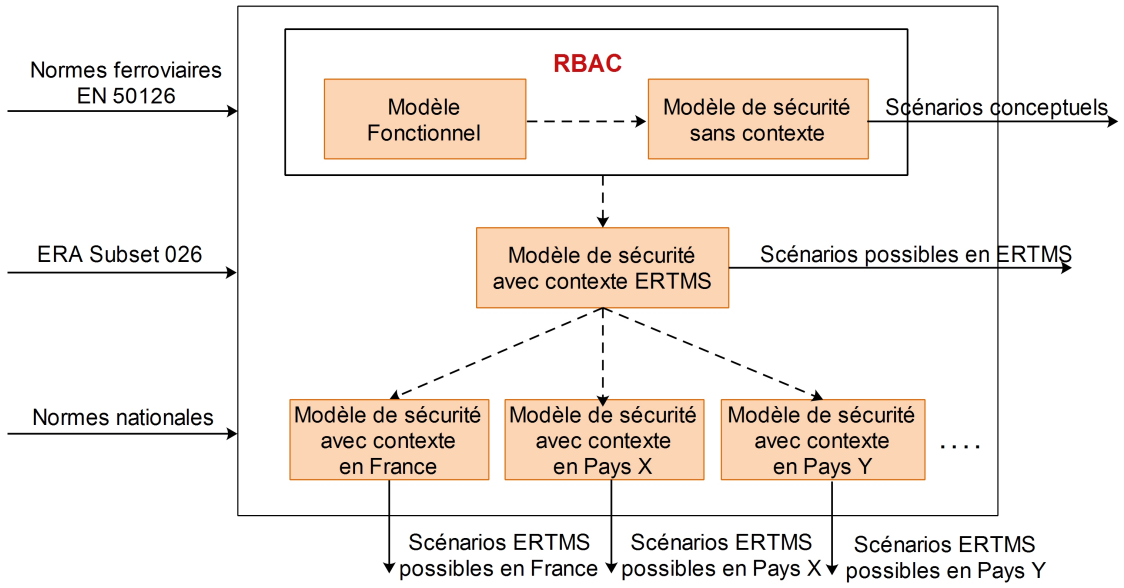

Figure 5.2 – Vision sur l'interopérabilité des systèmes ferroviaires

La figure 5.2 décrit comme point d'entrée les documents de spécifications européennes ERTMS et nationales pour les pays suivants : France, Pays X, Pays Y, etc. La modélisation avec le modèle RBAC permet d'avoir un modèle fonctionnel et un modèle de sécurité augmentés par des exigences de sécurité. De cette modélisation résultent les scénarios envisageables issus des documents de spécifications qui ne tiennent pas compte des organisations. Le modèle de sécurité résultant de RBAC contient toutes les permissions possibles associées aux rôles et ainsi aux utilisateurs jouant ces rôles. Notre objectif est de construire un modèle de sécurité pour l'organisation mère ERTMS avec son contexte approprié. Ce modèle de sécurité contient un sous ensemble d'autorisations de l'ensemble des autorisations résultantes de RBAC propres à l'organisation mère ERTMS. De même pour les organisations filles, chaque modèle de sécurité contient les autorisations héritées de l'organisation mère ERTMS en supprimant éventuellement quelques unes et/ou ajoutant des autorisations de l'ensemble des autorisations résultantes de RBAC. De ces modèles de sécurité, nous pouvons valider les scénarios possibles en ERTMS avec le contexte

d'ERTMS qui représentent le noyau des règles d'exploitation européennes, les scénarios possibles en France (la ligne LGV-Est par exemple), les scénarios possibles dans un Pays X, les scénarios possibles dans un Pays Y, etc. Ces scénarios à valider correspondent aux différentes implémentations nationales d'ERTMS en Europe.

5.5 Modélisation UML

5.5.1 Méta-modèle Or-Bac étendu

Dans le cadre des travaux implantés dans la plate-forme B4MSecure, le méta-modèle a été enrichi par la notion d'organisation. À cet égard, le travail de [Mokni, 2012] a proposé une formalisation en B des politiques de sécurité en tenant compte de la notion d'organisation. Ce travail a fait la continuité des travaux de formalisation en B des politiques de contrôle d'accès RBAC de [Idani *et al.*, 2010]. Néanmoins, la transformation UML/B, suite à l'intégration de la notion d'organisation, n'est pas encore implémentée.

Dans cette section, nous proposons une extension du méta-modèle Or-Bac à un niveau conceptuel indépendant des plate-formes (PIM). Cette extension couvre, en plus du concept d'organisation, la variété des types de contextes d'Or-Bac. Les différents types de contextes définis dans la section 5.3.2 permettent de structurer les contraintes en plusieurs variantes temporelle, spatiale, prérequis, provisionnelle, déclarée par l'utilisateur et dynamique. La structuration des contextes du modèle Or-Bac répond à notre logique métier. Chacune de ces variantes peut décrire, sous forme de contraintes OCL, des circonstances du système qui permet d'analyser les scénarios des règles d'exploitation, comme par exemples :

- Dans le contexte déclaré par l'utilisateur, une information supplémentaire sur l'identité de l'utilisateur permet de détecter le responsable d'une éventuelle défaillance du système pouvant découler de l'activation de cette action.
- Dans le contexte provisionnel, une information supplémentaire sur l'historique des actions précédentes effectuées permet de détecter une éventuelle défaillance du système pouvant découler du non respect de la séquence d'actions décrites par les règles d'exploitation.

Dans ce méta-modèle de la figure 5.3, les associations entre organisation, rôle, permission, utilisateur, session et action sont définies de la même manière que dans le méta-modèle de [Mokni, 2012] :

- Une permission (méta-classe *Permission*) est associée à un rôle (méta-classe *Role*) et une ou plusieurs organisations (méta-classe *Organization*).
- Dans une organisation, plusieurs rôles peuvent être joués.
- Un utilisateur (méta-classe *User*) est affecté à un rôle dans une organisation par l'intermédiaire d'une méta-classe *Assignment*.
- La hiérarchie de rôles par l'association réflexive sur la méta-classe *Role* permet de

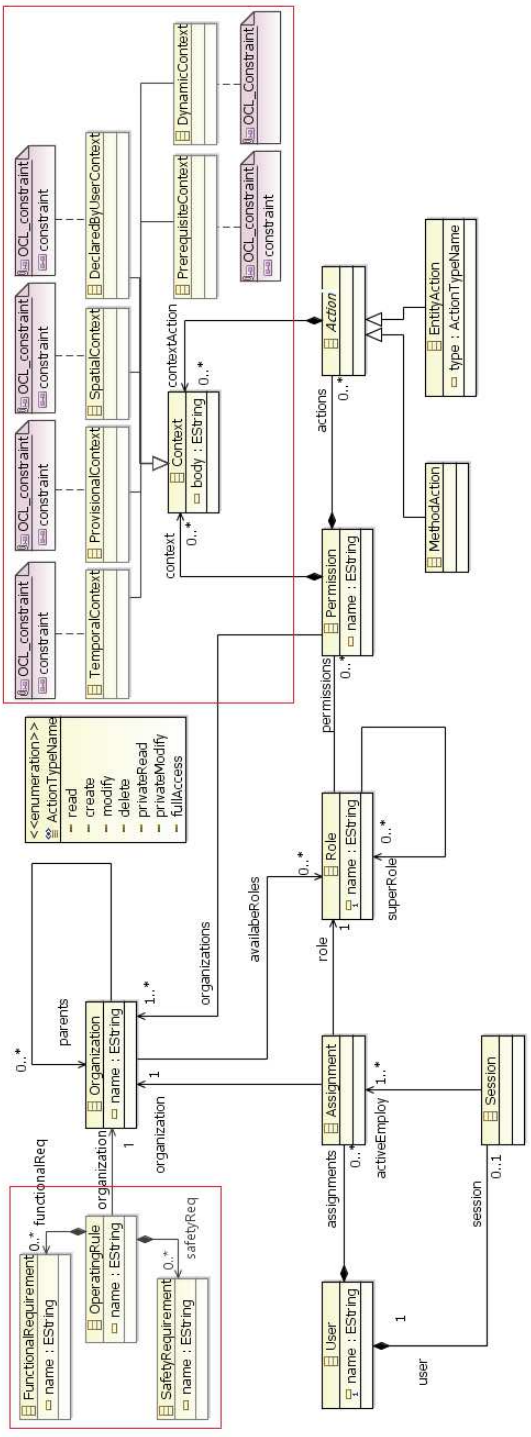

Figure 5.3 – Le méta-modèle Or-Bac étendu

factoriser certaines permissions. En effet, le mécanisme d'héritage des rôles permet d'accorder toutes les permissions d'un rôle à ses sous-rôles.

— La hiérarchie des organisations par l'association réflexive sur la méta-classe *Organization* permet d'accorder des permissions d'une organisation mère à toutes ses sous-organisations filles.

Comme extension de ce méta-modèle, une permission est composée de contextes permettant de délivrer cette permission. Certains contextes sont spécifiques à une action parmi l'ensemble des actions d'une permission. Les différents types de contextes sont représentés par des relations d'héritage et peuvent être exprimés en contraintes OCL. C'est la partie encadrée à droite de la figure 5.3.

Dans une optique de traçabilité des exigences, nous avons ajouté dans ce méta-modèle une méta-classe qui représente une règle d'exploitation (méta-classe *OperatingRule*). Une règle d'exploitation est définie dans une organisation. Elle peut être liée à plusieurs exigences fonctionnelles (méta-classe *FunctionalRequirement*) et exigences de sécurité (méta-classe *SafetyRequirement*). C'est la partie encadrée à gauche de la figure 5.3. L'étude de cet aspect de traçabilité figure parmi les perspectives de ces travaux.

5.5.2 Profil UML pour Or-Bac

Nous avons conçu un profil UML pour Or-Bac (figure 5.4) comme une spécialisation du méta-modèle ci-dessus. Ce profil contient des stéréotypes, des valeurs marquées (tagged values) et des contraintes OCL.

En effet, nous trouvons :

— les stéréotypes *Role* et *Organization*, extensions de la méta-classe *Class* du méta-modèle UML,

— le stéréotype *Action*, extension de la méta-classe *Operation* du méta-modèle UML,

— les stéréotypes *EntityAction* et *MethodAction*, spécialisations du stéréotype *Action*,

— le stéréotype *User*, extension de la méta-classe *InstanceSpecification* du méta-modèle UML,

— le stéréotype *Permission* , extension de la méta-classe *AssociationClass* du méta-modèle UML, les *tagged values organisation*, *context* et *actions* du stéréotype *Permission*,

— le stéréotype *Context*, extension de la méta-classe *Constraint* du méta-modèle UML,

— les stéréotypes *TemporalContext*, *SpatialContext*, *PrerequisiteContext*, *DeclaredByUserContext*, *ProvisionalContext* et *DynamicContext*, spécialisations du stéréotype *Context*,

— des contraintes OCL pour guider le concepteur dans sa modélisation UML en respectant le profil ainsi que le méta-modèle ; par exemple le stéréotype *Permission* est une classe associative entre une classe (du modèle fonctionnel) et une classe stéréotypée *Role*.

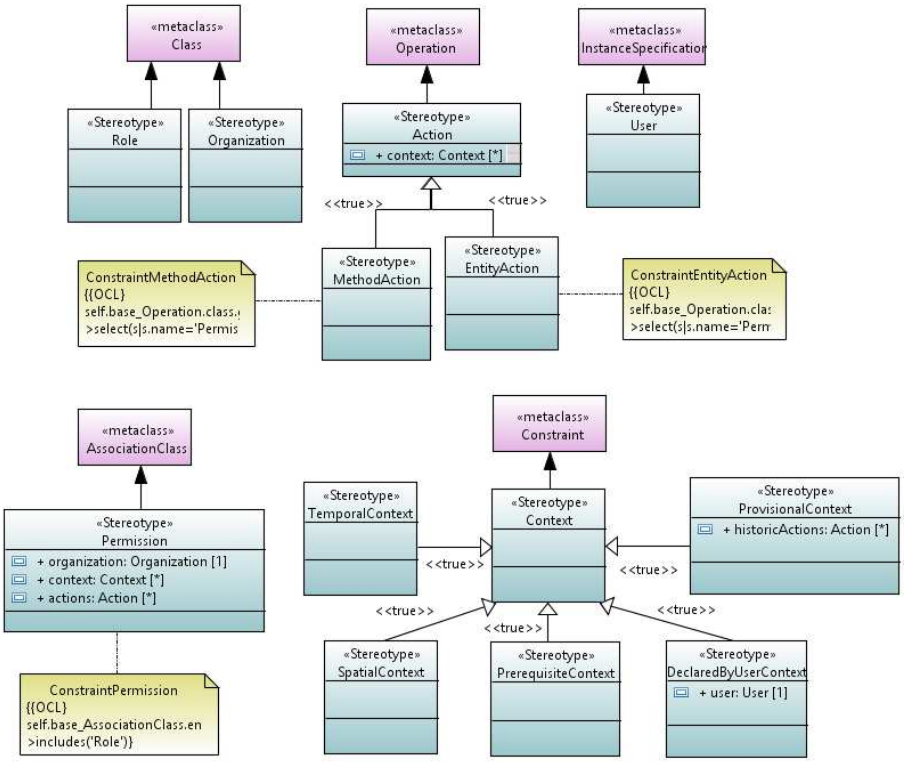

Figure 5.4 – Le profil Or-Bac

5.5.3 Classification des organisations

Notre objectif est d'analyser des scénarios issus de l'application des règles d'exploitation ferroviaires d'ERTMS niveau 2 sur des lignes nationales. Par conséquent, nous proposons une organisation mère pour ERTMS/ETCS contenant les autorisations possibles au niveau d'ERTMS et des organisations filles pour les lignes nationales particulières qui héritent des autorisations d'ERTMS, ajoutent et suppriment des autorisations selon les règles nationales en vigueur. Ainsi, l'ajout des autorisations se traduit par l'ajout d'autres permissions qui n'existent pas dans l'organisation mère, alors que la suppression des autorisations se traduit par l'interdiction de quelques permissions de l'organisation mère. Toutefois, dans le modèle RBAC, la permission est le seul type d'autorisation. Une interdiction sur une action peut être alors considérée comme étant l'absence de permission sur cette même action. Par analogie, nous considérons, comme dans le modèle RBAC, qu'une action n'est opérationnelle pour un rôle que si ce rôle a la permission de l'exécuter. En d'autre termes, une action interdite pour un rôle est une action non permise, et donc si un rôle ne possède pas une permission sur une action, cette dernière est considérée interdite.

Afin de pouvoir raisonner sur la permission, nous devons formaliser un ensemble de

permissions dans l'organisation mère. Les organisations filles héritent de toutes les permissions de l'organisation mère, peuvent ajouter d'autres permissions, mais ne peuvent pas en supprimer. Dans le cas d'une organisation mère ERTMS, nous centralisons l'ensemble des permissions ERTMS communes à toutes les lignes particulières dans un « noyau » ERTMS, appelé « Kernel_ERTMS ». Chacune des organisations filles hérite alors des permissions de ce noyau et y ajoutent ses propres permissions. Le choix d'un noyau pour les permissions communes est aussi motivé par le fait que la plupart des permissions sont maintenues par les organisations filles et que très peu de permissions ne sont pas supportées par ces organisations.

La figure 5.5 montre les organisations mère et filles qui correspondent à la figure 5.2. Afin de valider des scénarios possibles en ERTMS, nous considérons une organisation fille ERTMS qui contient toutes les permissions en ERTMS, ce qui correspond à un « ERTMS complet ». De même, les organisations filles « LGV_Est », « Ligne_PaysX » et « Ligne_PaysY » permettent de valider des scénarios dans les lignes nationales particulières LGV_Est en France, les lignes du Pays X et les lignes du Pays Y.

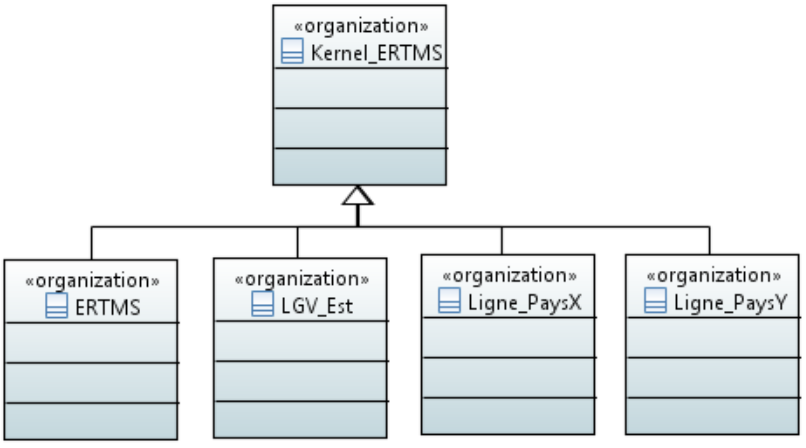

Figure 5.5 – Héritage des organisations

L'objectif majeur de l'application du modèle Or-Bac pour les systèmes ferroviaires est de modéliser nos scénarios de manière à formaliser le problème de sécurité. Cette formalisation se matérialise par le fait qu'aucune séquence valide d'actions ne puisse amener le système dans un état permettant la violation d'une propriété de sécurité. Ceci permet l'utilisation d'une méthode formelle pour valider les spécifications sur l'ensemble des autorisations accordées au regard des exigences de sécurité.

5.6 Formalisations en B

Comme nous l'avons déjà introduit dans la section 5.5.1, les travaux de [Mokni, 2012] étendent le modèle RBAC par la notion d'organisation, concept central d'Or-Bac. Cette extension engendre une modification directe dans les machines B en intégrant la notion d'organisation. Par exemple, la relation d'affectation des utilisateurs aux rôles est remplacée par la relation d'affectation des utilisateurs aux rôles dans des organisations. Dans nos travaux, nous proposons d'ajouter des nouvelles machines centrées sur les organisations. Ceci nous permet de vérifier et de valider des scénarios qui ne tiennent pas compte des organisations tels que définis dans le chapitre 4. De plus, l'ajout des nouvelles machines permet de tirer profit du concept d'organisation dans l'optique d'une interopérabilité ferroviaire.

Dans cette section, nous détaillons la formalisation en B de la notion d'organisation après avoir présenté une stratégie d'architecture pour une implémentation des différentes organisations.

5.6.1 Architecture en B

Nous avons conçu une architecture B (voir figure 5.6) telle que chaque organisation possède un modèle fonctionnel et un modèle de sécurité. Dans cette architecture, nous considérons un seul niveau d'héritage : une organisation mère et des organisations filles qui héritent de l'organisation mère.

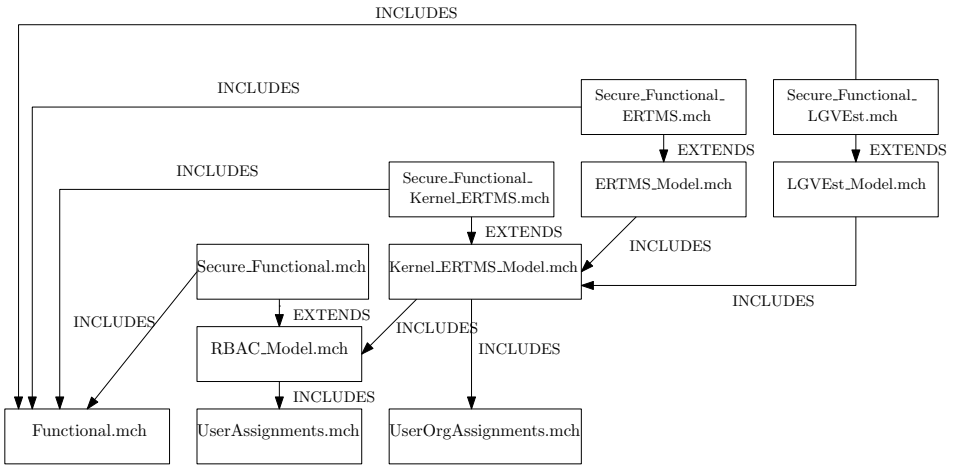

Figure 5.6 – L'architecture des composants B

Une partie de cette architecture concerne le modèle RBAC du chapitre 4. Les machines relatives à cette partie sont :

— *Functional.mch* qui représente le modèle fonctionnel.

— *UserAssignments.mch* qui permet d'associer les utilisateurs aux rôles.

— *RBAC_Model.mch* qui définit toutes les variables, les définitions et les opérations pour attribuer les permissions aux rôles. Le calcul des permissions est effectué à l'aide d'une relation *isPermitted* entre les rôles et les opérations. Cette machine contient également des opérations de connexion et de déconnexion des utilisateurs du système en fonction des rôles qu'ils jouent.
— *Secure_Functional.mch* qui est un filtre de sécurité sur les opérations fonctionnelles. Elle étend la machine *RBAC_Model.mch* pour pouvoir utiliser les opérations de connexion et de déconnexion.

Dans le chapitre précédent, la machine *Secure_Functional.mch* fait partie de la machine *RBAC_Model.mch*. Nous avons procédé ici à un découpage dans le but de pouvoir calculer les permissions dans les autres machines d'organisations sans utiliser le filtre de sécurité sur les opérations fonctionnelles. En effet, la machine *RBAC_Model.mch* contient les permissions accordées aux utilisateurs en fonction des rôles qu'ils jouent sans tenir compte des organisations. Selon l'organisation, les permissions dans les machines d'organisations seront alors calculées en tant que sous-ensemble des permissions de la machine *RBAC_Model.mch*. De la machine *RBAC_Model.mch* résultent les scénarios envisageables qui ne tiennent pas compte des organisations.

Les autres machines de cette architecture concernent les machines propres à chacune des organisations :

— *UserOrgAssignmets.mch* qui permet d'associer des utilisateurs aux rôles dans des organisations.
— *Kernel_ERTMS_Model.mch* et *Secure_Functional_Kernel_ERTMS.mch* qui sont des machines de l'organisation mère *Kernel_ERTMS_org*. La machine *Kernel_ERTMS_Model.mch* permet alors le calcul des permissions dans cette organisation, alors que la machine *Secure_Functional_Kernel_ERTMS.mch* constitue le filtre de sécurité pour les opérations du modèle fonctionnel. De la machine *Secure_Functional_Kernel_ERTMS.mch* résultent les scénarios possibles dans cette organisation *Kernel_ERTMS_org*.
— *ERTMS_Model.mch* et *Secure_Functional_ERTMS.mch* qui sont des machines de l'organisation fille *ERTMS_org*. La machine *ERTMS_Model.mch* permet alors le calcul des permissions dans cette organisation, alors que la machine *Secure_Functional_ERTMS.mch* constitue le filtre de sécurité pour les opérations du modèle fonctionnel. De la machine *Secure_Functional_ERTMS.mch* résultent les scénarios possibles dans cette organisation *ERTMS_org*.
— *LGVEst_Model.mch* et *Secure_Functional_LGVEst.mch* qui sont des machines de l'organisation fille *LGVEst_org*. La machine *LGVEst_Model.mch* permet alors le calcul des permissions dans cette organisation, alors que la machine *Secure_Functional_LGVEst.mch* constitue le filtre de sécurité pour les opérations du modèle fonctionnel. De la machine *Secure_Functional_LGVEst.mch* résultent les scénarios possibles

dans cette organisation *LGVEst_org*.

— De même pour l'organisation Pays X (respectivement Pays Y) qui contient deux machines : *Ligne_PaysX_Model.mch* et *Secure_Functional_Ligne_PaysX.mch* (respectivement *Ligne_PaysY_Model.mch* et *Secure_Functional_Ligne_PaysY.mch*).

5.6.2 Calcul des permissions des différentes organisations

Étant donné que le noyau d'ERTMS contient les permissions ERTMS communes à toutes les lignes nationales particulières, les permissions d'ERTMS complet sont l'union des permissions du noyau ERTMS et de ses propres permissions. De même pour les lignes nationales, les permissions de l'organisation LGV-Est sont l'union des permissions du noyau ERTMS et des permissions propres à la ligne LGV-Est. Nous procédons de la même manière pour les autres organisations.

D'un point de vue mathématique, nous présentons les permissions de chacune des organisations comme suit :

Définition 6 *On suppose que* $\boldsymbol{P_{om}}$ *représente l'ensemble des permissions d'une organisation* $\boldsymbol{om}$*, alors l'héritage de cette organisation mère* $\boldsymbol{om}$ *par une organisation fille* $\boldsymbol{of}$ *engendre* $\boldsymbol{P_{of} = P_{om} \cup P_{pf}}$ *avec* $\boldsymbol{P_{pf}}$ *l'ensemble des permissions propres à l'organisation fille tel que* $\boldsymbol{P_{om} \cap P_{pf} = \emptyset}$.

Dans ce qui suit, nous présentons des extraits des modèles de sécurité de RBAC et des organisations permettant le calcul des permissions.

La figure 5.7 est un extrait du modèle de sécurité RBAC qui ne tient pas compte des organisations. Ce modèle spécifie toutes les opérations permises accordées aux rôles sous forme de couples (rôle, opération). Cette extraction se fait à partir des droits de création par les constructeurs, des droits de destruction par les destructeurs, des droits de lecture par les accesseurs (getters) et des droits de modification par les mutateurs (setters). La définition *permissions* représente l'union de tous ces couples (rôle, opération). Cette définition n'est pas spécifiée dans la figure 5.7 mais elle est utilisée par la définition *allPermissions*.

Dans la figure 5.7, la variable *isPermitted* représente les relations entre l'ensemble des rôles *ROLES* et l'ensemble des opérations *Operations*. Par ailleurs, la définition *allPermissions* permet la déduction des couples *(ro,op)* à partir de *permissions* en tenant compte de la hiérarchie des rôles. En effet, *Roles_Hierarchy* traduit l'héritage entre les rôles et se définit en tant que relation entre les rôles filles et les rôles mères : *Roles_Hierarchy* : $ROLES \leftrightarrow ROLES$.

L'opération *setPermissions* permet d'affecter toutes les permissions sous forme de couple (rôle, opération) de la définition *allPermissions* à la variable *isPermitted*.

```
VARIABLES
   isPermitted
INVARIANT
   isPermitted : ROLES ↔ Operations
DEFINITIONS
   allPermissions == {ro, op | ro ∈ ROLES ∧ op ∈ Operations ∧ op ∈
ran(permissions) ∧ ro ∈ (dom(permissions ▷ {op}) ∨
closure1(Roles_Hierarchy∼)[(dom(permissions ▷ {op}))])}
 OPERATIONS
   setPermissions =
        PRE isPermitted = ∅
        THEN isPermitted := allPermissions
   END;
```

Figure 5.7 – Extrait du modèle de sécurité RBAC « RBAC_Model.mch »

```
VARIABLES
   isPermitted_in_Kernel_ERTMS
INVARIANT
   isPermitted_in_Kernel_ERTMS : ROLES ↔ Operations ∧
   ∀(ro, op).(ro ∈ ROLES ∧ op ∈ Operations ∧ (ro,op) ∈
isPermitted_in_Kernel_ERTMS ⇒ ((ro,op) ∈ isPermitted ∧ ro ∈ ran(Kernel_ERTMS_org
◁ allOrgRoles)))
DEFINITIONS
   allPermissions_in_Kernel_ERTMS == {ro, op | ro ∈ ROLES ∧ op ∈ Operations ∧ op
∈ ran(permissions_in_Kernel_ERTMS) ∧ ro ∈ (dom(permissions_in_Kernel_ERTMS ▷
{op}) ∨ closure1(Roles_Hierarchy∼)[(dom(permissions_in_Kernel_ERTMS ▷ {op}))])}
 OPERATIONS
   setPermissions_in_Kernel_ERTMS =
        PRE isPermitted = ∅ ∧ isPermitted_in_Kernel_ERTMS = ∅
        THEN
             setPermissions
             || isPermitted_in_Kernel_ERTMS := allPermissions_in_Kernel_ERTMS
   END;
```

Figure 5.8 – Extrait du modèle de sécurité du noyau ERTMS « Kernel_ERTMS_Model.mch »

Dans la figure 5.8, la variable *isPermitted_in_Kernel_ERTMS* et la définition *permissions_in_Kernel_ERTMS* sont définies de la même manière que *isPermitted* et *permissions* de la figure 5.7.

Dans le modèle de sécurité du noyau ERTMS, un prédicat universellement quantifié dans l'invariant permet de spécifier que pour tout couple *(ro, op)* tel que *ro* appartient à l'ensemble *ROLES*, *op* appartient à l'ensemble de *Operations* et le couple *(ro, op)* ap-

partient à *isPermitted_in_Kernel_ERTMS* implique que le couple *(ro, op)* appartient à *isPermitted* tel que le rôle *ro* appartient à l'ensemble des rôles de l'organisation *Kernel_ERTMS_org*. Dans cet invariant, *allOrgRoles* est une définition permettant de retrouver les rôles associés à toutes les organisations du système. En d'autres termes, cet invariant permet de vérifier si tous les couples (ro, op) dans l'organisation *Kernel_ERTMS_org* forment un sous ensemble des couples de *isPermitted* tel que *ro* est un rôle de cette organisation.

La définition *allPermissions_in_Kernel_ERTMS* permet la déduction des couples *(ro,op)* à partir de *permissions_in_Kernel_ERTMS* en tenant compte de la hiérarchie des rôles.

L'opération *setPermissions_in_Kernel_ERTMS* permet d'affecter toutes les permissions sous forme de couple (rôle, opération) de la définition *allPermissions_in_Kernel_ERTMS* à la variable *isPermitted_in_Kernel_ERTMS*. Étant donné que la variable *isPermitted_in_Kernel_ERTMS* dépend de la variable *isPermitted*, cette opération permet d'appeler en parallèle l'opération *setPermissions* du modèle de sécurité de RBAC pour affecter la variable *isPermitted*. Cette opération respecte l'invariant défini ci-dessus.

```
VARIABLES
   isPermitted_in_ERTMS
INVARIANT
   isPermitted_in_ERTMS : ROLES ↔ Operations ∧
   ∀(ro, op).(ro ∈ ROLES ∧ op ∈ Operations ∧ (ro,op) ∈ isPermitted_in_ERTMS ⇒
((ro,op) ∈ isPermitted ∧ ro ∈ ran(ERTMS_org ◁ allOrgRoles)))
DEFINITIONS
   allPermissions_in_ERTMS == {ro, op | ro ∈ ROLES ∧ op ∈ Operations ∧ op ∈
ran(permissions_in_ERTMS) ∧ ro ∈ (dom(permissions_in_ERTMS ▷ {op}) ∨
closure1(Roles_Hierarchy∼)[(dom(permissions_in_ERTMS ▷ {op}))])}
 OPERATIONS
   setPermissions_in_ERTMS =
       PRE isPermitted_in_ERTMS = ∅ ∧ isPermitted_in_Kernel_ERTMS = ∅
       THEN
            setPermissions_in_Kernel_ERTMS
            || isPermitted_in_ERTMS := allPermissions_in_ERTMS
   END;
   setAllPermissions_in_ERTMS =
       PRE isPermitted_in_ERTMS = allPermissions_in_ERTMS
       THEN
       isPermitted_in_ERTMS := isPermitted_in_ERTMS ∪ isPermitted_in_Kernel_ERTMS
   END;
```

Figure 5.9 – Extrait du modèle de sécurité d'ERTMS « ERTMS_Model.mch »

De même pour la figure 5.9, un invariant de quantification universelle permet de vérifier si tous les couples (ro, op) dans l'organisation *ERTMS_org* forment un sous ensemble des couples de *isPermitted* tel que *ro* est un rôle de cette organisation.

La définition *allPermissions_in_ERTMS* permet la déduction des couples *(ro,op)* à partir de *permissions_in_ERTMS* en tenant compte de la hiérarchie des rôles.

L'opération *setPermissions_in_ERTMS* permet d'affecter toutes les permissions sous forme de couple (rôle, opération) de la définition *allPermissions_in_ERTMS* à la variable *isPermitted_in_ERTMS*. Cette opération permet d'appeler en parallèle l'opération *setPermissions_in_Kernel_ERTMS* du modèle de sécurité du noyau ERTMS (figure 5.8) pour affecter la variable *isPermitted_in_Kernel_ERTMS* qui a son tour fait un appel à l'opération *setPermissions* pour affecter la variable *isPermitted*. Cette opération permet de calculer les permissions propres à l'organisation *ERTMS_org*. Une deuxième opération *setAllPermissions_in_ERTMS* permet d'ajouter les permissions de l'organisation mère *Kernel_ERTMS_org* à l'ensemble des permissions propres. Par conséquent, *isPermitted_in_ERTMS* est l'union de *isPermitted_in_ERTMS* (contenant les permissions propres *allPermissions_in_ERTMS*) et de *isPermitted_in_Kernel_ERTMS*.

```
VARIABLES
   isPermitted_in_LGVEst
INVARIANT
   isPermitted_in_LGVEst : ROLES ↔ Operations ∧
   ∀(ro, op).(ro ∈ ROLES ∧ op ∈ Operations ∧ (ro,op) ∈ isPermitted_in_LGVEst ⇒
((ro,op) ∈ isPermitted ∧ ro ∈ ran(LGVEst_org ◁ allOrgRoles)))
DEFINITIONS
   allPermissions_in_LGVEst == {ro, op | ro ∈ ROLES ∧ op ∈ Operations ∧ op ∈
ran(permissions_in_LGVEst) ∧ ro ∈ (dom(permissions_in_LGVEst ▷ {op}) ∨
closure1(Roles_Hierarchy∼)[(dom(permissions_in_LGVEst ▷ {op}))])}
 OPERATIONS
   setPermissions_in_LGVEst =
        PRE isPermitted_in_LGVEst = ∅ ∧ isPermitted_in_Kernel_ERTMS = ∅
        THEN
             setPermissions_in_Kernel_ERTMS
             || isPermitted_in_LGVEst := allPermissions_in_LGVEst
   END;
   setAllPermissions_in_LGVEst =
        PRE isPermitted_in_LGVEst = allPermissions_in_LGVEst
        THEN
        isPermitted_in_LGVEst := isPermitted_in_LGVEst ∪ isPermitted_in_Kernel_ERTMS
   END;
```

Figure 5.10 – Extrait du modèle de sécurité de la ligne LGV_Est « LGVEst_Model.mch »

De même pour la figure 5.10, un invariant de quantification universelle permet de vérifier si tous les couples (ro, op) dans l'organisation *LGVEst_org* forment un sous ensemble des couples de *isPermitted* tel que *ro* est un rôle de cette organisation. De même pour la définition *allPermissions_in_LGVEst*, elle permet la déduction des couples *(ro,op)* à partir de *permissions_in_LGVEst* en tenant compte de la hiérarchie des rôles.

De même pour l'opération *setPermissions_in_LGVEst*, elle permet d'affecter toutes les permissions sous forme de couple (rôle, opération) de la définition *allPermissions_in_LGVEst* à la variable *isPermitted_in_LGVEst*. L'opération *setAllPermissions_in_ERTMS* permet d'ajouter les permissions de l'organisation mère *Kernel_ERTMS_org* à l'ensemble des permissions propres. D'où, *isPermitted_in_LGVEst* est l'union de *isPermitted_in_LGVEst* (contenant les permissions propres *allPermissions_in_LGVEst*) et *isPermitted_in_Kernel_ERTMS*.

5.7 Exemple de scénario d'accident de *Saint-Romain-en-Gier*

Dans cette section, nous présentons un scénario d'accident de Saint-Romain-en-Gier mettant en exergue la notion de contexte. La description de ce scénario n'est pas formelle. C'est un accident réel qui a lourdement affecté la vie d'êtres humains. En effet, il s'agit de la collision entre un train de travaux et un train commercial.

Une description sommaire et synthétique est illustrée dans ce qui suit [1].

Sur la ligne de Lyon à Saint-Étienne, le lundi 5 avril 2004, une rame TGV-Duplex vide a heurté un train de travaux au km 532,730 sur voie 2 (voir figure 5.11). Le rapport du BEATT [BEA-TT, 2004] précise les circonstances et causes de l'accident et émet des recommandations préventives visant à réduire le niveau de risque dans de telles situations impliquant des trains de travaux. « *La SNCF a en particulier conduit sur cet événement une analyse approfondie du facteur humain, faisant apparaître les mécanismes de l'accident dans le contexte de travail vécu par les acteurs concernés* » [BEA-TT, 2004].

On mettra délibérément en exergue l'expression « contexte de travail vécu par les acteurs concernés ». C'est sur cette notion de contexte que nous nous sommes appuyés pour analyser le scénario d'accident.

Dans cet accident, au moins, trois règles de sécurité ont été mises en échec. Notre objectif est de souligner la criticité des informations manipulées par les acteurs du système en fonction du contexte.

1. Une description détaillée des circonstances de l'accident et l'analyse de la BEATT est disponible dans le lien suivant : http ://www.bea-tt.developpement-durable.gouv.fr/saint-romain-en-gier-r21.html.

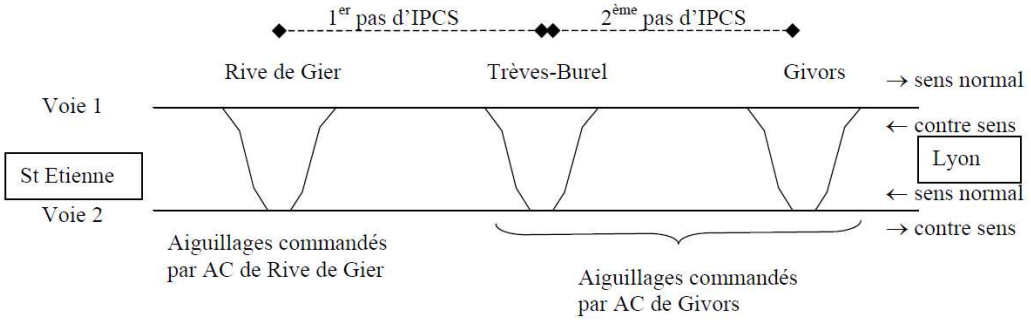

Figure 5.11 – Présentation de l'infrastructure [BEA-TT, 2004]

5.7.1 Présentation du scénario

Nous décrivons le scénario de l'accident extrait du rapport de BEATT de la façon suivante :

- « Des travaux sur l'infrastructure sont organisés dans la zone Rive de Gier/Givors, aussi bien sur la voie 1 que sur la voie 2. Leur programmation a prévu l'engagement de deux trains de travaux sur la voie 1 et sur la voie 2. Le réalisateur a entamé les procédures de demande de protection pour obtenir l'interception des voies 1 et 2 entre Rive de Gier et Givors et permettre l'introduction des trains de travaux. Le chantier de la voie 2 rencontre des difficultés d'exécution. Le planning horaire initialement prévu en subit les conséquences. L'intervalle horaire théorique s'étend de 21h40 le dimanche soir jusqu'à 05h14 le lundi matin. En réalité, la voie 2 interceptée n'est pas encore rendue à cette heure-là. Le lundi 5 avril à 04h15, l'agent circulation (AC) de nuit de Givors-ville remet le service à son collègue de matinée. À ce moment-là, cinq dépêches de protection ont toujours cours. Peu après, l'interception de la voie 1 est levée, les trains de travaux ayant quitté cette voie et les protections correspondantes ayant été levées ; la voie 2 reste interceptée. Le PC (Poste de commandement) de Lyon annonce une circulation se dirigeant de Lyon vers Saint-Étienne. L'agent circulation de Givors-ville établit l'itinéraire permettant d'engager ce TGV par son itinéraire normal voie 2 vers Rive de Gier, puis Saint-Étienne. Après avoir traversé la gare de Givors-ville, à une distance d'environ 4,5 km, le TGV entre en collision contre le train de travaux qui circulait en sens inverse sur cette voie 2 pour regagner la gare de Givors. »

L'analyse qui va suivre a pour but de mettre en évidence l'incidence des différents éléments de contexte sans chercher à conclure.

5.7.2 Analyse du comportement du système et raisonnement de sécurité

Nous définissons tout d'abord, ce qu'est une règle de sécurité. Ensuite, nous décrivons les actions de protection utilisées pour protéger le système.

Une règle de sécurité est définie par le rapport confidentiel de [Collart-Dutilleul, 2009] comme *un ensemble d'actions à mettre en œuvre pour amener l'ensemble des opérations à un niveau de sécurité acceptable, elle inclut :*

1. *un contexte d'application (lieu, moment, type d'intervention)*,
2. *des conditions (contraintes validant l'application de la règle) ; par exemple arrivée signalée d'un train à une distance déterminée*,
3. *une liste d'actions à mettre en œuvre (par exemple déplacement en zone de garage).*

Nous pouvons remarquer que, structurellement, une règle comprend un contexte se déclinant en contexte temporel, spatial, prérequis ou autre. Par ailleurs, nous pouvons trouver une description amenant à considérer l'historique des différents changements d'états du système. Ces subtilités ne sont pas embarquées dans le modèle RBAC, mais elles le sont dans le modèle Or-Bac.

Nous énumérons ci-dessous les trois actions de protection utilisées.

5.7.2.1 La demande d'interception de voie (DIV)

Selon le rapport de [BEA-TT, 2004], la DIV est définie en tant que « *procédure permettant au service chargé de la maintenance de l'infrastructure d'effectuer des travaux sur une portion de ligne ou de voie, dépendant de deux agents de circulation, avec la garantie qu'il n'y aura pas de circulation commerciale pendant une période de temps déterminée* ».

Dans le scénario concerné, une DIV a bien été déposée pour la voie 1 et la voie 2. Deux éléments critiques sont à considérer :

1. Cette DIV était déposée toutes les nuits à l'identique sur la voie 2, mais le jour de l'accident cette dernière avait été étendue. Pour la première fois cette nuit-là, la DIV s'étend de Rive de Gier à Givors, à la différence des DIV des nuits précédentes qui étaient des DIV simples couvrant Rive de Gier à Trèves Burel.
2. Au moment de l'accident, la DIV qui concernait la voie 1 avait été libérée.

Le rapport de [BEA-TT, 2004] mentionne que « dans le modèle cognitif de l'agent », la portion Rive de Gier-Givor n'a pas à être occupée. La DIV était supposée avoir la même emprise que celle de la veille.

Toute modification aurait dû être signalée. Reste qu'une protection contre le malentendu existe et qu'elle a été appliquée : l'agent précédent avait appliqué des caches physiques qui interdisaient matériellement d'ouvrir un itinéraire sur la voie 2 de Givors vers Rive de Giers. Ces caches devaient être démontés lors de la levée de l'interdiction de circulation sur la voie 2. Donc, si les caches sont en place, l'interdiction n'est pas levée. L'utilisation du contexte provisionnel est, dans ce cas, envisagée.

Sous la pression du fonctionnement, l'agent de circulation a utilisé la pointe de son stylo pour passer outre la protection. Remarquons la nature des informations associées à une DIV :

— Une DIV va être levée et donc la circulation pourra reprendre.
— Une DIV est planifiée pour une période donnée, mais tant que cette dernière n'est pas levée la circulation n'est pas autorisée.

Dans le cas d'une DIV, le contexte temporel n'est pas de sécurité : il sert à la planification.

5.7.2.2 La protection de chantier

L'agent de sécurité responsable du chantier doit protéger le chantier, indépendamment des protections qui sont demandées par ailleurs (la DIV). L'agent de sécurité doit donc déposer un drapeau en travers de la voie (appelé signal d'arrêt à main, SAM) et des pétards (feux d'artifices qui se déclencheront si un train force le passage). Tout train pénétrant dans l'emprise du chantier est donc forcément conscient qu'il est dans une zone de chantier.

Cette action n'a pas été effectuée par l'agent de sécurité dans le cadre de notre scénario.

5.7.2.3 Installation permanente de contre sens (IPCS)

Les lignes à fort trafic peuvent être équipées d'installations de signalisation permettant aux trains de circuler dans les deux sens sur chaque voie. Dans ce type d'installation, la signalisation est utilisée dans un sens ou dans l'autre. L'interprétation du « ou » logique dans cette phrase est exclusive. La signalisation du sens non circulé est éteinte et les conducteurs sont supposés s'arrêter devant un signal éteint. C'est une protection contre le face à face entre deux trains.

Dans le cas qui nous intéresse, le conducteur du train de travaux s'est engagé en franchissant un signal éteint. Ce signal était éteint car l'agent de circulation permettait le passage d'un train commercial dans le sens opposé au sien.

La collision a eu lieu à basse vitesse (20km/h pour le train de chantier et 10km/h pour le train commercial). Le train commercial avait été ralenti par la signalisation. Cette signalisation garantit la non collision des trains circulant dans le même sens. Malheureusement, le train commercial et le train de chantier se déplaçaient dans des sens opposés. On peut penser que le conducteur du train de chantier se pensait protégé par la DIV.

5.7.3 Discussion

Une partie des erreurs qu'on peut noter dans cet accident pourrait s'expliquer par des erreurs d'appréciations du contexte. Le raisonnement de sécurité général considère l'ensemble des actions de sécurité. Toute analyse partielle du contexte peut nous amener à prendre des initiatives contraires à la sécurité. Il est donc judicieux d'utiliser un formalisme précis et structuré pour manipuler ces informations contextuelles critiques. Le

formalisme Or-Bac nous semble particulièrement intéressant parce qu'il introduit un attribut de contexte structuré en différents champs. Une spécialisation spécifique au contexte ferroviaire doit cependant être envisagée.

5.8 Conclusion

Dans ce chapitre, nous avons présenté une deuxième contribution de nos travaux. Cette contribution consiste à utiliser un autre modèle « Organization Based Access Control » pour les politiques de sécurité des systèmes d'information centré sur la notion d'organisation. Ce modèle est considéré comme extension du modèle RBAC. En effet, il se base sur des notions déjà introduites par le modèle RBAC telles que rôle, permission, utilisateur et action. En revanche, il ajoute d'autres notions comme la hiérarchie des organisations, la notion de contexte et d'autres types d'autorisations (interdiction, obligation, recommandation).

Afin de modéliser les règles d'exploitation ferroviaires, nous avons exploité les deux notions de hiérarchie des organisations et de contexte. La hiérarchie des organisations met en avant l'interopérabilité en ERTMS, tandis que la notion de contexte permet de structurer les contraintes permettant de délivrer une autorisation. Par conséquent, nous avons tout d'abord présenté l'intérêt de ces deux notions dans le contexte ferroviaire et nous l'avons illustré par des exemples concrets. Puis, nous avons décrit le méta-modèle d'Or-Bac étendu ainsi que le profil UML associé. Ensuite, nous avons proposé l'architecture en B et la formalisation des permissions de chacune des organisations selon la hiérarchie des organisations. Ces dernières ont été considérées comme extension du modèle RBAC. En effet, nous avons ajouté des machines pour chacune des organisations en nous appuyant sur le modèle de sécurité de RBAC. Cependant, la transformation UML/B n'est pas encore implantée dans la plate-forme B4MSecure. En outre, nous n'avons pas encore proposé une formalisation en B qui tient compte des différentes variantes de contextes que nous avons définies en UML. Jusqu'à présent, la formalisation du contexte est exploitée par une traduction manuelle des contraintes OCL en B sous forme d'invariants ou de pré-conditions dans les opérations B. Enfin, nous avons illustré la notion de contexte par un scénario réel d'accident de *Saint-Romain-en-Gier*.

Chapitre 6

CONCLUSION ET PERSPECTIVES

La gestion de la signalisation dans le système européen de gestion du trafic ferroviaire (ERTMS) doit collaborer avec des règles nationales propres à chaque pays. Dans ce cadre, le projet *PERFECT* contribue à l'évaluation du système en termes d'exigences de sécurité tout en assurant la cohérence et la consistance des règles d'exploitation au regard des règles nationales et européennes. C'est dans ce contexte que s'articulent les travaux de cette thèse au sein du projet *PERFECT*. Nos contributions portent sur deux volets, à savoir : la phase de spécification de la réglementation et la phase de vérification et de validation des spécifications avec une méthode formelle. Dans cette démarche, nous avons commencé par étudier le couplage semi-formel/formel des notations permettant la formalisation des spécifications informelles. Suite à cette étape, les notations UML et B ont été choisies. D'une part, UML, par ses notations semi-formelles et ses aspects graphiques, permet une vue synthétique et intuitive du système. D'autre part, la méthode formelle B offre la précision et la rigueur lors de la formalisation des spécifications.

Suite au choix des notations, nous avons poursuivi notre démarche par l'analyse des règles d'exploitation. L'objectif de cette analyse est de proposer une solution méthodologique dirigée par les modèles pour la modélisation de ces règles et leur validation. Une telle solution a nécessité d'extraire les composants conceptuels permettant la définition et la description de ces règles. En effet, ces règles consistent en des autorisations de déplacement des trains effectuées par des acteurs humains ou informatiques du système en fonction des contraintes. Ces concepts sont définis de façon analogue à la description des politiques de sécurité de contrôle d'accès pour les systèmes d'information. Cette problématique est en effet liée à la sécurité-confidentialité pour les systèmes d'information. D'autre part, nos travaux portent sur la sécurité-innocuité des systèmes ferroviaires. Malgré la différence entre ces deux problématiques, nous avons réussi à exploiter les approches existantes de la première pour répondre à la deuxième. Pour ce faire, nous avons choisi un cas d'étude ferroviaire fondé sur un scénario nominal d'autorisation de mouvement « MA » et un scénario exceptionnel de franchissement d'un arrêt ETCS « Override EOA ». Ces règles sont extraites des principes et des règles d'exploitation du système ETCS sur la ligne à grande vitesse LGV-Est Européenne du sol français.

Dans la première partie de nos contributions, nous avons utilisé le modèle Role Based Access Control (RBAC) centré sur le concept de rôle dans le domaine ferroviaire à l'aide de la plate-forme B4MSecure. Cette approche nous a permis une séparation des préoccupations des aspects fonctionnels et des aspects de sécurité. À cet effet, nous avons élaboré un modèle UML fonctionnel contenant les différentes entités du système et un modèle UML de sécurité contenant les permissions (autorisations) accordées aux rôles. Ces modèles ont ensuite été traduits en spécifications B sur deux machines, l'une fonctionnelle et l'autre de sécurité. À ces spécifications B résultantes, nous avons ajouté des propriétés de sécurité liées à l'étude de cas choisie. Nous avons par la suite utilisé l'animation de *ProB* et la preuve de l'*Atelier B* pour vérifier ces propriétés et valider les scénarios.

Dans la deuxième partie de nos contributions, nous avons étudié et proposé l'utilisation du modèle Or-Bac, comme une extension du modèle RBAC, sans pour autant l'implémenter. Centré sur le concept d'organisation, le modèle Or-Bac permet de favoriser l'interopérabilité, l'un des enjeux majeurs d'ERTMS. À cet égard, nous avons proposé une vision sur la hiérarchie des organisations telle qu'une ligne particulière hérite des autorisations ERTMS tout en exprimant le contexte national correspondant à cette ligne. Les différentes variantes de contexte du modèle Or-Bac répondent à notre logique métier. De plus, nous avons ajouté un nouveau type de contexte lié au mouvement dynamique du train. Suite à nos réflexions sur l'intérêt du modèle Or-Bac pour une meilleure modélisation et validation des règles ferroviaires en vue de l'interopérabilité, nous avons proposé d'ajouter les différents types de contextes au méta-modèle Or-Bac, extension du méta-modèle RBAC. De plus, nous avons proposé une formalisation en B des modèles de sécurité selon la hiérarchie des organisations en nous basant sur les machines RBAC.

Nous désirons proposer, dans un premier temps, une formalisation en B des différents types de contextes. Cette proposition permettra de structurer les contraintes OCL en B de façon à garder une traçabilité orientée objet du contexte. Ceci facilitera les activités de vérification et de validation des règles d'exploitation. De plus, la formalisation des scénarios d'accidents tels que l'accident de *Saint-Romain-en-Gier* en fonction des contextes permet de dégager des informations pertinentes pour fournir une grille d'analyse de l'accident. Dans un deuxième temps, nous envisageons d'implémenter les règles de transformation qui concernent la formalisation de la notion de contexte ainsi que la formalisation de la hiérarchie des organisations que nous avons proposée.

Quelques améliorations de l'implémentation du modèle RBAC dans la plate-forme B4MSecure sont envisagées dans le but de réduire les obligations de preuve générées. Nous proposons de procéder par un mécanisme de raffinement du modèle fonctionnel par le modèle de sécurité. Nous devrions également réduire l'espace d'état de nos modèles

générés automatiquement afin de profiter du model checking de *ProB*.

Nous visons dans des travaux futurs à explorer les diagrammes UML dynamiques tels que les diagrammes de séquence. En effet, la plupart des scénarios à valider sont prédéfinis et connus à l'avance. Une transformation automatique du diagramme de séquence en B permettrait donc d'automatiser l'animation des scénarios et de valider la séquence des opérations. L'utilisation de la preuve de l'*Atelier B* permet alors, de vérifier automatiquement et/ou interactivement la cohérence de la séquence des opérations. Dans la définition des règles de transformation du diagramme de séquence en B, nous devrions ajouter les invariants permettant de prendre en considération sa cohérence avec le diagramme de classes, à savoir :

- Le nom de chaque message du diagramme de séquence doit correspondre à une méthode d'une classe.
- Chaque objet d'une ligne de vie du diagramme de séquence doit correspondre à une classe du diagramme de classes.
- Chaque message du diagramme de séquence doit correspondre à une opération de l'objet destinataire. Autrement dit, le message du diagramme de séquence doit correspondre à une méthode dans la classe de l'objet de la ligne de vie recevant le message.

Dans nos travaux, nous n'avons pas suivi de démarche spécifique pour la traçabilité des exigences. Pourtant, cette dernière permet de faciliter la traduction des spécifications informelles en spécifications semi-formelles et ensuite en spécifications formelles. Nous envisageons donc par la suite d'étudier la traçabilité des exigences fonctionnelles et de sécurité ferroviaire en utilisant les diagrammes d'exigences de SysML (System Modeling Language) dans le cadre de l'ingénierie système. Pour cette raison, nous avons pensé à enrichir le méta-modèle Or-Bac par des méta-classes « FunctionalRequirement » et « SafetyRequirement ».

Le modèle Or-Bac, selon sa version définie pour la modélisation des politiques de sécurité de contrôle d'accès, est structurée en deux couches, une couche abstraite et une couche concrète. Dans la couche abstraite, une activité est une abstraction d'un ensemble d'actions qui ont un même objectif dans une organisation. Nous pourrons exploiter et intégrer cette entité dans le domaine ferroviaire. En effet, une activité peut être vue comme une abstraction des différentes implémentations de la même action sur différentes lignes ou encore une abstraction d'un scénario composé d'actions (séquence et/ou parallélisme d'actions). De plus, le modèle Or-Bac permet de spécifier différents types d'autorisations à savoir : les permissions, les interdictions, les recommandations et les obligations. Seules les permissions sont abordées par l'approche utilisée. En revanche, une interdiction sur une action se traduit par l'absence de permission sur cette même action. Cette considération

nous a conduit à définir une organisation mère fictive d'un noyau ERTMS contenant les permissions communes. Les organisations filles des lignes nationales particulières héritent alors de cette organisation mère et ajoutent des permissions. Il est indéniable que cela simplifie la formalisation des règles d'exploitation, surtout en B, mais ne spécifie pas les interdictions d'une façon explicite. Nous proposons donc d'étudier la possibilité d'intégrer l'interdiction dans l'approche de modélisation ainsi que les autres types d'autorisations.

Enfin, les travaux de cette thèse ouvrent des horizons vers un projet de l'IRT Railenium intitulé « ERTMS régional » dont une partie s'appuie sur les approches que nous avons utilisées pour la modélisation des règles d'exploitation, de la signalisation et de l'enclenchement du système ferroviaire.

Bibliographie

[Abdallah et Khayat, 2006] Abdallah, A. E. et Khayat, E. J. (2006). Formal Z Specifications of Several Flat Role-Based Access Control Models. *In Software Engineering Workshop, 2006. SEW '06. 30th Annual IEEE/NASA*, pages 282–292. (Cité dans la page 85.)

[Abou EL Kalam, 2003] Abou EL Kalam, A. (2003). *Politiques et Modèles de Sécurité pour les domaines de la santé et des affaires sociales.* Thèse de doctorat, Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS). (Cité dans les pages 26 and 105.)

[Abrial, 1996] Abrial, J. R. (1996). *The B-Book : Assigning Programs to Meanings.* Cambridge University Press. (Cité dans les pages 65, 66, 67 and 68.)

[Abrial, 2003] Abrial, J.-R. (2003). B : passé, présent, futur. *TSI. Technique et science informatiques*, 22(1):89–118. (Cité dans la page 51.)

[Abrial, 2010] Abrial, J.-R. (2010). *Modeling in Event-B : System and Software Engineering.* Cambridge University Press, New York, NY, USA, 1st édition. (Cité dans la page 57.)

[Akoka *et al.*, 2010] Akoka, J., Aubonnet, T., Bucumi, J. S., Comyn-Wattiau, I., Idani, A., Labiadh, M. A., Lammari, N., Ledru, Y. et Richier, J.-L. (2010). Intégration des propriétés de sécurité dans UML - Livrable n°2.1 du projet SELKIS : Une méthode de développement de systèmes d'information médicaux sécurisés : de l'analyse des besoins à l'implémentation. Rapport technique, Laboratoire CEDRIC, CNAM Paris, France. (Cité dans les pages 83 and 87.)

[Anastasakis *et al.*, 2007] Anastasakis, K., Bordbar, B., Georg, G. et Ray, I. (2007). Uml2alloy : A challenging model transformation. *In* Engels, G., Opdyke, B., Schmidt, D. et Weil, F., éditeurs : *Model Driven Engineering Languages and Systems*, volume 4735 de *Lecture Notes in Computer Science*, pages 436–450. Springer Berlin Heidelberg. (Cité dans les pages 55 and 56.)

[Anastasakis *et al.*, 2010] Anastasakis, K., Bordbar, B., Georg, G. et Ray, I. (2010). On challenges of model transformation from uml to alloy. *Software & Systems Modeling*, 9(1):69–86. (Cité dans les pages 55 and 56.)

[ANSI, 2004] ANSI (2004). *American National Standard for Information Technology - Role Based Access Control*. INCITS 359-2004. (Cité dans les pages 83 and 84.)

[ANSI, 2012] ANSI (2012). *American National Standard for Information Technology - Role Based Access Control*. INCITS 359-2012. (Cité dans la page 83.)

[Badreau et Boulanger, 2014] Badreau, S. et Boulanger, J.-L. (2014). *Ingénierie Des Exigences : Méthodes et bonnes pratiques pour construire et maintenir un référentiel*. Dunod. (Cité dans la page 73.)

[Barais, 2007] Barais, O. (2007). Séparation des préoccupations en phase de méta-modélisation. http ://www2.lifl.fr/ mullera/CoMo07.htm. (Cité dans la page 29.)

[Bazex *et al.*, 2003] Bazex, P., Bodeveix, J.-P., Le Camus, C., Millan, T. et Percebois, C. (2003). Vérification de modèles uml fondée sur ocl. *In INFORSID, Nancy, 03/06/2003-06/06/2003*, pages 185–200, http ://praxinsa.insa-lyon.fr. INFORSID (actes électroniques). (Cité dans la page 57.)

[BEA-TT, 2004] BEA-TT (2004). Rapport d'enquête technique sur l'accident ferroviaire du 5 avril 2004 à saint-romain-en-gier. Rapport technique, Ministère de l'Équipement, des Transports, de l'Aménagement du Territoire, du Tourisme et de la Mer, METATTM. (Cité dans les pages viii, 123, 124 and 125.)

[Behm *et al.*, 1999] Behm, P., Benoit, P., Faivre, A. et Meynadier, J. M. (1999). Météor : A Successful Application of B in a Large Project. *In* Wing, J., Woodcock, J. et Davies, J., éditeurs : *FM'99 - Formal Methods*, volume 1708 de *Lecture Notes in Computer Science*, pages 369–387. Springer Berlin Heidelberg. (Cité dans les pages 12, 58 and 65.)

[Bell et LaPadula, 1976] Bell, D. E. et LaPadula, L. J. (1976). Secure Computer Systems : Unified Exposition and Multics Interpretation, MTR 2997 Rev. 1. MITRE corp. Bedford (Massachussetts, USA). (Cité dans la page 26.)

[Ben Ayed *et al.*, 2014a] Ben Ayed, R., Bon, P. et Collart-Dutilleul, S. (2014a). Checking the European Railways Traffic Management System (ERTMS) operating rules using UML and B method. *In 14th International conference on Railway Engineering Design and Optimization*, pages 139–149, Italy. (Cité dans la page 71.)

[Ben Ayed *et al.*, 2014b] Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Idani, A. et Ledru, Y. (2014b). B Formal Validation of ERTMS/ETCS Railway Operating Rules.

In 4th International ABZ Conference, pages 124–129, France. (Cité dans les pages 71 and 90.)

[Ben Ayed *et al.*, 2014c] Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Ledru, Y. et Idani, A. (2014c). Modélisation et validation formelle des règles d'exploitation ferroviaires. *In Approches Formelles dans l'Assistance au Développement de Logiciels*, pages 1–15, France. (Cité dans les pages 71 and 90.)

[Ben Ayed *et al.*, 2015] Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Ledru, Y. et Idani, A. (2015). Formalismes basés sur les rôles pour la modélisation et la validation des règles d'exploitation ferroviaires. *In Technique et Science Informatiques*, pages 495–521, France. Lavoisier. Extension AFADL 2014. (Cité dans la page 71.)

[Booch *et al.*, 2005] Booch, G., Rumbaugh, J. et Jacobson, I. (2005). *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional. (Cité dans la page 60.)

[Booth, 2006] Booth, P. D. (2006). Intermittent and continuous automatic train protection. Railway Signalling and Control Systems, the 11th IET Professional Development Course on. (Cité dans la page 5.)

[Boulanger, 2012] Boulanger, J.-L. (2012). *Outils de mise en œuvre industrielle des techniques formelles*. Lavoisier. (Cité dans la page 52.)

[Boulanger, 2014] Boulanger, J. L. (2014). *B-RAIL : UML to B Transformation in Modeling a Level Crossing*. John Wiley & Sons, Inc. (Cité dans la page 101.)

[Butler, 2009] Butler, M. (2009). Incremental design of distributed systems with event-b. *In* Broy, M., Sitou, W. et Hoare, T., éditeurs : *Engineering Methods and Tools for Software Safety and Security - Marktoberdorf Summer School 2008*, pages 131–160. IOS Press. Chapter : 4. (Cité dans la page 57.)

[Bézivin, 2004] Bézivin, J. (2004). Sur les principes de base de l'ingénierie des modèles. *RSTI-L'objet*, 10(4):145–157. (Cité dans la page 27.)

[Bézivin, 2005] Bézivin, J. (2005). On the unification power of models. *Software and Systems Modeling*, 4(2):171–188. (Cité dans les pages vii, 27, 28 and 29.)

[Bézivin *et al.*, 2004] Bézivin, J., Blay, M., Bouzhegoub, M., Estublier, J., Favre, J.-M., Gérard, S. et Jézéquel, J. M. (2004). Rapport de Synthèse de l'AS CNRS sur

le MDA (Model Driven Architecture). CNRS. (Cité dans les pages 29, 34, 35 and 36.)

[Bézivin *et al.*, 2001] Bézivin, J., No, C. D. G., Nantes, J. B., Gerbé, O. et Montréal, H. (2001). Towards a Precise Definition of the OMG/MDA Framework. (Cité dans la page 27.)

[Cancila *et al.*, 2009a] Cancila, D., Dubois, H. et Adedjouma, M. (2009a). Sûreté de Fonctionnement dans un processus basé sur l'Ingénierie Dirigée par les Modèles. *In Actes des 5èmes journées sur l'Ingénierie Dirigée par les Modèles*, pages 73–78. (Cité dans la page 42.)

[Cancila *et al.*, 2009b] Cancila, D., Terrier, F., Belmonte, F., Dubois, H., Espinoza, H., Gérard, S. et Cuccuru, A. (2009b). Sophia : a modeling language for model-based safety engineering. (Cité dans la page 42.)

[Cariou *et al.*, 2009] Cariou, E., Belloir, N., Barbier, F. et Djemam, N. (2009). OCL contracts for the verification of model transformations. *ECEASST*, 24. (Cité dans la page 57.)

[Cariou *et al.*, 2004] Cariou, E., Marvie, R., Seinturier, L. et Duchien, L. (2004). Ocl for the specification of model transformation contracts. *In in Proceedings of Workshop OCL and Model Driven Engineering.* (Cité dans la page 57.)

[Carrozza *et al.*, 2012] Carrozza, G., Faella, M., Fucci, F., Pietrantuono, R. et Russo, S. (2012). Integrating mdt in an industrial process in the air traffic control domain. *In Proceedings of the 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, ISSREW '12, pages 225–230, Washington, DC, USA. IEEE Computer Society. (Cité dans la page 37.)

[Carrozza *et al.*, 2013] Carrozza, G., Faella, M., Fucci, F., Pietrantuono, R. et Russo, S. (2013). Engineering air traffic control systems with a model-driven approach. *IEEE Software*, 30(3):42–48. (Cité dans la page 37.)

[Cavalli *et al.*, 2014] Cavalli, A., Santos, J., Nguyen, H.-N., Behrens, M., Rieger, S., Braunstein, C., Steinke, U., Lucet, B., Güdemann, M., Gombault, B., Petit-Doche, M., Nitsch, A., Beichler, B., Zilio, S. D., Ge, N. et Pantel, M. (2014). D.4.2.1 1st interim V&V report on the applicability of the V&V approach to the formal abstract model. Rapport technique. (Cité dans la page 101.)

[CENELEC, NF EN 50126, 2000] CENELEC, NF EN 50126 (2000). Applications fer-

roviaires : Spécification et démonstration de la fiabilité, de la disponibilité, de la maintenabilité et de la sécurité (FMDS). (Cité dans la page 25.)

[CENELEC, NF EN 50128, 2001] CENELEC, NF EN 50128 (2001). Applications ferroviaires : Systèmes de signalisation, de télécommunication et de traitement - Logiciels pour systèmes de commande et de protection ferroviaire. (Cité dans la page 25.)

[CENELEC, NF EN 50129, 2003] CENELEC, NF EN 50129 (2003). Applications ferroviaires : Systèmes de signalisation, de télécommunication et de traitement - Systèmes électroniques de sécurité pour la signalisation. (Cité dans la page 25.)

[Chettaoui, 2008] Chettaoui, H. (2008). *Interopérabilité entre modèles hétérogènes en conception coopérative par des approches d'Ingénierie Dirigée par les Modèles.* Theses, Institut National Polytechnique de Grenoble - INPG. (Cité dans la page 35.)

[Clearsy System Engineering, 2009] Clearsy System Engineering (2009). Manuel de Référence du Langage B - Version 1.8.8. Rapport technique. (Cité dans les pages 65 and 68.)

[Clearsy System Engineering, 2011a] Clearsy System Engineering (2011a). Atelier B. Prouveur Interactif Manuel de référence - Version 4.0. Rapport technique. (Cité dans la page 99.)

[Clearsy System Engineering, 2011b] Clearsy System Engineering (2011b). Obligations de preuve Manuel de référence- Version 3.7. Rapport technique. (Cité dans la page 69.)

[Colin *et al.*, 2005] Colin, S., Petit, D., Rocheteau, J., Marcano, R., Mariano, G. et Poirriez, V. (2005). Brillant : an open source and xml-based platform for rigourous software development. *In Software Engineering and Formal Methods, 2005. SEFM 2005. Third IEEE International Conference on*, pages 373–382. (Cité dans la page 70.)

[Collart-Dutilleul, 2009] Collart-Dutilleul, S. (2009). Construction et structuration des règles de sécurité - convention de recherche resciproc. Rapport technique. (Cité dans la page 125.)

[Collart-Dutilleul *et al.*, 2013] Collart-Dutilleul, S., Bon, P. et Ferlin, A. (2013). Livrable D1.1 du projet PERFECT. Description d'une méthodologie nationale et proposition de cas d'étude. Rapport technique. (Cité dans les pages vii and 9.)

[Combemale, 2008a] Combemale, B. (2008a). *Approche de métamodélisation pour la simulation et la vérification de modèle – Application à l'ingénierie des procédés*. Thèse de doctorat, Institut National Polytechnique, Université de Toulouse. in french. (Cité dans les pages vii and 33.)

[Combemale, 2008b] Combemale, B. (2008b). Ingénierie dirigée par les modèles (IDM) - État de l'art. (Cité dans les pages 27, 30 and 32.)

[Combemale *et al.*, 2006] Combemale, B., Rougemaille, S., Crégut, X., Migeon, F., Pantel, M. et Maurel, C. (2006). Expériences pour décrire la sémantique en ingénierie des modèles. *In* Sciences/Lavoisier, H., éditeur : *2ème journées sur l'Ingénierie Dirigée par les Modèles (IDM, in french)*, pages 17–34, Lilles, France. (Cité dans la page 29.)

[Cook, 2004] Cook, S. (2004). Domain-Specific Modeling and Model Driven Architecture. *In MDA Journal*. (Cité dans la page 36.)

[Cunha *et al.*, 2015] Cunha, A., Garis, A. et Riesco, D. (2015). Translating between alloy specifications and uml class diagrams annotated with ocl. *Software & Systems Modeling*, 14(1):5–25. (Cité dans les pages 55 and 56.)

[Cuppens et Miege, 2003] Cuppens, F. et Miege, A. (2003). Modelling contexts in the or-bac model. *In Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 416–425. (Cité dans les pages 104 and 105.)

[Cuppens et Miège, 2004] Cuppens, F. et Miège, A. (2004). Or-Bac. Organization Based Access Control. Journées Druide, Le Croisic. (Cité dans les pages viii, 104 and 106.)

[Czarnecki et Helsen, 2003] Czarnecki, K. et Helsen, S. (2003). Classification of model transformation approaches. *In 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*, Anaheim, CA, USA. (Cité dans la page 34.)

[Dahl, 2004] Dahl, O.-J. (2004). The birth of Object Orientation : the Simula Languages. *In* Owe, O., Krogdahl, S. et Lyche, T., éditeurs : *From Object-Orientation to Formal Methods*, volume 2635 de *Lecture Notes in Computer Science*, pages 15–25. Springer Berlin Heidelberg. (Cité dans la page 26.)

[de Roquemaurel *et al.*, 2011a] de Roquemaurel, M., Bodeveix, J.-P. et Filali, M. (2011a). Réécriture de contraintes OCL (regular paper). *In Journées sur l'Ingénierie Di-*

rigée par les Modèles (IDM), Lille, 07/06/2011-10/06/2011, pages 127–133, http ://gdr-gpl.cnrs.fr. CNRS - GDR GPL. (Cité dans les pages 55 and 56.)

[de Roquemaurel *et al.*, 2011b] de Roquemaurel, M., Polacsek, T., Rolland, J.-F., Bodeveix, J.-P. et Filali, M. (2011b). Assistance à la conception de modèles à l'aide de contraintes. *In Journées Francophones sur les Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL)*, pages 121–136. (Cité dans la page 55.)

[Dollé *et al.*, 2003] Dollé, D., Essamé, D. et Falampin, J. (2003). B dans le transport ferroviaire : L'expérience de Siemens Transportation Systems. *TSI. Technique et science informatiques*, 22(1):11–32. (Cité dans les pages 12 and 65.)

[Dos Santos *et al.*, 2010] Dos Santos, O., Woodcock, J., Paige, R. et King, S. (2010). The use of model transformation in the iness project. *In* de Boer, F., Bonsangue, M., Hallerstede, S. et Leuschel, M., éditeurs : *Formal Methods for Components and Objects*, volume 6286 de *Lecture Notes in Computer Science*, pages 147–165. Springer Berlin Heidelberg. (Cité dans la page 42.)

[Duarte *et al.*, 2003] Duarte, R., Júnior, J. et A.Mota (2003). Precise Modeling with UML : Why OCL ? *In Workshop of Formal Methods*. (Cité dans la page 55.)

[Dupuy, 2000a] Dupuy, S. (2000a). *Integrating semi-formal and formal notations for information system specification*. Theses, Université Joseph-Fourier - Grenoble I. (Cité dans la page 48.)

[Dupuy, 2000b] Dupuy, S. (2000b). Vers une prise en compte des contraintes en UML grâce à Z. *In INFORSID*. (Cité dans la page 55.)

[El Amraoui et Mesghouni, 2014] El Amraoui, A. et Mesghouni, K. (2014). Colored petri net model for discrete system communication management on the european rail traffic management system (ertms) level 2. *In Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on*, pages 248–253. (Cité dans la page 72.)

[El Miloudi *et al.*, 2013] El Miloudi, K., El Amrani, Y. et Ettouhani, A. (2013). Using formal specification for ensuring consistency in multi-view modeling. *Journal of Theoretical and Applied Information Technology*, 57(3):407–411. (Cité dans la page 56.)

[El-Miloudi et Ettouhami, 2015] El-Miloudi, K. et Ettouhami, A. (2015). A Multi-View Approach for Formalizing UML State Machine Diagrams Using Z Notation.

WSEAS Transactions on Computers, 14:72–78. (Cité dans la page 56.)

[Eriksson, 2006] Eriksson, L.-H. (2006). Use of domain theories in applied formal methods. Rapport technique, Uppsala University, Dept. of Information Technology, 2006-029. (Cité dans la page 57.)

[Faivre *et al.*, 2015] Faivre, A., Lapitre, A., Lanusse, A., Périn, M., Rangra, S., Sallak, M. et Schön, W. (2015). Two methods for modeling and verification of safety properties of railway infrastructures. (Cité dans la page 73.)

[Fanmuy *et al.*, 2001] Fanmuy, G., Levy, G., Foisseau, J., Lamothe, P., Hermans, B., De Chazelles, P. et Choveau, E. (2001). Recommandations pour l'élaboration d'un référentiel d'exigences techniques. Rapport technique. (Cité dans la page 21.)

[Favre *et al.*, 2006] Favre, J.-M., Establier, J. et Blay-Fornarino, M., éditeurs (2006). *L'ingénierie dirigée par les modèles : au-delà du MDA*. Hermes-Lavoisier, Cachan, France. (Cité dans la page 43.)

[Ferier *et al.*, 2011] Ferier, L., Pinte, S. et Blasband, D. (2011). ERTMS Formal Specs : a domain specific language to formalize ERTMS specifications for onboard unit development. *In 9th World Congress on Railway Research - Lille*. (Cité dans la page 101.)

[Ferraiolo et Kuhn, 1992] Ferraiolo, D. F. et Kuhn, D. R. (1992). Role-based access controls. *In 15th National Computer Security Conference, Baltimore MD*, pages 554–563. (Cité dans la page 83.)

[Ferraiolo *et al.*, 2001] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R. et Chandramouli, R. (2001). Proposed NIST Standard for Role-based Access Control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274. (Cité dans les pages vi, 83 and 151.)

[Fondement *et al.*, 2010] Fondement, F., Muller, P.-A., Wittman, B., Ambert, F., Bouquet, F., Lasalle, J., Oudot, E., Peureux, F., Legeard, B., Alter, M. et Scherrer, C. (2010). Vetess : Idm, test et sysml. *Génie Logiciel*, (93):43–48. (Cité dans la page 37.)

[Fraser *et al.*, 1991] Fraser, M. D., Kumar, K. et Vaishnavi, V. K. (1991). Informal and formal requirements specification languages : Bridging the gap. *IEEE Trans. Softw. Eng.*, 17(5):454–466. (Cité dans la page 54.)

[Gervais, 2004] Gervais, F. (2004). EB$\hat{4}$: Vers une méthode combinée de spécification formelle des systèmes d'information. Rapport technique. Rapport pour un examen de spécialité. (Cité dans la page 56.)

[Groupe de Travail Ingénierie Système de l'AFIS, 2009] Groupe de Travail Ingénierie Système de l'AFIS (2009). Découvrir et comprendre l'ingénierie système - version 3. Rapport technique. (Cité dans la page 21.)

[Hallerstede *et al.*, 2013] Hallerstede, S., Leuschel, M. et Plagge, D. (2013). Validation of formal models by refinement animation. *Science of Computer Programming*, 78(3):272 – 292. Abstract State Machines, Alloy, B and Z - Selected Papers from {ABZ} 2010. (Cité dans la page 53.)

[Hammoudi, 2010] Hammoudi, S. (2010). *Contribution à l'étude et à l'application de l'ingénierie dirigée par les modèles.* Habilitation à diriger des recherches, Université d'Angers. (Cité dans la page 29.)

[Hardebolle, 2008] Hardebolle, C. (2008). *Composition of models for the multi-paradigm modeling of the behavior of systems.* Theses, Université Paris Sud - Paris XI. (Cité dans la page 35.)

[Haxthausen, 2010] Haxthausen, A. E. (2010). *An Introduction to Formal Methods for the Development of Safety-critical Applications.* (Cité dans les pages 52 and 53.)

[Héon *et al.*, 2010] Héon, M., Basque, J. et Paquette, G. (2010). Validation de la sémantique d'un modèle semi-formel de connaissances avec OntoCASE. *In* DESPRES, S., éditeur : *21èmes 21es Journées Francophones d'Ingénierie des Connaissances*, page 55 à 66, NÎmes, France. Ecole des Mines d'Alès. (Cité dans la page 48.)

[Hinchey et Bowen, 1995] Hinchey, M. G. et Bowen, J. P. (1995). *Applications of Formal Methods.* Prentice Hall International Series in Computer Science. (Cité dans les pages 50 and 53.)

[Hu *et al.*, 2006] Hu, V. C., Ferraiolo, D. F. et Kuhn, D. R. (2006). Assessment of Access Control Systems, Interagency Report 7316. Rapport technique, National Institute of Standards and Technology. (Cité dans la page 25.)

[Huynh *et al.*, 2014] Huynh, N., Frappier, M., Mammar, A., Laleau, R. et Desharnais, J. (2014). Validating the RBAC ANSI 2012 Standard Using B. *In* Ait Ameur, Y. et Schewe, K.-D., éditeurs : *Abstract State Machines, Alloy, B, TLA, VDM, and*

Z : 4th International Conference, ABZ 2014, Toulouse, France, June 2-6, 2014. Proceedings, pages 255–270, Berlin, Heidelberg. Springer Berlin Heidelberg. (Cité dans la page 85.)

[Idani, 2006] Idani, A. (2006). *B/UML : Mise en relation de spécifications B et de descriptions UML pour l'aide à la validation externe de développements formels en B*. Thèse de doctorat, Université Joseph Fourier, Grenoble 1, Informatique. (Cité dans les pages vii, 59, 60 and 64.)

[Idani *et al.*, 2010] Idani, A., Labiadh, M.-A. et Ledru, Y. (2010). Infrastructure dirigée par les modèles pour une intégration adaptable et évolutive de uml et b. *Ingénierie des Systèmes d'Information*, 15(3):87–112. (Cité dans les pages 59, 87, 95 and 112.)

[Idani *et al.*, 2013] Idani, A., Ledru, Y. et Labiadh, M. (2013). B4MSecure : une plateforme IDM pour la modélisation et la validation de politiques de sécurité en Systèmes d'Information. *In Journées Francophones sur les Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL)*, pages 85–89. (Cité dans la page 86.)

[Idani *et al.*, 2014] Idani, A., Ledru, Y. et Radhouani, A. (2014). Modélisation graphique et validation formelle de politiques RBAC en systèmes d'information. plateforme b4msecure. *Ingénierie des Systèmes d'Information*, 19(6):33–61. (Cité dans les pages vii, 86, 87 and 88.)

[IEC, IEC 61508, Norme Internationale, 1998] IEC, IEC 61508, Norme Internationale (1998). *Sécurité fonctionnelle des systèmes électriques électroniques programmables relatifs à la sécurité*. (Cité dans la page 24.)

[Ingrey *et al.*, 2007] Ingrey, A., Lerévérend, P. et Hildebrandt, A. (2007). *Safety Integrity Level IEC 61508/61511*. (Cité dans la page 24.)

[ISO/IEC 15408-1, Norme Internationale, 1999] ISO/IEC 15408-1, Norme Internationale (1999). *Common Criteria for Information Technology Security Evaluation, Part 1 : Introduction and general model*. (Cité dans les pages 25 and 47.)

[ITSEC, 1991] ITSEC (1991). *Information Technology Security Evaluation Criteria, v 1.2*. Office des publications officielles des Communautés Européennes, Luxembourg. (Cité dans la page 25.)

[Jackson, 2002] Jackson, D. (2002). Alloy : a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290. (Cité dans la page 55.)

[James *et al.*, 2014] James, P., Moller, F., Nguyen, H. N., Roggenbach, M., Schneider, S. et Treharne, H. (2014). Decomposing scheme plans to manage verification complexity. *FORMS/FORMAT*. (Cité dans la page 58.)

[Kaâniche, 1999] Kaâniche, M. (1999). *Évaluation de la sûreté de fonctionnement informatique. Fautes physiques, fautes de conception, malveillances*. Habilitation à diriger des recherches, Institut National Polytechnique de Toulouse - INPT. (Cité dans la page 22.)

[Konaté, 2009] Konaté, J. (2009). *Approche système pour la conception d'une méthodologie pour l'élicitation collaborative des exigences*. Theses, Université Paul Sabatier - Toulouse III. (Cité dans la page 22.)

[Kotonya et Sommerville, 1998] Kotonya, G. et Sommerville, I. (1998). *Requirements Engineering : Processes and Techniques*. John Wiley & Sons, Inc. (Cité dans la page 22.)

[Koudri *et al.*, 2013] Koudri, A., Guychard, C., Guerin, S., Beugnard, A., Champeau, J. et Dagnat, F. (2013). De la nécessité de fédérer des modèles dans une chaîne d'outils. *Génie logiciel*, (105):18 – 23. (Cité dans la page 35.)

[Kruchten, 1995] Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50. (Cité dans la page 62.)

[Laleau, 2002] Laleau, R. (2002). *Conception et développement formels d'applications bases de données*. Thèse de doctorat, CEDRIC (CNAM), University of Evry. Habilitation à diriger des recherches. (Cité dans la page 59.)

[Lampson, 1971] Lampson, B. (Jan. 1971). Protection. *In Proc. 5th Princeton Conf. on Information Sciences and Systems, Princeton, 1971. Reprinted in ACM Operating Systems Rev. 8, 1 (Jan. 1974)*, pages 18–24. (Cité dans la page 25.)

[Lano, 2009] Lano, K. (2009). *UML 2 Semantics and Applications*. John Wiley & Sons, Inc., New York, NY, USA. (Cité dans la page 61.)

[Lano et Haughton, 1996] Lano, K. et Haughton, H. (1996). *Specification in B : An Introduction Using the B Toolkit*. World Scientific Publishing Co., Inc., River Edge, NJ, USA. (Cité dans la page 70.)

[Lanusse *et al.*, 2014] Lanusse, A., Ferlin, A., Ayed, R. B., Sun, P., Boudi, Z., Perin, M. et Faivre, A. (2014). Livrable D2.1 du projet PERFECT. Propositions méthodologiques préliminaires pour le support à la certification à l'aide de méthodes

formelles selon une approche orientée modèles. Rapport technique. (Cité dans les pages vii, 41, 71 and 72.)

[Laprie *et al.*, 1995] Laprie, J. C., Arlat, J., Blanquart, J. P., Costes, A., Crouzet, Y., Deswarte, Y., Fabre, J. C., Guillermain, H., Kaâniche, M., Kanoun, K., Mazet, C., Powell, D., Rabéjac, C. et Thévenod, P. (1995). *Guide de la sûreté de fonctionnement*. Editions Cépaduès, Toulouse. (Cité dans les pages 22 and 23.)

[Ledang, 2002] Ledang, H. (2002). *Traduction systématique de spécifications UML en B*. Thèse de doctorat, Université de Nancy 2. (Cité dans la page 59.)

[Ledang et Dubois, 2010] Ledang, H. et Dubois, H. (2010). Proving model transformations. *In Theoretical Aspects of Software Engineering (TASE), 2010 4th IEEE International Symposium on*, pages 35–44. (Cité dans la page 57.)

[Ledang et Souquières, 2002] Ledang, H. et Souquières, J. (2002). Integration of UML and B Specification Techniques : Systematic Transformation from OCL Expressions into B. *In* Society, I. C., éditeur : *Ninth Asia Pacific Software Engineering Conference - APSEC'2002*, page 10 p, Queensland, Australia. Colloque avec actes et comité de lecture. internationale. (Cité dans la page 55.)

[Ledang *et al.*, 2003] Ledang, H., Souquières, J. et Charles, S. (2003). ArgoUML+B : un outil de transformation systématique de spécifications UML en B. *In Approches Formelles dans l'Assistance au Développement de Logiciels - AFADL'2003*, IRISA, Rennes, France. Colloque avec actes et comité de lecture. nationale. (Cité dans la page 59.)

[Ledru *et al.*, 2011] Ledru, Y., Idani, A., Milhau, J., Qamar, N., Laleau, R., Richier, J.-L. et Labiadh, M.-A. (2011). Taking into account functional models in the validation of is security policies. *In 23th International Conference on Advanced Information Systems Engineering (CAiSE'11)*, pages 592–606, London, UK. Lecture Notes in Business Information Processing, Lecture Notes in Business Information Processing. (Cité dans les pages 56, 57 and 87.)

[Leuschel et Butler, 2003] Leuschel, M. et Butler, M. (2003). ProB : A model checker for B. *In* Araki, K., Gnesi, S. et Mandrioli, D., éditeurs : *FME 2003 : Formal Methods*, LNCS 2805, pages 855–874. Springer-Verlag. (Cité dans la page 70.)

[Levendovszky *et al.*, 2014] Levendovszky, T., Rumpe, B., Schätz, B. et Sprinkle, J. (2014). Model evolution and management. *CoRR*, abs/1409.2361. (Cité dans la page 36.)

[Lodderstedt *et al.*, 2002] Lodderstedt, T., Basin, D. A. et Doser, J. (2002). SecureUML : A UML-Based Modeling Language for Model-Driven Security. *In UML*, pages 426–441. Springer-Heidelberg. (Cité dans les pages vii, 86 and 93.)

[Lonchamp, 2015] Lonchamp, J. (2015). *Analyse des besoins pour le développement logiciel : Recueil et spécification, démarches itératives et agiles.* Dunod. (Cité dans la page 55.)

[Mammar et Laleau, 2006a] Mammar, A. et Laleau, R. (2006a). A formal approach based on UML and B for the specification and development of database applications. *Automated Software Engineering*, 13(4):497–528. (Cité dans les pages 57 and 59.)

[Mammar et Laleau, 2006b] Mammar, A. et Laleau, R. (2006b). UB2SQL : A tool for building database applications using UML and B formal method. *Journal of Database Management*, 17(4):70–89. (Cité dans la page 59.)

[Mariano, 1997] Mariano, G. (1997). *Evaluation de logiciels critiques développés par la méthode B : une approche quantitative.* Thèse de doctorat. (Cité dans la page 50.)

[Marrone *et al.*, 2014] Marrone, S., Flammini, F., Mazzocca, N., Nardone, R. et Vittorini, V. (2014). Towards model-driven v&v assessment of railway control systems. *International Journal on Software Tools for Technology Transfer*, 16(6):669–683. (Cité dans la page 41.)

[Meyer, 2001] Meyer, E. (2001). *Développements formels par objets : utilisation conjointe de B et d'UML.* Thèse de doctorat, Université de Nancy 2. (Cité dans la page 59.)

[Miao *et al.*, 2002] Miao, H., Liu, L. et Li, L. (2002). Formalizing uml models with object-z. *In* George, C. et Miao, H., éditeurs : *Formal Methods and Software Engineering*, volume 2495 de *Lecture Notes in Computer Science*, pages 523–534. Springer Berlin Heidelberg. (Cité dans la page 56.)

[Milhau *et al.*, 2011] Milhau, J., Idani, A., Laleau, R., Labiadh, M. A., Ledru, Y. et Frappier, M. (2011). Combining UML, ASTD and B for the formal specification of an access control filter. *In Innovations in Systems and Software Engineering*, pages 303–313. Springer-Heidelberg. (Cité dans la page 87.)

[Minsky, 1965] Minsky, M. (1965). Matter, mind, and models. *Semantic Information Processing.* (Cité dans la page 37.)

[Mokni, 2012] Mokni, A. (2012). *Spécification et validation en B de politiques de sécurité en systèmes d'Information*. Mémoire de master, Laboratoire d'Informatique de Grenoble. (Cité dans les pages 112 and 117.)

[Moller *et al.*, 2013] Moller, F., Nguyen, H., Roggenbach, M., Schneider, S. et Treharne, H. (2013). Defining and model checking abstractions of complex railway models using csp——b. *In* Biere, A., Nahir, A. et Vos, T., éditeurs : *Hardware and Software : Verification and Testing*, volume 7857 de *Lecture Notes in Computer Science*, pages 193–208. Springer Berlin Heidelberg. (Cité dans la page 58.)

[Muchandi, 2007] Muchandi, V. (2007). Applying 4+1 view architecture with UML2. White paper. Sparx Systems. (Cité dans les pages vii and 63.)

[Nuseibeh et Easterbrook, 2000] Nuseibeh, B. et Easterbrook, S. (2000). Requirements Engineering : A Roadmap. *In Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 35–46, New York, NY, USA. ACM. (Cité dans la page 22.)

[OFTA, 1997] OFTA (1997). *Application des techniques formelles au logiciel*. Lavoisier TEC & DOC. (Cité dans les pages vii, 10, 11, 49, 50, 51, 56, 57 and 65.)

[Okalas Ossami *et al.*, 2005] Okalas Ossami, D. D., Jacquot, J.-P. et Souquières, J. (2005). Consistency in UML and B Multi-view Specifications. *In* Romijn, J., Smith, G. et van de Pol, J., éditeurs : *Integrated Formal Methods*, volume 3771 de *Lecture Notes in Computer Science*, pages 386–405. Springer Berlin Heidelberg. (Cité dans la page 59.)

[Okalas Ossami *et al.*, 2004] Okalas Ossami, D. D., Souquières, J. et Jacquot, J.-P. (2004). Opérations de construction de spécification multi-vues UML et B. *In Approches Formelles dans l'Assistance au Développement de Logiciels - AFADL'2004*, page 15 p, Besançon, France. Colloque avec actes et comité de lecture. nationale. (Cité dans la page 59.)

[Okalas Ossami *et al.*, 2006] Okalas Ossami, D. D., Souquières, J. et Jacquot, J.-P. (2006). Developing Specifications by using Operators : a Process to guarantee correctness by construction. (Cité dans la page 59.)

[OMG, 2011] OMG (2011). *OMG Unified Modeling Language (OMG UML), Superstructure - Version 2.4*. (Cité dans les pages vii and 61.)

[OMG, 2014] OMG (2014). *Object Constraint Language - Version 2.4.* (Cité dans la page 55.)

[OMG, 2015] OMG (2015). *OMG Meta Object Facility (MOF) Core Specification - Version 2.5.* (Cité dans la page 29.)

[Onisto *et al.*, 2014] Onisto, H., de Moraes Machado, T., Cravo Fernandes, R., Dantas de Medeiros Junior, J., Tamagno, I., Dezotti, T. et Bertuzzo, J. (2014). Model-driven engineering applied to the development of embedded software for b-mode ultrasound imaging systems - a case study. *In Ultrasonics Symposium (IUS), 2014 IEEE International*, pages 1261–1264. (Cité dans la page 37.)

[Peleska *et al.*, 2012] Peleska, J., Feuser, J. et Haxthausen, A. (2012). *The Model-Driven openETCS Paradigm for Secure, Safe and Certifiable Train Control System*, pages 22–52. IGI global. 2012 ; 2. (Cité dans la page 41.)

[Petit, 2003] Petit, D. (2003). *Génération automatique de composants logiciels sûrs à partir de spécifications formelles B.* Thèse de doctorat. (Cité dans la page 64.)

[Petit-Doche *et al.*, 2015] Petit-Doche, M., Breton, N., Courbis, R., Fonteneau, Y. et Güdemann, M. (2015). Formal verification of industrial critical software. *In* Núñez, M. et Güdemann, M., éditeurs : *Formal Methods for Industrial Critical Systems*, volume 9128 de *Lecture Notes in Computer Science*, pages 1–11. Springer International Publishing. (Cité dans la page 101.)

[Petit-Doche et Güdemann, 2013] Petit-Doche, M. et Güdemann, M. (Juin 2013). OpenETCS process. Definition of the overall process for the formal description of ETCS and the rail system it works in. Rapport technique. (Cité dans la page 40.)

[Potet, 2002] Potet, M.-L. (2002). *Spécifications et développements formels : Étude des aspects compositionnels dans la méthode B.* Habilitation à diriger des recherches, Institut National Polytechnique de Grenoble - INPG. (Cité dans la page 68.)

[Qiu *et al.*, 2014a] Qiu, S., Kaza, G.-L., Sallak, M. et Schön, W. (2014a). A reachability analysis for verification of safety properties of railway infrastructures. *In 10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems Syposium, Tool Exhibition and Tutorium, FORMS/FORMAT 2014*, Braunschweig, Germany. (Cité dans la page 72.)

[Qiu *et al.*, 2014b] Qiu, S., Sallak, M., Schön, W. et Cherfi-Boulanger, Z. (2014b).

Availability assessment of railway signalling systems with uncertainty analysis using statecharts. *Simulation Modelling Practice and Theory*, 47:1 – 18. (Cité dans la page 72.)

[RFF, 2012] RFF (2012). Principes et règles d'exploitation du système ETCS - Particularités en cas de superposition à un autre système de signalisation. Rapport technique, Réseau ferré de France. (Cité dans les pages vi, viii, 8, 10, 78, 79, 89, 109, 110, 157 and 158.)

[Roe *et al.*, 2002] Roe, D., Broda, K. et Russo, A. (2002). Mapping UML Models incorporating OCL Constraints into Object-Z. *In Imperial College Workshops*. (Cité dans la page 55.)

[Sadoun, 2014] Sadoun, D. (2014). *From natural language specifications to formal specifications via an ontology as a pivot model*. Theses, Université Paris Sud - Paris XI. (Cité dans la page 48.)

[Sallak et Schön, 2013] Sallak, M. et Schön, W. (2013). Livrable D1.2 du projet PERFECT. État de l'art concernant les méthodes de conception sûres en vue d'une certification. Rapport technique. (Cité dans la page 23.)

[Sandhu *et al.*, 2000] Sandhu, R., Ferraiolo, D. et Kuhn, R. (2000). The NIST Model for Role-based Access Control : Towards a Unified Standard. *In Proceedings of the Fifth ACM Workshop on Role-based Access Control*, pages 47–63. (Cité dans les pages 26 and 83.)

[Sannier, 2013] Sannier, N. (2013). *INCREMENT an Hybrid approach for modeling and analyzing regulatory safety requirements in the large*. Theses, Université Rennes 1. (Cité dans la page 37.)

[Sannier et Baudry, 2012] Sannier, N. et Baudry, B. (2012). Toward multilevel textual requirements traceability using model-driven engineering and information retrieval. *In Proceedings of the Model-Driven Requirements Engineering workshop (MoDRE'12) at RE'12*, pages –, Chicago, USA. (Cité dans la page 37.)

[Schön *et al.*, 2014] Schön, W., Larraufie, G., Moëns, G. et Poré, J. (2014). *Signalisation et automatismes ferroviaires Tome 3*. La vie du rail. (Cité dans la page 7.)

[Scippacercola *et al.*, 2015] Scippacercola, F., Pietrantuono, R., Russo, S. et Zentai, A. (2015). Model-driven engineering of a railway interlocking system. *In Procee-*

dings of the 3rd International Conference on Model-Driven Engineering and Software Development, pages 509–519. (Cité dans les pages vii, 38 and 39.)

[Snook et Butler, 2004] Snook, C. et Butler, M. (2004). U2b - a tool for translating uml-b models into b. *In* Mermet, J., éditeur : *UML-B Specification for Proven Embedded Systems Design*, numéro DSSE-TR-2003-3. Springer. Chapter : 6. (Cité dans la page 59.)

[Snook et Butler, 2006] Snook, C. et Butler, M. (2006). Uml-b : Formal modeling and design aided by uml. *ACM Trans. Softw. Eng. Methodol.*, 15(1):92–122. (Cité dans la page 59.)

[Sprinkle *et al.*, 2014] Sprinkle, J., Rumpe, B., Vangheluwe, H. et Karsai, G. (2014). Metamodelling : State of the art and research challenges. *CoRR*, abs/1409.2359. (Cité dans les pages 29, 30 and 36.)

[Sun, 2015] Sun, P. (2015). *Model based system engineering for safety of railway critical systems*. Thèse de doctorat, École centrale de Lille. (Cité dans les pages 9, 58 and 72.)

[Svendsen *et al.*, 2012] Svendsen, A., Haugen, Ø. et Møller-Pedersen, B. (2012). Synthesizing software models : Generating train station models automatically. *In* Ober, I. et Ober, I., éditeurs : *SDL 2011 : Integrating System and Software Modeling*, volume 7083 de *Lecture Notes in Computer Science*, pages 38–53. Springer Berlin Heidelberg. (Cité dans la page 57.)

[Truong et Souquières, 2004] Truong, N. T. et Souquières, J. (2004). Validation des propriétés d'un scénario UML/OCL à partir de sa dérivation en B. *In Approches Formelles dans l'Assitance au Développement de Logiciels - AFADL'2004*, page 15 p, Besançon, France. Colloque avec actes et comité de lecture. nationale. (Cité dans les pages 55 and 59.)

[Union Internationale des Chemins de fer, 2000] Union Internationale des Chemins de fer (2000). L'interopérabilité du système ferroviaire. Rapport technique. (Cité dans la page 8.)

[UNISIG, 2006] UNISIG (2006). System Requirements Specification, UNISIG SUBSET-026. Rapport technique, European Railway Agency. Version 2.3.0. (Cité dans les pages 10, 78 and 89.)

[Vermesan et Coenen, 1999] Vermesan, A. et Coenen, F. (1999). *Validation and verification of Knowledge Based Systems : Theory, Tools and Practice*. Kluwer Academic

Publishers, Boston. (Cité dans les pages 51, 52 and 53.)

[Vu *et al.*, 2014] Vu, L., Haxthausen, A. et Peleska, J. (2014). *A Domain-Specific Language for Railway Interlocking Systems*, pages 200–209. Technische Universität Braunschweig. (Cité dans la page 42.)

Annexe A

Définitions formelles des différentes variantes du modèle RBAC [Ferraiolo *et al.*, 2001]

Nous présentons les différentes variantes du modèle RBAC dont les définitions formelles sont extraites des travaux de [Ferraiolo *et al.*, 2001]. Chaque variante du modèle RBAC comprend des ensembles des éléments de base, des relations et des fonctions entre ces ensembles.

A.1 Définition formelle du noyau RBAC

Le noyau RBAC définit une collection minimale des éléments de RBAC, des ensembles d'éléments de base, des relations et des fonctions afin d'atteindre un système complet de RBAC (voir la figure A.1) :

- USERS, ROLES, OPS et OBS, respectivement ensembles des utilisateurs, rôles, opérations et objets.
- UA $\subseteq$ USERS $\times$ ROLES, relation associant rôle à utilisateur.
- $assigned\_users : (r : ROLES) \rightarrow 2^{USERS}$, fonction associant un rôle r à un ensemble d'utilisateurs.
  Formellement : $assigned\_users(r) = \{u \in USERS | (u, r) \in UA\}$.
- PRMS $= 2^{(OPS \times OBS)}$, ensemble des permissions.
- PA $\subseteq$ PRMS $\times$ ROLES, relation associant permission à rôle.
- $assigned\_permissions : (r : ROLES) \rightarrow 2^{PRMS}$, fonction associant un rôle r à un ensemble de permissions.
  Formellement : $assigned\_permissions(r) = \{p \in PRMS | (p, r) \in PA\}$.
- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$, fonction associant une permission p à l'ensemble des opérations associées.
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$, fonction associant une permission p à l'ensemble des objets associés.
- SESSIONS, ensemble des sessions.

- $user\_sessions(u : USERS) \rightarrow 2^{SESSIONS}$, fonction associant un utilisateur u à un ensemble de sessions.
- $session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}$, fonction associant une session s à un ensemble de rôles.
  Formellement : $session\_roles(s_i) \subseteq \{r \in ROLES | (session\_users(s_i), r) \in UA\}$.
- $avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS}$, les permissions disponibles pour un utilisateur dans une session, $\bigcup_{r \in session\_roles(s)} assigned\_permissions(r)$.

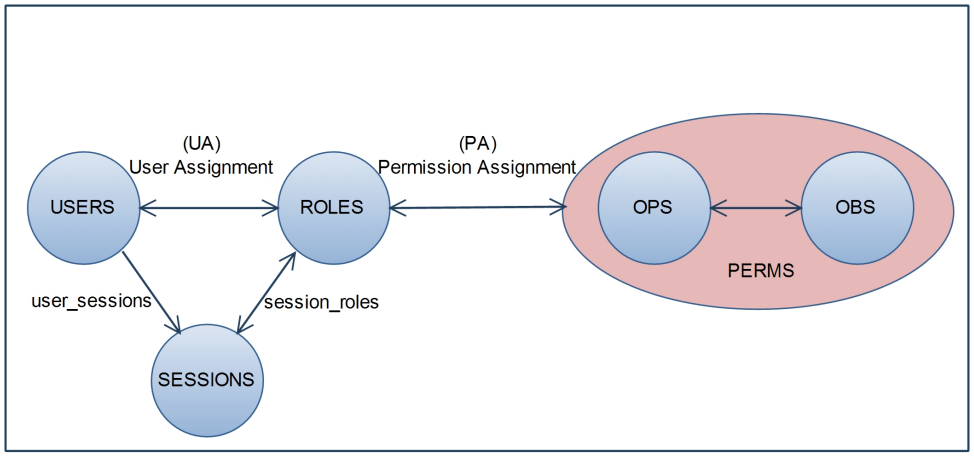

Figure A.1 – Le noyau RBAC

A.2 Définition formelle de RBAC hiérarchique

La deuxième variante, appelée RBAC hiérarchique (Hierarchical RBAC), introduit la notion d'héritage de rôle (*Role Hierarchy*). L'héritage des rôles est un moyen pour la structuration des rôles dans le but de mettre en œuvre les autorités et les responsabilités d'une organisation (voir la figure A.2). L'introduction de cette notion permet ainsi de simplifier la définition des rôles et de faciliter l'héritage des permissions assignées aux rôles.

La relation d'héritage est décrite en termes de permissions r_1 hérite du rôle r_2 si toutes les privilèges de r_2 sont aussi des privilèges de r_1. Il existe deux types de hiérarchie de rôles : la hiérarchie générale de rôles et la hiérarchie limitée de rôles.

La hiérarchie générale de rôles

La hiérarchie générale de rôles inclut le concept d'héritage multiple des permissions et des utilisateurs.

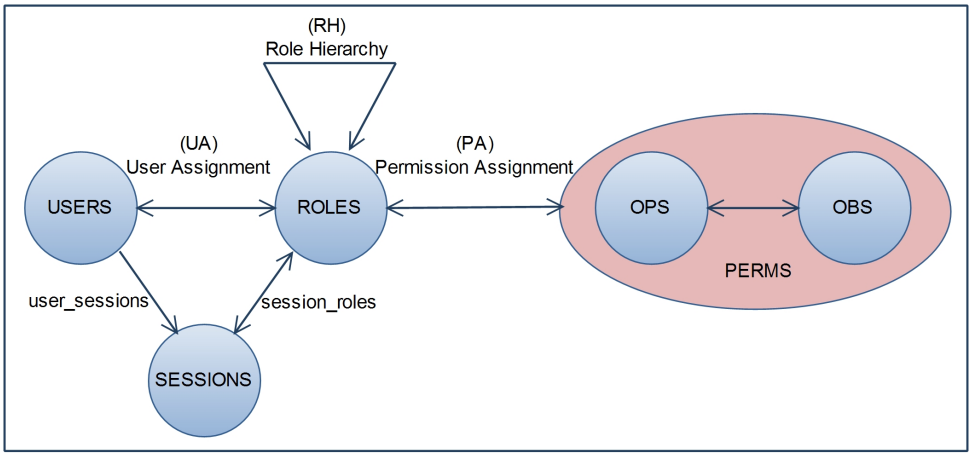

Figure A.2 – Le modèle RBAC hiérarchique

- RH $\subseteq$ ROLES $\times$ ROLES, est un ordre partiel sur les rôles ROLES appelé relation d'héritage, écrit à l'aide du symbole $\succeq$ tel que $r_1 \succeq r_2$ seulement si toutes les permissions de r_2 sont aussi des permissions de r_1 et toutes les utilisateurs de r_1 sont aussi des utilisateurs de r_2. Formellement : $r_1 \succeq r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1) \wedge authorized\_users(r_1) \subseteq authorized\_users(r_2)$.
- $authorized\_users(r : ROLES) \rightarrow 2^{USERS}$, fonction associant un rôle r à un ensemble d'utilisateurs en présence de la hiérarchie de rôles. Formellement : $authorized\_users(r) = \{u \in USERS | r' \succeq r, (u, r') \in UA\}$.
- $authorized\_permissions(r : ROLES) \rightarrow 2^{PRMS}$, fonction associant un rôle r à un ensemble de permissions en présence de la hiérarchie de rôles. Formellement : $authorized\_permissions(r) = \{p \in PRMS | r' \succeq r, (p, r') \in PA\}$.

La hiérarchie limitée de rôles

La hiérarchie limitée de rôles impose des restrictions telles que la structure de l'arbre hiérarchique.

- $\forall r, r_1, r_2 \in ROLES, r \succeq r_1 \wedge r \succeq r_2 \Rightarrow r_1 = r_2$.

A.3 Définition formelle de RBAC avec séparation des droits statiquement

Le modèle RBAC avec séparation des droits statiquement introduit les relations permettant de gérer les conflits entre les rôles(voir la figure A.3). La séparation statique des droits est spécifiée par un ensemble de rôles rs, un sous-ensemble de rôles t de rs et un entier n tel que $2 \leq n \leq card(rs)$. Ce type de contrainte spécifie qu'aucun utilisateur ne peut être affecté à n rôles ou plus de rs :

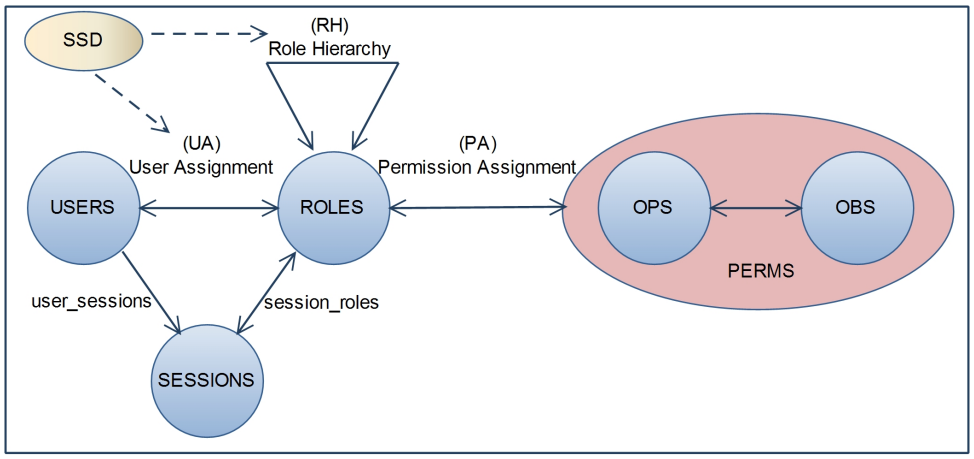

Figure A.3 – Le modèle RBAC avec séparation des droits statiquement

- $SSD \subseteq (2^{ROLES} \times \mathbb{N})$.
- $\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} assigned\_users(r) = \emptyset$.
- En présence de hiérarchie de rôles :
  $\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} authorized\_users(r) = \emptyset$.

A.4 Définition formelle de RBAC avec séparation des droits dynamiquement

Le modèle RBAC avec séparation des droits dynamiquement introduit les relations permettant de gérer les contraintes sur les rôles actifs pendant une session (voir la figure A.4). La séparation dynamique des droits est spécifiée par un ensemble de rôles rs et un entier n tel que $2 \leq n \leq card(rs)$. Ce type de contrainte spécifie qu'aucun utilisateur ne peut être activé par n rôles ou plus de rs :

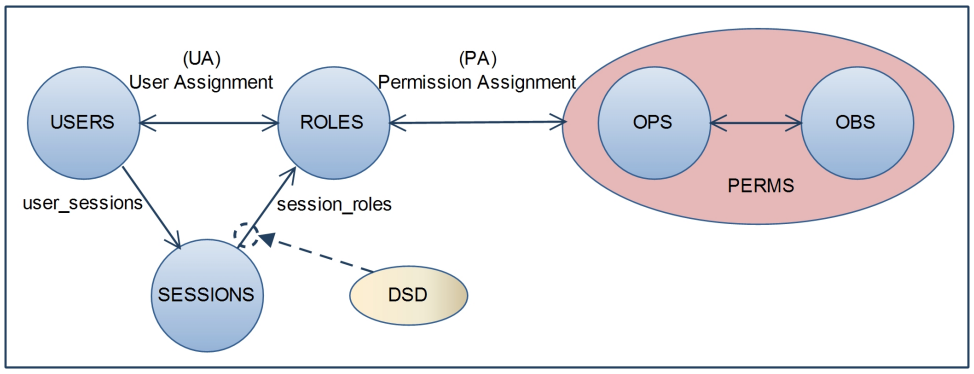

Figure A.4 – Le modèle RBAC avec séparation des droits dynamiquement

- $DSD \subseteq (2^{ROLES} \times \mathbb{N})$.
- $\forall rs \in 2^{ROLES}, n \in \mathbb{N}, (rs, n) \in DSD \Rightarrow n \geq 2 \wedge |rs| \geq n$ et
  $\forall s \in SESSIONS, \forall rs \in 2^{ROLES}, \forall role\_subset \in 2^{ROLES}, \forall n \in \mathbb{N}, (rs, n) \in DSD, role\_subset \subseteq rs, role\_subset \subseteq session\_roles(s) \Rightarrow |role\_subset| < n$.

Annexe B

Extraits des règles d'exploitation du document [RFF, 2012]

Article 103. Fonctionnement

La ligne est découpée en cantons qui ne sont pas matérialisés sur le terrain.

103.1. Début de mission

Au début d'une mission, après connexion par GSM-R[a]*, le train peut être localisé "en sécurité" : sa position est connue du RBC*[b]*. Pour permettre au train de se déplacer, le RBC autorise le mouvement en OS*[c] *(allocation d'une MA en OS). Le conducteur s'avance en marche à vue. Dès le passage d'un PI ETCS*[d] *lié à un joint de circuit de voie de cantonnement, le train envoie sa position au RBC qui, si toutes les conditions sont réunies, lui alloue une MA en FS*[e]*. Le train circule alors normalement. Grâce aux capteurs odométriques, le train calcule sa position sur la ligne et envoie régulièrement celle-ci au RBC. Le RBC confirme alors la position au train s'il n'y a pas d'erreur. La lecture des PI par le train lui permet de recaler régulièrement sa position.*

Si, pour diverses raisons, le train ne peut pas être localisé en sécurité en début de mission, le RBC ne peut pas allouer de MA au train qui part alors en mode SR[f]*, après avoir reçu de l'agent circulation l'ordre écrit correspondant, jusqu'au passage d'un PI, qui permet alors au train de se localiser.*

---

a. Global System for Mobile Communications - Railways. (Système de communication téléphonique pour mobile dédié aux chemins de fer).

b. Radio Block Centre. Système centralisé fonctionnant avec les enclenchements afin d'établir et de contrôler l'espacement et le mouvement des trains en envoyant et recevant des informations par radio.

c. On Sight. (À vue). Mode qui autorise le conducteur à s'avancer en marche à vue.

d. Point d'information ETCS. Constitué d'une ou plusieurs balises transmettant des informations vers le bord.

e. Full Supervision. Mode technique dans lequel le train est supervisé en vitesse et déplacement. Marche normale attribuée au train.

f. Staff Responsible. Mode de déplacement utilisé dans les situations dégradées. Il est utilisable sous la responsabilité respective de l'agent-circulation et du conducteur.

103.2. Protection contre le rattrapage

En FS, une vitesse maximale seule affichée signifie voie libre. La MA constituée d'un ou plusieurs cantons est allouée canton par canton, par ajout d'un canton. Si le canton en aval du dernier canton alloué est occupé, le train doit être en mesure de s'arrêter avant l'entrée de ce canton. Le conducteur est alors avisé par un double bip d'attention que sa MA n'est pas renouvelée : le DMI[a] affiche une vitesse but égale à 0 et une distance but égale à la distance le séparant de l'EOA, distant, en principe de 200 m de l'entrée du canton occupé. Le conducteur doit ralentir de façon à être en mesure de s'arrêter avant l'EOA, tout en respectant la vitesse autorisée (voir figure B.1).

Si le canton qui était occupé se libère, la MA est prolongée et l'affichage au DMI se modifie en conséquence.

a. Driver machine interface. Dispositif ETCS/GSM-R qui permet la communication entre le système bord et le conducteur.

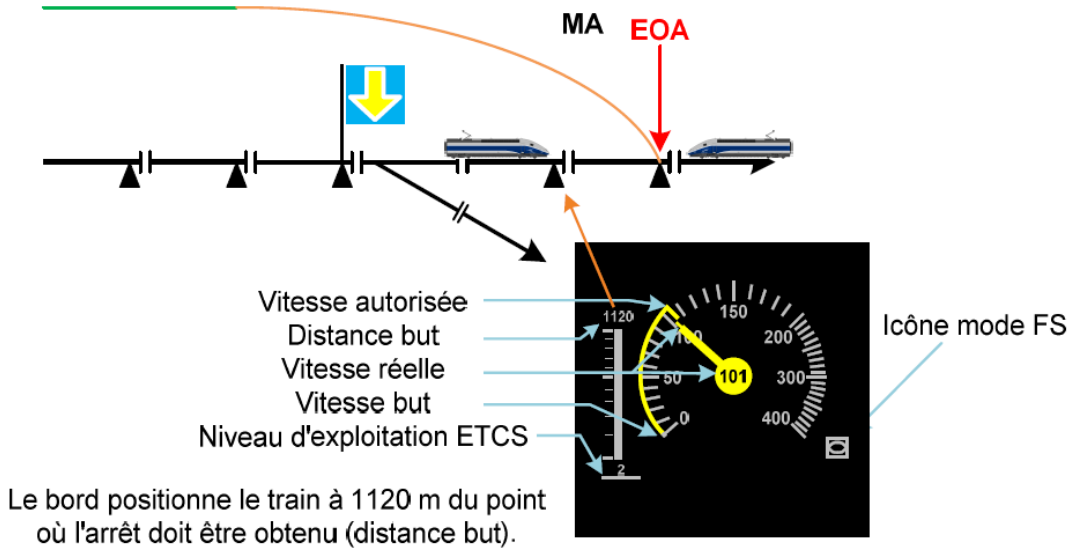

Figure B.1 – MA et ses caractéristiques [RFF, 2012]

Modélisation UML/B pour la validation des exigences de sécurité des règles d'exploitation ferroviaires

Résumé : La sécurité est un enjeu majeur dans le cycle de développement des systèmes critiques, notamment dans le secteur du transport ferroviaire. Cette thèse vise la modélisation, la vérification et la validation des règles d'exploitation ferroviaires au regard des exigences de sécurité. Ces règles ont pour but de définir les autorisations de déplacement des trains sur des lignes ferroviaires nationales équipées du système européen de gestion du trafic ferroviaire (ERTMS). De manière analogue, on trouve les concepts liés aux autorisations dans la description des politiques de contrôle d'accès des systèmes d'information. Par conséquent, nos contributions portent sur l'adaptation d'une approche UML/B pour le contrôle d'accès des systèmes d'information afin de modéliser et de valider les règles d'exploitation ferroviaires. Dans un premier temps, nous avons adapté le modèle Role Based Access Control (RBAC) sur une étude de cas ferroviaire extraite des règles d'exploitation appliquées sur la ligne à grande vitesse LGV Est-Européenne en France. La plate-forme B4MSecure nous a permis de modéliser ces règles à l'aide d'un profil UML de RBAC inspiré de SecureUML. Ensuite, ces modèles sont transformés en des spécifications B qui ont été enrichies par des propriétés de sécurité ferroviaire et soumises à des activités de vérification et de validation formelles. Aux concepts du modèle RBAC, le modèle Organization Based Access Control (Or-Bac) introduit la notion d'organisation, au centre de ce modèle, et la notion de contexte. Nous avons donc proposé d'utiliser ce modèle en tant qu'extension du modèle RBAC dans l'optique d'une interopérabilité ferroviaire en ERTMS.
Mots clés : règles d'exploitation ferroviaires, sécurité ferroviaire, modélisation UML/B, vérification formelle, validation formelle, modèle RBAC, modèle Or-BAC.

UML/B modeling for the safety requirements validation of railway operating rules

Abstract : The safety is a major issue in the development cycle of the critical systems, in particular in the rail transportation sector. This thesis aims at the modeling, the verification and at the validation of the railway operating rules with regard to the safety requirements. These rules intend to define the authorizations of trains movement on national railway lines equipped with the European Rail Traffic Management System (ERTMS). In a similar way, we find the concepts of authorizations in the description of access control policies of information systems. Consequently, our contributions concern the adaptation of an UML/B approach for the access control of information systems to model and validate the railway operating rules. At first, we adapted the Role Based Access Control (RBAC) model on a railway case study extracted from the operating rules applied on the LGV-Est-Européenne line in France. The B4MSecure platform enables the modeling of these rules by means of a UML profile of RBAC inspired by SecureUML. Then, these models are transformed into B specifications. which are enhanced by railway safety properties and formally verified and validated. In addition to the concepts of the RBAC model, the Organization Based Access Control (Or-Bac) model introduces the notion of organization, in the center of this model, and the notion of context. We have therefore proposed to use this model as extension of the RBAC model in the context of railway interoperability in ERTMS.
Keywords : railway operating rules, railway safety, UML/B modeling, formal verification, formal validation, RBAC model, Or-BAC model.