Table des matières

Table des figures

Introduction générale

Préambule

Ce mémoire présente une synthèse des travaux de recherche réalisés dans le cadre de mon doctorat au sein de l'unité de recherche ESTAS [1] de l'INRETS [2] et de l'équipe SED [3] du LAGIS [4]. Ces travaux ont été rendus possibles grâce au co-financement de la région Nord-Pas-de-Calais et de l'INRETS. En effet, ils s'intègrent dans le pôle de compétitivité I-Trans porté par la région Nord-Pas-de-Calais et relatif aux innovations dans le domaine du transport ferroviaire. De plus, ces travaux s'inscrivent dans un contexte plus général puisqu'il s'inscrit au niveau national dans le projet SAFECODE et au niveau européen dans des projets tels que SELCAT ou PANSAFER.

Motivation

Ce sujet de thèse se situe dans un contexte général de sécurité dans les transports ferroviaires et plus particulièrement autour de l'utilisation de méthodes discrètes et formelles visant à évaluer et valider certaines exigences de sécurité. Il s'inscrit également dans une démarche de spécification et de conception appliquée à des systèmes présentant une forte composante dynamique. En effet, l'augmentation de la complexité des systèmes automatisés induit une difficulté croissante pour l'évaluation de leurs propriétés structurelles et comportementales. Nous nous intéressons à une grande partie du

1. Évaluation des Systèmes de Transport Automatisés et de leur Sécurité ;
2. Institut National de Recherche sur les Transports et leur Sécurité ;
3. Systèmes à Événements Discrets ;
4. Laboratoire d'Automatique, Génie Informatique et Signal.

processus de conception, depuis la formalisation des exigences jusqu'aux activités d'intégration, de validation et de vérification.

Nos objectifs peuvent être caractérisés par les éléments suivants :

- Un domaine applicatif privilégié : l'exploitation sûre et performante des réseaux de transports guidés,
- Des fondements scientifiques et techniques : les modèles et les méthodes formelles associés, prenant en compte le temps, les aspects de criticité et de synchronisation,
- Un contexte méthodologique : le développement de méthodes et d'outils permettant d'assister la conception de systèmes complexes.

Problématique : Conception de systèmes critiques

Un système est considéré comme *critique* lorsque une défaillance est susceptible d'entraîner des conséquences dramatiques ou inacceptables, telles que des pertes humaines, des blessés graves, des dommages matériels importants, des pertes économiques ou financières, ou des conséquences graves pour l'environnement. De tels systèmes existent notamment dans les domaines économique, médical, énergétique ou le transport. Le contexte de notre travail est celui de la sécurité des transports ferroviaires.

Par conséquent, la conception de systèmes critiques doit permettre de garantir l'exigence qualité du « zéro défaut », dès la mise en service du système. Se pose alors la question de la certification du développement de tels systèmes. Plusieurs domaines de compétences sont nécessaires pour répondre à cette question : d'abord, des experts de la sûreté de fonctionnement donnent une vision globale du système et en évaluent les risques. Ainsi ils établissent des règles fixant un cadre formel de conception.

Ensuite, un processus de développement respectant ce cadre est nécessaire. De plus, ce processus s'intègre dans un contexte législatif et normatif. Ainsi, depuis 1999, les standards européens de sûreté sont définis par la norme IEC 61508, qui peut soit être directement utilisée, soit servir de base pour le développement de standards spécifiques aux différents secteurs d'activités ; ce fut notamment le cas pour le secteur ferroviaire. Plus précisément, les standards européens pour le développement et l'implémentation des certifications

ferroviaires ont été publiés dans les normes CENELEC 50126, 50128 et 50129.

Nous nous sommes intéressés à la norme EN 50128, qui traite plus particulièrement de la conception de système sûrs, et qui, d'abord met en avant la nécessité de gérer les exigences et leur traçabilité, et ensuite recommande l'utilisation de méthodes formelles.

Ainsi dans un premier temps, nous nous situons dans le domaine général de l'ingénierie système, qui regroupe l'ensemble des activités visant à concevoir, faire évoluer et vérifier un système complexe. Au sein de ce domaine, l'ingénierie des exigences a pour objectif de vérifier si une exigence est bien respectée à travers son application. La phase de conception se base donc sur la notion d'exigence, définie comme l'expression d'une caractéristique à laquelle doit impérativement répondre un système. Généralement elle provient d'une nécessité ou d'une demande d'un utilisateur. La gestion des exigences consiste à conserver l'historique des liens tout au long du développement et de la vie du système. Elle inclut la traçabilité des exigences qui consiste à gérer la correspondance entre les résultats obtenus et les exigences. Ainsi, il nous revient de clairement définir les exigences imposées à l'application avant de la concevoir, et de les suivre tout au long du cycle de conception.

De plus, le facteur humain et la complexité croissante des systèmes apparaissent comme deux éléments critiques. Ils sont à l'origine de la nécessité de recourir aux mathématiques pour supprimer les ambiguïtés. Les méthodes formelles visent à fournir un cadre précis et strict permettant de décrire les systèmes à élaborer.

Dans un second temps, la norme recommande donc les méthodes formelles pour la spécification et la conception de systèmes critiques. Elles sont déjà utilisées dans un cadre industriel, comme par exemple la méthode *B* lors de la conception de la ligne METEOR du métro parisien ; toutefois, les méthodes formelles manipulent des concepts parfois difficiles à appréhender, ce qui peut être un frein à leur développement. Aussi, il est pertinent de les coupler avec des modèles graphiques plus accessibles et répandus, afin de permettre un meilleur échange et une meilleure compréhension.

Contribution

Face à ces défis, notre objectif est de construire une méthodologie originale contribuant à un processus d'automatisation sûr, intégrant au mieux les exigences des utilisateurs en utilisant un ensemble de modèles et de méthodes existants.

Nous nous intéressons plus particulièrement à la modélisation et à la vérification formelle des systèmes temporisés. La généralisation des systèmes offrant une qualité de service garantie (temps de réponse, taux de pertes. . .) nécessite des modèles explicitant les contraintes temporelles. La criticité de ces systèmes rend impérative leur certification, tant au niveau qualitatif (comportement du système) que quantitatif (cohérence temporelle).

Tout d'abord, nous nous sommes attachés à trouver un modèle pertinent pour la spécification de ces contraintes temporelles. Ensuite, nous avons construit une méthodologie pour nous permettre d'analyser le modèle proposé et visant à garantir un bon comportement du système même en cas de défaillance technique.

Pour ce faire, nous avons choisi dans un premier temps de modéliser le système étudié sous forme de réseau de Petri, outil graphique permettant de valider et de vérifier notre modèle par rapport aux exigences. Cet outil dispose de méthodes d'analyse pour permettre la vérification de systèmes complexes. Nous décrivons dans ce mémoire différentes méthodes permettant cette analyse avant d'en proposer quelques autres originales permettant de répondre au contexte de notre travail.

Tout au long du processus de conception que nous proposons, nous distinguons différents types de modèles en fonction des objectifs que nous voulons atteindre. Dans ce cadre, nous justifions l'utilisation de diverses extensions temporelles des réseaux de Petri en fonction du contexte de modélisation ; ainsi, nous construisons un modèle prescriptif des exigences basé sur l'outil réseau de Petri p-temporel. Ensuite, notre méthode demande la construction d'un modèle du procédé, correspondant au modèle précédent, et décrivant une solution proposée pour la mise en œuvre du système. Ce modèle constructif doit permettre de mettre en évidence les propriétés souhaitées ; nous utilisons les réseaux de Petri t-temporels pour répondre à cet objectif. Nous proposons également une méthode permettant de vérifier que ce dernier modèle est conforme au modèle des exigences.

Enfin, nous envisageons de transformer ce réseau de Petri en machines abstraites B pour permettre de poursuivre un processus B classique se basant sur le raffinement et la preuve afin d'obtenir un système certifiable. Il s'agit d'aboutir à des règles de transformation garantissant la traçabilité des exigences lors du passage d'un modèle source réseau de Petri à un modèle cible machine B dans le cadre d'une ingénierie dirigée par les modèles.

Structure du mémoire

Par la suite, le mémoire s'organise de la manière suivante :

Une première partie composée des trois premiers chapitres propose un état de l'art concernant le sujet traité. Elle vise à définir le cadre de ces travaux et présente différentes méthodes et différents outils qui sont utilisés pour bâtir la méthodologie que nous présentons. Le chapitre 1 décrit le contexte général de ce travail de thèse, plus précisément celui de la conception de systèmes ferroviaires critiques. Dans un premier temps, il décrit la modélisation des systèmes complexes dans le cadre de l'ingénierie système. Nous justifions l'utilisation des systèmes à événements discrets pour la modélisation de systèmes de transport. Nous nous concentrons par la suite sur la description des systèmes critiques, soumis à de très fortes contraintes de sûreté de fonctionnement. Plus particulièrement, nous nous concentrons sur les contraintes temporelles contenues dans le cahiers des charges. Dans un second temps, nous décrivons le contexte ferroviaire. Après une introduction générale au transport ferroviaire, nous présentons l'influence de la politique de transport ferroviaire européenne en terme de sécurité et d'interopérabilité, ce qui demande d'importants besoins de certification.

Le chapitre 2 s'intéresse aux méthodes utilisées pour garantir la sécurité des systèmes. D'abord, l'ingénierie des exigences nous donne les outils pour extraire, analyser et tracer les exigences tout au long du processus de conception. Ensuite, les méthodes formelles sont présentées, ainsi que leur utilisation pour la validation et la vérification des systèmes critiques. Une présentation succincte de l'ingénierie dirigée par les modèles vient conclure ce deuxième chapitre.

Le chapitre 3 présente les outils que nous utilisons pour un processus de conception visant à tracer et vérifier les exigences contenues dans le cahier

des charges, en particulier les exigences temporelles. Plus précisément, notre choix s'est porté sur les extensions temporelles de l'outil réseau de Petri. Nous décrivons leurs propriétés et les différentes manières disponibles pour leur analyse. En fin de chapitre, la méthode *B* est présentée.

La seconde partie de ce mémoire contient notre contribution pour la conception de systèmes critiques soumis à des contraintes temporelles. Le chapitre 4 décrit notre proposition méthodologique visant à fournir une aide à la certification pour l'évaluation des systèmes discrets temporisés. Cette dernière se base sur l'utilisation et l'analyse de deux extensions temporelles de l'outil réseau de Petri. Le chapitre 5 propose d'appliquer et de valider notre approche au moyen d'un cas d'étude ferroviaire : le passage à niveau, composant critique du système ferroviaire. Enfin, le chapitre 6 s'intéresse à l'implantation de notre méthode. Dans un premier temps, une proposition de transformation de modèle est présentée. Ensuite, nous évoquons en perspective l'utilisation de l'ingénierie dirigée par les modèles pour une transformation plus formelle.

Ce mémoire se termine par une conclusion générale qui rappelle les principaux points développés dans ce mémoire et envisage plusieurs perspectives de recherche.

Chapitre 1

Conception de systèmes ferroviaires

Sommaire

L'objectif de ce chapitre est de présenter et de situer le cadre de notre travail. Nous souhaitons modéliser le comportement de systèmes critiques en utilisant différents langages formels de spécification qui nous permettent de vérifier les propriétés attendues, plus particulièrement en termes de sécurité et de dynamique.

Dans cette optique, le processus de conception de systèmes vise à construire un ensemble de modèles décrivant une solution répondant aux exigences, à la fois imposées par le cahier des charges, explicitant les besoins du client ou du demandeur, mais aussi devant respecter les contraintes normatives et législatives associées au cadre d'utilisation du système. Ce chapitre contient les éléments nécessaires permettant d'établir un cadre clair pour le travail de thèse développé dans ce mémoire. Il débute par une réflexion sur la modélisation des systèmes complexes justifiant la nécessité de construire plusieurs modèles complémentaires en fonction de l'objectif que l'on souhaite atteindre. Nous justifions de plus l'utilisation des systèmes à événements discrets pour la modélisation des systèmes de transport complexes. Ensuite nous présentons la classe de système que nous étudions plus particulièrement : les systèmes soumis à des contraintes temporelles. Enfin, les caractéristiques inhérentes au contexte général de la sécurité des transports ferroviaires sont mises en lumière afin de justifier des méthodes que nous avons appliquées tout au long de notre travail.

Notre problématique se situe donc à l'intersection de deux grands domaines : celui de la modélisation et de l'analyse des systèmes complexes, et celui de l'étude de la sécurité des systèmes critiques ferroviaires. Ce cadre étant posé, notre objectif n'est pas de décrire de manière exhaustive les méthodes utilisées dans chacun des deux domaines mais de proposer des méthodes de modélisation pertinentes pour la conception de système ferroviaires critiques.

1.1 Modélisation de systèmes complexes

Cette section décrit le contexte académique de notre travail : il s'agit de la modélisation et de l'analyse des systèmes complexes. Nous introduisons également les enjeux et les concepts de l'ingénierie système, et comment ils s'intègrent dans un cycle global de conception. Ensuite, nous décrivons de

manière synthétique les différents types de modèles permettant de décrire et d'analyser un système complexe, ainsi que les différentes propriétés qu'il doit posséder, exprimées notamment dans son cahier des charges. Plus précisément, nous nous focalisons sur l'étude des systèmes à événements discrets soumis à des contraintes temporelles.

1.1.1 Pourquoi modéliser ?

La modélisation est indispensable pour la compréhension et l'analyse des systèmes industriels et constitue une base du processus de conception. Elle regroupe un ensemble de techniques permettant de disposer d'une représentation mathématique du système à étudier. Un modèle est une approximation, une vue plus ou moins abstraite de la réalité afin de l'appréhender plus simplement, selon un point de vue établi pour un objectif donné.

La notion de complexité permet de qualifier des systèmes qui, par certains de leurs comportements, par leur taille et par la diversité de nature des phénomènes mis en jeu, présentent des difficultés d'analyse. Ainsi, un système complexe ne peut pas être réduit à un modèle unique. Un modèle ne rend compte que d'une partie de la connaissance du système étudié et en masque une autre partie, considérée superflue ou inutile à l'objectif désiré. Il n'est qu'une représentation d'un système dans son environnement d'utilisation.

Plus qu'un modèle unique, c'est donc un ensemble de modèles complémentaires qui doit être élaboré en fonction des usages et des objectifs des différents acteurs d'un projet.

De plus, il en va de même pour le choix des outils utilisés pour représenter ces différents modèles ; le contexte d'application devra ainsi être clairement défini afin de choisir de façon pertinente des outils spécifiques et adaptés. Entre autres, les outils de modélisation peuvent être textuels ou graphiques, fonctionnels ou analytiques, informels ou formels.

1.1.2 Ingénierie système

Tout d'abord, il semble nécessaire de clarifier la notion de *système* utilisée dans ce mémoire. Il existe en effet plusieurs définitions de ce terme en fonction du domaine étudié ou du point de vue adopté. Nous reprenons la définition 1.1 énoncée dans [Meinadier98].

Définition 1.1 (Système)
Un système est un ensemble composite de personnels, de matériels et de logiciels organisés pour que leur interfonctionnement permette, dans un environnement donné, de remplir les missions pour lesquels il a été conçu.

De nombreuses difficultés se présentent lors de la construction de systèmes devenant de plus en plus complexes : des besoins mal exprimés ou mal perçus, des spécifications imprécises ou incomplètes, des solutions non justifiées ou non validées, une communication entre acteurs non maîtrisée. Pour tenter de résoudre ces problèmes, l'ingénierie système (définition 1.2) a été développée afin de répondre aux besoins de compréhension et de maîtrise de ces systèmes. Elle s'applique dans divers domaines industriels : aérospatial, télécommunications, systèmes d'informations, transports, etc.

Définition 1.2 (Ingénierie Système [IEE99])
L'ingénierie système est une démarche méthodologique coopérative et interdisciplinaire englobant l'ensemble des activités pour concevoir, développer, faire évoluer et vérifier un système et apportant une solution optimisée pour le système équilibrée sur l'ensemble de son cycle de vie, satisfaisant aux attentes d'un client et acceptables par tous.

Les méthodes de l'ingénierie système reposent sur des approches de modélisation, de vérification et de simulation pour valider les exigences et évaluer les solutions tout au long du processus. Il ne s'agit pas ici de décrire de manière exhaustive l'ingénierie système, plus de précisions pourront être trouvées dans [Meinadier98, Incose06]. L'intérêt de sa mise en œuvre est particulièrement visible dans les phases amont de spécification et de conception du système puisque la plus grande partie du coût global de développement d'un système provient de décisions prises lors de ces phases. Les principales activités inhérentes à une démarche système sont de tracer les exigences, de justifier les choix, de vérifier et valider tous les résultats. Dans ce cadre, un des premiers processus déployé dans une telle démarche est l'ingénierie des exigences que nous décrirons plus particulièrement au chapitre 2.

Dans ce travail de recherche, l'ingénierie système fournit un cadre en ce qui concerne la conception de système critiques, et plus particulièrement les méthodes de modélisation et les méthodes d'analyse.

Selon l'*AFIS* (Association Française de l'Ingénierie Système), Le cycle de vie d'un système est l'ensemble des phases par lesquelles passe le système depuis l'émission des besoins qui le concerne jusqu'à son retrait du service. Ainsi, il se compose des phases de conceptualisation, de conception, de réalisation, d'intégration, d'exploitation et enfin de retrait de service [Afis07]. Nous décrivons plus précisément les cycles de développement et leur intérêt dans la section suivante.

1.1.3 Cycles de développement

Le processus de développement d'une solution peut être découpé en différentes phases ; ainsi, un cycle de développement est un enchaînement d'activités comprenant les méthodes et les outils utilisés. L'objectif de ce découpage est de définir des jalons tout au long du processus de conception : en effet, plus une erreur est détectée tardivement, plus son coût risque d'être élevé. Par conséquent, il convient de détecter et de corriger au plus tôt ses erreurs.

Par exemple, la méthode du cycle en V (figure 1.1) est couramment utilisée dans les grandes industries française, notamment dans le domaine ferroviaire [Boulanger06]. Dans ce modèle, la partie descendante correspond aux activités de conception, menant à la réalisation du système ou du logiciel, alors que la partie remontante est liée aux aspects de vérification et de validation. Le développement se fait par étapes, et chaque étape de conception est mise en correspondance avec une étape de validation.

Le cycle en V propose donc un cadre structurant le processus d'automatisation. Cependant, il est difficile en pratique de dissocier ces différentes phases. Il est courant de se rendre compte d'erreurs ou d'incohérences dans les spécifications initiales seulement au moment de l'implantation. De plus, de telles erreurs détectées tard dans le cycle entraînent la plupart du temps de grandes pertes en terme de temps ou de moyens financiers. Cette méthode illustre bien la nécessité de construire pas à pas le système en précisant de manière de plus en plus fine la solution choisie correspondant aux spécifications initiales.

Nous pouvons également évoquer d'autres cycles de développement utilisés dans l'industrie : la méthode du cycle de développement en cascade [Royce70] ou en spirale [Boehm88].

En conclusion de cette section nous tenons à souligner qu'il est nécessaire

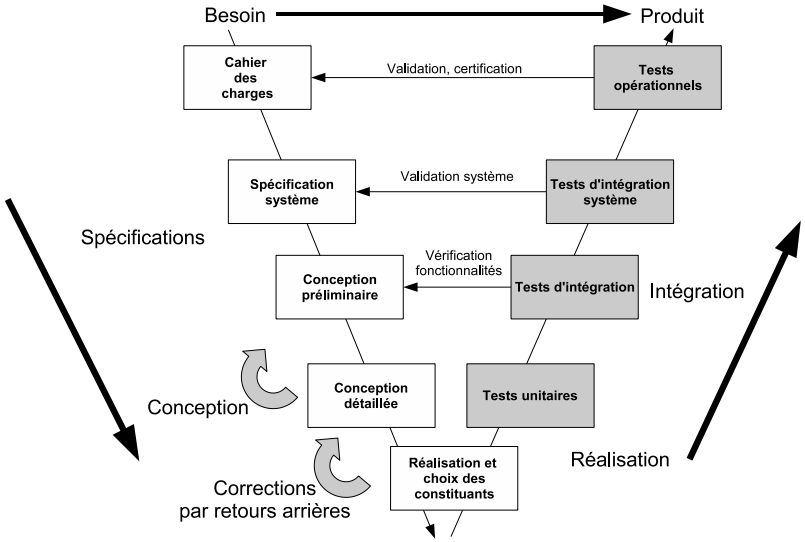

FIGURE 1.1 – Cycle de développement en V

de détecter les erreurs au plus tôt dans le processus de conception. Dans cette optique, l'utilisation de méthodes formelles permet d'obtenir un système validé par construction. Enfin, le découpage du processus de conception en différentes phases en fonction de l'objectif visé nous semble indispensable et motive le fait d'utiliser différents modèles complémentaires.

1.1.4 Types de modèles et niveaux de description d'un système

Nous l'avons vu dans les sections précédentes, il existe plusieurs types de modèles en fonction du contexte ou de l'objectif voulu par le concepteur. Dans ce cadre, il est nécessaire de définir plusieurs niveaux de description d'un système en fonction de l'étape que l'on souhaite réaliser au sein de ce processus de développement :

- les besoins et les exigences des usagers,
- un ensemble de solutions possibles,
- une solution particulière,
- la structure interne de la solution choisie, la description des diverses modules,

– des algorithmes/machines/raffinements utilisés par chaque module,
– le code final.

Il existe donc différents types de modèles utilisés tout au long de ce cycle de développement. Une typologie de ces différents types en fonction de leur objectif et de leur usage est proposée par la méthode d'analyse et de conception Sagace [Penalva99] développée à partir de 1993 au centre à l'énergie atomique (*C.E.A.*). Cette démarche consiste à observer le système depuis différents angles et en plusieurs étapes. L'apport central de cette méthode est une matrice de points de vue de modélisation, proposant trois visions possibles du système (fonctionnelle, organique et opérationnelle) et trois perspectives temporelles d'étude (figure 1.2).

Nous précisons les trois différentes visions du système :
– la vision *fonctionnelle* représente ce que fait le système par rapport à son environnement ;
– la vision *organique* représente ce qu'est le système ;
– la vision *opérationnelle* représente ce que décide le système pour remplir la mission.

	Action	Fonctionnement	Transformation
Vision fonctionnelle	Fonctions	Processus	Scénarios
Vision organique	Sous-système opérant	Sous-système de commande	Sous-système auxiliaire
Vision opérationnelle	Conduite	Gestion	Anticipation

FIGURE 1.2 – Matrice des modèles SAGACE

Nous nous inspirons de cette méthode pour définir différents modèles en fonction de l'objectif que l'on souhaite atteindre et de la phase que l'on traite dans le processus de développement (figure 1.3) .

Afin de réaliser un système conforme aux besoins énoncés dans le cahier de charges, il est nécessaire de décrire ce que le système doit satisfaire au sein d'un modèle dit *prescriptif*. Ce modèle peut être décrit sous la forme

d'exigences, que l'on peut séparer en exigences *fonctionnelles* (ce que doit faire le système) et *non-fonctionnelles* (performance, fiabilité, sécurité..). Par exemple, dans le cadre de ce travail, la spécification des systèmes soumis à des exigences de sécurité doit clairement identifier ce qui ne doit pas arriver.

La conception d'un système a pour objectif la construction d'un modèle dit *constructif* décrivant une solution répondant aux exigences auxquelles le système doit répondre. Ce modèle permet de représenter un système en mettant en évidence les propriétés souhaitées.

Ensuite, il est nécessaire de construire des modèles *prospectifs* dont l'objectif est de déduire le comportement du système dans des situations nouvelles à partir d'un état connu. Ces modèles sont de deux types :

1. Modèles *formels :* capables de prévoir, de valider, de prouver des comportements ;
2. Modèles *analytiques :* permettant l'estimation de performance, l'analyse de la sûreté de fonctionnement, la simulation.

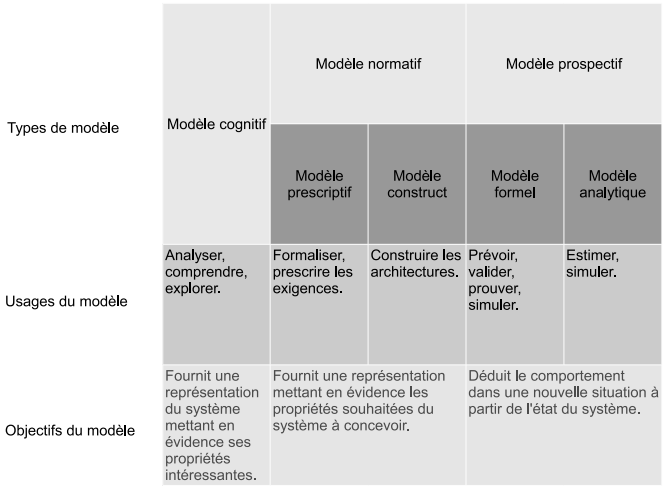

FIGURE 1.3 – Typologie et objectifs des modèles SAGACE

De plus, nous pouvons citer [Chapurlat07] qui insiste sur l'importance du rôle du modeleur dans le processus de conception :

– La modélisation est tout d'abord guidée par l'objectif souhaité par le modeleur : construire un modèle de simulation, de pilotage, de contrôle..,

– elle est ensuite rendue possible par la maîtrise qu'a le modeleur des outils et des méthodes utilisées.

Enfin, cette méthode s'appuie sur un ensemble de modèles purement descriptifs se basant sur un langage de modélisation souple, intuitif, proche du langage naturel mais insuffisant dans notre contexte de travail. Dans le chapitre 3 nous justifions les langages que nous avons choisi pour décrire un processus de conception de systèmes critiques ferroviaires.

De plus, de notre point de vue, lorsque l'on représente les exigences contenues dans le cahier des charges, il faut être attentif à ne pas pré-sélectionner a priori une solution, notamment parce que cela pourrait entraîner l'exclusion de solutions alternatives.

1.1.5 Types de propriétés

Nous avons expliqué dans la section précédente l'intérêt d'utiliser différents types de modèles durant le processus de conception. Ces modèles permettent de vérifier différentes propriétés en fonction de l'objectif à atteindre.

Nous présentons donc ici différents types de propriétés (définition 1.3) à spécifier et à vérifier lors de la conception d'un système complexe.

Définition 1.3 (propriété)
Une propriété est une qualité propre, une caractéristique intrinsèque (fonctionnelle, comportementale, structurelle ou temporelle) que doit posséder un système. Une propriété traduit une attente, une exigence, une finalité à laquelle l'entité doit répondre [Chapurlat07].

D'abord, les propriétés peuvent être *statiques*. Elle permettent dans ce cas d'assurer la cohérence du système. Dans les approches basées sur les transitions d'états, elles sont spécifiées sous la forme d'invariants ou de garde sur les opérations. Elles caractérisent plus particulièrement les architectures du système.

Ensuite, une propriété est dite *dynamique* si elle traite de l'occurrence ou de l'ordonnancement des événements.

Nous pouvons ici référer à la section 3.1 où sera justifié le choix des outils utilisés dans notre travail ; les réseaux de Petri serviront l'étude des propriétés dynamiques du système alors que la méthode B est envisagée pour vérifier ses propriétés statiques.

Les propriétés dynamiques peuvent être qualifiées d'invariantes, de temporelles ou d'événementielles. Parmi ces propriétés, nous pouvons citer les propriétés de sûreté ou de vivacité par exemple, qui caractérisent la capacité du système à délivrer des services conformes aux exigences, même dans des conditions particulières.

Les propriétés de sûreté sont de la forme « quelque chose d'interdit ne doit jamais arriver ». L'absence de blocage, l'exclusion mutuelle sont des exemples de propriétés de sureté.

Les propriétés de vivacité sont quant à elles de la forme « quelque chose d'attendu arrivera nécessairement ». Dans un langage formel basé sur les transitions d'état, ce type de propriété est difficile à vérifier puisque les séquences de transitions nécessaires à cette vérification n'y sont pas naturelles. La terminaison d'une application, la garantie de service sont des exemples de propriétés de vivacité.

Cet inventaire montre que le choix d'un outil de modélisation dépend notamment du type de propriété à vérifier lors de la conception d'un système. De plus, il justifie l'utilisation de différents types de modèles complémentaires pour pouvoir vérifier un ensemble de propriétés.

Les deux prochaines sections précisent les types de systèmes que nous étudions plus particulièrement. Dans un premier temps, nous décrivons les systèmes à événements discrets, types de systèmes que l'on retrouve de façon naturelle dans la modélisation des systèmes informatiques, des systèmes de production et, ce qui nous intéresse particulièrement, les systèmes de transport. Ensuite, nous précisons que nous nous intéressons à des systèmes soumis à des contraintes temporelles fortes devant être respectées pour garantir leur sécurité.

1.1.6 Système à événements discrets

Des systèmes, de plus en plus complexes et flexibles, sont représentés à un niveau de conceptualisation adapté à leur contrôle ou à leur commande. Ce type de systèmes se trouve dans des champs d'application tels que les systèmes informatiques ou de production, les réseaux de télécommunications ou de transport. Un Système à événements discrets est une abstraction utile pour décrire de nombreux systèmes dynamiques dont l'espace d'état est décrit par un ensemble discret.

Pour ce type de systèmes, l'occurrence d'événements détermine l'évolution de ces systèmes dont le comportement est souvent caractérisé par des phénomènes de concurrence et de synchronisation vis-à-vis des ressources ou des utilisateurs. Dans ces domaines, le progrès technologique, couplé à une nécessité de compétitivité, impose de disposer d'outils formels et de techniques d'analyse et de synthèse de commande de plus en plus complexes et efficaces de façon à respecter le cahier des charges.

1.1.7 Systèmes à contraintes temporelles

Nous nous intéressons plus particulièrement à la modélisation et à la vérification formelle des systèmes temporisés. La généralisation des systèmes offrant une qualité de service garantie (temps de réponse, taux de pertes..) nécessite des modèles explicitant les contraintes temporelles. La criticité de ces systèmes rend impérative leur vérification, tant au niveau qualitatif (comportement du système) que quantitatif (cohérence temporelle).

Les travaux présentés dans ce rapport traitent de systèmes à contraintes temporelles. Par exemple, le procédé peut se retrouver dans un état interdit si un résultat nécessaire à sa bonne évolution est produit trop tôt ou trop tard. Pour cette classe de système, le facteur temps est une composante primordiale. En effet, ce dernier n'affecte pas seulement les performances du système mais également sa validité fonctionnelle. Pour ce type de système, il est nécessaire de disposer de structures de commande assurant un bon comportement (c'est à dire respectant les spécifications du cahier des charges) et respectant l'ensemble des contraintes temporelles mise en jeu. Dans ce contexte, nous avons porté notre intérêt sur l'analyse et le contrôle des systèmes à contraintes de temps de séjour.

1.1.8 Systèmes critiques

Un système est considéré comme critique lorsque une défaillance implique de graves conséquences en terme de vies humaines, de pertes économiques ou de dégâts sur l'environnement.

Les domaines concernés sont nombreux : aérospatiale, énergie, santé, transport. On peut d'ailleurs citer quelques exemples concrets de dysfonctionnement de systèmes critiques dans différents domaines :

Thérac - 25 Entre Juin 1985 et Juillet 1987, une défaillance logicielle d'un appareil médical utilisé pour traiter les cellules cancéreuses a provoqué la mort de plusieurs personnes à cause d'une exposition trop importante aux radiations [Leveson et al.93].

Ariane 5 Le 4 juin 1996, le lanceur de la fusée Ariane 5 explose en vol après avoir dévié de sa trajectoire à cause d'une erreur logicielle dans la spécification d'une fonction. Le rapport de la commission d'enquête recommande le développement de méthodes de vérification [Lions96].

Ce type de système nécessite un haut niveau de qualité selon divers critères tels que la fiabilité, la sécurité ou la disponibilité. Il s'agit donc de système soumis à de fortes exigences de sûreté de fonctionnement (voir définition 1.4). Du reste, il est nécessaire de prendre en compte ces éléments dans les phases de conception, de réalisation et d'exploitation.

Ces systèmes nécessitent de prendre des précautions souvent fixées par des normes lors de leur développement. Par exemple, ces normes peuvent imposer des impératifs de traçabilité. C'est à dire que le système doit respecter les spécifications du cahiers des charges tout au long du processus de conception en formant une chaîne complète de traçabilité jusqu'à la mise en œuvre du système.

1.1.9 Sureté de fonctionnement

Comme nous l'avons montré dans la section précédente, les systèmes critiques ont un besoin fondamental en sûreté de fonctionnement (définition 1.4).

Définition 1.4 (Sûreté de fonctionnement)
La sûreté de fonctionnement d'un système est la propriété qui permet de placer une confiance justifiée dans le service qu'il délivre [Laprie et al.95].

La sûreté de fonctionnement se caractérise par un ensemble d'attributs (notés FMDS) :

- la fiabilité : le système remplit sans défaillance les fonctions demandées,
- la maintenabilité : le système est réparable ou modifiable,
- la disponibilité : le système est toujours prêt à utilisation,

– la sécurité : le système ne génère aucun événement aux conséquences néfastes ou catastrophiques.

Il convient de préciser la notion de sécurité qui nous intéresse plus particulièrement en distinguant la sécurité-innocuité (*safety* en anglais) et la sécurité-confidentialité (*security*). La première concerne la prévention des catastrophes, la protection des systèmes contre les accidents dus à l'environnement. La seconde concerne la prévention des malveillances (agressions, intrusions, vandalisme). Dans la suite de ce mémoire, le terme « sécurité » sera employé au sens « safety ».

Pour gérer au mieux la sûreté de fonctionnement, deux équipes travaillent en parallèle :

– une équipe dédiées à la sûreté de fonctionnement réalise les analyses liées à la maîtrise des risques. Elle définit l'ensemble des risques auxquels le système à concevoir peut être soumis ;
– une équipe de conception du système ayant pour objectif de prendre en compte les exigences contenues dans le cahier des charges et de démontrer leur bonne prise en compte tout au long du processus.

L'objectif de ce mémoire n'étant pas de faire un état de l'art des concepts utilisés en sûreté de fonctionnement, nous proposons au lecteur intéressé de se référer à [Laprie et al.95] pour obtenir toutes les définitions nécessaires à la bonne compréhension de ce domaine. Nous citons quand même certaines méthodes parmi les plus utilisées pour les études de sûreté de fonctionnement : l'analyse préliminaire de risques (APR), l'analyse des modes de défaillance, de leur effets et de leurs criticité (AMDEC) ou les arbres de défaillances. Il s'agit de méthodes d'analyse prévisionnelles utilisées par l'équipe dédiée à la sûreté de fonctionnement.

Les études de sûreté de fonctionnement sont le point de départ de la spécification, notamment pour les parties critiques, et permettent de comparer et d'optimiser des solutions. Elles permettent, selon les objectifs visés et les méthodes utilisées, de prévoir, d'éliminer ou d'éviter les fautes auxquelles est soumis le système. Elles définissent les exigences non-fonctionnelles (exigences de sécurité, de disponibilité...), alors que les exigences fonctionnelles sont identifiées après analyse directe du cahier des charges.

Dans le domaine ferroviaire qui est le cadre de notre travail, les principales méthodes de conception de systèmes sûrs de fonctionnement sont la

sécurité « intrinsèque », la redondance, le surdimensionnement ou le codage d'applications informatiques sécuritaires.

Dans ce mémoire, nous nous situons dans le cadre de la conception de systèmes nécessitant la traçabilité et la vérification des exigences dans le but de valider des solutions. Il ne s'agit pas de fournir une nouvelle méthode d'analyse de sûreté de fonctionnement mais de proposer, à partir d'une spécification des exigences fournie, une approche permettant de construire un système conforme et certifiable comme sûr de fonctionnement.

1.2 Le contexte ferroviaire

Comme nous l'avons expliqué dans la section 1.1.1, il est indispensable de définir précisément le contexte applicatif pour permettre une évaluation et des choix d'outils de modélisation pertinents. Nous présentons donc ici le contexte général de la sécurité des systèmes de transport ferroviaires dans lequel se situent nos travaux.

1.2.1 Introduction au transport ferroviaire

Le transport ferroviaire est un système de transport guidé terrestre, c'est à dire comportant des véhicules assujettis à circuler le long d'une voie rigide [IEC05]. Il a pour caractéristique de se déplacer sur une seule dimension. Ceci a pour conséquence de rendre impossible tout dépassement ou évitement d'un obstacle. De surcroit, son adhérence et ses frottements sont faibles, ce qui implique une distance de freinage importante. À titre indicatif, un train roulant à 100 km/h a une distance de freinage de 1000 mètres, tandis qu'un TGV roulant à 300 km/h mettra 3500 mètres pour s'arrêter. Ces contraintes induisent une conduite en aveugle, basée sur l'anticipation plutôt que sur la vision. En effet, la distance d'arrêt d'un train est bien supérieure à la distance de visibilité du conducteur. Par conséquent, le conducteur du train doit être prévenu au plus tôt pour pouvoir réagir à un quelconque événement redouté.

Les transports guidés ferroviaires se scindent en deux types principaux :

- un type régulier, fermé, urbain, regroupant par exemple le métro ou le tramway,
- un autre type ouvert et partagé pour le transport de personnes ou de

biens sur de plus longues distances, tels que les trains de fret, les trains régionaux ou les trains à grande vitesse.

Le chemin de fer est un transport de masse dont il est impératif de maîtriser les risques. En effet, de nombreux exemples illustrent qu'une défaillance dans un système de transport ferroviaire peut entraîner des conséquences dramatiques, telles que des blessés graves ou des morts, des dommages matériels importants.

En outre, bien que le chemin de fer soit beaucoup plus sûr que l'automobile par exemple (voir tableau 1.1), l'acceptation du risque est beaucoup plus faible ; en effet, les passagers sont passifs et intransigeants avec le transporteur qui doit assumer la sécurité.

Moyen de transport	**Nombre de tués par cent millions de passagers - km**
Route (total)	0,95
Voiture	0,7
Aérien (civil)	0,035
Rail	0,035

TABLE 1.1 – Accidentologie des différents moyens de transport [Koornstra03].

La responsabilité du transport ferroviaire se partage entre le gestionnaire d'infrastructures, responsable des installations fixes, et les entreprises ferroviaires, responsables des matériels roulants et donc de la sécurité des circulations. En France, le gestionnaire de l'infrastructure est la société Réseau Ferré de France (RFF), alors que la Société Nationale des Chemins de Fer (SNCF) est l'entreprise ferroviaire « historique ». Désormais, la fin du monopole de la SNCF a fait apparaitre de nouvelles entreprises spécialisées dans le transport du fret qui circulent déjà sur le réseau ferré. L'activité du transport de voyageurs suit une évolution similaire.

1.2.2 Sécurité des systèmes ferroviaires

Les aspects de sécurité font partie du cœur de métier, assumé et revendiqué par les acteurs du transport ferroviaire, qui ont d'ailleurs bâti leur image sur la fiabilité et la sécurité. Les différents dangers pouvant survenir lors de l'exploitation d'un système ferroviaire sont principalement des accidents de

la circulation de type collision (par rattrapage, en nez à nez ou par prise en écharpe) ou déraillement (dû à une sur-vitesse ou à une défaillance technique). Pour éviter tout accident, des fonctions de sécurité ont été mises en place :

- le cantonnement des trains : il repose sur la détection de l'occupation de la voie. Il s'agit de délimiter des zones à l'intérieur desquelles il ne peut y avoir qu'un seul train,
- la signalisation : elle indique au conducteur du train les conditions de circulation pour lui permettre de régler la conduite du train. Elle se compose entres autres des signaux d'arrêt, de limitation de vitesse ou de manœuvre. Ces signaux peuvent être fixes ou mobiles, lumineux ou mécaniques, permanents ou temporaires,
- les aiguillages : ils permettent différentes possibilités d'itinéraires pour les circulations des trains,
- le contrôle de la conduite du train.

Ensuite, des procédures réglementent l'exploitation ferroviaire et sont mises en place pour contraindre la sécurité en appliquant ces différentes fonctions de sécurité. Elles permettent d'appliquer les principes de base de la sécurité : le maintien de la voie libre, l'itinéraire vers voie libre, l'assurance de pouvoir arrêter les trains.

1.2.3 Politique de transport ferroviaire européenne

Bien que le niveau de sécurité des transports ferroviaires européens soit très élevé, surtout par rapport au transport routier [Koornstra03] qui est son principal concurrent, le livre blanc [White paper01] a soulevé des lacunes en matière de sécurité et d'interopérabilité. Pour répondre à ces lacunes, des directives européennes sont apparues concernant notamment :

- la sécurité des chemins de fer communautaires [Dir04],
- l'interopérabilité du système ferroviaire de la communauté européenne [Dir08],
- la certification des conducteurs de trains sur le réseau ferré européen [Dir07].

D'après la communauté européenne, les chemins de fer doivent être libéralisés pour être concurrentiels [White paper01]. Cette démarche implique un renforcement de la régulation chez les exploitants ferroviaires et une meilleure

intégration des réseaux ferroviaires pour faciliter les opérations aux frontières. En ce sens, cette politique a déjà conduit à la séparation de la gestion de l'infrastructure et des entreprises ferroviaires pour permettre l'ouverture à la concurrence. Néanmoins, cette ambition est rendue délicate à cause des différentes pratiques dans chaque pays et des spécificités des normes nationales.

C'est pourquoi l'union européenne a élaboré un système européen de gestion du trafic ferroviaire (ERTMS, *European Rail Traffic Management System* [ERT09]). Ce système est un ensemble de spécifications techniques désormais obligatoires sur toute nouvelle ligne ferroviaire reliant au moins deux pays de l'union. Les nouveaux systèmes doivent démontrer qu'ils sont conformes à ces spécifications en obtenant un certificat de conformité.

L'objectif est donc la mise en place d'un système ferroviaire européen unique, plus compétitif et plus sûr, couvrant tout le marché communautaire. Pour y parvenir, il est nécessaire d'établir des principes communs en ce qui concerne la sécurité et l'interopérabilité.

Ce constat nous amène à présenter plus particulièrement les concepts d'interopérabilité et de sécurité ferroviaires.

1.2.3.1 L'interopérabilité ferroviaire

L'interopérabilité est l'aptitude d'un réseau de transport à permettre une exploitation optimale en réduisant les contraintes techniques, culturelles, législatives, organisationnelles et socio-économiques. Elle doit permettre la circulation de trains, de différentes sortes, provenant de différents pays et appartenant à différentes entreprises, dans l'ensemble des états concernés.

Des exigences en matière de sécurité au niveau des sous-systèmes ont été établies dans des directives [Dir04] et précisées par les spécifications techniques d'interopérabilité (*S.T.I.*).

Enfin, *ERTMS* (European Rail Traffic Management System) propose une solution pour la gestion d'un système de transport ferroviaire unique. À long terme, ce système devra remplacer en les harmonisant les différents systèmes nationaux.

1.2.3.2 La sécurité ferroviaire

Le livre blanc [White paper01], publié par la commission européenne en 2001, considère que la sécurité est un des aspects majeurs afin de mettre en place un marché ferroviaire unique.

Le niveau de sécurité du transport ferroviaire en Europe est très élevé. Le projet d'un réseau unique implique le maintien voire l'amélioration de ce niveau. C'est pourquoi des critères d'acceptation des risques ont été mis en place par les états membres. Ainsi, en France, les nouvelles technologies doivent être globalement au moins aussi bonnes que les anciennes (principe G.A.M.E. : Globalement Au Moins Équivalent [El koursi et al.07]). Nous pouvons ici citer d'autres critères, tels que le principe ALARP au Royaume-uni ou le principe MEM. en Allemagne. De plus, il est nécessaire de prendre des mesures assurant l'interopérabilité des réseaux ferrés. Évidemment, il reste des équipements qui n'intègrent pas les nouvelles technologies. D'un point de vue général, le critère d'interopérabilité doit être étudié lorsque cohabitent plusieurs technologies différentes, même s'il n'est pas possible d'affirmer que l'une est meilleure que l'autre.

Par exemple, les logiciels de contrôle/commande intervenant dans les systèmes de transport ferroviaires sont considérés comme critiques et donc doivent disposer de méthodes de conception permettant de garantir de manière certaine qu'ils ne failliront pas. Par conséquent, des normes ont été créées pour définir un cadre de conception précis visant à éviter toute défaillance dans les systèmes considérés comme critiques.

Ces normes et leurs conséquences sur le processus de conception sont expliquées dans la section suivante.

1.2.4 Contexte normatif et législatif

Comme nous l'avons expliqué dans la section 1.2.3, le contexte européen nécessite une harmonisation des normes pour permettre l'interopérabilité du réseau.

La norme CEI 61508 [CEI98] est une norme générique, créée en version européenne par le CENELEC en 2002, qui définit les standards de sécurité et qui est aujourd'hui utilisée comme référentiel par tous les grands secteurs industriels. Elle traite de la sécurité fonctionnelle des systèmes élec-

triques/électroniques et électroniques programmables. Elle a amené des nouveautés dans la façon d'intégrer et de réaliser les activités de sûreté de fonctionnement dans le cycle de développement d'un système. Cette norme a permis de définir des niveaux d'intégrité dans la gestion du risque pour des systèmes prenant en compte aussi bien les aspects quantitatifs que qualitatifs.

Depuis sa création, plusieurs dérivés de cette norme ont vu le jour. À chaque fois, ces normes filles ont été créées dans le but de rendre applicable l'EN 61508 pour les différents secteurs concernés. Ce fut notamment le cas pour le secteur ferroviaire. Ainsi, les standards européens pour le développement et l'implémentation des certifications ferroviaires ont été publiés dans les normes CENELEC 50126, 50129 et 50128 [EN100, EN103, EN101] qui sont à la base du processus de démonstration de la sécurité d'un système ferroviaire. Cette dernière nous intéresse plus particulièrement puisqu'elle concerne les aspects logiciels de commande.

Les concepts de base liés au contexte normatif dans la sûreté de fonctionnement sont :

- les fonctions de sécurité,
- les systèmes concernés par la sécurité,
- les niveaux d'intégrité de sécurité (SIL : *Safety Integrity Level*).

La norme EN 50128 met en avant deux notions :

- les exigences,
- la traçabilité des exigences sur l'ensemble du processus.

De plus elle recommande l'utilisation des méthodes formelles pour la spécification et la conception de systèmes critiques (SIL 3/4).

Par ailleurs, l'Union Européenne a mis en place de nouvelles spécifications techniques d'interopérabilité (S.T.I.) auxquelles doivent répondre les nouvelles constructions ferroviaires et qui recommandent l'utilisation des normes 5012X. Dans cette optique, il semble intéressant d'évaluer et de valider les modèles de composants détaillés dans ces spécifications. De nouvelles méthodologies de conception, de validation et d'évaluation sont nécessaires. Les contraintes en termes de qualité de service, d'exigences temporelles, de sûreté de fonctionnement se présentent comme les principaux défis à être abordés.

1.2.5 Certification des systèmes de sécurité ferroviaire

La conception d'un système de sécurité est difficile. Sa vérification l'est tout autant, voire plus. Ainsi, de nombreux travaux ont été faits sur l'application des techniques formelles dans le développement des systèmes critiques, notamment dans le domaine de la vérification et de la validation. L'objectif est qu'à la fin du processus de développement formel le système puisse être certifié comme sécuritaire.

Ainsi, les systèmes critiques sont soumis à des autorités de certification qui doivent vérifier que les recommandations prévues par les normes ont été respectées. Dans le contexte français, l'organisme certifié pour des systèmes ferroviaires est CERTIFER.

Par ailleurs, comme nous l'avons vu dans la section 1.2.3, la conformité des nouveaux systèmes ferroviaires à ERTMS doit être attesté par un certificat de conformité. En France, ce certificat est délivré par des experts ou organismes qualifié agréé (EOQA) reconnu par le ministère des transports, tels que CERTIFER. Nous pouvons également citer ici l'EPSF (Établissement Public de Sécurité Ferroviaire), organisme disposant des compétences nécessaires en matière de sécurité ferroviaire, et qui reste indépendant des différents opérateurs ferroviaires. Cependant, toujours dans un souci d'harmonisation, l'objectif est d'instaurer un certificat communautaire unique, conforme aux exigences contenues dans les S.T.I., au droit communautaire et aux règles de sécurité nationales.

Notre ambition est de proposer une méthodologie fournissant une aide à la certification. Il s'agit de fournir aux experts du domaine des outils et des méthodes permettant de valider et de vérifier des systèmes en vue de leur certification. En effet, il nous parait indispensable d'assumer que certaines décisions ne peuvent être prises sans l'avis d'un expert. Nous ne nous plaçons pas dans un contexte de recherche d'optimalité, mais nous cherchons à obtenir l'assurance que le système étudié est conforme aux exigences, particulièrement en termes de sécurité et de temporalité.

1.3 Bilan sur la conception de système ferroviaires critiques

Pour résumer, dans le cadre de cette thèse, nous cherchons à combiner plusieurs domaines afin de respecter les différentes contraintes imposées tout d'abord par le contexte de la sécurité ferroviaire étudié et en particulier les recommandations données par ses normes, et ensuite par la modélisation et l'évaluation de systèmes complexes.

Notre objectif est de couvrir une partie du cycle de développement classique d'automatisation industrielle, de la spécification à l'implantation du système.

Dans cette optique, nous cherchons à combiner différentes approches telles que l'ingénierie système, et plus particulièrement l'ingénierie des exigences nécessaire à la traçabilité et à la validation de ces exigences, les méthodes formelles et l'ingénierie dirigée par les modèles.

Chapitre 2

Méthodes utilisées pour garantir la sécurité des systèmes

Sommaire

Ce chapitre présente différentes méthodes et techniques permettant de vérifier la sécurité lors du développement de systèmes complexes. Comme nous l'avons vu en conclusion du chapitre 1, ces méthodes sont recommandées par des normes dans un contexte métier de sécurité des systèmes de transport ferroviaires.

Ainsi, nous présentons dans un premier temps l'ingénierie des exigences, en précisant les concepts de propriétés et d'exigences, et en montrant comment ces dernières peuvent être formalisées, tracées et analysées tout au long du processus de conception. Ensuite, la section 2.2 définit ce qu'est une spécification formelle en décrivant différents types de langages formels afin de justifier nos choix. De plus, les différentes méthodes existantes pour valider et vérifier ces spécifications dans un objectif d'aide à la certification de système sont présentées. Enfin, notre choix s'étant porté sur l'utilisation complémentaires de deux outils formels différents, nous présentons l'ingénierie dirigée par les modèles qui permet de passer d'un modèle à l'autre de manière formalisée.

2.1 Ingénierie des exigences

La phase de conception se base sur la notion d'exigence (définition 2.1), concept emprunté au domaine de l'ingénierie système (voir section 1.1.2) traitant du développement d'applications logicielles complexes. Ainsi, l'ingénierie des exigences consiste à clairement définir les exigences imposées à l'application avant de la concevoir et à les gérer pendant tout le cycle de vie du système.

Définition 2.1 (exigence)
Une exigence est une caractéristique d'un système dont un utilisateur ou un client a besoin pour résoudre un problème ou atteindre un objectif dans un contexte donné [IEE90] .

L'expression des exigences consiste à rechercher, ordonner, caractériser, hiérarchiser et valoriser les fonctions souhaitées. La spécification des exigences représente pour les acteurs du projet une référence documentée commune de leur compréhension des produits ou des services. Elle peut permettre de stipuler les exigences fonctionnelles ainsi que les exigences en ma-

tière de rendement, de renseignements, de capacités, de sécurité, d'ergonomie, d'exploitation, de maintenance, d'interface et de qualifications. Le processus d'ingénierie des exigences est un processus critique dans la recherche d'adéquation d'un système aux attentes de ses utilisateurs. En effet, au départ, le besoin peut être exprimé, mais également implicite ou potentiel (notion d'évolution des exigences). L'objectif est alors de ne le rendre que « exprimé ».

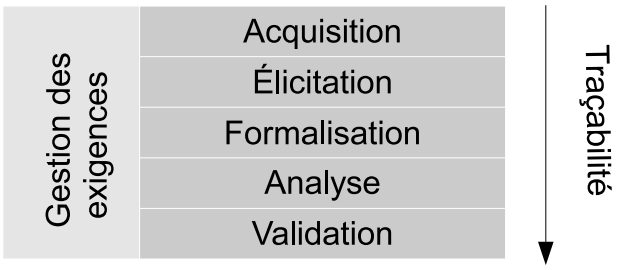

FIGURE 2.1 – Phases de l'ingénierie des exigences.

L'ingénierie des exigences est donc l'étape initiale de tout développement d'un système (ou d'un logiciel). Elle met en œuvre un ensemble de techniques permettant aux utilisateurs d'exprimer leurs besoins et à l'équipe de développement de les comprendre. Elle passe par en ensemble d'activités (voir figure 2.1) permettant de découvrir, analyser, valider et faire évoluer l'ensemble des exigences relatives à un système [Gti07]. Ces étapes débouchent sur un document contractuel pour le développement ultérieur : le cahier des charges. Ce dernier contient les spécifications nécessaires pour assurer la cohérence des exigences et pour les gérer au cours du processus de développement. Nous pouvons nous référer à [Hellouin02] qui propose une démarche permettant d'améliorer le contenu des spécifications pour la conception des systèmes embarqués.

2.1.1 Propriétés, exigences et spécifications

Il nous parait pertinent de préciser à ce niveau du mémoire les termes « besoin », « propriété (définition 1.3) », « exigence (définition 2.1) » et « spécification ».

La gestion des exigences peut avoir un très gros impact sur le succès de la réalisation d'un projet. Sa mise en œuvre implique cependant de bien

identifier ce qu'est un exigence et comment elle doit être prise en compte tout au long du processus de conception [Hull et al.05]. Au tout début de ce processus, les exigences peuvent être trouvées dans un document fourni par le client ou le demandeur. Elles sont alors plutôt exprimées sous forme de besoins. En effet, elles représentent alors la perception qu'a l'utilisateur du système [Essame02]. Mais les exigences ne se réduisent pas aux besoins du client ; elles se trouvent également dans des normes, des décrets ou dans le référentiel métier. Il est donc nécessaire de passer par une phase d'acquisition et d'extraction des exigences visant à fournir une liste de recommandations complètes, consistantes et non ambiguës. En effet, la plupart du temps le liste d'exigences initiale est mal exprimée, celles-ci étant par exemple non applicables, trop générales, trop précises, contradictoires,... Cette liste de recommandations doit être réalisée de concert avec le client et définit les exigences à prendre en compte lors de la réalisation du système. Ces dernières doivent être identifiables, vérifiables et modifiables pendant tout le processus.

Ensuite, il nous semble intéressant de faire les liens entre ces concepts, par exemple celui entre les exigences contenues dans le cahier des charges et les propriétés attendues du systèmes. Ainsi lors des phases de conception, une exigence est traduite sous la forme d'une ou de plusieurs propriétés sur les composants ou les sous-systèmes (voir section 1.1.5).

Enfin, une spécification décrit les services qui sont attendus du système à réaliser et les conditions dans lesquelles ces services doivent être rendus. En fait, elle décrit une solution spécifique qui va satisfaire aux exigences et qui va respecter les contraintes. Elle regroupe l'ensemble cohérent et complet des exigences (on parle d'ailleurs de « spécification des exigences ») et contient les concepts nécessaires à l'utilisation du système, en décrivant particulièrement le point de vue de l'utilisateur. Une « bonne »spécification doit permettre aux différents acteurs du projet de bien communiquer. Dans cette optique, plusieurs critères sont nécessaires pour caractériser une spécification, telles que sa clarté, sa complétude, sa correction ou sa concision.

2.1.2 Élicitation des exigences

Dans un processus d'ingénierie des exigences, la première étapes est l'élicitation. Elle consiste à mettre en évidence et à collecter l'ensemble des exigences à partir de toutes les sources concernées (client, normes, référentiel

métier...). Ainsi, lors de l'énoncé d'une exigence, il est nécessaire d'associer un critère ou une procédure permettant de vérifier qu'elle est bien respectée.

Dans un premier temps, les exigences sont divisées en deux catégories :

- les exigences système : elles sont de haut-niveau et extraites du cahier des charges ;
- les exigences techniques : elles sont relatives à un choix technologique ou architectural proposé lors du raffinement d'une solution.

2.1.3 Caractéristiques des exigences

On distingue deux classes d'exigences de haut-niveau dans un système :

1. les exigences fonctionnelles : elles représentent les différentes fonctionnalités que devra satisfaire le système et correspondent à l'aspect opérationnel du système. Généralement elles correspondent à des propriétés générales telles que la possibilité de revenir à l'état initial, l'absence de « deadlock »...
2. les exigences non-fonctionnelles : elles correspondent aux critères de qualité attendus du système. Ainsi on peut citer pour exemple les critères temporels (délais min, max) que doit satisfaire le système.

Au sein de ces exigences non-fonctionnelles temporelles, on peut distinguer deux catégories :

- Celles où le temps est exprimé de manière qualitative ou logique (ex : passage du train après fermeture des barrières).
- Celles où le temps est exprimé de manière quantitative. On considère alors l'ordre des événements mais également les distances temporelles entre ces événements (ex : passage du train 10s après fermeture des barrières).

2.1.4 Analyse et formalisation

Le processus d'analyse consiste à compléter, arranger et tracer l'ensemble des exigences pour éliminer toute ambiguïté ou redondance. L'objectif est d'obtenir un document complet et cohérent, présenté généralement en langage naturel. Cette spécification définit complètement le problème que les activités de conception doivent résoudre en apportant une solution.

Après cette phase d'analyse, il faut construire un modèle du système intégrant toutes ces exigences ; il s'agit de proposer une solution comprenant l'ensemble des exigences soumises. Cette solution peut être décrite par de nombreux modèles, de préférence des modèles formels basées sur des notations mathématiques.

2.1.5 Traçabilité

La notion de traçabilité des exigences constitue un élément majeur de l'évolution des exigences. Il s'agit de conserver une trace des exigences et des liens entre plusieurs niveaux d'exigences.

La traçabilité permet de mieux comprendre les exigences et de prendre en compte leurs évolutions au cours du projet. De plus, elle s'accompagne d'une nécessité de vérifier et valider la solution proposée pour que le système soit conforme aux besoins à satisfaire [Laval00]. Pour tracer une exigence, il faut tout d'abord l'identifier et l'enregistrer tout au long de son évolution. Elle est en effet transversale tout au long du processus de conception. Mais la traçabilité ne se limite pas à établir un historique, il s'agit également d'établir et de souligner les liens entre les exigences (liens historiques, de déclinaison, de dépendance). Par exemple, des liens de traçabilité doivent montrer qu'une exigence du niveau n dans le processus de conception est liée à un besoin au niveau $n-1$, et réciproquement qu'aucune exigence n'a été oubliée.

Ainsi, pour appliquer une bonne traçabilité, il faut identifier les exigences à satisfaire de manière formelle, puis établir les liens entre ces exigences et prendre en compte leur évolution. La section 2.2 présente comment les spécifications peuvent être présentées de manière formelle pour permettre leur validation et leur vérification.

2.1.6 Un outil pour la gestion des exigences : SysML

Cette section présente un exemple de normalisation mis en œuvre pour une gestion des exigences. L'objectif est de définir un cadre pour la définition d'exigences. Ce langage de modélisation graphique a récemment été adopté par l'OMG [1]. L'intérêt de cette section est d'évoquer des approches

1. Object Management Group : organisme international de normalisation à l'origine entre autres des standards UML et COBRA.

proposées actuellement pour la synthèse et la formalisation des pratiques de modélisation en ingénierie système. La notation UML [Booch et al.99] permet de modéliser des systèmes complexes au moyen de plusieurs formalismes (graphe de classes, diagramme de séquence, diagramme états/transitions,...). SysML (System Modeling Language [Friedenthal et al.08]) reprend une partie du langage UML 2.0 en ajoutant des compléments associés aux aspects système et traitement des exigences. Il propose notamment un diagramme d'exigences et introduit des notions importantes telles que *requirement* qui associe aux exigences des attributs, tels que leur identification, leur description textuelle, leur source ou leur type, ou *testcase*. Il est possible d'établir les liens entre les différentes entités grâce à des relations de composition, de dérivation, de satisfaction ou de dérivation. SysML permet ainsi de décrire des exigences au sein d'un modèle UML et permet de construire un modèle commun les spécifications, les contraintes et les paramètres de l'ensemble du système.

2.2 Spécifications formelles

Comme nous l'avons expliqué dans la section 1.2, l'utilisation des méthodes formelles pour la conception de systèmes critiques est recommandée par les normes du secteur ferroviaire [EN101]. Les méthodes formelles permettent de développer des systèmes sûrs, fiables et robustes. Elles sont ainsi particulièrement pertinentes dans les domaines dits critiques (sécurité, coût financier, vies humaines...). En effet, elles permettent de vérifier de manière prouvée la correction d'un logiciel ou d'un système par rapport à sa spécification. Cependant, cette puissance de vérification s'accompagne de la nécessité d'une forte connaissance de la logique pour en faire bon usage.

Ce second point est un obstacle à une vaste utilisation des méthodes formelles. Une des solutions pour remédier à ce problème est de spécifier le système en se basant sur un modèle graphique et de le traduire ensuite en modèle formel. La spécification résultant de cette traduction peut alors être exploitée pour poursuivre un raisonnement rigoureux sur le même système. L'objectif est qu'à la fin du processus de développement formel le système puisse être certifié comme sûr.

2.2.1 Langages semi-formel ou formel ?

Les langages de modélisation peuvent être classés en trois types :

1. Les langages informels sont basés sur une description du système à modéliser en langage naturel. Si cette représentation est pratique pour la communication, elle n'en reste pas moins imprécise et souvent ambiguë ;
2. Les langages semi-formels sont, en général, des notations graphiques normalisées dotées d'une syntaxe précise (par exemple UML et SADT). Le caractère graphique de ces langages est utile pour les échanges entre les différents acteurs des projets, mais leur manque de sémantique précise ne permet pas de lever les imprécisions et les ambiguïtés ;
3. Les langages formels ont une syntaxe et une sémantiques précises basées sur les mathématiques. Par conséquent, ils sont précis et non ambigües, et permettent d'analyser le respect des propriétés grâce à des preuves mathématiques.

Les sections suivantes décrivent plus en détails les méthodes formelles et leur utilisation dans un processus de conception de systèmes critiques.

2.2.2 Méthodes formelles

Les méthodes formelles répondent à la question : « quelle méthode peut garantir que le système conçu répond bien à ses exigences de criticité ? ». Leur utilisation permet de concevoir des systèmes sûrs. En effet, l'interprétation des spécifications repose sur des méthodes liées aux mathématiques et non sur des choix ou des intuitions humaines. Dans cette section nous fournissons une définition puis une classification des méthodes formelles. Ensuite, nous décrivons comment elles peuvent être utilisées au cours de la conception d'un système, que ce soit pour la spécification, la validation ou la vérification.

2.2.2.1 Définition

Définition 2.2 (Méthode formelle)
Une méthode est dite formelle si elle dispose :

1. *d'un langage : moyen pour concevoir et exprimer un système (symbolique, graphique..) ;*

2. *d'une sémantique : signification mathématique des éléments de ce langage ;*
3. *de règles de validation : possibilité d'exprimer les propriétés désirées du système et d'utiliser la sémantique pour le valider.*

En résumé, une spécification formelle est exprimée dans un langage à syntaxe et à sémantique précises, construit sur une base théorique solide et permettant des vérifications automatisés. De plus, des règles de déduction permettent de démontrer des propriétés de la spécification.

Un projet formel est structuré par un ensemble de relations entre les spécifications, et certaines de ces relations peuvent être prouvées formellement.

On peut distinguer deux taxonomies applicables aux langages formels :

- d'une part, les langages qui relèvent du « correct par construction », tels que le langage *B*. Le modèle évolue par une série de transformations garantissant à chaque étape la préservation des propriétés du système.
- d'autre part, les langages qui impliquent une vérification du modèle a posteriori et de manière systématique après chaque étape du cycle de développement.

2.2.2.2 Classification

Les méthodes formelles peuvent être classées en quatre catégories : les méthodes algébriques, logiques, dynamiques et ensemblistes [Meyer01].

- Les méthodes logiques sont basées essentiellement sur la théorie des types et des logiques d'ordre supérieur. Les théories de démonstration automatiques ou semi-automatiques de théorèmes sont utilisées pour la démonstration de programme. *Coq* est un assistant de démonstration basé sur ce type d'approche, constitué d'un langage de spécifications et d'un système de preuves [Huet et al.01],
- Les méthodes dynamiques s'intéressent plus particulièrement aux interactions entre les processus. On trouve ce type de spécifications dans les systèmes de transitions (automates, réseaux de Petri), les algèbres de processus (CSP) ou en logique temporelle. La section 3.2 décrit plus particulièrement les réseaux de Petri et leurs extensions temporelles que nous utilisons comme modèle graphique pour la vérification des contraintes temporelles contenues dans les spécifications,

– Les méthodes ensemblistes utilisent des types abstraits pré-définis pour modéliser l'état du système. Chaque opération est décrite par son effet sur l'état du système. La syntaxe des méthodes ensemblistes correspond aux types et aux opérations de ce modèle abstrait ; leur sémantique repose la théorie correspondante à ce modèle abstrait (théorie des ensembles, théorie des types, logique du premier ordre).
VDM [Jones89], Z [Spivey89] et B [Abrial96] sont des exemples de langages de spécification ensemblistes. La section 3.3 décrit la méthode B que nous utilisons dans notre proposition méthodologique.

Toutes ces méthodes sont souvent combinées afin de former des méthodes hybrides pour profiter au mieux de leurs avantages respectifs.

2.2.2.3 Utilisation

Il est nécessaire de disposer de règles formelles permettant de laisser le moins de place possible à l'intuition, à l'ambiguïté et aux erreurs. Il s'agit donc idéalement de vérifications automatiques, sans intervention du raisonnement humain. On utilise les mathématiques pour exprimer la sémantique d'un algorithme ou d'un modèle (ce qu'il signifie) et sa correction (répond-il au besoin pour lequel il est conçu ?).

Les méthodes formelles ont donc été proposées afin d'exprimer le besoin par un formalisme rigoureux permettant de :

– définir correctement les termes employés pour limiter les incompréhensions entre développeurs et utilisateurs (Spécification) ;
– faciliter la preuve de propriétés et donc la validation des logiciels produits (Validation et Vérification).

Utilisation des méthodes formelles pour la spécification : Comme nous l'avons montré au chapitre 1, la spécification est une étape cruciale du processus de développement. Des erreurs de spécification sont fréquentes et souvent très couteuses à corriger [Boehm82]. Une grande part de ce problème peut être résolu grâce à l'utilisation de méthodes formelles dont la base mathématique permet d'éviter les ambiguïtés lors de l'interprétation du langage.

Utilisation des méthodes formelles pour la vérification et la validation : La vérification de propriétés contenues dans les spécifications au moyen de méthodes formelles peut se faire en utilisant des techniques de preuve ou d'évaluation de modèles (*model-checking*). Ces techniques sont présentées plus précisément dans la section 2.2.3.

2.2.3 Validation, vérification et certification

Dans le cadre du développement de systèmes complexes et critiques, la notion de *conception sûre* est fondamentale. Il est donc nécessaire d'avoir la certitude de construire le bon système (validation) et celle de bien construire le système (vérification). La validation et la vérification ont ainsi pour objectif de s'assurer que les modèles et le système développés correspondent aux besoins exprimés par les utilisateurs dans la spécification des exigences.

2.2.3.1 La validation

Le processus de validation (définition 2.3) permet d'analyser les propriétés de complétude et de pertinence du modèle par rapport aux exigences contenues dans le cahiers de charges.

Définition 2.3 (Validation)
La validation est la confirmation que les exigences particulières pour un usage spécifique sont satisfaites. Elle répond à la question : « Construit-on le bon modèle ? »[ISO95].

Cette validation s'effectue principalement au moyen de tests à la fin du développement pour s'assurer que le système répond bien aux spécifications des besoins.

2.2.3.2 La vérification

Il est donc fondamental de garantir une sûreté de fonctionnement des systèmes critiques. Il est notamment nécessaire de vérifier si le système réagit de manière attendue à son environnement. Pour ce faire, il faut exprimer clairement le comportement souhaité du système. Cependant, les erreurs d'un système trouvent souvent leur origine du fait de la rédaction des spécifications

en langage naturel. Il est par conséquent très difficile de garantir la correction du système par rapport aux spécifications initiales.

Définition 2.4 (Vérification)
La vérification est la confirmation prouvée que les exigences spécifiées ont été satisfaites. Elle répond à la question : « Construit-on correctement le modèle ? » [ISO95].

Les méthodes formelles englobent trois types de démarche pour la vérification des systèmes critiques : les tests, la vérification de modèles et la preuve assistée.

Simulation/Tests La première mesure applicable pour tenter de vérifier la correction d'un système est d'effectuer des tests sur le comportement du système. Les méthodes habituellement utilisées lors du développement de systèmes consiste à tester un ensemble de scénarios correspondant au cadre d'utilisation du système. Néanmoins, ces méthodes ne permettent pas de tester toutes les situations possibles puisque généralement l'ensemble des scénarios est infini. Ainsi, les tests permettent de montrer l'existence de certaines erreurs, mais ne permettent pas d'en démontrer l'absence.

Model-checking/Vérification de modèles La vérification de modèle repose d'abord sur la construction d'un modèle de comportement du système étudié, ensuite sur la formalisation de la propriété attendue et enfin sur la construction d'algorithmes d'exploration de l'espace d'états pour vérifier que le modèle satisfait bien la propriété.

Preuve automatique Elle consiste à prouver qu'une propriété est vérifiée en utilisant des règles de déduction logique.

Cette approche repose la plupart du temps sur la logique du premier ordre et des formalismes ensemblistes. Pratiquement, des méthodes formelles de développement telles que *Z* [Spivey89] ou *B* [Abrial96] possèdent des assistants de preuves. La méthode *B* offre même un mécanisme formel de raffinement des modèles permettant d'assurer la conservation des propriétés préalablement prouvées.

Bilan sur la vérification Aucune des méthodes présentées n'est suffisante pour garantir le bon fonctionnement d'un système. Ainsi une vérification doit être associée à des tests classiques permettant de connaître le comportement du système dans des conditions réelles. En effet, on peut souligner que le simple fait de modéliser un système rend difficile son analyse dans un environnement réel. En fait, les méthodes présentées dans cette partie sont complémentaires. Il s'agit de trouver un compromis acceptable entre les différentes méthodes de vérification en fonction du type de système étudié et de contraintes à vérifier.

2.2.3.3 La certification

Nous l'avons vu dans la section 1.2.5, l'utilisation de techniques formelles dans le développement de systèmes critiques permet de valider et de vérifier leur processus de conception. L'objectif étant qu'à la fin de ce processus de développement formel, le système puisse être certifié comme sécuritaire. Notre travail a pour objectif de fournir des méthodes et des outils efficaces pour aider à la certification des systèmes complexes soumis à de fortes contraintes temporelles.

2.3 Ingénierie dirigée par les modèles

Nous avons décrit dans la première partie de ce chapitre l'ingénierie des exigences et son intérêt dans un contexte de conception de systèmes critiques. Dans cette seconde partie, nous continuons la description des méthodes dont nous avons justifié l'utilisation dans notre approche méthodologique au chapitre 1 en présentant l'ingénierie dirigée par les modèles.

L'apparition de l'ingénierie dirigée par les modèles (IDM) constitue un changement dans la façon de penser et de concevoir une application. Ainsi il convient de séparer les connaissances et les outils techniques disponibles de la partie « métier », comprenant les savoir-faire d'une entreprise ou d'un laboratoire dans un domaine particulier, afin de les capitaliser, puisqu'ils sont souvent plus stables.

L'ingénierie dirigée par les modèles comprend donc un ensemble de besoins à définir : la prise en compte des aspects fonctionnels et non-fonctionnels

du système, l'intégration de différents points de vue dans tout le processus de conception, l'utilisation des technologies de génération de code.

2.3.1 Concepts de base de l'IDM :

Les concepts fondamentaux de l'ingénierie dirigée par les modèles sont les modèles et les méta-modèles ; de même, la transformation de modèle en est le mécanisme principal.

Définition 2.5 (modèle)
Un modèle est une abstraction d'un système modélisé construite dans une intention particulière. Un modèle doit pouvoir être utilisé pour répondre à des questions sur le système modélisé.

Définition 2.6 (métamodèle)
Un méta-modèle est une spécification explicite d'une abstraction. Cette spécification permet d'identifier l'ensemble des concepts utilisés pour exprimer des modèles ainsi que les différentes relations entre ces concepts. Il définit donc la terminologie à adopter pour définir les modèles.

La création d'un métamodèle dépend aussi de l'utilisation que l'on veut en faire. Par exemple, dans le cadre qui nous intéresse d'une transformation de modèle exogène, le métamodèle représente les différents concepts du formalisme de façon plus ou moins complète : seuls quelques concepts peuvent être nécessaires. Par contre, si l'objectif est par exemple de concevoir un éditeur de modèle, il devra être le plus complet possible.

2.3.2 Règles de transformation

Une règle de transformation définit la manière dont un ensemble de concepts du métamodèle source et transformé en un ensemble de concepts du métamodèle destination.

L'exécution de l'ensemble de ces règles doit permettre de construire l'ensemble du modèle de sortie. Le processus de transformation se compose de trois phases :

1. la vérification de la condition : analyse du modèle source pour détecter la présence d'un ensemble de concepts correspondant à ce que la règle attend en entrée ;

2. L'exécution : les concepts validant la condition sont transformés en accord avec la règle ;
3. la création : un ensemble de concepts sont générés dans le modèle de sortie, dont les champs sont remplis par les variables affectées lors de l'exécution.

Il s'agit de décrire les règles de transformation permettant de passer d'un modèle source réseau de Petri à un modèle cible machine B. Tout d'abord, la traduction de la structure du réseau de Petri est décrite. Par exemple, le marquage de chaque place du réseau correspond à une variable d'état dans la machine B. De même, chaque transition du réseau sera transformée en une opération B.

2.3.3 Hiérarchisation de modélisation :

Un système est donc représenté par un modèle, lui-même conforme à un métamodèle. De plus, le concept de métamodèle doit permettre l'interopérabilité des modèles en facilitant la transformation entre deux langages. Ainsi, est défini un méta-métamodèle qui permet d'utiliser les mêmes références conceptuelles pour vérifier la conformité des différents outils.

Par conséquent, l'ingénierie dirigée par les modèles se fonde sur quatre niveaux de modélisation. Cette architecture à quatre niveaux constitue un des standards de l'IDM imposés par l'OMG. La figure 2.2 illustre la hiérarchisation de ces niveaux de modélisation pour les outils utilisés dans cette thèse, les réseaux de Petri et la méthode B.

2.4 Bilan : méthodes et techniques de la conception système

Nous avons donc expliqué en quoi la notion était devenue fondamentale pour la conception de systèmes, d'autant plus si ces derniers ont un caractère critique.

De plus, il est nécessaire de gérer ces exigences tout au long du processus de conception. L'ingénierie des exigences décrite en section 2.1 nous fournit un cadre pour traiter cette problématique. Elle implique un besoin de formalisation de ces exigences, afin de permettre leur analyse et leur traçabilité.

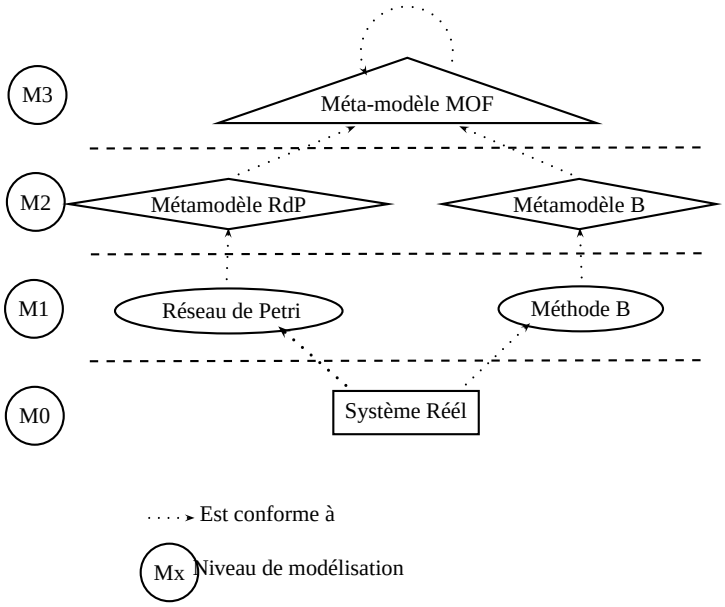

FIGURE 2.2 – Hiérarchie de modélisation

Nous avons donc besoin de langages de modélisation et d'un processus (d'une méthode). Le langage de modélisation a besoin d'une notation claire et d'une sémantique précise, il doit être générique, expressif, flexible et posséder une syntaxe et une sémantique bien définis. Nous avons décrit dans la section 2.2 différentes méthodes formelles, en précisant dans quelle mesure elles sont pertinentes pour la validation, la vérification et la certification de systèmes automatisés.

Enfin, nous avons évoqué l'ingénierie dirigée par les modèles qui nous a inspiré pour l'utilisation de différents modèles complémentaires.

Le chapitre suivant présente donc les différents modèles autour desquels nous avons décidé de construire notre méthodologie de conception, particulièrement axée sur la modélisation et la vérification des contraintes de temps de séjour.

Chapitre 3

Modèles utilisés pour l'analyse des systèmes critiques

Sommaire

Nous avons démontré dans le chapitre 1 la nécessité de l'utilisation des méthodes formelles pour la spécification des systèmes complexes critiques. Plus particulièrement, nous justifions du choix des outils réseau de Petri et méthode *B* comme modèles tout au long du processus de conception. Ainsi, nous expliquons les raisons qui nous ont amenés à utiliser différents modèles afin de profiter de leurs avantages respectifs, en réduisant de la même manières leurs « défauts ». Ces modèles sont décrits dans ce chapitre. De plus, nous nous focalisons particulièrement sur les modèles pertinents dans notre cadre de travail, à savoir la modélisation et la validation des exigences temporelles critiques.

3.1 Introduction sur les langages formels : Choix de deux modèles complémentaires

Comme le définit le chapitre précédent, un langage de spécification est dit *formel* lorsque ses notations et sa syntaxe sont rigoureusement définis par une sémantique mathématique précise. Il existe de nombreux langages réunissant ces caractéristiques. Le choix dépend du contexte applicatif, du problème traité, des habitudes et des connaissances du modeleur, des objectifs que l'on souhaite atteindre et des caractéristiques que l'on souhaite vérifier principalement.

Il est important de rappeler à ce stade que nous avons choisi de représenter les systèmes de transport étudiés comme des systèmes à événements discrets. De plus, nous nous intéressons à des systèmes présentant des contraintes temporelles fortes. Ainsi, nous nous intéressons aux exigences temporelles contenues dans le cahiers des charges de systèmes critiques. Il s'agit de trouver le meilleur compromis entre la puissance de preuve, de validation et de vérification des modèles et leur capacité à traiter la dynamique et les contraintes temporelles que contiennent les systèmes à modéliser.

Dans ce sens, les langages de spécification formels peuvent être classés en deux types : les langages basés sur les *états* et les langages basés sur les *événements*. Les langages basés sur les états se composent de deux modules : la partie statique modélise les entités et les états constituant le système, alors que la partie dynamique représente les changements des ces états. Le langage *Z* [Spivey89] , VDM [Jones89] ou le langage *B* [Abrial96] sont des exemples

de langages basés sur les états.

Les approches basées sur les événements sont capables de modéliser le comportement d'un système grâce à des séquences ou des arbres d'événements. Ainsi, les langages CSP [Hoare et al.85], LOTOS [Van eijk et al.89] ou les réseaux de Petri [Petri62] sont basés sur les événements.

Par conséquent, nous choisissons d'utiliser deux outils qui nous semblent complémentaires.

Dans un premier temps, la méthode B permet par une suite de raffinements successifs d'obtenir un modèle implémentable prouvé et sûr par construction. Cette capacité à garantir la correction de la conception par rapport à la spécification est fondamentale pour l'implantation de systèmes où l'aspect sécuritaire est critique. Cependant, la méthode B éprouve des difficultés à prendre en compte le comportement des événements. Pour répondre à ces difficultés, le B-événementiel a été développé [Abrial00].

De même, on trouve dans la littérature des approches collaboratives visant à profiter des avantages complémentaires de différents méthodes formelles (UML/B [Boulanger et al.06]). Dans cette même optique, il nous parait pertinent de proposer une approche associant à la méthode B un autre modèle capable de traiter plus efficacement la dynamique des systèmes étudiés et permettant de présenter de manière graphique l'ensemble du problème traité.

Ainsi, dans un second temps, nous proposons d'utiliser l'outil réseau de Petri. Les réseaux de Petri sont un formalisme graphique permettant de saisir le fonctionnement global des systèmes étudiés et donc constituent un outil de communication utile aux différents acteurs intervenant dans le processus de conception. Contrairement à d'autres formalismes de types automates par exemple, ils permettent une représentation concise qui diminue les phénomènes d'explosion combinatoire du nombre d'états. Il nous semble important de souligner ici que de notre point de vue, les automates et les réseaux de Petri sont également des modèles complémentaires. Nous proposons d'ailleurs une méthode d'analyse des réseaux de Petri p-temporels en utilisant des automates temporisés dans la section 4.2.2.

De plus, il sont un outil pertinent pour la représentation de la gestion de ressources, du non-déterminisme et des phénomènes de synchronisation et de parallélisme. De nombreux résultats permettant leur analyse peuvent

être trouvés dans la littérature [Murata89]. Cet outil est particulièrement puissant pour l'étude dynamique des systèmes. Enfin, certaines de leurs extensions permettent de traiter efficacement la présence de contraintes temporelles dans les spécifications du système. Par ailleurs, le standard IEC 60300-3-1 (dependability management) liste explicitement les réseaux de Petri comme outil d'analyse de sûreté et du risque.

Ces deux modèles sont de plus déjà utilisés dans le développement de grands projets industriels et particulièrement ferroviaires [Antoni et al.08, Behm et al.99].

Nous utilisons ces deux modèles et nous proposons de les inclure dans une démarche instrumentée globale, comportant des étapes successives clairement identifiées. Cette approche propose notamment de transformer des réseaux de Petri temporels en machines abstraites B décrivant les propriétés structurelles et comportementales du réseau de Petri source dans son intégralité. Le chapitre 6 décrit la manière dont nous procédons pour transformer notre modèle de processus en réseau de Petri t-temporel en une machine abstraite B.

La section 3.2 présente les réseaux de Petri et ses extensions temporelles, ainsi que leurs méthodes d'analyse. Nous y justifions également du choix des différents modèles en fonction des étapes de modélisations au cours du processus de développement. Ensuite, la section 3.3 décrit la méthode B et ses principaux fondements que sont, par exemple, les machines abstraites et le raffinement. Enfin, la section 3.4 démontre la complémentarité des modèles choisis et comment ils peuvent s'intégrer dans une démarche de conception.

3.2 Les réseaux de Petri

Les réseaux de Petri, introduits dans la thèse de Carl Adam Petri en 1962, [Petri62] sont des outils graphiques et mathématiques permettant de modéliser le comportement dynamique des systèmes à événements discrets comme les systèmes manufacturiers, les systèmes de télécommunications, les réseaux de transport. D'un côté, leur représentation graphique permet de visualiser d'une manière naturelle le parallélisme, la synchronisation, le partage des ressources, les choix. De l'autre, leur représentation mathématique permet d'établir les équations d'état, à partir desquelles il est possible d'évaluer les

propriétés du modèle et de les comparer avec le comportement du système modélisé [Murata89].

Un critère important qui peut conduire au choix des réseaux de Petri est le besoin de pouvoir établir des preuves formelles de propriétés (vivacité ou atteignabilité par exemple) devant absolument être satisfaites par l'application. La possibilité d'effectuer des simulations permet en outre de mesurer les performances que l'on peut espérer du système. Enfin, la représentation graphique d'un modèle et les règles de fonctionnement en font un outil essentiel pour la communication entre les différents acteurs du projet. Les paragraphes suivants proposent quelques propriétés pouvant être démontrées grâce à cet outil, ainsi que les méthodes d'analyse permettant de les vérifier.

La première partie de cette section est consacrée aux réseaux de Petri élémentaires. Leur définition et leurs propriétés dynamiques et structurelles sont rappelées. Puis, différentes extensions de ces réseaux prenant en compte le facteur temps et susceptibles de modéliser notre problématique sont présentées.

3.2.1 Le modèle élémentaire

Les réseaux de Petri sont utilisés afin de modéliser le comportement dynamique de systèmes discrets. Un réseau de Petri est un graphe orienté, pondéré et biparti, c'est à dire comprenant deux types de noeuds : les places et les transitions. Les arcs peuvent relier une place à une transition ou une transition à une place.

Définition 3.1 (Réseau de Petri)
Un réseau de Petri est un quadruplet $R =< P, T, Pre, Post >$ *avec :*

- P *est un ensemble fini de places ;*
- T *est un ensemble fini de transitions ;*
- $Pre : P \times T \longrightarrow N$ *est l'application places précédentes ;*
- $Post : P \times T \longrightarrow N$ *est l'application places suivantes ;*

Graphiquement, les transitions sont représentées par des barres (ou des rectangles) et les places par des cercles (figure 3.1).

Les arcs peuvent être annotés par un entier positif k, on dira alors que l'arc est pondéré. Un tel arc peut être interprété comme un ensemble de k-arcs

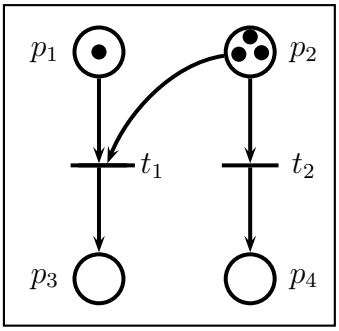

FIGURE 3.1 – Exemple de réseau de Petri

parallèles. Lorsque l'arc est annoté par 1 (annotation unitaire), l'annotation est généralement omise.

Un marquage affecte à chaque place p_i un entier k positif ou nul. Si k est strictement positif, on dit que la place p_i est marqué avec k jetons. Graphiquement, le marquage est représenté par des ronds noirs (figure 3.1).

Définition 3.2 (Réseau de Petri marqué)
Un réseau de Petri marqué est un couple $N =< R, M_0 >$ *avec :*

- R *est un réseau de Petri ;*
- $M_0 : P \longrightarrow N$ *est l'application marquage initial.*

$M(p)$ est le nombre de jetons contenus dans la place p.

Un marquage M est caractérisé par un vecteur de rangs m, où m est le nombre total de places. La $n^{\text{ième}}$ composante du vecteur M représente le nombre de jetons de la $n^{\text{ième}}$ place du réseau. À un instant donné, le marquage définit l'état du réseau, ou plus précisément l'état du système décrit par le réseau.

L'ensemble des entrées d'une transition t_j est l'ensemble, noté ${}^{\cdot}t_j$, de toutes les places p_i qui ont un arc reliant p_i à t_j. On parle aussi de prédécesseurs. L'ensemble des sorties d'une transition t_j est l'ensemble, noté $t_j^{\cdot}$, de toutes les places p_i qui reçoivent un arc de t_j vers p_i. On parle aussi de successeurs.

Formellement, on a :

Définition 3.3 (entrées/sorties d'une transition)
Soit un réseau $RP = (P, T, A, W, M_0)$. *On a pour* $t \in T$ *:*

– ${}^{\cdot}t = \{p|(p,t) \in A\}$, *l'ensemble des prédécesseurs de t,*
– $t^{\cdot} = \{p|(t,p) \in A\}$, *l'ensemble des successeurs de t.*

Pour les places on a une définition analogue :

Définition 3.4 (entrées/sorties d'une place)
Soit un réseau $RP = (P, T, A, W, M_0)$. *On a pour* $p \in P$:
– ${}^{\cdot}p = \{t|(t,p) \in A\}$, *l'ensemble des prédécesseurs de p,*
– $p^{\cdot} = \{t|(p,t) \in A\}$, *l'ensemble des successeurs de p.*

3.2.1.1 Règles de fonctionnement

Définition 3.5 (Transition franchissable)
Une transition t est franchissable si et seulement si :

$$\forall p \in P, M(p) \geqslant Pre(p,t)$$

La mise à feu d'une transition t a pour conséquences :
– de retirer $Pre(p,t)$ jeton(s) de chaque place appartenant à ${}^{\cdot}t$,
– de rajouter $Post(p,t)$ jeton(s) de chaque place appartenant à $t^{\cdot}$.

De cette manière, si t est franchissable pour le marquage M, le franchissement de t donne le marquage M' tel que :

$$\forall p \in P, M'(p) = M(p) - Pre(p,t) + Post(p,t)$$

La figure 3.2 illustre le tir de la transition t_1. On remarque qu'un jeton a été retiré des places p_1 et p_2 et qu'un jeton a été ajouté à la place p_3.

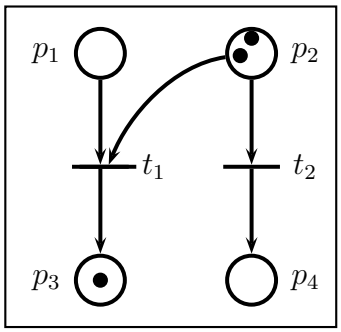

FIGURE 3.2 – Exemple de tir de la transition t_1 dans la figure 3.1

On peut noter qu'une transition sans place d'entrée est appelée transition *source* et qu'une transition sans place de sortie est appelée transition *puits*.

Une transition *source* est toujours validée et le tir d'une transition *puits* consomme des jetons mais n'en produit aucun.

On note $A(R, M_0)$ l'ensemble des marquages accessibles d'un réseau de Petri à partir du marquage initial M_0.

Définition 3.6 (Matrice d'incidence)
La matrice d'incidence d'un réseau de Petri est appelée C, telle que :

$$C = Post - Pre$$

Une colonne de cette matrice correspond à la modification du marquage apportée par le franchissement de la transition correspondante.

Soit une séquence de franchissement s, représentant un ensemble de transitions franchissables successivement, partant d'un marquage initial M_0 et aboutissant au marquage M ; l'équation fondamentale du réseau s'écrit :

$$M = M_0 + C.S$$

avec S le vecteur caractéristique correspondant à la séquence de franchissement s.

Cette équation fondamentale donne le marquage final en fonction du marquage initiale et de la séquence de franchissement. Cependant, elle ne vérifie pas si le franchissement d'une séquence donnée est possible ; il ne s'agit que d'une condition nécessaire d'accessibilité.

3.2.1.2 Propriétés des réseaux de Petri

Elles peuvent être classées en deux groupes [Murata89] :

1. les propriétés structurelles : elles sont indépendantes du marquage initial et sont liées à la topologie du réseau. Leur analyse repose essentiellement sur les techniques de l'algèbre linéaire et leur but est de bâtir une passerelle entre la structure du réseau étudié et son comportement,
2. les propriétés dynamiques : elles dépendent du marquage initial du système et sont liées à l'évolution du réseau. Leur vérification nécessite bien souvent la construction du graphe des marquages accessibles. Elles permettent par exemple d'apporter des réponses aux questions concernant l'accessibilité d'un marquage particulier, le caractère borné, la vivacité qui sont décrites par la suite.

Places et réseaux bornés :

Définition 3.7 (Place bornée)
Une place p est k-bornée (pour un marquage initial M_0) si et seulement si :

$$\forall M' \in A(R, M_0), M'(p) \leqslant k$$

Si $k = 1$, la place est dite *binaire*.

Définition 3.8 (Réseau borné)
Un réseau marqué N est k-borné (resp. binaire ou sauf) si et seulement si toutes ses places sont k-bornées (resp. binaires).

Lorsque le réseau n'est pas borné, le nombre d'états susceptibles d'être atteints est infini. Un de nos objectifs est de vérifier le respect des contraintes temporelles tout au long du processus de conception. Ce problème n'est calculable qu'en présence de résultats sur le caractère borné du réseau de Petri étudié.

Vivacité :

Définition 3.9 (Réseau vivant)
Un réseau marqué est dit vivant si et seulement si pour tout marquage accessible à partir de M et pour toute transition t, il existe une séquence de franchissement contenant t.

Une situation de blocage est telle qu'aucune transition n'est validée. Une telle situation signifie souvent qu'une opération initialement prévue ne peut être effectuée, le système ne peut plus évoluer. La vivacité est donc une qualité souhaitable.

Conflits :

Définition 3.10 (Conflit structurel)
Un conflit structurel correspond à un ensemble de transitions qui ont au moins une place d'entrée en commun.

Définition 3.11 (Conflit effectif)
un conflit effectif correspond à l'existence d'un conflit structurel et d'un marquage tel que le nombre de places du conflit est inférieur au nombre de marques nécessaires au franchissement de chaque transition de sortie validée.

Graphe d'événements : Nous donnons ici les définitions du graphe d'événements (définition 3.12) et du graphe d'événements fortement connexe (définition 3.13), qui sont des types particuliers de réseau de Petri, dont nous utiliserons les caractéristiques par la suite.

Définition 3.12 (Graphe d'événements)
Un Graphe d'événements est un réseau de Petri dans lequel chaque place a exactement une transition d'entrée et une transition de sortie. Il est caractérisé par l'absence de conflits structurels.

Définition 3.13 (Graphe d'événements fortement connexe)
Un Graphe d'événements est Fortement Connexe si et seulement si il existe un chemin orienté qui relie tout sommet (place ou transition) à tout autre sommet.

3.2.1.3 Analyse des propriétés d'un réseau de Petri :

Pour vérifier qu'un réseau de Petri possède les propriétés présentées dans la section 3.2.1.2, il existe trois méthodes principales :

- l'analyse par énumération (Arbre de couverture et énumération de marquages) ;
- l'analyse structurelle (Calcul des composantes) ;
- l'analyse par transformation (Réduction).

Analyse énumérative : L'analyse énumérative permet la validation d'un réseau de Petri. Elle consiste à établir le graphe des marquages accessibles du réseau. Ce graphe d'accessibilité se compose de nœuds, correspondant aux marquages accessibles, et d'arcs, correspondant aux franchissements des transitions permettant le passage d'un état à un autre.

Dans le cas d'un réseau borné, le graphe d'accessibilité est donc fini, et les propriétés facilement vérifiées. Par contre, dans les cas d'un réseau non borné, le graphe sera infini et impossible à construire. On utilise alors un autre graphe appelé graphe de couverture.

Cette méthode est donc parfois difficile, voire impossible à appliquer en raison de l'explosion combinatoire du nombre d'états.

Analyse structurelle : L'analyse structurelle se base sur l'algèbre linéaire puisqu'on se base sur l'équation d'état permettant d'utiliser des algorithmes étudiant les propriétés du réseau de Petri. Ces méthodes reposent sur la recherche d'invariants, qui permettent de caractériser certaines propriétés des marquages atteignables et des transitions franchissables, quelle que soit l'évolution du marquage.

Il existe deux types d'invariants :

- les p-invariants, également appelés p-semiflots ou composantes conservatives ;
- les t-invariants, également appelés t-semiflots ou séquences répétitives. Ils permettent de repérer les comportements cycliques ; ces séquences de franchissement permettent de revenir au marquage initial lorsqu'elles sont appliquées à celui-ci.

Analyse par transformation : Dans le cas de réseaux de Petri de grande taille, l'analyse des propriétés peut devenir difficile à cause de l'explosion combinatoire du nombre d'états. Il peut ainsi être judicieux de diminuer la taille du réseau de Petri marqué initial afin d'étudier une propriété particulière au moyen de règles de réduction. Dans ce cas, le réseau de Petri réduit ne peut s'interpréter comme le modèle d'un système ; il ne sert qu'à l'étude d'une propriété particulière.

3.2.2 Les réseaux de Petri et le temps

Le modèle de base des réseaux de Petri permet donc d'étudier les propriétés logiques des systèmes, c'est-à-dire les propriétés qui dépendent de l'ordre des événements, mais sans faire référence aux instants où ils arrivent.

Cependant, le facteur temps est une composante primordiale. Il n'affecte pas seulement les performances du système mais aussi sa validité fonctionnelle. Le procédé peut se retrouver dans un état interdit si un résultat nécessaire à sa bonne évolution est produit trop tôt ou trop tard. Pour ce type de systèmes, il s'avère nécessaire de disposer de structures de commande leur assurant un bon comportement (au sens du respect du cahier des charges) et attestant du respect de l'ensemble des contraintes temporelles mises en jeu.

Ainsi, lorsque la commande du réseau de Petri autonome est définitivement résolue, des problèmes concernant la gestion du temps peuvent appa-

raitre : l'évaluation de performance, la modélisation des contraintes temporelles, la synthèse d'une commande temporelle (c'est-à-dire le réglage des instants de tirs des transitions), la vérification du respect des contraintes.

Dans un souci de tenir compte de manière efficiente aussi bien des différentes fonctionnalités du système, que de ces caractéristiques temporelles de manière explicite, des extensions temporelles au modèle réseau de Petri capables de prendre en compte les contraintes temporelles quantitatives ont été proposées dans la littérature. Le temps peut être associé aux transitions, aux places ou aux arcs. Il peut être associé de deux manières : on associe à un type de noeud soit une durée (modèle temporisé), soit un intervalle temporel (modèle temporel). La description et la comparaison précises des différentes extensions temporelles sont faites dans la thèse de M. Boyer [Boyer01].

3.2.2.1 Les réseaux de Petri temporisés

Chronologiquement, ce modèle fut la première extension temporelle de réseaux de Petri. Cet outil permet de représenter les mécanismes temporels associés à un processus avec une logique de description des temps nécessaires aux opérations. Il est performant pour la spécification et la validation des systèmes comprenant des temps minimum. Par exemple, le modèle *t*-temporisé est surtout utilisé dans la cadre de l'évaluation de performances [Ramamoorthy et al.80, Zuberek91].

Le temps peut être associé aux transitions [Ramachandani74] (modèle t-temporisé) ou aux places [Sifakis77] (modèle p-temporisé). [Sifakis80] a montré que tous ces modèles sont équivalents. Nous choisissons de présenter le modèle p-temporisé (définition 3.14).

Définition 3.14 (Réseau de Petri p-temporisé)
Un réseau de Petri p-temporisé est un doublet $< R, D >$ *avec :*

- R *est un réseau de Petri autonome* $< P, T, Pre, Post >$ *avec un marquage initial* M_0 *;*
- D *est une fonction durée minimale de séjour d'une marque dans une place, qui à chaque place fait correspondre un nombre rationnel positif décrivant la durée d'indisponibilité des jetons* ($D : P \rightarrow Q^*$).

La sémantique est que les marques doivent rester dans la place p_i au moins le temps d_i associé à cette place. Pendant d_i la marque est indisponible ; elle

ne participe pas à la validation des transitions. Donc d_i représente :

- la durée d'indisponibilité de la marque pour la validation des transitions ;
- le temps minimum de séjour d'une marque dans une place.

Règles de fonctionnement : On utilisera la notion d'état (définition 3.15) pour caractériser la situation du réseau à un instant donné.

Définition 3.15 (État d'un réseau de Petri temporisé)
Un état est un doublet $< M, I >$ *où :*

- *M est une application de marquage, assignant à chaque place du réseau un certain nombre de marques ;*
- *I est une application de temps d'indisponibilité, assignant à chaque marque k dans la place* p_i *un temps* θ_i^k*, correspondant à la durée qui reste à la marque k pour terminer son temps de séjour minimal dans la place* p_i*.*

Les temporisations associées aux places permettent de prendre en compte les durées opératoires minimales. Par conséquent une transition validée au sens des réseaux de Petri autonomes peut ne pas être obligatoirement franchie. Une transition est franchissable si elle est validée au sens des réseaux de Petri autonomes et si les marques qui la valident sont disponibles.

Conclusion : Ce modèle possède donc des propriétés utiles pour l'évaluation de performances, particulièrement pour le respect des contraintes de temps [Ramamoorthy et al.80, Chretienne83]. Cependant, sans entrer dans les détails des différents modèles vus précédemment, nous pouvons toutefois remarquer qu'une marque peut séjourner indéfiniment dans une place. En effet, si la décision de tirer une transition n'est pas imposée, alors il n'y aura aucune obligation de franchissement de cette dernière. Ainsi, ce type de modèle ne peut garantir qu'une durée de séjour minimum dans une place, et de ce fait, ne permet pas de modéliser efficacement les systèmes à contraintes de temps de séjour, où des durées de séjour minimum et maximum sont imposées.

3.2.2.2 les réseaux de Petri t-temporels

Le modèle de Merlin [Merlin74], communément appelé réseau de Petri t-temporel (défintion 3.16), a été conçu pour l'étude des problèmes de recouvrement pour les protocoles de communication. Dans ce modèle, à chaque transition est attachée une contrainte temporelle de type intervalle, et non plus ponctuelle comme dans le modèle t-temporisé.

Définition 3.16 (réseau de Petri t-temporel)
Un réseau de Petri t-temporel est un doublet $= (R, IS)$ *tel que :*

1. R *est un réseau de Petri marqué*
2. IS *est l'application intervalle statique telle que :*

$$IS : T \rightarrow Q^+ \times (Q^+ \cup \{+\infty\})$$

$$t_i \rightarrow IS(t_i) = [a_i, b_i] \; avec \; a_i \leqslant b_i$$

Une transition doit être sensibilisée pendant le délai minimum a_i avant de pouvoir être tirée et ne peut rester validée au-delà du délai maximum b_i sans être tirée ; a_i est la date statique de tir au plut tôt et b_i est la date de tir au plus tard de t_i (figure 3.3).

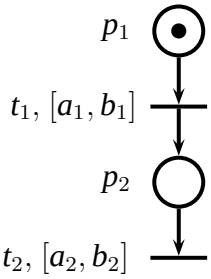

FIGURE 3.3 – Un réseau de Petri t-temporel.

États et règles de tir : Ici encore, la notion d'état est utilisée pour caractériser la situation du réseau t-temporel à un instant donné.

Définition 3.17 (État d'un réseau de Petri t-temporel)
L'état d'un réseau de Petri t-temporel est défini par une paire $E =< M, I >$, *telle que :*

- *M est le marquage du réseau ;*
- *I est une application qui associe à chaque transition du réseau un intervalle de temps pendant lequel elle est franchissable.*

Les intervalles de tir associés aux transitions du réseau dans un état E différent la plupart du temps des intervalles initiaux. L'état initial E_0 est constitué du marquage initial M_0 et de l'application I_0 telle que :

- si $M_0 \geqslant Pre(k)$ alors $I_0(k) = IS(k)$;
- sinon $I_0(k) = \varnothing$.

L'intervalle $[a_i, b_i]$ associé à la transition t_i est relatif au moment où la transition devient validée. Supposons que t_i soit validée à l'instant T, alors elle peut être franchie dans l'intervalle matérialisé par les quantités $(a_i + T)$ et $(b_i + T)$, sauf si elle est désensibilisée à cause du franchissement d'une autre transition avec laquelle elle est en conflit.

D'après les règles de fonctionnement de ce modèle, on ne commence à compter le temps que si la transition est validée. Ainsi une marque peut rester jusqu'à l'infini dans une place en amont d'une transition de synchronisation (jusqu'à la validation de cette transition). C'est la dernière marque qui valide une transition de synchronisation qui fixe l'échéance de tir de cette transition.

Le tir d'une transition t sensibilisée à l'instant τ depuis l'état $E =< M, I >$ conduit à un nouvel état $E' =< M', I' >$ tel que :

- M' est donné par l'équation classique :
$$M'(p) = M(p) - Pre(p,t) + Post(p,t),$$
- I' est défini ainsi :
  - pour les transitions non sensibilisées par M' :
  $$I' \text{ est vide,}$$
  - pour les transitions sensibilisées par M' et pas en conflit avec t :
  $$I' = [max(0, a_k - \tau), b_k - \tau],$$
  - pour les autres transitions :
  $$I' = I,$$

Ainsi, la règle de franchissement, imposant que l'horloge locale associée à chaque transition du modèle ne soit déclenchée qu'à la validation logique de celle-ci, ne permet pas la prise en compte de façon naturelle du phénomène de synchronisation sous obligation.

Néanmoins, le formalisme réseau de Petri t-temporel reste probablement le meilleur outil de spécification des protocoles et de processus temporels.

De nombreux résultats autour de ce formalisme sont disponibles dans la littérature. Le fonctionnement au plus tôt est notamment toujours réalisable. Enfin, une instrumentation logicielle performante permet la modélisation, l'analyse et la simulation grâce à l'outil logiciel TINA [Berthomieu et al.03].

Représentation d'un chien de garde : L'extension t-temporelle des réseaux de Petri a été introduite pour permettre la surveillance d'un système, en partant de l'hypothèse que les risques sont connus a priori. Ces derniers sont modélisés par des mécanismes spécifiques, tel que les chiens de garde (ou watchdog) [Combacau91]. Un chien de garde est très important dans un système temporel : il permet de vérifier qu'une action donnée a bien lieu avant une date donnée. Sinon, il signale une erreur et il est alors possible de corriger le fonctionnement.

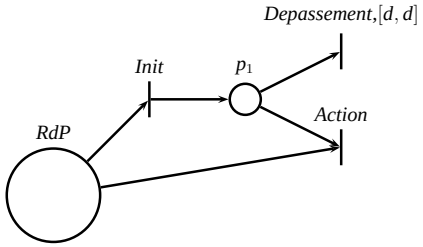

FIGURE 3.4 – Chien de garde modélisé par un réseau t-temporel.

La figure 3.4 illustre comment est modélisé un chien de garde au moyen d'un réseau de Petri t-temporel. La transition *init* envoie un jeton dans la place p_1 qui représente le chien de garde. Si l'*action* se produit, elle consomme ce jeton et le chien de garde est désactivé. Si elle ne se produit pas avant une durée d, l'action *Depassement* a lieu, ce qui correspond à une erreur et nécessite une correction.

Analyse énumérative : Elle consiste à construire un arbre (ou graphe) de couverture pour permettre l'étude des propriétés dynamiques du réseau. Elle se base sur la notion de classes d'états (voir la section 4).

Une classe d'état représente un ensemble d'états, ayant le même marquage, et dont la réunion des intervalles dynamiques constitue le domaine de tir associé à la classe. Il est démontré dans [Berthomieu et al.91] que ce

domaine de tir peut être défini comme l'ensemble des solutions d'un système d'inégalités.

3.2.2.3 Les réseaux de Petri p-temporels

Le modèle de Khansa [Khansa97], communément appelé réseau de Petri p-temporel (définition 3.18) a été développé afin de pouvoir modéliser et analyser les systèmes à contraintes de temps et permettent la prise en compte de manière naturelle de la situation de synchronisation sous obligation [Collart dutilleul et al.98].

Définition 3.18 (réseau de Petri p-temporel)
Un réseau de Petri p-temporel et un doublet $= (R, IS)$ *tel que :*

1. *R est un réseau de Petri marqué*
2. $IS : P \rightarrow Q^+ \times (Q^+ \bigcup \infty)$ *avec* Q^+ *l'ensemble des nombres rationnels positifs.*

$$p_i \rightarrow IS(p_i) = [a_i, b_i] avec\ a_i \leqslant b_i$$

$IS(p_i)$ définit l'intervalle statique associé à la place p_i (voir figure 3.5). La sémantique de cet intervalle est la durée de séjour admissible d'une marque dans cette place : ainsi, un jeton contenu dans la place p_i ne participera à la validation des transitions dont p_i est une place d'entrée que si il a séjourné pendant au moins a_i unités de temps. Au-delà de b_i unités de temps, le jeton sera considéré comme mort, et de ce fait ne participera plus à la validation des transitions. L'état de marque morte est spécifique au modèle des réseaux de Petri p-temporels, de même que la notion de séquence de tir marque-vivante.

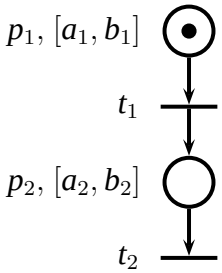

FIGURE 3.5 – Un réseau de Petri p-temporel.

Ce dernier aspect est utilisé pour la spécification des contraintes temporelles. En fait, la mort d'une marque représente un non respect des spécifications et correspond à un état interdit dans la terminologie qui a été définie par [Ramadge et al.87]. Par conséquent, on est amené à contrôler les instants de tir des transitions.

Le fait d'associer les intervalles temporels aux places du réseau permet de conserver le traitement des conflits : toutes les transitions d'un conflit ont la même priorité de franchissement, comme pour les réseaux de Petri temporisés et élémentaires. Mais contrairement aux réseaux de Petri t-temporels, les temporisations n'imposent pas de contraintes sur les franchissements des transitions d'un conflit.

D'autre part, ce modèle impose que les dates d'arrivées des marques dans les places en amont des transitions de synchronisation soient compatibles, si on veut éviter la violation d'une spécification de temps de séjour, soit en d'autres termes, l'apparition de marques mortes dans le réseau.

États d'un réseau de Petri p-temporel : Pour un instant donné, la notion d'état (définition 3.19) est utilisée pour caractériser la situation du réseau p-temporel.

Définition 3.19 (État d'un réseau de Petri p-temporel)
L'état d'un réseau de Petri p-temporel est défini par une paire $E =< M, Q >$, *telle que :*

- M *est le marquage du réseau ;*
- Q *est une application temps de séjour qui associe à chaque jeton* k *présent dans la place* p_i *un nombre réel* q_i^k *correspondant à l'âge de ce jeton, c'est à dire le temps écoulé depuis son arrivée dans la place.*

Notons $[a_i, b_i]$ l'intervalle statique associé à la place p_i. L'âge q_i^k d'un jeton doit alors être supérieur ou égal à a_i pour qu'il puisse participer à la validation des transitions de sortie de p_i. De même, il doit être inférieur ou égale à b_i sinon le jeton sera considéré comme mort.

Condition de tir d'une transition : Une transition t_v est potentiellement tirable à partir de l'état $E =< M, Q >$ si et seulement si :

- elle est validée au sens des réseaux de Petri autonomes,

- $\forall p_i \in {}^{\cdot}t_v$, il existe au moins $Pre(p_i, t_v)$ jetons telles que $\cap[a_i^k, b_i^k] = [a_i^i, b_i^i] \neq \emptyset$ avec $k \in 1, 2, ..., Pre(p_i, t_v)$,
- $\forall p_i \in {}^{\cdot}t_v$, il n'existe pas de jetons ne participant pas au franchissement de t_v dont les bornes supérieures de l'intervalle dynamique soient strictement inférieures à a_i^i.

Sinon le réseau sera marques-mortes.

Enfin, l'intervalle d'intersection de toutes les places d'entrée de t_v, noté $[a_v, b_v]$, doit être non-vide. Cet intervalle représente celui pendant lequel la transition est tirable.

Calcul de l'état suivant : Le passage d'un état $E =< M, Q >$ à un état $E' =< M', Q' >$ peut se faire de deux manières : soit par écoulement du temps, soit par franchissement d'un transition.

L'état $E =< M, Q >$ conduit à un nouvel état $E' < M', q' >$ par écoulement du temps τ si et seulement si :

- $M' = M$,
- $\forall j$ une marque contenue dans la place p_i,

$$q_i^k = q_i^k + \tau \leqslant b_i$$

D'autre part, le tir d'une transition sensibilisée t_v depuis $E =< M, Q >$ conduit à un nouvel état $E' < M', Q' >$ si et seulement si :

- t_v est directement franchissable à partir de E (les âges de toutes les marques qui valident t_v sont supérieurs ou égaux aux bornes inférieures des intervalles statiques associés à leurs places),
- $\forall p \in P, M'(p) = M(p) - Pre(p, t_v) + Post(p, t_v)$,
- les marques qui ne se déplacent pas gardent le même âge ; les marques nouvellement créées prennent les intervalles statiques associés à leur nouvelles places d'accueil (et donc l'âge zéro).

Les deux règles de calcul de l'état suivant précédent définissent les relations d'accessibilité entre différents états d'un réseau de Petri p-temporel. L'ensemble des séquences de tir réalisables depuis l'état initial caractérise le comportement du réseau.

Représentation du parallélisme et des synchronisations : Les réseaux de Petri p-temporels imposent l'obligation de franchissement de toutes

les transitions du réseau. Par conséquent, les instants d'arrivée des jetons dans le places en entrée de transition de synchronisation (par exemple la transition t_6 dans la figure 3.6) doivent être compatibles.

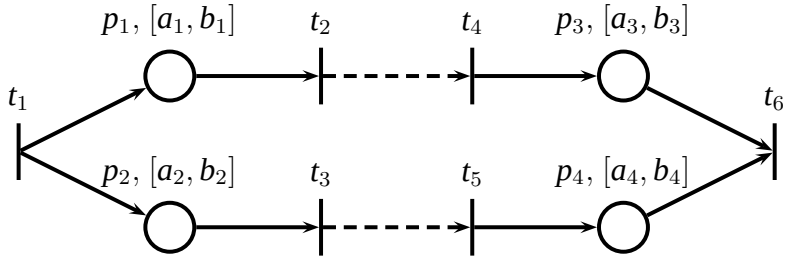

FIGURE 3.6 – Structure parallélisme/synchronisation (p-temporel).

3.2.3 Choix des extensions temporelles en fonction du niveau de modélisation

Dans cette partie, nous avons présenté les outils de modélisation et d'analyse qui ont retenu notre attention pour ce travail de recherche. Il s'agit des extensions temporelles des réseaux Petri, capables de modéliser la plupart des contraintes temporelles intervenant dans le fonctionnement d'un système à événements discrets temporisé.

Le modèle p-temporel fournit une représentation naturelle et concise des synchronisations temporelles. Une des ses limitations est l'impossibilité de la modélisation des chiens de garde. En fait, la recherche de dépassement temporel au moyen de ce modèle se fait par la recherche de jetons morts dans le graphe des états. L'apparition de marques mortes dans le réseau correspond à une violation d'une spécification de temps de séjour. On est ainsi amené à contrôler les instants de tir des transitions. Le respect des temps de séjour implique l'obligation du franchissement d'une transition dont l'échéance fait référence à la première marque qui aura épuisé son temps de séjour maximum. L'étude de cet outil nécessite la prise en compte de l'histoire de marques. En particulier, les dates d'arrivée des marques dans les places en amont des transitions de synchronisation doivent être compatibles (notion de synchronisation sous obligation) ; ce modèle considère d'abord la synchronisation temporelle, et seulement après la synchronisation logique.

L'outil t-temporel offre plus de possibilités pour la modélisation des conflits. Leur interprétation est plus simple par rapport au modèle p-temporel puisque leur évolution n'est déterminée que par un seul type d'événement, le franchissement des transitions, alors que celle du modèle p-temporel dépend également de la mort des marques dans les places. De plus, comme nous l'avons vu précédemment, ils permettent de modéliser le passage du système en mode dégradé par l'intermédiaire d'alarmes (ou *watchdog*).

Finalement, nous avons choisi de construire une méthode s'appuyant sur la complémentarité de ces deux extensions temporelles des réseaux de Petri :

- tout d'abord, nous construisons un modèle des exigences au moyen d'un réseau de Petri p-temporel,
- ensuite, un modèle de processus construit avec un réseau de Petri t-temporel est proposé afin de vérifier si la solution proposée correspond bien aux spécifications et si la traçabilité des exigences est respectée.

La section suivante présente la méthode B, que nous proposons d'utiliser à la fin du processus de modélisation pour une implantation formelle du système étudié. L'idée ici est d'associer les capacités des réseaux de Petri, qui en font un modèle lisible et expressif, avec la puissance des systèmes de preuve de la méthode B. Chaque modèle présenté dans ce chapitre possède des caractéristiques propres, propices à représenter différentes phases du processus de conception.

3.3 La méthode B

La méthode B est une méthode formelle qui permet de passer d'une spécification abstraite à une implantation concrète. Elle englobe le processus complet, de la spécification au code exécutable, permettant de prouver que l'implantation est conforme à la spécification. Elle est basée sur la théorie des ensembles. La machine abstraite est une des notions de base de la méthode B. Elle modélise un système décrit par un ensemble de données ou variables et par les opérations associées qui modifient leur état ou valeur. L'autre notion fondamentale de la méthode B est le processus de raffinements successifs.

3.3.1 Historique et applications industrielles

Le méthode B est une technique utilisée notamment par la RATP : elle semble en effet très adaptée à l' élaboration de logiciels critiques, d'une part grâce à sa bonne couverture du cycle de développement (de la phase de spécification jusqu'à la génération automatique de code), et d'autre part en raison de ses capacités de vérification formelle. Ainsi, elle est utilisée dans le développement d'autres applications industrielles couronnées de succès, notamment dans le domaine ferroviaire. Nous pouvons d'ailleurs citer le succès de l'application de la méthode au système METEOR (ligne automatique 14 du métro parisien [Behm et al.99]).

Par contre, l'intégration de la méthode B dans le développement de logiciels critiques n'est pas évident puisque par exemple, en B, l'élaboration de la spécification formelle et la conception de l'architecture logicielle se font simultanément et sans distinction, ce qui ne respecte pas l'ordre classique des étapes de la construction d'un logiciel.

3.3.2 Fondements

La méthode B est fondée sur deux modélisations complémentaires : une modélisation des aspects statiques, décrivant les entités constituant le système et les états pouvant leur être associés, et une modélisation des aspects dynamiques, décrivant les changements d'états autorisés à l'aide d'actions définis sur ces entités. La méthode B se base principalement sur les notions de machines abstraites et de raffinements qui sont décrites dans la suite de cette section.

3.3.2.1 Machines abstraites

La spécification d'un programme en B prend la forme d'une machine abstraite. Cette spécification mathématique abstraite se base sur la théorie des ensembles et sur la logique du premier ordre pour les propriétés statiques. Les propriétés dynamiques des machines B sont spécifiées en utilisant un langage de substitution. Une machine abstraite B est définie par les clauses de classe suivantes :

- une clause MACHINE : donne un nom à la machine,
- une clause SET : définit les types utilisés dans la spécification,

- une clauses INITIALISATION : représente l'état initial de la machine,
- une clauses VARIABLES : représente l'état de la machine,
- une clause INVARIANT : prédicat définissant les conditions de sécurité de la machine B,
- une clause OPERATIONS : définit les changements d'état possibles pour la machine,
- des clauses INCLUDES et IMPORT : modularité rendant accessibles les entités des machines,

3.3.2.2 Raffinements

Le raffinement permet au concepteur de développer par étapes successives le projet, jusqu'à l'implémentation finale. Il est donc nécessaire d'enrichir et de transformer la représentation abstraite des données et des opérations en une représentation concrète équivalente, pour obtenir un modèle exécutable. Le raffinement est un processus progressif et plusieurs étapes intermédiaires peuvent être nécessaires. De plus, la méthode permet de valider chaque étape par la preuve de certaines conditions.

3.3.3 Extensions : méthode B et temporalité

Cette section présente une extension de B appelée B-événementiel (ou B-système), dans laquelle un modèle représente un système dynamique pris dans son ensemble.

Notre objectif est de spécifier des contraintes dynamiques. L'idée du B-événementiel est de spécifier un projet B « normalement » et d'ajouter des contraintes dynamiques. Ces contraintes dynamiques doivent être compatibles avec les contraintes statiques déjà spécifiées.

Alors qu'en B classique on parle de « machine », ici on parle de « système » ou de « modèle », qui est la notion de base du B-événementiel. Ainsi, un système B possède un état enregistrant l'information pertinente. De plus, les événements (actions activables) ne peuvent s'exécuter que si certaines conditions sont remplies : on dit que les événements sont gardés par des conditions. En fait, un événement s'execute si sa garde est vraie dans l'état courant (événement habilité/activable). Si plusieurs événements sont habilités, alors un événement choisi au hasard s'exécute (pas d'execution en parallèle). Enfin,

un modèle fonctionne tant qu'un de ces événements est habilité. Sinon, soit le calcul est terminé, soit le système est bloqué.

3.4 Conclusion

Dans un premier temps, la section 3.2 a présenté l'outil réseau de Petri et ses extensions temporelles. Ces dernières nous ont semblé complémentaires pour évaluer les contraintes temporelles auxquelles sont soumis certains systèmes.

D'un côté, le modèle p-temporel permet de capturer ces exigences temporelles dans la structure même du modèle. Ce dernier représente un formalisme puissant et reconnu pour la modélisation de l'obligation de respect des temps de séjour. Il permet de modéliser aisément les phénomènes de synchronisation sous obligation. Il ne possède pas de fonctionnement par défaut comme le fonctionnement au plus tôt.

Quant à lui, le modèle t-temporel est principalement destiné à l'étude de protocoles de communication dont les évolutions dépendent des contraintes de type temps de réponse. Il est en effet particulièrement efficace pour la modélisation de chiens de garde. Il reste d'ailleurs probablement le meilleur outil de spécification de protocoles et de processus temporels. Son fonctionnement au plus tôt est toujours réalisable. Sa faiblesse concernant la formalisation des exigences de temps de séjour ne remet ainsi pas du tout en cause la puissance de ce modèle.

Enfin, la méthode B est une méthode déjà utilisée pour spécifier des systèmes ferroviaires complexes. Elle modélise le système au moyen de machines abstraites dont les raffinements successifs aboutissent à un code implémenté et prouvé. Cette méthode nous semble assez difficile à utiliser, en particulier au début du processus de conception, où les échanges entre les différents acteurs du projet sont nombreux. Il parait donc intéressant d'associer les capacités des réseaux de Petri, qui en font un outil lisible et expressif, avec la puissance des systèmes de preuve B. Notre contribution à cette dernière étape n'est pas exhaustive, mais prometteuse. Elle sera décrite au chapitre 6.

Il nous semble donc particulièrement pertinent de proposer une méthodologie combinant les avantages de ces modèles. C'est ce que le chapitre suivant

se propose de faire.

Chapitre 4

Modélisation et validation des exigences temporelles

Sommaire

Introduction

Ce chapitre décrit notre contribution à la conception de systèmes critiques. Notre objectif est de proposer une méthode et des outils pour modéliser et évaluer la sécurité des systèmes à événements discrets temporisés.

Tout d'abord, la section 4.1 expose la méthodologie générale que nous avons élaborée. Elle prend en compte les exigences temporelles tout au long du processus de conception, depuis les spécifications contenues dans les cahiers des charges jusqu'à l'implantation du système final.

Ensuite, la section 4.2 s'intéresse particulièrement à la capture des spécifications temporelles dans un modèle des exigences. Il s'agit également de fournir des outils permettant de s'assurer que les modèles construits vérifient bien ces exigences.

La description de processus de conception se poursuit dans la section 4.3. Un modèle de processus est construit afin de modéliser une solution proposée pour répondre aux spécifications. Ce modèle introduit des choix technologiques ou architecturaux. Nous proposons également une méthode permettant d'assurer la traçabilité des exigences et de vérifier que ce modèle est bien conforme au modèle des exigences, c'est-à-dire qu'il respecte bien toutes les contraintes temporelles.

4.1 Vers une méthodologie de conception prenant en compte les exigences temporelles

La définition d'une méthodologie repose sur plusieurs éléments :

1. un périmètre d'étude : il s'agit de déterminer comment est orientée la méthodologie considérée (modélisation, validation, vérification...),
2. un langage de modélisation, composé d'un cadre sémantique (concept et définition du langage) et d'une syntaxe prédéfinie,
3. un processus de développement méthodologique.

Le cadre de notre travail a été défini dans le chapitre 1. Il s'agit de la conception de systèmes critiques et plus particulièrement la prise en compte des contraintes temporelles contenues dans le cahier des charges au cours du processus de conception.

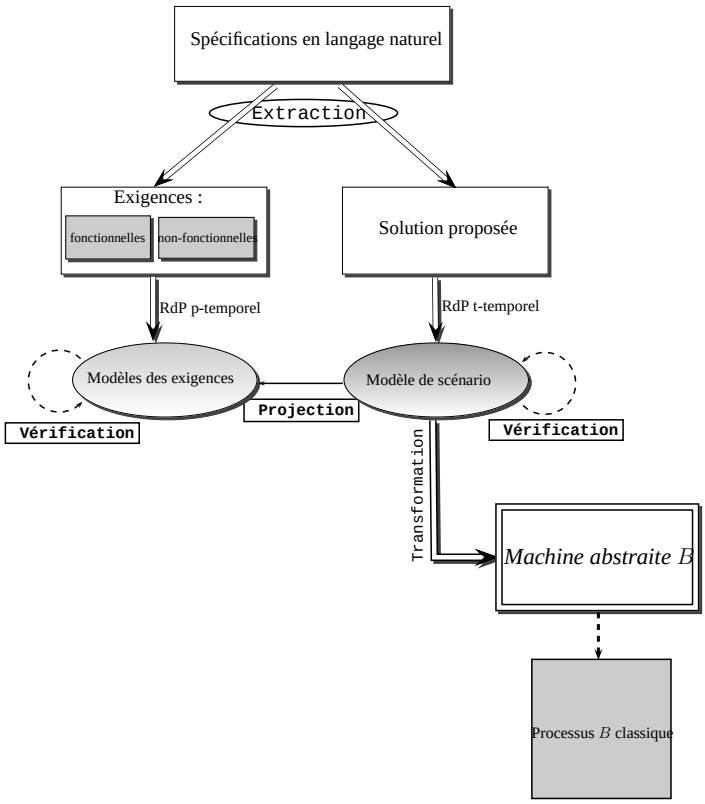

FIGURE 4.1 – Méthodologie générale

Les outils que nous avons choisis pour mettre en œuvre cette méthodologie sont décrits au chapitre 3. Nous proposons d'utiliser les extensions temporelles des réseaux de Petri. Nous sommes convaincus que des modèles graphiques permettent une meilleure compréhension du système et fournissent un support de communication entre les différents acteurs du projet, qu'ils soient clients, concepteurs ou experts du domaine. D'autre part, afin d'évaluer le respect des exigences, il est nécessaire d'utiliser des outils formels capables de prendre en compte les contraintes temporelles et se basant sur des notions mathématiques permettant de vérifier le système.

Ainsi, nous rappelons que les réseaux de Petri p-temporels sont utilisés pour modéliser et vérifier le modèle des exigences, alors que le modèle de processus représentant une solution proposée pour répondre à ces exigences est modélisé au moyen de réseaux de Petri t-temporels. En effet, nous proposons d'utiliser conjointement les deux extensions de réseaux temporels décrits

précédemment dans le cadre d'une démarche d'ingénierie des exigences.

Notre proposition méthodologique (voir figure 4.1) s'appuie dans un premier temps sur la construction d'un modèle des exigences prenant en compte toutes les contraintes que doit vérifier le système. Nous développons des outils et des méthodes capables de valider et de vérifier ce modèle par rapport aux exigences, notamment temporelles, contenues dans le cahier des charges. Ils sont décrits dans la section 4.2. Nous rappelons que notre travail se place dans un cadre restreint puisque l'on se focalise sur la prise en compte des contraintes de temps de séjour, exprimées sous forme d'intervalles. Le modèle des exigences construit au moyen de réseaux de Petri p-temporels permet donc d'étudier une classe bien particulière de contraintes dans un cadre applicatif particulier : il permet de capturer les exigences temporelles dans la structure du modèle.

Dans cette optique, la première partie de ce chapitre présente deux méthodes que nous avons développées pour la vérification du modèle des exigences. Il s'agit de contrôler le réseau de Petri p-temporel modélisant les exigences afin de s'assurer que ces dernières soient bien respectées.

À la fin de cette première étape, nous obtenons un modèle des exigences validé. Notre proposition demande alors la construction d'un modèle de procédé décrivant une solution proposée pour la mise en œuvre du système. Ce modèle illustre des exigences techniques relatives à un choix technologique ou architectural. L'objectif est double : tout d'abord il est nécessaire de vérifier la traçabilité des exigences tout au long du processus de conception ; ensuite, il faut vérifier que l'ensemble des exigences sources est bien implémenté dans la solution préconisée et dans sa mise en œuvre. Une fois le modèle de procédé obtenu, il devient donc essentiel de s'interroger sur sa capacité à satisfaire les spécifications imposées par le cahier des charges. La réponse à cette question est généralement apportée par des techniques de vérification s'appuyant sur une génération de l'espace d'états. Ainsi, la deuxième partie de ce chapitre décrit une méthode basée sur la construction des graphes de classes d'état de nos modèles et leur comparaison en vue de vérifier si les solutions proposées vérifient bien les spécifications. Cette partie s'inscrit également dans une démarche globale d'ingénierie des exigences, notamment en prenant en compte leur traçabilité. Finalement, nous obtenons un modèle de solution qui vérifie les contraintes contenues dans le cahier des charges et qui peut

nous permettre ensuite de passer à la phase d'implémentation du modèle.

Cette démarche se situe sur la phase « descendante » du cycle de conception (voir figure 4.2).

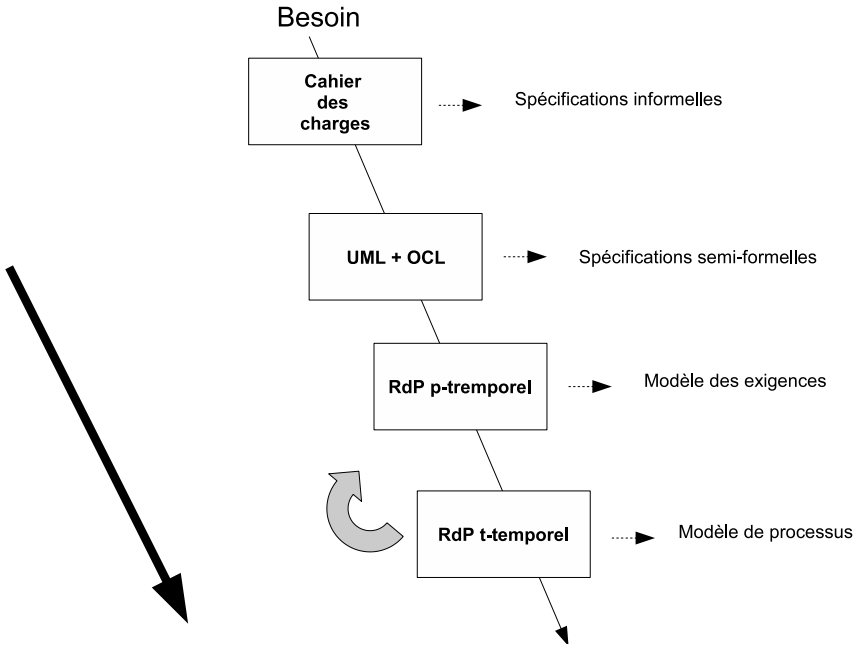

FIGURE 4.2 – Méthodologie de conception.

4.2 Construction et synthèse de contrôle d'un modèle des exigences

Nous proposons une méthodologie pour nous permettre d'analyser le modèle proposé pour la vérification du respect des exigences. Dans le cadre de cette thèse, il s'agit donc de proposer une contribution permettant de contrôler un réseau de Petri *p*-temporel.

Nous nous concentrons exclusivement sur le respect des contraintes de temps, et partant de l'hypothèse que le comportement des réseaux de Petri autonomes sous-jacents est correct.

Dans cette section nous décrivons l'approche que nous proposons pour la synthèse de la commande par supervision de ce modèle. Deux méthodes sont proposées pour répondre à notre problématique : tout d'abord, nous présentons une méthode consistant à transformer le modèle réseau de Petri en automate afin d'en définir la commande. Ensuite nous proposons une autre méthode basée sur l'étude du cographe associé au modèle, pour peu que celui-ci ait un fonctionnement répétitif.

4.2.1 Méthodes de supervision des SED temporisés

Différentes approches existent dans la littérature pour permettre d'étudier le comportement d'un système à événements discrets et de synthétiser une commande qui permette de réaliser un fonctionnement désiré. La synthèse de commande consiste, à partir d'une spécification de la structure et du comportement du système physique à contrôler et des objectifs à atteindre, à spécifier une politique de commande pour ce système et à générer un contrôleur. Un contrôleur est un agent capable d'activer ou désactiver les transitions contrôlables d'un système discret en fonction des occurrences d'événements survenant dans le système physique.

La théorie de la Supervision des systèmes à événements discrets fut initiée par [Ramadge et al.87]. Cependant, ces travaux ne traitent que de manière qualitative du fonctionnement du système. En effet, ils ne s'intéressent qu'à l'occurrence des événements (ou leur impossibilité d'occurrence) et à leur succession. Ainsi cette théorie ne peut traiter de la commande des systèmes à événements discrets et ne prend pas en compte explicitement la contrainte temporelle.

L'apparition du concept d'événements forçables [Charbonnier96] a permis aux superviseurs d'être utilisés comme système de commande, garantissant le respect de spécifications imposées par le cahier des charges.

La prise en compte du temps est également nécessaire pour modéliser des systèmes réels, ce qui augmente la richesse mais également la complexité du modèle. Pour répondre à ce problème, les travaux de [Brandin et al.94] reposent sur un modèle à temps discret (le passage du temps est modélisé comme l'occurrence d'un événement). Cette méthode donne de bons résultats, mais malheureusement le fait qu'elle nécessite de discrétiser le temps provoque une explosion combinatoire du nombre d'états.

Afin de résoudre ce problème, l'outil automate temporisé a été introduit pour l'analyse et la synthèse des systèmes à événements discrets temporisés [Maler et al.95]. Néanmoins, il est incapable de représenter des mécanismes tels que la synchronisation ou le parallélisme par exemple. Par conséquent, des approches associant la capacité d'analyse des automates temporisés à la puissance du formalisme réseaux de Petri ont été proposées. Ces approches modélisent le système à commander par des réseaux de Petri ou leurs extensions (t-temporels [Sava01]), dont le comportement est analysé au moyen d'automates à temps continu, qui permettent également de synthétiser la commande du système.

Un des aspects intéressants de l'approche ci-dessus [Ramadge et al.87] est qu'on a effectué la synthèse d'une commande optimale, au sens où elle restreint au minimum les degrés de liberté du système, tout en respectant les objectifs fixés (c'est là tout le mérite du *maximaly permissive control*). Cette faculté revêt une importance particulière dans un contexte de conception. En effet, en conception, restreindre inutilement le domaine de prospection revient à écarter des solutions potentielles. Cette qualité est louable, cependant contrebalancée par la discrétisation utilisée pour la gestion du temps. On est donc en face d'une solution qui est optimale si le temps est discret, ce qui n'est concrètement pas le cas. Bien plus, nous nous intéressons à des exigences temporelles de sécurité, et dans ce cas, ce qui est critique, c'est le respect effectif de ces exigences de sécurité. L'introduction d'une approximation, sous la forme d'une discrétisation du temps, aussi tôt dans le processus de conception n'est pas acceptable. C'est précisément sur le paramètre critique pour la sécurité, le temps dans ce travail, que des approximations sont effectuées. S'il faut bien reconnaitre les qualités des formalismes mis en œuvre dans [Sava et al.08], nous préconisons, pour la gestion des exigences temporelles de sécurité, d'éviter la discrétisation du temps ou de la retarder au plus tard possible dans le processus. Les sections suivantes proposent des solutions originales allant dans ce sens.

4.2.2 Synthèse de contrôleur par analyse d'un automate associé

La méthode décrite dans cette section se base sur l'utilisation d'un automate pour commander un système sujet à des contraintes de temps de séjour et modélisé par un réseau de Petri p-temporel [Collart dutilleul et al.06, Defossez et al.07]. Cette méthode doit nous permettre d'analyser le modèle proposé et de garantir un bon comportement du système, même en cas de défaillance technique. La synthèse de commande proposée s'attache à éviter que le modèle ne se trouve dans un état interdit en utilisant un automate. L'utilisation de l'outil automate temporisé a été introduit afin de réduire le problème de l'explosion combinatoire du nombre d'états. Cependant, il ne permet pas de représenter des mécanismes tels que la synchronisation ou le parallélisme. Cette méthode consiste à analyser le comportement d'un réseau de Petri p-temporel en le convertissant en automate continu. Pour la construction du comportement de cet automate, il s'agit de considérer toutes les évolutions possibles et de calculer les intervalles de temps de séjour valides. Un algorithme nous permet de définir ces intervalles de temps en vue d'éviter les états interdits du système.

4.2.2.1 Modélisation de contraintes de temps de séjour avec synchronisation

Les transitions en amont des places situées immédiatement devant une transition de synchronisation doivent être tirées dans un temps compatible avec leurs intervalles de temps de séjour comme nous le montre la figure 4.3.

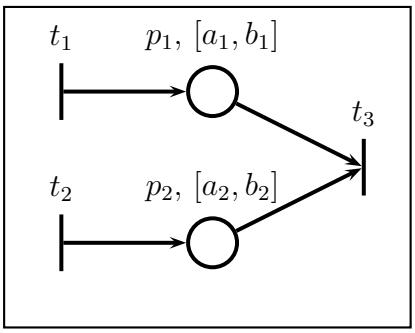

FIGURE 4.3 – Une synchronisation dans un réseau de Petri p-temporel

Cette portion de réseau nous donne les relations suivantes :

$$S^{\circ}p_1(n) + a_1 \leqslant S^{\circ}p_3(n)$$

$$Sp_3{}^{\circ}(n) \leqslant S^{\circ}p_1(n) + b_1$$

$$S^{\circ}p_2(n) + a_2 \leqslant S^{\circ}p_3(n)$$

$$Sp_3{}^{\circ}(n) \leqslant S^{\circ}p_2(n) + b_2$$

Ce qui nous conduit aux deux relations :

$$Max\{S^{\circ}p_1(n) + a_1; S^{\circ}p_2(n) + a_2 \ \leqslant S^{\circ}p_3(n)\}$$

$$S^{\circ}p_3(n) \leqslant min\{S^{\circ}p_1(n) + b_1; S^{\circ}p_2(n) + b_2\}$$

La première relation correspond à la $n^{ème}$ date de validation au plus tôt des transitions p_i, la seconde à celle au plus tard. Il est évident que lorsque l'instant de tir au plus tôt est plus grand que celui au plus tard, ces conditions ne peuvent pas être respectées.

4.2.2.2 Contrôle d'un réseau de Petri p-temporel

Les réseaux de Petri permettent à la fois de modéliser le procédé et la commande. Dans ce contexte, les spécifications ou contraintes se déclinent en états interdits (marquages interdits). Pour cela, on restreint par le contrôle le fonctionnement du procédé afin que seuls les marquages autorisés puissent être atteints en interdisant le tir de certaines transitions contrôlables.

Ainsi, afin de gérer le problème de contrôle du réseau, nous utilisons une spécialisation de réseaux de Petri p-temporels, appelés réseaux de Petri p-temporels contrôlés [Jerbi et al.04].

Un réseau de Petri p-temporel contrôlé est un 4-uplet $Rpc = (Rp, \varphi, U, U_0)$ où :

- Rp est un réseau de Petri p-temporel qui décrit un système en boucle ouverte,
- φ est une application de l'ensemble des transitions T de Rp vers l'ensemble des opérations Γ, définie comme $\varphi : T \rightarrow \Gamma$,
- U est le contrôle externe des transitions de Rp utilisant des hypothèses sur les événements observables internes ou externes du système. Il est

défini comme $U : T \rightarrow \{0, 1\}$,

– U_0 est la valeur initiale du vecteur de contrôle.

Une transition contrôlable correspond toujours à un événement physiquement contrôlable. Une transition incontrôlable correspond soit à un événement physiquement incontrôlable, soit à un événement physiquement contrôlable, mais pour lequel la transition est toujours validée si l'état du système, représenté par son marquage, le permet.

4.2.2.3 Utilisation des automates à temps continus

Le modèle de commande proposé par [Culita et al.01] s'attache à éviter que le modèle ne se trouve dans un état interdit en utilisant un automate. En effet, ici l'analyse du comportement d'un réseau de Petri p-temporel est basée sur sa conversion en automate continu. Une variable Clock dans l'automate est affectée à chaque transition. Un arc dans le graphe des marquages du réseau de Petri correspond à une horloge activée dans une place dans l'automate. Une horloge activée décrit un couple (jeton, transition) dans le modèle réseau de Petri. Le contrôle de notre modèle *p*-temporel s'effectue donc ici en utilisant un automate à temps continu (définition 4.1).

Définition 4.1 (Automate à temps continu)
Un automate à temps continu est un 3-uplet $A =< S, X, E >$ *où* S *est un ensemble fini de lieux,* X *un ensemble fini de variables associées appelées Clock et* E *un ensemble fini de transitions discrètes.*

– $s, s' \in S$ *et* s *est le sommet source et* s' *le sommet de destination,*

– ψ *est une contrainte associée à la transition. Elle s'exprime par des combinaisons booléennes de contraintes linéaires* $L(X'), X' \subset X$,

– γ *est une fonction qui affecte une valeur réelle à une horloge donnée.*

4.2.2.4 Calcul des intervalles d'horloge valides

Pour illustrer ce calcul, nous nous intéressons au comportement d'un automate à trois états représenté par la figure 4.4.

Nommons $t^{min}_{l_j l_k}$ le temps de séjour minimal dans la place l_j avant son évolution vers la place l_k :

$$t^{min}_{l_j l_k} = max(0, a_{l_j l_k} - b_{l_i l_j})$$

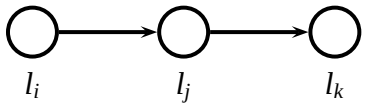

FIGURE 4.4 – Automate à trois états

Alors, $t^{max}_{l_j l_k}$ est la valeur maximale de temps de séjour dans la place l_j avant son évolution vers la place l_k.

$$t^{max}_{l_j l_k} = (b_{l_j l_k} - a_{l_i l_j})$$

Lorsqu'il y a une structure de choix sur la place l_j, la formule du temps de séjour maximum devient :

$$t^{max}_{l_j} = min(t^{max}_{l_j l_k})$$

Le vecteur d'horloge peut évoluer selon les deux cas suivants :

$$\forall h \in H, (h \notin R(l_i l_j) \wedge h \in Y_{l_i} \cap Y_{l_j}),$$

où $R(l_i l_j)$ est l'ensemble des variables d'horloge qui apparaissent comme des contraintes pour l'évolution de l_i vers l_j :

$$[a_{l_j l_k}, b_{l_j l_k}] \leftarrow [a_{l_i l_j} + t^{min}_{l_j l_k}, b_{l_i l_j} + t^{max}_{l_j}]$$

$$E_{l_j l_k} = E_{l_i l_j} + [t^{min}_{l_j l_k}, t^{max}_{l_j}]$$

et

$$\forall h \in H, (h \in R(l_i l_j) \wedge h \in Y_{l_j}),$$

$$[a_{l_j l_k}, b_{l_j l_k}] \leftarrow [a_{l_i l_j} + t^{min}_{l_j l_k}, b_{l_i l_j} + t^{max}_{l_j}]$$

$$E_{l_j l_k} = (E_{l_i l_j} + [t^{min}_{l_j l_k}, t^{max}_{l_j}]) \cap [a_{l_j l_k}, b_{l_j l_k}]$$

Pour la construction du comportement de l'automate, nous devons considérer toutes les évolutions possibles et calculer les intervalles de temps de séjour valides.

4.2.2.5 Évitement des états interdits

Il s'agit maintenant d'éviter les états interdits. L'algorithme qui le permet se décompose en deux règles de contrôle :

- Le premier scénario correspond à une évolution de l_j vers l_k qui est contrôlable. Dans ce cas, $E_{l_j l_k}$ est restreinte de telle sorte que l'évolution interdite n'arrive jamais. Cette première règle est appelée règle de contrôle direct. Lorsque l_f est une place interdite et $b^{new}_{l_j l_k}$ est le nouveau seuil supérieur de $E_{l_j l_k}$, les conditions deviennent :

$$\begin{cases} t^{max}_{l_j l_k} > t^{min}_{l_j l_k} \\ a_{l_j l_k} \leqslant b^{new}_{l_j l_k} \leqslant b_{l_j l_k} \end{cases}$$

  Lorsqu'il n'est pas possible de respecter les spécifications décrites ci-dessus, la seconde règle de contrôle doit être appliquée.
- Dans ce second cas, les contraintes sur les arcs menant à l_j sont considérées. Les contraintes associées à l'évolution $l_i l_j$ sont modifiées afin d'empêcher l'évolution vers des états interdits : cette règle est appelée règle de contrôle avec retour arrière.

Quand ces deux règles ne permettent pas d'éviter les états interdits l_f , la place l_j est considérée comme interdite. Par conséquent, l'algorithme va essayer la première puis la seconde règle de l'algorithme sur l_i et ainsi de suite.

4.2.3 Analyse structurelle d'un graphe d'événement

Lorsqu'il s'agit de vérifier que certains états jugés dangereux pour un système critique sont inaccessibles, l'existence de contraintes temporelles définies sur un temps dense empêche une exploration directe des états par énumération car ils sont en nombre infini. Dès lors, il semble pertinent de trouver une méthode nous permettant de nous dispenser de construire le graphe de marquage de notre réseau de Petri p-temporel et ainsi éviter d'être confronté au problème de l'explosion combinatoire du nombre d'états [Defossez et al.08b]. Il convient donc d'utiliser une approche analytique (analyse structurelle).

Ainsi un système ayant un fonctionnement répétitif peut être modélisé par un graphe d'événements fortement connexe [Lafit92]. Un graphe d'évé-

nements se caractérise par l'absence de conflit structurel. En effet, il s'agit d'un réseau de Petri pour lequel chaque place a exactement une transition d'entrée et une de sortie. Un graphe d'événements est fortement connexe si et seulement si il existe un chemin orienté qui relie tout sommet à tout autre sommet. Plus précisément, notre problématique nous invite à utiliser l'outil graphe d'événements fortement connexe p-temporel (notés p-GEFC), décrit dans [Khansa97].

4.2.3.1 Propriétés des graphes d'événements fortement connexes p-temporels

Ce paragraphe présente quelques propriétés intéressantes des p-GEFC :

Théorème 1 (Khansa)
Les conditions nécessaires et suffisantes d'existence des séquences marques-vivantes répétitive et complètes sont également des conditions nécessaires et suffisantes d'existence des fonctionnements 1*-périodiques.*

Une séquence de franchissement sera appelée séquence marques-vivantes si et seulement si tous les états accessibles par cette séquence sont des états marques-vivantes. La démonstration de ce théorème peut être trouvé dans [Khansa97]. Le contrôle de tels modèles peut être obtenu par des algorithmes polynomiaux.

Théorème 2
Soient G un p-GEFC, $[C_{min(G)}; C_{max(G)}]$ l'intervalle de temps de cycle de G. Une condition nécessaire et suffisante d'existence d'un fonctionnement périodique de période C pour G est que :

$$C \in [C_{min(G)}; C_{max(G)}]$$

4.2.3.2 Spécification des instants de tir

Les instants de tir peuvent être calculés avec un algorithme polynomial [Collart dutilleul et al.98]. Un graphe G' est associé au p-GEFC G dans un mode de fonctionnement mono-périodique de période C :

- les noeuds de G' sont les transitions de G,
- les arcs de G' sont obtenus à partir des places de G : deux arcs sont associés à chaque place p :

1. le premier de ${}^\circ p$ à p° dont la valeur est :

$$v_p = a_p - C.m_p$$

2. le deuxième de p° à ${}^\circ p$ dont la valeur est :

$$v'_p = -b_p + C.m_p$$

Le contrôle périodique des instants de tir est obtenu avec l'algorithme suivant :

1. choisir une transition t_s
2. associer $St_s(l) = 0$ avec t_s
3. associer à chaque transition $t_u \in T$, la première date de franchissement suivante :

$$St_u(l) = \max_{l_{su}} \sum_{p \in l_{su}} v_p$$

où l_{su} est un chemin élémentaire entre s et u.

Cet algorithme est en temps polynomial (en $O(n^3)$). Le contrôle n'est pas unique et, pour un temps de cycle donné, les instants de tir au plus tôt et au plus tard peuvent être calculés.

4.2.4 Exemple applicatif : un atelier

Dans cette section, nous illustrons les méthodes décrites précédemment autour d'un exemple simple comportant une synchronisation sous obligation. Considérons un robot qui doit gérer des produits (voir la figure 4.5). Ces produits arrivent sur un tapis roulant avançant à vitesse fixe sans s'arrêter. Lorsque le produit arrive en bout de tapis, il tombe et s'écrase sur le sol : ceci représente un état interdit pour le système. Le robot doit être contrôlé pour assurer le fonctionnement du système.

Le modèle des exigences est représenté par le réseau de Petri p-temporel de la figure 4.6. Ce modèle n'introduit pas de choix structurel pour la modélisation du système : il s'agit d'un graphe d'événements fortement connexe. Il est alors simple de calculer un temps de cycle. Cependant, lorsque les contraintes temporelles ne sont pas respectées, il n'existe aucun marquage

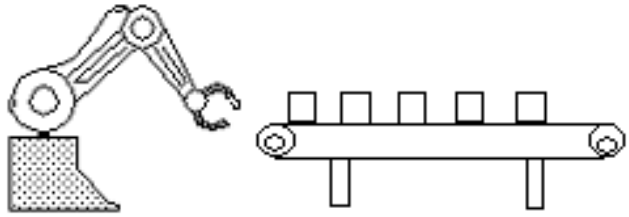

FIGURE 4.5 – Exemple d'un atelier.

modélisant l'état interdit, ou le passage en mode dégradé. Le modèle des exigences ne décrit pas ce genre de problématique, il ne s'intéresse qu'au respect des exigences temporelles.

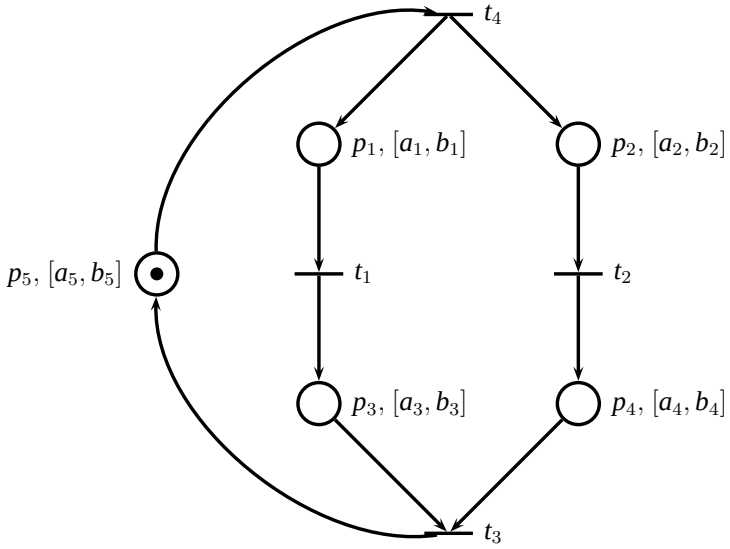

FIGURE 4.6 – Modèle des exigences de l'atelier.

Il est nécessaire de contrôler le système afin qu'il ne se trouve pas dans un état interdit. Nous allons illustrer sur cet exemple les deux méthodes décrites dans les deux sections précédentes.

4.2.4.1 Contrôle au moyen d'un automate associé

Nous appliquons ici à l'exemple la méthode décrite dans la section 4.2.2.

La figure 4.7 représente l'automate associé à la figure 4.6. L'état l_f représente l'état interdit correspondant à la mort d'une marque, ce qui correspond

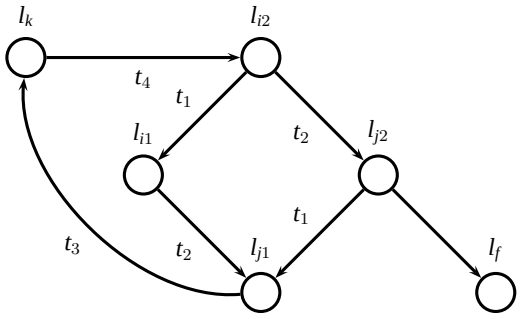

FIGURE 4.7 – Automate associé au réseau de la figure 4.6.

à une violation des spécification. Sur cet automate, l'arc qui mène à l_k est contrôlable. Par contre, l'évolution vers l'état interdit l_f ne l'est pas. Les évolutions vers l_{j1} sont contrôlables, le comportement du robot étant maitrisable.

Etat	Marquage	Situation
l_k	5	Produit sur Robot
l_{i2}	1,2	Robot et produits en mouvement
l_{i1}	2,3	Robot prêt
l_{j2}	1,4	Produit prêt
l_{j1}	3,4	Robot et Produit prêts
l_f	$\emptyset$	Dépassement temporel

FIGURE 4.8 – Etat de l'automate 4.7.

La figure 4.8 décrit les différents états représentés par l'automate de la figure 4.7. Nous appliquons alors l'algorithme de contrôle décrit plus haut.

À partir de l_{j2}, les règles de contrôle donnent :

$$[a_3, b_3] \leftarrow [a_3, min(b_3, b_4)]$$

Ce résultat est correct si $a_3 < a_2 + a_4 - b_1$. Alors, l'évolution appropriée des contraintes est :

$$[a_3, b_3] \leftarrow [a_3, a_2 + a_4 - b_1]$$

Si la relation précédente n'est pas vérifiée, la règle de contrôle arrière est appliquée. La solution n'est pas unique, mais nous proposons :

$$[a_3, b_3] \leftarrow [a_3, a_3]t_{l_{j1}l_{j2}}$$

$$[a_1, b_1] \leftarrow [a_1, a_2 + a_4 - b_3]t_{l_{i2}l_{j1}}$$

Ceci définit un contrôle valide, mais non unique, et il n'y a aucun moyen d'affirmer qu'il est optimal du point de vue du taux de production, ce qui n'est pas notre souci principal dans ce travail.

4.2.4.2 Contrôle au moyen d'un cographe

Ensuite, nous illustrons par cet exemple la méthode décrite en section 4.2.3. Remarquons que le réseau de Petri de la figure 4.6 est un graphe d'événements fortement connexe (définition 3.13 à la section 3.2.1.2).

D'après le théorème 1, le domaine de validité du taux de production est calculé en utilisant un algorithme polynomial. À partir de là, il est possible de calculer les instants de tir des transitions pour que la production soit correcte. Pour un cycle de fonctionnement donné C, le cographe associé est décrit en figure 4.9.

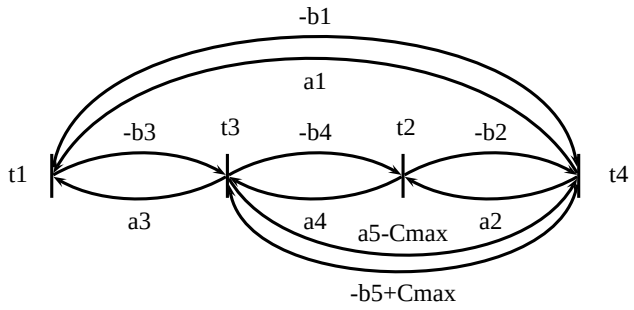

FIGURE 4.9 – Cographe associé au réseau de la figure 4.6.

Les instants de tir au plus tôt et au plus tard peuvent être calculés sur les chemins orientés considérés du graphe 4.9. Les marges de fonctionnement peuvent également être calculées afin d'intégrer certaines incertitudes sur les durées d'opération.

Considérons que $St_5 = 0$:

$$St_1 max = max a_1, a_2 + a_4 - b_3$$

$$[a_1, b_1] \leftarrow [a_1, a_2 + a_4 - b_3]t_{l_{i2}l_{j1}}$$

Nous obtenons donc les mêmes résultats avec les deux méthodes. Néanmoins, le dernier résultat correspond au calcul du plus long chemin dans un graphe, ce qui correspond à un problème polynomial bien connu.

4.2.5 Discussion sur les méthodes de synthèse de commande

Nous avons donc utilisé deux méthodes pour synthétiser le contrôle d'un système soumis à des contraintes temporelles. La première méthode nécessite de construire un automate temporisé associé au modèle réseau de Petri p-temporel à partir de son graphe de marquage. Ceci peut entrainer un problème d'explosion combinatoire lorsque le nombre d'états du système devient trop important. La deuxième méthode proposée répond à cet inconvénient puisqu'elle ne nécessite pas de construire le graphe de marquage du réseau. Néanmoins, elle ne fonctionne que si le système étudié peut être représenté par un graphe d'événements fortement connexe. C'est notamment le cas pour les systèmes en fonctionnement répétitif. Par contre, lorsque le système à contrôler comporte des indéterminismes (des choix), l'approche présentée par Sava dans son mémoire [Sava01] a toute sa pertinence.

Par ailleurs, en présence d'un problème de commande difficile comme un suivi de trajectoire, les approches s'appuyant sur les algèbres tropicales ont toute leur pertinence [Declerck et al.08, Didi alaoui05]. Il faut noter que les travaux que nous citons s'appliquent notamment à des modèles réseaux de Petri p-temporels. Ces approches ne sont donc pas concurrentes mais complémentaires à celles proposées dans cette section. On proposerait ainsi, de coupler le modèle graphique réseaux de Petri p-temporels, utile pour l'expert métier et la capture des exigences temporelles, avec un formalisme très puissant propre aux spécialiste de la synthèse de commande. On souligne pour conclure que l'ensemble de ces méthodes n'ont pas recours à la discrétisation du temps.

4.3 Projection de classes d'états pour la vérification d'une solution

Cette partie poursuit la description de notre méthodologie de conception visant à analyser et évaluer la sûreté des systèmes à événements discrets temporisés [Defossez et al.08c]. Dans la section précédente, nous proposons des méthodes permettant de valider un modèle des exigences construit à partir d'un réseau de Petri p-temporel. Ce modèle nous permet notamment d'identifier et d'extraire les états interdits du système en s'intéressant aux exigences de haut-niveau.

Ce modèle des exigences doit contenir toutes les contraintes contenues dans le cahier des charges, qui doivent être respectées, et plus particulièrement les contraintes temporelles, qui sont au cœur de ce travail. Notre objectif est toujours de fournir une aide à l'évaluation des spécifications. À la suite du travail présenté en section précédente, nous considérons que le modèle des exigences obtenu est valide au sens où il répond au cahier des charges et où il assure que les contraintes temporelles sont respectées.

À partir d'une solution proposée par un des acteurs du projet étudié, nous construisons un modèle de processus du fonctionnement du système au moyen d'un réseau de Petri t-temporel. Nous avons justifié du choix des modèles dans le chapitre 3 ; deux différents outils sont ainsi utilisés pour deux problèmes distincts, bien qu'ils soient très proches au niveau sémantique. Ce dernier aspect est important pour l'étape de vérification de l'adéquation du modèle de processus avec le modèle des exigences. Il s'agit alors de vérifier si ce modèle de processus correspond bien au modèle des exigences. De plus, dans un souci de respect du processus d'ingénierie des exigences, il est également nécessaire de respecter leur traçabilité. La construction du modèle de processus doit ainsi nécessairement respecter des règles pour permettre la comparaison des modèles et la propagation des exigences doit être documentée, ce qui permet également d'aider à la validation de la solution étudiée. Nous proposons de construire les classes d'états des deux réseaux afin d'explorer tous les comportements possibles du système. Ensuite, les deux ensembles de classes d'états sont comparés pour s'assurer que le modèle de processus de la solution proposée est valide.

Cette partie décrit dans un premier temps la construction des classes

d'états des deux extensions temporelles de réseaux de Petri utilisées dans cette méthodologie. Ces deux modèles ne sont pas équivalents et il convient de préciser quelle méthode de construction nous utilisons afin de pouvoir comparer les résultats obtenus.

Ensuite nous proposons de comparer les ensembles de classes obtenus afin de vérifier la validité de la solution proposée par rapport aux spécifications.

4.3.1 Le concept de classe d'état

Comme nous l'avons évoqué dans le section 3.2.1.3, l'analyse d'un réseau de Petri peut être effectuée grâce à deux méthodes : l'analyse structurelle et l'analyse énumérative. Cette dernière est utilisée afin de vérifier les propriétés dynamiques du système. Cette approche est basée sur la construction d'un arbre de couverture. La vérification que les états interdits ne sont pas atteignables ou l'analyse de contraintes temporelles entre des événements d'un scénario donné demande la recherche exhaustive de tous les états du système. Cependant, comme nous l'avons déjà précisé, dans le cas de contraintes temporelles, les états du système sont en nombre infini, ce qui rend leur exploration direct par énumération impossible à cause de l'explosion de l'espace d'états. Il est donc nécessaire de couvrir ces états par un nombre fini de classes, qui sont des abstractions d'états symboliques. Ainsi, une classe représente un ensemble d'états qui ont le même marquage et qui ont été obtenus à partir du marquage initial par le tir d'une séquence de transitions qui vérifie un ensemble de contraintes temporelles. Les différents état d'une même classe ont donc le même marquage M, et le domaine de tir associé à cette classe correspond à la réunion des intervalles dynamiques des différents états le constituant. Le passage de la notion d'état à celle de classes d'états est schématisé dans la figure 4.10.

Par conséquent, un graphe de classe peut être construit afin d'exprimer les différents états et les contraintes temporelles qui leur sont associés, qui permettent la transition d'une classe vers une autre. En fait, la notion de classes d'états permet une analyse similaire à la méthode des arbres de couverture pour les modèles temporisés en évitant l'explosion combinatoire. Une définition formelle d'une classe d'état est donnée ci-dessous.

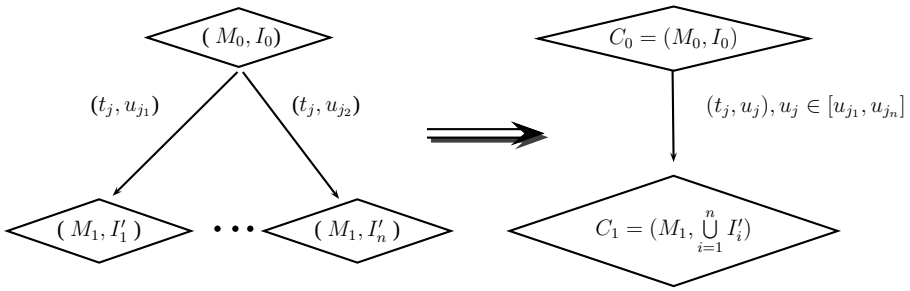

FIGURE 4.10 – Passage de la notion d'états à celle de classe d'états

Définition 4.2 (classe d'état d'un réseau de Petri temporel)
Une classe d'états C est un doublet $< M, D >$ avec :

- *M le marquage du réseau ;*
- *D le domaine de tir de la classe.*

Rappel sur les différents modèles Nous l'avons vu en section 3.2.3, même si ils sont proches, les pouvoirs d'expression des deux extensions temporelles des réseaux de Petri que nous utilisons ne se confondent pas. En effet, les contraintes temporelles s'exprimant différemment dans les deux modèles, la génération des classes ne se fait pas de la même façon.

Chronologiquement, [Berthomieu et al.91] a proposé le concept de classes d'états pour une analyse énumérative d'un réseau de Petri t-temporel. Des approches similaires furent proposées pour analyser le modèle p-temporel par [Khansa97] puis par [Bonhomme01]. Nous présentons, dans un premier temps, comment nous abordons ce type d'analyse dans le cadre de notre approche sur le modèle des exigences construit grâce à un réseau de Petri p-temporel. Ensuite, nous décrivons comment nous construisons le graphe de classes d'un réseau de Petri t-temporel afin de le comparer avec l'espace des exigences, ce qui nous permet de valider la solution proposée.

4.3.2 Analyse énumérative d'un réseau p-temporel

Le définition formelle des réseaux de Petri p-temporels se trouve à la section 3.2.2.3. Nous rappelons simplement ici qu'intuitivement, l'état d'un réseau de Petri p-temporel à un instant donné est défini par son marquage et par le temps écoulé individuellement pour chaque jeton, depuis sa création.

Cette section décrit les concepts de classe d'états pour cette extension des réseaux de Petri et explique la méthode choisie pour construire le graphe de classe d'états de notre modèle des exigences.

Nous nous basons sur l'approche initiée par [Bonhomme01]. Cette dernière repose sur l'établissement des conditions de mise à feu des transitions du réseau, évaluées aux différents instants de tir. À l'instar des outils d'analyse proposés par TINA pour les réseaux de Petri t-temporels, elle permet de regrouper un ensemble d'états équivalents en une seule classe afin d'en avoir une représentation finie [Berthomieu et al.03]. Il faut ici, marquer le choix qui a été fait et qui est différent de celui effectué antérieurement par Khansa en 1996. Comme la finalité de l'outil de modélisation était de garantir le respect des temps de séjours des marques dans les places, les premières classes représentaient les intervalles de temps de séjours possibles associés à un marquage donné. Cette approche, pour intuitive qu'elle soit, comporte un inconvénient majeur dans l'optique de nos travaux de thèse. En effet, cet ensemble de classe d'état ne capture pas en son sein l'ensemble des contraintes du système. Plus précisément, il n'intègre pas dans sa structure les couplages existants entre les temps de séjours des jetons dans différentes places. En d'autres termes, le fonctionnement temporel décrit par le graphe de classes d'état de Khansa fournit le plus petit hyperpavé, défini sur les valeurs des temps de séjours, qui contient l'espace des solutions conforme aux exigences. Cette dernière caractéristique montre que l'inclusion du comportement du système dans le modèle comportemental spécifié par les graphes de classes d'état donne une condition nécessaire, mais non suffisante du respect des exigences temporelles. C'est de tout évidence rédhibitoire, en effet le but recherché est la certification du respect des exigences sources.

Nous allons illustrer cet état de fait sur un exemple simple :

Soit le réseau de Petri p-temporel de la figure 4.11.

Si on calcule les classes d'états de ce réseau en utilisant les outils fournis par [Khansa97], nous obtenons les résultats répertoriés dans le tableau 4.1.

Par contre, en appliquant la méthode expliquée par [Bonhomme01], nous trouvons les classes d'états présentées dans le tableau 4.2

Dans le tableau 4.2 , les classes $C0$ et $C1$ sont équivalentes (voir annexe). La seule différence est que $C0$ correspond à l'état initial alors que $C3$ représente les ré-initialisation du réseau.

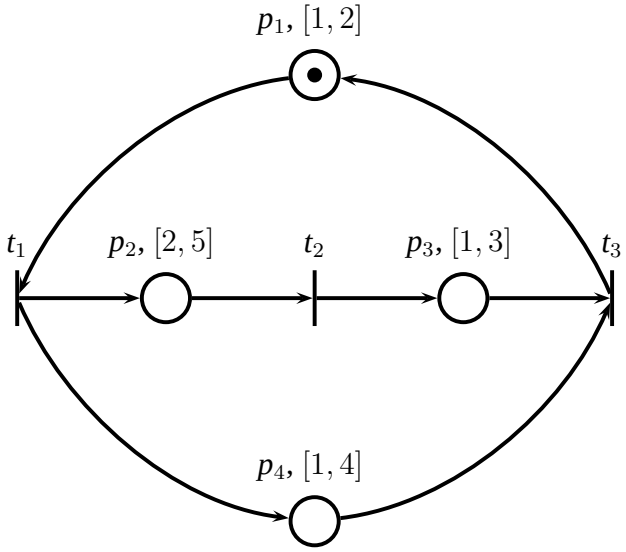

FIGURE 4.11 – Exemple RdP p-temporel.

Classe	Marquage	Domaine de tir
$C0$	p_1	$\theta_1 \in [1,2]$
$C1$	p_2, p_4	$\theta_2 \in [2,3]$
$C2$	p_3, p_4	$\theta_3 \in [1,2]$

TABLE 4.1 – Classes d'état du réseau 4.11 : méthode Khansa.

En comparant les résultats contenus dans les deux tableaux 4.1 et 4.2, on constate que la seule différence provient de l'existence d'une contrainte supplémentaire pour la classe $C3$: $TE(1,3) = 4$.

Concrètement, si on observe les classes $C1$ et $C2$ du tableau 4.1, rien n'interdit à la somme de θ_2 plus θ_3 d'être égale à 5. Cependant, cette valeur de 5 aboutit immanquablement à la mort du jeton dans la place p_4. On est donc bien en train de décrire une condition nécessaire mais non suffisante pour un comportement temporel correct du système. Remarquons bien,que c'est précisément, ce scénario qui est exclu par la contrainte $TE(1,3)$. Le detail des calculs se trouve en annexe, rappelons simplement que $TE(1,3)$ représente le temps maximal écoulé entre le tir de la première transition et le tir de la troisième transition.

Nous avons donc sur cet exemple simple illustré les lacunes de l'analyse

Classe	Marquage	TS	TE
$C0$	p_1	$TS(0) = t_1$	$TE = NULL$
$C1$	p_2, p_4	$TS(1) = t_2$	$TE(0,1) = 2$ $TE(1,0) = -1$
$C2$	p_3, p_4	$TS(2) = t_3$	$TE(1,2) = 4$ $TE(2,1) = -2$
$C3$	p_1	$TS(0) = t_1$	$TE(2,3) = 2$ $TE(3,2) = -1$ $TE(1,3) = 4$

TABLE 4.2 – Classes d'état du réseau 4.11 :méthode Bonhomme.

énumérative de Khansa pour les applications qui nous intéressent.

Comme ce sont les classes d'états de Bonhomme qui rendent compte de l'intégralité des contraintes, ce sont elles que nous utiliserons dans notre analyse. Elles permettront de construire une condition nécessaire et suffisante du respect des exigences sources exprimées sous formes de contraintes temporelles. Remarquons que nous suivons vraiment l'intention de ce mémoire en mettant à la disposition des experts métiers un outil graphique accessible qui leur permet d'exprimer les exigences sources. Par ailleurs nous exploitons la puissance mathématique inhérente au modèle pour construire une analyse comportementale complexe. On cherche donc à éviter au gens du métier la spécification du système sous un format mathématique sophistiqué.

Tout d'abord, nous expliquons comment sont construites les classes d'états dans ce contexte. Ensuite, nous décrivons les mécanismes qui vont nous permettre d'obtenir le graphe des classes d'états nécessaires à notre approche.

4.3.2.1 Construction des classes d'états

Dans un réseau de Petri p-temporel, comme pour un réseau de Petri élémentaire, un état peut être atteint à partir de l'état initial en suivant une séquence de tirs, que l'on appelle s. Un instant de tir, appelé u est associé à chaque transition de cette séquence.

Le domaine de tir D d'une classe d'états peut être défini comme l'ensemble des solutions d'un système d'inégalités où les variables sont associées aux différentes marques présentes dans le marquage de la classe $C = (M, D)$

considérée. Ainsi, D prend la forme :

$$D = \{\Theta, A.\Theta \leqslant b\}$$

où :
- A est une matrice de coefficients ;
- b est un vecteur de constantes ;
- Θ est un vecteur de variables, fonction des marques présentes dans le marquage de la classe considérée. On note θ_i^n la date de sortie de la marque n contenue dans la place p_i.

Pour trouver le domaine de tir D, il est nécessaire de prendre en compte non seulement les jetons situés dans les places amont de la transition sensibilisée considérée, mais également tous les jetons dans les places amont des autres transitions du réseau étudié.

Une première approche fut initié dans [Khansa97] inspirée par les travaux sur les réseaux t-temporels décrits dans [Menasche82, Berthomieu et al.91].

La méthode que nous avons choisie se base sur la notion d'instant de tir, qui fut introduite par [Boucheneb et al.93]. Un de ces objectifs principaux est de réduire la complexité des opérations à effectuer pour construire les classes d'états.

4.3.2.2 Calcul de la classe d'états suivante

La méthode proposée dans [Bonhomme01] repose sur l'établissement des conditions de mise à feu des transitions du réseau, évaluées au premier, au second et au $q^{ème}$ instants de tir. Elle reprend le concept de classe d'état pour contourner l'explosion du nombre d'états en regroupant en un unique état un ensemble d'états équivalents. Entre deux instants de tir consécutifs, tous les états ayant le même marquage sont agrégés afin de construire une seule classe d'états. Classiquement, dans les méthodes citées précédemment (notamment dans [Khansa97]), le calcul de ces classes d'états nécessitait la résolution d'un système d'inégalités linéaires non triviale. Dans l'approche utilisée ici, le calcul des classes fait donc référence à la notion d'instants de tir, introduite dans [Boucheneb et al.93] pour le modèle t-temporel. L'adaptation au modèle p-temporel requiert un nouveau formalisme pour évaluer les instants de tir. Ce formalisme et l'algorithme de construction des classes d'états que nous avons utilisés sont décrit en annexe.

4.3.2.3 Construction du graphe des classes d'états

Les graphe de classes d'états, comme le graphe des marquages accessibles pour le modèle autonome, fournissent un outil pour la validation permettant d'étudier le comportement du système étudié et de vérifier certaines propriétés, telles que la vivacité, le caractère borné, l'existence de fonctionnements répétitifs par exemple.

4.3.3 Analyse énumérative d'un réseau t-temporel

La définition formelle des réseaux de Petri t-temporels se trouve à la section 3.2.2.2. Nous rappelons ici qu'ils sont obtenus en associant des dates *min* et *max* à chaque transition d'un réseau de Petri. Si une transition t est sensibilisée à une date θ, elle ne peut être tirée qu'après la date $\theta + min$ et doit être tirée avant la date $\theta + max$ (sauf si le tir d'une autre transition désensibilise t).

Leur domaine d'application est large puisqu'ils sont capable d'exprimer aussi bien des spécifications en délais qu'en durées.

L'analyse énumérative consiste généralement à construire le graphe de couverture du réseau étudié afin d'en vérifier les propriétés dynamiques. Dans le cas des réseaux temporels, elle se base sur le concept de classe d'états. Cette approche se base sur les travaux de [Berthomieu01].

4.3.3.1 Classe d'états d'un réseau t-temporel

Le domaine de tir D d'une classe d'états peut être défini comme l'ensemble des solutions d'un système d'inégalités où les variables sont associées aux transitions validées par le marquage de la classe C considérée. Ainsi, D prend la forme :

$$D = \{v, A.v \leqslant b\}$$

où :
- A est une matrice de coefficients ;
- b est un vecteur de constantes ;
- v est un vecteur de variables, ayant une composante pour chaque transition sensibilisée par M.

Il est nécessaire de préciser que deux classes sont égales si et seulement si leurs marquages et leurs domaines d'états respectifs sont égaux.

Ensuite, la section suivante s'applique à définir une relation d'accessibilité pour passer d'une classe à la suivante, qui permet de construire le graphe des classes d'état du réseau t-temporel.

4.3.3.2 Calcul de la classe suivante

Nous décrivons ici les conditions nécessaires au tir d'une transitions à partir d'une classe donnée. Soit la classe C telle que $D = \{v, A.v \leqslant b\}$. Pour un vecteur $v \in D$, on note v_i la composante de v relative à la transition t_i.

Une transition t_i est tirable depuis la classe C si :

1. t_i est sensibilisée par M (c'est à dire $M \geqslant Pre(t)$),
2. t_i peut être franchie avant tout autre transition sensibilisée par C ($\forall j \neq i, v_i \leqslant v_j$).

Ensuite, nous présentons l'algorithme de calcul de la classe suivante $C' = (M', D')$. Dans un premier temps, l'évolution du marquage se fait de la même façon que pour les réseaux de Petri autonomes. Donc

$$M' = M - Pre(t_i) + Post(t_i)$$

Ensuite, le domaine D' est construit en quatre étapes :

1. On ajoute au système $D = \{v, A.v \leqslant b\}$ les conditions de tir de t_i qui expriment qu'elle est la première tirée :
$$v, A.v \leqslant b, v_i \leqslant v_j$$
2. les variables associées aux transitions en conflit avec t_i sont éliminées,
3. chaque variable v_j avec $j \neq i$, est remplacée par $v_i + v_j$; puis on élimine v_i,
4. pour chaque transition t_j nouvellement sensibilisée, on ajoute les contraintes :
$$Min(IS(j)) \leqslant v_j \leqslant Max(IS(j))$$

À partir de là, nous pouvons construire le graphe des classes d'états, qui permet de confirmer l'accessibilité d'un marquage particulier, de prouver la finitude du réseau ou de découvrir l'existence de fonctionnements répétitifs.

4.3.4 Validation du modèle de solution par projection des classes d'états

Cette projection se base sur des liens créés entre les exigences et une solution proposée. L'existence de ce lien est une hypothèse forte dans le cadre de ce travail.

Indépendamment de l'aspect mathématique de ce travail, il faut vraiment garder à l'esprit dans quel contexte applicatif on se trouve.

Pour donner un exemple concret, on pourra dire que le port d'un casque est suffisant pour répondre à une exigence de protection des personnels. Cette proposition doit être soumise à une validation d'un ou de plusieurs experts. C'est en effet une hypothèse forte qui vient avant toute considération temporelle, du type du temps de dégagement de la zone de danger ou temps de mise du casque.

L'objectif de ce travail est de vérifier si la projection de l'ensemble des classes d'états généré à partir du réseau de Petri t-temporel est inclus dans l'ensemble des classes d'états à partir du réseau de Petri p-temporel. Ceci peut être décrit d'une manière plus formelle de la façon suivante :

Notons :

- Sr : ensemble des classes d'états générées à partir du réseau de Petri p-temporel modélisant les exigences,
- Ss : ensemble des classes d'états générées à partir du réseau de Petri t-temporel modélisant une solution proposée pour le système,
- Req : l'espace des exigences, c'est-à-dire l'ensemble des nœuds du modèle des exigences,
- $R(A, B)$: une application représentant la projection de A vers B.

Dans la démarche proposée, l'expression des exigences sources nécessite un expert du métier. Dans un deuxième temps, à partir de ces dernières, un modèle des exigences est construit. La projection décrite dans cette section vise à vérifier la consistance entre le modèle d'une solution proposée et le modèle des exigences. Cependant, il convient de relativiser la portée mathématique du mot projection que nous utilisons dans ces lignes. De fait, affirmer qu'une partie de la solution veut revendiquer une partie des exigences est une activité critique qui ne peut se faire sans l'assistance d'un expert. C'est donc lui qui valide les dépendances fonctionnelles existant entre les deux modèles. En l'état actuel des choses, il nous semble déraisonnable de procéder autre-

ment.

Nous pouvons évoquer [Kirwan et al.02] qui affirme que « L'émetteur de la règle est leader de sa traduction ». On entend ici par traduction, l'instantiation des principes décrits dans une règle en fonction d'un contexte métier.

Enfin, pour en finir avec les réserves sur le mot projection, c'est l'expert qui au regard des résultats de l'analyse de consistance des modèles, va valider ou non la solution proposée. Il ne s'agit absolument pas d'une fonction mathématique booléenne. Nous avons juste voulu instrumenter une aide à la décision dudit expert.

La validation d'une solution reste malgré tout soumise à la subjectivité, que l'on essaie d'encadrer pour faciliter l'étude et le jugement du décideur. La décision finale appartient également à l'expert. Notre méthode va permettre de détecter une inconsistance.

L'analyse temporelle est un paramètre parmi d'autres, que l'on étudie une fois que les paramètres structurels ont été fixés. elle ne peut intervenir qu'après une étude préliminaire assurant de la mise en sécurité du système, travail qui est le fondement de la conception de systèmes sûrs de fonctionnement. Cette étude est une condition nécessaire à la mise en œuvre de notre proposition.

Dans un premier temps, la consistance structurelle est obtenue par construction [Defossez et al.10]. Il faut maintenant vérifier la consistance comportementale à l'aide des graphes de classes d'états. Rappelons que l'évolution d'une classe d'état à la suivante découle du tir d'une transition.

Notons $R(Ss, Req)$, le sous-ensemble de Ss, l'ensemble des états dont les classes d'états vérifient :

1. les classes d'états du modèle de solution résultant du tir d'une transition t appartenant au modèle des exigences,

2. les classes d'états du modèle de solution qui permette la mise à feu d'une transition t appartenant au modèle des exigences.

La consistance comportementale est vérifiée lorsque :

$$R(Ss, Req) \subset Sr$$

4.3.5 Exemple illustratif :

Afin d'illustrer les différentes étapes de notre méthodologie, nous proposons d'étudier l'exemple théorique suivant : Nous considérons un système de production avec une phase de transport. Le produit a une durée de production maximale, que l'on note *max*.

4.3.5.1 Approche appliquée à l'exemple : p-temporel

La figure 4.12 montre le réseau de Petri p-temporel correspondant à cette spécification succincte.

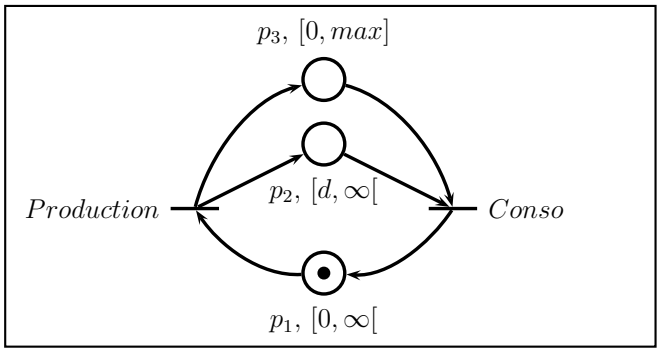

FIGURE 4.12 – Modèle des exigences de l'exemple

La transition *production* (resp. *consommation*) représente la phase de production (resp. consommation) du processus. La place p_3 représente la contrainte de durée de production à ne pas dépasser, la place p_2 la durée de transport du produit (qui est d'au moins d unités de temps), et la place p_1 correspond à la ré-initialisation du système.

La figure 4.13 donne le graphe de classes d'états correspondant au réseau de Petri p-temporel représentée par la figure 4.12.

Les classes d'états sont décrites dans le tableau 4.3. Les θ représentent les instants de tir potentiel des transitions permettant de passer d'une classe à une autre.

4.3.5.2 Approche appliquée à l'exemple : t-temporel

La seconde étape de notre proposition consiste à construire le modèle de solution du système au moyen d'un réseau de Petri t-temporel. Tous les

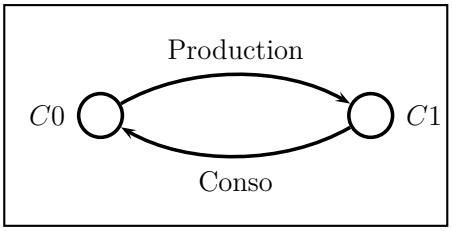

FIGURE 4.13 – Graphe de classes d'états du réseau p-temporel de l'exemple.

Classe	Marquage	Domaine
$C0$	1	$\theta_{production} \in [0, \infty[$
$C1$	2, 3	$\theta_{conso} \in [d, max[$

TABLE 4.3 – Marquages correspondant aux classes d'états de la figure 4.12.

nœuds du modèle de solution doivent avoir un nœud correspondant dans le modèle des exigences. Afin de respecter cette contrainte, il est recommandé de construire un manuel de construction du modèle de solution pour respecter la cohérence structurelle des modèles.

La figure 4.14 décrit le réseau de Petri t-temporel qui modélise un processus répondant aux spécifications modélisées dans la figure 4.12.

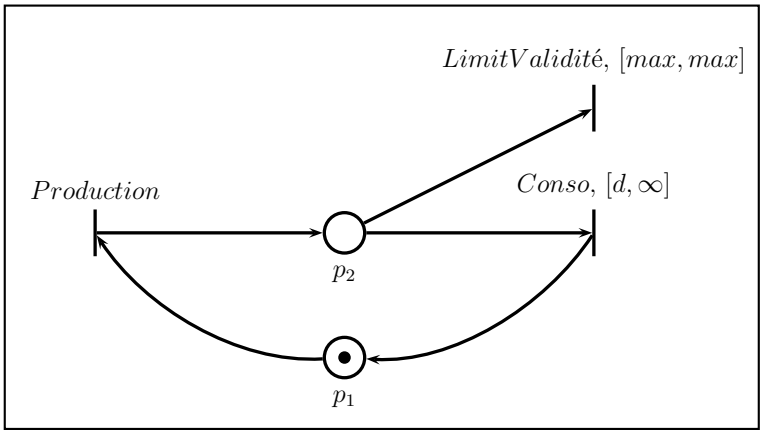

FIGURE 4.14 – Exemple de modèle de solution

Nous pouvons noter que les transitions (respectivement les places) gardent le même nom, exceptée la place modélisant la contrainte concernant la date

de consommation (voir table 4.4). Cette dernière est transformée en une transition, validée uniquement si la limite *max* est atteinte. Dans ce cas, les spécifications ne sont plus respectées.

Noeuds figure 4.12	Noeuds figure 4.14
Production	*Production*
Conso	*Conso*
p_1	p_1
p_2	p_2
p_3	t_{p3}

TABLE 4.4 – Correspondance des nœuds entre les deux modèles.

La description des classes d'état peut être trouvée en figure 4.15. La définition théorique de cette construction peut être trouvée en section 4.3.3.2 et l'outil logiciel TINA [Berthomieu et al.03] est utilisé afin de générer les classes d'états des réseaux *t*-temporels modélisant un processus solution.

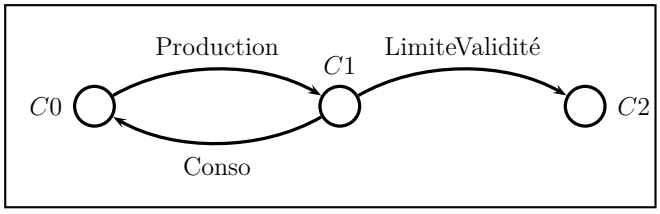

FIGURE 4.15 – Graphe de classes d'états du réseau *t*-temporel de l'exemple.

Classe	Marquage	Domaine
$C0$	1	$\theta_{production} \in [0, \infty[$
$C1$	2	$\theta_{conso} \in [d, max[$ $\theta_{LimiteValidité} \in [max, max]$
$C2$	$\emptyset$	$\emptyset$

TABLE 4.5 – Marquage correspondant aux classes d'états de la figure 4.14.

4.3.5.3 Approche appliquée à l'exemple : projection

Pour cet exemple simple, les correspondances entre les classes C0 et C1 sont triviales. La classe C2 correspond a la modélisation d'un état interdit qui n'est pas nécessaire lorsqu'on utilise les p-temporels. La vérification de consistance se restreint donc à la vérification de la cohérence des domaines temporels associés.

On remarquera le décalage dans les notations utilisées pour les deux modèles au niveau de l'expression des contraintes temporelles.

4.4 Conclusion

Nous avons donc proposé dans ce chapitre une méthodologie permettant d'une part d'intégrer les contraintes temporelles dans un modèle des exigences, d'autre part de représenter les spécifications temporelles dans un modèle associé à une solution proposée. Enfin, on a montré que des vérifications de consistance systématiques pouvaient être mis en œuvre sur les modèles comportementaux générés. Ces vérifications constituent un outil d'aide à la certification qu'il conviendrait cependant de valider sur un exemple proche des métiers du ferroviaire. En effet, si ce chapitre a illustré la faisabilité de notre approche, sa pertinence industrielle demande à être confirmée. Le chapitre suivant fera précisément l'objet d'une application des propositions du chapitre 4 sur un cas d'étude de la littérature qui concerne le passage à niveau.

Chapitre 5

Application et validation de l'approche proposée : cas du passage à niveau

Sommaire

Introduction

Ce chapitre a pour objectif d'illustrer la contribution méthodologique présentée au chapitre précédent. Nous nous sommes intéressés particulièrement à la problématique du passage à niveau.

Les S.T.I. (voir section 1.2.3) relatives au contrôle-commande du passage à niveau n'ont à l'heure actuelle pas encore été publiées. Nous avons choisi comme cas d'étude une description de passage à niveau trouvée dans la littérature [Jansen et al.00] et autour duquel nous avons trouvé un certain nombre de travaux [Bon00, Boulanger et al.06]. Après une présentation succincte du cahier des charges, le modèle des exigences est donc construit à l'aide des réseaux de Petri p-temporels [Collart dutilleul et al.06, Defossez et al.07, Defossez et al.08b]. Ensuite, une solution est spécifiée d'un point de vue temporel en utilisant des réseaux de Petri t-temporels [Defossez et al.08c]. Enfin, les deux modèles ainsi obtenus sont mis en correspondance pour valider l'adéquation entre les spécifications et les exigences [Defossez et al.10].

5.1 Le passage à niveau : un composant critique

Un passage à niveau est un croisement d'une ligne ferroviaire avec une voie routière ou piétonnière. Le terme *à niveau* signifie que ces voies se croisent à la même hauteur, par opposition aux tunnels ou aux ponts. La circulation des matériels ferroviaires est toujours prioritaire sur les usagers de la route. Dans le domaine de la sécurité des transports ferroviaires, le passage à niveau est un composant critique. Malgré les efforts des gouvernements et des gestionnaires et exploitants d'infrastructure routière et ferroviaire, les accidents survenant aux passages à niveau sont à l'origine de 420 morts par an en Europe [Khoudour et al.08]. Cette valeur représente 29 % des décès des accidents du trafic ferroviaire (contre 1% pour le routier). Les accidents au passage à niveau sont donc un problème majeur du point de vue de la sécurité ferroviaire. Nous pouvons ajouter que les dégâts occasionnés par les accidents au passage à niveau ont des conséquence très importantes en terme

de réparation et de perturbation du trafic, qui sont chiffrées à 110 millions d'euros par an.

En France, il restait 19133 passages à niveau au premier janvier 2005. Aucun passage à niveau n'est situé au croisement d'une ligne à grande vitesse. Cependant, des accidents peuvent impliquer des TGV circulant sur des lignes classiques à moins de 160 km/h.

Pour illustrer la criticité de ce composant, nous pouvons citer quelques accidents ayant eu lieu sur des passages à niveau :

- 8 Avril 1993 : Collision entre un train et un car scolaire près d'Aix en Provence (4 morts) ;
- 22 Septembre 1995 : Collision entre un train et une voiture à Agde (5 morts) ;
- 8 Septembre 1997 : Collision entre un autorail et un camion de fioul ayant forcé un passage à niveau à Porte Sainte Foy (13 morts et 43 blessés) ;
- 9 Juin 2005 : Collision entre un TER et un camion chargé de bouteilles de gaz à Saint Laurent Blangy.

Ces exemples montrent bien la pertinence du choix du passage à niveau comme étude de cas.

5.2 Cas d'étude proposé par Jansen

Ainsi, Lars Jansen et Eckehard Schnieder, de l'université de Braunschweig en Allemagne, ont publié en 2000 un article [Jansen et al.00] décrivant un système de contrôle de passage à niveau communiquant par fréquences radio. Cette étude de cas est destinée à la recherche scientifique, et plus particulièrement à la mise en application de nouvelles techniques formelles : sa spécification est à la fois plus réaliste et plus conséquente que les exemples théoriques traditionnels, mais reste toutefois moins complexe et moins hermétique que celle des projets d'envergure industrielle. Elle est décrite dans la suite de cette section. De plus, ce cas d'étude contient la description d'un protocole proposant une solution de fonctionnement du passage à niveau répondant aux exigences. Ceci est pertinent dans notre travail puisque l'on s'intéresse à la traçabilité des exigences : il s'agit de vérifier si les exigences contenues dans le cahier des charges sont bien respectées tout au long du

processus de conception. De plus, il doit être possible à tout moment de ce processus de suivre le traitement de ces exigences.

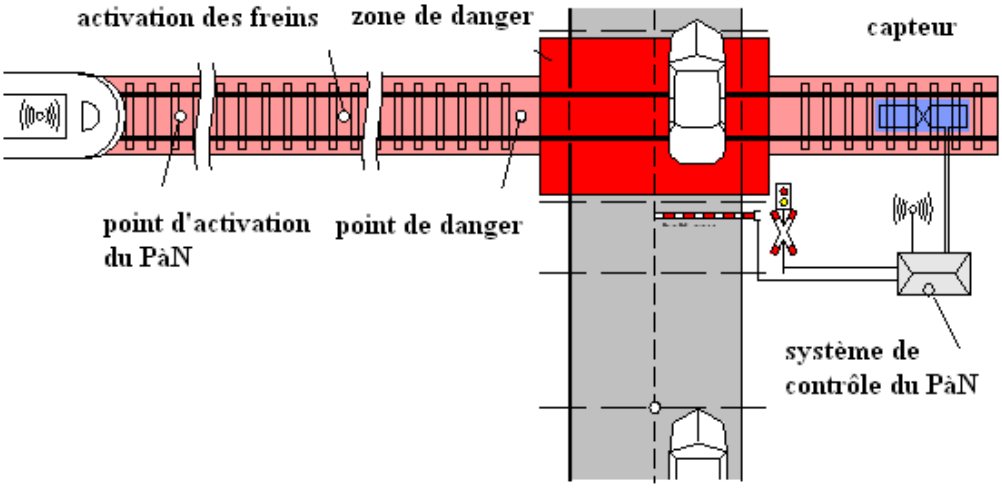

FIGURE 5.1 – Cas d'étude du passage à niveau.

Il est important de souligner que ce type de passage à niveau n'est pas implanté actuellement. En effet, comme nous l'avons précisé en introduction, les S.T.I. contrôle-commande du passage ne sont pas publiées. L'étude de cas présentée peut néanmoins servir de prototype à l'application d'ERTMS (voir section 1.2.3) et de son système de communication au passage à niveau.

5.2.1 Description du cas d'étude

Le croisement de la route et de la ligne de chemin de fer est appelé zone de danger (voir figure 5.1), dans la mesure où les trains et les usagers de la route (automobilistes, cyclistes, piétons) ne doivent pas y pénétrer en même temps, pour éviter tout risque de collision. Le passage à niveau est pourvu de demi-barrières, ainsi que de feux de signalisation commandés par le système de contrôle du passage à niveau. Ce dernier est activé lorsqu'un train approchant du passage à niveau est correctement détecté. Un autre système de contrôle est embarqué dans le train. Enfin, un centre d'opérations supervise les interactions entre les deux précédents composants. Les demi-barrières sont utilisées pour empêcher le franchissement du passage à niveau

aux usagers de la route lors du passage d'un train. L'utilisation de telles barrières rend possible le franchissement du passage à niveau par les usagers de la route qui ne respecteraient pas le code de la route et qui prendraient le risque de passer par la voie de gauche. On a pu observer cette infraction induite par un trop long temps d'attente devant un passage à niveau fermé. Pour tenir compte de cette possibilité, un temps maximal de fermeture du passage à niveau au delà duquel les barrières seront relevées est imposé dans la spécification. Ainsi, le train doit arriver au passage à niveau 240 secondes après avoir envoyé son ordre d'activation. Au bout de ces 240 secondes, les barrières seront relevées et les lumières éteintes. Il est donc important de contrôler que cette spécification ne rende pas notre système non-sécurisé par exemple dans le cas d'un train qui mettrait plus de 240 secondes pour dépasser le passage à niveau et qui le traverserait alors qu'il n'est pas fermé à la circulation.

5.3 Modèle des exigences

Comme décrit dans la partie 4.2, la première étape consiste en la construction d'un modèle des exigences. Nous proposons un modèle de réseau de Petri p-temporel pour décrire le système composé du passage à niveau et du train. Le réseau de Petri p-temporel représentant le modèle des exigences de la spécification présentée ci-dessus se trouve en figure 5.5.

5.3.1 Modélisation des exigences de sécurité

Nous avons choisi de présenter ici une modélisation spécifique permettant d'assurer un bon fonctionnement du système en mode dégradé.

Sur la figure 5.2, on peut observer deux séquences synchronisées lorsque certains événements ont lieu. L'exigence temporelle décrite en section 5.2.1 est représentée par la place p_{12}, qui modélise la limitation de la durée de fermeture du passage à niveau.

Il est intéressant de souligner que les spécifications temporelles n'introduisent pas de choix structurel sur la modélisation du système : ceci est un des avantages de l'utilisation des réseaux de Petri p-temporels.

Il existe deux sortes d'états interdits dans le modèle de la figure 5.2 :

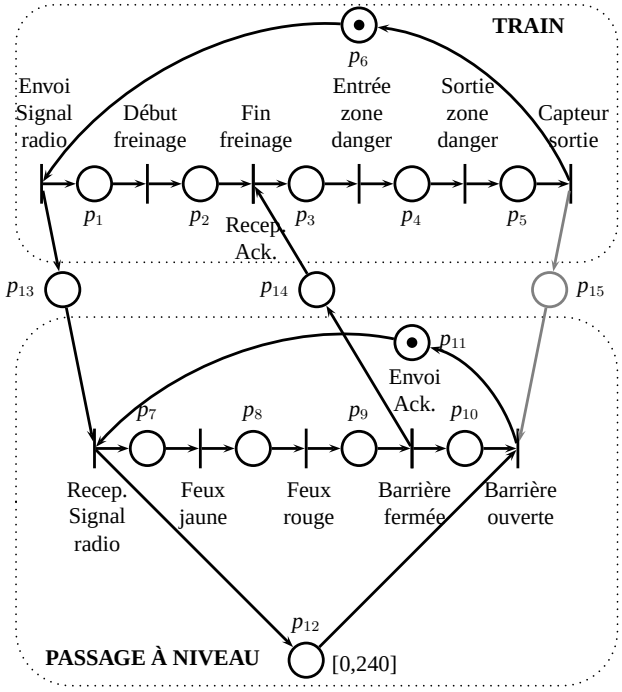

FIGURE 5.2 – Modèle des exigences du cas d'étude en mode dégradé

1. les premiers proviennent des spécifications temporelles en présence de synchronisations,
2. les seconds correspondent à la présence d'un train dans la zone de danger lorsque les barrières sont ouvertes. De fait, lorsque la place p_4 contient un jeton, ce qui correspond au passage d'un train dans la zone de danger, l'unique jeton dans la portion modélisant le passage à niveau doit se trouver obligatoirement dans la place p_{11} .Tous les autres états sont interdits. Cette dernière catégorie appartient à une classe d'états interdits largement abordée par la littérature [Krogh et al.91].

5.3.2 Contrôle du modèle p-temporel

On va donc désormais s'attacher à éviter que notre modèle de réseau de Petri temporel se trouve dans un état interdit. Nous nous sommes intéressés au cas où le passage à niveau ne reçoit pas l'information de sortie de zone de danger du train, soit par défaillance du capteur de sortie, soit par défaut

de communication entre le passage à niveau et son système de contrôle-commande. Ceci peut se traduire par une disparition de la place p_{15} (place grisée) dans la figure 5.5. Il faut alors contrôler le système afin qu'il ne se retrouve pas dans l'état interdit correspondant à l'ouverture des barrières de sécurité alors qu'un train est en zone de danger.

5.3.2.1 Analyse avec un automate associé

Il s'agit donc dans cette partie d'appliquer la méthode décrite en section 4.2.2 au cas d'étude passage à niveau afin de remplir les exigences de sécurité. La figure 5.3 représente l'automate simplifié correspondant au réseau de Petri de la figure 5.2 modélisant le passage à niveau.

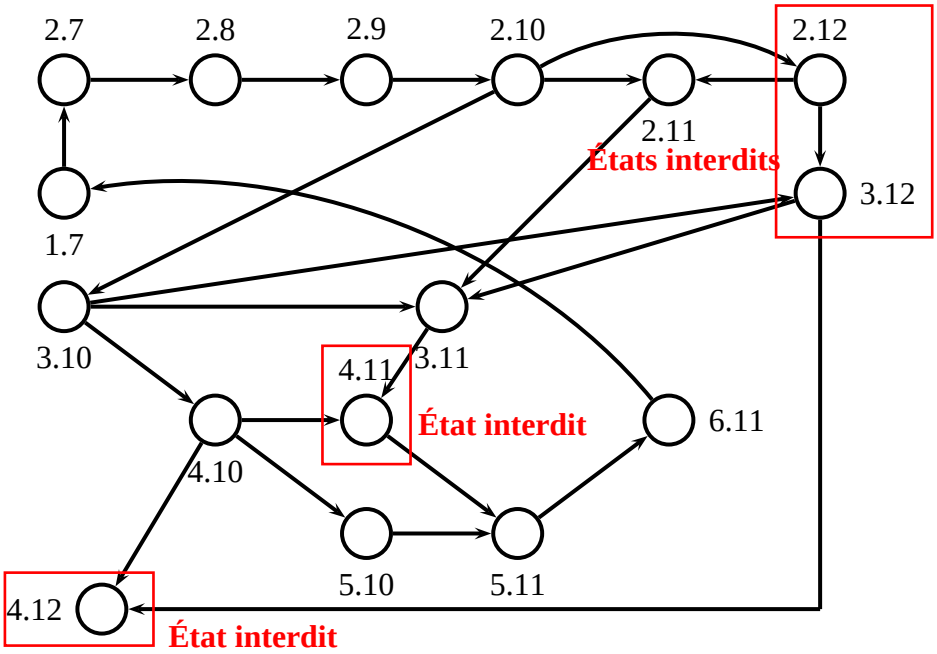

FIGURE 5.3 – Automate (simplifié) associé à la figure 5.2.

Par exemple, on peut s'intéresser à l'état interdit 4.11 à partir de l'état 4.10 (train dans la zone de danger et barrières fermées). Un point à prendre en compte est la contrôlabilité de la transition vers 5.10. Cette transition est considérée comme incontrôlable parce que le train n'est pas supposé accélérer dans la zone de danger. En fait, la seule transition de sortie contrôlable

de l'état 4.10 mène à l'état 4.11, qui est un état interdit. Par conséquent, les règles de contrôle directes ne résolvent pas le problème de l'évitement des états interdits. Il faut donc considérer les règles de contrôle avec retour arrière. La transition qui relie les places $p9$ et $p10$ dans le modèle réseau de Petri est considérée comme étant incontrôlable : le seul moyen de contrôle se situe dans la transition qui va de la place $p3$ à la place $p4$.

En utilisant le formalisme décrit dans la partie 4.2.2, l'arc menant à 4.10 sera actualisé en utilisant la valeur maximale de temps de séjour.

$$\begin{aligned} t^{max}_{4.10\ 4.12} &= (b_{4.10\ 4.12} - a_{3.10\ 4.10}) \\ &= 240 - (a_3 + a_2 + a_1) \end{aligned}$$

$$\begin{aligned} t^{min}_{4.10\ 4.12} &= min(0, (a_{4.10\ 4.12} - b_{3.10\ 4.10})) \\ &= min(0, (240 - (b3 + b2 + b1))) \end{aligned}$$

$$\begin{aligned} t^{max}_{4.10\ 5.10} &= (b_{4.10\ 5.10} - a_{3.10\ 4.10}) \\ &= (b4 + b3 + b2 + b1) - (a3 + a2 + a1)) \end{aligned}$$

$b_{4.10\ 5.10}$ est la valeur maximale d'horloge associée au jeton dans p_4 et $a_{3.10\ 4.10}$ est la valeur minimum de l'horloge associée au jeton dans la place p_3. Manifestement, la valeur à observer est b_4. L'état 4.10 peut être contrôlé grâce aux règles de contrôle direct si :

$$\begin{cases} b_4^{new} < 240 - (b_3 + b_2 + b_1) \\ a_4 < b_4^{new} < b_4 \end{cases}$$

Comme b_3 peut être infinie, le problème de contrôle ne peut pas être résolu à ce niveau et on doit appliquer les règles de contrôle avec retour arrière. Donc, si la valeur d'horloge associée à la place p_{12} est plus grande que $(b_{12} - a_4)$, la transition vers la place p_{10} n'est pas validée du point de vue du contrôle, alors que la transition vers la place p_{11} l'est. Le fait d'éviter que l'on se retrouve dans l'état 4.11 à partir de l'état 3.11 est un simple problème de contrôle puisque la transition est contrôlable.

5.3.2.2 Analyse structurelle au moyen d'un graphe associé

Le réseau de Petri p-temporel modélisant le passage à niveau (figure 5.2) est un p-GEFC. Ainsi, d'après le théorème 1, le fonctionnement valide de notre système peut être calculé en temps polynomial. Il est donc possible de

choisir un comportement sécurisé du système et de calculer les instants de franchissement.

Pour un cycle de fonctionnement donné C, le graphe développé associé au p-GEFC de la figure 5.2 est décrit par la figure 5.4. Les arcs reliant les transitions t_{10} et t_3 sont en pointillés par souci de visibilité et sont normalement annotés par $-b_{14}$ et a_{14}.

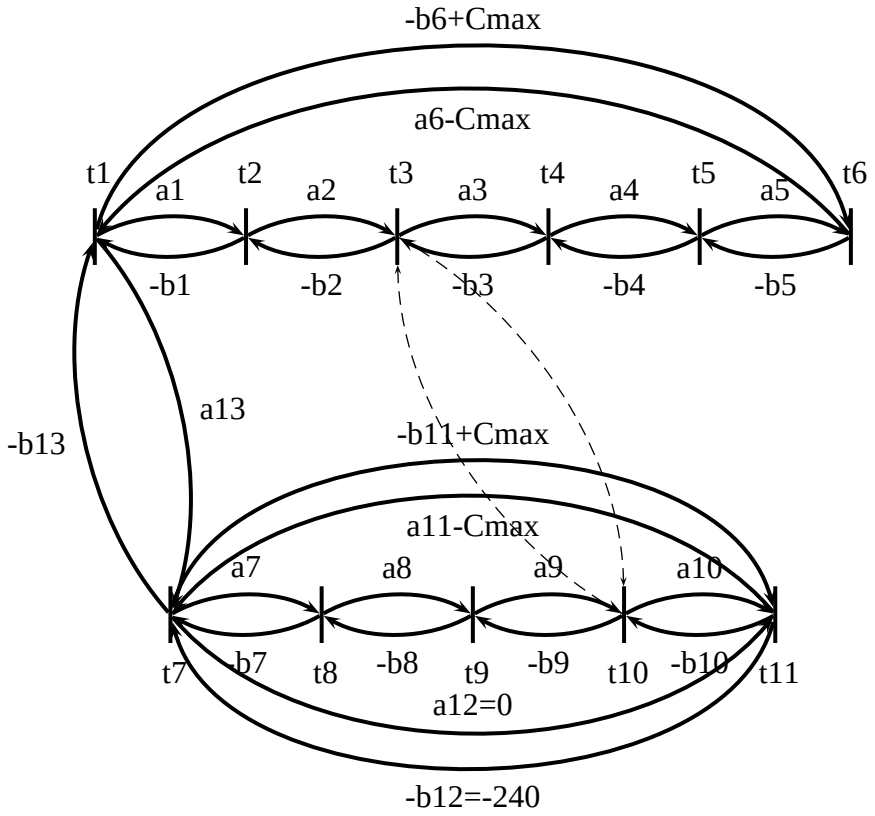

FIGURE 5.4 – Graphe associé à la figure 5.5

Dans le cas d'étude qui nous intéresse, il s'agit d'éviter que le train se retrouve dans la zone de danger alors que les barrières du passage à niveau sont relevées. Il s'agit donc de déterminer une date de franchissement au plus tard de t_4 (entrée zone de danger) pour que t_{11} (barrière ouverte) ne soit pas tirée.

On obtient donc :

$$St_4 < St_{11}$$

$$St_{11min} = (\sum_{i=7}^{11} a_i) \in [0, 240]$$

d'où

$$St_{4max} < 240$$

$$\sum_{i=1}^{3} a_i < 240$$

De plus,

$$a_1 + a_2 = a_7 + a_8 + a_9$$

On obtient donc :

$$a_3 < 240 - (a_2 + a_1)$$

$$a_3 < 240 - (a_7 + a_8 + a_9)$$

et $a_7 + a_8 + a_9$ correspond à la séquence d'allumage des feux et de descente de la barrière qui d'après [Jansen et al.00] dure 18 secondes. On obtient finalement :

$$a_3 < 222s$$

Pour éviter que le train passe la zone de danger alors que les barrières se relèvent, il faut imposer un modèle de commande entre les transitions t_1 et t_4, c'est à dire une place temporisée par l'intervalle de temps [0,222[.

5.3.2.3 Discussion

Nous avons donc utilisé deux méthodes pour synthétiser le contrôle du passage à niveau soumis à des contraintes temporelles. La première méthode nécessite la construction d'un automate temporisé associé au modèle réseau de Petri p-temporel à partir de son graphe de marquage. Ceci peut entraîner un problème d'explosion combinatoire lorsque le nombre d'états du système devient trop important. La deuxième méthode proposée répond à cet inconvénient puisqu'elle ne nécessite pas de construire le graphe de marquage du réseau. Néanmoins, elle ne fonctionne que si le système étudié est périodique.

Ce travail souligne également la difficulté de traiter des contraintes temporelles dans un domaine où la sécurité est une donnée majeure. En effet, le choix fait dans [Jansen et al.00] de lever les barrières après un temps maximal de quatre minutes pour éviter une trop longue attente des usagers de

la route devant un passage à niveau peut être discuté. Notre étude propose, dans un tel environnement, qu'aucun train ne traverse la zone de danger lorsque les barrières se relèvent. Si il ne reçoit pas d'information sur l'état du passage à niveau alors qu'il est arrêté au point de danger, le conducteur du train doit utiliser le contrôle manuel du train pour le faire redémarrer. Alors, il peut prendre la décision de traverser la zone de danger malgré tout, ou d'envoyer à nouveau un signal radio afin de demander la fermeture de la barrière. On peut dès lors s'interroger sur des spécifications plus précises du fonctionnement du passage à niveau, surtout du point de vue de la sécurité.

Par la suite, nous utilisons un modèle des exigences qui capture les exigences temporelles décrites dans la section 5.2.1. Ce dernier est représenté par la figure 5.5.

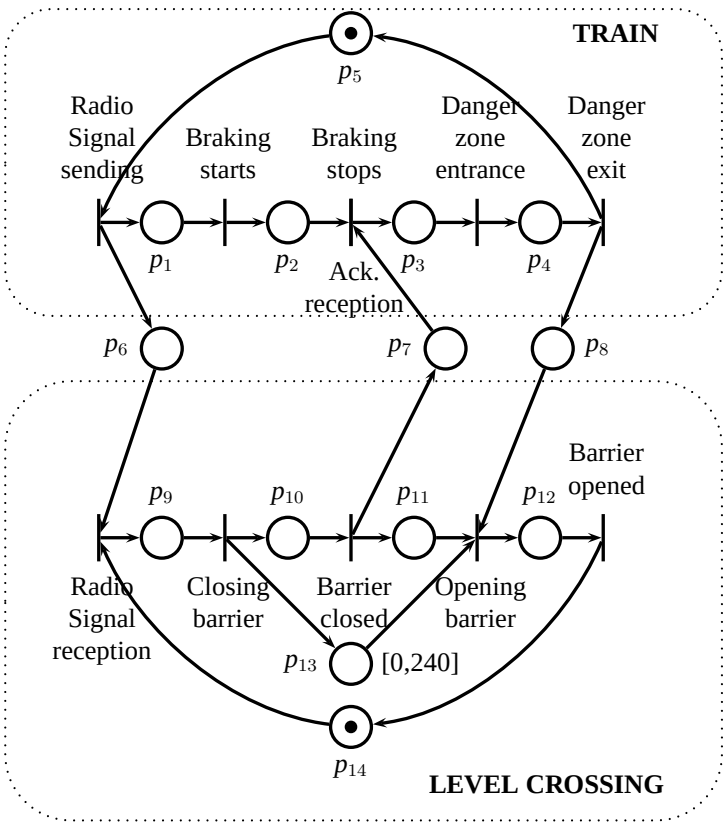

FIGURE 5.5 – Modèle des exigences du cas d'étude.

Ce modèle sert d'étalon pour l'évaluation et la certification de solutions

qui peuvent être proposées pour remplir les spécifications du système. La section suivante décrit une solution trouvée dans [Jansen et al.00].

5.4 Proposition d'un modèle de solution

Le processus correspondant à la solution proposée est maintenant modélisé à l'aide des réseaux de Petri t-temporel. De fait, cet outil a été présenté au chapitre 3 comme particulièrement performant pour décrire une spécification temporelle. L'ensemble des comportements possibles est donc capturé au sein d'un graphe de classes.

5.4.1 Modélisation d'une solution répondant aux exigences de sécurité

Le modèle de la figure 5.6 propose une solution censée suivre les exigences décrites dans les spécifications du systèmes.

Les deux séquences en parallèle représentent d'une part l'évolution du train (p_1, p_2, p_3, p_4) et d'autre part le comportement du passage à niveau $(p_9, p_{10}, p_{11}, p_{12})$. En fait, de la place p_1 à la place p_{14}, le modèle est le même que celui de la figure 5.5, à l'exception notoire de la place p_{13}. Cette place qui modélisait une contrainte de temps de séjour dans le modèle p-temporel est remplacée par une structure de « chien de garde » dans le modèle t-temporel. L'évolution du marquage, une fois que la place correspondant au chien de garde est marquée correspond à un processus de recouvrement qui ramène le système vers un état normal lorsque le temps de séjour n'a pas été respecté.

Sur la figure 5.6, si un jeton reste plus de 240 secondes dans p_{10} (respectivement p_{11}), la transition t_{11} (respectivement t_{12}) est tirée et la procédure dont le but est d'éviter l'ouverture de la barrière lorsque le train est dans la zone du passage à niveau est enclenchée. Cette procédure est censée correspondre à la mort d'un jeton dans la place p_3 du modèle p-temporel de la figure 5.5. Ce comportement spécifique est développé dans le paragraphe suivant.

Premièrement il y a deux possibilités : soit le train, qui a signalé sa présence par radio, est toujours dans la zone du passage à niveau (il y a donc une marque dans une des places allant de p_1 à p_4), soit le train est sorti

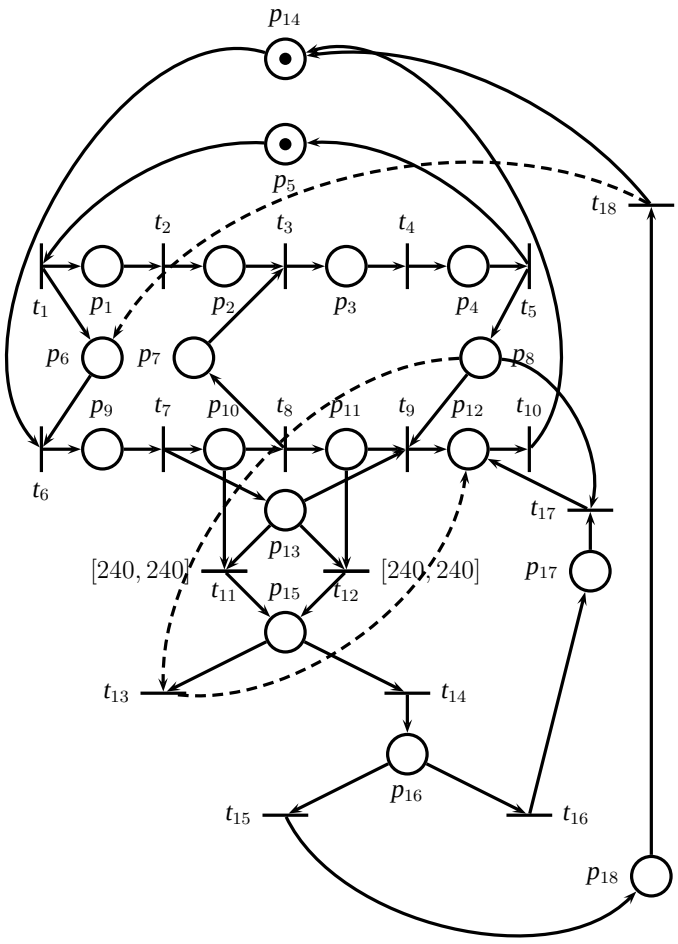

FIGURE 5.6 – Modèle d'une solution répondant aux exigences du cas d'étude.

du passage à niveau (dans ce cas la place p_8 est marquée). Dans le premier scénario (mise à feu de t_{12}), la station de contrôle doit décider si le train peut être arrêté. Dans l'affirmative, ce dernier est arrêté avant la zone de danger et la procédure commençant par l'envoi d'un message radio (t_{15} est tirée) qui déclenche la séquence d'allumage des feux et de fermeture de la barrière. Une temporisation a été associée à la transition t_{18}. C'est une interprétation de la spécification qui tombe sous le sens puisque l'exigence de temps de fermeture maximum de 240 secondes découle des risques de franchissement de barrière par les automobilistes quand le temps d'attente est trop long au passage à niveau. Cet ajout correspond à l'exigence fonctionnelle suivante : les voitures en attente doivent avoir le temps de passer avant la nouvelle fermeture des

barrières. Dans l'autre cas, le train doit passer malgré le dépassement de temps (t_{16} est mise à feu). Si le centre de contrôle détecte que le train n'est plus dans la zone du passage à niveau, il ordonne l'ouverture des barrières (t_{11} est tiré).

Noeuds de la figure 5.5	Noeuds de la figure 5.6
p_i , $i \in \{1, 12\}$	p_i , $i \in \{1, 12\}$
p_{13}	p_i, $i \in 13 \cup \{15, 18\}$
p_{14}	p_{14}

TABLE 5.1 – Correspondance des nœuds entre les deux modèles.

5.4.2 Classes d'états du modèle p-temporel

Une représentation finie des états accessibles est obtenue en construisant un graphe de couverture des classes comme il est décrit dans la partie 4.3.2. Un état est atteignable à partir d'un état initial par l'exécution d'une séquence de tir de transition s. À chaque séquence de mise à feu est associé un temps u. Ainsi une classe d'état associée à une séquence s est l'ensemble des états accessibles à partir de l'état initial en effectuant la mise à feu des transitions de la séquence s.

La figure 5.7 représente le graphe du réseau de Petri p-temporel des classes de la figure 5.5 . Il est composé par des classes C_i et des arcs qui les connectent entre elles. Un arc (C_i, C_j) représente le tir de la transition qui mène de C_i à C_j. Le détail du marquage des 17 classes du graphe de la figure 5.7 est décrit dans le tableau 5.8 ci-dessous.

La figure 5.8 référence les différentes classes d'états de la figure 5.7. De plus, elle contient les marquages recouverts par ces classes, ainsi que les domaines de tirs des marques caractérisant ces mêmes marquages. La figure 5.9 décrit plus précisément les classes d'états avec le formalisme de [Bonhomme01] décrit en annexe.

Une fois le graphe des classes d'états construit pour les exigences, il faut maintenant effectuer le même type de traitement pour le modèle du processus proposé comme solution. Après cela, il sera possible de confronter les spécifications aux exigences.

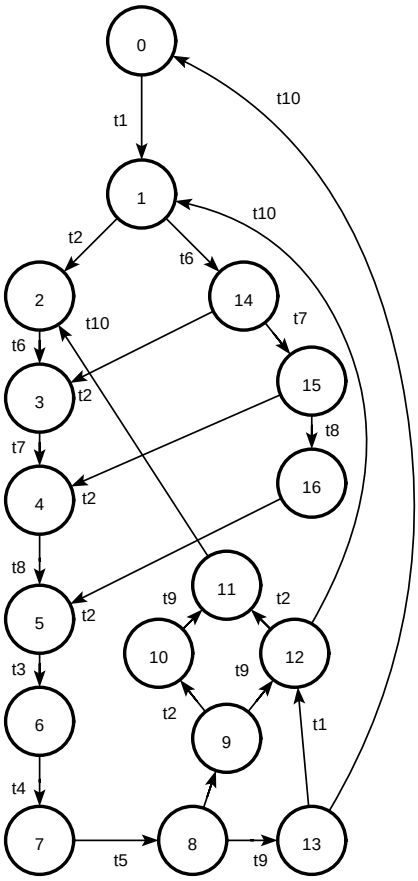

FIGURE 5.7 – Classes d'états du modèle des exigences de la figure 5.5.

Les informations contenues dans le tableau 5.3 nous donne des résultats détaillés quant aux domaines temporels acceptables pour respecter les exigences de sécurité. On pourrait alors s'attendre à des informations d'une même qualité pour la description de la solution proposée par [Jansen et al.00]. En fait, on verra que ce n'est pas le cas. De ce fait, la vérification de consistance temporelle ne pourra pas se faire de manière systématique.

5.4.3 Classes d'états du modèle t-temporel

La figure 5.10 décrit le graphe correspondant au procédé de la figure 5.6 .

La description des 38 classes du graphe de la figure 5.10 est donnée dans

Classe	Marquage	Domaine	Classe	Marquage	Domaine
$C0$	$5, 14$	$\theta_5, \theta_{14} \in [0, \infty[$	$C9$	$1,$ $6, 8,$ $11, 13$	$\theta_1 \in [0, 249],$ $\theta_6, \theta_8 \in [0, 0],$ $\theta_{11}, \theta_{13} \in [0, 240]$
$C1$	$1,$ $6,$ 14	$\theta_1 \in [0, 249],$ $\theta_6 \in [0, 0],$ $\theta_{14} \in [0, \infty[$	$C10$	$2,$ $6, 8,$ $11, 13$	$\theta_2 \in [0, 249],$ $\theta_6, \theta_8 \in [0, 0],$ $\theta_{11}, \theta_{13} \in [0, 240]$
$C2$	$2,$ $6,$ 14	$\theta_2 \in [0, 249],$ $\theta_6 \in [0, 0],$ $\theta_{14} \in [0, \infty[$	$C11$	$2,$ $6,$ 12	$\theta_2 \in [0, 249],$ $\theta_6 \in [0, 0],$ $\theta_{13} \in [0, 240]$
$C3$	$2,$ 9	$\theta_2 \in [0, 249],$ $\theta_9 \in [9, 9]$	$C12$	$1,$ $6,$ 12	$\theta_1 \in [0, 249],$ $\theta_6 \in [0, 0],$ $\theta_{12} \in [0, \infty[$
$C4$	$2,$ $10, 13$	$\theta_2 \in [0, 249],$ $\theta_{10}, \theta_{13} \in [0, 240]$	$C13$	$5, 12$	$\theta_5, \theta_{12} \in [0, \infty[$
$C5$	$2,$ $7,$ $11, 13$	$\theta_2 \in [0, 249],$ $\theta_7 \in [0, 0],$ $\theta_{11}, \theta_{13} \in [0, 240]$	$C14$	$1,$ 9	$\theta_1 \in [0, 249],$ $\theta_9 \in [9, 9]$
$C6$	$3, 11, 13$	$\theta_3, \theta_{11}, \theta_{13} \in [0, 240]$	$C15$	$1,$ $10, 13$	$\theta_1 \in [0, 249],$ $\theta_{10}, \theta_{13} \in [0, 240]$
$C7$	$4, 11, 13$	$\theta_4, \theta_{11}, \theta_{13} \in [0, 240]$	$C16$	$1,$ $7,$ $11, 13$	$\theta_1 \in [0, 249],$ $\theta_7 \in [0, 0],$ $\theta_{11}, \theta_{13} \in [0, 240]$
$C8$	$5,$ $8,$ $11, 13$	$\theta_5 \in [0, \infty[,$ $\theta_8 \in [0, 0],$ $\theta_{11}, \theta_{13} \in [0, 240]$			

FIGURE 5.8 – Marquages et instants de tir du graphe 5.7.

le tableau 5.2. La figure 5.2 ne contient pas les domaines de tirs potentiels afin qu'elle reste lisible. La construction de ces classes d'états est décrite en section 4.3.3.1. De plus, l'outil TINA [Berthomieu et al.03] peut être utilisé à cette fin. Par exemple, la classe $C0$ possède les contraintes temporelles suivantes :

- $0 \leqslant \theta_5^1 \leqslant \infty$
- $0 \leqslant \theta_{14}^1 \leqslant \infty$

En fait, le modèle de la figure 5.6 est une représentation simplifiée de la spécification technique, qui ne représente pas explicitement le comportement du calculateur. En effet, ce dernier n'est pas décrit dans la spécification proposée. Le mécanisme temporel de chien de garde est ainsi biaisé.

Classe	TS	TE	Classe	TS	TE
$C0$	$TS(0) = t_1$	$TE = NULL$	$C9$	$TS(9) = t_2, t_9$	$TE(8,9) = \infty$ $TE(9,8) = 0$ $TE(5,9) = 231$ $TE(4,9) = 240$
$C1$	$TS(1) = t_2, t_6$	$TE(0,1) = \infty$ $TE(1,0) = 0$	$C10$	$TS(10) = t_9$	$TE(8,9) = \infty$ $TE(10,9) = 0$ $TE(8,10) = 240$
$C2$	$TS(2) = t_6$	$TE(1,2) = \infty$ $TE(2,1) = 0$ $TE(1,2) = \infty$	$C11$	$TS(11) = t_{10}$	$TE(10,11) = \infty$ $TE(11,10) = 0$ $TE(8,11) = \infty$ $TE(9,11) = \infty$
$C3$	$TS(3) = t_7$	$TE(2,3) = \infty$ $TE(3,2) = 0$	$C12$	$TS(10) = t_2$	$TE(9,10) = \infty$ $TE(10,9) = 0$ $TE(4,10) = 240$ $TE(5,10) = 231$ $TE(8,10) = 240$
$C4$	$TS(4) = t_8$	$TE(3,4) = 9$ $TE(4,3) = -9$ $TE(2,4) = \infty$	$C13$	$TS(9) = t_{10}$	$TE(8,9) = \infty$ $TE(9,8) = 0$ $TE(4,9) = 240$
$C5$	$TS(5) = t_3$	$TE(4,5) = 9$ $TE(5,4) = -9$ $TE(2,5) = \infty$ $TE(3,5) = 249$	$C14$	$TS(2) = t_7$	$TE(1,2) = \infty$ $TE(2,1) = 0$
$C6$	$TS(6) = t_4$	$TE(5,6) = \infty$ $TE(6,5) = 0$ $TE(2,6) = \infty$	$C15$	$TS(3) = t_8$	$TE(2,3) = 9$ $TE(3,2) = -9$ $TE(1,3) = \infty$
$C7$	$TS(7) = t_5$	$TE(6,7) = \infty$ $TE(7,6) = 0$ $TE(5,7) = 231$ $TE(4,7) = 240$	$C16$	$TS(4) = t_2$	$TE(3,4) = 9$ $TE(4,3) = -9$ $TE(1,4) = \infty$
$C8$	$TS(8) = t_9$	$TE(7,8) = \infty$ $TE(8,7) = 0$ $TE(4,8) = 240$			

FIGURE 5.9 – Classes d'états du modèle p-temporel.

5.5 Projection des espaces d'états

Les classes soulignées dans le tableau 5.2 sont celles qui sont directement incluses dans la projection sur l'espace des exigences (il s'agit donc du graphe de couverture du modèle p-temporel). Il faut rappeler que les trajectoires d'état qui démarrent par le déclenchement du chien de garde sur le modèle t-temporel ne correspondent pas aux exigences.

Cette proposition n'est pas à condamner pour autant. En fait, aucune procédure de recouvrement ou mode dégradé n'est décrite dans les exigences. La projection sur l'espace des exigences élimine donc toutes les classes d'état

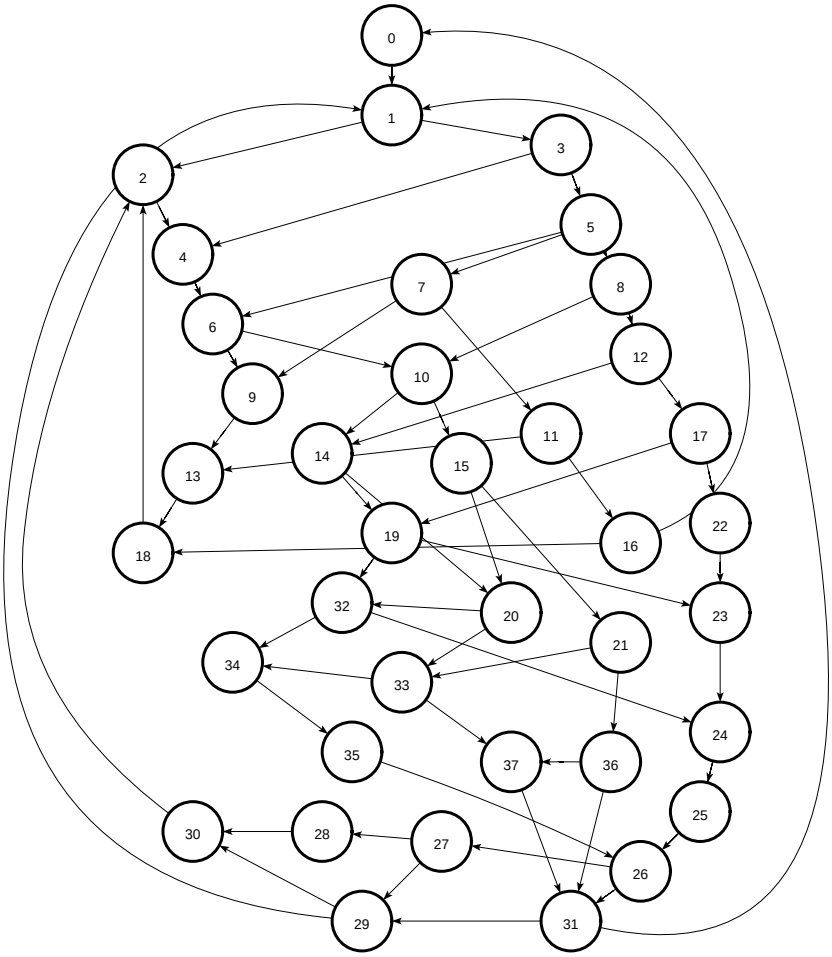

FIGURE 5.10 – Classes d'états du modèle de solution.

qui ne sont accessibles que par le tir des transitions t_{11} ou t_{12}. Cette projection restreint donc les classes du tableau 5.2 aux classes qui sont soulignées : celles pour lesquelles les exigences sont remplies.

Le lecteur averti sera resté sur sa faim concernant l'analyse du comportement temporel. Le tableau 5.2 ne contient pas les intervalles temporels associés aux classes d'états. En fait, la spécification de [Jansen et al.00] précise que le conducteur de la motrice, aidé d'un calculateur embarqué, doit contrôler la vitesse du train de façon à pouvoir s'arrêter avant la zone de danger d'une part et à être en mesure de franchir le passage à niveau dans les temps d'autre part. Ainsi, quelles que soient les exigences temporelles, la

Classe	Marquage	Classe	Marquage	Classe	Marquage
$\underline{C0}$	$5, 14$	$C13$	$2, 16$	$C26$	$5, 8, 17$
$\underline{C1}$	$1, 6, 14$	$C14$	$2, 7, 15$	$C27$	$1, 6, 8, 17$
$\underline{C2}$	$2, 6, 14$	$\underline{C15}$	$3, 11, 13$	$C28$	$2, 6, 8, 17$
$\underline{C3}$	$1, 9$	$C16$	$1, 18$	$\underline{C29}$	$1, 6, 12$
$\underline{C4}$	$2, 9$	$C17$	$1, 7, 16$	$\underline{C30}$	$2, 6, 12$
$\underline{C5}$	$1, 10, 13$	$C18$	$2, 18$	$\underline{C31}$	$5, 12$
$\underline{C6}$	$2, 10, 13$	$C19$	$2, 7, 16$	$C32$	$3, 16$
$C7$	$1, 15$	$C20$	$3, 15$	$C33$	$4, 15$
$\underline{C8}$	$1, 7, 11, 13$	$\underline{C21}$	$4, 11, 13$	$C34$	$4, 16$
$C9$	$2, 15$	$C22$	$1, 7, 17$	$C35$	$5, 8, 16$
$\underline{C10}$	$2, 7, 11, 13$	$C23$	$2, 7, 17$	$\underline{C36}$	$5, 8, 11, 13$
$C11$	$1, 16$	$C24$	$3, 17$	$C37$	$5, 8, 15$
$C12$	$1, 7, 15$	$C25$	$4, 17$		

TABLE 5.2 – Marquages correspondants aux classes d'états du modèle t-temporel.

spécification est réputée y répondre. Par contre, la traçabilité des exigences temporelles doit être mise en œuvre afin que cette belle ellipse textuelle ne reste pas lettre morte. Les exigences temporelles capturées dans le graphe des classes p-temporelles doivent être transmises dans les modes opératoires du conducteur et dans le cahier des charges du calculateur. Cette remarque est plus subtile qu'il n'y paraît, car les intervalles temporels du graphe des classes intègrent les couplages minimaux entre les réseaux de Petri p-temporels : modélisation et validation d'exigences temporelles exigences. Si ces couplages sont négligés, l'assemblage de la solution ne passera pas l'étape de validation quand bien même les exigences sur chacun des composants seraient localement remplies.

5.6 Discussion

La figure 5.6 présente la solution spécifiée par Jansen pour répondre aux exigences sources. Le travail présenté dans cette section a mis en évidence une exigence implicite. En effet, le temps maximum de fermeture de la barrière est de 240 secondes. Cette exigence peut rentrer en conflit avec une exigence

implicite, mais essentielle : la barrière ne doit jamais s'ouvrir lorsqu'un train est dans la zone de danger. Plusieurs interprétations ont du être faites sur les spécifications textuelles de [Jansen et al.00]. Ces interprétations se sont appuyées sur les exigences sources. C'est une illustration de la nécessité de tracer les exigences. Comme le modèle des exigences ne décrit que le comportement nominal, on ne peut rien conclure sur la validité de la procédure de recouvrement qui constitue l'essentiel de la contribution de [Jansen et al.00]. Il faudrait donc réécrire les exigences en s'aidant d'une approche plus formelle. Une sémantique plus appropriée que le langage naturel est notamment souhaitable pour la spécification temporelle. De fait, des imprécisions ont été révélées par la modélisation en réseaux de Petri t-temporels. Ces imprécisions sont surprenantes dans une spécification de référence de la littérature scientifique ferroviaire. Elles plaident pour l'intégration d'outils formels fournissant une sémantique temporelle rigoureuse.

Chapitre 6

Implantation d'une solution : contribution et perspectives

Sommaire

Les exigences de correction et de sécurité amènent à l'utilisation de méthodes de spécification formelles capables de les évaluer et de les garantir. Nous avons justifié dans les chapitres précédents du choix des réseaux de Petri comme outil de modélisation des systèmes à événements discrets critiques. Leur représentation graphique en font de plus un outil de communication efficace. Le chapitre 4 a permis d'obtenir un modèle de solution respectant les spécifications, particulièrement les exigences temporelles, au moyen d'un réseau de Petri t-temporel. L'idée ici est d'associer les capacités des réseaux de Petri, qui en font un modèle lisible et expressif, avec la puissance des systèmes de preuve de la méthode B. Nous proposons une approche visant à transformer de façon systématique le modèle de processus en machine abstraite B afin de poursuivre une procédure formelle B classique.

Notre objectif est d'utiliser deux méthodes formelles afin de développer des spécifications des systèmes complexes qui modélisent les propriétés dynamiques. Il s'agit donc de proposer un cadre méthodologique cohérent permettant d'expliciter des transformations d'un modèle réseau de Petri en un modèle B.

Le caractère formel peut aussi avoir des inconvénients : en effet, l'intégration de plusieurs méthodes formelles est difficile. Les spécifications peuvent d'une part être redondantes ou contradictoires. Elles sont d'autre part difficiles à analyser et à comprendre, car la définition sémantique de l'intégration pose souvent problème.

En fait, le problème dépend des propriétés et des caractéristiques souhaitées pour la combinaison.

On peut distinguer deux approches principales pour la dérivation de modèles :

1. L'approche compilée (ou par traduction) :
2. l'approche interprétée (ou par méta-modélisation) :

Ainsi, l'intérêt de la transformation de modèle est de pouvoir ensuite utiliser les outils de la méthode B pour valider et pour vérifier des propriétés sur le système. De plus, elle permet d'obtenir à la fin du processus de raffinements successifs un code prouvé, exécutable et implémentable. La section 6.1 propose d'effectuer une discrétisation temporelle pour transformer le modèle de processus en réseau de Petri de haut-niveau. En effet, nous avons proposé dans [Defossez et al.08a] une méthode permettant de traduire ce type

de réseau en machine abstraite. Enfin, la section 6.2 présente une perspective intéressante pour nos travaux : la transformation de la solution validée en machine abstraite dans le cadre de l'ingénierie dirigée par les modèles.

6.1 Discrétisation du temps et transformation par construction

Nous proposons une démarche intégrant la structuration et la modélisation du comportement du système en réseaux de Petri à partir d'un cahier des charges et la génération d'une spécification abstraite B à partir de ce réseau de Petri. On distingue dans cette phase quatre étapes distinctes : la modélisation, l'interprétation, la formalisation et enfin la traduction.

6.1.1 Discrétisation du temps

Dans le chapitre 4.2.1, nous avons argumenté pour que les variables de temps restent continues aussi loin que possible dans le processus de conception. À ce niveau, nous sommes dans la phase d'implantation d'une solution qui est complètement spécifiée et validée. Même s'il serait souhaitable de la faire encore plus loin, c'est ici que nous nous autorisons à passer à un temps discret. En effet, on trouve dans la littérature des propositions de transformations de réseaux de Petri de haut-niveau, annotés par un langage de premier ordre vers une spécification B [Bon00, Defossez et al.08a]. Pour pouvoir exploiter le résultat ci-dessus, nous devons donc transformer le modèle de processus décrit au moyen d'un réseau de Petri t-temporel en réseau de Petri de haut-niveau. Comme le réseau de Petri de haut-niveau est annoté avec un langage de premier ordre exprimé sur des ensembles finis, les intervalles continus de temps doivent être discrétisés. Cette étape est délicate car il y a perte de précisions sur une variable considérée comme critique pour la sécurité. Cette problématique devrait faire l'objet de travaux spécifiques plus approfondis.

6.1.2 Construction de machines B à partir d'un réseau de Petri coloré

Ceci a fait l'objet d'une publication [Defossez et al.08a] sélectionné pour un numéro spécial « Safety and security » à paraitre.

6.1.2.1 Les réseaux de Petri de haut-niveau :

Les réseaux de Petri de haut-niveau manipulent des jetons structurés et sont annotés par un langage logique du premier ordre. Le marquage du réseau de Petri est un multi-ensemble de jetons, et le tir d'une transition équivaut à la transformation de multi-ensembles. On peut distinguer plusieurs formes de réseaux de Petri de haut-niveau : les réseaux de Petri prédicats/transitions, les réseaux de Petri colorés ou les réseaux de Petri algébriques. Un réseau de Petri coloré est un réseau de Petri classique auquel on a ajouté un ensemble de couleurs pour distinguer les jetons entre eux. Ils sont basés sur un langage de premier ordre typé.

6.1.2.2 Modélisation et interprétation du cahier des charges

Dans la méthode proposée par [Bon00], il s'agit de représenter le système décrit par le cahier des charges de façon graphique, sous la forme d'un réseau de Petri. La capture des exigences fonctionnelles et sécuritaires constitue le point central de cette phase. L'identification des actions élémentaires (traitements, fonctionnalités) des données du système, des contraintes à respecter et des aspects dynamiques (synchronisation, parallélisme..) permet de structurer le modèle. L'objectif est de capturer un maximum d'informations avec la structure du réseau de Petri afin de simplifier au maximum les annotations en langage naturel et par conséquent les formules mathématiques qui en découlent.

Cette étape de la méthode ne nous est pas utile. En effet, nous disposons, après la phase de discrétisation, d'un modèle réseau de Petri de haut-niveau représentant une solution validée du système. Il nous reste maintenant à transformer ce modèle en machine abstraite.

6.1.2.3 Transformation en machines abstraites

Dans cette dernière phase, on applique au réseau de Petri obtenu une transformation en machines abstraites B. L'approche consiste à définir, dans un premier temps, une machine propre au système spécifié en réseau de Petri, puis à compléter cette machine en intégrant les définitions de propriétés structurelles d'un réseau de Petri. Enfin, on termine par la spécification des propriétés dynamiques du réseau de Petri.

6.2 Perspective : ingénierie dirigée par les modèles

Nous envisageons d'utiliser l'ingénierie dirigée par les modèles (IDM) afin de simplifier, voire même d'éviter, le problème de la discrétisation du temps.

6.2.1 La transformation de modèles : Conception de métamodèles

Un méta-modèle est une spécification explicite d'une abstraction. Cette spécification permet d'identifier l'ensemble des concepts utilisés pour exprimer des modèles ainsi que les différentes relations entre ces concepts. Il définit donc la terminologie à adopter pour définir les modèles.

Les avantages d'une démarche IDM sont :

- la maîtrise de la complexité des éléments techniques existants,
- l'indépendance vis-à-vis des plateformes technologiques,
- la production plus rapide des résultats techniques au travers de l'affinement des modèles,
- l'amélioration de la communication au sein de l'organisation (référence commune non ambigüe pour permettre un dialogue efficace entre les différents intervenants).

Chaque langage est décrit dans un métamodèle (en général, un diagramme de classe UML, complété de contraintes logiques). Un exemple de métamodèle pour les réseaux de Petri à notamment été proposé par [Breton et Bézivin].

Un métamodèle peut être implémenté sous la forme d'un « langage spécifique à un domaine » (DSL). Il est alors interprété comme des types de

données et des fonctions utilisés pour adapter un langage général à un domaine particulier.

6.2.2 Métamodèle de réseau de Petri

Cette section aborde les réseaux de Petri selon un point de vue ingénierie dirigée par les modèles. Plusieurs métamodèles candidats ont été trouvés dans la littérature, la proposition décrite ci-dessous nous semble la plus pertinente.

Le méta-modèle proposé ici est défini dans le contexte IDM proposé par l'OMG puisque sa définition se base sur le MOF (Méta-Object Formalism). L'idée de départ est qu'un méta-modèle définit un ensemble de concepts et de relations.

Le Petri Net Marking Langage (PNML) est un format d'échange standard (ISO/IEC 15909) basé sur XML pour les réseaux de Petri. Il est flexible et extensible : il supporte tous les types de réseaux. Il est sans ambiguïté de sorte qu'une représentation PNML détermine un réseau de façon unique. Le "PNML Core model" définit les concepts communs à tous les types de réseaux de Petri. Ensuite, les "Petri nets type definitions" sont des métamodèles UML pour définir les concepts spécifiques à des réseaux de Petri particuliers. Actuellement, PNML est un standard pour la syntaxe de transfert de trois types particuliers de réseaux de Petri : les réseaux places/transitions, les réseaux de Petri de haut-niveau et les réseaux symétriques. Il définit également le PNML core model, concept commun à tous les réseaux de Petri.

PNML est extensible et ouvert à de nouveaux types de réseaux de Petri. De plus, il permet le transfert de résultats associés à l'analyse de réseaux de Petri. Afin d'obtenir cette flexibilité, le format de transfert considère le réseau de Petri comme un graphe dirigé et labellisé, où tout type d'information spécifique est représenté dans une catégorie "labels". Ces labels peuvent être associés aux noeuds, aux arcs, aux pages ou avec le réseau lui-même. Le PNML peut représenter n'importe quel type de réseau, il n'y a pas de restrictions aux labels.

6.2.3 Métamodèle de machines abstraites B

En ce qui concerne la méthode B, [Idani06] propose un métamodèle dans sa thèse. Ce dernier donne une vue structurelle de haut niveau, reprenant les

notions définissant une syntaxe de B. D'abord, la spécification d'un métamodèle correspondant à une machine abstraite est proposée. S'ajoutent à cela un métamodèle exprimant le typage en B et un autre relatif aux compositions et aux raffinements.

6.2.4 Règles de transformation

Lorsque les métamodèles des réseaux de Petri t-temporels et ceux des machines abstraites B auront été confirmés, il restera encore la plus grande partie du travail, à savoir la formalisation des règles de transformation pour passer de manière automatique d'un modèle à l'autre.

Une règle de transformation définit la manière dont un ensemble de concepts du métamodèle source et transformé en un ensemble de concepts du métamodèle destination.

L'exécution de l'ensemble de ces règles doit permettre de construire l'ensemble du modèle de sortie. Le processus de transformation se compose de trois phases :

1. la vérification de la condition : analyse du modèle source pour détecter la présence d'un ensemble de concepts correspondant à ce que la règle attend en entrée ;
2. l'exécution : les concepts validant la condition sont transformés en accord avec la règle ;
3. la création : un ensemble de concepts sont générés dans le modèle de sortie, dont les champs sont remplis par les variables affectées lors de l'exécution.

Ces règles sont constituées de langages déclaratifs, qui décrivent ce qui est créé par la règle, et de langages impératifs, qui décrivent comment la règle est exécutée.

Les propositions faites ici ne sont qu'un ébauche de réflexion. Ce travail fait l'objet d'une recherche dans le cadre d'un Master au laboratoire ESTAS et devrait se poursuivre par une thèse.

Conclusion Générale

L'objectif du travail présenté dans ce mémoire est de développer une méthodologie d'aide à l'évaluation des systèmes à événements discrets soumis à de fortes contraintes temporelles. Il s'inscrit dans le contexte métier bien précis de la sécurité dans les transports ferroviaires.

Il est évidemment impossible de démontrer a priori qu'un système est sans erreur ; il s'agit plutôt de proposer un processus de conception et de contrôle permettant d'obtenir l'intime conviction que le système est proche du zéro défaut.

Dans ce sens, le contexte méthodologique des études de sécurité ferroviaires repose sur les techniques de sûreté de fonctionnement. La démarche d'évaluation de la sécurité vient donc s'ajouter au processus de conception du système. À chaque étape du cycle de vie, une démarche de certification doit valider cette évaluation. Il est donc nécessaire de fournir aux experts et aux certificateurs des méthodes et des outils pour cette évaluation de la sécurité.

Cette thèse porte sur le développement d'un processus de conception se basant sur des méthodes formelles pour l'évaluation qualitative et quantitative des propriétés d'un système automatisé soumis à des contraintes temporelles de sécurité.

Dans le cadre de l'ingénierie des exigences, il est nécessaire d'identifier et de formaliser un ensemble d'exigences afin de construire un modèle du système permettant de couvrir, d'analyser et d'évaluer cet ensemble. Au sein d'un cahier des charges, les exigences temporelles font partie des contraintes les plus courantes et les plus critiques. Ces dernières affinent les besoins une fois les grandes fonctions du système explicitées.

Parmi les problèmes que comporte la représentation d'un cahier des charges, il y a la tentation d'orienter la modélisation en vue d'une solution pré-

sélectionnée. En effet, une perception a priori du système et de ses besoins peut amener à privilégier certaines solutions et donc exclure certaines autres alternatives. Il nous semble indispensable de nous appuyer sur une approche instrumentée comportant des étapes clairement identifiées.

Pour répondre à ces préoccupations, nous nous sommes orientés vers l'utilisation de modèles et de méthodes formelles, relevant des systèmes à événements discrets, afin de couvrir les dimensions comportementales et structurelles. De plus, nous nous plaçons dans le cadre de l'ingénierie système et dans un contexte métier fort. Pour atteindre cet objectif, nous proposons une approche basée sur une utilisation complémentaire d'outils pertinents pour chaque phase du processus de conception. En outre, il nous semble important de disposer de langages de modélisation à la fois explicites, voire graphiques, et permettant l'expression d'une certaine abstraction.

Dans cette optique, le modèle réseau de Petri est approprié :

- il représente la structure et la comportement dynamique du système,
- il possède de fortes bases formelles.

Plus particulièrement, l'intérêt que nous avons porté aux contraintes temporelles nous a amené à utiliser différentes extensions temporelles de cet outil. Les réseaux de Petri t-temporels et p-temporels, parfois présentés dans la littérature comme concurrents, prouvent chacun leur utilité dans une phase de l'analyse. Tout d'abord, nous nous sommes attachés à expliquer en quoi les réseaux de Petri temporels étaient une solution pertinente pour la spécification des exigences temporelles. Ensuite nous nous sommes servis de cet outil pour modéliser notre cas d'étude. Nous construisons deux modèles, un modèle des exigences et un modèle de processus, afin de vérifier la prise en compte de toutes les exigences contenues dans le cahier des charges tout au long du processus de développement.

Que ce soit pour la description des exigences par les p-temporels ou la modélisation des spécifications à l'aide des t-temporels, les imprécisions et les problèmes de consistance des descriptions textuelles apparaissent de manière impitoyable. Ainsi, nous avons proposé des méthodes qui nous permettent d'analyser les modèles proposés et de valider la traçabilité des exigences nécessaires.

Ensuite, la capture du comportement dans un graphe de classe d'état permet des vérifications systématiques de consistance entre les exigences et

les spécifications. Le cas d'étude typique du passage à niveau a donc servi de support illustratif des outils qui ont été présentés dans ce mémoire. Il nous a permis d'appliquer notre méthode puisqu'il propose d'une part un ensemble d'exigences constituant un cahier des charges assez fourni, et d'autre part une solution visant à répondre à ces exigences. Par ailleurs, il a mis en évidence la difficulté d'exprimer et de tracer toutes les exigences tout au long du processus de conception. Ainsi, nous avons montré que, certes, les exigences de haut-niveau étaient bien respectées par la solution proposée, mais qu'une exigence implicite existe en mode dégradé. Finalement, notre méthodologie a permis d'évaluer la consistance entre le modèle des exigences et le modèle de solution en mode normal, mais nécessite l'avis d'un spécialiste pour évaluer et corriger le fonctionnement en mode dégradé.

Les techniques permettant d'extraire les propriétés d'un modèle p-temporel pour construire une commande conforme aux exigences ont été présentées (section 4.2). Une première méthode de synthèse de commande a été proposée, reposant sur l'utilisation d'un automate associé. Cette méthode est comparée avec une autre approche s'appuyant sur l'analyse structurelle d'une certaine catégorie de réseaux de Petri : les graphes d'événements fortement connexes. Ces méthodes sont discutées et permettent de construire un modèle des exigences capturant et vérifiant toutes les exigences temporelles auxquelles est soumis le système.

Notre proposition méthodologique se poursuit par la construction d'un modèle de solution au moyen de réseau de Petri t-temporel. Des outils spécifiques à ce modèle (notamment TINA) permettent d'une part la simulation du système, et d'autre part la vérification de propriétés structurelles et dynamiques. Dans un premier temps, la construction de ce modèle nécessite de respecter certaines règles pour garantir la cohérence structurelle des modèles et la traçabilité des exigences. Ensuite, nous proposons de vérifier si la projection de l'ensemble des classes d'états généré à partir du réseau de Petri t-temporel est inclus dans l'ensemble des classes d'états à partir du réseau de Petri p-temporel.

Pour finir, nous avons commencé à explorer la transformation de notre modèle solution en machine abstraite B afin d'aboutir à une implantation finale au moyen d'un processus B classique. Nous avons proposé de passer par les réseaux de Petri coloré après discrétisation du temps ; ces derniers

sont transformés par construction en machines B.

En ce qui concerne les perspectives, en amont des travaux exposés dans ce mémoire, il nous semble pertinent de s'intéresser à l'extraction des exigences contenues dans le cahier des charges et à leur formalisation. Pour cela, nous pensons appliquer à un cahier des charges des techniques de traitement automatique du langage afin de fournir un modèle exploitable, avec dans un premier temps l'utilisation de l'outil semi-formel UML.

Pour que les méthodes proposées dans ce mémoire puissent concrètement être utilisées par des experts du domaine ferroviaire, il est important qu'elles soient outillées. Il parait donc indispensable de fournir un logiciel permettant une aide à la construction, à la comparaison et l'exploitation des modèles, et qui permette une validation naturelle de la traçabilité des exigences temporelles.

Comme nous l'avons exprimé au chapitre 6, nous avons commencé à étudier la transformation de notre modèle de solution en machine abstraite B pour une implantation de notre système. Il nous parait pertinent d'utiliser les approches proposées par l'ingénierie dirigée par les modèles pour formaliser cette transformation et prouver l'équivalence des modèles.

Pour finir, nous tenons à insister sur la nécessité d'exprimer toutes les exigences et de les tracer tout au long du processus de conception en évitant absolument les formulations implicites. L'utilisation d'outils et de méthodes formelles et graphiques nous parait indispensable pour fournir une aide à la certification des système critiques, en particulier dans le domaine ferroviaire.

Bibliographie

[Abrial96] Abrial (Jean-Raymond). – *The B Book - Assigning Programs to Meanings*. – Cambridge University Press, août 1996.

[Abrial00] Abrial (Jean-Raymond). – *Event driven sequential program construction*. – Rapport technique, Matisse project, 2000.

[Afis07] AFIS. – *Ingénierie système, Pourquoi ? Comment ?* – Rapport technique, Association Française d'Ingénierie Système, 2007.

[Antoni et al.08] Antoni (M.) et Ammad (N.). – Formal validation method and tools for french computerized railway interlocking systems. *Railway condition monitoring, 4th IET International*, juillet 2008.

[Behm et al.99] Behm (Patrick), Benoit (Paul), Faivre (Alain) et Meynadier (Jean-Marc). – METEOR : A successful application of B in a large project. *FM'99 – Formal Methods*, pp. 369–387. – 1999.

[Berthomieu et al.91] Berthomieu (B.) et Diaz (M.). – Modeling and verification of time dependant systems using time petri nets. *IEEE Transactions on Software Engineering*, vol. 17, n3, mars 1991, pp. 259–273.

[Berthomieu et al.03] Berthomieu (D), Ribet (P.) et Vernadat (F.). – *L'outil TINA : construction d'espaces d'états abstraits pour les réseaux de Petri et réseaux temporels*. – MSR'2003, Hermes, 2003.

[Berthomieu01] Berthomieu (B.). – La méthode des classes d'états pour l'analyse des réseaux temporels - mise en œuvre, extension à la mulit-sensibilisation, modélisation des systèmes réactifs. *MSR'2001, Hermes*, 2001.

[Boehm82] Boehm (B.). – les facteurs du coût logiciel. *Techniques et sciences informatiques*, vol. 1, mai 1982, pp. 5–24.

[Boehm88] Boehm (B.). – A spiral model of software development and enhancement. *Computer*, mai 1988, pp. 61–72.

[Bon00] Bon (Philippe). – *Du cahier des charges aux spécifications formelles : une méthode basée sur les réseaux de Petri de haut niveau.* – Thèse de doctorat, Université des Sciences et Techniques de Lille, juin 2000.

[Bonhomme01] Bonhomme (Patrice). – *Réseaux de Petri p-temporels : contribution à la commande robuste.* – Thèse de doctorat, Université deSavoie, 2001.

[Booch et al.99] Booch (Grady), Rumbaugh (James E.) et Jacobson (Ivar). – The unified modeling language user guide. *J. Database Manag.*, vol. 10, n4, 1999, pp. 51–52.

[Boucheneb et al.93] Boucheneb (H.) et Berthelot (G.). – Towards a simplified building of time petri nets reachability graph. *Proceedings of PNPM'93*, pp. 45–56. – 1993.

[Boulanger et al.06] Boulanger (Jean-Louis), Bon (Philippe) et Mariano (Georges). – From UML to B - a level crossing case. *COMPRAIL*, pp. 351–359. – 2006.

[Boulanger06] Boulanger (Jean-Louis). – *Expression et validation des propriétés de sécurité logique et physique pour les systèmes informatiques critiques.* – Thèse

de doctorat, Université de Technologie de Compiègne, 2006.

[Boyer01] Boyer (Marc). – *Contribution à la modélisation des systèmes à temps contraint et application multimédia*. – Thèse de doctorat, Université Toulouse III, Paul Sabatier, 2001.

[Brandin et al.94] Brandin (B.) et Wonham (W.M.). – Supervisory control of timed discrete event systems. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*. pp. 329–341. – IEEE Transaction on Automatic Control, 1994.

[CEI98] *Norme CEI 61508 - Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité*. – Rapport technique, Norme internationale, 1998.

[Chapurlat07] Chapurlat (Vincent). – *Vérification et validation de modèles de systèmes complexes : application à la modélisation d'entreprise*. – Habilitation à diriger des recherches, Université de Montpellier II, 2007.

[Charbonnier96] Charbonnier (François). – *Commande supervisée des systèmes à événements discrets*. – Thèse de doctorat, Institut national polytechnique de Grenoble, 1996.

[Chretienne83] Chrétienne (P.). – *Les réseaux de Petri temporisés*. – Thèse de doctorat, Université Pierre et Marie Curie, Paris VI, 1983.

[Collart dutilleul et al.98] Collart Dutilleul (S.) et Denat (J.P.). – p-time petri nets and the hoist scheduling problem. *IEEE Conference on Systems, Man, and Cybernetics (SMC'98)*, pp. 558–563. – San Diego, CA, USA, october 1998.

[Collart dutilleul et al.06] Collart Dutilleul (S.), Defossez (F.) et Bon (P.). – Safety requirements and p-time petri nets : a level crossing case study. *IMACS multiconference on computational engineering in systems applications (CESA)*, pp. 1118–1123. – Pékin, Chine, octobre 2006.

[Combacau91] Combacau (M.). – *Commande et surveillance des systèmes à événements discrets : Application aux ateliers flexibles.* – Thèse de doctorat, Université Paul Sabatier, Toulouse, 1991.

[Culita et al.01] Culita (J.), I.Caramihai (S.) et A.M.Stanescu. – Monitoring the desynchronized dysfunctions in flexible manufacturing systems. *IFAC LSS2001 Symposium.* – Bucarest, Roumanie, 2001.

[Declerck et al.08] Declerck (P.), Guezzi (A.) et Gros (C.). – Temps de cycle des graphes d'événements temporisés et p-temporels. *CIFA, Conférence internationale francophone d'automatique*, pp. CD–ROM. – Bucarest, Roumanie, Septembre 2008.

[Defossez et al.07] Defossez (F.), Collart Dutilleul (S.) et Bon (P.). – Formal methods and temporal safety requirements : a level crossing case study. *FORMS/FORMAT*, pp. 220–230. – Braunschweig, Allemagne, octobre 2007.

[Defossez et al.08a] Defossez (F.), Bon (P.) et Collart Dutilleul (S.). – Taking advantage of some complementary modelling methods to meet critical system requirement specificatins. *Comprail, 11th international conference on computer system design and operation in the railway systems*, pp. 220–230. – Tolède, Espagne, septembre 2008.

[Defossez et al.08b] Defossez (F.), Collart Dutilleul (S.) et Bon (P.). – Synthèse de contrôle de systèmes à événements discrets temporisés : application au passage

à niveau. *CIFA, Conférence internationale francophone d'automatique*, pp. CD–ROM. – Bucarest, Roumanie, Septembre 2008.

[Defossez et al.08c] Defossez (F.), Collart Dutilleul (S.) et Bon (P.). – Temporal requirements checking in a safety analysis of railway systems. *FORMS/FORMAT*, pp. 273–280. – Budapest, Hongrie, october 2008.

[Defossez et al.10] Defossez (F.), Collart Dutilleul (S.) et Bon (P.). – A formal model of requirements. *Open Transportation Journal*, 2010.

[Didi alaoui05] Didi Alaoui (M.). – *Étude et supervision des graphes d'événements temporisés et temporels : vivacité, estimation et commande.* – Thèse de doctorat, ISTIA, Université d'Angers, 2005.

[Dir04] *Directive 2004/49/CE du parlement européen et du conseil concernant la sécurité des chemins de fer communautaires.* – Rapport technique, Directive européenne, 2004.

[Dir07] *Directive 2007/59/CE du parlement européen et du conseil relative à la certification des conducteurs de train assurant le conduite de locomotives et de trains sur le réseau communautaire.* – Rapport technique, Directive européenne, 2007.

[Dir08] *Directive 2008/57/CE du parlement européen et du conseil relative à l'interopérabilité du système ferroviaire au sein de la communauté.* – Rapport technique, Directive européenne, 2008.

[El koursi et al.07] El-Koursi (E.), Mitra (S.) et Bearfield (G.). – Harmonizing safety management systems in the european railway sector. *Safety science monitor*, p. 14. – 2007.

[EN100] *Norme EN 50126 - Applications ferroviaires - Spécification et démonstration de la fiabilité, de la dis-*

ponibilité, de la maintenabilité et de la sécurité. – Rapport technique, Norme Européenne, 2000.

[EN101] *Norme EN 50128 - Applications ferroviaires - de signalisation, de télécommunication et de traitement logiciels pour systèmes de commande et de protection ferroviaire.* – Rapport technique, Norme Européenne, 2001.

[EN103] *Norme EN 50129 - Applications ferroviaires - Systèmes de signalisation, de télécommunications et de traitement systèmes électroniques de sécurité pour la signalisation.* – Rapport technique, Norme Européenne, 2003.

[ERT09] *Compendium on ERTMS - European Rail Traffic Management System.* – Rapport technique, UIC, 2009.

[Essame02] Essame (D.). – *La méthode B et l'ingénierie système. Réponse à un appel d'offre.* – Rapport technique, IUT-Nantes, Université de Nantes, 2002.

[Friedenthal et al.08] Friedenthal (Sanford), Moore (Alan) et Steiner (Rick). – *A Practical Guide to SysML : The Systems Modeling Language.* – 2008.

[Gti07] GTI (Groupe de Travail Ingénierie des Exigences). – *Ingénierie des exigences.* – Rapport technique, Association Française d'Ingénierie Système, 2007.

[Hellouin02] Hellouin (L.). – *Contribution à l'ingénierie des exigences et à la traçabilité.* – Thèse de doctorat, LAAS-CNRS, INP Toulouse, 2002.

[Hoare et al.85] Hoare (C. A. R.) et Hoare (C. A. R.). – Communicating sequential processes. *Communications of the ACM*, vol. 21, 1985, pp. 666–677.

[Huet et al.01] Huet (G.), Kahn (G.) et Paulin-Mohring (Ch.). – *The* Coq *Proof Assistant - A tutorial - Version 7.1*, octobre 2001. http ://coq.inria.fr.

[Hull et al.05] Hull (E.), Jackson (K.) et Dick (J.). – *Requirements engineering.* – 2005.

[Idani06] Idani (Akram). – *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 PhD, Université de Grenoble 1, November 2006.

[IEC05] *IEC 62267 - railway application - automated urban guided transport safety requirements.* – Rapport technique, Norme internationale, 2005.

[IEE90] *Standard IEEE 610.12 - Glossary of software engineering terminology.* – Rapport technique, International electronic and electrical engineers standard, 1990.

[IEE99] *Standard IEEE 1220 - Standard for application and management of the systems engineering process.* – Rapport technique, International electronic and electrical engineers standard, 1999.

[Incose06] INCOSE. – *INCOSE Systems Engineering Handbook : a guide for system life cycle processes and activities.* – 2006volume 3.0.

[ISO95] *Management de la qualité et assurance de la qualité : vocabulaire.* – Rapport technique, norme Afnor, 1995.

[Jansen et al.00] Jansen (L.) et Schnieder (E.). – *Traffic Control Systems Case Study : Problem Description and a Note on Domain-based Software Specification.* – Rapport technique, Technical University of Braunschweig, 2000.

[Jerbi et al.04] Jerbi (N.), Collart Dutilleul (S.), Craye (E.) et Benrejeb (M.). – Robust control of multi-product job-shops in repetitive functioning mode. *IEEE Conference on Systems, Man, and Cybernetics*

(SMC'04), pp. 4917–4922. – The Hague, Netherlands, october 2004.

[Jones89] Jones (C. B.). – *Systematic Software Development Using VDM.* – Prentice Hall, 1989.

[Khansa97] Khansa (W.). – *Réseaux de Petri P-temporels : Contribution à l'Etude des systèmes à Evenements Discrets.* – Thèse de doctorat, Université de Savoie, 1997.

[Khoudour et al.08] Khoudour (L.), Ghazel (M.), Heddebaut (M.) et El-Koursi (E.). – Sécurité aux interactions route/rail : le cas du passage à niveau. *Transport Environnement Circulation*, pp. 34–39. – 2008.

[Kirwan et al.02] Kirwan (B.), Hale (A.) et Hopkins (A.). – *Changing regulation : controlling risk in society.* – Insights into safety regulation, 2002.

[Koornstra03] Koornstra (M.). – *Transport safety performance in the EU : a statistical overview.* – Rapport technique, European transport safety council, 2003.

[Krogh et al.91] Krogh (B.), Magott (J.) et Holloway (L.). – On the complexity of forbidden state problems for controlled marked graphs. *Proceedings of the IEEE internal conference on decison and control*, pp. 85–91. – Brighton, dec 1991.

[Lafit92] Lafit (S.). – Optimisation of invariant criteria for event graph. *Proceedings of the IEEE Transaction on Automatic Control*, pp. 547–555. – 1992.

[Laprie et al.95] Laprie (J.C), Arlat (J.), Blanquart (J.P.), Costes (A.), Crouzet (Y.), Y. (Deswarte), Fabre (J.C.), Guillermain (H.), Kanoun (K.), Mazet (C.), Powell (D.), Rabéja (C.) et Thévenod (P.). – *Guide de la sûreté de fonctionnement.* – Laboratoire d'Ingénierie de la Sûreté de fonctionnement, Toulouse, France, 1995.

[Laval00] Laval (C.). – Traçabilité des exigences ou traçabilité des logiques de conception et de maitrise des risques ? *AFAV*. – 2000.

[Leveson et al.93] Leveson (N.) et Turner (C.). – Investigation of the therac - 25 accidents. *IEEE Computer*, vol. 26, n 7, mars 1993, pp. 18–41.

[Lions96] Lions (J.L.). – *Ariane 5 - Flight 501 failure*. – Rapport technique, The inquiry board, 1996.

[Maler et al.95] Maler (O.), Puneli (A.) et Sifakis (J.). – On the synthesis of discrete controllers for timed systems. *STACS'95*, pp. 229–242. – 1995.

[Meinadier98] Meinadier (J.P.). – *Ingénierie et intégration des systèmes*. – Hermès, 1998, études et logiciels informatiques édition.

[Menasche82] Menasche (M.). – *Analyse des réseaux de Petri temporisés et application aux systèmes disctribués*. – Toulouse, Thèse de PhD, Université Paul Sabatier, 1982.

[Merlin74] Merlin (P.). – *A Study of the Recoverability of Computer Systems*. – Irvine, California, Thèse de PhD, University of California, 1974.

[Meyer01] Meyer (E.). – *Développement formels par objets : utilisation conjointe de B et UML*. – LORIA, Thèse de PhD, Université Nancy 2, 2001.

[Murata89] Murata (T.). – Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, vol. 77, n4, avril 1989, pp. 541–574.

[Penalva99] Penalva (Jean-Michel). – Méthode sagace : le systémographe. *CEA*, 1999.

[Petri62] Petri (Carl Adam). – *Kommunikation mit Automaten*. – Thèse de PhD, Bonn : Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[Ramachandani74] Ramachandani (C.). – Analysis of asynchronous concurent systems using petri nets. *Technical report N°120*. – 1974.

[Ramadge et al.87] Ramadge (P. J. G.) et Wonham (W. M.). – Supervisory control of a class of discrete event processes. *SIAM Journal of Control Optimization*, vol. 25, n 1, janvier 1987, pp. 206–230.

[Ramamoorthy et al.80] Ramamoorthy (C. V.) et Ho (Gary S.). – Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Trans. Software Eng.*, vol. 6, n5, 1980, pp. 440–449.

[Royce70] Royce (W.). – Managing the development of large software systems. *IEEE Wescon*, août 1970, pp. 1–9.

[Sava et al.08] Sava (A.), Achour (Z.), Rezg (N.) et Malki (N.). – Synthèse d'une structure de controle tolérente aux fautes basée sur les réseaux de petri. *CIFA, Conférence internationale francophone d'automatique*, pp. CD–ROM. – Bucarest, Roumanie, Septembre 2008.

[Sava01] Sava (A.). – *Sur la Synthèse de la Commande des Systèmes à évenements discrets temporisés*. – Thèse de doctorat, Laboratoire d'Automatique de Grenoble - INPG, 2001.

[Sifakis77] Sifakis (J.). – Use of petri nets for performance evaluation. *Symposium on modelling and evaluation, IFIP*, pp. 75–93. – 1977.

[Sifakis80] Sifakis (J.). – Performance evaluation of systems using nets. *Lecture notes in computer science*, vol. 84, 1980, pp. 307–319.

[Spivey89] Spivey (J. M.). – *The Z notation : a reference manual*. – Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1989.

[Van eijk et al.89] Van Eijk (P.) et Diaz (Michel). – *Formal Description Technique Lotos : Results of the Esprit Sedos Project*. – New York, NY, USA, Elsevier Science Inc., 1989.

[White paper01] White paper. – *European transport policy for 2010 : time to decide*. – 2001.

[Zuberek91] Zuberek (W. M.). – Timed petri nets definitions, properties, and applications. *Microelectronics and Reliability*, vol. 31, n4, 1991, pp. 627–644.

Annexe

Analyse énumérative des RdPs p-temporels

Pour l'évaluation des instants de tirs, l'analyse énumérative originale proposée par [Bonhomme01] requiert un nouveau formalisme. C'est en effet la situation temporelle des jetons qui est fondamentale. Ainsi, un jeton est identifié par la place dans laquelle il se trouve, mais également il porte l'information de son instant d'arrivée dans cette place, et celle de son instant de consommation.

Ces considérations nous amènent à la définition de la fonction TOK explicitée en définition 6.1.

Définition 6.1 (TOK)
$TOK(j,n) = p \in P$, il existe un jeton contenu dans p, créé par le j^{eme} tir de transition et consommé par le n^{eme}, avec j<n où P est l'ensemble des places du réseau considéré.

Dans le cas général, $TOK(j,n)$ est un multi-ensemble. Dans cette thèse ne sont considérés que les arcs unitaires. Par ailleurs, la construction des ensembles TOK dépend directement de la séquence de tir étudiée.

Par convention, $\bigcup TOK(0,m)$ est l'ensemble des jetons présents dans le réseau à l'instant initial si ce dernier est quasi-vivant.

À partir de ces ensemble $TOK(j,n)$, les durées de disponibilité minimum et maximum des marques peuvent être établies comme exprimé en définition 6.2

Définition 6.2 (Disponibilités)

$$Dsmin(j,n) = \begin{cases} \max(a_i), p_i \in TOK(j,n) \\ 0 \ si \ TOK(j,n) = \emptyset \end{cases}$$

$$Dsmax(j,n) = \begin{cases} \min(b_i), p_i \in TOK(j,n) \\ +\infty \ si \ TOK(j,n) = \emptyset \end{cases}$$

La méthode présentée ici fait référence à la notion d'instants de tir, introduite sur le modèle t-temporel par [Boucheneb et al.93].

Ainsi la quantité x_1 représente l'instant de tir de la première transition tirée, $x_2 + x_3$ la durée écoulée entre le premier et le troisième instant de tir. Par conséquent, l'instant absolu de tir de la ieme transition mise à feu correspond à la somme $\sum_{k=1}^{i} x_k$.

Représentation des q^{eme} premiers instants de tir

Les quantités c_{uq} et d_{jk} sont introduites pour représenter plus simplement les résultats :

Définition 6.3 ()

$$c_{uq} = \begin{cases} Dsmin(u,q) \ si \ u \in SEN(q) \\ 0 \ sinon \end{cases}$$

$$d_{jk} = \begin{cases} Dsmax(j,k), siTOK(j,k) \neq \emptyset \\ +\infty sinon \end{cases}$$

Les quantités c_{uq} font référence à un ensemble de jetons validant une transition particulière et initiant le franchissement de cette dernière. Les quantités d_{jk} quant à elles font référence à l'ensemble des jetons présents dans le réseau à l'instant d'observation.

Une séquence de transitions $t_1, t_2, .., t_q$ peut être franchie aux instants 1, 2, .., q si et seulement s'il existe $x_1, x_2, .., x_q$ positifs tels que :

$$S(q) = \begin{cases} c_{0k} \leqslant x_1 \leqslant d_{0k}, k \in \{1,n\} \\ \max(c_{0k}, c_{1k}) \leqslant x_1 + x_2 \leqslant \min(d_{0k}, d_{1k} + x_1), k \in \{2,n\} \\ ... \\ \max(c_{jk} + \sum_{s=0}^{j} x_s) \leqslant \sum_{s=0}^{q} x_s \leqslant \min(d_{jk} + \sum_{s=0}^{j} x_s), j \in \{0, q-1\}, k \in \{q,n\} \end{cases}$$

Représentation des classes d'états

Si la classe C a été générée par l'instant de tir $(q-1)$, cette dernière sera alors définie par le triplet (M,TS,TE) avec :

- M le marquage du p-RdP,
- TS(i) = $\{t \in T, t$ soit sensibilisée par le ieme instant de tir $\}$,
- TE(i,j) = $\begin{cases} Smax(i+1,j,q-1) si (i<j) \\ -Smin(j+1,i,q-1) si (i>j) \\ 0 \ si \ (i=j) \end{cases}$

$TE(i,j)$ fournit la durée maximale ou minimale écoulée entre l'instant de tir i et l'instant de tir j.

Condition de tir d'une transition partant d'une classe

Partant de la classe d'états C obtenue par l'instant de tir (q-1), une transition t_q, sensibilisée par le $i^{ème}$ instant de tir (ie $t_q \in TS(i)$), est franchissable la $q^{ème}$ si et seulement si :

$\forall j \in \{0, i\}$

$$\max(c_{jq} \leqslant \min(\min(d_{rk} + TE(j,r)))$$

Calcul de la classe suivante

Partant de la classe d'états C=(M,TS,TE) créée à l'instant de tir (q-1), le tir de la transition $t_q \in TS(i)$ conduira à la classe C'=(M',TS',TE') obtenu ainsi :

$$M' = M - Pre(.,t_q) + Post(.,t_q)$$

$$TS'(q) = \{t \in T\}, tsensibiliséepar M'$$

$$TE'(j,q) = \min(\min(d_{rk} + TE(j,r)))$$

$$TE'(q,j) = \min(\min(TE(r,j) - c_{rq}), 0)$$

Graphe des classes d'états

L'arbre des classes d'états est construit à l'aide d'une relation d'accessibilité entre les différentes classes calculées. Les règles de tir entre les différentes

classes permet de construire au fur et à mesure l'arbre des classes d'états, ayant pour racine la classe initiale.

Équivalence entre classes

Deux classes d'états C=(M,TS,TE) et C'=(M',TS',TE') créées respectivement aux instants de tir (q-1) et (q'-1) seront dites équivalentes si :

1. M = M'
2. il existe une bijection g telle que :
$$\begin{cases} TS(j) = TS'(g(j)), \\ TE(i,j) = TE'(g(i), g(j)) \end{cases}$$

Graphe des classes et propriétés

Les objectifs de la construction du graphe des classes d'états sont les mêmes que ceux du graphe des marquages accessibles classique, ou du graphe des classes d'états du modèle t-temporel.

Il s'agit de la connaissance des états accessibles à partir d'un état donné, la caractérisation des séquences de tir réalisables et donc la vérification des propriétés de caractère borné, de vivacité, de fonctionnement stationnaire et, dans le cas particulier des réseau de Petri p-temporels, de la recherche de séquences marques-vivantes.

Titre : Modélisation discrète et formelle des exigences temporelles pour la validation et l'évaluation de la sécurité

Résumé :

Le but de ce rapport est de présenter une méthode globale de développement à partir de spécifications informelles, depuis la modélisation graphique des exigences temporelles d'un système ferroviaire critique jusqu'à une implantation systématique au moyen de méthodes formelles. Nous proposons d'utiliser ici les réseaux de Petri temporels pour décrire le comportement attendu du logiciel de contrôle-commande à construire. Tout d'abord nous construisons un modèle des exigences prenant en compte toutes les contraintes que doit vérifier le système. Nous proposons des outils et des méthodes capables de valider et de vérifier ce modèle. Ensuite, il s'agit de construire un modèle de processus solution. Ce modèle illustre des exigences techniques relatives à un choix technologique ou architectural. L'objectif est double : tout d'abord il est nécessaire de vérifier la traçabilité des exigences ; ensuite, il faut vérifier que l'ensemble des exigences sources sont bien implémentées dans la solution préconisée et dans sa mise en œuvre. Enfin, nous proposons des pistes permettant de transformer le modèle de processus en machine abstraite B afin de poursuivre une procédure formelle B classique. Finalement, le cas d'étude du passage à niveau, composant critique dans le domaine de la sécurité ferroviaire est décrit.

Title : Temporal requirements checking in a safety analysis of critical systems

Abstract :

The introduction of new European standards for railway safety, coupled with an increasing use of software technology changes the method of development of critical railway systems. Therefore the appropriate safety level of critical systems has to be proved in order to obtain the necessary approval from the authorities. Accordingly a high level of reliability and correctness must be reached by the use of mathematical proofs and then formal methods. This document describes a methodology to analyse the safety of timed discrete event systems. First, our goal is to take out the forbidden state highlighted by a p-time Petri net modelling. This model deals with the requirements of the considered system and has to contain all the constraints that have to be respected. Then we aim at describing a process identified as a solution of the system functionning. This method consists in exploring all the possible behaviours of the system by means of the construction of state classes. Finally, we check if the proposed process corresponds to the requirements model previously built. Our case-study is the level crossing, a critical component for the safety of railway systems.

Laboratoires :
INRETS-ESTAS, 20 rue Élisée Reclus, F-59650, Villeneuve d'Ascq, France
LAGIS-UMR CNRS 8530, École Centrale de Lille, Villeneuve d'Ascq, France