Thèse

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

En vue de l'obtention du titre de
Docteur de l'Université de Toulouse, délivré par l'Université Toulouse III - Paul Sabatier

Spécialité : **Systèmes Industriels**

Par
Nabil SADOU

Aide à la conception des systèmes embarqués sûrs de fonctionnement

Soutenance le 06 Novembre 2007 devant le jury :

Robert VALETTE	Président
Jean-François AUBRY	Rapporteur
Pierre-Etienne LABEAU	Rapporteur
Dimitri LEFEBVRE	Examinateur
Hamid DEMMOU	Directeur de thèse
Jean-Claude PASCAL	Co-Directeur de thèse

Avant-propos

Le travail présenté dans ce mémoire a été réalisé au Laboratoire d'Analyse et d'Architecture des Systèmes (L.A.A.S.) du C.N.R.S de Toulouse, au sein du groupe de recherche Ingénierie Systèmes et Intégration (ISI).

Je remercie tout d'abord Monsieur Malik GHALLAB et Monsieur Raja CHATILA, directeurs successifs du LAAS, de m'avoir accueilli dans ce laboratoire pendant ces années de travail.

Je remercie également Monsieur Mario PALUDETTO de m'avoir accueilli dans son groupe de recherche ISI.

Ce travail a été dirigé par Hamid DEMMOU et Jean-Claude PASCAL à qui j'exprime ma gratitude pour leurs conseils, leur soutien et leur confiance mais aussi pour le climat d'amitié qu'ils ont su instaurer entre nous.

Je tiens à remercier Monsieur Robert VALETTE, directeur de recherche au LAAS – CNRS de m'avoir fait l'honneur de présider mon jury de soutenance.

Je suis très reconnaissant envers Monsieur Jean-François AUBRY, Professeur à l'Institut National Polytechnique de Lorraine, d'avoir accepté avec Monsieur Pierre Etienne LABEAU, Professeur à l'université libre de Bruxelles, d'étudier mes travaux et d'en être les rapporteurs ainsi que pour l'intérêt et l'attention qu'ils ont accordés à cette étude.

J'exprime toute ma gratitude à Monsieur Dimitri LEFEBVRE Professeur à l'université du Havre d'avoir accepté d'examiner ce travail.

Je voudrais étendre cette gratitude à Monsieur Robert VALETTE et Monsieur Abd-El-Kader SAHRAOUI à qui je dois également la réalisation et la finalisation de mes travaux.

Que ceux qui ont participé de près ou de loin à la réalisation de ce travail trouvent ici le témoignage de ma reconnaissance.

Que tous les collègues (de recherche ou d'enseignement) que j'ai pu côtoyer durant ces années de thèse, reçoivent ici le témoignage de mon amitié. La bonne ambiance n'a rendu ces années que plus agréables.

Enfin, je dédie ce paragraphe à tous ceux qui m'ont aidé et supporté dans cette épreuve. Je pense à mes amis et à ma famille grâce à qui j'ai pu en arriver jusque là.

Table des matières

Table des figures

Introduction générale

De nos jours, les systèmes embarqués envahissent notre quotidien. Beaucoup de fonctions, jadis, réalisées manuellement sont aujourd'hui automatisées grâce à l'électronique et l'informatique embarquée. Ceci a apporté des améliorations notables en termes de confort, un confort auquel, il est de plus en plus difficile de renoncer.

La part grandissante de l'électronique et de l'informatique n'est pas sans conséquences sur les méthodes de conception. La complexité croissante des systèmes embarqués nécessite une adaptation des processus, méthodes et outils existants par rapport aux spécificités de ces systèmes pour mieux répondre aux exigences, notamment celles liées à la sûreté de fonctionnement. En effet, la préoccupation des industriels est de proposer à leurs clients des produits intégrant les nouvelles innovations technologiques avec une qualité et des performances de plus en plus améliorés mais aussi des produits de plus en plus sûrs.

La criticité de ces systèmes nécessite de garantir un niveau de fiabilité et de sécurité convenable. Des études de sûreté de fonctionnement doivent être menées tout au long du cycle de développement du système. Ces études permettent une meilleure maîtrise des risques et de la fiabilité. Les points faibles sont ainsi mis en évidence et permettent aux concepteurs de spécifier des stratégies de reconfiguration avant la phase de prototype réel et les tests réels. Les études de sûreté de fonctionnement doivent être menées au plus tôt dans la phase de conception, afin de réduire les coûts et le nombre de prototypes nécessaires à la validation du système.

Un système embarqué est un système commandé et contrôlé par un calculateur combinant diverses technologies (mécanique, hydraulique ou électrique). Il doit répondre à des impératifs de criticité, de réactivité, d'autonomie, de robustesse et de fiabilité. L'interaction se fait par l'intermédiaire de capteurs qui fournissent des informations sur l'état du système. Ces informations sont utilisées par le calculateur pour générer la commande à appliquer au système au moyen d'actionneurs.

La présence d'une dynamique continue liée à la partie énergétique et d'une dynamique événementielle liée à la commande, aux défaillances et reconfiguration donne aux systèmes

embarqués un caractère hybride. Un des problèmes principaux rencontrés lors d'une étude de sûreté de fonctionnement de ces systèmes est la prise en compte de manière efficace et réaliste des dépendances. Dans de tels systèmes, la séquence de fonctionnement normal ou dangereux résulte de l'occurrence commune de deux types d'événements. Les premiers sont liés à l'évolution déterministe des paramètres physiques susmentionnés, alors que les deuxièmes sont dus aux sollicitations et aux défaillances des composants du système et sont donc de nature aléatoire.

Les méthodes classiques de sûreté de fonctionnement atteignent vite leurs limites face à la complexité de ces systèmes. Les méthodes combinatoires (arbres de défaillance, arbres d'événements, diagrammes de fiabilité) permettent uniquement d'identifier et d'évaluer les combinaisons des événements menant à l'occurrence d'une catastrophe. Elles ne tiennent pas compte de l'ordre d'occurrence des événements qui les composent. Ceci exclut toute possibilité de prendre en compte la dépendance et les délais entre événements. Les méthodes basées sur les systèmes à événements discrets répondent bien à la problématique d'ordre entre événements (réseau de Petri, automates) mais sont limitées par le problème de l'explosion combinatoire.

A partir de ce constat, et pour contourner tous ces problèmes, [Khalfaoui, 2003] dans sa thèse a utilisé directement le modèle Réseau de Petri (RdP) pour extraire les scénarios redoutés sans générer le graphe d'accessibilité associé. L'approche s'appuie sur la logique linéaire comme cadre formel. En effet, grâce à la logique linéaire, il est possible d'extraire les scénarios menant vers un état critique sans avoir à explorer tout le système. L'approche est locale et se focalise sur la partie pouvant être concernées par l'état critique. L'autre avantage de la logique linéaire est qu'elle permet de construire un ordre partiel de franchissement des transitions. Cette approche est basée sur l'équivalence entre l'accessibilité dans les RdP et la prouvabilité du séquent associé en logique linéaire.

Pour prendre en compte la nature hybride de la dynamique des systèmes embarqués, le choix du modèle s'est porté sur les réseaux de Petri associés aux équations différentielles. Le modèle RdP décrit le fonctionnement nominal, les défaillances et les mécanismes de reconfiguration. Les équations différentielles représentent l'évolution des variables continues de la partie énergétique du système.

L'approche conduit à la génération des scénarios redoutés qui permettent au concepteur de comprendre les raisons de la dérive du système vers l'état critique. Il peut donc prévoir les reconfigurations nécessaires qui permettent d'éviter cet état. A partir d'une connaissance partielle de l'état critique (redouté), il est possible de revenir en arrière à travers la chaîne des relations de cause à effet et d'extraire tous les scénarios possibles menant vers l'état critique. Chaque scénario est donné sous la forme d'un ordre partiel entre les évènements nécessaires à l'apparition de l'évènement redouté.

Khalfaoui propose un algorithme de recherche de scénarios pour mettre en œuvre l'approche. Les limites de cet algorithme sont dues au fait qu'il opère uniquement sur l'aspect discret du modèle et que de nombreux scénarios incohérents vis-à-vis de la dynamique continue sont générés. De plus, l'ordre d'occurrence, dû à cette dynamique continue, des événements n'est pas pris en compte et les scénarios générés ne sont pas minimaux.

[Medjoudj, 2006] dans sa thèse a poursuivi le travail et a introduit partiellement l'aspect continu. En effet, la partie continue est prise en compte grâce à des abstractions temporelles de la dynamique continue. Un modèle réseau de Petri temporel est ainsi déterminé et analysé. La prise en compte partielle de la partie continue a permis d'éliminer un certain nombre de scénarios générés dans la première version et qui sont incohérents avec la dynamique continue, mais pas la totalité.

Objectif de la thèse

L'approche, proposée dans les travaux de [Khalfaoui 03] améliorée et implémentée dans les travaux de [Medjoudj, 2006], basée sur l'extraction des scénarios redoutés directement à partir du modèle réseau de Petri nous semble bien adaptée pour faire face à la complexité croissante des systèmes embarqués en termes d'analyse qualitative. Nous présentons dans ce manuscrit la poursuite des recherches sur cette approche.

Le but est de développer une approche multi-modèles pour la conception des systèmes dynamiques hybrides sûrs de fonctionnement. Pour cela, à partir des travaux existants et présentés ci-dessus, différents aspects, très liés, ont été étudiés.

Pour prendre en compte l'aspect hybride dans toute leur complexité il est nécessaire de développer une approche de modélisation structurée et modulaire. Pour cela, il est nécessaire d'intégrer les travaux basés sur les approches orientées objets [Paludetto, 1991] et [Villani & al, 2004] dans la cadre de la sûreté de fonctionnement. L'algorithme de recherche de scénarios redoutés doit être adapté à cette approche.

Pour que les scénarios redoutés générés soient pertinents, ils doivent satisfaire certaines propriétés. La minimalité est une de ces propriétés. La notion de minimalité doit être alors définie dans le cadre des réseaux de Petri et ensuite intégrée dans l'algorithme de recherche de scénarios redoutés. Ceci permettra de générer uniquement les scénarios minimaux. La complétude des scénarios est une autre exigence à laquelle doit répondre la méthode, la définition formelle de la complétude doit être, elle aussi, intégrée dans la méthode.

Pour prendre en compte la dynamique continue et devant la limitation des approches d'abstraction de cette dynamique, il faut mettre en place une simulation hybride. La simulation hybride doit associer un solveur d'équations différentielles à l'algorithme de recherche de scénarios. C'est l'algorithme de génération de scénarios redoutés qui doit déterminer les équations à intégrer. Ceci permettra d'effectuer une simulation locale et de réduire considérablement le coût de la simulation.

Traditionnellement, dans la conception des systèmes, chaque fonction pouvait être étudiée et développée indépendamment des autres et l'implication de la sûreté de fonctionnement se résumait à la réutilisation de modèles génériques issus du retour d'expérience. Ce type d'approche ne permet pas de prendre en compte les risques liés à l'intégration de plusieurs technologies. Il est donc important de formuler les exigences de sûreté de fonctionnement non seulement localement mais globalement. Cela revient à formuler ces exigences au niveau du système complet et ensuite à les décliner à des niveaux plus bas (jusqu'aux simples

composants). L'ingénierie système (IS) est une démarche méthodologique pour maîtriser la conception des systèmes et produits complexes. Il faut donc intégrer la sûreté de fonctionnement dans les processus de l'ingénierie système en s'appuyant sur la norme IS *EIA-632*. Ceci permet ainsi de formuler, définir, formaliser, analyser les exigences de sûreté de fonctionnement et d'en établir un modèle de traçabilité. Cette traçabilité permet de s'assurer de la prise en compte de ces exigences tout au long du cycle de vie du système.

Plan du document

Ce document est composé de six chapitres. Nous évoquerons dans le premier chapitre les concepts relatifs à la sûreté de fonctionnement, les systèmes embarqués et les difficultés rencontrées lors de leur analyse. Un tour d'horizon des récents travaux dans le domaine de la fiabilité dynamique suivra ensuite. Nous discuterons les différentes méthodes de modélisation et d'analyse en montrant les avantages et limites de chaque méthode. Nous porterons une attention particulière à l'approche de recherche de scénarios redoutés développée au LAAS et qui est à l'origine du travail présenté dans ce manuscrit.

Dans le deuxième chapitre, nous aborderons l'intégration de la sûreté de fonctionnement dans des processus d'ingénierie système. Nous présenterons notre approche basée sur la norme *EIA-632*. La norme sera présentée brièvement puis les processus de la norme seront déployés et traduits en terme de sûreté de fonctionnement.

Le but du troisième chapitre est de présenter et de discuter le problème de modélisation des systèmes dynamiques hybrides. L'étape de modélisation a un rôle crucial dans la conception et l'analyse des systèmes. Le formalisme présenté dans ce chapitre pour la modélisation des systèmes dynamiques hybrides résulte de l'application du paradigme orienté objets (*OO*) aux réseaux de *Petri Prédicats Transitions Différentiels Stochastiques* (*RdP PTDS*). Nous exposerons également les motivations qui nous ont poussés à faire ce choix.

Au cours du quatrième chapitre, nous proposerons une définition formelle basée sur la logique linéaire de la notion de scénario. Cette définition formelle nous permet ensuite de définir formellement la notion de minimalité. En effet, le but final de notre approche est de déterminer les scénarios minimaux, sachant qu'un scénario minimal représente une classe de scénarios. Pour que la détermination des scénarios redoutés soit efficace, outre l'obtention des scénarios minimaux, il est nécessaire de générer tous les scénarios pouvant mener vers un état redouté. La complétude des scénarios est donc une autre exigence. A partir de la définition du scénario minimal, la définition de l'ensemble d'événements complet sera proposée en fin de chapitre.

Dans le cinquième chapitre, nous présenterons la nouvelle version de la méthode de recherche de scénarios redoutés. Elle est basée sur une recherche de cause à effet à partir du modèle réseau de Petri du système et permet de caractériser les scénarios redoutés sous la forme d'un ensemble de relations d'ordre entre les événements menant d'un état de bon fonctionnement à un état redouté. Nous introduirons, toutes les améliorations apportées à l'algorithme, notamment l'introduction de l'approche objet, la formalisation des relations de précédence indirectes, mais aussi la notion de minimalité. Un exemple illustrera la méthode et permettra de montrer les apports du travail de thèse.

Enfin dans le sixième chapitre, une approche de simulation hybride basée sur le couplage de l'algorithme de génération de scénarios et d'un solveur d'équations différentielles sera présentée. Elle permet de prendre en compte la dynamique continue du système sans passer par une abstraction de cette dynamique.

Chapitre 1 : sûreté de fonctionnement des systèmes embarqués

I Introduction

De nos jours, les systèmes embarqués nous entourent et nous sommes littéralement envahis par eux. Il suffit de regarder autour de soi au quotidien pour s'en rendre compte. On retire de l'argent dans un distributeur, c'est un système embarqué. On prend sa voiture, le régulateur de vitesse, la direction assistée, le système de contrôle de trajectoire c'est des systèmes embarqués. Le pilote automatique dans un avion est un système embarqué. Bien qu'installé en général dans une station sol, on considère aussi comme système embarqué le système de contrôle d'un drone et les exemples ne manquent pas.

Dans divers domaines, au cours des ces dernières années, le progrès technologique a fait que beaucoup de fonctions réalisées manuellement, mécaniquement et hydrauliquement se dématérialisent au profit de l'électronique et de l'informatique embarquée. Un confort considérable auquel il est de plus en plus difficile de renoncer.

Dans notre travail, nous nous intéressons plus particulièrement aux systèmes embarqués critiques et temps réel. Les systèmes critiques sont des systèmes dont les pannes peuvent avoir des effets catastrophiques sur l'utilisateur et son environnement. Ils sont en général soumis à des règlements de certification. Les systèmes temps réels doivent satisfaire des contraintes de ponctualité et de qualité de service, compte tenu de la dynamique de leur environnement.

Les systèmes embarqués ne sont pas sans complexité. Ils font intervenir des technologies différentes, en plus de comporter une partie logicielle et une partie matérielle. Cette complexité n'est pas sans conséquence sur l'analyse de la sûreté de fonctionnement. En effet, d'une part toutes ces technologies nouvelles ne bénéficient pas encore d'un retour d'expérience, d'autre part la composante logicielle qui n'existait pas dans les systèmes traditionnels peut être sujette à erreurs.

II Sûreté de fonctionnement, concepts et méthodes

Cette section est une synthèse des principaux concepts de base de la sûreté de fonctionnement. Un bref rappel est donné de toutes les notions définies dans [Laprie & al, 1996], [Laprie & al, 2004] et [Villemeur, 1988]. Le lecteur pourra se référer à ces travaux. Il y trouvera une vue plus détaillée de l'ensemble du domaine.

II.1 Concepts de la sûreté de fonctionnement

La sûreté de fonctionnement regroupe les activités d'évaluation de la fiabilité, de la maintenabilité, de la disponibilité et de la sécurité (FMDS) d'une organisation, d'un système, d'un produit ou d'un moyen. Ces évaluations permettent, par comparaison aux objectifs ou dans l'absolu, d'identifier les actions de construction (ou amélioration) de la sûreté de fonctionnement de l'entité. Ces évaluations sont prédictives et reposent essentiellement sur des analyses inductives ou déductives des effets des pannes, dysfonctionnements, erreurs d'utilisation ou agressions de l'entité.

La sûreté de fonctionnement (SDF) est un ensemble d'outils et de méthodes qui permettent, dans toutes les phases de vie d'un produit, de s'assurer que celui-ci va accomplir la (les) mission(s) pour laquelle (lesquelles) il a été conçu, et ce, dans des conditions de fiabilité, de maintenabilité, de disponibilité et de sécurité bien définies. La SdF doit être prise en compte tout au long du cycle de vie du produit.

Villemeur [Villlemeur, 1988] définit la sûreté de fonctionnement comme la science des défaillances. Elle inclut ainsi leur connaissance, leur évaluation, leur prévision, leur mesure et leur maîtrise. Au sens strict, elle est l'aptitude d'une entité à satisfaire une ou plusieurs fonctions requises dans des conditions données.

Selon [Laprie & al, 2004], la sûreté de fonctionnement d'un système est son aptitude à délivrer un service de confiance justifiée. Cette définition mettant l'accent sur la justification de la confiance, cette dernière peut être définie comme une dépendance acceptée, explicitement ou implicitement. La dépendance d'un système vis-à-vis d'un autre système est l'impact, réel ou potentiel, de la sûreté de fonctionnement de ce dernier sur la sûreté de fonctionnement du système considéré.

La sûreté de fonctionnement peut être définie par les attributs suivants :

- La *fiabilité* : au sens strict, c'est la probabilité qu'une entité fonctionne sur l'intervalle de temps *[0,t]*. Elle représente l'aptitude de l'entité à satisfaire une fonction requise dans des conditions données
- La *maintenabilité* : au sens strict, c'est la probabilité que la maintenance d'une entité défaillante soit achevée à un instant *t* sachant que la défaillance s'est produite à l'instant *t=0*. C'est l'aptitude d'une entité à être maintenue ou remise en état de fonctionnement, état dans lequel elle peut accomplir une fonction requise.

- La *disponibilité* : au sens strict, c'est la probabilité qu'un composant ou un système soit en état de marche à un instant donné *t*. Elle représente l'aptitude d'une entité à être en état d'accomplir une fonction requise.
- La *sécurité-innocuité* : C'est l'aptitude d'un produit à respecter, pendant toutes les phases de vie, un niveau acceptable de risques d'accident susceptible d'occasionner une agression du personnel ou une dégradation majeure du produit ou de son environnement. Elle représente l'absence de conséquence catastrophique pour l'environnement.

Pour [Laprie, 1996], la sûreté de fonctionnement a pour objectif de spécifier, concevoir, réaliser et exploiter des systèmes où la faute est naturelle, prévue et tolérable. Elle est alors la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre.

II.2 Quelques définitions

Défaillance

C'est la cessation d'une entité à accomplir une fonction requise. En fonction de ses effets une défaillance peut être mineure, significative, critique ou catastrophique. En fonction de ses causes, une défaillance peut être première, seconde ou de commande.

Modes de défaillance

C'est l'effet par lequel la défaillance est observée, la manière dont la défaillance apparaît. C'est la forme perceptible du dysfonctionnement d'un produit ou d'une opération.

Causes de défaillance

Les causes entraînent les défaillances et peuvent dépendre de circonstances liées à la conception, la fabrication et/ou à l'emploi. Les causes d'une défaillance peuvent être internes ou externes à l'entité.

Effets d'une défaillance

Pour chaque mode de défaillance recensé, il correspond les effets (conséquences sur la fonction du produit) chez le client en termes d'insatisfaction ou de non-conformité (le client peut être l'utilisateur final du produit ou de l'opération suivante du processus).

Reconfiguration

Une *reconfiguration* est l'action de modifier la structure d'un système qui a défailli (figure I.1), de telle sorte que les composants non défaillants permettent de délivrer un service acceptable bien que dégradé (fonctionner en mode dégradé, c'est tenter de fournir le service jugé indispensable, en manquant de ressources complètes ou fiables). L'exemple de la figure I.1 représente un système reconfigurable. En effet, le switch permet de reconfigurer le système en utilisant la ressource secours en cas de défaillance de la ressource primaire.

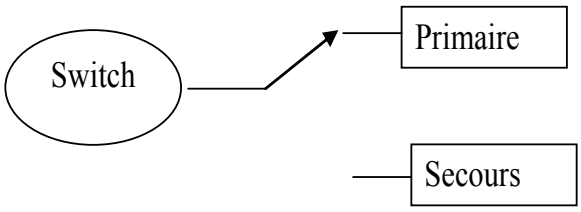


Figure I.1. Reconfiguration dynamique

Evénement redouté

Nous appellerons événement redouté, les conséquences de l'occurrence d'une défaillance unique ou d'une séquence de défaillances menant le système dans une situation de blocage en l'absence de réparation. Le terme « *redouté* » est assez général, il englobe les termes : indésirable, critique, dangereux et catastrophique.

Analyse qualitative

Elle a pour but la démonstration de propriétés et/ou le listage des scénarios de défaillance (un scénario de défaillance est une séquence d'événements menant le système d'un état de fonctionnement normal vers un état redouté). Elle est généralement suivie d'une analyse quantitative.

Analyse quantitative

L'objectif dans l'analyse quantitative est de quantifier les scénarios déterminés suite à l'analyse qualitative en termes de probabilité d'occurrence. C'est suite à cette analyse que le système est validé ou non.

Remarque : Dans certains cas, notamment lorsqu'on s'intéresse à des systèmes très nouveaux, les données de fiabilité font totalement défaut et il faut se contenter d'une étude qualitative. L'analyse qualitative à elle seul peut présenter un intérêt. En effet lorsqu'on dispose de données de fiabilité, elle sert de support à des calculs de probabilités simplifiés.

II.3 Méthodes d'analyse de la sûreté de fonctionnement

L'évaluation de la sûreté de fonctionnement d'un système consiste à analyser les défaillances des composants pour déterminer leurs causes et estimer leurs conséquences sur le service rendu par le système.

L'analyse de sûreté de fonctionnement peut être qualitative et/ou quantitative. Les principales méthodes sont réalisées suivant deux types de raisonnement logique (figure I.2) :

- Approche inductive : raisonnement du particulier au général. Dans ce cas, on recherche les effets d'une défaillance sur le système ou son environnement.

- approche déductive : raisonnement du général au particulier. Dans ce cas on suppose que le système est défaillant et on recherche les causes possibles de la défaillance

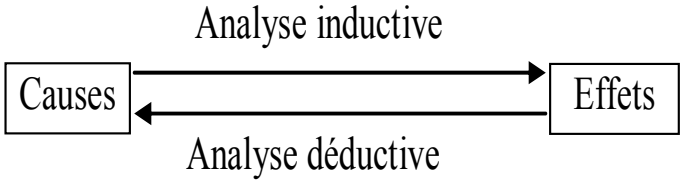


Figure I.2. Différents raisonnements logiques

Dans la partie qui va suivre nous allons passer en revue les principales méthodes classiques d'analyse de sûreté de fonctionnement. Mais avant cela nous présentons d'abord l'analyse fonctionnelle qui est une étape nécessaire pour toute étude de sûreté de fonctionnement.

II.3.1 L'analyse fonctionnelle

Pour analyser les défaillances d'un système, il est nécessaire auparavant de bien identifier à quoi doit servir ce système : c'est à dire de bien identifier toutes les fonctions que ce système doit remplir durant sa vie de fonctionnement et de stockage.

D'après la norme (AFNOR NF X 50-151), l'analyse fonctionnelle est une démarche qui consiste à rechercher, ordonner, caractériser, hiérarchiser et / ou valoriser les fonctions du produit (matériel, logiciel, processus, service) attendues par l'utilisateur.

Une fonction est l'action d'un élément constitutif d'un système exprimée exclusivement en terme de finalité (par ce qu'il « fait »). Chaque fonction doit être exprimée, formulée, par un verbe à l'infinitif suivi d'un ou plusieurs compléments.

L'analyse fonctionnelle est utilisée au début d'un projet pour créer (conception) ou améliorer (reconception) un produit. Elle est un élément indispensable à sa bonne réalisation. On détermine donc, par exemple, les fonctions principales, les fonctions secondaires et les fonctions contraintes d'un produit. Il est important de faire ce recensement afin d'effectuer un dimensionnement correct des caractéristiques du produit.

Une analyse fonctionnelle, précède donc une étude de sûreté de fonctionnement. Une première analyse fonctionnelle dite externe permet de définir avec précision les limites matérielles du système étudié, les différentes fonctions et opérations réalisées par le système ainsi que les diverses configurations d'exploitation. L'analyse fonctionnelle interne permet de réaliser une décomposition arborescente et hiérarchique du système en éléments matériels et/ou fonctionnels. Elle décrit également des fonctions dans le système.

II.3.2 L'Analyse Préliminaire des Risques

L'Analyse Préliminaire des Risques (APR) est une extension de l'Analyse Préliminaire des Dangers (APD). Elle est réalisée après l'analyse fonctionnelle. Elle a pour but d'établir une liste

aussi exhaustive que possible des incidents ou accidents pouvant avoir des conséquences sur la sécurité du personnel ou du matériel [Villemeur, 1988]. Un autre objectif de L'*APR* est d'évaluer la gravité des conséquences liées aux situations dangereuses et les accidents potentiels.

La méthode permet de recenser les dangers et déduire ensuite tous les moyens et toutes les actions correctrices permettant d'éliminer ou de maîtriser les situations dangereuses et les accidents potentiels. Il est recommandé de commencer l'APR dès les premières phases de la conception. Cette analyse sera vérifiée et complétée au fur et à mesure de l'avancement dans la réalisation de système. L'APR permet de mettre en évidence les événements redoutés critiques qui devront être analysés en détail dans la suite de l'étude de sûreté de fonctionnement, en particulier par la méthode des arbres de défaillances.

II.3.3 Analyse des Modes de Défaillance, de leurs Effets et de leurs Criticités

L'AMDEC (analyse des modes de défaillance, de leurs effets et de leur criticité) est une technique spécifique de la sûreté de fonctionnement, mais aussi et surtout une méthode d'analyse de systèmes statique (systèmes au sens large composé d'éléments fonctionnels ou physiques, matériels, logiciels, humains ...) s'appuyant sur un raisonnement inductif (causes conséquences), pour l'étude organisée des causes, des effets des défaillances et de leur criticité.

Cette technique a été utilisée pour la première fois au milieu des années 60 pour l'analyse de la sécurité des avions [Villemeur, 1988] et son utilisation se limitait au début à des études de fiabilité sur le matériel (notamment dans le domaine de l'aéronautique pour tenter de limiter la fréquence et la gravité des accidents aériens).

Elle est désormais extrêmement répandue dans tout ce qui a trait à la sécurité, la maintenance ou la disponibilité. Jadis cantonnée au « simple » domaine du matériel, elle sert désormais de référence en matière de système, de fonctionnel et de logiciel.

Malgré sa popularité et son utilisation dans de nombreux domaines, l'*AMDEC* n'est pas sans inconvénients. Le plus grand inconvénient de la méthode est son incapacité à prendre en compte des défaillances multiples (systèmes redondants) [Faucher, 2004]. L'*AMDEC* considère uniquement les défaillances simples. La méthode des arbres de défaillance (*AdD*) qui sera présentée un peu plus loin dans le manuscrit est elle bien adaptée à l'analyse des systèmes redondants et donc aux défaillances multiples.

II.3.4 Les Arbres de Défaillances

L'analyse par Arbre de Défaillance (*AdD*) [Chatelet, 2000], [Villemeur, 1988] suit une démarche déductive. Elle permet de représenter graphiquement les diverses combinaisons possibles d'événements qui conduisent à la réalisation de l'événement redouté recensé lors de L'*APR* ou de l'*AMDEC*. La représentation graphique d'un *AdD* possède une structure arborescente :

- L'événement sommet représente l'événement redouté,

- Les niveaux successifs forment l'arborescence d'événements où chaque événement est défini par une condition logique sur les événements du niveau inférieur à l'aide d'opérateurs logiques (*ET*, *OU*,…)
- Le processus déductif est poursuivi jusqu'à ce que les événements, dits événements élémentaires (de base), dont on connaît la probabilité d'occurrence et qui sont statistiquement indépendant, soient obtenus.

L'analyse qualitative par arbre de défaillance consiste à déterminer l'ensemble des coupes minimales. Une coupe est un ensemble d'événements entraînant l'événement redouté. Une coupe est minimale lorsque le retrait d'un événement de la coupe n'entraîne plus l'événement redouté (un arbre de défaillance a un nombre fini de coupes minimales). La recherche des coupes minimales se fait à partir d'une transformation de l'arbre en une expression booléenne et l'utilisation des lois de l'algèbre de Boole pour obtenir une expression booléenne réduite de l'événement redouté [Rauzy & Dutuit, 1997]. Une autre méthode basée sur les diagrammes de décision binaire (*BDD*) est présentée dans [Sinnanmon & Andrew, 1997].

L'analyse quantitative consiste à assigner à chaque événement de base une probabilité d'occurrence. Une méthode d'évaluation de la probabilité d'occurrence de l'événement sommet, basée sur les diagrammes de décision binaire est présentée dans [Rauzy & Dutuit, 1997].

L'analyse par arbre de défaillance est largement utilisée dans les études de sûreté de fonctionnement car elle caractérise de façon claire les liens de dépendance, du point de vue dysfonctionnement, entre les composants d'un système. Bien que cette méthode soit efficace, elle présente des limites. L'une de ces limites est que l'ordre d'occurrence des événements menant vers l'état redouté n'est pas pris en compte. Or comme nous allons le voir par la suite, cette notion d'ordre dans les événements menant vers une défaillance est primordiale dans les systèmes embarqués de par leur caractère hybride. Nous présentons dans la section suivante ces systèmes.

II.3.5 Diagramme de Fiabilité

Un Diagramme de Fiabilité (*DdF*) permet le calcul de disponibilité ou la fiabilité du système modélisé, mais avec les mêmes restrictions qu'un Arbre de Défaillance, voire pires (pas d'événements répétés). Tous les chemins entre l'entrée et la sortie décrivent les conditions pour que la fonction soit accomplie. On suppose que les composants n'ont que deux états de fonctionnement (fonctionnement correct ou panne).

III Sûreté de fonctionnement des systèmes embarqués

III.1 Système embarqué

Un ***système embarqué*** est un système électronique, piloté par un logiciel, qui est complètement intégré au système qu'il contrôle. On peut aussi définir un système embarqué comme un système électronique soumis à diverses contraintes.

Le premier système moderne embarqué reconnaissable a été le système de guidage de la mission lunaire Apollo, développé par Charles Stark Draper du Massachusetts Institute of Technology. Chaque mission lunaire était équipée de deux systèmes (*Apollo Guidance Computer*), un chargé du système de guidage inertiel et un pour le Module lunaire.

Un *système embarqué* combine généralement diverses technologies qui relèvent des domaines de la mécanique, de l'hydraulique, de la thermique, de l'électronique et des technologies de l'information.

Un système embarqué (figure I.3) peut être décomposé en quatre entités en interaction : les capteurs, la partie opérative, le système de commande et de reconfiguration et les actionneurs. Les *capteurs* mesurent des grandeurs physiques continues caractéristiques de la partie opérative. Le système de commande et de reconfiguration établit en fonction de ces mesures les actions à réaliser. Les actionneurs agissent sur la partie opérative.

Les systèmes embarqués ont des impératifs. Ce sont principalement:

- **La criticité**: Les systèmes embarqués sont souvent critiques et les systèmes critiques sont presque toujours embarqués. En effet, comme un tel système agit sur un environnement physique, les actions qu'il effectue sont irrémédiables. Le degré de criticité est fonction des conséquences des déviations par rapport à un comportement nominal, conséquences qui peuvent concerner la sûreté des personnes et des biens, la sécurité, l'accomplissement des missions ou encore la rentabilité économique.
- **La réactivité**: Ces systèmes doivent interagir avec leur environnement à une vitesse qui est imposée par ce dernier. Ceci induit donc des impératifs de temps de réponse. Un système embarqué et généralement un système temps réel.
- **L'autonomie**: Les systèmes embarqués doivent en général être autonomes, c'est-à-dire remplir leur mission pendant de longues périodes sans intervention humaine. Cette autonomie est nécessaire lorsque l'intervention humaine est impossible, mais aussi lorsque la réaction humaine est trop lente ou insuffisamment fiable.
- **La robustesse, sécurité et fiabilité**: L'environnement est souvent hostile pour des raisons physiques (chocs, variations de température, impact d'ions lourds dans les systèmes spatiaux, ...) ou humaines (malveillance). C'est pour cela que la sécurité (au sens de la résistance aux malveillances) et la fiabilité (au sens continuité de service) sont souvent rattachées à la problématique des systèmes embarqués.

Au cours du développement d'un système embarqué, le concepteur doit choisir entre plusieurs solutions d'architectures matérielles et logicielles répondant à des critères de performances et de sûreté de fonctionnement exprimés dans les spécifications.

Tout système n'est pas à l'abri d'erreur de conception est donc à l'abri de l'apparition de défaillance au cours de son cycle de vie. Le concepteur doit alors disposer de moyens pour éviter les fautes.

Une fiabilité absolue n'est jamais garantie lors de la conception d'un système. Le concepteur introduit alors des mécanismes de tolérance aux fautes afin d'éviter que des erreurs ou des fautes entraînent une défaillance du système. La redondance est un moyen de tolérer des fautes. Un système redondant est composé d'un élément primaire, un élément redondant pouvant réaliser tout ou une partie des fonctions de l'élément primaire, un élément de détection et un dispositif de réaction à cette erreur. L'élément redondant doit assurer la fonction dédiée au système suite à une *reconfiguration* du système.

Deux types de redondance peuvent être distingués [Aubry, 1987] :

- ***La redondance statique*** : le nombre d'éléments redondés peut être important (redondance *massive*) et chacun d'entre eux participe à la réalisation de la fonction. Il s'agit notamment des dispositifs à *vote majoritaire* pour lesquels le résultat final issu du vote résulte de la comparaison entre les différentes sorties des éléments redondés.
- ***La redondance dynamique*** : l'élément redondant ne participe à la fonction qu'après détection et réaction à l'erreur.

[Laprie, 1996] détaille un certain nombre de stratégies de reconfiguration pouvant être mises en œuvre suite à la détection d'une erreur.

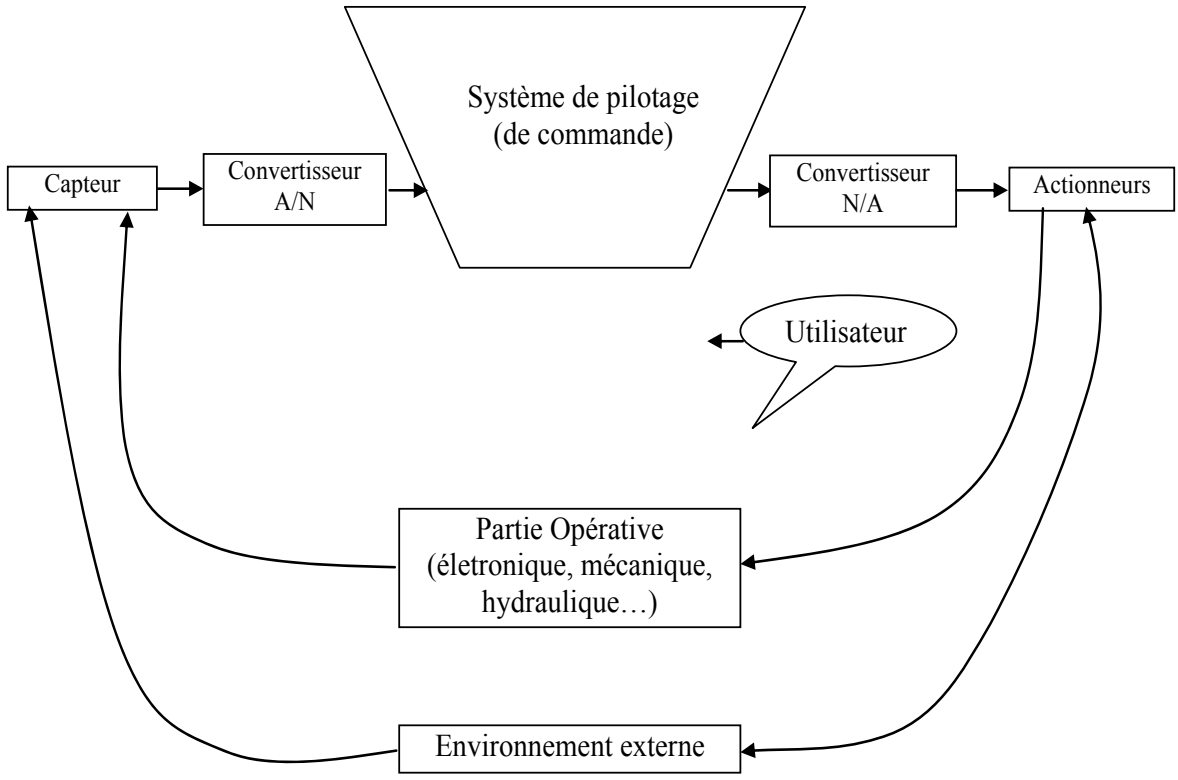


Figure I.3. Système embarqué typique

III.2 Fiabilité dynamique

Un système embarqué combine généralement plusieurs technologies : hydraulique, mécanique, électrique, et électronique. Il présente également des aspects continus et événementiels qui lui donnent un caractère hybride. La dynamique continue représente la partie énergétique du système et la partie événementielle représente les changements d'état discrets, les reconfigurations et les défaillances.

L'analyse de la sûreté de fonctionnement de ces systèmes dynamiques hybrides est souvent appelée fiabilité dynamique [Labeau, 2002]. Le paragraphe suivant donne une définition de cette notion de fiabilité dynamique.

Dans un système dynamique hybride, l'évolution dynamique de l'ensemble des variables physiques associées au processus (température, débit, pression,…) est contrôlé à l'aide d'un dispositif de contrôle-commande en surveillant, le plus souvent, le dépassement de certains seuils caractéristiques. Ainsi des variations de certaines grandeurs physiques provoquent des changements d'état du dispositif de contrôle qui agit en conséquence sur ces grandeurs pour les maintenir dans un intervalle de sécurité. Inversement, l'apparition d'une défaillance sur ce système provoque une variation (généralement non souhaitée) dans les variables continues du système. Cette défaillance se répercute donc directement sur l'évolution dynamique du processus physique.

Dans de tels systèmes, le comportement du système résulte donc de l'interaction entre la composante dynamique et déterministe de l'évolution des grandeurs physiques, la composante discrète et la composante stochastique associée au processus de défaillance.

L'évaluation des paramètres de la sûreté de fonctionnement nécessite de prendre en compte ce couplage entre processus déterministe (continu et/ou discret) et processus stochastique. C'est le domaine connu sous le nom de *fiabilité dynamique*.

[Labeau, 2002] en donne une définition : « c'est la partie des analyses probabilistes de sûreté qui étudie de manière intégrée le comportement des systèmes affectés par une évolution dynamique sous-jacente ».

Initialement, la fiabilité dynamique a été qualifiée de *dynamique probabiliste* [Labeau, 2002]. La formulation proposée est intéressante dans le contexte de nos travaux. Le comportement dynamique d'un système peut être vu comme une succession de sections d'évolutions déterministes connectées entre elles par des événements aléatoires régis par des lois probabilistes. Un tel comportement est décrit par un *processus stochastique déterministe par partie*.

III.3 Problématique de sûreté de fonctionnement et fiabilité dynamique

Un des problèmes principaux rencontré lors d'une étude de sûreté de fonctionnement des systèmes embarqués (systèmes dynamiques hybrides par définition) est la prise en compte de manière efficace et réaliste des dépendances ; plus exactement, des interactions dynamiques existant entre les paramètres physiques (pression, température, débit, niveau,…) particulier au

processus industriel développé et le comportement nomina1 ou dysfonctionnel des composants du système, qui constituent le milieu matériel du processus.

Dans de tels systèmes, la séquence de fonctionnement normal ou dangereux résulte d'une occurrence commune de deux types d'événements. Les premiers sont liés à l'évolution déterministe des paramètres physiques susmentionnés, alors que les deuxièmes sont dus aux sollicitations et aux défaillances des composants du système et sont donc de nature aléatoire.

Le problème que nous venons d'évoquer limite donc la validité des méthodes principales citées dans la première partie de ce chapitre. Les méthodes combinatoires (arbres de défaillance, arbres d'événements, diagrammes de fiabilité) permettent uniquement d'identifier et d'évaluer les combinaisons des événements menant à l'occurrence d'un autre événement, indésirable ou pas. De telles combinaisons, ne tiennent pas compte de l'ordre de l'occurrence des événements qui les composent. Elles éliminent toute notion de dépendance entre ces événements. En outre, ces méthodes combinatoires présupposent une occurrence simultanée de tous les événements. Ceci exclut toute possibilité de prendre en compte la dépendance et les délais entre événements.

Le deuxième ensemble de méthodes comprend les approches markoviennes [Kaaniche, 1999]. Ces méthodes sont totalement ou partiellement exemptes du problème précédent mais sont limitées par d'autres contraintes. En effet, il est difficile de modéliser correctement des délais déterministes et des processus se déroulant en parallèle.

IV Méthodes et outils pour la fiabilité dynamique

Nous présentons brièvement les différents travaux relatifs à la modélisation en vue de l'analyse de sûreté de fonctionnement et les méthodes relatives à la fiabilité dynamique des systèmes complexes. Une présentation détaillée de tous ces travaux peut être consultée dans [Khalfaoui, 2003] et [Mejoudj, 2006].

IV.1 Modélisation des systèmes pour des études de SDF

IV.1.1 Langage de modélisation AltaRica

Le langage AltaRica est issu d'une collaboration en 1996, entre le LaBRI (Laboratoire Bordelais de Recherche en Informatique) et plusieurs industriels. L'objectif est d'établir divers ponts entre la sûreté de fonctionnement et les méthodes formelles, les analyses quantitatives des dysfonctionnements et les analyses qualitatives des comportements fonctionnels, afin de fournir aux ingénieurs concepteurs de systèmes complexes, un atelier outillé.

Une description du système dans ce langage, basé sur les automates à contraintes, permet de simuler un modèle et de générer un arbre de défaillance ou de vérifier qu'il satisfait des propriétés logiques. L'objectif depuis 1996, est de développer un outil générique permettant l'analyse des systèmes incorporant :

- un langage de spécification de haut niveau AltaRica [Griffault et al 98]

- un model-checker MecV [Arnold et al 94]
- des outils de sûreté de fonctionnement [Aralia 96]

Dans l'atelier AltaRica, un modèle AltaRica est transformé en des formalismes de plus bas niveau, comme les systèmes de transitions, et rend ainsi possible la vérification grâce au model checking.

L'atelier AltaRica est une réponse à certains aspects du contrôle de processus. Une de ses limites est l'impossibilité d'exprimer des modèles nécessitant des opérateurs de modélisation temps réel. Les travaux de [Pagetti 04] avaient pour but d'étendre le modèle AltaRica à ce genre de systèmes et ils ont donné naissance à Timed AltaRica, Hybrid AltaRica.

IV.1.2 Langage AADL

AADL, Architecture Analysis and Design Language, est un langage de description d'architecture système destiné aux systèmes embarqués (contextes : automobile, aéronautique et spatial notamment). Sa standardisation est en cours sous l'autorité de la «International Society of Automotive Engineers» (SAE). Une première version stable, AADL 1.0, a été publiée en novembre 2004.

Le principe est de décrire l'architecture pour mieux maîtriser sa complexité, et pouvoir vérifier quelques propriétés de ce système comme l'ordonnançabilité, la bonne transmission des messages ou le bon dimensionnement du matériel.

AADL décrit plusieurs composants qui modélisent une partie du système. Certains composants sont matériels (bus, processor, memory ...), d'autres logiciels (process, thread, subprogram, ...). Chaque composant peut avoir des propriétés, et peut contenir des sous-composants (un processus - process - pouvant par exemple contenir plusieurs fils d'exécution - threads). On peut également décrire plusieurs machines et les relier entre elles pour simuler des connexions réseaux.

Des travaux se basant sur AADL dans le domaine de la sûreté de fonctionnement ont été menés par [Rugina & al, 2006] et ont pour objectif de proposer une méthode structurée pour guider l'élaboration du modèle AADL (Architecture Analysis and Design Language) [SAE-AS5506] de sûreté de fonctionnement et proposer des règles de transformation de modèle pour générer des RdPSG (Réseau de Petri Stochastique Généralisé) à partir de modèles AADL de sûreté de fonctionnement. L'ensemble des règles de transformation de modèle est conçu pour être mis en œuvre dans un outil de transformation de modèle, de façon transparente à l'utilisateur.

L'approche vise à assister l'utilisateur dans la construction structurée du modèle AADL de sûreté de fonctionnement (le modèle d'architecture + les informations liées à la sûreté de fonctionnement). Le modèle AADL de sûreté de fonctionnement est transformé en un RdPSG qui peut être traité par des outils existants ([Ciardo & al, 1993], [Benardi & al, 2001], [Ciardo & al, 1993] et [Hirel & al, 2000]). Pour faciliter l'évolution du modèle, le modèle AADL de sûreté de fonctionnement est construit de manière itérative. Les comportements des composants architecturaux sont d'abord modélisés en présence de leurs propres fautes et événements de

réparation. Ensuite, les dépendances entre composants sont modélisées dans les itérations suivantes.

La transformation du modèle AADL en RdPSG est conçue pour être transparente à l'utilisateur. Par conséquent, elle est basée sur des règles systématiques et rigoureuses, destinées à une mise en œuvre automatique par des outils. La transformation de modèle peut être effectuée de manière itérative à chaque fois que le modèle AADL de sûreté de fonctionnement est enrichi. Ainsi, la sémantique du RdPSG peut être validée progressivement. Par conséquent, le modèle AADL correspondant peut être aussi validé progressivement et corrigé si nécessaire.

L'intérêt que suscite le langage AADL ces dernières années a motivé les auteurs des ces travaux pour exploiter le langage AADL, qui fournit une notation textuelle et graphique standardisée pour décrire des architectures matérielles et logicielles, comme langage de départ. Ceci dit la question qui se pose est l'efficacité du modèle RdPSG pour l'analyse de sûreté des systèmes complexes et l'éternel problème de l'explosion combinatoire.

IV.2 Arbre de défaillance haut niveau

L'approche par arbres de défaillance (*AdD*) présentée précédemment a l'avantage d'être facilement compréhensible par des personnes autres que le créateur même de l'arbre. Néanmoins, l'arbre classique possède un certain nombre de limites trop fortes dans le cadre de la fiabilité dynamique qui nous intéresse. Les arbres de défaillance, qui sont la représentation d'une formule booléenne sous forme de graphe ne tiennent donc pas compte de l'ordre d'apparition des événements et des dépendances fonctionnelles, caractéristiques pourtant très importantes dans les systèmes physiques. Afin de combler ces lacunes, de nouvelles approches intégrant l'aspect dynamique des systèmes ont été étudiées. Celle proposée par Dugan [Manian & al, 1998] de l'Université de Virginie (Etats-Unis) est basée sur la décomposition de l'arbre initial en une partie statique et une partie dynamique. La partie statique de l'arbre est alors traitée de manière classique à l'aide des diagrammes de décision binaire (*BDD*) [Bryant, 1986], la partie dynamique étant, quant à elle, traitée à l'aide d'une autre méthode appelée méthode de Markov [Howard, 1960]. L'inconvénient de cette approche est que toute possibilité d'analyse qualitative est perdue puisque la notion de coupe dynamique est absente. D'autres outils, appelés "générateurs de séquences" permettent de trouver des coupes munies d'un ordre total. Là encore un problème surgit ; certaines séquences d'événements ne mènent pas à l'événement redouté de manière permanente. En effet, certaines configurations d'un système peuvent autoriser l'occurrence de cet événement de manière temporaire, sans causer pour autant la perte définitive du système.

Cepin et Mavko [Cepin & Mavko, 2002] ont proposé une approche d'arbre dynamique basée sur le principe de la "composition" de sous arbres (figure I.4) correspondant à chacune des configurations du système. L'évolution de manière discrète dans le temps des événements est encodée dans une matrice. Cette matrice permet, lors du traitement de l'arbre, d'activer ou de désactiver certaines branches de l'arbre et donc de se placer dans une configuration précise à un instant t donné.

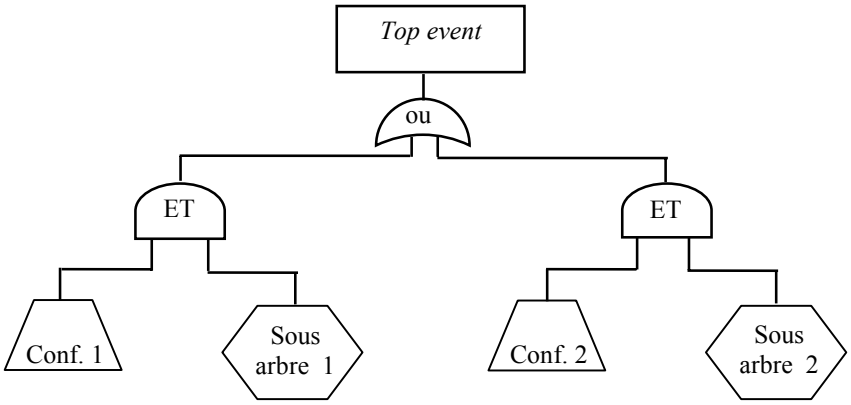


Figure I.4. Arbre de défaillance dynamique

D'autres travaux, notamment ceux de [Barrangan Santiago & al, 2006] ont porté sur la vérification des systèmes logiques. L'étape de formalisation des propriétés à vérifier est basée sur l'utilisation des arbres de défaillance classiques dans lesquels on a introduit l'information sur l'ordre entre les événements et les informations temporelles grâce a de nouvelles portes (figure I.5, I.6, I.7) qui modélisent les contraintes temporelles. Le comportement de ces nouvelles portes est alors formalisé par des automates. La vérification est alors effectuée avec l'outil UPPAAL [Larsen & al, 1997].

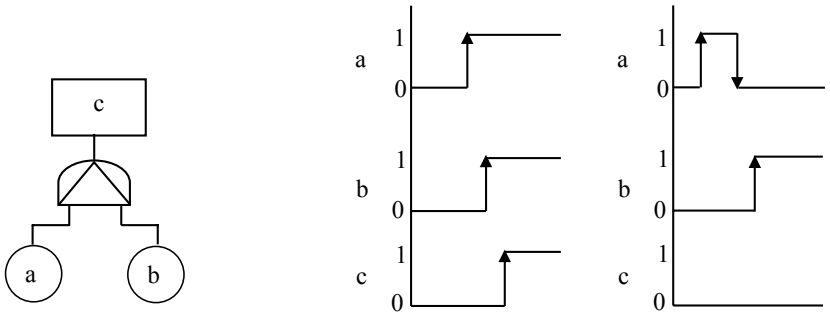


Figure I.5. Porte ET à priorité.

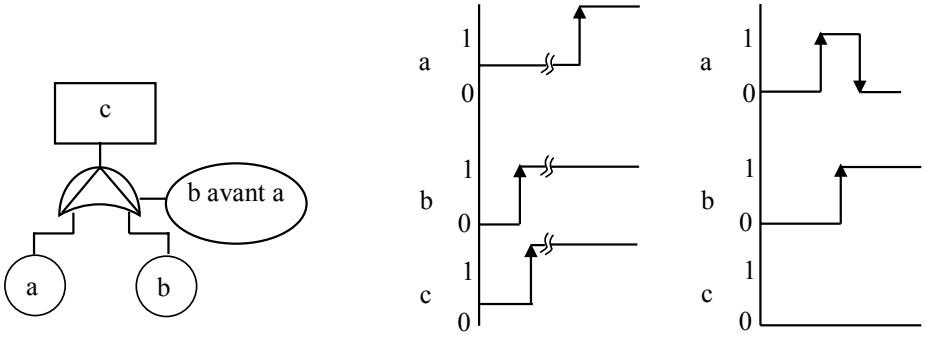


Figure I.6. Porte OU exclusive avec condition sur une entrée.

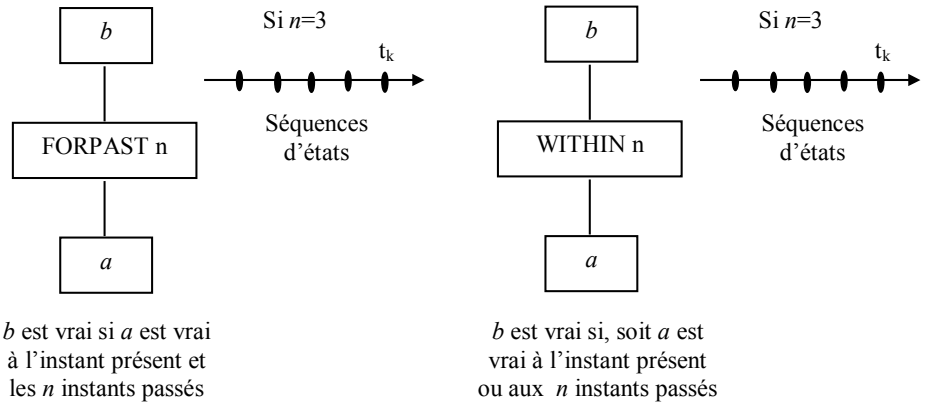


Figure I.7. Portes temporelles

IV.3 Méthode des graphes de flux dynamiques

La méthode des graphes de flux dynamiques (Dynamic Flowgraph Methodology DFM) [Garret, 1993] est une méthode de modélisation et d'analyse pour l'étude des risques des systèmes embarqués. L'approche consiste à identifier les scénarios redoutés et à élaborer une stratégie de test des systèmes. Elle permet de prendre en compte l'aspect dynamique des systèmes et l'interaction existante entre la partie logicielle et la partie matérielle du système.

La méthode DFM est composée de deux étapes :

1. L'étape de modélisation consiste à construire un modèle, sous forme d'un réseau de nœuds, exprimant le comportement logique et dynamique du système. Ces nœuds sont reliés entre eux par des connecteurs pour exprimer les relations de causalité et les relations temporelles entre les variables physiques de la partie opérative et les paramètres de commande du système de contrôle.
2. L'étape d'analyse a pour but de déterminer comment le système atteint un état donné (normal ou indésirable). L'analyse se fait en inversant les relations de causalité à partir du modèle DFM pour construire les Arbres de Défaillance temporisés.

Cette approche permet donc de déterminer les séquences menant à un événement redouté. La modélisation prend en compte explicitement les interactions entre la partie opérative et la partie commande. Un arbre de défaillance est alors construit pour un état redouté donné. Dans cette approche l'unicité du modèle et la construction automatique des Arbres de défaillance (AdD) donne une grande flexibilité aux concepteurs. Il suffit de faire les modifications sur le modèle et de générer automatiquement les AdD pour plusieurs états redoutés. Un autre avantage de cette méthode est qu'elle est basée sur la construction d'AdD (temporisés) bien connus dans la communauté des fiabilistes. L'inconvénient de cette méthode est le problème récurent de l'explosion combinatoire.

IV.4 Boolean logic Driven Markov Process (BDMP)

Les BDMP [Bouissou & Bon, 2003], [Bouissou, 2005] est un nouveau formalisme qui permet de modéliser et de calculer la fiabilité et la disponibilité de systèmes réparables complexes ayant des capacités de reconfiguration. Il s'agit d'un modèle markovien optimisé de manière à réduire considérablement la combinatoire du problème.

La structure globale d'un BDMP est donnée par une fonction logique de type arbre de défaillances. Un BDMP est constitué des éléments suivants :

- un arbre de défaillances (multi-tops) cohérent F,
- un événement top principal r,
- un ensemble de "gâchettes" T,
- un ensemble de "processus de Markov pilotés" Pi associés aux événements de base de l'arbre F,
- la définition de deux catégories d'états (marche et panne) pour les processus Pi.

Le principal événement top (r) du BDMP est censé représenter l'ensemble des états de panne du processus markovien global.

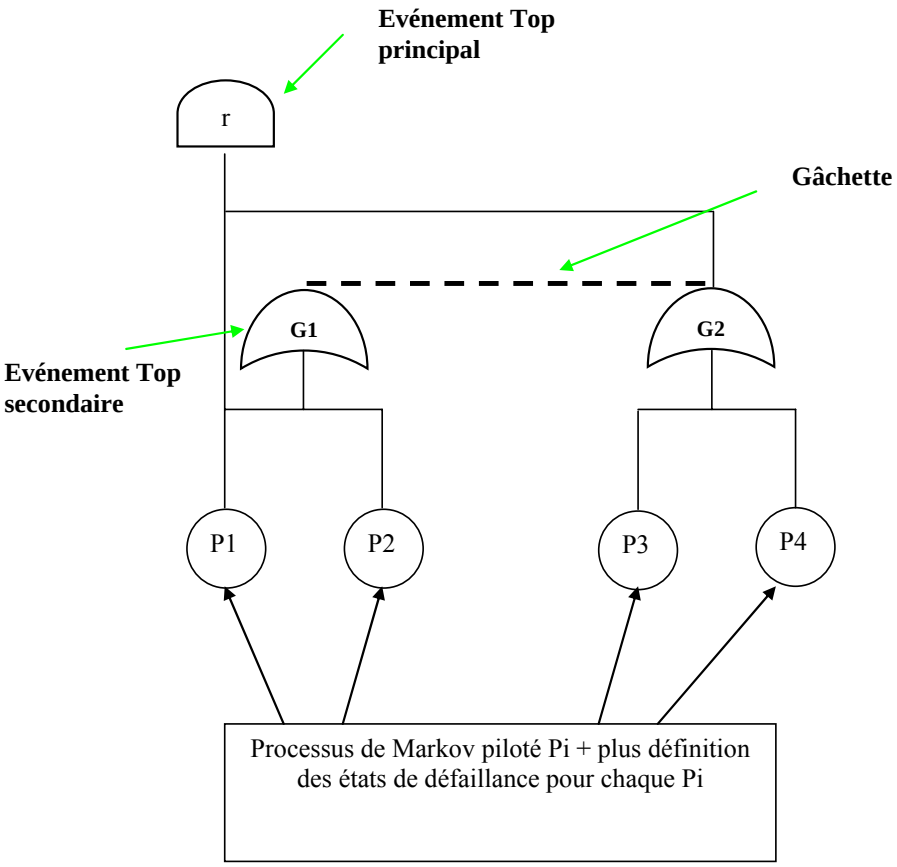


Figure I.8. Exemple de PDMP

Un BDMP sans gâchette équivaut à un arbre de défaillances classique. Dans un BDMP avec gâchette, les défaillances des composants ne sont pas toutes possibles dans l'état initial : seules celles des événements sollicités le sont.

Dans un BDMP, les portes sans parent (telles que G1 et r dans l'exemple de la figure I.8) sont sollicitées par défaut. Ces sollicitations se propagent de "père" en "fils" tout au long des branches du BDMP jusqu'à ce qu'elles rencontrent l'arrivée d'une gâchette. La présence d'une telle arrivée conditionne le passage du signal de sollicitation ; ainsi la porte cible transmet la sollicitation à ses descendants seulement si l'événement qui est à l'origine de la gâchette est VRAI 2. Si c'est le cas, la sollicitation est ensuite transmise aux « fils » suivant le même principe.

L'utilisation des gâchettes permet de modéliser simplement toutes sortes de dépendances entre les composants d'un système, en permettant de préciser dans quels contextes les défaillances à la sollicitation ou en fonctionnement des composants doivent être envisagées.

L'explosion combinatoire des traitements à effectuer pour quantifier un modèle BDMP est considérablement limitée par l'emploi de la notion d'événement pertinent. En effet, dès qu'un des fils d'une porte OU est à VRAI, les défaillances des autres fils (et descendants) ne sont plus "pertinentes" (à condition qu'elles ne puissent pas agir ailleurs dans l'arbre) et sont inhibées. On évite ainsi de nombreuses séquences contenant des événements "non pertinents" puisque agissant sur des parties de système déjà perdues. Par exemple, le filtrage des événements pertinents permet d'inhiber des modes de défaillance supplémentaires d'un composant déjà défaillant, ce qui est souvent plus réaliste que de continuer à les autoriser (cas des modes de défaillance mutuellement exclusifs, d'un composant qui ne peut défaillir que s'il est alimenté...). Toutefois, on a la possibilité de forcer localement le déclenchement de certaines défaillances au niveau de chaque porte ou feuille.

IV.5 Motifs formels d'architectures de systèmes pour la SDF

L'objectif des travaux de thèse de [Kehren, 2005], initiés par AIRBUS France, est de proposer des méthodes et outils d'aide à la modélisation et à l'évaluation de l'architecture de sûreté de fonctionnement des systèmes embarqués complexes. L'approche est basée sur des modèles généraux d'architectures de systèmes (*safety patterns*) correspondant à des éléments de sûreté (redondances, duplication …). « Un *pattern* est une solution récurrente qui décrit et résout un problème général dans un contexte particulier ».

L'idée de départ à la base de l'approche compositionnelle, est d'utiliser ou de réutiliser des composants génériques, abstractions d'architectures de sûreté, dont les propriétés ont été préalablement vérifiées. La réutilisation des assemblages, rendus génériques et que l'on appelle motifs d'architecture de sûreté, permet de simplifier les différentes tâches de conception (ou prototypage) et d'analyse d'architectures de systèmes complexes.

Le but est, entre autres, d'appliquer l'approche *design patterns* du génie logiciel au domaine de la sûreté de fonctionnement en exploitant les solutions proposées par les *design patterns* des problèmes récurrents. Ils renseignent sur les conséquences et le contexte d'utilisation et proposent des noms génériques aux *patterns* permettant de définir un vocabulaire précis.

Les *safety patterns* renferment un ensemble d'hypothèses (sur leur environnement, leurs entrées, les modes de défaillances, …), une spécification correspondant à l'architecture de l'élément de sûreté que représente le *pattern* et enfin des exigences garanties qui vont dépendre à la fois des hypothèses et de l'architecture. Un ensemble d'attributs est défini dans le but de distinguer les différents *patterns* par un nom générique et une structure propre leur conférant certaines propriétés sous certaines hypothèses. Ces attributs permettent de capitaliser le savoir d'experts des différents domaines d'utilisation ainsi qu'une meilleure utilisation/choix des *patterns* lors de la phase amont de conception d'architectures. Pour faciliter cette conception, un catalogue de *patterns* d'architectures de sûreté a été réalisé.

Le langage AltaRica a été utilisé pour formaliser des systèmes en présence de pannes. Les *patterns* correspondent à des abstractions d'architectures concrètes et donc requièrent une modélisation plus déclarative, i.e. à l'aide de propriétés. Ces propriétés étant en général dynamiques, le choix s'est porté sur la Logique Temporelle Linéaire (LTL), pour les modéliser. La description des *patterns* est donc constituée d'une partie AltaRica et d'une partie LTL.

Pour caractériser le comportement d'un système défini de manière mixte (opérationnel/dénotationnel). La sémantique d'AltaRica est plongée dans les Structures de Kripke Etiquetée (SKE). En effet, les SKE définissent la sémantique de State-Event LTL. De plus, afin obtenir des structures manipulables informatiquement, les propriétés de SE-LTL ont été traduites en automates de Büchi. Le produit de ces deux structures permet d'obtenir un automate de Büchi [Vardi & Wolper, 1986], [Vardi, 1996] caractérisant le comportement satisfaisant les contraintes fixées à la fois par la partie AltaRica et par la partie SE-LTL. La syntaxe AltaRica a été étendue pour permettre de spécifier et contraindre le comportement des composants par des propriétés temporelles à la fois sur les états et sur les événements.

IV.6 Graphe de Markov agrégé pour la fiabilité dynamique

Dans le cadre des travaux effectués au laboratoire CRAN en collaboration avec GFI Consulting, [Schoenig, 2004] s'est intéressé à la fiabilité dynamique des systèmes hybrides. Les travaux avaient pour objectif de définir une méthodologie de conception des systèmes de contrôle-commande tout au long des différents cycles de vie. Une approche basée sur la construction d'un graphe de Markov agrégé a été proposée. L'originalité de son approche tient dans sa capacité de répondre à un problème de représentation et d'évaluation de la fiabilité des systèmes dynamiques hybrides. Les principales étapes consistent à découpler la dynamique du système et la dynamique du processus de défaillance grâce à la théorie des perturbations singulières, puis d'identifier et estimer les grandeurs du système influençant la dynamique des défaillances.

Le principe général de l'approche consiste à coupler les deux méthodes d'analyse classiques : la simulation et les graphes de Markov, afin de contourner les limites intrinsèques à ces méthodes (temps de simulation, explosion combinatoire). L'approche proposée consiste à réaliser une agrégation des états élémentaires d'un graphe de Markov. L'influence du système sur le processus de défaillance est intégrée dans le modèle par le biais de termes de pondération des

taux de défaillance. Ceux-ci permettent de prendre explicitement en compte la dynamique interne du système. Ces coefficients sont évalués par une simple simulation, permettant ainsi de traiter des systèmes complexes. Ce type d'approche, qualifié de « dynamique », permet de traiter des systèmes fortement dépendant du temps (par exemple, en cas d'existence de reconfigurations matérielles ou logicielles).

IV.7 Simulation de Monte Carlo

La simulation de Monte Carlo [Batut, 1986], [Desieno & Stine, 1964], [Dubi, 2000] est une technique utilisée pour estimer la probabilité de résultats en répétant un grand nombre de fois une expérience à l'aide de la simulation et en utilisant des nombres aléatoires. La simulation est une méthode qui a pour but d'imiter un système réel. La simulation de Monte Carlo est utilisée lorsque d'autres analyses sont mathématiquement trop complexes ou trop difficiles à reproduire.

Un des avantages de la simulation de Monte Carlo est sa faible sensibilité à la complexité et à la taille des systèmes. Cependant, dans le cadre de la sûreté de fonctionnement, le modèle simulé est régi par des événements très rares (les défaillances) et des événements très fréquents (fonctionnement normal du système), et ce, simultanément. La simulation est alors cadencée par de nombreuses occurrences d'événements fréquents qui ne reflètent pas le comportement du système en présence de défaillances. C'est le problème de simulation des événements rares. Un nombre important d'histoires est nécessaire pour voir apparaître un événement redouté, ce qui implique des temps de calcul importants. De nombreuses techniques d'accélération de la simulation permettent de réduire ces temps [Garnier, 1998]. Elles sont basées soit sur une diminution de la complexité du modèle, soit sur la réduction du nombre de scénarios à simuler, en favorisant l'apparition des événements rares. Toutefois, ces méthodes ne sont pas toujours faciles à mettre en œuvre, car selles impliquent des hypothèses assez fortes, et/ou ne fournissent pas forcément des estimateurs de qualité.

IV.8 Recherche de scénarios redoutés

Dans sa thèse, réalisée au LAAS en collaboration avec PSA, G. Moncelet [Moncelet, 1998] s'est intéressé à la sûreté de fonctionnement des systèmes mécatroniques d'une manière qualitative et quantitative. La démarche consiste à modéliser le système par un réseau de Petri haut niveau, plus exactement les réseaux de Petri colorés [Jensen, 1992]. A partir du modèle réseau de Petri, le graphe des marquages accessibles est généré. Tous les chemins menant vers un état redouté sont ensuite identifiés et caractérisés en termes d'enchaînements d'actions et de changement d'états des composants concernés du système. Les séquences d'événements redoutés sont ainsi déterminées. Une simulation de Monte Carlo permet ensuite d'estimer leurs probabilités d'occurrence.

La simulation du modèle global du système avec une simulation de Monte Carlo est confrontée au problème des événements rares. Le simulateur passe la majorité de son temps à simuler le fonctionnement normal du système qui n'apporte aucune information sur la sûreté de

fonctionnement. Deux voies ont été abordées pour accélérer la simulation. La première consiste à remplacer le joueur de réseau de Petri, très lent, par un code compilé en langage ML (langage fonctionnel supporté par DesignCPN). Une accélération sensible a été obtenue. Toutefois, cette solution reste très contraignante car elle oblige le concepteur à traduire le réseau de Petri en langage ML. La deuxième voie consiste à caractériser, au préalable, les scénarios redoutés de manière qualitative afin de ne simuler que les comportements susceptibles de conduire vers une situation dangereuse. Ceci a donc conduit à effectuer une analyse qualitative. Cette analyse qualitative est basée sur la génération du graphe d'occurrence à partir d'un modèle qualitatif. Cette étude a montré que ce graphe explose avec le nombre de défaillances et le nombre de niveaux discrets nécessaires pour prendre en compte la dynamique continue. La recherche de scénarios par exploration de ce graphe est d'autant plus difficile qu'il contient bon nombre d'informations sans intérêt car décrivant des comportements sans défaillances.

Dans la continuité des travaux de Gilles Moncelet, le travail de thèse de [Khalfaoui, 2003] se focalise sur l'analyse qualitative de la sécurité des systèmes mécatroniques en vue de l'obtention des scénarios redoutés. La connaissance de ces scénarios permet d'évaluer leurs probabilités d'occurrence et de valider les lois de reconfiguration pour orienter le choix des concepteurs quant aux différents types d'architectures possibles proposés pour le système.

Une méthode de recherche de scénarios redoutés a été développée au cours de cette thèse. Elle est basée sur la modélisation d'un système mécatronique à l'aide de Réseaux de *Petri Prédicats transitions différentiels* [Champagnat, 1998] afin de prendre en compte l'aspect hybride des systèmes mécatroniques. Cette modélisation hybride associant les RdP et les équations différentielles présente l'avantage de séparer clairement les aspects discrets et continus. Ceci permet une analyse logique (fondée sur la logique Linéaire) des causalités [Girard, 1987] résultant des changements d'états. Grâce à cette analyse, il est possible à partir d'un état redouté de remonter les chaînes de causalité et de mettre ainsi en évidence tous les scénarios possibles conduisant à une situation critique. Chaque scénario est donné sous la forme d'un ordre partiel entre les événements nécessaires à l'apparition de l'état redouté. L'originalité de l'approche est qu'elle n'implique pas une énumération brutale et globale de tous les états accessibles du système. Au contraire elle permet de se focaliser sur le voisinage de l'état redouté en faisant une énumération locale d'états partiels. Autrement dit, elle ne considère que les états des composants directement impliqués dans l'apparition de l'état redouté. Dans le travail de thèse de Khalfaoui les réseaux de *Petri prédicats transitions différentiels* ont été étendus pour prendre en compte l'aspect stochastique caractérisant les défaillances et les réparations des différents composants du système. Les réseaux de *Petri prédicats transitions différentiels stochastiques* (RdP PTDS) ont été donc introduits. Ils permettent d'une part de prendre en compte l'aspect hybride des systèmes mécatroniques ainsi que l'aspect stochastique, nécessaire lors des analyses de sûreté de fonctionnement.

Ce travail était une première étape et l'algorithme de construction des scénarios ne prenait en compte que la vue discrète du système. Or il s'avère, même sur des exemples assez simples, que de nombreux scénarios produits ne vérifient pas les contraintes provenant de la vue continue du

système et sont contradictoires avec la dynamique de la partie énergétique du système. Il semble plus efficace d'en éliminer, au moins un certain nombre, dès la construction des scénarios.

Suite à cette première version de l'algorithme, les travaux de thèse de [Medjoudj, 2006] ont permis de réduire considérablement le nombre de scénarios redoutés générés. Les travaux ont donné naissance à une deuxième version de l'algorithme en prenant en compte partiellement la partie continue. En effet, grâce à des abstractions temporelles de la dynamique continue du système de nombreux scénarios incohérents avec la dynamique continue ont été éliminés.

Son approche est également orientée vers la vérification de certaines propriétés des systèmes pilotés par calculateur. Ces propriétés peuvent être de type temporel (la durée maximale d'un scénario ou la durée entre deux commandes) ou de type accessibilité entre deux états.

Dans le cadre de cette thèse un premier outil (ESA_Petri net) a été développé. L'outil permet de générer automatiquement les scénarios redoutés à partir du modèle réseau de Petri du système et d'effectuer également la vérification des contraintes temporelles.

L'automatisation de la méthode de recherche de scénarios a permis de conforter les choix fait initialement sur le modèle RdP PTDS ainsi que le caractère local de la méthode. L'outil a été validé sur plusieurs cas d'études assez conséquent inspirés de l'industrie aéronautique.

Toutefois, même si la nouvelle version de l'algorithme réduit considérablement le nombre de scénarios générés (notamment ceux incohérents avec la dynamique continue), l'ensemble des scénarios généré ne se limite pas aux scénarios minimaux et certains scénarios ne mènent pas à l'état redouté.

Il faut souligner qu'il existe d'autres travaux qui se sont intéressés à la fiabilité dynamique. Le lecture intéressé pourra les consulter dans [Moncelet, 1998], [Khalfaoui , 2003] et [Medjoudj, 2006].

V Conclusion

Après une brève introduction aux systèmes embarqués, nous avons passé en revue dans ce chapitre les principales méthodes classiques de sûreté de fonctionnement mais aussi les principaux travaux relatifs à la fiabilité dynamique.

Le domaine de la fiabilité dynamique doit permettre de répondre à la problématique liée aux insuffisances des méthodes statiques limitées par un certains nombre d'hypothèses notamment l'absence de la notion d'ordre entre les événements ainsi que la notion de l'instant d'apparition des événements.

Les travaux de S. Khalfaoui et M. Medjoudj ont montré la bonne adéquation des réseaux de Petri à la modélisation de systèmes dynamiques hybrides et la nécessité d'une analyse qualitative qui permettrait de déterminer au préalable les scénarios redoutés avant de faire une évaluation quantitative de ces scénarios. Ceci permet de s'affranchir du problème des événements rares rencontrés lors de la simulation de Monte Carlo par exemple en faisant une simulation ciblée. L'autre aspect intéressant de leur approche est l'exploitation directe du

modèle réseau de Petri, ceci permet d'éviter le problème de l'explosion combinatoire dont souffrent certaines approches. L'introduction de la logique linéaire comme cadre formel pour l'analyse des scénarios redoutés permet de déterminer les relations de cause à effet dans le modèle et ainsi une meilleure caractérisation des scénarios redoutés.

Après ce tour d'horizon des méthodes et outils, de la fiabilité dynamique, nous constatons l'absence d'approches globales pour l'analyse de sûreté de fonctionnement. Il nous a paru nécessaire, d'aborder ce problème. Dans le chapitre, qui suivant nous allons donc exposer la problématique d'intégration de la sûreté de fonctionnement dans les processus de l'ingénierie système et présenter ensuite une approche d'intégration basée sur la norme *EIA-632*.

Chapitre 2 : Sûreté de fonctionnement et processus d'ingénierie système

I Introduction

Lors de la conception d'un système, le concepteur se doit de respecter les exigences exprimées par les différentes parties prenantes [Calvez, 1992]. Ces exigences sont soit fonctionnelles soit non fonctionnelles (les délais, les coûts, la sûreté de fonctionnement, les performances).

De nombreuses méthodes et outils ont été développés pour faire face à la complexité croissante des systèmes embarqués. Malgré cela, une méthodologie avec une démarche systémique est nécessaire pour passer du simple cahier des charges, qui exprime les besoins et exigences, à la réalisation du produit final.

Les études montrent que les principaux défauts de conception sont dus à [Lenoir, 2000]:

- des besoins mal spécifiés ou exigences mal formulées,
- une évolution des besoins/exigences dans le temps,
- une modification spontanée, parfois faite avec de bonnes intentions,
- la non accumulation de savoir-faire et manque de retour d'expérience,
- au pari technologique,
- une définition erronée d'interfaces,
- la pression de la concurrence,
- une extension d'exigences fonctionnelles.

Traditionnellement, dans la conception des systèmes, chaque fonction pouvait être étudiée et développée indépendamment des autres et l'implication de la sûreté de fonctionnement se résumait à la réutilisation de modèles génériques issus du retour d'expérience. Cette approche

traditionnelle ne permet pas de prendre en compte les risques liés à l'intégration de plusieurs technologies. Il est donc important de formuler les exigences de sûreté de fonctionnement non seulement localement « in the small » mais globalement (niveau système, « in the large »). Cela revient à formuler ces exigences au niveau du système complet et, ensuite, à les décliner à des niveaux plus bas (jusqu'aux simples composants).

Le « *cycle en V*» (figure II.1) est généralement utilisé dans la description du cycle de développement d'un système embarqué. Le cycle est composé de deux branches. La branche descendante, qui correspond à une démarche de raffinements successifs, décrit les phases de conception allant du général, qui démarre avec l'expression des besoins, au particulier. La branche ascendante détaille les phases d'intégration et de validation correspondant à chaque phase de conception.

La conception d'un système suivant le cycle en V permet de retarder le choix de la technologie de réalisation. Les efforts sont concentrés sur la spécification et la conception.

Les phases de spécification et de conception conduisent à définir des niveaux de description de plus en plus détaillés. Les phases d'intégration, de test et de vérification permettent d'évaluer la conformité de la réalisation pour chaque niveau de la conception.

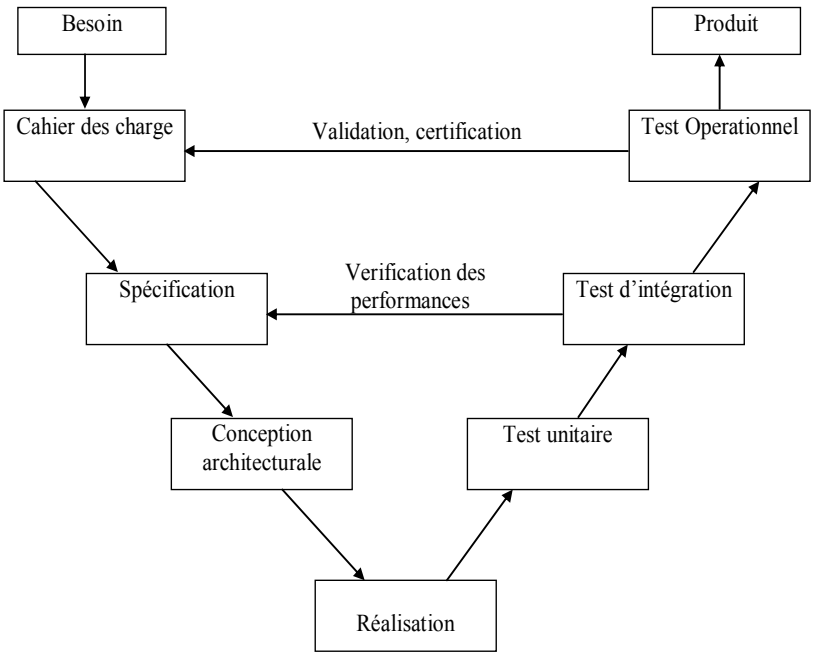


Figure II.1. Cycle de développement en V d'un système embarqué

Les systèmes actuels sont de plus en plus complexes car ils intègrent de plus en plus de fonctionnalités, de plus en plus de technologies variées et doivent être opérationnels sur des périodes de plus en plus longues. Ces systèmes demandent de plus en plus souvent de longues périodes de développement pendant lesquelles les besoins évoluent. Les coûts et les délais sont

de plus en plus sévères compte tenu d'un environnement où la compétitivité mondiale est de rigueur. Pour prendre en compte ces contraintes, un cadre a été défini. Il s'agit de l'Ingénierie Système (IS).

La particularité de l'IS est de considérer à la fois les aspects techniques et les aspects logistiques (financiers, stockage, transport). L'IS a été traditionnellement une discipline appliquée aux systèmes physiques ; depuis une dizaine d'années, une révolution tranquille s'opère dans l'ingénierie des systèmes complexes. Les domaines où ce changement a été prépondérant sont les télécommunications, les systèmes d'information qu'ils soient civils ou militaires (CCC), les systèmes de transport, les services financiers et autres applications allant du domaine médical au génie des procédés.

Plusieurs définitions existent pour l'Ingénierie Système. D'après la norme IEEE P1220 [STA-IEEE], l'Ingénierie Système est une approche interdisciplinaire et concerne la capacité de réussir la réalisation des systèmes.

L'Ingénierie Système est une démarche méthodologique générale qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante aux besoins d'un client tout en satisfaisant l'ensemble des parties prenantes. Elle a pour objectifs :

- la satisfaction des parties prenantes (en répondant à leurs demandes),
- la recherche de qualité technique,
- la tenue des délais,
- la tenue des coûts en respectant le budget alloué.

L'Ingénierie Système focalise sur les besoins du client et les fonctionnalités requises très tôt dans le cycle de développement. En considérant une vue complète et globale du problème, elle s'intéresse particulièrement aux aspects:

- Opérationnel
- Performance
- Test
- Fabrication
- Coût et planification des délais
- Formation et SAV (service après vente)
- Mise au rebut

Lors de la conception d'un système complexe incluant de nouvelles technologies, afin de mener dans les meilleures conditions le projet, les concepteurs utilisent des normes qui les aident à accomplir leur mission durant le développement. Parmi ces normes, on peut citer la norme *EIA-632* [STA-EIA].

II La norme EIA-632

II.1 Structure d'un système selon l'EIA-632

Cette norme complète les processus techniques de définition du système en couvrant la réalisation des produits jusqu'à leur mise en service (transfert vers l'utilisation). De plus, elle inclut les processus contractuels d'acquisition et de fourniture.

C'est l'une des normes les plus populaires et efficaces, et la plus utilisée dans le domaine de l'Ingénierie Système. Une telle norme a été déployée principalement dans des industries de l'aéronautique, de production et militaires. Selon les informations recueillies sur l'échec de projets industriels [Sahraoui & al, 2004], généralement la cause de l'échec du projet est une négligence des produits qui contribuent à la conception du système désiré. En effet, la plupart des parties prenantes se focalisent uniquement sur le développement des produits finaux (*End products*). La norme *EIA-632*, introduit le concept de produit capacitant (enabling product) : il s'agit des produits qui permettent d'obtenir un produit final et peuvent être utilisés par plusieurs produits finaux.

L'importance de développer le système en suivant la norme industrielle (*EIA-632*) est de décomposer le système entre produits finaux et produits capacitants (figure II.2) et d'explorer un ensemble de processus qui sera appliqué pour le développement des produits finaux.

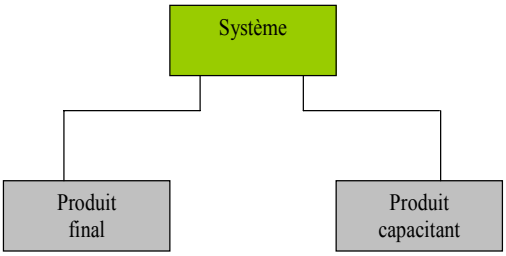


Figure II.2. Décomposition du système (EIA-632)

L'infrastructure pour le développement se fonde fortement sur ce que l'on appelle souvent les problèmes de la logistique, qui sont considérés comme des produits capacitants dans le cadre de l'*EIA-632*.

Ces produits capacitants seront utilisés pour exécuter l'ensemble des processus associés au développement, à la production, au déploiement, à la formation des opérateurs afin d'utiliser les produits finaux et au retrait des produits quand ils ne seront plus utilisables.

La figure II.3 présente sept types de produits capacitants : le développement du produit final, la production du produit final, le test du produit final, le déploiement du produit final, la formation des opérateurs pour savoir utiliser le produit final, le support du produit final, et la retraite du produit final (recyclage).

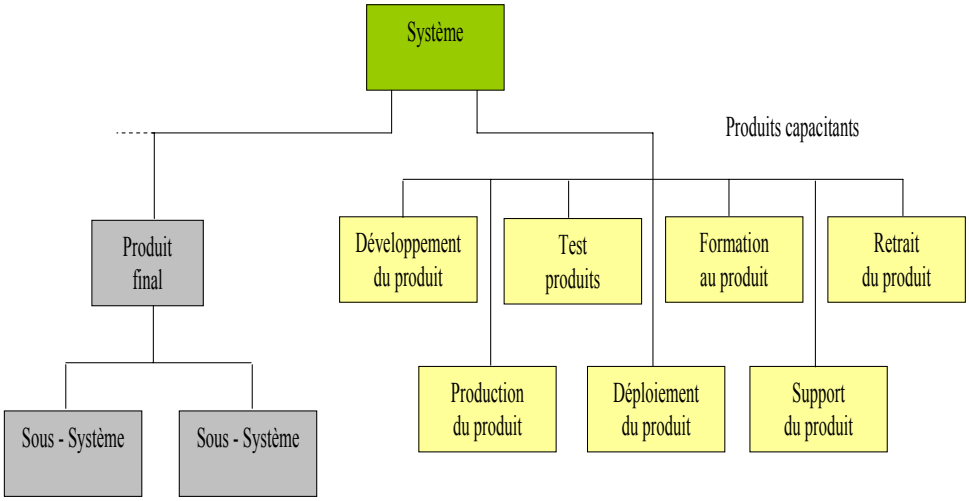


Figure II.3. Un module dans le cadre de l'IEA-632

Après la décomposition du système en produits finaux et produits capacitants, le concepteur obtient une hiérarchisation selon plusieurs niveaux (figure II.4).

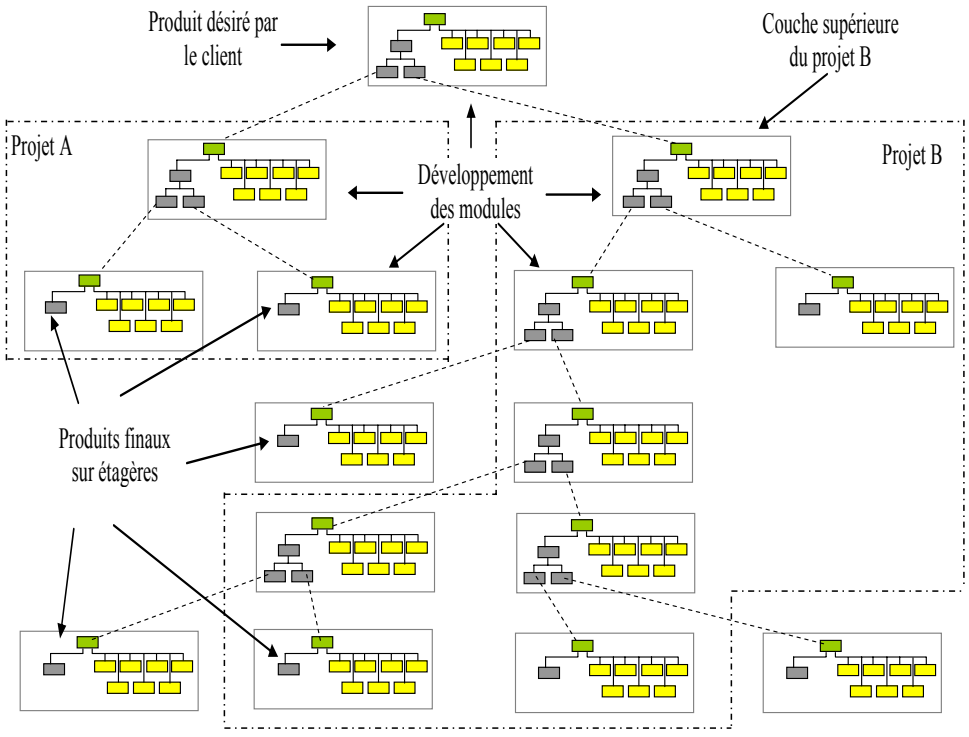


Figure II.4. Exemple de structure d'un système dans L'IEA-632

Le développement d'un système (produit désiré) est décomposé en une hiérarchie de sous-systèmes définis en tant que modules.

Le développement des modules de niveau inférieur est lancé dès que le module de niveau supérieur est complètement spécifié. A partir de là, le module est considéré comme un produit qui a des caractéristiques et des exigences identifiées et fait l'objet d'un développement de même type que le système désiré.

La décomposition se poursuit jusqu'à l'identification de trois catégories de produits finaux :

- Produits finaux sur étagère,
- Produits finaux pouvant être implémentés directement,
- Produits finaux pouvant être fournis par un sous-traitant.

II.2Processus de l'EIA-632

La norme *EIA-632* définit 13 processus (figure II.5) :

- les 3 processus de Management Technique
- les 2 processus d'Acquisition et de Fourniture
- les 2 processus de Conception
- les 2 processus de Réalisation
- les 4 processus d'Evaluation Technique

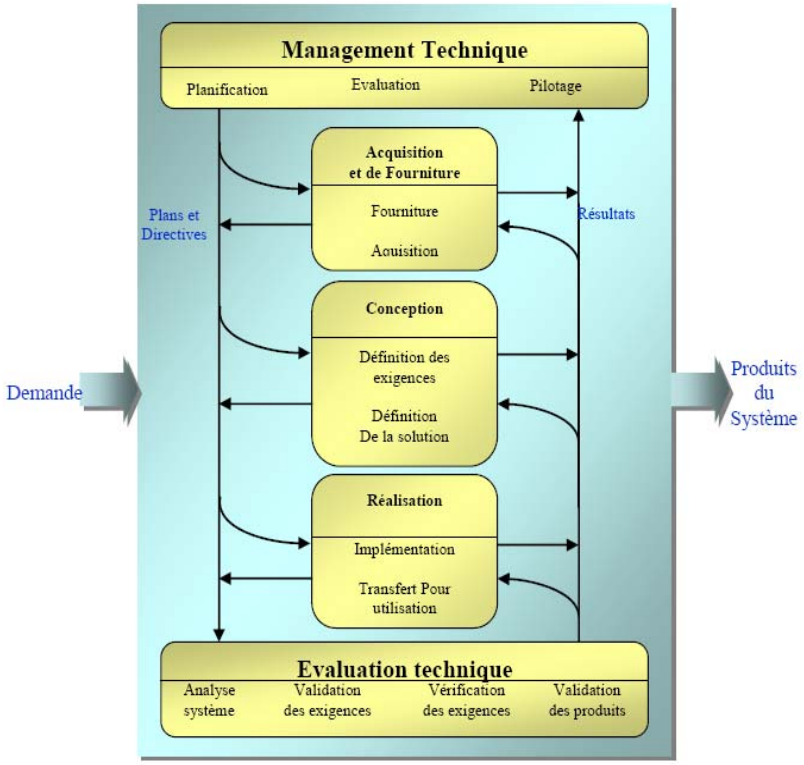


Figure II.5. Processus pour le développement des produits finaux.

Ces processus sont utilisés à chaque niveau de hiérarchisation des produits finaux, pour chaque module, pour les spécifier, les modéliser et, bien sûr, pour les développer.

Un processus est un ensemble d'activités interactives [AFIS] coordonnées pour transformer progressivement des éléments d'entrée en produits. Un processus normalisé décrit les types d'activités à réaliser et de résultats attendus de ces activités.

Ces processus doivent répondre à un certain nombre d'exigences. La norme en définit 33 [EIA, 1999] (figure II.6)

SUPPLY PROCESS REQUIREMENTS
1—Product Supply

ACQUISITION PROCESS REQUIREMENTS
2—Product Acquisition
3—Supplier Performance

PLANNING PROCESS REQUIREMENTS
4—Process Implementation Strategy
5—Technical Effort Definition
6—Schedule and Organization
7—Technical Plans
8—Work Directives

ASSESSMENT PROCESS REQUIREMENTS
9—Progress Against Plans and Schedules
10—Progress Against Requirements
11—Technical Reviews

CONTROL PROCESS REQUIREMENTS
12—Outcomes Management
13—Information Dissemination

REQUIREMENTS DEFINITION PROCESS REQUIREMENTS
14—Acquirer Requirements
15—Other Stakeholder Requirements
16—System Technical Requirements

SOLUTION DEFINITION PROCESS REQUIREMENTS
17—Logical Solution Representations
18—Physical Solution Representations
19—Specified Requirements

IMPLEMENTATION PROCESS REQUIREMENTS
20—Implementation

TRANSITION TO USE PROCESS REQUIREMENTS
21—Transition to Use

SYSTEMS ANALYSIS PROCESS REQUIREMENTS
22—Effectiveness Analysis
23—Tradeoff Analysis
24—Risk Analysis

REQUIREMENTS VALIDATION PROCESS REQUIREMENTS
25—Requirement Statements Validation
26—Acquirer Requirements Validation
27—Other Stakeholder Requirements Validation
28—System Technical Requirements Validation
29—Logical Solution Representations Validation

SYSTEM VERIFICATION PROCESS REQUIREMENTS
30—Design Solution Verification
31—End Product Verification
32—Enabling Product Readiness

END PRODUCTS VALIDATION PROCESS REQUIREMENTS
33—End Products Validation

Figure II.6. Exigence pour la conception de systèmes

III Prise en compte de la sûreté de fonctionnement

III.1 Problématique

Si les attributs de la *SdF* sont pris en compte à une échelle composant/équipement ou plus généralement par composant homogène (mécanique, électronique, etc ...) ce qui correspond à une approche ascendante (botom up)- il n'en demeure pas moins que beaucoup de pistes sont ouvertes dans la démarche système, type descendante (top down). Une des pistes à explorer est comment intégrer cet aspect dans un cadre Ingénierie Système.

Dans une approche système, la prise en compte de la sûreté de fonctionnement doit donc suivre toutes les étapes de la démarche Ingénierie Système. Si l'on considère, par exemple, une exigence de fiabilité définie sur le système de manière globale, sa formalisation et son analyse devront permettre de s'assurer que les solutions techniques choisies au fur et à mesure de l'avancement de la conception et de la réalisation permettent de prendre en compte correctement cette exigence au niveau des sous systèmes et de leur intégration. Le processus de vérification et de validation doit apporter les techniques et les solutions pour garantir le degré de fiabilité spécifiée. Les différentes méthodes et outils de l'analyse de la sûreté de fonctionnement sont ainsi déterminants dans le choix des solutions techniques à envisager. Si l'on considère, par exemple, la conception d'un système avec une exigence de fiabilité donnée, en considérant les défaillances possibles, une étude sur les scénarios redoutés devra permettre d'orienter la recherche des solutions techniques (reconfiguration, redondance, réduction des taux de défaillances, etc…) permettant de respecter cette exigence tout au long de la démarche Ingénierie Système, en tenant compte des autres exigences.

III.2 Approche d'intégration

La prise en compte de la sûreté de fonctionnement concerne tous les processus de l'Ingénierie Système. Mais, dans cette première approche d'intégration de la sûreté de fonctionnement, nous nous sommes focalisés sur :

- les processus de conception.
- les processus d'évaluation technique

Les exigences de sûreté doivent être prises en compte dans le processus de définition des exigences qui permet de les formuler, les définir, les formaliser, les analyser et d'en établir un modèle de traçabilité afin de s'assurer de leur prise en compte tout au long du cycle de vie du système. Ces exigences de sûreté influent sur les exigences de l'acquéreur (*Acquirer requirements*), des autres parties prenantes (*Other stakeholder requirements*), techniques (*System technical requirements*) et sur la solution logique (*Logical solution representations*). Dans notre approche nous n'avons pas considéré la solution physique (figure II.6).

Les processus de validation et de vérification au sens de cette norme sont assez explicites et définissent 8 types d'exigences allant de la validation de la déclaration/description de l'exigence (*requirement statements validation*) jusqu'à la vérification pendant l'exploitation (*enabled product readiness and transition to use*). Là aussi, nous n'avons pas pu prendre en compte tous ces types d'exigences. Les plus utiles à notre démarche concernent la validation des déclarations des exigences (*requirements statements validation*), la validation des exigences des autres parties prenantes (*Other stakeholder requirements validation)*, la validation des exigences techniques du système (*System technical requirements validation*), la validation de la représentation de la solution logique (*Logical solution representations validation*) et la vérification de la solution de conception (d*esign solution verification*).

Notre approche d'intégration de la sûreté de fonctionnement dans le processus d'Ingénierie Système consiste à traduire les exigences de la norme *EIA-632* en exigences exprimées en termes

de sûreté de fonctionnement et à les inclure dans le processus de développement et de conception.

IV Traduction des exigences de l'EIA-632 en exigences de SDF

Nous allons maintenant aborder les processus de L'IEA-632, avec un point de sûreté de fonctionnement.

IV.1 Les processus de conception

IV.1.1 Le processus de définition des exigences

Pour la définition des exigences fonctionnelles et non fonctionnelles lorsque cette distinction n'est pas faite au niveau du processus de l'élicitation des exigences, c'est à l'analyste de catégoriser les exigences. Le processus de définition des exigences comprend les exigences d'acquéreur (E14), les exigences des parties prenantes (E15), et les exigences techniques du système (E16).

Les entrées du processus de définition des exigences sont de trois types :

(1) exigences venant de contrats, d'autres documents, et individus ou groupes pour qui le système représente un enjeu.

(2) exigences sous forme de résultats d'autres processus comme les plans et décisions techniques des revues techniques,

(3) changements demandés ou approuvés des exigences du premier type.

Remarque :

Les exigences définies par ce processus viennent des parties prenantes impliquées dans le système. L'acquéreur du système est considéré comme partie prenante principale.

Le processus de définition des exigences est employé pour transformer des exigences des parties prenantes en un ensemble d'exigences (spécifications) techniques du système. Ces exigences sont énoncées en termes techniques acceptables et représentent une description raisonnablement complète du problème qui doit être résolu pour fournir un ensemble de produits finaux et de produits capacitants qui satisfont les besoins de l'acquéreur et des autres parties prenantes.

Le processus de définition des exigences est refait, selon les besoins, toutes les fois qu'un changement dans les exigences affecte la conception des produits. De tels changements ont pu être provoqués par des limitations de technologie, des programmes de projet des anomalies de coût ou de nouvelles conditions.

Le processus de définition des exigences, comprend 3 types d'exigences que nous allons présenter.

E14 - Exigences d'acquéreur :

Le concepteur doit définir un ensemble validé d'exigences d'acquéreur pour le système ou une partie du système. Pour cela il faut :

- Identifier, collecter, hiérarchiser les exigences.
- Assurer que les exigences répondent/prennent en compte tous les besoins de l'acquéreur (cela correspond à l'exigence E26 de l'*EIA-632*: validation des exigences acquéreur).

Dans le cadre de la sûreté de fonctionnement, les exigences d'acquéreur se traduisent par des contraintes sur le système. Il faut donc identifier et collecter toutes les contraintes imposées par l'acquéreur en vue d'obtenir un système sûr de fonctionnement. La hiérarchisation consiste à pondérer les exigences selon leur criticité.

Si on considère l'exemple d'un avion, l'immobilisation du système, si jamais un événement redouté survient, peut être considérée comme une exigence d'acquéreur. La hiérarchisation consiste, par exemple, à associer une priorité (selon la criticité) de 10 pour l'exigence de fiabilité du système d'atterrissage et de 4 pour le système de climatisation de l'avion.

Remarque : toutes ces exigences sont indépendantes du contexte.

E15 - Exigences des parties prenantes :

Le concepteur doit définir un ensemble validé d'exigences des parties prenantes du système ou d'une partie du système. Pour cela il faut :

- Identifier, collecter, hiérarchiser les exigences.
- Assurer que les exigences répondent aux besoins des autres parties prenantes (voir E27 : validation des exigences parties prenantes).
- Identifier collecter les autres exigences (parties prenantes) qui peuvent avoir des contraintes sur les produits capacitants.

La même démarche que celle liée à l'acquéreur (E14) s'applique sur les parties prenantes.

Remarque :

Les exigences (E15) définies à cette étape combinées avec les exigences d'acquéreur (E14) seront utilisées pour définir les exigences techniques du système (E16) et les exigences des produits capacitants.

E16 – Exigences techniques :

Le concepteur doit définir un ensemble validé d'exigences techniques de système. Pour cela il faut :

- Établir les règles de transformation, les priorités, les entrées, les sorties, les états, les modes et les configurations requises appropriés à chaque produit du système.
- Définir les conditions opérationnelles du système.

- Définir l'exigence de performance (avec quelle performance chaque fonctionnalité doit être accomplie), y compris l'identification des paramètres de performance critiques.
- S'assurer que l'ensemble des exigences techniques du système est conforme à l'exigence E28 (validation des exigences techniques du système).

Les exigences techniques du système doivent être sans ambiguïté, conformes, réalisables, vérifiables, et nécessaires et suffisantes pour la conception du système.

Dans le cadre de la sûreté de fonctionnement les exigences techniques du système se traduisent en termes de Performances du système. Cela revient à définir les données techniques et les attributs de sûreté de fonctionnement ; MTBF, MTBR, taux de défaillance, etc…

Remarque :

Les exigences définies à cette étape sont dépendantes du contexte. En effet, si on s'intéresse par exemple aux attributs de sûreté de fonctionnement, il est clair que l'exigence sur un attribut diffère d'un contexte à un autre.

IV.1.2 Le processus de définition de la solution logique

Cette étape consiste à faire une modélisation du système qui dépendra forcément de la culture de l'entreprise. On distingue les modèles de conception formels qui facilitent la validation par simulation ou autres types de modèles de validation semi-formelle sur lesquels on peut bâtir des passerelles du semi-formel vers le formel (Statecharts vers VDM). La démarche est illustrée par l'interaction des processus comme le montre la figure II.7.

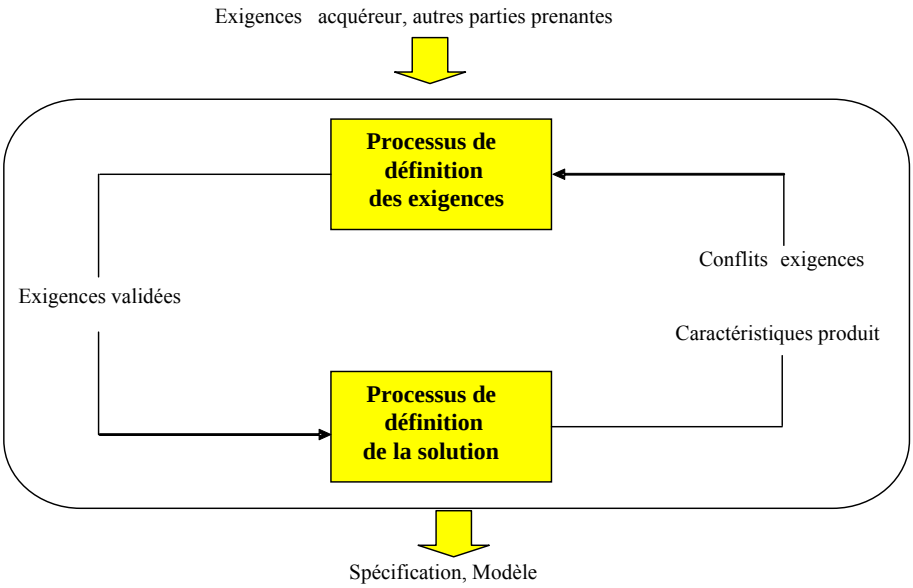


Figure II.7. Processus de conception du système.

E17 – Représentation de la solution logique :

Le concepteur doit définir un ou plusieurs ensembles validés de représentations de la solution logique conformes aux exigences techniques du système. Pour cela il faut :

- Choisir et implémenter une ou plusieurs approches appropriées pour fournir une définition abstraite de la solution répondant aux exigences techniques du système.

Remarque

L'analyse fonctionnelle, l'analyse orientée objet, l'analyse structurée sont des approches possibles pour développer les représentations de la solution logique en termes, par exemple, de fonctionnalités, de représentations comportementales, d'états et transitions entre modes, de chronologies, de flux de commande et flux de données, de modèles d'information, de services et attributs des objets, de diagrammes de contexte, de structures de données et de modes de défaillance fonctionnels et leurs effets.

- Assigner les exigences techniques de système (particulièrement les exigences de performance et les contraintes provenant des exigences techniques) aux éléments des représentations de la solution, par exemple, sous fonctions, groupes de sous fonctions, objets, structures de données.
- Identifier et définir les déclarations d'exigences techniques dérivées résultant des tâches. S'assurer que les exigences techniques dérivées sont énoncées en conformité avec l'exigence E25 (Validation des déclarations des exigences).
- S'assurer que chaque ensemble de représentations de la solution logique est conforme à l'exigence E29 (validation de la représentation de la solution logique).

Le modèle formel dépendra forcement de la politique de l'entreprise. Dans notre cas, le modèle formel sera les réseaux de Petri.

IV.2 Les processus d'évaluation technique

Il s'agit des processus :

- d'analyse du système avec comme sous processus le processus d'analyse des risques,
- de validation des exigences,
- de vérification du système,

Les interactions entre les processus de validation du produit final sont données dans la figure II.8. Ils sont utilisés pour :

(1) fournir une base rigoureuse pour la prise de décision technique et évaluer les solutions physiques alternatives ;

(2) déterminer la satisfaction des exigences techniques;

(3) la gestion des risques;

(4) s'assurer que des décisions sont prises seulement après évaluation du coût, et des effets des risques sur le système.

IV.2.1 Le processus d'analyse du système

Nous n'avons uniquement pris en compte l'exigence d'analyse de risque.

E24 – analyse de risque :

Le concepteur doit réaliser une analyse de risque pour développer des stratégies de gestion des risques et de prise de décision. Pour cela, il faut :

- Identifier les risques techniques et les risques sur le projet en estimant les probabilités d'une conséquence indésirable et son effet sur la solution physique.
- Caractériser les risques par leurs causes, leurs possibles effets ou leurs conséquences, leurs probabilités d'occurrence et les solutions pour gérer le risque.
- Donner la priorité aux risques qui sont les plus probables qui ont le plus d'effets néfastes sur le système.
- Évaluer les manières d'éviter le risque et déterminer le coût, le programme.
- Définir et mettre en application un plan ou une approche pour éviter chaque risque significatif.

Plusieurs techniques peuvent être utilisées pour une analyse des risques, nous pouvons citer les arbres de défaillances ou l'analyse préliminaire des risques. Cette étape est très importante, c'est elle qui permet de répertorier tous les risques liés au système et son utilisation. C'est l'analyse de ces risques et leur évaluation qui permettra ensuite de valider le système ou non.

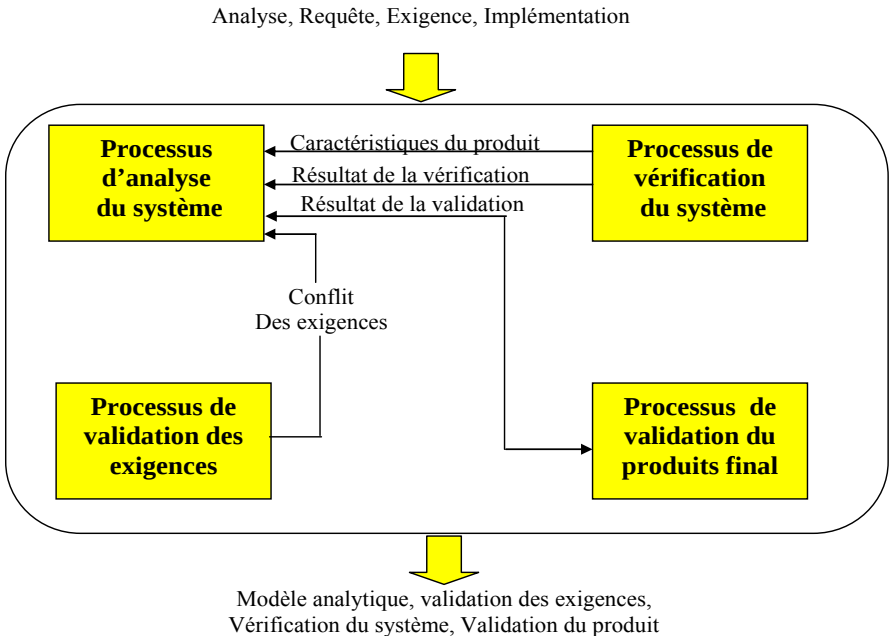


Figure II.8. Processus d'évaluation technique

IV.2.2 Le processus de validation des exigences

La validation des exigences est une étape cruciale pour la réussite d'un projet de conception. Elle permet de s'assurer que les exigences sont nécessaires et suffisante pour mener à bien le projet de conception.

E25 - Validation des déclarations des exigences :

Le concepteur doit s'assurer que les déclarations des exigences techniques et les déclarations des exigences spécifiées sont bien formulées. Pour cela il faut :

- Analyser chaque déclaration d'exigence par rapport aux exigences E16, E17, E18, et E19 et s'assurer de la capacité de préserver la compétitivité, la clarté, l'exactitude, la faisabilité, l'objectif, l'implémentabilité (implementability), la modifiabilité (modifiability), la suppression de l'ambiguïté, la singularité, l'aptitude au test et la possibilité de vérification.
- Analyser les déclarations des exigences par rapports aux exigences E16, E17, E18, et E19 par paire et par ensemble pour s'assurer de l'absence de la redondance, de la connectivité et de la suppression des conflits.

Les déclarations d'exigences techniques validées sont employées pour guider le développement des solutions de conception du système.

Une solution semi-formelle permet de s'assurer de la bonne formulation des exigences. Nous pouvant citer les langages *UML* et *SysML* pour aider à cette formulation.

E26 - Validation des exigences d'acquéreur :

Le concepteur doit s'assurer que l'ensemble des exigences d'acquéreur définies est conforme aux attentes et besoins de l'acquéreur.

Pour cela le concepteur doit prendre en compte les considérations suivantes :

- Choisir les méthodes et définir les procédures pour valider les exigences d'acquéreur définies à partir de l'exigence E-14,
- Analyser et comparer les exigences d'acquéreur collectées et identifiées avec les exigences d'acquéreur définies afin de déterminer la traçabilité descendante.
- Analyser et comparer l'ensemble des exigences d'acquéreur définies l'ensemble des exigences d'acquéreur collectées et identifiées afin de déterminer la traçabilité ascendante.

E27 - Validation des exigences des autres parties prenantes :

Le concepteur doit s'assurer que l'ensemble des exigences des autres parties prenantes définies est conforme aux attentes et besoins des autres parties prenantes.

La même démarche liée à la validation des exigences d'acquéreur doit être appliquée à la validation des exigences des autres parties prenantes.

E28 - Validation des exigences techniques du système :

Le concepteur doit s'assurer que l'ensemble des exigences techniques est conforme aux exigences validées de l'acquéreur et des autres parties prenantes. Pour cela il faut :

- Choisir les méthodes et définir les procédures pour valider les exigences techniques du système à partir de l'exigence (processus *EIA-632*) E16,
- Analyser et comparer l'ensemble des exigences validées d'acquéreur et des autres parties prenantes avec l'ensemble des exigences techniques du système afin de déterminer la traçabilité descendante.
- Analyser et comparer l'ensemble des exigences techniques du système définies avec l'ensemble validé des exigences d'acquéreur et des autres parties prenantes afin de déterminer la traçabilité ascendante.
- Re-valider les exigences techniques du système à chaque fois qu'un changement d'exigence affecte les exigences d'acquéreur, les exigences des autres parties prenantes ou les exigences techniques du système.

Les résultats de cette étape montrent que l'ensemble des exigences techniques du système a une traçabilité à partir des exigences validées des autres parties prenantes et sont nécessaires et suffisantes pour définir la représentation de la solution logique.

E29 - Validation de la représentation de la solution logique:

Le concepteur doit s'assurer que chaque ensemble de représentations de la solution logique est conforme au sous-ensemble assigné d'exigences techniques du système. Pour cela il faut :

- Choisir les méthodes et définir les procédures pour valider les ensembles des représentations de la solution logique et les exigences techniques déduites de E17,
- Analyser et comparer l'ensemble des exigences techniques validées du système aux ensembles des représentations de la solution logique définies et les exigences techniques dérivées pour déterminer la traçabilité descendante.
- Analyser et comparer l'ensemble des représentations de la solution logique définies et les exigences techniques dérivées et toute exigence technique non affectée avec l'ensemble validé des exigences techniques validées du système pour déterminer la traçabilité ascendante.
- Analyser les propositions faites pour définir les ensembles des représentations de la solution logique et les exigences techniques dérivées pour s'assurer qu'elles sont conformes aux exigences techniques du système.
- Re-valider les ensembles de représentations de la solution logique à chaque fois qu'un changement d'exigence est effectué qui affecte les exigences d'acquéreur, les exigences des autres parties prenantes ou les exigences techniques du système.

IV.2.3 Processus de vérification du système

E30 - Vérification de solution de la conception :

Le concepteur doit vérifier que chaque produit final défini par la solution de conception du système répond aux exigences de la représentation de la solution physique choisie. Pour cela, il faut :

- Prévoir la vérification de la solution de conception selon le plan de vérification.
- Effectuer la vérification de la solution de conception prévue suivant les méthodes et les procédures choisies dans l'environnement de vérification établi,
- Revérifier selon le plan de vérification remodelée ou les méthodes de test quand des conflits ont été déterminés.
- Enregistrer les résultats de vérification, incluant les actions correctives, l'expérience acquise, les résultats, les analyses de risque réalisées, les principales décisions prises et les tests accomplis.

La simulation est un outil puissant qui pourrait répondre à cette exigence (E30), notamment la simulation de Monte Carlo.

V Mise en œuvre de l'approche

L'approche n'exploite pas tous les processus de la norme *EIA-632*, elle se focalise sur les processus les plus importants :

V.1 Les exigences de sécurité

Parmi toutes les exigences formulées par l'acquéreur, certaines d'entre elles sont des exigences de sûreté et de sécurité. Les événements redoutés définissent des exigences de sûreté de fonctionnement dans le sens où leur prise en compte doit mener à une conception d'un système capable d'éviter ces événements.

Cette phase correspond aux exigences d'acquéreur et les parties prenantes, soit E14 et E15 de la norme *EIA-632*.

V.2 La modélisation des exigences et la solution logique

Afin de prendre en compte les exigences fonctionnelles et non fonctionnelles, une modélisation par réseau de Petri sera effectuée. La modélisation prend en compte les exigences élicitées. Concernant les exigences de sûreté précédentes, elles seront modélisées par les défaillances et les reconfigurations dans le réseau de Petri.

V.3 L'analyse de risque

L'analyse de risque permet d'identifier les risques techniques menant à des situations dangereuses. Une fois ces risques identifiés, il faut rechercher les causes probables et proposer des solutions pour les éviter.

Cette analyse peut être basée sur la recherche de scénarios redoutés [Sadou & al, 2006]. Elle sera présentée dans le chapitre 5 du manuscrit. La détermination des scénarios redoutés et leur analyse permettent au concepteur de proposer des solutions et de les évaluer.

V.4 La validation

La validation est décrite dans la norme à travers les exigences E24, E25, E28.

L'étape d'analyse de risque permet de déterminer les scénarios redoutés menant vers les états dangereux recensés. Suite aux solutions proposées par le concepteur pour éviter des états dangereux, une nouvelle procédure de recherche de scénarios redoutés sera effectuée. La quantification des probabilités d'apparition des nouveaux scénarios générés permet de valider le système d'un point de vue sûreté de fonctionnement s'il répond aux exigences de sûreté et de sécurité. Sinon la procédure est réitérée. Une nouvelle recherche de scénarios sera donc effectuée suivie d'une nouvelle analyse qualitative et quantitative des nouveaux scénarios générés avec une proposition de nouvelles solutions. Le schéma de la figure II.9 illustre la procédure itérative d'analyse de risque et de validation.

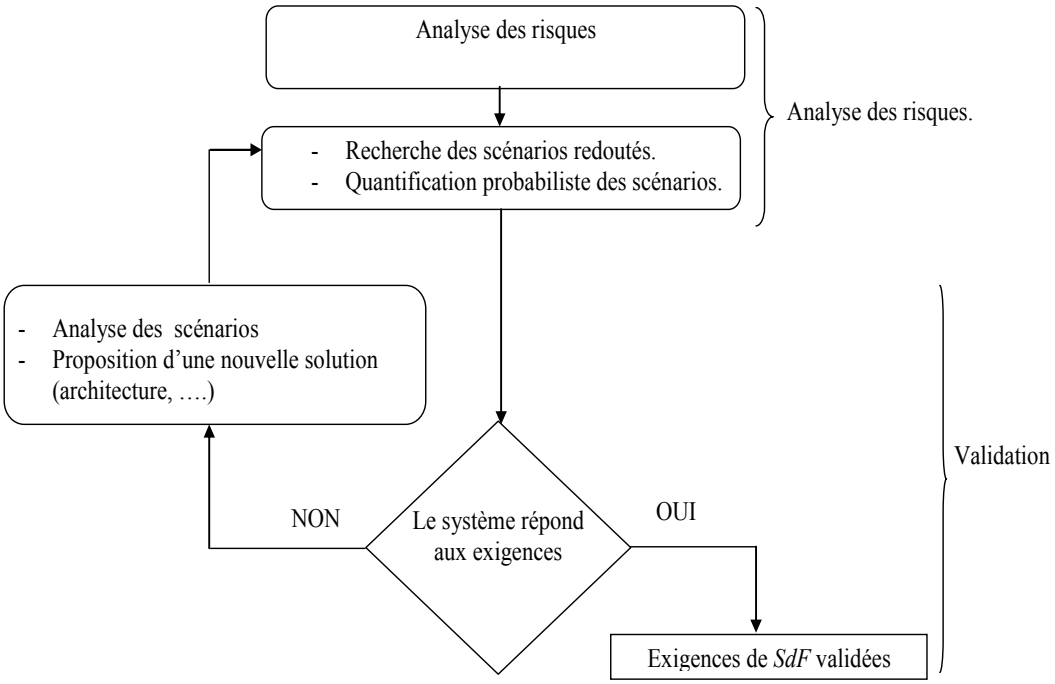


Figure II.9. Processus d'analyse des risques

VI Conclusion

Dans ce chapitre nous avons proposé une démarche pour l'intégration de la sûreté de fonctionnement dans les processus d'Ingénierie Système en s'appuyant principalement sur la norme *EIA-632*. Cette démarche peut être résumée comme suit:

- Eléments de la démarche : processus -> méthodes -> outils.
- Prise en compte de la norme *EIA-632* pour les processus d'ingénierie discutée précédemment.
- Les processus sont les processus d'Ingénierie Système.
- La prise en compte de la *SdF* sera définie comme processus spécifique (voir également les processus d'évaluation, vérification et validation adapté à la SDF)

Nous avons proposé une approche de mise en œuvre de certains processus spécifiques concernant les exigences, la solution logique et la validation. Ceci dit, beaucoup de travail reste à faire dans ce domaine. Les solutions possibles pour l'intégration de la sûreté de fonctionnement dans un cadre Ingénierie Système consistent à explorer les points suivants ; la modélisation formelle du processus d'ingénierie, l'intégration des normes de sécurité en vigueur (ARP 7454 pour l'aéronautique ou DO-178B pour la partie systèmes embarqués) et l'introduction de la notion de cycle de vie de la SdF des exigences jusqu'au retrait.

Chapitre 3 : Approche de modélisation

I Introduction

Dans les chapitres précédents nous avons présenté la problématique de la sûreté de fonctionnement et son intégration dans un cadre ingénierie système. Nous avons présenté différentes méthodes pour l'analyse de la fiabilité dynamique. Il convient maintenant de déterminer le modèle, l'étape de modélisation ayant un rôle crucial rôle crucial dans la conception et l'analyse des systèmes. Le formalisme choisi influe sur la manière d'aborder le problème. Une fois le formalisme choisi, il limite les méthodes et outils pour l'analyse de sûreté de fonctionnement.

Comme nous l'avons vu précédemment les systèmes embarqués sont typiquement des systèmes dynamiques hybrides. Les modèles de systèmes dynamiques hybrides (*SDHs*) sont utilisés dans plusieurs domaines d'application, tels que les systèmes automatisés, les systèmes de gestion du trafic (aérien et urbain), le contrôle des systèmes embarqués (avioniques, automobile…), les systèmes de production, les processus chimiques, la robotique, les réseaux de communication. Leur large applicabilité a inspiré beaucoup de chercheurs des domaines de la théorie de la commande et de l'informatique théorique. Une revue des différentes applications des *SDHs* est disponible dans [Alur & Al, 2000].

Les systèmes dynamiques hybrides (*SDHs*) sont des systèmes comportant des aspects discrets et continus. Selon le domaine de l'application, les *SDHs* sont modélisés comme des systèmes dynamiques avec une commande discrète [Gollu & Varaiya, 1989]. Les composantes continues sont fréquemment modélisées avec des équations différentielles ordinaires (*EDO ou ODES en anglais*). A l'occurrence d'un événement discret, le système décrivant la composante continue du *SDH* évolue jusqu'à un nouveau changement discret.

L'intérêt qu'ont suscité les systèmes dynamiques hybrides durant les 20 dernières années a donné naissance à une grande variété de modèles et de formalismes. La revue détaillée de ces différents modèles n'est pas présentée dans ce manuscrit. Plusieurs références traitent de ce

sujet, nous pouvons citer [Antsaklis & Koutsoukos, 2003], [Gueguen & Lefebvre, 2001], [Champagnat, 1998] et [Zaytoon, 01].

Brièvement, les formalismes de modélisation de systèmes hybrides peuvent être regroupés en trois classes. La première classe contient toutes les approches qui consistent à étendre le modèle continu en introduisant des variables discrètes par l'ajout de variables booléennes [Zaytoon, 01] afin de recouvrir toutes les configurations. La deuxième classe comprend des formalismes à événements discrets où de nouveaux éléments sont introduits pour représenter la dynamique continue, tel les réseaux de Petri hybrides [Alla & David, 2004]. Le réseau de Petri contient des places et transitions continues afin de modéliser le flux continu. La troisième classe contient des formalismes qui combinent un formalisme continu, décrit par des systèmes d'équations, avec un formalisme discret tel les réseaux de Petri ou les automates. On peut citer les automates hybrides [Henzinger & al, 1996] et les réseaux de *Petri Prédicats Transitions Différentiels* (*RdP PTD*) [Champagnat & al, 1998]. Dans les deux cas, une interface est définie pour représenter l'interaction entre les formalismes continus et discrets.

L'extension des formalismes continus tend à restreindre la flexibilité de modélisation de la partie continue. Ceci est également valable dans le cas de l'extension des formalismes discrets (restriction de la modélisation de la partie discrète). Les approches qui combinent un formalisme continu et un formalisme discret présentent une grande puissance de modélisation et de flexibilité par rapport aux deux premières classes [Gueguen & Lefebvre, 2001]. Ce sont ces arguments qui ont motivé notre choix d'un formalisme de la troisième classe.

Parmi les approches de cette classe, nous avons privilégié les formalismes dérivés des réseaux de Petri en raison de leurs avantages et leur capacité à modéliser le parallélisme, la synchronisation et le partage de ressources.

Les réseaux de Petri mixtes [Valentin-Roubinet, 2000] et les réseaux de *Petri Prédicats Transitions Différentiels* [Champagnat, 1998] sont deux exemples qui utilisent les formalismes de la troisième classe et qui adoptent les réseaux de Petri pour modéliser la partie discrète. Les aspects discrets et continus sont pris en compte par deux formalismes différents, chacun étant bien adapté à l'aspect visé [Andreu, 1996]. Contrairement aux autres approches où ils sont pris en compte de manière unifiée. Cette séparation des deux aspects discret et continu nous offre la possibilité de traiter séparément les deux aspects, comme nous allons le voir dans les chapitres suivants.

L'autre motivation qui nous a poussés à faire ce choix est que dans les que les réseaux de *Petri Prédicats Transitions Différentiels* sont plus appropriés que les réseaux de Petri mixtes en raison de l'absence de variables globales et la contrainte de capacité des places (marquage sauf des places).

Les RdP Prédicats-Transitions Différentielles sont bien adaptés pour une approche modulaire et compositionnelle de représentation des systèmes complexes, ce qui permet une conception structurée du modèle. Par contre, cette approche n'est possible que lorsque la décomposition en modules de la partie discrète (ensemble de réseaux de Petri partiels) coïncide bien avec la décomposition de la partie continue (en sous-ensembles de variables continues et en sous-ensembles d'équations différentielles et algébriques). Quand ces décompositions ne coïncident

pas, ceci implique la présence de variables continues partagées entre les modules. Dans ce cas, on parle d'interaction continue. Il faut donc spécifier explicitement que certaines variables sont partagées entre certains modules [Champagnat 1998].

Cependant, aucun de ces deux formalismes n'offre la possibilité d'une décomposition du système ou une modélisation progressive, ce qui rend la tâche de modélisation mais aussi l'analyse des systèmes complexes difficile. Une solution pour faciliter la modélisation des systèmes complexes est d'introduire le paradigme Orienté Objets (OO) dans le formalisme [Villani & al, 2004]. Le paradigme OO sera présenté et discuté dans ce chapitre. Son objectif principal est de structurer la décomposition du système et maîtriser sa complexité. En effet, la correspondance directe entre les objets du modèle et les entités réelles du problème offre une possibilité de modification et/ou de mise à jour, améliorant la réutilisation des modèles.

Dans la prochaine section, une brève introduction aux réseaux de Petri est présentée. Elle est suivie de la description des réseaux de *Petri Prédicats Transitions Différentiels*.

II Les réseaux de Petri

Si les automates [Alur & Dill, 1994] ont longtemps constitué l'approche classique de modélisation des systèmes à événements discrets, les réseaux de Petri (*RdP*) [Petri, 1962], [Murata, 1989] ont connu depuis leur invention un réel succès en raison de leur simplicité mathématique, des avantages de la représentation graphique et de leur compacité [Moody & Antsaklis, 1998]. C'est l'un des outils les plus populaires pour la modélisation de systèmes à événements discrets et les domaines d'applications sont très vastes.

II.1Rappel sur les réseaux de Petri

Définition III.1. (Réseau de Petri) [Valette, 2002] : Un *RdP* marqué est un 5-uplet *R*= <*P,T,Pré,Post,M*$_0$> avec :

- P : un ensemble fini de places,
- T : un ensemble fini de transitions, tel que $P \cap T = \{\ \}$
- *Pré* : $P \times T \rightarrow \mathrm{N}$ l'application *places précédentes* qui indique le nombre de jetons prélevés par les transitions dans les places en amont,
- *Post* : $T \times P \rightarrow \mathrm{N}$ l'application *places suivantes* qui indique le nombre de jetons déposés par les transitions dans les places en aval,
- M_0 : le marquage initial.

Un *RdP* peut être vu comme un graphe avec deux types de sommets : des places et des transitions. Les places sont marquées par des jetons et le réseau évolue par le franchissement (tir) des transitions selon les applications *Pré* et *Post*. On utilise également la notation *C*=*Post*-*Pré* où *C* est la matrice d'incidence.

II.2RdP t-temporels

Définition III.2. Un réseau de Petri t-temporel est une paire N_{tl}=<*R, D*> où *R* est un réseau de Petri <*P,T,Pré,Post,M0*>marqué et *D* est une fonction qui à chaque transition t_i du réseau fait correspondre un intervalle statique de temps $d\ (t_i) = [d_{imin}(t_i), d_{imax}(t_i)]$ qui décrit une durée de sensibilisation.

L'intervalle de temps est associé aux transitions. La transition t_i doit rester sensibilisée durant au moins d_{imin} unités de temps et au plus d_{imax} unités de temps avant d'être franchie. Par contre les jetons peuvent à tout moment être consommés par une autre transition.

II.3Réseaux de Petri de haut niveau

Les réseaux de Petri ont été utilisés avec succès dans un certain nombre d'applications réelles dans divers domaines. Cependant, deux inconvénients principaux ont limité leur utilisation pour des applications de grande complexité.

Le premier est que le réseau de Petri n'est pas adapté pour la manipulation des données. Même pour des problèmes simples, la structure du réseau devient trop complexe. Le deuxième inconvénient est qu'il n'y a pas de hiérarchie dans le réseau de Petri et qu'il n'est donc pas possible de construire le modèle d'un système complexe comme une composition de sous modèles.

Pour résoudre ce problème de la manipulation de données, beaucoup de travaux ont proposé l'extension des formalismes réseau de Petri. Ces extensions sont appelées réseaux de Petri de haut niveau. On peut citer, les réseaux de Petri Colorés [Jensen, 1997] et les réseaux de *Petri prédicats transitions* [Genrich, 1987].

Dans les réseaux de Petri Colorés, le pouvoir de description est augmenté en associant des couleurs aux jetons, aux places et aux transitions. Chaque jeton a une couleur qui lui permet d'être distingué des jetons portant d'autres couleurs. Un ensemble de couleurs est associé à chaque place et détermine les couleurs des jetons qui peuvent être déposés dans la place. Pour chaque transition est associé un ensemble de couleurs qui représente les différentes manières de tirer la transition.

Les réseaux de *Petri Prédicats Transitions* (*RdP PT*) [Genrich, 1987], introduisent le concept de variables. Chaque transition possède une fonction de sensibilisation spécifiée comme une formule logique avec des variables. Les transitions sont des règles d'un système logique de premier ordre, qui est un système logique avec des variables.

Une définition simplifiée d'un réseau de *Petri Prédicats Transitions* est la suivante :

Définition III.3 (*RdP PT*) Un réseau de *Petri prédicats transitions* est un 3-tuple *RdP PT* = <*R, A,* M_0>, où :

- *R* est la structure de réseau de Pétri Définie par le 4-tuple < *P, T, Pré, Post*>,
- *A* est l'annotation du *RdP PT*, définie par le 4-tuple *A*=<*X*, A_x, A_c, A_a >, où :

- X est un ensemble de variables,
- A_x est une application associant à chaque arc un vecteur de variable X,
- A_c est une application associant à chaque transition une condition sous la forme d'un prédicat utilisant les variables,
- A_a est une application associant à chaque transition une action sous la forme d'une suite d'affectation de valeurs aux variables.
- M_0 est le marquage initial du réseau de Petri. Chaque jeton est un vecteur de variables. Le marquage initial définit les valeurs des variables du jeton.

Dans un réseau de *Petri Prédicats Transitions*, une transition est sensibilisée (ou non) pour un ensemble défini de jetons de ses places d'entrée. Si les variables associées à l'arc sont remplacées par les valeurs des variables des jetons et la condition de sensibilisation est vraie, alors la transition est sensibilisée. Une fois sensibilisée, la transition peut être tirée. L'action associée à une transition définit les valeurs des variables des arcs sortants. Ce sont les mêmes valeurs des jetons générés dans les places de sortie par le tir de la transition.

II.4 Réseaux de Petri Prédicats Transitions Différentiels

L'utilisation des variables dans les réseaux de *Petri Prédicats Transitions* a motivé son application pour la modélisation des systèmes hybrides, ce qui a donné naissance aux réseaux de *Petri Prédicats Transitions Différentiels* (*RdP PTD*) [Champagnat, 1998]. Dans les réseaux de *Petri Prédicats Transitions*, les variables associées aux jetons ne peuvent être modifiées que par un franchissement de transition. En d'autres termes, ces variables ne peuvent avoir une évolution continue. L'idée des réseaux de *Petri Prédicats Transitions Différentiels* est que chaque marquage modélise une configuration du système. Un système d'équations différentielles est associé à chaque place. Ce système activé par l'arrivée de jetons, décrit l'évolution continue dans le temps des variables associées aux jetons présents dans la place.

Les réseaux de *Petri Prédicat Transitions Différentiels* possèdent des fonctions de sensibilisation associées aux transitions. Elles définissent des seuils sur les variables continues mises en jeu dans les équations associées aux places d'entrées. Lorsque la variable atteint le seuil défini pour la fonction de sensibilisation, la transition sensibilisée à laquelle est associée cette fonction est tirée. Un autre élément des réseaux de *Petri Prédicats Transitions Différentiels* est la fonction de jonction. Les fonctions de jonction sont utilisées pour introduire des discontinuités sur les variables continues et sont similaires aux actions dans le cas des réseaux de *Petri Prédicats Transitions*.

La définition des réseaux de *Petri Prédicats Transitions Différentiels* est la suivante :

Définition III.4. (*RdP PTD*) un réseau de *Petri Prédicats Transitions Différentiel* est un 3-uplet, *RdP PTD* = < R, A, M_0 >, où :

- R est la structure du réseau de Petri définie par le 4-tuple < P, T, *pré*, *Post*>,
- A est l'annotation du *RdP PTD*, définie par le 4-tuple A=<X, A_p, A_f, A_e, A_j >, où :

- X est l'ensemble des variables de A,
- A_p est une application qui associe à la place P_i un vecteur de variable X_{pi} de l'ensemble des variables X, avec :

$$X = \cup_{pi \in P} X_{pi}$$

- A_f est une application qui associe à chaque place P_i un système d'équations différentielles et/ou algébriques F_i :

$$F_i(\dot{X}_{i,} X_i, t) = \begin{bmatrix} f_{ij}(\dot{X}_{i,} X_i, t) \\ \vdots \\ f_{ik}(\dot{X}_{i,} X_i, t) \end{bmatrix}, \quad j = 1...k$$

Où $f_{ij}…fik$ sont des équations représentant l'évolution des variables continues. Ainsi chaque place du réseau (qui représente un état discret) peut décrire l'évolution de l'ensemble des variables continues X_i (grâce au système d'équations F_i).

- A_e est une application qui associe à chaque transition t_i une fonction de sensibilisation e_i. La fonction de sensibilisation utilise les variables A_p qui sont associées aux places d'entrée de la transition. Le seuil e_i associé à la transition t_i est défini comme la première solution de :

$$e_i(\dot{X}_i, X_i, t) = 0$$

- A_j est une application qui associe à chaque transition t_i une fonction de jonction j_i. Une fonction de jonction est activée lors du franchissement de la transition correspondante. La fonction de jonction j_i associée à la transition t_i, lors du franchissement de t_i à la date t calcule les valeurs des variables ainsi que leurs dérivées.

$$j_i : \begin{cases} \dot{X}(t^+) = j_{i\ \dot{X}}(\dot{X}_i, \dot{X}_i, t^-) \\ X(t^+) = j_{i\ X}(\dot{X}_i, X_i, t^-) \end{cases}$$

Ce qui signifie que les valeurs des variables, et de leurs dérivées, juste après t (à t^+) sont calculées à partir de valeurs des variables, et de leurs dérivées, juste avant t (à t^-). Ceci permet de redéfinir les attributs des jetons en cas de discontinuité.

- M_O est le marquage initial. Chaque jeton est un vecteur de variables identique au vecteur de variables associé à la place. Le marquage initial définit les valeurs des variables associées au jeton à l'instant $t=0$.

Exemple

La figure III.1 représente un *RdP prédicats transitions différentiel* décrivant le remplissage d'un réservoir avec un débit *qe*. A chaque place est associé un système d'équations qui décrit le comportement continu du système dans la configuration concernée. La place P1 est initialement marquée par un jeton portant l'information sur le volume courant dans le réservoir (variable V). Lorsque la place P3 est marquée par un jeton portant l'information sur le débit d'entrée, la transition t1 est sensibilisée. La fonction de sensibilisation qui lui est associée étant définie vraie, la transition est franchie. La place P2 est alors marquée, le système d'équations F2 est activé et les variables mises en jeu sont initialisées, par la fonction de jonction j1, aux valeurs portées par le jeton (volume courant et débit d'entrée). Le volume est calculé et lorsqu'il atteint la valeur Vmax spécifiée par la fonction de sensibilisation e2 associée à la transition t2, celle-ci est franchie. La place P1 est marquée par un jeton dont l'information sur le volume a été mise à jour par la fonction de jonction j2 à la valeur calculée par l'équation F2.

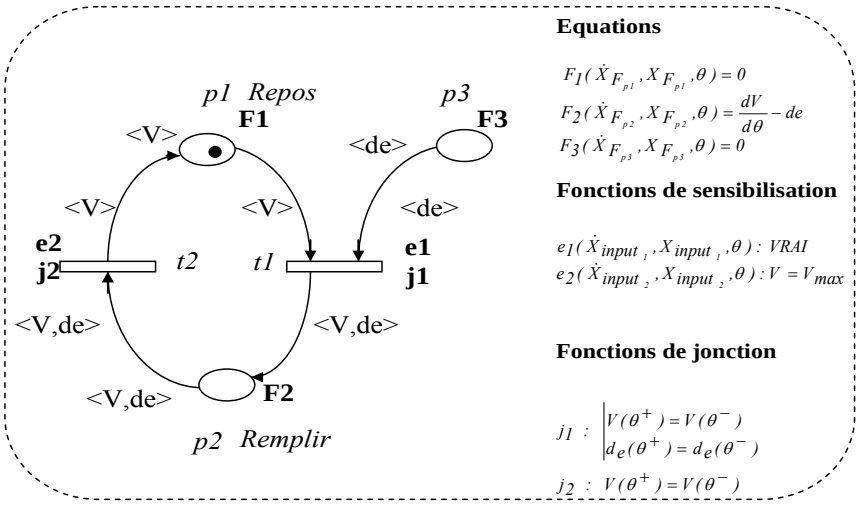


Figure III.1. Exemple de RdP PTD

III Les réseaux de Petri et le paradigme Orienté Objets

L'origine du paradigme orienté objets (*OO*) remonte aux années 60 avec l'introduction de la notion d'encapsulation, regroupant les données et les opérations dans des entités appelées *objets*. Initialement, le paradigme *OO* a été utilisé pour la structuration et l'organisation des programmes informatiques [Paludetto, 1991]. A partir des années 80, son utilisation s'est étendue à la conception des systèmes.

Le principe du paradigme orienté objets consiste à décomposer le système en un certain nombre d'objets qui interagissent entre eux. Un objet est une entité avec des attributs, un comportement, une mémoire et une identité [Booch, 1994], [Rumbaugh & al, 2004]. Les attributs représentent les données du système. Le comportement est composé d'opérations et de méthodes. La mémoire signifie que l'état de l'objet n'est pas réinitialisé à chaque fois qu'il est visité. L'identité distingue

les objets du système les uns des autres, même lorsqu'ils ont les mêmes attributs et comportement.

Outre le concept d'objet, les bases du paradigme orienté objets incluent les concepts d'*encapsulation*, *classification* et *héritage*. L'*encapsulation* stipule qu'un objet est constitué d'un corps (implémentation interne) et une interface (représentée par les méthodes qui permettent à d'autres objets d'agir sur son comportement). L'interface détermine également comment l'objet peut agir sur d'autres objets. La structure interne est cachée et garantit que la vue externe de l'objet est indépendante de l'exécution interne. L'interface contient toute l'information nécessaire pour élaborer une communication avec l'objet. La connaissance de l'exécution interne n'est pas nécessaire.

Les objets d'un système sont regroupés en *classes*. Une classe regroupe les objets partageant les mêmes attributs, opérations, relation et sémantique. Les objets d'une classe ont le même comportement et la même structure de données. Une classe est utilisée comme modèle de création de nouveaux objets, appelés instances de la classe.

L'*héritage* permet de définir une classe (couramment appelée classe *enfant*) à partir de la définition d'une autre classe (couramment appelée classe *parent*). La classe *enfant* hérite automatiquement de toutes les méthodes et les attributs de la classe *parent*. La réutilisation fournie par l'héritage est l'un des principaux avantages du paradigme orienté objets.

III.1 Approches d'intégration des réseaux de Petri et du paradigme OO

La combinaison du paradigme *OO* et des réseaux de Petri a été l'objet de plusieurs travaux. Ces travaux peuvent être organisés en trois groupes.

III.1.1 Objets dans les réseaux de Petri

Cette approche considère les jetons qui se déplacent dans le réseau de Petri comme des objets. L'idée est de modéliser les états d'un ensemble d'entités identiques par un réseau de Petri unique dans lequel chaque jeton représente une entité particulière. Ces jetons doivent donc être individualisés et porteurs d'informations. Quand une transition est tirée, elle exécute la méthode d'un objet et modifie les valeurs de ses attributs. Le *HyNet* (*Hybride high-level Petri net*) [Wieting, 1996] est une approche de ce groupe pour la modélisation des systèmes hybrides.

III.1.2 Réseaux de Petri dans les objets

Une autre façon d'aborder le paradigme OO dans les RdP est d'utiliser les réseaux de Petri pour décrire le comportement interne des différents objets. Le marquage du réseau de Petri définit l'état courant de l'objet. Les méthodes fournies par l'objet sont associées aux places. Elles représentent l'interface de l'objet et sont accessibles aux autres objets. D'autres places et transitions modélisent le comportement interne de l'objet et sont encapsulées. The hybrid object net [Drath, 1998] est un exemple utilisé pour la modélisation des systèmes hybrides. Un exemple

pour la modélisation des systèmes à événements discrets est le G-CPN (g-coloured Petri net) [Guerrero & al, 2001].

III.1.3 Approches mixtes

Les deux approches présentées ci-dessus ne sont pas incompatibles et peuvent se compléter. Les approches du troisième groupe combinent les principes des deux premières. Une structure hiérarchique est créée à l'intérieur du réseau de Petri. Elle permet d'avoir plusieurs niveaux d'intégration des objets dans les *RdP* et des *RdP* dans les objets. Ces approches consistent à modéliser le système avec un réseau de Petri. Un jeton dans le réseau du système est un objet. Le comportement d'un objet est décrit par le réseau de Petri le modélisant. Les jetons dans les réseaux des objets peuvent être eux même des objets, d'où la hiérarchisation. Ainsi dans ce cas, le réseau d'un objet est le réseau du système d'un point de vue jeton [Valk, 1998]. Une des approches de ce groupe est les *réseaux de Petri objets* (*objet Petri nets*) [Lakos, 1995].

L'approche de modélisation qui concerne un formalisme intégrant les aspects orientés objets et les réseaux de *Petri prédicats transitions différentiels* que nous allons présenter (et que nous utiliserons par la suite) doit prendre en compte les remarques suivantes :

1) la possibilité des construire un modèle global du système à partir des modèles représentant les différents objets qui le composent. Ceci offre la possibilité de visualiser le comportement du modèle complet et de détecter la présence d'inconsistances dans le modèle.
2) L'encapsulation des objets doit fournir une définition claire des interfaces des différents objets. Les objets communiquant uniquement à travers leurs interfaces, assurant ainsi l'intégrité de leur comportement.
3) L'utilisation des mécanismes d'hiérarchisation doit être modérée. En effet l'utilisation de règles complexes peut compromettre la lisibilité et la signification graphique du réseau de Petri (un des avantages de ce formalisme).

III.2 Les réseaux de Petri Prédicats Transitions Différentiels Orientés Objets

Les remarques de la section précédente sont le point de départ de l'introduction des concepts orientés objets dans les réseaux de Petri proposée par Emilia Villani [Villani & al, 2004], [Villani & al, 2007]. Le formalisme résultant pour la modélisation des systèmes hybrides est appelé réseaux de *Petri Prédicats Transitions Différentiels Orientés Objets* (*RdP PTD-OO*).

III.2.1 Modélisation des classes et des objets

Les concepts de base du paradigme orienté objets sont les classes et les objets. La définition des *RdP PTD-OO* est basée principalement sur ces deux concepts. Le système est modélisé par un *RdP PTD-OO* composé d'un ensemble de sous-réseaux *RdP PTD-OO*. Chaque sous-réseau est

associé à une classe et modélise son comportement. Le marquage de chaque sous-réseau décrit l'état courant des objets de la classe.

Définition III.5. (*RdP PTD-OO*) : un réseau de *Petri Prédicats Transitions Différentiel* Orienté Objets : *RdP PTD-OO* = $\{C_1, C_2, \ldots, C_n\}$, avec n le nombre de classes qui composent le modèle du système. Chaque classe est modélisée par un sous-réseau de *Petri prédicats transitions différentiel orienté objets* C_i.

La définition d'un sous-réseau *RdP PTD-OO* est basée sur la définition d'un *RdP PTD*. Les variables d'une classe *Ci* représentent les attributs de la classe.

Définition III.6. *(Sous-réseau RdP PTD-OO)* : chaque sous-réseau *RdP PTD-OO* est un 3-uplet, $C_i = < R_i, A_i, M_{0\_i} >$, où :

- R_i est une réseau de Petri défini par le 4-uplet $< P_i, T_i, \Pr é_i, Post_i >$, où :
  - $P_i = \{P_{1\_i}, P_{2\_i}, \ldots, P_{m\_i}\}$ est un ensemble fini de place,
  - $T_i = \{T_{1\_i}, T_{2\_i}, \ldots, T_{n\_i}\}$ est un ensemble fini de transitions,
  - $P_i \cap T_i = \{\ \}$, $P_i \cup T_i \neq \{\ \}$,
  - $\Pr é_i : P_i \times T_i \rightarrow (0,1)$. (0 si aucun arc ne relie la place à la transition, sinon 1),
  - $Post_i : P_i \times T_i \rightarrow (0,1)$.
- A_i est l'annotation de C_i, A_i=<X_i, A_{pi}, A_{fi}, A_{ei}, A_{ji} >:
  - X_i est l'ensemble des variables (voir définition III.9),
  - A_{pi} est une application qui associe à place P_{k_i} un vecteur de variable X_{pk_i} de l'ensemble des variables X_i (voir définition III.8),
  - A_{fi} est une application qui associe à chaque place P_{k_i} un système d'équations différentielles et/ou algébriques F_{k_i}.
  - A_{ei} est une application qui associe à chaque transition t_{k_i} une fonction de sensibilisation e_{k_i}. La fonction de sensibilisation utilise les variables A_{pi} qui sont associées aux places d'entrée de la transition.,
  - A_{ji} est une application qui associe à chaque transition t_{k_i} une fonction de jonction j_{k_i}.
- M_{O_i} est le marquage initial du sous-réseau *RdP PTD-OO*.

Exemple (Chaudière à vapeur):

L'exemple de la chaudière à vapeur [Abrial, 1995] a constitué un benchmark pour la spécification et la vérification formelle. La chaudière à vapeur se compose d'un réservoir d'eau (*Res1*), de quatre pompes (on se limite à deux pompes dans ce manuscrit *Pom1* et *Pom1*) et de

capteurs qui mesurent les débits d'alimentation des pompes, le débit d'évacuation de la vapeur et le niveau d'eau (figure III.2). Le contrôle s'opère à l'aide du calculateur qui doit maintenir le niveau d'eau dans la chaudière entre les niveaux *N1* et *N2*.

La loi de commande utilise simplement l'information sur le niveau d'eau pour décider du nombre de pompes, alimentant la chaudière, qui doivent être actives. On suppose aussi qu'à l'état initial le volume d'eau dans la chaudière est compris dans l'intervalle *[N1, N2]* et que les pompes sont inactives. Ceci est assuré par l'étape d'initialisation du système qui consiste à activer les pompes ou ouvrir la vanne de vidange pour ramener le niveau d'eau à un niveau compris dans l'intervalle *[N1, N2]*. Dans le mode de fonctionnement normal, la loi de commande se base sur quatre seuils de niveau d'eau *M1, L', L, U, U'* et *M2* pour décider du nombre de pompe à activer. Le tableau III.1 illustre la stratégie de commande.

Le *RdP PTD-OO* est composé de 3 classes : *C1 – Pompe*, *C2 – réservoir* et *C3 –* système de commande.

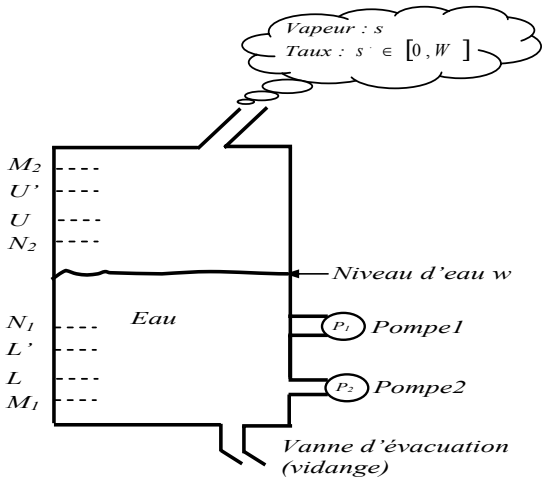


Figure III.2. Chaudière à vapeur

Niveau d'eau w	Pompes
$L' \leq w \leq L$	Les deux pompes actives
$L \leq w \leq N_1$	Une seule pompe active
$N_1 \leq w \leq N_2$	Aucune action
$N_2 \leq w \leq U$	Une seule pompe active
$U \leq w \leq U'$	Les deux pompes inactives

Tabueau III.1. Stratégie de commande de la chaudière à vapeur.

La figure III.3 représente les sous-réseaux des classes *C1* – Pompe et *C2* - réservoir. La classe *C1 – Pompe* possède deux états discrets : ouverte et fermée. La variable *d* représente le débit de la pompe. La classe *C2 – réservoir* possède 3 états discrets : remplissage, remplissage-vidange (en fonction de la vapeur dégagée, il peut être en phase de remplissage ou en phase de vidange) et vidange. Les variables *w*, *s*, *d1.1* et *d2.1* représentent respectivement le volume dans le réservoir,

le taux de vapeur qui se dégage et les débits d'alimentation des deux pompes. *l1* et *l2* sont des variables liées à l'interface de la classe *C2* (elles seront présentées en détail plus loin dans le manuscrit).

Une fois que les classes et leurs sous-réseaux *RdP PTD-OO* sont définis, l'étape suivante est la définition de l'ensemble d'objets, instances des différentes classes. Les objets de la classe *Ci* sont $O_{1.i}, O_{2.i}, \ldots, O_{n.i}$, où *n* est le nombre d'objets de la classe *Ci*. La configuration discrète d'un objet *Oj.i* est représentée par un ou plusieurs jetons du sous-réseau de la classe à laquelle il appartient (*mj.i*). L'état continu, quand à lui est modélisé par l'instanciation *Xj.i* des variables *Xi*.

Le marquage du sous-réseau *RdP PTD-OO* de la classe est la composition des sous marquages qui représentent l'état des différents objets de la classe.

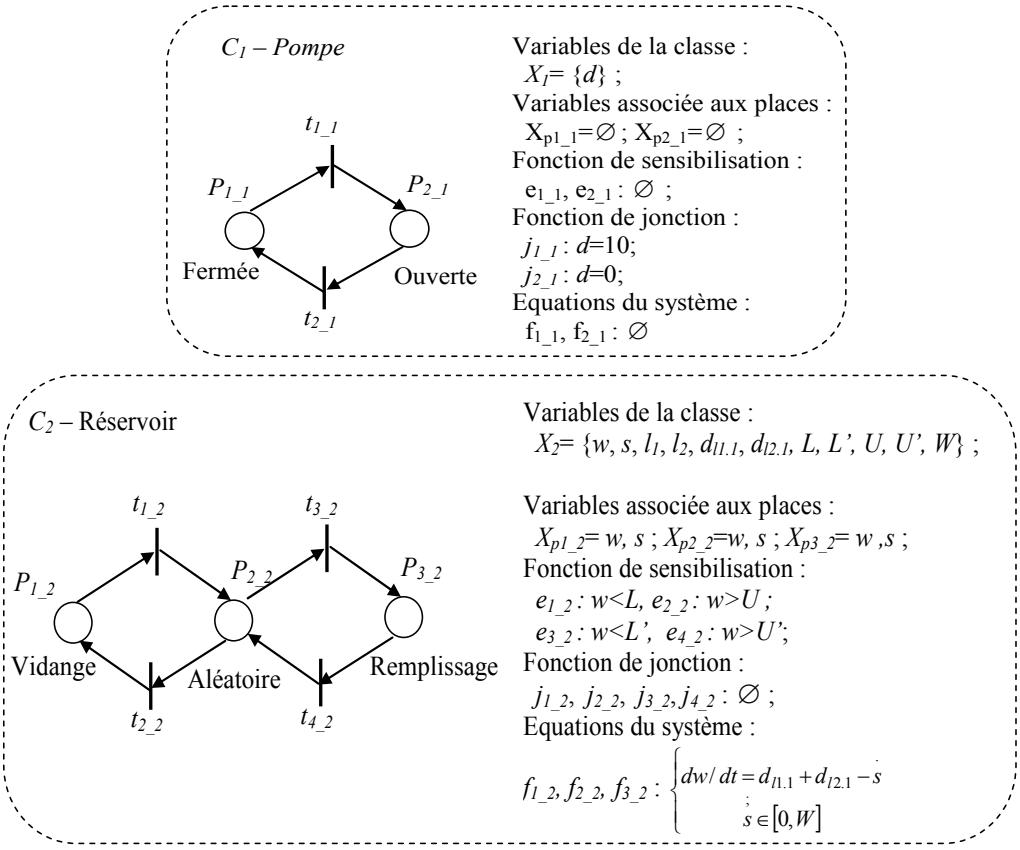


Figure III.3. Classes correspondant à la chaudière à vapeur

Définition III.7. (Marquage d'un sous-réseau *RdP PTD-OO*) : Le marquage d'un sous-réseau *RdP PTD-OO* est composé d'un ensemble de marquages, $M_i = \{O_{1.i}, O_{2.i}, \ldots, O_{n.i}\}$ qui représente l'état des objets de la classe C_i :

- $O_{j.i}$ est un 2-uplet $O_{j.i} =< X_{j.i}, m_{j.i} >$, où :
  - $X_{j.i}$ est une instanciation de l'ensemble des variables X_i du sous-réseau.

- $m_{j.i} : P \rightarrow (0,1)$ définit les jetons du sous-réseau qui représentent la configuration discrète des objets

La définition II.7 impose qu'un objet n'est représenté qu'avec un seul jeton dans une place.

Exemple (Chaudière à vapeur) :

La chaudière à vapeur est composée des objets *O1.1 – Pom1* et *O2.1 – Pom2* de la classe *C1 – Pompe* et *O1.2 – Res1* de la classe *C2 – réservoir*. Un marquage possible de ces objets est présenté dans la figure III.4. La configuration discrète (marquages) est présentée dans la figure III.4.

- $O_{1.1}$ – Pom_1 :
  - Instanciation des variables : $X_{1.1}$: d=0 ;
  - Marquage du réseau de Petri : $m_{1.1}$= {1,0} ;
- $O_{2.1}$– Pom_2 :
  - Instanciation des variables : $X_{2.1}$: d=10 (le débit de la pompe est de 10 *l/min*) ;
  - Marquage du réseau de Petri : $m_{2.1}$= {0,1} ;
- $O_{1.2}$ – Res_1 :
  - Instanciation des variables : $X_{1.2}$: w=20 ; s= 1; l_1=1 ; l_2=2 ; $d_{l1.1}$=0 ; $d_{l2.1}$=10.

    L=$cste_1$, L'= $cste_2$, U= $cste_3$, U'= $cste_4$, W= $cste_5$
- Marquage du réseau de Petri : $m_{1.2}$= {0, 1, 0} ;

Pour chaque objet *Oj.i,* une seule place définit à la fois la valeur des variables *Xj.i.* Le marquage initial de *Oj.i* doit être défini de telle sorte que tous les marquages accessibles *mj.i* à partir du marquage initial obéissent à cette règle

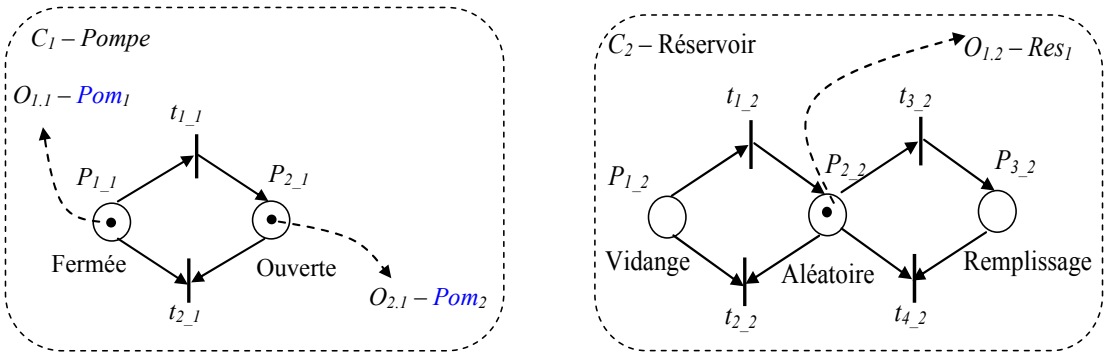


Figure III.4. Sous marquage des objets de la chaudière à vapeur.

Définition III.8. : Si $m_{j.i}$ est un marquage accessible de l'objet $O_{j.i}$ et $\{P_{a\_i}, P_{b\_i}\} \subset m_{j.i}$ alors $X_{pa\_i} \cap X_{pb\_i} = \varnothing$

L'ensemble des variables d'une classe est composé des paramètres (constantes) (*Xco_i*), variables internes (*Xint_i*), variables publiques (*Xpb_i*), variables images (*Xim_i*) et variables externes (*Xext_i*).

L'intersection entre ces différents sous-ensembles est nulle :

$$X_{co\_i} \cap X_{\text{int}\_i} \cap X_{pb\_i} \cap X_{im\_i} \cap X_{ext\_i} = \varnothing .$$

Les constantes ne varient pas avec l'évolution du temps dans le cycle de vie de l'objet. Cependant, deux objets d'une même classe peuvent avoir des constantes de valeurs différentes. Les variables externes sont des variables non modélisées dans le réseau *RdP PTD-OO*. Ce sont des signaux d'entrée (de commande). La différence entre les variables internes, externes et images est liée à la communication entre objets. Deux objets peuvent partager des données en partageant des variables. Les variables internes (*Xin_j.i*) d'un objet sont accessibles en lecture et écriture uniquement à l'objet (*Oj.i*) auquel elles appartiennent. Les variables publiques peuvent être accessibles aux autres objets mais uniquement en lecture. Si une variable *M* de *Xpb_j.i* est lue par un autre objet *Ok.l*, alors *M* fait partie des variables image de l'objet *Ok.i* et appartient donc à *Xim_k.l*. La variable *M* sera donc spécifiée dans l'ensemble des variables publiques (*Xpb_i*) de la classe *Ci* et dans l'ensemble des variables images (*Xim_l*) de la classe *Cl*.

Définition III.9 : Toute variable image de l'ensemble X_{im_l} de la classe C_l est associée à une variable publique de l'ensemble X_{pb_i} d'une classe C_i :

- A chaque variable de l'ensemble X_{im_l} est associée une variable X_l appelée l_n, où n est un nombre entier.
- l_n spécifie l'objet qui contient la variable qui va être lue.
- La variable de X_{im_l} est appelée $M_{ln.i}$, où M est le nom de la variable dans la classe d'origine (C_i).

Exemple

Ci-dessous, la classification (X_{co_i}, X_{int_i}, X_{pb_i}, X_{im_i} et X_{ext_i}) des variables des différentes classes de la chaudière à vapeur.

- C_1 – Pompe :

$$X_{co\_1} = \varnothing;\quad X_{\text{int}\_1} = \varnothing;\quad X_{pb\_1} = \{d\};\quad X_{im\_1} = \varnothing;\quad X_{ext\_1} = \varnothing;$$

- C_2 – Réservoir :

$$X_{co\_2} = \{L, L', U, U', l_1, l_2, W\};\quad X_{\text{int}\_2} = \varnothing,\quad X_{pb\_2} = \{w, s\}$$

$$X_{im\_2} = \{d_{l1.1}, d_{l2.1}\};\quad X_{ext\_2} = \varnothing;$$

La figure III.5 présente le partage de variables entre les différents objets ; $O_{1.1}$ – Pom_1, $O_{2.1}$ – Pom_2, $O_{1.2}$ – Res_1.

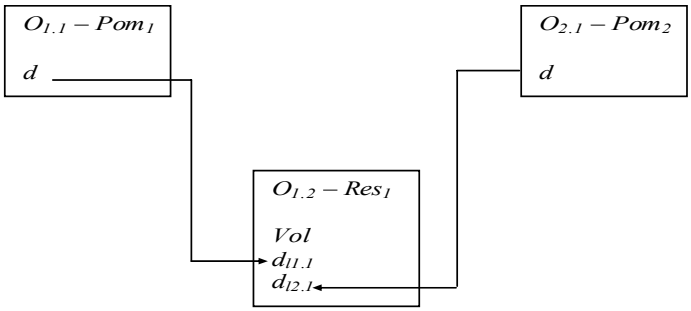


Figure III.5. Partage de variable entre les différents objets.

III.2.2 Communication entre objets

Dans un *RdP PTD_OO*, les communications entre les objets sont de deux natures différentes. Les communications continues sont représentées par le partage de variables (présentées dans la section précédente). Certains objets accèdent en lecture aux variables publiques d'autres objets et peuvent ainsi mettre à jour leurs variables images. L'autre nature des communications est discrète. Les objets communiquent par des appels et offres de méthodes.

L'interface discrète d'un objet est composée de deux ensembles de transitions ; les méthodes offertes par la classe et les méthodes utilisées par la classe. Un objet *O1* peut appeler une méthode offerte par l'objet *O2*. Les deux objets communiquent par la fusion des deux transitions. La première transition représente l'appel de méthode et la seconde l'offre de méthode.

Définition III.10 : l'interface offerte par une classe C_i est composée de :

- Un ensemble de variables publiques X_{pb_i}.
- Un ensemble de transitions T_{p_i} avec $T_{p\_i} \subset T_i$.

Définition III.11 : l'interface utilisée par une classe C_i est composée de :

- Un ensemble de variables images X_{im_i}.
- Un ensemble de transitions T_{u_i}, avec $T_{u\_i} \subset T_i$.

Quand un objet *Oj.k* de la classe *Ck* appelle (transition *tb_k*) une méthode *ta_i* de la classe *Ci*, il doit connaître quel objet de la classe *Ci* offre cette méthode. La variable *ln* de la classe *Ck* mémorise l'information concernant la méthode appelée.

Quand les transitions *ta_i* et *tb_k* sont sensibilisées, elles sont tirées simultanément (considérées comme une seule transition, fusion dynamique).

Définition III.12 : A chaque transition T_{u_k} de la classe C_k est associée à une transition T_{p_i} de la classe C_i.

- A chaque transition T_{u_k} est associée une variable l_n de X_k, avec n un nombre entier.

- l_n spécifie l'objet de la classe C_i qui offre la méthode appelée.

Dans la représentation graphique des réseaux *RdP PTD-OO*, les transitions représentant des appels de méthodes (T_{u_i}) sont représentées par des rectangles pleins (noirs), les transitions représentant des offres de méthodes par des rectangles vides (blancs).

Exemple :

Dans l'exemple de la chaudière à vapeur, la figure III.6 représente les sous-réseaux *RdP PTD-OO* des interfaces des deux classes *C1* - pompe et *C2* - réservoir. Seuls les offres et appels de méthodes sont représentés, les différentes fonctions telles que les fonctions de sensibilisation sont omises.

L'alimentation du réservoir en eau se fait suite à l'ouverture de (s) la pompe (s) (appels à méthodes représentés par les transitions *t1_2* et/ou *t3_2*) *Pomp1* et/ou *Pomp2*. Le réservoir appelle (l'appel est représenté par la transition *t1_1*) la méthode offerte par la classe *C1* pour les deux pompes (identifiée par les variable *l1* et *l2*). Ces méthodes exécutées par *t1_2* → *t1_l1.1* et *t3_2* → *t1_l2.1*. Le réservoir est alors en phase de remplissage. Une fois le réservoir alimenté et en cas de dépassement des seuils, l'une ou les deux pompes sont fermées suite aux appels à méthode représentés par les transitions *t2_2* et *t4_2*. Les méthodes sont exécutées par *t2_2* → *t1_l1.1* et *t4_2* → *t1_l2.1*.

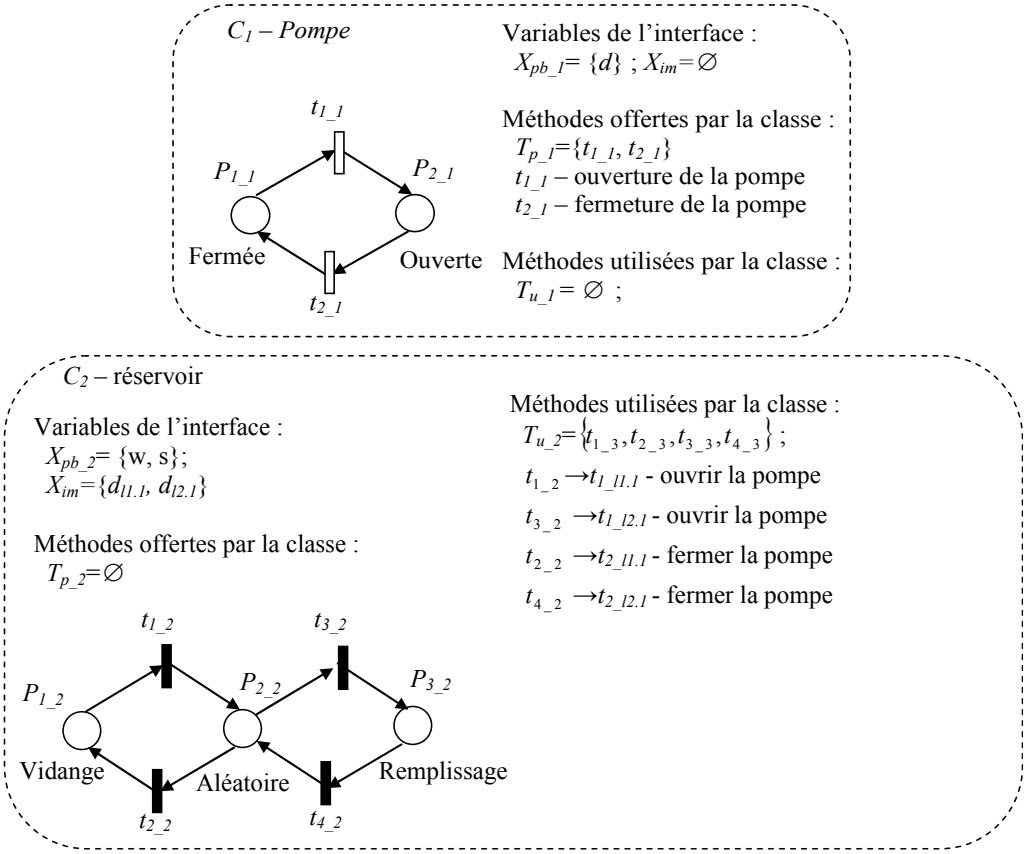


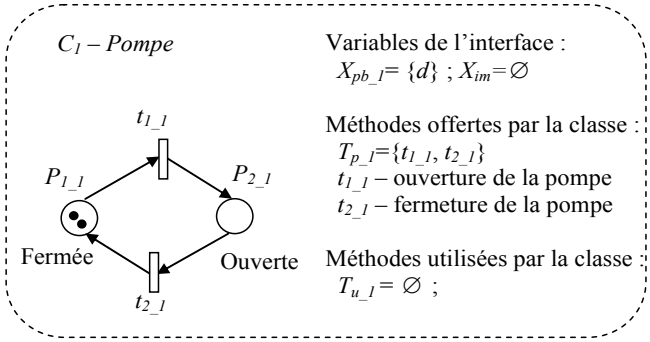
Figure III.6. Interface des classes

Un appel à méthode *tb_k →ta_ln1.i* est effectif si la transition représentant l'appel à méthode (transition *tb_k*) ainsi que celle représentant l'offre de méthode (transition *ta_i*) sont sensibilisées en même temps. Considérons deux objets *Om.k* et *Oj.i*, la transition *tb_k* est sensibilisée dans la classe *Ck* si pour l'objet *Om.k*, le marquage *mm.k* est tel que les places d'entrée de la transition *tb_k* sont marquées et que la fonction de sensibilisation qui lui est associée est vraie. La transition *ta_i* est elle aussi sensibilisée si pour l'objet *Oj.i*, le marquage *mj.ik* est tel que les places d'entrée de la transition *ta_i* sont marquées et que la fonction de sensibilisation qui lui est associée est vraie. L'autre condition pour que l'appel soit effectif (tir des transitions appelant et offrant la méthode) concerne la valeur de *ln* de *Xm.k* est telle que *ln=j.* Dans ce cas l'exécution de l'appel et de l'offre de méthode peut s'effectuer.

Exemple :

Dans l'exemple de la chaudière à vapeur, pour le marquage représenté par la figure III.7, les deux pompes sont fermées et le réservoir est en état de vidange (le débit de vapeur évacuée est supérieur au débit des deux pompes qui sont fermées). Dans ce cas, la transition *t1_1* est sensibilisée pour les deux objets *O1.1* (*Pom1*) et *O1.1* (*Pom2*) dans la classe *C1*. D'autre part, la transition *t1_2* est sensibilisée pour la classe *C2*. La variable *l2* indique quel objet offre la méthode *t1_2 →t1_l2.1*. Comme la variable *l1*est égale à *1*, c'est donc l'objet *O1.1* qui sera utilisé pour le tir de la transition *t1_1*. Les transitions *t1_1* et *t1_2* sont tirées simultanément.

L'exécution d'une méthode (composée de l'appel et de l'offre de méthode) peut être considérée comme un seul événement. Dans ce cas, elle est représentée par la fusion de transitions. Une méthode peut être composée d'une séquence d'événements ou d'une activité continue. Dans le cas où la méthode est composée d'une séquence d'événements, la méthode est modélisée par une double fusion de transitions. La première fusion est l'appel à méthode et la deuxième méthode est la réponse à l'appel (la méthode a été bien exécutée). La méthode s'exécuté entre ces deux fusions.



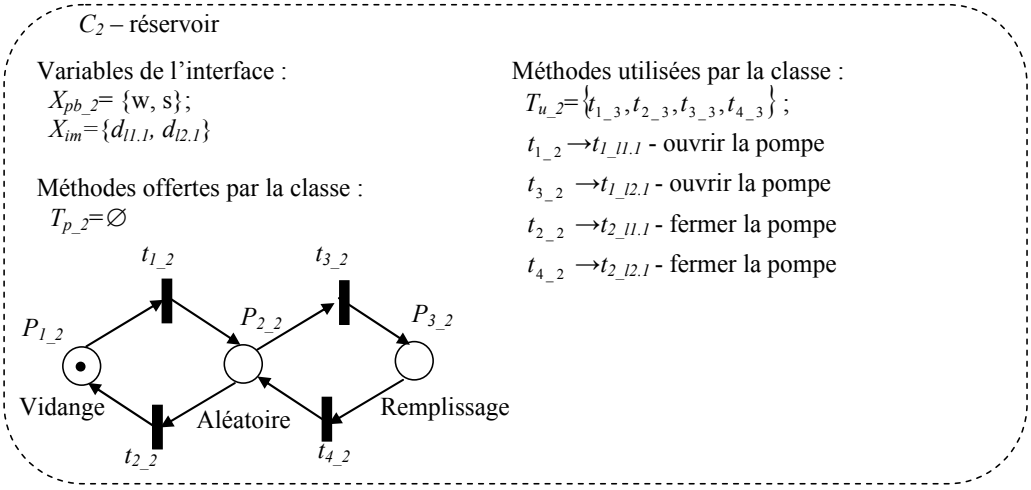


Figure III.7. Offres et appels de méthode

Lors d'un appel à méthode, il est possible de transmettre des données. En effet l'offre de méthode peut s'accompagner d'une transmission de données. Les données sont définies dans la signature de la méthode.

Définition III.13 : Une signature est définie pour chaque transition *tb_k* de l'ensemble *Tu_k* de la classe *Ck*, elle contient un sous ensemble de variables publiques *Xpb_k* transmises lors d'un appel à méthode.

Définition III.14 : Une signature est définie pour chaque transition *ta_i* de l'ensemble *Tp_i* de la classe *Ci*, elle contient un sous-ensemble de variables *Xi* qui recevront les valeurs transmises lors d'appel à méthode.

III.2.3 Communication avec l'environnement externe

Tout comme les réseaux de Petri ordinaires, dans les *RdP PTD-OO*, les interactions avec l'environnement externe ne peuvent être représentées. Les communications doivent être représentées dans le modèle du *RdP* et sont donc une partie du modèle représentant le système.

Dans les *RdP PTD-OO*, les interactions avec l'environnement externe sont spécifiées dans l'interface de l'objet. Les valeurs des variables des classes ainsi que leurs méthodes peuvent être adressées par des entités externes.

Définition III.15 : l'interface entre une classe C_i et son environnement externe est composée de :

- un ensemble de variables externes X_{ext_i}.
- Chaque variable X_{ext_i} est associée à une variable X_i appelée E_n, où n est un entier. E_n spécifie le signal d'entrée qui détermine les valeurs des variables. Les variables de X_{ext_i} sont appelées M_{En}.
- Un ensemble de transitions T_{ext_i} qui représente les méthodes offertes par la classe à son environnement externe.

La définition des variables externes est surtout nécessaire lors de la simulation d'un *RdP PTD-OO*. Pour la vérification des systèmes ou leur analyse de fiabilité, les bonnes propriétés doivent être vérifiées indépendamment de l'environnement externe du système.

III.3 RdP Prédicats Transitions Différentiels Stochastique OO

Dans une démarche d'analyse de sûreté de fonctionnement d'un système, les aspects stochastiques relatifs aux mécanismes de défaillance et de réparation doivent pris en compte. Dans son travail de thèse [Khalfaoui, 2003] a introduit les réseaux de *Petri Prédicats Transitions Différentiels Stochastiques* (RdP PTDS). Dans la partie qui va suivre nous allons introduire les réseaux de *Petri Prédicats Transitions Différentiels stochastiques Orientés Objet*. Ce modèle se prête bien à l'analyse de sûreté de fonctionnement des systèmes hybrides complexes. Ils sont l'extension des RdP PTD-OO défini dans les sections précédentes. Dans les RdP PTD-OO, une fonction de sensibilisation est associée à chaque transition. Cette fonction de sensibilisation est déterministe. La transition est franchie à la date à laquelle le seuil est atteint. Dans le cas de RdP PTDS-OO, deux types de transitions sont présents ; des transitions déterministes et des transitions stochastiques. Aux transitions déterministes sont associées des fonctions de sensibilisation déterministes et aux transitions stochastiques sont associées des fonctions stochastiques. Ces fonctions stochastiques sont associées à toute transition modélisant un mécanisme de défaillance ou de réparation (occurrence non déterministe d'une défaillance d'un composant ou sa réparation). Des lois de probabilités définissant les processus stochastiques de défaillance et de réparation sont donc associées aux transitions du RdP PTDS - OO. La loi de probabilité peut être quelconque, mais les plus utilisées en sûreté de fonctionnement les lois : uniforme, exponentielle, Weibull.

Nous allons donner la définition d'un sous-réseau *RdP PTDS-OO*, sachant qu'un réseau *RdP PTDS-OO* est composé de sous-réseau *RdP PTDS-OO*.

Définition III.16 : *(sous-réseau RdP PTDS-OO)* : chaque sous-réseau *RDP PTDS-OO* est un 3-uplet, $C_i =< R_i, A_i, M_{0\_i} >$, où :

- R_i est un réseau de Petri défini par le 4-uplet $< P_i, T_i, \Pr e_i, Post_i >$, où :
  - $P_i = \{P_{1\_i}, P_{2\_i}, \ldots, P_{m\_i}\}$ est un ensemble fini de place.
  - $T_i = \{T_{1\_i}, T_{2\_i}, \ldots, T_{n\_i}\}$ est un ensemble fini de transitions. Les transitions du réseau sont réparties en deux sous ensembles disjoints : T_{iS} et T_{iD} tel que $T_i = T_{is} \cup T_{id}$ et $T_{is} \cap T_{id} = \varnothing$. T_{iS} est l'ensemble des transitions stochastiques et T_{iD} est l'ensemble des transitions déterministes.
  - $P_i \cap T_i = \{\ \}$, $P_i \cup T_i \neq \{\ \}$
  - $\Pr e_i : P_i \times T_i \rightarrow (0,1)$.
  - $Post_i : P_i \times T_i \rightarrow (0,1)$.
- A_i est l'annotation de C_i, A_i=<X_i, A_{pi}, A_{fi}, A_{ei}, A_{ji} >:

- X_i est l'ensemble des variables.
- A_{pi} est une application qui associe à place P_{k_i} un vecteur de variable X_{pk_i} de l'ensemble des variables X_i.
- A_{fi} est une application qui associe à chaque place P_{k_i} un système d'équations différentielles et/ou algébriques F_{k_i}.
- A_{ei} est une application qui associe à chaque transition $t_{k\_id} \in T_{id}$ une fonction de sensibilisation e_{k_id} et à chaque transition $t_{k\_is} \in T_{is}$ une fonction stochastique. La fonction de sensibilisation utilise les variables A_{pi} qui sont associées aux places d'entrée de la transition. La fonction stochastique peut être immédiate ou temporisée. La fonction stochastique temporisée est décrite par un couple <*I, D*>, où *I* est un intervalle de tir associé à la transition stochastique temporisé et *D* est une densité de probabilité associée à cette même transition.
- A_{ji} est une application qui associe à chaque transition $t_{k\_id} \in T_{id}$ une fonction de jonction j_{k_id} et à chaque transition stochastique $t_{k\_is} \in T_{is}$ une fonction de jonction j_{k_is}. Une fonction de jonction est activée lors du franchissement de la transition correspondante. Les fonctions de jonction associées aux transitions stochastiques permettent une représentation simple des défaillances qui se traduisent par l'affectation de certaines valeurs aux variables continues.

- M_O est le marquage initial du sous-réseau *RdP PTD_OO*.

Exemple :

Reprenons l'exemple de la chaudière à vapeur. Si on considère maintenant les défaillances possibles des pompes ainsi que leur réparation (figure III.8), nous obtenons le nouveau modèle réseau de *Petri Prédicats Transitions différentiel Stochastique Orienté Objets* de la classe pompe. La transition *defo_1* représente la défaillance de la pompe quand elle est dans l'état « ouverte ». La transition *deff_1* représente la défaillance de la pompe quand elle est dans l'état « fermée ». A ces deux transitions est associée une loi de probabilité uniforme sur l'intervalle [2, 6]. Les transitions *repo_1*et *repf_1* modélisent le processus de réparation de pompe. Ces deux transitions peuvent être stochastiques comme elles peuvent être déterministes.

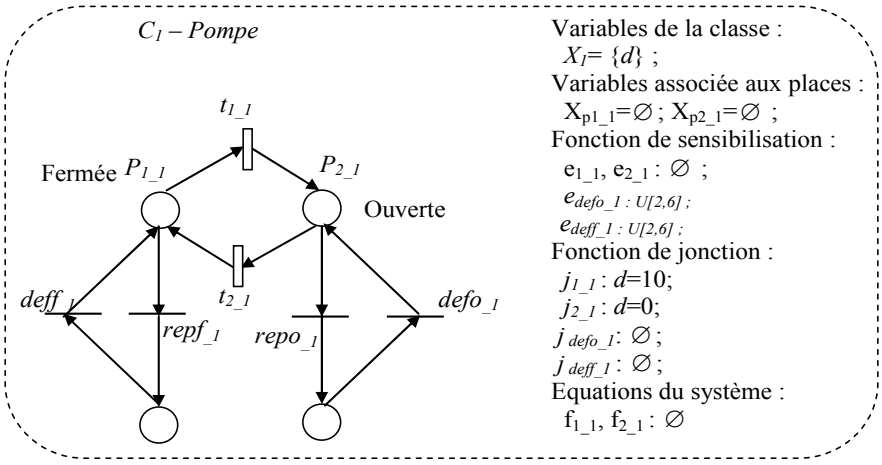


Figure III.8. Modèle de la classe pompe avec prise en compte des défaillances

IV Construction d'un modèle pour la fiabilité dynamique

Maintenant que nous avons fait le choix du modèle, en l'occurrence Les *RdP PTDS–OO* qui, comme nous l'avons vu, permettent de prendre en compte l'aspect hybride des systèmes embarqués, dans la partie qui va suivre nous allons présenter une méthodologie d'aide à la construction de modèles pour la fiabilité dynamique.

L'approche part d'une description *UML* du système et notamment les différents diagrammes *UML*, puis étape par étape le modèle *RdP PTDS - OO* est construit.

Avant de présenter l'approche, un bref rappel des notions relatives à *UML* s'impose.

IV.1 Langage UML

Parmi les langages utilisés pour la modélisation orientée objet, *Unified Modeling Language* (*UML*) est sans doute le plus connu. C'est un standard adopté par le milieu universitaire mais aussi par le milieu industriel. *UML* [OMG, 1997] a été défini par l'OMG (Object Management Group) en novembre 1997 comme la notation normalisée des méthodes OO. C'est une notation générique pouvant s'adapter à n'importe quelle méthode *OO* et pour n'importe quel type d'application. Différentes versions d'*UML* ont été depuis publiées. Nous n'allons pas nous attarder sur le langage *UML*. De nombreux ouvrages expliquent la notation *UML* [Booch, 1998] [Muller, 1997]. Nous nous contentons de rappeler uniquement les concepts utiles à dans notre démarche.

UML est un langage, qui à partir, d'un ensemble d'éléments et de relations entre ces éléments détermine un ensemble de diagrammes qui fournissent une représentation de différentes vues d'un même système. Les éléments de base du langage *UML* sont les structures telles les classes, les interfaces et les composants et les éléments relatifs au comportement du système comme les états, les activités et les interactions.

UML contient 9 diagrammes (13 diagrammes pour *UML* 2.0). Parmi ces 9 diagrammes, 4 sont utilisés dans notre démarche :

Les diagrammes de cas d'utilisation sont des diagrammes utilisés pour donner une vision globale du comportement fonctionnel d'un système. Un cas d'utilisation représente une unité discrète d'interaction entre des utilisateurs (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés *acteurs*, ils interagissent avec les cas d'utilisation.

Les diagrammes d'interaction sont des diagrammes qui décrivent les échanges de messages entre les différents objets pour un scénario donné. Un scénario est une séquence d'activités et d'opérations décrivant un cas d'utilisation. Deux types de diagramme d'interaction existent : le diagramme de séquences qui présente l'ordre temporel des échanges de messages entre objets et le diagramme de collaboration qui présente les messages reçus et envoyés par les différents objets.

Le diagramme de classes est un schéma utilisé pour présenter les classes et les interfaces d'un système ainsi que les différentes relations entre celles-ci. Ce diagramme appartient à la partie statique d'*UML* car il fait abstraction des aspects temporels et dynamiques. Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe. Une classe est un ensemble de fonctions et de données (attributs) qui sont liées par un champ sémantique.

IV.2 Principe de l'approche

L'approche est inspirée des travaux [Ouardani & al, 2006], [Esteban & al, 2005] et ceux de [Villani, & al 2004]. Elle est composée de deux étapes, une première étape consiste à déterminer le modèle fonctionnel du système à partir du cahier des charges en s'appuyant sur l'analyse fonctionnelle du système et les différents diagrammes *UML*. Un modèle réseau de *Petri Prédicats Transitions Différentiel Orienté Objets* est alors construit. La deuxième étape, permet de construire le modèle réseau de Petri dysfonctionnel du système. En prenant en compte les exigences de sûreté de fonctionnement, suite à une analyse préliminaire des risques, les mécanismes de défaillance et de réparation des différents composants du système, mais aussi les différents états redoutés sont recensés et modélisés.

IV.2.1 Etape 1 : construction du modèle fonctionnel

Analyse fonctionnelle

Pour analyser un système, il est nécessaire auparavant de bien identifier à quoi doit servir ce système : c'est à dire de bien identifier toutes les fonctions que ce système doit remplir durant sa vie de fonctionnement et de stockage. L'analyse fonctionnelle nous permet donc de spécifier les cas d'utilisation effectué à l'étape 1.2. Cette étape est étroitement liée à l'exigence de représentation de la solution logique des processus de la norme *EIA-632*.

Spécification des cas d'utilisation

Dans cette étape, les diagrammes des cas d'utilisation *UML* sont déterminés. Un diagramme de cas d'utilisation permet d'avoir une vue du système en ainsi que les différents acteurs. Les

acteurs sont les utilisateurs du système, le système de commande, etc… Il faut noter que dans notre cas le système de commande fait partie du modèle. Donc le modèle contiendra les objets commandés et le système de commande.

Construction du diagramme d'activité

Un diagramme d'activité est construit pour chaque cas d'utilisation. Le diagramme d'activité montre la séquence d'activités produite pour accomplir le cas d'utilisation. Il illustre la synchronisation, le parallélisme, et la concurrence.

L'utilisation des réseaux de Petri nous permet de se passer de l'étape de construction du diagramme d'activité. En effet, le réseau de Petri peut être vu comme un diagramme d'activité.

Spécification des classes et des objets

Dans cette étape, l'ensemble des objets qui composent le système est spécifié. Ces objets sont regroupés en classes. En règle générale, les candidats principaux pour la spécification sont les ressources et les composants du système.

Un critère qui peut être utilisé pour la spécification des objets est le degré d'interaction. Chaque objet doit avoir une forte cohésion interne et une faible interaction avec les autres objets [Paludetto, 1991]. Afin d'obtenir un ensemble d'objets avec un niveau de décomposition adéquat, trois règles de décomposition ont été définies [Villani & al, 2004]. Chaque règle limite les interactions qui peuvent exister entre les objets.

La première règle est appliquée et si les objets résultant sont toujours complexes, la première règle est remplacée par la deuxième et certains objets peuvent ainsi être décomposés en objets plus simples. Si la complexité des objets persiste après l'application de la deuxième règle, la troisième est alors appliquée.

1er règle de décomposition : les objets résultant de la décomposition du système peuvent avoir uniquement des interactions discrètes entre eux. Cela signifie que les sous *RdP PTD – OO* ne partagent pas de variables continues entre eux. Uniquement les interactions discrètes (appels et offres de méthodes) sont permises. Ceci dit, les appels et offres de méthodes doivent être minimisés.

L'application de la première règle peut avoir comme conséquence des classes complexes, un compromis entre la modularité et l'autonomie doit être trouvé. Ainsi, la deuxième règle est employée pour la décomposition.

2ème règle de décomposition : les objets résultant de la décomposition peuvent avoir des interactions continues entre eux, mais les interactions continues sont restreintes à des intervalles limités dans le temps.

Si l'application de cette nouvelle règle n'est pas suffisante pour décomposer le système en ensemble de classes simples, la troisième et dernière règle peut alors être appliquée.

3ème règle de décomposition : les objets résultant de la décomposition peuvent avoir des interactions continues sans restriction sur l'intervalle de temps, mais ne peuvent partager les

variables continues en boucle fermée (l'objet 1 utilise une variable de sortie de l'objet 2 qui lui utilise une variable de l'objet 1).

La restriction imposée par la 3ème règle garantie un minimum d'indépendance pour les objets. Quand un ensemble d'objets partage des variables continues en boucle fermée, cela signifie que les équations qui définissent la dynamique de chaque objet de la boucle doivent être fusionnées avec les équations des autres objets et l'ensemble d'équations est résolu comme un système d'équation unique. Les évolutions des objets sont si étroitement liées qu'aucune décomposition n'est possible.

Si la décomposition proposée pour le système ne répond pas à la troisième règle, la décomposition doit être revue.

Une fois la décomposition effectuée, les objets doivent être regroupés en classe. Les objets qui possèdent un même comportement sont regroupés dans la même classe.

Construction du diagramme de collaboration

Le diagramme d'activité établi le modèle dynamique du système sans indiquer quel objet doit exécuter chaque activité. Une fois les objets et les classes du système déterminés, la construction du diagramme de collaboration permet d'indiquer les interactions entre les objets qui sont nécessaires pour exécuter les activités d'un cas d'utilisation. Le diagramme de collaboration met en évidence les appels de méthode de chaque objet et par conséquence les offres de méthode. Il est important de se soumettre à la contrainte que chaque diagramme montre seulement un scénario, c'est-à-dire, une des évolutions possibles pour le système. Dans le cas de la simultanéité, plus d'un diagramme doit être produit pour chaque cas d'utilisation afin de représenter toutes les évolutions possibles.

Une fois que les diagrammes de collaboration ont été établis et que les communications entre objets ont été identifiées, les méthodes et les attributs des classes doivent être spécifiés. D'éventuelles nouvelles méthodes et nouveaux attributs peuvent être rajoutés à la liste une fois le sous *RdP PTD - OO* établi.

Construction du RdP PTD – OO des classes et des diagrammes de classe

Le *RdP PTD – OO* des classes doit être conforme aux diagrammes établis dans les étapes précédentes. A fin de garantir la conformité du modèle les considérations suivantes doivent être prises en compte :

- A partir de l'analyse fonctionnelle, chaque activité est associée à un objet. Les activités continues correspondent aux places du RdP PTD – OO, tandis que les activités discrètes correspondent aux transitions.
- Les communications des diagrammes de collaboration correspondent aux appels et offres de méthodes dans le cas des communications discrètes et au partage des variables continues dans le cas des communications continues.

La définition des sous- *RdP PTD – OO* revient à construire le diagramme de classe *UML*, où les relations entre classes sont illustrées.

Une fois les sous *RdP PTD – OO* construits, il est nécessaire de vérifier la consistance du modèle construit. Dans ce manuscrit nous donnons uniquement les grandes lignes et étapes de vérification de la consistance du modèle. Le lecteur pourra trouver d'avantage de détails dans [Villani & al, 2007].

Vérification de la consistance du modèle :

Pour que le *RdP PTD – OO* et les diagrammes de collaboration soient consistants, les propositions suivantes doivent être vraies :

Chaque communication discrète doit correspondre à un appel de méthode associé à une transition.

Chaque communication continue doit correspondre à une variable partagée entre les objets.

Deux relations entre classes sont particulièrement importantes pour la conception de système et la réutilisation : la composition/décomposition et spécialisation /généralisation [Booch, 1994].

Dans la relation de composition/décomposition un objet d'une classe fait partie d'un objet d'une autre classe. Dans les *RdP PTD-OO*, la composition est réalisée par la fusion des transitions qui correspondent aux appels à méthodes entre les classes. D'autre part, la décomposition peut être faite en identifiant des invariants de places et la division du sous-réseau en autant d'invariants.

La relation de composition/décomposition est utile pour la définition des classes avec un niveau de détail adéquat. Elle est également importante pour la définition des classes complexes qui ne peuvent pas être divisées en raison des boucles fermées dans le partage des variables continues.

Une autre relation entre deux classes est la hiérarchie, notamment la généralisation/spécialisation, dans laquelle une classe est un cas spécial des autres. Les classes *enfant* héritent des méthodes et les attributs de la classe *parent* avec éventuellement des méthodes et/ou des attributs additionnels. Dans le cas de *RdP PTD–OO*, une classe est considérée comme la spécialisation d'une autre classe si elle a les mêmes éléments (place, transitions, variables, etc…) avec des éléments additionnels.

IV.2.2 Etape 2 : modèle dysfonctionnel et états redoutés

Une fois le modèle fonctionnel du système établi (étape 1), pour pouvoir mener des études de sûreté de fonctionnement la construction du modèle dysfonctionnel est nécessaire. C'est l'objectif de la deuxième étape.

Dans la première étape de construction de modèle, nous nous sommes intéressés aux objets en prenant en compte leur dynamique. Dans cette partie, chaque objet est analysé afin de déterminer les défaillances qui lui sont associées. Ces défaillances sont ainsi introduites dans le modèle.

Recensement des défaillances

Les composants du système qui sont maintenant représentés par les objets et modélisés par des sous RdP PTD – OO sont sujets à défaillance. Afin de réaliser une étude de fiabilité, ses défaillances et réparations doivent être prises en compte. Une fois recensées, elles sont

introduites dans le modèle RdP PTD–OO. Les défaillances et les réparations sont généralement modélisées par des transitions stochastiques.

Recensement des évènements redoutés

Le recensement de ces événements redoutés se fait habituellement à l'aide d'une analyse préliminaire des risques (APR). Les situations critiques dont on cherche à déterminer les scénarios pouvant mener à ce genre de situations à partir d'un état de fonctionnement normal doivent être recensées.

Une fois les événements redoutés recensés, ils sont ensuite introduits dans le modèle. Dans le modèle réseau de Petri du système, ces états redoutés seront modélisés par une place. Généralement les places représentant les états redoutés sont des places puits. Aucune évolution n'est possible à partir de ces états.

V Conclusion

Dans ce chapitre nous avons vu différentes approches et différents modèles pour la modélisation des systèmes dynamiques hybrides. Notre choix s'est porté sur les réseaux de *Petri Prédicats Transitions différentiels.* Modèle qui combine deux formalismes distincts ; les réseaux de Petri pour la modélisation des aspects discrets et les équations différentielles pour la partie continue.

Afin de faciliter la modélisation et l'analyse des systèmes complexes, le paradigme orienté objets a été introduit dans les RdP PTD. Le modèle RdP PTD -OO qui en résulte, grâce à sa structure modulaire fournit une flexibilité dans la modélisation mais aussi dans l'analyse des systèmes complexes. En effet, le RdP PDT-OO d'un système complexe est facilement établi suite à sa décomposition en classes et objets.

Un des avantages de l'approche objets est la réutilisation des modèles. Lors de l'analyse de sûreté de fonctionnement le concepteur aura à tester plusieurs configurations, plusieurs architectures. Grâce à la réutilisabilité offerte par l'approche objets, en cas de modification d'architecture, certains composants du modèle initial pourront être réutilisés, le modèle initial ne sera pas complètement remis en cause.

Comme nous l'avons cité auparavant, l'approche de modélisation basée sur les réseaux de Petri et un ensemble d'équations différentielles, permet de séparer clairement les aspects discret et continu. Une analyse logique de la partie discrète est donc possible. Elle est basée sur la logique linéaire de Girard. Le chapitre suivant, présente la cadre formel qui est la logique linéaire, la traduction du réseau de Petri en séquent de logique linéaire qui nous permettra ensuite de formaliser la notion de scénario ainsi que ses propriétés.

Chapitre 4 : Définition et propriétés des scénarios avec la logique linéaire

I Introduction

Les études de sûreté de fonctionnement des systèmes embarqués commencent par une analyse qualitative. L'analyse qualitative vise à mettre en évidence tous les types de comportement amenant à des états pour lesquels la sécurité des utilisateurs n'est plus assurée. Elle a pour but de déterminer les scénarios redoutés et de les caractériser en termes d'actions et de changements d'états.

Comme nous l'avons vu dans le chapitre 1, les approches basées sur les réseaux de Petri consistant à explorer le graphe d'accessibilité permettent de trouver l'ensemble des chemins menant vers l'état redouté en question [Moncelet, 1998]. L'inconvénient de ces approches est qu'elles se heurtent au problème de l'explosion combinatoire du nombre d'états en raison de la complexité des systèmes embarqués. Au-delà du problème de l'explosion combinatoire, les scénarios obtenus ne sont pas minimaux car ils sont entrelacés avec des événements qui n'ont aucune relation de cause à effet avec l'accessibilité de l'état partiel redouté. En effet, tous les événements se produisant en parallèle avec le scénario redouté, par exemple dans des éléments du système non concernés par l'état partiel redouté, vont apparaître dans le scénario obtenu. Pourtant, seuls les scénarios minimaux contribuent à une bonne compréhension des conséquences qualitatives des choix de conception. De plus, l'évaluation quantitative de la sûreté de fonctionnement doit, elle aussi, reposer sur des scénarios minimaux si l'on veut être efficace.

Un moyen de contourner le problème de l'explosion combinatoire est d'utiliser directement le modèle réseau de Petri pour établir les scénarios possibles plutôt que d'utiliser le graphe d'accessibilité associé qui contient bon nombre d'informations sans intérêt du point de vue sûreté de fonctionnement car décrivant des comportements sans défaillance. La logique linéaire [Girard, 1987] offre un cadre théorique permettant d'interpréter les modèles réseaux de Petri et

d'en extraire des scénarios [Khalfaoui, 2003]. Le point clé de cette approche est l'équivalence entre l'accessibilité dans le réseau de Petri et la prouvabilité du séquent associé en logique linéaire.

Dans ce chapitre nous proposons une définition formelle, basée sur la logique linéaire, de la notion de scénario. Cette définition formelle nous permet ensuite de définir formellement la notion de minimalité. Le but final de notre approche est de déterminer les scénarios minimaux, sachant qu'un scénario minimal représente une classe de scénarios.

Pour que la détermination des scénarios redoutés soit efficace, en plus d'avoir uniquement des scénarios minimaux, il est nécessaire de générer tous les scénarios pouvant mener vers un état redouté. La complétude des scénarios est donc une autre exigence. A partir de la définition du scénario minimal, la définition de l'ensemble d'événements complet est proposée en fin de chapitre.

II Réseau de Petri et logique linéaire

La logique linéaire (*LL*) proposée par J.Y. Girard [Girard, 1987] est une restriction de la logique classique (propositionnelle) qui permet de prendre en compte la notion de ressource. De nouveaux connecteurs sont introduits afin de distinguer la présence d'une ressource unique p (exprimée par la proposition p) de la présence d'une copie double de la même ressource (proposition $p \otimes p$) [Girard, 1987]. Le calcul de séquents associé à la logique linéaire est basé sur un ensemble de nouveaux connecteurs et règles. La principale différence avec la logique classique est l'absence des règles de contraction et d'affaiblissement (les formules $a \wedge b \Rightarrow a$ et $a \Rightarrow a \wedge a$ ne sont plus vraies en logique linéaire). Ces règles sont justement supprimées pour prendre en compte la notion de ressources à cause de l'équivalence en logique classique des propositions « $p \wedge p$ » et la proposition p. Les propositions logiques sont considérées, non plus comme des vérités pérennes, mais comme des ressources qui sont consommées et produites pendant le processus d'établissement des preuves.

II.1Règle de transformation

Plusieurs connecteurs ont été introduits en Logique Linéaire (*LL*). Dans le travail qui nous intéresse et qui concerne la traduction des réseaux de Petri en logique linéaire, le fragment *MILL* (*Multiplicative Intuitionnist Linear Logic*) contient les connecteurs nécessaires [Rivière, 2003], [Khalfaoui, 2003].

Ce fragment comprend le connecteur multiplicatif « *Fois*» et le connecteur « *implication linéaire* ». Il n'y a pas de négation et le méta-connecteur «, » est commutatif. Le lecteur intéressé pourra trouver une présentation détaillée des autres connecteurs dans [Girault, 1997].

Le connecteur Fois $\otimes$ est utilisé pour représenter l'accumulation de ressources (la formule $a \otimes b \otimes b$ exprime la disponibilité d'une copie de la ressource a et deux copies de la ressource b) et l'implication linéaire (représentée par le symbole —*o*) permet de prendre en compte la production et la consommation des ressources. Par exemple, la formule 'a—*o*b' traduit la

consommation de la ressource 'a' pour produire la ressource 'b'. La translation d'un réseau de Petri en logique linéaire a été présentée dans [Pradin-Chézalviel & al, 1999]. Pour un réseau de Petri donné, la translation se fait de la manière suivante :

- Une proposition atomique *P* est associée à chaque place *P* du réseau de Petri.
- Un monôme, utilisant la conjonction multiplicative ⊗ (FOIS), est associé à chaque marquage, Pre-condition *Pre ( )* et Post-condition *Post ( )* des transitions.
- Pour chaque transition *t* du réseau de Petri, une formule implicative est définie sous la forme :

$$t : \bigotimes_{i \in \Pr e(pi,t)} P_i \multimap \bigotimes_{o \in Post(po,t)} P_o$$

Tout séquent de la forme $M, t_1, \ldots, t_p \vdash M'$ exprime l'accessibilité entre les marquages *M* et *M'* et inclut les transitions tirées $(t_1, \ldots, t_p)$. La preuve du séquent est déduite de manière canonique [Rivière, 2003]. Le séquent spécifie quelles sont les transitions franchies.

II.2Preuve d'un séquent

La preuve d'un séquent consiste à montrer qu'il est syntaxiquement correct, c'est-à-dire entièrement construit à partir d'un ensemble de règles introduisant les atomes (propositions) et les connecteurs.

Une preuve de séquent de logique linéaire est conduite de bas en haut en posant comme racine de l'arbre de preuve le séquent à prouver. On applique ensuite une à une les règles pour éliminer successivement les connecteurs du séquent. Chaque nœud de l'arborescence est alors un séquent prémisse plus simple que celui dont il est conclusion. Ce séquent est prouvé si chaque feuille de l'arbre se termine par un séquent axiome (dans le sous-ensemble MILL, le seul axiome est le séquent identité). On dit qu'un séquent est prouvable si et seulement si il existe une preuve dont il est la racine.

La preuve est menée en introduisant le connecteur « *Fois* » à gauche de l'arbre de preuve $(\otimes_L)$. L'introduction du connecteur *Fois* permet de rendre les ressources indépendantes (appliquer cette règle sur la formule $a \otimes b$ permet de rendre les propositions '*a*' et '*b*' indépendantes). Cela veut dire qu'elles peuvent être consommées indépendamment l'une de l'autre. En appliquant la règle $\multimap_L$, il est alors possible d'extraire les relations de causalités entre les différents atomes entre les marquages *M* et *M'*.

II.3Equivalence entre RdP et logique linéaire

Un marquage est un monôme, noté $A_1 \otimes A_2 \otimes, \ldots A_k$, où les A_i sont les noms des différentes places : si une place A_i contient plusieurs jetons (par exemple *n*), *n* instances de la proposition A_i apparaissent.

Une transition est une formule $M_1 \multimap M_2$, avec M_1 et M_2 deux marquages (fonction *Pre* et *Post* de la transition). L'expression $M_1 \multimap M_2$ représente le tir de la transition et apparaît dans le séquent autant de fois que la transition est tirée.

Un scénario peut être représenté par un séquent de logique linéaire. Le marquage initial et l'ensemble des franchissements de transition constituent la prémisse du séquent, alors que le marquage final représente la conclusion. La preuve du séquent est équivalente à l'accessibilité du marquage final à partir du marquage initial.

Le séquent $M, \sigma \vdash M'$ représente un scénario avec $\sigma = t_1, t_2, \ldots t_n$ une liste ordonnée des différentes instances de tirs des transitions concernées. M et M' sont respectivement le marquage initial et le marquage final

La construction de la preuve génère un arbre de preuve qui démarre avec le séquent et qui se termine par les axiomes identités. Il est alors possible d'extraire des informations sur l'ordre de tir des transitions à partir de l'arbre de preuve du séquent [Rivière, 2003].

II.4Preuve de séquent et arbre de preuve annoté

La preuve d'accessibilité dans un modèle réseau de Petri est équivalente à la preuve du séquent correspondant. L'objectif d'une preuve de séquent est de déterminer l'ordre partiel entre les transitions tirées (l'ensemble de transitions λ_0). La construction d'un arbre de preuve de séquent est une démarche itérative qui consiste à éliminer à chaque étape chaque transition tirée après vérification que les atomes nécessaires à son tir sont disponibles (produits). Cette étape doit être exécutée une fois à chaque tir (franchissement d'une transition). Elle permet de déterminer les relations de précédence imposées par la structure et le marquage du réseau de Petri. Pour chaque atome, l'application de l'étape itérative détermine la transition qui a produit l'atome est celle qui l'a consommé. Ces relations de précédence sont déterminées grâce à l'arbre de preuve annoté [Rivière, 2003] et définissent formellement un scénario. La preuve d'un séquent donne un arbre de preuve pour chaque séquence de tir de transitions. En prouvant un séquent, *m* arbres de preuve peuvent être déduits. L'annotation de l'arbre de preuve peut produire *m* graphes de précédence.

Exemple *(illustration de l'arbre de preuve annoté)*

Soit le réseau de Petri de la figure IV.1. La figure IV.2 et la figure IV.3 montrent la construction de l'arbre de preuve, pour le séquent $P_1 \otimes P_2,\ t_a \otimes t_b \otimes t_c \vdash P_3 \otimes P_4$. Ces deux arbres correspondent à un franchissement particulier de la séquence (*a, b, c*). Chaque arbre de preuve induit des relations de précédence différentes. Dans l'arbre de la figure IV.2, le tir de la transition *a* précède le tir de la transition *c*. En effet c'est le jeton produit dans la place P_3 par le tir de la transition *a* qui a été consommé lors du tir de la transition *c*, d'où une relation de causalité (précédence) entre les événements associés au tir des transitions *a* et *c*. Dans le deuxième arbre de preuve (figure IV.3), c'est le jeton produit par la transition *b* qui a servi au franchissement de la transition *c*.

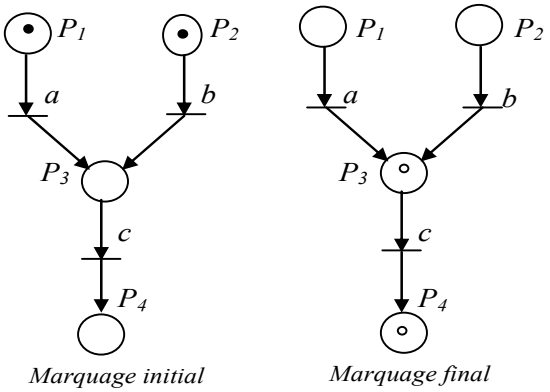


Figure IV.1. Modèle réseau de Petri

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P_1(I^1)\vdash P_1(a^1)}id \quad \dfrac{}{P_2(I^2)\vdash P_2(b^1)}id}{P_1(I^1),P_2(I^2),\ P_1\multimap P_3,P_2\multimap P_3\vdash P_3(a^1),P_2,P_2\multimap P_3,P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\otimes_L}{P_1(I^1),P_2(I^2),\ P_1\multimap P_3,P_2\multimap P_3,P_3\multimap P_4\vdash P_3(a^1),P_3(b^1),P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\multimap\ a^1}{P_3(a^1),P_3(b^1),P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\multimap\ b^1}{P_3(b^1),P_4(c^1)\vdash P_3(F^1)\otimes P_4(F^2)}\otimes_R \quad \dfrac{}{P_3(a^1)\vdash P_3(C^1)}id \quad \dfrac{}{P_4(c^1)\vdash P_4(F^2)}id}{P_3(a^1)\vdash P_3(F^1)}id}{P_3(a^1),P_3(b^1),P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\multimap\ c^1$$

Figure IV.2. 1er arbre de preuve

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P_1(I^1)\vdash P_1(a^1)}id \quad \dfrac{}{P_2(I^2)\vdash P_2(b^1)}id}{P_1(I^1),P_2(I^2),\ P_1\multimap P_3,P_2\multimap P_3\vdash P_3(a^1),P_2,P_2\multimap P_3,P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\otimes_L}{P_1(I^1),P_2(I^2),\ P_1\multimap P_3,P_2\multimap P_3,P_3\multimap P_4\vdash P_3(a^1),P_3(b^1),P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\multimap\ a^1}{P_3(a^1),P_3(b^1),P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\multimap\ b^1}{P_3(b^1),P_4(c^1)\vdash P_3(F^1)\otimes P_4(F^2)}\otimes_R \quad \dfrac{}{P_3(b^1)\vdash P_3(C^1)}id \quad \dfrac{}{P_4(c^1)\vdash P_4(F^2)}id}{P_3(a^1)\vdash P_3(F^1)}id}{P_3(a^1),P_3(b^1),P_3\multimap P_4\vdash P_3(F^1)\otimes P_4(F^2)}\multimap\ c^1$$

Figure IV.3. 2ème arbre de preuve

III Notion de scénario

Maintenant que nous avons vu la traduction des réseaux de Petri en logique linéaire, dans cette section, nous allons présenter la notion d'événement et de scénario.

III.1 Evénements et ensembles d'événements

Définition IV.1. (*Evénement*) : Soit un réseau de Petri (P, T, Pre, Post) et soit M_0 son marquage initial. Un événement est un franchissement particulier d'une transition $t \in T$.

L'ensemble des événements est noté *E*. Par exemple, si lors d'une évolution du réseau de Petri à partir de M_0 la transition ti est franchie pour la $j^{ème}$ fois, nous dirons que c'est l'occurrence de l'événement e_i^j . L'ensemble des événements E est donc potentiellement infini, mais il est construit à partir de l'ensemble fini T des transitions. Les événements e_i^j pour tout $j \in [1,\ldots n]$ correspondent tous aux franchissements de la même transition ti. Les exposants sont arbitraires et ne font pas référence à l'exécution d'une séquence particulière. Ce sont des variables formelles qui ne servent qu'à différentier les divers franchissements d'une même transition.

Définition IV.2. (*Ensemble d'événements*) : Tout sous-ensemble $l \subset E$ est un ensemble d'événements.

Un ensemble d'événements *l* peut être considéré comme un multi-ensemble construit sur *T*, ou encore comme un vecteur d'entiers positifs ou nuls ayant pour dimension celle de *T*, c'est-à-dire le nombre de transitions du réseau de Petri.

On peut également écrire les multi-ensembles *l* sous la forme de monômes en $\otimes$ (ce connecteur est commutatif et non idempotent) de formules implicatives en logique linéaire.

En ce qui concerne les notations, par analogie avec le vecteur caractéristique d'une séquence de franchissements d'un réseau de Petri, nous noterons $\bar{l}$ le vecteur mentionné ci-dessus. Par rapport à la vision multi-ensemble, ses composantes sont les degrés d'occurrence de *t* dans le multi-ensemble *l*. Si *l* comprend *k* éléments e_i^j pour *i* fixé, alors $l(t_i) = k$.

Considérons le réseau de Petri de la figure IV.4. Deux exemples d'ensemble d'événements sont :

$$l_1 = \{e_a^1, e_a^2, e_b^1, e_c^1\} \qquad l_2 = \{e_a^1, e_a^2, e_b^1, e_c^1, e_d^1\}$$

Ces mêmes ensembles, considérés comme des multi-ensembles et écrits sous la forme de monômes en $\otimes$, donnent : $l_1 = t_a \otimes t_a \otimes t_b \otimes t_c$ et $l_2 = t_a \otimes t_a \otimes t_b \otimes t_c \otimes t_d$

t_a représente la formule : $P_1 \otimes P_3 \text{—o} P_2 \otimes P_4$

t_b représente la formule : $P_2 \text{—o} P_1$

t_c représente la formule : $P_4 \otimes P_4 \otimes P_5 \text{—o} P_3 \otimes P_3 \otimes P_6$

t_d représente la formule : $P_6 \text{—o} P_5$

Si maintenant nous considérons les vecteurs, nous aurons par exemple :

$$\bar{l}_1 = \begin{matrix} t_a \\ t_b \\ t_c \\ t_d \end{matrix} \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \bar{l}_2 = \begin{matrix} t_a \\ t_b \\ t_c \\ t_d \end{matrix} \begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

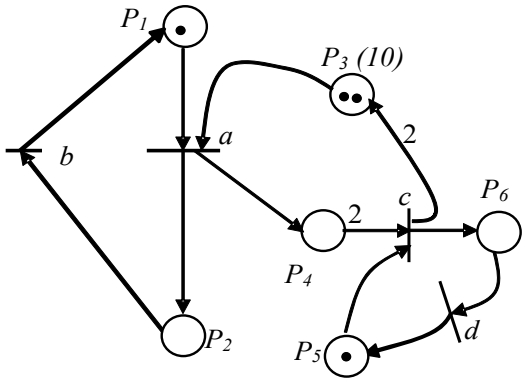


Figure IV.4. Modèle réseau de Petri

Définition IV.3. (*Concaténation d'ensembles d'événements*) : Soient deux ensembles d'événements l_1 et l_2 écrits sous la forme de monômes (multi-ensembles). Le résultat de la concaténation de l_1 et l_2 est l'ensemble d'événements l_3 tel que : $\bar{l}_3 = \bar{l}_1 + \bar{l}_2$

Du point de vue des ensembles, il faut simplement renuméroter tous les exposants associés aux événements de l_2 pour qu'aucun événement de l_2 ne soit identique à un événement de l_1 avant de faire l'union des deux ensembles. Les exposants sont en effet des entiers arbitraires ne servant qu'à distinguer les différentes instances de franchissement d'une même transition.

Toujours pour l'exemple de la figure IV.4 nous aurons :

$$l_3 = l_1 + l_2 = \left\{ e_a^1, e_a^2, e_a^3, e_a^4, e_b^1, e_b^2, e_c^1, e_c^2, e_d^1 \right\}.$$
$$l_3 = t_a \otimes t_a \otimes t_a \otimes t_a \otimes t_b \otimes t_b \otimes t_c \otimes t_c \otimes t_d .$$

III.2 Scénario

La définition d'un scénario est basée sur la notion d'événement et les relations entre les événements. Nous définirons dans ce qui suit, un scénario dans le cas des réseaux de Petri ordinaires mais aussi dans celui des réseaux de Petri temporel.

III.2.1 Cas des réseaux de Petri ordinaires

Considérons un réseau de Petri *P = (P, T, Pre, Post)* avec un marquage initial M_0 et un marquage final M_f et I un ensemble d'événements associés aux créations des jetons du

marquage M_0 (un événement par jeton). Soit F un ensemble d'événements associés aux consommations des jetons du marquage M_f. (Un événement par jeton).

Définition IV.4. (*Scénario*) : Un scénario sc, noté $sc = (l, \prec_{sc})$ associé au réseau de Petri P et au couple de marquages M_0 et M_f, est un ensemble d'événements $l \subset (E \cup I \cup F)$ muni d'un ordre partiel strict $\prec_{sc}$ défini sur les événements de *l*.

Si pour $e_1, e_2 \in l$ nous avons $e_1 \prec e_2$, cela veut dire que l'événement e_1 précède l'événement e_2 dans le scénario *sc*.

Remarquons que les événements sont définis sur un ensemble plus grand que précédemment. Nous avons besoin de la notion d'événements initiaux et finaux pour composer des scénarios complexes à partir de scénarios plus élémentaires.

Quand nous considérerons des séquences de franchissement dans le cadre d'un scénario, il s'agira toujours d'une linéarisation de la restriction du scénario aux événements de *E*.

Considérons à nouveau le réseau de Petri de la figure IV.4 avec les marquages initial $M_0 = P_1 \otimes P_3 \otimes P_3 \otimes P_5$ et final $M_f = P_2 \otimes P_3 \otimes P_3 \otimes P_6$. Cela introduit quatre événements initiaux de type *I* (I_1, I_2, I_3 et I_4) correspondant à la création du jeton dans la place P_1, des deux jetons dans la place P_3 et du jeton dans la place P_5 et quatre événements finaux de type *F* (consommation des jetons dans les places P3, P4 et P6).

Dans notre représentation de scénarios, un ordre partiel peut être représenté par un graphe de précédence (graphe orienté) (*E, A*) où *E* est un ensemble de franchissements de transitions (ti) et A un ensemble de paires (ti, tj) telles que ti précède tj (ti et tj étant des franchissements de transitions). L'ensemble A contient les différents arcs du graphe de précédence auxquels sont associés des atomes qui représentent le jeton consommé par le tir de la transition tj. L'annotation des arcs est tirée de l'arbre de preuve annoté. Le poids sur les arcs correspond aux atomes consommés à chaque franchissement de transition.

Le graphe de précédence de la figure IV.5 représente le scénario *SC* associé au réseau de Petri de la figure IV.4 entre les marquages $M_0 = P_1 \otimes P_3 \otimes P_3 \otimes P_5$ et $M_f = P_2 \otimes P_3 \otimes P_3 \otimes P_6$ tel que :

$SC : P_1 \otimes P_3 \otimes P_3 \otimes P_5,\ t_a \otimes t_a \otimes t_b \otimes t_c \vdash P_2 \otimes P_3 \otimes P_3 \otimes P_6$

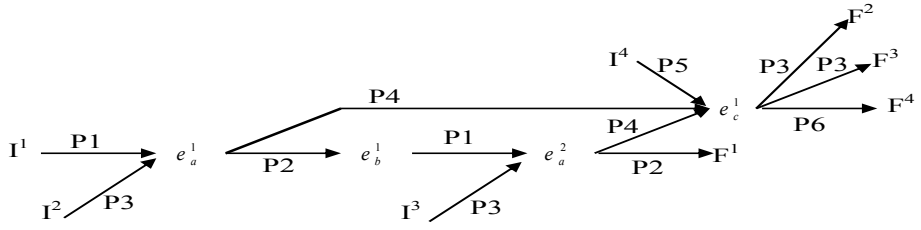


Figure IV.5. Graphe de précédence

III.2.2 Cas des réseaux de Petri temporels

La définition d'un réseau de Petri temporel est donnée dans le chapitre 3 de ce manuscrit, la définition suivante (IV.5) correspond à celle du scénario dans le cas d'un réseau de Petri temporel.

Définition IV.5. (*Scénario*) **:** Un scénario $sc = (l, \prec_{sc})$ associé au réseau de Petri temporel N_{tl} (réseau de Petri t-temporel), au couple de marquages M_0 et M_f et à un ensemble d'événements $l \subset (E \cup I \cup F)$ est un ordre partiel strict $\prec_{sc}$ défini sur les événements de l. L'ordre partiel $\prec_{sc}$ est composé des relations d'ordre dues aux causalités présentes dans le modèle réseaux de Petri (partie discrète) qu'on notera $\prec_{PNsc}$ mais aussi des relations d'ordre générées par les contraintes temporelles qu'on notera $\prec_{tsc}$.

Exemple :

La figure IV.6 représente un fragment de réseau de Petri temporel. A la transition t_1 est associé le seuil temporel $t = \tau_1$ et à la transition t_2 le seuil $t = \tau_2$ avec $\tau_1 < \tau_2$. Si la place P_3 est marquée (P_1 et P_2 étant marquées), la transition t_1 sera franchie avant t_2 puisque le seuil temporel associé à t_1 est inférieur à celui de t_2. Dans ce cas, on ne considère pas le scénario associé au franchissement de t_2. Par contre, si t_3 est déjà franchie (place P_3 vide) et si les places P_1 et P_2 sont marquées, on ne peut plus franchir t_1. t_2 sera alors franchie.

On peut conclure que le franchissement de t_3 est une cause du franchissement de t_2. Nous avons donc une relation de précédence entre le franchissement de la transition t_3 (consommation du jeton de la place P_3) et le franchissement de la transition t_2. Pourtant aucune place ne relie t_3 à t_2.

Cette relation de précédence est donc une conséquence de la dynamique continue (qui peut se traduire en contraintes temporelles) et des seuils temporels associés aux transitions t_1 et t_2. Nous parlerons dans ce cas de relation de précédence indirecte et de causalité indirecte. L'ordre partiel $\prec_{tsc}$ contient toutes ces relations de précédence indirectes. Ce cas est souvent rencontré dans les conflits chiens de garde temporel [Sadou & al, 2006a] (cas de la figure VI.6). Cette notion de conflit chien de garde sera détaillée dans le chapitre 5

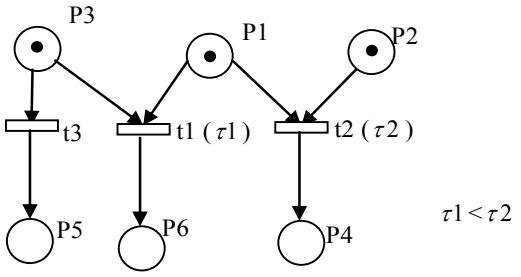


Figure IV.6. Exemple de réseau de Petri temporel

La figure IV.7 montre le graphe de précédence du scénario correspondant au séquent $P_1 \otimes P_2 \otimes P_3,\ t_2, t_3 \vdash P_4 \otimes P_5$ (franchissement de t_3 puis de t_2).

La flèche en pointillés représente la relation de causalité indirecte induite par les contraintes temporelles associées aux transitions t_1, t_2 et t_3.

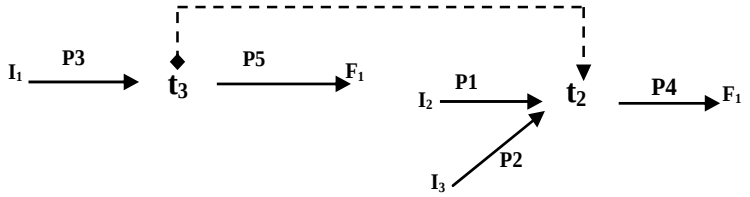


Figure IV.7. Scénario généré à partir du réseau de Petri temporel de la figure IV.6

IV Conditions suffisantes

Avant de définir ce qu'est un ensemble d'événements suffisant et un scénario suffisant, nous allons rappeler la notion d'inclusion d'ordres.

Définition IV.6 (*Inclusion d'ordres*) **:** Soient deux ordres partiels $\prec_1$ et $\prec_2$, tous les deux définis sur le même ensemble d'événements *l*, nous dirons que $\prec_1$ est inclus dans $\prec_2$ si : $\forall e_i, e_j \in l,\ (e_i \prec_1 e_j)$ implique $(e_i \prec_2 e_j)$.

La définition implique que toutes les linéarisations de $\prec_2$ sont des linéarisations de $\prec_1$. Remarquons que le mot inclusion fait référence aux relations de précédences, l'ordre $\prec_1$ a moins de relations de précédences donc il est moins contraint et représente plus de séquences possibles.

Remarque : Le mot *inclusion* fait référence au fait que si l'ordre partiel est exprimé sous la forme d'un graphe et si l'ordre $\prec_1$ est inclus dans $\prec_2$ alors tous les arcs de $\prec_1$ sont des arcs de $\prec_2$. Il y a bien inclusion de l'ensemble des arcs de $\prec_1$ dans celui de $\prec_2$. Par contre, plus il y a d'arcs et plus en fait l'ordre est restrictif, c'est-à-dire qu'il caractérise moins de séquence. L'ordre $\prec_2$ caractérise donc moins de séquences que l'ordre $\prec_1$.

Définition IV.7 (*égalité d'ordres*) **:** Soient deux ordres partiels $\prec_1$ et $\prec_2$ tous les deux définis sur le même ensemble d'événements *l*. Nous dirons que $\prec_1$ est égal à $\prec_2$ $(\prec_1 = \prec_2)$ si :

$$\forall e_i, e_j \in l,\ (e_i \prec_1 e_j) \text{ implique } (e_i \prec_2 e_j).$$
$$\text{et} \quad \forall e_i, e_j \in l,\ (e_i \prec_2 e_j) \text{ implique } (e_i \prec_1 e_j).$$

IV.1 Ensemble suffisant

Considérons un réseau de Petri *(P, T, Pre, Post)* avec un marquage initial M_0 et un marquage final M_f.

Définition IV.8. **(*Ensemble suffisant*) :** L'ensemble d'événements $l \subset E$ est suffisant pour passer de M_0 à M_f si le séquent $M_0, l \vdash M_f$ est prouvable.

Soit A_i, i = 1…n l'ensemble des arbres canoniques de preuve du séquent $M_0, l \vdash M_f$ et soit $\prec_j$, j= 1…m l'ensemble des ordres partiels que l'on peut déduire sur les éléments de l à partir de ces arbres de preuve par annotation. Plusieurs preuves canoniques peuvent donner le même ordre partiel (et donc le même scénario), mais une preuve peut aussi donner plusieurs ordres partiels différents par des annotations différentes [Rivière, 2003] (cas des conflits de jetons).

Si nous revenons au réseau de Petri de la figure IV.4 avec les marquages $M_0 = P_1 \otimes P_3 \otimes P_3 \otimes P_5$ et $M_f = P_2 \otimes P_3 \otimes P_3 \otimes P_6$, nous pouvons vérifier que l'ensemble $\{e_a^1, e_a^2, e_b^1, e_c^1\}$ est suffisant. En effet le séquent : $P_1 \otimes P_3 \otimes P_3 \otimes P_5,\ t_a \otimes t_a \otimes t_b \otimes t_c \vdash P_2 \otimes P_3 \otimes P_3 \otimes P_6$ est prouvable. Par contre, l'ensemble $\{e_a^1, e_a^2, e_b^1\}$ n'est pas suffisant entre les marquages $M_0 = P_1 \otimes P_3 \otimes P_3 \otimes P_5$ et $M_f = P_2 \otimes P_3 \otimes P_3 \otimes P_6$ car le séquent $P_1 \otimes P_3 \otimes P_3 \otimes P_5,\ t_a \otimes t_a \otimes t_b \vdash P_2 \otimes P_3 \otimes P_3 \otimes P_6$ n'est pas prouvable.

IV.2 Remarque sur l'équation caractéristique

Il est intéressant de noter que notre notion d'ensemble d'événements suffisant est plus stricte que l'équation fondamentale. En effet l'équation fondamentale :

$$C\bar{l} = M_f - M_0 \qquad \text{(avec C= } post\text{-}pre\text{)}$$

est une *condition nécessaire mais non suffisante* pour que l'ensemble d'événements l soit *suffisant* (ceci est dû en particulier aux boucles élémentaires).

Considérons par exemple le réseau de Petri de la figure IV.8. Sa matrice d'incidence est :

$$C = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{cc} t_a & t_b \end{array} \\ \begin{bmatrix} -1 & 1 \\ 0 & 0 \\ 1 & -1 \end{bmatrix} \end{array}$$

Si nous considérons le vecteur $\bar{l} = \begin{array}{c} t_a \\ t_b \end{array} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

et les marquages initial et final $M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $M_f = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

alors nous avons bien : $M_f = M_0 + C \times \bar{l}$

mais l'ensemble d'événements l n'est pas suffisant car il faut un jeton dans la place P_2 pour pouvoir franchir la transition t_a, même si après le franchissement de t_a le marquage de cette place reste inchangé (présence d'une boucle élémentaire).

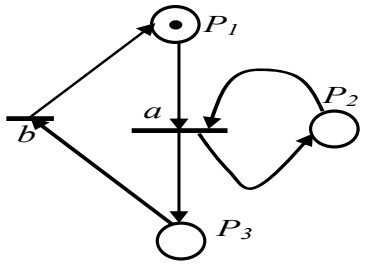


Figure IV.8. Exemple de boucle élémentaire

IV.3 Scénario suffisant

IV.3.1 Cas des réseaux de Petri ordinaires

Considérons un réseau de Petri *P = (P, T, Pre, Post)* avec un marquage initial M_0 et un marquage final M_f.

Définition IV.9 (***Scénario faiblement suffisant***) Soit un ensemble d'événements $l \subset (E \cup I \cup F)$ et soit $l' = l \cap E$ la restriction de *l* à *E*, le scénario $sc = (l, \prec_{sc})$ est faiblement suffisant pour passer de M_0 à M_f si le séquent $M_0, l' \vdash M_f$ est prouvable et s'il existe un ordre partiel $\prec_j$ résultant d'un arbre de preuve A_i tel que $\prec_j$ est inclus dans $\prec_{sc}$.

Cela veut dire que toute séquence qui est une linéarisation de l'ordre du scénario *sc* permettra de passer du marquage initial au marquage final puisqu'elle est aussi une linéarisation d'au moins un ordre partiel produit par au moins une preuve du séquent. Par contre, si l'on n'a pas $\prec_j = \prec_{sc}$, le scénario *sc* n'exprime pas clairement les relations de causalité présentes dans le réseau de Petri. C'est-à-dire que les relations de précédence de *sc* ne sont pas nécessairement des relations de causalité. Il existe au moins une relation de précédence parasite dans *sc* qui n'est pas imposée par la structure et les marquages du réseau de Petri.

Définition IV.10. (***Scénario suffisant***) : Soit un ensemble d'événements $l \subset (E \cup I \cup F)$ et soit $l' = l \cap E$ la restriction de *l* à *E*, le scénario $sc = (l, \prec_{sc})$ est *suffisant* pour passer de M_0 à M_f si le séquent $M_0, l' \vdash M_f$ est prouvable et s'il existe un ordre partiel $\prec_j$ résultant d'un arbre de preuve A_i tel que $\prec_j = \prec_{sc}$.

Cela veut dire qu'il faut non seulement que l'ensemble des événements soit suffisant pour faire passer de M_0 à M_f, mais qu'en plus l'un des ordres déduits de la preuve (et donc directement déduits des relations de causalité exprimées par le réseau de Petri et les deux marquages M_0 et M_f) soit exactement celui du scénario.

Exemple :

Considérons à nouveau le réseau de Petri de la figure IV.4 entre les marquages $M_0 = P_1 \otimes P_3 \otimes P_3 \otimes P_5$ et $M_f = P_2 \otimes P_3 \otimes P_3 \otimes P_6$. L'arbre de preuve du séquent

$M_0,\ t_a \otimes t_a \otimes t_b \otimes t_c \vdash M_f$ donne l'arbre de précédence de la figure IV.9. Le scénario est donc suffisant.

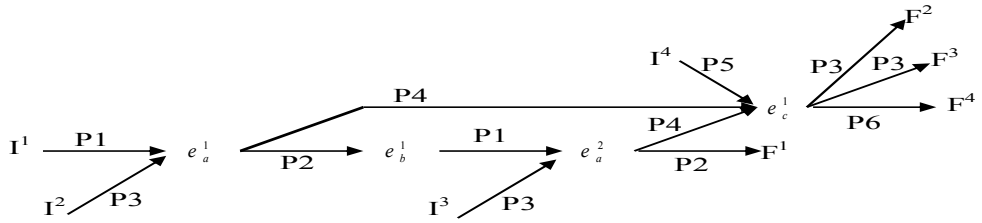


Figure IV.9. Graphe de précédence produit par la preuve par étiquetage

IV.3.2 Cas des réseaux de Petri temporels

Définition IV.11. (*Scénario suffisant*) **:** Soit un ensemble d'événements $l \subset (E \cup I \cup F)$ et soit $l' = l \cap E$ la restriction de l à E. Le scénario $sc = (l, \prec_{sc})$ associé au réseau de Petri temporel N_{tl} est *suffisant* pour passer de M_0 à M_f si et seulement si :

- Le séquent $M_0, l' \vdash M_f$ est prouvable.
- Il existe un ordre partiel $\prec_j$ résultant d'un arbre de preuve A_i tel que $\prec_j = \prec_{PNsc}$.
- Il existe un autre ordre partiel $\prec_T$ résultant des contraintes temporelles associées aux transitions du réseau de Petri tel que $\prec_T = \prec_{tsc}$

Rappelons que l'ordre partiel d'un scénario généré à partir d'un modèle réseau de Petri temporel est composé des relations d'ordres tirées de l'arbre de preuve du séquent ($\prec_{PNsc}$) et celles induites par les contraintes temporelles $\prec_{tsc}$.

IV.3.3 Ensemble nécessaire

Considérons un réseau de Petri *P = (P, T, Pre, Post)* avec un marquage initial M_0 et un marquage final M_f,

Définition IV.12. **(*Ensemble nécessaire*)** : Soit $l_k \subset E$ un ensemble d'événements suffisant pour passer de M_0 à M_f. L'ensemble d'événements l_k est nécessaire si :

$$\forall l_i \subset E \text{ suffisant pour passer de } M_0 \text{ à } M_f \text{ alors } \exists l_c \subset E,\ l_i = l_k \otimes l_c .$$

Cette notion exprime le fait que certains événements sont nécessaires pour passer d'un marquage à un autre. Tout ensemble d'événements permettant de passer du marquage initial au marquage final comprendra ces événements.

Si on considère le réseau de Petri de la figure IV.8, l'ensemble d'événements $l_k = t_a$ est nécessaire pour passer du marquage $M_0 = P_1 \otimes P_2$ au marquage $M_f = P_2 \otimes P_3$. En effet, on aura

également les ensembles $l_i = t_a \otimes (t_a \otimes t_b)^n$ (n est l'entier qui représente le nombre de fois que la boucle t_a, t_b est franchie), mais ils comprennent tous l_k .

V Minimalité

Dans un premier temps nous allons proposer une définition de la minimalité des scénarios dans le cas trivial où les marquages initial et final sont complètement connus. Nous verrons ensuite le cas où ces marquages ne sont que partiellement connus. Comme nous allons le voir, la définition est donnée pour des marquages initial et final minimaux, définis également dans cette section.

V.1 Minimalité dans le cas des marquages initiaux et finaux fixés

Dans le cas où les marquages initial et final sont connus et fixés d'avance, la définition de la minimalité se base sur la définition d'un ensemble minimal (définition IV.12) mais aussi des conditions de suffisance d'un scénario présentées dans les sections précédentes.

V.1.1 Ensemble minimal

Considérons un réseau de Petri *(P, T, Pre, Post)* avec un marquage initial M_0 et un marquage final M_f.

Définition IV.13. **(*Ensemble minimal*)** : Soit $l_k \subset E$ un ensemble d'événements suffisant pour passer de M_0 à M_f, l'ensemble d'événements l_k est minimal si et seulement il n'existe pas d'ensemble $l_i \subset E$ qui soit suffisant et qui soit un sous-ensemble de l_k .

Propriété **:** Si un ensemble d'événements est nécessaire pour passer de M_0 à M_f, il est minimal.

En effet, si l_k est nécessaire, tout autre ensemble d'événements suffisant l_i comprend nécessairement tous les événements de l_k , il ne peut donc pas être un sous-ensemble de l_k .

Ainsi, si nous considérons à nouveau le réseau de Petri de la figure IV.8, l'ensemble d'événements $l_k = t_a$ est minimal pour passer du marquage $M_0 = P_1 \otimes P_2$ au marquage $M_f = P_2 \otimes P_3$. Les ensembles $l_i = t_a \otimes (t_a \otimes t_b)^n$ ne le sont pas.

Remarque : la réciproque est fausse, c'est-à-dire qu'un ensemble d'événements peut très bien être minimal sans être nécessaire.

Par exemple (figure IV.10), bien que l'ensemble d'événements $l_1=f$ ne soit pas nécessaire entre les marquages $M_0 = P_4 \otimes P_5$ et $M_f = P_1$, il est minimal car d'autres ensembles d'événements suffisants comme par exemple $l_2 = d \otimes e \otimes e$ ne sont pas des sous-ensembles de l_1. Nous pouvons d'ailleurs remarquer que l_2 est également minimal. C'est-à-dire que pour couvrir toutes les trajectoires entre M_0 et M_f, il faut considérer les deux ensembles d'événements de l_1 et de l_2.

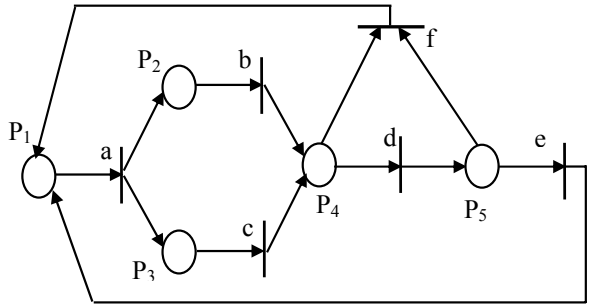


Figure IV.10. Exemple illustrant la minimalité.

V.1.2 Scénario minimal

V.1.2.1. Cas des réseaux de Petri ordinaires

Définition IV.14. (*Scénario minimal*) Soit un ensemble d'événements $l \subset (E \cup I \cup F)$ et soit $l' = l \cap E$ la restriction de l à E, le scénario $sc = (l, \prec_{sc})$ est minimal pour passer de M_0 à M_f s'il est suffisant et si l'ensemble d'événements l' est minimal.

Il faut donc que l'un des ordres partiels obtenus par étiquetage de l'un des arbres de preuve du séquent $M_0, l' \vdash M_f$ soit exactement $\prec_{sc}$.

Exemple (cas RdP ordinaire) :

Dans l'exemple de la figure IV.10, on peut remarquer que le scénario de la figure IV.11 est minimal alors que celui de la figure IV.12 ne l'est pas. En effet, dans le scénario de la figure IV.12 la relation de précédence entre les événements e_b^1 et e_c^1 est une relation parasite qui n'existe pas dans la structure du réseau de Petri.

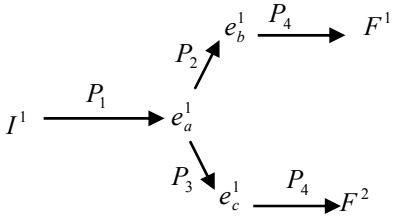


Figure IV.11. Scénario minimal

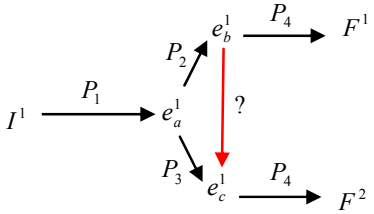


Figure IV.12. Scénario non minimal

V.1.2.2. Cas des réseaux de Petri temporels

La définition du scénario minimal est la même que dans le cas des réseaux de Petri ordinaires. Dans le cas des réseaux de Petri temporels, elle se base sur la définition du scénario suffisant qui contient les relations de précédence directes (tirées de l'arbre de preuve) et les relations de précédence indirectes (déduites à partir des contraintes temporelles).

Exemple *(cas temporel)* :

Dans le réseau de Petri de la figure IV.13, le scénario de la figure IV.14 n'est pas minimal entre les marquages $M_0 = EV1\_OK \otimes EV3\_OK$ et $M_f = EV1\_KO \otimes EV3\_KO \otimes Ered\_1$. La relation de précédence indirecte entre les événements *def1* et *def2* ne correspond à aucune contrainte temporelle dans le réseau de Petri. En effet, il n'y a aucun ordre entre les transitions *def1* et *def2*.

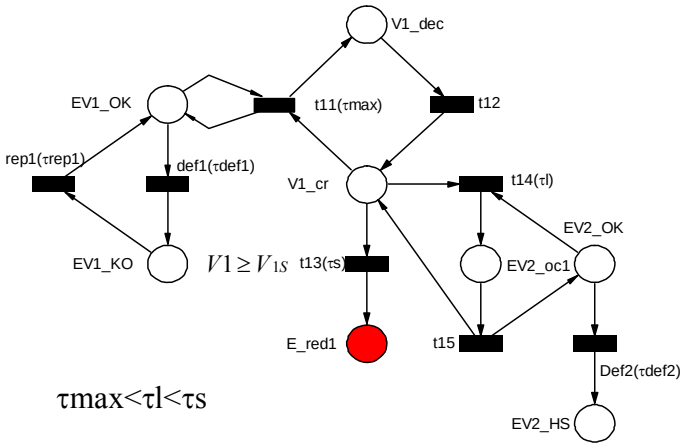


Figure IV.13. Exemple simplifié d'un système de régulation de volume.

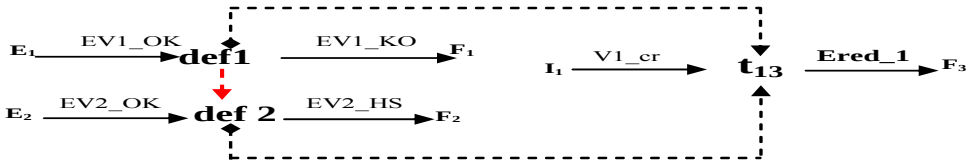


Figure IV.14. Exemple de scénario non minimal

V.2 Marquages initiaux et marquages finaux partiellement connus

Comme nous allons le voir dans le chapitre 5, la recherche des scénarios redoutés est limitée aux parties intéressantes du système d'un point de vue analyse de fiabilité [Sadou & al, 2005]. Nous n'analysons que les parties concernées par le scénario redouté. Dans ce cas, l'information concernant l'état (marquage) initial n'est que partielle et de l'état (marquage) final on ne connaît que la partie correspondant à l'état redouté. Nous allons voir que le choix des marquages, initial et final, a un impact sur la minimalité des scénarios.

Pour illustrer l'influence du choix des marquages sur la minimalité des scénarios, considérons l'exemple de la figure IV.15. Le modèle peut être vu comme trois composants représentés par les trois sous réseaux de Petri. On considère que l'état redouté correspond soit au marquage partiel de la place KO_s , soit au marquage simultané des places KO_1 et KO_2.

L'ensemble d'événements $l_1 = d_{1s} \otimes d_{2s}$ est minimal entre le marquage initial $M_0 = OK_s$ et le marquage final $M_f = KO_s$.

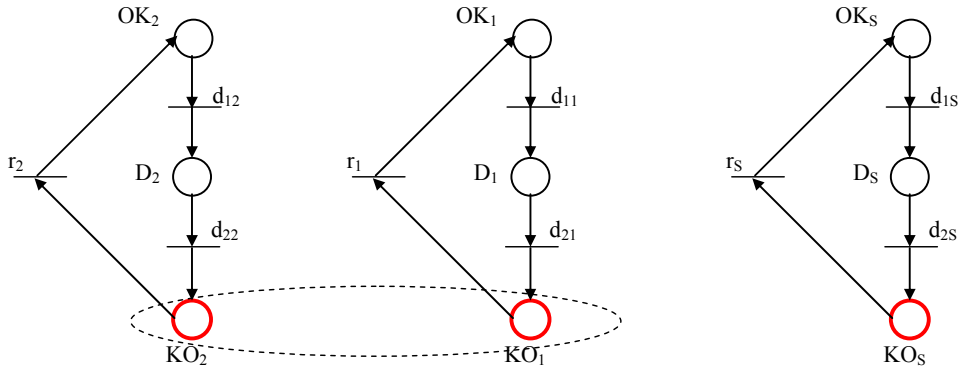


Figure IV.15. Réseau de Petri d'un système multi- composants

Entre les mêmes marquages, l'ensemble d'événements $l_1 = d_{1s} \otimes d_{2s} \otimes d_{11}$ n'est pas minimal car il n'est pas suffisant. En effet, le séquent $(OK_s,\ d_{1s} \otimes d_{2s} \otimes d_{11} \vdash KO_s)$ n'est pas prouvable. Par contre, il est suffisant entre les marquages $M_0 = OK_s \otimes OK_1$ et $M_f = KO_s \otimes D_1$.

Si on considère, par exemple, la séquence $s=d_{1S}\ ;d_{2S}\ ;d_{11}$, elle mène vers un état qui appartient à la classe des états redoutés (marquage de la place KO_S). D'un point de vue relations de causalité, on voudrait avoir uniquement les événements qui sont des causes directes de l'apparition de l'état redouté (marquage de la place KO_s dans l'exemple). Or, l'événement d_{11} n'a pas de relation de causalité avec l'avènement de l'état redouté représenté par le marquage de la place KO_s. Ainsi dans le scénario minimal l'événement d_{11} ne doit pas apparaître pour la première classe d'état redouté (marquage de la place KO_s) mais si on considère le marquage $M_f = KO_s \otimes D_1$ (marquage de la place D_1), comme marquage final, d_{11} apparaît nécessairement.

Le constat est que la caractérisation d'un scénario est dépendante du marquage final qui représente l'état redouté. Si le marquage final contient des états partiels inutiles (des jetons dans un modèle réseau de Petri), le scénario contiendra des événements inutiles. Il est donc nécessaire de définir un marquage final minimal.

V.2.1 Marquages initial et final minimaux

Afin de pouvoir définir la minimalité des scénarios dans le cas où les marquages ne sont que partiellement connus la définition doit être donnée par rapport à un marquage final minimal qui

représente l'état final. Ce marquage final minimal est basé sur la notion de coupe minimale [Sadou & Demmou, 2006c].

V.2.2 Coupe minimale associée à l'état redouté

Un système est composé d'un nombre d'entités qui interagissent entre elles. Si le modèle du système est un réseau de Petri, la délimitation entre les entités n'est pas explicite, mais à un moment donné, pour un marquage donné, on peut supposer que les jetons représentent le nombre d'entités actives.

Si on considère *B (P)* un booléen associé à la place *P*, le marquage associé à l'état redouté peut être représenté par une fonction booléenne. On note *B(P)* le booléen associé à la place *P*. Si *B(P)* est vrai cela veut dire que la place *P* est marquée et si *B(P)* est faux, la place est vide.

On suppose que la fonction booléenne *R* associée à l'état redouté est monotone et que *C={Ci}* est un ensemble de coupes minimales de *R*.

Dans l'exemple de la figure IV.15, la fonction *R* associée à l'état redouté est : (marquage de la place KO_S ou marquage simultané des places KO_1 et KO_2).

Deux coupes minimales associées à la fonction *R* sont :

$C_1 = B(KOs)$ et $C_2 = B(KO_1) \wedge B(KO_2)$.

Afin de considérer des marquages finaux minimaux, chaque coupe minimale représente un marquage minimal associé à l'état redouté. Il faut donc considérer et chercher les scénarios menant à chacune de ces coupes.

Définition IV.15 *(coupe minimale associée à un marquage final minimal) :* Le marquage final associé à une coupe C_i est caractérisé par $M_{fi} = C_i \otimes Cont_i$ où $Cont_i$ représente un contexte non défini. Le contexte $Cont_i$ est défini au fur et à mesure de la construction du scénario. Il correspond à des effets de bord (marquage de certaines places du réseau de Petri) conséquence du marquage des places correspondant à la coupe minimale.

Le marquage initial associé à chaque marquage final n'est pas connu. Le marquage initial du réseau de Petri est un multi-ensemble de jetons. Le marquage initial partiel est un sous ensemble du marquage initial complet $M_{0S}(M_0 \subset M_{0s})$, avec M_{0S} le marquage correspondant à l'état du système où tous les composants sont à l'état initial. M_{0s} correspond au marquage initial des composants réellement impliqués dans un état donné du système.

Remarque 1 : pour déterminer les coupes minimales, on suppose que les places associées à l'état redouté ont toutes un marquage sauf. Dans le cas contraire, une transformation est toujours possible afin de rendre le marquage des ces places sauf. L'idée est d'ajouter une place avec un marquage sauf et une transition qui consomme les jetons des places qui peuvent avoir un marquage non sauf pour produire un jeton dans la place avec le marquage sauf (un poids est donc associé à l'arc entrant de la transition).

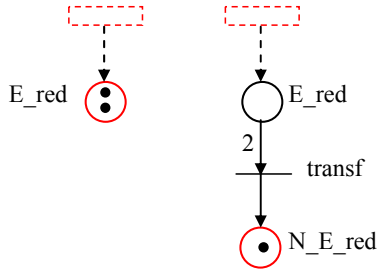


Figure IV.16. Transformation du marquage des places redoutées en un marquage sauf

Remarque 2 : Quand on travaille avec la logique linéaire on peut contourner le problème du contexte non spécifié. En effet, une preuve reste vraie si on enrichit de façon linéaire le contexte.

V.2.3 Scénario minimal associé à une coupe minimale

Maintenant que nous avons défini un marquage final minimal associé à une coupe minimale, nous allons donner la définition du scénario minimal associé à une coupe minimale. La définition du scénario minimal associé à une coupe minimale est liée à la notion de graphe de précédence restreint. Nous définissons ici ce qu'est un graphe de précédence restreint à un sous-ensemble d'événements, dans le cas de scénarios associés à un réseau de Petri ordinaire mais aussi temporel. La restriction consiste à supprimer certains événements du graphe et à le compléter par les relations de précédence induites par transitivité par les éléments supprimés.

Définition IV.16 (*graphe de précédence restreint*)**:** soit le graphe de précédence G associé au scénario $sc_i = (l_i, \prec_{sci})$ et soit l'ensemble d'événement l_j un sous ensemble de l_i.

Soit :

* C_m une composante connexe du graphe de précédence G composée par les événements de l'ensemble $(l_i - l_j)$.

* e_i événement de C_m.

* e_k événement de l_j tel que l'événement e_k précède au moins un événement e_i

* e_l événement de l_j tel qu'il existe au moins un événement e_i qui précède l'événement e_l.

Le graphe de précédence restreint *G_rest* restriction de G aux éléments de l_j est le graphe de précédence G où chaque composante connexe est supprimée et remplacée par :

- Les relations de précédence induites par les éléments de C_m dans G par transitivité entre les événements e_k et e_l .
- Un événement final s'il n'existe pas d'événement e_i qui précède l'événement e_l
- Un événement initial s'il n'existe pas d'événement e_k qui précède l'événement e_i.

Exemple :

Le graphe de précédence de la figure IV.17 correspond au scénario composé de l'ensemble d'événements $l_1 = e_a^1 \otimes e_a^2 \otimes e_a^3 \otimes e_b^1 \otimes e_b^2 \otimes e_c^1$. Sa restriction au sous ensemble d'événements $l_2 = e_a^1 \otimes e_a^3 \otimes e_b^1 \otimes e_c^1$ est obtenu en supprimant la composante connexe (arcs discontinues) et en complétant le graphe par l'arc auquel est associé l'atome P_1 (arc qui traduit la transitivité) liant les événements e_b^1 et e_a^1. On obtient ainsi le graphe de la figure IV.18.

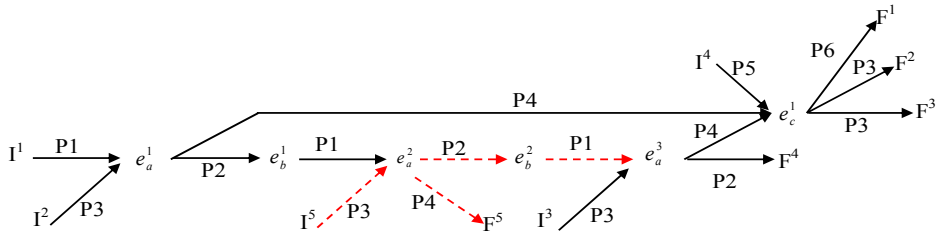


Figure IV.17. Exemple de graphe de précédence

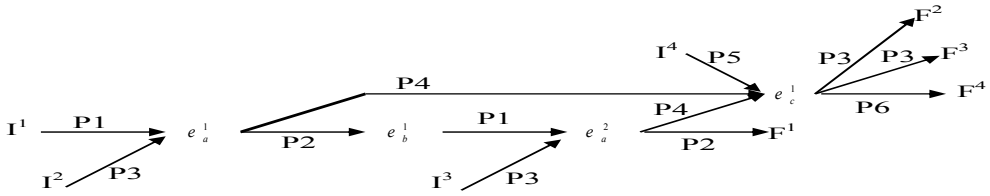


Figure IV.18. Graphe de précédence restreint obtenu à partir du graphe de la figure IV.16

***Définition IV.17* (*scénario minimal pour une coupe*)** : considérons un réseau de Petri et un marquage final $M_{fi} = C_i \otimes Cont_i$ associé à une coupe minimale C_i pour un état redouté. M_{0s} est le marquage initial du réseau. Soit un ensemble d'événements $l_i \subset (E \cup I \cup F)$ avec $l' = l \cap E$ la restriction de l_i à E et M_{0i} $(M_{0i} \subset M_{0s})$ un marquage initial donné. Le scénario $sc_i = (l_i, \prec_{sci})$ ayant pour graphe de précédence G est minimal pour la coupe C_i entre les marquages M_{0i} et M_{fi} s'il est suffisant entre M_{0i} *et* M_{fi}, et si et seulement s'il n'existe pas de scénario $sc_j = (l_j, \prec_{scj})$ ayant pour graphe de précédence G' tel que :

i) $sc_j = (l_j, \prec_{scj})$ est suffisant pour passer de M_{0j} à M_{fj} avec $(M_{0j} \subset M_{0s})$ et $M_{fj} = C_i \otimes Cont_j$ (même coupe minimale que pour le marquage M_{fi})

ii) $l_j \subset l_i$.

iii) $M_{0j} \subseteq M_{0i}$

iv) Il existe M_b un marquage (marquage partiel du réseau de Petri) tel que le séquent :

$Cont_j - (Cont_i \cap Cont_j) \otimes (M_{0i} - M_{0j}) \otimes M_b,\ (l_i - l_j) \vdash\ Cont_i - (Cont_i \cap Cont_j) \otimes M_b$ est prouvable (les termes *Cont* sont des multi-ensembles le signe '-' représente la soustraction).

v) *G'* est identique à la restriction (graphe de précédence restreint) de *G* aux éléments de l_j complétée par les relations de précédence induites par les éléments de *G* par transitivité.

Remarque sur la condition (iv) et (v) :

La condition *(iv)* traduit la présence d'une boucle qui correspond au séquent $Cont_j - (Cont_i \cap Cont_j) \otimes (M_{0i} - M_{0j}) \otimes M_b,\ (l_i - l_j) \vdash\ Cont_i - (Cont_i \cap Cont_j) \otimes M_b$ avec comme événements de la boucle la liste $(l_i - l_j)$.

M_b peut être un ensemble nul. Dans ce cas cela traduit la présence d'événements qui se déroulent en parallèle avec le scénario minimal.

La condition *(v)* traduit le fait que les événements de boucle sont inutiles. En effet, si le marquage M_b n'est pas consommé par au moins un événement de l_j alors la séquence composée des événements de $(l_i - l_j)$ produit des atomes qui ne sont nécessaires pour l'accessibilité au marquage final (associé à la coupe minimale).

Quand les conditions ***(iv)*** et ***(v)*** sont vérifiées, cela implique la présence de certains événements (événements de l'ensemble $(l_i - l_j)$) qui ne sont pas nécessaires. En effet, leur suppression ne modifie pas les relations de précédence entre le reste des événements (événements de l'ensemble l_j) qui sont suffisants pour atteindre le marquage final associé à la coupe minimale.

Exemple 1:

Dans l'exemple de la figure IV.19, si on considère la coupe minimale C_1=P_6, entre les marquages $M_{01} = P_1 \otimes P_3 \otimes P_3 \otimes P_5$ et $M_{f1} = P_2 \otimes P_3 \otimes P_4 \otimes P_6$, le scénario sc_1 correspondant à M_{01}, $l_1 \vdash M_{f1}$ composé de l'ensemble d'événements $l_1 = t_a \otimes t_a \otimes t_a \otimes t_b \otimes t_b \otimes t_c$ n'est pas minimal. En effet, il existe un scénario sc_2 correspondant à $M_{02}, l_2 \vdash M_{f2}$ composé de l'ensemble d'événements $l_2 = t_a \otimes t_a \otimes t_b \otimes t_c$ suffisant (condition *i*) entre les marquages $M_{02} = P_1 \otimes P_3 \otimes P_3 \otimes P_5$ et $M_{f2} = P_2 \otimes P_3 \otimes P_3 \otimes P_6$, tel que :

$l_2 \subset l_1$, (condition *ii*)

$M_{02} \subseteq M_{01}$ (condition iii)

Le séquent $Cont_j - (Cont_i \cap Cont_j) \otimes (M_{0i} - M_{0j}) \otimes M_b,\ (l_i - l_j) \vdash\ Cont_i - (Cont_i \cap Cont_j) \otimes M_b$ qui correspond dans cet exemple à $P_3 \otimes M_b,\ t_a, t_b \vdash P_4 \otimes M_b$ est prouvable avec $M_b = P_1$. (condition *iv*).

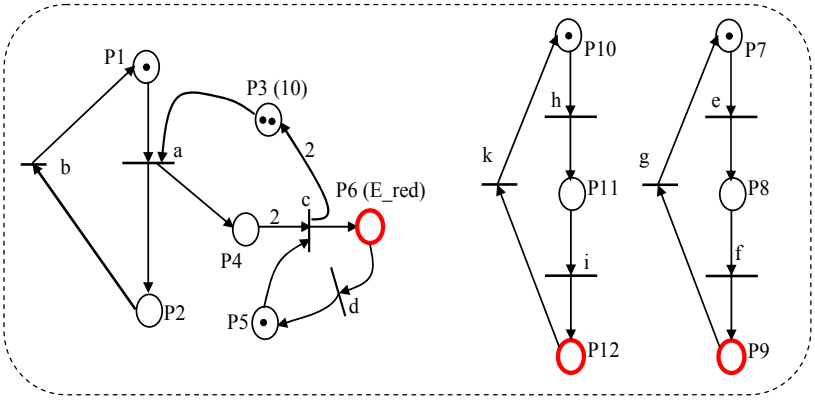


Figure IV.19. Réseau de Petri d'un système multi- composants

Le graphe de la figure IV.17 correspond au graphe de précédence du scénario sc_1. On peut remarquer que le graphe restreint aux éléments de *l2* correspond au graphe de la figure IV.18 qui représente le graphe de précédence du scénario sc_2.

Exemple 2 :

Dans l'exemple de la figure IV.20, si on considère la coupe minimale $C_1 = P_9$, entre les marquages $M_{01} = P_1 \otimes P_3 \otimes P_3 \otimes P_5 \otimes P_7$ et $M_{f1} = P_2 \otimes P_4 \otimes P_4 \otimes P_9$, le scénario sc_1 de la figure IV.21, composé des événements de l'ensemble $l_1 = t_a \otimes t_a \otimes t_b \otimes t_c \otimes t_d \otimes t_e$, est minimal. On peut noter que le scénario sc_2 (lui aussi minimal) de la figure IV.22 qui correspond au séquent : $P_1 \otimes P_3 \otimes P_5 \otimes P_7$, $t_a \otimes t_b \otimes t_c \otimes t_d \otimes t_e \vdash P_2 \otimes P_4 \otimes P_4 \otimes P_9$ satisfait les conditions ***(i)*** à ***(iii)*** mais pas la condition ***(v)***. Le tir de la séquence (t_a, t_b) dans le scénario sc_1 rend possible le franchissement en parallèle des transitions t_c et t_d, ce qui n'est pas le cas dans le scénario sc_2. Le graphe de précédence restreint du scénario sc_1 est différent du graphe de précédence du scénario sc_2. Les deux scénarios sont donc différents.

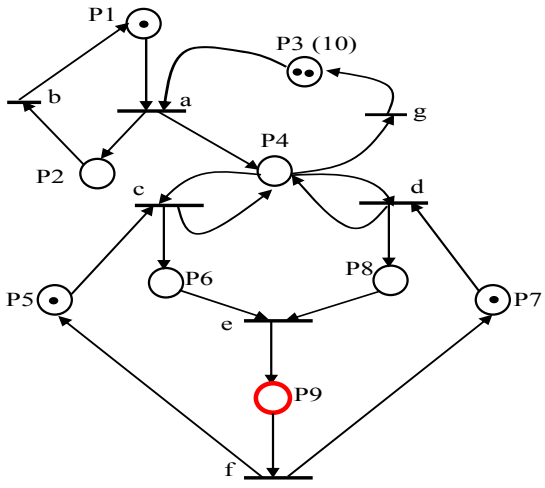


Figure IV.20. Exemple de réseau de Petri avec possibilité de parallélisme

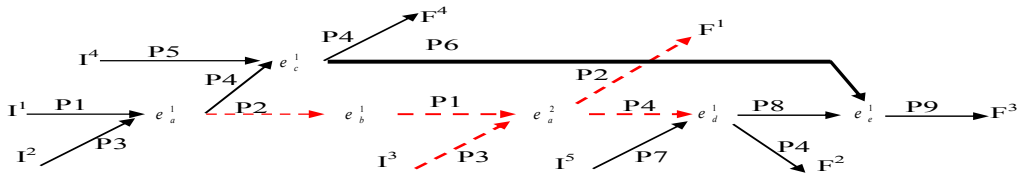


Figure IV.21. Scénario avec le franchissement en parallèle des transitions c et d.

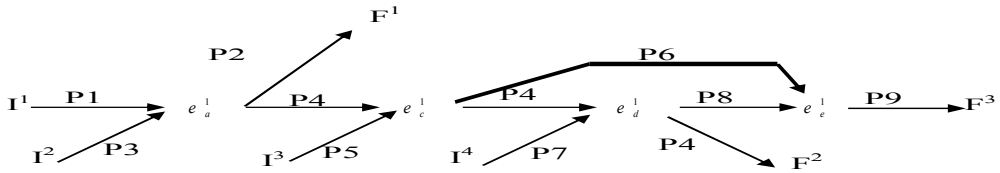


Figure IV.22. Scénario avec le franchissement en séquence des transitions c et d.

Exemple 3: (exemple de réseau de Petri temporel)

Reprenons l'exemple de la figure IV.13. Si on considère la coupe minimale $C_1 = E\_red1$, entre les marquages $M_{01} = EV1\_OK \otimes V1\_cr \otimes EV2\_OK$ et $M_{f1} = EV1\_KO \otimes EV2\_HS \otimes E\_red1$, le scénario *sc1* correspondant à M_{01}, $l_1 \vdash M_{f1}$ (figure IV.23) composé de l'ensemble d'événements $l_1 = def1^1 \otimes def1^2 \otimes rep1 \otimes def2 \otimes t13$, n'est pas minimal. En effet, il existe un scénario sc_2 (figure IV.24) suffisant entre les marquages $M_{02} = EV1\_OK \otimes V1\_cr \otimes EV2\_OK$ et $M_{f2} = EV1\_KO \otimes EV2\_HS \otimes E\_red1$. Ce scénario correspondant à M_{02}, $l_2 \vdash M_{f2}$ est composé de l'ensemble d'événements $l_2 = def1^1 \otimes def2 \otimes t13$ avec $l_2 \subset l_1$. Le séquent :

$Cont_j - (Cont_i \cap Cont_j) \otimes (M_{0i} - M_{0j}) \otimes M_b$, $(l_i - l_j) \vdash Cont_i - (Cont_i \cap Cont_j) \otimes M_b$ qui correspond dans cet exemple à M_b, $def1 \vdash M_b$ (il s'agit d'une boucle élémentaire, tous les autres contextes sont vides) est prouvable avec $M_b = EV1\_OK$.

Le graphe de la figure IV.23 correspond au graphe de précédence du scénario sc_1. On peut remarquer que le graphe complété par les relations de précédence induites par les éléments de $(l_1 - l_2) = def1$ par transitivité est identique à celui de la figure IV.24, qui lui correspond bien au graphe de précédence du scénario sc_2.

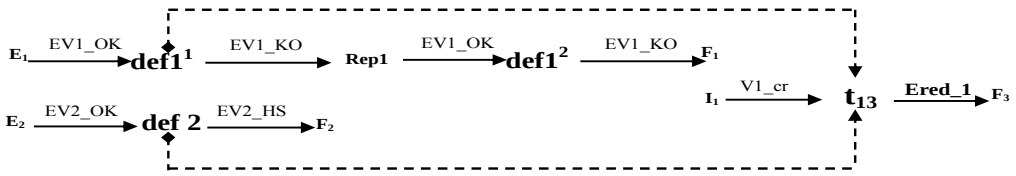


Figure IV.23. Graphe de précédence associé au scénario SC_1 (scénario non minimal)

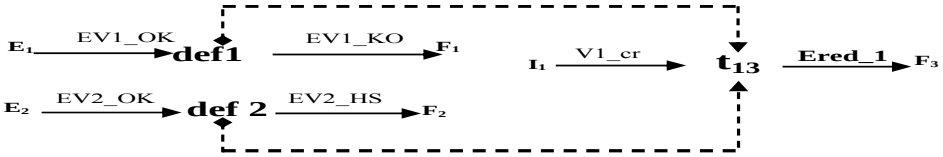


Figure IV.24. Graphe de précédence associé au scénario SC_2 (scénario minimal)

V.2.4 Algorithme de filtrage des scénarios minimaux

La définition du scénario minimal n'étant pas constructive, pour la construction des scénarios minimaux l'idée est donc de construire les différents scénarios possibles et d'éliminer ensuite les scénarios non minimaux. L'algorithme de filtrage des scénarios minimaux se base sur la définition donnée du scénario minimal (définition IV.16) en comparant chaque scénario avec les autres scénarios construits correspondant à la même coupe minimale.

Algorithme :

```
Soit Ci une coupe minimale associée à un état redouté.
Soit SCi ={sc1, sc2, …scm} l'ensemble des scénarios associé à la
coupe Ci
Pour j allant de 1 à m
     Sc= SCi[j] (avec sci = (li, <sci)
     Pour k allant de j+1 à m
     Si lk est inclus dans li
          Si M0k est inclus dans M0i
               Si Cont_j − (Cont_i ∩ Cont_j) ⊗ (M_0i − M_0j) ⊗ M_b, (l_i − l_j) • Cont_t − (Cont_t ∩ Cont_j) ⊗ M_b
               est prouvable
                    si le graphe de précédence associé a la
restriction du scénario  j au élément de k   est identique à
celui de k.
                         -  Supprimer le scénario scj.
                         sinon
                    Sinon
          sinon
     sinon
K++
J++
```

VI Complétude

La définition de la complétude des scénarios est liée à la minimalité. Tout comme pour la minimalité on distingue deux cas: le cas où les marquages initial et final sont complètement connus et le cas où le marquage initial et final ne sont que partiellement connus. Dans le premier cas, la définition est assez triviale, par contre quand le contexte (marquages) n'est que partiellement connu (dans le cas de la recherche de scénarios redoutés), la définition est donnée pour un marquage initial minimal et un marquage final minimal. Le deuxième cas sera traité dans le chapitre 5 une fois l'approche de recherche de scénarios redoutés présentée.

Un ensemble de scénarios est complet entre deux marquages s'il contient tous les scénarios suffisants permettant de passer du marquage initial au marquage final. On peut distinguer deux formes d'ensemble de scénarios complets ; un ensemble de scénarios complet et un ensemble de scénarios complet et minimal. Dans le deuxième cas, l'ensemble doit contenir tous les scénarios minimaux et uniquement les scénarios minimaux.

Complétude (marquages initial et final connus)

Dans un premier temps nous donnons la définition d'un ensemble de scénarios complet dans le cas trivial où les marquages initial et final sont complètement connus.

Définition II.18 (*ensemble complet de scénarios*): soit le réseau de Petri $P = (P, T, Pre, Post)$, les marquages initial et final M_0 et M_f.

Soit $SC = \{sc_1, sc_2, \ldots, sc_n\}$ un ensemble de scénarios suffisants entre les marquages M_f et M_0.

L'ensemble SC est complet entre les marquages M_0 et M_f si et seulement s'il n'existe pas de scenario sc_i minimal entre les marquages M_f et M_0 tel que $sc_i \notin SC$.

La définition implique que tous les scénarios minimaux appartiennent à l'ensemble SC.

Exemple :

Dans l'exemple de la figure IV.25 entre les marquages $M_0=P_1$ et $M_f=P_4$, l'ensemble de scénarios $SC = \{sc_1, sc_2, sc_3, sc_4\}$ est complet entre ces deux marquages ($M_0=P_1$ et $M_f=P_4$) avec :

- $sc_1 : P_1, a, c \vdash P_4$
- $sc_2 : P_1, b, d \vdash P_4$
- $sc_3 : P_1, a, e \vdash P_4$
- $sc_4 : P_1, a, f, a, c \vdash P_4$

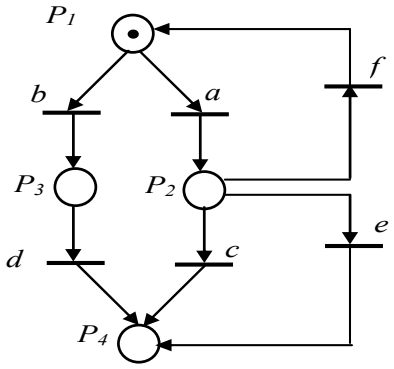


Figure IV.25. Exemple d'illustration d'un ensemble de scénarios complet.

Définition IV.19 (*ensemble complet et minimal de scénarios*): soit le réseau de Petri $P = (P, T, Pre, Post)$, les marquages initial et final M_0 et M_f.

Soit $SC = \{sc_1, sc_2, \ldots, sc_n\}$ un ensemble de scénarios suffisants entre les marquages M_f et M_0.

L'ensemble SC est complet et minimal entre les marquages M_0 et M_f si est seulement si :

- Tout scénario de *SC* est minimal entre les marquages M_f et M_0.
- Il n'existe pas de scénario sc_i minimal entre les marquages M_f et M_0, tel que $sci \notin SC$

L'ensemble de scénarios est complet et minimal s'il contient tous les scénarios minimaux et uniquement les scénarios minimaux.

Dans l'exemple de la figure IV.25, l'ensemble de scénarios $SC=\{ sc_1, sc_2, sc_3\}$ est complet et minimal entre ces deux marquages ($M_0=P_1$ et $M_f=P_4$), par contre l'ensemble $SC' =\{sc_1, sc_2, sc_3, sc_4\}$ est complet mais non minimal. En effet, le scénario sc_4 n'est pas minimal.

VII Conclusion

Dans ce chapitre, nous avons présenté la notion de minimalité des scénarios critiques (qui mènent vers un état redouté) générés à partir d'un modèle réseau de Petri. En effet, un scénario peut mener vers un état redouté sans qu'il soit minimal (il contient les événements qui ne sont pas strictement nécessaires pour atteindre l'état redouté final). La nouvelle représentation du réseau de Petri avec des formules de logique linéaire nous permet de définir formellement la notion de scénario minimal dans le cas d'un réseau de Petri ordinaire et dans le cas des réseaux de Petri temporels.

Pour obtenir le scénario minimal, nous devons considérer trois aspects : (*i*) les relations d'ordre entre les événements doivent être présentes dans le modèle réseau de Petri et générées par les contraintes temporelles qui lui sont associées (causalité présente dans le système), (*ii*) la liste d'événements du scénario doit être minimale (sans événements de boucle du système) (*iii*) le marquage final correspondant à l'état redouté doit être minimal.

Nous avons donc proposé une définition d'un marquage final minimal associé à l'état redouté. Ce marquage minimal se base sur la notion de coupe minimale et conduit à la définition due scénario minimal.

A partir de la définition de la minimalité nous avons proposé une définition pour la complétude dans le cas trivial où les marquages initial et final sont complètement connus mais aussi dans le cas où ils ne sont que partiellement connus.

Comme nous allons le voir dans le chapitre suivant qui sera consacré à l'approche de génération de scénarios redoutés, la minimalité est prise en compte grâce à l'algorithme que nous avons établi. L'approche permet ainsi de générer uniquement les scénarios redoutés minimaux qui contribuent à une bonne compréhension des conséquences qualitatives des choix de conception. De plus, l'évaluation quantitative de la sûreté de fonctionnement doit, elle aussi, reposer sur des scénarios minimaux si l'on veut être efficace. Une définition de la complétude dans le cas des marquages minimaux, sera présentée.

Chapitre 5 : Approche orientée objets pour la génération de scénarios redoutés

I Introduction

Dans ce chapitre nous allons voir la méthode de recherche de scénarios redoutés proposée par [Khalfaoui, 2003] et améliorée par [Medjoudj, 2006]. La méthode permet de générer les scénarios redoutés dans un système embarqué, à partir d'une modélisation réseau de Petri. Dans la première version de la méthode [Khalfaoui, 2003], même si la modélisation est basée sur les réseaux de *Petri Prédicats transitions différentiels stochastiques*, l'approche ne prend en compte que les aspects discrets. L'évolution de la dynamique continue n'est pas prise en compte. [Medjoudj, 2006] a introduit partiellement l'aspect continu. En effet, la partie continue est prise en compte grâce à des abstractions temporelles de la dynamique continue. Un modèle réseau de Petri temporel est ainsi déterminé et analysé. La prise en compte partielle de la partie continue a permis d'éliminer un certain nombre de scénarios générés dans la première version et qui sont incohérents avec la dynamique continue.

La méthode de recherche de scénarios redoutés s'appuie sur un cadre formel qui est la logique linéaire. Ce cadre formel permet d'extraire les scénarios directement du modèle réseau de Petri. La logique linéaire permet aussi de focaliser l'analyse sur les parties intéressantes du système d'un point de vue fiabilité, évitant ainsi l'exploration du système complet et l'éternel problème de l'explosion combinatoire.

Nous allons présenter dans ce chapitre la version orientée objets de la méthode de génération de scénarios redoutés. L'approche objets permet d'aborder la recherche des scénarios d'une manière structurée et en ne s'intéressant qu'aux objets réellement concernés par les scénarios. Cette approche orientée objets simplifie l'étape d'analyse des scénarios.

Outre l'approche orientée objet, de nouvelles procédures ont été introduites dans l'algorithme, ces procédure seront présentées en détail.

La notion de minimalité présentée dans le chapitre précédent est elle aussi, prise en compte par l'algorithme. L'ensemble des scénarios générés se limite donc aux scénarios minimaux.

Il a été nécessaire de remodeler l'algorithme défini dans [Medjoudj, 2006] afin de le rendre plus performant et prendre en compte les aspects cités plus haut.

II Méthode de recherche des scénarios redoutés

II.1Principe de la méthode

La méthode est basée sur une analyse qualitative du modèle réseau de Petri du système. Elle permet d'extraire et d'identifier clairement les scénarios redoutés (scénarios menant vers un état redouté) à partir d'une connaissance partielle de l'état redouté nécessaire pour faire l'analyse.

Le but de la méthode est de mettre en évidence les suites d'actions qui mènent aux états redoutés. Cela permet d'expliquer ce qui fait que le système quitte le fonctionnement normal pour aller vers l'état redouté ou de vérifier que l'état redouté n'est pas atteignable. La méthode consiste à remonter la chaîne de causalités par un raisonnement arrière en partant de l'état partiel redouté. Ce raisonnement est poursuivi jusqu'à ce que l'on arrive à un état de fonctionnement normal. Ensuite un raisonnement avant est mené à partir de l'état partiel normal obtenu afin de mettre en évidence les bifurcations entre le comportement normal et le comportement redouté. L'analyse de ces bifurcations (qui apparaissent sous la forme de conflits entre transitions) permet d'appréhender dans quelles conditions le comportement redouté a lieu.

L'introduction des réseaux de *Petri Prédicats Transitions différentiels orientés objets* permet non seulement d'aborder la modélisation du système d'une manière progressive et structurée, mais aussi son analyse, notamment la recherche des scénarios redoutés en ne considérant que les objets concernés. Cette modélisation orientée objets consiste à décomposer le système en un ensemble d'objets regroupés en classes, un réseau de Petri PTD modélise alors le comportement d'une classe. Les communications entre objets résultent des interactions entre les entités du système.

La connaissance partielle des conditions (contexte) d'occurrence de l'événement redouté, par rapport au marquage du réseau de Petri (marquage partiel) et par rapport aux objets impliqués, nous amène à enrichir le contexte au fur et à mesure que l'analyse progresse. On entend par enrichissement du contexte, d'une part l'enrichissement du marquage de certaines places du réseau de Petri et d'autre part l'implication de nouveaux objets dans l'analyse.

Les concepts orientés d'objets permettent de faire une recherche modulaire. Uniquement les objets concernés par le scénario redouté sont explorés, l'enrichissement du contexte par rapport aux objets correspond donc à la prise en compte dans l'analyse de nouveaux objets pouvant avoir un lien avec l'état redouté.

L'analyse commence donc dans l'objet (ou les objets) directement concerné par l'état redouté. Progressivement les objets qui sont en interaction avec les objets déjà analysés sont pris en compte. L'étude du comportement des objets qui sont en conflit avec l'événement redouté (et

par conséquent favorise l'occurrence de l'événement redouté) nous fournit des informations plus précises sur le contexte (si l'objet est impliqué ou pas et comment) du scénario redouté.

II.2Etapes de la méthode

La méthode proposée contient quatre étapes. Son but est de déterminer systématiquement et formellement les conditions d'occurrence de l'événement redouté sous forme d'un scénario ou au contraire de montrer que l'occurrence de l'événement redouté est impossible. Les étapes de la méthode sont les suivantes :

1. Détermination des états de normaux du système
2. Détermination des états cibles (états partiels redoutés ou états à analyser).
3. Raisonnement arrière à partir de l'objet (les objets) contenant l'état de cible.
4. Raisonnement avant à partir de l'objet (des objets) qui contient les états conditionneurs.

II.2.1 Détermination des états normaux

La première étape consiste à déterminer les places dont le marquage représente un état de fonctionnement normal. Ces places nominales seront utilisées comme critère d'arrêt du raisonnement arrière. Cette étape peut être réalisée de deux manières : soit en utilisant une connaissance a priori des états de bon fonctionnement du système soit en effectuant une simulation de Monte Carlo du modèle sur une courte fenêtre temporelle pour déterminer la probabilité de marquage des places du réseau. Celles qui auront une probabilité de marquage non négligeable seront assimilées à des places normales.

II.2.2 Détermination des états cibles

La deuxième étape détermine l'état cible à étudier. Cet état cible peut être soit un état partiel redouté soit un autre état partiel ayant un lien de causalité direct ou indirect avec cet état redouté (par exemple une place qui représente la disponibilité d'une ressource pour assurer un fonctionnement dégradé évitant l'occurrence de l'événement redouté).

Une fois les états redoutés déterminés, on exprime le marquage correspondant sous forme de fonction booléenne. Les coupes minimales associées à cette fonction booléenne sont ensuite déterminées. Pour garantir la minimalité des scénarios générés, les coupes minimales déterminées seront considérées comme marquage final minimal.

II.2.3 Raisonnement en arrière

La troisième étape génère l'ensemble des chemins qui mènent vers l'état partiel redouté. On effectue un raisonnement sur le modèle réseau de Petri inversé des différents objets. Dans ce réseau inversé, on prend comme marquage initial l'état partiel redouté (la coupe minimale associée au marquage de l'état redouté, voir chapitre 4) et l'on cherche de façon exhaustive tous les scénarios minimaux (aucun franchissement de transition non nécessaire n'est effectué)

permettant de consommer le marquage initial et aboutissant à un marquage final uniquement formé de places associées au fonctionnement normal. Au cours de cette étape, on est en général amené à enrichir le contexte : enrichissement du marquage initial (introduction de jetons dans les places non marquées) et enrichissement de l'ensemble d'objets à analyser.

L'enrichissement de marquage se fera pour rendre franchissables les transitions potentiellement franchissables (seules certaines places d'entrée de la transition sont marquées) menant de l'état redouté vers un fonctionnement. Les jetons ajoutés lors du processus d'enrichissement du marquage correspondent à des états partiels qui sont des conséquences logiques de l'apparition des scénarios redoutés et qui seront donc nécessairement observés lors de l'évolution du système vers l'état redouté.

La prise en compte d'objets autres que ceux considérés initialement se fait à chaque fois qu'il y a interaction (appel ou offre de méthodes ou partage de variables continues) avec les objets initiaux.

En inversant les scénarios obtenus lors de cette étape, nous aurons les suites d'actions possibles menant d'un état normal à l'état redouté. Cet état normal est nommé état conditionneur.

II.2.4 Raisonnement avant

La dernière étape de la méthode consiste à construire un raisonnement à partir du modèle réseau de Petri initial en partant de chaque état conditionneur déterminé à l'étape précédente. Le but est de déterminer si l'état redouté peut être atteint et d'identifier les scénarios qui mènent vers lui ou de conclure que l'état redouté ne peut jamais être atteint. Il faut pour cela localiser les bifurcations entre le comportement redouté et le fonctionnement normal du système ainsi que les conditions (marquage de certaines places du réseau et objets concernés) associées à ces bifurcations. L'analyse de ces bifurcations fournit des informations sur les événements qui se sont produits et qui sont les causes d'occurrence de l'événement redouté. Comme dans l'étape du raisonnement arrière, les objets sont introduits de manière progressive.

II.3Raisonnement dans un contexte inconnu

Ce qui nous intéresse, c'est de trouver une suite d'actions (tirs de transitions), avec le contexte nécessaire correspondant (objets impliqués), qui aboutisse à l'apparition de jetons dans les places représentant l'état partiel jugé dangereux. On ne connaît donc pas le marquage initial, et du marquage final, on ne connaît que le fragment correspondant à l'état partiel dangereux. Cette connaissance partielle du contexte d'occurrence de l'événement redouté (marquage partiel du réseau de Petri) nous amène à enrichir le contexte au fur et à mesure que l'analyse progresse. Cela permet de prendre en compte les autres composants du système qui pourraient avoir un lien avec l'état redouté. On entend par enrichissement du contexte, l'enrichissement de marquage de certaines places du réseau de Petri. Nous avons identifié et défini deux types d'enrichissement de marquage [Sadou & al, 2006a] : enrichissement minimal et enrichissement maximal.

Avant de définir les différents enrichissements possibles, considérons d'abord les différents conflits possibles de transitions dans un réseau de Petri PTD.

Définition V.1 : Soit *R* un réseau de Petri marqué. Une *transition potentiellement franchissable* est une transition dont au moins une place amont est suffisamment marquée et au moins une place amont ne l'est pas (il lui manque au moins un jeton).

II.3.1 Différents conflits de transitions

Dans un réseau de Petri PTD, une fonction de sensibilisation est associée à chaque transition qui constitue un prédicat (seuil) pour son franchissement et qui dépend des variables associées aux jetons des places d'entrée. A cause de la dynamique du système, et du marquage partiel du réseau de Petri, on peut rencontrer deux types de conflits entre transitions franchissables et transitions potentiellement franchissables.

Un premier cas de conflit entre une transition franchissable et une transition potentiellement franchissable est celui où les seuils associés à ces deux transitions sont indépendants. Dans ce cas l'ordre de tir des transitions n'est pas déterminé. Le second cas est celui où les seuils associés aux transitions ne sont pas indépendants, c'est-à-dire qu'un ordre de franchissement entre ces transitions est défini par ces seuils.

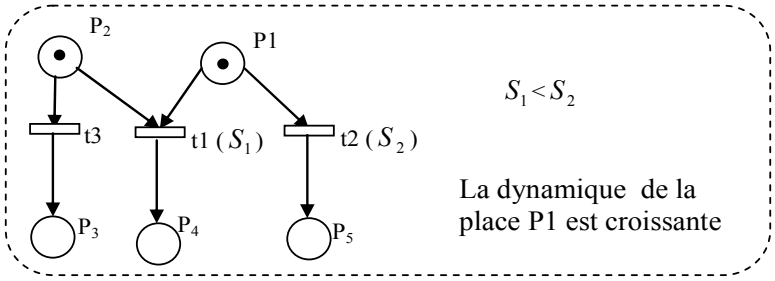


Figure V.1. Exemple de conflit chien de garde

La figure V.1 illustre le deuxième cas. Elle montre un fragment de réseau de Petri PTD. La variable continue *S* est associée à la place P_1, la dynamique de cette variable est croissante.

Les seuils S_1, respectivement S_2, sont associés aux transitions t_1, respectivement t_2, avec $S_1<S_2$. Dans ce cas, il est clair que la transition t_1 (transition prioritaire) est toujours franchie avant t_2 à condition que la place P_2 soit marquée. La transition t_2 n'est franchie que si la transition t_1 n'est plus franchissable.

Dans le cadre de la recherche de scénarios dans un contexte inconnu, nous avons appelé ce type de conflit : conflit du type chien de garde hybride.

Définition V.2 (Conflit chien de garde hybride)**:** Deux transitions t_1 et t_2 sont en conflit de type chien de garde hybride si :

- La transition t_1 est en conflit structurel avec la transition t_2.

- Les seuils continus associés à la transition t_1 et à la transition t_2 sont tels que la transition t_1 est prioritaire par rapport à la transition t_2 (t_1 est franchissable avant t_2)
- La transition t_1 est potentiellement franchissable et la transition t_2 est franchissable.

Cette notion de conflit chien de garde hybride est utile dans notre démarche de recherche de scénarios. Le franchissement de la transition t_2 est forcément dû au non franchissement de la transition t_1. Si lors de l'analyse un tel conflit est rencontré, il représente un comportement anormal (dangereux) possible du système, causé par l'absence de jetons dans la place d'entrée vide de la transition t_1 (P_2 dans l'exemple). Il faut donc déterminer les événements qui ont consommé les jetons dans les places d'entrée vides de la transition t_1 afin de reconstituer le scénario complet du fonctionnement 'anormal' qui correspond au franchissement de la transition t_2.

L'analyse de ces conflits mettant en jeu des transitions potentiellement franchissables se fait après enrichissement de marquage. Selon le type de conflit nous définissons les enrichissements de marquage suivants :

II.3.2 Différents enrichissements de marquage

Le processus d'enrichissement de marquage consiste à ajouter les jetons manquants à la transition potentiellement franchissable de sorte qu'elle devienne franchissable. Mais il faut vérifier que cet enrichissement n'introduit pas d'incohérences dans le système, par exemple qu'un unique composant du système ne se trouve pas dans deux configurations différentes en même temps ce qui serait contradictoire avec la réalité du système physique. Il faut donc interdire les enrichissements de marquage contradictoires avec la structure du système.

1) ***Enrichissement maximal de*** **marquage :** l'enrichissement maximal de marquage consiste à introduire le ***maximum de jetons possible*** dans les places d'entrée non marquées des transitions potentiellement franchissables impliquées dans un conflit du type chien de garde hybride. Cet enrichissement permet de retrouver les conditions normales qui feraient de la transition prioritaire une transition franchissable autant de fois que possible. Tant que la transition prioritaire est franchissable on est dans le fonctionnement 'normal' du système.

2) ***Enrichissement minimal de marquage*** : l'enrichissement minimal de marquage consiste à introduire le ***minimum de jetons nécessaire*** dans les places d'entrée non marquées des transitions potentiellement franchissables non impliquées dans un conflit de type chien de garde hybride afin de les rendre franchissables.

Dans les deux cas, l'enrichissement est effectué après vérification de la cohérence avec les invariants de marquage du modèle.

Pour l'enrichissement minimal, le nombre de jetons à introduire doit être égal au minimum de jetons qui rendrait la transition franchissable, et ce minimum doit être inférieur ou égal à l'invariant de marquage associé à la place. Dans ce cas, seuls les événements nécessaires qui ont un lien avec l'état redouté sont recherchés afin de garantir une minimalité pour les scénarios.

Par contre, pour l'enrichissement maximal le nombre de jetons à introduire doit être égal au poids associé à la place dans l'invariant, c'est-à-dire le maximum de jetons possible. Le fait de considérer le maximum de jetons possible permet de garantir que toutes les ressources représentées par des jetons ont été considérées.

II.3.3 Mécanisme de contrôle de la cohérence

Le mécanisme de contrôle de la cohérence de l'enrichissement du marquage se fait par le calcul des invariants de places [David & alla, 1992] du réseau. Le but est de vérifier que l'ajout d'un ou plusieurs jetons dans une ou plusieurs places du réseau ne viole pas un invariant de place. Si c'est le cas, l'enrichissement de marquage correspondant n'est pas autorisé.

Les invariants sont calculés une fois pour toutes en prenant pour marquage initial celui qui correspond à l'état initial de tous les composants du système. La valeur obtenue pour le marquage correspondant à l'étape courante de l'analyse doit donc toujours être inférieure ou égale aux invariants. Elle sera inférieure si certains composants du système impliqués dans l'invariant ne sont pas nécessaires au scénario étudié. Si la valeur est supérieure, cela veut dire qu'il y a incohérence. Il serait en effet impossible d'associer un état du système complet à l'étape courante du raisonnement en ajoutant des jetons.

Exemple : La figure V.2 représente un système qui peut être dans deux états : en marche (P_1 marquée) ou en panne (P_2 marquée). Comme le système ne peux être dans deux états au même temps, l'invariant de places suivant est associé au modèle réseau de Petri :

$M(P_1)+ M(P_2) = 1$

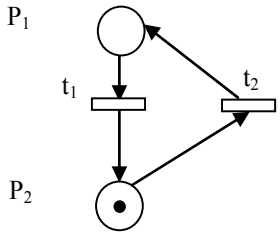


Figure V.2. Contrôle de la cohérence du marquage

L'enrichissement de marquage d'une des deux places n'est donc pas possible si l'autre place est marquée.

III Nouvelle version de l'algorithme de recherche de scénarios

L'algorithme est constitué de différentes procédures et structures de données manipulant des ensembles de transitions qui peuvent être en conflit pour pouvoir générer tous les scénarios possibles. L'algorithme initial [Medjoudj, 2006] a été remodelé ; de nouvelles procédures et structures de données ont été rajoutées afin de prendre en compte tous les aspects développés dans cette thèse.

III.1 Structures de données

Les structures de données utiles au déroulement de l'algorithme peuvent être classées selon trois groupes différents : les données d'entrée, les données internes et les données de sortie.

III.1.1 Données d'entrée

Les données d'entrée de l'algorithme se composent exclusivement de la liste des jetons initiaux Li, en plus du (des) réseau(x) de Petri étiqueté(s) dans le but d'indiquer les places normales. C'est à partir de cette liste que démarreront une ou plusieurs exécutions de l'algorithme.

III.1.2 Données internes

La première donnée interne correspond à l'ensemble des contextes : *C = { c1, c2, … }*, où *ci* contient les informations nécessaires pour dériver un nouvel ordre partiel. Un contexte *ci*, donnant un état courant du système et fournissant un enregistrement des évènements passés, est en fait composé des différentes listes courantes *{E, E2, A, B, C, Le, Lc, Lc_ord, Lc_int, Lint1, Lint_mc, Lint_ord}*.

Ensuite, viennent les listes courantes à partir desquelles l'algorithme s'exécute et qui représentent le contexte courant :

- *E* : liste d'événements (ou d'instances de tirs), qui sont attachés aux transitions
- *E2* : liste d'événements qui sont issus de transitions appelées.
- *A* : liste des liens directs, reliant des éléments de *E* et/ou des éléments de E2.
- *B* : liste des liens indirects, reliant des éléments de *E* et/ou des éléments de *E2*.
- *C* : liste des liens d'appels de méthodes, reliant des éléments de *E* avec des éléments de *E2*.
- *Le* : liste des jetons d'enrichissement de transitions. Un jeton de cette liste est identifié par une paire *(e, p)*, où *e* est l'événement d'enrichissement et *p* est la place qui contient le jeton.
- *Lc* : liste des jetons courants, qui est mise à jour par l'algorithme après chaque franchissement de transition. Un jeton est identifié par une paire *(evt, p)*, où *evt* est l'événement producteur du jeton (événement initial *i*, enrichissement de marquage *e* ou tir d'une transition t_i) et *p* est la place qui contient le jeton.
- *Lc_ord* : liste des jetons d'enrichissement des transitions en **Conflit Chien de Garde Hybride** (CCGH). Les jetons de cette liste sont identifiés par un couplet *((e, p), t)*, où *(e, p)* est le jeton d'enrichissement (également présent dans *Le*) et *t* est la transition franchissable du CCGH.
- *Lc_int* : liste courante des jetons qui ne sont plus utilisables pour garantir la minimalité par rapport au CCGH.

- *Lint1* : liste des transitions interdites.
- *Lint_mc* : liste de groupes de transitions qui permet de gérer les choix multiples de franchissement de transitions.
- *Lint_ord* : liste de groupes de transitions franchissables ou potentiellement franchissables en conflit. Cette liste gère ces groupes de transitions selon une politique LIFO (last in, first out). Chaque groupe correspond à un CCGH et est composé de transitions rangées selon leurs seuils décroissants. Ces transitions seront retirées une à une (dans l'ordre croissant) suite aux différents enrichissements de marquage, lorsque c'est au tour du groupe correspondant d'être analysé.

Puis, il reste cinq listes temporaires de transitions à présenter, qui sont complétées, utilisées, puis vidées à chaque itération de l'algorithme :

- *Tfsc* : transitions franchissables sans conflit.
- *Tfcf* : transitions franchissables en conflit *uniquement* avec des transitions franchissables.
- *Tfcpf* : transitions franchissables en conflit avec *au moins une* transition potentiellement franchissable.
- *Tpfsc* : transitions potentiellement franchissables sans conflit.
- *Tpfc* : transitions potentiellement franchissables en conflit (avec au moins une transition franchissable ou potentiellement franchissable).

En fait, ces cinq dernières listes ne contiennent que des transitions internes ou des transitions appelantes. Si une transition appelée intervient parce qu'elle est franchissable ou potentiellement franchissable, l'algorithme fait participer toutes les transitions appelantes qui peuvent appeler cette transition en les classant dans ces listes.

III.1.3 Données de sortie

En sortie de l'algorithme, nous retrouvons un ensemble d'ordres partiels correspondant aux différents scénarios générés. Chaque ordre partiel est défini par un ensemble *(E, E2, A, B, C)* et une liste de jetons d'enrichissement *Le*, où $l = E \cup E2$ (l'ensemble d'événements du scénario) donne l'ensemble des événements du scénario et $A \cup B \cup C$ définit la relation d'ordre partiel stricte $\prec_{sc}$ du scénario.

III.2 Les procédures utilisées

Après avoir présenté les structures de données utilisées par l'algorithme, nous allons maintenant détailler les différentes procédures utiles à l'exécution de ce dernier.

III.2.1 Tirer Transition

Cette procédure permet, comme son nom l'indique, de franchir une transition. Elle met alors à jour la liste courante des jetons lorsqu'elle est appelée, en supprimant les jetons consommés et

en créant les jetons produits. Elle sert également à mémoriser les événements (instances de franchissement de transition) dans *E* et *E2*, ainsi que tous les arcs correspondant à une relation de précédence entre deux événements dans *A*, *B* et *C*.

Un arc relie soit l'événement qui a produit les jetons à l'événement qui consomme ces jetons (ce qui correspond à un *lien direct* défini dans la liste *A*), soit un événement à un autre pour décrire un lien de cause à conséquence (ou *lien indirect*, défini dans la liste *B*), soit une instance de franchissement d'une transition appelante à une instance de franchissement de l'une des transitions appelées (pour la liste *C*). Cette procédure de tir de transition est définie comme suit :

- ***Fire_Transition(tk)*** :

- ***FT0* :** Supprimer de Lint_mc tous les groupes de transitions contenant tk et supprimer de Lint1 les transitions qui sont dans ces groupes.
- ***FT1* :** Ajouter tk dans E.
- ***FT2* :** Pour tout atome (ti,p) nécessaire pour le franchissement de tk, supprimer (ti,p) de la liste Lc ou de Lc_ord et ajouter (ti,tk) dans A. Si l'atome vient de Lc_ord, soit ((e,p),t_{CCGH}) cet atome, ajouter (tk, t_{CCGH}) dans B.
    - *Si tk est une transition appelante, toutes les transitions appelées correspondantes sont franchies aussi :*

      Pour tout atome (ti,p) nécessaire pour le franchissement de tAppelée (ajoutée à E2), supprimer (ti,p) de la liste Lc ou de Lc_ord, ajouter (ti, tAppelée) dans A et ajouter (tk, tAppelée) dans C. Si l'atome vient de Lc_ord, soit ((e,p),t_{CCGH}) cet atome, ajouter (tAppelée, t_{CCGH}) dans B.
- ***FT3*** :

  Si tk (et ses tAppelées) ne consomme que des jetons de Lc : pour toute place pi de sortie de tk : ajouter autant d'atomes (tk, pi) dans Lc.

    - *Si tk est une transition appelante : pour toute place pi de sortie de tAppelée : ajouter autant d'atomes (tAppelée, pi) dans Lc.*

      Sinon, tk (ou ses tAppelées) consomme au moins un jeton de Lc_ord : pour toutes places pi de sortie de tk : ajouter autant d'atomes (tk, pi) dans Lc_int (minimalité CCGH).
    - *Si tk est une transition appelante : pour toute place pi de sortie de tAppelée : ajouter autant d'atomes (tAppelée, pi) dans Lc_int*

III.2.2 Enrichir Marquage

Cette procédure contrôle la cohérence de l'enrichissement de marquage vis-à-vis des invariants de places et autorise cet enrichissement s'il est possible. En fait, deux types de procédures sont à distinguer : la première procédure, appelée *Enrich_Marking1*, opère sur une transition appartenant à l'ensemble Tfcpf et enrichit le marquage de toutes les transitions potentiellement franchissables en conflit avec celle-ci, et ce de la manière suivante :

- ***Enrich_Marking1(tk) :*** *(tk est la transition franchissable du conflit, tj sont les transitions en conflit avec elle)*

Pour toutes les transitions tj en conflit avec tk et pour toutes les places d'entrée vides pl de tj (et pour toutes places d'entrée vides des transitions appelées) :

  - *Si tj est en conflit sans seuil : ajouter le minimum de jetons nécessaires dans Lc et Le.*
  - *Si tj est en conflit CCGH avec ti : ajouter le maximum de jetons possible dans Lc_ord et Le. (ajouter un jeton ((e , p) , tk) dans Lc_ord et Le)*
  - *Si tj ne peut être enrichie : la mettre dans Lint1.*

La deuxième procédure, notée *Enrich_Marking2*, opère sur une transition potentiellement franchissable, afin d'enrichir son marquage si cela est possible. Elle procède de la manière suivante :

- ***Enrich_Marking2(tk)*** *: (tk est la transition potentiellement franchissable à enrichir)*
  - *Pour toutes les places d'entrée pl vides de tk (et pour toutes les places d'entrée vides des transitions appelées) : ajouter le nombre de jetons (ek, pl) nécessaire dans Lc et Le.*
  - *Si tk ne peut être enrichie : la mettre dans Lint1.*

III.2.3 Mémoriser contexte

Lorsqu'un conflit de transitions apparaît et que plusieurs possibilités de tir de transition sont envisageables, il est nécessaire de mémoriser un contexte. Il sera alors possible de revenir à ce contexte par la suite, afin d'explorer les autres possibilités encore non-exploitées.

La mémorisation de contexte se fait par rapport à un premier choix de transition impliquée dans le conflit, pour ne pas s'occuper plusieurs fois de la même possibilité. Un enrichissement de marquage, par un appel à la procédure *Enrich_Marking1* sur *tk*, peut éventuellement s'effectuer si demandé, ceci pour le contexte enregistré. Cette procédure opère de la manière suivante :

- ***Store_context****(tk, boolean avecEnrich_Marking1) :*
  - *ajouter un nouvel élément {E, E2, A, B, C, Le, Lc, Lc_ord, Lc_int, Lint1, Lint_mc, Lint_ord}, qui représente le contexte courant, en ayant ajouter tk dans Lint1 (pour le contexte enregistré seulement) et en ayant fait Enrich_Marking1(tk) si demandé (toujours pour le contexte enregistré).*

III.2.4 Cohérence Marquage

Cette méthode vérifie la cohérence du marquage. Elle fait appel à une liste interne L qui contient des éléments de type (ti, P) qui sont les jetons d'enrichissement; la liste des invariants et la liste de la composante conservative de chaque invariant qui sont des données d'entrée.

- ***Cohérence Marquage*** *(Le, tk) :*
  - *Pour chaque jeton pl de L,*

- *Vérifier pour chaque invariant s'il contient pl*
- *Si le nombre de jetons dans les places constituant cet invariant est supérieur à la composante conservative de cet invariant alors on supprime pl de L.*
- *Si le nombre d'éléments dans L est supérieur à 0 alors l'enrichissement est cohérent, on rajoute les atomes de L dans les listes Lc et Le.*
- *Sinon on rajoute tk à la liste des transitions qui ne peuvent pas être enrichies Lint pour éviter les boucles.*

III.3 Algorithme

Comme nous l'avons vu précédemment, l'algorithme de recherche des scénarios redoutés débute par un raisonnement arrière. Un raisonnement avant est ensuite déroulé à partir de chaque liste de jetons issue du raisonnement arrière. Nous présentons ci-dessous uniquement l'algorithme du raisonnement avant, sachant que celui du raisonnement arrière en découle directement. Pour l'obtenir, il suffit tout simplement de supprimer l'aspect temporel représenté par les seuils des transitions. De cette manière, tous les conflits de transitions correspondront à des conflits sans seuil (puisqu'il n'y a plus de temps).

%Algorithme %

Pas Initial : Étape d'initialisation de C pour la construction du premier ordre partiel

Un nouveau contexte ci est créé. Il contient Lc = Li, E = I = { i1, i2, …} et E2=A=B=C=Le=Lc_ord=Lc_int=Lint1=Lint_mc=Lint_ord={ }

- *ci = { E, E2, A, B, C, Le, Lc, Lc_ord, Lc_int, Lint1, Lint_mc, Lint_ord }*
- *C = {ci}*

Pas 1 : nouvel ordre partiel

Si C est vide : aller à l'étape finale

Sinon :

- *Mémoriser le dernier élément de C dans les différentes listes courantes (E, E2, A, B, C, Le, Lc, Lc_ord, Lc_int, Lint1, Lint_mc, Lint_ord) et le supprimer de C.*
- *Aller en 2*

Pas 2 : Génération de différentes listes de transitions

- *Générer à partir de (Lc, Lc_ord) toutes les listes de transitions, après avoir éliminé :*
  - *Les transitions de E qui ne sont pas dans B (pour éviter les boucles),*
  - *Les transitions sensibilisées par au moins un jeton disponible dans une place appartenant à l'état redouté (label=D) ou une place d'un état défaillant (label=I ou IR),*
  - *Les transitions de Lint1 et de Lint_ord,*

- *Les transitions qui sont sensibilisées uniquement par des jetons dans des places normales qui ne sont pas initiaux (jetons crées suite un franchissement de transition).*

- *Aller en 3*

Pas 3 : Critère d'arrêt

Si toutes les listes différentes de Lint_ord sont vides :

Si Lint_ord n'est pas vide :

- *Retirer la dernière transition de Lint_ord*
- *Aller en 2*

Sinon : Aller en 9

Sinon : Aller en 4

Pas 4 : transition(s) franchissable(s) sans conflit

Si Tfsc est vide : aller à l'étape 5

Sinon :

- *Soit tk la première transition de Tfsc*
- ***Fire_Transition(tk)***
- *Allez en 2*

Pas 5 : transitions franchissables en conflit avec des transitions franchissables

Si Tfcf est vide : aller à l'étape 6

Sinon :

- *Soit tk la transition de plus petit seuilMax de Tfcf*

Si tk est en conflit avec des transitions sans conflit de seuil :

- *ajouter un groupe contenant tk et ces transitions à Lint_mc*
- ***Store_Context(tk, false)***

- *Mettre les transitions en conflit avec tk de seuils supérieurs à celui de tk dans Lint1*
- ***Fire_Transition(tk)***
- *Aller en 2*

Pas 6 : transition(s) franchissable(s) en conflit avec au moins une transition potentiellement franchissable

Si Tfcpf est vide : aller à l'étape 7

Sinon :

- *Soit tk la transition de plus petit seuilMax de Tfcpf*

Si tk est en conflit avec des transitions franchissables ou potentiellement franchissables sans conflit de seuils :

- *Ajouter un groupe contenant tk et toutes ces transitions dans Lint_mc*

Si tk est en conflit avec des transitions franchissables sans conflit de seuils :

- *Store_context(tk, false)*
- *Mettre les transitions franchissables en conflit avec tk dans Lint1*

- *Mettre les transitions franchissables de Tfcpf et les transitions potentiellement franchissables de Tpfc, de seuils supérieurs à celui de tk et en conflit avec tk, dans Lint1.*

Si tk est en conflit avec des transitions potentiellement franchissables de seuils inférieurs :

- *Mettre tk et ces transitions dans Lint_ord*
- ***Enrich_Marking1(tk)**. Les jetons d'enrichissement correspondant aux transitions potentiellement franchissables de seuil inférieur sont mis dans une liste particulière Lc_ord, pour permettre de faire le test de minimalité correspondant au CCGH. Le jetons d'enrichissement correspondant aux transitions sans conflit de seuil avec tk sont mis dans Lc.*
- *Aller en 2*

Sinon :

Si tk est en conflit avec des transitions potentiellement franchissables sans conflit de seuil :

- ***Store_Context(tk, true)**, en ayant fait **Enrich_Marking1(tk)** (tous les jetons dans Lc)*
- ***Fire_Transition(tk)** (travailler avec le contexte avant enrichissement)*
- *Aller en 2*

Pas 7 : transitions potentiellement franchissables sans conflit

Si Tpfsc est vide : aller à l'étape 8

Sinon :

- *Soit tk la première transition de Tpfsc*
- ***Enrich_Marking2(tk)***

*Si **Enrich_Marking2(tk)** n'est pas possible :*

- *Mettre tk dans Lint1*
- *Aller en 2*

Sinon :

- *Fire_Transition(tk)*
- *Aller en 2*

Pas 8 : transitions potentiellement franchissables en conflit

- *Soit tk la transition de plus petit seuilMax de Tpfc*
- *Ajouter un groupe contenant tk et les TPF en conflit avec tk dans Lint_mc*
- ***Store_Context(tk, false)***
- *Enrich_Marking2(tk)*

Si ***Enrich_Marking2(tk)*** *n'est pas possible :*

- *Mettre tk dans Lint1*
- *Aller en 2*

Sinon :

- *Fire_Transition(tk)*
- *Aller en 2*

Pas 9 : construire l'arbre de précédence, relations directes et indirectes

- *Pour tout atome (ti, p) de Lc, Lc_int et Lc_ord :*
  - *ajouter (ti, f) dans A (f : événements finaux)*
  - *ajouter f dans E*
- *Mémoriser l'ordre partiel dérivé*
- *Aller en 1*

Pas final :

Tous les ordres partiels sont dérivés

IV Complétude (état initial minimal et état final minimal)

Dans le cas qui nous intéresse c'est-à-dire le cas où les marquages initial et final ne sont que partiellement connus, la complétude est définie pour un marquage final minimal associé à une coupe minimale et un ensemble de marquages initiaux minimaux partiels (déterminés suite au raisonnement arrière).

La détermination des marquages initiaux minimaux est nécessaire. Si ces marquages initiaux ne sont pas déterminés, la complétude n'a pas de sens. En effet, le nombre de marquages initiaux peut être infini.

Pour chaque coupe minimale associée à un marquage final, suite au raisonnement arrière on détermine un ensemble de marquages initiaux partiels MP_i. Les marquages MP_i sont les marquages initiaux minimaux qui seront considérés dans l'étape du raisonnement avant (états conditionneurs). La définition de la complétude sera donnée entre les marquages initiaux MP_i et le marquage final associé à la coupe minimale.

Raisonnement arrière :

Pour chaque marquage associé à la coupe minimale C_i tel que $M_f = C_i \otimes Cont\_back_i$ ($Cont\_back_i$ est le contexte non spécifié défini progressivement par l'exécution de l'étape de raisonnement arrière), on obtient, suite à l'étape du raisonnement arrière, les marquages initiaux $M_{0j} = MP_j$ avec (*j= 1 à n*).

Les scénarios obtenus sont exprimés comme suit :

$C_i \otimes Cont\_Back_i,\ l_{back} \vdash MP_j$ *j=1 à n* (en considérant le réseau de Petri inversé).

l_{back} est l'ensemble des transitions franchies lors du raisonnement arrière.

Raisonnement avant :

Pour chaque marquage initial MP_j obtenu suite au raisonnement arrière, on détermine un ensemble de scénarios menant de ces marquages au marquage associé à la coupe minimale C_i . Ces scénarios ont la forme suivante :

$MP_j \otimes Cont\_Forw_k,\ l_{av} \vdash C_i \otimes Cont\_Back_i \otimes Cont\_Forw_l$ *k =1 to m* (en considérant le réseau de Petri initial),

où:

$Cont\_Forw_k$ est le contexte nécessaire pour atteindre le marquage final associé à la coupe minimale à partir des marquages initiaux MP_j **.**

$Cont\_Forw_l$ est le contexte obtenu suite au marquage des places associées à l'état redouté.

Remarque : L'ensemble de scénarios complet doit être défini pour chaque coupe minimale en considérant tous les marquages initiaux minimaux.

Définition V.3 (*Ensemble de scénarios complet associé à une coupe minimal*)**:** Soit un réseau de Petri P = (P, T, Pre, Post), soit Ci une coupe minimale associé à un marquage final, soit SC un ensemble de scénarios, l'ensemble SC est complet pour la coupe minimale C_i si et seulement si tout scenario sc_i minimal entre les marquages initiaux $MP_j \otimes Cont\_Forw_k$ et le marquage final $C_i \otimes Cont\_Back_i \otimes Cont\_Forw_l$ appartient à SC.

Définition V.4 (*Ensemble de scénarios complet et minimal associé à une coupe minimale*): Soit un réseau de Petri *P = (P, T, Pre, Post),* soit *Ci* une coupe minimale associée à un marquage final, soit *SC* un ensemble de scénarios, L'ensemble *SC* est complet et minimal pour la coupe minimale C_i si et seulement si l'ensemble SC contient tous les scénarios minimaux et uniquement les scénarios minimaux entre les marquages initiaux $MP_j \otimes Cont\_Forw_k$ et le marquage final $C_i \otimes Cont\_Back_i \otimes Cont\_Forw_l$ appartient à *SC.*

V Exemple d'application

L'exemple choisi pour illustrer le nouvel algorithme de recherche de scénarios redoutés est l'exemple du système de control des trains d'atterrissage d'un avion de type *Rafale* de *Dassault Aviation.* Nous avons choisi cet exemple pour montrer les améliorations apportées à l'algorithme de recherche de scénarios redoutés. En effet cet exemple à déjà été présenté dans [Medjoudj, 2006] avec une approche de modélisation et d'analyse classiques. Nous introduirons donc les aspects orientés objets mais aussi toutes les notions d'enrichissement minimal et d'enrichissement maximal sans oublier la prise en compte de la minimalité présentée au chapitre précédent. Comme nous allons le voir les résultats montrent les apports de la nouvelle version de l'algorithme.

V.1 Description générale

Il s'agit d'un système de contrôle des trains d'atterrissage d'un avion de type *Rafale* de *Dassault Aviation.* Ce système est une version modifiée et simplifiée d'un étalon (benchmark) proposé par le groupe de travail français Systèmes Temps Réel et Qualité de Service [STRQDS, 2002], qui a été présenté par [Boniol & Carcenac, 2002] et étudié par [Villani & al, 2003] dans sa version hybride. Le but est d'appliquer notre approche pour extraire les scénarios redoutés.

Ce système est composé de trois trains d'atterrissage (un train avant, un train gauche et un train droit) qui doivent être en position rentrée pour le vol et en position sortie pour l'atterrissage. Ces trois trains sont interdépendants. Chaque train comporte (figure V.3):

- un boîtier d'accrochage du train en position rentrée.
- une trappe (porte) qui s'ouvre avant la sortie ou la rentrée d'un train et se ferme automatiquement à l'achèvement du mouvement.

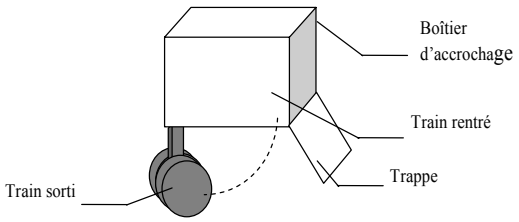


Figure V.3. Train d'atterrissage i

V.2 Fonctionnement du système

Le contrôle des trains est assuré par trois commandes *E*, *R* et *B* :

- la commande *E* réalise en séquence l'ouverture des trappes, l'extension des roues et la fermeture des trappes. Cette séquence est présentée sur la figure V.4
- la commande *R* réalise en séquence l'ouverture des trappes, la rentrée des roues et la fermeture des portes.

- la commande *B* est intermédiaire, dans ce cas, les roues sont bloquées dans leur position courante.

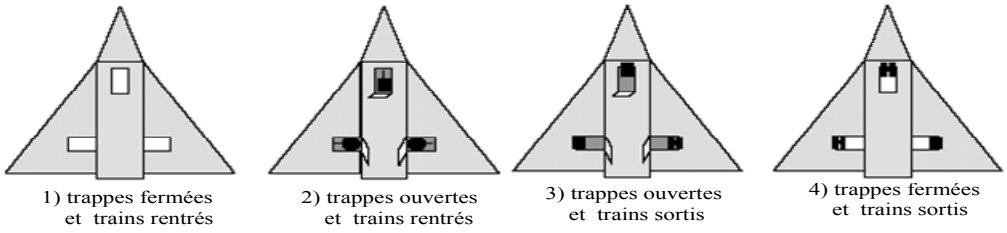


Figure V.4. Séquence d'ouverture des trappes

V.3 Composition du système

Le système est constitué d'un calculateur qui commande trois ensembles train/ trappe. En fonction de la position des trains, trappes (fournie par les capteurs) et de l'état de l'avion, le calculateur établit la commande à réaliser sur le système via des vérins hydrauliques.

Les différents composants du système de contrôle des trains d'atterrissage sont :

- **Un calculateur** qui émet la commande E pour la sortie des trains, la commande R pour la rentrée des trains et une commande de secours qui ne permet que la descente des trains en cas de blocage en sortie. Pour ordonner la sortie ou la rentrée des trains, le pilote dispose dans la cabine d'une palette UP/DOWN de commande à deux positions UP (commande E pour la sortie des trains) et DOWN (commande R pour la rentrée des trains).
- **Trois ensembles train/ trappe** actionnés par des vérins hydrauliques. Chaque vérin d'une trappe est commandé par une électrovanne d'ouverture qui déclenche l'ouverture de la trappe et une électrovanne de fermeture qui déclenche sa fermeture. Chaque trappe ferme un logement d'un train. Chaque vérin d'un train est commandé par une électrovanne de montée qui déclenche la montée d'un train et une électrovanne de descente pour la sortie du train. Les vérins des trains ont deux fonctionnalités : entrée/sortie et verrouillage en position basse.
- **6 électrovannes d'ouverture / fermeture** à chaque trappe sont associées deux électrovannes. Une électrovanne d'ouverture qui déclenche l'ouverture de la trappe et une électrovanne de fermeture qui déclenche sa fermeture.
- **Quatre électrovannes de secours** pour débloquer un train d'atterrissage en sortie en cas de défaillance de l'une des électrovannes spécifiées (blocage de vérin). La première électrovanne de secours (partagée) remplace une des trois électrovannes d'ouverture des trois trappes. La deuxième électrovanne de secours remplace une des trois électrovannes qui commande la sortie des trains. La troisième électrovanne remplace une des trois électrovannes de fermeture des trappes et la quatrième intervient en cas de blocage en rétraction des trains. L'électrovanne de secours ne peut commander qu'un vérin à la fois. Le calculateur commande l'ouverture d'une électrovanne (interrupteur) pour alimenter le vérin

d'une trappe i et commande la fermeture des autres pour que l'électrovanne de secours ne soit utilisée que par une seule trappe à la fois (donner le maximum de pression).

- **Une pompe et un électro-robinet** fournissent l'énergie hydraulique au circuit.
- **Des capteurs de position** indiquant la position des trains et des trappes.
- **Un relais analogique** isolant le calculateur des électrovannes. Le relais reste ouvert un certain temps après le dernier changement de la commande (palette UP/DOWN). Ce temps est supposé suffisant pour la sortie ou la rentrée des trains d'atterrissage. Quand le pilote manœuvre à nouveau la palette UP/DOWN, le relais se ferme. Au sol, il permet l'interdiction de l'excitation de l'électro-robinet par le calculateur et donc la mise en pression du circuit hydraulique pour qu'un mécanicien puisse intervenir sur l'atterrisseur.

V.4 Exigences de sécurité

Si nous faisons le lien avec notre approche d'intégration de la sûreté de fonctionnement dans les processus d'ingénierie système, cette phase correspond au processus E14 (ou E15) de la norme *EIA-632* qui concerne les Exigences d'acquéreur (ou les exigences des parties prenantes).

La définition des exigences de sûreté de fonctionnement et leur prise en compte doit mener à une conception d'un système capable d'éviter des situations dangereuses. Pour la sortie du train on peut définir les exigences suivantes :

- es1 : la porte doit s'ouvrir suite à la commande d'ouverture de la trappe,
- es2 : le train doit sortir suite à la commande E.

Les exigences techniques du système concernent les données de fiabilité associées aux deux exigences.

V.5 Analyse de risque

L'exigence d'analyse de risque (E24) qui est constitué des deux étapes : analyse préliminaire des risques et recherche de scénarios redoutés.

Une analyse des risques nous permet de recenser les états redoutés du système. Nous nous intéressons uniquement aux événements redoutés liés à la sortie du train. Nous avons trois événements redoutés pour chaque train :

- Non-ouverture de la trappe pour la sortie du train,
- Non-sortie du train,
- Non-fermeture de la trappe après la sortie du train.

Les événements redoutés liés à la rentrée des trains peuvent être obtenus d'une façon similaire. Dans cette partie nous ne tenons pas compte des événements redoutés liés à la commande B. Nous ne décrivons pas la commande *B* ainsi que les événements redoutés qui lui sont associés.

La deuxième étape consistant à générer les scénarios redoutés sera abordée plus tard.

V.6 Modélisation du système et solution logique

Afin de prendre en compte les exigences fonctionnelles et non fonctionnelles, une modélisation par réseau de Petri est proposée. La modélisation prend en compte les exigences élicitées. Concernant les exigences de sûreté précédentes, elles seront modélisées par les défaillances et les reconfigurations dans le réseau de Petri. La détermination du réseau de Petri modélisant le système rentre dans la démarche globale exposée dans le chapitre 3. Cette étape correspond à l'exigence de définition de la solution logique (E17) de la norme *EIA-632*.

V.6.1 Diagramme de cas d'utilisation

La figure V.5 montre le diagramme UML des cas d'utilisation relatif à l'ouverture de la trappe.

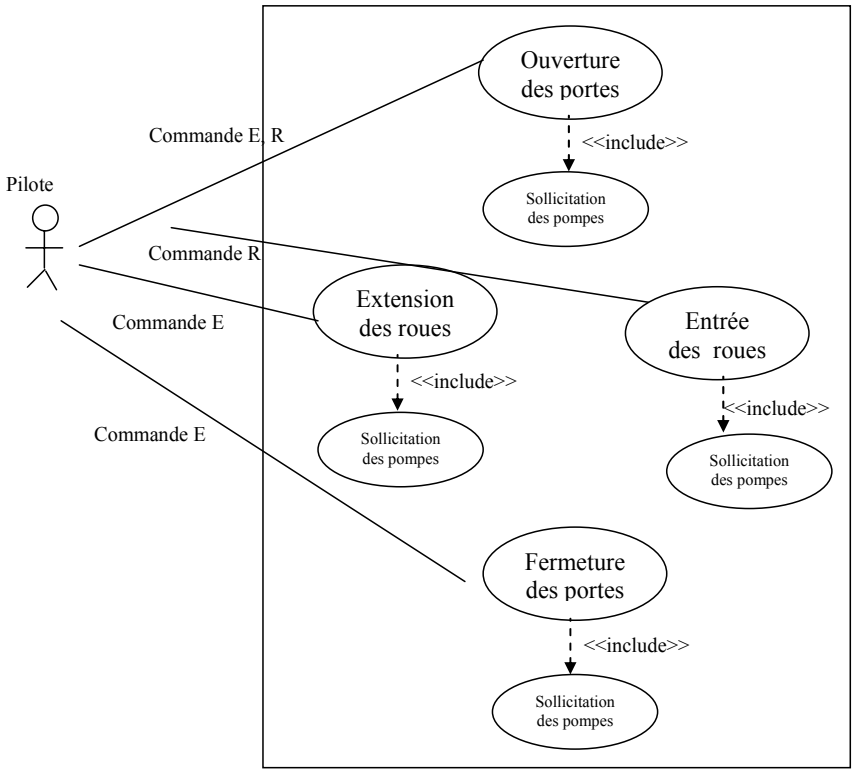


Figure V.5. Digramme des cas d'utilisation UML

V.6.2 Diagramme de collaboration

Un scénario décrit un cas d'utilisation. Le digramme de collaboration représente ce scénario. Le diagramme de collaboration de la figure V.6 montre les différentes interactions des composants des systèmes, en l'occurrence les trappes et les électrovannes. Le diagramme montre les différents appels et offres de méthode entre eux.

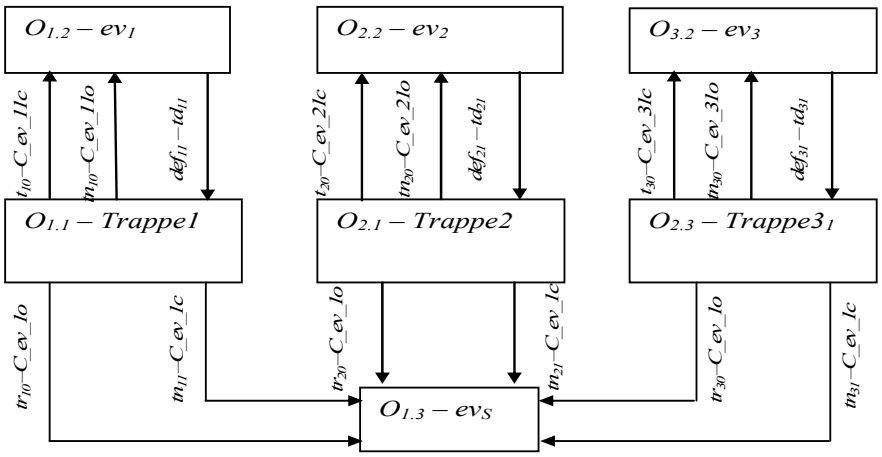


Figure V.6. Diagramme de collaboration

V.6.3 Classe du système

Nous allons considérer uniquement le processus d'ouverture des trappes. En ce qui concerne les défaillances des différents composants du système, nous supposons que le calculateur, la pompe, l'électro-robinet et les capteurs assurent leur fonction et ne sont pas défaillants. Ils ne seront donc pas modélisés. Dans cet exemple nous considérons uniquement les défaillances des électrovannes qui commandent les différents vérins des trappes et des trains.

Deux types de composants nous intéressent pour notre étude : les trappes et les électrovannes. Nous proposons 3 classes (figure V.7) pour la modélisation du système. Une classe trappes et deux classes électrovanne. En effet, la première classe électrovanne est réparable et la seconde est définitivement hors service en cas de défaillance. Nous définissons trois objets (trappe 1, trappe 2 et trappe 3) instances de la classe trappes et 3 électrovannes instance de la première classe électrovannes et comme on ne s'intéresse qu'à l'ouverture de la trappe, on ne considérera qu'une seule électrovanne de secours instance de la deuxième classe électrovanne.

Classe C_1 - Trappe	Classe C_2 - Electrovanne	Classe C_2 – Electrovanne de secours
t1 : temps d'ouverture de la trappe	t2 : temps d'ouverture de l'électrovanne	t3 : temps d'ouverture de l'électrovanne de secours
Demande d'ouverture Demande de fermeture	Ouverture Fermeture	Ouverture Fermeture

Figure V.7. Méthodes et attributs des classes du système.

Nous rappelons que dans la représentation graphique du modèle RdP PTD d'une classe, les transitions représentées par un rectangle blanc sont des méthodes offertes par la classe, celles représentées par un rectangle noir sont des appels à méthodes et celles représentées par une ligne sont des transitions internes.

V.6.4 Classe trappe

La figure V.8 montre le modèle réseau de Petri des 3 objets de la classe trappe. Nous décrivons uniquement l'objet trappe 1 de la classe, les deux autre objets (trappe 2 et trappe 3) étant similaires. Dans l'objet 1 de la classe qui représente la première trappe, la place PS_{10} représente l'état où la trappe est dans l'état fermée. La transition tn_{10} (transition partagée) est un appel à méthode qui représente la demande d'ouverture de l'électrovanne ev_1, méthode offerte par l'objet ev_1 (figure V.9) et représentée par la transition *C_ev_11o*. Si l'appel à méthode est satisfait par l'exécution de la méthode, le processus d'ouverture de la trappe commence (marquage de la place P_{10}). Dans ce cas, la place P_{10} correspond à l'ouverture de la porte de la trappe. Sa dynamique continue est prise en compte par une abstraction temporelle. La durée minimale pour l'ouverture de la trappe d'un train *i* est 10 secondes et sa durée maximale est 12 secondes. Le temps correspondant à la contrainte temporelle est représenté par l'intervalle [10, 12] associé à la transition t_{10}. Comme il n y a pas d'autres places d'entrée de la transition t_{10}, ceci signifie que le jeton doit rester dans la place p_{10} au moins 10 secondes et au plus 12 secondes avant que cette transition soit franchie (marquage de la place p_{11} correspondant à la position trappe ouverte). Suite au franchissement de la transition t_{10} qui est une transition partagée, l'électrovanne ev_1 est libérée.

Le temps de réponse de l'électrovanne ev_1 est 2 secondes d'où l'intervalle temporel [0,2[associé à la transition tn_{10}. Ceci signifie qu'elle est défaillante s'il y a absence de mouvement de l'ouverture de la trappe *1* après l'excitation de l'ouverture des trappes pendant deux secondes. Dans ce cas, l'appel à méthode n'est donc pas satisfait. Le calculateur demande alors l'utilisation de l'électrovanne de secours ev_S avec un temps de réponse de 4 secondes représenté par l'intervalle temporel [2,6[associé à la transition tr_{10} qui représente l'appel à méthode pour l'ouverture de l'électrovanne de secours ev_S (transition *C_ev_1o*). Si cette électrovanne est en bon fonctionnement, la transition tr_{10} est franchie et la place ps_{11} (représentant une reconfiguration du système) marquée. Comme il n'y a aucune autre transition de sortie qui peut consommer ce jeton, il ne peut rester dans ps_{11} plus de 12 secondes. Une fois le processus d'ouverture fini, l'appel à méthode représenté par la transition tn_{11} correspond à la demande de fermeture de l'électrovanne de secours ev_S, méthode offerte par l'objet de classe 3 (électrovanne de secours) représentée par la transition *c_ev_1c*. L'électrovanne de secours est alors libérée (marquage de la place ev_{1_f}) par le franchissement de la transition tn_{11}.

Si l'électrovanne ev_1 est défaillante et l'électrovanne de secours ev_S est hors service ou occupée par une des deux autres trappes, alors la première transition red_{11} menant vers l'état redouté est franchie 6 secondes après le marquage de ps_{10}. Nous avons alors le marquage de la place E_red_{11}.

La dynamique de la place p_{10} peut être interrompue par le franchissement de la transition immédiate td_{11} suite à la défaillance de l'électrovanne ev_1 (franchissement de la transition def_{11}_o de l'objet $ev_{\_11}$) après le début de l'ouverture de la trappe. Ceci correspond à un blocage de la trappe en ouverture représenté par le marquage de la place ps_{12}. Dans ce cas, l'électrovanne de secours ev_S est sollicitée. Si elle n'est pas défaillante ou occupée par une des deux autres trappes, la transition tr_{11} correspondant à une reconfiguration après blocage est franchie. Sinon la deuxième transition red_{12} menant vers l'état redouté est franchie.

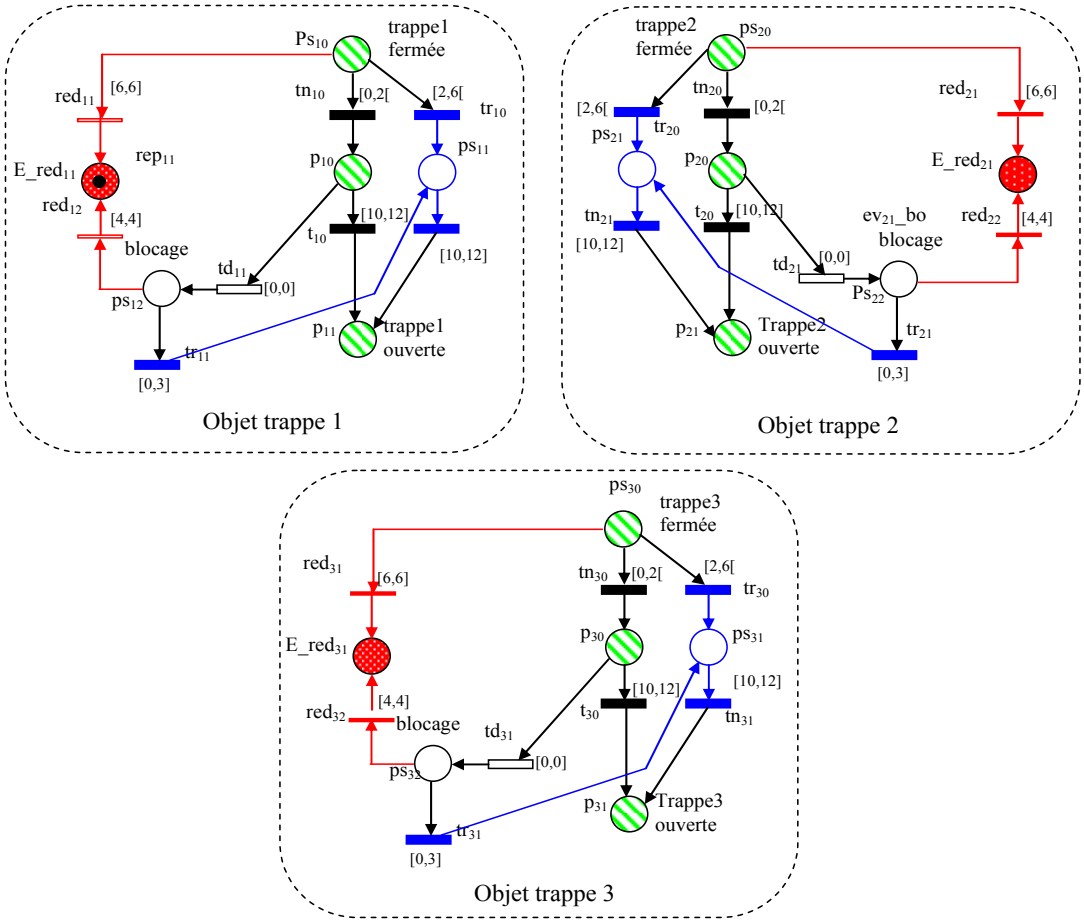


Figure V.8. Modèles réseaux de Petri des différentes instances de la classe trappe

V.6.5 Classe électrovanne

Nous décrivons le modèle de l'électrovanne ev_1 (figure V.9) qui est associée à la trappe 1. Les modèles des autres électrovannes ev_2 et ev_3 sont identiques.

L'électrovanne ev_1, quand elle est ouverte (marquage de la place ev_{11_o}), permet l'ouverture de la trappe 1. L'ouverture se fait suite à l'appel effectué par l'objet trappe 1 et le franchissement de la transition $C\_ev_{11}\_o$. L'électrovanne est fermée une fois le processus d'ouverture de la trappe terminé, suite au franchissement de la transition $C\_ev_{11}\_f$.

En fonctionnement normal, les deux méthodes offertes par ev_1 garantissent l'ouverture de la trappe 1 et la libération de l'électrovanne ev_1 une fois que la trappe est ouverte. Les défaillances de l'électrovanne ev_1 sont représentées par le tir des transitions def_{11}_o (blocage en ouverture) ou def_{11}_f (blocage en fermeture). La transition def_{11}_o est une transition partagée avec l'objet trappe 1. En effet, le partage de la transition permet de transmettre l'information à la trappe 1 sur la défaillance de l'électrovanne ev_1 interrompant ainsi le processus d'ouverture de la trappe

si celui est en cours d'exécution. Les transitions rep_{11}_o et rep_{11}_f représentent la réparation de l'électrovanne ev_1.

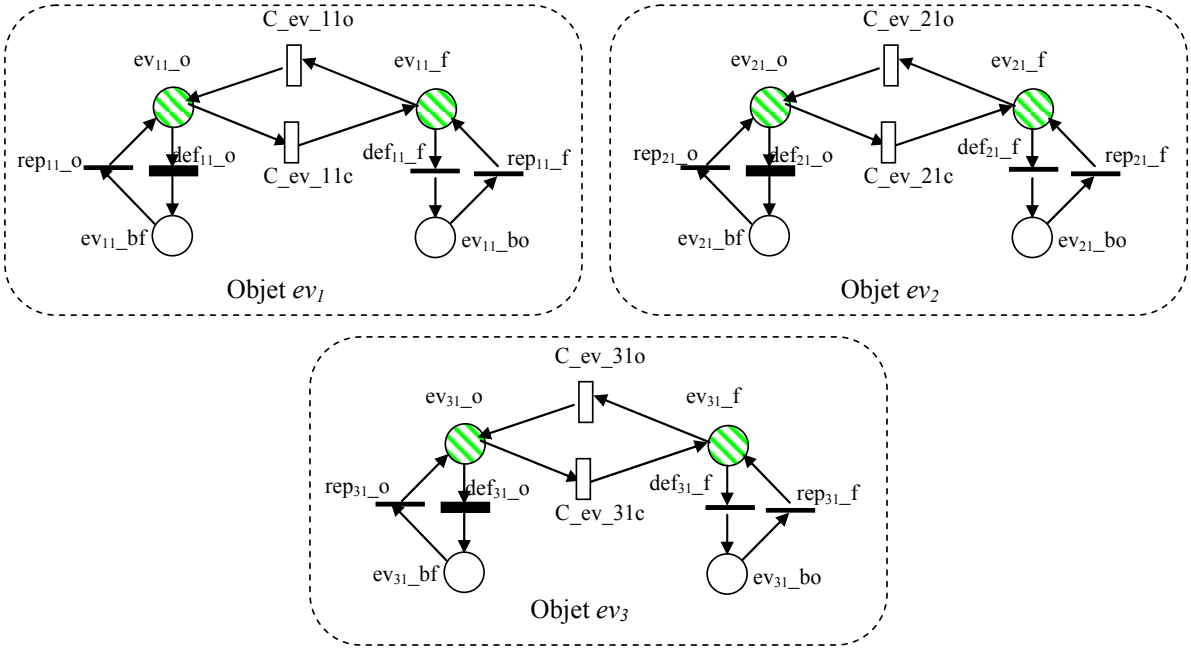


Figure V.9. Modèle des instances de la classe électrovanne.

V.6.6 Classe électrovanne de secours

L'électrovanne de secours ev_S est partagée par les trois trappes. Son modèle est présenté sur la (figure V.10). La place *ev1_o* représente la disponibilité de l'électrovanne de secours. La transition *C_ev_1o* représente la méthode '*ouverture* ev_S' offerte par ev_S, qui peut être appelée par une des trois trappes. La transition *C_ev_1c* représente la méthode fermeture ev_S' offerte par l'électrovanne de secours ev_S, appelée par une des trois trappes dépasse une fois le processus d'ouverture terminé pour la libérer. L'électrovanne *ev1* peut tomber en panne (franchissement de la transition def1_o ou la transition def1_f). Dans ce cas, la place *ev1_hs* est marquée et l'électrovanne de secours ev_S est définitivement hors service.

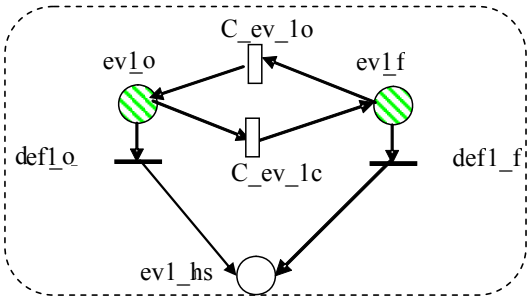


Figure V.10. Modèle de la classe électrovanne de secours ev_S

V.7 Application de la méthode

Nous allons maintenant appliquer le nouvel algorithme de génération de scénarios sur l'exemple des trappes. La recherche de scénarios rentre dans le cadre du processus (exigence d'analyse de risque E24) de la norme *EIA-632*. Il s'agit de la deuxième étape du processus d'analyse de risque

Nous ne présenterons pas les différentes étapes et procédure de l'algorithme dans les détails. Nous allons juste illustrer la génération d'un scénario, la procédure étant la même pour les autres scénarios générés.

V.7.1 Détermination des états normaux

Dans les différents réseaux de Petri modélisant les différentes classes, les places avec rayées représentent les états normaux du système.

V.7.2 Détermination des états cibles

Les états redoutés recensés correspondent aux marquages des places *E_redi1* (*i*=1, 2 ou 3) des objets trappe de la classe trappe. La formule booléenne associée associé à l'état redouté est : $R = B(E\_red_{11}) \vee B(E\_red_{21}) \vee B(E\_red_{31})$. Trois coupes minimales sont associées à cette fonction booléenne :

$C_1 = B(E\_red_{11})$.

$C_2 = B(E\_red_{21})$.

$C_3 = B(E\_red_{31})$.

On ne s'intéressera qu'à la première coupe minimale ($C_1 = B(E\_red_{11})$), la symétrie du système permet de déduire les scénarios associés aux deux autres coupes minimales ($C_2 = B(E\_red_{21})$ et $C_3 = B(E\_red_{31})$).

V.7.3 Raisonnement arrière

L'analyse se fait en considérant les réseaux de Petri inversés des différents objets. Dans un premier temps, seul l'objet trappe 1 est concerné (il contient le marquage associé à la coupe minimale C_1). On considère donc comme marquage initial du raisonnement arrière le marquage de la place E_red_{11}.

Avec le marquage de la place E_red_{11} deux transitions red_{11} et red_{12} sont franchissables. Si on considère par exemple la transition red_{11} (la transition red_{12} sera considérée dans un autre temps afin de générer tout les scénarios menant à cet état redouté grâce à la procédure de mémorisation de contexte) son franchissement produit un jeton dans la place P_{s10}. Cette place étant une place normale, le raisonnement arrière s'arrête avec le marquage de la place P_{s10} de

l'objet trappe 1, comme état conditionneur. Le marquage de la place P_{s10} sera considéré comme marquage initial de l'étape du raisonnement avant.

V.7.4 Raisonnement avant

Le raisonnement commence par le marquage correspondant à un jeton dans la place P_{s10} (objet trappe 1). Les transitions t_{n10}, t_{r10} et red_{11} sont alors en conflit chien de garde. En effet la transition red_{11} est franchissable les deux autres transitions (t_{n10} et t_{r10}) sont potentiellement franchissables et le seuil associé à la transition t_{n10} (intervalle [0, 2[) est inférieur au seuil de la transition t_{r10} (intervalle [2, 6[)) qui est lui-même inférieur à celui de la transition red_{11} (intervalle [6, 6]). Suite à ce conflit mettant en jeu des transitions partagées (t_{n10} et t_{r10}), les objets partageant ces transitions sont introduits (le diagramme de collaboration montre clairement l'interaction entre l'objet trappe 1 et les objets, *ev1* et ev_S). L'introduction des objets ev_1 et ev_S s'effectue avec un enrichissement de marquage maximal des places d'entrée des transitions *E_ev_11o* et *C_ev_1o*. Ces transitions qui représentent les offres de méthode appelées par l'objet trappe 1 via les transitions t_{n10} et t_{r10}. Les deux places à enrichir ev_{11}_f et ev_1_f appartiennent aux invariants de places suivants :

- $M(ev_{11}\_f) + M(ev_{11}\_o) + M(ev_{11}\_bo) + M(ev_{11}\_bf) = 1$
- $M(ev_1\_f) + M(ev_1\_o) + M(ev_1\_hs) = 1$

L'enrichissement maximal consiste donc à introduire un jeton dans chacune des places ev_{11}_f et ev_1_f. Les événements minimaux qui consomment les jetons d'enrichissement sont les tirs de la transition def_{11}_f pour l'objet ev_1 et def_1_f pour l'objet ev_S. Suite au franchissement de ces deux transitions, la transition red_{11} est franchie menant ainsi vers l'état redouté.

Scenario1: défaillance de l'électrovanne ev_1, défaillance de électrovanne de secours ev_S, suivi du tir de la transition red_{11}.

Dix scénarios redoutés menant vers la marquage de la place *E_red11* ont été générés. La figure V.11 représente un scénario. Les 9 autres scénarios se trouvent en Annexe.

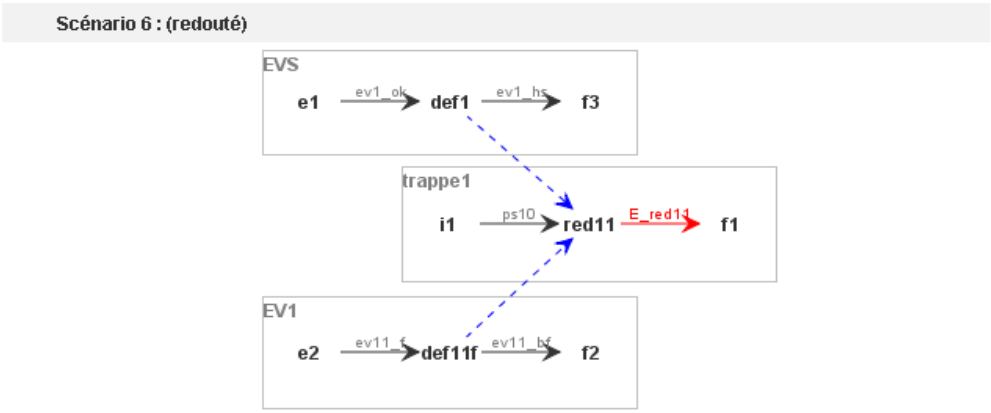


Figure V.11. Exemple de scénario redouté de l'ouverture de la trappe1

Le graphe de précédence de la figure IV.11 fait clairement apparaître les différents objets qui interviennent dans les scénarios, ainsi que les liens entre ces objets. Nous retrouvons l'aspect orienté objets des réseaux de Petri du côté des graphes de précédence, qui facilite leurs lectures et leurs analyses.

Tous les scénarios générés sont minimaux. En effet, l'introduction de la définition de la minimalité nous a permis de générer uniquement les scénarios minimaux menant à l'état redouté.

Pour résumer l'information fournie par les graphes, la trappe 1 se bloque lorsque son électrovanne ev_1 est défaillante et lorsque l'électrovanne de secours ev_S est soit défaillante, soit déjà utilisée par les autres trappes. Cette dernière pouvant être réquisitionnée par la trappe 2 ou la trappe 3 si leurs électrovannes respectives sont défaillantes. A toutes ces possibilités, il faut également ajouter le fait que la défaillance d'une électrovanne peut survenir avant le début de l'ouverture de la trappe ou pendant son ouverture.

Le processus de validation décrit dans la norme *EIA-632* à travers les exigences E24, E25, E28, passe par une analyse des scénarios générés et une quantification des probabilités d'apparition de ces derniers (l'analyse quantitative n'est pas abordée dans cette thèse). Dans le cas où les exigences de sécurité ne sont pas respectées, des solutions doivent être proposées. Suite aux solutions proposées par le concepteur, une nouvelle procédure de recherche de scénarios redoutés est effectuée. La quantification des probabilités d'apparition des nouveaux scénarios générés permet de valider le système d'un point de vue sûreté de fonctionnement s'il répond aux exigences de sûreté et de sécurité.

Remarque

La nouvelle version de l'algorithme et de l'outil qui l'implémente génère les 10 scénarios redoutés minimaux qui mènent vers l'état redouté. La première version de l'algorithme [Khalfaoui, 2003] et [Medjoudj, 2006] générait quant à elle 87 scénarios certains de ces 87 scénarios sont incohérents avec la dynamique continue du système. En effet, ni la minimalité ni l'aspect continu n'était pris en compte. Dans la deuxième version de l'algorithme [Medjoudj, 2006], prenant en compte la dynamique continue par des abstractions temporelles, 21 scénarios ont été générés mais tous ne sont pas minimaux.

VI Conclusion

Au cours de ce chapitre, nous avons présenté la nouvelle version de la méthode de recherche de scénarios redoutés. Elle est basée sur la recherche de cause à effet à partir du modèle réseau de Petri du système et permet de caractériser les scénarios redoutés sous la forme d'un ensemble de relations d'ordre entre les événements menant d'un état de bon fonctionnement à un état redouté.

Nous avons introduit les nouvelles notions développées au cours de cette thèse afin de formaliser les relations de précédence indirecte.

Nous avons aussi introduit les concepts orientés objets, afin de faciliter la modélisation de systèmes complexes et faciliter l'analyse des scénarios générés.

L'autre aspect important est la minimalité des scénarios. En effet l'algorithme dans sa nouvelle version permet de générer uniquement les scénarios minimaux.

L'exemple traité dans ce chapitre a permis de monter tous ses apports de la nouvelle version de l'algorithme.

Pour le moment l'algorithme opère sur un modèle réseau de Petri temporel. La partie continue est prise en compte par l'intermédiaire d'abstractions temporelles. Le chapitre suivant propose une approche de simulation hybride qui permet de prendre en compte cette dynamique continue.

Chapitre 6 : Algorithme hybride pour la génération de scénarios redoutés

I Introduction

Jusqu'à maintenant, nous n'avions pas pris en compte que des cas simple de dynamique continues : des cas où les variables continues associées au système sont linéaires. Dans ce cas et grâce à des abstractions temporelles de cette dynamique, un réseau de Petri temporel peut suffire. La recherche des scénarios est ensuite effectuée sur le modèle réseau de Petri temporel.

L'abstraction temporelle consiste à définir une approximation la dynamique hybride. Les seuils continus associés aux différentes transitions sont transformés en des durées qui correspondent au temps que met le système pour les atteindre quand les transitions sont sensibilisées. L'approche consiste à résoudre le système d'équations associé à la place afin de déterminer les conditions de sortie de l'état représenté par le marquage de la place en question sous la forme de seuils temporels. Grâce à ces seuils, un ordre de franchissement des transitions est déterminé et permet ainsi de caractériser un scénario donné.

En plus de la limitation de l'approche à certains types de dynamiques continues, les intervalles de temps déterminés ne sont pas dynamiques, mais statiques. En effet ces intervalles sont déterminés en amont de l'étape d'analyse. Il serait plus intéressant de les déterminer au fur et à mesure de l'évolution du système.

Dans ce qui suit nous proposons une extension de l'algorithme de recherche de scénarios redoutés aux cas des réseaux de *Petri prédicats transitions différentiels*. Algorithme qui permet de prendre en compte la dynamique continue du système sans approximation. En effet, comme nous allons le voir dans ce chapitre, les approches consistant à approximer la dynamique continue ne sont adaptées pour l'analyse des systèmes complexes. Le principe consiste à coupler l'algorithme de génération de scénarios redoutés avec un solveur d'équations différentielles.

Avant cela, nous allons présenter les principales approches d'abstraction de la dynamique continue, ainsi que leurs limites.

II Approximation de la partie continue

Dans la littérature plusieurs travaux ont abordé le problème de la correspondance d'un modèle discret avec un modèle continu [Travé-Massuyès & al, 1997]. Struss [Struss, 2002] s'est intéressé à la génération automatique d'un modèle qualitatif à partir d'un modèle numérique de simulation. Trois difficultés principales sont soulignées:

- lors d'une abstraction automatique d'un modèle, l'un des risques est l'explosion du nombre d'états discrets.
- lorsqu'un modèle comporte un grand nombre de variables intermédiaires, des erreurs d'approximation peuvent être accumulées.
- la détermination des seuils pour la définition des variables qualitatives qui s'établit entre la connaissance experte du système et les conséquences de ces seuils sur le résultat fait entièrement partie du travail de modélisation et peut être lourd.

De nombreux travaux s'intéressent à l'abstraction d'un système continu (ou hybride) en un système à événements discrets à des fin de contrôle du système [Caines & Wei, 1998], [Cury & al., 1998], [Naud, 2003], [Nerode & Kohn, 1993], [Pappas & Sastry, 1996], [Stiver et al., 1996]. Dans tous ces travaux le paramètre temps n'est pas pris en considération [Hélias, 2003].

II.1 Linéarisation d'automates hybrides.

Face à l'impossibilité d'utiliser les automates hybrides non linéaires à des fins d'étude des systèmes, Henzinger [Henzinger et al., 1998a] propose deux méthodes permettant d'approximer ce type de modèle par des automates hybrides linéaires, pour lesquels des outils d'analyse automatique existent (Hytech [Henzinger & al., 1997]).

En ce qui concerne la première méthode (i.e., "clock translation"), il précise que pour certains automates non linéaires, il est possible de substituer les variables continues par des horloges et obtenir ainsi un automate temporisé (un automate temporisé est un automate hybride linéaire où toutes les dérivées sont égales à un). Pour qu'une variable puisse être remplacée, il est en effet nécessaire que :

- elle soit indépendante,
- pour tout sommet de l'automate, son état initial soit connu et sa dynamique monotone,
- lors d'une transition entre sommets, si l'arc ne comporte aucune condition sur la variable, sa dynamique doit être identique dans les deux sommets (la même équation).

L'ensemble des constantes des contraintes de l'automate liées à cette variable est ensuite déterminé en fonction de l'équation associée à la variable.

La seconde méthode (i.e., "phase-portrait approximation") est applicable sur l'ensemble des automates hybrides non linéaires mais il s'agit d'une procédure qui reste manuelle. Le principe est de simuler l'automate hybride en « relâchant » les activités, les contraintes liées aux gardes et aux invariants, les sauts, les états initiaux ainsi que les événements pour obtenir un automate hybride linéaire de telle sorte que le comportement de celui-ci représente l'automate initial. En d'autres termes, il s'agit d'encadrer les valeurs des dérivées entre deux enveloppes sur des fenêtres temporelles pour linéariser le système.

II.2La méthode d'abstraction de Hélias

Une méthode d'abstraction qualitative des dynamiques complexes, avec une connaissance partielle par une représentation discrète a été proposée par Arnaud Hélias dans sa thèse pour un traitement des effluents d'élevage, notamment le processus de digestion anaérobique [Hélias & al, 2002], [Hélias, 2003], [Hélias & al, 2004]. Face au caractère hybride de ce type de système l'approche de Hélias consiste à trouver une forme de représentation homogène à partir d'une connaissance continue (des relations analytiques) et une connaissance issue d'une expertise discrète (des seuils de commutation de classes et des intervalles). Il propose l'abstraction de l'espace d'état d'un système continu non linéaire par des intervalles sur les valeurs d'entrée de façon à obtenir un modèle de prédiction basé sur un automate temporisé. La définition d'un partitionnement issu d'une expertise et la simulation des relations continues imprécises (intervalles sur les trajectoires) permettent l'obtention des intervalles de temps des transitions d'état $[\tau_{min}\ \tau_{max}]$ pour chaque variable représentée par un automate temporisé (figure VI.1).

Les relations continues imprécises sont décrites par les équations d'état $\xi' = f(\xi,\zeta)$, $\xi(t_0) = \xi_0$ et de sortie $\zeta = g(p,t)$ (p un ensemble de paramètres), avec une connaissance partielle sur les entrées et l'état initial définie par des intervalles, $\forall i,\ \xi_i^-(t_0) \le \xi_i(t_0) \le \xi_i^+(t_0)$ et $\forall j,\ \forall t,\ \zeta_j^-(t_0) \le \xi_j(t_0) \le \xi_j^+(t_0)$ (figure VI.1). Ainsi la double équation différentielle ordinaire (EDO) permet d'estimer les intervalles $[\xi_i^-, \xi_i^+]$ avec l'hypothèse $\forall t,\ \forall i,\ \xi_i^-(t) \le \xi_i(t) \le \xi_j^+(t)$. Les fenêtres temporelles $[\tau_{min}, \tau_{max}]$ sont alors obtenues lors du franchissement d'un seuil v auquel est affectée une étiquette précisant le sens de franchissement (signe de la dérivée) : $\Im$ (v) : $\tau_{min} = \tau^+ \Rightarrow$ ''v^{Δ}'' et $\tau_{min} = \tau^- \Rightarrow$ ''v^{∇}''.

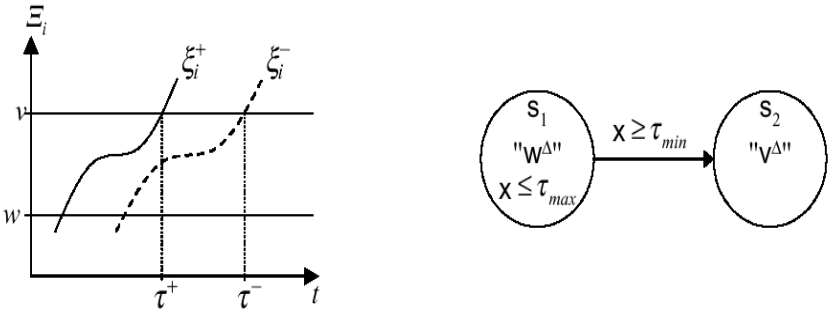


Figure VI.1. L'automate temporisé de représentation de la trajectoire du seuil w vers v

L'espace d'état qualitatif du procédé est reconstruit à travers le mécanisme de composition synchrone d'automates [Lafortune & Chen, 1991], chaque automate représentant les trajectoires qualitatives d'une des variables.

L'analyse et le diagnostic du procédé sont fondés sur le mécanisme de model-checking et les outils Kronos.

Ce que nous pouvons noter par rapport à cette approche basée sur une représentation dans l'espace d'état et la composition synchrone d'automates, c'est qu'elle amène toujours au problème d'explosion combinatoire. La taille de l'automate final est fonction de la dimension du système continu, du nombre de seuils et du nombre de franchissements de seuils.

II.3Abstraction des dynamiques continues par réseau de Petri temporel flou

Dans son travail de thèse Loures [Loures, 2006] s'est intéressé à la surveillance et la supervision des installations industrielles. Devant la complexité en raison de leur caractère hybride des systèmes mis en jeu et la complexité des relations entre variables, il note qu'il n'est pas toujours facile et forcément nécessaire d'avoir une modélisation précise de la dynamique du procédé. Pour cela, Loures propose une approche qualitative qui permet une représentation avec un degré d'abstraction plus en adéquation avec le niveau haut de surveillance considéré. Il propose une abstraction des dynamiques continues basée sur un raisonnement temporel et événementiel. Cette abstraction est basée sur un partitionnement temporel flou de la dynamique des variables importantes définissant ainsi un ensemble d'états qualitatifs. Des mécanismes de vérification et de rétablissement de cohérence temporelle entre les variables sont proposés de façon à décrire les relations dynamiques locales existantes. Pour son pouvoir de représentation et pour rester cohérent avec une approche hiérarchique basée réseau de Petri, les Réseaux de Petri Temporels Flous [Cardoso, 1999] ont été choisis comme modèle.

III Approche Hybride de génération de scénarios redoutés

Devant la difficulté et les limitations des approches consistant à approximer la dynamique continue des systèmes hybrides pour des fins d'analyse, la simulation constitue une approche alternative. Notre approche de simulation se déroule en deux étapes.

Une première étape consiste à effectuer une simulation Monte Carlo sur un intervalle de temps suffisamment grand limitée en nombre d'histoires. L'analyse par la simulation Monte Carlo est purement qualitative. Par le mot qualitatif nous entendons qu'aucune estimation de probabilité n'est effectuée. En effet, l'objectif de cette première étape est de déterminer les domaines des variables continues et particulièrement les domaines des valeurs initiales des variables continues.

L'algorithme de génération de scénarios redouté peut être vu comme un simulateur de réseaux de Petri basée sur la logique linéaire. La deuxième étape, est donc une simulation hybride basée sur l'algorithme de génération de scénarios redoutés couplé avec un solveur d'équations

différentielles. Elle a pour objectif de déterminer les scénarios redoutés, en se basant sur les domaines des variables continues déterminés dans la première étape.

La première étape constitue donc un travail préliminaire pour la deuxième étape. Comme nous allons le voir par la suite, à cause du caractère local de la simulation hybride basée sur l'algorithme de génération de scénarios, les domaines des variables continues ne sont pas toujours disponibles. En effet, le marquage dans lequel évolue l'algorithme n'est que partiel. Comme nous l'avons vu dans le chapitre précédent, la procédure d'enrichissement de marquage permet de prendre en compte de nouveaux états. Le problème qui se pose alors est l'absence d'information sur les valeurs initiales des variables continues associées aux places enrichies, d'où la nécessité de les déterminer au préalable grâce à la première étape.

III.1 Première étape (simulation de Monte Carlo)

Le principe de la simulation de Monte Carlo consiste à étudier l'évolution d'un système en simulant un modèle générique, représentant le comportement du système, au cours de ce qu'on appelle une histoire. La quantification de la grandeur recherchée est alors basée sur l'étude d'un certain nombre de scénarios différents, permettant d'en extraire des résultats statistiques.

Dans le cadre de la sûreté de fonctionnement, le modèle simulé est régi par des événements très rares (les défaillances) et des événements très fréquents (fonctionnement nominal du système), et ce simultanément. La simulation est alors cadencée par de nombreuses occurrences d'événements fréquents qui ne reflètent pas le comportement du système en présence de défaillances. C'est le problème de simulation des événements rares. C'est ce que nous avons vu au premier chapitre. Un nombre important d'histoires est nécessaire pour voir apparaître un événement redouté, impliquant des temps de calcul importants. Ces temps deviennent faramineux si, en plus, l'on souhaite obtenir un intervalle de confiance acceptable.

Dans notre cas, la simulation est effectuée avec un nombre d'histoire limité pour éviter des temps de simulation importants. L'objectif étant de déterminer les domaines des variables continues, notamment les domaines des valeurs initiales des variables continues.

Nous avons donc implémenté la simulation Monte Carlo des réseaux de Petri Prédicat transition différentiels stochastiques.

Nous avons opté pour une simulation événement par événement qui se prête mieux à la simulation des réseaux de Petri. La simulation est réalisée selon l'algorithme suivant :

- Pour toutes les transitions sensibilisées :
  - Les dates de tir des transitions stochastiques sont tirées au sort,
  - Les dates de tir des transitions déterministes sont calculées suivant la spécification du système.
- Ces temps sont rangés dans un calendrier par ordre croissant, et le minimum est sélectionné. C'est celui-ci qui détermine l'instant et la nature de la prochaine transition subie par le système complet.

Dans le cas où les réseaux de Petri sont le support du Monte Carlo, l'agenda des événements est constitué des délais jusqu'au tir des différentes transitions validées à un instant donné par le marquage du réseau.

La simulation permet de déterminer les domaines des variables continues et notamment les domaines des valeurs initiales des variables continues. En effet lors du parcours des différents états du système, les valeurs initiales des variables continues sont déterminées grâce aux fonctions de jonctions associées au réseau de *Petri Prédicats transitions différentiel*. Ces différentes valeurs sont mémorisées sous forme d'intervalles.

Comme nous allons le voir, dans la deuxième étape qui correspond à l'exécution de l'algorithme de génération de scénarios dans sa version hybride, la simulation n'est pas globale, mais locale, concentrée sur les parties du système intéressantes d'un point de vue fiabilité dynamique. Les conditions initiales des variables continues ne sont pas toujours disponibles. Grâce à la première simulation, les domaines des variables continues déterminés seront exploités par l'algorithme de recherche de scénarios dans sa version hybride.

Les domaines des variables continues, peuvent être des constantes ou des intervalles. Dans le cas d'intervalles, il existe plusieurs méthodes de résolution d'équations différentielles avec des valeurs initiales sous forme d'intervalles. Ceci permet d'éviter de faire la résolution (double résolution) avec comme valeurs initiales les bornes de l'intervalle.

III.2 Deuxième étape (algorithme hybride de recherche de scénarios redoutés)

Le principal problème auquel on est confronté lorsqu'on effectue une recherche de scénarios redoutés sur un modèle réseau de *Petri prédicats transitions différentiel*, est la détermination de l'ordre de franchissement entre les transitions. En effet, à la rencontre d'un conflit de transitions, à partir des équations associées aux places d'entrée des transitions concernées par le conflit et des fonctions de sensibilisation associées à ces dernières, il est difficile, sauf dans des cas très simples, de déterminer l'ordre de franchissement.

Afin de prendre en compte l'aspect continu, l'idée est de transformer les seuils continus (représentés par les fonctions de sensibilisation) en seuils temporels. Ceci est possible en simulant (résolvant) les équations différentielle associées aux différentes places du réseau de *Petri Prédicats transitions différentiel*. On a alors une seule variable qui est le temps. Variable linéaire qui permet de déterminer l'ordre de franchissement entre les transitions et ainsi l'ordre partiel entre les événements des scénarios redoutés. Dans ce cas, les données temporelles ne sont plus statiques déterminées au préalable, mais dynamiques déterminées en fonction de l'évolution du système.

L'algorithme hybride de génération de scénarios redoutés, consiste à coupler l'algorithme initial présenté dans le chapitre V avec un solveur d'équations différentielles.

La résolution de toutes les équations associées aux places du réseau de Petri PTD peut être lourde en temps de calcul. Une solution est de limiter la résolution des équations à certaines

places du réseau de Petri. En effet, la recherche de scénarios redoutés est locale (elle s'intéresse uniquement aux objets qui sont concernés par le scénario redouté).

L'exécution de l'algorithme de recherche de scénarios redoutés permet de déterminer le choix des équations à résoudre en fonction de l'évolution du marquage du réseau de Petri.

Les différentes procédures de l'algorithme qui font changer la configuration discrète du système sont le tir de transitions, l'enrichissement de marquage et l'introduction des jetons initiaux. A chaque changement de configuration discrète, le système d'équations qui lui est associé et qu'il faudra résoudre est déterminé. Ce système est transmis au solveur d'équations différentielles, qui après résolution renvoie à l'algorithme les différentes dates de franchissement des transitions sensibilisées. Ces dates permettent de déterminer l'ordre de franchissement de ces transitions.

Plus précisément, lorsque une place est marquée suite à un tir de transition, un enrichissement de marquage ou une introduction de jetons initiaux, le solveur déclenche l'intégration des équations différentielles associées à la place. L'intégration s'arrête une fois que tous les seuils des transitions de sortie de la place concernée sont atteints.

Pour l'intégration des équations les valeurs initiales des variables continues, sont déterminées de deux façons : soit grâce aux fonctions de jonction associées aux différentes transitions du réseau de Petri, dans le cas de changement de configuration suite à un franchissement de transition, soit en considérant les domaines déterminés lors de la première étape de simulation dans le cas d'un enrichissement de marquage ou de l'introduction des jetons initiaux (marquage initial). En effet, dans le cas de l'enrichissement de marquage, sauf dans des situations simples, les valeurs initiales des variables continues ne sont pas connues.

Etapes de l'algorithme hybride de génération de scénarios redoutés

L'algorithme est basé sur un couplage de l'algorithme de génération de scénarios redoutés basé sur la logique linéaire et un solveur d'équations différentielles. Le solveur se charge de la partie continue et l'algorithme de la partie discrète.

L'algorithme hybride de génération de scénarios redoutés est composé de 5 étapes (figure VI.2):

La première étape modifie la structure du système d'équation. Les équations associées aux places démarquées sont supprimées du système d'équation, tandis que celles qui correspondent aux places dans lesquelles un jeton est déposé sont ajoutées au système d'équations. Dans cette étape, c'est le simulateur discret (algorithme de recherche de scénarios redouté) qui est activé. C'est lui qui se charge de la mise à jour du marquage du réseau de Petri ainsi que de la mise à jour du système d'équations.

La deuxième étape consiste à déterminer les conditions de sortie de l'état discret courant. L'algorithme détermine la liste des transitions franchissables ainsi que la liste des fonctions de jonction à superviser. Ces deux listes sont ensuite communiquées au simulateur continu.

La troisième étape consiste à exécuter les fonctions de jonction correspondant aux transitions franchies. Il s'agit de modifier les valeurs des variables continues. C'est le simulateur continu qui s'en charge.

L'intégration des équations démarre et se poursuit jusqu'à l'apparition de tous les seuils associés aux événements (dates de franchissement des différentes transitions) contrairement à une simulation classique où l'intégration s'arrête dès l'apparition du premier seuil. Cette quatrième étape concerne le simulateur continu. Il faut noter que certains seuils peuvent être vrais dès l'exécution de la fonction de jonction, l'intégration est court-circuitée dans ce cas et l'intervalle de temps transmis est [0,0].

Finalement une fois que tous les seuils sont atteints, les différentes dates sont transmises au simulateur discret (Algorithme) qui exécute ses différentes procédures en fonction des données transmises (dates de franchissement des transitions).

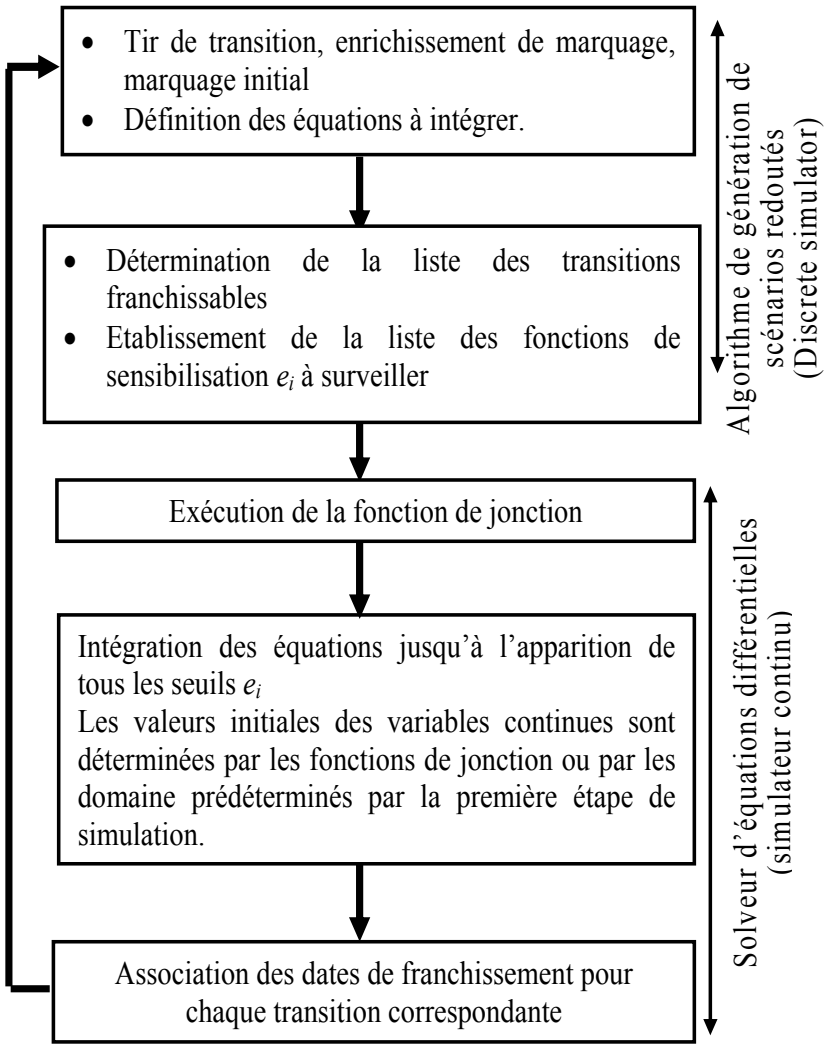


Figure VI.2. Etapes de l'algorithme hybride de génération de scénarios redoutés.

III.3 Exemple d'application

III.3.1 Présentation

Il s'agit d'un système de régulation du volume de deux réservoirs (figure VI. 3). Il est constitué d'un calculateur, de deux pompes, de trois électrovannes (tout ou rien), de deux capteurs de volume et des deux réservoirs régulés (Réservoir 1, Réservoir 2) et d'un troisième réservoir de vidange. Les deux réservoirs régulés alimentent des utilisateurs selon un besoin prédéfini (fonction du temps). Le volume dans chaque réservoir (1 ou 2) doit rester dans un intervalle donné [V_{imin}, V_{imax}] (i= *1 ou 2*). Le contrôle s'opère à l'aide du calculateur qui décide, selon la valeur du volume (délivrée par le capteur), d'approvisionner (ou non) le réservoir en question en alimentant (ou non) l'électrovanne concernée. Pour chaque réservoir, on distingue donc deux phases de fonctionnement selon que l'électrovanne alimentant ce réservoir est ouverte ou fermée :

- Une phase de conjonction lorsque l'électrovanne est ouverte. Le volume dans le réservoir est croissant durant cette phase.
- Une phase de disjonction lorsque l'électrovanne est fermée. Le volume dans le réservoir est par conséquent décroissant.

La loi de contrôle du calculateur pour chaque réservoir est telle que lorsque le volume dépasse la limite supérieure de commande V_{imax} pendant la phase de conjonction, le calculateur commande la fermeture de l'électrovanne. Lorsque le volume devient inférieur à V_{imin} (limite inférieure de commande) durant la phase de disjonction alors le calculateur demande à l'électrovanne de s'ouvrir et on change par conséquent de phase de fonctionnement. Ce système doit assurer l'approvisionnement des utilisateurs tout en évitant le débordement de l'un des réservoirs. Une troisième électrovanne de secours est prévue pour cet effet. Elle est partagée entre les deux réservoirs et assure leur vidange avant qu'ils débordent. Elle ne peut être utilisée que par un seul réservoir à la fois. Quand le volume dans l'un des réservoirs dépasse la limite supérieure de sécurité (*ViL*), le calculateur commande l'ouverture de cette électrovanne du côté du réservoir qui risque de déborder, et ce jusqu'à ce que le volume devienne inférieur à *Vimin*.

Nous supposons que seuls les actionneurs (les électrovannes) peuvent subir des défaillances. Les défaillances des capteurs sont intégrées dans celles des électrovannes. Les électrovannes *ev1* et *ev2* (prévues pour l'alimentation des réservoirs) peuvent être bloquées en ouverture (ou en fermeture) avec une possibilité de réparation. En cas de défaillance de l'électrovanne 3 de secours (*ev3*), celle-ci est définitivement mise hors service (on ne peut pas l'ouvrir pour faire la vidange).

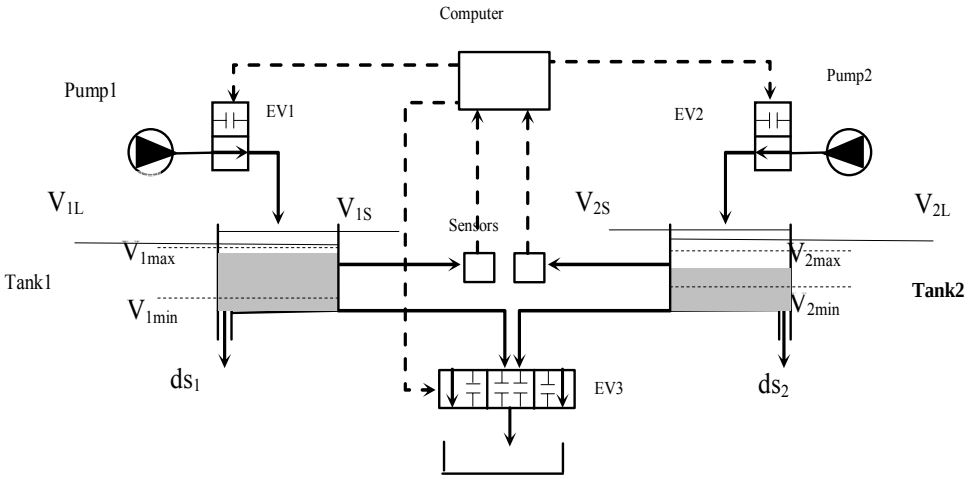


Figure VI.3. Cas d'étude.

Maintenant, supposons que l'une des électrovannes (ex *ev1*) soit bloquée en ouverture c'est-à-dire qu'on ne peut pas la fermer. Le volume dans le réservoir 1 continue de croître jusqu'à atteindre V_{1L}. Dans ce cas, l'électrovanne *ev3* de secours doit être ouverte pour vidanger le réservoir 1. Si l'électrovanne *ev3* est aussi hors service (elle ne peut être ouverte) ou si elle est occupée par la vidange du réservoir 2, alors le volume dans le réservoir 1 dépassera V_{1L} pour atteindre V_{1S}, ce qui impliquera le débordement de ce réservoir. Le débordement du réservoir est considéré comme un premier événement redouté. Un raisonnement analogue mené sur le réservoir 2 recenserait un deuxième événement redouté.

Dans cet exemple deux événements redoutés sont considérés : débordement du réservoir 1 et celui du réservoir 2.

III.3.2 Modèle du système

Nous proposons trois classes pour modéliser ce système: une classe réservoir et deux classes électrovanne. Nous définissons deux objets (réservoir 1 et de réservoir 2) instances de la classe réservoir, deux électrovannes (*ev1* et *ev2*) instances de la première classe électrovanne et une troisième *ev3* instance de la deuxième classe électrovanne (de secours).

Modèle des réservoirs

La figure VI.4 représente le modèle du réservoir 1. La place *V1_dec* représente la phase de disjonction (le volume décroît). Quant à la place *V1_cr*, elle représente la phase de conjonction où le volume croît. Quand le volume excède le seuil V_{1max} le réservoir appelle la méthode `fermeture electrovanne_1' (appel qui correspond à la transition $t_{11}$) fourni par l'électrovanne *ev1*. L'appel de la méthode `ouverture electrovalve_1' (transition t_{12}) est fait par le réservoir 1 quand le volume devient inférieur à V_{1min}. Quand le volume dans le Réservoir1 dépasse le seuil limite de sécurité V_{1L} la méthode 'ouverture électrovanne_3' (transition t_{14}) fournie par l'électrovanne de secours est appelée afin de vidanger le réservoir 1.

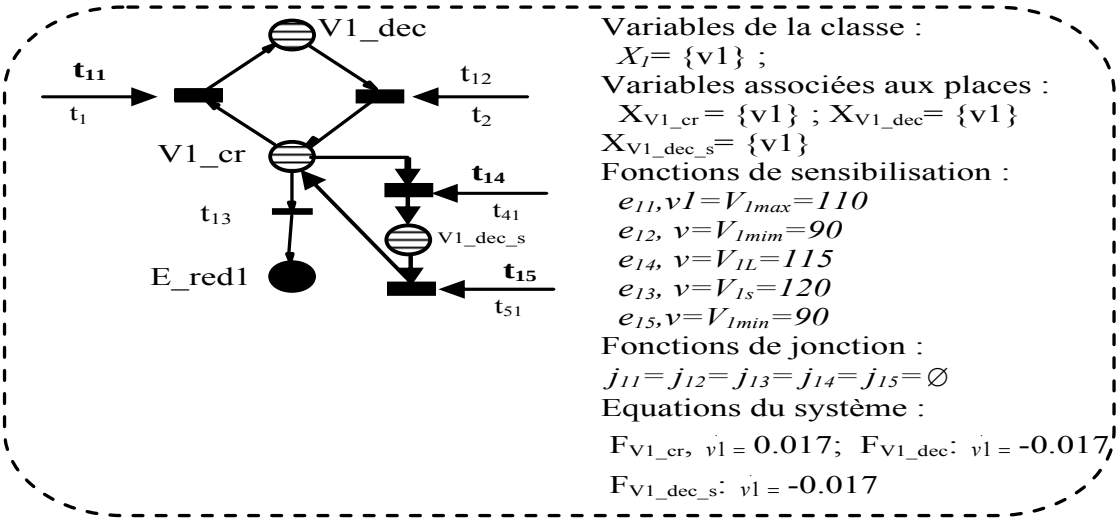


Figure VI.4. Modèle du réservoir 1

Cette phase de vidange dure le temps nécessaire au volume pour atteindre le seuil bas V_{1min}. Une phase de conjonction peut alors commencer (la place *V1_cr* marquée après le tir de la transition t_{15}). Le débordement du réservoir correspond au tir de la transition t_{13}.

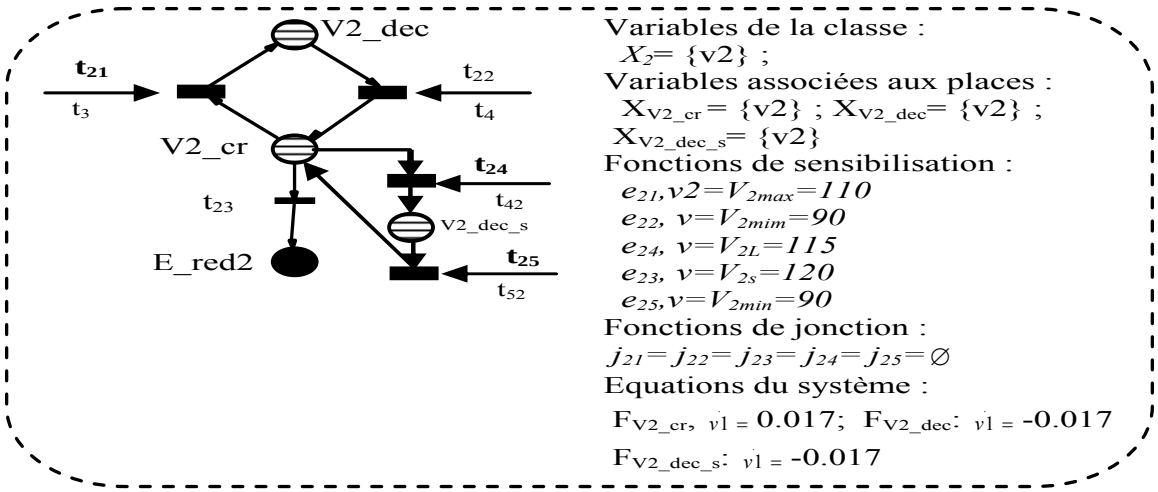


Figure VI.5. Modèle du réservoir 2

Modèle de l'électrovanne 1

L'électrovanne 1 (*ev1*) quand elle est ouverte approvisionne le réservoir 1 dans la phase de conjonction (figure VI.6). *ev1* est fermée (tir de la transition t_1) quand le réservoir 1 appelle la méthode 'fermeture électrovanne 1'. La transition t_2 représente l'ouverture de l'électrovanne. En

fonctionnement normal, les deux méthodes offertes par *ev1* garantissent que le volume dans le réservoir 1 reste à l'intérieur de l'intervalle indiqué [V_{1min}, V_{1max}].

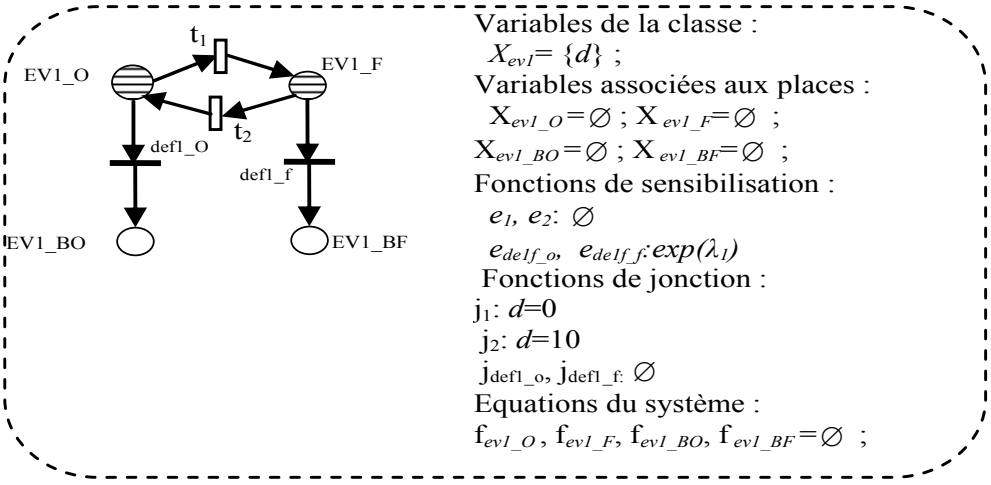


Figure VI.6. Modèle de l'électrovanne ev1

Les défaillances de l'électrovanne sont représentées par le tir des transitions def1_O (blocage en ouverture) ou def1_F (blocage en fermeture). Le débit des électrovannes est égal a *d* il est le même pour les deux pompes.

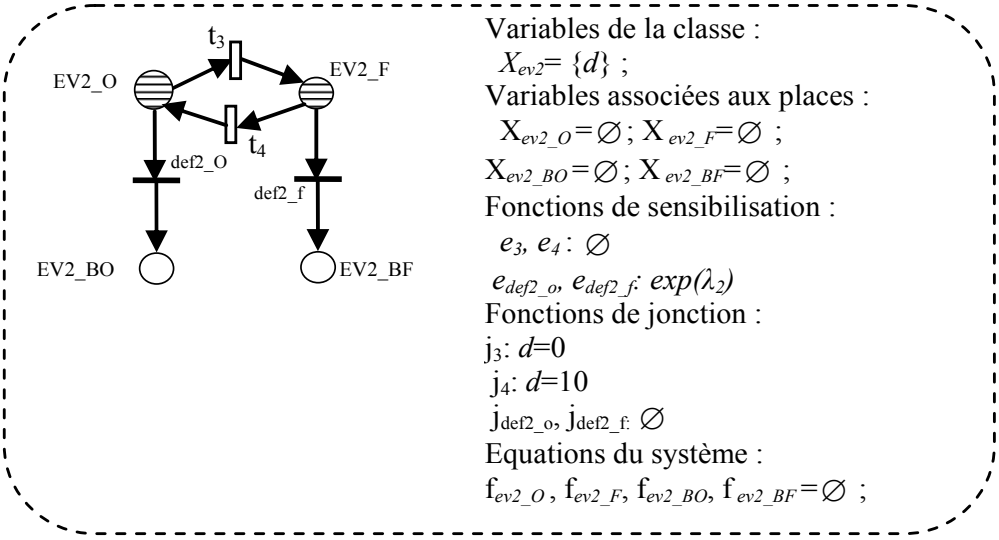


Figure VI.7. Modèle de l'électrovanne ev2

Modèle de l'électrovanne de secours

L'électrovanne de secours est partagée entre les deux réservoirs, son modèle est présenté sur la (figure VI.8). La place *EV3_OK* représente la disponibilité de l'électrovanne. La transition t_{41} (respectivement t_{42}) représente la méthode 'ouverture l'electrovalve_3' offerte par *ev3*, qui peut

être appelée par le réservoir 1 (respectivement réservoir 2) quand le volume dépasse le seuil limite V_{1L} (respectivement V_{2L}). L'électrovanne 3 peut tomber en panne (franchissement de la transition *def3*). Dans ce cas, la place *EV3_HS* est marquée et l'électrovanne est définitivement hors service.

Le débit de vidange de l'électrovanne de secours est égal a d_{vid}.

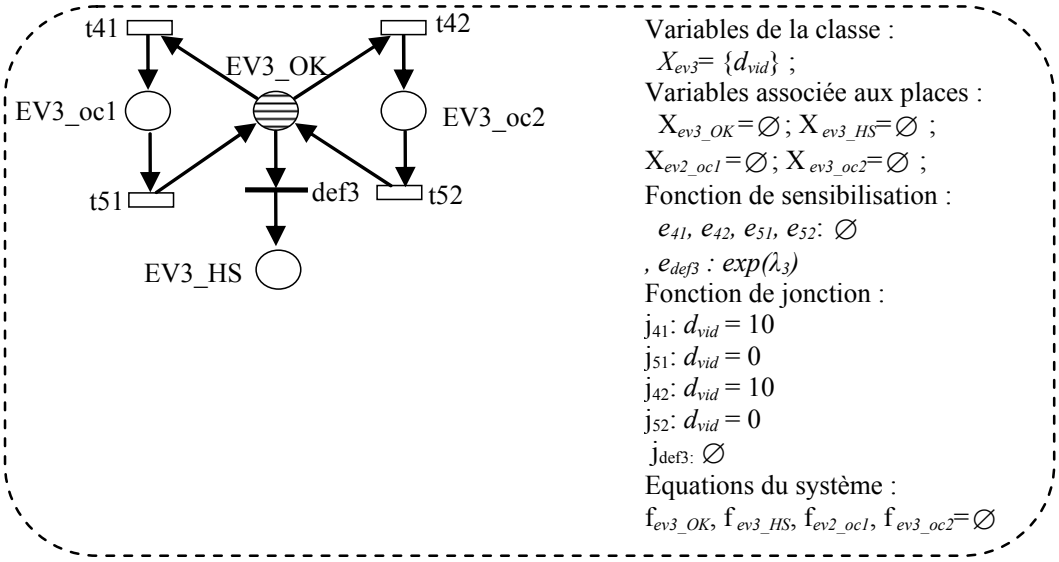


Figure VI.8. Modèle de l'électrovanne de secours ev3

III.3.3 Hypothèses de dysfonctionnement

Nous supposons que seuls les actionneurs (électrovanne) sont susceptibles de défaillir. Les défaillances des capteurs sont intégrées dans celles des électrovannes. Ces composants sont mutuellement indépendants et non réparables. Les occurrences des défaillances des actionneurs suivent des lois exponentielles de taux constants. Pour chacun d'eux, plusieurs modes de défaillance sont envisagés. Ils sont répertoriés dans le tableau suivant :

Actionneurs	Mode de défaillance	Taux de défaillance
Electrovanne *ev1*	Blocage en ouverture	λ_1=2,7.10^{-3}h^{-1}
	Blocage en fermeture	λ_1= 2,7.10^{-3} h^{-1}
Electrovanne *ev2*	Blocage en ouverture	λ_2= 2,7.10^{-3} h^{-1}
	Blocage en fermeture	λ_2= 2,7.10^{-3} h^{-1}
Electrovanne de secours *ev3*	Hors service	λ_3=9.10^{-4} h^{-1}

Tableau VI.1. Récapitulatif des défaillances.

III.3.4 Simulation et recherche de scénarios

Première étape : simulation de Monte Carlo

L'objectif de cette étape est de déterminer les domaines des valeurs initiales des variables continues mises en jeu.

Nous avons utilisé l'outil de simulation de réseaux de *Petri Prédicats transitions différentiels* que nous avons développé. Et nous avons simulé 10 histoires sur un horizon de simulation (durée d'une histoire) de 5000 heures.

Cette première étape a permis de déterminer les conditions initiales des variables continues *V1* (volume dans le réservoir 1) et *V2* (volume dans le réservoir 2) tel que :

- *V1(0)= 89.99*, pour le marquage de la place *V1_cr*,
- *V1(0)= 109,21*, pour le marquage de la place *V1_dec*,
- *V1(0)= 114,99*, pour le marquage de la place *V1_dec_s*,
- *V2(0)= 90.01*, pour le marquage de la place *V2_cr*,
- *V2(0)= 109,99*, pour le marquage de la place V2_dec,
- *V2(0)= 114,99*, pour le marquage de la place *V2_dec_s*,

Ces données seront utilisées dans la deuxième étape par l'algorithme qui transmettra au solveur dès qu'il aura à résoudre un système d'équations différentielles.

Deuxième étape : Application de l'algorithme hybride de recherche de scénarios redoutés

L'objectif de cette étape est de déterminer les scénarios redoutés non générés par la première étape de simulation Monte Carlo. L'exécution de l'algorithme hybride de génération de scénarios redoutés nous a permis de générer deux scénarios menant vers l'état redouté débordement du réservoir 1.

Le Premier scénario (Figure VI.9) est composé des événements suivants : défaillance de l'électrovanne ev1, défaillance de l'électrovalve de secours, suivi du débordement du réservoir 1.

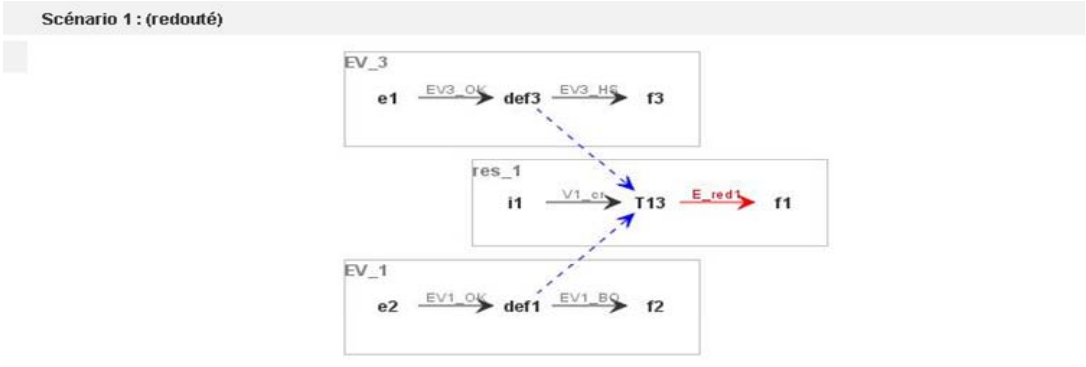


Figure VI.9. Premier scénario redouté

Le deuxième scénario (Figure VI.10) est constitué des événements suivants : défaillance de l'électrovanne 2, utilisation de l'électrovanne de secours par le réservoir 2 suite du débordement du réservoir 1.

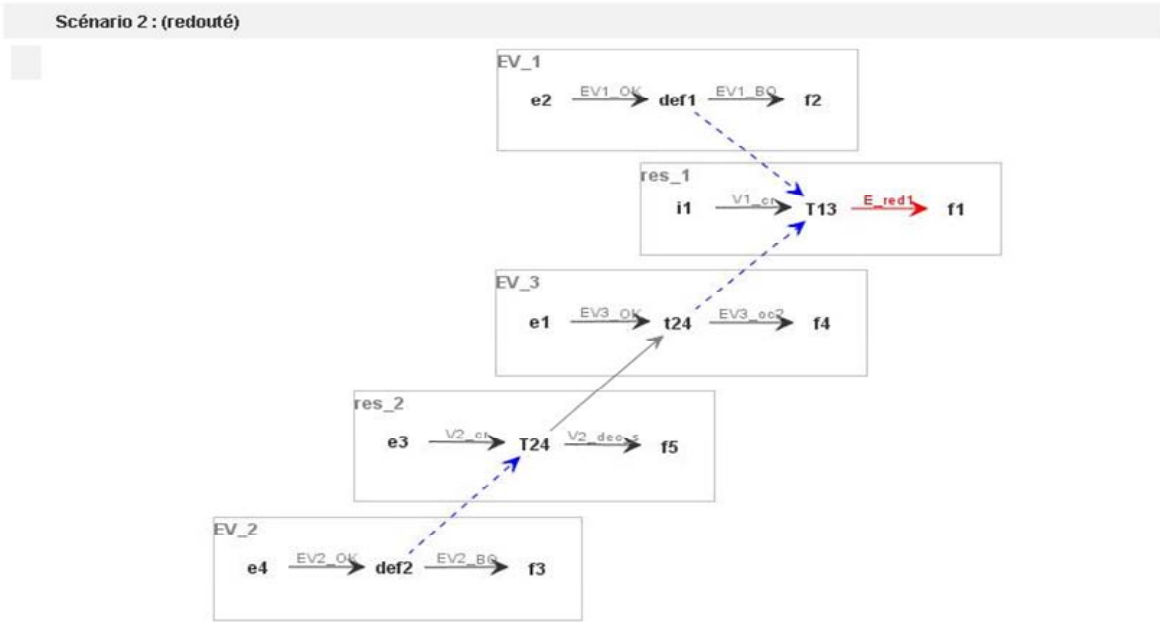


Figure VI.10. Deuxième scénario redouté

Remarque 1 :

L'horizon de simulation Monte Carlo doit être suffisamment grand pour permettre de parcourir la totalité des places dites normale afin de déterminer les valeurs initiales des variables continues. Ceci dit, l'intervalle de simulation doit être raisonnable pour éviter des temps de simulation faramineux. La question qui peut se poser, c'est comment définir à l'avance l'intervalle de simulation d'une histoire ? Un avis expert est donc nécessaire pour bien paramétrer la simulation.

Remarque 2 :

Pour accélérer la simulation et pouvoir s'assurer de visiter tous les états du système, une solution est de modifier les lois de probabilité associées aux transitions stochastiques, afin de s'affranchir du problème des événements rares. En effet, l'objectif n'étant pas d'estimer les probabilités d'apparition des événements redoutés mais la détermination des domaines des variables continues, cette adaptation des probabilités n'a aucun effet sur les domaines déterminés tout en permettant d'accélérer la simulation.

Remarque 3 :

Il faut noter que l'algorithme hybride de génération de scénarios redoutés, n'exploite les valeurs initiales déterminées par la simulation Monte Carlo, que lorsque cela est nécessaire. C'est-à-dire, dans le cas d'un enrichissement de marquage d'une place continue ou la définition du marquage initial. Dans le cas, d'un marquage suite à un franchissement de transition, l'algorithme exécute la fonction de jonction associée à la transition franchie.

IV Conclusion

Dans ce chapitre nous avons présenté une nouvelle approche pour la génération des scénarios redoutés dans les systèmes hybrides. L'approche permet de prendre en compte la dynamique continue associée au système sans passer par des abstractions. L'approche consiste à coupler l'algorithme de génération de scénarios redoutés avec un solveur d'équations différentielles. L'algorithme se charge de la partie discrète (changement de configuration) et le solveur de la partie continue. L'originalité de l'approche réside dans son caractère local. En effet, il n'est plus nécessaire de simuler la totalité du système mais uniquement les parties intéressantes pour la recherche de scénarios. Cela réduit considérablement le temps de simulation. A cause du caractère local de l'algorithme de recherche de scénarios, les valeurs initiales des variables continues ne sont pas toujours disponibles. Il est donc nécessaire d'effectuer un travail préliminaire qui permet de déterminer les domaines des variables continues. Il consiste à réaliser une simulation de Monte Carlo avec un nombre limité d'histoires qui permet de définir domaines de variables continues, en particulier, les domaines des valeurs initiales des variables continues.

Conclusion générale

Les systèmes embarqués sont des systèmes énergétiques contrôlés par un calculateur. Ils sont utilisés dans de nombreuses applications, aussi bien dans les domaines de la défense, de l'avionique, du spatial, de l'automobile, que dans celui du nucléaire. La flexibilité offerte par l'implémentation logicielle des fonctions de contrôle est le principal avantage de ces systèmes. La facilité d'implémentation des lois de contrôle et de commande, permet d'intégrer de plus en plus de fonctions, cette capacité d'intégration les rend de plus en plus complexes et pose des problèmes de sûreté de fonctionnement. La maîtrise de leur fiabilité est rendue difficile.

Les travaux développés dans cette thèse sont une contribution à l'aide à la conception des systèmes embarqués sûrs de fonctionnement et plus particulièrement la recherche de scénarios redoutés.

Pour mettre en évidence les scénarios redoutés, [Khalfaoui 03] dans sa thèse a proposé une méthode d'extraction de scénarios redoutés à partir d'un modèle réseau de Petri. Son approche est basée sur la logique linéaire et permet de générer les scénarios redoutés sans avoir à explorer la totalité du système. L'approche est basée sur une analyse des relations de cause à effet et permet de déterminer les scénarios non sous forme de séquence d'événements mais par un ordre partiel entre les événements du scénario. Pour modéliser l'aspect hybride, Khalfaoui a utilisé le modèle réseau de *Petri prédicats transitions différentiel stochastique*. Néanmoins, la première version de l'algorithme de génération de scénarios redoutés ne prend en compte que les aspects discrets. La partie continue n'est pas prise en compte.

[Medjoudj, 2006], grâce à des abstractions temporelles de la dynamique continue prend en compte partiellement l'aspect continu et permet ainsi d'éliminer certains scénarios incohérents avec la dynamique continue du système. Ses travaux ont donné naissance à un premier prototype permettant d'automatiser la méthode de génération de scénarios redoutés.

Nous avons donc poursuivi les travaux précédents afin d'améliorer l'approche et d'aborder certains aspects qui n'ont pas été traités lors des deux thèses précédentes.

Intégration de la sûreté dans les processus d'ingénierie système

Devant la complexité de ces systèmes il nous est paru nécessaire de proposer une approche globale d'analyse de sûreté de fonctionnement qui permet de prendre en compte les risques liés à l'intégration de plusieurs technologies. Nous avons donc proposé une approche d'intégration de la sûreté de fonctionnement dans les processus d'ingénierie système. La norme *EIA-632* par ailleurs très utilisée dans l'industrie à été choisie. La norme décrit des processus pour le développement des systèmes allant de la formulation des exigences jusqu'au retrait. Ceci permet d'avoir une traçabilité des exigences de sûreté de fonctionnement et permet de s'assurer de la prise en compte de ces exigences tout au long du cycle de vie du système

Approche orientée objets

En nous appuyant sur les travaux de [Villani & al, 2004] et [Esteban & al, 2005], nous avons proposé une démarche de modélisation et d'analyse structurée et modulaire basée sur une approche orientée objets. Cette démarche permet d'effectuer l'analyse non plus en considérant le système dans sa globalité, mais en considérant au fur et à mesure les objets qui le composent. La nouvelle version de l'algorithme de génération de scénarios redoutés est donc elle aussi orientée objets. Ceci facilite l'analyse et réduit le temps nécessaire de recherche de scénarios.

Formalisation de la notion de scénario et ses propriétés

Un long travail théorique, nous a permis de mieux formaliser la notion de scénario. Une définition formelle basée sur la logique linéaire de la notion de scénario a été proposée. Cette définition nous a permis ensuite de définir formellement la notion de minimalité dans le cas trivial où le marquage du réseau est complet, mais aussi dans le cas où les marquages ne sont que partiellement connus. Dans le deuxième cas qui correspond à notre situation, un marquage final minimal basé sur la notion de coupe minimale a été défini. Le scénario minimal est alors défini par rapport au marquage final minimal associé à la coupe minimale représentant l'état redouté. La notion de minimalité a été introduite dans l'algorithme et l'outil de génération de scénarios redoutés. Ceci a permis d'éliminer un bon nombre de scénarios non minimaux générés dans les versions précédentes facilitant ainsi l'étape d'analyse.

Une autre propriété importante des scénarios et la propriété de complétude. Cette notion a été définie à partir de la définition du scénario minimal.

Nouvelle version de l'algorithme de recherche de scénario redouté

Devant la difficulté et les limitations des approches consistant à approximer la dynamique continue des systèmes hybrides pour des fins d'analyse, la simulation constitue une approche alternative. Dans notre cas, la simulation se fait en deux étapes, une première étape correspond à simulation de Monte Carlo sur un intervalle de temps suffisamment grand mais qui reste raisonnable. L'objectif de cette première étape est de déterminer un certain nombre de scénarios redoutés pouvant se produire dans l'intervalle de temps choisi, mais aussi les domaines des variables continues et particulièrement les domaines des valeurs initiales des variables

continues. La deuxième étape est une simulation hybride basée sur l'algorithme de génération de scénarios redoutés couplé avec un solveur d'équations différentielles. Elle permet de prendre en compte la dynamique continue du système sans passer par des approximations. L'exécution de l'algorithme de recherche de scénarios redoutés permet de déterminer le choix des équations à résoudre en fonction de l'évolution du marquage du réseau de Petri. Ces équations sont transmises au solveur qui se charge alors de les résoudre et de renvoyer à l'algorithme les différentes dates de franchissement des transitions sensibilisées. Ces dates permettent de déterminer l'ordre de franchissement de ces transitions.

Nouvel outil

Nous avons poursuivi l'automatisation de l'algorithme de génération de scénarios redoutés. La nouvelle version permet de prendre en compte tous les apports théoriques développés lors de cette thèse. L'outil est couplé avec un solveur d'équations différentielles et permet ainsi de mettre en mettre en œuvre la simulation hybride pour la génération de scénarios redoutés.

La procédure de recherche est incrémentale. En effet, si le scénario généré est pauvre en informations à cause du critère d'arrêt le concepteur peut relancer l'algorithme afin d'enrichir le scénario.

L'outil permet aussi d'effectuer une simulation de Monte Carlo à partir d'un modèle réseau de *Petri Prédicats Transitions différentiel stochastique*. Ainsi La première étape de l'algorithme (détermination des états de fonctionnement normal) a été, elle aussi, automatisée.

Perspectives

Beaucoup de points restent à améliorer. Concernant l'algorithme, il s'agira de préciser la notion de critère d'arrêt et de places normales. Nous pouvons avoir un ensemble de places qui, individuellement, sont fréquemment marquées sans qu'elles correspondent effectivement à un marquage associé à un état normal. En effet, la situation correspondant au marquage simultané de toutes ces places peut être très rare. Une solution pour remédier à ce problème est de considérer les probabilités de marquage des places individuellement puis la probabilité des différents marquages auxquels elles appartiennent.

Avant de penser à utiliser l'algorithme de recherche de scénario dans sa version hybride dans un contexte industriel, il est nécessaire de traiter des exemples plus conséquents. Ceci nous permettra de valider l'outil.

L'analyse quantitative est un point important à explorer pour estimer la probabilité d'apparition de l'état redouté. Car même si l'étude qualitative présente un intérêt pour les nouveaux systèmes pour lesquels les données de fiabilité ne sont pas disponibles, il est indispensable d'estimer la probabilité d'occurrence des scénarios.

Un travail effectué par Medjoudj et Labeau a pour objectif d'estimer par un algorithme Monte Carlo efficace la probabilité d'occurrence de scénarios redoutés. Cet algorithme est implémenté dans l'outil ESA_PetriNet (Extraction & Scenarios Analyser by PetriNet model) [Medjoudj, 2006] qui permet d'extraire qualitativement des scénarios redoutés à partir du modèle réseau de Petri

du système. L'originalité de leur estimation consiste en ce qu'elle est basée sur ces scénarios, préalablement identifiés, ce qui évite donc d'effectuer une simulation globale basée sur tout le réseau de Petri. En effet, ils ont optimisé la simulation Monte Carlo en forçant les histoires tirées au sort à suivre ces scénarios pré-identifiés. Il faudra donc poursuivre dans cette voie.

Un autre point important pour lequel il reste beaucoup de travail est l'approche d'intégration de la sûreté de fonctionnement dans les processus d'ingénierie système. L'approche proposée n'est qu'un premier jet d'un travail qui devra et sera poursuivi dans le cadre d'une thèse qui s'effectuera au LAAS. En effet l'approche ne prend en compte que trois processus de la norme *EIA-632*. Le processus des exigences, le processus de la représentation de la solution logique et le processus de validation. Il faudra donc prendre en compte les autres processus, mais aussi proposer des méthodes et outils adaptés à chaque processus.

Bibliographie

[Abrial, 1995] Abrial. J.R., Steam-boiler control specification problem, Material for the Dagstuhl meeting Methods for Semantics and Specification, June 4-9, 1995.

[AFIS] http://www.afis.fr/

[Alla & David, 2004] Alla H, David R, Discrete, continuous, and hybrid Petri nets. Springer Verlag, Berlin.

[Aubry, 1987] AUBRY, JF. Conception des systèmes de commande numériques des convertisseurs électromécaniques : vers une méthodologie intégrant la sûreté de fonctionnement. Thèse d'Etat. 1987. Institut National Polytechnique de Lorraine.

[Aubry & Zanne, 1991] AUBRY, J-F & ZANNE, C. Intégration de la sûreté de fonctionnement dans la conception des systèmes de commande numérique des processus électromécaniques. APII, AFCET - Sûreté de fonctionnement. 1991, vol. 25, p.297-324.

[Alur & Al 2000] R. Alur, T. Henzinger, G. Lafferriere, and G.Pappas. Discrete abstractions of hybrid systems. In Proceedings of IEEE, volume 88, pages 971-984, 2000.

[Alur & Dill 1994] R. Alur , D. Dill. A theory of timed automata. Theoretical Computer Science B, 126 :183–235, 1994.

[Andreu, 1996] David A, commande et supervision des procédés discontinus, thèse présentée au LAAS le 15 novembre 1996.

[Antsaklis & al, 2003] Antsaklis PJ, Koutsoukos XD, Hybrid Systems: Review and Recent Progress. In: Samad T, Balas G (eds.) Software-Enabled Control: Information Technologies for Dynamical Systems, IEEE Press.

[Aralia, 1996] Groupe Aralia, «Computation of prime implicants of a fault tree within Aralia», Reliability Engineering and System Safty, Special issue on selected papers from ESREL'95, 1996.

[Arnold & al, 1994] A. Arnold, D. Begay, P. Crubille, «Construction and analysis of systems with MEC», World Scientific, 1994.

[Barrangan & al, 2006] I.S. Barrangan Santiago, M Roth, J.M. Faure. Obtaining temporal and timed properties of logic controllers from fault tree analysis, 12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2006, Saint-Etienne, France, May, 17-19, 2006, pp. 243-248

[Batut, 1986] J. Batut. Fiabilité prévisionnelle du réseau à très haute tension d'edf. In 5eme Colloque international de fiabilité et de maintenabilité, Biarritz, France, October 1986.

[Benardi & al, 2001] S. Bernadi, et al., "GreatSPN in the new millenium", Tool Session of 9th Int. Workshop on Petri Nets and Performance Models, Aachen, Allemagne, 2001.

[Béounes & al, 1993] C. Béounes, et al., "Surf-2: a program for dependability evaluation of complex hardware and software systems", 23rd IEEE International Symposium on Fault Tolerant Computing, Toulouse, France, pp. 668-673, 1993.

[Bondavalli, & al, 2001] A. Bondavalli, et al., "Dependability Analysis in the Early Phases of UML Based System Design," Int. Journal of Computer Systems-Science&Engineering, vol. 16, pp. 265-275, 2001.

[Booch, 1994] Booch G., Object-oriented analysis and design with applications. 2.ed. Benjamin/Cummings Pub. Co., Redwood City

[Booch, 1998] G. Booch, J. Rumbaugh et I. Jacobson, "The Unified Modeling Language User Guide", Addison- Wesley, 1998.

[Bouissou & Bon, 2003] A new formalism that combines advantages of fault-trees and Markov models: Boolean logic Driven Markov Processes. Reliability Engineering and System Safety, Volume 82, Issue 2, November 2003, Pages 149-163.

[Bouissou, 2005] M. BOUISSOU Dix petits problèmes de modélisation (en sûreté de fonctionnement des systèmes) Revue Phoebus N°34 (Juillet-Août-Septembre 2005).

[Bouissou., 2002] M. Bouissou. Boolean logic driven markov processes : a powerful new formalism and solving very large markov models. In Proceedings of Probabilistic Safety Assessment and Management, 2002.

[Bryant, 1986] R.E. Bryant. Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers, C-35(8) :677–691, August 1986.

[Calvez, 1992] CALVEZ, J.P. Spécification et conception des systèmes – une méthodologie. Paris : Masson, 1992.

[Cardoso, 1999] J. Cardoso (1999). Time Fuzzy Petri Nets, Fuzziness in Petri Nets – Studies in Fuzziness and Soft Computing, Physica-Verlag, Janette Cardoso e Heloisa Camargo, vol. 22, pp.115-145.

[Cepin & Mavko, 2002] M. Cepin and B. Mavko. A dynamic fault tree. Reliability Engineering and System Safety, 75 :83–91, 2002.

[Champagnat & al, 1998] Champagnat, R, P. Esteban, H. Pingaud, R. Valette (1998) " Modelling and simulation of a hybrid system through Pr/Tr PN DAE model ", ADPM'98 3rd International Conference on Automation of Mixed Processes, 19-20 March, Reims, France p. 131-137.

[Champagnat, 1998] R. Champagnat, «Supervision des systèmes discontinus: définition d'un modèle hybride et pilotage en temps-réel », Thèse de Doctorat de l'Université Paul Sabatier de Toulouse, soutenue le 1er octobre 1998.

[Chatelet, 2000] E. Chatelet. Sûreté de fonctionnement : méthodes et outils de base. Université de technologie de troyes edition, 2000.

[Ciardo & al, 1993] G. Ciardo, K. S. Trivedi, "SPNP: The Stochastic Petri Net Package (Version 3.1)", 1st Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'93), San Diego, CA, USA, pp. 390-391, 1993.

[Cury & al., 1998] J. Cury, B. Krogh et T. Niinomi , "Synthesis of supervisory controllers for hybrid systems based on approximating automata", IEEE transactions on Automatic and Control, 43(4) : pp. 564-569, 1998.

[David & alla, 1992] Du GRAFCET aux réseaux de Petri. Hermes Sciences Publication.2ème édition révisée et augmentée, 1992

[Deavours & al, 2002] D. D. Deavours, et al., "The Mobius Framework and its Implementation," IEEE Transactions on Software Engineering, vol. 28, pp. 956-969, 2002.

[Demmou & al, 2002] H. Demmou , S. Khalfaoui , N. Rivière , E. Guilhem, Extracting critical scenarios from a Petri net model using linear logic. Journal Européen des Systèmes Automatisés (APII-JESA), Vol.36, N°7, pp.987-999, 2002.

[Demmou & al, 2007] H. Demmou , M. Messaadia, N. Sadou, A.E.K. Sahraoui, Intégration de la sûreté de fonctionnement dans les processus d'ingénierie système. 7ème Congres International de Génie Industriel, Trois Rivières (Canada), 5-8 Juin 2007, 11p.

[Desieno & Stine, 1964] C.F. Desieno and L.L. Stine. A probability method for determining the reliability of electric power systems. IEEE Transaction on Power Aparatus and Systems, 83 :174–181, February 1964.

[Drath, 1998] Drath R (1998) Hybrid object nets: an object oriented concept for modelling complex hybrid systems. The 3rd International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems (ADPM), Reims, pp 436-442

[Dubi, 2000] A. Dubi. Monte Carlo Applications in Systems Engineering. John Wiley & Sons, Ltd. England, 2000.

[EIA, 1999] EIA STANDARD, Processes for engineering a system, January 1999, Electronic industries alliance

[Esteban & al, 2005] A component based approach for system design and virtual prototyping, 12th Annual European Concurrent Engineering Conference (ECEC'2005), Toulouse (France), 11-13 Avril 2005, pp.85-90

[Faucher, 2004] J. faucher, « pratique de l'AMDEC », Edition DUNOD, Paris 2004.

[Garnier, 1998] GARNIER, R. Une méthode efficace d'accélération de la simulation des réseaux de Petri stochastiques. Th. Automatique, productique. 1998. Université Bordeaux I.

[Garret, 1993] C.J. Garret, S.B. Guarro, G.E. Apostolakis, «Development of a methodology for assessing the safety of embedded software systems», American institute of Aeronautics and Astronautics, 1993.

[Genrich, 1987] Genrich H, Predicate/transition nets. Petri Nets: Central Models and Their Properties, Advances in Petri Nets, Lecture Notes in Computer Science 254: 207-247.

[Girard, 1987] J.Y Girard., Linear Logic, Theoretical Computer Science, 50, 1987, p.1-102.

[Girault, 1997] F. Girault., Formalisation en Logique Linéaire du fonctionnement des réseaux de Petri, Thèse de Doctorat, N°2870, Université Paul Sabatier, Toulouse, 1997.

[Gollu et Varaiya 1989] A. Gollu and P. Varaiya. Hybrid dynamical systems.In Proc. 28th IEEE Conf. Decision Contr., pages 2708-2712, 1989. Tampa, Florida.

[Griffault & al, 1998] A. Griffault, S. Lajeunesse, G. Point, A. Rauzy, J.P. Signoret, and P. Thomas, «The AltaRica language», In Proceedings of International Conference on Safety and Reliability, ESREL'98. Balkema Publishers, June 20-24 1998.

[Gueguen & al, 2001] Gueguen H, Lefebvre M, A comparison of mixed specification formalisms. Journal Européen des Systèmes Automatisés 35(4): 381-394

[Guerrero & al, 2001] Guerrero DDS, Figueiredo JCA, Perkusich A (2001) An object-based modular CPN approach: its application to the specification of a cooperative editing environment. Advances on Petri Nets, Lecture Notes in Computer Science, 2001: 338-354

[Hélias & al., 2002] A. Hélias, F. Guerrin, J. Harmand, J. Steyer (2002). Abstraction en modèles discrets de modèle d'évolution de stocks continus : Application à la gestion des effluents d'élevage, Système d'Information, Modélisation, Optimisation, Commande en Génie des Procédés (SIMO 2002), Toulouse, France.

[Hélias & al., 2004] A. Hélias, F. Guerrin, J-P Steyer (2004). Abastracting Continuous System Behaviours into Timed Automata: Application to Diagnosis of an Anaerobic Digestion Process, International Workshop on Principles and Diagnosis (DX 2004), Carcassone, France.

[Hélias, 2003] A. Hélias (2003). Agrégation/Abstraction de Modèles pour l'Analyse et l'Organisation de Réseaux de Flux : Application à la Gestion des Effluents d'élevage à la Réunion, Thèse de Doctorat, Ecole Nationale Supérieure Agronomique de Montpellier.

[Henzinger & al, 1996] Henzinger. T. A, Howard Wong-Toi. "Using HYTECH to Synthesize Control Parameters for a Steam Boiler". Lecture Notes in Computer Science vol 1165, 1996.

[Henzinger & al., 1994] T. Henzinger, X. Nicollin, J. Sifakis et S. Yovine, "Symbolic model checking for real-time systems", Information and Computation, 111(2) : pp. 193-244, 1994.

[Henzinger & al., 1997] T. Henzinger, H. HO P et H. Wong-Toi , "HYTECH: a model checker for hybrid systems", International Journal on Software Tools for Technology Transfer, 1(1-2) : pp. 110-122, 1997.

[Henzinger & al., 1998a] T. Henzinger, H. HO P et H. Wong-Toi, "Algorithmic analysis of nonlinear hybrid systems", IEEE transactions on Automatic and Control, 43(4) : pp. 540-554, 1998a.

[Henzinger et al., 1998b] T. Henzinger, P. Kopke, A. Puri et P. Varaiya, "What's decidable about hybrid automata?", Journal of Computer and System Sciences, 57 : pp. 94-124, 1998b.

[Hirel & al, 2000] C. Hirel, R. Sahner, X. Zang, and K. Trivedi, "Reliability and performability modeling using SHARPE 2000", 11th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools, Schaumburg, IL, USA, 1789, pp. 345-349, 2000.

[Howard, 1960] R.A. Howard. Dynamic programming and Markov processes. T. MIP Press, Massachusetts, 1960.

[Jensen, 1992] K. Jensen, «Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use», Vol. 1, Basic Concepts, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1992.

[Jensen, 1997] Jensen K, Coloured Petri nets: basic concepts, analysis methods and practical use. 2.ed., Springer Verlag, Berlin.

[Kaaniche, 1999] M.KAANICHE. Evaluation de la sûreté de fonctionnement informatique. Fautes physiques, fautes de conception, malveillances. Rapport LAAS No99062. Habilitation, Institut National Polytechnique, Toulouse, 12 Février 1999.

[Kehren & al, 2004] C. Kehren, C. Seguin, P. Bieber, C. Castel, C. Bougnol, J.-P. Heckmann et S. Metge - Architecture patterns for safe design - AAAF International

Complex and Safe Systems Engineering (CS2E04), Arcachon, France, Juin 2004.

[Kehren & al, 2004] C. Kehren, C. Seguin, P. Bieber; C. Castel, C. Bougnol, J.P. Heckmann, S. Metge, «Advanced Multi-System Simulation Capabilities with AltaRica», proceedings of Safety Critial System Conference, Rhodes Island, USA, august 2004.

[Kehren & Seguin, 2003] C. Kehren et C. Seguin, «Evaluation qualitative de systèmes physiques pour la sûreté de fonctionnement», Journées FAC'2003, Formalisation des Activités Concurrentes à l'IRIT, Toulouse, les 12 et 13 mars 2003.

[Kehren, 2005] C. Kehren. Motifs formels d'architectures de systèmes pour la sûreté de fonctionnement. Thèse de Doctorat de l'école nationale supérieure de l'aéronautique et de l'espace, 20 décembre 2005.

[Khalfaoui, 2003] S. Khalfaoui «Méthode de recherche des scénarios redoutés pour l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile», thèse de Doctorat, No 03574, Institut National Polytechnique, Toulouse, 26 septembre 2003.

[Köster & al, 2001] Köster F, Schof S, Sonnenschein M, Wieting R (2001) Modelling of a library with THORNs. Concurrent Object Oriented Programming and Petri Nets, Lecture Notes in Computer Science 2001: 375-390.

[Labeau, 2002] LABEAU, P-E. & KERMISCH, C. Approche dynamique de la fiabilité des systèmes. Rapport MNFD 2002-07, Service de Métrologie Nucléaire, Université Libre de Bruxelles, juillet 2002.

[Lafortune & Chen, 1991] S. Lafortune, E. Chen (1991). A relational algebric approach to the representation and analysis of discret-event system, Proc. Amer. Contr. Conf., Boston, MA, pp.2893-2898.

[Lakos, 1995] Lakos C., From coloured Petri nets to object Petri nets. Lecture Notes in Computer Science 935: 223-228.

[Laprie & al, 1996] J.C. Laprie, J. Arlat, J-P. Blanquart, A. Costes, Y. Crouzet, Y. Deswarte, J-C. Fabre, H. Guillermain, M. Kaaniche, C. Mazet, D.Powell, C. Rabéjac et P. Thévenod., «Guide de la Sûreté de Fonctionnement », 2ème édition (Cépaduès), ISBN : 2.85428.382.1,1996.

[Laprie & al, 2004] J.C. Laprie, «Sûreté de fonctionnement des systèmes : concepts de base et terminologie », Rapport LAAS N°04520, 2004.

[Larsen & al, 1997] K. Larsen, P. Pettersson, Yi W, «UPPAAL in a nutshell», International Journal of Software Tools for Technology Transfer, 1(1-2): pp. 134-152, 1997.

[Lenoir, 20000] F.X. LENOIR,. S'engager sur la sécurité de fonctionnement. Industries et Techniques. N°818. 2000.

[Loures & Pascal, 2005] E.Rocha Loures, J.C. Pascal, A diagnosis framework of hybrid dynamic systems based on time fuzzy Petri nets. 16th IFAC World Congress, Prague (République Tchèque), 3-8 Juillet 2005, 6p.

[Loures, 2006] E. Rocha Loures, Surveillance et diagnostic des phases transitoires des systèmes hybrides basés sur l'abstraction des dynamiques continues par réseau de Petri temporel flou. Rapport LAAS No06125, Doctorat, Université Paul Sabatier, Toulouse, 18 Janvier 2006, 158p.

[Majzik & Bond, 1999] I. Majzik, A. Bondavalli. Automatic dependability modeling of system described in UML. International symposium on software reliability engineering, 1999.

[Manian & al, 1998] R. Manian, J.B. Dugan, D. Coppit, and K.J. Sullivan. Combining various techniques for dynamic fault tree analysis of computer systems. In Proceedings of International Symposium on High Assurance System Engineering, 1998.

[Mayer & Pirri, 1993] M. Cialdea Mayer, F. Pirri. First order abduction via tableau and sequent calculi. Bulletin of the Interest Group in Pure and Applied Logics (IGPL), 1:99-117, 1993.

[Medjoudj & al, 2004] M. Medjoudj, S. Khalfaoui , H. Demmou, R. Valette, A method for deriving feared scenarios in hybrid systems. Probabilistic Safety Assessment and Management (PSAM'7 - ESREL'04), Berlin (Allemagne), 14-18 Juin 2004, 6p.

[Medjoudj, 2006] M. Medjoudj «Contribution à l'analyse des systèmes pilotés par ordinateurs : extraction de scénarios redoutés et vérification de contraintes temporelles», thèse de Doctorat, Université Paul Sabatier, Toulouse, 9 mars 2006.

[Meinadier, 1998] J.P. Meinadier Ingénierie et intégration des systèmes. Hermes 1998.

[Messaadia & Sah, 2007] M. Messaadia, A-E-K Sahraoui., International Journal of Product Development, Vol.4, N°3/4, pp.382-395, 2007.

[MoD-0055P1] Requirements for Safety Related Software in Defence Equipment - 00-55 (Part 2: Guidance), The UK Ministry of Defence, August 1997.

[MoD-0055P2] Requirements for Safety Related Software in Defence Equipment - 00-55 (Part 1: Requirements), The UK Ministry of Defence, August 1997.

[Moncelet 1998] G. Moncelet, «Application des réseaux de Petri à l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile», Thèse de Doctorat, N°3076, Université Paul Sabatier, Toulouse, 9 octobre 1998.

[Moody & Ant, 1998] MOODY J. et ANTSAKLIS P., Supervisory control of discrete event systems using Petri nets, Kluwer, Boston (USA) : 186 p., 1998.

[Muller, 1997] P. A. Muller, " Modélisation objet avec UML" Eyrolles, 2-212-08966-X, 1997.

[Murata, 1989] Murata T Petri Nets: properties, analysis and applications. Proceedings of the IEEE 77(4): 541-580, 1989.

[Naud, 2003] O. Naud, Modélisation hybride pour la supervision de systèmes mécatroniques : application à la stabilité en pente de machines mobiles, Thèse de doctorat, Institut National des Sciences Appliquées : Toulouse (France) : 173 p., 2003.

[Nerode & Kohn, 1993] A. Nerode . et W. Kohn, "Models for hybrid systems: automata, topologies, controllability, observability", in Hybrid Systems, Grossman R., et al. (Edit), Lecture Notes in Computer Science, Springer-Verlag, Berlin (Allemagne). 736 : pp. 317-356, 1993.

[Nerode & Kohn., 1993] A. Nerode . et W. Kohn, "Multiple agent hybrid control architecture", in Hybrid Systems, Grossman R., et al. (Edit), Lecture Notes in Computer Science, Springer-Verlag, Berlin (Allemagne). 736 : pp. 297-316, 1993.

[OMG-1997b] OMG, " UML Notation Guide version 1,1", www.rational.com, 1997.

[Ouardani & al, 2006] A. Ouardani, P. Esteban, M. Paludetto, J.C. Pascal. A meta-modeling approach for sequence diagrams to Petri nets transformation within the requirements validation process. 2006 European Simulation and Modeling Conférence (ESM'2006), Toulouse (France), 23-25 Octobre 2006, pp.345-349.

[Paludetto, 1991] Paludetto M., Sur la commande de procédés industriels : une méthodologie basée objets et réseaux de Petri, thèse présentée au LAAS le 10 décembre 1991.

[Pappas & Sastry, 1996] G. Pappas G et S. Sastry, "Towards continuous abstractions of dynamical and control systems", in Hybrid Systems IV, Antsaklis P., et al. (Edit), Lecture Notes in Computer Science, Springer-Verlag, Berlin (Allemagne). 1273 :, pp. 329-341, 1996.

[Petri, 1962] PETRI C.A., Kommunikation mit Automaten, Ph.D., Institut für Instrumentelle Mathematik, University of Bonn (Allemagne) : 1962.

[Pradin & al, 1999] B. Pradin-Chézalviel, R. Valette, L.A. Künzle: "Scenario duration characterization of t-timed Petri nets using linear logic", IEEE PNPM'99, 8th International Workshop on Petri Nets and Performance Models, Zaragoza, Spain, September 6-10, 1999, p.208-217.

[Ramchandani, 1974] C. Ramchandani, "Analysis of asynchronous concurrent systems by Petri nets," Massachusetts Inst. Technol., Cambridge, Project MAC, TR-120, 1974.

[Rauzy & Dutuit, 1997] A. Rauzy, Y. Dutuit, «Exact and truncated computations of prime implicants of coherent and non coherent fault trees within Aralia», Reliability Engineering & System Safety, Vol. 58, 1997.

[Rivière & al, 2005] N. Rivière, H.Demmou , R. Valette, M. Medjoudj, Symbolic temporal constraint analysis, an approach for verifying hybrid systems. 16th IFAC World Congress, Prague (République Tchèque), 3-8 Juillet 2005, 6p.

[Rivière, 2003] N. Rivière., Modélisation et analyse temporelle par réseaux de Petri et logique linéaire, thèse de l'INSA de Toulouse, le 26 novembre 2003.

[Rugina & al, 2006] Modélisation de la sureté de fonctionnement de systèmes à partir du langage AADL. 15ème Congrès de Maîtrise des Risques et de Sûreté de Fonctionnement (Lambda Mu'15), Lille (France), 9-12 Octobre 2006, 8p.

[Rumbaugh & al, 2004] Rumbaugh J, Jacobson I, Booch G (2004) Unified Modeling Language Reference Manual. 2 ed, Addison-Wesley Longman, Inc. Harlow.

[Sadou & al, 2005] N. Sadou, H. Demmou, J.C. Pascla, R. Valette. Object oriented approach for deriving feared scenarios in hybrid systems. 2005 European Simulation and Modelling Conference, Porto (Portugal), 24-26 Octobre 2005, pp.572- 578.

[Sadou & al, 2006a] N. Sadou, H. Demmou, J.C. Pascal, R.valette. Fiabilité dynamique des systèmes hybrides : approche basée scénarios. Conférence Internationale Francophone d'Automatique (CIFA'2006), Bordeaux (France), 30 Mai - 1er Juin 2006, 6p.

[Sadou & al, 2006b] N. Sadou, H. Demmou, J.C. Pascal, R.valette. Continuous dynamic abstraction for reliability and safety analysis of hybrid systems. 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'2006), Beijing (Chine), 30 Août - 1er Septembre 2006.

[Sadou & al, 2006c] N. Sadou, H. Demmou. Minimality of critical scenarios in Petri net models. 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei (Taiwan), 8-11 Octobre 2006, pp.3422-3429.

[Sadou, 2007] N. Sadou, Multi-model approach for dependability analysis of hybrid system. 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2007), Edimbourg (UK), 25-28 Juin 2007, pp.297-299.

[SAE-AS5506] "Architecture Analysis and Design Language," Society of Automotive Engineers, Warrendale, PA 2004.

[Sahraoui & al, 2004] A.E.K.Sahraoui; D.M.Buede; and A.P.Sage: "Issues for systems engineering research".14th Annual International Symposium INCOSE 2004, Toulouse (France), 11p, 20-25 June 2004.

[Schoenig, 2004] R. Schoenig, «Définition d'une méthodologie de conception des systèmes mécatroniques sûrs de fonctionnement», Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, 26 octobre 2004.

[Sinnamon & al, 1997] R.M. Sinnamon & J.D. Andrews, «New approaches to evaluating fault trees», Reliability Engineering & System Safety, Vol. 58, 1997.

[Sofia] logiciel développé par la société SGTE Sofreten de génération automatique d'arbre de défaillance et de l'AMDE associée.

[STA-IEEE] http://www.afis.fr/doc/normes/normes2.htm.

[Stiver & al., 1996] J. Stiver, P. Antasaklis et M. Lemmon , "A logical DES approach to the design of hybrid control systems", Mathematical and Computer Modelling, 23(11-12) : pp. 55-76, 1996.

[Struss, 2002] P. Struss. "Automated abstraction of numerical simulation models - Theory and practicalexperience", proceedings of Model Based Systems and Qualitative Reasoning for Intelligent Tutoring Systems, San Sebastian, (Espagne) : pp. 161-168, 2002.

[Travé & al., 1997] L. Travé-MassuyèsL, P Dague et F. Guerrin, Le raisonnement qualitatif pour les sciences de l'ingénieur, Diagnostic et maintenance, Hermès Paris (France) : 505 p., 1997.

[Valentin, 2000] Hybrid Dynamic System verification with Mixed Petri Nets. The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic System, Dortmung, Shaker Verlag, Aachen, pp. 231-236.

[Valette, 2002] R. VALETTE, Les réseaux de Petri, LAAS-CNRS, Toulouse (France) : 86 p., 2002, http://www.laas.fr/~robert/.

[Valk, 1998] Valk R., Petri nets as token objects: an introduction to elementary object nets. Lecture Notes in Computer Science 1420: 1-25.

[Vardi & Wolper, 1986] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program. In Symposium on Logic in Computer Science (LICS'86),Washington, D.C., USA, June 1986. IEEE Computer Society Press.

[Vardi, 1996] M. Vardi. An automata-theoretic approach to Linear Temporal Logic, volume 1043 of Lecture Notes in Computer Science, pages 238–266. Springer-verlag edition, 1996.

[Villani & al, 2004] E.VILLANI , J.C.PASCAL , P.E.MIYAGI , R.VALETTE. Object oriented approach for cane sugar production: modelling and analysis. Control Engineering Practice, Vol.12, N°10, pp.1279-1289, Octobre 2004.

[Villani & al, 2007] E.Villani, P.E.Miyagi , R.Valette. Modelling and analysis of hybrid supervisory systems. A Petri net approach. Springer, N°ISBN 978-1-84628-650, 2007, 224p.

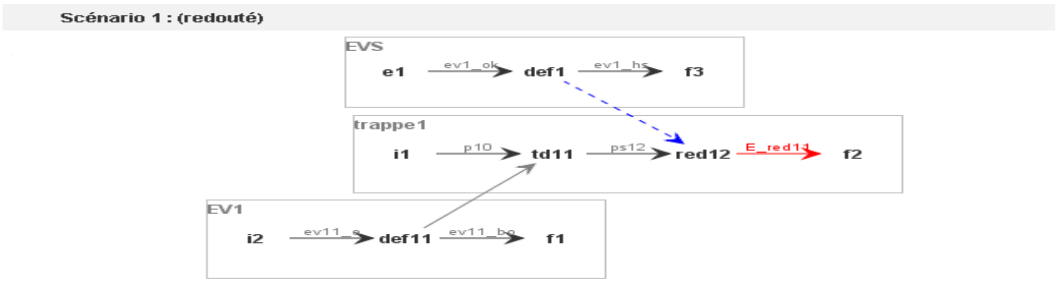
[Villemeur, 1988] A. Villemeur, «Sûreté de fonctionnement des systèmes industriels», collection de la Direction des Etudes et Recherche d'Electricité de France, 1988.

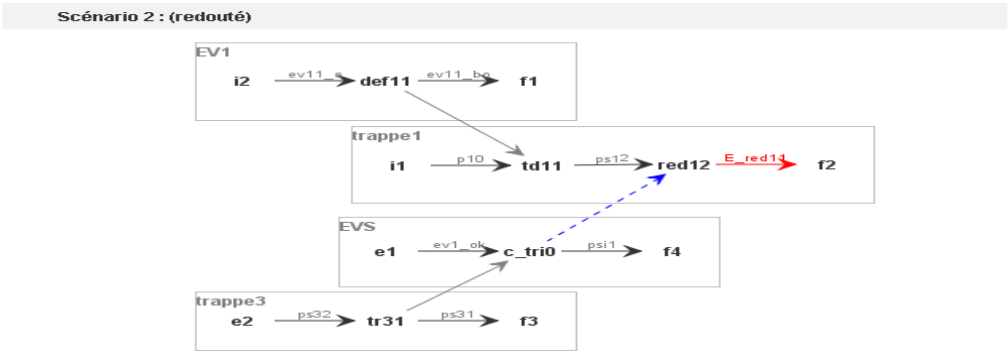
[Wieting, 1996] Wieting R (1996) Hybrid high-level nets. The 28th Winter Simulation Conference, Coronado, pp 848-855.

[Zaytoon, 2001] J. Zaytoon, «Systèmes dynamiques hybrides», ouvrage collectif sous la direction de, collection Hermes Science, ISBN 2-7462-0247-6.

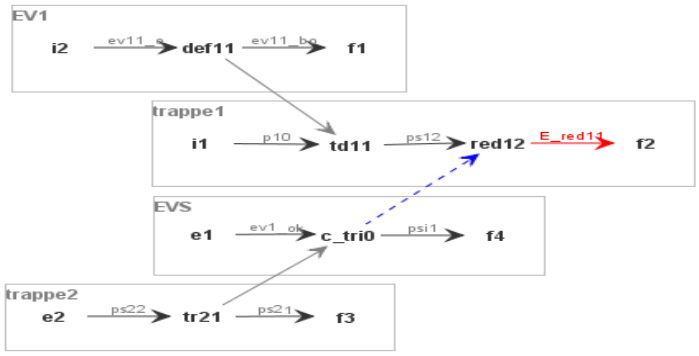
Annexe

Les différentes figures représentent les graphes de précédence associés aux scénarios générés de l'exemple de la trappe (chapitre V). Pour résumer l'information fournie par les graphes, la trappe 1 se bloque lorsque son électrovanne ev_1 est défaillante et lorsque l'électrovanne de secours ev_S est soit défaillante, soit déjà utilisée par les autres trappes. Cette dernière pouvant être réquisitionnée par la trappe 2 ou la trappe 3 si leurs électrovannes respectives sont défaillantes. A toutes ces possibilités, il faut également combiner le fait que la défaillance d'une électrovanne peut survenir avant le début de l'ouverture de la trappe ou pendant son ouverture.

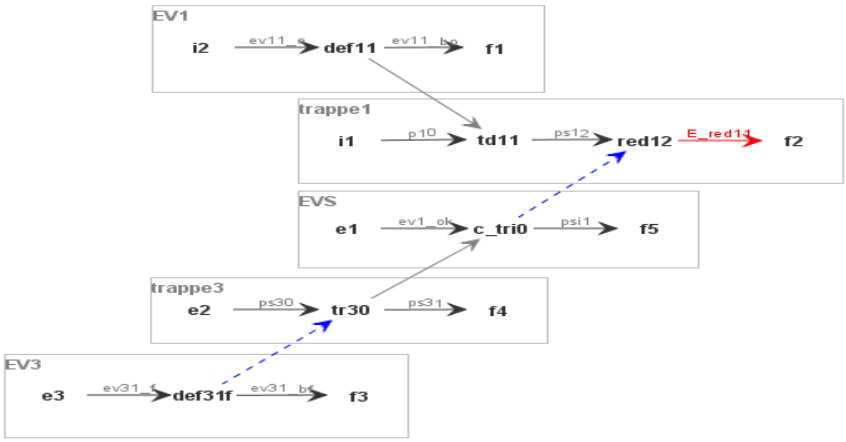




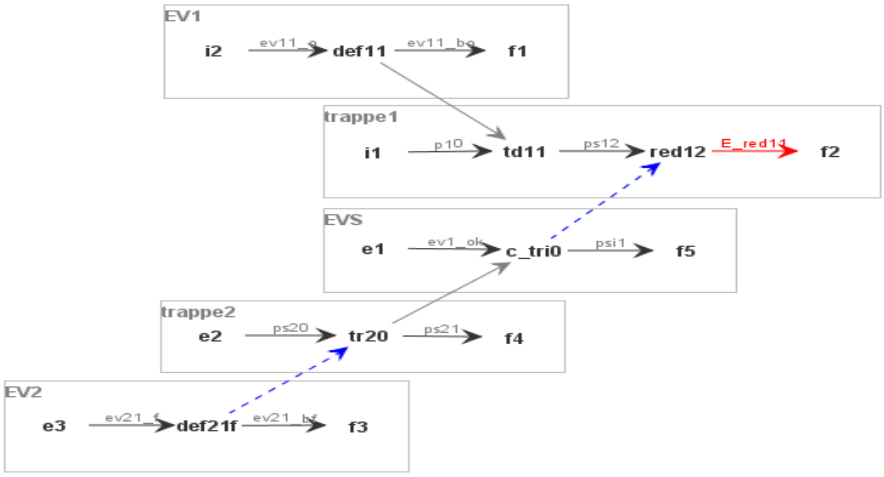
Scénario 3 : (redouté)



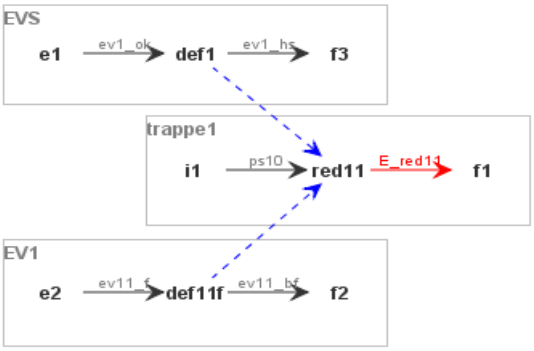
Scénario 4 : (redouté)



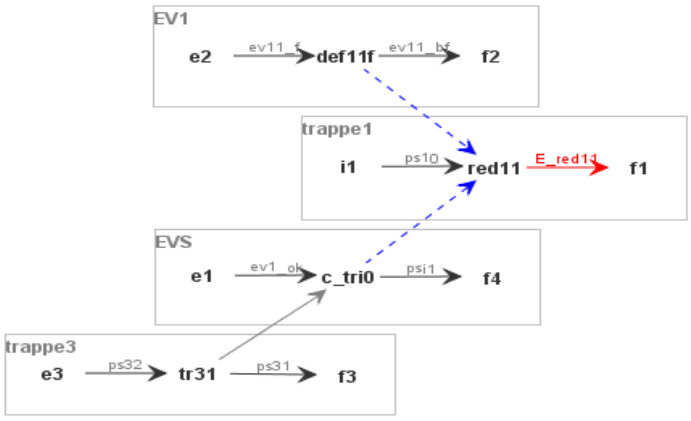
Scénario 5 : (redouté)



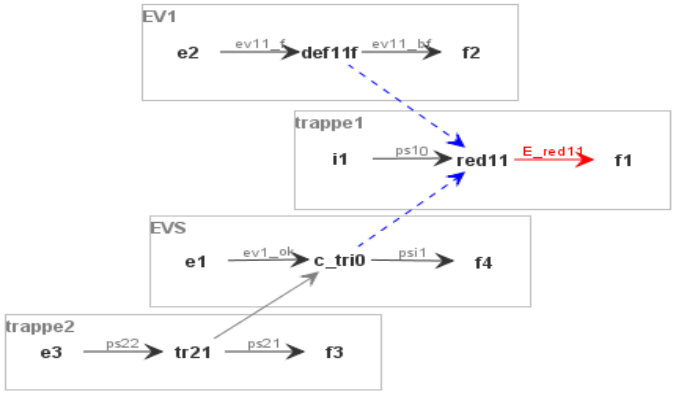
Scénario 6 : (redouté)



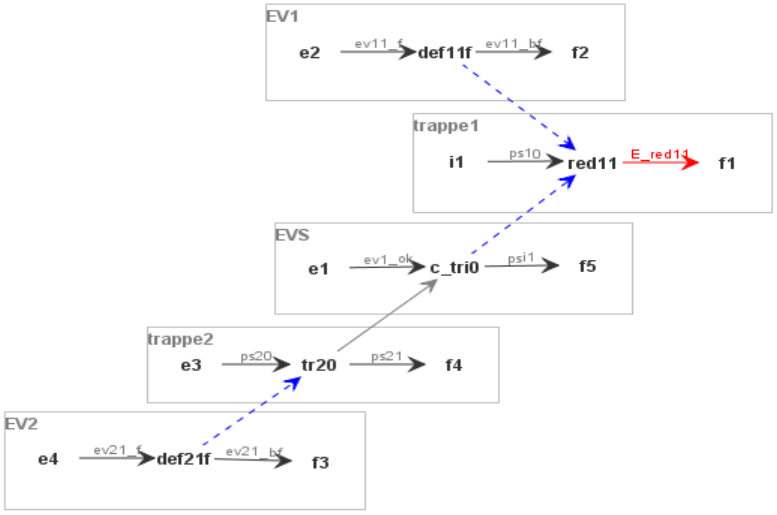
Scénario 7 : (redouté)



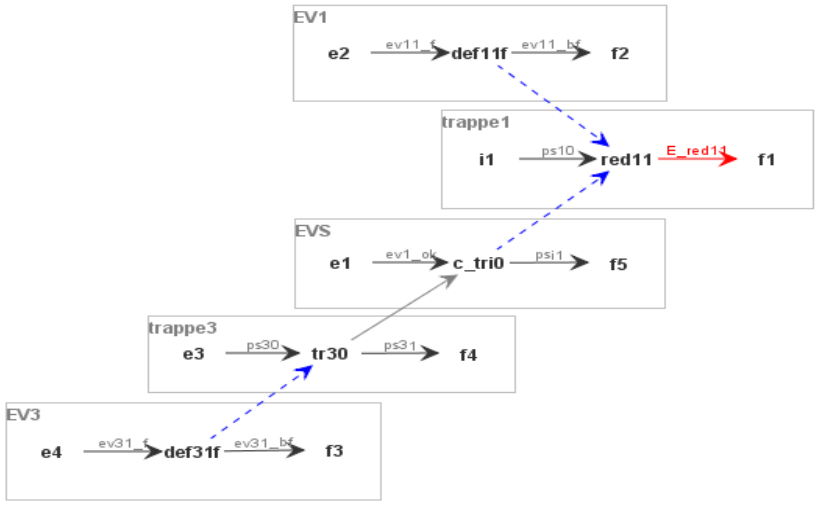
Scénario 8 : (redouté)



Scénario 9 : (redouté)



Scénario 10 : (redouté)



Aide à la conception des systèmes embarqués sûrs de fonctionnement

Résumé : L'avancée technologique que les systèmes embarqués ont connue lors de ces dernières années les rend de plus en plus complexes. Ils sont non seulement responsables de la commande des différents composants mais aussi de leur surveillance. A l'occurrence d'événement pouvant mettre en danger la vie des utilisateurs, une certaine configuration du système est exécutée afin de maintenir le système dans un état dégradé mais sûr. Il est possible que la configuration échoue conduisant le système dans un état appelé « état redouté » avec des conséquences dramatiques pour le système et l'utilisateur. La description des scénarios qui mènent le système vers l'état redouté à partir d'un état de fonctionnement 'normal' permet de comprendre les raisons de la dérive afin de prévoir les configurations nécessaires qui permettent de les éviter

Dans notre approche d'analyse de sûreté de fonctionnement des systèmes dynamiques, les scénarios sont générés à partir d'un modèle réseau de Petri. En s'appuyant sur la logique linéaire comme nouvelle représentation (basée sur les causalités) du modèle réseau de Petri, une analyse qualitative permet de déterminer un ordre partiel de franchissement des transitions et ainsi extraire les scénarios redoutés. La démarche est focalisée sur les parties du modèle intéressantes pour l'analyse de fiabilité évitant ainsi l'exploration de toutes les parties du système et le problème de l'explosion combinatoire.

L'objectif final consiste en la détermination de scénarios minimaux. En effet, un scénario peut bien mener vers l'état redouté sans qu'il soit minimal. Il contient des événements qui ne sont pas strictement nécessaires à l'obtention finale de l'état critique redouté. De même que la notion de coupe minimale a été définie dans le cadre des arbres de défaillance, nous proposons une définition de ce qu'est un scénario minimal dans le cas des réseaux de Petri.

Pour prendre en compte La nature hybride des systèmes, nous avons développé un simulateur hybride basé sur le couplage de l'algorithme de génération de scénarios redoutés avec un solveur d'équations différentielles. L'algorithme se charge de la partie discrète modélisée par le réseau de Petri et le solveur d'équations de la partie continue modélisée par un ensemble d'équations différentielles.

Afin d'avoir une approche système pour l'analyse de la sûreté de fonctionnement, nous proposons une approche qui permet de prendre en compte les exigences de sûreté dans le processus d'ingénierie des exigences qui permet d'établir un modèle de traçabilité afin de s'assurer de la prise en compte de ces exigences tout au long du cycle de vie du système. L'approche est basée sur une norme de l'ingénierie système, en l'occurrence l'EIA-632.

Mots clés — *Systèmes hybrides, réseaux de Petri, logique linéaire, Concepts orientés objets, Sûreté de fonctionnement, Scénarios redoutés, Simulation, Ingénierie système, EIA-632.*

Design of dependable embedded systems

Abstract: Embedded systems run the computing devices hidden inside another larger system or product. Embedded systems have the charge of controlling various types of sub-systems; they are also in charge of the monitoring of the whole system and coordination with other systems. This means that when some event affecting the safety of the system occurs, a reconfiguration action is executed in order to maintain the system in a safe degraded state. If the reconfiguration fails then the system will reach a feared (dangerous) state with dramatic consequences for users. So it is important to understand how the system reaches such feared states to set up the reconfiguration actions.

In our approach for safety analysis of dynamic systems, feared scenarios are derived from Petri net model. Based on linear logic as new representation (using the causality relations) of the Petri net model, a qualitative analysis allows to determine a partial order of transition firings and thus, to extract feared scenarios. The analysis is focalized on the parts of the model that are interesting for the reliability analysis, avoiding exploration of the global system and the problem of the state space explosion.

The final objective is to determine all minimal scenarios. Indeed, one scenario can lead to a feared state and contain events which are not strictly necessary to reach the final feared state. By analogy with the concept of minimal cutsets for the fault trees, we define the concept of minimal scenario in Petri net model.

To take into account the hybrid nature of systems, we developed a hybrid simulator which combines the deriving feared scenarios algorithm and differential equations solver. The algorithm is in the charge of the discrete part modelled by Petri net and the solver of the continuous part modelled by a set of differential equations.

In order to have a system approach for dependability analysis, we propose an approach which allows taking into account the safety requirements in the requirement engineering process. It makes possible the establishment of the traceability in order to make sure of taking into account of the safety requirements throughout the life cycle of the system. The approach is based on *EIA-632* engineering system standard.

Key words — *Hybrid systems, Petri net, Linear logic, Oriented objects concepts, Dependability, Feared scenarios, Simulation, System engineering, EIA-632.*