Contribution à l'analyse de sûreté de fonctionnement des systèmes complexes en phase de conception : application à l'évaluation des missions d'un réseau de capteurs de présence humaine

Pierre David

UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE SCIENCES ET TECHNOLOGIES

Institut PRISME

THÈSE présentée par :

Pierre DAVID

soutenue le : **9 novembre 2009**

pour obtenir le grade de : **Docteur de l'université d'Orléans**

Discipline : Sciences et Technologies Industrielles

Contribution à l'analyse de sûreté de fonctionnement des systèmes complexes en phase de conception : application à l'évaluation des missions d'un réseau de capteurs de présence humaine

THÈSE dirigée par :

Frédéric KRATZ	Professeur des Universités, ENSIB

RAPPORTEURS :

Jean-François AUBRY	Professeur des Universités, INPL/ENSEM
Jean-Jacques LESAGE	Professeur des Universités, ENS Cachan

JURY :

Youssoufi TOURE	Professeur des Universités, Université d'Orléans, Président
Jean-François AUBRY	Professeur des Universités, INPL/ENSEM
Jean-Jacques LESAGE	Professeur des Universités, ENS Cachan
Emmanuel ARBARETIER	EADS APSYS
Stéphane JUILLOT	Wirecom Technologies
Antoine RAUZY	Dassault Systèmes
Vincent IDASIAK	Maître de Conférences, ENSIB
Frédéric KRATZ	Professeur des Universités, ENSIB

Remerciements

Les travaux présentés dans ce mémoire ont été réalisés au Laboratoire Vision et Robotique (LVR) intégrant par la suite l'Institut PRISME et ont été accueillis au sein de sa partie située à l'Ecole Nationale Supérieure d'Ingénieurs de Bourges (ENSIB). Ils ont été soutenus par le pôle de compétitivité S2E2, à travers le projet CAPTHOM.

Je tiens en premier lieu à remercier Monsieur Youssoufi Touré, Professeur des Universités, Directeur successivement du LVR et de l'Institut PRISME, de m'avoir accueilli au sein de son laboratoire et d'avoir accepté d'être présent dans le jury. Je remercie également Monsieur Thierry Allard, Président du pôle S2E2, d'avoir permis la réalisation de ces travaux. Je remercie également Monsieur Joël Allain, pour la mise à disposition des moyens de l'ENSIB et de m'avoir ainsi permis de concrétiser ces travaux dans les meilleures conditions qui soient.

Ces travaux ont été dirigés par Monsieur Frédéric Kratz, Professeur des Universités à l'ENSIB et Monsieur Vincent Idasiak, Maître de Conférences à l'ENSIB, à qui je souhaite exprimer toute ma reconnaissance pour ces années de travail extrêmement enrichissantes, rigoureuses et conviviales. Je tiens à les remercier pour leur disponibilité, leurs grandes compétences scientifiques et leur soutien tout au long de cette thèse. Merci tout spécialement à Frédéric pour son encadrement 'à la virgule près'. Merci également à Joe Dariak pour cette traversée du Mékong qui restera pour moi comme un témoignage d'amitié et une aventure humaine fondatrice.

Je remercie Messieurs Jean-François Aubry et Jean-Jacques Lesage, respectivement Professeur des Universités à l'Ecole Nationale Supérieure d'Electricité et de Mécanique de l'Institut National Polytechnique de Lorraine et Professeur des Universités à l'Ecole Nationale Supérieure de Cachan, qui ont accepté la charge de rapporteur. Ils ont également accepté de participer à mon jury, qu'ils trouvent ici l'expression de ma plus profonde reconnaissance.

Je remercie également l'ensemble des membres du jury qui ont accepté d'examiner avec attention ces travaux : Monsieur Emmanuel Arbaretier, Responsable du pôle activité logicielle chez EADS/APSYS, Monsieur Stéphane Juillot, Directeur Recherche et Développements chez Wirecom Technologies et Monsieur Antoine Rauzy, Responsable CATIA Systems chez Dassault Systèmes.

Je souhaite remercier Monsieur Yves Parmentier, animateur du pôle Capteurs et Automatismes, ainsi que tous les personnels de l'Institut PRISME et de l'ENSIB pour la disponibilité et l'extrême compétence dont ils ont fait preuve, afin de rendre le déroulement de cette thèse très efficace et agréable. Je pense notamment à Madame Laure Spina dont l'aide m'a été particulièrement précieuse.

J'adresse aussi mes remerciements à tous les membres du projet CAPTHOM pour leur participation, réflexions, conseils et intérêt pour ces travaux.

Mes remerciements chaleureux vont vers les membres de l'équipe MCDS. Je souhaite également exprimer ma reconnaissance à tous les occupants du couloir pour la chaleureuse ambiance. Merci donc à Hélène, Serge, Adel et Benoit.

Je souhaite exprimer mes sincères remerciements à tous les docteurs et doctorants de l'Institut PRISME que j'ai pu côtoyer au cours de ces trois années. Je pense spécialement à Hazem, Hamid, Guillaume, Baptiste, Davidou et Rudi (aka Gunther). J'exprime une pensée particulière à mes collègues du projet CAPTHOM : Antoine, Yannick et Damien. Merci à Antoine pour ta classe, ton optimisme et ta présence (humaine assurément). Merci à Yannick pour ta juste 'vision' des choses, ton soutien et ton amitié. Mention spéciale pour Damien, dont j'ai eu l'honneur de partager le bureau pendant ces trois ans, tu es bien plus qu'un défouloir : un véritable ami !

Je remercie mes amis qui m'ont suivi de près et apporté leur soutien pendant ces trois années. Un grand merci à toute ma famille pour leur soutien et leur attention et une pensée très particulière pour mes grands-mères. Je remercie également toute la famille de Claire. Merci à Lucie, Vincent, Xavier, Julie, Bernard et Viviane. Merci à mon frère Damien et à Sandrine. Merci à mes parents d'être là depuis le début, à chaque pas de plus et pour leur amour inconditionnel. Enfin, mes plus grands remerciements vont à Claire. C'est grâce à toi que cela a été possible, tu sais tout ce que tu représentes à mes yeux. C'est à toi que je dédie ce mémoire.

Table des matières

Table des figures

Table des Tableaux

Glossaire

AADL : Architecture Analysis & Design Language
AD : Activity Diagram/Diagramme d'Activité
AdD : Arbre de Défaillance
AFIS : Association Française d'Ingénierie Système
AltaRica DF : AltaRica Data Flow
AMDEC : Analyse des Modes de Défaillance de leurs Effets et Criticité
ANSI : American National Standards Association
APR : Analyse Préliminaire des Risques
ATL : ATLAS Transformation Language
BCD : Base de données des Comportements Dysfonctionnels
BDD : Block Definition Diagram
BDMP : Boolean logic Driven Markov Processes
CAPTHOM : CAPTeur de présence HUMaine
CdC : Cahier des Charges
CM : Chaîne de Markov
COB : Composable Objects
COTS : Component On The Shelf
DRBD : Dynamic Reliability Block Diagram
DBF : Diagramme Blocs de Fiabilité
EDS : Environnement de Développement Système
E/E/EP : Systèmes Electriques/Electroniques/à Electronique Programmable
EIA : Electronic Industries Alliance
FMDS : Fiabilité Maintenabilité Disponibilité et Sécurité
FMEA/FMECA : Failure Modes and Effects (and Criticity) Analysis
FSS : FMECA Support System
IBD : Internal Block Diagram
IEC : International Electrotechnical Commision
IEEE : International Electronic and Electrical Engineers
INCOSE : INternational Council On System Engineering
IR : InfraRouge
IS : Ingénierie Système
ISBM : Ingénierie Système Basée sur les Modèles
MBM : Méthode Basée sur les Modèles
MBSE : Model Based System Engineering
MdD : Mode de Défaillance
MéDISIS : Méthode D'Intégration des analyses de Sûreté de fonctionnement au processus d'Ingénierie Système
MMB : Model and Mission Builder
OCL : Object Constraint Language
OMG : Object Management Group
OO : Orienté Objet
OOSEM : Object-Oriented System Engineering Method
PFS : Practical Formal Specification
PIR : Détecteur Pyroélectrique
QFE : Qualitative FMEA Engine
RD : Requirement Diagram/Diagramme d'exigences
RdP : Réseau de Petri

RdPSG : Réseau de Petri Stochastique Généralisé
RdPT : Réseau de Petri Temporel
REX : Retour d'Expérience
RPN : Risk Priority Number
S2E2 : Sciences et Systèmes de l'Energie Electrique
SD : Sequence Diagram/Diagramme de Séquence
SdF : Sûreté de Fonctionnement
SNOOPS : Sensor Networks : Optimization Of Placement by Simulation
SysML : System Modeling Language
UML : Unified Modeling Language
V&V : Vérification et Validation
XML : eXtensible Markup Lamguage
XMI : XML Metadata Interchange

Introduction Générale

Si l'on étudie les systèmes à fortes contraintes de Sûreté de Fonctionnement, on constate qu'ils ne cessent de se complexifier. Cette tendance se vérifie aussi bien pour la réalisation de systèmes de sécurité, que pour celle de transport de personnes ou de systèmes d'exploration spatiale. Cela transparaît notamment à travers la complexification des interfaces et composants remplissant les missions confiées aux systèmes. On note premièrement une augmentation significative de la taille de ces systèmes en terme de nombre de composants utilisés et une hétérogénéité accrue des technologies employées. En effet, la miniaturisation permet l'intégration d'un nombre important d'éléments mélangeant par exemple électronique, mécanique, hydraulique et logiciel.

De plus, le contexte actuel fortement concurrentiel, influe sur la conception de tels systèmes. Ceci participe à motiver la recherche de systèmes offrant de nouvelles fonctionnalités et améliorant les performances de l'existant, encourageant par la même la complexification des solutions développées. Cette concurrence impose également des délais et des coûts de réalisation toujours plus bas. Le défi des concepteurs est donc de maîtriser la complexité du système en un temps minimal tout en garantissant les performances exigées. Cette problématique est très présente dans le projet CAPTHOM. Ce projet, porté par le pôle de compétitivité S2E2[1], a pour but de développer un capteur innovant de présence humaine. Ce cadre de recherche a motivé les travaux présentés au cours de cette thèse.

La validation des performances est primordiale pour les systèmes à fortes contraintes de Sûreté de Fonctionnement. En effet, ces derniers sont soumis au respect de normes de performances précises, comme les SIL (Safety Integrity Level), et le non respect de tels objectifs conduit inévitablement à la non viabilité du système développé. Ces systèmes nécessitent donc une phase importante d'analyse de leur Sûreté de Fonctionnement et de validation de leur comportement global en matière de non atteinte d'états redoutés ou d'efficacité minimale. Le développement de ces systèmes nécessite donc l'emploi de techniques rigoureuses d'Ingénierie Système, ainsi que de pratiques avancées d'analyse de Sûreté de Fonctionnement afin de maîtriser tout au long de la conception, la complexité et les performances du système.

La démarche d'Ingénierie Système adoptée pour leur conception doit permettre de gérer toutes les tâches nécessaires au développement du système, de la détermination de l'architecture fonctionnelle à réaliser, à la validation des choix opérés et des performances. Les analyses de Sûreté de Fonctionnement doivent prendre une part importante de ce cycle de développement. Or, ce type de travail prend un temps non négligeable et requiert une forte expertise du domaine tant les techniques et formalismes à employer sont complexes. Le défi qui se dégage, est donc de favoriser l'interconnexion des activités de conception pures et des étapes d'étude de la Sûreté de Fonctionnement. Pour réussir cela, il faut chercher à comprendre les techniques employées pour ces deux facettes de la création des systèmes et en particulier analyser comment unifier et faire coopérer les outils employés par chaque communauté.

En effet, de tels développements ne se font sans l'utilisation de méthodes rigoureuses supportées par les formalismes et outils adéquats. L'utilisation de l'Ingénierie Système basée sur

[1] Le projet CAPTHOM regroupe au sein du pôle de compétitivité S2E2 (Sciences et Systèmes de l'Energie Electrique) : Université d'Orléans, Pôle Capteurs et Automatismes, Cresitt industrie, Legrand, Thermor, ST Microelectronics, Sorec, Agilicom, Wirecom technologies. http://www.s2e2.fr.

les Modèles est primordiale pour des projets de cette envergure. Nous considérons dans cette thèse les méthodes d'Ingénierie Système basées sur les modèles, comme la meilleure alternative d'ingénierie pour le développement des systèmes complexes à fortes contraintes de Sûreté de Fonctionnement. Nous avons donc cherché à dresser une méthodologie d'intégration des analyses de Sûreté de Fonctionnement au processus d'Ingénierie Système basée sur les modèles, que nous avons intitulée MéDISIS. Le but recherché est celui appelé de leurs vœux par les auteurs de [Rauzy et al. 2008] : « [dissoudre] les frontières entre les environnements de développement virtuel au service des Concepteurs, et les référentiels de simulation accompagnant le travail des Fiabilistes, Logisticiens et Risk Managers, […] pour laisser place à une discipline commune que l'on pourrait qualifier d'Assurance Performentielle… ».

Durant ces dernières années, la communauté des ingénieurs système a cherché à se doter d'un langage pouvant être utilisé comme modèle central au processus d'Ingénierie Système. Ce langage doit permettre de créer des modèles exprimant des concepts inhérents à toutes les technologies employées en Ingénierie Système. Le modèle doit servir également de point d'ancrage et orchestrer toutes les étapes d'analyses, de spécification, de conception, de vérification et validation du système. De ces réflexions est né le langage SysML (System Modelling Language). Il constitue à ce jour le langage le mieux adapté à une Ingénierie Système globale et transversale. C'est pourquoi, nous nous sommes focalisés sur l'analyse de Sûreté de Fonctionnement de systèmes dont les modèles sont exprimés en SysML. Dans ce travail nous supposons que les modèles que nous étudions ne sont le fruit d'aucune analyse de Sûreté de Fonctionnement préliminaire.

Nous avons étudié les relations entre les modèles SysML et les études de Sûreté de Fonctionnement sous différents aspects. Nous avons cherché à comprendre comment des modèles SysML fonctionnels pouvaient être analysés pour déduire le comportement dysfonctionnel du système. Nous avons étudié comment exprimer les phénomènes dysfonctionnels dans ce langage et comment se servir de tels modèles pour construire des bases de connaissances utilisables au cours de l'Ingénierie Système. Enfin, nous avons analysé la traduction et les passerelles pouvant exister entre SysML et les formalismes et langages spécifiques aux analyses de Sûreté de Fonctionnement. Pour ces différents travaux nous avons également considéré l'aide que des services d'analyses automatiques des modèles réalisant ces tâches pouvaient apporter aux parties prenantes du développement du système. Ceci dans le but de participer à la création d'un outil ou groupe d'outils commun au processus global d'Ingénierie Système. Dans nos travaux, nous supposons disposer en entrée, des modèles de description fonctionnelle du système, réalisés en SysML, et nous proposons d'obtenir à partir de cela l'expression de son comportement dysfonctionnel et de réaliser sa validation quantifiée.

Afin de traiter cette problématique, cette thèse est organisée de la façon suivante :

- Dans le chapitre I, nous présentons la problématique de la conception des systèmes complexes à fortes contraintes de Sûreté de Fonctionnement. Nous exposons les grands principes régissant l'Ingénierie Système, puis nous nous focalisons sur ceux de l'Ingénierie Système basé sur les modèles. Nous montrons notamment comment peuvent s'organiser autour d'un modèle commun les activités de conception préliminaire, d'analyse, de validation et de conception détaillée. Nous présentons ensuite le langage SysML, capable de constituer le langage support à l'Ingénierie Système basée sur les modèles. Nous énonçons ensuite les étapes nécessaires à l'analyse de Sûreté de Fonctionnement des systèmes afin de cibler les phases à intégrer dans le processus d'Ingénierie Système pour étudier les systèmes complexes à fortes contraintes de Sûreté de Fonctionnement. Nous réalisons un état de l'art

sur les travaux d'optimisation des études de type AMDEC (Analyse des Modes de Défaillance de leurs Effets et Criticité), afin de définir quels efforts sont à mener pour les intégrer au mieux à l'Ingénierie Système. Puis nous réalisons un état de l'art sur les exploitations faites des langages similaires à SysML dans le contexte de l'analyse de Sûreté de Fonctionnement des systèmes et notamment leur utilisation conjointe avec les formalismes spécifiques à ces analyses. Ces études bibliographiques nous permettent de dégager les phases et construction nécessaires à la constitution d'une méthode d'intégration des analyses de Sûreté de Fonctionnement à l'Ingénierie Système. Nous présentons alors les grandes lignes de la méthode MéDISIS, détaillée des chapitres II et III.

- Le chapitre II est consacré à l'étude de la réalisation d'AMDEC à partir de modèles fonctionnels du système. Ce type d'analyse est primordial dans le processus d'analyse de Sûreté de Fonctionnement car il permet d'identifier les risques pesant sur l'architecture proposée. Dans un premier temps, nous analysons la démarche intellectuelle réalisée au cours d'une AMDEC, afin d'étudier sa transposition sur l'analyse d'un modèle SysML. Cette réflexion est menée en considérant le métamodèle du raisonnement conduit lors d'une AMDEC et celui du langage SysML. Cette étude nous permet de formuler des règles d'analyse des modèles SysML permettant d'extraire l'information nécessaire à l'obtention d'une AMDEC préliminaire, préparant les décisions finales des experts de Sûreté de Fonctionnement. Nous évoquons également les possibilités d'automatisation de tels traitements dans l'optique de la création d'un outil d'analyse automatique des modèles SysML. L'ensemble de ces raisonnements fait appel à l'utilisation d'une base de connaissance des comportements dysfonctionnels écrite en SysML, dont nous présentons une première vue de l'organisation et de l'utilisation.

- Dans le chapitre III, nous exposons une connexion de SysML au langage AltaRica Data Flow. Ce dernier est un langage très puissant et populaire pour la réalisation d'études de Sûreté de Fonctionnement. Nous rappelons sa sémantique, avant d'étudier les moyens de traduction des modèles fonctionnels SysML vers le modèle AltaRica Data Flow support de l'étude de Sûreté de Fonctionnement. Cette traduction passe également par l'utilisation de la Base de Connaissances des comportements Dysfonctionnels, dont la version complète est alors détaillée à travers la présentation de son métamodèle suivant la spécification SysML. Les méthodes d'automatisation de cette transformation sont également évoquées.

- Le chapitre IV propose une application des travaux de cette thèse au projet CAPTHOM. Dans ce cadre, nous avons expérimenté l'usage de SysML et de la Méthode MéDISIS. Nous présentons dans le chapitre la problématique de CAPTHOM et la réalisation d'un modèle SysML reflétant les réflexions menées par les différents groupes de travail. L'utilisation de MéDISIS sur ce modèle nous a permis de dégager les besoins de simulation du projet. Nous présentons alors la solution logicielle développée à cet effet et sa place dans l'environnement de développement de CAPTHOM, dont le modèle SysML est le point central. Cette application montre la capacité de MéDISIS à intégrer les études de Sûreté de Fonctionnement à l'Ingénierie Système. De plus, ce cas d'étude illustre son aptitude à diriger le processus de conception par les résultats des analyses de Sûreté de Fonctionnement.

Chapitre I

Les défis du développement des systèmes à fortes contraintes de Sûreté de Fonctionnement

Résumé du Chapitre I :

Ce chapitre présente les développements actuels en matière d'Ingénierie Système destinés à la conception de systèmes complexes à fortes contraintes de Sûreté de Fonctionnement. Les démarches propres à l'Ingénierie Système Basée sur les Modèles sont mises en avant, en insistant sur le rôle du langage de modélisation qu'elles emploient et sur la constitution d'un ensemble d'outils formant l'Environnement de Développement Système venant les soutenir. Nous montrons que SysML est un langage parfaitement adéquat pour traiter la conception des systèmes que nous ciblons. En présentant les pratiques nécessaires à l'analyse de Sûreté de Fonctionnement d'un système, nous mettons en avant les besoins d'intégration de ces techniques à l'Ingénierie Système Basée sur les Modèles. L'état de l'art sur les travaux abordant cette problématique, aussi bien en terme d'analyses qualitatives que quantitatives, nous conduit à proposer notre propre méthodologie d'utilisation des techniques d'analyse de la Sûreté de Fonctionnement au sein du processus d'Ingénierie Système. Cette méthodologie, nommée MéDISIS, s'appuie sur l'analyse de modèles décrits en SysML et la formation d'une Base de connaissances des Comportements Dysfonctionnels rédigée en SysML.

I.1 Introduction

La conception de systèmes complexes, que nous considérerons dans cette thèse comme un ensemble de composants interagissant entre eux, est depuis ces dernières années un sujet au centre de toutes les préoccupations dans la communauté des ingénieurs système. Les exigences actuelles, en matière de développement de nouveaux systèmes, demandent une révolution des pratiques traditionnelles de conception. Les nouveaux systèmes destinés à être introduits sur le marché, doivent proposer toujours plus de nouvelles fonctionnalités et de meilleures performances dans un délai de réalisation plus court et pour des coûts restreints. Pour répondre à ces exigences, ces systèmes font appel à de multiples technologies, posant ainsi de nouveaux défis pour leur création. Comment intégrer et interfacer des composants de natures différentes ? Comment évaluer et valider de tels systèmes ? Comment organiser le travail d'équipes de spécialités différentes ? Comment assurer la communication entre elles en tenant compte de la diversité des cultures et des priorités parfois antagonistes ? C'est à ces problèmes que les recherches en Ingénierie Système veulent apporter des solutions. Les travaux présentés dans cette thèse se concentrent sur les phases d'analyse du système venant valider les avancées de la conception. A ce titre, nous mettons l'accent dans ce travail sur la construction des liaisons existant dans les processus d'Ingénierie Système et plus particulièrement sur le passage d'un modèle à son analyse. La première partie de ce chapitre sera consacrée à l'introduction des concepts utilisés en Ingénierie Système. Une revue des processus et méthodes actuellement utilisés en Ingénierie Système sera présentée. Nous nous focaliserons sur l'Ingénierie Système Basée sur les Modèles (ISBM[2], expression tirée de l'anglais *Model Based System Engineering* MBSE) qui constitue à ce jour la meilleure alternative pour répondre aux problèmes abordés dans cette thèse. Nous ciblerons plus précisément les phases de qualification du besoin d'analyse et leur place dans le processus.

Le processus d'Ingénierie Système ordonnance les tâches à réaliser pour obtenir le système désiré. Il intègre donc des phases très importantes d'évaluation des solutions proposées. Parmi ces évaluations nous nous intéressons, dans le cadre du développement de systèmes à fortes contraintes de Sûreté de Fonctionnement, aux analyses permettant de qualifier et quantifier le comportement dysfonctionnel des systèmes. La réalisation des études de Sûreté de Fonctionnement constitue un domaine d'expertise à part entière et la question de leur intégration à un processus d'Ingénierie Système reste complexe. Nous avons donc étudié comment intégrer ces pratiques aux processus d'Ingénierie Système déployés par une Méthode Basée sur les Modèles. Dans la seconde partie de ce chapitre, nous présentons les analyses de Sûreté de Fonctionnement menées lors de la réalisation d'un produit à fortes contraintes de Sûreté de Fonctionnement. Ce tour d'horizon s'attarde sur les méthodes d'analyse de risques et de recensement de Modes de Défaillance, puis sur les méthodes, langages et formalismes employés pour quantifier et valider les comportements dysfonctionnels. Nous aborderons par la suite des considérations d'ordre méthodologique portant sur la préparation de ces études et l'exploitation des résultats d'analyse. En conclusion de cette étude sur la place des analyses de Sûreté de Fonctionnement au cœur du processus d'Ingénierie Système, nous proposerons la Méthode D'Intégration des analyses de Sûreté de Fonctionnement au processus d'Ingénierie Système (MéDISIS), spécialement développée pour les Méthodes Basées sur les Modèles utilisant le langage de modélisation SysML. Dans cette partie, nous introduirons la méthode MéDISIS dans sa globalité avant d'en détailler les différentes étapes dans les autres chapitres de ce manuscrit. Cette méthode a pour but principal d'utiliser les informations dégagées lors

[2] L'ISBM est aussi connu sous le nom d'IS guidée par les modèles comme dans [Turki 2008].

des premières phases des processus d'Ingénierie Système pour préparer, accélérer et rendre cohérentes les analyses Sûreté de Fonctionnement.

I.2 Le processus d'Ingénierie Système

Comme nous l'avons indiqué en introduction, la réalisation de systèmes innovants nécessite l'emploi d'un processus à même de piloter le projet, de l'expression des besoins au déploiement du système. Ce type de processus est désigné par le terme de processus d'Ingénierie Système (IS). Il indique une suite de tâches à réaliser, nécessitant l'emploi de méthodes souvent supportées par des outils. Nous allons proposer les définitions des termes employés en IS, avant de présenter les principes de ces processus, leur standardisation, les méthodes utilisées pour leur déploiement et l'utilisation du langage SysML pour l'IS Basée sur les Modèles (ISBM).

I.2.1 L'IS et les termes employés

L'**Ingénierie Système** est une approche interdisciplinaire et concerne la capacité de réussir la réalisation des systèmes [IEEE P1220]. Afin d'assurer cet objectif, diverses techniques d'organisation et de réalisation des tâches sont employées.

Le but de l'IS est donc d'organiser et formaliser le processus de création afin de permettre la conception de nouveaux systèmes caractérisés par un fort besoin d'innovation [Friedenthal et al. 2008]. Le fait de rendre les systèmes développés innovants, et par-là même compétitifs, dépend de nombreux critères tels que :

- Intégrer de multiples technologies.
- Atteindre un haut niveau de complexité.
- Réduire les coûts de développement.
- Améliorer les fonctionnalités.
- Améliorer l'interopérabilité.
- Améliorer les performances.
- Améliorer la fiabilité.
- Réduire la taille.
- Intégrer et créer des Systèmes de Systèmes.

L'IS a trait au processus de développement des systèmes dans sa globalité, en intégrant toutes ses parties prenantes, dans toute sa durée. Les activités d'IS regroupent donc l'ensemble des activités exécutées de la naissance du besoin, à la livraison du produit final, voire jusqu'à sa fin de vie. D'un point de vue technique, la méthode utilisée pour l'IS doit permettre de traiter :

- Des exigences évoluant au cours du projet.
- Des niveaux d'abstraction différents.
- Des interconnexions multiples entre technologies multiples.
- Des contraintes liées à l'environnement du système et sa mission.

D'un point de vue général, l'IS inclut les activités techniques et de management de projet dont les objectifs sont :

- Satisfaction des parties prenantes.
- Recherche de qualité technique.
- Tenue des délais.
- Tenue des coûts.

Nous donnons dans cette partie les définitions des notions manipulées en IS, dans le but de clarifier notre exposé. Pour chacune de ces définitions, il existe de multiples variantes toutes aussi pertinentes dans la littérature. Nous avons choisi de conserver les définitions issues de consensus de communautés telles que l'AFIS[3] ou des définitions largement réutilisées dans la littérature.

Système : Selon l'AFIS un système est décrit comme un ensemble d'éléments en interaction entre eux et avec l'environnement, intégré pour rendre à ce dernier les services correspondant à sa finalité. Les normes IEC 61508, 61511 et l'INCOSE[4] dans son System Engineering Handbook [INCOSE 2004] insistent sur la dimension de composition en indiquant qu'un élément constitutif peut être un autre système appelé alors sous-système. Toutes ces définitions considèrent également que les éléments peuvent être de toute nature : matériel, logiciel, humain, documentaire …

Pour les trois définitions suivantes nous considérons celle données par [Martin 1996] et reprises par [Estefan 2008] afin de clarifier et unifier le vocabulaire utilisé en IS. Elles permettent d'éviter les nombreuses confusions existant entre les termes de *méthodologie*, *processus*, *méthode* et *outil*.

Processus : Un processus est une suite logique de tâches réalisées afin d'atteindre un objectif particulier. Un processus définit le « Quoi » en terme d'activités à accomplir sans préciser le « Comment ».

Méthode : Une méthode est un ensemble de techniques utilisées pour réaliser une tâche, en d'autres termes, elle définit le « Comment » de chaque tâche. A chaque niveau, les tâches d'un processus sont réalisées par l'emploi de méthodes. Cependant, chaque méthode peut être perçue comme un processus, avec une séquence de tâche à accomplir. En d'autres termes, le « Comment » d'un niveau d'abstraction donné devient le « Quoi » du niveau juste inférieur.

[Friedenthal et al. 2008] complètent ces définitions en insistant sur le fait que les méthodes d'IS se différencient du processus, dans le sens où elles précisent les types d'artefacts à produire pour la réalisation d'une tâche appelée par le processus d'IS.

Outil : Un outil est un instrument qui, lorsqu'il est appliqué suivant une méthode donnée, permet d'améliorer l'efficacité de la tâche accomplie ; à condition qu'il soit utilisé de façon adéquate par une personne qualifiée. Le but d'un outil devrait être de faciliter l'accomplissement des « Comment ». Plus généralement, un outil peut améliorer le « Quoi » et le « Comment ».

[3] AFIS : Association Française d'Ingénierie Système – www.afis.fr
[4] INCOSE : International Council on Systems Engineering – www.incose.org

Comme le propose [Estefan 2008], en considérant ces trois définitions, il est possible de donner une définition du terme *méthodologie*. Une **méthodologie** est un ensemble de processus, méthodes et outils relatifs les uns aux autres. Une méthodologie est l'agencement entre ces trois entités permettant de traiter une classe de problèmes donnée.

Ces différentes notions nous permettent de décrire les techniques utilisées pour organiser et mener à bien l'IS. Nous présentons, par la suite les caractéristiques de processus d'IS, les alternatives de processus standardisés, puis les méthodes utilisables pour les déployer. Nous présenterons plus en détail l'ISBM, incontournable pour la mise en place d'une IS de systèmes complexes à fortes contraintes de Sûreté de Fonctionnement (SdF). Nous introduirons enfin le langage SysML comme support aux méthodes et processus d'IS.

I.2.2 Le processus d'IS

On peut déduire des définitions données au paragraphe I.2.1 que l'objet d'un processus d'IS est de spécifier et ordonnancer les activités à réaliser pour développer un nouveau système. Le processus global est la réunion de deux processus : le processus de management et le processus technique. Le rôle du processus de management est de suivre la tenue des délais et coûts du projet ainsi que d'organiser la coordination entre les équipes et les parties prenantes. Le processus technique est appliqué pour spécifier, concevoir et valider le système développé. Dans ce manuscrit, nous nous intéressons à l'aspect technique du développement, avec pour but de fournir des méthodes permettant de traiter la complexité des études et d'assurer la qualité et l'efficacité des validations principalement en terme de SdF. L'aspect organisationnel ne sera donc pas détaillé et développé dans ce travail de thèse.

Les deux principales caractéristiques des processus d'IS sont leur application à l'ensemble de la conduite du projet et leur approche itérative. Ces deux aspects sont relevés très clairement par [INCOSE 2004] et [Friedenthal et al. 2008]. Les processus organisent l'avancée à travers les niveaux de détail et composition tout en ayant pour mission d'assurer la pérennisation des exigences, décisions et connaissances. Les tâches de base constitutives d'un processus d'IS sont définies dans [INCOSE 2004] par la liste suivante :

1. Définir les objectifs du système.
2. Etablir ses fonctionnalités.
3. Etablir les performances requises (exigences).
4. Développer l'architecture physique et logique.
5. Sélectionner une solution.
6. Vérifier la couverture des exigences par la solution.
7. Valider que la solution satisfait l'utilisateur final.
8. Passer au niveau d'analyse suivant (descente dans les niveaux de détail).

Le processus technique couvre ces 8 étapes, le processus de management organise les passages entre ces étapes et la poursuite des objectifs au niveau projet. Le processus technique d'IS est représenté de façon simplifiée à la Figure I.1. Cette vue permet de mettre en évidence les tenants et aboutissants du processus d'IS. Il s'agit de considérer les besoins des parties prenantes et de fournir en sortie un système satisfaisant au mieux ces besoins. Nous pouvons noter que l'on considère les besoins des parties prenantes au sens large, à savoir aussi bien les besoins de l'utilisateur final (une fonctionnalité, un service…) que ceux du concepteur (renta-

bilité, image de marque…). La première activité de spécification et conception est cruciale car elle conditionne l'ensemble de la réalisation. Elle détermine notamment les étapes de tests et validations car elle fournit les exigences suivies. Elle détermine aussi le lancement et les objectifs des étapes de conception des composants du système global. Cette description du processus permet également d'identifier la nature itérative du processus à travers la décomposition physique (Conception de composants), et également au travers des retours d'analyse et travaux menés (retour sur la conception, retour sur les tests et intégration).

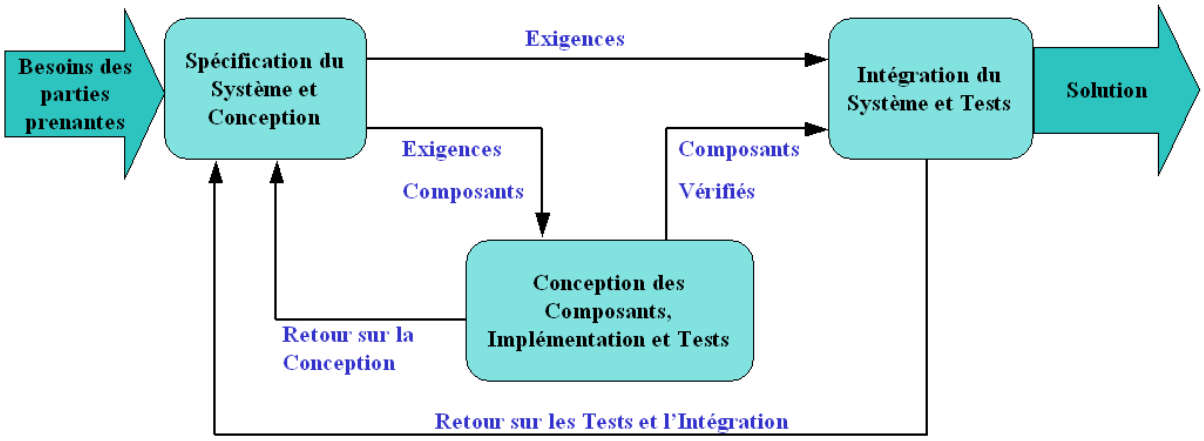


Figure I.1 *Processus technique d'IS (traduit de [Friedenthal et al. 2008])*

La première étape de Spécification du Système et Conception est donc déroulée pour définir une solution en rapport aux exigences des parties prenantes. De manière générale elle peut être vue comme un sous processus constitué des tâches suivantes :

- Elicitation et analyse des besoins des parties prenantes.
- Spécification du système par ses caractéristiques attendues en matière de qualité, de fonctionnalités, d'interfaces, de performances ou d'aspects physiques.
- Conception de solutions (plusieurs alternatives). Cette tâche nécessite le découpage en composants, constituant autant de sous-problèmes.
- Evaluation et comparaison des alternatives.
- Maintien de la traçabilité des exigences et objectifs à travers les décompositions du système.

Ce processus est à mettre en correspondance directe avec le traditionnel cycle en V, notamment la descente dans les niveaux de détail et les relations de composition. Ce type de cycle de développement est le plus utilisé dans le domaine de l'IS [Estefan 2008]. Il est utilisé comme processus dans de nombreux travaux de définition de méthodes de conception de systèmes sûrs : les systèmes embarqués pour [Sadou 2007] ou les systèmes mécatroniques chez [Schoenig 2004]. Sa représentation est donnée à la figure I.2. Ce cycle met l'accent sur la décomposition du problème posé, ainsi que sur le suivi du besoin, d'abord traduit en exigences, puis réutilisé sous cette forme pour valider les réalisations proposées.

Les processus d'IS sont mis en place afin de produire une dynamique de conception rigoureuse donnant le maximum de chances au projet d'atteindre ses objectifs. Pour atteindre la rigueur nécessaire à ce souhait, de nombreuses équipes d'ingénieurs système se sont dès les années 60 penchées sur les possibilités de standardisation de ces processus.

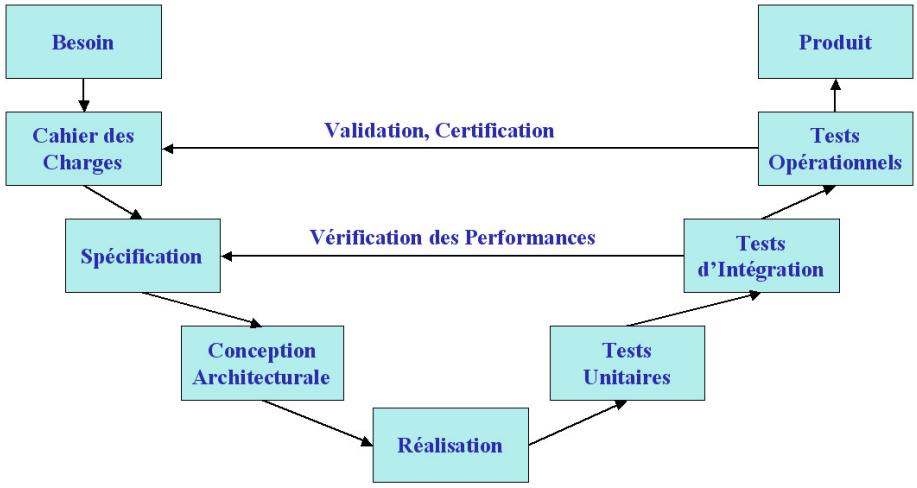


Figure I.2 *Le Cycle en V*

I.2.3 Les efforts de standardisation du processus d'IS

Historiquement, les définitions de processus d'IS ont été mises en place pour satisfaire les besoins de projets aérospatiaux ou militaires. Il s'agissait à l'époque de développer des systèmes employant des solutions techniques novatrices, auxquelles étaient confiées des missions critiques engageant des vies humaines. Ces projets ont pour la première fois nécessité une approche système, couvrant la conception d'engins, de plates-formes au sol, de systèmes de communication etc. Cette tendance à l'interdisciplinarité n'est que plus présente de nos jours, elle renforce le besoin d'institutionnalisation des pratiques d'IS permettant d'embrasser de multiples secteurs industriels.

La standardisation de l'IS est menée par diverses organisations d'ingénieurs telles que l'ANSI[5], l'EIA[6] ou l'IEEE[7]. Elle doit conduire à mieux encadrer les pratiques et permettre de tirer de nombreux bénéfices :

- Assurer la répétitivité des études.
- Contractualiser les études.
- Garantir la réalisation d'étapes prédéterminées.
- Organiser le déploiement du projet.
- Maîtriser la complexité.
- Institutionnaliser et améliorer l'IS.

Les trois principaux standards actuellement utilisés sont l'EIA 632, IEEE 1220 et l'IEC 15288. A eux trois, ils représentent les principes fondamentaux de l'IS et donnent les bases d'une approche d'IS. [Roedler 2002] différencie ces trois standards par leur propos :

- L'IEC 15288 établit un référentiel commun pour décrire le cycle de vie des systèmes.

[5] ANSI : American National Standards Association.
[6] EIA : Electronic Industries Alliance.
[7] IEEE : International Electronic and Electrical Engineers.

- L'AIE 632 propose un ensemble intégré de processus aidant le développeur dans la conception d'un système.
- L'IEEE 1220 fournit un standard focalisé sur le développement du système.

Ces trois standards s'inscrivent donc différemment dans le cycle de vie du système. La NASA admet la pertinence de ces standards et présente l'IEEE 1220 comme une application de l'AIE 632 spécialisée sur la phase de développement, alors que l'IEC 15288 fournit une approche globale pour l'ensemble de la vie du système et pour tout domaine d'application [NASA 2007]. Ces positionnements sont illustrés par la figure I.3. Le lecteur intéressé pourra trouver une description plus détaillée de l'EIA 632 dans [Sadou 2007] et de l'IEC 15288 dans [Turki 2008].

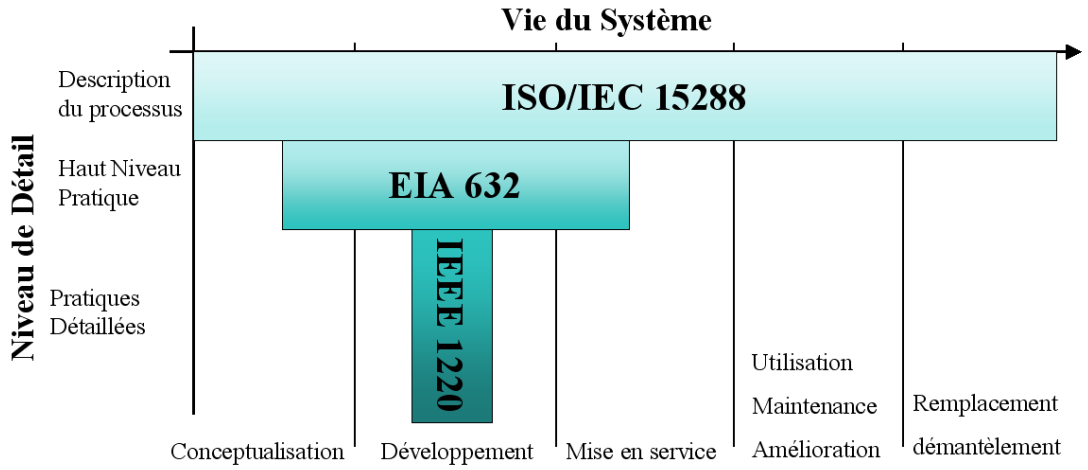


Figure I.3 *Positionnement des principaux Standards d'IS (traduit de[Roedler 2002])*

Dans cette thèse, nous nous intéressons plus particulièrement aux systèmes à fortes contraintes de sûreté de fonctionnement. Des travaux comme [Sadou 2007] ont largement utilisé les standards mentionnés ci-dessus pour l'ingénierie de ce type de systèmes. Cependant la communauté développant les systèmes à mission de sécurité a mis en place ses propres standards d'IS, aujourd'hui largement diffusés. L'IEC 61508 relative aux systèmes électriques, électroniques et électroniques programmables (E/E/EP) exerçant une fonction de sécurité est la norme la plus diffusée pour traiter ces applications. Cette norme de domaine d'application général met l'accent sur les activités d'analyse de risques et de validation de la sécurité du système étudié. Comme la norme IEC 15288, elle concerne l'ensemble des phases de vie du système jusqu'à sa destruction. La figure I.4 présente le processus d'IS préconisé par l'IEC 61508. La gestion de la sécurité passe par l'analyse des objectifs de sûreté (étape 3), la définition et l'allocation de ces objectifs (étapes 4 et 5), la planification de leur validation (étape 7) et leur validation globale (étape 13). L'étape 9 de réalisation comporte également des phases d'analyse et de validation des performances en terme de sûreté. Elle suit une philosophie de développement classique des processus techniques comme présenté à la figure I.1. Comme le mentionnent [Larish et al. 2008] et [Mkhida 2008], cette norme très générale possède de multiples normes « filles » relatives à des domaines sectoriels variés. L'industrie manufacturière pour l'IEC 62061, le transport ferroviaire pour l'EN 50126, le nucléaire pour l'IEC 61513 ou l'industrie de processus dans le cas de l'IEC 61511.

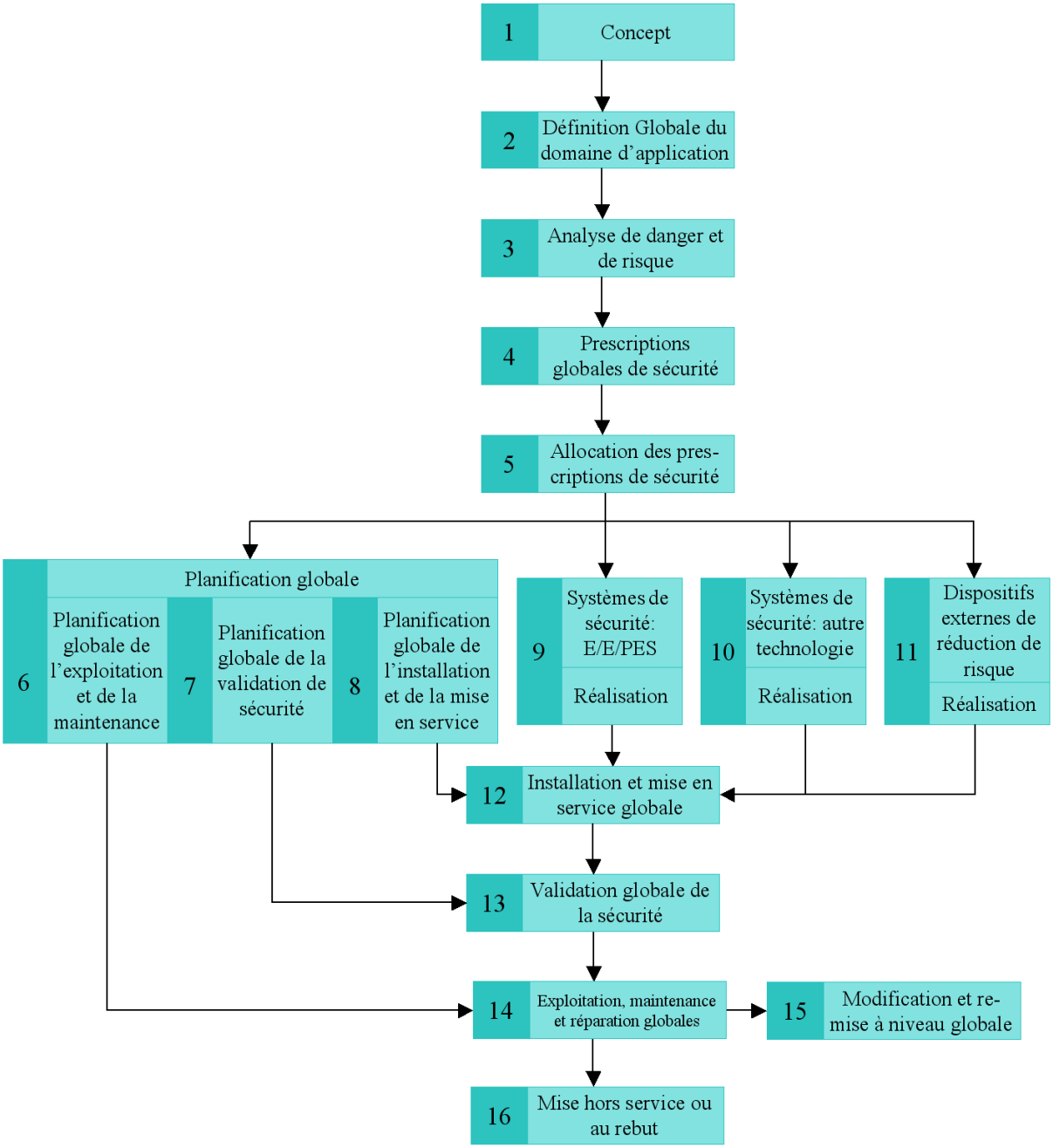


Figure I.4 *Cycle de vie de sécurité globale IEC 61508*

Ces processus définissent de façon efficace la suite de tâches à mener en précisant les objectifs de chaque activité. Ils donnent la marche à suivre pour le projet et coordonnent les travaux afin de constituer un cheminement cohérent et efficace. Néanmoins, ils ne comportent pas les informations nécessaires à leur utilisation concrète. En effet, dans un souci d'universalité et de large applicabilité, ces standards ne font évidemment pas de pré-requis sur la méthode d'IS employée pour leur implémentation.

Le rôle de la méthode d'IS est de retranscrire les directives du processus en actes d'ingénierie précis. La méthode décrit de façon factuelle les informations, documents, raisonnements ou tout autre élément tangible afin d'analyser et réaliser le système selon un processus donné. Des méthodes d'IS de natures différentes ont été développées, il y a encore peu de

temps, la plupart d'entre elles étaient basées sur la création documentaire (format papier ou informatique). Ces méthodes ont permis le développement de l'IS à Base Documentaire (ISBD), caractérisée par la création, l'accumulation et le classement de volumes de documents spécifiques aux formats hétérogènes. Cette base de connaissance est réunie pour motiver les prises de décision tout au long du processus de développement. D'autres initiatives comme les méthodes d'ingénierie basées sur les exigences proposent d'autres voies. Toutefois, la majorité des travaux actuels concernant les méthodes d'IS sont relatifs aux Méthodes Basées sur les Modèles (MBM). Ces méthodes dont la caractéristique fondamentale est l'utilisation centrale d'un modèle du système sont la base de l'ISBM (IS Basée sur les Modèles).

I.2.4 Principes et caractéristiques de l'ISBM

L'ISBD a longtemps été l'approche usuelle pour l'ingénierie des systèmes physiques. Puis certains domaines l'ont abandonnée au profit d'approche d'ISBM. Les ingénieurs en mécanique et systèmes électriques ont notamment abandonné le dessin technique pour l'utilisation d'outils de modélisation assistée par ordinateur au cours des années 80. Puis le monde du logiciel a adopté et largement amélioré l'ISBM pendant les années 90. L'ISBM se répand maintenant au sein des communautés d'ingénieurs système et devrait devenir le standard de développement des prochaines années, comme le souligne les prévisions de l'INCOSE [INCOSE 2007], [Friedenthal et al. 2007].

L'ISBM est l'utilisation formalisée de modèles comme supports aux activités de spécification, conception, analyse, vérification et validation des systèmes, débutant de la phase de conception préliminaire et se poursuivant à travers le développement jusqu'aux dernières phases de vie du système [INCOSE 2007]. Les techniques d'ISBM mettent l'accent sur l'évolution et la progression dans le niveau de détail lors de l'étude du système. L'ISBM est censée faire progresser l'IS dans de nombreux domaines comme la communication, la précision de l'analyse, l'intégration des résultats ou la réutilisation des connaissances produites. Ces critères sont les objectifs fondamentaux de l'IS comme nous l'avons vu au paragraphe I.2.1. Les bénéfices attendus dans l'utilisation de l'ISBM ont été réunis par [Friedenthal et al. 2008] :

- Réduire les risques liés au développement.
  - Validation et suivi continu des exigences.
  - Maîtrise des coûts.
- Amélioration de la communication entre les acteurs du projet.
  - Partage des connaissances et des informations perçues entre les équipes du projet.
  - Adoption de vues sur le système.
- Amélioration de la qualité de l'étude.
  - Exigences plus complètes, expressives et vérifiables.
  - Meilleure traçabilité entre exigences, conception, analyses et tests.
  - Intégrité du design.
- Amélioration de la productivité.
  - Evaluation de l'impact des modifications facilitée.
  - Réutilisation de modèles existant pour l'évolution du design.
  - Réduction des erreurs et tests d'intégration en temps réel.

    - Génération automatique de documents.
- Meilleur transfert des connaissances.
    - Capture des spécifications et données de conception dans un format normalisé.

Pour être mise en place, l'ISBM doit être déployée par une méthodologie. [Estefan 2008] indique qu'une méthodologie d'ISBM peut être décrite comme une collection de processus, méthodes et supports utilisés pour assister la réalisation de l'IS dans un contexte « basé modèle » ou « dirigé par les modèles ». Cette définition fait apparaître la notion centrale de modèle, en prise directe avec la ou les méthodes membres de la méthodologie. L'ISBM possède deux types d'entités fondamentales : le modèle du système et les bases de connaissances contenant le modèle et ses dérivés. La gestion du modèle et des connaissances qu'il génère impose l'utilisation d'un Environnement de Développement Système (EDS) réunissant la suite d'outil employée au cours de l'ISBM.

[Friedenthal et al. 2008] propose la définition suivante d'un modèle : Un **modèle** est une représentation d'un ou plusieurs concepts pouvant être réalisés dans le monde physique. Il décrit généralement un domaine d'intérêt. Une caractéristique clé des modèles est qu'ils sont une abstraction qui ne contient pas tous les détails de l'entité modélisée dans le domaine d'intérêt. Les modèles sont représentés sous de nombreuses formes : graphiques, mathématiques, représentations logiques, prototypes physiques... Cette définition introduit la notion extrêmement importante de vue. Une **vue** est une présentation partielle des connaissances sur l'élément modélisé. Les vues permettent de partitionner l'étude d'un système afin de proposer à l'analyste le sous-ensemble d'information nécessaire et suffisant à sa prise de décision. L'usage des vues permet donc de découper le problème global d'analyse ou de conception du système, en sous-problèmes plus facilement résolubles.

En ISBM, le modèle est constitué d'éléments souvent graphiques, qui représentent les exigences, l'architecture, le comportement, les cas de tests, les justifications et toutes les relations entre ces éléments. Le modèle est d'abord employé pour concevoir le système en rapport à ses spécifications. Il met en avant la décomposition du système et la distribution d'exigences qui en découle. Le modèle sert alors de support de transition de l'échelle système au niveau composant, préparant ainsi les développements particuliers induits. L'exploitation du modèle peut par exemple se traduire par la génération de spécifications de réalisation des sous-composants. Il sert également de base à la remontée des performances des sous-blocs et l'évaluation de leur participation aux performances globales. Enfin, ce modèle sert de base aux analyses et simulations du système, ainsi qu'à la génération de documents contractuels ou normatifs. Si le modèle est exécutable, ces fonctions peuvent être conduites par l'outil de modélisation, sinon des passerelles doivent exister entre les différents outils constituant l'EDS. Ces étapes induisent régulièrement des générations assistées de modèles. La place centrale occupée par le modèle au sein du développement des systèmes est illustrée par la figure I.5. On y observe que le modèle est le point de rencontre et la passerelle entre exigences, analyses, retour d'études, conception détaillée et production documentaire. Il sert aussi bien à préparer les études particulières qu'à répercuter leurs résultats et conclusions. Dans l'exemple de la figure I.5, le modèle du système est réalisé avec le langage SysML qui sera présenté au paragraphe suivant (I.2.5).

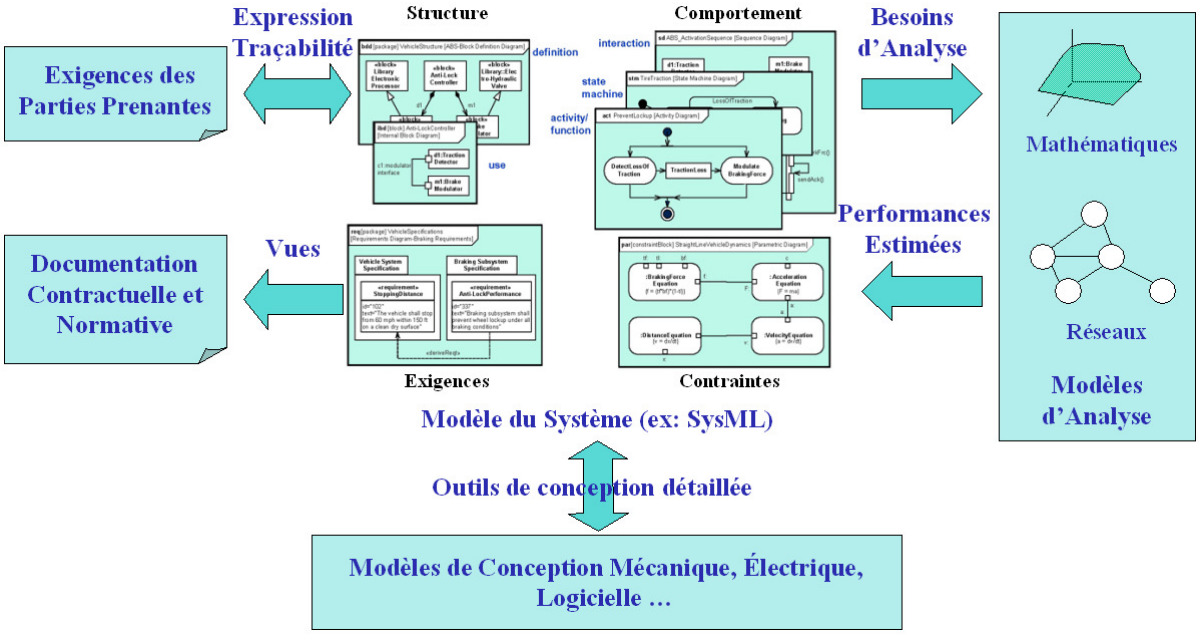


Figure I.5 *Le modèle du système au centre de l'IS (inspiré de[Friedenthal et al 2008])*

Les méthodologies d'ISBM doivent une grande part de leur efficacité à la puissance de l'EDS outillant la démarche. L'EDS est constitué de l'ensemble des outils utilisés pour réaliser les traitements mentionnés sur la figure I.5. [Friedenthal et al 2008] donne une définition complète des EDS : un EDS fait référence aux outils et bases de connaissances utilisés pour le développement d'un système. Les outils communément utilisés dans un EDS sont : les outils de modélisation, les outils d'analyse des composants physiques et logiciels, les outils de test et les outils de management de projet. Les outils et bases de connaissances sont considérés comme ayant un support logiciel, multi-utilisateurs et mis en réseau sur une infrastructure de stations de travail. Ce terme d'EDS implique la présence de liens logiques entre ces outils et bases de connaissances autorisant une ingénierie collaborative.

La cohérence du modèle exploité est donc la clé de voûte de la réussite de l'ISBM. Cette cohérence conditionne toutes les étapes d'analyse et de génération de connaissances à partir du modèle. Les deux éléments intervenant sur la qualité du modèle sont d'une part la méthode d'IS, qui va imprimer une rigueur à l'établissement du modèle, et le langage dans lequel il est exprimé. Le langage doit avoir un pouvoir d'expression suffisant, imposer une cohérence dans l'utilisation des artefacts de modélisation et être suffisamment structuré pour autoriser une analyse automatique, passant par une interprétation non ambiguë des éléments du modèle.

I.2.5 Le langage SysML

Le langage SysML est actuellement le langage le mieux adapté au paradigme de l'ISBM. Il est déjà largement diffusé comme langage de base pour des méthodes d'ISBM développées par nombre d'industriels ou associations d'ingénieurs. L'usage de ce langage se retrouve au sein de méthodes comme Harmony d'IBM Telelogic ([Douglass 2005], [Hoffmann 2006]), OOSEM (Object-Oriented, Systems Engineering Method, [Lykins et al. 2000], [Friedenthal et al. 2008]) ou RUP SE (Rational Unified Process for Systems Engineering [Cantor 2003]). Le langage SysML utilise l'approche Orientée-Objet et constitue un langage de modélisation graphique sans a priori technologique sur le système modélisé. Il est conçu à la fois pour aider

à l'analyse, la spécification, la conception, la vérification et la validation des systèmes complexes [OMG 2008]. Le langage offre la possibilité de modéliser des composants physiques ou logiciels, des acteurs humains, des flux de données, des procédures et tout ce qui peut composer un système ou un système de systèmes[8]. Il a été prévu pour fournir des descriptions préparant l'emploi de formalismes dédiés à un domaine d'étude ou un type d'analyse spécifique. Ce pouvoir d'expression est important dans l'utilisation de SysML, comme nous l'avons illustré par la figure I.5. SysML permet de modéliser les systèmes, leurs composants et la majorité des entités qui leur sont relatifs, il fournit des artefacts modélisant :

- La structure, les interconnexions et classifications.
- Les comportements basés sur des fonctions, des états et des messages.
- Les contraintes sur les propriétés physiques et les performances.
- Les exigences, leurs relations entre elles, leur affectation à la structure et la déclaration de leur méthode de test.
- Les allocations entre comportement, structure, exigences et contraintes.

Nous allons dans ce paragraphe présenter les origines de ce langage, puis nous aborderons les grandes lignes de sa syntaxe et de sa sémantique. Enfin, nous discuterons plus en détail son utilisation au cœur de l'ISBM.

I.2.5.1 Les origines de SysML

Le langage UML a été développé à la fin des années 90 pour répondre aux besoins des ingénieurs logiciel afin d'unifier et généraliser les pratiques de développement. Depuis son adoption en 1997 par l'OMG, le langage est devenu extrêmement populaire auprès de la communauté du logiciel et devient de nos jours le seul langage de modélisation graphique largement diffusé [Hause 2006] dans ce domaine. A partir de 2003 et la version 2.0, UML s'ouvre à l'IS en proposant de s'appliquer non plus aux « systèmes logiciels » mais aux systèmes en général. De nombreuses adaptations du langage pour des domaines spécifiques ont vu le jour grâce aux possibilités de stéréotypage et de création de « profils » supportés par la spécification UML 2.0.

Néanmoins, UML n'a pas rencontré le succès escompté auprès de la communauté des ingénieurs système, principalement à cause de ses notations trop orientées logiciel et d'un manque d'expressivité pour des problématiques spécifiques aux systèmes physiques [Hause 2006], [Willard 2007]. De plus, la création de profils réalisés par des groupes restreints de développeurs, pour des applications très ciblées, n'a pas permis une réelle pénétration de ces solutions dans l'IS en général. Les outils du marché ne pouvaient notamment pas suivre et implémenter chacune de ces expériences individuelles afin de permettre leur dissémination. C'est pourquoi l'OMG a lancé en 2003 une demande de proposition spécifiant un besoin d'extension formalisée d'UML pour l'IS [OMG 2003]. En réponse l'INCOSE et l'OMG ont développé le langage SysML adopté en 2007 [OMG 2007] et mis à jour dans une version 1.1 en novembre 2008 [OMG 2008]. SysML est construit comme un sous-ensemble des notations UML, augmenté de nouveaux artefacts destinés à traiter des spécificités de l'IS. SysML est donc constitué d'une partie commune avec UML, appelée UML4SysML, et d'une partie spécifique correspondant au profil SysML, comme illustré par la figure I.6. Cette nouvelle spéci-

[8] Un système de systèmes désigne un ensemble d'entités pouvant être considérées séparément comme un système à part entière, montrant une organisation cohérente et remplissant une mission commune.

fication a alors devant elle des défis majeurs à relever pour s'imposer comme le langage de l'ISBM, ces derniers sont résumés par [Willard 2007] :

- Trouver une large acceptation dans la communauté de l'IS (à l'opposé des profils UML trop spécialisés).
- Se démarquer de son prédécesseur UML 2.0.
- Offrir rapidement une standardisation stable.
- Disposer rapidement d'une large gamme d'outils supportant son utilisation.

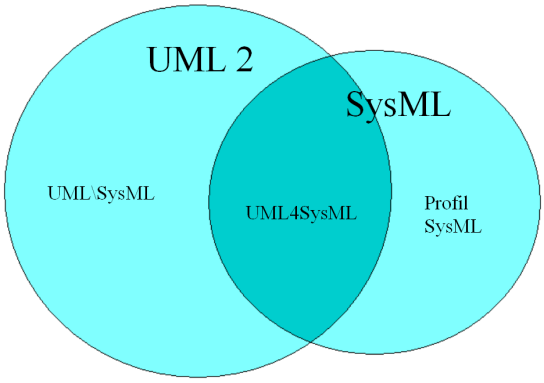


Figure I.6 *D'UML à SysML [OMG 2008]*

La spécification SysML a donc été mise en place pour répondre au mieux à ces challenges et aux besoins de l'ISBM, tels que nous les avons énoncés au paragraphe I.2.4. Pour cela les auteurs de la spécification ont cherché à donner au langage des caractéristiques de précision, concision, consistance, intelligibilité et justesse [OMG 2008].

I.2.5.2 Présentation synthétique de SysML

Outre la spécification de l'OMG [OMG 2008], le langage est présenté dans différents ouvrages [Friedenthal et al 2008]. Nous en faisons ici une présentation synthétique permettant d'aborder les notions utilisées dans cette thèse. Le but des créateurs de SysML était de proposer aux ingénieurs système un langage de modélisation simple et à la fois suffisamment expressif. C'est pourquoi le vocabulaire aux connotations fortement logicielles de l'UML a été abandonné. De même, le nombre de diagrammes utilisables a été réduit et de nouveaux artefacts, pour transcrire les exigences et les contraintes introduites. L'organisation des diagrammes utilisables est présentée à la figure I.7. On y retrouve les deux nouveaux types de diagrammes, ainsi que les extensions et réutilisations du langage UML. Les diagrammes sont regroupés par axe de modélisation, on distingue ainsi les diagrammes relatifs à l'expression de la structure et ceux relatifs à la description du comportement. On peut noter que la gestion des exigences a nécessité l'introduction d'un troisième axe, reflétant les besoins de traçabilité des demandes issues du cahier des charges durant le processus d'IS. Les principales caractéristiques d'un modèle utilisant SysML sont énoncées dans la spécification [OMG 2008]. Ce modèle est :

- Soumis à des exigences : SysML propose des artefacts pour organiser, gérer et spécifier des exigences.
- Composé : les systèmes sont modélisés en terme d'associations de composants. Le modèle est organisé en répertoires.

- Hiérarchisé : le modèle peut être organisé en répertoires dénotant le niveau de granularité des descriptions. Les artefacts peuvent posséder des relations de filiation.
- Interopérable : les modèles SysML sont destinés à être échangeables et partageables entre différents outils de l'EDS.

Nous allons présenter les possibilités de modélisation offertes par SysML selon trois axes : l'expression de la structure, l'expression du comportement et les besoins de modélisation spécifiques à l'IS. Mais avant cela, introduisons la notion centrale manipulée dans le langage : le *block*.

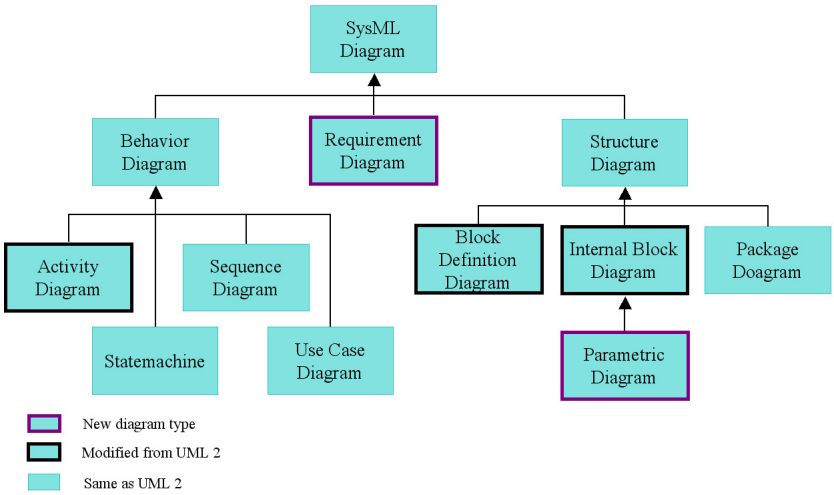


Figure I.7 *Organisation des diagrammes SysML [OMG 2008]*

Note de style : A partir de ce point nous utilisons les notations en *italique* pour nommer les éléments du langage SysML. Dans la plupart des cas, nous conservons les noms anglophones utilisés dans la spécification, afin de ne pas introduire de confusions dues à la traduction en français.

I.2.5.2.1 La notion de block

Le *block* est la brique de base dans la déclaration des concepts constitutifs d'un système. Les *blocks* sont utilisés pour définir un type de système, de composant, de flux ou d'entité abstraite représentant un concept. Un *block* modélise les caractéristiques (structurelles ou comportementales) que possède chaque instance de sa définition.

Les *propriétés* (*properties*) définissent les caractéristiques structurelles des *blocks*. Une *propriété* est typée par un *block* dont elle est une instance, dans le contexte défini par le *block* auquel elle appartient. Il existe différentes formes de *propriétés* : les *value properties*, les *part properties* (communément appelées *part*), les *port properties* (nommées *ports*), les *references properties* (nommées *references)* et les *constraint properties*. Une *propriété* hérite des caractéristiques du *block* qui lui confère son type, c'est-à-dire qu'elle possède les *propriétés* déclarées pour son type.

Les *value properties* décrivent les caractéristiques quantifiables des *blocks* (performances, aspects physiques). Les *value properties* sont typées par un *value type* exprimant l'ensemble de valeurs valides. Les *value types* possèdent une *unité* et une *dimension*.

Les *parts* sont issus d'un lien de composition entre le *block* accueillant le *part* et celui lui conférant son type. Ils représentent le rôle d'un *block* dans le contexte défini par le *block* possédant cette *propriété*. Le nom du rôle est le nom du *part*.

Les *port properties* sont des caractéristiques structurelles indiquant les points d'interaction offerts par le *block*. Ils sont utilisés pour connecter entre eux les *parts* (héritant des *ports* de leur type) d'un *block*. Il existe deux sortes de *ports* : les *flow ports* usuellement employés pour spécifier les flux manipulés et les *standard ports* présentant les interfaces de demande ou proposition de services.

Les *references* sont les instances de *blocks* extérieurs, en relation avec les instances que contient le *block* défini. Les *references* ne sont pas liées au *block* par une relation de composition. Les *references* peuvent par exemple être partagées par plusieurs *blocks*.

Les *constraint properties* expriment les contraintes vérifiées par les *value properties* du *block*. Une telle propriété est typée par un *constraint block*. Ces derniers sont une forme de *block* spécifique. Le concept de contrainte utilisé dans SysML est détaillé plus loin dans ce paragraphe.

Les caractéristiques comportementales des *blocks* sont identifiées par les *operations*. Elles définissent les traitements réalisables par les instances du *block*. Les *operations* sont employées dans l'axe comportemental du langage.

I.2.5.2.2 L'axe structurel

L'axe structurel est représenté sur deux principaux types de diagramme : les Block Definition Diagrams (BDD) et les Internal Block Diagrams (IBD). Les BDD sont utilisés pour déclarer les *blocks* et leurs *propriétés*, et mentionner les relations existant entre eux. Ces relations sont de quatre types :

- Les associations exprimant de façon générale une relation entre *blocks* (échange de flux, action commune …).
- Les généralisations exprimant la notion d'héritage entre deux concepts.
- Les agrégations symbolisant la déclaration d'une propriété *reference* dans le block source de la relation.
- Les compositions indiquant la déclaration d'un *part*.

Un BDD peut modéliser un *package*, un *block* ou un *constraint block*. Le *package* est utilisé pour montrer un regroupement logique de notions. Les BDD relatifs à un *block* ou un *constraint block* présentent un *block* et ses caractéristiques.

Les IBD montrent la structure interne des *blocks*. Les *parts* (instances des blocks) et leurs connexions y sont figurés. Les *ports* des différents *parts* sont indiqués et constitueront les points d'ancrage des *connecteurs*. Un *connecteur* dénote une égalité entre les entités situées à chacune de ses extrémités. Les IBD mettent en évidence le passage des flux d'un *part* à un autre. L'orientation des flux et *ports* permet de dénoter l'évolution du flux. Les changements de type des flux montrent également une partie des traitements effectués par les *parts*. Un exemple d'IBD est donné à la figure I.8. La structure interne d'un *block* définissant le concept

de capteur est proposée. Les *parts* de type différents sont connectés entre eux par le biais de leurs *flow ports*. Le type de ces *ports* indique le changement de nature du flux au fur et à mesure des traitements réalisés par le système. Le flux entrant sur la cellule sensible est un flux physique quelconque transformé en signal électrique, celui-ci est ensuite conditionné avant que l'unité de traitement l'analyse et que sa décision soit transmise sous un format donné par le module communication.

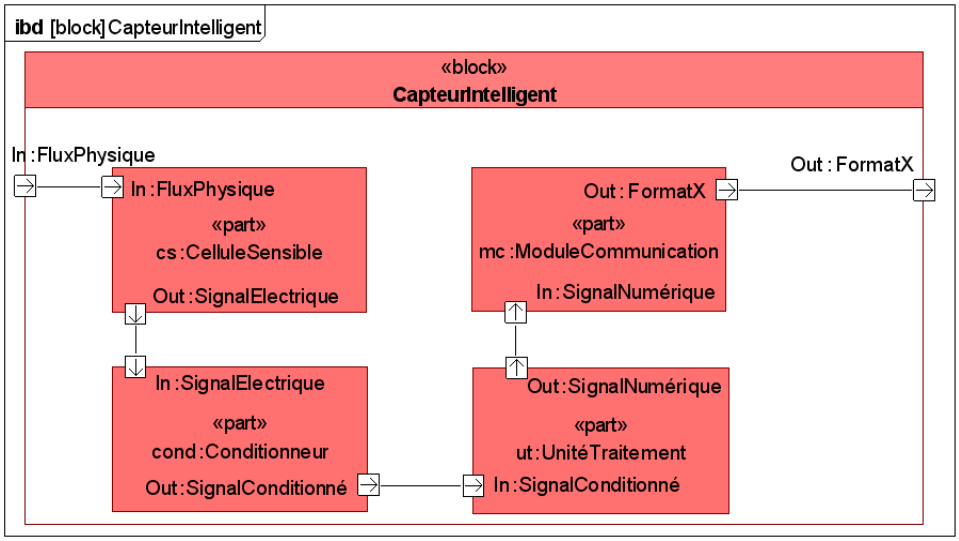


Figure I.8 *IBD du block CapteurIntelligent*

I.2.5.2.3 L'axe comportemental

Cet axe comporte 4 types de diagramme dont 3 sont dédiés à la modélisation du comportement des *blocks*. Le quatrième, constitué des Use Case Diagrams (UC), est une vue très haut niveau des fonctionnalités du système. L'environnement et les éléments en interaction avec le système, ainsi que les liens existants entre les diverses fonctionnalités, y sont représentés.

Le comportement des *blocks*, donc des éléments constitutifs du système, est modélisé par trois formalismes comportementaux et leurs diagrammes associés :

- Les *activités* modélisent la transformation des entrées en sorties des *blocks*. Une *activité* est constituée d'une suite d'*actions*. Une *action* est au choix, l'appel d'une *activité* de plus bas niveau ou l'exécution d'une *operation*. Les *activités* modélisent donc le comportement basé sur le traitement des flux au sein de Diagrammes d'Activité (AD). Les flux sont de la matière, de l'information ou de l'énergie et peuvent être modélisés par des *object nodes*. Cette notion avait été introduite par UML 2.0.

- Les *statemachines* sont utilisées pour modéliser comment les *blocks* se comportent face à l'arrivée d'événements. Le formalisme utilisé est basé sur les statecharts de David Harel [Harel 1987]. Le comportement du *block* est défini par un ensemble d'états interconnectés par des transitions, les événements provoquent des changements d'état déclenchant des actions. Les statemachines sont représentées sur les statemachines Diagrams (STM).

- Les *interactions* indiquent comment les *parts* d'un *block* interagissent par l'envoi de *messages*. Les *messages* modélisent des envois de signaux ou des invocations de services. Les *interactions* sont modélisées par des Diagrammes de Séquences (SD). Les SD sont privilégiés pour représenter des communications de type requête de services, adaptées à la définition de comportements discrets. Les *parts* sont représentés par leur ligne de vie (*lifeline*) permettant d'évoquer les instants de création et de destruction des objets. Ces *interactions* ont une dimension temporelle indiquant l'ordonnancement des tâches ou traitement.

I.2.5.2.4 Le support à l'IS

Le rôle de SysML au sein des spécifications produites par l'OMG est de franchir le pas séparant l'usage d'UML et les besoins de l'IS. De nouveaux diagrammes, fournissant de nouvelles possibilités de modélisation, ont donc été introduits en SysML. Ces mécanismes permettent de traiter la gestion d'exigences, l'expression de contraintes mathématiques et les allocations permettant de lier différents éléments du modèle.

En IS, les exigences découlent de l'analyse du besoin et des spécifications. Elles expriment une capacité que doit avoir le système ou une condition devant être vérifiée. Les exigences sont modélisées par les *requirements*. Elles sont principalement modélisées sur les Requirement Diagrams (RD) ou des tables d'exigences, mais peuvent aussi apparaître sur des diagrammes tels que les BDD. Les *requirements* possèdent un identifiant (id#) et peuvent être accompagnés d'une description textuelle. Il est possible d'utiliser de nombreuses relations pour organiser les exigences et ainsi soutenir leur gestion. Les relations employées sont ici listées :

- « deriveRqt » : indique une relation de dérivation entre deux *requirements*. La relation dénote que le *requirement* dérivé est déduit sémantiquement du premier.
- « copy » : indique la réutilisation d'un précédent *requirement*.
- « trace » : indique une relation faible entre un *requirement* et un autre élément du modèle. Ex : indiquer un document relatif au *requirement*.
- « refine » : indique qu'un élément fournit une explication sur l'origine ou le sens du *requirement*. L'élément apporte une précision levant une ambiguïté sur le *requirement*.
- « satisfy » : indique qu'un élément du modèle satisfait un *requirement*. La relation est utilisée pour allouer la réalisation des exigences aux éléments de la structure ou du comportement.
- « verify » : relation entre un *requirement* et un *test case*. Les *test case*s sont les techniques de tests mis en place pour vérifier un *requirement*.

Les Parametric Diagrams (Diagrammes Paramétriques, PAR) constituent un nouveau type de diagramme, permettant d'exprimer dans le modèle les relations mathématiques vérifiées par les caractéristiques d'un *block*. Ces derniers sont une extension des IBD, ils sont utilisés pour modéliser les *constraint properties* (possédées par les *blocks*) et leurs liens avec les *value properties* du *block* et de ses *parts*. Une *constraint property* est typée par un *constraint block*. Le *constraint block* possède des *constraint parameters* et une définition textuelle de la relation vérifiée par ses *parameters*. Cette relation peut être logique, algébrique ou de n'importe quelle autre forme mathématique. Lorsqu'un *block* possède une *constraint property*, il est possible de réaliser le Parametric Diagram modélisant les *value properties* liées par la

contrainte. Ces contraintes permettent la modélisation de propriétés allant de l'invariant algorithmique à l'expression de modes physiques de fonctionnement.

Les *value properties* sont liées par un *connecteur* aux *constraint parameters* auxquels elles confèrent leur valeur. Il est intéressant de noter que la relation ainsi proposée n'est pas orientée. Ce mécanisme permet de modéliser des analyses ou relations très différentes, comme le calcul d'un indicateur de performance, d'un indice pour mener une étude comparative de solutions ou la relation physique vérifiée par les caractéristiques d'un composant matériel. Une définition plus détaillée de ce type de diagramme et de leur utilisation est donnée dans [Peak et al. 2007a, b]. La figure I.9 donne un exemple simple de Parametric Diagram. Un *block* Rectangle possède une contrainte (calculSurface) reliant sa largeur, sa longueur et sa surface. Les *constraint parameters* de la contrainte calculSurface apparaissent comme des ports, les *value properties* du *block* Rectangle sont indiquées en rose.

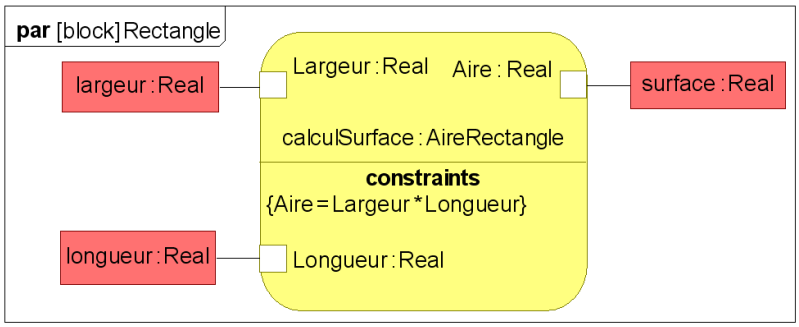


Figure I.9 *Exemple de Parametric Diagram*

La construction d'un modèle pour l'IS nécessite la spécification de liens entre les éléments du modèle, par exemple afin de dénoter des allocations entre structure et comportements. SysML propose un type de relation très générale nommée *allocation*, elle permet de lier deux éléments du modèle quelle que soit leur nature. Une telle relation possède deux extrémités, la première nommée « from » la seconde « to ». Un élément du modèle A est considéré « allocated to » un élément B, lorsque A se trouve à l'extrémité « from » et B à l'extrémité « to ». L'utilisation de l'*allocation* est laissée libre à l'utilisateur, elle permet par exemple de lier un comportement à un composant ou une contrainte à une opération. De façon globale, les *allocations* sont utiles pour augmenter la cohérence du modèle et préparer la déclaration de relations plus précises et fortes. La définition et l'usage d'une méthode d'utilisation du langage sont fondamentaux pour un emploi pertinent des *allocations*.

Ce tour d'horizon confirme que SysML atteint son principal objectif, être un langage qui permet de traiter les problèmes rencontrés en IS. Des éléments de modélisation sont disponibles pour décrire de façon efficace la structure et le comportement des systèmes. D'autres artefacts permettent de traiter les aspects inhérents au processus d'IS ainsi que ses contraintes telles que les exigences ou l'expression du calcul d'indice de performance. Le langage comporte également de multiples mécanismes de modélisation permettant d'obtenir une représentation cohérente des différents axes d'un système. SysML offre donc la possibilité de constituer un modèle destiné à être le cœur de l'ISBM. Ce modèle est destiné à être créé par une méthode d'IS spécifiant comment utiliser les différents artefacts proposés par le langage. De telles méthodes existent et ont été mentionnées en introduction de ce paragraphe I.2.5. La constitution rigoureuse du modèle central du processus d'IS, permet son exploitation lors des phases préconisées par le processus.

I.2.5.3 SysML au cœur de l'ISBM

Bien que SysML soit un langage récent, sa capacité à soutenir le processus d'IS a déjà été prouvée. Il a déjà été utilisé pour différents standards de processus d'IS, et diverses transitions vers des modèles d'analyse ou de conception détaillée ont été réalisées.

[Bonhomme 2008] a exploité SysML pour modéliser des systèmes de gestion de confort, sécurisation et maintien à domicile pour l'habitat. L'auteur réalise la recherche de solution logique et l'établissement du modèle physique en conformité avec les recommandations de l'EIA 632. Pour cela des AD, SD, BDD et IBD sont construits. La logique de commande est notamment modélisée par des AD. Ces diagrammes sont ensuite exploités pour construire des modèles Réseaux de Petri Temporel (RdPT), afin de valider la logique retenue. Cette thèse fournit un exemple de processus d'ISBM dont le modèle SysML est le pivot. La transformation vers les RDPT illustre un passage possible du modèle à des instruments d'analyse dédiés (ici à la validation temporelle de la logique de commande).

[Turki 2008] introduit sa propre méthodologie de conception de systèmes mécatroniques, nommée MISSYM, elle repose sur l'utilisation de SysML dans le cadre de l'IEC 15288. Le travail décrit l'utilisation de modèles SysML pour la réalisation des processus techniques de l'IEC 15288. La méthodologie intègre également une passerelle vers des modèles d'analyse spécifiques utilisant le langage Modelica [Modelica 2007]. Des approches similaires sont décrites dans [Pop et al. 2007] et [Johnson et al. 2008]. Le passage vers Modelica permet la simulation de systèmes à dynamique continue. La traduction automatisée nécessite une méthode stricte dans la saisie du modèle SysML ainsi que l'usage de stéréotypes définissant un nouveau profil UML/SysML. Dans ses travaux, Skander Turki utilise également une extension de SysML (uniquement les AD de UML) vers les Bond Graph [Turki et al. 2005] permettant d'introduire un modèle Bond Graph en tant qu'AD dans un modèle global SysML. Puis dans [Turki 2008] le profil migre vers l'utilisation exclusive d'IBD. Ici encore la méthode nécessite la définition d'un profil, la constitution de l'EDS se fait ainsi en grande partie par l'ajout de fonctionnalités à l'outil de saisie du modèle SysML.

Dans le domaine des systèmes de sécurité, on peut mentionner le travail de [Larish et al. 2008] qui étudient comment les modèles SysML peuvent être employés pour le déploiement de l'IEC 61508. Pour chaque type de diagramme, les auteurs indiquent durant quelle phase ils doivent être produits ou exploités. L'article ne décrit pas comment construire et analyser le diagramme, mais s'attache à montrer que l'information nécessaire peut y être modélisée.

Les Parametric Diagrams sont des diagrammes extrêmement efficaces pour réaliser des connexions vers les outils d'analyse des EDS. [Peak et al. 2007a] présentent les origines de ce type de diagramme, issu des Composable Objects (COB) [Peak 2000]. Les Parametric Diagrams représentent les systèmes de relations mathématiques applicables au système modélisé, mais pas leur résolution. Pour cela les systèmes d'équations doivent être transmis à un outil de calcul. [Peak et al. 2007a] réalisent une connexion au logiciel XaiTools™ [Peak 2000] permettant de résoudre les contraintes numériques. Le même type d'approche existe également pour des couplages avec Matlab/Simulink. [Vanderperren & Dehaene 2006] dressent une liste des outils commerciaux réalisant la connexion entre éditeurs UML/SysML et Simulink.

I.3 Intégrer les analyses SdF à l'ISBM et à l'EDS

Au cours du processus d'ISBM, le modèle commun d'ingénierie utilisé doit permettre de connecter et initier les différentes activités d'analyse et de conception détaillée nécessaires à la production du système désiré. Pour l'analyse de systèmes à fortes contraintes de SdF, le processus passe par d'importantes analyses du comportement du système, incluant l'observation des activités fonctionnelles et dysfonctionnelles du système. La validation formelle et la recherche de scénarios de défaillance sont les deux grands axes de travail relevés dans la littérature. La validation formelle du comportement est effectuée par la preuve de propriétés telles que l'atteignabilité d'états. Le calcul d'indicateurs de Fiabilité, Maintenabilité, Disponibilité et Sécurité (FMDS) utilise les techniques de recherche de scénarios, couplées à des simulations de Monte Carlo.

Le processus d'analyse de SdF peut être décomposé en deux phases d'un point de vue du rapport aux connaissances. Dans un premier temps, il nécessite la création et l'agrégation de nouvelles connaissances dépassant la vue fonctionnelle du système. Il s'agit du recensement du comportement dysfonctionnel de chaque élément du système. Cette partie requiert l'emploi de méthodes de création ou récupération d'informations, procédant par analyse des connaissances fonctionnelles disponibles et utilisation de jugements d'experts. Ce sont d'une part l'analyse fonctionnelle menée dans les premières phases du processus d'IS, et la réalisation d'Analyse des Modes de Défaillance de leurs Effets et Criticité (AMDEC) et d'Analyse Préliminaire des Risques (APR) d'autre part.

Dans un second temps, l'analyse de la SdF du système se poursuit par l'exploitation des connaissances précédemment rassemblées. Le but est alors de pouvoir qualifier et quantifier des aspects plus généraux du modèle, tel que l'atteignabilité d'un état redouté ou le calcul de la disponibilité globale du système. Durant cette phase, la connaissance nécessaire a déjà été produite, il reste à l'analyser pour produire les vérifications souhaitées. Ceci nécessite la création d'un modèle formel du système reflétant les facettes fonctionnelles et dysfonctionnelles de celui-ci. Les analyses produites à partir de ces modèles procèdent de deux manières distinctes : l'analyse par scénarios de fonctionnement du système (scénarios unitaires ou ensembles de scénarios), la preuve mathématique de propriétés du modèle. De nombreux outils, méthodes, langages et formalismes ont été développés pour réaliser cette partie de l'analyse. On peut distinguer deux approches principales par leur diffusion : l'utilisation d'automates à états finis et les Réseaux de Petri (RdP).

I.3.1 Les notions de l'analyse de SdF des systèmes

Le concept générique de la SdF a principalement été défini par J.-C. Laprie. On retrouve toute la terminologie dédiée à la SdF dans des ouvrages comme [Laprie1995] et [Villemeur 1988]. La SdF d'un système est d'abord définie comme « la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le service que [ce système] leur délivre ». Dans cette définition, on identifie la notion d'utilisateur qui peut être considéré comme un humain ou tout autre système utilisant les services du système étudié. La notion de service désigne le comportement perçu du système. La SdF se mesure par ses attributs permettant de juger de la qualité du système.

Les attributs représentent les différentes dimensions de la SdF. En effet, suivant le type de système ou le type de mission du système, les exigences à considérer pour le qualifier de sûr seront différentes. Par exemple, le fait qu'il ne soit pas dangereux pour son environnement physique n'a pas de sens pour un système purement logiciel, alors que pour un système médical cet aspect est incontournable. Les experts ont donc choisi de définir six attributs de la sûreté de fonctionnement qui sont autant de facettes de cette notion. On distinguera donc la **disponibilité**, la **fiabilité**, la **sécurité-innocuité**, la **confidentialité**, l'**intégrité** et la **maintenabilité** déjà bien définies dans la littérature ici citée.

La notion de **défaillance** est centrale à l'analyse de la SdF. On juge qu'il y a défaillance si le système ne remplit plus sa fonction. Quand elle caractérise le comportement d'un composant, on parle de **Mode de Défaillance** (MdD) de celui-ci. Prenons l'exemple d'un module devant délivrer un signal, on reconnaît en général pour ce dernier, quatre façons de ne pas remplir sa fonction : le module ne délivre pas de signal alors qu'il devrait, le module délivre un signal alors qu'il ne devrait pas, le module délivre un signal dégradé ou le module cesse de délivrer le signal, on parle alors de cessation de service. Ces quatre façons de dévier de la fonction sont appelées les MdD génériques du module. Lors de l'analyse de SdF d'un système, on va chercher à qualifier et quantifier les MdD de ces composants, afin de déterminer son comportement global. Les analyses permettront alors de juger les attributs de la SdF du système.

I.3.2 Du modèle fonctionnel à la connaissance du comportement dysfonctionnel

L'analyse de SdF d'un système passe par l'étude de ses MdD, elle conduit à pouvoir évaluer le comportement global du système. L'analyse de SdF est possible et initiée dès lors que l'on dispose d'une analyse fonctionnelle du système suffisamment avancée. Dans la norme IEC 61508, les activités d'analyse de SdF prennent place après la première phase de conceptualisation. A cet instant, le défi est de rassembler les connaissances sur les façons dont le système pourrait devenir défaillant. La seule donnée d'entrée disponible est la connaissance purement fonctionnelle du système avec des descriptions structurelles et comportementales. On cherche alors à rassembler les MdD des unités fonctionnelles ou des composants du système afin d'enrichir la connaissance du système mis en place.

A cet effet, les spécialistes du domaine ont développé des méthodes d'analyse permettant d'organiser la réflexion pour des systèmes complexes. La philosophie globale est de partir de l'étude des composants isolés, puis de déterminer la propagation des défaillances à l'échelle du système. L'analyse utilise un mode inductif de réflexion avant de passer à une approche déductive. Pour réaliser cela, l'AMDEC (Analyse des Modes de Défaillance de leurs Effets et Criticité) est la méthode standard la plus utilisée en conception [Tumer et al. 2003]. Nous allons maintenant présenter cette méthode en détaillant ses objectifs, ses points forts et ses faiblesses avant de présenter un état de l'art des multiples travaux traitant de sa réalisation.

I.3.2.1 L'AMDEC : présentation et objectifs

L'AMDEC a d'abord été déployée dans le secteur aéronautique au cours des année 60, avant de s'étendre à l'industrie manufacturière dans les années 80. Elle est maintenant utilisée dans la quasi-totalité des domaines. [Ridoux 1999] décrit l'AMDEC ainsi : « l'AMDEC est

une méthode d'analyse prévisionnelle de la fiabilité qui permet de recenser systématiquement les défaillances potentielles d'un dispositif puis d'estimer les risques liés à l'apparition de ces défaillances, afin d'engager les actions correctives à apporter au dispositif ». C'est une méthode inductive et qualitative qui propose d'explorer le système composant par composant. Pour chacun d'eux on recherche les modes de défaillance et leurs effets sur le système, en détaillant leur criticité et leur probabilité d'occurrence pour ainsi souligner les points sensibles du système. On tente ensuite de proposer des solutions pour combattre les problèmes soulevés. Le but d'une AMDEC est de permettre l'examen complet d'un système afin de mettre en évidence ses points critiques et permettre aux concepteurs de réaliser une solution atteignant ses objectifs en terme de SdF. L'AMDEC organise donc les activités suivantes [Pillay & Wang 2003] :

- Evaluer les effets des MdD des composants sur les performances.
- Identifier les composants critiques pour la sécurité.
- Développer des améliorations du système pour renforcer la sécurité et la fiabilité du système.

[Hecht et al. 2004] relève l'importance des AMDEC au cœur des activités de V&V (Vérification et Validation), où l'AMDEC a pour rôle de prouver que tous les MdD ont été identifiés et qu'une réponse satisfaisante leur est apportée. Elle a également pour rôle d'organiser l'effort de réduction des risques et l'activité de validation en soulignant les phénomènes dangereux dont l'étude est prioritaire.

La réalisation d'une AMDEC passe par les étapes suivantes :

- Rechercher la décomposition du système et identifier les fonctions.
- Répertorier pour chaque composant/fonction les modes de défaillances (MdD) envisageables.
- Déterminer la cause des défaillances.
- Rechercher les effets des défaillances à l'échelle du composant, du système et de l'environnement.
- Réaliser la cotation des risques: gravité (**g**), fréquence (**f**), (parfois détectabilité (**d**)).
- Calculer le RPN (Risk Priority Number). RPN = $\mathbf{g} \times \mathbf{f} \times \mathbf{d}$
- Elaborer des actions correctives et des moyens de détection pour réduire le risque.

Les cotations sont établies par les membres du groupe de travail de façon numérique. Des échelles de cotation doivent préalablement être fournies pour chaque notion. Le RPN peut être obtenu sous forme numérique ou suivre une table de résultat comme celle proposée dans [Guiochet 2003]. De très bons exemples d'échelles de cotations peuvent être trouvés dans [Bowles & Pelaez 1995]. De nombreuses alternatives peuvent être trouvées dans les documents normatifs décrivant les AMDEC : MIL-STD1629A, IEC 60 812. L'AMDEC aboutit à la rédaction d'un tableau dont chaque ligne décrit un risque encouru par le système. Ces lignes sont généralement ordonnées dans le document final par importance de leur RPN.

I.3.2.2 Les forces et faiblesses de l'AMDEC

L'AMDEC est née d'attentes très précises en matière d'analyse et de conception. Elle permet une recherche systématique et exhaustive des risques encourus par le système, facilitant le travail des experts [Picardi et al. 2004a]. Cette méthode est applicable durant tout le cycle de vie du système. Réalisée durant la phase de conception, elle permet de détecter très tôt les points sensibles sur lesquels focaliser toutes les attentions [Price 1996]. Réalisée en phase d'exploitation, elle devient une démarche d'amélioration du système [Teoh & Case 2004a]. L'AMDEC est très appréciée pour la constitution de base de données concernant l'ensemble des composants rencontrés au cours des études [Bassetto 2005]. Le recensement systématique des modes de défaillance et de leurs cotations permet de constituer une base importante pour utiliser le retour d'expérience dans les nouvelles créations. Le format de présentation est très facile à réorganiser en structure de données accélérant les travaux futurs. De ce fait, l'AMDEC est très répandue et largement acceptée dans le milieu industriel [Ridoux 1999], [Tumer et al. 2003], [Teoh & Case 2004a,b]. Enfin, la méthode est applicable à un ensemble très varié de systèmes, pouvant faire appel à diverses technologies (mécanique, électronique, logiciel …).

L'AMDEC est un outil de recensement et détection des risques à un niveau bas dans le processus de conception. Cependant l'AMDEC connaît un grand nombre de détracteur et souffre même parfois d'une certaine impopularité lorsqu'il s'agit de la rendre opérationnelle. Les nombreuses années de pratique de cette méthode ont permis aux chercheurs et utilisateurs d'identifier les entraves à sa mise en œuvre efficace. De nombreuses contraintes organisationnelles et opérationnelles éloignent bien souvent les ingénieurs des bénéfices qu'ils souhaitaient tirer de l'étude. De nombreux auteurs mentionnent ainsi les freins à la parfaite réussite des AMDEC. L'AMDEC est en effet perçue par la majorité de ses utilisateurs comme une analyse longue à mettre en œuvre, peu flexible, fastidieuse et souvent très propice aux erreurs [Ridoux 1999], [Hughes et al. 1999], [Price & Taylor 2002], [Papadopoulos et al. 2004], [Picardi et al. 2004a], [Teoh & Case 2004a,b], [Bassetto 2005]. Pour de très nombreuses personnes la réalisation d'AMDEC relève uniquement d'un impératif administratif et contractuel visant à satisfaire la hiérarchie ou le client [Ridoux 1999], [Papadopoulos et al. 2004], [Teoh & Case 2004a,b].

Les industriels expriment également leurs difficultés à lier les résultats de l'analyse au déclenchement d'actions correctives et de reprise de la conception [Bassetto 2005]. L'AMDEC perd ainsi son pouvoir d'aiguillage dans le processus de conception. La lourdeur de la méthode semble avoir plusieurs origines. Ce sont d'une part les difficultés à réunir régulièrement les personnes compétentes pour la réalisation et d'autre par le fort volume d'information à produire, présenter et comprendre. En effet, l'établissement d'une AMDEC doit se faire en présence des experts de chaque domaine ou partie du système ou processus étudié. L'étude donne pour résultat un long tableau sous forme papier, qu'il est bien souvent difficile d'appréhender, eu égard à l'explosion combinatoire du nombre de lignes crées dans une étude complète [Ridoux 1999], [Papadopoulos et al. 2004], [Bassetto 2005].

Ces différentes contraintes ralentissent la finalisation des AMDEC. Celles-ci se trouvent alors réalisées bien trop tard dans le processus de conception annihilant ainsi une des principales missions de l'AMDEC : identifier le plus tôt dans le cycle de développement les risques et défaillances probables du système [Papadopoulos et al. 2004], [Teoh & Case 2004a,b]. Sa rédaction en langage naturel est également présentée comme un frein par [Bowles & Pelaez

1995], [Pillay & Wang 2003], [Yeh & Hsieh 2007] qui pointent la subjectivité et l'imprécision de l'usage du langage naturel et des RPN. Un tel usage peut être source d'imprécision voire même d'erreur dans l'ordonnancement ou l'évaluation des risques. Enfin l'emploi du langage naturel entraîne des difficultés de communication et de passage des connaissances entre experts. Une dernière critique relevée sur l'AMDEC est l'impossibilité de mettre en évidence des effets de défaillances multiples [Ridoux 1999], [Price & Taylor 2002], [Picardi et al. 2004a].

I.3.2.3 Les travaux relatifs aux AMDEC

L'AMDEC souffre donc de nombreux désavantages, mais les services qu'elle rend justifient toujours sa large utilisation. C'est pourquoi, de multiples travaux ont été entrepris afin de faire évoluer et améliorer cette technique, pour la rendre plus efficace. Pour cela, les recherches disposent de deux leviers : l'aspect organisationnel et l'ajout d'un support logiciel. [Bassetto 2005] fort d'une expérience menée à grande échelle fournit quelques conseils d'organisation pour impliquer les ingénieurs responsables des AMDEC et améliorer leur perception sur l'efficacité de la méthode :

- Définir l'objet et les limites de l'analyse.
- Constituer une équipe réunissant spécialistes et utilisateurs du système analysé.
- Former les participants à la méthode.
- Assurer la régularité des rencontres.
- Obtenir l'adhésion du management et des équipes opérationnelles en faisant valoir leur intérêt dans la méthode.

Pour résoudre les problèmes liés à l'ambiguïté des cotations et du vocabulaire employé [Bowles & Pelaez 1995] proposent un emploi de la logique floue permettant de transcrire de façon plus explicite et en langage naturel les cotations des risques. La méthode employée cherche au moyen des règles d'inférences à être plus précise sur la qualification de risques à indicateurs RPN similaires mais à vecteurs de risques différents. Dans ce travail, la cotation des paramètres de fréquence, gravité et détectabilité sont notés de façon classique, l'emploi de la logique floue permet en sortie de caractériser le risque de façon textuelle. La méthode décrite a été utilisée sur de nombreux problèmes comme explicité dans [Yeh & Hsieh 2007] qui appliquent notamment la méthode à une station d'épuration. [Pillay & Wang 2003] complètent ce type d'approche en couplant le raisonnement à l'usage de la théorie des « Grey Systems » et permettent par leurs travaux une classification des risques plus précise et sûre.

L'intégration de l'AMDEC à un processus d'ISBM ne peut se faire sans un support logiciel adapté. Dans ce sens de nombreuses recherches se sont penchées sur l'aide que pouvait apporter l'outil informatique à la création d'AMDEC. Ainsi des solutions proposant d'automatiser certaines parties du raisonnement des AMDEC ont vu le jour. Ces solutions apportent plus de rapidité, cohérence et justesse à l'AMDEC. Les solutions, que nous allons maintenant présenter, ont principalement pour objectif de mener la recherche des effets des MdD sur le système.

[Teoh & Case 2004b] présentent de façon claire la problématique commune à tous ces travaux d'automatisation. Les auteurs posent les bases des raisonnements et modèles nécessaires à la réalisation des AMDEC. Ils posent le principe qu'une étude AMDEC nécessite l'utilisation d'un modèle apportant une vue structurelle et une vue fonctionnelle du système pour son automatisation. L'automatisation de l'AMDEC nécessite de recréer le raisonnement

humain en utilisant des techniques d'Intelligence Artificielle. L'AMDEC est conduite pour établir des relations causes/effets à travers les fonctions et composants du modèle. Trois types généraux de raisonnements peuvent être utilisés pour réaliser une AMDEC :

- *Rule based reasoning* (raisonnement basé sur les règles) les règles de réactions de chaque composant au MdD doivent être spécifiées explicitement pour chaque composant ou fonction, [Bouti et al. 1994].
- *Model-based reasoning* (raisonnement basé sur les modèles). Dans les faits, les modèles embarquent les règles de réaction pour fournir les réactions attendues en simulation. Ces méthodes nécessitent l'emploi de librairies de composants et comportements afin de pouvoir automatiser certaines phases de l'étude. Ces raisonnements sont parfois difficiles à utiliser lors des premières phases de développement car les modèles manquent souvent de précision et de richesse, [Atkinson et al. 1992], [Hughes et al. 1999], [Price 2000].
- *Case based reasonning* (raisonnement sur des cas d'études). L'approche n'a jamais été utilisée pour la génération d'AMDEC. Elle semble difficile à employer car elle nécessite une masse très importante d'information pour être initiée.

Dans les faits, les travaux les plus aboutis utilisent tous la seconde possibilité. [Teoh & Case 2004a] revendiquent l'utilisation d'un 4ème type : le *knowledge fragment reasoning* (raisonnement sur connaissances partielles). L'approche mixe étude de modèle fonctionnel faisant appel au *Model-based reasoning* et connaissances partielles sur des rapports d'experts précédents (constituant une base de connaissance des comportements dysfonctionnels). Les auteurs développent une méthode de génération d'AMDEC (FMAG : FMEA Generation) analysant des modèles de conception préliminaires utilisant des diagrammes IDEF 3[9] afin d'exprimer les enchaînements de fonctions et des diagrammes fonctionnels, présentant pour chaque opération les opérateurs (composants) et les opérandes (objets manipulés par l'opération). L'analyse de ce modèle est réalisée par un algorithme de raisonnement « knowledge fragment » organisant le réemploi des précédents avis d'experts (constituant la base de « knowledge fragment ») et analysant les liaisons logiques existantes dans le modèle fonctionnel proposé. L'analyse automatique ne peut avoir lieu que si une base de connaissances des comportements dysfonctionnels est déjà renseignée pour indiquer les différents états des opérateurs et opérandes atteignables dans le modèle. Des ensembles de pré et post conditions doivent être enregistrés pour les MdD connus de ces composants ou flux.

Dans ses travaux Christopher Price s'est intéressé à de nombreux aspects des AMDEC. Il aborde les AMDEC avec les outils du model-based reasoning. Il a tout d'abord mis en place la première solution de génération automatique d'effet pour l'AMDEC de systèmes électriques nommée FLAME System [Price et al. 1995]. Le système est alors capable de simuler la propagation des défaillances à travers un modèle dédié du circuit électronique considéré. Ce modèle comprend le comportement fonctionnel du système ainsi que la connaissance des effets locaux de chaque MdD des composants. Dans [Price 1996], l'auteur complète ce travail en renforçant l'outil par une analyse des modifications du modèle afin de pouvoir conduire des AMDEC incrémentales, c'est à dire réaliser une nouvelle AMDEC à chaque nouveau pas du design. Pour cela, l'outil recalcule les effets des MdD et met uniquement en évidence les risques ayant évolué. Dans [Price 2000], l'auteur présente l'outil AutoSteve d'aide à la concep-

[9] IDEF 3 : Integrated Definition for Function Modeling, langage développé par le National Institute of Standards and Technology.

tion de systèmes électroniques complexes à destination de l'automobile. L'outil possède un générateur d'AMDEC. La génération est réalisée par simulation qualitative d'un modèle de la carte électronique réunissant la partie fonctionnelle et la partie dysfonctionnelle des fonctions, sans nécessité l'architecture physique du système. Dans [Price & Taylor 2002], les auteurs proposent une amélioration de la solution permettant de traiter les défaillances multiples. Le but de la méthode est de contenir l'explosion combinatoire par des moyens « d'élagage » de l'arbre des scénarios de défaillance. Ainsi une méthode et des règles de sélection des risques significatifs sont utilisées. Ces solutions d'automatisation d'AMDEC ont été exclusivement réalisées pour des systèmes électriques. On trouve dans la littérature des approches similaires dédiées aux systèmes mécaniques pour [Hughes et al. 1999] ou aux systèmes hydrauliques pour [Atkinson et al. 1992]. Dans chacun de ces cas, le modèle analysé est dédié à la technologie employée, ces cas n'abordent pas l'exploitation de ces travaux pour des systèmes multi-technologiques.

[Papadopoulos et al. 2004] développent une approche cherchant à rester généraliste et applicable à tous types de technologie. Le modèle fonctionnel du système est saisi sous Matlab-Simulink puis enrichi par une série d'arbres de défaillances créés à partir de la saisie du caractère dysfonctionnel des composants. L'AMDEC est ensuite réalisée par l'étude des arbres de défaillances. A partir de la connaissance des défaillances particulières de chacun des composants, cette méthode permet de déduire l'effet de ces dernières à l'échelle du système, ainsi que de considérer les défaillances simultanées.

L'automatisation des AMDEC a fait l'objet d'une initiative européenne avec le projet AUTAS[10] [Picardi et al 2004a]. Le domaine d'application fixé pour cette étude était l'aéronautique. L'outil est créé afin de permettre le respect des standards, utilisés dans ce domaine, de façon efficace. La solution proposée est basée sur une utilisation de modèles des composants employés et une logique de raisonnement qualitatif. Le logiciel est prévu pour remplir les fonctions suivantes :

- Gestion et création d'une base de données de modèles de composants et sous systèmes. Ces modèles doivent être génériques, réutilisables et composables. Les bases intègrent au fur et à mesure des études les nouveaux modèles créés. Ces modèles portent la description des composants et des effets qu'ils subissent en cas de défaillance.
- Edition de missions et scénarios. Cette fonctionnalité permet de considérer le système durant ses diverses phases de vie.
- Réalisation des étapes déductives de l'AMDEC. La recherche des conséquences des défaillances aux divers niveaux hiérarchiques est automatisée en exploitant le modèle des composants.
- Mise en forme des résultats en respectant les formats normalisés utilisés en aéronautique.

AUTAS est formé de trois outils : Models and Mission Builder (MMB), Qualitative FMEA Engine (QFE), FMECA Support System (FSS). Les auteurs ont créé leur propre langage de définition des modèles intégrés à la base de donnée. Le format est basé sur des schémas XML. Le langage permet la déclaration des composants, des effets qu'ils subissent, des intervalles de variations des variables du modèle et de la mission considérée. Le module MMB fournit au QFE le modèle qualitatif réunissant la définition de la mission et du système. Le QFE analyse

[10] AUTAS : Automating FMECA for aircraft systems. EU Project – GRD – 2001 – 40133

les inférences entre composants et défaillances. Le MMB fournit au FSS les directives de création du document, le FSS édite ensuite le rapport AMDEC. La partie traitement du système est réalisée par le QFE qui simule le système étudié à travers le modèle fournit. Les différents MdD possibles pour les composants sont étudiés en propageant les changements de variables induits par les MdD parmi les composants. La logique de résolution de contraintes est décrite plus précisément dans [Picardi et al. 2004b].

Ces travaux aboutissent à des solutions performantes améliorant significativement la réalisation des AMDEC. Leur but est d'aider l'analyste dans la phase d'étude de second niveau des MdD. L'outil d'analyse intervient lorsque les défaillances individuelles des composants du système sont identifiées et modélisées. En effet, les solutions simulent qualitativement le système en employant un modèle fonctionnel du système et un modèle des défaillances particulières. L'AMDEC devient plus rapide et systématique permettant une utilisation plus performante au sein d'un processus d'IS. Le point critique de l'utilisation de l'AMDEC dans un tel processus est de devoir fournir les conclusions dans un délai acceptable pour les autres activités de conception.

Ces méthodes se rapprochent également de la philosophie de l'ISBM car l'ensemble de ces travaux utilisent des modèles comme base du raisonnement et comme moyen permettant l'automatisation. Cependant, chacune de ces solutions utilise un langage ou une méthode de constitution du modèle exclusive à la problématique de l'AMDEC et souvent dédiée à une unique technologie de système. Pour progresser encore sur la voie de l'intégration des AMDEC à l'ISBM, il nous semble nécessaire d'une part de réaliser l'étude sur des modèles intégrant toutes les technologies envisageables et d'autre part sur un modèle commun à l'ensemble de l'EDS.

Les travaux précédents ne proposent pas d'aide aux premières phases de l'AMDEC, qui est l'analyse d'un modèle purement fonctionnel, pour déduire le comportement dysfonctionnel. Cette phase est, à notre sens, la plus utile pour l'ISBM, car elle est une étape de création de l'information. L'AMDEC est réalisée à partir de l'analyse fonctionnelle du système, reflétée durant l'ISBM par le modèle commun de l'EDS. Nous souhaitons donc dans cette thèse montrer que l'AMDEC peut profiter du support fournit par un processus d'ISBM, que cette collaboration permet d'avancer dans la conception du système et qu'il est possible d'automatiser les AMDEC pour l'étude de systèmes multi-technologiques.

I.3.3 Evaluation du comportement global du système : Quantification et Validation des attributs de SdF

L'AMDEC permet d'isoler les comportements défaillants à prendre en compte dans la suite du développement du système. Afin de vérifier formellement l'impact de ces MdD et de quantifier les performances du système, il est nécessaire d'utiliser une représentation du système permettant ces analyses. Nous présentons donc un rapide tour d'horizon des modèles utilisés pour mener ces analyses de SdF. Nous mentionnerons l'usage de Réseaux de Petri (RdP) et des langages AltaRica Data Flow et FIGARO. Les méthodes qui vont être présentées ici ont été développées dans le but de pouvoir traiter les systèmes à fiabilité dynamique. Ces systèmes ont pour caractéristiques de pouvoir exercer des fonctions de reconfiguration, changeant le comportement du système au fil des événements de sa vie et correspondent au niveau de complexité des systèmes manipulés dans cette thèse. L'analyse de leur niveau de SdF n'est

pas abordable par les méthodes classiques utilisant la logique booléenne comme les Arbres de Défaillances (AdD) ou les Diagrammes Bloc de Fiabilité (DBF), dans la mesure où ils ne permettent pas de tenir compte de l'ordre d'apparition des événements puis d'un changement de mode de fonctionnement du système [Dutuit et al. 1997]. Ces modèles ont donc pour points communs de pouvoir modéliser des systèmes dont le comportement dépend de facteurs temporels et d'événements.

I.3.3.1 Les RdP Stochastiques Généralisés

L'utilisation des RdP pour les analyses de fiabilité est née de la nécessité de considérer les systèmes multi-états reconfigurables [Zio 2009]. Les RdP permettent de traiter de façon plus souple que les approches markoviennes, la modélisation de systèmes présentant des évolutions parallèles ou l'utilisation de délais déterministes. Les RdP [Murata 1989] sont des graphes orientés bipartites constitués de deux types de nœuds : les places et les transitions. Ils permettent de représenter le système sous une vue états/transitions. Leur extension vers les RdP Stochastiques Généralisés (RdPSG) permet d'affecter des lois de probabilité aux passages des transitions, le concept de transition simple étant conservé par affectation de délai de type Dirac 0. L'état du système est modélisé par le marquage des places formant le réseau. Pour la quantification du comportement du système, les RdPSG sont simulés. On emploie des simulations de Monte Carlo pour évaluer des caractéristiques probabilistes du réseau, comme le temps passé dans un état sûr ou le nombre de franchissement d'une transition exprimant une défaillance pendant une période donnée. L'utilisation de ce formalisme pour les analyses quantitatives de SdF est discutée dans de nombreuses publications auxquelles le lecteur intéressé pourra se référer, [Maholtra & Trivedi 1995], [Dutuit et al. 1997], [Sachdeva et al. 2007], [Signoret et al. 2007], [Kleyner & Volovoi 2008], [Clavareau & Labeau 2009]. Ces articles présentent de nombreux cas d'utilisation des RdP pour l'évaluation de divers types de systèmes industriels intégrant par exemple des politiques de remplacement et maintenance variées. La principale difficulté dans l'utilisation des RdP reste la construction du modèle. L'expert doit alors toujours équilibrer l'expressivité et la taille du modèle pour ne pas atteindre un seuil critique de complexité. L'utilisation directe de RdP nécessite une grande expertise, c'est pourquoi, plusieurs travaux de recherche se sont tourner vers le développement de langages permettant la saisie simple de modèles dysfonctionnels. Ces modèles peuvent ensuite être traduits ou mis en correspondance avec des formalismes mathématiques de plus bas niveau permettant la quantification des risques. Les langages ainsi formés doivent permettre une capitalisation plus simple des modèles de SdF créés, pour des réutilisations au cours d'études futures, comme nous allons le voir dans les deux paragraphes suivants.

I.3.3.2 Le langage AltaRica Data Flow

Le langage AltaRica Data Flow[11] est un des langages permettant une utilisation industrielle des modèles de comportement dysfonctionnel. Depuis sa création, il a bénéficié d'un intérêt croissant de la communauté scientifique pour l'élaboration de modèles dédiés à l'analyse qualitative et quantitative de SdF. Le langage cherche à proposer les concepts minimaux utilisés dans les représentations traditionnelles d'analyse de fiabilité comme les AdD ou les RdPSG. Ainsi, AltaRica Data Flow s'oriente vers une représentation compositionnelle des systèmes, dans lesquels les composants sont modélisés par des automates à états. Les changements d'états des composants montrent leurs différents états de fonctionnement et permettent donc

[11] Site web dédié à AltaRica et ses variantes : http://altarica.labri.u-bordeaux.fr/wiki.

de modéliser leurs défaillances. Ces transitions sont accompagnées d'assertions contraignant les variables du système. Les modèles ainsi édités sont analysables pour la génération d'AdD ou l'évaluation de critères quantifiés de SdF [Rauzy 2002], [Boiteau et al. 2006], [Rauzy et al. 2008]. Ce langage est déjà instrumenté par différentes solutions commerciales comme BPA-DAS[12] de Dassault Systèmes ou SIMFIA[13] d'APSYS. Nous avons choisi de présenter dans cette thèse la connexion de SysML à ce langage. Il fera donc l'objet d'une présentation plus détaillée dans le chapitre III.

I.3.3.3 Le langage FIGARO et les BDMP

Le second langage que nous pouvons mentionner, a été développé par EDF et porte le nom de FIGARO [Bouissou et al. 1991]. Le but de FIGARO est de généraliser les formalismes classiques de SdF tout en permettant de les manipuler avec leurs représentations graphiques habituelles. Le langage est tourné vers la constitution de bases de connaissances rassemblant les éléments des formalismes usuels de SdF. Ainsi, FIGARO est organisé en deux niveaux de langage, le FIGARO d'ordre 1 permet la déclaration des concepts d'objet qui seront ensuite manipulés et connectés en utilisant le FIGARO d'ordre 0. Le FIGARO sert ainsi à créer les éléments de base d'un système mais également a créer les éléments constituant un RdP ou un graphe de Markov. Le langage est outillé par KB3[14], la solution développée par EDF et propose des services de générations d'arbres de défaillances et de gestion des formalismes implémentés par les diverses bases de connaissance à ce jour disponibles comme les BDMP (Boolean logic Driven Markov Processes) [Bouissou & Bon 2003]. Dans [Bouissou & Seguin 2006], les auteurs comparent les langages AltaRica Data Flow et FIGARO et dressent les conclusions suivantes : FIGARO offre la possibilité de s'adapter à tous les formalismes de SdF et est donc orienté pour la construction d'outil ; quant à AltaRica Data flow, il convient à l'obtention de modèles pour conduire les études de systèmes et est donc destiné directement aux analyses de SdF.

Ce tour d'horizon est loin d'être exhaustif en ce qui concerne les méthodes d'analyse quantitatives de SdF des systèmes. On pourrait mentionner des travaux sur les réseaux bayésiens, les processus de markov [Zio 2009] ou les nouvelles techniques abordant les arbres de défaillance comme les Priority Dynamic Fault Trees (PDFT) [Merle et al. 2009]. Néanmoins, nous avons pu présenter succinctement quelques-uns des formalismes employés. Le défi de l'ISBM des systèmes à fortes contraintes de SdF est de pouvoir intégrer dans le processus l'usage de tels formalismes et langages.

I.3.4 Intégrer les analyses de SdF à l'ISBM

Le défi du développement de systèmes complexes à fortes contraintes de SdF, comme les systèmes de sécurité ou les systèmes en interaction critique avec l'Homme ou son environnement, est de réussir à introduire efficacement les méthodes d'analyse de SdF dans le processus de conception. Pour la réalisation de ce type de système, nous avons conclu de l'état de l'art qu'il était nécessaire d'employer un processus d'IS utilisant des modèles. Nous avons insisté sur les avantages des modèles utilisant le concept Orienté Objet (OO) et avons préconisé l'emploi du langage SysML. Nous avons ensuite évoqué les méthodes d'analyses de SdF qu'il

[12] BPA-DAS : http://www.3ds.com.
[13] SIMFIA : http://www.apsys.eads.net.
[14] KB3 : http://rdsoft.edf.fr.

était nécessaire de déployer. Nous étudions dans les paragraphes suivants, la mise en place de ces analyses au cœur de l'ISBM, en mentionnant les travaux précédents, mêlant analyse de SdF et emploi de modèle OO, puis en proposant notre MÉthode D'Intégration des analyses Sdf à l'Ingénierie Système (MéDISIS).

I.3.4.1 Utilisation d'UML/SysML pour les analyses de SdF

L'adoption de SysML est encore récente, nous avons mentionné les travaux l'exploitant pour la conception de systèmes complexes (§I.2.5.3). Néanmoins, aucun d'entre eux ne s'est clairement intéressé à l'utilisation du langage pour préparer et mener les analyses de SdF. SysML hérite du langage UML, celui ci est utilisé depuis de nombreuses années et plusieurs travaux se sont penchés sur son emploi lors d'analyse de SdF. SysML dispose de nombreuses représentations communes avec UML. Il peut donc bénéficier des expériences que nous allons maintenant décrire. Comme UML est majoritairement employé au sein de la communauté du logiciel, la plupart des travaux référencés ici sont relatifs à la SdF de composants logiciels.

Cette problématique de prise en compte d'exigences de SdF et de réalisation d'analyse de SdF à partir de modèles UML révèle de nombreux challenges pour la construction des modèles et leur exploitation. La problématique de la modélisation de la SdF de composants logiciels, décrits en UML, est présentée dans [Leangsukun et al. 2003]. Les auteurs proposent de définir de nouveaux stéréotypes afin de décrire le comportement dysfonctionnel du système en utilisant UML. Les modèles utilisant ces stéréotypes sont automatiquement convertis sous le format d'entrée du logiciel SHARPE[15] utilisé pour générer des AdD et chaînes de Markov. Plusieurs générations de RdP et AdD à partir de représentations UML ont été développées [Zarras & Issarny 2001], [Bernardi et al. 2002], [Merseguer et al 2002], [Lopez-Grao et al. 2004]. Ces travaux sont réalisés à des fins d'évaluation de performances. Les aspects dysfonctionnels sont modélisés en UML en utilisant des règles de stéréotypages. [Zarras & Issarny 2001] ont réalisé un profil UML reprenant tous les concepts du domaine de la SdF tels qu'ils sont exposés dans [Laprie 1995], les stéréotypes y sont renforcés par l'utilisation de contraintes OCL (Object Constraint Language) [OMG 2006]. Le projet HIDE[16] regroupe les travaux les plus aboutis en terme d'usage d'UML pour les analyses de SdF. Le projet décrit dans [Bondavalli et al. 1999a], [Bondavalli et al. 2001] a pour objectif de faciliter les études SdF des systèmes à logiciel. Pour cela, le langage UML est utilisé comme modèle central afin de bénéficier de sa souplesse et son expressivité pour tous les acteurs d'un projet. Ensuite, les participants du projet ont défini un ensemble de transformations automatiques vers des formalismes et outils d'analyses SdF dédiés. Les modèles d'entrée UML expriment le comportement global du système, incluant les comportements défaillants. Afin de rendre possible les transformations de modèle, les auteurs définissent des restrictions d'utilisation d'UML et introduisent leurs propres stéréotypes relatifs à la SdF. [Bondavalli et al. 1999b] présentent un passage d'UML aux Réseaux de Petri Temporisés (RdPT). [Bernardi et al. 2002] et [Merseguer et al. 2002] proposent l'exploitation combinée des statemachines et SD pour obtenir des RdP composés. [Lopez-Grao et al. 2004] poursuivent ces travaux par la génération de Réseaux de Petri Stochastiques Généralisés (RdPSG). Des travaux comme [Pai & Dugan 2002] se sont inspirés des travaux du projet HIDE et ont développé d'autres types de transforma-

[15] SHARPE : Symbolic Hierarchical Automated Reliability and Performance Evaluator developé par Duke University. http://www.ee.duke.edu.

[16] HIDE : High-level Integrated Design Environment for dependability, projet européen ESPRIT LTR 27439 HIDE.

tions comme la création de Dynamic Fault Trees à partir de diagrammes de classe et de *statemachines*.

Les types de diagramme les plus utilisés dans la littérature sont les SD, Statemachines et AD, en effet les SD et AD offrent des vues séquentielles se prêtant parfaitement à l'approche des AdD et RdP. Quant aux *statemachines*, elles constituent une représentation d'automates à états, à la base du formalisme des RdP.

Les *statemachines* sont l'objet de multiples travaux d'analyse de SdF [King & Pooley 1999], [Pap et al. 2005], [Gilmore & Kloul 2005], [Iwu et al. 2007]. Ceci est dû au fait qu'ils sont la représentation la plus formelle offerte par UML et SysML. La rigueur de ces diagrammes permet le développement de méthodes d'analyse du modèle prouvant le respect de certains critères. [Pap et al. 2005] et [Iwu et al. 2007] proposent des méthodes et outils vérifiant la non atteignabilité d'états redoutés. [Iwu et al. 2007] proposent d'intégrer l'emploi de modèles UML avec la méthode PFS de formalisation d'exigence (Practical Formal Specification), l'utilisation d'UML ne se cantonne pas aux *statemachines*. Le modèle UML est utilisé dans une démarche l'associant à la formalisation de règle PFS. Les SD du système sont analysés afin de définir les risques à évaluer. Dans une approche voisine de celle de [Guiochet 2003], les messages échangés sont analysés afin de définir les erreurs et conséquences possibles. Les statemachines sont ensuite étudiées pour vérifier les règles définies par l'analyse précédente. [Pap et al. 2005] se focalisent sur la déclaration et l'analyse de *statemachines* afin de modéliser et prouver les caractéristiques du système. Pour réaliser une analyse formelle, les auteurs modifient et restreignent les méthodes de saisie des *statemachines*. Ils obtiennent ainsi deux formes canoniques de *statemachines* : les « restricted statemachines » et les « safe statemachines ». Ces descriptions sont analysées après définition des critères recherchés par des règles OCL. Les analyses sont réalisées avec l'outil SPIN[17], après une transformation du modèle UML sous le langage Promela (utilisé par l'outil) comme cela avait été présenté dans [Latella et al. 1999]. Enfin, on peut mentionner l'initiative de [Gilmore & Kloul 2005] analysant des *statemachines* pour l'évaluation de performances. Une fois encore le modèle est transformé pour atteindre un modèle formalisé et outillé : un « Multi-Terminal Binary Decision Diagram ».

Nous notons qu'UML, apprécié pour les avantages de l'orienté-objet, a permis de nombreuses modélisations pour la préparation d'analyse de SdF. L'objectif commun de tous ces travaux était d'intégrer les analyses de SdF au cycle de développement système. L'utilisation d'UML dans ces travaux avait pour but de faciliter la communication entre les équipes de développement et d'analyse de SdF autour d'un modèle commun facilement compréhensible. Ces travaux utilisent le langage UML à une étape du développement où le comportement dysfonctionnel individuel des composants est connu. C'est-à-dire que les représentations, à partir desquelles les modèles d'analyse SdF sont créés, sont obtenues après une étape d'identification similaire à la technique des AMDEC présentée au paragraphe I.3.2. Ces modèles ne reflètent pas uniquement les aspects fonctionnels du système, mais s'intéressent aussi aux mécanismes de la SdF. C'est pourquoi leur création nécessite l'emploi de stéréotypes d'entités UML capables de modéliser les défaillances et leur propagation. Les principaux désavantages de ces approches sont qu'elles restent spécifiques au traitement de systèmes logiciels et que l'utilisation de stéréotypes rend leur réemploi standard sur différents outils UML impossible. Toutefois, ces travaux ont tracé de nombreuses correspondances entre les modèles UML et des formalismes permettant la réalisation d'analyses de performances et de SdF va-

[17] Page de ressources sur le model-checker Spin : http://spinroot.com/spin/whatispin.html.

riées, comme le calcul de disponibilité, la prévision de fiabilité, la vérification de caractéristiques temporelles, la possibilité de reconfiguration dans des états sûrs.

Dans la suite de cette thèse, nous nous appuyons sur les points forts de ces travaux et montrons que le passage au langage SysML permettra de progresser dans l'intégration de la SdF à la conception, et ceci pour tout type de systèmes. Dans [Bondavalli et al. 2001], [Zarras & Issarny 2001] et [Iwu et al. 2007], les auteurs présentent des méthodes intégrant l'usage d'UML et des techniques dédiées à la SdF au sein d'EDS mêlant outils de saisie UML et outils de traitement de modèles formels. Ces EDS utilisent comme point central, des modèles UML contraints exprimant les défaillances du système. Nous souhaitons appréhender le problème plus en amont. Nous proposons une méthode décrivant l'enchaînement des études SdF, allant du recensement du comportement dysfonctionnel particulier des composants, à la connexion aux formalismes spécifiques aux analyses de SdF, présentés lors du paragraphe I.3.3.

I.3.4.2 MéDISIS : Les analyses SdF au cœur de l'ISBM

Le but de la méthode, développée au cours de cette thèse, est d'intégrer efficacement le sous-processus d'analyse de SdF des systèmes au processus plus global d'ISBM. Cette méthode intitulée MéDISIS (Méthode D'Intégration des analyses de Sdf à l'Ingénierie Système) a fait l'objet de diverses publications [David et al. 2008a,b], [David et al. 2009a,b]. Son but est de proposer un enchaînement de traitements de l'information et des connaissances, incluant la création, l'expression, l'analyse, la pérennisation et la réutilisation. Ces traitements doivent être supportés par des outils et répertoires formant un EDS cohérent. Nous avons souhaité définir cette méthode pour l'étude de systèmes complexes et multi-technologiques. Nous faisons également le choix de réutiliser les outils des industriels tout en proposant une méthode ne ciblant, en particulier, aucun d'entre eux. Ces travaux doivent être utilisables sur un large ensemble d'outils de conception. Les objectifs à atteindre par la méthode sont les suivants :

- Faciliter la transmission de connaissances entre équipes.
- Accélérer la réalisation des études de SdF.
- Organiser l'exploitation commune des connaissances sous forme de modèles.
- Permettre la réutilisation des connaissances entre projets.
- Identifier les besoins d'analyse et réaliser le suivi de leurs résultats.
- Améliorer la cohérence et la qualité des analyses SdF.

Nous avons montré au paragraphe I.2.5 que le langage SysML est le plus prometteur pour soutenir l'ISBM de systèmes physiques. Ses larges capacités d'expressivité en font le candidat idéal pour écrire le modèle d'entrée de la méthode MéDISIS. Nous considérons donc que la source d'information initiale à MéDISIS est un modèle SysML purement fonctionnel, résultat de l'analyse fonctionnelle. La méthode a pour objectif de fournir le plus vite possible les indicateurs sur la SdF du système sous forme de scénarios de défaillance et de caractéristiques quantifiées. Pour cela, nous proposons des procédures permettant de réaliser les analyses de SdF décrites aux paragraphes I.3.2 et I.3.3, reprenant le cycle complet d'analyse de la SdF.

La méthode est soumise à différentes contraintes dues à son positionnement et à son rôle dans le processus d'ISBM. Elle doit être utilisable à divers niveaux de détails du système, être itérative (permettre la descente dans ces niveaux de détails) et répétable. Les connaissances

collectées au cours de l'étude concernant les composants, sous-systèmes et systèmes doivent être réutilisables pour d'autres projets afin de remplir le rôle des approches modernes d'IS. Pour cela, une Base de données des Comportements Dysfonctionnels (BCD) constitue l'élément central de MéDISIS. Cette BCD regroupe les connaissances collectées sur les mécanismes de défaillances des entités composant le système analysé. Elle sera décrite dans les chapitres II et III de ce manuscrit. MéDISIS inclut des services de génération automatique d'analyses ou de modèles, utilisés pour réaliser le plus efficacement possible chaque étape de la démarche. La figure I.10 schématise MéDISIS en présentant l'enchaînement des procédures effectuées et l'utilisation de la BCD.

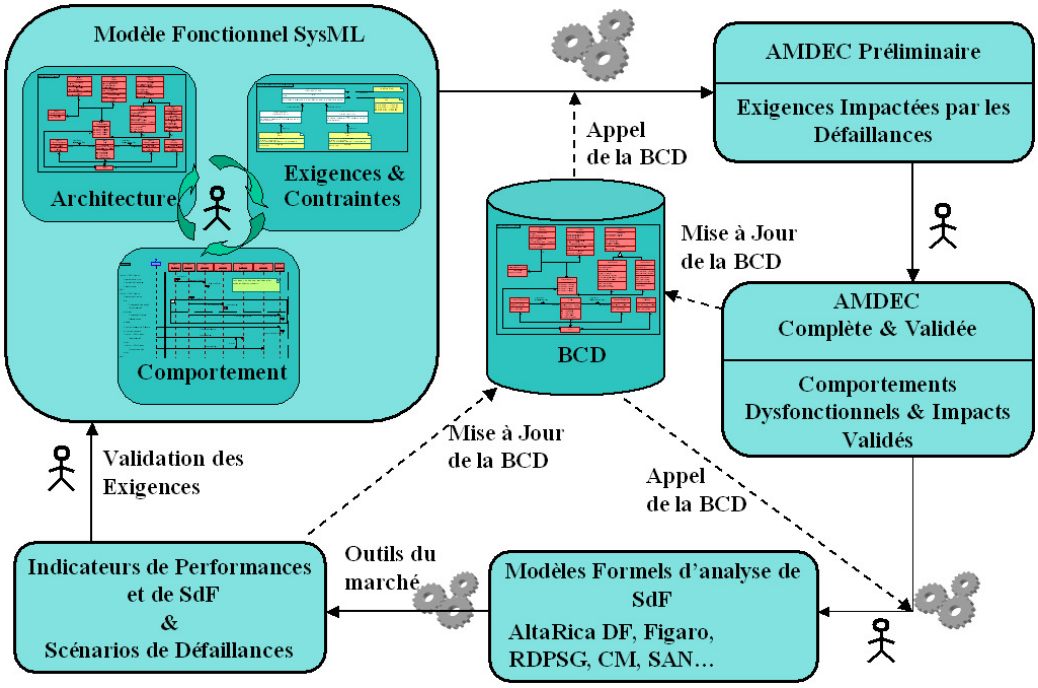


Figure I.10 *La méthode MéDISIS*

MéDISIS est une méthode déductive et incrémentale se déroulant en 3 phases majeures :

- Recherche du comportement dysfonctionnel des composants et des influences entre composants voisins. Identification des exigences impactées. Cette phase est réalisée par une AMDEC munie d'une assistance automatisée.
- Construction assistée d'un modèle utilisant une représentation formelle du comportement fonctionnel et dysfonctionnel du système.
- Analyse outillée des modèles formels, validation des exigences, retour sur la conception.

L'analyse est conduite sur le système ou ses sous-systèmes. Le lancement est précédé d'une définition des limites de l'étude, facilitée par l'organisation naturelle des modèles SysML. On considère qu'un modèle présentant l'architecture et le comportement purement fonctionnel est disponible, ainsi qu'une première répartition des exigences. Ce modèle est issu des premières phases du processus d'ISBM. Un premier recensement des exigences de SdF peut avoir été réalisé au préalable en utilisant par exemple une Analyse Préliminaire des Risques [Mortureux 2002].

La première phase, détaillée dans [David et al. 2008b], [David et al. 2009a], consiste à la réalisation d'une AMDEC du système. Comme nous l'avons vu au paragraphe I.3.2, l'AMDEC permet un recensement rapide des MdD de chaque composant. Cependant, la lourdeur de la méthode rend peu efficace son utilisation au sein d'une démarche d'IS. La littérature (§ I.3.2.3) a proposé de remédier à ce problème par l'automatisation de parties de l'analyse. Nous suivons cet axe de recherche en proposant un service de génération assistée d'AMDEC. Nous nous démarquons des travaux précédents en analysant des modèles purement fonctionnels et non dédiés à une technologie spécifique. Nous analysons les modèles SysML fournis par le concepteur. Ce traitement est détaillé au second chapitre de cette thèse. L'AMDEC permet de réaliser une détection systématique des risques pesant sur le système très tôt dans la conception. Ces risques sont analysés et classifiés afin de diriger la suite de l'étude. Cette première phase est critique pour la suite de l'analyse car elle est chargée de définir le comportement défaillant des composants qui sera quantifié et qualifié pour le système global dans la suite de l'étude. L'AMDEC est réalisée en conjonction avec l'utilisation de la BCD. Cette base permet de faire appel au Retour d'EXpérience (REX) sur des systèmes ou composants similaires déjà utilisés. Ce REX est primordial pour les AMDEC où seuls les jugements d'experts s'expriment. Dans le cadre de MéDISIS, cette connexion à la BCD est double. En effet, les résultats de l'AMDEC validés par les experts sont également réintroduits dans la BCD pour préciser et enrichir ses connaissances. Cette phase est découpée en deux étapes : la création d'un rapport AMDEC préliminaire destiné à soutenir les experts et éliminer la lourdeur de l'étude, et une finalisation manuelle de l'AMDEC par les experts dont le jugement et la prise de décision finale ne peuvent être automatisés.

La seconde phase de l'étude consiste à construire les modèles qui vont permettre une analyse formelle de la SdF du système. Comme nous l'avons vu au paragraphe I.3.3, ces études nécessitent l'emploi de langages et formalismes complexes et souvent difficiles à mettre en place. Il est donc judicieux, comme le mentionnent également [Dumas et al.2008] de fournir une aide à la génération de ces modèles au cours du processus d'ISBM. Ces modèles doivent reprendre la définition de la structure et des comportements fonctionnels et dysfonctionnels des composants. Ils redéfinissent donc une partie déjà couverte par le modèle du concepteur. Il est alors intéressant d'assister la création de ce modèle en traduisant le modèle SysML fonctionnel. Pour cela, nous adoptons une philosophie proche des travaux présentés au paragraphe I.3.4.1, à la différence que nous analysons des modèles dénués de stéréotypes particuliers, qui expriment uniquement l'aspect fonctionnel du système. Cette étape permet d'accélérer l'analyse de SdF et d'assurer une plus grande cohérence entre les modèles du projet, manipulés dans l'EDS. Cette étape est complétée par l'utilisation de la BCD. Elle permet de faire appel à des éléments de modèles dysfonctionnels déjà connus, qui viennent se greffer à la partie fonctionnelle. La BCD peut alors être vue comme une bibliothèque de modèles dysfonctionnels. Les modèles cibles de cette transformation sont ceux présentés au paragraphe I.3.3. Dans ce travail, nous nous sommes focalisés sur l'emploi du langage AltaRica Data Flow (DF) dont la sémantique proche de l'OO permet une traduction efficace de modèles SysML, comme nous l'évoquons dans [David et al. 2009b]. Ce langage particulièrement bien outillé permet de créer des modèles exploitables pour le calcul d'indicateurs quantifiés et la recherche de scénarios de défaillance. Cette étape est l'objet du troisième chapitre.

La dernière phase de MéDISIS est l'exploitation du modèle ainsi créé pour analyser le respect des exigences formulées sur le système. Les conclusions de cette exploitation permettent de juger l'architecture proposée et de valider les choix de conception. Au cours de cette étude on obtient des résultats d'analyse qui permettent de fournir les documents normatifs attendus

lors du développement. Les conclusions orientent également l'effort d'analyse en pointant par exemple la nécessité d'étudier plus en détail un sous-système crucial pour la SdF du système. La cohérence des modèles employés permet d'obtenir un suivi rigoureux des exigences et une bonne focalisation des tests. L'usage de la BCD centralise la gestion des connaissances créées et réutilisées tout au long de MéDISIS.

I.4 Conclusion

Dans ce premier chapitre, nous avons présenté la problématique du développement des systèmes complexes à fortes contraintes de Sûreté de Fonctionnement. Les nouveaux besoins de l'industrie motivent le développement de systèmes multi-technologiques devant remplir leur mission en garantissant leur sûreté de fonctionnement. Les attentes en terme de coûts et de durée du développement, ainsi que la complexité des systèmes conçus ont conduit les ingénieurs à adopter de nouvelles méthodes d'Ingénierie Système. Ces méthodes offrent une meilleure structuration et expressivité des études en utilisant des modèles du système. Elles constituent ce que nous appellerons l'Ingénierie Système Basée sur les Modèles. Nous avons vu que l'Ingénierie Système Basée sur les Modèles se déroule suivant un processus d'Ingénierie Système, organisant l'utilisation de diverses méthodes recevant l'appui d'outils. Ces outils et les répertoires qui les accompagnent doivent former un Environnement de Développement Système cohérent afin de permettre une conception efficace.

Le défi de la conception des systèmes à fortes contraintes de Sûreté de Fonctionnement est la nécessité de réaliser efficacement une importante phase d'analyse des attributs de Sûreté de Fonctionnement. Ces analyses doivent être intégrées au mieux au processus d'Ingénierie Système Basée sur les Modèles et à l'Environnement de Développement Système le soutenant. Nous avons détaillé le déroulement attendu des analyses de Sûreté de Fonctionnement en soulignant les phases de création et de réutilisation de l'information concernant le comportement dysfonctionnel du système. L'analyse de Sûreté de Fonctionnement débute par un recensement des Modes de Défaillance des composants du système effectué par une étude de type Analyse des Modes de Défaillance de leurs Effets et Criticité (AMDEC). Ensuite, on cherche à qualifier et quantifier de manière formelle le comportement du système global. Pour cela, des techniques utilisant des modèles Réseaux de Petri ou des langages tels qu'AltaRica Data Flow sont déployées. Comme nous l'a enseigné la littérature, ces techniques ralentissent le processus d' Ingénierie Système et sont difficiles à mettre en place car elles nécessitent une expertise du langage de modélisation et du système modélisé. L'axe de recherche que nous avons dégagé est donc d'optimiser l'emploi de ces techniques dans le cadre de l'Ingénierie Système Basée sur les Modèles.

Nous avons identifié le langage SysML comme étant le meilleur candidat pour la conception de systèmes physiques complexes et multi-technologiques. Nous avons présenté rapidement sa structure et les possibilités de modélisation qu'il offre. Nous avons observé qu'il satisfait aux besoins de modélisation de l'Ingénierie Système Basée sur les Modèles. Le modèle central du processus d'ingénierie donne le découpage structurel du système, spécifiant les flux échangés entre les composants physiques dans une représentation de type boîte noire. Ceci se retrouve dans le cas d'un modèle exprimé avec SysML dans l'axe structurel du langage que sont les Block Definition Diagrams et Internal Block Diagrams. Sur cette vue structurelle viennent se superposer des descriptions du comportement individuel des éléments, exprimant leur façon de faire évoluer les variables qu'ils manipulent. Ces aspects sont généralement donnés par les fonctions de transfert réalisées et les changements d'états possibles au sein d'un automate à états (particulier à chaque composant). Ces définitions sont assurées en SysML par l'utilisation de *statemachines*, de diagrammes paramétriques ou de diagrammes d'activités spécifiques à un composant donné. La synchronisation de ces comportements permet de décrire le comportement complet du système. Elle se fait au niveau structurel par la connexion de différents flux, et de façon dynamique par la spécification du déclenchement de

fonctions, la création de ressources ou la survenue d'événements. En SysML, ces aspects sont précisés par le biais des diagrammes d'activités et de séquences.

Nous avons évoqué l'utilisation de son prédécesseur UML au sein des analyses de Sûreté de Fonctionnement et avons étudié les travaux d'amélioration des AMDEC. En fin de ce chapitre, ceci nous a permis de proposer la méthode MéDISIS pour la réalisation d'analyses de Sûreté de Fonctionnement au sein du processus d'Ingénierie Système Basée sur les Modèles. Nous avons présenté l'enchaînement de tâches proposé par MéDISIS, qui organise l'analyse de Sûreté de Fonctionnement, de la recherche des défaillances particulières des composants à l'évaluation du comportement dysfonctionnel du système global. Nous avons évoqué les objectifs de ces différents traitements et mentionné l'utilisation d'une Base de données des Comportements Dysfonctionnels comme organe de gestion de la connaissance. Les différentes activités de MéDISIS sont détaillées dans les prochains chapitres. Le chapitre II se focalise sur la création d'AMDEC à partir de modèles SysML purement fonctionnels. Le chapitre III s'intéresse à la création assistée de modèles formels pour l'analyse de Sûreté de Fonctionnement.

Chapitre II

La réalisation d'AMDEC à partir de modèles SysML : un support, des apports, une assistance au déploiement

Résumé du Chapitre II :

Ce chapitre présente l'étude réalisée pour la mise au point de la première phase de la méthode MéDISIS. Il s'agit de faciliter la réalisation d'AMDEC à partir de l'étude des modèles fonctionnels SysML du système créés par son concepteur. Pour cela, nous dégageons le métamodèle sous-jacent des données constituant une AMDEC et nous ciblons comment ces informations sont disponibles dans les modèles SysML en se référant au métamodèle de ce dernier. Des règles d'analyse de modèles SysML pour la réalisation d'AMDEC sont alors formulées. A partir de ces règles nous proposons une procédure de traitement des modèles SysML pour la génération d'AMDEC préliminaire, pour laquelle nous présentons différentes possibilités d'automatisation. Cette procédure fait appel au retour d'expérience que nous choisissons de gérer à travers une Base de données des Comportements Dysfonctionnels, pour laquelle nous fournissons un métamodèle organisant les connaisances et respectant la spécification SysML. Cette analyse nous permet de formuler des conseils de modélisation en SysML, favorisant l'expressivité du modèle SysML du concepteur en vue d'une étude de type AMDEC.

II.1 Introduction

Nous nous intéressons, dans le cadre de la méthode MéDISIS, à la réalisation d'AMDEC prévisionnelles afférentes aux systèmes en cours de conception. Ce terme est utilisé en opposition aux AMDEC opérationnelles réalisées sur des matériels ou processus existants. La littérature met l'accent sur le fait que la réalisation des AMDEC doit être conduite au cœur du processus de développement des systèmes [Price 1996], [Theoh & Case 2004a, b], [Papadopoulos et al. 2004], [Bassetto 2005]. L'outil de réalisation d'AMDEC a donc pour vocation d'être intégré au sein de l'Environnement de Développement Système en tant qu'outil d'analyse du modèle (§ I.3.4.2). Pour ce faire, les modèles développés dans l'Environnement de Développement Système doivent offrir une sémantique portant, de façon claire et non ambiguë, les informations nécessaires aux AMDEC. Les bénéfices attendus pour la réalisation d'AMDEC s'expriment alors aussi bien en terme de qualité que de rapidité et pérennité d'étude. Ces travaux abordent des aspects de la construction d'AMDEC encore peu explorés par les travaux antérieurs (§ I.3.2.3), comme le recensement exhaustif des composants à étudier ou l'assistance à la navigation du modèle de conception analysé. Le langage SysML est porteur de la sémantique ad hoc, les progrès notés pour la création d'AMDEC résultent de l'utilisation de sa syntaxe. Cette création est facilitée par l'exploitation des fichiers générés par les outils de modélisation. Le pouvoir de modélisation de SysML permet en outre d'accéder à de nouvelles dimensions d'analyse permettant d'ancrer un peu plus l'AMDEC dans les processus d'Ingénierie Système et ainsi de valoriser son utilisation. Nous évoquerons en particulier la prise en compte des exigences, la recherche de Modes de Défaillance et des variables de flux que ces Modes impactent.

Les AMDEC produit sont initiées par le recensement d'entités permettant de balayer l'ensemble du système étudié. Ces entités sont généralement les fonctions ou les composants du système. Dans le premier cas, on qualifie l'AMDEC de fonctionnelle, dans le second on constitue une AMDEC composant. Nous allons comparer les deux approches et présenter comment un modèle fonctionnel rédigé en SysML peut être exploité pour ces études. Nous développerons ce point par l'étude des métamodèles de données sous-jacents et mentionnerons comment la réalisation d'AMDEC à partir de modèles SysML peut s'appuyer sur une Base de données de Comportements Dysfonctionnels dont le métamodèle sera détaillé. Nous expliciterons comment la recherche d'information au cœur du processus AMDEC peut être partagée entre l'exploitation automatisée de modèles SysML, l'accès à une Base de données de Comportements Dysfonctionnels et l'analyse du modèle par les experts participant à l'élaboration de l'AMDEC. Les algorithmes et les alternatives majeures pour créer un outil de synthèse automatique d'AMDEC composant seront présentés.

II.2 Formalisation de la sémantique des AMDEC

II.2.1 Les notions de composant et fonction

Dans le cadre de la méthode que nous décrivons, nous retenons le métamodèle de la Figure II.1 comme expression d'un composant. On y exprime qu'un composant peut être détaillé en sous-composants. Un composant peut : avoir un identifiant, exécuter des fonctions et posséder des interfaces, ces dernières sont reliées entre elles par des connecteurs permettant l'assemblage des composants. La Figure II.2 présente les relations entre fonctions, composants, interfaces et flux. Deux types d'interfaces se distinguent, les entrées par lesquelles le composant reçoit les flux et les sorties qui fournissent des flux à l'extérieur du composant. La fonction exprime les relations entre flux entrants et sortants.

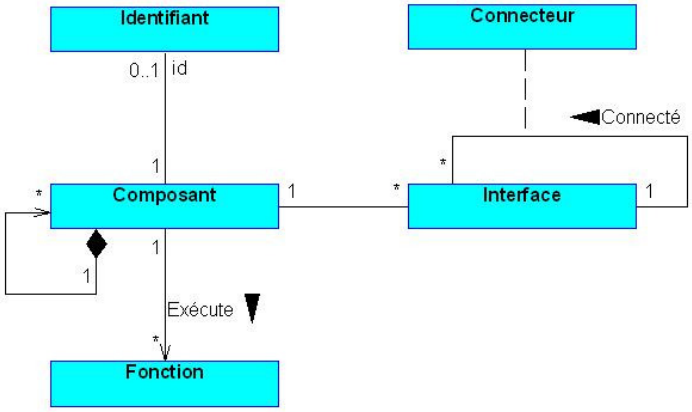


Figure II.1 *Métamodèle d'un composant pour l'ingénierie système*

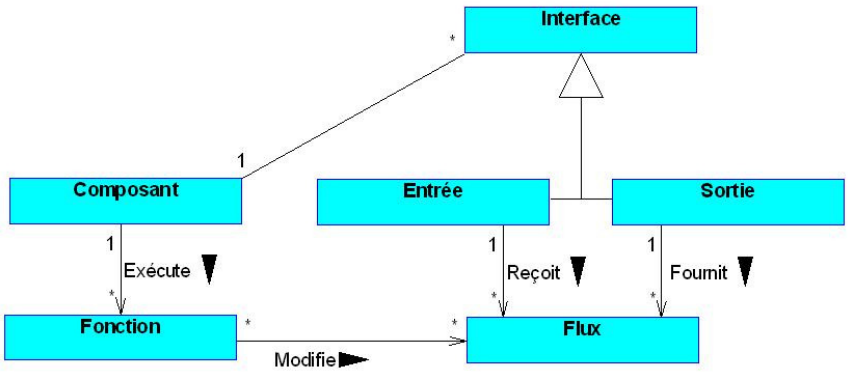


Figure II.2 *Métamodèle du traitement de flux*

II.2.2 L'AMDEC fonctionnelle

Ce type d'étude est normalisé dans différentes normes : MIL-STD1629A, IEC 60 812. Ces normes décrivent les différentes étapes que nous allons détailler dans la suite de cette section. Cette analyse consiste à étudier les unes après les autres les différentes fonctions conduisant le système à remplir sa mission. Ces fonctions ont été identifiées et définies au cours de l'analyse fonctionnelle du système, pendant laquelle les concepteurs ont dégagé la suite d'actions nécessaire pour réaliser la mission confiée au système développé. Pour chacune de

ces fonctions, l'étude consiste à recenser les modes de défaillance conduisant à une dégradation du service rendu par la fonction. Pour chacun de ces modes il s'agit de recenser leurs causes, leurs effets à différents niveaux (local, système), les moyens de détection, les moyens de correction et de qualifier la criticité du mode à travers sa gravité, son occurrence et sa détectabilité. Les fonctions sont usuellement identifiées par un nom ou une référence (ex : freiner, F1.1). Les causes considèrent l'ensemble des phénomènes et mécanismes susceptibles de faire apparaître le mode ainsi que les entités responsables de ces phénomènes. Ces entités peuvent être indifféremment internes ou externes au système étudié. Les effets locaux décrivent la fonction dans les conditions du mode. Les effets aux niveaux supérieurs décrivent les conséquences du mode sur les attributs du système, son comportement ainsi que sur son environnement. Les moyens de détection et de correction indiquent les fonctions et les composants nécessaires respectivement à la détection et réduction du risque lié au mode. Les différentes cotations (gravité, occurrence, détectabilité) peuvent être vues comme des attributs des MdD détaillés. Elles sont fixées par les experts. Pour l'analyse de certains systèmes, on peut également détailler la phase de vie ou de mission du système durant laquelle la défaillance peut intervenir, nous aborderons cette problématique au paragraphe II.3.4.3.

L'AMDEC fonctionnelle est, traditionnellement, réalisée très tôt durant le processus de conception à un instant où les composants réalisant les traitements ne sont pas encore clairement définis. Ce type d'étude peut également être pertinent lorsque le nombre de composants à étudier n'est pas en adéquation avec le temps alloué et les résultats attendus pour l'étude.

Une forme tabulaire classique de l'AMDEC fonctionnelle est donnée Figure II.3.

Identifiant	Fonction	MdD	Causes	Effets locaux	Effets Système	Gravité	Occurrence	Détectabilité	Criticité	Moyens de détection	Moyens de correction

Figure II.3 *Tableau générique pour AMDEC fonctionnelle*

II.2.2.1 Le métamodèle

Comme spécifié au paragraphe I.3.4.2, pour analyser le processus de réalisation d'une AMDEC, nous avons choisi d'étudier la structure de données créée au travers de son métamodèle. Ce métamodèle sera un point de comparaison avec l'AMDEC composant. Il sera également mis en correspondance avec le métamodèle SysML, afin d'exprimer les points de recouvrement autorisant la synthèse d'AMDEC du système à partir de son modèle SysML. Le métamodèle proposé pour l'AMDEC fonctionnelle est présenté par la Figure II.4.

Lors de la réalisation d'une AMDEC fonctionnelle les analystes sont amenés à rechercher les différentes informations présentes dans le métamodèle. Le raisonnement parcourt le métamodèle en passant d'une entité connue aux entités manquantes et recherchées. Le point de départ de l'AMDEC fonctionnelle est l'identification des fonctions. Ces fonctions possèdent généralement un identifiant pour la cohérence et la traçabilité de l'analyse vis-à-vis des analyses précédentes.

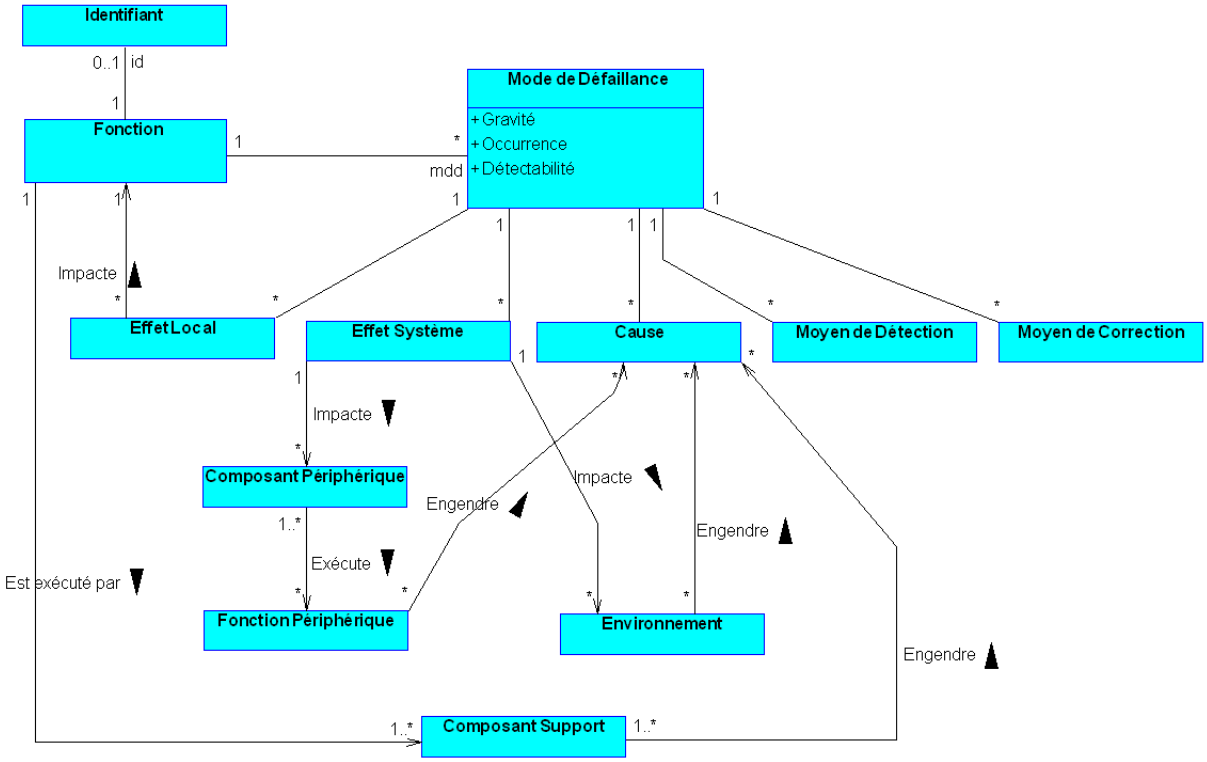


Figure II.4 *Métamodèle d'une AMDEC fonctionnelle*

II.2.2.2 Expression des MdD

Le premier travail conduit est la recherche des MdD. Cette recherche s'appuie sur le retour d'expérience ou l'utilisation d'une systématique de raisonnement. Dans le premier cas, le retour d'expérience peut être présent sous différents niveaux de formalisation. Il peut s'agir d'un avis d'expert, du résultat de l'utilisation d'une étude précédente ou d'un référentiel normalisé et formaté dans lequel les fonctions et leurs modes de défaillance sont répertoriés. En l'absence de ce type de données, il est possible de mener l'étude par l'utilisation d'une systématique. Une telle systématique peut également être utile en complément du retour d'expérience. D'un point de vue fonctionnelle, les normes utilisées fournissent des lignes directrices pour la recherche des MdD. Par exemple, dans la norme MIL-STD 1629A, pour chaque fonction il convient de considérer les MdD suivants :

- Fonction exécutée de façon prématurée.
- Fonction exécutée trop tardivement.
- Fonction intermittente.
- Fonction refusant de s'arrêter.
- Défaillance en cours d'exécution.
- Fonction dégradée.
- Autres conditions de défaillances propres à la fonction étudiée.

II.2.2.3 Causes, effets : naissance et propagation des défaillances

La suite de l'étude est la recherche de la cinétique de la défaillance. Il s'agit de mettre en évidence les éléments ayant causé la défaillance, ainsi que les éléments impactés à différentes échelles par la défaillance. Pour dégager clairement ces notions, il est nécessaire de faire un point sur la mécanique d'exécution des fonctions. Une fonction est vue comme une suite d'opérations délivrant des sorties en fonction d'entrées et de paramètres d'influence. La fonction est considérée défaillante si les sorties obtenues sont en dehors de leurs domaines attendus. Les caractéristiques surveillées pour les sorties sont relatives à leur amplitude, leur variation ou leur délai et durée de mise à disposition. Les entrées peuvent être des sorties des fonctions amont ou être imposées par l'utilisateur, la sortie peut être consommée par l'utilisateur ou être l'entrée d'une autre fonction. On nommera ces fonctions, fonctions périphériques. Les paramètres d'influences sont liés aux caractéristiques des composants exécutant la fonction, ainsi qu'aux conditions environnementales dans lesquelles se déroulent la fonction étudiée. Nous désignerons les composants participant à la réalisation de la fonction comme composants support. Leurs variables d'état influant sur le résultat des opérations constituant la fonction sont des paramètres d'influence. Il en est de même pour toutes les variables d'environnement modifiant le résultat de ces opérations.

On décrit les effets locaux traduisant le comportement dysfonctionnel de la fonction, c'est à dire les sorties fournies dans l'état de défaillance considéré. Le déroulement dysfonctionnel de la fonction impacte le système au travers des flux reçus par les composants périphériques modifiant les entrées des opérations qu'ils supportent. L'évaluation des effets de la modification de ces entrées donne de proche en proche les effets à l'échelle du système. Vient ensuite la recherche des causes de la défaillance. Trouver les causes de la défaillance de la fonction, revient à identifier les paramètres d'influence et les entrées pouvant faire dévier les sorties de leurs valeurs attendues. On recherche donc les paramètres environnementaux, les caractéristiques des composants support et les entrées fournies par les fonctions périphériques. On recherche ensuite ce qui pourrait conduire ces paramètres à prendre des valeurs conduisant ou participant à la déviation des sorties de la fonction de leurs valeurs attendues. Ceci peut être, par exemple, la casse d'un élément des composants support, la dégradation des conditions environnementales ou la défaillance d'une fonction périphérique.

L'expression des causes et effets dépend grandement du niveau de détail et des connaissances disponibles au moment de l'étude. En effet, si l'AMDEC fonctionnelle est réalisée avec une absence totale de connaissance des composants qui réaliseront les fonctions, l'analyse ne peut mettre en évidence que des risques d'erreurs des fonctions et de l'environnement. Dans ce cas, le concept de composant support demeure tout de même viable. Ces derniers sont simplement considérés comme une boîte noire réalisant la fonction et possédant autant d'entrées et sorties qu'il a été défini pour la fonction supportée.

II.2.2.4 Cotations et gestion du risque

Les dernières phases de l'AMDEC fonctionnelle sont la cotation de chaque risque identifié et l'organisation de la gestion de ces risques. La cotation des risques est réalisée pour classer ces derniers et donner des priorités dans leur traitement. La cotation permet de justifier une décision de traiter ou non un MdD. Elle fournit également une indication sur le levier à utiliser pour réduire le risque. En effet, la cotation passe par le calcul du RPN (Risk Priority Number) constitué de 3 composantes : la gravité, l'occurrence et la détectabilité. Obtenir les

trois indices permet de savoir comment réduire efficacement le risque. Enfin, il est souhaité de répertorier les moyens de détection possibles pour chaque MdD, ainsi que la déclaration des actions correctives à mener pour réduire le risque. Les moyens de détection peuvent indifféremment porter sur la détection des causes ou des effets du MdD. Il reste néanmoins préférable de pouvoir observer les causes dans un souci de prévention du MdD. Les moyens de correction visent à réduire une ou plusieurs des composantes du risque. Les moyens mis en place sont directement en relations avec les variables identifiées pour l'environnement et les composants. Les moyens de détection cherchent à observer de telles variables. Les moyens de correction visent à influer sur les valeurs atteintes par ces variables.

L'AMDEC fonctionnelle permet de mettre en évidence très tôt durant le processus de conception, les risques pesant sur la structure fonctionnelle choisie. La précision des résultats est directement reliée à la complétude des connaissances sur l'architecture fonctionnelle et physique du système. Lorsque l'on dispose du niveau de détail suffisant, il nous semble plus pertinent de réaliser l'AMDEC composant du système. Cela permet d'analyser plus finement le système et de mettre en évidence les parades matérielles à mettre en place. Néanmoins, l'AMDEC fonctionnelle est un bon moyen pour diriger les efforts d'analyse et de conception sur les parties et fonctions critiques du système.

II.2.3 L'AMDEC composant

Cette analyse consiste à déterminer pour chaque composant du système le comportement dysfonctionnel possible de chacun d'entre eux. L'analyse doit être précédée d'une phase de délimitation des frontières de l'étude. En particulier, il est primordial de définir le niveau de détail utilisé pour décrire l'architecture physique du système analysé. La démarche globale de ce type d'analyse est la même que pour les AMDEC fonctionnelles à l'exception près que l'étude porte sur la caractérisation des risques relatifs aux composants physique et non aux fonctions. Cette variante implique des changements fondamentaux dans l'analyse, notamment dans la façon de recenser et de considérer la propagation des défaillances. L'analyse permet de mettre en évidence des MdD au plus proche des phénomènes physiques responsables des défaillances. Les MdD de chaque composant identifié sont recensés, on exprime la fonction remplie par le composant, les effets des MdD sur cette fonction et le composant, ainsi que les effets sur le système et les causes des MdD. Chaque risque est évalué par un RPN. Des moyens de détection et de correction sont proposés. La forme tabulaire de l'AMDEC composant est très proche de celle de l'AMDEC fonctionnelle à la différence que la tête du tableau est tenue par les composants et que fonctions et composants apparaissent clairement. Un exemple de trame pour l'AMDEC composant est donné en Figure II.5.

Identifiant	Composant	Fonction	Mode de défaillance	Cause	Effet local	Effet Système	Gravité	Occurrence	Détectabilité	Criticité	Moyen de détection	Moyen de correction

Figure II.5 *Tableau générique pour AMDEC composant*

II.2.3.1 Le métamodèle & spécificités induites

Nous proposons un métamodèle figurant l'organisation des données d'une AMDEC composant sur la Figure II.6. Ce métamodèle est à comparer à celui de l'AMDEC fonctionnelle (Figure II.4). On y retrouve globalement les mêmes entités, cependant les associations y sont différentes. L'élément de base est ici le composant pour lequel on recherche les différentes informations présentes dans le métamodèle. Ce point bouleverse les raisonnements déployés dans l'AMDEC fonctionnelle. En effet, on ne considère plus les MdD de la même façon, les effets sont considérés à l'échelle du composant et les fonctions sont recensées à des fins de compréhension de la cinétique des défaillances. La compréhension des effets sur le système passe par une étape d'analyse supplémentaire, la défaillance au niveau composant doit être répercutée au niveau de la fonction exécutée. Cette fonction réalise les sorties fournies par le composant étudié, ces sorties potentiellement modifiées ont une influence sur le reste du système.

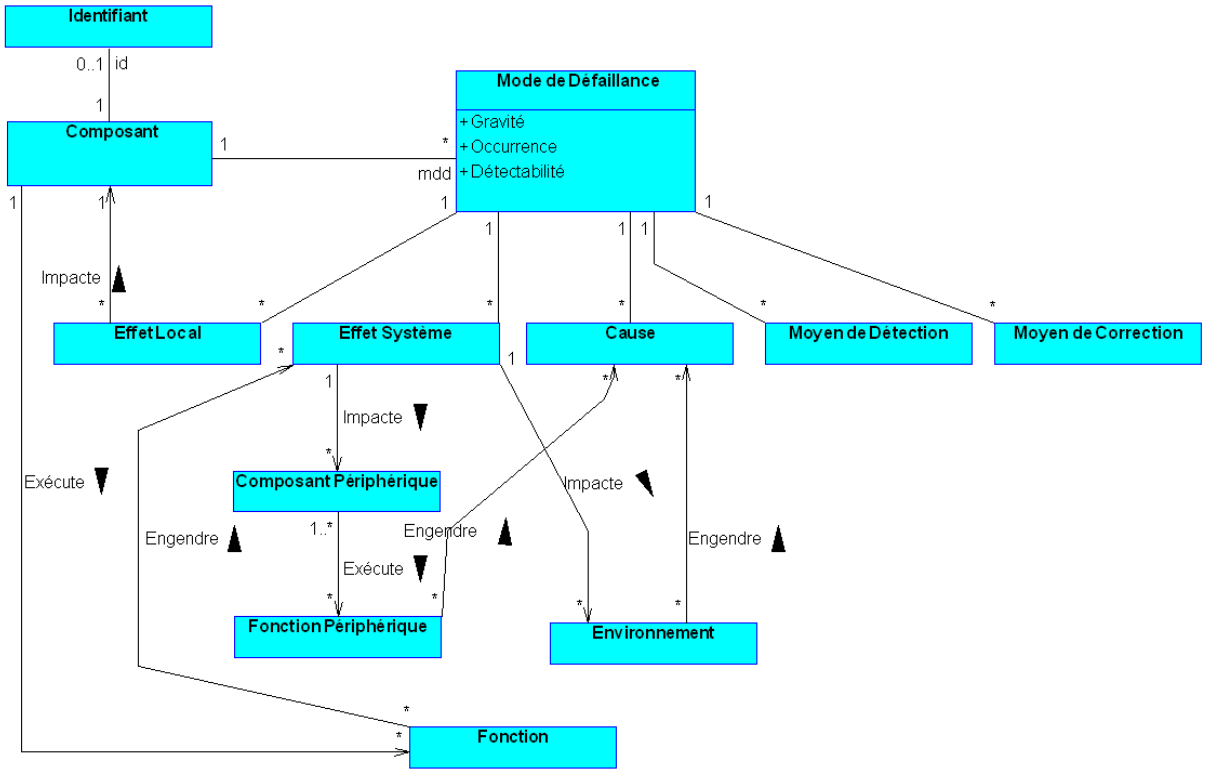


Figure II.6 *Métamodèle d'une AMDEC composant*

II.2.3.2 Expression des MdD

Le cœur de l'AMDEC composant est donc de retrouver tous les MdD que peuvent présenter les composants de l'architecture du système. Ces MdD peuvent être vus comme des états atteignables par le composant, à la différence des MdD des fonctions exprimées sous forme d'opération. Par exemple, pour un composant vanne, un MdD est le blocage de la vanne. On introduit ainsi un nouvel état pour le composant. Cet état est appelé état dysfonctionnel. Si on avait considéré la fonction remplie par la vanne comme objet de l'étude, on aurait retenu comme MdD : fluide délivré à tort, fluide bloqué à tort ou fluide délivré partiellement. Néanmoins, il arrive couramment que le MdD de la fonction réalisée par le composant soit confon-

du et retenu (à tort) comme le MdD du composant. L'étude d'un capteur conduit à proposer comme MdD : pas de détection et détection intempestive, qui sont en fait les MdD de la fonction « détecter un phénomène physique ». Les MdD correspondants devraient être : cellule sensible non réactive, cellule sensible hyper-réactive. Cependant, dans le cas de composant élémentaire, où la fonction et le composant support sont uniques, il est plus direct et léger de traiter les MdD de la fonction comme MdD du composant et les MdD du composant comme cause des MdD déclarés.

Quel que soit le point de vue retenu par l'analyste en charge de l'AMDEC, la source de connaissance des MdD est principalement le retour d'expérience sous forme de base de donnée ou d'avis d'expert. Dans le cas où aucun retour n'est disponible, il est nécessaire de revenir à une approche systématique. Or, devant l'extrême variété des composants pouvant être étudiés, il est nécessaire de revenir à une systématique portant sur l'évaluation des MdD envisageables pour les fonctions supportées par le composant analysé. La systématique employable est alors la même que celle présentée pour l'AMDEC fonctionnelle. La découverte des MdD composant relève alors, comme nous l'avons dit précédemment, de la recherche des causes de ces MdD fonctionnels.

II.2.3.3 Recherche des causes et effets

Comme nous l'avons dit dans le paragraphe précédant, la recherche des effets de MdD composant passe par l'impact de la dégradation du composant sur la fonction remplie. Les résultats de la fonction sont propagés ensuite au travers de l'architecture par identification des liaisons entre le composant étudié et ses voisins. Les causes peuvent provenir de composants périphériques exécutant leurs fonctions ou de l'environnement. La recherche des causes passe également par la connaissance des interfaces possédées par le composant. Ces interfaces sont, une fois le composant déployé, reliées par des connecteurs à d'autres interfaces de composant ou directement à l'environnement.

II.2.3.4 Cotations et gestion du risque

L'opération de cotation et de définition des priorités des risques est similaire à celle menée lors de l'AMDEC. La recherche de moyens de détection doit s'appuyer sur l'étude des variables d'influence identifiées comme cause du MdD étudié et sur celle des variables modifiées par le MdD. La mise en place de moyens de détection dépend de l'observabilité de ces variables ainsi que du coût des systèmes à implanter. De même, les actions correctives vont permettre de contrôler les variables influençant ou influencées par le MdD. Il est également possible d'introduire une nouvelle variable d'influence limitant ou interdisant la survenue de sorties inattendues et redoutées.

L'AMDEC composant nous permet donc d'identifier les comportements dysfonctionnels que peuvent présenter chacun des composants de l'architecture étudiée. Cette étude peut être menée dès qu'un premier niveau de description physique est disponible. La définition du niveau de détail de l'étude a une grande incidence sur la pertinence et la faisabilité de l'AMDEC. De plus, nous pouvons observer qu'une AMDEC composant réalisée à un niveau de décomposition « boîte noire » est similaire à la réalisation de l'AMDEC fonctionnel du système. Les boîtes noires devenant un simple artefact de modélisation pour représenter l'enchaînement entre fonctions.

II.2.4 Synthèse du raisonnement produisant les AMDEC

Dans les paragraphes précédents (II.2.2 et II.2.3) nous avons décrit le cheminement de la réflexion induite lors de la réalisation d'une AMDEC. Afin de montrer comment un modèle SysML peut être utilisé comme support pour ces études, nous allons isoler les étapes de réflexion à mener lors de l'établissement des AMDEC. Nous mettrons plus particulièrement en évidence les éléments recherchés et le raisonnement employé pour identifier chacun d'eux. Nous reprenons la définition du raisonnement de [Theoh & Case 2004a] : un raisonnement est un processus de prise de décision basée sur la compréhension des informations disponibles. Lors d'une AMDEC, ce processus de prise de décision est mené pour établir des relations de cause à effet à partir de vues structurelles et fonctionnelles du système pour générer un tableau AMDEC. L'étude AMDEC passe par les étapes suivantes :

1. Recensement des composants à étudier.

2. Pour chaque composant, rechercher les MdD.

3. Pour chaque MdD, rechercher les causes possibles.

4. Pour chaque couple MdD/causes possibles, rechercher les effets possibles sur les composants périphériques.

5. Pour chaque effet, réaliser la cotation du risque (RPN), proposer des moyens de détection et spécifier les actions à mener pour diminuer le risque.

Toutes ces étapes engendrent la constitution d'un tableau regroupant les résultats de l'AMDEC. L'étape 1 se base sur une vue structurelle du système, les éléments à identifier sont les composants du système. L'étape 2 fait appel aux connaissances des experts (organisées ou non dans des bases de données), les éléments relevés sont le nom des MdD. L'étape 3 est la première faisant appel au raisonnement de causalité, elle est basée sur l'expérience des experts et l'analyse des vues du système. En effet, comme nous l'avons vu au paragraphe II.2.3.3 la défaillance peut provenir de l'entourage du composant affecté. Pour cela, il convient d'identifier les interfaces du composant ouvertes à l'influence des composants périphériques. Pour l'étape 3, on recense les causes connues par les experts, les interfaces offertes par le composant et les composants périphériques, puis on conserve ceux dont la fonction peut provoquer le MdD. L'étape 4 fait également appel au raisonnement causal : le MdD du composant induit un changement de l'exécution de ses fonctions. Ces fonctions dégradées produisent leurs sorties, transmises par les interfaces du composant aux composants périphériques. La réception de ces flux potentiellement erronés provoque des changements d'état des composants périphériques et de leurs traitements, ces informations sont les effets du MdD. Les éléments à produire ici sont les comportements des fonctions altérées, les flux altérés, les interfaces les relayant et les composants périphériques touchés ainsi que leur comportement modifié. Enfin, l'étape 5 fait appel aux jugements et connaissances des experts pour produire le RPN et communiquer les solutions adéquates pour la gestion du MdD.

Afin de rendre les AMDEC plus opérationnelles et de les intégrer au mieux à une MBM et un EDS, nous avons, au paragraphe I.3.4.2, expliqué qu'il serait bénéfique de réaliser l'AMDEC à partir de modèles utilisant un langage bien spécifié et d'automatiser certaines parties du raisonnement. L'EDS que nous mettons en avant dans la méthode MédiSIS prend

le parti d'employer SysML comme langage support. Nous allons donc montrer dans la suite de cette section, d'une part que le raisonnement induit par l'AMDEC peut être mené en considérant les informations contenues dans les modèles SysML (§ II.3), et d'autre part que ce raisonnement peut être en partie automatisé (§ II.4). Pour cela, nous expliciterons comment isoler les éléments nécessaires à chaque étape dans un modèle SysML. Nous argumenterons en nous référant systématiquement au métamodèle du langage SysML afin de montrer la présence de l'information et d'identifier son organisation. De nombreux éléments de cette réflexion ont déjà été publiés dans [David et al. 2008a,b] et [David et al. 2009a].

II.3 Exploitation de SysML pour l'AMDEC

II.3.1 Motivations

L'objet de cette sous-section est de montrer comment retrouver les informations répertoriées au paragraphe II.2.4, nécessaires à l'AMDEC. Nous souhaitons montrer que les différents éléments à produire tels que les composants, les fonctions ou les liens entre eux peuvent être isolés de façon sûre dans un modèle SysML. Pour cela, nous analysons le métamodèle du langage SysML afin d'identifier comment sont représentés ces concepts. Il existe souvent plusieurs artefacts de modélisation pour un seul concept recherché, nous détaillons donc l'ensemble des composantes à considérer lors de l'analyse du modèle SysML.

II.3.2 Référentiel d'étude

Nous supposons que le travail que nous décrivons se déroule une fois l'analyse fonctionnelle du système effectuée. Ceci afin d'obtenir les informations mises en valeur dans le paragraphe II.2.4. C'est pourquoi, nous nous axons sur l'étude de modèles issus de la phase de conception préliminaire, car nous pensons que c'est durant cette phase que l'utilisation de SysML pour l'AMDEC est le plus pertinent et bénéfique. Nous cherchons, en effet, à rendre l'AMDEC moins fastidieuse grâce à l'analyse d'un modèle structuré. Or, l'AMDEC composant est la plus difficile à mener car elle est réalisée sur des définitions plus avancées et plus complexes du système et parce qu'elle mêle la recherche de défaillances physiques et fonctionnelles. Les bénéfices de la méthode que nous présentons pour la réalisation d'AMDEC se feront d'autant plus sentir que le modèle étudié sera complexe. De plus, le langage SysML se montre beaucoup plus puissant pour les phases de conception où une architecture physique est, ou commence à être, dégagée.

II.3.3 Les modèles SysML comme support aux AMDEC

Nous allons maintenant montrer que les modèles SysML sont un bon support pour la réalisation d'AMDEC et que les informations dégagées pour le processus AMDEC peuvent être identifiées dans un modèle SysML.

II.3.3.1 Identifier les composants du système

L'AMDEC est réalisée sur les bases d'une analyse fonctionnelle. Nous supposons que le modèle SysML que nous étudions est obtenu à la suite de cette analyse, mais nous ne spécifions pas de méthodologie d'IS employée. La première tâche effectuée au cours d'une AMDEC, est d'identifier tous les composants qui doivent être analysés au niveau de décomposition fixé par l'analyste (étape 1 du raisonnement AMDEC).

En considérant les diagrammes constituant l'axe architectural de SysML, il est possible d'identifier l'ensemble des composants d'un système à étudier. Pour cela, il faut recenser l'ensemble des *parts* utilisés dans le modèle. Les *parts* apparaissent dans plusieurs diagrammes des modèles SysML, au niveau de la définition des concepts au sein des BDD, dans les IBD où le déploiement des composants est montré ou dans les diagrammes de séquences où leur participation aux interactions est modélisé. Chaque composant du système est ainsi repré-

senté par un *part* à l'identifiant unique au sein de tout le modèle. De plus, les *parts* ne modélisent que des relations de composition et sont par la même une utilisation d'un *block* dans un contexte particulier. Ceci garantit que chaque *part* du modèle est un composant du système réel.

Identifier les *parts* revient ainsi à identifier les composants du système. Pour chacun d'entre eux, on obtient un nom désignant le rôle du *part* dans le contexte du modèle. Ce rôle est celui joué par le *block* typant le *part*. On obtient ainsi le type de composant que l'on aura à étudier. La connaissance du *part* et du *block* instancié donne une dénomination et une identification des composants à analyser au cours de l'AMDEC. Si on prend l'exemple simple d'une voiture on trouvera dans le modèle 4 *parts* de type Roue dont les noms respectifs seront : roueAvantDroite, roueAvantGauche, roueArrièreDroite et roueArrièreGauche. Chacun de ces éléments fera l'objet d'une portion de tableau AMDEC. Ils ont pu être identifiés par l'analyse du modèle de conception préliminaire du système réalisé en SysML. Une première partie du métamodèle du processus est donc couvert par le modèle SysML, à savoir le découpage du système autour du concept de composant. Ce concept est explicité dans la partie du métamodèle SysML hérité de l'UML (UML4SysML) présenté figure II.7. Il y est représenté l'association entre *block* et *part*. Les *parts* sont des *properties* possédées par les *blocks* et des instances contenues par ce type d'élément. Un *block* est un *classifieur* permettant de représenter un concept d'élément de modélisation aux propriétés connues. Les *propriétés* d'un *block* peuvent être organisées entre elles, par exemple par l'utilisation de *connecteurs*. Le *block* possède donc une structure propre, il est donc vu comme un *classifieur structuré*. Les *propriétés* d'un *block* dont les *parts* font partie possèdent un *classifieur* définissant leur type propre. En SysML ce *classifieur* est évidemment un *block*. On retrouve ici la notion recherchée de composant dont le type est connu. C'est à ce type que sont rattachés la majorité des éléments du modèle que nous exploiterons pour remplir l'AMDEC.

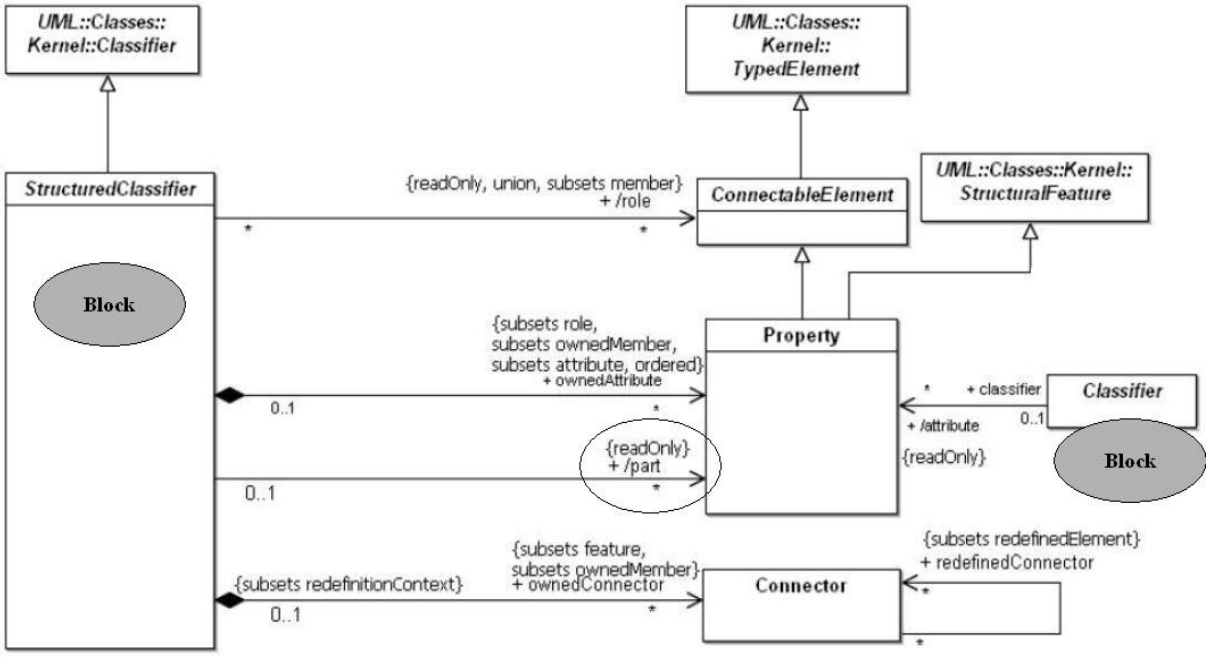


Figure II.7 *Métamodèle UML4SysML de la notion de « structured classifier »*

Du point de vue des notations employées dans les modèles, il est simple de repérer les *parts* et leur type. Au sein d'un IBD la notation est conforme à celle indiquée sur la figure II.8a, au sein d'un BDD les *parts* sont indiqués dans un des compartiments de *propriétés* des *blocks* représenté comme cela est montré sur la figure II.8b. Dans les deux cas, la syntaxe est de la forme « nom_du_part: Nom_du_Block_type ».

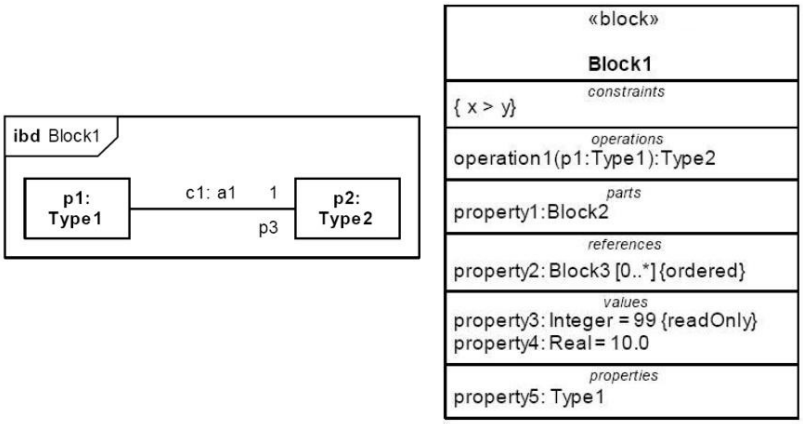


Figure II.8 *Notations pour* blocks *et* parts

Règle N°1 : Les composants identifiés par l'AMDEC sont les *parts* du modèle. Leur nom donne le nom du composant. Leur type est donné par le *block* dont le *part* est une instance.

II.3.3.2 Identifier les fonctions réalisées par les composants

Comme nous l'avons remarqué sur le métamodèle de l'AMDEC composant, il est primordial de connaître les fonctions exécutées par les composants étudiés. Il faut donc retrouver dans le modèle SysML les traitements affectés aux différents composants que l'on aura relevés. En SysML, la déclaration des fonctions réalisées fait partie de l'axe fonctionnel de la modélisation. La déclaration des *operations* est associée aux *blocks*, il est donc nécessaire d'avoir au préalable relevé le type de chaque *part*. En effet, SysML offre la possibilité de définir le comportement valable pour une classe d'objets. Ce comportement prend en compte des paramètres dépendant du contexte d'emploi du *block*, c'est-à-dire les *value properties* associées aux *parts*. Les fonctions interviennent dans la description du comportement du système, à travers les trois artefacts suivants :

- Les *activités* transformant au sein des *blocks*, les entrées en sorties.
- Les *statemachines* principalement utilisées pour décrire comment un *block* se comporte en présence d'événements.
- Les *interactions* exprimant comment les *parts* interagissent par envois de messages.

Au sens AMDEC, la fonction d'un composant désigne le service que va rendre le composant du point de vue du système global. La fonction recherchée est donc la participation du composant à la mission du système et non l'ensemble des traitements internes qu'il réalise.
La fonction d'un composant est l'expression définissant la mise à disposition de sorties correctes aux composants et sous-systèmes qui lui sont connectés. L'information recherchée

pour renseigner les AMDEC est le nom des *operations* qu'il exécute au cours des *interactions* auxquelles il participe et les noms des *activités* exercées par le composant. Le détail des *activités*, les *statemachines* et les *interactions* internes au composant définissent et expriment comment ces fonctions sont réalisées et ne sont donc pas à examiner à cet instant de la réalisation de l'AMDEC. En effet, à ce stade nous construisons la structure de l'AMDEC. Le comportement des défaillances est à évaluer une fois celle-ci obtenue. Ceci participe à la volonté de ne pas noyer le processus AMDEC dans un flot d'informations risquant de diluer les points importants lors de la mise en évidence des dangers au niveau système et mission.

Les éléments utilisés pour identifier les fonctions remplies par les composants, à disposition de l'analyste chargé de mener l'AMDEC différeront grandement en fonction de l'avancement de la phase de design. En effet, il sera rare que le modèle SysML soit développé jusqu'à la projection de la partie comportementale sur les composants de l'architecture. Dans un grand nombre de projets, l'architecture et le comportement sont modélisés de façon disjointe et le lien formel entre les composants et l'exécution des fonctions est souvent absent. Les éléments les plus souvent indiqués sont les *opérations*, elles apparaissent comme des propriétés des blocks et sont ainsi facilement repérables comme le montre la figure II.8b. Dans la pratique, les fonctions à considérer pour les composants sont les *opérations* du *block* typant le *part* analysé. Identifier les *activités* exécutées par les composants s'avère tout de même plus pertinent que de ne considérer que les seules *opérations*. En effet, les *activités* sont représentées au sein de diagrammes d'activités couvrant toute la chaîne de transformation de données, flux et énergie utilisée par le système pour remplir sa mission.

Les techniques liant formellement une *activité*, ou une *action* la composant, à un *block* sont variées et doivent être connues de l'analyste AMDEC afin d'isoler les interactions à étudier dans le modèle SysML.

- La première technique utilisée est l'allocation d'*action* à un *block* ou directement à un *part*. Cette représentation, également désignée par le terme de « partition » ou en anglais par le terme « swimlane », consiste à créer un lien formel indiquant les entités responsables de l'exécution d'une ou d'un groupe d'activités. La relation en SysML est nommée *allocate from* une *action*, *to* un *block* ou un *part*. Graphiquement, elle peut être indiquée dans une table d'allocation (ex. figure II.9), par l'utilisation de *swimlanes* ou d'un compartiment (ex. figure II.10). Dans le cadre de l'AMDEC, il convient de rechercher les *actions* liées directement au *part* étudié ou au *block* lui conférant son type.

table [activity] ProvidePower [Allocation Tree for Provide Power Activities]

type	name	end	relation	end	type	name
activity	a1:ProportionPower	from	allocate	to	block	PowerControlUnit
activity	a2:ProvideGasPower	from	allocate	to	block	InternalCombustionEngine
activity	a3:ControlElectricPower	from	allocate	to	block	ElectricalPowerController
activity	a4:ProvideElectricPower	from	allocate	to	block	ElectricalMotorGenerator
objectNode	driveCurrent	from	allocate	to	itemFlow	i1:ElectricCurrent

Figure II.9 *Exemple de table d'allocation*

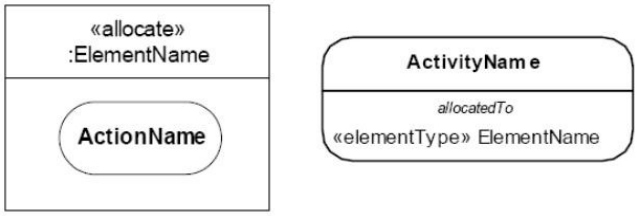


Figure II.10 *Allocation d'activité*

- La seconde technique utilisée pour dédier une action à un block particulier est de la déclarer comme un appel d'opération. L'action devient une « call operation action », et porte le nom d'une opération définie pour un block particulier. Ici, les call operation actions sont recensées pour nommer les fonctions réalisées par le composant de la même façon que l'on a identifié les opérations.

Règle N°2 : Les fonctions de chaque composant sont identifiées en relevant :

- Les *operations* du *block* typant le *part*,
- Les *actions* allouées par une *swimlane* au *part* ou au *block* le typant,
- Les *actions* de type « call operation action » appelant une *operation* du *block* typant le *part*.

La connaissance des fonctions exécutées va permettre de poursuivre la réalisation de l'AMDEC en étudiant les causes et effets probables des défaillances des composants. Néanmoins, dans le cas (le plus fréquent) où le comportemental est décorrélé de l'architecture, l'analyse du modèle architectural en SysML va permettre d'obtenir des indications précieuses pour la suite de l'étude. En effet, comme nous l'avons vu précédemment l'AMDEC passe par l'analyse du voisinage de chaque composant, à travers l'étude des relations inter-composants structurées dans le modèle SysML.

II.3.3.3 Identifier les relations entre composants

Si le comportement exact de chaque composant peut être longtemps méconnu, on dégage généralement les relations et connexions existant entre composants plus tôt dans l'étude. L'analyse des connexions existantes permet de mettre en évidence des chemins de propagation et d'influence pour la survenue et le développement de MdD. Cette étude permet de couvrir une nouvelle partie du métamodèle de l'AMDEC composant (figure II.6), à savoir, la recherche des composants périphériques et par extension du paragraphe précédent la déduction des fonctions périphériques.

L'analyse s'effectue selon deux axes : l'architecture du système et les interactions composants. La prise en compte de modèles SysML non finalisés, au sens où certains liens fonctionnels et certaines communications ne sont pas projetés sur la structure, est alors possible. Cependant, l'analyse de la partie comportementale permettra de préciser l'utilisation faite par le système des liens présents dans l'architecture. Ainsi, l'utilisation de modèle SysML permet d'appréhender les deux aspects des relations entre composants : les interfaces offertes entre composants et ce qui transite ou peut transiter par ces interfaces.

Analyse de l'architecture

Lors de l'AMDEC, l'analyse est réalisée pour chaque composant. La première phase consiste à relever tous les liens existants dans l'architecture entre les composants. La spécification SysML met l'accent sur la déclaration de telles relations à travers la modélisation des *ports* et de leurs liaisons. Comme nous l'avons vu précédemment (§ I.2.5.2), les *blocks* et *parts* peuvent posséder deux types de ports : les *flow ports* et les *standard ports*. Le mécanisme de *ports* tel qu'il est implémenté dans SysML est détaillé sur le métamodèle de la figure II.11. Cette figure présente la notion de ports héritée d'UML ; les interfaces sont uniquement utilisées par les *standard ports*. Les *ports* y sont présentés comme des propriétés rattachées à des éléments de type « EncapsulatedClassifier » eux mêmes héritant des « StructuredClassifier ». Ces éléments sont des *blocks*, leurs instances (*parts*) héritent naturellement des mêmes propriétés, donc des mêmes *ports*. La figure II.12 présente la définition SysML des *flow ports* (spécifiques à SysML) héritant des *ports* UML. Les *ports*, quels qu'ils soient, sont les points d'échanges disponibles à la frontière des composants. Cette notion recouvre celle d'interface que nous avons utilisée pour décrire les composants soumis à l'AMDEC (Figures II.1 et II.2).

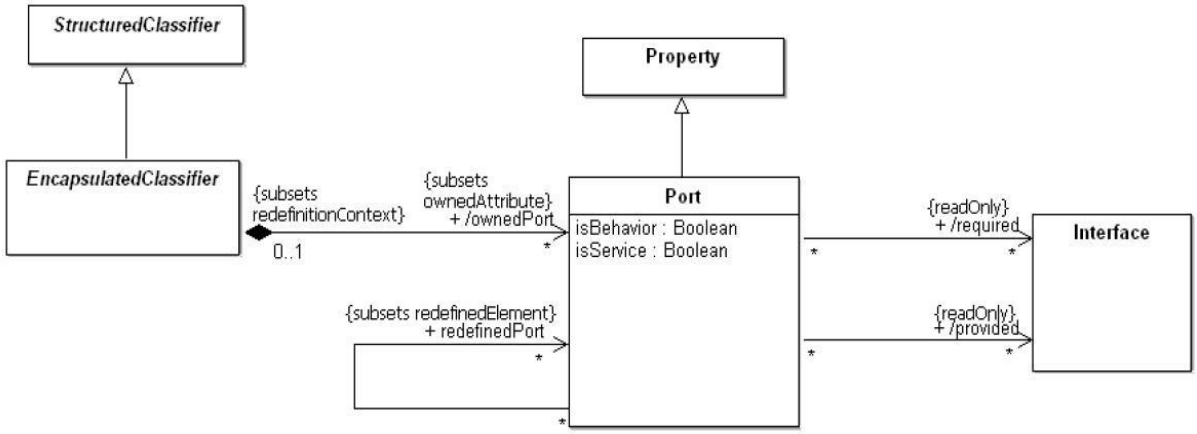


Figure II.11 *Métaclasse Port*

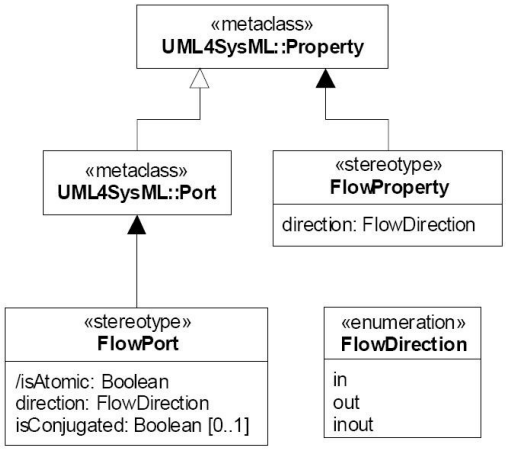


Figure II.12 *Les Flow Ports SysML*

En tant que propriétés, les *ports* sont typés ; ceci permet en SysML d'indiquer ce qui peut traverser le point de passage créé par le *port*. Ce type identifie les flux échangés par les composants, flux susceptibles de transmettre une défaillance, révélant les vulnérabilités des com-

posants étudiés. Les *flow ports*, introduits par SysML, permettent d'aller plus loin dans l'analyse grâce à leurs attributs spécifiques concernant le sens de passage des flux : *direction* et *isConjugated*. La *direction* (in, out ou inout) donne les sens de propagation possible et permet d'identifier si le *port* est une entrée ou une sortie. L'attribut *isConjugated* indique si le sens du *port* est piloté par le composant ou celui auquel il est connecté. On est alors en mesure de connaître également le sens de propagation potentiel des défaillances.

Les *flow ports* permettent de couvrir tous types d'interfaces (au sens AMDEC) ; ils sont utilisés pour passer des flux continus, des flux de données, des flux physiques et en particulier des flux d'énergie. Les *ports* sont répertoriés dans deux types de diagrammes : au sein des BDD, où les *ports* apparaissent dans des compartiments dédiés au même titre que les autres propriétés, ou au sein des IBD, où la représentation graphique des *flow ports* et *standard ports* est employée. Les interfaces des composants sont donc faciles à identifier dans les modèles SysML.

En SysML, les composants sont connectés entre eux par l'intermédiaire de leurs *ports*. Ces derniers sont reliés par des *connecteurs*. Le métamodèle des *connecteurs* est donné à la figure II.13. Les *connecteurs* possèdent au moins deux extrémités de type « ConnectorEnd », chacune de ces extrémités est associée à un objet de type « ConnectableElement ». Comme l'indique le métamodèle (figure II.7), les propriétés héritent des « ConnectableElement ». Les *ports,* en tant que propriétés, sont donc éligibles comme *connectorEnd*. Ainsi les liaisons à considérer lors de l'analyse AMDEC sont présentes sous la forme de *ports* connectés en SysML.

Graphiquement, ces *connecteurs* apparaissent sur les IBD du modèle, exprimant le contexte dans lequel sont déployés les *parts*. En identifiant les extrémités de chaque *connecteur,* aboutissant sur les *ports* d'un composant étudié, on peut relever l'ensemble de ses composants périphériques et ainsi progresser dans l'analyse des effets et des causes possibles des MdD.

Règle N°3 : Les interfaces et relations structurelles sont identifiées par l'analyse des *ports*. Les *ports* reliés entre eux par les *connecteurs* et les *parts* qui les possèdent sont respectivement les interfaces et les composants périphériques. Le sens de l'interface est déduit en considérant la *propriété direction* des *flow ports*.

Il est intéressant de noter que la spécification SysML autorise la connexion des *parts* entre eux mais également la connexion avec des éléments extérieurs, soit sous la forme de composants appartenant à un autre sous-ensemble (*part references*), soit sous la forme d'*acteurs* représentant des éléments environnementaux ou des systèmes extérieurs au système étudié. On accède ainsi aux interactions entre le système et son environnement.

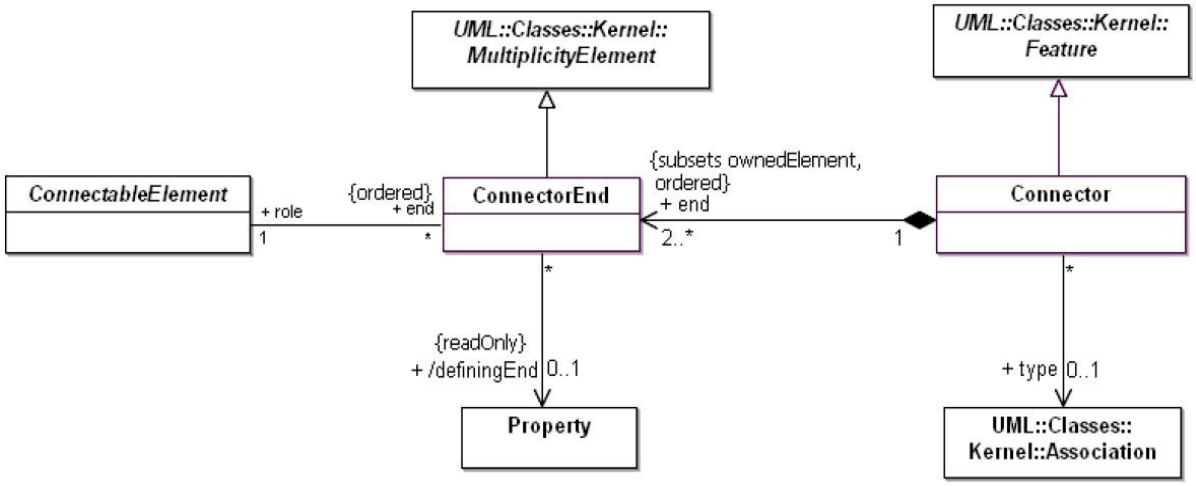


Figure II.13 *Métamodèle des connecteurs*

Les *item flows* couvrent une nouvelle partie des informations recherchées lorsque l'on réalise une AMDEC. Il s'agit de la recherche des flux échangés et traversant les interfaces comme nous l'avons mentionné sur la figure II.2. Le type des *flow ports* indique ce qui peut traverser, là où l'*item flow* indique ce qui traverse réellement. On obtient ainsi une connaissance des flux échangés à travers une dénomination précise accompagnée d'un type et optionnellement de diverses propriétés. Néanmoins, en l'absence de spécification précise de l'*item flow,* il reste pertinent de considérer le type des *ports* dont dispose le composant étudié. Toutes ces constructions de *ports*, *connecteurs* et flux permettent d'identifier les connexions structurelles dont disposent les composants du système. Néanmoins, si on considère que cette étude est menée précocement durant le processus de design, on ne peut garantir que le modèle SysML étudié sera suffisamment détaillé. Par exemple, des communications entre composants peuvent être décrites sans que le support physique de l'interaction ait été modélisé sur un IBD. Il faut donc maximiser l'ensemble des diagrammes analysés pour l'identification des composants périphériques et des relations établies avec eux, afin de réaliser une AMDEC de qualité.

L'analyse des interactions

Ainsi, il est nécessaire d'étudier l'axe comportemental du modèle pour mieux connaître les relations entre composants. Pour cela, nous nous intéressons aux diagrammes de séquences. En effet, comme montré au paragraphe I.2.5.2.3 ces diagrammes présentent les interactions existant entre les composants du système. Les interactions modélisent des relations de services entre les composants. Elles fournissent un nouvel angle de vue sur les composants et leur environnement. Formellement, une interaction est une suite d'échange de *messages* entre *parts*, dont les métamodèles sont repris figures II.14 et II.15. Les interactions sont composées de messages et de lignes de vie (« lifeline »). Les *lifelines* sont les représentants de *ConnectableElements* qui dans les faits seront principalement des *parts*. Chaque *part* participant à l'interaction se voit affilier une *lifeline* utilisée comme expéditeur et récepteur de *messages*. Aux *messages* sont associées suivant le *messagekind* une ou deux extrémités sous forme de *MessageEnd* : les *lifelines* réceptrices ou émettrices. Les *messages* sont des requêtes de service que l'émetteur demande au récepteur, cela consiste en l'appel d'une *operation* que le *part* receveur doit exécuter. En identifiant les *parts* en relation, par le biais de *messages*, avec le *part* étudié, on obtient les composants périphériques ainsi que les fonctions déclenchées.

L'étude des interactions est donc un moyen complémentaire pertinent pour poursuivre l'analyse AMDEC à partir de modèles SysML.

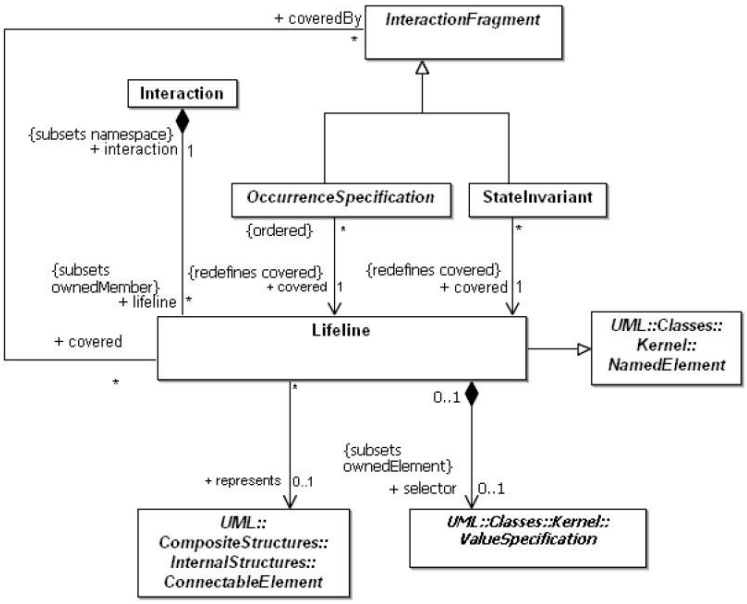


Figure II.14 *Métamodèle des « lifelines »*

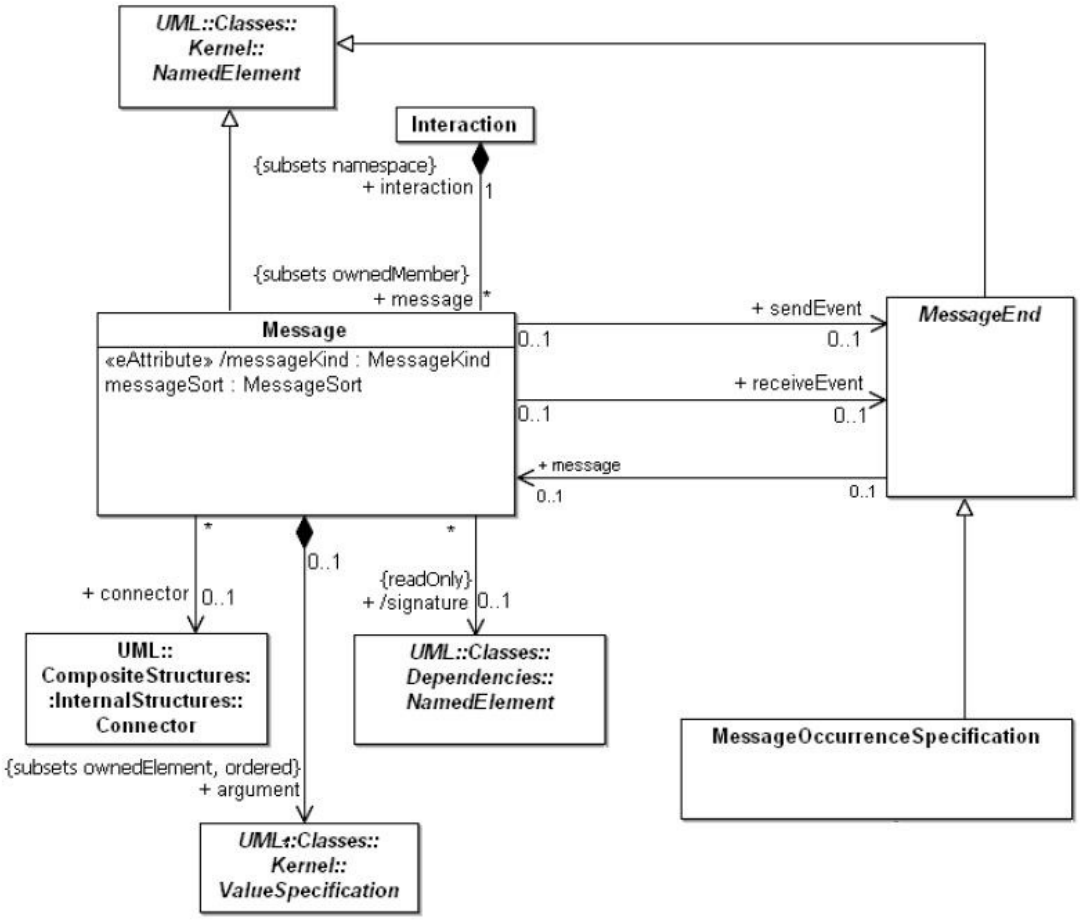


Figure II.15 *Métamodèle des messages*

Des connexions entre composants sont aussi présentes dans les Diagrammes d'Activité (AD). Les AD modélisent des *activités* réalisées par le système et décomposées en *actions*. Une *action* est au choix : la réalisation d'une *activité* (*CallBehaviorAction*) ou l'appel d'une *operation* (*CallOperationAction*). Cependant, on ne pourra retrouver les liens entre composants que si les *actions* constituant les *activités* sont projetées sur la structure par le biais d'*allocations*. Les AD décrivent des successions d'*actions* communicant entre elles par échange d'éléments. Ces échanges sont modélisés par des connections entre ports (*pins*) ou par des transmissions d'objets (*object nodes*). Les connexions sont des *activityEdges* reliant *pins* et *object nodes* aux *actions*. Les objets et *actions* sont regroupés dans le métamodèle (Figure II.16) sous la notion d'*activityNode*, désignant les composants d'une *activité*. On navigue dans le diagramme en parcourant les *incoming* et *outgoing activityEdges*. La mise en évidence d'un lien entre deux *actions*, allouées à des *parts* ou *blocks* différents, souligne un lien entre ces *parts* ou *blocks*.

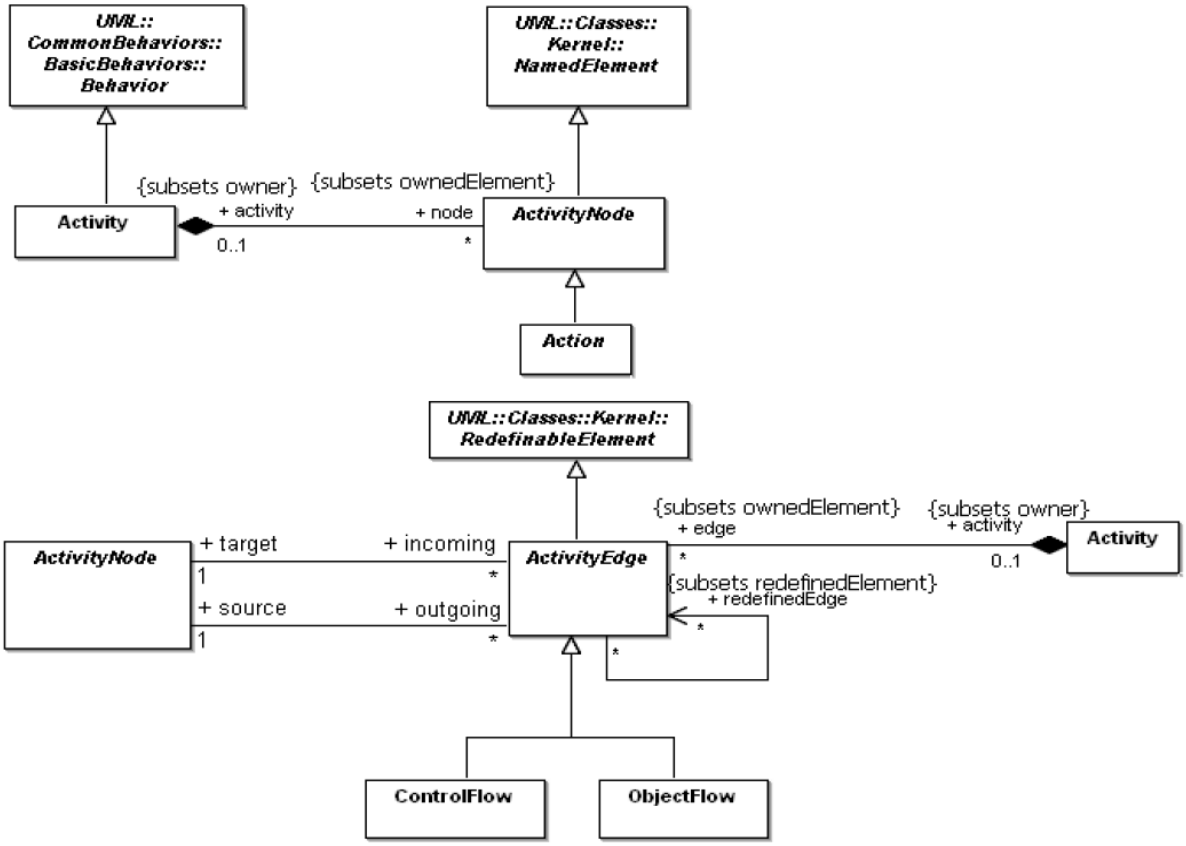


Figure II.16 *Métamodèles des liens entre* ActivityNodes

Règle N°4 : Les composants périphériques sont, en complément de ceux identifiés par la règle N°3, les *parts* recevant/émettant des *messages* du/vers le *part* étudié et les *parts* responsables d'*actions* en relation avec celle du *part* étudié.

Les processus que nous avons présentés dans ce paragraphe permettent de lancer la recherche d'informations relatives à des causes et effets des MdD. Les *parts* connectés sur un *port* de *direction in* ou *inout* et les *parts* émetteurs d'un *message* reçu par le *part* étudié sont susceptibles d'être cause de défaillances. Symétriquement, les *parts* connectés sur un *port* de

direction out ou *inout* et ceux récepteurs de *messages* émis par le *part* étudié ont de fortes chances de subir les effets des MdD considérés. La connaissance des types de connexion, des flux échangés et des messages envoyés permettra de préciser les différents aspects du développement des MdD à travers le système. Pour l'évaluation des causes et effets, la partie comportementale du modèle SysML se montrera très utile.

II.3.3.4 Identifier les causes et impacts des MdD par l'analyse du comportement fonctionnel

L'analyse de la partie comportementale du modèle permet de réaliser la phase dans laquelle on cherche à comprendre le déroulement des MdD. Comme nous l'avons vu précédemment, le langage SysML offre de nombreuses possibilités de modélisation du comportement. Ces éléments sont généralement associés aux composants du système. L'ensemble des types de comportement est couvert par l'utilisation des diagrammes SysML. On pourra analyser les transmissions de flux et leurs transformations avec les digrammes d'activités, les relations de requêtes de services inter-composants avec des contraintes temporelles au sein des diagrammes de séquences, et le comportement interne des composants en réactions aux événements au travers des statemachines. La stratégie d'exploitation, de ces modèles dans le cadre d'une AMDEC, consiste à identifier la partie de comportement modifiée par le MdD. Pour la recherche d'effets, on propage les modifications induites par le MdD dans le modèle structurel et fonctionnel. Pour la recherche de causes, on recherche les éléments amont pouvant conduire à la modification observée. Pour ces deux opérations, on utilise les interactions identifiées du paragraphe précédent (§ II.3.3.3). L'expression d'un MdD peut être par exemple l'absence d'un message au cours d'une interaction, la modification d'un flux traité lors d'une activité ou la suppression d'une transition d'une statemachine. Les possibilités d'analyse dépendent évidemment de la précision du modèle SysML, néanmoins on peut affirmer que celui-ci met à disposition tous les artefacts de modélisation nécessaires à l'expression des phénomènes de propagation de défaillances tels qu'ils sont développés au cours des AMDEC.

Règle N°5 : L'analyse de la propagation des défaillances des composants à travers sa représentation fonctionnelle, s'appuie sur les diagrammes de l'axe comportemental, afin de rechercher les causes et effets de ces défaillances. La propagation est réalisée dans le sens permis par les interfaces identifiées par la règle N°3 et les interactions relevées par la règle N°4.

II.3.4 Les apports de l'analyse de modèles SysML pour la réalisation d'AMDEC

Les modèles SysML prennent en compte bien d'autres aspects que ceux que nous utilisons jusqu'à présent pour la réalisation d'AMDEC. Nous allons maintenant montrer comment l'analyse de modèles SysML permet d'augmenter la portée des AMDEC au sein du processus de conception et d'analyse SdF. Nous préciserons les possibilités offertes pour le suivi des exigences afférentes au système et l'identification d'impacts quantifiés sur les caractéristiques et les performances du système. Nous évoquerons également les bénéfices en matière de conduite de projet et de gestion de l'information. Ces résultats ont été en partie présentés dans [David et al. 2009a].

II.3.4.1 Identifier les exigences satisfaites par les composants

Partie intégrante de tout projet de développement d'un système, la gestion des exigences est une priorité du langage SysML. Les liens forts existants au sein des modèles SysML entre composants et exigences, permettent d'intégrer le suivi d'exigences de façon directe dans les études AMDEC. Le metamodèle SysML (figure II.17) montre que les *requirements* (exigences) peuvent être liées à des éléments du modèle par le biais des relations *SatisfiedBy*, *RefinedBy* et *TracedTo* relatives à la notion très générale de « NamedElement ». L'information nécessaire pour le suivi d'exigences durant l'AMDEC est recélée par la relation *SatisfiedBy*. Elle est utilisée pour allouer la réalisation d'une exigence à un élément du modèle, notamment aux éléments de structure que sont les *blocks* et *parts*. Cette relation peut également lier les *requirements* à des éléments du modèle décrivant le comportement ou des caractéristiques physiques (ports, variables …). Il est alors nécessaire d'évaluer si ces entités (ports, variables …) sont possédées par les composants étudiés, afin de déterminer si ces derniers sont impliqués dans le respect des exigences. Les exigences peuvent également être composées et hiérarchisées avec les relations de composition (*containment*) et de dérivation (*derived*). Ainsi, si on recherche les exigences auxquelles un composant participe, il faut considérer les *requirements* en relation avec ceux satisfaits par le composant.

D'une façon similaire à la création assistée d'AMDEC que nous proposons ici, une telle analyse est automatisable. Elle découle notamment d'une partie de l'étude réalisée par les algorithmes présentés dans la suite de ce chapitre. Les deux relations restantes (*RefinedBy*, *TracedTo*) représentent des dépendances faibles au cœur du modèle et correspondent à une aide à la navigation du modèle d'exigences. Ces relations identifient les parties du modèle proposant une illustration ou des précisions sur les exigences désignées. L'analyse des exigences peut être poussée jusqu'au recensement des activités de suivi des moyens de vérification dénotés par la relation *VerifiedBy*, il est alors intéressant de considérer que les tests associés peuvent être un moyen de détection des MdD. Les tests rendent observables certains éléments du système et fournissent ainsi des leviers de détection à considérer au cours de l'AMDEC, à condition que ces moyens soient utilisables en ligne.

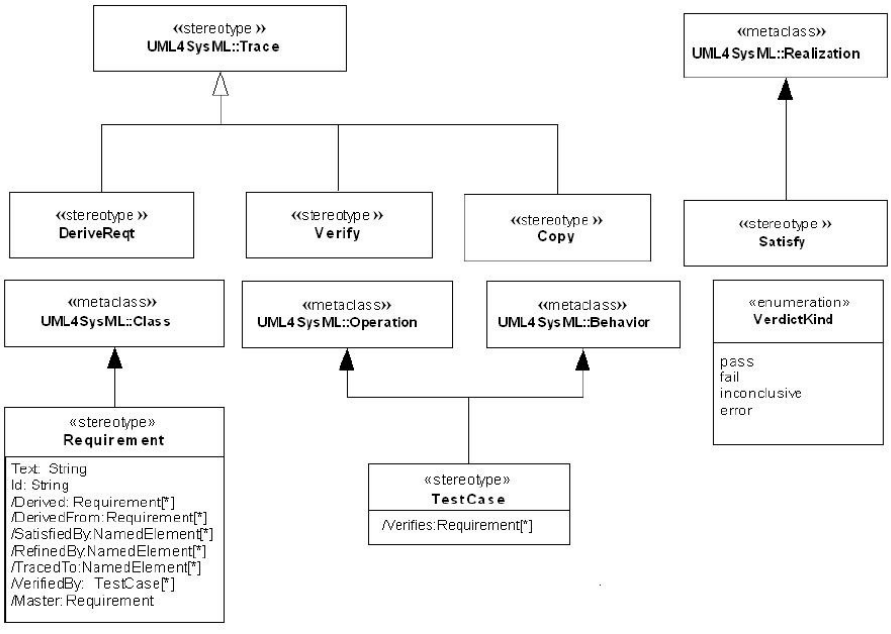


Figure II.17 *Syntaxe du stéréotype* Requirement

Règle N°6 : Les exigences relatives aux composants sont identifiées en considérant les relations *satisfy* entre les *parts* et les *requirements*, entre les *typing blocks* ou *owning blocks* et les *requirements*, et entre les *properties* des *parts* ou *typing blocks* et les *requirements*. Les *requirements* dérivés (*derived*) ou possédant un lien de *containment* avec les *requirements* satisfaits par le *part* sont également liés au composant.

II.3.4.2 Identifier les impacts sur les contraintes et valeurs d'influence

Les modèles SysML offrent une grande traçabilité des indicateurs de performance du système, ainsi que des caractéristiques physiques ou comportementales. Les diagrammes paramétriques modélisent les relations entre les caractéristiques des composants à travers leurs connexions aux *constraint properties*. Ils permettent d'exprimer comment les composants du système participent aux performances du système. Dans le cas où les diagrammes paramétriques du projet sont employés pour déclarer les performances exigées du système, il devient très utile d'exploiter les liens existants entre les composants et ces indicateurs. Les *value properties* des composants sont connectées aux contraintes par des « binding connectors » dénotant une relation d'égalité entre les deux membres de la connexion. Il est donc intéressant une fois les *value properties* d'un composant identifiées, de vérifier à quelles contraintes ces dernières participent. L'AMDEC bénéficie ainsi de la structuration offerte par SysML pour accéder rapidement aux informations nécessaires à la qualification des risques. La navigation au sein du modèle SysML peut être orientée pour porter l'attention des experts sur les points d'intérêt. L'analyse des diagrammes paramétriques pendant l'AMDEC revêt un triple intérêt : les effets de la dégradation du composant sur les performances et les caractéristiques du système peuvent être identifiés, qualifiés et quantifiés. A partir de la dégradation des *value properties* induites par les MdD, on sera capable de calculer les nouvelles valeurs prises par variables des composants liés et ainsi de qualifier et quantifier l'effet de la défaillance. Le traitement numérique des contraintes est extérieur à SysML et doit être confié à des outils périphériques exploitant le modèle SysML, comme mentionné au paragraphe I.2.5.3.

Règle N°7 : Les contraintes en relation avec les composants sont identifiées en considérant les *constraint properties* dont les *constraint parameters* sont liés aux *values properties* des *parts*. Ce sont en particulier toutes les *constraint properties* du *typing block* des *parts*.

Nous proposons des règles de modélisation afin d'exploiter les diagrammes paramétriques durant les AMDEC. Elles consistent à identifier chaque variable du composant étudié, participant à une contrainte, puis à exprimer les nouveaux ensembles de variation de ces variables au cours des MdD considérés. Les contraintes exprimées sont alors recalculées avec ces nouvelles valeurs. Les résultats expriment des changements dans les indicateurs de performances ou dans les caractéristiques des composants périphériques. Les impacts ainsi identifiés permettent de qualifier et quantifier le MdD à l'échelle du composant et à celle du système. L'identification des variables impliquées permet de guider la recherche de moyens de détection et d'actions correctives. L'expression poussée des impacts permet également de réaliser de façon précise les cotations des MdD. Le niveau de détail du modèle limite le déroulement de l'approche, dans de nombreux cas l'étude ne pourra aller jusqu'à la quantification qui né-

cessite l'affectation de valeurs numériques à chaque participant aux contraintes étudiées. Cependant, réussir à identifier les performances qui seront impactées par la dégradation des composants, est déjà un gain significatif dans l'analyse du système.

Lors de la saisie du modèle SysML, on identifie efficacement les contraintes sur les variables d'un composant, car ces dernières (*constraint properties*) sont liées au *block* définissant le type du composant par une relation de composition. Les *value properties* du *block* peuvent être liées aux *constraint parameters*, de même des variables appartenant à d'autres *blocks* peuvent être utilisées comme *references*. De plus, l'organisation de ces contraintes dans différents *packages* du projet permettra de distinguer l'objectif de représentation des différentes contraintes. Par exemple, certaines sont créées pour décrire un indicateur sur les performances du système, quand d'autres sont déclarées pour modéliser des relations physiques entre les variables des composants.

II.3.4.3 L'étude AMDEC par phase de vie ou phase de mission du produit

Pour de nombreuses applications comme le développement de systèmes aéronautiques [Picardi et al. 2004a], l'AMDEC est pertinente si elle prend en compte les phases de vie durant lesquelles le système est analysé. Ce besoin peut être aisément comblé à partir de l'étude d'un modèle SysML. En effet, la norme autorise le nommage ainsi que la multiplication des diagrammes afin d'obtenir diverses vues sur le système. Si la méthode d'IS retenue propose la création de diagrammes séparés pour les différentes phases de vie, il sera aisé de limiter ou focaliser l'étude AMDEC sur les diagrammes représentatifs des phases souhaitées.

Nous venons de voir dans ces deux derniers paragraphes que la profondeur des modèles SysML ainsi que leur grande habileté à lier et rendre cohérentes les différentes notions modélisées, permettait de conduire des AMDEC plus en adéquation avec une approche projet d'IS. Précédemment, nous avions analysé le métamodèle de SysML pour démontrer la pertinence de l'analyse de modèle SysML par une AMDEC. Nous avons montré que l'identification de l'architecture du système, des mécanismes de propagation à travers la structure et des impacts des défaillances peut être supportée de façon efficace par l'analyse de modèle SysML. Néanmoins, il demeure un point pivot de l'AMDEC que nous n'avons pas encore détaillé, à savoir la déclaration des MdD pouvant affecter les composants.

II.3.5 La recherche des Modes de Défaillance

Pour cette étape nous allons indiquer deux approches exploitant le langage SysML de façons bien distinctes. D'une part, nous étudierons le développement d'analyse d'expert sur la stricte analyse du modèle fonctionnel. Cette technique constitue une analyse du langage SysML, dans un cadre où le Retour d'Expérience (REX) n'est pas exploitable. D'autre part, nous évoquerons l'utilisation et l'organisation du REX dans une Base de données de Comportements Dysfonctionnels (BCD) décrite en SysML. Cette seconde proposition représente une proposition méthodologique pour l'utilisation de SysML dans le contexte de l'expression et l'organisation de MdD.

II.3.5.1 Aide à la formation d'avis d'experts

Si le REX n'est pas exploitable, la recherche des MdD se fait en réunissant les avis d'experts des différentes spécialités rencontrées dans la création du système. Cette recherche ne se fait généralement pas de façon empirique, mais s'appuie sur l'usage d'une systématique proposant par exemple de considérer si les MdD basiques des fonctions peuvent s'appliquer au composant et s'exprimer de façon précise. Nous proposons ici une approche différente, s'appuyant sur les éléments qu'un modèle SysML nous permet de retrouver pour chaque composant. L'AMDEC que nous souhaitons produire est, comme nous l'avons mentionné précédemment, orientée composant. Pour chaque composant du système, nous sommes en mesure de dégager les variables qui le caractérisent et qu'il manipule, ainsi que l'ensemble des fonctions qu'il exécute. Nous proposons d'utiliser dans un premier temps une systématique sur les variables des composants.

Pour chaque *part*, on identifie les *value properties*. Une défaillance du composant aura un impact sur au moins une de ces variables, qui regroupe aussi bien des descriptions de l'état du composant que les flux qu'il manipule. Pour chacune de ces variables, il convient de s'interroger sur la façon dont elle pourrait quitter son domaine nominal. Pour déterminer ce domaine les analystes peuvent notamment considérer le type de la variable tel qu'il est déclaré dans le modèle SysML. Pour une *value property* censée être constante, comme les variables d'état, on doit envisager les cas où la variation de la valeur est positive ou négative. Pour une variable telle que les variables de flux, on doit considérer les sorties du domaine de variation mais également la dynamique de la valeur :

- Signe de la dérivée du signal inattendue (positive, négative ou nulle).
- Valeur de la dérivée du signal inattendue (trop faible, trop grande).
- Forme de la dérivée inattendue.

Enfin, il convient de considérer les relations vérifiées par ces variables en examinant comment les contraintes liant les variables analysées pourraient être brisées.

Ensuite, on peut clore la recherche de MdD par l'application de la méthode présentée dans le cadre d'une AMDEC fonctionnelle réalisée sur les fonctions identifiées pour le *part* considéré (*opérations*, *activités*).

Cependant, l'utilisation d'un EDS intégrant les analyses de risque demande un investissement conséquent pour une structure. Un des bénéfices recherchés dans la mise en place de telles méthodes est de rendre pérennes les résultats d'études et de créer ainsi un référentiel réutilisable pour les analyses de risques. Il est donc préférable dans ce cadre de focaliser la gestion des risques sur l'exploitation et la création du REX. Nous présentons donc une méthode de recensement des MdD s'appuyant sur une BCD.

II.3.5.2 Organisation et utilisation du REX sous forme de BCD

Le REX doit intégrer l'ensemble des bases de données utilisées dans l'EDS et pourra ainsi servir pour l'assistance à la réalisation des AMDEC. Pour cela, les propriétés de SysML en tant que langage orienté objet sont très précieuses. En effet, la BCD va pouvoir s'appuyer sur le concept de *block* pour organiser et référencer son contenu. Ce type d'organisation est

conforme aux entrées utilisées dans les bases de données professionnelles relatives au recensement des MdD (Mil HDBK 217f, FIDES[18], OREDA[19]). Ce type de base de données est primordial pour une identification rapide des risques et une cohérence à travers les études. Pour la synthèse d'AMDEC préliminaires, il faut être capable d'accéder aux informations relatives à la SdF des composants étudiés. Nous avons choisi de définir une architecture générale pour la BCD en utilisant SysML. La cohérence obtenue avec les modèles analysés permet une intégration plus rapide des comportements dysfonctionnels à considérer lors de la modélisation du système global.
Cette partie de l'étude n'est donc plus une analyse du langage SysML, mais son utilisation pour la création d'une BCD. Elle possède donc une dimension méthodologique importante.

La BCD que nous développons n'est pas destinée uniquement à la réalisation des AMDEC, mais fait partie intégrante des répertoires utilisés dans l'EDS. Elle est utile pour toutes les opérations relatives à l'analyse, la qualification et la quantification de la SdF du système. Elle est donc constituée pour être exploitable à ces différentes fins. Dans ce manuscrit, sa présentation est scindée en deux parties, afin d'aborder les différentes vues nécessaires : la vue qualitative et la vue quantitative. Nous nous focalisons dans cette partie sur la présentation de l'intégration d'éléments qualitatifs nécessaires à la réalisation d'AMDEC. La partie relative à la pérennisation des éléments quantitatifs sera détaillée au chapitre III. Deux aspects sont à considérer dans la vue qualitative : ce sont d'une part, l'identification des MdD afférents à la technologie étudiée et d'autre part, la mise en évidence du changement de comportement induit par les défaillances. Ces deux éléments permettent respectivement de couvrir l'association entre composant et MdD et la modification des fonctions comme expression des effets locaux.

II.3.5.2.1 Métamodèle de la BCD : Vue Qualitative

Ces différents constituants sont présentés sur la vue « étude qualitative » du métamodèle de la BCD proposée sur la figure II.18.

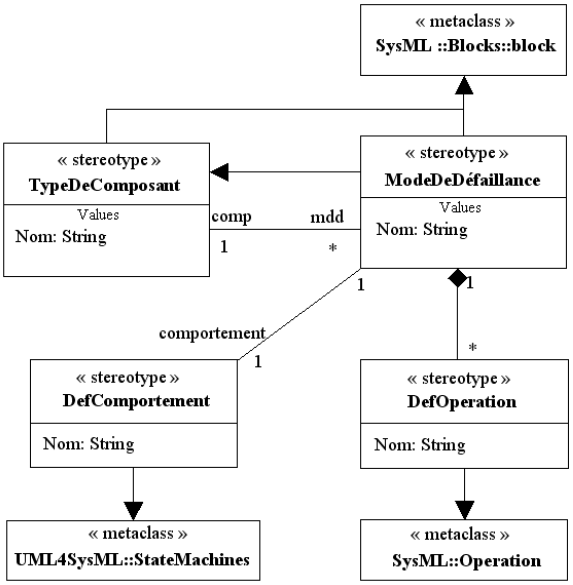


Figure II.18 *Vue « étude qualitative » du métamodèle de la BCD*

[18] FIDES : http://www.fides-reliability.org.
[19] OREDA : http://www.sintef.no/static/tl/projects/oreda.

A chaque type de composant sont associés les MdD connus. Chaque *block* définissant un type de composant est stéréotypé par « TypeDeComposant ». Ceux définissant un mode de défaillance utilisent le stéréotype « ModeDeDéfaillance ». Les champs Nom permettent de définir un type de composant ainsi que la dénomination des MdD. La partie déclaration des MdD recensés dans l'AMDEC est donc assurée par cette association de la BCD. Les MdD héritent du type de composant, de sorte qu'ils constituent une redéfinition des composants intégrant une partie dysfonctionnelle. Les propriétés du composant sont reprises, ainsi elles pourront être redéfinies ou réutilisées lors de la définition du comportement défaillant du composant. La définition du comportement dysfonctionnel est nécessaire pour exprimer les effets locaux des MdD déclarés. Nous proposons l'utilisation de deux types d'éléments venant s'associer : les *operations* et les *statemachines*. Les *operations* sont utilisées comme *actions d'état* dans la *statemachine*. Cette dernière permet de décrire le ou les changements d'état menant à la défaillance du composant. Les *actions d'état* définies en entrée (*entry : action*) transcrivent l'effet de la défaillance au moment du changement d'état. Quand les actions réalisées dans l'état défaillant (*do : action*) modélisent le traitement accompli par le composant dans cet état de défaillance.

La BCD est organisée de façon classique grâce aux *packages* SysML. Les types de composants peuvent être organisés par appartenance à un même projet ou à un même domaine (composants électroniques, hydrauliques …). Les différentes parties d'une telle BCD pourraient être importées dans différents projets. Le système de classement des membres de la BCD est associé au nom des *blocks* référencés comme type de composant. Pour créer un lien avec le modèle des concepteurs il est nécessaire que les taxonomies utilisées pour le nom des *blocks* se recouvrent. Une solution envisageable pour assurer ce point est de guider le nommage des entités lors de la saisie du modèle en proposant une bibliothèque de noms admis, voire de composants. La BCD est donc à mettre en correspondance en terme de nommage avec l'ensemble des bases constituant les répertoires d'éléments de l'environnement de développement système. La BCD reprend ces types de composants jusque dans la définition de leurs propriétés. Ces dernières peuvent en effet participer aux opérations décrivant le comportement du composant en état défaillant. A ce titre, un bon moyen de modéliser une *opération* est de lui allouer une *constraint property* instanciée dans un diagramme paramétrique, exprimant par une relation mathématique la transformation des entrées du composant en ses sorties.

II.3.5.2.2 Exemple de BCD relative à une vanne

Nous allons maintenant présenter la création de la BCD avec l'exemple d'une vanne. Cet exemple, nous permet d'illustrer des choix méthodologiques de modélisation pour la création de la BCD. Le point d'entrée dans la base est le nom du type de composant référencé. Dans ce cas, nous définissons un *block* de stéréotype « TypeDeComposant ». Pour notre exemple, ce *block* possède le même nom que le *block* définissant le concept de vanne dans les modèles fonctionnels. Il en possède également les propriétés (*values*, *opérations* …).
Nous incluons dans la BCD un *block* Vanne tel que représenté sur la figure II.19. Les *values* « inFlow » et « outFlow » représentent la valeur du flux en entrée et sortie de la vanne. La *value* « opened » est un booléen vrai si la vanne est ouverte et faux si elle est fermée. L'*operation* « TransmissionFluide » représente le passage du fluide entre « inFlow » et « outFlow ». L'*operation* « ActionVanne » présente la partie commande de la value « opened » de la vanne. Comme l'indique son cartouche, ce diagramme représente la partie du *package* BCD relative au concept de Vanne. On mentionne sur ce diagramme l'existence de 2

MdD connus pour ce type de composant, par la définition de 2 *blocks* stéréotypés « ModeDeDéfaillance ». Le nom de ces *blocks* indique le nom des MdD à considérer pour une AMDEC : « VanneBloquée » et « FuiteVanne ». Une relation de généralisation existe entre ces *blocks* et le *block* « Vanne », elle permet de déclarer un réemploi des caractéristiques fonctionnelles qui viendront être modifiées dans les situations de défaillance modélisées. Des propriétés telles que des *values* peuvent être rajoutées afin de caractériser le MdD. Par exemple, le *block* « FuiteVanne » possède une nouvelle *value* nommée « fuite », caractérisant l'importance de la défaillance. L'ensemble des *blocks* ainsi connectés représente l'ensemble des MdD connus pour ce type de composant, ils permettent de décrire la relation composant-MdD recherchée dans l'AMDEC. La suite des éléments de cette BCD permet d'atteindre les effets locaux des MdD.

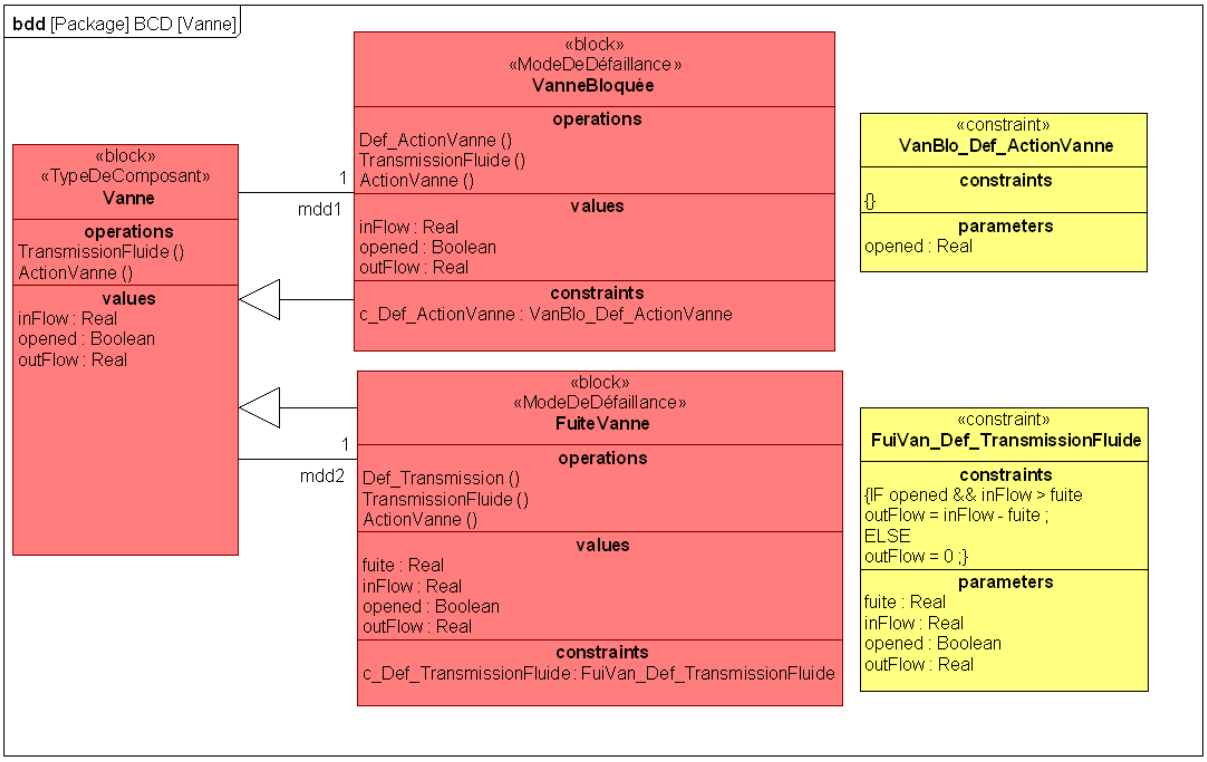


Figure II.19 *BCD des composants de type Vanne*

Chaque *block* « ModeDeDéfaillance » possède un pendant dysfonctionnel aux *operations* modifiées par la survenue du MdD. Nous avons choisi comme convention de nommage de préfixer la redéfinition dysfonctionnelle d'une *opération* par « Def_ ». Ainsi le *block* VanneBloquée possède une nouvelle *opération* : Def_ActionVanne, quand le *block* FuiteVanne reçoit la nouvelle *operation* : Def_TransmissionFluide. Bien que cela n'apparaisse pas sur le diagramme de la figure II.18, ces 2 *operations* ont été stéréotypées par « DefOperation ». Ceci permet en plus de l'usage de préfixe, d'identifier les opérations propres au dysfonctionnel. Pour spécifier le comportement de ces *operations* nous avons décidé de leur associer une *constraint property* détaillée dans un diagramme paramétrique. Le nommage de ces *constraint properties* reprend le nom des *operations* en les préfixant par « c_ ». Des *constraint blocks* sont définis pour typer ces *constraint properties*, leur nom reprend celui du *block* auquel appartient la *constraint property* et celui de l'*operation* illustrée, donnant pour l'*operation* Def_ActionVanne de VanneBloquée : VanBlo_Def_ActionVanne. La figure II.20 présente les diagrammes paramétriques qui instancient les deux *constraint properties* illustrant les *operations* dysfonctionnelles. Le cartouche des diagrammes spécifie le nom du MdD possédant

l'*operation* en qualité d'entité détenant la *constraint property*. La partie nom du diagramme permet de spécifier le nom de l'*operation* illustrée. Ces diagrammes illustrent notamment le nouveau comportement de transmission du flux lorsque la vanne fuit. Le diagramme paramétrique relatif au *block* FuiteVanne et à l'*operation* Def_TransmissionFluide montre la relation entre entrée, sortie, état et importance de la fuite. La valeur de sortie est celle de l'entrée diminuée de la valeur de la fuite lorsque la vanne est ouverte (la sortie est nulle si la fuite est trop importante).

Enfin, le comportement du type de composant dans les différents MdD est présenté grâce à une *statemachine* relative aux *blocks* « ModeDeDéfaillance ». La figure II.21 présente les *statemachines* réalisées pour les MdD des composants de type Vanne. Ces *statemachines* font appel aux *operations* déclarées. Nous avons pris le parti de représenter le passage du fonctionnement normal à la défaillance en réutilisant la *statemachine* fonctionnelle. La *transition* entre état fonctionnel et dysfonctionnel est volontairement laissée vierge. Sa caractérisation sera relative à la partie quantification des défaillances abordée au chapitre III. Ce type de déclaration est ici possible car le comportement fonctionnel est déclaré comme action d'état. Ces *statemachines* reprennent l'ensemble du comportement du composant détaillé, ceci permet de fournir directement la partie de modèle à importer pour tester l'intégration d'un composant défaillant dans l'architecture globale du système. Dans les exemples de la figure II.21, nous avons considéré des composants non réparables. La nomenclature des diagrammes permet d'identifier les MdD décrits. Si la vanne fuit, on constate que le composant exécute l'*operation* Def_Transmission décrite à la figure II.20, les changements d'états restent possibles. Quand la vanne est bloquée, elle réalise l'*operation* de transmission fonctionnelle, mais l'*operation* Def_ActionVanne vient remplacer ActionVanne, interdisant tout changement d'état.

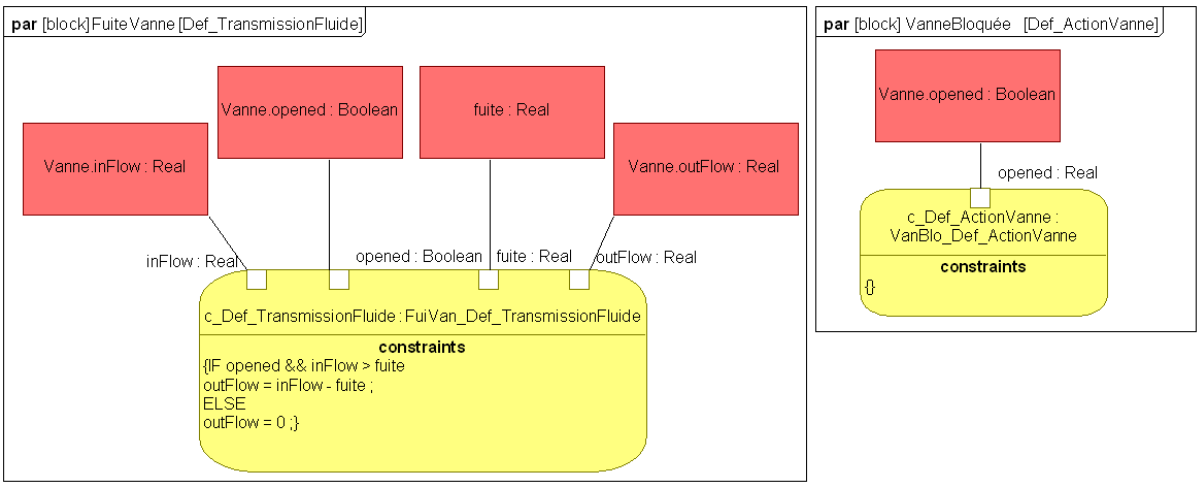


Figure II.20 *Diagrammes paramétriques des opérations dégradées*

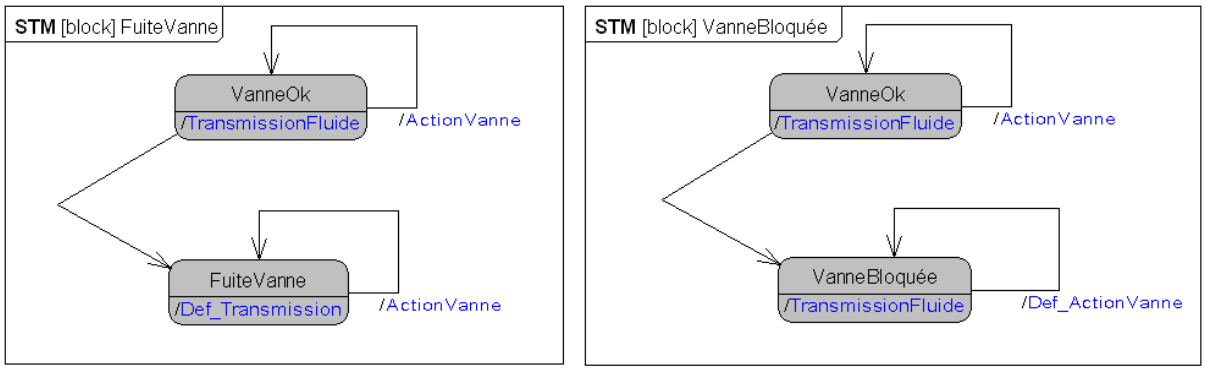


Figure II.21 *Statemachines des MdD*

II.3.5.2.3 Politique de gestion de la BCD

La politique de gestion de la BCD doit être stricte et bien formalisée pour venir s'intégrer aux différentes bases du projet et afin d'assurer une bonne qualité des informations contenues. Pour l'ingénieur exploitant la base et concevant le système, il convient d'éviter les doubles définitions et de respecter la taxonomie fixée. La politique de modification et création d'éléments de la BCD doit être réservée à l'expert SdF. Les ajouts et modifications de la base de MdD dépendent de la validation d'un ou plusieurs spécialistes et de la réutilisation de résultats d'études antérieures validés. Ainsi, la partie description des MdD par la réalisation des diagrammes exprimant les opérations ou les changements d'états doit être réalisée par les experts SdF. Néanmoins, ces derniers doivent intégrer les définitions fonctionnelles des types de composant réalisées par les concepteurs. Le suivi de version et la gestion des *packages* utilisés sont de la responsabilité du manager de projet ou du responsable de la branche SdF.

L'analyse des modèles SysML que nous réalisons, vise à assister la conduite d'AMDEC par la rédaction automatique d'un rapport préliminaire. L'AMDEC doit donc être ensuite finalisée et validée par les analystes SdF. De fait, cette AMDEC finale recèle des informations de qualité à réinjecter dans la BCD. Ceci est la première étape vers une mise en place systématique du REX. On peut ainsi ajouter les MdD nouvellement proposés ou affiner des comportements encore non référencés.

Dans cette section, nous venons de montrer que les modèles SysML contiennent les informations nécessaires à la réalisation d'AMDEC. Les métamodèles d'AMDEC proposés sont couverts par les éléments représentés dans le langage SysML. Nous avons également exposé que la réalisation d'AMDEC de modèles SysML peut être l'occasion d'intégrer encore mieux ce processus à la démarche globale d'ISBM réclamant notamment une traçabilité des exigences ainsi que la quantification d'indicateurs de performances. Nous avons enfin indiqué que le couplage AMDEC/SysML peut devenir bilatéral s'il est possible d'adjoindre à l'analyse, l'utilisation et la gestion d'une BCD. Nous avons à ce titre proposé une approche de modélisation de la BCD et l'avons illustrée par un exemple. L'utilisation d'un langage semi-formel comme SysML permet de réaliser une grande partie de l'analyse de façon automatique. En effet, le modèle créé au cœur de l'ISBM est produit grâce au support d'outils générant des fichiers informatiques décrivant le modèle. Dans cette partie, nous avons porté notre attention sur l'analyse au niveau métamodèle de l'AMDEC et du langage SysML afin de dessiner les contours d'une analyse de risques à partir d'un modèle SysML classique. Dans les paragraphes suivants nous exposons comment ces principes peuvent être mis en œuvre au sein d'un service d'analyse automatisé.

II.4 L'assistance à la création d'AMDEC par l'analyse automatisée de modèles SysML

Comme nous l'avons exposé au paragraphe I.3.2, l'AMDEC est une étape indispensable au processus d'analyse de SdF, bien que son caractère systématique impose naturellement une certaine lourdeur d'exploitation. La bibliographie a souligné la nécessité d'outiller ce processus et plusieurs tentatives d'automatisation de certaines de ses parties ont été relevées. Ce besoin devient encore plus impérieux lorsqu'il s'agit d'intégrer les analyses de type AMDEC au processus d'IS. L'automatisation et la réalisation d'AMDEC sur des modèles utilisant un langage spécifié sont des leviers importants pour la réussite de cette intégration. L'AMDEC pourra bénéficier des mêmes avantages que ceux relevés lors de l'utilisation de MBM, à savoir : meilleure qualité d'étude, meilleure communication entre équipes, meilleure productivité et meilleur transfert des connaissances. Cette approche se démarque des travaux existants, qui ne géraient automatiquement des phases de l'AMDEC que lorsqu'un modèle dysfonctionnel avait pu être bâti. En effet, par l'analyse de modèles purement fonctionnels exprimés en SysML, il devient possible d'automatiser des étapes comme la recherche de composants et de guider l'analyse de l'expert lors de la rédaction de l'AMDEC finale. L'objectif de l'assistance à la création d'AMDEC peut être résumé comme la volonté de consacrer 80% du temps de l'expert aux 20% de l'étude les plus délicats. On fait bien souvent le constat inverse pour les AMDEC classiques. Dans cette partie, nous présentons les algorithmes de création de listes d'entités en relation permettant d'établir l'AMDEC préliminaire. Nous montrons ensuite comment gérer une base de connaissances reflétant les données de l'AMDEC et permettant de gérer l'évolution des AMDEC au cours du design. Ensuite, nous évoquerons les choix technologiques pour l'implémentation de ces outils au sein d'un outil de modélisation SysML, ou comme solution indépendante. Nous relèverons les difficultés de ces différentes options et discuterons de l'impact de la méthodologie de saisie du modèle choisi.

II.4.1 Création et exploitation d'une base de connaissances d'exploitation des AMDEC

Il est important de noter que certaines des analyses proposées dans ce chapitre ne sont pas automatisables. En effet, les étapes de création d'information telles que l'appel au jugement d'experts ne peuvent être confiées à un outil, mais peuvent néanmoins être assistées, par exemple, par des mécanismes facilitant l'exploitation et la navigation du modèle. Nous proposons donc de réaliser un service d'analyse du modèle, portant sur l'identification de la structure du système et des relations inter-composants, comme nous l'avons proposé par les règles formulées au cours de ce chapitre. L'algorithme créé parcourt le modèle SysML pour dégager des listes d'entités en relation, relatives à chaque composant du système. Ces listes servent ensuite à créer un rapport AMDEC préliminaire montrant les résultats du recensement des composants et proposant les connexions et communications à considérer pour identifier les chemins de défaillance. Couplée à une base de type BCD, l'étude propose également un appel automatique au REX. Ce rapport préliminaire doit ensuite être validé et complété par les analystes de SdF. Les informations ainsi générées peuvent être gérées dans une base de connaissances AMDEC.

L'algorithme est centré sur l'étude des *parts* présents dans les modèles fonctionnels SysML. Dans ce paragraphe, nous ne faisons pas d'a priori sur le type de fichier support ni

sur la technique d'implémentation. Les *parts* sont analysés selon deux axes : leurs propriétés et leur contexte d'utilisation. Les propriétés reflètent les caractéristiques du composant et sont héritées directement de la définition du *block* donnant son type au *part*. Dans un modèle SysML complet le *part* est représenté dans différents contextes d'utilisation donnant des informations primordiales sur son utilisation et son implantation. Ce contexte est exprimé dans divers diagrammes :

- Les Internal Block Diagrams, exprimant l'implantation et les connexions physiques du composant.
- Les Parametric Diagrams, déclarant les relations vérifiées par les *value properties* du *part*.
- Les Sequence Diagrams, présentant les échanges de type commandes entre les *parts*.
- Les Activity Diagram (à condition que les *actions* soient allouées), présentant les enchaînements de traitements et l'échange d'information, énergie ou matière, entre *parts*.

Pour chaque composant du système, nous cherchons à constituer une structure de données relative au *part* le modélisant. Cette structure cible les points suivants :

1. Le nom ou identifiant, permettant la traçabilité du composant à travers le processus d'IS.
2. Le type, donnant accès à des bases de connaissances sur le composant (base de comportement, base de MdD, constructeurs …) (*typing block*).
3. Le groupe de composant auquel participe le composant (*owning block*), parfois appelé équipement.
4. Les exigences (*Requirements*), montrant la participation du composant à la tenue des objectifs.
5. Les contraintes, indiquant les relations logiques ou algébriques respectées par le composant (i.e. ses caractéristiques).
6. Le voisinage physique, identifiant les composants connectés en amont et en aval du composant et la spécification des interfaces et de la nature des flux échangés.
7. Le voisinage fonctionnel, relevant les composants communicants avec le composant et spécifiant la nature des échanges.

Ce travail est réalisé sous l'hypothèse que les modèles SysML analysés ne suivent pas de méthode d'IS particulière. Ceci a pour conséquence de compliquer la recherche des éléments à inscrire dans la structure, mais permet également d'appliquer l'algorithme au plus large ensemble de modèles possible. Cette étude met également en évidence les points de modélisation nécessitant une méthodologie de saisie afin de produire un modèle optimal pour l'analyse de type AMDEC. Ceci nous permet de formuler quelques recommandations de modélisation afin d'optimiser la réalisation de la structure recherchée. L'algorithme réalise l'analyse en reprenant les éléments de métamodèles mis en évidence dans les paragraphes précédents de ce chapitre et en suivant les règles énoncées au cours du paragraphe II.3.

II.4.1.1 Recherche des composants

Les composants sont les *parts* de la partie structurelle physique du modèle (Règle N°1). Une structure est créée pour chaque *part* du modèle. L'identification des composants est l'occasion d'enregistrer différentes caractéristiques importantes pour la suite de l'analyse et la création de la structure. Le nom du part est le premier champ consigné accompagné du type du composant. L'obtention du type permet d'accéder aux caractéristiques du *part* que sont ses *value properties* et *port properties* (pour simplifier l'exposé de l'algorithme nous regroupons *standard* et *flow ports*), ces dernières serviront dans la suite de l'analyse. Il est nécessaire d'identifier le *block* auquel appartient le *part* étudié, nous le nommerons *owning block*. D'un point de vue algorithmique il faut trouver le *block* **b** tel que le *part* étudié soit un *part property* de **b**. A ce stade, nous avons enregistré pour le *part* étudié : son nom, son type (*typing block*), son *owning block* ainsi que ses *value properties* et *port properties*. La navigation à travers les relations entre ces entités suit le métamodèle de la figure 2.7. Nous obtenons ainsi les trois premiers éléments de la structure : le nom, le type et le groupe d'appartenance. Nous inscrivons dans la structure le champ *name* du *part*, le champ *type* (champ *name* du *typing block*) et le champ *name* du *owning block*.

Remarque : Les *parts* sont identifiés soit dans le répertoire du modèle[20], soit par leurs représentations graphiques. On peut notamment les voir apparaître comme propriétés des *blocks* au sein des BDD, ou en tant qu'élément graphique du modèle dans les IBD. Ce choix dépend du type de fichier employé pour effectuer l'analyse.

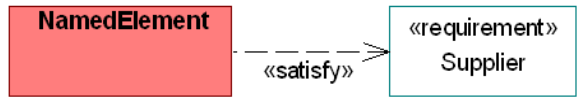
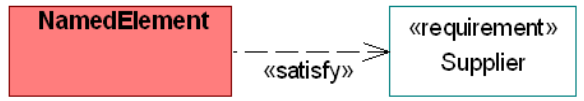
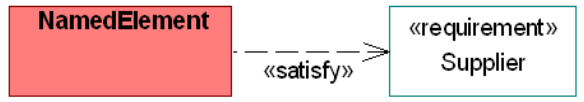
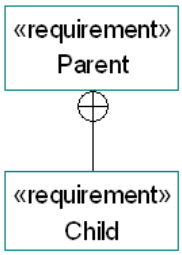
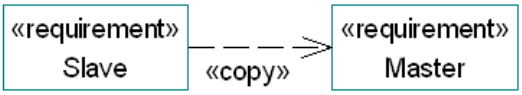
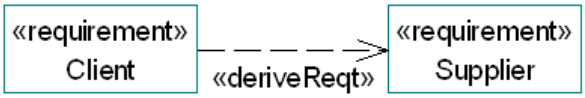
II.4.1.2 Recherche des exigences portées par le composant

Cette étape de l'algorithme correspond à l'exploitation des liens entre composants et exigences présentés au paragraphe II.3.4.1, relatif au métamodèle des *requirements*. Pour chaque *part* étudié on recherche les exigences potentiellement impactées par ses défaillances. Les éléments que l'on souhaite inscrire dans la structure de chaque *part* sont les noms des exigences qui lui sont liées, ainsi qu'un degré de dérivation indiquant la causalité entre exigences. La partie exigence de la structure sera donc constituée de couples nom d'exigence/degré de dérivation. La recherche s'effectue comme suit : on recherche les exigences telles que le *part* ou son *typing block* leur soient liés par une relation « *satisfy* ». On note le champ *name* de chacune d'elles et on leur affecte le degré de dérivation 1. Ainsi on identifie les exigences unies par une relation directe au *par*t analysé. Ensuite, on recherche les exigences identiquement liées à l'*owning block*, ces dernières se voient affectées le degré de dérivation 2. Ce degré indique que le composant n'est pas directement responsable des exigences, mais pourrait contribuer à sa satisfaction en tant que participant à l'équipement. En effet, une exigence liée à l'*owning block* peut concerner les caractéristiques ou le comportement des composants du même groupe, mais indépendant du *part* étudié. La suite de cette recherche est constituée de l'étude des liens entre exigences, afin de retrouver les exigences auxquelles participent les exigences affectées au *part*, ainsi que celles qui leur sont dérivées. Pour cela on étudie trois types de relation : « containment », « copy » et « derived from». On recherche ces liens pour chaque exigence liée au *part* et on enregistre chaque nouvelle exigence avec son degré de dérivation. Le degré de dérivation s'incrémente de 1 pour chaque relation employée. Le traitement des liens se fait en étudiant les exigences degré par degré. Si une exigence est retrou-

[20] Le répertoire centralise les déclarations des artefacts de modélisation.

vée deux fois durant cette recherche, seule l'occurrence de plus bas degré de dérivation est conservée. L'opération se fait suivant les indications du tableau II.1.

Tableau II.1 *Traitement des liens entre« requirements »*

Relation	Elément analysé (N:Degré de dérivation)	Requirement identifié	Degré de dérivation conféré
	Part	Supplier	1
	Typing block	Supplier	1
	Owning block	Supplier	2
	Child (N)	Parent	N + 1
	Master (N)	Slave	N + 1
	Client (N)	Supplier	N + 1

Remarque 1 : On peut choisir de relever également la relation « verify » par laquelle on lie un test à une exigence. Le test pourrait alors être enregistré avec l'exigence dans la structure.
Remarque 2 : L'étude est réalisée également en consultant les tables d'exigences représentant les mêmes types de relation, mais de façon tabulaire.

II.4.1.3 Recherche des éléments de contrainte sur les caractéristiques

La recherche de contraintes se fait par l'analyse des propriétés du *part* mais également par l'analyse de son contexte (§II.3.4.2). Dans un premier temps, on identifie les contraintes étant des propriétés du *typing block* du composant. Leur identification est directe. On note dans la cinquième partie de la structure le nom et le type des contraintes attachées au *part*. Dans un second temps, on recherche si le *part* étudié participe à des contraintes de son *owning block*. Pour cela il faut considérer les *constraint properties* de l'*owning block* et vérifier si leur contexte d'emploi fait ou non appel aux caractéristiques du *part*. On étudie alors les parametric diagrams exprimant le contexte de déploiement des *constraint properties*. Pour chacune d'entre elles, on étudie si un de leur paramètre (*constraint parameter*) est lié à une caractéristique du *part* (*value property* du *part*, applelée sous la forme PartName.ValueName:Type). Le cas échéant la contrainte est ajoutée à la structure de la même façon que les contraintes du *typing block*. La figure II.22 donne un exemple de modèle à analyser. Dans le cas proposé et

en application de la procédure ici décrite, on considère comme contraintes pour le part « p » : c1 : Ctype et c3 : Ctype. « c1 » est repérée comme *constraint property* du *typing block*. « c3 » est identifiée comme *constraint property* dont un *constraint parameter* est connecté à une *value property* du *part* étudié.

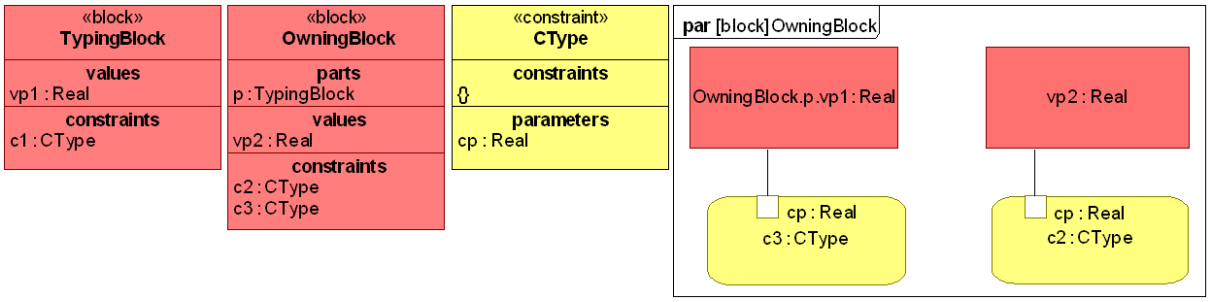


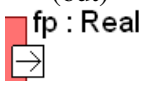
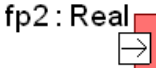
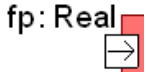
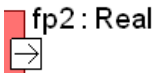
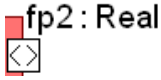
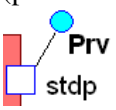
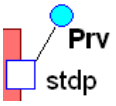
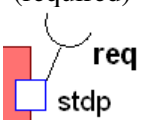
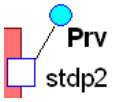
Figure II.22 *Exemple de contraintes*

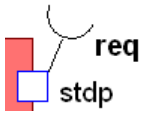
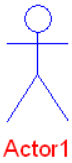
II.4.1.4 Recherche du voisinage structurel

Le traitement se poursuit par la recherche des composants connectés physiquement au composant étudié. Cette partie de l'algorithme porte sur l'étude du contexte du *part*. Pour exprimer ces contraintes structurelles, le concepteur utilise les IBD. Les liens physiques sont exprimés par les connecteurs (Figure 2.13), la 6ème partie de la structure se détermine par l'analyse, donnée par la Règle N°3, des relations proposées au II.3.3.3. Pour chaque composant en relation, on cherche à noter : le nom et le type du composant, la paire de ports utilisée et leur type, le nom et le type du flux traversant la connexion. Les composants sont organisés en deux sous-listes : les composants connectés en amont et les composants connectés en aval.

Les propriétés du *part* permettent de lancer la recherche. Un composant est connecté à son environnement par des ports. Ces ports sont des propriétés du *part* héritées du *typing block*. Le ou les IBD à étudier sont détenus par l'*owning block* et sont nommés selon la nomenclature suivante : ibd [Block] nom.owningBlock [nom diagramme]. Le tableau II.2 présente les cas rencontrés lors de cette recherche. On considère chaque port du *part* et on recherche l'autre extrémité du *connecteur* qu'ils reçoivent. Pour les *standard ports* on fait la recherche pour chaque *interface* du port. Suivant la direction du port ou le type d'*interface*, l'élément connecté sera ajouté à la liste des entités « amont » ou « aval » du composant. On enregistre dans ces sous-listes le maximum de connaissances sur la connexion identifiée. D'abord, on note le composant connecté et son type, puis les informations sur l'interface réalisant la connexion : le nom et le type des ports utilisés puis s'il est disponible le nom et le type du flux échangé (l'*item flow*). La première colonne du tableau II.2 est le port possédé par le *part* étudié, la seconde est l'entité connectée à l'extrémité du connecteur entrant sur le *part*. La troisième colonne présente le type d'entité possédant la seconde extrémité du connecteur. La quatrième donne le nom de l'item flow porté par le connecteur étudié. La dernière colonne montre un exemple de notation de l'information recueillie, qui sera inscrite dans la sous liste désignée par la cinquième colonne. Par exemple, la première ligne illustre le cas où le *part* étudié possède un *Flow port* (fp) de type « Real » et *direction* « out », connecté à un *Flow port* (fp2) de type « Real » et *direction* « in », ce port est possédé par un *part* (p2) de *Typing block* « Type ». La connexion porte l'*item Flow* (It) de type « ItType ». Cette configuration entaine l'ajout de la description de la connexion dans la sous-liste « Aval ».

Tableau II.2 *Recherche du voisinage structurel*

Type de port (Direction)	*Connector End*	Propriétaire du *Connector End*	*Item Flow* du connecteur	Sous-liste modifiée	Elément ajouté
Flow Port (out) 	*Flow Port* (in) 	*Part* p2 : Type *OwningBlock*	It : ItType	Aval	p2 : Type [fp : Real – fp2 : Real] (It : ItType)
			∅	Aval	p2 : Type [fp : Real – fp2 : Real]
			It : ItType	Aval	*OwningBlock* [fp : Real – fp2 : Real] (It : ItType)
			∅	Aval	*OwningBlock* [fp : Real – fp2 : Real]
Flow Port (in) 	*Flow Port* (out) 	*Part* p2 : Type *OwningBlock*	It : ItType	Amont	p2 : Type [fp : Real – fp2 : Real] (It : ItType)
			∅	Amont	p2 : Type [fp : Real – fp2 : Real]
			It : ItType	Amont	*OwningBlock* [fp : Real – fp2 : Real] (It : ItType)
			∅	Amont	*OwningBlock* [fp : Real – fp2 : Real]
Flow Port (inout) 	*Flow Port* (inout) 	*Part* p2 : Type *OwningBlock*	It : ItType	Aval & Amont	p2 : Type [fp : Real – fp2 : Real] (It : ItType)
			∅	Aval & Amont	p2 : Type [fp : Real – fp2 : Real]
			It : ItType	Aval & Amont	*OwningBlock* [fp : Real – fp2 : Real] (It : ItType)
			∅	Aval & Amont	*OwningBlock* [fp : Real – fp2 : Real]
Standard Port (provided) 	*Standard Port* (required) 	*Part* p2 : Type *OwningBlock*	It : ItType	Aval	p2 : Type [stdp : Prv – stdp2 : req] (It : ItType)
			∅	Aval	p2 : Type [stdp : Prv – stdp2 : req]
			It : ItType	Aval	*OwningBlock* [stdp : Prv – stdp2 : req] (It : ItType)
			∅	Aval	*OwningBlock* [stdp : Prv – stdp2 : req]
Standard Port (provided) 	*Actor* 	Actor1	It : ItType	Aval	Actor1 [stdp : Prv – Actor1] (It : ItType)
			∅	Aval	Actor1 [stdp : Prv – Actor1]
Standard Port (required) 	*Standard Port* (provided) 	Part p2 : Type *OwningBlock*	It : ItType	Amont	p2 : Type [stdp : req – stdp2 : Prv] (It : ItType)
			∅	Amont	p2 : Type [stdp : req – stdp2 : Prv]
			It : ItType	Amont	*OwningBlock* [stdp : req – stdp2 : Prv] (It : ItType)
			∅	Amont	*OwningBlock* [stdp : req – stdp2 : Prv]

Standard Port (required) 	*Actor* 	Actor1	It : ItType	Amont	Actor1 [stdp : req – Actor1] (It : ItType)
			∅	Amont	Actor1 [stdp : req – Actor1]

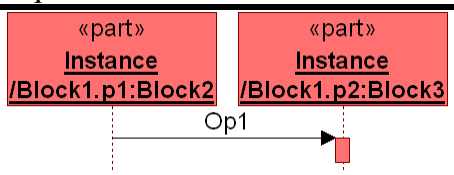
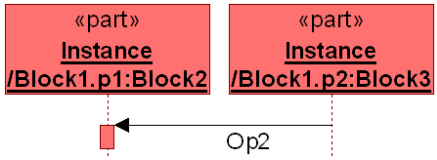
Remarque : Le déroulement de cet algorithme peut être l'occasion de contrôler des éléments de cohérence du modèle analysé. On peut par exemple vérifier la cohérence des directions des *flow ports* connectés entre eux, ainsi que la direction de l'*item flow* porté par la connexion.

II.4.1.5 Recherche du voisinage fonctionnel

On recherche enfin les composants communicant avec le *part* étudié ainsi que la chaîne d'*actions* à laquelle il participe, en application des Règles N°2 et N°4. On entend par voisinage fonctionnel l'ensemble des composants, *operations* et *actions* venant interagir avec le composant étudié au sein du comportement du système. Pour cela, il faut étudier deux types de diagrammes modélisant les comportements auxquels participe le composant. On construit 4 sous-listes pour le 7ème élément de la structure : 2 relatives aux Diagrammes de Séquences (SD) et 2 relatives aux Diagrammes d'Activité (AD).

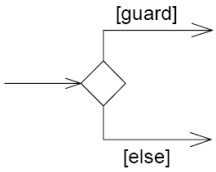
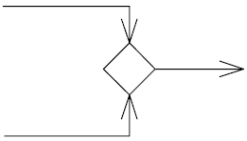
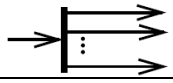
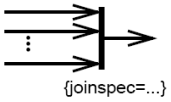
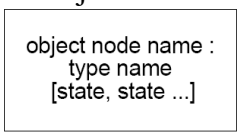
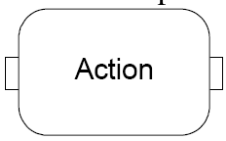
Lors de l'analyse des SD, on identifie les *parts* connectés au composant étudié, leur type et l'*operation* portée par le *message* échangé. On analyse ces diagrammes en suivant le métamodèle de la figure II.15. L'ordonnancement des envois de *messages* n'a pas à être considéré ici, car nous souhaitons simplement relever qu'un lien existe entre le composant étudié et certains autres *parts*. C'est pourquoi l'algorithme se limite à identifier la ou les *lifelines* associées au *part* et à considérer l'ensemble des *messages* liés à ces *lifelines*. On remonte à chaque fois à l'autre extrémité du *message* (*message end*) et on relève le nom du *part* lié à la seconde *lifeline*. On relève également le nom et le prototype de l'*opération* portée par le *message*. Suivant l'extrémité du *message* occupée par le *part* étudié (*sender* ou *receiver*), on inscrit les informations dans la sous-liste « aval » ou « amont » des listes dédiées à l'étude des SD. On peut choisir également de relever le nom de l'interaction à laquelle les messages participent. Nous proposons le type de notation suivante dans la structure : nom.part [nom.operation] (nom.interaction), afin de pouvoir se référer facilement au modèle. Le tableau II.3 répertorie les deux cas de figure où le *part* (sa *lifeline* associée) est *sender* ou *receiver* du *message*. Nous illustrons dans la dernière colonne l'élément inscrit dans les sous-listes pour chaque éventualité.

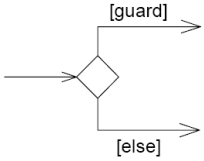
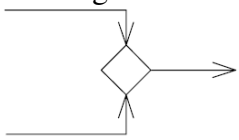
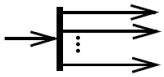
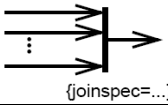
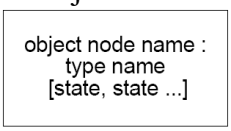
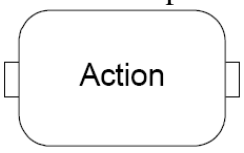
Tableau II.3 *Traitement des Diagrammes de Séquences (élément analysé : p1)*

Séquence	*Sender/Receiver*	Sous-liste modifiée	Elément ajouté
	p1/p2	Aval	p2.Block3 [Op1]
	p2/p1	Amont	p2.Block3 [Op2]

Comme nous l'avons mentionné au paragraphe II.3.3.3, la recherche de composants n'est réalisable sur les AD que si les *actions* ont préalablement été allouées aux *parts* ou *blocks* du modèle. Nous considérons donc pour cette étude que les *actions* sont allouées. Pour chaque *action* allouée au *part* ou à son *typing block*, on effectue le traitement présenté au tableau II.4, qui exploite le métamodèle présenté figure II.16. L'analyse porte sur les connexions (*activityEdges*) qui entrent ou sortent des *actions* allouées au *part*. Ces *acticityEdges* sont « fixées » soit directement à l'*action* soit à un de ses *pins*. Les *activityEdges* sortantes sont les *outgoing*, les entrantes sont les *incoming*. En prenant pour point de départ l'*action* réalisée par le *part* étudié, on recherche les *actions* réalisées juste à la suite de cette dernière. Pour cela, on parcourt les *activityEdges* qui forment le modèle. Suivant le type de nœud rencontré, la recherche peut se dupliquer ou suivre un simple chemin. Dès que l'on aboutit à une *action*, on note l'élément de structure auquel elle est allouée. En fonction du sens de la recherche l'élément identifié est noté, avec l'action qu'il réalise, dans la sous-liste « amont » ou « aval ». Dans le cas d'un *fork node* ou d'un *decision node*, on duplique la recherche et on note l'action trouvée pour chaque route. Si des gardes sont franchis, il est intéressant de les noter pour considérer les conditions de propagation selon cet axe. Au cours de la recherche, on peut être amené à rencontrer des *object nodes*. Ils représentent les flux, objets ou informations échangés entre *actions*. Ces objets et leur type sont enregistrés comme membres du chemin identifié et ajoutés dans les listes. On note également, les *pins* empruntés et leur type pour une parfaite connaissance de la connexion identifiée. La première partie du tableau présente la recherche des *actions* se déroulant avant celles réalisées par le *part* étudié. Pour cela, on recherche les *actions* sources des *incoming activity edges* des *actions* effectuées par le part. La seconde partie du tableau II.4 exprime la recherche des *actions* effectuées après l'intervention du *part*, en analysant les *action*s connectées à ses *outcoming activity edges*.

Tableau II.4 *Traitement des activityEdges connectées aux actions des parts étudiés*

Type de l'*activityEdge* vue du *Node* étudié	*ActivityEdge* Source/Target ❶/❷	Traitement
Incoming ❶	Initial Node	Ne rien faire
	Decision Node 	Enregistrer *guard* (!*guard* si else). Considérer le *node* courant. Aller en ❶.
	Merge Node 	Considérer le *node* courant. ∀ *incoming activityEdge*, aller en ❶.
	Fork Node 	Considérer le *node* courant. Aller en ❶.
	Join Node 	Considérer le *node* courant. ∀ *incoming activityEdge*, aller en ❶.
	Object Node 	Enregistrer *node* name : type name. Considérer le *node* courant. ∀ *incoming activityEdge*, aller en ❶.
	Action & pin 	Enregistrer *pin* name : type name. Enregistrer *action* name. Rechercher « allocated to » *block* ou *part*. Inscrire dans liste « amont » : *part* ou *block* name [*Action* name] (*pin* name et type, *guards*, *object node* name et type,)

Outgoing ❷	Flow Final 	Ne rien faire
	Activity Final 	Ne rien faire
	Decision Node 	Considérer le *node* courant. ∀ *outgoing activityEdge*, enregistrer *guard* ou !*guard* aller en ❷.
	Merge Node 	Considérer le *node* courant. Aller en ❷.
	Fork Node 	Considérer le *node* courant. ∀ *outgoing activityEdge*, aller en ❷.
	Join Node 	Considérer le *node* courant. Aller en ❷.
	Object Node 	Enregistrer *node* name : type name. Considérer le *node* courant. ∀ *outgoing activityEdge*, aller en ❷.
	Action & pin 	Enregistrer *pin* name : type name. Enregistrer *action* name. Rechercher « allocated to » *block* ou *part*. Inscrire dans liste « aval » : *part* ou *block* name [*Action* name] (*pin* name et type, *guards*, *object node* name et type,)

II.4.1.6 Exploitation de la structure pour la création de l'AMDEC préliminaire

Toutes les informations ainsi rassemblées dans la structure permettent de créer un rapport AMDEC préliminaire qui sera utilisé pour établir l'AMDEC finale du système. Ce premier rapport constitue un support efficace pour les analystes SdF. Pour chaque composant du système à étudier, les éléments pertinents du modèle à considérer seront proposés. Le tableau AMDEC sera alimenté par la base de données constituée de l'ensemble des structures créées pour les composants et par une base similaire à la BCD proposée en II.3.5.2.2. De plus, les modifications apportées par les experts à l'AMDEC viendront faire évoluer les éléments de ces bases. L'AMDEC pourra alors être vue comme une interface de gestion de ces différentes bases. Le tableau créé reprend la trame proposée figure II.23.

Composants	MdD	Causes	Effets Locaux	Effets Système	Effets Exigences	Gravité	Occurrence	Détectabilité	Criticité	Moyens de détection	Moyens de correction

Figure II.23 *Trame de l'AMDEC préliminaire*

Dans un premier temps, les structures créées par l'algorithme sont utilisées pour préremplir ce tableau. Pour chaque composant, on dispose d'une structure de données qui seront être réparties dans la table AMDEC. On inscrit d'abord le nom du *part* et le nom du *typing block* dans le champ composant. Ceci permet de réaliser la cohérence des dénominations au cours de l'analyse. On peut éventuellement indiquer le nom de l'*owning block* déterminant le type de sous-système auquel participe le composant. On note ensuite les MdD dont pourrait être victime le composant. Pour cela on exploite la BCD disponible, le nom du *typing block* sert à accéder aux connaissances sur le type de composant. Si aucun élément de la base ne correspond, on peut soit laisser le champ MdD vide, soit proposer les MdD standards prescrits dans les documents normatifs de l'AMDEC (vus au § I.3.2.1). Ensuite, on propose pour chaque MdD les éléments de modèles permettant de donner ou rechercher les causes. Il faut alors indiquer les connexions structurelles ou fonctionnelles arrivant en amont du composant. Il s'agit des composants et connaissances sur leurs connexions, identifiés dans les IBD, SD et AD que nous avons obtenus aux paragraphes II.4.1.4 et II.4.1.5.

La colonne « effets locaux » est constituée des contraintes identifiées pour le *typing block* (§II.4.1.3) et des contraintes définies dans la BCD pour le MdD analysé. Ces contraintes expriment les relations entre caractéristiques qui seront potentiellement altérées par la défaillance. En particulier, la contrainte de la BCD, telle que modélisée dans notre exemple (§II.3.5.2.2), exprime la modification de l'exécution de la fonction du composant. L'analyste disposera ici des éléments de calcul nécessaires à l'expression des effets sur le composant. Pour déterminer les effets système, on considère les connexions structurelles et fonctionnelles en aval du composant. Ainsi de façon similaire à la colonne cause, on emploie les relations révélées par les IBD, SD et AD. L'identification des *requirements* (§II.4.1.2) permet de spécifier une colonne dédiée au suivi d'exigences dans laquelle on indique les exigences potentiel-

lement menacées par la défaillance, ainsi que leur degré de dérivation afin de fournir des priorités d'étude.

On obtient ainsi un tableau AMDEC pré-rempli donnant des éléments de réponses et de raisonnement pour l'établissement de l'AMDEC finale. Dans ce tableau, sont automatiquement proposés :

- Le nom de tous les composants du système ainsi que leur type.
- L'ensemble des MdD connus pour ce type de composant.
- L'ensemble des composants périphériques pouvant causer la défaillance.
- L'ensemble des composants périphériques pouvant subir la défaillance.
- La nature et l'objet de la connexion entre les composants périphériques et le composant étudié.
- Les effets locaux connus et les relations internes au composant, susceptibles d'être altérées par la défaillance.
- L'ensemble des exigences dont le respect est menacé par les défaillances.
- La désignation précise de chacun de ces éléments dans le modèle SysML, afin de pouvoir remonter facilement à la source de l'information.

Ces informations sont la base nécessaire pour débuter l'AMDEC et ainsi enclencher la première phase de la méthode MéDISIS. Cet algorithme est implémenté dans un outil qui constitue le premier outil support de la méthode. Cette AMDEC préliminaire est obtenue dès qu'un premier modèle fonctionnel est disponible. Le lancement de la méthode est ainsi extrêmement rapide, ce qui constitue un pas primordial pour l'intégration des analyses SdF aux méthodes d'IS basées sur les modèles. MéDISIS se poursuit par la réalisation de l'AMDEC finale et l'intégration de ses résultats au processus d'IS suivi.

II.4.1.7 La réalisation de l'AMDEC finale et le suivi de modifications

Pour être intégrée au processus IS, l'AMDEC doit d'une part, s'appuyer sur le modèle créé (ce qui est réalisé en instrumentant les algorithmes présentés), et d'autre part proposer ses résultats sous une forme intégrable à l'ensemble des informations créées. L'AMDEC doit également s'adapter à un processus itératif conduisant à des évolutions de design. A ce titre, il faut être capable de faire évoluer l'AMDEC, sans devoir refaire à nouveau toute l'étude lors de chaque changement dans le modèle de conception, comme cela a été souligné dans [Price 1996]. Pour cela, nous préconisons d'utiliser le tableau de l'AMDEC comme une interface de gestion d'une base de connaissances rassemblant les informations créées par l'AMDEC. Ainsi, la réalisation de l'AMDEC nécessite l'emploi de 3 bases de connaissances : une BCD, un ensemble de structures d'analyse du modèle SysML (créé en utilisant les algorithmes que nous avons décrits) et une base AMDEC.

La base AMDEC contient pour chaque composant les informations portées par les lignes d'AMDEC. Cette base est remplie et modifiée par l'analyste SdF à travers l'utilisation du tableau. Dans la base AMDEC, chaque composant reçoit comme champs ses MdD, leurs causes, effets, cotations, moyens de détection et actions menées. Chaque élément de la structure relative à un composant du modèle SysML est associé à un champ de la base AMDEC. Par exemple une liaison amont est associée à un élément cause de la base AMDEC. Si au cours de l'étude la cause est détaillée ou reformulée, l'élément de la structure qui lui est associé reste

conservé, bien que pas nécessairement affiché dans le tableau. Ce mécanisme permettra de gérer les évolutions du modèle et les changements à répercuter dans l'AMDEC. On affecte également un statut à chaque élément de la base AMDEC. Ce statut indique si l'élément a été supprimé lors d'une analyse précédente, s'il s'agit d'un nouvel élément ou si l'élément est conservé. Ceci permet de gérer les évolutions du modèle ou de la BCD.

Si le modèle SysML est modifié, on pourra réutiliser l'algorithme de création d'AMDEC préliminaire et pointer les nouveaux composants, connexions ou exigences à étudier, tout en ne reproposant pas les éléments écartés ou inchangés. En comparant les éléments de la structure créée par l'outil support de MéDISIS et ceux liés à la base de gestion du tableau AMDEC provenant de la version précédente, on pourra identifier les nouveaux éléments. Si par exemple, une nouvelle connexion apparaît comme cause probable, on suggérera à l'analyste de l'étudier en affectant un statut « nouveau ». De même, si une connexion avait été proposée à une étape précédente, on ne la proposera pas à nouveau. Enfin, les éléments de la base AMDEC au statut « supprimé » sont comparés aux nouvelles propositions, mais n'apparaissent pas dans le tableau final de l'AMDEC. Le même type de raisonnement est applicable aux éléments de la BCD. Enfin, l'analyste est invité à ajouter à la BCD du projet chaque nouveau MdD et à en définir les caractéristiques. Cette flexibilité dans les études AMDEC est obtenue grâce à l'exploitation des modèles.

Les algorithmes de création d'AMDEC présentés sont basés sur l'exploitation de l'organisation des informations telles qu'elles sont présentées dans le métamodèle du langage SysML. Ceci nous permet de ne faire aucun a priori sur le format de fichiers analysés ou le langage de programmation employé pour implémenter l'algorithme. Dans le paragraphe suivant nous présentons rapidement les alternatives disponibles pour l'implémentation.

II.4.2 Implémenter l'algorithme d'analyse du modèle SysML

Le choix de la méthode d'implémentation dépend de l'EDS choisi. Cependant, nous avons exploré au cours de nos travaux quelques alternatives qui nous permettent de commenter le choix des techniques à employer. Trois possibilités majeures semblent envisageables : l'utilisation des fichiers d'échanges au format XMI, l'utilisation des fichiers natifs des outils de modélisation ou l'utilisation des services de création de code propres à ces mêmes outils. Les deux premières solutions peuvent s'appuyer sur l'utilisation de langages de transformation de modèles comme ATL (ATLAS Transformation Language) [Jouault & Kurtev 2005], [Chenouard et al. 2008] ou l'outil KerMeta [Muller et al. 2005].

II.4.2.1 Utilisation du format XMI

L'utilisation des fichiers d'échanges semble la solution la plus appropriée pour satisfaire notre volonté d'interconnecter les outils d'IS. En effet, ce format de fichier est développé et normalisé afin de soutenir les approches de développement de l'ISBM, en facilitant les échanges de modèles. Cependant le format employé n'est à ce jour pas optimisé pour réaliser le type d'étude que nous souhaitons mener. La recherche des éléments ciblés par l'algorithme de création de structures pour AMDEC préliminaires se fait par l'identification de balises dans le fichier XMI à la structure respectant le XML. Les informations telles que les noms et les types apparaissent comme des attributs des balises. Le parcours des indexations imbriquées des balises permet l'accès aux éléments désirés tels que les *part properties* ou les *value properties*. Néanmoins, le principal défaut de ces fichiers est qu'ils ne reflètent pas le contexte

d'utilisation des artefacts de modélisation employés dans le modèle. Il comprend la déclaration des éléments SysML mais en aucun cas leur déploiement au sein des diagrammes. On ne peut donc pas accéder aux diagrammes ayant été constitués ni à leur nom, ce qui interdit par exemple l'analyse par vues du système. De plus, les éléments « connecteurs » utilisés lors de la saisie de connexions dans les IBD ou Parametric Diagrams ne sont pas répercutés dans le fichier : ceci conduit à une très faible expression du contexte dans le fichier XMI. Seuls les éléments tels que les *messages* et *item flows* permettent d'identifier les éléments liés entre eux, car leurs extrémités sont considérées comme des attributs. De même, les relations d'allocation et les relations définies sur les *requirements* sont détectables dans le fichier et ainsi exploitables. Ces différents points ont été constatés sur deux outils : Artisan Studio[21] et le projet Open Source TopCased[22]. Nous avons pu néanmoins développer une part importante de l'algorithme en exploitant les fichiers fournis par Artisan Studio.

L'outil développé est capable par analyse des fichiers XMI, de construire pour chaque *part* une structure contenant tous les éléments recherchés à l'exception des connexions non porteuses d'*item flow* ou utilisant des *standard ports*. L'énumération des contraintes est exhaustive mais moins précise que prévue car on ne peut différencier si celles liées à l'*owning block* utilisent ou non le *part* étudié. Il est possible de compenser certains de ces problèmes par la méthode de modélisation. Par exemple, nous encourageons l'emploi de *flow ports* et *item flows* afin de pouvoir identifier toutes les connexions des IBD dans le fichier XMI. Ainsi, l'implémentation de l'algorithme est quasi-totalement couverte en employant quelques règles de saisie du modèle et le format XMI. Nous souhaitons tout de même apporter une dernière remarque à l'utilisation de ce format. Nous avons constaté, avec les outils que nous avons testés, que ces derniers ne génèrent pas le même XMI pour un même modèle. Ceci va à l'encontre de l'objectif du développement de ce format et empêche la réutilisation immédiate de l'implémentation développée d'un outil à l'autre. En effet, l'analyse du XMI passe par la recherche de balises pouvant se retrouver à des niveaux d'indentation différents suivant les outils.

Nous avons réalisé notre solution en utilisant des règles XSLT[23] (eXtensible Stylesheet Language Transformations), pour effectuer un « parsing » du fichier XMI menant à l'obtention des structures sous forme de texte [Cressent 2009]. Une alternative est d'utiliser un langage de transformation de modèles comme ATL. Ce langage outillé par la plate-forme AMMA [Bézivin et al. 2005] permet notamment la transformation de descriptions XMI vers des formats de fichiers textuels. Pour cela, il est nécessaire de fournir le métamodèle des données présentes dans le fichier XMI, ainsi que celui du fichier cible et d'écrire en ATL la routine de transformation de métamodèles.

II.4.2.2 Utilisation des fichiers natifs des outils de modélisation SysML

La seconde alternative est d'utiliser le format de fichier utilisé par l'outil de modélisation employé. Cette solution nécessite le développement d'un complément par outil. De plus, l'étude nécessite un temps d'analyse du format bien souvent spécifique à chaque éditeur. Ce type de fichier embarque généralement toutes les informations nécessaires à l'implémentation de l'algorithme. Ces fichiers contiennent aussi bien la déclaration des éléments du modèle que

[21] Artisan Studio est édité par Artisan Software Tools Inc. http://www.artisansoftwaretools.com.
[22] TopCased : http://www.topcased.org.
[23] XSLT spécification de la W3C : http://www.w3.org/TR/xslt.

leur organisation graphique. Dans les outils que nous avons testés, nous avons constaté que ces formats contiennent des informations plus riches. L'utilisation de telles solutions est à privilégier lorsque les outils de l'EDS sont bien ciblés et lorsqu'il est possible de développer des solutions conjointement avec le fournisseur de l'outil de modélisation. Dans ce cas aussi, il est possible de réaliser le service avec une transformation décrite en ATL. Il est alors nécessaire de connaître le métamodèle du fichier source. Pour un souci d'interopérabilité et de neutralité vis à vis des fournisseurs du marché, nous avons préféré ne pas développer ce type de solution et nous focaliser sur l'exploitation de fichiers XMI, malgré les défauts de leur utilisation.

II.4.2.3 Utilisation des services de génération de code

La dernière solution d'implémentation envisageable, est l'utilisation des générateurs de code proposés par certains outils. Ces services proposent la génération de morceaux de code type pour les différentes entités des modèles. On peut donc envisager de créer les structures de données adéquates via ces générateurs. Cette solution possède l'avantage de pouvoir bénéficier de l'ensemble des entités créées dans le modèle sans les restrictions trouvées dans le format XMI. Cependant, cette solution est comme la précédente spécifique aux outils sélectionnés. Nous ne l'avons donc pas adoptée.

II.4.3 Adaptation de la méthode de modélisation pour la génération d'AMDEC

Comme nous l'avons précisé au paragraphe II.4.2, nous avons considéré dans la création de notre algorithme qu'aucune méthode de saisie du modèle n'est employée. Ceci nous a permis de spécifier une routine d'analyse de modèle SysML adaptable à toutes approches de modélisation et s'affranchissant de l'usage de stéréotypes. Dans un cas d'application industrielle, il est primordial de définir clairement la méthode de saisie du modèle du système. Cette méthode précise comment les concepts sont modélisés (composants, fonctions, interfaces…) et quels artefacts SysML doivent êtré utilisés pour le faire (*block*, *flow port*, *constraint*…). La connaissance ou la mise en place de cette méthode permettra d'affiner l'algorithme proposé et de fournir une analyse automatique de plus en plus pertinente. Nous allons dans cette partie proposer quelques règles de modélisation pouvant être intégrées à une méthode d'IS utilisant SysML comme langage de modélisation. Ces règles permettent notamment une meilleure exploitation des fichiers au format XMI.

Les propositions données ici peuvent pour la plupart être ajoutées aux méthodes existantes. Elles sont dissociables les unes des autres et sont à sélectionner vis à vis des besoins et habitudes de la structure les intégrant. Ces règles sont classées en trois groupes de niveaux sémantiques différents : certaines portent sur le type d'élément à utiliser, d'autres sur les liens à déclarer entre entités, les dernières sont de simples règles de nommage. Les deux premiers types ont une portée méthodologique bien plus forte que les règles de nommage. Elles définissent des choix de modélisation ainsi que des techniques de réalisation des modèles bien formés pour les analyses de SdF.

II.4.3.1 Règles sur les choix des éléments de modélisation SysML

Il existe deux degrés de contraintes dans l'utilisation de règles de choix d'éléments pour la modélisation de concepts définis : l'usage des éléments du langage strictement limité à la spécification et l'usage de stéréotypes supplémentaires spécifiques à l'application, appliqués aux différents artefacts de modélisation.

Les règles que nous allons définir portent sur la façon de modéliser les composants, leurs propriétés et liens physiques. Ces règles permettent de clarifier le modèle et de simplifier son analyse automatique.

1) Déclarer un *block* pour chaque type de composant et sous-système détaillé dans le modèle.

2) Déclarer une *part properties* pour chaque composant d'un type de système ou sous-système.

3) Déclarer une *value property* pour chaque flux manipulé par le composant et pour chaque caractéristique physique du composant.

4) Déclarer un *flow port* pour chaque interface du composant modélisé et lui allouer la ou les *value properties* de flux (définies règle 3) qu'il relaye.

5) Utiliser les *blocks* uniquement pour définir un type de composant du système, de sous-système ou un système appartenant à son environnement.

6) Utiliser des *item flows* pour chaque connexion entre composants, représentée sur un IBD.

7) Utiliser uniquement des connexions passant par des *flow ports* dans les IBD.

8) Déclarer chaque fonction exercée par un composant comme *operation*.

Les 4 premières règles sont utilisées pour la création du modèle, elles permettent de se rapprocher du métamodèle de composant proposé au début de ce chapitre (figures 2.1 et 2.2).

La règle 4 induit que l'on ne souhaite pas utiliser les *standards ports*, cela est possible car un *flow port* peut toujours remplacer un *standard port* et cela simplifie l'écriture des modèles. On analysera ainsi plus facilement le modèle et on le rendra plus cohérent en éliminant l'ambiguïté du choix entre les deux types de port. De plus, en couplant cette règle avec les règles 8 et 9 on obtient un modèle XMI analysable au niveau des connexions figurant dans les IBD. Néanmoins, ce choix implique l'utilisation de *blocks* pour la modélisation de l'environnement du système (spécifié règle 5) car un *acteur* SysML ne peut être connecté à un *flow port*. Dans un souci de clarté, ce choix peut être couplé à l'usage d'un stéréotype « environnement » pour les *blocks* n'appartenant pas au système, ceci afin de simplifier l'usage de l'algorithme d'analyse des modèles SysML. Cependant, cette mesure ne devrait pas être nécessaire car les éléments d'environnement ne devraient pas se voir attribuer de *parts properties*.

La règle 5 permet de réaliser la recherche de l'algorithme d'analyse en garantissant que l'on ne remontera que des composants du système. La règle 2 garantit que tous les composants seront identifiés.

La règle 5 peut être remplacée par une utilisation de stéréotype lorsque la méthodologie d'IS le nécessite. Par exemple, lorsque l'on utilise l'OOSEM le concept de *block* est utilisé pour définir des composants logiques et des composants physiques. L'OOSEM demande donc l'emploi de stéréotypes différenciant les entités. L'utilisation de *packages* pour séparer les entités constitue également une solution viable, mais moins souple car ensembliste.

Les règles 6 et 7 ont été définies pour permettre la réalisation de l'algorithme d'analyse de modèle SysML. Comme nous l'avons remarqué au paragraphe II.4.2.1, les connexions ne sont interprétables dans le fichier XMI que si on analyse les *item flows* qu'elles portent. La règle 7 permet de spécifier qu'un composant nécessitera une interface dès lors qu'il voudra échanger un flux avec un composant proche. Ceci permet de mettre en évidence dès les premières étapes de conception un besoin de spécification et d'analyse des interfaces.

La règle 8 permet de rejoindre simplement le métamodèle présenté figures II.1 et II.2 et d'identifier les fonctions réalisées par les composants. A cette règle, nous pouvons suggérer d'ajouter les techniques de modélisation utilisées pour la définition de la BCD présentée au paragraphe II.3.5.2 (usage de *constraints* et Parametric Diagram pour définir les *opérations*). Cette façon de procéder demande l'ajout d'éléments exprimant formellement la création de liens entre artefacts du modèle.

II.4.3.2 Règles d'allocation d'éléments du modèle

La deuxième catégorie de règles que nous utilisons porte sur la déclaration de liens formels entre éléments du modèle. Les objectifs sont : d'obtenir des associations explicites et exploitables de façon automatique et de rendre le modèle plus cohérent et plus facilement navigable. Les éléments SysML employés sont les relations *allocate* et *satisfy*.

1) Allouer aux *blocks* les *requirements* afférents au type de composant par une relation « satisfy » : Normes, Caractéristiques demandées à toutes les unités produites. Le *requirement* est « satisfied by » *block*.

2) Allouer aux *parts* les *requirements* concernant un composant particulier par une relation « satisfy » : Caractéristiques et comportements attendus dans le contexte d'utilisation du composant. Le *requirement* est « satisfied by » *part*.

3) Allouer les *actions* réalisées par n'importe quel individu d'un type de composant au *block* par une relation « allocate ». L'*action* est « allocated to » *block*.

4) Allouer les *actions* spécifiques à une utilisation d'un composant dans le contexte du système à un *part* par une relation « allocate ». L'*action* est « allocated to » *part*.

5) Allouer par une relation « allocate » les *value properties* à la *constraint properties* les utilisant. La *value property* est « allocate to » *constraint property*.

Les règles 1 et 2 sont utilisées pour une allocation pertinente des exigences sur la structure du système. Elles permettent de faciliter la première phase de l'analyse du modèle.

Les règles 3 et 4 sont utilisées pour permettre la recherche de liens entre entités physiques au sein des Diagrammes d'Activités. Elles viennent compléter l'exemple donné au chapitre 15 de la norme SysML [OMG 2008]. L'utilisation de ces règles rend donc le modèle plus expressif en matière de portée des *actions*.

La règle 5 est purement dédiée à notre application d'implémentation de l'algorithme d'analyse sur les fichiers XMI. Déclarer une telle allocation permet de réaliser l'analyse des contraintes la plus précise possible, comme nous l'avons mentionné au paragraphe II.4.2.1. En effet, en exploitant ces allocations on sera capable de retrouver toutes les *constraint properties* faisant appel aux *value properties* du *part* étudié.

II.4.3.3 Utilisation de règles de nommage

Si l'outil d'analyse employé est basé sur l'identification du nom des entités, on peut substituer le stéréotypage par des règles de nommage. Cependant, ce moyen très artificiel n'est guère utilisable car très peu robuste aux changements de règles introduits au cours du temps. Néanmoins, l'usage de règles de nommage est conseillé pour la réalisation du modèle. Ces règles le rendront beaucoup plus lisible et donc navigable. Le but des règles de nommage est de permettre l'unicité des identifiants à travers le modèle, tout en obtenant des dénominations suffisamment expressives pour retrouver intuitivement les relations entre éléments. Dans l'exemple de BCD donné au paragraphe II.3.5.2, nous en avons proposé pour les redéfinitions dysfonctionnelles des *operations* du composant décrit. Elles peuvent être extrêmement variées employant des préfixes, des casses ou des compositions différentes. Nous ne pouvons donc que suggérer des exemples de stratégie de nommage : préfixer différemment les *value properties* relatives à des flux ou des caractéristiques, préfixer le nom d'une *action* par les trois premières lettres du *part* l'exécutant…

II.5 Conclusion

Cette partie montre que les méthodes d'étude initiatrices des analyses de SdF peuvent faire partie intégrante d'une méthode d'Ingénierie Système Basée sur les Modèles. Nous avons en particulier étudié le cas de la réalisation d'Analyse des Modes de Défaillance de leurs Effets et Criticité (AMDEC) à partir de modèles SysML purement fonctionnels. Pour une intégration efficace dans le processus d'Ingénierie Système, cette analyse doit respecter différents critères : être applicable sur les modèles communs au processus, être suffisamment rapide pour conserver la dynamique du processus d'ingénierie et fournir des résultats traçables et intégrables aux bases de connaissances du processus. Ces investigations portent sur la première partie de la méthode MéDISIS, utilisée pour initier l'analyse Sûreté de Fonctionnement en recherchant les comportements dysfonctionnels primaires du système.

Nous avons étudié dans ce chapitre comment réaliser la recherche de connaissances sur le système développé lors de la création d'une AMDEC composant. Pour cela, nous avons proposé un métamodèle de données illustrant le processus. Ceci nous a permis de mettre en exergue les éléments à rechercher dans les modèles SysML fonctionnels afin de réaliser l'AMDEC. Nous avons ensuite analysé le métamodèle du langage SysML pour trouver les points de recouvrement entre les deux métamodèles. Cela nous a permis de montrer que les modèles SysML sont analysables pour la création d'AMDEC. Ce raisonnement au niveau métamodèle nous permet d'adopter un degré d'abstraction suffisant pour proposer une méthode d'analyse indépendante des méthodes et outils de modélisation choisis pour construire le modèle fonctionnel du système.

Nous avons détaillé un algorithme d'analyse automatique des modèles SysML pour la création d'une structure d'éléments nécessaires à la réalisation d'AMDEC. Nous avons ensuite introduit l'utilisation de ces informations pour assister la création des AMDEC, avant de mentionner comment gérer le tableau AMDEC au sein du processus d'Ingénierie Système. Ces recherches ont permis de montrer les bénéfices de l'usage de modèles SysML pour la réalisation d'AMDEC :

- L'AMDEC est intégrée au processus d'Ingénierie Système par la traçabilité des entités analysées et par l'intégration des résultats aux bases de connaissances communes.
- Le processus AMDEC est accéléré par l'automatisation de ses phases les plus lourdes, i.e. recensement des composants, des Modes de Défaillance, navigation du modèle. Ce point n'avait jusqu'alors jamais été traité dans les travaux sur l'automatisation des AMDEC.
- L'analyse d'un modèle SysML améliore la qualité des AMDEC et permet de mettre en évidence différents points clés de l'étude (interfaces, communications, liens structurels).
- L'analyse d'un modèle SysML donne des leviers techniques pour exploiter automatiquement les bases de retour d'expériences incontournables pour ces études (notamment par l'exploitation des types des composants).

Nous avons ensuite évoqué les différentes alternatives utilisables pour l'implémentation d'un tel algorithme d'analyse et discuté des apports de l'emploi d'une méthodologie de saisie du modèle pour l'analyse automatisée du système. Nous avons montré comment les premières

phases d'une analyse de risques et Sûreté de Fonctionnement pouvaient être intégrées au processus d'Ingénierie Système et quels bénéfices elles tiraient d'une telle démarche. Cette première étape sert au recensement des comportements dysfonctionnels à considérer si l'on veut aller plus loin dans l'analyse du système. Suivant les cas d'étude considérés, les besoins d'analyse ne seront pas les mêmes. Il pourra être nécessaire de conduire des analyses de disponibilité, de sécurité ou de fiabilité. Nous allons dans le prochain chapitre étudier comment poursuivre l'analyse en intégrant les résultats de cette première étude, ainsi qu'en exploitant les ressources offertes par l'Environnement de Développement Système. Nous allons montrer comment intégrer les méthodes et outils utilisés pour la validation et la quantification formelle des caractéristiques de Sûreté de Fonctionnement, au sein du processus d'Ingénierie Système Basée sur les Modèles.

Chapitre III

De SysML aux modèles AltaRica Data Flow

Résumé du Chapitre III :

Ce chapitre poursuit la description des phases de la méthode MéDISIS. Il présente la mise en place d'une traduction des modèles SysML vers une description utilisant AltaRica Data Flow, destinée à l'analyse quantitative de la SdF des systèmes analysés. La création des modèles AltaRica Data Flow est organisée en deux étapes. Dans un premier temps, la partie fonctionnelle du modèle est obtenue par traduction directe du modèle SysML du concepteur. Puis, la partie concernant le comportement dysfonctionnel du système est ajoutée en exploitant les données recueillies grâce à l'AMDEC et gérées à travers la Base de données des Comportements Dysfonctionnels accompagnant MéDISIS. Nous donnons dans ce chapitre des règles de traduction du modèle SysML vers AltaRica Data Flow et proposons des techniques permettant d'optimiser ces traitements. Nous proposons également une architecture de la Base de données des Comportements Dysfonctionnels permettant de gérer tous les aspects inhérents à la vue dysfonctionnelle des composants du système. Le métamodèle complet de la base est décrit, ainsi que son utilisation au cœur de MéDISIS. Ce travail nous permet également, comme au chapitre II, de proposer des conseils de modélisation pour l'établissement du modèle SysML fonctionnel du système analysé.

III.1 Introduction

L'objectif poursuivi dans cette thèse est de contribuer à l'intégration des étapes d'analyse et de validation de la Sûreté de Fonctionnement au processus global d'Ingénierie Système Basée sur les Modèles. La constitution d'un modèle formel du système reflétant son comportement dysfonctionnel est un point crucial de ces phases de validation. Les qualités que l'on cherche à atteindre au cours de cette étape sont en tous points identiques à celles recherchées pour le processus AMDEC : traçabilité envers le modèle central du système et ses exigences. La volonté principale est donc d'assister, voire d'automatiser la création du modèle formel à partir du modèle fonctionnel du système et des connaissances du comportement défaillant. Nous proposons donc de réaliser une traduction des modèles SysML vers le langage AltaRica Data Flow, dans une approche similaire à [Dumas et al. 2008] qui ont exploré la conversion de modèles AADL[24] en descriptions AltaRica Data Flow.

Nous nous plaçons maintenant dans une phase de l'étude du système où l'AMDEC a été réalisée. Par cette analyse, constituant le lancement de MéDISIS, les analystes ont obtenu les Modes de Défaillance de chaque composant et ont pu cibler leurs effets potentiels sur le système. Dans le processus de validation de la Sûreté de Fonctionnement du système (par exemple étape 12 du processus de l'IEC 61508, figure I.4), il est nécessaire de pouvoir qualifier et quantifier ces risques de défaillance. Ces validations viennent en réponse aux exigences formulées par les parties prenantes ou en prévision d'une demande de certification. Cette étape est la seconde phase de MéDISIS, nous avons ici étudié comment intégrer ce type d'analyse, faisant appel à des langages et représentations formels, à l'Ingénierie Système Basée sur les Modèles et son Environnement de Développement Système. Il s'agit donc d'utiliser toutes les connaissances contenues dans le modèle fonctionnel commun au processus d'Ingénierie Système, mais également d'exploiter les connaissances relevées sur le comportement dysfonctionnel au cours de l'AMDEC.

Durant cette thèse, nous avons plus particulièrement étudié l'emploi du langage AltaRica Data Flow dans un tel contexte. Dans ce chapitre, nous proposerons une rapide présentation d'AltaRica Data Flow et justifierons le choix de ce langage comme première cible d'étude pour l'intégration à l'Environnement de Développement Système. Nous montrerons, en particulier, que cette solution a déjà été employée pour de nombreuses problématiques de validation de critères de Sûreté de Fonctionnement, recherchant des indicateurs variés. Nous étudierons ensuite l'aide à la saisie du modèle dysfonctionnel en AltaRica Data Flow, en ciblant la traduction des éléments du modèle SysML en AltaRica Data Flow. Cette construction se fera en trois phases et sous deux vues : la vue fonctionnelle et la vue dysfonctionnelle. Nous détaillerons la retranscription de la structure du système, puis de son comportement fonctionnel et enfin l'adjonction du comportement dysfonctionnel. Pour les deux premières parties, nous travaillerons à partir du modèle SysML issu de l'analyse fonctionnelle. Pour la finalisation du modèle nous discuterons de la gestion et de l'utilisation d'une Base de données des Comportements Dysfonctionnels dont les premières parties ont été proposées au paragraphe II.2.5.2.

Note de style : Dans ce chapitre les éléments du langage AltaRica Data Flow sont indiqués en **gras**.
Rappel de style : Les éléments du langage SysML sont indiqués en *italique*.

[24] AADL : Architecture Analysis & Design Language [SAE 2009].

III.2 Le langage AltaRica Data Flow

Le langage AltaRica Data Flow (DF) est une adaptation du langage AltaRica lui-même décrit dans [Arnold et al. 2000]. Il est plus précisément dédié à la réalisation d'études de SdF, comme les études de disponibilité et suivi de production [Boiteau et al. 2006] ou les études de sécurité [Bieber et al. 2004]. Le langage emploie les automates de modes comme formalisme sous-jacent [Rauzy 2002], les systèmes modélisés prennent ainsi la forme de structures de Kripke, automates finis non déterministes. Cette version simplifiée d'AltaRica forme un langage de description inspiré de spécifications telles que les Statecharts et peut être vue comme une généralisation des Réseaux de Petri, grafcets et diagrammes Bloc [Rauzy 2002], [Rauzy et al. 2008]. Les simplifications apportées (interdiction des ports bidirectionnels et de la diffusion d'événements) permettent la déduction de formules logiques booléennes, utilisées lors de la construction d'arbres de défaillance et la recherche de scénarios de défaillance. Les systèmes sont saisis en AltaRica DF par une approche compositionnelle. Les composants peuvent être agencés de façon hiérarchique ; ils communiquent entre eux par l'échange de flux ; la synchronisation d'actions est réalisée par le biais d'événements. Un modèle AltaRica DF est un système arborescent dont la racine est le système global et dont les nœuds sont des composants ou groupe de composants. Dans ce langage, les évolutions du système sont considérées asynchrones (les changements d'état entraînent instantanément le changement du traitement des flux).

Un modèle AltaRica DF représente le système de façon arborescente. Le système global est modélisé par un **nœud** principal (**node main**), il possède souvent des sous-nœuds décrivant les composants du système. Un modèle AltaRica DF possède donc toujours un **node main**.

III.2.1 Les nœuds

Les **nœuds** sont les constituants de base des modèles AltaRica DF. Ils servent à déclarer les concepts instanciés par les composants. Un **nœud** (**node**) est caractérisé par ses états atteignables (**state**), les flux qu'il traite (**flow**), les événements (**event**) auxquels il est sensible. L'état du composant est la valuation de ses variables d'état, les variables de flux servent à la communication avec l'environnement. Les variables de flux reçus (**in**) sont les éléments traités par le composant ; les variables de flux sortants (**out**) sont l'impact du composant sur l'environnement et les phénomènes transmis aux composants voisins. Ces différentes variables peuvent être typées sous quatre formes : booléen (**bool**), entier (**integer**), réel (**float**) ou énumération (**domain**). Le traitement des flux est défini relativement à l'état du composant par l'intermédiaire d'assertions (**assert**). Les assertions relient états courants, flux entrants et valeurs des flux sortants. Ces assertions sont des expressions logiques ou algébriques. Les composants sont sensibles à des événements modélisant les possibilités d'évolution du système. Les transitions (**trans**) décrivent le changement des variables d'état dans le cas où un événement devient effectif. Les événements modélisent donc également les évolutions internes aux composants. Cela peut être une panne, un changement de mode de fonctionnement ou une réparation du composant. Un **nœud** peut posséder des informations complémentaires utilisées pour la simulation comme les états initiaux (**init**). La modélisation étant hiérarchique, un **nœud** peut également posséder des sous-nœuds (**sub**). Nous donnons ci-dessous un exemple de **nœud** représentant une vanne. La variable « ouverte » donne son état, f1 et f2 sont les flux d'entrée et de sortie. La transition inverse l'état de la vanne. L'assertion donne l'état du flux sortant en fonction du flux entrant et de l'état courant.

```
node Vanne
      state ouverte : bool ;
      flow
            f1 : bool : in ;
            f2 : bool : out ;
      event inversion ;
      init ouverte = true ;
      trans
            true |- inversion -> ouverte := ~ouverte ;
      assert
            if ouverte
                  f2 = f1 ;

            else
                  f2 = false ;
  edon
```

III.2.2 Synchronisation, priorité, loi de probabilité

AltaRica DF propose une représentation asynchrone pour l'évolution des états et des flux, mais permet la synchronisation d'événements. Il est ainsi possible de synchroniser l'apparition d'événements forçant ainsi des réactions simultanées des composants. Ceci permet la description de la chaîne de conséquences entre divers phénomènes ou de lancer des tâches parallèles entre plusieurs composants. Dans des considérations de SdF, ce mécanisme permet de modéliser des défaillances simultanées expressions d'un mode commun. L'AltaRica DF autorise la synchronisation à n'importe quel niveau de hiérarchie dans la description du système. Il est également possible de définir des priorités entre les différents événements ainsi que des exclusions mutuelles. Enfin, afin d'utiliser ces modèles en simulation, il est possible d'affecter des lois de probabilité aux événements (**law**). Par exemple, le modeleur peut indiquer qu'un événement suit une loi exponentielle dont il fixera le paramètre. Ceci est particulièrement adéquat pour la modélisation de défaillances dont le taux d'apparition suit une règle donnée.

III.2.3 Outils

Le langage AltaRica DF est exploité par de nombreux outils dédiés aux études de SdF. [Boiteau et al. 06] mentionnent l'existence d'un outil de simulation interactif, d'un générateur d'arbres de défaillances et séquences, d'un vérificateur de modèles, d'un analyseur de processus markoviens et d'un simulateur stochastique. L'outil professionnel BPA-Das développé par Dassault Systèmes intègre ce type d'outils et permet une exploitation opérationnelle du langage. De plus, les modèles AltaRica DF sont exploitables pour la simulation de Monte Carlo, ce qui permet d'envisager l'analyse du modèle sans passer par une génération de processus markovien. De nombreux outils ont été développés par la société Arboost, ils ont pour objectif de faciliter l'utilisation d'AltaRica, et de le rendre exploitable pour l'analyse SdF. On peut trouver les modules suivants :

- alta-chk : un type-checker.
- alta-ata : un compilateur transformant une description hiérarchisée en un simple nœud.
- alta-a2b : un compilateur d'AltaRica Data Flow en arbres de défaillance.
- alta-sim : un simulateur graphique interactif.
- alta-a2g : un outil calculant le graphe d'accessibilité de description en AltaRica DF.

- alta-mrk : un outil d'analyse de processus de Markov.
- alta-sto : un simulateur stochastique.
- alta-msg : un générateur de séquences pour graphe de Markov.
- alta-sop : un générateur de coupes minimales.
- alta-seq : un générateur de séquences décrites par des automates.
- alta-std : un outil de description.

AltaRica DF constitue une solution de modélisation en vue d'analyse SdF très pertinente car déjà bien implantée dans la communauté scientifique et industrielle. Il a déjà prouvé son efficacité pour résoudre des problématiques allant de l'analyse de risque, aux calculs de disponibilité de systèmes industriels. Le langage bénéficie, comme nous venons de le voir, d'outils variés et performants pour soutenir son utilisation. L'offre actuelle est ainsi composée de traducteurs et analyseurs de modèles performants, mais également d'un environnement de saisie accessible, maintenu et soutenu efficacement. Le point déterminant est donc de vérifier si ce langage peut constituer la suite logique de l'ISBM utilisant SysML vers les analyses SdF. Pour cela nous avons étudié comment établir une collaboration entre les deux langages.

III.3 Exploitation de SysML pour la génération de modèle AltaRica DF

Afin de résoudre les problèmes de complexité de modélisation des grands systèmes multi-technologiques, SysML propose de suivre la voie d'une représentation hiérarchique et compositionnelle. Le concept de *block* se retrouve donc au centre de la représentation. Dans un second temps, la spécification met l'accent sur la représentation des liens constituant le maillage de l'architecture étudiée. Les concepts de flux se sont donc vus attribuer une modélisation structurée par leur typage, orientation et valeurs. Enfin, SysML propose de nombreux concepts de modélisation offrant la possibilité de caractériser les changements d'états et le séquencement des traitements réalisés. Ces stratégies pour aborder la modélisation d'un système se retrouvent dans le langage AltaRica DF, dont la notion de composants s'échangeant des flux influencés par et influençant leur comportement est fondatrice. La suite de cette section a pour objet d'explorer ces similitudes et de proposer une méthode de traduction entre ces modèles pouvant aller jusqu'à l'automatisation du passage d'une représentation à l'autre.

La construction du modèle AltaRica est effectuée en deux étapes. Dans un premier temps, le modèle fonctionnel est constitué. Ensuite, la partie dysfonctionnelle est ajoutée sur le modèle précédemment constitué dans lequel apparaissent les composants physiques de l'architecture et leur comportement nominal. Cette seconde étape est réalisée grâce aux informations glanées lors de l'étude AMDEC et par l'utilisation de la BCD commune au processus d'ingénierie global. Cette construction utilise clairement les modèles et données de l'EDS, garantissant ainsi une traçabilité et une exploitation maximale de ces connaissances. Pour chaque élément de la transformation de modèles, nous identifions les éléments du modèle SysML à prendre en compte et leur expression en AltaRica DF. Pour chacune des phases de la transcription, nous analyserons les restrictions ou adaptations nécessaires en matière de modélisation SysML pour automatiser le processus.

III.3.1 Traduction de la structure

Nous créons tout d'abord la partie du modèle correspondant à la description structurelle du système. Le but général de la création du modèle AltaRica DF est d'atteindre une bonne traçabilité et une cohérence avec le modèle SysML commun à l'EDS. C'est pourquoi, la structure du système est directement traduite du modèle SysML. La description structurelle est créée en deux phases : les concepts de composants (les types) sont énumérés, puis les composants sont instanciés et connectés.

III.3.1.1 Traduction des éléments manipulés dans le modèle

La déclaration des types de composant existant dans le modèle se fait par les BDD. La première étape de la traduction concerne donc la transcription des *blocks*. Les deux langages proposent une approche Orientée Objet, cette traduction est donc directe. Chaque *block* du modèle SysML utilisé pour expliciter un type composant est traduit par un **nœud** dans le fichier AltaRica DF. Pour chacun des éléments traduits, le nom affecté est évidemment celui utilisé dans le modèle SysML. Ensuite, il est nécessaire de transcrire les propriétés du *block*. Cela se fait par la déclaration des variables d'état et de flux ainsi que des sous-composants. Les *part properties* sont les sous-composants, ils sont donc traduits par un élément **sub**, le type de cet élément est le même que celui du *part*. Les *value properties* correspondant à des

variables d'état sont traduites par un **state**, celles correspondant à des variables de flux sont modélisées par un **flow**. De même, les *port properties* correspondent à une variable de flux notée dans le modèle par **flow**.

La philosophie OO des deux langages rend cette mise en correspondance assez simple de prime abord. Néanmoins, cette phase recèle de nombreuses difficultés causées par la totale liberté offerte pour le typage des propriétés en SysML et l'absence d'une méthodologie de modélisation imposée par nos hypothèses. La première difficulté est de différencier les variables de flux des variables d'état. Dans le modèle SysML les deux concepts sont définis sous la forme de *value properties*, or AltaRica DF différencie les deux types d'élément. Plusieurs techniques peuvent être employées pour en définir la traduction. La première est d'utiliser la méthodologie proposée dans [Arnold et al. 2005] en répondant à la série de questions suivante pour déterminer la nature de la *value property* :

1. La variable est-elle modifiable par l'environnement sans que le composant s'en aperçoive ?
2. La variable est-elle une information dont l'environnement a besoin ?
3. La variable est-elle modifiable seulement par le composant ?
4. Est-il nécessaire de mémoriser la variable pour les décisions ultérieures ?

En cas de réponse affirmative pour une des deux premières questions, la variable doit être une variable de flux (traduite en **flow**). En cas de réponse affirmative pour une des deux dernières questions, la variable est une variable d'état (traduite en **state**). Ces questions aident au choix mais les auteurs ne garantissent pas qu'elles puissent couvrir tous les cas rencontrés. De plus, ce résultat n'est pas automatisable pour intégrer un outil de l'EDS car il nécessiterait l'intervention d'une intelligence artificielle.

La seconde solution est de substituer ces questions à une interrogation relative au modèle SysML : la *value property* peut-elle être allouée à un port ? Dans le sens où l'information de ce port correspondrait à la valeur de la *value property*. Si la réponse est oui, la *value property* doit être traduite en **flow**, sinon par un **state**. Ce type de question peut être tranché par l'emploi d'une méthodologie de modélisation imposant de déclarer et d'allouer une *value property* pour chaque port du *block*, cette règle est notamment compatible avec la règle 4 proposée au paragraphe II.3.3.1. Dans ce cas on crée un **flow** par *value property* allouée à un port et un **state** par *value property* non allouée. Cette règle permet a priori de réaliser la traduction, néanmoins, l'automatisation nécessite l'emploi d'une méthodologie de conception. Cette deuxième alternative est préférable car comme nous le remarquions en fin de second chapitre, il sera incontournable d'employer une méthodologie de saisie du modèle SysML pour que le langage et la méthode MéDISIS offrent leurs bénéfices maximums au processus d'ISBM.

Cependant, une ambiguïté subsiste dans la déduction des variables de flux, venant du fait que le port et la variable de flux sont confondus en un même objet en AltaRica DF, quand SysML distingue les deux concepts. Il est nécessaire pour définir complètement ces flux en AltaRica de définir leur direction. En SysML cette information est portée par le port (l'attribut *direction* des *flow ports* ou le type d'interface des *standard ports*), lors de la traduction il est alors nécessaire de considérer le port alloué aux *values* traduites. Si la direction est *in* (respectivement *out*), le **flow** sera qualifié en **in** (respectivement **out**) en AltaRica DF. Un problème se pose si le port est de *direction inout* ; en effet, un **flow** ne peut être de direction **inout** en AltaRica DF. Il est alors nécessaire de dissocier la *value property* en deux **flows** de direction **in** et **out**. Une convention de nommage doit alors être adoptée pour garder la cohérence entre

modèle (par exemple préfixer le nom de la variable par in_ ou out_). Il est également nécessaire d'ajouter une assertion assurant l'égalité permanente de ces flux dissociés représentant un unique concept de type : `assert in_f = out_f ;`. Cette construction séduisante est néanmoins à bannir pour une utilisation future du modèle dans des outils d'analyse du modèle AltaRicaDF. En effet, elle induit une boucle dans le modèle et remplace un élément (sens **inout**) qui a été retiré d'AltaRica DF pour permettre la recherche de scénarios de défaillance. Dans l'absolu, l'utilisateur de la transformation de modèle devra choisir un sens unique de propagation, définissant la direction dans laquelle il souhaite considérer la propagation des défaillances.

Le second point difficile de la transformation est la gestion du type des variables. En SysML le typage est laissé totalement libre à l'utilisateur, quant en AltaRica DF il est limité à l'emploi de quatre types que nous avons mentionnés au paragraphe III.2.1. La difficulté est donc de transcrire le type de la *value* ou du *port property* dans celui qui lui correspond en AltaRica. En se référant à la spécification de SysML, seule la traduction d'une *énumération* peut être convertie automatiquement en un **domain** listant les mêmes éléments et réutilisant le même nom. Pour tous les autres types, il n'existe aucune définition établie dans la spécification SysML. La traduction doit donc se faire en considérant pour chaque variable le type le plus approprié. Ce choix du type dépend fondamentalement de l'analyse que l'on souhaite faire du modèle. Pour la recherche de chemins de défaillance, l'utilisation de booléens et d'énumérations peut suffire car dans ces cas, l'utilisateur souhaite seulement juger si les états et les flux délivrés sont valides. Lorsqu'il s'agit d'évaluer une production, l'emploi d'entiers et réels est nécessaire. Pour chaque variable il faut donc se poser les questions suivantes :

1. La variable prend-t-elle seulement deux valeurs ?
2. Si non, la variable possède-t-elle un nombre fini de valeurs ?
3. Peut-on se contenter d'une vue qualitative de la variable (valide/non-valide) pour réaliser l'étude de SdF ?
4. Peut-on se contenter d'un nombre fini de valeurs de la variable pour réaliser l'étude de SdF ?

Tableau III.1 *Identification du type des variables*

Question	Réponse	Type accordé à la variable
1	Oui	**Bool**
2	Oui	**Domain**
3	Oui	**Bool**
4	Oui	**Domain**
1, 2, 3, 4	Non	**Integer, Float**

Ce type de réflexion peut s'utiliser dans le cadre d'une méthodologie de saisie du modèle SysML. La définition de types de référence pour la constitution du modèle permettra d'organiser la génération du fichier AltaRica DF. De plus, intégrer des règles de typages des variables ne rendra l'ISBM que plus pertinente et cohérente. Nous proposons le type de règle suivant : définir trois *value types* « entier », « réel » et « booléen », puis pour chaque type défini, indiquer par une relation de spécialisation, si le nouveau type représente une quantité booléenne, réelle ou entière. L'utilisation d'*énumérations* est toujours autorisée et ne nécessite pas de lien avec les types « réel », « entier » et « booléen ». Le respect d'une telle règle de modélisation permet une traduction automatisée des types des variables : de façon directe, les types dérivés de « réel » seront traduis en **float**, ceux dérivés d' «entier » en **integer**, ceux dérivés de « booléen » en **bool** et les *énumérations* en **domain**.

Si le *flow port* traduit est typé par une *flow specification*, il est nécessaire de déclarer autant de variables de flux, qu'il y a de *flow properties* participant à la *flow specification*. Pour chacune d'entre elles, le sens défini dans la *flow specification* est repris pour orienter la variable. De façon identique, le type est utilisé de la même façon que pour les *flow ports* à typage simple.

Le dernier point limitant la traduction automatique des concepts utilisés dans l'axe structurel est relatif à l'identification des blocks exprimant les types de composant. Comme nous l'avons mentionné pour la création d'AMDEC, les *blocks* SysML ne modélisent pas uniquement des concepts physiques. C'est pourquoi, avant de réaliser la traduction vers AltaRica DF, il est nécessaire d'identifier les *blocks* relatifs aux concepts des composants. Pour cela, seule la méthodologie de saisie du modèle peut être utilisée. Les règles à employer sont les mêmes que celles énoncées pour favoriser la création de l'AMDEC préliminaire (règle 5 du II.3.3.1). La distinction peut se faire également par l'emploi de stéréotypes ou règles de nommage.

III.3.1.2 Instanciation et connexion des composants

La création de la structure se poursuit par la réalisation du système, en déclarant et connectant les composants utilisant les concepts définis par les **nœuds**, c'est-à-dire sur l'analyse des relations définies dans les IBD. Les composants sont instanciés par les instructions **sub** créées lors de la traduction des *blocks*. Pour chaque **nœud**, les composants ont été nommés comme les *parts* dont ils sont issus et partagent le même type. Pour les connecter, il convient de déclarer les égalités entre leurs variables de flux (une sortie donne la valeur d'une entrée d'un autre composant). Ces relations sont celles définies par la création des connecteurs entre ports sur les IBD. Si l'on applique les règles d'utilisation des ports et *value properties* mentionnées au II.2.1.1, la traduction est simple. Pour chaque paire de ports connectés entre eux, on déclare dans la clause **assert** du **nœud** AltaRica DF une égalité entre les variables de flux déduites des ports considérés. Pour illustrer ce propos, nous donnons l'exemple de la partie de code AltaRica DF équivalente à l'analyse du diagramme de la figure I.8.

```
node CapteurIntelligent
    flow
          In : float : in ;
          Out : bool :out ;
    sub
          cs :CelluleSensible ;
          cond : Conditionneur ;
          ut : UnitéTraitement ;
          mc : ModuleCommunication ;
    assert
          In = cs.In ; cs.Out = cond.In ;
          cond.Out = ut.In ; ut.Out = mc.In ;
          mc.Out = Out ;
edon
```

Remarque : le capteur possède deux variables de flux provenant des ports du *block* « CapteurIntelligent ». En appliquant la réflexion proposée au paragraphe précédent (III.2.1.1), nous avons conclu que ces flux étaient respectivement un réel et un booléen.

Cette phase de traduction permet de retrouver fidèlement la structure du système ainsi que les caractéristiques manipulées. Néanmoins, comme nous l'avons répété au cours de l'analyse, la traduction est assujettie à l'emploi d'une méthode de modélisation SysML pour

être automatisée. Ce type de traitement permet d'obtenir très rapidement un squelette complet du modèle, fidèle à la modélisation des concepteurs et prêt à accueillir la déclaration des comportements que l'on souhaite étudier durant les analyses de SdF.

III.3.2 Traduction du comportement fonctionnel

La description de la structure du système, obtenue par les étapes de traduction précédentes, peut être enrichie par la déclaration du comportement fonctionnel tel qu'il est exprimé dans le modèle SysML central de l'EDS. En AltaRica DF, ces aspects dynamiques sont modélisés par l'expression des transitions et des assertions. Nous étudions dans cette section quels artefacts SysML sont à l'origine de la création de ces deux types de construction. Les transitions traduisent le comportement événementiel des composants répondant à des sollicitations de son environnement ; les assertions modélisent le comportement interne du composant dépendant de son état et de ses entrées. Cette définition comportementale est affectée en AltaRica DF aux composants, il est donc possible de la déduire uniquement d'artefacts SysML décrivant le comportement et affectés à l'architecture.

III.3.2.1 Identification des transitions

Une transition est construite autour d'une garde, d'un événement et d'une affectation. La garde (**guard**) donne la condition de sensibilité à l'arrivée d'un événement (**event**) qui provoque une affectation des variables d'état. Il est difficile de trouver une traduction systématique d'un modèle SysML vers ces entités tant le nombre d'artefacts différents à considérer est important. De plus, un même élément SysML ne se traduit pas nécessairement tout le temps de la même façon en AltaRica DF. Enfin, différents éléments peuvent être utilisés en SysML pour représenter un seul artefact du modèle AltaRica DF, comme cela est le cas pour les *value properties* et les *flow ports*.

Pour cette partie de la transformation, les diagrammes utilisés dans le modèle SysML conditionnent fortement le travail. Si le comportement des *blocks* est décrit dans une *statemachine* la traduction est très efficace, sinon l'analyse ne peut donner que des pistes de traduction. Nous scindons donc notre réflexion entre les modèles contenant *des statemachines* et ceux n'en incluant pas. AltaRica DF repose sur les automates de mode pouvant être vus comme des machines à états [Rauzy 2002], c'est pourquoi la traduction des *statemachines* en AltaRica DF est assez directe. Les transitions représentent le même phénomène dans les deux langages. Une transition de *statemachine* est composée d'une garde, d'un événement déclencheur (que nous appellerons déclencheur) et d'une action de transition comparable à une affectation. La seule précaution à prendre durant la traduction, est d'analyser si les variables utilisées dans la *statemachine* sont bien des variables d'état dans l'affectation et des variables de flux entrant ou d'état dans la garde. Cette vérification peut notamment faire partie d'une vérification de modèle. Si les conditions sont réunies, l'expression des gardes, des déclencheurs et actions de transition sont reprises et l'événement (objet de type *event* ou *operation* dans la spécification SysML) est déclaré dans le champ **event** et recopié comme événement de la transition. Deux cas de figure se présentent pour l'action de transition : soit il s'agit d'une affectation de variable, auquel cas l'opération est reprise, soit il s'agit de l'émission d'un autre événement, auquel cas le phénomène doit être noté dans le bloc **sync** dans lequel on indique le second événement créé. Ensuite l'affectation est enrichie du changement d'état réalisé par la transition : on recopie dans l'affectation les valeurs spécifiées dans l'état d'arrivée de la transition pour les variables d'état. Il en est de même pour la garde qui est augmentée des conditions sur les valeurs des variables d'état du composant dans l'état origine

de la transition. Si ces valeurs ne sont pas précisées, la garde et l'affectation sont laissés telles quelles. Enfin, les affectations sont également constituées des traitements effectués en sortie de l'état source et en entrée du nouvel état. Ces constructions sont respectivement préfixées dans la *statemachine* par les mots clefs *exit* et *entry*. Nous donnons un exemple simple de traduction des transitions d'une *statemachine* sur la figure III.1. Cette dernière illustre la traduction des transitions d'ouverture et fermeture d'une vanne. La transition de fermeture à ouverture (passage de l'état « opened » à « closed ») est déclenchée par l'*operation* « InvertState », repris comme **event**. L'état source est caractérisé par une valeur « True » pour la variable d'état « opened », formant ainsi la garde « opened = True » pour la transition de fermeture. L'action « entry » réalisée dans l'état d'arrivé (« closed ») constitue l'affectation de la transition, donnant « opened = False ».

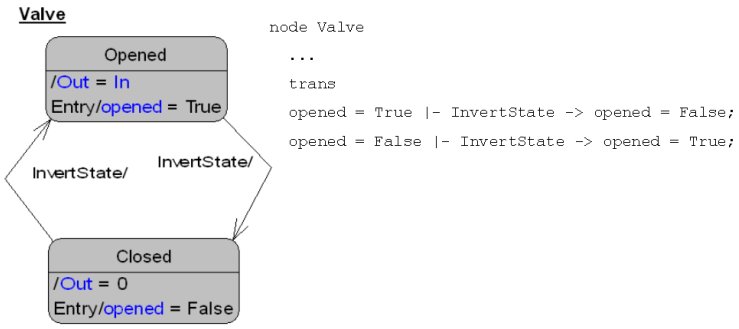


Figure III.1 *Exemple simple de traduction d'une statemachine*

Les autres diagrammes pouvant receler les informations traduites sous forme de transitions sont les SD et AD car ils décrivent le comportement inter-composants du système. Néanmoins, ces modèles donnent des informations moins précises car ces deux notations sont beaucoup moins formelles que les *statemachines*. Les SD modélisent les interactions inter-composants et adoptent donc une vue différente des transitions modélisant les réactions internes aux composants. Cependant, les informations qu'ils contiennent sont intéressantes pour déterminer la synchronisation des événements. Cette synchronisation peut être indiquée à l'échelle du *block* possédant les *parts* décrits dans le SD considéré. Chaque *message* est un *event* ou une *operation*, chacun d'entre eux est traduit par un **event** dans le **nœud** correspondant au type du *part*. Puis, dans le **nœud** traduction du *block* englobant, on inscrit les relations de synchronisation (**sync**) traduisant les enchaînements de *messages*. Soit e1 un *message* entrant sur un *part* p1 et e2 le message sortant vers un *part* p2 juste après le traitement de e1, alors la synchronisation s'écrit : <p1.e1, p2.e2>. Soit maintenant e3 sortant de p1 et allant vers un *part* p3 dans le même pas de temps que e2, si e2 et e3 sont parallèles (embranchement *par*) la synchronisation devient : <p1.e1, p2.e2, p3.e3>. Si e2 et e3 sont mutuellement exclusifs (embranchement *alt*), la synchronisation notée est : <p1.e1, p2.e2?, p3.e3?> = 1 ; le « ? » indique que e2 et e3 sont optionnels, « = 1 » indique qu'un et un seul des deux événements est produit à la suite de e1. Cette dernière construction est valable en AltaRica mais banni en AltaRica DF pour permettre l'exploitation du modèle dans la recherche de scénarios de défaillance. Il faudra donc choisir de retranscrire seulement une des deux suites d'événements suivant le scénario que l'on souhaite vérifier avec le modèle AltaRica DF. Dans ce cas, on ne peut donc que proposer des alternatives de traduction à l'analyste SdF. Enfin, des gardes peuvent apparaître sur l'envoi des *messages*, dans ce cas si elles sont valides, elles seront recopiées en amont des transitions déclenchées par l'événement symbolisant le *message*.

Les AD ne sont exploitables pour la traduction que si les *actions* et *activités* sont alloués aux composants physiques. Ce que l'on cherche à traduire par une transition est le déclenchement d'une *action*, qu'elle soit l'appel d'une *operation* ou d'une sous *activité*. De façon similaire au traitement des SD, on créera un événement par *operation* ou *action* possédée par un type de *part*. On notera ensuite, pour la mise au point des synchronisations, les *actions* précédentes et suivantes. Des gardes peuvent également apparaître dans les AD, elles doivent être également prises en compte. Pour l'expression des affectations réalisées à la suite d'une *operation* (issue d'un SD ou AD) ou d'une *action*, il est nécessaire d'identifier les éléments de modèles associés (*allocated*) à ces artefacts. S'il s'agit de diagrammes paramétriques explicitant le déroulement du traitement, on peut retranscrire ses contraintes dans l'affectation. Si on considère une *operation*, le traitement peut être indiqué par le corps de la fonction ainsi exprimée. Si aucun de ces artefacts n'est associé, l'affectation devra être précisée par l'analyste de SdF.

III.3.2.2 Identification des assertions

Les assertions sont utilisées pour modéliser le comportement interne des composants. Elles modélisent notamment, la détermination des variables de sortie et les relations statiques dans le sens où elles sont valables quels que soient les événements se produisant. Ces assertions proviennent donc premièrement de la traduction des *constraint properties* des *blocks*, qui par définition modélisent ce type de relations. Ces contraintes relient uniquement des *value properties* du *block*, il faut donc interpréter la formule qu'elles contiennent et les connections de leurs *constraint parameters* présentes dans les diagrammes paramétriques. Néanmoins, rien ne peut prouver que ces contraintes expriment des relations vérifiées à n'importe quel instant de la vie du composant. Cela dépend une fois de plus de la méthodologie employée. Cette analyse permet de fournir uniquement des suggestions de traduction comme nous donnons les relations à analyser lors de la création d'AMDEC préliminaire. Lors de l'écriture du modèle AltaRica DF final, le concepteur devra se poser la question suivante : la contrainte est-elle toujours vraie ? Si la réponse est oui, elle exprime une assertion. Si la réponse est non, plusieurs cas de figure se présentent : la contrainte est erronée, une condition a été oubliée lors de son écriture ou la contrainte est associée à un événement. Dans les deux premiers cas, cette étape permet de vérifier le modèle et d'apporter une correction en coordination avec le concepteur. Dans le dernier cas, si une allocation à un événement existe, la contrainte est traduite dans l'affectation correspondante. Sinon, il faut déterminer l'événement en question, créer l'allocation dans le modèle SysML et traduire le motif ainsi formé.

Comme pour la création des transitions, l'analyse des *statemachines* est très précieuse lorsqu'il s'agit d'identifier les assertions. Les assertions sont modélisées dans les *statemachines* par les *actions d'état*. La traduction s'effectue en recopiant l'expression des *actions d'état* de type *do* et en proposant comme condition à ces expressions (condition **if** dans l'assertion) les valeurs prises par les variables d'état dans l'état contenant les *actions* traduites. La figure III.2 complète l'exemple de la figure III.2 en présentant la traduction des assertions. Par exemple, dans l'état « Opened » (caractérisé par la variable d'état « opened » fixée à « True ») l'action d'état fixe la variable « Out » égale à la variable « In ». Ceci est traduit par les instructions : « If opened = True Out = In ; ».

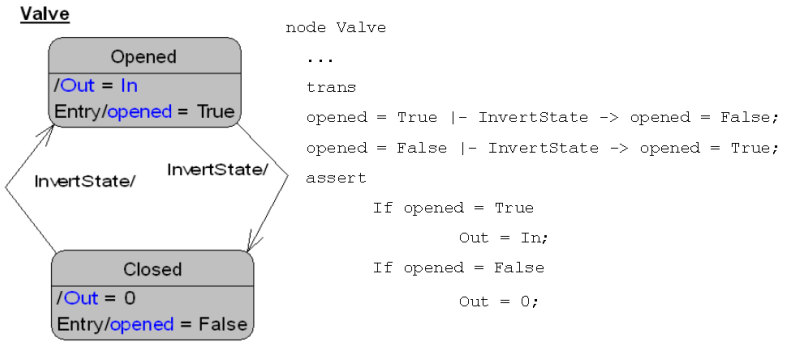


Figure III.2 *Exemple simple de traduction d'une statemachine (suite)*

La réalisation de ces diverses identifications réutilise directement les listes d'éléments en relations autour des *parts*, définies lors de la constitution de l'AMDEC préliminaire. Ce point est particulièrement intéressant pour renforcer la cohérence entre modèles d'ingénierie et d'analyses de SdF et pour gagner un temps précieux lors de la construction de ces derniers. Cependant, on peut constater que cette assistance à la création du modèle AltaRica est très fortement tributaire de l'emploi d'une méthode de saisie du modèle SysML favorisant l'automatisation de l'étude. On peut tout de même observer que les aménagements nécessaires permettent de constituer une méthode de réalisation du modèle plus formelle et expressive, notamment en insistant sur la création de liens entre entités et sur l'utilisation d'artefacts ciblés pour la transcription des notions intervenant dans la définition d'un système. Néanmoins, en l'absence de méthodologie, il reste possible de proposer différentes possibilités de traduction que l'analyste SdF devra valider ou adapter.

III.3.3 Ajout du comportement dysfonctionnel

Afin de finaliser l'analyse de SdF du système, il est nécessaire d'adjoindre aux éléments précédents, la description de leur comportement dysfonctionnel. La recherche de cette information est l'objet de l'AMDEC réalisée à l'étape précédente de MéDISIS. Un des objectifs de MéDISIS est d'organiser le transfert et la réutilisation des connaissances glanées au cours de l'AMDEC. C'est pourquoi, il est important que pour chaque MdD identifié la BCD soit correctement mise à jour. Dans le cas contraire, l'analyste SdF doit reprendre à la main les résultats de l'AMDEC et les transcrire lui-même en AltaRica DF. L'ajout de ce comportement dysfonctionnel nécessite d'introduire pour chaque **nœud**, de nouveaux états, de préciser les transitions vers les MdD et de définir les assertions vérifiées dans les états de panne. Dans cette section, nous présentons l'organisation de la BCD enregistrant les résultats de l'AMDEC, puis nous évoquons l'étape de traduction de ces informations vers AltaRica DF et leur intégration au modèle AltaRica DF fonctionnel déjà créé.

III.3.3.1 Organisation de la BCD pour la génération de modèles AltaRica DF

Nous reprenons la description de la BCD définie pour MéDISIS, présentée au paragraphe II.2.5.2. Dans ce paragraphe nous avions présenté une vue de la BCD telle qu'elle est abordée pour les études qualitatives de SdF, nous présentons ici son complément par une expression des quantifications des MdD. Nous détaillons les constructions employées pour rendre compte des probabilités d'apparition des défaillances et pour préciser leur effet sur les variables du composant. Nous donnons sur la figure III.3 le métamodèle complet de la BCD rendant compte de toutes les connaissances caractérisant et expliquant un MdD.

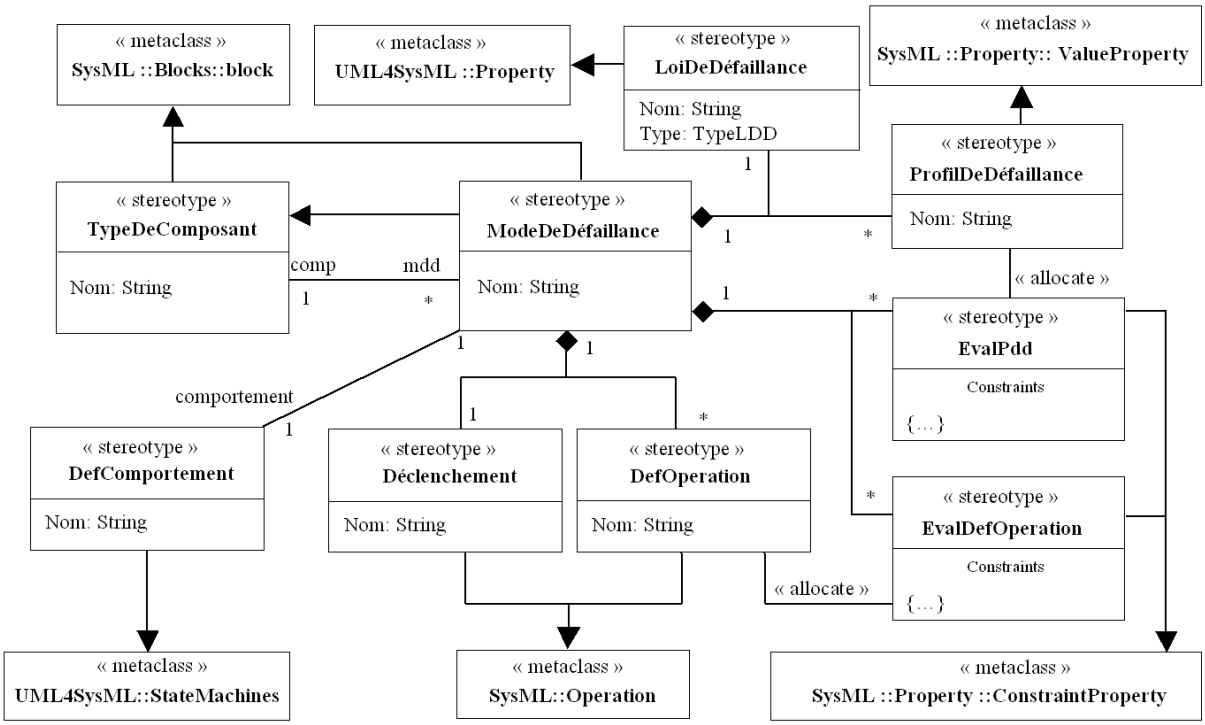


Figure III.3 *Métamodèle de la BCD*

Cette organisation de la BCD fait partie de la méthodologie définie par MéDISIS, elle propose donc une utilisation particulière de SysML. Un MdD possède ici un ensemble de propriétés permettant de le caractériser. Un MdD représente dans la BCD un cas de dysfonctionnement pour un type de composant donné. Si l'AMDEC a permis d'identifier trois MdD pour un type de composant, il y aura trois éléments de stéréotypes « ModeDeDéfaillance » associés au *block* concerné dans la BCD. Un élément « ModeDeDéfaillance », que nous assimilerons ici à la notation MdD, possède tout d'abord des *operations* spécifiques en plus de celles dont il hérite du type de composant. Il dispose des redéfinitions dysfonctionnelles des *operations* du composant, comme nous l'avons présenté au II.2.5.2, mais également d'une *operation* répondant au stéréotype d'*operation* intitulé « Déclenchement ». Cette dernière est utilisée pour modéliser la survenue du MdD, notamment dans la *statemachine* modélisant le passage au comportement défaillant et les actions réalisées dans cet état. Une propriété de stéréotype « LoiDeDéfaillance » est allouée à cette *operation* (le lien n'est pas figuré sur le métamodèle afin de clarifier la figure) et est utilisée pour spécifier le type de loi de probabilité utilisé pour modéliser la survenue de la défaillance. Dans cette relation, l'*operation* est le membre *from* de l'allocation et la propriété « LoiDeDéfaillance » est l'extrémité *to*. Le nom de cette loi est donné par l'attribut « Type » de classe « TypeLDD » que nous avons définie comme une énumération des lois couramment utilisées en SdF (loi exponentielle, de Weibull, de Poisson, etc.). Afin de caractériser totalement ce mécanisme de déclenchement, il est nécessaire de connaître les paramètres de ces lois. C'est pourquoi, le MdD possède également des propriétés de stéréotype « ProfilDeDéfaillance » modélisant les paramètres intervenant dans la loi de probabilité de défaillance. Ces derniers lui sont alloués par une relation *allocate* (la loi est le membre *from*, les paramètres sont les extrémités *to*). Les valeurs de ces paramètres dépendent des caractéristiques physiques des composants considérés. Les recueils de MdD utilisés dans l'industrie proposent des relations de calcul de ces paramètres en fonction des paramètres du système et de son environnement. Pour noter ces relations dans la BCD, nous proposons d'ajouter des *constraint properties* au MdD, stéréotypées par « EvalPdd », exprimant les relations entre les *value properties* correspondant aux caractéristiques physiques du composant et

celles exprimant le profil de défaillance. Lorsque ces expressions font appel aux paramètres environnementaux, il est nécessaire de les déclarer comme propriétés faisant partie du profil de défaillance. Les paramètres de stéréotype « ProfilDeDéfaillance » couvrent également les variables liées au MdD qui expriment ces caractéristiques, comme son ampleur et les phénomènes qu'il induit. Dans l'exemple d'une conduite et de son MdD « Fuite », il est possible d'ajouter une variable spécifiant l'amplitude de la fuite, cette variable fera partie du profil de défaillance. Ces variables bénéficient également de *constraint properties* allouées à leur détermination.

Nous avons également spécifié sur ce métamodèle la construction que nous avions évoquée dans le paragraphe II.2.5.2.2 concernant l'expression des *operation*s dysfonctionnelles. Le MdD possède une *constraint property* stéréotypée par « EvalDefOperation » pour chaque redéfinition des *operations* modifiées par le MdD, représentant les nouvelles relations entre les paramètres du composant créées par la défaillance. Les deux entités sont également liées par une relation d'allocation.

III.3.3.2 Reprise de l'exemple de BCD relative à une vanne

Nous proposons maintenant de compléter l'exemple de la BCD créée pour une vanne au paragraphe II.2.5.2.2. Dans cet exemple (résumé par les figures II.19, II.20 et II.21), une vanne possède deux MdD nommés « VanneBloquée » et « FuiteVanne ». Dans la première partie de cet exemple, nous avons déjà illustré l'utilisation des *operations* redéfinies et leur expression par une contrainte. La figure III.4 rassemble les éléments ajoutés au MdD « FuiteVanne », il s'agit des variables relatives au profil de défaillance, de l'*operation* de déclenchement, de la contrainte de calcul des éléments du profil et de la mise à jour de la *state-machine* du composant.

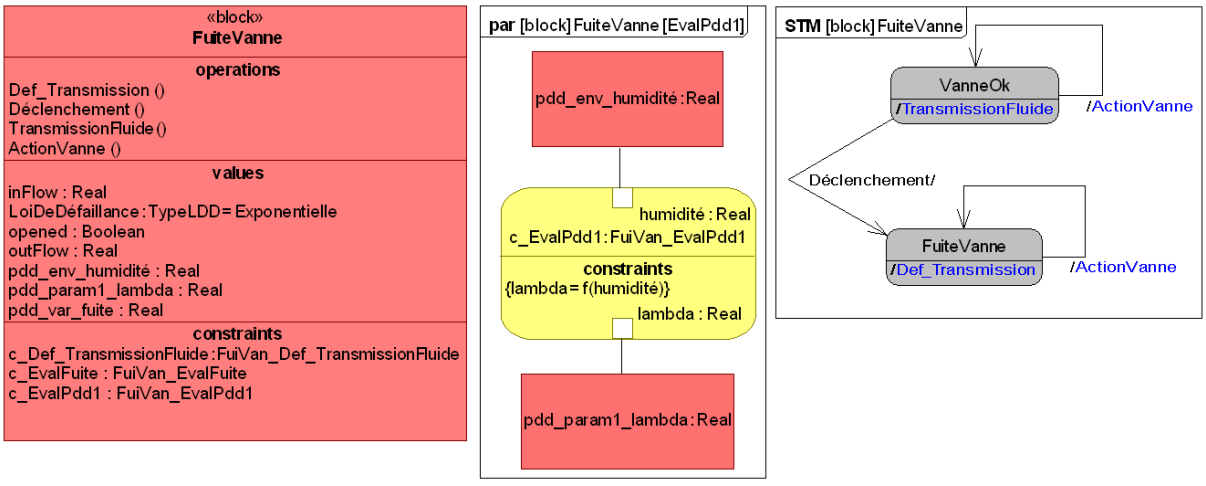


Figure III.4 *Développement du MdD « FuiteVanne »*

Nous avons donc ajouté l'*operation* « Déclenchement » nous permettant de caractériser la survenue du MdD. La loi de défaillance, qui lui est allouée, a pour valeur initiale « Exponentielle », indiquant le type de loi à utiliser pour la simulation ou les calculs de fiabilité. Cette loi a pour paramètre le célèbre « λ », fixé par la variable « pdd_param1_lambda » (dans le cas d'un loi de Weibull, deux paramètres seraient déclarés). Nous avons choisi d'utiliser des préfixes explicites en plus des prototypages pour identifier rapidement les différents types de variables employées dans la description des MdD. Le préfixe « pdd » indique que la variable est relative au profil de défaillance, « param1 » nous indique qu'il s'agit du

premier paramètre utilisé par la loi à laquelle il est alloué. La valeur de ce λ dépend, dans le cas d'une vanne, de différentes caractéristiques du système et de son environnement. Nous avons illustré ici que le λ peut dépendre du taux d'humidité de l'environnement dans lequel est implantée la vanne. Ainsi un élément « pdd_env_humidité » est ajouté au profil de défaillance (le préfixe « pdd_env » indiquant un paramètre d'environnement) et une *constraint property* de calcul du λ nommée « c_EvalPdd1 » est déclarée. Cette contrainte est illustrée et liée aux paramètres du MdD dans le diagramme paramétrique « par [Block] FuiteVanne [Evalparam1] ». Nous présentons ici un exemple très succinct de calcul d'un paramètre du profil de défaillance ; dans les cas réels beaucoup plus d'éléments entreront en jeux dans leur détermination. Les équations portées par les contraintes peuvent être reprises des bases de connaissances sur les défaillances, telles que : Mil HDBK 217f, FIDES, OREDA (citées § II.3.5.2).

La contrainte ainsi déclarée permet de quantifier la défaillance du composant et caractérise le déclenchement du MdD à travers l'*operation* « Déclenchement ». Nous pouvons donc compléter la *statemachine* de la figure II.21 en spécifiant comme déclencheur de la transition de l'état normal à l'état défaillant l'*operation* « Déclenchement ». Cette *statemachine* et les artefacts alloués à ses éléments définissent complètement le comportement du MdD. Le passage au mode défaillant est exprimé par une *operation* à laquelle sont alloués le type de loi utilisée, ses paramètres et la manière de déterminer leurs valeurs. Le comportement dans le mode défaillant est spécifié par des *operations* exprimant les traitements effectués grâce aux différentes contraintes et variables qui leur sont allouées. La variable donnant l'importance de la fuite est ainsi spécifiée et ajoutée au profil de défaillance avec le nom « pdd_var_fuite ». Une *constraint property* lui est allouée pour déterminer sa valeur, de même les *operations* défaillantes possèdent leurs propres *constraint properties*. Enfin l'héritage du *block* définissant le comportement fonctionnel permet de conserver les variables de fonctionnement et les comportements inchangés par la défaillance.

III.3.3.3 Traduction du comportement dysfonctionnel

La traduction en AltaRica DF des informations contenues dans cette base est le dernier pas dans la création du modèle d'analyse de SdF. La base utilisant des spécifications SysML, la traduction reprend les mécanismes définis aux paragraphes III.3.1 et III.3.2. Cependant, l'utilisation d'une méthode et d'un métamodèle spécifiques à notre BCD, permet d'obtenir une traduction automatisable.

Nous considérons que l'étape de création du modèle fonctionnel AltaRica DF est réalisée et que le découpage en **nœuds** suit la déclaration des *blocks* du modèle SysML. L'ajout des MdD se fait pour chaque *block* de la façon suivante :

- Déclaration d'une variable d'état indiquant l'état de défaillance courant du composant.
- Déclaration des événements déclencheurs des MdD.
- Ajout des variables propres au MdD.
- Ajout de transition vers chacun des états défaillants.
- Séparation des éléments de comportement (transitions et assertions) par ajout de gardes sur l'état de défaillance courant.
- Saisie des assertions dysfonctionnelles.

Nous faisons l'hypothèse que les MdD apparaissent de façon disjointe pour un composant. Une variable d'état de défaillance est ainsi déclarée pour chaque *block*, son type est une énu-

mération (**domain**) regroupant les MdD recensés pour le composant. Pour l'exemple de la vanne, un domaine est déclaré : `domain MdDVanne {ok, FuiteVanne, VanneBloquée}` ; une variable d'état est ajoutée : `state etatVanne : MdDVanne`. La valeur initiale est évidemment fixée au bon fonctionnement : `init etatVanne = ok ;`. Un événement est déclaré pour chaque MdD, en reprenant le nom de l'*operation* de déclenchement et le nom du MdD afin de suivre les noms de la BCD et d'éviter les possibles conflits de nommage. On obtient le code : `event Déclenchement_FuiteVanne, Déclenchement_VanneBloquée ;`. De même, les variables propres au MdD, spécifiées dans le profil de défaillance, sont ajoutées. Pour le MdD « FuiteVanne », il s'agit du paramètre « pdd_var_fuite ». Ensuite, les transitions vers les différents états défaillants sont créées ; pour la vanne on obtient les deux transitions suivantes :

```
etatVanne = ok |- Déclenchement_FuiteVanne -> etat = FuiteVanne ;
etatVanne = ok |- Déclenchement_VanneBloquée -> etat = VanneBloquée ;
```

La BCD présente les aspects quantitatifs qu'il est nécessaire de traduire afin d'exploiter le modèle en simulation. L'élément « LoiDeDéfaillance » donne le type à considérer, la traduction vers AltaRica DF est donc simplifiée, seule l'évaluation des paramètres de la loi peut se montrer complexe. En effet, il faut que les valeurs des paramètres intervenant dans la contrainte de calcul des paramètres de la loi de défaillance soient connues. Si ces valeurs sont disponibles, le traducteur résout la contrainte, sinon la main est laissée à l'expert de SdF pour quantifier le paramètre. Dans le cas de la vanne, la loi de déclenchement d'une fuite peut être ainsi spécifiée : `extern law <event Déclenchement_FuiteVanne> = exponential (lambda) ;` (la valeur de lambda étant à préciser).

Il est ensuite nécessaire de composer les différents comportements accessibles pour le système en fonction des MdD. Pour cela, nous proposons de préfixer chaque bloc d'assertions par une condition sur l'état de défaillance du système. Ainsi, on redéfinira le comportement du composant sous chaque hypothèse de défaillance. Dans l'exemple que nous suivons, la condition à écrire est : `If etatVanne == ok`, pour les assertions fonctionnelles (déjà obtenues) et : `If etatVanne == FuiteVanne`, pour les assertions relatives au MdD d'une vanne ayant des fuites. Le modèle AltaRica DF est ensuite complété en traduisant les états de la *statemachine* reflétant le comportement dans le MdD traduit. Les *operations* et transitions afférentes à l'état dysfonctionnel sont ainsi retranscrites comme décrit au paragraphe III.2.2. La méthode de saisie employée dans MéDISIS autorise une analyse automatique de ces modèles. En effet, nous avons spécifié que chaque *operation* serait explicitée par une contrainte montrant le fonctionnement lié à cette dernière. De plus, pour un degré optimum d'intégration avec AltaRica DF, il est possible d'utiliser ce langage pour spécifier les contraintes. Cette solution est justifiée car la personne mettant en place la BCD est l'expert SdF, lui-même utilisateur du modèle AltaRica DF créé. Cette solution a été employée dans l'exemple de la vanne pour les contraintes de la figure II.20. L'exemple finalisé de la vanne donne le code suivant :

```
domain MdDVanne {ok, FuiteVanne, VanneBloquée} ;

node Vanne
      state
            opened : bool ;
            etatVanne : MdDVanne ;
            pdd_var_fuite : integer;
      flow
            inFlow : integer : in ;
            outFlow : integer : out ;
      event
            ActionVanne, Def_ActionVanne, Déclenchement_FuiteVanne ;
```

```
        Déclenchement_VanneBloquée  ;
    init
        opened = true ;
        etatVanne = ok ;
        pdd_var_fuite = X1 ;
    trans
        etatVanne = ok |- ActionVanne -> opened := ~opened ;
        etatVanne = ok |- Déclenchement_FuiteVanne -> etat = Fuite-
        Vanne ;
        etatVanne = ok |- Déclenchement_VanneBloquée -> etat = Vanne-
        Bloquée ;
        etatVanne = FuiteVanne |- ActionVanne -> ouverte := ~opened ;
        etatVanne = VanneBloquée |- Def_ActionVanne -> ;

    assert

        if etatVanne = ok
            if opened
            outFlow = inFlow ;
            else
            outFlow = 0 ;

        if etatVanne = FuiteVanne
            if opened & inFlow > pdd_var_fuite
            outFlow = inFlow - pdd_var_fuite ;
            else
            outFlow = 0 ;

        if etatVanne = VanneBloquée
            if opened
            outFlow = inFlow ;
            else
            outFlow = 0 ;

    extern
        law <event Déclenchement_FuiteVanne> = exponential (X2) ;
        law <event Déclenchement_VanneBloquée> = exponential (X3) ;

edon
```

La partie dysfonctionnelle du code est principalement créée grâce à la traduction de la *statemachine* des MdD. Sur ce squelette, il reste à l'expert à définir les valeurs que nous avons notées X1, X2 et X3 respectivement l'amplitude de la fuite et les taux de défaillance. Il est important de noter que les formules de calcul de ces valeurs sont précisées sous la forme de contraintes dans la BCD. L'expert a donc à sa disposition tous les éléments nécessaires à la création du modèle d'analyse de SdF.

Nous venons de montrer que les éléments nécessaires à la création du modèle d'analyse de SdF sont présents dans les descriptions SysML et qu'une traduction largement automatisable peut être mise en place entre les deux représentations. Ceci est fait dans une volonté de cohérences entre les modèles et pour la recherche d'efficacité et le réemploi des études spécifiques de SdF. La traduction du modèle fonctionnel reste très dépendante de la qualité du modèle et des techniques de modélisation employées par le concepteur. L'ajout de la partie dysfonctionnelle à partir de la BCD, centralisant et réalisant le pivot dans la gestion des informations dysfonctionnelles, se montre très efficace grâce à la méthode adoptée pour l'écriture de la BCD. Cette écriture se concentre sur la couverture de tous les éléments intervenant dans la description de l'axe dysfonctionnel et répertorie ainsi tous les éléments nécessaires à l'établissement du modèle final d'analyse et de validation de la SdF.

III.4 Conseils de modélisation pour la génération de modèles dysfonctionnels

Lors de cette analyse de la traduction de modèles SysML en AltaRica DF, nous avons considéré que le modèle fonctionnel traduit respectait uniquement la spécification SysML. Il est apparu que cette traduction pouvait être rendue complexe, indécise et incomplète en fonction des artefacts utilisés. L'intervention de l'expert doit alors dépasser la simple initialisation des valeurs comme c'est le cas pour la traduction de la BCD. Nous souhaitons donc, dans ce paragraphe, formuler plusieurs conseils de modélisation SysML améliorant l'obtention d'un modèle plus facilement analysable et traduisible. Cette volonté est en parfaite adéquation avec les objectifs de l'ISBM, dont un des objectifs principaux est l'interaction optimum des modèles employés. Comme nous l'avons expliqué au II.3.3, les techniques à disposition définissant une méthodologie de saisie de modèles SysML sont d'imposer l'utilisation d'artefacts ciblés de la spécification SysML, l'emploi de stéréotypes et la définition de règles de nommage des éléments déclarés. Dans ce paragraphe, nous évoquons des règles de modélisation utilisant ces trois possibilités. Un premier groupe de règles est formulé pour la description de la structure et un second est défini dans le but d'encadrer la modélisation du comportement du système. Les règles présentées ici viennent compléter celles énoncées en II.3.3.

III.4.1 Règles de modélisation : axe structurel

Les points importants dans la description de la structure, en vue d'une exploitation avec AltaRica DF, sont d'isoler les différentes classes des variables employées (état, flux), les types de ces variables et les connexions entre les composants. Il faut également que le modèle permette une rapide identification des composants physiques du système et de ses limites. Le but recherché est de permettre une interprétation non ambiguë des artefacts ainsi qu'une sémantique proche de celle utilisée en AltaRica DF. Nous énonçons les règles suivantes pour la déclaration de la vue architecturale du système :

1) Utiliser le stéréotype « V_Etat » pour déclarer les *value properties* correspondant à des variables d'état.

2) Utiliser le stéréotype « V_Flux » pour déclarer les *value properties* correspondant à des variables de flux.

3) Déclarer trois *value types* correspondant respectivement à la notion d'entier, de réel et de booléen.

4) Définir les types des variables avec des *value types*. Lier par une relation d'héritage ces *value types* à un des trois types « entier », « réel » ou « booléen ». Une exception est admise pour les *value types* représentant une énumération, qui ne doivent pas posséder ces liens d'héritage.

5) Déclarer un *flow port* par variable de flux identifiée. Définir la direction de ce port pour exprimer le sens de passage du flux. Utiliser le même type que la variable de flux pour spécifier le *flow port*. Utiliser le même nom que la variable pour le port en préfixant l'expression par « p_ ».

6) Déclarer une relation d'allocation entre chaque couple port/variable de flux ainsi formé. Le port est l'extrémité *allocated from*, la variable est l'*allocated to*.

7) Identifier les *block* décrivant un concept de composant utilisé par le système, par le stéréotype « C_Système ».

8) Utiliser uniquement les *blocks* ainsi stéréotypés pour spécifier les *parts* des *blocks* « C_Système ».

9) Réaliser un IBD pour chacun de ces *blocks*, modélisant les connexions entre *parts*. Utiliser pour cela uniquement des connecteurs passant par les *flow ports*. Vérifier la compatibilité des types de ports ainsi connectés.

Les règles 1 et 2 sont à utiliser en conjonction avec les propositions de reconnaissance de la nature des variables proposées au paragraphe III.3.1.1, afin de déterminer quel stéréotype doit leur être affecté. L'utilisation de ces deux règles introduit une nouvelle classification dans les *value properties* des *blocks* SysML, permettant une meilleure appréhension de la nature des variables et donc une meileure compréhension de leur utilisation possible. Cette règle permet en outre l'automatisation de la traduction des variables vers le modèle AltaRica DF, l'exploitation des stéréotypes de variable fournissant un moyen simple d'identification de leur nature.

La règle 3 spécifie le besoin d'utiliser des *value types* de référence unifiant les types des *value properties* figurant dans les modèles. Une déclinaison plus complète de cette règle serait de créer un *package* réunissant les définitions du Système International d'unités. En spécifiant la règle 4, nous imposons l'utilisation systématique de ces types de référence afin d'obtenir un modèle cohérent et évidemment plus facilement traduisible en AltaRica DF sans pour autant perdre la liberté de déclarer des *value types* spécifiques.

La règle 5 encadre l'utilisation des *flow ports*, une nouvelle fois dans un souci de cohérence et d'expressivité du modèle. Une variable de flux est nécessairement en relation avec l'environnement, c'est pourquoi il est utile de spécifier explicitement cette interface par un *flow port* du modèle. Afin de spécifier cette mise en correspondance, les types des deux entités doivent être identiques, nous préconisons également d'adopter une convention de nommage permettant de reconnaître facilement les couples port/variable. La règle 6 renforce cette association en demandant la déclaration d'une relation d'allocation venant définir explicitement la correspondance. Ces deux règles permettent aussi une meilleure traduction du modèle SysML en AltaRica DF pour lequel ports et variables de flux sont modélisés par une unique entité. En effet, avec une telle règle de construction, il est aisé de remonter de la variable de flux au port et d'en déduire alors la direction.

Les règles 7 et 8 participent à une clarification du modèle SysML, permettant ainsi de distinguer les artefacts relatifs aux composants d'implémentation et les constructions purement figuratives. Le respect de ces règles garantie la cohérence dans la constitution des *blocks* physiques, en forçant la constitution des composants physiques par l'utilisation d'agrégats de composants physiques. D'un point de vue de la traduction, il apparaît que les *block* à traduire seront aisément identifiables grâce au stéréotype « C_Système ».

La règle 9 impose la représentation des liens structurels entre composants, ainsi que l'utilisation d'interfaces comme vecteurs des interactions. Ces liens entre ports alloués aux

variables de flux des différents composants permettent d'obtenir simplement la connexion des **nœuds** dans le code AltaRica DF.

III.4.2 Règles de modélisation : axe comportemental

Une description organisée de la structure du système est obtenue en appliquant les règles proposées ci-dessus. Ce modèle est formalisé pour accueillir la définition du comportement du système. Nous présentons maintenant des règles explicitant les constructions à employer pour modéliser le comportement individuel des composants, ainsi que les interactions entre eux formant le comportement global du système.

1. Modéliser tout processus, traitement ou événement modifiant les variables détenues par le composant par une *operation*.

2. Allouer à chaque *operation* une *constraint property* exprimant le traitement réalisé. Si la formulation ne peut être établie, indiquer uniquement les variables participant à l'*operation*. La *constraint property* est l'extrémité *allocated from*, l'*operation* est l'*allocated to*. Réaliser le Parametric Diagram exprimant la *constraint property*.

3. Réaliser la *statemachine* présentant l'utilisation des *operations*. Privilégier les déclencheurs de transition entre états pour mentionner les *operations* modifiant les variables d'état et les actions d'état (*do*) pour celles déterminant les variables de flux.

4. Réaliser des SD indiquant les enchaînements de déclenchement des *operations*.

La première règle est née de la volonté de centraliser, sous l'utilisation d'un unique artefact, la déclaration des phénomènes assimilables à l'expression du comportement, c'est-à-dire modifiant les variables détenues par les composants. En SysML, une *operation* peut être utilisée pour décrire un traitement réalisé en continu ou intervenant sporadiquement, cette notion est donc suffisamment flexible pour modéliser un ensemble varié de comportement. L'écriture du modèle est ainsi simplifiée, ce dernier devient donc plus lisible. La traduction vers AltaRica DF profitera également de cette mesure en considérant la règle 3.

Le comportement des *operations* est ensuite modélisé à l'aide de *constraint properties* leur étant spécialement allouées. Les contraintes sont décrites par des expressions mathématiques logiques ou algorithmiques reflétant le traitement réalisé par les *operations*. Ceci constitue un moyen simple de description du comportement, qui sera traduisible en AltaRica DF, si l'expression indiquée dans la contrainte est exprimable dans le langage. Cette règle 2 peut être complétée, dans le cadre d'un modèle destiné à la traduction AltaRica DF, par la nécessité de respecter la syntaxe AltaRica DF pour l'expression des contraintes.

La règle 3 formule des conseils de modélisation pour la réalisation de *statemachines* formalisant le comportement des composants. L'usage de *statemachines* permet d'illustrer les différentes *operations* déclarées intervenant dans le comportement des composants. Le placement des *operations* dans la *statemachine* permet de déduire leur traduction en AltaRica DF. En suivant les instructions du paragraphe III.3.2, il est possible de déterminer si elles donneront naissance à une transition ou à une assertion.

La dernière règle force le concepteur à déterminer la synchronisation des comportements inter-composants par la description des séquences d'activation des *operations* à travers un SD. Ce diagramme permet de compléter le modèle AltaRica DF par la synchronisation inter-composants des déclenchements de traitements.

III.5 Conclusion

Dans ce chapitre, nous avons illustré le déploiement de la seconde phase de MéDISIS. Toujours dans le but de placer l'étude de la Sûreté de Fonctionnement au cœur du processus d'Ingénierie Système Basée sur les Modèles des systèmes à fortes contraintes de Sûreté de Fonctionnement, nous avons étudié la création des modèles formels dédiés aux analyses de Sûreté de Fonctionnement. Afin de tirer le meilleur parti de l'approche basée sur les modèles, la génération de ces modèles spécifiques doit se faire en concordance avec le modèle central de l'Environnement de Développement Système et doit pouvoir s'appuyer sur l'exploitation des bases de connaissances gérant les informations sur le comportement dysfonctionnel. Ainsi, le modèle dédié à la Sûreté de Fonctionnement doit naître de l'union de la définition fonctionnelle du système exprimée par le modèle SysML et de la Base de données des Comportements Dysfonctionnels organisant les connaissances rassemblées lors des premières phases de MéDISIS.

Nous avons choisi d'approfondir cette étape en analysant la problématique de l'assistance à la génération de modèles utilisant AltaRica Data Flow. Ce langage très largement utilisé dans la communauté des experts de Sûreté de Fonctionnement, permet de créer des modèles analysables formellement par différents outils. Ces modèles sont notamment exploitables pour la recherche de scénarios de défaillances ou l'estimation de la fiabilité globale d'un système. Ce langage est donc utilisable pour un panel très complet d'analyses de Sûreté de Fonctionnement et justifie donc pour nous le déploiement d'efforts de traduction du modèle central SysML vers son formalisme.

Dans ce chapitre, nous avons apporté trois contributions principales que sont la présentation de la traduction de modèles SysML en AltaRica Data Flow, la formulation de règles de modélisation pour la mise en place d'une méthodologie de saisie des modèles SysML et la description complète de la Base de données des Comportements Dysfonctionnels développée pour MéDISIS. La traduction est réalisée en deux phases, la première est la retranscription en AltaRica Data Flow de la représentation fonctionnelle SysML, la seconde est l'exploitation de la Base de données des Comportements Dysfonctionnels pour adjoindre le comportement dysfonctionnel au modèle complet d'analyse Sûreté de Fonctionnement. La présentation de la traduction est faite sans a priori de méthodologie de modélisation. Néanmoins, nous montrons que cette traduction peut être rendue plus efficace en considérant des règles de saisie du modèle SysML. Nous donnons en fin du chapitre des règles pour l'écriture des modèles SysML permettant une traduction plus efficace vers AltaRica Data Flow et se montrant aussi cohérente et bénéfique dans le processus d'Ingénierie Système Basée sur les Modèles.

Nous avons détaillé le métamodèle de la Base de données des Comportements Dysfonctionnels, modélisant et organisant les aspects relatifs aux Modes de Défaillance des composants. Cette Base de données des Comportements Dysfonctionnels est décrite en SysML et se nourrie des résultats obtenus par l'AMDEC des systèmes. Les éléments enregistrés couvrent les aspects qualitatifs des Modes de Défaillance en décrivant comment ces derniers modifient le comportement des composants. Ceci n'exclut pas de montrer la dimension quantitative de ces phénomènes pour lesquels il est possible de spécifier les lois d'apparition, ainsi que la détermination de leurs paramètres. Un ensemble d'allocations est créé entre ces artefacts afin d'obtenir une représentation cohérente et expressive des connaissances sur le comportement dysfonctionnel des systèmes.

L'étude des mécanismes de traduction entre les deux langages a permis de souligner les atouts d'une telle assistance à la création des modèles AltaRica Data Flow. Les bénéfices remarqués sont une plus grande rapidité de la préparation des analyses de Sûreté de Fonctionnement, une cohérence et une traçabilité améliorées entre les différents modèles déployés dans l'Environnement de Développement Système. Nous avons également pu mettre en avant la nécessité de l'utilisation d'une méthodologie précise de modélisation SysML afin de maximiser l'intérêt de la traduction.

Ce chapitre constitue une illustration de l'utilisation de MéDISIS sur un support AltaRica Data Flow. Cependant, il est important de noter que ce langage n'est pas la seule alternative utilisable pour le déploiement de la méthode. MéDISIS est utilisée pour optimiser le passage des analyses prospectives des risques à l'élaboration des modèles spécifiques de Sûreté de Fonctionnement. La construction et le formalisme utilisés pour la constitution de ces modèles sont extrêmement dépendants des besoins d'analyse ou de validation spécifiques aux cas d'étude. C'est pourquoi, MéDISIS peut également conduire à la réalisation d'autres types de modèles comme des Réseaux de Petri ou des processus de Markov. Dans le chapitre suivant, nous illustrons l'emploi de MéDISIS au cours du projet industriel CAPTHOM. Dans ce cadre, l'utilisation de MéDISIS a conduit à la réalisation d'un outil spécifique d'analyse et de validation.

Chapitre IV

Le Projet CAPTHOM : Analyse sous contraintes spatiales et contraintes de missions

Résumé du Chapitre IV :

Ce dernier chapitre présente les apports de l'utilisation de MéDISIS au cœur du projet CAPTHOM, dont la mission est le développement d'un capteur de présence innovant pour l'habitat individuel et les bâtiments tertiaires. Ce projet nécessite l'association de nombreux outils et dispositifs de développement, il tire donc nombre bénéfices d'une approche de type Ingénierie Système Basée sur les Modèles. Le modèle SysML de CAPTHOM est présenté avant d'exposer son analyse par MéDISIS. Le système est ainsi analysé par une AMDEC réalisée grâce aux techniques exposées dans le second chapitre. De cette AMDEC, les besoins d'analyse plus précis en terme de simulation du comportement des capteurs sont dégagés. L'AMDEC met en évidence les éléments à intégrer dans la solution de simulation de capteurs que nous avons spécialement développée pour le projet CAPTHOM. Ce logiciel, nommé SNOOPS, finalise l'analyse des produits développés dans CAPTHOM. SNOOPS permet la simulation de réseaux de capteurs multi-technologiques dans un environnement typique aux missions de CAPTHOM. La solution logicielle accompagne la conception des réseaux de capteurs jusqu'à l'optimisation de leur placement, grâce à l'emploi d'algorithmes génétiques. Enfin, on observe que l'approche prônée dans MéDISIS conduit ici, entre autres choses, à la création d'un logiciel intégré dans un Environnement de Développement Système. En effet, les résultats fournis par le simulateur viennent préparer les travaux connexes de développement liés au projet CAPTHOM dans sa globalité.

IV.1 Introduction

Dans cette dernière partie, nous illustrons sur une application particulière, l'usage de l'Ingénierie Système Basée sur les Modèles et de MéDISIS. Cette étude nous permet de mettre en évidence la flexibilité de MéDISIS, en montrant l'intégration d'un outil d'analyse spécifique au cas développé. Dans ce dernier chapitre, nous appliquons les techniques de l'Ingénierie Système Basée sur les Modèles au projet CAPTHOM. Nous utilisons la méthode MéDISIS et employons le langage SysML pour la modélisation du projet CAPTHOM. Nous aborderons deux aspects importants : comment l'usage de SysML et MéDISIS permet de soutenir un projet comme CAPTHOM ? Comment l'usage de SysML et MéDISIS, nous a permis de développer un outil de simulation dédié au projet et partie prenante de l'Environnement de Développement Système CAPTHOM ? Comme nous le verrons en première partie de ce chapitre, le projet procède d'un fort besoin d'innovation et peut donc profiter lors de sa réalisation, d'une approche utilisant des modèles SysML. Cette innovation passe principalement par une amélioration des performances des systèmes existants. Pour ce projet, différentes alternatives techniques sont à explorer et le développement de nouvelles solutions, notamment de fusion de données, sont nécessaires. Le projet possède des contraintes de Sûreté de Fonctionnement pour assurer un bon niveau de performance, ce qui nécessite l'emploi de MéDISIS. Nous expliquerons en quoi l'approche adoptée nous a assistée dans la formulation de la problématique, la mise en évidence de points clés du projet, l'identification et la mise en place d'une solution de simulation.

Nous débuterons par la présentation du projet CAPTHOM en insistant sur ses objectifs et l'expression des besoins spécifiques de tests et simulations. Nous détaillerons ensuite le modèle SysML développé pour le projet. Nous mentionnerons l'aide fournie par les différents diagrammes réalisés et leur apport dans la compréhension ou la prise de décision. Ce modèle est analysé en suivant MéDISIS. Nous présenterons les résultats de l'AMDEC du projet qui nous ont permis de développer le simulateur de réseaux de capteurs. Nous avons pu dégager, grâce à l'AMDEC, les objectifs de simulation et arrêter un niveau de détail de la modélisation. Le logiciel SNOOPS fait l'objet de la dernière partie de ce chapitre. Nous présenterons la modélisation utilisée pour les capteurs et leur environnement, puis nous évoquerons les possibilités de simulation de scénarios et la génération de résultats utilisables en fusion de données. Nous évoquerons en dernière partie l'optimisation de placement de réseaux de capteurs. Ce dernier service constitue une nouvelle « brique » de l'Environnement de Développement Système CAPTHOM, dédiée à l'équipement de bâtiments.

IV.2 Le projet CAPTHOM

Dans ce paragraphe nous présentons le projet CAPTHOM et ses objectifs. Nous insisterons sur les besoins de simulation inhérents au projet et notamment les besoins d'analyses nécessitant l'emploi de solutions dédiées, inclues dans l'EDS.

IV.2.1 Présentation et Objectifs

La recherche en domotique est depuis ces 10 à 20 dernières années, extrêmement active. De nombreux projets ont vu le jour dans le domaine de l'habitat intelligent. Les recherches actuelles portent sur l'amélioration de la gestion de l'énergie et du confort (éclairage, chauffage) ainsi que la sécurité et le maintien à domicile de personnes vulnérables (intrusion, détection de situations anormales). [Bonhomme 2008] présente une bibliographie complète des projets de domotique à travers le monde et recense plus de 35 initiatives de constitution d'habitats intelligents. Ces projets reposent sur la mise en place de systèmes de contrôle et commande de l'habitat. Leur mode de fonctionnement est constitué de trois activités principales : la prise d'informations sur le contexte, l'analyse et la prise de décision en rapport à ce contexte et la commande d'équipements venant faire évoluer le contexte. Le contexte regroupe toutes les variables d'intérêt pour la mission du système, cela comprend aussi bien la physique de l'environnement, que l'état des équipements de la maison ou la position des habitants. Les stratégies déployées par ces systèmes sont extrêmement dépendantes de la qualité de la prise d'informations.

Pour ces systèmes, l'information principale est bien souvent la connaissance de la présence des personnes. En effet, tous ces projets, qu'ils aient une dimension sécuritaire ou non, sont centrés sur l'humain. La détection de présence humaine est donc de prime importance pour le marché émergent de l'habitat intelligent. De nombreuses solutions de détection de présence humaine sont disponibles sur le marché, cependant aucune d'entre elles n'apporte de réelle satisfaction aux besoins réels de l'habitat intelligent. En effet, l'information recherchée est la présence d'humains immobiles ou non, parfois la localisation et l'activité des personnes. On constate que bien souvent les dispositifs disponibles sont incapables de répondre à ces critères et plus particulièrement à la détection de personnes immobiles [Gobeau 2006], [Belconde & Kratz 2008]. En reprenant et complétant l'étude des détecteurs de présence proposée par [Bonhomme 2008] nous fournissons le tableau IV.1 résumant les caractéristiques des solutions envisageables.

Tableau IV.1 *Comparatif des détecteurs de présence*

Types de détecteur	Information[25]						Coût	Remarques
	Détection de mouvement	Détection de personnes immobiles	Localisation	Activité	Posture	Discrimination		
Infrarouge passif	+	-	+/-	-	-	-	Faible	Possible de déduire un sens de passage. Triangulation possible.
Barrière IR/US/ lumineuse	+	+/-	-	-	-	-	Faible	Possible de réaliser un comptage. Pas de détection dans la scène.
Ultrason	+	+	+/-	-	-	-	Faible	Pas de discrimination entre les différents stimuli de détection.
Tapis détecteurs de pression	+	+	+	+/-	+/-	+/-	Elevé	Installation coûteuse et très intrusive.
Micro-capteur Infrarouge	+	+	+/-	-	-	-	Faible	Faible portée.
Caméra Vidéo	+	+	+	+	+	+	Faible à Moyen	Luminosité minimum nécessaire. Algorithmes maîtrisés, coût en nette baisse.
Caméra Infrarouge	+	+	+	+	+	+	Très élevé	Pas de contraintes de luminosité. Prix dissuasif.

Actuellement l'immense majorité des détecteurs utilisés pour la détection de présence humaine sont les capteurs infrarouges passifs [Gobeau 2006]. Ces derniers sont incapables de détecter un humain immobile mais restent la solution privilégiée car la moins coûteuse. Cependant, leur utilisation n'est plus satisfaisante pour les projets ambitieux d'habitats intelligents où la fiabilité et la richesse de l'information sont primordiales.

Dans ce contexte, le projet CAPTHOM, mis en place au sein du pôle de compétitivité S2E2, a pour but de développer un nouveau capteur de présence humaine à destination des solutions d'habitats intelligents. Pour cela, le projet se dote d'une phase importante d'évaluation des solutions existantes (tableau IV.1). Le projet intègre des travaux visant aussi bien l'amélioration des composants existants, que la mise au point de traitements de leur signal plus efficaces. Le but est de créer une nouvelle solution venant remplacer l'existant, à savoir les détecteurs pyroélectriques (PIR) fonctionnant sur le principe des détecteurs infrarouges passifs. Le nouveau capteur doit venir corriger les défauts des PIR tout en conservant ses performances comme le temps de réponse ou la portée de détection. Le nouveau système doit également arriver sur le marché avec une solution à bas coût et un niveau d'encombrement similaire aux détecteurs PIR. Il doit être porteur d'une réelle innovation, et être capable de détecter une personne humaine immobile sur une zone comparable à une pièce d'habitation, en garantissant une très grande fiabilité de détection (détecter à coup sûr, éviter les fausses détections). L'ensemble des exigences du projet sera détaillé de façon plus approfondie dans la suite de ce chapitre à travers la présentation du modèle SysML utilisé (§ IV.2.2).

[25] La localisation est la position de la personne dans l'environnement, la posture est son attitude corporelle (assis, debout …), la discrimination évoque le fait de pouvoir distinguer l'humain d'un animal ou d'autres phénomènes susceptibles d'être captés par le détecteur.

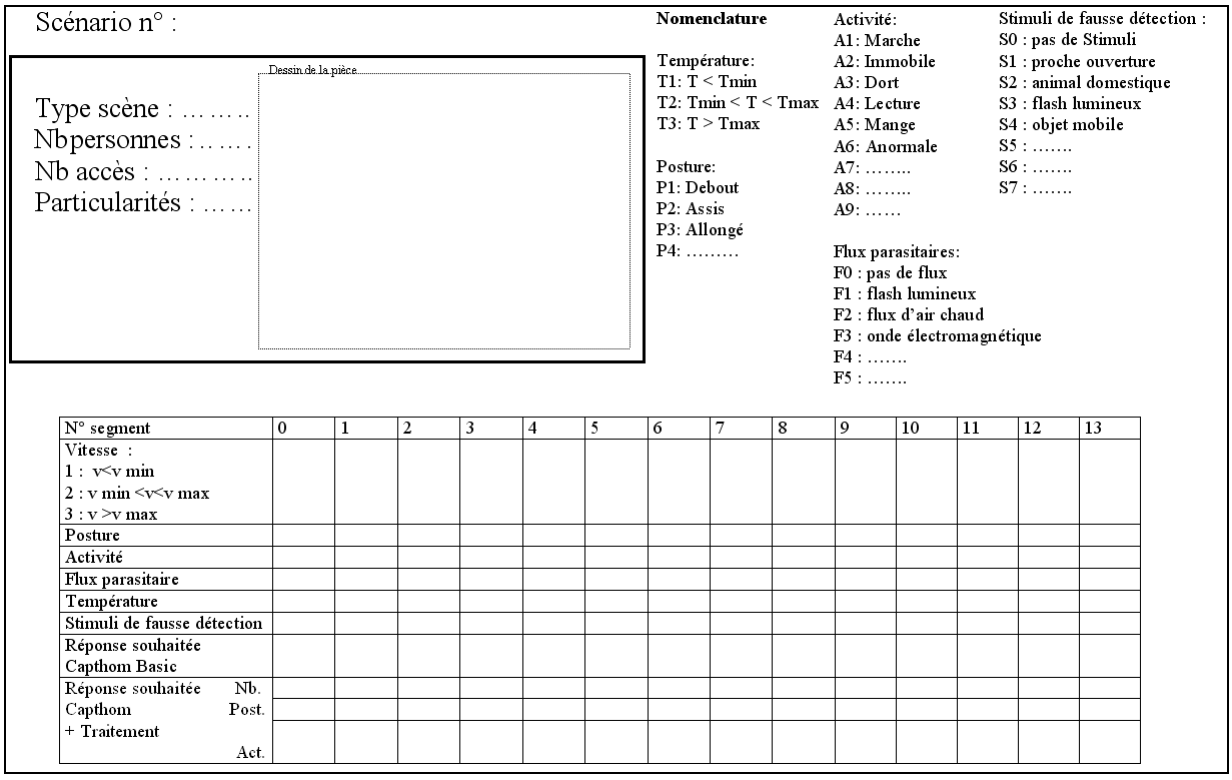
IV.2.2 Expression des besoins et simulation du projet CAPTHOM

Le nouveau capteur de présence humaine, que nous appellerons « capthom », a pour but de réaliser une mission que nous avons pu définir sous forme de scénarios. Ces scénarios représentent les séquences d'événements, issus du contexte, auxquelles le capthom doit apporter une réponse spécifique. Ces scénarios regroupent les situations attendues lors de l'occupation de bâtiments tertiaires ou d'un habitat privé. Afin de conduire l'évaluation des technologies candidates, il est nécessaire de confronter les différentes alternatives aux différents scénarios d'emplois identifiés. Cette première évaluation est faite en réalisant des tests des capteurs retenus face à des stimuli précis que nous avons pu identifier dans les scénarios. Néanmoins, au-delà de tests particuliers des capteurs face à des stimuli planifiés, il est important pour nous de considérer comment les solutions envisagées pour construire les capthoms pourraient être utilisées en réseau de capteurs, au sein d'une solution d'habitat intelligent. Pour cela, il est nécessaire de développer un système de simulation de réseaux de capteurs et de leur réaction, au cours de scénarios d'occupation de bâtiment. Ce système doit modéliser l'implantation de multiples capteurs dans un environnement désiré et fournir les réactions individuelles de chacun d'entre eux.

Dans cette partie, nous présentons la phase d'élicitation des besoins des partenaires du projet, obtenus grâce à l'établissement de scénarios d'emploi du système. Nous exposerons la classification obtenue à partir de leur regroupement. Ensuite, nous présenterons les alternatives technologiques que nous avons dû aborder au cours du projet, puis les besoins spécifiques de simulation du projet pour la recherche de politiques de fusion de données. Enfin, nous évoquerons l'extension du besoin de simulation vers la recherche de solutions d'implantation de réseaux de capteurs pour la surveillance de scènes intérieures.

IV.2.2.1 Les scénarios d'utilisation de CAPTHOM

Les situations dans lesquelles les détecteurs PIR sont mis en défaut ont été clairement identifiées, grâce à plus d'une dizaine d'années d'emploi à grande échelle de cette technologie. Ces configurations permettent d'exprimer les défis à relever par les futurs capthoms. La diversité des situations d'emploi des capteurs de présence humaine impose de traduire la simple exigence « le système doit détecter la présence humaine » en un ensemble très varié de configurations de détection. Afin d'expliciter le plus clairement possible ces exigences sur le comportement attendu des capthoms, nous avons proposé un format de description de ces scénarios s'efforçant de permettre l'expression des spécificités de chaque configuration. Les scénarios dénotent des aspects temporels (ordonnancement d'événements, simultanéité d'actions…), spatiaux (configuration de la scène, placement des capthoms…), physiques (température de la pièce, mouvement d'air…) et doivent exprimer le comportement des participants à la scène, ainsi que la réponse attendue du dispositif. Pour réunir toutes ces informations dans un format accessible et analysable, par l'ensemble des partenaires nous avons développé les fiches de scénarios dont la trame est représentée figure IV.1.



Scénario n° :

Type scène : ……..
Nbpersonnes : ……
Nb accès : ……..
Particularités : ……

Dessin de la pièce

Nomenclature

Température:
T1: T < Tmin
T2: Tmin < T < Tmax
T3: T > Tmax

Posture:
P1: Debout
P2: Assis
P3: Allongé
P4: ……

Activité:
A1: Marche
A2: Immobile
A3: Dort
A4: Lecture
A5: Mange
A6: Anormale
A7: ……..
A8: ……..
A9: ……

Flux parasitaires:
F0 : pas de flux
F1 : flash lumineux
F2 : flux d'air chaud
F3 : onde électromagnétique
F4 : …….
F5 : …….

Stimuli de fausse détection :
S0 : pas de Stimuli
S1 : proche ouverture
S2 : animal domestique
S3 : flash lumineux
S4 : objet mobile
S5 : …….
S6 : …….
S7 : …….

N° segment	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Vitesse : 1 : v<v min 2 : v min <v<v max 3 : v >v max														
Posture														
Activité														
Flux parasitaire														
Température														
Stimuli de fausse détection														
Réponse souhaitée Capthom Basic														
Réponse souhaitée Nb.														
Capthom Post.														
+ Traitement Act.														

Figure IV.1 *Trame des scénarios d'utilisation des capthoms*

Cette fiche propose de découper le scénario en plusieurs phases correspondant à un intervalle de temps durant lequel le contexte présent est spécifié et constant. Le contexte comporte la position des occupants, les phénomènes physiques (inscrits sur le dessin), ainsi que la posture, l'activité et la vitesse de déplacement des occupants, mais également la température de l'environnement et la présence de stimuli de fausses détections. Pour chaque phase, on indique la réponse souhaitée par le capthom. Deux évolutions du système capthom sont envisagées, une venant donner un niveau d'information basique (présence/non présence), l'autre fournissant une information plus poussée (nombre d'occupants, posture et activité).

L'élicitation des besoins, en terme de détection, menée auprès des partenaires du projet CAPTHOM, a donné lieu à l'écriture de plus de 30 scénarios dont les déclinaisons conduisent à plus de 100 tests concrets. Ces derniers regroupent une large variété de situations de vie rencontrées dans un habitat individuel ou un bâtiment tertiaire. Cela couvre les situations de la plus basique à la plus critique vis à vis de la détection. Ces scénarios expriment les besoins des partenaires vis-à-vis de la mission du système. Afin de valider la tenue de ces objectifs, ces scénarios doivent se traduire en autant de cas de tests pour les solutions proposées. Couvrir l'ensemble des situations décrites étant trop gourmand en temps, nous voyons apparaître un premier besoin de simulation, qui est d'être capable d'évaluer les technologies dans des environnements aux dimensions et caractéristiques très variées. Le projet devait donc se doter d'un outil permettant la réalisation de scénarios de test très variés, offrant par exemple des configurations de pièce multiples.

De plus, pour réaliser une campagne de tests réels efficace, nous avons dû dégager de l'ensemble des scénarios obtenus, une classification des cas rencontrés. Nous avons ainsi défini quatre types de situations représentatives des missions confiées aux futurs capthoms. La pre-

mière classe est l'occupation normale et active d'une pièce : personne debout ou assise exerçant une activité où l'agitation est moyenne (marche, lecture, travail de bureau, cuisine…). La seconde classe est le sommeil dans un lieu classique (lit, canapé, fauteuil). La troisième regroupe les activités anormales (chutes, agitations, comportements en décalage avec le lieu). La dernière classe est celle des stimuli de fausses détections (flashs lumineux, ondes électromagnétiques, objets mobiles…) et des causes d'inhibition de détection (vêtements froids, occultations…). Des exemples de chacune de ces classes sont donnés en annexe de cette thèse.

IV.2.2.2 Les alternatives technologiques

Une revue de brevets et l'état de l'art des détecteurs de présence nous ont conduits à envisager diverses solutions technologiques pour la conception des capthoms. Pour faciliter leur comparaison, il est utile de pouvoir simuler le fonctionnement de chacune d'elles afin de pouvoir tester leurs performances dans un nombre très important de situations d'emploi et notamment dans l'ensemble des scénarios envisagés. Par rapport aux propositions faîtes au tableau 4.1, le projet s'est focalisé sur quatre alternatives majeures. Nous allons maintenant présenter sommairement ces quatre types de dispositifs que sont les barrières virtuelles, les PIR, les micro-capteurs infrarouges et les dispositifs utilisant une caméra vidéo.

Les barrières virtuelles regroupent l'ensemble des dispositifs détectant la coupure d'un faisceau immatériel, réalisé par un flux d'onde continu. Ils sont composés d'un émetteur et d'une cellule réflective ou d'un récepteur. L'émetteur détecte l'absence de retour du flux émis ou le récepteur signale l'absence de flux entrant. Les technologies employées sont les flux infrarouges, ultrasons ou lasers. Ces dispositifs sont extrêmement fiables quant à la détection de la coupure du faisceau, néanmoins leur utilisation pour la détection de présence humaine est très limitée. En effet, ces techniques ne permettent pas d'identifier ce qui a coupé le faisceau et ne couvrent que des points de passage de l'environnement. De plus, leur utilisation dans une application de comptage d'entrées/sorties est très critique et ne permet pas de recalage simple (sans intervention humaine).

Les détecteurs PIR utilisent des cellules sensibles aux variations d'émissions de flux infrarouges. Le signal identifiable de telles cellules est constitué par la réponse à une polarisation suivie d'une dépolarisation. Pour capter de telles suites, les cellules sont placées derrière une lentille de Fresnel jouant le rôle de concentrateur et séparateur du flux émis par les phénomènes cibles du capteur. Ainsi, les zones à partir desquelles le capteur reçoit un flux sont entrecoupées de zones d'ombre, de sorte que le déplacement des cibles d'une zone à l'autre est détecté par le PIR [Keller & Cima 1983], [Keller 2000], [Gobeau 2006]. Cette technique de détection a pour inconvénient majeur de n'être sensible qu'aux mouvements des émetteurs de flux infrarouge. Elle possède l'avantage d'être de longue portée (12 mètres), peu coûteuse, facile d'installation et de longue durée de vie. La fiabilité des détections n'est malheureusement pas le point fort du PIR, pour lequel de nombreux stimuli de fausse alarme existent.

Les micro-capteurs infrarouges sont des thermocouples se comportant comme des générateurs de tension lorsqu'ils captent un rayonnement infrarouge. Ils possèdent l'énorme avantage de pouvoir détecter un individu immobile [Escriba 2005], [Haffar 2007], [Godts et al. 2008]. Ils disposent d'une portée moindre comparée aux PIR et ne peuvent fournir une localisation précise de la personne.

Le dernier type de dispositif utilisé sont les caméras vidéo munies d'unité de traitement. La baisse spectaculaire du coût des caméras, constatée ces dernières années, rend cohérente leur utilisation pour la réalisation d'un capteur de présence. Le traitement du flux vidéo permet de déterminer de très nombreuses informations sur l'occupation de la scène observée, la présence [Benezeth et al. 2008], le nombre d'occupants, leur posture, leurs activités [Benezeth et al. 2009] ou leur localisation [Brulin et al. 2009]. La portée des dispositifs standards est comparable à celle des PIR ; le seul point faible réside dans la dépendance aux conditions d'éclairage, en particulier l'impossibilité de détecter la présence avec une très faible luminosité.

Un des besoins qui se dégage, au niveau du projet, est d'évaluer l'utilisation et la combinaison de ces systèmes pour la détection de présence humaine, dans les scénarios et les environnements recensés pendant l'expression des besoins. Il est important d'évaluer si la solution de détection de présence ne devrait pas être réalisée par une combinaison de capteurs supervisée par une politique de fusion de données adéquate.

IV.2.2.3 Simulation pour la fusion de données

La fusion de données est un des éléments clé pour la réussite du projet CAPTHOM. En effet, la solution de détection de présence dans un bâtiment ne viendra pas d'un capteur seul mais de combinaison de techniques variées. Les politiques de fusion de données à développer doivent répondre à des objectifs très divers :

- Fusionner des informations de natures différentes.
- Fusionner des informations venant de capteurs différents.
- Apporter une information de haut niveau à partir de connaissances plus élémentaires.
- Apporter une solution s'adaptant à des scénarios divers dans des environnements très variés.

Pour mettre au point et tester ces politiques, il est important de disposer d'enregistrements des réponses des capteurs à fusionner, dans l'ensemble des configurations attendues. Ainsi la simulation de réseaux de capteurs permettra d'enrichir les cas de tests réels avec autant d'exemples simulés qu'on le désirera. De plus, la simulation permet d'obtenir un premier aperçu des configurations les plus délicates à réaliser dans la réalité.

Le développement d'algorithmes de fusion de données est un besoin spécifique au projet. Il nécessite l'emploi d'un outil dédié. Cet outil fait partie de l'EDS de CAPTHOM. C'est un outil de conception détaillé au sens donné par la figure I.5 du paragraphe I.1.4.

IV.2.2.4 Simulation pour l'implantation de réseaux de capteurs

Le dernier besoin, lié au projet, est relatif à la phase de déploiement des futurs capthoms. Il s'agit de l'étape de design du réseau implanté dans une pièce particulière. Pour équiper un bâtiment donné, il est utile d'optimiser le placement des capteurs afin de réaliser la meilleure observation avec le minimum de capteurs. Un tel outil prend en compte les contraintes inhérentes au bâtiment dans lequel sera implanté le réseau, mais également les caractéristiques de la mission qui sera fixée au système. Pour cela, une fonctionnalité de placement de capteurs doit être ajoutée à l'EDS CAPTHOM. Ce service sera fusionné avec la simulation de réseau de capteurs dont il reprend le modèle du système.

IV.3 Analyse du projet CAPTHOM par modélisation SysML et MéDISIS

L'analyse du produit objectif du projet est réalisée avec le support d'un modèle SysML. La nature et la complexité du système conçu ne nécessitent pas l'usage d'une des normes de processus d'IS détaillées au chapitre I. Néanmoins, nous avons choisi d'adopter une procédure classique d'analyse fonctionnelle, qui sera basée sur les modèles. Nous utilisons ainsi une démarche d'ISBM dont le modèle SysML est le pivot. Nous utilisons MéDISIS pour analyser la problématique de CAPTHOM et faire ressortir les besoins d'analyses nécessitant des outils d'étude dédiés. Nous verrons ainsi que le modèle et son exploitation permettent d'aiguiller et affiner les développements nécessaires au projet.

L'organisation du modèle reprend les phases de l'approche d'ISBM que nous avons utilisée. Les tâches suivantes ont été réalisées :

- Définir le cadre de l'étude (cas d'utilisation des capthoms).
- Saisir le modèle d'exigences.
- Modéliser une base architecturale de la solution.
- Traiter les alternatives technologiques.
- Modéliser la détection de présence :
    - Modéliser l'environnement,
    - Modéliser le comportement de la détection,
    - Modéliser la physique de la détection,
    - Modéliser les scénarios d'emploi.
- Réaliser l'AMDEC d'un capthom (utilisation de MéDISIS).

L'agencement des tâches n'est pas strictement chronologique. Des itérations sont notamment utilisées entre la modélisation de l'architecture et du comportement. De plus, des liens entre les vues sont construits au fur et à mesure, en employant les relations d'allocation et les liens spécifiques au maniement des *requirements*. On crée par exemple des liens entre exigences et comportements ou exigences et architecture.

La figure IV.2 montre l'organisation des *packages* du modèle Capthom. Nous avons réalisé un répertoire contenant la définition des flux employés dans le modèle, afin d'obtenir une construction indépendante de la solution réalisée, donc utilisable telle quelle dans d'autres projets. Les alternatives technologiques sont regroupées dans un répertoire séparé venant spécifier des options de conception pour l'architecture, décrite de façon globale dans le répertoire « Structure ». Des *packages* distincts sont utilisés pour regrouper le fruit de chacune des tâches de la procédure : les exigences, les cas d'utilisation et le comportement. Le *package* « Environnement » regroupe la déclaration des phénomènes intervenant dans une détection ainsi que leurs attributs physiques. Le *package* « Physique de la détection » présente plus en détail les relations induites par l'activité de détection.

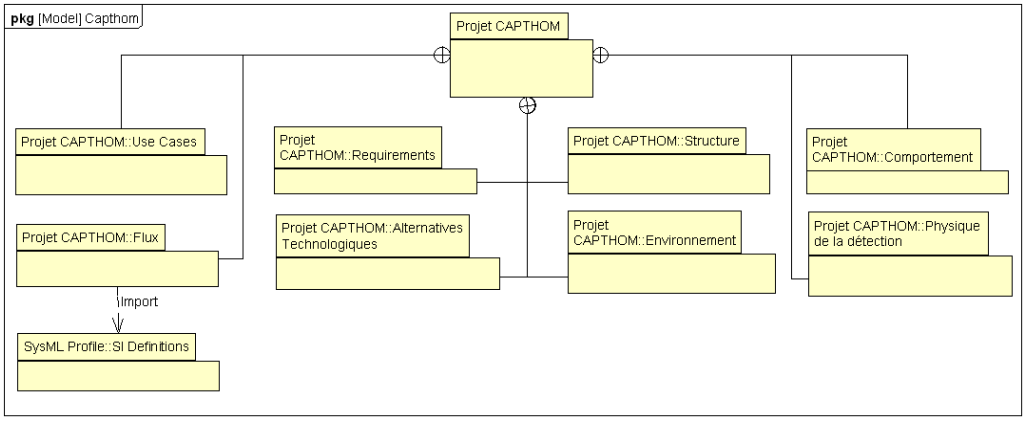


Figure IV.2 *Diagramme de Packages du projet CAPTHOM*

Remarque de modélisation : le *package* « Flux » importe les définitions des types du Système International, présentes par défaut dans le profil SysML de l'outil que nous utilisons[26].

IV.3.1 Le cadre du projet CAPTHOM

Pour exprimer la portée du projet Capthom, il est naturel de commencer par définir les limites de l'étude. Pour cela, nous définissons les cas d'utilisation du système développé (Figure IV.3). La représentation permet d'isoler les éléments en relation avec les futurs capthoms. Nous identifions deux types d'acteurs humains que sont l'installateur et l'utilisateur du système. Le *Use Case* « Utilisation de Capthom » met en évidence que l'environnement rentre en concurrence avec l'utilisateur lorsque le capthom doit capter les flux trahissant la présence humaine. En effet, le capthom est ouvert à tous les éléments présents dans la zone qu'il surveille. Le diagramme révèle ainsi un point crucial de la détection de présence : la capacité de discrimination du dispositif vis à vis des flux qu'il reçoit.

La représentation fait apparaître également les principales composantes permettant de communiquer l'état d'occupation de la pièce. On identifie l'étape de captation des flux, le premier traitement de discrimination du signal significatif (« identifier l'être humain ») et l'analyse de plus haut niveau conduisant à qualifier l'état de l'humain identifié (« Déterminer l'activité », « Déterminer la posture », « Déterminer la localisation »).

Le diagramme illustre enfin que le capthom est toujours considéré comme une entité venant se connecter à un superviseur global. Cette relation exprime une externalisation de l'intelligence de traitement à l'échelle d'un habitat intelligent. Le capthom effectue des traitements pour étudier les flux qu'il reçoit, cependant les activités de commande de l'habitat ne lui incombent pas. Ceci permet de spécifier clairement les limites de développement du capthom. Les capthoms sont des systèmes élémentaires sans « conscience » des autres membres du réseau. Le diagramme a permis de montrer des frontières physiques (ouverture du capthom à tous les flux ambiants) et fonctionnelles, en spécifiant que le capthom n'est qu'une brique d'un réseau plus étendu pour lequel il est uniquement un fournisseur d'un type d'information.

[26] Artisan Studio 7.0. Copyright © 1997-2008 Artisan Software Tools.

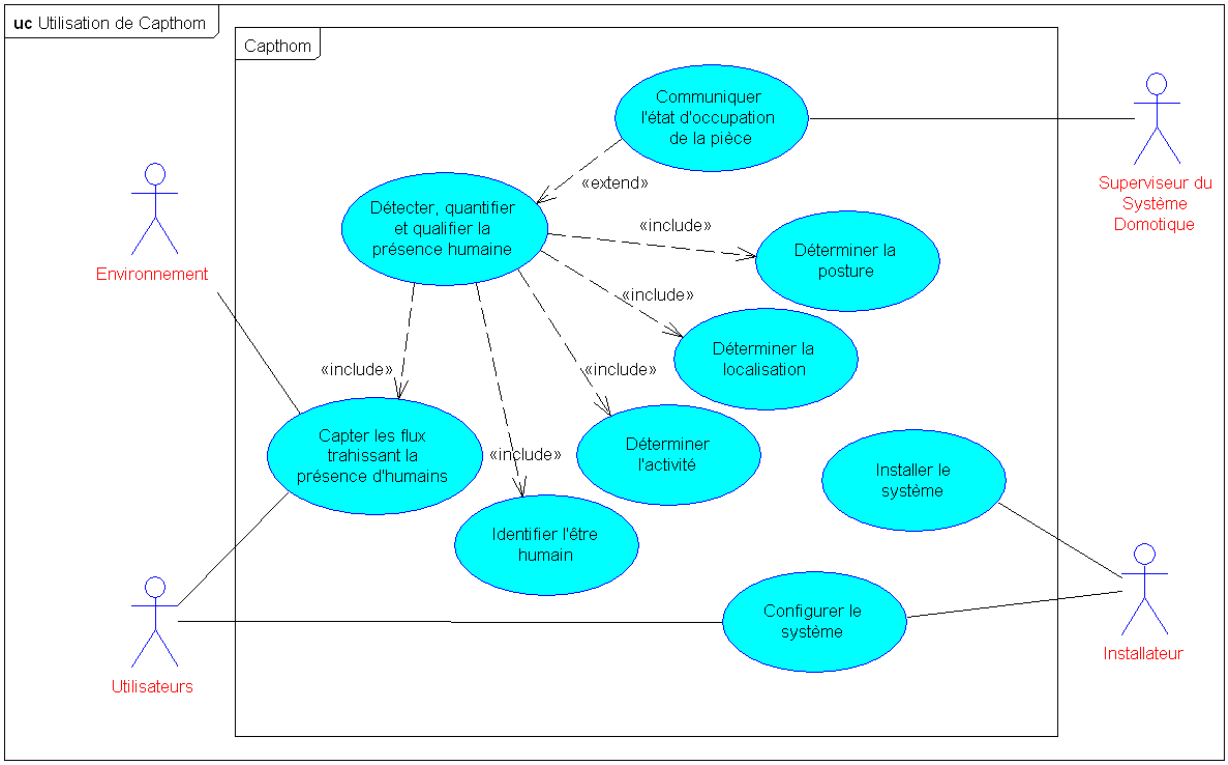


Figure IV.3 *Cas d'utilisation d'un capthom*

IV.3.2 Le modèle d'exigences

Le modèle d'exigences est utilisé pour structurer les besoins exprimés par l'ensemble des partenaires, au cours des différents cahiers des charges (CdC) successifs. Les *Requirements Diagrams* (RD) donnent la possibilité d'organiser les exigences de façon précise et permettent de décomposer chacune d'elles, afin d'obtenir des demandes simples, qu'il est possible d'affecter et de lier à des éléments ciblés du modèle. Le CdC du projet CAPTHOM était le résultat de l'agrégation des CdC particuliers de chaque participant au projet. Les exigences exprimées souffraient du manque d'une représentation commune, d'un vocabulaire commun et d'une hiérarchisation. Cela conduisait à des exigences dupliquées, peu claires pour l'ensemble des intervenants et grandement désorganisées. La rédaction du modèle d'exigences en SysML, par l'intermédiaire de RD, nous a permis de palier à nombre de ces problèmes et d'obtenir un modèle d'exigences simple, précis et robuste face aux différentes modifications.

Nous avons identifié un pôle d'exigences relatif au processus de détection, formulant des attentes sur sa qualité, sa nature et les conditions dans lesquelles il doit être mené (Figure IV.4). Nous avons organisé le RD en trois *requirements* représentant chacun de ces thèmes. Ces derniers sont décomposés en *requirements* plus particuliers exprimant pour la plupart des exigences quantifiées. Par exemple, l'exigence de « qualité de la détection » est spécifiée en trois *requirements* exprimant le « temps de réponse » attendu, « la portée » de la détection et sa « fiabilité ».

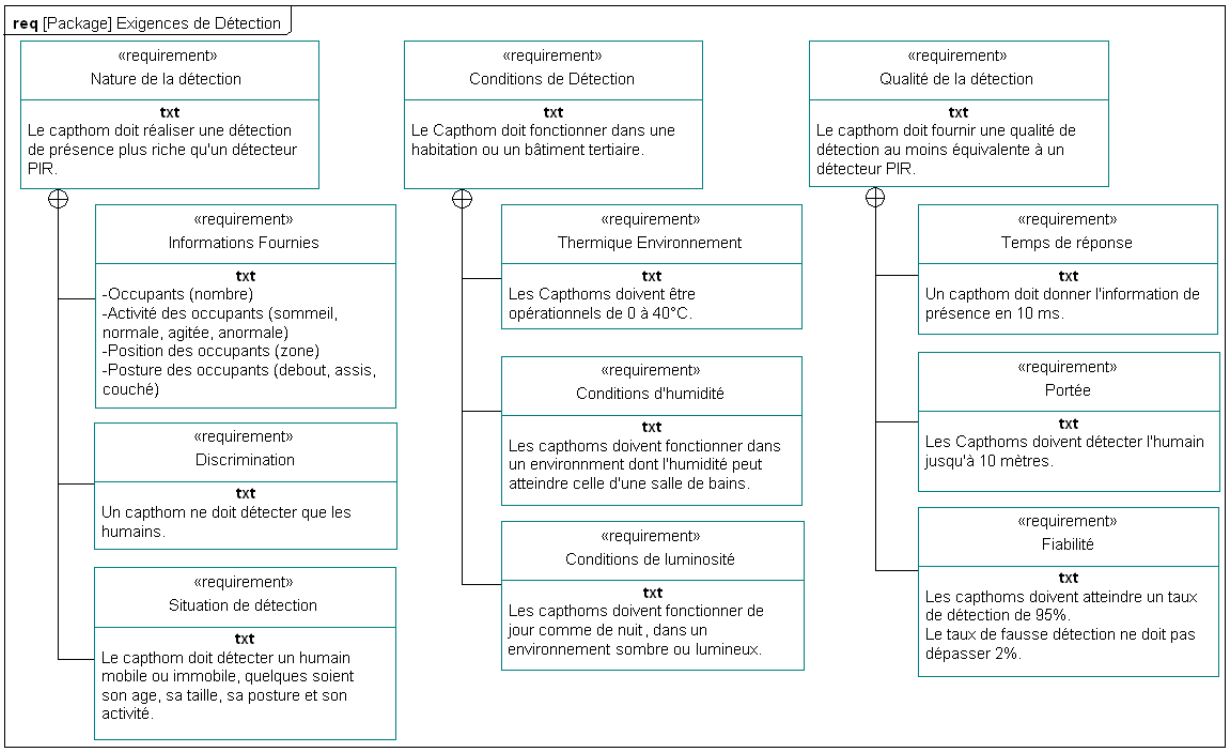


Figure IV.4 *Requirement Diagram des Exigences de Détection*

Un second RD (Figure IV.5) regroupe les exigences générales, relatives à l'alimentation, la fixation, le prix… Ces exigences émanent des besoins de remplacer les systèmes existants par une solution au moins aussi bonne. Le capthom doit s'intégrer dans son lieu d'implantation sans nécessiter de modification des installations électriques usuelles, tout en utilisant un design déjà bien accepté par le public pour ce genre de détecteur. L'exigence de communication mentionne que les capthoms sont conçus pour s'intégrer à un système d'habitat intelligent. Ils doivent donc respecter des protocoles de communications fixés et être identifiables en tant qu'« individus ».

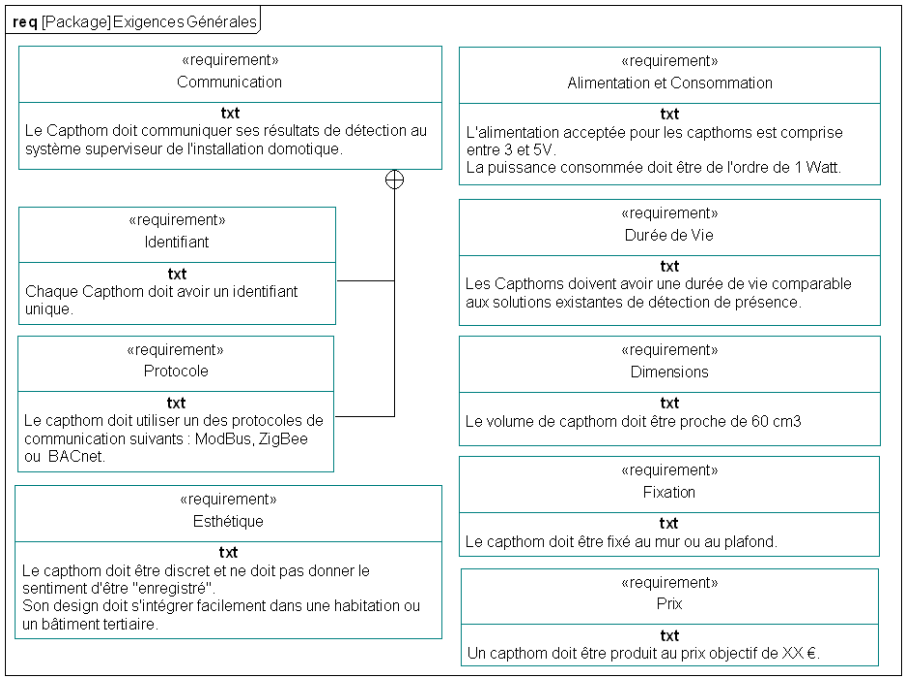


Figure IV.5 *Requirement Diagram des Exigences générales*

IV.3.3 Architecture de capthom

Capthom devra être capable de capter les signaux traduisant la présence humaine, mais également de traiter ces signaux pour fournir en sortie une information de haut niveau résultant d'une première analyse. Nous avons pris le parti de baser le modèle d'architecture du capteur sur celui des capteurs décrits dans l'ouvrage de [Robert et al. 1993] formalisant le concept et généralisant les architectures utilisées pour construire de tels systèmes. Ces travaux, adaptés à notre problématique, nous ont permis de dégager l'architecture de capthom, telle qu'elle est représentée sur l'IBD de la figure IV.6.

Le modèle spécifie les composants principaux du futur capteur de présence. On identifie trois composants intervenant dans le traitement du signal. Le « conditionneur » effectue une mise en forme du signal issu du « transducteur », l' « unité de traitement » élabore la réponse du capthom en fonction du signal transmis par le « conditionneur ». Cette prise de décision peut nécessiter des connaissances stockées dans une « mémoire ». Le module chargé de rendre le capthom communicant est représenté par l'« interface de communication » reliée au « système superviseur ». Le « transducteur » est le point d'entrée du capthom, il rend sensible le capteur à son environnement. Il constitue un point crucial de l'architecture, car il est le seul composant en relation avec le phénomène recherché : les flux provenant des humains. Néanmoins, on remarque aussi qu'il est en relation avec l'environnement et qu'il est donc soumis à plus d'influences que nous le désirons.

Remarque de modélisation : le « système superviseur » et l'«alimentation électrique » sont modélisés par des *references*, car ils ne font pas partie du capthom mais entrent en relation avec ses composants.

Les variations de typage des *flow ports* expriment, dans ce modèle, la transformation du flux intervenant au cœur du capthom, jusqu'à l'obtention de la « réponse capthom » transmise sous forme de « signal numérique ». L'alimentation est connectée à un port relatif au capthom dans son ensemble, afin de ne pas surcharger le diagramme par l'ensemble des connexions d'alimentation des composants particuliers. Les ports utilisent les types définis dans le *package* « Flux » du modèle Capthom.

Remarque de modélisation : Par la définition du port correspondant à la « cellule sensible » du « transducteur », notre intention était de spécifier une connexion entrante unidirectionnelle, modélisant les flux arrivant au capteur et le fait que celui-ci n'est qu'à l'écoute de son environnement. Cependant, nous souhaitions également spécifier que les flux entrants étaient multiples et de natures différentes. Nous avons pour cela défini une *flow specification* dans le *package* regroupant les flux du modèle. Ce dernier rassemble tous les types de flux rencontrés dans l'environnement (ondes mécaniques, ondes électromagnétiques, flux lumineux…). Or, la spécification de SysML stipule qu'un port typé par une *flow specification* (un port non atomique) est obligatoirement bidirectionnel. Néanmoins, toutes les *flow properties* de cette *flow specification* sont de *direction* « in ».

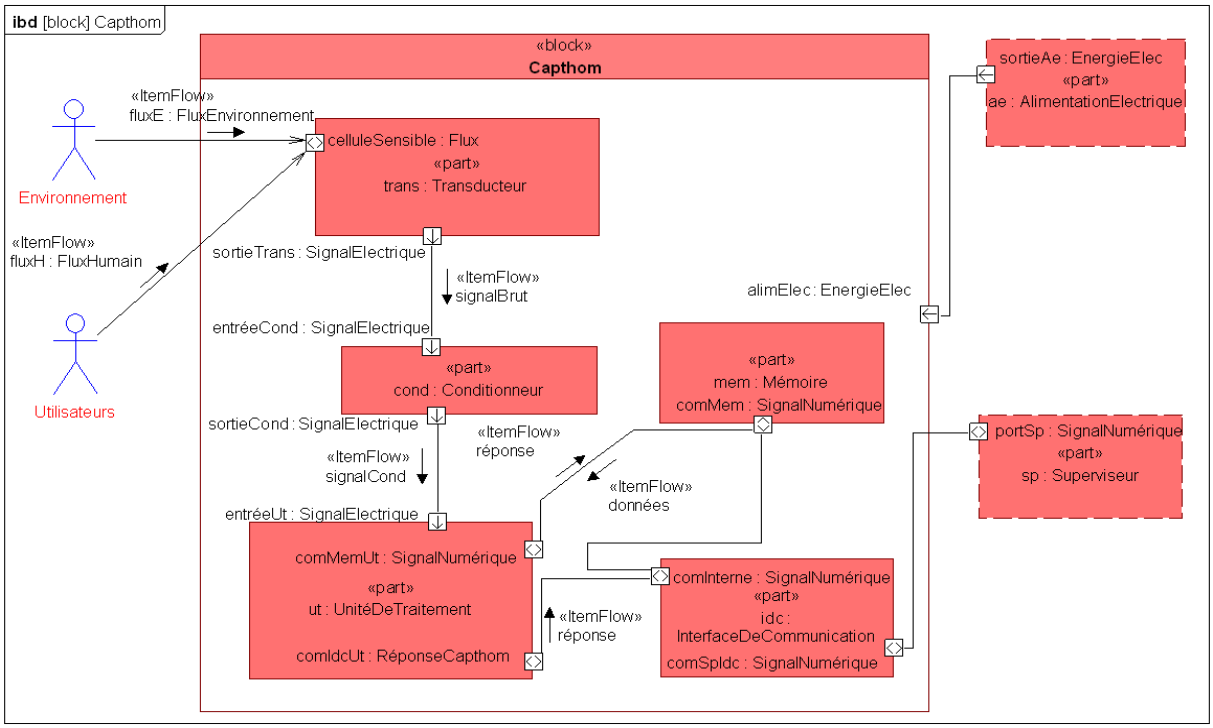


Figure IV.6 *IBD Capthom*

IV.3.4 Traiter les alternatives technologiques

Si on calque le modèle général du capthom sur les alternatives technologiques spécifiées au paragraphe IV.2.2.2, on remarque que la différence fondamentale intervient au niveau du « transducteur ». Le BDD « capthoms » du *package* « alternatives technologiques » (figure IV.7) illustre les pistes évoquées pour le capthom. La définition du *block* Capthom sert de généralisation pour les alternatives à considérer. Pour chacune d'elles, on effectue une redéfinition du *part* « trans » modélisant le transducteur. Cette redéfinition est réalisée par l'emploi de types spécialisés du *block* « Transducteur » venant préciser la nature du transducteur dans les différentes alternatives.

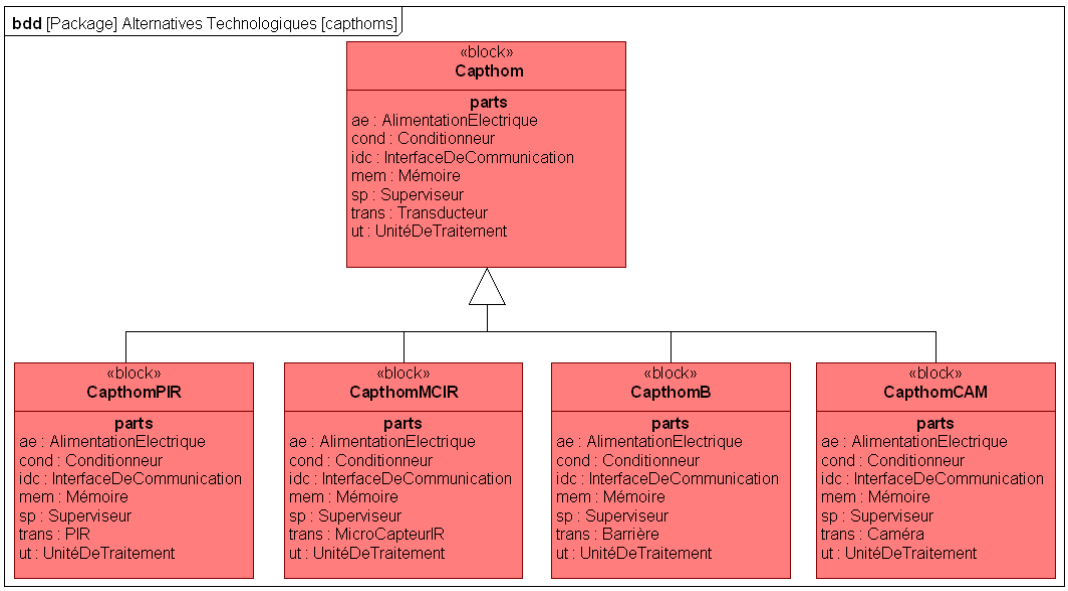


Figure IV.7 *BDD des alternatives de capthoms*

La définition du *block* « Transducteur » et de ses spécialisations (figure IV.8) est l'occasion d'identifier les caractéristiques générales de ces composants. La position et l'orientation du transducteur sont ce qui va conditionner les flux que pourra capter le détecteur. C'est pourquoi, nous avons choisi de définir les attributs de placement, que sont la localisation (« locCapteur ») et l'« orientation », au transducteur plutôt qu'au capthom. Le transducteur est défini par la zone qu'il couvre [David et al. 2007]. La déclaration de cette caractéristique utilise la profondeur de champ (« dof ») et l'« azimut » donnant l'angle d'ouverture. Nous avons défini deux *value properties*, représentant ce qui transitera par les ports du transducteur. « fluxReçu » indique le flux entrant, il est volontairement laissé sans type, car l'éventail des flux possibles est très large. « signalEmis » est le signal transmis par le port de sortie « sortieTrans », il est de type « signal électrique » dans la mesure où nous utilisons des convertisseurs de flux physiques en flux électriques. Ces deux *value properties* sont allouées respectivement aux ports « celluleSensible » et « sortieTrans » par une relation *allocate from*, afin de les lier explicitement.

Le comportement des transducteurs a ensuite été caractérisé par le biais d'*operations* et de leurs contraintes allouées. Nous avons scindé les activités d'un transducteur en deux parties : une partie détection qui exprime le fait que le transducteur est positionné pour recevoir uniquement une partie des flux de son environnement et une partie réaction définissant la création du signal de sortie en fonction des caractéristiques du flux reçu. Pour chacun de ces éléments de comportement, nous avons déclaré une *operation*, à laquelle est allouée une *constraint property* déclarant la relation mathématique sous-jacente. Ainsi, l'*operation* « Détection » est allouée à la *constraint property* « detect », « Réaction » est allouée à « réa ». Pour chaque alternative technologique, une redéfinition de la contrainte de réaction est effectuée. En effet, chaque technologie possède une relation entrées/sorties différente. Les redéfinitions de ces *constraint properties* sont réalisées par une spécialisation de leur type (donné par un *constraint block*). La déclaration des *constraint blocks* (figure IV.9) permet de cibler les paramètres entrant en jeu dans ces comportements.

A ce niveau de modélisation, nous sommes capables d'identifier les *constraint parameters* utilisés par ces contraintes, et donc d'identifier les éléments clés du comportement. La détection de flux dépend de la localisation de la cible, de celle du capteur et de son champ de vision. Nous utilisons le booléen « detection » pour mentionner si le capteur reçoit ou non un flux. Ce dernier est utilisé pour activer la seconde contrainte : « RéactionTransducteur », ayant pour paramètres le flux reçu et le signal émis. Nous ne pouvons pas encore exprimer la relation mathématique vérifiée, ainsi le champ « constraints » des *constraint block* est laissé vide. De même, il nous faudra connaître les caractéristiques physiques des capteurs pour résoudre les contraintes de détection. La construction de ce modèle permet ainsi d'identifier un besoin d'analyse spécifique lié au projet. Il s'agit de réaliser la caractérisation réelle des capteurs que nous souhaitons tester. Ceci a été indiqué dans le diagramme par une note « problem ». Ce problème sera vite élucidé, puisque partie prenante de la thèse d'Antoine Belconde, membre du projet CAPTHOM, qui sera soutenue au premier semestre de l'année 2010.

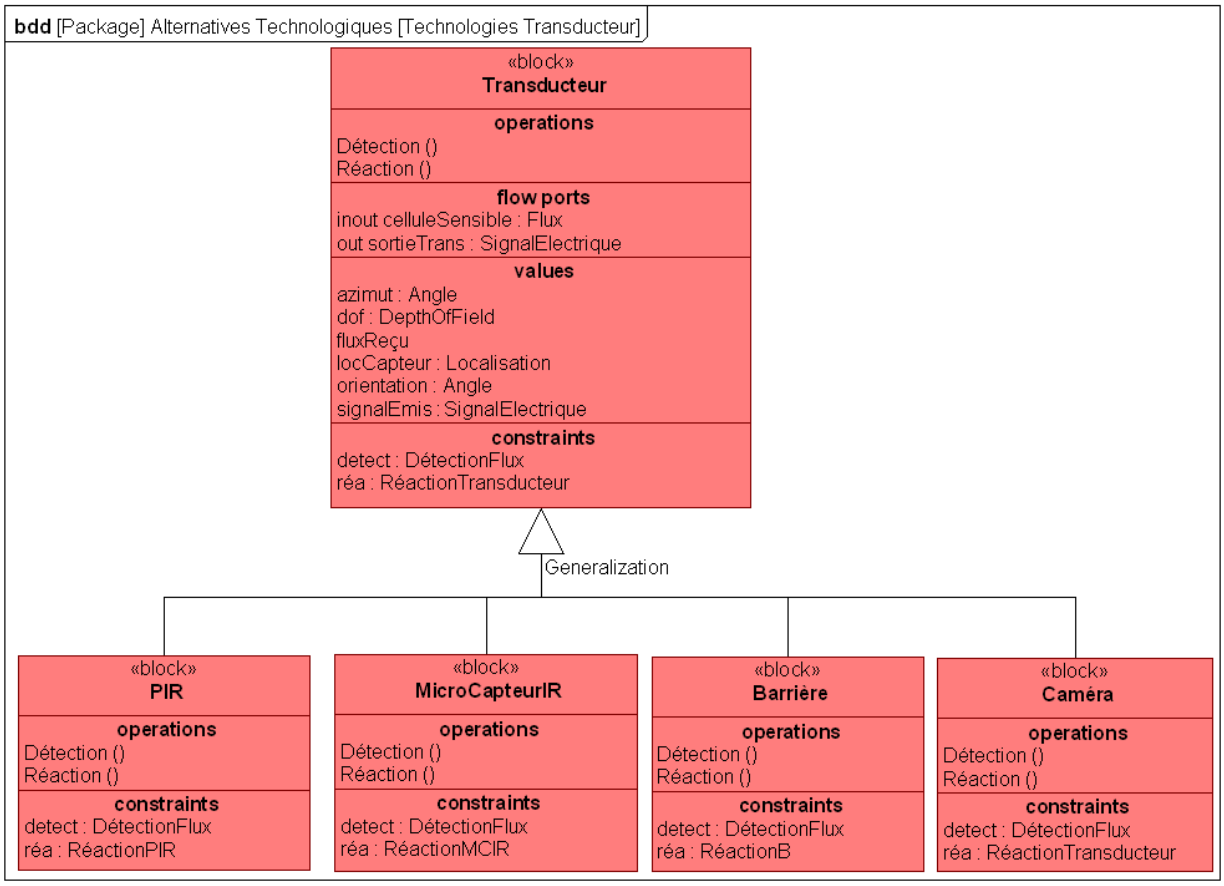


Figure IV.8 *BDD des alternatives de transducteurs*

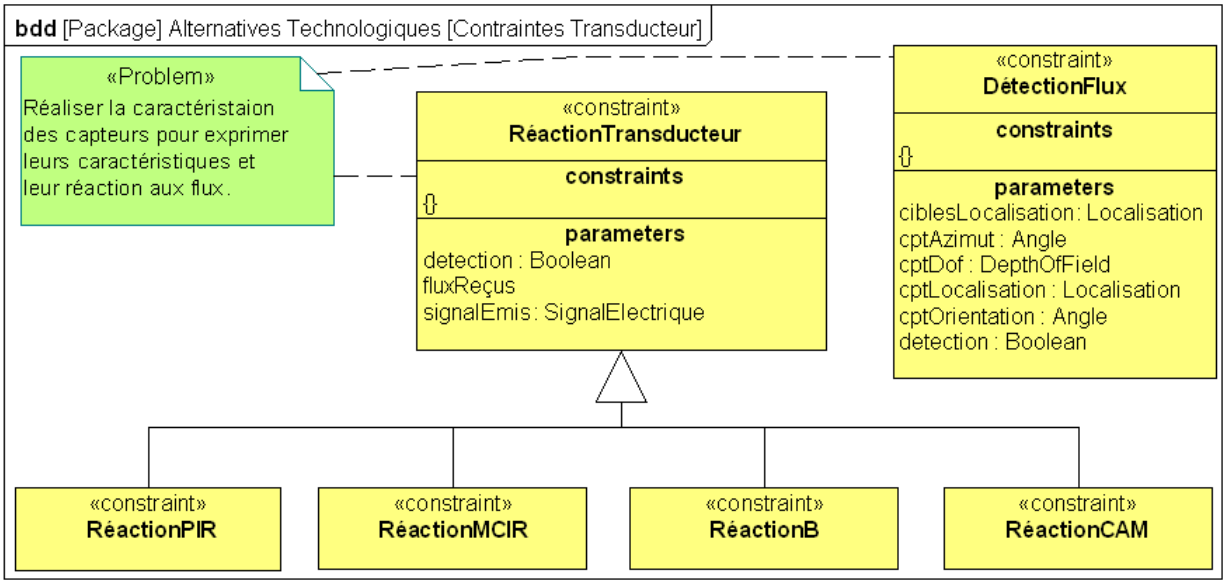


Figure IV.9 *BDD de définition des contraintes des transducteurs*

IV.3.5 Modéliser la détection de présence

Analyser l'environnement, avant de modéliser le comportement est une étape des analyses système. La détection de présence est le phénomène central à l'analyse des capthoms. Il est donc de prime importance que d'identifier et caractériser toutes les parties prenantes de ce phénomène. Les contraintes mathématiques, traduisant des contraintes physiques, liées à la détec-

tion doivent être analysées, les scénarios d'emploi et de comportement de l'environnement doivent aussi être modélisés afin de disposer de la vision la plus large possible des mécanismes impliqués dans la détection de présence humaine.

IV.3.5.1 L'environnement, la cible des capthoms

Le rôle des capthoms est de capter les flux provenant de leur environnement et de révéler par leur analyse si un humain est présent. Nous avons modélisé ces phénomènes par des *item flows* sur l'IBD de la figure IV.6. Nous avons typé ces *item flows* par des *blocks*, de manière à pouvoir détailler facilement leurs caractéristiques. En effet, nous avons souhaité utiliser une notation compacte unifiant les émetteurs et le flux émis dans un même objet. Sur le BDD de la figure IV.10, nous avons recensé les composantes des deux ensembles de flux que nous avons dégagés. Nous avons utilisé une relation de composition pour noter tous les éléments contributeurs aux flux de l'environnement. Les flux humains sont représentés en un seul *block*, car tous issus de la même entité physique. Pour les deux *blocks*, nous avons déclaré une *value property* « fluxEmis » regroupant l'ensemble des composantes, et une seconde dénotant la localisation physique des émetteurs des flux.

Nous avons listé tous les flux susceptibles de trahir la présence d'un humain : sa température, ses émissions de rayonnements infrarouges, de bruit ou de CO_2, son poids, ses contours, ses mouvements et les vibrations qu'il induit. Pour les flux environnementaux, nous avons listé les éléments émettant des flux pouvant masquer l'humain ou induire en erreur le capteur. Pour chacun d'eux, regroupés ensuite par le *block* « FluxEnvironnement », on liste comme pour l'humain les flux générés.

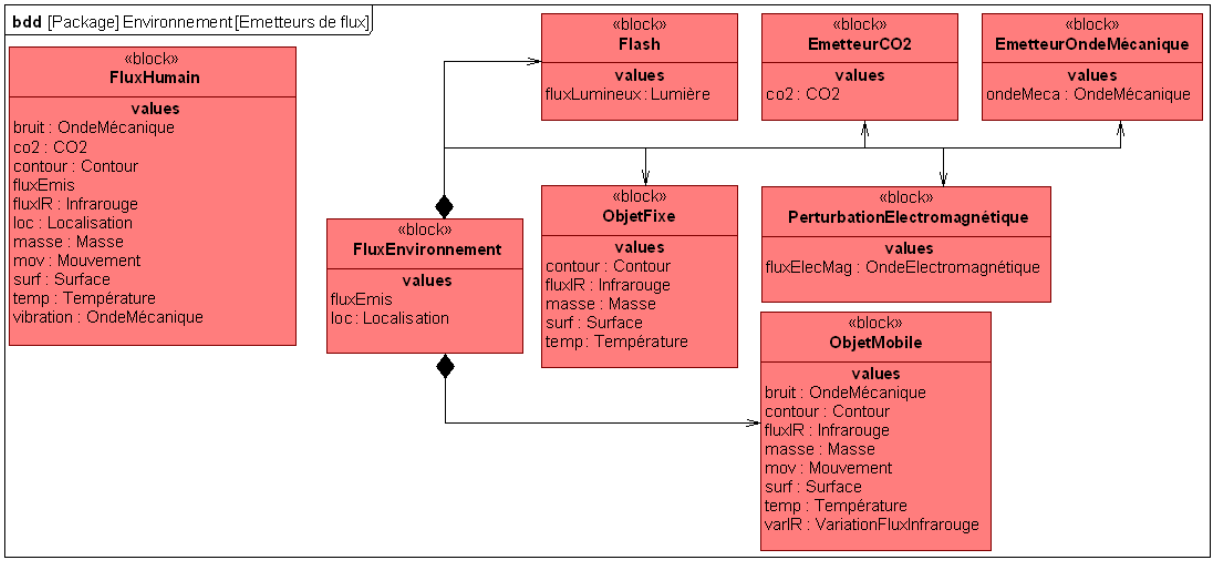


Figure IV.10 *BDD définition des types de flux*

IV.3.5.2 Le comportement de la détection

La description du fonctionnement des capthoms et la mise en évidence d'une première distribution des traitements à travers la structure, sont réalisées par deux diagrammes : un AD et un SD.

- Sur le premier (figure IV.11), l'activité de détection a été allouée au *block* « Capthom » et les traitements intermédiaires réalisés par les composants ont été déclarés par des sous-activités. Ces dernières sont projetées sur la structure

par leur allocation aux *blocks* typant les *parts* correspondants. L'AD permet de mettre en évidence les échanges de flux en créant des objets (*object nodes*) transmis d'une activité à l'autre. Les *object nodes* que nous avons utilisés, sont en réalité les *item flows* que nous retrouvons de façon cohérente sur l'IBD du Capthom (figure IV.6). Ce mécanisme permet de montrer les relations entre les diverses vues du système et notamment entre la vue structurelle et comportementale. L'allocation des activités aux composants du capthom et l'utilisation des *item flows*, fournissent un diagramme que l'on peut calquer sur la structure. On identifie alors clairement quelles sont les connexions employées pour les relations entre activités. De même, les *pins* d'entrée et sortie de l'activité, utilisés pour admettre et relayer les objets de l'activité, sont alloués à la « celluleSensible » et au port de sortie de l'interface de communication.

On remarque, une nouvelle fois, l'arrivée simultanée des flux à traiter. Le signal est ensuite traité en étapes successives. Tout d'abord le signal est mis en forme (conditionné) pour être interprétable par l'unité de traitement. L'activité de traitement du signal conduit à formuler la réponse du capthom. Cette réponse est élaborée à partir de deux ressources : le signal conditionné et les données stockées en mémoire. Ces données peuvent être de natures variées : il peut s'agir de l'historique des détections ou d'un modèle utilisé par l'algorithme de traitement. Enfin, la réponse est relayée par l'interface de communication vers l'extérieur du capthom. L'expression de ces points clefs du traitement va être précieuse lors de l'AMDEC du système qui permettra de faire apparaître les composants et fonctions de première importance.

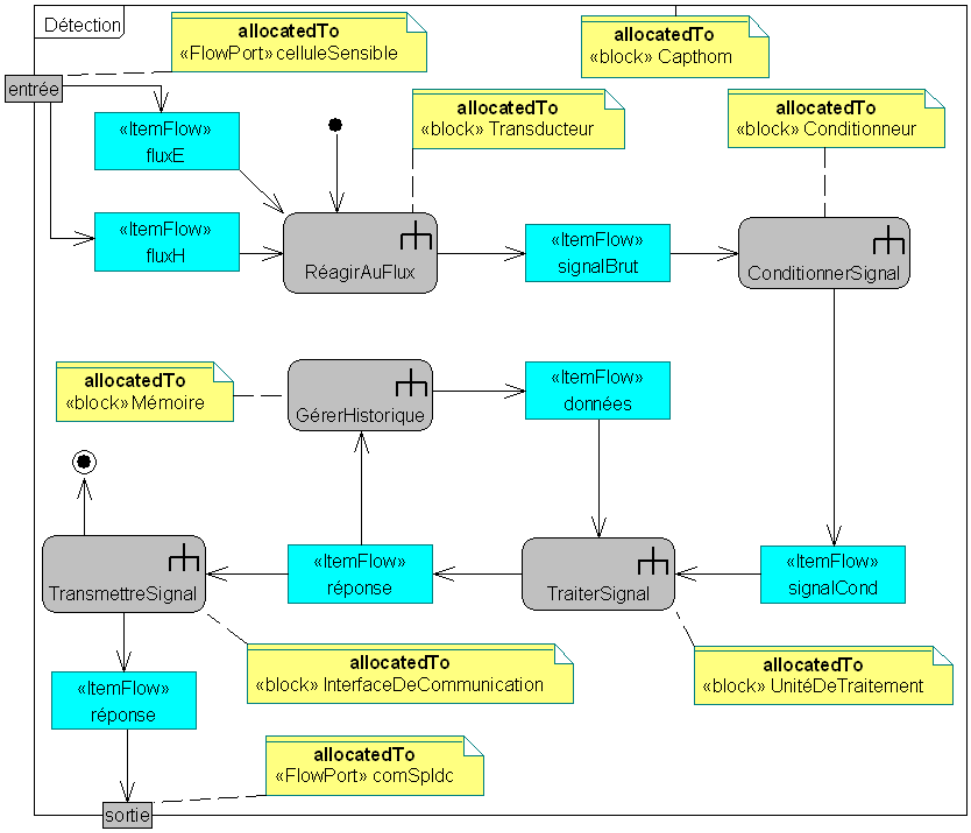


Figure IV.11 *Diagramme de l'Activité de détection*

- Sur le second (SD de la figure IV.12), nous avons décrit les interactions ayant lieu au cours de la détection. Ce format de diagramme nous permet de représenter directement les *parts* impliqués dans la détection de présence. Cette vue extérieure au Capthom, nous donne l'occasion d'illustrer les relations avec les éléments extérieurs, comme le système superviseur ou l'environnement. Dans la mesure où l'allocation des activités sur la structure a été faite de façon complète dans l'AD, les deux vues sont très similaires et on peut noter le parallèle entre les activités et le déclenchement des *operations* illustrant une instanciation de l'activité.

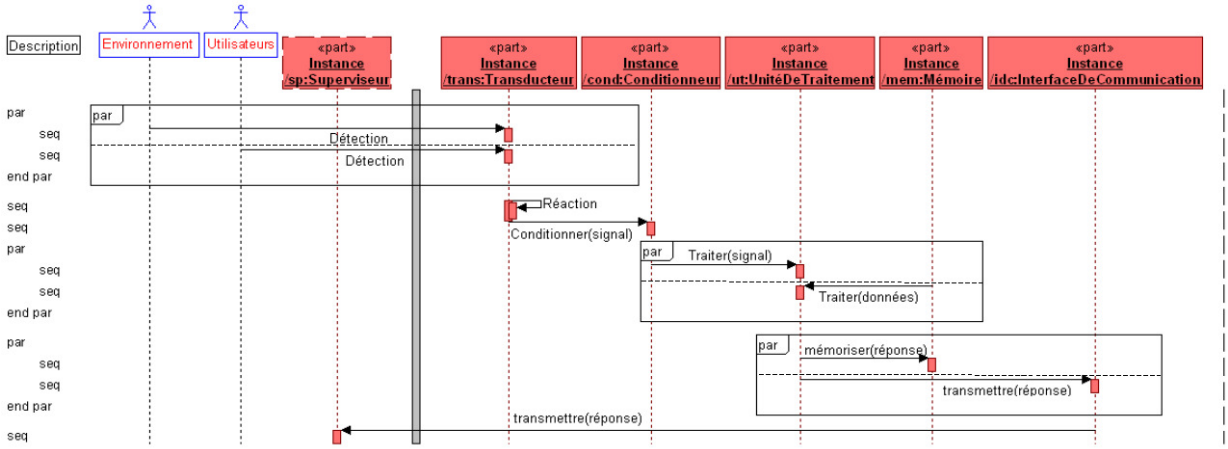


Figure IV.12 *Diagramme de la Séquence de détection*

Remarque de modélisation : les traitements concurrents sont modélisés par l'utilisation de blocs de parallélisme. Le SD met l'accent sur les *operations* déclenchées au cours du temps et non sur les flux échangés, cette vue met plus en avant les relations de causalité entre les composants et leurs interventions.

IV.3.5.3 La physique de la détection

Afin d'appréhender le fonctionnement d'un capthom, il est primordial de comprendre les phénomènes réels régissant les *operations* de détection et réaction au niveau du transducteur. Pour cela, nous avons cherché à détailler les contraintes induites par ces phénomènes et surtout à identifier les paramètres prenant place lors d'une détection de présence humaine. Nous avons défini un *block* « ScèneDeDétection » regroupant les flux humains, ceux de l'environnement et un transducteur. Le *block* « ScèneDeDétection » possède une contrainte « physiqueDeDétection » composée des deux contraintes liées au transducteur. Ainsi nous pouvons réaliser le Parametric Diagram de la détection (figure IV.13), en ayant accès aux *value properties* des flux extérieurs et du transducteur. Ceci permet de montrer les participations conjointes des différents éléments à une même contrainte.

Remarque de modélisation : cette modélisation constitue un biais dans la modélisation de la structure du système. Ce biais est nécessaire, car les *constraint parameters* d'une *constraint property* ne peuvent être connectés qu'aux *value properties* du *block* possédant la contrainte ou à ceux de ses *parts*. Ainsi, les trois éléments participant à la scène de détection (le transducteur, l'environnement et les humains) doivent être regroupés en un ensemble de plus haut niveau.

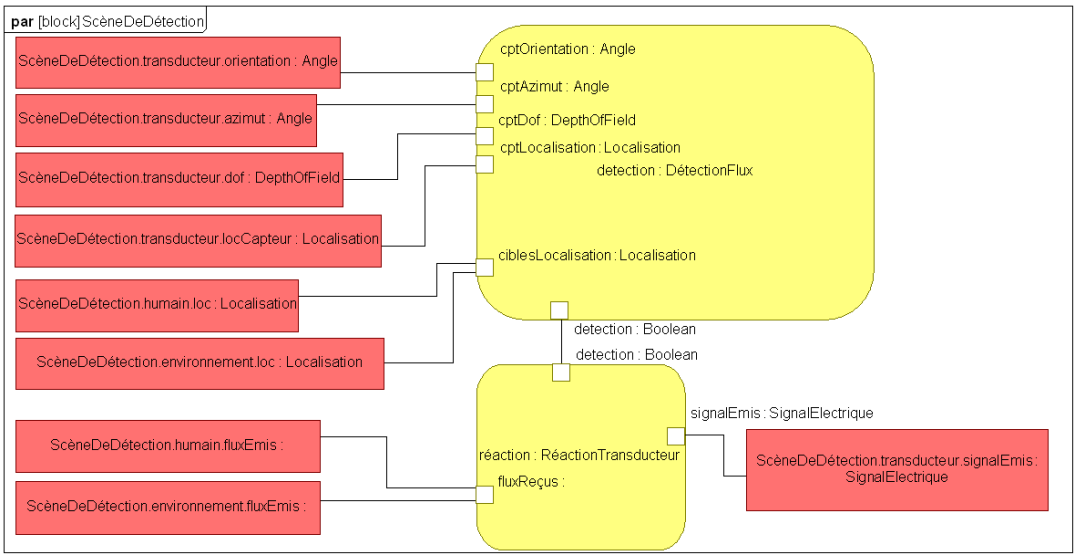


Figure IV.13 *Parametric Diagram de la détection de présence*

IV.3.5.4 Les scénarios d'emploi du capteur

Les scénarios définis durant l'établissement du CdC doivent être retranscrits dans le modèle. Cependant, comme nous l'avons mentionné au paragraphe IV.2.2.1, les aspects très hétérogènes de ces scénarios nous ont poussés à utiliser un format spécifique pour leur déclaration. Nous avons donc voulu généraliser, par le modèle SysML, les scénarios listés et les situations qu'ils comportaient. Pour cela, nous avons modélisé par une *statemachine* (figure IV.14) les différents états remarquables de la cible du capthom : l'Humain. Nous avons donc déclaré un *block* « Humain » modélisé tel qu'il est vu par la problématique de CAPTHOM (figure IV.15). D'après les attentes de la détection de présence, l'humain est caractérisé par trois *value properties* : la localisation, la posture et l'activité. Ces trois variables peuvent être modifiées par les trois *operations* qui leur sont dédiées. Un humain émet un flux dont nous avons déclaré le type (figure IV.10).

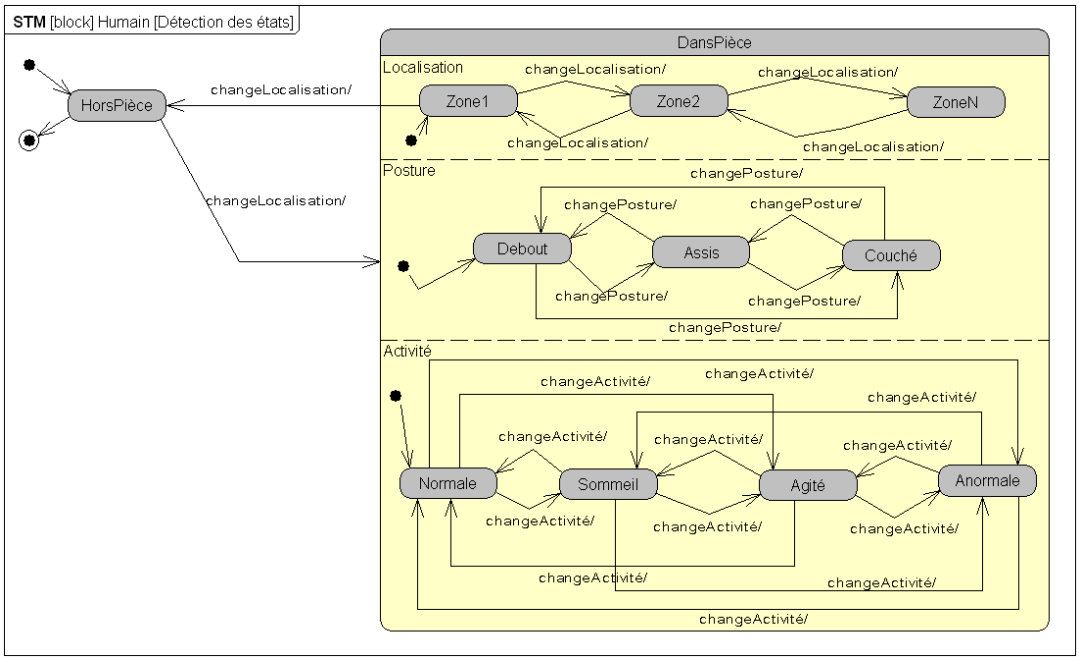


Figure IV.14 *Statemachine d'un humain*

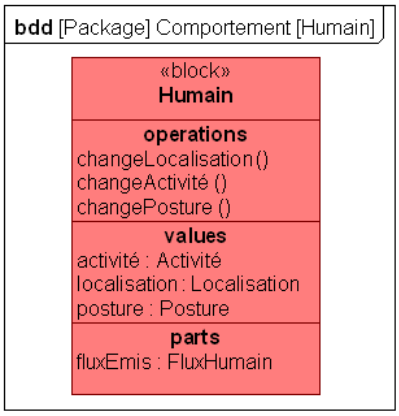


Figure IV.15 Block *Humain*

Chaque état de la statemachine modélise une valeur prise par les *value properties* de l'Humain. Les notations de parallélisme permettent de réaliser une notation compacte des différents états que capthom devra être capable d'identifier. On peut remarquer sur cette statemachine que les états de posture et d'activité ne sont pas détaillés hors de la pièce surveillée par le capthom. Ceci permet de mettre en évidence la zone d'intérêt du système, qui doit d'abord donner l'information de présence (la personne est à l'extérieur ou à l'intérieur de la pièce), avant de détailler le cas échéant l'état de la personne.

Afin de couvrir l'ensemble des scénarios, notamment ceux incluant des éléments perturbateurs, il faudrait considérer une combinaison des statemachines les représentant. Cependant, il reste impossible de faire apparaître des contraintes spatiales et géométriques telles que l'occultation du flux par des objets de la scène, et la représentation des scénarios reste donc sommaire. Néanmoins, on peut maintenant identifier les contraintes génériques d'émission des flux humains. En effet, la valeur de ces flux sera influencée par les changements d'états de l'humain : le changement de posture modifiera la surface d'émission, le changement d'activité pourra influencer l'intensité.

IV.3.6 Allocations d'exigences

Une fois ce premier modèle obtenu, nous avons pu réaliser l'allocation des exigences aux éléments structuraux et comportementaux du modèle. Ceci permet d'une part de voir si le modèle couvre l'ensemble des exigences et d'autre part de fournir des objectifs pour la conception détaillée des éléments du système. Le tableau IV.2 présente un extrait de la table d'allocation rassemblant l'ensemble des relations qui ont été déclarées entre les exigences et les éléments du modèle. La taille de ce tableau justifie à lui seul l'utilisation de services automatisés pour le suivi d'exigences comme nous le proposons au cours de l'étude AMDEC.

Tableau IV.2 *Extrait de la table d'allocation des exigences du projet CAPTHOM*

Name	Txt	Satisfied By
Alimentation et Consommation	L'alimentation acceptée pour les capthoms est comprise entre 3 et 5V. La puissance consommée doit être de l'ordre de 1 Watt.	«block» CapthomB (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomCAM (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomMCIR (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomPIR (Projet CAPTHOM::Alternatives Technologiques) «block» AlimentationElectrique (Projet CAPTHOM::Structure) «block» Capthom (Projet CAPTHOM::Structure)
Communication	Le Capthom doit communiquer ses résultats de détection au système superviseur de l'installation domotique.	«block» CapthomCAM (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomMCIR (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomPIR (Projet CAPTHOM::Alternatives Technologiques) «block» InterfaceDeCommunication (Projet CAPTHOM::Structure) «Activity» TransmettreSignal (Projet CAPTHOM::Comportement::Détection) «block» CapthomB (Projet CAPTHOM::Alternatives Technologiques) «block» Capthom (Projet CAPTHOM::Structure)
Identifiant	Chaque Capthom doit avoir un identifiant unique.	«block» CapthomB (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomCAM (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomMCIR (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomPIR (Projet CAPTHOM::Alternatives Technologiques) «block» InterfaceDeCommunication (Projet CAPTHOM::Structure) «Activity» TransmettreSignal (Projet CAPTHOM::Comportement::Détection) «block» Capthom (Projet CAPTHOM::Structure)
Protocole	Le capthom doit utiliser un des protocoles de communication suivants : ModBus, ZigBee ou BACnet.	«block» CapthomCAM (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomMCIR (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomPIR (Projet CAPTHOM::Alternatives Technologiques) «block» InterfaceDeCommunication (Projet CAPTHOM::Structure) «Activity» TransmettreSignal (Projet CAPTHOM::Comportement::Détection) «block» CapthomB (Projet CAPTHOM::Alternatives Technologiques) «block» Capthom (Projet CAPTHOM::Structure)
Conditions de Détection	Le Capthom doit fonctionner dans une habitation ou un bâtiment tertiaire.	«block» Capthom (Projet CAPTHOM::Structure) «block» CapthomB (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomCAM (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomMCIR (Projet CAPTHOM::Alternatives Technologiques) «block» CapthomPIR (Projet CAPTHOM::Alternatives Technologiques) «block» Transducteur (Projet CAPTHOM::Alternatives Technologiques) «block» Barrière (Projet CAPTHOM::Alternatives Technologiques) «block» Caméra (Projet CAPTHOM::Alternatives Technologiques) «block» MicroCapteurIR (Projet CAPTHOM::Alternatives Technologiques) «block» PIR (Projet CAPTHOM::Alternatives Technologiques) «Activity» Détection (Projet CAPTHOM::Comportement) «Activity» RéagirAuFlux (Projet CAPTHOM::Comportement::Détection) «Sequence Diagram» Sequence de détection (Projet CAPTHOM::Comportement)

Ces relations vont être étudiées avec la méthode MéDISIS, ce qui va nous permettre de cibler les points critiques du développement des capthoms et de dégager les besoins d'analyses spécifiques et de conception détaillée du projet.

IV.3.7 Utilisation de MéDISIS au sein du projet CAPTHOM

Lors de l'analyse du modèle avec MéDISIS, nous cherchons principalement à repérer les points critiques nuisant à la fiabilité de la détection de présence humaine. Par cette analyse, nous cherchons à guider les tâches de développement spécifiques et à mettre en évidence les éléments à étudier pour valider le système. Nous réalisons une AMDEC préliminaire issue de l'analyse automatique du modèle SysML de Capthom. Puis, nous réalisons l'AMDEC finale, de laquelle nous tirons les besoins d'analyse et les points critiques du système pour la réussite de sa mission.

L'usage des règles définies au chapitre II permet de dresser l'AMDEC préliminaire du projet CAPTHOM. Les éléments les plus significatifs recensés dans l'architecture des capthoms, sont les transducteurs. En effet, les transducteurs sont la source de la création du flux à partir duquel l'unité de traitement devra décider de la présence d'un humain dans la zone couverte par le capteur. Nous choisissons donc de présenter cet exemple significatif de l'AMDEC du projet complet. Une rapide analyse préliminaire des risques a permis de mettre en évidence que les deux principaux événements redoutés sont la fausse détection et l'absence de détection. Nous cherchons donc à évaluer par l'AMDEC les possibilités de défaillances des composants menant à ces événements.

L'utilisation des règles d'analyse du modèle SysML nous a donné la structure caractérisant le transducteur (tableau IV.3).

Tableau IV.3 *Structure créée pour le part « trans » modélisant le transducteur*

Type d'entité	**Eléments relevés**
Nom du *part*	Trans
Typing block	Transducteur
Owning block	Capthom
Operations	Détection Réaction
Constraints	Detect : DétectionFlux Réa : RéactionTransducteur
Composants amont (IBD)	Environnement [celluleSensible : Flux – Environnement] (fluxE : FluxEnvironnement) Utilisateurs [celluleSensible : Flux – Utilisateurs] (fluxH : Flux-Humain)
Composants amont (SD)	Environnement [Détection] Utilisateurs [Détection] Trans : Transducteur [Réaction]
Composants amont (AD)	Capthom [Détection] (entrée, fluxE) Capthom [Détection] (entrée, fluxH)
Composants aval (IBD)	Cond : Conditionneur [sortieTrans : SignalElectrique – entrée-Cond : SignalElectrique] (signalBrut) Environnement [celluleSensible : Flux – Environnement] Utilisateurs [celluleSensible : Flux – Utilisateurs]
Composants aval (SD)	Cond : Conditionneur [Conditionner(signal)] Trans : Transducteur [Réaction]
Composants aval (AD)	Conditionneur [ConditionnerSignal] (SignalBrut)
Exigences de niveau 1	Conditions de détection Conditions d'humidité Conditions de luminosité Thermique de l'environnement Nature de la détection Discrimination Informations Fournies Situation de détection Qualité de détection Fiabilité Portée Temps de réponse
Exigences de niveau 2	Alimentation et Consommation Dimensions Communication Durée de vie Esthétique Fixation Prix
Exigences de niveau 3	Identifiant Protocole

Nous constatons que la structure présente les nombreuses relations et propriétés du composant. On peut d'ores et déjà remarquer que la modélisation est cohérente au niveau des canaux

physiques mentionnés (IBD) et des relations définies (SD, AD). En effet, les interactions et communications entre activités (détection, réaction, demande de conditionner le signal) peuvent s'appuyer sur les ports définis dans l'IBD (celluleSensible, sortieTrans). Ces informations sont ensuite mises en forme pour constituer l'AMDEC préliminaire du système. Les deux MdD pris en compte pour les transducteurs sont la fausse détection et la non détection. Le tableau IV.4 donne l'extrait de l'AMDEC préliminaire proposée pour le transducteur « trans », représenté dans le diagramme exprimant la constitution générale d'un capthom (Figure IV.6).

Tableau IV.4 *AMDEC préliminaire du transducteur*

Composant	Fonction	MdD	Cause	Effet Local	Effet Exigence
Trans : Transducteur	Détection	Fausse Détection	Environnement [celluleSensible : Flux – Environnement] (fluxE : FluxEnvironnement)	Cond : Conditionneur [sortieTrans : SignalElectrique – entréeCond : SignalElectrique] (signalBrut)	Conditions de détection [1]
	Réaction	Pas de détection	Utilisateurs [celluleSensible : Flux – Utilisateurs] (fluxH : FluxHumain)	Cond : Conditionneur [Conditionner(signal)]	Conditions d'humidité [1]
			Environnement [Détection]	Trans : Transducteur [Réaction]	Conditions de luminosité [1]
			Utilisateurs [Détection]	Conditionneur [ConditionnerSignal] (SignalBrut)	Thermique de l'environnement [1]
			Trans : Transducteur [Réaction]	Detect : DétectionFlux	Nature de la détection [1]
			Capthom [Détection] (entrée, fluxE)	Réa : RéactionTransducteur	Discrimination [1]
			Capthom [Détection] (entrée, fluxH)		Informations Fournies [1]
			Detect : DétectionFlux		Situation de détection [1]
			Réa : RéactionTransducteur		Qualité de détection [1]
					Fiabilité [1]
					Portée [1]
					Temps de réponse [1]
					Alimentation et Consommation [2]
					Dimensions [2]
					Communication [2]
					Durée de vie [2]
					Esthétique [2]
					Fixation [2]
					Prix [2]
					Identifiant [3]
					Protocole [3]

Cette AMDEC préliminaire est ensuite reprise comme base de l'AMDEC finale des composants. Nous présentons (Tables 4.5 et 4.6) les résultats de cette AMDEC pour le transducteur. Dans ces tables, les éléments notés en noir sont issus de l'AMDEC préliminaire, quant à ceux notés en bleu montrent l'interprétation faite pour l'AMDEC finale

Tableau IV.5 *AMDEC du transducteur MdD 'Fausse détection'*

Composant	Fonction	MdD	Cause	Effet Local	Effet Exigence	Effet Système	Action corrective
Transducteur	Détection	Fausse Détection	Environnement [cellule-Sensible : Flux – Environnement] (fluxE : FluxEnvironnement) Hypersensibilité de la cellule sensible au flux recherché. Sensibilité aux autres flux de l'environnement.	Cond : Conditionneur [sortieTrans : SignalElectrique – entréeCond : SignalElectrique] (signalBrut) Transmission d'un signal de détection au conditionneur	Qualité de détection [1] Fiabilité [1] Baisse de la fiabilité de la détection due aux fausses détections	Un signal correspondant à une détection est analysé par la chaîne de traitement de capthom (conditionneur, unité de traitement). Puis il est relayé au système superviseur par l'interface de communication. Le capthom transmet une information erronée au système superviseur pouvant mener à une mauvaise décision. Perte de confiance au produit, perte des gains de consommation ou fausse alerte dans le cadre du maintien à domicile.	Analyser et améliorer l'isolation de la cellule sensible contre les flux non désirés de l'environnement (cf. bdd [Package] Environnement [Emetteurs de flux])
	Réaction		Environnement [Détection] Idem	Cond : Conditionneur [Conditionner(signal)] Idem	Nature de la détection [1] Discrimination [1] Il devient impossible de distinguer ce qui a sensibilisé le transducteur		Caractériser la zone couverte par capthom pour connaître les origines possibles de perturbation lors de l'implantation.
			Trans : Transducteur [Réaction] Auto sensibilisation (perturbations internes au transducteur).	Trans : Transducteur [Réaction] Réaction en l'absence d'humain			Caractériser les réponses aux fausses détections, pour élaborer des politiques de traitement robustes aux stimuli de l'environnement.
			Réa : RéactionTransducteur Hypersensibilité du transducteur.	Conditionneur [Conditionner-Signal] (SignalBrut) Transmission d'un signal de détection au conditionneur			
			Detect : DétectionFlux Trop de flux dans le champ de détection.	Detect : DétectionFlux Idem			
				Réa : RéactionTransducteur Réaction en l'absence d'humain			

Tableau IV.6 *AMDEC du transducteur MdD 'Pas de détection'*

Composant	Fonction	MdD	Cause	Effet Local	Effet Exigence	Effet Système	Action corrective
Transducteur	Détection	Pas de détection	Environnement [celluleSensible : Flux – Environnement] (fluxE : FluxEnvironnement) Perturbation par les flux de l'environnement, rendant la cellule insensible au flux humain.	Cond : Conditionneur [sortieTrans : SignalElectrique – entréeCond : SignalElectrique] (signalBrut) Pas de transmission d'un signal de détection au conditionneur.	Conditions de détection [1] Conditions d'humidité [1] Conditions de luminosité [1] Thermique de l'environnement [1] Si la non détection provient de l'environnement, les exigences de conditions environnementales ne sont pas forcément respectées.	Aucun signal correspondant à une détection n'est analysé par la chaîne de traitement de capthom (conditionneur, unité de traitement). Une absence de personne est relayée au système superviseur par l'interface de communication. Le capthom transmet une information erronée au système superviseur pouvant mener à une mauvaise décision. Perte de confiance au produit, mise en danger de l'utilisateur, confort non géré...	Analyser le phénomène de diffusion des flux de l'humain et son environnement. Déterminer les conditions de « masquage » de la présence humaine.
	Réaction		Environnement [Détection] Idem	Cond : Conditionneur [Conditionner(signal)] Idem			Caractériser la zone couverte par capthom pour connaître les possibilités d'implantation maximisant les détections.
			Utilisateurs [celluleSensible : Flux – Utilisateurs] (fluxH : FluxHumain) Insensibilité au flux humain, pouvant être plus faible que les valeurs nominales attendues (habillement, posture)	Trans : Transducteur [Réaction] Pas de réaction à la présence humaine.	Qualité de détection [1] Fiabilité [1] Baisse de la fiabilité de la détection due aux non détections.		Améliorer la sensibilité des cellules sensibles aux flux humains.
			Utilisateurs [Détection] Idem	Conditionneur [ConditionnerSignal] (SignalBrut) Pas de transmission d'un signal de détection au conditionneur			Etudier la faisabilité de la surveillance des scènes objectifs de CAPTHOM pour les caractéristiques de transducteurs envisagées.
			Trans : Transducteur [Réaction] Insensibilité du transducteur.	Detect : DétectionFlux Idem	Nature de la détection[1] Situation de détection [1] Si la non détection provient d'une posture ou d'un habillement de la personne empêchant la détection, les exigences de situation de détection ne sont plus respectées.		
			Réa : RéactionTransducteur Insensibilité du transducteur.	Réa : RéactionTransducteur Pas de réaction à la présence humaine.			
			Detect : DétectionFlux Pas assez de flux dans le champ de détection. (Vue trop faible des cibles)				

La partie d'étude du transducteur met en évidence (cf. colonne « action corrective) un fort besoin de détermination des caractéristiques des transducteurs testés, aussi bien en terme de sensibilité au flux de l'environnement et des humains, qu'en terme de caractérisation de la zone qu'ils couvrent. Pour cela, nous voyons qu'il est nécessaire de quantifier les propriétés, opérations et contraintes liées au transducteur. Le but des tests physiques déployés pour CAPTHOM est donc de quantifier pour les transducteurs, les éléments mis en évidence sur les diagrammes SysML des figures IV.8 et IV.9. L'AMDEC met en évidence les phénomènes à tester en faisant référence aux flux d'environnement « fluxE » de type FluxEnvironnement et au flux humain « fluxH ». Le responsable des tests peut alors se référer au diagramme BDD de la figure IV.10 sur lequel sont mentionnés les phénomènes relevés.

Ce type d'étude sur le transducteur nous permet de pointer les cibles d'amélioration possible. La mise en évidence des ports permet de mentionner qu'un travail peut être mené sur l'encapsulation de la cellule sensible pour s'affranchir des perturbations, sources de mauvaises ou fausses détections. La mise en évidence de la contrainte « Detect : DétectionFlux » renvoie aux travaux sur la mise au point de lentilles et de systèmes optiques venant conditionner le champ de vision des capteurs. La recherche des effets systèmes a mis en évidence le besoin de simulation de réseau global pour attester de la faisabilité de couverture des locaux représentant les objectifs d'implantation des capthoms.

L'analyse AMDEC globale du projet, portant sur un modèle au niveau de détail assez faible, a été précieuse pour le processus d'IS. En effet, l'AMDEC fait apparaître les éléments impliqués et leur impact sur la détection. Nous avons pu relever la suite d'éléments venant peser sur la formation de la réponse de capthom. Cela nous permet de cibler les leviers disponibles d'amélioration des performances du dispositif. Il peut s'agir d'améliorer l'unité de traitement, le conditionneur, le transducteur ou d'essayer de supprimer la relation avec l'Environnement, tout en conservant celle avec l'Humain. Nous avons ainsi pu définir deux axes d'amélioration de l'existant pour concevoir les capthoms :

- Amélioration de la chaîne de traitement du signal (« conditionneur » + « untité de traitement »).
- Amélioration du « transducteur » par une meilleure sensibilité aux flux humains et une ségrégation vis à vis des perturbations environnementales.

Au cours de MéDISIS, le but de l'AMDEC est de relever les comportements dysfonctionnels individuels et préparer leur intégration à un outil de validation finale. Dans le cas de CAPTHOM, cette validation finale est relative aux scénarios d'utilisation de CAPTHOM définis pour exprimer les besoins des partenaires. L'AMDEC nous a permis d'identifier les éléments à simuler dans le logiciel SNOOPS (présenté au § IV.4) pour réaliser cette validation. L'AMDEC a ici servi à prototyper l'application. Nous avions déjà utilisé une approche similaire pour la réalisation d'un simulateur de risque, pour lequel les éléments a reproduire ont été recensés en analysant les diagrammes de séquences en UML des procédures de gestion d'accident [Idasiak & David 2007].

L'AMDEC réalisée sur le modèle entier du système a donc parcouru les alternatives technologiques envisagées pour CAPTHOM (BDD figure IV.7). Chacune de ces alternatives est confrontée au besoin de fournir une réponse capthom (sortie de l'« unité de traitement ») reflétant l'exigence de richesse des informations fournies (exigence « Informations Fournies », figure IV.4). Chaque technologie a donc été qualifiée en terme d'informations fournies (position,

présence, posture, activité). Il en résulte pour les détecteurs PIR, l'apparition de l'impossibilité de détecter un humain immobile. Ce type de résultat permet de préparer le simulateur de réseau global en indiquant le type d'information relevé par chaque type de système, qu'il est souhaitable de simuler. De même, il faudra simuler les zones de détection des capteurs comme nous l'avons vu lors de l'analyse du transducteur.

Nous avons relevé de nombreux avantages dans l'utilisation de l'approche MéDISIS pour la réalisation de l'AMDEC. L'AMDEC a été réalisée rapidement, notamment grâce aux nombreux points d'interactions avec le modèle remontés dans la structure de l'AMDEC préliminaire. Cette mise en avant de l'information lors des points clefs de l'AMDEC contribue largement à sa qualité et à sa contribution au pilotage des développements. L'utilisation du service de création d'AMDEC préliminaire nous a permis d'obtenir très rapidement un premier document de travail. Ce dernier utilise les notations du modèle, ce qui permet de se référer rapidement aux éléments de modèle concernés, lors de la réalisation de l'AMDEC finale. La mise en évidence des exigences inscrites dans le modèle SysML a été par exemple très bénéfique lors de la détermination des principaux risques créés par les MdD sur la réussite du projet. Nous relevons ici que la formalisation des exigences telle que présentée dans SysML nous a permis de mettre en place une automatisation de leur rappel et de leur suivi dans une phase d'analyse de risque. Il nous a alors été possible de mettre en correspondance de façon significative les développements nécessaires pour leur respect. Enfin, il faut noter que la réalisation de cette AMDEC à travers un modèle SysML aide à la définition des tâches de développement du projet. Nous avons ainsi pu définir les tests de caractérisation nécessaires, les leviers d'amélioration possibles, les phénomènes physiques à comprendre ainsi que les besoins de simulation.

Les artefacts de modélisation SysML favorisent l'orientation efficace de la réflexion ; l'étude des éléments issus des IBD permet de pointer les éléments physiques prenant part à la mission et ainsi de formuler les pistes d'améliorations techniques. Les relations de propagation très précieuses pour la recherche des effets systèmes sont mises en évidence par les AD et SD. L'étude des exigences est très bénéfique pour la qualification et la quantification (bien que non présentée ici) des risques. En effet, la confrontation directe aux exigences donne des éléments très significatifs sur l'évaluation de la gravité des défaillances. Les informations issues des contraintes et du typage des flux ont permis de mettre en évidence les besoins de tests et caractérisations, mais également de spécifier le type de phénomènes et les ensembles de variation à prendre en compte.

IV.4 L'outil SNOOPS

Afin de répondre aux besoins d'analyse des réseaux de capteurs du projet CAPTHOM, nous avons développé le logiciel SNOOPS (Sensor Networks : Optimisation Of Placement and Simulation). La mission dégagée pour cet outil comporte les points suivants :

- Simuler les environnements d'emploi des capthoms.
- Estimer la couverture spatiale des capthoms.
- Modéliser les alternatives technologiques identifiées pour les capthoms.
- Simuler un réseau de capthoms et sa réponse en terme d'information sur les occupants.
- Simuler les scénarios d'emploi des capthoms.
- Optimiser le placement de capthoms dans un environnement.

Ce logiciel a pour objectif la validation des exigences formulées sous la forme des scénarios d'utilisation. Il s'intègre à l'EDS comme outil d'analyse spécifique et s'interface avec les outils de mise au point des algorithmes de fusion de données et de maintien à domicile.

Dans cette partie, nous allons présenter SNOOPS et sa réalisation. Après un bref état de l'art sur la réalisation de simulateur de placement de capteurs, nous détaillerons ensuite la modélisation utilisée pour les capteurs et leur environnement. Nous présenterons enfin les services de simulation de scénarios et d'optimisation du placement de réseaux de capteurs.

IV.4.1 Les travaux de placement de capteurs de détection

SNOOPS doit majoritairement répondre à des considérations spatiales : les capteurs couvrent-ils toutes les zones d'intérêt de la pièce ? L'effort de simulation ne s'est donc pas situé sur la restitution des phénomènes physiques de propagation de flux ou les mécanismes de sensibilisation des capteurs, mais sur la représentation des contraintes spatiales. La première priorité est donc de modéliser le placement des capteurs dans une scène donnée avec des objectifs d'observation préalablement fixés. Cette problématique est au cœur des activités de recherche sur le placement de caméras. Nous avons donc étudié les techniques employées pour le placement, puis nous avons œuvré à généraliser ces principes pour les différents types de capteurs que nous souhaitions simuler.

Les problèmes de placement de caméras ont été traités pour des domaines variés : la photogrammétrie, la vidéosurveillance, la navigation dans des univers virtuels ou la construction de plans pour le cinéma. Les approches les plus simples ont été employées dans le domaine du placement de caméras dans des scènes virtuelles, où l'objectif est d'aider l'utilisateur à placer et contrôler ses caméras dans l'environnement 3D [Philips et al. 1992], [State et al. 2006]. Ces derniers simulent différents placements de caméras dans un cas réel de scène d'opération chirurgicale et donnent à l'utilisateur des informations sur l'efficacité du réseau choisi : résolution, couverture. Dans cette étude, le placement est confié à l'utilisateur, [Lienhart & Horster 2006] traitent un problème similaire tout en optimisant le placement automatiquement. [Philips et al. 1992] se focalisent sur un placement manuel des caméras aidé par le calcul des champs de vision du dispositif. Les problèmes de photogrammétrie nécessitent des politiques de placement de caméras très précises. Ces techniques sont utilisées lorsque la forme des objets mesurés ou leur taille, interdisent l'usage des méthodes de mesure usuelles. Pour la mesure de grands bâti-

ments, il est notamment essentiel de simuler par avance le placement optimal du dispositif. On retrouve ainsi des travaux sur la mesure de pièces industrielles [Olague 2002], [Dunn et al. 2006] ou de bâtiments complexes [Saadat et al. 2004]. Le dernier domaine utilisant des simulations de placement de caméras est la conception de réseau de caméras de vidéosurveillance. Le but est de positionner de façon optimale le minimum de caméras afin de couvrir un environnement donné. Classiquement, ce problème est résolu en représentant l'environnement par une grille de couverture que le réseau de caméras doit respecter en minimisant le coût de l'installation [Gonzalez-Barbosa et al. 2009]. Nous avons utilisé cette technique dans [David et al. 2007], comme l'avaient fait [Chakrabarty et al. 2002], [Dhillon & Chakrabarty 2003], [Erdem & Sclaroff 2006]. Quand ces auteurs se sont focalisés sur l'utilisation de divers types de caméras, nous avons étendu ces principes aux différents types de capteurs utilisés pour CAPTHOM et proposé des méthodes de résolution adaptées.

IV.4.2 La modélisation dans SNOOPS

La modélisation des éléments simulés se construit autour de l'utilisation des grilles de couverture. Nous allons détailler la modélisation de la scène, des capteurs, des réseaux de capteurs ainsi que des scénarios de test.

IV.4.2.1 Modélisation de la scène

Notre problématique est de faire l'étude d'une scène qui, d'après le cahier des charges du projet, se présente comme un bâtiment tertiaire ou un ensemble de pièces d'habitation. Une scène décrit une zone que l'on veut surveiller et son environnement. C'est donc un ensemble d'éléments décrivant l'aspect physique de la pièce (forme, flux présent) couplé avec des informations sur les objectifs de détection du système de capteurs dans cette scène. La forme des pièces de la scène doit être décrite ainsi que l'emplacement et la forme de l'ameublement. Les perturbations des flux physiques doivent également être indiquées, ainsi que des informations permettant de décrire la mission du système. Cette mission est définie par un zonage de la scène indiquant diverses priorités de détection des différents espaces de la pièce. La description de la scène ne se limite pas aux zones à observer mais tient aussi compte de l'environnement proche, afin de modéliser certains problèmes, comme le fait de détecter au-delà de l'entrebâillement d'une porte. Les flux physiques seront représentés par la définition de différentes zones, pouvant être des obstacles, des zones interdites ou encore des zones prioritaires. Chacune de ces informations pouvant être définie indépendamment, elles sont quantifiées chacune par une couche propre reprenant la topologie de la scène.

La scène est ainsi définie par différentes couches successives qui forment au final l'environnement et la mission des systèmes de captures. La description complète passe par les étapes successives suivantes :

1. Définition des limites de la pièce et de l'environnement proche (murs, fenêtres, portes)
2. Définition du mobilier, des obstacles et zones de perturbation.
3. Définition de la mission du réseau de capteurs

Le premier point fixe la géométrie de la pièce et considère les espaces visibles à travers les portes et fenêtres pouvant être source de perturbation. Les obstacles sont ensuite définis de façon géométrique, il peut s'agir de mobiliers ou de tout autre type d'élément venant empêcher la

propagation des flux trahissant la présence humaine. Enfin, une cartographie des zones à surveiller dans la scène vient définir la mission. Des priorités de détection peuvent être affectées aux différents espaces.

La modélisation choisie suit l'utilisation de grilles d'occupation. La scène est représentée par une liste de points caractérisant une partie de l'espace. Elle est modélisée par un vecteur dont chaque élément correspond à un point de la scène. Les points sont numérotés dans l'ordre de balayage de la scène et donnent la matrice, par la suite noté **scene**. Cette matrice est porteuse de toutes les informations nécessaires, chaque élément étant un n-tuplé suivant le nombre d'aspects à refléter. Nous codons dans cette matrice tout le zonage inhérent à la mission. Dans un cas simple, sans priorités d'observations, les points à surveiller sont codés par un 1 dans l'élément qui porte leur indice et par un 0 s'ils ne représentent pas un objectif à couvrir. La figure IV.16 présente un cas simple où l'intérieur de la zone jaune représente les points à surveiller. Le vecteur **scene** possède 4 éléments à 1, représentant les 5,6,8 et 9ème points de l'environnement.

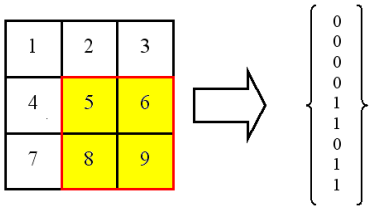


Figure IV.16 *Création du vecteur de description d'une scène*

La construction du vecteur représentant la pièce se fait en deux étapes. La première partie est la description mathématique de la scène par les sommets, arêtes et polygones modélisant les pièces et le mobilier. Ensuite la scène est discrétisée sous forme d'un vecteur dont les éléments portent une information sur la nature d'un point de la scène. On peut influer sur la finesse de la discrétisation en augmentant le nombre de points représentés dans le vecteur de la scène. Le vecteur de description de la scène est renseigné par des tests d'appartenance à des polygones. Pour chaque point appartenant à la discrétisation, on vérifie s'il appartient à l'ensemble des points à observer ou à ceux à éviter. Ceci nous permet d'obtenir le modèle numérique de la scène que nous pourrons ensuite exploiter pour développer l'algorithme d'optimisation choisi, ainsi que pour simuler des scénarios d'emploi de la scène.

IV.4.2.2 Modélisation des capteurs

La modélisation des capteurs est adaptée à la grille utilisée pour la scène. Chaque capteur implanté dans l'environnement est modélisé par un vecteur de même dimension que **scene**. Pour chaque capteur, on calcule son polygone de « vision » par tracés de rayons. Ce polygone représente la partie de la scène à partir de laquelle il reçoit les flux auxquels il est sensible. La détermination de ce polygone dépend à la fois de l'emplacement et de l'orientation du capteur (comme nous avons pu l'identifier sur le diagramme paramétrique de la figure IV.13), mais aussi des caractéristiques de profondeur de champ et d'azimut (identifiées sur le BDD de la figure IV.8). A partir de la localisation du capteur, l'algorithme de tracés de rayons projette des rayons de longueur dof(α) (« dof » : la fonction déterminant la profondeur de champ en fonction de l'azimut α) et calcule l'intersection avec les murs et obstacles de la scène. Chaque point d'intersection donne un sommet du polygone qui est ensuite numérisé en déterminant

l'appartenance de chaque point de la scène à celui-ci. On obtient le vecteur représentant le capteur de façon similaire à la création du vecteur **scene** illustré sur la figure IV.16.

Les différentes modélisations relatives aux différents types de capteur proviennent des variantes dans le calcul du polygone de détection, ainsi que dans la modélisation de la réponse du capteur en fonction de la position de la cible dans le polygone. L'important dans le cas d'étude de CAPTHOM est de pouvoir représenter la fiabilité de la réponse du capteur. La fiabilité en terme de détection est liée à la capacité de détecter le phénomène recherché quand celui-ci a lieu. Suivant la technologie employée, la fiabilité de la détection provient d'aspects extrêmement variés. D'une part comme nous l'avons vu par l'AMDEC de CAPTHOM, elle provient de chaque étage de traitement du flux, de sa captation à la décision de l'unité de traitement.

- En fonction des développements du projet, nous n'avons pas considéré la fiabilité caractérisant les capteurs au même niveau dans la chaîne de traitement. Pour les caméras et les capteurs PIR du commerce nous avons considéré l'information donnée à la sortie de l'unité de traitement. En effet, Les caméras et PIR ont fait l'objet de développements d'algorithmes de traitement de l'information dont nous avons intégré les résultats au simulateur.

- Pour les micro-capteurs infrarouges, nous avons considéré le signal fourni en sortie du conditionneur. Ceci est dû au fait que les travaux concernant l'utilisation des micro-capteurs infrarouges se sont focalisés au cours du projet sur l'amélioration de la cellule sensible et le conditionnement du signal. La fiabilité de l'information fournie par ces capteurs est grandement impactée par les problèmes de discrimination entre des signaux trahissant la présence humaine et le bruit de mesure. C'est pourquoi, nous avons intégré dans le simulateur la réponse en terme de signal électrique, qui d'après les tests de caractérisation menés, dépend de la distance et de l'angle de vue de la cible. Pour chaque point de la scène, on précise dans le vecteur de modélisation des capteurs la fiabilité de la réponse proposée par le capteur à cet emplacement. Cela pour une cible de dimensions et émission standards, définie dans les expérimentations du projet[27].

Chacun de ces capteurs est aussi caractérisé par son profil de détection. Nous présentons ici le profil d'une caméra, des détecteurs PIR muraux et omnidirectionnels, ainsi que des micro-capteurs IR testés. Ce profil est constitué de la forme du polygone de détection, de la fonction de fiabilité et du type d'information que peut contenir la réponse du capthom (position, posture, activité). Ces données sont directement liées aux résultats de caractérisation des capteurs, gérés grâce au modèle SysML. En effet, comme mentionné au paragraphe IV.3.4, ces paramètres sont les caractéristiques des alternatives de *blocks* « transducteur ». Les profils de réponse des capteurs que nous allons présenter ici sont des instanciations des diagrammes paramétriques du paragraphe IV.3.5.3.

- La caméra est comme nous l'avons indiqué au paragraphe IV.2.2.2 le plus performant des capteurs de présence humaine, en ce qui concerne la richesse des informations fournies. Dans SNOOPS, nous considérons la caméra comme le système complet incluant la matrice de cellules photosensibles, le système de création du flux vidéo et l'unité de traitement implémentant les algorithmes proposés dans [Benezeth et

[27] Travaux conduits par Antoine Belconde dont la thèse sera soutenue au premier semestre de l'année 2010.

al. 2008], [Benezeth et al. 2009] et [Brulin et al. 2009]. Ainsi une caméra fournit une réponse donnant la présence, la position de la cible dans la scène, sa posture et son activité. Ces traitements étant adaptés uniquement aux caméras fixes directionnelles standards, nous ne traitons pas les caméras mobiles comme [Erdem & Sclaroff 2006] ou les caméras omnidirectionnelles de [Gonzalez-Barbosa et al. 2009]. Ainsi le polygone de détection utilise une fonction dof(α) constante. Cette constante et l'intervalle de variation de α dépendent du modèle choisi. Ces paramètres sont réglables dans SNOOPS. En ce qui concerne la fiabilité de la réponse, les études menées n'ont pas permis de montrer une corrélation entre position dans le champ de vision et taux de non-détection. La fiabilité de la réponse est donc considérée constante.

- Les détecteurs PIR sont les seconds capteurs disponibles dans SNOOPS. Ils ne trahissent que la présence d'une personne en mouvement dans leur champ de vision. La zone couverte par ces capteurs dépend de la lentille de Fresnel utilisée. Pour les dispositifs muraux, la géométrie de la zone est similaire au cône de détection d'une caméra avec une profondeur de champ fixe. Cependant, l'ouverture du dispositif se situe usuellement entre 90 et 180 degrés. Pour les installations en plafonnier, la lentille permet une sensibilité omnidirectionnelle. La forme de la zone couverte projetée sur le plan de la scène est elliptique. La fonction dof(α) est alors de la forme : $dof(\alpha) = \frac{K_1}{\sqrt{1 - K_2^2 \cos^2 \alpha}}$; avec K_1 et K_2 les constantes de forme de l'ellipse. Comme pour les caméras, la fonction de fiabilité est jugée constante dans l'espace délimité par le polygone de détection. Cependant, elle exprime seulement la probabilité de détecter quelqu'un en mouvement.

- Le dernier type de capteur est celui dont la partie physique fait l'objet des plus larges développements. Il s'agit de micro-capteurs IR, fonctionnant comme des thermocouples sensibles aux radiations IR. Ces dispositifs fournissent une information sur la présence humaine. Ils sont équipés de lentilles pouvant faire varier la forme du polygone de détection. La version qui est utilisée pour CAPTHOM dispose d'un champ de vision conique avec un angle d'ouverture de 60 degrés et une profondeur de champ de 4 mètres. La fiabilité du capteur étant liée à la détectabilité du signal nous utilisons une fonction de fiabilité proportionnelle au signal émis. Le signal (U en Volt) suit la forme suivante avec d la distance à la cible et α l'azimut (l'obtention de cette formule sera présentée dans la thèse d'Antoine Belconde) :

$$U = \frac{0{,}26(\alpha^2.10^{-4} + 2{,}1\alpha.10^{-3} - 0{,}9177)}{d + 0{,}58} + 2{,}44$$

IV.4.2.3 La saisie de la scène et des capteurs dans SNOOPS

L'interface graphique de SNOOPS est constituée de diverses zones de saisie afin de spécifier la scène et les capteurs utilisés. La spécification des scènes est abordée comme la construction d'un plan d'architecture du bâtiment constituant la scène. Les diverses pièces sont décrites par l'utilisation de polygones fermés dont les ouvertures percées par des portes ou fenêtres sont mentionnées dans un second temps. Les obstacles physiques à la propagation des flux humains sont ensuite placés dans la scène et caractérisés également par des polygones. SNOOPS fournit un aperçu de la scène déclarée afin de faciliter cette définition.

Nous donnons en exemple le plan d'un site pilote de CAPTHOM : l'appartement témoin d'une maison de retraite. Le plan obtenu avec SNOOPS est donné figure IV.17. Nous avons considéré comme des ouvertures les portes situées à l'intérieur de l'appartement. La porte d'entrée a été laissée fermée comme ce sera le cas pendant les scénarios de détection. Les placards muraux ainsi que les éléments de la cuisine intégrée ont été considérés comme les limites de la scène et sont donc figurés comme des murs. Nous avons annoté le plan fourni afin de détailler le type de pièces présentes, ces indications ne sont pas saisies dans SNOOPS.

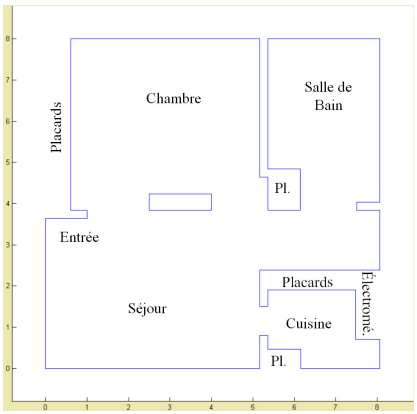
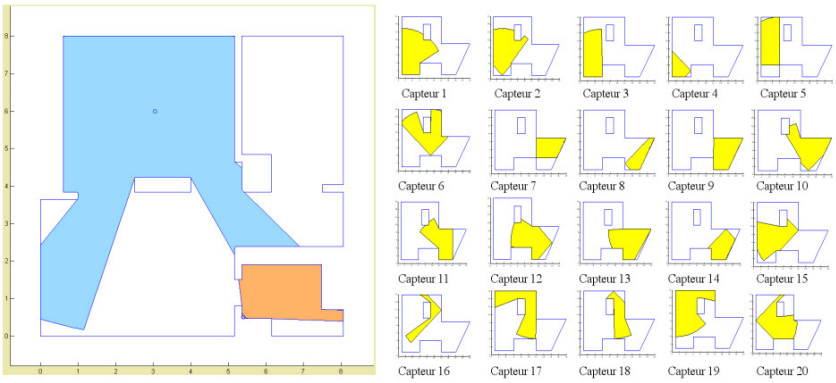


Figure IV.17 *Plan de l'appartement témoin dans SNOOPS*

La simulation des capteurs débute par leur implantation dans la scène et le calcul de leurs performances dans un tel environnement. SNOOPS offre la possibilité de spécifier les capteurs en sélectionnant leur type parmi une liste reflétant les capteurs présentés au paragraphe IV.4.2.2. Les capteurs sont caractérisés par les paramètres identifiés dans le modèle SysML, à savoir leur position, orientation, profondeur de champ et angle d'ouverture. Une représentation de la zone couverte par les capteurs est obtenue par tracés de rayon. Cette opération permet d'obtenir le polygone de détection ensuite utilisé lors du calcul des vecteurs de modélisation des capteurs. Un exemple de placement de deux capteurs dans l'appartement témoin est illustré figure IV.18a. Le premier capteur (en bleu) est un PIR placé au plafond de la chambre, le second (en orange) est un micro-capteur IR placé à l'entrée de la cuisine. Afin de préparer les phases d'optimisation et de simulation de réseaux de capteurs, les multiples capteurs utilisables sont saisis en envisageant, par exemple, les multiples possibilités d'orientation pour un même emplacement, comme montré à la figure IV.18b.



(a) **(b)**

Figure IV.18 *Exemple de placement de 2 capteurs*

Ces déclarations de la scène et des capteurs permettent d'obtenir les éléments d'entrée nécessaires à la construction du modèle mathématique du problème. La structure de données créée est constituée des différents vecteurs de discrétisation de la scène et des capteurs suivant la modélisation proposée au paragraphe IV.4.2.

IV.4.3 La simulation de scénarios et la réponse d'un réseau de capteurs

Le modèle SysML de CAPTHOM, son analyse par une AMDEC et l'élicitation des exigences des partenaires ont révélé les besoins de validation et de conception spécifique des futurs capthoms. Ces besoins sont nés de la difficulté de recréer réellement l'ensemble des scénarios auxquels doivent répondre les capthoms. La solution logicielle développée doit permettre de simuler le comportement des capteurs envisagés pour CAPTHOM vis-à-vis des sollicitations spécifiées dans le CdC. Ses réponses sont utiles pour juger de la capacité des capteurs à couvrir des cas d'emploi des scènes établies et pour permettre la création de politiques de traitement des informations fournies par les capteurs. Ces techniques de fusion de données ont pour but d'améliorer les connaissances fournies par les capteurs, grâce à leur mise en réseau, et l'application de ces résultats pour le maintien à domicile des personnes dépendantes.

Le niveau d'analyse du comportement des capteurs est donc fixé à la description de la signification de leur réponse en matière de présence humaine. La simulation à mener ne concerne donc pas la physique des détections, mais leurs résultats en terme de richesse des informations fournies par les capteurs. Les scénarios définissant les missions des capthoms sont exprimés de manière séquentielle, le système d'étude est donc à considérer comme un système à événements discrets. Pour chacune des phases de ces scénarios, on considère un changement d'état de l'humain occupant la pièce, un changement d'état de détection des capteurs tout en évaluant, pour des capteurs comme les thermopiles, une fonction linéaire indiquant leur réponse.

La modélisation des scénarios permet de reprendre les éléments mis en évidence pendant l'élicitation des exigences (§IV.2.2.1). Les aspects de description de la scène (encadré de la fiche de scénario figure IV.1) sont déjà présents dans la saisie de la scène. Les scénarios à tester sont discrétisés en phases. Pour chacune de ces phases, on définit la position des occupants de la scène, leur posture (debout, assis, allongé) et leur activité (marche, course, occupation calme, agitation) ; on doit connaître également les phénomènes caractérisant l'environnement, tels que la température de la zone. Pour chaque phase, l'information recherchée est spécifiée, ces critères permettent de juger de la réussite du suivi du scénario par le réseau choisi. Pour chaque phase et chaque capteur, l'algorithme vérifie si la cible est dans la zone de détection de chaque capteur grâce au vecteur exprimant sa zone de couverture. Il relève alors le signal associé au point où se trouve la cible. La réponse de chaque capteur est ainsi créée et enregistrée dans une matrice dont les colonnes représentent les phases du scénario et chaque ligne un capteur du réseau. Cette matrice constitue le résultat du test et peut être exploitée pour simuler des politiques de traitement de l'information. Pour répondre au besoin d'évaluation de la fiabilité de l'information fournie par le capteur et le réseau (ciblé lors de l'AMDEC), nous donnons également un indicateur sur la fiabilité des réponses données. La figure IV.19 donne un exemple de scénario dont le tracé et la réponse du réseau de capteurs sont indiqués. Il s'agit d'un scénario en 10 phases dans l'appartement témoin, pour lequel un micro-capteur (capteur 1, en orange) et un PIR plafonnier (capteur 2, en bleu) sont considérés. La personne est en mouvement dans la pièce, à l'exception de la phase 8 durant laquelle elle est immobile. Les réponses des capteurs,

après simulation, sont données en bas de la figure. On peut remarquer la perte d'intensité de la réponse du micro-capteur, à mesure que la cible s'éloigne, avant une rupture totale lorsque la cible sort de son champ de détection. Le PIR perd la cible de vue à deux reprises : durant la phase 4 lorsque la personne sort de la zone couverte et durant la phase 8 lorsque la cible est immobile.

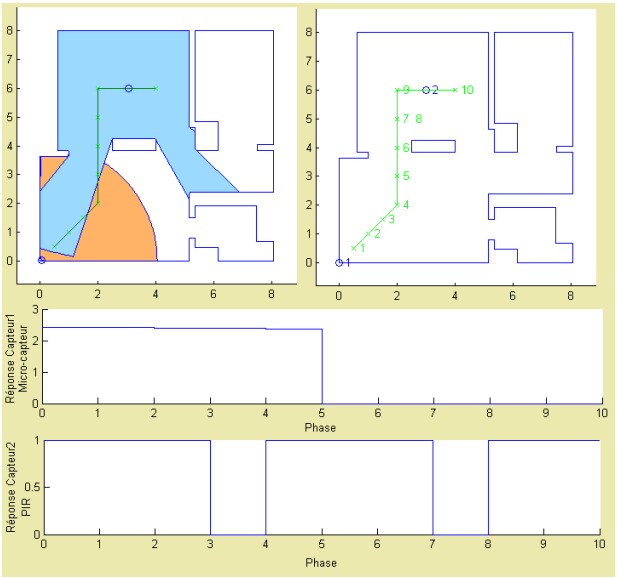


Figure IV.19 *Exemple de scénario et réponse du réseau*

IV.4.4 L'optimisation du placement de réseaux de capteurs

Avec pour objectif la mise en place de tests réels des capteurs, nous avons été confrontés à la problématique de l'optimisation du placement de réseaux de capteurs. L'optimisation a pour but de trouver le meilleur réseau en terme de type de capteurs employés et de leur positionnement, afin d'optimiser un critère d'évaluation de la solution proposée. Ce critère dépend de multiples paramètres incluant le prix du réseau, le type d'information qu'il est possible de déduire, le taux de couverture de la scène, la fiabilité de la solution et d'autres éléments tels que le nombre de capteurs observant un point donné. Le problème est donc de proposer le placement optimal pour la surveillance d'une scène avec des critères de mission variés. Cette fonctionnalité centrale de SNOOPS permet de fournir une aide à l'implantation d'un réseau de capteurs de présence humaine dans des environnements complexes.

En matière de détection d'humains, ce type de solution a été plus généralement utilisé pour la conception de réseaux de caméras de vidéosurveillance. Dans [Bares et al. 2000], l'auteur définit les quatre propriétés fondamentales d'un système de placement de capteurs de détection de présence. Ce sont : la capacité à calculer un placement différent et adapté à chaque requête de l'utilisateur, la capacité d'effectuer le traitement pour des environnements complexes, la non-modification de la scène pour répondre aux contraintes et la capacité de toujours proposer la solution la meilleure si la solution idéale n'existe pas. Ce module de SNOOPS a été détaillé dans [David et al. 2007].

IV.4.4.1 Formulation du problème de placement

Le mécanisme de résolution d'un tel problème repose sur la définition d'une fonction d'évaluation à optimiser, sur la formulation des contraintes liées au problème et sur la spécification d'une méthode de résolution du système défini. Nous avons considéré comme données d'entrée au problème : la forme, l'ameublement et le zonage de la scène (priorités de détection) ainsi que les positions et orientations possibles pour les capteurs utilisables dans la scène. Toutes ces données forment les contraintes de ce problème d'optimisation. Le zonage de la pièce est ce que nous appelons la mission du système. Dans les travaux utilisant les grilles d'occupation ([Erdem &Sclaroff 2006], [Gonzalez-Barbosa et al. 2009]), le mode opératoire est similaire. Cependant, la recherche est uniquement tournée vers l'obtention de solution assurant 100% de couverture de la mission définie sans considérer de priorités d'observation. De plus, ces travaux se limitent à l'utilisation de caméras et non à d'autres types de systèmes de surveillance. Nous avons donc également mis au point les représentations des capteurs PIR et thermopiles sous forme de grilles d'occupation.

Pour résoudre le problème de placement de capteurs, nous pouvons nous placer dans le cas classique de problèmes de programmation linéaires sous contraintes. C'est-à-dire que le problème peut être formulé sous la forme :

Minimiser $f(x) = c^T x$, $x \in [0,1]$, tel que

$$A \times x > b \qquad (1)$$
$$ub \geq x \geq lb$$

La construction des modèles de la scène et des capteurs est adaptée à cette formulation. En effet, dans ce contexte, un point de la scène est vu par le système de capture si sa valeur dans le vecteur de détection est positive. De même, nous définissons un vecteur de mission dans lequel les points qui doivent être observés possèdent une valeur positive, contrairement à l'addition des vecteurs de détection de chaque composant qui permet la représentation du système de capteurs et donne le vecteur de sélection, ce dernier indique alors les capteurs utilisés, parmi ceux de la liste initiale, pour satisfaire la mission du système. Il a pour dimension le nombre de capteurs pouvant être installés et porte un 1 pour l'élément représentant un composant sélectionné. Dans la formule (1), le vecteur « b » est donc le vecteur de description de la scène (**scene**), le vecteur « x » est celui de sélection. La matrice « A », de dimension Nombre de Point $\times$ Nombre de capteurs possibles, est construite en concaténant à droite tous les vecteurs de détection des capteurs candidats dans le même ordre que celui utilisé pour le vecteur de sélection. Ainsi, on peut vérifier pour chaque point de la scène s'il apparaît dans les points vus par le système, c'est à dire si $A \times x > b$. Le processus est schématisé par la figure IV.20.

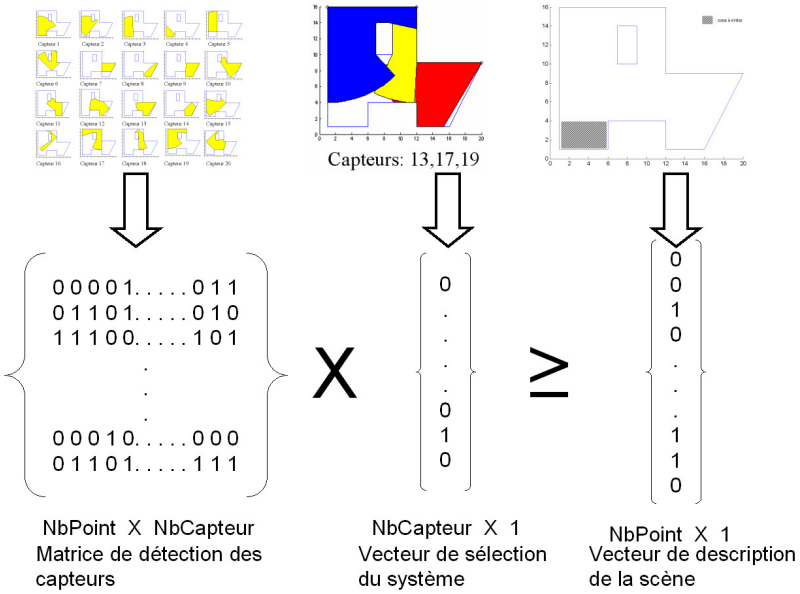


Figure IV.20 *Formulation simplifiée du problème de placement*

La clé de l'optimisation réside dans la construction de la fonction de coût « f ». Cette fonction permet d'introduire des aspects comme le coût réel lié au prix du matériel employé, de l'installation, de la consommation en énergie. Elle peut également représenter une qualité du matériel ou d'autres évaluations plus empiriques. La fonction finale peut pondérer ces différents aspects pour former une fonction multicritère paramétrable en fonction de la politique d'installation souhaitée. Cette fonction donne, pour chaque capteur possible, le coût global de son installation et de son exploitation, les différents paramètres doivent être fixés en fonction des priorités souhaitées. On peut donner comme exemple de priorité, une basse consommation d'énergie qui conduirait à affecter un coût très élevé aux systèmes très consommateurs d'énergie. Si l'on souhaite au contraire simplement minimiser le nombre de systèmes utilisés, il suffit d'affecter un poids unitaire à tous les capteurs. Pour résoudre le problème d'optimisation précédemment posé, différents usages de la programmation linéaire ont été employés [Erdem & Sclaroff 2006], [Gonzalez-Barbosa et al. 2009]. Nous avons utilisé dans un premier temps, un algorithme de branch and bound pour valider la modélisation, puis nous avons utilisé des algorithmes génétiques offrant plus de flexibilité et permettant de considérer des objectifs plus complexes comme l'équilibre entre le coût de l'installation et son efficacité.

IV.4.4.2 L'utilisation d'algorithmes génétiques pour l'optimisation du placement des capteurs

Les algorithmes génétiques ont été déjà utilisés par le passé pour des problèmes de placement de caméras [Olague 2002], [Dunn et al. 2006]. Pour notre problème, nous avons utilisé un algorithme génétique binaire dont le fruit de la recherche est le vecteur de sélection des capteurs. Nous avons travaillé principalement sur la fonction d'adaptation pour réaliser différentes politiques d'optimisation. Nous avons utilisé une fonction de sélection dont la probabilité de sélection est proportionnelle à l'adaptation de l'individu, basée sur la loi géométrique, pour conserver une sélection en partie aléatoire et ainsi contribuer à l'évitement des optimum locaux.

Suivant les propriétés recherchées, les algorithmes de sélection de capteurs peuvent être diversement évalués. Les qualités recherchées peuvent être, la rapidité d'obtention des résultats,

le nombre de résultats proposés, la variété de ces derniers ou la correspondance à des exigences chiffrées. Dans notre cas, ces exigences concernent plus particulièrement l'équilibre à donner entre le coût associé à l'implantation d'un nouveau système de capteurs et le gain en efficacité de la nouvelle solution. L'efficacité concerne dans nos premiers cas la couverture de la mission.

Nous avons d'abord opté pour des politiques de fonction d'adaptation visant des taux de couverture de la mission acceptables en rapport au coût ou à la complexité de l'installation du système de capteurs. Nous avons considéré 4 principaux paramètres à fixer pour obtenir des résultats satisfaisant nos attentes. Celles-ci consistent en l'évitement maximum des zones à exclure, le non-dépassement du nombre maximum de capteurs souhaités dans une solution (fixé par l'utilisation de l'algorithme Branch and Bound) et l'équilibre entre le gain de couverture d'un nouveau capteur et son coût. Ceci signifie que nous fixons pour chaque politique utilisée un pourcentage de couverture minimale supplémentaire que doit induire un nouveau capteur pour être légitimement mis en place. Pour influer sur les trois aspirations que nous venons de décrire, nous avons construit la fonction d'adaptation autour de 4 paramètres prenant la forme, en terme additif, de bonus ou de malus. L'algorithme génétique que nous utilisons cherche à trouver une valeur maximale pour le résultat de la fonction d'adaptation. La fonction d'adaptation (figure IV.21) prend en entrée les matrices A (matrice de détection des capteurs) et a (vecteur de description de la mission), représentant respectivement la visibilité des capteurs et l'ensemble des points à observer dans la scène. Sont également considérés en entrée, le vecteur x de sélection des caméras (x est la solution recherchée), le nombre de points de la scène « NbPoint », le nombre de capteurs utilisés dans la solution du Branch and Bound sous condition de l'existence de cette dernière « MaxCapt », le nombre maximum de capteurs dans la scène « NbCapt » et la valeur de couverture supplémentaire que doit apporter un nouveau capteur « Couv ». « val » est la valeur qui sera donnée à la fonction d'adaptation pour un individu x donné. La fonction « sum » est la somme de tous les éléments d'une matrice donnée.

Ces différents principes s'appliquent à la fonction d'adaptation et favorisent l'émergence de solutions avec un taux de couverture important et un nombre réduit de capteurs. Elle permet d'accéder à un équilibre entre zone couverte et nombre de capteurs utilisés. En effet, une solution avec un taux de couverture plus important n'est pas obligatoirement mieux notée qu'une solution avec moins de capteurs. Des résultats de l'exploitation de cette méthode sont présentés dans [David et al. 2007] et montrent la convergence de l'algorithme, ainsi que l'obtention de résultats plus fins que ceux obtenus par programmation linéaire.

Si lors de l'exploitation de SNOOPS, nous désirons aborder l'aspect fiabilisation du système de détection par redondance de couverture, nous devons introduire de nouveaux concepts dans l'élaboration de la fonction d'adaptation. Nous avons donc en particulier cherché à proposer des solutions pour lesquelles chaque point de la mission est vu par N capteurs, permettant de réaliser une triangulation pour déterminer la localisation des cibles. L'utilisation de l'optimisation de placement est réalisée en utilisant l'interface « Mission » du simulateur. L'utilisateur peut saisir la mission du réseau à travers la définition des zones-objectifs et de leur degré de priorité (de 0 à 3). Ensuite, l'utilisateur fixe les paramètres de l'algorithme génétique pour spécifier le type de réponse attendue (détermination de la présence, de la position, de la posture ou de l'activité). L'utilisateur peut également fixer l'équilibre entre l'implantation d'un nouveau capteur et les connaissances que ce capteur apporte. Le logiciel donne alors la liste des capteurs choisis et propose un tracé de la solution (Figure IV.22).

```
val ← 0
pour i=0..NbPoint
    si  a ( i ) > 0                      on s'intéresse au point à surveiller
        si (A × x)( i ) ≥ a( i )         si le point est vu par les capteurs
        val ← val + 100 / sum( a )       cette fraction représente le pourcentage de la
                                         zone à observer que couvre un de ses points
    si (A × x)( i ) < 0                  si un point à éviter est dans le champ des capteurs
        val ← val – 100                  on pénalise très fortement la détection d'un point
                                         interdit
si sum( x ) > MaxCapt                    si le nombre de capteurs utilisés est supérieur au
                                         nombre de capteurs maximum souhaités
    val ← val – 100                      on pénalise fortement cet excès

val ← val + Couv × ( NbCapt – sum( x ) ) Un capteur supplémentaire coûte autant que la
                                         récompense du nombre suffisant de points pour
                                         couvrir Couv % de la pièce
```

Figure IV.21 *Exemple de fonction d'adaptation utilisée*

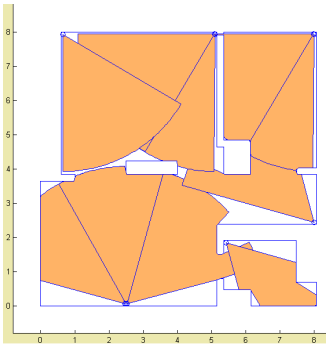


Figure IV.22 *Exemple de placement dans l'appartement témoin*

IV.4.4.3 Bilan de l'outil SNOOPS

L'outil SNOOPS permet d'évaluer par simulation, l'utilisation des capteurs développés dans le projet. Celle-ci prend place dans le cadre d'emploi final d'un réseau domestique pour l'habitat intelligent. Le logiciel répond aux besoins particuliers d'analyse exprimés en terme de surveillance d'un environnement. Ces derniers incluent des contraintes spatiales et des contraintes sur le comportement des cibles. Cet outil prend sa place dans l'EDS du projet en intégrant les résultats des caractérisations réelles des capteurs et en permettant de valider la réussite des scénarios exprimant les besoins des parties prenantes. Il permet d'analyser l'utilisation des solutions individuelles proposées par les développeurs. La solution proposée pour la modélisation des capteurs est évolutive et permet de prendre en compte rapidement les avancées des développements techniques. Il est possible de représenter des technologies hétérogènes en se focalisant sur le type d'informations données, mais ceci n'interdit pas de considérer des réponses plus basiques sur la réponse du capteur, comme cela a été le cas pour les micro-capteurs IR. En effet, la modélisation adoptée nous a permis de représenter des capteurs

vus par la sortie de l'unité de traitement (PIR, caméra), comme des capteurs limités à la réponse des étages « transducteur » et « conditionneur » (micro-capteur IR).

IV.5 Conclusion

Dans ce chapitre, nous avons proposé le déploiement des techniques d'Ingénierie Système Basée sur les Modèles et de MéDISIS pour traiter la problématique du projet CAPTHOM. Nous avons ainsi pu approuver l'utilisation de SysML dans le cadre d'un projet industriel réunissant différents partenaires, ainsi que l'utilisation du service de génération d'Analyse des Modes de Défaillance de leurs Effets et leur Criticité (AMDEC) présenté au chapitre II de cette thèse et les bénéfices d'une AMDEC précoce dans le processus de développement pour la détection de besoins d'analyse, de développement et de simulation. Nous avons enfin présenté le logiciel SNOOPS que nous avons développé pour les besoins d'analyse et de déploiement des systèmes de détection de présence conçus au cours du projet CAPTHOM.

Nous avons présenté le projet CAPTHOM et ses spécificités pour la recherche de détecteurs innovants de présence humaine. Nous avons employé le langage SysML, afin de modéliser une première architecture du système, à partir de laquelle, nous avons proposé une solution de modélisation des alternatives technologiques et des phénomènes physiques en relation avec le transducteur du capteur. Nous avons ainsi pu illustrer l'emploi de SysML pour la modélisation d'un système physique et insister sur les étapes de gestion des exigences et d'allocation du comportement à la structure. Les artefacts de modélisation de SysML nous ont permis d'aborder toutes les facettes nécessaires à l'expression de la problématique de CAPTHOM, des exigences à l'expression des contraintes exprimant la détection et la définition des différents comportements du système.

Nous avons ensuite illustré l'utilisation de MéDISIS à partir du modèle CAPTHOM. L'AMDEC préliminaire du projet a été réalisée suivant les règles définies au chapitre II. Ce document a servi de base à l'élaboration de l'AMDEC finale. En s'appuyant sur l'analyse du transducteur de CAPTHOM, nous avons illustré les résultats obtenus. Nous avons constaté, avec cette étude, l'aide apportée par MéDISIS au cours du processus de développement de CAPTHOM. L'outil de réalisation d'AMDEC préliminaire nous a permis d'obtenir rapidement une AMDEC de qualité. Cette dernière a permis de dégager les besoins de caractérisation précise des technologies testées, mais aussi les éléments du système sur lesquels concentrer les efforts de développement. Enfin, cette AMDEC fut utile pour identifier les éléments à intégrer au sein de SNOOPS.

La dernière partie de cette application au projet CAPTHOM a donc été la présentation du logiciel spécialement développé pour la validation de l'emploi des capthoms dans un réseau domestique. Nous avons présenté cette solution spécifique au projet intégrée à l'Environnement de Développement Système de CAPTHOM. Le logiciel est en effet conçu grâce aux observations menées sur le modèle SysML et fournit des bases de travail pour la mise au point des politiques de fusion et de maintien à domicile. SNOOPS permet de simuler l'emploi des différentes alternatives technologiques envisagées pour CAPTHOM. Il propose un service de simulation de scénarios d'utilisation générant les réponses données par un réseau de capteurs confronté à une utilisation spécifiée de la scène modélisée. Il intègre aussi une fonction d'optimisation de placement de capteurs, employée pour la mise en place des essais en situation des capteurs dans l'appartement témoin utilisé pour le projet. La modélisation employée dans le logiciel utilise les grilles d'occupation et a permis de représenter les aspects de couverture spatiale de chaque capteur. Ces grilles ont également intégré le calcul des réponses des capteurs issues des essais de caractérisation réels. Pour le service d'optimisation de placement, nous avons employé une solution à base d'algorithmes génétiques permettant l'adaptation aux différentes stratégies pos-

sibles d'implantation, ainsi qu'à l'intégration de contraintes hétérogènes, tels que des coûts d'installation et l'efficacité de la solution.

Ce chapitre a donné un exemple d'emploi concret de l'Ingénierie Système Basée sur les Modèles et de l'utilisation de MéDISIS dans un tel processus. L'exemple proposé a ainsi illustré l'emploi et l'analyse de modèles SysML, le déploiement concret de la méthode et son utilisation en lien avec un outil d'analyse spécifique. Cette relation a été développée jusqu'à l'aide au prototypage de la solution finale d'analyse et validation des futurs capthoms.

Conclusion Générale

La conception des nouveaux systèmes à fortes contraintes de Sûreté de Fonctionnement demande, par leur complexité technologique et dimensionnelle, ainsi que par les exigences contractuelles des parties prenantes à leur développement, l'emploi de méthodes d'Ingénierie Système avancées. Il est précieux d'appréhender la conception de tels systèmes en employant l'Ingénierie Système Basée sur les Modèles. Ces techniques organisent la conception autour de modèles renforçant l'expressivité et la cohérence des descriptions du système. L'Ingénierie Système Basée sur les Modèles s'appuie sur un ensemble d'outils permettant une gestion efficace de l'information portée par les différents modèles. Ces outils et les répertoires qui les accompagnent doivent former un Environnement de Développement Système cohérent, afin de permettre une conception efficace. Les analyses de Sûreté de Fonctionnement nécessaires à l'élaboration de systèmes embarqués ou à mission de sécurité doivent être intégrées au processus d'Ingénierie Système, sans pour autant ralentir celui-ci. Les techniques, modèles et outils spécifiques utilisés pour ces analyses doivent être intégrés le plus efficacement possible aux Environnements de Développement Système employés. Cette cohésion des outils est actuellement loin d'être effective et de nombreux développements sont nécessaires afin d'intégrer au mieux les analyses de Sûreté de Fonctionnement au processus d'Ingénierie Système Basée sur les Modèles. C'est à cette problématique que cette thèse a apporté sa contribution.

Nous avons fait l'hypothèse d'un processus d'Ingénierie Système Basée sur les Modèles utilisant le langage SysML. Nous avons montré dans le chapitre I, que ce langage satisfaisait aux besoins de développement des systèmes physiques, complexes et multi-technologiques à fortes contraintes de Sûreté de Fonctionnement. Après un état de l'art sur l'Ingénierie Système et les analyses de Sûreté de Fonctionnement nous avons pu dégager une méthodologie d'intégration des études de Sûreté de Fonctionnement au processus d'Ingénierie Système. Cette méthode, nommée MéDISIS, organise l'analyse de Sûreté de Fonctionnement, de la recherche des défaillances particulières des composants, à l'évaluation du comportement dysfonctionnel du système global. Nous proposons à travers elle une succession de traitements des modèles et de l'information recueillie sur le système, organisant l'analyse de Sûreté de Fonctionnement du système durant sa conception. MéDISIS peut être qualifiée de méthode déductive et incrémentale capitalisant les résultats d'analyses au sein d'une Base de données des Comportements Dysfonctionnels et exploitant les modèles SysML directement issus de la modélisation fonctionnelle. MéDISIS se déroule en trois étapes :

- Recherche, par une AMDEC munie d'une assistance automatisée, du comportement dysfonctionnel des composants et des exigences impactées.
- Construction assistée d'un modèle du comportement fonctionnel et dysfonctionnel du système, en utilisant une représentation formelle.
- Analyse outillée des modèles formels, validation des exigences, retour sur la conception.

Afin d'optimiser l'application de MéDISIS pour des projets de développement utilisant le langage SysML, nous avons analysé les mécanismes de réalisation de ces différentes phases. Nous avons successivement analysé comment un modèle SysML pouvait supporter l'analyse AMDEC d'un système, puis comment optimiser la création d'un modèle AltaRica Data Flow en exploitant le modèle central à l'Environnement de Développement Système et les connaissances révélées par l'AMDEC.

Nous avons, dans le chapitre II, étudié la réalisation d'AMDEC à partir de modèles SysML fonctionnels issus de l'analyse de conception préliminaire. Pour cela, nous avons dégagé le métamodèle d'organisation des informations relevées lors d'une AMDEC et l'avons mis en correspondance avec le métamodèle SysML. Ceci nous a permis de définir des règles d'analyse de modèles SysML pour l'obtention d'AMDEC composant. L'étude réalisée au niveau des métamodèles permet d'obtenir une systématique de traitement applicable à tout type de modèles SysML, quelles que soient les habitudes de modélisation du concepteur. Ces règles sont utilisées pour dégager les données nécessaires à l'établissement d'une AMDEC. La finalisation nécessite tout de même l'intervention d'un expert. Nous facilitons cette dernière en proposant une AMDEC préliminaire, automatiquement générée en appliquant les règles de traitement. Les techniques employables pour mettre en place un tel outil ont été discutées afin de ne privilégier aucune solution commerciale particulière.

Dans le troisième chapitre, nous avons développé une solution de traduction de modèle venant soutenir la seconde phase de MéDISIS. Nous nous sommes focalisés sur la création assistée de modèles d'analyse de Sûreté de Fonctionnement construits en AltaRica Data Flow. Ces descriptions, principalement dédiées aux analyses de Sûreté de Fonctionnement, sont largement utilisées dans l'industrie pour analyser quantitativement le comportement dysfonctionnel des systèmes et ainsi valider leurs performances. Cette traduction de SysML vers AltaRica Data Flow permet d'intégrer l'usage de ce formalisme au processus d'Ingénierie Système, en apportant une meilleure cohérence et traçabilité entre les modèles, ainsi qu'une plus grande facilité et rapidité de construction. Nous avons défini une méthode de traduction de modèles SysML en AltaRica Data Flow, permettant d'automatiser la création partielle des modèles de Sûreté de Fonctionnement. Cette création est associée à l'utilisation de la Base de données des Comportements Dysfonctionnels support de MéDISIS, permettant d'adjoindre la partie dysfonctionnelle du système, identifiée au cours de l'AMDEC. L'implémentation de la solution proposée a été discutée, illustrant ainsi les possibilités d'unification des outils dédiés à l'utilisation de SysML et d'AltaRica Data Flow.

La Base de données des Comportements Dysfonctionnels décrite au cœur de ces deux chapitres nous a permis de montrer comment modéliser les aspects d'un comportement dysfonctionnel en SysML. Cette étude est primordiale dans la volonté d'intégrer les notions de Sûreté de Fonctionnement au processus d'Ingénierie Système Basée sur les Modèles. Les constructions que nous avons proposées permettent d'obtenir des bases de connaissances cohérentes avec les modèles des concepteurs, constituant un format de données pérennes à travers les différents projets de développement que peut mener un industriel. Cette Base de données des Comportements Dysfonctionnels exploitant le langage SysML, prouve la capacité du langage à modéliser toutes les dimensions d'un système, le rendant ainsi pleinement exploitable pour décrire des modèles orchestrant toutes les activités relatives à l'Ingénierie Système Basée sur les Modèles. De plus, le métamodèle fourni pour cette Base de données des Comportements Dysfonctionnels peut être réalisé sur n'importe quel outil de modélisation déployant SysML.

Dans le dernier chapitre de cette thèse, nous avons illustré l'utilisation de MéDISIS au sein d'un projet de développement d'un système innovant : le projet CAPTHOM. Cette étude nous a permis de montrer une autre qualité de notre méthode, qu'est sa capacité à faire apparaître les besoins d'analyse du système ainsi qu'à guider la réalisation des outils d'analyse. Cet exemple est développé jusqu'à l'obtention du logiciel de simulation et d'optimisation de réseaux de capteurs, que nous avons baptisé SNOOPS. Ce logiciel permet de simuler le comportement des capteurs de technologies différentes testés et développés au cours de CAP-

THOM. SNOOPS simule les capteurs au cœur de leur mission, en considérant leur lieu d'implantation et les activités ayant lieu au sein de cet environnement. Enfin, le logiciel dispose également d'un service d'optimisation de placement de ces capteurs permettant de calculer les positions optimum des capteurs, pour l'instrumentation d'un environnement de test ou d'un bâtiment réel. SNOOPS, conçu dans une philosophie d'Ingénierie Système Basée sur les Modèles, s'intègre à l'Environnement de Développement Système CAPTHOM et fournit des bases de développement pour les travaux de recherches menés dans CAPTHOM et faisant l'objet de trois autres thèses. Ces travaux étudient la modélisation de la présence humaine, la fusion de données multi-capteurs et la détection d'humains par vision.

Les perspectives offertes à ce travail sont multiples. L'emploi de MéDISIS ne pourra être pleinement évalué que par la réalisation d'études de différents types de systèmes avec cette méthodologie. Il sera important de pouvoir tester les principes énoncés sur des projets à l'envergure plus importante que le projet CAPTHOM. De telles expérimentations devront être comparées aux techniques actuellement utilisées par les ingénieurs en charge des études de développement, afin de pouvoir mieux quantifier les bénéfices de MéDISIS. Une telle étude sera menée dans les trois prochaines années au sein de l'équipe MCDS (Modélisation Commande et Diagnostic des Systèmes) de l'Institut PRISME. En effet, les concepts de MéDISIS y seront testés pour la conception de l'électronique de bord d'une fusée stratosphérique conçue au sein du projet LEA (Lietaoutnii Experimentalnii Apparat) conduit par la société MBDA. Une telle étude sera également l'occasion de développer une première Base de données de Comportements Dysfonctionnels de taille importante et de tester sa réutilisation pour d'autres cas d'application.

Comme nous l'avons mentionné aux chapitres II et III, les services de création d'AMDEC et de traduction vers AltaRica Data Flow gagneront en efficacité lors de leur adaptation à des méthodologies de saisie de modèle particulières. La définition de ces méthodologies est nécessaire à l'utilisation de SysML dans un cadre industriel. A ce titre, des référentiels d'utilisation de SysML doivent être créés au sein des organisations développant leurs systèmes avec SysML. Nous avons, dans cette thèse, fourni des pistes et exemples de règles de modélisation pouvant inspirer de tels référentiels. Une fois ces derniers connus, il sera bénéfique d'adapter à ces méthodes les traitements proposés aux chapitres II et III, afin d'obtenir des générations d'AMDEC préliminaires et de modèles AltaRica Data Flow extrêmement pertinentes.

Notre expérience dans l'utilisation de SysML, nous a permis de constater que ce langage offre de nombreuses possibilités d'adaptation vers les différentes techniques employées au cours de l'Ingénierie Système Basée sur les Modèles. De nombreuses études restent à mener quant à l'optimisation du passage de descriptions SysML, à l'utilisation de modèles de conception détaillée. La génération de modèles de Sûreté de Fonctionnement doit également être affinée. Le niveau de détail utilisé dans un modèle de Sûreté de Fonctionnement dépend énormément du besoin d'analyse lié au système. A ce titre, nous pressentons que l'utilisation des vues dans les modèles SysML et les notions de généralisation utilisées dans leur construction, peuvent permettre d'atteindre plus facilement les degrés d'abstraction pertinents pour l'élaboration des modèles dédiés à l'étude de la Sûreté de Fonctionnement. Nous pensons également que l'intégration des COTS (Component On The Shelf) est un autre challenge inhérent au développement des systèmes complexes, qui pourra bénéficier de l'usage de SysML. Les constructions SysML devraient permettre une meilleure expression et gestion des exigences ainsi que des caractérisations de tels composants, facilitant leur choix et leur intégration au système global. L'aspect Sûreté de Fonctionnement de leur comportement devra

par ailleurs faire partie intégrante de leur spécification. L'utilisation des constructions que nous avons définies dans cette thèse au moyen de la Base de connaissances sur les Comportements Dysfonctionnels pourra s'avérer pertinente pour de telles tâches.

Annexe – Les classes de scénarios dans CAPTHOM

Dans cette annexe, nous donnons des exemples de scénario illustrant les missions des capthoms. Ces scénarios, réalisés par les partenaires du projet, ont été saisis en utilisant la grille proposée au paragraphe IV.2.2.1 et présentée sur la figure IV.1. Nous les avons organisés en quatre classes représentatives des grands types de mission affectés à un capteur de présence humaine destiné à l'habitat et aux bâtiments tertiaires.

Scénario d'occupation tertiaire

Type de pièce:

Bureau : armoire + bureau + accès + radiateurs + lumières (lampe de bureau, éclairage général)
Couloir : espace longitudinal avec de multiples accès + radiateurs + lumières (éclairages généraux)

Scénarios attendus :

Bureau : déplacement, travail assis devant le bureau, immobilité à divers endroits de la pièce.
Couloir : déplacements à grande vitesse, déplacement à vitesse normale, immobilité.

Similarité avec les scénarios suivant :

Occupation d'une cuisine (assis pour manger, debout pour cuisiner), occupation d'une salle à manger, occupation d'un salon (hors sommeil).

Critères de réussite du scénario :

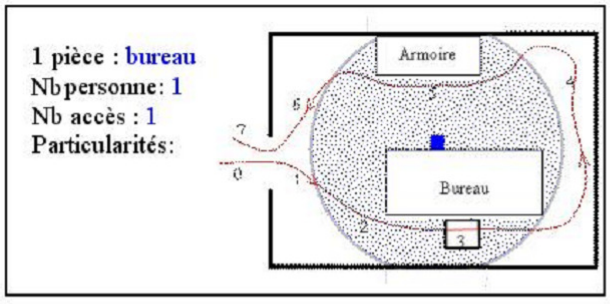
Capthom basic : passage à l'état détection lors de la présence d'au moins un humain dans la zone surveillée, passage à un état « négatif » lors de l'absence d'humain dans la zone surveillée.
Capthom évolué : même comportement de base que le Capthom basic, le système doit en outre, être capable de donner le nombre d'occupant, leur position, leur activité et leur posture.

Points clés et difficultés techniques :

Le système ne doit pas perdre la détection d'une personne immobile. Le Capthom évolué devra pouvoir différencier plusieurs personnes.

Exemple :

Scénario n°: 1 (bureau occupé)



Nomenclature

Température:
1: T < Tmin
2: Tmin < T < Tmax
3: T > Tmax

Posture:
P1: Debout
P2: Assis
P3: Allongé
P4: .....

Activité:
A1: Marche
A2: Immobile
A3: Dort
A4: Lecture
A5: Mange
A6: .....
A7: .....
A8: .....
A9: .....

Flux parasitaire:
F0 : pas de flux
F1: flash lumineux
F2 : flux d'air chaud
F3 : onde électromagnétique
F4 : .....
F5 : .....

Stimuli de fausse détection :
S0 : pas de Stimuli
S1 : proche ouverture
S2 : animal domestique
S3 : flash lumineux
S4 : jouet mobile
S5 : .....
S6 : .....
S7 : .....

N° segment	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Vitesse : 1 : v<v min 2 : v min <v<v max 3 : v >v max	2	2	2	1	2	1	2	2						
Posture	P1	P1	P1	P2	P1	P1	P1	P1						
Activité	A1	A1	A1	A4	A1	A2	A1	A1						
Flux parasitaire	F0	F0	F0	F0	F0	F0	F0	F0						
Température	2	2	2	2	2	2	2	2						
Stimuli de fausse détection	S0	S0	S0	S0	S0	S0	S0	S0						
Réponse souhaitée Capthom Basic	0	1	1	1	1	1	1	0						
Réponse souhaitée Capthom Etendu Nb.														
Post.														
Act.														

Scénario du sommeil

Type de pièce :

Pièce où il est possible de dormir (salon avec canapé, chambre).

Scénarios attendus :

Une personne entre dans la pièce et se couche pour dormir. Deux cas sont alors possibles, la personne dort sous des draps ou sans draps (ex : canapé).

Deux personnes sont dans la pièce, l'une d'elle dort, la seconde peut avoir tout type d'activité.

Critères de réussite du scénario :

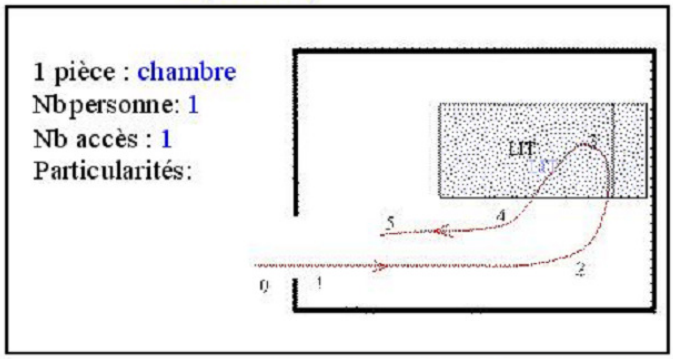
Capthom basic : passage à l'état détection lors de la présence d'au moins un humain dans la zone surveillée, passage à un état « négatif » lors de l'absence d'humain dans la zone surveillée.
Capthom évolué : même comportement de base que pour le capthom basic, le système doit en outre être capable de donner le nombre d'occupants, leur position, leur activité et leur posture. En particulier, il doit indiquer quand une personne dort, il doit également dissocier les activités et posture de plusieurs occupants (ex : l'un dort, l'autre lit).

Points clés et difficultés techniques :

Une des difficultés majeures est de pouvoir détecter une personne sous les draps (savoir qu'elle n'est pas repartie peut être suffisant). Il est primordial d'isoler le comportement de chaque personne.

Exemple :

Scénario n°: 5 (chambre)



Nomenclature

Température:
1: T < Tmin
2: Tmin < T < Tmax
3: T > Tmax

Posture:
P1: Debout
P2: Assis
P3: Allongé
P4: .....

Activité:
A1: Marche
A2: Immobile
A3: Dort
A4: Lecture
A5: Mange

Flux parasitaire:
F0 : pas de flux
F1: flash lumineux
F2 : flux d'air chaud
F3 : onde électromagnétique

Stimuli de fausse détection :
S0 : pas de Stimuli
S1 : proche ouverture
S2 : animal domestique
S3 : flash lumineux
S4 : jouet mobile
S5 :
S6 : .....
S7 : .....

N° segment	0	1	2	3	4	5								
Vitesse : 1 : v<v min 2 : v min <v<v max 3 : v >v max	2	2	2	1	2	2								
Posture	P1	P1	P1	P3	P1	P1								
Activité	A1	A1	A1	A3	A1	A1								
Flux parasitaire	F0	F0	F0	F0	F0	F0								
Température	2	2	2	2	2	2								
Stimuli de fausse détection	S0	S0	S0	S0	S0	S0								
Réponse souhaitée Capthom Basic	0	1	1	1	1	1								
Réponse souhaitée Capthom Etendu Nb.	0	1	1	1	1	1								
Post.		P1	P1	P3	P1	P1								
Act.		A1	A1	A3	A1	A1								

Scénario de l'activité inhabituelle

Type d'activité :

Seront testées, comme activités inhabituelles, les activités suivantes : les chutes, agitations (mouvements désordonnés).

Type de pièce :

Il n'y a pas de pré-requis sur la nature de la pièce dans laquelle se déroule l'activité. *(étude en zone libre de perturbations et d'occultations, i.e. capteur plein flux)*

Scénarios attendus :

Une personne entre dans la pièce et chute (chute brutale, chute lente).
Une personne entre dans la pièce et s'agite sur place (mouvements amples et rapides, ou mouvements lents).

Critères de réussite du scénario :

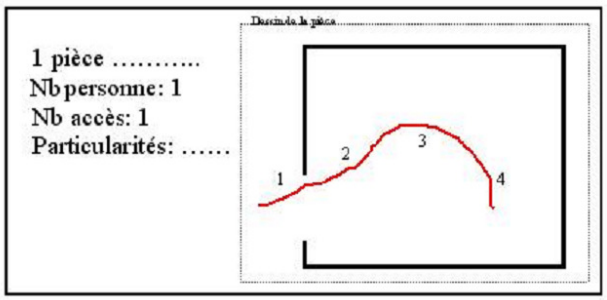
Capthom basic : passage à l'état détection lors de la présence d'au moins un humain dans la zone surveillée, passage à un état « négatif » lors de l'absence d'humain dans la zone surveillée.
Capthom évolué : même comportement de base que pour le capthom basic, le système doit en outre être capable de donner le nombre d'occupants, leur position, leur activité et leur posture. En particulier on doit indiquer quand la personne chute et reconnaître une activité d'agitation.

Points clés et difficultés techniques :

Détecter la chute de façon différente d'un changement de position. Idem pour l'agitation.

Exemple :

Scénario n° :



Nomenclature

Température:
1: T < Tmin
2: Tmin < T < Tmax
3: T > Tmax

Posture:
P1: Debout
P2: Assis
P3: Allongé
P4: .....

Activité:
A1: Marche
A2: Immobile
A3: Dort
A4: Lecture
A5: Mange

Activité (suite):
A6:chute
A7: .....
A8: .....
A9: .....

Flux parasitaire:
F0 : pas de flux
F1: flash lumineux
F2 : flux d'air chaud
F3 : onde électromagnétique
F4 : .....
F5 : .....

Stimuli de fausse détection :
S0 : pas de Stimuli
S1 : proche ouverture
S2 : animal domestique
S3 : flash lumineux

Stimuli (suite):
S4 : jouet mobile
S5 : .....
S6 : .....
S7 : .....

N° segment	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Vitesse : 1 : v<v min 2 : v min <v<v max 3 : v >v max		2	2	2	1									
Posture		P1	P1	P1	P3									
Activité		A1	A1	A2	A6									
Flux parasitaire		F0	F0	F0	F0									
Température		2	2	2	2									
Stimuli de fausse détection		S0	S0	S0	S0									
Réponse souhaitée Capthom Basic		1	1	1	1									
Réponse souhaitée Nb.		1	1	1	1									
Capthom Etendu Post.		P1	P1	P1	P3									
Act.		A1	A1	A1	A6									

Scénarios de tests des stimulus de fausses détections

Type de pièce :

Il n'y a pas de pré-requis sur la nature de la pièce dans laquelle se déroule l'activité. *(étude en zone libre de perturbations et d'occultations, i.e. capteur plein flux)*

Scénarios attendus :

Il s'agit de tester les principaux stimulus de fausses détections répertoriées.
Flux d'air chaud ou froid dans la zone sensible du capteur (ex : chauffage électrique soufflant, sèche cheveu, climatiseur, ventilateur).
Ondes électromagnétiques dans la zone sensible du capteur (ex : téléphone portable, four micro-onde, ampoule à décharges, télévision à tube cathodique).
Passage d'une personne dans l'entrebâillement d'une porte ou derrière une fenêtre.
Flash lumineux. Extinction de l'éclairage (soleil direct).
Changement lent de luminosité.
Passage d'un animal.
Objet mobile dans la pièce (ex : voiture télécommandée, imprimante, volet roulant, volet « qui claque »).
Surface chaude ou froide (ex : bouilloire, plaque de cuisson, récipient chaud, glaçons, objet froid)

Critères de réussite du scénario :

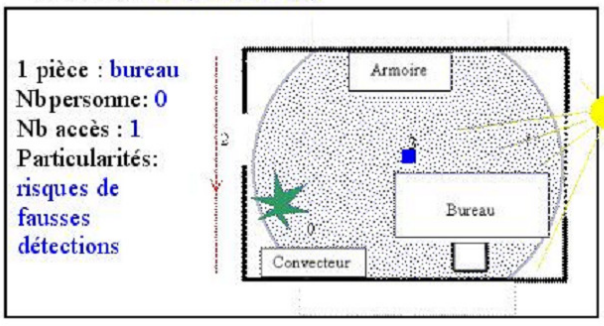
Que se soit pour le Capthom basic ou le Capthom évolué, il ne faut pas indiquer de détection dans chacun de ces cas.

Points clés et difficultés techniques :

Rester insensible à toutes sortes de stimulus. Deux voies sont possibles, l'insensibilité de la cellule sensible à ces stimulus ou l'ajout d'une correction lors du traitement.

Exemple :

Scénario n°: 3 (bureau vide)



Nomenclature

Température:
1: T < Tmin
2: Tmin < T < Tmax
3: T > Tmax

Posture:
P1: Debout
P2: Assis
P3: Allongé
P4: .....

Activité:
A1: Marche
A2: Immobile
A3: Dort
A4: Lecture
A5: Mange
A6: .....
A7: .....
A8: .....
A9: .....

Flux parasitaire:
F0 : pas de flux
F1: flash lumineux
F2 : flux d'air chaud
F3 : onde électromagnétique
F4 : flux solaire
F5 : .....

Stimuli de fausse détection :
S0 : pas de Stimuli
S1 : proche ouverture
S2 : animal domestique
S3 : flash lumineux
S4 : jouet mobile
S5 : Mouvement d'objet (plante, rideau)

N° segment	0	1	2	3										
Vitesse : 1 : v<v min 2 : v min <v<v max 3 : v >v max			2											
Posture			P1											
Activité			A1											
Flux parasitaire	F2	F4	F0	F2										
Température	2	2	2	2										
Stimuli de fausse détection	S5	S0	S0	S0										
Réponse souhaitée Capthom Basic	0	0	0	0										
Réponse souhaitée Capthom Etendu Nb.														
Post.														
Act.														

Liste des publications de l'auteur

Conférences Internationales

[1] V. Idasiak & P. David. *Accident simulation: Design and results*. Proceedings of ESREL 2007 Risk, Reliability and Societal Safety – Aven & Vinnem (eds) © 2007 Taylor & Francis Group, London, ISBN 978-0-415-44786-7 VOL II pp. 953-960, Stavanger, Norvège, 25-27 juin 2007.

[2] P. David, V. Idasiak & F. Kratz. *A Sensor Placement Approach for the Monitoring of Indoor Scenes*. Proceedings of EUROSSC 2007, In Lecture Notes in Computer Science, Springer Berlin / Heidelberg, ISBN 978-3-540-75695-8, Vol. 4793, pp. 110-125, Kendal, 23-25 octobre, 2007.

[3] P. David, V. Idasiak & F. Kratz. *Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models*. Proceedings of ESREL 2008 and 17th SRA-EUROPE annual conference, Martorell, Guedes Soares & Barnett (eds) © 2009 Taylor & Francis Group, London, ISBN 13 978-0-415-48513-5, pp. 2259-2266, Valencia, Espagne, 22-25 septembre 2008.

[4] P. David, V. Idasiak & F. Kratz. *Use and improvements of SysML in reliability study*. Proceedings of the 55th Annual Reliability and Maintainability Symposium, RAMS 2009, Fort Worth, Texas, USA, 26-29 janvier 2009.

[5] P. David, V. Idasiak & F. Kratz. *Automating the synthesis of AltaRica Data-Flow models from SysML*. Proceedings of ESREL 2009, Prague, République Tchèque, 7-10 septembre 2009.

Conférence Nationale

[1] P. David, V. Idasiak & F. Kratz. *Etude pour une meilleure intégration des données de conception dans les analyses de fiabilité*. 16ème Congrès Lambda Mu : les nouveaux défis de la maîtrise des risques, Avignon, France, 7 au 9 octobre 2008.

[2] P. David, V. Idasiak & F. Kratz. *Placement de réseau de capteurs pour l'observation de scènes intérieures*. Sixième Colloque capteurs, Capteurs 2008, Bourges, France, 5-6 mars 2008.

Article Soumis

[1] P. David, V. Idasiak & F. Kratz. *Reliability study of complex physical systems using SysML*. International Journal in Reliability Engineering and System Safety.

Autres Communications

[1] A. Belconde, Y. Benezeth, D. Brulin, P. David & E.A. Fall. *La recherche dans CAPTHOM* .Poster au Sixième Colloque capteurs, Capteurs 2008, Bourges, France, 5-6 mars 2008.

[2] A. Belconde, Y. Benezeth, D. Brulin, P. David & E.A. Fall. *CAPTHOM : Vers une évolution des détecteurs de présence humaine*. Poster au Sixième Colloque capteurs, Capteurs 2008, Bourges, France, 5-6 mars 2008.

Références

[Arnold et al. 2000] : A. Arnold, A. Griffault, G. Point & A. Rauzy. *The AltaRica language and its semantics*. Fundamenta Informaticae, Vol. 34, pp. 109-124, 2000.

[Arnold et al. 2005] : A. Arnold, A. Griffault, G. Point & A. Rauzy. *AltaRica Manuel méthodologique*. Rapport LaBRI, Université Bordeaux I et CNRS, 26 septembre 2005.

[Atkinson et al. 1992] : R. Atkinson, M. Montakhab, K. Pillay, D. Woollons, P. Hogan, C. Burrows & K. Edge. *Automated Fault Analysis for Hydraulic Systems: Part1 – Fundamentals*. Journal of Systems and Control Engineering, Vol. 206, pp. 207-214, 1992.

[Bares et al. 2000] : W. Bares, S. Thainimit & S. McDermott. *A model for constraint-based camera planning*. Proceedings of the 2000 AAAI Spring Symposium, Stanford, pp. 84-91, 20-22 mars 2000.

[Bassetto 2005] : S. Bassetto. *Contribution à la qualification et amélioration des moyens de production ~ Application à une usine de recherche et production de semi conducteurs*. Thèse de doctorat de l'Ecole Nationale Supérieure d'Arts et Métiers, soutenue le 28 juin 2005.

[Belconde & Kratz 2008] : A. Belconde & F. Kratz. *Problèmes liés à la détection de présence humaine et modélisation*. 6ème Colloque Capteurs, Bourges, 5-6 mars 2008.

[Benezeth et al. 2008b] : Y. Benezeth, B. Emile, H. Laurent & C. Rosenberger. *A real time human detection system based on far infrared vision*. Proceedings of International Conference on Image and Signal Processing, Cherbourg, 1-3 juillet 2008.

[Benezeth et al. 2009] : Y. Benezeth, B. Emile, H. Laurent & C. Rosenberger. *Détection de la présence humaine et caractérisation de l'activité*. XXIIème Colloque GRETSI, Dijon, 8-11 septembre 2009.

[Bernardi et al. 2002] : S. Bernardi, S. Donatelli & J. Merseguer. *From UML Sequence Diagrams and StateCharts to analyzable Petri Net models*. In Proceedings of the 3rd Int. workshop on software on performance, Rome, Italy, 2002.

[Bézivin et al.2005] : J. Bézivin, F. Jouault & D. Touzet. *An Introduction to ATLAS Model Management Architecture*. ATLAS GDD, Rapport de Recherche N°05-01, LINA, février 2005.

[Bieber et al. 2004] : P ; Bieber, C. Bougnol, C. Castel, J.-P. Hechmann, C. Kehren, S. Metge & C. Seguin. *Safety Assessment with AltaRica*. IFIP International Federation for Information Processing, Vol. 156, pp. 505-510, 2004.

[Boiteau et al. 2006] : M. Boiteau, Y. Dutuit, A. Rauzy & J.-P. Signoret. *The AltaRica data-flow language in use : modeling of production availability of a multi-state system*. Reliability Engineering & System Safety, Vol. 91, pp. 747 755, 2006.

[Bondavalli et al. 1999a] : A. Bondavalli, M. Dal Cin, D. Latella & A. Pataricza. *High-level Integrated Design Environment for dependability (HIDE)*. Proceedings of the 5th International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS-99), pp.87-92, 1999.

[Bondavalli et al. 1999b] : A. Bondavalli, I. Majzik & I. Mura. *Automated Dependability Analysis of UML Designs*. Proceedings of the 2nd IEEE International Symposium on Object-oriented Real-time distributed Computing, Saint Malo, France, 1999.

[Bondavalli et al. 2001] : A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza & G. Savoia. *Dependability analysis in the early phases of UML-based system design*. International Journal of Computer Systems Science & Engineering, Vol. 16 n°5, pp. 265-275, septembre 2001.

[Bonhomme 2008] : S. Bonhomme. *Méthodologie et Outils pour la Conception d'un Habitat Intelligent*. Thèse de doctorat de l'Institut National Polytechnique de Toulouse, soutenue le 15 mai 2008.

[Bouissou et al. 1991] : M. Bouissou, H. Bouhadana, M. Bannelier & N. Villatte. *Knowledge modelling and reliability processing: presentation of the FIGARO language and associated tools*. Proceedings of the Safecomp'91, Trondheim, novembre 1991.

[Bouissou & Bon 2003] : M. Bouissou & J.-L. Bon. *A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes*. Reliability Engineering and System Safety, Vol. 82, pp. 149-163, 2003.

[Bouissou & Seguin 2006] : M. Bouissou & C. Seguin. *Comparaison des langages de modélisation AltaRica et FIGARO*. 15ème Congrès Lambda Mu : Risques et Performances, Lille, 10-12 octobre 2006.

[Bouti et al. 1994] : A. Bouti, D. Ait Kadi & K. Dhouib. *Automated manufacturing systems failure analysis based on a functional reasoning*. In Proceedings of the 10th ISPE-IFAC International Symposium on CAD/CAM, Robotics and factories of the Ftuture, pp. 423-429, 1994.

[Bowles & Pelaez 1995] : J.B. Bowles & C.E. Pelaez. *Fuzzy logic priorization of failures in a system failure mode, effects and criticality analysis*. Reliability Engineering and System Safety, Vol. 50, pp. 203-213, 1995.

[Brulin et al. 2009] : D. Brulin, E. Courtial & G. Allibert. *Visual Receding Horizon Estimation for human presence detection*. ICRA 2009, Workshop People Detection and Tracking of the IEEE International Conference on Robotics and Automation, Kobe, 12-17 mai 2009.

[Cantor 2003] : M. Cantor. *Rational Unified Process® for Systems Engineering, RUP SE® Version 2.0*. IBM Rational Software white paper, IBM Corporation, 8 mai 2003.

[Chakrabarty et al. 2002] : K. Chakrabarty, H. Qi & E. Cho. *Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks*. IEEE transactions on computers, Vol. 51, pp. 1448-1453, 2002.

[Chenouard et al. 2008] : R. Chenouard, L. Granvilliers & R. Soto. *Model-Driven Constraint Programming*. Proceedings of 10th International Symposium on Principles and Practice of Declarative Programming, Valence (Espagne), 15-17 Juillet, 2008.

[Clavareau & Labeau 2009] : J. Clavareau & P.-E. Labeau. *A Petri net-based modelling of replacement strategies under technological obsolescence*. Reliability Engineering and System Safety, Vol. 94, pp.357-369, 2009.

[Cressent 2009] : R. Cressent. *Comparaison et mise en placed'une méthode d'analyse et de modélisation de systèmes complexes, temps réel et embarqués*. Rapport de stage Ingénieur de l'ENSIB, soutenu à Bourges le 1er septembre 2009.

[David et al. 2007] : P. David, V. Idasiak & F. Kratz. *A Sensor Placement Approach for the Monitoring of Indoor Scenes*. Proceedings of EUROSSC 2007, In Lecture Notes in Computer Science, Springer Berlin / Heidelberg, ISBN 978-3-540-75695-8, Vol. 4793, pp. 110-125, Kendal, 23-25 octobre, 2007.

[David et al. 2008a] : P. David, V. Idasiak & F.Kratz. *Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models*. Proceedings of ESREL 2008 and 17th SRA-EUROPE annual conference, Martorell, Guedes Soares & Barnett (eds) © 2009 Taylor & Francis Group, London, ISBN 13 978-0-415-48513-5, pp 2259-2266, Valence, Espagne, 22-25 septembre 2008.

[David et al. 2008b] : P. David, V. Idasiak & F.Kratz. *Etude pour une meilleure intégration des données de conception dans les analyses de fiabilité*. Actes du 16ème Congrès Lambda Mu : les nouveaux défis de la maîtrise des risques, Avignon, France, 7-9 octobre 2008.

[David et al. 2009a] : P. David, V. Idasiak & F.Kratz. *Use and improvements of SysML in reliability study*. Proceedings of the 55th Annual Reliability and Maintainability Symposium, RAMS 2009, Fort Worth, Texas, USA, Jan. 2009.

[David et al. 2009b] : P. David, V. Idasiak & F.Kratz. *Automating the synthesis of AltaRica Data-Flow models from SysML*. Proceedings of ESREL 2009, Prague, République Tchèque, 7-10 septembre 2009.

[Dhillon & Chakrabarty 2003] : S. Dhillon & K. Chakrabarty. *Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks*. Wireless Communication and Networking, Vol. 3, pp. 1609-1614, mai 2003.

[Douglass 2005] : B. Douglass. *The Harmony Process*. I-Logix white paper, I-Logix, Inc., 25 mars 2005.

[Dumas et al. 2008] : X. Dumas, C. Pagetti, L. Sagaspe, P. Bieber & P. Dhaussy. *Vers le génération de modèles de sûreté de fonctionnement*. 15ème Conférence francophone sur les Langages et Modèles à Objets (LMO) & Architectures Logicielles (CAL), Montréal, 5-7 mars 2008.

[Dunn et al. 2006] : E. Dunn, G. Olague & E. Lutton. *Parisian Camera Placement for Vision Metrology*. Pattern Recognition Letters 27, pp. 1209-1219, 2006.

[Dutuit et al. 1997] : Y. Dutuit, E. Châtelet, J.-P. Signoret & P. Thomas. *Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases*. Reliability Engineering and System Safety, Vol. 55, pp. 117-124, 1997.

[EIA 632] : Electronic Industries Alliance. *Processes for Engineering a System*. 1999.

[EN 50126] : European Committee for Electrotechnical Standardization. *Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Saftey (RAMS)*. 1er septembre 1999.

[Erdem & Sclaroff 2006] : U. Erdem & S. Sclaroff. *Automated Camera Layout to Satisfy Task-Specific and Floor Plan-Specific Coverage Requirements*. Computer Vision and Image Understanding, Vol. 103, pp.156-169, 2006.

[Escriba 2005] : C. Escriba. *Conception, Réalisation et Caractérisation de capteurs infrarouges à thermopiles, Application à la détection de présence passive dans l'habitat*. Thèse de doctorat de l'Université Paul Sabatier Toulouse III, soutenue le 5 décembre 2005.

[Estefan 2008] : J. Estefan. *Survey of Model-Based Systems Engineering (MBSE) Methodologies, Rev. B*. INCOSE MBSE Initiative, 23 Mai 2008.

[Forsberg et al. 1995] : K. Forsberg et H. Mooz. *Application of the Vee to Incremental and Evolutionary Development*. In Proceedings of the 5th Annual International Symposium of the National Council on Systems Engineering, St. Louis, Juillet 1995.

[Friedenthal 1998] : S. Friedenthal. *Object Oriented Systems Engineering*. In Process Integration for 2000 and Beyond: Systems Engineering and Software Symposium. New Orleans : Lockheed Martin Corporation, Mai 1998.

[Friedenthal et al. 2007] : S. Friedenthal, R. Griego & M. Sampson. *INCOSE MBSE Initiative*. International Council on Systems Engineering 2007, San Diego, 24-29 juin 2007.

[Friedenthal et al. 2008] : S. Friedenthal, A. Moore & R. Steiner. *A Practical Guide to SysML : The Systems Modeling Language*. The MK/OMG press, Elsevier, 2008.

[Gilmore & Kloul 2005] : S. Gilmore & L. Kloul. *A unified tool for performance modelling and prediction*. Reliability engineering and system safety. Vol. 89, pp. 17-32, 2005.

[Gobeau 2006] : J-F. Gobeau. *Détecteurs de mouvement à infrarouge passif*. 5ème rencontres capteurs Capteurs, Bourges, 18-19 octobre 2006.

[Godts et al. 2008] : P. Godts, M. Haffar, K. Ziouche, Z. Bougrioua & D. Leclerq. *Nouveaux microcapteurs de flux thermique et de rayonnement infrarouge planaires à faible coût en technologie silicium pour applications domotiques*. 6ème Colloque Capteurs, Bourges, 18-19 mars 2008.

[Gonzalez-Barbosa et al. 2009] : J-J. Gonzalez-Barbosa, T. Garcia-Ramirez, J. Salas, J-B. Hurtado-Ramos & J-d-J. Rico-Jimenez. *Optimal Camera Placement for Total Coverage*. Proceedings of IEEE International Conference on Robotics and Automation, pp. 844-848, Kobe, 12-17 mai 2009.

[Guiochet 2003] : J. Guiochet. *Maîtrise de la sécurité des systèmes de la robotique de service, approche UML basée sur une analyse du risque système*. Thèse de doctorat de l'Institut National des Sciences Appliquées de Toulouse, soutenue le 9 juillet 2003.

[Haffar 2007] : M. Haffar. *Etude et réallisation de matrices de microcapteurs infrarouges en technologie silicium pour imagerie basse résolution*. Thèse de doctorat de l'Université des Sciences et Technologies de Lille, soutenue le 29 novembre 2007.

[Harel 1987] : D. Harel. *Statecharts: a Visual Formalism for Complex Systems*. Journal of Science of Computer Programming, Vol. 8, pp. 231-274, 1987.

[Hause 2006] : M. Hause. *The SysML Modelling Language*. Proceedings of the 5th European Systems Engineering Conference, Edimbourg, 18-20 septembre 2006.

[Hause & Thom 2008] : M. Hause & F. Thom. *Building Bridges Between Systems and Software With SysML and UML*. Proceedings of International Symposium of the INCOSE, Utrecht, Pays Bas, 15-19 juin, 2008.

[Hecht et al. 2004] : H. Hecht, X. An & M. Hecht. *Computer-aided Software FMEA*. The Annual Reliability and Maintainability Symposium, Los Angeles, 26-29 janvier, 2004.

[Hoffmann 2006] : H-P Hoffmann. *Harmony-SE/SysML Deskbook: Model-Based Systems Engineering with Rhapsody*. Telelogic/I-Logix white paper, Telelogic AB, 24 mai 2006.

[Hughes et al. 1999] : N. Hughes, E. Chou, C. Price & M. Lee. *Automating Mechanical FMEA Using Functional Models*. Proceedings of the 12th International FLAIRS, 1999.

[Idasiak & David 2007] : V. Idasiak & P. David. *Accident simulation: Design and results*. Proceedings of ESREL 2007 Risk, Reliability and Societal Safety – Aven & Vinnem (eds) © 2007 Taylor & Francis Group, London, ISBN 978-0-415-44786-7 VOL II, pp 953-960, Stavanger, 25-27 juin 2007.

[IEC 15288] : International Electrotechnical Commission. *Systems Engineering – System Life Cycles Processes*. 15 novembre 2005.

[IEC 60812] : International Electrotechnical Commission. *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*. International Standard, Second Edition, janvier 2006.

[IEC 61508] : International Electrotechnical Commission. *Functional Safety of Electrical /Electronic /Programmable Electronic Safety-Related Systems*. Parts 1 to 7, 1998.

[IEC 61513] : International Electrotechnical Commission. *Nuclear power plants – Instrumentation and control for systems important to safety – General requirements for systems*. 2002.

[IEC 61511] : International Electrotechnical Commission. *Functional safety - Safety instrumented systems for the process industry sector*. 2004.

[IEC 62061] : International Electrotechnical Commission. *Safety of machinery Functional safety of safety-related electrical, electronic and programmable electronic control systems*. 2005.

[IEEE 1220] : Institute for Electrical and Electronic Engineers. *IEEE standard for Application and Management of the Systems Engineering Process*. 8 décembre 1998.

[INCOSE 2004] : International Council on Systems Engineering. *Systems Engineering Handbook*. Version 3, 2004.

[INCOSE 2007] : International Council on Systems Engineering. *Systems Engineering Vision 2020*. Version 2.03, TP-2004-004-02, Septembre 2007.

[Iwu et al. 2007] F. Iwu, A. Galloway, J. McDermid & I. Toyn. *Integrating safety and formal analyses using UML and PFS*. Reliability engineering and system safety. Vol. 92, pp.156-170, 2007.

[Johnson et al. 2008] : T. Johnson, C. Paredis & R. Burkhart. *Integrating Models and Simulations of Continuous Dynamics into SysML*. Proceedings of the 6th International Modelica Conference, Bielefeld, 2008.

[Jouault & Kurtev 2005] : F. Jouault & I. Kurtev. *Transforming Models with ATL*. Proceedings of 8th International Conference on Model Driven Engineering Languages and Systems (MODELS Satellite Events 2005), LNCS, Vol. 3844, pp. 128-138, 3 octobre 2005.

[Keller & Cima 1983] : H. Keller & D. Cima. *Achieving high reliability in passive infrared intruder alarms with lithium tantalate pyroelectric detectors*. SPIE International Conference/Europe, Genève, 18-22 avril 1983.

[Keller 2000] : H. Keller. *30 Years of Passive Infrared Motion Detectors – a Technology Review*. OPTO/IRS^2, Erfurt, 11 mai 2000.

[King & Pooley 1999] : P. King & R. Pooley. *Using UML to Derive Stochastic Petri Net Models*. Proceedings of 15th Annual UK Performance Engineering Workshop, 1999.

[Kleyner & Volovoi 2008] : A. Kleyner & V. Volovoi. *Reliability prediction using Petri nets for on-demand safety systems with fault detection*. Proceedings of ESREL 2008 and 17th SRA-EUROPE annual conference, Martorell, Guedes Soares & Barnett (eds) © 2009 Taylor & Francis Group, London, ISBN 13 978-0-415-48513-5, Valence, Espagne, 22-25 septembre 2008.

[Laprie 1995] : J.-C. Laprie, J. Arlat, J.-P. Blanquart, A. Costes, Y. Crouzet, Y. Deswarte, J.-C. Fabre, H. Guillermain, M. Kaâniche, K. Kanoun, C. Mazet, D. Powell, C. Rabéjac & P. Thévenod. *Guide de la sûreté defonctionnement*. Toulouse, France : Cépaduès –Éditions, 1995.

[Larish et al. 2008] : M. Larish, A. Hänle, U. Siebold & I. Häring. *SysML aided functional safety assessment*. In: Proceedings of ESREL 2008 and 17th SRA-Europe Annual Conference, Valencia, Spain, Sept 2008.

[Latella et al .1999] : D. Latella, I. Majzik & M. Massink. *Automatic verification of a behavioral subset of UML statechart diagrams unsing the SPIN model-checker*. Journal of Formal Aspects Computing, Vol. 11(6), pp. 637-664, 1999.

[Leangsukun et al. 2003] : C. Leangsukun, H. Song & L. Shen. *Reliability modeling using UML*. In Proceedings of the International conference on software engineering research and practice, Las Vegas, USA, Juin, 2003.

[Lienhart & Horster 2006] : R. Lienhrat & E. Horster. *On the optimal placement of multiple visual sensor*. Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks, pp. 111-120, octobre 2006.

[López-Grao et al. 2004] J-P. López-Grao, J. Merseguer & J. Campos. *From UML Activity Diagrams to Stochastic Petri Nets: Application to software performance engineering*. Proceedings of the 4th Int. workshop on software and performance, Redwood shores, USA, 2004.

[Lykins et al. 2000] : H. Lykins, S. Friedenthal, A. Meilich. *Adapting UML for an Object-Oriented systems engineering method (OOSEM)*. In: Proceedings of the 10th annual INCOSE symposium, Juillet 2000.

[Martin 1996] : J. Martin. *Systems Engineering Guidebook: A Process for Developing Systems and Products*. CRC Press, Inc.: Boca Raton, Florida, 1996.

[Meilich et al.1999] : A. Meilich et M. Rickels. *An Application of Object-Oriented Systems Engineering to an Army Command and Control System : A New Approach to Integration of Systems and Software Requirements and Design*. In Proceedings of the INCOSE international Symposium, Brighton, England, 6-11 Juin 1999.

[Merle et al. 2009] : G. Merle, J.-M. Roussel, J.-J. Lesage & A. Bobbio. *Algebraic Expression of the Structure Function of a subclass of Dynamic Fault Trees*. Proceedings of 2nd IFAC Workshop on Dependable Control of Discrete Systems (DCDS'09), Bari, 2009.

[Merseguer et al. 2002] : J. Merseguer, J. Campos, S. Bernardi & S. Donatelli. *A Compositional Semantics for UML State Machines Aimed at Performance Evaluation*. In Proceedings of the 6th Int. Workshop on Discrete Event Systems (WODES'02), 2002.

[MIL HDBK 217f] : USA Department of Defense. *Reliability Prediction of Electronic Equipment*. Military Standard. 2 décembre 1991.

[MIL-STD1629A] : USA Department of Defense. *Procedures for Performing a Failure Mode, Effects and Criticality Analysis*. Military Standard. 20 novembre 1980.

[Mkhida 2008] : A. Mkhida. *Contribution à l'évaluation de la sûreté de fonctionnement des Systèmes Instrumentés de Sécurité intégrant de l'Intelligence*. Thèse de doctorat de l'Institut National Polytechnique de Lorraine, soutenue le 14 novembre 2008.

[Modelica 2007] : The Modelica Association. *Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling*. Langage Specification, version 3.0, 5 septembre 2007.

[Mortureux 2002] : Y. Mortureux. *Analyse Préliminaire des Risques*. Techniques de l'Ingénieur, SE 4010, 10 octobre 2002.

[Muller et al. 2005] : P.-A. Muller, F. Fleurey & J.-M. Jézéquel. *Weaving executability into object-oriented meta-languages*. In Proceedings of MODELS/UML 2005, 2005.

[Murata 1989] : T. Murata. *Petri nets: properties, analysis and applications*. Proceedings of IEEE, Vol. 77, N° 4, pp. 541-580, 1989.

[NASA 2007] : NASA Procedural Requirement 7123.1A. *NASA Systems Engineering Processes and Requirements*. National Aeronautics and Space Administration, Washington, D.C., 6 mars 2007.

[Olague 2002] : G. Olague. *Automated photogrammetric network design using genetic algorithms*. Photogrammetric Engineering and Remote Sensing, Vol. 68, N° 5, pp. 423-431, 2002.

[OMG 2003] : Object Management Group. *UML for Systems Engineering RFP*. OMG document number ad/2003-03-41, 28 mars 2003.

[OMG 2006] : Object Management Group. *Object Constraint Language*. OMG Available Specification, Version 2.0, 1er mai 2006.

[OMG 2007] : Object Management Group. *OMG Systems Modeling Language (OMG SysML) V1.0*. 1er septembre 2007.

[OMG 2008] : Object Management Group. *OMG Systems Modeling Language (OMG SysML) V1.1*. 1er novembre 2008.

[OMG 2009] : Object Management Group. *Unified Modeling Language*. OMG Specification – UML 2.2 Superstructure & UML 2.2 Infrastructure, 2 février 2009.

[Pai & Dugan 2002] : G. Pai & J. Dugan. *Automatic Synthesis of Dynamic Fault Trees from UML System Models*. Proceedings of the 13th International Symposium on Software Reliability Engineering, Annapolis, USA, pp. 243-254, 2002.

[Pap et al. 2005] : Z. Pap, I. Majzik, A. Pataricza & A. Szegi. *Methods of checking general safety criteria in UML StateChart specifications*. Reliability engineering and system safety. Vol. 87, pp. 89-107, 2005.

[Papadopoulos et al. 2004] : Y. Papadopoulos, D. Parker & C. Grante. *A Method and Tool Support for Model-Based Semi-automated Failure Modes and Effects Analysis of Engineering Designs*. The 9th Australian Workshop on Safety Related Programmable Systems (SCS'04), Vol. 47, 2004.

[Peak 2000] : R. Peak. *X-Analysis Integration (XAI) Technology*. EIS lab Report EL002-2000A, Georgie Tech, Atlanta, 2000.

[Peak et al. 2007a] : R. Peak, R. Burkhart, S. Friedenthal, M. Wilson, M. Bajaj & I. Kim. *Simulation-based design using SysML Part 1: A parametrics primer*. In: Proceedings of International Symposium of the INCOSE., San Diego, USA, mai, 2007.

[Peak et al. 2007b] : R. Peak, R. Burkhart, S. Friedenthal, M. Wilson, M. Bajaj & I. Kim. *Simulation-based design using SysML Part 2: Celebrating diversity by example*. In: Proceedings of International Symposium of the INCOSE., San Diego, USA, mai, 2007.

[Philips et al. 1992] : C. Philips, N. Badler & J. Granieri. *Automatic viewing control for 3D direct manipulation*. Proceedings of the 1992 Symposium on Interactive 3D Graphics, pp.71-74, 1992.

[Picardi et al. 2004a] : C. Picardi, L. Console, F. Berger, J. Breeman, T. Kanakis, J. Moelands, S. Collas, E. Arbaretier, N. De Domenico, E. Girardelli, O. Dressler, P. Struss & B. Zilbermann. *AUTAS: a tool for supporting FMECA generation in aeronautic systems*. 16th European Conference on Artificial Intelligence, ECAI 2004, Valence, Espagne, 22-27 août, 2004.

[Picardi et al. 2004b] : C. Picardi, L. Console & D. Theseider Dupré. *Model Synthesis for Model-Based Fault Analysis*. Proceedings of 15th International Workshop on the Principles of Diagnosis, DX 04, Carcassonne, France, 23-28 juin 2004.

[Pillay & Wang 2004] : A. Pillay & J. Wang. *Modified failure mode and effects analysis using approximate reasoning*. Reliability Engineering & System Safety, vol. 79, pp. 69-85, 2003.

[Pop et al. 2007] : A. Pop, D. Akhvlediani &P. Fritzson. *Towards Unified Systems Modeling with the Modelica ML UML Profile*. International Workshop on Equation-Based Object-Oriented Languages and Tools, Linköping University Electronic Press, Berlin 2007.

[Price et al. 1995] : C. Price, D. Pugh, M. Wilson & N. Snooke. *The Flame system: Automating Electrical Failure Modes & Effect Analysis (FMEA).* The Annual Reliability and Maintainability Symposium, janvier, 1995.

[Price 1996] : C. Price. *Effortless Incremental Design FMEA*. The Annual Reliability and Maintainability Symposium, Las Vegas, janvier, 1996.

[Price 2000] : C. Price. *AutoSteve: Automated Electrical Design Analysis*. Proceedings of ECAI conference 14, pp. 721-725, 2000.

[Price & Taylor 2002] : C. Price, N. Taylor. *Automated multiple failure FMEA*. Reliability Engineering and System Safety Vol. 76, pp. 1-10, 2002.

[Rauzy 2002] : A. Rauzy. *Mode automata and their compilation into fault trees*. Reliability Engineering & System Safety, Vol. 78, pp. 1-12, 2002.

[Rauzy et al. 2008] : A. Rauzy, Z. Brik & E. Arbaretier. *Sûreté de Fonctionnement et Analyse de Performance*. Actes du 16ème Congrès Lambda Mu : les nouveaux défis de la maîtrise des risques, Avignon, France, 7-9 octobre 2008.

[Ridoux 1999] : M. Ridoux. *AMDEC- Moyen*. Techniques de l'Ingénieur, AG 4220, 10 juillet 1999.

[Robert et al. 1993] : M. Robert, M. Marchandiaux & M. Porte. *Capteurs intelligents et méthodologie d'évaluation*. Ouvrage aux éditions HERMES, ISBN 2-86601-382-4, Paris, mars 1993.

[Roedler 2002] : G. Roedler. *What is ISO/IEC 15288 and Why Should I Care ?* . ISO/IEC JTC1.SC7/WG7, Genève : International Organization for Standardization, 23 Septembre 2002.

[Saadat et al. 2004] : S. Saadat, F. Samdzadegan, A. Azizi, & M. Hahn. *Camera Placement for Network Design In Vision Metrology based On Fuzzy Inference System*. XXth ISPRS Congress, Istanbul, 12-23 juillet 2004.

[Sachveda et al. 2007] : A. Sachveda, D. Kumar & P. Kumar. *Reliability modelling of a industrial system with Petri nets*.Proceedings of ESREL 2007 Risk, Reliability and Societal Safety – Aven & Vinnem (eds) © 2007 Taylor & Francis Group, London, ISBN 978-0-415-44786-7, Stavanger, 25-27 juin 2007.

[Sadou 2007] : N. Sadou. *Aide à la conception des systèmes embarqués sûrs de fonctionnement*. Thèse de Doctorat Soutenue le 06 novembre 2007 à l'Université Toulouse III - Paul Sabatier, 2007.

[SAE 2009] : Society of Automotive Engineers. *SAE Architecture Analysis &Design Language*. Specification V2, janvier 2009.

[Schoenig 2004] : R. Schoenig. *Définition d'une Méthodologie de Conception des Systèmes Mécatroniques Sûrs de Fonctionnement*. Thèse de Doctorat Soutenue le 26 octobre 2004 à l'Institut National Polytechnique de Lorraine, 2004.

[Signoret et al. 2007] : J.-P. Signoret, Y. Dutuit & A. Rauzy. *High Integrity Protection Systems (HIPS): Methods and tools for efficient Safety Integrity Levels (SIL) analysis and calculations*. Proceedings of ESREL 2007 Risk, Reliability and Societal Safety – Aven & Vinnem (eds) © 2007 Taylor & Francis Group, London, ISBN 978-0-415-44786-7, Stavanger, 25-27 juin 2007.

[State et al. 2006] : A. State, G. Welch & A. Ilie. *An Interactive Camera Placement and Visibility Simulator for Image-Based VR Applications Stereoscopic Displays and Virtual Reality Systems*. Proceedings of the SPIE, Vol. 6055, pp.640-651, 2006.

[Teoh & Case 2004a] : P. Teoh & K. Case. *Failure modes and effects analysis through knowledge modelling*. Journal of Materials Processing Technology 153-154, pp. 253-260, 2004.

[Teoh & Case 2004b] : P. Teoh & K. Case. *Modelling and reasoning for failure modes and effect*. Proceedings of the Institute of Mechanical Engineers, Part B. Journal of engineering manufacturing. Vol. 218 n°3, pp. 289-300, 2004.

[Tumer et al. 2003] : I.Y. Tumer R.B. Stone & D.G. Bell. *Requirements for a Failure Mode Taxonomy for Use in Conceptual Design*. ICED 03 Stockholm, 19-21 août 2003.

[Turki et al. 2005] : S. Turki, T. Soriano & A. Sghaier. *A SysML profile for mechatronics integrating Bond Graphs*. Proceedings of the 9th WSEAS, International Conference on Circuits, Communications, Computers, Athènes, 11-16 juillet 2005.

[Turki 2008] : S. Turki. *Ingénierie système guidée par les modèles : Application du standard IEEE 15288, de l'architecture MDA et du langage SysML à la conception des systèmes mécatroniques*. Thèse de doctorat de l'Université du Sud Toulon-Var, soutenue le 2 octobre 2008.

[Vanderperren & Dehaene 2006] : Y. Vanderperren & W. Dehaene. *From UML/SysML to Matlab/Simulink: current state and future perspectives*. Proceedings of the conference on Design, automation and test in Europe, Munich, 6-10 mars 2006.

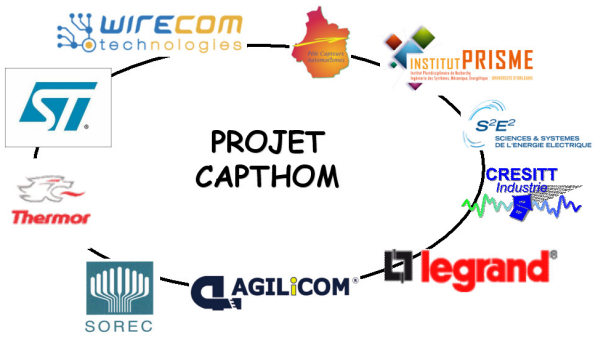
[Villemeur 1988] : A. Villemeur. *Sûreté de fonctionnement des systèmes industriels*. Editions Eyrolles, 1988.

[Willard 2007] : B. Willard. *UML for systems engineering*. Computer standard and interface, Vol. 29, pp. 69-81, 2007.

[Yeh & Hsieh 2007] : R. Yeh & M.-H. Hsieh. *Fuzzy assessment of FMEA for a sewage plant*. Journal of the Chinese Institute of Industrial Engineers, Vol. 24, No. 6, pp. 505-512, novembre 2007.

[Zarras & Issarny 2001] : A. Zarras & V. Isarny. *A UML-Based Framework for Assessing the Reliability of Software Systems*. Proceedings of the 1st ICSE workshop on describing software architecture with UML, pp. 36-40, Toronto, mai 2001.

[Zio 2009] : E. Zio. *Reliability engineering: Old problems and new challenges*. Reliability Engineering and System Safety, Vol. 94, pp. 125-141, 2009.



Nous tenons à remercier les partenaires du projet Capthom. Ce travail a été réalisé avec le soutien financier de la Région Centre et du Ministère de l'Industrie dans le cadre du projet Capthom du pole S^2E^2, www.s2e2.fr.

Pierre DAVID

Contribution à l'analyse de sûreté de fonctionnement des systèmes complexes en phase de conception : application à l'évaluation des missions d'un réseau de capteur de présence humaine

Résumé :

La complexité des nouveaux systèmes ne cesse de grandir, en terme d'intégration de multiples technologies, de nombre de composants, ainsi qu'en terme de performances attendues. De plus, leur phase de conception doit également garantir la tenue de délais stricts pour un coût maîtrisé. Beaucoup de ces systèmes doivent notamment fournir des performances élevées en terme de Sûreté de Fonctionnement (SdF). Les analyses de SdF à mener dans ces cas doivent concilier temps et qualité d'étude, être réalisées à des niveaux de description parfois peu élevés et être capables de traiter de multiples technologies. C'est pourquoi nous avons cherché durant cette thèse à décrire une méthode d'analyse de la SdF des systèmes complexes, intégrée aux méthodes d'Ingénierie Système (IS) utilisées, que nous avons nommé MéDISIS.

Les travaux de cette thèse s'attachent donc à rendre plus efficaces les analyses de SdF au cours du processus d'IS, en créant des outils et méthodes intégrés aux outils actuels de conception et proposant un support efficace pour les études SdF. Les méthodes d'Ingénierie Système Basée sur les Modèles sont les plus efficaces pour la création de tels systèmes. Nous plaçons donc ces travaux dans un référentiel utilisant un modèle central, que nous supposons écrit en SysML. Nous avons alors approfondi 3 axes de recherches : Utiliser le langage SysML comme point d'ancrage du processus d'IS intégrant les analyses de SdF, Extraire des diagrammes SysML fonctionnels les informations nécessaires aux études de risques et Exploiter les représentations SysML pour les études formelles de SdF. Dans ce contexte, nous illustrons l'emploi de MéDISIS sur la problématique du projet CAPTHOM, dont l'objectif est de bâtir une solution innovante pour la détection de présence humaine. Cette recherche nous conduit à la réalisation d'un simulateur de réseau de capteurs capable d'évaluer la réponse à des scénarios de stimulation et d'optimiser le placement des capteurs.

Mots clés : Ingénirie système basée sur les modèles, Analyse de SdF, SysML.

Contribution to realibility analysis of complex systems during their design phase: application to the evaluation of human sensors networks missions

Summary:

The design processes of new innovative systems must be able to master the increasing complexity brought by the need of integrating multiple technologies, utilized by even more components with high awaited performances. Moreover, stakeholders impose strict delays and costs on their development phase. Many of those systems are intended to possess strong reliability performances. This kind of system must include in their design phase complicated reliability analysis, within a limited time and with limited resources. Therefore, we tackle in this thesis the construction of a methodology, named MéDISIS, for integrating reliability analysis in the design processes using Model-Based System Engineering techniques.

The main goal of this work is to make the reliability analysis more efficient during the engineering process, by creating methods and tools using or being included in the current design softwares utilised by system engineers. As we assume that SysML is the best language for Model-Based System Engineering, we developed 3 axes: Using SysML as the language used for the central model of all developments. Extracting the needed information on the dysfunctional behaviour, from functional SysML description studied with classic FMEA-like techniques. Setting translations between SysML descriptions and formal models for reliability studies. Finally, the use of MéDISIS is exemplified on the CAPTHOM project, aiming at developing a new human presence sensor. This study leads us to create a simulation and optimisation software for sensor networks design.

Keywords: Model-Based System Engineering, Reliability Analysis, SysML.

Institut PRISME
ENSIB
88, Boulevard Lahitolle
18020 Bourges Cedex