Approche pour la conception de systèmes aéronautiques innovants en vue d'optimiser l'architecture. Application au système portes passager

Jean Verries

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :

Systèmes Informatiques Critiques

Présentée et soutenue par :
Jean VERRIES

le : 21 Janvier 2010

Titre :

APPROCHE POUR LA CONCEPTION DE SYSTEMES AERONAUTIQUES INNOVANTS EN VUE D'OPTIMISER L'ARCHITECTURE APPLICATION AU SYSTEME PORTES PASSAGERS

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
LAAS-CNRS

Directeur(s) de Thèse :
Abd-El-Kader SAHRAOUI
Mario PALUDETTO

Rapporteurs :
Jean-Pierre BOURREY, Professeur à l'Ecole Centrale de Lille (LGIL)
Thierry SORIANO, Professeur, SUPMECA (LISMMA)

Autre(s) membre(s) du jury
Laurent GROUX, Ingénieur, Latécoère
David DELFIEU, Maitre de conférence à l'université de Nantes (IRCCyN)
Thierry GAYRAUD, Maître de conférence à l'université de Toulouse (LAAS)

Remerciements

Les travaux présentés dans ce mémoire ont été effectués dans le cadre d'une collaboration entre le Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS) au sein du groupe Ingénierie Système et Intégration (ISI), et le groupe Latécoère. Je tiens tout d'abord à remercier les directeurs successifs du LAAS, Monsieur Malik Ghallab et Monsieur Raja Chatila de m'avoir permis de réaliser cette thèse.

J'adresse ensuite toute ma gratitude à mes directeurs de thèses, Monsieur Abd-El-Kader Sahraoui et Monsieur Mario Paludetto. Leurs précieux conseils m'ont aidé, orienté, et m'ont évité de nombreux égarements dans mon travail. Je les remercie infiniment pour la confiance et l'amitié qu'ils ont eue à mon égard. Elles ont contribué à faire de ces trois années une période agréable et enrichissante.

J'adresse aussi toute mes remerciements à Monsieur Daniel Estève pour ses précieux conseils, sa disponibilité et ses encouragements, ainsi qu'à l'ensemble des membres du groupe Ingénierie Système et Intégration.

Durant ces années, j'ai pu bénéficier d'une collaboration étroite avec la société Latécoère. Je tiens ainsi à remercier Monsieur Laurent Groux de m'avoir permis d'intégrer le service de la Direction des Equipements et Systèmes et de confronter mon travail à la réalité industrielle. Mes remerciements s'adressent également à MM. Marc Langon et Jean-Charles Bernier pour leur aide et leur implication dans ma démarche.

Un grand merci à toute l'équipe de la Direction Equipements et Systèmes, ainsi que l'ensemble des personnes de la société Latécoère que j'ai côtoyé pendant ces années de thèse et qui m'ont apporté leur collaboration et leur soutien.

Je voudrai également remercier MM. Thierry Soriano et Jean-Pierre Bourey d'avoir accepté le rôle de rapporteur de ce mémoire, et à Monsieur Delfieu d'avoir pris part au jury. Merci également à Monsieur Thierry Gayraud d'avoir accepté la responsabilité de présider mon jury.

Mes remerciements vont aussi à tout le personnel technique et administratif du LAAS et en particulier à Madame Briand et à Monsieur Berty pour leur collaboration précieuse, et tout particulièrement au cours des derniers mois.

Je voudrai également remercier Corinne, ainsi que mes collègues du groupe Ingénierie Système et Intégration : Romaric, Jacqueline, et Vincent pour nos discussions.

Enfin, je voudrai remercier ma famille, et en tout premier lieu mes parents. Cette thèse n'aurait pas abouti sans leur soutien sans faille et leurs encouragements.

TABLE DES MATIERES

CHAPITRE 1. INTRODUCTION GENERALE 9

CHAPITRE 2. PRESENTATION DE L'INGENIERIE SYSTEME.....15

2.1. Historique de l'ingénierie système..... 15

2.2. Ingénierie Système: problématique industrielle..... 17
2.2.1. Complexité liée aux processus..... 18
2.2.2. Complexité liée au produit..... 18
2.2.3. Complexité liée à la performance système..... 19
2.2.4. Complexité liée au réseau d'acteurs projet..... 19
2.2.5. Complexité liée aux coûts et délais..... 19
2.2.6. Cohérence globale et décisions : problématique industrielle de l'ingénierie système..... 20
2.2.7. Questionnaire pratique..... 20

2.3. Système, conception système et IS..... 20
2.3.1. Systèmes..... 20
2.3.2. La conception système..... 22
2.3.3. L'ingénierie système..... 22

2.4. Principes généraux de l'ingénierie système..... 24

2.5. Processus et normes d'ingénierie système..... 25
2.5.1. Introduction : Processus et cycles de vie..... 25
2.5.2. Cycles de vie de l'ingénierie système..... 26
2.5.3. Définition d'un processus..... 27
2.5.4. Normes d'ingénierie système..... 29
2.5.5. La norme EIA 632..... 30
2.5.6. Les normes IEEE 1220 et ISO 15288..... 36

2.6. Conclusion..... 38

CHAPITRE 3. L'INGENIERIE SYSTEME BASEE SUR LES MODELES 39

3.1. Modèles pour l'ingénierie système..... 39
3.1.1. Différents rôles de modèles d'ingénierie systèmes..... 41
3.1.2. Modèle cognitifs..... 43
3.1.3. Modèle normatifs..... 44
3.1.4. Modèles prédictifs..... 47

3.2. SysML : un langage de modélisation unifié pour l'ingénierie système..... 47
3.2.1. SysML Pour la modélisation des exigences..... 49
3.2.2. SysML pour la modélisation de structures..... 49
3.2.3. SysML pour la modélisation du comportement..... 50
3.2.4. SysML pour la modélisation paramétrique..... 50
3.2.5. Faiblesses et améliorations attendues du langage SysML..... 51

3.3. Méthodes et outils d'IS basée sur les modèles..... 52
3.3.1. Evoluer vers les méthodes d'IS basées sur les modèles..... 52
3.3.2. Méthodologies d'ingénierie système basées sur les modèles..... 55

3.4. Problématique liée à l'interopérabilité des modèles..... 63
3.4.1. Travaux sur l'interopérabilité des modèles en IS..... 64
3.4.2. Modèle de données d'ingénierie système AFIS..... 69

3.5. Conclusion 72

CHAPITRE 4. METHODOLOGIE POUR L'INGENIERIE DE CONCEPTION CONJOINTE DE SYSTEMES AERONAUTIQUES (MICCSA) 73

4.1. Contexte et description de la méthodologie 74
4.1.1. Contexte de la méthodologie et points spécifiques 74
4.1.2. Structure et organisation 77

4.2. Modèle d'informations de la méthodologie 79
4.2.1. Description générale du modèle d'informations 80
4.2.2. Principaux éléments du modèle 81
4.2.3. Eléments support des activités d'Analyse et de V&V. 90

4.3. Application des modèles dans la méthodologie 93
4.3.1. Organisation du modèle et points de vues 93
4.3.2. Propriétés des modèles 94
4.3.3. Ingénierie des modèles et vérification des propriétés 99
4.3.4. Ingénierie des modèles et application de patrons de modélisation 100
4.3.5. Constitution des vues d'ingénierie système 104

4.4. Instanciation du modèle d'ingénierie système 105
4.4.1. Processus d'ingénierie système 105
4.4.2. Evolution du modèle, référentiels et jalons de projets 108
4.4.3. Opérations d'ingénierie des modèles de la méthodologie 111

4.5. Conclusion 114

CHAPITRE 5. MODELES ET APPLICATION DE LA METHODOLOGIE MICCSA ... .115

5.1. Cas d'étude : système porte passagers 116
5.1.1. Exigences sources 116
5.1.2. Eléments principaux du système porte passagers 118

5.2. Profil SysML adapté à la méthodologie 119
5.2.1. Démarche de spécialisation de SysML à la méthodologie 119
5.2.2. Organisation du modèle du projet 121
5.2.3. Modélisation des exigences projet 124
5.2.4. Modélisation de l'architecture opérationnelle 127
5.2.5. Modélisation des architectures logique et physique 129
5.2.6. Modélisation pour les vues validation et comparaison d'architectures 135

5.3. Utilisation de VHDL-AMS 136
5.3.1. Mécanisme d'instanciation de composants 137
5.3.2. Spécificités du profil SysML pour la modélisation des systèmes conservatifs 139

5.4. Mise en œuvre de la méthodologie dans un environnement Eclipse 140
5.4.1. Vérification des propriétés sur les modèles 142
5.4.2. Transformation des modèles SysML 144

5.5. Conclusion 149

CHAPITRE 6. CONCLUSION GENERALE 151

TABLE DES FIGURES 155

LISTE DES TABLEAUX 156

BIBLIOGRAPHIE 157

ANNEXES 161

6.1. ANNEXE A - Axes d'amélioration des pratiques I.S..... 161

6.2. ANNEXE B - catégories d'exigences systèmes 163

6.3. ANNEXE C - Objets et prescriptions selon les types de propriétés systèmes..... 164

6.4. ANNEXE D – Règles et Patrons de modélisation..... 167

6.5. ANNEXE E - Patrons de modélisation..... 168

6.6. ANNEXE F - Contraintes associés aux stéréotypes du profil 172

6.7. ANNEXE G - Métamodèle SysML..... 173

6.8. ANNEXE H - Règles XTend de transformation de modèle 175

6.9. ANNEXE I - Règles XPAND de génération de code..... 178

6.10. ANNEXE J - Code de simulation VHDL-AMS simplifié du système d'assistance à l'ouverture de porte. 180

Chapitre 1. Introduction Générale

Afin de consolider sa position de leader mondial de fabricant de portes d'avions commerciaux, la société LATECOERE a engagé en 2006 un programme d'étude et technologie devant déboucher sur un *démonstrateur* de portes innovantes destinées aux avions de futures générations. La porte *nouvelle génération* devra intégrer des évolutions technologiques dans sa structure et ses équipements au sein d'une architecture optimisée.

Souhaitant s'appuyer sur les méthodologies et outils de pointe dans le domaine de la conception et la simulation des systèmes, LATECOERE a engagé une collaboration recherche-industrie avec le LAAS dans un projet commun de recherche. Ce projet de recherche consiste à développer et valider une méthodologie nouvelle de conception d'un *système porte* permettant de garantir le respect des exigences du client et des autorités de certification avec un produit performant et compétitif.

Le **contexte** des travaux de recherche, objets de ce mémoire, est ainsi celui de la démarche de Conception Système. Dans nombre d'entreprises, les objectifs de productivité et de réduction des délais ont incité les chercheurs et ingénieurs à faire émerger des démarches descendantes de conception qui s'appuient sur des représentations virtuelles du système avant de décider de son intégration matérielle. Sur cette base, le LAAS et la Société LATECOERE ont établit une collaboration sous forme d'un projet afin de lever certains verrous technologiques que pose cette démarche globalement descendante. La stratégie de collaboration entre l'entreprise et le laboratoire concerne la conception système, depuis l'écriture des spécifications jusqu'à une représentation virtuelle du produit: prototypage virtuel dont les composantes sont des représentations de la réalité physique, matérielle et logicielle. Le travail de cette thèse concerne la démarche conceptuelle comportant les étapes suivantes:

a) L'écriture des spécifications conformément au standard des métiers et aux exigences des méthodes et outils.

b) L'élaboration de plusieurs architectures systèmes et leur modélisation logique haut niveau

c) La validation et la vérification de la modélisation par rapport aux spécifications.

d) Le choix des architectures systèmes répondant au mieux, à la fois, aux exigences réglementaires/clients et aux technologies des composants du système.

Le **cadre méthodologique** est celui de l'Ingénierie Système, des processus afférents et des exigences et normes spécifiques aux systèmes avioniques en général et au système à concevoir en particulier (GRESS, ABD 200, EIA632).

Ainsi les **objectifs** de ces travaux s'inscrivent dans une démarche globalement descendante de conception qui s'appuie sur des représentations virtuelles du système avant de décider de son intégration matérielle. Sur cette base, une première ambition de modélisation est d'aboutir à plusieurs représentations virtuelles et complètes du système, placées dans son environnement d'utilisation. Sur la base des connaissances et des pratiques actuelles, ces représentations virtuelles sont réalisées par le couplage de modèles nombreux et hétérogènes. Cette approche fait appel à la modélisation multi-niveaux pour exprimer, dans un même domaine disciplinaire, la hiérarchie des modèles possibles entre la modélisation physique, la modélisation comportementale et la modélisation système. Elle fait également appel à la modélisation multi-modèles pour exprimer le caractère pluridisciplinaire des systèmes, impliquant d'élaborer des modèles hétérogènes complémentaires qui sont ensuite associés dans la simulation globale.

Un objectif fort de nos travaux de recherche est de déterminer les solutions d'architecture système répondant aux différentes exigences élaborées en début du projet, démarche dite *top-down*. Le cadre et le contexte industriels exigent également que soient considérées les solutions technologiques de types composants systèmes, solutions émanant de travaux antérieurs ou parallèles, et donc de démarches de type *bottom-up*, afin d'établir plusieurs architectures et d'opérer des choix circonstanciés.

Il n'existe pas une méthodologie universellement reconnue répondant à une telle problématique. Il existe des méthodologies plus ou moins dédiées. Celles-ci s'appuient essentiellement sur un cahier des charges ad-hoc et considèrent rarement un cahier des charges à construire. Les premières exigences de l'approche à développer consistaient ainsi à dégager de l'expertise de l'entreprise et de ses métiers autour du système à concevoir :

a) des spécifications qui expriment les besoins fonctionnels et les exigences fonctionnelles et non fonctionnelles,

b) une approche regroupant méthode, modèles et outils pour assurer, ce qu'il est convenu d'appeler en Ingénierie Système (IS), l'Ingénierie des Exigences (IE - maîtrise des évolutions, de la volatilité, des incohérences, … des besoins et des exigences),

c) une modélisation du système à concevoir dans un cadre MDA (Model Driven Architecture),

d) un prototypage virtuel du système permettant, en association avec l'évaluation formelle des modèles, une première évaluation des performances des architectures étudiées ; le tout permettant d'effectuer certains choix relativement tôt dans le cycle de vie du système.

Un autre objectif fort du projet était qu'il débouche sur une plateforme de conception système qui permettrait aux ingénieurs de l'entreprise de procéder à des tests systèmes et d'évaluer les approches, les méthodes et les outils. Cette plateforme serait le siège de recommandations opérationnelles : spécifications, évaluation de performances, dimensionnement, test par simulation et co-simulation, définition du domaine de validité, mise en place de procédures de *tolérance* (analyse de sensibilité, allocation de tolérances, etc.). L'approche à étudier devait se fonder sur les propriétés permettant d'exploiter les liens

qui peuvent être instaurés dans une modélisation multi-niveaux. Pour le système à concevoir, une telle modélisation s'étend depuis les matériaux jusqu'au système lui-même ainsi que depuis les spécifications jusqu'au prototypage virtuel.

Concernant les spécifications, un verrou technologique à lever concernait l'élicitation et la modélisation des exigences. Leur catégorisation en termes d'exigences fonctionnelles et d'exigences non fonctionnelles (principalement de sécurité) permet une modélisation dès les premières phases du cycle de vie du système. Le point de départ du sujet est donc le cahier des charges où se retrouvent toutes les spécifications, contraintes et exigences. Le travail de thèse consistait d'abord à tirer du cahier des charges les éléments d'une formalisation en fonction de la dite catégorisation.

A priori, le support envisagé pour la représentation du système et de ses abstractions était, entre autres, le langage SysML en cours de standardisation. Les aspects V&V (Validation et Vérification) allaient imposer rapidement l'extension de ce langage vers des notations formelles. SysML étant un langage et non une méthode, cette extension, support ou profil devait également être définie et adaptée à la construction d'une démarche de description pas à pas conduisant à une formalisation en des éléments du langage. Cette formalisation pouvait être directe ou se faire par couplage au sein d'une plateforme du laboratoire en cours de développement ; l'intérêt de la plate-forme réside dans la facilité des couplages des outils de simulation tels que Matlab-simulink, VHDL-AMS, systemsC ou autres.

Une autre approche consiste à envisager la plateforme autour d'Eclipse en intégrant les produits du marché s'appuyant sur SysML et l'EIA632 comme démarche d'intégration. L'approche envisagée serait exploitée et orientée suivant deux objectifs essentiels:

- la traçabilité pré et post ; la frontière se situe entre les phases d'élaboration des exigences jusqu'à leur adoption et à partir de leur adoption jusqu'à leur mise en œuvre; la traçabilité au plus haut niveau concerne, elle, toute la partie gestion des exigences,

- la vérification, pour laquelle sont envisagées des méthodes nouvelles: vérification de cohérence des différents points de vue le plus tôt possible dès le début de la formalisation.

La suite de la démarche se focaliserait sur l'implémentation de processus de vérifications fondés sur la simulation et la co-simulation. Elle bénéficierait largement de l'expertise et des outils de la communauté, du laboratoire et de l'entreprise afin de:

- proposer une architecture système, basée MDA, faite de composants réutilisables pour modéliser des objets physiques comportant du matériel et du logiciel. Ce partitionnement doit répondre à des critères fonctionnels et non fonctionnels qu'il convient de définir et de mettre en œuvre. Cette architecture contribuerait ainsi au couplage des méta-modèles, des simulateurs et des outils existants.

- spécifier des composants dans la perspective d'un appel d'offre du maître d'ouvrage à destination de ses fournisseurs. Les outils de représentation généraux et de vérification associés continueront à jouer un rôle d'arbitres de cette étape.

Enfin, ce travail devait s'inscrire à chaque étape d'un processus de référence global et intégrer des outils et des méthodes de vérification créés à partir d'approches formelles et de simulations partielles et globales.

Ce sont tous ces objectifs que nous avons tenté d'atteindre pendant cette thèse dont les travaux sont décrits dans ce mémoire. Ainsi, la méthodologie que nous avons définie, est destinée à proposer au groupe LATECOERE un moyen de progresser dans son aptitude à développer des systèmes aéronautiques répondant aux attentes de ses clients, et de manière optimale face aux contraintes industrielles du domaine de l'aéronautique et spécifiques à l'entreprise. Elle propose, en particulier, **une méthode adaptée au type de produit spécifique à cette entreprise, dont la particularité et de mêler étroitement des éléments de structure aux éléments plus traditionnellement associés au domaine *système* : composants, calculateurs et cartes électroniques embarquant du logiciel**. Enfin, la mise en place d'une telle méthodologie doit permettre de rendre son activité plus profitable, d'accroitre les compétences du groupe en tant qu'acteur majeur de l'industrie aéronautique et de gagner de nouveaux marchés.

L'approche Ingénierie Système (IS) peut être vue comme un moyen de maitriser la complexité tout en assurant les objectifs fixés. Elle permet, par un ensemble de bonnes pratiques et de contraintes déclinées sur les processus et les méthodes, de rendre plus efficace et plus sûr le développement d'un système complexe ainsi que sa maîtrise tout au long de sa vie opérationnelle.

Le travail réalisé au cours de cette thèse vise donc essentiellement à proposer à LATECOERE une approche de l'IS innovante pour la société et conforme aux bonnes pratiques accumulées dans le domaine. Le sens de l'innovation doit être vu en relation avec les méthodes employées, à adapter au contexte de l'entreprise, et leur mise en œuvre sur des outils logiciels. Ainsi l'approche propose un usage intensif de la modélisation et utilise la rigueur apportée par le respect des formalismes de modélisation pour guider et assister le développement d'un système complexe. **De ce fait, le travail présenté se situe au carrefour de l'IS et de l'Ingénierie Dirigée par les Modèles (IDM).**

Pour cela, ce mémoire de thèse est **organisé autour de cinq chapitres**. Après la problématique et les objectifs énoncés dans cette introduction générale, le chapitre II présente le domaine de l'IS en général mais également avec les attentes LATECOERE en filigrane. Il retrace rapidement l'historique de cette discipline, sa naissance et les principales étapes qui ont marqué sont évolution. L'approche d'IS sera également justifiée en précisant quels en sont les enjeux techniques, au niveau des industries en général, puis au niveau de la société LATECOERE en particulier. Les définitions principales du domaine sont ensuite énoncées et discutées. Les descriptions de ces définitions amènent à aborder les principes généraux et les bonnes pratiques classiques du domaine de l'IS avant de présenter et de comparer les principaux standards d'IS actuellement en usage.

Le chapitre III poursuit l'étude en se focalisant tout d'abord sur les principaux langages de modélisation utilisables dans le domaine de l'IS. En particulier, le langage SysML, utilisé dans ces travaux, est détaillé. Ensuite, les principales méthodes d'IS basées sur les modèles utilisées aujourd'hui sont décrites et comparées entre elles. Les limitations inhérentes à ces méthodes sont présentées et des axes d'amélioration identifiés. Ce chapitre constitue donc une introduction et un état de l'art technique pour aborder le chapitre suivant entièrement axé sur le travail demandé.

Le chapitre IV présente la méthodologie pour l'IS proposée à LATECOERE. Cette méthodologie basée sur les modèles s'appuie sur plusieurs concepts de départs qui seront explicités. Ensuite, elle est présentée selon deux perspectives. Tout d'abord, le modèle d'information ou métamodèle autour duquel elle s'articule sera décrit. Ensuite, l'instanciation des éléments de ce modèle d'information est formalisée par la définition de processus d'ingénierie système adaptés aux systèmes aéronautiques fabriqués par LATECOERE et conformes aux standard EIA-632. Un ensemble de méthodes orientées modèles utilisant des patrons de modélisation est également présenté comme partie prenante de la méthodologie. Pour des facilités d'écriture et de présentation nous avons adopté l'acronyme MICCSA (Méthodologie pour l'Ingénierie de Conception Conjointe de Systèmes de l'Aéronautique).

Le chapitre V propose une mise en œuvre de la méthodologie MICCSA au niveau des outils (Eclipse, OAW, VHDL-AMS). Elle est illustrée à travers d'un cas d'étude qui est le *système porte passager*. Cet exemple sera tout d'abord introduit afin de démontrer sa représentativité et de mieux comprendre ses caractéristiques. Ensuite, les modèles mis en place grâce à l'application de la méthodologie sont présentés. Viendront également dans ce chapitre les outils utilisés pour la mise en œuvre de la méthodologie ainsi que les résultats produits par ces derniers. Cette mise en œuvre a nécessité, en particulier, des transformations de modèles SysML vers VHDL-AMS pour des besoins de simulation. La validation formelle s'appuie, quant à elle, sur le langage de contraintes OCL et de manière classique sur les modèles formels de type STD (State transition Diagrams).

Les objectifs étant ambitieux, la dernière partie de ce mémoire effectuera, dans un chapitre conclusion, un bilan du travail effectué, une évaluation des objectifs atteints par rapport aux objectifs fixés initialement et ouvrira les perspectives raisonnables, au sens atteignables, à court terme et à plus long terme.

Chapitre 2. Présentation de l'ingénierie système

Ce chapitre est une introduction à l'IS. Il présente la problématique générale du développement d'un système complexe. Un historique rapide permet de comprendre l'origine de l'IS, d'abord vue comme un ensemble de techniques et de bonnes pratiques peu formalisées, puis comme une véritable discipline soutenue par des organisations nationales et internationales, des normes, et des méthodes.

Les enjeux actuels de l'IS appliquée au développement des systèmes et équipements aéronautiques sont présentés, avant de présenter les différents aspects de l'IS par le biais de sa **définition par les organismes et les standards reconnus** dans le domaine.

Enfin, un certain nombre de définitions sont retenues comme référence pour la suite de ce rapport, et les principes généraux et les **normes** actuelles de l'IS seront présentées.

2.1. Historique de l'ingénierie système

L'IS trouve son origine au cours de la seconde guerre mondiale. Durant cette période de l'histoire se conjuguent une progression rapide des technologies et une demande forte de systèmes aptes à faire face aux enjeux majeurs militaires et politiques.

Ainsi, on peut situer les premiers efforts d'IS, ou de réflexions quant à l'approche et aux techniques aptes à satisfaire aux enjeux techniques et organisationnels que posent le développement, la production et le maintien en opération de systèmes complexes aux années 1940. Le terme *ingénierie système* apparait alors pour la première fois dans le cadre des travaux du laboratoire Bell et du projet *Nike. Nike* constitue à sa livraison, en 1953, le premier système de défense de l'espace aérien destiné à l'armée américaine. La vitesse et l'altitude de vol des avions, nouvellement équipés de moteur à réaction, rendaient les moyens de défenses inefficaces. L'enjeu d'un développement tel que celui du système *Nike* était alors d'intégrer un ensemble d'équipement complexe de détection, de commande et de guidage de missiles rapides.

L'enjeu majeur de l'effort d'ingénierie de ce système est de contrôler et maîtriser les propriétés et les caractéristiques locales du système complexe, en maîtrisant les effets induits par les interactions de chacun de ces constituants. Auparavant, plusieurs organisations avaient eu recours implicitement a une approche système, sans l'avoir définie clairement comme une activité en tant que telle. L'IS va, à partir des années 1950, concerner les grands systèmes impliquant de nouvelles technologies et des êtres humains, et jouant un rôle important face aux enjeux militaires ou de sécurité.

Dans le contexte historique de la guerre froide, le système de défense anti-aérien SAGE-ADS (*Semi-Automatic Ground Environment – Air Defense System*), initié et développé par le MIT entre les années 1955 et 1960, a modernisé le système de surveillance et de défense de l'espace aérien américain, et a fait évoluer l'IS par son approche d'intégration système à grande échelle. En particulier, le système reposait sur l'intégration des technologies RADAR et des capacités informatiques au sein d'un système complexe, réparti entre des éléments centraux informatiques et des unités de détection RADAR, utilisant le réseau téléphonique comme moyen de communication. L'ensemble du système devait, par ailleurs, respecter des contraintes temporelles fortes liées à sa mission de protection de l'espace aérien.

L'intégration de ces technologies récentes dans une approche système a, par ailleurs, contribué au développement du système de contrôle de trafic aérien civil, et à l'intégration au niveau national des dispositifs de commandement, de contrôle et de communication des systèmes militaires.

Par la suite, deux grandes institutions américaines, l'USAF et le DoD ont tenté, dans les années 1960, de mieux maîtriser la complexité de leurs programmes militaires et d'exploration spatiale au travers de pratiques industrielles plus rationnelles et standardisées. Les invariants dans les activités à mener et les points communs dans les méthodes appliquées sur divers projets à grandes échelles les ont menées à définir des processus tels que le processus SIMILAR [Bah_05], à définir des normes et des standards et à capitaliser les bonnes pratiques.

Arthur Hall, alors ingénieur chez AT&T et acteur majeur du domaine de l'IS, publie en 1962 *A methodology for engineering of systems* [Hal_62], et pose les premières pierres des éléments actuels de l'IS :

- spécification des exigences au niveau système,
- gestion des interfaces,
- respect de jalons de développement,
- gestion du changement,
- techniques d'analyse de compromis et de prise de décisions.

Ces techniques se développent et sont adoptées progressivement dans les industries développant des grands systèmes. En 1969, le standard militaire Mil-Std 499 [MIL_69], constitue la première norme regroupant les bonnes pratiques de l'IS. La norme a alors pour objectif de permettre au gouvernement d'évaluer objectivement la compétence en IS de ses sous-traitants potentiels, et charge ces derniers d'améliorer leurs pratiques de l'IS et de la conduite de l'effort d'ingénierie tout au long d'un développement.

En 1990, la *National Council on Systems Engineering* (NCOSE), premier organisme mondial d'ingénierie des systèmes, est créée à l'initiative d'un grand nombre d'entreprises américaines. Le NCOSE a alors pour mission d'améliorer et de fédérer l'utilisation et la diffusion, notamment par l'enseignement, des pratiques en ingénierie des systèmes : processus techniques, processus de management, méthodes et outils.

La NCOSE est devenu officiellement une organisation internationale, l'INCOSE, en 1995. Elle est aujourd'hui l'organisation principale de la communauté de l'IS, devenue une discipline à part entière. Divers organismes nationaux, tels l'Association Française d'IS

(AFIS), sont partenaires de l'INCOSE et permettent aux grandes entreprises industrielles de mener des réflexions et des travaux communs sur les différents aspects de l'IS.

Conjointement aux travaux de l'INCOSE, l'institut des ingénieurs en électricité et électronique (IEEE), et l'Alliance des Industries de l'Electronique (EIA), un regroupement d'entreprises américaines dans le domaine des hautes technologies, ont défini des processus d'IS génériques sous la forme de normes. Ces normes modernisent, complètent et élargissent le périmètre d'application du standard MIL-499, bien qu'elles en reprennent aussi de nombreux éléments. Les grandes organisations telles que le département de la défense américaine (DoD), la NASA [NAS_95] ou le Centre National d'Etudes Spatiales (CNES) ont aussi développé leur propres standards d'IS.

Une date importante de l'histoire de l'IS est l'année 1994. Les Etats-Unis mènent alors une politique de réduction des dépenses de l'armée américaine, et des initiatives sont engagées afin de rationnaliser l'utilisation des standards militaires pour le développement, la production et le maintien en opération des matériels militaires [Perr_94]. Déconseillant l'usage de la plupart des standards militaires, alors trop contraignants, trop coûteux, et trop nombreux (plus de 30.000 standards sont alors en service), le secrétaire de la défense préconise l'utilisation systématique de la norme IEEE1220 [IEE_99] comme standard de référence d'IS, applicable, en particulier, pour guider l'activité d'acquisition du matériel ainsi que des standards industriels tels l'ISO9000.

Cette décision marque un pas significatif dans l'adoption des techniques d'IS par de grandes institutions. L'ingénierie des systèmes est alors vue comme une discipline à part entière et incontournable dans le développement des grands systèmes et des systèmes complexes.

2.2. Ingénierie Système: problématique industrielle

L'IS est une discipline vaste et implique plusieurs domaines. Cette section précise le périmètre de l'IS tel qu'il est traité dans ce document.

Le travail de cette thèse porte sur l'ingénierie des systèmes aéronautiques, en particulier sur le développement des équipements et systèmes comportant une partie mécanique, une partie électronique et une partie logicielle.

La partie mécanique peut ou non comporter des éléments dynamiques et des actionneurs de nature électromécanique, pneumatiques, hydraulique ou basé sur toute autre technologie.

Les parties électroniques et logicielle concernent principalement des éléments calculateurs, des moyens de communications et d'informations simples ou complexes.

Cette section précise, dans un premier temps, les enjeux de l'IS dans ce contexte en les déclinant sur différents aspects, puis identifie la problématique de la thèse du point de vue industriel au regard de chacun de ces aspects.

L'enjeu de l'IS revient à la maîtrise des complexités inhérentes au projet et au produit, plus précisément, différentes sources de complexité, décrites ci-dessous, peuvent être identifiées.

2.2.1. Complexité liée aux processus

Le développement d'un produit est intrinsèquement un processus interdisciplinaire nécessitant l'implication de différents métiers. Dans ce contexte, l'ingénierie simultanée est mise en œuvre pour réduire le temps de développement et prendre en compte au plus tôt les activités et également les moyens à mettre en œuvre tout au long du développement du système (moyens logistiques, de production, de support client, etc.).

Les contraintes de sécurité associées nécessitent que les processus employés pendant le développement se conforment au processus standard de développement des systèmes aéronautique ARP4754 [ARP_96], incluant des analyses et des évaluations de sécurité. Le développement des sous-ensembles électroniques et logiciels doivent se conformer respectivement aux exigences de processus des DO254 [DO_00] et DO178B [DO_92].

De plus, les processus utilisés pour le développement au niveau système doivent se conformer aux exigences de processus du client. Ces exigences concernent, en particulier, la gestion des exigences, la gestion de configuration, le contrôle des évolutions des phases de spécification, de conception, d'intégration, de validation, de vérification et de qualification. Pour chacun de ces aspects, les moyens à mettre en œuvre et les résultats attendus peuvent faire l'objet d'exigences spécifiques.

Le développement actuel est articulé autour d'un système documentaire qui structure les étapes et le déroulement des activités du développement. La maîtrise de la réalisation des processus de l'entreprise appliqués dans le cadre d'un projet spécifique est un facteur principal de réussite ou d'échec du projet.

2.2.2. Complexité liée au produit

Il s'agît de la maîtrise de l'architecture, des interfaces et des constituants du produit. La maîtrise des interfaces externes (les interfaces en contact avec l'environnement du système) et internes est capitale pour la réussite du projet.

La spécification, l'analyse et le développement des interfaces jouent un rôle prépondérant dans la maîtrise des effets indésirables produits par l'interaction entre les constituants du système, entre constituants et environnement opérationnel, et également dans la capacité du produit à évoluer et à être maintenu.

Maîtriser la complexité du produit revient donc en grande partie à définir une architecture dont les interactions entre sous-ensembles sont telles que :

- les effets émergents permettent de réaliser les fonctions attendues au niveau système,
- les effets émergents indésirables sont identifiés, évités, confinés ou compensés.

L'interface externe avec l'opérateur doit respecter des contraintes spécifiques pour l'ergonomie et la sécurité d'utilisation.

Le choix des constituants et de leur organisation doit être fait en cohérence avec les capacités de l'entreprise et les décisions concernant les activités sous-traitées. La réutilisation de solutions doit être envisagée à chaque étape du développement. Ce choix a un impact important sur la planification et la conduite des activités d'intégration, de validation, de vérification et de qualification. Par ailleurs, la configuration du système doit être suivie et maîtrisée dès la définition d'une solution de conception détaillée et ce, jusqu'à la fabrication. Les écarts par rapport aux exigences initiales et les performances du produit doivent être mesurés et communiqués.

2.2.3. Complexité liée à la performance système

Les multiples exigences de performances formulées sur le produit sont très souvent contradictoires et nécessitent, dans toutes les phases de conception, des activités de comparaison de solutions et de recherche de compromis.

La prédiction des performances du produit au plus tôt dans les phases de conception, lorsque le produit n'existe pas encore physiquement, joue un rôle prépondérant pour éviter des itérations dans le développement. Associé à la complexité intrinsèque du produit, elle est une source de complexité supplémentaire qui influence l'adéquation du système au besoin initial.

2.2.4. Complexité liée au réseau d'acteurs projet

Cette source de complexité peut être déclinée selon deux aspects : tout d'abord, un réseau d'acteurs internes à l'entreprise. Quelque soit le type de projet et de produit, plusieurs métiers et plusieurs compétences sont impliquées dans son développement. Une communication efficace entre chacun des acteurs joue un rôle prépondérant pour le bon déroulement du projet. Cet aspect collaboratif doit permettre de réaliser des analyses justes et les bons arbitrages tout au long du projet, et en particulier lors des phases préliminaires de définition d'architecture, conditionnant la réussite du projet.

Généralement, un projet implique au moins deux acteurs : le(s) client(s) (avionneurs ou compagnies aérienne), et l'entreprise elle-même. Celle-ci peut également décider de sous-traiter une partie de ses activités (conception de sous-ensemble, essais de qualification…), produisant ainsi de nouvelles interfaces et de nouveaux besoins de communication entre les acteurs du projet. L'efficacité de la communication entre ces acteurs et la maîtrise des aspects contractuels des éléments transmis entre les acteurs du projet joue un rôle clé dans la réussite du projet et la satisfaction des objectifs de l'entreprise.

2.2.5. Complexité liée aux coûts et délais

Le développement d'un système aéronautique se déroule dans un programme avion, à partir duquel des contraintes fortes de coûts et de délais sont associées et contractualisées entre le client et l'entreprise, puis entre l'entreprise et ses sous-traitants. Ainsi, chaque source de complexité peut avoir un impact important sur les coûts et délais de développement, ainsi que sur le coût global du système.

2.2.6. Cohérence globale et décisions : problématique industrielle de l'ingénierie système

Chaque source de complexité présentée ci-dessus peut être vue comme un un facteur de probabilité de réussite ou d'échec du projet, par rapport aux objectifs de l'entreprise. Dans la conduite du projet, et particulièrement dans ses phases les plus amonts, chacun des ces aspects doit être pris en compte (au regard des informations disponibles ou anticipées). Les moyens et les activités associées doivent être planifiés, les risques associés identifiés et quantifiés et maîtrisés tout au long du développement.

La maîtrise de ces aspects constitue l'objectif principal de l'IS dans notre travail de thèse. Le problème peut être reformulé sous la forme de la question suivante :

Comment traduire le besoin opérationnel du demandeur du système en objectifs techniques sur chacun des aspects, et comment s'assurer que ces objectifs techniques pourront être atteints en respectant les objectifs et les contraintes propres de l'entreprise ?

2.2.7. Questionnaire pratique

Dans l'objectif de préciser la problématique d'IS de la thèse sur la base des pratiques de l'entreprise LATECOERE, un questionnaire a été conçu et soumis aux principaux métiers de l'entreprise.

Cette démarche a permis d'identifier certaines difficultés pour l'entreprise à pratiquer de la *bonne* IS. En particulier, les pratiques de gestion des exigences par les différents métiers ont été analysées pour remonter aux difficultés liées à la pratique de l'IS tel qu'elle est réalisée actuellement. Une synthèse des éléments résultants de cette démarche est présentée dans l'annexe A.

2.3. Système, conception système et IS

Ce paragraphe présente une brève description de trois notions qui seront utilisées tout au long de ce document. Il s'agit d'une analyse rapide l'activité de conception, des notions de système et d'IS, accompagnée d'un ensemble de définitions servant de référence.

2.3.1. Systèmes

Charles S. Wasson [Was_06] définit un système comme *un ensemble intégré d'éléments interopérables, dont les capacités sont spécifiées et bornées, et fonctionnant en synergie pour réaliser une tache à valeur ajoutée, afin de satisfaire les besoins opérationnels d'un utilisateur dans un environnement opérationnel particulier, et avec un résultat et une probabilité de succès spécifiée.*

Cette définition permet d'aborder les points principaux qui caractérisent un système :
Un ensemble intégré d'éléments : un système peut être perçu comme un ensemble de constituants interfacés entre eux. Ces constituants peuvent être des dispositifs de natures

technologiques différentes (dispositifs électroniques, mécaniques, hydauliques, etc.) mais aussi des moyens humains, des installations, des services et des procédés.

fonctionnant en synergie : Sur ce point, Brian W. Mar [Mar_97] rappelle plusieurs propriétés permettant de faire apparaitre la caractéristique d'émergence liée aux interactions entre les constituants d'un système. Ce principe est communément résumé ainsi : *le système pris dans sa globalité est plus que la simple somme de ses constituants*. D'un autre côté, le système pris dans son ensemble détermine la nature de ses constituants.

pour réaliser une tâche à valeur ajoutée : Le système existe car il réalise une ou plusieurs tâches ou services qui apportent une valeur aux yeux de son utilisateur ou de celui qui le possède.

dans un environnement opérationnel : La notion d'environnement est également centrale dans la définition d'un système, car elle sous-tend qu'un système possède une frontière le séparant de l'extérieur et qu'il interagit avec celui-ci d'une manière connue et maîtrisée. Une partie de cette interaction constitue les services ou la mission qui sont attendus du système. L'environnement opérationnel est l'environnement dans lequel le système est plongé lorsqu'il rend ces services.

D'autre définitions, plus concises, ont été intégrées dans les normes d'ingénierie système et/ou reconnu par les organismes du domaine. Ainsi, pour l'INCOSE, un système est *un ensemble intégré d'éléments qui accomplissent un objectif défini* [INC_04]. La définition qui sera retenue dans ce travail, très similaire, est celle qui est retenue par l'AFIS et l'ISO :

Un système est un ensemble d'éléments en interaction, organisés pour atteindre un ou plusieurs résultats déclarés [Source : AFIS, ISO 15288]

Dans le domaine de l'industrie aéronautique civile, la notion de système a une définition particulière liée à la décomposition du produit avion en sous-ensembles. Un système désigne alors un ensemble de constituants réalisant une, ou participant à plusieurs, fonction(s) de niveau avion. Les fonctions de niveau avion sont elle-même identifiées selon un découpage commun dans l'ATA-100 (Air Transport Association, Spec. 100).
Dans ce contexte, le terme *système* est donc associé à un certain niveau hiérarchique dans la décomposition structurelle d'un avion en ses constituants. Ensuite, un système peut être décomposé en sous-ensembles ou en équipements, désignant ainsi des éléments qui participent avec d'autres équipements et sous-ensembles à une ou plusieurs fonctions de niveau avion. La définition retenue pour un système dans le domaine aéronautique est donc la suivante :

Dans le contexte aéronautique, un système est une combinaison d'éléments interconnectés pour implémenter une fonction ou un groupe de fonctions de niveau avion. [Source : ARP 4754]

Afin de rester cohérent, le terme de *système* sera employé dans sa définition la plus générique. Lorsqu'il sera utilisé dans sa définition propre à l'aéronautique, le système sera alors appelé explicitement *système aéronautique*.

L'IS s'appuie par ailleurs sur la conception système. La section suivante présente les grandes lignes qui constituent l'activité de conception.

2.3.2. La conception système

La conception d'un système est une activité complexe. Elle consiste à mener conjointement des activités afin de parvenir à une solution satisfaisant un besoin formulé et un ensemble de contraintes.

Les activités regroupées sous le terme de conception sont multiples. Nous pouvons en citer quelques unes :

- collecte, structuration et traitement de données techniques,
- décomposition de problème global en problème locaux plus simples,
- établissement et accroissement de la connaissance autour d'un problème spécifique,
- recherche de solutions par analyse intuitive et créative,
- prise de décision sur la décomposition d'éléments ou sur des éléments de solution,
- application et ajustement de schémas de résolutions génériques à un problème spécifique,
- optimisation de paramètres liés entre eux, etc.

Toutes ces tâches sont menées conjointement, pour parvenir à une solution satisfaisante. Celle-ci fait appel à la fois a des caractéristique génériques d'ordonnancement, de maîtrise des incertitudes et des risques, et à des compétences techniques liées au(x) domaine(s).

Des travaux de recherche permettent d'adopter une approche de conception adaptée à un problématique spécifique ou au développement d'un système particulier. M. Sautreuil a par exemple proposé une approche [Sau_09] permettant une conception optimisée des organes de génération électrique du réseau électrique avion. L'approche qui est proposée dans ces travaux tient compte de la complexité liée au développement conjoint des organes de génération, de servitudes, et d'un réseau complexe d'interconnexions et de contrôle, puis de l'intégration de ces éléments. Pour cela, la notion de robustesse d'un composant système a été utilisée comme un moyen de mieux maîtriser la capacité d'un équipement à être intégré dans son environnement final. Cette notion a été déclinée au niveau technique par un indicateur lié aux lois de commande du système étudié.

Une fois les activités ci-dessus ramenées à une problématique générale et aux enjeux de maîtrise des complexités abordées au paragraphe 1.2, l'existence du domaine de l'IS, comme contribution à répondre à cette problématique, apparaît clairement justifié.

2.3.3. L'ingénierie système

Il existe une grande variété de définitions de l'IS. Cette variété est probablement liée à la diversité des contextes dans lesquels peuvent être appliquées les approches d'IS : contexte culturel, technique, organisationnel, …

Il existe, cependant, des constantes dans ces définitions. Cette section propose plusieurs définitions, puis tente d'apporter une interprétation synthétique de celles-ci et définit enfin celle qui sert de référence pour ce travail.

Pour l'INCOSE, l'IS est simplement *une approche et un ensemble de moyens pour assurer la réalisation satisfaisante d'un système* [INC_04].

La NASA adopte une définition pragmatique et insiste sur les phases de cycle de vie et sur les bonnes pratiques caractéristiques de l'IS : *L'ingénierie système est une approche robuste pour la conception, la fabrication et l'exploitation d'un système. En termes simples, l'approche consiste en l'identification et la quantification des buts du système, la création d'alternatives de conception, la réalisation de compromis lors de la conception, la sélection, de l'implémentation, l'intégration et la vérification et validation des meilleurs concepts par rapport aux buts fixés* [NAS_95].

Nous pouvons trouver également des définitions plus personnelles telles que de D. Hitchins, président de l'INCOSE en 2007 l'annonce : *l'IS est l'art et la science de créer des systèmes efficaces, utilisant le système dans son intégralité et dans l'intégralité de la vie opérationnelle du système ; ou l'art et la science de créer une solution optimale à un problème complexe.*
Cette définition désigne l'activité d'IS comme *un art et une science* soulignant ainsi la difficulté liée à la multitude des arbitrages et à la valeur ajoutée irremplaçable apportée par l'humain dans la recherche d'une *solution optimale*.

Jean-Pierre Meinadier définit l'IS comme *la recherche d'un équilibre entre savoir, savoir-faire et créativité*. Il parle ainsi du travail de l'ingénieur système : *le technicien maîtrise sa technique; l'ingénieur possède l'art de concevoir des produits pour un marché en adaptant des techniques existantes; l'ingénieur système exerce l'art de concevoir des systèmes mettant en jeu un grand nombre de métiers, dont il ne peut maîtriser tous les gé*nies.

L'ingénieur système est l'ingénieur généraliste qui maintient une vision globale. La méthode employée est celle d'une approche collaborative. Sur un système de grande dimension, une équipe de scientifiques et d'ingénieurs, de généralistes et de spécialistes, exercent un effort collectif pour définir une solution et la réaliser physiquement. Cette technique a été appelée *approche système* ou *méthode de développement par équipes* [Goo_57].

La méthode d'ingénierie système part du principe que chaque système est un tout même s'il est constitué de structures et de sous-fonctions nombreuses et spécialisées. De plus, l'IS défend l'idée selon laquelle chaque système possède un certain nombre d'objectifs et que l'équilibre existant entre ces objectifs peut différer largement d'un système à l'autre. L'approche système cherche à optimiser les fonctions globales du système selon les objectifs pondérés afin d'obtenir la meilleure compatibilité entre ses éléments [Ches_65].

Pour sa part, l'AFIS décrit l'IS comme :

- un processus coopératif et interdisciplinaire de résolution de problème,
- s'appuyant sur les connaissances, méthodes et techniques issues de la science et de l'expérience,

- mis en œuvre pour définir, faire évoluer et vérifier la définition d'un système (ensemble organisé de matériels, logiciels, compétences humaines et processus en interaction),
- apportant une solution à un besoin opérationnel identifié conformément à des critères d'efficacité mesurables,
- qui satisfasse aux attentes et contraintes de l'ensemble de ses parties prenantes et soit acceptable pour l'environnement,
- en cherchant à équilibrer et optimiser sous tous les aspects l'économie globale de la solution sur l'ensemble du cycle de vie du système.

Pour disposer d'une définition unique tout au long de la thèse, nous retiendrons la définition suivante :

L'Ingénierie Système (ou ingénierie de systèmes) est une démarche méthodologique générale qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante aux besoins d'un client tout en satisfaisant l'ensemble des parties prenantes. [Source : AFIS]

Selon la définition de l'AFIS, l'IS est donc une démarche méthodologique. Celle-ci s'appuie sur des principes généraux, établis notamment par le retour sur l'expérience accumulée dans chaque domaine industriel et sur la mise en pratique de ces principes par l'utilisation cohérente de processus, de méthodes, et d'outils.

2.4. Principes généraux de l'ingénierie système

Au-delà des définitions présentées précédemment et selon les concepts fondamentaux reliés à la notion de système, notre travail de thèse s'appuie sur les principes de base de l'ingénierie système qui suivent. Ces principes seront mis en pratique par les éléments de la méthodologie MICCSA qui sera présentée aux chapitres 3 et 4.

Formuler le problème avant de chercher une solution. S'assurer que le problème est formulé de manière complète et correcte, au bon niveau d'abstraction et de détail, constitue un préalable indispensable à la recherche des solutions. Il s'agit d'une condition nécessaire à une exploration efficace des solutions possibles. Ce principe peut souvent être décliné simplement : *définir le quoi avant le comment*, spécifier avant de concevoir, ou encore faire précéder les descriptions *boîte noire* avant les descriptions *boîte blanche*.

Privilégier la recherche des solutions par la **définition d'alternatives de solutions, puis la sélection d'une alternative**. Cela garantit que l'exploration d'un minimum de solutions a été réalisée avant d'en sélectionner une. Toutefois, le fait de définir des alternatives ne garantit pas qu'une de ces alternatives soit optimale.

Privilégier une **démarche de conception descendante** (*Top-down*). Dans la succession des descriptions virtuelles du système, représentatives de solutions intermédiaires en terme de complétude de niveau d'abstraction et de niveau de détail, les descriptions de haut niveau doivent être définies et figées préalablement aux solutions de bas niveau. Ainsi, commencer à haut niveau d'abstraction permet de faciliter l'analyse et l'exploration du problème et des solutions.

Un autre principe consiste à diminuer la complexité inhérente au système, tant dans la formulation du problème que dans celle de la solution, en identifiant puis en **exprimant séparément les préoccupations** de chaque acteur du projet.

Assurer la cohérence entre chaque vues du système tout au long de son cycle de vie. Concrètement, cela implique de définir des référentiels et des points de vues permettant de maintenir la cohérence entre chaque représentation.

Adapter les niveaux d'abstraction parmi les différentes vues du système. Cela consiste à assurer l'interopérabilité des méthodes et des outils (entre les disciplines à un niveau d'abstraction égal et entre les niveaux d'abstraction).

Faire entrer la prise en compte des propriétés émergentes du système dans le processus de conception. De l'intégration des composants, et de leur interaction, naissent des phénomènes qu'il est essentiel d'identifier et de maîtriser. Lorsque l'architecture du système est considérée, il s'agit alors de **maîtriser les interfaces** fonctionnelles et physiques du système.

Diminuer la complexité du problème global en **procédant par étapes**, dans lesquelles l'ensemble des données projet (exigences et définition du système) sont dans un état cohérent et peuvent être considérées comme un référentiel fixe.

Vérifier la complétude et la justesse des solutions intermédiaires au plus tôt et à chacune de ces étapes, dès les phases amont d'analyse des exigences initiales afin de réduire et maîtriser l'incertitude.

Dans la suite de ce travail, nous présentons une description générale des processus développés en IS ainsi qu'une présentation des principales méthodes et outils associés. Les méthodes et outils basées sur l'utilisation de modèles seront ensuite décrits plus en détail dans le chapitre 2.

2.5. Processus et normes d'ingénierie système

La définition de cycle de vie et des processus structure les activités de l'IS. Ce sont des éléments invariants, qui permettent de s'assurer que la démarche est conduite de manière cohérente. Ce paragraphe présente une description des principaux cycles de vie applicables à l'IS et définit ce qu'est un processus.

2.5.1. Introduction : Processus et cycles de vie

Processus et cycle de vie fournissent un moyen de structuration différent des activités de développement d'un système. Un cycle de vie peut être défini comme une organisation dans le temps des états dans lequel le système (ou sa représentation virtuelle) doit se trouver au cours du développement du système. Cette organisation permet de découper le développement en phases distinctes.

Un processus fournit une abstraction des transformations du système au cours de son développement et de la consommation de ressources pour lesquelles les acteurs et les responsables sont définis mais non organisés dans le temps.

Les deux descriptions peuvent donc être considérées comme indépendantes ou orthogonales. L'AFIS propose une description de la relation existant entre processus et cycle de vie ou les phases de développement du système [AFIS] : *les processus constituent une vision opératoire du métier d'IS, complémentaire à la vision séquentielle fournie par le cycle de vie.* Par exemple les activités du processus d'intégration ne se limitent pas à la phase d'intégration. Elles commencent pendant la phase de conception avec la définition du plan et des moyens d'intégration et sont reprises pendant la phase d'exploitation lors des opérations de maintenance évolutive.

Les cycles de vie peuvent être considérés comme des ancêtres des processus de développement, mais aussi comme l'agencement des processus en séquences organisées et délimitées par des jalons.

2.5.2. Cycles de vie de l'ingénierie système

Historiquement, les cycles de vie sont particulièrement appliqués dans le développement de logiciels. De nouveaux cycles de vie sont apparus et ont suivis l'évolution continues des approches et des paradigmes du développement logiciel (approche fonctionnelle, méthodes objets, développement par aspects, méthodes agiles, …).

Les cycles de vie de développement logiciel sont pour une grande partie à l'origine de leur utilisation dans le domaine de l'IS. Trois familles de cycles de vie se sont ainsi dégagées de cette évolution : le cycle en cascade, le cycle en V et le cycle en spirale. Ces trois modèles sont largement connus et sont présentés figure 2.1. Nous les décrivons brièvement ci-dessous.

Dans le cycle en cascade, les activités sont réalisées de manière séquentielle, chaque activité étant susceptible d'occasionner un retour sur une activité précédente.

Le cycle en V est devenu le cycle de développement standard de l'IS. Le cycle est séparé en deux branches. Dans la branche descendante sont réalisés la spécification et la conception du système, les exigences spécifiées sont validées au regard du niveau précédent et les activités de vérification et de validation système sont anticipées et planifiées en termes d'attendus. Une fois la phase de réalisation des composants élémentaires réalisée, la branche de droite du cycle constitue l'intégration, la vérification et la validation du système au regard du besoin initial.

Le cycle de développement en spirale reprend les différentes étapes du cycle en V, mais permet que le cycle complet soit répétés autant de fois que nécessaire, chaque itération apportant une plus grande complétude et une meilleure robustesse des solutions apportées.

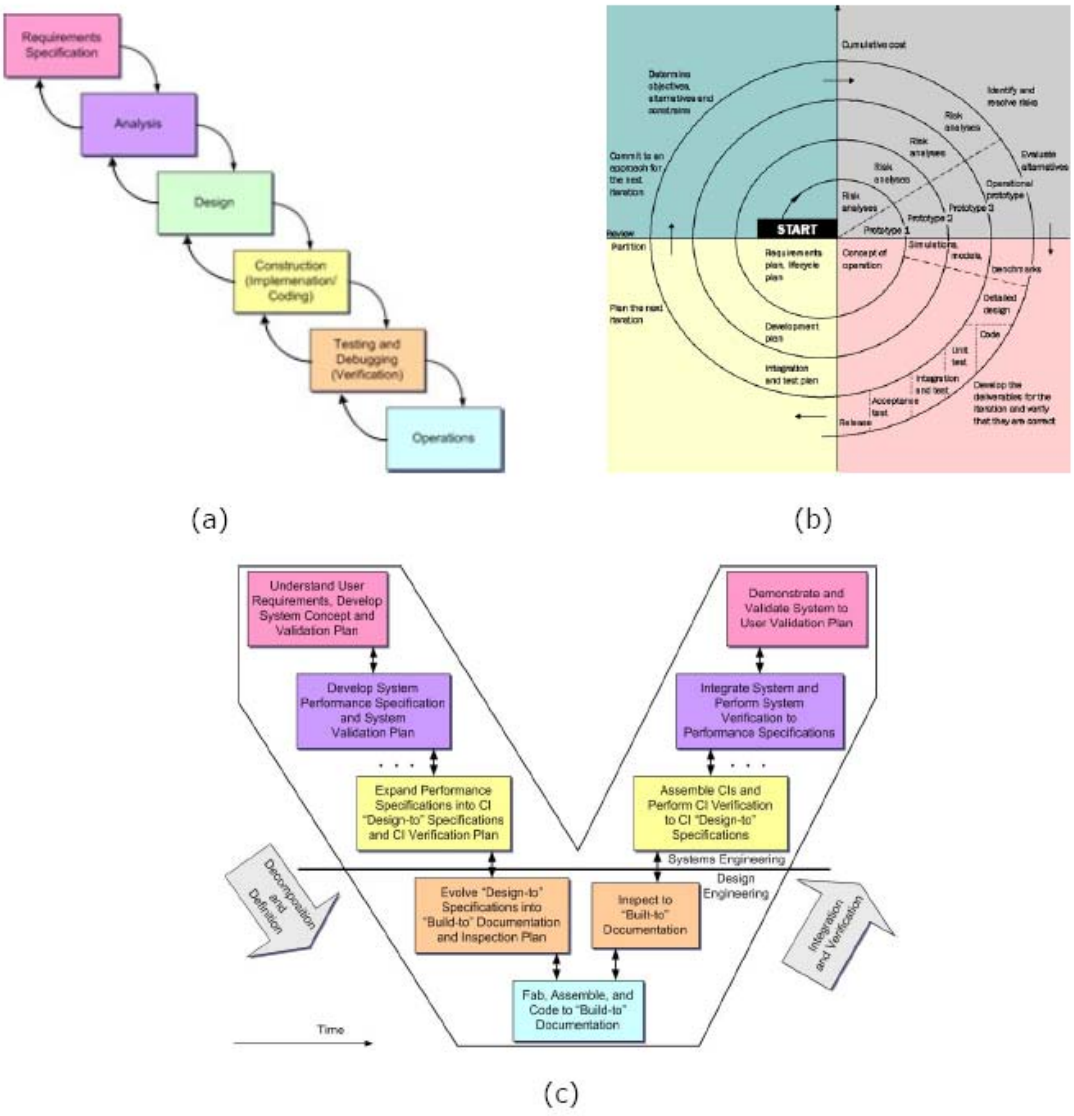


Figure 2.1: Modèles des principaux cycles de vie, [Est 07]

Autour de ces modèles se sont développées des variantes pour répondre à des spécificités d'un domaine technique, à des contraintes d'organisations ou à des outils particuliers.

2.5.3. Définition d'un processus

Selon la définition de l'AFIS, un processus est *un ensemble d'activités qui transforment un ensemble d'éléments d'entrée en éléments de sortie*. Un processus précise donc le résultat attendu ainsi que les ressources disponibles et les contraintes appliquées à cette transformation. Par contre, un processus ne précise en rien comment cette transformation est réalisée et ne fournit qu'une vision opératoire et non séquentielle. Il constitue la partie invariante de l'enchaînement des activités d'un projet à un autre. Selon le niveau de détail, ces processus peuvent être appliqués indifféremment d'un domaine technique à un autre.

Par ailleurs, de ce modèle générique de processus, l'AFIS distingue quatre groupes de processus à réaliser au cours d'un projet :

- **Les processus techniques** structurent les activités techniques autour du produit à concevoir et permettent de transformer, dans le contexte spécifique au projet, le besoin du client en solution, et de veiller à son utilisation opérationnelle. Les processus techniques peuvent être organisés selon un cycle de vie en V, en spirale, ou autre.

- **Les processus de management** permettent la gestion des activités techniques dans le contexte spécifique au projet.

- **Les processus de gestion des relations contractuelles** entre maître d'ouvrage et maître d'œuvre dans le contexte spécifique au projet.

- **Les processus d'entreprise** qui permettent de gérer de manière transversale aux projets, les domaines communs qui font appel à l'IS au sein de l'entreprise.

Par la définition de processus articulés entre eux, les normes d'IS visent à :

- la maîtrise de la complexité inhérente au projet et au produit,
- la maîtrise des risques sur les projets et les produits,
- la maîtrise des coûts, des délais, et la gestion des incertitudes,
- la communication transdisciplinaire et la réalisation de compromis globaux optimisés,
- l'amélioration globale de l'adéquation du système au besoin des parties prenantes.

Ces objectifs sont communs aux trois principales normes actuelles de l'IS : l'IEEE 1220 [IEE_98], l'EIA 632 [EIA_98], et l'ISO 15288 [ISO_08]. Celles-ci couvrent différemment les groupes de processus présentés précédemment, et avec un niveau de détail assez différent :

- description des méthodes à appliquer pour réaliser ces processus (cas de l'IEEE1220),
- recommandations générales et pratiques recommandées (cas de l'EIA632),
- spécification des résultats attendus (cas de l'ISO15288).

Ces normes adoptent donc des niveaux de détails différents dans la description de ces processus (axe vertical des ellipses, voir figure 1.2). En contrepartie, le niveau de couverture du cycle de vie du système produit est inversement lié au niveau de détail, comme le reflète la dimension verticale des ellipses correspondant à chaque norme de la figure 2.2.

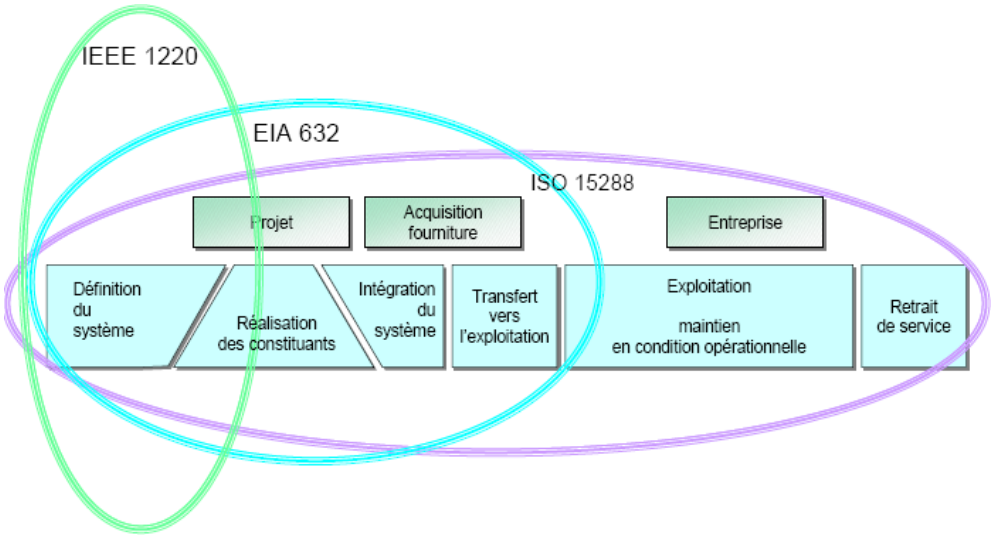


Figure 2.2: Scope et niveau de détail des principales normes de l'IS [AFIS]

Dans cette partie, les normes ISO 15288 et IEEE 1220 seront présentées de manière générale. La norme EIA 632 sera présentée de manière plus détaillée car ses processus structurent la méthodologie MICCSA, objet des chapitres 3&4.

2.5.4. Normes d'ingénierie système

Cette section présente les principales normes d'IS en application aujourd'hui. Ces principales normes sont comparables, de par leur contenu technique, leur couverture des différents aspects d'IS traités, le type de système auxquelles elles sont destinées. Elles décrivent des processus généraux, indépendants du domaine technique d'application et du niveau de complexité du (des) système(s) étudié(s).

La mise en œuvre, partielle ou complète, des processus décrits dans ces normes est conditionnée par de multiples contraintes, dont les suivantes :

- L'adaptation aux lois nationales et internationales en vigueur,
- L'adaptation à la politique et aux objectifs propres de l'entreprise : la culture et l'expérience guident la définition et l'évolution des processus internes, puis le choix de ses méthodes et outils à mettre en œuvre.
- Le domaine de l'entreprise : il peut imposer des standards d'IS spécifiques qui spécialisent, partiellement ou intégralement, les processus de l'EIA 632 (exemple ECSS-E10 pour le domaine spatial).
- Les exigences processus imposés par les clients (c.à.d. les normes et standards de l'avionneur).
- Les domaines techniques (métiers) des produits : ces processus et méthodes sont à mettre en œuvre et à interfacer aux processus et méthodes métiers relatifs aux solutions implémentées : électroniques (DO-254), logicielles (DO-178).

Tandis que la norme IEEE 1220 s'adresse en priorité au système produit matériel (la notion de système dans l'IEEE 1220 étant peu formalisée au moment de sa création), les normes EIA 632 et ISO 15288 adoptent une définition du système avec un sens plus large.

Pour ces deux normes, un système peut ainsi être constitué de produits matériels, logiciels, de services, de personnes, de techniques, d'installations, de procédures, ou d'un ensemble composé de ceux-ci, et peut être commercial ou non, simple ou complexe, innovant ou préexistant.

Les concepts et les techniques présentées dans chacune de ces normes sont souvent comparables, parfois équivalent. Au-delà du rapprochement entre normes d'ingénierie et normes d'ingénierie logicielle, un effort d'harmonisation est en cours, principalement entre les normes IEEE 1220 et ISO 15288.

Par ailleurs, un modèle de maturité, EIA 731, permet aujourd'hui d'évaluer la conformité d'un ensemble de processus implémentés aux processus de référence prescrits dans l'EIA 632. La figure 2.3 retrace l'évolution des différentes normes de l'IS.

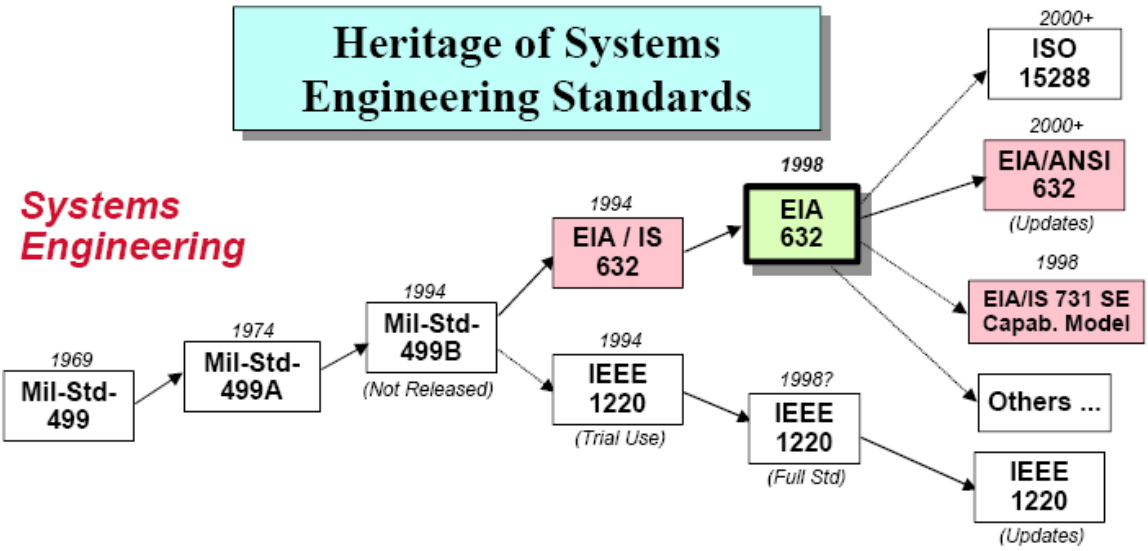


Figure 2.3. : Evolution des principales normes de l'IS

2.5.5. La norme EIA 632

La norme EIA 632 est utilisée comme une référence pour la méthodologie présentée dans ce document. Ce paragraphe inclut une description générale de cette norme d'IS et en présente les principaux aspects / concepts.

2.5.5.1.Présentation générale

En Juin 1994, une première version provisoire de la norme pour l'ingénierie des systèmes EIA 632 a été développée conjointement par l'EIA et l'INCOSE. Cette version provisoire, *l'EINIS 632 Interim Standard*, était destinée autant aux entreprises privées qu'aux organismes nationaux ainsi qu'à leurs sous-traitants.

En 1995, l'EIA soutenu par l'INCOSE établit un groupe de travail pour définir un standard officiel de la norme EIA 632. La norme provision a alors été retravaillée dans le sens d'un plus haut niveau d'abstraction afin de la rendre plus générique et applicable à divers domaines industriels et technologiques. Une version préliminaire a été diffusée en 1998. Elle a été depuis lors à l'origine de la norme l'EIA 731, permettant le déploiement des processus de l'EIA 632 au sein d'une organisation.

La norme EIA 632 est applicable à tout type de système. Ses processus sont applicables pour chaque phase du cycle de développement d'un système comme le montre la figure 2.4.

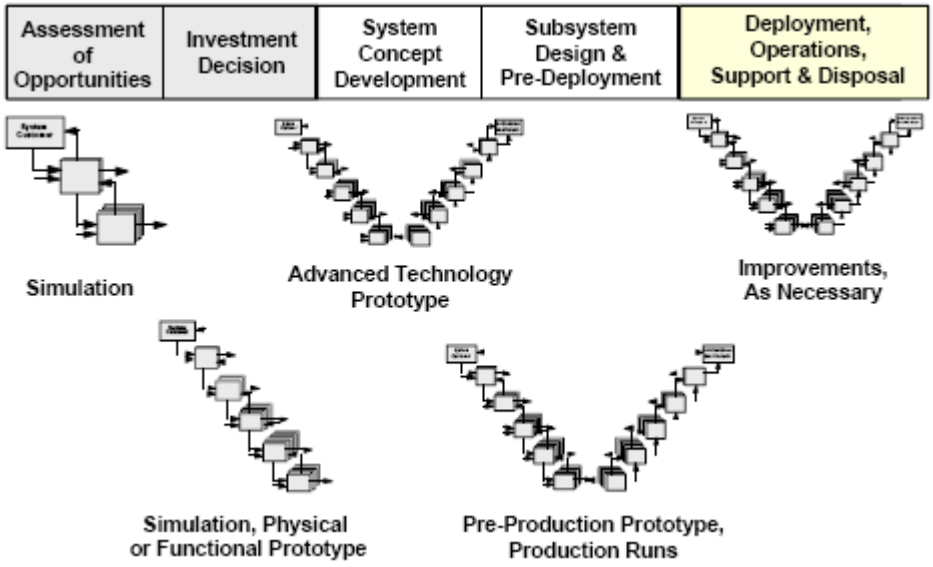


Figure 2.4. : Domaines couverts par la norme EIA632

Son niveau de généricité, les concepts et les processus qu'elle décrit la rendent bien adaptées à notre application.

2.5.5.2.Principaux concepts de la norme EIA 632

Notion de système pour l'EIA 632

L'EIA 632 définit un système comme l'agrégation d'un ensemble de produits, un produit étant lui-même défini comme tout ensemble constitué de composants (matériels et/ou logiciels) de données, de services, de personnes, de fournitures, d'installation technique, de techniques spécifiques et de documentation.

De plus, la définition distingue, à l'intérieur du même concept de système, deux types de produits (voir figure 2.5) :

- Les produits finaux sont livrés au client, et réalisent les fonctions opérationnelles du système.
- Les produits contributeurs constituent l'ensemble des produits qui ne réalisent pas les fonctions opérationnelles du système, mais qui permettent, au cours d'une ou plusieurs phases du cycle de vie du système, de réaliser celui-ci.

Cette distinction permet d'éviter que l'effort technique ne soit trop centré sur le produit final, au détriment des moyens à mettre en œuvre autour de ce produit final.

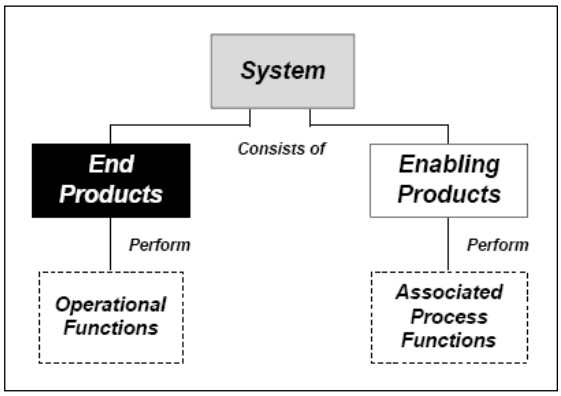


Figure 2.5. : Le concept de système dans la norme EIA 632.

Les produits contributeurs incluent les processus d'IS associés par exemple au développement, aux tests, à la production, au déploiement, au support et au retrait de service des produits finaux (voir figure 2.6). Produits finaux et contributeurs permettent au système de réaliser son objectif.

Notion de bloc de construction de l'EIA 632 ou Building block

L'ensemble de la norme s'appuie sur le concept de *building block*, ou **bloc de construction** comme élément de base. Un bloc de construction, tel que représenté sur la figure 1.7, peut être associé à tout système ainsi qu'à ses produits constituants, finaux ou contributeurs. Un bloc de construction, associé à un produit, peut par conséquent être considéré lui-même comme un système possédant ses propres produits finaux et contributeurs (figure 1.8).

Le bloc de construction possède un rôle central dans la norme EIA 632. Il constitue la *brique élémentaire* permettant de structurer l'organisation du système, de contrôler le flot des exigences produit et processus associées, de définir et d'utiliser des représentations du système adaptées aux niveaux d'abstraction et de détails.

La norme EIA632 utilise donc le bloc de construction comme un cadre conceptuel, manipulé par l'ensemble de ses processus. Il permet d'organiser les principaux apports de la norme en termes d'ingénierie des exigences, de définition des solutions, d'activités d'intégration, de validation et de vérification.

Le système peut alors être vu comme une succession de niveaux de blocs de construction. Les solutions définies dans les blocs de niveaux supérieurs, décrits par un ensemble d'exigences spécifiées, sont allouées comme exigences d'entrée aux blocs du niveau inférieur. Finalement, la décomposition d'un bloc s'arrête lorsque celui-ci peut être réalisé par un composant sur étagère, être fabriqué ou être transmis en tant qu'élément spécifié à un sous-traitant.

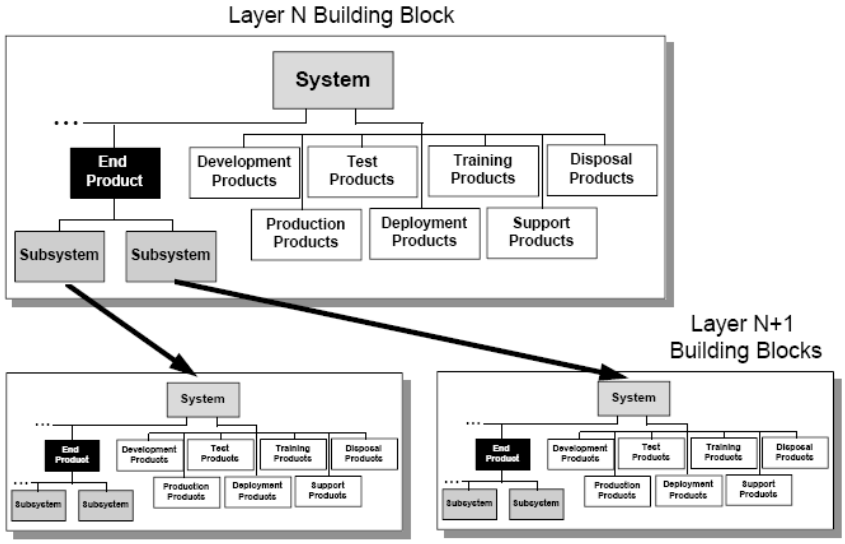


Figure 2.6. : Niveaux de définition des blocs de construction

Bien que cela ne soit pas explicité dans la norme, celle-ci prescrit ainsi une démarche de conception globalement descendante, partant des exigences du système global, pour obtenir par la décomposition en blocs de construction, les spécifications (ou exigences spécifiées) des blocs de constructions élémentaires.

Tous les blocs possèdent une structure descriptive fixe, représentée graphiquement sur la figure 1.7. Les données techniques en entrée d'un bloc sont les exigences allouées depuis le niveau supérieur, et les exigences additionnelles. Celles-ci correspondent aux exigences provenant d'autres parties prenantes (normes applicables au niveau de détail et au domaine concerné par le *building block*) telles que les exigences clients affectées au bloc. Ces exigences des différentes parties prenantes constituent, ensemble, le problème qui doit être résolu dans le périmètre du building block.

L'analyse de ces différentes sources d'exigences permet la définition puis la validation d'un ensemble d'exigences, établies comme les exigences techniques du système. Cet ensemble d'exigences constitue un référentiel à partir duquel la recherche, la comparaison et la définition de solutions techniques sont rendues possibles.

La norme préconise, en particulier, l'usage et la distinction de deux types de solutions techniques :

- Des solutions logiques, tout d'abord, qui représentent une abstraction des solutions proposées, indépendantes des technologies choisies. Les exigences techniques systèmes doivent autant que possible être allouées aux solutions logiques.
- Des solutions physiques, ensuite, décrivant les solutions techniques envisagées. Les solutions envisagées constituent, un fois sélectionnées, une solution de conception intermédiaire qui servira de base à la vérification.

A partir des solutions logiques et physiques, un ensemble validé d'exigences dérivées techniques est défini. Ces exigences sont liées aux choix de conception associés aux solutions logiques et physiques. Lorsqu'elles sont dérivées des solutions logiques, elles sont utilisées

comme aide pour définir et/ou contraindre les solutions physiques. Ces exigences doivent être affectées aux solutions logiques et physiques au même titre que les exigences techniques systèmes du bloc.

Enfin, une solution physique de conception est choisie. A partir des solutions logiques et physiques correspondantes, des exigences sont spécifiées, puis affectées aux blocs de construction du niveau suivant.

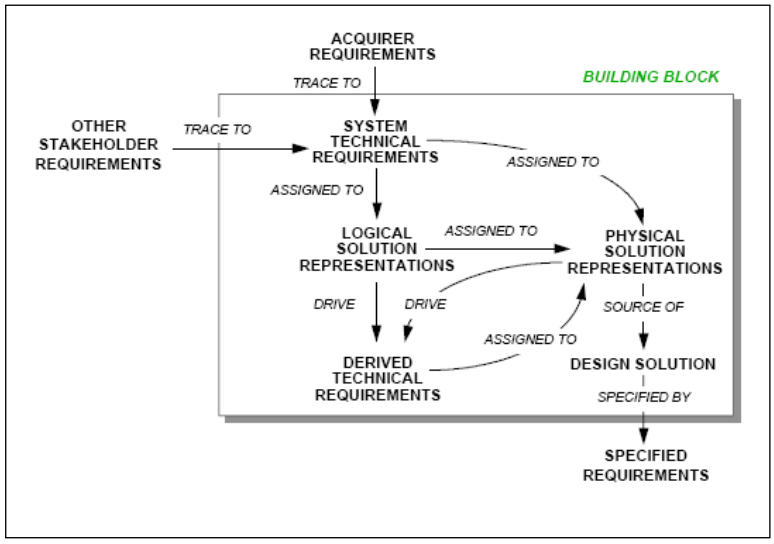


Figure 2.7. : Aspects internes du Building block ou bloc de construction de l'EIA 632

Une fois les spécifications des blocs de constructions élémentaires obtenues, l'intégration et la vérification se fait de manière ascendante, niveau par niveau en vérifiant chaque bloc de construction assemblé.

2.5.5.3. Processus de la norme EIA 632

L'EIA 632 est organisée autour de 13 processus, rassemblés en 5 groupes, représentés sur la figure 1.8.

Les différents types de processus présentés au paragraphe 1.5.3 sont présents dans la norme dans laquelle :

- les Processus techniques génériques sont assurés par les processus de conception système, de réalisation produit et d'évaluation technique,
- le processus générique de management est le processus de management technique,
- le processus générique de gestion des relations contractuelles correspond au processus d'acquisition et de fourniture.

La norme EIA 632 décrit, par l'intermédiaire d'un ensemble d'exigences sur ces processus, comment ceux-ci doivent être appliqués sur chaque bloc de construction constituant le système. Ces exigences précisent quelles sont les tâches à accomplir, quels résultats doivent être fournis et préconisent, à titre indicatif, les méthodes permettant de réaliser ces tâches.

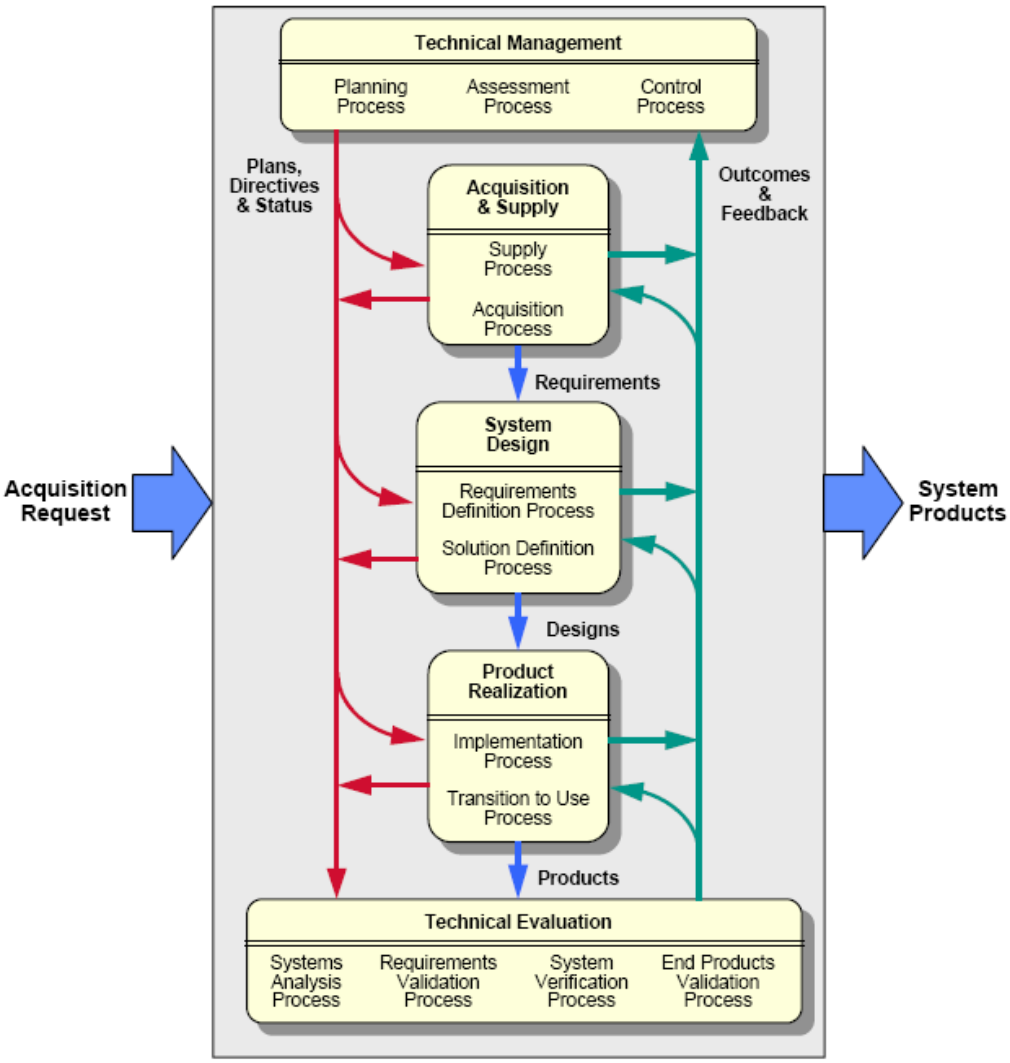


Figure 2.8. : Processus de la norme EIA 632

2.5.5.4. Les processus de conception système

Les processus de conception système constituent le nœud central de la norme. Ils permettent de transformer les exigences contractuelles de l'acquéreur et des autres parties prenantes en un produit spécifié qui satisfait ces exigences.

Dans cet objectif, deux processus sont réalisés itérativement : le processus de définition des exigences et le processus de définition des solutions. Ceux-ci sont répétés à chaque niveau de conception durant la démarche descendante tant que les blocs de constructions considérés doivent être décomposés.

Le processus de définition des exigences vise à capturer ou identifier les exigences de l'acquéreur et des autres parties prenantes et à les convertir en exigences techniques système valides.

Le processus de définition de la solution permet, quant à lui, l'obtention d'une solution de conception qui répond aux exigences techniques système, c'est-à-dire la solution obtenue par l'application du processus de définition des exigences. Les solutions logiques et physiques induisent à leur tour des exigences techniques dérivées.

2.5.6. Les normes IEEE 1220 et ISO 15288

2.5.6.1.Norme IEEE 1220

La norme IEEE 1220 décrit des processus techniques destinés à la conception des systèmes-produits. Elle est donc, de part le type de système visé et son champ d'application, beaucoup plus restrictive que la norme EIA632 précédemment décrite. Basée sur le standard militaire MIL STD 499B, la première version de l'IEEE1220 a été diffusée en 1994.

Dans cette norme, le développement du système est séparé en trois niveaux. Tout d'abord, le niveau d'analyse, de définition et de validation des exigences. Ensuite, vient une phase d'analyse, de définition et de validation des fonctions. Enfin, le troisième niveau concerne la définition d'une architecture physique issue de l'activité de synthèse.

La norme est construite autour de 8 processus organisés selon la figure 2.9. Les processus d'analyses des exigences et d'analyse fonctionnelle permettent de définir une architecture système validée, allouée sur des composants physiques, avant l'étape de synthèse permettant d'obtenir l'architecture physique à construire.

Un processus spécifique d'analyse système permet une évaluation des solutions envisagées (alternatives) et la résolution de conflits ou la définition de compromis à chaque niveau d'abstraction : exigences, fonctions et solutions physiques. Ce processus permet de systématiser la technique employée pour la prise des décisions de conception.

Enfin, un processus de maîtrise est dédié au contrôle des activités d'IS décrites dans les autres processus.

Par ailleurs, des activités de validation et de vérification permettent de définir, niveau par niveau, un référentiel d'exigences, d'architecture fonctionnelle et physique, vérifié du projet. La norme suit, par conséquent, une démarche descendante et n'explicite pas les activités d'analyse système dans le cas de système partiellement ou intégralement préexistants.

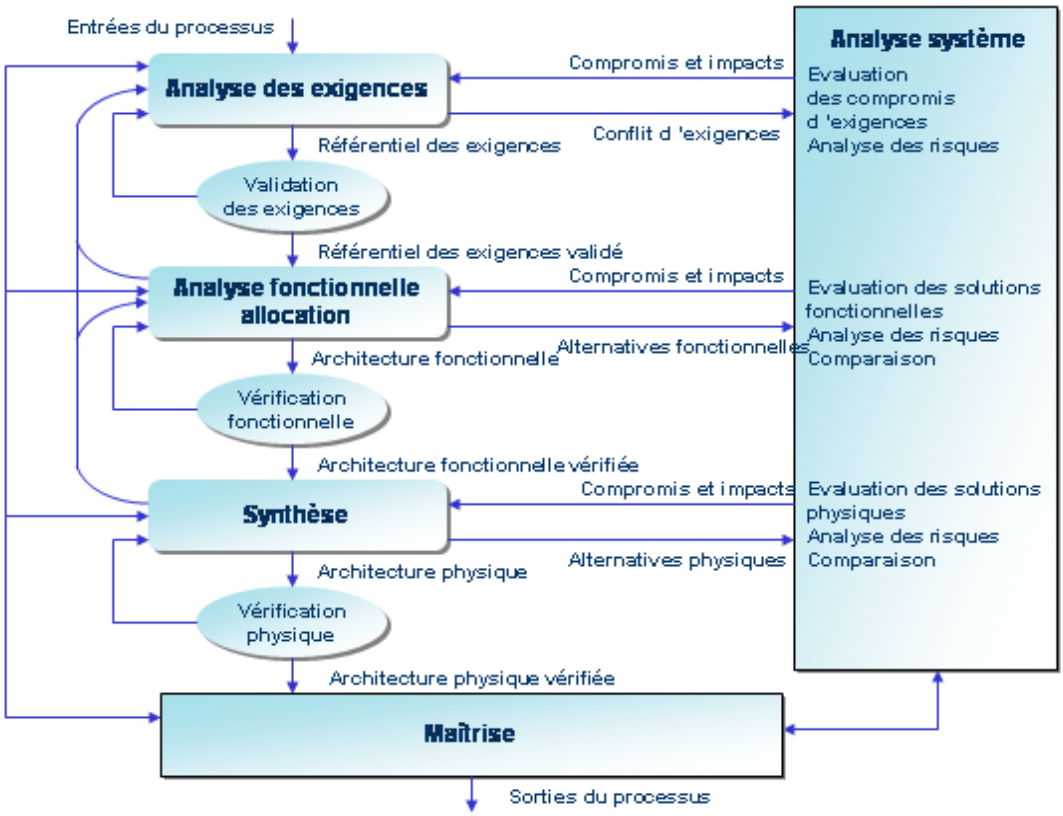


Figure 2.9. : Les processus de l'IEEE 1220 (selon l'AFIS)

2.5.6.2.Norme ISO15288

La norme ISO 15288 a été développée depuis 1990 en reprenant la structure de la norme ISO/CEI 12207 – AFNOR Z 67- 150 destinée au développement de logiciel (source AFIS, voir figure 1.10). Elle définit un standard applicable au développement de systèmes de nature et de taille quelconque et sans a priori sur le partitionnement entre logiciel et matériel. Elle est donc proche de l'EIA 632 dans sa définition du système, comme étant composé du système-produit final accompagné de l'ensemble de ses produits contributeurs (y compris les processus associés). Le standard prescrit les cinq types de processus principaux identifiés précédemment et couvre l'intégralité du cycle de vie du système.

Elle complète les processus s'appliquant aux projets par des processus dits d'entreprise, qui ont pour objectif de déployer au mieux le potentiel de l'IS au sein de l'entreprise, en gérant les domaines communs au profit des projets d'IS.

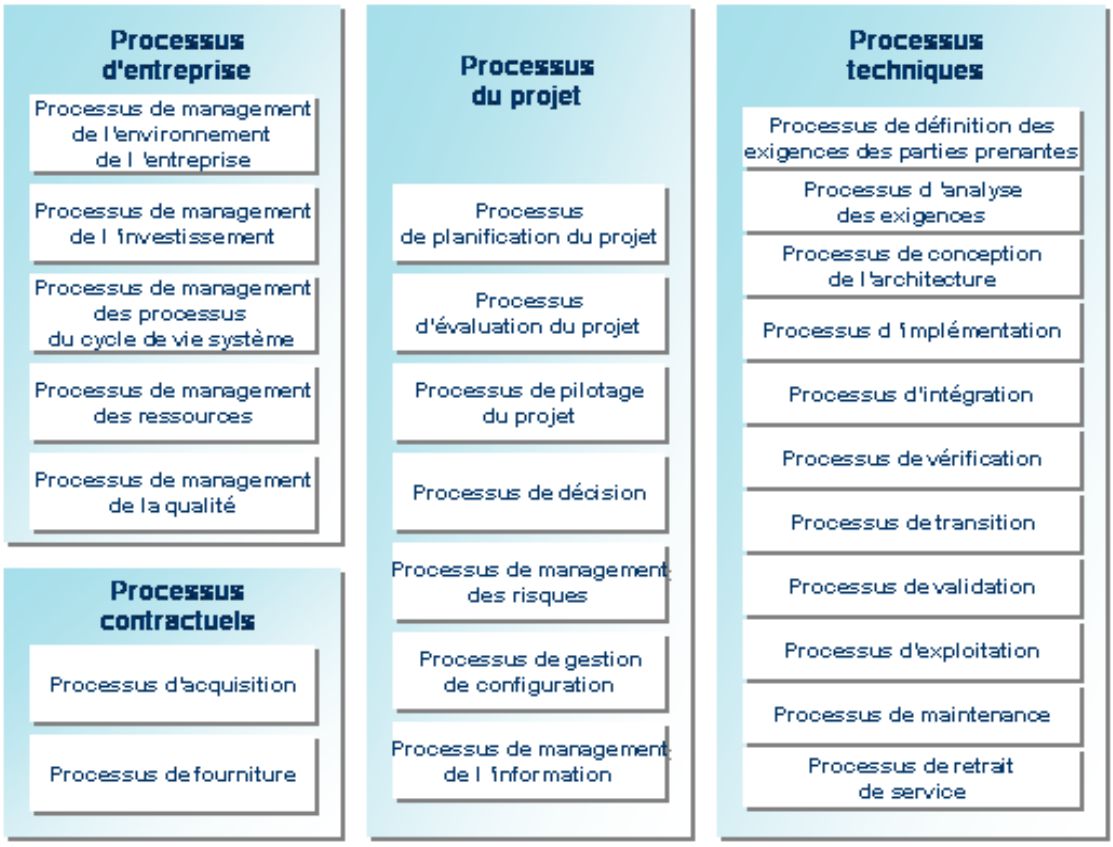


Figure 2.10. : Les processus de l'ISO 15288 (selon l'AFIS)

2.6. Conclusion

Ce premier chapitre a tout d'abord permis d'établir les définitions fondamentales et nécessaires à la suite de ce document. Il a également permis de justifier l'existence de l'IS en tant que discipline, à la fois comme une réponse au besoin général de maîtrise de la complexité inhérente au développement des systèmes, mais également pour répondre à un besoin plus spécifique de l'entreprise LATECOERE, par l'intermédiaire d'une étude pratique au sein de métiers techniques de l'entreprise.

L'étude des principales normes existantes dans le domaine a enfin permis d'aborder et de poser les thèmes fondamentaux de la thèse que sont les processus techniques, et les pratiques générales liés au développement et à la gestion des exigences, des spécification et des représentations de solutions techniques.

Les normes d'IS présentés dans ce chapitre permettent de guider l'entreprise dans la définition de ses processus et dans l'instanciation de ceux-ci au cours de ses projets. Toutefois, elles ne spécifient pas comment les processus peuvent être réalisés, c.à.d. les méthodes et outils qu'il est nécessaire de mettre en œuvre pour mener à bien les processus. Le chapitre suivant propose de répondre à cette limite et restreint le domaine d'étude à une certaine catégorie de méthodes, appelées méthodes d'IS basées sur les modèles.

Chapitre 3. L'Ingénierie Système basée sur les modèles

Ce chapitre commence par un état de l'art de l'utilisation des modèles dans le cadre de l'ingénierie système. Le premier sous-chapitre permet de comprendre ce que l'utilisation de modèles et les activités liées à ces modèles peuvent apporter dans le domaine de l'ingénierie des systèmes pour la description d'un produit.

Le langage de modélisation SysML est ensuite présenté. Ce langage permet d'exprimer, dans une approche basée sur les modèles, les principes de l'ingénierie système. Ceci nous amènera à identifier les piliers d'une modélisation système complète et cohérente et à situer les différents langages de modélisation par rapport à cette référence.

Les langages présentés sont mis en œuvre conformément à des méthodes spécifiques, appelées méthodes d'ingénierie système basées sur les modèles. Les principales méthodes utilisées en contexte industriel seront présentées, ce qui permettra d'en définir les principes communs ainsi que leurs limites.

Enfin, un état de l'art présentera les directions actuellement envisagées afin d'amélioration du bénéfice apporté par ces méthodes dans la réalisation des processus d'IS en général et sur certains aspects spécifiques en particuliers, tels que la recherche de composants en phase de conception architecturale et la validation des exigences système.

3.1. Modèles pour l'ingénierie système

Dans son sens techniques et scientifique le terme *modèle* a été employé initialement dans les domaines de la cybernétique et de la systémique à partir du milieu du XXe siècle. Un modèle est la représentation d'un système conforme à la fois à la syntaxe concrète et à la syntaxe abstraite d'un formalisme de modélisation. Ce formalisme peut être qualifié de langage.

La syntaxe concrète constitue la notation graphique ou le format avec lequel chaque élément du formalisme peut être exprimé (visuellement sur un écran, ou sauvegardé dans un fichier informatique par exemple), tandis que la syntaxe abstraite correspond à la signification ou la sémantique associée à ces éléments. La syntaxe abstraite d'un formalisme est donc représentative des concepts qui peuvent être modélisés à l'aide celui-ci.

Lorsqu'un modèle est explicatif, il constitue un moyen efficace pour simplifier, analyser, formaliser ou communiquer par une représentation abstraite les caractéristiques d'un objet ou d'un phénomène observé ou attendu dans le monde réel. Si l'objet modélisé est préexistant à sa modélisation, le modèle est dit descriptif. Dans le cas contraire, le modèle est généralement

qualifié de modèle prédictif, ou de modèle exécutable si celui-ci est utilisé par un moteur d'exécution (ex : moteur de simulation) pour analyser la dynamique d'un système et sa réponse aux stimuli que lui fournit son environnement.

Face à la nécessité de développer des systèmes de plus en plus complexes soumis à des contraintes croissantes, les industries cherchent à adapter leurs pratiques de l'IS vers l'utilisation de modèles et des techniques qui leurs sont associées.

Ces modèles constituent un outil :

- Pour la communication entre les divers acteurs industriels de l'entreprise (métiers et services), mais aussi dans le réseau complexe d'acteurs industriels impliqués aux différents stades d'avancement d'un projet.

- Pour la maîtrise de la complexité, formuler, communiquer et valider le domaine du problème et celui de la solution, depuis les éléments préliminaires et abstraits jusqu'à une représentation détaillé d'un produit ou d'un processus.

Dans le contexte de la conception de systèmes, Loyd Baker identifie plusieurs rôles fondamentaux à l'utilisation de modèle par l'ingénieur système [bak_97]. Des *modèles schématiques* permettent de représenter différentes structures d'organisation d'un système, alors que des *modèles de performances* décrivent une structure exécutable permettant d'évaluer la réponse du système aux stimuli de son environnement. Des *modèles de conception* permettent de décrire la conception détaillée d'un système et, des *modèles physiques* permettent des expérimentations et des démonstrations réalistes du système étudié. Ainsi, ce premier découpage s'appuie sur des critères empiriques divers tels que la fonction ou le rôle attribué aux modèles, leur capacité à être exécutés ou leur nature physique ou virtuelle.

La distinction entre modèle descriptifs et modèles prescriptifs constitue une classification plus rigoureuse [Béz_05]. Les premiers sont construits à partir de l'analyse d'un système déjà existant. L'objectif est d'en améliorer la compréhension ou d'extraire certaines informations qui permettront d'améliorer la conception d'un autre système. Ces modèles ont donc une finalité d'analyse et de compréhension. Dans le standard IEEE 610.3, les modèles descriptifs sont utilisés pour décrire le *comportement ou les propriétés d'un système existant ou d'un type de système*. Les modèles prescriptifs sont, quant à eux, une représentation du problème, construits à partir d'autres modèles, avec pour finalité la conception d'un nouveau système. Un modèle prescriptif constitue *un cadre conceptuel qui peut être utilisé comme une base pour la modélisation d'un système. Il constitue donc le fondement du modèle descriptif* [IEE_89].

Ces deux types de modèles résultent donc d'une phase d'analyse. Ils sont la représentation d'un système existant pour le descriptif et à concevoir pour le prescriptif. La distinction permettant cette classification repose à la fois sur la préexistence physique des éléments représentés dans les modèles et sur le rôle final du modèle [Béz_05].

L'AFIS accorde également un rôle majeur à l'élaboration et à l'utilisation des modèles dans le domaine plus large de l'IS. L'AFIS explique qu'*il est nécessaire de s'appuyer sur des représentations tant du problème que de ses solutions possibles à différents niveaux d'abstraction pour appréhender, conceptualiser, concevoir, estimer, simuler, valider et*

justifier des choix. Elle propose une typologie des modèles d'IS, apportant ainsi une clarification sur l'objectif de modélisation et d'utilisation des modèles. Cette typologie qui peut être qualifiée de typologie fonctionnelle est présentée ci-après.

3.1.1. Différents rôles de modèles d'ingénierie systèmes

L'AFIS distingue dans sa typologie trois types principaux de modèles. Ces modèles sont distingués selon le rôle qu'ils jouent dans la conduite des activités d'IS.

D'une part, les ***modèles cognitifs*** sont utilisés pour l'analyse et l'exploration du problème ou du besoin à l'origine du système et pour la validation des concepts opérationnels de la solution apportée. Les activités principales pour le développement de tels modèles concernent particulièrement l'identification du système d'intérêt (périmètre du système et des interactions avec son environnement), l'analyse du comportement des éléments d'environnement, des éléments constitutifs de la mission et des scenarios opérationnels. Leur rôle est donc de participer et de faciliter l'analyse du système à haut niveau.

Des ***modèles normatifs*** participent aux processus de définition des exigences et de définition de la solution au niveau de la conception architecturale du système. En d'autres termes, ils ont un rôle dans la formalisation du besoin (modèles normatifs de types prescriptifs) et des solutions apportées (modèle normatifs de types constructifs). Les modèles normatifs sont donc décomposés en modèles ***prescriptifs*** représentant un ensemble de caractéristiques ou contraintes qui doivent être satisfaites et en modèles ***constructifs*** représentatifs des solutions apportées ou envisagées.

Enfin, les ***modèles prédictifs*** ont pour rôle d'anticiper et de valider le comportement du système vis-à-vis de ses exigences. Ces modèles prédictifs sont eux-mêmes décomposés en modèles ***formels*** et en modèles ***analytiques*** selon les techniques mises en œuvre et les mécanismes de preuves associées.

A ces rôles, l'AFIS associe un ensemble de rôles qui pourraient être qualifiés de rôles transverses, car ils sont communs à chacun des types mentionnés ci-dessus. Ainsi, le rôle de partage de la connaissance sur le système a été identifié. Elle permet d'appréhender les différents aspects du système ou de se former à son utilisation. Le rôle de support pour la capitalisation est aussi un apport important des modèles : un modèle peut facilement être conservé et réutilisé, et les voies étudiées ou envisagées lors de l'exploration des solutions de conception peuvent par exemple être capitalisées. Par ailleurs, la capacité des modèles à exprimer différents niveaux d'abstraction permet de leur donner une certaine indépendance par rapport aux contextes d'utilisation et aux technologies qui permette de les réaliser et de les utiliser. Cette classification des modèles par l'AFIS est résumée par la figure 3.1.

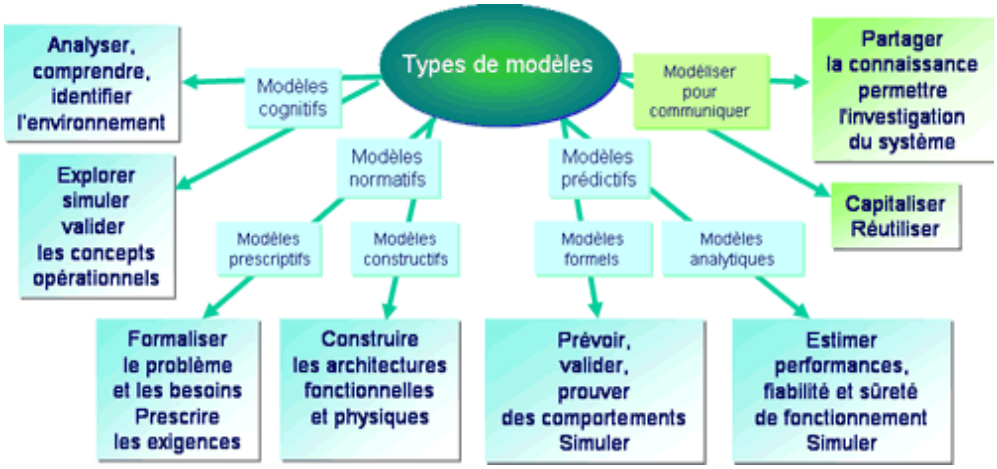


Figure 3.1 Typologie fonctionnelle des modèles IS proposée par l'AFIS

Plusieurs observations peuvent être formulées concernant cette typologie. Tout d'abord, les modèles étudiés sont limités à la modélisation du système produit et écarte la modélisation des processus utilisée en IS. Par ailleurs, le choix de limiter les modèles formels à un rôle de prédiction de comportement peut paraitre arbitraire, car de nombreux langages de modélisation formels (ex : langage B) participent aux tâches de formalisation du problème et des besoins (modèle prescriptif), ainsi qu'aux tâches de validation de concepts opérationnels ou de construction d'architectures physiques (modèle constructif). Enfin, la typologie pourrait être plutôt considérée comme une distinction des rôles des modèles pour l'IS, plutôt qu'une typologie au sens strict du terme, laissant penser qu'un modèle ne peut entrer que dans une seule et unique catégorie identifiée.

Un dernier point, absent de cette typologie, pourrait être la caractérisation des rôles communs ou *transverses* des modèles pour l'IS. Dans quelle mesure peuvent-ils supporter les processus de l'IS ? De par leurs autres rôles identifiés et les informations produites et utilisées pendant les processus d'IS, les modèles d'IS apportent fréquemment une redondance d'informations par rapport aux documents constituants l'ossature des processus traditionnels. La question se pose alors de considérer les modèles d'IS comme une alternative possible aux documents. Ce point sera traité plus en détail dans le sous-chapitre 3.4.

Néanmoins, cette typologie constitue une description suffisamment complète des rôles attribués aux modèles en IS pour être retenue comme une référence pour la suite de ce document. Dans ce chapitre, elle permettra de présenter et d'étudier la sémantique de plusieurs langages de modélisation destinés ou applicables à l'IS. Par ailleurs, cette typologie permettra d'organiser la méthodologie présentée au chapitre 3 autour de modèles dont les rôles au sein de la démarche sont clairement définis et formalisés.

De nombreux travaux de recherche ont porté sur la modélisation des exigences en IS. Une partie de ces travaux se base sur une expression des exigences en langage naturel, d'autres impliquent une formalisation de ces exigences. Le champ des exigences étudiées et de systèmes ciblés est généralement spécifié (exigences temps réel, exigences de qualité de service sur application logicielle, exigences de sécurité sur système avionique, etc..).

3.1.2. Modèle cognitifs

Comme décrit précédemment, les modèles cognitifs sont utilisés pour l'analyse, l'exploration du problème et la validation de concepts opérationnels. Les modèles qui ont ce rôle adoptent un point de vue extérieur au système et considèrent celui-ci comme une boîte noire en interaction avec son environnement pour réaliser sa fonction.

Un modèle applicable dans ce cadre est le **modèle de cas d'utilisation**. Le cas d'utilisation s'intègre dans une approche de modélisation destinée initialement au développement du logiciel et a été inventé par Jacobson [Jac_92]. Toutefois, la construction du modèle de cas d'utilisation peut être aisément transposée au niveau d'un système complet, produit ou service. Le modèle des cas d'utilisation permet d'établir un vue utilisateur du système étudié en représentant les fonctionnalités attendues par différentes classes d'utilisateurs nommées *acteurs* du système. Ces acteurs utilisent les fonctionnalités ou services de haut niveau fournis par le système étudié. Ces services peuvent être organisés entre eux par des liens de raffinement, d'extension et d'inclusion. Dans le cas du développement d'un système, le modèle de cas d'utilisation, s'il ne contient pas une grande quantité d'informations, apporte un bénéfice à l'étude d'un système car il amène à distinguer et à analyser chaque fonctionnalité essentielle (fonction opérationnelle, ou service) dans l'utilisation opérationnelle du système telle qu'attendue par le client et proposée par le développeur.

Les modèles de cas d'utilisation sont régulièrement associés à des modélisations d'interactions sous la forme de diagramme de séquence par exemple, dont l'origine est basée sur le MSC (*Message Sequence Chart*) du langage SDL [SDL_99]. Ce formalisme est adapté pour représenter une succession d'interactions discrètes, synchrones ou asynchrones, entre le système et les éléments de son environnement. Il est particulièrement adapté pour la modélisation de séquences d'interactions dans un contexte spécifique d'opération du système étudié, d'où son utilisation intensive pour la description de scenarii d'utilisation.

D'autres modèles comparables à celui des cas d'utilisation quant aux points de vue et niveau d'abstraction, peuvent jouer le rôle de modèle cognitif. Parmi eux, le *modèle pieuvre*, extrait des approches MISME / APTE™ en est probablement le meilleur exemple. Celui-ci se fonde sur la représentation des interactions entre le système et les éléments de son environnement sous forme de fonctions principales et de fonctions de contraintes. Ce concept est illustré par la figure 3.2. Les fonctions principales apportent la valeur ajoutée du système. Elles utilisent un élément de l'environnement pour modifier, par l'action du système étudié, l'état d'un autre élément de cet environnement. Les fonctions de contraintes permettent de maîtriser l'effet de l'environnement sur le système ou l'interaction du système sur son environnement. Cette distinction entre fonctions principales et fonctions de contraintes permet d'identifier le but du système étudié par l'analyse de son action sur son environnement direct.

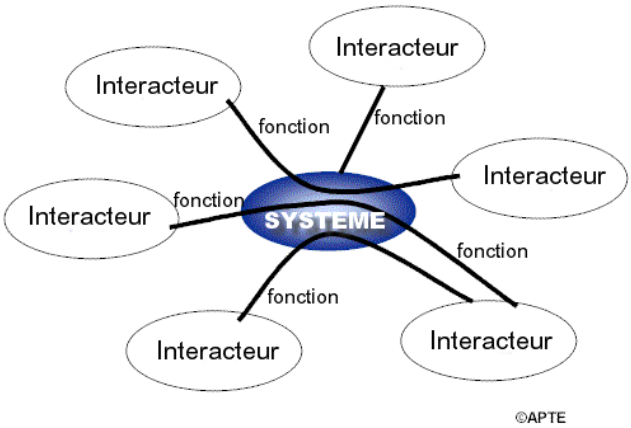


Figure 3.2. Vision des interactions avec le diagramme pieuvre

La représentation matricielle DSM (Design Structure Matrix) est aussi communément utilisée pour identifier et analyser les dépendances et interaction existantes entre les entités d'un système. Il s'appuie initialement sur la désignation des constituants ou composantes d'un système comme éléments d'entrée d'une matrice. Ce formalisme est à la base de nombreux travaux de recherche, dont une synthèse a été réalisée par Harmel et al. [Har_07]. Lorsqu'une telle matrice possède deux dimensions et que ses entrées lignes et colonnes comporte les mêmes éléments constitutifs d'une l'architecture produit, une telle matrice est appelée matrice N². Celle-ci est par exemple utilisée dans les méthodes d'ingénierie système préconisées par la FAA [FAA_ 06]. L'avantage de telles approches et de favoriser une analyse exhaustive des relations et de permettre l'usage de méthodes formalisées par des opérations sur ces matrices.

De nombreux modèles issus de logiciels de modélisation propriétaire permettent, de manière comparable au modèle des cas d'utilisation, de concentrer l'étude sur les objectifs d hauts niveaux du système étudié. Ainsi, les modèles *orientés buts* tels que les modèles *objectiver*™ permettent de représenter les objectifs de haut niveaux, puis par raffinement et décomposition, décliner ceux-ci en objectifs techniques, exploitables lors du développement technique du système et de ses composants.

Ces modèles permettent l'analyse du système et sont utilisés pour explorer, identifier et représenter les exigences fonctionnelles, de performances et les exigences non fonctionnelles du système.

3.1.3. Modèle normatifs

De nombreux modèles peuvent être considérés comme des modèles normatifs. Pour autant, aucun d'entre eux ne permet de couvrir intégralement ce qui est décrit par l'AFIS comme un modèle prescriptif (modèle support de la définition des exigences système) ou un modèle constructif (support de la construction des architectures du système). Les formalismes présentés entre donc partiellement dans l'une ou l'autre de ces catégories.

Le formalisme SADT (Structured Analysis and Design Technique) [Ross_77] permet par exemple de modéliser un système selon son organisation fonctionnelle, en descendant niveau par niveau dans la description interne fonctionnelle du système étudié. Il permet ainsi, de construire une architecture fonctionnelle du système.

De nombreux modèles et formalismes s'attachent à spécifier le comportement ou la dynamique des systèmes. La modélisation et la représentation des comportements peuvent être abordées de multiples façons, selon les hypothèses adoptées, le domaine d'application du système étudié, son niveau de complexité, etc.

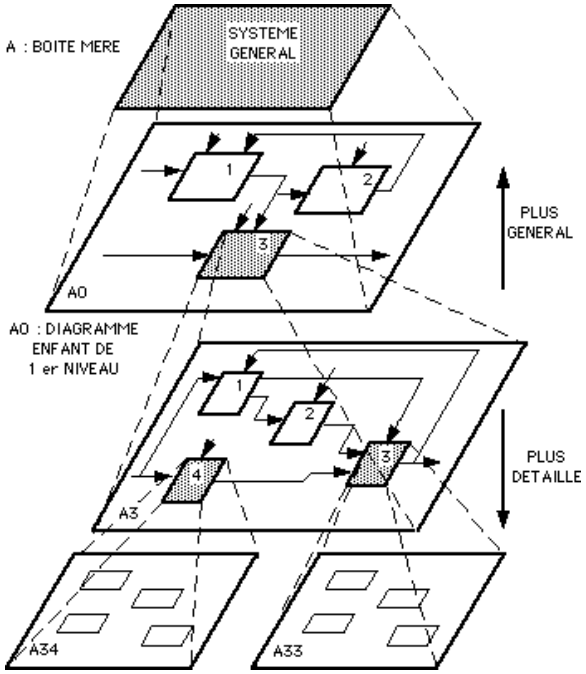


Figure 3.3. Architecture fonctionnelle SADT

Dans cet objectif, un formalisme couramment utilisé dans le domaine de l'IS est le diagramme EFFBD (Enhanced ou Extended Functionnal Flow Block Diagram). Celui-ci résulte principalement de l'ajout de flux de données au formalisme initial FFBD, permettant d'organiser des fonctions entre elles au moyen de structures de contrôle. Sur un EFFBD, les flux de données représentés peuvent être de type *trigger* ou *data store*. Dans le premier cas, une définition de la valeur de la variable déclenche l'exécution d'une fonction, à condition que celle-ci ait été activée au préalable par un flux de contrôle. Dans le deuxième cas, une donnée est simplement disponible pour être utilisée par une ou plusieurs fonctions, sans lien de causalité entre la définition de la donnée et l'exécution de ces fonctions. Une description plus complète de ce formalisme a notamment été réalisée par Seidner [Sei_06]. La figure 3.4 donne un aperçu des possibilités de modélisation par EFFBD.

La modélisation de la dynamique d'un système peut être vue comme la modélisation d'un état décrit au sens de l'évolution d'un vecteur de variables d'états continues ou discrètes. La modélisation du temps s'étale sur une échelle qui s'étend depuis la simple définition de relations de causalité entre événements discrets, comme un ensemble d'événements datés de manière relative ou absolue, jusqu'à la description de l'évolution continue dictée par des équations dépendantes du temps. Lorsqu'un modèle met en jeu une composition des catégories définies à partir de ces critères, le modèle est dit hybride ou hétérogène.

Certains de ces modèles s'attachent à représenter le comportement du système dans un contexte d'utilisation particulier, tandis que d'autres sont adaptés à une représentation du comportement global du système étudié.

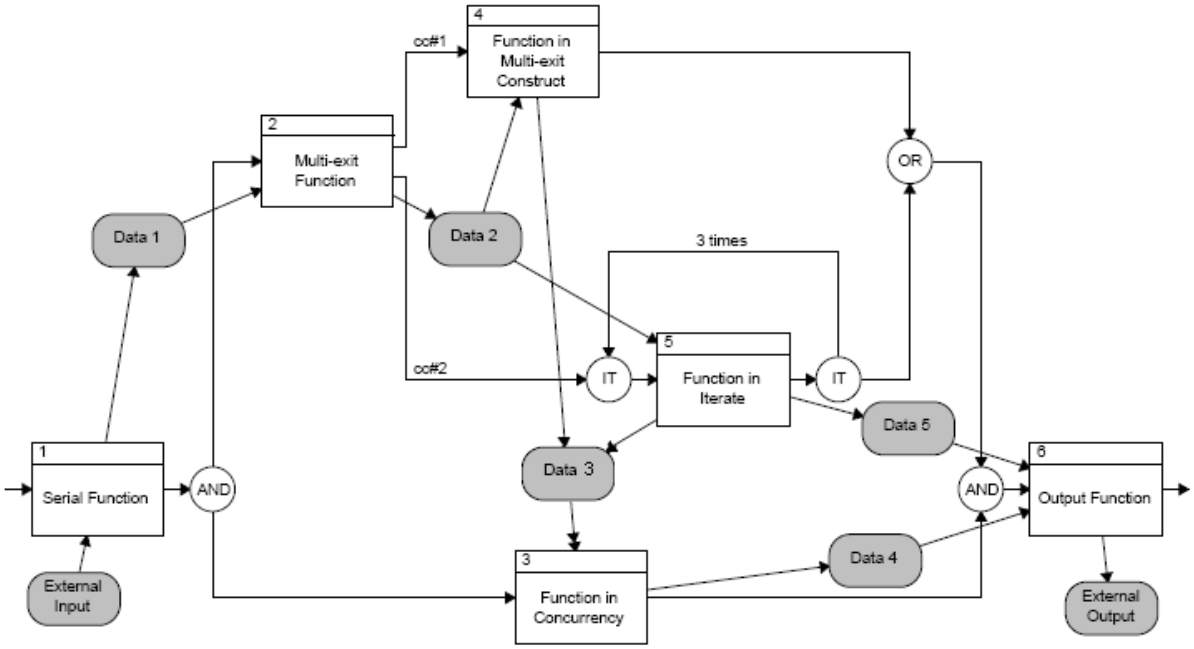


Figure 3.4. : Modèles représentés selon le formalisme EFFBD

Ainsi, le diagramme de séquence déjà abordé comme un modèle cognitif peut être utilisé comme un moyen de spécification d'exigences système, ou, à un niveau plus détaillé incluant les composants internes du système étudié, un moyen de spécification des composants du système.

Le modèle StateChart [Har_84], offre un moyen de spécifier un système et ses composants en précisant l'état (état total du système étudié ou état partiel « composite ») dans lequel celui-ci doit se trouver, au cours de son fonctionnement. L'évolution se fait par le biais de transitions auxquelles sont associées des conditions.

Le modèle Réseau de Petri [Pet_62] permet de modéliser, sur la base d'une théorie mathématique, l'évolution de l'état d'un système par le biais d'un graphe et d'un ensemble de variables discrètes associées, et d'évaluer certaines propriétés structurelles caractéristiques du réseau sur la base de cette théorie.

Tandis que ces modèles permettent de représenter l'aspect contrôle de l'évolution du système, d'autres modèles se concentrent sur l'évolution de données descriptives de son fonctionnement. De telles approches sont exploitées dans les modèles Z par exemple, et les modèles B s'attachent à montrer l'évolution des données de manière formelle.

Enfin, l'aspect structurel des systèmes est, lui aussi, fortement supporté par l'utilisation de modèles. Le modèle diagramme bloc permet ainsi de représenter des blocs en interaction. Ces blocs peuvent avoir un sens fonctionnel (dans ce cas, ils sont représentatifs de la réalisation d'une fonction, sans présager des détails de sa réalisation au sens physique) ou un sens physique. Dans les deux cas, le diagramme bloc est un modèle utilisé au quotidien permettant de décrire la structure du système étudié. Le diagramme de classe, issu du domaine du développement logiciel, est lui aussi applicable à la description d'un système.

3.1.4. Modèles prédictifs

Les Modèles prédictifs permettent d'anticiper le fonctionnement d'un système afin d'en prévoir ses caractéristiques fonctionnelles et non fonctionnelles.

Pour cela, les langages de simulation système permettent de construire des modèles exécutables permettant d'étudier l'évolution de l'état du système ou de composants du système lorsque celui-ci et soumis à des conditions initiales et des stimuli provenant de son environnement.

La modélisation de l'évolution des systèmes continus est possible à l'aide d'outil de modélisation générique des systèmes continus. Simulink permet, par exemple dans un langage propriétaire, de modéliser des systèmes multidisciplinaires.

Les langages SystemC et VHDL, même s'ils s'apparentent à des langages de description de matériel traditionnellement associés à la phase d'implémentation, peuvent être utilisés à un niveau d'abstraction plus élevé. Ils permettent ainsi de modéliser l'évolution des systèmes à événements discrets. Leurs extensions respectives, SystemC-AMS et VHDL-AMS permettent, par ailleurs, la modélisation hybride et la simulation. Le langage Modelica permet également de définir des modèles hétérogènes adaptés à la modélisation système.

Le formalisme HiLeS (High Level System Designer) permet la modélisation de systèmes hybrides par la modélisation conjointe d'une structure de contrôle sous la forme d'un réseau de Petri, et d'un ensemble de modèles VHDL-AMS associés aux composants élémentaires du système. Ce type de modèle combine les intérêts d'un modèle formel et d'un modèle analytique, au prix d'une modélisation parfois complexe. Ce formalisme a notamment été employé dans le domaine des systèmes aéronautiques [Ham_05] et des microsystèmes [Mau_05].

L'ensemble des modèles présentés ci-dessus sont donc adaptés pour répondre à une partie des préoccupations de l'IS. L'approche employée dans la spécification et le développement du langage SysML est différente, car la sémantique du langage a été conçue pour couvrir chacun des principaux aspects de l'IS. C'est la raison pour laquelle sa description fait l'objet de la section suivante.

3.2. SysML : un langage de modélisation unifié pour l'ingénierie système.

Le besoin d'un langage unifié de modélisation des systèmes, couvrant les différents aspects de l'IS a été formalisé conjointement par l'INCOSE et l'OMG en 2003 sous la forme d'un appel à proposition [SysRFP]. Une contrainte supplémentaire était d'appuyer ce langage d'IS sur le langage UML, afin qu'il bénéficie de la connaissance assez large d'UML au sein de la communauté de l'IS et de la disponibilité d'outils déjà existants.

La première version de SysML, présentée sous la forme d'une spécification de langage, a été soumise puis adoptée par l'OMG en 2005. Elle est le fruit d'un partenariat entre des industries majeures de différents domaines, notamment de l'aéronautique et de l'électronique ainsi que des développeurs d'outils de modélisation pour le développement de logiciel.

Plusieurs versions concurrentes ont été proposées et une nouvelle version enrichie a été adoptée par l'OMG en 2006. La version actuelle de SysML est la version 1.2 ; elle inclut les évolutions résultantes de ses premières utilisations concrètes.

La description du langage ci-dessous est générale. Une description plus complète est disponible dans le document de spécification émis par l'OMG [OMG_07].

Le langage SysML est structuré autour de quatre domaines de modélisation : modélisations des structures, des comportements, des exigences et modélisation paramétrique. Chacun de ces domaines, souvent qualifié de *pilier* du langage SysML possède :

- des éléments de base appelés *constructs* dont les caractéristiques internes sont spécifiées en termes d'attributs, de relations avec les autres éléments de base et de contraintes applicables, constituant ainsi la syntaxe abstraite du langage,

- un ensemble de représentations ou de notations graphique associées à ces *constructs*, ainsi qu'un ou plusieurs diagrammes (voir figure 2.5) permettant de visualiser les éléments de manière standardisées par les outils. Ceci constitue la syntaxe concrète, ou notation SysML.

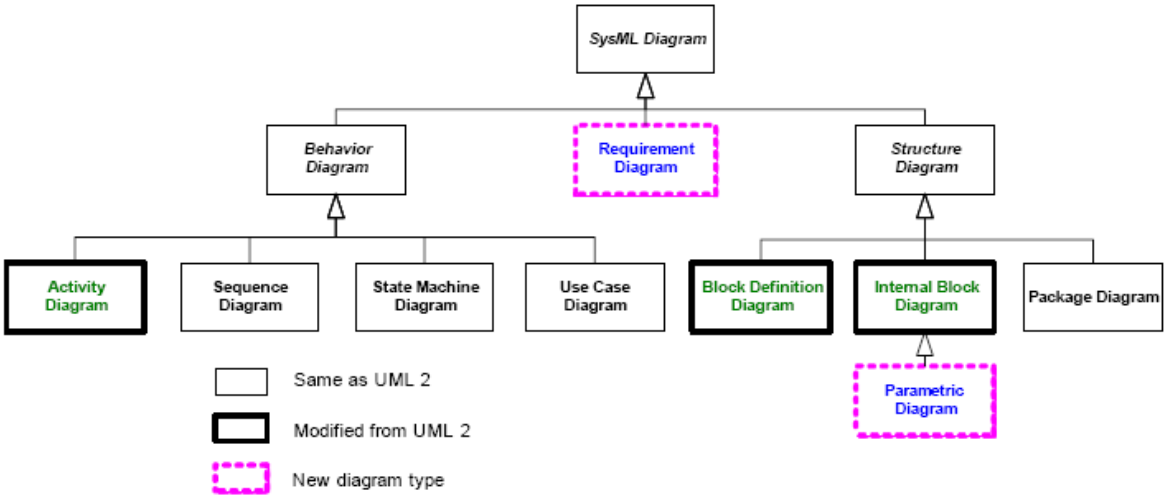


Figure 3.5. : Les différents diagrammes du langage SysML

Le langage SysML inclut un sous-ensemble du langage d'UML2, désigné par le terme «UML4SysML», et exclut la partie d'UML2 non adaptée à l'IS ou trop spécifique au domaine du logiciel. Il élargit une partie de la sémantique importée d'UML pour que ceux-ci puissent convenir dans un sens système en non purement logiciel. Par ailleurs, des éléments permettant de l'adapter à l'IS sont ajoutés. Les piliers du langage SysML sont décrits dans les sections suivantes

3.2.1. SysML Pour la modélisation des exigences

Nous avons vu que SysML inclut de nouveaux concepts propres à l'IS. C'est le cas des blocs d'exigences, permettant d'intégrer dans le modèle les exigences textuelles, structurés entre elles par des relations hiérarchiques et des liens de traçabilité. Les exigences peuvent être ainsi représentées graphiquement, structurées entres elles et reliées aux éléments de solutions au moyen de diagramme d'exigences.

De part la nécessiter à les lier (et être alloués) aux différents éléments que comporte le langage, les exigences sont souvent qualifiées d'éléments transversaux de SysML (ou *crosscutting constructs*). Elles imposent notamment la réalisation et la modélisation des activités de validation et de vérification très tôt dans le cycle de vie et tout le long des processus de l'IS.

3.2.2. SysML pour la modélisation de structures

L'élément principal pour la modélisation de structure en SysML est l'élément appelé Bloc. Celui-ci hérite du concept UML de classe, mais il a été adapté afin de permettre la modélisation structurelle d'un système. Un objet instancié à partir d'un bloc est appelé un *Part*. Un bloc peut contenir ou faire référence à des *Parts*, permettant ainsi de modéliser la structure interne d'un système. Notons que contrairement à UML, SysML n'est pas un langage Orienté Objets de par la volonté même des auteurs.

A la frontière entre les blocs et leur environnement, les ports permettent de spécifier les interfaces de ces blocs en distinguant les interfaces issues du logiciel (appelé port standard, directement importé du langage UML2), et les ports de flux, ou « flowport ». Ces derniers permettent de spécifier une interface physique, support de l'entrée et/ou de la sortie de matière, d'information continue, ou d'énergie. Les instances de bloc peuvent être interconnectées au travers de leurs ports par des connecteurs pouvant transporter des flux continus d'information, de matière et d'énergie. Les connecteurs sont les supports des éléments de flux (ou *item flow*), permettant de modéliser ce qui circule effectivement entre les instances de blocs. Les éléments de flux doivent, bien sûr, être définis conformément aux spécifications des ports par lesquels ils circulent.

Deux diagrammes permettent de représenter les structures d'un système :

- Le diagramme de description de bloc. Il permet de représenter graphiquement des relations de composition et d'agrégation de blocs, des ports d'interfaces, des attributs et opérations associés aux blocs. Ce diagramme est, par conséquent, le pendant du diagramme de classe UML.
- Le diagramme de description interne de bloc, permettant de montrer les interconnexions réalisées entre les instances (parts) à l'intérieur d'un bloc.

3.2.3. SysML pour la modélisation du comportement

SysML définit un ensemble d'éléments et de diagrammes pour modéliser le comportement dynamique d'un système. Un des éléments principaux est la modélisation d'activités. Par rapport au formalisme des diagrammes d'activités d'UML2, SysML a ajouté les aspects suivants, permettant, une fois encore, de l'adapter au domaine de la modélisation système :

- Modélisation d'échanges et de comportements continus. Les flux entre activités peuvent être discrets ou continus. Cette caractéristique est décrite dans la propriété *flow rate* des connexions.
- Support du contrôle d'une activité pendant son exécution. Une action peut être interrompue ou alimentée continuellement ou par instants avec des données d'entrées nouvelles. Cet ensemble de fonctionnalités est appelé *streaming*. Son utilisation pour la modélisation de la dynamique des systèmes continus est détaillée notamment par Johnson [Joh_07].
- Redéfinition des activités comme appartenant à un bloc. Ceci permet de définir une activité comme faisant partie de la définition d'un bloc et participe à la description de modèles systèmes.

SysML permet également de décrire des interactions, qui peuvent être représentées dans un diagramme unique, le diagramme de séquence, hérité lui aussi d'UML (et précédemment du langage SDL). Les diagrammes de séquence permettent essentiellement de représenter des messages en contexte synchrone ou asynchrone, au niveau système (les constituants de l'environnement et le système sont considérés comme une boîte noire), ou au niveau des constituants du système (ses *Parts*). Des fragments combinés permettent de composer entre eux des sous-ensembles d'interactions, afin de construire des interactions plus complexes, prenant en compte des conditions logiques, des assertions, des sections critiques, etc.

Le diagramme à états a été inclus également dans le langage SysML. Il permet de modéliser des comportements de type état transition, et n'apporte pas de modification par rapport au formalisme UML. De même, le diagramme de cas d'utilisation, permettant de représenter les fonctionnalités attendues du système et les liens entre ces fonctionnalités et les acteurs identifiés, est directement repris du langage UML.

Enfin, une extension non-normative (non incluse dans la spécification du langage SysML), restreint le diagramme d'activité de SysML dans sa forme d'origine pour être conforme au formalisme EFFBD (Enhanced Functionnal Flow Block Diagram), présenté précédemment.

3.2.4. SysML pour la modélisation paramétrique

Un ensemble d'éléments permettant la modélisation paramétrique a été inclus dans le langage SysML, afin de permettre la modélisation d'analyse quantitative dans de multiples contextes tels que l'analyse de performances, de fiabilité, ou le support d'analyse de métriques et la réalisation d'analyse de compromis.

La modélisation paramétrique est rendue possible par la définition de *bloc de contrainte*, héritant du bloc standard. C bloc de contrainte permet de définir :

- une relation de contrainte qui peut être formulée dans un langage quelconque, naturel ou spécifique,
- des attributs servant de paramètres à la contrainte,
- des instances d'autres blocs de contraintes.

Une fois instanciés et reliés aux instances des blocs système, des réseaux de contraintes sur les propriétés de ces derniers peuvent être réalisés et représentés graphiquement dans des diagrammes paramétriques.

Les blocs de contraintes sont réutilisables. Ils peuvent être représentés graphiquement sur un diagramme paramétrique. Ils sont organisés dans des packages spécifiques à un domaine particulier ou une application particulière. En outre, ils peuvent être imbriqués entre eux pour spécifier des contraintes complexes sur la base de contraintes simples.

3.2.5. Faiblesses et améliorations attendues du langage SysML.

D'autres langages basés sur UML visent l'ingénierie de système embarqués ou systèmes temps réels. Il y aura donc probablement des rapprochements à effectuer entre ces langages.

Par ailleurs, certains éléments SysML manquent de précision quand à leur sémantique (peu formalisés) ou chevauchent d'autres éléments. Par exemple, il existe une ambigüité quand à l'utilisation conjointe d'une approche basée sur les cas d'utilisation et des blocs d'exigences, les cas d'utilisation étant historiquement utilisés en UML comme un moyen de spécifier les fonctionnalités attendues du logiciel. Certains travaux de recherche ont porté sur la spécialisation de SysML, ou son utilisation dans un contexte particulier. B Fontan a par exemple proposé [Fon_08] une méthode permettant d'améliorer la prise en compte de exigences temps réels, de leur formulation, jusqu'à leur vérification sur de modèles exécutable au moyen d'observateurs générés automatiquement à partir des exigences initiales. Cette approche associé aux modèles d'exigences SysML, un profil UML préalablement définit et dédié au domaine des systèmes temps réèl. [San_

Cependant, des travaux académiques traitant de SysML tels que ceux de R. Cloutier [Clo_08], défendent le principe selon lequel une utilisation rationnelle de SysML, appuyée sur une méthodologie et adaptée au contexte et aux contraintes des entreprises et des projets, favorise une bonne efficacité de la modélisation. C'est également ce que nous essayons de démontrer dans le chapitre 4 de ce document.

Les langages et les formalismes présentés ci-dessus ont permis de préciser les rôles affectés aux modèles d'IS tels que définis par L'AFIS. Le langage SysML a aussi été présenté comme un langage adapté à ces différents rôles. Une typologie telle que celle qui a été présentée permet de les situer entre eux du point de vue de leur apport au domaine de l'IS.

De manière générale, plus la sémantique d'un langage ou d'un formalisme de modélisation est large et couvre un nombre important de concepts, et plus une démarche à appliquer pour la modélisation s'impose comme une nécessité. Ainsi, l'absence de méthode

de modélisation est considérée comme une faiblesse du langage SysML et un frein à son adoption par la communauté de l'IS.

Les langages et les formalismes doivent être resitués dans le cadre général de l'IS, en particulier relativement aux méthodes et aux processus, comme le résume la figure 2.6. Dans ce sens, ils permettent ou facilitent la réalisation des méthodes. Les méthodes permettent, quant à elles, de réaliser et de se conformer aux processus.

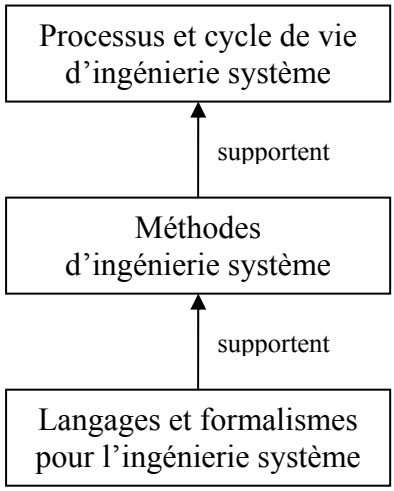


Figure 3.6.: Organisation des processus, méthodes et langages pour l'IS

Le chapitre I a présenté plusieurs standards permettant de mettre en place des processus d'IS au sein d'une entreprise. Le paragraphe suivant décrit plusieurs méthodes permettant d'utiliser les modèles d'IS étudiés jusqu'ici dans un cadre d'IS guidé par les modèles.

3.3. Méthodes et outils d'IS basée sur les modèles

Selon l'INCOSE, l'IS basée sur les modèles est *l'application formelle des techniques de modélisation pour réaliser les activités de spécification système, de conception, d'analyse, de vérification et de validation, en partant de la phase de conceptualisation, tout au long du développement, et sur l'ensemble des phases du cycle de vie* [INCTP_04]. L'application de ces méthodologies nécessitent de redéfinir les principes de bases utilisés dans un contexte ou l'IS est basée sur les documents.

3.3.1. Evoluer vers les méthodes d'IS basées sur les modèles

Dans *fundational concepts for model-driven system design* [Bak_97], Loyd Baker présente un point de vue initial sur les concepts de conception système basée sur les modèles (Model Driven System Design), en tant qu'alternative à la conception centrée sur un système documentaire (*document-centered system design*). Il présente un moyen de décrire par la modélisation les différents aspects d'un système tout au long de son cycle de vie pour apporter un gain de productivité et un moyen d'assurance qualité.

Il propose, en outre, des stratégies de transition entre une approche d'IS centrée sur les documents et une approche basée modèle, puis en présente les bénéfices attendus.

Dans ces travaux, le modèle est l'élément central des activités de V&V, lesquelles sont réalisées par interrogation du modèle. Le modèle est ensuite raffiné et modifié et les activités de V&V sont réalisées itérativement. La validation, les analyses de compromis et les évaluations nécessaires pour établir un référentiel d'exigences, une architecture fonctionnelle ainsi qu'une architecture physique sont des activités qui doivent être accomplies à travers le développement, le raffinement et l'exécution du modèle système.

Les principales activités spécifiques à la conception dirigée par les modèles qui nous intéressent sont les suivantes :

- Définition d'une modélisation exécutable du système et des spécifications des interfaces, des sous-systèmes, et des spécifications préliminaires des sous-systèmes.
- Etablissement d'un référentiel d'exigences (*design-to baseline*) et d'un référentiel système (*build-to baseline*), tous deux compréhensibles par une machine.
- Etablissement d'un modèle de performance dont le niveau de détail permet de vérifier la conformité par rapport aux spécifications.
- Exécution du modèle système pour montrer que le référentiel système satisfait aux prévisions de coûts et aux exigences de performances.
- Réalisation des revues de conception appropriées au niveau de définition, pour inclure la validation du modèle système.
- Evaluation du modèle système par rapport aux référentiels d'exigences pour évaluer la conformité avec les spécifications (vérification système).
- Validation du modèle par rapport aux données de test sur des composants fabriqués comme prescription dans *la build-to baseline*.
- Utilisation du modèle pour s'assurer qu'une fois développé, il répondra au besoin du client (validation système).

Pour organiser ces activités, L. Baker propose de se baser sur les processus de l'IEEE1220, puis définit quatre phases de développement successives à partir de ces processus: définition du système, conception préliminaire, conception détaillée, et qualification de la conception.

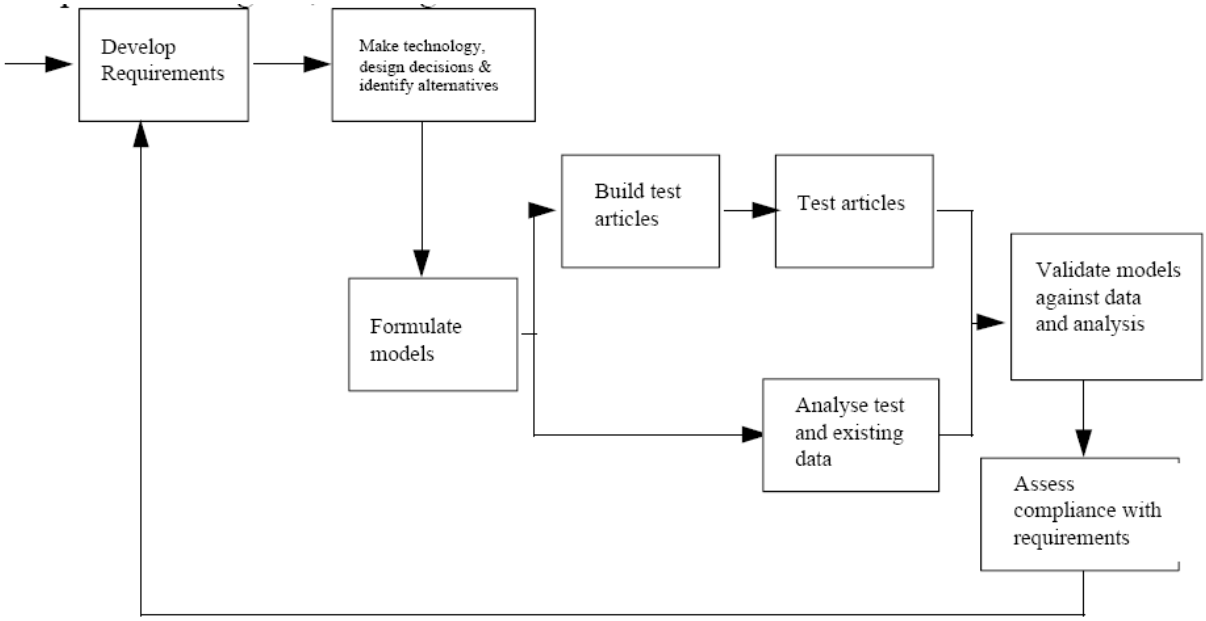


Figure 3.7. : Sous-processus MDSD [Bak_97]

A chacune de ces phases doit être appliquée le *sous-processus* décrit par la figure 3.7. Ce sous-processus doit donc être appliqué avec un niveau de détail croissant selon la phase de développement en cours.

Dans cette approche, les activités de validation et de vérification sont réalisées par interrogation du modèle système, permettant des ajustements itératifs sur les exigences et la conception jusqu'à ce que les critères de complétude soient satisfaits.

Le modèle système évolue au fur et à mesure du développement. Initialement, le modèle est de faible niveau de détail et de faible fidélité. Il est utilisé pour valider ce qui est attendu du système et pour guider la décision préliminaire de conception. Par la suite, les activités de validation, d'étude de compromis et d'évaluation doivent être accomplies à travers le développement de modèles de plus en plus détaillés. Finalement, le modèle de conception est soumis au test de conformité vis-à-vis des exigences initiales.

L'utilisation de ce processus d'ingénierie dirigé par les modèles doit apporter à l'entreprise :

- une meilleure visibilité de l'information disponible autour du système à concevoir,
- une meilleure capacité à choisir et à développer une conception optimisée,
- une meilleure rigueur dans la communication autour du système,
- une vision globale du système,
- une possibilité de réutilisation accrue.

Baker identifie aussi de nombreux avantages du point de vue du client, tels qu'une meilleure traduction de ses besoins à son fournisseur et une communication accrue sur l'avancement, les performances du projet, les solutions techniques envisagées et choisies.

Diverses entreprises appliquent déjà des méthodes d'IS basées sur les modèles ou sont en phase de transition vers ce type de pratiques. Les méthodes existantes sont en grande partie liées et définies par les développeurs d'outils qui sont, pour la plupart, déjà positionnés dans

le domaine des outils de développement logiciel. La section suivante décrit les principales méthodologies d'IS basées sur les modèles.

3.3.2. Méthodologies d'ingénierie système basées sur les modèles

Ces méthodologies sont abordées sur la façon dont elles permettent, par l'usage de cycle de vie, d'activité, et de formalismes particulier, de mettre en œuvre les principes MBSE présentés précédemment.

3.3.2.1.La méthode CORE

La méthode **Core** (Vitech) est une méthode basée sur une approche fonctionnelle et sur un cycle de vie spécifique appelé cycle *en oignon*, de par la démarche qui consiste à passer d'un niveau de détail (granularité) du modèle au suivant jusqu'à parvenir au cœur du système étudié. Ce cycle a été décrit en détail notamment par Childers et al. [Chil_94]. Il est organisé autour de quatre activités principales d'IS (voir figure 2.8) : analyse des exigences sources, synthèse - architecture, analyse des comportements, validation et vérification de la conception.

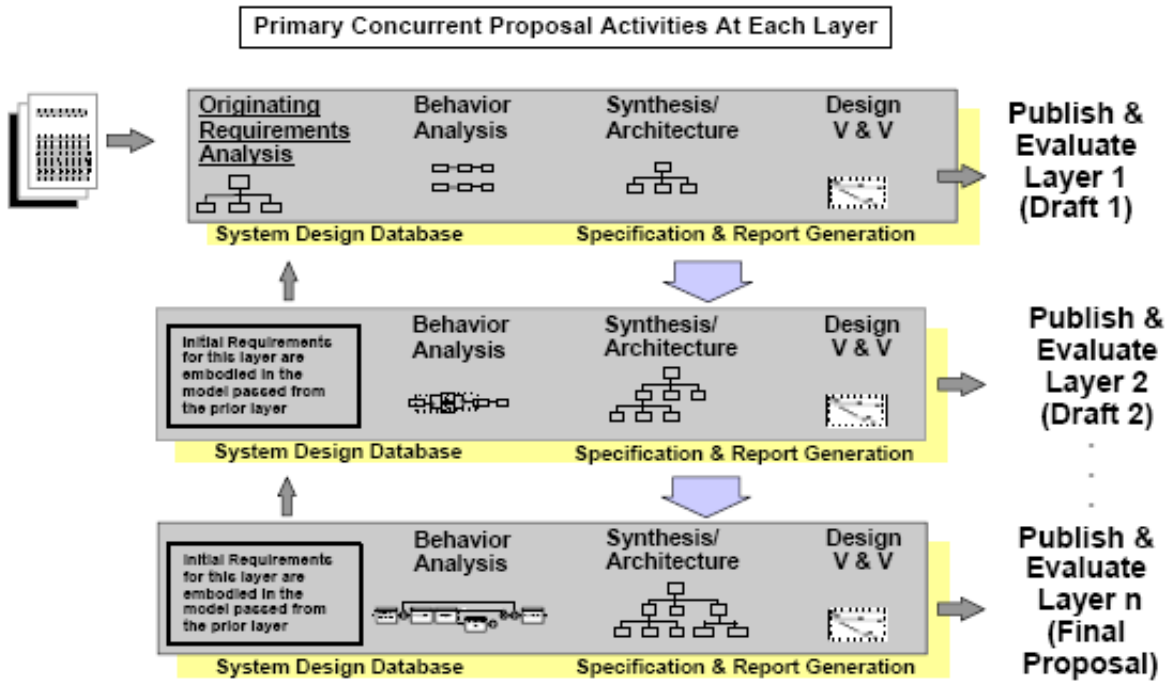


Figure 3.8. : Le cycle de vie mis en œuvre par la méthodologie Core

Ce cycle de vie comprend tout d'abord la modélisation de l'espace du problème et de la solution par la construction d'un modèle d'information. Ensuite, la conception se déroule par des itérations des activités primaires d'IS, niveau après niveau, sur l'ensemble du système à concevoir.

Ensuite, une nouvelle itération de ces activités permet de consolider la première description, afin d'en améliorer :

- La complétude : l'équipe d'IS doit compléter une couche de description avant de poursuivre sur la suivante.
- La convergence : l'équipe d'IS ne doit pas revenir en arrière de plus d'une couche.

Ce découpage en couches cohérentes de définition du système permet de réaliser dés le début du projet une définition préliminaire validée du système et, par conséquent, de réduire les risques liés aux incertitudes du projet. La décomposition en niveaux s'arrête lorsque le niveau atteint correspond au niveau de détail requis (composant / pièce élémentaire, sous-ensemble, équipement, etc.).

La méthodologie s'appuie sur les formalismes de modélisation EFFBD et UML, ainsi que sur un langage de description système (SDL : System Description Langage) qui permet de modéliser et de définir comment les éléments du modèle sont gérés et validés et comment celui-ci peut être communiqué et *tracé* par rapport aux autres éléments.

Un inconvénient de l'outil Core est qu'il rend difficile la définition de plusieurs alternatives de solutions, car une seule architecture système peut être définie lorsque le développement passe d'un niveau au suivant.

3.3.2.2. La méthode Harmony-SE

La méthode **Harmony-SE** combine à la fois l'IS et l'ingénierie logicielle, pour les systèmes à dominante logicielle. Elle constitue une extension au processus *Harmony* destiné à la conception de logiciel (voir figure 2.9).

Harmony-SE suit un cycle en spirale, adapté du processus ROPES (Rapid Optimizing Process for Embedded Systems) [Dou_99], pour lesquelles les itérations de conception (design itération) sont des étapes de spécification, de conception, de modification des exigences incomplètes ou ambigües, d'implémentation et enfin de validation. Dans la méthodologie, chacune de ces étapes, y compris la spécification, est réalisée par l'intermédiaire de l'exécution du modèle et de sa validation. Les itérations correspondent à une augmentation des fonctionnalités du système par l'analyse et le raffinement des cas d'utilisation, définis en début de développement et qui permettent progressivement d'enrichir le modèle de solution et de spécifier les caractéristiques des composants.

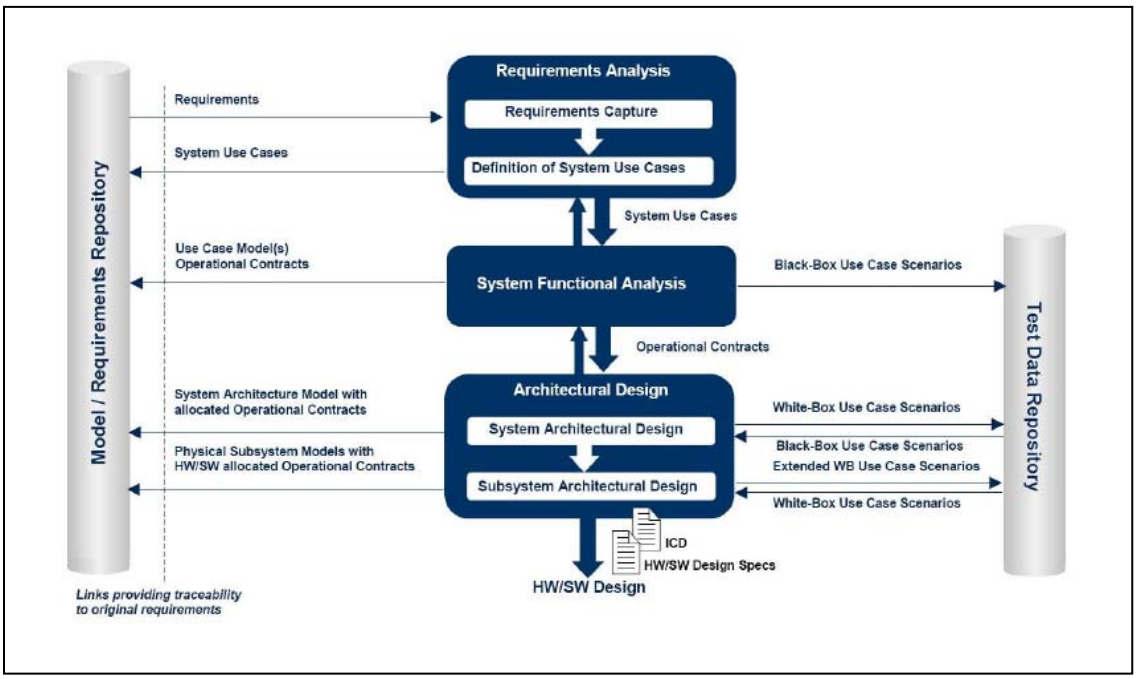


Figure 3.9. : Le processus globalement descendant d'harmony SE

Pour cela, le processus s'appuie sur les langages SysML et UML2 et utilise les mêmes outils. Il permet ainsi de transférer, de la conception système à la conception logicielle de manière cohérente, les exigences et l'architecture. Les exigences font partie intégrante du modèle. Elles sont basées modèle et non texte. Les diagrammes de séquences sont utilisés pour spécifier mais également pour décrire les vecteurs de test intermédiaires pendant la construction incrémentale du système ainsi que les tests finaux.

La méthode d'IS poursuit les objectifs principaux suivant :

- Identifier ou déduire les fonctionnalités du système.
- Identifier les états et modes associés aux fonctionnalités.
- Allouer les fonctionnalités et les modes du système à l'architecture physique.

Les tâches de la méthode Harmony-SE sont donc organisées autour de trois activités :

- Analyse des exigences.
- Analyse fonctionnelle au niveau système par la définition de cas d'utilisation.
- Conception architecturale.

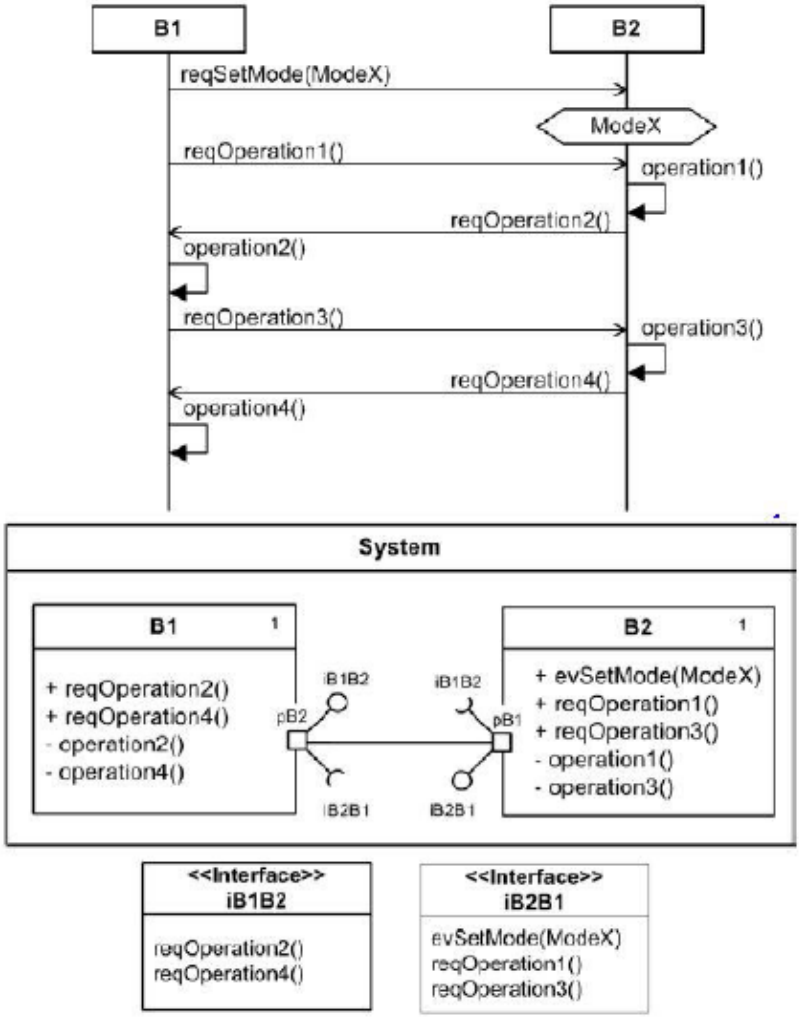


Figure 3.10. : Définition de contrats opérationnels

Harmony-SE utilise une approche de modélisation orientée service. L'analyse du système permet de définit un ensemble de cas d'utilisation validés individuellement. Ensuite, une phase de synthèse permet d'établir une représentation système boîte noire, indépendante de solutions techniques, sur laquelle est représentée un ensemble de contrats opérationnels (*operationnal contracts*) qui représente les services attendus du système (voir figure 3.10). Après définition d'une architecture et du partitionnement matériel/logiciel, les cas d'utilisation (et les scénarios opérationnels) sont projetés sur chaque composant de l'architecture. Les contrats opérationnels associés sont alloués sur chaque composant de l'architecture, permettant de définir les ports et les activités au niveau des composants. La synthèse de chaque cas d'utilisation, au niveau de l'architecture système permet de définir les états / modes attendus pour chaque composant, ainsi que leurs ports d'interface. Ces étapes sont normalement réalisées en utilisant la notation SysML.

Par ses processus et les outils qui permettent de la mettre en oeuvre, la méthode Harmony-SE permet de réconcilier la démarche fonctionnelle / procédurale qui guide le travail des ingénieurs systèmes avec l'approche orientée objets utilisée par les ingénieurs logiciel. Par conséquent, Harmony peut facilement être adaptée aux pratiques de *co-design* Matériel/Logiciel, et se conformer aux standards DO-178B et CMMI (*Capability Maturity Model Integration*). Du fait de l'utilisation de cas d'utilisation et la définition de contrats

requête / service pour représenter le domaine du problème, l'application de cette méthodologie est adaptée principalement aux systèmes à logiciel prépondérant.

3.3.2.3. La méthode OOSEM

Contrairement à Harmony-SE, la méthode d'IS orientée objets **OOSEM** (*Object Oriented Systems Engineering Method*) [Lyk_01] suit le cycle en V traditionnel de l'IS, appliqué de manière récursive sur chaque sous-système identifié. La figure 3.11 donne un aperçu de cette approche.

Les objectifs de cette méthode sont cités dans les travaux de J. A. Estefan [Est_08]. Il s'agit principalement de :

- L'intégration des méthodes orientées objets logiciel/matériel et des bonnes pratiques d'IS.
- La capture et l'analyse des exigences ainsi que leur évolution, les risques associés, leur lien de traçabilité avec les scénarios opérationnels, les composants logiques et les ressources physiques à travers une base de données.
- L'optimisation et l'évaluation d'alternatives tout au long du processus. Des modèles paramétriques permettent d'optimiser les solutions candidates logiques et physiques et de les comparer selon des critères objectifs et traçables à partir des exigences du système et de l'entreprise.
- Le support de la réutilisation au niveau système et de l'évolution de la conception.

Pour cela, OOSEM s'appuie sur une approche descendante basée modèle. Elle utilise le langage SysML pour réaliser la spécification, l'analyse, la conception et la vérification des systèmes.

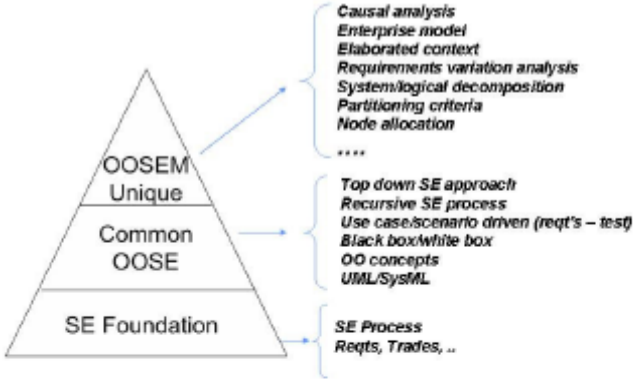


Figure 3-6. Foundation of OOSEM.

he OOSEM supports a SE process as illustrated in Figure 3-7.

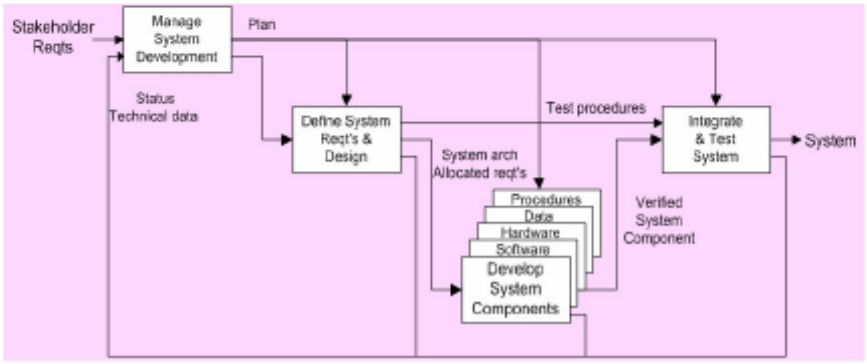


Figure 3-7. OOSEM Activities in the Context of the System Development Process.

Figure 3.11. : Eléments de base et activités de OOSEM [Lyk_01]

Dans OOSEM, la démarche s'appuie sur la réalisation d'un ensemble d'activités, facilitée par la réalisation de sous-activités communes. La première activité définie consiste en une étude de l'environnement du système à développer (utilisateurs et systèmes en interaction) et une modélisation de l'entreprise et des moyens disponibles pour développer le système. Ceci permet de définir le concept opérationnel et les exigences de la mission du système en fonction du contexte opérationnel et des capacités et limitations de l'entreprise. A partir des scénarios développés, les exigences fonctionnelles, d'interface, de performances, de fiabilité, etc. sont définies et collectées dans une base de données en conservant un lien de traçabilité avec les scénarios d'origine.

Ensuite, une architecture logique est constituée de manière cohérente avec les interactions décrites dans la vision scénario/boîte noire précédente selon des critères de partitionnement tels que la cohérence, la fiabilité, le découplage, l'évolutivité, la performance, etc.

Les composants logiques de l'architecture interagissent pour satisfaire aux exigences de niveau système définies précédemment. La conception de l'architecture logique sert à réduire le risque de changement des exigences sur la conception et à faciliter l'évolution du système (liées par exemple aux changements de technologie).

Les composants logiques sont ensuite distribués sur des ressources physiques génériques, éléments matériels, logiciels, données et procédures, et chaque composant physique résultant de l'allocation est spécifié dans la base de données d'exigences. Une fois qu'une architecture physique a été optimisée et sélectionnée, OOSEM requiert le développement des plans de

vérification, des procédures et des méthodes de vérification (cas de test) à partir des scénarios opérationnels précédents et des exigences définies tout au long des phases de définition de l'architecture logique retenue et de l'architecture physique. La figure 3.12 résume ces étapes.

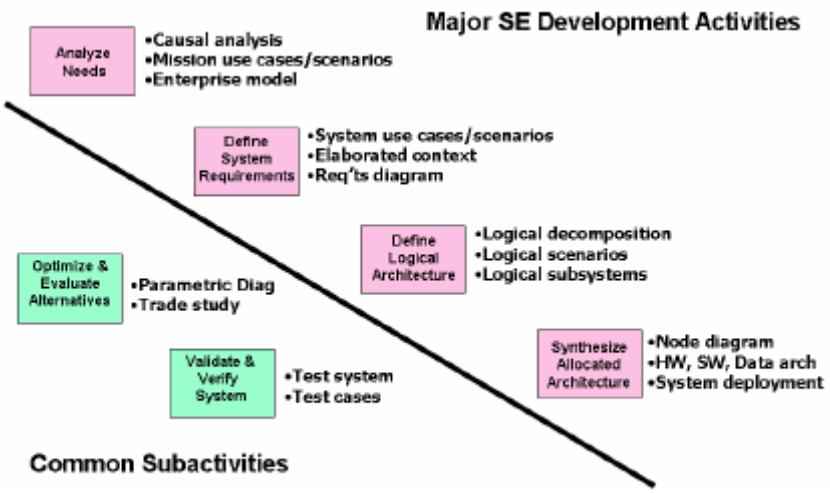


Figure 3-8. OOSEM Activities and Modeling Artifacts.

Figure 3.12. : Organisation des activités OOSEM

Une des activités communes remarquables de la méthode OOSEM est l'activité d'optimisation et d'évaluation des alternatives. Des modèles paramétriques sont utilisés pour modéliser les performances, la fiabilité, la disponibilité, les coûts sur le cycle de vie et d'autres aspects spécifique de l'ingénierie spécialisée. Ces modèles permettent d'analyser et d'optimiser les architectures candidates au niveau requis afin de comparer les alternatives et d'en sélectionner une. Les critères et les facteurs de pondération utilisés pour réaliser les analyses de compromis sont traçables jusqu'aux exigences système et aux mesures d'efficacité. Cette activité inclus aussi la surveillance des mesures de performances techniques et identifie les risques potentiels.

La méthodologie OOSEM a été mise en œuvre par la société « Lockeed Martin » dans le cadre de la refonte de son système d'information.

D'autres méthodologies d'IS basées sur les modèles, telles que Mega ont été développées ou sont en cours de développement.

3.3.2.4. Synthèse des méthodologies d'ingénierie système

Les principaux éléments des méthodologies présentées ci-dessus sont résumés dans le tableau 3.1.

	Core	**Harmony-SE**	**OOSEM**
Cycle de vie / Paradigme de modélisation	Cycle en *oignons* spécifique à la méthodologie.	Cycle en spirale guidée par la définition et le raffinement de cas d'utilisation	Cycle en V récursif
Langages et formalismes	Diagrammes EFFBD et UML.	SysML / UML.	SysML.
Support des outils existants	Core	Tout outils supportant SysML / UML.	Tout outils supportant SysML / UML.

Tableau 3.1. : Récapitulatif des méthodes d'IS guidées par les modèles.

Malgré les différentes solutions associées à chacune de ces méthodologies, celle-ci respectent un ensemble de principes communs, rappelés et résumés ci-dessous. Ceux-ci peuvent être considérés comme des principes méthodologiques pour l'IS guidée par les modèles.

Principe 1. *Interopérabilité entre les modèles utilisés pendant le développement d'un système.*

Les méthodologies MBSE (*Model-Based Systems Engineering*) s'appuient sur l'articulation de modèles utilisant chacun un point de vue différent du système. Des mécanismes permettant de s'assurer de la cohérence entre ces différentes vues sont donc nécessaires.

Principe 2. *Intégration de la modélisation du problème et des exigences dans le modèle système.*

Les méthodologies permettent de modéliser le système d'un point de vue extérieur et sur les fonctionnalités attendues (exigences, cas d'utilisation, concepts d'opération, contrats opérationnels, etc.).

Principe 3. *Modélisation de problèmes et de solutions interprétables par l'outil informatique.*

La modélisation conjointe des deux domaines permet que les modèles de solutions soient évalués par rapport aux modèles du problème tout-au-long de la conception. Ce lien est généralement réalisé par une relation de traçabilité dont la sémantique est plus ou moins formellement définie, entre les différents éléments du modèle.

Principe 4. *Maintient de la cohérence du modèle dans chaque phase de développement.*

La cohérence des modèles doit être vérifiée au fur et à mesure de l'avancement dans le développement. Les méthodologies incluent souvent des moyens d'évaluer l'impact d'une évolution d'un élément sur le reste de la modélisation.

Principe 5. *Support des analyses et des décisions de conception.*

En particulier, plusieurs modèles de solutions doivent pouvoir être développés et évalués simultanément pendant la phase préliminaire du développement du système. L'analyse d'impact des évolutions doit pouvoir se faire.

Principe 6. *Support des activités de validation des exigences et de la conception.*

Les méthodologies facilitent généralement l'activité de validation système par la traçabilité des éléments du modèle et par l'animation ou l'exécution de celui-ci (pour la validation qualitative des comportements).

Les derniers principes ci-dessus sont pauvrement implémentés dans les méthodologies actuelles. Une des causes de cette limite repose :

- sur le manque de formalisation et de moyens théoriques qui pourraient être mis en œuvre.
- sur le manque d'interopérabilité entre les modèles.

Sur le premier point, des travaux de recherches tels que ceux de G. Harmel [Har_07] proposent de maîtriser les couplages existant entre conception et conduite de projet par une formulation matricielle du problème et l'application de méthodes associées. En particulier, la recherche d'architecture optimisée du point de vue conception et conduite de projet est réalisée comme le regroupement optimal d'éléments au moyen de méthodes matricielles et d'algorithmes (génétique, logique flou).

3.4. Problématique liée à l'interopérabilité des modèles

L'étude des méthodes existantes d'IS dirigées par les modèles a permis d'établir les principes généraux retenus. Ce sous-chapitre poursuit l'étude en identifiant les points d'amélioration nécessaires pour mieux tirer profit de l'apport des modèles en IS. Pour cela, il est nécessaire :

- d'évaluer quels niveaux d'interopérabilité des modèles sont assurés par les méthodologies décrites précédemment,
- d'identifier les moyens (méthodes et outils) disponibles afin de permettre une progression dans l'interopérabilité des modèles.

Par interopérabilité, nous entendons la capacité que possède un modèle à être lié et intégré à un ensemble de modèles et à être exploités avec eux pour satisfaire un objectif ou un ensemble d'objectifs précisément définis. L'interopérabilité repose donc sur l'utilisation de modèles de manière non individuelle mais comme un ensemble.

L'interopérabilité est une notion générale qu'il est nécessaire de préciser et de lier aux méthodes étudiées précédemment. Plus précisément, la notion d'interopérabilité pourrait être déclinée en interopérabilité *fonctionnelle*, découplée des techniques mises en œuvre pour assurer celle-ci. Appliquée aux modèles d'IS des méthodologies décrites, plusieurs niveaux peuvent être distingués.

Le niveau le plus faible d'interopérabilité d'un modèle peut être vu comme la manipulation par plusieurs modèles d'un même ensemble de concepts dont la sémantique est

commune, au moins partiellement, et exprimée selon une syntaxe commune. Ce niveau d'interopérabilité peut alors être considéré comme une simple compatibilité entre modèles. (ex : compatibilité de deux fichiers).

A un niveau supérieur, l'interopérabilité des modèles désigne la capacité à rendre possible un échange de données entre deux modèles dont les sémantiques diffèrent afin d'atteindre un objectif plus facilement que par l'utilisation individuelles des modèles. L'échange de données peut alors être facilité par la définition de standards pour l'échange de données (ex. : XML) ou de métadonnées (ex. : XMI) entre les modèles.

Enfin, un dernier niveau peut être défini lorsque les échanges de données effectués entre les modèles ayant des sémantiques différentes sont organisés selon un cadre méthodologique précis. Dans ce cas, les échanges sont structurés entre eux et c'est l'enchaînement structuré de ces échanges qui permet d'atteindre l'objectif d'interopérabilité. C'est le cas des méthodologies étudiées au sous-chapitre 2.3.

Des travaux se basent sur une interopérabilité accrue entre les modèles ou entre leurs sémantiques pour apporter, dans un domaine d'application particulier, une assistance ou une amélioration des activités de spécification, de conception et de validation. Certains de ces travaux sont présentés dans la section suivante.

3.4.1. Travaux sur l'interopérabilité des modèles en IS

L'interopérabilité des modèles appliquée au domaine de l'IS consiste à construire des ponts efficaces, notamment entre :

a) La définition interne du modèle et les éléments du modèle manipulés et vus par les utilisateurs.
b) Les différents modèles utilisés depuis l'identification du besoin opérationnel jusqu'à la définition des solutions techniques, et en particulier pour identifier des éléments de solutions et/ou faciliter leur intégration dans une solution globale.
c) Les solutions techniques localisées (ex : les composants d'une architecture) et les propriétés émergentes attendues répondant aux exigences fonctionnelles exprimées au niveau système.

Ainsi l'approche présentée par Johnson et al. [Joh_08] vise à améliorer l'utilisation conjointe du langage SysML et de modèles spécifiques aux domaines par une approche de transformation de modèle basée sur :
- les métamodèles des deux langages,
- la transformation par des règles déclaratives de transformation de graphe.

L'approche adoptée est représentée en figure 3.13. Elle est une manière efficace et pérenne d'assurer l'interopérabilité entre deux modèles dont les sémantiques sont différentes, mais dont l'une peut être (partiellement) déduite de l'autre.

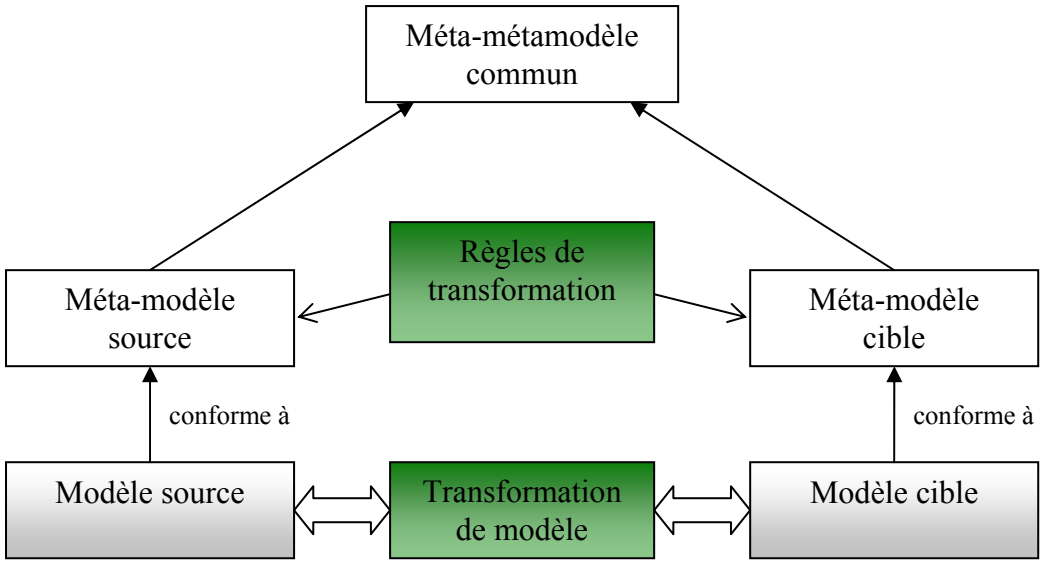


Figure 3.13. : Technique de transformation de modèle basée sur les métamodèles

Dans le cas de ces travaux, une transformation bidirectionnelle est réalisée entre des modèles SysML et des modèles Modelica.

L'approche décrite propose un cadre méthodologique reposant sur l'interopérabilité des modèles pour assurer la sélection de composants en phase de conception et la validation automatique de la cohérence et de la complétude des exigences bas niveaux spécifiant les systèmes embarqués. La mise en œuvre de l'approche permet ainsi une réduction importante du temps de développement en évitant les itérations liées à cette phase de validation.

Ces étapes sont réalisées sur la base d'une sémantique globale permettant de réunir à la fois les exigences produit et les fonctionnalités et caractéristiques internes des composants.

Les exigences sont modélisées sous la forme d'un modèle SysML ou UML et sont formalisées par la définition d'éléments (produits, applications, contraintes…) et de relations entre ces éléments.

Un modèle de données en OWL (*Web Ontology Language*) est développé pour capturer :

- les concepts spécifiques au domaine d'application (ex : description des liens entre capacités des composants et contraintes),
- les concepts communs pour la spécification des exigences produits conformes au domaine d'application (ex : contraintes de limitations de ressources),

et pour l'évaluation de contraintes sur les composants et sur l'architecture logicielle (ex : éléments descriptif de QoS).

Dans ce contexte, l'utilisation des techniques de modélisation et d'ingénierie des modèles est intéressante pour :

- produire automatiquement des structures partielles UML sous la forme de configurations de composants satisfaisants à un ensemble d'exigences produit.
- Utiliser des règles applicables pour la validation des exigences produit et la vérification de la couverture de celles-ci par l'architecture la vérification de la structure du modèle et la vérification des contraintes numériques.
- Utiliser des requêtes pour la recherche automatique de composants ou de configurations de composants satisfaisant aux exigences produits.

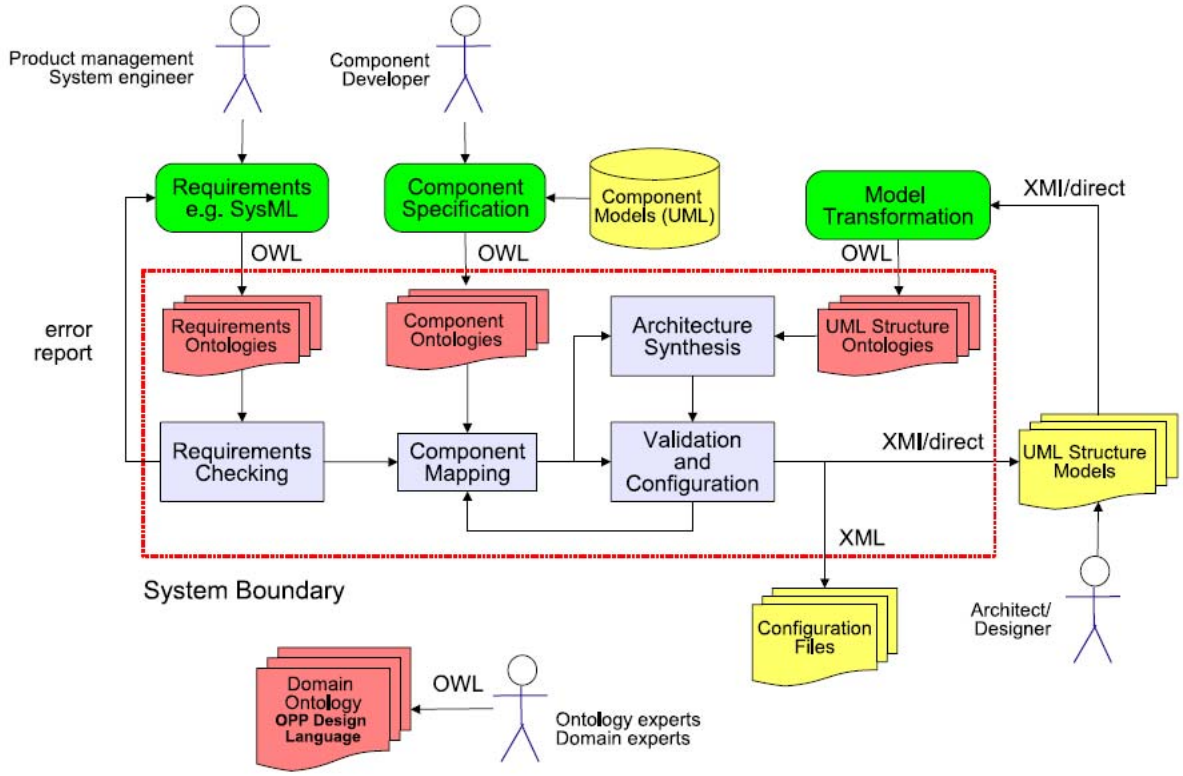


Figure 3.14 : Interopérabilité des modèles [Car_07]

Une vision de l'interopérabilité des modèles est fournie par la figure 3.14. Plusieurs limites ont été identifiées par les auteurs de ce travail. D'une part, la démarche ne couvre pas, pour l'instant, les exigences sur le comportement (par exemple, les exigences exprimées par la définition de cas d'utilisation, de diagrammes de séquence et d'activités.). Par ailleurs, la vérification de règles reste limitée à la relation entre éléments spécifiés (requis) et éléments de la conception. D'autres vérifications pourraient concerner la cohérence entre éléments spécifiés.

Notons également que la mise en œuvre de l'approche par l'utilisation de nombreux outils interconnectés parait difficile à maintenir sur des échelles de temps compatibles par exemple avec la durée d'un programme aéronautique, et à faire évoluer, notamment vers la prise en compte d'exigences systèmes actuellement non couvertes par la méthode (exigences processus, exigences comportementales et exigences de performances).

Cette démarche a toutefois démontré son efficacité pour la spécification et la conception d'un système dans le contexte *domain specific* de la conception d'appareils électroniques embarqués grand public.

Sur ces les objectifs b) et c) présentés en début de sous-section, Robert Cloutier, dans *Model Driven Architecture for Systems Engineering* [Clo_08a], [Clo_08b] montre l'applicabilité des mécanismes d'utilisation et de transformations de modèles de l'approche MDA et les bénéfices potentiels de son utilisation dans le contexte de l'IS.

L'approche MDA est une approche de développement logiciel basée sur l'usage et la transformation successive de modèles. MDA distingue en particulier les quatre types de modèles ci-dessous afin de parvenir à un modèle permettant de générer du code :

- Le modèle indépendant du calcul (C.I.M. – *Computer Independent Model*) décrit le point de vue du client ou de l'utilisateur du système, exprimé conformément et dans les termes relatifs à son domaine ou à son métier. Il s'agit donc d'une vue descriptive du besoin et des exigences.

- Le modèle indépendant de plates-formes (P.I.M. – *Platform Independent Model*) permet de définir une vue du système à un niveau d'abstraction permettant une certaine indépendance vis-à-vis des plate-formes sur lesquelles l'application peut être développée.

- Le modèle de plate-forme (P.M. – *Platform Model)* regroupe une représentation des concepts et des mécanismes internes à une plateforme afin qu'elle soit utilisable pour définir une modèle spécifique de plate-forme, dans le cadre du développement d'un système.

- Le modèle spécifique à une plate-forme (P.S.M.) représente un modèle PIM qui a été adapté aux spécificités d'une plate-forme cible. La définition d'un modèle spécifique à une plate-forme se base donc sur un modèle de plate-forme. C'est depuis ce modèle que le code de l'application peut être généré.

En particulier, R. Cloutier [Clo08b] montre de quelle façon ces différentes classes de modèles peuvent être transposées dans le domaine de l'IS. Dans l'approche SE-MDA (System Engineering MDA) proposée par l'auteur, le C.I.M. permet d'exprimer les concepts d'opérations du système, en définissant le système comme une boîte noire, et ses interactions avec son environnement, ses objectifs, ses exigences opérationnelles, ainsi que l'expression des besoins et du marché. Dans SE-MDA, ce modèle C.I.M. est considéré comme étant une architecture opérationnelle. Ensuite, un P.I.M. définit, éventuellement sur plusieurs niveaux de décomposition, comment le système réalise ces fonctionnalités au niveau système. Des groupements réalisés dans les modèles permettent de faire émerger les sous-systèmes et les constituants du système, sans toutefois entrer dans le détail de la conception. Le P.I.M. permet alors de décrire l'architecture système, et par conséquent comment les exigences opérationnelles sont décomposées en exigences techniques. La figure 2.15 montre l'évolution des modèles CIM vers PSM.

Les phases de définition des modèles C.I.M. et P.I.M. sont itératives, et constituent ensemble la partie d'ingénierie du système. La transition suivante, des P.I.M. descriptifs de l'architecture système au modèle P.S.M. et à la spécification des composants de celui-ci correspond au travail réalisés par les métiers de la conception détaillée électronique, logiciel, etc.

Pour l'auteur, l'objectif à terme est de passer de C.I.M. à P.I.M. par le biais d'une transformation automatisée. Il propose pour cela l'usage de patrons de conception (*design pattern*) au niveau des C.I.M. et qui peuvent ensuite être décomposés sous la forme de P.I.M.

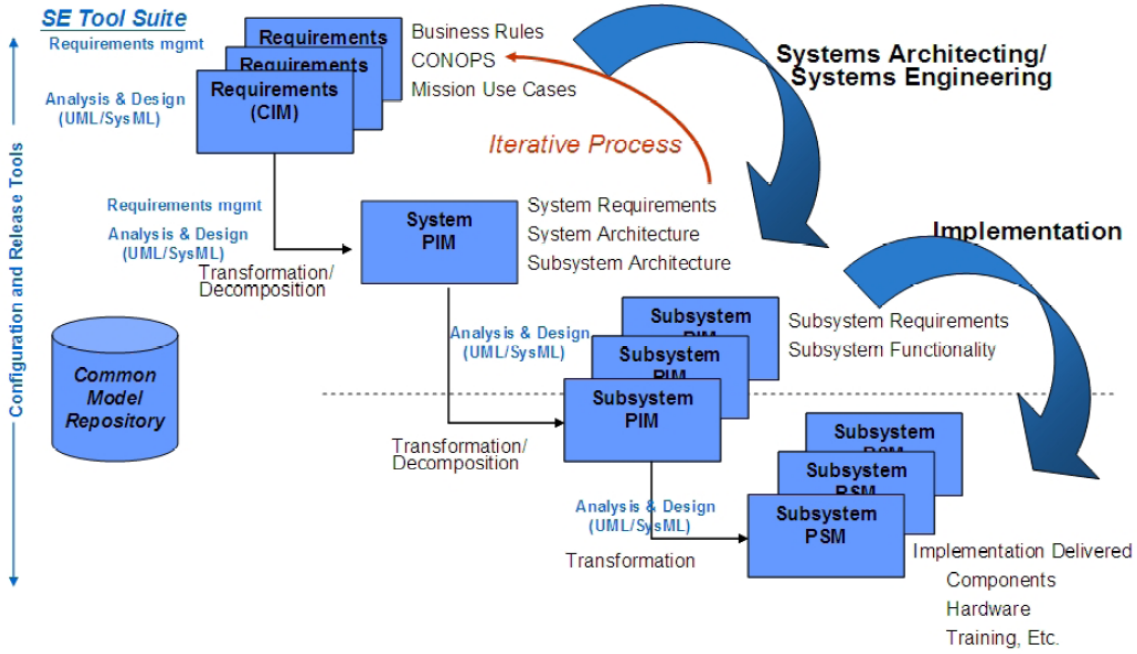


Figure 3.15. : Application de MDA en ingénierie MBSE [Clo08b]

Ramenées aux méthodologies d'IS existantes guidées par les modèles, les démarches présentées ci-dessus constituent des avancées intéressantes pour améliorer l'interopérabilité des modèles et son bénéfice sur le développement d'applications ou de systèmes.

Relativement à la problématique de cette thèse, les démarches que nous avons présentées couvrent une partie du périmètre envisagé. Trois aspects font défaut :

- La conformité de ces méthodologies à un standard d'IS n'est pas évaluée. Dans ce sens, l'adaptabilité d'une telle approche à des problèmes différents (autres domaines d'application de l'IS) ou une approche plus générique paraît difficile.

- Elles n'incluent généralement pas la phase d'élicitation et de spécification des exigences système, ne proposent pas de moyen de répondre aux besoins de formalisation propres à l'ingénierie des exigences (bien que ce point ait été identifié partiellement comme une perspective dans les travaux de I. Cardei et al. [Car_07]).

- Leur périmètre est strictement limité au système-produit final. Lorsque les décisions de conception sont fortement influencées ou contraintes par les produits contributeurs (ce qui est particulièrement le cas en ingénierie simultanée) ces approches doivent être adaptées.

Ces travaux présentent toutefois un intérêt important en tant que démonstration de la faisabilité d'une méthode et d'une mise en œuvre pour la spécification, la conception et/ou la validation automatisée basée sur les modèles dans le contexte de l'IS.

La prise en compte d'un standard d'ingénierie système a récemment été réalisée dans la méthodologie MISSyM [Tur_08], destinée au développement des systèmes mécatroniques. Cette méthodologie permet, par une approche guidée par les modèles, de se conformer aux activités d'ingénierie du standard ISO/IEEE15288 [ISO_O8]. Le langage SysML a été adapté au moyen d'un profil afin de s'adapter au contexte (mécatronique) et au standard I.S. Il permet aussi de produire un code dans le langage de simulation système Modelica.

Nous avons pu voir au travers de ces travaux que la disponibilité d'une sémantique commune et partagée entre les différents sous-modèles est un facteur important d'interopérabilité. Pouvoir disposer d'un modèle sémantique global du domaine de l'IS est donc une priorité pour progresser dans l'interopérabilité de l'IS guidée par les modèles. C'est dans cet objectif que l'AFIS a développé le modèle de données présentés au paragraphe suivant.

3.4.2. Modèle de données d'ingénierie système AFIS

L'AFIS, par le biais des modèles de données d'IS décrits ci-après, associe une sémantique précise aux différents types de données d'IS et aux relations entre ces données. Cette sémantique se veut commune à chaque domaine d'application possible de l'IS. Elle est partagée par chaque partie prenante. L'objectif de la définition de ces modèles de données est de bâtir une base conceptuelle de référence, pour les échanges entre industriels et avec les organismes de standardisation ou de normalisation ainsi qu'avec les éditeurs d'outils [Afi].

Le modèle de données I.S. de l'AFIS est divisé en un ensemble de vues : Concepts généraux, affaires, exigences, gestion de configuration, et IVV (Intégration, Validation et Vérification). Chacune de ces vues est modélisée sous la forme d'un diagramme de classe et contient un ensemble de données et de relations entre ces données. Une description associée à chaque donnée est faite dans la vue d'origine. Dans la suite de ce paragraphe, nous présentons les principaux modèles de données utiles pour la suite de la démarche.

Tout d'abord, l'AFIS propose une description des concepts généraux de l'IS et du périmètre couvert par les modèles de données détaillés selon la figure 3.16.

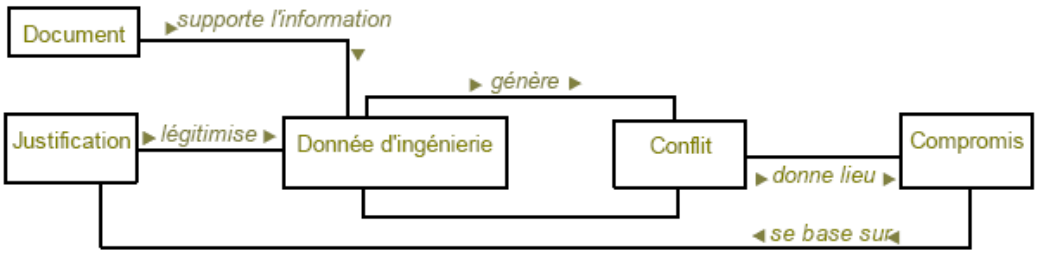


Figure 3.16 : Concept généraux de l'IS vus par l'AFIS

Les données d'ingénierie sont les différents éléments manipulés dans un processus d'IS. La donnée *document* doit être comprise dans son sens général de support de la donnée d'ingénierie. Elle peut être un document papier, une fiche informatique, un modèle. Le périmètre du système est le système *final* de l'EIA 632. La notion de conflit est décrite ici

comme une *incohérence entrainant un risque d'échec par rapport à un but défini dans un contexte donné*.

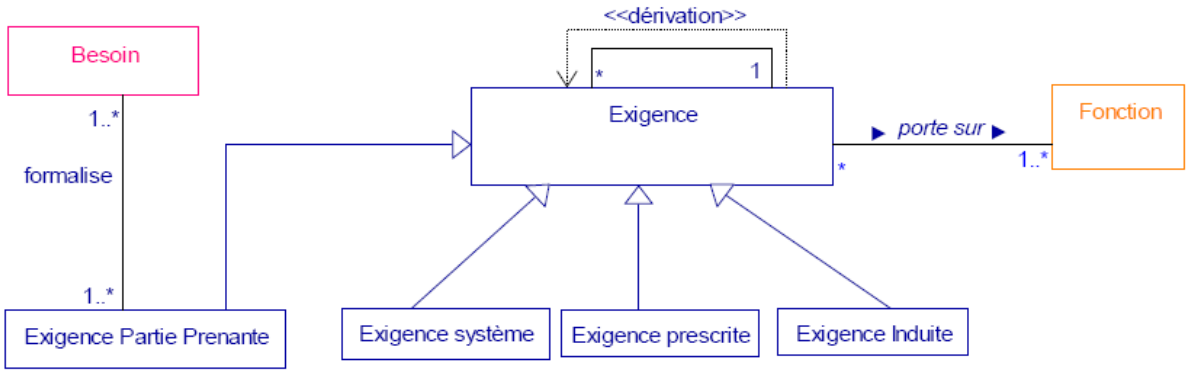


Figure 3.17. : Vues AFIS des Exigences

L'AFIS fournit les définitions suivantes aux différents types d'exigences présentées en figure 2.17. Une exigence induite est générée par un choix d'architecture ou une contrainte de réalisation . Une Exigence prescrite est une exigence technique dérivée, d'une ou plusieurs exigences système allouée à un composant physique et ayant valeur contractuelle. Elle correspond aux exigences allouées (*assigned specified requirement*) de l'EIA632. L'exigence système est une exigence technique allouée aux bornes d'un système.

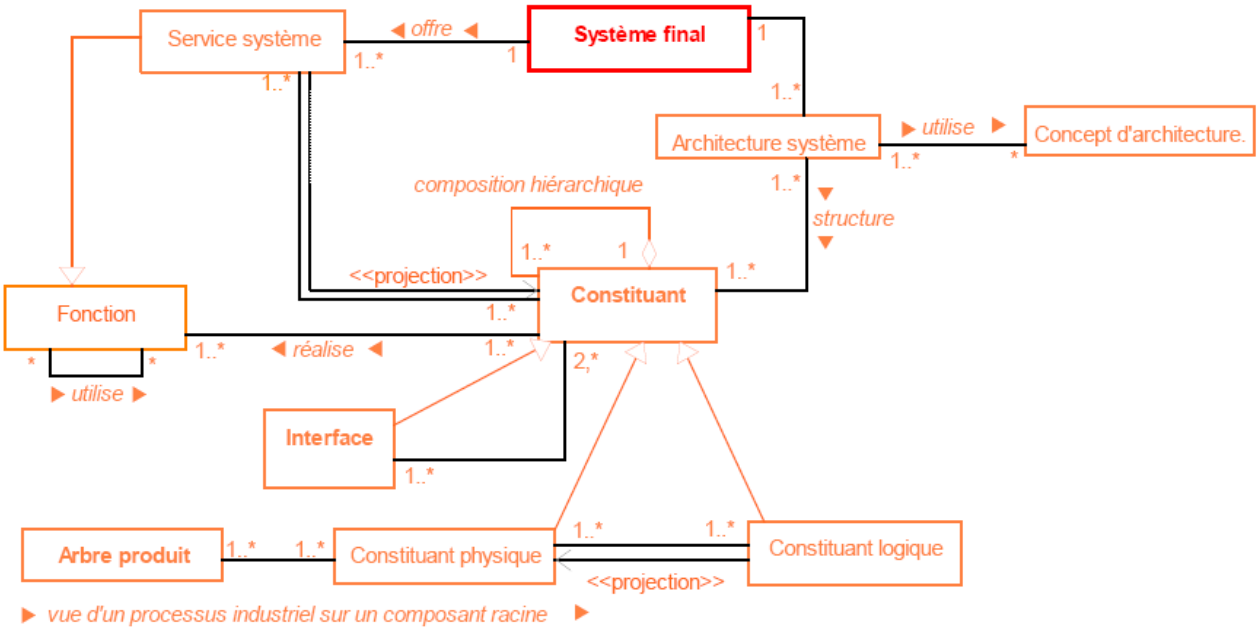


Figure 3.18. Vue AFIS de l'architecture système

Les constituants ou éléments issus d'un choix de décomposition système sont de nature logique (indépendant des solutions physiques et technologiques) ou physique (figure 2.18). Ils peuvent, en outre, constituer des interfaces. Ces constituants réalisent les fonctions et sont structurés dans une architecture système. Ici, le terme architecture système est conforme à la définition de la norme IEEE-1471 [IEE_00]. Une architecture peut être *candidate* ou *effective*, selon qu'elle entre dans une phase de sélection parmi d'autres architecture ou qu'elle a été

retenue pour être réalisée. Par ailleurs, l'AFIS inclut la notion de concept d'architecture. Ainsi, une architecture système utilise des concepts d'architectures, c'est-à-dire des *architectures types capitalisées*. La distinction entre constituants physique et constituants logiques peut être appliquée également aux concepts d'architecture.

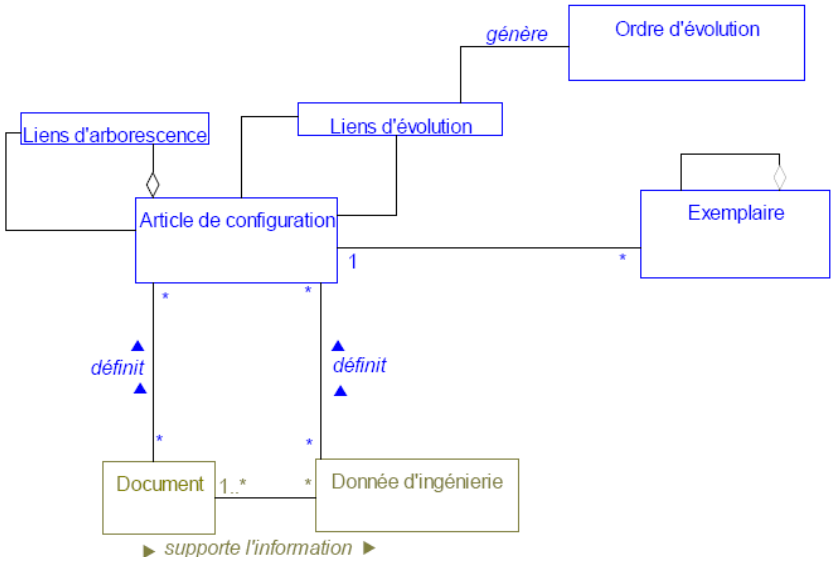
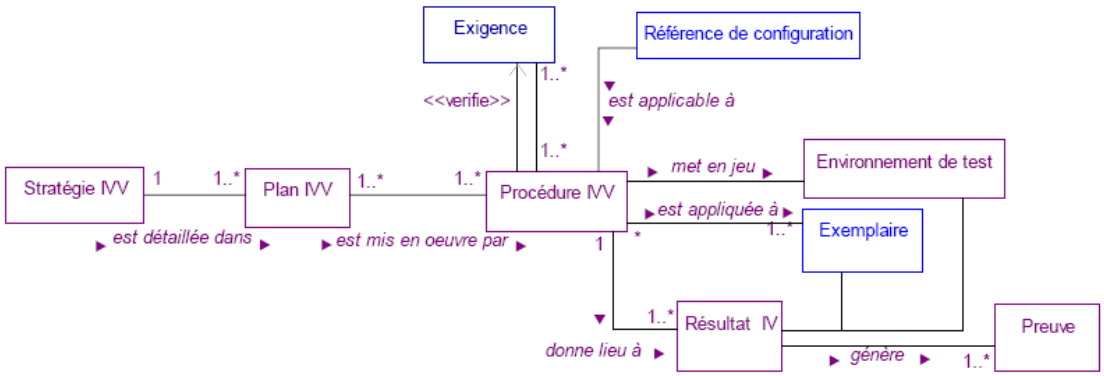


Figure 3.19. : Vue AFIS : Gestion de configuration



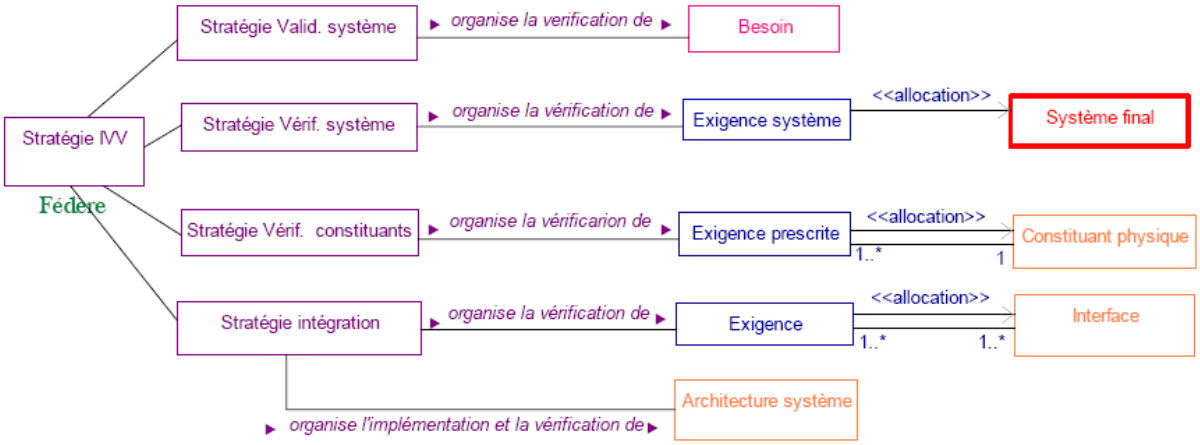


Figure 3.20. : Vue AFIS : Intégration, Validation, Vérification

Ici, le modèle AFIS permet de définir une procédure IVV, qui inclut explicitement un exemplaire physique du système et un environnement de test (figures 2.19 et 2.20).

Dans le chapitre III, nous proposerons une évolution de ce modèle de données AFIS, en le spécialisant en fonction des objectifs et de la méthodologie définie dans cette thèse. De par la méthode et le moyen utilisé pour spécialiser ce modèle de données, notre modèle de données reste compatible avec le modèle de données d'origine de l'AFIS et donc conforme à l'objectif de ce modèle de données, à savoir **bâtir une base conceptuelle de référence, pour les échanges** entre industriels avec les organismes de standardisation ou de normalisation et les éditeurs d'outils [Afi].

3.5. Conclusion

Ce chapitre a permis d'introduire la modélisation comme un outil efficace de l'IS. Une première classification des modèles basée sur la typologie de l'AFIS a été présentée. Ensuite, une description des principales méthodologies d'IS dirigées par les modèles a permis de montrer comment des modèles peuvent être enchaînés de manière complémentaire et conformément à un cycle de développement, afin de progresser dans l'analyse, la spécification et la conception d'un système ou de son architecture.

En particulier, il est apparu que la formalisation du besoin, des exigences et des solutions apportée par ces modèles pourrait être mieux utilisée en améliorant l'interopérabilité des différents modèles développés. Un état de l'art des travaux de recherche visant à mieux exploiter cette formalisation et son exploitation par la machine, notamment par l'application de l'approche MDA au niveau global de l'IS a donc été réalisé.

La méthodologie présentée dans le chapitre quatre et son application au chapitre cinq poursuit ce même objectif et tente de progresser par rapport aux limites identifiées à l'issu de cet état de l'art. Elle propose notamment d'établir un cadre plus large à la modélisation en IS, en s'appuyant sur des normes et standard existants, puis sur un langage de modélisation adapté.

Chapitre 4. Méthodologie pour l'Ingénierie de Conception Conjointe de Systèmes Aéronautiques (MICCSA)

Concepts de base et processus

Dans ce chapitre, sont présentés les principaux concepts de la méthodologie d'ingénierie système résultant de notre travail de thèse. Cette méthodologie tient compte, d'une part, des particularités liées au développement dans le domaine aéronautique, et d'autre part des spécificités de la pratique actuelle d'ingénierie système à LATECOERE. A ce titre, le terme *Conception Conjointe* a pris dans ce travail une connotation *hardware/software* étendue, notamment au niveau matériel, à l'électronique bien sûr, mais également à l'électricité et à la mécanique qui sont encore plus prépondérantes que l'électronique et le logiciel dans les systèmes envisagés.

Le sous-chapitre 4.1 présente tout d'abord les axes d'amélioration attendus par l'usage de la méthodologie. Il présente ensuite l'organisation de celle-ci et décrit les processus techniques qui ont été définis et autour desquels sont organisés les éléments de méthodes.

Dans la méthodologie MICCSA, les concepts manipulés sont regroupés et organisés autour et avec un modèle d'informations unique. Ces concepts sont présentés au sous-chapitre 3.2 sous forme d'un modèle entités-relations.

Le sous-chapitre 4.3 présente les éléments permettant d'organiser et d'exploiter le modèle projet conformément aux principes de l'ingénierie système dirigée par les modèles. Ce modèle projet est instancié à partir du modèle d'informations présenté précédemment.

Enfin, le sous-chapitre 4.4 décrit la mise en application des processus techniques et des méthodes sur un modèle projet. L'application des principes décrits permet notamment la génération automatique d'éléments de modèle au travers de l'utilisation de patrons de modélisation.

4.1. Contexte et description de la méthodologie

4.1.1. Contexte de la méthodologie et points spécifiques

La méthodologie est destinée à faciliter l'adoption d'une approche système commune aux aspects aérostructure et électronique embarqué. Cette approche système est déclinée sur un ensemble de processus techniques et de méthodes, basées sur :

- la modélisation des exigences,
- la modélisation des solutions techniques,
- la formalisation, par une démarche d'ingénierie de modèles, des éléments intermédiaires d'intégration, de validation, d'évaluation et d'analyse de la conception.

Dans ce cadre, l'application de la méthodologie permettra d'améliorer la pratique actuelle d'ingénierie système de LATECOERE sur trois axes principaux impliqués dans le développement des projets, et présentés ci-dessous. Ces axes d'amélioration ont été identifiés suite à la démarche de retour d'expérience réalisée au moyen de questionnaires soumis à différents services de l'entreprise comme déjà évoqué au chapitre II.

Tout d'abord, un axe principal d'amélioration identifié est la conduite des activités :

- d'identification des exigences sources applicables à un projet, aux différents niveaux de développement,
- de maîtrise du flot d'exigences décliné à partir de celles-ci,
- de maitrise des évolutions de cet ensemble d'exigences et de répercussion de celles-ci sur les projets et sur les processus associés.

Ces ensembles d'exigences applicables sont hétérogènes et diverses en nature, niveau d'abstraction et de priorité. Elles incluent notamment les normes réglementaires en vigueur (textes de certification, normes DO-178 / DO-254), les référentiels d'exigences des clients (GRESS, GRAMS, D6) et les exigences produits et processus spécifiques (voir figure 4.1). Chacune de ces exigences est susceptible d'évoluer et les référentiels d'exigences associés à chaque projet doivent être mis à jour. De plus, l'impact de ces évolutions sur la définition des produits doit pouvoir être identifié et analysé.

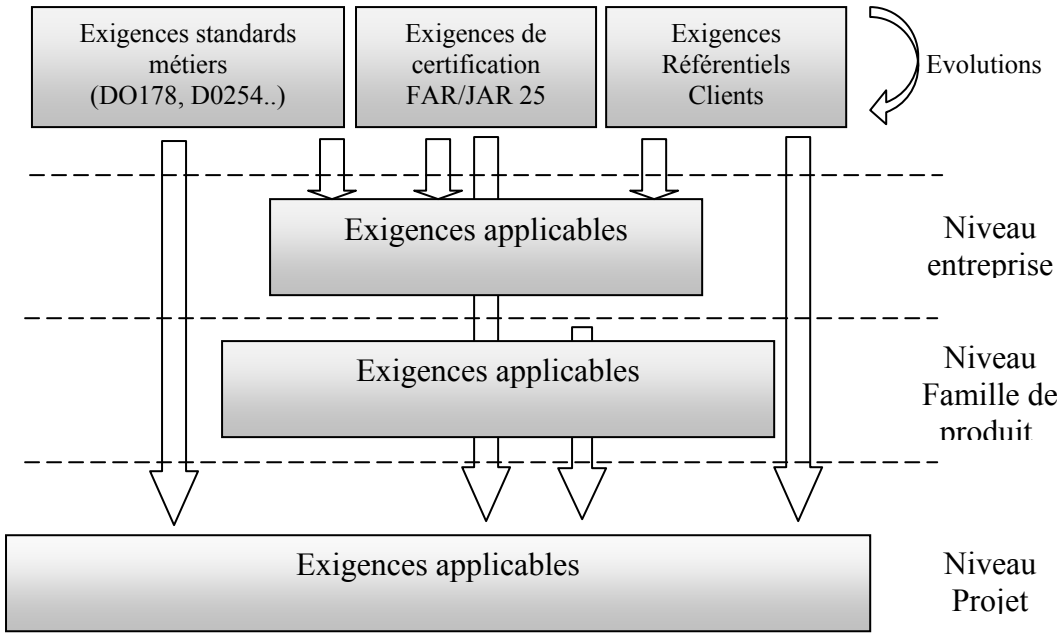


Figure 4.1. : Applicabilité des exigences sources à un projet

Dans cet objectif, la méthodologie s'appuie sur la formalisation de ces exigences et sur la formalisation des modèles descriptifs des solutions techniques. Cette approche permet d'évaluer automatiquement l'applicabilité des exigences sources et d'analyser l'impact de leur évolution sur la définition d'un produit.

Un autre axe d'amélioration identifié est lié à la phase d'analyse et de comparaison des solutions techniques en phase de conception préliminaire. Rappelons que l'un des objectifs est de faciliter l'exploration de l'espace des solutions techniques lors des étapes de conception architecturale, d'allocation et de dimensionnement des constituants. La figure 4.2 trace les grandes lignes de la recherche et de l'intégration de solutions.

Pour cela, la méthodologie tire parti de l'utilisation de modèles semi-formels pour assister la recherche et la définition de solutions de conception. Elle définit une sémantique commune pour les éléments architecturaux des domaines mécaniques, électriques, électroniques et logiciels sur des niveaux d'abstractions adaptés.

A partir de cette sémantique commune, elle facilite d'une part la recherche de composants d'architecture adéquats pour répondre aux exigences, et d'autre part leur intégration dans l'architecture. Dans ce contexte, la prise en compte des modèles d'exigences est aussi nécessaire que celle des modèles de solutions, afin d'évaluer l'impact potentiel d'une décision techniques ou d'une évolution locale sur la définition du système et sur les exigences spécifiées.

Les améliorations attendues concernent à la fois la maîtrise de la conception et les évolutions du modèle de solution architecturale dans les phases préliminaires de développement (par exemple lors d'une évolution d'interface) mais aussi dans la phase d'exploitation opérationnelle des produits (par exemple lors d'une modification liée à l'obsolescence d'un composant).

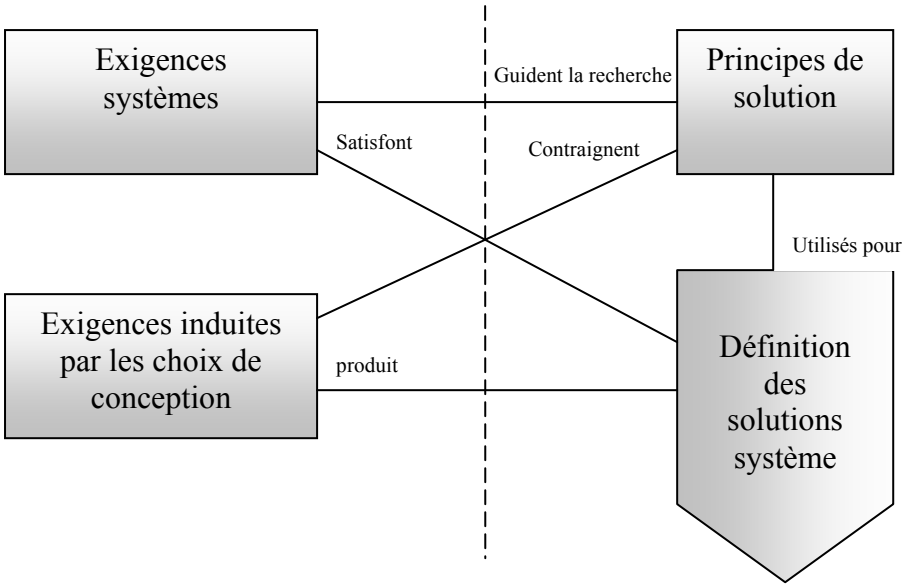


Figure 4.2. : Recherche et intégration de principes de solutions

Enfin, un dernier axe spécifiquement étudié dans la méthodologie est l'activité de validation des exigences système, réalisée lors de la phase descendante de conception, de spécification des exigences et de définition de l'architecture système.

De nombreux dépassements de coûts et de délais sur les projets sont imputables à une activité de validation défaillante ou inadaptée en termes de périmètre, de niveau de détail (*profondeur* de validation) ou de moyen utilisé. La méthodologie apporte une contribution sur l'amélioration de la pratique actuelle des activités de validation des exigences. Cette amélioration concerne plus généralement l'ingénierie des exigences liée au développement d'un produit, depuis l'identification des exigences sources, jusqu'à la spécification détaillée d'un sous-ensemble.

Pour atteindre cet objectif, l'approche utilisée consiste à valider les modèles d'exigences et de solutions (spécifiées par des exigences) par la vérification de règles, et à faciliter la conception de modèles validés par l'application de patrons de modélisation. Ces derniers permettent de compléter et de corriger le modèle en y ajoutant ou en y modifiant des éléments.

Afin de lever les verrous présentés ci-dessus, la démarche employée est basée sur une approche MBSE dont les principes de base ont été introduits au chapitre II. Elle s'appuie sur l'élaboration, l'enchaînement et la transformation de modèles qui permettent la réalisation des processus d'IS tels que spécifiés par les exigences de la norme EIA 632.

La méthodologie se conforme également à l'approche MDA [MDA_03]. La portabilité, l'interopérabilité et la réutilisabilité sont les premiers objectifs d'une telle démarche. La méthodologie MICCSA est basée sur l'hypothèse selon laquelle **les objectifs et les principes de MDA appliqués au développement logiciel peuvent être transposés à l'ingénierie des systèmes** moyennant quelques adaptations et/ou aménagements. Pour effectuer cette transposition, une condition préalable est que la méthodologie s'appuie sur des modèles formalisés pour la représentation des exigences, des éléments existants et du système produit.

4.1.2. Structure et organisation

La méthodologie est organisée autour d'un cycle de développement et d'un ensemble de processus techniques. Ces derniers s'appuient, au niveau méthode, sur l'utilisation de modèles du produit tout au long du développement.

4.1.2.1. Cycle de développement

Le cycle de développement adopté est le cycle en V, traditionnellement appliqué par la société LATECOERE et dans l'ensemble de l'industrie aéronautique. Dans le contexte du développement de système, ce cycle est structuré par des jalons et par la production de documents techniques de spécification et de définition du système (SRD - *System Requirements Document* - et SDD – *System Definition Document*), puis de spécification de ses sous-ensembles (PTS - *Purchaser Technical Specification*). La figure 3.3 présente une vue générale du cycle de développement de la méthodologie et décrit le périmètre couvert par celle-ci. Alors que les branches classiques du V représentent la conduite des tâches basées sur les documents (*document-oriented*), une seconde branche a été adjointe à gauche afin de représenter les activités d'élaboration des modèles produits. Ces modèles permettent d'engendrer les documents de spécification et de définition.

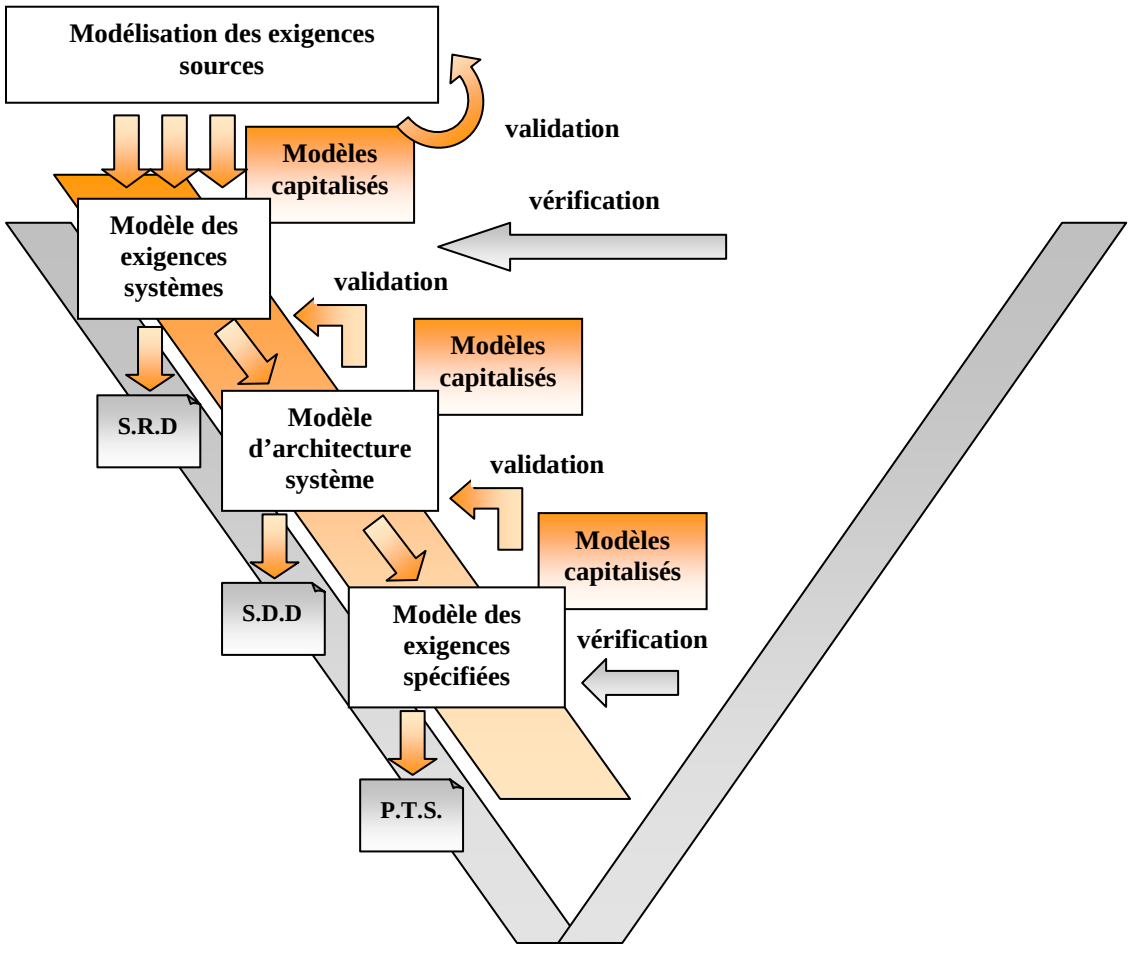


Figure 4.3. : Description générale de la méthodologie

La démarche débute par la définition et la validation d'un modèle des exigences sources du projet. Par la suite, une phase d'ingénierie des exigences permet d'établir un modèle représentant le référentiel d'exigences système validé, permettant ensuite la conception d'une architecture système. Une fois le modèle d'architecture système validé, les exigences liées aux constituants de l'architecture sont spécifiées et validées.

4.1.2.2. Processus techniques d'ingénierie système

La réalisation du Cycle de développement de la figure 4.3, se fait conformément à des processus techniques spécifiques à la méthodologie, représentés en figure 4.4. Ces processus permettent d'identifier les activités à mener et les informations consommées et produites par celles-ci.

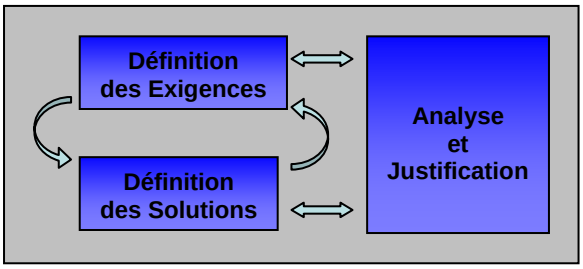


Figure 4.4. : Vue générale des processus de la méthodologie MICCSA

4.1.2.3. Processus de définition des exigences

Le processus de définition des exigences est utilisé pour la capture des exigences et la construction des spécifications d'exigences. Ce processus repose principalement sur :

- les décisions sur l'application des exigences source,
- la définition des exigences techniques,
- l'établissement de compromis dans les spécifications d'exigence.

Le processus d'analyse et de justification est utilisé en support afin de fournir une information complète et correcte pour la prise de ces décisions, ainsi que pour leur justification et leur capitalisation.

4.1.2.4. Processus de définition des solutions

Le processus de définition des solutions permet de définir les modèles de solutions de conception à différents niveaux d'abstraction tels que spécifiés dans la section 3.2. Ce processus repose principalement sur :

- la sélection des solutions techniques à mettre en œuvre,
- les décisions de compromis dans le choix des constituants,
- la résolution des problèmes d'intégration entre les constituants,
- la définition de solutions systèmes.

Le processus d'analyse et de justification est utilisé ici afin de fournir une information complète et correcte, pour l'évaluation des solutions existantes, ainsi que pour la justification et la capitalisation des décisions.

4.1.2.5. Processus d'analyse et de justification

Le processus d'analyse et de justification est un processus ayant un rôle :

- de support des processus décrits ci-dessus,
- d'interface avec les activités spécifiques de qualification et de certification,
- de justification des décisions et de capitalisation de l'expérience acquise.

Dans son rôle de support des processus de définition des exigences et des solutions, le processus d'analyse permet d'améliorer l'exactitude et la complétude des analyses réalisées au cours du développement. Le processus d'analyse est utilisé pour valider les exigences définies dans le processus de définition correspondant. Dans le contexte de la conception, le processus d'analyse est utilisé pour la recherche des solutions techniques à chaque niveau d'abstraction (démarche d'analyse de type *bottom-up*) ainsi que pour l'analyse des systèmes existant en interne et des systèmes compétiteurs (démarche d'analyse de l'existant basée *reverse engineering*). Ces analyses sont ensuite utilisées par les processus de définition afin de guider les décisions techniques au cours du développement.

Dans son rôle d'interface, le processus d'analyse permet, par son aspect systématique, d'évaluer l'applicabilité des exigences sources aux différents niveaux de conception, d'allouer des exigences de processus (ex : qualification & certification) aux exigences produits et aux exigences processus déclinées (en dehors du périmètre du modèle produit).

Enfin, le processus d'analyse sert également à réaliser les interfaces avec les éléments de justification de la conception et permet de formaliser la connaissance et les décisions (décisions de spécification et décisions de conception) afin de capitaliser l'expérience acquise.

Les processus introduits ci-dessus sont implémentés et présentés sous la forme d'un enchaînement de modèles, (ou *model workflow*), adaptés à l'ingénierie système dans laquelle l'entreprise a inséré ses spécificités.

Afin d'assurer la cohérence des descriptions à travers chacun des processus, l'ensemble des concepts manipulés sont organisés au sein d'un modèle d'informations unique. Celui-ci est présenté dans le sous-chapitre suivant.

4.2. Modèle d'informations de la méthodologie

Les concepts importants de la méthodologie (systèmes, équipements aéronautiques et éléments de structure) sont présentés dans le modèle d'informations. Les éléments du modèle d'informations sont instanciés dans le contexte d'un projet particulier par la réalisation des processus techniques décrit au sous-chapitre 4.4.

L'approche adoptée est une approche de modélisation commune à deux familles de produits travaillés actuellement dans l'entreprise LATECOERE. Il s'agit des produits de type équipements d'une part et les produits dits d'aérostructure d'autre part. Ces deux familles sont regroupées dans le cadre du développement du système porte passagers, présenté au chapitre cinq.

Par souci de simplification et d'aide à la compréhension, le modèle entité-relations de la méthodologie est présenté par fragments séparés.

4.2.1. Description générale du modèle d'informations

Ce modèle d'informations inclut les concepts et les éléments nécessaires permettant d'implémenter les processus techniques dans un contexte d'ingénierie dirigée par les modèles.

Le modèle d'informations comporte **un ensemble normatif** organisé autour d'un bloc de construction qui constitue la **description des référentiels systèmes** à un niveau de décomposition donné. Il s'agit des référentiels d'exigences et des référentiels de définition (de solutions) du système. Cette partie du modèle constitue la référence du système tout au long du projet. Elle est le support des activités d'ingénierie système en phase de développement et pendant la vie du système. Cette partie centrale permet, entre autres, d'engendrer les documents de spécification et de définition du système (documents SRD et SDD). Les modifications apportées aux modèles référentiels (exigences des parties prenantes, exigences système, architecture logique et physique) se font via un mécanisme de modification conforme aux règles de gestion du changement.

Le modèle permet également de définir des **modèles intermédiaires ou des extraits** de modèle préexistants de ce modèle système à des fins **d'analyse** pour la conception, la validation, la vérification (dans un contexte, une préoccupation et un point de vue particulier). Ces modèles sont appelés modèles de conception et modèle d'analyse. Ils permettent l'évaluation, la validation, la vérification du système ou de la partie du système qui fait l'objet de l'étude. Ils contiennent les éléments qui sont analysés, la formalisation du périmètre de l'analyse, les éléments de raisonnement, les éléments support et tout autre élément qui doit être capitalisé tout au long du projet et à l'échelle de l'entreprise. Ils permettent d'engendrer les documents de synthèse des efforts d'analyse de compromis, de validation et de vérification.

Modèles constituants des blocs de construction

La partie du modèle décrite par la figure 4.5 permet de regrouper les modèles constitutifs dans un bloc de construction défini pour un périmètre et un niveau d'abstraction. Ceux-ci sont regroupés en *RequirementSet* et en *Logical / PhysicalSolutionRepresentation*.

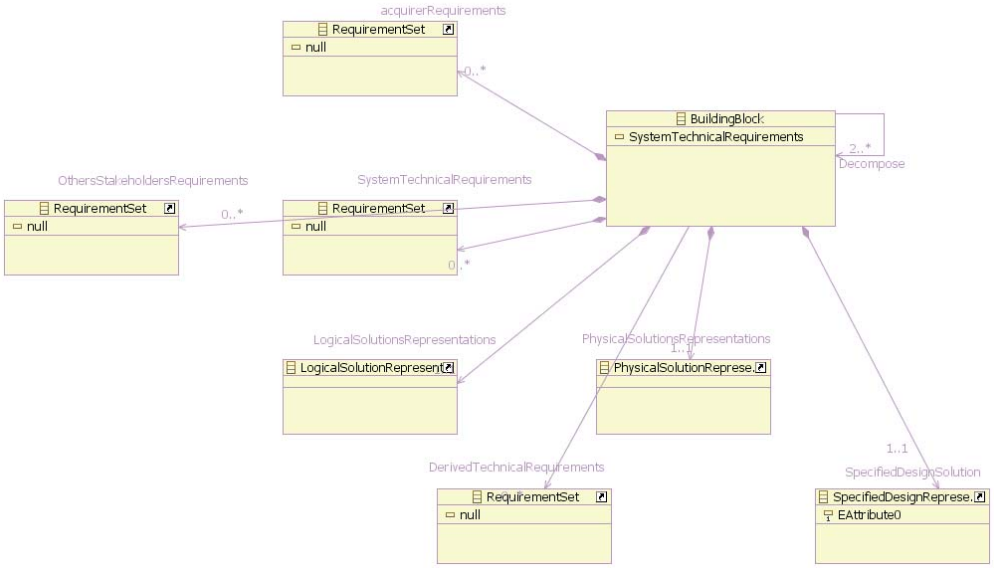


Figure 4.5. : Eléments constitutifs du building block dans le modèle d'informations MICCSA

4.2.2. Principaux éléments du modèle

4.2.2.1. Données d'ingénierie

L'élément de base du modèle est la donnée d'ingénierie. Elle est modélisée par le diagramme de la figure 4.6. Elle correspond à un élément qui est utilisé ou produit par une activité d'ingénierie système. Son rôle est de :

- Relier l'information associée à toute donnée d'ingénierie pendant la réalisation des processus d'IS et de définir un propriétaire de l'élément.
- Assurer l'interopérabilité des modèles d'ingénierie système et formaliser les relations entre les éléments d'ingénierie en interaction pendant les processus d'IS.
- Assurer la cohérence des référentiels de données d'IS pendant la réalisation des processus.
- Formaliser l'évolution de ces référentiels, en associant à toute donnée d'ingénierie un cycle de vie, défini différemment selon le type concret de donnée d'ingénierie : exigences système, constituant logique, …

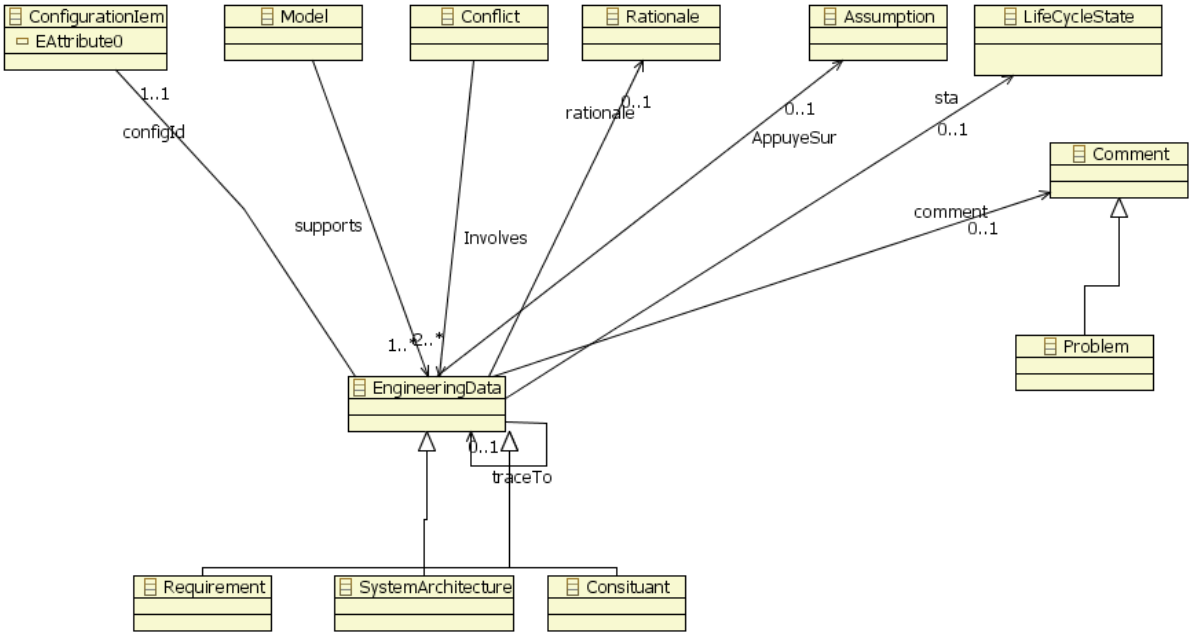


Figure 4.6. : Données d'ingénierie dans le modèle d'informations MICCSA

Une donnée d'ingénierie peut être associée à :

- une hypothèse, donnée utilisée par la donnée d'ingénierie mais non prouvée,
- un élément de justification permettant d'expliciter un choix associé à la donnée d'ingénierie,
- un identifiant de configuration permettant d'intégrer la donnée d'ingénierie dans un référentiel (référentiel d'exigence ou de solution).
- un modèle (ex : modèles associés aux outils d'ingénierie / de conception assistée par ordinateur CATIA, ADAMS, SPICE), ou la définition de résultats (ex : de calcul de modèle è éléments finis) fournis en support de la donnée d'ingénierie.

Les exigences, les architectures systèmes ainsi que leurs constituants sont, par ailleurs, considérés comme des données d'ingénierie.

4.2.2.2. Modèle d'Exigences

Ce modèle permet de modéliser formellement le lien entre exigences et éléments descriptifs des solutions. Une exigence étant une donnée d'ingénierie, elle est gérée en configuration et s'intègre dans un référentiel système.

La définition du modèle d'exigences inclut un ensemble d'attributs applicables à toute exigence du projet et un ensemble d'attributs spécifiques dépendant de la catégorie à laquelle elle appartient. Ces attributs permettent de mettre en œuvre la vérification de l'exigence au moyen de leurs évaluations, ces dernières jouant le rôle de propriétés pour les modèles descriptifs de solutions.

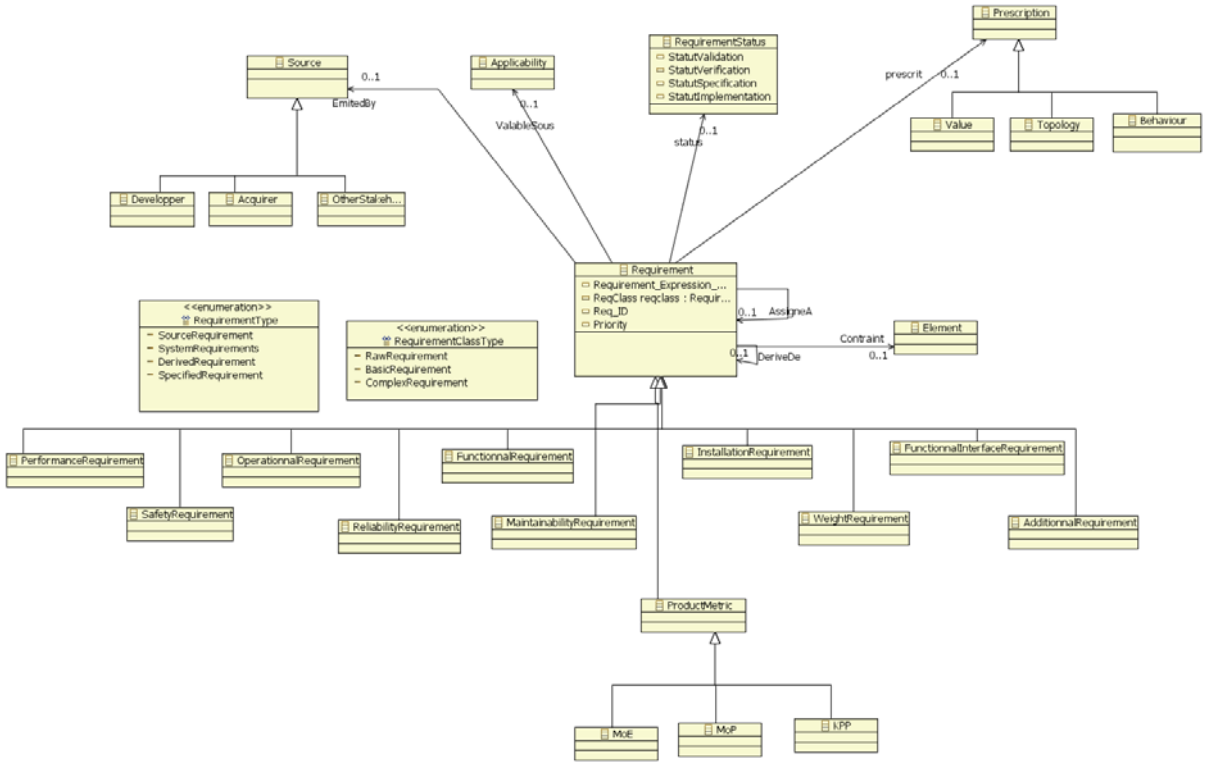


Figure 4.7 : Eléments d'exigences

Les éléments génériques d'exigences sont les suivants (voie figure 4.7) :

- Une **source** ou émetteur. Les sources définies sont l'acquéreur, les autres parties prenantes, et les développeurs. Une exigence est contenue dans une spécification, et appartient à une catégorie d'exigence. Par ailleurs, une exigence possède une relation avec les métriques techniques du projet.

- Une **formulation textuelle**. Une exigence est *brute* si la conformité de cet attribut aux règles générales d'écriture d'exigence n'a pas été évaluée. Les exigences *élémentaires* sont des exigences conformes aux règles d'écriture qui ne peuvent être décomposées sans perte d'information. Enfin, Les exigences *complexes* sont des exigences conformes aux règles d'écriture. Elles contiennent plusieurs exigences élémentaires et peuvent donc être décomposées.

- La conformité de l'objet à cette propriété est jugée nécessaire par la source ou l'émetteur de l'exigence, au moyen d'un éventuel **élément de justification.**

- Le degré de nécessité pour que l'objet soit conforme à sa prescription est défini par un niveau de **priorité**. La définition d'un niveau de priorité pour une exigence est une étape de l'ingénierie des exigences.

4.2.2.3. Eléments d'exigences formalisés pour la V&V

Une exigence possède **un objet** sur lequel s'applique la prescription de l'exigence. C'est sur cet objet que peut être évaluée la satisfaction de l'exigence par le modèle descriptif de

solution. L'objet de l'exigence est un autre élément ou ensemble d'éléments identifié(s) dans le modèle. La définition de l'objet d'une exigence fait partie des activités de mise en relation des modèles prescriptifs et constructifs, et constitue une étape d'ingénierie des exigences.

L'objet des exigences dans la méthodologie sont des éléments d'architectures opérationnelles, logiques ou physiques du système.

- Une exigence possède également un **élément prescriptif, ou prescription** qui constitue une formalisation de celle-ci dans un langage compréhensible par la machine et évaluable sur le modèle constructif. Dans le cas ou elle ne peut être évaluée, le modèle constructif est incomplet au regard de cet élément prescriptif, et donc vis-à-vis de l'exigence qui le contient. Dans le cas où il peut être évaluée, son évaluation fournit un résultat sous forme d'une information binaire de type {conforme, non conforme}, ou un écart quantitatif par rapport à un objectif quantitatif prescrit.

Selon le type de propriété (cf. §3.2.1.1.) prescrit, les éléments objets et prescription d'une exigence sont conformes au tableau ci-dessous. Les éléments de prescription peuvent prendre principalement trois formes :

- prescription inclusive ou exclusive sur le type, le nom, la structure interne, d'un constituant ou d'un ensemble de constituants d'architecture : composant ou constituant d'interface, concepts de solution, descriptions de comportements, moyens de réalisation (voir sections 4.2.3. sur la description des architectures),

- prescription inclusive ou exclusive sur la topologie des connexions entre constituants d'une architecture d'un sous-ensemble d'architecture (concepts de solution, moyens de réalisation),

- prescription inclusive ou exclusive sur la topologie de l'allocation des constituants d'une architecture sur les constituants d'une autre.

Les catégories d'exigences systèmes présentées en annexe B couvrent l'ensemble des prescriptions qui peuvent être formulées sur un modèle de type entité-relations-attributs.

Une exigence possède un cycle de vie spécifique. Le cycle de vie est basé sur la complétude des éléments appartenant à sa description et la complétude de sa formalisation en lien avec les modèles descriptifs de solution.

L'élément exigence possède également deux attributs concernant respectivement le statut de validation et de vérification.

Les éléments du modèle d'exigence permettent de qualifier les exigences conformément au besoin de la méthodologie :

- Une exigence qui possède un élément de justification associé à un élément de type constituant d'architecture est une exigence dérivée technique (découlant d'un choix de conception).
- Une exigence est une exigence spécifiée si elle est associée à la solution de conception, et/ou une exigence attribuée (ou affectée) si elle est attribuée au modèle

prescriptif d'un bloc de construction aval dans la conception (de niveau supérieur dans la hiérarchie de bloc de construction).

4.2.2.4. Eléments d'architecture

Le modèle d'informations inclut les éléments de description des représentations des solutions logiques et physiques (voir figure 4.8). Pour répondre au besoin spécifique de Latécoère, il permet de décrire, sur plusieurs niveaux d'abstractions adaptés, les concepts de solution. Ceux-ci sont définis et évalués, puis utilisés afin de décrire les architectures logiques et physiques qui constituent le référentiel système.

Le type *Architecture Système* est un type commun aux architectures logiques et physiques. Celles-ci sont composées de constituants génériques ainsi que d'éléments d'ingénierie spécialisée.

Par ailleurs, chaque constituant peut être un constituant de l'environnement. Ainsi, tout constituant décrit dans l'architecture peut être défini comme appartenant à l'environnement du système étudié, mais intégré à la description de l'architecture. Les constituants n'appartenant pas à l'environnement appartiennent au périmètre du système.

Un constituant, qu'il soit logique ou physique, est un élément descriptif du produit et doit être géré en configuration.

Les éléments d'ingénierie spécialisée présentés ici correspondent :

- Aux éléments issus des analyses de sûreté de fonctionnement prescrits par l'ARP4754 tels que réalisés par Latécoère.
- Les éléments d'analyse de maintenabilité des architectures.

Les constituants, logiques et physiques sont les éléments de base des architectures. Certains constituants sont identifiés comme des éléments d'interfaces et relient entre eux au moins deux constituants.

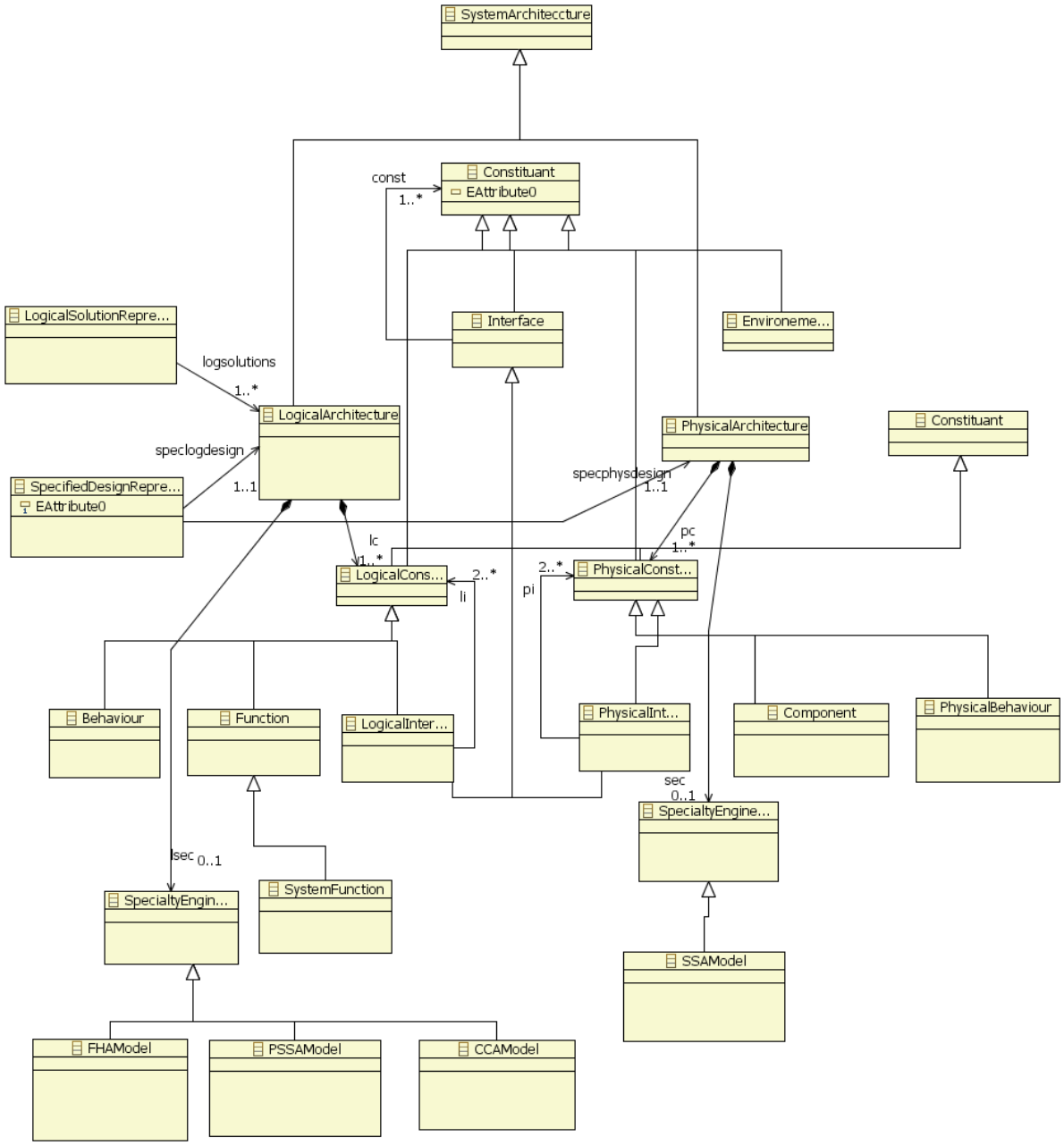


Figure 4.8.: Eléments d'architectures logiques et physiques

4.2.2.5. Eléments d'architecture logique

Un constituant logique peut être une fonction (élément statique de l'architecture logique) ou un élément de comportement (élément dynamique de l'architecture logique).

Les fonctions décrivent un effet attendu d'une partie du système comme un constituant d'architecture, interfacé avec d'autres fonctions pour réaliser, à un niveau global, les fonctions initiales, et finalement la mission.

Pour les besoins de la méthodologie, plusieurs niveaux d'abstractions ont été définis. Le niveau le plus abstrait consiste à caractériser le système comme un réseau de fonctions de nœud et de zones courantes.

A un niveau supérieur, les fonctions de nœuds sont caractérisées comme des constituants d'interface, de transformation ou de stockage, tandis que les fonctions de zones courantes sont associées aux fonctions de communication. Les flux échangés dans ce réseau sont des flux de matières, d'énergie et/ou d'information.

Un autre niveau d'abstraction a été défini pour identifier des fonctions techniques plus spécifiques. Ces fonctions complexes peuvent être vues comme des combinaisons des fonctions élémentaires présentées ci-dessus. Elles peuvent être rattachées aux principes de solution mis en œuvre. Ainsi, des fonctions telles que : *assister le réglage*, *permettre l'étalonnage*, *permettre l'autotest*, *assurer le conditionnement de signal*, peuvent être associées aux éléments capteurs dans une architecture physique, tandis que des fonctions de type *transmettre effort mécanique*, *assurer le retour de position angulaire*, *transformer mouvement translation en rotation,* peuvent être associées à des sous-ensembles mécaniques..

Enfin, au niveau le plus élevé, se trouvent les fonctions appelées fonctions de services système. Celles-ci correspondent aux fonctions opérationnelles décrivant le besoin du client. Le principe de distinction stricte entre domaine du problème et domaine de la solution est justifié par le fait que certaines fonctions font partie du domaine du problème (celle identifiée comme services système dans le modèle de référence) et d'autres font partie de la solution.

Les fonctions identifiées comme services système dans le modèle de données incluent des métriques d'efficacité MoE qui font partie intégrante du domaine du problème. L'activité de distinction entre fonctions identifiées comme du domaine du problème et celle du domaine de la solution est une activité d'élicitation des besoins et des exigences.

Les comportements représentent la dynamique de l'architecture logique. Ils sont modélisés au moyen de diagrammes états-transitions.

Par rapport au modèle de données développé par l'AFIS [AFI] présenté au chapitre II, la méthodologie MICCSA inclut une vue logique à base de constituants logiques. Cette approche permet de réunir les éléments *constituants logiques* et *fonctions,* du modèle de l'AFIS, dans le modèle de la vue architecture système. Par ailleurs, l'élément *service système* sera compris comme des fonctions opérationnelles ou fonctions de *haut niveau* du système. Les fonctions de haut niveau sont définies par les parties prenantes et sont issues d'une phase d'analyse et non de conception.

4.2.2.6. Eléments d'architecture physique

Un constituant physique peut être un composant (définition physique statique du composant), ou un comportement physique (modèle dynamique physique). Comme pour l'architecture logique, plusieurs niveaux d'abstraction sont définis pour l'architecture physique.

Le niveau le plus abstrait consiste à identifier une architecture de concepts techniques de solutions. Ces éléments sont représentatifs des choix techniques principaux faits en début de projet. (Exemple : structure métallique ou composite.)

Un niveau suivant précise les choix techniques d'architecture du produit. Il précise généralement comment les fonctions de haut niveau sont projetées sur l'architecture technique du produit.

Un niveau suivant identifie les principes constructifs sélectionnés. Ils sont l'équivalent physique des fonctions techniques de l'architecture logique. Les principes constructifs peuvent être déclinés.

4.2.2.7. Eléments d'interface sur les architectures

Les interfaces sont des constituants d'architectures qui possèdent certaines caractéristiques spécifiques. Ainsi, toute interface relie au minimum deux constituants de l'architecture. Les interfaces principales d'un système concernent :

- Les interfaces externes du système avec d'autres systèmes, la structure, l'environnement et les outils de support au sol.
- Les interfaces logiques entre les fonctions partagées avec d'autres systèmes.
- Les interfaces physiques et logiques avec les ressources partagées (alimentation en puissance, bus de données).
- Les interfaces pour l'équipage et les opérateurs de maintenance,
- Les interfaces internes entre constituants du système.

La nature des interfaces est également spécifiée (électronique, électrique, fluidique, mécanique, Optique, Homme-machine, etc.). De plus, les caractéristiques concernant une information échangée peuvent être spécifiées, telles que la nature de l'information, le format, et la dynamique de l'échange de l'information.

Le modèle inclut également une description d'**architecture opérationnelle** (figure 4.9). L'architecture opérationnelle inclut la description de l'environnement du système et la description des concepts opérationnels et des scenarios (ou profils opérationnels). La description opérationnelle mène à l'identification des mesures d'efficacité qui seront intégrées au référentiel d'exigences système et déclinées pendant le développement. La relation entre architecture opérationnelle et les architectures logiques et physiques est détaillée dans la suite du manuscrit.

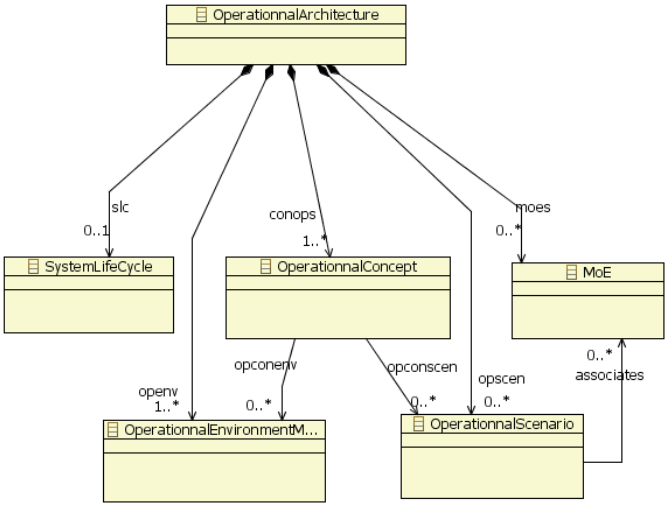


Figure 4.9. : Eléments d'architecture opérationnelle.

4.2.2.8. Eléments partiels d'architecture

Des modèles de **concepts de solutions** représentent des éléments de solutions partielles modélisées individuellement. Ils sont divisés en deux catégories dans le modèle d'informations. Ce dernier comporte, d'une part les modèles de concepts de solution opérationnels qui héritent de l'entité architecture opérationnelle et constituent des éléments de modèles cognitifs et d'autre part, les modèles de concepts de solutions système qui héritent de l'entité architecture système et constituent des éléments de modèles normatifs constructifs (figure 4.10). Intégrés dans une architecture, un concept de solution est caractérisé par :

- son adéquation aux prescriptions affectées à l'architecture,
- les interfaces internes de l'architecture impliquées dans l'intégration,
- l'impact (ou le coût) de son intégration, formalisé par les exigences dérivées du concept de solution et de son intégration.

Des propriétés provenant des modèles prescriptifs sont évaluées sur ces éléments de modèles constructifs.

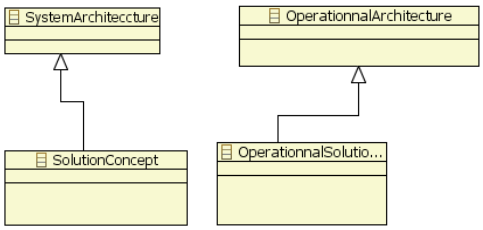


Figure 4.10.:Les concepts de solutions héritent des éléments d'architecture système et d'architecture opérationnelle.

Les modèles de concepts de solution permettent de représenter les solutions logiques et physiques envisagées lors des projets. Ils sont regroupés dans des bases de données de retour d'expérience organisées de manière matricielle par domaine de solutions (mécanique, hydraulique, pneumatique, électrique, logiciel), par type de moyen de réalisation (moyen de

transformation, moyen de communication, de stockage, d'alimentation, d'interfaçage) ou selon des critères spécifiques. Un concept de solution est importé dans une architecture solution, des propriétés spécifiques lui sont attribuées afin de l'intégrer au reste de l'architecture.

Par ailleurs, un sous-ensemble d'architectures physiques peut constituer un **moyen de réalisation**. Un moyen de réalisation est un ensemble de constituants qui permet de réaliser une fonction particulière. L'intérêt de définir des tels sous-ensembles est :

- vérifier l'existence d'un moyen de réalisation pour s'assurer de la réalisation de la fonction,
- vérifier certaines propriétés associées au moyen de réalisation. Par exemple, son périmètre par rapport à l'architecture complète, ses interfaces, sa topologie interne, sa relation avec un autre moyen de réalisation ou avec des éléments spécifiques de l'architecture.

Ces éléments sont plus détaillés dans la section 4.3.1 traitant l'ingénierie des modèles pour la définition des exigences.

4.2.3. Eléments support des activités d'Analyse et de V&V.

Cette partie du modèle permet d'articuler les activités IS réalisées autour d'instances du modèle d'informations. Selon l'activité d'IS considérée (analyse, validation, définition) et le(s) élément(s) de modèle en entrée de l'activité (ex : une exigence) celle-ci est décomposée en objectifs, en moyens, et en résultats. Les moyens sont exprimés en termes de Tâches élémentaires d'Ingénierie du Modèles (T.I.M.). Ces éléments objectifs, moyens et résultats sont présentés de manière synthétique sur les vues du modèle. Ils sont une représentation graphique de la couverture du processus IS correspondant.

4.2.3.1. Eléments support pour l'analyse système et la validation

Les éléments du modèle d'informations présentés permettent de conduire les activités d'ingénierie système et d'alimenter les vues d'analyse du modèle. Ces éléments permettent de lier les modèles prédictifs à la partie normative du modèle. Les différents types d'éléments sont répertoriés dans le diagramme de la figure 4.11.

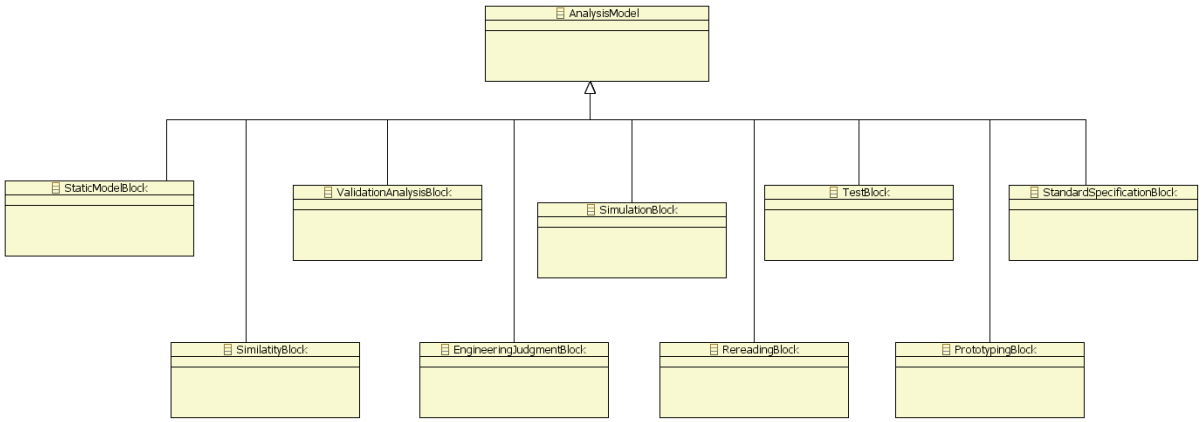


Figure 4.11. : Types de modèles d'analyse et de validation

Un modèle d'analyse consiste en un ensemble d'éléments susceptibles de produire le résultat d'analyse ou de validation attendu, par la réalisation d'un ensemble de Tâches d'Ingénierie de Modèles (T.I.M.). Les TIM sont définis à partir des activités d'analyse système et de validation. Ces dernières sont détaillées dans les sections suivantes.

Par ailleurs, le modèle d'analyse et de validation est utilisé comme élément de justification lors de la réalisation des documents de conception et de validation de la définition.

Un modèle d'analyse et de validation peut contenir des exigences, des constituants et des concepts de solutions, mais il peut inclure tout élément de formalisme quelconque (ex : un plan Catia™, une feuille de calcul Excel™, une trace de simulation Matlab™, ou des résultats de calculs). La seule contrainte est que cet élément soit désigné à l'intérieur du bloc et que ce bloc possède comme élément de sortie les informations permettant de fournir les résultats conformes aux objectifs de validation définis dans l'activité de validation.

4.2.3.2. Eléments support spécifique pour la validation

La partie du modèle de la figure 3.12 permet de modéliser les activités liées à la validation des exigences (cf. exigences EIA 25, 26, 27, 28). Les éléments principaux de cette partie importante du modèle de MICCSA sont décrits en suivant.

Le plan de validation. La définition d'un plan de validation permet de préciser la stratégie globale de validation ainsi que les activités planifiées pour la validation des exigences à un instant donné du développement système. La stratégie globale consiste à améliorer l'efficacité des activités de validation en réduisant les ressources nécessaire pour la réalisation de ces activités. Dans la méthodologie, cette stratégie est exprimée comme un ensemble de règles. Les règles de stratégies permettent, d'assurer la qualité de l'activité de validation, d'optimiser et d'adapter l'effort de validation aux spécificités du projet (risques, expériences acquises, introduction d'innovation, criticité du projet).

Les règles de stratégie qui peuvent être définies sont de plusieurs types :

- Associer une catégorie d'exigences à un type ou un ensemble de types de méthodes de validation (ex : toutes les exigences de sécurité doivent être vérifiées par …).
- Associer une source d'exigences à un type ou un ensemble de types de méthodes de validation (ex : toutes les exigences dont la source est FAR25 doivent être vérifiées par …).
- Associer un sous-ensemble d'exigences à une activité commune de validation.

Enfin, la stratégie de validation inclut la stratégie de traçabilité des exigences. L'ensemble des activités de validation sont décrites dans le plan de validation avec leurs attributs associés. L'ensemble des données de validation, y compris les données de traçabilité, sont des données d'ingénierie système.

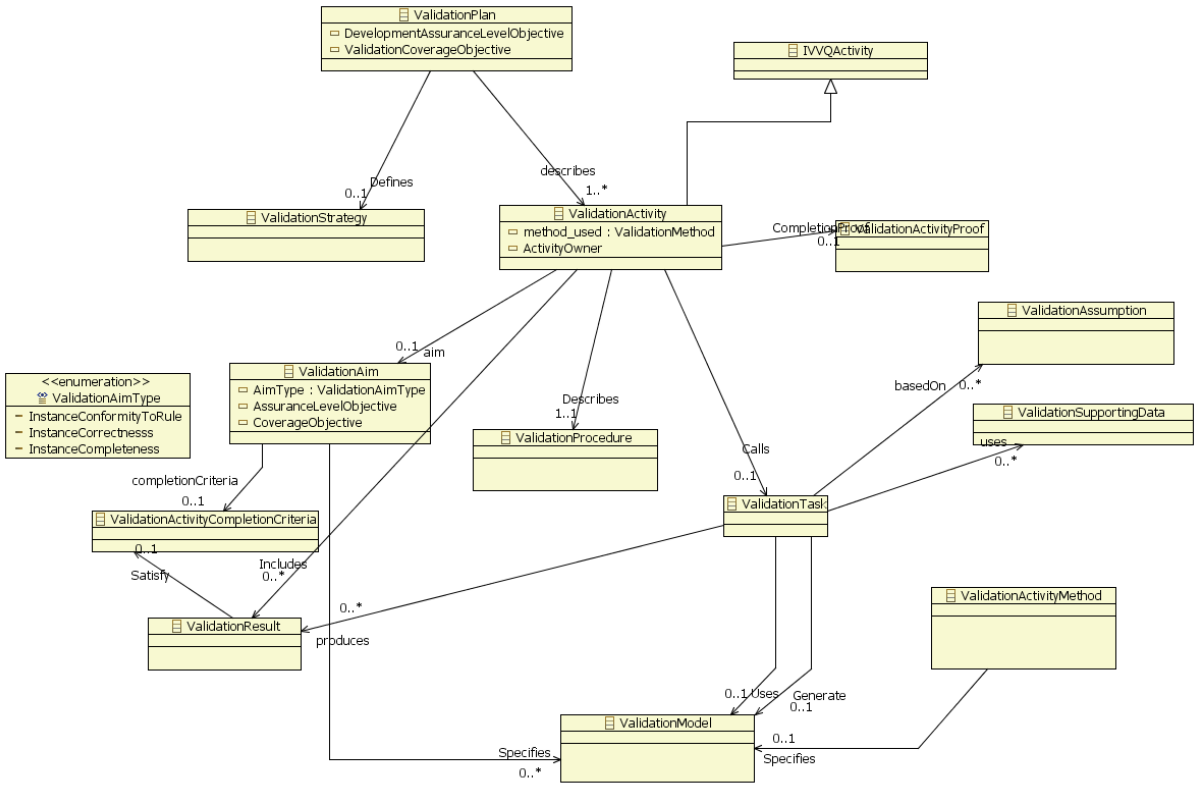


Figure 4.12. : Eléments généraux de validation

L'activité de validation. L'activité de validation correspond à une tâche réalisée dans le but de satisfaire à un objectif de validation spécifié par l'élément *ValidationAim*. L'objectif de validation permet de formaliser un ou plusieurs critères de complétude de l'activité de validation. L'activité de validation va être menée par une procédure de validation utilisant un ou plusieurs modèles de validation. L'ingénierie du modèle de validation est assurée par un ensemble de tâches de validation formalisées à partir de la procédure de validation. Une tâche de validation produit un résultat de validation qui peut être évalué au regard du critère de complétion de l'activité. Par ailleurs, elle utilise éventuellement des hypothèses et des données techniques de support qui doivent être formalisées.

Le modèle de validation. Le modèle de validation est un modèle d'analyse.

4.3. Application des modèles dans la méthodologie

Ce sous chapitre introduit les éléments permettant d'utiliser le modèle d'informations dans une démarche d'ingénierie dirigée par les modèles. Il décrit notamment comment la démarche peut être structurée au moyen de points de vues, de propriétés systèmes et de patrons de modélisation.

4.3.1. Organisation du modèle et points de vues

Le modèle d'ingénierie système développé est un ensemble d'éléments d'ingénierie structurés entre eux, sans présager de la façon dont ses éléments doivent être présentés à l'utilisateur. Les points de vue définissent comment présenter au développeur du modèle les informations pertinentes lors de la réalisation des processus d'I.S. Cette partie montre comment la méthodologie MICCSA permet d'organiser le modèle d'ingénierie système et précise comment la constitution de points de vue sur le modèle permet d'atteindre ces objectifs.

4.3.1.1. Organisation des modèles

La méthodologie s'appuie sur un ensemble cohérent de modèles dont les natures et les finalités sont différentes.

D'une part, elle associe des modèles normatifs aux sous-ensembles identifiés dans le bloc de construction de la norme EIA-632 (cf. EIA, Annexe G). Ces modèles rassemblent les exigences (de l'acquéreur, des autres parties prenantes, techniques, système, exigences spécifiées) et des représentations de solutions développées au cours du projet (solutions logiques et physiques). Ils constituent respectivement les référentiels d'exigences et de définition du système. Organisés selon une hiérarchie de blocs de construction qui couvrent à la fois le système produit final et ses produits contributeurs associés, les modèles normatifs permettent de supporter l'ensemble des activités d'ingénierie système de la méthodologie.

D'autre part, la méthodologie intègre des modèles non normatifs, afin d'assister la définition, l'analyse et la validation des modèles normatifs. Les modèles non normatifs sont ainsi liés aux modèles normatifs précédents.

La méthodologie fait usage de la typologie des modèles de l'AFIS (cf. chapitre II) et affecte à chaque type de modèle un rôle particulier, tel que cela est représenté sur la figure 4.13.

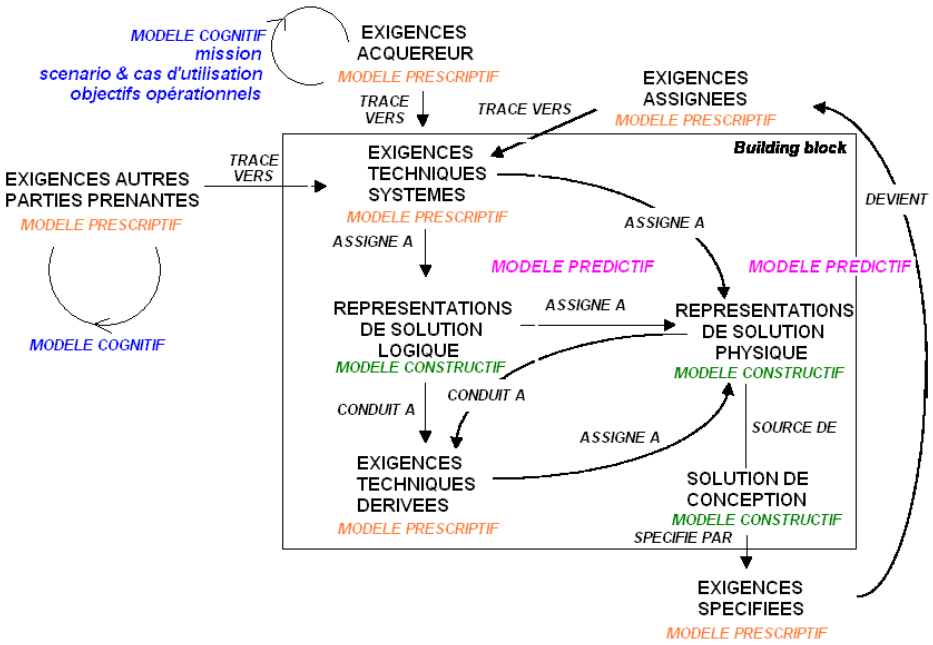


Figure 4.13. : Utilisation des modèles dans MICCSA

Ainsi, nous identifions deux types de besoins d'interopérabilité entre les différents modèles représentés :

- D'une part, l'interopérabilité concerne la relation entre modèles normatifs. Ce besoin est résumé notamment par les flèches liant chaque modèles normatifs : *tracé vers* , *assigné à* , *conduit à* , *source de*,
- D'autre part, l'interopérabilité entre modèles normatifs et non normatifs. Par exemple, le modèle cognitif associé aux exigences acquéreur devra être exploitable par le modèle prescriptif correspondant. De même, un modèle prédictif de solution physique devra être exploitable par le modèle constructif.

4.3.2. Propriétés des modèles

Dans la méthodologie, l'interopérabilité présentée précédemment est assurée notamment par l'usage de propriétés caractéristiques, dont le formalisme est partagé entre la modélisation du domaine du problème et celle du domaine de la solution.

4.3.2.1. Classes de propriétés du modèle

Le modèle d'ingénierie système, au fur et à mesure de son élaboration, regroupe et formalise un ensemble d'informations. Ces informations sont soit directement contenues dans le modèle (ex : les attributs de nom et de type d'un constituant d'un modèles constructifs), soit déduites du modèle par une analyse de celui-ci (par exemple : un ensemble de constituants directement ou indirectement connectés à un autre constituant d'un modèle constructif).

Ces informations, évaluées au regard d'un objectif système spécifique, constituent les propriétés du modèle.

L'approche préconisée par la méthodologie repose sur l'évaluation, a priori, d'un ensemble de propriétés sur le modèle d'ingénierie système, sans connaissance préalable de celui-ci. Le problème peut alors être décomposé en deux sous-problèmes :

- S'assurer qu'un ensemble de propriétés est évaluable sur le modèle. Cela revient à évaluer si le modèle contient les informations pertinentes qui permettront d'évaluer cet ensemble de propriétés.
- S'assurer que toutes les propriétés sont satisfaites par le modèle.

Ces deux sous-problèmes sont complémentaires et couvrent les aspects **d'évaluation de la complétude et de l'exactitude du modèle,** relativement à l'ensemble de propriétés considéré.

Les propriétés possèdent des caractéristiques communes. Ainsi, une propriété peut être inclusive (elle spécifie, dans ce cas, ce qui doit être trouvé dans le modèle), ou exclusive (ce qui ne doit pas être trouvé dans le modèle).

Des propriétés appelées propriétés de connectivité assurent que chaque élément est relié de manière cohérente avec les autres éléments du modèle d'ingénierie système. La vérification de la connectivité des éléments du modèle est déclinée sur différents éléments de ce dernier et constitue un moyen d'évaluer sa complétude et sa cohérence.

Par ailleurs, des propriétés permettent d'énumérer des éléments du modèle, ou de comparer des valeurs d'attributs à un seuil ou un intervalle prédéterminé. Enfin, d'autres propriétés représentent des métriques sur le modèle représentatif de l'état d'avancement d'une phase, d'un taux de couverture ou d'un niveau de complexité. Plusieurs exemples simples de propriétés évaluables sur le modèle sont représentés graphiquement sur la figure 4.14.

La figure 4.14 présente six exemples généraux de propriétés évaluées sur un modèle. Les trois premiers exemples peuvent être évalués par des propriétés de connectivités entre les éléments descriptifs des exigences, des fonctions, et des composants du modèle. En voici une brève description :

- L'exemple A permet de s'assurer de la satisfaction de l'ensemble des exigences par les éléments d'une solution.
- L'exemple B présente un cas dans lequel une fonction du système doit être implémentée sur un composant spécifique. Une propriété de connectivité peut alors évaluer si la condition précédente est satisfaite.
- L'exemple C présente un cas ou deux fonctions distinctes doivent être réalisées par deux sous-ensembles distincts de la conception physique. L'allocation des fonctions sur les composants et l'existence de connexions entre les composants physiques peuvent être évaluées par une propriété de connectivité.
- L'exemple D présente un cas dans lequel un attribut unique du modèle (par exemple, un attribut « dimension » associé à un élément de conception physique) est contraint par deux exigences. Il est alors intéressant de détecter cette situation pour informer sur la dépendance existante entre les deux exigences et leur conflit potentiel.
- L'exemple E décrit une caractéristique d'incohérence car une exigence fonctionnelle ne doit être allouée directement qu'à un élément descriptif d'une fonction sur le modèle.

- Enfin, l'exemple F décrit simplement le fait qu'une exigence prescrit une valeur ou un intervalle autorisé sur un élément de solution (par exemple, le poids associé à un sous-ensemble). Ce cas correspond aux propriétés de type évaluation de valeurs dans le modèle.

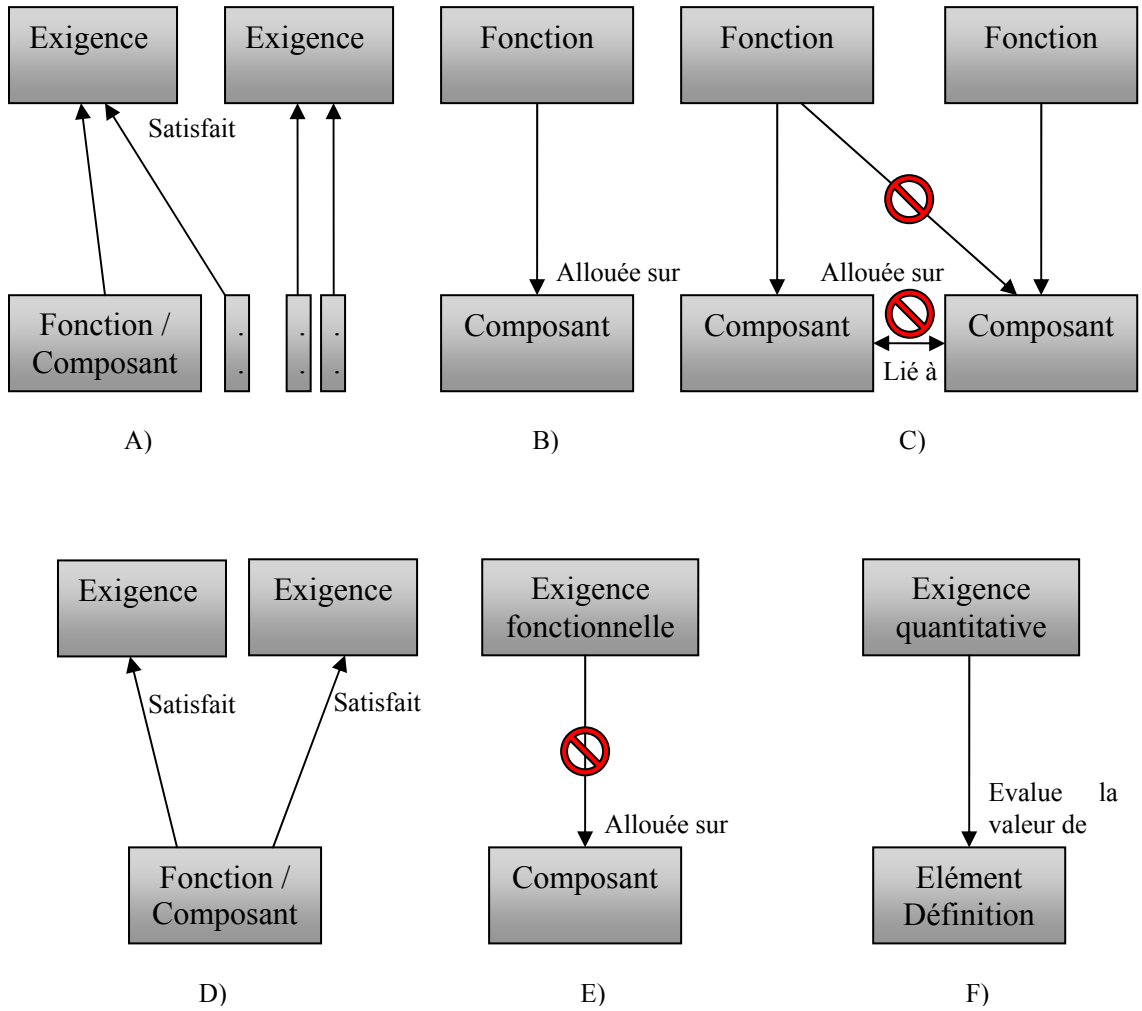


Figure 4.14. : Exemples de propriétés

Dans la méthodologie MICCSA, les caractéristiques de complétude et d'exactitude du modèle évoquées précédemment sont déclinées selon trois classes de propriétés représentées sur la figure 4.15. Les propriétés méthodologiques permettent de formaliser la complétude et l'exactitude du modèle par rapport aux règles de modélisation définies dans la méthodologie. Elles permettent de formaliser, à un instant particulier du développement, l'exactitude et la complétude du modèle, relativement à ce qui est attendu de lui à cet instant particulier.

Les propriétés systèmes permettent de formaliser l'exactitude et la complétude du modèle par rapport aux propriétés intrinsèques au système à développer. Enfin, une dernière classe de propriétés concerne la fidélité de la modélisation par rapport au problème réel (besoin et attentes) et à la solution réelle (le système).

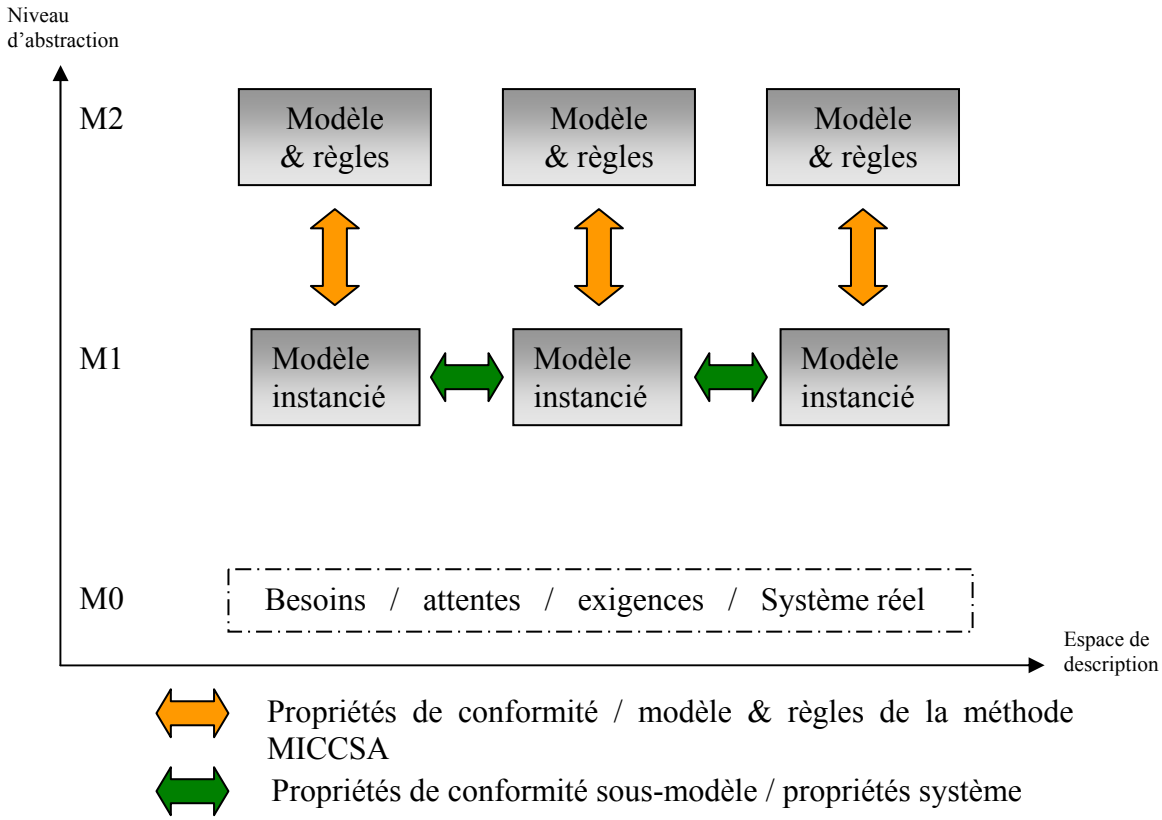


Figure 4.15. : Classes de propriétés à vérifier sur le modèle

4.3.2.2. Classe des propriétés système

La classe des propriétés système est liée aux exigences spécifiées sur le système. Rappelons que l'AFIS [AFI] décrit une exigence comme un élément qui *prescrit une propriété dont l'obtention est jugée nécessaire*. Son énoncé peut être une fonction, une aptitude, une caractéristique ou une limitation à laquelle doit satisfaire un système, un produit ou un processus.

Partant de cette définition, la satisfaction d'un ensemble d'exigences système peut alors être évaluée sur le modèle par la vérification d'un ensemble de propriétés. Ces propriétés doivent être à la fois :

- formalisées dans la partie prescriptive du modèle,
- évaluables par analyse de la partie constructive du modèle,
- significatives en regard du système modélisé (i.e. représentatives des propriétés que le système devra posséder).

La communauté scientifique a effectué des travaux sur l'établissement de typologies exhaustives de propriétés attribuables à un système technique et à son développement. Une étude synthétique de ces travaux est présentée dans la littérature de l'ingénierie système.

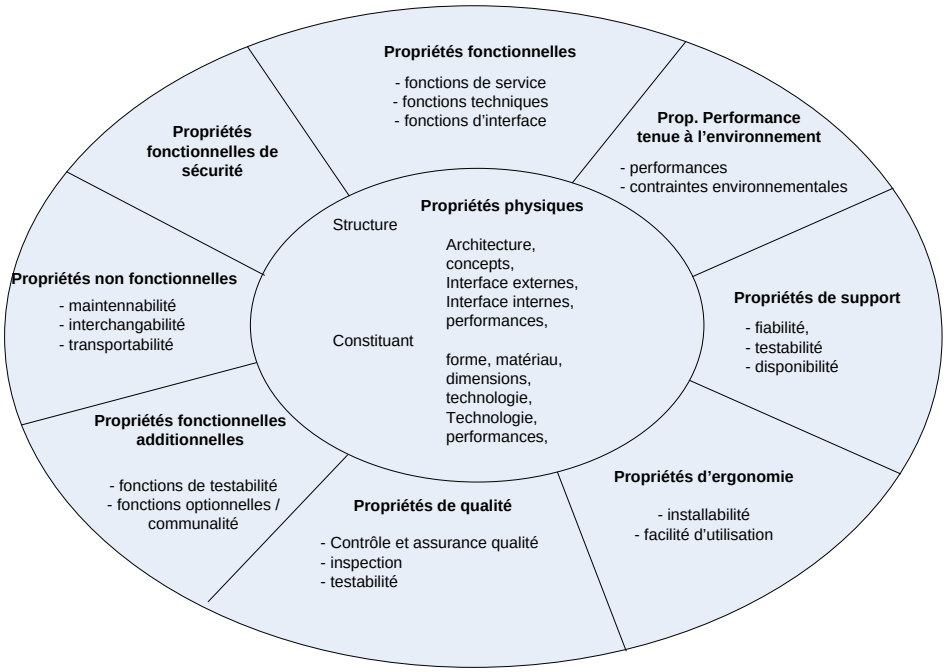


Figure 4.16. : Typologie des propriétés générales des systèmes techniques

La figure 4.16 représente une partie de ces propriétés. Les propriétés situées en périphérie sont dites propriétés externes et peuvent être évaluées indépendamment de la structure interne du système. Les propriétés situées au centre de la figure dépendent directement de la structure, de la géométrie et des propriétés physiques internes du système.

Les propriétés systèmes retenues dans la méthodologie sont les propriétés évaluables sur les architectures opérationnelles, logiques et physiques dont l'évaluation est nécessaire pour la spécification complète du système.

Le tableau présenté en annexe C représente les catégories de propriétés systèmes qui peuvent être prescrites dans le modèle. Celles-ci sont définies à partir des catégories d'exigences standards du domaine aéronautique.

4.3.2.3.Classe des propriétés méthodologiques

Les propriétés méthodologiques du modèle sont les caractéristiques du modèle souhaité afin que celui-ci constitue une aide utile à la conduite des processus d'ingénierie système. La formalisation de ces propriétés est utile afin d'assister le développeur du modèle dans les activités d'ingénierie système.

Les règles de syntaxe de certains langages de modélisation système (ex : SysML) forcent déjà le développeur du modèle à respecter certaines règles de modélisation. Elles le guide ainsi vers un modèle conforme à des bonnes propriétés d'IS.

Cependant, le travail réalisé dans cette thèse permet d'étendre le champ d'application de ces règles à un ensemble de propriétés méthodologiques, permettant d'assister l'ingénieur dans le développement du modèle, et donc, dans la conduite des processus d'ingénierie système. A l'inverse des règles de syntaxe existant au niveau des langages et outils, la

vérification des règles méthodologiques tient compte de la sémantique de modélisation, du contexte constitué des éléments du modèle déjà existant et de la phase de développement du modèle.

Les principaux résultats attendus par la formalisation de propriétés méthodologiques sont les suivants :

- identification et indication de manques (objets, attributs, liens, valeurs) dans le modèle, associées éventuellement à une indication d'action corrective,
- identification et indication d'erreur ponctuelles ou de configurations erronées ou indésirables dans le modèle, associées éventuellement à une indication d'action corrective,
- indication sur la conduite des processus et des activités,
- automatisation de l'élaboration de certaines parties du modèle,
- amélioration de la représentation des éléments pour la maîtrise des risques, des incertitudes, etc.

Ces règles sont déclinées à partir des exigences sur les processus définies dans l'EIA 632. Elles permettent d'assurer le développement d'un modèle conforme aux bonnes pratiques de l'ingénierie système et d'assister les activités de spécification, de conception, de validation et d'analyse du modèle.

4.3.2.4. Utilisation des propriétés dans la méthodologie

Les propriétés systèmes sont formalisées dans la partie prescriptive du modèle d'ingénierie système. Leur formalisation permet d'évaluer les éléments constructifs du modèle pendant les phases préliminaire d'élaboration et d'analyse d'architecture, puis de leur validation.

Les propriétés méthodologiques sont formalisées sous la forme de règles statiques d'une part et de patrons de modélisation d'autre part. Les règles statiques peuvent être vues comme des règles qui doivent être vérifiées à tout moment (invariants) ou dans un contexte particulier du développement du modèle sous la forme de pré-conditions ou de post-conditions. Les patrons de modélisation appliqués sur les modèles prescriptifs ou constructifs permettent d'utiliser la formalisation des propriétés de la méthodologie de manière dynamique pour engendrer de nouveaux éléments dans le modèle et ainsi assurer la conformité aux règles précitées.

4.3.3. Ingénierie des modèles et vérification des propriétés

Les propriétés système précédemment décrites doivent être vérifiées tout au long de l'élaboration des modèles de solutions logiques et physiques. Une phase d'élaboration d'un modèle constituant est ainsi suivie d'une phase de vérification d'un ensemble de propriétés systèmes afin de valider celui-ci.

La démarche constituée par ces deux phases est itérative, et la démarche globale conduit à suivre un enchaînement des activités conforme au *workflow* MDA. Les résultats de

vérifications sont présentés au développeur et la démarche de vérification est généralement suivie d'une phase de travail sur le modèle afin de corriger ou compléter celui-ci.

Plus précisément, les problèmes de validation rencontrés lors de la vérification de propriétés système sur le modèle peuvent être de deux natures : incomplétude des éléments d'entrée pour évaluer une propriété ou inexactitude des éléments d'entrées pour la satisfaction d'une propriété.

Dans le cas d'une incomplétude du modèle, une activité supplémentaire d'ingénierie système doit être conduite afin de compléter le modèle. Dans le cas d'une inexactitude du modèle, des activités correctives d'ingénierie système doivent être menées.

4.3.4. Ingénierie des modèles et application de patrons de modélisation.

La méthodologie s'appuie également sur l'application de patrons sur le modèle, dans chaque phase de développement. Dans ce contexte, un patron est un motif générique appliqué au modèle lorsque celui-ci se trouve dans un contexte particulier identifiable par un ensemble de pré-conditions.

Les caractéristiques apportées par l'application de patrons sont souhaitables au regard de la qualité du modèle, de sa complétude ou de l'effort nécessaire au développement de celui-ci. Dans tous ces cas, les patrons appliqués sont un moyen de réalisation des processus d'ingénierie système associé.

L'utilisation de patron en conception système a également été explorée dans l'objectif de l'optimisation de caractéristiques spécifiques des systèmes et/ou dans le cadre de domaine technique spécifique sureté de fonctionnement [Keh_05] ou l'amélioration de la réutilisation de modèles [Rin_05].

Dans notre travail, les bénéfices attendus de l'application de patrons de modélisation sont les suivants :

- Guider l'utilisateur dans la réalisation des processus techniques d'ingénierie système de la méthodologie.
- Réduire le nombre d'itérations nécessaires pour obtenir un modèle normatif valide.
- Améliorer l'usage des vues sur le modèle pour l'analyse de celui dans le cadre d'une préoccupation spécifique d'utilisation du modèle.

Pour cela, les propriétés méthodologiques évaluées sur le modèle ont deux rôles. D'une part, elles sont les pré-conditions de l'application des patrons. D'autre part, Les patrons de générations de modèles permettent d'améliorer la conformité du modèle vis-à-vis des propriétés méthodologiques.

Pour cela, l'approche utilisée a été de définir et de regrouper les patrons selon les exigences de processus de conception et d'évaluation système définis dans l'EIA-632. Ainsi, des patrons sont associés à la définition des exigences. Ceux-ci permettent d'accélérer le

processus de définition d'exigences pour lequel la validité sera ensuite évaluée par des patrons de validation aux niveaux cohérence et complétude. La définition des solutions logiques et physiques s'appuie également sur des patrons qui permettent la sélection automatique et l'intégration de composants dans les architectures. Ils permettent, en particulier, de faciliter la composition des concepts de solution (ou principes constructifs) constitutifs des solutions logiques et physiques. Enfin, des patrons sont associés à l'analyse système. Ils permettent d'évaluer des architectures validées ou des sous-ensembles d'architectures validés afin de guider la décision de l'utilisateur dans la conception architecturale.

La méthodologie distingue trois types de patrons applicables sur les modèles décrits ci-dessous. Ces types de patrons se distinguent par leurs conditions d'application sur le modèle et par le type d'actions réalisées.

4.3.4.1. Patrons de développement du modèle

Ces patrons permettent de créer de nouveaux éléments dans le modèle et de connecter ceux-ci au reste de la modélisation. L'objectif est de faciliter et d'accélérer le développement d'un modèle complet et conforme aux besoins des processus d'ingénierie système. Sur la partie normative du modèle, l'objectif attendu est de favoriser sa complétude et la réduction des itérations nécessaires à son élaboration.

4.3.4.2.Patron pour l'applicabilité des exigences

Au fur et à mesure du développement des modèles constructifs, de nouvelles exigences applicables sont introduites automatiquement par le patron d'applicabilité. L'applicabilité d'une exigence au produit en cours de développement peut être considérée comme l'évaluation d'un certain nombre de conditions à la fois sur l'exigence elle-même et sur le produit.

Le principe d'évaluation de l'applicabilité des exigences à partir des modèles constructifs est schématisé par la figure 4.17. Dans l'exemple A, une exigence est allouée à un type de constituant. Dans ce cas, l'exigence est affectée aux constituants instanciés directement ou indirectement à partir de ce type. Par exemple, un constituant est instancié à partir du type *bielle à ressort*, lui-même héritant du type *bielle*. Dans ce cas, les exigences applicables à ces types sont allouées sur le constituant.

Dans l'exemple B, une exigence est affectée à un sous-ensemble de constituants. Cette exigence peut être, sous certaines conditions, appliquées au constituant. Ainsi, une exigence sur le traitement de surface appliqué à un ensemble de constituants est allouée sur le constituant lui-même.

Dans l'exemple C, un constituant appartient à une catégorie de constituants. Une catégorie peut désigner l'acquéreur du système, le niveau de criticité du constituant, l'autorité de certification concernée (FAA / EASA), etc.

Les conditions d'applicabilités précitées peuvent être composées entres elles. Ainsi, un client peut formuler une exigence sur un type de constituant particulier en restreignant son applicabilité à un sous-ensemble spécifié.

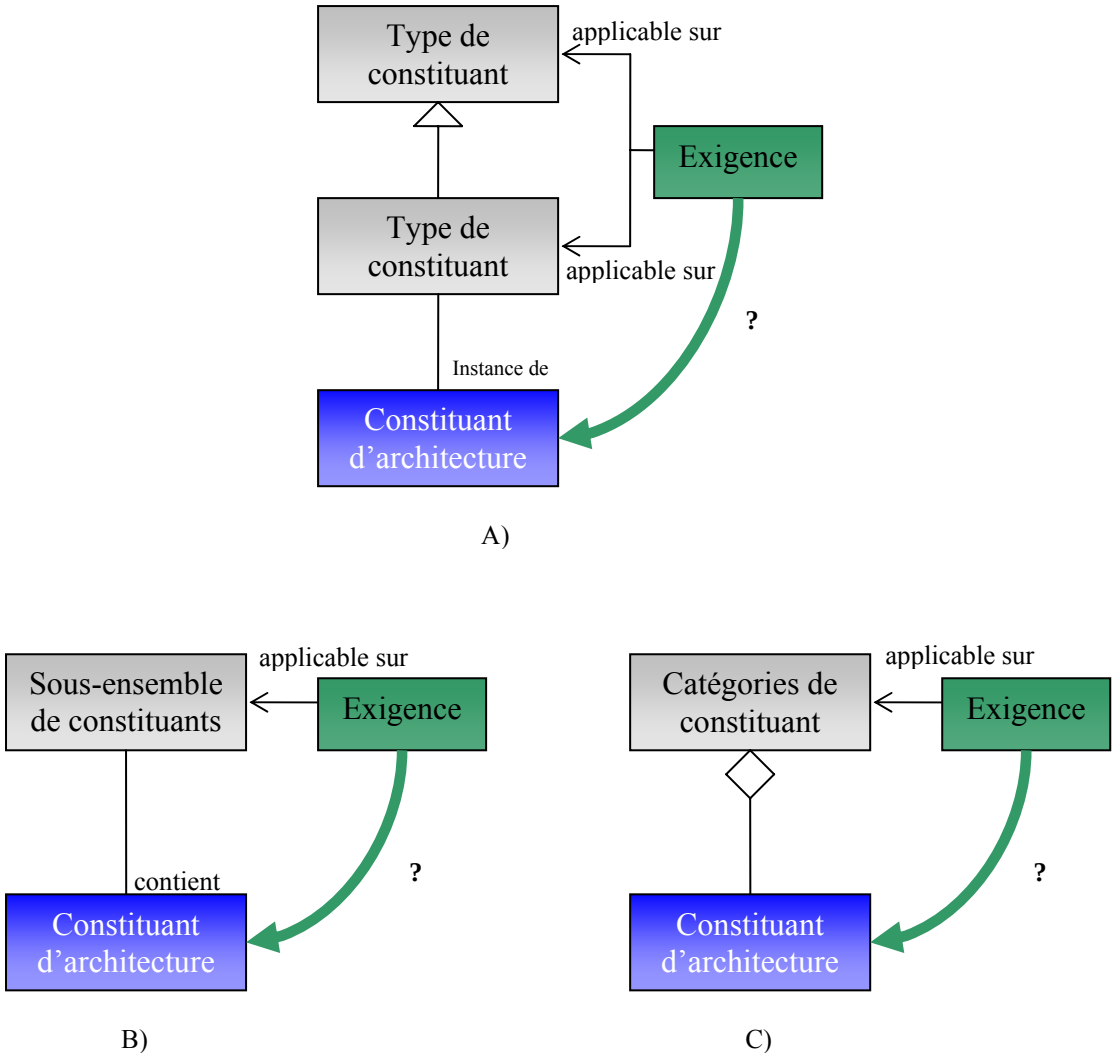


Figure 4.17. : Conditions d'application des exigences sur les constituants des modèles de solution

4.3.4.3. Patrons d'extension de modèle

Des patrons permettent d'accélérer et d'automatiser la production de certaines parties répétitives du modèle. Ainsi, un patron permettra par exemple de créer des structures d'exigences conformes à ce que prescrivent les règles méthodologiques et adaptées aux structures de spécification standard de LATECOERE. (Ces spécifications couvrent les exigences systèmes, spécification des exigences dérivées techniques, allocation des exigences systèmes sur les constituants d'architecture logiques et physiques.) D'autres patrons permettront de générer automatiquement des éléments justificatifs d'exigences ou de constituants d'architecture.

4.3.4.4. Patrons de transformation entre types de modèles

Des patrons permettent d'assurer l'interopérabilité entre les modèles exprimés dans différents langages. Cette interopérabilité entre les modèles a été abordée au chapitre II. Sur la partie non normative du modèle, l'objectif principal consiste en une meilleure formalisation

de l'utilisation des informations contenues dans les modèles cognitifs et en l'amélioration de l'utilisation des modèles prédictifs (notamment des modèles de simulation système) dans les processus d'ingénierie système.

De manière générale, les patrons de modélisation qui créent de nouveaux éléments dans le modèle sont appliqués à un instant particulier du développement lié à la réalisation du processus d'ingénierie système associé.

4.3.4.5. Patrons de génération des vues du modèle

La complexité du modèle d'ingénierie système rend la définition de vues sur celui-ci laborieuse et sujette à erreurs. Des patrons peuvent être définis afin d'importer dans une vue l'ensemble des éléments du modèle pertinents pour celle-ci. Ils n'ajoutent rien au modèle, si ce n'est des éléments de connections permettant la construction de la vue.

Comme détaillé à la section 3.5, les vues permettent de réaliser les activités de validation et d'analyses des solutions logiques et physiques, notamment la conduite des compromis nécessaires à la définition de la solution physique.

Les patrons de génération de vues sont donc destinés à maintenir des vues cohérentes sur le modèle tout au long du développement de celui-ci. Ces patrons doivent donc être appliqués à chaque évolution du modèle afin que les vues soient redéfinies.

4.3.4.6. Patrons d'analyse et de correction du modèle

Des patrons de correction d'incomplétude et d'inexactitude permettent de détecter les manques ou erreurs dans le modèle. Le principe d'identifier des patrons caractéristiques de configurations indésirables d'un modèle (anti-pattern) a déjà été exploré dans les travaux de Brown dans le domaine du développement logiciel [Bro_99]. Ce principe a été étendu et décliné en patron d'incomplétude et d'inexactitude du modèle.

Lors de la détection d'une configuration d'incomplétude ou d'inexactitude du modèle, deux stratégies peuvent être employées :

- le patron peut créer les éléments correctifs du modèle accompagnés d'un élément explicatif de l'action réalisée sur le modèle.
- le patron peut créer un élément descriptif du défaut identifié.

Ces deux stratégies sont complémentaires et des éléments explicatifs et/ou descriptifs du problème sont donc créés à l'intention du développeur. Ces éléments doivent être inclus dans une vue du modèle afin d'être rendus visible à l'utilisateur.

Comme les patrons de développement du modèle, les patrons de correction sont destinés à être appliqués à un instant particulier du développement lié à la réalisation du processus d'ingénierie système associé.

Des exemples de mises en œuvre de tels patrons sont présentés au chapitre IV sur la plate forme Eclipse / Open Architecture Ware.

4.3.5. Constitution des vues d'ingénierie système

L'évaluation des propriétés et l'application des patrons du *workflow* MDA sont rassemblés dans des vues du modèle. Ces vues se conforment au point de vue présentés ci-dessous.

4.3.5.1. Point de vue validation d'architecture

La vue validation d'architecture permet de représenter graphiquement les éléments de validation d'un ensemble d'exigences allouées à une architecture logique ou physique. Cela inclut les éléments de traçabilité entre exigences système et architectures logiques et physiques, entre exigences dérivées et architecture, les objectifs, les moyens mis en œuvre et les résultats de validation obtenus. La vue validation permet donc d'améliorer la confiance qu'aura le développeur sur la conduite de l'activité de validation tout au long de la spécification et du développement système. Ainsi, elle contribuera à améliorer implicitement la conception du système.

4.3.5.2. Point de vue sélection de solution

Ce point de vue permet l'analyse comparative de solutions d'architecture validées. Cette analyse permet de construire un référentiel de comparaison cohérent afin de comparer des concepts de solutions physiques ou les alternatives de solutions. Ces derniers permettent de représenter graphiquement les éléments communs à l'analyse comparative de solutions, et les éléments propres à chaque alternative afin de faciliter l'analyse de compromis et la sélection d'une solution.

4.3.5.3. Point de vue performances

La vue performances permet de montrer quels moyens sont mis en œuvre et quels sont les éléments utilisés pour définir les exigences de performances, affecter les exigences de performances aux architectures logiques et physiques, et les moyens mis en œuvre pour vérifier que les performances obtenues lors des activités de vérification système sont conformes aux exigences spécifiées.

4.3.5.4. Point de vue analyse d'impact et analyse de risque

La vue analyse d'impact montre les éléments affectés ou potentiellement affectés lors d'une évolution de la partie normative du modèle (référentiel d'exigences ou de définition). Elle est propagée entre les différents modèles constituants le système. Une variation de la partie prescriptive du modèle correspond à une évolution des exigences client et des autres parties prenantes ou des exigences systèmes. Une évolution de la partie normative correspond à une modification de la définition du système (par exemple liée à l'obsolescence d'un composant de l'architecture). La vue analyse d'impact s'attache donc à décrire la variabilité des éléments du modèle, notamment des architectures logiques et physiques. Ces variabilités sont ensuite traduites en risques coûts, délais, et performances produit.

4.3.5.5. Point de vue partie prenante

Le point de vue partie prenante permet de rassembler les éléments du modèle pertinents au regard d'une partie prenante identifiée. Elle rassemble les éléments prescrits (exigés) par la partie prenante, les exigences systèmes possédant un lien de traçabilité vers les exigences de

la partie prenante ainsi que les éléments qui permettent de répondre à ces exigences, les moyens mis en œuvre et les résultats destinés à prouver cette conformité (moyens de validation et de vérification).

4.3.5.6. Point de vue métiers

Les points de vue métiers permettent de garder dans le modèle les éléments pertinents relativement aux préoccupations et aux compétences spécifiques d'un métier. Les métiers identifiés sont : Bureau d'étude - essais qualification / certification, BE - Calcul, Achat, Méthodes et outils, chiffrage, préparation technique, outillages, support client.

L'ensemble des éléments présentés dans ce sous-chapitre permettent de préciser la manière dont les processus d'ingénierie système doivent être réalisés. Le sous-chapitre suivant décrit, au niveau des processus puis des méthodes à appliquer, comment la méthodologie doit être employée. La mise en œuvre au moyen des langages et outils sera présentée au chapitre IV.

4.4. Instanciation du modèle d'ingénierie système

Cette partie concerne la description des processus, et des méthodes de la méthodologie. Elle utilise les principes présentés dans les parties précédentes et met en œuvre les éléments génériques présentés dans le modèle d'informations. La réalisation des activités de la méthodologie consiste à élaborer les modèles décrits précédemment et à valider ces modèles selon les processus de la méthodologie. Ces deux activités définissent un enchaînement dans la modélisation conforme à la prescription de l'approche MDA.

4.4.1. Processus d'ingénierie système

Cette section présente les processus techniques définis dans la méthodologie, décrits de manière générale dans la section 4.2.2. Ces processus techniques :

- satisfont aux exigences des processus techniques de l'EIA 632,
- correspondent, quant au contenu et au niveau de détail, aux produits ciblés dans la méthodologie.

L'enchainement des processus d'ingénierie système à un niveau de développement donnée est présentée figure 4.18.

4.4.1.1.Processus de définition des exigences

Les activités ci-dessous font partie du processus de définition des exigences :

- identifier les sources d'exigences,
- collecter les exigences d'une source,
- éliciter les exigences d'une source,
- définir les exigences systèmes,
- définir les exigences dérivées,

- spécifier une solution système.

Ces activités sont détaillées en annexe D.

4.4.1.2. Processus de définition des solutions

Les activités ci-dessous, déclinées sur les architectures opérationnelles, logiques, et physiques, font partie du processus de définition des solutions :

- Concevoir des alternatives innovantes de solution (opérationnelle, logique, physique),
- intégrer des alternatives de solution (opérationnelle, logique, physique),
- sélectionner une alternative de solution (opérationnelle, logique, physique).

Ces activités sont détaillées en annexe D.

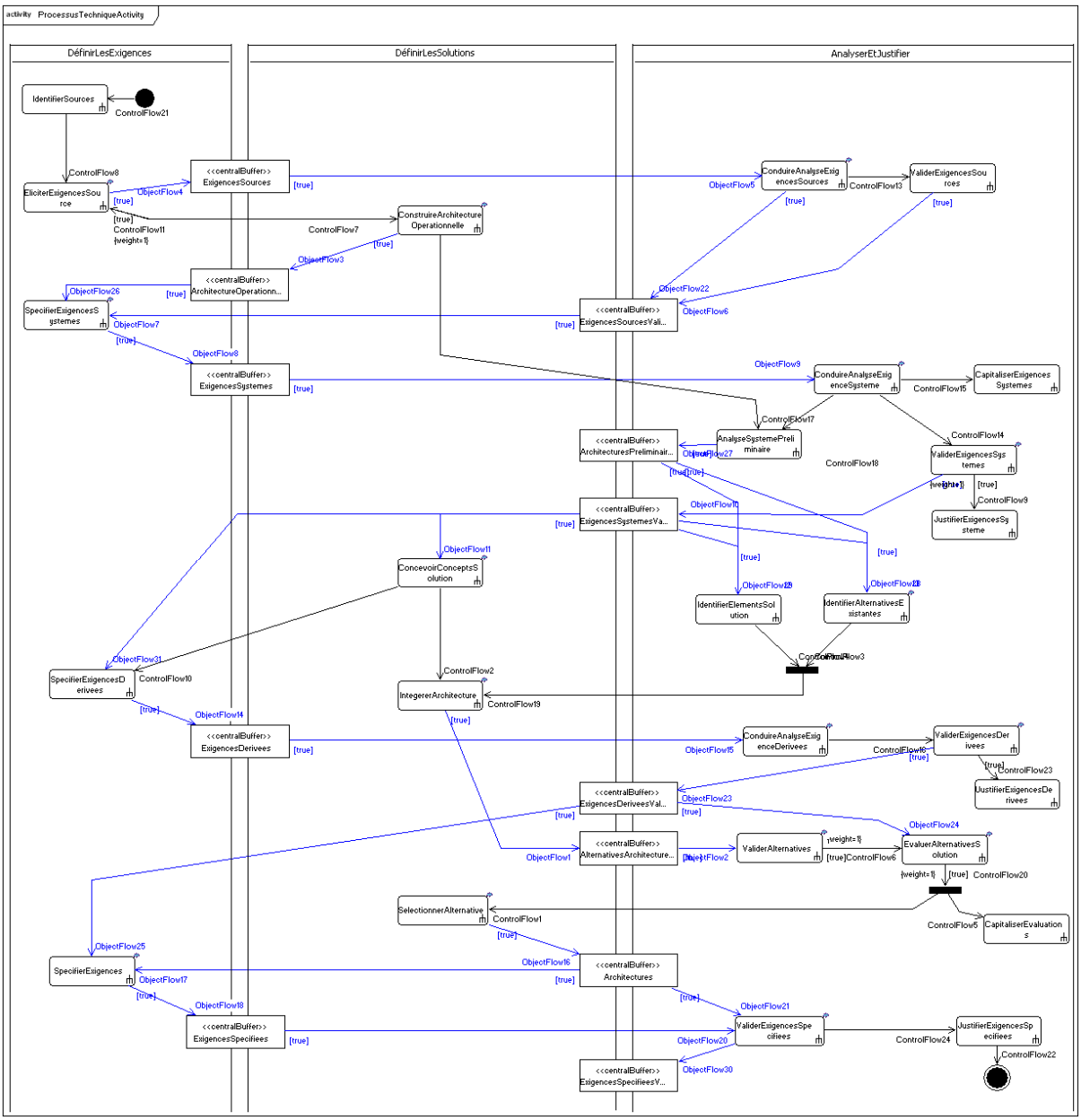


Figure 4.18. : Processus de conception système MICCSA

4.4.1.3.Processus d'analyse et de justification

Le processus d'analyse et de justification est réalisé au moyen des activités suivantes :

- Identifier des alternatives de solution existantes (opérationnelle, logique, physique),
- analyser les solutions système existantes (opérationnelle, logique, physique),
- valider les alternatives de solution (opérationnelle, logique, physique),
- évaluer les alternatives de solution (opérationnelle, logique, physique),
- justifier la sélection d'une alternative de solution (opérationnelle, logique, physique),
- valider l'expression des exigences,
- valider les exigences systèmes,
    - o résoudre les conflits,
    - o établir un compromis,
    - o valider la complétude,
    - o valider l'exactitude,

- valider les exigences dérivées,
- valider les exigences spécifiées,
- valider les solutions système (opérationnelle, logique, physique),
- justifier les exigences (système, dérivées, et spécifiées),
- adapter une solution à un contexte,
- capitaliser une solution (opérationnelle, logique, physique),
- capitaliser les décisions techniques hors sélection de solution (opérationnelle, logique, physique).

Ces activités sont détaillées en annexe D.

4.4.1.4. Processus et méthodes

Les processus présentés ci-dessus, ainsi que la figure 4.18, spécifient les activités à conduire ainsi que les éléments qui doivent être produits et consommés par ces activités. Ces activités sont réalisées à l'aide de méthodes basées sur l'ingénierie des modèles (IDM). Les caractéristiques de ces méthodes sont présentées dans les paragraphes suivants.

4.4.1.5.Formalisation des jalons dans le développement basée sur les modèles

Afin d'organiser les opérations d'ingénierie du modèle entre elles (vérifications de propriétés, application de patrons de modélisation), celles-ci ont été organisées en un nombre limité de jalons de développement. L'ensemble des transformations sont réalisées entre les jalons ainsi définis.

Chaque jalon est associé à un ensemble :
- de pré-conditions nécessaires à la réalisation des activités d'ingénierie de modèle,
- de post-conditions permettant de valider le passage du jalon.

Une fois un jalon atteint dans le développement du modèle, une évolution des pré-conditions associées à ce jalon implique de réaliser à nouveau les étapes d'ingénierie des modèles associées. Cette organisation permet de mettre en place la gestion d'impact et la maîtrise les évolutions du modèle.

Les paragraphes suivants décrivent les jalons définis pour le développement du modèle d'ingénierie système conformément aux activités présentées sur la figure 3-18. Ces jalons se conforment aux processus et permettent la réalisation de ceux-ci par des activités dirigées par les modèles.

4.4.2. Evolution du modèle, référentiels et jalons de projets

L'enchaînement des modèles s'appuie sur le principe de transformations successives du modèle dont la forme générique est présentée figure 4.19. Les transformations sont réalisées par l'application des patrons de modélisation, ou réalisées manuellement par le développeur. La validité des modèles avant et après transformation est assurée par la vérification de propriétés sous la forme d'ensemble de pré-conditions, de post-conditions.

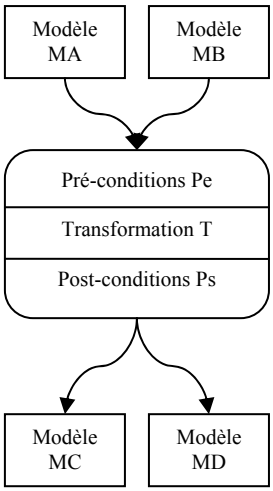


Figure 4.19. : Transformation de modèles génériques dans la méthodologie MICCSA

4.4.2.1. Jalon des référentiels d'exigences parties prenantes et d'architecture opérationnelle

Ce jalon permet de débuter la définition d'un ensemble d'exigences système sur la base d'un ensemble validé d'exigences sources et d'un ensemble de concepts d'architecture opérationnelle. Les conditions ci-dessous doivent être réunies pour l'établissement de ce jalon :

- L'ensemble des parties prenantes et des sources d'exigences ont été identifiées dans le modèle.
- Le référentiel d'exigences acquéreurs et affectées a été validé.
- Les référentiels d'exigences des autres parties prenantes ont été validés. Cela inclut, en particulier ,les exigences des autorités de certification.
- Les concepts d'architectures opérationnelles ont été identifiés dans le modèle.

4.4.2.2. Jalon du référentiel d'exigences systèmes :

Ce jalon marque la disponibilité d'un premier référentiel validé d'exigences système obtenu à partir des exigences sources et des premières définitions d'architectures logiques et physique. Il marque aussi l'établissement d'une architecture opérationnelle solution. Les conditions suivantes doivent donc être réunies :

- les exigences système ont été validées.
- les fonctions de service systèmes et les concepts préliminaires d'architecture physiques ont été identifiés.
- l'architecture opérationnelle du système est établie en cohérence avec les exigences système,
- l'analyse système préliminaire est conduite et produit plusieurs vues des modèles qui seront exploitées dans la phase de sélection de l'architecture.

4.4.2.3. Jalon de conception logique :

Ce jalon marque la disponibilité dans le modèle de plusieurs solutions logiques et de leurs exigences dérivées correctement liées aux exigences systèmes :

- plusieurs alternatives d'architectures logiques ont été définies,

- les objets des prescriptions des exigences systèmes ont été affectés à ces solutions logiques,
- les exigences dérivées des alternatives d'architectures logiques ont été définies.

La réalisation de ce jalon donne lieu aux étapes d'ingénierie de :

- Validation des solutions logiques.
- Validation des exigences dérivées des alternatives logiques.
- Evaluation des propriétés systèmes prescrites affectées à la solution logique,
- Analyse système préliminaire (niveau logique) est conduite et produit plusieurs vues du modèle qui seront exploitées dans les phases de sélection de l'architecture.
- Définition de plusieurs alternatives de solutions physiques.

4.4.2.4. Jalon de conception physique

Ce jalon marque la disponibilité dans le modèle de plusieurs solutions physiques correctement liées aux exigences systèmes :

- plusieurs alternatives d'architectures physiques ont été définies,
- les objets des prescriptions des exigences systèmes et des exigences dérivées logiques ont été affectés à ces solutions physiques,

La réalisation de ce jalon donne lieu aux étapes d'ingénierie de :

- Evaluation des propriétés systèmes prescrites affectées aux solutions physiques, et définition des tâches de validation et de vérification à conduire pour chaque architecture physique,
- Analyse système, analyse d'efficacité et d'impact pour chaque alternative de solution.

4.4.2.5. Jalon de sélection d'architecture (préconception)

Ce jalon marque la disponibilité dans le modèle :

- d'un ensemble validé d'exigences systèmes,
- d'alternatives de solutions logiques validées et de solutions physiques,
- d'un ensemble validé d'exigences dérivées et affectées aux solutions physiques,
- d'un ensemble de vues liées à l'analyse des exigences systèmes, des solutions logiques et physiques et des tâches de validation et de vérification à réaliser sur chaque solution.

Lorsque ce jalon est atteint, la conception architecturale peut être terminée :

- par la sélection à l'aide de l'analyse de compromis de l'architecture physique (et son architecture logique associée) retenue comme référentiel de solutions pour les phases suivantes du projet,
- par l'analyse d'impact des exigences systèmes sur l'architecture physique et ses constituants (composants et interfaces).

4.4.2.6. Jalon de définition des exigences spécifiées (conception préliminaire)

Ce jalon marque la disponibilité dans le modèle :

- d'un ensemble validé d'exigences systèmes et d'exigences dérivées et les résultats des tâches de validation associées,

- une solution (logique et physique) retenue,
- un ensemble de vues d'analyse système justifiant la sélection de l'architecture,
- un ensemble de vues réalisées sur les analyses d'impact du référentiel d'exigence sur l'architecture physique.

Lorsque ce jalon est atteint, l'architecture physique retenue peut être spécifiée sous la forme d'exigences systèmes et d'exigences affectées au niveau suivant de conception.

4.4.2.7. Jalon de validation des exigences spécifiées

Ce jalon marque la disponibilité dans le modèle :

- d'un ensemble validé d'exigences spécifiées à partir de l'architecture physique.
- d'ensembles validés de spécifications pour les sous-ensembles constituants de l'architecture physique.

Lorsque ce jalon est atteint, l'architecture physique retenue est validée et les composants constitutifs de l'architecture physique peuvent être à leur tous considérés comme des blocs de constructions. Les blocs correspondants aux produits finaux peuvent être fabriqués, achetés ou sous-traités.

L'ensemble de ces jalons peuvent être associés aux phases de spécifications et de définitions systèmes produisant respectivement les documents d'exigences systèmes ainsi que les documents de description système et de spécification d'équipements.

Phase de dévelop. du modèle IS / modèle constituants	Pré conception faisabilité	Conception préliminaire	Conception détaillée	Acceptation de la conception
Exigences acquéreurs et parties prenantes	**Validés**	-	-	-
Exigences Systèmes	**Spécification Préliminaire**	**Premier référentiel validé**	-	-
Définition Logique	-	**Concepts identifiés**	**Architecture sélectionnée et validées**	-
Définition Physique	-	**Concepts identifiés**	**Architecture Sélectionnée**	-
Exigences Dérivées	-	**Définies**	**Validées**	-
Exigences spécifiées	-	-	**Définies**	**Validées**

Tableau 4.1 : Synthèse phases de développement du modèle

4.4.3. Opérations d'ingénierie des modèles de la méthodologie

Les opérations d'ingénierie des modèles décrites aux paragraphes 4.3.4 permettent de réduire le temps nécessaire au développement du modèle. Elles permettent également d'améliorer la qualité du modèle et de le rendre utilisable pour la réalisation de chaque processus d'I.S. mis en place dans la méthodologie.

Les opérations d'ingénierie des modèles sont décomposées en éléments plus simples, nommés règles et patrons de modélisation. Les règles permettent l'évaluation de pré- et post-conditions et des invariants. Les patrons de modélisation permettent de réaliser les transformations sur le modèle et ses différentes vues. Plus précisément, les patrons de modélisation permettent :

- de créer de nouveaux éléments du modèle (patrons de génération / extension / transformation du modèle décrits au paragraphe 4.3.4),
- de construire les vues présentées au paragraphe 4.3.5.

Les deux sections suivantes présentes comment l'approche MICCSA, en s'appuyant sur certains principes de MDA, définit les exigences et les solutions. Les modèles caractéristiques de MDA sont ainsi projetés sur les modèles prescriptifs et les modèles d'architectures, sur trois niveaux de descriptions distincts : les architectures opérationnelles, logiques et physiques. Celles-ci correspondent, respectivement, aux modèles spécifiques de plate-forme opérationnelles, logiques et physiques (O-PSM, L-PSM, P-PSM).

En amont de ces architectures-solutions, chaque niveau est défini à partir :

- d'un modèle indépendant des plateformes (O/L/P-PIM) représenté par un modèle prescriptif.
- d'un ensemble de plateformes représentatives des solutions (O/L/P-PM), obtenues par analyse système ou par conception. Ces plateformes sont modélisées par des concepts de solutions dont le niveau d'abstraction se conforme aux niveaux présentés dans le paragraphe 3.2.3.

Les modèles de plate-forme sont des éléments qui permettent de construire, une fois intégrés entre eux, le modèle spécifique de plate-forme, ou architecture à un niveau d'abstraction donné. Ils constituent les concepts de solutions globaux aux niveaux opérationnel, logique et physique. Chaque modèle de plate-forme possède ces caractéristiques internes qui doivent être prises en compte lors de la construction du modèle spécifique de solution.

Le processus descendant permettant d'obtenir les architectures (modèles spécifiques de plate-forme) est vu comme une transformation réalisée, au moyen des patrons de modélisation, à partir des modèles d'entrée. Ils correspondent respectivement aux activités de définitions des architectures opérationnelles, logiques, et physiques.

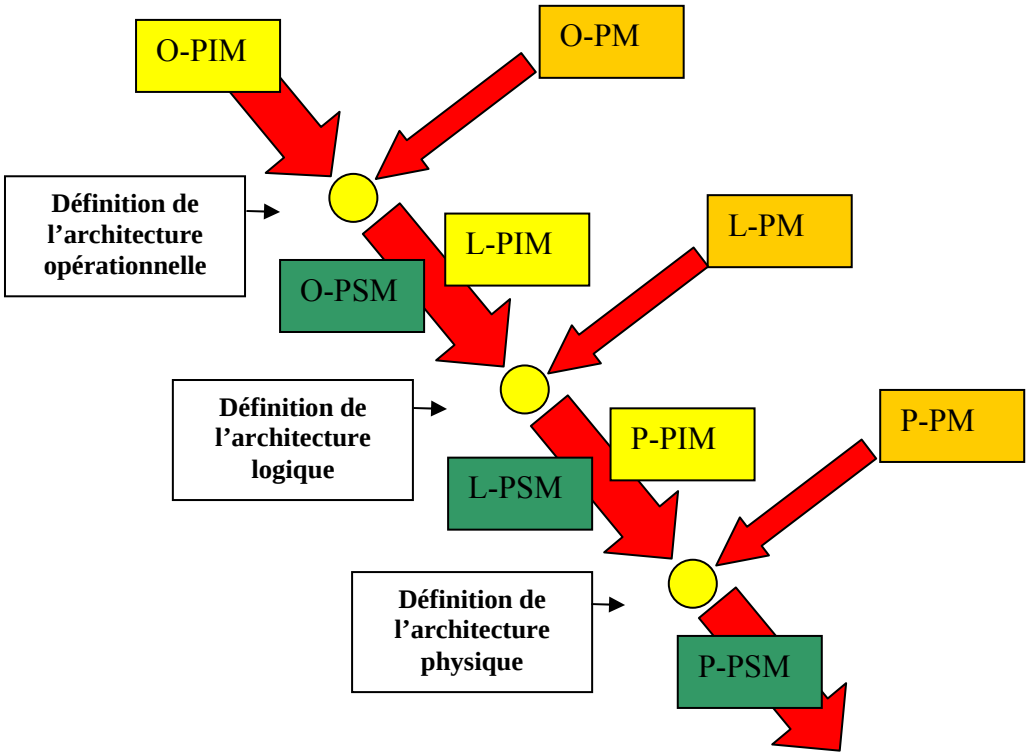


Figure 4.20. : Transformations MDA appliquées aux architectures opérationnelles, logiques et physiques, et Processus de définition des solutions

Lorsqu'un modèle indépendant de plate-forme est associé à un modèle de plate-forme, les tâches suivantes doivent être réalisées:

a) Définition des objets des exigences du modèle prescriptif sur le modèle de plate-forme.

b) Formulation des exigences en propriété(s) systèmes exprimable(s) par rapport aux modèles de plate-forme (PM). Dans le cas de la définition de solutions logiques, les mesures d'efficacité doivent être traduites en exigences / mesures de performances par la prise en compte des caractéristiques internes de la plate-forme logique. Celles-ci sont ajoutées à l'ensemble des exigences dérivées. Dans de cas de la définition physique, les mesures de performances doivent être traduites en mesure de performances techniques par la prise en compte des caractéristiques de la plate-forme physique.

c) Définition d'exigences dérivées du P.M., hors exigences de mesures de performances et de performances techniques. Les exigences dérivées par la plate-forme sont identifiées puis ajoutées à l'ensemble d'exigences dérivées à partir de ses caractéristiques internes.

d) Introduction dans le modèle des paramètres internes de la plate-forme. Ces paramètres permettent le dimensionnement de la plate-forme et la prise en compte des caractéristiques internes de celle-ci.

En réalité, le processus de définition d'une plate-forme spécifique n'est pas linéaire mais itératif. Ainsi, lorsqu'un modèle de plate-forme est choisi, une plate-forme spécifique incomplète peut déjà exister. L'apport du modèle de plate-forme permet de compléter la plate-forme spécifique existante, tel que représenté dans la figure 4.20.

La démarché décrit ci-dessus a été prise en compte lors de la définition des règles méthodologiques (introduits au paragraphe 4.3.3.) et des patrons de génération et d'extension de modèle (introduits au paragraphe 4.3.4).

Les règles et les patrons proposés pour mettre en œuvre les processus d'ingénierie système ont été regroupés en opérations d'ingénierie système et sont détaillés respectivement en annexe E et F.

4.5. Conclusion

La méthodologie MICCSA présentée dans ce chapitre offre un moyen d'assister le concepteur dans la réalisation des processus de définition des exigences et des solutions par l'utilisation d'un formalisme spécifique, et par l'utilisation de mécanismes permettant d'améliorer automatiquement la complétude et l'exactitude des modèles. Ces mécanismes ont été décrits en particulier pour la réalisation des activités d'analyse système et de validation. Ensuite, une approche qui tire parti de la formalisation des objectifs de validation pour réaliser un modèle de simulation apte à répondre à ces objectifs de validation a été décrit.

La chapitre V présente une mise en œuvre de la méthodologie MICCSA, en illustrant ses mécanismes d'ingénierie des modèles présentés dans ce chapitre à l'aide d'un outil de modélisation et d'un outil conforme à la norme MDA.

Chapitre 5. MODELES ET APPLICATION DE LA METHODOLOGIE MICCSA

Ce chapitre décrit comment les processus et les méthodes présentés au chapitre IV ont été appliqués au niveau des langages de modélisation et des outils à travers un cas d'étude. Cet exemple est présenté en tête de chapitre afin d'illustrer les modèles et l'application de la méthodologie tout le long de ce chapitre.

La mise en application de la méthodologie mise en application a consisté à :

- choisir des langages répondant au besoin de modélisation,

- adapter l'usage de ces langages ou limiter leur sémantique afin de rendre leur utilisation plus simple et plus conforme à la méthodologie,

- choisir les outils logiciels permettant leur mise en œuvre, ainsi que celle des mécanismes d'ingénierie des modèles définis dans la méthodologie.

Après une étude comparative des langages de modélisation système existants, deux langages de modélisation complémentaires ont été choisis pour proposer une mise en œuvre de la méthodologie. Le langage SysML a tout d'abord été choisi car sa sémantique couvre les besoin formalisés par le modèle d'information défini au chapitre précédent. En particulier, sa capacité à modéliser le domaine des exigences, le domaine des solutions sous la forme de description structurelle et comportementale le rend adapté à notre problématique. Toutefois, l'usage de SysML uniquement ne permet pas de couvrir intégralement le besoin pour les raisons suivantes :

- Les modèles réalisables dans ce langage ne sont pas simulables, et ne peuvent donc pas être utilisés comme modèles prédictifs de comportement, de performances ou de fiabilité, nécessaires aux activités d'analyses et de validation. Par conséquent, notre choix s'est également porté vers le langage VHDL-AMS pour répondre à cette limitation inhérente à SysML. VHDL-AMS a été utilisé de manière complémentaire aux modèles structurels SysML afin d'apporter la capacité de simulation.

- La sémantique de SysML est trop générique pour être utilisée efficacement dans le cadre de la méthodologie. Le choix de ce langage a nécessité la spécialisation de sa sémantique au moyen de la constitution d'un profil, dans l'objectif de répondre, d'une part aux besoins spécifiques de la méthodologie et d'autre part, au besoin d'interopérabilité avec le langage VHDL-AMS.

Cette démarche de spécialisation de la sémantique de SysML est décrite et justifiée dans le sous-chapitre 5.2. Les caractéristiques du langage VHDL-AMS permettant son intégration comme un outil dans la méthodologie sont ensuite décrites dans le sous-chapitre 5.3.

L'ingénierie des modèles de vérification de propriétés et de transformation de modèles a ensuite été réalisée respectivement à l'aide des langages Check et XTend. Ces deux langages, spécifiques au domaine MDA, sont associés à la plate-forme d'ingénierie des modèles *OpenArchitectureWare* qui sera également présentée. La mise en œuvre de l'aspect ingénierie des modèles est décrite dans le sous-chapitre 5.4.

5.1. Cas d'étude : système porte passagers

Ce sous-chapitre présente l'exemple principal qui a été utilisé pour appliquer la méthodologie MICCSA. L'exemple concerne une porte passager d'avion civil. Celle-ci est utilisée en tant que moyen :

- d'embarquement et de débarquement en situation normale d'exploitation,
- d'évacuation et d'accès au sol depuis la cabine en situation anormale, et d'urgence.

Les exigences applicables à ce système et ses principales caractéristiques sont présentées brièvement ci-dessous.

5.1.1. Exigences sources

Le système, porte passagers, étudié peut être considéré comme un produit complexe. Des exigences nombreuses et hétérogènes doivent être prises en compte. La source principale d'exigences du système, porte passagers est l'ensemble des autorités de certifications. Du fait de son utilisation potentielle pour l'évacuation des passagers, des exigences additionnelles doivent ainsi être considérées.

5.1.1.1. Exigences de certification

Les exigences règlementaires occupent une place importante parmi l'ensemble des exigences émises sur le produit. Dans les toutes dernières décennies, des incidents et accidents survenus suite à une mauvaise opération ou à des défaillances de portes passagers et soutes, ainsi que l'apparition ou l'accroissement de nouveaux risques dans le transport aérien, ont amené les autorités de réglementation américaine et européenne à faire évoluer notablement les exigences de certification concernant ce type de produit. La liste des textes règlementaires émis par les autorités de certification et applicables au système porte passagers est présentée sur le tableau 4.1.

FAR 25	14 CFR Part 25 Design Standards for Fuselage doors on Transport Category Airplanes; Final Rule Revision: Monday, May 3 2004
AC 25.783-1A	Advisor Circular "Fuselage doors and hatches" Issued 4/25/05
JAR 25	Part 25 Certification for large aeroplanes; Final Rule Revision: 1 November 2004
NPA 25D-301	Fuselage doors JAR.783 DRAFT. Issued January 2003

Tableau 5.1 : Textes règlementaires applicables pour la certification

5.1.1.2. Exigences client

A ces exigences de certification s'ajoutent les exigences fonctionnelles auxquelles doit satisfaire une porte passagers ainsi que des exigences sur la maintenabilité du produit, l'interchangeabilité de ses composants, ses performances en terme d'isolation thermique, de discrétion acoustique, d'ergonomie et de facilité d'utilisation, de qualité aérodynamique et d'esthétique.

Les exigences client peuvent être décomposées en deux sources d'exigences distinctes :

- Un ensemble d'exigences spécifiques au projet, fonctionnelles et non fonctionnelles (installation, support, fiabilité, sécurité) applicables sur les processus et le produit.
- Un ensemble d'exigences processus et produit génériques liées aux référentiel d'exigences client.

5.1.1.3. Exigences internes

Les métiers impliqués dans le développement du produit émettent des recommandations initiales pour optimiser le produit par rapport à leurs perspectives. En particulier, le métier support participe à la définition du concept de maintenance du produit qui sera intégré à l'architecture opérationnelle.

Exigence système d'évolutivité :

L'évolutivité désigne l'aptitude du système à être modifié afin de prendre en compte de nouvelles attentes fonctionnelles du client ou une évolution des exigences de certification. Appliqué au système porte passagers, les exigences sources d'évolutivités sont déclinées en exigences systèmes de disposition supplémentaires, (définition en annexe B).

Exigences de communalité :

La caractéristique de *communalité* désigne la capacité du système produit à être adaptable à divers environnements opérationnels et techniques avec un minimum d'évolution de son architecture physique.

Exigence d'installabilité :

Elle désigne la capacité du système à être installé en un minimum de temps, de moyens et de ressources. La caractéristique d'*installabilité* est une exigence majeure pour ce type de produit. Les produits existant nécessitent une phase importante de réglages mécaniques et

électroniques, avant, et lors de leur installation sur l'avion. L'exigence d'installabilité conduit à écarter ou à pénaliser les éléments de solutions exigeant des phases de réglage et d'étalonnage.

Exigence de maintenabilité :

Celle-ci désigne l'aptitude du système à être maintenu facilement. Les exigences systèmes peuvent répondre à cet objectif en :

- ajoutant des éléments de diagnostic et de surveillance à l'architecture logique,
- améliorant la modularité et le découplage des constituants physiques de l'architecture,
- adaptant les interfaces physiques aux opérations de maintenance, notamment du point de vue de l'ergonomie (position opérateur, détrompages, etc.).

5.1.1.4. Autres exigences

D'autres sources d'exigences ont été prises en compte. En particulier, le standard *Aerospace Recommended Practice 488* du SAE regroupe un ensemble de recommandations techniques pour la conception des portes passager des avions de transport.

Ces recommandations sont prises en compte comme des exigences de faible niveau de priorité dans le développement afin de réduire le risque de non-certifiabilité du système.

5.1.2. Eléments principaux du système porte passagers

Les portes passagers existantes intègrent étroitement des éléments d'aérostructure et d'équipements. Un tel produit se prête donc particulièrement bien à l'application de la méthodologie MICCSA et à l'approche système adoptée par celle-ci. Le produit possède une complexité suffisante pour constituer un cas d'étude pertinent. Pour une meilleure compréhension de la problématique de la conception du système, les principaux éléments qui doivent être étudiés sont présentés de manière qualitative ci-dessous.

5.1.2.1. Moyens de sécurisation exigés par la certification

Afin de satisfaire aux exigences de certification, les portes passagers possèdent deux moyens distincts de sécurisation de la position *porte fermée*. L'indépendance de fonctionnement de ces moyens appelés *latch* et *lock* doit également être démontrée par analyse.

Le mécanisme de *latch* sert à maintenir la porte dans sa position fermée, et à éviter qu'un ensemble d'éléments défavorables réunis n'amènent à l'ouverture de la porte. Le mécanisme de *latch* doit, en outre, disposer de son propre moyen de sécurisation, indépendant du mécanisme de *lock*. Le dispositif *lock* est utilisé pour sécuriser le maintien du *latch* sur la porte. Cette exigence sur les moyens à mettre en œuvre pour assurer les fonctions de *latch* et de *lock* est conforme à l'exigence générique de *panne simple*. Celle-ci stipule qu'une panne simple (un événement unique, suivi éventuellement des événements découlant directement du premier) ne doit pas provoquer un événement catastrophique sur le système complet.

Deux exigences contraignent le fonctionnement interdépendant de ces deux moyens :

- Le moyen de *lock* ne doit pouvoir être activé que si le dispositif de *latch* a été correctement activé.
- Le moyen de *latch* ne doit pas pouvoir être ouvert si le mécanisme de *lock* est en position fermé.

Enfin, l'ensemble de l'utilisation de ces mécanismes ne doit pas compliquer l'opération du système porte passager. Cela peut être traduit par le fait que l'interface utilisateur contrôlant ces mécanismes doit être commune et partagée avec la commande d'ouverture de porte.

5.1.2.2. Moyen d'évacuation

Le système étudié est aussi utilisable en cas d'évacuation des passagers. Il inclut, pour cela, un moyen d'accès au sol depuis la cabine capable d'être mis en place automatiquement, ainsi qu'un moyen d'assistance à l'ouverture de l'avion. Il est principalement utile dans le cas particulier où la position angulaire de l'avion, et/ou une déformation mécanique de la structure exigerait un effort important de l'opérateur. Le moyen d'accès au sol et le moyen d'assistance à l'ouverture doivent respecter ensemble des contraintes de temps, justifiées dans le contexte d'une évacuation en situation d'urgence. Un moyen de maintien de la porte en position ouverte est aussi exigé.

5.1.2.3. Moyens de surveillance de l'état du système

Les autorités de certification exigent également que le système porte passagers dispose d'un moyen de surveillance de sont état à destination de l'opérateur et d'une capacité de prise de décision pour la commande des différents actionneurs éventuels qu'il comporte. De plus, le système doit être en lien permanent avec le réseau de bord de l'avion afin de transmettre les informations au poste de pilotage.

La réglementation impose la mise en place d'un moyen de prévenir l'ouverture intentionnelle ou par inadvertance d'une porte passagers en vol par une personne. Ce système vient s'ajouter aux mécanismes de *latch* et de *lock* précités.

5.2. Profil SysML adapté à la méthodologie

Ce sous-chapitre présente comment des langages de modélisation ont été utilisés pour mettre en œuvre la méthodologie et pour réaliser les modèles, notamment les modèles du cas d'étude.

5.2.1. Démarche de spécialisation de SysML à la méthodologie

Plusieurs approches ont été envisagées afin de spécialiser l'usage du langage SysML à la méthodologie ainsi qu'au domaine d'application. Plus précisément, la spécialisation du langage avait comme objectif de :

- permettre le support du modèle d'information (ou métamodèle) associé à la méthodologie MICCSA et défini au chapitre 3.2,

- supporter les objectifs d'interopérabilité des modèles en adaptant, si nécessaire, la sémantique d'origine du langage à une sémantique plus restrictive / spécifique, permettant d'aborder le passage d'un langage à un autre comme une étape d'ingénierie de modèles MDA,

- permettre d'adapter la sémantique générique de SysML au domaine de produits étudiés et, en particulier, à celui de l'exemple principal présenté dans ce chapitre.

Pour parvenir à ces objectifs, trois alternatives de spécialisation du langage SysML pour la méthodologie et sa mise en application sur l'exemple ont été étudiées :

La première alternative consiste à définir un métamodèle, instancié pour chaque projet. L'ensemble des spécificités liées aux points précédents doit, dans ce cas, être modélisé au travers d'un métamodèle. Les concepts de la méthodologie sont modélisés sous forme de types spécialisés à partir des types fournis par la spécification SysML. Pour les descriptions d'architecture, il s'agit donc essentiellement de spécialiser un ensemble de blocs, de ports, de connecteurs et de spécifications de flux. Le modèle est alors instancié par les projets. Les concepts spécifiques manipulés au niveau du projet sont ensuite définis par des relations d'héritage à partir des classes prédéfinies comme éléments du modèle. Dans cette alternative, la définition de types permet donc à la fois de modéliser les concepts de la méthodologie et les éléments génériques partagés entre les projets.

Une autre alternative envisagée est la spécialisation du langage par la définition d'un profil. La spécialisation s'appuie alors sur la définition :
- de stéréotypes,
- de valeurs annotées (*tagged values*),
- de contraintes spécifiant les conditions d'usages des éléments ci-dessus,
- d'éléments de notations graphiques (c.à.d. la syntaxe concrète du profil).

Un avantage notable de cette seconde approche est de pouvoir restreindre, dès sa définition, le contexte d'utilisation des éléments du langage afin de faciliter l'élaboration de modèles valides.

Enfin, une dernière approche possible est de combiner les deux précédentes. Cette approche permet de profiter des avantages propres aux deux approches ci-dessus. Ainsi, la définition d'un profil réalisé sur la base de stéréotype et de valeurs annotées permet de supporter les concepts présents dans le métamodèle de la méthodologie (cf. Chap III.3). Des contraintes associées aux stéréotypes et aux valeurs annotées permettent de restreindre l'usage des éléments du profil pour assurer leur utilisation correcte.

Ensuite, les éléments du profil sont importés et utilisés dans les modèles. L'utilisation de plusieurs niveaux de types dans la méthodologie permet de structurer la connaissance acquise autour des modèles. Les types de blocs utilisés pendant la définition des architectures logiques et physiques héritent de blocs capitalisés, possédant leurs caractéristiques internes et leurs définitions d'interface. Ces blocs peuvent être organisés entre eux par des liens de généralisation et de spécialisation. Ces blocs sont alors instanciés dans la description d'architecture.

Cette dernière alternative, représentée sur la Figure 5.1 a été retenue pour mettre en application la méthodologie avec SysML et réaliser l'exemple.

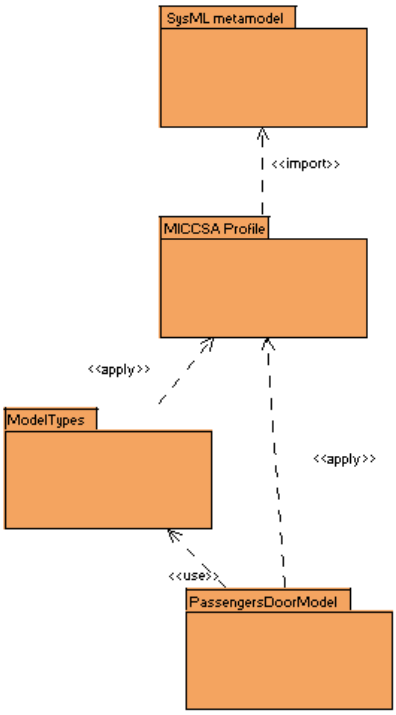


Figure 5.1 : Démarche de spécialisation de SysML retenue.

Les éléments du modèle d'informations MICCSA sont donc représentés par :

- Des éléments du langage SysML directement utilisés, lorsque cela est possible. Les éléments du langage SysML sont utilisés principalement pour réaliser les points de vue et les exigences textuelles.

- Des éléments d'un profil SysML (profil *MICCSA*) : La définition d'un profil du langage SysML permet de préciser la sémantique en fonction des besoins de la méthodologie.

Les éléments caractéristiques de MICCSA et des éléments de profil SysML sont définis dans le tableau section 4.2.2. Cette section présente de manière synthétique comment les principaux éléments du modèle d'informations sont couverts par le profil SysML.

5.2.2. Organisation du modèle du projet

Dans le contexte de la modélisation de la structure du projet, la définition du profil permet notamment d'exprimer :

- les ensembles d'exigences regroupées sous la forme de spécifications conformes aux processus de la méthodologie,
- les ensembles de représentations des solutions logiques et physiques.

Ces ensembles sont intégrés dans un *package* qui regroupe l'ensemble des données associées à un bloc de construction tel que présenté Figure 5.1. Un tel paquetage est accompagné d'attributs permettant de décrire le périmètre technique couvert par le bloc, ainsi

qu'une description qualitative du niveau de détails permettant d'assurer la cohérence des descriptions contenues dans ce paquetage. Les contraintes associées à chacun de ces paquetages spécialisés sont présentées en annexe G.

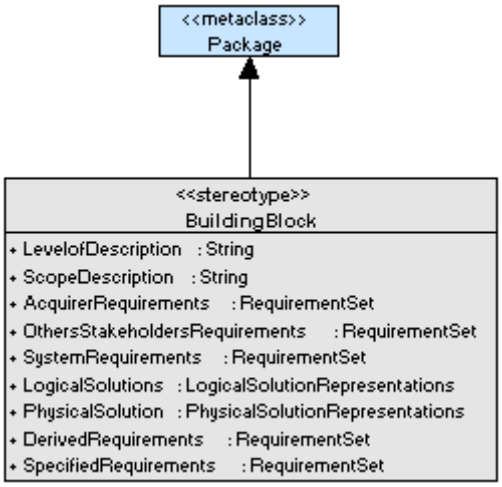


Figure 5.2 : Définition du stéréotype bloc de construction

Les stéréotypes définis pour l'organisation du modèle du projet sont résumés par le tableau 4.1.

Elément du modèle d'information MICCSA	**Elément stéréotypés**	**Remarques et justification**
BuildingBlock	Package	Un paquetage spécifique permet de décrire dans une unité le concept de bloc de construction.
RequirementSet	Package	Un ensemble d'exigences constituant une spécification sont regroupés sous la forme d'un paquetage spécifique.
LogicalSolution Representation	Package	Les descriptions logiques associées à un *building block* sont regroupées dans un paquetage spécifique
PhysicalSolution Representation	Package	Les descriptions physiques associées à un building block sont regroupées dans un paquetage spécifique

Tableau 5.2 : Eléments de structure du profil SysML-MICCSA

La Figure 5.3 présente le bloc de construction du projet (bloc système) appliqué à l'exemple porte-passagers. Les exigences sources sont constituées des exigences clients, des exigences de certification, des recommandations techniques et des exigences métiers.

Pour illustrer les parties suivantes, un deuxième niveau de blocs de construction est représenté sur la Figure 5.4. Ce deuxième niveau représente les différents sous-ensembles fonctionnels constitutifs de la porte. Chacun de ces sous-ensembles doit être considéré avec ses propres éléments d'exigences parties prenantes. Le sous-ensemble correspondant au moyen d'armement toboggan est illustré plus en détail dans ce chapitre. Les autres sous-ensembles modélisés représentent le moyen d'assistance à l'ouverture de la porte et les sous-ensembles d'ouverture / fermeture et *latch* / *lock*.

Une partie des exigences spécifiées du bloc de construction principal sont affectées au bloc de construction *armement toboggan*.

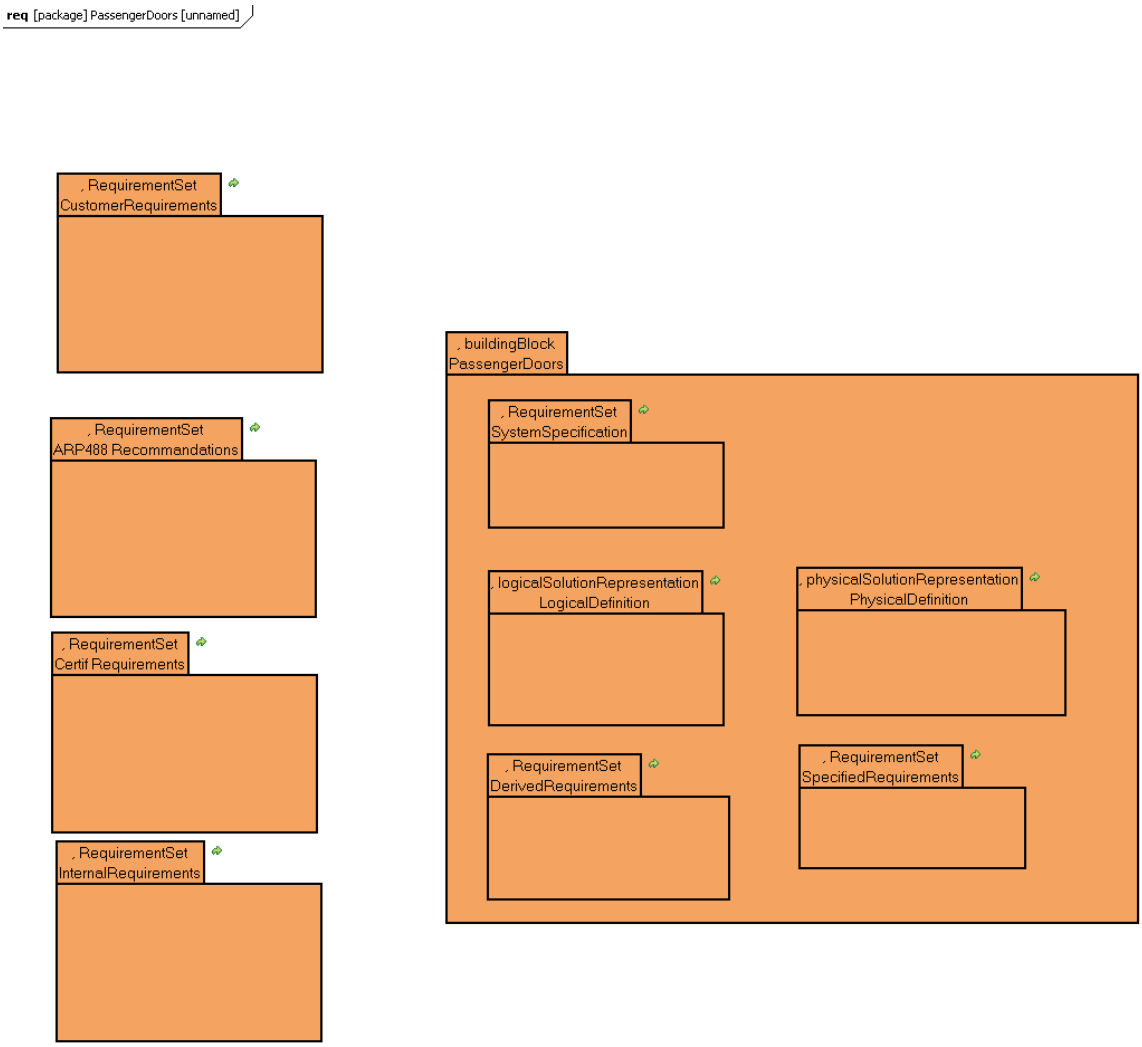


Figure 5.3 : Partie modèles constituants du projet

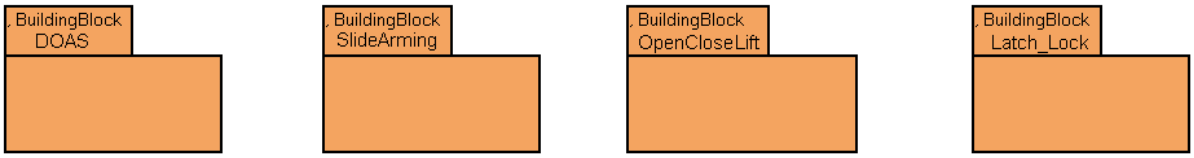


Figure 5.4 : Bloc de construction de niveau 2

5.2.3. Modélisation des exigences projet

Le profil SysML inclut des éléments permettant la modélisation des exigences tels que décrits dans le modèle d'informations de la méthodologie. Cette modélisation inclut la formalisation des exigences sources, techniques système, dérivées et spécifiées.

Les blocs d'exigences permettent de définir les différentes catégories d'exigences utilisées à chaque niveau. La définition du profil permet d'associer à ces blocs :

- le type d'exigences (sources, affectées, systèmes, dérivées, spécifiées),
- la catégorie d'exigence, (cf. Chapitre III, tableau 3.1 pour le niveau système).
- des attributs d'ingénierie d'exigences adaptés,
- un ou plusieurs bloc(s) de contrainte contenant la prescription de l'exigence. Celle-ci contient l'élément de liaison formalisé avec le reste du modèle, en particulier les éléments d'architectures opérationnelles, logiques et physiques,

Les exigences sont liées entre elles et aux éléments des architectures opérationnelles, logiques et physiques par les relations standards SysML. Néanmoins, le profil permet d'associer à chaque relation une contrainte permettant de s'assurer que le type de relation est utilisé à bon escient. La Figure 5.5 présente les relations entre les exigences et les éléments architecturaux dans le profil SysML que nous avons défini.

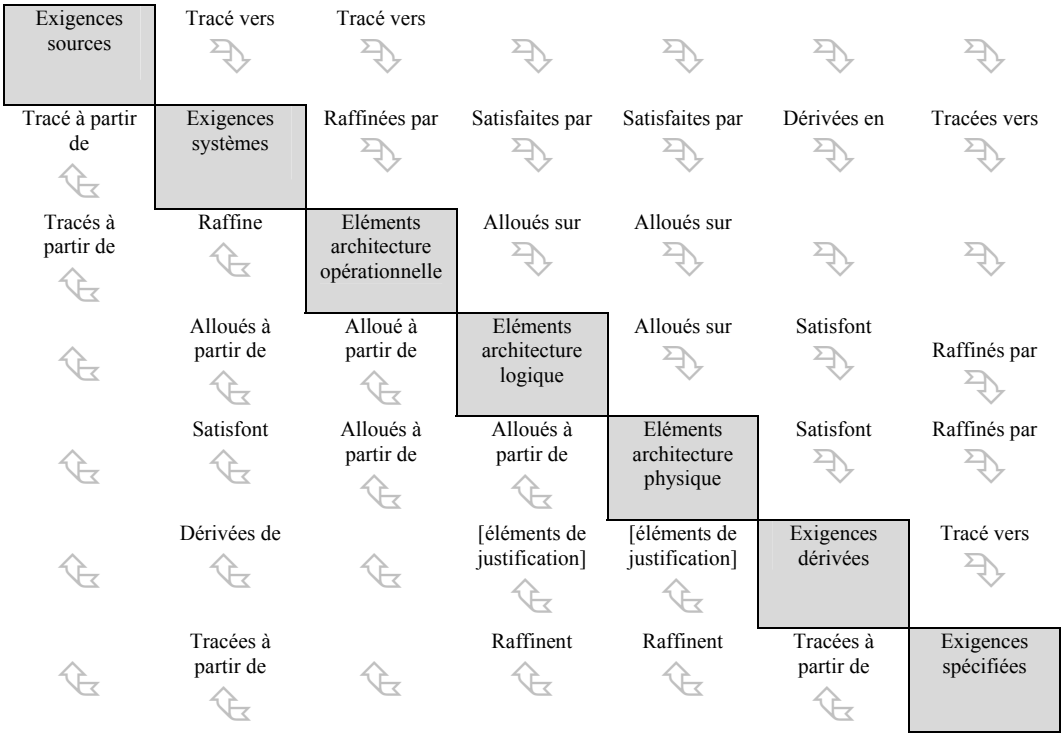


Figure 5.5 : Relations d'ingénierie d'exigence dans le profil SysML-MICCSA

Le lien *derive* est défini entre une exigence système et une ou plusieurs exigences dérivées. Les exigences dérivées sont identifiées par le biais des solutions logiques et physiques et sont liées avec elles par des éléments de justification.

Le lien *satisfy* est utilisé pour identifier les éléments des solutions logiques et physiques qui satisfont les exigences systèmes et les exigences dérivées.

Le lien *refine* est utilisé pour identifier les exigences spécifiées à partir des solutions logiques et physiques. Il est aussi utilisé pour établir l'architecture opérationnelle à partir des exigences système.

Les éléments d'architecture opérationnelle, logique et physique sont liés entre eux par des liens d'allocation.

Enfin, les éléments des architectures logiques et physiques sont utilisés comme éléments justificatifs des exigences dérivées et spécifiées.

Les éléments du profil pour l'ingénierie des exigences sont résumés dans le tableau 4.3.

Elément du modèle d'information MICCSA	**Elément du langage SysML**
Exigence	Bloc d'exigence, stéréotypes de catégories d'exigences
Allocation	Lien d'allocation
Lien de décomposition	Relation *refine*
Lien de dérivation	Relation *derive*
Lien de satisfaction	Relation *satisfy*
Type d'exigence	Stereotypes *source, assigned, system, derived, specified*
Prescription	Bloc de contrainte associée au bloc d'exigence, Contrainte
Eléments de justification	stéréotype *requirement rationale*

Tableau 5.3 : Eléments d'ingénierie d'exigences du profil SysML-MICCSA

Lors du traitement de l'exemple porte passagers, un ensemble d'exigences système à été défini. Parmi celles-ci, plusieurs exigences sont présentées ci-dessous. Ces exigences sont représentatives de l'ensemble des exigences système traitées. Elles ont été formalisées puis vérifiées sur les modèles d'architecture logique et/ou physique selon les techniques présentées à la section 4.4.1. Les caractéristiques de complétude et d'exactitude des alternatives d'architectures proposées ont ainsi pu être évaluées.

Commençons par les exigences systèmes provenant des exigences sources réglementaires :

a) *External doors must have provisions to prevent the initiation of pressurization of the airplane to an unsafe level if the door is not fully closed and locked.* [From FAR25.783]

Ce type d'exigence spécifie la présence d'une fonction spécifique qui doit être identifiée dans l'architecture logique puis réalisée sur l'architecture physique.

b) *Latching locking mean must be functionally and physically independent.* [From FAR25.783]

Ce type d'exigence spécifie à la fois une contrainte d'indépendance entre deux moyens de réalisation sur les architectures logiques et physiques. Elle est formalisée par le découplage entre fonctions techniques et entre constituants d'architecture physique. Enfin, elle induit

aussi des conditions (absence de causes communes et de modes de défaillances communs) sur les modèles d'ingénierie spécialisée sécurité.

c) *It must be shown by safety analysis that inadvertent opening is extremely improbable*. [From FAR25.783]

Cet autre type d'exigence impose un moyen de vérification (analyse de sécurité) associé la réalisation de la fonction d'ouverture :

d) *The handle shall be a hinged device measuring at least 5 inches (12.7 cm) wide.* [From ARP488]

Cette autre exigence provient des recommandations techniques ARP488 et apporte une contrainte à la fois sur le type de constituant à utiliser : *hinge device* et sur une caractéristique technique de ce constituant de l'architecture physique (*width*).

e) *Overall weight should not exceed 90 Kg.* [From Customer Requirements]

Enfin cette dernière exigence peut être évaluée en réalisant la somme des poids associés aux constituants physiques de l'architecture. Elle correspond à une métrique technique qu'il est nécessaire d'évaluer dans la phase de sélection de l'architecture physique.

Ainsi, les types principaux d'exigences système sont :

- la sécurité qualitative et quantitative,
- la métrique technique de performance (poids),
- les choix techniques demandés par le client ou par une recommandation technique ; ils ont été modélisées et formalisées conformément aux modèles d'information de la méthodologie.

Plusieurs autres exigences concernant d'autres propriétés système ont été vérifiées. Ces propriétés entrent dans les catégories suivantes :

- Propriété de redondance. Une fonction (ou un sous-ensemble de fonctions) de l'architecture logique est implémentée par plusieurs moyens de réalisation sur l'architecture physique.
- Propriété de non-similarité. Elle correspond à plusieurs moyens de réalisation d'une même fonction (redondants) différents de par leurs composants et leur architecture.
- Propriété d'isolation / ségrégation fonctionnelle. Deux moyens de réalisation n'ont pas de constituant (composant et interface) en commun, ou n'ont pas de cause commune ou de mode commun de défaillance
- Propriété sur le type d'éléments d'architecture : un moyen de réalisation inclut un type de réalisation particulier défini comme un type de composant ou d'interface.
- Propriété sur les caractéristiques d'un ou plusieurs éléments d'architecture : une exigence de performance ou de fiabilité est modélisée sous la forme d'un attribut dans l'architecture logique et d'une valeur minimale ou d'un intervalle autorisé pour la valeur associée à cet attribut.

- Propriété sur les caractéristiques d'un modèle comportemental. Il s'agit d'exigences sur la dynamique associée à une fonction ou à un ensemble de l'architecture physique.

La formalisation des exigences décrites ci-dessus est réalisée à l'aide du langage Check apparenté OCL pour la vérification de contraintes.

5.2.4. Modélisation de l'architecture opérationnelle

Dans le profil SysML que nous proposons, l'architecture opérationnelle est modélisée sous la forme de cas d'utilisation et de scénarios associés, ainsi que de diagrammes représentant l'utilisation du système par les opérateurs avec les différents modes d'utilisation et de fonctionnement. A titre d'exemple, la figure 5.6 donne les principaux cas d'utilisation de la porte passager de notre cas d'étude. La Figure 5.7 décrit, par un diagramme de séquence, la séquence normale de l'armement toboggan.

Plusieurs concepts d'opérations ont été identifiés à partir des cas d'utilisation présentés sur la Figure 5.6 et des scénarios opérationnels suivants :

- installation, réglage / étalonnage du système,
- actions de maintenances planifiés et non planifiées,
- utilisation en conditions normales,
- utilisation en conditions dégradées,
- retrait de service.

Les scénarios opérationnels ont permis d'identifier les concepts d'opération et les concepts de maintenance de l'architecture opérationnelle. Ainsi, ils ont donné la possibilité d'aborder la définition des solutions logiques et physiques avec un niveau d'abstraction adapté. Ils ont permis également d'établir et de valider les exigences opérationnelles système parallèlement à la définition des exigences système.

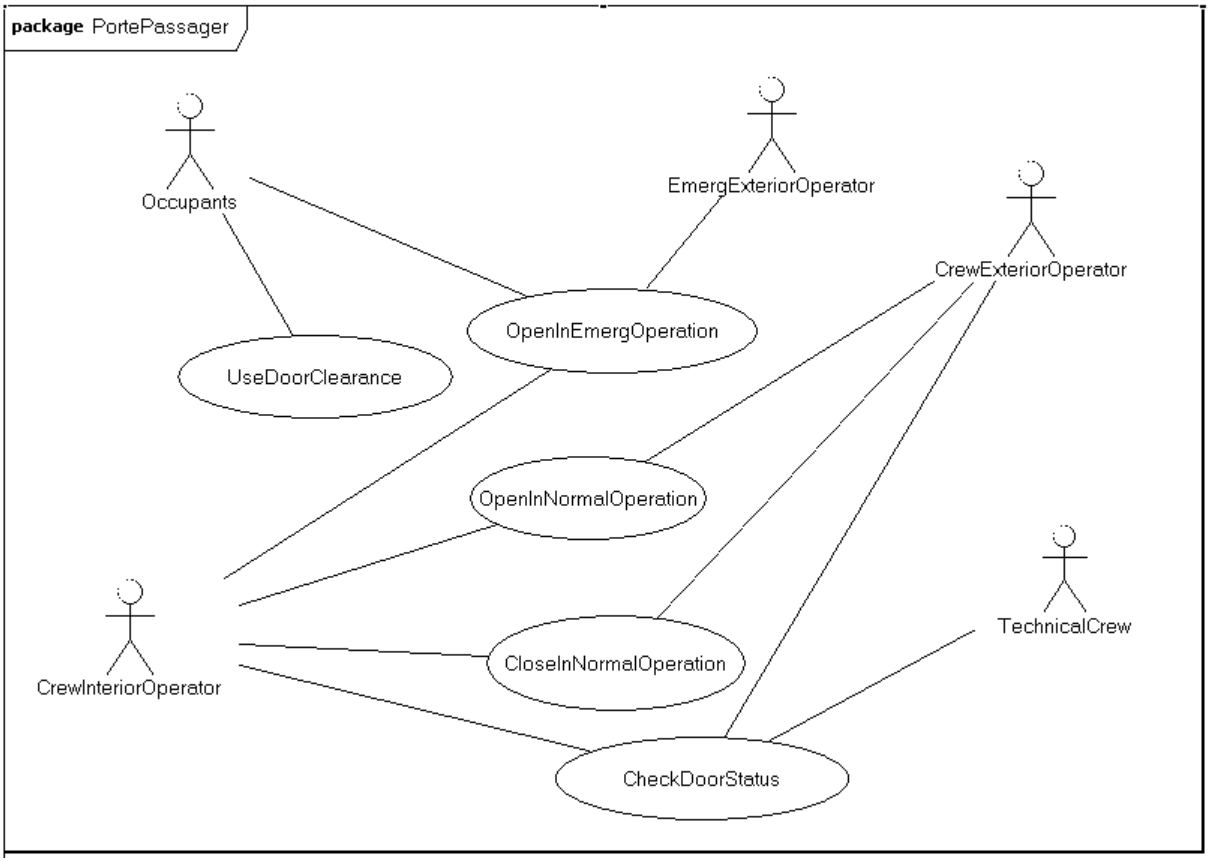


Figure 5.6 : Principaux cas d'utilisation en phase d'exploitation

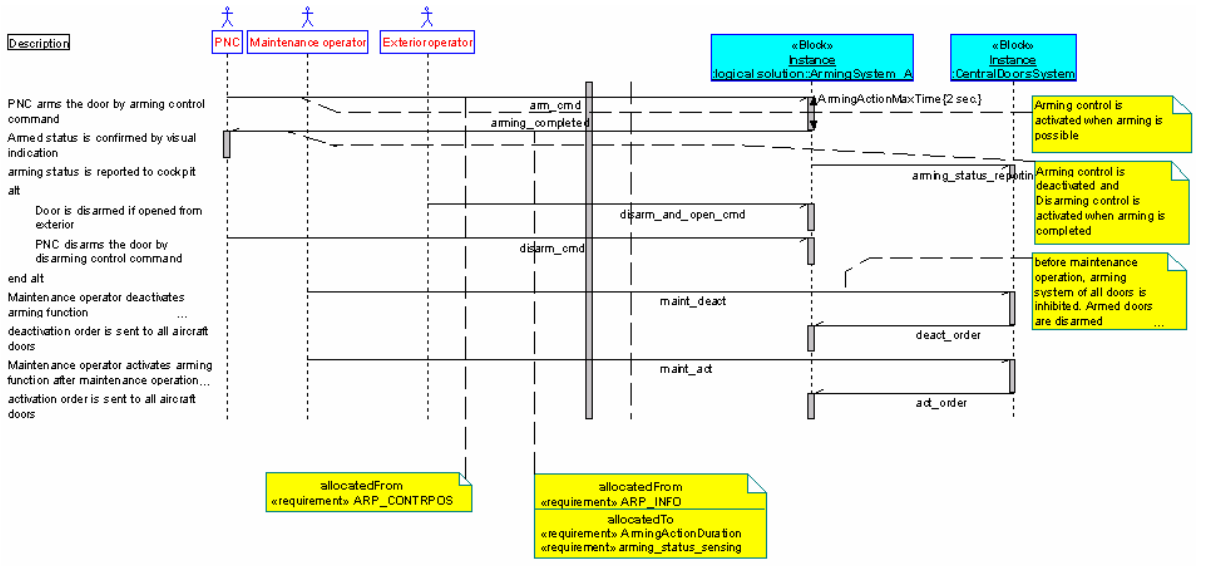


Figure 5.7 : Scénario d'utilisation normale de l'armement toboggan

5.2.5. Modélisation des architectures logique et physique

Dans le profil proposé, les descriptions d'architectures se font principalement à l'aide de bloc, de ports et de connecteurs. Ces éléments sont associés à des données d'ingénierie métiers produites par les activités d'analyse et de conception de ces métiers. Dans le profil présenté ici, les données d'ingénierie peuvent également être intégrées en support au modèle sous la forme de commentaires spécialisés et associés aux éléments des architectures.

Le bloc est l'élément principalement utilisé pour représenter les constituants logiques et physiques. Son usage est spécialisé dans l'objectif :

- d'adapter sa sémantique aux types d'architectures prévues dans la méthodologie et notamment aux niveaux d'abstraction, tels que représentés sur la Figure 5.8.

- de spécialiser (donc, de contraindre) le contexte d'utilisation et d'instanciation des blocs en fonction des méthodes permettant la réutilisation et la capitalisation,

- d'associer à ces blocs des éléments permettant de supporter les activités d'analyse et de conception au niveau des métiers.

Le bloc est donc le support d'information permettant de réaliser le lien entre, d'une part un constituant défini dans le modèle d'ingénierie système et, d'autre part les modèles métiers et les données d'ingénieries concernant ce constituant. Il permet de spécifier le besoin d'analyse et de conception au niveau métiers et de rassembler les données d'ingénierie remontées par ces derniers (la géométrie, les caractéristiques de résistance aux contraintes mécaniques, à l'environnement, la représentation 3D, etc.). Ces données descriptives sont définies sous forme de données d'ingénierie métier dans notre profil.

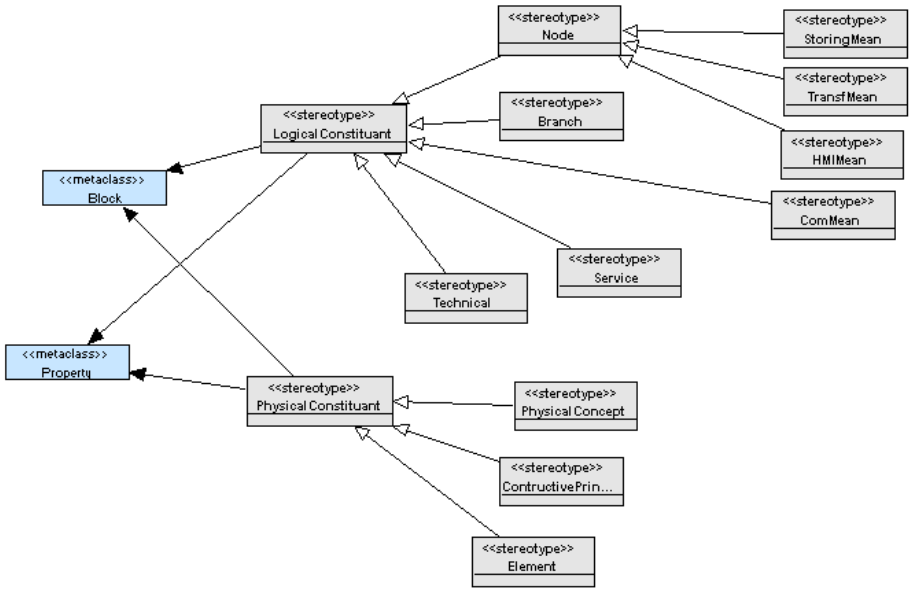


Figure 5.8 : Niveaux d'abstraction des blocs logiques et physiques du profil SysML

Les éléments de ports d'interfaces et les connecteurs sont également utilisés dans les représentations des structures logiques et physiques.

5.2.5.1. Exemple de modélisation de l'architecture logique

Une architecture logique de haut niveau est présentée sur la Figure 5.9. Il s'agit de l'architecture des fonctions de services attendus au niveau global pour le système porte passagers.

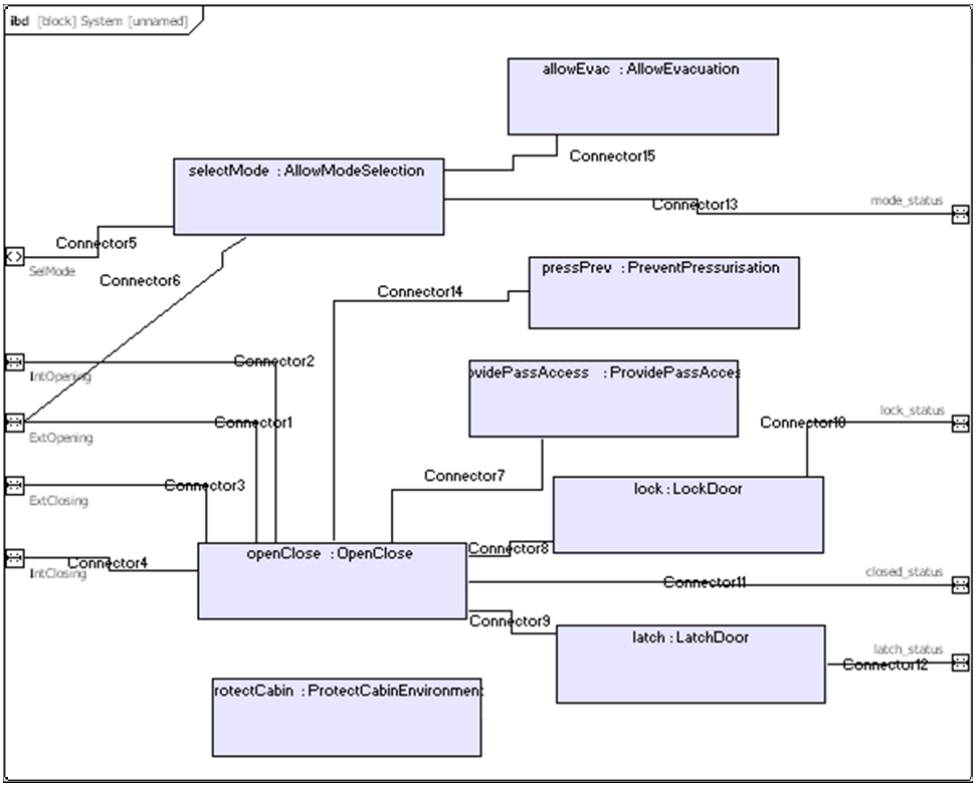


Figure 5.9 : Architecture logique système porte passagers

Cette architecture logique a permis :

- d'identifier les concepts principaux d'architecture physique du système porte passagers, et un découpage physique en blocs de construction tel que présenté à la section 4.2.2.
- d'associer à chaque constituant logique de l'architecture les caractéristiques attendues. Celles-ci ont été déclinées sur les constituants de niveau d'abstraction plus faible, tel que présentés sur la figure 5.10.

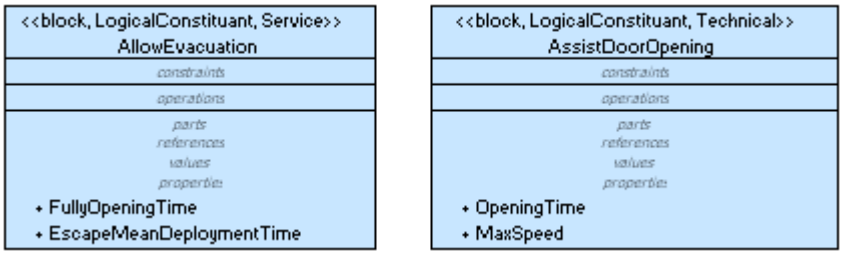


Figure 5.10 : Caractéristiques des constituants d'architecture logique

Un des sous-ensembles du découpage physique réalisé sur le cas d'étude est le mécanisme d'armement toboggan. L'architecture logique de ce sous-ensemble a été modélisée. Celle-ci est représentée en figure 5.10. L'architecture logique permet :

- L'identification des fonctions de service et des fonctions techniques du sous-ensemble fonctionnel d'armement toboggan.
- L'adaptation des concepts de solution logique au contexte de modélisation.

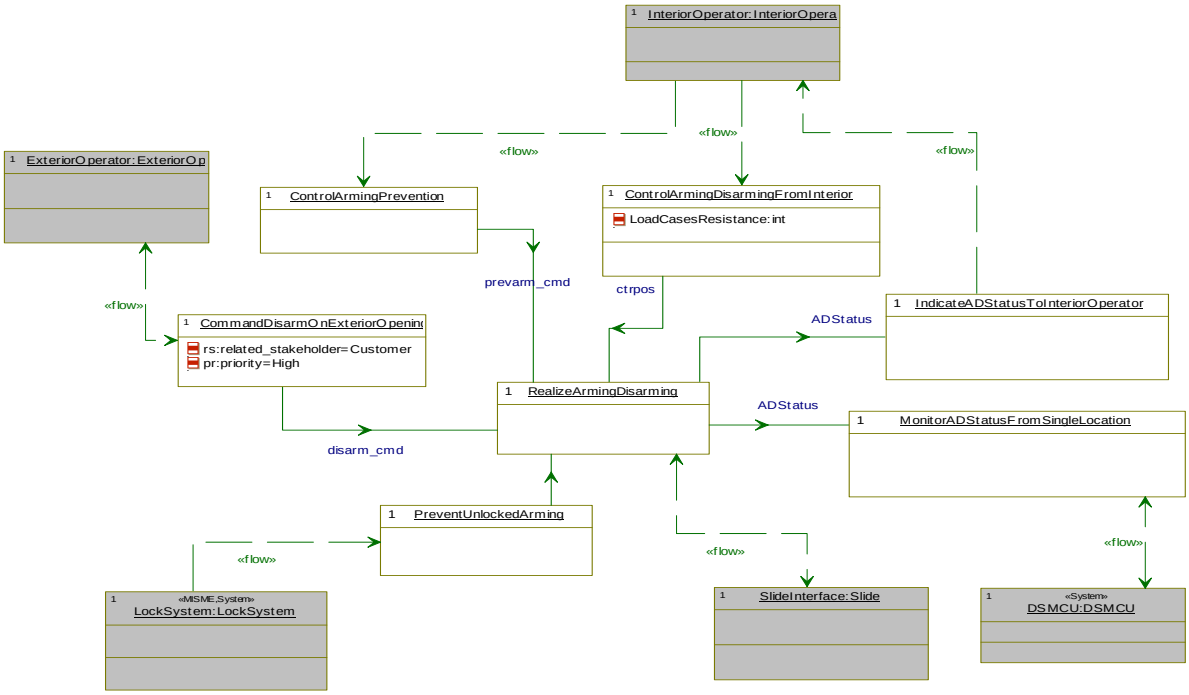


Figure 5.11 : Architecture logique sous-ensemble fonctionnel armement toboggan

5.2.5.2. Exemple de modélisation de l'architecture physique

Les éléments du profil SysML pour la modélisation de l'architecture physique permet de modéliser le système à concevoir aux différents niveaux d'abstraction spécifiés par la méthodologie.

Par exemple, à un bloc physique peut être associé un ensemble de caractéristiques internes sous la forme d'attributs tel que représenté sur la figure 5.11, ainsi qu'une description réalisée par un outil CAD ou un outil de simulation sous la forme d'un commentaire stéréotypé et rattaché au bloc. Chacun de ces outils permet, par son utilisation dans les phases d'analyse et de validation, de fixer la valeur des attributs du bloc et ainsi d'accroître la complétude et l'exactitude du modèle.

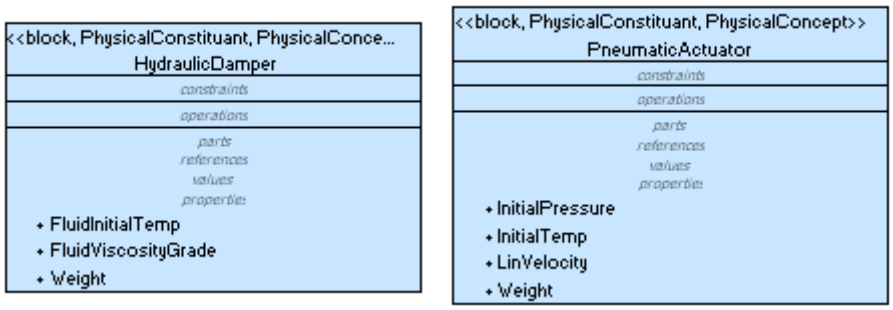


Figure 5.12 : Caractéristiques des constituants d'architecture physique

Un bloc physique peut, par exemple, contenir des caractéristiques internes qui pourront être évaluées par simulation au moyen d'un modèle VHDL-AMS simulant le bloc. Les liens réalisés entre ces blocs et les éléments de modèles d'analyse permettent de définir les éléments à observer dans le modèle VHDL-AMS.

La définition des solutions logiques a été développée sous la forme de diagrammes descriptifs de blocs et d'interfaces logiques.

Un ensemble de concepts de solutions ont été modélisés pour être intégrés à l'architecture logique et à l'architecture physique. Les concepts de solutions modélisés intègrent les aspects logiques et physiques et un lien d'allocation entre les deux (exemple, voir Figure 5.11).

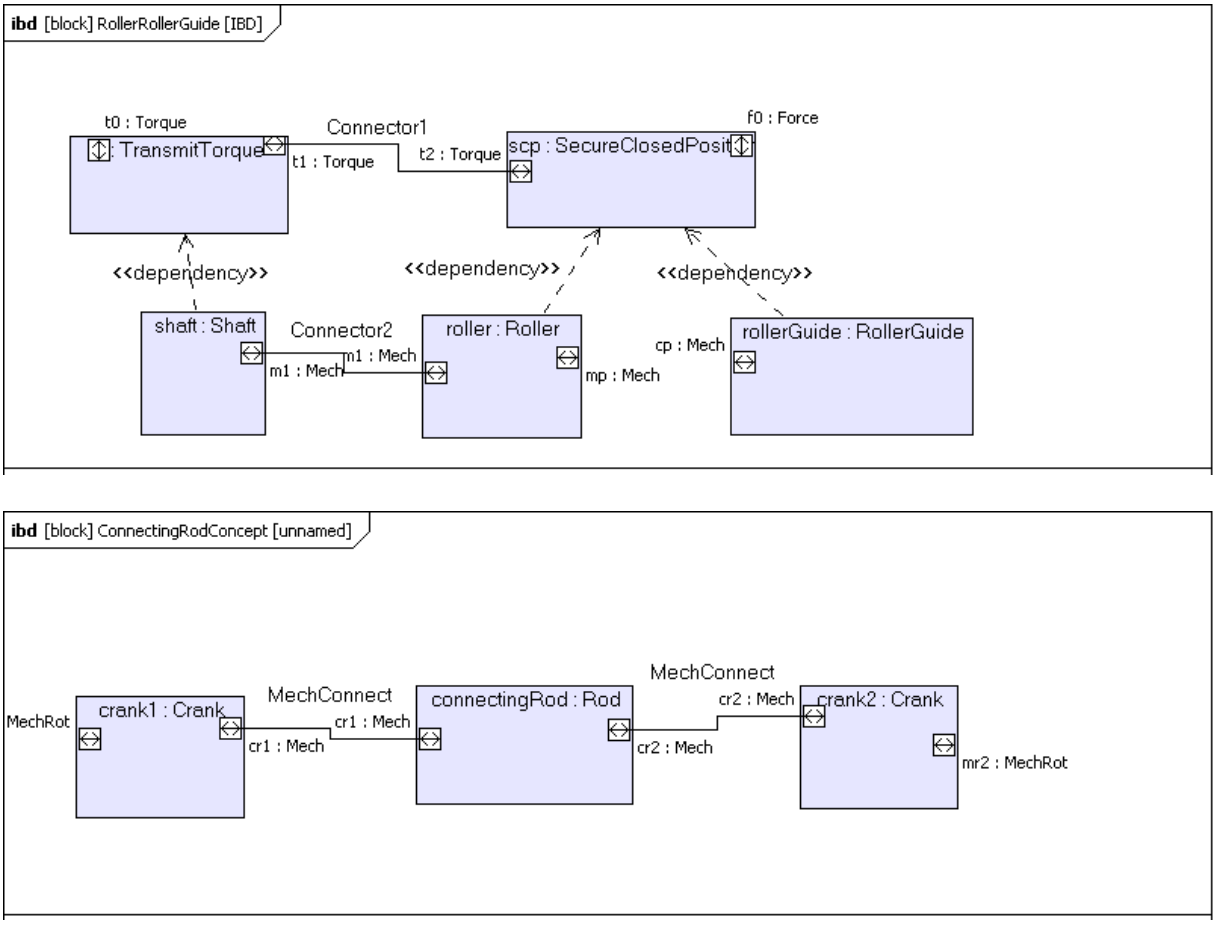


Figure 5.13 : Exemple de concepts de solution

Plusieurs solutions physiques ont été modélisées, décrivant le système réel, ainsi que plusieurs sous-ensembles physiques réalisant une partie du périmètre couvert par l'architecture logique.

La vue analyse système permet de montrer l'impact de la sélection de concept de solution spécifique sur l'architecture logique. La figure 5.14 permet de voir comment les fonctions techniques du sous-ensemble d'armement toboggan sont projetées sur les nœuds de l'architecture physique.

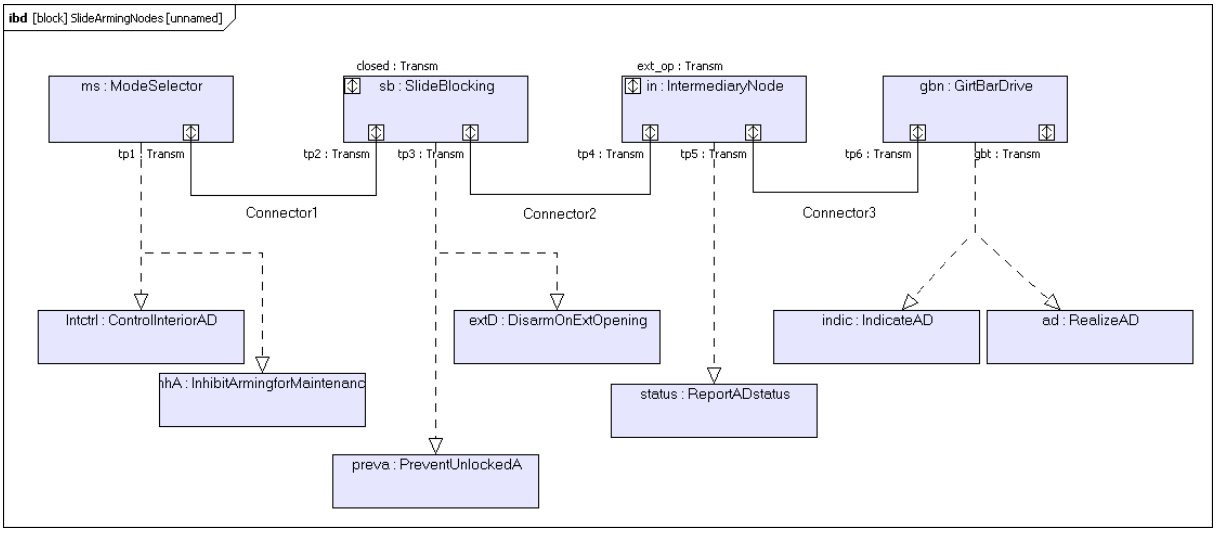


Figure 5.14 : Topologie du sous-ensemble fonctionnel armement toboggan

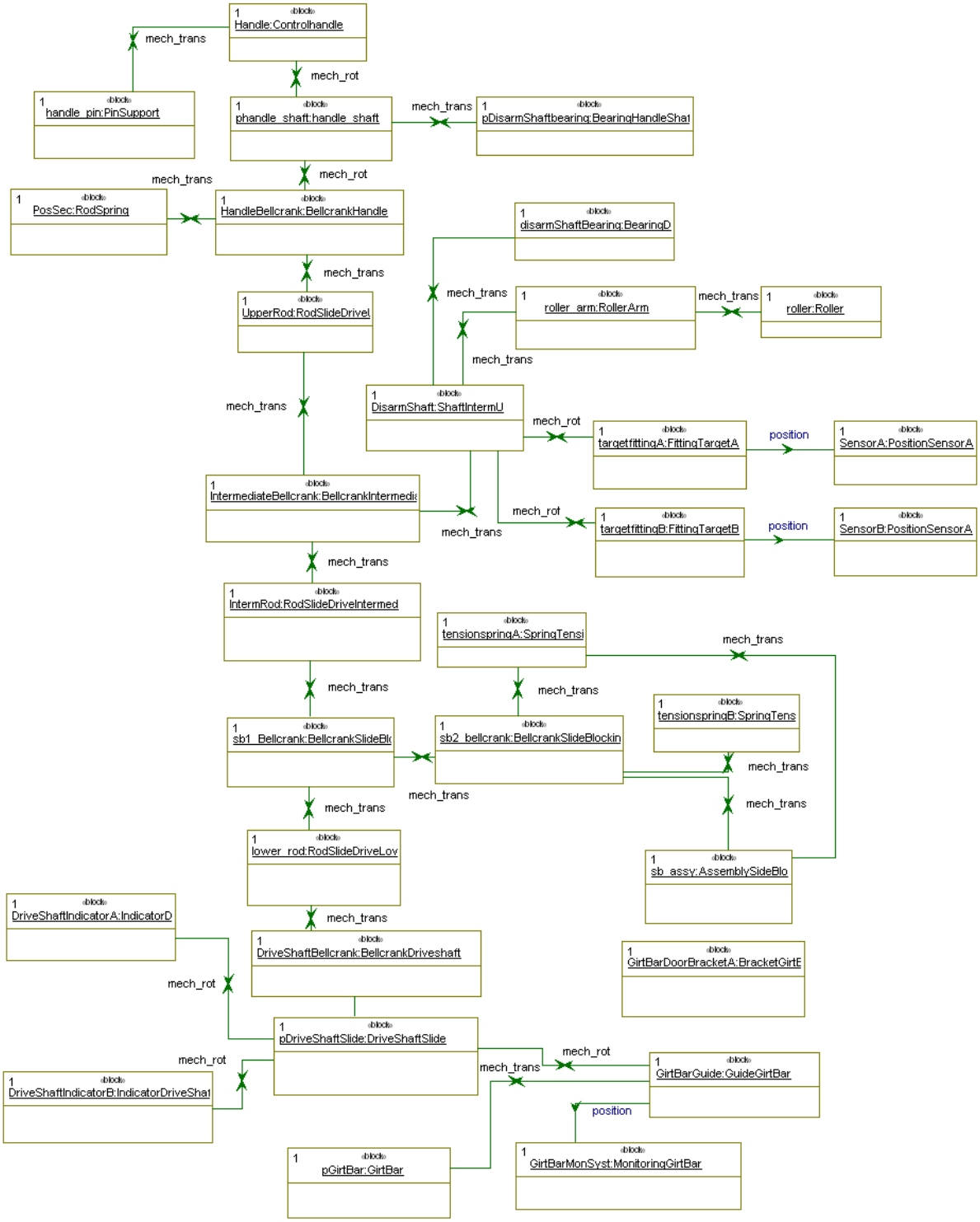


Figure 5.15 : Principes constructifs du sous-ensemble d'armement toboggan

5.2.6. Modélisation pour les vues validation et comparaison d'architectures

La modélisation des analyses et de la validation par le profil s'appuie sur la définition de stéréotypes spécifiques, ainsi que sur la définition en SysML des vues spécifiées au chapitre III. Les vues et les stéréotypes du profil sont utilisés dans les patrons d'analyse système spécifiés également au chapitre trois, et dont la mise en œuvre est décrite dans le sous-chapitre 4.4.

Le profil SysML définit des vues spécifiques. Celles-ci constituent une extension du stéréotype *view* à l'élément *package* présent dans la spécification de SysML. Les vues de validation et de comparaison d'architectures sont constituées des:

- entités du modèle prescriptif constitué de blocs d'exigences enrichis par les données supplémentaires spécifiques à la méthodologie,

- éléments objets de l'analyse (concept de solution, ensemble de solutions logiques / physiques),

- éléments transverses permettant l'analyse et la validation.

A l'intérieur de ces vues les stéréotypes permettent d'exprimer les différents éléments explicitant les problèmes rencontrés et les éléments de raisonnement utilisés lors de la spécification des exigences système puis lors de la définition d'architectures. De ce fait, ils constituent ou sont associés aux liens existant entre l'ensemble d'exigences et les architectures. Il s'agit d'une part, d'associations de type *satisfy* qui explicitent le lien de couverture entre les exigences du modèle prescriptif et les architectures, et d'autre part d'associations de type *allocated* entre les différentes architectures. Enfin, ils permettent l'application des techniques d'ingénierie des modèles présentés au sous-chapitre 3.4, basée sur la détection et l'application automatique des patrons de la méthodologie.

Le profil présenté que nous avons défini, étend les métaclasses SysML pour exprimer les différents problèmes caractéristiques pris en compte dans la méthodologie. Ces problèmes concernent la cohérence, la justification, la traçabilité, la complétude de la validation et l'exactitude de la validation.

5.2.6.1. Stéréotypes de modélisation des éléments transverses

La définition du profil SysML inclut les éléments d'analyses et de validation précités en tant que stéréotypes des métaclasses *Problem* et *Rationale*, comme présentés sur les figures 5.16.

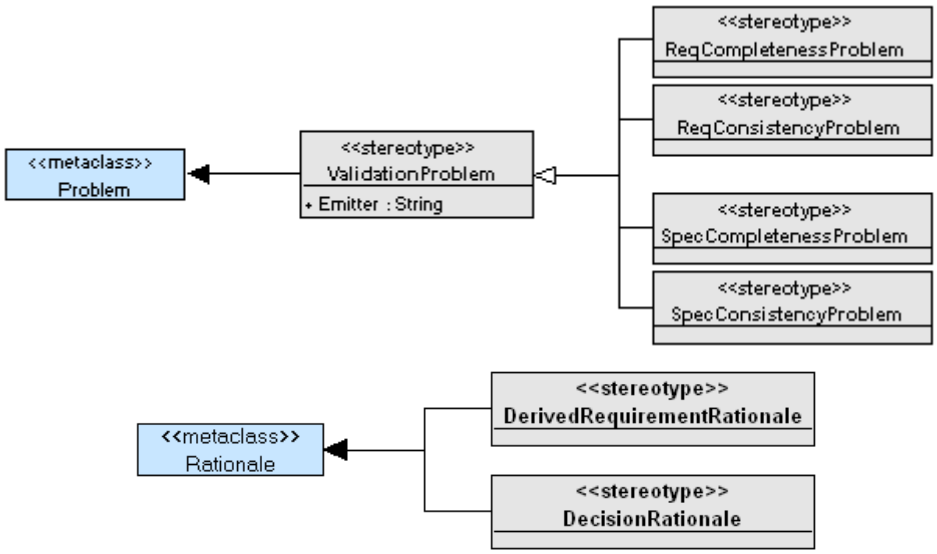


Figure 5.16 : Définition des éléments explicatifs problème et justificatifs

Un problème de complétude ou d'exactitude de la validation est crée et associé à une exigence par l'utilisateur ou lorsqu'un *pattern* de correction du modèle (tel que décrit au chapitre III) concernant la complétude ou l'exactitude du modèle est appliqué.

Des stéréotypes définis à partir du type *rationale* permettent de représenter explicitement des éléments de justification des décisions prises :

- Lors de la définition d'exigences dérivées des architectures logiques et physiques. Les éléments sont alors des *derived requirement rationale*.

- Lors de la définition d'un élément d'architecture. Celui-peut être relié à un autre élément de l'architecture ou d'une autre architecture par le lien de *decision rationale*.

D'autres éléments définis à partir du stéréotype *problem* ont également été définis dans l'objectif de couvrir les résultats d'application des patrons de modélisation présentés au chapitre III. Ces éléments sont inclus dans le profil et sont créés à l'aide des techniques présentées à la section 4.4.1.

L'ensemble de ces éléments définis dans le profil SysML permettent de mettre en place les mécanismes d'analyse et de validation système décrits dans la méthodologie.

5.3. Utilisation de VHDL-AMS

Le langage de modélisation VHDL-AMS a été présenté au chapitre II. Il est utilisé ici pour construire des modèles prédictifs exécutables de certains éléments d'architecture. Deux caractéristiques principales du langage du point de vue de son utilisation dans la méthodologie sont détaillées ci-dessous.

5.3.1. Mécanisme d'instanciation de composants

Les modèles crées dans notre approche tirent parti de la capacité de VHDL-AMS à instancier des composants système à partir de composants génériques. A partir d'un modèle générique, un composant peut être, d'une part instancié dans une architecture et, d'autre part connecté à la fois aux interfaces définies dans l'entité correspondante et aux autres ports et variables internes (quantité, signaux) de l'architecture. Un exemple d'application de ce mécanisme est présenté sur la figure 5.17.

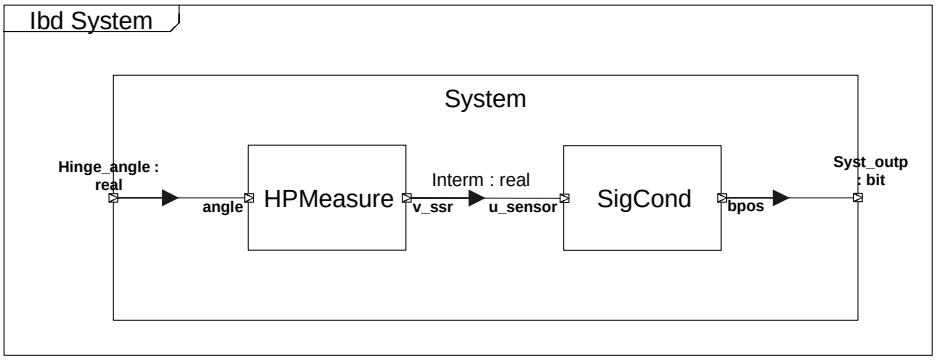
```
ENTITY comp_a IS
 PORT (QUANTITY i : in real; o : out real);
END;

ENTITY systeme IS
 PORT (QUANTITY extin : IN real;
       QUANTITY extout : OUT real
       );
END;

LIBRARY disciplines;

ARCHITECTURE arch OF systeme IS
   QUANTITY ploc : real;
BEGIN
   u1:ENTITY comp_a(beh) PORT MAP (extin,ploc);
   u2:ENTITY comp_a(beh) PORT MAP (ploc,extout);
END;
```

Figure 5.17 : Exemple d'instanciation de composant

Dans cet exemple, l'architecture `arch` de l'entité `systeme` est constituée de deux composants U1 et U2 instanciés à partir de l'entité `comp_a`. Les composants utilisent une architecture `beh`, non représentée ici.

Sur les ports (formels) i et o des instances de l'entité `comp_a,` sont connectés les ports (réels) `ext_in` et `ext_out` définis dans la déclaration d'entité, ainsi que la quantité `ploc` déclarée dans l'architecture. Cette connexion des ports formels sur les ports réels lors de l'instanciation est réalisée par l'instruction PORT_MAP.

Ce mécanisme est utilisé dans la méthodologie pour traduire en VHDL-AMS les structures modélisées en SysML. Les blocs sont réalisés comme des entités. Leurs instances, crées comme des objets appartenant à un bloc, sont définies telles des composants dans l'entité englobante. Un exemple de cette équivalence est présenté sur la figure 5.18. Ce mécanisme permet donc de mettre en œuvre les mécanismes d'instanciation SysML sur VHDL-AMS, bien que ce dernier langage ne soit pas orienté objets. Notons que SysML en soi n'est pas non plus un langage orienté objets. Les concepts à objets que nous utilisons dans la méthodologie sont issus d'UML2.0, en particulier de la partie commune SysML-UML telle qu'elle est définie par le groupe SysML de l'OMG.




```
Entity System is
    Port (Hinge_angle : IN real; syst_outp :
OUT bit);
End entity;

Architecture struct_System  of System is
    Signal interm : real;
Begin

      sensor : Entity HPMeasure(HPM_arch)
      PORT MAP(angle => Hinge_angle, v_ssr =>
interm);

      signcond : Entity SigCond(SignCond_arch)
      PORT  MAP(usensor  =>  interm,  bpos  =>
syst_outp);

End struct_System;

Entity HPMeasure
    Port (angle : IN real; v_ssr : OUT real);
End Entity;

Entity SigCond
    Port (usensor : IN real; bpos : OUT bit);
End Entity;
```

Figure 5.18 : Représentation d'un système constitué d'un capteur de position angulaire et d'un élément de conditionnement du signal capteur.

5.3.2. Spécificités du profil SysML pour la modélisation des systèmes conservatifs

Pour cette application, la modélisation conservative désigne les systèmes dont l'énergie totale est conservée au cours du temps.

En VHDL-AMS, de tels modèles sont généralement construit sur la base d'objets de type *terminal*. Un objet terminal définit un nœud auquel sont associées une variable de flux et une variable d'effort. Les terminaux sont classés par nature, suivant le type d'énergie considéré. Les natures électrique, thermique, mécanique en rotation, mécanique en translation, et hydraulique sont définies par défaut dans une librairie.

Pour chaque terminal déclaré dans le modèle, le moteur de simulation du VHDL-AMS résout, sur l'intervalle de simulation, le système d'équations constitué par les lois de Kirchhoff généralisées. Ces lois imposent que la somme des variables de flux soit nulle sur un nœud (appelé terminal), et que les variables d'effort sur ce terminal soient égales. Les terminaux sont des éléments de modélisation déclarative puisqu'ils n'imposent pas une valeur ou un sens de circulation du flux. Les modèles intégrant des ports terminaux sont donc des modèles déclaratifs aisément réutilisables dans des contextes différents.

La modélisation conservative a été rendue possible dans la modélisation d'architecture physique SysML par la définition du profil. Le profil permet la définition :

- de ports spécifiques reprenant les terminaux et les natures de VHDL-AMS. Ces ports sont les supports à la définition des ports dans les déclarations d'entités des unités de conception VHDL-AMS.
- de connecteurs permettant d'établir le lien entre deux terminaux. Ces connecteurs permettent de définir explicitement les connexions établies entre les ports des unités de conception et les quantités de branches.
- de flux définissant les variables échangées.

La figure 5.19 représente le modèle SysML d'un sous-ensemble fonctionnel étudié dans le cadre de l'exemple, porte passagers. Ce sous-ensemble a pour objectif de décrire le concept réalisé comme moyen d'assistance à l'ouverture d'urgence d'une porte.

Le modèle est construit pour prendre en compte l'aspect conservatif et l'échange entre l'énergie potentielle du gaz sous pression stocké dans un accumulateur, l'énergie mécanique en translation produite par un système de vérin associé à un amortisseur hydraulique et la transformation en un couple de rotation.

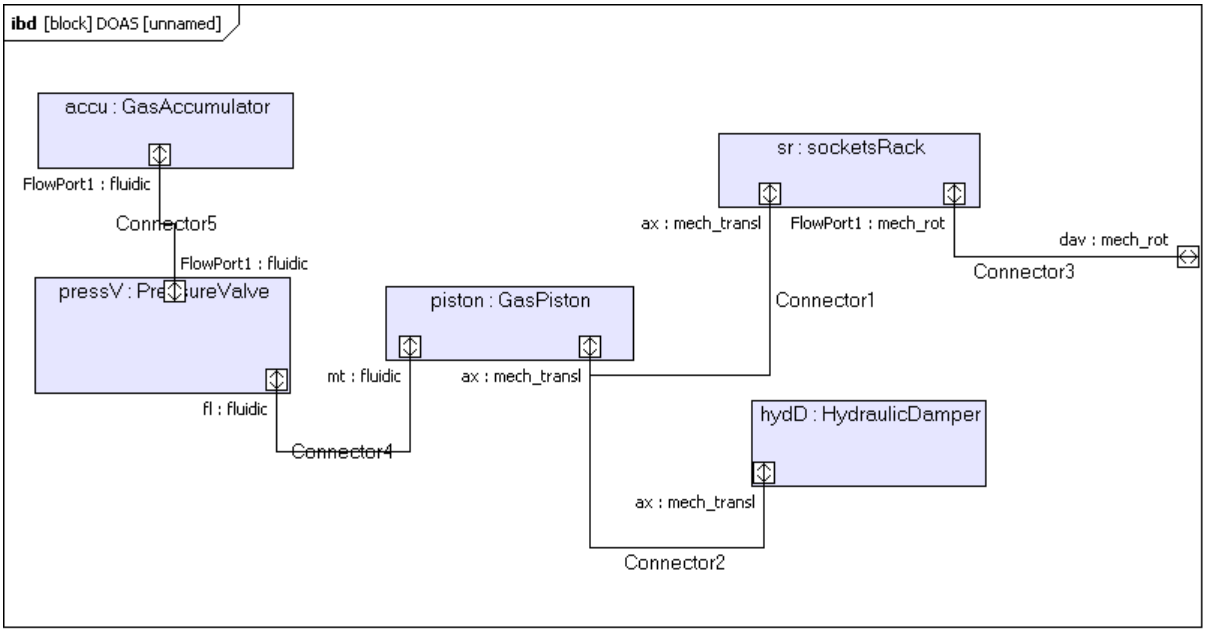


Figure 5.19 : Modélisation conservative du sous-ensemble fonctionnel d'éjection de porte

5.4. Mise en œuvre de la méthodologie dans un environnement Eclipse

Une mise en œuvre de la démarche a été réalisée sur la plate-forme de modélisation Eclipse et l'outil OpenArchitectureWare.

Eclipse est un environnement de développement logiciel apparu en 2001, *open-source* et multi-langages. L'architecture de cette application, basée sur un noyau et un ensemble de *plug-in*s permettant de réaliser toutes ses fonctionnalités, rend son extension très simple. En particulier, des *plug-ins* additionnels permettant la modélisation dans le langage SysML sont utilisés pour appliquer la méthodologie à notre cas d'étude.

Topcased (Toolkit in Open Source for Critical Applications and Systems Development) est un logiciel permettant le développement et la validation de modèle dans le domaine des systèmes et logiciels critiques. Topcased est intégré à la plate-forme Eclipse, et offre un moyen de modélisation efficace et conforme aux spécifications des langages supportés (UML, SysML, AADL).

OpenArchitectureWare est un outil dédié à la mise en œuvre d'un processus d'ingénierie des modèles selon un cadre MDA. Une introduction à l'utilisation de cet outil logiciel a été réalisée notamment par A. Haase [Haa_07]. Il est utilisable dans un contexte de développement basé aussi bien sur le formalisme EMF, qu'UML, XML, ou encore JavaBeans. OpenArchitectureWare est intégré à Eclipse-EMF.

Le métamodèle du profil SysML associé à la méthodologie, ainsi que son instanciation dans le cadre de l'exemple système porte passagers sont réalisés sur la plate-forme Eclipse.

Ces modèles peuvent ensuite être exploités par l'outil OAW afin de réaliser les opérations d'ingénierie des modèles.

L'application de la méthodologie avec *OpenArchitectureWare* s'appuie essentiellement sur :

- la définition des opérations de vérification de propriétés sur le modèle d'ingénierie système à l'aide d'un langage d'expression et d'évaluations de contraintes,
- la définition d'opérations de transformation de modèles,
- et la définition d'un script (*workflow*) définissant un enchaînement quelconque à partir des opérations ci-dessus.

Les opérations de transformation permises par *OpenArchitectureWare* concernent la transformation de modèle-à-modèle (M2M), de modèle-à-texte (M2T) et de texte-à-modèle (T2M). Elles sont réalisées au moyen d'instructions exprimées dans un langage dédié à la transformation de modèle nommé *XTend*, et du langage *XPand* permettant la génération de code à partir d'un modèle.

Des opérations de vérification peuvent être réalisées sous la forme de contraintes déclaratives. Pour cela, *OpenArchitectureWare* inclut le langage *Check*. *Check* est un langage spécifique à la vérification de contrainte, apparenté à OCL.

Ces trois langages s'appuient sur un système de types et un ensemble d'expressions communs, facilitant son apprentissage et son utilisation.

Le système de types permet de manipuler les objets des méta-modèles sous forme de collections, listes et ensembles et d'extraire leurs attributs et leurs opérations. Il définit des types génériques tels que *object, void, string, boolean*. De plus, le système de types intègre le système de types propre au métamodèle utilisé. Ainsi, le système de types associé au formalisme *ECORE* (meta-métamodèle) est intégré au système de types par défaut *OAW*.

Les instructions communes sont essentiellement apparentées aux langages Java et OCL. En particulier, des opérations communes telles que *.select(), .typeselect(), .reject(), .exists()* permettent, à la manière d'OCL, de manipuler des ensembles, des collections et des objets. Une description précise de ces langages se trouve dans les documents de référence du langage OAW [OAW].

Sur *OpenArchitectureWare*, les opérations de transformation ou de vérification sont considérées comme des composants de processus (*workflow component*). Un tel composant représente une partie élémentaire du processus MDA exécuté par le moteur OAW. D'autres composants OAW nécessaires au processus sont :

- des éléments de recherche en mémoire de modèles,
- des éléments de copie temporaire de modèles en cours de processus MDA,
- des éléments de stockage des modèles intermédiaires ou finaux.

Un *workflow* de génération est ainsi constitué d'un nombre quelconque de composants de *workflow* exécutés séquentiellement par le moteur OAW sur une machine virtuelle Java. Le *workflow* est écrit conformément à une syntaxe simple basée sur XML.

Le besoin en terme de transformation de modèles et de vérification de propriétés à pu être adapté au choix des langages explicités ci-dessus : notre profil SysML, VHDL-AMS, OAW-Check et OAW-XTend.

Ainsi, les opérations à réaliser dans le cadre de la mise en application de la méthodologie sont les suivantes :

- Vérification des propriétés méthodologiques sur le modèle instancié SysML-MICCSA, au moyen du langage Check.
- Vérification des propriétés système sur le modèle instancié SysML-MICCSA, au moyen du langage Check.
- Transformations liées à la réalisation des patrons de conception : l'application des patrons sera une transformation endogène au langage SysML-MICCSA à l'aide du langage XTend.
- Transformation entre modèles normatifs et modèles prédictifs : Il s'agit de la transformation depuis SysML vers VHDL-AMS également effectuée à l'aide du langage XTend.

Le paragraphe suivant présente le langage Check ainsi que son application pour la vérification des propriétés de la méthodologie et les propriétés système.

5.4.1. Vérification des propriétés sur les modèles

Ce paragraphe décrit une implémentation possible de la vérification de propriétés décrites dans la méthodologie MICCSA. Comme stipulé dans le chapitre III, la vérification de propriétés est réalisée à deux niveaux, eux-mêmes décrits dans les paragraphes 4.3.1 et 4.3.2.

Les paragraphes suivants présentent des propriétés OAW-Check écrites relativement à l'instanciation de modèles MICCSA en SysML dans le cadre de l'exemple présenté en début de ce chapitre IV.

5.4.1.1. Caractéristiques du langage OAW-Check

Le langage *Check* est un langage spécifique à un domaine, spécialisé pour la validation de modèles dans un contexte MDA. Il est basé sur le système de types et les expressions communes à *OpenArchitectureWare* présentées précédemment.

Dans le processus classique MDA appliqué au développement logiciel, *Check* permet de valider l'état d'un modèle plusieurs fois dans le processus, après chaque transformation par exemple, permettant ainsi de contrôler l'état du processus au cours de sa réalisation.

Le lange Check est composé d'éléments de base appelés contraintes. La syntaxe classique d'une contrainte est de la forme :

context TypeName [if guard-predicate] (ERROR|WARNING) msg-expression: predicate;

Si un élément du modèle correspond au type spécifié par le contexte, la contrainte est évaluée. Si la contrainte n'est pas satisfaite, l'expression *msg-expression* est évaluée et la

chaîne de caractères, éventuellement associée, est stockée comme un problème, accompagnée de l'indication du niveau de sévérité spécifié :

- Une contrainte de type *WARNING* non satisfaite implique l'affichage d'un message d'information sur la console permettant d'informer l'utilisateur que la contrainte n'est pas satisfaite. Le processus n'est cependant pas arrêté.
- Une contrainte de type *ERROR* non satisfaite implique l'arrêt d'exécution du processus MDA (*Workflow*) et l'affichage d'un message d'erreur.

A l'issue de la réalisation du processus MDA, la liste des problèmes rencontrés pendant l'exécution du processus et enregistrée puis affichée à l'écran.

Enfin, une garde facultative est écrite juste après le nom du type d'élément. Celle-ci permet de n'évaluer la contrainte que si la condition spécifiée est vérifiée.

5.4.1.2. Application pour le respect des jalons de la méthodologie

Les conditions associées aux jalons de développement du modèle ont été traduites en contraintes *Check* sur le bloc de construction du sous-ensemble d'armement toboggan. Celles-ci ont été appliquées sur :

- les ensembles d'exigences système et d'exigences dérivées définies,
- les architectures logiques et physiques.

Une propriété concernant la validité des exigences systèmes est présentée sous la forme de règles *Check* listées ci-dessous. Nous pouvons noter la :

- Présence d'une méthode de validation définie pour chaque exigence.
- Présence d'un élément de justification pour chaque exigence identifiée comme orpheline

```
context Requirement WARNING "No upward tracability link / Rationale : " +
this.name:
      this.annotatedBy == null && this.sourceOf == null;
context Requirement WARNING "No validation mean specified : " + this.name:
      this.validation_mean == null;
context Requirement WARNING "No allocation in logical architecture : " +
this.name:
      this.prescr == null ||
this.prescr.notExists(e|e.metaType(LogicalArchitectureConstituant))
context Requirement WARNING "No allocation in physical architecture : " +
this.name:
      this.prescr == null ||
this.prescr.notExists(e|e.metaType(PhysicalArchitectureConstituant))
```

Par ailleurs, les propriétés associées aux jalons définis dans le chapitre III ont été appliquées sur le sous-ensemble fonctionnel de l'armement toboggan. Des exemples de règles permettant d'évaluer les pré-conditions définies au chapitre III sont présentées ci-dessous.

5.4.1.3. Application pour la vérification de propriétés systèmes

La vérification des propriétés système correspondent à celles qui ont été formalisées à partir des exigences systèmes présentées au paragraphe 4.2.3. Ces propriétés, constitutives des modèles prescriptifs, sont évaluées dans les modèles constructifs des architectures logiques et physiques.

Représentation du modèle prescriptif en langage SysML/OCL.

Des évaluations conformes au processus analyse système ont été réalisées pour évaluer les solutions logiques et physiques envisagées.

Le patron *prescription non évaluable* a aussi été appliqué sur l'exigence ci-dessous, affectée au sous-ensemble fonctionnel correspondant au mouvement initial de porte passagers.

d) *The handle shall be a hinged device measuring at least 5 inches (12.7 cm) wide.* [From ARP488]

Une règle de complétude s'assure que l'attribut *width* est bien associé au bloc *Handle*. La propriété ci-dessous a été écrite en *Check* et évaluée sur le modèle d'architecture physique :

```
// Vérifie la présence d'un attribut nommé width dans le bloc concernée :
Context PhysicalBlock WARNING " prescription non évaluable" :
this.ownedAttribute.exists(e|e.name.toLowerCase() == "width");
```

Sur cette même exigence système, le patron *prescription non satisfaite* peut être appliqué. Pour cela, une première propriété assure qu'une valeur a été associée à l'attribut. La propriété suivante vérifie la valeur associée à l'attribut *width* dont l'existence a été évaluée :

```
// Vérifie la présence qu'une valeur a été attribuée à un attribut:
context Block WARNING "Egalité Propriété" :
      this.ownedAttribute.select(e|e.name.toLowerCase()=="width").defaultVa
lue.exists(e|e);

// Vérifie que la valeur de l'attribut est dans une plage définie :
context PhysicalBlock WARNING "Intervalle Propriété" +
(this.ownedAttribute.select(e|e.name == « width").defaultValue.get(0).
name.asInteger()).toString()  + "est inférieur à une valeur seuil"
(this.ownedAttribute.select(e|e.name.toLowerCase() ==
"width").defaultValue.name.asInteger()> 12);
```

5.4.2. Transformation des modèles SysML

Les patrons de modélisation définis dans la méthodologie MICCSA ont été mis en œuvre sur l'outil MDA comme la transformation du modèle SysML vers lui-même. Les patrons de modélisation sont réalisés sous la forme de règles XTend de transformation de modèle. Ces règles sont ensuite appelées lors de l'exécution du *workflow*.

5.4.2.1. Caractéristiques du langage OAW-XTend

Le langage XTend à été utilisé pour réaliser les étapes de transformation de modèle. Les transformations ont été appliquées sur le modèle d'ingénierie système SysML (transformation endogène) et également pour la traduction de modèle d'architecture SysML en modèle d'architecture VHDL-AMS simulable.

5.4.2.2. Transformation du modèle SysML orientée vers les patrons de modélisation

Les règles XTend ont été écrites pour enrichir le modèle d'ingénierie système :

- Par l'application des patrons de la méthodologie.
- Par génération de vues sur le modèle, telles que décrites au paragraphe 4.2.6.

Le patron de *dépendance directe* a, par exemple, été appliqué lorsque deux exigences dérivées contraignent les éléments cibles (*target_fitting*) du dispositif de capteur de positions. Ce patron de validation d'exigence a été utile pour identifier deux exigences contradictoires sur le positionnement des pièces cibles utilisées dans le principe construction modélisé. La règle XTend associée à ce patron est donnée par le listing ci-dessous.

```
toDirectDependencyPattern(SModel m) :
      let comment = new Comment :
      (m.elements.typeSelect(Trace).addAll(m.elements.typeSelect(Trace).sou
rce).size >
m.elements.typeSelect(Trace).addAll(m.elements.typeSelect(Trace).source).to
Set().size)?

      comment.setBody(m.elements.typeSelect(Trace).addAll(m.elements.typeSe
lect(Trace).source).name.toString())
      :
      comment.setBody("dépendance directe ok")->
comment;
```

Vérification de propriétés de validation sur le modèle prescriptif : Le patron *incomplétude d'exigence* a été appliqué sur les exigences système du sous-ensemble armement toboggan. Lors d'une modification de la solution logique du modèle, un élément d'architecture dont la présence était justifiée par la présence d'un autre élément a été maintenu par erreur. La Règle *XTend* pour l'application du patron *incomplétude d'exigence* est donnée ci-dessous.

```
toRequirementStructurePattern(Requirement r) :
      let comment = new Comment :
      (r.obj != null)&&((r.prescr != null)||(r.annotatedBy != null)) ?
            comment.setBody("Requirement structure validated")
      :
            comment.setBody("Requirement structure problem :" +
r.name.toString())->
comment;
```

Par ailleurs, le patron *donnée amont manquante ou invalidée* a été appliqué au niveau de l'architecture physique concernant l'élément bielle à ressort (tension *spring rod*) relié mécaniquement à l'élément d'architecture du levier de blocage du mécanisme d'armement (*slide blocking bellcrank*) (voir la règle ci-dessous). L'application de ce patron a permis

d'évaluer et de justifier la nécessité de cet élément pour le fonctionnement de l'ensemble du mécanisme.

```
toArchiRationale(Block b) :
      let comment = new Comment :
      comment.setBody("architecture  ok")->
      (b.annotatedBy == null)? // & b.sourceOf.eContents.isEmpty)
            comment.setBody(b.name+" not covered by rationale elements")
      :
            comment.setBody("architecture rationale ok")->
      comment.setAnnotates(b)->
      comment;
```

Des vues d'évaluation et d'analyse système ont été produites par transformation de modèles. La transformation a pour but de regrouper l'ensemble des éléments permettant d'évaluer deux alternatives de solution.

5.4.2.3. Transformation du profil SysML en VHDL-AMS

Le langage *XTend* a également été utilisé pour réaliser la traduction des modèles constructifs de solutions logiques et physiques vers les modèles prédictifs VHDL-AMS correspondants. Un modèle prédictif permet de conduire les activités d'analyse, d'évaluation et de validation. Il peut utiliser des modèles existants, déjà développés au sein de l'entreprise, ou disponibles auprès d'autres parties prenantes.

Pour réaliser la transformation d'une description d'architecture et de constituants SysML vers un programme exécutable VHDL-AMS, les étapes suivantes ont dû être effectuées :

- Réalisation du métamodèle conforme au langage SysML associé au profil présenté dans ce chapitre. Celui-ci est réalisé dans le formalisme ECORE.

- Réalisation d'un métamodèle ECORE conforme au langage VHDL-AMS et répondant aux exigences de la transformation.

- Définition d'un ensemble de règles XTend permettant la transformation d'un modèle SysML en un modèle conforme au métamodèle VHDL-AMS.

- Définition d'un ensemble de règles XPand permettant la génération du code conforme à la syntaxe concrète VHDL-AMS à partir du modèle VHDL-AMS.

La transformation du modèle SysML vers des modèles VHDL-AMS a donc nécessité le développement des métamodèle de ces langages, dans l'objectif de réaliser une transformation de modèles conforme à la Figure 5.20. Ceux-ci ont été réalisés dans le formalisme de méta-modélisation *ECORE*, associé à Eclipse et exploitable comme méta-métamodèle d'entrée et de sortie par XTend.

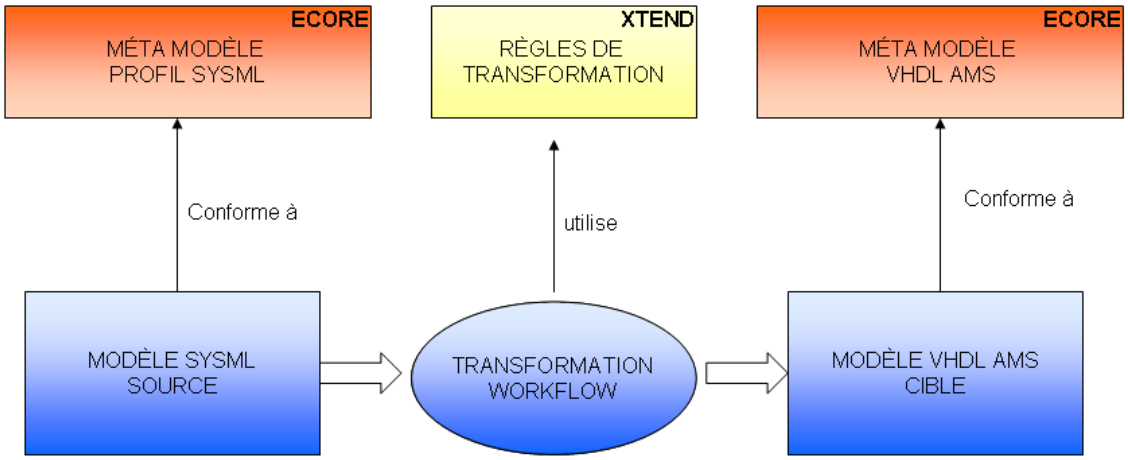


Figure 5.20 : Principe de la transformation de modèle SysML vers VHDL-AMS

La méta-modélisation du profil SysML définit dans le cadre de ces travaux est représentée sur la Figure 5.21. Les stéréotypes définis dans ce profil, représentés seulement en partie sur cette figure pour des raisons de lisibilité, ont été modélisés sous la forme de:

- relations d'héritage vers les métaclasses SysML,
- contraintes exprimées en Check.

Cette méthode a été utilisée afin de s'affranchir des limitations actuelles de modélisation d'un profil SysML sur la plate-forme Eclipse et sa prise en compte par l'outil OAW.

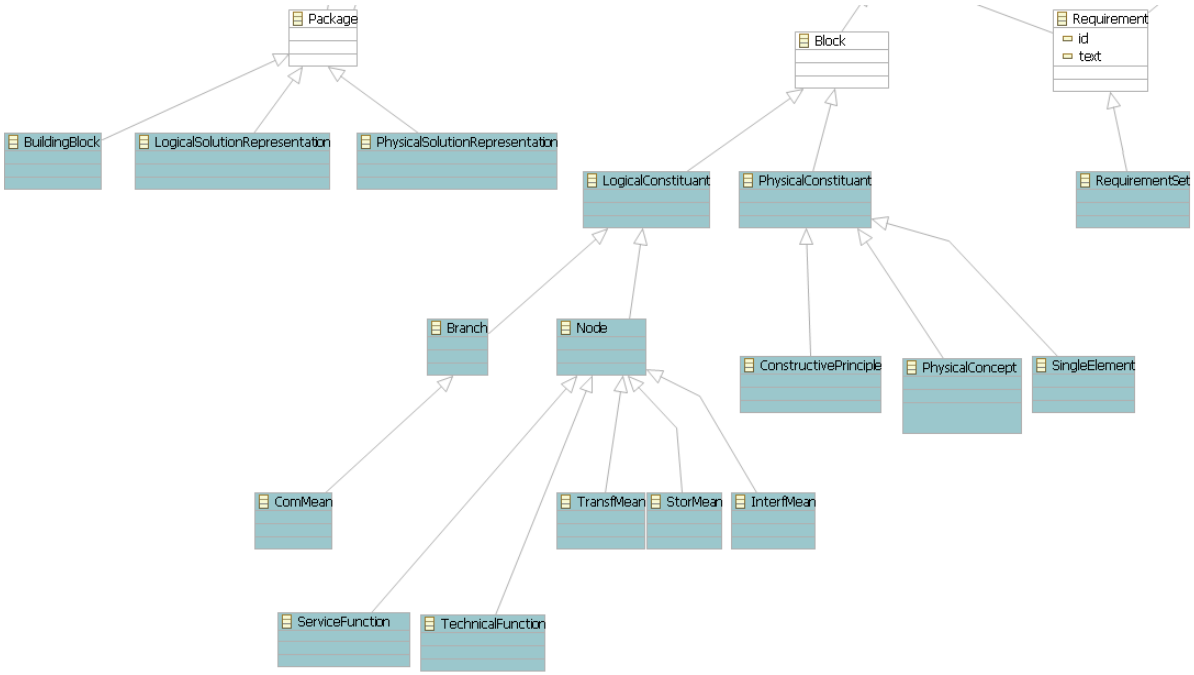


Figure 5.21 : Exemple d'éléments du profil SysML modélisés en ECORE.

Une vue générale du métamodèle réalisé pour SysML est présenté dans l'annexe H.

La méta-modélisation du langage VHDL-AMS, dont quelques éléments apparaissent en Figure 5.22, a été réalisée dans la lignée des travaux successifs de V. Albert [Alb_05] et de G.

Savaton [SDC_06] sur la réalisation d'un métamodèle du VHDL, ainsi que des travaux spécifiques au VHDL-AMS réalisés par D. Guihal [Gui_07].

L'extension réalisée dans le cadre de cette thèse est basée sur la grammaire BNF du langage VHDL-AMS, décrite par exemple dans [VHA_05]. Celle-ci est destinée à apporter plusieurs fonctionnalités du langage qui ne pouvaient être réalisées sur la base des métamodèles précités, dont :

- Le support du mécanisme d'instanciation de composants. Un modèle générique de composant peut ainsi être instancié, et ses ports connectés dans le contexte d'un système spécifique. Cette fonctionnalité, présentée au paragraphe 5.3.1, permet de se rapprocher du concept de *block* et de *part* supporté par SysML.

- La prise en compte des équations de comportement en tant que contraintes associées aux blocs. Ainsi, la structure du programme VHDL-AMS engendrée peut être complétée par des équations de comportement spécifiées dans le modèle SysML.

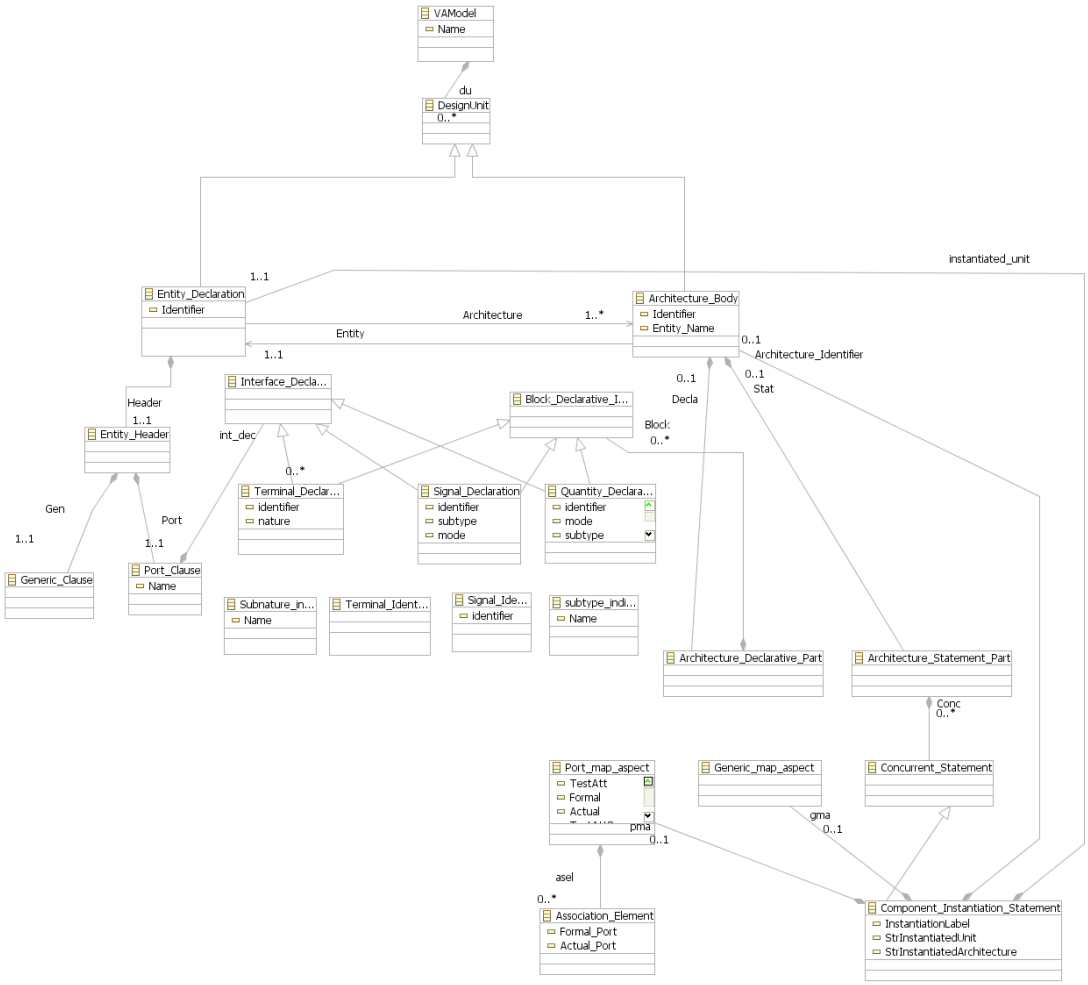


Figure 5.22 : Métamodèle des éléments de structure du langage VHDL-AMS

Suite à la définition de ce métamodèle sur la plate-forme Eclipse, les règles de transformation XTend ont été écrites. L'ensemble des règles de transformation se trouvent

dans l'annexe I. D'un point de vue qualitatif, l'approche suivie pour la transformation est la suivante :

- Génération d'une entité pour chaque bloc SysML. Pour chaque bloc constituant (logique ou physique) l'architecture à concevoir, une entité VHDL-AMS est instanciée. Celle-ci est complétée par les ports correspondant à ceux du modèles SysML.

- Génération d'un composant pour chaque *part* SysML. Dans l'architecture de l'entité du bloc contenant, le composant engendré est typé par l'entité correspondant au bloc.

- Connexion des ports formels du composant définis dans l'entité sur les ports réels de l'architecture qui le contient.

- Ajout des équations de comportement du composant à l'intérieur de son architecture.

La transformation crée ainsi un modèle VHDL-AMS, contenant les entités, les architectures et les composants instanciés et connectés sur un nombre de niveaux de décomposition quelconque.

Ce modèle permet ensuite de créer du code conforme à la syntaxe concrète du langage. Cette étape est réalisée par l'application de règles XPand sur le modèle VHDL-AMS. Cette transformation génère un fichier .vhd.

5.4.2.4. Génération de code VHDL-AMS

L'écriture de ces règles a, ensuite, été complétée par un ensemble de règles permettant de créer le code conforme à la syntaxe du langage VHDL-AMS. L'ensemble des règles XPand de génération de code se trouvent dans l'annexe J.

Le code engendré à partir de l'architecture, décrit dans l'annexe K, a permis de simuler puis de déterminer le profil de vitesse d'ouverture de la porte passager par le système d'assistance à l'ouverture, en considérant une inertie angulaire correspondante à la masse de la porte passagers.

5.5. Conclusion

Ce chapitre a permis de présenter une mise en œuvre possible des processus et des méthodes présentées au chapitre III. La démarche d'adaptation du langage SysML à la méthodologie MICCSA, au moyen d'un profil, a été détaillée ainsi que le contexte dans lequel le langage VHDL-AMS a été utilisé. Nous avons montré les diverses transformations de modèles nécessaires permettant d'aboutir à des modèles pour lesquels la validation des exigences est possible. L'application de la méthodologie au système, porte passagers a été utilisée pour illustrer les principaux aspects de cette modélisation.

L'usage de l'outil MDA OpenArchitectureWare a ensuite été présenté comme un moyen possible de mettre en application la partie ingénierie des modèles de la méthodologie. Les

langages de vérification de propriétés et de transformation de modèle ont été appliqués pour mettre en œuvre les règles et patrons de modélisation, abordés au niveau méthodes au chapitre III. Enfin, plusieurs de règles de vérification et de transformation, ont été illustrées avec le système porte passagers.

Chapitre 6. Conclusion générale

Les travaux de thèse présentés dans ce manuscrit se situent à l'intersection des domaines de l'ingénierie système et de l'ingénierie dirigée par les modèles.

Nous avons tout d'abord proposé un ensemble de **processus d'ingénierie système** répondant au besoin de la société LATECOERE sur des axes identifiés tels que l'ingénierie des exigences, l'analyse d'architectures système en phase de conception préliminaire, avec comme objectif transverse, l'intégration des disciplines spécifiques dans une phase commune de conception amont de l'architecture système. Lors de la définition de ces processus techniques d'ingénierie système, une attention particulière a été portée aux activités de validation des exigences et au processus d'analyse guidant la prise de décisions.

Ainsi, l'analyse du besoin industriel nous a naturellement conduits à définir une ***méthodologie***, que nous avons nommée ***MICCSA*** afin de supporter les processus, les méthodes, modèles et outils nécessaires pour atteindre l'objectif principal qui était l'évaluation dans les phases amont d'architectures en vue de l'optimisation. La contribution apportée concerne la définition de méthodes d'ingénierie système basées sur les modèles, sur leur **conformité à un standard d'I.S**. reconnu et au **processus MDA**. Ces méthodes et modèles, traditionnellement utilisés dans le domaine du développement logiciel, ont alors dus être adaptées/modifiées, souvent au niveau de concepts, afin de répondre au besoin industriel de l'entreprise.

Pour cette méthodologie MICCSA, orientée sur la conception des systèmes hétérogènes complexes, nous avons proposé un **modèle d'information** permettant une formalisation des données techniques échangées au cours de la réalisation de ses processus. Le modèle d'information développé permet d'apporter une sémantique répondant au besoin de modélisation des produits du groupe LATECOERE, pour lesquels l'architecture système est **principalement organisée autour d'organes mécaniques, et de composants systèmes**.

Les formalismes que nous avons introduits permettent d'assurer une **indépendance de la démarche vis-à-vis d'une mise en œuvre** particulière. La mise en place du modèle d'information associé permet de proposer l'utilisation de patrons de modélisation répondant aux besoins des processus d'ingénierie système et pouvant être exécutés par logiciel sur des modèles système exécutables.

Notre contribution concerne ensuite la définition d'un profil SysML adapté à la méthodologie et donc à la problématique et au besoin du groupe LATECOERE. Il est conforme à un standard d'I.S. (EIA 632) et permet l'interopérabilité avec un outil de simulation multidisciplinaire. Il fallait également doter la méthodologie de moyens de vérification et de validation, des exigences en particulier. Pour ce faire, nous avons introduit

au sein de la méthodologie la partie de SysML commune à UML2 qui est le langage de contraintes OCL. Etant donné que la vérification formelle est non exhaustive vis-à-vis de la vérification des propriétés système, nous avons également inclus au sein de la méthodologie le langage VHDL-AMS afin d'apporter des compléments de validation par simulation des modèles. Ainsi, nous avons pu définir une approche de vérification formelle de propriétés systèmes, et des caractéristiques permettant d'assister les phases de validation au niveau système, cela d'abord de manière indépendante du domaine, ensuite de façon adapté au domaine par le biais de stéréotype au niveau du profil SysML et de patrons au niveau méthodologique. Cette approche nous a contraints à étudier un certains nombre de transformations de modèles. Pour cela nous avons choisi de travailler au niveau des métamodèles.

Ainsi, la méthodologie rend possible **l'usage des patrons de modélisations** proposés, et guide le développeur à suivre les exigences d'un **standard** d'ingénierie système. Elle fournit également les moyens de **détecter des incohérences** des modèles au niveau de leurs sémantiques et **d'automatiser en parties** leur élaboration.

Enfin, l'ingénierie des modèles a aussi été mise en œuvre dans le cadre de la **transformation de modèles vers des modèles simulables VHDL-AMS**. Cette transformation ouvre ainsi la voie du prototypage virtuel de systèmes multidisciplinaires à partir de leur modélisation en SysML.

Les travaux présentés dans ce mémoire ont permis de couvrir une partie importante des objectifs définis par la problématique initiale et par l'entreprise. En particulier, les aspects démarche descendante (et partiellement montante), représentation et prototypage virtuel du système et vérification et validation des exigences nous semble répondre correctement aux attendus de l'entreprise, au moins au niveau étude. Cependant, d'autres objectifs n'ont été que partiellement, peu ou pas du tout atteints.

Sur le plan des méthodes, l'approche employée n'a pas pu conduire à l'application stricte des principes des l'approche MDA telle qu'envisagée initialement, centrée sur la production automatique de modèles spécifiques de solution. L'approche préconisée dans ce travail reste une approche d'élaboration et d'utilisation d'éléments de modèles formalisés selon les principes MDA, à l'aide d'outils informatiques plus ou moins sophistiqués. Mais dans cette approche l'ingénieur et les équipe de conception restent au cœur du système, et donc l'intervention manuelle reste encore importante.

L'objectif de prendre en compte les exigences des référentiels d'exigences clients tels que le GRESS n'a pu être atteint que partiellement, du fait du choix de limiter, dans la mise en œuvre et dans l'exemple proposé, le périmètre de la modélisation sur le produit final. Les exigences de processus présentes dans ces référentiel nécessiteraient un périmètre de modélisation plus large, incluant les produits contributeurs mis en place ou disponibles dans l'entreprise.

Le développement d'une plate-forme spécifique n'a également pas été possible, même si les choix réalisés au niveau des outils nous ont conduits à un usage cohérent d'une plate-forme unique, Eclipse, couplée à un outil MDA, pour réaliser la plus grande partie de la mise en œuvre. La mise en place d'une plateforme industriellement opérationnelle nécessite d'abord de rendre la méthodologie mature, et ensuite des ressources que peu d'entreprises peuvent financer. C'est la raison pour laquelle, notre choix s'est orienté vers des plateformes

existantes, même si celles-ci n'étaient pas toujours bien adaptées à nos besoins. Par ailleurs, le couplage avec des outils de CAE tels que Matlab-Simulink n'a pas put être établi. L'approche utilisée pour la transformation des modèles vers VHDL-AMS pourrait néanmoins être étendue à ce type d'application, moyennant des adaptations.

Des perspectives de travaux futurs ont été identifiées au cours de ce travail. Tout d'abord, l'approche employée a été volontairement fortement **orientée sur des descriptions et des modélisations du produit**. Ce choix est justifié par la volonté première de LATECOERE de faciliter les choix d'architecture produit, et donc le besoin d'associer l'information nécessaire pour guider de telles décisions à une représentation de l'architecture produit.

L'effort de formalisation et de conformité à un standard, l'EIA632, a permis de définir, au sein de la méthodologie des moyens de s'assurer, par la formalisation de propriétés, que les processus d'ingénierie système sont correctement réalisés. Cependant, une limite identifiée à cette approche est le manque de capacités d'adaptation à des variantes liées aux types de produits ou aux métiers impliqués.

Une autre voie a été prise notamment dans les travaux de S. Rochet [Roc_08], basés sur la formalisation des processus d'ingénierie eux-mêmes. En particulier, les processus formalisés du standard EIA632 peuvent être instanciés dans des processus d'entreprise puis des processus projet, et les liens entre chacune de ces instances peuvent, eux-mêmes, être formalisés. Cette approche permet clairement d'apporter une meilleure capacité d'adaptation lors de l'application des processus d'ingénierie système. Pour autant, cette approche ne permet pas de rattacher cette formalisation à des modèles produit, contrairement aux travaux présentés dans ce mémoire.

Le couplage de ces deux approches, duales, basées respectivement sur la formalisation des processus et des produits avec la capacité d'instanciation du général vers le spécifique, la capacité de vérifier formellement des propriétés système et la capacité de manipuler des patrons de modélisation, apparaît clairement comme une perspective de travail intéressante dans le domaine de l'ingénierie système guidée par les modèles.

TABLE DES FIGURES

Figure 2.1: Modèles des principaux cycles de vie, [Est 07] 27
Figure 2.2.: Scope et niveau de détail des principales normes de l'IS[AFIS] 29
Figure 2.3. : Evolution des principales normes de l'IS 30
Figure 2.5. : Le concept de système dans la norme EIA 632. 32
Figure 2.7. : Aspects internes du Building block ou bloc de construction de l'EIA 632 34
Figure 2.8. : Processus de la norme EIA 632 35
Figure 2.9. : Les processus de l'IEEE 1220 (selon l'AFIS) 37
Figure 2.10. : Les processus de l'ISO 15288 (selon l'AFIS) 38
Figure 3.1 Typologie fonctionnelle des modèles IS proposée par l'AFIS 42
Figure 3.2. Vision des interactions avec le diagramme pieuvre 44
Figure 3.3. Architecture fonctionnelle SADT 45
Figure 3.4. : Modèles représentés selon le formalisme EFFBD 46
Figure 3.5. : Les différents diagrammes du langage SysML 48
Figure 3.6.: Organisation des processus, méthodes et langages pour l'IS 52
Figure 3.7. : Sous-processus MDSD [Bak_97] 54
Figure 3.8. : Le cycle de vie mis en œuvre par la méthodologie Core 55
Figure 3.9. : Le processus globalement descendant d'harmony SE 57
Figure 3.10. : Définition de contrats opérationnels 58
Figure 3.11. : Eléments de base et activités de OOSEM [Lyk_01] 60
Figure 3.12. : Organisation des activités OOSEM 61
Figure 3.13. : Technique de transformation de modèle basée sur les métamodèles 65
Figure 3.14 : Interopérabilité des modèles [Car_07] 66
Figure 3.15. : Application de MDA en ingénierie MBSE [Clo08b] 68
Figure 3.16 : Concept généraux de l'IS vus par l'AFIS 69
Figure 3.17. : Vues AFIS des Exigences 70
Figure 3.18. Vue AFIS de l'architecture système 70
Figure 3.19. : Vue AFIS : Gestion de configuration 71
Figure 3.20. : Vue AFIS : Intégration, Validation, Vérification 72
Figure 4.1. : Applicabilité des exigences sources à un projet 75
Figure 4.2. : Recherche et intégration de principes de solutions 76
Figure 4.3. : Description générale de la méthodologie 77
Figure 4.4. : Vue générale des processus de la méthodologie MICCSA 78
Figure 4.5. : Eléments constitutifs du building block dans le modèle d'informations MICCSA 81
Figure 4.6. : Données d'ingénierie dans le modèle d'informations MICCSA 82
Figure 4.7 : Eléments d'exigences 83
Figure 4.8.: Eléments d'architectures logiques et physiques 86
Figure 4.9. : Eléments d'architecture opérationnelle. 89
Figure 4.10.:Les concepts de solutions héritent des éléments d'architecture système et d'architecture opérationnelle. 89
Figure 4.11. : Types de modèles d'analyse et de validation 91
Figure 4.12. : Eléments généraux de validation 92
Figure 4.13. : Utilisation des modèles dans MICCSA 94
Figure 4.14. : Exemples de propriétés 96
Figure 4.15. : Classes de propriétés à vérifier sur le modèle 97
Figure 4.16. : Typologie des propriétés générales des systèmes techniques 98

Figure 4.17. : Conditions d'application des exigences sur les constituants des modèles de solution 102
Figure 4.18. : Processus de conception système MICCSA 107
Figure 4.19. : Transformation de modèles génériques dans la méthodologie MICCSA 109
Figure 4.20. : Transformations MDA appliquées aux architectures opérationnelles, logiques et physiques, et Processus de définition des solutions 113
Figure 5.1 : Démarche de spécialisation de SysML retenue. 121
Figure 5.2 : Définition du stéréotype bloc de construction 122
Figure 5.3 : Partie modèles constituants du projet 123
Figure 5.4 : Bloc de construction de niveau 2 123
Figure 5.5 : Relations d'ingénierie d'exigence dans le profil SysML-MICCSA 124
Figure 5.6 : Principaux cas d'utilisation en phase d'exploitation 128
Figure 5.7 : Scénario d'utilisation normale de l'armement toboggan 128
Figure 5.8 : Niveaux d'abstraction des blocs logiques et physiques du profil SysML 129
Figure 5.9 : Architecture logique système porte passagers 130
Figure 5.10 : Caractéristiques des constituants d'architecture logique 130
Figure 5.11 : Architecture logique sous-ensemble fonctionnel armement toboggan 131
Figure 5.12 : Caractéristiques des constituants d'architecture physique 132
Figure 5.13 : Exemple de concepts de solution 132
Figure 5.14 : Topologie du sous-ensemble fonctionnel armement toboggan 133
Figure 5.15 : Principes constructifs du sous-ensemble d'armement toboggan 134
Figure 5.16 : Définition des éléments explicatifs problème et justificatifs 136
Figure 5.17 : Exemple d'instanciation de composant 137
Figure 5.18 : Représentation d'un système constitué d'un capteur de position angulaire et d'un élément de conditionnement du signal capteur. 138
Figure 5.19 : Modélisation conservative du sous-ensemble fonctionnel d'éjection de porte 140
Figure 5.20 : Principe de la transformation de modèle SysML vers VHDL-AMS 147
Figure 5.21 : Exemple d'éléments du profil SysML modélisés en ECORE. 147
Figure 5.22 : Métamodèle des éléments de structure du langage VHDL-AMS 148

Liste des tableaux

Tableau 3.1. : Récapitulatif des méthodes d'IS guidées par les modèles. 62
Tableau 4.1 : Synthèse phases de développement du modèle 111
Tableau 5.1 : Textes règlementaires applicables pour la certification 117
Tableau 5.2 : Eléments de structure du profil SysML-MICCSA 122
Tableau 5.3 : Eléments d'ingénierie d'exigences du profil SysML-MICCSA 125

Bibliographie

[AFIS] Association Française d'Ingénierie Système (AFIS), www.afis.fr.

[Alb_05] V. Albert, "Traduction d'un modèle de système hybride basé sur réseau de Petri en VHDL-AMS", Mémoire de stage Master, Université Paul Sabatier Toulouse III, Rapport LAAS N°07245, 6 Septembre 2005, 105p.

[APV_04] L. Apvrille, J-P. Courtiat, C. Lohr, P. de Saqui-Sannes, "TURTLE : A Real Time UML Profile Supported by a Formal Validation Toolkit", IEEE Transactions on Software Engineering, July 2004, 30(7), pp. 473-487.

[ARP_96] Society of. Automotive Engineers (SAE), "Certification Considerations For Hightly Integrated Or Complex Aircraft Systems", http://www.sae.org/technical/standards/ARP4754, November 1996

[Bah_05] Bahill, Dean, "What is System Engineering?". International Council on Systems Engineering, 2005. http://www.incose.org/practice/whatissystemseng.aspx .

[Bak_97] Baker, Loyd, Clemente, *et al.*, "Foundational Concepts for Model Driven System Design," white paper, INCOSE Model Driven System Design Interest Group, International Council on Systems Engineering, Jul. 15, 2000.

[Béz_05] Jean Bézivin, "On the Unification Power of Models". Software and Systems Modeling, 4(2):171-188, May 2005.

[Bro_99] W. J. Brown, John Wiley, Scott W. Thomas, "Anti-Patterns and Patterns in Software Configuration Management", *John Wiley & Sons*, April 23, *1999, 336 p.*

[Car_07] I. Cardei, M. Fonoage, R. Shankar, "Framework for Requirements-Driven System Design Automation", Systems Conference, 2007 1st Annual IEEE. 09/05/2007; DOI: 10.1109/SYSTEMS.2007.374671

[Car_08] T. Cardei, M. Fonoage, R. Shankar, "Model Based Requirements Specification and Validation for Component Architectures", SysCon 2008 - IEEE International Systems Conference Montreal, Canada, 2008.

[Car_98] Carson, R. S. "*Requirements completeness: A deterministic approach*", Proc. 8th Annual International Symposium of the INCOSE, 1998

[Ches_65] H. Chestnut, "Systems Engineering Tools", John Wiley & Sons, September 1965, 649 p. ISBN 978-0471154464

[Chil_94] S. Childers, J. Long, "A Concurrent Methodology For The System Engineering Design Process", Vitech Corporation Whitepaper, 1994.

[Clo_08a] Cloutier, Robert, "Model Driven Architecture for Systems Engineering" (Presentation Slides), Stevens Institute of Technology, presented at INCOSE International Workshop 2008, Albuquerque, NM, Jan. 25, 2008.

[Clo_08b] Cloutier, Robert, "Model Driven Architecture for Systems Engineering," Paper #220, Proceedings of the Conference on Systems Engineering Research (CSER), CSER 2008, University of Southern California, Los Angeles, CA, Apr 4-5, 2008.

[Cza_03] K. Czarnecki and S. Helsen, "Classification of Model Transformation Approaches," Proceedings of the 2nd Workshop on Generative Techniques in the Context of MDA, Anaheim, CA (2003), http://www.swen.uwaterloo.ca/~kczarnec/ECE750T7/czarnecki_helsen.

[DO_00] Radio Technical Commission for Aeronautics (RTCA), "RTCA/DO 254 : Design Assurance Guidance For Airborne Electronic Hardware", 2000.

[DO_92] Radio Technical Commission for Aeronautics (RTCA) / European Organization for Civil Aviation Equipment (EUROCAE), "DO-178B : Software Considerations in Airborne Systems and Equipment Certification", 1992.

[Dou_99] B. P. Douglas, "Doing Hard Time, Developping Real-Time Systems with UML, Objects, Frameworks, and Patterns", Addison-Wesley, ISBN 0-201-49837-5, 1999.

[EIA_98] EIA/ANSI 632, "Processes for Engineering, a System Electronic Industries Alliance (EIA) et American National Standards Institute (ANSI)", 1998.

[Est_08] Jeff A. Estefan. Survey of model-based systems engineering (MBSE) methodologies, Technical report, revB, INCOSE MBSE Focus Group, 23 May 2008, p.47.

[Fon_08] B. Fontan. « Méthodologie de conception de systèmes temps reel et distributes en contexte UML/SysML ». Rapport LAAS N°08047. Doctorat, Université Paul Sabatier, Toulouse, 17 Janvier 2008, 132p.

[FAA_06] Federal Aviation Administration (FAA), National Aispace System (NAS) – "System Engineering Manual". Version 3.1, 2006. www.faa.gov/about/office_org/headquarters_offices/ato/service_units/operations/sysengsaf/seman/

[Goo_57] Goode, R. Machol, "System Engineering : An introduction to the design of large scale system", 1957. Wiley-Interscience; March 27, 2000.

[GRESS] Airbus Directive, General Requirements for Equipement and System Supplier.

[Gui_07] David Guihal. « Modélisation en langage VHDL-AMS des systèmes pluridisciplinaires. » Rapport LAAS N°07250. Doctorat, Université Paul Sabatier, Toulouse, 25 Mai 2007, 190p.

[Haa_07] Arno Haase, Markus Völter, Sven Efftinge, Bernd Kolb, "Introduction to openArchitectureWare 4.1.2", MDD-TIF07 (Model-Driven Development Tool Implementers Forum), 2007

[Hal_62] A. Hall, "A methodology for Systems Engineering", van Nostrand, 1962, 478p.

[Ham_05] J-C Hamon, « Méthodes et outils de la conception amont pour les systèmes et les microsystèmes » Rapport LAAS No05064 Doctorat, Institut National Polytechnique, Toulouse, 1er Février 2005, 178p. 2005.

[Har_84] D. Harel , "Statecharts: a visual approach to complex systems". Science of Computer Programming 8 3 (1987), pp. 231–274.

[Har_07] G. Harmel. « Vers une conception conjointe des architectures du produit et de l'organisation du projet dans le cadre de l'Ingénierie Système ». Doctorat. Université de Franche-Comté, 2007. tel-00206133, version 1.

[Hue_79] D. Huebner, "Systems Validation through Automated Requirements Verification," In Proceedings of COMPSAC'79, IEEE Computer Society Press, Washington, DC, pp. 222-227.

[IEE_00] ANSI/IEEE Std 1471-2000, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", American National Standards Institute/Institute for Electrical and Electronic Engineers, Sep. 21, 2000.

[IEE_89] Institute for Electrical and Electronic Engineers (IEEE) Std 610.3-1989 "IEEE standard glossary of modeling and simulation terminology", May 1989.

[IEE_98] Institute for Electrical and Electronic Engineers (IEEE), IEEE 1220, "Standard for application and Management of the Systems Engineering Process", 1998.

[IEE_99] Institute for Electrical and Electronic Engineers (IEEE), « Standard for application and management of the system engineering process », 1999.

[Inc_00] American Institute for Aeronautics and Astronautics (AIAA), Society of Automotive Engineers (SAE), IEEE, Federal Aviation Authority (FAA), INCOSE, "Framework for the Application of Systems Engineering in the Commercial Aircraft Domain". DRAFT, Version 1.2a, July 28, 2000

[INC_04] INCOSE : "INCOSE System Engineering Handbook", 1994.

[INCTP_04] INCOSE-TP-2004-004-02, Version 2.03, September 2007.

[ISO_O8] Organisation internationale de normalisation, International Electrotechnical Commission, ISO/IEC 15288, "System Engineering – System Life-Cycle Processes", association française de Normalisation (AFNOR), 2000.

[Jac_92] Ivar Jacobson : "Object-Oriented Software Engineering: A Use Case Driven Approach", Addison-Wesley Professional, July 10, 1992, 552p.

[Joh_07] Thomas Johnson, Jonathan Jobe, Christiaan Paredis, Roger Burkhart, "Modeling Continuous System Dynamics in SysML", Proceedings of the IMECE 2007, 2007.

[Joh_08] T.A. Johnson, C.J.J. Paredis, R.M. Burkhart "Integrating Models and Simulations of Continuous Dynamics into SysML.", Proc 6th Intl Modelica Conf., 2008.

[Keh_05] Christophe KEHREN : « Motifs formels d'architectures de systèmes pour la sûreté de fonctionnement », Thèse Ecole nationale supérieure de l'aéronautique et de l'espace (ENSAE), 20 décembre 2005, 160 p.

[Lon_95] J. Long., Relationships between common graphical representations in System Engineering. In 5th International Symposium of the INCOSE, 1995. Mise à jour juillet 2002.

[Lyk_01] Lykins, Friedenthal, Meilich, "Adapting UML for an Object Oriented Systems Engineering Method (OOSEM)", INCOSE meeting 15 January 2001.

[Mar_97] Brian W. Mar, "back to basics again. A scientific definition of systems engineering", University of Washington, Seminar Proceedings of annual INternational Council On System Engineering (INCOSE), 1997.

[Mau_05] R. Maurice : « Contribution à la méthodologie de conception système : application à la réalisation d'un microsystème multicapteurs communicant pour le génie civil». Rapport LAAS N°05646 Doctorat, Institut National Polytechnique, Toulouse, 15 Décembre 2005, 151p.

[MDA_03] Object Modeling Group (OMG) Specification, "Model Driven Architecture Guide" version 1.0.1, Juin 2003.

[MIL_69] Military specification and standard "System Engineering Management", 1969.

[NAS_95] NASA Systems Engineering Handbook NASA Systems Engineering Handbook. Foreword to the September 1992 Draft .

[NAS_95] National Aeronautics and Space Administration (NASA), R. Shishko; R. Aster, R.C. Cassingham, "NASA Systems Engineering Handbook", Boston Library Consortium Member Libraries, juin 1995

[OAW] Open Architecture Ware, www.openarchitectureware.org

[Par_08] J. J. Paredis, T. Johnson, "Using OMG's sysml to support simulation" Proceedings of the 40th Conference on Winter Simulation, December 07-10,

[Pea_07a] RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim, "Simulation Based Design Using SysML - Part 1: A Parametrics Primer". INCOSE Intl. Symposium, San Diego, 2007

[Pea_07b] RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim, "Simulation-Based Design Using SysML - Part 2: Celebrating Diversity by Example". INCOSE Intl. Symposium, San Diego. 2007

[Perr_94] Perry, William, Memorandum from the Secretary of Defense to the Secretaries of the Military Departments, "Specifications & standards -- A new way of doing business", The Pentagon, Office of the Secretary of Defense. June, 29, 1994

[Pet_62] C.A. Petri. „Kommunikation mit Automaten". PhD thesis, Institut fur instrumentelle, 1962.

[Rin_05] Fernando Rincon, Francisco Moya, Jesus Barba, Juan Carlos Lopez, "Model Reuse through Hardware Design Patterns", Design, Automation and Test in Europe" 2005, p: 324 - 329.

[Rob_03] Robertas Damaševičius , Giedrius Majauskas , Vytautas Štuikys, "Application of design patterns for hardware design", Proceedings of the 40th conference on Design automation, June 02-06, 2003, Anaheim, CA, USA

[Ross_77] Douglas T. ROSS, "Structured Analysis (SA): A Language for Communicating Ideas". in: *IEEE Trans. Software Eng.* 3(1): pp. 16-34. 1977.

[SDC_06] G. Savaton, J. Delatour, K. Courtel, « Roll your own Hardware Description Language: An Experiment in Hardware Development using Model Driven Software Tools », Model-Driven Enterprise Information Systems 2006 Workshop, 2006.

[SDL_99] ITU-T Z.100, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, "Specification and Description Language" (SDL). Novembre 99.

[Sei_06] C. Seidner : Étude des représentations haut-niveau en Ingénierie Système et de leur aptitude à supporter des vérications formelles ». Séminaire bibliographique de D.E.A., École Centrale de Nantes, mai 2006.

[Roc_08] S. Rochet, « Formalisation des processus de l'Ingenireie Système: Proposition d'une méthode d'adaptation des processus génériques à différents contextes d'application », Doctorat de l'Université Paul Sabatier - Toulouse III, 26 novembre 2007.

[Sau_09] M. Sautreuil. « La Robustesse : une nouvelle approche pour l'intégration des systèmes de génération aéronautique », G2ELab, Université Joseph Fourier, Juillet 2009.

[SysRFP] A. Friedenthal, "UML for systems engineering RFP", OMG Document: ad/03-03-41, 2003.

[Tur_08] Skander Turki. «Ingénierie des modèles appliquée à la conception et la simulation de systèmes mécatroniques». PhD thesis, LISMMA, université de Toulon, 2008.

[Ver_08] J. Verries. Design Rationale in System Design, Proceeding of ICSEng, Las Vegas, Aug. 2008.

[Ver_08b] J.Verries, M. Paludetto, A-E-K. Saharaoui. From design with SysML to VHDL-AMS simulation, Proceeding of ESM'08, Le Havre, Oct. 2008, pp.115-120.

[Wall_04] Wallace GW, "Balancing Conflicting Stakeholder Requirements". Journal for Quality & Participation, Vol 18, No 2, pp 84-89.

[Was_06] Charles S. Wasson, "System Analysis, Design, and Development Concepts, Principles, and Practices", Wiley-Interscience, December 2005, 832 p.

[Whi_04] J. Whittle, I. Krueger, "A Methodology for Scenario-Based Requirements Capture." Proceedings of the ICSE 2004, Workshop on Scenarios and State Machines (SCESM), 23-28 May 2004.

ANNEXES

6.1. ANNEXE A - Axes d'amélioration des pratiques I.S.

Des entretiens ont été conduits avec des représentants des principaux métiers impliqués dans le développement : Programme, Bureau d'étude, Qualité, Production, Support client. Ces entretiens ont été menés à la fois sur le secteur des équipements et systèmes et sur l'aérostructure. Ils ont été organisés autour du questionnaire ci-dessous, orienté sur les pratiques d'ingénierie des exigences.

Questionnaire

1/ Dans votre activité quotidienne, quel est votre rôle et votre responsabilité vis à vis des exigences ?

2/ Selon vous qui doit formuler des exigences au début d'un projet ?
 Le client ?
 L'utilisateur ?
 Les autorités de certification ?
 Latécoère (programme, bureau d'étude, production, support, …) ?
Pourquoi ?

3/ Comment les exigences sont-elles tracées pendant le développement ?

4/ Les éléments de définition du produit sont-ils reliés aux exigences initiales ? Si oui, comment ?

5/ Au début d'un projet, plusieurs solutions sont-elles évaluée par rapport à l'ensemble des exigences ? Une méthode particulière est-elle utilisée pour choisir une solution de conception ?

6/ Selon vous, quelle est l'utilité des exigences ? Pensez-vous que certaines exigences sont inutiles ou sujettes à interprétation ? Comment éviter cela ?

7/ Les exigences sont-elles utilisées comme un référentiel partagé par tous les intervenant du projet ? Quels moyens sont ou pourraient être utilisés pour cela ?

8/ Selon vous l'impact de chaque exigence est-il suffisamment analysé (impact sur le produit en terme de fonctions, performances, de maintenabilité… et sur les contraintes de coût / délais). Comment l'impact de l'évolution d'une exigence est-il ou pourrait être pris en compte ?

9/ Considérez-vous que les exigences initiales sont suffisamment tracées tout au long du projet ? Pensez-vous que les moyens utilisés pour réaliser cette traçabilité sont satisfaisant ?

10/ Pensez-vous que la vérification d'une exigence (critère mesurable et procédure de vérification) doit être prévue et planifiée dés sa formulation ?

11/ Pensez-vous qu'une démarche systématique et des moyens de gestion des exigences (moyen de traçabilité et de modélisation) pourraient faciliter les activités de conception et de vérification ?

Synthèse des réponses

Suite à ces entretiens, les activités suivantes ont pu être identifiées comme des axes d'amélioration prioritaire.

Activité : Analyser les exigences émises par le client, et analyser leur évolution.

- Valider les exigences avec le client. (Modélisation du contexte technique et opérationnel du système).
- Analyser la formulation des exigences émises par le client (périmètre, critère quantifiable).
- Analyser l'impact d'une exigence technique sur le design.
- Montrer au client l'impact d'une exigence (et d'une évolution d'exigence) sur le design.

Activité : Définir les exigences applicables à un niveau de conception donnée.

- Améliorer la communication des normes et exigences applicables aux concepteurs.
- Améliorer la prise en compte l'aspect des contraintes liées à la certification dans le plan de management.

Activité : Sélectionner et définir les solutions techniques.

- Eviter le développement trop linéaire et favoriser des représentations permettant de revenir à l'expression du besoin initial.
- Conserver une vue fonctionnelle ou abstraite sur chaque niveau de conception.
- Favoriser des solutions techniques adaptables aux contraintes de bas niveau (ex : orientation des butées, contraintes géométriques). (AS)

Activité : Justifier les exigences de bas niveau.

- Conserver entre les projets la logique et le raisonnement qui mènent aux choix techniques.
- Améliorer le lien entre définition de la solution et documents de justification.

Activité : Spécifier les exigences à partir des solutions de conception.

- Conserver la vue globale du produit.
- Eviter la dérive des exigences déclinées par rapport aux exigences initiales (risque d'écart / non tenue des exigences initiales).
- Systématiser les méthodes de validation des exigences (cohérence, complétude).
- Valider la conformité des solutions par rapport aux exigences en évitant les essais.

Activité : Prendre en compte le retour d'expérience dans la déclinaison des exigences.

- Disposer d'une base de connaissance partagée sur le produit.
- Mieux analyser les possibilités de réutilisation de sous-ensembles, d'équipement.

Activité : Communiquer les exigences aux personnes chargées de définir les solutions de conception.

- Améliorer la communication des exigences (formalismes utilisés)
- Systématiser la prise en compte des exigences par les personnes.
- Améliorer la communication entre les métiers impliqués dans les choix techniques préliminaires (BE AS)

Activité : Assurer la traçabilité des exigences :

- Améliorer la réalisation de la traçabilité au niveau méthode et outils. (gestion de l'évolution d'exigences, analyse d'impact, utilisation pratique des outils).

6.2. ANNEXE B - catégories d'exigences systèmes

Chaque catégorie d'exigences systèmes est modélisée par un type de propriétés système, décrit ci-dessous.

Type de propriétés système	Description
Propriété fonctionnelle	Spécification d'une fonction en termes d'effet(s) attendu(s), contribuant à la fonction globale ou mission du système.
Propriété de performance	Spécification quantitative d'une performance attendue
Propriété opérationnelle	Spécification d'une fonctionnalité liée au contexte d'opération, non directement traduisible en Propriété fonctionnelle.

Propriété d'architecture	Spécification d'une contrainte sur l'organisation, la disposition, le couplage des organes logiques et/ou physique du système.
Propriété de sécurité qualitative	Spécification d'une contraintes sur l'organisation, la disposition, le couplage des organes logiques et/ou physique du système, afin d'éviter un risque ayant un impact sur la sécurité.
Propriété de sécurité quantitative	Spécification quantitative de probabilité d'occurrence d'un événement lié à un ou plusieurs organes de l'architecture logique et/ou physique, ayant un impact sur la sécurité.
Propriété de fiabilité qualitative	Spécification d'une contraintes sur l'organisation, la disposition, le couplage des organes logiques et/ou physique du système, afin d'éviter un risque ayant un impact sur la fiabilité.
Propriété de fiabilité quantitative	Spécification quantitative de probabilité d'occurrence d'un événement lié à un ou plusieurs organes de l'architecture logique et/ou physique, ayant un impact sur la fiabilité.
Propriété de maintenabilité	Spécification d'une fonction ou d'un moyen sur l'architecture physique, ou spécification quantitative lié à un ou plusieurs organes de l'architecture logique et/ou physique, d'une caractéristique ayant un impact sur la maintenabilité (MTBUR, MTBF).
Propriété d'interchangeabilité	Spécification de la capacité d'interchangeabilité d'un composant logique ou physique, et/ou contraintes associées aux interfaces associées.
Propriété d'installation et d'environnement	Spécification de la capacité d'interchangeabilité d'un composant logique ou physique, et/ou contraintes associées aux interfaces.
Propriété de poids	Spécification quantitative du poids du système ou de certains de ces sous-ensembles ou composants physiques.
Propriété de dispositions supplémentaires	Spécification d'un composant logique ou physique, ou d'interfaces en vue de l'évolution des fonctionnalités ou de l'environnement du système.
Propriétés comportementales	Spécification de caractéristiques sur le comportement du système, et /ou ses contraintes temporelles.

6.3. ANNEXE C - Objets et prescriptions selon les types de propriétés systèmes

Propriétés système	**Objet et descriptif de l'exigence**	**prescription sur le modèle**
spécifications applicable	Eléments sources d'exigences, niveaux de priorités associés.	Existence d'éléments sources d'exigences.
propriétés fonctionnelles	Constituants logiques et leurs interconnexions directes dans l'architecture logique. La formalisation des exigences fonctionnelles est utile pour veiller à la réalisation des fonctions par les	Lien bidirectionnel avec les constituants logiques de niveau N-1 (fonction définies au niveau avion) afin d'évaluer la contribution du système aux fonctions avions. L'existence de ports d'interfaces logiques caractéristiques sur les constituants logiques et la nature des

	éléments de l'architecture physique (moyens de réalisation).	éléments de communication entre les fonctions de l'architecture. Les caractéristiques fonctionnelles associées aux fonctions demandées.
propriétés de performances	Sous-ensembles d'architecture logique et physique.	Attributs de performances liés aux constituants logiques de l'architecture. Attributs de performances liés aux constituants physiques dont l'existence dans l'architecture est exigée.
propriétés opérationnelles	Les constituants de l'architecture opérationnelle. L'allocation des constituants de l'architecture opérationnelle sur les architectures logiques (fonctions opérationnelles) et physiques (moyens de réalisation associés aux fonctions opérationnelles). Les exigences opérationnelles sont les exigences qui décrivent une fonctionnalité qui ne peut être directement assignée à partir d'une exigence fonctionnelle de niveau avion (ou déclinée), mais qui provient de l'utilisation opérationnelle (et du contexte opérationnel) du système.	Les exigences opérationnelles prescrivent l'existence d'un élément de description opérationnel représenté dans un modèle cognitif validé. Cela inclut en particulier les exigences d'interface homme-machine pour l'exploitation et la maintenance de l'avion. Ces exigences prescrivent par ce type d'exigences est la nature des propriétés spécifiques (comportements) associées aux éléments d'interface logique et physique de l'architecture système.
propriétés de sécurité qualitative	Allocation de l'architecture logique sur l'architecture physique en particulier sur la topologie entre architecture logique et moyens de réalisation. Sur les moyens de réalisation dans l'architecture physique (interfaces, topologie interne des moyens de réalisation, types d'éléments associés, éléments partagés avec autres moyens de réalisations…). Modèles d'ingénierie spécialisée «safety» des architectures logiques et physiques.	Règles topologiques relatives aux principes de réduction des risques à respecter (évitement de cause commune, ségrégation fonctionnelle, dissimilarité).
propriétés de sécurité quantitative	Sous-ensembles de l'architecture logique. Modèles d'ingénierie spécialisée «safety» pour la définition de pannes acceptables pour le dispatch (MMEL), sous-ensemble de modes de défaillances dans les analyses d'ingénierie spécialisée.	Existence et valeur d'attributs de fiabilité ou de sécurité associée à la réalisation de constituants logiques. (Probabilité cible pour les modes de défaillances pertinents), un niveau de criticité associée, une fréquence de maintenance périodique réduite. Existence d'un moyen spécifique de démonstration de la valeur.
propriétés de fiabilité qualitative	Sous ensembles des architectures logiques et physiques. Modèles d'ingénierie spécialisée «safety» pour la définition de pannes acceptables pour le dispatch (MMEL),	Règles topologiques relatives aux principes de dissimilarité, de redondance. Existence de moyen intégré de test, de détection / confinement d'erreur.
propriétés de fiabilité	Sous-ensembles de l'architecture	Existence et valeur d'attributs de

quantitative	logique. Sous-ensembles de l'architecture physique. Modèles d'ingénierie spécialisée «safety» pour la définition de pannes acceptables pour le dispatch (MMEL),	probabilité cibles, MTBF/MCBF, pour les modes de défaillances pertinents.
propriétés de maintenabilité	Sous-ensemble de l'architecture opérationnelle, Sous-ensembles de l'architecture logique et physique, exigences de parties prenantes validées sur les objectifs généraux de maintenabilité du système.	attributs de maintenabilité du système, scenario opérationnels de maintenance, traçabilité / spécification du support et de la maintenance applicable au niveau avion. Sur l'architecture système, les objectifs de durée de vie.
propriétés d'interchangeabilité	Sous-ensembles de l'architecture physique pour assurer l'interopérabilité entre type d'avion et variantes d'avion.	caractéristiques d'interface des constituants,
propriétés d'installation et d'environnement	Projection entre architecture logique et architecture physique. Constituants de l'architecture physique. Les exigences doivent être compatibles avec les caractéristiques et la spécification avion.	Allocation architecture logique / physique, interfaces des constituants physiques en externe avec un élément de l'environnement direct du système. Attributs sur les caractéristiques de volumes, d'enveloppe, d'environnements, associés aux éléments de l'architecture physique. Attribut sur la nature, le type, le nombre, et les standards applicables aux éléments d'interface (électrique, hydraulique,..). Attribut sur les caractéristiques environnementales (pression ,température, vibrations, compatibilité CEM,).
propriétés d'interface fonctionnelle imposée sur le système	Constituants de l'architecture physique.	Attributs sur la nature et les caractéristiques des constituants d'interfaces physiques.
propriétés de contrainte physique.	- constituants d'interface physique avec les éléments en interface (ex : équipement de support au sol.) - moyens de réalisation : type de moyen de réalisation utilisés, - interface avec des moyens de réalisation.	Attributs de caractéristiques de consommation électrique ou autre sources de puissance, tolérance sur l'alimentation en source de puissance, fiabilité, consommation. Attribut sur les interfaces physiques avec les éléments en interface en particulier lorsque des équipements peuvent être fournit par différents fournisseurs. - caractéristiques d'interface avec les ressources partagées de l'avion telle que les bus de données. - caractéristiques d'interface applicables aux sorties du système et utilisés par d'autres système (nature,

		caractéristique, fiabilité, ..).
propriétés de poids	Attributs de sous-ensembles de l'architecture physique.	Attributs sur les constituants d'architecture physique
propriétés additionnelle	Les éléments ou sous-ensembles de l'architecture physique.	Existence d'un constituant d'architecture physique particulier (ex : utilisation par des concurrents, réutilisation). Existence dans l'architecture physique d'un équipement ou d'un système disponible sur étagère. Existence de plusieurs sources pour certains constituants de l'architecture physique. Caractéristiques de modifiabilité de constituant des l'architecture physique.
propriétés de dispositions supplémentaires	- sous-ensembles spécifiques de l'architecture logique et physiques. Ces exigences définissent des dispositions spécifiques à prévoir pendant le développement.	Existence de certains constituants considérés comme disposition supplémentaires dans l'architecture logique. Constituants de l'architecture physique prévue pour satisfaire les exigences de développement (moyen des tests, etc..).
propriétés comportementales	Eléments de description de comportement des architectures logiques et physiques -	- caractéristiques de fonctionnements liés à des contraintes temps réel, - comportement d'éléments ou du système sur certains évènements, événement de communication, de perte de transmission, - des contraintes sur des séquences d'initialisation, mise en route, arrêt, changement de phase de vol, - des caractéristiques de comportements en fonction de la température, de l'environnement.

.

6.4. ANNEXE D – Règles et Patrons de modélisation

Les règles de validation associées aux jalons de développement du modèle sont décrites de manière informelle dans le tableau ci-dessous :

MISRS_VAL01	Les méthodes de validation des exigences techniques système est conforme à la stratégie de validation du plan de validation.
MISRS_VAL02	La traçabilité descendante des exigences acquéreur et autre parties prenantes vers les exigences techniques système est complète.
MISRS_VAL03	La traçabilité ascendante des exigences système technique vers les exigences acquéreur et autre parties prenantes est complète.
MISRS_VAL04	Toutes les hypothèses liées aux exigences techniques systèmes sont dans le statut hypothèse valides et supportée par un élément d'analyse.

MISRS_VAL05	Les exigences qui ne possèdent pas de lien amont dans les référentiels acquéreurs et autres parties prenantes doivent posséder un élément de justification dont le statut est valide.
MISRS_VAL06	Les exigences qui ne possèdent pas de lien amont dans les référentiels acquéreurs et autres parties prenantes et non justifiées doivent amener à une revue de l'ensemble identifié des parties prenantes.
MISRS_VAL07	Les écarts de prescription existant entre une exigence amont et une exigence système technique tracés entre eux doivent être justifiés ou supportés par un élément d'analyse.
MISRS_VAL08	Evaluer l'existence d'un élément fonctionnel de l'architecture logique qui n'est pas prescrit dans le référentiel d'exigences système doit pouvoir être tracé vers une exigence dérivée technique ou un élément de justification.
MISRS_VAL09	Evaluer l'existence d'un élément d'architecture physique qui n'est pas prescrit dans le référentiel d'exigences spécifié.

6.5. ANNEXE E - Patrons de modélisation

Les patrons de modélisation peuvent être regroupés selon les processus de la méthodologie.

Définition des exigences	Formalisation des objets et prescriptions des exigences systèmes Génération d'exigences dérivées de l'intégration des concepts de solution
Définition des solutions	Composition de modèles constructifs de solution logique, physique.
Analyse et justification	Identification et résolution des conflits entre exigences, Génération des vue d'évaluation & comparaison des concepts de solutions envisagés, pour analyse de faisabilité, vérifiabilité et activités de validation système Conformité aux propriétés système. Compatibilité interface logique ou physique. Validation complétude architecture logiques et physique. Validation exactitudes architecture logique et physique.

Des exemples de patrons sont présentés ci-dessous :

EXEMPLES DE PATRONS POUR LA DEFINITION DES EXIGENCES

Nom du patron	Pré-condition(s) associée	Eléments généré dans le modèle
Formalisation objet d'exigences source	La catégorie de l'exigence a été définie.	Lien vers les éléments d'architecture
Formalisation des éléments prescriptifs	Le lien sur l'objet de l'exigence a été défini.	Expression de la propriété système évaluable sur l'objet.

Formalisation moyen de validation	Plan de validation formalisé.	Un moyen de validation est déterminé à partir du plan de validation.
Structure spécification système	Les ensembles d'exigences sources sont dans le statut validé.	Génère élément catégories d'exigences
Formalisation des éléments prescriptifs logique	l'objet de l'exigence a été défini sur l'architecture logique (préliminaire).	Expression de la propriété système évaluable sur un constituant logique.
Formalisation des éléments prescriptifs physique Identification des	l'objet de l'exigence a été défini sur l'architecture physique (préliminaire)	Expression de la propriété système évaluable sur un constituant physique.
Identification exigences/mesures d'efficacité		Liens à partir de l'architecture opérationnelle et des exigences sources

EXEMPLES DE PATRONS POUR LA DEFINITION DES SOLUTIONS

Nom du patron	**Pré-conditions & condition d'application**	**Eléments générés sur le modèle**
Adéquation concept de solution	Un concept de solution satisfait aux exigences systèmes qui lui sont assignées.	Génération d'exigence dérivée (MoP) à partir des MoE système.
Intégration concept de solution	Un concept satisfait aux exigences dérivées techniques qui lui sont assignées.	Les exigences dérivées correspondantes sont dans le statut satisfaites.
Connexion concept de solution	Les interfaces du concept de solution sont compatibles avec celles de l'architecture.	Exigence dérivées générées par la connexion des éléments d'interface du concept de solution.
Génération d'exigences d'interface	L'intégration d'un concept de solution induit des contraintes d'interface fonctionnelle ou physique dans l'architecture.	Génération d'élément d'interface. Génération d'exigence dérivée.
Génération d'exigences fonctionnelle	Un concept de solution implique de nouvelles fonctions (communication, stockage, alimentation, transformation) dans l'architecture logique et/ou physique.	Génération de constituants fonctionnels dans l'architecture fonctionnelle. Génération d'exigences dérivées.
Impact concept de solution	Les exigences dérivées générées par ce concept sont valides.	

EXEMPLES DE PATRONS POUR L'ANALYSE SYSTEME

Nom du patron	**Pré-conditions & condition d'application**	**Eléments générés sur le modèle**
Allocation	Un élément attendu de l'allocation entre architecture logique et physique est absent (valeur associée).	Génération élément absent.
Vérification propriétés système prescrites	La complétude et l'exactitude des prescriptions d'exigences systèmes sont évaluées sur les éléments objets.	Application des patrons prescription non évaluable et non satisfaite.

Prescription non évaluable (incomplétude modèle cible)	Un constituant d'architecture est désigné par un objet d'exigences mais ne permet pas l'évaluation de la prescription de l'exigence.	Génération élément problème sur l'architecture évaluée.
Prescription non satisfaite (inexactitude modèle cible)	L'architecture logique et/ou physique répond négativement à au moins une prescription d'exigence.	Génération élément problème sur les constituants évalués.
Impact exigences dérivées	L'impact d'une exigence dérivée sur l'architecture logique et physique n'est pas identifié.	Génération élément commentaire

EXEMPLES DE PATRONS POUR LA VALIDATION DES EXIGENCES

Nom du patron	**Pré-conditions & conditions d'application**	**Eléments généré sur le modèle**
Génération objectifs de validation	L'exigence est dans le statut basic. Les éléments d'objet et de prescription sont définis.	Génère objectifs de validation (en fonction du plan de validation et des exigences)
Tâches de validation	Les règles de validation sont définies. Les objectifs de validations sont définis.	Génère les tâches de validation en fonction du plan de validation (stratégie de validation) et des exigences.
Incomplétude interne	Un élément essentiel de l'exigence est absent (objet, prescription, formulation, justification).	Modifie statut de validation de l'exigence. Génère élément absent et un commentaire.
Donnée manquante ou invalidée	Une donnée en amont est invalidée ou manquante.	Modifie statut de validation de l'exigence, génère un commentaire.
Incohérence Objet / catégorie	L'objet n'est pas conforme à la catégorie identifiée de l'exigence.	Génère un élément problème associé à l'objet et la catégorie
Incohérence Prescription / catégorie	La prescription n'est pas conforme à la catégorie identifiée de l'exigence.	Génère un élément problème associé à la prescription et la catégorie
Remontée d'exigence	L'élément objet d'une exigence désigne un élément de l'architecture présent au niveau d'un bloc de construction amont.	L'élément bloc de construction amont et ses référentiels associés sont invalidés.
Incohérence ensemble	Les objets désignés concernent des éléments de blocs de construction amont et/ou aval.	Elément généré commentaire sur les exigences concernées.
Dépendance directe	Deux prescriptions sont directement liées.	Elément généré commentaire sur les exigences concernées.
Incomplétude spécification standard	Une catégorie de spécification standard applicable n'est pas couverte.	Elément généré commentaire sur la catégorie d'exigence.
Non couverture de validation	Un constituant de l'architecture n'a pas d'élément de validation associé.	Génération élément commentaire sur le constituant à vérifier.
Conflit moyen de validation	Les éléments de validation associé à un constituant de l'architecture logique ou physique n'est pas conforme au élément prescriptif de validation.	Génération élément problème sur le constituant à valider.

EXEMPLES DE PATRONS POUR LA VERIFICATION DES EXIGENCES

Nom du patron	**Pré-conditions & condition d'application**	**Eléments généré sur le modèle**

Non couverture de vérification	Un constituant de l'architecture n'a pas d'élément de vérification associé	Génère élément commentaire sur le constituant non vérifié.
Conflit moyen de vérification	Les éléments de vérification associé à un constituant de l'architecture logique ou physique n'est pas conforme aux éléments prescriptifs de vérification.	Génère élément problème sur le constituant d'architecture à vérifier.

EXEMPLES DE PATRONS D'ANALYSE D'ARCHITECTURE LOGIQUE

Nom du patron	Pré-conditions & condition d'application	Eléments générés sur le modèle
Couverture logique descendante	Toutes les exigences fonctionnelles sont reliées à au moins une fonction dans la solution logique.	Génère élément problème sur les exigences non couvertes.
Couverture logique ascendante	Toute fonction de l'architecture logique est reliée à au moins une exigence fonctionnelle dans l'ensemble prescriptif système.	Génère élément problème sur les constituants non couverts.
Quantification mesures d'efficacité	Toutes les exigences – métriques d'efficacité ont été converties en métriques techniques sur l'architecture logique.	Génère élément commentaire sur les métriques non couvertes.

EXEMPLES DE PATRONS D'ANALYSE SYSTEMES TRANSVERSES

Nom du patron	Pré-conditions & condition d'application	Eléments générés sur le modèle
Couverture propriété physique	Toutes les propriétés produit prescriptive assignées à la solution physique sont évaluables et satisfaite par l'architecture physique.	Génère élément problème sur les propriétés non couvertes.
Couverture topologie physique	Toutes les propriétés topologiques prescriptives sont vérifiées par l'architecture physique.	Génère élément problème sur les propriétés non couvertes.
Couverture propriété projection	Toutes les propriétés topologiques impliquant solution logique et solution physique sont vérifiées.	Génère élément problème sur les éléments prescriptifs non couverts.
Allocation logique physique	Tous les éléments du modèle physique sont reliés à au moins un bloc fonctionnel dans la solution logique.	Génère élément problème sur les constituants d'architecture logique / physique.
Allocation interfaces logiques physique	Toutes les propriétés prescriptives sur les **constituants logiques d'interface** sont vérifiées par le modèle de solution physique.	Génère élément problème sur les constituants logiques d'interface.

EXEMPLES DE PATRONS D'ANALYSE D'IMPACT

Nom du patron	Pré-conditions & condition d'application	Eléments générés sur le modèle
Impact évolution d'exigence source	Exigence source invalidée.	Génère vue sur les exigences systèmes et les constituants d'architectures impactés.
Impact d'évolution d'exigence système	Exigence système invalidée.	Génère vue sur les exigences source et les constituants d'architectures impactés.
Impact d'évolution d'architecture logique et/ou physique		Génère vue sur les exigences source, système, spécifiée et les constituants d'architectures impactés.

métrique robustesse aux évolutions d'exigences.	Les exigences sources, systèmes, et les architectures logiques sont définies.	Génère élément commentaire associé aux constituants d'architecture physique.

6.6. ANNEXE F - Contraintes associés aux stéréotypes du profil

La définition du profil SysML MICCSA est complétée par les contraintes décrites ci-dessous :

Package BuildingBlock :
[ ] Les éléments stéréotypés LogicalSolutionRepresentation, et PhysicalSolutionRepresentation doivent être définis.
[ ] Les éléments stéréotypés RequirementSet doivent être définis.
[ ] Le package ne peut être lié par une relation de contenance qu'à d'autres packages stéréotypés BuildingBlock

Package RequirementSet :
[ ] Un élément stéréotypé RequirementSet ne peut être associé par un lien de composition qu'à un élément de type exigence.

Package LogicalSolutionRepresentation :
[ ] L'ensemble des éléments contenus dans le package LogicalSolutionRepresentation doit être typé comme constituant logique.

Package PhysicalSolutionRepresentation :
[ ] L'ensemble des éléments contenus dans le package LogicalSolutionRepresentation doit être typé comme constituant physique.

Exigences :
[ ] Une exigence source ne peut être associée par un lien de traçabilité qu'à une exigence système.
[ ] Une exigence système ne peut être associé par un lien de traçabilité qu'à une exigence source ou un constituant d'architecture opérationnelle.
[ ] Un bloc d'exigence stéréotypé MoE ne peut être associé qu'à un élément d'architecture opérationnelle ou logique.
[ ] Un bloc d'exigence stéréotypé MoP ne peut être associé qu'à une fonction technique ou un constituant physique.
[ ] Un élément de justification de conception doit être associé à au moins un constituant d'architecture logique ou physique.
[ ] Un élément de justification d'exigences dérivée doit être associé à au moins un constituant d'architecture logique ou physique.

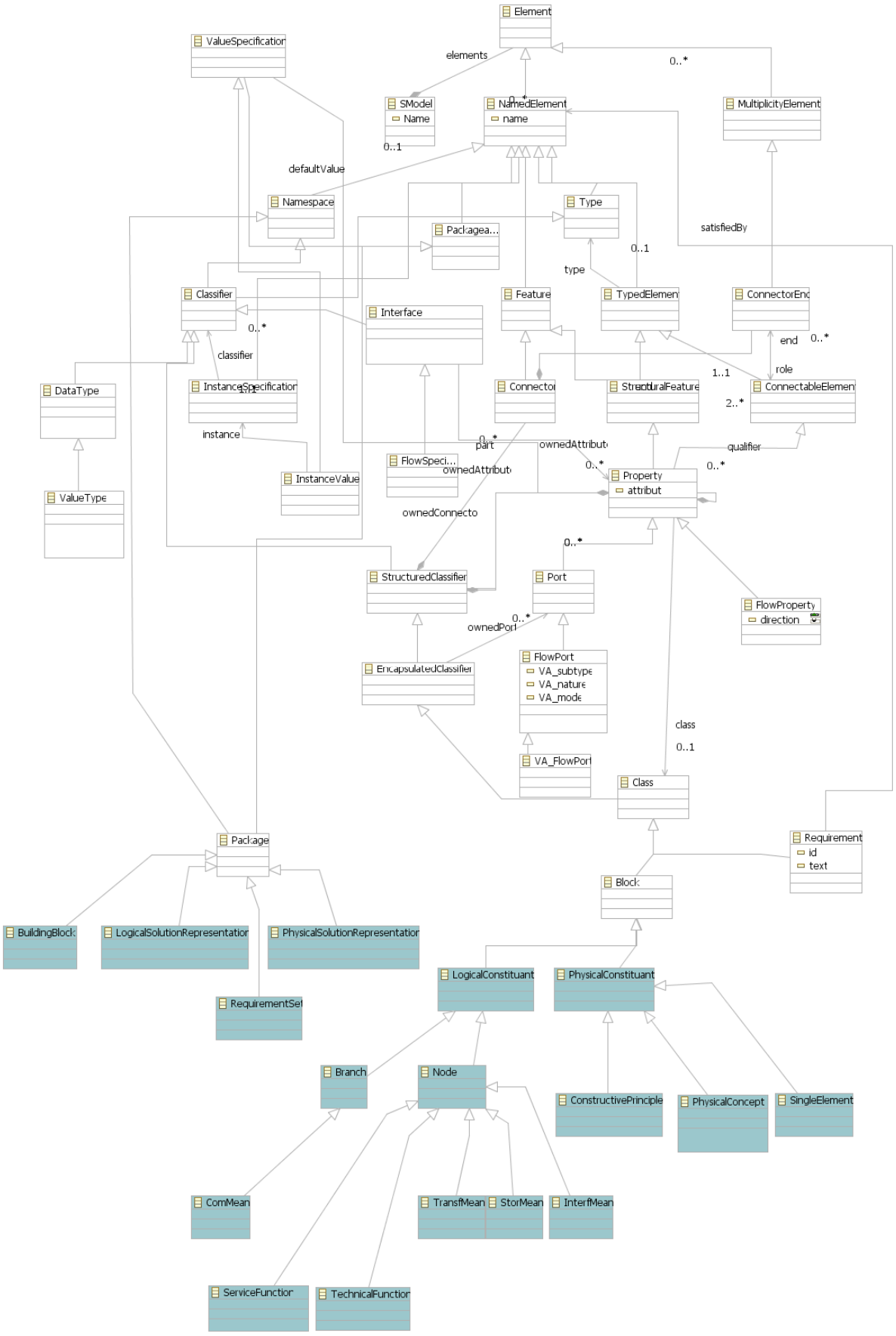
Constituants d'architecture logique :
[ ] Un constituant logique de type Branch ne peut être associé qu'à des constituants logique de type Node.
[ ] Un constituant logique d'interface doit être associé à au moins un constituant logique de type ComMean ou Branch.

[ ] Un constituant logique de type ComMean ne peut être associé qu'à des constituant logique de type TransfMean, StoringMean, HMIMean.

Constituants d'architecture physique :
[ ] Un constituant physique de type élément simple ne peut être associé qu'à d'autres éléments simple.
[ ] Un constituant physique de type principe constructif ne peut être associé qu'à d'autres éléments de type principe constructif.
[ ] Un constituant physique de type concept d'architecture physique ne peut être associé qu'à d'autres éléments de type concept d'architecture physique ou principe constructif.

6.7. ANNEXE G - Métamodèle SysML

Le métamodèle présenté ci-dessous représente le sous-ensemble du métamodèle SysML utilisé pour les besoins de la méthodologie.


Element
ValueSpecification
elements
0..*
SModel
Name
0..1
NamedElement
name
MultiplicityElement
defaultValue
Namespace
Type
Packagea...
satisfiedBy
0..1
type
Classifier
Interface
Feature
TypedElement
ConnectorEnd
0..*
end
0..*
classifier
role
1..1
DataType
InstanceSpecification
1..1
Connector
StructuralFeature
ConnectableElement
2..*
instance
part
0..*
ownedAttribut
qualifier
0..*
FlowSpeci...
ownedAttribut
InstanceValue
Property
attribut
ValueType
ownedConnecto
0..*
StructuredClassifier
Port
FlowProperty
direction
0..*
ownedPort
FlowPort
VA_subtype
VA_nature
VA_mode
EncapsulatedClassifier
class
0..1
VA_FlowPort
Class
Package
Requirement
id
text
Block
BuildingBlock
LogicalSolutionRepresentation
PhysicalSolutionRepresentation
LogicalConstituant
PhysicalConstituant
RequirementSet
Branch
Node
ConstructivePrinciple
PhysicalConcept
SingleElement
ComMean
TransfMean
StorMean
InterfMean
ServiceFunction
TechnicalFunction

6.8. ANNEXE H - Règles XTend de transformation de modèle

Le code suivant permet de traduire une architecture décrite en SysML sous la forme de blocs instanciés en part, puis connectés entre eux via des ports et connecteurs, en une instance de modèle VHDL-AMS.

```
import mm_sysml; // metamodele source
import mm_vhda;  // metamodele cible
extension template::extensions;

// point d'entree : Genere un modele vhdl ams a partir du modele sysml :
toVHDAModel(mm_sysml::SModel sysml) :
      let vhda = new VAModel :
      vhda.setName("VHDL-AMS Model")->
      vhda.du.addAll(sysml.elements.typeSelect(Block).toEntityDeclaraion())
->   // appelle la fonction qui génère l'entité
      vhda.du.addAll(sysml.elements.typeSelect(Block).toArchitecture_Body(v
hda))->  // appelle le corps de l'architecture
      vhda;

// renvoie une entité générée a partir d'un bloc
toEntityDeclaraion(Block b) :
      let ed = new Entity_Declaration :
      ed.setIdentifier(b.name)->
      ed.setHeader(new Entity_Header)->
      ed.Header.setPort(new Port_Clause)->
      ed.Header.Port.setName("-- definition des ports")->
      //ed.Header.Port.setInt_dec(b.ownedPort.typeSelect(FlowPort).toInterf
aceDeclaration())->
      // Definition des ports de l'entité, traité selon la classe du port
(Signal, quantity, Terminal, Variable, Constant, File) :
      ed.Header.Port.setInt_dec(b.ownedPort.typeSelect(FlowPort).select(e|e
.type.name == "VASignal").toSignalDeclaration())->
      ed.Header.Port.int_dec.addAll(b.ownedPort.typeSelect(FlowPort).select
(e|e.type.name == "VATerminal").toTerminalDeclaration())->
      ed.Header.Port.int_dec.addAll(b.ownedPort.typeSelect(FlowPort).select
(e|e.type.name == "VAQuantity").toQuantityDeclaration())->
      //ed.Header.Port.addAll(b.ownedPort.typeSelect(FlowPort).toPortClause
())->
      ed;

toInterfaceDeclaration(FlowPort f) :
      let id = new Interface_Declaration :
      id;

toSignalDeclaration(FlowPort f) :
      let sd = new Signal_Declaration :
      sd.setIdentifier(f.name) ->
      sd.setMode(f.VA_mode) ->
      sd.setSubtype(f.VA_subtype) ->
      sd;

toTerminalDeclaration(FlowPort f) :
      let td = new Terminal_Declaration :
      td.setIdentifier(f.name) ->
      td.setNature(f.VA_nature) ->
      td;

toQuantityDeclaration(FlowPort f) :
```

```
      let qd = new Quantity_Declaration :
      qd.setIdentifier(f.name) ->
      qd.setMode(f.VA_mode) ->
      qd.setSubtype(f.VA_subtype) ->
      qd;


// renvoie une architecture générée a partir d'un bloc
toArchitecture_Body(Block b, VAModel m) :
      let ab = new Architecture_Body :
      ab.setIdentifier("arch_" + b.name)->
      // definit la reference sur l'entité correspondante, ainsi que la
reference inverse (de l'entite vers cette architecture)
      ab.setEntity(m.du.typeSelect(Entity_Declaration).select(e|e.Identifie
r == b.name).first())->
      ab.Entity.Architecture.add(ab)->
      // fixe la partie statement de l'architecture :
      ab.setStat(new Architecture_Statement_Part)->
      // instancie le(s) composant(s) dans l'architecture pour chaque
property .part du block b :
      ab.Stat.Conc.addAll(b.part.typeSelect(Property).toComponent_Instancia
tion_Statement())->
      ab.setDecla(new Architecture_Declarative_Part)->
      ab.Decla.Block.addAll(b.ownedConnector.select(e|
(e.end.get(0).role.type.name == "VASignal"||e.end.get(1).role.type.name ==
"VASignal")).toArchitectureSignal())->
      ab.Decla.Block.addAll(b.ownedConnector.select(e|
(e.end.get(0).role.type.name == "VATerminal"||e.end.get(1).role.type.name
== "VATerminal")).toArchitectureTerminal())->
      ab.Decla.Block.addAll(b.ownedConnector.select(e|
(e.end.get(0).role.type.name == "VAQuantity"||e.end.get(1).role.type.name
== "VAQuantity")).toArchitectureQuantity())->
      //ab.Decla.Block.addAll(b.ownedConnector.toArchitectureSignal())->
      ab;

// Pour chaque connecteur dans block, crée des signaux dans l'architecture
créée.
toArchitectureSignal(Connector conn) :
      let s = new Signal_Declaration :
      s.setIdentifier(conn.name)->
      s.setMode(((VA_FlowPort)(conn.end.get(0).role)).VA_mode) ->
      s.setSubtype(((VA_FlowPort)(conn.end.get(0).role)).VA_subtype) ->
      s;

toArchitectureTerminal(Connector conn) :
      let s = new Terminal_Declaration :
      s.setIdentifier(conn.name + "terminal")->
      s.setNature(((VA_FlowPort)(conn.end.get(0).role)).VA_nature) -> // va
chercher l'information sur un des deux VA flowport relié au connecteur
      s;

toArchitectureQuantity(Connector conn) :
      let s = new Quantity_Declaration :
      s.setIdentifier(conn.name + "quantity")->
      s.setMode(((VA_FlowPort)(conn.end.get(0).role)).VA_mode) ->
      s.setSubtype(((VA_FlowPort)(conn.end.get(0).role)).VA_subtype) ->
      s;
```

```
// renvoie une instanciation de composant générée a partir d'un property
(part) envoyé en paramètre.
// Cette instanciation comporte un label basé sur le nom du property,
l'entité Block qui type la property,
// et si nécessaire les parties port map et generic map qui connectent la
property avec son environnement
toComponent_Instanciation_Statement(Property prop) :
      let cis = new Component_Instantiation_Statement : // création de
l'entité d'instanciation d'un composant

      cis.setInstantiationLabel("label_"+prop.name)-> // définit le label
de la ligne d'instantiation
      // definit la chaine de caractere qui indique de quelle entité (vhdl
ams) ce composant est une instance. La difficulté est qu'il s'agit
d'établir un pointeur sur un élément appartenant au modèle cible (une
entité vhdl) et non au modèle source.
      // la difficulté est contournée en indiquant le nom du bloc typant la
propriété, car ce nom est le même que l'entité vhdl qui est crée a partir
de ce bloc :
      cis.setStrInstantiatedUnit(prop.type.name)->
      // Définit le mapping des ports qui permet de connecter le composant
à son environnement. Il faut d'abord regarder
      // Si le composant comporte des ports :
      ((EncapsulatedClassifier)prop.type).ownedPort.size > 0 ?
      // évalue le nombre de ports contenus dans le qualifieur :
      cis.setPma(prop.toPort_Map_Aspect()):
      cis.setStrInstantiatedArchitecture(((EncapsulatedClassifier)prop.type
).ownedPort.size.toString())->
      // ecrit la liste des ports formels de la propriété :
      cis.pma.setFormal((StructuredClassifier)(prop.type).ownedPort.name.to
String())->

// pour la liste des ports reels,
// Pour chaque port formel:
//          regarder la liste des connecteurs du block contenant la
propriété,
//          si un des connecteurs a comme terminaison le port concerné :
//                si ce connecteur a comme terminaison un port du block
contenant la propriété,
//                      affecter le port formel ce port comme port reel
//                sinon (ce connecteur a comme terminaison des ports de
property)
//    affecter le port formel au signal intermédiaire crée
      cis;

// appellé par la component instanciation statement pour generer le port
map aspect de la propriété
// Il y a un seul port map aspect par propriété, et il peut contenir
plusieurs mapping de ports
toPort_Map_Aspect(Property p) :
      let pma = new Port_map_aspect :
      // crée la liste des elements d'association réalisant le mapping
entre formel et reel :
      pma.setAsel(((EncapsulatedClassifier)p.type).ownedPort.typeSelect(Flo
wPort).toAssociation_Element())->
      pma.setTestAtt(p.class.ownedConnector.select(e|((e.end.get(0).role.na
me == pma.asel.get(0).Formal_Port)||(e.end.get(1).role.name ==
pma.asel.get(0).Formal_Port))).name.toString())->

      pma.asel.forAll(e|e.AssociateActualToFormalPort(p))->
      //pma.asel.get(0).Link2(p)->
```

```
      //pma.ae.setComment("test")->
      //pma.ae.fp.setName(port.name)->
      pma;

// definit le nom du connecteur qui est connecte au port formel specifie
l'element d'association ael, a partir de la propriété :
Boolean AssociateActualToFormalPort(Association_Element ael, Property prop)
:
      ael.setActual_Port(prop.class.ownedConnector.select(e|((e.end.get(0).
role.name == ael.Formal_Port)||(e.end.get(1).role.name ==
ael.Formal_Port))).name.toString())->
      true;


toAssociation_Element(FlowPort pt) :
      let assel = new Association_Element :
      // le port formel correspond simplement au nom du port définit dans
sa déclaration :
      assel.setFormal_Port(pt.name)->
      // le nom du port reel dépend de la connexion du port avec son
environnement. Il faut passer en revue l'ensemble des connecteurs situé
dans le block contenant la propriété, et voir si un de ses connecteurs a
comme terminaison le port formel correspondant.
      assel.setActual_Port(pt.name)->
      assel;


// point d'entrée pour générer un nouveau modèle SysML comprenant les
exigences dérivées :
toSysMLOutputReqModel(mm_sysml::SModel ssml) :
      let reqmode = new SModel :
      reqmode.elements.addAll(ssml.elements.eAllContents.typeSelect(Connect
or).toReq())->
      reqmode;

toReq(Connector c) :
      let req = new Requirement :
      req.setName(c.name)->
      req;
```

6.9. ANNEXE I - Règles XPAND de génération de code

Le code suivant permet d'écrire un programme en VHDL-AMS à partir du modèle VHDL-AMS.

```
«IMPORT mm_vhda»

«IMPORT extensions»

«DEFINE main FOR VAModel»

      «EXPAND genEntityDeclaration FOREACH
this.du.typeSelect(Entity_Declaration)»

«ENDDEFINE»

«DEFINE genEntityDeclaration FOR Entity_Declaration»
      «FILE this.Identifier+".vhd"»
```

```
entity «this.Identifier» is
      -- generic declaration
      -- port declaration
      Port(«FOREACH
this.Header.Port.int_dec.typeSelect(Terminal_Declaration) AS
Interface_Declaration ITERATOR iter»
            terminal
«((Terminal_Declaration)this.Header.Port.int_dec.typeSelect(Terminal_Declar
ation).get(iter.counter0)).identifier» :
      «((Terminal_Declaration)(this.Header.Port.int_dec.typeSelect(Terminal
_Declaration).get(iter.counter0))).nature»; «ENDFOREACH»
            «FOREACH
this.Header.Port.int_dec.typeSelect(Signal_Declaration) AS
Interface_Declaration ITERATOR iter_b»
            signal
«((Signal_Declaration)this.Header.Port.int_dec.typeSelect(Signal_Declaratio
n).get(iter_b.counter0)).identifier» :
«((Signal_Declaration)this.Header.Port.int_dec.typeSelect(Signal_Declaratio
n).get(iter_b.counter0)).mode»
«((Signal_Declaration)this.Header.Port.int_dec.typeSelect(Signal_Declaratio
n).get(iter_b.counter0)).subtype»; «ENDFOREACH»
            «FOREACH
this.Header.Port.int_dec.typeSelect(Quantity_Declaration) AS
Interface_Declaration ITERATOR iter_a»
            quantity
«((Quantity_Declaration)this.Header.Port.int_dec.typeSelect(Quantity_Declar
ation).get(iter_a.counter0)).identifier» :
«((Quantity_Declaration)this.Header.Port.int_dec.typeSelect(Quantity_Declar
ation).get(iter_a.counter0)).mode»
«((Quantity_Declaration)this.Header.Port.int_dec.typeSelect(Quantity_Declar
ation).get(iter_a.counter0)).subtype»; «ENDFOREACH»
      );
end entity «this.Identifier»;
      «REM» appelle la fonction expand pour chaque architecture de l'entité
actuelle : «ENDREM»
      «EXPAND genArchitecture FOREACH this.Architecture»
      «ENDFILE»
«ENDDEFINE»

«DEFINE genArchitecture FOR Architecture_Body»

architecture «this.Identifier» of «this.Entity.Identifier» is
      -- signal / terminal declaration
begin
«FOREACH this.Stat.Conc.typeSelect(Component_Instantiation_Statement) AS
Component_Instantiation_Statement ITERATOR iter3»
«this.Stat.Conc.typeSelect(Component_Instantiation_Statement).get(iter3.cou
nter0).InstantiationLabel» :  entity
«this.Stat.Conc.typeSelect(Component_Instantiation_Statement).get(iter3.cou
nter0).StrInstantiatedUnit»
(«this.Stat.Conc.typeSelect(Component_Instantiation_Statement).get(iter3.co
unter0).StrInstantiatedArchitecture») port map (
      («FOREACH
this.Stat.Conc.typeSelect(Component_Instantiation_Statement).get(iter3.coun
ter0).pma.asel AS Association_Element ITERATOR iter4»
      «this.Stat.Conc.typeSelect(Component_Instantiation_Statement).get(ite
r3.counter0).pma.asel.get(iter4.counter0).Formal_Port»=>«this.Stat.Conc.typ
eSelect(Component_Instantiation_Statement).get(iter3.counter0).pma.asel.get
(iter4.counter0).Actual_Port»,«ENDFOREACH»);

      «ENDFOREACH»
```

```
end «this.Identifier»;
«ENDDEFINE»
```

6.10. ANNEXE J - Code de simulation VHDL-AMS simplifié du système d'assistance à l'ouverture de porte.

Code du modèle Amortisseur VHDL-AMS

```
---------- VHDLAMS MODEL amort4 ----------
LIBRARY ieee;
USE   ieee.ALL;

---------- ENTITY DECLARATION amort4 ----------
ENTITY      amort4 IS

GENERIC(   va: VELOCITY :=0.039175 ;
           vb: VELOCITY :=0.04072 ;
           fa: FORCE := (50.0 * 4.45) ;          ---- Newton
           fb: FORCE := (5200.0 * 4.45) ;        ---- Newton
           kfrot: REAL:= 100.0 ;
           m1: mass:= 1.0);

 PORT (TERMINAL meca : TRANSLATIONAL);

END ENTITY amort4;

---------- ARCHITECTURE DECLARATION arch_amort4 ----------
ARCHITECTURE arch_amort4 OF amort4 IS

 QUANTITY  s  across  f  through meca to TRANSLATIONAL_REF;
 QUANTITY m : FORCE;
 QUANTITY v : VELOCITY;
 QUANTITY acc : ACCELERATION;
 QUANTITY frot, inertie :real;

      BEGIN
           v == 1.0 * s'dot ;
           frot == kfrot * v**2;
           inertie == m1 * v'dot;
           v'dot == acc;
           f == m ;

       procedural

           BEGIN
                IF (v < va) THEN
```

```
                m := fa + frot + inertie;
            ELSIF (v < vb) THEN

                m := ((fb - fa) / (vb - va) * (s'dot - va)) + frot + inertie;
            ELSE

                m := fb+frot+inertie;
        END IF;
    END procedural;

END ARCHITECTURE arch_amort4;

---------- END VHDLAMS MODEL amort4 ----------
```

modèle vérin VHDL-AMS :

```
LIBRARY ieee;
USE  ieee.ALL;

---------- ENTITY DECLARATION verin ----------
ENTITY      verin IS

GENERIC(radius1 : REAL:=0.02;
        s : REAL:=(math_pi * ( radius1**2.0))
);

PORT(TERMINAL n1, n2: fluidic;
   TERMINAL trans: TRANSLATIONAL_V
);

END ENTITY verin;

---------- ARCHITECTURE DECLARATION arch_verin ----------
ARCHITECTURE arch_verin OF verin IS

QUANTITY p  ACROSS q THROUGH n1 TO n2;

QUANTITY  position  ACROSS force  THROUGH trans TO TRANSLATIONAL_REF  ;

QUANTITY vlineaire: velocity ;

BEGIN
----p*q ==0.9 * p * q;
        p == force / s;

        vlineaire == position'dot;

        q == ( vlineaire * s );
```

```
END ARCHITECTURE arch_verin;

---------- END VHDLAMS MODEL verin ----------
```

Code du modèle vérin VHDL-AMS :

```
---------- VHDLAMS MODEL compression ----------

LIBRARY ieee;
USE    ieee.ALL;

---------- ENTITY DECLARATION compression ----------

ENTITY       compression IS


GENERIC(
        S1 : real := 3.03858e-3; -- section piston
        M : real := 1.0;     -- masse piston
        cp: real:=1004.0;
        kfrot: real:= 0.0);

PORT (terminal e1 : FLUIDIC;
        terminal piston : TRANSLATIONAL;
   QUANTITY c : IN STIFFNESS := 1.0E6;
   QUANTITY damping : IN DAMPING := 1.0E6;
   QUANTITY sul : IN REAL := 1.0;
   QUANTITY sll_val : IN REAL := 0.0);



END ENTITY compression;

---------- ARCHITECTURE DECLARATION arch_compression ---------
-

ARCHITECTURE arch_compression OF compression IS

quantity p1 across m1 through e1 to FLUIDIC_REF;                    --
pression / debit volumique
quantity x across f,f1 through piston to TRANSLATIONAL_REF; -- position /
force
quantity V1 : real := 9.1157e-4;
quantity Temp : real := 273.0;
quantity W : real := 0.0;
quantity W1 : real := 0.0;
constant r : real := 287.0;
```

```
constant Ts : real := 273.0;
constant gama : real := 1.4;
quantity v : VELOCITY;
signal lower_crossing, upper_crossing : BOOLEAN;

BEGIN
  v == x'dot;
        IF x'above(sul) USE
        f1 == c*(x - sul) + v*damping;

        ELSIF NOT x'above(sll_val) USE
        f1 == c*(-x + sll_val) + v*damping;

        ELSE
        f1 == 0.0;
END USE;
  lower_crossing <= not x'above(sll_val);
  upper_crossing <= x'above(sul);
  BREAK ON lower_crossing;
  BREAK ON upper_crossing;

break p1 => 100000.0 , x => 0.0;

V1 == 9.1157e-4 - x*S1; -- volume

        IF x'dot > 0.0 USE
                f == p1 * S1+ kfrot*v**2.0; -- force
        ELSE
                f == p1 * S1- kfrot*v**2.0;
END USE;
Temp'dot == -(gama - 1.0) * Temp / V1 * V1'dot; -- temperature
W == p1 * V1; -- travail =  integ PdV
W1 == f * x;
p1'dot == -(gama * p1 / V1) * V1'dot + (gama * r * Ts / V1) * m1;

END ARCHITECTURE arch_compression;
```

AN APPROACH FOR AERONAUTIC SYSTEMS DESIGN. APPLICATION FOR IMPROVEMENT OF SYSTEM ARCHITECTURE OF PASSENGERS DOORS SYSTEM.

ABSTRACT
This thesis lies at the crossroads of systems engineering and model-driven engineering. A set of requirements engineering and architectural design processes have been defined. We adapted MDA approach concepts to derive design and validation methods and define an information model. This information model has been implemented as a SysML profile, and allow formal verification of methodological rules and system properties.
VHDL-AMS models have been used as a complementary mean for system validation purpose. A metamodel-based transformation between SysML model and VHDL-AMS has been defined and implemented.

Keywords : System engineering, System validation, Model-based engineering, System simulation.

VERRIES Jean

APPROCHE POUR LA CONCEPTION DE SYSTEMES AERONAUTIQUES INNOVANTS EN VUE D'OPTIMISER L'ARCHITECTURE.

APPLICATION AU SYSTEME PORTES PASSAGERS

Directeurs de Thèse :

Abd-El-Kader SAHRAOUI Professeur à l'Université de Toulouse (LAAS)
Mario PALUDETTO Professeur à l'Université de Toulouse (LAAS)

Thèse soutenue le 21 Janvier 2010, au LAAS-CNRS.

RESUME

Les travaux de cette thèse se situent à l'intersection des domaines de l'ingénierie système et de l'ingénierie dirigée par les modèles. Un ensemble de processus d'ingénierie des exigences et de conception architecturale système a été proposé.
Sur la base du standard MDA, des méthodes de conception et de validation ont été définies et mise en œuvre autour d'un modèle d'information. Celui-ci a été réalisé comme un profil du langage SysML, et permet la vérification formelle de règles et de propriétés systèmes au moyen de contraintes OCL. Afin de compléter la méthodologie sur le plan de la validation système, des modèles en langage VHDL-AMS ont été intégrés à la méthodologie. Cette approche nous a amené à étudier un certains nombre de transformations de modèles, pour lesquelles nous avons choisi de travailler au niveau des métamodèles.

Mots-clés : Ingénierie système, Validation système, Ingénierie dirigée par les modèles, Simulation.

Discipline : Systèmes Informatiques Critiques.

LAAS-CNRS, 7, avenue colonel Roche, 31077 Toulouse Cedex 4