Modélisation sémantique conceptuelle pour l'ingénierie de performances comportementales de produits complexes

Serigne Diagne

UNIVERSITE DE STRASBOURG

Ecole Doctorale **M**athématiques, **S**ciences de l'**I**nformation et de l'**I**ngénieur (**MSII** ED 269)

UdS – INSA de Strasbourg

Laboratoire de **Gé**nie de la **Co**nception (**LGéCo** EA 3938)

THÈSE

Présentée par :

Serigne DIAGNE

Soutenue le : **7 Juillet 2015**

Pour obtenir le grade de :

Docteur de l'Université de Strasbourg

Discipline : Sciences pour l'ingénieur

Spécialité : Génie Informatique – Génie Mécanique

Modélisation sémantique conceptuelle pour l'ingénierie de performances comportementales de produits complexes

Jury

Rapporteurs	**:**	M. Daoud AIT-KADI, Professeur des universités, Université Laval
		M. Kondo ADJALLAH, Professeur des universités, LCOMS, Ecole Nationale d'Ingénieur de Metz
Examinateurs	**:**	M. François DE BERTRAND DE BEUVRON, MCF, ICUBE, INSA de Strasbourg
		M. Jean RENAUD, Professeur des universités, LGéCo, INSA de Strasbourg
		M. Mbaye SENE, MCF, LID, Université Cheikh Anta DIOP de Dakar
Directeur de thèse	**:**	M. Amadou COULIBALY, MCF-HDR, LGéCo, INSA de Strasbourg

UNIVERSITÉ DE STRASBOURG

Serigne DIAGNE

Modélisation sémantique conceptuelle pour l'ingénierie de performances comportementales de produits complexes

Résumé

La complexification des produits manufacturés notamment mécatroniques requiert la mise en place d'outils et méthodes pour la gestion de leur processus de conception. Ce processus va du cahier des charges à l'obtention de prototypes satisfaisant les exigences structurelles, fonctionnelles et comportementales. Pour développer des produits performants, sûrs de fonctionnement et à moindre coût tout en respectant les délais ce processus doit être maitrisé. Les travaux menés durant cette thèse ont pour objectif de proposer une démarche générique de conception et de modélisation de produits mécatroniques tout en permettant l'évaluation de leurs performances comportementales. La démarche proposée couvre tout le processus allant de la spécification des besoins à l'identification et l'élaboration de prototypes numériques des produits répondant à ces exigences. Elle est basée essentiellement sur trois étapes successives que sont la conception sémantique conceptuelle (CSC), la modélisation sémantique conceptuelle (MSC) et l'ingénierie de performances comportementales (IPC). Ces contributions théoriques sont ensuite implémentées dans un logiciel nommé Product-BPAS et développé dans cette thèse.

Mots-clés : Produit mécatronique, Conception Sémantique Conceptuelle, Modélisation Sémantique Conceptuelle, Ingénierie de Performances Comportementales, CAO-SC, Product-BPAS

Abstract

The increasing complexity of manufactured product such mechatronics products requires tools and methods to manage their design process. This process covers the steps going from the requirements specification to the definition of the digital mockup that fulfils the structural, functional and behavioral requirements. To develop high quality products with good performance and low cost while respecting delay this process must be optimized and mastered. The research conducted during this thesis is directed to propose a generic approach for mechatronics products design and behavioural performance assessment. This approach covers the process going from the specification of the requirements to the identification and the design of the digital mockups of the products that meet those requirements. This approach is essentially based on three successive steps that are conceptual semantic design (CSD), conceptual semantic modeling (CSM) and behavioral performance engineering (BPE). These theoretical contributions have been implemented in the Product-BPAS software for test and illustrate purposes.

Keywords: Mechatronic product, Conceptual Semantic Design, Conceptual Semantic Modeling, Behavioral Performance Engineering, Conceptual Semantic CAD, Product-BPAS

Remerciements

Je remercie sincèrement et chaleureusement **M. Amadou COULIBALY** qui non seulement a dirigé les travaux de cette thèse, mais a été tout au long de ces années un conseiller, un père, un oncle. Sa disponibilité, sa rigueur scientifique et ses conseils ont permis de réaliser ce travail ;
Mes remerciements sincères vont également à l'endroit de **M. Mbaye SENE**, co-encadrant de cette thèse, avec qui je travaille depuis bientôt dix ans et qui ne cesse de me remonter le moral à chaque fois que le doute m'envahit ;
Je remercie vivement **M. François DE BERTRAND DE BEUVRON** co-encadrant de cette thèse pour sa disponibilité et pour ses conseils et suggestions qui m'ont été d'une importance capitale ;

Je remercie le Service de Coopération et d'Action Culturelle (**SCAC**) de la France au Sénégal de m'avoir offert une bourse ainsi que le laboratoire **LGéCo**, à travers son Directeur **M. Jean RENAUD**, de même que **M. Denis CAVALUCCI** et **M. Remy HOUSSIN** pour leur ouverture, leur disponibilité et les conditions favorables dans lesquelles ils m'ont mis me permettant de réaliser ce travail ;
Je remercie les autorités de l'Université Assane SECK de Ziguinchor pour le soutien moral et financier et pour les facilités qui me sont accordées pour faire ce travail. Je voudrais citer parmi eux le Recteur **M. Courfia K. DIAWARA**, le Directeur de l'UFR ST **M. Alassane DIEDHIOU**, le Chef du département Informatique **M. Khalifa GAYE**, le Vice-Recteur chargé de la Recherche et de la Coopération **M. Diouma KOBOR** ainsi que **M. Salomon SAMBOU** ;

Je remercie également **M. Daoud AIT-KADI** et **M. Kondo ADJALLAH** d'avoir accepté de participer à ce jury et d'évaluer ce travail avec toute la rigueur scientifique qu'il faut ;

Je prie pour **mes parents (Matar DIAGNE et Daba TINE)**, que Dieu ait pitié de leurs âmes, et renouvelle ma fierté de les avoir comme parent. Ils m'ont inculqué le culte du travail dans la discipline, l'abnégation et la patience. Je remercie ma tante **Anta TINE**, **Papa Basse** et leur famille ainsi que mes frères et sœurs, **Pape Mamadou**, **Abdoulaye**, **Samba**, **Mor** et **Ndeye** et tous mes amis ;
Je remercie profondément ma femme, **Thioumane Diouf NDOUR**, pour sa patience, son soutien discret et efficace, ses conseils et sa compréhension malgré les longues absences. Je remercie également mon fils **Mouhamadou Moukhtada DIAGNE** de la joie et la bonne humeur qu'il a installées dans notre vie depuis sa naissance. Que Dieu lui donne longue vie et bonne santé.

Je remercie tous les **collègues de l'UASZ** de leurs conseils et de l'exemplarité des relations qui nous lient et les **doctorants du LGéCo** pour les moments de bonheur que nous avons partagés ces 3 années.

Dédicaces

À la mémoire de mon père et à celle de ma mère,

À Cheikh Mouhamadou Mouhdata MBACKE,

À ma tendre femme et mon adorable fils,

À tous mes frères, sœurs et amis,

À la mémoire de Omar DIOP

« *Trop souvent, nous imposions notre gamme aux consommateurs en leur proposant des produits et des fonctionnalités pour lesquels ils n'étaient pas prêts à dépenser leur argent... C'était une manière très onéreuse de gérer une entreprise* »

Robert Lane, Président de la Kellogg School of Management, 2003

Table des matières

Table des illustrations

Liste des tableaux

Liste des abréviations

API	: Application Programing Interface
BD	: Base de Données
BDO	: Base de Données Objet
BDR	: Base de Données Relationnelle
BDRO	: Base de Données Relationnelle Objet
BPE	: Behavioral Performance Engineering
Brep	: Boundary Representation
CAO	: Conception Assistée par Ordinateur
CAO-SC	: CAO Sémantique Conceptuelle
CdCF	: Cahier des Charges Fonctionnel
CSC	: Conception Sémantique Conceptuelle
CSD	: Conceptual Semantic Design
CSG	: Constructive Solid Geometry
CSM	: Conceptual Semantic Modeling
DSE	: Domaine de Solutions Eligibles
DSM	: Design Structure Matrix
ECS	: Espace Conceptuel de Solutions
EDI	: Environnement de développement Intégré
EOV	: Evaluation Optimisation Validation
FAST	: Function Analysis System Technique
FMDS	: Fiabilité, Maintenabilité, Disponibilité, Sécurité
GSPN	: Generalised Stochastic Petri Net
IGES	: Initial Graphics Exchange Specification
IHM	: Interface Homme-Machine
INCOSE	: International Council on Systems Engineering
IPC	: Ingénierie de Performances Comportementales
ISO	: International Standardisation Organisation
JSDAI	: Java Standard Data Access Interface
MCD	: Modèle Conceptuel de Données
MDT	: Mean Down Time
MLD	: Modèle Logique de Données

MSC : Modélisation Sémantique Conceptuelle

MSX-CDSM : Multi-Solution eXtended Conceptual Design Semantic Matrix

MTBF : Mean Time Between Failures

MTTF : Mean Time To Failure

MTTR : Mean Time To Repair

MUT : Mean Up Time

OMG : Object Management Group

PHP : Hypertext Preprocessor

PLib : Part Library

Product-BPAS : Product Behavioral Performance Analysis System

RdP : Réseau de Pétri

SADT : Structured and Analysis Design Technique

SA-RT : Structured Analysis – Real Time

SDAI : Standard Data Access Interface

SDF : Sûreté De Fonctionnement

SET : Spécification du standard d'Echange et de Transfert

SGBD : Système de Gestion de Base de Données

SGBDRO : Système de Gestion de Base de Données Relationnelles Objet

SQL : Structured Query Language

STEP : STandard for the Exchange of Product model data

SysML : System Modeling Language

TMD : Temps Moyen de Disponibilité

TMI : Temps Moyen d'Indisponibilté

UML : Unified Modeling Language

VDA-FS : Verband Der Automobilinsdustrie Flächen Schnittstelle

X-DSM : eXtended Design Structure Matrix

X-CDSM : eXtended Conceotual Design Semantic Matrix

XMI : XML Metadata Interchange

XML : eXtended Markup Language

Chapitre 1 : Introduction Générale

Chapitre 1

Introduction générale

1.1. Contexte

Le développement croissant de la technologie et l'augmentation des exigences des utilisateurs poussent les fabricants à améliorer de plus en plus les performances des produits manufacturiers modernes. Dans le même temps, ces produits deviennent chaque jour plus complexes car intégrant des composants de technologies différentes et avec plus de fonctionnalités. Ainsi, le besoin d'évaluer leurs performances dès la phase de conception augmente. L'analyse des performances d'un produit en conception permet de prédire son comportement, d'anticiper sur ses défaillances, de faciliter sa maintenance, d'améliorer sa productivité, de réduire son coût, de respecter les délais de fabrication, etc. Pour cela il est nécessaire de pouvoir concevoir la représentation la plus fidèle possible du produit à fabriquer. Cette représentation peut se faire avec un langage de modélisation, un logiciel de modélisation géométrique 3D, un outil de création de prototypes physiques, les matrices DSM, etc. Ces outils présentent des lacunes pour une représentation de familles de produits mécatroniques en vue d'une bonne évaluation de leurs performances comportementales (fiabilité, maintenabilité, disponibilité, sécurité, recyclabilité).

En effet, SysML dérivé d'UML 2 est un langage de modélisation orienté-objet conçu pour les systèmes complexes. Cependant, il ne permet pas de représenter dans un même diagramme les composants et liaisons entre eux. Par contre, une matrice DSM classique donne une représentation des composants et liaisons d'un produit, mais ne permet pas de les caractériser. Il faut aussi noter que ce type de DSM ne peut représenter qu'un produit à la fois. Les logiciels de conception assistée par ordinateur (CAO) sont des outils de modélisation géométriques 3D très utilisés pour modéliser des produits en conception. Toutefois, ils ne permettent pas de représenter toutes les données utiles pour une bonne description de ces produits (liens entre les composants, mouvements possibles entre composants, caractéristiques sémantiques des composants). De plus, ces logiciels sont orientés métier (mécanique, électronique, bâtiment, etc.) et ne permettent donc pas de modéliser des produits mécatroniques. Ils ne permettent également pas de représenter une famille de produits.

1.2. Problématique

Au vu de ce constat fait sur la phase amont de la fabrication de produits mécatroniques complexes nos recherches portent sur les trois points suivants :

1. la conception conceptuelle ;

2. la modélisation de produits complexes ;
   a) modélisation orienté-objet ;
   b) modélisation géométrique 3D.
3. l'évaluation de performances comportementales en conception.

1.2.1. Conception conceptuelle

La conception de produits complexes est généralement faite en 4 étapes : 1. l'analyse du cahier des charges, 2. la conception conceptuelle, 3. la conception préliminaire et 4. La conception détaillée [Mucalet, 2004]. Le cahier des charges contient des spécifications, contraintes et exigences que le produit doit satisfaire. De plus, pour chaque type de produit, il existe des normes nationales et/ou internationales qu'il doit respecter. La conception d'un nouveau produit doit également se faire dans un délai bien déterminé tout en respectant un budget fixé à l'avance. Les contraintes et exigences peuvent être structurelles, fonctionnelles, comportementales, financières, etc. Leur sérialisation suivant leur niveau d'importance et de priorité durant la phase de conception est importante pour l'atteinte des objectifs. Dans la plupart des travaux portant sur la conception préliminaire de produits complexes [Pahl, 2007][Ghemraoui, 2009], cette différentiation n'est pas faite. La subdivision de ces contraintes et exigences de la plus importante à la moins importante et suivant un ordre de priorité permet de partir d'un ensemble assez vaste de produits candidats à un ensemble plus réduit. De plus elle permet de distinguer plusieurs familles de produits selon les fonctionnalités appliquées, la structure spécifiée, le coût engendré.

1.2.2. Modélisation

A partir du cahier des charges et suivant l'étude qu'on en fait, le concepteur dresse une première représentation du futur produit. Cette représentation peut se faire en utilisant plusieurs outils différents tels que les langages de modélisation orienté-objet (UML, SysML, Express, etc.), les logiciels de CAO, les logiciels de simulation numérique (Open Modelica), les DSM, etc.

1.2.2.1. Modélisation orienté-objet

Le langage UML est utilisé depuis quelques années dans l'industrie informatique pour la modélisation des logiciels et des systèmes d'information. Il a montré son utilité et a beaucoup apporté dans ce domaine [OMG UML, 2005a][OMG UML, 2005b]. Cependant, il a aussi montré ses limites quant à la modélisation de systèmes industriels complexes. C'est ainsi qu'un nouveau langage de modélisation orienté-objet dérivé de UML et dédié à la modélisation de ces systèmes est adopté par l'OMG en novembre 2005 [OMG SysML, 2006][Roques, 2013]. Ce langage nommé SysML (System

Modeling Language) permet donc de modéliser les produits complexes multi-composant avec 9 diagrammes dont certains sont les mêmes que ceux de UML. Cependant, les liaisons entre composants restent à être plus caractérisées pour prendre en compte certaines informations utiles et nécessaires à l'évaluation des performances comportementales de ces produits. Dans cette thèse, nous avons choisi d'utiliser SysML pour la modélisation des produits complexes multi-composant et multi-technologique en y ajoutant une méthodologie de modélisation des caractéristiques des liaisons mécaniques et fonctionnelles.

De plus, pour représenter les composants et liaisons de manière simplifiée les DSM sont le plus souvent utilisés [Sharon, 2009][Coulibaly, 2010]. Ils permettent de représenter un produit complexe sous la forme d'un tableau à deux dimensions. Cependant, ces DSM ne permettent pas de représenter les composants et liaisons sous forme de classes d'objets comme dans le modèle SysML.

1.2.2.2. Modélisation géométrique 3D

L'une des modélisations les plus utilisées dans l'industrie manufacturière est la modélisation géométrique 3D. Elle permet de concevoir un prototype numérique en 3 dimensions pour représenter un produit mécanique, électronique, etc. à concevoir. Les outils utilisés pour concevoir de tels modèles sont des logiciels de conception assistée par ordinateur (CAO). La CAO a vu le jour vers les années 1960 aux Etats Unis et a beaucoup évolué depuis en passant de la CAO 2D (DAO: Dessin Assisté par Ordinateur) à la CAO 5D définie comme étant CAO 3D + Temps + Coût par certains chercheurs [Heesom, 2004][O'Brien, 2000] ou comme étant CAO 3D + cognition + apprentissage par d'autres [Serrafero, 2012]. Il faut, cependant remarquer qu'il reste du chemin à faire. A ce jour, il n'est pas possible de représenter un modèle conceptuel dans les logiciels de CAO actuels. De plus ces logiciels permettent de faire de la simulation numérique, mais ne permettent pas d'évaluer les performances comportementales de produit complexes telles la maintenabilité, la disponibilité, la fiabilité, etc. S'y ajoute le fait que ces logiciels sont spécialisés soit conception mécanique, soit électronique, soit génie civil, etc. Avec l'avènement des produits mécatroniques regroupant des composants mécaniques, électroniques et logiciels, il urge de passer aux outils de CAO mécatronique qui permettent de les modéliser et d'évaluer leurs performances comportementales en phase de conception.

1.2.3. Evaluation de performances comportementales en conception

Pour concevoir un nouveau produit, le cahier des charges décrivant ses fonctionnalités, ses performances, sa structure et parfois une estimation de son coût et/ou des délais pour sa commercialisation est conçu. Le produit final doit alors respecter ces exigences en plus de normes préétablies. Beaucoup de travaux ont portés sur l'évaluation des performances comportementales de

produits en phases de conception [Zwingmann, 2005][Menye, 2009][Gaye, 2011][Huang, 2011][Coulibaly, 2010]. Des méthodologies de calcul basées sur des métriques bien choisies avec des formules pour les calculer sont proposées dans ces travaux. Il reste, néanmoins, à améliorer certaines d'entre elles et à les adapter à la modélisation sémantique conceptuelle et à les intégrer dans des outils d'évaluation de performances comportementales dans le but d'automatiser leur calcul. Ces outils d'évaluation de performances comportementales doivent pouvoir s'intégrer dans les logiciels de CAO sous forme de plugin ou add-on ou fonctionner de manière autonome sous forme de standalone en supportant les formats de fichiers fournis par les logiciels de CAO.

1.2.4. Questions de recherche

A la lumière des points discutés dans les sections 1.2.1, 1.2.2 et 1.2.3, nous nous posons les questions de recherche suivantes :

1. Comment à partir d'un cahier des charges identifier les produits susceptibles de satisfaire les exigences et contraintes tout en respectant les normes internationales ?
2. Comment modéliser ces produits en utilisant :
   i. les outils de modélisation orienté-objet ?
   ii. les matrices DSM ?
   iii. les logiciels de CAO ?
3. Comment automatiser l'évaluation des performances comportementales des produits complexes en phase de conception ?

Les réponses à ces questions constituent les points qui sont énoncés dans la Section 1.3 sous forme d'objectifs à atteindre durant cette thèse.

1.3. Objectifs

Les principaux objectifs des travaux de cette thèse peuvent être organisés en 5 parties :

- **La proposition d'une méthodologie de conception sémantique conceptuelle (CSC) de produits complexes**. En effet, la prise en compte du caractère multi-composant et multi-technologique de tels produits à une étape précoce de la conception permet de réduire les délais de livraison et le coût de la conception/fabrication. En outre, l'identification des acteurs devant participer à la conception, à la modélisation et à la fabrication et leur implication dès la phase de conception conceptuelle leur permet de bien s'imprégner des objectifs visés et d'améliorer la gestion des informations entre eux ;
- **La proposition d'une méthodologie de modélisation sémantique conceptuelle (MSC) de**

produits complexes. Au sortir de la conception conceptuelle, plusieurs familles de produits peuvent remplir les exigences et contraintes du cahier des charges. La modélisation de ces familles de produits est une étape très importante vers l'évaluation des performances comportementales de leurs instances. Ainsi, une méthodologie de modélisation sémantique conceptuelle permettant de modéliser ces familles de produits tout en facilitant l'extraction des données pour l'évaluation des performances comportementales est nécessaire ;

- **La proposition d'un algorithme d'ingénierie de performances comportementales (IPC) de produits complexes en phase de conception.** Les modèles sémantiques conceptuels obtenus durant la phase précédente servent d'entrée à l'ingénierie des performances comportementales. Cette troisième phase vise l'amélioration de métriques pour l'évaluation de performances comportementales de produits complexes. Elle propose également un algorithme permettant l'ingénierie de performances comportementales de ces produits ;
- **La description des fonctionnalités d'une nouvelle génération de systèmes de CAO appelée CAO sémantique conceptuelle (CAO-SC)**. La CAO classique ne permet pas actuellement de modéliser une famille de produits. Pour aller vers la CAO, le concepteur est obligé d'instancier le modèle pour ne représenter qu'une instance de produit. Dans le but de modéliser une famille de produits en CAO, une nouvelle génération de CAO est décrite pour supporter la modélisation sémantique conceptuelle ;
- **Le développement d'un logiciel d'aide à la MSC et à l'IPC de produits mécatroniques en phase de conception**. Jusque-là la création des modèles SysML et DSM est faite séparément. De même, l'évaluation de performances comportementales est faite pour un domaine comportemental choisi et s'applique sur une instance donnée. L'automatisation du passage d'un modèle SysML à la matrice correspondante ainsi que celle du processus d'évaluation de performances comportementales de familles de produits complexes reste alors un domaine à explorer. Nous proposons dans ce mémoire les fonctionnalités et l'architecture d'un logiciel permettant de gérer ces tâches. En guise d'illustration le module d'évaluation du temps de remplacement de composants défaillants est implémenté.

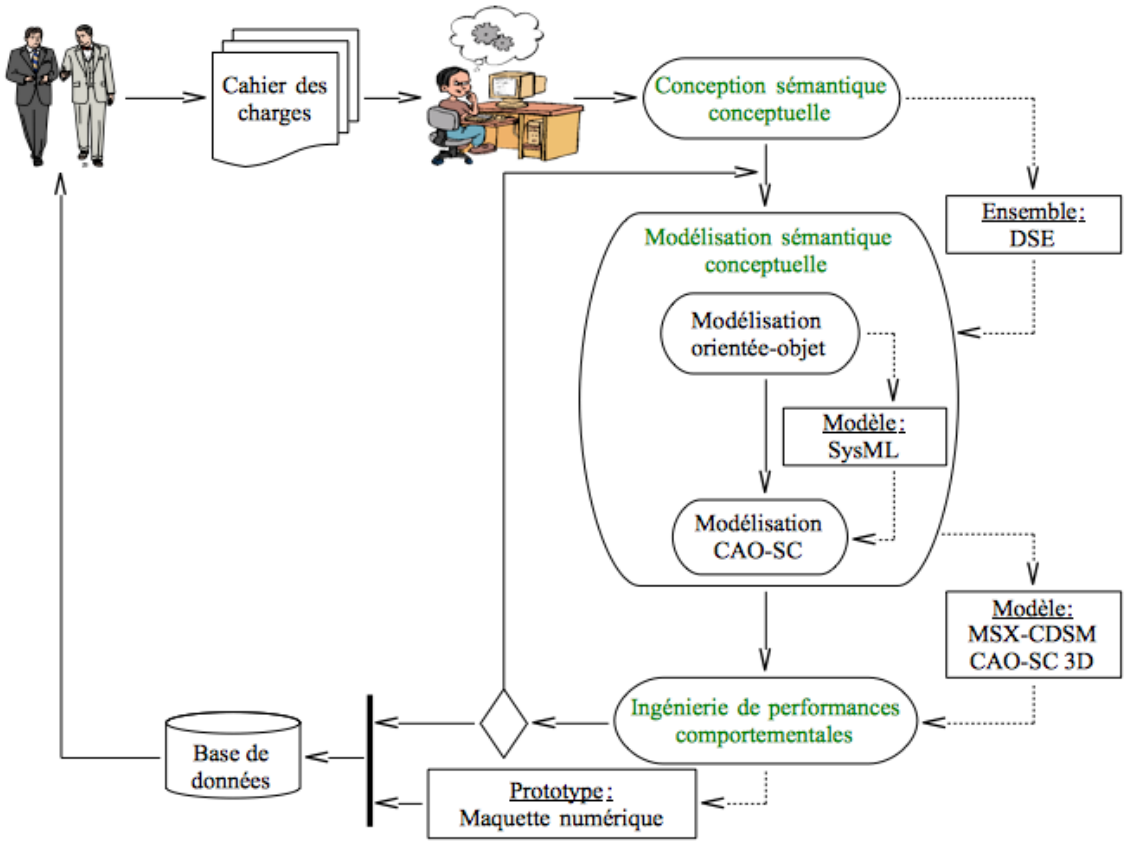

Figure 1.1 : Processus de développement de produits complexes

1.4. Contributions scientifiques

Nos contributions durant cette thèse sont :

1. **Articles de revue :**

R1. "Complex Product Modeling based on a Multi-Solution eXtended Conceptual Design Semantic Matrix for behavioral performance assessment" Computers In Industry, 2015

2. **Communications en conférences internationales :**

C1. "Towards a Conceptual Semantic Design for Mechatronic Product's Family Development" IEEE ICIDM, 2014;

C2. "Complex Mechatronic Product Modeling using a Multi-Solution, Multi-Instance eXtended Conceptual Design Semantic Matrix" DSM Conference, 2014;

C3. "Improvement of Optimal Disassembly Sequences of Complex Systems family Using Petri Nets" Elsevier CIRP LCN, 2015;

C4. "Mechatronics Product's Families Functional Modeling using SysML and X-CDSM For Reliability Assessment" IEEE SysCon, 2015;

C5. "Complex Product's Behavioral Performance Engineering at Design Stage" IAMOT, 2015.

1.5. Organisation du rapport de mémoire

La suite de ce mémoire est divisée en 4 parties : dans la première partie nous faisons un tour d'horizon des points abordés dans ces travaux, dans la deuxième partie nous donnons nos contributions concernant la CSC et la MSC, dans la troisième partie nous décrivons nos contributions portant sur l'évaluation de performances comportementales et un algorithme d'IPC et dans la quatrième partie nous parlons du logiciel permettant d'instrumenter la MSC avec les matrices X-CDSM et l'ingénierie de performances comportementales de produits mécatroniques en phase de conception. Un cas d'étude sera également présenté dans cette partie. Enfin, la conclusion générale et les perspectives sont proposées. Ces parties sont ainsi détaillées :

1. **La première partie** donne l'état de l'art sur la recherche en conception, modélisation et évaluation de performances comportementales de produits complexes. Elle est traitée dans le **chapitre 2** dans lequel nous parlons :
   a) d'abords, des systèmes complexes, puis des produits mécatroniques ;
   b) ensuite, des méthodologies de conception conceptuelle de produits complexes existantes. Nous faisons également une étude comparative de ces méthodologies avant de dégager une synthèse nous permettant de donner la direction de nos travaux.
   c) puis, de la modélisation de produits complexes en faisant un bilan des outils et méthodologies utilisés. Cette Section s'intéresse à la modélisation orientée-objet et à la modélisation CAO. Une étude comparative entre ces outils et méthodes sera également faite avant de donner les outils que nous avons choisis dans cette thèse pour modéliser les produits mécatroniques.
   d) enfin, de l'évaluation de performances comportementales de produits complexes en conception. Nous donnons, alors des métriques et formules proposées pour évaluer les performances comportementales de produits complexes en général, celles pouvant être évaluées en phase de conception préliminaire en particulier.
2. **La deuxième partie** concerne les contributions et propositions faites durant les travaux de cette thèse et portant sur les phases de conception conceptuelle et de modélisation conceptuelle. Elle est divisée en trois chapitres :
   a) Dans le **chapitre 3**, la méthodologie de conception sémantique conceptuelle proposée dans cette thèse est exposée. Les méthodologies sur lesquelles elle s'appuie sont d'écrite et la manière dont elles sont utilisées est détaillée.
   b) Dans le **chapitre 4**, les outils de modélisation choisis sont présentés. Les méthodologies de modélisation sémantique conceptuelle proposées sont décrites et les modes de passage d'un modèle à l'autre sont exposés.

c) Dans le **chapitre 5**, les fonctionnalités d'une nouvelle génération de CAO appelée CAO sémantique conceptuelle (CAO-SC) sont décrites et ses apports par rapport à la CAO classique sont donnés. L'utilisation des normes STEP et PLib dans la CAO-SC est expliquée.

3. **La troisième partie** traite les contributions concernant l'évaluation de la maintenabilité et de la fiabilité et propose un algorithme d'ingénierie de performances comportementales en phase de conception préliminaire. Elle contient un chapitre :
   a) Dans le **chapitre 6**, nous décrivons l'évaluation de performances comportementales basée sur la modélisation sémantique conceptuelle. Dans ce chapitre nous proposons une méthodologie de recherche de chemin optimal de désassemblage basée sur les réseaux de Pétri. Nous avons également proposé de nouvelles formules de calcul du temps de remplacement de composants défaillants et de la fiabilité de produits basée sur la possibilité ou non d'accomplir les fonctionnalités pour lesquelles ils sont conçus. Le processus d'ingénierie de performances comportementales allant de l'ensemble des matrices X-CDSM des familles de produits à leurs instances qui satisfont les exigences du cahier des charges y est également décrit.
4. **La quatrième partie** décrit les fonctionnalités et l'architecture d'un logiciel d'aide à la modélisation conceptuelle sémantique et à l'ingénierie de performances comportementales de produits complexes en phase de conception. Dans cette partie, nous donnons également un exemple permettant d'illustrer les propositions faites durant nos travaux. Elle contient deux chapitres :
   a) Dans le **chapitre 7**, l'architecture du logiciel Product-BPAS (Product Behavioral Performance Analysis System) est décrite. Ses interactions avec les systèmes de CAO classiques seront également décrites ;
   b) Dans le **chapitre 8**, l'implémentation du logiciel est décrite et son interface graphique ainsi que quelques démonstrations seront présentées. Un cas d'étude permettant d'illustrer les concepts proposés y est présenté. Les différentes méthodologies proposées durant cette thèse (Conception sémantique conceptuelle, modélisation sémantique conceptuelle, évaluation de performances comportementales) seront appliquées à ce cas d'étude pour illustrer leur faisabilité et leur apport.
5. Enfin, nous faisons le bilan des travaux effectués durant cette thèse. Ainsi, les réponses apportées aux questions de recherche sont énumérées et leurs apports sont déclinés. Les limites de ces réponses et la direction de nos futurs travaux sont également données.

Première partie : Etat de l'art

Chapitre 2 : Etat de l'art sur le domaine d'étude et les outils utilisés

Chapitre 2 : Etat de l'art sur le domaine d'étude et les outils utilisés

Chapitre 2

Etat de l'art sur le domaine d'étude et les outils utilisés

Introduction

La fabrication de nouveaux produits est un processus allant de la mise en place du cahier des charges à l'obtention du produit fini. Ce processus peut être subdivisé en plusieurs étapes parmi lesquelles :

- la mise en place du cahier des charges ;
- la conception conceptuelle ;
- la modélisation ;
- l'évaluation des performances comportementales ;
- le prototypage et l'usinage.

De nombreux travaux ont porté sur une ou plusieurs de ces étapes ces dernières années pour améliorer les performances des produits et leur adaptation aux besoins des utilisateurs. Nous donnons dans ce chapitre l'état actuel des recherches sur les différentes étapes du processus de développement de nouveaux produits complexes. Nous commençons, cependant, par définir notre objet d'étude à savoir les produits complexes et les produits mécatroniques en particulier.

2.1. Produit complexe

Dans cette section, nous définissons notre objet d'étude. Ainsi, nous donnons d'abord une définition générale de la notion de système complexe dans la Section 2.1.1, puis une définition permettant de comprendre les produits complexes dans la Section 2.1.2, enfin nous nous intéressons aux produits mécatroniques sur lesquels porte notre étude dans la Section 2.1.3. La notion de complexité des produits mécatroniques est étudiée dans la Section 2.1.4. Dans la Section 2.1.5 nous parlons des domaines d'utilisation des produits mécatroniques et une synthèse est donnée dans la Section 2.1.6.

2.1.1. Système complexe

2.1.1.1. Qu'est-ce qu'un système ?

Le mot "système" est issu du grec ancien "systema" signifiant "ensemble organisé" [web 1]. Il est défini par l'INCOSE comme étant :

« ***Un ensemble d'éléments interdépendants qui interagissent entre eux de façon organisée et***

forment un ensemble unique » [INCOSE, 2004]

2.1.1.2. Notion de complexité

Le mot "complexe" est issu du latin "cum plexus" qui fait référence au nombre élevé d'interactions et au fait que beaucoup d'éléments soient liés entre eux. La complexité, quant à elle, est une notion utilisée en philosophie, en épistémologie, en sociologie, en écologie, en physique, en biologie, en ingénierie, en informatique, en sciences de l'information [web 2]. La définition que l'on peut lui donner est donc fortement liée au domaine d'application. Il faut noter cependant que la notion de complexité d'une chose fait référence le plus souvent à la difficulté de le comprendre, de le saisir, de le représenter, de l'étudier, de prévoir son comportement, etc.

2.1.1.3. Qu'est-ce qu'un système complexe ?

D'après A. Lesne [web 3] un consensus existe sur des propriétés communes de la plupart des systèmes complexes :

- Le système est composé d'un grand nombre d'éléments ;
- Souvent les éléments sont de plusieurs types et possèdent une structure interne qui ne peut être négligée ;
- Les éléments sont reliés par des interactions non linéaires, souvent de différents types ;
- Le système est soumis à des influences extérieures à différentes échelles.

Rouquier et Bonneaud donnent entre autres caractéristiques d'un système complexe le fait qu'il contient beaucoup d'entités interconnectées avec beaucoup d'interactions pendant son évolution. Il présente un graphe d'interaction non trivial, avec boucles de rétroaction et est dépourvu d'organisation centrale [Rouquier, 2005][Bonneaud, 2008].

La définition suivante est donnée pour décrire les systèmes complexes :

« ***Un système complexe est un ensemble constitué d'un grand nombre d'entités en interaction qui empêchent l'observateur de prévoir sa rétroaction, son comportement ou évolution par le calcul*** » [web 4].

A la lumière de tout ce qui précède, nous avons donné une définition de système complexe :

« ***Un système complexe est un ensemble d'éléments interdépendants et organisés, présentant des rétroactions et ayant un graphe d'interaction non trivial avec une structure difficile à représenter et à étudier, un comportement difficile à prévoir et qui fournit des fonctionnalités dont la valeur ajoutée est supérieure à celles des éléments qui le composent*** »

2.1.2. Produit complexe

Un produit est caractérisé par un ensemble de **constituants** (matériels technologiques, logiciels, opérateurs humains, matériaux, procédures, services) qui sont en forte interaction et écoulent des **flux** de matières, d'énergies, et d'informations dans un environnement ou contexte donné. Cet ensemble satisfait des **besoins**, des attentes ; il accomplit une **mission** assortie d'**objectifs** prescrits permettant d'atteindre ou de répondre à une **finalité** [web 5].

Pour définir complètement un produit, il est nécessaire d'instancier chacun des termes ci-dessus et caractériser l'ensemble selon 4 aspects majeurs (Figure 2.1) :

- L'aspect attentes, besoins et exigences qui explique l'attendu (le prescriptif) en terme de finalité, de mission ou de services, d'objectifs ou de performances, de concepts d'utilisation, de scénarios opérationnels et de contraintes ;
- L'aspect frontière du produit qui explique ses limites, ses interactions et ses interfaces avec l'extérieur (nommé contexte d'utilisation) ;
- L'aspect architecture qui explique son fonctionnement et sa structure physique/organique en constituants concrets ;
- L'aspect composition hiérarchique en blocs qui explique la structure du produit en couches et en sous-systèmes et constituants terminaux.

« ***Un produit complexe est donc un système complexe échangeant avec le milieu extérieur et dont les constituants, pouvant être regroupés dans des sous-ensembles pouvant être autonomes, sont reliés par des interactions physiques et fonctionnelles et des échanges de flux de différentes sortes pour la réalisation d'un objectif donné*** »

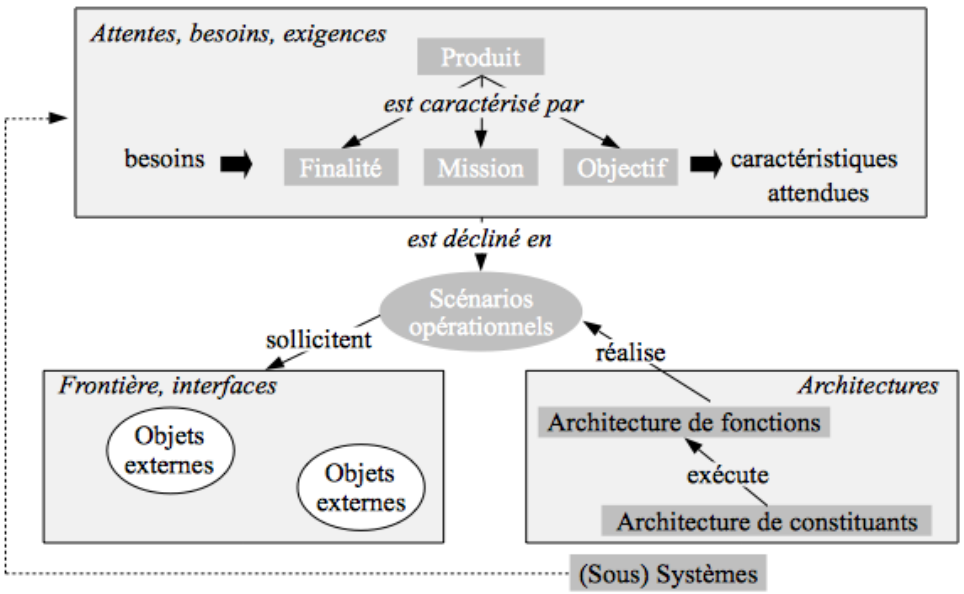

Figure 2.1 : Modèle de produits complexes [web 5]

2.1.3. Produit mécatronique

La mécatronique a été introduite par un ingénieur de la compagnie japonaise "Yasukawa Electric Corporation" en 1969. A l'origine, le mot *mechatronics* est une concaténation des mots *mechanism (mecha)* et *electronics (tronics)* [Handbook]. La définition a évolué depuis et la mécatronique est devenue aujourd'hui une nouvelle approche de l'ingénierie de conception. Elle permet d'intégrer des fonctions intelligentes dans les produits : optimisation de la consommation énergétique, meilleur intégration dans l'environnement, réactivité accrue par rapport aux phénomènes extérieurs [web 6]. Le mot *mécatronique* est apparu officiellement en France dans le Larousse 2005 [web 7]. Plusieurs définitions existent parmi lesquelles celle :

- du journal IEEE *Transactions on Mechatronics* : "***Mechatronics is the synergetic combination of precision mechanical engineering, electronic control and systems thinking in the design of products and manufacturing processes***" [Comerford, 1994] ;
- de W. Bolton : "**A mechatronics system is not just a marriage of electrical and mechanical systems and is more than just a control system; it is a complet integration of all of them**" [Bolton, 1999] ;
- du Larousse 2005 : "***une technique industrielle consistant à utiliser simultanément et en symbiose la mécanique, l'électronique, l'automatisme et l'informatique pour la conception et la fabrication de produits nouveaux***" [Larousse, 2005] ;
- de la norme NF E 01-010 "***une démarche visant l'intégration en synergie de la mécanique, l'électronique, l'automatique et l'informatique dans la conception et la fabrication d'un produit en vue d'augmenter et/ou d'optimiser sa fonctionnalité***" [NF E 01-010].

La notion de produit mécatronique est définie par la norme NF E 01-010 comme étant :
"***Un produit ayant la capacité de percevoir son milieu environnant, de traiter l'information, de communiquer et agir sur son milieu, et présentant un niveau complet d'intégration mécatronique, du point de vue fonctionnel et physique***" [NF E 01-010].

Les domaines technologiques et leurs sous-domaines sont représentés sur la Figure 2.2.

Un produit mécatronique contient essentiellement 3 parties [web 8][web 7] :

- La partie opérative qui est le squelette du produit. Elle regroupe l'ensemble des mécanismes et actionneurs du système global. Elle est à dominante mécanique et électromécanique ;
- La partie commande qui est l'intelligence embarquée du produit. Elle assure la gestion de son comportement global. Elle est à dominante électronique et informatique temps réel ;
- La partie interface qui donne la forme géométrique du produit et assure le dialogue entre les parties opérative et commande. Elle est à dominante ergonomique et esthétique. Elle permet

également le dialogue entre le produit et ses utilisateurs (IHM).

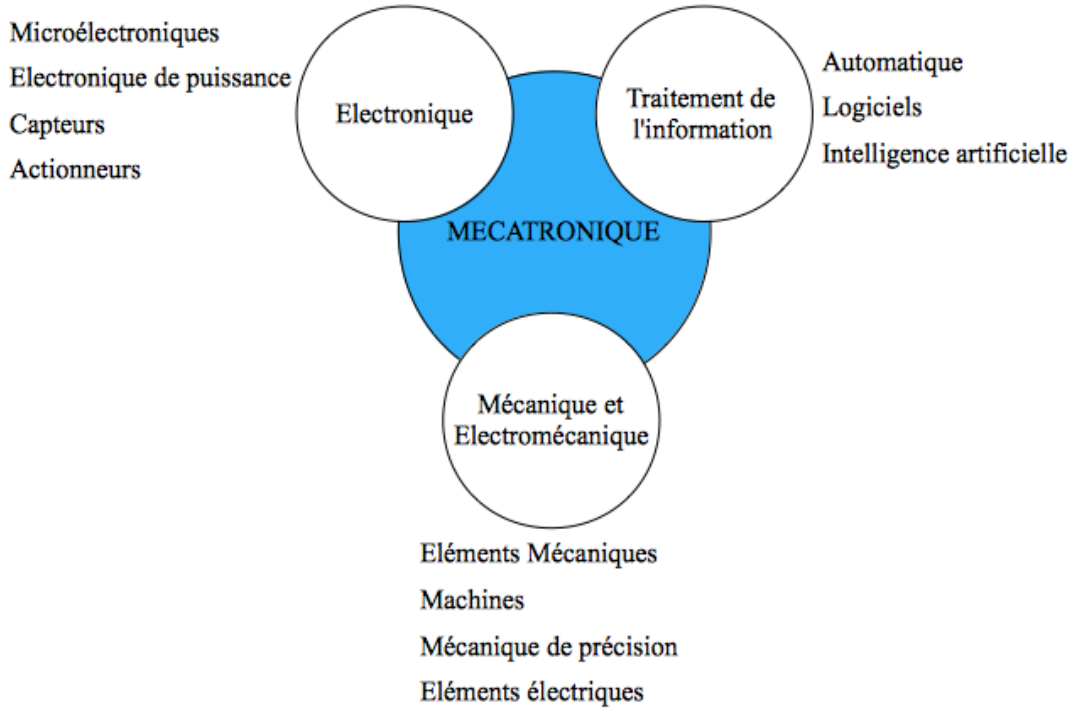

Figure 2.2 : Mécatronique : Intégration synergique de différentes disciplines [ElFeki, 2011]

Nous donnons dans la Figure 2.3 la chaine classique de la mécatronique.

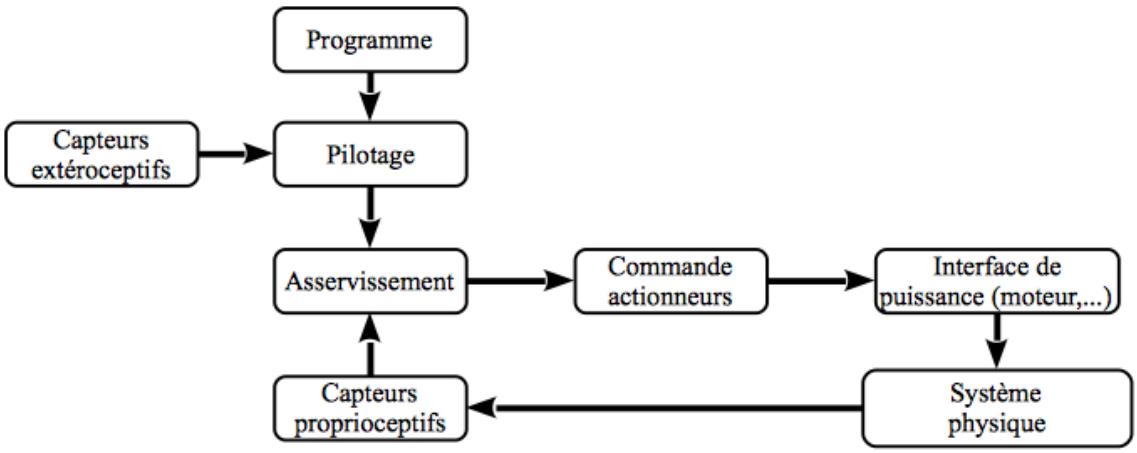

Figure 2.3 : Chaine classique de la mécatronique [Gies]

La mécatronique est en pleine expansion, aussi bien au niveau des équipements industriels, que des composants et des processus de production. La complexité d'un produit mécatronique peut découler de sa structure, ses fonctionnalités, les technologies qu'il englobe, etc. Ces produits peuvent combiner plusieurs composants de technologies différentes avec des structures et fonctionnalités différentes. Dans la section suivante, nous parlons de la notion de complexité des produits mécatroniques.

2.1.4. Complexité d'un produit mécatronique

Les méthodes pour caractériser la simplicité ou la complexité d'un produit sont nombreuses (nombre de composants, diversité des composants, niveau d'interdépendance entre eux, comportement du produit, informations qu'il manipule, etc.) [Simon, 1990]. Pour les produits mécatroniques, la complexité est généralement structurelle, technologique et/ou fonctionnelle.

2.1.4.1. Complexité structurelle

La complexité structurelle d'un produit mécatronique implique essentiellement ses composants, leur hiérarchisation, les liaisons qui les relient, les interactions fonctionnelles entre eux, etc. Un composant mécatronique est défini comme étant "*un élément constitutif d'un système mécatronique présentant un niveau partiel d'intégration mécatronique, du point de vue fonctionnel et physique, associant les technologies de la mécanique et de l'électronique et permettant le traitement de l'information*" [Devalan, 2010][NF E 01-010]. L'intégration physique est définie par la norme française NF E 01-010 comme étant "*une interpénétration des supports mécanique et électronique (qui embarquent les fonctions automatique et informatique)*" [NF E 01-010]. L'obligation d'une description fine des composants et de leur intégration (interconnexion, assemblage, câblage, ...) nécessaire à leur réalisation (géométrique 2D, 3D) avec optimisation volumique, simulation multi-physique, et vérification des contraintes de montage et de fabrication est soulignée par Devalan [Devalan, 2010].

On voit, ainsi, le niveau de complexité structurelle que présentent les produits mécatroniques dès leur phase de conception. A cela s'ajoute la complexité technologique

2.1.4.2. Complexité technologique

La complexité technologique en mécatronique vient de la nécessité d'une intégration complète de plusieurs technologies dans un produit. En cela, Bolton disait qu'un produit mécatronique n'est pas seulement un mariage entre l'électronique et la mécanique et est plus qu'un simple système de contrôle, mais une intégration complète de tout ceci [Bolton, 1999]. Ainsi, pendant la phase de conception, une collaboration étroite des concepteurs de différentes spécialités est nécessaire. Du fait d'un long cloisonnement des experts des différents domaines (mécanique, électronique, automatique et informatique) entre eux, de la diversité des méthodes de conception, des normes à respecter, du problème de management et de l'échange de connaissances entre ces experts [Pärttö, 2012], etc. on assiste de plus en plus à la formation de purs ingénieurs mécatroniciens pour réussir cette intégration des différentes technologies (mécanique, électronique, automatique et informatique) qu'utilisent un produit mécatronique dès les premières heures de sa conception. Cette intégration technologique doit

permettre de donner aux produits plus de fonctionnalités avec des délais et coûts réduits.

2.1.4.3. Complexité fonctionnelle

La complexité fonctionnelle d'un produit mécatronique découle du besoin d'une intégration fonctionnelle des différents composants. L'intégration fonctionnelle est définie dans la norme NF E 01-010 comme étant "*un apport de fonctions de détection, de communication, de traitement de l'information et de rétroaction aux fonctions mécaniques de base*" [NF E 01-010]. Pendant la phase de conception d'un produit mécatronique il est nécessaire de décrire les fonctions générales du produit et les fonctions associées aux différents modes de fonctionnement. De même la description des relations entre les différentes fonctions et les liens structurels, le comportement des composants interconnectés par des flux de puissance et d'information et des lois de commande est obligatoire. L'utilisation de modèles de simulation permettant une analyse multi-domaine du système s'impose, ainsi, aux concepteurs [Devalan, 2010]. Il est également à noter que la valeur ajoutée d'un produit mécatronique doit être supérieure à la somme des valeurs ajoutées de ses différents composants [Aprim, 2013][NF E 01-010].

Les produits mécatroniques sont de nos jours très présents dans presque tous les secteurs de la vie. Nous donnons dans la Section 2.1.5 un schéma récapitulatif des domaines d'application de tels produits.

2.1.5. Domaines d'application des produits mécatroniques

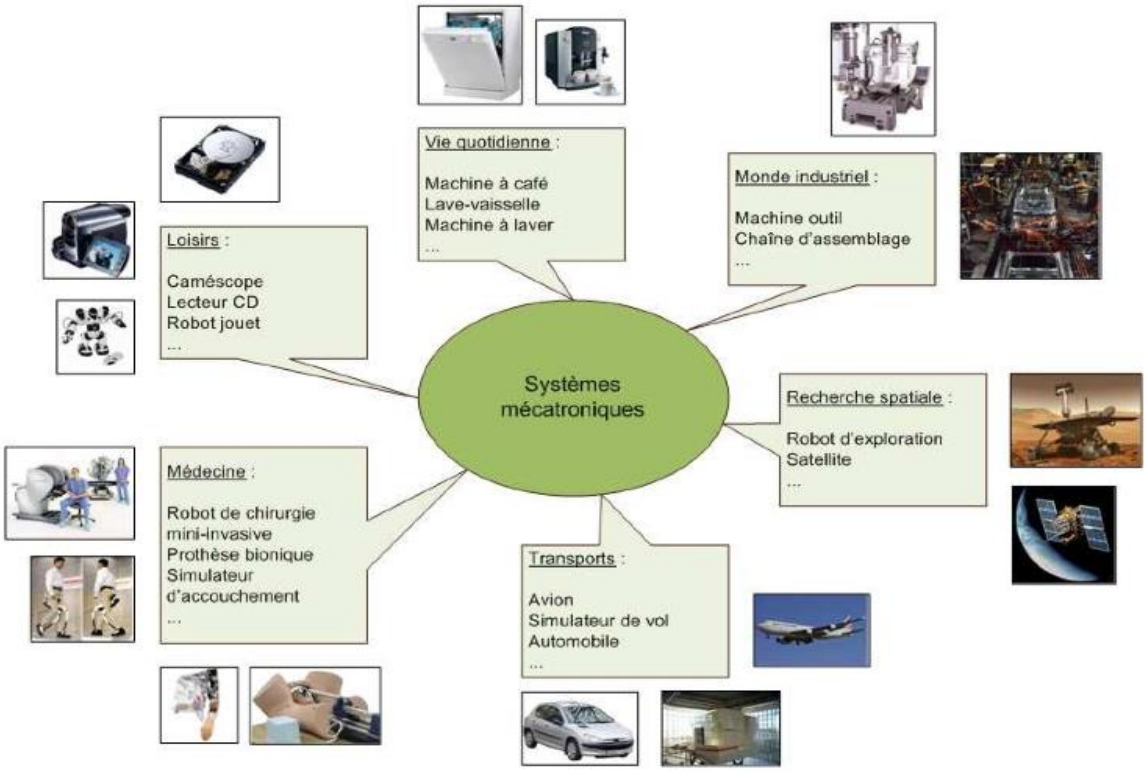

Figure 2.4 : Exemple de systèmes mécatroniques [Jardin, 2010]

Les applications potentielles des produits mécatroniques concernent quasiment tous les secteurs industriels et en particulier la santé, l'environnement, l'énergie, les transports, le spatial, etc. La Figure 2.4 donne différents domaines d'utilisation de ces produits.

2.1.6. Synthèse

La mécatronique touche plusieurs domaines technologiques et est utilisée dans presque tous les domaines socioprofessionnels. Un produit mécatronique est un système complexe de par sa structure (multi-composant et multi-physique), ses composants (mécanique, électronique, informatique) et ses fonctionnalités. Un système mécatronique intègre également l'automatisme dans son fonctionnement. C'est donc un produit multi-composant, multi-technologique, multi-physique. Sa conception demande l'intervention de plusieurs acteurs de diverses compétences qui doivent travailler en synergie dès la phase de conception préliminaire. Ainsi, la Section 2.2 donne un état des lieux de la conception conceptuelle des produits mécatroniques.

2.2. Conception de produits mécatroniques

L'étude des théories et méthodologies de conception de produits complexes est un domaine qui a fait l'objet de beaucoup de travaux de recherche ces dernières années [Wilhelms, 2005][Motte, 2008]. Pour concevoir des produits modernes (multi-technologiques, multi-composants), performants et robustes tout en réduisant les coûts et les délais, les concepteurs doivent prendre en compte les exigences et contraintes tout au long du processus de conception. Les contraintes peuvent provenir du client, des concepteurs, du processus de conception, du processus de fabrication, des outils disponibles, etc. Plusieurs travaux ont porté sur la conception des produits mécaniques, des produits électroniques, des logiciels ou même sur des produits intégrant plusieurs de ces domaines. Ainsi, nous donnons quelques définitions de la conception avant de parler des types et approches de conception existants et finir par positionner nos travaux de recherche.

2.2.1. Qu'est-ce que la conception ?

La conception est un processus complexe permettant de passer d'un besoin (fonctionnalités, performances) formulé avec des contraintes à la description détaillée d'une ou plusieurs solutions. Le besoin et les contraintes sont généralement écrits sous forme textuelle dans un cahier des charges. Le processus de conception est généralement itératif avec comme objectif de répondre aux exigences tout en satisfaisant les contraintes. Ainsi, Gero définit la conception comme étant :

« ***Un processus permettant d'aboutir à une forme à partir d'une description sans forme*** » [Gero, 1989].

Dixon considère la conception comme une activité intellectuelle et cognitive très complexe [Dixon, 1988]. Dans le domaine mécanique, l'activité de conception peut être définie comme étant : « ***Un processus de création et de définition d'une ou plusieurs descriptions d'un produit*** » [Hadj-Hamou, 2002].

Jeantet définit la conception d'un produit comme étant :

« ***L'expression d'un besoin à la définition des caractéristique d'un objet permettant de le satisfaire et à la détermination de ses modalités de fabrication*** » [Jeantet, 1998].

A partir de ces définitions, nous pouvons définir la conception comme étant :

« ***Un processus permettant, à partir d'un besoin avec des contraintes, de spécifier les caractéristiques d'un objet les satisfaisant et de proposer des instructions pour sa fabrication*** »

Il existe plusieurs types de conceptions qui peuvent se faire suivant des processus différents. Nous parlons dans la section suivante, des processus de conception.

2.2.2. Le processus de conception

La conception d'un produit complexe fait intervenir des concepteurs de compétences différentes et des informations de natures différentes qui peuvent être exprimées différemment selon le domaine de compétence des concepteurs. Ces concepteurs doivent donc travailler en synergie pour proposer des solutions dont la description est claire et bien détaillée. Ces solutions doivent répondre aux attentes tout étant techniquement réalisables.

Un problème de conception est spécifié par a) un ensemble de fonctions que doit remplir le produit et un ensemble de contraintes à satisfaire, b) un ensemble de composants prédéfinis et un ensemble de relations entre ces différents composants [Chandrasekaran, 1990]. D'après cette définition, résoudre un problème de conception revient à proposer un ensemble de composants avec leurs interactions (liaisons physiques, fonctionnelles, échange de flux, etc.) permettant de réaliser les fonctions dans le respect des contraintes. Il peut cependant, arriver que certains composants n'existent pas encore au moment de la conception. Dans ce cas leur description doit permettre de les créer avant de les intégrer dans le produit.

Le processus de conception se fait en plusieurs étapes cohérentes et liées. La sortie de l'étape précédente est généralement l'entrée de la suivante. Plusieurs propositions sont données dans ce sens. Nous donnons deux propositions parmi celles-ci :

1. Le processus de conception d'un produit complexe peut être résumé en trois étapes [web 5] :
   - établissement de la comptabilité fonctionnelle et physique du produit avec les besoins et les contraintes ;
   - équilibrage de l'économie globale de la solution sur toutes les étapes de la vie du produit

(vue de l'utilisateur) ;

- recherche de l'équilibre entre contraintes, performances, coûts, délais et risques (vue du concepteur).

2. Pahl et al. proposent une décomposition du processus de conception en quatre étapes [Pahl, 1996] :
   - élaboration du cahier des charges ;
   - formalisation et spécification des principes ;
   - conception d'ensemble ;
   - conception détaillée.

Un choix fait dans une étape précédente élimine certains choix dans les étapes suivantes. Ceci n'en fait pas pour autant un processus séquentiel car c'est un processus itératif et dynamique [Pahl, 1996] [Brown, 1998].

Pour concevoir un produit, on peut partir d'idées nouvelles pour aboutir à un produit nouveau ou à une modification et amélioration de produits existant.

2.2.3. Les types de conception

Il existe deux types de conception : la conception initiale et la re-conception. Chacun de ces deux types est décomposé à son tour en deux sous-types de conception (Figure 2.5) [Serrafero, 2012].

2.2.3.1. La conception initiale

Il s'agit de concevoir et d'inventer un produit ou un processus pour la première fois, sans aucun cas de référence existants ni aucune expérience passée réussie permettant de s'inspirer pour copier, de reconcevoir ou d'améliorer des solutions existantes. Ce type de conception est elle-même divisée en deux : la conception créative et la conception innovante [Serrafero, 2012].

2.2.3.1.1. La conception créative

Elle porte sur un produit inconnu donc nouveau. L'innovation porte déjà sur la définition des fonctions du cahier des charges [STIDD]. Elle est par essence la plus aventureuse et la plus risquée de toutes puisque rien n'est connu au départ, pas même le concept physico-chimique de fonctionnement de l'objet [Serrafero, 2012]. Elle intervient quand il n'existe aucune solution produit à priori et que toutes les connaissances relatives au produit et au processus de conception sont à spécifier [Hadj-Hamou, 2002][Menand, 2002].

2.2.3.1.2. La conception innovante

Dans cette conception, la décomposition du problème est connue au départ, il s'agit alors de

définir de nouvelles solutions alternatives pour chacun des sous problèmes [Menand, 2002]. L'expression du besoin, les technologies à utiliser sont souvent connues à l'avance et sont clairement définies, mais les stratégies de conception restent à identifier [Hadj-Hamou, 2002]. Elle remet en cause certains principes de conception ainsi que le processus lui-même [STIDD]. Elle développe l'innovation projet dans l'architecture du produit à partir de la connaissance d'un concept physico-chimique de fonctionnement connu de l'objet à concevoir. L'architecture est alors totalement inconnue au début de la conception [Serrafero, 2012].

2.2.3.2. La re-conception

La re-conception, quant à elle, réutilise les idées des concepts déjà établis mais en modifiant leur structure et géométrie (soit le modèle produit) pour satisfaire un nouveau cahier des charges ou de nouvelles contraintes [Menand, 2002]. Il s'agit, alors, de re-concevoir un produit ou un process existant motivé par une logique d'optimisation des performances fonctionnelles, de l'architecture, des coûts, du poids, du volume, etc. Elle englobe la conception routinière et la conception paramétrique [Serrafero, 2012].

2.2.3.2.1. La conception routinière

Le concept et l'architecture sont connus à l'avance, mais l'objet à concevoir est alors à configuration inconnue parmi une grande variété de combinaisons et de topologies possibles [Serrafero, 2012][Menand, 2002]. Toutes les connaissances à mettre en œuvre sont totalement disponibles et identifiées. De plus, les stratégies de conception sont globalement connues à l'avance par le concepteur [Hadj-Hamou, 2002]. Le cahier des charges ne varie que d'une manière quantitative, les fonctions restent les mêmes mais les principes et le processus de conception ne changent pas [STIDD].

2.2.3.2.2. La conception paramétrique

A concept, architecture et configuration connus à l'avance, l'objet à concevoir est à valeurs des paramètres inconnues [Serrafero, 2012]. Fréquemment, le composant paramétrique est "sur étagère" fait l'objet d'un catalogue standard et l'activité de conception revient souvent à le dimensionner et à effectuer le choix correspondant au dimensionnement dans le catalogue du fabriquant éventuellement avec l'aide d'un configurateur numérique permettant de transformer le besoin fonctionnel en solution [Serrafero, 2005].

Kota et al. Considèrent qu'un processus de conception créatif devient innovant lorsque les fonctions du produit ont été définies et ce processus devient routinier lorsque les choix de principes technologiques ont été effectués [Kota, 1991]. Ainsi, ils proposent un schéma global de conception pour illustrer ce fait (Figure 2.6).

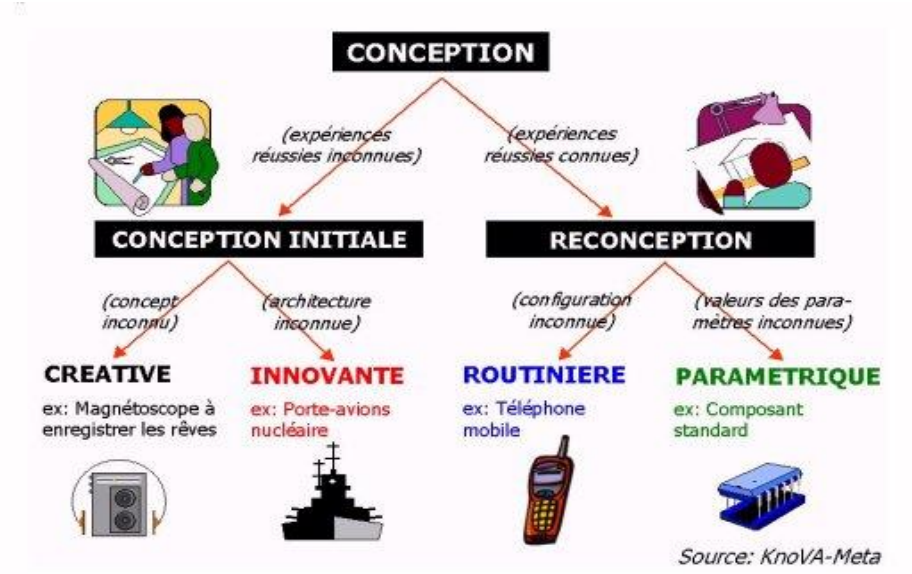

Figure 2.5 : Différentes classes de problèmes de conception [Serrafero, 2012]

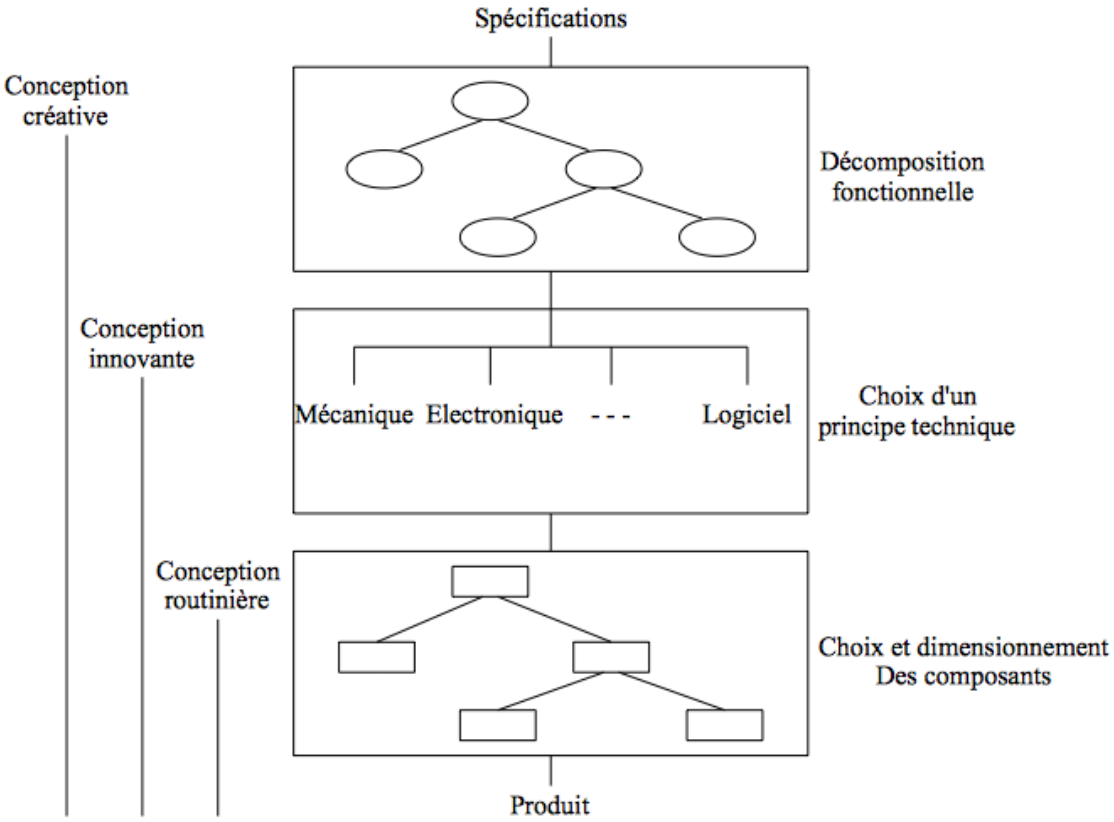

Figure 2.6 : Schéma global de conception [Kota, 1991]

2.2.4. Le cycle en V pour la conception de produits mécatroniques

Pour développer des produits robustes, performants et à moindre coût, il est nécessaire de bien maitriser le processus de conception tout en vérifiant l'atteinte des objectifs et le respect des contraintes. Une approche couramment utilisée en conception de produits est le cycle en V.

Issu du génie logiciel, le cycle en V permet l'organisation générale du travail, la décomposition et

la distribution des tâches. Il a, par la suite, été utilisé comme modèle de développement dans différents domaines : la mécanique, l'électronique [Mihalache, 2007]. Les mécatroniciens l'ont, ensuite, adapté à la conception de systèmes mécatroniques [VDI 2206]. De nos jours, le cycle en V est devenu le cycle de développement standard de l'ingénierie système [Verries, 2010]. Il permet de détecter très tôt d'éventuelles anomalies et de limiter le retour aux phases précédentes [Hammadi, 2012]. Il est composé de deux grandes parties : la partie descendante (Top down) pour la spécification et la conception du produit et la partie ascendante (Bottom up) pour l'intégration des différentes technologies et les tests pour valider les choix de conception. Ce cycle peut être appliqué à la totalité du produit à développer ou à une partie jugée critique [XP E 01-013].

La structuration du cycle en "V" permet de mettre en regard deux à deux les étapes de ses deux principales parties (Spécification fonctionnelle/Validation fonctionnelle, Spécification système/Validation système, Définition sous-système/Validation sous-système et Définition composants/Validation composants) Figure 2.7. Il permet de passer du plus général au plus détaillé lors de la réalisation, puis du plus détaillé au plus général lors de la validation.

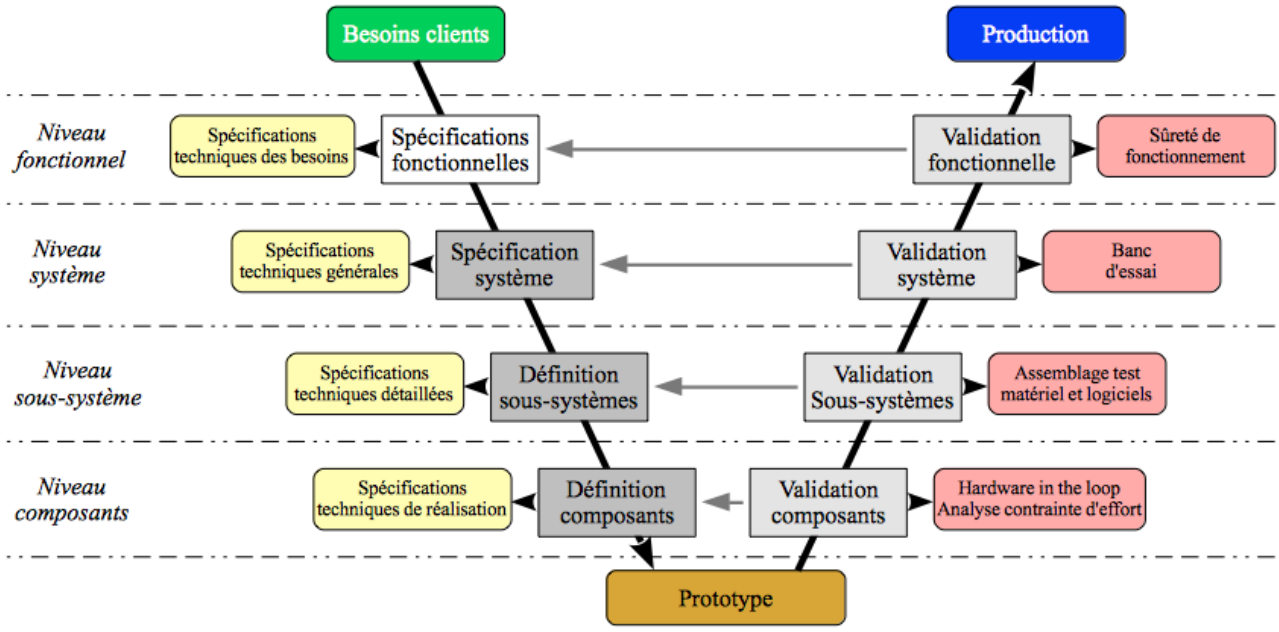

Figure 2.7 : Cycle en V pour la conception de systèmes mécatroniques [Casner, 2013]

Mihalache décompose le cycle en V en 5 parties : analyse/spécification et conception pour la partie descendante, vérification et validation pour la partie ascendante et réalisation entre les deux [Mihalache, 2007]. De manière plus détaillée, Jardin le décompose en 9 parties : spécification fonctionnelle, spécification système, définition sous-système et définition composants pour la partie descendante, validation fonctionnelle, validation système, validation sous-système et validation composants pour la partie descendante et réalisation prototype entre les deux [Jardin, 2010]. Alors, selon Jardin, les tâches de conception de systèmes mécatroniques sont divisées en quatre parties ou

niveaux :

- *Le niveau fonctionnel :* Il correspond à la définition des fonctionnalités, des interfaces, des contraintes et des exigences du système ainsi que la préparation du plan qualité, du plan de validation et de l'étude de faisabilité. La définition des contraintes de performances comportementales (fiabilité, maintenabilité, sécurité, disponibilité, recyclabilité) est aussi effectuée durant cette phase [Mihalache, 2007]. De même, les informations relatives à l'encombrement, l'impact environnemental, le coût et les délais sont prises en comptes. C'est la phase durant laquelle, la traduction des attentes exprimées par les clients en spécifications techniques est réalisée [Jardin, 2010].
- *Le niveau système :* Il correspond à la définition de l'architecture du système. C'est lors de cette phase que le choix d'un ou de plusieurs sous-système permettant de réaliser chaque fonctionnalité est fait. Les interactions que devront avoir ces différents sous-systèmes sont également proposées. Au sortir de cette étape, les spécifications techniques générales du produit sont décrites.
- *Le niveau sous-systèmes :* C'est à ce niveau que les sous-systèmes choisis dans le niveau précédent sont à leur tour décrits. Chaque sous-système peut être décomposé à son tour en sous-systèmes jusqu'aux composants élémentaires (pièces). Là aussi, à l'intérieur de chaque sous-système les interactions entre ses composants sont définies. Au sortir de cette phase, les spécifications techniques détaillées sont décrites.
- *Le niveau composant :* C'est la dernière phase de la partie descendante du cycle en V. Ce niveau achève alors la conception du produit. Les composants sont décrits en détail et les spécifications techniques de réalisation sont données. Au sortir de cette phase, le prototypage virtuel doit être possible. Ce prototype permet de tester, d'optimiser et de valider le système sans recourir à des prototypes réels.

Le niveau fonctionnel décrit ci-dessus correspond à la phase analyse/spécification d'écrite par Mihalache et les trois niveaux suivants correspondent à la phase de conception [Mihalache, 2007].

D'autres types de cycles de développement sont aussi utilisés : le cycle en cascade, le cycle en spirale sont les plus connus [Mihalache, 2007][Hammadi, 2012][Trabelsi, 2014]. Nous nous sommes essentiellement intéressés au cycle en V sur lequel s'appuie nos travaux en matière de conception. Ainsi, nous présentons dans la Section 2.2.5 l'approche de conception systématique et donnerons son intégration dans le cycle en V.

2.2.5. L'approche de conception systématique

L'approche de conception systématique est un processus de conception qui ne s'intéresse pas seulement au problème à résoudre, mais aussi à l'environnement et aux autres systèmes liés au problème [web 9]. Elle vise à optimiser la conception de produits en adoptant la stratégie appelée breadth-first/top-down. Cette stratégie consiste à partir d'un grand nombre de solutions candidates (breadth-first) pour aboutir aux solutions concrètes (top-down) [Motte, 2008]. Cette approche comprend quatre phases selon Pahl et Beitz [Pahl, 2006] :

- *Planification et classification des tâches :* Durant cette phase, la connaissance du produit à développer est recherchée. Une étude de marché est faite est les retours d'expériences, s'il en existe, sont répertoriés. La liste des exigences est créée suivant les contraintes et les besoins [Pahl, 2006][Rahman, 2007].
- *Conception conceptuelle :* Durant cette phase la principale solution aux problèmes liés aux exigences du produit sera déterminée. Cela se fait en faisant abstraction des problèmes secondaires, en créant les structures de fonction, en cherchant des principes de travail convenables, et en les combinant dans une structure de travail [Pahl, 2006][Rahman, 2007].
- *Embodiment design :* Au cours de cette phase, un concept est élaboré dans la structure de conception d'un système technique en conformité avec les critères techniques et économiques. Ainsi, un plan préliminaire est produit et la disposition définitive sous forme d'une structure de conception (plan d'ensemble) est déterminée [Pahl, 2006][Rahman, 2007].
- *Conception détaillée :* Pendant cette dernière phase, l'arrangement, les formes, les dimensions et les propriétés de surface de toutes les pièces individuelles sont enfin fixées. Les documents concernant les matières spécifiées, les possibilités de production, le coût estimé, et tous les dessins ainsi que les autres documents de production doivent être produits. Tous les documents nécessaires à la spécification du produit et de sa production devraient être prêts [Pahl, 2006][Rahman, 2007].

Au vu des différentes phases qui constituent l'approche de conception systématique, elle est parfaitement compatible avec le cycle en V de la Figure 2.7. Nous donnons dans le Tableau 2.1 les correspondances entre les phases du cycle en V et de l'approche systématique.

Tableau 2.1 : Correspondance des phases du cycle en V et de l'approche systématique

Cycle en V	Conception systématique
Spécification des exigences	Planification et classification des tâches
Spécification fonctionnelle	Conception conceptuelle
Spécification système	Conception conceptuelle et Embodiment design
Définition sous-systèmes	Embodiment design
Définition composants	Conception détaillée

D'autres approches de conception sous souvent utilisées approche itérative [Whitten, 2004], approche systémique [Menand, 2002][Demri, 2009], etc. Nous nous sommes essentiellement intéressés à l'approche systématique sur laquelle s'appuie notre travail. Après synthèse des approches décrites ci-dessus dans la Section 2.2.6, nous positionnons nos contributions dans la Section 2.2.7.

2.2.6. Synthèse et analyse

La conception est la première étape dans le développement de nouveaux produits. Il existe principalement deux types de conception que sont la conception initiale et la re-conception. Chacun de ces deux types de conception est à son tour décomposé en deux sous-types de conception. Le processus de conception peut être globalement décomposé en 3 étapes (voir Section 2.2.2) [web 5]. Des méthodologies de conception implémentant ces étapes sont utilisées pour la conception de produits complexes. Pour la conception de produits mécatroniques le cycle en V et la conception systématique sont très souvent utilisés. Nous donnons dans la Section 2.2.7 le positionnement de nos travaux et les approches sur lesquelles ils s'appuient.

2.2.7. Positionnement de nos contributions

Durant ces travaux, le développement de nouveaux produits est divisé en 3 parties : la conception, la modélisation et l'ingénierie de performances comportementales. Les travaux concernant la partie conception sont décrits dans le Chapitre 3. Dans ce chapitre, nous proposons une méthodologie de conception sémantique conceptuelle de familles de produits mécatroniques. Elle permet d'aller du cahier des charges à la description des différentes classes de solutions (familles de produits) susceptibles de satisfaire les exigences en respectant les contraintes. Elle s'appuie sur les méthodologies telles que le cycle en V et l'approche de conception systématique auxquels elles apportent des améliorations. Elle correspond à la première étape du processus de conception qui est l'établissement de la comptabilité fonctionnelle et physique du produit avec les besoins et les contraintes et aux deux premières étapes du processus d'écrit dans [Pahl, 1996] qui est l'élaboration du cahier des charges et la formalisation et la spécification des principes.

2.3. Modélisation de produits complexes

La modélisation d'un produit en conception a pour avantage principal de permettre aux fabricants de satisfaire les exigences tout en diminuant le coût et la durée de fabrication. Elle consiste à le représenter par un modèle qui prend en compte ses caractéristiques physiques, fonctionnelles, comportementales, etc. dans le but d'étudier son fonctionnement par simulation. Modéliser permet d'appréhender le réel et de proposer, après la formulation d'hypothèses, sa représentation graphique, symbolique ou équationnelle, pour comprendre son fonctionnement, sa structure et son comportement. Nous allons d'abord donner une réponse à la question "qu'est-ce que la modélisation ?" dans la Section 2.3.1 avant de parler, dans la Section 2.3.3, de quelques types d'analyse permettant de concevoir le modèle d'un produit complexe.

2.3.1. Qu'est-ce que la modélisation ?

Plusieurs définitions sont proposées dans la littérature pour répondre à cette question. Chacune d'elles met plus l'accent sur un aspect plutôt qu'un autre. Ainsi, un modèle est défini comme étant :
« ***Une représentation abstraite et simplifiée (i.e. qui exclut certains détails), d'une entité (phénomène, processus, système, etc.) du monde réel en vue de le décrire, de l'expliquer ou de le prévoir*** » [Audibert].

Cette définition met plus l'accent sur la représentation abstraite et simplifiée d'un objet à étudier et sur l'objectif d'un modèle. Une deuxième définition considère un modèle comme étant :
« ***Une représentation théorique qui ne prétend pas décrire fidèlement l'objet d'étude, mais qui revendique au contraire son caractère délibérément schématique, en même temps que sa fécondité eu égard à un objectif spécifié*** » [Soler, 2000].

Cette définition met l'accent sur la possibilité de représenter schématiquement un objet, mais en faisant abstraction de certains de ces caractéristiques. Elle donne, cependant, plus de précisions que la première en ce sens qu'elle précise que l'abstraction doit se faire selon les objectifs visés. Donc le même objet peut être représenté par plusieurs modèles différents selon l'objectif à atteindre. Une troisième définition considère un modèle comme étant :
« ***Une représentation physique de l'interaction téléologique, temporelle et récursive existant entre un sujet et un objet, où l'objet est un concept ou une perception*** » [Eriksson, 2003].

Dans cette définition, l'accent est surtout mis sur l'interaction entre le sujet (celui qui modélise) et l'objet que l'on modélise. Cependant, l'auteur parle d'interaction téléologique, temporelle et récursive donc l'intervention du sujet sur le modèle de l'objet selon le temps et les objectifs avec possibilité de faire des boucles d'optimisation. Une quatrième définition plus complète car représentant une sorte de synthèse de trois précédentes dit :

« ***Un modèle est une représentation d'un objet d'étude, sous une forme donnée, issue de son abstraction, et satisfaisant un besoin associé à son utilisation*** » [Vernat, 2004].

Dans sa définition, l'auteur aborde 4 aspects importants de la modélisation à savoir l'objet d'étude, la possibilité d'avoir plusieurs types de modèle et l'abstraction selon les objectifs.

A partir de ces quatre définitions, nous pouvons tirer une définition de la modélisation ainsi formulée :

« ***La modélisation est une représentation théorique ou physique, abstraite et plus ou moins fidèle d'un objet du monde réel avec un outil de modélisation pour résoudre un problème posé*** »

Il est alors nécessaire de faire un compromis selon le problème à résoudre. Après avoir défini la modélisation, nous parlons de quelques types d'analyse de produits complexes dans la Section 2.3.2.

2.3.2. Analyse de produits complexes en conception

L'analyse consiste à faire une étude fonctionnelle, structurelle et comportementale d'un système, dans le but de comprendre son fonctionnement de choisir son architecture. Nous allons ainsi parler, dans cette section, de 3 types d'analyse très utilisées en conception de produits complexes : l'analyse fonctionnelle, l'analyse structurelle et l'analyse comportementale.

2.3.2.1. L'analyse fonctionnelle

Elle permet de décrire les modes de fonctionnement du système étudié et de ses sous-systèmes identifiés ainsi que la connaissance de ses fonctions externes et internes à partir du profil de mission [Hammouda, 2013]. Elle établit de façon systématique et exhaustive les relations fonctionnelles à l'intérieur et à l'extérieur de ce système. En d'autres termes, l'analyse fonctionnelle consiste à rechercher et à caractériser les fonctions offertes par un système pour satisfaire les besoins de son utilisateur [Demri, 2009]. Elle a pour objectif de décomposer le produit pour y distinguer :

- les fonctions de service qui permettent de répondre au besoin des clients [Menand, 2002] ;
- les fonctions techniques qui permettent d'assurer les fonctions de service [Menand, 2002] ;
- les fonctions de contraintes qui sont intrinsèques au cycle de vie du produit à l'exception de la phase d'usage [Menand, 2002]. Elles permettent de gérer l'organisation de l'ensemble des fonctions, c'est-à-dire le recensement de leur interaction.

Russély et al. considèrent que l'analyse fonctionnelle du produit permet, à partir des fonctions de service donc du CdCF (Cahier des Charges Fonctionnel), de trouver les fonctions techniques, puis de rechercher des solutions qui permettront les choix de composants [Russéry, 2007].

L'analyse fonctionnelle est définie par la norme NF X50-150, comme étant :

« ***Une démarche qui consiste à rechercher, caractériser, ordonner, hiérarchiser et/ou valoriser les***

fonctions » [NF X50-150].

L'analyse fonctionnelle permet de construire l'architecture fonctionnelle du système à faire qui est définie comme étant un :

« Agencement cohérent de fonctions et de flux fonctionnels nécessaires pour remplir la mission opérationnelle du système à faire » [Lo, 2013].

Il existe différentes méthodes d'analyse fonctionnelle telles que le SADT, le SA-RT, le digramme FAST, etc.

2.3.2.1.1. La méthode SADT

La méthode *Structured and Analysis Design Technique* (SADT) est une méthode d'analyse et de conception de systèmes développée en 1977 par Douglas T. Ross [Demri, 2009]. Elle est introduite en Europe à partir de 1982 par Michel Galiner. C'est un langage pluridisciplinaire, qui cherche à favoriser la communication entre les utilisateurs et les concepteurs. Elle permet de réaliser la description d'un système technique de façon structurée et hiérarchisée en s'appuyant sur une représentation graphique qui met en évidence l'organisation fonctionnelle et structurelle du système en allant du plus général au plus détaillé. Elle fait partie de la famille des méthodes d'analyse fonctionnelle descendante. Le modèle SADT d'un système porte sur ses actions (actigrammes), et sur les données qu'il doit traiter (datagrammes) dans une structure arborescente. Dans le cadre de l'analyse fonctionnelle où on privilégie les activités, ce sont les actigrammes (boîte représentant une activité et transformant des flux d'entrées en flux de sorite) qui sont utilisés (Figure 2.8).

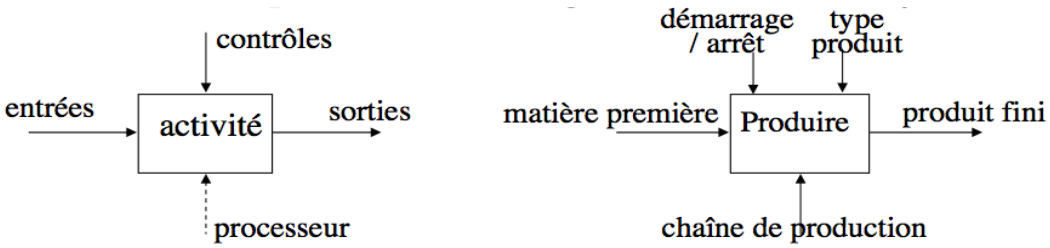

Figure 2.8 : Représentation d'une fonction en SADT [Pollet]

La méthode SADT permet de modéliser un système existant ou futur pour en comprendre le fonctionnement et envisager des solutions. Elle ne permet, cependant, pas de représenter l'aspect dynamique de systèmes tels que les systèmes mécatroniques. D'autres méthodes permettent de combler cette lacune.

2.3.2.1.2. La méthode SA-RT

La méthode *Structured Analysis – Real Time* (SA-RT) a été mise au point à la fin des années 80 pour exprimer les spécifications des applications temps réel [Cottet, 2005]. Elle est surtout utilisée pour la modélisation des logiciels temps réels. C'est une méthode d'analyse fonctionnelle et

opérationnelle qui permet de réaliser une description graphique et textuelle de l'application en termes de besoins [Demri, 2009]. Elle intègre trois aspects fondamentaux d'une méthode de spécification [Cottet, 2005] :

- L'aspect fonctionnel (ou transformation de données) : représentation de la transformation que le système opère sur les données et spécification des processus qui transforment les données ;
- L'aspect évènementiel (piloté par les évènements) : représentation des évènements qui conditionnent l'évolution d'un système et spécification de la logique de contrôle qui produit des actions et des évènements en fonction d'évènements en entrée et fait changer le système d'état ;
- L'aspect informationnel (données) : spécification des données sur les flots ou dans les stockages. Ce dernier aspect, qui est en général assez négligé dans ce type d'application, doit faire l'objet d'une description spécifique.

La méthode SA-RT comprend trois diagrammes représentant le modèle d'un produit : le diagramme de contexte qui définit le contexte et l'environnement du système, le diagramme préliminaire qui est une décomposition du processus fonctionnel initial et ne contient qu'un seul processus de contrôle et le diagramme d'état/transition qui explique le fonctionnement du processus de contrôle [Demri, 2009]. SA-RT prend en compte l'aspect dynamique du système analysé. Elle est fondée sur la méthode Structured Analysis (SA) [Levitt, 2000] et est bien adaptée aux applications à fort comportement dynamique et est plutôt orientée partage du travail et IHM.

2.3.2.1.3. Le diagramme FAST

La méthode *Function Analysis System Technique* (FAST), introduite par l'américain Charles W. Bithenay, complète la boîte à outils d'analyse fonctionnelle [Demri, 2009]. Elle permet de faire l'analyse fonctionnelle d'un produit existant [Caillaud, 2011]. Elle a pour objectif d'ordonner les fonctions, de vérifier la logique et l'exhaustivité. Elle sert, également, de support à la démarche de recherche de solutions. C'est un instrument graphique de communication entre les intervenants qui porte sur les relations entre produits et fonctions. Cet instrument permet de représenter la logique des relations entre les fonctions par répétition de « pourquoi/comment/quand » posés à chaque étape de l'analyse comme le montre la Figure 2.9.

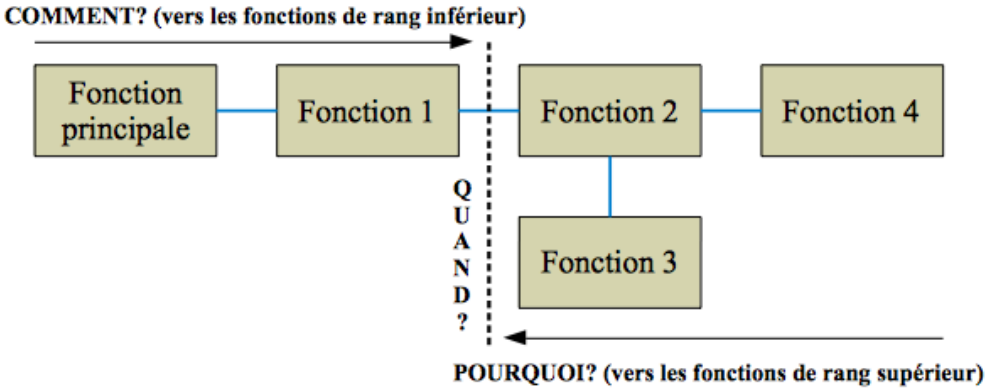

Figure 2.9 : Principe de la méthode FAST [Demri, 2009]

La lecture d'un diagramme FAST se fait de la gauche vers la droit (réponse à la question « Comment ? » et de haut en bas (réponse à la question « Quand ? ») [Russély, 2007]. Comme pour les fonctions de service, les fonctions techniques sont formulées par un verbe à l'infinitif suivi d'un complément. Pour faciliter la lecture, on peut les faire précéder par « *Le produit doit* ».

La méthode FAST présente les relations existant entre produits et fonctions à l'intérieur d'un domaine strictement délimité. Elle permet de relier les diverses solutions techniques qui pourraient convenir pour répondre aux besoins. Elle produit un diagramme qui permet aux concepteurs d'expliquer et de justifier ces solutions techniques.

2.3.2.2. L'analyse structurelle

L'analyse structurelle a pour objet l'étude des solutions constructives qui réalisent les fonctions techniques. Elle permet de décrire la structure physique (l'architecture) d'un produit en conception avec l'ensemble des organes et composants et leurs interfaces pour répondre aux fonctions techniques attendues [Boujut, 1993]. L'architecture d'un système est décrite comme étant :

« ***Le schéma d'aménagement des composants technologiques ou des sous-ensembles technologiques dans le cadre de la conception d'un produit complexe*** » [Mekhilef, 1998].

L'architecture physique est définie par Lô comme un :

« ***Agencement cohérent de composants, de liens et d'interfaces physiques (matérielles / logicielles), organisationnelles et logiques pour remplir la mission opérationnelle du système à faire*** » [Lo, 2013].

La Figure 2.10 montre les étapes permettant de passer des besoins aux solutions techniques en passant par l'analyse fonctionnelle et l'analyse structurelle.

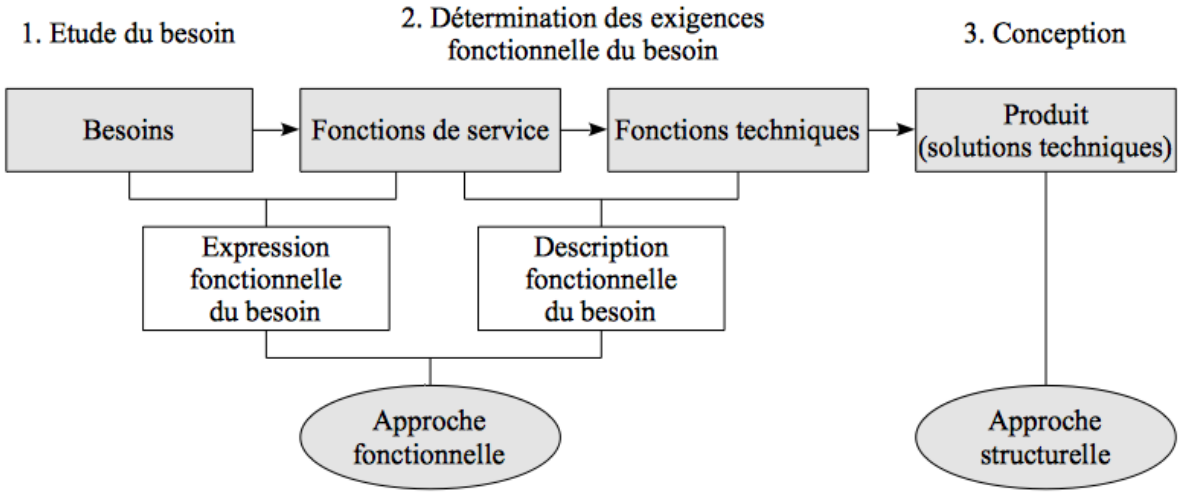

Figure 2.10 : Des besoins aux solutions techniques [Bounie][CI1]

A l'issu de l'analyse structurelle, le modèle géométrique, le modèle topologique et le modèle paramétrique du produit peuvent être créés.

2.3.2.2.1. La modélisation géométrique

La modélisation géométrique a commencé à se développer vers les années 1960 dans l'industrie automobile, au moment où les ordinateurs ont proposé des outils permettant de générer des modèles numériques pour la production : de la conception à la fabrication [Dang, 2014]. Elle peut être définie comme étant l'ensemble des outils mathématiques, numériques et informatiques qui combinés permettent de construire un modèle virtuel d'un objet réel [Daniel, 2009]. Elle peut être utilisée pour un objet existant ou en conception et rend l'observation de certaines caractéristiques plus facile. Les simulations se faisant sur des modèles géométriques, la construction de maquette est évitée et le coût de la fabrication devient plus faible. Comme exemples de modeleurs géométriques, nous avons : le modèle BRep, le modèle CSG, le modèle paramétrique, etc.

- **Le CSG** (**C**onstructive **S**olid **G**eometry) **:** Il permet la représentation volumique d'un solide en le représentant comme une combinaison de primitives telles que cylindre, cône, sphère, tore, cube [Pentcheva, 2010][Dang, 2014]. L'objet est défini par une suite d'opérations ensemblistes (union, intersection, complément, soustraction, etc.) sur ces primitives [Laug, 2005][Pentcheva, 2010]. Un objet 3D est donc représenté par une structure arborescente où les primitives sont des feuilles et les opérateurs associent les nœuds intermédiaires [Pentcheva, 2010][Dang, 2014].
- **Le BRep (B**oundary **Rep**resentation) **:** BRep encore appelé représentation par les bords est utilisé comme standard dans les logiciels de CAO. Un modèle BRep décrit une scène par les éléments non volumiques qui composent sa surface [Pentcheva, 2010]. Il permet la représentation surfacique d'un objet qu'il représente par la géométrie et la topologie des

courbes et des surfaces qui recouvrent sa frontière. Chaque surface élémentaire, appelée carreau, est définie par une application d'un domaine paramétrique plan vers l'espace tridimensionnel [Laug, 2005]. Ainsi, pour modéliser un objet, deux parties sont définies : les entités topologiques et les entités géométriques. Les entités géométriques sont des surfaces, courbes et points tandis que les entités topologiques sont des faces, arêtes et sommets [Pentcheva, 2010][Dang, 2014]. La topologie d'un objet est donnée par les ensembles V_i, F_i, A_i, S_i et L_i [Daniel, 2014] :

- ✓ V_i est l'ensemble de volumes de l'objet ;
- ✓ F_i est l'ensemble de faces de l'objet ;
- ✓ A_i est l'ensemble des arêtes de l'objet ;
- ✓ S_i est l'ensemble des sommets de l'objet ;
- ✓ L_i est l'ensemble des liens entre les volumes, les faces, les arêtes et les sommets.

- **La modélisation paramétrique :** L'approche paramétrique est la continuation naturelle des premiers modeleurs (BRep, CSG…) [Mucalet, 2003]. D'un objet géométrique avec une configuration (positions, dimensions, orientations) parfaitement définie, on passe à un objet défini par des paramètres (ou variables). Le paramétrage est dimensionnel et fonctionnel. L'axe de la modélisation paramétrique regroupe les paramètres clés du système et les spécifications (relations mathématiques sur ces paramètres). Il sert de pivot entre les spécifications, l'analyse fonctionnelle et l'analyse structurelle. Il permet de tester différentes variantes et donc de facilement définir des familles de produits.

Le Figure 2.11 donne un récapitulatif de différents types de modèles de produits.

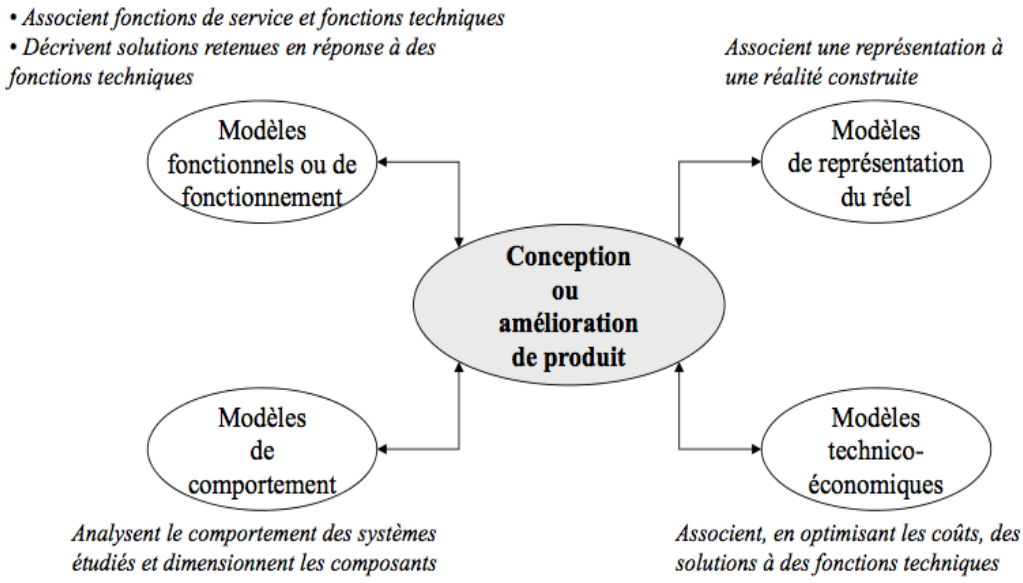

Figure 2.11 : Différents types de modèles d'un système [Bounie]

Le CSG, le BRep et la modélisation paramétrique sont utilisés par les systèmes de CAO pour modéliser des produits. Les débuts véritables de la CAO remontent aux années 60 et consistaient essentiellement à informatiser la planche à dessin traditionnelle (DAO). Dans les années 70 et 80, la DAO a ainsi permis aux entreprises d'entamer le processus d'informatisation de leurs méthodes de dessin 2D. Les années 90 ont vu l'essor des modélisations volumiques et solides à 3 dimensions. On est passé de la DAO à la CAO [Kuate, 2006].

La ***conception assistée par ordinateur*** (CAO) peut être définie comme la modélisation d'un produit avec un ordinateur à l'aide de logiciels spécifiques. Elle englobe également les techniques de modélisation structurelle et de simulation numérique (calcul de résistance, optimisation de matériaux, etc.).

2.3.2.2.2. La modélisation sémantique

La modélisation sémantique permet de décrire et représenter les caractéristiques sémantiques de produits complexes. Le modèle sémantique d'un produit représente les objets et les concepts qu'il manipule, à travers son activité, indépendamment des moyens qu'il met en œuvre [Vauquier, 2007]. En conception de produits complexes, elle peut se baser sur une décomposition structurelle ou fonctionnelle du produit [Coulibaly, 2008a]. En plus des composants du produit et de ses fonctions les liaisons entre ses composants doivent également être prises en compte. Ainsi, la modélisation sémantique de produits complexes englobe la représentation des :

- données sémantiques des composants [Coulibaly, 2008a] ;
- données sémantiques des liaisons entre composants [Coulibaly, 2008a];
- fonctions et modules que les composants du produit doivent exécuter.

Pour chaque type de modèle plusieurs outils peuvent être utilisés pour sa conception. Nous présentons dans la Section 2.3.3 quelques outils de modélisation de produits complexes.

2.3.3. Quelques outils de modélisation

Plusieurs outils permettent de modéliser des produits complexes. Nous allons donner brièvement quelques-uns de ces outils que nous utiliserons dans la suite de nos travaux.

2.3.3.1. Le langage SysML

Le langage SysML (System Modeling Language), dérivé de UML (Unified Modeling Language), a pour objectif de modéliser les systèmes complexes [Roques, 2013]. Ce langage permet non seulement de modéliser la structure de ces systèmes (diagramme de définition de bloc, de bloc interne, paramétrique, de package), mais aussi leurs comportements (diagramme de cas d'utilisation, d'état,

de séquence, d'activité) et leurs contraintes (diagramme d'exigences) [OMG SysML, 2006]. Une particularité de SysML est de regrouper trois familles de représentations (fonctionnelle, structurelle, comportementale) au sein d'un unique modèle *"multipoint de vue"*. Une telle approche est dite **basée sur un modèle**, par opposition aux approches **basées sur des documents** qui reposent sur une collection de modèles *"mono-point de vue"* disjoints.

SysML compte 9 diagrammes dont certains sont identiques à ceux d'UML 2 (diagramme de cas d'utilisation, diagramme de séquence, diagramme d'état et diagramme de package), d'autres sont des diagrammes d'UML 2 modifiés (diagramme de définition de blocs, diagramme de bloc interne et diagramme d'activité) et d'autres sont de nouveaux diagrammes (diagramme d'exigences et diagramme paramétrique) comme le montre la Figure 2.12.

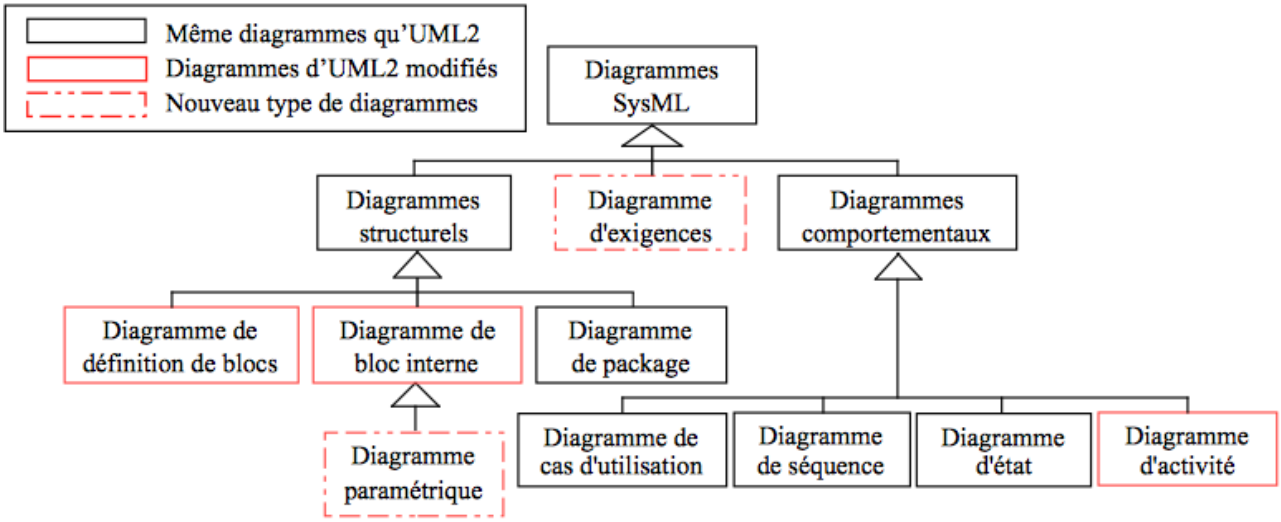

Figure 2.12 : Les 9 diagrammes de SysML [OMG SysML, 2006]

2.3.3.2. Les DSM

Les DSM (**D**esign **S**tructure **M**atrix) ont été créés par Steward en 1981 [Turki, 2008]. A l'origine, les DSM sont des matrices carrées dont chaque entité du système modélisé est représentée par une ligne et une colonne. Les cellules non diagonales permettent de représenter les relations entre les entités. Ainsi, si les entités E_i et E_j ont une relation entre eux, cette relation est représentée dans la cellule Cel_{ij}. Les cellules diagonales ne jouaient alors aucun rôle [Turki, 2008]. Elles sont généralement utilisées en Ingénierie Système et en Management de projets pour modéliser la structure des systèmes complexes et des processus. Pour les premiers, elles sont communément utilisées pour identifier et analyser les dépendances et interaction entre les entités d'un système [Verries, 2010]. Pour les seconds, elles servent à planifier l'exécution des différentes tâches et à organiser le déroulement d'un projet [web 10]. Il existe deux grandes familles de DSM : les DSM statiques et les DSM temporelles ou dynamiques [Harmel, 2007][Hong, 2009].

2.3.3.2.1. Les DSM statiques

Les DSM statiques permettent de représenter les relations entre les éléments d'un système en faisant abstraction du temps. Les DSMs les plus utilisées parmi celles-ci sont : les DSMs basées sur les composants et les DSMs basées sur les équipes de travail [Harmel, 2007][Hong, 2009].

L'objectif principal des DSMs basées sur les composants est de représenter l'architecture des systèmes multi-composants. Elles permettent de représenter les relations entre les entités qui sont les objets dans le système. Ces DSM sont binaires et permettent de représenter l'existence ou non d'un type d'interaction étudiée [Harmel, 2007] (Figure 2.13). Une représentation des relations entre les différentes entités ou composants du produit sous forme de vecteur de dimension 4 dont chaque élément est binaire pour différencier les types de relations est proposée dans [Harmel, 2007].

Les DSMs basés sur les équipes de concepteurs permettent d'assurer l'interface entre les équipes. Elles sont utilisées dans l'organisation de la conception et des projets multi-équipes [Sharon, 2009]. Ces DSM permettent également de représenter les interactions entre les acteurs dans une équipe de conception [Harmel, 2007].

Composants	C_1	C_2	- -	- -	C_n
C_1		1	0	0	1
C_2			0	0	1
- -				0	0
- -					0
C_n					

Figure 2.13 : Matrice basée sur les composants

2.3.3.2.2. Les DSM temporelles

Les DSM temporelles permettent de représenter les systèmes dont les éléments sont liés par des relations de précédence, qui peuvent être de type temporel (DSM Processus) ou des relations de hiérarchie ou de contrainte (DSM Paramètres) [Harmel, 2007]. Trois types de DSMs temporelles ont été identifiés [Browning, 2001] : les DSM basées paramètres, les DSM basées sur les fonctions, les DSM basées processus.

Les DSM basées sur les Paramètres sont utilisées pour représenter les relations de précédence et les contraintes entre les paramètres et les variables en conception. Elles permettent d'optimiser les boucles de conception entre groupes de paramètres [Harmel, 2007][Deniaud, 2011].

Les DSM basées sur les Fonctions permettent de rapprocher les fonctions techniques qui ont des échanges importants de flux de matières, d'énergie ou d'informations [Deniaud, 2011].

Les DSM basées sur les Processus permettent de modéliser les précédences entre tâches au sein d'un processus et de minimiser les boucles de retours-arrières dans le projet [Deniaud, 2011]. Elles

expriment, ainsi, les interdépendances entre les tâches d'un processus [Turki, 2008].

Partant de cette classification, nous pouvons alors résumer les différents types de DSM comme le montre la Figure 2.14.

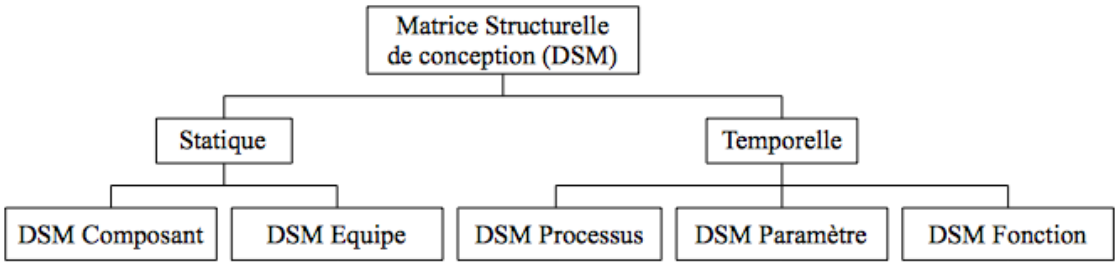

Figure 2.14 : Différents types de DSM [Browning, 2001]

Parmi ces DSM certaines décrivent le produit à concevoir (système à faire), d'autre le processus permettant de le faire (système pour faire). La Tableau 2.2 montre cette différenciation des types de DSM.

Tableau 2.2 : Typologie des DSM [Deniaud, 2011]

Vue	**Domaine**	
	Système à faire : le produit	**Système pour faire :** l'organisation
Statique	DSM Composant	DSM Equipe
Temporelle	DSM Paramètre DSM Fonction	DSM Processus

Les DSM basées sur les composants sont très souvent utilisés par les concepteurs de produits complexes. Elles sont utilisées pour représenter l'architecture d'un produit. Ces DSM ont été améliorées par Coulibaly et al. qui y ajoutent la possibilité de représenter les caractéristiques des composants dans les cellules diagonales et le type de liaison dans les cellules non diagonales (Figure 2.15) [Coulibaly, 2007][Coulibaly, 2008a][Coulibaly, 2010]. En plus de la représentation des caractéristiques des composants et des types de liaisons, d'autres informations sémantiques telles que la criticité, la fiabilité, l'impact environnemental des composants sont également ajoutées sous forme de colonnes. Le nombre de liaisons que chaque composant partage avec d'autres est représenté sur une ligne supplémentaire. Ces matrices sont nommées X-DSM (matrices sémantiques ou DSM étendue).

Composants	C_1	C_2	--	--	C_n	Criticité	Fiabilité	--
C_1	C_{11}	V_{12}			V_{1n}	K_1	R_1	--
C_2		C_{22}			V_{2n}	K_2	R_2	--
--			--			--	--	--
--				--		--	--	--
C_n					C_{nn}	K_n	R_n	--
Nb Liaisons	d_1	d_2	--	--	d_n	$K_{0(Seuil)}$	$R_{0(Seuil)}$	--

Figure 2.15 : Matrice sémantique

2.3.3.3. Les réseaux de Pétri

Plusieurs outils de modélisation tels que les graphes, les réseaux de Pétri, etc. sont souvent utilisés pour la recherche de chemins de désassemblage. Les graphes représentent un outil mathématique assez puissant pour la recherche de chemin. Cependant, avec l'objectif de prendre en compte les données sémantiques des produits et de prioriser des chemins de désassemblage sur d'autres les réseaux de Pétri donne plus de possibilités. En effet, dans un réseau de Pétri il est possible de pondérer les places, les transitions mais aussi les arcs reliant les places et les transitions. A cet effet, ils constituent un outil très utile pour la prise en compte des données sémantiques des composants et liaisons. Dans nos travaux, nous utilisons les réseaux de Pétri pour modéliser les produits mécatroniques dans le but de rechercher le chemin optimal de démontage de composants défaillants. Ainsi, nous donnons dans cette section la définition d'un réseau de Pétri stochastique généralisé qui sera utilisé dans le Chapitre 6.

Les réseaux de Pétri (RdP) ont été inventés en 1962 par Carl Adam Pétri lors de sa thèse [Da Silveira, 2003]. C'est un outil mathématique et graphique très utile pour la modélisation de systèmes concurrents, asynchrones, distribués, parallèles, non déterministes, ou stochastiques. Ils permettent de modéliser l'aspect dynamique de tels systèmes. Ils permettent également d'évaluer les performances des systèmes concurrents en terme de relations de type cause à effet. Au début, les RdP ordinaires faisaient abstraction du temps. Ils ont évolué avec l'introduction du temps (SPN ***:*** ***S****tochastic* ***P****etri* ***N****et*), puis l'ajout de transitions immédiates (GSPN ***:*** ***G****eneral* ***S****tochastic* ***P****étri* ***N****et*) et l'introduction de couleurs associées aux marques et des gardes associées aux transitions (SWN ***:*** ***S****tochastic* ***W****ell formed Petri* ***N****ets*) [Diagne, 2007]. Un RdP ordinaire est constitué de 4 éléments de base (jeton ou marque, place, transition et arc orienté) [Moore, 2001] (Figure 2.16).

Figure 2.16 : Eléments d'un RdP ordinaire : a) jeton ; b) place ; c) transition ; d) arc

Un GSPN est caractérisé par (P, T, Pré, Post, Inh, M_0, W) [Diagne, 2007] avec :

- **P** l'ensemble des places ;
- **T** l'ensemble des transitions qui peuvent être immédiates ou temporisées ;
- **Pré :** P x T → N^+ est l'application d'incidence avant telle que si un arc de poids n ≥ 1 relie P_i à T_j alors Pré(P_i, T_j) = n, sinon Pré(P_i, T_j) = 0 ;
- **Post :** P x T → N^+ est l'application d'incidence arrière telle que si un arc de poids m ≥ 1 relie T_j à P_i alors Post(P_i, T_j) = m, sinon Post(P_i, T_j) = 0 ;
- **Inh :** P x T → N est l'application des arcs inhibiteurs ;
- **M_0** est le marquage initial ;
- **W :** T → N est l'application qui à chaque transition temporisée associe un délai de franchissement et à chaque transition immédiate associe une probabilité de franchissement.

2.3.4. Normes pour l'échange d'informations de produits et composants

2.3.4.1. La norme STEP (ISO 10303)

Pour permettre l'échange de données entre concepteurs travaillant sur un même projet, mais utilisant des systèmes différents, l'ISO a standardisé une norme internationale nommée STEP (**ST**andard for **E**xchange of **P**roduct Model Data) [ISO, 10303-1:1994]. Elle est développée au sein du groupe ISO/TC184/SC4 [Chambolle, 1999]. Cette norme est issue de celles qui l'ont précédée (SET, VDA, IGES) et ne faisaient pas l'objet de normalisation ISO. Elle représente une unification de ces normes et les complète. Pour harmoniser les modèles des différents concepteurs et systèmes, un langage de modélisation est créé dans STEP, normalisé ISO 10303:11 et appelé Express.

- La norme ***SET*** (**S**pécification du standard d'**E**change et de **T**ransfert) [SET, 1989] est une norme française de l'*AfnorZ 68-300*, publiée en norme expérimentale en août ***1985*** et homologuée en décembre ***1993*** [Villard, 2007]. Elle gérait l'échange d'informations entre différents systèmes CAO, et permettait aux différents intervenants, constructeurs, équipementiers, sous-traitants et fournisseurs, de pouvoir se comprendre.
- La norme ***VDA-FS*** (**V**erband **D**er **A**utomobilinsdustrie **F**lächen **S**chnittstelle) [VDA, 1986] est le format de l'industrie automobile allemande pour les échanges de surface. C'est un standard de qualité et d'échange initiée par le ***VDA*** (**V**erband **D**er **A**utomobilinsdustrie ou encore union de l'industrie automobile). C'est un format qui ne permet de transmettre que de la géométrie, la couleur, le texte et les autres ne sont pas pris en compte.
- La norme ***IGES*** (**I**nitial **G**raphics **E**xchange **S**pecification) [IDES, 1980] est un format d'exportation de données graphiques. Elle permet aux systèmes CAO d'échanger des

informations graphiques. Elle est lancée en janvier ***1980*** par le laboratoire américain ***NBS*** (**N**ational **B**ureau of **S**tandards). C'est un type de format d'export présentant une universalité assez forte dans le domaine des données CAO 3D surfaciques et filaires.

Ces normes ont montré leurs limites au cours de leur utilisation. C'est ainsi qu'est initié ***STEP*** qui est une unification de ces normes. Lancée au début des années ***1990*** elle est normalisée par ISO en 1994 [ISO, 10303-1:1994]. ***STEP*** est donc une norme internationale contrairement à ces devancières. Les objectifs essentiels de cette norme sont :

- La définition d'un format neutre, interprétable par tous les systèmes informatiques indépendamment du système particulier utilisé pour gérer les données ;
- La couverture d'un très vaste domaine de connaissance, correspondant à l'ensemble des catégories de produits (pièces élémentaires, assemblages, mécanismes, etc.) et à tous les métiers (électronique, mécanique, ingénierie, etc.) ainsi que toutes les phases du cycle de vie (conception, analyse, fabrication, maintenance, démantèlement).

Pour y arriver, il était nécessaire de faire en sorte que les données soient indépendantes des systèmes. Un langage formel de spécification des données appelé ***EXPRESS*** [ISO, 10303-11:1994] [Chambolle, 1999] et des formats neutres d'échange et de stockage des données décrites dans ce langage sont alors définis.

2.3.4.2. La norme PLib (ISO 13589)

Avant de parler de la norme PLib (**P**art **Lib**rary), dans la Section 2.3.4.2.2, nous commençons par définir la notion d'ontologie, dans la Section 2.3.4.2.1, qui est utilisée par cette norme pour décrire les composants et harmoniser la structure de leur description.

2.3.4.2.1. La notion d'ontologie

Plusieurs définitions ont été données pour décrire une ontologie dans le domaine de l'informatique ou plus généralement dans les sciences de l'information. Ainsi une ontologie est décrite comme étant :

« ***Un ensemble structuré des termes et concepts représentant le sens d'un champ d'informations, que ce soit par les métadonnées d'un espace de noms, ou les éléments d'un domaine de connaissances*** » [web 11].

« ***Un ensemble d'outils qui permettent de représenter précisément un corpus de connaissances sous une forme utilisable par une machine*** » [web 12].

« ***Une spécification formelle et explicite d'une conceptualisation faisant l'objet d'un consensus*** » [Gruber, 1993].

« ***La définition d'un vocabulaire commun pour les chercheurs qui ont besoin de partager***

l'information dans un domaine. Elle inclut des définitions lisibles en machine des concepts de base de ce domaine et de leurs relations » [Noy, 2000].

D'après ces définitions, nous pouvons dire que :

« ***Une ontologie est une représentation partagée et consensuelle de la connaissance sur un domaine en vue de son automatisation*** ».

La norme PLib utilise les ontologies de domaine pour modéliser, structurer, référencer les données de composants et de produit pour faciliter leur stockage et leur échange entre fournisseurs et concepteurs ou entre différents concepteurs. Une ontologie de domaine est fonctionnelle et orientée objet. Elle est utilisée pour représenter un domaine sous forme de base de connaissances. Elle présente les concepts clés du domaine : objets, attributs, instances,... et aussi les relations entre ces différents concepts.

2.3.4.2.2. La norme PLib

Pour concevoir de nouveaux produits, le concepteur utilise souvent des produits fabriqués par des entreprises du domaine. Ces produits qui entrent dans la fabrication d'un autre produit sont appelés composants. Avec l'avènement des produits complexes, le nombre de composants utilisés pour concevoir un produit est important. Ces composants sont souvent fabriqués par des entreprises différentes utilisant des systèmes différents. Pour le stockage et la structuration des données décrivant ces composants, le concepteur aura donc à manipuler des informations venant de divers horizons avec des formats non identiques. Pour fédérer ces données, il est donc nécessaire de trouver le moyen d'harmoniser leur format sans perte d'informations. En outre, il peut arriver que plusieurs composants jouant le même rôle dans le fonctionnement d'un produit mais avec des caractéristiques différents (structure, forme géométrique, dimensions, puissance, ...) puissent être utilisés pour sa conception. Le concepteur doit donc faire son choix et sélectionner des composants selon ses objectifs, les critères fixés et les performances à atteindre. C'est ainsi que le développement d'une nouvelle norme est initié d'abord en 1987 au niveau européen [Pierra, 1989] puis au niveau ISO depuis 1990. Cette norme est nommée PLib (Part Library) et est standardisée ISO 13584 [ISO, 13584-20:1998][ISO, 13584-24:1998][ISO, 13584-42:1998]. Elle est développée par le groupe ISO/TC184/SC4 et traite des librairies de composants [Chambolle, 1999].

L'objectif principal de cette norme est de donner aux concepteurs un formalisme neutre et universel de structuration, de stockage et d'échange des données de composants et de produits. Elle permet l'échange et le référencement de catalogues informatisés de composants et objets techniques préexistants [Pierra, 1994]. Elle permet également d'associer à un objet catalogue un nombre quelconque de représentations, propre à chacune des disciplines qui manipulent l'objet (une vie solide,

un modèle de comportement pour la simulation) [Pierra, 2002]. Elle permet enfin d'intégrer dans un environnement homogène et cohérent des bibliothèques fournies par différentes sources [Pierra, 1997].

2.3.5. Synthèse et analyse

La modélisation est une représentation théorique ou physique, abstraite et qui se veut le plus fidèle possible d'un objet du monde réel. Elle est faite avec un outil de modélisation dans le but de représenter le point de vue fonctionnel et/ou structurel et/ou comportemental pour résoudre un problème posé. Plusieurs méthodologies et outils de modélisation sont utilisés pour modéliser des produits complexes. Pour la modélisation de produits mécatroniques, les outils et langages orientés-objet (SysML, Express, etc.) ainsi que la modélisation géométrique (CAO) sont très utilisés. Des normes internationales (STEP, PLib) sont également définies pour permettre l'échange de données de produits ou de composants. Nous donnons dans la Section 2.3.6 le positionnement de nos travaux ainsi que les outils et méthodologies utilisés.

2.3.6. Positionnement de nos contributions

La modélisation des classes de solutions issues de la CSC est notre deuxième étape dans le processus de développement de produits mécatroniques. Ainsi, nous proposons dans le Chapitre 4, une méthodologie de MSC de produits mécatroniques qui se fait en 2 phases : la modélisation structurelle et la modélisation fonctionnelle. Ces deux types de modélisation utilisent le langage SysML et les DSM auxquels nous avons apportés des améliorations pour pouvoir modéliser des familles de produits mécatroniques. Nous posons également, dans le Chapitre 5, les bases d'une nouvelle génération de CAO appelée CAO sémantiques conceptuels (CAO-SC). Ce type de CAO permettra de prendre en compte la modélisation de familles de produits mécatroniques. Cependant, son architecture et son implémentation sont prévues dans nos futurs travaux. La MSC se positionne au niveau des deux dernières étapes du processus décrit dans [Pahl, 1996] que sont la conception d'ensemble et la conception détaillée.

2.4. Evaluation de performances comportementales

Avec l'objectif principal de fournir des produits innovants, performants, répondant aux exigences des utilisateurs et à moindre coût, les domaines comportementaux tels que la maintenabilité, la fiabilité, la sécurité, la disponibilité, la recyclabilité etc. doivent être évalués en phase de conception préliminaire. Leur évaluation durant cette phase permet également de bien documenter le produit à l'intention de ses futurs utilisateurs. Cette documentation fournira entre autre le temps qu'il faut pour

remettre le produit en état de fonctionnement en cas de panne, la durée de vie des composants, les composants recyclables, la probabilité que le produit soit en état de fonctionnement en cas de besoin...

Nous parlons dans cette section de travaux portant sur l'évaluation des performances comportementales de produits complexes en phase de conception. Ainsi, nous faisons un tour d'horizon d'indicateurs de performances et de formules proposés avant de parler de quelques méthodologies utilisées pour évaluer les performances comportementales de produits complexes. Nous finissons cette partie en faisant une synthèse et en donnant le positionnement de nos travaux par rapport à l'existant.

2.4.1. Le FMDS

Pour tout produit manufacturier, les utilisateurs souhaitent le trouver en bon état de fonctionnement (***disponible***) en cas de besoin et capable d'accomplir la fonction pour laquelle il est sollicité (***fiable***). Ils veulent également pouvoir s'en servir sans risque de blessure, ou de détérioration de leur santé dans le court, moyen ou long terme (***sûr***). Un produit doit pouvoir être remis en bon état de fonctionnement avec un minimum de perte de temps lorsqu'une défaillance survient (***maintenable***). Ces quatre caractéristiques sont appelées FMDS (Fiabilité, Maintenabilité, Disponibilité, Sécurité) [CIMI, 2011]. Il faut cependant, noter que depuis un certain nombre d'années, avec le réchauffement climatique et le concept d'éco-conception [Diehl, 2005][Gaye, 2011], l'étude de la **recyclabilité** des produits manufacturiers est de plus en plus abordée dans les phases préliminaires de conception.

Les temps les plus utilisés en FMDS sont [Castaneda, 2009] (Figure 2.17) :

- **MTTF** (**M**ean **T**ime **T**o **F**ailure) : durée moyenne de fonctionnement avant défaillance, espérance mathématique de la durée de fonctionnement avant défaillance ;
- **MTBF** (**M**ean **T**ime **B**etween **F**ailures) durée moyenne entre deux défaillances consécutives d'une entité réparée ;
- **MTTR** (**M**ean **T**ime **T**o **R**epair) durée moyenne de panne ou moyenne des temps pour la remise en état de fonctionnement, espérance mathématique de la durée de panne ;
- **MUT** (**M**ean **U**p **T**ime) ou TMD temps moyen de disponibilité, espérance mathématique de la durée de disponibilité ;
- **MDT** (**M**ean **D**own **T**ime) ou TMI temps moyen d'indisponibilité, espérance mathématique de la durée d'indisponibilité.

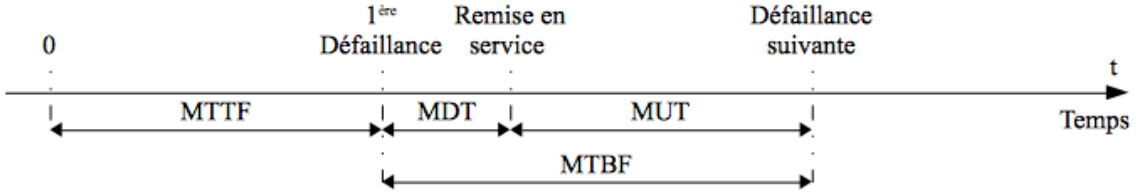

Figure 2.17 : Chronologie des temps calculés en sûreté de fonctionnement

En plus de ces paramètres tels que la fiabilité, la disponibilité, la maintenabilité et sécurité, nous parlons, dans cette section, de la recyclabilité qui est de plus en plus étudiée dès la phase de conception de produits manufacturiers.

2.4.1.1. La fiabilité

Elle est définie par la norme NF X 60-010 comme étant :

« ***La caractéristique d'un système, exprimée par la probabilité qu'il accomplisse la fonction pour laquelle il a été conçu, dans des conditions données et pendant une durée donnée*** » [NF X 60-010][Ait-Kadi, 2007].

La norme NF X 60-500 considère, quant à elle, la fiabilité comme étant :

« ***L'aptitude d'un bien à accomplir une fonction requise dans des conditions données pendant un temps donné. Caractéristique d'un bien exprimée par la probabilité qu'il accomplisse une fonction requise dans des conditions et un temps données*** » [NF X 60-500][Khalfoui, 2003]

Elle est caractérisée par la probabilité R(t) que l'entité E accomplisse ses fonctions, dans les conditions données pendant l'intervalle de temps [0, t], sachant que l'entité n'est pas en panne à l'instant 0 [Khalfoui, 2003][Demri, 2009]. L'évaluation de cette probabilité peut être faite différemment selon la nature des entités considérées ou selon les moyens dont on dispose pour le faire [Castaneda, 2009].

Pour calculer la fiabilité R(t) d'un produit à partir des fiabilités $R_i(t)$ de ses n composants avec des pannes indépendantes, la formule suivante est utilisée [Zwingmann, 2005][Coulibaly, 2008b] :

$$R(t) = \prod_{i=1}^{n} R_i(t) \qquad (2.1)$$

2.4.1.2. La Disponibilité

Elle est définie par l'IEC 60050-191 comme étant :

« ***La probabilité pour que l'équipement ou le système utilisé dans les conditions prévues soit en état d'accomplir une fonction requise à un instant donné*** » [IEC 60050-191].

Elle est également définie par la norme NF X 60-000 comme étant :

« ***L'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données, à un instant donné ou pendant un intervalle de temps donné, en supposant que la fourniture des moyens extérieurs nécessaires soit assurée*** » [NF X 60-000]

La disponibilité est caractérisée par la probabilité A(t) que l'entité E soit en état, à l'instant t, d'accomplir les fonctions requises dans des conditions données [Demri, 2009].

Si la MTBF est la moyenne des temps de bon fonctionnement, et la MTTR la moyenne des temps techniques de réparation, on peut alors calculer la disponibilité stationnaire UTR (Up Time Ratio) avec la formule [Coulibaly, 2008a][Menye, 2009] :

$$URT = \frac{MTBF}{MTBF + MTTR} \qquad (2.2)$$

Soit $\bar{M}$ la moyenne combinée des temps actifs de maintenance corrective et préventive et MTBM (Mean Time Between Maintainances) la moyenne des temps entre maintenances, on calcule la disponibilité inhérente A_a avec la formule suivante :

$$A_a = \frac{MTBM}{MTBM + \bar{M}} \qquad (2.3)$$

2.4.1.3. La sécurité

Elle est définie par Demri comme étant :

« ***La sécurité est l'aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques*** » [Demri, 2009].

La NF EN 292-1 définit la sécurité comme étant :

« ***L'aptitude d'une machine à accomplir sa fonction, à être transportée, installée, mise au point, entretenue, démontée et mise au rebut dans les conditions d'utilisation normales spécifiées dans la notice d'instructions, sans causer de lésions ou d'atteinte à la santé*** » [NF EN 292-1, 1991]

Elle est caractérisée par la probabilité S(t) que l'entité E ne laisse pas apparaître dans des conditions données, des événements critiques ou catastrophiques [Demri, 2009].

Soient **FRis** le facteur de risque qui indique s'il y a risque ou non et **IRis** l'index de risque qui permet de qualifier et quantifier le risque. L'indicateur de sécurité I_s est défini par [Coulibaly, 2008b] :

$$I_S = FRis + IRis \qquad (2.4)$$

2.4.1.4. La maintenabilité

« ***Pour une entité donnée, utilisée dans des conditions données, la maintenabilité est la probabilité pour qu'une opération donnée de maintenance active puisse être effectuée pendant un intervalle***

de temps donné, lorsque la maintenance est assurée dans des conditions données et avec l'utilisation de procédures et de moyens prescrits » [IEC 60050-191].

La norme NF X 60-010 (AFNOR (1991)), définit la maintenabilité comme :

« ***l'ensemble des actions permettant de maintenir ou de rétablir un bien dans un état spécifié ou en mesure d'assurer un service déterminé*** » [NF X 60-010].

La norme NF X 60-000, donne la définition suivante à la maintenabilité :

« ***Dans des conditions données d'utilisation, aptitude d'une entité à être maintenue ou rétablie, sur un intervalle de temps donné, dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données, avec des procédures et des moyens prescrits*** » [NF X 60-500].

Elle est caractérisée par la probabilité M(t) que l'entité E soit en état, à l'instant t, d'accomplir ses fonctions, sachant que l'entité était en panne à l'instant 0 [Demri, 2009]. Selon Zwingmann [Zwinmann, 2005], Ményé [Menye, 2009] et Coulibaly [Coulibaly, 2008b] si :

- S_N^k est le chemin optimal pour accéder au composant C_k et N le nombre de composants à enlever pour atteindre C_k ;
- Remove(S_{N-i+1}^k, i) est le temps nécessaire pour enlever le composant C_i après avoir enlevé les i-1 composants ;
- R_{time}^k est le temps nécessaire pour atteindre un composant C_k.

On a :

$$R_{time}^k = \sum_{i=1}^{N} Remove\left(S_{N-i+1}^k,\ i\right) \qquad (2.5)$$

L'indicateur de maintenabilité générale I_M du produit est obtenu avec la formule :

$$I_M = \sum_{k=1}^{n} R_{time}^k \qquad (2.6)$$

La maintenabilité peut également être obtenue la MTTR (Moyenne des Temps Techniques de Réparation) avec la formule :

$$MTTR = T_{Diag} + T_R + T_C \qquad (2.7)$$

Où

- T_{Diag} est la durée de diagnostic du produit ;
- T_R est la durée de réparation du produit ;
- T_C est la durée du contrôle de bon fonctionnement après réparation.

2.4.1.5. La recyclabilité

Le **recyclage** est un procédé de traitement des déchets (déchet industriel ou ordures ménagères)

qui permet de réintroduire, dans le cycle de production d'un produit, des matériaux qui composaient un produit similaire arrivé en fin de vie, ou des résidus de fabrication [web 13].

« ***La recyclabilité en fin de vie d'un produit concerne l'aptitude de ses matériaux constitutifs à être réutilisés comme matière première en vue de son recyclage.*** » [NSX 400-630]

2.4.2. Synthèse et analyse

Plusieurs travaux ont porté ces dernières années sur l'évaluation de performances comportementales de produits complexes. Les domaines comportementaux concernés par cette étude sont la fiabilité, la disponibilité, la sécurité, la maintenabilité et la recyclabiliité. Des métriques permettant de les évaluer en phase préliminaire de conception et des formules permettant de calculer ces métriques sont proposées. De plus, un système d'information et un modèle organisationnel permettant le recyclage des produits manufacturés est proposé par Gaye [Gaye, 2011]. La modélisation par réseaux de Pétri pour évaluer le désassemblage de produits est également très utilisée en phase de conception. L'évaluation de ces domaines comportementaux en phase de conception permet diminuer les temps d'indisponibilité des produits, d'augmenter leur sécurité et la recyclabilité de leurs composants en fin de vie. Ils s'appliquent cependant sur des instances précises de produits complexes.

2.4.3. Positionnement de nos contributions

L'évaluation des performances comportementales des instances de classes de solutions (produits mécatroniques) est la troisième étape de notre processus de développement de produits mécatroniques. C'est ainsi que nous traitons dans le Chapitre 6, l'ingénierie de performances comportementales de produits mécatroniques. Elle permet d'extraire à partir du modèle d'une famille de produits mécatroniques les instances satisfaisant les exigences fonctionnelles, structurelles et comportementales du cahier des charges. Elle se positionne sur les deux dernières étapes du processus de conception décrit dans [web 5] que sont l'équilibrage de l'économie globale de la solution sur toutes les étapes de la vie du produit et la recherche de l'équilibre entre contraintes, performances, coûts, délais et risques.

Conclusion

Dans ce chapitre nous avons d'abord décrit le domaine d'application de notre étude (produits mécatroniques) et donné l'état de la recherche sur les domaines couverts par les travaux de cette thèse : la conception conceptuelle, la modélisation et l'évaluation de performances comportementales en phase de conception. Ensuite, nous avons présenté les outils que nous comptons utilisés pour répondre

aux questions de recherche données dans la problématique. Enfin, pour chaque domaine abordé, nous avons situé les contributions scientifiques apportées durant les travaux de cette thèse.

Ces contributions sont décrites et expliquées dans les chapitres 3 (conception sémantique conceptuelle), 4 (modélisation sémantique conceptuelle), 5 (CAO sémantique conceptuelle) et 6 (ingénierie de performances comportementales). Les fonctionnalités et l'architecture d'un logiciel d'aide à la modélisation sémantique conceptuelle et à l'ingénierie des performances comportementales de produits complexes en phase de conception sont décrits dans le chapitre 7, son implémentation est traitée dans le chapitre 8 et un cas d'application y est traité.

Deuxième partie : Conception de produits complexes

Chapitre 3 : Conception sémantique conceptuelle de produits complexes

Chapitre 4 : Modélisation sémantique conceptuelle de produits complexes

Chapitre 5 : CAO sémantique conceptuelle

Chapitre 3 : Conception sémantique conceptuelle de produits complexes

Chapitre 3

Conception sémantique conceptuelle de produits complexes

Introduction

La conception conceptuelle constitue la première phase dans le processus de développement de produits complexes. Beaucoup de méthodologies ont été proposés ces dernières années. Parmi ces méthodologies nous pouvons cités la conception systémique [Motte, 2008], le VDI 2206 [Gausemeier, 2003][VDI, 2004], le V-Model [Ziemniak, 2009], etc. La combinaison du V-Model avec la conception systémique est aussi proposée par Rahman et al. [Rahman, 2007]. Dans [Eder, 2013], l'auteur propose une amélioration de la conception systémique. Il faut, cependant, noter que ces travaux portent sur tout le processus de conception, modélisation, vérification et validation du produit. La conception sémantique conceptuelle de produits mécatroniques traitée dans ce chapitre est la première phase de la partie descendante du V-Model et la première tâche de la conception systémique. C'est la partie allant du cahier des charges, spécifiant les attentes des utilisateurs, à l'obtention d'un ensemble de solutions satisfaisant les exigences et susceptibles de respecter les contraintes de performances. Elle permet également de prendre en compte les diverses compétences des acteurs qui participent à la conception, à la fabrication et même au suivi du fonctionnement du produit tout au long de son cycle de vie. En effet avec les caractères multi-composants, multi-technologiques et multi-physiques des produits mécatroniques, leur développement fait intervenir des compétences différentes qui doivent s'échanger des informations tout au long du processus. Mieux, ces compétences doivent travailler ensemble pour réduire le coût et le temps de développement. En outre, la conception de tels produits intègre la prévision de leurs performances comportementales (fiabilité, maintenabilité, disponibilité, sécurité, recyclabilité, etc.). Il est, alors, nécessaire de regrouper les tâches de conception selon des critères définis préalablement pour gagner du temps et de l'efficacité. Pour un produit mécatronique, il est aussi nécessaire de bien intégrer toutes les technologies qui interviennent dans son fonctionnement dès la conception conceptuelle et de bien répartir les tâches entre les différents intervenants tout en gérant l'échange de connaissances et d'informations entre eux sans perte de données. Dans ce chapitre, nous proposons une démarche de conception conceptuelle sémantique comprenant trois phases [ICIDM] permettant de passer du cahier des charges à l'ensemble des solutions susceptibles de satisfaire les exigences et contraintes du cahier des charges. Ainsi, dans la Section 3.1 nous parlons de la classification des exigences et contraintes. L'identification des sous-systèmes et de leurs interactions est détaillée dans la Section 3.2. L'identification du domaine de

solutions éligibles est expliquée dans la Section 3.3. Enfin, le processus de conception conceptuel est décliné dans la Section 3.4.

3.1. Classification des exigences et contraintes

Dans cette étude, nous partons d'une étude fonctionnelle, structurelle et comportementale du cahier des charges. Après cette étude, les exigences fonctionnelles et structurelles sont listées, les contraintes supplémentaires sont identifiées et les performances attendues du futur produit sont répertoriées. Notre approche combine la conception collaborative [Choley, 2005], la conception concurrente [Choley, 2005] et la conception paramétrique basée contraintes. Elle est également compatible avec le V-Model [Ziemniak, 2009] et la méthodologie de conception systémique [Motte, 2008].

A la fin de l'analyse du cahier des charges, une première classification des exigences par niveau (de A_0 à A_3) selon leur type est faite. Nous avons 4 groupes d'exigences et contraintes :

- ✓ le groupe A_0 contient les exigences fonctionnelles ;
- ✓ le groupeA_1 contient les exigences structurelles ;
- ✓ le groupe A_2 contient les exigences comportementales ;
- ✓ le groupe A_3 contient les contraintes additionnelles : les délais de fabrication, les coûts, ...

Après cette première classification, une deuxième classification est opérée à l'intérieur des groupes obtenus ci-dessus. Ainsi, nous avons à l'intérieur du groupe :

1. A_0, 3 sous-groupes :
   - 1.1. les fonctionnalités principales du futur produit sont dans le sous-groupe A_{01} ;
   - 1.2. les fonctionnalités secondaires sont dans le sous-groupe A_{02} ;
   - 1.3. les services additionnels sont dans le sous-groupe A_{03}.
2. A_1, 4 sous-groupes :
   - 2.1. le sous-groupe A_{11} contient la liste des composants ;
   - 2.2. le sous-groupe A_{12} contient la liste des liaisons entre composants ;
   - 2.3. le sous-groupe A_{13} contient les contraintes de dimensions, de volumes, de matières, ... ;
   - 2.4. le sous-groupe A_{14} contient les contraintes d'assemblage.
3. A_2, 4 sous-groupes :
   - 3.1. les exigences de fiabilité sont classées dans le sous-groupe A_{21} ;
   - 3.2. les exigences de disponibilité sont classées dans le sous-groupe A_{22} ;
   - 3.3. les exigences de sécurité sont classées dans le sous-groupe A_{23} ;

3.4. les exigences de maintenabilité sont classées dans le sous-groupe A_{24}.

La Figure 3.1 montre la hiérarchisation des exigences de conception.

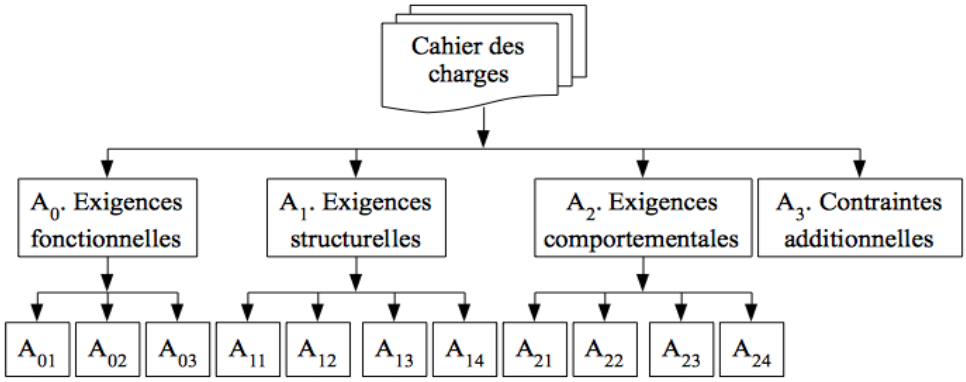

Figure 3.1 : Hiérarchisation des contraintes et exigences

Il faut noter que les composants et les liaisons sont représentés sous forme de classes d'objets dans le groupe A_1. Durant cette étape, pour chaque exigence fonctionnelle, les concepteurs fournissent une liste de composants et de sous-ensembles permettant de l'implémenter. Les composants contenus dans les différentes listes sont regroupés en familles de composants. Pour deux composants pris au hasard, tous les types de liaisons possibles entre eux sont identifiés. Les liaisons sont également regroupées dans des familles de liaisons. Ainsi, pour chaque famille de composants ou de liaisons ses fonctionnalités, ses propriétés, ses méthodes et ses contraintes sont identifiées. Après l'identification des exigences et contraintes, l'identification des sous-systèmes et de leurs interactions est traitée dans la Section 3.2.

3.2. Identification des sous-systèmes et de leurs interactions

La conception d'un produit mécatronique demande l'implication de personnes de compétences différentes. Ces personnes doivent non seulement s'échanger des informations, mais travailler ensemble dès les premières heures du processus. Ainsi, trois étapes sont définies pour identifier les sous-systèmes et les interactions qu'ils, ont entre eux, à partir des exigences fonctionnelles.

1. D'abord, à partir de la liste des exigences fonctionnelles, le produit est décomposé en sous-ensembles cohérents dont chacun permet de remplir une ou plusieurs fonctionnalités ;
2. Ensuite, partant de la liste des exigences structurelles, des sous-ensembles définis et de la liste des compétences requises, la structure de chaque sous-ensemble est dressée ;
3. Puis, partant de la liste des contraintes supplémentaires et des structures des différents sous-ensembles, les interactions entre les différents sous-ensembles sont listées ;

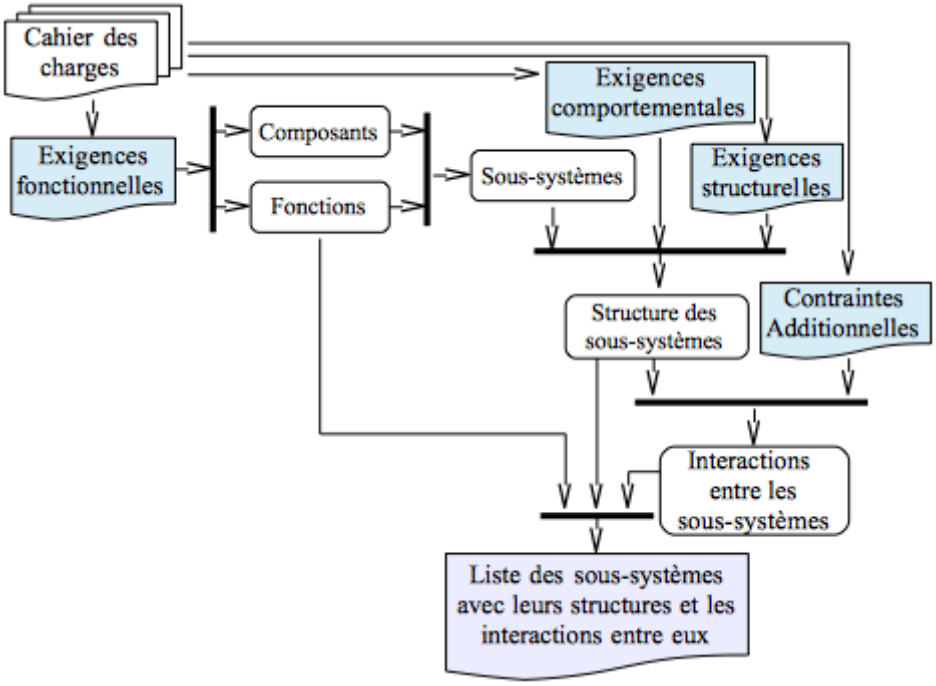

Figure 3.2 : Processus de répartition des tâches

A la suite de l'identification des sous-systèmes et de leurs interactions, pour chacun d'eux plusieurs instances peuvent candidater. Ces différents sous-systèmes et les interactions qu'ils ont entre eux permettent de concevoir le domaine des solutions éligibles (DSE) comme le montre la Section 3.3.

3.3. Détermination du domaine des solutions éligibles (DSE)

A partir de la liste des sous-systèmes obtenus dans la phase précédente, la dernière phase du processus de conception sémantique conceptuelle peut alors débuter. Pour chaque sous-système les fonctionnalités qu'il doit satisfaire sont ordonnées par priorité et/ou importance.

La priorité d'une exigence dépend de son rôle dans le fonctionnement du produit. Une exigence est prioritaire sue une autre si le rôle qu'elle joue est plus critique. Quand deux exigences ont la même priorité, l'importance dépend de la valeur ajoutée de l'exigence dans le fonctionnement du produit. Pour chaque sous-système, le processus d'identification de toutes les instances susceptibles de satisfaire ses exigences est fait en trois étapes que sont :

1. D'abord, la classification des exigences par priorité de P_1 à P_n (n >= 1) et par importance de I_1 à I_m (m >= 1) ;
2. Puis, l'identification d'un premier ensemble de solutions en considérant toutes les versions possible pour ce sous-système ;
3. Enfin, l'application des exigences fonctionnelles et structurelles selon leurs priorités (de P_1 à

P_n) et leurs importances (de I_m à I_1) si deux exigences ont la même priorité. Après chaque ensemble de priorités d'ordre P_i (1 <= i <= n) un ensemble de sous-systèmes est identifié. A la fin, un ensemble de sous-systèmes éligibles est obtenu. A l'intérieur de chaque ensemble, les sous-systèmes peuvent ne pas avoir le même nombre de composants.

A noter que les nombres n et m sont déterminés en fonction du produit à concevoir et que plusieurs espaces intermédiaires notées ECS_i (1 <= i < n), peuvent apparaître entre ECS et DSE comme le montre la Figure 3.3. Si :

- la priorité de E_x est P_x et son importance est I_x ;
- la priorité de E_y est P_y et son importance est I_y.

Alors E_x est prioritaire sur E_y si x est inférieur à y. Si les exigences E_x et E_y ont la même priorité, E_x est plus importante que E_y si x est supérieur à y.

Si deux exigences n'ont pas la même priorité, leur importance n'intervient pas car celle ayant la plus grande priorité est satisfaite avant l'autre.

Ainsi, pour chaque sous-système identifié dans la phase 2, un ensemble de sous-systèmes éligibles est obtenu. L'ensemble des solutions obtenues en faisant des combinaisons de sous-systèmes éligibles sans application des interactions est appelé l'espace conceptuel de solutions (ECS). Si trois sous-systèmes (SE_1, SE_2 et SE_3) sont identifiés à partir du cahier des charges, trois ensembles de sous-systèmes éligibles sont obtenus (S_1, S_2 et S_3) :

- S_1 contient les sous-systèmes candidats pour représenter SE_1 dans le produit ;
- S_2 contient les sous- systèmes candidats pour représenter SE_2 dans le produit ;
- S_3 contient les sous- systèmes candidats pour représenter SE_3 dans le produit.

Si n_1, n_2 et n_3 représentent respectivement les cardinalités des ensembles S_1, S_2 et S_3 alors le nombre de solutions contenu dans ECS et noté N_0, est obtenu avec l'équation suivante :

$$N_0 = n_1 * n_2 * n_3 \qquad (3.1)$$

Pour obtenir le domaine de solutions éligibles (DSE), les contraintes d'interaction identifiées entre les sous-systèmes sont appliquées. Les combinaisons ne respectant pas ces contraintes sont alors éliminées. Un ensemble de solutions respectant ces contraintes d'interaction est alors obtenu comme le montre la Figure 3.3.

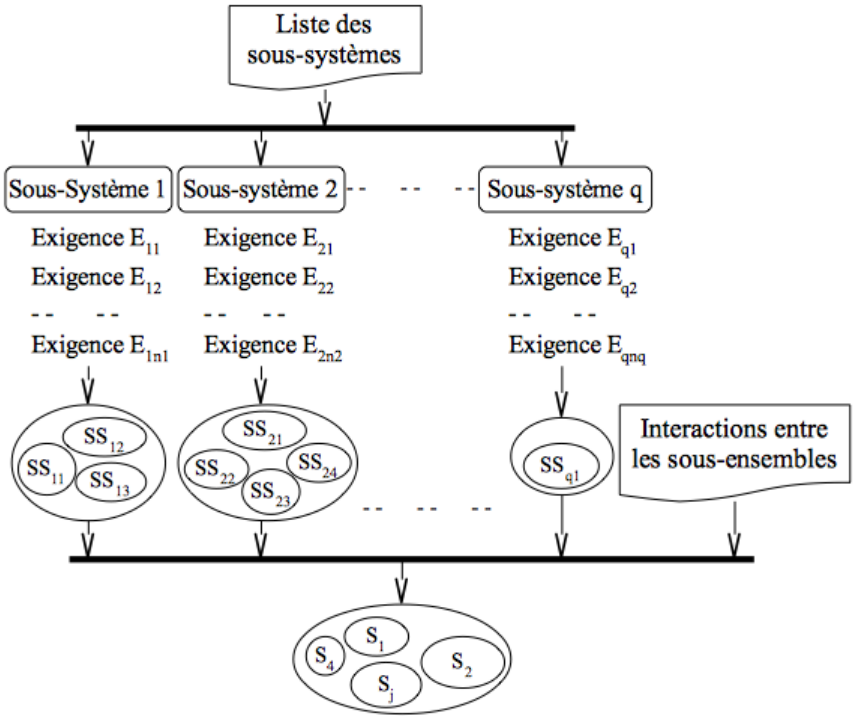

Figure 3.3 : Création du domaine de solutions éligibles

Chaque solution dans cet ensemble est une solution éligible et l'ensemble de ces solutions est le domaine des solutions éligibles (DSE). Deux solutions éligibles peuvent avoir différent nombre de composants. Dans le DSE, nous pouvons avoir plusieurs classes de solutions éligibles. Chaque classe de solutions éligibles est une famille de produits complexes. Les familles de solutions (produits complexes) du DSE peuvent être différents de par leur :

- Nombre de composants ;
- Les exigences fonctionnelles qu'elles satisfont ;
- Les technologies utilisées pour concevoir leurs instances ;
- Les exigences structurelles qu'elles satisfont ;

Si M_0 est le nombre de solutions éligibles dans le DSE, nous avons :

$$M_0 \leqslant N_0 \qquad (3.2)$$

Si $M_0 = 0$ signifie que le DSE est un ensemble vide (DSE = Ø), les concepteurs peuvent modifier les contraintes d'interaction entre sous-ensembles, ou les structures des sous-ensembles ou apporter des modifications au cahier des charges. Après modification du cahier des charges, le processus est repris à partir de la 1ère, 2ème ou 3ème étape selon les modifications apportées. La Figure 3.4 donne une représentation du DSE à l'intérieur de l'ECS.

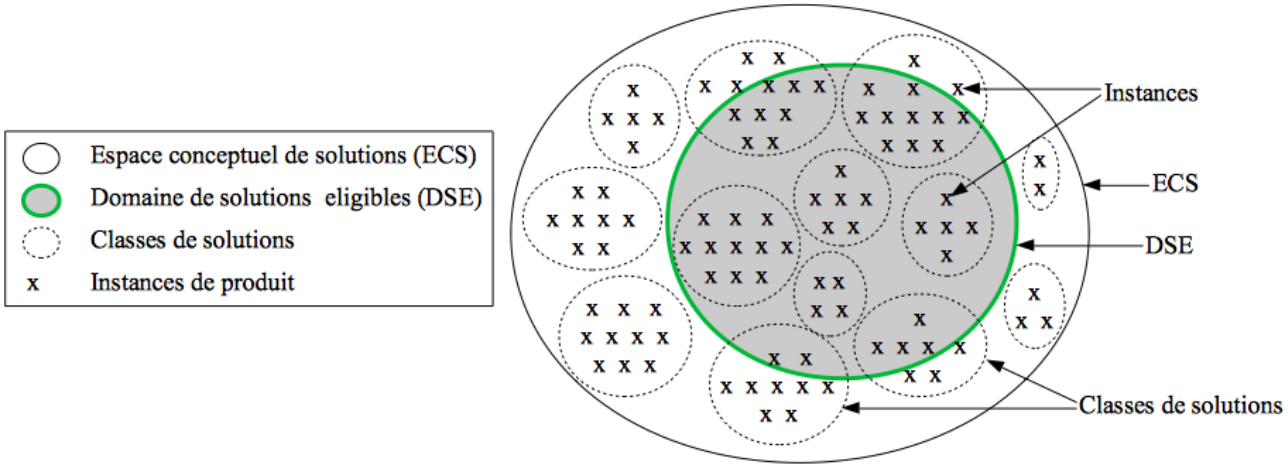

Figure 3.4 : Espace Conceptuel de Solutions et Domaine de Solutions Eligibles

Dans cette figure, chaque classe de solutions est une famille de produits complexes.

Le processus de conception sémantique conceptuelle permettant de passer du cahier des charges au DSE est détaillé dans la Section 3.4.

3.4. Processus de conception sémantique conceptuelle

Comme précisé ci-dessus, la démarche de conception sémantique conceptuelle proposée fait appel à des méthodologies existantes. Ces méthodologies sont utilisées comme suit :

a) La conception collaborative permet de prendre en compte dès le début l'aspect multi-technologique des produits complexes ;

b) La conception concurrente quant à elle permet de prendre en compte, dès la phase de conception, tous les acteurs qui interviendront durant tout le cycle de vie du produit;

c) La conception paramétrique et la conception par contraintes permettent d'exprimer les exigences fonctionnelles et les contraintes par des fonctions et équations qui seront utilisées pour la création des sous-systèmes permettant d'obtenir l'ECS puis le DSE.

L'approche proposée est également compatible à l'approche V-Model et à la méthodologie de conception systémique. Ainsi, ces deux méthodologies peuvent utiliser cette approche pendant la conception primaire pour remplacer leur étape de conception conceptuelle.

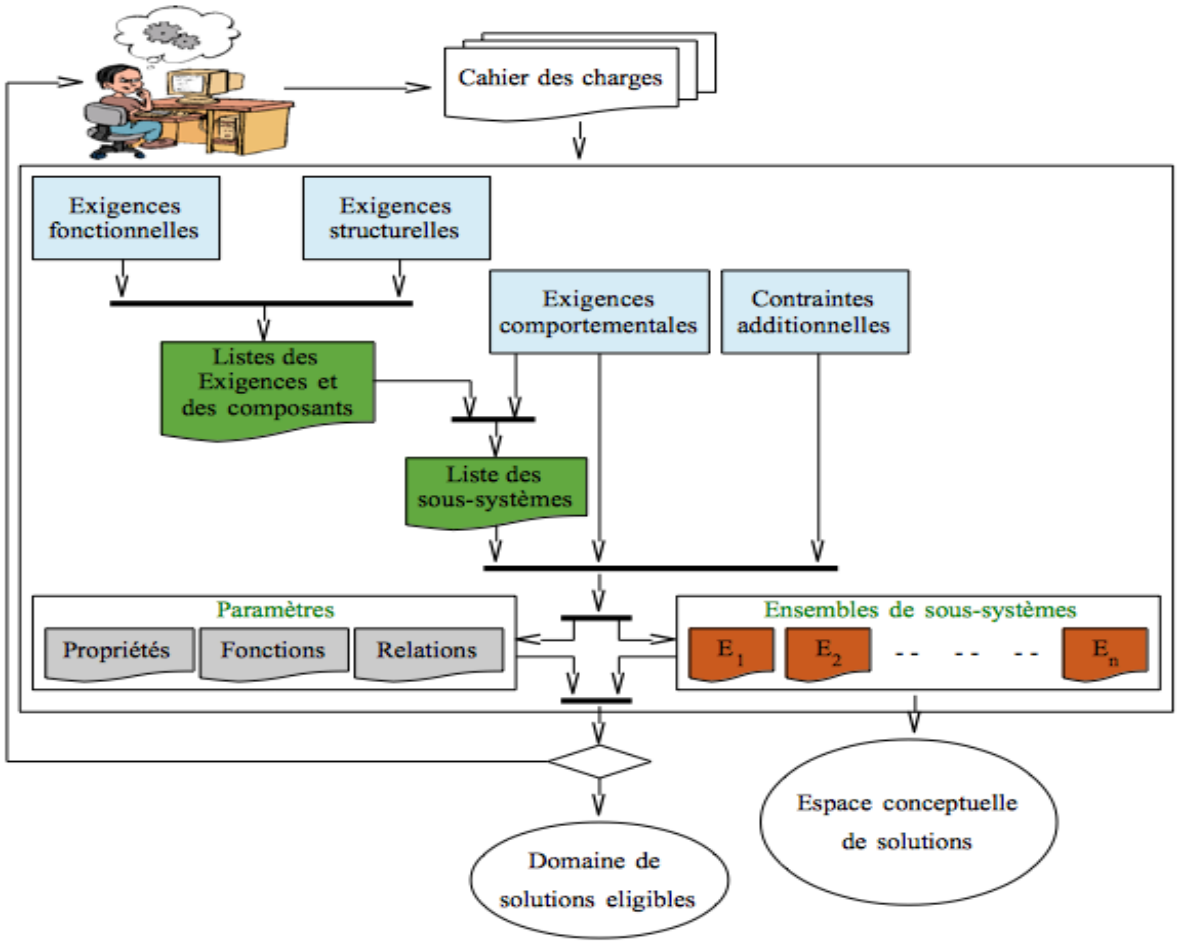

Figure 3.5 : Le processus de conception sémantique conceptuelle

Il faut aussi noter qu'à chaque étape du processus de conception sémantique conceptuelle décrite sur la Figure 3.5, le résultat est enregistré dans une base de données distribuée accessible à l'ensemble des acteurs pour permettre une répartition efficace et en temps réel des informations.

Conclusion

La conception des produits mécatroniques est une activité en constante progression depuis des années. Pour aller vers une modélisation sémantique conceptuelle de tels produits, nous avons proposé une méthodologie de conception sémantique conceptuelle basée sur la classification des contraintes et exigences suivants leur priorité et le rôle qu'elles jouent durant la conception, la fabrication et l'exploitation du produit. Cette méthodologie de conception utilise des méthodologies déjà existantes telles que la conception collaborative, la conception concurrente, la conception paramétrique et la conception par contraintes. Elle permet de créer des ensembles de solutions du plus générique (ECS) au plus spécifique (DSE). L'avantage est de pouvoir en même temps avoir des gammes de produits différents selon les contraintes et exigences satisfaites, les technologies utilisées, etc. Le passage du DSE aux modèles conceptuels est décrit dans le chapitre 4.

Chapitre 4 : Modélisation sémantique conceptuelle de produits complexes

Chapitre 4

Modélisation sémantique conceptuelle de produits complexes

Introduction

La modélisation du DSE est une étape très importante dans le processus de développement de produits complexes. Elle permet de représenter les solutions éligibles avec des modèles en vue de l'évaluation de leurs performances comportementales. Plusieurs outils peuvent être utilisés pour cela. Dans ces travaux, nous utilisons le langage SysML et les matrices DSM pour concevoir le modèle objet des classes de solutions contenues dans le DSE. Le langage SysML est de plus en plus utilisé ces dernières années pour modéliser les produits complexes [Fontan, 2008][Skander, 2008][Dobre, 2010][Lasalle, 2012]. On a, cependant, montré ses limites dans le chapitre 2 quant à la représentation de toutes les informations nécessaires à l'évaluation des performances comportementales de produits complexes en phase de conception préliminaire. De plus, il ne donne pas un modèle unique de la structure d'un produit. Ainsi, nous utilisons SysML pour concevoir le modèle des composants, le modèle des liaisons et le modèle des exigences fonctionnelles des classes de produits complexes. Puis, nous utilisons une version améliorée des DSM pour concevoir un modèle unifié des composants et des liens qui les relient. En effet, ces dernière années, les DSM ont été améliorées pour permettre une représentation plus complète des produits complexes [Coulibaly, 2010][Hong, 2009][Sharon, 2009]. Cependant, nous avons là aussi constaté des manquements qui rendent les modèles à base de DSM incomplets et peu fiables pour l'évaluation des performances comportementales de produits complexes.

Partant de ces constats, nous proposons dans ce chapitre une méthodologie de modélisation sémantique conceptuelle comprenant deux étapes [DSMConf, ComInd] : la modélisation des classes de composants, des classes de liaisons et des classes d'exigences fonctionnelles de produits complexes avec le langage SysML (Section 4.1) et la représentation et l'enrichissement de ces modèles avec une matrice sémantique conceptuelle multi-solution étendue (Section 4.2).

4.1. Modélisation de produits complexes avec le langage SysML

La modélisation avec UML est une pratique très établie dans l'industrie logicielle. Bien que le langage permette par son caractère à usage général d'adresser de nombreux besoins pour l'Ingénierie Système (IS), il est nécessaire de l'adapter par la définition d'un nouveau "profils UML" pour la modélisation des systèmes complexes. C'est ainsi que SysML qui est un profil de UML 2 a été créé.

SysML est fait pour décrire, spécifier et analyser la structure et le fonctionnement des systèmes complexes. Il permet également de concevoir des systèmes composés de sous-systèmes, mais aussi de vérifier et de valider la faisabilité d'un système avant sa réalisation [OMG SysML, 2006].

Un produit mécatronique est un produit complexe multi-composant et multi-technologique. Il est, ainsi, constitué d'un assemblage de composants de diverses technologies et fonctionnalités. Son assemblage est fait en utilisant des liaisons de différentes natures (mécanique, fonctionnelle, échange de flux) [DSMConf]. Un tel produit peut être modélisé en utilisant les outils de modélisation orienté-objet tels que UML, Express, SysML, etc. Durant les travaux de cette thèse, nous utilisons le langage SysML pour modéliser les composants des produits complexes, les liaisons qui les relient et les fonctionnalités qu'ils doivent remplir. Ces composants, liaisons et fonctionnalités sont alors représentés sous forme de classes d'objets avec des propriétés, méthodes et contraintes. Le modèle obtenu est un méta-modèle objet représentant plusieurs classes (familles) de produits complexes.

Nous parlons dans la Section 4.1.1 de la méthodologie de modélisation des classes de composants. Dans la Section 4.1.2, la méthodologie de représentation des classes de liaisons est expliquée. Dans la Section 4.1.3, la modélisation des classes de fonctions donnant les fonctionnalités du produit est détaillée.

4.1.1. Modélisation des composants

Un produit mécatronique peut être assimilé à un réseau de composants interconnectés par des liaisons et devant collaborer pour la satisfaction d'un objectif donné. Ces composants sont généralement de diverses technologies (mécanique, électronique, logiciel, etc.) et ont des interactions de diverses natures. Il est, alors, nécessaire de pouvoir les caractériser et d'étudier leurs interactions éventuelles dès la phase de conception préliminaire afin d'améliorer les performances comportementales des produits. Le diagramme de définition de blocs (DDB) SysML permet de représenter les composants d'un système complexe.

Ainsi, nous proposons de caractériser les différents types de composants mécatroniques dans la Section 4.1.1.1 et de fournir un modèle objet des composants mécatroniques et de leurs interactions en utilisant le diagramme de classes UML dans la Section 4.1.1.2.

4.1.1.1. Caractérisation des classes de composants

Un composant mécatronique est un constituant d'un produit mécatronique présentant un niveau partiel d'intégration mécatronique, du point de vue fonctionnel et physique, associant mécanique et électronique et permettant le traitement de l'information [NF E 01-010]. Il peut être une pièce élémentaire ou un sous-ensemble présentant un niveau partiel d'intégration mécatronique. Un tel

composant peut être caractérisé par des propriétés, méthodes et contraintes qui lui sont propres. Cependant, d'autres contraintes et exigences peuvent être dictées par le client, les concepteurs ou les conditions de fonctionnement. Ces contraintes sont souvent des contraintes d'assemblage, de performance ou des contraintes environnementales, etc. Les contraintes propres à un composant sont généralement la durée de vie donnée par le fabriquant, les conditions dans lesquelles il doit être utilisé (température, humidité), etc.

Ainsi, nous proposons des propriétés et méthodes génériques permettant de caractériser les composants mécaniques (Section 4.1.1.1.1), électroniques (Section 4.1.1.1.2) et logiciels (Section 4.1.1.1.3).

4.1.1.1.1. Composant mécanique

Pour un composant mécanique nous proposons les propriétés suivantes :

- ***Forme* :** donne la forme géométrique du composant ;
- ***Dimension* :** donne les dimensions du composant ;
- ***Poids* :** donne le poids du composant ;
- ***Matière* :** donne la/les matière(s) avec laquelle le composant est fabriqué ;
- ***Usure* :** donne le pourcentage d'usure du composant ;

En plus de ces propriétés des méthodes telles que ***Fabriquer()***, ***Casser()***, ***Recycler()***, sont proposées pour les composants mécaniques.

4.1.1.1.2. Composant électroniques

En ce qui concerne les composants électroniques nous proposons les propriétés suivantes :

- ***Type* :** donne le type du composant (résistance, condensateur, etc.) ;
- ***Matière* :** donne la matière avec laquelle le composant est fabriqué ;
- ***Usure* :** donne le pourcentage d'usure du composant ;
- ***Garantie* :** donne la durée de garantie du composant.

Nous proposons les mêmes méthodes que celles des composants mécaniques pour les composants électroniques ***Fabriquer()***, ***Casser()***, ***Recycler()***.

4.1.1.1.3. Composant logiciel

Pour un composant logiciel nous proposons les propriétés et méthodes suivantes :

- ***Effaçable* :** est un booléen qui permet de savoir si le programme est effaçable ou non ;
- ***Plate-forme* :** donne la/les plate-forme(s) sur lesquelles le logiciel peut tourner ;
- ***Taille* :** donne la taille du logiciel en kiloBit, MégaBit ou GigaBit.

Les composants logiciels (programmes applicatifs) sont caractérisés par les méthodes

Développer(), ***Installer()***, ***Désinstaller()***, ***SePlanter()***.

Après avoir proposé un certain nombre de propriétés et méthodes permettant de caractériser les composants mécaniques, électroniques et logiciels, nous donnons dans la Section 4.1.1.2 un modèle de classes UML des composants mécatroniques.

4.1.1.2. Modèle objet des composants mécatroniques

La représentation objet des différents types de composants mécatroniques est donnée dans la Figure 4.1. Dans ce modèle, on voit que :

- un composant est soit une pièce élémentaire ou un ensemble de pièces cohérent appelé sous-ensemble ;
- un sous-ensemble est composé d'un ou de plusieurs pièces élémentaires ;
- une pièce élémentaire est soit une pièce mécanique, électronique ou logiciel ;

Les propriétés et méthodes proposées dans la Section 4.1.1 sont représentées dans ce modèle.

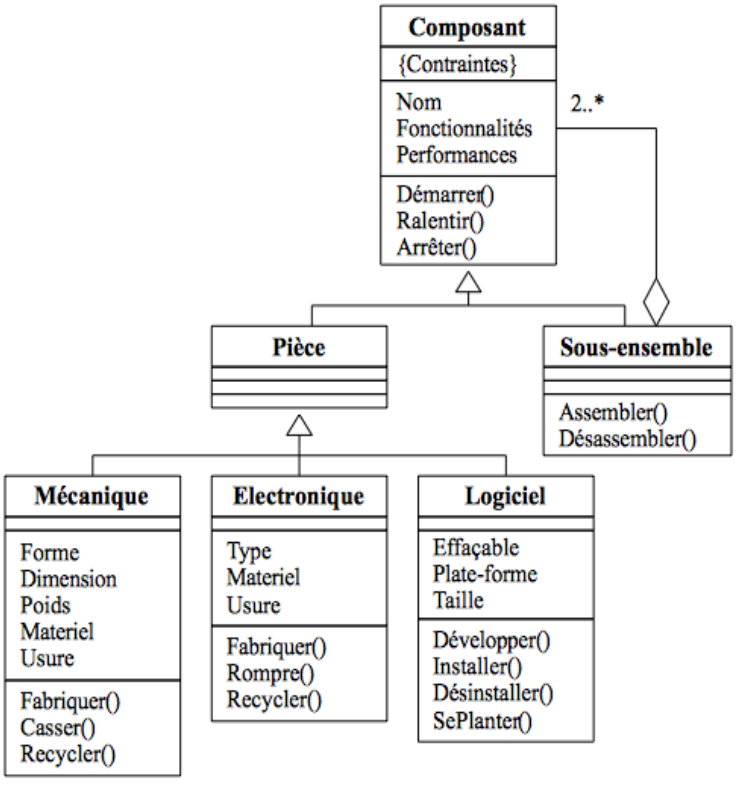

Figure 4.1 : Modèle objet des composants mécatroniques

Les classes de composants d'une famille de produits mécatroniques sont représentées par le diagramme de définition de blocs (DDB) SysML. En effet, ce diagramme permet de modéliser les composants sous forme de blocs avec une hiérarchisation allant des composants de niveau 1 au

composants de niveau le plus bas [Roques, 2013]. La modélisation des liaisons est décrite dans la Section 4.1.2.

4.1.2. Modélisation des liaisons

Les composants d'un produit mécatronique sont reliés par des liaisons de différentes natures. Parmi ces liaisons certaines permettent d'assembler les composants mécatroniques qui forment le produit (liaison mécanique). D'autres permettent de matérialiser les échanges (échange de flux) ou les interactions fonctionnelles (liaison fonctionnelle) entre ces composants. Généralement, dans la phase de conception d'un produit mécatronique seules les liaisons mécaniques sont prises en compte et leur description reste souvent incomplète. Ainsi, pour chaque type de liaison nous proposons une description permettant de le caractériser dans le modèle SysML du produit. Le langage SysML à travers son diagramme de bloc interne permet de représenter les échanges de flux entre composants. Cependant, cette représentation est assez pauvre car elle ne prend pas en compte beaucoup d'informations utiles pour la caractérisation des flux échangés. De plus, les autres types de liaison ne sont pas pris en compte dans le diagramme de blocs internes de SysML.

Ainsi, nous parlons dans la Section 4.1.2.1 de la caractérisation des classes de liaisons avant de proposer un modèle objet des liaisons avec le diagramme de classes UML dans la Section 4.1.2.2 et de modéliser les classes de liaisons en utilisant le diagramme de blocs interne SysML auquel nous apportons quelques améliorations dans la Section 4.1.2.3.

4.1.2.1. Caractérisation des classes de liaisons

Une liaison reliant deux composants peut être décrite par plusieurs caractéristiques. Certaines de ces caractéristiques sont très importantes pour l'évaluation des performances comportementales des systèmes complexes. En effet, le comportement d'un système dépend en grande partie de ses constituants (ses composants) mais aussi des relations entre ses constituants. Il est donc nécessaire, pour avoir un modèle complet à même de permettre une bonne analyse et d'obtenir des résultats fiables, de compléter le modèle des composants d'un système par celui des relations entre ces composants. La Figure 4.2 donne un ensemble de caractéristiques pouvant décrire un objet liaison.

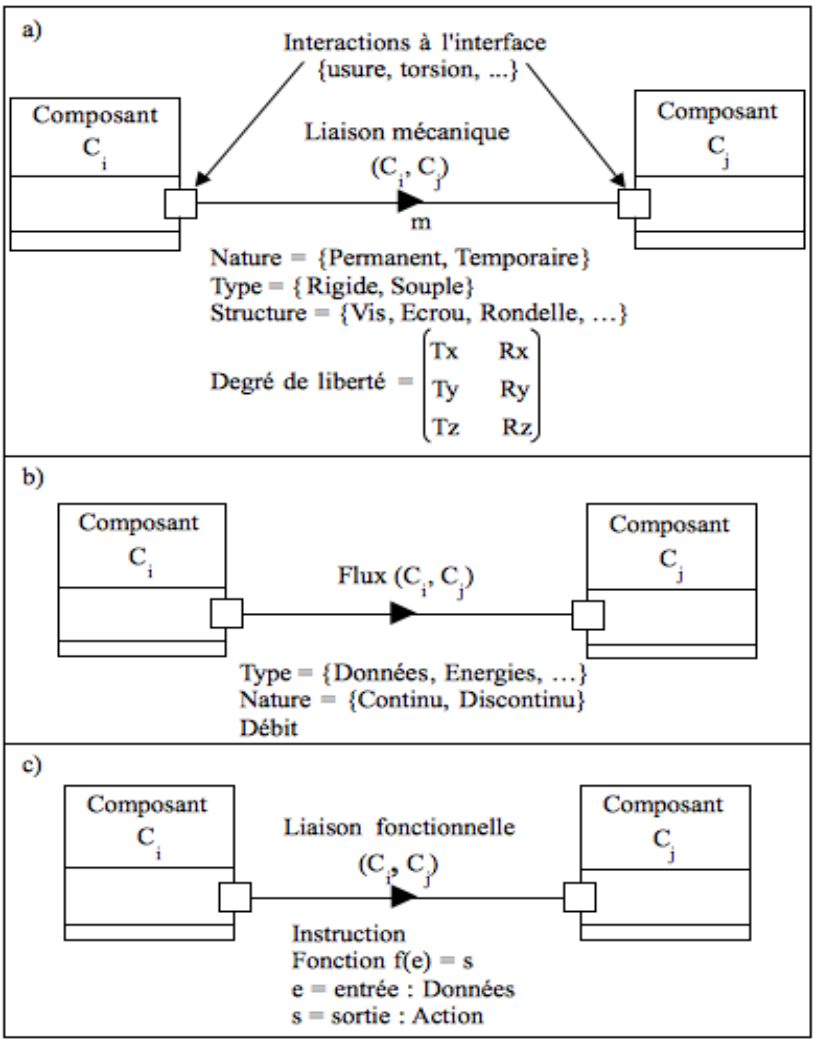

Figure 4.2 : Caractérisation des types de liaisons entre composants

Nous donnons ci-dessous un ensemble de caractéristiques associées à un objet liaison en fonction du type de liaison. Les caractéristiques principales sont représentées sur la Figure 4.2.

a. Liaison mécanique

- ***nature* :** Une liaison peut être permanente ou temporaire. Une liaison est permanente si on ne peut plus l'enlever alors qu'une liaison temporaire peut être défaite à tout moment. S'il s'agit d'un contact, il est soit ponctuel, filaire ou surfacique [Mekhilef, 1998].
- ***type* :** Une liaison peut être rigide ou souple. Une liaison rigide est une liaison qui ne permet pas de mouvement des composants à son interface (pas de degré de liberté) alors qu'une liaison souple permet un mouvement à l'interface de la liaison.
- ***degré de liberté* :** Lorsqu'une liaison est souple, le degré de liberté ***d*** doit être spécifié (rotation autour d'un axe : d = 2, translation suivant un axe : d = 1, etc...). Chacun de ces mouvements pouvant s'effectuer selon un axe donné. Ainsi le degré de liberté est une matrice de trois lignes

et 2 colonnes donnant les mouvements possibles à l'interface de la liaison.

- ***structure* :** Certaines liaisons utilisent des éléments de liaison (vis, écrou, rondelle, etc.) qui peuvent être pris dans des bibliothèques préexistantes.

Certaines décisions ne peuvent être prises qu'au moment de la matérialisation physique de la liaison. Ces décisions font apparaître d'autres caractéristiques qui permettent de compléter la description de la liaison. Ces caractéristiques doivent aussi être prises en compte dans le modèle. Parmi ces caractéristiques nous avons :

- ***nombre d'instances* :** Une liaison entre deux composants C_i et C_j d'un système complexe donné peut être mono-instance ou multi-instance. Une liaison entre deux composants est multi-instance s'il y a plus d'un type de liaison entre les deux composants. Par exemple il peut y avoir entre C_i et C_j une instance de vissage et une instance de soudage en même temps. Dans ce cas il y a deux instances de liaisons entre les deux composants.
- ***nombre d'occurrences d'une instance* :** Chacune des instances d'une liaison peut être mono-occurrence ou multi-occurrence. Par exemple, C_i peut être vissé trois fois sur C_j. Alors, l'instance de type vissage entre C_i et C_j a 3 occurrences. Une occurrence peut utiliser plusieurs éléments (vis, écrou, rondelle, ...).
- ***interaction à l'interface* :** Il est aussi possible qu'il y ait usure, torsion, etc. à l'interface d'une liaison entre deux composants. Ces éléments sont très importants pour prévenir une rupture ou une cassure de la liaison.
- ***sens* :** Quand deux composants sont liés entre eux, il est souvent possible de dire lequel des deux vient s'ajouter à l'autre. Nous le matérialisons par une flèche qui indique le sens de la liaison. Dans le cas d'un vissage, on pourra alors dire qu'un composant C_1 est vissé sur un composants C_2. Cette propriété est très importante pour déterminer le chemin de désassemblage et pour calculer le temps que dure ce désassemblage.

Ces huit caractéristiques sont très importantes pour décrire une liaison mécanique entre deux composants.

b. Echange de flux

Pendant son fonctionnement, les composants d'un produit mécatronique s'échangent des flux pour le bon fonctionnement du produit. Ces flux peuvent être des données numériques, de l'énergie, des instructions, etc. Ces flux doivent être représentés dans le modèle du produit en phase préliminaire de conception. Ils peuvent être caractérisés par les propriétés suivantes :

- ***type*** de flux (données, énergie, instruction, etc...) ;
- ***nature*** du flux (continu ou discontinu) ;

- ***débit*** de l'envoi ;
- ***sens*** du flux qui donne le composant qui envoie et le composant qui reçoit.

c. Liaison fonctionnelle

Deux composants C_i et C_j sont liés par une liaison fonctionnelle si l'un peut déclencher une action de l'autre. Ainsi, une instruction émise par C_i déclenche l'exécution de fonction(s) par C_j. La fonction peut être de n'importe quel type allant d'un simple calcul au démarrage d'une action cinématique. Elle peut être définie comme étant "une description abstraite et générale d'un effet d'interaction entre des données d'entrée et des données de sortie et l'état du système pour la réalisation d'un objectif" [Mekhilef, 1998]. Ainsi, une liaison fonctionnelle peut être décrite par les propriétés suivantes :

- ***l'instruction*** qui déclenche la fonction ;
- ***la fonction*** déclenchée par l'instruction venant de l'autre composant ;
- ***les données d'entrée*** de la fonction ;
- ***l'action*** (l'algorithme) exécutée après déclenchement de la fonction
- ***les données de sortie*** de la fonction ;

Il faut noter, cependant, que ces listes de propriétés ne sont pas exhaustives. Pendant la conception, le concepteur, compte tenu de ses objectifs et des contraintes du cahier des charges, jugera des propriétés à donner aux liaisons. Après avoir caractérisé les différents types de liaison, nous proposons un modèle objet des liaisons dans la Section 4.1.2.2.

4.1.2.2. Modèle objet des liaisons d'un produit mécatronique

Dans cette section nous donnons la représentation objet des différents types de liaison sur la Figure 4.3.

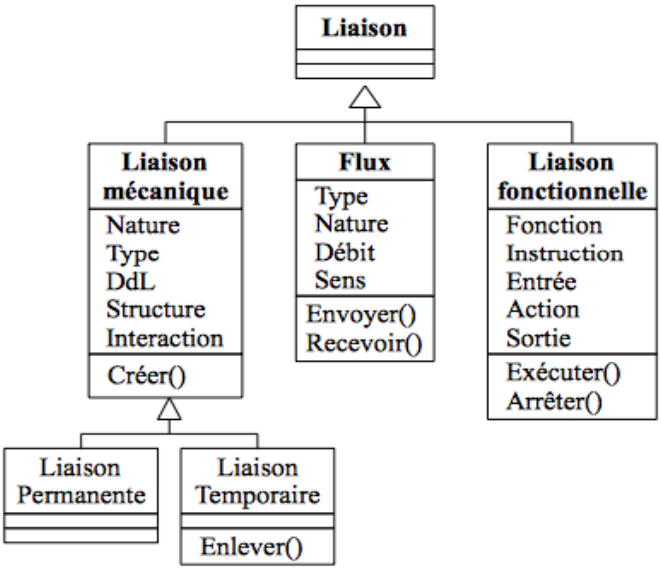

Figure 4.3 : Modèle objet des liaisons de produits mécatroniques

Nous proposition, dans la Section 4.1.2.3 une amélioration du diagramme de blocs interne de SysML pour permettre de prendre en compte les caractéristiques proposées dans la Section 4.1.2.1.

4.1.2.3. Modélisation des classes de liaisons avec SysML

Pour modéliser les classes de liaisons en langage SysML, nous utilisons le diagramme de bloc interne auquel nous ajoutons d'autres informations caractérisant les types de liaisons cités ci-dessus. Ainsi, nous montrons dans la Figure 4.4 comment représenter une classe de liaisons. La liaison représentée dans cette figure est une liaison mécanique, mais tous les types de liaisons peuvent être représentés de la même manière.

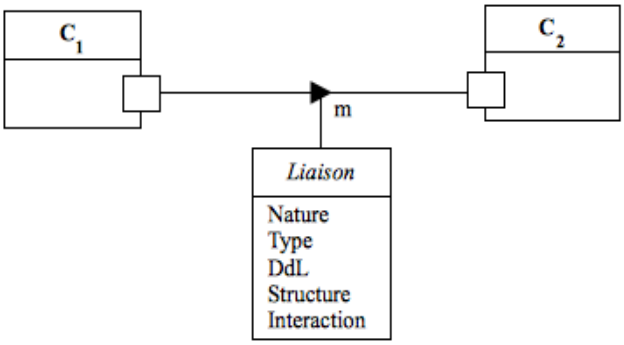

Figure 4.4 : Modèle SysML d'une classe de liaisons

Interprétation du modèle : Nous avons un système de 2 composants C_1 et C_2. Tout ce que nous savons au moment de la modélisation est qu'il y a une liaison entre C_1 et C_2. On ne connait pas en ce moment les types de liaison qui seront utilisés, ni leur nombre d'instances encore moins le nombre d'occurrences de chaque instance ou les éléments de liaison (vis, boulon, écrou, etc.) qui seront utilisés pour les occurrences. Ainsi, nous créons une classe d'objets appelée ***Liaison*** qui modélise les types de liaison possibles (vissage, boulonnage, soudure, etc.).

La classe ***Liaison*** est reliée à chacune des liaisons entre deux composants. Au niveau de chaque liaison nous avons le sens de la liaison, matérialisé par une flèche, et le nombre d'occurrences, matérialisé par un nombre entier (m). Les autres propriétés (Nature, Type, DdL, Structure et Interaction) sont décrites dans la Section 4.1.2.1 (Liaison mécanique).

Après avoir caractérisé les différents types de composant et les différents types de liaison que l'on peut trouver dans un produit mécatronique, nous montrons comment modéliser les exigences fonctionnelles avec les X-CDSM dans la Section 4.1.3.

4.1.3. Modélisation des exigences fonctionnelles de produits mécatroniques

La modélisation fonctionnelle de produits mécatroniques au stade précoce de la conception est très importante pour l'étude de leurs performances comportementales durant leur phase d'exploitation.

Le cahier des charges fonctionnel contient les fonctionnalités essentielles que doit remplir le produit à concevoir. Dans cette section, nous traitons la modélisation des exigences fonctionnelles de familles de produits complexes avec le langage SysML. SysML donne avec son diagramme des exigences la possibilité de modéliser les exigences d'un système complexe. Nous traduisons les exigences fonctionnelles en fonction(s) composée(s) d'un ou de plusieurs modules. Ces fonctions et modules seront représentés par des classes d'objets dans le modèle fonctionnel de la famille de produits.

Pour ce faire, nous commençons d'abords par caractériser les fonctions et leurs modules éventuels dans la Section 4.1.3.1. Ensuite, nous donnons le modèle objets des fonctions et de leurs modules avec le diagramme de classes UML dans la Section 4.1.3.2.

4.1.3.1. Caractérisation des classes de fonctions et de leurs modules

Les exigences fonctionnelles répertoriées dans le cahier des charges fonctionnel sont traduites en fonctions à remplir par le produit. Chaque exigence est satisfaite par une ou plusieurs fonctions. Ces fonctions peuvent être elles-mêmes constituées chacune d'un ou de plusieurs modules. Les exécutions des modules peuvent être dépendantes les unes les autres. L'exécution d'un module peut dépendre des exécutions de plusieurs modules. Nous posons comme hypothèse que chaque module ne peut être exécuté que par et un seul composant.

Pour caractériser les fonctions et modules, nous proposons les propriétés suivantes :

- ***Rôle* :** qui contient l'exigence que la fonction permet de satisfaire ou rôle que joue le module dans la fonction ;
- ***Entrée* :** qui contient les données que la fonction ou le module prend en entrée ;
- ***Sortie* :** qui contient les données ou le service que la fonction ou le module produit ;
- ***Durée* :** qui contient le temps que dure l'exécution de la fonction ou du module ;
- ***Fiabilité* :** qui contient le pourcentage de fiabilité de la fonction ou du module ;
- ***Criticité* :** qui contient le pourcentage de criticité de la fonction ou du module.

En plus de ces propriétés, les méthodes ***exécuter()***, ***arrêter()***, ***valider()*** et ***évaluer()*** qui sont très importantes pour l'exécution et le test d'une fonction ou d'un module sont proposées. Cependant, d'autres fonctions et modules peuvent être ajoutés à celles-ci. La modélisation objet des fonctions et modules est traitée dans la Section 4.1.3.2.

4.1.3.2. Modèle objet des fonctions et de leurs modules

Le modèle objet permettant de représenter les fonctions et leurs modules ainsi que les relations entre eux est donné dans la Figure 4.5. La caractérisation et le modèle objet permettent de pouvoir modéliser les classes de fonctions ainsi que les classes de leurs modules.

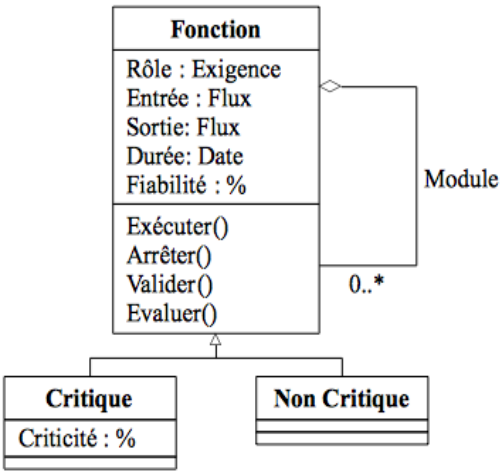

Figure 4.5 : Modèle objet des fonctions

Les fonctions (respectivement modules) sont représentées avec le diagramme des exigences de SysML. Ce diagramme permet de représenter les fonctions tout en spécifiant leurs modules et les relations entre elles. Ainsi, pour chaque instance d'une famille de produits complexes dont les fonctionnalités sont représentées par ce modèle fonctionnel, ses fonctions sont des instances des familles de fonctions représentées dans ce modèle. Un exemple de modèle fonctionnel est représenté sur la Figure 8.10 (Chapitre 8). La modélisation objet des produits mécatroniques permettant de représenter les caractéristiques des composants, les liaisons, les fonctions et les modules et de donner les interactions entre eux est traitée dans la Section 4.1.4.

4.1.4. Modèle objet de produits mécatroniques

Nous commençons par donner le modèle objet représentant les relations entre les composants, les liaisons et les fonctions d'un produit mécatronique représentés sur les Figures 4.1, 4.3 et 4.5 respectivement. Ce modèle est représenté sur la Figure 4.6.

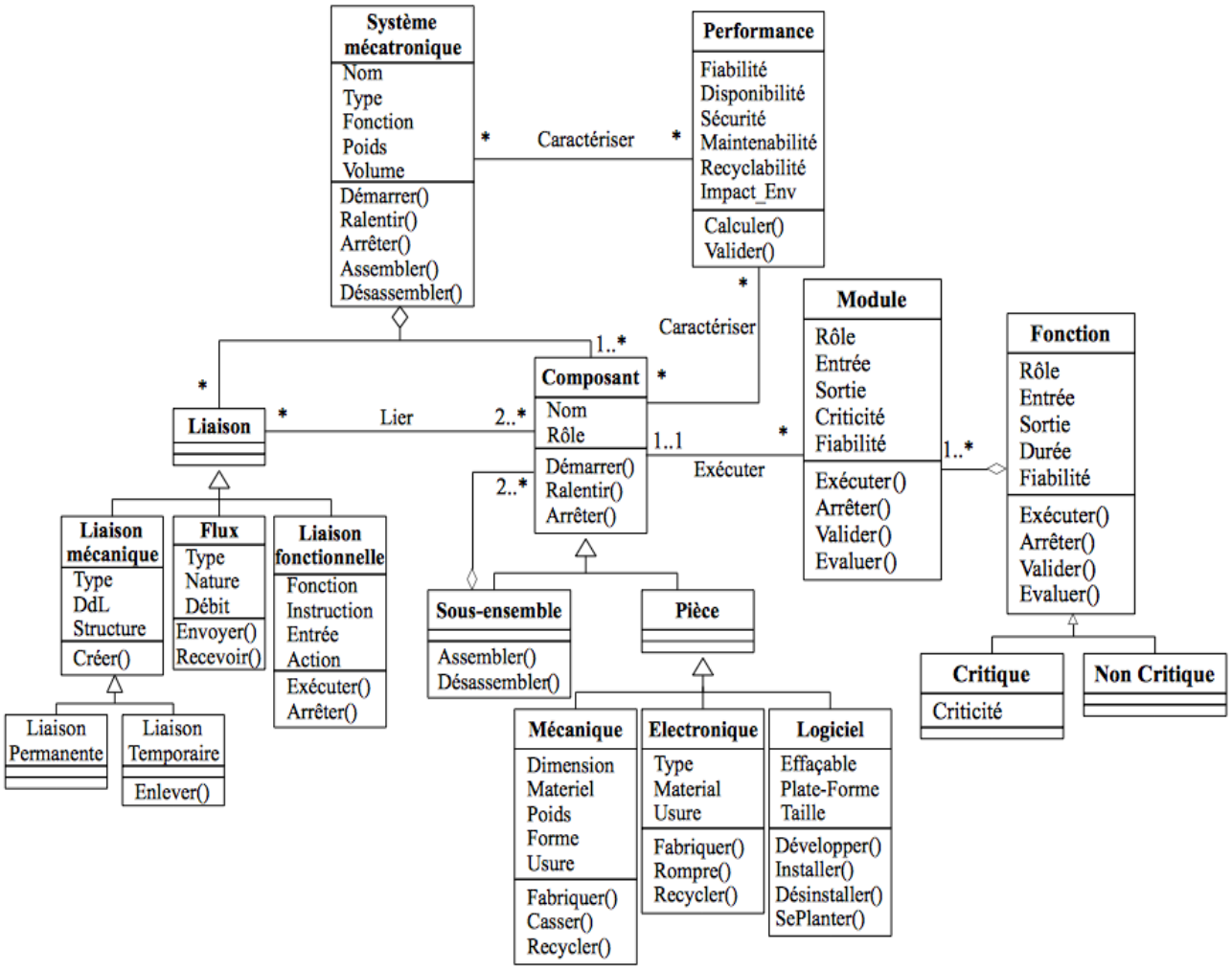

Figure 4.6 : Modèle objet des classes de composants et liaisons d'un produit mécatronique

Cette méthodologie de modélisation de produits mécatroniques avec le langage SysML sera utilisée dans le chapitre 8 pour modéliser le cas d'étude. Nous donnons dans la Section 4.2 la méthodologie de modélisation de produits mécatroniques avec les matrices DSM étendues.

4.2. Modélisation avec les X-CDSM

En vue de la création du modèle géométrique correspondant au modèle SysML, ce dernier est par la suite représenté par une Matrice Sémantique Conceptuelle Etendue X-CDSM [ComInd]. Une X-CDSM est une amélioration de la X-DSM [Coulibaly, 2008a][Coulibaly, 2010] qui est elle-même une amélioration de la DSM basée composants. Ainsi, nous partons des DSM classiques basées sur les composants traitées dans la Section 4.2.1 aux X-CDSM décrites dans la Section 4.2.3 en passant par les X-DSM (Section 4.2.2) en expliquant toutes les améliorations qui leur sont apportées. Puis nous donnons la description d'une MSX-CDSM (Section 4.2.4) qui est un ensemble de X-CDSM.

4.2.1. DSM basées sur les composants

Une DSM est une représentation simple, compacte et visuelle d'un système sous la forme d'une matrice [web 10]. Il existe deux types de DSM : les DSMs statiques et les DSMs dynamiques ou temporelles [Hong, 2009][Sharon, 2009]. Les DSMs temporelles ont été étudiées par T. Browning dans [Browning, 2001]. Dans cette section nous nous intéresserons surtout aux DSMs basées sur les composants. En effet, ces DSM permettent de caractériser les composants des produits complexes, mais aussi de représenter les liaisons qu'ils ont entre eux [Coulibaly, 2007][Coulibaly, 2010]. Dans une telle DSM, chaque composant est représenté sur une ligne et une colonne. A l'intersection des lignes et des colonnes les liaisons entre composants sont représentées sauf si la ligne et la colonne représentent le même composant. Une telle matrice est représentée sur la Figure 4.7.

Composants	C_1	C_2	--	--	C_n
C_1		1	0	0	1
C_2			0	0	1
--				0	0
--					0
C_n					

Figure 4.7 : DSM basée sur les composants

Soit D_{ij} l'intersection de la ligne i et de la colonne j.

- Si i = j, alors D_{ij} est une cellule vide ;
- Si i ≠ j, alors D_{ij} contient V_{ij} qui représente la liaison entre C_i et C_j :
  a) Si V_{ij} = 0, il n'y a pas de liaison entre C_i et C_j ;
  b) Si V_{ij} = 1, il y a liaison entre C_i et C_j.

Pour une représentation plus complète des produits complexes, ces DSMs ont été enrichies pour passer aux X-DSM [Coulibaly, 2007][Coulibaly, 2008a][Coulibaly, 2010] décrites dans la Section 4.2.2.

4.2.2. DSM étendues (eXtended Design Structure Matrix : X-DSM)

Pour enrichir les DSM basées sur les composants, une ligne donnant le nombre de liaisons pour chaque composant et des colonnes donnant la Criticité, la Fiabilité, l'Impact environnemental, etc... y sont ajoutées. Ces informations permettent d'améliorer la description des produits complexes et de rendre leurs performances comportementales obtenues à partir de tels modèles plus fiable. Ces DSMs seront appelées ***DSM étendues*** (X-DSM : eXtended Design Structure Matrix). La Figure 4.8 montre un exemple de X-DSM.

Composants	C_1	C_2	--	--	C_n	Criticité	Fiabilité	--
C_1	C_{11}	V_{12}			V_{1n}	K_1	R_1	--
C_2		C_{22}			V_{2n}	K_2	R_2	--
--			--			--	--	--
--				--		--	--	--
C_n					C_{nn}	K_n	R_n	--
Nb Liaisons	d_1	d_2	--	--	d_n	$K_{0(Seuil)}$	$R_{0(Seuil)}$	--

Figure 4.8 : DSM étendue (X-DSM)

Dans une telle matrice, la caractérisation des composants commence à apparaître dans les cellules diagonales. Cependant, en ce qui concerne les liaisons seuls leurs types (boulonnage, vissage, soudage, etc.) sont représentés par un entier entre 0 et 10 donnant leurs poids. Le Tableau 3.1 donne les poids des différents types de liaisons et plus une liaison est difficile à enlever, plus son poids est élevé. Une X-DSM est décrite comme suit :

Soit D_{ij} la cellule à l'intersection de la ligne i et de la colonne j ;

- Si i = j, D_{ij} contient C_{ii} qui donne les caractéristiques d'une instance de composant C_i ;
- Si i ≠ j, D_{ij} contient V_{ij} qui donne le poids du type de liaison entre les composants C_i et C_j ;
- R_i et K_i sont des estimations numériques caractérisant les données sémantiques (fiabilité et la criticité respectivement) du composant C_i ;
- R_0 et K_0 sont respectivement le seuil d'acceptabilité de la fiabilité et de la criticité ;
- d_i est le nombre de liaisons du composant C_i.

Tableau 4.1 : Poids des types de liaisons [Coulibaly, 2008b]

Type de liaison	Poids
Sans contact	$q_1 = 0$
Contact	$q_2 = 1$
Clipsage	$q_3 = 2$
Vissage	$q_4 = 3$
Boulonnage	$q_5 = 4$
Rivetage	$q_6 = 6$
Carter	$q_7 = 7$
Collage	$q_8 = 8$
Soudage	$q_9 = 10$

Ainsi, les cellules diagonales contiennent les données caractérisant les composants et les cellules non-diagonales contiennent les poids des types de liaison. Les caractéristiques des composants peuvent être des caractéristiques physiques (structurelles) ou fonctionnelles. Cependant, pour chaque composant, seule une instance est représentée. Alors, chaque X-DSM représente un seul produit. Avec la création des X-CDSM, une seule matrice peut représenter toute une famille de produits complexes comme le décrit la Section 4.2.3.

4.2.3. DSM sémantiques conceptuelles étendues (X-CDSM)

Dans une matrice X-CDSM deux approches de modélisations (DSM et modélisation objet) sont combinées pour permettre de modéliser toute une famille de produits avec une seule matrice. Nous avons deux types de X-CDSM : les X-CDSM structurelle décrites dans la Section 4.2.3.1 et les X-CDSM fonctionnelles décrites dans la Section 4.2.3.2.

4.2.3.1. X-CDSM structurelle

Dans une X-CDSM structurelle, les composants et les liaisons sont représentés par des classes d'objets [DSMConf, ComInd]. Ceci permet de mieux décrire les composants et de représenter pour chaque composant plusieurs variantes. Cette représentation permet également une description plus fine des liaisons et de donner aux concepteurs la possibilité de choisir, le moment venu, l'instance de liaisons la plus appropriée entre deux composant. Ainsi, une X-CDSM représentée sur la Figure 4.9 donne la possibilité de modéliser plusieurs instances d'une famille de produits complexes.

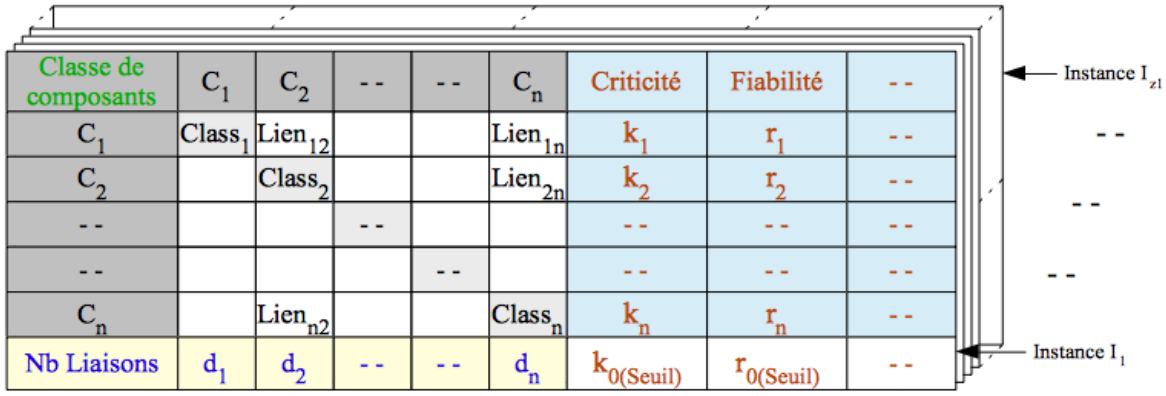

Figure 4.9 : X-CDSM structurelle

Dans cette matrice,

- $Class_i$ est une classe de composants dont les instances sont des composants possibles pour le produit modélisé par la matrice ;
- $Lien_{ij}$ est une classe de liaisons dont les instances sont les liaisons possibles entre des instances de composants des classes $Class_i$ et $Class_j$;

- r_i et k_i sont des variables représentant respectivement la fiabilité et la criticité des instances de la classe de composants $Class_i$. Cependant, selon le cahier des charges, d'autres attributs sémantiques tels que l'impact environnemental, la biodégradabilité, la réutilisabilité, etc. peuvent apparaître ;
- r_0 et k_0 sont des variables contenant les seuils d'acceptabilité de la fiabilité et de la criticité respectivement ;
- d_i est une variable contenant le nombre de liaisons de l'instance choisie dans la $class_i$.

A partir d'un tel modèle, plusieurs concepteurs pourront, selon les besoins et les exigences exprimés dans le cahier des charges, concevoir des produits différents aussi bien au niveau des caractéristiques des composants et des liaisons qu'au niveau des performances. Durant nos travaux, nous utilisons ces matrices sémantiques étendues pour modéliser un ensemble de produits qui peuvent être différents structurellement (nombre de composants, différence des liaisons) en leur donnant la possibilité de modéliser plusieurs instances d'une solution. Ainsi, chaque X-CDSM représente plusieurs instances d'une même solution. La matrice X-CDSM d'une solution S_i ayant z_i instances sera alors définie comme suit :

- Cl_i est l'ensemble contenant toutes les classes de composants de S_i ;
- Lk_i est l'ensemble contenant toutes les classes de liaisons de S_i ;
- n_i est la cardinalité de Cl_i ;
- m_i est la cardinalité de Lk_i ;
- I_{ij} est l'instance numéro j de la solution S_i.

Alors :

- $S_i = \{I_{ij} / 1 \leq j \leq z_i\}$;
- $Cl_i = \{Class_j / 1 \leq j \leq n_i \leq n\}\ Cl_i \neq \emptyset$;
- $\begin{cases} \text{Si } m_i = 0 \Rightarrow Lk_i = \emptyset ; \\ \text{Sinon Si } m_i \neq 0 \Rightarrow Lk_i = \{Lien_{ij} / (i \neq j) \cap (1 \leq i \leq n_i) \cap (1 \leq j \leq n_i)\} \end{cases}$

Si nous prenons comme exemple la solution S_1 ayant z_1 instances et décrite comme suit :

- $S_1 = \{I_{1j} / 1 \leq j \leq z_1\}$;
- $n_1 = 3$;
- $m_1 = 2$;
- $Cl_1 = \{Class_j / 1 \leq j \leq n_1\} = \{Class_1, Class_2, Class_3\}$;
- $Lk_1 = \{Lien_{xy} / (x \neq y)$ et $(1 \leq x \leq n_1)$ et $(1 \leq y \leq n_1)\} = \{Lien_{12}, Lien_{23}\}$.

La matrice X-CDSM de cette solution est représentée sur la Figure 4.10.

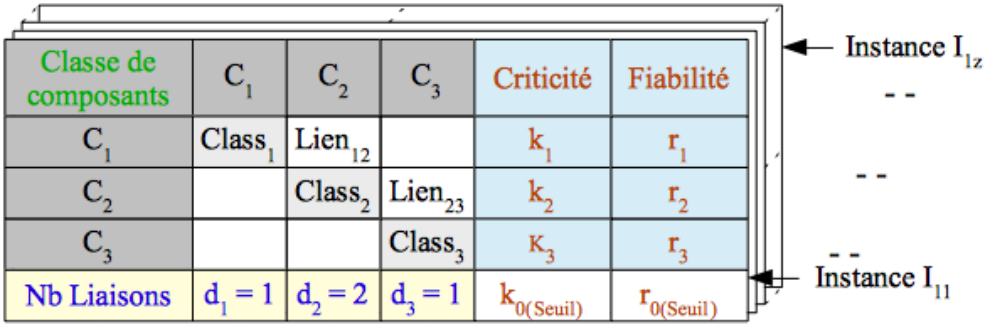

Figure 4.10 : X-CDSM de la solution S_1

Chaque instance de cette matrice est un produit complexe ayant 3 composants et 2 liaisons pouvant être représenté par une X-DSM. Donc, après instanciation on obtient une matrice X-DSM. Une X-CDSM est alors un ensemble de X-DSM. De la même manière que les exigences structurelles des produits, les X-CDSM peuvent représenter leurs exigences fonctionnelles. Cette représentation est décrite dans la Section 4.2.3.2.

4.2.3.2. X-CDSM fonctionnelle

Une X-CDSM fonctionnelle permet de représenter les exigences fonctionnelles mentionnées dans le cahier des charges fonctionnel [SysCon]. Les fonctions à exécuter par le produit pour satisfaire ces exigences sont représentées ainsi que leurs modules éventuels. Les relations entre les différents modules d'une fonction sont représentées ainsi que celles entre les modules de fonctions différentes. La Figure 4.11 donne un exemple de X-CDSM fonctionnelle.

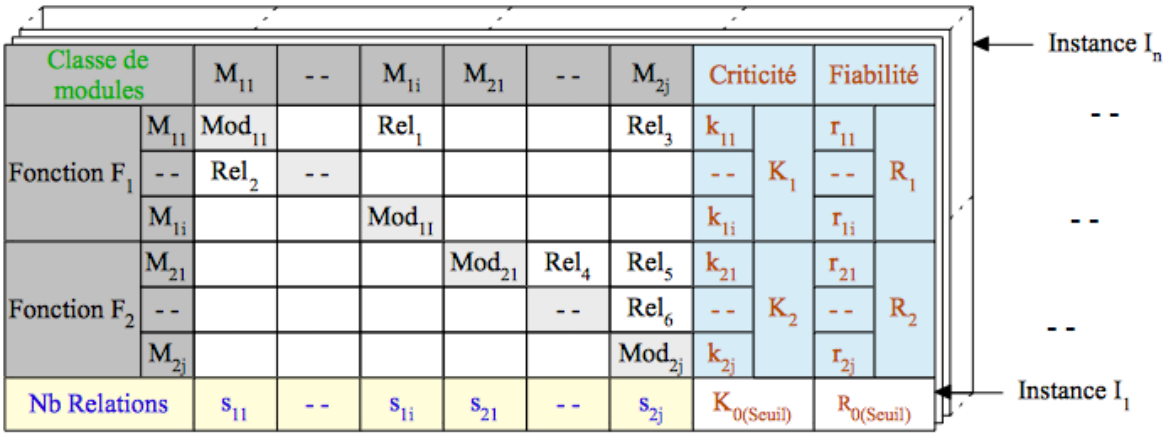

Figure 4.11 : X-CDSM fonctionnelle

Une X-CDSM fonctionnelle à la même structure qu'une X-CDSM structurelle. Cependant, dans une X-CDSM fonctionnelle :

- Les cellules diagonales contiennent les classes de modules ;
- Les cellules non diagonales contiennent les classes de relations entre les classes de modules ;
- Mod_{ij} est une classe de modules qui représente les propriétés, méthodes et contraintes

éventuelles du module M_{ij} ;

- Rel_{ij} est une classe de relation entre les classes de modules Mod_{ii} et Mod_{jj}. Elle est également caractérisée par des propriétés, méthodes et contraintes ;
- s_{ij} est le nombre de relations qu'a le module M_{ij} avec d'autres modules ;
- k_{ij} et r_{ij} sont respectivement la criticité et la fiabilité du module M_{ij} ;
- K_i et R_i sont respectivement la criticité et la fiabilité de la fonction F_i ;
- K_0 et R_0 sont respectivement les seuils de criticité et de fiabilité des fonctions.

Si nous avons une relation dans la cellule située à l'intersection de la ligne i et de la colonne j, alors l'exécution du module situé à la colonne j dépend de celle du module située à la ligne i. D'après la Figure 4.11, l'exécution du module M_{2j} dépend de celle du module M_{i1} à cause de la relation Rel_3. On note également :

- Cf_i l'ensemble contenant toutes les fonctions de la solution S_i ;
- Cm_i l'ensemble contenant toutes les classes de modules de la solution S_i ;
- x_i la cardinalité de l'ensemble Cf_i ;
- y_i la cardinalité de l'ensemble Cm_i ;
- Rm_i l'ensemble contenant toutes les classes de relations de la solution S_i.

Si on prend l'exemple d'un cahier des charges fonctionnel dont l'étude permet d'identifier 2 fonctions F_1 et F_2 avec respectivement 2 modules (M_{11} et M_{12}) et 1 module (M_{21}) pour la solution S_k. Alors, la Figure 4.12 représente la X-CDSM correspondante avec p instances.

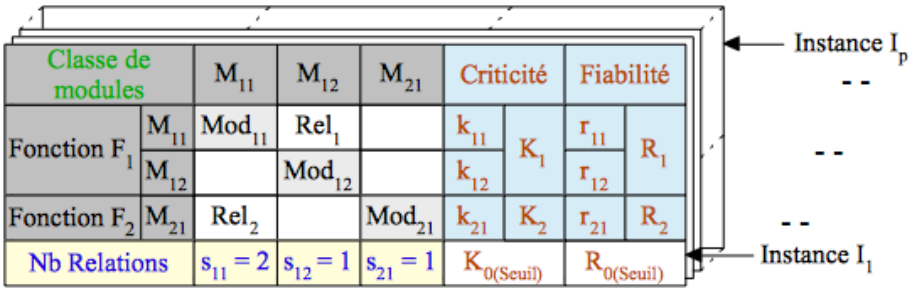

Figure 4.12 : X-CDSM fonction de la solution S_k

- $x_k = 2$;
- $y_k = 3$;
- $Cf_k = \{F_1, F_2\}$;
- $Cm_k = \{Mod_{11}, Mod_{12}, Mod_{21}\}$;
- $Rm_1 = \{Rel_1, Rel_2\}$.

Chaque instance de cette matrice est la matrice fonctionnelle d'une instance de produit. Ainsi, avec une telle X-CDSM, on peut donner plusieurs ensembles de fonctions différents (plusieurs instances de la X-CDSM) qui répondent tous aux exigences d'un même cahier des charges. Ainsi, chaque instance de produits aura une instance de X-CDSM fonctionnelle qui représente ses fonctionnalités. Ces instances satisfont les mêmes exigences avec des performances différentes.

Pour élargir le champ des solutions pendant la phase préliminaire de conception nous enrichissons les X-CDSM pour modéliser plusieurs solutions. Les matrices obtenues, décrites dans la Section 4.2.4, sont appelées matrice sémantique conceptuelle étendues multi-solutions.

4.2.4. DSM sémantique conceptuelle étendue multi-solution (MSX-CDSM)

Les matrices sémantiques étendues permettent de modéliser une solution unique à partir d'un cahier des charges. Notre objectif est de rendre ces matrices plus ouvertes avec la possibilité de modéliser un ensemble de solutions différentes contenant elles-mêmes des instances différentes. Ainsi, partant du cahier des charges, plusieurs étapes sont nécessaires pour concevoir cette matrice.

Un modèle sémantique multi-solution est un ensemble de X-CDSM. Il permet de modéliser des produits qui peuvent ne pas avoir la même nomenclature, les mêmes fonctionnalités, les mêmes performances. Les produits ayant la même nomenclature sont regroupés dans une même classe de solutions. A l'intérieur de chaque classe les produits pourront se différentier par leurs performances, leurs fonctionnalités, etc. Chaque élément (produit) d'une classe de solutions est alors nommé instance.

Les classes de liaisons nommées $Lien_{ij}$ montrent que la direction de la liaison est du composant C_i vers le composant C_j.

Une MSX-CDSM structurelle (notée MS) est un ensemble de X-CDSM structurelle représentant plusieurs solutions (S_i) dont chacune est composée d'un ensemble de composants et de liaisons peut être exprimée de la manière suivante :

- $MS = \{S_i / 1 \leq i \leq k\}$: k étant le nombre de classes de solutions ;
- $S_i = C \cup L / C \neq \emptyset$: C étant l'ensemble des classes de composants et L l'ensemble des classes de liaisons ;
- $C = \{C_j / 1 \leq j \leq n\}$: n est le nombre de classes de composants du système ;
- $$\begin{cases} \text{Si } m = 0 \Rightarrow L = \emptyset ; \\ \text{Sinon } L = \{L_p = Lien_{xy}, (\&1 \leq p \leq m) \cap (1 \leq x \leq n) \cap (1 \leq y \leq n) \cap (x \neq y)\} \end{cases}$$
- m est le nombre de classes de liaisons. Sa valeur maximale est m_{max} telle que :

$$m_{max} = n * n - n = n(n-1) \qquad (4.1)$$

Les classes de solutions d'une MSX-CDSM structurelle peuvent ne pas avoir le même nombre de classes de composants, alors que toutes les instances d'une même classe de solutions doivent avoir le même nombre de classes de composants.

Les MSX-CDSM fonctionnelles sont décrites de la même manière que les MSX-CDSM structurelles.

La Figure 4.13 et le Tableau 4.2 donnent un récapitulatif des améliorations et des avantages et inconvénients des différents types de DSM.

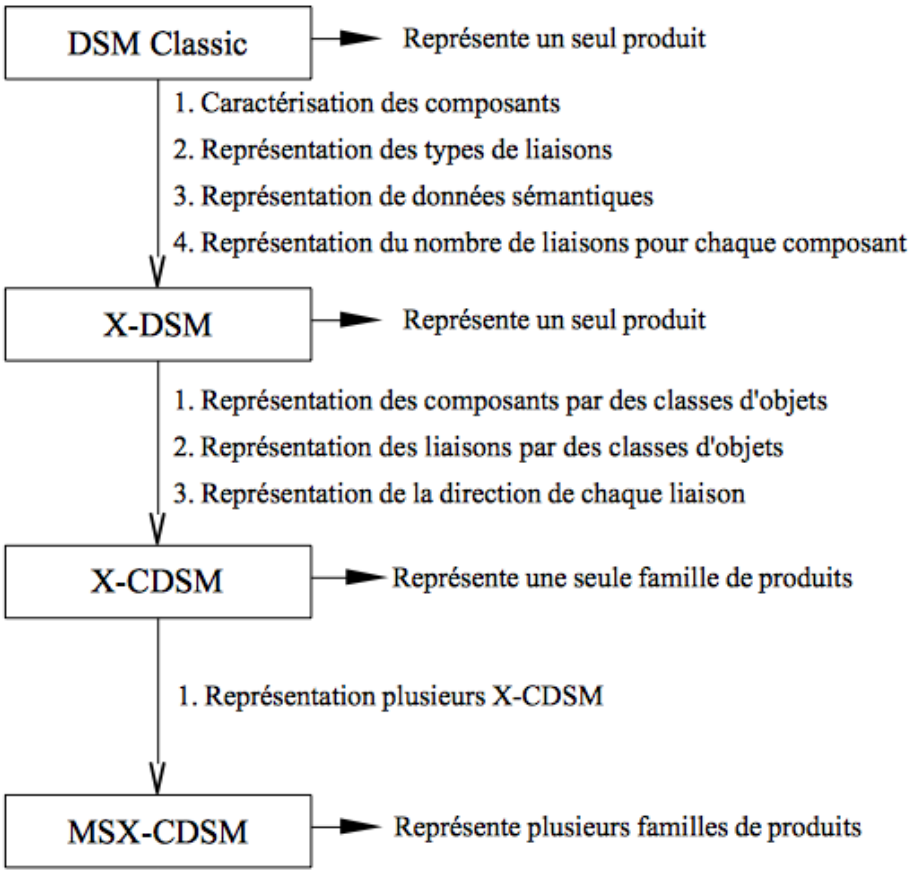

Figure 4.13 : Améliorations apportées au DSM classiques

Tableau 4.2 : Avantages et inconvénients des différents types de matrices

Type de DSM	Représentation des composants et des liaisons	Avantages (+) et inconvénients (-)
DSM basées composants traditionnelles	1. Les caractéristiques des composants ne sont pas représentées; 2. Seules la présence ou l'absence de liaison entre deux composants sont représentées.	+ Donne une représentation visuelle des composants et des liaisons ; + Permet de savoir les composants qui sont liés entre eux ; - Représente une seule instance de produit ; - Impossible de connaître le chemin de démontage ou de calculer la durée de remplacement d'un composant défaillant [Coulibaly, 2007] ; - Impossible d'évaluer la fiabilité d'un produit [Coulibaly, 2007] ; - Difficile de la représenter s'il y a plusieurs composants.
X-DSM	1. Les caractéristiques des composants sont représentées mais pas sous forme de classe d'objet ; 2. Seul le type de la liaison (vissage, boulonnage, collage, etc.) entre deux composants est représenté. Ces informations sont représentées par des poids (Table 4.1)	+ Donne une représentation visuelle des composants et des liaisons ; + Permet d'évaluer la fiabilité du produit en se basant sur celles de ses composants critiques [Coulibaly, 2008b] ; + Fournit l'impact environnemental, la recyclabilité, etc. des composants ; + Permet de savoir quels composants sont liés entre eux et le type de leur liaison ; - Représente une seule instance de produit ; - Impossible de connaître le chemin de démontage ou de calculer la durée de remplacement d'un composant défaillant [Coulibaly, 2007] ; - Difficile de la représenter s'il y a plusieurs composants.
X-CDSM	1. Les composants sont représentés sous forme de classes d'objets 2. Les liaisons sont représentées sous forme de classes d'objets 3. La direction de la liaison $Lien_{xy}$ est de la classe $Class_x$ vers la Classe $Class_y$.	+ Donne une représentation visuelle des composants et des liaisons ; + Représente plusieurs instances de produit (une famille de produits) ; + Permet de construire le chemin de démontage d'un composant défaillant ; + Permet de calculer la durée de remplacement d'un composant défaillant ; + Permet d'évaluer la fiabilité du produit en se basant sur celles de ses composants critiques [Coulibaly, 2008b] ; + Fournit l'impact environnemental, la recyclabilité, etc. des composants ;

Type de DSM	Représentation des composants et des liaisons	Avantages (+) et inconvénients (-)
		+ Permet de savoir les composants qui sont liés entre eux, le type précis de leur liaison et le sens de chaque liaison ; - Difficile de la représenter s'il y a plusieurs composants.
MSX-CDSM	Généralisation des X-CDSM	+ Identique aux avantages des X-CDSM mais permet aussi de modéliser plusieurs familles de produits - Difficile de la représenter s'il y a plusieurs composants.

Conclusion

Dans ce chapitre une méthodologie de modélisation de produits mécatroniques est proposé. Elle comprend deux parties, une modélisation objet des composants et liaisons avec le langage SysML et une vue synthétique de l'ensemble du système avec les matrices X-CDSM.

La modélisation objet est faite en trois étapes que sont : 1) la caractérisation et la modélisation des composants, 2) la caractérisation et la modélisation des liaisons permettant d'assembler les composants et 3) la modélisation des fonctionnalités de la famille de produits à concevoir. Une amélioration du diagramme de bloc interne du langage SysML est proposée pour permettre de représenter sur un même diagramme toutes les liaisons entre les composants d'un produit mécatronique. Ce modèle des liaisons permet de compléter le modèle des composants d'une famille de produits mécatroniques qui est fait avec le diagramme de définition de blocs.

La modélisation avec les X-CDSM permet de passer du modèle objet SysML vers l'ingénierie de performances comportementales traitée dans le chapitre 6. Pour cela, une amélioration des DSM est proposée pour permettre de représenter sur un même modèle plusieurs familles de produits complexes. Cette amélioration consiste en une combinaison des DSM et de la modélisation orientée-objet. Nous montrerons dans le chapitre 6 comment un tel modèle est utilisé pour calculer le temps de remplacement d'un composant défaillant ou la fiabilité basée sur les exigences fonctionnelles. Le module d'ingénierie de performances comportementales de produits complexes peut prendre en entrée un modèle CAO. Les modèles conceptuels obtenus après la MSC ne sont pas pris en compte par la CAO classique. Nous proposons une nouvelle génération de CAO appelée CAO sémantique conceptuelle dans le Chapitre 5.

Chapitre 5 : CAO sémantique conceptuelle

Chapitre 5

CAO sémantique conceptuelle

Introduction

La fabrication de produits robustes, performants, à moindre coût et dans les délais impartis (Qualité/Performance/Coût/Délai) requiert une étude approfondie de leurs structures, leurs fonctionnalités et leurs performances dès la phase de conception. Cette étude passe par une bonne modélisation et une évaluation de leurs performances en phase de conception. Nous avons parlé dans les chapitres 3 et 4 de la conception sémantique conceptuelle [ICIDM] et de la modélisation sémantique conceptuelle [DSMConf, ComInd, SysCon] respectivement. Ainsi, nous avons décrit une méthodologie permettant de concevoir des modèles conceptuels représentant des familles de produits complexes. Cependant, ce type de modèles n'est pas pris en compte par les systèmes de CAO actuels. La CAO est une pratique très établie dans l'industrie manufacturière. Elle permet de concevoir des modèles géométriques en 3 dimensions de produits et d'y effectuer des simulations numériques. Cependant, la plupart des systèmes existants sont spécialisés dans des domaines précis (mécanique, électronique, bâtiment). En outre, ils se limitent à gérer des configurations de pièces ou de produits mais ne permettent pas de gérer des modèles conceptuels pouvant donner plusieurs produits différents. Ainsi, nous posons, dans ce chapitre, les bases d'une nouvelle génération de CAO appelée CAO sémantique conceptuelle (CAO-SC) permettant de gérer plusieurs instances d'une famille de produits complexes. Elle permet, ainsi, de prendre en compte, en plus des données structurelles des produits et des composants (forme, dimension, matière utilisée, type de liaison, etc.), leurs exigences fonctionnelles, leurs données sémantiques (criticité, fiabilité, impact environnemental, etc.), etc.

Dans cet objectif, nous présentons dans la Section 5.1 les fonctionnalités de la CAO-SC. Puis, la Section 5.2 parle de l'utilisation des normes STEP et PLib en CAO-SC. Enfin, une comparaison entre la CAO-SC et la CAO classique est faite dans la Section 5.3.

5.1. Fonctionnalités de la CAO-SC

Les systèmes de CAO 3D actuels proposent deux méthodes de modélisation : la modélisation paramétrique basée sur les fonctions, et la modélisation explicite ou directe [CAD-Mag N° 180]. Ces deux méthodes se distinguent essentiellement par la prise en compte ou non de l'historique et de l'arborescence des fonctions utilisées. Avec ces deux méthodes, le passage d'un produit à un autre se fait en modifiant le modèle ou en créant un nouveau. L'objectif de la CAO-SC est de fournir un modèle conceptuel pouvant être instancié en fonction des exigences et contraintes. Elle peut être

décrite comme une CAO qui prend en entrée un modèle conceptuel représentant une famille de produits. Un tel modèle permettra de passer d'une version d'un produit à une autre.

Dans un système de CAO classique le modèle géométrique d'un produit est un assemblage de pièces et de sous-ensembles à l'aide de liaisons physiques. Ces systèmes sont généralement orientés métier (mécanique, électronique, etc.) comme le montre les modèles représentés sur la Figure 5.1.

a. CAO mécanique b. CAO électronique c. Building Information Modeling

Figure 5.1 : Modèles géométriques 3D conçus en environnement CAO

Avec l'intégration de plusieurs technologies dans un même produit, il est impossible de concevoir un modèle représentant toutes les caractéristiques d'un tel produit. Il est alors nécessaire de pouvoir modéliser des composants de technologies différentes, mais aussi de passer d'une version d'un produit à une autre sans être obligé de créer un nouveau modèle. Les systèmes de CAO doivent donc être adaptés pour prendre en compte la modélisation de produits mécatroniques et la modélisation sémantique conceptuelle. Ainsi, on pourra modéliser plusieurs familles de produits mécatroniques (classes de solutions) et plusieurs versions de chaque produit (instances d'une classe de solutions). La CAO-SC est proposée pour prendre en compte ces types de modélisation. Un système de CAO-SC est un système :

- pouvant gérer l'aspect multi-technologique des produits ;
- prenant en compte les modèles sémantiques conceptuels ;
- pouvant échanger avec les bibliothèques de composants PLib.

Nous détaillons ces trois aspects dans les sections suivantes. C'est ainsi que, dans la Section 5.1.1 nous traitons la gestion de l'aspect multi-technologique des produits. Dans la Section 5.1.2 nous parlons de la prise en compte de la modélisation sémantique conceptuelle. Dans la Section 5.1.3 nous décrivons l'interopérabilité entre les systèmes de CAO-SC et les normes STEP et PLib.

5.1.1. Gestion de l'aspect multi-technologique des produits mécatroniques

Ces dernières années, certains systèmes de CAO ont commencé à proposer d'intégrer la

modélisation électrique aux systèmes de CAO mécaniques (Siemens PLM software, Autodesk software) [web 14]. D'autres permettent une interopérabilité entre systèmes de CAO mécanique et systèmes de CAO électrique (Autodesk Inventor et AutoCAD electrical software, SolidWorks CircuitWorks, etc) [web 15, web 16]. Il est cependant à noter que jusqu'à présent la modélisation des produits mécatroniques en environnement CAO est un chalenge pour les concepteurs. Les produits mécatroniques étant composés généralement de parties mécaniques, électroniques et des logiciels, il est nécessaire de pouvoir matérialiser ces différentes parties dans le modèle géométrique 3D de tels produits. Ainsi, pour gérer la cohabitation des composants de différentes technologies dans un même système de CAO, il est nécessaire de pouvoir :

- concevoir les pièces mécaniques ;
- concevoir les pièces et circuits électroniques ;
- gérer les logiciels, les transferts de données, le traitement des informations ;
- gérer l'assemblage de composants de technologies différentes ;
- gérer les stratégies de contrôle-commande entre les composants.

Les deux premiers points sont assurés depuis de nombreuses années par les systèmes de CAO traditionnels. Cependant, la plupart d'entre eux se chargent soit de l'un soit de l'autre. Dans un système de CAO-SC, les composants mécaniques et électroniques ainsi que les modules informatiques et les interactions (liaisons mécaniques, échanges de flux, etc.) entre eux sont gérés par le même système. Ainsi, un système de CAO-SC intègre, entre autres, les fonctionnalités suivantes :

- conception de pièces mécaniques ;
- conception de pièces électroniques ;
- conception de circuits intégrés ;
- matérialisation d'un module informatique sur un composant ;
- gestion du transfert de données et d'énergies entre composants ;
- gestion du transfert d'instructions de contrôle entre composants ;
- gestion du transfert de commande entre composants ;
- gestion du traitement des informations par un composant ;
- conception des liaisons mécaniques entre composants ;
- conception des liaisons fonctionnelles entre composants ;
- etc.

5.1.2. Gestion de la modélisation sémantique conceptuelle

Une classe de produits mécatroniques est caractérisée par des classes de composants et des classes de liaisons. La modélisation sémantique conceptuelle en environnement CAO passe par la prise en

compte de ces classes par le système de CAO. La CAO-SC propose une méthodologie de modélisation permettant de représenter un modèle sémantique conceptuel SysML ou X-CDSM dans un système de CAO. Nous décrivons la modélisation des classes de composants dans la Section 5.1.2.1 et celle des classes de liaisons dans la Section 5.1.2.2. La modélisation d'une famille de produits en CAO-SC est décrite dans la Section 5.1.2.3.

5.1.2.1. Modélisation sémantique conceptuelle des composants en CAO-SC

Chaque produit est une instance d'une classe de solutions éligibles obtenue à partir du cahier des charges. Les différents produits d'une même classe de solutions ont le même nombre d'instances de composants. Cependant, deux produits d'une même classe peuvent contenir deux instances différentes pour une même classe de composants. Ces instances de composants pouvant avoir des caractéristiques et performances différentes [ComInd]. Elles sont caractérisées par des propriétés, des méthodes et éventuellement des contraintes (Figure 5.2). Chaque composant appartient à une famille dont les instances se distinguent par leur structure, leur performance, les contraintes à respecter, etc.

Pour gérer les classes de composants, le système de CAO est relié à une base de données relationnelle objet. Le concepteur donne les différentes classes du modèle objet avec leurs caractéristiques.

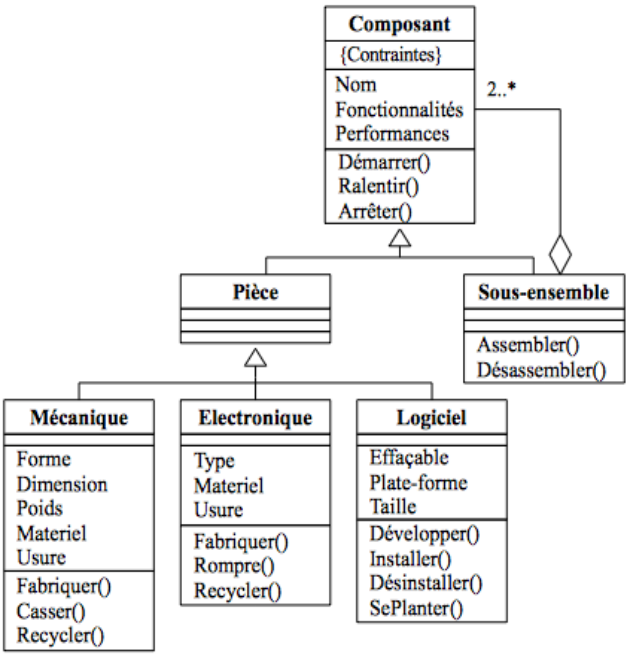

Figure 5.2 : Caractérisation des composants d'un produit mécatronique

Ainsi un composant n'est précisément décrit que quand ses propriétés sont instanciées en respectant les contraintes. Le système de CAO-SC peut donc chercher dans une bibliothèque de composants PLib une pièce qui correspond aux spécifications données. S'il ne le trouve pas, le

concepteur pourra alors concevoir le modèle géométrique 3D correspondant en se basant sur les valeurs des propriétés. Dans ce cas, le modèle est enregistré dans la bibliothèque de composants du système pour une utilisation ultérieure. La Figure 5.3 montre le processus de recherche d'une instance de composant correspondant aux données entrées par un concepteur.

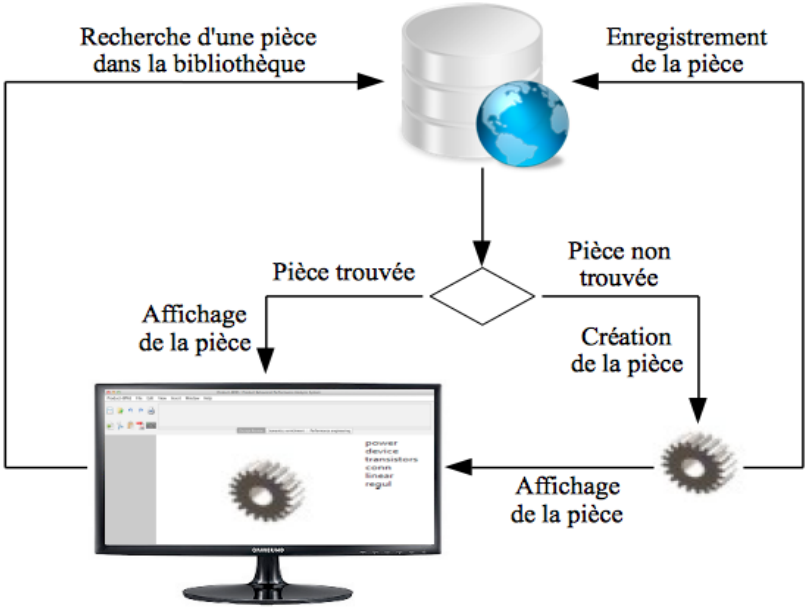

Figure 5.3 : Instanciation d'une classe de composants

La Figure 5.4 quant à elle présente 4 instances différentes d'une même classe de composants.

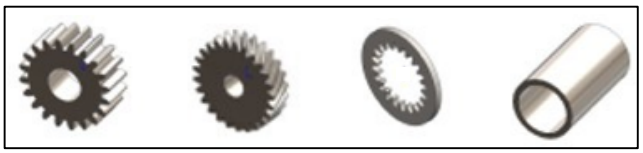

Figure 5.4 : Différentes instances d'une même classe de composants

Après avoir instancié toutes les pièces, l'assemblage des sous-ensembles qui composent le produit ainsi que l'assemblage du produit lui-même est fait. Il passe par l'instanciation des classes de liaisons en respectant les contraintes que doivent respecter les composants, les sous-ensembles et le produit. La modélisation des classes de liaisons est décrite dans la Section 5.1.2.2.

5.1.2.2. Modélisation sémantique conceptuelle des liaisons en CAO-SC

Nous avons proposé une description et une représentation des liaisons mécaniques, des liaisons fonctionnelles et des échanges de flux entre composants de produits mécatroniques [DSMConf] (Figure 5.5). Nous proposons, ici, une méthodologie de caractérisation et de représentation de ces liaisons dans un système de CAO-SC.

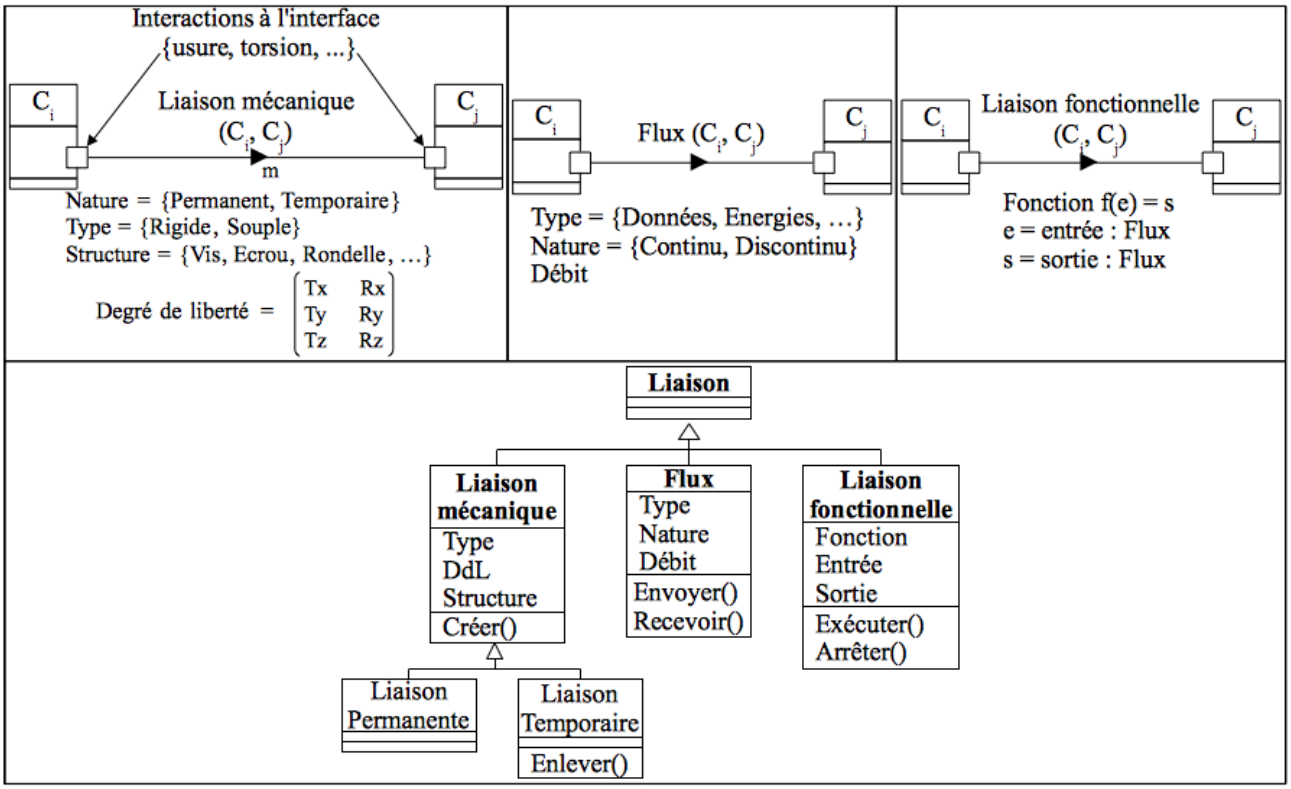

Figure 5.5 : Caractérisation des liaisons entre composants d'un produit mécatronique

Dans les logiciels de CAO commerciaux, les liaisons ne sont pas bien caractérisées. Une bonne partie des informations nécessaires à la description des liaisons n'est pas prise en compte par ces systèmes. Ainsi, ces systèmes ne permettent pas une représentation complète d'un produit. Pendant la création du modèle conceptuel, on crée en même temps que les classes de composants, toutes les classes de liaisons possibles entre ces composants. Chaque classe de liaisons pourra alors être instanciée. Il faut cependant noter qu'il y a une vérification à faire si une classe de liaisons doit être instanciée car après avoir choisi des instances de composants, il y a des instances de liaisons qui seront exclues entre eux. Ainsi, si une classe de liaisons L_{xy} relie les classes de composants C_x et C_y et si une instance de C_x est choisie, puis une instance de C_y, certaines propriétés de la classe L_{xy} seront fixées et certaines instances de liaisons exclues. Le système doit à chaque moment vérifier la cohérence du modèle. Ceci, nous amène à parler de la modélisation conceptuelle d'une famille de produits mécatroniques dans la Section 5.1.2.3.

5.1.2.3. Modélisation sémantique conceptuelle d'une famille de produits en CAO-SC

Une famille de produits est définie par un ensemble de classes de composants et un ensemble de classes de liaisons. Une instance d'une telle famille est une combinaison d'instances de composants et d'instances de liaisons pour les lier. Ces composants pouvant coopérer pour la satisfaction des exigences fonctionnelles et structurelles en remplissant les exigences comportementales. Pour la

représentation d'une famille de produits, le système de CAO-SC permet la représentation des classes de composants et de liaisons et la celle des instances tout en gérant leur cohérence. Pour ce faire le système de CAO-SC permet de :

1. d'instancier les classes de composants :
   a) saisi des valeurs pour les différentes propriétés de chaque classe de composants ;
   b) importation du modèle géométrique correspondant dans une bibliothèque de composants ou sa conception par le concepteur ;
2. d'instancier les classes de liaisons :
   a) saisi des valeurs des propriétés de chaque classe de liaisons ;
   b) vérification de la conformité de la liaison choisie par rapport aux composants à assembler ;
   c) assemblage des composants selon les liaisons choisies.
3. Vérification de la cohérence de l'assemblage (du modèle obtenu)

Un système de CAO implémentant les fonctionnalités décrites ci-dessus doit pouvoir importer des composants dans des bibliothèques préexistantes. Ainsi, l'interopérabilité entre ces systèmes et les bibliothèques de composants PLib est traitée dans la Section 5.1.3.

5.1.3. Interopérabilité PLib, STEP et CAO-SC

Pour représenter les modèles sémantiques conceptuels, un système de CAO-SC est couplé avec un système de gestion de bases de données relationnelles objets (SGBDRO) pour la sauvegarde des données de produits. La transformation du format des données issues du SGBDRO en un format acceptable par le système de CAO (format STEP) est nécessaire. Inversement, les données issues du logiciel de CAO (en format STEP) seront mises au format accepté par le SGBDRO (script SQL 3). L'API permettant cette conversion utilise les fichiers STEP venant du logiciel de CAO et transforme leur contenu en un script SQL 3 pour dialoguer avec le SGBDRO. A l'inverse, les données issues du SGBDRO seront empaquetées dans un fichier STEP (Figure 5.7). D'autres données proviennent de bibliothèques de composants PLib préexistantes ou de bibliothèques propres aux concepteurs.

Nous parlons dans la Section 5.1.3.1 de parties de la norme STEP qui seront utilisées en CAO-SC et dans la Section 5.1.3.2 de parties de la norme PLib.

5.1.3.1. STEP (ISO 10303) pour la CAO-SC

STEP est une norme standardisée ISO pour l'échange de donnée de produits. Il est largement utilisé par les logiciels de CAO classiques actuels. Dans le but de le rendre ouvert et permettre son utilisation par des concepteurs de différents domaines sous différentes plateformes, les initiateurs de

STEP y ont introduit la notion de ressources intégrées. Ces dernières fournissent des informations génériques, indépendantes du contexte, qui peuvent être utilisées par plusieurs applications. Ils définissent deux types de ressources intégrées [El Khalkhali, 2002] :

- ***les ressources intégrées génériques*** qui sont composées d'un ensemble de modèles de données génériques, indépendants du type de produit, du type d'application et du processus de fabrication.
- ***les ressources intégrées d'application*** qui permettent de définir les données applicatives qui ne sont utilisées que pour un domaine applicatif particulier ou à une série d'applications similaires.

Cependant, du fait que la caractéristique de chaque métier est d'avoir élaboré son propre savoir-faire par spécialisation du savoir-faire commun aux différents métiers techniques, il est constaté que les approches génériques ne suffisent pas. D'un métier à l'autre, le langage et/ou les objets utilisés peuvent sensiblement être différents. Il est donc nécessaire de pouvoir prendre en compte ces différences. La solution apportée est la création de ***protocoles d'application*** dans STEP, de façon à intégrer le mieux possible les besoins des utilisateurs. Plusieurs protocoles d'application sont donc développés, chacun d'eux gérant un domaine spécifique souvent assez vaste et complexe. Les plus utilisés dans le domaine de la CAO sont les protocoles d'application 203 et 214 (AP-203 et AP-214). L'AP-203 définit un modèle de données pour la mise en œuvre du système de gestion de données techniques couplé à des systèmes CAO dans les industries mécaniques. L'AP-214, quant à lui, propose une définition des données nécessaires pour représenter des données aussi complexes que les véhicules automobiles tout au long de leur cycle de vie. La mise en œuvre de tels protocoles d'application peut être très difficile principalement dans le cas où le domaine couvert est large et/ou complexe. Dans le but de faciliter leur implémentation, ces protocoles d'application font l'objet d'un découpage en sous-domaines distingués en fonction des types de fonctionnalités recherchées. Ces sous-domaines sont appelés ***classes de conformité***. Une classe de conformité pouvant être définie comme le regroupement d'une ou plusieurs ***Unités de Fonctionnalité*** (***UoF***). Une UoF étant un regroupement d'objets participant à une fonction donnée dans le domaine d'application. La Figure 5.6 donne un récapitulatif de ces principes fondamentaux définis par la norme STEP.

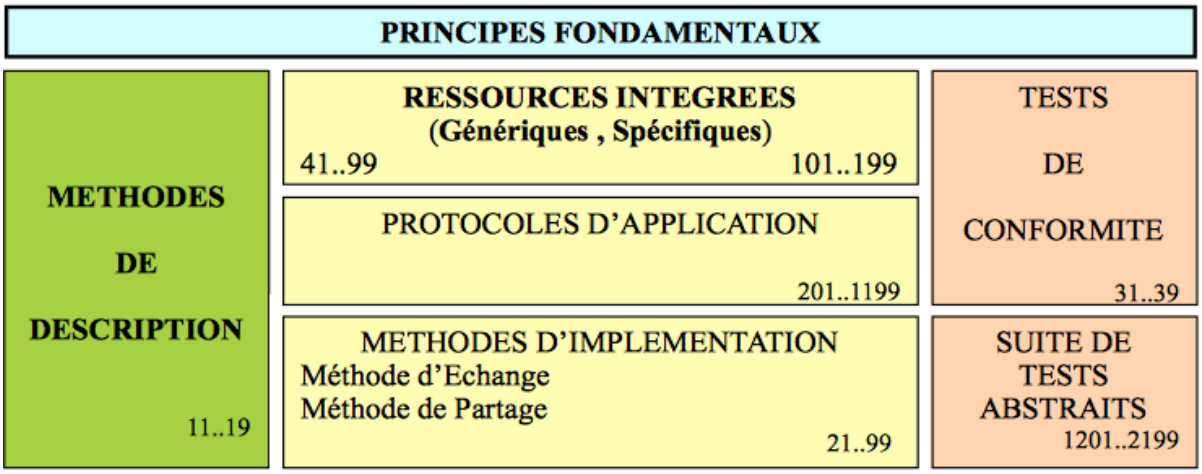

Figure 5.6 : Principe fondamentaux de la norme STEP [Plantec]

5.1.3.1.1. Le langage Express ISO 10303-11

Pour permettre aux concepteurs de décrire et échanger leurs modèles STEP utilise un langage appelé Express. Ce langage décrit dans la partie 11 de la norme STEP (ISO 10303-11) a pour objectif principal la description de modèles d'information dans le domaine technique, en vue de l'échange de données qui les représentent de façon fiable et non ambiguë [Bouazza, 1995] [ISO, 10303-11:1994]. Dans ce langage, l'accent est surtout mis sur la précision du modèle et particulièrement sur les contraintes que doivent respecter les données pour être acceptées comme conforme au modèle. Pour résoudre le problème des anomalies observées lors des transferts de données, identifier les responsabilités entre l'émetteur et le récepteur, voire les erreurs ou imprécisions de la spécification elle-même, la norme STEP définit pour chaque AP :

- Comment doivent être testées les interfaces ?
- Sur quels critères établir leur conformité ?

5.1.3.1.2. Le SDAI ISO 10303-22

Il est défini par la partie 22 de la norme STEP (***ISO 10303-22***) sous forme d'une API pour opérer sur des données d'application spécifiées par un schéma sous Express. Il spécifie aussi comment organiser les données des applications dans un environnement distribué, y compris la validation, le contrôle des transactions et les métadonnées. C'est une spécification abstraite sur la façon de traiter les données définies dans Express et sur la façon de les organiser. Il est défini indépendamment d'un langage de programmation spécifique. Cependant, il peut être mappé à divers langages de programmation et plateformes à travers des bindings de langage qui permettent l'utilisation d'une bibliothèque logicielle dans un autre langage de programmation que celui avec lequel elle a été développée. Le JSDAI (Java™ language binding of the SDAI ***ISO 10303-27***) est défini pour prendre en compte le langage de programmation Java. Il donne des classes et interfaces Java pour les entités

Express et des méthodes pour accéder à leurs attributs.

Le JSDAI est une API pour la lecture, l'écriture et la manipulation de données orientées objets définies par un modèle de données basé sur Express [JSDAI]. C'est une implémentation complète de l'***ISO 10303-22*** et l'***ISO 10303-27*** avec la licence Open Source ***AGPL*** (**GNU A**ffero **G**eneral **P**ublic **L**icense). Il supporte en parallèle ***early binding*** et ***late binding*** pour les applications. ***Early binding*** fait la conversion d'un schéma Express en classes et interfaces Java. Pour chaque entité Express, une classe et une interface Java dédiées sont disponibles ainsi que des méthodes dédiées pour chacun de ses attributs. Cependant ***late binding***, ne convertit la spécification Express qu'en une population de packages jsdai.dictionary. Il contient des classes Java génériques pour tout type d'entité Express et des méthodes pour accéder aux valeurs d'attributs. En plus JSDAI est capable d'exécuter des expressions, des fonctions et des procédures définies dans des schémas Express et effectuer une validation sur les instances de données en tenant compte de toutes les règles locales et globales définies dans Express.

Pour la gestion des composants des produits, une norme nommée PLib (ISO 13584) est définie.

5.1.3.2. PLib (ISO 13584) pour la CAO-SC

La norme PLib (ISO 13584) est destinée à permettre la modélisation, l'échange et le référencement à partir de STEP de bibliothèques informatisées de composants ou d'objets techniques préexistant [Pierra, 1999]. Elle fournit une représentation de données de bibliothèques de composants et décrit les mécanismes et définitions nécessaires à l'échange, l'utilisation et la mise à jour de ces données. L'échange peut avoir lieu entre divers systèmes informatiques et environnements associés au cycle de vie complet des produits dans lesquels des composants de bibliothèque peuvent être utilisés y compris la conception, la fabrication, l'utilisation, la maintenance et l'élimination de produits. Elle va au-delà de STEP et considère un composant comme étant un ensemble de propriétés spécifiques de chaque famille d'objets et qui représente aussi bien la nature que la fonction des composants de la famille. Elle permet de répondre aux questions :

1. Pourquoi tel composant a été choisi plutôt que tel autre ?
2. Quels ont été les critères ou les propriétés qui ont amené à le choisir ?

5.1.3.2.1. La notion d'ontologie dans PLib

Il existe plusieurs définitions possibles du mot ontologie selon la spécialité considérée. PLib définit une ontologie comme étant une collection de descriptions explicites, complètes et consensuelles de concepts dans le contexte le plus large où ces concepts ont un sens précis et sans aucune restriction ou règle correspondant à une utilisation particulière. Elle permet de définir les

familles de composants et les types de propriétés qui leur sont associés. Pour résoudre les différences de nominations d'un même composant qui peuvent exister entre fournisseurs et utilisateurs (concepteur de produits), entre des techniciens de domaine différent (mécaniciens, informaticiens, électroniciens, etc.) un schéma universel de nomination (ISO 13584-42:1998) est défini. Cependant, elle ne permet ni de sélectionner un composant particulier dans une famille de composants en calculant toutes les valeurs de ses propriétés, ni d'en générer des représentations. Ce domaine correspond à la notion de bibliothèque.

5.1.3.2.2. La notion de bibliothèque de composants dans PLib

La conception d'un produit complexe fait souvent appel à des composants préexistants. Le choix de ces composants revient au concepteur qui répertorie d'abord les composants dont il a besoin et les exigences à remplir pour chacun d'eux. Les fournisseurs de composants mettaient à la disposition de leurs clients des catalogues papiers dans lesquels leurs produits étaient décrits mais le besoin d'automatiser ces catalogues s'est fait sentir. La norme PLib apporte un mécanisme permettant de le faire indépendamment des fournisseurs et des systèmes informatiques en introduisant la notion de bibliothèques de composants [Sardet]. Une bibliothèque de composants doit, en particulier, contenir les informations permettant à un concepteur de mettre en œuvre un processus de sélection d'un composant technique (répondant à des exigences particulières). Les exigences à remplir par un composant peuvent porter sur sa structure (matière utilisée pour sa construction, diamètre, poids…), les contextes de son utilisation (charge à supporter, durée de vie), les résultats à produire, etc.

Dans la norme PLib, les composants sont classés en familles de composants. Pour ce faire, le concept de généralisation/spécialisation du paradigme objet est utilisé pour hiérarchiser les composants sous forme de classes. Le choix d'un composant dans un tel modèle se fait alors en suivant les étapes suivantes :

- Choix de la (ou les) famille(s) de composants répondant à la fonction à remplir ;
- Choix, au sein de la (ou les) famille(s), du composants précis qui répond exactement aux exigences du contexte et des résultats à produire.

Pour modéliser les critères de choix, PLib utilise les sélecteurs de classes. Dans la plupart des langages orientés objets (C++, Java,…) il existe la notion de variable de classe appelée en PLib sélecteur de classe. Un sélecteur de classe est une variable particulière qui ne peut prendre qu'une seule valeur pour une classe donnée, et s'applique donc à toutes les instances de la classe où elle est définie. Une telle variable est, comme toutes les caractéristiques définies dans les classes, héritée au travers de l'arbre d'héritage. La matérialisation de cette variable, dans l'arborescence des classes, pour représenter divers points de vue sur une même hiérarchie de classe est la suivante. Un sélecteur

de classes est défini à un certain niveau de la hiérarchie et comme toute variable, il possédera un type, qui est ici un type énuméré. Le nombre de valeurs de ce type dépend de la cardinalité du point de vue. Enfin, cette variable sera redéfinie comme possédant une valeur particulière (constante) dans chacune des classes feuilles qui sont des sous-classes de la superclasse où elle est définie.

Une bibliothèque de composants est conceptualisée comme étant un ensemble de familles de composants organisées entre elles selon des relations de généralisation/spécialisation, chaque nœud de cette hiérarchie factorisant des propriétés destinées à être héritées par ses familles filles.

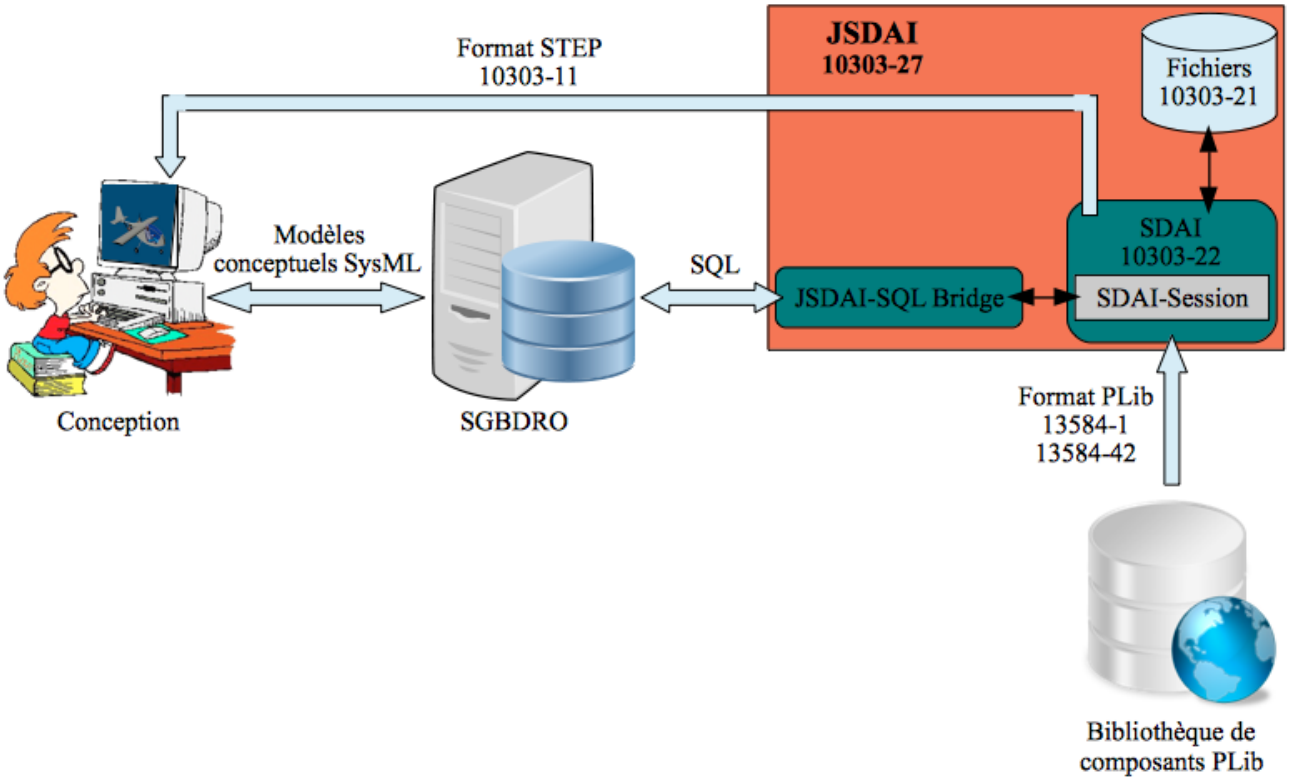

Figure 5.7 : Du modèle conceptuel objet à la modèle CAO-SC

Nous montrons dans la Section 5.2 comment les normes STEP et PLib seront utilisées en CAO-SC. Nous parlons également de leur coopération pour l'exécution des fonctionnalités des systèmes de CAO-SC.

5.2. Utilisation de STEP, PLib et les BDRO en CAO-SC

Les composants et liaisons constituant un produit sont représentés par des classes d'objets dans les modèles conceptuels objets SysML et X-CDSM [ComInd]. Nous pouvons les représenter et les sauvegarder dans une base de données relationnelle objet (BDRO). Par l'intermédiaire de l'API JSDAI, le contenu de la base de données peut être exprimé en langage Express dans un fichier au format STEP pour être utilisé par un système de CAO-SC. L'outil de CAO-SC pourra alors lire et

représenter le contenu du fichier STEP sous forme de modèle géométrique 3D. Ce même processus peut être utilisé entre les données des composants représentées dans des bibliothèques de composants PLib, la base de données et les systèmes de CAO-SC.

En outre, les normes PLib (ISO 13584) et STEP (ISO 10303) sont développées par le même comité ISO TC184/SC4. Elles partagent une technologie commune basée sur l'utilisation du langage de spécification de données Express, le format du fichier physique STEP (ISO 10303-21) et l'interface standard d'accès aux données SDAI. Ces points communs facilitent le déploiement conjoint de STEP comme standard pour la représentation et l'échange de données de produits et PLib comme standard pour la représentation et l'échange de bibliothèques de composants. La coopération entre STEP et PLib en CAO-SC comporte deux aspects essentiels :

- STEP est utilisé pour fournir les représentations des composants définis dans PLib ;
- PLib est utilisé pour référencer les données de produits de STEP pour éviter la duplication des informations.

Ainsi, les outils suivants seront utilisés en CAO-SC :

- STEP ISO 10303-11 (Express) : Pour la description des modèles d'information ;
- STEP ISO 10303-21 (Fichier physique) : Pour le format d'échange de données ;
- STEP ISO 10303-22 (SDAI) : Pour API permettant de passer de Express à un autre langage ;
- STEP ISO 10303-27 (JSDAI) : Pour mapper le SDAI au langage JAVA ;
- PLib ISO 13584-1 (Principes fondamentaux) : Pour représentation des données des bibliothèques de composants et la description des mécanismes et définitions nécessaires à l'échange, l'utilisation et la mise à jour de ces données ;
- PLib ISO 13584-42 (Schéma Universel de Nomination) : Pour la structuration de familles de pièces et la nomination universelle des composants ;

Cette liste non exhaustive donne des parties de STEP et de PLib qui seront utilisées en CAO-SC.

Les Sections 5.1 et 5.2 ont permis de voir les fonctionnalités supplémentaires que la CAO-SC apporte à la CAO classique ainsi que l'utilisation de normes STEP et PLib en CAO-SC. Nous faisons une brève comparaison, dans la Section 5.3, entre ces types de CAO.

5.3. CAO classique Versus CAO-SC

La CAO classique regroupe les fonctionnalités essentielles suivantes :

- la modélisation numérique ;
- la simulation mécanique et le calcul de structure de matériaux ;

- la représentation graphique ;
- le dessin de plan ;
- la manipulation d'objets 3D ;
- la gestion de grands assemblages.

Tableau 5.1 : CAO classique Vs CAO-SC

Fonctionnalité	CAO classique	CAO-SC
Modélisation de produits	Modélise une famille de produits suivants différentes configurations	Modélise plusieurs familles de produits et plusieurs versions de chaque produit
Modélisation de composants	Modélisation basée sur l'historique des fonctions ou modélisation directe	Modélisation basée sur l'historique des fonctions ou modélisation directe ; Représentation sous formes de classes d'objets ; Exportation du modèle d'un composant dans une bibliothèque existante.
Modélisation des liaisons entre composants	Liaisons physiques uniquement	Liaisons physiques, liaisons fonctionnelles, échanges de flux ; Représentation sous formes de classes d'objets.
Modélisation objet des composants	Non	Oui
Modélisation objet des liaisons	Non	Oui
Assemblage d'un produit	Oui	Oui
Interface avec PLib	Non	Oui

Conclusion

Ce chapitre a fait la présentation des fonctionnalités d'une nouvelle génération de CAO appelée CAO sémantique conceptuelle (CAO-SC). Les systèmes implémentant une telle CAO permettent représenter des modèles sémantiques conceptuels et de les instancier au besoin. L'utilisation des normes STEP et PLib par ces systèmes est expliquée. En particulier les parties de ces normes qui seront utilisées et leurs rôles dans le fonctionnement de ces systèmes.

Dans nos futurs travaux, l'architecture et l'implémentation de ces systèmes seront étudiées en collaboration avec des sociétés de conception de systèmes de CAO. L'intégration du processus d'ingénierie de performances comportementales, développé dans le Chapitre 6, dans de tels systèmes

ou l'ajout du logiciel Product-BPAS traité dans la quatrième partie de ce mémoire sous forme de plugin dans de tels système seront également étudiés.

La Figure 5.8 donne le positionnement des différentes étapes des méthodologies proposées dans cette deuxième partie (Conception Sémantique Conceptuelle et Modélisation Sémantique Conceptuelle) dans le cycle en V.

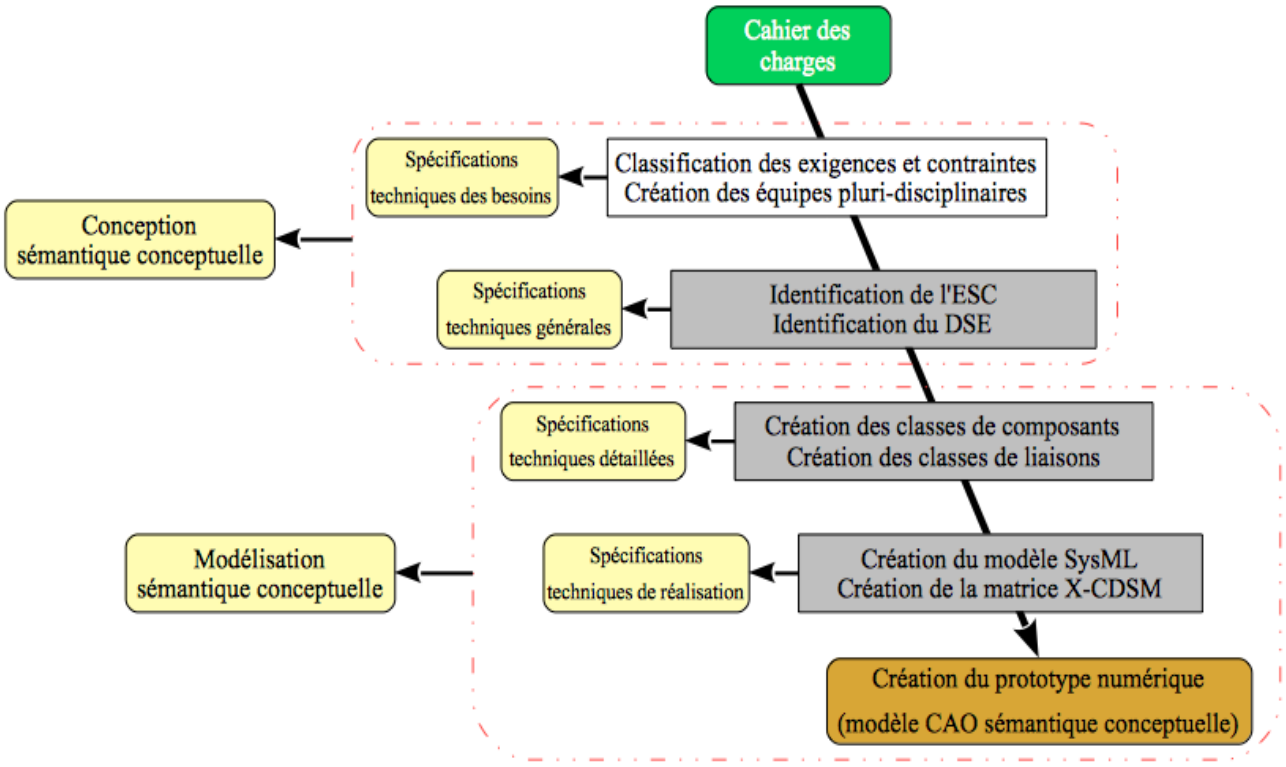

Figure 5.8 : Positionnement des étapes de la CSC et de la MSC dans le cycle en V

Troisième partie : Ingénierie de performances comportementales

Chapitre 6 : Ingénierie de performances comportementales

Chapitre 6 : Ingénierie de performances comportementales

Chapitre 6

Ingénierie de performances comportementales

Introduction

L'un des principaux objectifs de cette étude est de permettre aux concepteurs de fournir des produits innovants, performants et répondant aux exigences des utilisateurs. Ceci passe par l'évaluation des performances comportementales des produits en phase de conception préliminaire. L'évaluation des performances comportementales permet également de concevoir des produits dans les meilleurs délais et à moindre coût. Plusieurs travaux de recherches ont porté sur cette problématique ces dernières années. Des métriques d'évaluation de domaines comportementaux tels que la fiabilité, la maintenabilité, la disponibilité, la sécurité et la recyclabilité pouvant être évaluées en phase précoce de conception sont proposées et des formules pour les calculer sont fournies. Cependant, avec la nouvelle méthodologie de modélisation de familles de produits complexes basée sur les matrices X-CDSM structurelles et fonctionnelles que nous avons proposée dans le Chapitre 4 il est nécessaire d'améliorer ces formules pour tirer profit des avantages qu'offre ce type de modélisation. Il est également nécessaire de pouvoir identifier parmi les instances des familles de produits, modélisées par une MSX-CDSM, celles qui répondent aux exigences structurelles, fonctionnelles et comportementales données dans le cahier des charges.

Ainsi, dans ce chapitre, nous proposons une méthodologie de recherche du chemin optimal de désassemblage (Section 6.1) [CIRP LCE], une amélioration de la formule de calcul du temps de remplacement de composants défaillants (Section 6.2) [IAMOT] et de celle de la fiabilité basée sur les exigences fonctionnelles [SysCon] représentées par une matrice X-CDSM fonctionnelle (Section 6.3). Le processus d'ingénierie de performances comportementales est décrit dans la Section 6.4.

6.1. Recherche de chemin optimal de désassemblage

Le désassemblage d'un produit multi-composant est le démontage des liaisons utilisées pour assembler ses composants (pièces et sous-ensembles de pièces). Ce désassemblage peut concerner une partie des composants du produit (**désassemblage incomplet**) ou tous ses composants (**désassemblage complet**). Dans le cas du recyclage des composants d'un produit en fin de vie, son désassemblage est généralement complet. Cependant, en cas de maintenance (remplacement d'un composant défaillant) seuls les composants permettant de l'atteindre sont désassemblés. Ce désassemblage est non seulement incomplet mais **sélectif**. Il mène au choix d'une séquence de désassemblage optimisée en fonction de ce composant unique [Julie, 2009]. Au lieu de reposer sur

un seul composant, le **désassemblage partiel** permet plutôt d'identifier la meilleure séquence de désassemblage en fonction des limites de faisabilité, c'est-à-dire la meilleure séquence et le meilleur nœud d'arrêt pour un produit donné [Julie, 2009]. Ainsi, identifier le chemin optimal de désassemblage et calculer le temps nécessaire pour le faire sont utiles aussi bien pour évaluer la maintenabilité que pour évaluer la recyclabilité. On parle de la dés-assemblité dans la Section 6.1.1, une amélioration de la formule de calcul de l'indicateur de désassemblage est donnée dans la Section 6.1.2 et la recherche du chemin optimal de désassemblage est traitée dans Section 6.1.3.

6.1.1. Dés-assemblité d'un produit mécatronique

Dans ces travaux, nous considérons qu'une séquence de désassemblage est une succession de liaisons mis dans le bon ordre pour désassembler un produit complexe. Avant de passer au désassemblage d'un produit, il faut d'abord évaluer sa dés-assemblité. Dans leurs travaux, Chekh-Waiss et. al ont défini des indicateurs permettant d'évaluer la dés-assemblité d'un produit qu'ils ont définie comme étant "l'aptitude d'un produit à être facilement désassemblé" [Chekh-Waiss, 2013]. Ils ont également proposé un indice de dés-assemblité (ID) et une échelle permettant de classer les produits (Tableau 6.1).

Tableau 6.1 : Seuils de satisfaction de l'indice de dés-assemblité

0	25%	50%	75%	100%
Mauvaise	Pas satisfaisante	Satisfaisante	Bonne	

L'évaluation de la dés-assemblité est un préalable à l'identification de la séquence optimale de désassemblage. C'est à l'issue de cette phase que l'on décide si oui ou non on passe à l'identification de la séquence optimale. Nous considérons que si la dés-assemblité est satisfaisante ou bonne, on peut alors passer à l'étape suivante (identification de la séquence optimale), sinon, le processus s'arrête. Pour rechercher la séquence optimale de désassemblage, nous proposons une amélioration de la méthode de calcul du temps de désassemblage des produits complexes (Section 6.1.2) et l'utilisation des réseaux de Pétri pour identifier le chemin optimal.

6.1.2. Calcul du temps de désassemblage

Le temps de désassemblage d'un produit est souvent calculé en faisant la somme des durées (ou difficulté) de démontage de chaque liaison participant à son assemblage. En phase de conception, on donne un poids q, indiquant le niveau de difficulté pour enlever une liaison. Ainsi, à chaque type de liaison L_i on attribue un poids q_i. Plus la liaison est difficile à enlever, plus son poids est élevé. Le Tableau 4.1 donne les valeurs des q_i selon les types de liaison. Avec ces poids, le temps nécessaire

pour désassembler un produit est donné par l'Equation 6.1 [DSMConf] :

$$R_d = \sum_{i=1}^{n} m_i * q_i \qquad (6.1)$$

Dans cette équation m_i est le nombre d'occurrences de la liaison L_i et n le nombre de liaisons à enlever. Pour améliorer cette formule, en plus des poids attribués aux types de liaison nous prenons en compte l'accessibilité et la position des liaisons dans le calcul de R_d dans la Section 6.1.2.1.

6.1.2.1. L'accessibilité et la position d'une liaison

L'accessibilité d'une liaison traduit le niveau de difficulté pour atteindre la liaison et l'enlever. Elle change au fur et à mesure que le désassemblage avance. Une liaison non accessible ou difficilement accessible peut devenir accessible durant le désassemblage (après avoir enlevé des composants). Ainsi, dans le calcul du temps de désassemblage on considère l'accessibilité d'une liaison au moment où elle est la prochaine liaison à enlever dans la séquence. La position d'une liaison est donnée par le nombre de composants qui séparent celui à partir duquel elle doit être enlevée et celui à partir duquel la liaison précédente est enlevée. Alors la position d'une liaison dépend de celle de la liaison qui la précède dans la séquence.

Ainsi, dans la formule de calcul du temps de désassemblage, la position de la prochaine liaison est donnée par *p* et son niveau d'accessibilité est donné par *a*. Les valeurs des paramètres *p* et *a* sont obtenues comme suit :

- Si la liaison suivante dans la séquence peut être enlevée à partir :
  - ✓ du composant actuel, le paramètre position prend la valeur $p = 0$;
  - ✓ d'un composant voisin de l'actuel, *p* prend la valeur $p = \mu$;
  - ✓ du nième composant à partir du composant actuel, *p* prend la valeur $p = n*\mu$.
- Si la liaison suivante dans la séquence est :
  - ✓ facilement accessible, le paramètre accessibilité prend la valeur 1 ($a = 1$) ;
  - ✓ difficile à atteindre, *a* prend la valeur ß ($a = ß$). Nous pouvons avoir plusieurs niveaux de difficulté selon la structure du produit.

L'indicateur de désassemblage (D_d) est calculé en fonction des paramètres *q*, *p* et *a* comme le montre l'Equation 6.2 :

$$D_d = \sum_{i=1}^{n} m_i * a_i * q_i + \sum_{j=2}^{n} p_j \qquad (6.2)$$

Dans cette formule, pour chaque liaison L_i on multiplie son poids q_i par son accessibilité a_i et son nombre d'occurrences m_i. Ainsi, le temps que dure l'enlèvement d'une liaison est proportionnel à son accessibilité. Après avoir enlevé une liaison, on ajoute le coefficient de position de la liaison suivante

s'il y en a.

Il faut cependant constater qu'il peut arriver que plusieurs liaisons puissent être enlevées simultanément. La prise en compte de cette éventualité, traitée dans la Section 6.1.2.2, permet de diminuer le temps de désassemblage.

6.1.2.2. Les sous-séquences simultanées

Pendant le désassemblage, il est possible que certaines liaisons qui se suivent dans la séquence puissent être enlevées simultanément. On appelle un tel ensemble de liaisons une sous-séquence simultanée (SSS). On suppose, dans la suite, que les outils nécessaires sont disponibles. On note t_i le temps qu'il faut pour enlever la liaison L_i et λ_j le temps qu'il faut pour enlever les x liaisons de la SSS notée S_j. Dans ce cas si :

- x personnes sont disponibles, alors le temps nécessaire pour enlever les x liaisons est égal au plus grand temps parmi ceux des x liaisons.

$$\lambda_j = max(t_1, t_2, \ldots, t_x) \tag{6.3}$$

- y personnes (y < x) sont disponibles, on a :

$$\lambda_j = \sum_{i=1}^{\lfloor \frac{x}{y} \rfloor} max(t_{y*(i-1)+1}, \ldots, t_{y*i}) + max(t_{y*\lfloor \frac{x}{y} \rfloor+1}, \ldots, t_x) \tag{6.4}$$

D'après l'équation 6.3, on a :

$$t_i = m_i * a_i * q_i \tag{6.5}$$

Dans une séquence S où il y a s SSS notées S_1 à S_s de temps λ_1 à λ_S et r autres liaisons de temps t_1 à t_r et n'appartenant à aucune SSS, l'indicateur de désassemblage (D_d) est calculé comme suit :

$$D_d = \sum_{i=1}^{s} \lambda_i + \sum_{j=1}^{r} t_j + \sum_{k=2}^{s+r} p_k \tag{6.6}$$

Si :

- n est le nombre de liaisons de la séquence S ;
- n_i est le nombre de liaisons de la SSS notée S_i ;
- p_k est la position de la prochaine liaison L_k dans la séquence ou celle de la liaison ayant la plus petite valeur de p parmi les liaisons de la SSS qui vient après la dernière liaison (SSS) enlevée dans la séquence S.

Alors, la valeur de r est obtenue avec l'Equation 6.7 :

$$r = n - \sum_{i=1}^{s} n_i \tag{6.7}$$

Pour une SSS donnée la valeur de λ_i dépend du nombre de personnes et de l'ordre des liaisons. Si

le nombre de liaisons est inférieur ou égal au nombre de personnes, la valeur de λ_i est égale à la plus grande valeur parmi les t_i. Dans ce cas l'ordre des liaisons n'est pas important.

Dans la Section 6.1.3, nous utilisons les réseaux de Pétri dans lesquels les paramètres ci-dessus sont pris en compte pour chercher le chemin optimal de désassemblage.

6.1.3. Recherche du chemin optimal de désassemblage

Le modèle sémantique conceptuel X-CDSM contient toutes les classes de composants et toutes les classes de liaisons qui définissent une famille de produits [DSMConf]. Elle permet ainsi de connaître le sens, le nombre d'occurrences, le type, etc. de chaque liaison entre deux composants. Elle contient les informations permettant de construire toutes les séquences de désassemblage possibles avant d'en extraire celle qui est la plus optimale. L'identification de la séquence optimale de désassemblage est faite en 3 étapes à partir de la matrice X-CDSM (Figure 6.1) :

1. Choisir le niveau de désassemblage ;
2. Construire le réseau de Pétri correspondant ;
3. Identifier la séquence optimale.

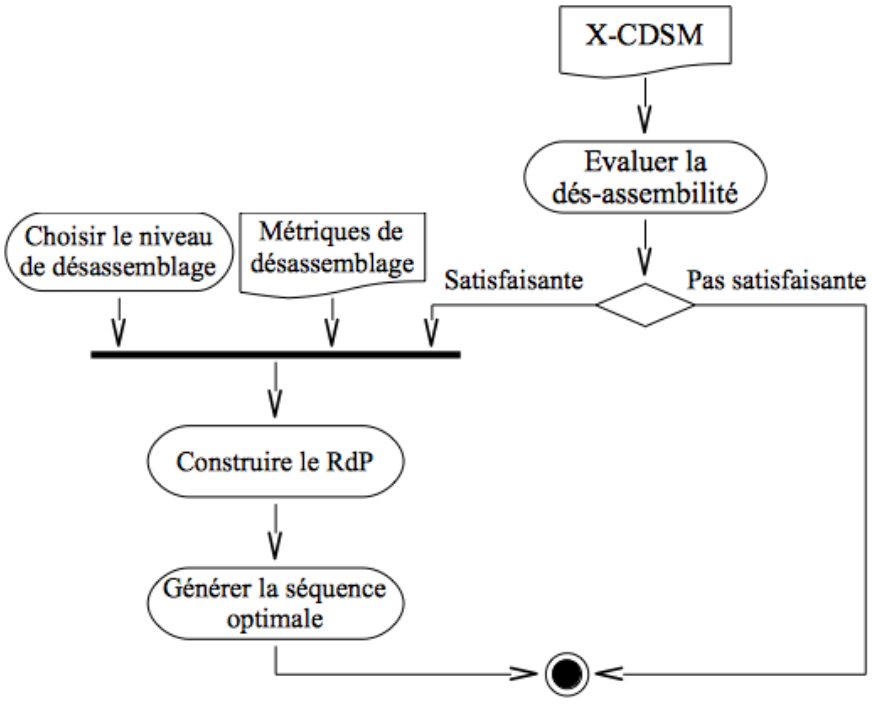

Figure 6.1 : Processus d'identification de la séquence optimale

Le choix du niveau de désassemblage étant un préalable à la recherche du chemin optimal de désassemblage, nous parlons dans la Section 6.1.3.1 de la notion de niveau dans un diagramme des composants d'une famille de produits mécatroniques.

6.1.3.1. Niveau de désassemblage

Un produit complexe peut avoir plusieurs niveaux de désassemblage (Figure 6.2). Le niveau le plus élevé étant celui où toutes les pièces élémentaires sont séparées les unes aux autres. Le niveau le plus bas est celui où seuls les sous-ensembles de niveau 1 sont séparés. Ainsi Pour désassembler un produit complexe, il est obligatoire de choisir un niveau de désassemblage.

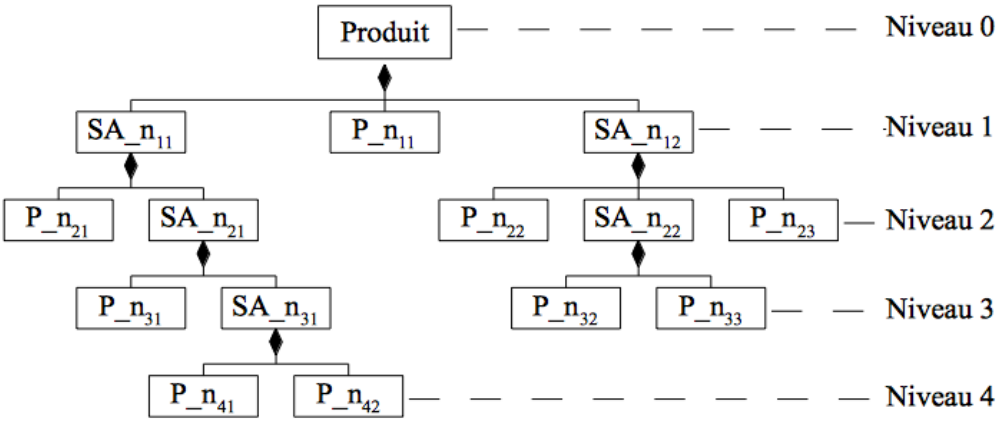

Figure 6.2 : Différents niveaux d'un produit complexe

Ce choix permettra de construire le réseau de Pétri de la famille de produits. La description et la construction du réseau de Pétri sont détaillées dans la Section 6.1.3.2.

6.1.3.2. Construction du RdP d'une famille de produits mécatroniques

Le réseau de Pétri d'une famille de produit complexes pour l'identification de la séquence optimale de désassemblage de ses instances est défini comme suit :

- Les classes de composants sont représentées par les places du RdP ;
- Les classes de liaisons sont représentées par les transitions du RdP ;
- Le sens d'une liaison est donné par le sens de l'arc reliant la place à la transition ;
- Chaque place P_i représentant une classe de composants $Class_i$ est caractérisée par les paramètres k_i et r_i représentant la criticité et la fiabilité de la classe $Class_i$ respectivement ;
- Chaque transition T_i représentant une classe de liaisons $Link_i$ est caractérisée par un délai de tir f_i telle que :

$$f_i = t_i = m_i * a_i * q_i \qquad (6.8)$$

Avec

- ✓ m_i = le nombre d'occurrences de $Link_i$;
- ✓ q_i = le niveau de difficulté pour enlever $Link_i$;
- ✓ a_i = l'accessibilité de $Link_i$;

- Si deux transitions T_1 et T_2 sont franchissables à un marquage M_i la transition ayant la plus

petite valeur de f_i sera prioritaire sur l'autre ;

- Pour chaque place ayant au moins une transition de sortie, la somme des valeurs des arcs vers ses transitions de sortie doit être égale à la somme des valeurs des arcs de ses transitions d'entrée à laquelle on ajoute son nombre de marques au marquage initial M_0 ;

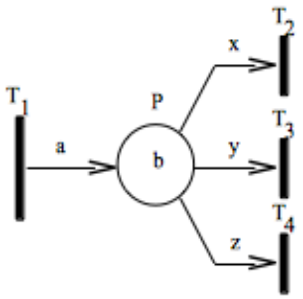

Figure 6.3 : Poids des arcs entrant et sortant d'une place

$$a + b = x + y + z \qquad (6.9)$$

- Selon le niveau de désassemblage, on peut regrouper plusieurs transitions en une seule :
  - ✓ Si le désassemblage doit aller jusqu'au dernier niveau (niveau pièce élémentaire) et si la place P_1 (représentant la classe de composants $Class_1$) est la place de sortie de 3 transitions T_x, T_y et T_z (représentant les classes de liaisons $Link_x$, $Link_y$ et $Link_z$ respectivement) ayant pour places d'entrée les places P_x, P_y et P_z (représentant les classes de composants $Class_x$, $Class_y$ et $Class_z$ respectivement), on crée une transition T_j telle que :
    - Post(P_1, T_j) = 1 ;
    - Pré(P_x, T_j) = 1 ;
    - Pré(P_y, T_j) = 1 ;
    - Pré(P_z, T_j) = 1 ;

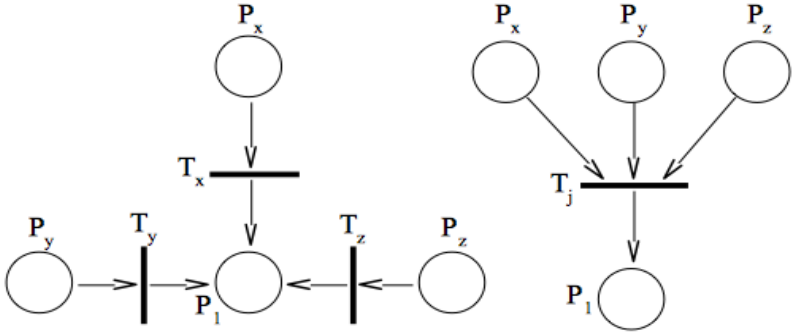

Figure 6.4 : Regroupement de transitions

  - ✓ Si le désassemblage doit s'arrêter à un niveau intermédiaire N_j, les classes de liaisons reliant les sous-ensembles de ce niveau ou de niveau supérieur doivent être défaites. On ne peut donc pas regrouper une transition représentant une de ces classes de liaisons avec une autre représentant une classe de liaisons interne à ce niveau ou à un niveau inférieur ;
- Si plusieurs transitions successives dans une séquence sont franchissables pour un marquage donné, elles forment une SSS ;

- Le marquage final M_f est obtenu en utilisant la formule suivante :

$$M_f = M_0 + C * S^{\#} \quad (6.10)$$

Avec

- ✓ M_0 le marquage initial ;
- ✓ C la matrice d'incidence du RdP ;
- ✓ $S^{\#}$ la matrice uni-colonne de la séquence S telle que chaque transition soit franchie une et une seule fois.

Pour montrer l'apport des réseaux de Pétri par rapport aux graphes dans la recherche du chemin optimal de désassemblage, nous faisons une comparaison des séquences obtenues avec un RdP et le graphe correspondant dans la Section 6.1.3.3.

6.1.3.3. RdP vs Graphe de liaisons

Le graphe de liaison correspondant au RdP de la Figure 6.5, est donné sur la Figure 6.6.

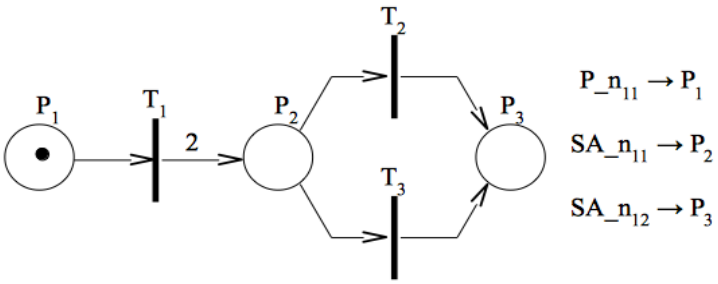

Figure 6.5 : RdP correspondant au niveau 1 de la Figure 6.2

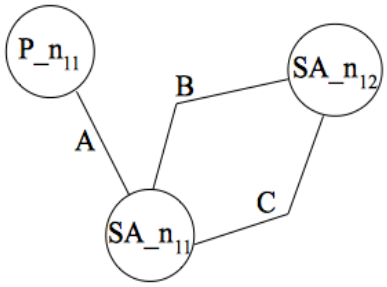

Figure 6.6 : Graphe de liaisons correspondant au niveau 1 de la Figure 6.2

a. Séquences obtenues à partir du graphe

Avec ce graphe de liaisons représenté sur la Figure 6.6, nous avons 6 séquences de désassemblage possibles :

- S_1 = ABC
- S_2 = ACB
- S_3 = BAC
- S_4 = BCA

- S_5 = CAB
- S_6 = CBA

On suppose qu'avant d'enlever la liaison A les liaisons B et C ont une accessibilité plus importante à cause de la présence du composant SE_n_{12}. De plus enlever B (respectivement C) suivie de A suivie de C (respectivement B) engendre des vas et vient entre composants et augmente la valeur de l'indicateur D_d à cause du paramètre position. Il est alors plus judicieux d'enlever d'abord A puis les deux autres (S_1 et S_2) dans le sens que l'on veut. Ce qui revient à choisir les séquences S_1 et S_2.

b) Séquences obtenues à partir du réseau de Pétri

A partir du RdP représenté sur la Figure 6.5, seules les deux séquences S_1 et S_2 sont possibles. L'indicateur de désassemblage pour ces deux séquences est :

$$D_d(S_1) = a_1*m_1*q_1 + p_2 + a_2*m_2*q_2 + p_3 + a_3*m_3*q_3 \qquad (6.11.a)$$

Après avoir défait T_2 on reste sur P_2 pour défaire T_3, alors $p_3 = 0$ et on a :

$$D_d(S_1) = a_1*m_1*q_1 + p_2 + a_2*m_2*q_2 + a_3*m_3*q_3 \qquad (6.11.b)$$

$$D_d(S_2) = a_1*m_1*q_1 + p_3 + a_3*m_3*q_3 + p_2 + a_2*m_2*q_2 \qquad (6.12.a)$$

Après avoir défait T_3 on reste sur P_2 pour défaire T_2, alors $p_2 = 0$ et on a :

$$D_d(S_2) = a_1*m_1*q_1 + p_3 + a_3*m_3*q_3 + a_2*m_2*q_2 \qquad (6.12.b)$$

A partir de P_1, nous avons $p_2 = p_3$ car un seul saut est fait pour atteindre P_2 pour défaire soit T_2 soit T_3, d'où :

$$D_d(S_1) = D_d(S_2) \qquad (6.13)$$

Alors la valeur de D_d est la même que l'on commence par T_2 ou T_3 après avoir enlevé T_1.

Les deux séquences S_1 et S_2 sont donc équivalentes. Mieux, en considérant la SSS constituée par T_2 et T_3 on a :

$$D_d = a_1*m_1*q_1 + p_3 + max(a_3*m_3*q_3, a_2*m_2*q_2) \qquad (6.14)$$

Ceci permet de diminuer le temps de désassemblage de ce système s'il y a assez de personnels et d'outils.

Dans la Section 6.2, nous utilisons les poids des types de liaison (q_i) et les paramètres accessibilité (a_i) et position (p_i) des liaisons pour calculer le temps nécessaire au remplacement d'un composant défaillant.

6.2. Calcul du temps de remplacement de composants défaillants

La maintenabilité définie entre autres comme étant « ***l'ensemble des actions permettant de maintenir ou de rétablir un bien dans un état spécifié ou en mesure d'assurer un service déterminé*** » [NF X 60-010]. Elle englobe alors la maintenance préventive et la maintenance corrective. En phase de conception préliminaire, la métrique de maintenabilité la plus souvent considérée pour évaluer la maintenabilité est le MTTR (Moyenne des Temps Techniques de Réparation) dont la formule est donnée par l'Equation 6.15 :

$$MTTR = T_{Diag} + T_R + T_C \qquad (6.15)$$

Dans cette formule, T_{Diag} est le temps de diagnostic, T_R est le temps de réparation et T_C est le temps de contrôle après réparation.

Dans ces travaux, nous nous intéressons au temps de réparation T_R. Ce temps contient le temps de désassemblage pour atteindre le composant défaillant, le temps de remplacement du composant défaillant et le temps de réassemblage de tous les composants enlevés.

Le temps de remplacement (T_i) d'un composant défaillant (C_i) est la somme de son temps de démontage (T_{di}) et du temps de montage du nouveau composant qui le remplace (T_{mi}) [IAMOT]. Soient :

- E_i de cardinalité g_i l'ensemble contenant C_i et tous les composants à enlever pour l'atteindre ;
- LE_i de cardinalité h_i l'ensemble contenant les liaisons à enlever pour démonter les g_i composants dans E_i ;
- tc_i le temps nécessaire pour enlever les g_i composants dans E_i.

On a alors :

$$tc_i = \sum_{l=1}^{h_i} a_l * m_l * q_l + \sum_{k=2}^{h_i} p_k \qquad (6.16)$$

Soit Ec_i l'ensemble des indices des composants contenus dans E_i, alors :

$$T_{di} = \sum_{j \in Ec_i} tc_j \qquad (6.17)$$

Pour simplifier, nous considérons que le temps de démontage (T_{di}) d'un composant défaillant est égal au temps de montage du composant qui le remplace. Ainsi, on a :

$$T_{di} = T_{mi} \qquad (6.18)$$

$$T_i = T_{di} + T_{mi} = 2 * T_{di} \qquad (6.19)$$

Alors :

$$T_i = 2 * \left(\sum_{j \in Ec_i} tc_j \right) \quad (6.20)$$

D'où :

$$T_i = 2 * \left(\sum_{j \in Ec_i} \left(\sum_{l=1}^{h_j} a_l * m_l * q_l + \sum_{k=2}^{h_j} p_k \right) \right) \quad (6.21)$$

Dans la Section 6.3 nous utilisons les matrices X-CDSM fonctionnelles pour évaluer la fiabilité de produits complexes basée sur la possibilité ou non de fournir une fonctionnalité donnée.

6.3. Evaluation de la fiabilité basée X-CDSM fonctionnelles

La fiabilité est un domaine comportemental très important pour un produit mécatronique. Elle donne la probabilité que le produit puisse exécuter les tâches pour lesquelles il est conçu. Chaque tâche correspond à une exigence fonctionnelle du cahier des charges fonctionnel. Nous utilisons la matrice X-CDSM fonctionnelle décrite dans le chapitre 4 (Section 4.2.3.2) pour l'évaluation de la fiabilité des fonctions et modules d'une famille de produits complexes. Pour évaluer la fiabilité de familles de produits à partir d'une X-CDSM fonctionnelle, nous donnons les formules permettant de calculer la fiabilité :

- d'un module qui est la probabilité que ce module puisse être exécuté correctement ;
- d'une fonction qui est la probabilité que cette fonction puisse être exécutée correctement. Elle correspond également à la probabilité que tous les modules de la fonction puissent être exécutés correctement ;
- d'une famille de produits complexes qui est la probabilité que toutes les fonctions critiques soient correctement exécutées.

Une fonction est considérée comme critique si sa criticité dépasse le seuil de criticité k_0.

Dans la suite de ce chapitre nous considérons que :

- un module ne peut être exécuté que par un et un seul composant ;
- si l'exécution d'un module ne dépend pas de celle d'un autre module et si le composant qui doit l'exécuter fonctionne, alors ce module peut être correctement exécuté ;
- si l'exécution d'un module dépend de celle d'autres modules, il ne peut être exécuté que si le composant qui doit l'exécuter fonctionne et que tous les modules dont il dépend puissent être correctement exécutés ;
- les probabilités de panne des composants sont indépendantes ;
- si un module M_{ij} d'une fonction F_i ne peut pas être exécuté, alors la fonction F_i ne pourra pas

être exécutée non plus ;

- un module M_{ij} d'une fonction F_i peut dépendre d'un ou de plusieurs modules de la même fonction ou de plusieurs fonctions différentes ;

La relation de dépendance entre modules que nous notons $<_d$ est alors définie comme suit :

- Un module ne dépend pas de lui-même : la relation est irréflexive ;
- Si M_1 dépend de M_2 ($M_1 <_d M_2$) alors l'inverse est fausse : la relation est antisymétrique ;
- Si M_1 dépend de M_2 ($M_1 <_d M_2$) et M_2 dépend de M_3 ($M_2 <_d M_3$) alors M_1 dépend de M_3 ($M_1 <_d M_3$) : la relation est transitive ;

La relation est, ainsi, une relation d'ordre strict partiel. Partant des hypothèses données ci-dessus et de l'hypothèse qui en découle (la relation de dépendance est une relation d'ordre strict partiel), nous donnons les formules de calcul de fiabilité des modules et fonctions de produits complexes dans la Section 6.3.1.

6.3.1. Fiabilité d'un module

La fiabilité d'un module est calculée à partir de celle du composant qui doit l'exécuter. Ainsi, si les modules M_{ij} et M_{ls} doivent être exécutés par les composants C_x et C_y respectivement et que nous notons :

- $P(C_x) = f_x$ la fiabilité du composant C_x (la probabilité qu'il fonctionne correctement) ;
- $P(M_{ij}) = r_{ij}$ la fiabilité du module M_{ij} (la probabilité qu'il puisse être exécuté correctement).

La fiabilité de M_{ij} est obtenue comme suit :

1. Si l'exécution du module M_{ij} ne dépend d'aucun autre module, sa fiabilité r_{ij} est égale à celle du composant C_x comme le montre l'Equation 6.22 :

$$P(M_{ij}) = P(C_x) \quad (6.22)$$

2. Si l'exécution du module M_{ij} dépend de celle du module M_{ls} uniquement (Figure 6.7), la fiabilité r_{ij} de M_{ij} est obtenue avec l'Equation 6.23 :

$$P(M_{ij}) = P(C_x) * P(M_{ls}) \quad (6.23)$$

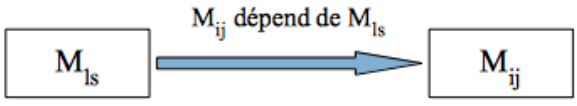

Figure 6.7 : L'exécution du module M_{ij} dépend de celle du module M_{ls}

3. Si l'exécution du module M_{ij} dépend de celles des modules M_{dl} et M_{qr}. Alors, la fiabilité r_{ij} du module M_{ij} est obtenue avec l'Equation 6.24.b :

$$P(M_{ij}) = P(C_x \cap M_{dl} \cap M_{qr}) \quad (6.24.a)$$

$$P(M_{ij}) = P(C_x) * P(M_{dl} \cap M_{qr}) \quad (6.24.b)$$

- Si les modules M_{dl} et M_{qr} sont indépendants l'un de l'autre (Figure 6.8), la probabilité $P(M_{dl} \cap M_{qr})$ est obtenue en faisant le produit de $P(M_{dl})$ et $P(M_{qr})$ comme le montre l'Equation 6.25.a :

$$P(M_{dl} \cap M_{qr}) = P(M_{dl}) * P(M_{qr}) \quad (6.25.a)$$

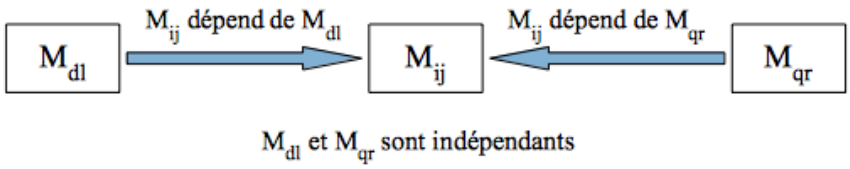

Figure 6.8 : L'exécution du module M_{ij} dépend de celles des modules M_{dl} et M_{qr}

Alors en remplaçant $P(M_{dl} \cap M_{qr})$ par sa formule dans 6.24.b, on obtient l'Equation 6.24.c :

$$P(M_{ij}) = P(C_x) * P(M_{dl}) * P(M_{qr}) \quad (6.24.c)$$

- Si l'exécution de l'un des deux modules dépend de celle de l'autre, la probabilité $P(M_{dl} \cap M_{qr})$ est obtenue avec l'Equation 6.26 en supposant que c'est l'exécution de M_{dl} qui dépend de celle de M_{qr} (Figure 6.9) :

$$P(M_{dl} \cap M_{qr}) = P(M_{dl} / M_{qr}) * P(M_{qr}) \quad (6.25.b)$$

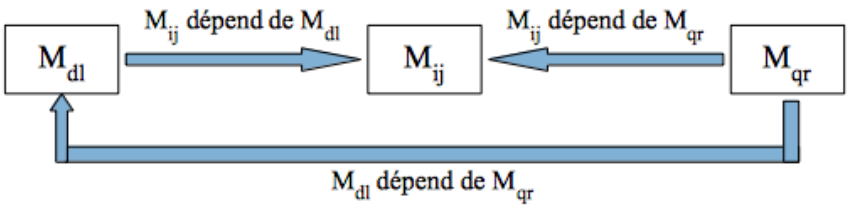

Figure 6.9 : L'exécution du module M_{ij} dépend de celles des modules M_{dl} et M_{qr}

Alors :

$$P(M_{ij}) = P(C_x) * P(M_{dl} / M_{qr}) * P(M_{qr}) \quad (6.24.d)$$

4. Si l'exécution du module M_{ij} dépend de celles de p modules (p > 2) et Q l'évènement tel que les P modules soient exécutés. La même méthode est utilisée pour calculer sa fiabilité. Alors :

$$P(M_{ij}) = P(C_x) * P(Q) \quad (6.25.c)$$

En assumant que :

- Z est l'ensemble contenant les p modules ;
- W est l'ensemble contenant les indices des p modules ;
- les exécutions des p modules sont indépendantes deux à deux.

La formule de P(Q) est donnée par l'Equation 6.27 :

$$P(Q) = \prod_{k \in W} P(C_k) \qquad (6.27)$$

En remplaçant P(Q) par sa formule dans l'Equation 6.25.c, on a l'Equation 6.24.e :

$$P(M_{ij}) = P(C_x) * \prod_{k \in W} P(C_k) \qquad (6.24.e)$$

A partir de la fiabilité de chaque module d'une fonction on calcule la fiabilité de cette fonction comme nous le montrons dans la Section 6.3.2.

6.3.2. Fiabilité d'une fonction

La fiabilité d'une fonction dépend de celles de ses modules. Ainsi, si la fonction F_i a :

1. Un seul module M_{ij}, alors sa fiabilité R_i est égale à celle de M_{ij} comme le montre l'Equation 28.a :

   $P(F_i) = P(M_{ij})$ (6.28.a)

   $R_i = r_{ij}$ (6.28.b)

2. n_i modules ($n_i > 1$) et soient :
   - K l'ensemble contenant les k tels que M_k soit un module de F_i: {k / Mk }
   - J l'ensemble contenant les j tels qu'il existe k dans M_k tel que $M_k <_d M_j$.

   Alors la fiabilité R_i de la fonction F_i est obtenue à partir de l'Equation 6.29 :

   $$P(F_i) = \prod_{j \in J} P(C_j) \qquad (6.29)$$

La fiabilité d'un produit est calculée en utilisant celles de ces fonctions critiques comme nous le montrons dans la Section 6.3.3.

6.3.3. Fiabilité des instances d'une famille de produits complexes

Pour calculer la fiabilité des instances d'une famille de produits complexes, nous cherchons d'abords la fiabilité de chaque fonction critique. D'une manière concrète, une fonction est critique si sa non-exécution entraine l'arrêt du produit ou la non-exécution de fonctions critiques. Si nous assumons que :

- Z_0 est l'ensemble contenant les indices des fonctions critiques ;
- RI_x la fiabilité de l'instance I_x ;
- Les fonctions critiques sont indépendantes.

Alors la fiabilité RI_x d'une instance I_x est donnée par l'Equation 30 :

$$RI_x = \prod_{j \in Z_0} R_j \qquad (6.30)$$

Si nous avons une seule fonction critique, alors la fiabilité du produit est égale à celle de cette fonction.

6.4. Ingénierie de performances comportementales

6.4.1. Qu'est-ce que l'ingénierie de performances comportementales ?

L'ingénierie de performances comportementales est le processus allant de la conception des modèles à l'identification des instances satisfaisant les exigences du cahier des charges. C'est une démarche visant l'identification et l'optimisation d'instances de produits satisfaisant les attentes spécifiées dans le cahier des charges dans une grande masse de produits (familles de produits). Elle est basée sur le triptyque Evaluation-Optimisation-Validation (EOV). Cependant, il est possible d'abandonner des instances qui ne répondent pas aux exigences. Il est également possible, dans le cas où il y a plusieurs instances de produits retenues, de procéder à une comparaison permettant de choisir la meilleure d'entre elles. En effet, parmi les instances retenues, certaines peuvent avoir de meilleures performances (plus fiables, plus sures, ...), ou une meilleure ergonomie, ou un plus faible coût, …

Un algorithme d'ingénierie de performances comportementales permettant à partir d'une MSX-CDSM d'identifier les instances de produits satisfaisant les exigences tout en respectant les contraintes est présenté dans la Section 6.4.2.

6.4.2. Algorithme d'ingénierie de performances comportementales

L'ingénierie de performances comportementales débute par le choix du/des domaine(s) comportemental(aux) (maintenabilité, fiabilité, sécurité, disponibilité, recyclabilité) à évaluer. Les données d'entrée proviennent de la MSX-CDSM [DSMConf, ComInd] représentant plusieurs familles de produits. L'algorithme d'ingénierie de performances comprend 5 principales étapes (Figure 6.11) :

- extraction des données à partir de la matrice représentant l'instance choisie :
  - ✓ choix d'une famille de produits ;
  - ✓ choix d'une instance dans cette famille ;
  - ✓ extraction des informations dans la matrice représentant l'instance choisie.
- choix du/des domaine(s) comportemental(aux) à évaluer ;
- calcul des métriques du/des domaine(s) choisi(s) ;
- validation de chaque domaine comportemental ;
- validation de l'instance (validation multi-domaines) ;

Pour une même instance de produit, le processus peut se répéter autant de fois que de domaines à évaluer. Si l'instance ne satisfait pas aux exigences, soit elle est améliorée et le processus recommence, soit elle est abandonnée et enregistrée et une autre instance est choisie. Si par contre l'instance satisfait les exigences de l'ensemble des domaines évalués, elle est retenue et enregistrée.

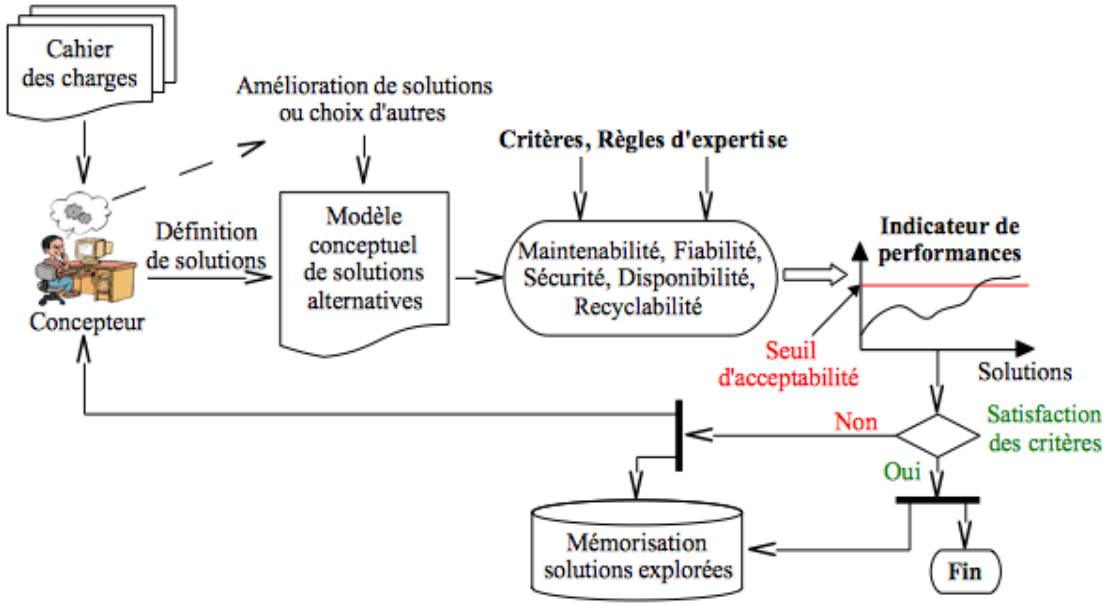

Figure 6.10 : Processus de validation d'une instance pour un domaine choisi

Il est possible, au cas où plusieurs domaines comportementaux doivent être évalués, de faire un compromis lors de la validation multi-domaine. En effet, une instance peut satisfaire les exigences de certains domaines mais pas d'autres, dans ce cas il est nécessaire de disposer de données permettant de prendre une décision. La Figure 6.10 donne le processus de validation d'une instance pour un domaine comportemental.

L'algorithme donnant le processus d'ingénierie de performances comportementales est représenté sur la Figure 6.11.

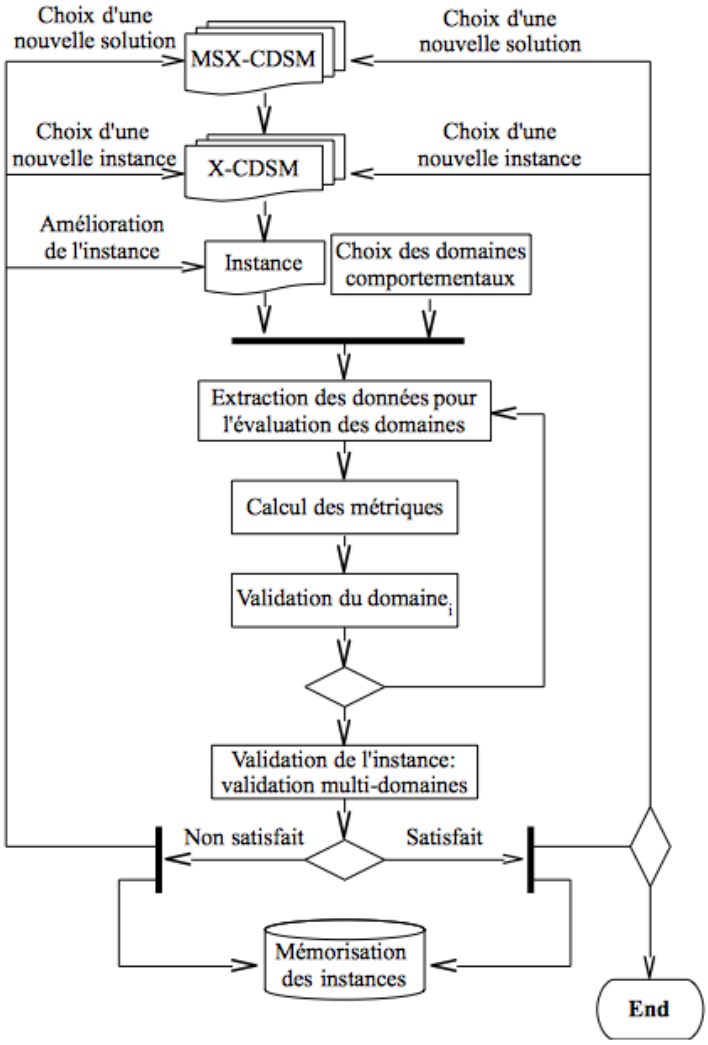

Figure 6.11 : Algorithme d'ingénierie des performances comportementales

Cet algorithme permet de partir d'un ensemble de classes de solutions (familles de produits) aux instances qui satisfont les exigences (fonctionnelles, structurelles, etc.). L'algorithme prend en entrée une MSX-CDSM représentant les classes de solutions (les X-CDSM). A partir de cette matrice on choisit une classe de solutions modélisée par une matrice X-CDSM. Cette dernière est instanciée pour obtenir une instance (produit) de la classe choisie. Les domaines comportementaux à évaluer sont choisis et l'instance est validée pour chaque domaine. En effet, les domaines sont pris un par un et à chaque fois qu'un d'entre eux est sélectionné, les données nécessaires à son évaluation sont extraites à partir de la matrice de l'instance. La métrique de ce domaine est calculée et le résultat obtenu est comparé au seuil (Figure 6.10). Le même processus est repris pour les autres domaines choisis. Après validation de tous les domaines, une validation multi-domaine est effectuée pour décider si l'instance est à retenir, améliorer ou abandonner. Cette instance est enregistrée dans une base de données quel que soit la décision prise. Au cas où l'instance est abandonnée, soit une nouvelle instance de la même classe de solutions est sélectionnée, soit une instance d'une nouvelle classe de solutions est sélectionnée. Si l'instance est retenue, il est également possible de choisir une autre instance pour

l'évaluer et la comparer aux instances retenues ou terminer le processus.

En utilisant cette méthode, la maquette numérique d'un produit est testée et validée avant de passer à sa phase de fabrication.

Conclusion

Dans ce chapitre nous avons proposé une amélioration des indicateurs de maintenabilité tels que le temps de désassemblage de produits complexes et le temps de remplacement d'un composant défaillant. Nous avons également proposé une méthodologie de recherche du chemin optimal de désassemblage facilitant le recyclage des composants en fin de vie. Dans les formules proposées, nous avons ajouté des paramètres tels que la position des liaisons et leur accessibilité. Nous avons également pris en compte la possibilité de défaire plusieurs liaisons simultanément. L'évaluation de la fiabilité de produits complexes à partir des fonctionnalités qu'ils doivent remplir est aussi traitée. Pour ce faire, nous avons proposé une méthodologie de modélisation fonctionnelle de familles de produits avec les X-CDSM pour calculer la probabilité qu'un produit puisse exécuter ces fonctionnalités critiques. Pour terminer, un algorithme d'ingénierie de performances comportementales est proposé. Il permet de passer du modèle MSX-CDSM représentant plusieurs familles de produits à leurs instances qui satisfont les exigences du cahier des charges. Il permet également de comparer des instances d'une même famille ou celles de plusieurs familles différentes. Cet algorithme sera implémenté dans le logiciel Product-BPAS dont les fonctionnalités et l’architecture sont décrites dans le Chapitre 7.

Quatrième partie : Implémentation et application

Chapitre 7 : Fonctionnalités, Architecture et analyse du Product-BPAS

Chapitre 8 : Implémentation et Mise en œuvre

Chapitre 7 : Fonctionnalités, Architecture et analyse du Product-BPAS

Chapitre 7

Fonctionnalités, Architecture et analyse du Product-BPAS

Introduction

Durant les travaux de cette thèse, l'ingénierie des performances comportementales est la troisième étape du processus de développement de nouveaux produits après la conception sémantique conceptuelle et la modélisation sémantique conceptuelle. L'analyse des performances peut porter sur le modèle objet structurel ou fonctionnel représenté avec le langage SysML ou avec les matrices X-CDSM ou sur le modèle géométrique conceptuel représenté en CAO-SC. Dans ce chapitre nous donnons l'architecture du logiciel Product Behavioral Performance Analysis System (Product-BPAS) dont l'objectif est d'instrumenter la MSC et l'IPC. Puis nous faisons l'analyse des diagrammes UML permettant son implémentation. Ce logiciel implémente la modélisation sémantique conceptuelle et l'ingénierie de performances comportementales de produits mécatroniques.

Ainsi, nous parlons dans la Section 7.1 des fonctionnalités du Product-BPAS. Son architecture est décrite dans la Section 7.2. La Section 7.3 présente les différents diagrammes UML et fait leur analyse.

7.1. Fonctionnalités

Le processus de développement de nouveaux produits comprend plusieurs étapes allant de l'élaboration du cahier des charges à la production de prototypes satisfaisant les exigences structurelles et fonctionnelles ainsi que les contraintes de conception. Les travaux de cette thèse s'intéressent à la conception conceptuelle, à la modélisation conceptuelle et à l'ingénierie de performances comportementales. La conception conceptuelle et une partie de la modélisation conceptuelle telle que la modélisation avec SysML et la CAO-SC sont faites en amont du logiciel Product-BPAS. Ainsi, le logiciel Product-BPAS qui est un outil d'aide à la conception de produits mécatroniques propose essentiellement les fonctionnalités de modélisation sémantique conceptuelle, avec la création de matrices X-CDSM structurelles et fonctionnelles, et d'ingénierie de performances comportementales. Ses fonctionnalités peuvent donc être déclinées ainsi qu'il suit :

1. Aide à la modélisation sémantique conceptuelle :
   a) Importation de modèles SysML et CAO-SC ;

b) Décomposition de modèles SysML et CAO-SC et création de nomenclatures ;
c) Création de matrices X-CDSM (Structurelle et fonctionnelle) ;
d) Contrôle de cohérence ;

2. Ingénierie de performances comportementales :
   a) Evaluation des performances comportementales d'instances de produit (fiabilité, maintenabilité, disponibilité, sécurité, recyclabilité) ;
   b) Optimisation d'instances de produits ;
   c) Identification des instances de produits satisfaisant les exigences structurelles, fonctionnelles et comportementales.

L'implémentation de ces fonctionnalités est décrite dans les sections suivantes. Ainsi, la Section 7.1.1 parle de la fonctionnalité d'Aide à la modélisation sémantique conceptuelle et la Section 7.1.2 traite la fonctionnalité d'Ingénierie de performances comportementales.

7.1.1. Aide à la modélisation sémantique conceptuelles

Le DSE contient un ensemble de familles de produits susceptibles de satisfaire les exigences structurelles et fonctionnelles du cahier des charges. La modélisation des produits qu'il contient est une étape très importante du processus de développement de nouveaux produits décrite dans cette thèse. En effet, pour évaluer les performances comportementales de ces produits, il est nécessaire de les représenter de la manière la plus fidèle possible. Ainsi, le modèle obtenu est utilisé pour identifier les instances de produits qui satisfont les exigences.

Le logiciel Product-BPAS implémente la fonctionnalité d'aide à la MSC permettant de concevoir la matrice X-CDSM d'une famille de produits appartenant au DSE. Pour cela le modèle SysML et/ou CAO-SC de la famille de produits est importé comme décrit dans la Section 7.1.1.1. Ce modèle est ensuite décomposé et la nomenclature des composants et liaisons est créée comme nous le détaillons dans la Section 7.1.1.2. La Section 7.1.1.3 parle de la création de la matrice X-CDSM à partir du modèle SysML et/ou CAO-SC. Le contrôle de la cohérence des composants et liaisons choisis fait l'objet de la Section 7.1.1.4.

7.1.1.1. Importation de modèles

La fonctionnalité *Importation de modèles* permet d'importer un modèle SysML ou CAO-SC conçu avec un logiciel spécifique vers le Product-BPAS. En effet, la conception du modèle SysML fait suite

à la création du DSE après analyse du cahier des charges. Le modèle CAO-SC correspondant peut ensuite être créé. Puis, pour aller vers l'IPC on importe le fichier contenant soit le modèle SysML soit le modèle CAO-SC soit les deux dans le Product-BPAS. Les exigences structurelles, fonctionnelles et comportementales (seuils d'acceptabilité des différents domaines comportementaux : fiabilité, maintenabilité, disponibilité, sécurité et recyclabilité) sont également renseignées en vue de la validation des instances de produits. La nomenclature des instances de produits est alors élaborée après décomposition du/des modèle(s) comme décrit dans la Section 7.1.1.2.

7.1.1.2. Décomposition de modèles et création de nomenclatures

La décomposition du modèle SysML ou CAO-SC d'une famille de produits permet de dresser la liste de ses classes de composants, de liaisons et de fonctions. La nomenclature des instances d'une famille de produits permettant de voir la hiérarchisation de leurs différents sous-ensembles et pièces est alors conçue. Elle donne également pour chaque sous-ensemble ou pièce élémentaire les liaisons auxquelles il participe et les fonctions qu'il exécute. Ainsi, les matrices X-CDSM structurelle et fonctionnelle de la famille de produit peuvent être créées comme nous le décrivons dans la Section 7.1.1.3.

7.1.1.3. Création de matrices X-CDSM

Pour chaque famille de produits nous avons deux matrices X-CDSM : la X-CDSM structurelle et la X-CDSM fonctionnelle. La première représente les classes de composants et montre pour chacun d'eux les liaisons auxquelles il participe. Le sens de chaque liaison est représenté et le nombre de liaisons auxquelles participe chaque classe de composants est connu. Les données sémantiques telles que la fiabilité, la criticité, l'impact environnemental de chaque composant sont représentées. La X-CDSM fonctionnelle par contre permet de représenter les fonctions permettant d'implémenter les exigences fonctionnelles. Chaque exigence étant décomposée en une ou plusieurs fonctions pouvant à leur tour être décomposées en modules. Les interactions entre les différentes fonctions à travers leurs modules sont représentées pour permettre de voir les interdépendances entre modules et entre fonctions. Après conception des matrices, les instances de produits peuvent être identifiées. L'identification des instances se fait par instanciation de chaque classe de composants, de chaque classe de liaisons et de chaque classe de fonctions. Avant de passer à l'évaluation de performance, la cohérence des assemblages doit être validée (Section 7.1.1.4).

7.1.1.4. Contrôle de cohérence

Le contrôle de cohérence est la première vérification faite sur une instance avant d'évaluer ses performances comportementales. En effet, après avoir choisi les instances de composants et de liaisons, il est nécessaire de vérifier si ces composants peuvent aller ensemble et s'il est possible d'établir ces liaisons entre eux. Il faut également vérifier si les composants choisis peuvent exécuter les fonctions à eux confiées. Si les choix sont cohérents, alors, le modèle géométrique 3D de l'instance ainsi obtenue peut être conçu.

La Figure 7.1 donne le processus allant de l'importation de modèles à la création de matrices X-CDSM comme décrit dans la Section 7.1.1.

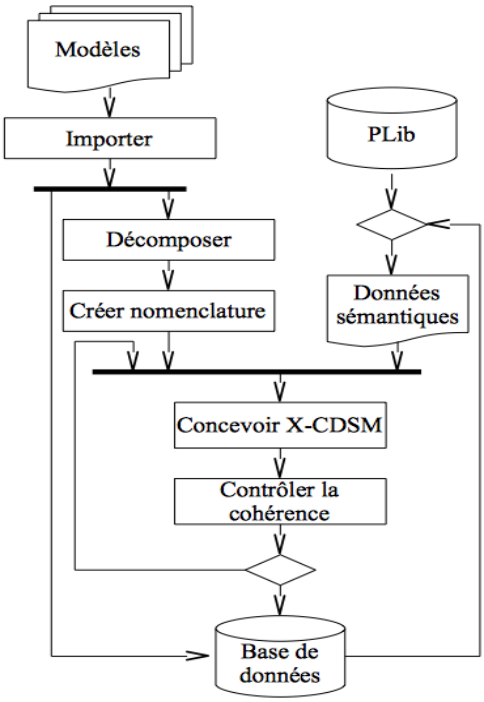

Figure 7.1 : Algorithme de conception de matrices X-CDSM

Après ces étapes, l'évaluation de performances comportementales de l'instance permettant de voir si oui ou non elle satisfait les exigences structurelles, fonctionnelles et comportementales peut alors démarrer. Ce processus est décrit dans la Section 7.1.2.

7.1.2. Aide à l'ingénierie de performances comportementales

Pour identifier les produits satisfaisant les exigences du cahier des charges, il est nécessaire de calculer les valeurs des indicateurs de performances des différents domaines concernés et de les

comparer avec les seuils fixés. Les données permettant ce calcul proviennent de la X-CDSM. L'ingénierie de performances comportementales est donc un processus qui part du modèle représentant une famille de produits complexes (matrice X-CDSM) et va jusqu'à l'identification des instances qui répondent aux critères de performances comportementales. Ce processus inclut l'évaluation des domaines comportementaux (Section 7.1.2.1), l'optimisation des produits (Section 7.1.2.2) et l'identification des instances satisfaisant les exigences (Section 7.1.2.3).

7.1.2.1. Evaluation des performances comportementales

Le processus d'évaluation de performances comportementales est composé de plusieurs étapes que sont : l'extraction des données à partir de la matrice de l'instance et de son modèle géométrique, le calcul des valeurs des métriques des domaines comportementaux choisis et leur comparaison aux seuils d'acceptabilité prédéfinis. A ce niveau, chaque instance de produit est validée pour chaque domaine comportemental. Ainsi, une instance peut satisfaire les exigences de fiabilité et ne pas satisfaire celles de la maintenabilité par exemple. Dans ce cas, plusieurs solutions sont possibles : l'abandonner, l'améliorer pour qu'elle satisfasse les exigences de maintenabilité (Section 7.1.2.2) ou passer à la validation multi-domaine (Section 7.1.2.3).

7.1.2.2. Optimisation d'instances de produit

L'optimisation consiste à modifier les instances de composants et/ou de liaisons tout en veillant à la cohérence de l'instance. Elle peut se faire à deux niveaux : lors de la validation de l'instance pour les domaines comportementaux ou lors de la validation finale de l'instance (validation multi-domaines). Si elle est faite dans le premier cas, elle permet de satisfaire les exigences d'un domaine comportemental donné. Cependant, dans le deuxième cas, elle permet de satisfaire les exigences inter-domaine décrites dans la Section 7.1.2.3.

7.1.2.3. Identification des instances satisfaisant les exigences comportementales

Pour qu'une instance de produit soit retenue comme satisfaisant les exigences du cahier des charges, elle doit passer la validation multi-domaine. Cette validation concerne toutes les exigences tant structurelles, fonctionnelles que comportementales. En ce qui concerne les performances comportementales, une métrique et un seuil d'acceptabilité inter-domaine sont définis. La métrique inter-domaine permet de faire un compromis entre les domaines comportementaux. Pour calculer sa valeur un coefficient est attribué à chaque domaine comportemental. Le résultat trouvé est comparé au seuil inter-domaine. En effet, une instance peut satisfaire les exigences de certains domaines

comportementaux mais pas les autres. Cependant, pour chaque domaine comportemental un seuil éliminatoire en deçà duquel l'instance ne peut être acceptée est fixé. Ainsi, pour qu'une instance soit acceptée il faut que pour chaque domaine comportemental la valeur de sa métrique dépasse ce seuil éliminatoire et que la valeur de la métrique inter-domaine soit supérieure ou égale au seuil inter-domaine. De plus, il faut que l'instance implémente toutes les fonctionnalités requises tout en satisfaisant les exigences structurelles.

Les fonctionnalités sont regroupées dans 3 modules que sont : le Design review, l'enrichissement sémantique et l'ingénierie de performances comportementales comme le montre la Figure 7.2. Ce logiciel prend en entrée des modèles venant d'outils de modélisation tels que les logiciels de CAO-SC ou les outils de modélisation SysML. Le modèle SysML contient une description des différentes classes de composants, classes de liaisons, classes de fonctions et de modules qui décrivent les instances d'une famille de produits. Tandis que le modèle CAO-SC représente un modèle SysML dans un système de CAO-SC.

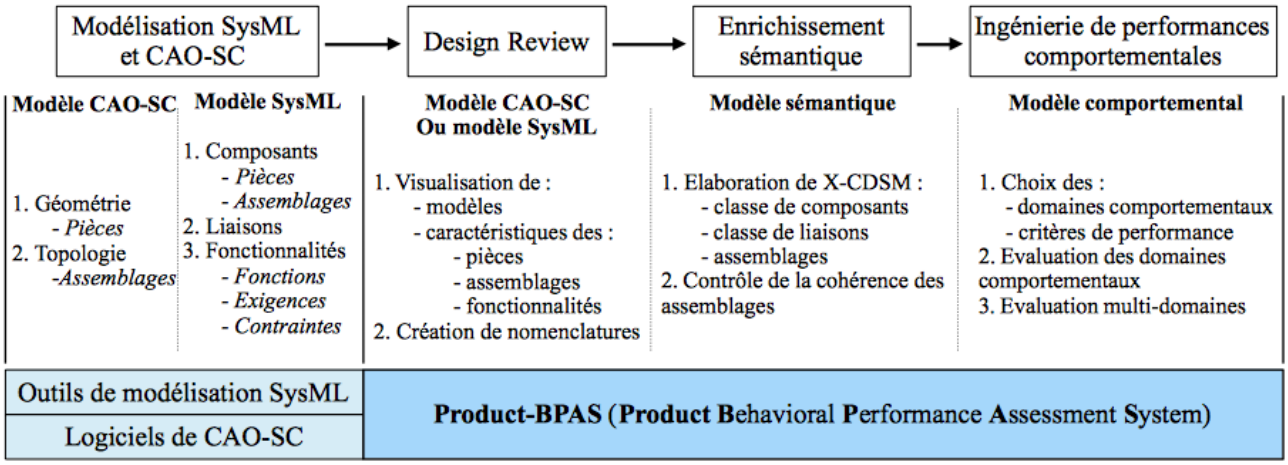

Figure 7.2 : Fonctionnalités du logiciel Product-BPAS

Pour implémenter les fonctionnalités décrites ci-dessus, nous donnons dans la Section 7.3 suivante l'architecture du logiciel Product-BPAS.

7.2. Architecture

Les fonctionnalités du logiciel seront implémentées suivant l'architecture données dans la Figure 7.3 de la Section 7.2.1. Ces fonctionnalités sont exécutées par 3 principaux modules présentés dans la Section 7.2.2.

7.2.1. Succession d'exécution des fonctionnalités

Pour créer de nouveaux produits et partir de l'importation de modèles SysML à l'ingénierie de performances, l'utilisation des fonctionnalités qu'offre le logiciel se font suivant un ordre précis. La Figure 7.3 donne l'architecture du logiciel avec les différentes fonctionnalités et leur ordre d'exécution durant le processus de création de produits.

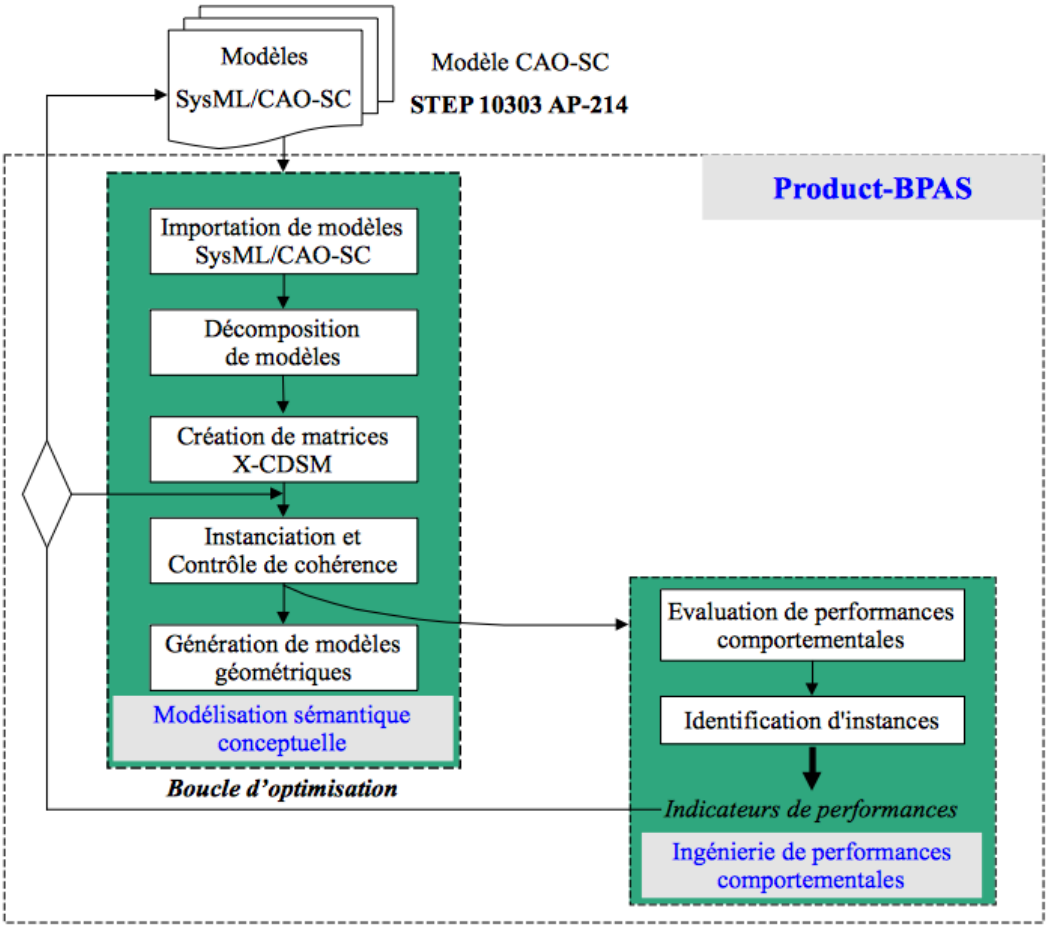

Figure 7.3 : Architecture du logiciel Product-BPAS

Pour l'implémentation de ces fonctionnalités, le logiciel est subdivisé en trois modules. Ces modules sont décrits dans la Section 7.2.2.

7.2.2. Les modules du logiciel

Conformément aux fonctionnalités citées dans la Section 7.1, le logiciel contient 3 modules : ***Design review***, ***Enrichissement sémantique*** et ***Ingénierie de performances comportementales***. Les fonctionnalités implémentées par chaque module du logiciel sont décrites dans les sections 7.2.2.1

(***Design review***), 7.2.2.2 (***Enrichissement sémantique***) et 7.2.2.3 (***Ingénierie de performances comportementales***).

7.2.2.1. Le module Design Review

Ce module est le premier à être exécuté en cas de création d'un nouveau projet. Il gère la création de nouveaux projets avec l'enregistrement des exigences du cahier des charges et implémente les fonctionnalités *Importation de modèles* SysML et/ou CAO-SC, *Importation de données de composants PLib* et *Décomposition de modèles et élaboration de nomenclatures*. Il gère également la visualisation de modèles déjà enregistrés. Les fonctions permettant de gérer ces tâches sont :

- l'importation de modèles CAO-SC et SysML ;
- l'importation de données de composants ;
- la visualisation de modèles CAO-SC et SysML déjà enregistrés ;
- la visualisation des caractéristiques des composants et liaisons ;
- la visualisation des caractéristiques des sous-assemblages ;
- la décomposition des modèles SysML et CAO-SC pour la création de nomenclatures avec la liste de pièces et de liaisons entre elles.

7.2.2.2. Le module Enrichissement sémantique

La fonctionnalité principale implémentée par ce module est la *Conception de matrices X-CDSM*. Elle se fait à partir de nomenclatures élaborées par le module Design review. Il implémente également la fonctionnalité *Contrôle de la cohérence*. Les fonctions permettant de les gérer sont :

- l'édition des composants ;
- l'édition des liaisons entre composants ;
- l'élaboration de la matrice sémantique de chaque instance à partir de sa nomenclature ;
- l'enrichissement de cette matrice avec les données sémantiques supplémentaires (fiabilité, criticité, impact environnemental) ;
- le contrôle de la cohérence des assemblages.

7.2.2.3. Le module Ingénierie de performances comportementales

Ce module permet d'identifier dans une famille de produits les instances satisfaisant les exigences et contraintes du cahier des charges. Il implémente les fonctionnalités *Evaluation de performance comportementales, Optimisation d'instances* et *Identification d'instances satisfaisant les exigences*. Les fonctions permettant de mener à bien ces fonctionnalités sont :

- le choix du ou des domaine(s) comportemental(aux) à évaluer ;
- le choix des métriques conformément à chaque domaine choisi ;
- l'extraction des données conformément au(x) domaine(s) choisi(s) ;
- le calcul des valeurs des métriques choisis ;
- la validation de chaque instance par rapport à chaque domaine choisi ;
- la validation finale de chaque instance en considérant tous les domaines évalués.

7.3. Analyse des diagrammes

Nous allons dans cette section, décrire et analyser les différents diagrammes permettant d'implémenter les fonctionnalités décrites dans la Section 7.1. Ainsi, nous parlons dans la Section 7.3.1 du diagramme de cas d'utilisation. La Section 7.3.2 présente le diagramme de classe du logiciel. La Section 7.3.3 présente le diagramme de séquence. Les diagrammes d'activités sont décrits dans la Section 7.3.4. Enfin, le diagramme de composant est présenté dans la Section è.3.5.

7.3.1. Diagramme de cas d'utilisation

Conformément aux fonctionnalités du logiciel, nous avons un certain nombre de cas d'utilisation représentés sur la Figure 7.4 et dont les plus importants sont :

- ***Créer projet :*** Ce cas d'utilisation est le premier à être exécuté quand on commence un nouveau projet. Pour créer un nouveau projet, on renseigne le cahier des charges (***Renseigner cahier des charges***) et on importe le modèle SysML des composants, liaisons et fonctionnalités (***Importer modèle SysML***) ;
- ***Concevoir matrice X-CDSM :*** C'est la création de la X-CDSM représentant une famille de produits complexes. Elle se fait à partir de la nomenclature conçue à partir du modèle SysML (***Elaborer nomenclature***). La création de la nomenclature passe par la décomposition du modèle (***Décomposer modèle***). Cette nomenclature est enrichie avec des données sémantiques telles que la fiabilité, la criticité, l'impact environnemental des composants (***Ajouter données sémantiques***). A ce stade, la cohérence des composants et liaisons choisis est également vérifiée (***Vérifier cohérence***) ;
- ***Evaluer performances :*** Elle consiste à chercher parmi les instances d'une famille de produits celles qui satisfont les exigences comportementales. A la fin de ce processus, ces instances sont validées et enregistrées. La validation est faite en plusieurs étapes allant du choix des domaines comportementaux à évaluer à la validation multi-domaine des instances. Ainsi,

après conception de la matrice X-CDSM, et choix des domaines comportementaux à évaluer (***Choisir domaines comportementaux***), les données permettant le calcul des métriques (***Calculer métriques***) de ces domaines sont extraites à partir de la X-CDSM (***Extraire données***). Les résultats obtenus sont comparés aux seuils d'acceptabilité (***Comparer résultats aux seuils***) des domaines en vus de la validation de ces domaines un à un (***Valider domaines comportementaux***). La validation d'une instance (***Valider instances***) suit celles des domaines comportementaux et se base sur leurs résultats. Ces deux validations peuvent être étendues par une optimisation du modèle de l'instance (***Optimiser modèle***) ;.

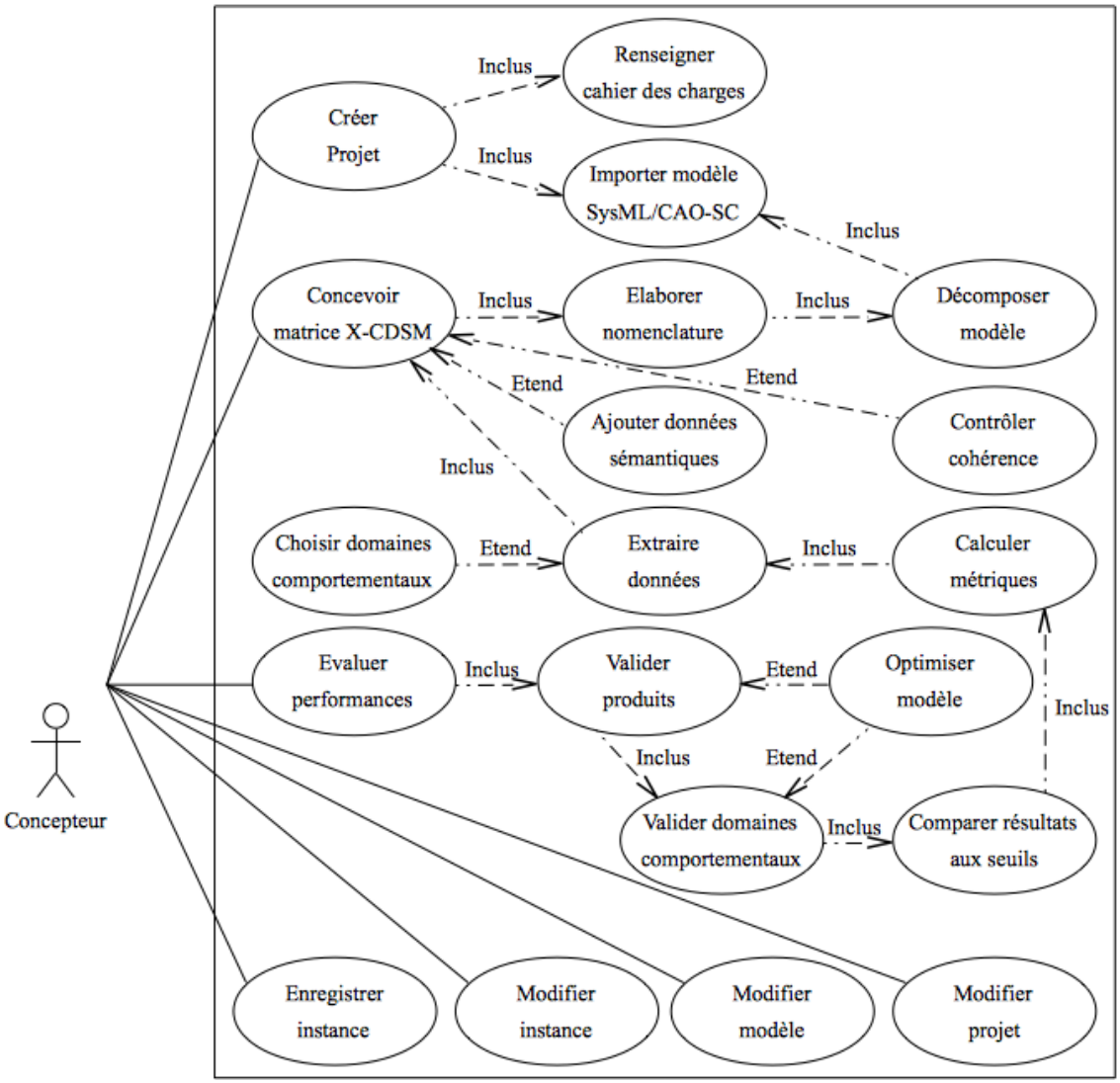

Figure 7.4 : Diagramme de cas d'utilisation

Les autres cas d'utilisation comme ***Enregistrer instance***, ***Modifier instance***, ***Modifier modèle***, ***Modifier Projet*** permettent respectivement d'enregistrer une instance déjà évaluée qu'elle satisfasse

les exigences ou non, de modifier une instance déjà évaluée et enregistrée, de modifier un modèle déjà enregistré en cas de besoin et de modifier un projet déjà existant au cas où des améliorations sont apportées au cahier des charges par exemple

7.3.2. Diagramme de classes

Le logiciel est articulé autour de trois modules. Nous montrons sur le modèle représenté sur la Figure 7.5 les différentes classes qui seront manipulées dans ce logiciel et les liens qui les relient. A partir du cahier des charges décrivant les exigences fonctionnelles et structurelles ainsi que les attentes en termes de délai, coût et performances comportementales, les modèles représentant les familles de produits susceptibles de le satisfaire sont conçus. Nous avons alors les classes ***Familles de produits***, ***Modèle SysML***, ***Modèle CAO-SC*** et ***Matrice X-CDSM***.

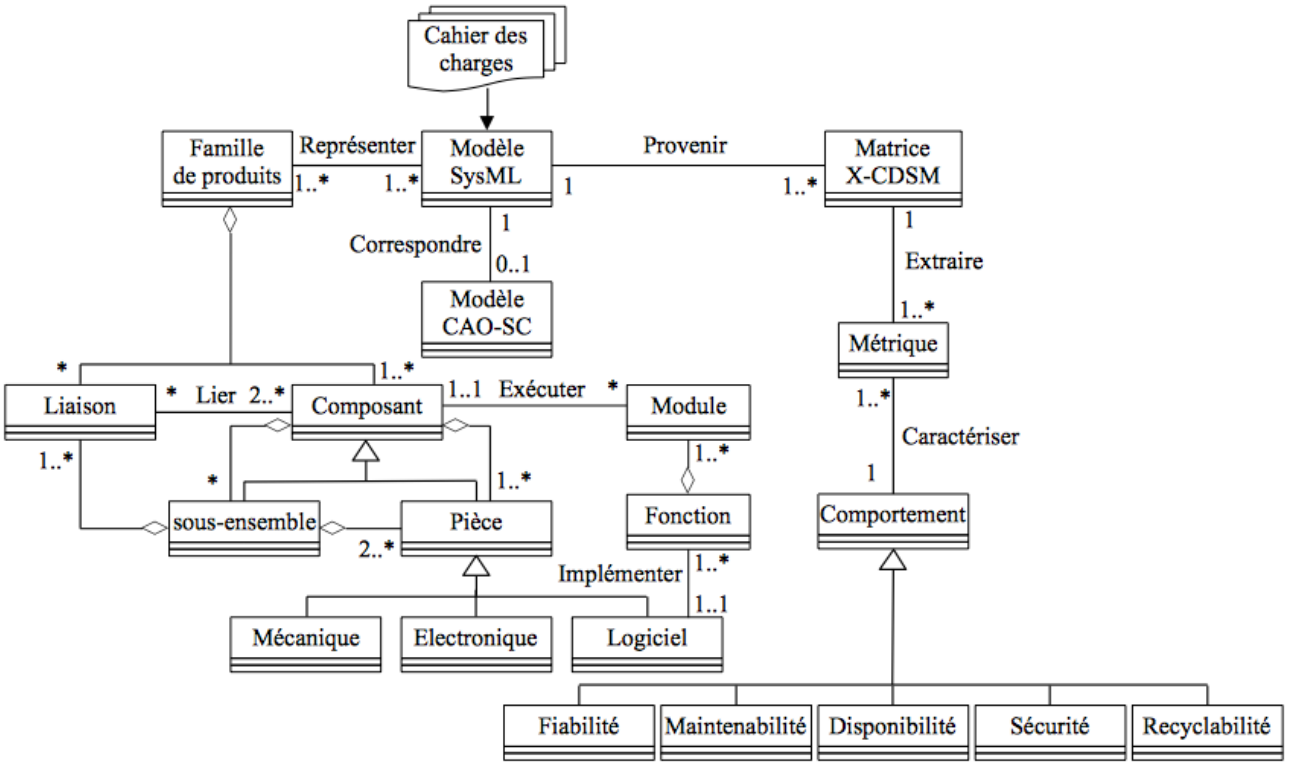

Figure 7.5 : Diagramme de classe du Product-BPAS

Les produits sont constitués de ***composants*** reliés par des ***liaisons***. Un composant peut être un ***sous-ensemble*** ou une ***pièce*** élémentaire. Une pièce peut être de type ***mécanique***, ***électronique*** ou une application informatique (***Logiciel***). Chaque application informatique implémente une ou plusieurs fonctionnalités du futur produit déclinées en ***fonctions*** qui peuvent être décomposées en ***modules*** exécutés par les composants. Le modèle CAO-SC correspondant au modèle SysML peut

être conçu. Ce modèle peut contenir d'autres informations sur les composants et liaisons. A partir de la matrice X-CDSM conçue à partir du modèle SysML et/ou CAO-SC d'une famille de produit, on extrait les différentes instances pour aller vers l'ingénierie de performances comportementales. Ainsi, les domaines comportementaux (***fiabilité***, ***maintenabilité***, ***disponibilité***, ***sécurité*** et ***recyclabilité***) à évaluer sont choisis et pour chacun d'eux, la ***métrique*** le caractérisant et permettant d'obtenir son niveau de satisfaction est calculée.

Les échanges d'informations entre les différents modules pour l'exécution des cas d'utilisation sont représentés dans le diagramme de séquences décrit dans la Section 7.3.3.

7.3.3. Diagramme de séquences

Nous montrons sur le diagramme de séquence de la Figure 7.6 l'échange d'informations entre les modules et la base de données ainsi que celles venant de fichiers externes.

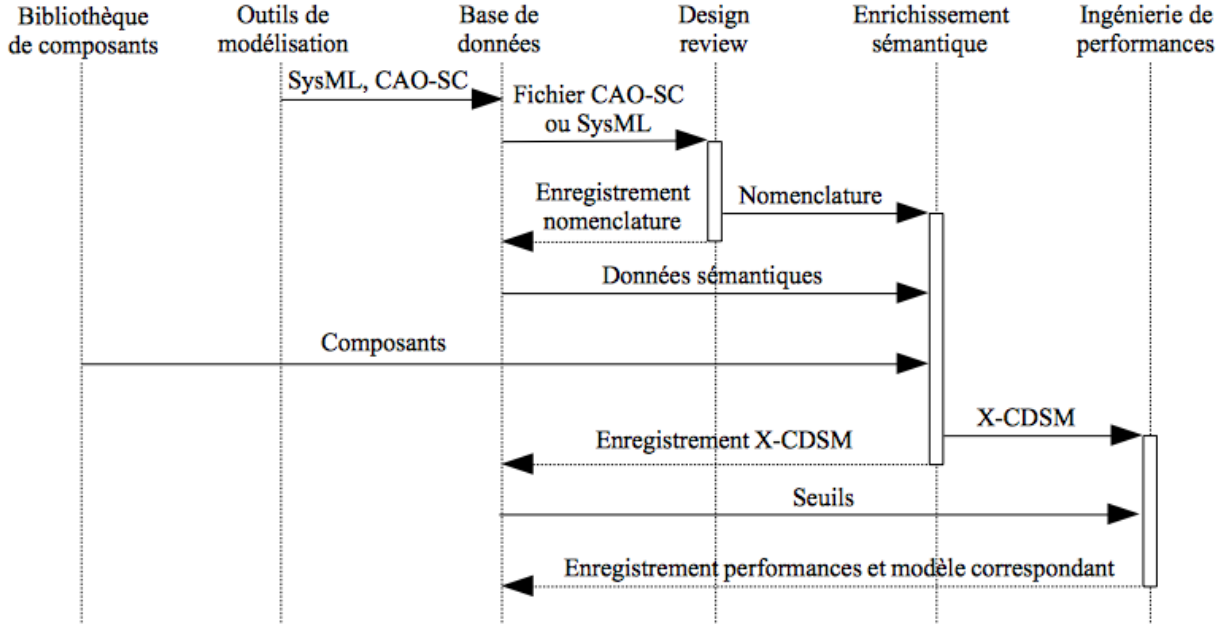

Figure 7.6 : Diagramme de séquences

Les d'informations échangées sont traitées par les modules suivant les activités représentées sur le digramme d'activités décrit dans la Section 7.3.4.

7.3.4. Diagramme d'activités

Le déroulement des activités permettant d'implémenter les cas d'utilisation (Figure 7.4) en prenant en compte les séquences (Figure 7.6) est représenté sur la Figure 7.7.

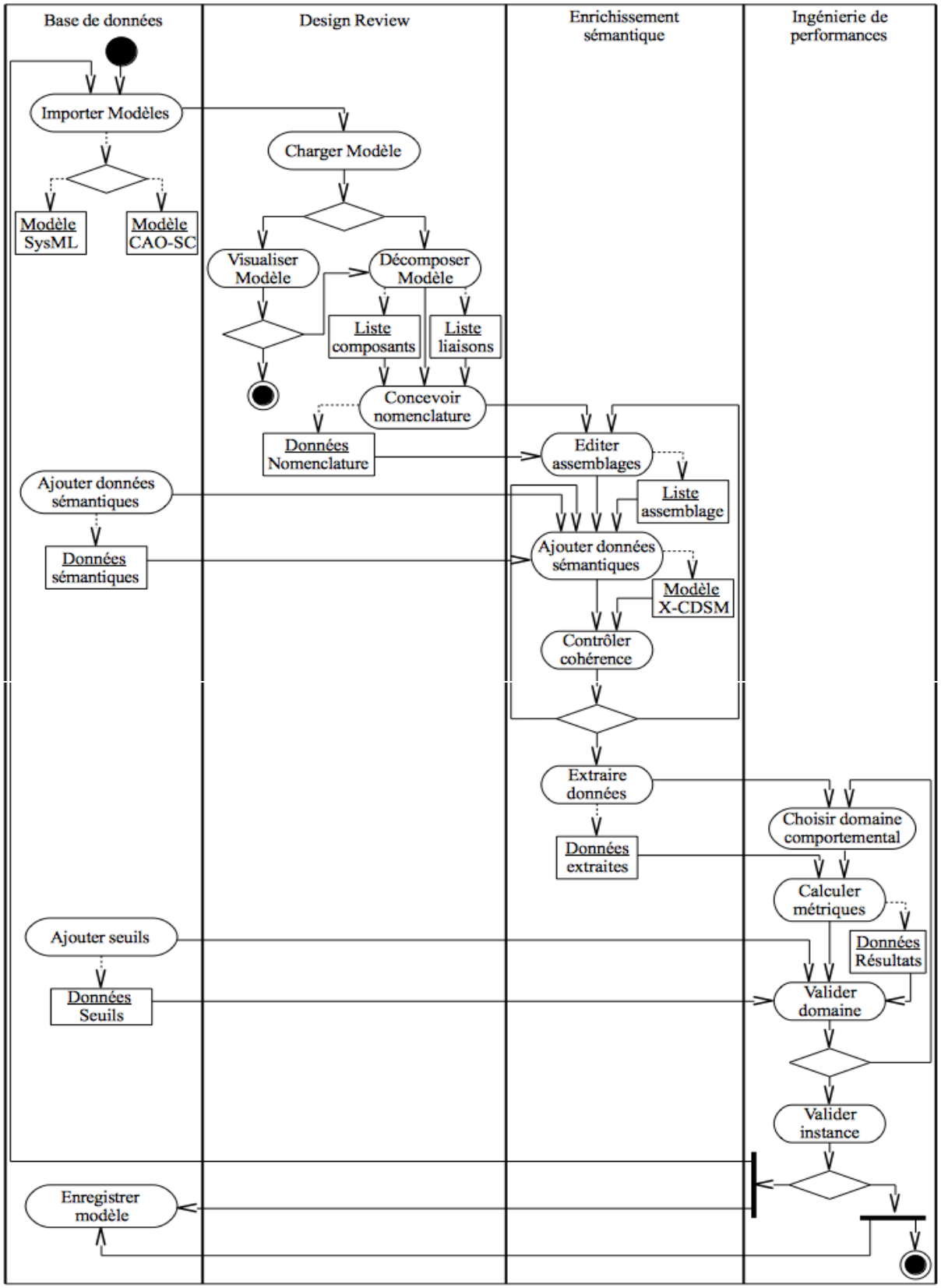

Figure 7.7 : Diagramme d'activités

Le déroulement d'un projet de conception de nouveaux produits complexes suit plusieurs étapes dont certaines ont lieu en amont du logiciel. Ces étapes telles que la modélisation conceptuelle objet avec SysML et la modélisation géométrique 3D avec la CAO-SC ne sont pas décrites dans le diagramme d'activités. Le logiciel donne cependant la possibilité d'importer des modèles SysML ou CAO-SC puis faire leur décomposition pour créer la nomenclature de produits. Ces nomenclatures sont ensuite utilisées pour la conception de la matrice X-CDSM qui servira d'entrée au module d'ingénierie de performances comportementales comme montré sur la Figure 7.7.

Les activités représentées sur la Figure 7.7 sont implémentées par les sous-modules dont les relations sont représentées sur la Figure 7.8 décrite dans la Section 7.3.5.

7.3.5. Diagramme de composants

Le diagramme de composant représenté sur la Figure 7.8 donne les différents modules du logiciel avec leurs sous modules. Il montre les relations entre les sous-modules d'un même module mais aussi celles entre deux modules différents. Nous y représentons également les relations entre ces modules et la base de données ainsi que celles entre cette dernières et d'autres logiciels tels que les logiciels de CAO-SC, de modélisation SysML et les bibliothèques de composants PLib. On voit dans ce diagramme que :

- Des fichiers venant de différentes sources peuvent être importés dans la base de données et manipulés par le logiciel ;
- Les modèles (CAO-SC et SysML) importés sont décomposés par le module Design review qui élabore par la suite la nomenclature et l'enregistre dans la base de données ;
- La nomenclature conçue est utilisée par le module Enrichissement sémantique pour concevoir la matrice X-CDSM. Pour la conception de cette matrice, ce module a également besoin de données sémantiques supplémentaires venant de la base de données. Il effectue finalement un contrôle de cohérence pour voir si la matrice peut être validée et enregistrée dans la base de données ;
- A partir de la matrice X-CDSM, le module Ingénierie de performances comportementales extrait les données nécessaires pour le calcul des valeurs des métriques des différents domaines comportementaux. Le résultat de ce calcul est utilisé pour valider l'instance de produit. Cette validation se fait suivant deux étapes :
  - ✓ la validation de l'instance pour chaque domaine comportemental. Pour valider une

instance suivant un domaine donné, le résultat obtenu après calcul de la métrique de ce domaine est comparé au seuil d'acceptabilité de ce domaine ;

- ✓ la validation globale de l'instance suivant une analyse multi-domaine. Cette validation se fait suivant des compromis qui fixent dans quel cas une instance est considérée comme satisfaisant les exigences multi-domaines.

Quel que soit le résultat obtenu, l'instance est enregistrée avec le résultat de l'évaluation. Elle peut également être améliorée et réévaluée.

Nous avons ainsi trois sous-modules dans le ***Design Review***, deux sous-modules dans l'***Enrichissement sémantique*** et deux sous-modules dans l'***Ingénierie de performances comportementales***.

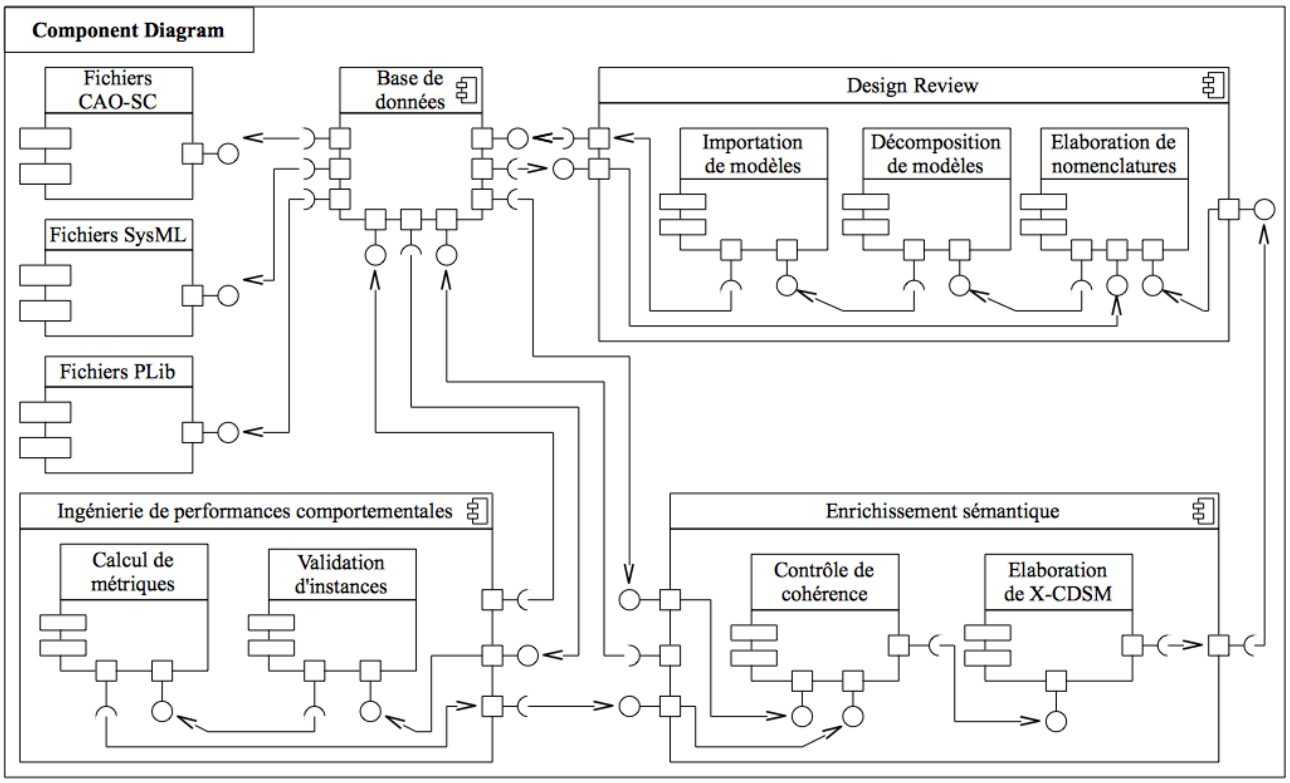

Figure 7.8 : Diagramme de composants

Conclusion

Nous avons décrit dans ce chapitre les fonctionnalités et l'architecture du logiciel Product-BPAS. Nous avons également donné les différents diagrammes permettant sa réalisation. C'est ainsi que nous avons proposé un diagramme de cas d'utilisation pour montrer les interdépendances entre les

différents cas d'utilisation. Puis, un diagramme de séquences montrant les échanges d'informations entre modules d'une part et entre les modules et la base de données d'autre part. Ensuite, un digramme d'activités donnant le déroulement de l'exécution des cas d'utilisation en prenant en compte les informations échangées est représenté. Pour finir, un diagramme de composants permettant de voir les interactions entre les sous-modules d'un même module et entre les sous-modules de différents modules est donné. Ce diagramme montre également les interactions entre la base de données et les modules ainsi que l'importation de fichiers externes par le logiciel.

Le chapitre 8 parle de l'implémentation des modules décrits dans ce présent chapitre. Il décrit, ainsi, les technologies utilisées et leur interopérabilité et présente un cas d'application permettant d'illustrer le calcul du temps de remplacement de composants défaillants via le logiciel Product-BPAS.

Chapitre 8 : Implémentation et mise en œuvre

Chapitre 8

Implémentation et mise en œuvre

Introduction

L'implémentation des fonctionnalités du logiciel Product-BPAS décrites dans le Chapitre 7 sera faite en utilisant plusieurs outils existants. En effet, il sera développé avec un langage de programmation dans un environnement de développement. Il utilisera une base de données pour sauvegarder des informations (modèles, exigences du cahier des charges, contraintes de conception, etc.) ou charger des données déjà enregistrées. Il devra également supporter des types de fichiers venant d'outils externes existant pour pouvoir les importer et les traiter. Ainsi, dans ce chapitre, nous allons répondre aux questions suivantes :

1. Quel langage de programmation sera utilisé pour le développer ?
2. Quel environnement de développement sera utilisé ?
3. Quel SGBD sera utilisé ?
4. Comment fonctionnera le logiciel par rapport aux logiciels existant ?
5. Quelles sont les données qu'il va traiter ?
6. D'où viennent ces données ?
7. Sous quel format seront-elles chargées dans le logiciel ?
8. Quelles seront les données en sortie de chaque module ?
9. Sous quel format seront les données en sortie de chaque module ?

Cette liste de questions, non exhaustive, donne un aperçu du travail qui devra aboutir à l'implémentation du logiciel Product-BPAS. Pour répondre à ces questions, ce chapitre est subdivisé en 4 sections. Dans la Section 8.1 on décrit les outils choisis pour développer le logiciel. Dans la Section 8.2 on parle des types de données qui seront supportés par le logiciel. Dans la Section 8.3 on donne l'algorithme permettant de calculer le temps de remplacement de composants défaillants et on présente un cas d'étude permettant d'illustrer l'apport des méthodologies proposées et de présenter le logiciel Product-BPAS.

8.1. Choix des outils et technologies

Le développement d'une application informatique nécessite l'utilisation d'outils spécifiques. Ces outils sont essentiellement un langage de programmation, un environnement de développement, un serveur de base de données. Le choix de ces outils doit se faire conformément aux fonctionnalités du

logiciel à développer et aux données qu'il va traiter. Nous allons dans cette section parler du langage de programmation choisi pour développer le logiciel (Section 8.1.1), de l'environnement de développement intégré (Section 8.1.2) et du SGBD (Section 8.1.3) utilisés.

8.1.1. Le langage de programmation

8.1.1.1. Introduction

La programmation consiste à rédiger la description d'une méthode de résolution d'un problème en une forme « compréhensible » pour la machine [Girardot, 2013]. Pour programmer, il faut alors un problème à résoudre, une méthode de résolution de ce problème (l'algorithme) et un langage permettant à la machine de comprendre les activités à exécuter. Un langage de programmation est un ensemble de règles permettant aux programmeurs de dialoguer avec la machine pour la résolution d'un problème. Il existe plusieurs types de langages de programmation différents. Parmi ces langages, nous avons les langages procéduraux (Pascal, C, Basic, ADA, ...), les langages fonctionnels (LISP, Scheme, ML, ...), les langages logiques (PROLOG, ...), les langages orientés-objet (SIMULA, SmallTalk, Eiffel, VB.Net, ...), les langages hybrides (Java, C++, Delphi, C#, …), etc. [web 17]. Chaque type de langage est créé pour un objectif bien précis et a ses avantages et ses inconvénients.

8.1.1.2. Choix d'un langage de programmation

Les modèles de produits et les données de composants que le logiciel va devoir manipuler sont éventuellement sous forme de classes d'objets. Ainsi, nous avons choisi le langage ***Java*** vu son caractère hybride. En effet, il nous permet de bénéficier des avantages de la programmation objet et de ceux de la programmation procédurale. Java permet également de faire de la programmation événementielle. De plus, sa philosophie WORA (Write Once, Run Anywhere) lui procure une portabilité très intéressante.

Alors, en prenant en compte ce choix, nous allons parler dans la Section 8.1.2 des environnements de développement intégrés pour en choisir un nous permettant de développer notre application.

8.1.2. L'environnement de développement intégré

8.1.2.1. Définition

Un environnement de développement intégré (EDI), est un logiciel regroupant un ensemble d'outils nécessaires au développement logiciel dans un (ou plusieurs) langage(s) de programmation [Labatut, 2007]

Un environnement de développement intégré est un ensemble d'outils conçus pour augmenter la productivité des développeurs de logiciels. Ces outils servent à développer des logiciels et sont

destinés à être utilisés ensemble [web 18].

Il inclut au minimum :

- un éditeur de texte spécialisé (avec coloration syntaxique, indentation automatique, complétion automatique, . . .) ;
- un compilateur (ou au moins l'intégration d'un compilateur existant) ;
- un débogueur (ou au moins l'intégration d'un débogueur existant) :
- des outils d'automatisation de la compilation et de gestion de projets

Pour développer une application avec le langage Java, il existe plusieurs EDI dont Eclipse, NetBeans, JBuilder.

8.1.2.2. Choix d'un EDI

NetBeans est né en 1997 d'un projet d'étudiants de la faculté de mathématiques et physique de l'université Charles de Prague. Plus tard, une société se forme autour du projet et commence la production de versions commercialisées d'un EDI NetBeans. Il est finalement acheté par Sun Microsystems en 1999 qui le place en open source depuis 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). Actuellement NetBeans est un Atelier de Génie Logiciel (AGL) et est très utilisé par les développeurs en Java.

Avec ***NetBeans*** on peut développer en Java, C, C++, XML, PHP, etc. Il est développé en langage Java et peut tourner sur Windows, Linux, Solaris et Mac Os X. Il est disponible en 23 langues et peut être enrichi par ajout de greffons.

8.1.3. La base de données

Depuis l'avènement de l'informatique le besoin de sauvegarder des informations pour une utilisation selon les besoins des utilisateurs s'est sentir. Ainsi, plusieurs générations de méthodes de structuration et de stockage de données se sont succédées. Ces méthodes vont des fichiers permettant une structuration assez sommaire des données aux SGBD relationnels-objets supportant les BDRO qui intègre le paradigme objet dans les bases de données relationnelles. Entre ces deux nous avons les SGBD hiérarchiques, les SGBD réseaux, les SGBD relationnels et les SGBD orientés-objet. Avant de donner notre choix dans la Section 8.1.3.4, nous allons parler dans la Section 8.1.3.1 des bases de données relationnelles, dans la Section 8.1.3.2 des bases de données objet et dans la Section 8.1.3.3 des bases de données relationnelles-objet.

8.1.3.1. Les bases de données relationnelles

Le modèle relationnel est caractérisé par des *relations* ou *tables* liées entre elles grâce à des *clés étrangères*. Chaque table a des *attributs* ayant chacun un *type* donnant les valeurs qu'il peut contenir.

Un sous-ensemble des attributs de chaque table forme sa *clé primaire* permettant de différencier de manière unique l'ensemble de ses *enregistrements*. Une table est représentée sous forme d'un tableau dont les attributs constituent les colonnes et les enregistrements les lignes. Ainsi, la compréhension des bases de données relationnelles est assez simple. Leur manipulation est facilitée par l'algèbre relationnelle qui contient des opérations ensemblistes dont les opérandes sont des tables et qui permettent d'agir sur elles.

Avantages

- Fondées sur une théorie rigoureuse et des principes simples ;
- Mature, fiable et performantes ;
- Indépendance entre programme et données ;
- Les bases de données relationnelles sont les bases de données les plus utilisées. Elles sont donc connues et maitrisées ;
- SQL est un langage permettant d'implémenter le modèle relationnel avec des API pour la plupart des langages de programmation ;
- Les systèmes de gestion de base de données relationnels incluent des outils performants de gestion de requêtes, de générateurs d'applications, d'administration, d'optimisation, etc.

Inconvénients

- La structure de donnée en tables est pauvre d'un point de vue de la modélisation logique ;
- Le *mapping* MCD vers MLD entraîne une perte de sémantique ;
- La manipulation de structures relationnelles par des langages objets entraîne une *impedance mismatch,* c'est à dire un décalage entre les structures de données pour le stockage et les structures de données pour le traitement (ce qui implique des conversions constantes d'un format à l'autre) ;
- La normalisation entraîne la genèse de structures de données complexes et très fragmentées, qui peuvent notamment poser des problèmes de performance ou d'évolutivité ;
- Le SQL doit toujours être combiné à d'autres langages de programmation pour être effectivement mis en œuvre ;
- La notion de méthode ne peut être intégrée au modèle logique, elle doit être gérée au niveau de l'implémentation physique ;
- Les types de données disponibles sont limités et non extensibles.

8.1.3.2. Les bases de données objet

Les bases de données objet ont été créées pour gérer des structures de données complexes, en profitant de la puissance de modélisation des modèles objets et de la puissance de stockage des BD

classiques. Leurs principaux objectifs sont :

- d'offrir aux langages de programmation orientés-objets des modalités de stockage permanent et de partage entre plusieurs utilisateurs ;
- d'offrir aux BD des types de données complexes et extensibles ;
- de permettre la représentation de structures complexes et/ou à taille variable.

Avantages

- Le schéma d'une BD objet est plus facile à appréhender que celui d'une BD relationnelle (il contient plus de sémantique, il est plus proche des entités réelles) ;
- L'héritage permet de mieux structurer le schéma et de factoriser certains éléments de modélisation ;
- La création de ses propres types et l'intégration de méthodes permettent une représentation plus directe du domaine ;
- L'identification des objets permet de supprimer les clés artificielles et donc de simplifier le schéma ;
- Les principes d'encapsulation et d'abstraction du modèle objet permettent de mieux séparer les BD de leurs applications (notion d'interface).

Inconvénients

- Gestion de la persistance et de la coexistence des objets en mémoire (pour leur manipulation applicative) et sur disque (pour leur persistance) complexe ;
- Gestion de la concurrence (transactions) plus difficile à mettre en œuvre ;
- Interdépendance forte des objets entre eux ;
- Gestion des pannes ;
- Complexité des systèmes (problème de fiabilité) ;
- Problème de compatibilité avec les SGBDR classiques.

8.1.3.3. Les bases de données relationnelles-objets

Les bases de données relationnelles-objet sont nées du double constat de la puissance nouvelle promise par les BDO et de leur insuffisance pour répondre aux exigences de l'industrie des BD classiques. Leur approche est plutôt d'introduire dans les BDR les concepts apportés par les BDO plutôt que de concevoir de nouveaux systèmes. Elles constituent à cet effet une extension des BDR à l'aide de types de données abstraits. Leurs principaux objectifs sont de :

- gérer des données complexes (temps, géo-référencement, multimédia, types utilisateurs, etc.) ;
- rapprocher le modèle logique du modèle conceptuel ;
- réduire l'impédance mismatch ;

- réduire les pertes de performance liées à la normalisation et aux jointures.

Avantages

- Encapsulation des données des tables ;
- Préservation des acquis des systèmes relationnels (indépendance données/traitements, fiabilité, performance, compatibilité ascendante…) ;
- Extension du langage SQL (norme SQL3) ;
- Mise en œuvre des concepts objets (classes, héritage, méthodes).

Inconvénients

- Modèle de données qui ne repose pas sur des principes simples / une théorie rigoureuse ;
- Pas de syntaxe commune de la part des éditeurs de SGBD (IBM, Oracle, PostgreSQL, SAP…) ;
- Il est facile de migrer d'un modèle relationnel vers un modèle objet, mais le retour en arrière est complexe ;
- Potentiel baisse d'efficacité due en particulier à la gestion de l'héritage sous forme de table.

8.1.3.4. Synthèse et choix d'un type de base de données

Nous avons parlé de trois types de bases de données avec leurs avantages et inconvénients. Les BDR ont beaucoup de forces, mais leur inconvénient principal par rapport à nos objectifs est qu'il est très difficile de gérer les objets avec elles. Les BDO ont apporté des innovations sur des aspects que les BDR ne savent pas faire, mais sans être au même niveau sur ce que les BDR savent bien faire. Les BDRO par contre se sont positionnées comme un mixage des deux précitées. Elles bénéficient des avantages des BDR tout en y ajoutant les innovations apportées par les BDO.

8.1.3.4.1. Choix du type de base de données

Durant son fonctionnement, le logiciel aura à manipuler des données sous forme de classes d'objets mais aussi des données sous forme d'enregistrements contenant des valeurs scalaires simples. Ainsi, avec la diversité de structure de ces données modèles de produits sous format STEP ou SysML, données de composants sous format PLib, notre choix porte sur les ***bases de données relationnelles-objets*** (***BDRO***) pour les sauvegarder.

8.1.3.4.2. Choix d'un SGBDRO

Etant donné que notre choix porte sur les BDRO, le SGBD doit supporter ce type de base de données. Plusieurs SGBD permettent d'implémenter et d'administrer des BDRO existent dont certains sont gratuits et open source comme PostgreSQL, d'autres propriétaires et commerciaux comme Oracle. Cependant, la manière de gérer les objets varie d'un SGBD à l'autre. Notre choix porte sur le

SGBDRO ***PostgreSQL*** vue qu'il est open source et gratuit. De plus, durant sa thèse, El Hadj Mimoune a proposé une méthodologie de représentation des données de composants de produits basée sur l'intégration du modèle de données PLib au sein du SGBDRO PostgreSQL [Mimoune, 2004]. Cette intégration a permis de favoriser le passage d'une manière automatique d'un modèle à un autre. La partie qui nous intéresse le plus est, cependant, le transfert du modèle et des instances de composants PLib dans le SGBDRO cible qui contient les données des modèles de produits.

8.2. Les données supportées par le logiciel

Durant son fonctionnement, le logiciel va importer des fichiers de diverses natures et formats. Parmi ces fichiers nous avons :

- des diagrammes SysML créés avec des outils de modélisation spécifiques ;
- des modèles CAO-SC conçus avec des systèmes de CAO-SC sous format STEP ;
- des données de composants issues de bibliothèque PLib.

8.2.1. Format des fichiers SysML

Les fichiers contenant les diagrammes SysML proviennent de logiciels de modélisation tels que Topcased, Papyrus for SysML, Visual Paradigm, SysML Toolkit, … Ces fichiers ont pour la plupart leur propre format qui dépend de l'éditeur du logiciel dont ils sont issus. Cependant, la plupart des outils de modélisation SysML donne la possibilité d'exporter un digramme sous format XMI. Ainsi, le Product-BPAS importe les diagrammes SysML sous ce format pour son exploitation et la création de la matrice sémantique conceptuelle. Cependant, un diagramme peut également être exporté sous format image pour son affichage.

8.2.2. Format des fichiers CAO-SC

Pour faciliter la représentation et l'échange de données de produits entre concepteurs, la norme STEP (ISO 10303) est standardisée par l'ISO. Les systèmes de CAO classiques donnent la possibilité d'enregistrer les modèles géométriques 3D au format STEP. La CAO-SC décrite dans le chapitre 5 permettra également d'enregistrer les modèles au format STEP. Ainsi, les modèles issus des systèmes de CAO-SC seront importés sous format STEP. La norme STEP utilise le langage Express pour représenter les modèles de produits.

8.2.3. Format des fichiers des données de composants

Les données de composants issus de bibliothèques de composants seront au format PLib. PLib (ISO 13584) est une norme internationale standardisée par l'ISO. La norme PLib utilise le langage

Express pour représenter les composants.

Le logiciel va alors supporter divers formats de fichiers dont les formats STEP, PLib, XMI.

8.3. Mise en œuvre

Dans cette section nous présentons un cas d'application pour illustrer les méthodologies de CSC et de MSC proposées. L'évaluation de la maintenabilité est appliquée à ce cas d'application. La métrique utilisée est le temps de remplacement de composants défaillants. Ainsi, nous commençons par donner les algorithmes de calcul de ce temps dans la Section 8.3.1. Puis, dans la Section 8.3.2 le cas d'application est présenté, ses exigences fonctionnelles, structurelles et comportementales sont spécifiées et les méthodologies de CSC et MSC sont lui sont appliquées. Enfin, les chemins de remplacement de deux composants sont recherchés via le Product-BPAS et pour chacun d'eux le temps nécessaire pour le faire est calculé.

8.3.1. Calcul du temps de remplacement de composants défaillants

L'algorithme présenté est composé de trois fonctions (**Position()**, **Voie()** et **Duree()**). La fonction **Duree()** qui permet de calculer le temps de remplacement de composants défaillants est présenté dans le tableau 8.2. Cette fonction s'appuie sur d'autres fonctions représentées sur le Tableau 8.1 et qui permettent de chercher la position d'un composant (**Position()**) et le chemin de remplacement de composants défaillants (**Voie()**). La fonction **Duree()** prend en entrée la position du composant à remplacer (résultat de la fonction **Position()**), le tableau contenant les composants à enlever pour atteindre le composant à remplacer (résultat de la fonction **Voie()**) et la matrice contenant les caractéristiques des composants et des liaisons.

Tableau 8.1 : Algorithme de recherche de chemin de remplacement

Comptage du nombre de liens des composants	**Construction du chemin d'accès à un composant**
// n est le nombre de composants **Type** tab : tableau[1..n+1] d'entiers tab1 : tableau[1..n, 1..n] d'entiers tab2 : tableau[1..n] de chaines	**Fonction Voie**(p : entier, t_p : tab1) : tab **variable** x, y, i, j, k : entier ch, chemin, sor : tab **Debut** k ← 0 **Si** (p >= 0) **Alors** **Pour** i ← 0 **à** n-1 **Faire** **Si** ((t_p[i, p] > 0) ∧ (i <> p)) **Alors** **Si** (k = 0) **Alors** chemin[k] ← i k ← k + 1 **Sinon** x ← 0 **Pour** y ← 0 **à** k – 1 **Faire**

```
Fonction Nb_Links(tp : tab1) : tab
   variable
            i, j : entier
            nb : tab
Debut
   Pour i ← 1 à n Faire
      Pour j ← 1 à n Faire
         Si (i <> j) Alors
            Si (tp[i, j] <> 0) Alors
               nb[i] ← nb[i] + 1
               nb[j] ← nb[j] + 1
            FinSi
         FinSi
      FinPour
   FinPour
   Retourner nb
FinFonction


Fonction Position(c : chaine, tc : tab2) : entier
  variable
            j, pos: entier
Debut
   j ← 0
   Tant Que ((tc[j] <> c) ∧ (j < n)) Faire
      j ← j + 1
   FinTQ
   Si (j > n) Alors
      pos ← -1
   Sinon
      pos ← j
   FinSi
   Retourner pos
FinFonction
```

```
            Si (chemin[y] = i) Alors
               x ← 1
            FinSi
         FinPour
         Si (x = 0) Alors
            chemin[k] = i
            k ← k + 1
         FinSi
      FinSi
      Pour j ← 0 à n-1 Faire
         Si ((tp[j, i] > 0) ∧ (j <> i)) Alors
            y ← 0
            Tant Que ((j <> chemin[y]) ∧ (y < k)) Faire
               y ← y + 1
            FinTQ
            Si (y = k) Alors
               x ← chemin[k – 1]
               chemin[k – 1] ← j
               chemin[k] ← x
               ch = Voie(j, tp)
               Si (ch[0] > 0) Alors
                  Pour x ← 1 à ch[0] Faire
                     y ← 0
                     Tant Que (ch[x] <> chemin[y]) ∧ (y < k) Faire
                        y ← y + 1
                     FinTQ
                     Si (y = k) Alors
                        chemin[k + 1] ← chemin[k]
                        chemin[k] ← chemin[k – 1]
                        chemin[k – 1] ← ch[x]
                        k ← k + 1
                     FinSi
                  FinPour
               FinSi
               k ← k + 1
            FinSi
         FinSi
      FinPour
   FinSi
FinPour
FinSi
chemin[k] ← p
sor[0] ← k
Pour  x ← 1 à k + 1 Faire
   sor[x] ← chemin[x – 1]
Retourner sor
FinFonction
```

- p est la position du composant C_i (dont on cherche le chemin) dans le tableau $t_C[i]$
- chemin[i] est le tableau dans lequel on met les composants à enlever pour atteindre le composant C_i
- n est le nombre de composants du système et k - 1 est le nombre de composants à enlever pour atteindre C_i

Dans le Tableau 8.1 une autre fonction **Nb_Links()** permettant de calculer le nombre de liaisons que chaque composant a avec les autres est également représenté.

Tableau 8.2 : Algorithme de calcul du temps de remplacement

Calcul du temps de remplacement

```
Fonction Duree(p : entier, chem : tab, mat : tab1) : entier
    variable
            dur, tempo, i, j, q, x, y : entier
Début
    dur ← 0
    tempo ← 0
    Si ((p >= 0) ∧ (ch[0] = 0)) Alors                    // Si le composant peut être enlevé directement
        Pour q ← 0 à n-1 Faire
            Si (q <> p) Alors
                dur ← dur + mat[p, q]
            FinSi
        FinPour
        Pour y ← 0 à n-1 Faire
            Si (y <> p) Alors
                dur ← dur + mat[y, p]
            FinSi
        FinPour
        dur ← 2 * dur
    Sinon Si ((p >= 0) ∧ (ch[0] > 0)) Alors              // S'il y a d'autres composants à enlever pour l'atteindre
        Pour y ← 1 à ch[0] + 1 Faire
            Pour q ← 0 à n Faire
                Si (ch[y] <> q) Alors
                    x ←0
                    Pour i ← 1 à ch[0]+1  Faire
                        Si (q <> ch[i]) Alors
                            x ← 1
                        FinSi
                    FinPour
                    Si (x = 1) Alors
                        tempo ← tempo + mat[ch[y], q]
                    Sinon
                        dur ← dur + mat[ch[y], q]
                    FinSi
                FinSi
            FinPour
            Pour j ← 0 à n-1 Faire
                Si (j <> ch[y]) Alors
                    x ← 0
                    Pour i ← 1 à ch[0]+1 Faire
                        Si (j = ch[i]) Alors
                            x ← 1
                        FinSi
                    FinPour
                    Si (x = 1) Alors
                        tempo ← tempo + mat[j, ch[y]]
                    Sinon
                        dur ← dur + mat[j, ch[y]]
                    FinSi
                FinSi
            FinPour
        FinPour
        dur ← 2 * dur + tempo
```

FinSi **Retourner** dur **FinFonction**
p est la position du composant C à remplacer **chem** est le tableau contenant la liste des indices des composants à enlever pour atteindre C **mat** est la matrice du produit contenant le composant C

Ces fonctions sont implémentées dans le logiciel Product-BPAS et leurs résultats sont montrés en Section 8.3.2.4.

8.3.2. Application

8.3.2.1. Description du cas d'application

Le cas d'étude choisi pour illustrer les contributions scientifiques faites durant les travaux de cette thèse est un fauteuil roulant intelligent. On part du général (Siège) au spécifique. Le rôle principal d'un siège est de permettre aux personnes de s'asseoir. Cependant, les sièges peuvent avoir d'autres rôles en plus de celui-ci notamment déplacer une personne d'un point A vers un point B, assister des personnes avec un handicap lourd dans certaines tâches. Un tel objet est généralement muni de pieds qui peuvent être des pieds fixes, des pieds sur roulettes, des roues, … Il peut également être pourvu de moteur. Un siège peut être installé dans un bureau, un véhicule, un avion, un train, etc. Il peut également être un fauteuil roulant pour handicapés. Nous avons donc un ensemble très vaste contenant tous les types de sièges que l'on peut trouver. On peut donner comme exemple les tabourets (A), les chaises rigides (B), les chaises pliantes (C), les fauteuils fixes (D), les fauteuils roulants (E), ...

Figure 8.1 : Exemples d'instances des classes appartenant à l'ESC

Chacun de ces types de sièges A, B, C, D, E représente un ensemble de sièges ayant des similitudes structurelles mais pouvant avoir des différences très profondes (forme, matière utilisée, type d'assemblage, etc.). Nous les appelons des classes de sièges. Nous donnons dans la Figure 8.1 une instance de chacune de ces classes de sièges.

L'ensemble contenant toutes ces classes de sièges est appelée l'*espace de solutions conceptuelle* (***ESC***). Cependant, les solutions qui seront retenues à la fin de cette étude devront respecter les exigences décrites dans les sections 8.3.2.1.1 (exigences fonctionnelles), 8.3.2.1.2 (exigences structurelles) et 8.3.2.1.3 (exigences comportementales).

8.3.2.1.1. Exigences fonctionnelles

Les principales fonctionnalités attendues des sièges sont de :

- permettre de s'asseoir ;
- permettre de se déplacer ;
- fournir une assistance à ses utilisateurs (intelligece) : localisation d'une adresse à l'aide d'un GPS, conduite automatique, téléphone et radio intégrés.

8.3.2.1.2. Exigences structurelles

Les sièges doivent être composés d'une assise, d'un dossier, d'un piètement, des accoudoirs. En plus de ces composants, ils doivent être motorisés avec un contrôleur électronique pour son pilotage. Les sièges doivent également être munis de repose-pieds. Chacun de ces éléments peut être un sous-ensemble contenant d'autres composants (sous-ensembles ou pièces élémentaires).

8.3.2.1.3. Exigences comportementales

Les domaines comportementaux qui seront étudiés dans ce chapitre sont la maintenabilité et la fiabilité en utilisant les formules d'évaluation décrites dans le chapitre 6.

Nous allons dans les sections suivantes appliquer la conception sémantique conceptuelle (Section 8.3.2.2), la modélisation sémantique conceptuelle (Section 8.3.2.3) avant d'évaluer la maintenabilité et la fiabilité des sièges en utilisant leur représentation matricielle (X-CDSM) (Section 8.3.3).

8.3.2.2. Conception sémantique conceptuelle

L'espace conceptuel correspondant est composé de tous les types d'objet conçus et ayant comme objectifs de permettre aux personnes de s'asseoir, de se déplacer et pouvant assister les personnes handicapées. Puis, en se basant sur les exigences décrites dans la section 8.3.2.1, et en les appliquant aux classes de sièges représentées sur la Figure 8.1, certaines classes peuvent être éliminées, d'autres peuvent perdre certaines de leurs instances. Les classes restantes sont les classes de sièges éligibles et constituent les éléments du *domaine de solutions éligibles* (***DSE***). Dans notre cas, seuls les sièges

de la classe F dont une instance est représentée sur la Figure 8.2 sont susceptibles de satisfaire les exigences.

F. Fauteuil roulant motorisé

Figure 8.2 : Exemple d'instance appartenant au DES

Après obtention du domaine de solutions éligibles, ces instances sont ensuite modélisées en vue de l'évaluation de leurs performances comportementales. Nous utilisons des outils de modélisation objet et plus précisément SysML qui convient bien à la modélisation des systèmes complexes.

8.3.2.3. Modélisation sémantique conceptuelle

La modélisation sémantique conceptuelle du DSE est faite en deux étapes : d'abord la modélisation avec le langage SysML (Section 8.3.2.3.1), puis la création de la matrice X-CDSM correspondante (Section 8.3.2.3.2).

8.3.2.3.1. Modèle SysML de l'objet siège

Dans cette partie nous allons utiliser le formalisme SysML pour la modélisation des sièges de la classe F. Nous donnons les diagrammes SysML permettant de représenter les sièges respectant les exigences du cahier des charges. Ainsi, les classes de composants sont représentées en utilisant le diagramme de définition de Blocs (Figure 8.3).

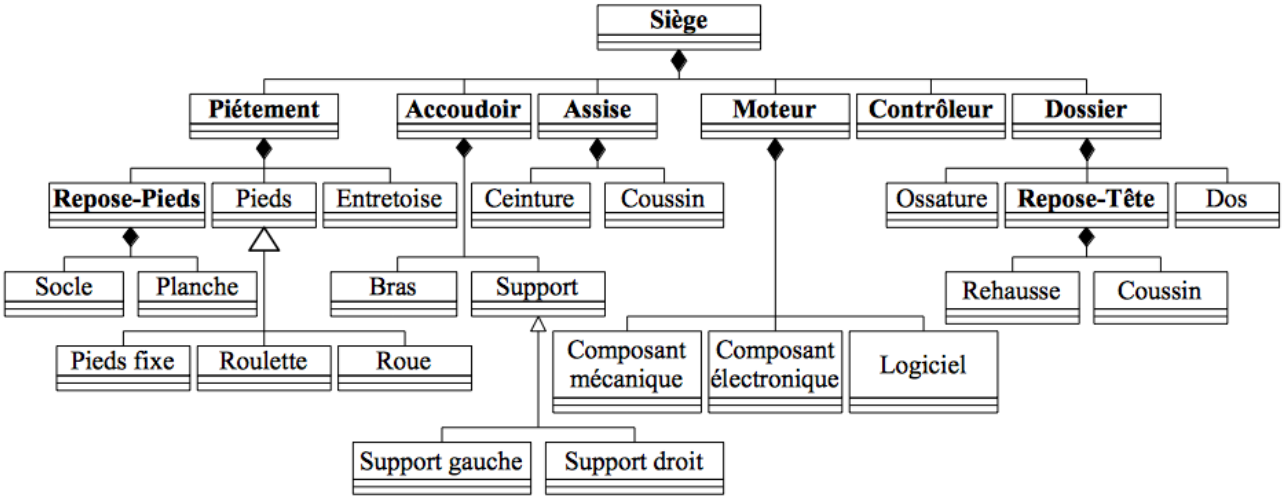

Figure 8.3 : Modèle SysML des composants d'un siège

Les classes de liaisons sont représentées en utilisant le diagramme de blocs internes (Figures 8.4, 8.5, 8.6, 8.7, 8.8, 8.9) et les fonctionnalités des produits sont représentées en utilisant le diagramme des exigences (Figure 8.10).

Le diagramme des composants, donne tous les composants du système sous une forme hiérarchique avec les différents niveaux. Le diagramme ci-dessus montre que le système est constitué de 6 classes de composants (Assise, Dossier, Piétement, Accoudoir, Moteur, Contrôleur). Chaque instance de ces 6 classes de composants est un sous-assemblage qui est à son tour composé d'autres classes de composants. Ce diagramme est constitué de 4 niveaux notés de 0 à 3. Nous retrouvons au niveau 0 le produit lui-même, au niveau 1 les 6 classes de composants Assise, Dossier, Piètement, Moteur, Contrôleur et Accoudoir, au niveau 2 les classes de composants Ceinture, Coussin, Pied, Repose-Pieds, Entretoise, Bras, Support, Repose-Tête, Dos, Ossature, Composant mécanique, composant électronique et logiciel. Il faut cependant, noter que le contrôleur est lui aussi un sous-ensemble de composants et peut contenir des composants électroniques et des logiciels. Pour simplifier, nous ne les avons pas représentés sur la Figure 8.3. Au dernier niveau (niveau 3), nous n'avons que des classes de pièces élémentaires Socle, Planche, Rehausse et Coussin. Ce diagramme ne donne pas les classes de liaisons entre les différentes classes de composants du système. Ces classes de liaisons sont représentées dans le modèle des liaisons par des digrammes de blocs internes.

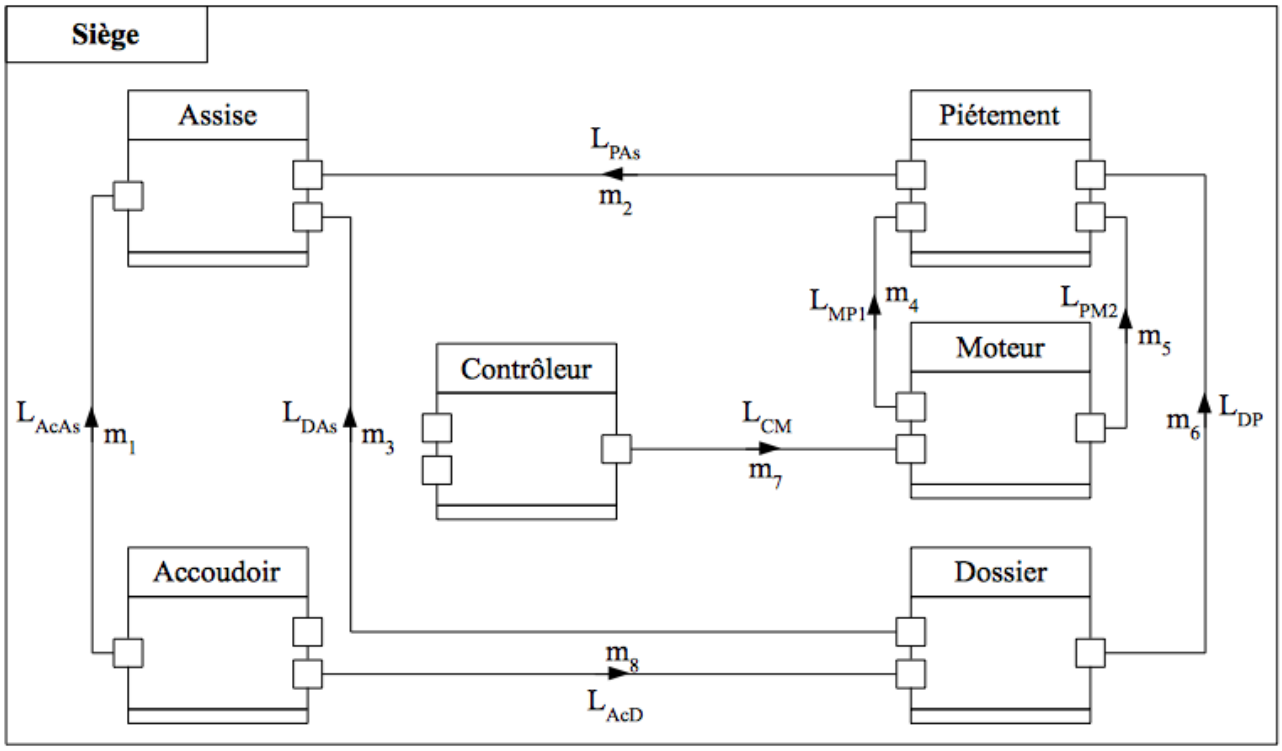

Figure 8.4 : Modèle SysML des liaisons entre les composants d'un siège

La Figure 8.4 montre les classes de liaisons entre les différentes classes de composants de niveau

1. Pour chaque classe de liaisons, seul son sens est indiqué. Le nombre d'occurrences (m_i) d'une liaison ainsi que sont type et les valeurs de ses autres propriétés ne seront connues que lors de l'instanciation de la classe de liaison. De la même manière que les composants de niveau 1, les classes de liaisons entre les composants de niveau inférieur peuvent également être représentées. Nous donnons dans la Figure 8.5, la représentation des classes de liaisons entre les classes de composants du sous-assemblage **Piétement**.

Les classes de liaisons (L_{IP1}, L_{IP2} et L_{IRP}) reliant les classes de composants du sous-ensemble Piétement sont représentées sur la Figure 8.5. Sur cette même figure, nous voyons également les classes de liaisons (L_{MP1}, L_{MP2}, L_{PAs} et L_{DP}) qui relient ce sous-ensemble aux autres sous-ensembles que compte le système et qui sont représentées sur la Figure 8.4. Cette figure permet de connaître le sous-composant du composant Piétement qui est réellement relié au composant Moteur, Assise ou Dossier. Nous avons également montré la classe de liaisons (L_{IRP}) qui relient les sous-composants (Socle et Planche) du composant Repose-Pieds qui est lui aussi un sous-ensemble.

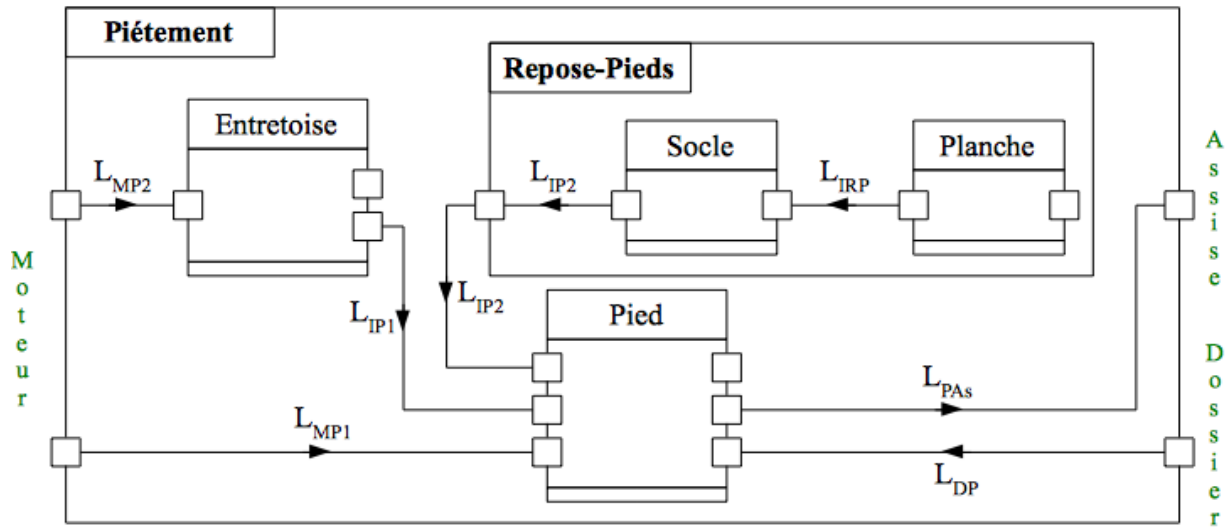

Figure 8.5 : Modèle SysML des liaisons entre les composants du sous-ensemble Piétement

La Figure 8.6 représente les classes de liaisons (L_{ID1}, L_{ID2} et L_{IRT}) entre les classes de composants du sous-ensemble Dossier. Les classes de liaisons (L_{DP}, L_{DAs}, L_{AcD}) entre le Dossier et d'autres classes de composants de niveau 1 sont également représentées. Nous avons également montré la classe de liaisons (L_{IRT}) qui relient les sous-composants (Rehausse et Coussin) du composant Repose-Tête qui est lui aussi un sous-ensemble.

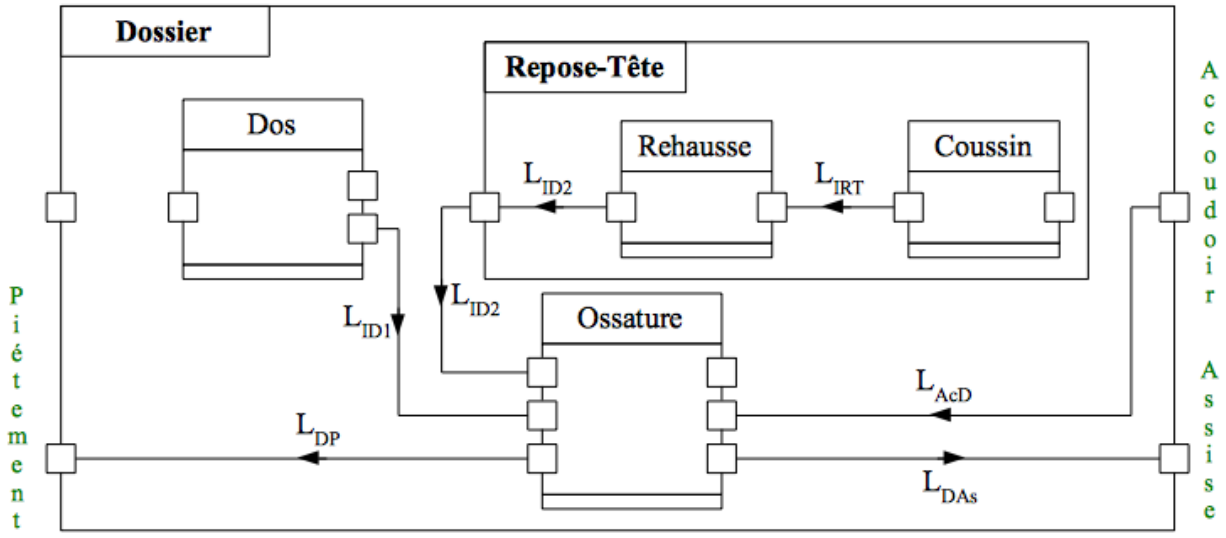

Figure 8.6 : Modèle SysML des liaisons entre les composants du sous-ensemble Dossier

La Figure 8.7 représente la classe de liaisons (L_{IAs1}) entre les classes de composants du sous-ensemble Assise. Les classes de liaisons (L_{PAs}, L_{DAs}, L_{AcAs}) entre l'Assise et d'autres classes de composants de niveau 1 sont également représentées.

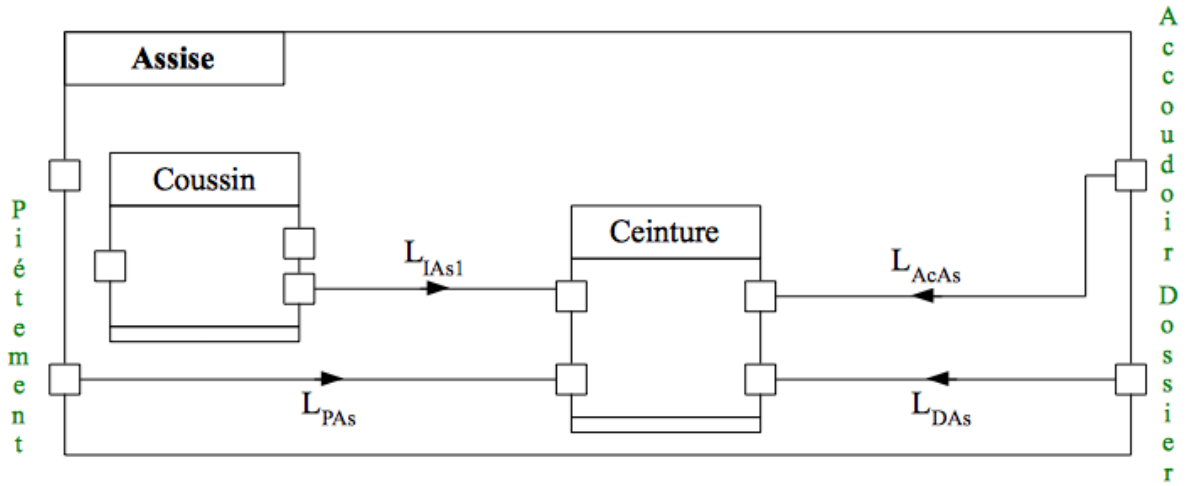

Figure 8.7 : Modèle SysML des liaisons entre les composants du sous-ensemble Assise

La Figure 8.8 représente la classe de liaisons (L_{IAc1}) entre les classes de composants du sous-ensemble Accoudoir. Les classes de liaisons (L_{AcAs}, L_{AcD}) entre l'Accoudoir et d'autres classes de composants de niveau 1 sont également représentées.

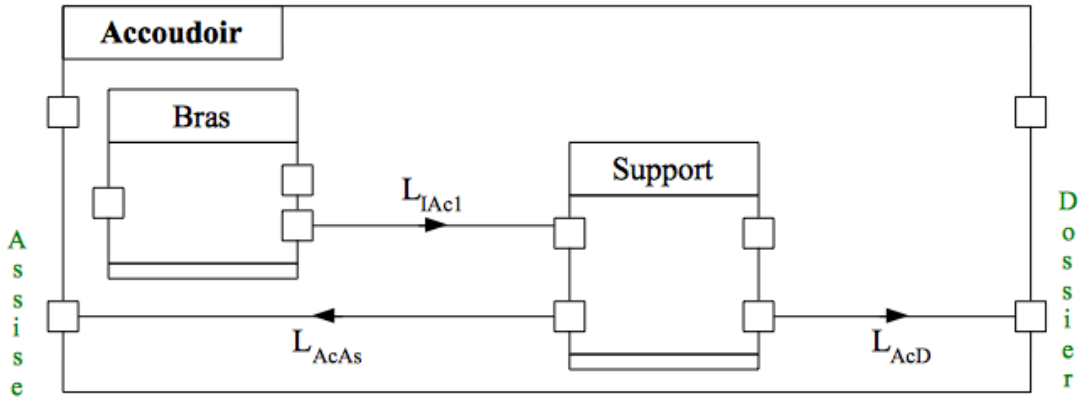

Figure 8.8 : Modèle SysML des liaisons entre les composants du sous-ensemble Accoudoir

La Figure 8.9 représente les classes de liaisons (L_{IM1}, L_{IM2}) entre les classes de composants du sous-ensemble Moteur. Les classes de liaisons (L_{CM}, L_{MP1}, L_{MP2}) entre le Moteur et d'autres classes de composants de niveau 1 sont également représentées.

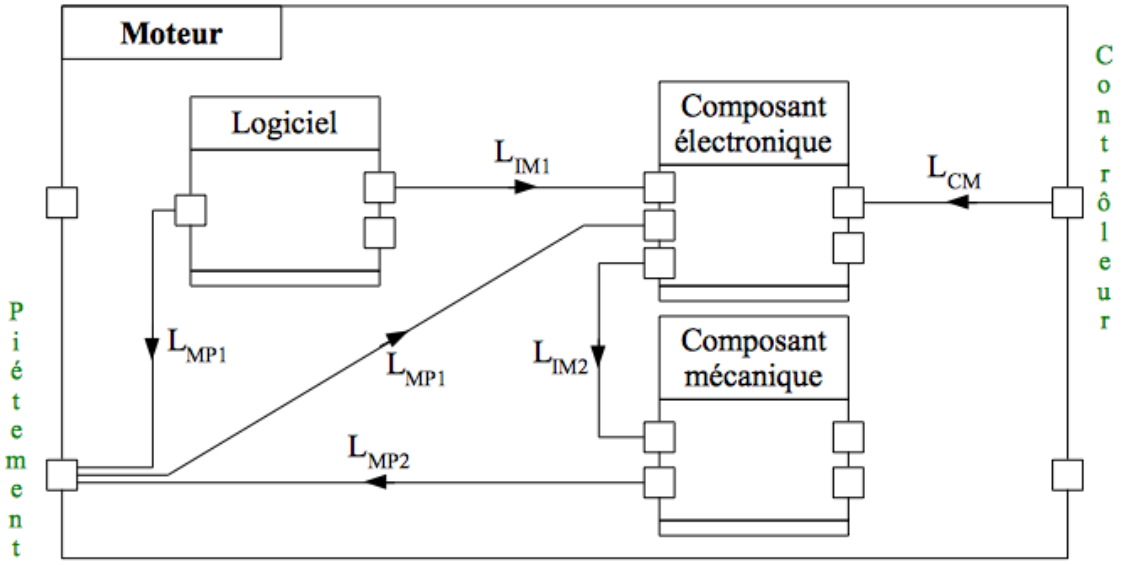

Figure 8.9 : Modèle SysML des liaisons entre les composants du sous-ensemble Moteur

Les Figures 8.4, 8.5, 8.6, 8.7, 8.8, 8.9 ainsi que le modèle des liaisons du sous-ensemble Contrôleur qui n'est pas représenté ici donnent le modèle des liaisons du système. Dans la Figure 8.4, les classes de liaisons entre les différentes classes de composants du système sont représentées sans tenir compte des classes de liaisons au sein des sous-systèmes du système. Dans les Figures 8.5, 8.6, 8.7, 8.8 et 8.9 les modèles de liaisons des sous-systèmes Piétement, Dossier, Assise, Accoudoir et Moteur sont respectivement représentés. Le modèle SysML du système est complété par le diagramme des exigences qui permet de représenter les exigences fonctionnelles du système. Ce diagramme est représenté sur la Figure 8.10.

Sur la Figure 8.10, les différentes fonctionnalités principales de système à concevoir sont représentées. Nous avons 2 fonctionnalités principales que sont le transport d'une personne (Ft_1) et son assistance (Ft_2). Pour transporter une personne, il faut pouvoir la contenir et rouler d'un point A à un point B, d'où les deux sous-fonctionnalités Rouler (Ft_{11}) et Contenir (Ft_{12}). La fonctionnalité d'assistance est subdivisée en 3 sous fonctionnalités : la localisation de la position exacte de la personne ou celle d'une adresse de destination (Ft_{21}), la conduite automatique (Ft_{22}) et la possibilité d'émettre et de recevoir des appels (Ft_{23}). Il faut aussi remarquer que pour assister une personne il faut la transporter d'abord.

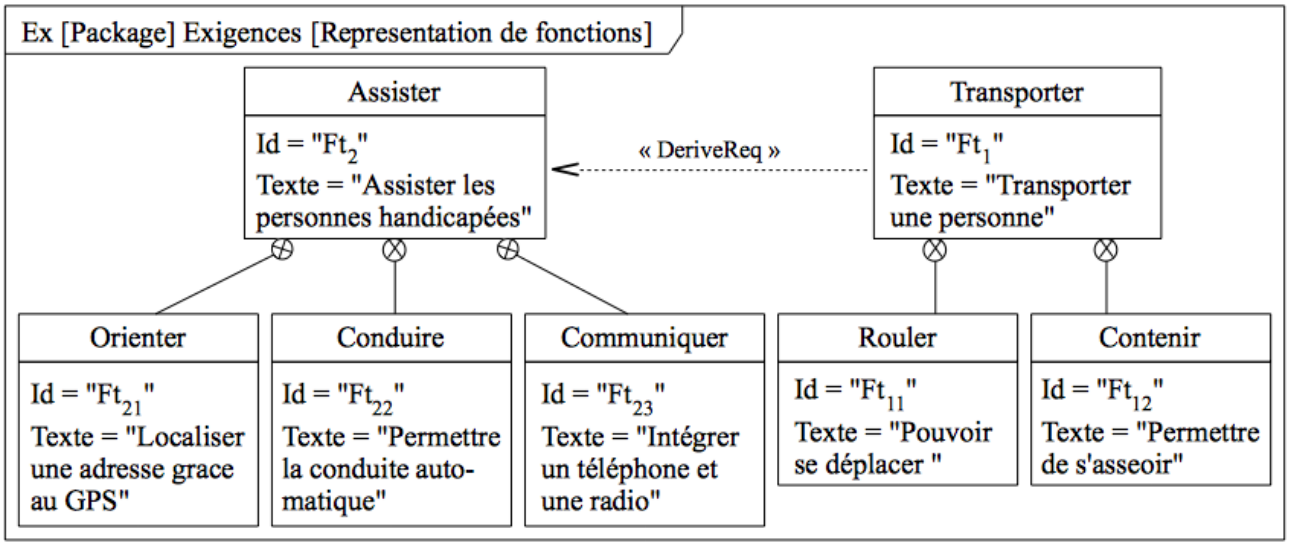

Figure 8.10 : Modèle des fonctionnalités du système

Le diagramme de la Figure 8.10 donne le modèle fonctionnel du système et complète son modèle SysML. Ainsi, pour aller vers l'ingénierie de performances comportementales, la matrice X-CDSM correspondant à ce modèle est créée.

8.3.2.3.2. Matrice sémantique conceptuelle de l'objet siège

A partir des diagrammes représentés sur les Figures 8.3, 8.4, 8.5, 8.6, 8.7, 8.8 et 8.9 on a toutes les classes de composants ainsi que toutes les classes de liaisons qui les relient. A partir de ces diagrammes donnant le modèle structurel du système, le logiciel Product-BPAS peut créer la matrice sémantique de ce système. Pour illustrer le module maintenabilité, nous entrons dans le logiciel : le nom du système, son nombre de niveaux, son nombre de composants pour chaque niveau, le nom de chaque composant et le numéro du sous-ensemble auquel il appartient ainsi que les liaisons entre ces composants. Les fenêtres permettant d'entrer ces informations sont représentées sur les Figures 8.11, 8.12 et 8.13.

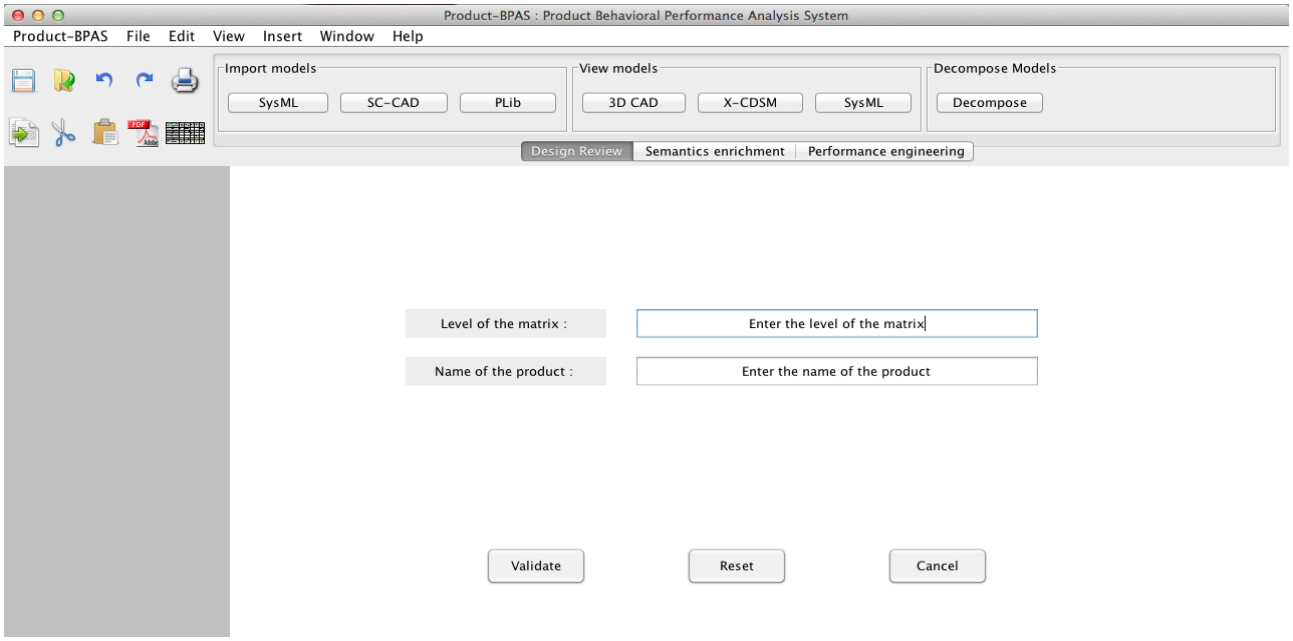

Figure 8.11 : Fenêtre permettant de donner le nombre de niveaux et le nom du produit

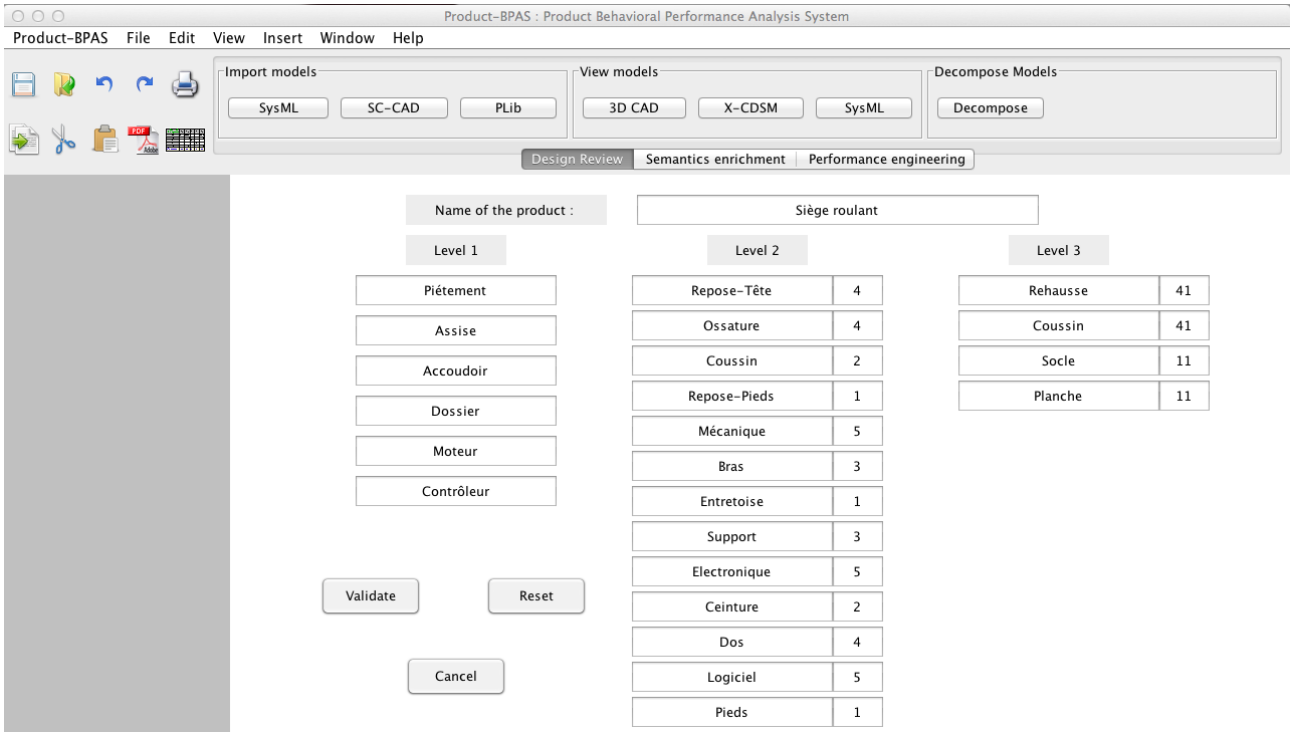

Figure 8.12 : Fenêtre permettant de donner les noms des composants et leurs niveaux

A partir de ces informations, le logiciel crée la matrice X-CDSM et la nomenclature du produit comme le montre la Figure 8.13.

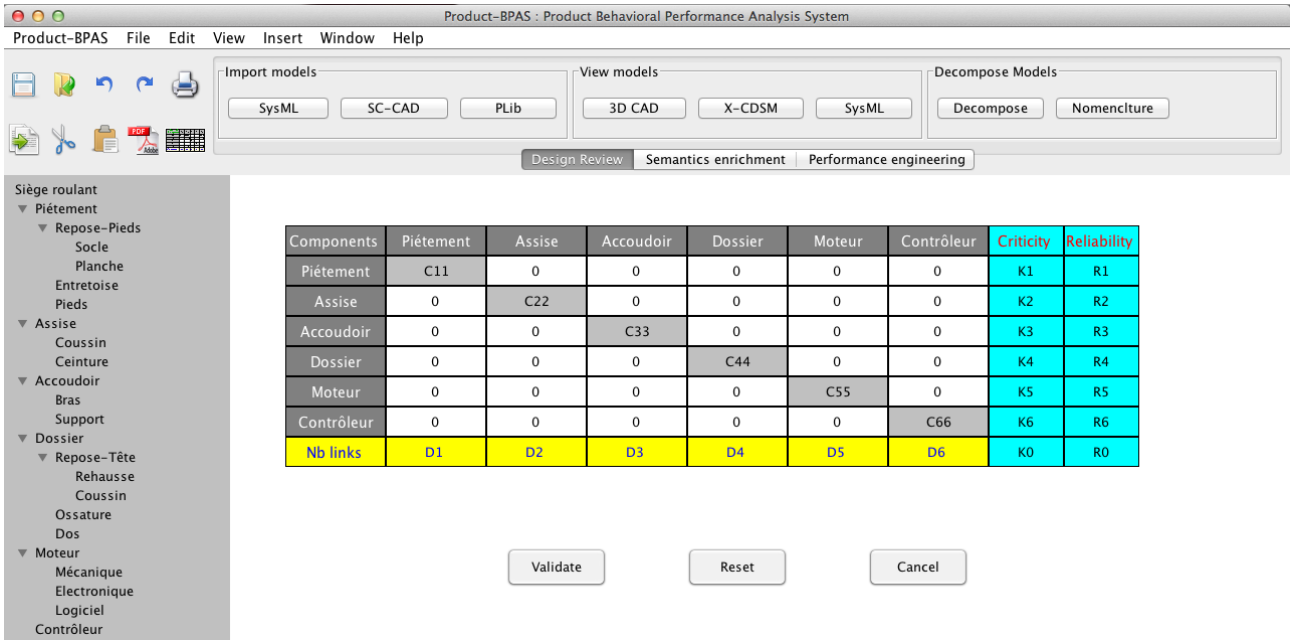

Figure 8.13 : Fenêtre permettant d'instancier ne matrice X-CDSM

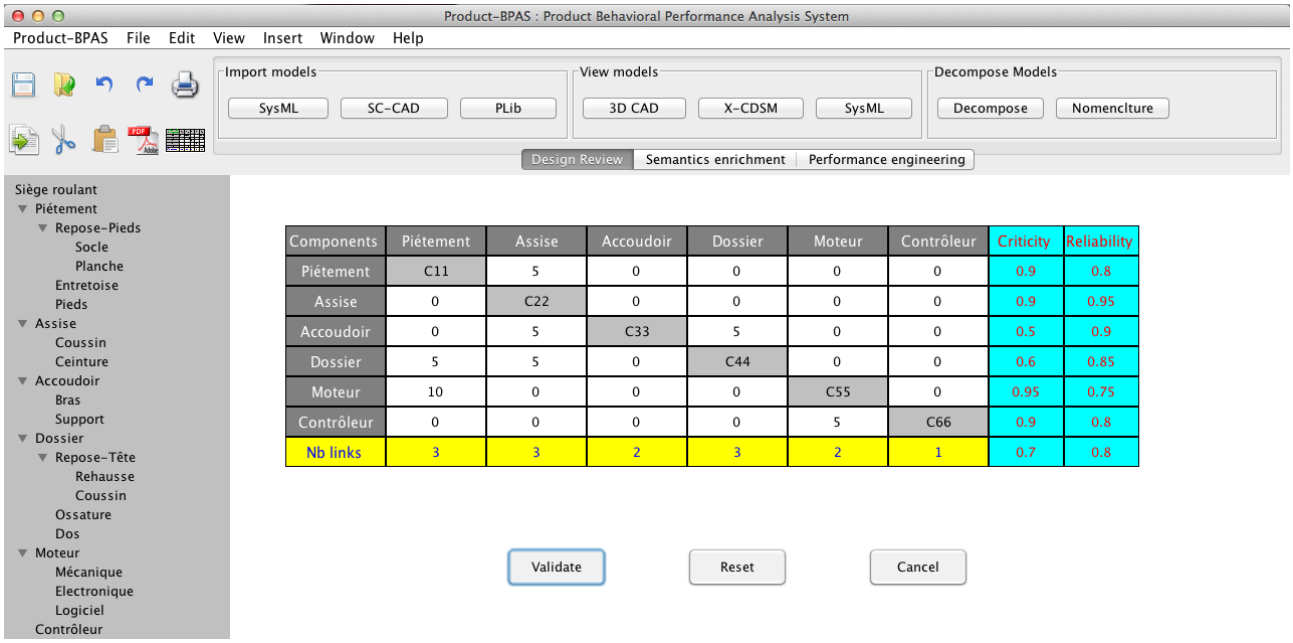

Figure 8.14 : Matrice d'une instance de produit

La Figure 8.14 montre la matrice instanciée représentant donc une instance de produit. A partir de cette matrice, le logiciel peut chercher le chemin de désassemblage d'un composant et calculer le temps nécessaire pour le faire comme le montre les Figures 8.15 et 8.16.

8.3.2.4. Evaluation de la Maintenabilité

Dans cette section, nous considérons l'indicateur temps de remplacement d'un composant défaillant. Pour établir ce temps il faut d'abord connaître les composants à enlever pour atteindre ce

dernier. Ainsi, pour un composant donné, le système donne la liste des composants qu'il faut enlever pour l'atteindre dans le bon ordre ainsi que le temps nécessaire pour le faire (Figure 8.16). Le système demande d'abord de lui indiquer le composant à remplacer, si le composant peut être enlevé directement (sans avoir besoin d'enlever un autre composant) le système nous le dit et nous donne son temps de remplacement comme le montre la Figure 8.15.

Il faut également remarquer que le système compte le nombre de liaisons auxquelles participe chaque composant et renseigne la ligne Nb_Links (Figures 8.15 et 8.16).

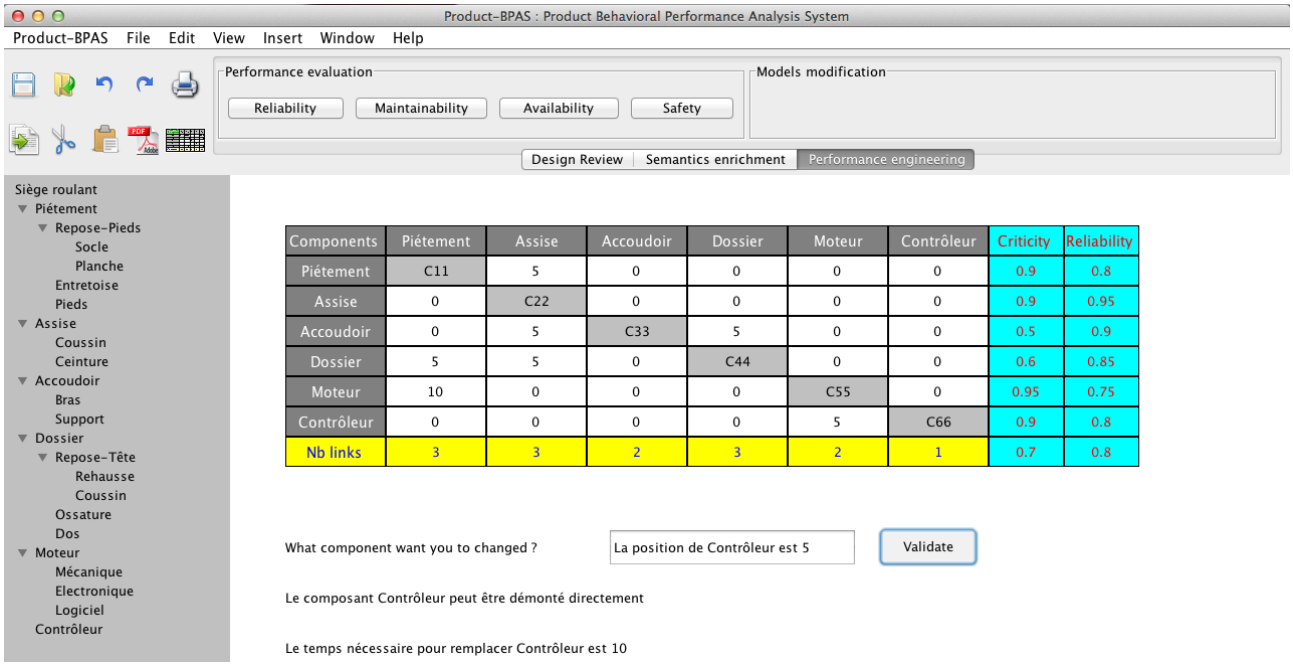

Figure 8.15 : Résultat du calcul du temps de démontage du composant Contrôleur

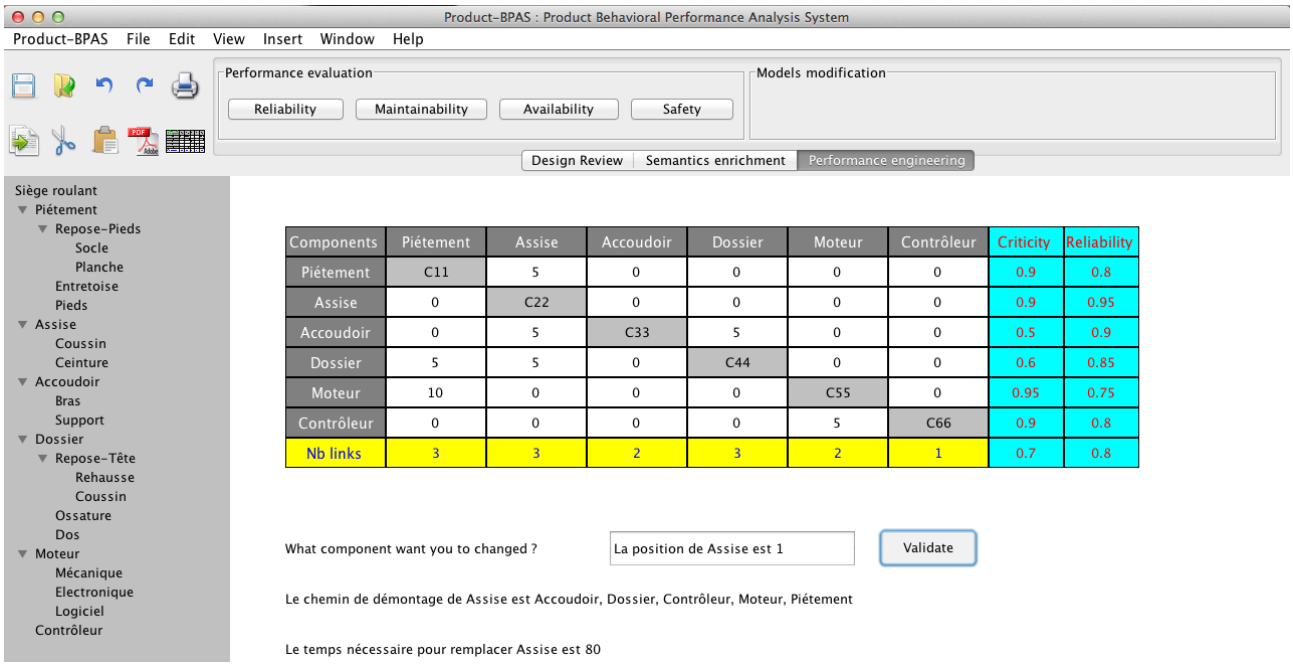

Figure 8.16 : Résultat du calcul du temps de démontage du composant Assise

Conclusion

Dans ce chapitre nous avons présenté l'implémentation du logiciel Product-BPAS. C'est un logiciel qui utilise des données provenant d'autres logiciels. Il est articulé autour de trois modules dont chacun implémente un certain nombre de fonctionnalités. Pour le développer, nous avons choisi le langage Java sous l'environnement de développement Netbeans. Les données sont enregistrées dans une base de données relationnelle-objet crée dans le SGBD PostgreSQL. C'est un logiciel autonome (standalone) qui peut néanmoins être enrichi avec des plugins. Nous avons également présenté un cas d'étude pour lequel le DSE est conçu à partir du cahier des charges, puis le modèle SysML (diagramme de définition blocs pour représenter les classes de composants, diagrammes de blocs internes pour représenter les classes de liaison et diagramme des exigences pour représenter les exigences fonctionnelles) de ces instances du DSE est créé, ensuite la matrice X-CDSM est conçue et enfin le temps de remplacement d'un composant défaillant est calculé et son chemin de désassemblage est identifié en guise d'illustration. Il faut remarquer que le logiciel Product-BPAS n'est utilisé qu'après conception du modèle SysML. Dans l'exemple développé, les informations contenues dans ce modèle sont entrées directement par l'utilisateur. Le module permettant d'importer le format XML d'un modèle SysML, de l'exploiter et de concevoir la matrice X-CDSM avec les informations qu'il contient est en cours de développement de même que tous les autres modules du logiciel.

Conclusion générale et Perspectives

Conclusion générale et Perspectives

Dans cette thèse nous avons présenté la problématique de validation de maquettes numériques de produits complexes passant par la modélisation sémantique conceptuelle et l'ingénierie de performances comportementales en phase de conception. Elles visent à mettre en place un cadre permettant aux concepteurs de concevoir des produits performants, robustes, sûrs de fonctionnement et à moindre coût. Le processus de conception couvre toutes les parties allant du cahier des charges aux prototypes numériques satisfaisant les exigences structurelles, fonctionnelles et comportementales. Elle est essentiellement composée de la conception conceptuelle, de la modélisation conceptuelle et de l'évaluation de performances comportementales. Nous avons, par conséquent, proposé des approches pour améliorer chacune de ces étapes dans le but d'optimiser les produits mécatroniques.

A. Récapitulatif

L'utilité et l'utilisation des produits mécatroniques sont de plus en plus importantes dans la vie de tous les jours. Ces produits interviennent dans pratiquement tous les domaines de la vie : santé, industrie, transport, communication, loisirs, … Ainsi, leur adaptation aux besoins des utilisateurs, l'amélioration de leur fiabilité, la facilitation de leur maintenance, l'augmentation de leur sécurité, l'accroissement de leur disponibilité, la réutilisation de leurs composants en fin de vie sont des chantiers sur lesquels beaucoup de chercheurs ont travaillé ces dernières années. Nos contributions dans cette thèse ont essentiellement porté sur la phase amont du développement des produits mécatroniques. Ainsi, nous avons proposé :

1. une approche de conception sémantique conceptuelle permettant à partir d'un cahier des charges d'identifier toutes les familles de produits susceptible de répondre aux exigences qui y sont exprimées. Elle propose une collaboration étroite et très tôt des spécialistes de compétences différentes basée sur une identification précoce des tâches de conception à exécuter. Elle aboutit au domaine de solutions éligibles contenant les familles de produits susceptibles de satisfaire les exigences du cahier des charges ;
2. une méthodologie de modélisation sémantique conceptuelle basée sur une extension des DSM. Elle est basée sur l'intégration de la modélisation orientée-objet dans les matrices DSM. Ainsi, les composants et liaisons sont représentés sous forme de classes d'objet permettant de rendre ces matrices conceptuelles et génériques. Une telle matrice peut alors représenter plusieurs instances de produits appartenant à une même famille. Elle produit une matrice X-CDSM pour chaque famille de produits appartenant au domaine de solutions éligibles ;

3. une description d'une nouvelle génération de systèmes de CAO appelés CAO sémantique conceptuelle. La CAO-SC résulte de l'adaptation de la CAO classique à la méthodologie de conception sémantique conceptuelle. Elle propose de nouvelles fonctionnalités à intégrer dans les systèmes de CAO classiques pour pouvoir implémenter un modèle conceptuel sans pour autant être obligé de l'instancier auparavant. Elle permet d'implémenter les matrices X-CDSM des différentes familles de produits dans un système de CAO ;
4. une méthodologie d'ingénierie de performances comportementales. Elle est basée sur trois concepts essentiels que sont l'Evaluation, l'Optimisation et la Validation (EOV). C'est la phase d'identification des instances de produits devant être retenues ou rejetées. Elle part d'un ensemble de matrices X-CDSM pour en extraire les instances de produits satisfaisant les exigences structurelles, fonctionnelles et comportementales. Nous avons également proposé une amélioration de la formule de calcul du temps de remplacement de composants défaillants, de la méthodologie de recherche du chemin optimal de désassemblage et une méthodologie de calcul de la fiabilité de produits basée sur leurs exigences fonctionnelles ;
5. une description des fonctionnalités et de l'architecture d'un logiciel permettant d'instrumenter la modélisation sémantique conceptuelle et l'ingénierie de performances comportementales. Ce logiciel nommé Product-BPAS prend en entrée un modèle SysML sous format XML et/ou CAO-CS sous format STEP avant de concevoir la matrice X-CDSM correspondante. Cette dernière est utilisée pour évaluer les instances de produits.

Ces différents points donnent les contributions scientifiques et techniques faites durant les travaux de cette thèse. Ils nous permettent également de mesurer le niveau de satisfaction des réponses apportées aux questions de recherche de la problématique.

B. Etat de résolution des questions de recherche

Les questions de recherche posées dans la Section 1.2.4 de l'introduction générale (Chapitre 1) sont au nombre de trois dont la deuxième peut être décomposée en trois sous-questions. Elles ont pour objectifs de décrire les étapes du processus permettre de passer du cahier des charges aux instances de produits satisfaisant les exigences. Dans ce présent rapport de mémoire, nous avons apporté des réponses à chacune de ces questions. Ainsi, nous discutons ces réponses dans cette section.

B.1. Comment à partir d'un cahier des charges identifier les produits susceptibles de satisfaire les exigences et contraintes tout en respectant les normes internationales ? La réponse apportée à cette question est décrite dans le Chapitre 3. Elle passe par trois étapes principales : d'abord classifier

les exigences et contraintes, puis identifier les sous-systèmes, enfin créer le domaine de solutions éligibles. Elle est appelée conception sémantique conceptuelle et s'appuie sur la conception collaborative, la conception concurrente et la conception paramétrique.

B.2. Comment modéliser ces produits en utilisant :

B.2.1. les outils de modélisation orienté-objet ? Pour répondre à cette question nous avons utilisé dans le Chapitre 4 le langage SysML. Ainsi, le diagramme de définition de blocs est utilisé pour modéliser les classes de composants, des améliorations sont apportées au diagramme de blocs internes pour lui permettre de modéliser les classes de liaisons et le diagramme d'exigences est utilisé pour modéliser les exigences fonctionnelles des produits.

B.2.2. les matrices DSM ? Dans le même chapitre (Chapitre 4), une amélioration des matrices DSM est proposée pour répondre à cette question. En effet, une intégration de la modélisation orientée-objet dans les DSM est proposée. Les nouvelles matrices obtenues sont appelées X-CDSM. Dans une telle matrice les cellules contiennent les descriptions (propriétés, méthodes, contraintes) des classes de composants (cellules diagonales) et des classes de liaisons (cellules non diagonales).

B.2.3. les logiciels de CAO ? Pour représenter les produits en environnement CAO, nous avons donné dans le Chapitre 5 les fonctionnalités d'une nouvelle génération de CAO appelée CAO sémantique conceptuelle. Elle permet la représentation de modèles génériques conceptuels par les systèmes de CAO. Ainsi, l'instanciation se fait dans le système de CAO.

B.3. Comment automatiser l'évaluation des performances comportementales des produits complexes en phase de conception ? La description du processus d'automatisation est faite dans le Chapitre 7. Elle intègre également l'optimisation des instances de produits pour améliorer leurs performances. L'évaluation d'une instance donnée se fait en deux étapes : l'évaluation de l'instance pour chaque domaine et son évaluation inter-domaine pour la validation finale.

B.4. Limites : Il faut cependant noter un certain nombre de points à améliorer et qui seront répertoriés dans les perspectives. Nous en citer principalement 3 :

1. L'adaptation des métriques de calcul de la disponibilité, de la sécurité et de la recyclabilité à la MSC ainsi les algorithmes permettant de implémenter dans le Product-BPAS n'est pas encore faite ;

2. Seul le calcul du temps de remplacement de composants défaillants est implémenté dans le logiciel ;

3. Les spécifications techniques de la CAO-SC restent à être améliorées et son architecture à définir.

C. Perspectives

Ces travaux et les contributions qu'ils ont apportées ouvrent d'autres perspectives et appellent à relever d'autres défis. Certaines des perspectives ouvertes sont une suite logique et directe des résultats de ces travaux alors que d'autres sont une amélioration des méthodologies proposées ou un approfondissement des résultats obtenus.

C.1. Développement d'application instrumentant les contributions

Dans ce manuscrit, nous avons proposé de nouvelles fonctionnalités à ajouter aux systèmes de CAO classiques pour leur permettre de prendre en compte la modélisation conceptuelle. Nous avons également décrit les fonctionnalités et l'architecture d'un outil logiciel permettant d'instrumenter la modélisation sémantique conceptuelle et l'ingénierie de performances comportementales. Ainsi, la suite dans le court terme est :

1. le développement du logiciel Product-BPAS. L'interface graphique est déjà développée et les algorithmes de recherche de chemin et de calcul du temps de remplacement de composants défaillant sont implémentés. D'autres modules sont en cours d'implémentation sous forme de sujets de stage de Master 2 d'étudiants de Génie logicielle de l'Université Assane SECK de Ziguinchor. La collaboration avec une entreprise de développement d'applications informatique est aussi envisagée ;
2. Affinement de la description du fonctionnement de la CAO-SC et proposition de son architecture. Puis, développement de nouveaux systèmes de CAO-SC et/ou intégration des fonctionnalités de la CAO-SC dans des systèmes de CAO classiques existants. Sur ce point, nous envisageons une collaboration avec des sociétés de conception de systèmes de CAO.

C.2. Amélioration et approfondissement des résultats

Les contributions apportées durant cette thèse portent essentiellement sur la CSC, la MSC et l'IPC. Ainsi, la suite envisagée de ces travaux portera sur :

1. l'amélioration et l'adaptation des formules de calcul des métriques de la disponibilité, de la sécurité et de la recyclabilité à la MSC comme c'est le cas de la maintenabilité et de la fiabilité ;
2. l’utilisation des réseaux bayésiens pour affiner les relations probabilistes de dépendance/défaillance des composants ;
3. l’analyse multicritères pour affiner les arbitrages entre critères comportementaux ;
4. l'extension des fonctionnalités du Product-BPAS pour y intégrer la CSC.

Bibliographie

---------- A ----------

[Ait-Kadi, 2007] D. Ait-Kadi : « Fiabilité des systèmes » Notes de cours, département de génie mécanique, Université Laval, 2007.

[Andrews, 2012] J. Andrews, H. Jin, C. Séquin : « Interactive Inverse 3D Modeling » Computer-Aided Design & Applications, Volume 9, Issue 6, Pages 881-900, 2012.

[Aprim, 2013] APRIM & Associés : « Atlas des compétences mécatronique des Alpes Maritimes » CCI Nice Côte d'Azure, Septembre 2013.

[Audibert] L. Audibert : « Introduction à la modélisation objet » http://laurent-audibert.developpez.com/Cours-UML/?page=introduction-modelisation-objet.

---------- B ----------

[Bártolo, 2001] H. M. G. Bártolo, P. J. S. Bártolo : « Computer-Aided Concurrent Design » 7th Annual ARCOM Conference, A. Akintoye (ed), 5-7 September 2001, Association of Researchers in Construction Management, University of Salford. Vol. 1, Pages 731-7399, 2001.

[Bathelt, 2007] J. Bathelt, J. Meile : « Computer Aided Methods Supporting Concurrent Engineering when designing Mechatronic Systems Controlled by a PLC » In Proceedings of International Conference on Manufacturing Automation (ICMA07), Singapore, May 2007.

[Bonneaud, 2008] S. Bonneaud : « Des agents-modèles pour la modélisation et la simulation de systèmes complexes : application a l'ecosystemique des pêches » Thèse de doctorat de l'Université de Bretagne occidentale - Brest, Soutenue le 14 Octobre 2008.

[Bolton, 1999] W. Bolton : « Mechatronics: Electrical Control Systems in Mechanical and Electrical Engineering » 2nd Ed., Addison-Wesley Longman, Harlow, England, 1999.

[Bouazza, 1995] M. Bouazza : « Le langage Express » Hermes Sciences Publicat. (21 novembre 1995), ISBN-10: 2866015037, ISBN-13: 978-2866015039, 1995.

[Boujut, 1993] J. F. Boujut : « Un exemple d'intégration des fonctions métier dans les systèmes de CAO : la conception de pièces forgées tridimensionnelles » Thèse de doctorat, Institut national polytechnique de Grenoble, 1993.

[Bounie] D. Bounie : « Ingénierie de projet en conception » Polytech'Lille, IAAL - L'usine agro-alimentaire.

[Brown, 1998] D. C. Brown : « Intelligent Computer Aided-Design » Encyclopedia of Computer

Science and Technology, ed. Williams and Sochats, 1998.

[Browning, 2001] T. R. Browning : « Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions » IEEE Transaction on Engineering Management, Volume 48, Issue 3, Pages 292–306, August 2001.

---------- C ----------

[CAD-Magazine N° 180] « Modélisation : paramétrique Vs directe » CAD-Magazine N° 180, Juillet-Août 2014.

[Caillaud, 2011] E. Caillaud : « Méthodes de conception » Cours de conception, 5 mars 2011.

[Castaneda, 2009] G. A. Perez Castaneda : « Evaluation par simulation de la sûreté de fonctionnement de systèmes en contexte dynamique hybride » Thèse de doctorat de l'Institut National Polytechnique de Lorraine, Soutenue 30 Mars 2009.

[Casner, 2013] D. Castner, R. Houssin, D. Knittel, J. Renaud : « Une démarche de conception et d'optimisation de systèmes mécatroniques à partir de l'optimisation multidisciplinaire et basée sur le retour d'experiences » 21ème Congrès Français de Mécanique, Bordeaux, 26-30 Août 2013.

[Choley, 2005] J.-Y. Choley : « Mécatronique : une nouvelle démarche de conception des systèmes complexes » Technologies et Formations. Delagrave Editions S.A., October 2006, Numéro 127, Pages 29-35, 2005.

[Chambolle, 1999] F. Chambolle : « Un modèle produit piloté par les processus d'élaboration : Application au secteur automobile dans l'environnement STEP » Thèse de doctorat de l'Ecole Centrale Paris, Soutenue le 29 Avril 1999.

[Chandrasekaran, 1990] B. Chandrasekaran : « Design Problem Solving: A Task Analysis » AI Magazine, Vol. 11(4), pp. 59-71, 1990.

[Chandrasegaran, 2013] S. K. Chandrasegaran, K. Ramani, R. D. Sriram, I. Horvath, A. Bernard, R. F. Harik, W. Gao : « The evolution, challenges, and future of knowledge representation in product design systems » Computer-Aided Design, Volume 45, Pages 204-228, 2013.

[Chekh-Waiss, 2013] H. S. Chekh-Waiss, P. Mitrouchev, M. Tollenaere : « Indicateurs d'évaluation de la dés-assemblité des produits » 21ème Congrés Français de Mécanique, Bordeaux, France, 26 – 30 Août 2013.

[CI1] ATS 21 : « Analyse fonctionnelle et structurelle » Sciences industrielles pour l'ingénieur, Savoir communiquer au sujet d'un système.

[CIMI, 2011] Les rencontres du CIMI : « Sûreté de fonctionnement des installations industrielles »

18 Octobre 2011.

[Comerford, 1994] R. Comerford : « Mecha...what? » Spectrum IEEE, Vol. 31, N° 8, Pp. 46-49, August 1994.

[Cottet, 2005] F. Cottet, E. Grolleau : « Systèmes temps réel de contrôle-commande, conception et implémentation » Dunod, 2005.

[Coulibaly, 2007] A. Coulibaly, B. Mutel, D. Ait-Kadi : « Product modeling framework for behavioral performance evaluation at design stage » Computers In Industry, August 2007, Volume 58, Issue 6, Pages 567-577, 2007.

[Coulibaly, 2008a] A. Coulibaly : « Modélisation Sémantique et Evaluation de Performances, Comportementales de Produits en conception » Habilitation à diriger des recherches, Université Louis Pasteur - Strasbourg I, Soutenue le 19 Juin 2008.

[Coulibaly, 2008b] A. Coulibaly, R. Houssin, and B. Mutel : « Maintainability and safety indicators at design stage for mechanical products » Computer In Industry, Volume 59(Issue 5):Pages 438–449, May 2008.

[Coulibaly, 2010] A. Coulibaly, F. De Bertrand De Beuvron, J. Renaud : « Maintainability Assessment at early design stage using advanced CAD systems » Proceedings of IDMME - Virtual Concept 2010 Bordeaux, France, October 20 - 22, 2010.

---------- D ----------

[Da Silveira, 2003] M. Da Silveira : « Distribution avec redondance partielle de modèles à événements discrets pour la supervision de procédés industriels » Thèse de doctorat de l'Université Paul Sabatier de Toulouse, Soutenue le 13 Octobre 2003.

[Dang, 2014] Q. V. Dang : « Similarité dans des modèles BRep paramétriques : Détection et applications » Thèse de doctorat de l'Université de Toulouse, Soutenue le 22 Septembre 2014.

[Daniel, 2009] M. Daniel : « Modélisation géométrique : Introduction » Cours Master SIS, Septembre 2009.

[Daniel, 2014] M. Daniel : « Cours, Modélisation géométrique » Cours Master SIS, Octobre 2014.

[Demri, 2009] A. Demri : « Contribution à l'évaluation de la fiabilité d'un système mécatronique par modélisation fonctionnelle et dysfonctionnelle » Thèse de doctorat de l'université d'Angers, Soutenue en Septembre 2009.

[Deniaud, 2011] I. Deniaud, J. P. Micaellli, E. Bonjour : « Innovation produit et innovation organisationnelle : quel apport de l'ingénierie système ? » Congrès International de Génie Industriel,

Montréal, 2011.

[Devalan, 2010] P. Devalan : « Intraduction à la mécatronique » http://www.techniques-ingenieur.fr/base-documentaire/mecanique-th7/mecatronique-42509210/introduction-a-la-mecatronique-bm8000/composants-et-systemes-mecatroniques-bm8000niv10003.html, 2010.

[Diehl, 2005] J. C. Diehl, J. C. Brezet : « International ecodesign education: Personalised design knowledge transfer » International Conference on Engineering Design (ICED 2005) Melbourne, August 15-18, 2005.

[Dixon, 1988] J. R. Dixon : « Designing with features: building manufacturing knowledge into more intelligent CAD systems » ASME Manufacturing International Conference, Atlanta, 1988.

[Dobre, 2010] D. Dobre : « Contribution A La Modélisation D'un Système Interactif D'aide A La Conduite D'un Procédé Industriel » Thèse de Doctorat de l'Université Henri Poincaré, Nancy 1, Spécialité Automatique, Traitement du Signal et Génie Informatique, Soutenue le 15 Novembre 2010.

---------- E ----------

[Eder, 2013] W. Ernst Eder : « Conceptualize-Design Enhancement Of Systematic Design Engineering Method » Proceedings of the Canadian Engineering Education Association, Montreal, Quebec, Canada, June 17-20, 2013.

[ElFeki, 2011] M. El Feki : « Analyse et synthèse de tolérance pour la conception et le dimensionnement des systèmes mécatroniques » Thèse de doctorat de l'Université de Lyon, Soutenue le 05 Juillet 2011.

[El Khalkhali, 2002] I. El Khalkhali : « Système intégré pour la modélisation, l'échange et le partage des données de produits » Thèse de doctorat de l'INSA de Lyon, Soutenue le 8 Octobre 2002.

[Eriksson, 2003] D. M. Eriksson : « A framework for the constitution of modelling processes: A proposition » European Journal of Operational Research, Volume 145, p. 202-215, 2003.

---------- F ----------

[Fontan, 2008] B. Fontan : « Méthodologie De Conception De Systèmes Temps Réel Et Distribués En Contexte Uml/Sysml » Thèse de Doctorat de l'université de Toulouse 3 - Paul Sabatier, Discipline : Informatique, Soutenue le 17 janvier 2008.

---------- G ----------

[Gausemeier, 2003] J. Gausemeier, S. Möhringer : « New Guideline VDI 2206 – A flexible procedure model for the design of mechatronic systems » Proceedings of the 14th International Conference on Engineering Design (ICED), Stockholm, 2003.

[Gaye, 2011] Kh. Gaye : « Système d'information et modèle organisationnel pour le recyclage des produits manufacturés » Thèse de doctorat, Université de Strasbourg, Soutenue le 21 Janvier 2011.

[Gero, 1989] J. S. Gero : « Artificial Intelligence in design » Computational Mechanics Publications, Springer-Verlag, Southampton, UK, 1989.

[Ghemraoui, 2009] R. Ghemraoui : « Méthodologie de conception innovante intégrant la sécurité des utilisateurs : application aux liaisons tracteur-outils » Thèse de doctorat, Ecole nationale supérieure de Cachan, soutenue le 17 Novembre 2009, N° ENSC-2009-190, 2009.

[Gies] V. Gies : « Systèmes mécatroniques asservis : Introduction » ENSTA.

[Girardot, 2013] J.-J. Girardot, M. Roelens : « Langages et concepts de programmation : Introduction à la programmation en langage C » Ecole nationale des mines de Saint-Etienne, Cours 1A 2013-2014.

[Gruber, 1993] T. Gruber : « A translation approach to portable ontology specification, Knowledge Acquisition » pp. 7, 1993.

[Gujarathi, 2011] G. P. Gujarathi, Y.-S. Ma : « Parametric CAD/CAE integration using a common data model » Journal of Manufacturing Systems, Volume 32, Issue 3, Pages 118-132, August 2011.

---------- H ----------

[Hadj-Hamou, 2002] Kh. Hadj-Hamou : « Contribution à la conception de produits à forte diversité et de leur chaîne logistique : une approche par contraintes » Thèse de doctorat de l'Institut National Polytechnique de Toulouse, Soutenue le 10 Décembre 2002.

[Hammadi, 2012] M. Hammadi : « Contribution à l'intégration de la modélisation et la simulation multi-physique pour la conception des systèmes mécatroniques » Thèse de doctorat de Centrale Paris, Soutenue le 12 Janvier 2012.

[Hammouda, 2013] N. Hammouda, G. Habchi, C. Barthod, O. Duverger : « Mise en œuvre d'une méthodologie d'évaluation de la fiabilité pour les systèmes mécatroniques » 21ème Congrès Français de Mécanique, Bordeaux, France. pp.1-6, Août 2013.

[Hamon, 2005] J.-C. Hamon : « Méthodes et outils de la conception amont pour les systèmes et les micro-systèmes » Thèse de doctorat, Institut National Polytechnique de Toulouse, Soutenue le 1er Février 2005.

[Handbook] R. H. Bishop : « The mechatronics handbook » Technology and Engineering, 2002.

[Harmel, 2007] G. Harmel : « Vers une conception conjointe des architectures du produit et de l'organisation du projet dans le cadre de l'Ingénierie Système » Thèse de doctorat de l'Université de Franche-Comte, Soutenue le 5 Juillet 2007.

[Heesom, 2004] D. Heesom, L. Mahdjoubi : « Trends of 4D CAD applications for construction planning » Journal of Construction Management and Economics, Volume 22, Issue 2, Pages171-182, February 2004.

[Hoffmann, 2001] C. M. Hoffmann, K.-J. Kim : « Towards valid parametric CAD models » Computer-Aided Design, Volume 33, Issue 1, Pages 81-90, January 2001.

[Hoffmann, 2005] C. M. Hoffmann : « Constraint-based CAD » J. Comput. Inf. Sci. Eng., volume 5, Issue 3, Pages 182-197, 2005.

[Hong, 2009] E.-P. Hong, G.-J. Park : « Decomposition Process Engineering System Using Axiomatic Design and Design Structure Matrix » ICAD 2009, Campus de Caparica, March 25-27, 2009.

[Huang, 2011] Y. Huang : « Elaboration de Document de Maintenance Personnalisé Pour les Produits Mécaniques Complexes » Thèse de doctorat, Université de Strasbourg, Soutenue le 21 Janvier 2011.

---------- I ----------

[IEC 60050-191] IEC 60050-191 : « International Electrotechnical Vocabulary – Chapter 191 : Dependability and quality of service » International Standard, International Electrotechnical Commission, IEC, December 1990.

[IGES, 1980] IGES 1980 : « Initial Graphics Exchange Specification » ANSI Y 14.26M, ANSI – American National Standard Institute, USA, 1980.

[INCOSE, 2004] INCOSE : « Systems engineering handbook » International Council on Systems Engineering (INCOSE) Working Group, 2004.

[Ishikawa, 2000] H. Ishikawa, H. Yuki, S. Miyazaki : « Design Information Modeling in 3D CAD System for Concurrent Engineering and Its Application to Evaluation of Assemblability/Disassemblability in Conceptual Design » Concurrent Engineering : Research and Applications, Volume 8, Numero 1, Pages 24-31, March 2000.

[ISO, 13584-20:1998] ISO 13584-20 : « Industrial automation systems and integration – Parts library – Part 20: Logical resource: Logical model of expressions » 1999.

[ISO, 13584-24:1998] ISO DIS 13584-24 : « Industrial automation systems and integration – Parts library – Part 24: Logical resource: Logical model of supplier library » 1999.

[ISO, 13584-42:1998] ISO 13584-42 : « Industrial Automation Systems and Integration – Parts Library – Part 42: Description methodology: Methodology for Structuring Parts Families » ISO, Genève, 1998.

[ISO, 10303-1:1994] ISO 10303-1. Part 1: « Overview and fundamental principles » 1994.

[ISO, 10303-11:1994] ISO 10303-11. Part 11: « EXPRESS Language Reference Manual » 1994.

[Issa, 2013] H. Issa, E. Ostrosi, M. Lenczner, R. Habib : « Fuzzy Functional Modelling in CAD Systems » In concurrent Engineering Approaches for Sustainable Product Development in a multi-diciplinary Environment, Springer London, Pages 595-608, 2013.

---------- J ----------

[Jardin, 2010] A. Jardin : « Contribution à une méthodologie de dimensionnement des systèmes mécatroniques : analyse structurelle et couplage à l'optimisation dynamique » Thèse de doctorat de l'INSA de Lyon, Soutenue le 15 Janvier 2010.

[Jeantet, 1998] A. Jeantet : « Les objets intermédiaires dans la conception. Eléments pour une sociologie des processus de conception » Sociologie du travail 3, 1998.

[Julie, 2009] J. Latrémouille-Viau : « Traitement en fin de vie des avions et valorisation de l'aluminium » Mémoire présenté en vue de l'obtention du diplôme de Maître ÈS Sciences Appliquées, Université de Montréal, (Génie Industriel) Décembre 2009.

[JSDAI] JSDAI™ : « Le JSDAI » www.jsdai.net.

---------- K ----------

[Khalfoui, 2003] S. Khalfoui : « Méthode de recherche des scénarios redoutés pour l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques automobiles » Thèse de doctorat de l'Institut National Polytechnique de Toulouse, Soutenue le 26 Septembre 2003.

[Kim, 2008] J. Kim, M. J. Pratt, R. G. Iyer, R. D. Sriram : « Standardized data exchange of CAD models with design intent » Computer-Aided Design, Volume 40, Issue 7, Pades 760-777, July 2008.

[Kota, 1991] S. Kota, A. C. Ward : « Functions, structures and constraints, in Conceptual Design » ed. A.C. Ward, University of Michigan, 1991.

[Kuate, 2006] G. Kuate : « Analyse d'activités de conception : Contribution à la traçabilité des intentions de conception dans les modèles CAO » Thèse de doctorat de l'Université de Technologie de Belfort-Montbéliard, Soutenue le 15 Décembre 2006.

---------- L ----------

[La Rocca, 2012] G. La Rocca : « Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design » Advanced Engineering Informatics, Volume 26, Issue 2, Pages 159-179, April 2012.

[Labatut, 2007] P. Labatut : « Environnements de développement intégrés : Introduction aux EDI, la plateforme Eclipse » Centre d'enseignement et de recherche en technologies de l'information et

systèmes, Ecole des ponts, http://www.normalesup.org/~labatut/ED6/cours-1.pdf, 2006-2007.

[Larousse, 2005] Collectif Larousse : « Le petit Larousse illustré 2005 » Larousse, 100ème édition, paru le 13 Juillet 2004.

[Lasalle, 2012] J. Lasalle : « Génération automatique de tests à partir de modèles SysML pour la validation fonctionnelle de systèmes embarqués » Thèse de Doctorat de l'Université de Franche-Comté, Spécialité Informatique, Soutenue le 29 juin 2012.

[Laug, 2005] P. Laug : « Topologie et maillage des surfaces paramétrées à partir d'une modélisation B-Rep » 17ème Congrès Français de Mécanique, Troyes, Septembre 2005.

[Lee, 2010] H. Lee, J. Kim, A. Banerjee : « Collaborative intelligent CAD framework incorporating design history tracking algorithm » Computer-Aided Design, Volume 42, Issue 12, Pages 1125-1142, December 2010.

[Levitt, 2000] D. Levitt : « Introduction to Structured Analysis and Design Technique » CS 2000: System Analysis and Design, 2000.

[Li, 2005] W. D. Li, W. F. Fu, J. Y. H. Fuh, Y. S. Wong : « Collaborative computer-aided design: research and development status » Computer-Aided Design, Volume 37, Issue 9, Pages 931-940, August 2005.

[Li, 2010] M. Li, F. C. Langbein, R. R. Martin : « Detecting design intent in approximate CAD models using symmetry » Computer-Aided Design, Volume 42, Issue 3, Pages 183-201, March 2010.

[Lo, 2013] M. Lo : « Contribution à l'évaluation d'architectures en Ingénierie Système : application en conception de systèmes mécatroniques » Thèse de doctorat de l'Université Montpellier 2, Soutenue le 19 Novembre 2013.

---------- M ----------

[Marchenko, 2011] M. Marchenko, B. A. Behrens, G. Wrobel, R. Scheffler, M. Plebow : « A New Method of Visualization and Documentation of Parametric Information of 3D CAD Models » Computer-Aided Design & Applications, Volume 8, Issue 3, Pages 435-448, 2011.

[Mathis, 2010] P. Mathis, S. E. B. Thierry : « A formalization of geometric constraint systems and their decomposition » Formal Aspects of Computing, Volume 22, Issue 2, Pages 129-151, 2010.

[Maurice, 2005] R. Maurice : « Contribution à la Méthodologie de Conception Système : Application à la Réalisation d'un Micro-système Multi-capteurs Communicant pour le Génie Civil » Thèse de doctorat, Institut National Polytechnique de Toulouse, Soutenue le 15 Décembre 2005.

[Mekhilef, 1998] M. Mekhilef, B. Yannou : « Conception intégrée assistée par ordinateur » BM 5

006, Techniques de l'Ingénieur, Traité Génie Mécanique, 1998.

[Menedero, 2000] J. Menedero : « Parametric design: a review and some experiences » Automation in Construction, Volume 9, Issue 4 , Pages 369-377, 2000.

[Menye, 2009] J.-B. Menye : « Validation de la maintenabilité et de la disponibilité en conception d'un système multi-composants » Thèse de doctorat en cotutelle, Université Laval - Québec, Université de Strasbourg – France, 2009.

[Menand, 2002] S. Menand : « Modélisation pour la réutilisation du processus de conception multi acteurs de produits industriels : Application à la conception fonctionnelle des systèmes de direction automobile » Thèse de doctorat de l'INPG, Soutenue le 10 Janvier 2002.

[Mihalache, 2007] A. G. Mihalache : « Modélisation et évaluation de la fiabilité des systèmes mécatroniques : Application sur système embarqué » Thèse de doctorat de l'université d'Angers, Soutenue le 27 Décembre 2007.

[Mimoune, 2004] M. El Hadj Mimoune : « Contribution à la modélisation explicite et à la représentation des données de composants industriels : application au modèle PLib » Thèse de doctorat en Informatique et Application de l'Université de Poitiers, soutenue le 12 Juillet 2004.

[Mohammed, 2008] J. Mohammed, J. May, A. Alavi : « Application of Computer Aided Design (CAD) In Knowledge Based Engineering » Proceedings of The 2008 IAJC-IJME International Conference, 2008.

[Moinet, 2014] M. Moinet, G. Mandil, Ph. Serre : « Defining tools to address over-constrained geometric problems in Computer Aided Design » Computer-Aided Design, Volume 48, Pages 42-52, March 2014.

[Moore, 2001] K. E. Moore, A. Güngör, S. M. Gupta : « Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships » European Journal of Operational Research, Volume 135, Pages 428-449, 2001.

[Motte, 2008] D. Motte : « A Review Of The Fundamentals Of Systematic Engineering Design Process Models » International Design Conference - Design 2008, Dubrovnik - Croatia, May 19-22, 2008.

[Mucalet, 2003] R. Mucalet, M. Daniel : « Conception, modélisation géométrique et contraintes en CAO : Une synthèse » Rapport de Recherche LSIS-2003-005, 2003.

[Mucalet, 2004] R. Mucalet, M. Daniel : « Conception, modélisation géométrique et contraintes en CAO : Une synthèse » Revue d'Intelligence Artificielle, Volume 18, Issue 5-6, Pages 619-645, 2004.

[Myung, 2001] S. Myung, S. Han : « Knowledge-based parametric design of mechanical products based on configuration design method » Expert Systems with Applications, Volume 21, Issue 2, Pages 99-107, August 2001.

---------- N ----------

[Nahm, 2006] Y.-E. Nahm, H. Ishikawa : « A new 3D-CAD system for set-based parametric design » Int J Adv Manuf Technol, Volume 29, Issue 1-2, Pages 137-150, 2006.

[NF E 01-010] Norme française NF E 01-010 : « Mécatronique - vocabulaire » ICS : 01.040.21 ; 01.040.31 ; 21.020, ISSN 0335-3931, Novembre 2008.

[NF EN 292-1] Norme française NF EN 292-1 : « Sécurité des machines : Notions fondamentales, principes généraux et conception. Partie 1 : Terminologies de base, méthodologie » Décembre 1991.

[NF X 50-150] Norme française NF X50-150 : « Analyse de la valeur - Analyse fonctionnelle. Vocabulaire » Norme AFNOR, Août 1990.

[NF X 60-010] Norme française NF X 60-010 : « Définition de maintenance Prédictive » 1991.

[NF X 60-500] Norme française NF X 60-500 : « Terminologie relative à la fiabilité – maintenabilité – disponibilité » Norme expérimentale X 60-500, 1988.

[Noy, 2000] N. F. Noy, D. L. McGuinness : **«** Développement d'une ontologie 101 : Guide pour la création de votre première ontologie » Université de Standford, 26 p. http://ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html, 2000.

[NS X 400-630] Schneider Electric : « Valorisation des Compact NSX400-630 en fin de vie » http://www2.schneiderelectric.it/CompactNSX/Catalogo_a_sezioni_Compact_NSX/3_Concezione/3_Rispetto_ambiente/Dossier_fine_vita_400-630.pdf.

---------- O ----------

[O'Brien, 2000] W. O'Brien : « Tawards 5D CAD – Dynamic Cost and Ressource Planning for Specialist Contractors » Construction Congress VI, Pages 1023-1028, 2000.

[OMG UML, 2005a] « Unified Modeling Language: Superstructure » Version 2.0, August 2005.

[OMG UML, 2005b] « Unified Modeling Language: Infratructure » Version 2.0, March 2006.

[OMG SysML, 2006] « System modeling language (SysML) specification » Version 1.0, Draft. OMG document ad/2006-03-01. *SysML*. [Online], http://www.sysml.org/ http://www.omgsysml.org/, 2006.

---------- P ----------

[Pahl, 1996] G. Pahl, W. Beitz : « Engineering Design: a Systematic Approach » Springer-Verlag, London, 2nd edition, 1996.

[Pahl, 2006] G. Pahl, W. Beitz : « Engineering Design: A Systematic Approach » Springer-Verlag (2nd edition), Berlin, Heidelberg, New York, Tokyo, 2006.

[Pahl, 2007] G. Pahl, W. Beitz, J. Feldhusen, K.-H. Grote : « Engineering design: A systematic approach » 3rd edition Berlin: Springer-Verlag, 2007.

[Pärttö, 2012] M. Pärttö, P. Saariluoma: « Explaining Failures in Innovative Throught Processes in Engineering Design » Procedia – Social and Behavioral Sciences, Vol. 41, pp 442-449, 2012.

[Penoyer, 2000] J. A. Penoyer, G. Burnett, D. J. Fawcett, S.-Y. Liou : « Knowledge based product life cycle systems: principles of integration of KBE and C3P » Computer-Aided Design, Volume 32, Issue 5-6, Pages 311-320, May 2000.

[Pentcheva, 2010] M. Pentcheva : « Conversion CSG-BRep de scènes définies par des quadriques » Thèse de doctorat de l'Université Nancy 2, Soutenue le 30 Septembre 2010.

[Pierra, 1989] G. Pierra : « Bibliothèque neutre de Composants Standard pour la CAO : le projet européen CAD/LIB » Revue internationale de CFAO et d'Infographie, vol. 4, n°2 , pp. 35-53, 1989.

[Pierra, 1994] G. Pierra : « Modeling classes of pre-existing components in a CIM perspective: the ISO13584/ENV 400014 Approach » Revue internationale de CFAO et d'Infographie, vol. 9, n°3, 1994 , pp. 435-454, 1994.

[Pierra, 1997] G. Pierra : « Intelligent Electronic Component Catalogues for Engineering and Manufacturing » Proc. of the Internat. Symposium on Glogal Engineering Networking, GEN'97, pp. 331-352, Antwerp, Belgium, April 2-24, ISBN 3-931466-20-5, 1997.

[Pierra, 1999] G. Perra : « Représentation et Echange de données techniques » Actes des Entretiens POLYMECA, Valenciennes, ENSIMEV, Pages 35-54, 24 Novembre 1999.

[Pierra, 2002] G. Pierra : « Un modèle formel d'ontologie pour l'ingénierie, le commerce électronique et le Web sémantique : Le modèle de dictionnaire sémantique PLIB » Journées Scientifiques WEB SEMANTIQUE, Paris, 10-11 Octobre 2002.

[Plantec] A. Plantec : « Outils et méthodes de la norme STEP (Standard ISO 10303) » Lab-STICC/CNRS UMR 6285, Université de Bretagne Occidentale, http://dossen.univ-brest.fr/apl/data/_uploaded/file/Echange de donnees/STEP/step.pdf.

[Pollet] Y. Pollet : « Modélisation des systèmes »

[Pu, 2009] S. Pu, G. Vosselman : « Knowledge based reconstruction of building models from terrestrial laser scanning data » ISPRS Journal of Photogrammetry and Remote Sensing, Volume 64, Issue 6, Pages 575-584, November 2009.

---------- R ----------

[Rahman, 2007] R. Rahman, U. Pulm, R. Stetter : « Systematic Mechatronic Design of a Piezo-Electric Brack » Proceedings of the 16th International Conference of Engineering Design (ICED'2007), 2007, Design Society, Paris, 28-31 August 2007.

[Red, 2011] E. Red, G. Jensen, D. French, P. Weerakoon : « Multi-User Architectures for Computer-Aided Engineering Collaboration » Proceedings of the 17th International Conference on Concurrent Enterprising (ICE 2011), Aachen, Germany, 2011.

[Red, 2013] E. Red, D. French, G. Jensen, Sh.S. Walker, P. Madsen : « Emerging Design Methods and Tools in Collaborative Product Development » Journal of Computing and information Science in Engineering, Volume 13, Issue 3, Page 031001, 2013.

[Roller, 1991] D. Roller : « An approach to computer-aided parametric design » Computer-Aided Design, Volume 23, No. 5, Pages 385-391, June 1991.

[Roques, 2013] P. Roques : « Modélisation de systèmes complexes avec SysML » Eyrolles 2013, ISBN : 978-2-212-13641-8, 2013.

[Rouquier, 2005] J. B. Rouquier : « Autour de la robustesse des systèmes complexes: le cas des automates cellulaires coalescents » Mémoire de DEA, Soutenu en Août 2005.

[Russéry, 2007] J. P. Russéry, Y. Cransac : « Analyse fonctionnelle du produit : Fonctions techniques – Solutions » Projet Clé USB, 2007.

---------- S ----------

[Sardet] E. Sardet : « Une approche pour la représentation des catalogues de composants industriels : Le modèle PLIB »

[Sharon, 2009] A. Sharon, D. Dori, O. de Weck : « Model-Based Design Structure Matrix: Deriving a DSM from an Object-Process Model » Second International Symposium on Engineering Systems MIT, Cambridge, Massachusetts, June 15-17, 2009.

[Serrafero, 2005] P. Serrafero, S. Gomes, D. Bonnivard, L. Jézéquel : « De la mémoire projet à la compétence métier: Vers la synthèse de connaissances métier en ingénierie robuste produits/process » 1-KKE-DS2.6-PS0-0508-F, Octobre 2005.

[Serrafero, 2012] P. Serrafero : « Evolution sémantique de l'ingénierie assistée par ordinateur » Décembre 2012.

[SET, 1989] SET Z 68-300 : « Industrial Automation – External Representation of Product Definition Data – Data Exchange and Transfert Standard Specification » AFNOR, France, 1989.

[Simon, 1990] H. Simon : « Sur la complexité des systèmes complexes » Revue internationale de systémique, Vol. 4, N° 2, Pages 125-145, 1990.

[Soler, 2000] L. Soler : « Introduction à l'épistémologie » Ellipses, 2000.

[Song, 2014] J. Song, S. Cho, S.Y. Baek, K. Lee, H. Bang : « GaFinC: Gaze and Finger Control interface for 3D model manipulation in CAD application » Computer-Aided Design, Volume 46, Pages 239-245, January 2014.

[Srinivasan, 2008] V. Srinivasan : « Standardizing the specification, verification, and exchange of product geometry: Research, status and trends » Computer-Aided Design, Volume 40, Issue 7, Pages 738-749, July 2008.

[STIDD] 1er STIDD/ITEC/Cours : « Phases d'un projet industriel » Cours.

---------- T ----------

[Thierry, 2007] S. E. B. Thierry, P. Mathis, P. Schreck : « Towards an homogeneous handling of under-constrained and well-constrained systems of geometric constraints » Procedings of the 2007 ACM symposium on Applied computing, New York, NY, USA, Pages 773-777, 2007.

[Thierry, 2011] S. E. B. Thierry, P. Schreck, D. Michelucci, Ch. Funfzig, J.-D. Génevaux : « Extensions of the witness method to characterize under-, over- and well-constrained geometric constraint systems » Computer-Aided Design, Volume 43, Issue 10, Pages 1234-1249, October 2011.

[Trabelsi, 2014] H. Trabelsi : « Contribution à la prise en compte d'exigences dynamiques en conception préliminaire de systèmes complexes » Thèse de doctorat de Central Paris et de l'Ecole nationale d'ingénieur de Sfax, Soutenue le 16 Janvier 2014.

[Turki, 2008] S. Turki : « Ingénierie système guidée par les modèles: Application du standard IEEE 15288, de l'architecture MDA et du langage SysML à la conception des systèmes mécatroniques » Thèse de doctorat de l'Université du sud Toulon-Var, Soutenue le 02 Octobre 2008.

---------- V ----------

[Vargas, 2006] R. H. D. Vargas : « Mise au Point de la Plate-Forme HiLeS » Master Recherche SMIS-EEAS, Systèmes automatiques Informatiques et décisionnels, 2006.

[Vauquier, 2007] D. Vauquier : « La modélisation sémantique : un aperçu des procédés de modélisation dans la méthode publique Praxeme » La Lettre d'ADELI n° 69 – Automne 2007.

[VDA, 1986] VDA-FS : « Vereinung Deutsche Autombilindustie Flächen Schnittstelle » DIN 66301, DIN-Deutsches Instiut für Normung, Germany, 1986.

[VDI 2206] Verein Deutscher Ingenieure 2206 : « Design methodology for mechatronics systems »

Norme VDI 2206, Verein Deutscher Ingenieure, Berlin, Beuth Verlag GmbH, juin 2004.

[Verries, 2010] Jean Verries : « Approche pour la conception de systèmes aéronautiques innovantes en vue d'optimiser l'architecture : application au système portes passagers » Thèse de doctorat de l'université de Toulouse, Soutenue le 21 Janvier 2010.

[Vernat, 2004] Y. Vernat : « Formalisation et qualification de modèles par contraintes en conception préliminaire » Thèse de doctorat de l'École Nationale Supérieure d'Arts et Métiers Centre de Bordeaux, Soutenue le 18 Novembre 2004.

[Villard, 2007] M. Villard : « Les "Standards de facto", étude de cas pour l'informatique » Colloque CNIDECA du 5 avril 2007.

---------- W ----------

[Whitten, 2004] J. L. Whitten, L. D. Bentley, K. C. Dittman : « Systems Analysis and Design Methods » Irwin/McGraw-Hill. ISBN: 025619906X, 2004.

[Wilhelms, 2005] S. Wilhelms : « Function- and Constraint-based Conceptual Design using Easy Exchangeable, Reusable Principle Solution Elements » AI EDAM, Vol. 19, No. 3, pp. 201-219, 2005.

[Wu, 2007] X. J. Wu, W. J. Liu, M. Y. Wang : « Modeling Heterogeneous Objects in CAD » Computer-Aided Design & Applications, Volume 4, Issue 6, Pages 731-740, 2007.

---------- X ----------

[XP E 01-013] Norme française XP E 01-013 : « Mécatronique – Cycle de vie et conception des produits » Octobre 2009.

---------- Z ----------

[Zha, 2002] X. F. Zha, H, Du : « A PDES/STEP-based model and system for concurrent integrated design and assembly planning » Computer-Aided Design, Volume 34, Issue 14, Pages 1087-1110, December 2002.

[Ziemniak, 2009] P. Ziemniak, M. Stania, R. Stetter : « Mechatronics Engineering On The Example Of An Innovative Production Vehicle » International Conference On Engineering Design, ICED'09, 24 - 27 August 2009, Stanford University, Stanford, CA, USA, 2009.

[Zwingmann, 2005] X. Zwingmann : « Modèle d'évaluation de la fiabilité et de la maintenabilité au stade de la conception » Thèse de doctorat en cotutelle, Université Laval – Québec, Université Louis-Pasteur – Strasbourg – France, 2005.

Bibliographie personnelle

[ICIDM, 2014] S. Diagne, A. Coulibaly, F. De Bertrand De Beuvron: « Towards a Conceptual Semantic Design for Mechatronic Product's Family Development » Proceedings of the International Conference on Innovative Design and Manufacturing (ICIDM 2014), Print ISBN 978-1-4799-6269-3, DOI 10.1109/IDAM.2014.6912677, Pages 94-99, Montreal, Quebec, Canada, August 13-15, 2014.

[DSMConf, 2014] S. Diagne, A. Coulibaly, M. Sene, F. De Bertrand De Beuvron: « Complex Mechatronic Product Modeling using a Multi-Solution, Multi-Instance eXtended Conceptual Design Semantic Matrix » Proceedings of the 16th International DSM Conference (DSM Conf 2014), Book Risk and change management in complex systems, Print ISBN 978-1-56990-491-6, DOI 10.3139/9781569904923.009, Pages 85-94, Paris, France, July 2-4, 2014.

[CIRP LCE, 2015] S. Diagne, A. Coulibaly, Mb. Sene: « Improvement of Optimal Disassembly Sequences of Complex Systems family Using Petri Nets » Proceedings of the CIRP Life Cycle Engineering (CIRP LCE 2015), Sydney, Australia, April 7-9, 2015.

[SysCon, 2015] S. Diagne, A. Coulibaly, F. De Bertrand De Beuvron: « Mechatronics Product's Families Functional Modeling using SysML and X-CDSM For Reliability Assessment » Proceedings of the 9th IEEE International Systems Conference (SysCon 2015), Vancouver, British Columbia, Canada, April 13-16, 2015.

[IAMOT, 2015] S. Diagne, A. Coulibaly, F. De Bertrand De Beuvron: « Complex Product's Behavioral Performance Engineering at Design Stage » Proceedings of the 24th International Association for Management of Technology (IAMOT 2015), Cap Town, South Africa, June 8-11, 2015.

[ComInd, 2015] S. Diagne, A. Coulibaly, F. De Bertrand De Beuvron: « Complex Product Modeling based on a Multi-Solution eXtended Conceptual Design Semantic Matrix for Behavioral Performance Assessment » Elsevier Computers In Industry Journal, DOI 10.1016/j.compind.2015.06.003, 2015.

[Diagne, 2007] S. Diagne: « Equilibrage de charges dans les systèmes distribués, modèle de performance dans le cas des systèmes transactionnels » Mémoire de DEA d'Informatique, Université Cheikh Anta DIOP de Dakar, Soutenu le 25 Août 2007.

Webographie

[web 1] http://fr.wikipedia.org/wiki/Systémique

[web 2] http://fr.wikipedia.org/wiki/Complexité

[web 3] Annick Lesne : http://www.lptmc.jussieu.fr/user/lesne/Lesne-ResumeCommisco.pdf.

[web 4] http://fr.wikipedia.org/wiki/Système_complexe.

[web 5] http://www.techniques-ingenieur.fr/fiche-pratique/genie-industriel-th6/deployer-l-innovation-dt30/ingenierie-des-systemes-complexes-0269/

[web 6] http://sfa.univ-poitiers.fr/automation/spip.php?article103

[web 7] http://fr.wikipedia.org/wiki/Mécatronique.

[web 8] http://www.iutren.univ-rennes1.fr/formations/LP/Mecatronique/Cestquoi/

[web 9] http://www.appropedia.org/Systematic_design.

[web 10] http://en.wikipedia.org/wiki/Design_structure_matrix

[web 11] http://fr.wikipedia.org/wiki/Ontologie_(informatique).

[web 12] http://www.journaldunet.com/developpeur/tutoriel/theo/070403-ontologie.shtml.

[web 13] Recyclage : http://fr.wikipedia.org/wiki/Recyclage

[web 14]http://www.plm.automation.siemens.com/fr_fr/products/nx/for-design/mechatronics-design

[web 15] http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=13949997.

[web 16] http://www.cadvision.fr/solidworks-logiciel-circuitworks-mecatronique-3d.htm.

[web 17] http://rmdiscala.developpez.com/cours/LesChapitres.html/Cours2/Chap2.1.htm

[web 18] http://fr.wikipedia.org/wiki/Environnement_de_développement

Annexes

```
/********************************** Fonction de recherche de position ***********************************/
   public static int Position(String c, String [] tc) {
      int pos1 ;
      j = 0 ;
      while ((tc[j].compareTo(c) != 0) && (j < tc.length)) {
         j = j + 1 ;
      }
      if (j > tc.length) {
         pos1 = -1 ;
      }
      else {
         pos1 = j ;
      }
      return pos1 ;
   }

/****************************** Fonction de comptage de nombre de liaisons ******************************/
   public static int [] nb_liaisons(int [][] tp1) {
      int [] nb = new int[tp1.length] ;
      int i ;
      for (i = 0; i < tp1.length; i++) {
         for (j = 0; j < tp1.length; j++) {
            if (i != j) {
               if (tp1[i][j] != 0) {
                  nb[i] = nb[i] + 1 ;
                  nb[j] = nb[j] + 1 ;
               }
            }
         }
      }
      return nb ;
   }

/***************************** Fonction de recherche de chemin de démontage *****************************/
   public static int [] voie(int p, int [][] tp) {
      int x, y, i, j, k ;
      int [] ch ; //= new int[N + 1] ;
      int [] chemin = new int[tp.length] ;
      int [] sor = new int[tp.length + 1] ;
      k = 0 ;
      if (p >= 0) {
         for (i = 0; i < tp.length; i++) {
            if ((tp[i][p] > 0) && (i != p)) {
               if (k == 0) {
                  chemin[k] = i ;
                  k = k + 1 ;
               }
               else {
                  x = 0 ;
                  for (y = 0; y <= k – 1; y++) {
                     if (chemin[y] == i) {
                        x = 1 ;
```

```
                    }
                }
                if (x == 0) {
                    chemin[k] = i ;
                    k = k + 1 ;
                }
            }
            for (j = 0; j < tp.length; j++) {
                if ((tp[j][i] > 0) && (j != i)) {
                    y = 0 ;
                    while ((j != chemin[y]) && (y < k)) {
                        y = y + 1 ;
                    }
                    if (y == k) {
                        x = chemin[k - 1] ;
                        chemin[k - 1] = j ;
                        chemin[k] = x ;
                        ch = voie(j, tp) ;
                        if (ch[0] > 0) {
                            for (x = 1; x <= ch[0]; x++) {
                                y = 0 ;
                                while ((ch[x] != chemin[y]) && (y < k)) {
                                    y = y + 1 ;
                                }
                                if (y == k) {
                                    chemin[k + 1] = chemin[k] ;
                                    chemin[k] = chemin[k - 1] ;
                                    chemin[k - 1] = ch[x] ;
                                    k = k + 1 ;
                                }
                            }
                        }
                        k = k + 1 ;
                    }
                }
            }
        }
    }
    chemin[k] = p ;
    sor[0] = k ;
    for (x = 1; x <= k + 1; x++)
        sor[x] = chemin[x - 1] ;
    return sor ;
}

/****************** Fonction de calcul du temps de remplacement de composants défaillants ******************/
public static int trmp(int cp, int [] ch, int [][] mat) {
    int temprmp, x, i, y, q, temp;
    temprmp = 0 ;
    tempo = 0 ;
    if (cp >= 0 && ch[0] == 0) {
        for (q = 0; q < mat.length; q++) {
            if (q != cp) {
                temprmp = temprmp + mat[cp][q] ;
            }
```

```
        }
        for (y = 0; y < mat.length; y++) {
           if (y != cp) {
              temprmp = temprmp + mat[y][cp] ;
           }
        }
        temprmp = 2 * temprmp ;
     }
     else if (cp >= 0 && ch[0] > 0) {
        for (y = 1; y <= ch[0] + 1; y++) {
              for (q = 0; q < mat.length; q++) {
                 if (q != ch[y]) {
                    x = 0 ;
                    for (i = 1 ; i <= ch[0] + 1; i++) {
                      if (q == ch[i]) {
                         x = x + 1 ;
                      }
                    }
                    if (x == 1) {
                      tempo = tempo + mat[ch[y]][q] ;
                    }
                    else {
                      temprmp = temprmp + mat[ch[y]][q] ;
                    }
                 }
              }
              for (j = 0; j < mat».length; j++) {
                 if (j != ch[y]) {
                    x = 0 ;
                    for (i = 1; i <= ch[0] + 1; i++) {
                      if (j == ch[i]) {
                         x = x + 1 ;
                      }
                     }
                     if (x == 1) {
                      temp = tempo + mat[j][ch[y]] ;
                    }
                    else {
                      temprmp = temprmp + mat[j][ch[y]] ;
                    }
                 }
              }
        }
        temprmp = 2 * temprmp + tempo ;
     }
     return temprmp ;
  }
```