Interopérabilité de modèles dans le cycle de conception des systèmes électromagnétiques via des supports complémentaires: Langage VHDL-AMS et composants logiciels ICAr

Abir Rezgui

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Génie Electrique**

Arrêté ministériel : 7 août 2006

Présentée par

Abir REZGUI

Thèse dirigée par **M. Laurent GERBAUD**
et codirigée par **M. Benoît DELINCHANT**

préparée au sein du **Laboratoire de Génie Électrique de Grenoble (G2eLab)**
dans **l'École Doctorale « Électronique, Électrotechnique, Automatique et Traitement du Signal »**

Interopérabilité de modèles dans le cycle de conception des systèmes électromagnétiques via des supports complémentaires:

Langage VHDL-AMS et composants logiciels ICAr

Thèse soutenue publiquement le **« 25/10/2012 »**,
devant le jury composé de :

M. Hervé MOREL
Directeur de recherche au CNRS Lyon (Président)
M. Yannick HERVE
Maître de conférence à l'université de Strasbourg (Rapporteur)
M. Hubert PIQUET
Professeur à l'université de Toulouse (Rapporteur)
M. Vincent LECONTE
Ingénieur à CEDRAT Grenoble (Examinateur)
M. Laurent GERBAUD
Professeur à l'Institut National Polytechnique Grenoble (Directeur de thèse)
M. Benoît DELINCHANT
Maître de conférence à l'Université Joseph Fourier Grenoble (Co-Directeur de thèse)
M. Sébastien FURIC
Ingénieur à LMS ImagineLyon (invité)

Remerciements

Le moment est venu de conclure les trois années exceptionnelles passées au sein du Laboratoire de Génie Électrique de Grenoble, durant lesquelles j'ai vécu toutes sortes de situations ; des plus inattendues aux plus gratifiantes. Je profite de cette occasion pour exprimer ma reconnaissance à l'égard de toutes les personnes que j'ai pu côtoyer et sans lesquelles cette aventure n'aurait été ni possible, ni formidable.

Je souhaiterais tout d'abord exprimer ma profonde gratitude envers mes deux directeurs de thèse : M. Laurent GERBAUD et M. Benoit DELINCHANT ; deux personnes complémentaires et qui ont énormément de qualités humaines. Je vous remercie pour votre confiance dans l'attribution de ce sujet, peu ordinaire au labo, pour votre encadrement de grande qualité scientifique, vos conseils constructifs, votre écoute et surtout votre patience. Ce fut un plaisir de travailler avec vous, malgré les moments difficiles. Je vous suis reconnaissante pour tout ce que vous m'avez appris et pour la confiance que vous m'avez accordée. Je n'oublierai jamais ces trois années de partage enrichissantes sur tous les plans ; personnel et professionnel.

Mes sentiments respectueux et reconnaissants s'adressent tout particulièrement aux membres du jury pour l'intérêt qu'ils ont porté à ce travail et de m'avoir offert une après soutenance riche en discussion et en échange technique :

- M. Hervé MOREL de m'avoir fait le très grand honneur de présider mon jury.
- M. Hubert PIQUET de avoir accepté d'être rapporteur et de s'être plongé dans ce sujet en émettant des remarques constructives.
- M. Yannick HERVE d'avoir été mon rapporteur. Je tiens plus particulièrement à te remercier pour ton aide et ta participation dans l'avancée de mes travaux. Tes conseils judicieux et ta grande pédagogie se sont avérés essentielles. Je n'oublierai jamais le séjour passé à Tunis lors de la conférence SM2ACD, ce fut pour moi l'occasion de faire la connaissance d'une personne admirable, de par ses qualités aussi bien scientifiques qu'humaines.
- M. Vincent LECONTE d'avoir examiné mon rapport, d'avoir souligné l'intérêt industriel de mon travail ainsi que pour toutes les réunions, discussions et échanges que nous avons pu partager autour du projet MoCoSyMec.
- M. Sébastien FURIC d'avoir participé à mon jury et pour vos commentaires pertinents sur mon travail.

Je tiens à remercier les personnes qui se sont investis à mes côtés sur mes divers travaux et qui m'ont apporté leur aide ; plus particulièrement :

- Bertrand DuPeloux avec qui j'ai eu beaucoup de plaisir à collaborer, pour ses idées très enrichissantes et son incroyable gentillesse : Merci d'avoir assisté à ma soutenance.
- Petre, Franck et Sylvain avec qui j'ai passé beaucoup de bons moments et qui n'ont pas ménagé leur effort pour m'aider, m'écouter et me conseiller : C'est toujours un réel plaisir de discuter avec vous.
- Pascal et toute l'équipe technique de Cedrat : Merci à vous d'avoir débugué mes modèles et de m'avoir fourni des réponses logiques.
- Julien, Roger et Xiadong qui m'ont offert la possibilité d'encadrer différents projets d'étudiants avec eux, pour l'intérêt qu'ils ont porté à mes travaux et pour tout ce qu'ils m'ont appris sur le monde industriel.
- Lucile et toute l'équipe de Dolphin Integration pour leur réactivité et pour toutes les réponses et les développements qu'ils m'ont fournis sur SMASH.
- Mes remerciements s'adressent tout particulièrement à Pr. Jean-Louis Coulomb, pour sa grande compétence scientifique, son recul, son humilité et sa disponibilité. Je te remercie énormément pour l'aide que tu m'as apporté sur Flux et Fgot principalement.

À présent, je tiens à remercier du fond de mon cœur tous les permanents, chercheurs, techniciens, ingénieurs et personnels administratifs du G2ELlab et ceux de CNRS, les anciens comme les nouveaux pour ces trois années. Les compétences scientifiques et humaines des uns et des autres m'ont permis de travailler dans un environnement agréable.

Une mention spéciale aux personnes exceptionnelles que j'ai eu la chance de côtoyer aux G2Elab, ENSE3 et GSCOPE : Afef, Sylvie, Jean-Michel, Gérard, Fred, Albert, Raphael, Seddik, Xavier, Gilles, Stéphane, Alain, Eric E. et Eric .Z.

Mes camarades thésards occupent une place majeure dans ces remerciements car c'est en leur compagnie que ces trois années ont pu être une agréable expérience et avec lesquels de réels liens d'amitiés se sont créés.

Je commencerai par mon amie et ma voisine Asma, avec qui j'ai partagé une longue et agréable route. Sans toi, je pense que j'aurai du mal à supporter Grenoble les premiers mois.

Un grand Merci à ma copine Yasmine, pour tous les supers instants qu'on a passé et qu'on passera ensemble. Merci pour ton aide dans les moments difficiles, pour ton écoute et ton attention. Rassures toi, je ne renoncerai jamais à faire du shopping avec toi.

Je tenais à te remercier Sana pour ton amitié et tous les moments passés à discuter ou à coder. J'avais la chance de travailler avec toi sur des aspects similaires, même si nos sujets sont très différents.

Merci à ma copine Rim pour son amitié profonde et sincère, qui m'a beaucoup soutenue pendant les moments difficiles et avec qui j'ai partagé énormément de choses. Je n'oublierai jamais les moments passés dans l'entourage de ta petite famille.

Une dédicace spéciale à mon ami Didier, un garçon très mystérieux et toujours serviable comme disait Yasmine. Je ne te remercierais jamais assez pour ton soutien et pour ton aide, aussi bien moral que scientifique et surtout d'être un bon ami. Je ne sais pas ce que j'aurai pu faire sans toi et Olivier P. pendant la rédaction et la soutenance. Merci pour tous les moments qu'on a passé et qu'on passera ensemble et pour ton amitié sincère. Je ne peux pas te citer Didier sans me rappeler Bill avec ses fameuses théories et Matthieu que j'admire et que je remercie pour son aide précieuse.

Merci à mes amis et mes collègues de la salle predis : Ghaith, Hervé, Hoang-Anh et Ardavan et le petit nouveau Jonathan. Merci de m'avoir accueilli dans la salle et me compter parmi vous, même si je ne travaillais pas comme vous sur les bâtiments. Merci aussi à tous mes amis Mathieu, Christophe, Fanny, Ni, Ando, Manel, Mikael, Ramzik, Ali, Sarra, Myriam, Nathalie, Hajer, Faten qui ont créé un cadre particulièrement agréable ces années.

Une pensée particulière à mes amis en dehors du laboratoire. Je tiens à remercier mes meilleures amies à Tunis Hannoun, Meriem, Mouna, Dosdos et Zeineb. Malgré la distance qui nous sépare, elles ont su être des véritables amies. Je remercie aussi toutes les personnes que j'ai rencontrées à Paris et avec lesquelles de réels liens d'amitiés se sont créés, principalement mes chères Lobna, Amira et Amina.

Un merci tout particulier à mon meilleur ami Camille qui, malgré ses occupations professionnelles, il a su m'épauler et m'encourager pendant ces travaux. Merci pour ta patience, ta confiance et ton soutien qui m'ont donné le courage et la volonté de mener ce travail jusqu'au bout. Merci d'avoir cru en moi, même quand je ne croyais plus.

Mes derniers remerciements vont évidemment aux personnes à qui je tiens le plus, ma famille : mes parents, à ma sœur (jumelle) Taysir et à mon frère Bilel, sans oublier ma cousine Ahlem, pour leur amour, leur présence, leur confiance et leur soutien. Je leur doit tout et les mots ne suffisent pas à leur exprimer mon amour et ma gratitude. Je leur dédie cette thèse. Que Dieu les garde et qu'ils trouvent ici ma reconnaissance éternelle !

Table des matières

Liste des abréviations

ANR	Agence Nationale de Recherche
API	Application Programming Interface
ATL	ATLAS Transformation Language
AUTOSAR	AUTomotive Open System Architecture
BDF	Backward Difference Formula
BIM	Boundary Integral Method
BNF	Backus Normal Form
CADES	Component Architecture for Design of Engineering System
DAE	Differential Algebraic Equation
DAEIS	Differential Algebraic Equations Initialization Subroutine
DASSL	Differential Algebraic System Solver
DASKR	Differential Algebraic System Solver using preconditioned Krylov with Root Finding
DASPK	Differential Algebraic System Solver using preconditioned Krylov
DASRT	Differential Algebraic System Solver with Root Finding
DLL	Dynamic Link Library
FEM	Finite Element Method
Fgot	Feature - Genuine Optimization Tool
FMI	Functional Mockup Interface
ICAr	Interface Component Architecture
IDA	Implicit Differential Algebraic Solver
IDM	Ingénierie Dirigée par les Modèles
IEEE	Institute of Electrical and Electronics Engineers
IHM	Interface Homme Machine
JAR	Java Archive, développé par *Sun Microsystems*
JNI	Java Native Interface
JVM	Java Virtual Machine
ODE	Ordinary Differential Equation
OMG	Object Management Group
MacMMems	Macro Modeler for Magnetic Mems
MDA	Model Driven Architecture ou Architecture Dirigée par les Modèles
MDE	Model-Driven Engineering
MEMS	Micro ElectroMechanical System
MoCoSyMec	Modélisation et Conception de Systèmes Mécatroniques
MOF	Metadata Object Facility

MUSE	Modèle Unifié pour les Systèmes Energétique des bâtiments
PDE	Partial Derivative Equation
PIM	Platform Independant Model
PLUMES	Plateforme Logicielle Unifiée de Modélisation pour l'Efficacité énergétique du bâtiment et de ses Systèmes
PSM	Platform Specific Model
PVF	Prototypage Virtuel Fonctionnel
SML	System Modeling Language (langage développé à G2Elab)
SysML	Systems Modeling Language
UML	Unified Modeling Language
VHDL	Very high speed integrated circuits Hardware Description Language
VHDL-AMS	Very high speed integrated circuits Hardware Description Language – Analog- Mixed Signal
XMI	XMLMetadata Interchage
XML	Extensible Markup Language
XSD	XML Schema Description
XSLT	Extensible Stylesheet Language Transformations

Liste des figures

Introduction générale

Pour mener à bien la conception d'un système complexe de façon optimale, il faut savoir gérer la complexité de l'organisation technique et humaine. Il faut aussi gérer la complexité des données et des connaissances tout au long du cycle de conception et du cycle de vie du produit. Ceci apparaît fortement dans la conception de systèmes mécatroniques (lieu d'interactions entre différentes technologies). Dans de tels systèmes, différents acteurs avec leurs compétences variées abordent des problèmes multidisciplinaires, multi-physiques et multi-métiers et doivent faire des études allant du composant au système. Cela demande un changement de méthode dans le processus de conception. En effet, même si auparavant, les systèmes complexes pouvaient utiliser différentes technologies issues de différents domaines, ils relevaient plus d'une juxtaposition de disciplines plutôt que d'une réelle synergie.

L'ingénierie des systèmes mécatroniques n'est possible qu'à la condition d'en maîtriser toutes les étapes de conception aussi bien d'un point de vue technique et opérationnel que managérial. Ainsi, s'appuyer sur une méthodologie de conception reposant sur le prototypage virtuel et sur des formalismes supports, des standards industriels et des normes internationales permet de minimiser les risques et les coûts et de garantir la capitalisation et la réutilisation des savoirs. Cependant, la construction de ces systèmes doit prendre en compte des interactions dynamiques entre métiers et entre domaines et phénomènes physiques.

Dans ce contexte, le projet ANR MoCoSyMec «Modélisation et Conception de Systèmes Mécatroniques», cadre de nos travaux de thèse, vise à répondre aux besoins de nombreux domaines industriels qui sont demandeurs de plus en plus d'outils facilitant la conception de leurs systèmes. Ce domaine de la conception étant très vaste, nous focaliserons notre étude sur la simulation de systèmes dynamiques en exploitant des applicatifs électromagnétiques (micro et macro) avec des champs magnétiques conduits et non-conduits.

Dans le cadre de notre projet, nous nous appuierons sur la méthodologie de Prototypage Virtuel Fonctionnel (PVF) comme support au cycle en V de conception ; dans l'objectif de :

- la contextualiser à la problématique de la simulation dynamique de systèmes électromagnétiques ;
- fournir les outils et les méthodes adéquats (supports à la modélisation) qui guideront les concepteurs dans leurs processus.

Cette méthodologie fait apparaître différents niveaux de modélisation (comportemental, structurel, physique) à chaque niveau du cycle en V de conception, en fonction des objectifs d'étude des concepteurs : élaboration de modèle, simulation et dimensionnement.

Les objectifs et les outils utilisés pour chaque niveau de conception conditionnent la modélisation en termes de finesse des modèles, de méthodes de résolution et de passage d'une étape à l'autre du processus. Ainsi, la complexité des dispositifs nécessite de créer une synergie à l'intérieur de la diversité d'outils métiers d'aide à la conception. En effet, ces outils ont leurs propres formalismes pour faire communiquer les modèles en interne. Ils souffrent généralement d'une formalisation non normalisée et au coup par coup pour des communications avec des modèles externes. De ce fait, ils se retrouvent limités en termes d'interopérabilité des modèles et des outils.

Nos travaux de thèse visent à assister les concepteurs durant cette activité complexe, en offrant des supports pour gérer l'interopérabilité, leur permettant de mettre en œuvre les modèles et manipuler les outils au cours des processus de conception. Nous proposons des solutions d'aide à la modélisation et à l'interopérabilité des modèles et des outils pour supporter un cycle de conception complet pour les systèmes électromagnétiques. Pour cela, nous allons nous intéresser aux supports et formalismes pérennes pour décrire des modèles aux différents niveaux et pour différents outils de conception. La modélisation est appréhendée de différentes manières :

- la modélisation est numérique ou analytique
- la modélisation concerne un composant physique seul ou dans un système
- un modèle est confidentiel ou non
- un modèle est modifiable ou non
- un modèle est réutilisable ou non.

Ainsi, ce mémoire se compose de cinq chapitres, durant lesquels nous partons de l'analyse et la caractérisation des besoins émergeants de l'activité de conception des systèmes mécatroniques (chapitre 1) et principalement de la modélisation. À partir de là, nous proposons et mettons en œuvre des solutions d'aide à la conception (chapitres 2, 3 et 4) et nous validons le tout sur un exemple concret (chapitre 5).

Dans le chapitre 1, nous allons introduire notre contexte applicatif de la mécatronique, en nous focalisant du point de vue illustratif sur des actionneurs électromécaniques et sur des microsystèmes magnétiques. Nous allons analyser leur conception en vue d'expliciter les besoins pour mener à bien le PVF. Cela nous conduira à faire le point sur les méthodologies de conception, plus particulièrement de point de vue de la modélisation et simulation dynamique en allant du composant au système.

Pour supporter le PVF, nous analyserons les formalismes ouverts et fermés aptes à supporter l'interopérabilité entre outils et permettant de développer, récupérer, partager et composer des modèles. Cela nous permettra de justifier les choix que nous ferons :

- le langage VHDL-AMS pour les modèles ouverts ; par ailleurs choisi dans le cadre du projet MoCoSyMec ;
- les composants logiciels ICAr pour les modèles fermés ; principalement utilisés en dimensionnement de systèmes.

Dans le chapitre 2, nous appréhendons la problématique précédente dans un cadre plus restreint ; celui de la simulation dynamique des systèmes. Nous commencerons par analyser et caractériser les spécificités des systèmes dynamiques mixtes (continus, discrets, évènementiels). Cela nous conduira à appréhender deux aspects : la nature des modèles et les méthodes associées de résolution temporelle.

Ces travaux nous permettrons d'étendre la norme des composants logiciels ICAr afin de couvrir les différentes facettes des modèles dynamiques et faciliter leur exploitation pour différents simulateurs dynamiques. Nous implémenterons aussi des solveurs dynamiques en vue de l'utilisation des ICAr à long terme afin de répondre aux problèmes d'interopérabilité entre des outils de conception : soit couplés en interne aux modèles dynamiques dans ces composants ICAr, soit pour de la co-simulation.

Dans le chapitre 3, nous nous intéresserons à la problématique de l'interopérabilité au travers des couplages de modèles et d'outils métiers de modélisation magnétique en exploitant les formalismes ICAr et VHDL-AMS. Nous proposerons des solutions logicielles et des modules de génération de modèles adaptés à plusieurs environnements de simulation système.

Dans le chapitre 4, nous verrons comment il est possible de passer d'un formalisme de modélisation à un autre. Nous nous appuierons sur le paradigme de l'ingénierie dirigée par les modèles (IDM) et les concepts de transformation et de projection de modèles. Partant de là, nous montrerons que ceci nous permettra de formaliser le passage d'outils de modélisation métier vers le langage VHDL-AMS. En effet, des outils métiers dédiés peuvent faciliter grandement la mise en œuvre des modèles. Cependant le formalisme VHDL-AMS n'est pas forcément apte à supporter ces modélisations.

Nous apporterons des solutions à ces problèmes, en spécifiant la transformation à partir des modélisations métiers issues de ReluctTool (systèmes électromécaniques modélisés par réseaux de réluctances) et MacMMems (MEMS magnétiques) vers le langage VHDL-AMS. Nous validerons des transformations automatiques sur l'outil ReluctTool.

Quant à lui, le dernier chapitre sera consacré à l'application de la méthodologie PVF vis-à-vis de la problématique de conception des actionneurs magnétiques. Nous étudierons les différentes étapes du cycle, en exploitant au mieux VHDL-AMS, grâce aux solutions d'interopérabilité proposées dans nos travaux.

CHAPITRE 1

Problématique de la conception des systèmes électromagnétiques

A. Positionnement et problématique de la thèse

Il existe aujourd'hui une nécessité de faire évoluer les outils de simulation pour les amener à un niveau d'outils de conception, afin de prendre en compte les nouveaux besoins des concepteurs. En effet, les systèmes sont de plus en plus complexes faisant cohabiter plusieurs disciplines scientifiques et techniques. L'objectif est de fournir les outils et les méthodes adéquats qui guideront les concepteurs dans leurs processus de conception.

En adressant les problématiques de la simulation et de la conception au niveau système, le projet «Modélisation et Conception de Systèmes Mécatroniques» (MoCoSyMec), financé par l'ANR dans le cadre du programme Cosinus 2008, propose de répondre aux besoins de nombreux domaines industriels. En effet, des domaines comme celui de l'énergie, du génie électrique, des transports terrestres, de l'aéronautique, du bâtiment et bien d'autres encore sont de plus en plus demandeurs d'outils facilitant la conception de leurs systèmes.

L'objectif du projet MoCoSyMec est de proposer des supports logiciels de nouvelle génération pour la modélisation et la conception de systèmes mécatroniques. Outre leurs capacités à simuler, les outils recherchés doivent être capables de supporter les contributions de toute une équipe de conception mécatronique par nature multidisciplinaire.

Nos travaux de thèse représente une contribution à ce projet, qui met en collaboration l'éditeur de logiciels Cedrat[1], les industriels utilisateurs Schneider Electric[2] et Alstom[3], ainsi que les laboratoires FEMTO-ST[4], Ampère[5], InESS[6] et G2Elab[7].

[1] http://www.cedrat.com/
[2] http://www.schneider-electric.com
[3] http://www.alstom.com/
[4] Institut Franche-Comté Electronique Mécanique Thermique et Optique - Sciences et Technologies http://www.femto-st.fr/
[5] Laboratoire génie électrique, électromagnétisme, automatique, microbiologie environnementale et applications http://www.ampere-lab.fr/

Les travaux à réaliser dans ce projet se répartissent comme suit :

- Cedrat, en tant qu'éditeur de logiciels pour le monde du génie électrique et de la mécatronique, a pour objectif de proposer une plateforme de simulation système qui sera une référence technologique pour la conception des systèmes mécatroniques.
- Adapted Solution, la filiale allemande de Cedrat, s'est occupée du développement logiciel de cette plateforme.
- Le laboratoire Ampère, à travers son équipe « Électronique de puissance et intégration », contribue à développer et enrichir la plateforme à travers l'usage des graphes de liens ainsi que de leurs bibliothèques de modèles dans le domaine.
- L'Institut InESS, à travers son équipe de « Technologie de la Conception », se focalise sur les méthodes de conception des systèmes complexes et notamment l'étape des spécifications et leur interaction avec le cycle de conception. Ces méthodes ont été valorisées suite aux travaux de Dr. Y. Hervé.
- Le laboratoire G2Elab, et plus particulièrement l'équipe « Modèles, Méthodes et Méthodologies Appliqués au Génie Électrique », se propose de contribuer en apportant des méthodes, des méthodologies et des outils dédiés aux systèmes électromagnétiques à travers les travaux de cette thèse.
- L'Institut FEMTO-ST, à travers son département « Énergie » spécialiste dans la modélisation, la conception et l'instrumentation des systèmes de conversion d'énergie, se positionne en tant qu'utilisateur final pour la conception des applications industrielles telles que des moteurs-roues pour véhicule lourd.
- Schneider Electric, spécialiste de la gestion de l'électricité et des automatismes et expert dans les métiers de la mécatronique, se positionne en tant qu'utilisateur final.
- ALSTOM TRANSPORT, spécialiste dans la conception des modules de puissances et chaînes de traction pour sa gamme de train, se positionne en tant qu'utilisateur final.

Ainsi, ces travaux vont apporter à ce projet des méthodes, méthodologies et outils dans les domaines du génie électrique (actionneurs électromagnétiques et microsystèmes magnétiques) en se focalisant plus particulièrement sur la simulation système. Nous allons spécifier la plateforme de prototypage virtuelle dans le contexte du génie électrique. Nous allons apporter des solutions d'interopérabilité et de capitalisation pour supporter la modélisation des systèmes électromagnétiques dans leurs processus de conception. Nous validerons cela notamment sur des applications industrielles en collaboration avec Schneider Electric.

En résumé, notre thématique de recherche est de gérer et proposer des solutions d'**Interopérabilité de modèles et des outils dans le cycle de conception des systèmes électromagnétiques via des supports complémentaires.**

[6] Institut d'Électronique du Solide et des Systèmes http://www-iness.c-strasbourg.fr/
[7] Laboratoire de Génie Electrique de Grenoble www.g2elab.grenoble-inp.fr/.

B. Contexte applicatif

B.1. Systèmes Mécatroniques

Avec l'intégration des systèmes électroniques et leur miniaturisation, de nouveaux systèmes de plus en plus complexes sont apparus et requièrent plusieurs disciplines scientifiques et techniques pour assurer au mieux une fonction. Ces systèmes sont rassemblés sous l'appellation «**mécatronique**». Au départ, la mécatronique se définissait comme une discipline transverse qui associe la mécanique (au sens large) et son électronique notamment pour la commande (capteur, contrôle, régulation, etc.). Les concepteurs ont profité assez vite de la richesse des possibilités de l'association d'autres disciplines techniques. Depuis, la mécatronique n'est plus simplement l'interface entre deux domaines mais inclut bien d'autres disciplines (magnétique, optique, thermique, etc.), comme l'illustre la Figure 1-1.

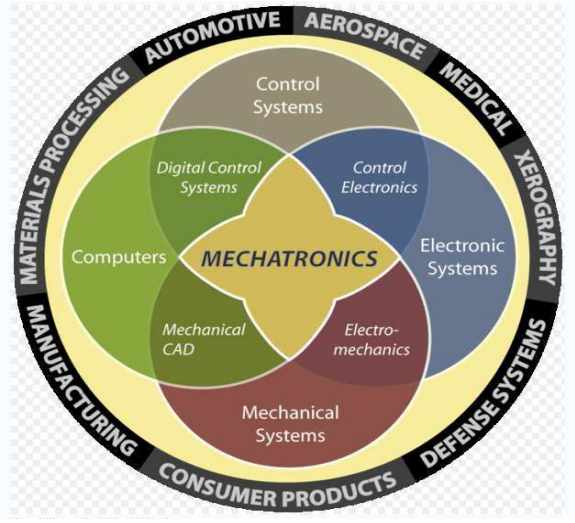


Figure 1-1 : Synergie de disciplines en Mécatronique [ISE-08]

La mécatronique est arrivée au niveau de maturité industrielle qui rend indispensable l'édition de normes spécifiques sur ses aspects qui ne sont pas couverts par les normes existantes. En 2008, la première norme spécifiquement mécatronique **« NF E01-010 »** a été élaborée, proposant à la communauté industrielle et scientifique (française puis européenne) un ensemble de termes et de définitions dédiés au domaine de la mécatronique [ANF-08].

La norme définie donc la mécatronique comme :

« 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» [ANF-08].

D'après les domaines scientifiques énumérés dans cette définition, la mécatronique concerne le développement de produits ayant la capacité de percevoir leur milieu environnant, de traiter l'information, de communiquer et d'agir sur leur milieu. Les mots « fonctionnalité » et « informatique » sont pris au sens large incluant respectivement la notion de valeur ajoutée et intégrant notamment le traitement de l'information et la communication.

Fondamentalement, un système mécatronique a pour finalité une action physique pilotée. Ainsi, les composants suivants sont abordés dans son champ d'applications :

- des actionneurs autonomes à faible et à forte puissance ;
- des organes de conversion, de stockage et de gestion de l'énergie ;
- des systèmes communicants, dont les technologies sans fil.

Nous allons nous intéresser, plus particulièrement, aux systèmes électromagnétiques qui représentent un sous-ensemble de la mécatronique.

B.2. Systèmes électromécaniques

Un système électromécanique est constitué d'un ou plusieurs éléments mécaniques assurant son mouvement ou sa déformation, des éléments électriques (et/ou électroniques) utilisés comme source d'énergie et des éléments de commande. Ces systèmes permettent la transformation de l'énergie électrique en énergie mécanique ou inversement, faisant potentiellement appel à une forme intermédiaire d'énergie. Celle-ci qualifie l'affiliation du système ; elle peut être créée par :

- un champ électrique pour les familles électrostatiques et piézoélectriques,
- un champ magnétique pour les familles magnétiques et magnétostrictifs,
- de l'énergie calorifique pour les familles thermiques.

Nous allons nous intéresser qu'aux systèmes magnétiques dans leur environnement électromécanique. Ce sont des systèmes multi-physiques intégrant des aspects mécaniques, électriques et magnétiques couplés. Ceux-ci peuvent être autonomes, en liaison avec les données issues du monde physique ou connectées à d'autres systèmes.

On peut noter que les études sur les lois de réduction d'échelle ont montré que les systèmes magnétiques doivent être étudiés différemment selon les échelles spatiales considérées [CUG-07]. C'est pour cette raison que nous allons étudier séparément les macro-systèmes de type actionneurs électromagnétiques et les microsystèmes électromécaniques, représentant une activité importante au G2Elab.

B.2.1 Les MEMS magnétiques

Un microsystème magnétique est un sous-ensemble de microsystèmes électromécaniques (MEMS : Micro Electro Mechanical System) qui représentent une classe très importante de systèmes ayant des applications allant des micro-capteurs et actionneurs, aux commutateurs radiofréquence ou encore des micro miroirs en optique [CUG-07]. Un MEMS magnétique est défini comme étant un dispositif électromagnétique de petite taille qui permet d'exécuter des fonctions de mesure (e.g. position, vitesse, accélération, etc. via des mesures magnétiques) ou des transformations énergétiques (déplacement, rotation, choc, génération d'électricité, …). Du point de vue structurel, un MEMS magnétique est, en soi, un tout petit système hétérogène exploitant différents phénomènes physiques, qui interagissent les uns avec les autres.

Les propriétés du magnétisme à petite échelle offre aux MEMS des possibilités d'actionnement avec une force conséquente et avec un plus grand déplacement relatif [CUG-06], en ne nécessitant pas des tensions importantes [NIA-03]. Leur inconvénient réside dans une consommation de courant engendrant des pertes. Par contre, l'exploitation d'effets

complémentaires multi-physiques tels que la piézo-électricité associée à la magnétostriction permettent de s'en affranchir [DEL-08]. La génération d'électricité par effet induit d'un champ en mouvement, comme c'est le cas à l'échelle classique, n'est pas adaptée ; d'autres solutions sont généralement envisagées [BEE-07]. La réduction d'échelle permet par contre des choses inimaginables autrement ; telles que la lévitation passive (sans apport d'énergie) par interaction entre aimants et particules diamagnétiques [CHE-07] [KAU-09]. Les MEMS magnétiques offrent des fonctionnalités intéressantes en actionnement, qu'il s'agisse d'efforts issus de sources de courant ou tout simplement grâce à des aimants.

À titre d'illustration, la Figure 1-2 présente un micro actionneur magnétique construit autour d'un aimant mobile, qu'il est possible de faire commuter suivant la direction verticale [ROS-05]. Il est maintenu dans les états stables par des aimants fixes et le changement d'état est assuré par le passage d'un courant dans les conducteurs. Il est particulièrement adapté pour la commutation rapide nécessaire aux systèmes radiofréquences.

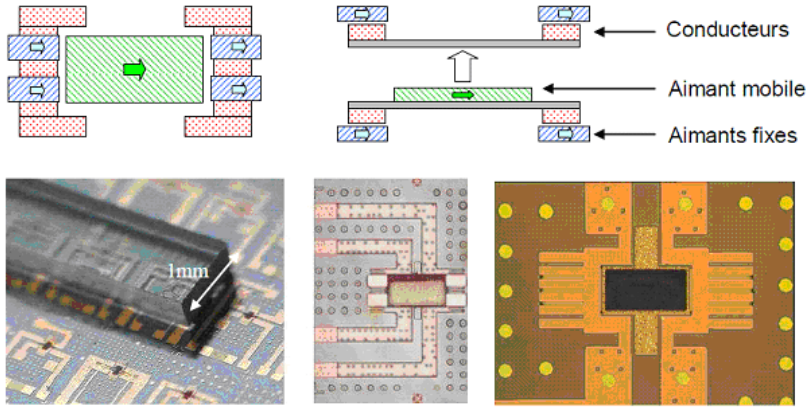
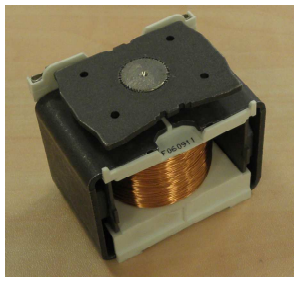


Figure 1-2 : Commutateur MEMS magnétique [ROS-05]

B.2.2 Les actionneurs électromagnétiques

Un actionneur électromagnétique est un mécanisme permettant de convertir de l'énergie électrique en énergie mécanique par l'intermédiaire d'une énergie magnétique. Les électroaimants, les électrovannes et les moteurs électriques sont des exemples d'actionneurs électromagnétiques. De tels dispositifs sont aussi massivement présents dans les installations électriques. C'est le cas des interrupteurs commandés servant à établir ou interrompre des courants électriques, tels que les contacteurs ou les disjoncteurs par exemple. Généralement, ils sont constitués de deux parties ferromagnétiques sur lesquelles les contacts électriques sont fixés (Figure 1-3) : une partie fixe et une partie mobile. La partie mobile est actionnée par un électroaimant qui crée ou redirige un flux magnétique, modifiant ainsi un équilibre de force conduisant au mouvement de la partie mobile [BRA-06].

(a) vue externe



(b) vue interne

Figure 1-3 : Contacteur électromagnétique

B.3. Modélisation des dispositifs électromagnétiques

L'équipe « Modèles, Méthodes et Méthodologies Appliqués au Génie Électrique » (MAGE) du G2Elab travaille principalement sur les problématiques de modélisation, de simulation et de dimensionnement des dispositifs électromagnétiques. Ces recherches se traduisent notamment par la réalisation d'environnements et d'outils logiciels dédiés.

Nous allons voir ici les principales méthodes de modélisation d'un dispositif électromagnétique pouvant être utilisées dans le cadre de la conception système. Chaque méthode étant plus ou moins bien adaptée aux usages et aux spécificités du dispositif étudié.

B.3.1 Méthode numérique : éléments finis

Les géométries complexes et la non linéarité des matériaux rendent la résolution analytique très complexe ; le recours à des méthodes numériques est impératif. Ainsi, la méthode des éléments finis (FEM) est souvent la plus utilisée dans l'étude des composants ou systèmes électromagnétiques. Elle permet d'effectuer des analyses de structures en mode statique et dynamique dans les domaines linéaires et non linéaires. En électromagnétisme, cette méthode exploite les équations de Maxwell en régime stationnaire et des lois de comportement des matériaux [MEU-08]. Il s'agit des méthodes numériques permettant de ramener le problème d'équations aux dérivées partielles à un système d'équations algébriques en discrétisant le domaine d'étude avec des conditions aux bornes et/ou à l'intérieur du domaine [COU-85].

Le concepteur dispose d'une panoplie d'outils de simulation lui permettant de créer ses modèles. Pour référence, nous citons les outils commerciaux comme Flux2D/3D[8] et Maxwell[9] spécialisés dans la simulation des dispositifs électromagnétiques, CoventorWare®[10] et MEMS Pro®[11] dédiés particulièrement aux microsystèmes et enfin COMSOL® Multiphysics[12] pour la simulation multi-physiques générique.

Cependant, l'utilisation de ces logiciels demande des compétences spécifiques et un niveau d'expertise avancé et se révèle parfois complexe. D'autre part, cette méthode débouche sur des modèles numériques assez lourds. Elle demande des temps de calcul conséquents et elle

[8] http://www.cedrat.com/en/software/flux.html
[9] http://www.ansys.com/Products/Simulation+Technology/Electromagnetics/Electromechanical+Design/ANSYS+Maxwell
[10] http://www.coventor.com/products/coventorware/
[11] http://www.softmems.com/mems_pro.html
[12] www.comsol.com/

est source de consommation importante de mémoire. Généralement, ces méthodes sont peu adaptées pour aborder un système à haut niveau. Par contre, elles le sont parfaitement dans les phases d'étude détaillée d'un composant du système.

B.3.2 Méthodes de réduction de modèle

La réduction de modèle est une approche permettant de créer des modèles légers à partir de simulations numériques en dégradant le moins possible leur précision. Cette approche est souvent cruciale pour accélérer la simulation et le dimensionnement des systèmes électromagnétiques à grande échelle [LIE-06]. La difficulté est de trouver le bon compromis entre gain de temps et perte de précision.

Une technique issue des travaux de Fischer [FIS-35], appelée plans d'expériences numériques, permet de construire une surface de réponse analytique extrapolée à partir d'un nombre minimum de calculs du modèle fin. Ces expériences permettent d'identifier un modèle réduit dans un but d'optimisation à partir de simulations ou de mesures [COU-08]. Pour les systèmes dynamiques, la réduction de modèles consiste à réduire l'ordre du système.

Ce type d'approche a été mis en œuvre dans FGOT[13] (un outil du G2Elab) et dans iSIGHT[14] pour la création et l'exploitation de ces modèles réduits, notamment pour du dimensionnement par optimisation [NUG-04] [GUE-05]. L'outil Lysis™[15] offre aussi un environnement intégré pour la modélisation, l'analyse et le dimensionnement en s'appuyant sur l'approche des « plans d'expériences ». Le simulateur ANSYS[16] incorpore des algorithmes pour la réduction de modèles mécaniques et l'exportation de leurs équations. Ces modèles peuvent être ensuite utilisés pour la modélisation au niveau système [SCH-05].

D'autres approches de modélisation comportementale sont retenues dans les outils industriels de type MEMS-Pro[17] pour les microsystèmes électromécaniques. Elles reposent essentiellement sur la réduction des modèles depuis des modèles FEM. Par contre, cet outil ne propose actuellement pas de composants électromagnétiques dans sa solution logicielle.

B.3.3 Méthode analytique : formalisme circuit

Dans les premières phases de conception, un modèle extrêmement précis n'est pas toujours nécessaire ; un modèle approché est envisageable. En électromagnétisme, un formalisme basé sur une représentation par circuit électrique équivalent permet d'élaborer de tels modèles : les réseaux de réluctances [ROT-41]. Cette technique permet de modéliser un système en régime statique ou dynamique par un schéma descriptif équivalent du circuit magnétique. Dans cette approche, les phénomènes sont représentés d'une manière macroscopique et un nombre restreint de composants est nécessaire à la reproduction du comportement du dispositif.

La principale difficulté de ce type de modélisation réside dans l'identification des différents tubes de flux, ce qui demande au concepteur un certain savoir-faire. De plus, leur mise en équation et leur implémentation informatique restent délicates et complexes. En

[13] http://www.cedrat.com/en/software/got-it.html
[14] http://www.3ds.com/products/simulia/portfolio/isight-simulia-execution-engine/overview/
[15] http://www.infiniscale.com/index.php?id=36
[16] http://www.ansys.com
[17] http://www.softmems.com/mems_pro.html

réponse à cette problématique, les travaux de B. du Peloux ont abouti à un outil de modélisation par réseau de réluctances pour le calcul et le dimensionnement statique des composants électromécaniques et des machines électriques, baptisé RelucTool[18] [DUP-06]. Une nouvelle version a été développée lors des travaux de T.P Do, mettant en œuvre la simulation dynamique de ces systèmes pour des actionneurs linéaires et capteurs [DO-10].

B.3.4 Méthode semi-analytiques : composition et formulations intégrales

Lorsque le dispositif électromagnétique ne permet pas de définir des tubes de flux, le formalisme circuit ne peut pas s'appliquer. C'est le cas dans la plupart des microsystèmes magnétiques (MEMS) [RAK-07], où la réalisation de parties ferromagnétiques pour conduire le flux est un défi technologique. Contrairement à des dispositifs macroscopiques, les MEMS sont généralement réalisés avec des formes géométriques simples (parallélépipèdes, secteurs cylindriques et sphères) pour faciliter la fabrication. De ce fait, un MEMS peut être décomposé en plusieurs entités modélisées analytiquement. Une approche de composition, basée sur le principe de composants en interaction, est utilisée pour les modéliser globalement [RAK-07]. Il s'agit de décrire des composants pouvant (Figure 1-4) :

- réagir à un champ magnétique d'excitation,
- créer des forces et/ou des couples mécaniques,
- produire un champ magnétique.

Un logiciel dédié à la modélisation des MEMS magnétiques, appelé MacMMems[19], a été développé durant la thèse de H. L. Rakotoarison au G2Elab [RAK-07]. Il met en œuvre l'approche d'interactions magnétiques distantes et non canalisées par des parties ferromagnétiques. Il se base sur une modélisation analytique et semi-numérique par des formules intégrales résolues formellement en fonction des géométries des composants.

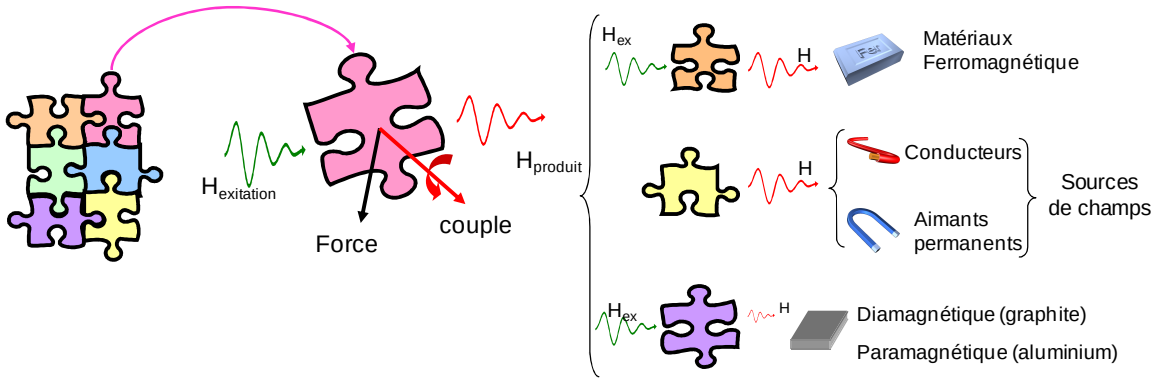


Figure 1-4 : Un composant magnétique dans MacMMems [RAK-07]

[18] http://www.g2elab.grenoble-inp.fr/recherche/reluctool-pre-dimensionnement-et-modelisation-statique-et-dynamique-par-reseaux-de-reluctances-de-systemes-electromagnetiques-286653.kjsp?RH=G2ELAB_R-MAGE

[19] http://www.g2elab.grenoble-inp.fr/recherche/macmmems-macro-modeler-for-magnetic-mems-220145.kjsp?RH=G2ELAB_R-MAGE

C. Conception des systèmes électromagnétiques

C.1. Une approche « système » nécessaire

Grâce aux avancées technologiques et à l'évolution des besoins du marché, les systèmes mécatroniques connaissent un essor important. Cependant, leur nature hétérogène couplée avec la multitude des phénomènes physiques, qui ont lieu au cours de leur fonctionnement, rendent leur conception complexe. De par leur aspect multi-domaines, la collaboration entre les différentes équipes métiers impliquées devient primordiale dès le début de la conception et une approche plus globale est nécessaire.

En effet, jusqu'alors, la conception se faisait au travers d'une juxtaposition de différentes disciplines et des différentes approches orientées « métiers », plus qu'au travers d'une réelle synergie de celles-ci. Cela conduit à décomposer un système en une arborescence de sous-systèmes et éléments «métiers» (mécanismes, capteurs, énergie, commande…). Ensuite, il s'agit de les étudier séparément afin de structurer le développement autour de différents périmètres fonctionnels et organiques (généralement spécialisés). Cette approche conventionnelle a abouti à ce qu'on appelle l'*ingénierie séquentielle* (appelée aussi approche métier) [PAH-96]. Par ailleurs, l'application des méthodes spécifiques à un domaine n'entraîne généralement pas une conception optimale du système. Même si chacun des spécialistes travaille de la manière la plus efficace possible, son isolement fait qu'il ne possède qu'une idée partielle sur la conception du système global. Il devient donc difficile de prévoir les interférences entre eux ou avec l'environnement, ainsi que les performances et la fiabilité du système global et de réaliser des optimisations globales sur le système. Les inconvénients de ce développement par discipline et l'intégration tardive du système sont présents dans de nombreuses publications, on notera [LI-01], [VAN-03], [ROT-02], [SCH-04] et [THR-05].

Ainsi, il apparaît important dans l'étude des systèmes mécatroniques de savoir intégrer les différentes technologies et disciplines afin qu'elles interagissent entre elles de manière optimale. Cette synergie nécessite une réelle communication entre les différents spécialistes durant leur activité et l'interconnexion entre les disciplines est prise en compte dès le début du processus de conception. L'étude de ces systèmes a induit un changement de méthode de conception, d'où l'apparition de l'*ingénierie concourante* (aussi appelée approche système) [SOH-92]. Cette approche exige une conception couplée par intégration des aspects mécaniques, électroniques, automatiques et informatiques et impose le concept de l'ingénierie collaborative [BOC-98] [TOL-98] [WAN-02]. Divers aspects relatifs à la conception collaborative pour le dimensionnement en génie électrique ont été abordés au G2Elab à travers les travaux d'I. Ammar [AMM-07].

La conception mécatronique est donc un sujet vaste. Elle peut être abordée sous différents angles, comme le montre la Figure 1-5.

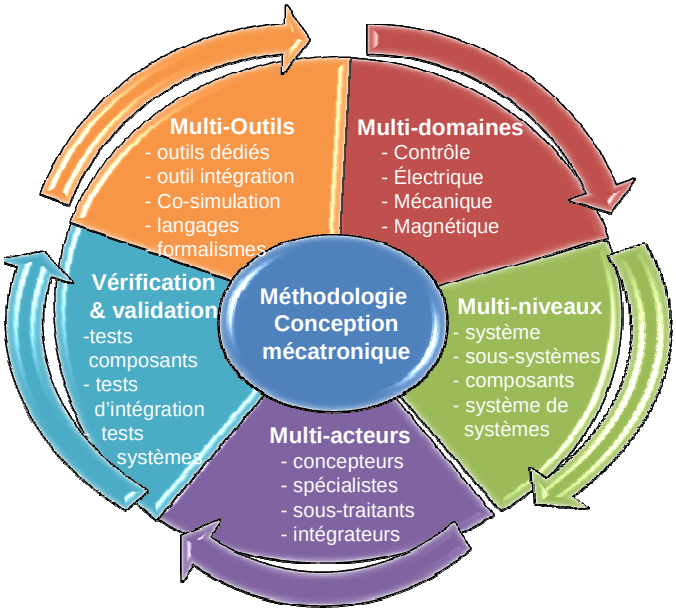


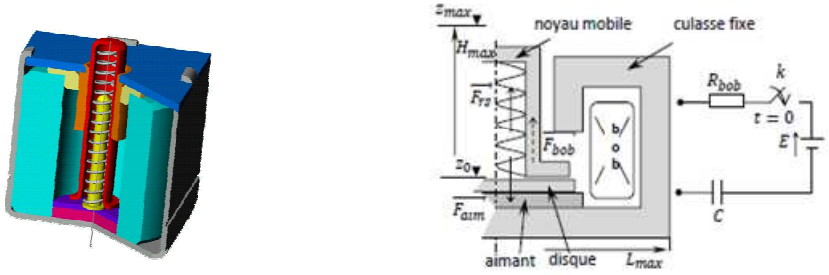
Figure 1-5 : Différents aspects de la conception mécatronique

Afin de se rapprocher du contexte d'une activité de conception multi-métiers et de délimiter les aspects traités dans notre problématique, nous allons illustrer la démarche de conception mécatronique ainsi que les difficultés et les besoins associés à travers un déclencheur électromagnétique [PER-98] [ATI-00]. Cependant, de grandes parties de cette démarche sont applicables à d'autres systèmes mécatroniques.

C.2. Conception d'un déclencheur électromécanique

C.2.1 Description du déclencheur électromagnétique

Le système étudié est un déclencheur électromécanique avec des pièces en mouvement dont le principe de fonctionnement est relativement simple, mais qui possède des caractéristiques multidisciplinaires suffisamment complexes (voir Figure 1-6).



(a) Vue 3D du dispositif (b) Demi section transversale avec ses éléments constitutifs

Figure 1-6 : Déclencheur électromagnétique

Ce déclencheur assure une conversion électromécanique permettant le déclenchement d'un mécanisme de coupure électrique. Cela se fait via un noyau mobile, qui passe de la position basse de repos à la position haute pour appuyer sur un actionneur provoquant la coupure d'un

courant. Ce type de dispositif a été étudié dans de nombreux travaux réalisés au G2Elab [PER-98] [GOW-03] [ATI-04] [DEL-04] [AMM-07] [ENC-09].

Le dispositif fonctionne comme suit :

- dans la phase non opérationnelle, le noyau mobile est en appui sur l'aimant en raison de sa force magnétique supérieure à celle du ressort.
- dans la phase opérationnelle, le courant circulant dans la bobine créé une force électromagnétique qui s'oppose à la force de l'aimant pour faire décoller le noyau. Celui-ci se déplace grâce au ressort et vient percuter un autre élément en fin de course.

C.2.2 Étapes de conception du déclencheur

Notre contexte de travail se situe dans le cadre de la conception en génie électrique. La conception d'un dispositif se déroule en plusieurs phases. Nous allons décrire ces diverses phases en partant de l'expression du besoin jusqu'à la réalisation des plans du dispositif.

C.2.2.1 Processus générique de conception en génie électrique

On peut définir le processus de conception comme étant l'exploration d'espaces de solutions et la vérification des solutions trouvées vis à vis d'un cahier des charges [DEL-04]. D'une manière générique, ce processus représente une stratégie de résolution d'un besoin tout en respectant un ensemble d'exigences formulées dans un cahier des charges, tout en gardant la possibilité de revenir en arrière lors d'une étape de la conception. En effet, chaque étape du processus peut remettre en cause une ou plusieurs étapes précédentes, obligeant le concepteur à revenir sur ses pas. Pour réaliser un tel processus, le concepteur peut dérouler une stratégie telle que celle décrite sur la Figure 1-7.

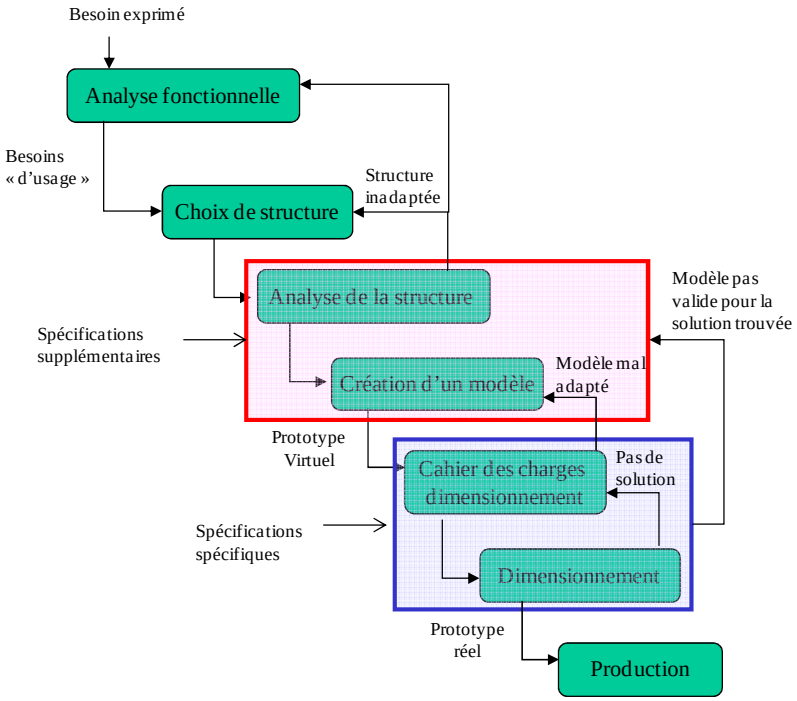


Figure 1-7 : Processus de conception des dispositifs

Ce processus débute généralement par une analyse fonctionnelle permettant de définir les fonctionnalités que doit offrir le dispositif afin de répondre aux besoins exprimés. Un

ensemble de jeu de spécification (cahier des charges) commence alors à être défini, puis sera affiné au fur et à mesure du processus. Cette analyse fonctionnelle permet ensuite de définir et choisir une structure du dispositif.

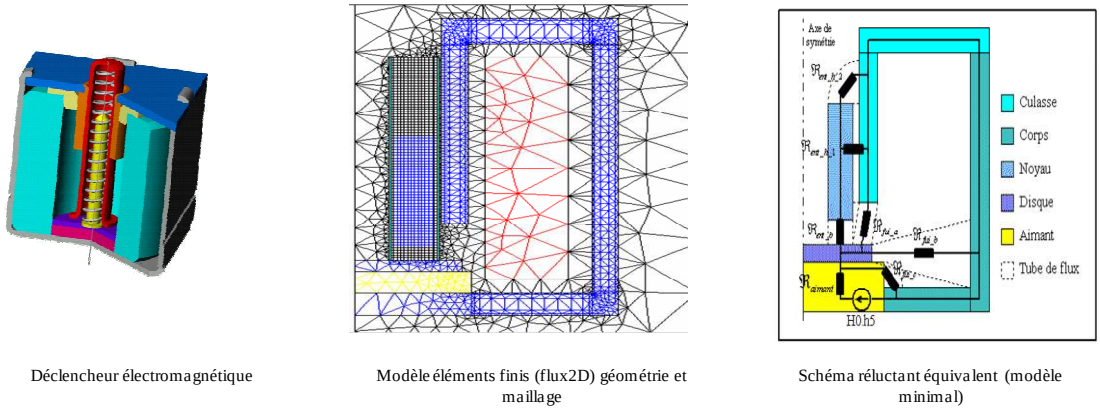
Une analyse de cette structure permet de vérifier le comportement du dispositif respectant son cahier des charges. Cette étape nécessite une ou plusieurs modélisations adéquates du dispositif en évaluant ses différents paramètres mis en jeu. Cette étape est validée et corrigée plus facilement (retour arrière et mise en question de la structure) par simulation, engendrant le besoin d'un prototype virtuel fonctionnel.

Ensuite, le concepteur entamera l'étape du dimensionnement afin d'affiner les modèles choisis selon différentes contraintes et les objectifs spécifiés définis dans un cahier des charges de dimensionnement. Enfin, une fois que le dimensionnement est achevé, il reste à vérifier et valider le fonctionnement du dispositif par simulation avant la construction des prototypes réels, faire de l'expérimentation, et enfin attaquer la production en série.

O note qu'en début de projet on n'a accès qu'aux besoins "d'usage". Ensuite, chaque étape de conception doit respecter ses spécifications d'entrée en tenant compte du modèle fourni à la sortie de l'étape précédente. Elle permet de fournir un nouveau jeu de spécification pour l'étape suivante. D'une façon générale, une étape de conception a donc en entrée un modèle, un jeu de spécifications et un jeu de contraintes et doit fournir un modèle validé (en accord avec les contraintes et les spécifications de tous les niveaux précédents et qui respecte la structure du modèle d'entrée) et un jeu de spécification pour le niveau suivant. Ainsi, le cahier de charges (les spécifications) est une structure hiérarchique, tout comme la conception. Il y aura un niveau de spécification par niveau de conception.

Nous pouvons, ainsi, conclure qu'avec la complexité croissante des systèmes actuels, la modélisation et la simulation deviennent primordiales à chaque étape de ce processus pour atteindre une véritable conception « systèmes ».

Dans le contexte de la conception des systèmes électromagnétiques, les modélisations à faire dépendent de ce que le concepteur cherche à étudier. Le modèle est fait dans un objectif d'étude précis ; par exemple, on n'utilise pas le même modèle pour la simulation système que pour une analyse fine d'un dysfonctionnement d'un composant. Il est donc parfois essentiel, pour le concepteur, de trouver des modèles simples pour une première étude haut niveau qui peut être réalisée dans un environnement de simulation unique. Il est aussi nécessaire de pouvoir ensuite décomposer ces systèmes progressivement jusqu'à leurs composants de base, pour s'assurer de leurs fonctionnalités. Cette étude détaillée est généralement faite dans l'environnement de simulation dédié à chaque composant. Le choix d'une modélisation (FEM, réseau de réluctances, etc.) plutôt qu'une autre dépend, à la fois des objectifs de la modélisation (usage ultérieur du modèle) et des outils à disposition (Figure 1-8). Ceci induit que chaque modèle doit être associé à une (ou des) application(s) spécifique(s) ainsi qu'à un besoin de modélisation particulier (conception amont, conception détaillée, ...) [WUR-08].



Déclencheur électromagnétique | Modèle éléments finis (flux2D) géométrie et maillage | Schéma réluctant équivalent (modèle minimal)

Figure 1-8 : Différentes modélisation du déclencheur

Nous allons nous concentrer sur les différentes modélisations du déclencheur mises en jeu dans le cycle de conception complet ainsi que les traitements qui y sont associés.

C.2.2.2 Différentes modélisations du déclencheur

- **Modélisation au niveau fonctionnel**

De façon générale, quand on commence un projet comme celui de la conception du déclencheur, on dispose d'un cahier des charges "utilisateur" sans contrainte sur la structure. La première étude à effectuer consiste à définir et analyser les spécificités de son fonctionnent (assurer une action mécanique à partir d'une commande électrique) et le cahier des charges associé. Par exemple, on dispose des contraintes ou des spécifications suivantes : la course à effectuer, le temps de déclenchement ou la force à piloter, alimentation, consommation, encombrement, coût, etc.

Le modèle du déclencheur peut être divisé en un ensemble de fonctions paramétrables répondant à ces spécifications. Ce partitionnement initial, dans notre cas, repose sur les aspects fonctionnels (blocage/mouvement) comme l'illustre la Figure 1-9, ainsi que ses différents états de fonctionnements et les conditions de passages entre eux.

À l'issu de cette étape, une étude sur l'architecture et la structure du système est effectuée par les concepteurs afin de proposer une structure bien adaptée (exploration architecturale). Dans le cas où on part d'un système connu, qui fonctionne dans un certain cadre, l'objectif de la conception est de le faire évoluer dans le cadre du nouveau cahier de charges (améliorer les performances par exemple).

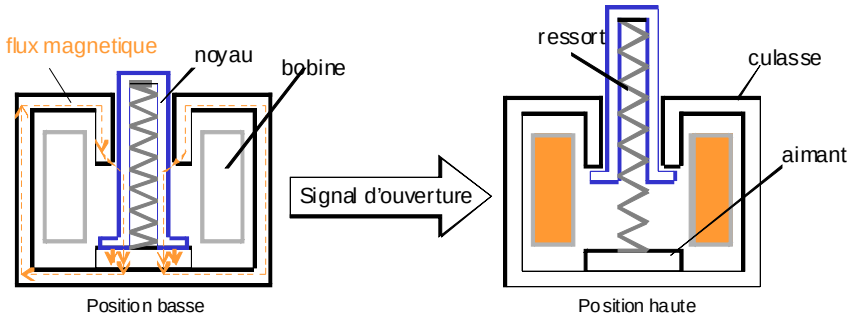


Figure 1-9 : Principe de fonctionnement du déclencheur

- **Modélisation au niveau système**

Une fois que les fonctions sont bien définies, il convient de proposer et d'analyser une structure pouvant répondre à celles-ci. Le déclencheur, présenté dans la Figure 1-9, assure une conversion électromécanique, via une énergie stockée dans un ressort, permettant le déclenchement d'un mécanisme de coupure électrique. Ce dispositif intègre donc des éléments mécaniques (ressort), électriques (bobine et son circuit), magnétiques (circuit magnétique avec aimant) et aussi des aspects de fabrication, le tout étant très couplé. Nous pouvons remarquer qu'il est difficile de définir le système sans décrire ses différentes parties et les interactions entre elles. La conception de l'ensemble nécessite l'intervention de compétences multiples : mécanique, magnétique, électrique, etc.

- **Modélisation à travers la décomposition du système**

Dans cette phase, nous décomposons le système en sous-systèmes et nous définissons leurs interactions. Le déclencheur contient alors un circuit magnétique, une bobine alimentée par un circuit électrique, un ressort en compression assurant le mouvement du noyau. Ce partitionnement doit prendre en considération les différentes interactions (Figure 1-10).

- Le circuit électrique est connecté au circuit magnétique car il alimente sa bobine,
- Le circuit magnétique est connecté à des pièces du système mécanique, qui viennent modifier ses propriétés lorsqu'elles sont en mouvement.

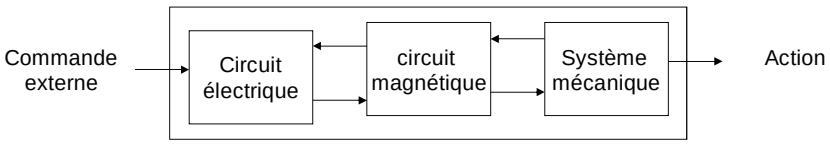


Figure 1-10 : Partitionnement du déclencheur.

À noter aussi que les concepteurs peuvent avoir d'autres difficultés de conception liées aux différents niveaux de granularités de décomposition d'un système. C'est-à-dire un composant peut lui-même devenir un système ou un système peut devenir un composant d'un autre système (notion de super-système). Il en ressort le besoin de différents niveaux de modélisation/conception et de pouvoir passer d'un niveau à un autre plus facilement.

Nous pouvons illustrer ceci sur le déclencheur, en ne nous limitant pas à considérer celui-ci seul, car qu'il est souvent nécessaire de considérer également le dispositif au sein duquel il sera utilisé. La Figure 1-11 présente une chaîne de coupure électrique, dans laquelle le déclencheur pourra être inséré en tant que composant du système plus global [ATI-04].

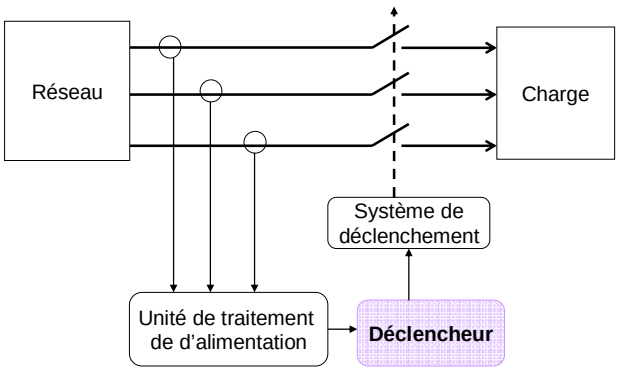


Figure 1-11 : Place du déclencheur dans une chaîne de coupure électrique

- **Modélisation détaillée des sous-systèmes/composants**

Une fois cette décomposition effectuée, il faut définir le contenu et le détail de chaque sous-système répondant à sa fonction et à son propre cahier des charges.

Dans notre exemple, les concepteurs experts de chaque domaine (électrique, mécanique, et magnétique) doivent intervenir afin de modéliser leurs sous-systèmes. Chaque concepteur pourra, par exemple, mettre en œuvre des modèles analytiques et/ou numériques, mais aussi des géométries paramétrées permettant de visualiser des optimisations ou des géométries 3D incluant la fabrication. Le choix des modèles dépend des besoins et des objectifs définis pour la conception (calcul, simulation fine ou grossière, dimensionnement).

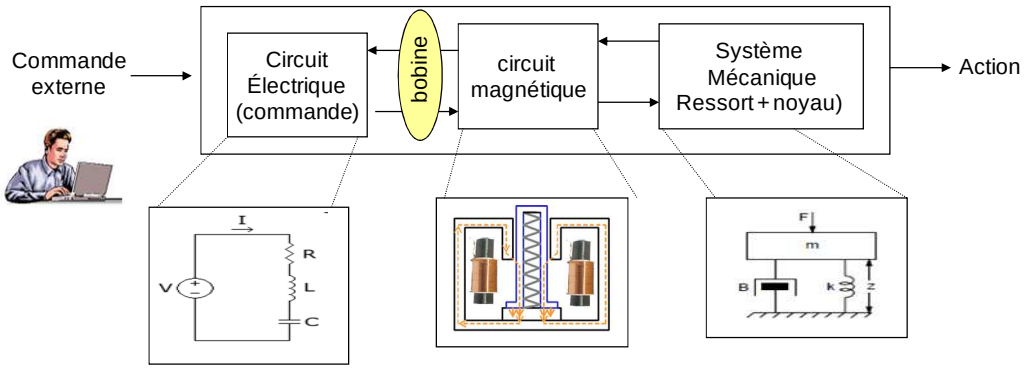


Figure 1-12 : Partitionnement et modélisation détaillée du déclencheur électromagnétique

La Figure 1-12 présente les différents modèles qui interviennent : le circuit d'alimentation simplifié est un circuit RLC commandé par un interrupteur et dont l'inductance dépend du flux magnétique du circuit magnétique. Le sous-système mécanique contient un ressort et le noyau subit des frottements. Ces deux systèmes garderont ce niveau de modélisation dans toute notre étude. Par contre, par l'activité du laboratoire G2Elab, nous nous situons en tant que concepteur des systèmes magnétiques. Nous pouvons alors exploiter différentes méthodes de modélisation. Le choix de la méthode dépend soit des niveaux de conception dans lesquels nous nous situerons durant le processus de conception, soit de l'objectif de l'étude.

- un modèle mathématique simplifié, élaboré dans les travaux de M. Perrault [ATI-00],
- une représentation sous forme de réseau de réluctances ; le calcul automatique de la force est obtenu par dérivation de la co-énergie [DUP-06],
- un modèle précis par éléments finis [PER-98].

Le modèle magnétique choisi dépendra donc des études visées initialement, par exemple pour une simulation système de haut niveau, un modèle simplifié est suffisant.

Fournir un prototype virtuel avec différents niveaux d'abstractions peut s'avérer intéressant, car selon ses besoins, le concepteur peut faire appel à un prototype plutôt qu'à un autre. Cela permet aussi le test d'un ensemble varié de scénarii de validation très tôt dans le cycle de conception.

À la fin de cette étape, les différents concepteurs spécialistes fournissent chacun leur modèle validé. On note aussi qu'une des principales conséquences de la diversité d'acteurs et de domaines est la multiplicité des outils de modélisation.

C.2.2.3 Exploitation et analyse des modèles du déclencheur

- **Analyse par simulation**

Dans les méthodes traditionnelles de conception, l'intégration du système est généralement traitée en fin de conception. Une fois que les paramètres du modèle sont bien identifiés et les modèles bien élaborés, les instances des sous-modèles sont assemblées pour une simulation du système global. Ceci dit, cette étape d'intégration et d'analyse du système global présente plusieurs difficultés que nous traiterons dans la suite de nous travaux :

- les modèles sont de natures différentes (magnétique, électrique, mécanique) et peuvent être sous différentes formes (circuit, schéma blocs, équations, etc.),
- les modèles sont souvent développés par des outils différents,
- l'environnement d'intégration (de simulation par exemple) doit autoriser le regroupement et la résolution de ces différents modèles,
- les concepteurs ont aussi besoin de communiquer entre eux, d'échanger leurs modèles et leurs résultats de simulation, et de les réutiliser dans d'autres contextes.

Ceci implique ainsi des difficultés en termes de compatibilités de formalismes de description ou de représentation des modèles, ainsi qu'en termes de couplages d'outils informatiques employés.

- **Dimensionnement**

Supposons qu'on dispose maintenant d'un modèle du système global, associé au déclencheur électromagnétique et qui soit simulable. Ce modèle de simulation est construit pour analyser le système, mais son utilisation seule est peu adaptée pour trouver une solution optimale ou du moins qui répond au cahier des charges. Il peut être intéressant de l'utiliser aussi dans un processus d'optimisation, comme le montre la Figure 1-13.

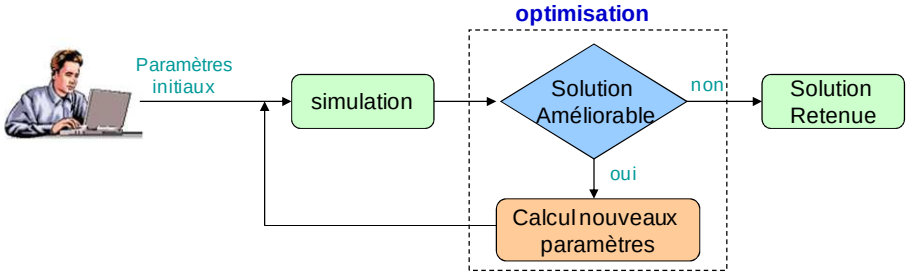


Figure 1-13 : Processus d'optimisation

Dans l'approche mécatronique, on ne teste pas le système comme un ensemble de composants technologiques assemblés, mais comme un ensemble de fonctions, ce qui permet la détermination des indicateurs de performance, qui sont les besoins réels perçus de l'utilisateur final. En appliquant ceci, l'optimisation permet de satisfaire la performance globale du système au lieu de ceux de chaque sous-système individuel. Cette approche est appelée *« optimisation au niveau système »*.

Toutefois, pour des composants à volume élevé de production, réutilisés dans plusieurs systèmes, il est plus rentable de les concevoir de manière générale sans présupposer des diverses applications possibles. Les inconvénients d'une optimisation indépendante de chaque sous-système suivie de leur intégration finale, sont analysés dans de nombreuses publications [LI-01] [VAN-03] [ROT-02] [SCH-04] [THR-05]. L'approche de l'optimisation système est relativement inexplorée en raison des nombreuses difficultés de mise en œuvre de modèles paramétrés du système.

C.2.3 Conclusion sur la conception système

La conception est une activité permettant de créer ou d'améliorer un produit, en partant des besoins exprimés, des moyens existants et des possibilités technologiques. Cette activité, illustré sur un dispositif électromagnétique, que l'on peut généraliser sur les systèmes mécatroniques est devenue complexe par :

- la diversité des domaines abordés : multidisciplinaire,
- la diversité des acteurs concernés : multi-acteurs,
- l'hétérogénéité des modèles développés : multi-formalismes,
- la diversité des outils utilisés : multi-outils.

Ainsi, les enjeux pour les équipes de conception de systèmes électromagnétiques sont nombreux et bien réels : systèmes de plus en plus complexes, technologiques disponibles, etc. De plus, tous les indicateurs semblent mettre en évidence une inadéquation entre ces enjeux et la démarche de co-conception actuellement utilisée.

C.3. Besoins émergeants des concepteurs systèmes

Devant la complexité croissante des systèmes mécatroniques, notamment dans le domaine électromagnétique, l'évolution des technologies et les démarches de co-conception actuellement utilisées ne permettent plus de maîtriser et limiter les temps de développement, tout en garantissant une qualité de la conception donnée. Les enjeux, essentiellement pluridisciplinaires, conduisent à la nécessité de développer de nouvelles méthodologies de conception et des nouvelles techniques de modélisation [HER-02].

Le Tableau 1-1 récapitule les enjeux et les besoins/solutions avancées afin d'augmenter la productivité de manière significative, en cohérence avec les avancées technologiques.

Enjeux	Besoins/Solutions
- Complexité croissante des systèmes - Évolution rapides des technologies - Réduction du Time-to-Market	➔ Approche de conception de haut niveau ➔ Approche de conception basée sur les modèles ➔ Approche multi-niveaux et multi-abstractions des modèles ➔ Portabilité des modèles (indépendance de la plateforme cible) ➔ Réutilisabilité des modèles
- Communications entre équipes multidisciplinaires et multi-compétences	➔ Formalismes communs pour les différentes équipes de conception ➔ Solutions d'interopérabilité entre modèles et outils ➔ Génération automatique de modèles et de documentations ➔ Capitalisation des compétences et de l'expérience
- Qualité du processus de développement	➔ Méthodologie de conception formalisée

Tableau 1-1 : Enjeux et besoins préconisés dans la conception de systèmes complexes

C.3.1 Besoin d'une méthodologie de conception adaptée

Les concepteurs de systèmes mécatroniques ont adapté la représentation du cycle de développement sous la forme d'un «V», selon le standard industriel VDI-2206 [VDI-04]. C'est un modèle couramment appliqué dans l'industrie, permettant l'organisation générale du travail, la décomposition et la distribution des tâches et la définition des étapes aboutissant à la production du système à partir des besoins. D'une manière générale, ce cycle convient très bien au développement de systèmes complexes dans lesquels interagissent un grand nombre de collaborateurs. En effet, chaque collaborateur peut réutiliser cette même structure d'analyse en «V» pour décomposer le développement de son propre sous-système en des étapes du même type. Cependant, la construction de ces systèmes complexes doit tenir compte des interactions dynamiques entre les différents phénomènes physiques mis en jeu. En effet, si ces interactions ne sont pas totalement prévues, comprises, anticipées et donc limitées ou non maîtrisées ; les conséquences peuvent provoquer des dysfonctionnements des équipements. Ces interactions entre les différents systèmes physiques ne sont pas clairement définies et gérées dans ce cycle, le suivi d'information entre les différents niveaux est absent. Les demandes actuelles des concepteurs sont telles que l'on doit se diriger vers des démarches de conception plus flexibles et robustes.

Le Prototypage Virtuel Fonctionnel [HER-06] permet de répondre à la majorité de ces contraintes dans la phase de développement des systèmes sur le plan technique. En effet, ceci se fait grâce aux capacités de simulation, mais aussi sur le plan de la communication au travers de la formalisation et modélisation du système sous forme de prototypes virtuels. Cette méthodologie permet de formaliser, d'échanger et de réutiliser le savoir des ingénieurs et ainsi de capitaliser de façon efficace la propriété intellectuelle.

C.3.2 Besoin d'une description multi-niveaux

Dans la plupart des problématiques de conception, il n'est pas possible de répondre convenablement aux besoins au moyen d'un modèle unique. Différents niveaux de modèles sont nécessaires pour décrire les différentes étapes de conception et réaliser différents scénarii de tests et analyses.

Nous avons identifié trois niveaux d'abstraction : fonctionnel, comportemental et physique.

- **Modélisation fonctionnelle.** Cela correspond au modèle caractérisant les fonctions offertes par le système pour satisfaire aux exigences du cahier des charges. Un diagramme d'états peut par exemple être utilisé pour cette modélisation.
- **Modélisation comportementale.** Le système est décrit soit par un modèle mathématique de son comportement, soit par un modèle de sa structure (circuit par exemple, pour lequel les éléments sont interconnectés en vérifiant les équations de Kirchhoff généralisées). Cette approche consiste à modéliser le système par l'évolution de ses entrées/sorties en réponse à différents stimuli.
- **Modélisation physique.** Le niveau physique est le plus proche de la réalité du système. Les propriétés et les lois comportementales des matériaux employées ainsi que la géométrie 2D ou 3D des structures sont introduites dans la description des dispositifs. Le comportement est décrit généralement par des équations aux dérivées partielles (PDE). les méthodes de résolution les plus utilisées sont les éléments finis (FEM[20]) ou les méthodes intégrales de frontière (BIM[21]).

Nous noterons tout de même qu'il n'y a pas toujours de correspondance exacte entre les niveaux de modélisation et ceux de conception, comme l'a montré [LEU-12].

C.3.3 Besoin de la capitalisation et réutilisation des modèles

Plus le système à concevoir est complexe, plus une collaboration entre acteurs est nécessaire. Un enjeu primordial pour une conception robuste et fiable de systèmes multidisciplinaires est la capitalisation et la réutilisation collaborative des connaissances et de l'expertise des ingénieurs. La capitalisation est le moyen de conserver, dans un but de réutilisation et de transmission, la connaissance du concepteur ou de l'entreprise. On définit par le terme réutilisabilité la démarche consistant à reprendre l'expérience aboutie d'une conception (de composant, de circuit ou d'une fonction) pour l'appliquer à un autre projet de conception. Cette approche peut aider le concepteur dans sa façon de gérer la complexité de son dispositif et lui permettre de tenir des délais liés à l'exécution de son processus. Elle permet de raccourcir les développements des modèles et de garantir les résultats grâce à des modèles robustes, déjà testés et validés. Ceci est possible grâce à divers supports qui garantissent une certaine pérennité des modèles [VAN-03] [BER-03].

Nous caractérisons la capitalisation de la connaissance selon deux critères principaux :

- **l'accès au savoir** qu'un concepteur a formalisé autour de modèles (bibliothèques de modèles). Une bibliothèque contiendrait idéalement :
  i. les modèles physiques, les spécifications remplies, la documentation associée, les limites d'utilisation, les notes d'application, etc.
  ii. le modèle exécutable pour divers environnements, réglages du simulateur (pour chaque test),
  iii. la possibilité d'adapter le modèle au mieux à un contexte différent [WUR-08].

[20] Finite Element Method (FEM)
[21] Boundary Integral Method (BIM)

- **la réutilisation de modèle** associée à la portabilité des formalismes utilisés par le créateur de modèle. Nous distinguons les formalismes permettant de décrire les sources du modèle (permettant une réadaptation) et ceux permettant d'exécuter le modèle dans des environnements différents.

Pour satisfaire ces critères, il faut proposer, développer et utiliser des formalismes, qui permettent de décrire des modèles portables et réutilisables.

C.3.4 Besoin en terme d'outils et de formalismes

Parallèlement au développement des méthodologies de conception, les dernières décennies ont connues un engouement très fort concernant les langages et formalismes pour la description de modèles de simulation.

- **Besoins en termes d'outils**

Pour être industriellement viable, un logiciel de conception se doit aujourd'hui :

- de présenter une interface graphique conviviale ou tout au moins intuitive pour pouvoir s'adresser au plus grand nombre d'acteurs,
- de prendre en charge la complexité des systèmes (interactions multi-physiques, interactions entre les domaines continu et discret, …),
- de faciliter l'accès des modèles aux concepteurs non spécialisés, ce qui se traduit par la gestion de bibliothèques de modèles et la capacité d'intégrer de nouveaux modèles,
- de posséder des fonctions de co-simulation et/ou d'import et d'export de modèles pour assurer l'interopérabilité avec d'autres outils logiciels de CAO[22],
- de supporter diverses analyses de sensibilité, voire d'être associé à des outils d'optimisation (tel que le projet Moscito pour la conception de MEMS [SCH-02]).

- **Besoins en termes de formalismes**

Pour appréhender au mieux les difficultés inhérentes à la modélisation et gérer la capitalisation, les concepteurs ont ainsi besoin d'un ou plusieurs formalismes :

- permettant de décrire d'une manière unifiée les différentes disciplines,
- assurant une continuité de conception en facilitant la communication entre acteurs,
- optimisant l'effort de la conception par la construction de bibliothèques,
- garantissant la réutilisation des modèles.

- **Synthèse sur ces besoins**

Les concepteurs systèmes ont donc besoin de langages ou de formalismes qui soient principalement **multi-domaines**, de **standards** pour l'échange de modèles et basés sur une approche **hiérarchisée**, et qu'ils soient supportés par les outils de modélisation, de simulation et de conception. Ces langages et formalismes de modélisation/programmation sont multiples comme nous le verrons par la suite, mais ils possèdent des limitations.

[22] CAO : Conception assistée par ordinateur

D. Vers un environnement de conception système

La réalisation d'une conception complète d'un système multi-physique complexe s'avère une tâche délicate confrontée à plusieurs problèmes aux niveaux des méthodologies, des modèles, des formalismes et des outils. Disposer d'un environnement dédié à la conception système est devenu indispensable de nos jours.

Nous allons présenter la solution de conception que nous proposons pour répondre aux problèmes soulevés. Ensuite, nous présenterons les solutions répondant plus particulièrement à la conception orientée simulation et/ou au dimensionnement. Enfin, nous étudierons différents formalismes existants permettant la mise en œuvre de ces solutions.

D.1. Méthodologie de conception pour la mécatronique

La plupart des processus de développement des systèmes mécatroniques suivent une variante du cycle de conception en V, tel que le décrit la directive VDI-2206 [VDI-04], développée par l'organisation allemande VDI[23]. La VDI-2206 fournit un cadre nécessaire pour la conception de tout système mécatronique. Elle définit un processus commun en identifiant ses principales phases, qui sont (Figure 1-14) :

- la conception système
- la conception spécifique
- l'intégration système

Lors de la conception système, à partir d'un cahier des charges d'usage, une équipe interdisciplinaire spécifie un concept de solution et un premier modèle système décrit les aspects fonctionnels, logiques et physiques les plus pertinents. La fonction globale du système est divisée en sous-fonctions, auxquelles un principe de fonctionnement ou un ensemble d'éléments de sa solution est assigné. Le but de cette solution fonctionnelle est de former une base commune pour le développement ultérieur spécifique à chaque discipline. Les performances de ses sous-fonctions sont vérifiées dans le contexte global.

La conception spécifique est consacrée à la définition détaillée de chaque élément de la solution système. Cette étape est généralement réalisée séparément par discipline technique (plusieurs petits projets de développement en parallèle) et par les spécialistes métiers correspondants, en utilisant pour chacun leurs propres méthodes, formalismes et outils. Diverses dépendances entre les processus et les modèles existent en général dans cette phase, ce qui pourrait provoquer de l'incompatibilité dans le système global. Par conséquent, les processus parallèles doivent être coordonnés et synchronisés.

Dans la phase d'intégration, les modèles de la phase conception spécifique sont intégrés dans un modèle du système global cohérent. Cette phase consiste à former et valider progressivement le système complet. Il est donc possible d'étudier les interactions entre les différents éléments du système ainsi que le comportement du système global.

À la fin du cycle, le système développé est évalué en comparant ses caractéristiques et ses performances avec les exigences du cahier des charges. Ceci doit se fait aussi à chaque niveau pour valider les modèles par rapport à leurs spécifications.

[23] VDI : Verein Deutscher Ingenieure (Association des ingénieurs allemands)

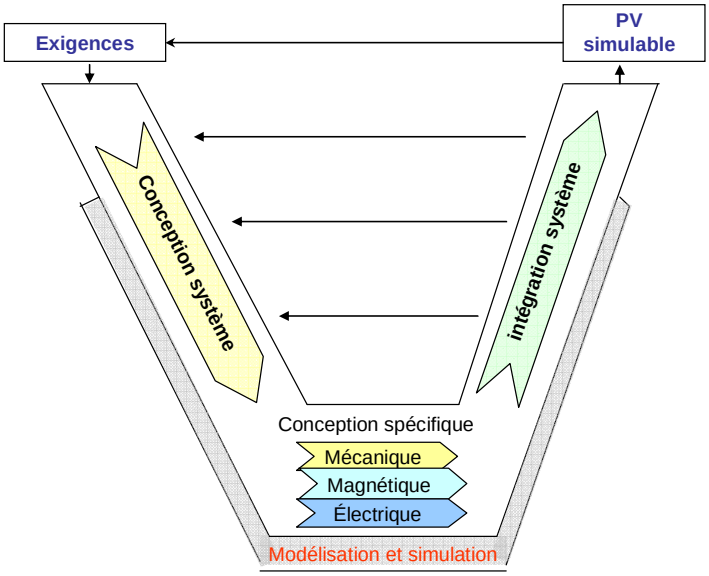


Figure 1-14 : Cycle de conception mécatronique [VDI-04]

La directive VDI-2206 définit théoriquement un processus de développement commun, des formalismes de modélisation complémentaires et une utilisation conjointe d'outils à travers les différentes disciplines. Toutefois, l'approche n'est pas suffisamment concrète pour soutenir effectivement la collaboration entre les différentes disciplines.

Il faut noter qu'un système complexe est rarement complètement réalisé en un seul cycle. En réalité, une série de cycles sont nécessaires. Ce flot de conception est un processus récursif, comme le montre la Figure 1-15. Un sous-système peut être étudié avec ce cycle de développement et peut être intégré comme un composant d'un autre système et dans un autre cycle. Les équipes en charge de la conception du produit vont ainsi développer et tester différentes solutions. Puis, le cycle en V sera successivement parcouru plusieurs fois, totalement ou partiellement, jusqu'à ce qu'une solution optimale soit trouvée.

Le concepteur crée et utilise des modèles de son système ou de ses éléments en fonction des différents stades de la conception, comme présenté sur la Figure 1-15. Par exemple, le premier cycle s'intéresse à l'analyse fonctionnelle. Le système est spécifié fonctionnellement, les principaux éléments sont définis, approximativement dimensionnés et leurs propriétés sont vérifiées dans le contexte global. Dans un deuxième cycle, la définition du système est approfondie pour aboutir à des prototypes fonctionnels plus réalistes, etc.

Par conséquent, selon l'avancement du développement, les objectifs de la conception ainsi que la complexité des problématiques rencontrées ; un certain nombre de cycles supplémentaires peuvent être nécessaires pour atteindre l'étape de production industrielle : pré-dimensionnement, analyse comportementale détaillée, analyse fine (simulation ou dimensionnement), analyse de sensibilité, etc. Un échange et une vérification entre les modèles issus de chaque cycle sont possibles.

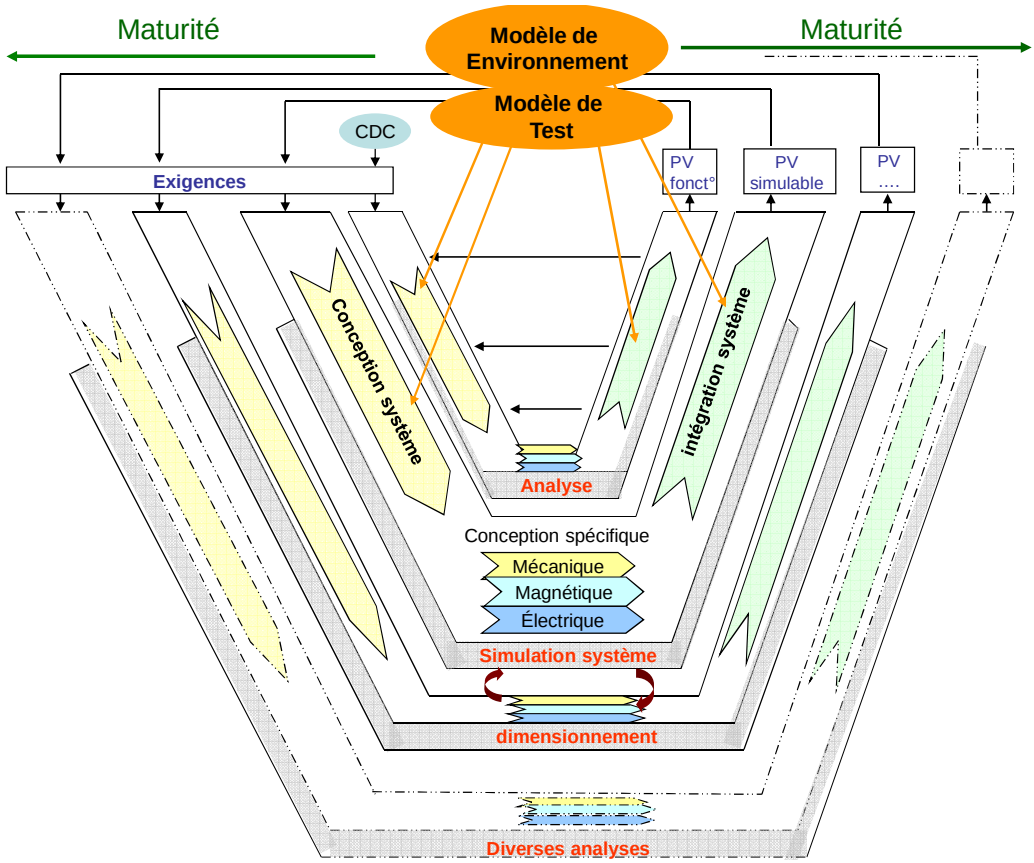


Figure 1-15 : Différents objectifs de conception [VDI-04]

Le but de cette approche de cycles multiples est d'appuyer les concepteurs et les collaborateurs du projet dans le processus de conception pour former des variantes de solutions selon leurs besoins. L'évaluation finale des solutions est soumise à des critères techniques et commerciaux spécifiques.

En résumé, l'analyse par simulation est une tâche primordiale dans le processus de conception. Quant au dimensionnement, il représente l'étape de recherche de la solution optimale dans ce processus itératif permettant, une fois que la structure du dispositif est fixée, d'optimiser les composants choisis (par simulations successives).

D.2. Formalismes spécifiques supports des étapes de conception

Nous allons maintenant nous concentrer sur les formalismes adaptés au prototype virtuel fonctionnel pouvant supporter les différentes étapes du processus de conception.

D.2.1 Analyse par simulation dynamique

Une approche de conception basée sur la modélisation consiste à intégrer massivement la simulation dans son processus de développement. Ceci présente un fort potentiel de réduction des coûts et des temps de développement et d'amélioration de la qualité et de la fiabilité.

D.2.1.1 Prototypage Virtuel Fonctionnel

Comme la directive VDI-2206 ne propose que le cadre méthodologique général, sans spécifier les détails de ses cycles de conception, une méthodologie de conception a été

proposée et formalisée par Dr. Y. Hervé en 2006, renforçant ce cycle. Cette approche, appelée «Prototypage Virtuel Fonctionnel», notée PVF par la suite, rassemble les acteurs du projet autour d'un langage commun basé sur les prototypages virtuels[24].

Le PVF est un ensemble de tâches bien organisées, utilisées pour concevoir ou améliorer un système à travers un ensemble de modèles «prouvés[25]» [HER-06]. En analysant les fonctions d'un système donné et en les caractérisant avec leurs spécifications, on peut construire un ensemble de modèles de complexité croissante qui permet une compréhension plus profonde du système à concevoir. Les études du système reposent sur des modèles descriptifs ou prédictifs de ses composants et sur la modélisation de l'environnement qui représente ses conditions d'exploitations. La Figure 1-16 illustre cette approche et montre comment les étapes de modélisation s'y inscrivent. Dans l'annexe 1-1, nous décrirons d'une manière générale ces différentes étapes et nous nous focaliserons sur l'application du PVF dans le chapitre 5.

Cette approche permet de formaliser, d'échanger, de capitaliser et de réutiliser des concepts sous la forme de modèles multi-niveaux d'un système multi-domaine. Le choix des outils et des formalismes adéquats permet également de faciliter le passage entre les niveaux d'une part, et les interactions entre les composants de diverses disciplines d'autre part.

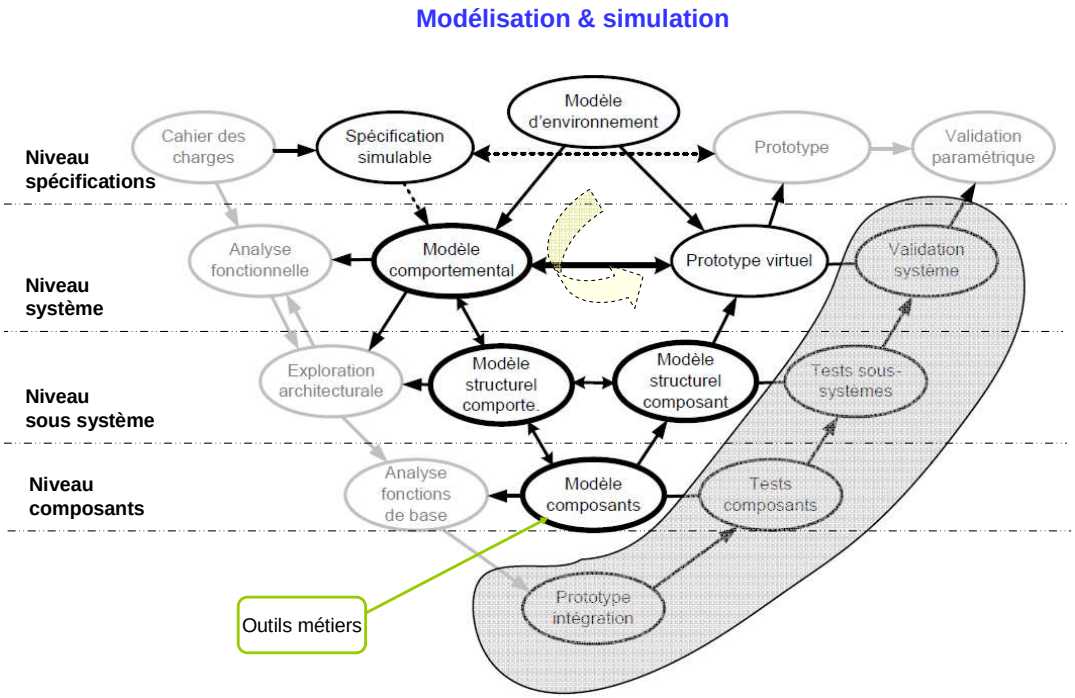


Figure 1-16 : Étapes du Prototypage Virtuel Fonctionnel [HER-06]

Il faut remarquer que cette méthodologie ne remplace en aucun cas les connaissances métiers. C'est pourquoi dans la pointe du cycle, il apparaît la phase de modélisation des composants. Cette modélisation doit s'appuyer sur les compétences et les outils métiers déjà en place pour tous les domaines impliqués, surtout quand un modèle prédictif très précis est

[24] Un prototype virtuel est un modèle multi-niveaux (comportemental, architectural, physique) d'un système multi-domaines (électronique, électrique, magnétique, hydraulique, mécanique, thermique ...).
[25] « Prouvé » : vérifiant les spécifications amont dans le cycle en V

requis. La richesse de l'approche est de permettre d'intégrer des modèles simplifiés des composants les plus complexes au niveau du système et de fournir des paramètres systèmes aux concepteurs de composants pour participer à des optimisations globales.

D.2.1.2 Modèles de simulation dans le cycle de conception

Comme le montre la Figure 1-17, différents types de modèles de simulation sont utilisés au cours d'un processus de conception. Dans la phase descendante du PVF, le système est progressivement défini et le niveau de détails des modèles s'accroît en conséquence. Dans un premier temps, on dispose d'une description fonctionnelle du système et donc de modèles fonctionnels de simulation (e.g. langage de spécification SysML[26]). Dans un deuxième temps, des concepts de solutions sont identifiés et les modèles de simulation physique ou comportementaux correspondants (équations analytiques généralement) sont élaborés. Durant la phase de conception spécifique, les différents composants sont définis plus en détails. Il est alors possible de réaliser des modèles numériques (par exemple Flux3D), qui permettent de vérifier leurs propriétés. Finalement, durant la phase d'intégration système, les différents composants sont assemblés pour former le système global et leurs performances sont validées.

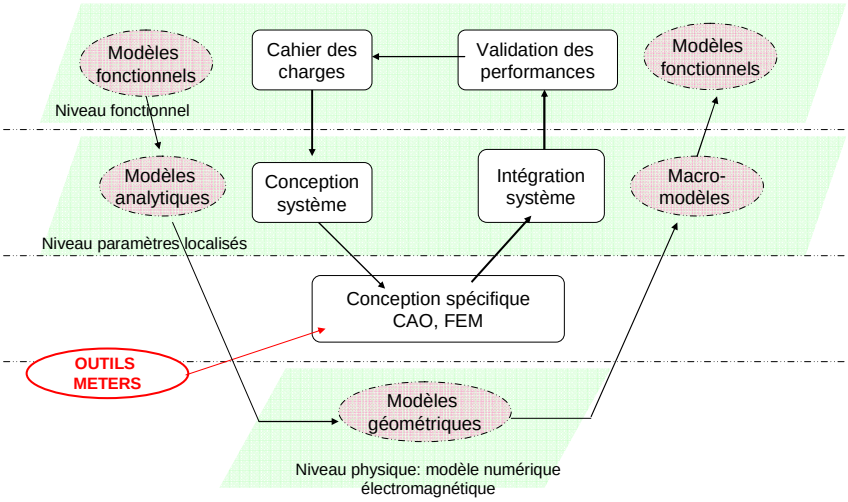


Figure 1-17 : Modèles de simulation dans le cycle de conception

L'objectif de la simulation système est d'évaluer le comportement global du système plutôt que de simuler très précisément des dispositifs. Les méthodes de résolution au niveau physique conviennent par exemple au calcul des déformations des structures MEMS. Cependant, les simulations FEM sont souvent des processus gourmands en temps et en ressources dans la conception des dispositifs complexes. La modélisation des propriétés physiques des dispositifs requiert nécessairement des paramètres d'entrée pour les modèles des niveaux supérieurs. Donc, pour la phase d'intégration, les modèles de simulation 3D, très détaillés, sont transformés en macro-modèles approchés moins détaillés : soit des modèles à paramètres localisés (réseau de réluctances), soit des modèles réduits. À chaque niveau, la difficulté réside dans une recherche d'un compromis temps/précision de calcul.

Pour un même niveau de détail, les différentes formes de modèles sont liées aux différentes disciplines rencontrées (mécanique, électrique, électromagnétisme, etc.). Cette particularité

[26] http://www.sysml.org/

soulève le problème de l'intégration des différents types de modèles dans un même environnement de simulation pour prendre en compte les interactions entre les différentes disciplines techniques et appliquer une approche de conception multidisciplinaire. Ceci pose une problématique complexe qui porte sur deux aspects :

- mettre en commun des connaissances spécifiques de chaque domaine,
- modéliser de manière précise ces éléments.

Il s'avère que ces deux aspects ne sont pas toujours corrélés. En effet, un modèle détaillé très complexe peut être difficilement exploitable, alors qu'un modèle trop simplifié risque de ne pas fournir l'information essentielle du système réel. Le type et la complexité de chaque modèle dépendent de son utilisation finale et du niveau de finesse des résultats attendus. Généralement, les modèles détaillés sont utilisés dans les étapes de conception avancées ; ils reposent sur les modèles développés au début du processus de conception. Ce dernier point soulève la problématique d'évolutivité des modèles au cours du PVF, dont la forme et le niveau de détail doit être adapté à chaque phase de conception.

D.2.2 Dimensionnement

Un concepteur est amené à rechercher une solution permettant de satisfaire au mieux des exigences, comme les besoins fonctionnels, des contraintes de fabrication et surtout une rentabilité économique. L'enjeu est de déterminer les différentes grandeurs relatives à son dispositif (dimensions géométriques, propriétés physiques des matériaux, coût économique, etc.). Ces grandeurs doivent satisfaire les contraintes exprimées par le cahier des charges et peuvent être de nature variée.

L'utilisation d'un logiciel qui supporte cette tâche, associée à une procédure d'optimisation, correspond à une stratégie performante largement utilisée : c'est le dimensionnement par optimisation. Ce type d'approche est d'ailleurs nécessaire quand le concepteur est amené à effectuer plusieurs étapes pour obtenir un modèle bien construit et une solution valide [DEL-05]. Il est possible de coupler automatiquement un modèle et un algorithme d'optimisation. La résolution d'un problème d'optimisation peut être faite via deux types d'algorithmes :

- les premiers sont déterministes et utilisent généralement des gradients (descente de gradient, quasi Newton, SQP[27]...). Ces méthodes sont rapides et supportent bien les problèmes contraints, mais se font piéger par des extrema locaux.
- les seconds sont stochastiques (algorithme génétique). Ils peuvent échapper à des extrema locaux mais sont moins performantes sur les contraintes et nécessitent souvent un très grand nombre d'évaluations du modèle.

La Figure 1-18 illustre cette approche. Le modèle de dimensionnement est chargé de calculer ses grandeurs de sorties pour un jeu imposé de grandeurs d'entrées. À partir de ces sorties, l'algorithme d'optimisation détermine alors un nouveau jeu pour les valeurs d'entrées et ainsi de suite dans un processus itératif qui permet de trouver un jeu de paramètres d'entrées pour que le dispositif réponde au cahier des charges.

[27] Sequential quadratic programming (SQP) : adapté aux modèles non linéaires sous contraintes.

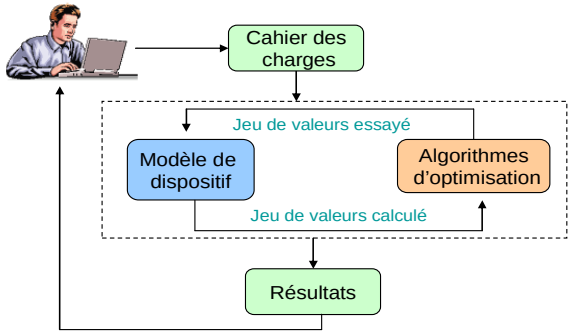


Figure 1-18 : Processus de dimensionnement

D'une manière similaire à celle présentée sur la Figure 1-17, différents types de modèles de dimensionnement sont utilisés dans les différentes étapes du cycle en V de conception. La Figure 1-19 illustre d'une manière simplifiée les différents modèles d'un actionneur électromagnétique mis en jeux dans un objectif de dimensionnement.

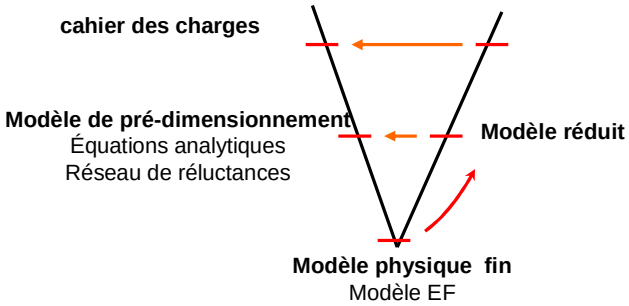


Figure 1-19 : Modèles de dimensionnement dans le cycle de conception

La phase de descente du cycle commence par l'extraction du cahier des charges de dimensionnent à partir des spécifications du dispositif. Cette tâche de traduction dépend fortement de l'expertise du concepteur vis-à-vis des grandeurs qu'il sera capable de fournir via son modèle. Ensuite, des modèles physiques ou comportementaux adaptés au dimensionnement sont élaborés. Cette étape peut correspondre à un contexte de pré-dimensionnement (dégrossissage du problème avec l'utilisation d'un modèle léger approché, pour rechercher les solutions possibles). Dans cette étape, on gère la majorité des contraintes de dimensionnement. Elle sera suivie du processus de dimensionnement ; qui consiste à affiner une solution sur des aspects délicats (saturation magnétique locale dans un actionneur). Cette étape utilise un modèle précis (conception spécifique) et travaille avec peu de contraintes sur ces modèles. Elle demande un temps de calcul plus conséquent.

Dans le cas général du dimensionnement de structures, la majorité des critères sont construits à l'aide de modèles macroscopiques ou approchés, on n'est donc pas obligé de parcourir la totalité du cycle. Il faut aussi noter que chaque étape est précédée par l'établissement de son propre cahier des charges, prenant en compte les différentes particularités du modèle. Par exemple, il est important d'ajouter des contraintes « non fonctionnelles », telles que des chaînes de côtes permettant d'éviter le chevauchement de pièces mécaniques, ou tout autre conditions permettant de contraindre le modèle dans un espace de validité.

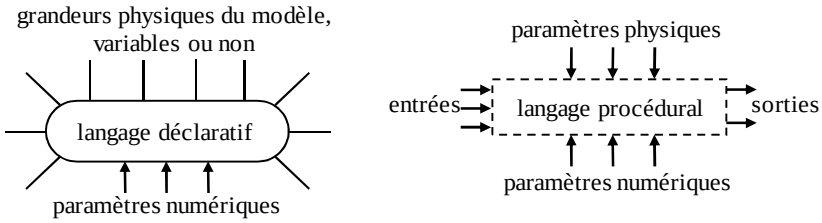
D.3. Formalismes de modélisation supports au PVF

Nous allons nous concentrer, par la suite, sur la modélisation orientée simulation dynamique et sur les différents niveaux de conception du PVF à l'exception du niveau de l'analyse fonctionnelle qui sera traitée « à la main ». Nous ne nous intéressons ici qu'aux outils logiciels et aux techniques nécessaires pour supporter et mettre en œuvre la méthodologie de conception proposée dans ce cadre restrictif. De plus, la spécificité de la conception des systèmes électromagnétiques conduit à s'orienter vers des moyens capables de prendre en compte les simulations dynamiques exigées dans un contexte multidisciplinaire et multi-acteurs, et ce à chaque niveau de développement du PVF. En conséquence, les concepteurs ont besoin de nouveaux outils en termes de langages et de formalismes de modélisation, mais aussi en termes de logiciels.

Parallèlement au développement des méthodologies de conception, les dernières décennies ont connu une véritable explosion de langages et de formalismes pour la description des problèmes de simulation. Ces langages et formalismes de programmation sont multiples mais ils ne sont pas tous appropriés à la conception système.

Nous allons distinguer deux grandes catégories de formalismes pouvant être adaptés à ce contexte et permettant la capitalisation. Ces catégories sont principalement issues de la description des modèles orientés ou non orientés [ALL-03].

Dans nos travaux, nous nous proposons d'appeler « boîte » un modèle orienté et « boule » un modèle non orienté, comme l'illustre la Figure 1-20. Cette distinction est importante car elle impacte fortement la façon dont on assemble les modèles et dont on traite leur résolution une fois qu'ils ont été composés.



La boule : modèle non orienté, acausal — La boîte : modèle orienté, causal

Figure 1-20 : Modèles non orientés (boule) et orientés (boîte) [ALL-03]

Les modèles peuvent être décrits de différentes manières sous la forme :

- d'un code explicite et ouvert (supporté par les langages descriptifs de modélisation),
- d'un code binaire fermé (supporté par la notion de composant logiciel)
- d'un compromis entre ces deux approches.

Les modèles peuvent être décrits dans un formalisme explicite, dans ce cas on parle du concept de «boite blanche» ou de « boule blanche ». Ce sont deux approches qui permettent de créer des modèles dont la connaissance (équations, fonctions, algorithmes) est explicitée et accessible en lecture et en écriture. Le concepteur peut ainsi profiter de l'expertise d'un autre, la modifier et l'enrichir pour que le résultat soit plus adapté à ses besoins. C'est par exemple le cas pour des modèles programmés dans des langages informatiques (Java, C, etc.), dans des

langages d'environnement numériques (Matlab ou Scilab[28]), ou encore dans des langages de modélisation normalisés (VHDL-AMS, Modelica). Nous nous concentrerons par la suite sur les langages de modélisation favorisant généralement la représentation de « boule blanche ».

Par opposition, le concept de «boîte noire» ou « boule noire » est plus classique, il permet d'appréhender un objet sans connaissance nécessaire sur son contenu (modèles accessibles sous forme binaire). Seules les relations entre les entrées et les sorties peuvent être observées, correspondant à son comportement et non pas à sa constitution. Généralement, les composants ainsi modélisés apparaissent dans les bibliothèques de modèles d'environnements de simulation (Matlab/Simulink, Portunus, Simplorer, Saber, Amesim, etc.). L'utilisateur accède à un élément graphique qu'il dépose sur une feuille et qu'il va connecter à d'autres éléments au travers de ses ports. Une aide permet de comprendre les hypothèses et le paramétrage du modèle, mais ses détails d'implémentation sont inaccessibles.

Enfin, un concept intermédiaire, ou un compromis entre ces deux approches, peut aussi exister, appelé «**boite grise**». Il s'agit de la description sous forme de code informatique du modèle dans lequel la représentation n'est que partiellement accessible et modifiable pour un concepteur non averti. Nous pouvons considérer ce concept comme un moyen de capitalisation acceptable dans certains contextes.

La Figure 1-21 détaille ces différentes représentations de modèles.

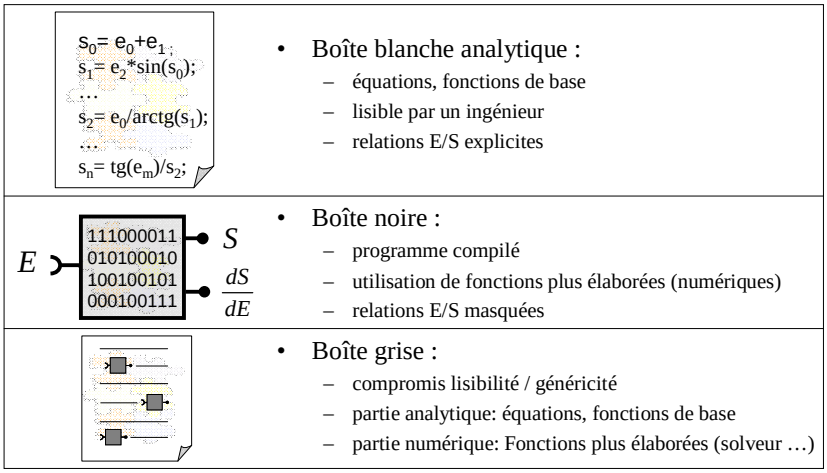


Figure 1-21 : Différentes modélisations : boîtes blanche, noire et grise [DEL-12]

Tous ces aspects méthodologiques exigent une forte base commune de formalismes et de langages de description, et des outils à disposition pour garantir un travail collaboratif d'une façon efficace, rapide et constructive, et ce pour chaque niveau d'abstraction du PVF. Plusieurs langages de modélisation et environnements ont été développés pour supporter les diverses phases. Nous explorons maintenant les langages ainsi que les outils associés ; susceptibles d'être de bons candidats pour répondre à nos objectifs de modélisation collaborative et de prototypage virtuel pour des systèmes électromagnétiques.

[28] www.scilab.org/

D.3.1 Modélisation par «boules blanches»

D.3.1.1 Choix d'un formalisme support aux boules blanches

La modélisation «boule blanche» est l'approche permettant de décrire un modèle sous forme d'une description formelle et textuelle, représentant les équations mathématiques mises en jeux (équations et fonctions) ou un assemblage de composants qui sont eux-mêmes des boules blanches. Cette appellation ne présuppose pas de notion d'entrées/sorties sur les grandeurs mises en relation, sauf cas particulier d'un calcul séquentiel explicite à priori.

Le passage de la notion boule (modèle acausal) à la notion boîte (modèle causal) permet de spécifier, en fonction des objectifs et du contexte environnemental du modèle, si ces grandeurs sont des entrées ou sorties. On a aussi des grandeurs qui sont systématiquement échangées sans notion d'orientation, par exemple celles au niveau des ports énergétiques (effort/flux au sens des graphes de lien ou VHDL-AMS).

Cette modélisation descriptive est en train de connaître un grand succès et elle s'appuie sur des langages dédiés à la modélisation multi-domaines. Ces langages permettent non seulement de développer des modèles de systèmes électriques mixtes (analogique et numérique), mais aussi des systèmes multi-physiques (électriques, mécaniques, magnétiques, thermiques, etc.) par l'utilisation d'une description textuelle. Parmi ces langages les plus utilisés, nous pouvons citer VHDL-AMS[29], Verilog-AMS[30], Modelica[31], SystemC-AMS, Simscape[32] et MAST. L'avantage de ces langages est que l'écriture du modèle est directe et la difficulté de sa résolution est transparente pour le modélisateur : elle est théoriquement assurée par tout simulateur acceptant ce langage. À part les langages MAST et Simscape qui sont propriétaires, les autres langages sont issus de consortium garantissant une certaine pérennisation du standard.

Depuis sa normalisation en 1999, le langage VHDL-AMS s'est clairement positionné dans le secteur de la modélisation multi-domaine. C'est principalement pour cette raison que les partenaires du projet MoCoSyMec ont choisi ce langage. De plus, dans le domaine du génie électrique, VHDL-AMS est très prôné par les acteurs de la microélectronique, il trouve naturellement son utilisation en électronique de puissance et pour les microsystèmes magnétiques et piézo-électriques.

Dans l'annexe 1-2, nous présentons les différents langages de modélisation, précédemment définis, en nous basant sur les travaux de [GUI-07] [ZHO-07] [IBR-09]. Nous en proposons une synthèse enrichie. Il en ressort que les mécanismes de modélisation associés à la norme VHDL-AMS apparaissent bien adaptés pour résoudre la problématique de la conception des systèmes électromagnétiques, que nous présenterons brièvement maintenant.

[29] Very high speed integrated circuits Hardware Description Language – Analog- Mixed Signal http://www.eda.org/vhdl-ams/
[30] http://www.eda.org/verilog-ams/
[31] https://modelica.org/
[32] http://www.mathworks.fr/products/simscape/

D.3.1.2 Le langage VHDL-AMS

Le langage VHDL-AMS est une extension du langage VHDL[33] spécifique à la description du système à temps discret (circuits numériques), aux systèmes à temps continu (systèmes analogiques) et mixtes (discret et continu). Ce langage a été spécialement conçu pour modéliser les systèmes multi-domaines [HER-02] [DES-04] et il a été normalisé sous la référence IEEE 1076.1 en 1999. Cette norme assure une description claire et structurée d'un système dynamique complexe, permettant une haute modularité et supportant la généricité. Il présente pour le concepteur les avantages suivants [ASH-02] :

- c'est un **langage normalisé :** VHDL-AMS est non propriétaire et indépendant des fournisseurs de logiciels. Il permet l'échange de données/modèles de manière standardisée entre les différents acteurs d'un projet,
- il propose une **approche multidisciplinaire :** Son approche multi-domaine native permet une communication aisée entre les différents domaines scientifiques ; un électronicien, un mécanicien ou même un chimiste peuvent dès lors modéliser la partie d'un dispositif qui le concerne directement sans problème de dialogue avec les parties,
- il supporte une **modélisation multi-niveau et multi-abstraction** : VHDL-AMS propose des mécanismes permettant de gérer aussi bien les abstractions comportementales (fonctions réalisées par le système qui est modélisé et non sa physique), que les abstractions structurelles (le système est divisé en sous-ensembles qui peuvent eux-mêmes être modélisés au moyen de différentes abstractions, …) ou bien de type workflow (enchaînement de blocs fonctionnels dont les entrées n'ont pas d'influence sur les sorties des blocs précédents). Les modèles créés avec VHDL-AMS peuvent donc aussi bien être descriptifs que prédictifs,
- il permet la **simulation mixte**. Il autorise de manière conjointe aussi bien la modélisation à temps continue qu'à évènements discrets ou mélangeant les deux. VHDL-AMS prévoit en outre la gestion des discontinuités et des conditions initiales, ainsi qu'une synchronisation explicite des noyaux temps continu et discret.
- **la modélisation est non orientée**. Les équations n'ont pas forcément l'inconnu dans le membre de gauche. À travers celles-ci, VHDL-AMS supporte l'utilisation des lois de Kirchhoff généralisées, fondement des expressions implicites entre les différents nœuds d'un système. L'écriture des équations devient donc naturelle pour le concepteur.

De part ces propriétés, le langage VHDL-AMS s'avère parfaitement adapté à la problématique de notre thèse et répond aux exigences de notre méthodologie de conception système multidisciplinaires. Il offre un support de base pour la modélisation de systèmes électromagnétiques comme les capteurs, les actionneurs et les microsystèmes. Un de ses principaux inconvénients est que pour l'instant le langage VHDL-AMS ne propose pas la description des systèmes sous forme d'équations aux dérives partielles (pouvant être par exemple nécessaires pour la modélisation physique de nos composants magnétiques). Il existe cependant de nombreuses recherches abordant ce problème et proposant des solutions. Par

[33] vhdl.org

exemple, Nikitin proposent d'appliquer une discrétisation spatiale pour contourner cette limite [NIK-07]. C'est aussi le cas pour les intégrations spatiales (non temporelles).

Dans le chapitre 2, nous présenterons plus en détails certains aspects de ce langage.

D.3.1.3 Environnement de simulation supportant le langage VHDL-AMS

Étant donné que l'émergence du langage VHDL-AMS est relativement récente, les outils simulateurs qui le supportent ne sont pas nombreux. Nous pouvons citer : hAMSter [34] (environnement autonome open-source), SMASH™[35] développé par Dolphin intégration ; AdvanceMS™[36] développé par Mentor Graphics, Saber™[37] de chez Synopsys, SIMPLORER™[38] développé par ANSOFT, et Portunus[39] développé par Adapted Solutions en lien avec Cedrat. Actuellement, aucun d'entre eux n'implémente complètement la norme, mais certains s'en approchent plus que d'autres. Les critères importants des concepteurs sont en effet pour ces outils : la couverture de la norme, une interface conviviale, une gestion efficace des bibliothèques de modèles, des simulations performantes en termes de précision et de temps de mise en œuvre rapide des modèles.

Une étude comparative entre différents outils a été présentée dans [HAM-05]. Cette étude, bien que datant de 2005, reste une référence pour les caractéristiques des différents outils. Les conclusions de cette étude que nous avons actualisées en nous positionnant sur un état de l'art datant du début de nos travaux (2009), sont présentées dans un tableau dans l'annexe 1-3.

Par ailleurs, durant le développement de nos modèles, trois simulateurs parmi ceux cités ont été utilisés : Portunus™, SMASH™ et Simplorer™. Une étude comparative des trois outils est résumée aussi dans l'annexe 1-3. Portunus, logiciel de modélisation des systèmes mécatroniques, a été choisi par le projet MoCoSyMec en tant qu'outil de développement de base et il a subi énormément d'amélioration parallèlement à nos travaux de. Néanmoins, nous avons travaillé conjointement sur les trois outils, profitant aux mieux de leurs spécificités : SMASH grâce à sa couverture de la norme, et Simplorer pour son assistant de « débuguage » de modèles et pour son interface graphique.

D.3.2 Modélisation par «boîtes noires»

Contrairement à la précédente, la modélisation «boîte noire» est une approche permettant de caractériser un modèle uniquement par ses entrées et ses sorties. Les relations entre elles sont masquées pour l'utilisateur. Un modèle boîte noire sera vu de la manière suivante (Figure 1-22), en décrivant : ses services requis, ses services fournis, ses paramètres, ses entrées et sorties. En ingénierie, l'implémentation d'une telle approche est réalisée grâce à un tableau de correspondances entrées/sorties (lookup table) ou un code informatique compilé. Ceux-ci sont soit propres à des outils, par exemple Simulink, Simplorer, Portunus ou Amesim ; soit ils sont

[34] ftp://ftp.eseo.fr/pub/perdriau/hAMSter/hamster.pdf
[35] http://www.dolphin.fr/medal/products/smash/smash_overview.php
[36] http://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/advance-ms/
[37] http://sabersolutions.com/
[38] http://www.ansys.com/Products/Simulation+Technology/Electromagnetics/Electromechanical+Design/ANSYS+Simplorer
[39] http://www.adapted-solutions.com/Web/AdaptedSolutionsEnglish/ASProductPortunus.html

portables, et dans ce cas on parle plutôt de composants logiciels. C'est sur ce dernier que nous allons nous concentrer.

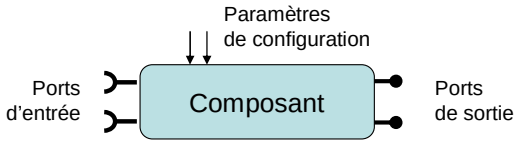


Figure 1-22 : Modèle « boîte noire »

Le composant logiciel est une notion que Szyperski définit comme "*une unité de composition avec des interfaces définies et des dépendances dans un contexte explicite seulement*" [SZY-98]. C'est une entité autonome de déploiement, encapsulant des codes informatiques et décrite par des interfaces de communication.

Le but de cette approche «boîte noire» est de fournir une solution prête à l'emploi sans que l'utilisateur puisse intervenir dans sa constitution, pour des raisons de protection intellectuelle, de compacité de l'outil et de robustesse. La définition d'une norme de composants logiciels, qui spécifie les interfaces et le format des informations échangées, permet d'avoir une architecture commune pour tout modèle, quel que soit sa nature physique. Elle facilite ainsi la connexion des modèles et leurs exploitations par les outils. Cela apporte un intérêt important à l'approche «boîte noire» : la possibilité de composer des modèles indépendamment de leur constitution interne grâce à la normalisation des interfaces de communication [DEL-04] [FIS-04].

La vision fonctionnelle d'un composant informatique est plus riche que celle des langages de modélisation descriptive actuels [DEL-11]. En effet, cette vision peut se décliner pour des modèles de simulation, des modèles de dimensionnement, des algorithmes d'intégration d'équations différentielles, des algorithmes d'optimisation, etc.

D.3.2.1 Les composants logiciels en industrie et en recherche

Beaucoup d'efforts ont été accomplis pour améliorer la modularité des outils afin de permettre l'utilisation des boîtes noires tels que :

- MATLAB/SIMULINK avec les S-Functions pour la simulation temporelle,
- SABER, SIMPLORER, P-SPICE, AMESIM et PORTUNUS avec leurs propres composants prédéfinis dans des bibliothèques ou le développement de nouveaux modèles avec les langages fortran/C, ensuite compilé dans des bibliothèques (DLL[40]).

Cependant, il y a un manque de standardisation des interfaces de programmation. Pour cette raison, l'industrie et la communauté informatique ont proposé un ensemble de normes techniques comme COM de Microsoft, CORBA de l'OMG et JavaBeans de Sun. D'autres communautés spécialistes, suite à leurs propres besoins spécifiques, ont proposé d'autres normes de composants comme AUTOSAR[41] (AUTomotive Open System Architecture). Ce dernier projet est né en 2003 suite à un partenariat solide entre de nombreux constructeurs automobiles. Il avait pour but de développer et d'établir une architecture logicielle

[40] DLL : Dynamic Link Library
[41] www.autosar.org/

standardisée et ouverte pour les véhicules. Ensuite, un projet européen MODELISAR[42] a débuté en 2009 visant à améliorer la conception des systèmes et des logiciels embarqués pour les véhicules. À son issue, une nouvelle norme de composant a été aussi spécifiée en 2010 : FMI (Functional Mock-up Interface) [FMI] [ITEA2]. Cette norme formalise des modèles de simulation dynamique en s'inspirant des interfaces de la S-Function de Matlab/Simulink. Ces composants peuvent être générés à partir de Modelica (Dymola, JModelica) par exemple et produisent un code spécifique en langage C.

Au sein du G2Elab, cette approche a aussi été longuement étudiée pour différents domaines applicatifs (électromagnétisme, circuits électroniques, microsystèmes, etc.). Il en résulte, depuis 2004, une norme nommée ICAr (Interface for Component ARchitecture) élaborée pendant les travaux de V. Fischer [FIS-04]. Elle vise à prendre en compte la complexité des modélisations et à formaliser un ensemble de services métiers. Ces services permettent de définir des interfaces d'échange spécifiques dérivant des équations algébriques ou des algorithmes numériques pour le dimensionnement et des équations différentielles ordinaires (ODE) pour la simulation dynamique. Récemment, le projet européen PLUMES a été initié en 2011 par le laboratoire G2Elab, visant à proposer une nouvelle extension multi-métiers et multi-physiques (introduction de la notion des ports physiques énergétiques) norme appelée MUSE (Modèle Unifiée pour les Systèmes Énergétiques des bâtiments)[43].

Nous avons retenu la norme ICAr des composants logiciels pour les raisons suivantes :

- ICAr représente une norme évolutive et mature bénéficiant d'une architecture ouverte. De nombreux travaux ont permis de faire évoluer et valider différentes fonctionnalités de ces composants et les outils dédiées à nos domaines d'applications visés les supportant (électromagnétisme, microsystèmes) [ALL-03], [MAG-05], [DUP-06], [RAK-07], [ENC-09], [DO-10] et [PHAM-11].
- la norme MUSE est en cours de développement. Elle a été initiée en même temps que nos travaux, pour des applicatifs dédiés aux bâtiments. Comme elle s'inscrit dans la continuité d'évolution de la norme ICAr, toute extension de cette norme s'intégrera dans la norme MUSE. Elle bénéficie à ce titre des résultats issus de nos travaux.
- La norme FMI définit aussi des interfaces de simulation dynamique basées sur les équations différentielles ordinaires (ODE). Elle ne supporte pas encore les descriptions nécessaires à l'optimisation [FMI-10].

Dans la suite, nous ne considérons que les composants logiciels munis de la norme ICAr, qu'on appellera composant ICAr.

D.3.2.2 Composants logiciels ICAr

Un composant ICAr contient un modèle de type «boîte noire» spécifique, qui possède les caractéristiques suivantes. Il est défini par des entrées/sorties et des services, déployés de façon normalisée. Il est autonome, programmé sous forme d'un code Java exécutable, accessible par des interfaces définissant des services et combinable à d'autres composants ICAr. L'utilisation du langage Java, qui est indépendante d'un système d'exploitation, offre au

[42] www.modelisar.com/modelisar.html
[43] Projet ANR-PLUME : « Unified Modelling Platform for Building and Systems Energy Efficiency »

composant plus de portabilité. Il est «pluggable» dans différents environnements, tels qu'Excel, Matlab, AMESim[44], iSight[45], FGot[46]. Un composant logiciel ICAr, pour des besoins d'interopérabilité, peut encapsuler les codes natifs (e.g. issu de langage C ou Fortran).

Un composant ICAr est généralement associé à un composant physique réel. Il peut proposer plusieurs services, par exemple un modèle de calcul (équations algébriques), un modèle de simulation (équations différentielles ordinaires ODE), une visualisation graphique, etc. Un composant ICAr peut également proposer différents modèles physiques (thermiques, électriques, magnétiques…). Il est aussi possible d'avoir plusieurs versions du modèle pour des besoins différents (pilotage temps réel, …). L'architecture du composant ICAr étant évolutive, il est donc possible de lui ajouter de nouvelles fonctionnalités.

La norme ICAr définit donc une interface de programmation unique et identique pour tous les composants, appelée «Component». Cette interface représente le point d'entrée du composant, dont le chargement dans un logiciel peut être effectué indépendamment de sa nature. Une fois celui-ci chargé, cette interface permet d'accéder aux ports des composants (entrées/sorties). C'est aussi grâce à elle que l'on peut retrouver la liste des différents services qu'il offre, sous forme de facettes (Figure 1-23). La notion de facette joue un rôle primordial dans le concept des composants ICAr [ENC-09]. En effet, chaque facette correspond à une action particulière, par exemple le calcul des dérivées des sorties par rapport aux entrées d'un modèle de dimensionnement. Grâce à toutes ces propriétés, la norme ICAr permet une exploitation identique des modèles, quel que soit leur nature physique, par les logiciels implémentant la norme.

On notera pour le dimensionnement, les services de calcul des sorties en fonction des entrées et le calcul du jacobien du modèle.

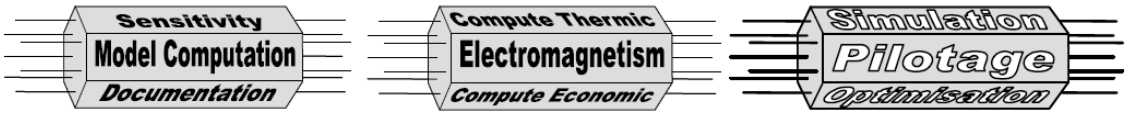


Figure 1-23 : Approche de composant multi-facettes

D.3.2.3 Environnement logiciel CADES

Une suite logicielle exploitant les composants ICAr a été réalisée au sein de l'équipe MAGE, dénommée CADES[47] (Component Architecture for Design of Engineering System) [DEL-12]. Aujourd'hui, CADES correspond plutôt à une architecture permettant la génération et le traitement automatique, via des générateurs spécifiques, de modèles pouvant être décrits sous une forme analytique et/ou algorithmique ou dans une forme plus «métier». Cette plateforme propose par exemple une description géométrique ou une modélisation par un circuit électrique équivalent, etc. Les premiers développements ont été réalisés dans le contexte du dimensionnement par optimisation, en offrant notamment le calcul automatique

[44] http ://www.lmsintl.com/amesim-platform
[45] http ://www.3ds.com/products/simulia/portfolio/isight-simulia-execution-engine/overview/
[46] Fgot : Feature - Genuine Optimization Tool http ://forge-mage.g2elab.grenoble-inp.fr/project/got/upload/releases/got_01_presentation_fr.pdf
[47] http ://vesta-system.cades-solutions.com/

des gradients du modèle et facilitant largement les procédures d'optimisation sous contraintes et l'analyse de sensibilité.

La plateforme CADES (Figure 1-24) est définie par :

- **des générateurs métiers** de modèles sous forme d'un composant ICAr issus d'une description métier du dispositif, tels que : CADESGenerator (modèle analytique écrit en langage SML (System Modelling Langage) [ENC-09], RelucTool (réseau de réluctances) [DUP-06], MacMMems (MEMS magnétiques) [RAK-07] et ECM (circuits électriques passifs) [DUR-10], Fgot (modèles réduits issus de simulation numériques) [GOT-06]
- des **outils associés aux services** : calcul, optimisation, simulation et analyse.

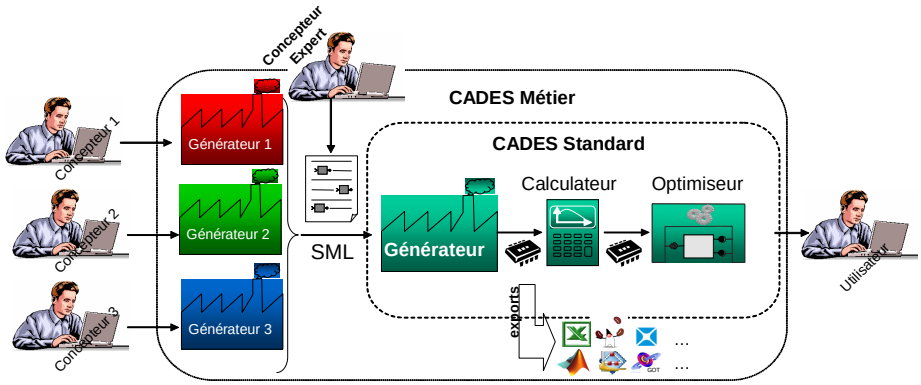


Figure 1-24 : Architecture modulaire de CADES [DEL-12]

D.4. L'environnement d'interopérabilité que nous proposons

La complexité des dispositifs multi-physiques à concevoir nécessite de créer une synergie à l'intérieur de cette diversité d'outils d'aide à la conception, qui souffrent d'une nécessaire formalisation de leurs modèles assurant leur interopérabilité. Wegner définit l'interopérabilité comme *« l'aptitude de deux systèmes ou plus, à communiquer, coopérer et échanger des données et services ; et ce malgré les différences dans les langages, les implémentations et les environnements d'exécution ou les modèles d'abstraction »* [WEG-96]. Konstantas réadapte cette définition en ajoutant un point crucial sur l'impact de ce concept sur les utilisateurs. Il la définit comme étant *« l'aptitude de systèmes à pouvoir travailler ensemble sans effort particulier pour les utilisateurs de ces systèmes»* [KON-05].

Sur les aspects techniques, plus d'interopérabilité dans les outils ou modèles signifie :

- plus de granularité dans les divers composants ou dans les systèmes eux-mêmes,
- possibilité de partager (mettre en commun) des composants ou des systèmes,
- interchangeabilité et réutilisabilité de modèles (plug and play),
- meilleure intégration/adaptation de nouveaux outils,
- possibilité d'évaluer les nouveaux concepts dans des environnements de référence.

En nous focalisant sur les aspects «multi-domaines», «multi-échelles» et «multi-outils» de l'activité de conception des systèmes électromagnétiques, nos travaux de thèse visent à offrir au concepteur un support pour gérer d'interopérabilité, dans lequel il pourra intuitivement mettre en œuvre les modèles et manipuler les systèmes, les outils et des données dont il a

besoin au cours des processus de conception qu'il réalisera. En pratique, il n'existe pas à proprement parler d'environnement d'interopérabilité spécifiques à la conception. Cependant, nous allons décrire ce qu'il peut être, en considérant la méthodologie de « Prototypage Virtuel Fonctionnel » comme support au cycle de conception et en nous appuyant uniquement sur les aspects de modélisation et de simulation dynamique.

Nous nous positionnerons sur une architecture multi-outils qui chapeaute plusieurs outils. Cette formalisation offre la possibilité de développer des supports informatiques, qui fiabilisent et accélèrent ces processus de conception. L'idée principale est l'unification par l'utilisation de services réalisés par d'autres outils, selon [DEL-04]. Cela permet d'exploiter des capacités existantes et de s'adapter tout en produisant une valeur ajoutée. Plus particulièrement, cela offre au concepteur et à l'utilisateur de grandes possibilités d'intégration et de mises en œuvre de leurs processus, en exploitant :

- d'une part, des outils et des méthodes de modélisation métiers spécifiques aux systèmes électromagnétiques, dont nous souhaitons profiter de leurs atouts.
- d'autre part, des outils plus génériques pour la mécatronique offrant des services de simulations et/ou de dimensionnement. Dans le projet MoCoSyMec, Portunus est le logiciel cible pour la simulation dynamique et pour supporter VHDL-AMS, bien que d'autres logiciels de ce type soient utilisés en raison de leur maturité sur VHDL-AMS par rapport à Portunus (SMASH et Simplorer) au début du projet.

Compte tenu du fait que les systèmes électromagnétiques ou leurs éléments peuvent être produits différemment et qu'ils répondent à des **besoins** spécifiques, l'idée de cet environnement consiste à définir une base explicite de travail : une **norme** ou un ensemble de normes que l'on devra respecter lors de sa modélisation. Le projet MoCoSyMec propose de s'appuyer sur le standard normalisé VHDL-AMS. Toutefois, il est souvent difficile de mettre sous forme d'équations algébro-différentielles des modèles de microsystèmes magnétiques ou d'actionneurs électromagnétiques de manière modulaire comme nous l'avons montré au début de ce chapitre et que nous détaillerons dans le chapitre 4.

Pour cela, le G2Elab a développé différents moyens d'aide à la modélisation pour ces systèmes, en générant des composants logiciels dédiés respectant la norme ICAr :

- la modélisation éléments finis des dispositifs magnétiques en s'appuyant sur Flux 2D/3D en collaboration avec Cedrat,
- l'identification de modèles de systèmes électromécaniques par plans d'expériences en s'appuyant sur Fgot,
- la création de modèles de systèmes électromagnétiques en s'appuyant sur le formalisme des réseaux de réluctances «Reluctool», pour le pré dimensionnement et la simulation dynamique en collaboration avec Schneider Electric,
- la modélisation magnétique de microsystèmes magnétiques en vue de leur dimensionnement «MacMMems».

Pour contribuer à l'interopérabilité, comme représenté dans la Figure 1-25, on doit donc fournir un support méthodologique et les moyens informatiques associés pour aider à la gestion des modèles dans le cycle de conception pour les macro-systèmes et microsystèmes magnétiques. Nous allons exploiter au mieux les deux formalismes VHDL-AMS et ICAr, en

les associant si besoin et en les enrichissant. Pour présenter notre contribution à cet environnement d'interopérabilité, nous allons agir successivement sur chacun de ces aspects.

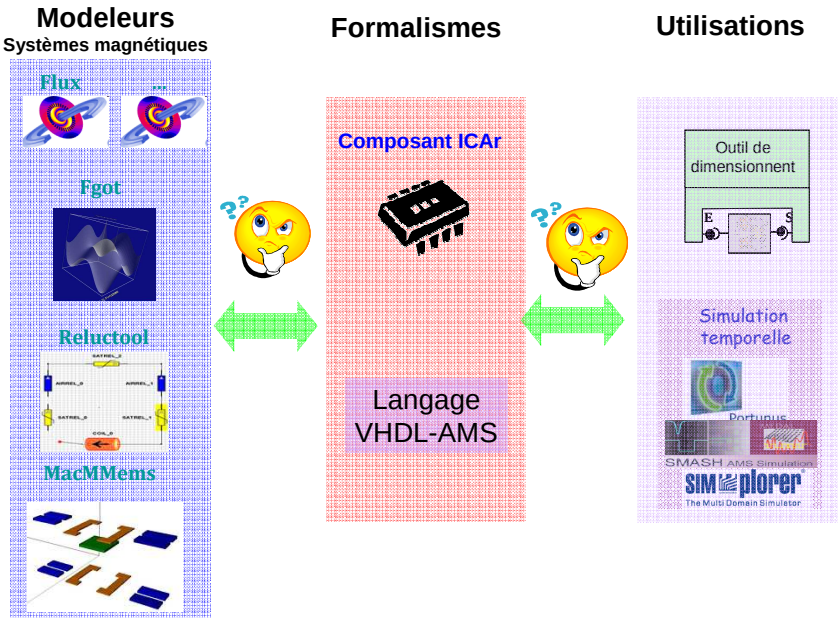


Figure 1-25 : Schéma de principe résumant la problématique de cette thèse

Dans le chapitre 2, nous commencerons par nous positionner par rapport à la problématique de la simulation système. En particulier, nous présenterons les moyens numériques mis en œuvre pour réaliser les étapes de la modélisation et de simulation dynamique des dispositifs électromagnétiques. Cette problématique sera étudiée tout particulièrement à travers deux aspects : la description des modèles et leurs résolutions temporelles. Nous explorerons la mise en place des moyens numériques pour formaliser les modèles dynamiques et les méthodes de résolution temporelle afin de réaliser des simulations dynamiques. À partir de ce point de départ, nous introduirons les nouvelles spécifications de modélisation permettant d'atteindre nos objectifs de simulation.

Dans le chapitre 3, nous nous intéresserons à la problématique de l'interopérabilité au travers des couplages des modèles et des outils métiers de modélisation magnétiques et les environnements de simulation mécatronique ; en exploitant les deux formalismes choisis. Il s'agira d'étudier et de proposer des solutions logicielles et des modules de génération de modèles adaptés à plusieurs environnements de simulation système. Dans ce chapitre nous proposerons et nous mettrons en œuvre une première approche de couplages reposant sur un paradigme de « Plug-in ».

Dans le chapitre 4, nous nous intéresserons à l'interopérabilité au travers la problématique de projection de modèles vers le langage VHDL-AMS. Nous allons étudier et mettre en œuvre les transformations des modélisations métiers issus de RelucTool et de MacMMems vers VHDL-AMS.

Le dernier chapitre sera consacré, quand à lui, à l'application de la méthodologie PVF vis-à-vis de la problématique de conception des actionneurs électromagnétiques. Il représente l'étape de validation de nos contributions à un environnement d'interopérabilité permettant la portabilité et le portage des modèles entre concepteurs et entre logiciels de calcul (simulation, dimensionnement, etc.).

CHAPITRE 2

Modélisation et simulation des systèmes dynamiques complexes

Introduction

Dans ce chapitre, nous nous intéressons à la problématique de la simulation système ; en particulier, la spécification des moyens numériques mis en œuvre pour réaliser les différentes étapes de la modélisation et la simulation dynamique des systèmes. Cette problématique sera étudiée à travers deux aspects : la description des modèles et leur résolution temporelle.

En première partie, nous explorerons les différentes formes temporelles des modèles, ainsi que les formalismes permettant leur description. Nous en détaillerons deux le langage VHDL-AMS et les composants logiciels ICAr. Nous mettrons en évidence leurs limitations et définirons une extension de la norme ICAr en réponse à certaines limites.

Ensuite, nous nous focaliserons sur les systèmes dynamiques décrits par des équations différentielles algébriques (DAE), en mettant l'accent sur leurs caractéristiques en termes de description et de résolution. Nous exposerons une étude de solveurs dédiés pour la résolution de ce type de système.

La dernière partie sera dédiée à la description de nos développements concernant l'extension de la norme ICAr et son utilisation par des solveurs dédiés.

A. État de l'art sur la simulation des systèmes dynamiques

A.1. Les systèmes dynamiques

Issue de la physique, la théorie des systèmes dynamiques (dont le comportement évolue dans le temps) fournit un cadre formel pour la représentation de systèmes dynamiques. La plupart des modèles se fondent sur les abstractions proposées par cette théorie, qui considère qu'un système peut être spécifié selon fondamentalement deux aspects [LEY-08] :

- son comportement externe : les réactions observables depuis l'extérieur du système.
- sa structure interne : son état et son fonctionnement intrinsèque (sa dynamique).

Certains systèmes dynamiques peuvent également être influencés par des apports extérieurs, qui peuvent représenter soit des troubles incontrôlables (le vent qui affecte le mouvement d'un avion) ou des signaux de contrôle (les commandes du pilote sur le contrôle des moteurs). Certains systèmes dynamiques peuvent aussi avoir des sorties, représentant des quantités qui peuvent être mesurées ou les quantités qui doivent être contrôlées. Les systèmes dynamiques avec entrées/sorties sont largement utilisés aujourd'hui [LEY-08].

Supposons que le système, au plus haut niveau d'abstraction, soit modélisé comme une boîte noire qui possède des entrées, des sorties et une structure interne (voir chapitre1). Le comportement externe est défini par la relation entre ses entrées et sorties. Celles-ci caractérisent la manière dont le système réagi du point de vue de son utilisation. La structure interne, quant à elle, se définit par les aspects suivants :

- son état interne (système d'état) qui est généralement représenté par une ou plusieurs variables appelées variables d'état.
- le mécanisme de changement d'état qui désigne l'évolution des variables d'état.
- le mécanisme de production qui fait référence à la production des sorties en fonction de son état interne.

Généralement, les deux derniers points sont implicitement regroupés lorsqu'on parle de la dynamique du système. Elle concerne les mécanismes d'évolution du système au cours du temps (modèle dynamique), par opposition à l'état du système qui fait référence à la situation dans laquelle le système se trouve à un instant précis dans le temps (modèle statique) ou dans certaines conditions en régime établi.

A.2. La simulation dynamique

Selon R. Shannon [SHA-98], la simulation peut être définie de la façon suivante :

"the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and /or evaluating various strategies for the operation of the system."

La problématique de la simulation est d'étudier un système réel, à partir d'un modèle et grâce à un programme informatique, de manière à comprendre son fonctionnement interne et/ou à prévoir son évolution sous certaines conditions. Il est essentiel que le modèle soit conçu de manière que le comportement du modèle imite la réponse du système réel à des stimuli qui se déroulent au fil du temps en facteur des objectifs à atteindre.

R. Shannon ajoute qu'un processus de simulation est composé de la construction du modèle et de son utilisation pour étudier le système réel. Ainsi, la modélisation associée peut être considérée comme une méthodologie expérimentale et appliquée visant à :

- décrire le comportement d'un système à travers un modèle,
- utiliser ce modèle pour prédire le comportement futur, c'est-à-dire les effets qui seront produits par des changements dans le système ou dans son mode de fonctionnement.

Ceci nous amène naturellement à une autre définition d'un processus de simulation proposée par P. Fishwick [FIS-97], qui donne un aperçu plus global et intuitif :

"Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output."

P. Fishwick illustre ses propos à l'aide du schéma suivant (Figure 2-1), présentant son processus itératif par trois tâches fondamentales et fortement interdépendantes :

- l'élaboration du modèle
- l'exécution du modèle sur ordinateur [48]: calcul / simulation
- l'analyse de l'exécution et des résultats obtenus.

Encore une fois, cette définition fait clairement apparaître l'importance du modèle. Par conséquent, une modélisation adéquate et un environnement de simulation permettent la compréhension du fonctionnement des systèmes multi-physiques complexe.

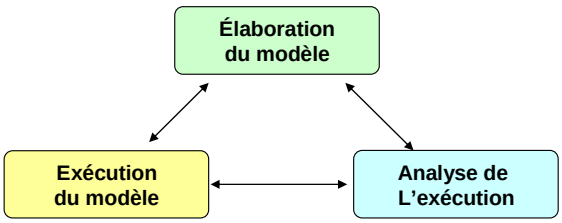


Figure 2-1 : La simulation informatique selon P. Fishwick [FIS-97].

A.3. Conclusion sur cette analyse

Nous ne traiterons ici que les deux premières étapes décrites par P. Fishwick :

- la formulation et la description des modèles (équations dynamiques couplées)
- leur résolution numérique par intégration dans le temps (simulation temporelle),

La Figure 2-2 résume de manière schématique cette distinction.

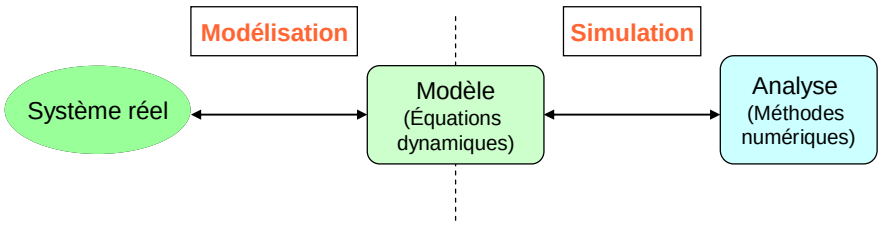


Figure 2-2 : Relation entre modélisation et analyse des systèmes mécatroniques

B. Représentation des systèmes dynamiques hybrides

B.1. Différentes descriptions temporelles

Un système dynamique est avant tout défini par la manière dont il évolue dans le temps, contrairement à un système statique où le temps n'intervient pas. Bien que les modèles puissent être fortement hétérogènes en fonction du type de leur état, différentes représentations de la dynamique du système sont envisageables. Nous pouvons les classer en trois grandes catégories [LEY-08] : les modèles continus, discrets et hybrides. Nous allons maintenant présenter ces trois grandes familles.

[48] Le traitement d'un modèle n'est pas forcément une simulation : (1) Résolution : Calcul qui donnent des résultats (résoudre des équations) ; (2) Simulation : émuler le temps et trouver les valeurs solutions pas à pas

B.1.1 Systèmes continus et hybrides

B.1.1.1 Systèmes continus

Les systèmes à temps continu sont caractérisés par une variation continue des variables d'état du système dans un intervalle fini de temps. La formulation mathématique apte à la représentation de la dynamique continue correspond aux équations différentielles, plus ou moins complexes, incluant les dérivées temporelles des variables d'états. Le formalisme de base associé à ce type de modèles est appelé spécifications de systèmes par équations différentielles (en anglais Differential Equations Specified Systems, DESS) [CEL-91]. Cette formulation permet de décrire plusieurs lois de la physique ; elle est donc naturellement adaptée à la modélisation des phénomènes physiques réels tels que l'évolution de la température d'une pièce ou d'un objet, la vitesse d'un mobile, le niveau dans un réservoir, etc.

On distingue différents types de description de l'évolution temporelle de l'état d'un système basée sur les équations différentielles :

- les équations différentielles ordinaires (ODE) pour une représentation explicites des dérivées des variables d'état des modèles,
- les équations différentielles algébriques (DAE) pour une représentation implicites des modèles,
- les équations aux dérivées partielles (PDE) incluant la dérivation temporelle et spatiale des variables d'états.

- **Les équations différentielles ordinaires (ODE) :**

Les équations différentielles ordinaires (ODE) sont présentes dans la plupart des modèles mathématiques de systèmes dynamiques continus. Un système d'équations ODE contient des fonctions d'une ou plusieurs variables indépendantes et de leurs dérivés temporelles.

Il existe de nombreuses formes générales que peut prendre un système ODE [KRE-72] [SIM-72]. Dans sa forme la plus simple, la structure générale d'un système ODE, explicite et linéaire ou non linéaire est la suivante (en notation vectorielle) :

$$\dot{X}(t) = f(X(t), U(t), p, t) \qquad \text{Eq 2-1}$$

avec X : vecteur des variables d'états
U : vecteur des entrées sources (ou de commande)
p : vecteur des paramètres (généralement invariables au cours du temps)
t : la variable temps.

- **Les équations différentielles algébriques (DAE) :**

Les systèmes continus peuvent être décrits aussi d'une manière implicite plus générale, via les équations différentielles algébriques. Ce type de système regroupe un ensemble d'équations différentielles avec des contraintes algébriques supplémentaires. Ceci est la principale différence avec les équations différentielles ordinaires (ODE), où il n'existe pas de telles contraintes, comme l'a souligné L. Petzold dans son article [PET-83].

Le comportement de nombreux systèmes physiques est décrit par des DAE. Dans le format le plus utilisé, la structure des DAE est présentée sous une forme semi-explicite :

$$\begin{cases} \dot{X}(t) = f(X(t), U(t), P, t) = 0 \\ g(Y(t), X(t), U(t), P, t) = 0 \end{cases} \quad \text{Eq 2-2}$$

avec X : vecteur des variables d'états et $\dot{X}$ son vecteur des dérivées

Y : vecteur des variables algébriques

U : vecteur des entrées sources

P : vecteur des paramètres

t : la variable temps.

D'une manière plus générale, la structure d'un système DAE est totalement implicite linéaire ou non linéaire. Elle est définie comme suit :

$$F(\dot{X}(t), X(t), Y(t), U(t), P, t) = 0 \quad \text{Eq 2-3}$$

Comme nous l'illustrerons dans la suite, l'équation **g** (Eq 2-2) fait apparaître des contraintes algébriques (présente implicitement dans Eq 2-3). Ces contraintes sont définies par une relation directe entre les composantes du vecteur d'états X par exemple, qui rendent la résolution directe de la DAE inconsistante. La résolution requière des traitements spécifiques, notamment par dérivation formelle pour passer à une forme ODE. Ceci sera détaillé dans le paragraphe C.

Notons que d'une manière générale, les ODE sont un sous-ensemble particulier des DAE. En ingénierie, la plupart des problèmes DAE pourraient être considérés comme une ODE implicite (cas des circuits électrique) ou une combinaison d'ODE avec des contraintes algébrique (la forme semi-explicite des DAE).

- **Les équations aux dérivées partielles (PDE) :**

Les modèles basés sur les équations aux dérivées partielles permettent de supporter, en plus des dérivées temporelles des variables d'état, leurs dérivées spatiales. C'est une modélisation plus adaptée aux systèmes ayant un nombre élevé de variables d'état continues. Les phénomènes modélisés par les PDE sont répartis dans l'espace dans une ou plusieurs dimensions (x, y, z par exemple), tels que par exemple la diffusion de la chaleur (eq.4 simplifiée), l'écoulement de fluide, etc. Pour être simulé, un système PDE est souvent discrétisé.

$$\frac{\partial^2 T}{\partial^2 x} - \frac{1}{D} \cdot \frac{\partial T}{\partial t} = f(x, t) \quad \text{Eq 2-4}$$

Où

- T est le champ de température sur le domaine,
- D est le coefficient de diffusivité thermique (en m^2/s).

B.1.1.2 Systèmes d'état hybrides

- **Systèmes discrets et/ou à événements**

Un système discret est un système qui met en jeu des informations qui ne sont prises en compte qu'à des moments précis. En général, ces instants sont espacés d'une durée constante appelée période d'échantillonnage.

Dans un système à évènements, l'état change seulement à certains instants lors de l'occurrence d'évènements particuliers ; l'état est stable entre ces instants (cas des commandes implémentées dans des exécutifs à temps réel [NOR-92]). Il s'agit des systèmes à événements pouvant être modélisés par des modèles à base d'événements (Event based approaches) [ZEI-00]. Il s'agit donc de systèmes dont le comportement est vu comme des transitions possibles entre différents états suite à l'occurrence d'événements. Les automates finis ainsi que les réseaux de Petri et leurs variantes sont des exemples de modèles souvent utilisés pour représenter le comportement de tels systèmes.

- **Systèmes d'états Hybrides**

Les systèmes d'états hybrides (SDH) ne sont ni des systèmes dynamiques continus, ni des systèmes dynamiques à événements. Comme son nom l'indique, les deux aspects continu et discret coexistent dans la modélisation des systèmes hybrides (HYSCOM[49]) [DAV-97].

Un tel système est généralement continu par morceaux, avec des événements permettant de passer d'une équation d'état à l'autre en cours de la simulation. Ainsi, un système d'état hybride peut être défini par :

- un ensemble d'états et de transitions comme dans un système à événements,
- les états du système, dont chacun est modélisé par un système d'état.

B.1.2 Approche causale et acausale

Il existe deux grandes familles d'approches pour la modélisation et la simulation numérique des comportements dynamiques des systèmes : causale et acausale.

B.1.2.1 Approche causale

L'approche causale ou «orientée», au sens de la physique, suit le principe de cause à effet : si un phénomène (nommé cause) produit un autre phénomène (nommé effet), alors l'effet ne peut précéder la cause.

En simulation dynamique, un modèle causal est composé des entrées/sorties et de variables d'états : les entrées introduisent les données provenant de l'environnement, les sorties exportent les données vers l'environnement, et les variables d'états sont utilisées pour calculer des quantités observables (grandeurs physiques mesurables) [FUR-07].La relation entre les entrées et les variables d'états contraignent les valeurs des sorties et les valeurs des dérivées des variables d'états. Le point clé, qui ressort, est que le flux de données est explicite.

A. Jardin complète cette définition, en montrant qu'elle «contraint l'ingénieur à mêler description physique du système et expérimentation (ce pour quoi le modèle a été écrit)»

[49] The Technical Committee on Hybrid Systems (HYSCOM) of IEEE Control System Society (CSS) http://www.dii.unisi.it/~hybrid/ieee/

[JAR-10]. Le concepteur est contraint de formuler un modèle causal pour chaque problème d'ingénierie traité (*e.g.* analyse, dimensionnement, synthèse paramétrique,...). Ceci implique que le modèle ne pourra être utilisé qu'à cette seule et unique fin (ce pourquoi il a été conçu).

B.1.2.2 Approche acausale

Contrairement au modèle causal, le modèle acausal, aussi appelé « non orienté », n'a pas de lien de cause à effet entre l'entrée et la sortie (on peut échanger les causes et les effets en d'autres termes). Selon S. Furic, un modèle acausal est composé de variables et de relations entre ces variables [FUR-07]. Puisque celles-ci sont fonctions du temps et des quantités observables, elles décrivent implicitement les changements à l'intérieur du modèle. Les relations entre les variables agissent ainsi comme des contraintes entre les valeurs à chaque instant. Selon A. Jardin, ce type de modèle est « une description du système totalement indépendante de l'utilisation qui lui est destinée ». Du point de vue mathématique, « il consiste en un ensemble d'équations implicites non ordonnées où les entrées et les sorties du modèle ne sont pas précisées » [JAR-10]. Par conséquent, un modèle acausal est utilisé à plusieurs fins, à condition d'avoir l'outil capable d'ordonner les équations acausales sous la forme d'une série d'affectations compatibles avec les grandeurs connues et recherchées. L'avantage de cette approche est de permettre aux concepteurs de se focaliser sur la description physique des phénomènes impliqués au sein du système, indépendamment de l'utilisation qui sera faite du modèle *a posteriori*.

L'approche acausale garantit la réutilisabilité des modèles en permettant, grâce à la déclaration d'équations non orientées, la séparation entre la description de la physique du système et l'objectif de calcul envisagé.

On peut noter aussi que les méthodes ou approches de modélisation nodales et nodales modifiées MNA (*Modified Nodal Approach)* sont des méthodes respectant l'approche acausale [VAL-94]. Ces méthodes appliquent une formulation implicite des lois de Kirchhoff généralisées conduisant à une représentation algébro-différentielle du circuit. Elles concernent plus particulièrement les circuits électriques et électroniques et les systèmes multi-physiques.

B.2. Formalismes supports de description des systèmes dynamiques

La simulation des systèmes mécatroniques multi-physiques se trouve confrontée à des modèles très hétérogènes au niveau de leurs natures et approches temporelles. Nous allons maintenant voir comment ces diverses représentations sont supportées par les deux formalismes retenues dans nos travaux : le langage VHDL-AMS et le composant logiciel ICAr. Nous ferons une brève description de chacun et des possibilités qu'ils offrent pour la modélisation. Ensuite, nous mettrons l'accent sur leurs apports et leurs limites.

B.2.1 Le langage VHDL-AMS

B.2.1.1 Présentation du langage

VHDL-AMS est un langage de description matériel normalisé (IEEE 1076.1) pour la simulation mixte. Il inclut toutes les propriétés du standard VHDL, avec en plus, la capacité de décrire les systèmes mixtes, analogiques et numériques par le biais de modèles multi-

abstractions, multidisciplinaires et hiérarchiques à temps continu et à événements [ASH-02]. Cette norme assure une description claire et structurée d'un système complexe, permettant une haute modularité et supportant la généricité.

Étant normalisé, VHDL-AMS présente l'avantage de proposer un support commun indépendant des fournisseurs d'outils de simulation dynamique et de la technologie. Cela permet l'échange des modèles de manière simple entre les différents collaborateurs d'un même projet. Il apporte aussi la possibilité de modéliser des systèmes multidisciplinaires grâce à son approche multi-domaines native. Cette souplesse d'emploi permet aux concepteurs (électronicien, mécanicien,..) de modéliser les parties d'un dispositif qui les concerne directement, sans problème d'échange avec les autres parties.

Ce langage permet la simulation mixte en autorisant de manière conjointe la modélisation à temps continu (domaine des systèmes analogiques) ainsi que la prise en compte des variables discrètes et évènements (circuits logiques) ou encore un mélange des deux [HER-02]. VHDL-AMS prévoit la gestion des discontinuités et des nouvelles conditions initiales ; ainsi qu'une synchronisation explicite des noyaux temps continu et temps discret, grâce aux notions d'équations simultanées et processus séquentiels. En terme de transmission de données, VHDL-AMS supporte aussi bien les systèmes conservatifs (réseaux de Kirchhoff, acausal) que les systèmes non conservatifs (flot de données, causal), comme l'illustre la Figure 2-3.

A cette flexibilité d'utilisation s'ajoute la possibilité pour les concepteurs d'aborder leurs modèles à différents niveaux d'abstraction. En effet, VHDL-AMS propose des mécanismes permettant de gérer :

- les abstractions comportementales (modélisation de la fonction du système),
- les abstractions structurelles (le système est divisé en sous-ensembles qui peuvent eux-mêmes être modélisés au moyen de différentes abstractions, …),
- les abstractions flot de données (signal flow) (enchaînement de blocs fonctionnels dont les entrées n'ont pas d'influence sur les sorties des blocs précédents).

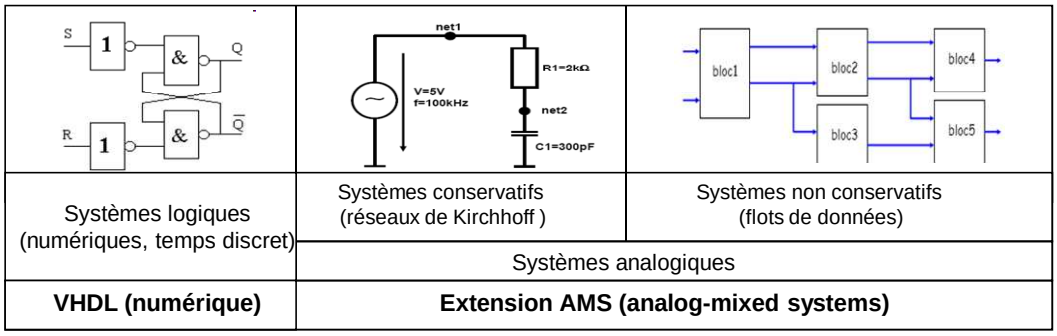


Figure 2-3 Différents modélisations supportées par le VHDL-AMS

VHDL-AMS offre des apports significatifs à la modélisation comportementale. Une caractéristique importante du langage réside dans ses capacités de traitement des équations implicites : un système complexe peut être décrit à l'aide d'un système d'équations différentielles non orientées et linéaires ou non. A travers celles-ci découle l'utilisation des lois de Kirchhoff généralisées, fondement des relations implicites entre les différents nœuds de connexions d'un système [IBR-09] [SNA-04]. Les lois traduisent l'égalité des efforts et la somme des flux égale à zéro à chaque point de connexion des composants modélisés.

B.2.1.2 Domaines analogiques et physiques

Le comportement d'un composant peut être décrit grâce aux équations différentielles, dont la forme dépend de sa nature physique. Cependant, pour les systèmes non électriques, les lois de la physique sont appliquées grâce à des analogies entre les différents domaines. VHDL-AMS est conçu pour gérer le multi-domaine. Pour cela, des terminaux de connexions énergétiques, nommés "*TERMINAL*", ont été définis et ils sont liés à des grandeurs physiques qui respectent implicitement les lois de Kirchhoff généralisées. On définit deux variables globales (voir la Figure 2-4) qui sont le "*flux*" (courant, vitesse, température…) et "l'*effort*" (tension, force,…). Le Tableau 2-1 récapitule les efforts/flux dans divers domaines physiques pour le VHDL-AMS. Les terminaux ne définissent explicitement aucune valeur, mais ils sont associés à des quantités (*QUANTITY*) pour former des DAE. Leur rôle est de faciliter la description des systèmes conservatifs.

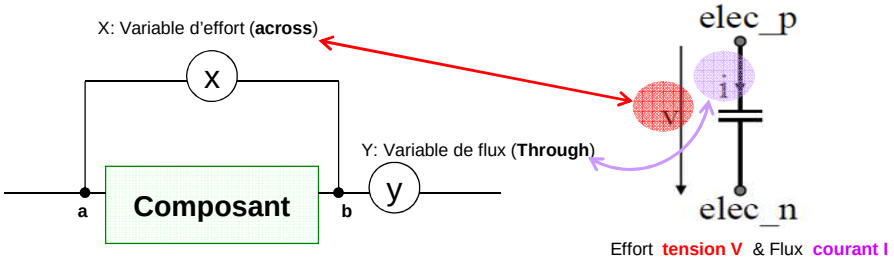


Figure 2-4 : Variables terminales : flux et effort

Domaine	Effort (across)	Flux (through)
Electrique	Tension	Courant
Mécanique (linéaire)	Vitesse	Force
Mécanique (rotation)	Vitesse angulaire	Couple
Magnétique	Force électromotrice (mmf)	Flux magnétique

Tableau 2-1 : Flux et efforts des différents domaines en VHDL-AMS [HAM-93]

Le langage offre aussi des opérateurs, facilitant la mise en équations des modèles, tels que la dérivation temporelle et l'intégration (*dot*, *integ*). Pour résoudre ces équations différentielles, tout simulateur implémentant ce langage tente simplement d'égaliser les deux parties d'une équation à une tolérance près. Une condition nécessaire de solvabilité en VHDL-AMS est d'avoir autant d'inconnues que d'équations.

B.2.1.3 Organisation du modèle VHDL AMS

Un modèle VHDL-AMS est une entité de conception constituée de deux parties : une déclaration d'entité et un ou plusieurs corps d'architecture. La spécification d'entité (ENTITY) correspond à la vue externe du modèle et l'architecture (ARCHITECTURE) est associée à sa vue interne. La Figure 2-5 illustre la structure complète d'un modèle VHDL-AMS.

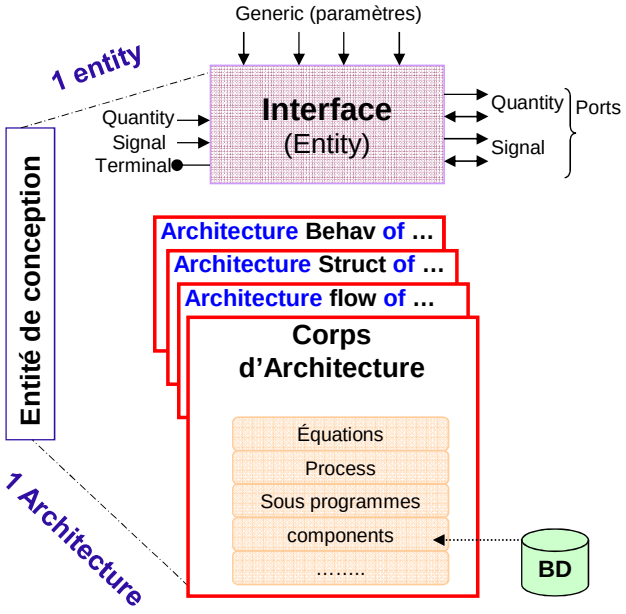


Figure 2-5 : Structure d'un modèle VHDL-AMS

- **Entité d'un modèle VHDL-AMS**

L'entité définit l'interface du modèle avec son monde extérieur (son environnement). Cette entité nécessite en premier lieu l'appel :

- des bibliothèques utiles (*LIBRARY*),
- des spécifications du contenu à exporter (*USE*) pour les fonctions associées à un domaine énergétique particulier par exemple.

L'entité permet de définir les paramètres génériques du modèle (*GENERIC*) et de décrire les ports (*PORT*) par lesquels les données sont échangées avec son environnement.

Les différentes informations échangées sont supportées par des ports de type :

- *SIGNAL* : support de l'information à événement discret / quantité numérique,
- *QUANTITY* : support de l'information analogique orientée,
- *TERMINAL* : support de l'information analogique de type nœuds génériques de connexion des modèles de différentes natures physiques.

- **Architecture d'un modèle VHDL-AMS**

Quant à elle, l'architecture définit le comportement et/ou la structure du modèle. La description structurelle précise l'interconnexion entre les éléments physiques qui peut être réalisée avec le langage de description d'architecture (ADL [MED-00]). La description comportementale est effectuée par des définitions de la fonction réalisée par le système (présentée analytiquement). Ceci s'effectue par l'intermédiaire :

- de l'*instanciation de composants* (définition d'une structure ou d'une Netlist représentant les connexions entre ces composants),
- des *instructions simultanées* décrivant le comportement continu par des équations différentielles ordinaires ODE ou algébriques DAE,
- des *instructions concurrentes* pour la manipulation des informations discrètes.

B.2.1.4 Description comportementale et structurelle

Le langage VHDL-AMS offre aux concepteurs la possibilité d'aborder leurs modèles à différents niveaux d'abstraction : comportemental et/ou structurel. Au niveau comportemental, le système est décrit directement par le biais de diverses affectations et d'équations différentielles algébriques (DAE). Le comportement physique est souvent décrit d'une manière simplifiée. En revanche, au niveau structurel, le système est divisé en composants qui peuvent eux-mêmes être modélisés selon différents niveaux de modélisation (fonctionnel, comportemental, etc..). Cette approche tire parti de la capacité de VHDL-AMS à instancier des composants au niveau système à partir de composants génériques. L'instanciation revient à prendre une copie d'un modèle dans une bibliothèque, la personnaliser, la configurer et l'intégrer dans son contexte applicatif.

Pour mieux connaître la structuration d'un modèle en VHDL-AMS, nous donnons un exemple sur la Figure 2-6, modélisant un circuit RLC. Nous avons défini deux architectures possibles à ce modèle. La première présente une description structurelle par l'assemblage des composants élémentaires via une Netlist, ou on instancie des modèles d'une bibliothèque et on les connecte entre eux. Ainsi, dans ce cas, les modèles de la résistance, de l'inductance et la capacité ont été développés séparément et introduits dans la bibliothèque *"Work"*. La deuxième décrit le comportement du circuit par une équation différentielle temporelle.

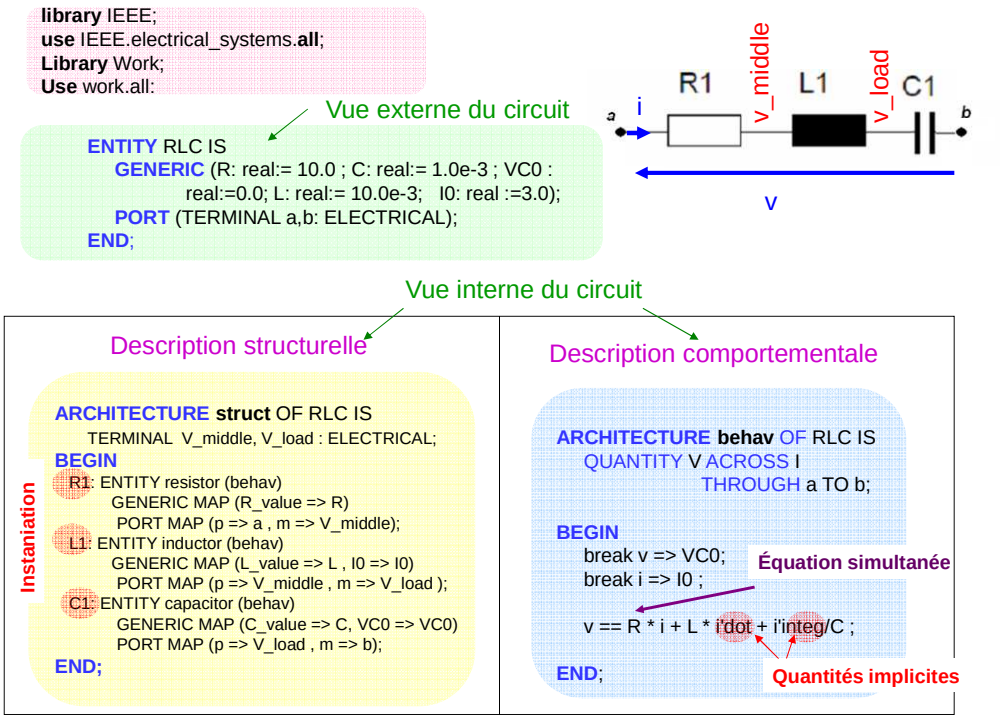


Figure 2-6 : Description VHDL-AMS d'un circuit RLC

B.2.1.5 Limitations de la modélisation VHDL AMS

Bien que VHDL-AMS soit un langage puissant et flexible, il possède des limitations «techniques» et la norme reste assez «discrète» sur certains points. Par exemple, la norme ne définit pas comment les tolérances, caractérisant les précisions des résolutions implicites, doivent être définies. Cet aspect n'étant pas normalisé pour tous les simulateurs, il peut être source des différences sensibles sur les résultats de simulations obtenus [BOU-10].

Même si le standard laisse à l'utilisateur la possibilité de définir ses propres domaines physiques ; il n'offre pas d'alternatives possibles à la sémantique de connexion faisant intervenir les lois de Kirchhoff généralisées. Cela devient un handicap lorsque l'on veut traiter d'autres systèmes de relations physiques.

Actuellement, VHDL-AMS ne peut pas proposer des instructions répondant à tous les besoins des concepteurs de modèles. Seules les dérivations et intégrations temporelles sont acceptées par le langage [IBR-07] [VAC-03] [HER-02]. Les dérivations et les intégrations spatiales ne sont pas prévues dans la norme ; ce qui rend difficile les modélisations géométriques en langage VHDL-AMS par exemple. En fait, la norme a intentionnellement laissé le traitement des équations aux dérivées partielles (PDE) lors du développement du standard en raison de sa complexité [SHI-95]. De même, la description du comportement des microsystèmes magnétiques est généralement effectuée par un ensemble d'équations aux dérivées partielles avec des conditions aux limites liées à la géométrie [BUS-01] ; sa modélisation en VHDL AMS est aussi limitée. Des solutions alternatives existent cependant, mais restent assez lourdes à mettre en œuvre.

L'exploitation des fonctions externes, grâce à l'attribut « *FOREIGN* », permet aux modèles VHDL-AMS d'effectuer des interfaçages avec autres codes et d'autres simulateurs. Nous étudierons cette fonctionnalité dans le chapitre 3 et nous verrons comment elle permet d'élargir les capacités de modélisation de VHDL-AMS. Dans nos travaux, nous avons rencontré quelques-unes de ces limites, essentiellement :

- la sémantique de connexion lors de la modélisation des MEMS magnétiques.
- la modélisation géométrique des systèmes,
- la dérivation et l'intégration spatiales.

Ces limitations seront développées plus en détails dans les chapitres 3 et 4 et des solutions pour y remédier seront proposées

Une autre limitation importante concerne les outils. Actuellement, aucun d'entre eux n'implémente complètement la norme, mais chacun s'en approche plus ou moins. Cependant, comme les concepteurs d'outils ne réalisent pas des implémentations totalement conformes à la norme ; certaines instructions ne correspondent pas à ce qui était prévu. Ceci engendre des problèmes potentiels sur la portabilité des modèles, un modèle fonctionnant dans un outil peut donner d'autres résultats dans un autre outil par exemple [BOU-10] [SUN-10].

Enfin, les simulateurs actuels reposent sur des extensions et des modifications d'anciens simulateurs. Cela implique des limitations dans les possibilités de simulation qui empêchent l'implémentation de certaines instructions du langage. C'est par exemple le cas de l'instruction

PROCEDURAL qui pourrait nous permettre de simuler un bloc d'instructions séquentielles à temps continu à chaque pas de calcul [SNA-05].

B.2.1.6 Synthèse par rapport à la description de modèles en VHDL AMS

Nous venons de présenter et justifier les raisons pour lesquelles nous pensons que le langage VHDL-AMS permet aux concepteurs de gérer des systèmes complexes et multidisciplinaires. Nous avons testé les aptitudes de ce langage dans le cadre de la modélisation d'un système électromagnétique complet à travers l'exemple du déclencheur électromagnétique (chapitre 1) et détaillé dans l'annexe 2-1.

Cette étude nous a permis de constater qu'une fois que le modèle analytique du dispositif est élaboré, la traduction en langage VHDL-AMS et sa simulation s'effectuent rapidement. Cependant, le modèle magnétique a été réalisé de manière analytique, spécifiquement pour cette application, constituant un travail non négligeable [PER-98]. Dans le cadre d'un outil de modélisation métier produisant un modèle VHDL-AMS à partir d'une description topologique du dispositif, nous montrerons au chapitre 4 que nous pouvons rapidement atteindre les limites du langage. Enfin, pour pouvoir exploiter des modèles plus fins ou préexistants dans d'autres outils, nous montrerons dans le chapitre 3, comment il est possible de les intégrer en VHDL-AMS, en re-exploitant l'exemple du déclencheur pour illustrer l'approche.

B.2.2 Les Composants ICAr

B.2.2.1 Composants logiciels ICar

ICAr (Interface for Component ARchitecture) est une norme de composants logiciels développée au G2Elab. Elle est définie initialement pour la conception des systèmes d'ingénierie électrique, dédiée pour le dimensionnement de systèmes depuis une dizaine d'années. Elle a évoluée et a été élargie par la suite pour couvrir d'autres métiers et d'autres domaines grâce au concept « muti-services » (chapitre 1). Elle permet d'accéder à plusieurs services (optimisation, simulation…) à travers les différentes facettes accessibles via son interface normalisée unique « Component » (Figure 2-7). Cette interface représente un moyen pour instancier un composant, accéder à ses ports et interagir avec ses différents services ou « facettes ».

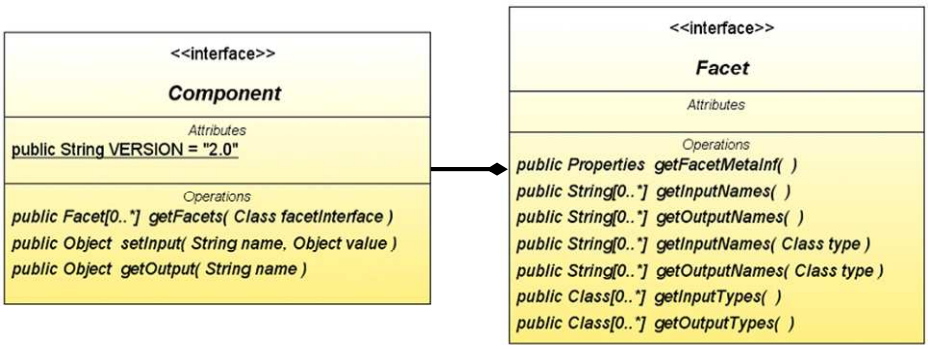


Figure 2-7 : Les interfaces d'un composant ICAr

La notion de facette joue un rôle primordial dans le concept des composants ICAr [FIS-04] [ENC-09]. En effet, chaque facette du composant correspond à une action particulière capable de, par exemple :

- fournir des sorties à partir des entrées
- calculer la sensibilité : le jacobien ou les dérivées du vecteur d'état,
- réaliser une simulation temporelle,
- proposer un modèle comportemental ou un modèle précis.

Un composant ICAr représente un modèle de type «boîte noire», défini par ses entrées/sorties. Il est indépendant de la plateforme d'utilisation vu qu'il est sous forme d'un code java exécutable et donc autonome. Il est déployable de façon normalisée grâce à ses interfaces standardisées, ainsi utilisable dans différents environnements. Un composant ICAr est associé à un système physique réel. Il peut proposer plusieurs vues du dispositif, par exemple un modèle de calcul, un modèle de simulation, une visualisation graphique, etc.

Cette norme répond au besoin du dimensionnement par optimisation et de la simulation dynamique hybride adoptant l'approche causale présentée précédemment [DO-10].

B.2.2.2 Les modèles de calcul ou de dimensionnement

Classiquement, le problème d'analyse ou de calcul statique est représenté comme un problème direct qui calcule les performances (sorties : forces, vitesse,…) d'un système donné (entrées : géométrie, propriétés physiques,…). A l'inverse, un problème de dimensionnement revient à résoudre le problème inverse qui consiste à définir le système via ses paramètres d'entrées pour atteindre des performances souhaitées, comme le montre la Figure 2-8.

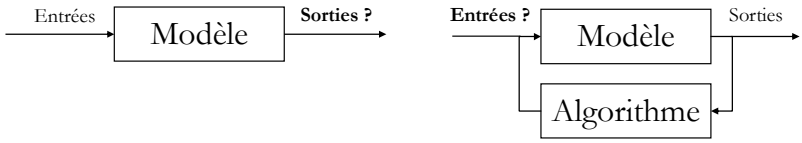


Figure 2-8 : Problème direct et problème inverse

Ce problème peut se formuler comme un problème d'optimisation sous contraintes, comme exprimé par l'équation 2-5

$$\begin{cases} \text{- minimiser} \quad F(X) \quad \text{avec} \quad X = \{x_1, x_2, ...x_n\} \in \Re^n \\ \qquad\qquad\qquad\qquad\quad x_i^{min} \le x_i \le x_i^{max} \quad i = 1,...,n \\ \text{- en reseptant les contraintes} \quad G_j(X) = 0, \quad j = 1,...,m \\ \qquad\qquad\qquad\qquad\quad H_k(X) \le 0, \quad k = 1,...,p \end{cases} \qquad \text{Eq 2-5}$$

La norme ICAr propose deux facettes principales qui héritent de l'interface « Facet » imposant des caractères communs. Une facette de calcul statique (non fonction du temps) et une facette de dimensionnement pour un modèle orienté [FIS-04]. La Figure 2-9 présente les différentes facettes d'un tel composant utilisé pour l'optimisation :

- "ModelSolver" : calcule les variables de sortie selon les entrées.
- "JacobianlSolver" : calcule les dérivées partielles des sorties en fonction des entrées (jacobien).
- "SelectiveJacobianSolver" : calcule les dérivées partielles selon certaines entrées
- "Documentation" : facette complémentaire pour obtenir la documentation du modèle
- "GeometricalView" : représente une vue paramétrée du système à concevoir.

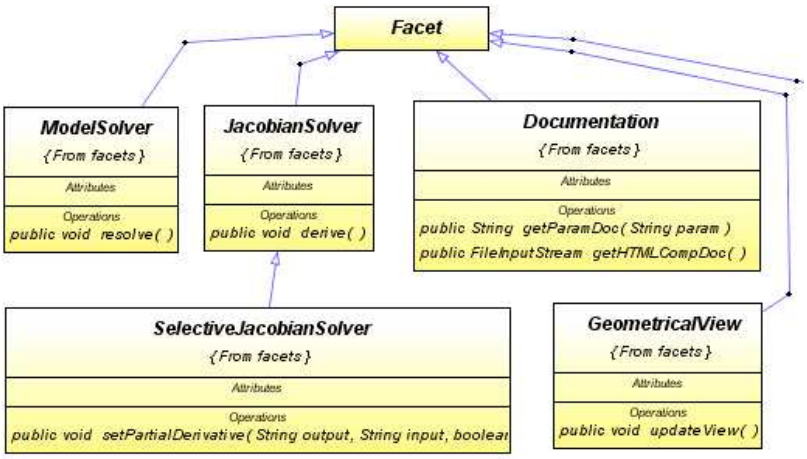


Figure 2-9 : Différentes facettes répondant au problème de dimensionnement

Le calcul exact des gradients (jacobien) est motivé par le fait que ce calcul, lorsqu'il est possible, conduit à une optimisation plus efficace. Pour la résolution d'un modèle de dimensionnement, le couplage de la facette de dimensionnement à l'algorithme d'optimisation sera fonction de la nature des ports (continus / discret) et de la disponibilité d'un Jacobien (Figure 2-10).

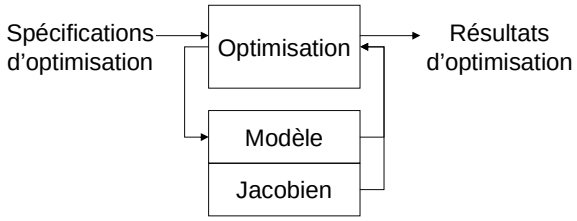


Figure 2-10 : Couplage facette de dimensionnement et algorithme d'optimisation

B.2.2.3 Les modèles de simulation dynamique mixte

La norme ICAr permet de décrire des modèles, pour la simulation dynamique [DO-10] :

- Discret, en permettant de gérer l'intégration en pas de temps fixe,
- continu sous la forme d'un système d'ODE (Ordinary Differential Equations),
- mixte, continu avec événements de type PDI (prochaine date importante) [NOR-92] ou franchissement de seuil (zero-crossing).

Un composant ICAr peut embarquer les modèles ODE d'ordre N (en se ramenant à N équations du premier ordre), explicites, linéaires ou non linéaires et associés à une équation vectorielle algébrique définissent les sorties du modèle en fonction du vecteur d'état, comme le montre la Figure 2-11.

De plus, associés aux équations d'états continues, il est possible de prendre en compte des états discrets ou des événements dans nos modèles pour réaliser des simulations hybrides avec une gestion à pas de temps variable.

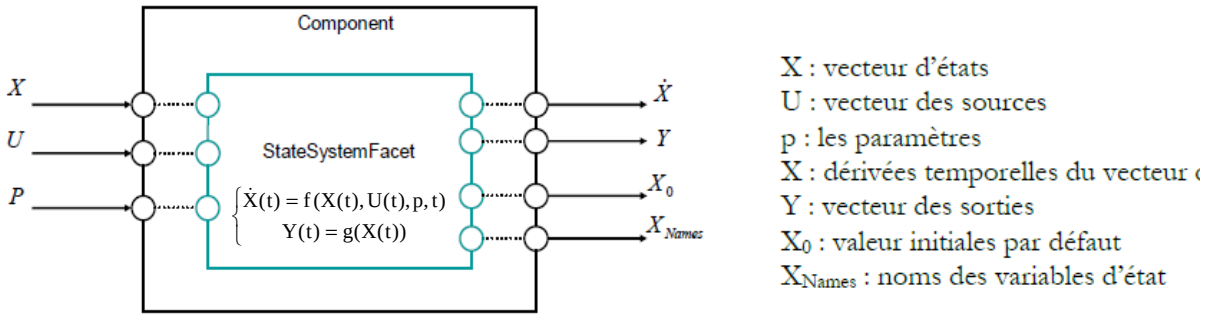


Figure 2-11 : Structure globale d'un composant pour la simulation d'ODE [DO-10]

P. DO-THAI [DO-10] a proposé des interfaces des composants ICAr pour la simulation dynamique hybride, adoptant l'approche causale et le formalisme ODE (Figure 2-12) :

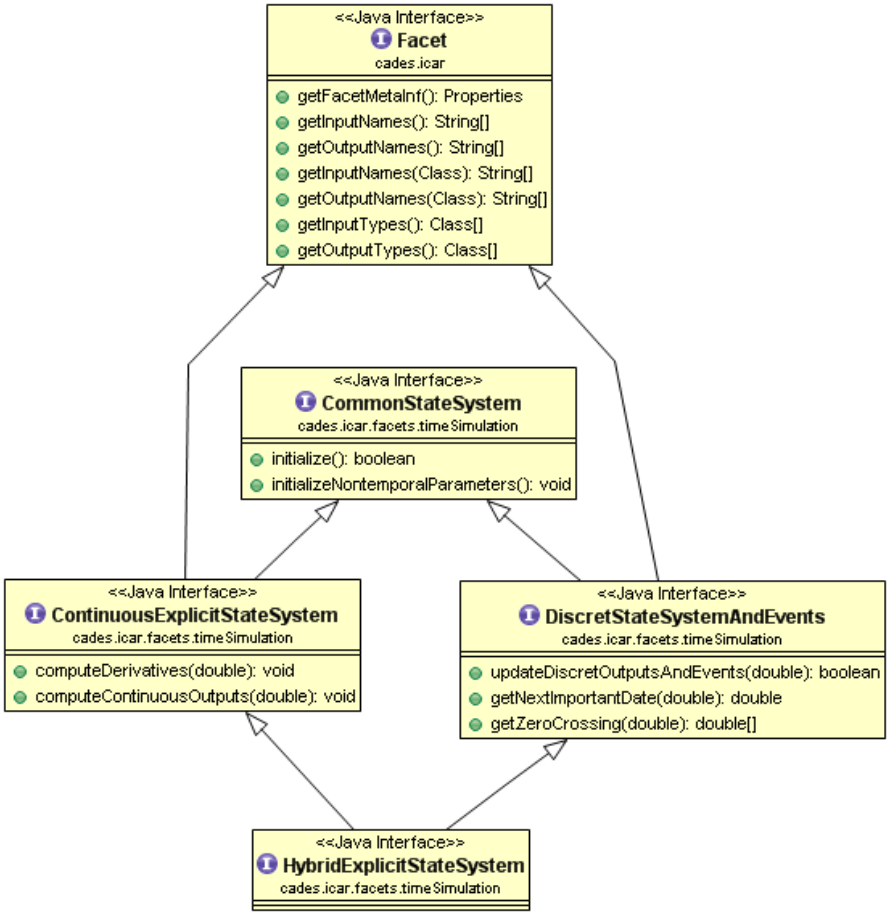


Figure 2-12 : Différentes facettes de la simulation dynamique ICAr [REZ-11]

- « CommonStateSystem » : regroupant les fonctionnalités communes comme l'initialisation,
- « ContinuousExplicitStateSystem » : définissent les phénomènes continus par une ODE,
- « DiscretStateSystemAndEvents » : gérant les états discrets et les événements,
- « HybridExplicitStateSystem » : hybridant le continu et le discret.

Pour la résolution de ces modèles, le couplage avec un algorithme d'intégration temporelle est nécessaire. Associés à ce type de modèle, des composants appelés «solveur» embarquant des algorithmes d'intégration de type Euler, Trapèze et Runge-Kutta ont aussi été mis en

œuvre [DO-10]. L'association modèle/solveur est appelée « composant ICAr de simulation dynamique » apte pour la co-simulation (voir chapitre 3).

B.2.2.4 Synthèse par rapport à la description de modèles en ICAr

Pour résumer, l'approche par composant logiciel offre un certain nombre d'avantages :

- les composants sont facilement distribuables et re-exploitables,
- l'interopérabilité est assurée par la normalisation, permettant ainsi une composition ou un couplage aisé des composants entre eux ou avec d'autres logiciels (chapitre 3),
- l'implémentation de la norme ICAr permet de répondre aux besoins de la capitalisation, notamment en termes de portabilité et de réutilisabilité des modèles.

Nous savons qu'un système multi-physique complexe, en première formulation, est un système implicite acausal, généralement décrit sous forme d'équations différentielles algébriques (DAE). Les composants logiciels ICAr, au début de nos travaux, ne supportant que l'approche causale par description d'ODE, elle s'avère peu compatible et moins naturelle malgré la facilité de cette description. Un processus supplémentaire de transformation de la description DAE vers une description ODE est alors indispensable. Ce processus peut reposer sur des manipulations matricielles, de l'ordonnancement, de la réduction d'index du système par des dérivations symboliques, comme nous le verrons plus loin.

Parallèlement à cette thèse, F. Verdière [VER-12] a mis en œuvre un générateur de modèle ICAr à partir d'une description Modelica[50] en s'appuyant sur la différentiation automatique [QUA-12] et le compilateur Jmodelica[51]. Cette approche exploite la transformation d'un modèle DAE en ODE, mais elle n'est applicable que sur des modèles spécifiques. Afin d'être plus générique, nous avons initié la génération directe des composants ICAr avec des DAE.

L. Petzold a montré que la traduction DAE vers ODE est très compliqué, nécessite un temps important pour un système DAE non linéaires et ne réussit pas dans certains cas. De plus, les équations algébriques sont des contraintes physiques ; les dérivées de ces équations peuvent ne pas être physiquement significatives. Enfin, certaines variables significatives peuvent être éliminées au cours de cette transformation [BRE-96].

Cette restriction de l'approche causale ODE, spécifique pour un objectif précis, représente une limite de l'utilisation des composants logiciels ICAr. L'approche acausale définissant des équations implicites non orientées nous semble intéressante à inclure dans ces composants. Ainsi, il apparaît le besoin d'avoir des interfaces et des services adaptés pour la simulation dynamique des systèmes DAE hybrides.

Dans nos travaux, nous nous sommes intéressés à l'intégration de la possibilité de modéliser le comportement continu ou hybride d'un système physique par des équations différentielles algébriques (DAE). Nous allons donc partir des spécifications des composants ICAr et nous allons chercher à les faire évoluer pour ajouter les fonctionnalités visées. Ainsi, dans la suite du chapitre, nous allons réaliser une simulation temporelle à partir d'une description DAE et envisager des co-simulations (composant DAE avec son solveur).

[50] https ://modelica.org/
[51] http ://www.jmodelica.org/

C. Résolution numérique des systèmes dynamiques mixtes

Nous allons présenter les spécificités des systèmes d'équations différentielles algébriques. Ensuite, nous nous concentrerons sur les méthodes de résolution de ces systèmes ainsi que leurs implémentations dans certains solveurs.

C.1. Caractéristiques des systèmes DAE continus

C.1.1 Formes implicite et semi-explicite

Le comportement continu des systèmes est décrit par des équations différentielles algébriques DAE. Dans sa forme générale totalement implicite, les DAE sont formulées par :

$$F(\dot{X}(t), X(t), Y(t), u(t), p, t) = 0 \qquad \text{Eq 2-6}$$

où

- F système implicite des équations différentielles
- X vecteur des variables d'états et $\dot{X}$ son vecteur des dérivées
- Y variables algébriques,
- u vecteur des entrées sources,
- p vecteur paramètres
- t variable temps.

Il faut noter que le jacobien de F par rapport à $\dot{X}$, noté $\left(\frac{\partial F}{\partial \dot{X}}\right)$, est non singulier [BRE-96].

Le système DAE (ci-dessus) peut être formulé de manière équivalente dans une forme semi-explicite; une combinaison d'un système ODE représenté par la fonction *f* et des contraintes sous forme d'un système d'équations implicites algébriques *g*.

$$\begin{cases} \dot{X}(t) = f(X(t), U(t), P, t) \\ g(Y(t), X(t), U(t), P, t) = 0 \end{cases} \qquad \text{Eq 2- 7}$$

où g est le système des contraintes (équations algébriques)

À noter que ces deux systèmes (Eq.2-6 et.2-7) sont similaires, la seule différence est que les dépendances entre les variables X et ses dérivées $\dot{x}$ sont explicites [BRE-96].

Nous résumons les différents cas de représentation d'une DAE dans le tableau suivant :

Définition	**Expression**
Equations différentielles ordinaires (ODE) explicite	$\dot{x}(t) = f(x(t), u(t), p, t)$
Equations différentielles ordinaires (ODE) implicite	$F(x(t), \dot{x}(t), u(t), p, t) = 0$ **avec** $\left(\frac{\partial F}{\partial \dot{x}}\right)$ non singulière
Equations différentielles ordinaires explicite avec contraintes : DAE semi-explicite	$\begin{cases} \dot{x}(t) = f(x(t), y(t), u(t), p, t) \\ g(x(t), y(t), p, t) = 0 \end{cases}$
DAE générale	$F(\dot{X}(t), X(t), Y(t), u(t), p, t) = 0$ avec $\left(\frac{\partial F}{\partial \dot{x}}\right)$ non singulière

Tableau 2-1 : Bilan sur les différentes représentations d'un système DAE

L'exemple le plus classique de DAE provient de l'étude d'un simple pendule [HAI-89] [KUN-06] [BRE-96]. Nous avons choisi cet exemple pour illustrer les traitements réalisés sur les DAE continus. Nous allons étudier la dynamique du pendule à partir de l'équation du mouvement d'une masse *m* suspendue au bout d'un fil (de masse négligeable et de longueur constante *l*). La masse est écartée de son état d'équilibre puis lâchée, on cherche alors l'équation de ce mouvement (Figure 2-13).

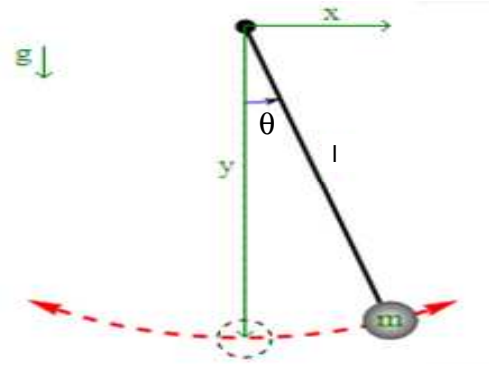


Figure 2-13 : Une pendule simple en mouvement

L'astuce classique utilisée est de ramener le problème à l'étude de l'angle θ que fait le fil avec la verticale ; le problème se ramène à une équation ODE d'ordre 2. Nous choisissons ici une mise en équation lagrangienne dans le repère cartésien. Les équations s'écrivent :

- l'énergie cinétique : $T = \frac{1}{2} m(\dot{x}^2 + \dot{y}^2)$
- l'énergie potentielle : $U = m \cdot g \cdot y$

avec *g* la constante de gravité

En utilisant l'équation suivante : $(x^2 + y^2) - l^2 = 0$, correspondant à la contrainte imposée par la longueur l du fil, nous obtenons le lagrangien L :

$$L = \frac{1}{2} m(\dot{x}^2 + \dot{y}^2) - m \cdot g \cdot y - \lambda.(x^2 + y^2 - l^2)$$

où λ est le paramètre de Lagrange.

Les équations du mouvement sont alors obtenues en dérivant le lagrangien puis en cherchant à l'annuler (principe de moindre action), ce qui donne la fonction suivante :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \left(\frac{\partial L}{\partial q}\right) = 0 \quad \text{avec} \quad q = x, y, \lambda,$$

Dans le cas du pendule, l'équation de Lagrange nous donne le système DAE suivant :

$$\begin{cases} V_x = \dot{x} \\ V_y = \dot{y} \\ m \cdot \dot{V}_x = -2 \cdot x \cdot \lambda. \\ m \cdot \dot{V}_y = -2 \cdot y \cdot \lambda. - m \cdot g \\ x^2 + y^2 = l^2 \end{cases} \qquad \text{Eq 2- 8}$$

où les variables Vx et Vy représentent les vitesses de la masse mobile en x et y.

C.1.2 Index d'une DAE

C.1.2.1 Définition

Les systèmes d'équations DAE sont caractérisés par leur index. La définition la plus courante de l'index d'un système, de type Eq.2.6, est le nombre minimal de manipulations symboliques nécessaires pour transformer tout ou une partie du système *F* en un système différentiel explicite. Il s'agit de déterminer $\dot{X}$ en fonction de *X* et de *t* par des manipulations algébriques [GEA-88], des dérivations temporelles successives par exemple (équation 2-9). Avec cette définition, de toute évidence, un système ODE est d'index zéro.

$$
\begin{aligned}
&F(X,\dot{X},t)=0\\
&\frac{dF(X,\dot{X},t)}{dt}=0\\
&\frac{d^{(2)}F(X,\dot{X},t)}{dt^{(2)}}=0\\
&\vdots\\
&\vdots\\
&\frac{d^{(n)}F(X,\dot{X},t)}{dt^{(n)}}=0
\end{aligned}
$$

Eq 2- 9

Prenons des exemples simples afin d'illustrer la notion d'index d'une DAE. Supposons que q(t) est une fonction donnée, nous envisageons de résoudre le problème trivial suivant pour la variable d'état x (t) :

$$x(t)=q(t)$$

Il est évident que cette équation scalaire est d'index 1, car il faut une dérivation pour obtenir l'équation ODE : $\dot{x}(t)=\dot{q}(t)$

Prenons maintenant le système suivant, dont les variables d'états sont $(x_1(t),x_2(t))$:

$$\begin{cases}x_1(t)=q(t)\\ x_2(t)=\dot{x}_1(t)\end{cases}$$

La première dérivation de l'équation $x_1(t)=q(t)$

nous donne $x_2(t)=\dot{x}_1(t)=\dot{q}(t)$

Ensuite la deuxième dérivation de l'équation $x_2(t)=\dot{x}_1(t)$

nous donne $\dot{x}_2(t)=\ddot{x}_1(t)$.

Le système ODE obtenu est donc :

$$\begin{cases}\dot{x}_1(t)=\dot{q}(t)\\ \dot{x}_2(t)=\ddot{x}_1(t)\end{cases}$$

Ainsi le système présenté est d'index 2 parce qu'il nécessite deux dérivations successives pour obtenir un système ODE.

C.1.2.2 Algorithmes de réduction d'index

En général, plus l'index d'un système est élevé, plus des difficultés numériques sont rencontrées en essayant de résoudre le système. Les systèmes avec des index supérieurs à 1 sont particulièrement difficiles à résoudre, ce qu'on appelle les DAE d'index supérieurs. Dans le domaine du génie électrique, diverses applications donnant lieu à des systèmes DAE de grande dimension proviennent de la modélisation des réseaux électriques. Généralement, ces problèmes donnent des systèmes DAE d'index 1. Cependant, il est possible de concevoir des circuits donnant lieu à des DAE d'index supérieurs, tels que les circuits contenant des amplificateurs différentiels, ce qui peut être réalisé en utilisant des amplificateurs opérationnels [BRE-96].

La solution directe des systèmes d'index élevé, sauf pour quelques formes spécifiques, n'est pas possible. Les résultats d'études menées dans cette thématique sont résumés dans [GEA-85] [BRE-96]. De plus, l'index n'est pas une caractéristique du système modélisé mais du modèle lui-même, c'est-à-dire de la manière dont sont écrites les équations. Par conséquent, les problèmes d'index élevé sont résolus après une réduction de leur index par manipulations symboliques, en dérivant les équations par rapport au temps un certain nombre de fois. Ensuite, le système résultant est résolu par des solveurs qui ne sont généralement capables que de traiter des systèmes dont l'index est inférieur ou égal à 1 [BRE-89]. Nous nous ne sommes restreints à la forme implicite des DAE d'index inférieur ou égal à1.

L'algorithme de réduction d'index le plus utilisé est l'algorithme de «Pantelides». C'est une procédure itérative réduisant l'index en utilisant la théorie des graphes pour établir l'ensemble minimum des équations qui doivent être dérivées afin d'éliminer une singularité structurelle [PAN-88]. D'autres algorithmes basés sur la dérivation symbolique sont définis dans [GEA-88] [BAC-90] et [CHU-90].

Une autre méthode appelée «dummy derivatives» ou «dérivés fictives» a été proposée [MAT-93]. Elle consiste à surcharger le système d'équations DAE original avec des dérivées des équations. Le système DAE résultant est complètement déterminé par le remplacement d'une dérivée temporelle avec une variable muette algébrique pour chaque équation supplémentaire dérivée. De cette façon, certaines dérivées sont retirées de la discrétisation employée par l'intégrateur numérique

C.1.2.3 Exemples de réduction d'index

Reprenons l'exemple du pendule en mouvement (Eq 2-8). Les variables d'état du système sont les éléments du vecteur (x, y, Vx, Vy, λ). Notre objectif est de trouver l'index de ce système DAE. Pour cela, nous allons effectuer des dérivations successives sur les différentes équations afin d'obtenir un système ODE c'est-à-dire expliciter les dérivées $(\dot{x}, \dot{y}, \dot{Vx}, \dot{Vy}, \dot{\lambda})$ des variables d'états.

Les quatre premières équations du système explicitent les quatre premières dérivées.

$$\begin{cases} \dot{x} = V_x \\ \dot{y} = V_y \\ m \cdot \dot{V}_x = -2 \cdot x \cdot \lambda \\ m \cdot \dot{V}_y = -2 \cdot y \cdot \lambda - mg \end{cases}$$

Il nous reste à expliciter uniquement la dérivée $\dot{\lambda}$ de la dernière variable d'état λ. Nous allons donc travailler sur l'équation des contraintes

$$x^2 + y^2 = l^2$$

La première dérivation de celle-ci nous donne :

$$x \cdot \dot{x} + y \cdot \dot{y} = 0.$$

En remplacent alors $\dot{x}$ et $\dot{y}$ par leurs expressions, nous obtenons

$$x \cdot V_x + y \cdot Vy = 0.$$

Nous dérivons encore une fois l'équation obtenue, qui donne l'expression suivante :

$$\dot{x} \cdot V_x + x \cdot \dot{V}_x + \dot{y} \cdot Vy + y \cdot \dot{V}y = 0.$$

Maintenant, en remplaçant $\dot{V}_x$ et $\dot{V}_y$ par leur expression afin de faire apparaître la variable λ, ainsi que $\dot{x}$ et $\dot{y}$, nous obtenons alors l'expression :

$$m \cdot (V_x{}^2 + Vy^2) - m \cdot g \cdot y + 2 \cdot (x^2 + y^2) \cdot \lambda = 0$$

Finalement en nous appuyant sur l'équation de la contrainte

$$x^2 + y^2 = l^2$$

et en dérivant une troisième fois, nous pouvons expliciter l'expression $\dot{\lambda}$:

$$2 \cdot l^2 \cdot \dot{\lambda} = -m \cdot (2.V_x.\dot{V}_x + 2.Vy.\dot{V}y) + m \cdot g \cdot \dot{y}$$

Ainsi, nous pouvons conclure que le système DAE du pendule est d'index 3, à cause de la forme de l'équation de contrainte, parce qu'il nécessite trois dérivations successives pour expliciter le système ODE. Le système ODE final (d'index 0) s'écrit :

$$\begin{cases} V_x = \dot{x} \\ V_y = \dot{y} \\ m \cdot \dot{V}_x = -2 \cdot x \cdot \lambda. \\ m \cdot \dot{V}_y = -2 \cdot y \cdot \lambda. - m \cdot g \\ 2 \cdot l^2 \cdot \dot{\lambda} = -m \cdot (2.V_x.\dot{V}_x + 2.Vy.\dot{V}y) + m \cdot g \cdot \dot{y} \end{cases}$$

Nous retenons, pour la suite de ce chapitre, le système d'équations équivalent d'index 1 du pendule, définie par :

$$\begin{cases} V_x = \dot{x} \\ V_y = \dot{y} \\ m \cdot \dot{V}_x = -2 \cdot x \cdot \lambda. \\ m \cdot \dot{V}_y = -2 \cdot y \cdot \lambda. - m \cdot g \\ m \cdot (V_x{}^2 + Vy^2) - m \cdot g \cdot y + 2 \cdot (x^2 + y^2) \cdot \lambda = 0 \end{cases}$$

C.1.3 Calcul d'une solution initiale consistante

Une fois les équations formant le modèle continu sont sélectionnées, il faut déterminer des conditions initiales cohérentes afin de pouvoir démarrer l'intégration temporelle. La spécification des conditions initiales pour les systèmes DAE est très différente des systèmes ODE. Pour ces derniers, un ensemble de conditions initiales permet de déterminer une solution de façon unique. En d'autres termes, nous sommes libres de choisir toutes les variables à initialiser. Le degré de liberté offert est égal à la taille du système en question. Contrairement à cela, dans les systèmes DAE, toutes les variables ne peuvent pas être librement initialisées. Les valeurs initiales des variables différentielles X et de leurs dérivées $\dot{X}$, ainsi que les variables algébriques Y (noté par X(0) ; $\dot{X}(0)$; Y (0)) doivent obligatoirement satisfaire l'équation (2.10) à l'instant initial t_0. Si cela est possible, la solution est donc appelée «solution consistante». Une autre caractéristique des systèmes DAE est que les valeurs initiales doivent aussi satisfaire les contraintes algébriques supplémentaires.

$$\begin{cases} F(\dot{X}_0(t), X_0(t), Y_0(t), u(t), p, t_0) = 0 \\ \quad g(Y_0(t), X_0(t), U(t), P, t) = 0 \end{cases} \qquad \text{Eq 2-10}$$

Différentes techniques existent concernant la détermination de l'état initial du système (X(0), Y(0), $\dot{X}(0)$) à partir d'informations suffisantes [PAN-88] [LEI-91] [KRO-92] [CAM-96] [BRO-98] [WU-00].

En ce qui concerne l'exemple du pendule en mouvement, une solution pour des conditions initiales consistantes peut se présenter ainsi :

$$\begin{cases} X = (x, y, V_x, V_y, \lambda) \\ X(t_0) = [l,0,0,0,0] \ et \ \dot{X}(t_0) = [0,0,0,g,0] \end{cases} \qquad \text{Eq 2-11}$$

C.1.4 Discontinuités dans les DAE hybrides

Un système d'équations DAE peut présenter des discontinuités, posséder une structure variable ou posséder des changements à un certain moment dans le temps. Ces types de systèmes sont appelés des DAE hybrides. L'idée est de diviser l'espace d'état du système en un ensemble de sous-espaces différents, où on se ramène à des DAE continues. Par exemple, un système DAE hybride (cas à deux états) peut être décrit comme :

$$F(\dot{X}(t), X(t), Y(t), u(t), p, t) == \begin{cases} f_1(\dot{X}(t), X(t), Y(t), u(t), p, t) & \text{si } S > 0 \\ f_2(\dot{X}(t), X(t), Y(t), u(t), p, t) & \text{Sinon} \end{cases} \qquad \text{Eq 2-12}$$

où *S* est la fonction de commutation. Ainsi, le système DAE possède deux modèles : le premier f_1 est valide pour où S> 0 et le second f_2 lorsque la condition n'est plus vraie. La commutation d'un système à l'autre provoque une discontinuité dans le système, que les solveurs continus ne peuvent pas intégrer. Afin de faire face à ce problème, la discontinuité doit être détectée et le solveur réinitialisé après le point de discontinuité. Par conséquent, dans les simulateurs hybrides, lors de l'intégration des équations, le simulateur lui-même doit prendre en compte les discontinuités. Celles-ci doivent être détectées à l'avance pour arrêter

l'intégration et la recommencer à nouveau après que la discontinuité ait été traitée. En effet, selon l'ordre d'intégration des DAE, il faut veiller à conserver la même DAE par pas de résolution temporelle pour éviter toute incohérence de résolution [NOR-92] [GER-93].

La différence par rapport à l'intégration d'un modèle purement continu est due au fait qu'on doit être capable de détecter et localiser des événements. C'est en effet l'instant d'occurrence de ces événements qui va déterminer l'arrêt de l'intégration, afin de les propager dans le système discret. Il est donc très important de connaître très précisément cet instant.

Pour détecter et localiser un tel instant, les solveurs utilisent souvent les fonctions de passage par zéro (zero-crossing). Il s'agit par exemple du passage d'une grandeur à un seuil par la détection de zéro dans une forme d'onde, comme l'illustre la Figure 2-14. Dans notre exemple, la fonction S(X, t) peut être utilisée comme la fonction de passage à zéro.

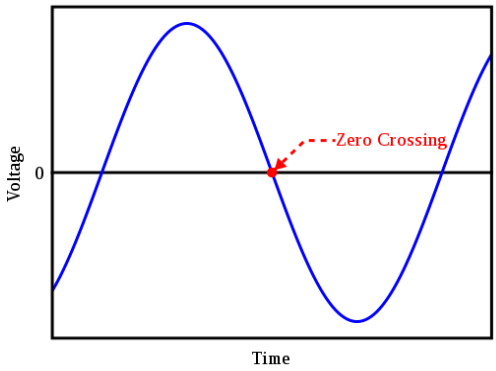


Figure 2-14 : Zero-crossing dans une onde représentant la tension vs du temps

La difficulté de localisation de ces événements réside dans le fait que l'instant d'occurrence n'est pas connu à l'avance, il peut se produire entre deux instants d'intégration. L'approche utilisée cherche à détecter si un événement s'est produit entre deux pas d'intégration, puis cherche à le localiser précisément dans cet intervalle. Pour qu'elle soit efficace, cette détection doit être couplée avec l'intégration continue.

La Figure 2-15 montre le même signal qui passe par zéro dans deux configurations différentes. Dans le premier cas, les itérations successives d'intégrations ne peuvent pas détecter l'événement (pas de changement de signe). Dans la seconde, le solveur détecte l'événement.

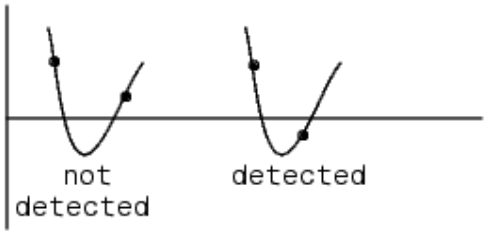


Figure 2-15 : Problèmes de détection des événements

Lorsqu'une discontinuité se produit, le système peut devenir inconsistant ; c'est-à-dire que les valeurs des variables continues ne satisfont plus l'égalité des équations à 0. Par conséquent, le simulateur doit calculer un nouvel état consistant.

Pour les modèles hybrides, l'exemple du pendule n'est plus adapté, c'est pourquoi nous allons travailler sur un exemple classique également, celui d'une balle rebondissant. Ce type de modélisation hybride se retrouve en génie électrique pour les actionneurs électromécaniques tel que le déclencheur électromagnétique présenté dans le chapitre 1 ; lorsque la partie mobile vient percuter une butée. C'est également le cas pour des circuits à composants tels que des diodes ou transistors modélisés par des interrupteurs idéaux.

La dynamique d'une balle rebondissant de masse *m* et soumise à l'action de la gravité *g* peut être modélisée comme un système dynamique hybride. La balle est lâchée d'une altitude initiale $x(t0) = h$ avec une vitesse initiale nulle. L'altitude x*(t)* de la balle suit donc l'équation différentielle issue de la mécanique classique :

$$m.\ddot{x}(t) = -mg$$

Quand elle touche le sol, cette balle perd de l'énergie à chaque rebondissement avec un coefficient d'amortissement *c* ($0 \le c \le 1$).

L'automate modélisant le comportement hybride de la balle bondissant (Figure 2-16) est donné par les équations suivantes :

- système DAE

$$\begin{cases} v(t) = \dot{x}(t) \\ \dot{v}(t) = -g \end{cases}$$

- condition de la transition :

$$\begin{cases} x(t) = 0 \\ v \le 0 \end{cases}$$

- réinitialisation de la vitesse *v* sur la transition :

$$v(t) = -c \cdot v(t)$$

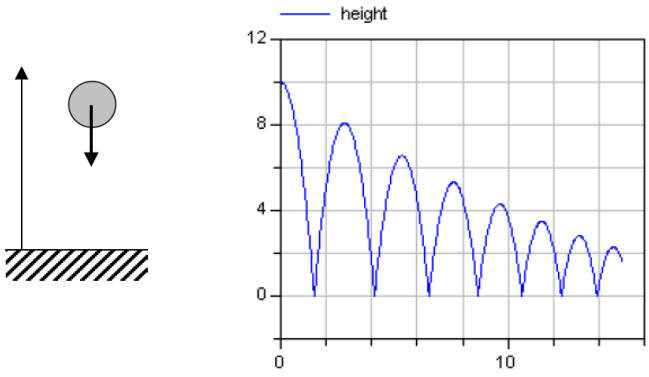


Figure 2-16 : Principe de la balle rebondissant

C.2. Résolution d'un système DAE

La résolution des équations différentielles algébriques (DAE) présente des difficultés analytiques et numériques ; qui sont très différentes de celles rencontrées dans les ODE en raison de l'existence des contraintes algébriques [PET-83]. L'étude systématique des DAE a commencé dans les années 80 et a donné lieu à plusieurs méthodes permettant leur résolution directe. Ces résultats de recherche sont résumés dans [GRI-86] [HAI-89] [HAI-91] [BRE-96].

C.2.1 Les solveurs DAE

Comme nous l'avons précisé précédemment, un système de DAE peut être résolu soit par des méthodes dédiées, soit par les méthodes de résolution des systèmes ODE après l'avoir transformé. Habituellement, il est recommandé de résoudre le système DAE de la première manière [PET-83].

Depuis les années 1970, des méthodes numériques pour la résolution des systèmes DAE ont été développés. La première méthode a été proposée par Gear [GEA-71], elle est basée sur une formule de différence en arrière (appelée méthode BDF[52]). Un grand nombre d'algorithmes ont été mis en œuvre sur ces méthodes. Nous citerons par exemple, LSODI [HIN-80], DASSL [PET-83], DASOLV [JAR-92] RADAU IIA [HAI-91] et IDA[53] qui sont capables de résoudre les systèmes DAE d'index 0 et 1.

Dans la suite, nous allons nous focaliser uniquement sur famille des solveurs DASSL (Differential Algebraic System Solver), les plus représentatives des solveurs DAE. Elle représente aujourd'hui l'une des plus performants sur le marché et elle est largement utilisée dans de nombreuses applications industrielles.

C.2.2 La famille des solveurs DASSL

La famille des solveurs DASSL contient principalement :

- le solveur DASSL [PET-83] et son successeur DASPK[54] [SHE-00] pour les systèmes DAE continus d'index 0 et 1,
- le solveur DASRT[55] et son successeur DASKR[56] (DASSL avec recherche des zéros, intégration de la gestion des événements) [BRO-94] pour la résolution des systèmes DAE Hybrides.

Ces solveurs proposent des sous-routines codées en langage Fortran. Ils mettent en œuvre l'algorithme BDF avec un pas de calcul et un ordre variables, qui sont sélectionnés de manière adaptative pour maintenir la stabilité et l'efficacité de la résolution.

La famille DASSL propose deux sous-programmes utilisateur :

- une routine qui définit la fonction résiduelle $F\,(\mathrm{F}(\mathrm{x}(\mathrm{t}), \dot{\mathrm{x}}(\mathrm{t}), \mathrm{u}(\mathrm{t}), \mathrm{p}, \mathrm{t}) = 0)$ du système DAE continu (DASSL et DASPK),
- une routine de «détection d'événement» dans laquelle des surfaces de zero-crossing sont définies (DASRT et DASKR).

DASSL, dans son fonctionnement de base, cherche à trouver une approximation de la solution du vecteur d'inconnu et de ses dérivées à un chaque pas de temps. Au départ, une valeur prédite de la solution et de ses dérivés, à l'instant de calcul, est formée par évaluation d'un polynôme prédicateur et de sa dérivée. Ensuite, un polynôme de correction est trouvé afin de valider les solutions. Finalement, DASSL emploie une interpolation pour calculer les solutions entre les points de discrétisation dans l'intervalle du pas de calcul temporel.

[52] BDF : Backward Difference Formula
[53] IDA : Implicit Differential Algebraic Solver
[54] DASKR : Differential Algebraic System Solver using preconditioned Krylov
[55] DASRT : Differential Algebraic System Solver with Root Finding
[56] DASKR : Differential Algebraic System Solver using preconditioned Krylov with Root Finding

La résolution de ce polynôme de correction, fait intervenir le calcul d'une matrice d'itération particulière contenant les dérivées partielles, appelée jacobian modifié J, sous la forme :

$$J = \sigma \cdot \frac{\partial F}{\partial \dot{X}} + \frac{\partial F}{\partial X}$$

où σ est un mesure donnée par le solveur.

Afin de résoudre son système, DASSL a besoin d'une estimation de cette matrice itérative (jacobian). Il propose, par défaut, d'approcher cette matrice numériquement par des différences finies. L'avantage de cette option est qu'elle est simple à mettre en œuvre. Cependant, une expression analytique de la matrice jacobienne obtenue par calcul symbolique est plus précise et conduit à une meilleure performance de simulation, si une description symbolique du système existe. DASSL offre donc à l'utilisateur la possibilité de fournir le Jacobian propre à son modèle. Lorsque le système d'équations est compliqué (formules mathématiques assez lourdes), il peut être intéressant d'utiliser un moteur de calcul formel ou de la dérivation de code pour générer le code source correspondant au jacobian. L'avantage est double : éviter une erreur de codage à la main et obtenir un calcul plus précis.

DASSL doit être aussi appelé par une routine externe de calcul des conditions initiales consistantes (sauf pour DASKR) ; les paramètres de contrôle sont définis à l'avance, tel que le sous-programme DAEIS[57] [WU-00].

D'une manière générale, nous présenterons la structure générale d'un algorithme de résolution d'une DAE hybride sur la Figure 2-17, mettant l'accent sur la structure principale plutôt que des détails. Cette approche simplifiée est la base de fonctionnement des solveurs DAE hybrides tels que DASRT et DASKR que nous présentons brièvement ici.

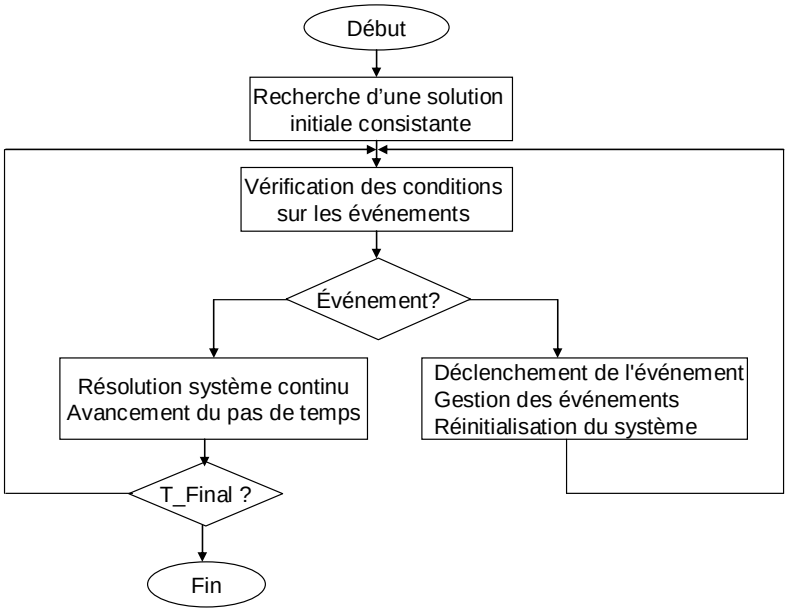


Figure 2-17 : Structure générale d'un algorithme de résolution des DAE hybrides [LUN-05]

[57] DAEIS : Differential Algebraic Equations Initialization Subroutine

Tout d'abord, un ensemble de valeurs initiales consistantes doit être trouvé sur la base des contraintes données. Ensuite, le solveur hybride DAE vérifie si toutes les conditions gérant les événements sont devenues *vraies* et lui permet de déclencher les événements. S'il n'y a aucun événement, le solveur continu (DASSL) est utilisé pour résoudre numériquement le système jusqu'à ce qu'un événement arrive ou que la fin de la simulation soit atteinte.

Si les conditions de détection d'un événement sont remplies, l'événement est déclenché, c'est-à-dire que les équations conditionnelles associées sont activées. Cela signifie que les variables dépendantes de l'événement sont déterminées et leurs nouvelles valeurs sont calculées. Puis, la réinitialisation du système (qui consiste à retrouver les nouvelles valeurs initiales consistantes) se déclenche afin de pouvoir redémarrer le solveur continu DAE.

Nous avons choisi d'implémenter cette famille de solveurs. Malgré les quelques inconvénients qu'ils présentent [BRE-96], ils restent puissants et adaptés à nos applicatifs.

D. Développements relatifs aux modèles et solveurs DAE hybrides

Dans le paragraphe suivant, nous détaillerons les différents développements réalisés relatifs aux modèles et aux solveurs DAE hybrides : extensions de la norme ICAr pour la description des modèles DAE et du simulateur de CADES pour leur exploitation.

D.1. Extension des ICAr pour les systèmes DAE continus et hybrides

Nous proposons maintenant une extension de la norme ICAr permettant une modélisation plus naturelle des systèmes physiques, en d'autres termes une approche acausale basée sur les systèmes des équations différentielles algébriques implicites. Pour se faire, nous devons créer de nouvelles facettes pour les composants logiciels ICAr.

D.1.1 Extension pour la simulation temporelle acausale continue

Un modèle dynamique continu représentant un système d'équations différentielles algébriques DAE, selon ce qui a été présenté précédemment, est principalement défini par :

- la fonction résiduelle *F*,
- la matrice des dérivées partielles : c'est le Jacobian de la fonction résiduelle F par rapport à chacune des variables (différentielles et algébriques) et de leurs dérivées
- et en option, les sorties que l'utilisateur souhaite récupérer (ces dernières ne font pas partie de la résolution du système, elles sont calculées généralement après chaque pas de résolution temporelle).

La Figure 2-18 présente la structure, que nous proposons, des composants ICAr pour la simulation dynamique acausale par description DAE continue. Ce composant contient une facette appelée « **ContinousDAESystem** » spécifiant le service rendu qui impose un certain nombre d'entrées et sorties prédéfinies. Ce composant assure l'interaction du modèle DAE qu'il contient avec un solveur DAE continu de type DASSL ou DASPK. L'interaction entre eux est définie comme suit :

- le composant récupère les valeurs initiales, les paramètres P et les entrées externes U ;
- le composant fournit au solveur la fonction résiduelle, la matrice de dérivées partielles et l'instant de calcul ;

- après résolution, le solveur lui retourne les valeurs des variables différentielles X et de ses dérivées $\dot{x}$, les variables algébriques Y et le prochain instant de calcul ;
- le composant produit des sorties supplémentaires.

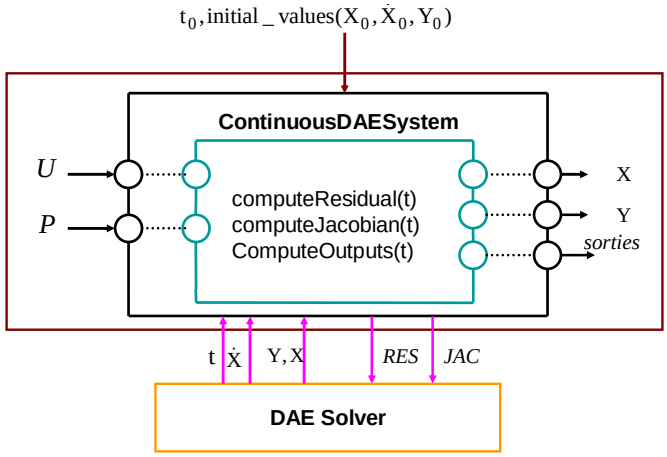


Figure 2-18 : Structure globale d'un composant pour la simulation DAE continue

La Figure 2-19 illustre la facette « ContinuousDAESystem » que nous avons créé. Elle contient les méthodes permettant de solliciter le calcul des équations du modèle, de manipuler les données du système et d'interagir avec un solveur DAE acausal (DASSL, DASPK) pour réaliser les étapes de la simulation dynamique d'un système implicite. Cette facette hérite, comme toutes les autres, de la facette principale « Facet » et «CommonStateExplicitSystem» pour l'initialisation des paramètres non temporels et des vecteurs d'entrées externes. La facette « ContinuousDAESystem » contient les méthodes suivantes.

- *initialize (double date)* : cette méthode sert à l'initialisation du système DAE. Elle est appelée une seule fois au début de la simulation pour vérifier si la solution est consistante ou pas, sinon elle essaie d'en calculer une nouvelle. Ensuite, cette méthode n'est appelée que si la simulation s'arrête avant le temps final prévu et redémarre.
- *setNbOfEquations (int n) :* cette méthode sert à définir la taille du système DAE
- *void computeResidualFunction (double date)* : cette méthode évalue la fonction résiduelle à la date de simulation courante.
- *void computePartialDerivativeMatrix (double date*) : cette méthode calcule la matrice des dérivées partielles du système implicite $J = \left[\frac{\partial F}{\partial \dot{X}}, \frac{\partial F}{\partial X}\right]$ à l'instant courant.
- *void computeContinuousOutputs (double date) :* cette méthode calcule des sorties souhaitées par l'utilisateur à la date courante.
- *double[] getResidus (double date*) : cette méthode retourne les valeurs de la fonction résiduelle à la date courante.
- *double[][] getJacobian (double date) :* cette méthode retourne les valeurs de la matrice des dérivées partielles à la date courante.

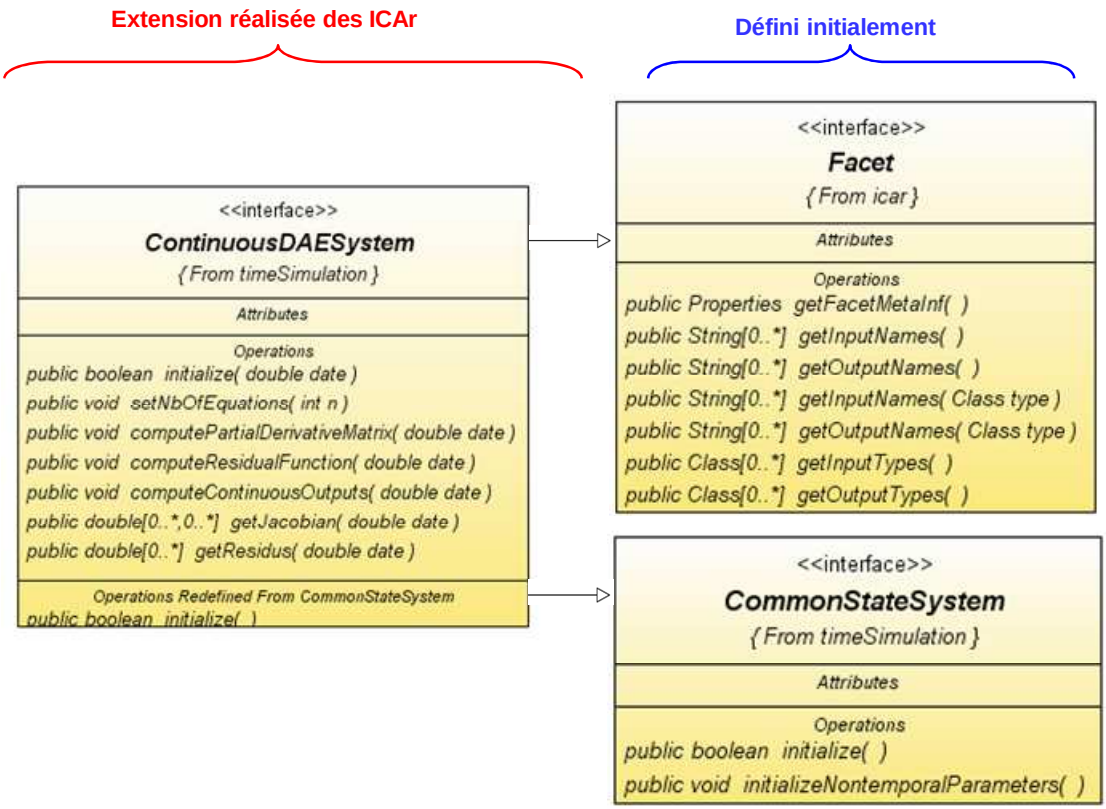


Figure 2-19 : Facette «ContinuousDAESystem» réalisée pour la simulation acausale

D.1.2 Extension CADES pour une simulation temporelle acausale hybride

Dans le cas du modèle DAE hybride (continu avec des événements), le modèle doit être capable de posséder ou calculer toutes les informations requises telles que la date de tout événement prochain [NOR-92]. Sinon, il doit définir une liste de variables à surveiller dans le but d'intercepter un passage par zéro [LUN-05]. Pour gérer ce type de modèle, nous proposons d'ajouter une extension à la facette DAE continue que nous venons de présenter, qui devra posséder les particularités suivantes afin de bien gérer la détection des événements :

- autoriser les conditions de branchement (notion de seuil et d'événement),
- autoriser la définition des variables discrètes,
- gérer les comparaisons entre les variables, lors d'un passage des variables par zéro.

Nous avons ainsi créé une facette nommée «**HybridDAESystem**», comme illustré sur la Figure 2-20. Elle contient les méthodes permettant de solliciter le calcul des équations du modèle, de manipuler les données du système et d'interagir avec un solveur DAE Hybride (DASRT, DASKR) pour réaliser une simulation dynamique d'un système implicite.

Cette facette hérite de la facette «**ContinuousDAESystem**». Les deux facettes interagissent à tous les pas de temps : la facette DAE continu s'occupe de traiter le fonctionnement du système continu jusqu'à l'apparition d'un événement. Quant à elle, la facette DAE hybride surveille l'évolution du système continu afin de détecter les événements.

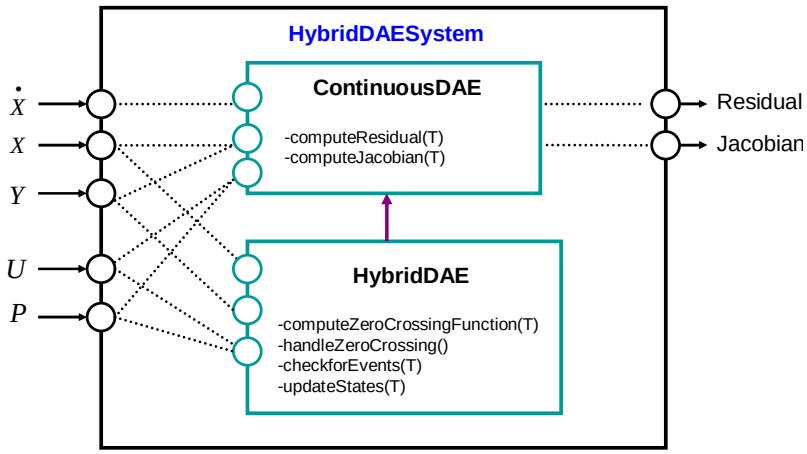


Figure 2-20 : Facettes d'un composant ICAr supportant les DAE hybrides

Un composant dynamique, contenant un système DAE continu ou hybride, peut désormais être couplé à un solveur tel que le simulateur système Portunus (voir chapitre 3). Le solveur de l'outil de simulation Portunus va intégrer le composant ICAr DAE et le résoudre.

En complément du couplage des composants ICAr DAE aux solveurs du marché (Portunus), nous proposons également une extension du module « ComponentSimulator », un service offert par la plateforme CADES pour la simulation temporelle (chapitre 1). Pour cela nous devons définir une facette de simulation dédiée, qui permet de prendre en charge l'interaction entre le modèle de composant ICAr DAE et les solveurs pour sa résolution numérique.

D.2. Couplage d'un ICAr à un solveur DAE dans CADES

D.2.1 Environnement de simulation que nous proposons

L'architecture du module « ComponentSimulator » que nous proposons, en tant que simulateur hybride, est présentée sur la Figure 2-21. Ce simulateur repose sur les méthodes numériques de simulation des systèmes dynamiques hybrides présentées dans les paragraphes précédents. Nous allons décrire les différents modules sur lesquels repose ce simulateur.

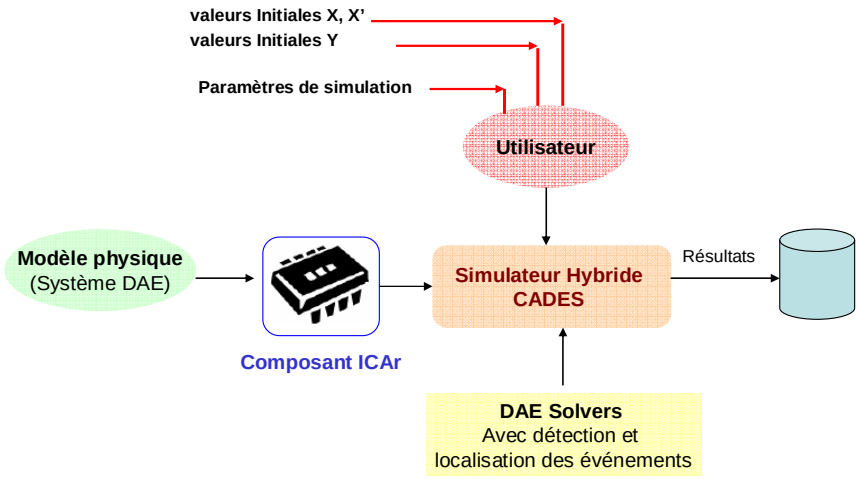


Figure 2-21 : Architecture de la simulation hybride

L'architecture proposée, comme le montre la Figure 2-22, met en avant trois principaux composants en interaction : l'entité de simulation, le modèle physique et le solveur.

L'entité de simulation représente le cœur de la simulation (Main loop) et met en évidence le modèle du système physique (composant ICAr DAE) et le solveur associé pour sa résolution (composant solveur). Cette entité contrôle l'exécution de la simulation ainsi que le fonctionnement du solveur. Elle assure également l'échange des données avec le modèle du système DAE, en fournissant par exemple ses valeurs initiales et en récupérant ses résultats.

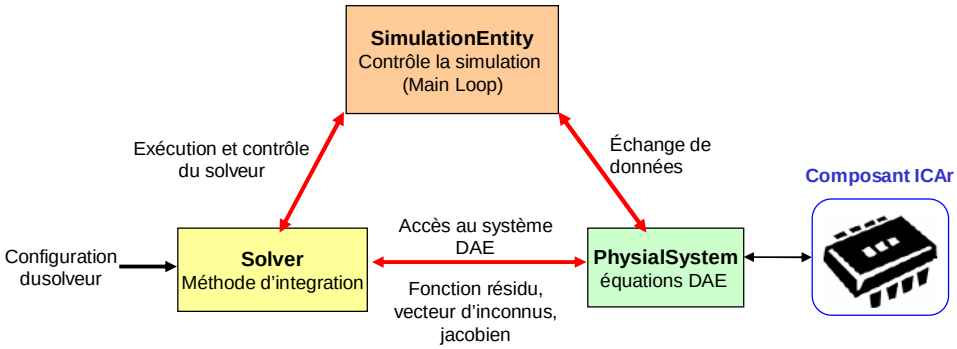


Figure 2-22 : Vue d'ensemble des Composants de la simulation hybride

Le modèle représentant le système physique, sous forme d'une DAE hybride ou continue, est décrit en implémentant les facettes de simulation temporelle que nous avons réalisées (décrites dans le paragraphe précédent) au sein d'un composant logiciel ICAr.

Le composant solveur encapsule des algorithmes d'intégrations temporelles numériques pour la résolution des systèmes DAE ; nous avons intégré la famille DASSL dans ces composants. Le solveur doit être configuré initialement soit par l'utilisateur, soit via une configuration par défaut et exécute les ordres de l'entité de simulation en échangeant des informations avec le modèle afin de le résoudre.

D.2.2 Interfaces de pilotage de la simulation hybrides et continues

Pour que le module « ComponentSimulator » puisse exploiter des composants ICAr DAE continue ou hybride, ou pour que la simulation d'un composant soit autonome sans avoir recours à un solveur externe, nous avons créé une facette « **ImplicitSystemSimulator** » pour définir les propriétés que doit implémenter un tel simulateur. Cette facette, présentée sur la Figure 2-23, contient les méthodes permettant :

- de manipuler les données d'entrées (paramètres de configuration et d'initialisation du modèle et du solveur) et de sorties (résultats obtenus) d'une simulation ;
- de résoudre le système d'équations DAE d'un modèle dynamique.

Nous pouvons retenir les méthodes suivantes :

- *setImplicitSystem (component, continuousDAESystem) :* pour associer le solveur au composant ICAr et à sa facette *continuousDAESystem*,
- *setMethod(method)* : pour définir le choix de la méthode de résolution DAE,
- *run ()* : pour activer la simulation,
- *calcul (t)* : pour interagir avec le solveur afin de résoudre le système à un instant *t*,
- *setTstart (T0) et setTstop(Ts)* : pour définir le départ et la fin de la simulation,

- InitModel (X, Y, Xprime) : pour initialiser le modèle au début de la simulation,
- InitSolver () : pour définir les paramètres nécessaires pour initialiser le solveur.

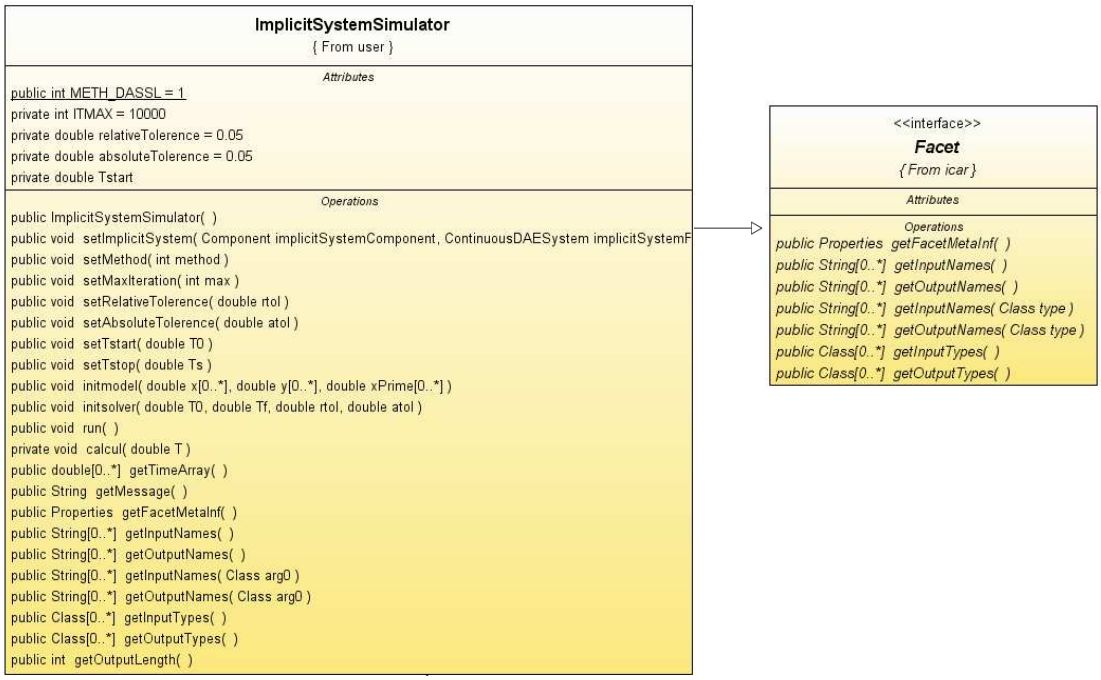


Figure 2-23 : Facette «ImplicitSystemSimulator » developpée

Le composant ICAr DAE peuvent implémenter ainsi les différentes facettes «**ContinuousDAESystem** », «**HybridDAESystem** » et « **ImplicitSystemSimulator** » ; lui permettant d'être exploitable pour une simulation avec ou sans solveur dynamique.

Pour résoudre numériquement les modèles algébro-différentiels réalisés, nous avons intégré différents solveurs de la famille DASSL dans le module « ComponentSimulator » :

- un solveur DAE continue : DASSL [BRE-96],
- un solveur DAE hybride (continu avec zero-crossing) : DASRT / DASKR [BRO-94],

Nous rappelons que ces différents solveurs proposent des sous-routines codées en langage Fortran pour la résolution des systèmes. Ils sont en libre accès[58].

Nous avons, en premier lieu, créé une interface spécifique «DAESolver » définissant les fonctionnalités communes et les propriétés de ces solveurs DAE (Figure 2-22).

Ensuite, nous nous sommes intéressés à l'implémentation des différentes méthodes de résolution dans ces solveurs. Le fait que les solveurs de la famille DASSL soient écrits en Fortran a rendu leur intégration dans notre simulateur délicate, parce que le simulateur, ainsi que les composants logiciel ICAr, sont programmés en langage JAVA.

Nous avons ainsi créé les interfaces de communications et une extension permettant d'appeler les routines Fortran des solveurs depuis un code JAVA et réciproquement. JAVA fournit une interface de communication (via la JNI[59]) avec le langage C mais pas directement en Fortran. Nous avons donc implémenté la couche d'interfaçage et de communication entre le code en Fortran et celui en C.

[58] http://www.netlib.org/ode/

[59] JNI : *Java Native Interface*

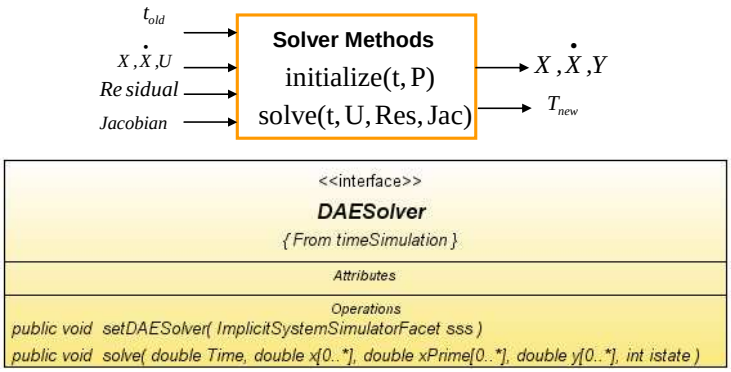


Figure 2-24 : Solveur DAE et facette correspondante

Les exemples de références du pendule et de la balle rebondissant ont été mis en œuvre et testés avec succès dans cet environnement.

Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur la simulation dynamique des systèmes complexes à travers les descriptions et approches temporelles des modèles ainsi que les caractéristiques de leur résolution et intégration temporelle.

Nous avons présenté plus particulièrement les modèles à travers une description par des équations différentielles algébriques implicites (DAE). Cette dernière n'étant pas disponible dans la norme actuelle des composants logiciels ICAr. Nous avons donc étendu cette norme pour la description des modèles DAE continus et hybrides. Nous avons aussi enrichi le simulateur de CADES par des solveurs DAE permettant leur exploitation.

Maintenant, nous disposons d'une formalisation (composant logiciel ICAr) pour supporter nos modèles de génie électrique sous la forme des DAE. Nous disposons également des solveurs DAE permettant leur simulation dans l'environnement CADES.

Les bases de la simulation dynamique ODE/DAE étant posées dans ce chapitre, le suivant développe la problématique d'interopérabilité des modèles et des outils que nous avons évoquée dans le chapitre 1. Dans ce cadre, dans le chapitre 3 nous verrons l'exploitation des composants ICAr par des solveurs ou en co-simulation dans les simulateurs systèmes.

CHAPITRE 3

Interopérabilité de modèles par approche de « Plug'in »

Introduction

Comme nous l'avons précisé dans le chapitre 1, les produits industriels requièrent l'utilisation de divers sous-systèmes de différentes natures interagissant entre eux. Habituellement, l'analyse de chaque sous-système est menée séparément par des outils «métiers». Chacun de ces derniers est souvent dédié à une problématique donnée, un type d'étude donnée, à un ou plusieurs domaines physiques avec en général un niveau de modélisation adapté aux objectifs d'analyse pour lequel il a été conçu. Ces outils métiers étant très spécialisés, et souvent très puissants, il est parfois intéressant de pouvoir continuer à profiter de leurs atouts dans des contextes plus généraux. Nous avons vu aussi qu'il est intéressant, dans ce contexte, de travailler avec des modèles de type «boule blanche» respectant certains standards (VHDL-AMS) pour des besoins de réutilisation et d'échange de modèles entre concepteurs et outils. En conséquence, les concepteurs s'attendent à ce que leurs modèles puissent être intégrés le plus fidèlement possible dans leur outil de simulation système pour la validation du système global.

Pour la modélisation des systèmes magnétiques (actionneurs, MEMS magnétique) le G2Elab dispose d'une large panoplie d'outils dédiés à la conception de modèles électromagnétiques : Flux2D/3D (systèmes magnétiques par éléments finis), Reluctool (systèmes magnétiques modélisés par réseaux de réluctances), MacMMems (microsystèmes magnétiques modélisé par une méthode intégrale) et FGOT (réduction de modèle par surfaces de réponse). Rappelons que ces outils utilisent leurs propres méthodes et approches de descriptions de modèles. Ils proposent d'une part un formalisme approprié aux caractéristiques des structures physiques étudiées facilitant leur modélisation (IHM dédiée, bibliothèques de composants, etc.) ; et d'autre part une projection (encapsulation ou traduction) vers un format commun (composant logiciel ICAr) permettant différentes analyses (simulation, optimisation, analyses de sensibilités, etc.).

En outre, les outils de simulation multi-domaine actuels choisis dans le cadre de ces travaux (Portunus, Simplorer, Smash, etc.), permettent d'analyser des systèmes complexes : de l'analyse comportementale des composants à l'analyse globale du système, dans un environnement de conception unique. Ces outils intègrent parfaitement des formalismes différents de modélisation système (circuits, diagrammes de blocs, machines d'états, équations) ainsi que des langages de modélisation (VHDL-AMS, C/C++, langages propriétaires) qui peuvent être utilisés simultanément dans le même schéma. Ils sont aussi souvent interfacés avec des outils de simulation fine, tels que Flux2D, Ansys, etc.

Nous proposons, dans nos travaux, d'assurer l'interopérabilité des modèles et des outils spécialisés via une approche de couplage ; pour mieux répondre à ces besoins des concepteurs et apporter de la précision de calcul sur des composants spécifiques. Ceci se fait moyennant :

- l'interfaçage de l'outil système avec des outils spécialistes (pilotage / co-simulation),
- l'intégration des modèles réduits (surface de réponse ou tables) issus des outils spécialisés sous leur format initial sans modification,
- l'import de modèles spécifiques qui ont été transformés en des modèles compatibles avec le standard ou le langage de modélisation qu'offre l'outil système,

Par conséquent, nous avons proposé deux solutions pour répondre à ces besoins pour la modélisation et la simulation d'un système complexe (incluant les aspects magnétiques).

- la première est l'utilisation d'un environnement de simulation «système», où les différentes composantes du système sont incluses en tant que blocs et interconnectées via l'environnement. Ceci permet de continuer à modéliser ou exploiter les sous-systèmes depuis leurs outils dédiés. Il suffit de les projeter ou d'interfacer les outils métiers, en respectant les protocoles d'échanges des différents outils de simulation.
- la seconde approche consiste à exporter les modèles issus des outils métiers vers un seul format spécifique (normalisé) utilisable par des concepteurs de compétences différentes. Cette méthode nécessite que les outils dédiés offrent la possibilité d'exporter leurs modèles vers ce format (par exemple VHDL-AMS dans notre étude).

Dans ce chapitre, nous allons étudier et mettre en œuvre la première solution assurant l'intégration des modèles existants ou développés sous forme de «boîtes noires» (composant logiciel ICar générés par les outils) dans un environnement de simulation système.

Le chapitre suivant traitera la deuxième solution ; en étudiant la problématique de la génération automatique des modèles VHDL-AMS depuis des outils métiers.

A. Interopérabilité en simulation dynamique de systèmes

Dans cette partie, nous allons étendre le concept d'exploitation d'un modèle dédié à l'usage de ce même modèle par un autre outil de calcul plus généralisé. Nous verrons que le formalisme des composants logiciel peut être un support à la simulation dynamique système via ses propriétés d'interopérabilité.

A.1. Les composants logiciels support à l'interopérabilité

A.1.1 Simulation dynamique avec l'intégration composants logiciels

La simulation dynamique multi-physique se présente selon deux voies:

- la simulation avec des blocs connectés entre eux dans un même outil.
- la co-simulation dynamique (modèle autonome) avec des outils ayant leur propre solveur et communiquant entre eux à chaque pas de simulation temporelle.

A.1.1.1 Simulation système avec des blocs intégrés

Dans ce contexte, l'utilisateur ou le concepteur construit chacun des composants ou des sous-systèmes de son dispositif indépendamment, puis les connecte dans l'environnement de simulation. Il récupère souvent des modèles de bibliothèques. Il est intéressant de pouvoir utiliser les modèles générés sous forme de « composant logiciels » par les outils de type «métiers » dans une simulation système.

Dans ce cas, un modèle issu d'un outil métier est vu comme un bloc qui est connecté avec les autres modèles (Figure 3-1), pour former un système dynamique global. Le simulateur fournit les algorithmes numériques pour résoudre les équations dynamiques du système.

L'avantage majeur de cette méthode est sa généralité et sa facilité d'utilisation. C'est aussi son inconvénient majeur ; car très souvent les modèles nécessitent des méthodes mathématiques spécifiques pour leur résolution qui ne sont pas toujours disponibles dans le simulateur. Par exemple, avec le langage VHDL-AMS, un logiciel tel que Portunus sait parfaitement résoudre des modèles de circuits, mais n'est pas apte à résoudre des intégrales spatiales requises pour une modélisation des microsystèmes magnétiques (chapitre 1).

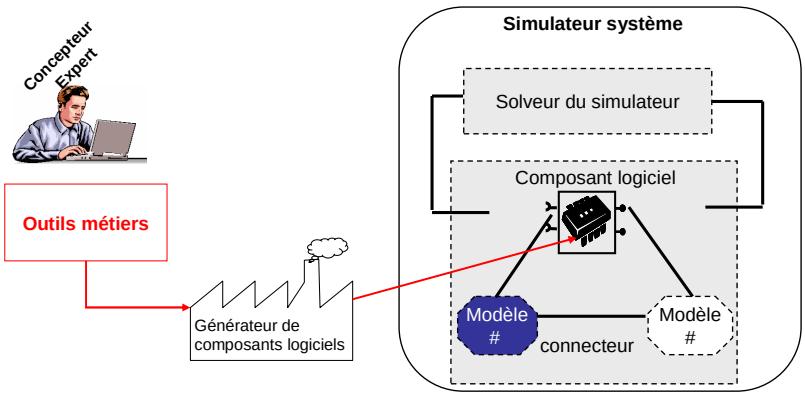


Figure 3-1 : Simulation avec des « blocs intégrés »

A.1.1.2 Co-simulation dynamique

La co-simulation est une autre manière d'intégrer un modèle issu d'un outil « métier » dans une simulation système multi-physique. Elle permet d'intégrer et de simuler un modèle issu de différents outils par interfaçage de ces derniers. La co-simulation peut faire intervenir des outils forts différents afin d'obtenir des résultats plus raffinés. Cette procédure nécessite de définir un outil qui pilote les autres selon les besoins, gère le pas de la simulation globale et les échanges de données entre les outils. Contrairement à la simulation intégrée, la co-

simulation s'effectue en exploitant plusieurs solveurs temporels simultanément (Figure 3-2 et Figure 3-3). En effet, chacun des modèles est résolu par son propre solveur et l'environnement de simulation établit les échanges entre ces modèles en synchronisant les différents solveurs pour obtenir une simulation système. La solution reposant sur la co-simulation ne nécessite pas de modifications importantes des logiciels utilisés et permet une communication simple entre les outils. Elle représente un moyen efficace pour profiter des avantages des modèles spécifiques dans les simulations systèmes.

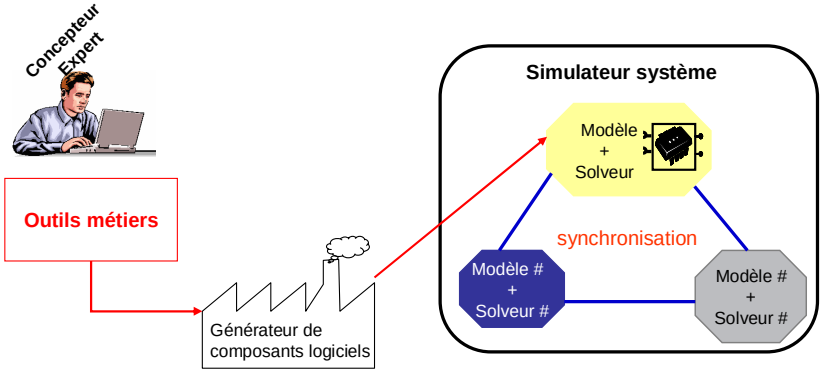


Figure 3-2 : Schéma de co-simulation exploitant les composants ICAr

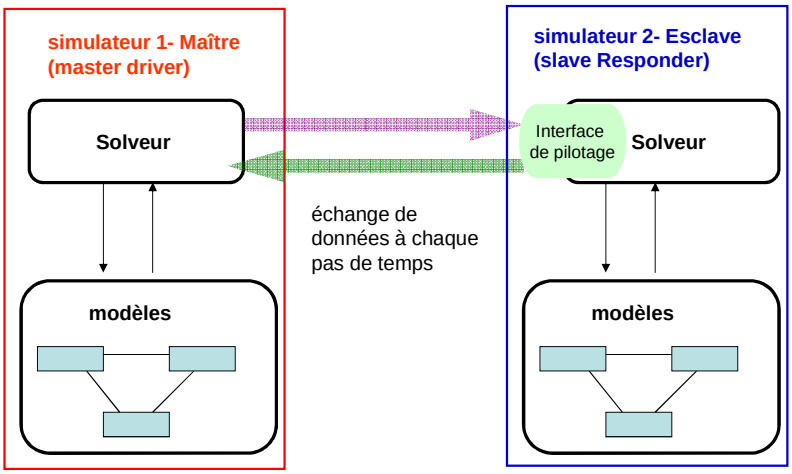


Figure 3-3 : Schéma de co-simulation ou pilotage entre outils

A.1.2 Les composants logiciels supports à l'interopérabilité

Comme nous l'avons présenté dans les chapitres 1 et 2, les composants logiciels sont des modèles de type « boîte noire » caractérisés par leurs services et leurs paramètres d'entrée et de sortie et dont le contenu est complètement protégé. L'implémentation de la norme ICAr ainsi que les extensions réalisées (chapitre 2) répondent bien aux besoins émergeants des concepteurs en terme de capitalisation, de portabilité et de réutilisabilité des modèles en protégeant leurs propriétés intellectuelles. De ce fait, le composant logiciel ICAr répond à la problématique de couplage de modèles et d'outils en vue de la simulation et la co-simulation dynamique tout en offrant une protection des modèles. Cette approche offre une interopérabilité importante [DEL-11].

Pour faciliter la compréhension par la suite, nous définissions les termes plug-in et plug-out (Figure 3-4) :

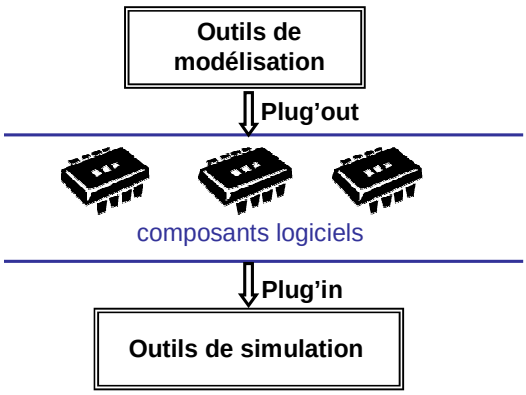


Figure 3-4 : Notions de plug-in et plug-out

- **« Plug-in » :** Il s'agit d'importer et d'exploiter un composant logiciel dans un outil de calcul ou de simulation (Portunus par exemple). Pour cela, une passerelle est nécessaire entre un composant logiciel et l'outil permettant son exploitation. Seules les facettes de calcul que l'outil est capable d'exploiter seront visibles.
- **« Plug-out » :** Il s'agit de générer ou encapsuler sous la forme d'un composant logiciel, un modèle défini dans un logiciel de modélisation (par exemple Flux2D), ou un outil implémentant un autre langage de description (par exemple Modelica). C'est une extension du concept de générateur de composant logiciel, qui peut ne pas être autonome. En effet, certains logiciels n'autorisent que des appels à leurs propres solveurs, ce qui contraint le composant à garder une dépendance vers le logiciel qu'il intègre. Cette notion sera exploitée par la suite afin de définir des types particuliers de co-simulations ou de pilotage d'outils, dans lesquels le composant logiciel gère la relation client/serveur.

Dans la suite, nous allons montrer comment ces composants logiciels peuvent être intégrés dans un simulateur «système» afin de réaliser des simulations temporelles, des co-simulations ou des échanges entre outils. Ainsi, nous allons appréhender :

- des moyens pour réaliser un plug'out (encapsulation d'un modèle de Flux2D)
- des moyens pour piloter un outil (modèle+solveur pour de la co-simulation)
- des moyens pour réaliser un plug'in (importation d'un modèle vers un simulateur)

A.2. Importation de composants logiciels dans un simulateur système

A.2.1 Principe de « plug-in »

Le principe du «plug-in» permet aux logiciels de simulation « d'appeler » un composant logiciel pour simuler le modèle qu'il contient. Différents travaux au sein de l'équipe MAGE du G2Elab ont abouti à la mise en œuvre des divers « plug-in » pour établir une connexion entre un composant logiciel ICAr et des logiciels de calcul. On retiendra les suivants :

- L. Rakotoarison a développé un couplage entre les composants logiciels ICAr statiques (modèle de dimensionnement) avec Matlab-Simulink, via les S-Functions (modèle quasi-statique). Il a été validé sur un modèle de MEMS magnétique permettant de déplacer une microgoutte en lévitation (modèle généré via MacMMems) [RAKO-07].

On notera que les modèles issus de MacMMems sont statiques (pas d'équations différentielles temporelles).

- S. Gamiche a développé un couplage entre les composants ICAr contenant un modèle dynamique avec Matlab-Simulink et AMESim. Il a validé ceci sur des modèles réluctants (générés par l'outil RelucTool en version dynamique) [GAL-08].

Plusieurs autres plug'in ont été réalisés en vue de la simulation système. On retiendra que S.Gaaloul a développé un couplage avec l'outil Dymola (simulation de modèles Modelica), appliqué à la simulation énergétique dans les bâtiments [GAA-11] [GAA-12]

Le principe de cet interfaçage plug-in (Figure 3-5) consiste à prendre le composant logiciel ICAr généré par n'importe quel outil, puis à l'encapsuler dans une «couche logicielle» adaptée pour qu'il soit accepté par le simulateur considéré. Cette méthode est faisable si la formalisation des modèles dans le simulateur accepte l'appel des programmes compilés (modèle boîte noire) en « byte code » (Java) ou en code natif (C, C++).

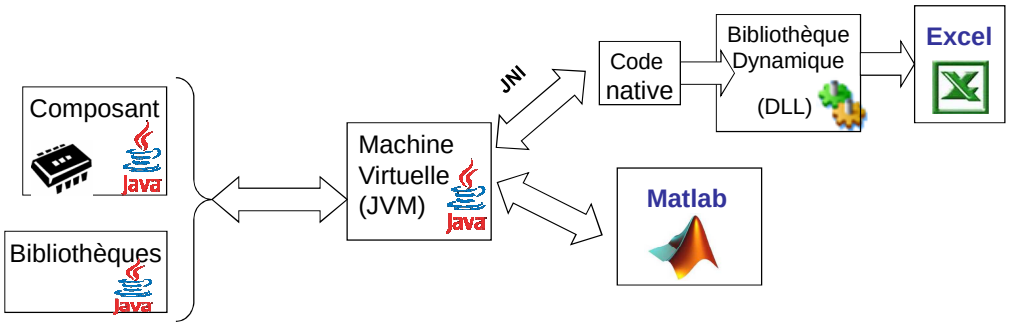


Figure 3-5 : Principe de l'interfaçage composant logiciel et outils externes

Un « plug-in » de composant logiciel ICAr nécessite la prise en charge et l'utilisation du composant logiciel à travers une machine virtuelle JAVA parce que le composant lui-même est développé dans ce langage. En conséquence, si les outils de simulation extérieurs possèdent une machine virtuelle Java (tel que Matlab/Simulink), l'appel des services et des fonctions du composant se fait d'une manière directe. Pour les logiciels qui ne possèdent pas une machine virtuelle Java (AMESim ou Dymola), un passage intermédiaire est nécessaire. Ce passage peut être réalisé par le biais d'une implémentation de la JNI[60] (Java Native Interface) qui permet l'échange entre des programmes en langage natif (compilé via des langages tels que C ou Fortran) et ceux en Java sous Windows. Ce programme est ensuite compilé dans une DLL (Dynamic Link Library) dont les fonctions peuvent être appelées facilement depuis un autre logiciel à travers le langage de programmation ou de modélisation qu'offre le logiciel (C, VBA, VHDL-AMS, Modelica, etc).

Dans nos travaux, nous allons appliquer cette approche pour des simulateurs systèmes de type « circuits » préférant des langages et des implémentations adoptant l'approche nodale.

[60] http ://java.sun.com/docs/books/jni/

A.2.2 Particularité des simulateurs systèmes étudiés

Les outils que nous utilisons dans le cadre de nos travaux sont essentiellement SMASH, Portunus et Simplorer. Ces outils offrent la possibilité de faire cohabiter :

- différents formats de composants définis par des langages natifs (C/C++), des langages propriétaires (SML de Simplorer), des standards descriptifs (VHDL-AMS)
- des couplages inter-logiciels (Portunus/Flux, Simplorer/Simulink, …) au sein du même schéma défini graphiquement (Portunus) ou codés textuellement (SMASH).

Nous cherchons à voir comment nous pouvons intégrer les composants logiciels dans un tel outil. Pour cela, nous avons effectué une étude sur l'interopérabilité entre les outils de simulations et un composant logiciel ICAr.

Dans le cadre du projet MoCoSyMec, nous nous intéresserons principalement au langage VHDL-AMS. Nous avons étudié en premier lieu les possibilités d'effectuer l'interfaçage des composants logiciels via ce langage. Cet interfaçage entre VHDL-AMS et les autres langages n'est pas une fonction inhérente au langage. En fait, la norme VHDL-AMS a prévu la possibilité d'appeler des fonctions externes (code compilé) depuis ce langage grâce à l'attribut «FOREIGN». Par contre, la norme ne dicte pas comment cet attribut doit être interprété et implémenté dans les logiciels de simulation, cela a été laissé à une libre interprétation pour chaque outil. Actuellement, seul le logiciel SMASH implémente cette fonctionnalité en passant par l'appel de fonctions DLL (C/Fortran…) ou directement par du code C/C++. Ceci rend la portabilité des modèles VHDL-AMS, entre les outils, extrêmement réduite.

Etant donné que les autres logiciels tels que Portunus et Simplorer n'offrent pas cette possibilité, leur interfaçage avec le composant logiciel doit être effectué à travers les autres langages qu'ils proposent. Chacun d'eux proposent une «interface C» de développement de modèles qui est presque identique (basée sur le même principe de résolution), ce qui a guidé notre choix vers cette solution.

Il en découle donc deux alternatives possibles de couplage dans ces outils, que nous allons étudier par la suite :

- l'interfaçage par VHDL-AMS via des fonctions externes à travers « FOREIGN »
- l'interfaçage par l'environnement de simulation via les interfaces C des outils.

Dans les deux cas, nous allons développer une interface C permettant de piloter le composant de calcul. L'étude et la mise en œuvre seront présentées par la suite.

B. Mise en œuvre des plug-ins via le langage VHDL-AMS (SMASH)

B.1. Le processus d'échange

Comme nous l'avons présenté précédemment, l'approche que nous proposons pour échanger des modèles entre deux outils de CAO (un outil de modélisation générant un composant logiciel et un outil de simulation système) consiste au pilotage des modèles en s'appuyant sur le langage VHDL-AMS [REZ-11a]. La Figure 3-6 montre l'ensemble du processus qui se déroule en quatre étapes :

(1) créer le composant logiciel, entité indépendante modélisant un système physique par un code de calcul à l'aide des générateurs dédiés,
(2) réaliser l'interface de communication «DATA Interface» entre le composant logiciel et un langage natif (C/C++) à travers la JNI (Java Native Interface),
(3) réaliser l'interface de communication «VHDL-AMS Adapter » entre les modèles VHDL-AMS et un langage natif (C/C++),
(4) importer le modèle dans le simulateur et le coupler avec d'autres systèmes.

Cette approche nécessite la mise en œuvre des interfaces de communications «DATA Interface» et «VHDL-AMS Adapter ».

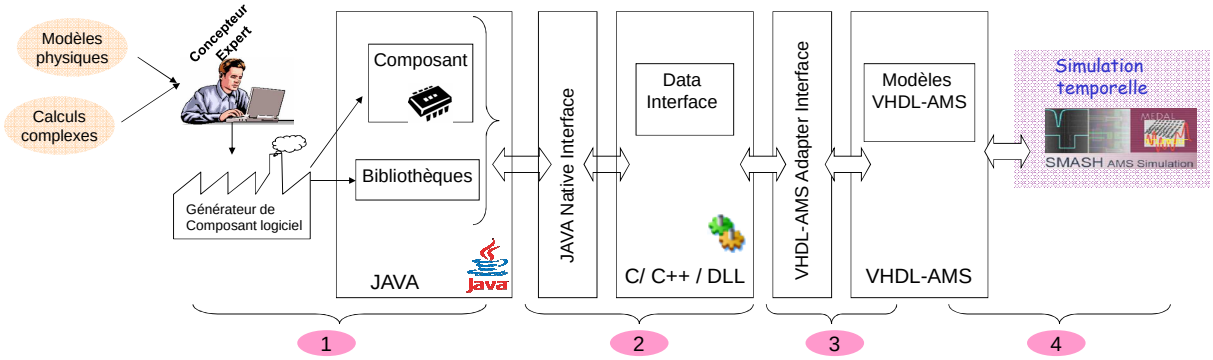


Figure 3-6 : Couplage entre le modèle VHDL AMS et le composant ICar dans SMASH

B.2. Mécanisme de l'interface « VHDL-AMS – Langage Natif»

B.2.1 Attribut « FOREIGN »

Il existe des langages de programmation plus adaptés que VHDL-AMS permettant de développer les algorithmes de traitement appropriés, par exemple les équations aux dérivées partielles. Pour ces cas, VHDL-AMS définit l'attribut « *FOREIGN* » dans son package « *STANDARD* »pour assurer la cohabitation des différents modèles multi-langages. Cet attribut est utilisé pour compléter les architectures ou les sous-programmes en VHDL-AMS. La Figure 3-7 montre l'utilisation de l'attribut « FOREIGN » pour spécifier qu'un sous-programme (fonction) est mis en œuvre dans un langage autre que VHDL-AMS. La valeur de l'attribut est ainsi décrite dans la partie « *implementation-dependent information* »

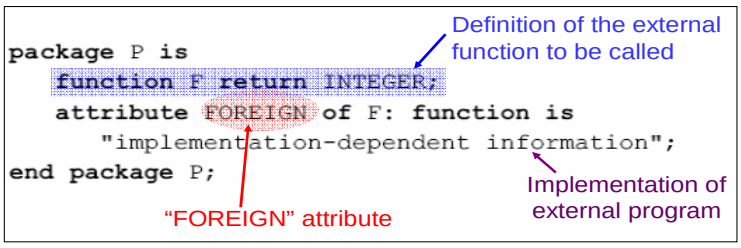


Figure 3-7 : Définition de l'attribut Foreign de VHDL-AMS [VHD-99].

La norme VHDL-AMS ne précise pas le traitement appliqué ni le mécanisme de sa mise en œuvre. L'outil SMASH implémente cette fonctionnalité et offre ainsi la possibilité d'étendre

les fonctionnalités du langage. Il autorise uniquement la création des fonctions externes décrites en langage C/C++ ou sous forme de DLL [SMA-09].

Nous allons analyser cette fonctionnalité et l'exploiter au mieux pour gérer les couplages des modèles complexes du génie électrique. A ce titre, la société Dolphin Integration a dû faire certains développements pour permettre de réaliser ce couplage.

B.2.2 Mécanisme de couplage VHDL-AMS / C

Prenons l'exemple simple de la résistance d'un fil électrique dépendant de la température et caractérisé par la loi d'Ohm. Ce modèle est simple pour que nous puissions nous focaliser sur la syntaxe du langage. Nous supposons que la variation de la résistance est fonction de la température et qu'elle est définie à partir d'une base de données obtenue par mesure interpolable par un code de calcul numérique. Le modèle VHDL-AMS va donc faire appel à un algorithme numérique via une fonction.

Dans l'exemple présenté sur la Figure 3-8, cet appel se fait à la deuxième ligne par le biais de la fonction «Interpolate» ; une « boîte noire » dont la définition est codée en langage C pour le simulateur VHDL-AMS. Ici la fonction «Interpolate» est notre fonction externe.

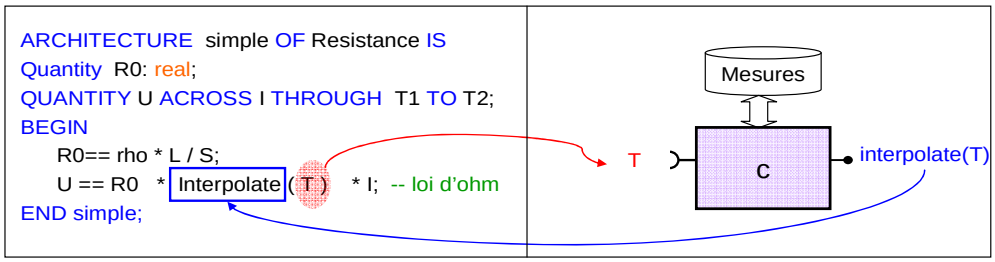


Figure 3-8 : Exemple présentant le principe de couplage de modèle

Ce que nous venons d'étudier est déjà réalisable dans l'outil SMASH. Il est assez facile d'intégrer des fonctions décrites en C dans des modèles VHDL-AMS. Dans ce cas, on n'a pas besoin de compiler le fichier de code source pour l'inclure. Il faut uniquement créer un fichier d'entête (Header) contenant le prototype de la fonction externe et le code source associés aux fonctions externes. Ensuite, un package spécial VHDL-AMS est crée ajoutant la déclaration des fonctions externes comme le montre la Figure 3-9.

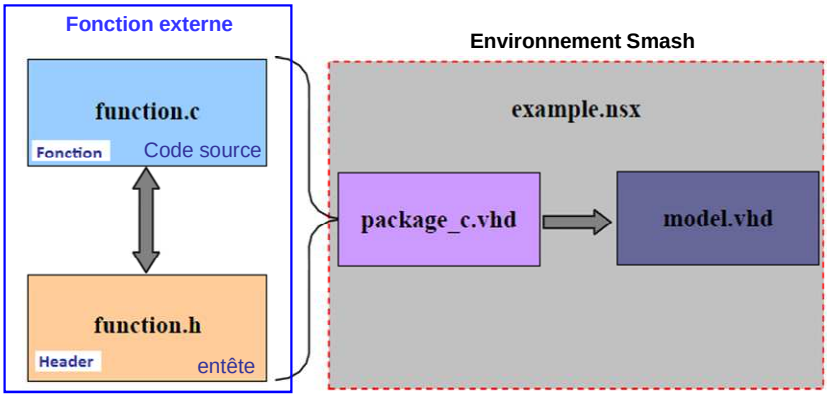


Figure 3-9 : Schéma pour inclure des fonctions C dans VHDL-AMS [SMA-09]

On note que SMASH compile les codes C fournis en même temps qu'il exécute la compilation du code VHDL-AMS. En fournissant directement une DLL, SMASH exploite les informations contenues dans cette DLL lors de la compilation du VHDL-AMS.

Les structures internes des fichiers de code source (fichier .C) et du header (fichier .h), ainsi que le package permettant de déclarer les fonctions à utiliser dans le modèle VHDL-AMS (fichier .vhd) sont présentées respectivement sur la Figure 3-10 et la Figure 3-11.

```
01: #ifndef __FUNCTION_H__
02: #define __FUNCTION_H__
03: extern "C" double Interpolate (double x);
04: #endif /* __FUNCTION_H__ */
```

(a)Function.h

```
01: #include <math.h>
02: #include "smash_api.h"
03: __declspec(dllexport)
04: double Interpolate(double x){
05: ……
07: }
```

(b) Function.c

Figure 3-10 : Codes C : (a) header, (b) source

Le fichier *package.vhd* est un package VHDL-AMS définissant la déclaration des fonctions externes à utiliser. Dans notre exemple sur la Figure 3-11, ce package déclare une fonction VHDL-AMS « INTERPOLATE» à la ligne 3. L'attribut «FOREIGN» lui associe la mise en œuvre C pour cette fonction. La valeur de cet attribut est composée des noms des fichiers source (function.c) et entêtes (function.h), suivis par le nom de la fonction «Interpolate», qui implémente la fonction externe de VHDL-AMS.

```
01: //Package.VHD
02: package FUNCTION_C is
03: function INTERPOLATE(X : in REAL) return REAL;
04:       attribute FOREIGN of INTERPOLATE :function is
05:       "function.c,function.h: INTERPOLATE ";
06: end FUNCTION_C;
```

Figure 3-11 : Code source du package VHDL-AMS

B.3. Mécanisme de l'interface « composant logiciel – langage natif »

B.3.1 Le composant «ICAr»

Le composant ICAr a été développé initialement pour des besoins de dimensionnement, et ensuite étendu à la simulation dynamique. L'utilisateur a la possibilité de choisir les facettes appropriées à son besoin. Chaque facette détermine une fonctionnalité précise permettant à partir d'entrées de fournir des sorties spécifiques. Pour illustrer ceci, considérons la facette baptisée «ModelSolver » (Figure 3-12) pour le calcul d'un modèle statique. Celui-ci implémente le calcul reliant les entrées aux sorties par le biais du modèle mathématique.

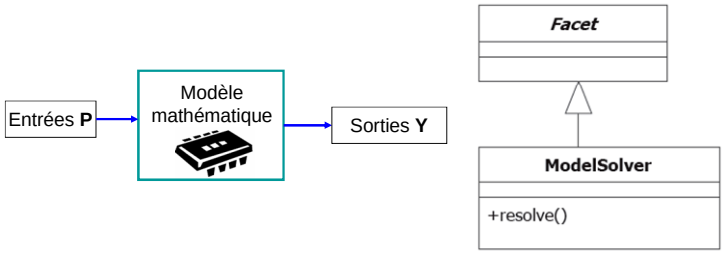


Figure 3-12 : Modèle et facette de calcul statique

Ensuite, pour faire face à la modélisation physique des systèmes dynamiques, plusieurs services de simulation ont été ajoutés dans les composants ICAr. L'approche adoptée repose sur une représentation ODE hybride (voir chapitre 2), avec des variables d'état de natures différente : continues, discrètes et avec des aspects événementiels [DO-10]. La Figure 3-13 présente sa structure.

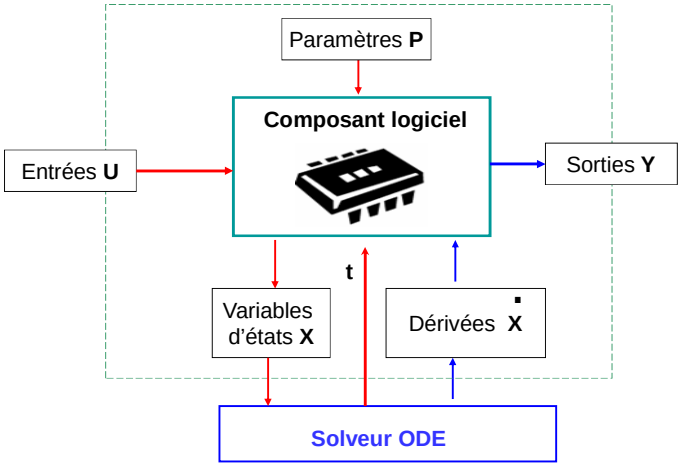


Figure 3-13 : Structure globale d'un composant pour la simulation [DUP-06] [DO-10]

B.3.2 Mécanisme de couplage ICar / C

Pour qu'un programme puisse « appeler » un composant ICAr (Java), il faut qu'il arrive à l'interpréter via ses différentes facettes. La solution proposée est d'utiliser les spécifications de l'interface JNI. C'est une interface puissante permettant l'échange entre des applications Java et des applications natives, comme le montre la Figure 3-14.

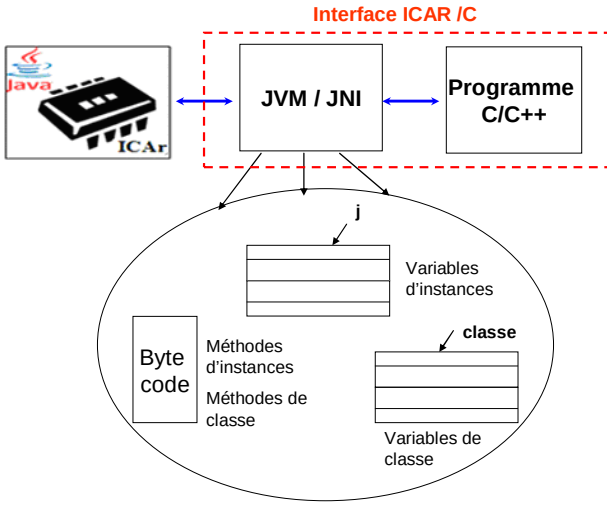


Figure 3-14 : Échanges entre le programme C, la JNI et le composant ICar

L'interface ICAr/C analyse le composant ICAr afin de déterminer ses facettes, ses différentes entrées et sorties et les différentes méthodes le constituant. Le nombre de fonctions contenues dans cette interface est maintenu à un minimum de manière à faciliter son utilisation externe. Elle contient donc des fonctions communes pour tous les composants et des fonctions spécifiques à la facette prise en charge lors de la simulation ou du calcul.

B.3.3 Mise en œuvre de l'interface ICAr / C

Les étapes de la mise en œuvre sont les suivants :

(1) Une partie commune pour tous les composants ICAr

Ce bloc définit les différentes fonctions relatives au lancement de la JVM *launchJVM()*, les routines d'adaptation via la JNI, le chargement du composant ICAr *loadComponent(path*) et l'introspection des interfaces et des méthodes permettant de récupérer les entrées et des sorties. Les valeurs sont ensuite affectées par la méthode *setInput(nom,valeur*) du composant.

(2) Le calcul des ICAr pour des modèles de dimensionnement ou modèles statiques

Pour les composants statiques, le corps de calcul est extrait suite à l'analyse de la facette *ModelSolver*. L'appel de la méthode *resolve()* permet de calculer les sorties à partir d'entrées selon l'algorithme implémenté (Figure 3-15). Les valeurs de sorties sont ensuite récupérées en appelant la méthode *getOutput()*. Cela s'apparente à une fonction à plusieurs entrées et à plusieurs sorties calculées à partir des entrées sans différentiation de celles-ci.

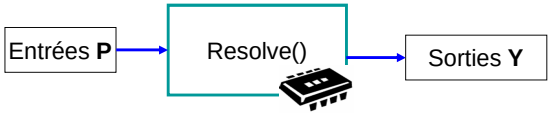


Figure 3-15 : Le calcul des modèles ICAr statiques

(3) Calcul des ICAr pour des modèles dynamiques de types ODE

Une particularité des composants supportant des modèles dynamiques consiste en un typage de certaines entrées et sorties (voir chapitre 2). Cela concerne par exemple la récupération des noms et du nombre des variables d'états définies comme des entrées du modèle. La description du comportement du modèle et les relations entre les variables sont donnés en interne dans le composant (Figure 3-16).

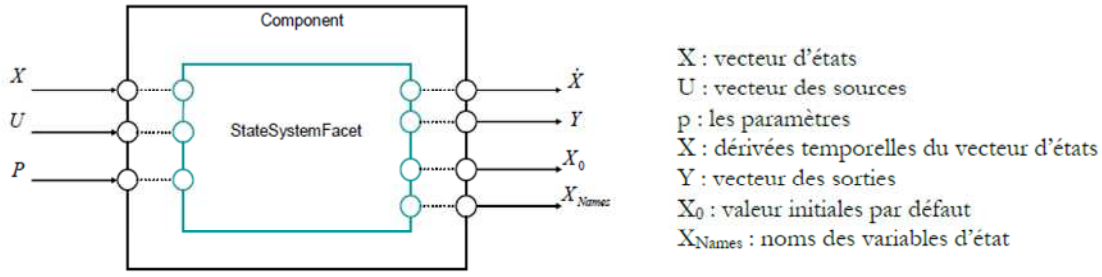


Figure 3-16 : Structure d'un composant pour la simulation d'ODE [DO-10]

Ces relations sont définies à travers une séquence bien définie (figure 3-17) :

- définir les conditions initiales, notamment celles des variables d'états : initialize()
- calculer les dérivées temporelles des variables d'états : computeDerivatives(time)
- calculer les sorties du modèle d'état : computeContinuousOutputs(time)

Dans le cas d'un composant ICAr DAE que nous avons spécifié dans le chapitre 2, nous avons défini le service de calcul par le calcul du résidu des équations à partir des variables d'état et des dérivées des variables d'états.

Nous avons mis l'accent ici sur les composants ICAr de type ODE plutôt que les DAE (plus adaptés aux solveurs nodaux). Ce choix a été fait puisque les composants ICAr DAE n'était pas disponibles au début de notre thèse (nous les avons développé par la suite, cf.

chapitre 2). Nous avons donc développé les interfaces de communications vers des composants ODE car on dispose des outils tels que Reluctool permettent de les générer contrairement aux composants DAE.

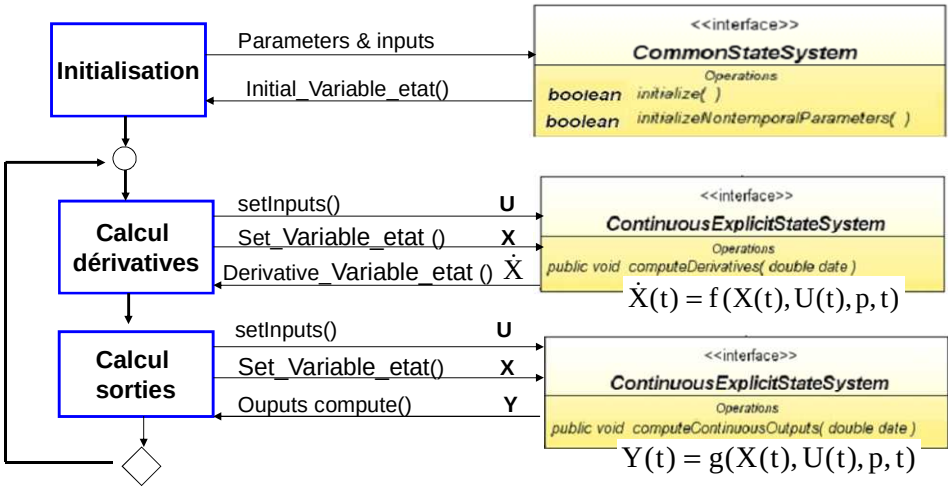


Figure 3-17 : Interactions entre le code C et le composant ICAr

B.3.4 Interfaçage « composant logiciel ICAr – VHDL-AMS »

En regroupant les deux interfaces génériques réalisées précédemment, nous sommes à présent en mesure d'interfacer un modèle VHDL-AMS avec un composant ICAr. Le composant ICAr est utilisé comme un noyau de calcul qui réalise certaines fonctions, par exemple la résolution des équations aux dérivées partielles non autorisée directement en VHDL-AMS. Cette stratégie repose sur l'utilisation d'un code natif (C/C++) permettant d'effectuer cet interfaçage. Nous en présentons la stratégie globale sur la Figure 3-18.

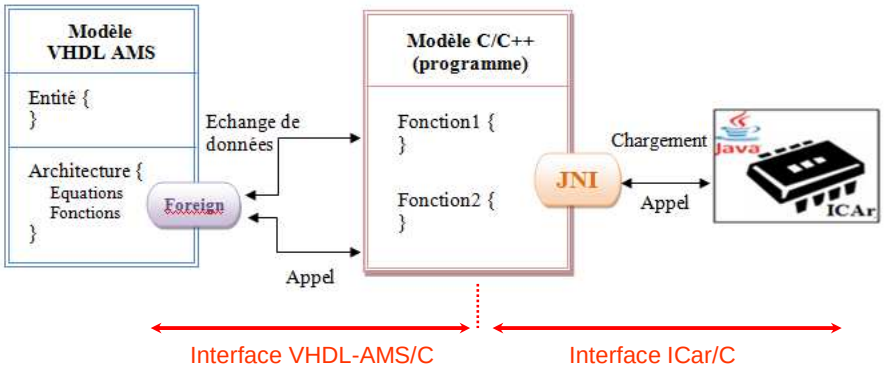


Figure 3-18 : Couplage entre le modèle VHDL AMS et le composant ICAr

La première interface ICAr/C (codée en C/DLL) prend en charge n'importe quel composant logiciel ICAr, l'analyse et détermine ses facettes et ses différentes entrées/sorties. Elle génère automatiquement les fonctions nécessaires à son exploitation dans un code en C. Ce code respecte le format imposé par SMSAH (comme expliqué au paragraphe B.3).

Quant à elle, la deuxième interface VHDL-AMS/C récupère ce code (C/DLL), génère le fichier entête correspondant et le package .VHD (comme expliqué au paragraphe B.3). Chaque fonction C compilée dans une bibliothèque peut être appelée ensuite par les fonctions

C externes de VHDL-AMS. Enfin, un squelette du modèle VHDL-AMS pourra être généré automatiquement et il ne reste qu'à remplir le contenu de l'entité et de l'architecture.

Cette approche reste valable pour tout simulateur offrant des mécanismes d'interfaçage avec l'attribut « *Foreign* » via des interfaces C/C++, ce qui n'est pas encore développé dans tous les simulateurs. Certes, certaines modifications seront à apporter essentiellement à l'interface VHDL-AMS/C selon l'implémentation du mécanisme dans le simulateur, puisqu'il n'est pas normalisé. Cette restriction limitera la portabilité des modèles couplés.

C. Mise en œuvre des plug-ins via des interfaces des simulateurs

La deuxième approche possible pour échanger les modèles est de passer par les interfaces ou les fonctions d'importation de l'outil de simulation. Les outils proposent diverses ressources aux utilisateurs les permettant de définir de nouveaux modèles utilisateurs :

(1) pour créer leurs propres modèles de composants (description mathématique),

(2) pour gérer des algorithmes de contrôle ou l'échange avec des outils externes.

Les outils de simulation système proposent diverses ressources concernant la description et la simulation des composants à l'aide des modèles mathématiques spécifiques. Cette description est dépendante d'une structuration des modèles. Par exemple, ils permettant leur mise en œuvre à travers une interface de programmation qui leur est propre via le langage C. Ces interfaces permettent de décrire le comportement des composants ou des systèmes par un ensemble d'équations sous forme de DAE non linéaires principalement (chapitre 2). Les outils fournissent aussi, pour la représentation des relations plus complexes, un ensemble de fonctions de modélisation adaptées et divers moyens pour contrôler et influencer le comportement du simulateur et les algorithmes internes de résolution.

Portunus[61] et Simplorer[62] offrent cette possibilité de créer de nouveaux modèles en utilisant le langage de programmation C/C++. Ces modèles sont traités uniquement par le solveur continu ; il n'est donc pas possible de caractériser un modèle de machine d'état. Le fonctionnement de cette interface est similaire pour les deux logiciels. Etant dans le cadre du projet MoCoSyMec, nous allons nous intéresser à l'outil Portunus et voir comment nous

Nous avons exploité l'approche de couplage que nous avons proposé dans le paragraphe précédent. Des modifications sur le code C ont été apportées afin de respecter le format imposé par l'interface C de Portunus. Le mécanisme d'interfaçage mis en œuvre est présenté sur la Figure 3-19 (les fonctions utilisées sont présentées dans le Tableau 3-1).

C.1. Spécificité de l'interface C de Portunus

Comme pour les autres simulateurs de type «circuit», l'approche retenue pour la simulation dynamique dans Portunus est l'approche nodale (chapitre 2). Portunus définit un objet issu d'une classe C++ afin de rendre possible la communication entre les modèles décrits en C et le simulateur. Cet objet a pour rôle de sauvegarder toutes les informations sur un modèle. Différentes fonctions virtuelles ont été définies exploitant cet objet.

[61] Adapted Solution, "Using the Portunus C-Programming Interface",
[62] Ansoft Simplorer, "Using the Simplorer C-Programming Interface"

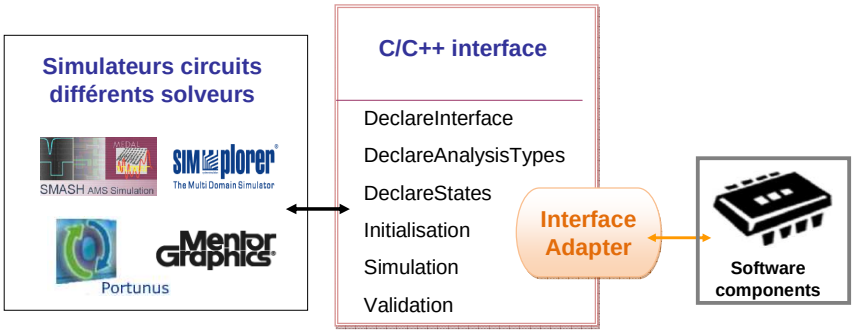


Figure 3-19 : Mécanisme d'interfaçage du composant ICAr avec Portunus

Les trois fonctions « DeclareName, DeclareInterface et Simulation » sont obligatoires. Le Tableau 3-1 les présente brièvement. Des détails supplémentaires sur le fonctionnement de cette interface sont fournis dans [ADA-07] avec des exemples illustratifs.

« DeclareName »	Elle permet de spécifier le nom du modèle.
« DeclareInterface »	Elle permet de spécifier l'interface de connexion avec l'extérieur (terminaux, paramètres, résultats de sorties). Elle est appelée au début d'une simulation.
« DeclareAnalysis Types »	Elle permet de définir les différents types d'analyses souhaitées (TR, AC, OP et DC). Elle est appelée au début d'une simulation
« DeclareStates »	Elle permet de définir les équations d'état supplémentaires qui sont ajoutées afin de résoudre le système complet. Chaque équation ajoutée crée une nouvelle ligne dans la matrice du système global.
« Initialisation »	Elle est appelée avant le début de la première itération, pour le réglage des valeurs initiales, le calcul des dérivées des paramètres ainsi que le réglage des entrées constantes dans la matrice jacobienne.
« Simulation »	Elle sert à calculer les entrées, la matrice jacobienne, puis les valeurs des sorties qui doivent être fournies par le modèle.

Tableau 3-1 : Les fonctions exploitant un objet externe dans Portunus

C.2. Plug-in du composant ICAr dans Portunus

Via des adaptations nécessaires, l'interface C de Portunus nous permet de charger un composant ICAr en s'appuyant sur le couplage C/ICar présenté dans le paragraphe précédent. Ce premier couplage permet de spécifier les paramètres, les entrées et les sorties du composant. Ensuite, l'interface permet de récupérer les résultats de résolution dépendant de sa nature : statique ou dynamique (voir paragraphe précédent). Enfin, nous avons développé le couplage entre les services proposés par un composant ICAr et la forme du modèle spécifique pour la simulation sous Portunus, comme l'illustre la Figure 3-20.

Nous nous sommes confrontés ici à la problématique de compatibilité du composant ICAr dans leur version ODE avec l'approche nodale implicite (DAE), comme nous l'avons signalé

dans le chapitre 2. En effet, le composant ICAr, dans sa version dynamique ODE décrit des modèles causaux et orientés, avec des notions d'entrées/sorties spécifiques ; or le simulateur Portunus adopte la résolution implicite des équations DAE (approche nodale).

Ainsi, nous avons fait évoluer les composants logiciels ICAr pour les adopter à cette approche de résolution et les rendre ainsi compatible avec l'approche nodale, acausale implicite, que nous avons présentée dans le chapitre 2.

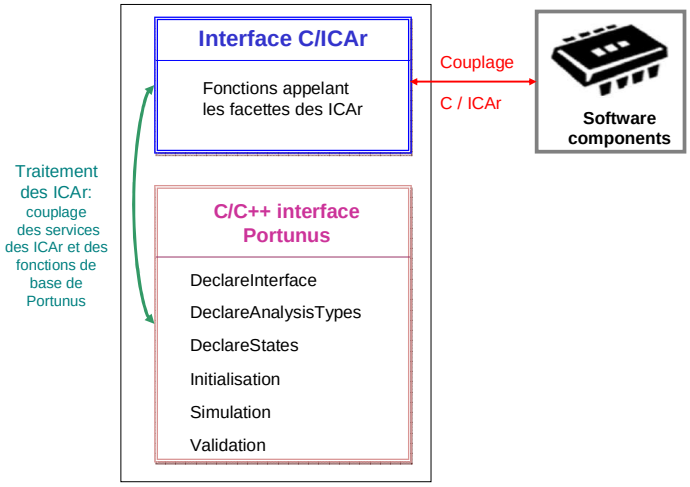


Figure 3-20 : Composant résultant de l'interfaçage d'un ICAr avec Portunus

En ce qui concerne le traitement des composants ICAr ODE, nous devons proposer une reformulation pour rendre le couplage faisable avec Portunus (gérer les ODE en tant que des équations implicites). Il nous est indispensable de fournir le calcul de sa matrice jacobienne.

Pour réaliser ces adaptations de format, des manipulations symboliques peuvent être introduites lors de la réalisation de l'interface de couplage C/ICAr afin de:

- transformer les ODE en DAE
- automatiser le remplissage du jacobien à partir du modèle décrit dans un ICAr.

Une fois le modèle réalisé et importé dans Portunus, le composant encapsulant l'ICAr peut être connecté à d'autres modèles via la schématique de Portunus.

A noter que le traitement des composants ICAr DAE dans l'interface C de Portunus revient à une reformulation syntaxique, alors que celui des composants ICAr ODE est une reformulation sémantique (manque d'information dans le composant ICAr, tel que le calcul de jacobian), comme l'illustre la Figure 3-21.

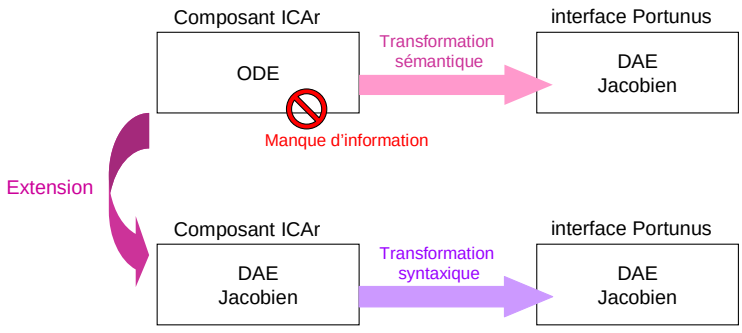


Figure 3-21 : Transformations des ICAr pour être compatibles avec l'interface C

Notons bien que cette stratégie est indépendante de l'outil Portunus. Ceci reste valable pour tout autre outil que Portunus, disposant d'une interface externe de communication et permettant le chargement d'un modèle externe. Les fonctions que nous avons présentées dans le Tableau 3-1 permettant de mettre les informations dans le bon format et sont nécessaires pour que le solveur du simulateur puisse intégrer temporellement le modèle. Chaque fonction constitue tout un processus de résolution qui est masqué pour l'utilisateur.

D. Applications

Idéalement, les applicatifs de nos travaux consistent à fournir des modèles réutilisables en VHDL-AMS pour des systèmes magnétiques issus de ces différentes modélisations. Chacun des modèles VHDL-AMS pourra contenir différents niveaux de modélisation :

- un schéma électrique équivalent (RelucTool)
- un schéma de composition pour les microsystèmes (MacMMems)
- un modèle léger décrit par surface de réponse (FGot).
- un modèle de co-simulation ou de pilotage (Flux) pour prendre en compte la saturation ou les effets dynamiques

Les outils RelucTool, MacMMems et Fgot permettent de créer des modèles compatibles avec la norme ICAr. La solution de couplage faible via le langage VHDL-AMS est une approche intéressante pour les utilisateurs, permettant d'utiliser les modèles ICAr générés dans différents simulateurs ou réutiliser des modèles existants sans avoir à les modifier ou les réécrire manuellement.

La Figure 3-22 présente l'état actuel du bus à composants logiciels ICAr, développé au laboratoire G2Elab, assurant l'interopérabilité entre différents outils et entre différents modèles issus d'autres langages ou d'autres composants logiciels.

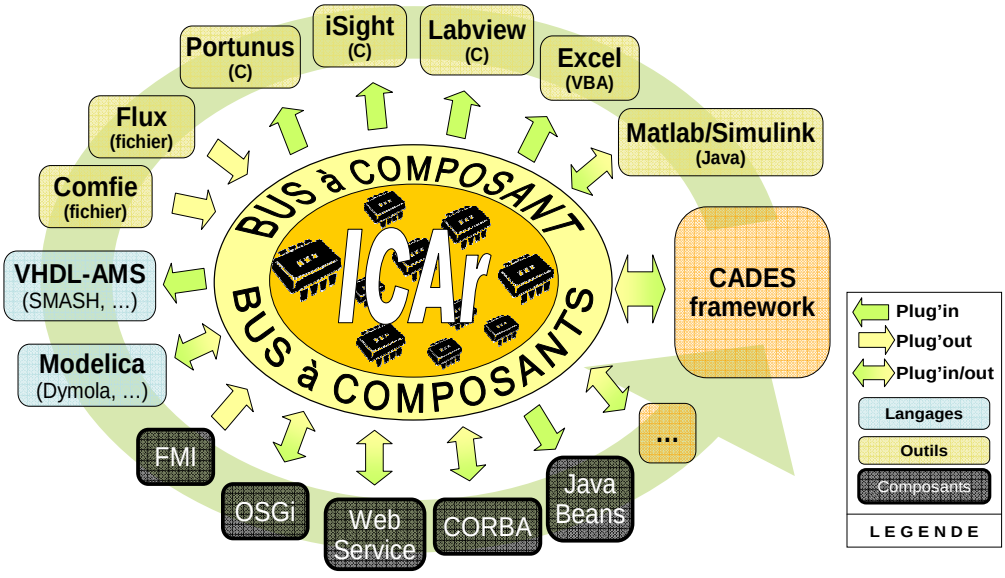


Figure 3-22 : Bus logiciel à base des ICAr dans un contexte d'interopérabilité [DEL-11]

Afin d'illustrer la nécessité d'utiliser des composants logiciels issus des différents outils dans une simulation système, nous allons mettre en œuvre le couplage du modèle de

déclencheur électromagnétique (chapitre 1) avec l'outil SMASH en fournissant une description VHDL-AMS complète du modèle du point de vue utilisateur.

En ce qui concerne l'outil Flux2D/3D, la co-simulation Flux2D/3D avec l'outil Portunus a été réalisée par Cedrat en même temps que nos travaux de thèse [DUP-08] [ADA-09], mais elle ne répond pas exactement à nos besoins car elle exige un modèle dynamique de Flux. Nous aborderons cette difficulté par la suite et la solution que nous proposons en nous basant aussi sur l'approche de couplage de modèles.

D.1. Pilotage de Flux2D comme serveur de calcul par VHDL-AMS

Nous cherchons ici à permettre l'appel d'un modèle éléments finis (FEM) par le simulateur système de type Portunus ou SMASH. Ceci est nécessaire pour le concepteur afin de pouvoir modéliser finement certains aspects de son système. Par exemple, l'inductance d'un déclencheur électromagnétique est calculée en grande partie par les flux de fuite qu'il est difficile d'évaluer autrement que numériquement (par éléments finis). Ainsi, dans la modélisation du système, la partie électrique (nécessitant la valeur de l'inductance L) sera dépendante de la précision des calculs effectués par la partie magnétique. Cet exemple montre que, dans le processus de conception, on ne peut pas se passer d'outils spécifiques de modélisation fine. Nous avons cité l'exemple de l'électromagnétisme, mais la problématique est la même pour toutes les autres disciplines scientifiques. Il faut donc que les outils de simulation des systèmes mécatroniques soient capables d'appeler de tels outils ou d'interagir avec eux. Autrement dit, le concepteur doit être en mesure de décrire un composant non plus par ces équations mais par un calcul effectué dans un logiciel extérieur.

Les caractéristiques des composants magnétiques FEM peuvent influencer la performance et le comportement du système global lorsqu'ils sont intégrés ensemble. La co-simulation permet donc d'étudier l'influence des composants modélisés par éléments finis sur les autres et d'évaluer la réponse globale du système pour l'améliorer. La co-simulation Flux-Portunus a été réalisée par Cedrat dans le projet MoCoSyMec [DUP-08]. Cette co-simulation exige que le modèle FEM soit lui aussi dynamique. Le pilotage de FLUX2D en tant que serveur de calcul dans le cas de simulations quasi-statiques (dynamique mécanique et calcul magnétique statique) nous a paru nécessaire. Le but de cette application est de coupler les logiciels systèmes Portunus ou SMASH avec le logiciel éléments finis Flux3D pour réaliser une simulation quasi-statiques globale, en exploitant les approches plug-in et plug-out que nous venons de développer. Nous avons donc réaliser cela v

Nous appliquons cela sur l'exemple d'un contacteur électromagnétique. Nous avons associé un modèle FEM au système magnétique et des modèles décrits en VHDL-AMS pour les parties mécanique et électrique [REZ-11]. Le logiciel FLUX2D est utilisé pour le calcul de la force magnétique et de la valeur de l'inductance (Figure 3-23).

Dans la 1ère approche (paragraphe C), Flux2D est contrôlé et relié au modèle VHDL-AMS grâce à la fonction externe "FOREIGN" dans le simulateur SMASH. Dans la 2ème approche (paragraphe D), il est relié au simulateur Portunus grâce à son interface C propriétaire. Pour les deux approches, le logiciel Flux2D est utilisé comme un serveur de calcul.

Pour pouvoir contrôler l'outil Flux2D (serveur) par un modèle VHDL-AMS ou par un simulateur système (client), nous avons proposé une stratégie permettant au client de générer un jeu de paramètres d'entrées et de commander le calcul dans le serveur et de récupérer les résultats. Ce dialogue entre les deux outils se fait à l'aide d'un protocole de communication basé sur la présence d'un répertoire d'échange permettant de se synchroniser et d'échanger des données. La Figure 3-24 présente ce protocole d'échange développé pour ce cadre.

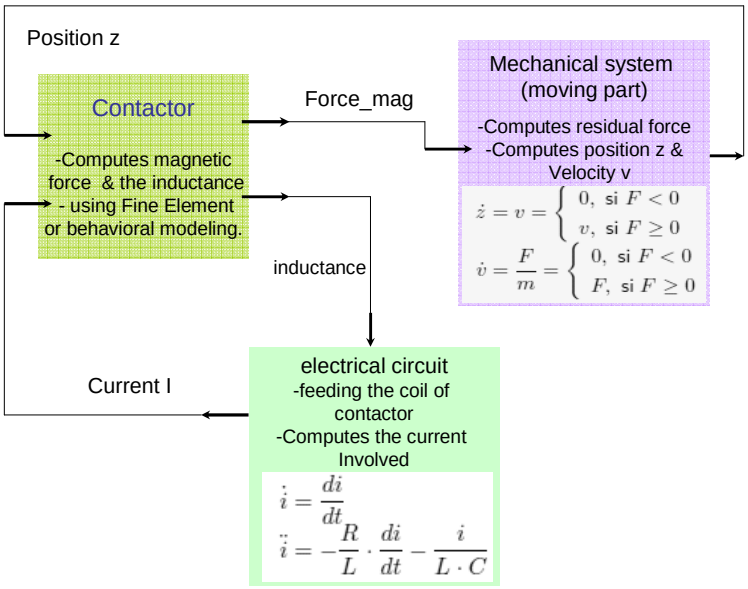


Figure 3-23 : Partitionnement du modèle dynamique du contacteur

En effet, le protocole client-serveur est géré par le client (simulateur système). Le serveur de calcul FLUX2D scrute le répertoire d'échange en attendant de nouveaux jeux de paramètres générés par le client. Dès qu'il y a présence de ces valeurs, Flux2D devient actif (en tant que serveur) et se met à exécuter une série de tâches menant au calcul des sorties attendues par le client. Une fois ces calculs terminés, FLUX2D repasse ces résultats au simulateur client et se remet en mode attente. Le simulateur réagit rapidement en récupérant ses résultats et finit son calcul. Cette opération s'effectue à chaque pas de temps.

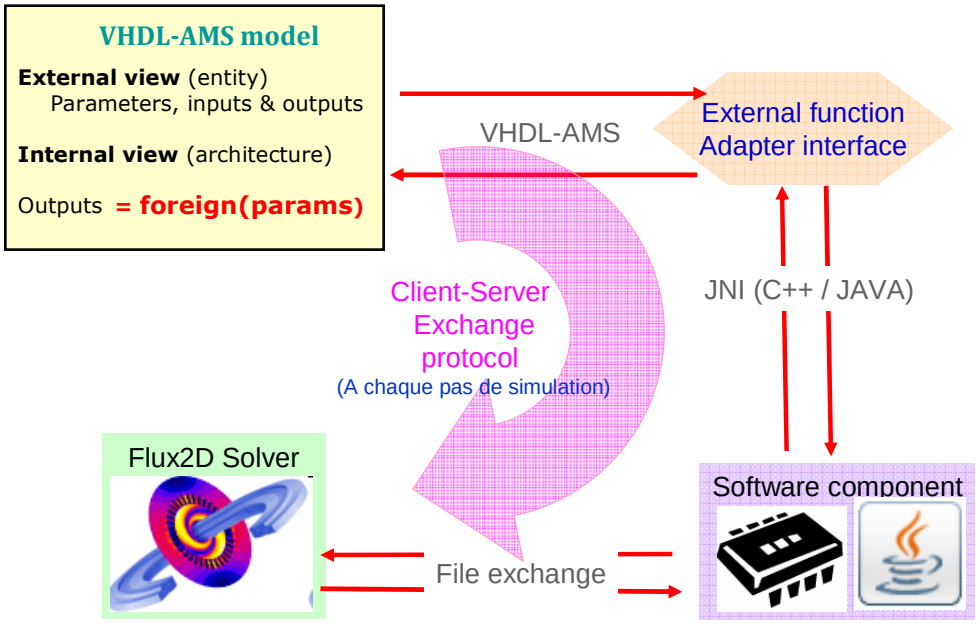


Figure 3-24 : Le protocole d'échange client-serveur contrôlé par un logiciel SMASH

Pour ce faire, nous avons développé un composant logiciel (ICAr) pour gérer la connexion entre le client et le serveur au moyen d'un protocole d'échange. Son but est de transmettre les demandes en provenance de l'un ou l'autre, par le transfert de fichiers de données. Du coté de FLUX2D, cette communication est rendue possible par l'implémentation d'un fichier de commande spécifique (en python) développé par Dr J.L. Coulomb initialement pour des objectifs de dimensionnement. Cela permet d'automatiser cet appel dans le serveur.

Ce protocole a été mis en œuvre dans l'outil SMASH, en considérant le contacteur comme un macro-modèle VHDL-AMS, appelant des fonctions externes (via les «Foreign») pour échanger les entrées et sorties avec le solveur Flux2D via le composant logiciel ICAr. Les résultats de cette implémentation sont donnés dans [REZ-11]. Ce même protocole a été aussi développé dans Portunus en suivant le couplage via l'interface C.

D.2. Intégration des modèles existants (ICAr) dans SMASH

Nous cherchons ici à importer un modèle d'actionneur généré par un outil dédié, pour l'intégrer dans un modèle système plus global. Nous reprenons l'exemple du composant ICAr définissant un déclencheur électromagnétique qui va être inclus dans le système de chaîne de coupure (chapitre 1). Supposons que le reste de la chaîne est modélisé en VHDL-AMS. Notre objectif est d'exploiter le déclencheur électromagnétique dans les outils de simulation de type Portunus et Smash, idéalement grâce au langage VHDL-AMS.

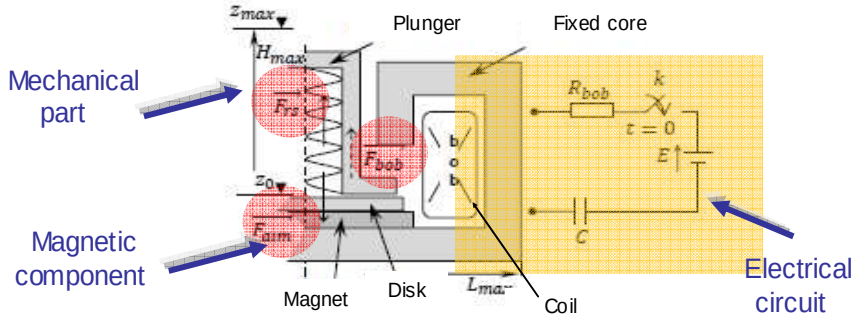


Figure 3-25 : Description simplifiée du mouvement du noyau de l'actionneur

Dans cette application, nous souhaitons faire un compromis entre la précision du modèle et sa vitesse de simulation. Nous disposons du modèle éléments finis précis du déclencheur; la solution de pilotage de Flux2D (paragraphe D.1) est envisageable, cependant elle demande un temps de calcul important. Le modèle mathématique simplifié (annexe 2-1) peut ne pas fournir la précision nécessaire pour le fonctionnement global du système. Nous adopterons donc la représentation sous forme d'un réseau de réluctances qui satisfait le compromis précision/temps de calcul sur la plage de validation du modèle global associé au système.

L'idéal est de fournir un modèle du déclencheur décrit entièrement en VHDL-AMS. Nous verrons cette alternative dans le chapitre 4. Ici, nous allons nous contenter d'exploiter l'outil RelucTool dédié à cette modélisation.

Cet outil génère à partir d'une description du réseau de réluctance associé au déclencheur un modèle sous forme de composant logiciel ICAr (Figure 3-26). C'est ce modèle que nous allons importer dans la simulation du système. Les approches de couplages proposées, via les plug-in/plug-out, présentent une solution intéressante dans ce contexte.

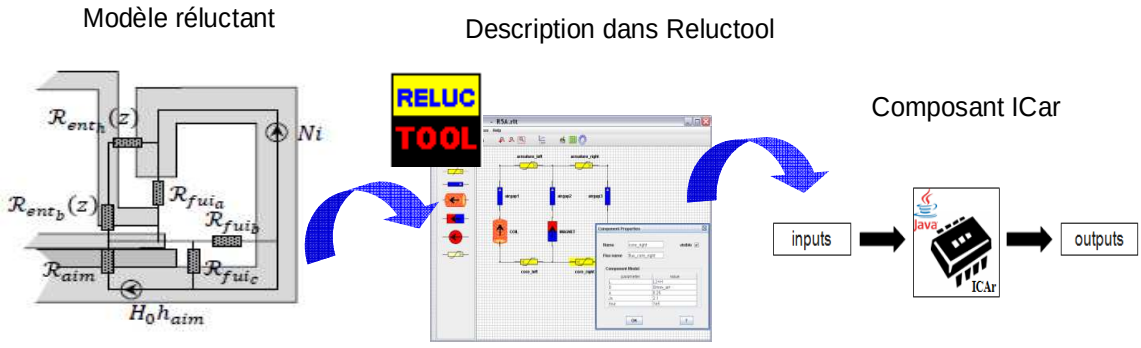


Figure 3-26 : Représentation du circuit magnétique réluctant

Tout d'abord, le composant magnétique est inséré dans un modèle VHDL-AMS grâce à la fonction externe "FOREIGN" et ensuite couplé à la chaine dans le simulateur SMASH. Le même composant est encapsulé dans un modèle C dans le simulateur Portunus grâce à son interface propriétaire. Nous retrouvons à ce stade le mécanisme présenté pour Flux2D.

La Figure 3-27 présente un extrait du modèle VHDL-AMS encapsulant le composant ICAr du composant magnétique et la Figure 3-28 le processus d'échange entre les deux modèles. Exporter le comportement magnétique du déclencheur (généré en tant que composant ICAr)

en VHDL-AMS consiste à définir une seule entité VHDL-AMS, avec son comportement dynamique représenté par une architecture faisant intervenir l'appel à des fonctions externes.

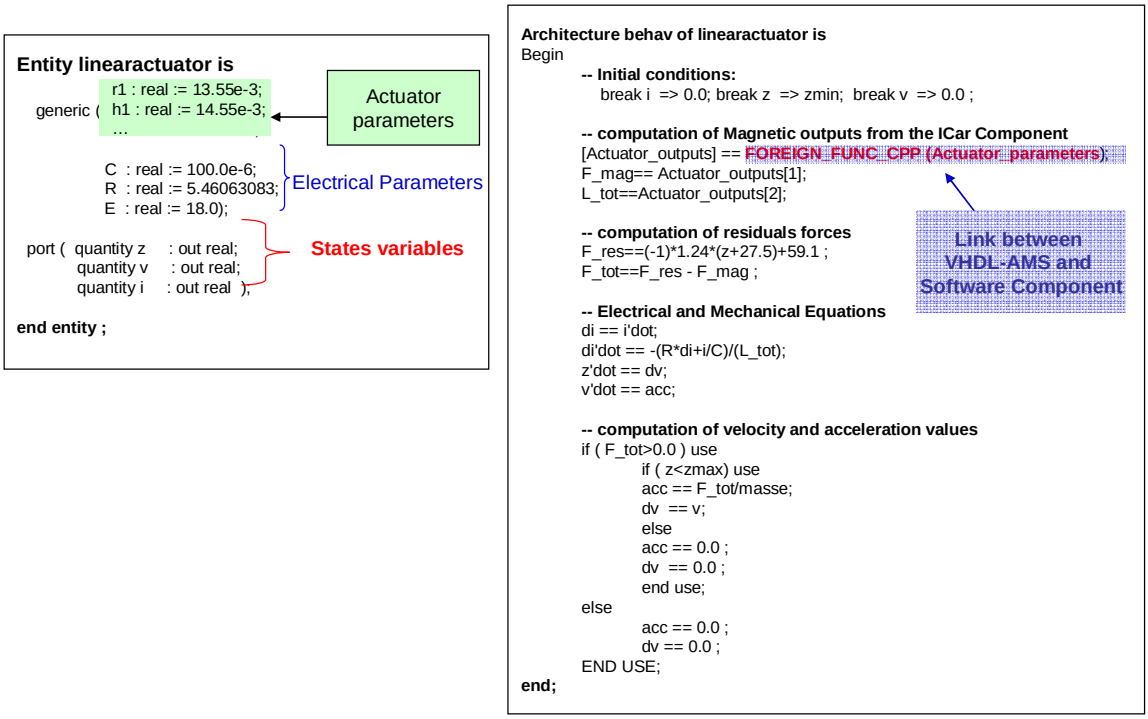


Figure 3-27 : Représentation du modèle VHDL AMS de l'actionneur avec un composant ICAr

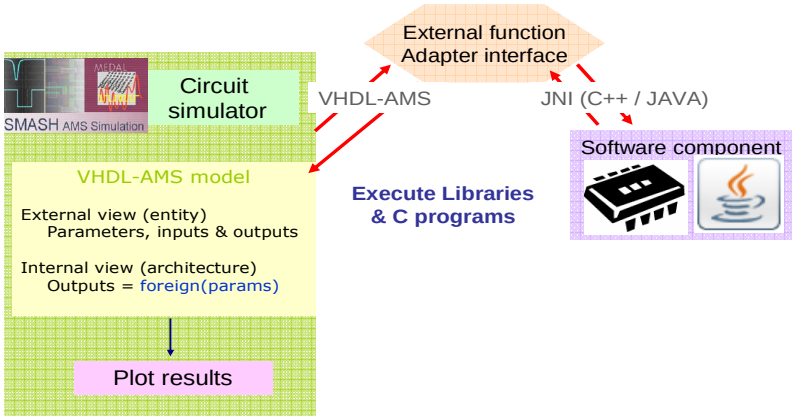


Figure 3-28 : Processus d'échange entre un modèle VHDL-AMS et un composant logiciel ICAr

Conclusion

Dans ce chapitre, nous avons montré l'importance des approches de couplages pour assurer l'interopérabilité des modèles et des outils dans le cycle de conception. Notre approche permet en particulier d'intégrer un modèle précis dans un composant logiciel spécifique. Cette approche, appelée « plug-in », a été étudiée dans le contexte de la conception des systèmes électromagnétiques. Elle a permis l'intégration des divers modèles (issus de différentes modélisations : Flux, Fgot, Reluctool et MacMMems) dans des outils de simulation globale (Portunus, Simplorer et SMASH). Nous avons donc mis en œuvre deux solutions et nous les avons validées pour diverses applications comme l'illustre la Figure 3-29.

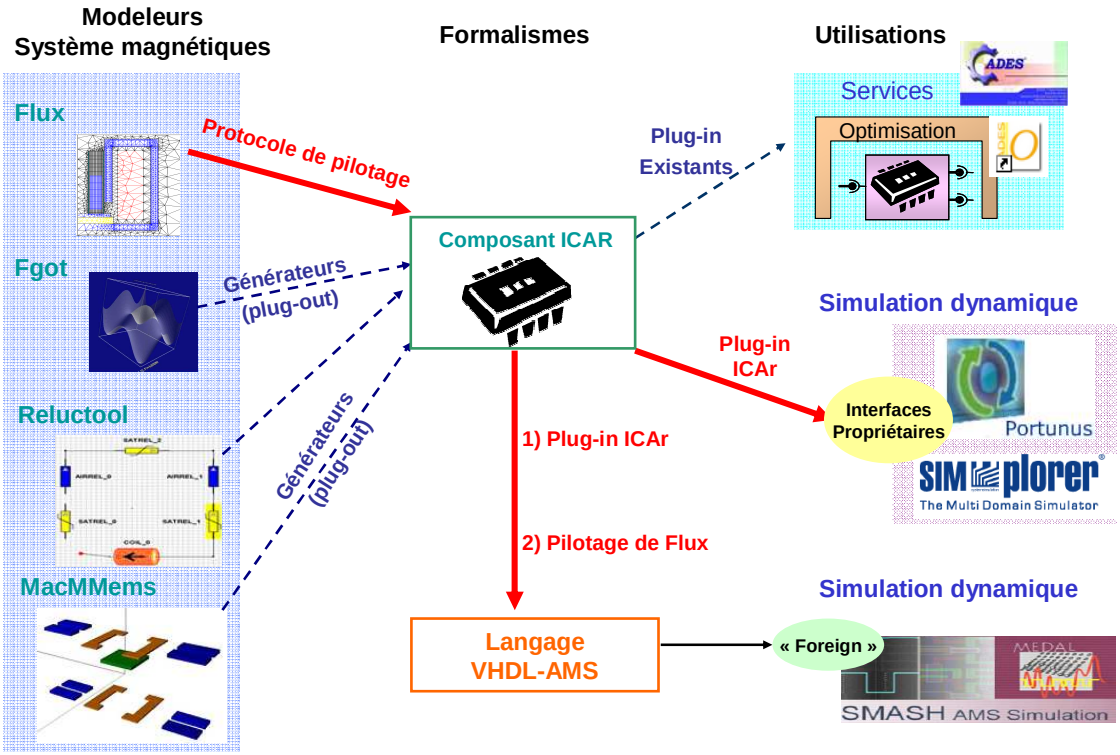


Figure 3-29 : Solutions d'interopérabilité proposées et développés dans nos travaux (en rouge)

Ces solutions d'interfaçage permettent d'importer des modèles existants sous forme de composants logiciels ICAr dans un simulateur système et/ou d'assurer une co-simulation ou un pilotage d'un outil par un autre (comme celui de Flux 2D/3D par Portunus) :

- la première solution via VHDL-AMS exploite ses capacités à appeler des fonctions externes via l'attribut « Foreign »,
- la deuxième solution est plus « propriétaire », elle fait intervenir les interfaces spécifiques des simulateurs.

Il est à noter que les solutions proposées sont applicables à n'importe quel modèle ou équations encapsulés dans un composant logiciel respectant la norme ICAr, et que l'on peut réaliser théoriquement les « plug-in » vers n'importe quel environnement désiré.

Ces solutions, malgré leurs importances, possèdent encore des freins à la portabilité et à la réutilisabilité des modèles ainsi développés. La première solution dépend de l'implémentation

de la fonctionnalité « *FOREIGN* » ; la norme ne prévoit pas les mécanismes d'interfaçage que cet attribut doit respecter et elle laisse libre choix aux simulateurs qui l'implémentent. La deuxième solution dépend des interfaces propriétaires des simulateurs. Bien que cette interopérabilité fonctionne, elle demande la mise en place de ces plug'in pour tout nouvel environnement de simulation. De plus, elle n'offre au concepteur que des composants disponibles « sur l'étagère », qu'il n'est parfois pas possible d'adapter. Enfin, ce mode de composition système peut être vue comme un « couplage faible », au sens où :

- le système d'équations global n'est pas constitué ; chaque composant se résout individuellement,
- il n'est pas possible de réorienter symboliquement le modèle en fonction de son utilisation.

Dans le chapitre suivant, nous allons répondre à cette limitation en étudiant la problématique de la génération automatique des modèles purement VHDL-AMS (sans fonction externe) depuis les outils métiers. Le langage VHDL-AMS permettra une interopérabilité par des couplages forts des modèles et assurera la réutilisation de modèles dans différents contextes (langage « boule/boîte blanche »).

CHAPITRE 4

Modélisation des réseaux de réluctances et des microsystèmes magnétiques en VHDL-AMS : Problématiques et solutions

Introduction

Afin d'améliorer l'interopérabilité entre les outils de modélisation et de simulation, nous proposons d'exporter les modèles issus des outils métiers vers un unique format standardisé. Nous nous positionnons dans une volonté de faciliter la migration des modèles existants (souvent écrits dans un langage propriétaire) vers le langage VHDL-AMS et ainsi préserver l'investissement ayant été fourni pour les obtenir.

Notre approche repose sur le paradigme de l'ingénierie dirigée par les modèles (IDM). Celle-ci vise à travailler à un niveau d'abstraction élevé puis à projeter le modèle dans un autre langage en fonction de besoins. Cette approche est très utilisée en informatique, par exemple où des modèles UML[63] définis graphiquement peuvent être automatiquement projetés vers des langages comme Java ou C++. En VHDL-AMS, l'exemple de SysML-Companion[64] peut être cité. Cela permet la projection d'une description haut niveau SysML en modèle VHDL-AMS simulable (approche top-down). Dans notre cas, les modèles seront également projetés en modèle VHDL-AMS simulable, mais à partir d'outils de modélisation des composants magnétiques (approche bottom-up). Nous mettrons l'accent sur l'adéquation du langage à modéliser ces dispositifs, ainsi que sur les solutions proposées pour contourner les limites rencontrées. Nous mettrons en œuvre cette approche principalement pour les outils **RelucTool** (modélisation par réseaux de réluctances), et **MacMMems** (modélisation par microsystèmes magnétiques).

Cette méthode d'interopérabilité par projection sera comparée à celle décrite au chapitre précédent qui exploite le pilotage de modèles « exécutables » via les composants logiciels.

[63] UML : Unifed Modeling Language
[64] http ://www.realtimeatwork.com/software/sysml-companion/

A. Ingénierie dirigée par les modèles et transformation de modèles

A.1. Description de l'approche MDA et de ses éléments

La transformation d'un modèle est une étape inévitable quand certains modèles sont écrits dans des langages de modélisations différents. Le modèle original (appelé source) va subir une transformation qui donnera lieu à un nouveau modèle (appelé cible). Ce modèle cible sera une version "enrichie", "simplifiée" ou "réécrite" du modèle source.

Au sein de l'équipe MAGE du G2Elab, depuis plusieurs décennies, la transformation de modèles est pratiquée couramment ; mais elle n'a pas été mise en regard des méthodologies et des concepts développés et formalisés pour le monde de l'informatique et l'ingénierie des modèles. Ainsi, nous souhaitons palier ce manque en positionnant ces concepts par rapport à nos solutions, en les implémentant et les contextualisant.

L'ingénierie dirigée par les modèles s'impose de plus en plus depuis la standardisation de la démarche MDA[65] (Model Driven Architecture) par l'organisme OMG[66]. Cette démarche est une application des concepts liés à l'ingénierie des modèles pour l'informatique, afin de permettre l'interopérabilité entre différents systèmes, réduire les coûts de développement et augmenter l'évolutivité (Figure 4-1).

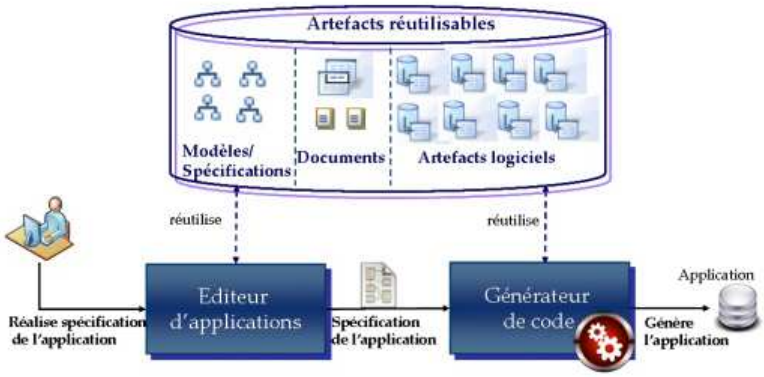


Figure 4-1 : principe MDA en génie logiciel [MAR-08]

La démarche MDA propose un cadre concret pour la modélisation de systèmes du génie logiciel, sans toutefois le restreindre à celui-ci. Il est impératif de comprendre que la MDA n'est pas une méthodologie de conception ; elle présente uniquement un cadre générique pour mettre en application l'utilisation de modèle dans un processus de développement d'un système. L'idée générale est de distinguer les aspects conceptuels (logique métier : modèles, spécification) des aspects d'implémentations (plateforme, logique d'implémentation : artefacts logiciels). De manière générale, l'ingénierie des modèles propose de mettre à disposition des outils de génération de code à partir de la description métier. Notre objectif de générer automatiquement du VHDL-AMS à partir des modèles réluctants ou des microsystèmes magnétiques, s'inscrit donc complètement dans cette démarche.

[65] Object Management Group, « MDA Guide Version 1.0.1 », omg/2003-06-01, Juin 2003.
[66] OMG (Object Management Group) : organisme international de normalisation. Son rôle est de promouvoir des standards garantissant l'interopérabilité entre applications. Il est à la base des standards tels que UML (Unified Modeling Language), MOF (Meta-Object Facility) et CORBA (Common Object Request Broker Architecture).

A.1.1 Architecture de la MDA

L'application du concept de méta-modèle (spécifications du langage de description des modèles) permet de construire des outils de modélisation capables de s'adapter à de nombreuses problématiques de résolution. Dans notre contexte, il fournit les outils d'abstraction nécessaires à la résolution des systèmes modélisés dans différents environnements de simulation. En ce sens, le langage VHDL-AMS constitue une approche MDA pour la simulation dynamique. En effet, il permet la séparation entre l'aspect métier de modélisation et l'aspect numérique de résolution des équations différentielles. Ainsi, chaque modèle VHDL-AMS (vérifiant le méta-modèle commun) peut être traité dans différents environnements de simulation basés sur ce même langage. Ces environnements réalisent par exemple la traduction automatique du modèle en langage C qui est ensuite évalué pour calculer des grandeurs numériques.

Il est classique de présenter la MDA par quatre niveaux d'abstraction (Figure 4-2) :

- **Le niveau d'exécution M0** (ou instance) : Il correspond au système concret, sous forme des objets exécutables dans une plateforme précise. Dans le cas de VHDL-AMS, il s'agit des valeurs calculées en cours d'exécution d'un modèle de simulation.
- **Le niveau modèle M1** : Il représente le modèle qui sera instancié pour être calculé ou transformé dans le but de devenir compatible avec un autre méta-modèle ; par exemple, le modèle écrit en VHDL-AMS ou en C.
- **Le niveau méta-modèle ou langage M2** : Il contient les méta-modèles des modèles du niveau M1. Il définit le langage de modélisation, la syntaxe et la grammaire de représentation de ces modèles ; par exemple, les spécifications du langage VHDL-AMS.
- **Le niveau méta-métamodèle ou méta-langage M3**. Ce niveau est le plus abstrait, il contient les méta-métamodèles : l'entité MOF (Model Object Facility) vise à décrire les méta-modèles de la couche M2. Il n'y a pas de couche M4 car le formalisme MOF s'auto décrit. Nous ne traiterons pas cette couche dans nos travaux.

Nous nous intéressons plus particulièrement au niveau modèle M1.

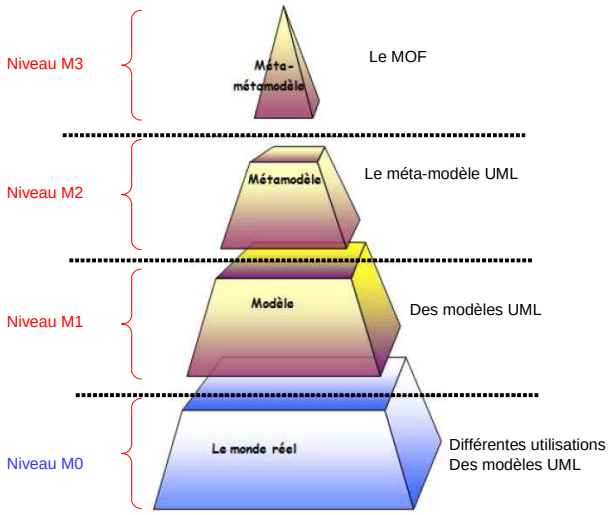


Figure 4-2 : Représentation des couches de méta-modélisation dans le cas d'UML [BEZ-03]

A.1.2 Niveau « modèles »

La MDA vise à séparer les modèles du domaine métier, des modèles liés à l'implémentation logicielle. Ceci permet de garder la pérennité de la logique de métier malgré le changement d'environnement et d'assurer l'indépendance du modèle applicatif du système des contraintes de la plateforme matérielle.

Pour cela, la MDA définit trois principaux modèles au niveau M1 (voir Figure 4-3) :

- un modèle indépendant de la plateforme (PIM « *Platform Independent Model* ») , spécifiant un savoir-faire métier indépendamment de la technologie de mise en œuvre ; par exemple une architecture de système décrite en SysML.
- un modèle dépendant de la plateforme (PSM « *Plateform Specific Model* »), dans lequel les spécifications d'implémentation de PIM dans une plateforme précise sont définies. À partir d'un même modèle PIM, nous pouvons générer différents PSM, correspondant chacun à une plateforme cible (Figure 4-4). Par exemple, un modèle VHDL-AMS décrivant la composition de composants VHDL-AMS disponibles dans une bibliothèque à partir d'une transformation du PIM SysML précédent.
- le modèle exécutable ou code exécutable, qui sera exploité pour instancier le modèle en niveau M0. Ce code est généré à partir du PSM et il est adapté à la plateforme cible ; par exemple, un code C ou sa DLL associée permettant de simuler concrètement le modèle.

Ces trois modèles disposent chacun de leur méta-modèle (ou langage, niveau M2) dont il faut tenir compte pour développer des outils de transformation.

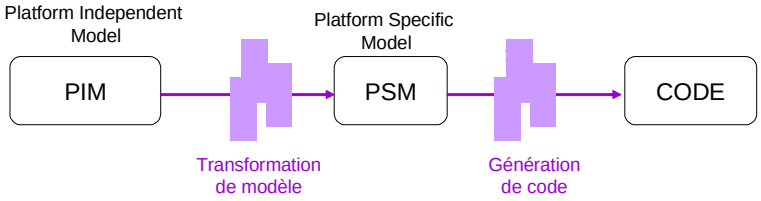


Figure 4-3 : Le processus de la MDA et ses modèles

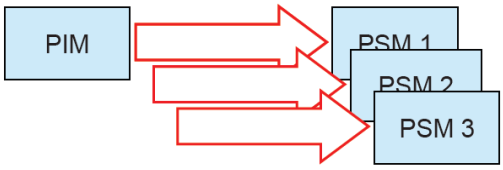


Figure 4-4 : Différents modèles PSM générés à partir d'un modèle PIM

A.2. Transformation des modèles

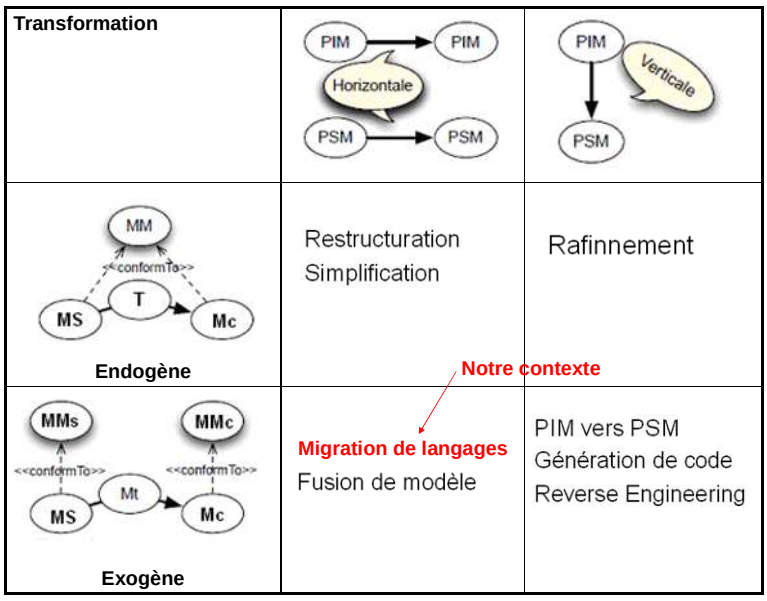
Le concept de transformation de modèle permet de passer d'une représentation d'un modèle à une autre représentation de niveau d'abstraction équivalent ou non (PIM-PSM, PSM-PSM, PSM-code, etc.). Les modèles source et cible doivent être conformes à une grammaire ou à un méta-modèle (différents ou non). Un tel processus n'est possible que s'il existe une ou plusieurs correspondances entre le modèle source et cible ; c'est-à-dire s'il existe des correspondances sémantiques entre leurs méta-modèles respectifs. Ces correspondances peuvent être concrétisées par des règles de transformation, écrites dans un

langage (XSLT[67], ATL[68], etc.). Ces règles précisent notamment en comment un élément du méta-modèle source sera traduit en un élément du méta-modèle cible.

Dans ses travaux, D. GUIHAL a présenté un état de l'art très riche sur les transformations de modèles, en abordant les différents types et techniques disponibles [GUI-07]. Ainsi, il en ressort quatre transformations :

- **La transformation endogène**. C'est une transformation dont les modèles sources et cibles sont conformes au même méta-modèle. Une telle transformation est utile dans le cadre d'une optimisation, d'une restructuration ou d'une simplification de modèle.
- **La transformation exogène**. C'est une transformation dont les modèles sources et cibles sont dans des langages différents. Il peut s'agir d'une migration (d'un langage à un autre), d'une synthèse (génération de code) ou de "reverse engineering".
- **La transformation horizontale**. Elle représente des modèles source et cible de même niveau d'abstraction ; tels que la fusion de deux modèles de même niveau afin d'obtenir un troisième modèle, la transformation PIM en PIM ou PSM en PSM, etc.
- **la transformation verticale**. Elle représente des modèles sources et cibles de niveaux différents d'abstraction. Elle peut illustrer un raffinement de modèle, le passage de PIM à PSM et inversement, etc.

La Figure 4-5 synthétise les différentes dimensions de transformation des modèles en indiquant leurs cas d'utilisation [GUI-07].



Légende:

Ms: modèle source
Mc : modèle cible ;
MM : méta-modèle
MMs: méta-modèle source
MMc: méta-modèle cible

Figure 4-5 : Double dimensionnalité des transformations de modèles [GUI-07]

[67] XSLT (Extensible Stylesheet Language Transformations) est un langage de feuilles de style permettant la mise en forme d'un fichier XML, et la transformaation un document XML en un autre format.
[68] ATL (ATLAS Transformation Language) est un langage de transformation de modèle au niveau des meta-modèles. Il a pour vocation de permettre l'expression de règles de transformation et de pouvoir les exécuter.

A.3. Apport de cette approche à notre problématique

A.3.1 Technique de transformation de modèle basée sur les méta-modèles

Dans notre problématique, nous souhaitons projeter nos modèles métiers vers le langage VHDL-AMS. Les transformations à appliquer vont dépendre essentiellement des modèles sources dont nous disposons. Ainsi, dans les phases amont du cycle de conception, où l'on s'intéresse essentiellement aux spécifications du système à réaliser, on se place dans le cas d'une transformation PIM vers PSM, appelée approche « top-down ». Le cas que nous allons traiter ici s'apparente au contraire à une approche « bottom-up » ; c'est-à-dire qu'on souhaite réutiliser des modèles de composants existants dans une composition système non directement compatible (problème d'interopérabilité). Dans ce cas, on se place plutôt dans une transformation PSM vers PSM ; l'idéal étant de disposer d'un PIM intermédiaire basé sur un méta-modèle rassemblant la sémantique commune aux deux PSM.

La Figure 4-6 présente le cas de figure (transformation PSM vers PSM) dans lequel nous nous plaçons. Le langage VHDL-AMS est la cible des transformations avec des modèles sources mettant en jeu des langages de modélisation généralement différents. Cette approche est une manière efficace et pérenne d'assurer l'interopérabilité entre deux modèles, même si les sémantiques entre les modèles ne pourront pas être totalement similaires en raison des contraintes de langage. Pour y parvenir, un travail important est nécessaire ; notamment définir les méta-modèles communs pour les différentes modélisations sources et définir une description de méta-modèle associé au modèle cible VHDL-AMS.

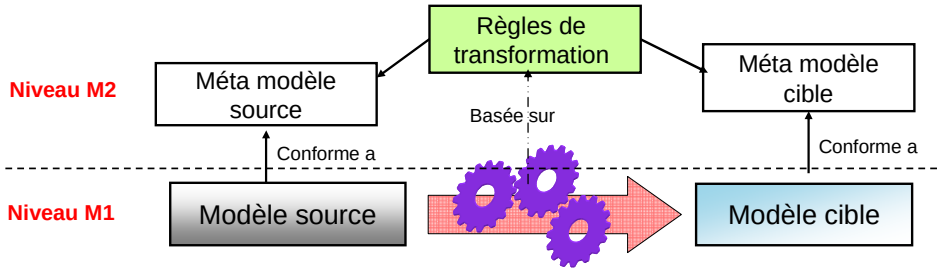


Figure 4-6 : Technique de transformation de modèle basée sur les méta-modèles

A.3.2 Méta-modèle VHDL-AMS

Afin de fournir un méta-modèle du langage VHDL-AMS, on peut citer les travaux de :

- D. Guihal [GUI-07] : qui a proposé un méta-modèle VHDL-AMS non exhaustif utilisé dans le cadre de transformation du langage MAST vers le langage VHDL-AMS,
- J. Verries [VER-10] : qui a réalisé un méta-modèle VHDL-AMS dans le cadre de la conception aéronautique avec de transformation SysML vers VHDL-AMS.

Les deux méta-modèles, pourtant du même langage, ne sont pas rigoureusement identiques. En effet, celui de D. Guihal est beaucoup plus proche de la grammaire concrète VHDL-AMS sous forme BNF[69]. Ils utilisent le même méta-langage mais celui-ci ne donne pas les méthodes permettant de les construire. Ces travaux montrent qu'il n'existe pas

[69] BNF : Acronyme de Backus Naur Form (ou Backus Normal Form), une notation méta-syntaxique utilisée pour spécifier la syntaxe des langages de programmation, des langages de commande et tout autre langage apparenté.

d'unicité dans l'écriture des méta-modèles. Il s'avère que la création d'un méta-modèle dépend de l'utilisation que l'on souhaite en faire [GAU-12].

A.3.3 Notre problématique de transformation de modèle

La Figure 4-7 illustre un schéma résumant la problématique abordée dans ce chapitre. Les différentes difficultés, auxquelles nous devons répondre, sont numérotées de 1 à 4:

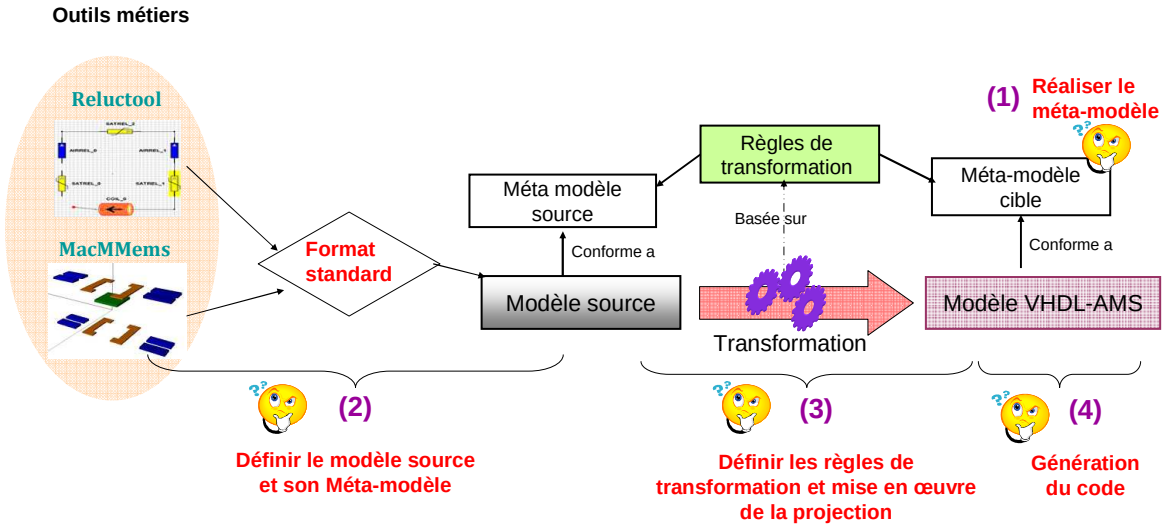


Figure 4-7 : Schéma de principe résumant la problématique de ce chapitre

(1) **Disposer d'un méta-modèle VHDL-AMS :** Le langage VHDL-AMS est la cible de toutes nos transformations. Un méta-modèle de ce langage est indispensable pour vérifier la conformité syntaxique d'un modèle généré. Dans nos développements, nous nous sommes référé au méta-modèle réalisé par D. Guihal [GUI-07] qui se veut le plus générique possible.

(2) **Définir les modèles sources :** Les modèles sources issus des outils métiers mettant en jeu des langages de modélisation différents (SML pour les microsystèmes et Java pour les réseaux de réluctances). Nous souhaitons proposer un format standard du modèle source qui fera la passerelle intermédiaire pour les transformations appliquées aux différents outils métiers. Notre choix s'est naturellement porté sur le langage XML[70] ; grâce à ses propriétés structurantes (simplicité, extensibilité, interopérabilité et ouverture) des données.

(3) **Définir les règles de transformation :** Il s'agit d'établir une correspondance entre la syntaxe du méta-modèle XML source et le méta-modèle VHDL-AMS.

(4) **Mettre en œuvre la génération du code VHDL-AMS.**

Afin d'illustrer cette approche, nous allons étudier la modélisation VHDL-AMS des systèmes magnétiques par réseau de réluctances, puis la projection vers le langage VHDL-AMS de leurs modèles issus de l'outil RelucTool. Ensuite, nous étudierons la projection des modèles de microsystèmes magnétiques issus de l'outil MacMMems.

[70] XML : Extensible Markup Language

B. Modélisation VHDL-AMS des réseaux de réluctances

Nous allons présenter le principe de l'approche « réseau de réluctances ». Ensuite, nous étudierons la modélisation VHDL-AMS d'un tel circuit, en mettant l'accent sur ses limites.

B.1. Description des systèmes magnétiques par réseaux de réluctances

B.1.1 Principe de l'approche [ROT-41]

La modélisation par réseaux de réluctances permet de représenter, en régime statique, un système par un schéma descriptif de son circuit magnétique. Cette méthode permet une description simplifiée du comportement magnétique d'un dispositif ; la distribution du champ peut être décrite avec un ensemble d'équations algébriques. La précision est raisonnable pour la conception grossière et la simulation système des actionneurs.

Le principe de cette approche repose sur la définition des tubes de flux magnétiques (Figure 4-8). Un tube de flux est un volume défini avec une répartition homogène du champ magnétique *H* et l'induction magnétique *B* dans cette région. Chaque tube de flux est défini par un élément magnétique caractérisé par sa géométrie et ses propriétés magnétiques.

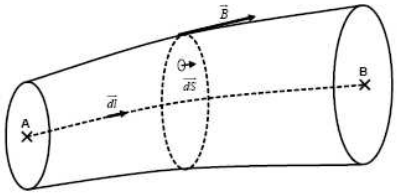


Figure 4-8 : Tube de flux

Sous les hypothèses du régime magnétostatique où les effets induits des conducteurs sont négligés, le flux φ_R et la différence du potentiel magnétique θ_A^B entre deux points extrêmes (A et B) d'un tube de flux sont calculés à l'aide des expressions intégrales des équations de Maxwell [ROT-41] [MEU-08] :

$$\varphi_R = \iint_{(S)} \vec{B}.\vec{dS} \qquad \text{Eq 4-1}$$

$$\theta_A^B = \int_A^B \vec{H}.\vec{dl} \qquad \text{Eq 4-2}$$

avec - *S* : Section associée au tube de flux

- *l* : Longueur moyenne du tube de flux

La réluctance magnétique du tube de flux entre A et B est définie par le rapport du potentiel magnétique et du flux magnétique :

$$R = \frac{\theta_B^A}{\varphi_R} = \int_A^B \frac{dl}{\mu \cdot S} \qquad \text{Eq 4-3}$$

où μ est la perméabilité magnétique du matériau.

En considérant que l'induction *B* et le champ de magnétique *H* sont homogènes et que la section *S* est constante le long du tube de flux, la formule de la réluctance devient :

$$R = \frac{H.l}{\varphi.S} \qquad \text{Eq 4-4}$$

Cette formulation est valable pour le calcul de réluctances dans l'air ou dans les matériaux magnétiques. Selon la nature de ce dernier, la relation entre $H(\varphi)$ ou $H(B)$ prend une expression spécifique. Cette réluctance magnétique représente donc une caractéristique de la géométrie et des matériaux du tube de flux.

Par analogie entre les circuits électriques résistifs et les circuits magnétiques, il découle la base de l'analyse par réseaux de réluctances qui consiste à décomposer un circuit magnétique en sous-éléments (réluctances ou sources de potentiel). Cette approche permet une résolution du circuit réluctant, comme un circuit électrique, en utilisant les lois de Kirchhoff généralisées pour trouver les flux dans les différentes branches du circuit. Ainsi, tout élément constituant un composant magnétique peut être modélisé par un élément d'un réseau de réluctances. La méthode des tubes de flux, son utilisation pour la conception de dispositifs et les nombreuses formes de tubes de flux sont expliquées en détail dans [ROT-41].

B.1.2 Éléments constitutifs d'un réseau réluctant

On peut distinguer deux types principaux d'éléments constitutifs d'un schéma réluctant et qui sont reliés par le chemin de circulation des flux :

- les réluctances : éléments passifs dans le circuit magnétique.
- les sources de potentiels : des bobines électroaimants ou des aimants permanents.

(1) Les réluctances de fuites et d'entrefers

La perméabilité de l'air étant constante, l'expression d'une réluctance correspondant à un tube de flux passant dans l'air est une constante ne dépendant que de la géométrie du tube de flux considéré. La formulation de l'équation 4-4 est valable pour le calcul de réluctances dans l'air. La relation entre $H(\varphi)$ ou $H(B)$ s'écrit :

$$H = \frac{B}{\mu_0} \qquad \text{Eq 4-5}$$

Pour un tube de flux rectiligne de section constante *S* et de longueur *l*, dans l'air, on aura :

$$\mathrm{R} = \frac{\mathrm{l}}{\mu_0 \mathrm{S}} \qquad \text{Eq 4-6}$$

où μ_0 est la perméabilité magnétique du vide.

Pour des éléments de géométrie plus complexe, le calcul se déroule en intégrant l'expression élémentaire de la réluctance sur le volume du tube de flux.

(2) Les réluctances linéaires et non linéaires

Les matériaux magnétiques sont modélisés par des modèles analytiques sans hystérésis. Nous pouvons en retenir deux parmi les nombreux modèles de matériaux existants :

- un modèle linéaire :

$$H = \frac{B}{\mu_0 . \mu_r} \qquad \text{Eq 4-7}$$

- un modèle non linéaire saturable fourni par Flux2D/3D :

$$H = \frac{(\mu_r - 2a + 1)B - \mu_r . J_s - J_s(2a - \mu_r)\sqrt{\left(\frac{(\mu_r - 1)B}{J_S(2a - \mu_r)}\right)^2 - \frac{4a(a - \mu_r)}{(2a - \mu_r)}}}{2\mu_0 . (\mu_r - a)} \qquad \text{Eq 4-8}$$

où : - J_s est la polarisation magnétique à saturation

- *a* est un paramètre relatif au coude de la courbe *B(H)*.

Pour différentes géométries de réluctance, des modèles spécifiques peuvent être définis : réluctances à section variable, réluctance de fuite entre deux surfaces circulaires, etc.

(3) Les sources

Nous pouvons aussi définir des éléments sources idéaux :

- une source d'une différence de potentiel magnétique *MMF*,
- une source d'un flux magnétique φ.

Ils sont destinés à être utilisés dans les calculs de flux stationnaires où aucun couplage n'est nécessaire, entre le sous-système magnétique et le circuit électrique l'alimentant, ou bien quand les effets dynamiques de ce couplage ne doivent pas être pris en considération.

Les autres éléments sources les plus utilisés sont les bobines et les aimants permanents.

o Bobine

Une bobine est une source d'ampères-tours. Elle fournit une force magnétomotrice F_m au circuit magnétique. Dans un contexte dynamique, elle représente un convertisseur électromagnétique de couplage au sous-système électrique. C'est un composant à quatre ports ; deux ports magnétiques connectés au réseau magnétique et deux ports électriques connectés au circuit électrique.

Les équations électriques sont liées aux équations électromagnétiques comme suit :

$$\begin{aligned} F_m &= N.I \\ V &= R.I - N.\frac{d\varphi}{dt} \end{aligned} \qquad \text{Eq 4-9}$$

où - *N* est le nombre de spires de la bobine,

- *I* le courant qui la traverse,
- *R* est la résistance interne de la bobine
- φ est le flux traversant la bobine.

Dans la plupart des dispositifs électromagnétiques, le flux φ est une fonction non linéaire qui dépend du courant *I* (en raison d'effets de saturation des composants ferromagnétiques) et de la position *x* de la pièce en mouvement dans le cas d'un actionneur linéaire.

o Aimant permanent

Les matériaux ferromagnétiques durs (champ coercitif élevé) sont utilisés dans les actionneurs pour la réalisation des aimants. Ces matériaux conservent mieux leurs aimantations qu'un matériau doux une fois aimanté. Ces aimants peuvent être modélisés par une caractéristique (courbe de désaimantation *B (H)*) linéaire sur la zone de fonctionnement (Figure 4-9), ce qui simplifie les calculs. Les valeurs caractéristiques de l'aimant sont

l'induction rémanente B_r et le champ coercitif H_c. La perméabilité relative du matériau μ_a est décrite par l'équation suivante :

$$\mu_a = \frac{B_r}{\mu_0 . H_c} \qquad \text{Eq 4-10}$$

On peut représenter ce type d'aimant par une source d'ampères-tours F_m en série avec une réluctance interne R_a, équivalent à une modélisation «Thévenin». La Figure 4-10 présente le modèle équivalent d'un aimant de section *Sa* et de longueur *La*.

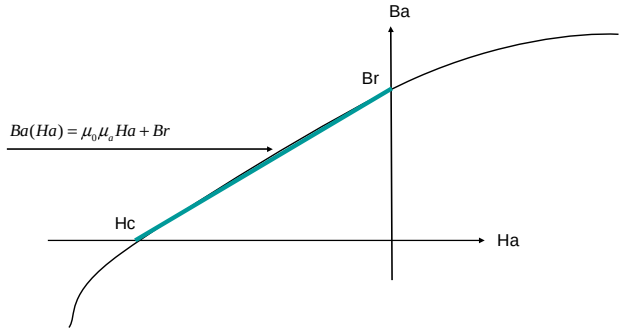


Figure 4-9 : Modèle linéaire d'un aimant

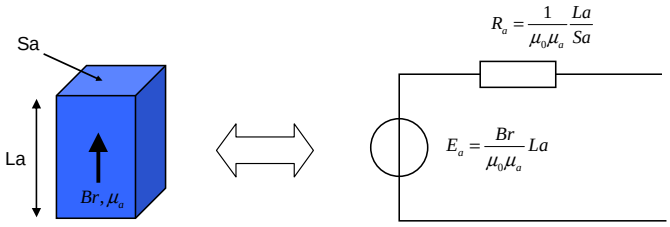


Figure 4-10 : Modélisation équivalente de type «Thévenin» d'un aimant

B.2. Description structurelle d'un réseau de réluctance en VHDL-AMS

Afin de modéliser un actionneur magnétique, nous devons décrire le circuit magnétique étudié à partir de ses réluctances. La principale difficulté de modélisation est de définir sa structure de manière correcte. Cette modélisation se fait en deux temps :

- l'étude des réluctances de l'actionneur lui-même, puis celle des réluctances de fuites.
- la résolution du système d'équations non linéaires qui en découle. Celle-ci permet d'obtenir la répartition des flux à l'intérieur du circuit magnétique.

Il faut noter qu'une autre difficulté principale réside dans la définition d'un modèle paramétré restant valable dans un large domaine de variation de ses paramètres.

B.2.1 Bibliothèque de composants réluctants

Basé sur le concept de tubes de flux magnétiques, nous avons réalisé une bibliothèque d'éléments réluctants de base en VHDL-AMS. Chaque élément (réluctance ou source de flux) est complètement décrit dans un modèle VHDL-AMS spécifiant :

- son interface externe *(Entity)* : le nom du modèle, ses paramètres de configuration (qui sont accessibles à l'utilisateur via la boîte de dialogue), la déclaration de ses ports de connexion (électriques ou magnétiques) et de ses paramètres de sorties ;

- son comportement interne *(Architecture)* : des expressions mathématiques impliquant les paramètres géométriques et les variables principales (réluctances, sources, efforts, flux, et les sorties souhaitées telles que la force magnétomotrice *MMF*, l'énergie et la co-énergie propres au composant, etc.).

Prenons l'exemple de la bobine électromagnétique qui représente un convertisseur d'énergie électromagnétique. Son modèle est développé en VHDL-AMS à partir de la loi d'Ampère et de la loi de Faraday (Eq 4-9).

Le modèle VHDL-AMS de l'interface de la bobine (*Entity*) et le comportement *(Architecture*) sont présentés ci-dessous (Figure 4-11). Il a quatre ports conservatifs :

- deux électriques (les variables effort et flux sont la tension et le courant)
- deux magnétiques (les variables effort et flux sont la flux magnétique φ et la force magnétomotrice F_m).

```
entity Coil is
   generic ( R : real :=1.0 ; --- résistance interne
             N : real :=1000.0 ; --- nombre de spires
             phi0 : real :=0.001); --- flux initial
   port (   terminal pm1 , pm2 : MAGNETIC;
            terminal pe1 , pe2 : ELECTRICAL);
end entity;

architecture behav of Coil is
    quantity Fm across phi through pm1 to pm2;
   quantity v   across i   through pe1 to pe2;
Begin
    break phi => phi0
    v  == R * i - N * (phi'dot); --Faraday's law
    Fm == N * i; -- Ampere's law
end architecture;
```

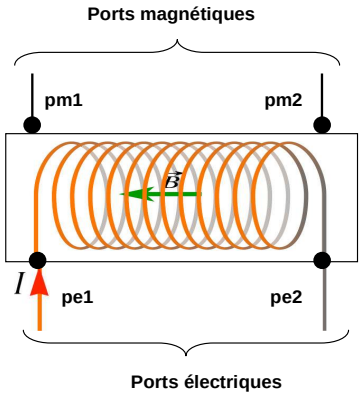


Figure 4-11 : Description VHDL-AMS d'un convertisseur électromagnétique

Cette bibliothèque VHDL-AMS est destinée au pré-dimensionnement et à la simulation système du circuit magnétique. On note que cette bibliothèque se limite actuellement aux composants définis dans la version 1.3 de RelucTool [DUP-06].

B.2.2 Modèle réluctant structurel en VHDL-AMS

Cette bibliothèque permet aux concepteurs de décrire rapidement et intuitivement un réseau réluctant dans n'importe quel simulateur implémentant la norme VHDL-AMS :

- soit en plaçant ses différents éléments (réluctances et sources de flux) dans l'interface du simulateur et en les connectant graphiquement.
- soit en décrivant la topologie du circuit totalement en VHDL-AMS, ce qu'on appelle un « modèle structurel », grâce à l'instanciation des composants.

L'avantage de la deuxième approche est qu'on peut facilement ajouter des équations complémentaires dans le modèle afin de lier certains paramètres du modèle, factoriser certaines expressions ou enrichir le modèle. Ce mécanisme, représenté sur la Figure 4-12, est appliqué à un contacteur électromagnétique de structure classique avec une culasse en forme de « E » et avec une bobine centrée sur la colonne du milieu.

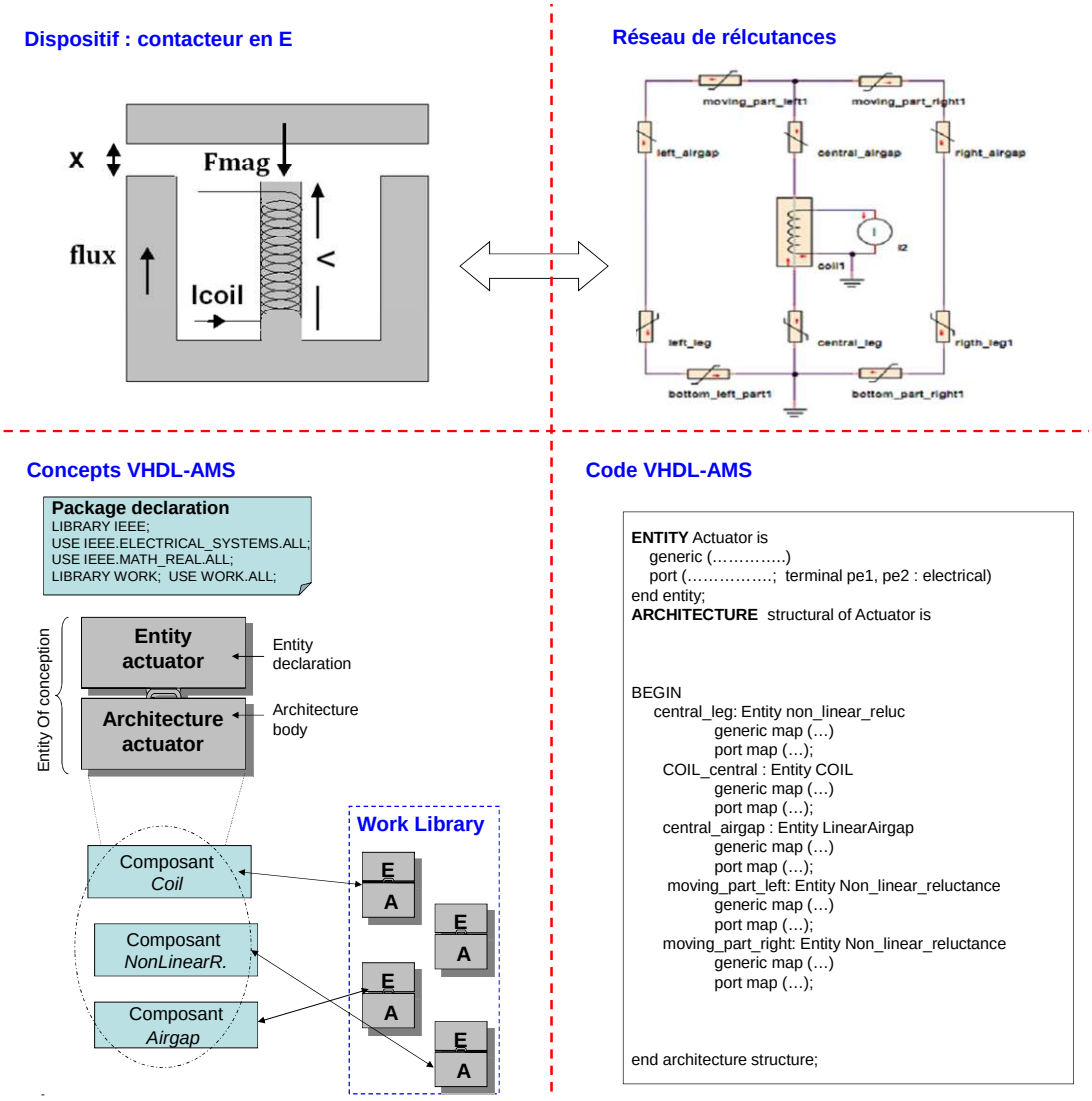


Figure 4-12 : Description structurelle en VHDL-AMS d'un contacteur en E

Concevoir des réseaux réluctants plus complexes met en évidence la structure hiérarchique du réseau et la répétitivité d'instanciation des composants réluctants dans les systèmes. Si les logiciels implémentent complètement la norme VHDL-AMS, nous devrions être en mesure d'offrir au concepteur :

- plus de flexibilité et moins d'efforts dans la gestion et la mise en œuvre de sa structure hiérarchique ;
- la possibilité de décrire le composant une seule fois, puis décrire la façon dont il doit être instancié plusieurs fois plutôt que de décrire chaque instanciation individuellement.

Le langage VHDL-AMS peut répondre à ces deux exigences, à l'aide des mots clés « *configuration* » et « *component* » et de l'instruction « *generate* » [ASH-02] [HER-02].

B.2.3 Calcul de la force magnétique

(1) Formulation

Pour déterminer le comportement dynamique d'un actionneur tel que le contacteur en « E », le modèle du circuit magnétique est couplé au circuit électrique alimentant la bobine et au système mécanique (ressort) relié au noyau mobile du contacteur (Figure 4-13). Lorsque la bobine est excitée en courant, le noyau mobile se déplace jusqu'à la fermeture de l'actionneur. Seul le mouvement linéaire vertical du noyau est possible. Le noyau est soumis à différentes forces mécaniques (amortissement visqueux et sec). La force magnétique générée *Fmag* doit surcompenser ces forces pour attirer le noyau mobile.

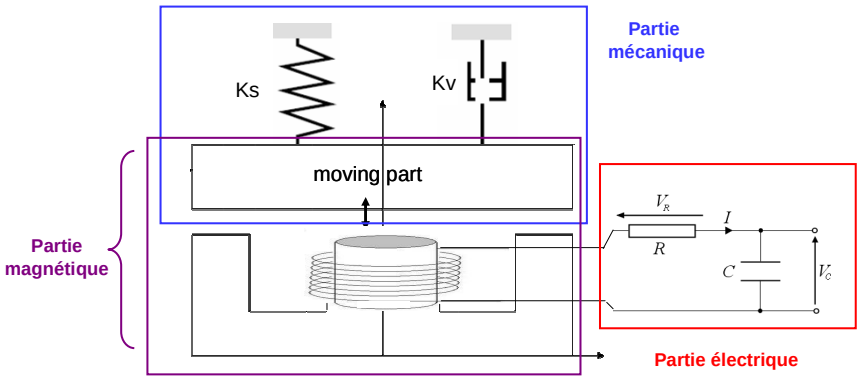


Figure 4-13 : Description simplifiée du contacteur magnétique

Dans le cas d'un mouvement rectiligne (à un degré de liberté), la force magnétique peut être calculée d'une manière générale en dérivant la coénergie E_{co} dans les réluctances en fonction de la variable de position *x* de la partie mobile [MAZ_04] :

$$F_{mag} = \frac{\partial E_{co}}{\partial x} \qquad \text{Eq 4-11}$$

Pour chaque élément du schéma réluctant, la densité d'énergie et de coénergie sont données par leur courbe *B(H)*. Ce sont respectivement les aires au-dessus et au-dessous de cette courbe (Figure 4-14). Lorsque les relations H(B) ou également B(H) sont connues analytiquement, il est facile d'en déduire les expressions analytiques de la densité d'énergie *W* et de coénergie W_{co} d'une réluctance.

$$W = \int_0^B H.dB \qquad \text{Eq 4-12}$$

$$W_{co} = \int_0^H B.dH \qquad \text{Eq 4-13}$$

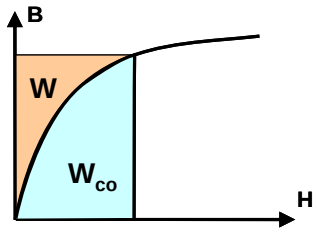


Figure 4-14 : Densités d'énergie W et de coénergie Wco magnétique

L'énergie et la coénergie d'un élément sont le résultat de la multiplication de ces densités par le volume de l'élément. En sommant ces grandeurs sur tous les éléments constitutifs du réseau, on obtient alors celles du dispositif complet. Par exemple, pour une réluctance définie par un tube de flux rectiligne de section constante *S*, de longueur *L* et ayant un matériau linéaire et un champ homogène, la force magnétique est calculée comme suit :

- densité de l'énergie et de la coénergie

$$W = W_{co} = \frac{1}{2} \cdot B \cdot H = \frac{1}{2} \cdot \frac{\varphi^2}{\mu_0 \cdot \mu \cdot S^2} \qquad \text{Eq 4-14}$$

- énergie *E* et de la co-énergie E_{co} totale

$$E = E_{co} = W \cdot Volume = \frac{1}{2} \cdot \frac{\varphi^2}{\mu_0 \cdot \mu \cdot S^2} \cdot (L.S) \qquad \text{Eq 4-15}$$

- force magnétique appliquée (la variable de déplacement étant x)

$$F_{mag} = \frac{\partial E_{co}}{\partial x} = \frac{1}{2} \cdot \frac{\varphi^2}{\mu_0 \cdot \mu \cdot S} \qquad \text{Eq 4-16}$$

(2) Traitement du calcul de la force à partir de la co-énergie

Afin de pouvoir obtenir la force magnétique créée par un circuit magnétique, nous avons introduit les équations algébriques analytiques issues de ces formulations intégrales du calcul de l'énergie et de la coénergie pour chaque élément réluctant [DUP-06].

• Calcul de la force

Les calculs de l'énergie, de la coénergie et de la force magnétique sont ensuite décrits d'une manière globale à l'issue de la description structurelle du circuit magnétique. En effet, à la mise en équations génériques de chaque réluctance, la variable de déplacement x n'est pas définie localement pour chaque réluctance. Ceci s'effectue en ajoutant :

- soit un composant spécifique de calcul de force, si le circuit est décrit graphiquement dans le simulateur ;
- soit des équations complémentaires de calcul de la force dans le modèle structurel décrit en VHDL-AMS (Figure 4-12).

• Problème de formulation rencontré en VHDL-AMS

Cette méthode de calcul de la force magnétique implique le calcul d'une l'équation aux dérivées partielles (généralisée aux dimensions d'espace, Eq 4-16). Cependant, nous avons expliqué dans le chapitre 2 et 3, que le langage VHDL-AMS et les simulateurs système ne supportent pas une telle description.

En pratique, à l'aide de l'analyse des mailles indépendantes, la mise en équations d'un circuit réluctant génère un système d'équations implicites (en raison des expressions des réluctances qui dépendent des flux). Les valeurs de ces flux sont obtenues grâce à la résolution numérique de ce système. Ainsi, le calcul des différentes densités de la co-énergie, nécessitant les expressions de flux traversant chaque élément, ne peut être effectué qu'après la résolution du système d'équations implicites.

Les solutions de couplage faible proposées dans le chapitre 3, par l'intermédiaire d'une fonction externe en VHDL-AMS, ne peuvent pas être appliquées ici pour le calcul de la force magnétique. En effet, le calcul devient délicat, car il dépend des dérivées de flux par rapport aux variables d'espace, et donc nécessite l'accès aux équations de flux qui ne sont pas accessibles avant la résolution du modèle du circuit.

- **Calcul de la force par approximation**

Dans le cas d'une structure simple ne comportant qu'un simple entrefer, la force magnétique *Fmag* peut être calculée avec une formule simplifiée (Eq 4-19). Celle-ci utilise l'expression de pression magnétique W_d (Eq 4-17),

- calcul de la pression magnétique W_d

$$W_d = \frac{B^2}{2.\mu_0} \qquad \text{Eq 4-17}$$

- calcul de la co-énergie totale E_{co}

$$E_{co} = x.S.W_d \qquad \text{Eq 4-18}$$

- calcul de la force F_{mag}

$$F_{mag} = \frac{\partial E_{co}}{\partial x} = \frac{\partial (x.S.W_d)}{\partial x} = S.\frac{B^2}{2.\mu_0} = \frac{\varphi^2}{2 \cdot \mu_0 \cdot S} \qquad \text{Eq 4-19}$$

où

- φ est le flux magnétique,
- S la surface du noyau mobile (section transversale de l'entrefer),
- x la variable de déplacement.

Cette simplification de l'expression s'appuie sur les hypothèses suivantes :

- le courant d'excitation dans la bobine est constant,
- seules les réluctances selon l'axe de déplacement sont prises en compte (on ne prend pas en compte les réluctances de fuites par exemple),
- on se limite à de l'air ou des matériaux linéaires (remplacer par $\mu_0 \cdot \mu$),
- on néglige l'énergie stockée dans les matériaux ferromagnétiques (qui est beaucoup plus faible que celle cachée dans l'entrefer).

Dans des systèmes complexes, cette formulation 4-19 n'est pas applicable, car elle dépend fortement de la topologie du circuit magnétique.

- **Vers le calcul exact de la force**

Nous proposons de nous appuyer sur l'approche plus générale de calcul des forces magnétiques, c'est-à-dire celle de la dérivation de la coénergie du système. Notre objectif est de générer automatiquement des modèles VHDL-AMS des circuits magnétiques à partir de l'outil dédié à la modélisation des réseaux de réluctances RelucTool ; qui un modèle analytique du circuit en offrant le calcul automatique de l'effort magnétique.

C. Projection de modèles de RelucTool vers VHDL-AMS

Notre objectif est de faire évoluer l'outil RelucTool pour faire face à la portabilité des modèles, notamment sous une forme explicite en VHDL-AMS.

Nous présenterons brièvement l'outil RelucTool. Ensuite, nous détaillerons les différentes étapes de la projection de modèles de RelucTool vers le langage VHDL-AMS. Enfin, nous présenterons les générateurs VHDL-AMS automatiques que nous avons réalisés.

C.1. RelucTool : une interface métier dédiée aux schémas réluctants

RelucTool a été développé dans le but de rendre aux concepteurs la description de réseaux réluctants facile et intuitive. Basé sur le formalisme analytique de modélisation par réseaux de réluctances, l'outil RelucTool permet de simuler très rapidement des dispositifs entièrement paramétrables de machines électriques en régime statique et de capteurs et actionneurs en régimes statique et dynamique [DUP-06] [DO-10]. Son objectif principal est le dimensionnement par optimisation de dispositifs électromagnétiques via des algorithmes s'appuyant sur l'exploitation des gradients du modèle statique. L'originalité de l'outil se trouve donc dans la mise en équation automatique du modèle de dimensionnement (incluant les calculs de gradients) à partir du réseau réluctant dessiné graphiquement par l'utilisateur.

Il propose différents formalismes pour la description des systèmes électromagnétiques :

- **pour la partie magnétique** : saisie des schémas réluctants ;
- **pour la partie électrique :** circuits électriques venant alimenter les bobines ;
- **pour la partie mécanique** pour les actionneurs : caractéristiques des efforts résistants de la partie mobile (ressort, etc.).

Nous ne nous intéressons qu'à la modélisation des circuits magnétiques d'un dispositif électromagnétique. RelucTool se compose (Figure 4-15) [DUP-06] :

- d'une interface graphique permettant la saisie et la configuration de schémas réluctants ;
- d'un module de génération qui transforme les informations métiers entrées dans l'interface en un modèle analytique (format XML non accessible à l'utilisateur) ;
- d'un module de projection de ce modèle vers un composant ICAr exploitable ;
- d'outils de calcul, de simulation et de dimensionnement de CADES [DEL-12] permettant d'exploiter le composant ICAr généré.

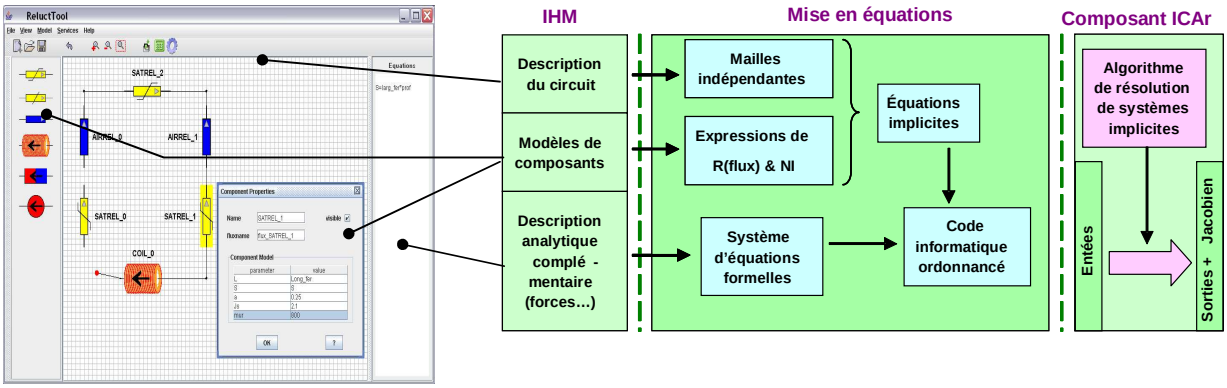


Figure 4-15 : Principe de création d'un modèle réluctant

La fonction principale de RelucTool est de générer automatiquement un modèle analytique (sous forme d'un composant logiciel ICAr) d'un circuit réluctant en calculant un ensemble de grandeurs physiques nécessaires pour la simulation. A partir d'une bibliothèque évolutive contenant les différentes réluctances et sources d'ampères-tours, le concepteur décrit la topologie de son réseau. RelucTool se charge de mettre le modèle en équations par extraction d'un jeu de mailles indépendantes qui permettront de calculer de manière formelle les expressions des flux élémentaires les parcourant. En parallèle, le calcul formel du jacobien du système est effectué en vue de l'optimisation. Même si la présence de réluctances saturables mène à des systèmes d'équations implicites qui doivent être résolus par des méthodes numériques, le théorème des fonctions implicites permet d'en obtenir les dérivées partielles de manière formelle [DUP-06]. Tous ces calculs sont ensuite regroupés dans un composant ICAr autonome (chapitre 2). Les différentes techniques utilisées pour la transformation du schéma réluctant en un composant ICAr et que nous avons rexploité sont présentées dans l'annexe 4-1.

C.2. Mise en œuvre de la projection de RelucTool en VHDL-AMS

Dans l'optique de projection des modèles réluctants issus de l'outil RelucTool vers le langage VHDL-AMS, nous allons nous appuyer sur l'approche de transformation de modèles présentée dans le paragraphe A.

Nous détaillerons les différentes étapes de la projection, comme présenté sur la Figure 4-7. Nous nous intéressons d'abord à l'étape de la définition du modèle source et de son méta-modèle associé. Ensuite, nous définirons un méta-modèle VHDL-AMS. Puis, nous mettrons en place les règles de transformation nécessaires et nous mettons en œuvre la génération de code en VHDL-AMS.

Nous n'allons pas détailler l'étape concernant la génération du méta-modèle VHDL-AMS. Nous notons que chaque modèle VHDL-AMS généré doit respecter la description du méta-modèle proposée par D.Guihal [GUI-07].

C.2.1 Représentation du modèle source

Afin de définir une représentation standard du modèle source issu de RelucTool, il est nécessaire de veiller à la réutilisation, notamment au niveau des modèles créés présentant un caractère générique. Nous proposons de nous appuyer sur une représentation intermédiaire du modèle issu de RelucTool ; et donc un modèle indépendant des outils (PIM[71] au sens MDA) qui agira comme une passerelle pour la transformation entre les représentations des modèles réluctants spécifiques à chaque outil (PSM[72]).

Notre choix s'est naturellement porté vers le langage XML comme modèle source grâce à ses propriétés structurantes (simplicité, extensibilité, interopérabilité et ouverture). Cette description générique d'un modèle réluctant doit permettre de générer des projections dans divers langages (VHDL-AMS, Modelica, Simscape etc.) plus facilement que s'il fallait repartir du format propriétaire de RelucTool.

[71] PIM : plateforme independant model
[72] PSM : plateforme specific model

L'enjeu est de trouver une représentation structurée et évolutive du modèle réluctant. Deux aspects sont à prendre en compte :

- les informations liées à la topologie proprement dite du réseau réluctant pour générer un modèle structurel en VHDL-AMS.
- la mise en équation du modèle global pour générer le modèle comportemental y compris le calcul de la force magnétique.

(1) Modèle initial généré par RelucTool

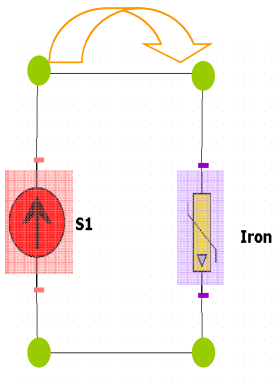
Dans RelucTool, lorsque le concepteur décrit la topologie de son réseau, une représentation objet du graphe correspondant au circuit associé est construite. Cette représentation a été proposée par B. Du Peloux, pour sauvegarder les informations graphiques sous forme d'un fichier XML [DUP-06]. La Figure 4-16 présente la représentation graphique d'une source d'ampère-tour dans un circuit magnétique saturable.

En effet, un réseau de réluctances est décrit par :

- ses différents composants (réluctances et sources de flux) ;
- leurs connexions (à travers connecteurs graphiques) ;
- les équations complémentaires pour enrichir le modèle.

Chaque réseau est décrit entre les balises <*MagneticComponent*> et possède un nom spécifié grâce à l'attribut *title* et ses différents éléments entre les balises <*container*>.

D'une manière générale, chaque composant est défini par son modèle (son type et ses équations) et ses ports auxquels on pourra connecter les fils. Ensuite, les connecteurs graphiques <*connector*> et les liaisons <*link*> entre les ports et connecteurs sont identifiés.




```
<magneticmodel title="Magnetic">
Modèle IRON
<container hashcode="5555373" teta="270" x="16" y="251">
    <edge «  name="Iron" referencename="SATREL" > :  <Model>
    <port anchor="8" edgeendpriotity="1" hashcode="14050342" id="2"></port>
    <port anchor="4" edgeendpriotity="0" hashcode="7792807" id="3"></port>
Modèle S1
<container hashcode="5555373" teta="270" x="16" y="251">
    <edge name="S1" referencename="SRC>  : <model>
    <port anchor="8" edgeendpriotity="0" hashcode="10769718" id="0"></port>
    <port anchor="4" edgeendpriotity="1" hashcode="24864323" id="1"></port>
Connecteurs graphiques
    <connector hashcode="15079989" x="70" y="160"></connector>
    <connector hashcode="16711231" x="230" y="160"></connector>
    <connector hashcode="12053597" x="70" y="380"></connector>
    <connector hashcode="33385450" x="230" y="380"></connector>
Liaisons entre ports & connecteurs
    <link hashcode="2091149"> <tail hashcode="24864323"></tail> <head hashcode="15079989">
    <link hashcode="19297865"> <tail hashcode="15079989"></tail> <head hashcode="16711231">
    <link hashcode="21616774"><tail hashcode="16711231"></tail><head hashcode="7792807">
    <link hashcode="9486447"><tail hashcode="10769718"></tail><head hashcode="12053597">
    <link hashcode="31664352"><tail hashcode="12053597"></tail><head hashcode="33385450">
    <link hashcode="27263487"><tail hashcode="33385450"></tail><head hashcode="14050342">
```

Figure 4-16 : Description existante d'un circuit magnétique dans RelucTool

Dans l'exemple, le modèle de réseau appelé « *Magnetic* » contient deux composants : une réluctance non linéaire IRON de type « *SATREL* » et une source de flux S1 de type « *SRC* ». Le modèle des équations associé à chaque composant est décrit entre les balises <*model*> et ses ports sont décrits grâce à la balise <*port*>.

(2) Modèle « source » proposé

Nous avons revu cette structuration topologique du modèle, en passant d'une description principalement graphique vers une description topologique de composition de modèle (Figure 4-17). Son avantage est l'obtention souple d'une représentation adéquate de la topologie du circuit sous forme d'un graphe dont les nœuds correspondent aux nœuds du circuit.

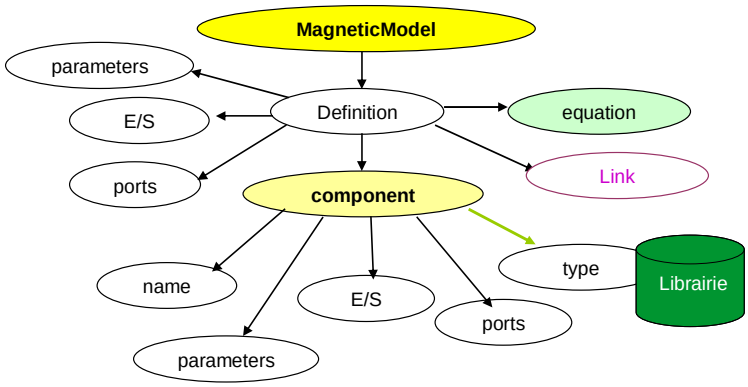


Figure 4-17 : Nouvelle structuration d'un circuit magnétique dans RelucTool

L'élément racine du schéma XML est naturellement l'élément « *magneticmodel* ». Nous déclarons que le circuit magnétique « *magneticmodel* » est composé des principaux éléments suivants (Figure 4-18) :

- une liste de paramètres, des entrées et sorties et des ports externes de connexions (électrique et magnétique, le cas de la bobine par exemple) ;
- une liste des composants : chacun a par son propre modèle défini dans la bibliothèque de composants de base ;
- une liste de la connectique entre les composants ;
- une liste d'équations qui viendront compléter le modèle (cela permet d'exprimer une force en fonction d'un flux et d'une section).

Pour chaque élément, on déclare le type à l'aide d'un attribut type et l'occurrence des éléments dans l'élément racine au moyen des attributs minOccurs et maxOccurs.

```
<xs:element name="magneticmodel">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="parameter"/>
      <xs:element maxOccurs="unbounded" ref="input"/>
      <xs:element maxOccurs="unbounded" ref="output"/>
      <xs:element maxOccurs="unbounded" ref="port"/>
      <xs:element maxOccurs="unbounded" ref="component"/>
      <xs:element maxOccurs="unbounded" ref="link"/>
      <xs:element maxOccurs="unbounded" ref="equation"/>
    </xs:sequence>
    <xs:attribute name="title" use="required" type="xs:NCName"/>
  </xs:complexType>
</xs:element>
```

Figure 4-18 : Extrait méta-modèle PIM XML-RelucTool correspondant au circuit magnétique

Pour donner un accès plus générique aux différents composants du circuit, nous avons introduit l'élément « *Component* » dont le méta-modèle est présenté sur la Figure 4-19. Nous décrivons la variable textuellement via les sous-éléments suivants :

- *name* : le nom du composant ;

- *type* : le type de composant (réluctance, source, aimant, etc.) ;
- *parameters* : l'ensemble des paramètres configurables ;
- *Entrees/sorties* : la liste des entrées sorties du composant ;
- *port* : la liste des ports de connexions ;
- *model* : le modèle associé au composant (liste des équations le définissant).

```
<xs:element name="component">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="model"/>
  </xs:sequence>
  <xs:attribute name="name" use="required" type="xs:NCName"/>
  <xs:attribute name="type" use="required" type="xs:NCName"/>
  <xs:element maxOccurs="unbounded" ref="parameters"/>
  <xs:element maxOccurs="unbounded" ref="EntreeSoties"/>
 <xs:element maxOccurs="unbounded" ref="port"/>
 </xs:complexType>
</xs:element>
```

Figure 4-19 : Extrait méta-modèle PIM XML-RelucTool correspondant au « Component »

L'annexe 4-2 présente le méta-modèle de PIM XML-RelucTool associé à cette description topologique. Elle est définie par un langage de description de document XML : XSD[73] (XML Schema Description).

Cette même représentation XML a été utilisée pour structurer la liste des équations détaillées du circuit réluctant sous la forme d'une représentation explicite et non orientée du modèle (Figure 4-15). En complément, nous avons défini les variables de position (nécessaire au mouvement avec potentiellement plusieurs degrés de liberté) et celui de la force à calculer automatiquement. La liste des équations précédentes devient un ensemble de blocs de calcul contenant des équations explicites et implicites et des fonctions. L'annexe 4-3 présente le méta-modèle en langage XSD associé à la description des équations générées par RelucTool.

C.2.2 Définition des règles de transformations

Notre problème de projection se ramène à un traitement de fichiers XML, générés par RelucTool, en définissant les règles de transformations de ces documents sources vers les modèles VHDL-AMS correspondants. Ces transformations sont des règles spécifiées dans un langage de transformation de modèle, appliquées sur le modèle source (XML) et exécutées par un moteur de transformation [KUR-03]. Le résultat est un ensemble d'instances de classes dans le modèle cible (VHDL-AMS).

Pour définir les règles de transformations du XML vers VHDL-AMS, nous avons établi un lien ou une correspondance entre leurs méta-modèles respectifs. La Figure 4-20 illustre cette correspondance.

Il existe aujourd'hui, trois approches répertoriées permettant la spécification de ces règles de correspondances [BLA-05] :

- **L'approche par programmation :** Elle consiste à utiliser les langages de programmation orientée objet pour manipuler les modèles. L'idée est de programmer une transformation de modèles de la même manière que l'on programme n'importe quelle autre application.

[73] W3C. XML Schema Part 0 : Primer, Part 1 : Structures. 2001

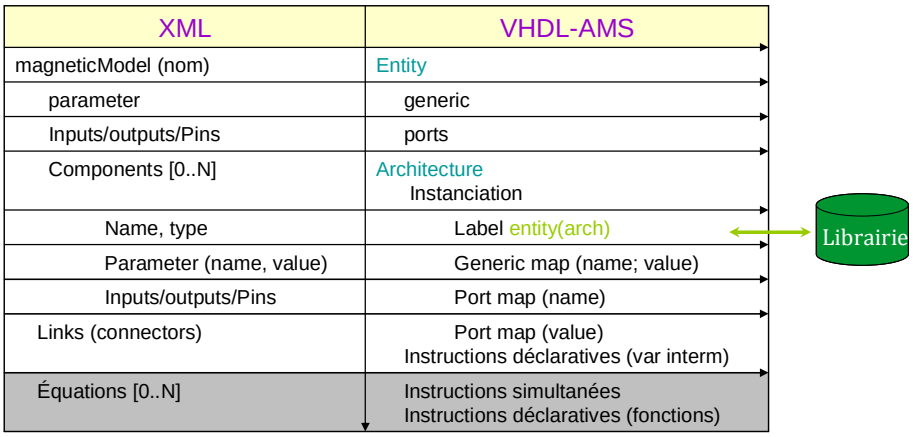


Figure 4-20 : Correspondance des modèles XML et VHDL-AMS

- **L'approche par canevas (template)** : Elle consiste à définir les canevas des modèles cibles souhaités en y déclarant des paramètres. Ces paramètres seront substitués par les informations contenues dans les modèles sources. Cette approche est très utilisée dans les développements de site Web avec des technologies comme PHP6. Elle a aussi été utilisée au G2Elab, de manière générique, avec l'outil HAGEL [ALL-03].
- **L'approche par modélisation** : Elle consiste à appliquer les concepts de l'ingénierie des modèles aux transformations des modèles. L'objectif est de modéliser les transformations de modèles et de rendre les modèles de transformation pérennes et productifs, en d'autres termes les règles de transformation sont méta-modélisées.

Nous avons commencé par une approche par canevas en utilisant le moteur de transformation XSLT[74] à base de règles. La complexité des règles, que nous obtenons pour la description topologique, nous a conduits à privilégier l'approche par programmation.

C.2.3 Mise en œuvre des générateurs de code VHDL-AMS

Grâce à tout ce que nous venons de définir, nous avons réalisé différents générateurs de modèle VHDL-AMS dans l'outil RelucTool, comme l'illustre la Figure 4-21 :

(1) un générateur de la structure du circuit pour des modèles simples, là où le concepteur peut ajouter ses équations de calcul de force manuellement,

(2) un générateur automatique des équations extraites du circuit avec le calcul automatique de la force magnétique par dérivation de la co-énergie.

(1) Générateur de la structure du circuit magnétique

La modélisation structurelle des actionneurs en langage VHDL-AMS est possible, et Cette modélisation s'effectue en deux étapes de génération (voir paragraphe B):

- de la bibliothèque de composants de base en VHDL-AMS
- de la topologie du circuit magnétique en instanciant ses éléments depuis la bibliothèque.

[74] XSLT : eXtended Stylesheet Language Transformation

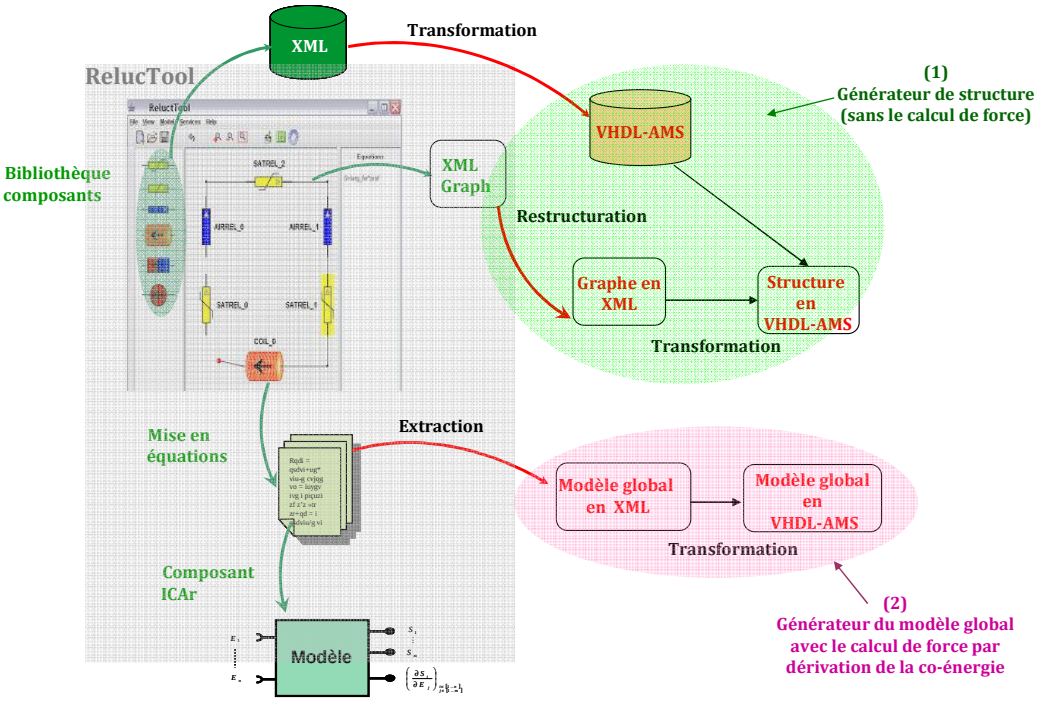


Figure 4-21 : Description des générateurs VHDL-AMS développés depuis RelucTool

- **Bibliothèque de composants**

Nous rappelons que les éléments de base utilisés dans RelucTool sont fournis dans une bibliothèque avec des modèles configurables. Chaque élément est entièrement décrit dans un fichier XML [DUP-07]. Pour que la bibliothèque des éléments en VHDL-AMS corresponde à celle de RelucTool, nous avons appliqué la technique de transformation à base de règles XSLT exploitant l'approche par canevas. Ce programme génère et met à jour automatiquement de la bibliothèque présentée au début de ce chapitre.

Un extrait des règles de transformation avec XSLT est donné sur la Figure 4-22.

```
<xsl:template match="root">

    <xsl:for-each select="MagneticComponent">

    ENTITY <xsl:value-of select="@name" /> IS
        generic (<xsl:for-each select="model">
                    <xsl:apply-templates select="param"/>
                    <xsl:apply-templates select="serial/source/param"/>
                 </xsl:for-each>
        );
        PORT (  <xsl:for-each select="model">
                    <xsl:apply-templates select="output"/>
                    <xsl:apply-templates select="reluctance"/>
                    <xsl:apply-templates select="source"/>
                 </xsl:for-each>
                 <xsl:apply-templates select="port"/>
        );
    END entity;
```

Figure 4-22 : Exemple de règles de transformation avec XSLT

La Figure 4-23 présente la transformation du modèle XML vers VHDL-AMS d'une réluctance linéaire.


```
<MagneticComponent name="AIRREL" type="reluctance" description="Airgap">
  <figure defaultWidth="22" defaultHeight="11">
    <image path="img\airrel.png"/>
  </figure>
  <port orientation="in" position="west"/>
  <port orientation="out" position="east"/>
  <model>
    <param name="L" configurable="true" value="L" unit="mm" format="1" precision="2"/>
    <param name="S" configurable="true" value="S" unit="mm2" format="1" precision="2"/>
    <reluctance name="R_@MODELNAME" intern="false" value="1/mu0*1000*L/S" fluxname="flux_R_@MODELNAI
    unit="H-1" format="1" precision="2"/>
    <output name="energy_R_@MODELNAME" intern="false"
    value="0.5*R_@MODELNAME*pow(flux_R_@MODELNAME,2)" unit="J" format="1" precision="2"/>
    <output name="coenergy_R_@MODELNAME" intern="false" value="energy_R_@MODELNAME" unit="J" format=
    precision="2"/>
    <output name="losse_R_@MODELNAME" intern="true" value="0.0"/>
```

```
ENTITY AIRREL IS
    generic ( L : real := 1.0  ;    S : real := 1.0);
    PORT (    quantity energy_R: out : real;
              quantity coenergy_R: out : real;
              quantity R: out : real;
              terminal pin: magnetic ;
              terminal pout: magnetic );
END entity;
ARCHITECTURE behav OF AIRREL IS
    constant  mu0 : real := 1.0e-7;
     quantity  mmf across  flux_R  through  pin to  pout ;
BEGIN
     energy_R==0.5 * R * flux_R  ** 2
     coenergy_R==energy_R;
     R==1/mu0*1000.0*L/S;
     mmf== R*flux_R;
END architecture behav;
```

Figure 4-23 : Traduction du modèle XML en modèle VHDL-AMS d'une reluctance linéaire

- **Description du générateur de la structure en VHDL-AMS**

Dans sa version initiale, l'outil RelucTool génère une description d'un réseau réluctant en XML. Nous lui avons ajouté la génération automatique de la description XML respectant la structuration topologique que nous avons proposée. Ensuite, notre outil de génération automatique de modèle VHDL-AMS exploite et analyse cette description XML afin de génère le modèle VHDL-AMS correspondant. La Figure 4-24 décrit le fonctionnement du générateur tel qu'il a été développé, respectant les correspondances définies sur la Figure 4-20.

Notre générateur comporte essentiellement trois modules :

(1) Nous avons mis en place la génération du fichier XML de la nouvelle représentation du réseau réluctant à partir de celle générée initialement dans RelucTool. Ceci s'effectue grâce à un parseur et un analyseur du fichier XML, ainsi qu'un parseur d'expressions mathématiques ; permettant d'analyser et d'extraire les informations utiles.

(2) Nous appliquons des règles de transformation sur la représentation XML créée, permettant la génération d'une entité et une architecture pour chaque circuit magnétique.

- Nous remplissons l'entité par les éléments définis dans le fichier XML généré :
  - (i) le nom du modèle et ses paramètres,
  - (ii) les ports de connexions et des entrées/sorties.

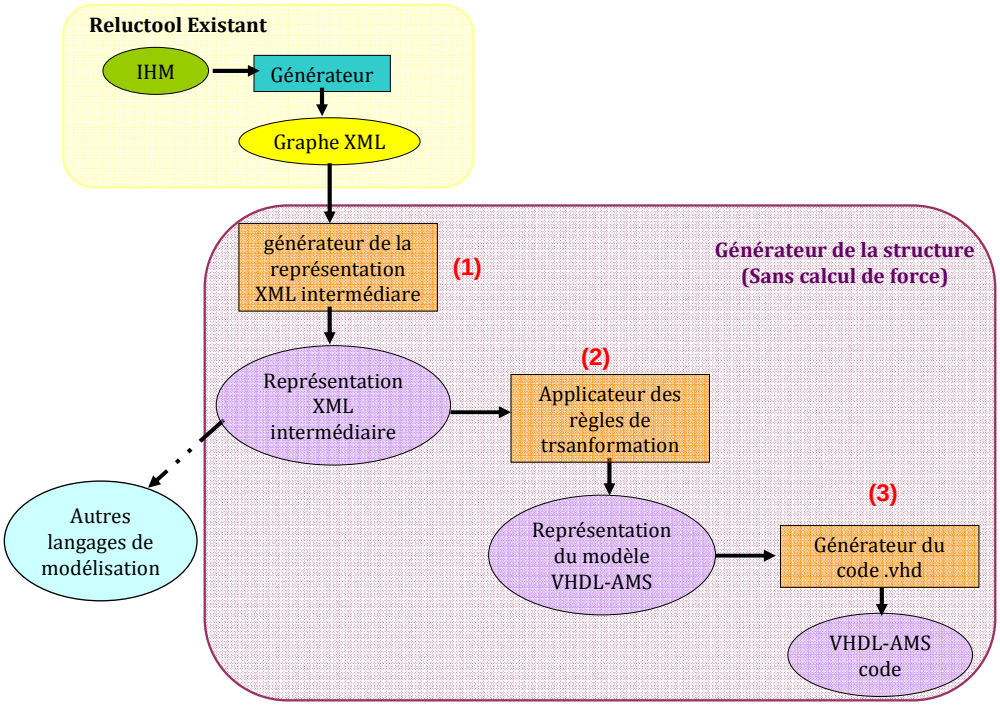


Figure 4-24 : Principe de projection et fonctionnement du générateur de la structure

- Nous générons l'architecture avec la topologie du modèle. Les différents modèles de composants du circuit réluctant décrits dans le fichier XML sont instanciés et complétés par leurs propres paramètres et les ports correspondants.
- Nous créons la connexion entre les ports du composant global définis dans l'entité et les ports réels de l'architecture qui le contient.
- Nous ajoutons les équations supplémentaires à l'intérieur de son architecture.

(3) Une fois que la structure du modèle VHDL-AMS est remplie, l'écriture du code VHDL-AMS est complétée par un ensemble de règles permettant de créer un code conforme à la syntaxe du langage.

En ce qui concerne le calcul de la force, nous pouvons intervenir sur le modèle généré pour ajouter manuellement l'expression simplifiée.

- **Difficultés rencontrées**

Il est à noter qu'il est nécessaire de reformuler certaines expressions mathématiques afin de les adapter au langage cible. Par exemple, la dérivée temporelle d'une variable x est définie par x'dot en VHDL-AMS et par der(x) en Modelica. Ainsi, nous avons utilisé un parseur d'expressions mathématiques et un applicateur de règles mathématiques RAMA ([ALL-03] [FIS-04]). RAMA est chargé d'analyser une expression mathématique donnée et d'y effectuer des calculs ou transformations formelles par application de règles génériques. Cela nous permet d'effectuer une bonne correspondance entre les expressions mathématiques décrites en langage JAVA et XML et celles définies dans le langage VHDL-AMS. Par exemple, la fonction exposant x^y s'écrit *Math.pow(x,y)* en java et x**y en VHDL-AMS.

(2) Générateur du modèle analytique global du circuit magnétique

Ce générateur de modèle analytique global d'un réseau réluctant permet d'exporter, en langage VHDL-AMS, un système magnétique en une seule entité et une seule architecture dont le comportement dynamique est définie par une liste d'équations et de fonctions.

- **Description du générateur du modèle global en VHDL-AMS**

Dans RelucTool, toutes les équations générées constituent un système implicite (à cause des expressions de flux). Toutes les dérivées formelles de flux sont créées ensuite et la dérivée de tout le système d'équations est générée pour compléter le modèle avec les expressions de la force magnétique exacte. On obtient un ensemble d'équations algébriques (en magnétostatique) ou différentielles algébriques DAE (en dynamique). Le système d'équations obtenu est un ensemble d'équations explicites et implicites (en fonction de flux). En outre, des fonctions spécifiques utilisées décrivant le comportement des matériaux (courbe H (B)) ou la démagnétisation (courbe B (H)) peuvent aussi apparaître dans ce système d'équations.

Nous avons proposé de récupérer le modèle analytique intermédiaire, suite à la mise en équations du réseau réluctant avec la dérivation automatique ; afin d'avoir accès à l'expression de la force magnétique. Nous avons proposé, ensuite, de le sauvegarder dans un fichier XML respectant la structuration des données qu'on a mise en place.

Nous avons ainsi crée un générateur VHDL-AMS d'un modèle réluctant sous sa forme analytique global proposant un calcul exact de la force magnétique. La Figure 4-25 décrit le fonctionnement de notre générateur tel que nous l'avons développé. Il comporte essentiellement quatre modules :

(1) un générateur XML qui stocke le système d'équations implicites généré par RelucTool, avant son ordonnancement pour la résolution des équations.
(2) un parseur XML qui manipule les données XML pour les transformer en un arbre. Il sera le point d'entrée du processus de génération de code ; qui extrait les informations nécessaires du fichier source avant transformation (paramètres, E/S, etc.).
(3) un ensemble de transformations basées sur la correspondance XML vers VHDL-AMS. Nous traitons les équations et les fonctions par le parseur d'expressions et applicateur de règles de RAMA [FIS-03].
(4) l'écriture du code VHDL-AMS est complétée par un ensemble de règles permettant de créer le code conforme à la syntaxe du langage.

- **Description du modèle VHDL-AMS**

Le code VHDL-AMS crée depuis RelucTool par le générateur, que nous avons développé, possède les caractéristiques suivantes (non exhaustives).

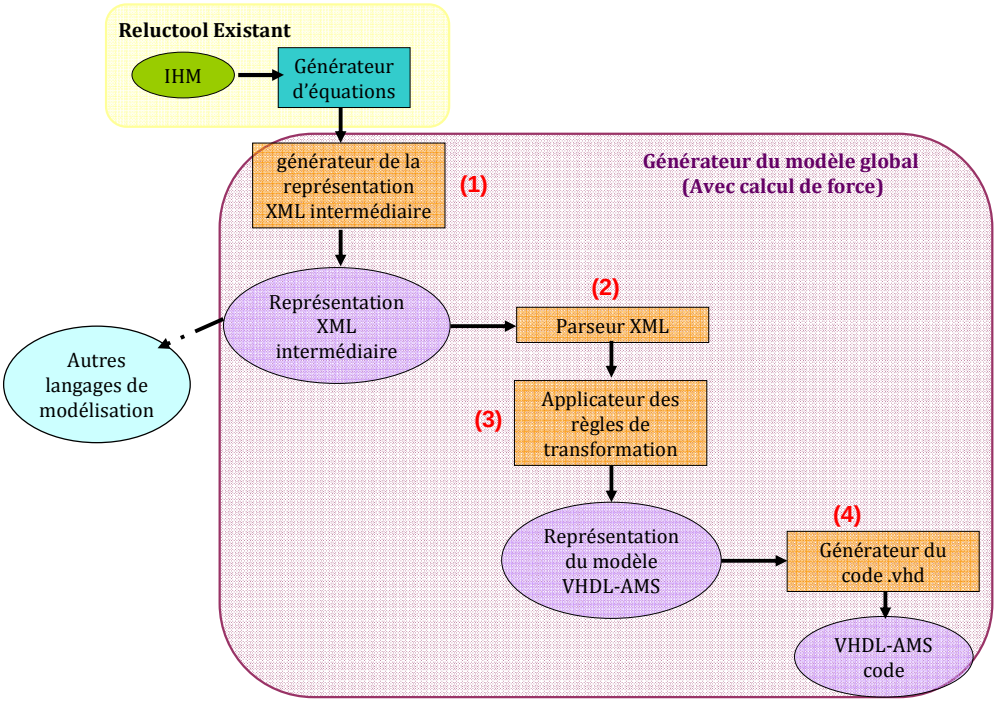


Figure 4-25 : Principe de projection et fonctionnement du générateur du modèle analytique

- Une entité et une architecture sont générées pour chaque circuit magnétique. Les paramètres géométriques qui existent dans le modèle sont transformés en *GENERIC*.
- Les forces appliquées sur le noyau mobile se transforment en port de sortie *QUANTITY*.
- Toutes les équations sont transformées en des déclarations *simultanées* et les fonctions en des déclarations de *fonction* dans la partie déclarative de l'architecture.
- Une instruction *BREAK* définit les conditions initiales. En option, il est également possible d'initialiser le système en utilisant la méthode *quiescent_domain*.

La Figure 4-26 présente les fichiers XML source et VHDL-AMS généré dans le cas d'un contacteur en « E », qui sera étudiée plus en détail dans le chapitre 5.

1) Générateur XML

```
<parameterList > .....
<positionvariable name="L" unit="mm" />
<forcevariable name="force" unit="N" />
```

Système d'équations

```
<equation "R_AIRREL_0 = (1.0/mu0)*1000.0*(L/S)" priority="3" />
<function "_impFunc_0(rltW0) = -(R_AIRREL_0*rltW0)+NI_S1" />
<equation "flux_R_AIRREL_0=rltW0" />
<equation "energy_R_AIRREL_0 = 0.5*R_AIRREL_0*pow(flux_R_AIRREL_0, 2.0)" />
<equation "system_energy=+energy_R_AIRREL_0" />
<equation "coenergy_R_AIRREL_0 = energy_R_AIRREL_0" priority="0" />
<equation "system_coenergy=+coenergy_R_AIRREL_0" intern="false" priority="0" />
```

Dérivée du système d'équations

```
<" function drond_impFunc_0drondX0(rltW0) = -R_AIRREL_0" />
<" function drond_impFunc_0drondL(rltW0) = -((1.0/mu0)*1000.0*(1.0/S)*rltW0)" />
<equation "wqS_flux_R_AIRREL_0wqE_L= (-1.0/drond_impFunc_0drondX0(rltW0))..
<equation "wqS_R_AIRREL_0wqE_L=(1.0/mu0)*1000.0*(1.0/S)" />
<equation "wqS_energy_R_AIRREL_0wqE_L=0.5*...../>
<equation "wqS_coenergy_R_AIRREL_0wqE_L=wqS_energy_R_AIRREL_0wqE_L" />
```

Force

```
<equation "force=-1000*wqS_system_coenergywqE_L" priority="0" />
```

2) Générateur VHDL-AMS

```
ENTITY Actuator is
   generic (……..)
   port (……..;  terminal pe1, pe2 : electrical;
        Quantity Force : out real )
end entity;

ARCHITECTURE  Behav  of Actuator is

  liste de variables intermédiaires
  liste des fonctions
  liste des dérivées des fonctions

BEGIN
        -- systeme d'equations
        Liste des équations explicite
        Liste des équations implicites

        -- derivée du systeme d'equations
        Liste des équations dérivées explicite
        Liste des équations dérivées implicites

-- formule de la force :
Fmag == d_coenergy_posvar;

end architecture Behav ;
```

Figure 4-26 : Descriptions XML et VHDL-AMS traitées par le générateur du modèle analytique

- **Difficultés rencontrées**

Dans la réalisation de ce générateur, nous avons rencontré une difficulté liée à la dérivation du système implicite faite par RelucTool pour le calcul de la force. Celle-ci n'est pas explicitée par l'outil. En effet, nous ne connaissons pas à l'avance par rapport à quelle variable de déplacement nous souhaitons dériver la co-énergie. RelucTool dérive celle-ci lors de la création du composant logiciel ICAr, sans passer par l'intermédiaire XML utilisé pour le reste du modèle. Ainsi, la dérivation du système d'équations et le calcul de la force sont implémentés dans le cœur du calcul de l'ICAr généré (Figure 1) [DUP-06] et sont masqués pour l'utilisateur, mais aussi développeurs de l'outil. Dans notre générateur du modèle XML, nous avons dû recoder toute la partie de dérivation du système implicite et le calcul de la force magnétique sous forme d'équations purement algébriques analytiques.

C.2.4 Conclusions sur la projection de RelucTool en VHDL-AMS

Nous venons de présenter et justifier les raisons qui nous poussent à penser que le langage VHDL-AMS permet aux concepteurs de gérer des systèmes complexes dans le cadre de la modélisation des systèmes électromagnétiques par réseau de réluctances.

Dans ce contexte, nous avons étudié la transformation d'une description de modèles par réseau de réluctances (d'un langage propriétaire de RelucTool) vers le langage VHDL-AMS à travers des concepts de l'ingénierie dirigée par les modèles. L'outil métier facilite grandement la mise en œuvre de ces modèles réluctants ; nous avons spécifié, formalisé et mis en œuvre la génération automatique de ses modèles sous deux représentations VHDL-AMS pour des actionneurs linéaires :

- La première représente une composition de modèle de composants issus d'une bibliothèque VHDL-AMS générée à partir de RelucTool.
- La seconde crée un modèle global des dispositifs, permettant ainsi une modélisation aisée des forces magnétiques par dérivation de la co-énergie.

Il est à noter ici que quel que soit le langage de modélisation désiré en sortie (VHDL-AMS, Modelica, Simscape), les procédures de projection que nous avons proposées ainsi que les étapes de transformations (outils, procédures) sont les mêmes. L'avantage de partir d'une structuration XML générique est de permettre de réaliser le générateur du langage désiré indépendamment de l'outil source : les règles de transformations diffèrent uniquement.

D. Modélisation des microsystèmes magnétiques en VHDL-AMS

Les microsystèmes magnétiques sont un sous-ensemble des MEMS (Micro ElectroMechanical System). Ils sont des dispositifs électromagnétiques multi-physiques de petite taille ; permettant d'exécuter des fonctions de mesure ou des transformations d'énergie. Les MEMS magnétiques sont des dispositifs magnétiques à champs magnétiques rayonnés (non canalisés par des matériaux ferromagnétiques) ; ils ne peuvent pas être décrits par des tubes de flux comme les réseaux de réluctance.

Nous disposons au G2Elab d'un outil dédié pour la modélisation des MEMS magnétiques appelé MacMMems (Macromodeler For Magnetic Mems). C'est un générateur métier de macro-modèles à partir d'une description 3D. Il a été développé durant la thèse de H. L. Rakotoarison au G2Elab [RAK-07] ; permettant de capitaliser les modèles, qui sont projetable en composant logiciel ICAr utilisable par le framework CADES [DEL-12] pour les besoins de la simulation ou de l'optimisation (chapitre 1).

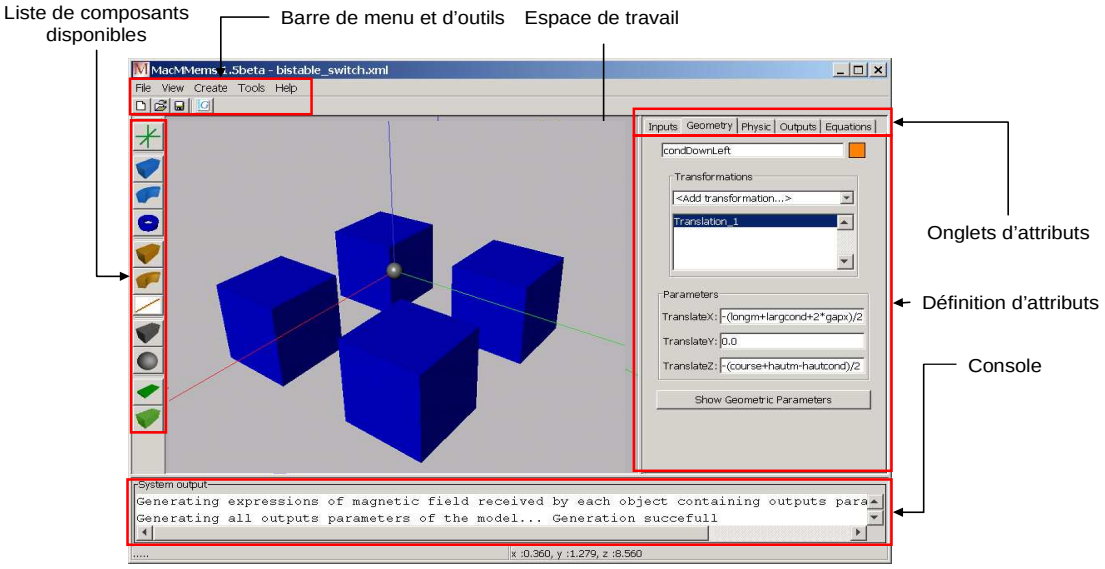


Figure 4-27 : L'environnement de MacMMems

Nous intéressons à la description des modèles des MEMS magnétiques en VHDL-AMS, dans l'optique de proposer des générateurs automatiques de modèles VHDL-AMS de ces composants depuis l'outil MacMMems. Les étapes de la projection à suivre sont identiques à celui de RelucTool (paragraphe C).

Notre objectif dans cette partie est de proposer une démarche cohérente de projection et nous allons étudier les différentes phases nécessaires pour la mise en œuvre de ce générateur, même si nous ne l'avons pas développé.

Nous allons commencer par décrire les MEMS magnétiques et l'approche de modélisation qui leur est dédiée. Ensuite, nous netterrons l'accent sur comment formaliser la description VHDL-AMS des systèmes électromagnétiques modélisés par champs rayonnés. Nous proposerons des solutions aux problèmes liés à des limites du langage lui-même concernant la composition de modèles et le calcul d'intégrales non temporelles.

D.1. Les MEMS et l'approche de composition dédiée

D.1.1 Définition d'un composant magnétique

En présence d'un champ magnétique d'excitation quelconque, un composant magnétique est soumis à une force et/ou à un couple. Il peut être aussi amené à s'aimanter et à produire lui-même un champ magnétique qui dépend ou non du champ excitateur. Chaque composant magnétique peut être représenté comme illustré sur la Figure 4-28.

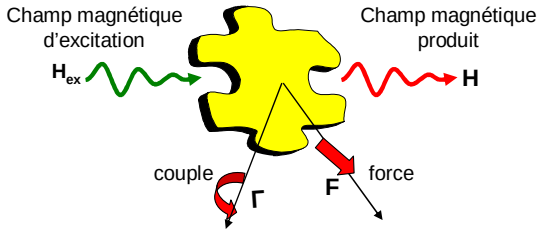


Figure 4-28 : Représentation d'un composant MEMS magnétique [RAK-07]

Ces composants magnétiques peuvent être réparties en trois classes d'éléments

- des sources de champ magnétique (aimants, conducteurs) ;
- des matériaux ferromagnétiques doux ;
- des matériaux diamagnétiques et paramagnétiques (champ induit négligé).

D.1.2 Les modèles des composants magnétiques

Les formules intégrales, de calcul de champ magnétique qu'un composant magnétique produit, la force et le couple qu'il subit et l'influence du champ magnétique d'excitation sur ces trois grandeurs, dépendent de la nature des composants (conducteurs, aimants permanents, matériaux diamagnétiques ou paramagnétiques). Ces formules peuvent être intégrées formellement complètement ou partiellement sur des géométries simples (pavés, cylindres, etc.). Elles donnent des formules analytiques ou semi-analytiques (voir [RAK-07]) exploitées par le logiciel MacMMems. Nous ne détaillons pas ici les composants ferromagnétiques dont la méthode numérique de modélisation est plus complexe.

(1) Les aimants

Avec l'hypothèse de rigidité, un aimant est caractérisé par son aimantation *M*. Le calcul de champ créé par les aimants s'appuie sur une approche coulombienne (charge équivalente) et revient en général à un calcul d'intégrale surfacique, parfois volumique.

$$\vec{H}(\vec{r}) = \iint_S \frac{\sigma_S \times (\vec{r} - \vec{r}')}{|\vec{r} - \vec{r}'|^3} ds + \iiint_V \frac{\sigma_V \times (\vec{r} - \vec{r}')}{|\vec{r} - \vec{r}'|^3} dv \qquad \text{Eq 4-20}$$

avec $\sigma_S = \vec{M}.\vec{n}$ et $\sigma_V = -\mathrm{div}(\vec{M})$

où - le vecteur $\vec{r}$ représente la position du point de calcul,

- le vecteur $\vec{r}'$ définit celle de l'élément volumique *dv* et/ou l'élément surfacique *ds*.

Le calcul de la force appliquée sur un aimant est obtenu en appliquant l'approche coulombienne en présence du champ extérieure H_{ext}.

$$\vec{F} = \iint_{s} \sigma_s . \overrightarrow{H_{ext}} ds + \iiint_{v} \sigma_v . \overrightarrow{H_{ext}} dv \qquad \text{Eq 4-21}$$

La Figure 4-29 présente un aimant de forme circulaire à aimantation radiale.

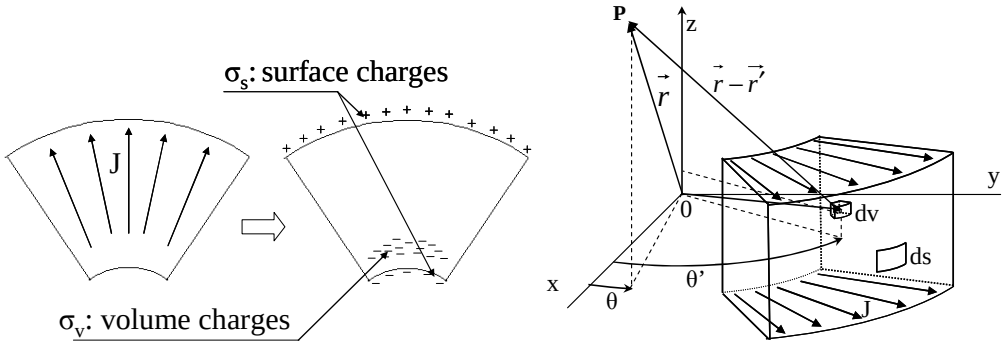


Figure 4-29 : Aimant de forme circulaire à aimantation radiale [RAK-07a].

(2) Les conducteurs

Le calcul de champ créé par des conducteurs en tout point de l'espace s'appuie sur la loi de Biot et Savart conduisant à des calculs d'intégrale volumique.

$$\vec{H}(\vec{r}) = \frac{1}{4\pi} \iiint_{v} \frac{j \times (\vec{r} - \vec{r}')}{|\vec{r} - \vec{r}'|^3} dv \qquad \text{Eq 4-22}$$

où *j* correspond à la densité surfacique de courant.

Le calcul de la force appliquée sur un conducteur est obtenu en appliquant la loi de Laplace

$$\vec{F} = \mu_0 . \iiint_{v} \vec{j} \times \overrightarrow{H_{ex}} dv \qquad \text{Eq 4-23}$$

où H_{ex} est le champ d'excitation.

(3) Les diamagnétiques et paramagnétiques

Les matériaux diamagnétiques et paramagnétiques sont des composants qui commencent à émerger dans les dispositifs microsystèmes. Ils sont caractérisés par la génération d'un champ induit négligeable en présence d'un champ excitateur. On trouve une utilisation potentielle de ces matériaux dans le domaine de la lévitation passive [CHE-07] |KAU-09].

Ces matériaux sont caractérisés par leurs successibilités magnétiques χ et le calcul de la force est donné par la formule (4-25)

$$\vec{F} = \frac{\mu_0}{2} . \iint_{s} [\chi] . \vec{n} . \left| \overrightarrow{H_{ext}} \right|^2 ds \qquad \text{Eq 4-24}$$

ou le vecteur *n* désigne la normale à la surface.

D.1.3 Description d'un MEMS par composition

Les MEMS sont généralement réalisés avec des formes géométriques simples (parallélépipèdes, secteurs cylindriques et sphères) pour faciliter leur fabrication. De ce fait, un MEMS peut être décomposé en plusieurs entités basiques modélisées analytiquement. Une approche de modélisation comportementale a été proposée et implémentée dans MacMMems

par L. Rakotoarison ; elle est basée sur le principe de composants en interaction forte [RAK-07]. Nous avons retenu cette approche de composition dans nos travaux, qui présente une modélisation par assemblage de composants (aimants, conducteurs, matériaux diamagnétiques, paramagnétiques, ferromagnétiques).

Le principe d'assemblage consiste à décomposer le dispositif en plusieurs composants de base (dotés chacun de propriétés géométriques et physiques). Ensuite, on les assemble pour modéliser analytiquement une grandeur caractérisant le microsystème (le couple ou la force appliquée sur une partie du dispositif, le champ magnétique calculé en un point de l'espace, la tension induite suite à une variation du flux magnétique, etc.).

On note que pour évaluer la force ou le couple sur un composant, on doit impérativement avoir en entrée le champ magnétique créé par ses voisins. L'état magnétique interne du composant (son aimantation) est nécessaire aussi pour le calcul des sorties mais n'apparaît pas en sortie. Pour cela, on identifie les composants qui permettent l'évaluation de cette grandeur caractéristique, qu'on appellera les « cibles ». Les composants restants contribuant à la modification de l'état des composants cibles sont appelés les « sources » [RAK-07]. La Figure 4-30 illustre un exemple de cette composition pour un dispositif MEMS.

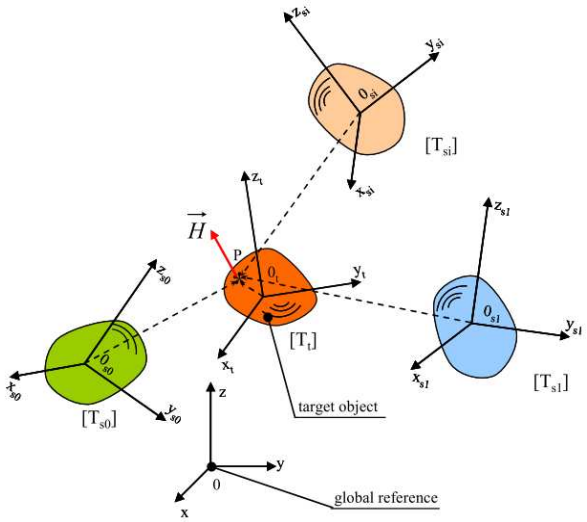


Figure 4-30 : Composant cible sous l'influence du champ produit par des composants sources [RAK-07]

D.2. Étude de la description des modèles VHDL-AMS des MEMS

Modéliser un microsystème magnétique nécessite la description de sa composition, telle que nous venons de la définir. Sa modélisation VHDL-AMS s'effectue de deux manières :

- soit d'une manière structurelle (instanciation de composants de base) ;
- soit sous forme d'un ensemble d'équations analytiques modélisant le système global.

Afin de bien comprendre ces modélisations et les difficultés associées, nous allons illustrer nos propos sur un dispositif didactique de lévitation diamagnétique : un matériau diamagnétique (gouttelette d'eau ou des billes de latex) interagissant avec quatre aimants permanents (parallélépipédiques identiques) aimantés vers le haut [CHE-07].

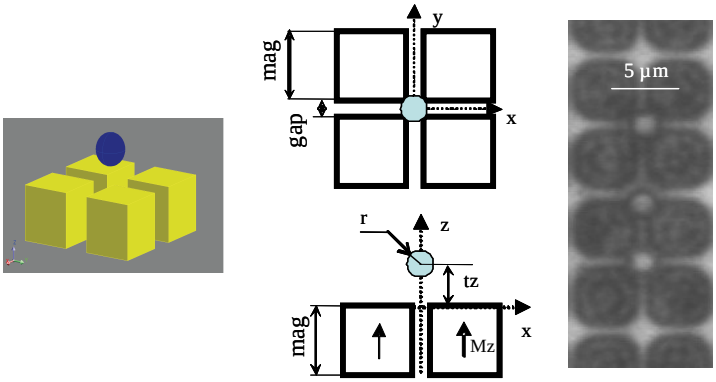


Figure 4-31 : Dispositif à lévitation diamagnétique, avec une photo d'un réseau d'aimants pour matricer des billes de latex [CHE-07]

D.2.1 Analyse de la modélisation structurelle d'un MEMS

Une modélisation structurelle des MEMS en VHDL-AMS s'effectue en deux étapes :

- création de la bibliothèque de composants élémentaires (capteurs de champs, aimants, conducteurs, matériaux diamagnétiques et des matériaux ferromagnétiques) ;
- création de la composition par instanciation des éléments de la bibliothèque.

(1) Bibliothèque de composants magnétiques en VHDL-AMS

Un composant magnétique est défini par ses paramètres géométriques, un champ d'excitation en entrée et différentes grandeurs en sortie essentiellement les expressions du champ induit, de la force et du couple. Cependant, ces formulations requièrent des intégrations spatiales qui sont difficilement exprimables en VHDL-AMS.

Toutefois, pour des géométries simples, ces calculs de champs peuvent être exprimés par des formules analytiques (un ensemble de sommation d'équations purement algébriques). Illustrons ceci sur l'exemple de la lévitation à quatre aimants parallélépipédiques :

- le modèle d'aimant parallélépipède est décrit par une formule purement algébrique,
- comme le champ d'excitation *Hex* n'est pas connu à l'avance, le calcul des forces (Eq. 4-24) ne peut pas être défini à l'avance et l'intégrale doit être résolue numériquement.

Ainsi, nous ne pouvons réaliser la bibliothèque des composants magnétiques de base que lorsque nous résoudrons la limite du VHDL-AMS à décrire des intégrales spatiales.

(2) Composition d'un MEMS magnétique en VHDL-AMS

En supposant qu'on dispose de la bibliothèque des composants magnétiques élémentaires, nous allons regarder comment gérer la composition du microsystème (éléments en interaction forte) à travers le langage VHDL-AMS.

Nous rappelons que l'approche de modélisation par composants en interaction consiste à assembler les composants en identifiant les *cibles* et les *sources* contribuant à la modification de l'état des composants cibles (Figure 4-30). Dans l'exemple de la lévitation, comme le montre la Figure 4-32, la bille sphérique (la cible) est maintenue en lévitation par la force magnétique produite par les champs magnétiques crées par les aimants (composants sources). Les interactions entre les différents aimants et la bille sont de nature unidirectionnelle ; d'une

manière plus générale, les interactions sont de nature multidirectionnelles (entre les aimants eux même par exemple). Pour calculer la force appliquée sur la bille, un certain nombre d'informations sur ces composants sont nécessaires, tels que les positions spatiales relatives, les propriétés physiques, etc.

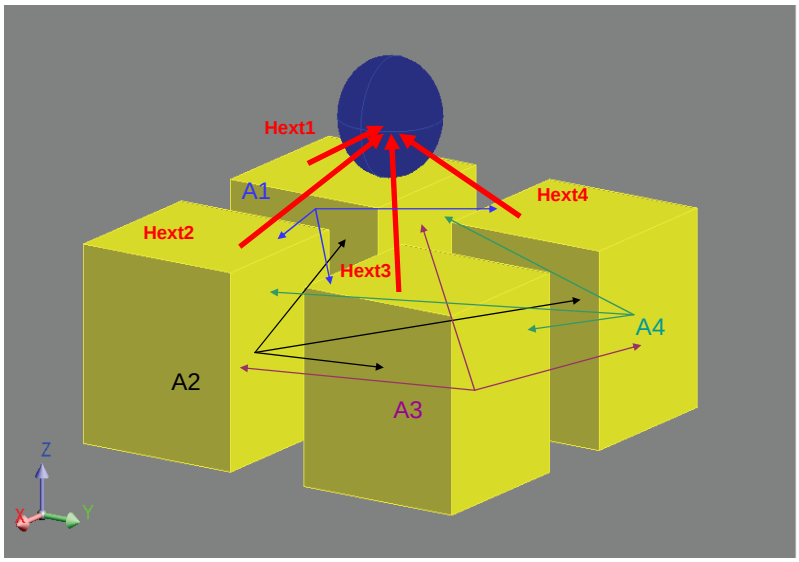


Figure 4-32 : Interactions entre les différents éléments dans le cas d'une bille en lévitation

On se retrouve donc confronté à résoudre une interaction qui dépend des positions et orientations d'espace d'une part et qui n'est pas point à point comme l'approche nodale supportée par le langage VHDL-AMS (chapitre 2).

(3) Conclusion sur la modélisation structurelle

En résumé, les composants interagissent entre eux indépendamment au travers des expressions intégrales, qui ne peuvent pas être exprimées sans la connaissance des autres composants. Une description modulaire des composants d'un MEMS est donc difficile en VHDL-AMS car elle n'explicite pas cette interaction forte.

Nous avons opté pour la modélisation VHDL-AMS des MEMS en exploitant une approche analytique globale générée à l'issue de la composition précédente.

D.2.2 Analyse de la modélisation analytique global d'un MEMS

Pour développer le modèle analytique de la lévitation, l'hypothèse qui néglige le champ induit dans un corps diamagnétique peut être considéré ; puisque les matériaux diamagnétiques ont une susceptibilité magnétique très faible (leur perméabilité relative est très proche de 1). Le modèle analytique de la lévitation comporte ainsi les étapes suivantes :

- calcul du champ magnétique émis par chaque composant source (aimant) dans son propre repère ;
- changement de repère vers celui du composant cible (la bille) ;
- calcul des champs magnétiques émis par les sources dans le repère de la cible ;
- calcul du champ magnétique global reçu par la cible (champ excitateur) en appliquant le théorème de superposition : la somme des champs émis par les aimants ;
- calcul des sorties du modèle : dans notre cas la force appliquée sur la bille.

Le champ magnétique créé par les aimants parallélépipédiques peut être conduit formellement jusqu'à l'obtention d'équations algébriques [AKO-84], c'est ce qu'implémente le logiciel MacMMems. Le passage d'un repère à un autre s'exprime aussi à travers des fonctions algébriques. L'ensemble de ces équations peut être exprimé en langage VHDL-AMS en s'appuyant sur les déclarations simultanées ou des fonctions. Ainsi, seul le calcul de la force peut poser problème dans cet exemple, en raison de sa formulation intégrale.

Un autre problème principal de l'exportation de ces équations en VHDL-AMS est d'écrire d'une manière générique un algorithme d'intégrale non temporelle et d'appeler dynamiquement les fonctions fournies en argument, comme l'illustre la Figure 4-33. Effectuer ce calcul intégral en VHDL-AMS nécessite la réalisation d'une fonction ou d'un composant spécifique ; qui prend en argument une fonction *f* à intégrer (un objet de type pointeur de fonction) et qui implémente l'algorithme d'intégration numérique.

Nous allons expliciter les problèmes apparaissant dans la modélisation des MEMS :

- l'appel dynamique d'une fonction
- l'algorithme d'intégration spatiale

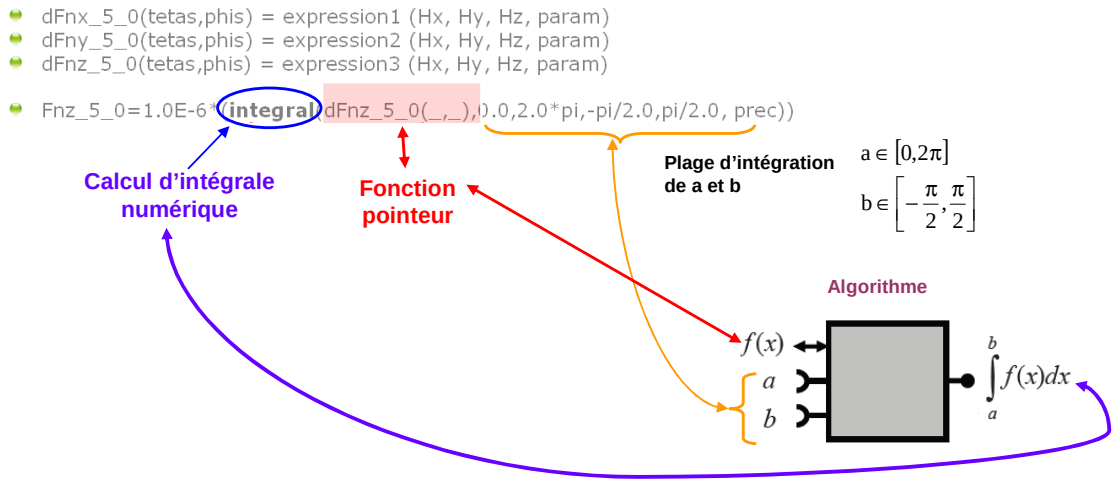


Figure 4-33 : calcul d'intégral pour les MEMS (en syntaxe SML de CADES)

(1) Problématiques rencontrées dans la modélisation des MEMS

o Appel dynamique d'une fonction

En informatique, il existe différentes méthodes pour appeler dynamiquement une fonction :

(1) un pointeur vers une fonction (on remplace le nom de la fonction par son adresse),

(2) une compilation dynamique (le code est compilé à chaque fois qu'on en a besoin).

Il faut bien noter que le langage VHDL-AMS n'est prévu que pour décrire et concevoir des systèmes physiques. Les notions purement informatiques ont volontairement été séparées afin de distinguer l'aspect métier de modélisation de l'aspect implémentation informatique, comme le préconise l'ingénierie dirigée par les modèles. Par conséquent, la notion d'accès par pointeur est considérée dans le langage pour accéder aux données uniquement, ne disposant donc pas de la notion de comportement dynamique ou d'instanciation. Ainsi, la méthode (1) n'est pas prévue en VHDL-AMS. Si une fonction appelle une autre fonction, la fonction englobante travaillera sur le résultat de la fonction englobée.

La méthode (2), quant à elle, implique qu'une partie du modèle soit écrite dans un langage compilé dynamiquement ou interprété et qu'à chaque évaluation par le solveur cette fonction soit effectivement recompilée.

- **Calcul et résolution d'une intégrale spatiale**

En ce qui concerne le traitement des intégrales spatiales, elles peuvent être calculées en utilisant un algorithme numérique. Par exemple, on peut utiliser la méthode des trapèzes ou une quadrature de Gauss. Il est également envisageable de revenir sur la modélisation physique et d'exprimer la force par un développement multipolaire par exemple, faisant intervenir un ensemble de dérivations spatiales successives [KAU-09]. Toutes ces méthodes sont réalisables en langage VHDL-AMS avec plus au moins de difficulté. Le choix de l'algorithme à implémenter dépend fortement de la précision souhaitée par le concepteur.

(2) Solutions que nous proposons pour cette problématique

Différentes solutions peuvent être envisagées pour résoudre les deux principaux problèmes évoqués précédemment en VHDL-AMS. Nous allons les présenter brièvement :

- **Instruction simultanée « Procedural »**

L'instruction « Procedural » permet de fabriquer des instructions simultanées à partir d'actions séquentielles. Cependant, les variables locales dans un PROCEDURAL sont dynamiques et redéfinies à chaque exécution (redéclarées, réinitilisées). Ce mécanisme ne permet donc pas de garder une mémoire entre deux exécutions [GUE-06]. Or, le code de l'algorithme d'intégration spatiale nécessite son évaluation à chaque pas de temps de calcul. On ne peut donc pas s'appuyer sur cette caractéristique.

- **Instruction « Generate »**

Nous pouvons aussi envisager d'écrire le code de l'algorithme d'intégration numérique avec des méthodes analytiques (uniquement des instructions algébriques) ou la méthode du développement multipolaire en exploitant conjointement les approximations des équations aux dérivées partielles (PDE) avec la génération d'instructions (*GENERATE* [75] statement). En d'autres termes les instructions concurrentes ou simultanées seront dupliquées en fonction du type d'instruction [ASH-02] [HER-02]. Afin que cet algorithme puisse être générique et utilisable automatiquement pour différentes fonctions, nous pouvons définir une séquence d'exécution des fonctions à l'aide de l'instruction *GENERATE*. Par exemple, on peut définir la structure suivante :

```
If fct=1 generate
    result == fct_1(x,y,z)   ;
end generate   ;
If fct=2 generate
    result == fct_2(x,y,z)   ;
end generate   ;
```

[75] L'instruction GENERATE permet l'écriture automatique itérative ou conditionnelle d'un code. La génération a lieu à l'élaboration, avant le début de la simulation.

Cette structure pourrait être écrite une fois pour toute dans un objet *COMPONENT* qui s'appellerait *INTEGRAL* dans un *PACKAGE*. L'interface de cet objet *COMPONENT* permettrait de transmettre les informations de calcul. Il pourrait être mis à jour au fur et à mesure qu'on avance dans la détermination de nouveaux comportements.

o Duplication de l'algorithme simplifié d'intégrale par fonction

Nous savons que, dans la description des MEMS, les expressions analytiques des fonctions de champs ou autres grandeurs ne sont pas facilement connues par le concepteur à l'avance. Ceci rend la méthode précédente délicate à mettre en œuvre. Cependant, on peut envisager de dupliquer le code de l'algorithme de l'intégration numérique précédant pour chaque fonction appelée. Ainsi, comme l'illustre la Figure 4-34, pour chaque appel de l'algorithme d'intégration_A on peut créer:

- dans le premier cas de figure, une seule fonction VHDL-AMS (appelée « Intergration_A » sur la figure) qui sera appelée pour intégrer n'importe quelle fonction F1 ou F2. Ceci n'est pas faisable.
- dans le second cas de figure, une fonction VHDL-AMS spécifique à l'intégration de chaque fonction (une pour F1 et une autre pour F2), même si c'est le même algorithme d'intégration qui est utilisé pour les deux fonctions.

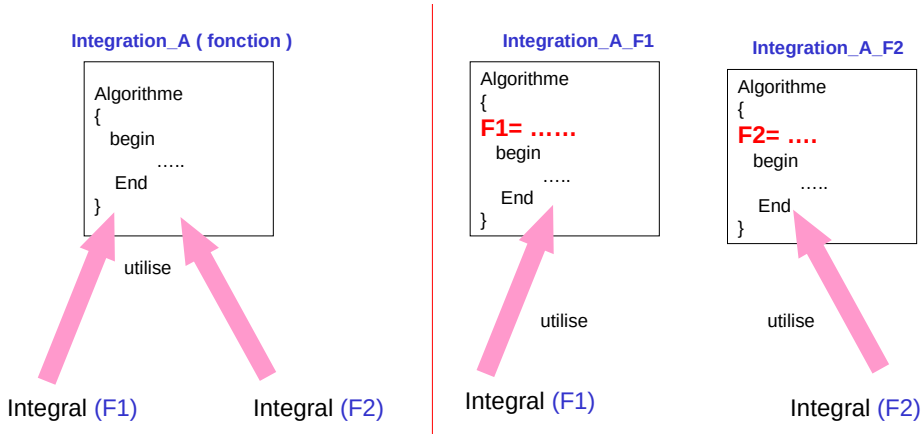


Figure 4-34 : duplication de l'algorithme d'intégration pour différentes fonctions

Reprenons l'exemple de la bille en lévitation (Figure 4-33), l'équation de calcul de la force donnée précédemment par l'équation suivante :

Fnz_5_0 == 1.0E-6*integral (dFnz_5_0(_,_),0.0,2.0*pi,-pi/2.0,pi/2.0, prec) ; Eq 4-25

sera remplacée par l'équation (Eq 4.26) :

Fnz_5_0 == 1.0E-6* integral_dFnz_5_0 (0.0,2.0*pi,-pi/2.0,pi/2.0, prec) ; Eq 4-26

o Instruction « Foreign »

Une dernière possibilité offerte par le langage VHDL-AMS est de faire appel à un langage externe interprété pour la description et l'exécution de l'algorithme numérique générique. Cette méthode peut fonctionner en s'appuyant sur le mécanisme d'appel de fonction FOREIGN comme nous l'avons expliqué dans le chapitre 3. Dans ce cas, pour que le traitement des appels de fonctions soit dynamique, on peut envisager de remplacer le pointeur

de la fonction à intégrer (dFnz_5_0 dans l'exemple) par une chaîne de caractère comportant son expression littérale. La fonction intégrale externe (codé en C ou en java) doit analyser cette expression et la mettre dans le bon format avant de la traiter.

(3) Implémentation de la solution retenue

En résumé, pour modéliser les MEMS en VHDL-AMS, nous pouvons appliquer une des deux dernières méthodes présentées : (1) soit passer par une fonction externe *FOREIGN* traitant le calcul d'intégrale, (2) soit le coder complètement ce calcul en VHDL-AMS et dans ce cas il sera dupliqué autant de fois qu'on a des fonctions à intégrer.

Dans nos travaux, nous avons implémenté ces deux méthodes. Nous illustrons sur la Figure 4-35 la modélisation complète du dispositif de la lévitation en langage VHDL-AMS.

```
entity levitation is
Generic (  -- dimensions
           -- referential positions ( tx, ty, tz));
Port (  QUANTITY Hz : out real := 10.0;
        QUANTITY Fz: out Real: = 10.0;
        QUANTITY weight: out Real: = 10.0 );
end levitation

architecture BehavFromMacMMems of levitation is
...
-- approximation for polynomial or sum function
constant tetaVector : real_vector (0 to N);
constant phiVector  : real_vector (0 to N);

-- getFielEmitedModel (from Magnets)
function HEmitx_0 (x, y, z ) return return real is
begin
...
end function HEmitx0;
...
--getFieldReceivedModel (on ball model)
function Hx_ball_0 (x, y, z ) return return real is
begin
...
end function Hx_ball_0;
...
--getOutputModel dF
function dFnz_5_0 (tetas: real ;phis: real )return real is begin
...
end function dFnz_5_0 ;

--numerial integral  algorithm
function integral_dFnz_5_0 (x, y, z ) return real is
variable compute : real :=0.0;
begin
        genX: for i in 0 to N generate
          genY: for  j in 0 to N generate
                    compute  := compute + dFnz_5_0(tetaVector(i) , phiVector(j));
                   end generate;
        end generate;
        return (compute )
end function integral_dFnz_5_0 ;

begin
        Hz==Hz_4_0(0.0,0.0,0.0);
        ……..
        Fnz_5_0_sph== coeff* integral_dFnz_5_0 (x, y, z ) )*1.0e-6* (mu0/2.0);
        Fz== Fnz_5_0_sph;

end architecture BehavFromMacMMems ;
```

Figure 4-35 : Modélisation de la lévitation diamagnétique en VHDL-AMS

À l'issue de cette étude, nous avons montré qu'il est possible d'écrire un modèle VHDL-AMS pour un dispositif de MEMS magnétique. Nous proposons alors d'étendre l'outil MacMMems afin qu'il puisse créer automatiquement ces modèles VHDL-AMS.

D.3. Etude exploratoire de la projection de MacMMems

Nous avons réalisé uniquement une étude exploratoire de la projection automatique des modèles de MEMS magnétiques vers le langage VHDL-AMS en s'appuyant sur l'outil MacMMems. Le projeteur en lui-même n'ayant pas été réalisé par manque de temps, mais s'apparente à ce que nous avons réalisé pour RelucTool.

Les modèles générés par MacMMems sont essentiellement des modèles semi-analytiques de type « boîte grise » intégrant des parties analytiques et numériques. Pour effectuer la projection de l'outil MacMMems, nous allons nous appuyer aussi sur l'approche MDA de transformation de modèle. Le modèle source, dans ce cas de figure, est définit par deux fichiers qui seront conjointement utilisés :

- la description de la géométrie du modèle et ses paramètres sont générés sous forme d'un fichier XML ;
- l'ensemble des équations générées (modèle analytique) est décrit dans un langage spécifique SML (System Modeling Language), développé au G2Elab et compatible avec le framework CADES.

La démarche que nous proposons, mais que nous n'avons pas implémentée, est la suivante:

- définir un méta-modèle neutre XML rassemblant les deux fichiers précédents (XML pour la structure et SML du modèle comportemental) ;
- définir une équivalence entre ce XML et VHDL-AMS en se basant sur l'étude que nous venons de mener (problématique d'appel dynamique de fonction et d'algorithme d'intégration spatiale) ;
- réaliser le projeteur automatique basé sur des règles de transformation ainsi définie.

D.4. Conclusion sur la projection de MacMMems en VHDL-AMS

En étudiant la projection de la modélisation aux microsystèmes magnétiques vers le langage VHDL-AMS, nous avons montré au début qu'une description séparée des éléments magnétiques basiques et une description de leurs connexions posent des limitations en termes de modélisation VHDL-AMS. Nous avons donc choisi une description comportementale avec des modèles analytiques du dispositif complet. Ainsi, un exemple illustratif de lévitation diamagnétique nous a permis de détecter les différents problèmes d'écriture de ces modèles en VHDL-AMS. Nous avons proposé et mis en œuvre des solutions. Afin de faciliter la mise en œuvre des modèles analytique de ces dispositifs, nous nous sommes appuyés sur le générateur métier MacMMems et nous avons montré que la réalisation d'un projeteur de ses modèles vers le langage VHDL-AMS est possible mais difficile.

Conclusion

Nous avons étudié et réalisé dans ce chapitre la traduction ou la projection des différents modèles utilisés pour les systèmes électromagnétiques (réseau de réluctance et microsystèmes magnétiques) en langage VHDL-AMS, afin d'offrir plus de portabilité e d'interopérabilité des modèles et des outils. Différents solutions ont été proposées dans ce contexte.

A l'issu des chapitres 3 et 4, nous pouvons comparer les différentes méthodes d'interopérabilité par projection et les couplages par pilotage de modèles « exécutables » via les composants logiciels. Nous illustrons les résultats obtenus sur la Figure 4-36. Nous constatons plusieurs choses :

- La modélisation éléments finis des dispositifs magnétiques n'est pas modélisable en VHDL-AMS en raison de la gestion des PDE. Seules les solutions de co-simulation et de couplage grâce aux composants logiciel ICAr sont exploitables.
- La modélisation de microsystèmes magnétiques par approche de composition est difficilement réalisable voir impossible en raison des interactions fortes et multidirectionnelles mises en jeu.
- La modélisation de microsystèmes magnétiques par approche semi-analytique est possible avec VHDL-AMS, mais reste délicate à gérer.
- La modélisation des systèmes électromagnétiques par approche des réseaux de réluctances est possible en VHDL-AMS. La modélisation de l'architecture avec un calcul précis de la force reste délicate.
- Des modèles de systèmes électromécaniques obtenus par plan d'expériences sous forme d'expressions polynomiales, en s'appuyant sur Fgot, peuvent facilement être traduits en VHDL-AMS.

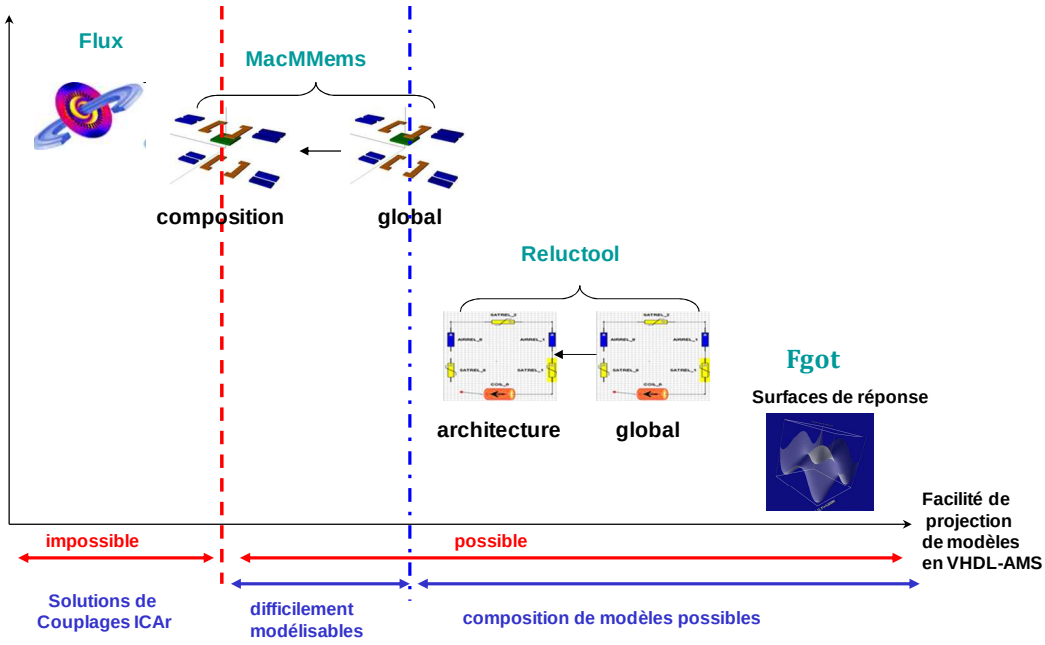


Figure 4-36 : Comparaison des approches développées dans notre thèse

CHAPITRE 5

Application du Prototypage Virtuel Fonctionnel sur un actionneur électromagnétique

Introduction

Dans ce dernier chapitre, nous allons exploiter nos différents travaux, résumés sur la Figure 5-1, au travers d'exemples concrets de conception et modélisation de macro et microsystèmes magnétiques en VHDL-AMS. Nous appliquerons cela aux différents niveaux de conception du Prototypage Virtuel Fonctionnel (PVF) sur un système d'actionneur magnétique développé par Schneider Electric. Cela nous permet d'illustrer nos contributions à l'interopérabilité entre modèles et outils dans un cycle de conception, offrant ainsi la portabilité et le portage des modèles entre concepteurs et entre logiciels de calcul (simulation, dimensionnement, etc.).

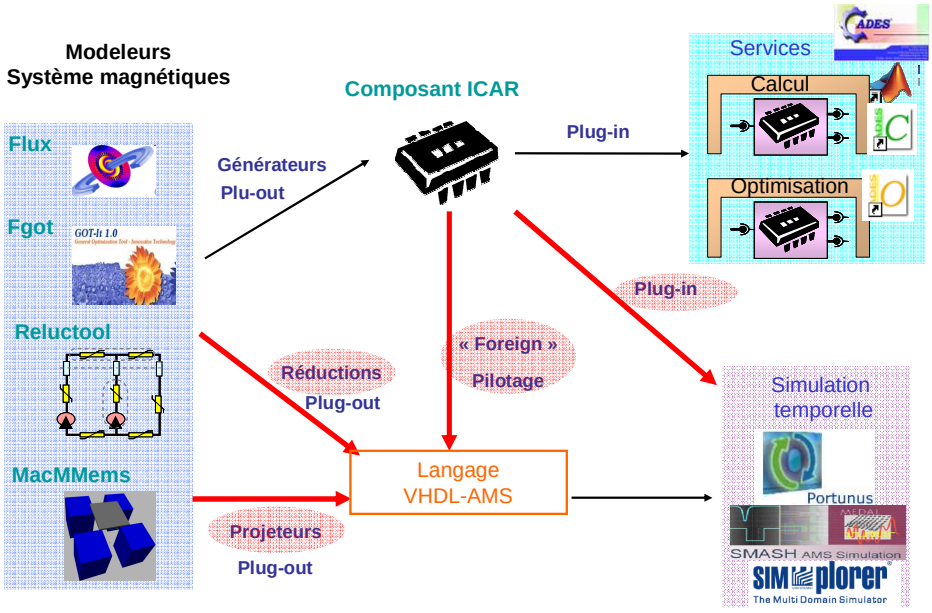


Figure 5-1: Schéma résumant nos contributions à cette thèse

Ce dernier chapitre sera aussi l'occasion d'illustrer la capacité du langage VHDL-AMS à traiter de la pluridisciplinarité comme nous l'avons fait avec l'exemple du déclencheur (chapitre 1 et annexe 2). Nous aborderons la réduction de modèles par des tables de Flux et par des plans d'expériences. Nous montrerons comment VHDL-AMS pourra supporter entièrement le cycle PVF de conception avec différents niveaux de modélisation coexistant dans une simulation.

Lorsque nous aborderons la question de validation de nos modèles créés et de leur réutilisabilité dans d'autres contextes, nous discuterons des caractéristiques, des hypothèses et des limites d'utilisations associées aux modèles.

A. Flot de conception mis en œuvre

La simulation système exige des modèles simples et rapides pour étudier les interactions entre les composants dans la modélisation hiérarchique du cycle de conception. Dans un flot de conception, l'approche de base commence à partir des spécifications pour atteindre un objectif de conception. Ensuite, un découpage en sous-systèmes est mis en place ainsi qu'un raffinement de modèles de composants. Enfin, l'étape de l'intégration dans le système globale permet de valider son fonctionnement.

Ce flot de conception est un processus récursif, c'est-à-dire qu'un sous-système ou un composant peut être étudié avec son propre cycle en V et peut être intégré comme un composant d'un autre système.

Un tel flot de conception s'applique aussi pour chaque objectif de conception : analyse, pré-dimensionnement et dimensionnement (chapitre 1). Ainsi, d'une manière similaire à la simulation dynamique, il est possible de décrire le PVF dans le contexte de dimensionnement par optimisation, comme le montre la Figure 5-2.

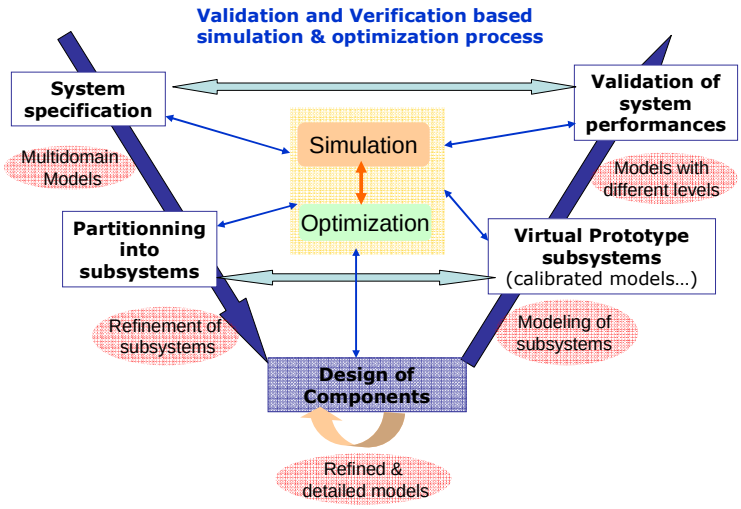


Figure 5-2 : Flot de conception renforcé par de la simulation et de l'optimisation

Nous allons présenter cette démarche de conception sur un actionneur électromagnétique, un contacteur en « E », qui a été étudié en collaboration avec Schneider Electric. En particulier, cette collaboration a permis d'élaborer différents niveaux d'abstraction et de

finesses associés à ces systèmes. Certains résultats ont été obtenus durant le stage de fin d'étude de Gregory HUMMLER [HUM-12] auquel j'ai participé activement.

B. Description du dispositif d'un contacteur électromagnétique

B.1. Description de l'actionneur en E

Un contacteur est un appareil électrotechnique commandé par un courant électrique, qui a pour fonction d'établir ou d'interrompre le passage du courant. Un tel appareil est capable de fonctionner à des courants importants. On les retrouve en entrée d'alimentations de moteurs industriels de petites ou de grandes puissances (du fait de leur pouvoir de coupure important), en milieu domestique (chauffage, chauffe eau) ou encore dans les véhicules (éclairage). L'organe moteur du contacteur est l'électro-aimant. Nous allons étudier principalement un actionneur en « E », présenté sur la Figure 5-3 et la Figure 5-4.

Figure 5-3 Actionneur en E étudié (bobine + circuit magnétique)

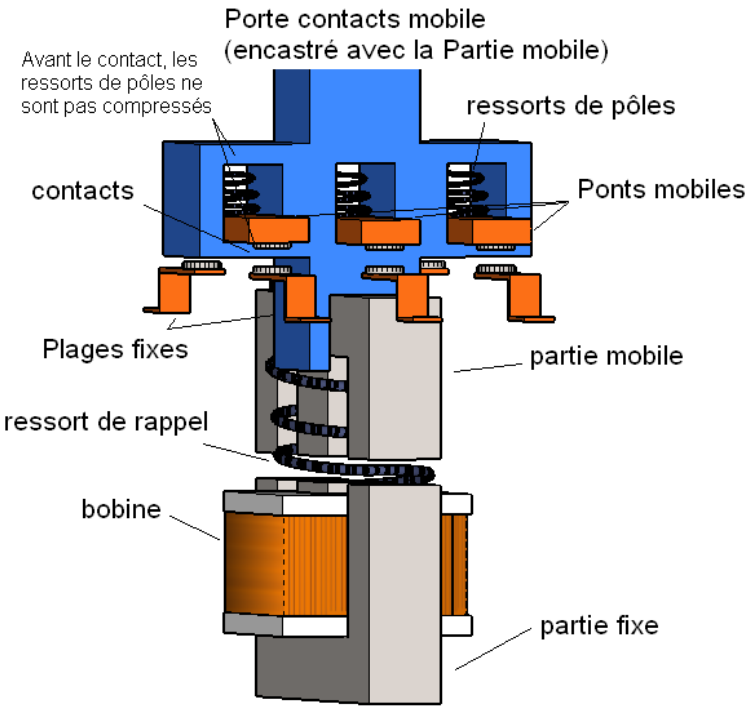


Figure 5-4 : Modèle CAO de l'actionneur, réalisé avec google sketchup [HUM-12]

B.2. Fonctionnement de l'actionneur en E

Dans la phase non opérationnelle, le contacteur est en position de repos ; la distance est maximale entre les deux circuits magnétiques fixe et mobile ainsi que celle des écartements des ressorts de pôles. Dans la phase opérationnelle, dès que le contacteur est mis sous tension, le circuit magnétique mobile se déplace vers le circuit magnétique fixe, entraînant avec eux les ressorts de pôle. Le déplacement final du circuit magnétique mobile comprime les ressorts des ponts mobiles afin d'obtenir une forte pression de ceux-ci sur les pôles fixes (Figure 5-5).

Dès la mise sous tension, la bobine crée un flux qui circule dans le circuit magnétique et génère ainsi une force électromagnétique permettant le déplacement du noyau mobile.

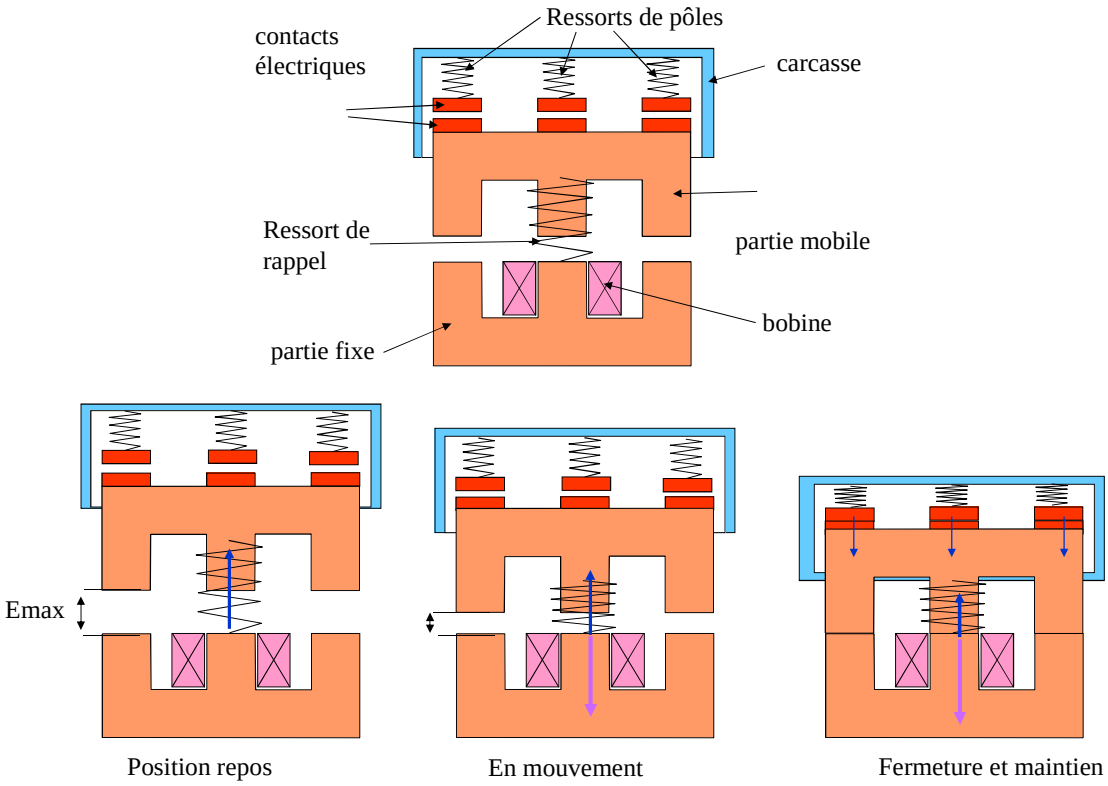


Figure 5-5 : Schéma simplifié du fonctionnement de l'actionneur

La Figure 5-6 montre l'évolution pendant la course de l'effort magnétique généré par la bobine et l'effort résistant généré par les différents ressorts du contacteur. Lors du déplacement, on remarque qu'au début seul le ressort de rappel génère une force opposée au mouvement. Puis, lors de l'impact des pôles (contact entre les ponts mobiles et les plages fixes), les ressorts de pôles vont à leur tour se compresser et on aura alors deux forces opposées aux mouvements : la force du ressort de rappel et celle des ressorts de pôles. On observe également qu'à l'impact des pôles, l'effort résistant devient supérieur à l'effort magnétique (entre les points B et C sur la Figure 5-6).

Afin de déterminer si l'actionneur peut monter complètement malgré ce changement de signe au niveau du bilan des forces, il suffit de réaliser une simulation dynamique. Une autre solution consiste à regarder si l'énergie emmagasinée par la partie mobile, pendant la première partie du mouvement, est suffisante pour combler ce changement de signe.

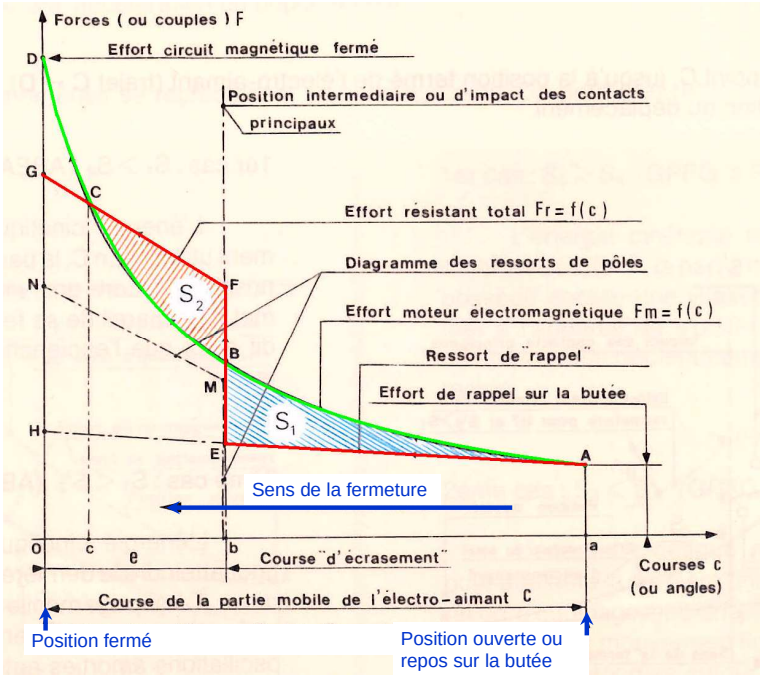


Figure 5-6 : Évolution des efforts magnétique et résistant en fonction de la course [SIF-88]

En comparant les deux aires S1 et S2 (sur la Figure 5-6), on obtient l'allure de l'évolution temporelle du contacteur (Figure 5-7) :

- si S1 < S2, l'actionneur n'a pas accumulé suffisamment d'énergie et va rester ouvert
- si S1 > S2, l'actionneur poursuit sa course et se ferme.

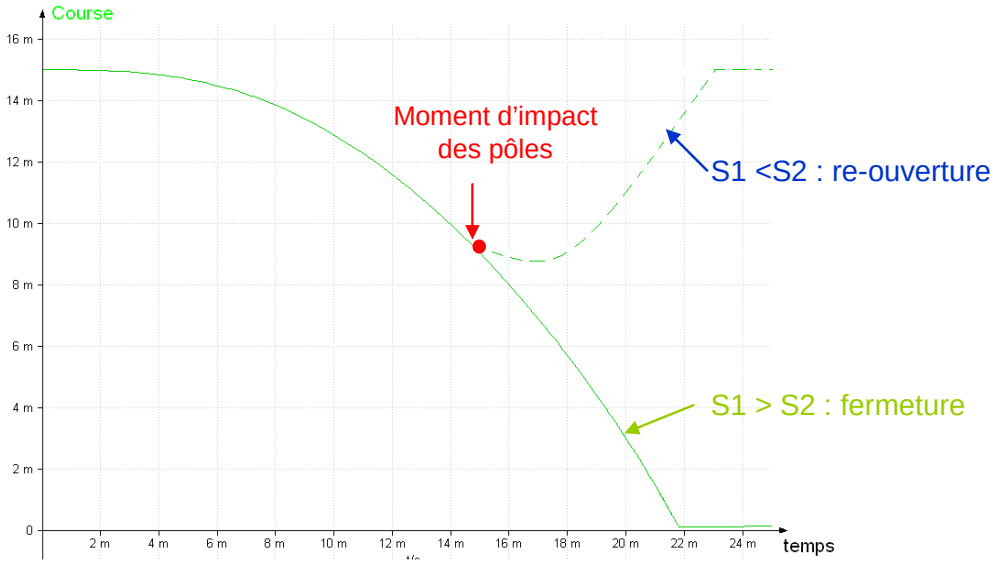


Figure 5-7 : Évolution temporelle de la course du circuit mobile

C. Conception de l'actionneur électromagnétique

Nous allons maintenant nous intéresser aux différentes étapes du cycle de conception de notre actionneur. La Figure 5-8 illustre le parcours du cycle en positionnant les différents paragraphes (de C.1 à C.4) correspondant aux étapes que nous allons étudier.

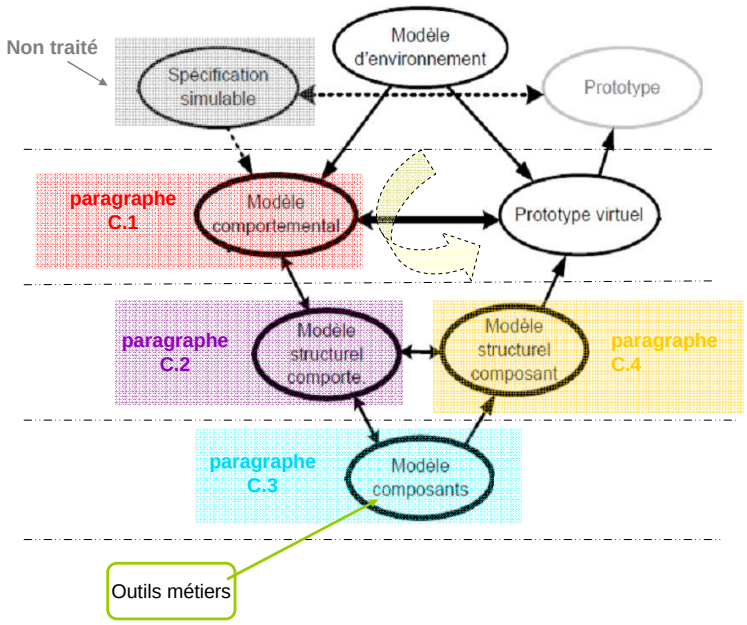


Figure 5-8 : Répartition des travaux suivants

Dans le PVF, une spécification est un ensemble explicite d'exigences à satisfaire par un dispositif à concevoir. Le modèle des spécifications simulables représente une description des besoins et des contraintes que doit respecter le prototype virtuel de l'actionneur. Ces besoins peuvent concerner le temps de montée, la tenue au choc ; les contraintes peuvent porter sur les matériaux utilisés, la course de l'actionneur, etc. Nous n'allons pas traiter ici du modèle des spécifications, nous nous placerons simplement dans des cas de simulation qui requièrent le calcul de la position du mobile lorsque le système est soumis à une commande d'actionnement électrique.

Dans le paragraphe C.1, nous allons donc commencer par élaborer le modèle comportemental issue d'une analyse fonctionnelle du système. Dans le paragraphe C.2, nous traiterons la décomposition du système en différents sous-modèles, en illustrant les différents éléments magnétiques, électriques et mécaniques mise en jeu. Nous montrerons que les modèles mécanique et électrique sont faciles à identifier.

Dans le paragraphe C.3, nous nous intéresserons aux modèles de composants associés aux différents sous-systèmes Nous détaillerons essentiellement différents niveaux de modèles et de précision associés aux modèles magnétiques en nous appuyant sur les différents outils que nous avons traités dans notre thèse.

Dans le dernier paragraphe C.4, nous allons nous positionner par rapport à la problématique d'intégration des différents composants développés.

C.1. Modèle comportemental

Dans la démarche de conception, nous commençons au plus haut niveau d'abstraction avec une analyse fonctionnelle du système depuis les besoins des clients et les contraintes associées. Cette analyse conduit à un modèle formalisé à travers des diagrammes de blocs ou des machines d'état, en vue d'atteindre un modèle simulable de la description du fonctionnement du système. Un exemple illustratif d'un modèle comportemental et fonctionnel de notre actionneur est présenté sur la Figure 5-9.

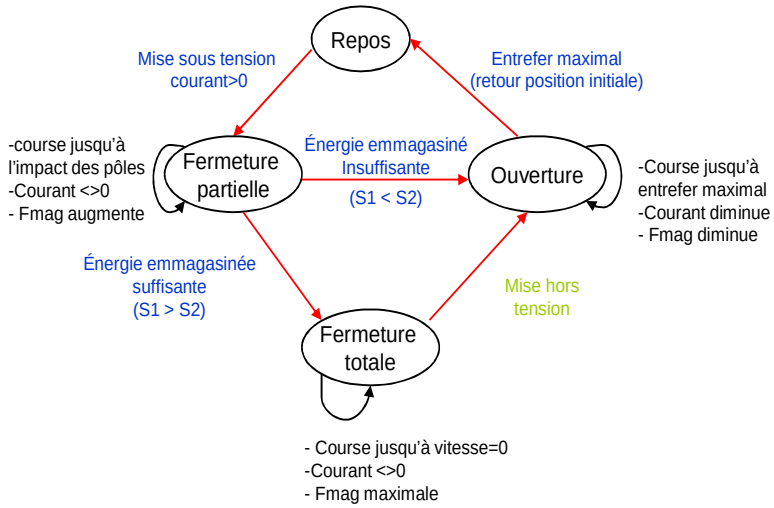


Figure 5-9 : Modèle comportemental fonctionnel de l'actionneur sous forme d'une machine d'état

Pour la conception de l'actionneur en E, il est parfois essentiel de trouver des modèles simples pour une première étude de haut niveau. Il est aussi nécessaire de pouvoir décomposer le système successivement jusqu'à ses composants de base, pour s'assurer de son fonctionnement. C'est ce que nous allons étudier maintenant.

C.2. Modèle structurel comportemental

Cet actionneur intègre différentes parties correspondant à différents domaines physiques, le tout étant très couplé :

- la partie électrique : la bobine et son alimentation ;
- la partie électromagnétique : le circuit magnétique coupé en deux parties (fixe et mobile) séparées par un entrefer variable ;
- la partie mécanique : ressort de rappel, ressorts de pôles, noyaux de fers (masse), butées.

Il est difficile de définir l'actionneur sans décrire ses différentes parties ainsi que les interactions entre elles. La Figure 5-10 présente la décomposition de notre actionneur et la connexion entre ses différents sous-modèles. Les modèles des différents sous-systèmes seront développés en utilisant VHDL-AMS dans les paragraphes suivants.

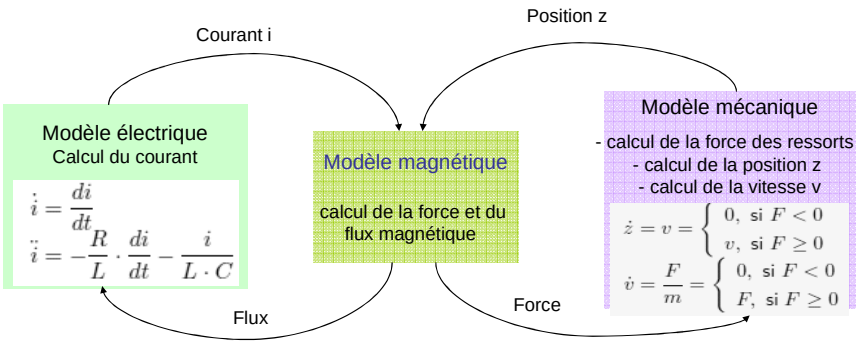


Figure 5-10 : Décomposition de l'actionneur en sous systèmes distincts

C.3. Modèles des composants

C.3.1 Modélisation des composants mécaniques

Dans le domaine mécanique, le mouvement de la partie mobile (noyau) est modélisé par :

- une masse mobile avec un unique noeud d'accès (la position étant la quantité *across* et la force la quantité *through*) ;
- un ressort précontraint représentant les différents effets résistifs de l'actionneur ;
- deux buttées limitant le déplacement minimum et maximum du noyau mobile.

- **Masse mobile**

Ce composant traduit les équations de la dynamique du mobile, à savoir $\text{Masse}.\ddot{x} = \sum \text{Forces}$.

où - la masse et les forces sont à définir et à paramétrer par l'utilisateur,
- *x* est la position du mobile
- $\ddot{x}$ est sa dérivée seconde.

- **Ressort**

Ce composant sert à modéliser l'ensemble des efforts résistifs de l'actionneur. On y retrouve le ressort de rappel et les ressorts de pôles (Figure 5-4). Les équations dépendent de l'impact des pôles (appelé écrasement) :

$$\begin{cases} F = F_{rappel} + F_{pôles} \text{ si } (\text{position} < \text{écrasement}) \\ F = F_{rappel} \text{ sinon} \end{cases}$$

- **La butée**

Ce composant sert à définir les limites de la position dans le système. La modélisation d'une butée est difficile. Plusieurs modèles existent selon les comportements souhaités. Une butée peut être idéalisée avec un modèle simple de vitesse nulle, ou représente un amortissement et une constante de raideur. Ce deuxième modèle est défini par les équations suivantes

$F = k \cdot x + damp \cdot \dot{x}$ si $\dot{x}$ ne change pas de signe

$F = 0$ sinon

Où

- x est la position du noyau mobile,
- $\dot{x}$ sa dérivée représentant la vitesse
- *K* la constante de raideur du ressort
- *damp* le coefficient d'amortissement à définir par l'utilisateur.

C.3.2 Modélisation des composants électriques

Un modèle simple d'alimentation en courant a été utilisé pour alimenter la bobine ; un élément de conversion magnétique/électrique (chapitre 4). Nous considérons ici que la bobine est un composant « purement électrique » et que son comportement est régi par l'équation suivante :

$$V = L.\frac{di}{dt} + R.i$$

où

- i est le courant qui la traverse,
- R est la résistance interne de la bobine
- L son inductance.

On définit aussi une architecture de modèle de l'actionneur où la résistance n'est pas constante et dépend de la température *T*. On a alors l'équation suivante :

$$V = L.\frac{di}{dt} + R(T).i$$

C.3.3 Modélisation des composants magnétiques

Les deux systèmes précédents garderont ce niveau de modélisation dans toute notre étude. Par contre, les systèmes magnétiques sont plus complexes. Nous allons étudier plus en détails leurs modélisations mises en jeu. Nous allons ainsi pouvoir exploiter les différentes méthodes de modélisations que nous avons présentées dans les chapitres précédents, en nous positionnant vis-à-vis des niveaux de conception dans lesquels nous nous situons durant le processus de conception. Notre objectif est d'appliquer ces différents aspects à la modélisation d'un dispositif électromagnétique, afin d'améliorer la précision du modèle global. De cette façon, différents niveaux de précisions des modèles magnétiques sont possibles et nous en avons développés plusieurs pour différentes analyses (Figure 5-11) :

- un modèle précis par éléments finis ;
- un modèle analytique issu d'un réseau de réluctance ;
- un modèle réduit avec des tables de flux ;
- un modèle réduit issu des plans d'expériences.

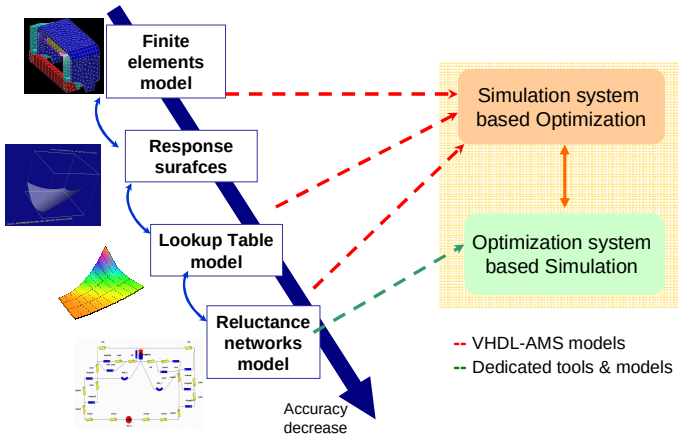


Figure 5-11 : Cycle de conception, renforcé par la simulation et l'optimisation des modèles multi-précisions

Le modèle magnétique choisi dépend des études visées initialement. Disposer d'un prototype virtuel avec différents niveaux d'abstraction permet de tester un ensemble varié de scénarii très tôt dans le cycle de conception. Nous allons voir comment utiliser ces différents modèles pour une simulation système en nous appuyant sur le langage VHDL-AMS.

C.3.3.1 Modèle par éléments finis

- **Description du modèle**

La méthode des éléments finis a été choisie pour l'analyse statique du contacteur. Sa description dans l'outil Flux3D™ est présentée sur la Figure 5-12. Ce modèle nous a été fourni par Schneider Electric.

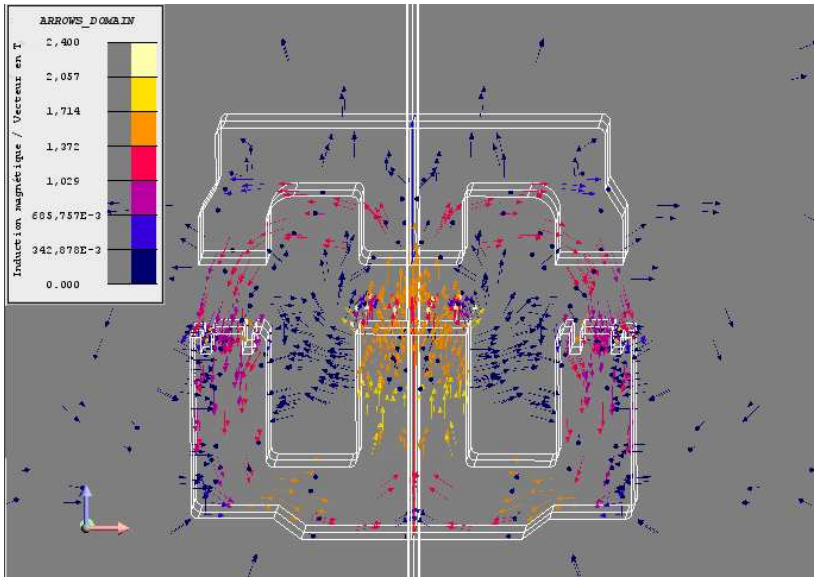


Figure 5-12 : Chemin de flux de l'actionneur en E dans l'outil Flux3D™

- **Modélisation VHDL-AMS**

Comme nous l'avons détaillé dans les chapitres précédents, un modèle éléments finis ne peut pas être formalisé directement en langage VHDL-AMS. Nous avons donc appliqué l'approche de pilotage du logiciel Flux3D, que nous avons développée dans le chapitre 3, afin de pouvoir simuler ce modèle à travers un modèle VHDL-AMS dans SMASH, ou à travers un

modèle C dans Portunus. Dans ce cas, la modélisation Flux du contacteur est pilotée à travers un composant logiciel ICAr.

C.3.3.2 Modèle analytique : réseau de réluctances

- **Description du modèle**

Afin d'élaborer le modèle réluctant du contacteur, nous commençons par une étude de la propagation des flux en utilisant le modèle éléments finis. Ainsi, nous avons étudié les chemins parcourus par le flux magnétique afin de définir le modèle réluctant correspondant, comme le montre la Figure 5-12. Nous avons aussi étudié les caractéristiques des matériaux (H(B)) et la forme géométrique des différents éléments du système pour pouvoir intégrer les principales variables du système dans notre modèle réluctant. Nous avons pris en compte les flux de fuite les plus importants pour obtenir un modèle relativement proche de la réalité.

La Figure 5-13 illustre ce modèle. Les réluctances en bleu représentent celles dans l'air et les réluctances en jaune celles dans le fer. Nous avons séparé la force électromotrice en deux pour pouvoir intercaler des fuites dans la partie située entre les «pattes» de la partie fixe. Dans notre étude, nous avons établi un modèle simplifié de réseau de réluctances associé au contacteur, obtenu après une simplification géométrique des différentes côtes du système.

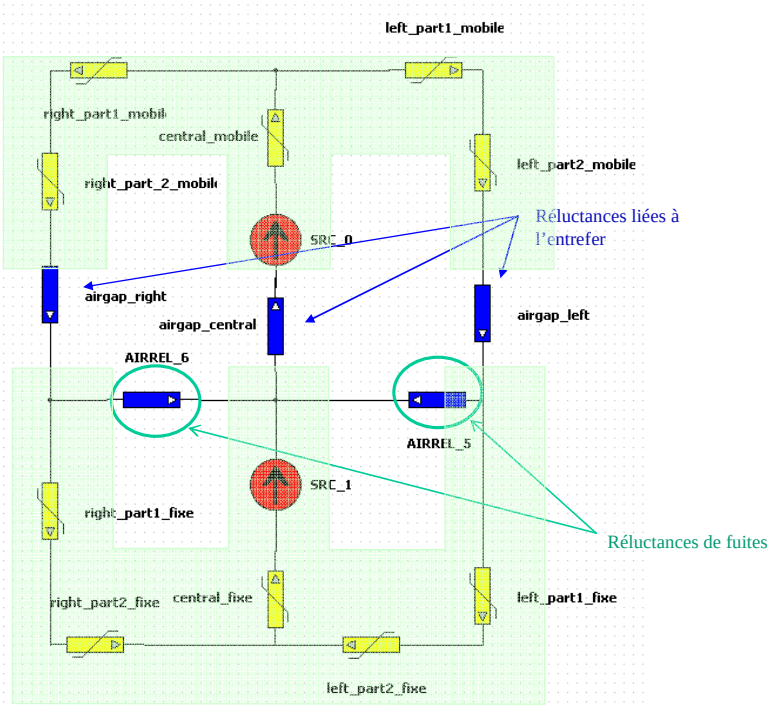


Figure 5-13 : réseau de réluctance associée au modèle de contacteur

Sur la Figure 5-14 nous comparons les résultats obtenus par RelucTool à ceux issus de Flux3D. Il s'agit du calcul de l'effort généré par l'actionneur pour une force électromotrice FEM constante.

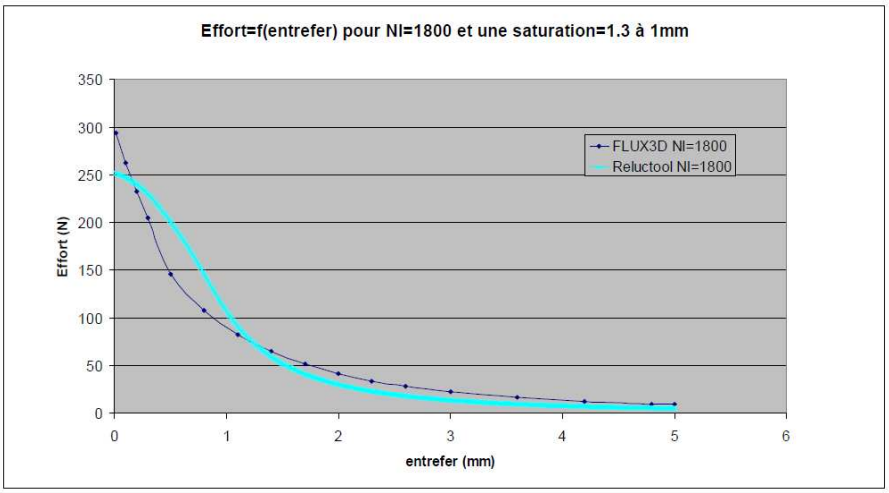


Figure 5-14 : Comparaison de l'effort magnétique calculé par Flux3D et RelucTool

Nous remarquons qu'à faible entrefer (<1mm) notre modèle Reluctool ne suit pas la courbe de flux3D. On peut l'expliquer car la saturation du fer est supérieure à 1.3T (plus l'entrefer est petit plus la saturation est grande). De plus, notre modèle réluctant ne peut pas tenir compte de saturations locales contrairement à flux3D qui discrétise chaque partie du fer. Enfin, la saturation entraîne de nombreux flux de fuites, beaucoup plus diffus, qui sont difficilement modélisables dans le réseau de réluctances. En revanche, pour des entrefers plus grand (>1.5mm), c'est-à-dire pour une saturation moindre, les courbes ont la même allure.

Pour pallier ce manque, il faudrait définir un modèle plus précis en prenant en compte la modélisation de l'épanouissement du flux au niveau des entrefers. Ceci a été fait dans les travaux de [GUI-11]. Dans notre étude, nous allons nous contenter de ce modèle simplifié.

- **Modélisation VHDL-AMS**

Nous avons réalisé différents modèles VHDL-AMS associés au schéma réluctant de l'actionneur en E selon les approches proposées dans le chapitre 4. Ainsi, depuis l'outil RelucTool, nous avons généré automatiquement :

- le modèle structurel (composition d'éléments de la bibliothèque) ;
- le modèle global (calcul de la force par dérivation symbolique automatique).

C.3.3.3 Modèle réduit avec les tables de flux

- **Description du modèle**

Le modèle de l'actionneur est modélisé par une lecture de tables numériques issues de Flux3D. Le principe de la modélisation par lecture de table est le suivant :

Dans un premier temps, on réalise une modélisation de notre actionneur sous Flux3D. Ensuite, on réalise un ensemble de simulations permettant de calculer l'effort magnétique et le flux total (circulant dans le circuit magnétique), en fonction du courant et de l'entrefer (position). Les résultats sont ensuite récupérés sous la forme de tables. Dans ce cas, nous obtenons deux matrices (table de la force magnétique et table du flux) en fonction du courant et de la taille de l'entrefer, comme le montre la Figure 5-15.

Chaque matrice est chargée dans un composant spécifique de Portunus (CHR 3D). Cela permet à chaque pas de temps d'identifier, par interpolation, les sorties (force et flux) à partir du courant et de l'entrefer.

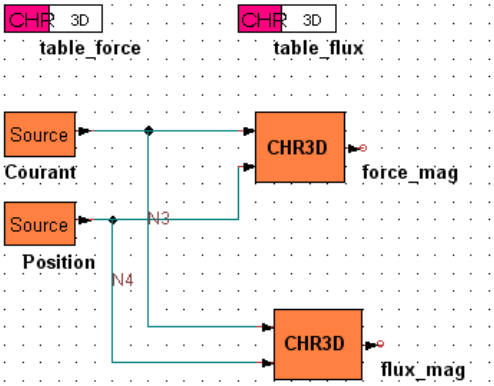


Figure 5-15 : Schéma de la lecture de tables sous PORTUNUS

- **Modélisation VHDL-AMS**

La solution disponible dans Portunus pour lire les tables ne nous satisfaisait pas car nous souhaitions simuler le dispositif dans tout outil compatible VHDL-AMS. Le problème rencontré est que VHDL-AMS ne dispose pas directement des éléments nécessaires aux traitements standardisés des tables et des fonctions d'interpolation. Il existe cependant un package standard appelé « TextIO » permettant la lecture et l'écriture des fichiers. Ce dernier n'est pas encore implémenté dans Portunus. Nous avons donc étudié et mis en œuvre cette fonctionnalité dans SMASH et Simplorer.

Pour gérer la lecture des tables dans nos modèles réduits, nous avons mis en œuvre ce traitement en deux étapes (Figure 5-16) :

- la prise en charge des fichiers Excel contenant les tables par le modèle VHDL-AMS en nous appuyant sur le package « TextIO »
- l'exploitation du modèle dans la simulation, pour laquelle il est nécessaire d'avoir un modèle gérant les interpolations.

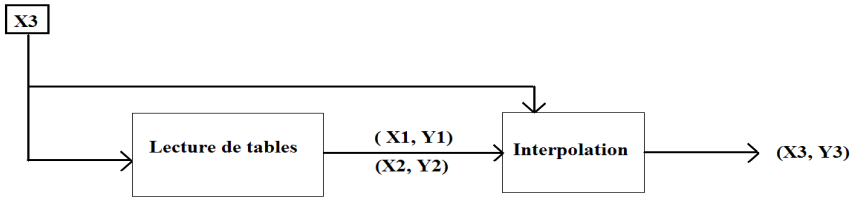


Figure 5-16 : Schéma bloc de principe pour la lecture et l'exploitation des tables

(1) Lecture des tables en VHDL-AMS

Nous avons réalisé des fonctions de base permettant de lire un fichier texte contenant les résultats de simulation Flux3D sous forme matricielle. Ce fichier est identifié par une chaîne de caractères contenant son nom. La sortie de cette fonction est l'ensemble des vecteurs qu'elle collecte depuis les différentes colonnes de la matrice.

(2) Interpolation en VHDL-AMS

Reprenons l'exemple de la Figure 5-16 afin d'illustrer le fonctionnement d'une interpolation linéaire et comment la coder en langage VHDL-AMS. Nous disposons d'une variable X3 en entrée (courant ou entrefer par exemple). Le bloc « Lecture de table » va alors chercher les deux valeurs les plus proches X1 et X2. Ensuite, il récupère également les sorties correspondante Y1, Y2. Les deux couples (X1, Y1) et (X2, Y2) ainsi que la valeur de X3 sont alors transmis au bloc « interpolation » qui va identifier un des trois cas suivants :

- X3 = X1, alors Y3 = Y1 (lecture classique de table)
- X3 = X2, alors Y3 = Y2
- X3 ≠ X1 et X3 ≠ X2, alors le bloc réalise une interpolation linéaire et on aura :

$$Y3 = Y1 + \frac{Y2 - Y1}{X2 - X1} \cdot (X3 - X1)$$

La Figure 5-17 présente le code VHDL-AMS associé à une interpolation linéaire.

```
function interpolate (x3,y2,y1,x2,x1 : in real)
      return real is
      variable m, yvalue : real;
   begin
      assert (x1 /= x2)
          report "interpolate: x1 cannot be equal to x2"
          severity error;
      assert (x3 >= x1) and (x3 <= x2)
          report "interpolate: x must be between x1 and x2, inclusively "
          severity error;

      m := (y2 - y1)/(x2 - x1);
      yvalue := y1 + m*(x3 - x1);
      return yvalue;
  end function interpolate;
```

Figure 5-17 : Fonction d'interpolation linéaire en VHDL-AMS

C.3.3.4 Modèle réduit avec des plans d'expériences

- **Description du modèle**

Nous proposons ici de fournir un modèle de l'actionneur selon la méthode des plans d'expériences. Cette méthode propose, en premier lieu, de déterminer les paramètres les plus influents dans notre dispositif étudié (dans notre cas l'entrefer (position) et le courant injecté). Ensuite, la méthode propose d'élaborer une « surface de réponse » donnant une approximation sous la forme d'un modèle de régression à partir d'un nombre minimum de calculs du modèle fin. C'est intéressant par rapport aux tables précédentes d'effort et de flux, car cela nécessite normalement beaucoup moins de simulations.

L'application de cette méthode nous permet ainsi d'identifier et de générer un modèle réduit de notre actionneur en régime statique. Nous avons utilisé l'outil GOT[76] : un logiciel d'optimisation développé au G2Elab. La Figure 5-18 illustre le modèle réduit par « plan d'expériences » généré par l'outil Fgot sur notre actionneur.

[76] Fgot : outil http ://www.cedrat.com/en/software/got-it.html

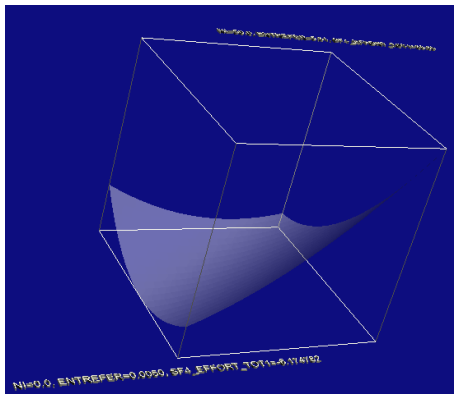


Figure 5-18 : Modèle « plan d'expériences » pour flux magnétique généré par GOT.

- **Modélisation VHDL-AMS**

Dans cette approche, nous choisissons de modéliser le composant magnétique par un bloc VHDL-AMS qui gère la surface de réponse générée pour notre actionneur. L'outil GOT génère deux fonctions paramétriques correspondantes au calcul de la force magnétique et au calcul du flux magnétique, en fonction des paramètres position et courant. La Figure 5-19 illustre un extrait du fichier obtenu par GOT. Il décrit brièvement la forme de la surface de réponse obtenue ainsi que la fonction estimée.

Construction du substitut de EFFORT_TOT

Courte description de SF1

GridRbfApproximationFactory (
levelCounts: [
5]
minRelativeMagnitude: 0.05
name: SF_Force
original: EFFORT_TOT
rbfMode: RbfP2
rbfType: Multiquadric
)

Résultat de SF1 comme texte

SF_Force_EFFORT_TOT1=
-5098.15+1523.35*Lin(NI,0,50)+(-789.7)*Lin(ENTREFER,0.0050,5.31)+562.8*Lin(ENTREFER,0.0050,5.31)**2+178.3*Lin(NI,0,50)**2+(-155.25)*Lin(ENTREFER,0.0050,5.31)*Lin(NI,0,50)+(-2.297309621132604E7)*MultiquadricRbf([ENTREFER, NI], [0.0050, 0], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+9.221105123528928E7*MultiquadricRbf([ENTREFER, NI], [0.0050, 12.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-1.6660502468246788E8)*MultiquadricRbf([ENTREFER, NI], [0.0050, 25], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+1.5975464135547996E8*MultiquadricRbf([ENTREFER, NI], [0.0050, 37.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-6.345200697238981E7)*MultiquadricRbf([ENTREFER, NI], [0.0050, 50], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+9.266088815842 36E7*MultiquadricRbf([ENTREFER, NI], [1.33, 0], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-3.7287176428422356E8)*MultiquadricRbf([ENTREFER, NI], [1.33, 12.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+6.646946367607831E8*MultiquadricRbf([ENTREFER, NI], [1.33, 25], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-6.234142711435231E8)*MultiquadricRbf([ENTREFER, NI], [1.33, 37.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+2.430206965599102E8*MultiquadricRbf([ENTREFER, NI], [1.33, 50], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-1.3344560412738778E8)*MultiquadricRbf([ENTREFER, NI], [2.66, 0], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+5.3934392454079 15E8*MultiquadricRbf([ENTREFER, NI], [2.66, 12.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-9.609739002884048E8)*MultiquadricRbf([ENTREFER, NI], [2.66, 25], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+8.9841117961781 4E8*MultiquadricRbf([ENTREFER, NI], [2.66, 37.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-3.4924408500910 96E8)*MultiquadricRbf([ENTREFER, NI], [2.66, 50], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+8.201863369082 65E7*MultiquadricRbf([ENTREFER, NI], [3.985, 0], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-3.344397009611 19E8)*MultiquadricRbf([ENTREFER, NI], [3.985, 12.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+6.029283980490266E8*MultiquadricRbf([ENTREFER, NI], [3.985, 25], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-5.710730261607547E8)*MultiquadricRbf([ENTREFER, NI], [3.985, 37.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+2.243584310136093E8*MultiquadricRbf([ENTREFER, NI], [3.985, 50], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-1.772167281219226 5E7)*MultiquadricRbf([ENTREFER, NI], [5.31, 0], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+7.348452968167247E7*MultiquadricRbf([ENTREFER, NI], [5.31, 12.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-1.3646313192593014E8)*MultiquadricRbf([ENTREFER, NI], [5.31, 25], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+1.338188506868313E8*MultiquadricRbf([ENTREFER, NI], [5.31, 37.5], [0.9378003685486586, 8.838834764831843], 15.877445891202147)+(-5.402853116105934E7)*MultiquadricRbf([ENTREFER, NI], [5.31, 50], [0.9378003685486586, 8.838834764831843], 15.877445891202147)

Figure 5-19 : Surface de réponse obtenue par GOT pour l'effort total.

Une fois que l'on obtient une surface illustrant bien le fonctionnement de notre actionneur, nous pouvons extraire les différentes fonctions obtenues sous forme d'un ensemble d'équations dans un fichier texte. Les surfaces de réponses étant des expressions analytiques, il est aisé de les décrire en VHDL-AMS et donc d'en automatiser la projection depuis l'outil Fgot. Ceci n'a pas été réalisé ici, mais sera à faire. Aussi, dans nos tests, nous avons fait cette transcription manuellement.

C.4. Modèle structurel de composants

Nous arrivons à la montée du cycle de conception avec la phase d'intégration. A l'issue des étapes précédentes, nous disposons d'un ensemble de modèles pour notre actionneur :

- un modèle du sous-système mécanique
- un modèle du sous-système électrique
- des modèles avec différents niveaux de précision du composant magnétique
    - (i) modèle éléments finis,
    - (ii) modèle réluctant,
    - (iii) modèle des tables,
    - (iv) modèle surface de réponse.

Réaliser un prototype virtuel de l'actionneur revient à assembler les sous-modèles magnétique, électrique et mécanique (Figure 5-20). En intégrant les différents modèles de composants magnétiques (on se retrouve avec quatre prototypes virtuels possibles du contacteur en E. Nous allons analyser le fonctionnement du contacteur, en nous intéressant à sa fermeture et à son ouverture. L'objectif final est de confronter ces différents modèles à des cas tests, permettant de réaliser le plan de vérification de l'actionneur.

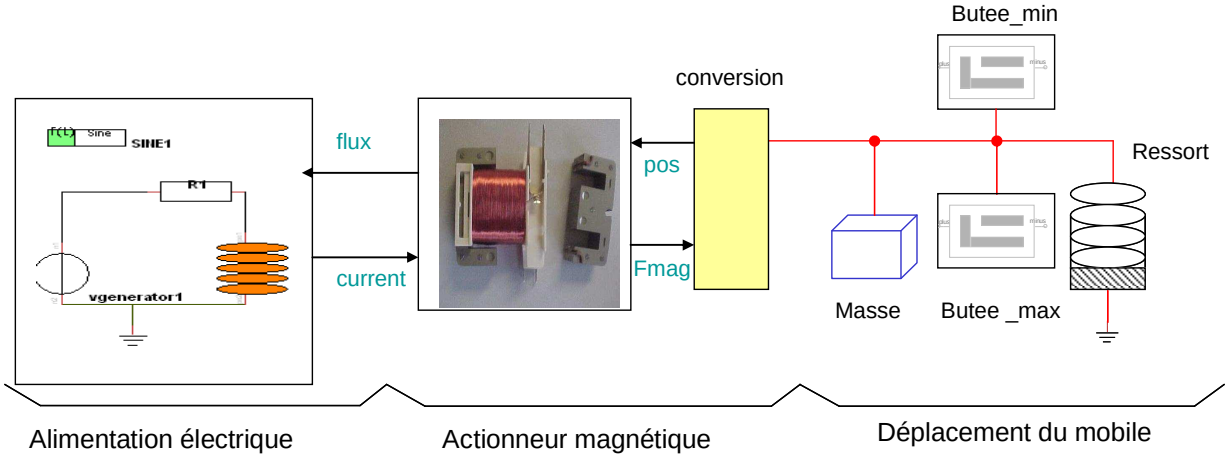


Figure 5-20 : Intégration des différents composants de l'actionneur

Chacun des modèles magnétiques réalisés est utilisé selon des objectifs spécifiques de l'étude. D'une manière générale, quand on aborde la phase d'intégration dans le but de valider le fonctionnement du système global, un concepteur système ou un intégrateur a besoin d'un modèle léger du sous-système magnétique. Dans ce cas, le pilotage d'un code éléments finis ne sera pas retenu. Par contre, si le modèle éléments finis et le modèle réduit sous forme de tables ou de surfaces de réponses sont disponibles, on privilégiera le plus simple. En outre, un modèle réluctant simple, obtenu rapidement et à moindre coût, peut également faire office de modèle pour l'intégration système.

Par exemple, dans le cas où l'objectif de conception est l'analyse de sensibilité ou le pré-dimensionnement, nous pouvons faire appel également aux modèles réluctants paramétrés. Nous rappelons que RelucTool a été conçu dans ce but. Ainsi, le modèle magnétique du contacteur issu de cet outil après son dimensionnement par optimisation, sera projeté en VHDL-AMS et intégré au reste du modèle. Ce modèle final servira ainsi à valider le

dimensionnement du composant dans son système. Il permettra également de réaliser une étude de sensibilité vis-à-vis des paramètres incertains (géométriques et physiques) du composant magnétique, dans son environnement système.

Sur la Figure 5-21, nous présentons les résultats de la simulation système obtenue, dans Portunus, pour le contacteur intégrant le modèle réluctant.

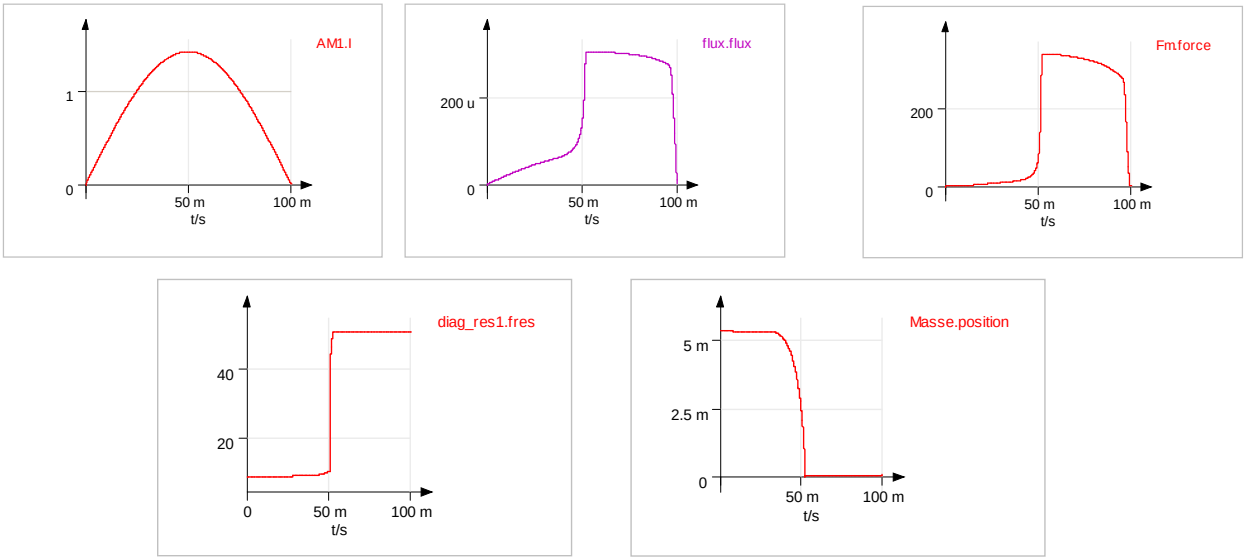


Figure 5-21 : Résultats de simulation du contacteur en E complet avec le modèle réluctant

Les différents niveaux de précision de ces modèles doivent être traités en association avec leur disponibilité (modèle numérique ou analytique disponible ou non) et leur propriétés (paramétrées ou pas, …). Ainsi, dans le cadre d'une simulation système en phase de conception détaillée, nous privilégierons plutôt les modèles magnétiques des tables de flux. Par exemple, cela permet de mieux prendre en compte les phénomènes de saturation que les réluctances gèrent mal. D'après les courbes de simulation de la Figure 5-22, nous constatons que le modèle réluctant sature plus que le modèle des tables issu de simulations numériques éléments finis. Ceci s'explique, car le modèle réluctant prend mal en compte les saturations.

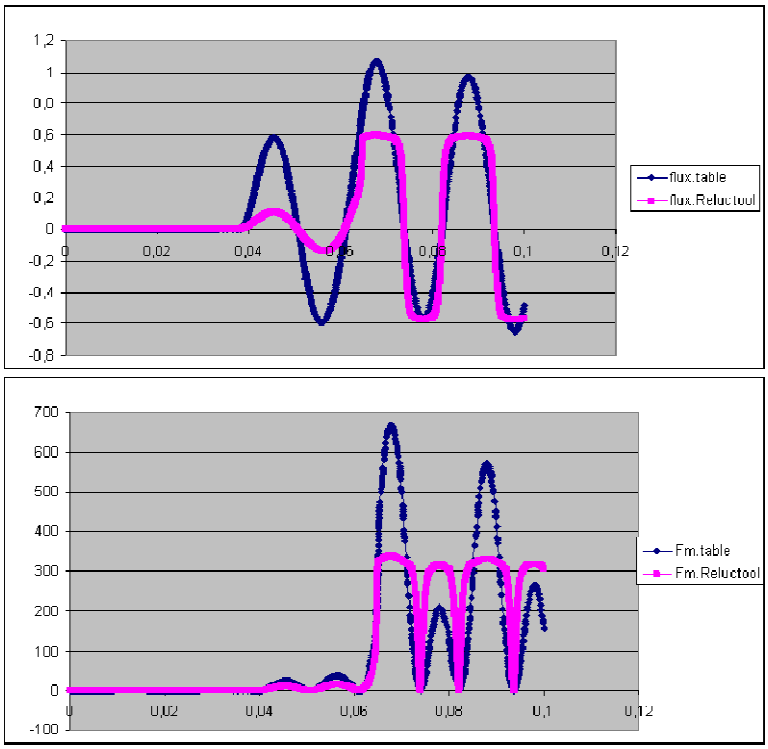


Figure 5-22 : Comparaison de la force et du flux magnétique obtenus par les modèles réluctant et par tables

Sur la Figure 5-23, nous présentons les différents résultats du modèle du contacteur complet, avec le modèle des tables, en position de fermeture.

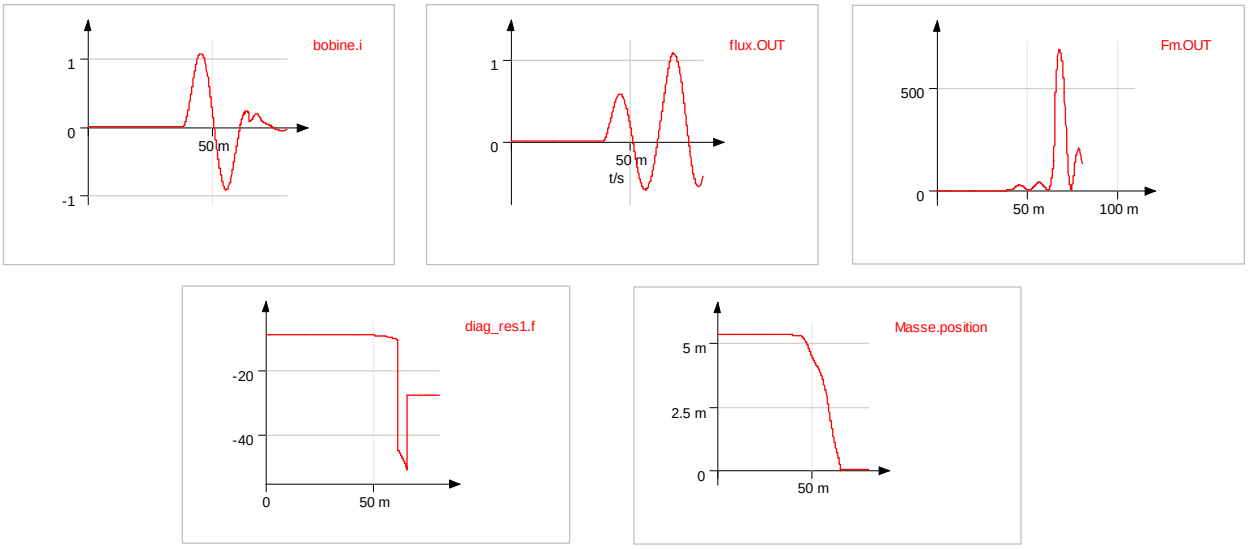


Figure 5-23 : Résultats de simulation du contacteur en E avec le modèle des tables d'effort/flux sous Portunus

D. Discussions des résultats de simulation et de leurs validations

Les différents modèles de l'actionneur ont été testés en régime dynamique. La validation a été réalisée par rapport à des cas tests expérimentaux et en définissant des plans de vérification de l'actionneur par Schneider Electric (non présenté dans ce chapitre). Certains résultats sont présentés dans le rapport de [HUM-12]. Les courbes obtenues respectent bien le fonctionnement théorique de l'actionneur en E présenté au tout début de ce chapitre.

La Figure 5-24 présente un exemple de cas de test qui a été réalisé, suite à des améliorations du modèle de fermeture du contacteur. Nous constatons que les courbes de simulations et d'essais sont quasiment superposables jusqu'à l'impact des pôles. Au-delà les frottements liés ont été difficiles à modéliser [HUM-12].

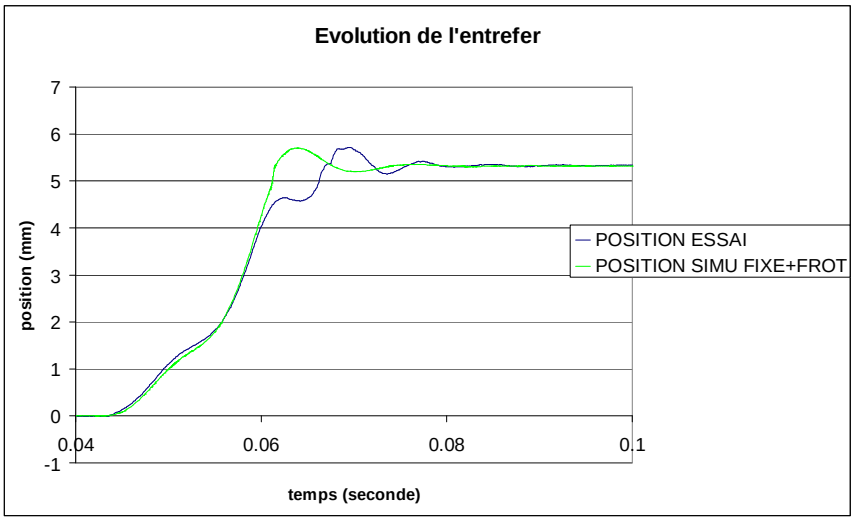


Figure 5-24 : Evolution de la position de la masse mobile (essais et simulations) [HUM-12]

Certaines précautions sont cependant à prendre en compte quant à la comparaison des résultats. Ceux-ci proviennent du fait que nous avons exploité des simulateurs VHDL-AMS différents. Il devient donc difficile de les comparer finement. En effet, dans le chapitre 1, nous avons précisé, que le même modèle VHDL-AMS peut présenter des différences de comportement d'un simulateur à un autre, même si les réglages des simulateurs, en termes de méthode d'intégration et de précision sont les mêmes.

D'une manière générale, quand on aborde la phase d'intégration, pour un concepteur du domaine du magnétisme, il est parfois souhaitable de tester le comportement sur des modèles plus réalistes et plus fidèles (par exemple, de type éléments finis) afin de détecter les défaillances et affiner des études. Cela concerne, par exemple, les saturations locales dans les matériaux magnétiques. En effet, des phénomènes physiques dont l'influence était ignorée en phase de « top down » ont étés mis en évidence au niveau composant, et il est pertinent de les prendre en compte. Cependant, ces modèles peuvent conduire, d'une manière générale, à une complexité inutile voire incompatible avec des objectifs de calcul souhaitable dans cette phase de validation de la conception (pour le concepteur intégrateur par exemple). Les modèles analytiques et les modèles réduits constituent une alternative intéressante.

Notons aussi que le format des modèles représentés par des tables de flux est le plus générique. Lorsqu'ils sont disponibles, ces modèles sont facilement interchangeables, puisqu'il suffit de charger les nouvelles tables et de paramétrer l'actionneur pour que le modèle soit à jour.

Une difficulté majeure, pour valider les résultats obtenus par ces différents modèles magnétiques, est leur domaine de validité par rapport à la dynamique mécanique globale. Nous allons illustrer ceci avec le modèle de fermeture de l'actionneur, en utilisant des tables de flux. Lorsque l'actionneur est fermé, dans le cas d'une alimentation continue, nous avons pu observer que l'effort magnétique est maximum. Cependant, pour les actionneurs

alternatifs, comme c'est le cas ici, le maintien est moins évident, car la tension oscille indéfiniment et par conséquent l'effort également. La Figure 5-25 présente certains résultats obtenus lors de simulations avec ce modèle.

Cette courbe montre bien que l'effort magnétique retombe à zéro périodiquement, ce qui n'est pas souhaitable puisqu'elles entraînent des rebonds au niveau des contacts. En effet, lorsque l'effort magnétique est supérieur à la force de rappel, le noyau mobile se déplace jusqu'à la fermeture de l'actionneur [HUM-12].

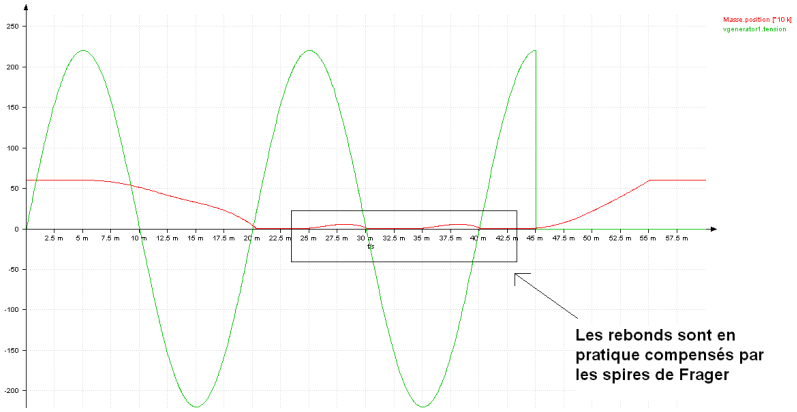


Figure 5-25 : Evolution de l'entrefer (rouge) avec une tension sinusoïdale d'entrée (vert)

En pratique, pour supprimer ce phénomène, les concepteurs ont recours à des spires de Frager enroulant une partie du noyau fixe de l'actionneur. Le rôle de ces spires est de déphaser une partie du flux qui traverse l'actionneur, en permettant de garder la composante continue de la force magnétique toujours supérieure à la valeur maximale de rappel du ressort [HUM-12]. La nécessité de modéliser ces spires de Frager conduit à re-développer un autre modèle pour simuler la phase de maintien du ressort. Dans ce cas, un réseau de réluctances est un excellent moyen de les modéliser rapidement et simplement (Figure 5-26).

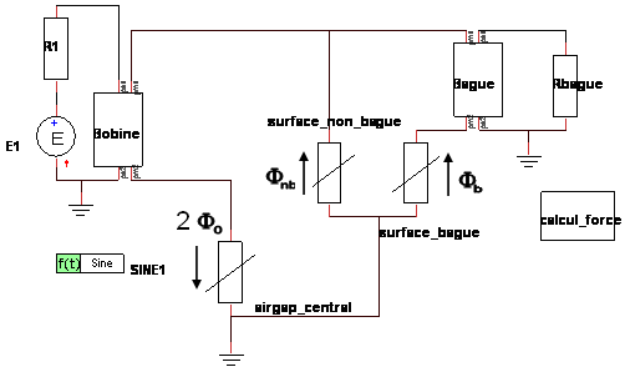


Figure 5-26 : Modèle de réseau de réluctances associé à l'introduction des spires de Frager [HUM-12]

La validation expérimentale, qui a été effectuée par Schneider Electric dans notre étude, peut faire émerger certaines différences par rapport au prototype virtuel. Ces différences, selon leur importance, conduit à faire un retour sur le processus de modélisation effectué, et à adapter les modèles concernés. Ce problème peut être rencontré pour des composants dont la modélisation « métier » n'a pas assez été poussée. C'est par exemple notre cas pour le modèle mécanique dans lequel tous les frottements ne sont pas pris en compte. Leur impact sur la courbe de déplacement du noyau est loin d'être négligeable, d'autant plus dans le cas des

actionneurs alternatifs. Les résultats expérimentaux peuvent nous aider à localiser les zones les plus affectées par les frottements, constituant ainsi des essais de calage de modèle dans le cycle de prototypage virtuel.

L'exemple du contacteur ainsi que les différents problèmes que nous avons évoqués pour la validation des systèmes, montre l'intérêt de l'approche multi-échelle que nous avons illustrée ici pour les composants magnétiques. En effet, elle permet de donner de la cohérence et de la prédictibilité aux modèles que l'on utilise en pratique.

Conclusion

Dans ce chapitre, nous avons illustré les aptitudes du langage normalisé VHDL-AMS à supporter un cycle de conception des systèmes multi-physiques à travers la modélisation et la simulation. Notre cas de test a consisté à modéliser un actionneur électromagnétique entièrement en VHDL-AMS. Pour cela, nous avons exploré :

- plusieurs modèles mettant en jeu des domaines physiques différents,
- plusieurs niveaux d'abstraction et de précision,
- différents caractères de généricité et de réutilisabilité,
- un mode d'utilisation descriptif ou prédictif.

Cela a aussi été l'opportunité de préciser les informations fondamentales qui doivent accompagner un modèle au-delà de son code.

Notre application a été réalisée sous différents outils au travers de différents tests, illustrés dans les travaux de G. Hummler tels que le maintien aux chocs avec des tensions seuil d'ouverture et de fermeture [HUM-12].

La modélisation a été enrichie avec la prise en compte des différents modèles fins, réduits et grossiers. Une comparaison avec des mesures expérimentales, en collaboration avec Schneider Electric, a permis de compléter ce prototypage virtuel par une validation expérimentale avec un prototype réel.

Conclusion générale

Dans nos travaux de thèse, nous avons traité un aspect de la problématique de la conception des systèmes mécatroniques, dans le cadre du projet ANR MoCoSyMec, en nous focalisant sur l'interaction des modèles dans les différents niveaux du cycle de conception. Nous avons plus particulièrement illustré nos solutions sur des applications de micro et macro-systèmes électromagnétiques.

Nous avons formulé cette gestion de l'interopérabilité des modèles dans le contexte du prototypage virtuel fonctionnel (PVF) afin de pousser le plus loin possible l'exploitation de modèles dans les différentes étapes de conception.

Nous avons appréhendé différentes problématiques concernant les modèles pour la simulation dynamique de systèmes en termes de formalisme de description, de méthodes de résolutions, de méthodes de couplages et de leurs différentes implémentations informatiques.

Ainsi, dans le chapitre 1, nous avons étudié la conception des systèmes mécatroniques, en nous focalisant cependant du point de vue illustratif sur des actionneurs électromécaniques et sur des MEMS magnétiques. Cet état de l'art, nous a permis de mettre en lumière les besoins émergeants des concepteurs pour la conception des systèmes multi-physiques complexes. Il en est ressorti la nécessité d'avoir recours à des méthodologies de conception plus adaptées, pour des objectifs de dimensionnement par optimisation et de simulation dynamique, en allant du composant physique au système physique. Nous avons retenu la méthodologie de prototypage virtuel fonctionnel (PVF).

Nous avons aussi montré l'importance de la phase de modélisation. Nous avons étudié les différents formalismes permettant de supporter l'interopérabilité et la réutilisabilité entre les modèles et les outils, tout au long du cycle de conception. Cela nous a permis de positionner nos choix : le langage normalisé VHDL-AMS pour les modèles ouverts (modélisation descriptive), et les composants logiciels ICAr pour les modèles fermés (boîte noire).

Dans le chapitre 2, nous nous sommes focalisés sur la problématique de la simulation dynamique des systèmes. Nous avons analysé et caractérisé les systèmes dynamiques mixtes en examinant tout particulièrement les difficultés liées à leur nature, leurs représentations et à la causalité des modèles. Nous avons aussi détaillé les deux formalismes retenus dans nos travaux en mettant en évidence leurs atouts et leurs limites pour supporter les différentes natures de modèles mises en jeu.

Nous nous sommes ensuite focalisés sur les systèmes décrits par des équations différentielles algébriques (DAE), en mettant l'accent sur leurs caractéristiques en termes de description et de méthodes numériques de résolution. Nous avons finalement spécifié et développé une extension de la norme ICAr pour supporter la description de modèles par des DAE. Nous avons enrichi l'environnement CADES de solveurs supportant la résolution de DAE, avec ou sans aspect évènementiel. Cet enrichissement permet de pouvoir exploiter ces solveurs en interne dans des composants ICAr, couplés aux modèles DAE, rendant ainsi le composant autonome en terme de simulation, et par exemple utilisable en dimensionnement. Ces modèles DAE peuvent aussi être implémentés dans des simulateurs dédiés (métiers) pouvant notamment supporter des ICAr.

Dans le chapitre 3, nous nous sommes intéressés à la problématique de l'interopérabilité dans le cycle de conception au travers des couplages entre modèles, outils métiers de modélisation magnétiques et outils de simulation multi-physiques, en exploitant les formalismes ICAr et VHDL-AMS.

Cela nous a conduit proposer des solutions logicielles et des modules de génération ou d'adaptation de modèles, et ce pour plusieurs environnements de simulation système (SMASH et Portunus). La solution d'interopérabilité proposée repose sur le concept de « plug-in ».

Nous avons spécifié et mis en oeuvre deux implémentations et nous les avons validées à travers diverses applications. La première solution exploite les fonctions externes de VHDL-AMS ; nous l'avons validée dans l'environnement SMASH, le seul à ce jour supportant cette fonctionnalité. La deuxième est une solution plus propriétaire faisant intervenir les API spécifiques des simulateurs ; elle a été implémentée dans le simulateur Portunus et devrait être facilement transposable dans le simulateur Simplorer.

En termes d'application, nous avons exploité ces mécanismes de « plug-in » pour piloter Flux au travers des ICAr, en tant que serveur de calcul. Nous avons aussi pu réexploiter des composants ICAr préexistants dans les environnements de simulation système. Ceci nous a permis d'illustrer les capacités de la norme ICAr à gérer l'interopérabilité au travers de ses facettes et des « plug-in » correspondants implémentés dans différents environnements logiciels.

Dans le chapitre 4, nous avons étudié la transformation d'une description de modèle vers une autre (principalement vers le langage VHDL-AMS) à travers des concepts de l'ingénierie dirigée par les modèles (IDM).

Nous avons formalisé la description VHDL-AMS des systèmes électromagnétiques modélisés par réseaux de réluctances et ceux modélisés par champs magnétiques rayonnés en solutionnant les problèmes liés à certaines limites du langage lui-même : les limites de la composition de modèles dans les réseaux de réluctances et le calcul d'intégrales non temporelles pour la modélisation de champs magnétiques rayonnés.

Ensuite, comme les outils métiers dédiés peuvent généralement faciliter grandement la mise en œuvre de ces modèles, nous avons spécifié, formalisé et mis en œuvre la génération

de modèles dans le langage VHDL-AMS à partir de deux outils de modélisation métiers : RelucTool et MacMMems. Nous avons fait uniquement les implémentations sous Reuctool, en permettant la génération automatique du modèle VHDL-AMS d'un actionneur linéaire sous deux représentations VHDL-AMS. La première se veut une composition de modèle de composants issus d'une bibliothèque VHDL-AMS, générée à partir de Reluctool, donc cohérente. La seconde crée un modèle VHDL-AMS global des dispositifs, permettant ainsi une modélisation aisée des forces magnétiques par dérivation de la co-énergie.

Dans le dernier chapitre, nous avons appliqué la méthodologie PVF à la problématique de conception des actionneurs magnétiques, en collaboration avec Schneider Electric. En particulier, nous avons étudié la modélisation multi-physique, de précision multi-niveau, dans les différentes phases de la conception. Le langage VHDL-AMS et les différentes solutions d'interopérabilité proposées dans nos travaux, nous ont permis d'étudier pleinement un actionneur linéaire dans le contexte du PVF.

Suite à ces travaux, de nombreuses perspectives nous sont offertes.

Concernant l'extension que nous avons réalisée dans la norme ICAr, il manque des générateurs dédiés pour les ICAr ODE et DAE. D'une part, ces générateurs pourront être réalisés à partir du langage Modelica ou du langage VHDL-AMS, mais cela est lié à disponibilité de compilateurs. D'autre part, ces générateurs pourront être obtenus à partir d'une reformulation interne des modèles dans des outils existants tels que Reluctool, pour générer directement les modèles dynamiques au format ICAr DAE/ODE.

Certains problèmes de dimensionnement s'appuient sur des simulations dynamiques. Le pilotage de simulateur peut être lourd et beaucoup de simulateurs n'offrent pas la possibilité de s'arrêter sur un critère évènementiel (par exemple, atteinte d'une position de butée pour un déclencheur EM). Pour traiter ces problèmes, il sera donc intéressant de compléter et d'intégrer les solveurs ODE/DAE en interne dans des ICAr. Il faudra alors permettre un pilotage conditionnel (par exemple arrêt sur évènement), gérer les paramétrages des solveurs pour supporter les variations de dynamique. Par exemple, ces dernières peuvent être introduites par le changement de paramètres de conception lors des itérations d'une optimisation (de nouveaux paramètres géométriques et physiques modifient les valeurs propres des systèmes d'états).

Dans le contexte d'une utilisation de modèle dynamique dans des procédures de dimensionnement par optimisation avec gradients, il serait intéressant de voir comment dériver ces simulations dynamiques autrement que par différences finies. Par exemples, on pourrait envisager de la différentiation automatique afin d'avoir une précision meilleure dans l'évaluation des jacobiens des modèles, et donc une meilleur convergence des algorithmes d'optimisation avec gradients.

Les méthodes de couplage que nous avons réalisées permettent de pallier certaines limites du langage VHDL-AMS. Maintenant, il est possible de l'étendre vers des applications plus complexes. Une limite peut être le risque de complexification du langage et la portabilité des modèles liée à la mise en œuvre des compilateurs VHDL-AMS.

Il nous reste aussi à concrétiser la mise en œuvre de plug-in ICAr dans Portunus, en rendant générique et automatique leur importation, à partir des implémentations que nous avons réalisées. Le fait d'avoir de nouvelles facettes de résolutions dynamiques dans les ICAr va nous permettre d'enrichir les plug-ins, notamment vers Matlab et Matlab/simulink.

En ce qui concerne la génération automatique des modèles vers le langage VHDL-AMS, il est important d'étendre le générateur de RelucTool vers le traitement des capteurs et la prise en compte du comportement de matériaux magnétiques dépendant de la dynamique (modèle d'hystérésis de Gill Altertons, Preisac, Chimique, etc. [DO-10]).

Du côté des MEMS magnétiques, il faut maintenant mettre en œuvre les spécifications que nous avons définies pour l'outil MacMMems. Cette réalisation concrète des générateurs des équations à partir de MacMMems pourra se fait de deux manières : soit en passant par le langage SML, soit en réadaptant le contenu de MacMMems pour générer le modèle dans le format XML proposé dans Reluctool.

Cependant, le langage SML est essentiellement adapté à la description de modèles de dimensionnement ou de modèles statiques. De par sa syntaxe, il ne permet pas, de formuler des modèles dynamiques ou aux dérivées partielles. Il faut donc se poser la question de l'extension de ce langage ? Une autre solution est de lui conserver ses fonctionnalités actuelles et de privilégier d'autres langages tels que VHDL-AMS ou Modelica pour les modèles dynamiques. La perspective à plus long terme est de pouvoir réaliser des dimensionnements sur des modèles dynamiques.

Liste des publications

Revues Internationales

A.Rezgui, L.Gerbaud, B.Delinchant, "VHDL-AMS to support DAE-PDE coupling and multilevel modelling", IEEE Transactions on Magnetics, Vol. 48, No. 2, February. 2012.

Conférences Internationales:

F. Verdiere, **A. Rezgui**, S. Gaaloul, B. Delinchant, L.Gerbaud, F. Wurtz, X. Brunotte, "Modelica Models Translation Into Java Components for Optimization and DAE Solving", 14th International Conference on Modelling and Simulation (UKSim 2012), Cambridge University - UK, Mars, 2012

A.Rezgui, L.Gerbaud, B.Delinchant, "VHDL-AMS to support DAE-PDE coupling and multilevel modelling", Compumag 2011, Sydney - Australia, July, 2011

A.Rezgui, L.Gerbaud, B.Delinchant, R.Barrak, A.Ghazel, "Unified modeling technique using VHDL-AMS and software components" ELECTRIMACS 2011, Cergy-pontoise – Paris, June 2011.

D. Duret, L. Gerbaud, F. Wurtz, **A. Rezgui**, B. Delinchant, B. Cogitore "Optimisation and simulation methodology for passive ADSL splitter design", XIth International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD-2010), Tunis-Gammarth - Tunisie, October, 2010

Références bibliographiques

[ABA-98] A. Abakar, J-L. Coulomb, Y. Maréchal, "*Radial Basis Function Network for Acceleration of Genetic Algorithm Optimization Procedures*", Symposium on Numerical Field Calculation in Electrical Engineering, Austria, 1998

[ADA-07] Adapted Solution, "*Using the Portunus C-Programming Interface*", document technique, 2007

[ADA-09] Adapted-Solutions/Cedrat, "*Flux-Portunus Co-Simulation*" document technique, 2009

[AFN-08] AFNOR, "*Mecatronique – vocabulaire : Norme NF E01-010*", 2008

[AKO-84] G. Akoun, J-P. Yonnet, "*3D analytical calculation of the forces exerted between two cuboidal magnets*", IEEE Transactions on Magnetics Vol. 20 No. 5, 1984

[ALC-98] D-G. Alciatore, M-B. Histand, "*Introduction to Mechatronics and Measurement Systems*", McGraw Hill, 1998

[ALL-03] L. Allain, "*Capitalisation et traitement des modèles pour la conception en génie électrique*", thèse de l'Institut National Polytechnique de Grenoble, 2003

[ALL-09] H. Allag, J.-P. Yonnet, B. Delinchant, "*Analytical Calculation of the Interactions between two Cylinder-shaped Magnets*", COMPUMAG, BRESIL, 2009

[AMM-07] I. Ammar, "*Conception collaborative en génie électrique*", thèse de l'Institut National Polytechnique de Grenoble, 2007

[ASH-02] P. J. Ashenden, G D. Peterson, D A. Teegarden, "*The System Designer's Guide to VHDL-AMS : Analog, Mixed-Signal, and Mixed-Technology Modeling*", Morgan Kaufmann Publishers, San Francisco, 2002,

[ASH-04] P.J. Ashenden, G. D. Peterson, D. A. Teegarden "*The System Designer's Guide to VHDL-AMS: Analog, Mixed-Signal, and Mixed-Technology Modeling*", Paris, Dunod, 2004.

[ATI-00] E. Atienza, M. Perrault, F. Wurtz, V. Mazauric, J. Bigeon "*A methodology for the sizing and the optimization of an electromagnetic release*", IEEE Transactions on Magnetics, 2000.

[ATI-03] E Atienza, "*Méthodologie et outils pour le dimensionnement*", Thèse de l'Institut National Polytechnique de Grenoble, soutenue le 4 juillet 2003.

[BAC-90] R. Bachmanna, L. Brülla, b, Th. Mrziglodb, U. Pallaskea, AG Bayer, "*On methods for reducing the index of differential algebraic equations*", Computers & Chemical Engineering, Vol 14, 1990

[BEE-07] S-P Beeby et al, "A *micro electromagnetic generator for vibration energy harvesting*". Micromech. Microeng, 2007

[BER-03] T. Bertram, F. Bekes, R. Greul, O. Hanke, C. Has, J. Hilgert, M. Hiller, O. Ottgen, P. Opgen-Rhein, M. Torlo, D. Ward, "*Modelling and simulation for mechatronic design in automotive systems*". Control Engineering Practice, 2003.

[BEZ-03] J. Bézivin, "*La transformation de modèles*". INRIA-ATLAS & Université de Nantes, Ecole d'Eté d'Informatique CEA EDF INRIA, 2003

[BLA-05] X.Blanc, "*MDA en action, Ingénierie logicielle guidée par les modèles*", Eyrolles, 2005

[BOC-98] J.-C. Bocquet, "*Ingénierie simultanée, conception intégrée", dans "Conception de produits mécaniques : méthodes, modèles, outils*", M. Tollenaere, Hermès, 1998.

[BOU-10] H. Boussetta, "*Modélisation multi-physiques et simulation globale de systèmes autonomes sur puce*", thèse 'Institut polytechnique de Grenoble, 2010

[BRA-06] J. Brauer, "*Magnetic Actuators and Sensors*", Johny wily and sons, INC publications, IEEE magnetic Society, 2006

[BRE-96] K. E Brenan, S. L Campbell, L. R Petzold, "*Numerical Solution of Initial-Value Problems in Di_erential-Algebraic Equations*", SIAM's Classics in Applied Mathematics. Siam, Philadelphia, 1996

[BRO-94] P. Brown, A. Hindmarsh, L. Petzold, "*Using Krylov methods in the solution of large-scale differential-algebraic systems*". SIAM J Sci Comput, 1994

[BRO-98] P. Brown, A. Hindmarsh, L. Petzold, "*consistent initial condition calculation for differential-algebraic systems*", SIAM J. SCI. COMPUT. Society for Industrial and Applied Mathematics, Vol. 19, 1998

[BUS-01] N. Bushyager, MM. Tentzeris, L. Gatewood, and J. DeNatale "*A novel adaptive approach to modeling MEMS tunable capacitors using MRTD and FDTD techniques*", In Microwave Symposium Digest, 2001

[CAL-01] M. Caldora Costa, J.-L. Coulomb, Y. Marechal, S-I Nabeta, "*An adaptive method applied to the diffuse element approximation in optimization process*", IEEE Transactions on Magnetics, Vol 37 Issue5, 2001

[CAM-96] S. Campbell, K. Brenan, L. Petzold, "*Numerical Solution of Initial Value Problems in Differential Algebraic Equations*", SIAM, Philadelphia, 1996

[CEL-91] F. Cellier, "*Continuous System Modeling*", Springer-Verlag, 1991.

[CHA-08] Y-A Chapuis, Y. Hervé, "*Conception de systèmes hétérogènes et futur de la CAO multi-physique*", Journée SOC-SIP, 2008

[CHE-07] H. L. Chetouani, "*Microsystèmes et micromanipulation à lévitation diamagnétique : conception, réalisation et application à la micro fluidique digitale et à la biologie*", Thèse Institut National Polytechnique de Grenoble, 2007

[CHU-90] Y. Chung, A. W. Westerberg, "*proposed numerical algorithm for solving nonlinear index problems*", Ind. Eng. Chem. Res., 1990

[CIA-90] P.G. Ciarlet, "*Introduction à l'analyse numérique matricielle et à l'optimisation*", Dunod, 1990.

[COU-85] J-L Coulomb, J-C. Sabonnadière, "*CAO en électrotechnique*", Éditeur : Paris, Hermès, 1985

[COU-08] J.L Coulomb, "*Optimization*", chapter 14 of "*The Finite Element Method for Electromagnetic Modeling* 3", Wiley ISTE, 2008.

[CUG-06] O. Cugat, , G.Reyne, J. Delamare, H. Rostaing., "*Novel magnetic micro-actuators and systems (MAGMAS) using permanent magnets*" Sensors and Actuators, 2006

[CUG-07] O. Cugat, , G.Reyne, J. Delamare "*Magnetic Microsystems : Mag-MEMS*", ISTE Publishing Company, 2007

[DAV-97] R. David, "*Modeling of Hybrid Systems by Continuous or Hybrid Petri Nets*", International Workshop Petri Nets and Performance Models, Saint-Malo, 1997.

[DEL-02] B. Delinchant, V. Riboulet, L. Gerbaud, P. Marin, F. Noël, F. Wurtz "*Cooperative Design among Mechanical and Electrical Engineers over the Internet: some Implications for Tools supporting Human Communication Between Various Professional Cultures*", IEEE Transactions on Professional Communication. Special Issue : International Communication, Technology, and Culture. 2002

[DEL-02] B. Delinchant, F. Wurtz, J. Fandino, "*Mixing of FEM and Analytical Modeling for the Preliminary Design of a Transformer*", International workshop on OIPE, Lodz, 2002.

[DEL-03] B. Delinchant, "*Un Environnement à base de Composants, Intégrant le Concepteur et ses Outils, pour de Nouvelles Méthodes de CAO*", thèse à Institut National Polytechnique de Grenoble, 2003

[DEL-08] J. Delamare, S. Basrour, "*Hybridization of Magnetism and Piezoelectricity for an Energy Scavenger based on Temporal Variation of Temperature*", DTIP, Nice, 2008

[DEL-11] B. Delinchant, "*La CAO et l'optimisation de systèmes, une approche par couplages dynamiques de composants*", HDR, Institut National Polytechnique Grenoble, 2011

[DEL-12] B. Delinchant, P.Y. Gibello, F. Verdière, F. Wurtz, "*Distribution des services de calcul pour la conception et la gestion optimale des bâtiments : perspectives des approches composants déployés via le cloud computing*", Rencontres de l'AUGC-IBPSA, Chambéry 2012

[DEL-12] B. Delinchant, L. Estrabaud, L. Gerbaud, F. Wurtz "*Multi-criteria design and optimization tools*", chapter 5 of Integrated Design by Optimization of Electrical Energy Systems, Edited by Xavier Roboam, Wiley ISTE, 2012.

[DES-04] P. Desgreys, "*Composant optoélectronique : diode laser*" in "Modélisation de systèmes complexes avec VHDL-AMS", EPFL, Ed. Lausanne, Presses Polytechniques Universitaires Romandes, 2004.

[DO-10] T. P. DO ; "*Simulation dynamique des actionneurs et capteurs électromagnétiques par réseaux de réluctances : modèles, méthodes et outils*", Thèse de l'Institut Polytechnique de Grenoble, 2010.

[DUP-06] B. Du Peloux, "*Modélisation des actionneurs électromagnétiques par réseaux de reluctances*", Thèse de l'Université Joseph Fourier de Grenoble, 2006

[DUP-08] B. Du Peloux, G. Lacombe, "*Engineering-focused software for the design of the drive of electrical machines*", International Conference on Electrical Machines, 2008

[DUR-10] D. Duret, L. Gerbaud, F. Wurtz, A. Rezgui, B. Delinchant, B. Cogitore, "*Optimisation and simulation methodology for passive ADSL splitter design*", International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD), Tunis, 2010.

[ENC-09] P. Enciu, "*Dérivation automatique pour le calcul des sensibilités appliqué au dimensionnement en génie électrique*", Thèse de l'Institut National Polytechnique de Grenoble, 2009

[FEL-02] F. Felgner, S. Agustina, B. Cladera, R. Mertz, L Litz, "*Simulation of thermal building behaviour in Modelica*", Modelica Conference, 2002

[FIS-35] R-A. Fisher, "*The design of experiments*", Oliver and Rod, 1935

[FIS-97] P. Fishwick, "*Computer simulation : growth through extension*". Transactions of the Society for Computer Simulation International, 1997

[FIS-03] V. Fischer, L. Allain, L. Gerbaud, "*RAMA :Alightweight rule-based tool for expressions analysis and code generation*", Proc. Simul. Symp. Exhibition, Delft,2003

[FIS-04] V. Fischer, "*Composants logiciels pour le dimensionnement en génie électrique. Application a la résolution d'équations différentielles*", Thèse de l'Institut National Polytechnique de Grenoble, 2004

[FMI-10] FMI for Model Exchange v1.0. http://www.functional-mockup-interface.org/specifications/FMI_for_ModelExchange_v1.0.pdf

[FRI-98] P. Fritzson, V. Engelson "*Modelica-A unified object-oriented language for system modeling and simulation*", Object Orienting Programming, Vol 1445, 1998

[FRI-04] P. Fritzson, "*Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*" John Wiley & Sons, 2004

[FUR-07] S. Furic, "*Hybrid acausal modeling using Modelica : Presentation of Modelica*", Journées « outils », INSA Lyon, 2007

[GAA-11] S. Gaaloul, F. Verdière, B. Delinchant, F. Wurtz, "*Interface Component Architecture for building simulation*", Conference of International Building Performance Simulation Association, Sydney, 2011.

[GAA-12] S. Gaaloul, X. H. Binh Le, B. Delinchant, F. Wurtz, S. Ploix, "*Architecture à composants de co-simulation appliquée au couplage de la thermique du bâtiment au comportement de l'usager*", Rencontres de l'AUGC-IBPSA, Chambéry 2012

[GAL-08] S. Galmiche, "*Méthode numériques et interfaces graphiques pour un outil de simulation des actionneurs électromécaniques*" ; Stage de fin d'étude, Schneider Electric et le laboratoire Génie Electrique de Grenoble, 2008.

[GAU-12] J-M. Gauthier, "*Vérification de propriétés SysML par transformation de modèle vers VHDL-AMS*", Rapport de recherche, université de franche-comte, 2012

[GEA-71] C. W Gear, "*The simultaneous numerical solution of differential algebraic equations*". IEEE Trans. Circuit Theory, 1971

[GEA-85] C. W Gear, B. Leimkuhler, G. K. Gupta, "*Automatic integration of Euler-Lagrange equations with constraints*". J. Comput. Appl. Math, 1985.

[GEA-88] C. W. Gear, "*Differential-algebraic equation index transformations*". SIAM. J. Sci. Stat. Comp, 1988.

[GIB-01] D. Gibson, H. Carter, C. Purdy. "*The use of hardware description languages in the development of microelectromechanical systems*", Analog Integrated Circuits and Signal Processing, 2001.

[GOW-03] S. Gowers, "*Développement d'un environnement de co-conception distribuée basé sur une étude entre cultures professionnelles distinctes*", Projet de Fin d'Etudes, Institut National Polytechnique De Grenoble, 2003.

[GRI-86] E. Griepentrog, R. Marz, "*Differential Algebraic Equations and Their Numerical Treatment*", Teubner-Textezur Math., Leipzig, 1986.

[GUE-04] R. Guelaz, D. Kourtiche, M. Nadi, and Y. Herve, "*Ultrasonic piezoceramic transducer modeling with VHDL-AMS IEEE*", Sensors Proceedings of IEEE, 2004.

[GUE-06] R. Guelaz, "*Modélisation de systèmes ultrasonores avec VHDL-AMS : Application à la mesure du paramètre de non linéarité B/A*", Thèse de l'Université Henri Poincaré, Nancy, 2006.

[GUE-05] S. Guerin, J.L. Coulomb, G. Cauet, "*Study of the inverse problem resolution quality. Application to low-.eld magnetostatic*", COMPEL, Vol. 24, 2005.

[GUI-07] D. Guihal, "*Modélisation en langage VHDL-AMS des systemes pluridisciplinaires*", Thèse de l'Université Toulouse II, 2007.

[GUI-11] P. Guitard, "*Automatisation du procédé de modélisation d'actionneurs électromagnétiques*", projet de fin d'étude, 2011

[HAM-93] D. C. Hamill, "*Lumped equivalent circuits of magnetic components : the gyrator-capacitor approach*", IEEE Trans. Power Electron., vol. 8, 1993

[HAM-05] J.C.Hamon. "*Méthodes et outils de la conception amont pour les systèmes et les microsystèmes*", thèse à l'Institut National Polytechnique de Toulouse, 2005

[HAI-80] E., Hairer, C.Lubich, M. Roche, "*The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*", Lecture Notes in Mathematics. Springer-Verlag, 1980.

[HAI-89] E. Hairer, C. Lubich, M. Roche, "*The numerical solution of differential-algebraic systems by Runge-Kutta methods*" Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1989.

[HAI-91] E. Hairer, G.Wanner, "*Solving Ordinary Differential Equations II-Stiff and Differential-Algebraic Problems*", Springer-Verlag, 1991

[HAI-08] E. Hairer, S. P. Norsett, G. Wanner, "*Solving OrdinaryDifferential Equations I-Nonstiff Problems*", Springer-Verlag, 2008.

[HAR-96] F. Harshama, M. Tomizuka, and T. Fukuda, "*Mechatronics-what is it, why, and how?-and editorial*", IEEE/ASME Trans. on Mechatronics, 1996.

[HER-02] Y.Hervé, "*VHDL-AMS : Applications et enjeux industriels*", Dunod, Paris, 2002.

[HER-03] Y. Hervé, "*Simple models for complex systems: A-FSM template", Forum on specification and Design Language*, 2003

[HER-03] Y.Hervé, "*Virtual prototyping with VHDL-AMS*", IEEE International Conference on Industrial Technology, 2003

[HER-02a] Y.Hervé," *VHDL-AMS : un atout pour la conception système* ", TAISA, 2002.

[HER-06] Y. Hervé, P. Desgreys "*Functional Virtual Prototyping Design Flow and VHDL-AMS*". FDL, 2006

[HER-12] Y. Hervé, P. Desgreys, "*Design Technology for Heterogeneous Embedded Systems*", ch11, 2012.

[HIN-80] A. C. HindMarsh, "*LSODE and LSODI, two new initial value ordinary differential equations solvers*". ACM SIGNUM Newsletter, 1980

[HIN-83] A. C. HindMarsh, "*ODEPACK, A Systematized Collection of ODE Solvers*", IMACS Transactions on Scientific Computation, Vol.1, 1983.

[HUM-12] G. Hummler, "*Modélisation système d'un actionneur électromagnétique*", projet de fin d'étude, 2012

[IBR-09] T. Ibrahim "*Contribution au développement de modèles pour l'électronique de puissance en VHDL-AMS*", Thèse, Institut National des Sciences Appliquées de Lyon, Jan-2009

[ISE-08] S. Vasilije Vasić, P. Mihailo Lazarević, "*Standard Industrial Guideline for MechatronicProduct Design* ", FME Transactions, VOL. 36, 2008

[JAN-10] J. L. G. Janssen , J. J. H. Paulides , J. C. Compter and E. A. Lomonova "*Three dimensional analytical calculation of the torque between permanent magnets in magnetic bearings*", IEEE Trans. Magn., vol. 46, 2010.

[JAR-92] R. B. Jarvis, C. C. Pantelides, "*DASOLV : a differential algebraic equation solver*", Technical report, Centre for Process Systems, Engineering, Imperial College, London. 1992

[JAR-10] A. Jardin, "*Contribution à une méthodologie de dimensionnement des systèmes mécatroniques : anlayse structurelle et couplage à l'optimisation dynamique* ", thèse Institut National des Sciences Appliquées de Lyon, 2010.

[JOH-97] H. John, H. Beverly "*Differential Equations : A Dynamical Systems*", 1997

|KAU-09] P. Kauffmann, "*lévitation diamagnétique sur micro aimants : Applications à la micro fluidique digitale et à la biologie*", Thèse Institut National Polytechnique de Grenoble, 2009

[KAU-09a] P. Kauffmann, V. Haguet, G. Reyne, B. Delinchant "*Efficient multipoles modeling for linear magnetized beads manipulations*" manuscript, published in "Scientific and Clinical Applications of Magnetic Carriers, Rostock : Germany, 2009

[KON-05] D. Konstantas, JP. Bourrières, M. Léonard, N. Boudjlida. "*Interoperability of enterprise software and applications*". INTEROP-ESA, Geneva Switzerland, 2005.

[KOS-06] R. Kossel, W. Tegethoff, M. Bodmann, N. Lemke "*Simulation of complex systems using Modelica and toll coupling*", Modelica Conference. 2006.

[KRE-72] E. Kreyszig "*Advanced Engineering Mathematics*", New York : Wiley, 1972

[KRO-92] A. Kröner, W. Marquardt, E.D. Gilles, "*Computing consistentinitial conditions for differential-algebraic equations*" , Computers & Chemical Engineering, Vol 16, 1992

[KUN-06] P. Kunkel, V. Mehrmann, "*Differential-algebraic equations*", EMS Textbooks in Mathematics. European Mathematical Society, Zurich, 2006.

[KUR-03] I. Kurtev, K. Berg "*Model Driven Architecture based XML Processing*" Software Engineering Group, University of Twente, Proceedings of the ACM symposium on Document engineering, New York, 2003

[LEB-04] L. Lebensztajn, C.A.R. Marretto, M.C Costa, J.-L Coulomb, "*Kriging : a useful tool for electromagnetic device optimization*", IEEE Transactions on Magnetics, Vol 40, 2004

[LEB-12] M. Lebrun, "*Simulation et CAO en automatique et mécatronique*", Techniques de l'Ingénieur, 2012

[LEI-91] B. Leimkuhler, L. R. Petzold, C. W. Gear, "*Approximation Methods for the Consistent Initialization of Differential- Algebraic Equations*", SIAM Journal on Numerical Analysis, Vol. 28, 1991

[LEY-08] J. Lygeros, C. Tomlin, S. Sastry, "*Hybrid Systems : Modeling, Analysis and Control*", 2008

[LI-01] Q. Li, W-J. Zhang, L. Chen, "*Design for Control – A Concurrent Engineering Approach for Mechatronic Systems Design*", IEEE Transactions on Mechatronics, Vol. 6, 2001.

[LI-09] Z. Li, L. Zheng, H. Zhang, "*Modelling and simulation of PDE problems in Modelica*", Journal international Materials and Structural Integrity, vol3, 2009

[LIE-06] J. Lienemann, B. R. B. Evgenii, J. G. Korvink, and Ferber, "*MST MEMS compact modelling meets model order reduction: Requirements and benchmarks*", Linear Algebra and its Applications, vol. Vol.415, Issues 2-3, pp. 469{498, June 2006.

[LUN-05] H. Lundvall, P. Fritzson, "*Event Handling in the OpenModelica Compiler and Runtime System*", Modelica conferencec, 2005

[MAG-04] D. Magot, "*Méthodes et outils logiciels d'aide au dimensionnement. application aux composants magnétiques et aux filtres passifs*", Thèse de l'Institut National Polytechnique de Grenoble, 2004

[MAN-07] A. Mantooth, A. Francis, Y. Feng, W. Zheng. "*Modelling tools built upon the hardware description language foundation"*. IET Computers & Digital Techniques, 2007.

[MAR-08] C. Marin, "*Une approche orientée domaine pour la composition de services*", thèse Informatique de l'Université Joseph Fourier, 2008

[MAT-93] S.E. Mattsson, G. Söderlind, "*Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives*", SIAM J. Sci. Stat. Comput, 1993.

[MAZ-04] V. Mazauric, "*From thermostatistics to Maxwells equations : a variational approach of electromagnetism*", IEEE Transactions on Magnetics, Volume 40, 2004

[MED-00] N. Medvidovic, R.N. Taylor, "*A Classification and Comparison Framework for Software Architecture Description Languages*", IEEE trans. software engineering, vol. 26, 2000

[MEU-08] G. Meunier, "*The Finite Element Method for Electromagnetic Modeling*", Wiley ISTE, 2008.

[MIC-09] J. Michaelsen, J. Eiden. "*HumanComfort Modelica-library thermal comfort in buildings and mobile applications*". Modelica Conference, 2009

[MRC-96] Z. Mrcarica, H. Detter, D. Glozic, V. Litovski. "*Describing space-continuous models of microelectromechanical devices for behavioral simulation*", Proceedings of the conference on European design. Automation, 1996

[NIA-03] D. Niarchos, "*Magnetic MEMS: key issues and some application*s", Sensors and Actuators, vol. 109, 2003.

[NIK-07] P.V. Nikitin, C.J.R. Shi, "*VHDL-AMS based modeling and simulation of mixed-technology Microsystems: a tutorial*". INTEGRATION, the VLSI journal, 2007.

[NIK-03] P.V. Nikitin, C.R. Shi, B. Wan. "*Modeling partial differential equations in VHDL-AMS [mixed signal systems* applications]", In Proceedings IEEE International Systems-on-Chip, 2003

[NOR-92] O. Normand, "*Conception d'un outil général de simulation des systèmes de conversion d'énergie électrique et de leur commande*", Thèse de doctorat en génie électrique de l'Institut National Polytechnique de Grenoble, 1992

[NUG-04] K. Nguessan, "*Optimisation d'un déclencheur par méthode analytique et numérique*", Master Recherche de Génie Électrique de L'institut National Polytechnique Grenoble, 2004.

[PAH-96] G.Pahl, W. Beitz, "*Engineering Design: A Systematic Approach*", Springer-Verlag, 2nde édition, 1996.

[PAN-88] C. Pantelides. "*The Consistent Initialization of Differential–Algebraic Systems*". SIAM Journal of Scientific and Statistical Computing, 1988.

[PER-98] M. Perrault "*Optimisation d'un déclencheur à noyau plongeur : Internet de l'approche P.A.S.C.O.S.M.A*", DEA Génie Electrique, Institut National Polytechnique De Grenoble, 1998.

[PET-80] L. R. Petzold, "*Automatic Selection of Methods for Solving Stiff and Nonstiff systems of Ordinary Differential Equations*", Sandia National Laboratories, 1980.

[PET-83] L. R. Petzold, "*A description of DASSL: A differential algebraic system solver*". Scientific Computing, 1983.

[PET-83] L. R. Petzold, "*Differential algebraic equations are not ODE's*", SIAM J. Sci. Statist. Comput., 3 :367{384}, 1983

[PET-95] P. E. Van Kekenad, D. A. Yuenbe,L. R. Petzold, "*DASPK : A new high order and adaptive time-integration technique with applications to mantle convection with strongly temperature-and pressure-dependent rheology*", Geophysical & Astrophysical Fluid Dynamics, 1995

[PHA-11] P Pham-Quang, "*Modélisation magnéto mécanique d'un nano commutateur. Optimisation sous contraintes de fiabilité par dérivation automatique des programmes en Java*", thèse de l'Institut National Polytechnique de Grenoble, 2011.

[PHA-12] P. Pham-Quang, B. Delinchant. "*Java Automatic Differentiation Tool Using Virtual Operator Overloading*". In Recent Advances in Algorithmic Differentiation. S. Forth et al, 2012.

[RAD-93] K. Radhakrishnan, A. C. Hindmarsh, "*Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations*", LLNL report UCRL-ID-113855, 1993.

[RAK-07] H. L Rakotoarison, "*Méthode et outil de génération automatique de modèle pour l'optimisation fortement contrainte des microsystèmes magnétiques*", Thèse de l'Université Joseph Fourier de Grenoble, 2007

[RAK-07a] H. L. Rakotoarison, J. P. Yonnet, B. Delinchant, "*Using Coulombian approach for Modeling Scalar Potential and Magnetic Field of a Permanent Magnet with Radial Polarization*", IEEE Transactions on Magnetics., Vol. 43, 2007

[REZ-11a] A. Rezgui, L. Gerbaud, and B. Delinchant, "*VHDL-AMS to Support DAE-PDE Coupling and Multilevel Modelin*g", IEEE transactions on magnetics, Vol. 48, 2012.

[REZ-11] A. Rezgui, L. Gerbaud, B. Delinchant, R. Barrak, A. Ghazel "*Unified modelling technique using VHDL-AMS and software components*" , Electrimacs 2011, Cergy-Pontoise, France.

[ROS-05] H. Rostaing, J. Stepanek, B. Delinchant, J. Delamare, O. Cugat, "*Conception, modélisation et prototypage d'un micro-relais bistable magnétique*", revue internationale de génie électrique (RIGE), vol 8, 2005

[ROT-41] H. C. Roters, "*Electromagnetic Devices*" .Publisher: John Wiley & Sons; 99 editions, January 15, 1941.

[ROT-02] R. Rothfuss, M. Lasa, H. M. Heinkel, P. Tirigari, "*Systems Engineering in the Design of Mechatronic Systems*", International Journal of Vehicle Design, Vol. 28, 2002.

[SAL-00] L. Saldamli, P. Fritzon, "*Object oriented modeling with Partial Differential Equation*", workshop Modelica, Sweden, 2000.

[SCH-02] P. Schneider, E. Huck, S. Reitz, S. Parodat1, A. Schneider, P. Schwarz "*A modular approach for simulation-based optimization of MEMS*", Design Modeling, and Simulation in Microelectronics, Singapore, 2000

[SCH-04] H. P. Schöner, "*Automotive Mechatronics*", Control Engineering Practice, Vol. 12, 2004.

[SCH-05] M. Schlegel, F. Bennini, J. Mehner, G. Herrmann, D. Müller, W.Dötzel "*Analyzing and Simulation of MEMS in VHDL-AMS Based on Reduced Order FE-Models*", IEEE sensors journal, 2005.

[SHA-98] R. Shannon, "*Introduction to the art and science of simulation*" Proceedings of the 30th conference on Winter simulation. IEEE Computer SocietyPress, Pages 7–14 of. 1998

[SHE-00] S. Li, L. Petzold "*Software and algorithms for sensitivity analysis of large-scale differential algebraic systems*", Journal of Computational and Applied Mathematics - Special issue on numerical anaylsis 2000

[SHI-95] C-J Shi, A. Vachoux, "*VHDL-AMS design objectives and rationale*". Current Issues in Electronic Modeling, Kluwer Academic Publishers,1995

[SID-03] M. Sida, R. Ahola, and D. Wallner, "*Bleutooth transceiver design and simulation with VHDL-AMS*", The Chip, 2003.

[SIM-72]. G. F. Simmons, "*Differential Equations with Applications and Historical Notes*", New York, McGraw-Hill, 1972

[SMA-09] SMASH, "*Foreign C functions in VHDL-AMS and SPICE models*", Release 5.13, 2009

[SNA-04] S. Snaidero "*Modélisation multidisciplinaire VHDL-AMS de systèmes complexes : vers le Prototypage Virtuel*", Thèse à l'Université Louis Pasteur de Strasbourg, 2004.

[SOH-92] G.Sohlenius, "*Concurrent Engineering*", Annals of the CIRP 41, 1992.

[SUN-11] Q. Sun "*Étude et conception d'un processeur neuronal analogique très faible consommation. Application au pilotage d'un pacemaker de nouvelle génération*", Thèse à l'Université Louis Pasteur de Strasbourg, 2011.

[SZY-98] C. Szypersky, "*Component Software – Beyond Object-Oriented Programming*", Addison-Wesley, 1998

[THR-05] K. Thramboulidis, "*Model-Integrated Mechatronics – Toward a New Paradigm in the Development of Manufacturing Systems*", IEEE Transactions on Industrial Informatics, Vol. 1, 2005.

[TOL-98] M. Tollenaere, "*Conception de produits mécaniques : méthodes, modèles, outils*", Hermès, 1998.

[VAC-03] A. Vachoux, "M*odélisation de systèmes analogiques et mixtes, Introduction a VHDL-AMS*", Ecole Polytechnique Federale de Lausanne, Ete 2003

[VAL-94] A. Vladimirescu, "*The Spice Book*", John Wiley & Sons, 1994.

[VAN-03] A. Van, "*Mechatronic design*", Mechatronics, 2003.

[VAN-03] J. Van Amerongen, P. Breedveld, "*Modelling of physical systems for the design and control of mechatronic systems*", Annual Reviews in Control, vol 27, 2003.

[VDI-04] Standard VDI 2206, "*Design methodology for mechatronic systems*", 2004.

[VER-10] J. Verries, "*Approche pour la conception de systèmes aéronautiques innovants en vue d'optimiser l'architecture application au système portes passagers*", Thèse Universite Toulouse III, 2010.

[VER-12] F. Verdiere, A. Rezgui, S. Gaaloul, B. Delinchant, L. Gerbaud, F. Wurtz, X. Brunotte, "*Modelica models translation into Java components for optimization and DAE solving using automatic differentiation*", UKSIM 2012

[WAN-02] L. Wang, W. Shen, H. Xie, J. Neelamkavil, A. Pardasani, "*Collaborative conceptual design—state of the art and future trends*", Computer-Aided Design, Vol 34, 2002

[WEG-96] P. Wegner "*Interoperability*", ACM Computing Survey, 1996

[WET-09] M. Wetter "*A Modelica-based model library for building energy and control systems*". Building Simulation, 2009

[WIL-04] P. Wilson, J. Ross, A. Brrown, and A. Rushton, "*Multiple domain behavioral modelling using VHDL-AMS*", Circuits and Systems, vol. 5, 2004

[WU-00] B. Wu, R.E. White, "*An initialization subroutine for DAEs solvers : DAEIS*", Computers and Chemical Engineering 25 301–311 November 2000

[WUR-08] F. Wurtz, "*Conceptions de la conception pour le génie électrique : de l'approche « Objets –Savoirs – Méthodes – Outil » à l'approche « Systèmes – Connaissances – Compétences – Organisations »* " HDR, Institut National Polytechnique de Grenoble, 2008

[WUR-08a] F. Wurtz, "*Vers de nouvelles approches pour la capitalisation des modèles pour la simulation et l'optimisation : l'expérience du projet DIMOCODE*", NUMELEC, 2008

[ZEI-00] B. P. Zeigler, H. Praehofer, T. G. Kim "*Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*", Academic Press, 2000.

[ZIE-05] O. C. Zienkiewicz, R. L. Taylor, J. Zhu, "*The Finite Element Method: Its Basis and Fundamentals*", Butterworth-Heinemann; 2005.

Annexes

Annexe 1-1 :

Prototypage Virtuel Fonctionnel

1. Vers une conception basée sur le Prototypage Virtuel Fonctionnel

Nous rappelons que le développement des systèmes actuels met en jeu de plus en plus de relations d'interdisciplinarité entre métiers et entre domaines physiques et techniques. Mener à bien ce développement nécessite des moyens méthodologiques permettant d'identifier clairement les niveaux hiérarchiques, aussi bien dans la phase de définition que celle de vérification. Dans ce contexte, la méthodologie basée sur le Prototypage Virtuel Fonctionnel (PVF) représente une approche de conception avancée répondant à ces exigences [HER-06]. Ce genre d'approche doit être supporté par les outils de simulation système pour passer vers des outils de conception système offrant des environnements intégrés support du processus de conception. Cet enjeu représente aussi l'un des objectifs de projet MoCoSyMec. Chaque niveau doit supporter des modèles de description métiers permettant :

- d'analyser et comprendre le système et son environnement,
- de formaliser les problèmes et les besoins,
- de construire les architectures fonctionnelles et physiques,
- de prévoir et valider les comportements des systèmes,
- de capitaliser et réutiliser des composants, et de capitaliser la connaissance.

Nous rappelons aussi, que le langage VHDL-AMS respecte un grand nombre de ces objectifs méthodologiques, en renforçant toutes les étapes de conception par l'utilisation de modèles et de simulations de prototypes virtuels. Y. Hervé a montré que le langage VHDL-AMS est un bon candidat et peut-être actuellement le seul à gérer correctement cette méthode, en permettant de décrire tous les niveaux grâce à un langage commun [HER-06]. Ainsi, grâce à la simulation, l'approche PVF utilise un langage de communication commun (VHDL-AMS) pour formaliser de spécifications globales et les vérifier tout au long du cycle de conception. Cette méthodologie permet de formaliser, d'échanger et de réutiliser des concepts sous formes de modèles multi-niveaux d'un système multi-domaine.

2. Cycle du Prototypage Virtuel Fonctionnel

A chaque étape de conception, le Prototypage Virtuel Fonctionnel est un enrichissement du cycle en V, par des modèles comportementaux décrits à un niveau d'abstraction bien choisi. Cette approche privilégie la modélisation et la simulation à chaque étape de conception, jusqu'à la mise en œuvre du prototype réel [HER-06]. La figure 1 illustre cette approche en reprenant le « cycle en V » classique et en y apportant les modifications décrites. Ainsi, les différentes étapes (notées de 1 à 6) sont ajoutées tant au niveau de la descente du cycle (phase de conception), qu'à celui de la remontée (phase de validation). Le renforcement de toutes les

étapes a pour but d'éviter les erreurs éventuellement intégrées lors de la phase de conception, en essayant de les détecter le plus tôt possible dans la phase de validation.

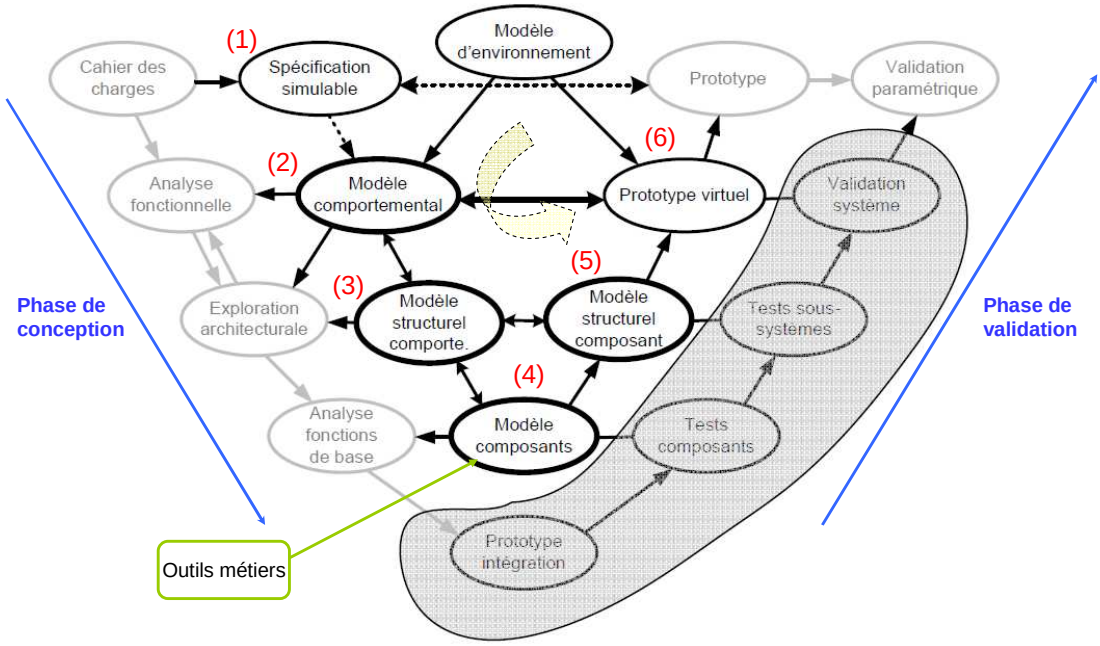


Figure 1 : Les étapes (de 1 à 6) du Prototypage Virtuel Fonctionnel [HER-06]

Nous allons maintenant détailler ces nouvelles étapes, qui vont être ajoutées à la phase de conception du cycle en V (1 à 4) et à celle de la phase de validation (5-6).

(1) Spécification simulable

Le cycle débute par la traduction de la définition du système en une description des caractéristiques simulables. Ceci revient à traduire les spécifications fonctionnelles, par exemple exprimées sous SysML[77], vers un langage de simulation comportementale comme VHDL-AMS[78].

(2) Modèle comportemental

Il s'agit de créer un modèle décrivant le comportement global du système. Ce modèle de haut niveau utilise les informations disponibles sur le système. Il doit retourner des performances compatibles avec les spécifications globales en tenant compte des contraintes de son environnement (échanges thermiques, effets mécaniques ou électromagnétiques, etc.). A ce stade, Les caractéristiques physiques des éventuels composants ne seront pas abordées.

(3) Modèle structurel comportemental.

Cette étape consiste à partitionner le système en plusieurs sous-modèles indépendants. Ce modèle permet d'analyser la structure du système et d'établir les différents composants ainsi que leurs interactions, indépendamment de leur mise en œuvre. La description des sous-modèles du modèle est en général de haut niveau.

[77] SysML : Systems Modeling Language
[78] SysML-Companion : http ://www.realtimeatwork.com/software/sysml-companion/

(4) Modèle composants.

Les modèles décrivant les composants au niveau physique seront établis en prenant en compte leurs caractéristiques intrinsèques et répondant aux spécifications propres à chacun d'eux. Avec ce modèle de bas niveau d'abstraction, nous pouvons simuler le comportement physique détaillé de chaque composant.

(5) Modèle structurel des composants.

Dans ce modèle, les sous-modèles du « modèle structurel comportemental » de la phase (3) de conception seront remplacés par les modèles des composants établis dans la phase (4). Ce niveau de modèle nous donne non seulement une vue globale du comportement du système, mais aussi la fonction de chaque composant, ainsi que les interactions entre eux.

(6) Prototype Virtuel.

Avec le « modèle structurel des composants » établi et validé dans l'étape précédente, nous pouvons formaliser et construire un « prototype virtuel ». En fonction de l'objectif recherché par le concepteur, les modèles des différents niveaux d'abstractions sont interchangeables dans le système complet (avec son environnement)

A chacune de ces étapes, il y a une comparaison entre les modèles au niveau physique (souvent fins) et les modèles au niveau système (plutôt globaux ou grossiers).

Il faut remarquer que cette méthodologie ne remplace en aucun cas l'approche et les connaissances métiers. C'est pourquoi un onglet « outils métier » apparaît dans la pointe basse du cycle, en relation avec la modélisation des composants au niveau physique. Cette modélisation doit s'appuyer sur les compétences et les outils métiers déjà en place dans tous les domaines impliqués, surtout quand un modèle prédictif très précis est requis.

Annexe 1-2 :

Les langages de modélisation mixte multi-domaines

Pour réaliser une conception performante en appliquant une méthodologie à base du prototypage virtuel fonctionnel, un langage de modélisation descriptive et multi-domaines et un outil de simulation associé doivent être choisis pour répondre aux différents besoins présentés dans le chapitre 1.

Parmi les langages les plus utilisés, nous pouvons citer VHDL-AMS[79], Verilog-AMS[80], Modelica[81], Simscape[82] et MAST[83]. L'avantage de ces langages est que l'écriture du modèle est directe et la difficulté de sa résolution est transparente pour le programmeur, elle est assurée par les simulateurs. A part les langages MAST et Simscape qui sont propriétaires, les autres langages sont beaucoup plus accessibles. Nous allons présenter ces langages.

1. Les langages de modélisation

(1) Le langage VHDL-AMS

Depuis sa normalisation en 1999, le langage VHDL-AMS (Very high speed integrated circuits Hardware Description Language – Analog- Mixed Signal) a accéléré sa conquête dans le secteur de la modélisation multi-domaine [HER-02] [DES-04]. Cette norme assure une description claire et structurée d'un système dynamique complexe, permettant une haute modularité et supportant la généricité.

La normalisation du langage présente l'avantage de proposer un support commun indépendant des fournisseurs de logiciels et de technologies. De ce fait, les collaborateurs d'un même projet travaillant avec différents simulateurs VHDL-AMS peuvent, théoriquement, facilement échanger et réutiliser leurs modèles.

La capacité native du langage pour la modélisation et la simulation des systèmes multi-domaines facilite la communication et la coopération entre partenaires de différents domaines scientifiques. Cette souplesse d'emploi permet aux concepteurs (électronicien, mécanicien,..) de modéliser les parties d'un dispositif qui le concerne directement, sans problème de dialogue avec les autres parties.

Le langage VHDL-AMS permet aussi de supporter la conception à plusieurs niveaux, ce qui est un avantage surtout lors des différentes étapes de la conception d'un dispositif. Il

[79] Very high speed integrated circuits Hardware Description Language – Analog- Mixed Signal
http ://www.eda.org/vhdl-ams/
[80] http ://www.eda.org/verilog-ams/
[81] https ://modelica.org/
[82] http ://www.mathworks.fr/products/simscape/
[83] http ://www.synopsys.com/Systems/Saber/Pages/MAST.aspx

supporte les modèles de simulations mixtes avec des modèles multi-abstractions. Il permet non seulement d'effectuer des modèles à haut niveau d'abstraction (comportemental) afin de valider rapidement la fonctionnalité du système, mais aussi de simuler des modèles structurels de bas niveaux avec des détails techniques plus au moins précis (physiques) [HER-02].

Cependant, un de ses principaux inconvénients est que pour l'instant le langage VHDL-AMS ne propose pas la description des systèmes sous forme d'équations aux dérives partielles. Il existe cependant de nombreuses recherches abordant ce problème et proposant des solutions. Par exemple, Nikitin proposent d'appliquer une discrétisation spatiale pour contourner cette limite [NIK-07]. C'est aussi le cas pour les intégrations spatiales.

(2) Le langage Modelica

Modelica est un langage modélisation orienté objet, unifié et normalisé, qui commence à s'imposer dans la modélisation de systèmes multi-physiques. Son développement et sa promotion sont organisés par l'association Modelica depuis 1997.

C'est un langage orienté objet qui permet de modéliser des systèmes complexes multidisciplinaires (thermique, mécanique, hydraulique, électronique, etc.).

Contrairement aux autres langages orientés objet (C++, JAVA), le modèle sous Modelica est une description mathématique déclarative, où l'analyse et la résolution sont supportées [KOS-06] [MIC-09] et dédiés à la simulation dynamique des systèmes. Modelica supporte différentes modélisations : la modélisation acausale (non orientée), les équations algébriques et différentielles, les machines d'état et les blocs diagrammes.

Tout comme VHDL-AMS, Modelica ne permet pas d'utiliser des équations aux dérivées partielles. Il existe cependant de recherches abordant ce problème et proposant des solutions [FRI-04] [SAL-00] [LI-09] [WET-09].

Les avantages souvent attribués à ce langage sont :

- un code source partagé,
- la facilité d'implémentation de modèles guidée par la physique,
- un langage basé les équations non orientées,
- la construction de modèles hiérarchisés acausaux,
- la réutilisation de modèles.

Modelica bénéficie d'une attention particulière de la part du monde académique et apparaît aujourd'hui comme suffisamment mature pour diverses applications industrielles. Dans le monde Modelica, on trouve principalement les outils suivants Dymola (Dynasim/Dassault System), AMESim (LMS Imagine), Scicos (INRIA), open modelica, jmodelica, etc. Modelica est largement utilisé dans le cadre des simulations dans le contexte des bâtiments [FRI-98] [FEL-02] [FRI-04] [GAA-11].

(3) MATLAB/Simulink et Simscape

MATLAB est un produit de MathWorks Inc, largement utilisés, pour le développement des techniques de calcul et de la conception basée sur les modèles. MATLAB est destiné essentiellement à réaliser des calculs mathématiques avancées, à partir de bibliothèques existantes ou en développement de nouveaux programmes utilisateurs. Simulink est une boîte

à outils pour la modélisation et la simulation des systèmes dynamiques. Cet environnement graphique est une plateforme de simulation multi-domaines, basée sur les flots de signaux (fonctions de transfert orientées). Il contient un ensemble de librairies qui peuvent être adaptées et/ou enrichies aux besoins du concepteur. Simulink offre aussi la possibilité de traduire les modèles des composants Simulink : des S-functions. Ils sont un ensemble de fichiers en différents langages (dont le langage propre à Matlab, C, ...). Le code généré peut être réutilisé par la suite dans le cadre de la co-simulation multi-langages. L'utilisation des S-functions permet de décrire ses propres modèles de façon plus efficaces que par l'assemblage de blocs de base (additionneurs, gains, multiplieurs, intégrateurs, dérivateurs, etc.)

Mathworks propose aussi un ensemble d'outils de modélisation physique propres à chaque domaine (SimElectronics™, SimMechanics™, SimDriveline™, SimHydraulics® et SimPowerSystems™). Cependant les équations implicites (lois de Kirchhoff) et les aspects multidisciplinaires ne sont pas nativement mis en œuvre. Pour remédier à cela, MathWorks propose un nouveau langage multi-physique : Simscape. Ainsi, une plateforme a été ajouté à Simulink pour une modélisation des systèmes mécatroniques plus intuitive. Le langage ressemble fortement à Modelica et à VHDL-AMS mais reste propriétaire. Actuellement, il ne supporte que les modèles d'états continus.

(4) Le langage MAST

Le langage MAST (Modeling Analog System with Template) est un langage propriétaire développé en 1984 et complètement lié au simulateur SABER. Celui-ci a été développé à l'origine par la société AnalogyTM et appartient actuellement à la société SYNOPSYS. Depuis plus de vingt ans, en tant que précurseur des langages de modélisation, MAST a dominé le marché des langages de description de matériel à signaux et technologie mixtes. Il a été notamment utilisé dans de nombreuses applications automobiles et aéronautiques. Ce langage propose néanmoins des mécanismes de modélisation obsolètes et non adaptés, notamment pour la modélisation de systèmes numériques.

Le langage MAST est exclusif à l'outil SaberTM, et présente le défaut de ne pas être normalisé. De plus, il permet d'avoir au sein même du code du modèle des instructions pilotant directement le cœur du simulateur et de contrôler la convergence. La mise en commun de différents modèles ayant chacun un accès au cœur de simulation rend les projets d'une complexité extrême.

(5) Verilog-AMS

Le Verilog-AMS a été créé sous la tutelle d'Accellera (Organisation de normalisation EDA) afin de mettre en place les extensions analogiques mixtes du Verilog (IEEE-1364).

Il permet de faire la description comportementale des systèmes analogiques et mixtes. Tout comme le VHDL-AMS, le Verilog-AMS peut être applicable aux systèmes électriques et non électriques. Ce langage permet de faire des descriptions de systèmes, en utilisant des concepts tels que les nœuds, les quantités de branche, et les ports. Les signaux de type analogique et numérique peuvent être présents dans le même module. D'ailleurs, le langage Verilog-AMS fournit aussi la possibilité de modéliser des systèmes multi-physiques. Ce langage ne présente

pas les mêmes hauts niveaux d'abstraction que VHDL-AMS. La norme est assez floue sur certains points rendant ses implémentations incompatibles[84].

2. Comparaison des langages de modélisation multi-domaines

Le Tableau 2 rassemble et compare les principales caractéristiques des cinq langages de modélisation, donnant une synthèse enrichie des différentes études et comparaisons faites entre ces langages de modélisation [GUI-07] [ZHO-07] [IBR-09].

	VHDL AMS	VerilogAMS	Modelica	MAST	Simscape
Aspects du langage					
Norme	IEEE 1076.1-1999	Recommandation	Non définie	Langage propriétaire	Non
Présentation	Extension du VHDL	Extension du Verilog	Langage orientée objet	Langage lié à l'outil Saber	Langage de Matlab
origine	Origine Ada	Sémantique proche du langage C	Langage orienté objet (java, c++)	proposé par Synopsys	Langage basé sur Matlab
Modularité	Vue externe, Vue interne Multi Architecture	Vue interne Interface externe Mono Architecture	Modèle et équations Mono Architecture	Mono bloc Mono Architecture	Modèle et équations Mono Architecture
Généricité	Paramètres Génériques Instruction generate	Paramètres Génériques	Paramètres Génériques	Paramètres Génériques	Paramètres Génériques
Gestion de Bibliothèques	Oui	Non	Oui	Non	Oui
Multi disciplinaires	Oui	Oui	Oui	Oui	Oui
Multi abstractions	Oui	Non	Oui	Oui	Non
Dépendance à un outil	Non	Non	Non	Oui	Oui
Expression de la Structure					
Structure	Instanciation de composants de manière hiérarchique				
Conservatif / flot de signal	Oui/Oui	Non/oui	Oui/Oui	Oui/Oui	Oui/Oui
Gestion de discontinuité	Oui ; Instruction Break	Oui	Oui	Oui Control section	Oui
Expression du Comportement					
Accès à la physique DAE/ODE	Forme explicite et implicite équations d'ordre N>=1	Formes explicite et implicite équations d'ordre 1	Formes explicite et implicite équations d'ordre 1	Forme explicite et implicite équations d'ordre 1	Forme explicite et implicite équations d'ordre 1

Tableau 2 : Comparatif des différents langages de modélisation mixte

[84] http ://www.eda.org/twiki/bin/view.cgi/VerilogAMS

Les critères de comparaison choisis sont les suivantes :

- les aspects génériques du langage normalisé ou pas, la modularité des modèles, leurs généricité, la multidisciplinarité des systèmes modélisés (différents domaines), les niveaux d'abstraction modélisés et la dépendance à un outil,
- la structuration des modèles : la gestion des signaux continus et discrets, l'instanciation de composants, les types de modèles envisageables (conservatifs ou pas).

D'après cette étude des langages et leur comparaison, il en ressort que les mécanismes de modélisation de la norme VHDL-AMS (multi-domaine, multi-abstraction et multi-architecture) apparaissent les mieux adaptés à résoudre la problématique de la conception des systèmes mécatroniques et électromagnétiques. Ce langage de haut niveau permet d'écrire un cahier des charges abstrait et simulable. Une architecture de système peut être faite de manière comportementale, structurelle ou fonctionnelle en associant différents modèles. Il permet aussi de décrire les composants à très bas niveau en utilisant les équations différentielles issues des études physiques. Toutes ces raisons confortent le choix fait par le consortium MoCoSyMec de travailler avec le langage VHDL-AMS.

Le tableau 2 valide aussi notre choix, en présentant le positionnement du langage VHDL-AMS par rapport aux autres langages que les concepteurs peuvent être amenés à utiliser.

	Level/Domain	Solid mechanic	Point mechanic	Multiphysic e.g optic, thermal	Analog electronic	Digital electronic	Control	Software
Continuous & discrete time simulation	Specification level				VHDL-AMS			UML
	Behavioral level		Modelica	Modelica				SystemC
	Architectural level		ADAMS	Modelica	Spice	VHDL	MATLAB	
	Component level	CAD Mesh simulation						
CAD	Physical model	CAD/CFM (e.g.: Catia)			Electronic CAD tools (e.g.: PCB, IC masks)			
	Components examples	Motor CAD, transformer: electro-magnetic field CAD	Airbag sensor, CAR movement	Coil, Magnetic head, heat-pressure sensor	Antenna, Microphone, transistor, power amplifier	Processor, Digital IC, DSP	Finite state machine, signal filters	Embedeed software, Hardware-software co design
	Applications		Car, Computer Hard Disk					
					Mobile phone			
				Smart label, integrated sensors			Network	
			CAR servo-electronic suspension, ABS					
			Solid state gyroscope			Cockpit simulation		

Tableau 3 : Positionnement du langage VHDL-AMS par rapport aux autres langages [HER-08]

Annexe 1-3 :

Environnements de simulation supportant le langage VHDL-AMS

Étant donné que l'émergence du langage VHDL-AMS est relativement récente, les outils simulateurs qui le supportent ne sont pas nombreux. Les principaux outils de simulation sont les suivants : hAMSter, SIMPLORER™, Portunus, SMASH™, AdvanceMS™ et Saber™ de chez Synopsys. Actuellement, aucun d'entre eux n'implémente complètement la norme, mais chacun s'en approche plus ou moins. Dans premier temps, nous présentons une description brève des différents outils de modélisation connus ; et dans deuxième temps, nous ferons une comparaison entre eux.

1. Les outils de modélisation VHDL-AMS

(1) hAMSter et Simplorer (Ansoft R)

hAMSter est un compilateur/simulateur du langage VHDL-AMS librement disponible. Ce n'est pas un outil industriel. Son principal avantage est sa simplicité d'utilisation, il permet à l'utilisateur de prendre en main très rapidement les démarches essentielles du travail avec le VHDL-AMS. Cet outil a été absorbé par Ansoft, un leader dans le développement de logiciel pour la conception et la simulation électronique de haute technologie. Un des produits phares d'Ansoft est Simplorer™, dont le moteur VHDL-AMS a été construit sur hAMSter.

(2) SMASH (Dolphin-integration R)

SMASH™ est en ce moment le simulateur du langage VHDL-AMS le plus complet sur le marché. Il s'approche d'une couverture totale de la norme, assez loin devant les autres simulateurs. Dans sa version de base, il ne dispose pas d'éditeur graphique. Une capacité très intéressante de l'outil SMASH™ est sa capacité multi-langage. Il est compatible avec SPICE, Verilog-HDL, VHDL, ABCD (internal C-language), C et VHDL-AMS et peut être interfacé avec différents outils logiciels comme MATLAB/SIMULINK.

(3) Portunus (Adapted Solutions/CEDRAT)

C'est un logiciel de modélisation des systèmes mécatroniques. L'objectif du projet MoCoSyMec, dans lequel s'inscrit ce travail, est d'aider à faire de Portunus une référence technologique pour la simulation et conception des systèmes mécatroniques en se basant sur le langage VHDL-AMS

(4) ADVance-MS (MENTOR-GRAPHICS)

C'est un outil issu et conçu pour le monde de l'électronique intégrée.

2. Comparaison des outils de simulation supportant VHDL-AMS

Une étude comparative entre différents outils a été présentée dans [HAM-05]. Cette étude, bien que datant de 2005, reste une référence pour les caractéristiques des différents outils. Nous présentons les conclusions de cette étude comparative que nous avons actualisée en nous positionnant sur un état de l'art datant du début de nos travaux (2009). Par ailleurs, durant le développement des modèles dans le cadre de la thèse, trois simulateurs parmi ceux cités ont tous été utilisés : Portunus™, SMASH™ et Simplorer™. Une étude comparative des trois outils est résumée aussi dans le tableau ci-dessous. Les critères importants des concepteurs sont en effet pour ces outils : la couverture de la norme, une interface conviviale, une gestion efficace des bibliothèques de modèles, des simulations performantes en termes de précision et de temps de mise en œuvre rapide des modèles.

	Simplorer	**ADVanceMS**	**Portunus**	**SMASH**	**Saber_HDL**
Editeur	Ansoft R Ansys	MENTOR-GRAPHICS	CEDRAT Adapted Solution	Dolphin-integration	Synopsys
Plateforme	PC/Windows	UNIX/linux	Windows	UNIX/linux Windows	Windows
Simulateurs	Simec	ELDO ModeSim		Single Kernel	Newton Calaveras
Interface graphique	oui	non	Oui	non	Oui
Code VHDL-AMS	Création automatique de l'entête	Utilisation d'un éditeur externe	Utilisation d'un éditeur interne	rien	Utilisation d'un éditeur externe
Bibliothèques de composants	Très fournies en SML Eléments de base en VHDL-AMS	Eléments de base en VHDL-AMS	Très fournies en C Pas de bibliothèque VHDL AMS	Rien	- Très fournies en MAST - Eléments de base en VHDL-AMS
Débuggage	A la création du composant	Au chargement dans le simulateur	A la création du composant	Au chargement dans le simulateur.	Au chargement dans le simulateur.
Implémentation de la norme	Une grande partie de la norme	Pas complète	Pas complète	Support complet de la norme	Pas complète

Tableau 4 : Comparatif des différents outils supportant VHDLAMS

Selon le tableau 3, SMASH est en ce moment le simulateur supportant le langage VHDL-AMS le plus complet sur le marché. Il s'approche d'une couverture totale de la norme, assez loin devant les autres simulateurs. Portunus, logiciel de modélisation des systèmes mécatroniques, a été choisi par le projet MoCoSyMec en tant qu'outil de développement de base et il a subi énormément d'amélioration parallèlement à nos travaux de ces dernières années. Par conséquent, dans notre thèse, nous avons travaillé conjointement sur les trois outils (Portunus, Simplorer et SMASH), profitant aux mieux de leurs spécificités, notamment SMASH grâce à sa couverture de la norme, et Portunus et Simplorer grâce à leurs interfaces graphiques.

Annexe 2-1 :

Modélisation VHDL-AMS du déclencheur électromécanique

La modélisation VHDL-AMS du déclencheur reprend l'approche structurelle présentée au chapitre 1 qui décomposait le système en trois parties principales :

- une partie mécanique (déplacement),
- une partie alimentation électrique
- une partie représentant un circuit magnétique.

Chaque sous-partie est validée indépendamment pour ensuite être intégrée au système complet. Cela permet une validation plus efficace des modèles.

Ici, nous visons une étude comportementale dynamique du déclencheur. Ainsi, nous avons choisi de mettre en place un modèle reposant sur les réseaux de réluctances pour la partie magnétique. Le modèle dynamique du dispositif représente un couplage entre le système d'état modélisant le circuit électrique d'alimentation de la bobine et les équations mécaniques de mouvement du noyau mobile.

1. Modélisation du circuit magnétique du déclencheur dynamique

Pour modéliser le déclencheur, nous avons repris les travaux de M.Perrault [PER-98], qui propose un schéma basé sur la formulation par réseau de réluctance. Nous avons fait des hypothèses simplificatrices afin de réduire la complexité du modèle. On suppose que les reluctances du circuit magnétique sont négligeables (perméabilité infinie) devant les reluctances d'entrefer, de fuite et de l'aimant. Les réluctances de fuite sont définies et ajustées suite à une analyse des allures des lignes de champs et des tubes de flux sous Flux2D en magnétostatique (Figure 1).

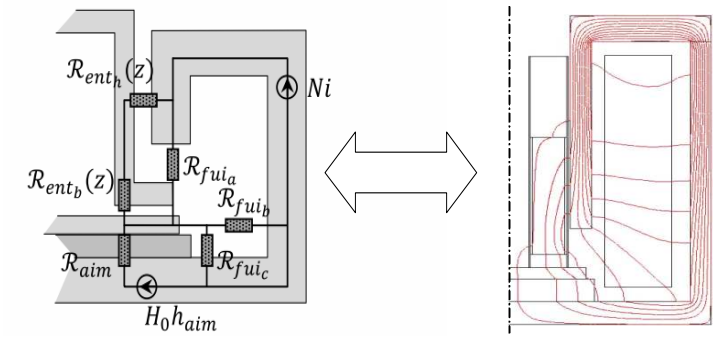


Figure 1 Schéma réluctant du déclencheur ; les tubes de flux magnétiques sont ajustés par simulation éléments finis.

Une fois le réseau de réluctance défini, nous obtenons les relations entre les grandeurs électromagnétiques nécessaires (flux, inductions et forces). Ceci se fait en appliquant les lois

des mailles. On en déduit le flux magnétique traversant le noyau mobile (Eq.2-1) ou φ_{noyau_aim} représente le flux créé par l'aimant et φ_{noyau_bob} celui créé par la bobine :

$$\varphi_{noyau}(z, i) = \varphi_{noyau\_aim}(z) - \varphi_{noyau\_bob}(z, i) \quad \text{(Eq.2 1)}$$

$$\varphi_{noyau\_bob}(z, i) = \frac{N.i}{R_{aim} + R_{ent}(z)} \quad \text{(Eq.2-2)}$$

$$\varphi_{noyau\_aim}(z) = \frac{R_{fui}}{R_{fui} + R_{ent}(z)} \cdot \frac{Br\_h_{aim}}{\mu_0 - \mu_{aim}} \cdot \frac{1}{R_{aim} + \frac{R_{ent}(z) \cdot R_{fui}}{R_{ent}(z) + R_{fui}}} \quad \text{(Eq.2-3)}$$

où h_{aim} est la hauteur de l'aimant.

La force magnétique F_{mag} peut être formulée par les Eq. 2-4 et 2-5, où F_{bob} représente la force créée par la bobine et F_{aim} celle de l'aimant :

$$F_{mag} = F_{bob} - F_{aim} \quad \text{(Eq. 2-4)}$$

$$F_{mag} = \frac{\varphi^2_{noyau}(z, i)}{\mu_0 . S_{noyau}} \quad \text{(Eq. 2-5)}$$

Le modèle VHDL-AMS du déclencheur magnétique contient donc l'ensemble des équations du modèle réluctant analytique liant la géométrie aux comportements et aux caractéristiques de ce dispositif. En d'autres termes, le modèle permet de calculer le flux magnétique traversant le noyau mobile ainsi que la force magnétique appliquée, dépendant de la position de noyau (*z)* et du courant dans la bobine (*i)* à l'instant *t*.

(1) Modélisation de l'alimentation électrique

Le circuit d'alimentation de la bobine est un circuit RLC commandé par un interrupteur. Ce circuit RLC série se modélise ; en VHDL-AMS par une équation différentielle d'ordre 2, qui peut se ramène à deux équations différentielles ordinaires. Le système d'état obtenu fait donc apparaître deux équations différentielles électriques (Eq. 2-6) où le vecteur d'état est représenté par les variables :

– *i* : le courant dans la bobine.

–*di/dt* : la dérive temporelle du courant, introduite pour réduire l'équation différentielle du second ordre à deux équations différentielles d'ordre 1 (ODE).

$$\begin{cases} \dot{i} = \frac{di}{dt} \\ \frac{d^2 i}{dt^2} = -\frac{R}{L(z)} . \frac{di}{dt} - \frac{i}{L(z).C} \end{cases} \quad \text{(Eq. 2-6)}$$

où L est l'inductance totale de la bobine qui dépend de la variation de l'entrefer et donc de la position du noyau mobile *z* à l'instant *t*. L'inductance de la bobine vérifie l'équation suivante :

$$L(z) = \frac{N^2}{R_{aim} + R_{ent\_b}(z) + R_{ent\_h}(z)} + 0{,}01702 \qquad \text{(Eq. 2-7)}$$

Les conditions initiales du système sont formulées dans Eq. 2-8 :

$$\begin{cases} i_0 = 0 \\ \dfrac{di}{dt} = \dfrac{E}{L(z_0)} \end{cases} \qquad \text{(Eq. 2-8)}$$

(2) Modélisation du mouvement du déclencheur

Comme nous l'avons présenté dans le chapitre 1, le déclencheur contient un ressort en compression, qui permet le mouvement du noyau. Après le déclenchement, ce mouvement s'effectue jusqu'à l'arrivée à la butée qui représente la limitation de la position (*Zmax*). Les caractéristiques du ressort sont modélisées en reprenant le modèle proposé par [GOW-03], et illustrés sur la figure 2.

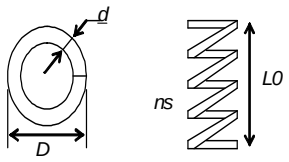


Figure 2 : Le ressort et ses paramètres

La force du ressort dépend de la position du noyau (z) et elle est donnée par l'équation analytique suivante

$$F_{ressort} = k.(L(z_0) - z) \qquad \text{(Eq. 2-9)}$$

où la raideur du ressort k dépend du diamètre du fil *d*, du diamètre extérieur *D*, du nombre de spires *ns* et du module de coulomb du matériau *G.* :

$$K = \frac{G.d^4}{8.n_s.D^3} \qquad \text{(Eq. 2-10)}$$

Le noyau d'un déclencheur électromagnétique est tenu en position basse par un effort résultant de la force magnétique créé par l'aimant et qui est supérieure à l'opposé de la force du ressort. Il en résulte une force résiduelle sur le noyau. Dans la position verticale, la plus contraignante, s'ajoute à la force du ressort le poids qu'il faut aussi compenser par la force de l'aimant.

$$F = F_{ressort} + F_{mag} + m.g \qquad \text{(Eq. 2-11)}$$

Avant le déclenchement, la force magnétique n'est due qu'à l'aimant. Au moment du déclenchement, la force magnétique est due à l'aimant et à la force créée par la bobine alimentée. Cette force est telle que la force du ressort va devenir prédominante et faire décoller le noyau. Dès que l'entrefer devient significatif, la force magnétique n'existe plus et le mouvement est essentiellement dû à la force du ressort.

De ce fait, les équations mécaniques définissant le mouvement du noyau mobile représente un système d'état faisant apparaître deux équations différentielles mécaniques (Eq. 4-12) ou l'état est représente par la position du noyau (*z)* et sa vitesse (*v)*. Afin d'éviter la simulation du déplacement du noyau mobile vers de bas, aspect qui n'est pas réalisable en réalité, nous formulons les conditions concernant le signe de la force résiduelle. Ceci introduit, dans la résolution du système d'état, une discontinuité à l'instant où le noyau mobile décolle.

$$\begin{cases} \dot{z} = \begin{cases} 0, \text{si } F < 0 \\ v, \text{si } F \geq 0 \end{cases} \\ \dot{v} = \begin{cases} 0, \text{si } F < 0 \\ \dfrac{F}{m}, \text{si } F \geq 0 \end{cases} \end{cases} \qquad \text{(Eq. 2-12)}$$

Les conditions initiales du système sont formulées comme suit :

$$\begin{cases} z_0 = z_{min} \\ v_0 = 0 \end{cases} \qquad \text{(Eq. 2-13)}$$

La modélisation met aussi en jeu les lois de Kirchhoff sur la conservation de l'énergie, qui sont implicites dans le langage VHDL-AMS. L'écriture des équations physiques se fait en VHDL-AMS de manière directe par le biais d'instructions simultanées.

(3) Intégration et résultats de simulation

La modélisation des différentes parties a été réalisée en VHDL-AMS à l'aide de l'outil Portunus. Le logiciel permet la saisie de modèles comportementaux avec un éditeur de modèle VHDL-AMS, mais aussi la saisie de schématique pour réaliser une architecture structurelle d'un système. Pour effectuer une validation du modèle complet du déclencheur, les différents modèles VHDL-AMS réalisés ont d'abord été validés de manière indépendante et ensuite intégrés au sein d'une schématique reprenant sa structure complète (Figure 3).

La figure 4 présente les différents résultats de simulation obtenus, à savoir les courbes de l'évolution du déplacement et de la vitesse du noyau mobile ; les courbes d'évolution de la force magnétique et de la force du ressort ; la courbe du courant électrique en sortie du circuit d'excitation de la bobine, ainsi que la variation de son inductance.

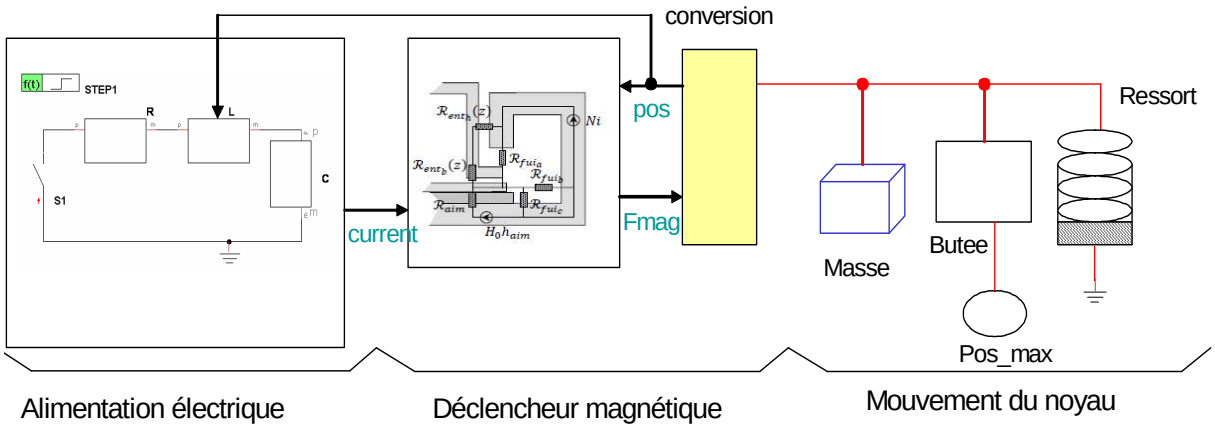


Figure 3 Modèle du déclencheur dynamique

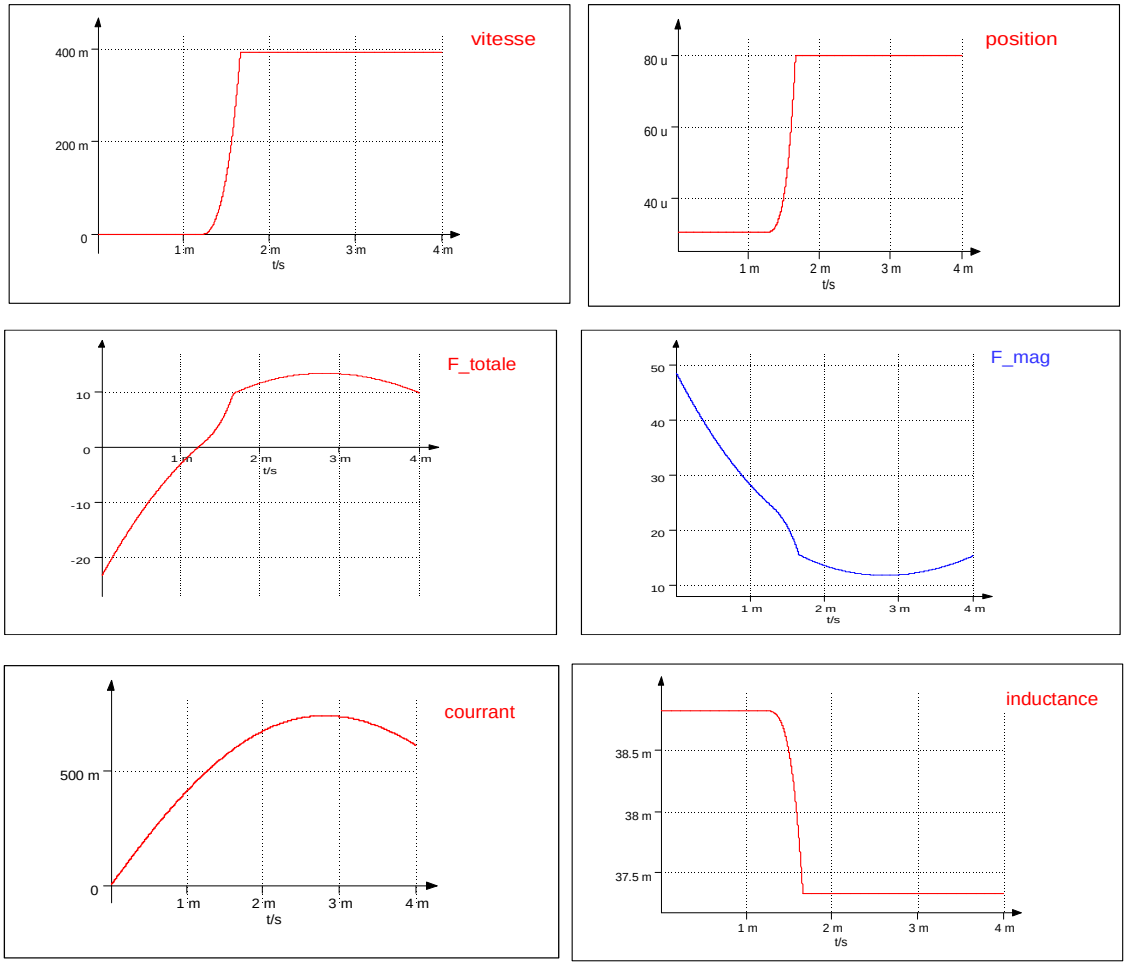


Figure 4 : Les résultats simulation du déclencheur en mouvement

2. Code source VHDL-AMS

```
LIBRARY IEEE ;
USE IEEE.ELECTRICAL_SYSTEMS.ALL ;
USE IEEE.MATH_REAL.ALL ;

entity déclencheur is
  generic (          r2  : real  := 12.2 ;
                     r3  : real  := 6.0 ;
                     r4  : real  := 5.5 ;
                     r5  : real  := 2.0 ;
                     r6  : real  := 4.1 ;
                     r7  : real  := 4.45 ;
                     r8  : real  := 5.85 ;
                     h1  : real  := 14.55 ;
                     h3  : real  := 20.2 ;
                     h5  : real  := 1.75 ;
                     h6  : real  := 0.9 ;
                     h8  : real  := 9.1 ;
                     C   : real  := 100.0e-6 ;
                     R   : real  := 5.46063083 ;
                     E   : real  := 18.0 ;
                     n   : real  := 540.0 ;
                     masse  : real  :=4.14998914e-3 ;
                     zmin   : real  := 0.03 ;
                     zmax  : real  :=  0.08
                     ) ;
-- variables d'états
  port (     quantity  z     : out real ;
             quantity  v     : out real ;
             quantity  i     : out real  ;
             quantity  di    : out real
              ) ;
end entity  ;

Architecture behav of déclencheur is
-- les constantes et les variables  intermédiaires
  constant Pi  : real  := 3.1415927 ;
  constant mu0  : real  := 4.0*1.0e-7*Pi  ;
  constant mu_aim  : real  := 1.1 ;
  constant B_aim  : real  := 0.95  ;
  constant L_zmin : real  :=0.03881084 ;
  quantity H_z , r_aim  : real ;
  quantity r_ent_b , r_ent_h , r_ent ,r_equ_a , r_equ_b , r_equ_c  : real ;
  quantity p_ent_a ,p_ent_b,p_ent_c  : real ;
  quantity r_fui_a ,r_fui_b ,r_fui_c ,r_fui  : real ;
  quantity fl_tot ,fl_fui, fl_noy, fl_noy_s  : real  ;
  quantity fl_fui_a,fl_fui_b,fl_fui_c,fl_noy_b,b_noy,b_cul_a  : real  ;
  quantity F_res , F_mag , F_tot , L_tot : real  ;
```

```
-- variables d'états intermédiaires
  quantity z_mm  : real  ;
  quantity acc    : real ;  -- accélétareur
  quantity dv     : real  ; -- dérivée de la vitesse v

Begin
  z_mm == z/0.001  ;
-- calcul de la hauteur H en mm
  H_z  == z_mm+h8+h6+h5+h1-h3 ;
-- reluctance de l'aimant  : constante -- à z
  r_aim == (1.0/(mu0*mu_aim))*(h5/(Pi*(r3*r3)))*(1.0/1.0e-3) ;
-- reluctance d'entrefer au niveau bas du noyau
  r_ent_b== (1.0*z_mm)/(mu0*Pi*(r6*r6-r5*r5)*(1.0e-3)) ;
-- reluctance d'entrefer au niveau de la circonférence du noyau
  p_ent_a==(mu0*2*Pi*H_z)/log(r7/r6)*1.0e-3 ;
  p_ent_b==3.3*mu0*((r7+r8)/2)*1.0e-3  ;
  p_ent_c==4.0*mu0*((r7-r5)-sqrt((r7-r6)*(r7-r5)))*log((r7-r5)/(r7-r6))*1.0e-3 ;
  r_ent_h== 1.0/(p_ent_a + p_ent_b + p_ent_c) ;
  r_ent==r_ent_b + r_ent_h ;
-- réluctances de fuites
  r_fui_a==(1.0/mu0)*(sqrt((h3-h1-h5-h6)*(h3-h1-h5-h6)+((r8+r7-r6-r4)/2)*((r8+r7-r6-
r4)/2)))/((Pi/2)*(r8*r8+r4*r4-r7*r7-r6*r6))*(1.0/1.0e-3) ;
  r_fui_b==(1.0/(mu0*2*Pi*(h6+h5)))* log(r2/r4)*(1.0/1.0e-3) ;
  r_fui_c==(1.0/(2*mu0*(r3+(h5/2))/(0.65*0.65)))*(1.0/1.0e-3) ;
  r_fui==1.0/((1.0/r_fui_a)+ (1.0/r_fui_b)+(1.0/r_fui_c))  ;
-- flux total traversant l'aimant
  fl_tot==((B_aim*h5*1.0e-3)/(mu0*mu_aim))/(r_aim+((r_ent*r_fui)/(r_ent+r_fui))) ;
-- flux total de fuite
  fl_fui==(r_ent*fl_tot)/(r_ent+r_fui) ;
-- flux traversant le noyau
  fl_noy==(r_fui*fl_tot)/(r_ent+r_fui) ;
--flux traversant la surface du bas du noyau
  fl_noy_s==0.9 * fl_noy ;
--Flux de fuite par la culasse à coté du noyau
  r_equ_a==r_fui_b * r_fui_c / r_fui_b + r_fui_c ;
  fl_fui_a==(r_equ_a *fl_fui)/(r_equ_a+r_fui_a) ;
--Flux de fuite par la culasse à travers la bobine
  r_equ_b==r_fui_a * r_fui_c / r_fui_a + r_fui_c ;
  fl_fui_b==(r_equ_b *fl_fui)/(r_equ_b+r_fui_b) ;
--Flux de fuite de l'aimant par l'intermédiare du corps
  r_equ_c==r_fui_b * r_fui_a / r_fui_b + r_fui_a ;
  fl_fui_c==(r_equ_c *fl_fui)/(r_equ_c+r_fui_c) ;
-- calcul du flux cree par la bobine dans le noyau
  fl_noy_b==n*i/(r_aim+r_ent) ;
-- calcul des inductions
-- 1. induction dans le noyau
  b_noy==fl_noy/(Pi*(r6*r6-r5*r5)*1.0e-6) ;
-- 2. induction dans la culasse au niveau du noyau
  b_cul_a==(fl_noy+fl_fui_a)/(Pi*(r8*r8-r7*r7)*1.0e-6) ;
-- calcul des forces
```

```
F_res==(-1)*1.24*(z+27.5)+59.1  ;
F_mag==((fl_noy_s - fl_noy_b)*(fl_noy_s - fl_noy_b))/(2*mu0*Pi*(r6*r6-r5*r5)*1.0e-6) ;
F_tot==F_res - F_mag  ;
-- calcul de l'inductance
L_tot==(n*n)/(r_aim+r_ent_b+r_ent_h)+0.01702 ;


-- conditions intiales   :
  break i  => 0.0 ;
  break di => E/L_zmin ;
  break z => zmin *0.001 ;
  break v  => 0.0  ;

-- Equations électriques
  di == i'dot ;
  di'dot == -(R*di+i/C)/(L_tot) ;

-- Equations mécaniques
  z'dot == dv ;
  v'dot == acc ;
  -- conditions de calcul de l'acc et de v :
  if (F_tot'above(0.0) ) use
        if (not (z_mm'above(zmax)) use
              acc == F_tot/masse ;
              dv  == v ;
        else
              acc == 0.0  ;
              dv  == 0.0  ;
        end use ;
  else
        acc == 0.0  ;
        dv == 0.0  ;
  END USE ;
end ;
```

Annexe 4-1 :

Fonctionnement de RelucTool

La génération automatique d'un modèle analytique, depuis RelucTool, se divise en différentes étapes principales [DUP-06] :

- mise en équation,
- ordonnancement pour résolution,
- dérivation,
- génération de code (composant ICAr).

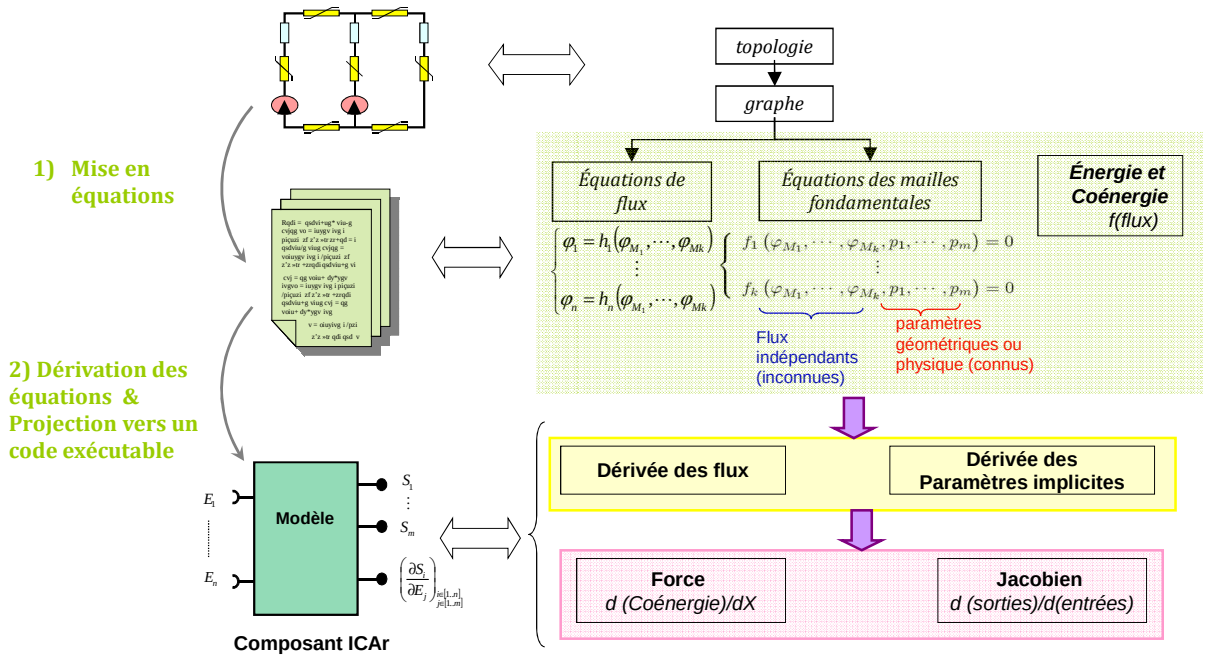


Figure 1 : Coeur de calcul du composant réluctant pour un modèle statique

(1) Traduction du schéma réluctant en un système d'équations

Dans RelucTool, les éléments de base utilisés pour construire le réseau de réluctances sont fournis dans une bibliothèque pouvant facilement être mise à jour. Les données sont décrites à l'aide du langage XML, qui permet de faciliter la structuration des données [DUP-06].

Lorsque le concepteur décrit la topologie de son réseau, une représentation objet du graphe correspondant au circuit associé est construite. Cette représentation graphique est analysée afin d'en déduire les équations des mailles indépendantes nécessaire à la résolution du système. En parallèle, les modèles associés à chaque élément du circuit sont convertis en équations ou des fonctions dépendant des paramètres que le concepteur a spécifiés. A la fin, toutes les équations générées constituent un système implicite (en raison des expressions des

réluctances qui dépendent des flux). Les calculs de l'énergie et de la coénergie associées à chaque élément (réluctance, source) sont ensuite générés. L'énergie, respectivement et la coénergie, du système correspond à la somme des énergies, respectivement des co-énergies, de tous les éléments. A ce système d'équations se rajoutent les équations complémentaires spécifiées par le concepteur. Les différents flux seront donc calculés à partir des résultats de la résolution du système implicite.

(2) Ordonnancement des équations en une séquence de résolution

A ce stade, toutes les équations du modèle sont analysées afin d'en extraire les relations de dépendance entre les paramètres du modèle. A l'aide d'une matrice d'occurrence [ALL-03], les entrées et les sorties du modèle peuvent être séparées et les équations peuvent être organisés dans une séquence de résolution. Ceci est structuré dans un fichier XML.

(3) Dérivation des équations pour le jacobien et la force magnétique et génération d'un composant logiciel ICAr

Le modèle analytique issu de ReluctTool est capable de fournir le jacobien, en vue de son dimensionnement par optimisation. Ainsi, la dérivation du flux est une étape importante pour obtenir le jacobien (dérivées formelles des sorties du modèle en fonction de ses entrées) ainsi que le calcul automatique de la force (dérivation de la coénergie du système). Cette dérivation s'effectue grâce au théorème des fonctions implicites [CIR-90] [DUP-06].

Pour toutes les dérivées des variables non implicites, ReluctTool utilise une technique de dérivation automatique des expressions analytiques, en se basant sur l'outil RAMA[85] dédié au calcul formel par application de règles génériques et développé au G2Elab [ALL-03] [FIS-03]. À la fin de ce processus, le modèle est converti en un langage de programmation portable (Java), et compilé.

[85]RAMA est l'acronyme de Rule Applicator for Mathematical Analysis est il représente un outil dédié à l'analyse des expressions mathématiques et au calcul formel des dérives.

Annexe 4-2 :

Fichier XSD pour la représentation de composition d'un circuit dans RelucTool

```
<?xml version="1.0" encoding="UTF-8"?>
<xs :schema xmlns :xs="http ://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

 <xs :element name="magneticmodel">
  <xs :complexType>
   <xs :sequence>
                <xs :element maxOccurs="unbounded" ref="parameter"/>
    <xs :element maxOccurs="unbounded" ref="input"/>
                <xs :element maxOccurs="unbounded" ref="output"/>
                <xs :element maxOccurs="unbounded" ref="port"/>
                <xs :element maxOccurs="unbounded" ref="component"/>
                <xs :element maxOccurs="unbounded" ref="link"/>
   </xs :sequence>
   <xs :attribute name="title" use="required" type="xs :NCName"/>
  </xs :complexType>
 </xs :element>

 <xs :element name="link">
  <xs :complexType>
   <xs :sequence>
    <xs :element ref="tail"/>
    <xs :element ref="head"/>
   </xs :sequence>
   <xs :attribute name="hashcode" use="required" type="xs :integer"/>
  </xs :complexType>
 </xs :element>
 <xs :element name="tail">
  <xs :complexType>
   <xs :attribute name="hashcode" use="required" type="xs :integer"/>
  </xs :complexType>
 </xs :element>
 <xs :element name="head">
  <xs :complexType>
   <xs :attribute name="hashcode" use="required" type="xs :integer"/>
  </xs :complexType>
 </xs :element>

 <xs :element name="component">
  <xs :complexType>
```

```
    <xs :sequence>
      <xs :element ref="model"/>
    </xs :sequence>
    <xs :attribute name="name" use="required" type="xs :NCName"/>
    <xs :attribute name="referencename" use="required" type="xs :NCName"/>
    <xs :attribute name="type" use="required" type="xs :NCName"/>
        <xs :element maxOccurs="unbounded" ref="parameters"/>
    <xs :element maxOccurs="unbounded" ref="EntreeSoties"/>
        <xs :element maxOccurs="unbounded" ref="port"/>
  </xs :complexType>
</xs :element>

<xs :element name="model">
  <xs :complexType>
    <xs :choice>
      <xs :element ref="output"/>
      <xs :element ref="param"/>
      <xs :element ref="reluctance"/>
                <xs :element ref="source"/>
    </xs :choice>
  </xs :complexType>
</xs :element>

<xs :element name="port">
  <xs :complexType>
    <xs :attribute name="anchor" use="required" type="xs :integer"/>
    <xs :attribute name="edgeendpriotity" use="required" type="xs :integer"/>
    <xs :attribute name="hashcode" use="required" type="xs :integer"/>
    <xs :attribute name="id" use="required" type="xs :integer"/>
  </xs :complexType>
</xs :element>
<xs :element name="parameter">
  <xs :complexType>
    <xs :sequence>
      <xs :attribute name="name" use="required"/>
                <xs :attribute name="value" use="required"/>
    </xs :sequence>
  </xs :complexType>
</xs :element>

<xs :element name="equation">
  <xs :complexType>
    <xs :attribute name="definition" use="required"/>
  </xs :complexType>
</xs :element>

<xs :element name="reluctance">
  <xs :complexType>
    <xs :attribute name="fluxname" use="required" type="xs :NCName"/>
    <xs :attribute name="name" use="required" type="xs :NCName"/>
```

```
      <xs :attribute name="value" use="required"/>
    </xs :complexType>
  </xs :element>
  <xs :element name="output">
    <xs :complexType>
      <xs :attribute name="name" use="required" type="xs :NCName"/>
      <xs :attribute name="value" use="required"/>
    </xs :complexType>
  </xs :element>
</xs :schema>
```

Annexe 4-3 :

Fichier XSD pour la représentation des équations d'un circuit dans RelucTool

```
<?xml version="1.0"?>
<xs :schema id="NewDataSet" xmlns="" xmlns :xs="http ://www.w3.org/2001/XMLSchema">

  <xs :element name="model">
    <xs :complexType>
      <xs :sequence>
        <xs :element name="parameterlist" minOccurs="0" maxOccurs="unbounded">
          <xs :complexType>
            <xs :sequence>
              <xs :element name="constant" minOccurs="0" maxOccurs="unbounded">
                <xs :complexType>
                  <xs :attribute name="definition" type="xs :string" />
                  <xs :attribute name="name" type="xs :string" />
                </xs :complexType>
              </xs :element>
              <xs :element name="input" minOccurs="0" maxOccurs="unbounded">
                <xs :complexType>
                  <xs :attribute name="format" type="xs :string" />
                  <xs :attribute name="name" type="xs :string" />
                  <xs :attribute name="precision" type="xs :string" />
                  <xs :attribute name="unit" type="xs :string" />
                </xs :complexType>
              </xs :element>
              <xs :element name="output" minOccurs="0" maxOccurs="unbounded">
                <xs :complexType>
                  <xs :attribute name="dependon" type="xs :string" />
                  <xs :attribute name="name" type="xs :string" />
                  <xs :attribute name="format" type="xs :string" />
                  <xs :attribute name="precision" type="xs :string" />
                  <xs :attribute name="unit" type="xs :string" />
                </xs :complexType>
              </xs :element>
              <xs :element name="internvariable" minOccurs="0" maxOccurs="unbounded">
                <xs :complexType>
                  <xs :attribute name="dependon" type="xs :string" />
                  <xs :attribute name="name" type="xs :string" />
                </xs :complexType>
              </xs :element>
              <xs :element name="positionvariable" minOccurs="0" maxOccurs="unbounded">
```

```
        <xs :complexType>
          <xs :attribute name="name" type="xs :string" />
          <xs :attribute name="unit" type="xs :string" />
        </xs :complexType>
      </xs :element>
      <xs :element name="forcevariable" minOccurs="0" maxOccurs="unbounded">
        <xs :complexType>
          <xs :attribute name="name" type="xs :string" />
          <xs :attribute name="unit" type="xs :string" />
        </xs :complexType>
      </xs :element>
    </xs :sequence>
  </xs :complexType>
 </xs :element>
 <xs :element name="calculusblock" minOccurs="0" maxOccurs="unbounded">
  <xs :complexType>
    <xs :sequence>
      <xs :element name="linearisation" type="xs :string" minOccurs="0" msdata :Ordinal="0" />
      <xs :element name="implicitparam" minOccurs="0" maxOccurs="unbounded">
        <xs :complexType>
          <xs :attribute name="name" type="xs :string" />
        </xs :complexType>
      </xs :element>
      <xs :element name="function" minOccurs="0" maxOccurs="unbounded">
        <xs :complexType>
          <xs :attribute name="definition" type="xs :string" />
          <xs :attribute name="dependon" type="xs :string" />
          <xs :attribute name="name" type="xs :string" />
        </xs :complexType>
      </xs :element>
      <xs :element name="equation" minOccurs="0" maxOccurs="unbounded">
        <xs :complexType>
          <xs :attribute name="definition" type="xs :string" />
          <xs :attribute name="priority" type="xs :string" />
          <xs :attribute name="intern" type="xs :string" />
        </xs :complexType>
      </xs :element>
    </xs :sequence>
    <xs :attribute name="priority" type="xs :string" />
    <xs :attribute name="type" type="xs :string" />
  </xs :complexType>
 </xs :element>
 <xs :element name="Dfunction" minOccurs="0" maxOccurs="unbounded">
  <xs :complexType>
    <xs :attribute name="definition" type="xs :string" />
    <xs :attribute name="name" type="xs :string" />
  </xs :complexType>
 </xs :element>
</xs :sequence>
<xs :attribute name="statut" type="xs :string" />
```

```
    </xs :complexType>
  </xs :element>
  <xs :element name="NewDataSet" msdata :IsDataSet="true" msdata :UseCurrentLocale="true">
    <xs :complexType>
      <xs :choice minOccurs="0" maxOccurs="unbounded">
        <xs :element ref="model" />
      </xs :choice>
    </xs :complexType>
  </xs :element>
</xs                                                                                    :schema>
```

Interopérabilité de modèles dans le cycle de conception des systèmes électromagnétiques via des supports complémentaires : langage VHDL-AMS et composants logiciels ICAr

Résumé

Cette thèse aborde les formalismes pour la modélisation multi-physique en support au cycle en V de conception. Ce travail a été réalisé dans le cadre du projet ANR–MoCoSyMec, selon la méthodologie du prototypage virtuel fonctionnel (PVF) et illustré sur des systèmes électromagnétiques.

Nous nous sommes principalement intéressés au langage VHDL-AMS, en tant que support aux différents niveaux de modélisation apparaissant dans le cycle en V de conception. Cela nous a conduits à traiter la portabilité et l'interopérabilité en VHDL-AMS de diverses méthodes et outils de modélisation. Nous avons proposé et validé, via le formalisme des composants logiciels ICAr, des solutions aux limites de l'utilisation de VHDL-AMS pour modéliser certains phénomènes physiques reposants sur des calculs numériques.

Nous avons étendu la norme ICAr pour supporter des modèles dynamiques décrits par des équations différentielles algébriques (DAE) ; et pour des besoins de co-simulation, nous pouvons également y associer un solveur. Ces développements sont désormais capitalisés dans le framework CADES.

Enfin, nous avons proposé une architecture pour le portage de modèles d'un formalisme à un autre. Elle a été définie et mise en œuvre plus particulièrement pour des modèles magnétiques réluctants (Reluctool) et des MEMS magnétiques (MacMMems) vers le VHDL-AMS.

Ces formalismes et méthodologies sont mis en œuvre autour du PVF d'un contacteur électromagnétique.

Mots clés : Prototypage virtuel fonctionnel, modélisation dynamique système, équations différentielles algébriques, VHDL-AMS, composants logiciels ICAr, interopérabilité de modèles et d'outils, portage de modèles, systèmes électromagnétiques

Abstract

This PhD report deals with modeling formalisms for multi-physical systems in the design V- cycle. This work was carried out within the French ANR-MoCoSyMec project, according to the methodology of functional virtual prototyping (PVF) and illustrated with electromagnetical systems.

The work focuses on the VHDL-AMS modeling language, as a support for several modeling levels appearing in the design V-cycle. In this work, the portability and interoperability problems have been studied, using VHDL-AMS, for various modeling methods and tools. Solutions have been proposed and validated for use limits of VHDL-AMS language, specifically for the modeling of some physical phenomena using numerical computations, through the software component formalism called ICAr.

The ICAr software component standard has been extended to support dynamic models described through differential algebraic equations (DAE). It has also been extended for co-simulation purposes in which a solver is associated to the dynamic model inside the ICAr component. These developed solutions are now available in the framework CADES.

Finally, architecture has been proposed for the transforming of models from a professional formalism into another, specifically into VHDL-AMS. It has been designed and implemented for reluctant magnetic models (RelucTool) and magnetic MEMS (MacMMems).

These formalisms and methodologies are implemented around the functional virtual prototyping (PVF) of an electromagnetic contactor.

Keywords: Functional virtual prototyping, dynamic modeling system, differential algebraic equations, VHDL-AMS, ICAr software components, model and software tool interoperability, model transforming, electromagnetical systems.