Un Environnement à base de Composants, Intégrant le Concepteur et ses Outils, pour de Nouvelles Méthodes de CAO.

Benoit Delinchant

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
/_/_/_/_/_/_/_/_/_/_/_/

THESE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : « Génie électrique »

Préparée au sein du **Laboratoire d'Electrotechnique de Grenoble**

dans le cadre de l'Ecole Doctorale
« Electronique, Electrotechnique, Automatique, Télécommunication, Signal **»**

présentée et soutenue publiquement par

Benoit DELINCHANT

Ingénieur ENSIEG

Le 10 juillet 2003

Un Environnement à base de Composants, Intégrant le Concepteur et ses Outils, pour de Nouvelles Méthodes de CAO

Directeur de thèse : Jean BIGEON
Encadrant : Frédéric WURTZ

Monsieur	Gilbert REYNE	Président
Messieurs	Laurent NICOLAS	Rapporteur
	Yves PERRIARD	Rapporteur
Messieurs	Jean BIGEON	Directeur de thèse
	Frédéric WURTZ	Encadrant
	Jean-François BOUJUT	Examinateur

Remerciements

Je souhaite remercier Messieurs Jean-Pierre Rognon et Yves Brunet qui ont, durant mes trois années de thèses, pris la direction du Laboratoire d'Electrotechnique de Grenoble, et m'ont permis d'effectuer ses travaux dans les meilleures conditions.

Je remercie également Jean Bigeon qui, alors directeur de l'équipe Conception et Diagnostic Intégrés (*CDI*), m'a initié à la problématique de conception et m'a fait découvrir qu'il existait des métiers passionnants de recherche sur de telles problématiques.

Je remercie d'une manière commune Laurent Gerbaud et Frédéric Wurtz, membres piliers de l'équipe CDI, au contact desquels j'ai pu trouver ma vocation. Je les remercie spécialement pour leurs conseils et le soutien qu'ils m'apportent dans ma quête d'un poste d'enseignement et de recherche.

Je dois m'attarder un peu plus longtemps sur la gratitude que j'ai envers Fred qui a été mon encadrant de thèse et qui a su assurer cette tâche de la meilleure manière qu'il soit. En effet, il est parvenu à me laisser libre de toutes mes actions et réflexions, me permettant un épanouissement total dans ma recherche, tout en maintenant le fil conducteur de ses idées afin de garantir les résultats de ces travaux. Je tiens à exprimer également ici toute la sympathie que j'ai pour lui et le plaisir que j'ai à travailler avec lui, qui s'explique peut être par nos origines lorraines communes.

Je remercie également deux autres membres permanents de l'équipe, Jaime Fandino, pour sa bonne humeur et sa philosophie de vie, ainsi qu'Alain Bolopion, pour sa disponibilité et sa conscience professionnelle mais par-dessus tout pour son humour.

Mes camarades thésards occupent une place majeure dans ces remerciements car s'est en leur compagnie que ces trois années ont pu être réussies. Je commencerais par mon condisciple Nuts (*Loig Alain*) avec qui j'ai partagé une longue et agréable route, laquelle, malgré une bifurcation inévitable, gardera je l'espère quelques chemins de traverses. Tous les anciens thésards (*les vieux*), que j'ai plaisir à revoir régulièrement, j'ai nommé Max. et Jean-Mich. (*Maxime Besacier et Jean Michel Guichon*), gros Ben (*Benoit Froidurot*), Pichu (*Armando Fonseca*), Bertrand (*B. Raison*) et enfin Eric (*E. Atienza*) qui, je dois me résigner à l'avouer, m'a influencé tant professionnellement que personnellement. Enfin les thésards

encore thésards (*les p'tits jeunes*), à commencer par David (*D. Magot*), qui, grâce à des collaborations fructueuses, m'a permis d'avancer plus rapidement dans mes travaux. Laule (*Laurent Albert*), que je remercie pour sa bonne humeur, son accent du sud-ouest et ses sujets de conversation toujours inattendus. Vince (*Vincent Fischer*) qui, malgré ses humeurs, sait dépanner des amis, mais aussi des motos sans toit. Et merci à Francky (*Franck Verdière*), le petit dernier, pour apporter un peu de mentalité nordique…

Si les personnes citées majoritairement pour le moment sont d'origine CDI, c'est qu'il existe dans cette équipe une convivialité qui m'a permis de passer 3 années très agréables en son sein.

Je n'oublie bien sûr pas les personnes du LEG, thésards ou chercheurs permanents, personnels administratifs ou techniques, avec qui j'ai pu avoir des contacts professionnels et personnels enrichissants. Je remercie chaleureusement toutes les personnes du LEG qui me soutiennent pour le métier que je souhaite exercer et j'apprécie la confiance qu'ils me portent.

J'embrasse enfin ma compagne Astrid, non pour la remercier d'avoir cherché à comprendre mon sujet de thèse, mais pour la compréhension qu'elle a de mon travail de chercheur avec les débordements sur ma vie personnelle que je n'arrive pas encore à bien gérer.

Le défaut fondamental des pères est de vouloir que leurs rejetons leur fassent honneur.

Bertrand Russell, mathématicien et philosophe.

à mes parents.

TABLE DES MATIÈRES

I. Problématique et Thèse

La conception en génie électrique est aujourd'hui outillée pour offrir au concepteur des solutions informatiques d'aide à la résolution de nombreux problèmes, allant des techniques de conception innovantes jusqu'à l'analyse de dispositifs complexes.

Pourtant, en dépit d'une richesse de la Conception Assistée par Ordinateur (CAO), il ne semble pas qu'un concepteur puisse facilement gérer ensemble les outils dont il a besoin au cours de son processus de conception. Or, la complexité des dispositifs à concevoir nécessite de créer une synergie à l'intérieur de cette diversité d'outils d'aide à la conception.

La thèse que nous avançons est d'offrir au concepteur un environnement de conception avec lequel il pourra facilement et intuitivement mettre en œuvre les outils et les modèles de calcul dont il a besoin pour travailler, au cours des processus de conception qu'il souhaitera traiter.

La solution que nous proposons s'appuie sur le concept de composants logiciels métiers avec lesquels le concepteur peut construire ses modèles et ses processus comme on assemble un circuit électronique.

Les bénéfices d'un tel environnement pour le concepteur sont multiples, mais peuvent être vus sous deux approches, une approche pragmatique et une approche méthodologique :

- **il pourra mettre à l'épreuve sa créativité par la mise en œuvre rapide de ses idées dans l'environnement**. Il pourra en effet créer et modifier, rapidement et sans programmation, des solutions informatiques (*modèles de calcul, enchaînement de tâches automatisées,* …) pour répondre à ses besoins de conception.
- **il pourra mettre en œuvre de nouvelles méthodologies de conception permettant d'améliorer ses solutions**. En effet, les capacités de l'environnement à fédérer dynamiquement un ensemble d'outils, offriront des moyens qui n'existaient pas jusqu'alors pour mettre en œuvre des méthodologies nouvelles.

II. Plan du document

Ce rapport comporte 4 chapitres au travers desquels nous allons construire l'environnement qui apporte ces bénéfices aux concepteurs du génie électrique.

II.A. Chapitre 1 : Analyse de la CAO et Nature de la Conception

Nous commencerons tout d'abord par décrire notre vision de la CAO et les enjeux qu'elle doit atteindre. Nous détaillerons alors quelques raisons conduisant aux difficultés d'une CAO efficace, en insistant sur deux points particuliers que notre environnement cherchera à prendre en compte.

Le premier concerne la complexité en conception qui se matérialise, par exemple, par les couplages entre les différents constituants qui caractérisent nos dispositifs (*domaines disciplinaires, technologies de réalisation, environnement d'utilisation,* …), mais aussi par le besoin de faire intervenir le concepteur à des moments stratégiques de la conception. Le deuxième point concerne la nature parfaitement imprévisible et dynamique du processus de conception et des difficultés que cela engendre pour les outils de CAO.

II.B. Chapitre 2 : Les Environnements de Conception

Un tour d'horizon des différents environnements sera fait en premier lieu, puis nous orienterons notre solution vers un environnement d'intégration à base de composants.

C'est alors que sera décrit le contexte de travail à partir duquel ont pu être mis en place tous les concepts nécessaires à l'architecture que nous proposons. Nous présenterons ainsi un patron de travail qui s'appuie sur des générateurs et des utilisateurs de composants. Nous définirons également des concepts permettant de travailler avec ces patrons (*intégration, projection, composition,* …).

II.C. Chapitre 3 : Méthodologies de l'Environnement

Deux méthodologies seront développées dans ce chapitre. La première concerne un type particulier de générateur de composants qui intègre les outils du concepteur dans l'environnement. La seconde répond à des besoins de composition. Deux compositions particulières ont été développées, celle qui crée des modèles de calcul par agrégation ainsi que celle qui décrit des processus de conception par l'enchaînement de tâches.

II.D. Chapitre 4 : Applications

Dans un objectif de dimensionnent d'un transformateur triphasé de tensions, nous détaillons tout d'abord comment nous avons pu définir, par composition, un modèle analytique comportant différents sous–modèles (*électromagnétique, géométrique, de calcul de pertes et économique*). Nous verrons ensuite comment ce modèle a pu bénéficier, rapidement et sans programmation, d'une partie de calcul numérique réalisée par un logiciel éléments finis, pour composer un modèle mixte (*analytique et numérique*).

Nous montrerons ensuite comment a pu être mise en œuvre, toujours sans programmation, un processus de dimensionnement semi–automatisé s'appuyant successivement sur les deux niveaux de modélisation précédents.

Deux autres applications de l'environnement et de ses méthodologies seront traitées, une première dans le cadre de la conception d'un micro actionneur magnétique, et une seconde dans le cadre de la conception collaborative d'un déclencheur électromécanique.

II.E. Annexes :

Nous finirons par des annexes afin de compléter des notions traitées mais non détaillées telles que l'annexe décrivant le modèle analytique du transformateur. Nous détaillerons ensuite le processus d'intégration d'un calcul Flux2D® dans l'environnement. Des compléments du dimensionnement de l'application du chapitre IV seront alors fournis. Une dernière annexe sera enfin consacrée à une problématique technologique de pilotage de logiciels informatiques.

Chapitre 1

ANALYSE DE LA CAO ET NATURE DE LA CONCEPTION

Chapitre 1

ANALYSE DE LA CAO ET NATURE DE LA CONCEPTION

Le contexte général dans lequel s'inscrit ce travail est la Conception Assistée par Ordinateur (*CAO*). Nous ferons dans ce premier chapitre une description générale de la conception et de ses outils d'assistance en génie électrique.

Pour cela, nous présenterons un aperçu de la CAO au travers de ses outils et des méthodes sur lesquels ceux-ci s'appuient. Une présentation des réels enjeux de la conception sera alors faite et associée aux limites des outils actuels.

Afin de mieux cerner les difficultés que doit gérer la CAO, nous présenterons dans une deuxième partie la nature complexe, dynamique et imprévisible d'un processus de conception

PARTIE I : OUTILS ET ENJEUX

DE LA CAO EN GENIE ELECTRIQUE

Nous pouvons définir les enjeux de l'aide à la conception en génie électrique selon trois axes, celui des méthodes, celui des outils et celui des environnements. Chacun étant plus ou moins lié aux autres, selon que les outils cherchent à mettre en œuvre des méthodologies, ou que les environnements cherchent à mettre en œuvre à la fois les outils et les méthodes, tout au long d'un processus de conception.

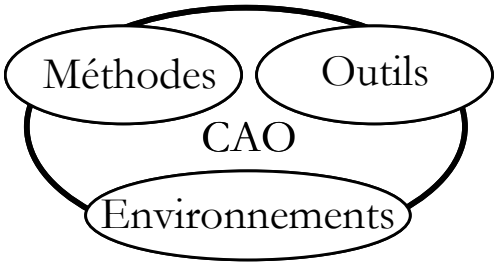


Figure 1 : La CAO peut être vue sous trois axes complémentaires, celui des méthodes, des outils et des environnements

Afin de repositionner convenablement les enjeux que posent les environnements de conception, nous allons présenter dans ce premier chapitre une vision des méthodes et des outils mis en œuvre dans la CAO et plus particulièrement dans le domaine du génie électrique.

I. Les outils de CAO

La conception est encore actuellement très mal prise en compte par nos systèmes informatiques. Le terme CAO peut prendre plusieurs formes selon le rôle que joue l'outil informatique qui lui est associé (*calculer, représenter ou fabriquer [IRM 96]*). Bien que la fonction première de tels outils soit d'aider les concepteurs, ceux-ci auront également des effets secondaires bénéfiques ou non, tels que ceux de *la prise en charge de la gestion des informations techniques, la coordination industrielle entre les différents acteurs, ou encore la standardisation des pratiques de conception [LAV 00].*

I.A. La place relative du concepteur dans la CAO

Derrière l'acronyme CAO, on imagine souvent un logiciel informatique. Or il s'agit d'un acte, celui de la conception, qui reçoit l'assistance de l'ordinateur. Il est certain que considérer la CAO par son simple instrument est totalement réducteur, il est nécessaire, comme nous le verrons tout au long de ce rapport, d'y inclure le concepteur.

> *La conception assistée par ordinateur est une technique dans laquelle on associe l'homme et la machine pour former une équipe capable de résoudre des problèmes de conception nouveaux exigeant des solutions originales, équipe qui accomplit plus facilement ces tâches que l'homme ou la machine seule.*
>
> *J.-L. Coulomb et J.-C. Sabonnadière [COU 85]*

Nous allons nous attacher à présenter quelques types d'outils de CAO utilisés dans le domaine du génie électrique, en les considérant par rapport aux méthodologies qu'ils mettent en œuvre et par rapport à la place occupée par le concepteur. Car en effet, l'utilisation d'un outil de CAO est souvent prescripteur d'une méthode de conception [LAV 00], imposant à la fois le rôle que doit jouer le concepteur et les méthodes qu'il doit suivre.

I.B. Quelques outils de la CAO

I.B.1. Outils d'aide à la modélisation, permettant de créer des modèles analytiques

Une première catégorie d'outils permet au concepteur de travailler comme il le ferait sans CAO, ne transcrivant sur informatique que ce qu'il faisait sur papier. Le concepteur doit en effet pouvoir développer des modèles de manière analytique par le biais d'hypothèses simplificatrices, d'expressions des lois comportementales et de relations empiriques.

Ces modèles ont pu trouver un support grâce à des outils génériques de traitement mathématique (*MatLab, MathCAD, Maple, TKSolver…*) permettant, par exemple, de résoudre les systèmes d'équations qui les décrivent. Avec ces outils mathématiques, de nombreux autres sont utilisés à des fins de modélisation, tels que des outils d'acquisition de données, des tableurs, des bases de données, etc.

I.B.2. Outils d'analyse, permettant de simuler et d'analyser le comportement virtuel du système

La complexité des phénomènes physiques mis en jeu dans les dispositifs à concevoir a conduit le domaine du génie électrique à s'outiller principalement en instruments de calcul. Des solutions diverses sont alors apparues permettant de simuler et d'analyser les dispositifs modélisés de manière intuitive pour le concepteur (*assemblage de composants prédéfinis, description structurale et physique*).

Le génie électrique s'est instrumenté, par exemple, d'outils de CAO qui offrent de résoudre les équations différentielles partielles à partir de méthodes numériques (*méthode des éléments finis*[1]*, méthode des éléments de frontières*[2]). Ces solutions ont permis de réduire considérablement le coût de modélisation et de calcul, grâce à une définition des dispositifs électromagnétiques par leurs géométries, propriétés physiques et domaine de calcul.

Cette seule phase de simulation n'est pas suffisante dans un processus de conception, et une phase d'analyse vient idéalement compléter les résultats de simulation. Cette dernière à su bénéficier des possibilités graphiques des ordinateurs, offrant des aides à l'analyse (*post-processing*) efficaces car proches des concepts que sait manipuler le concepteur.

I.B.3. Outils d'aide à la prise de décision, permettant le choix de structure

D'autres types d'outils d'aide à la conception peuvent être utilisés en génie électrique, notamment ceux apparaissant dans des phases plus amont, proposant au concepteur une aide à la prise de décision. C'est notamment le cas pour le choix des structures des dispositifs à concevoir.

L'analyse fonctionnelle correspond à une première technique d'aide au choix de structure. Elle s'appuie sur une logique de conception descendante [CHA 93], partant de la description des fonctions à remplir pour arriver à la spécification des solutions y répondant. Ces dernières spécifications peuvent donner lieu à un ensemble de choix de structures en raison d'une potentielle diversité de solutions technologiques répondant à une même fonction. Cette méthodologie de conception peut, par exemple, s'appliquer dans le cadre de la conception de convertisseurs statiques [LEC 98], pour lesquels divers composants physiques réalisent les mêmes fonctionnalités (*ex : cellule de commutation*).

Des techniques de systèmes experts ont également été utilisées pour l'aide au choix de structures. Ces outils cherchent à reproduire le raisonnement d'un expert du domaine, en suivant un raisonnement déductif à partir d'une liste de règles la plus exhaustive possible. De tels outils peuvent, par exemple, permettre de choisir des structures de divers appareillages (*moteur [TRI 91], contacteur [GEN 91][GEN 92], ensemble convertisseur-commande-machine [GER 93a][GER 93b]*) parmi diverses alternatives.

---

[1] FEM : finite element method

[2] BEM : boundary element method

I.B.4. Outils automatisant des parties d'analyse, permettant le dimensionnement

On peut considérer une autre avancée consistant à automatiser une partie de l'analyse des résultats de simulation, permettant ainsi d'itérer rapidement le cycle de mise au point du prototype. Le dimensionnement du produit à concevoir peut, en effet, être réalisé automatiquement par la résolution d'un problème inverse. En s'appuyant sur le problème direct, qui est simuler le comportement d'un dispositif soumit à régime de fonctionnement donné, on peut arriver, en formulant des contraintes sur le comportement souhaité (*approche déclarative*), à caractériser (*ou dimensionner*) ce dispositif. Des méthodes d'optimisation, qui permettent de résoudre ce problème inverse, peuvent être mises en œuvre dans deux approches complémentaires.

Tout d'abord celle du pré-dimensionnement, mettant en œuvre des modèles rapides de type analytiques, établis sur un certain nombre d'hypothèses simplificatrices. Ce genre d'approche permet au concepteur d'étudier rapidement le comportement de la solution de conception, par rapport à ses buts qui ont été exprimés sous la forme d'une « fonction objectif » et de contraintes [WUR 96a][COUT 99][SAU 00][ATI 03].

L'autre approche favorise un dimensionnement fin d'une structure simulée numériquement. L'objectif principal est alors de caractériser une solution optimale par rapport à certains paramètres prépondérants [VAS 94a][VAS 94b][SAR 99][CAL 01].

I.C. Quels peuvent être les futurs outils ?

La tendance des recherches en CAO n'a cessé de fluctuer entre différents types comme ceux évoqués précédemment. Certaines modes apparaissent et cherchent à mettre de côté les tendances passées. Il y a une dizaine d'années, les concepteurs de « systèmes intelligents » voyaient, par exemple, l'aboutissement de la « CAO conventionnelle » de simulation physique. D'autres pensaient que ces derniers outils étaient effectivement aboutis techniquement, mais qu'il restait à améliorer les méthodes de communication entre l'homme et la machine [TOM 85].

Des améliorations dans ce sens ont effectivement fait progresser les outils de CAO, grâce à des disciplines telles que l'interaction homme-machine [SHA 96] (*HCI : Human Computer Interaction*), allant même jusqu'à l'utilisation de technologies de réalité virtuelle pour simuler des expérimentations [DEW 01]. L'évolution de disciplines, comme l'interaction homme-machine, mène aujourd'hui à des recherches telles qu'en psychologie cognitive

ergonomique, dont les applications dans le domaine de la conception sont très prometteuses [DAR 01].

D'un côté, les performances d'outils numériques d'analyse exploitant pleinement les capacités de traitement informatique sont des outils nécessaires à la CAO. D'un autre côté, se frottant à des limites de formalisation du raisonnement humain, des outils tels que les systèmes experts, ne parviennent toujours pas à des solutions viables pour les bureaux d'étude en génie électrique.

En reconsidérant les outils de conception par rapport aux connaissances qu'ils doivent manipuler et qu'ils peuvent mettre en œuvre, nous allons poursuivre par ce que nous pensons être leurs enjeux pour une CAO plus efficace.

II. Les enjeux de la CAO

II.A. Considérer les informations nécessaires pour concevoir le produit

II.A.1. Le produit à concevoir : une entité physique

Les outils de CAO reposent, en général, sur un modèle des données du produit formalisé et exhaustif qui considère l'objet dans sa matérialité. Il s'agit d'un modèle comportemental destiné à simuler l'objet tel qu'il sera dans la réalité (*induction magnétique, trajectoires de flux, efforts exercés, températures*…).

Nous avons retenu des caractéristiques générales misent en œuvre par la plupart des outils de CAO dans leur modèle :

- Il est unique (*censé refléter le produit dans tout contexte*),
- il cherche à être à la fois complet et précis (*deux directions antinomiques*).
- Il est orienté « produit matériel » (*néglige la prise en compte de son environnement, de sa durée vie, de sa fabrication, de son recyclage, de son impact économique*…),

II.A.2. Le produit à concevoir : des modèles contextuels

Le premier constat est l'apparition de brèches techniques dans la vision d'un modèle unique, précis et complet. En effet, un tel modèle n'est parfois pas réalisable lorsque, par exemple, il est nécessaire de considérer des phénomènes microscopiques et macroscopiques simultanément [BAR 00]. La prise en compte de phénomènes de compatibilités électromagnétiques (*CEM*) dans un circuit électrique impose de travailler

dans large plage de fréquences, ce qui peut conduire à des problèmes de résolution numérique.

D'autre part, les besoins de la conception viennent également quelque peu modifier cette vision. En effet, la seule connaissance du comportement du système ne suffit pas à concevoir un dispositif. Il est nécessaire d'ajouter aux modèles comportementaux d'autres paramètres, tels que ceux liés aux contraintes de fabrication, aux enjeux économiques, à l'environnement dans lequel sera plongé le produit, ou à toute la durée de vie du produit incluant le recyclage.

Les concepteurs du domaine du génie industriel ont notamment pour objectif de prendre en compte la fabrication dans la conception, ce qu'ils appellent « l'intégration produit-process ». En génie électrique, les contraintes liées à la fabrication apparaissent également fortement, comme en technologie silicium [SIL 02] (*microélectronique de puissance, microsystèmes magnétiques, etc.*).

En se référant à la problématique d'intégration produit-process, des travaux de recherche de conception en génie industriel [LAU 00] soulignent deux caractéristiques fondamentales de l'activité de conception. La première correspond à la nature parfaitement imprévisible et nécessairement contextuelle de la conception. La deuxième exprime le besoin de traiter à la fois d'une dimension technique (*connaissance du produit*) et d'une dimension liée à l'activité de conception :

Un modèle de la CAO ne peut être exhaustif, de par la nature imprévisible de la conception. Il ne peut plus être qu'orienté produit, mais doit aussi intégrer les connaissances liées à l'activité de conception.

> *Il n'existe pas à proprement parler d'instrument d'intégration (produit-process). L'intégration apparaît sur le terrain, dans l'action effective de conception à travers la combinaison de plusieurs éléments, comme l'instrument mais également comme l'organisation ou le savoir. Deux dimensions sont présentes pour « l'instrument pour l'intégration » : une dimension technique et une dimension liée à l'activité de conception.*
>
> *Laureillard, thèse de génie industriel et sociologie [LAU 00]*

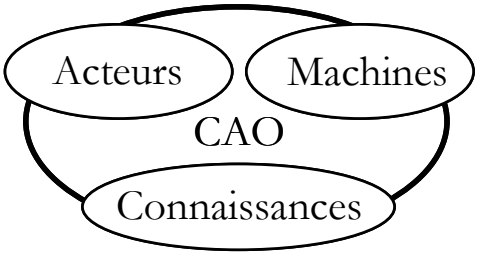
II.B. Considérer une connaissance propre à l'action du concepteur dans la CAO

Comme on peut le remarquer dans la présentation des outils pour la conception, une direction que peuvent prendre certains d'entre eux consiste à automatiser, par l'informatique, le maximum de tâches de conception. La course à l'automatisation connaît

tout de même une frontière, celle du traitement de la connaissance mis en jeu dans le processus de conception.

La CAO en génie électrique actuelle ne sait traiter efficacement qu'un type unique de connaissance, celui propre à l'artefact à concevoir. Cette connaissance de « la chose » (*lois physiques*) s'appuie sur une épistémologie cartésienne (*positiviste ou réaliste* [MOI 95]), dont les sciences de l'ingénieur, et plus particulièrement le génie électrique, font usage de manière intuitive.

Pourtant, la position particulière du concepteur au sein de la CAO et du processus de conception, induit la présence d'autres connaissances (*règles sociotechniques ou cognitives*). Ces connaissances ont trait à une autre épistémologie (*constructiviste* [MOI 95]) qui n'est pas familière au domaine du génie électrique. De ce fait, elle n'est que peu considérée durant la définition des outils de CAO. Pourtant des recherches effectuées dans le domaine de l'ergonomie cognitive pour la conception [DAR 01] définissent les enjeux de la prise en compte de telles connaissances. Pour le moment, seuls les domaines faisant intervenir une coopération entre acteurs, réussissent à intégrer ces enjeux [BOU 00a].



Un enjeu majeur pour les outils de CAO est de considérer les places relatives de l'homme et de la machine, par rapport aux connaissances mises en jeu.

Figure 2 : La CAO doit être considérée par les places relatives qu'occupent le concepteur, la machine et les connaissances mises en œuvres.

II.C. Libérer le concepteur de tâches fastidieuses et répétitives

Ignorer les relations entre concepteur et CAO, peut conduire à vouloir pousser à l'extrême le vœu d'automatiser la conception, positionnant ainsi les outils comme des substituts du concepteur.

***Actes cognitifs** : actes par lesquels un organisme acquiert des informations sur l'environnement et les élabore pour régler son comportement.*

Exemples :
- ➥ *processus créatif,*
- ➥ *formulation d'hypothèses,*
- ➥ *prise de décision,*
- ➥ …

Il faut repositionner les enjeux de la CAO qui sont principalement de soulager le concepteur dans ses tâches fastidieuses et répétitives, afin de libérer son temps à la pratique d'une activité plus **cognitive** de la conception. Une automatisation doit également servir à assurer un traitement de l'information sans erreurs pour des tâches de saisie ou de recopie, à mémoriser et à sauvegarder les informations traitées, etc.

Le traitement informatique de tâches compliquées, telles que l'analyse de milliers d'éléments de maillage reconstruisant un comportement global à partir de lois locales, peut faire bénéficier au concepteur de plus de temps pour imaginer des solutions alternatives originales et performantes. Il peut également développer de nouvelles méthodologies, conduisant à des conceptions plus rapides, avec de meilleurs résultats, intégrant plus de paramètres, etc. Cette automatisation apporte donc des perspectives que l'on n'envisage pas toujours lorsqu'elle n'est pas disponible.

Soulager le concepteur dans ses tâches fastidieuses et répétitives, afin de libérer son temps à la pratique d'une activité plus cognitive de la conception

III. Conclusions

Ayant décrit quelques outils utilisés durant un processus de conception, nous avons pu remarquer l'efficacité computationnelle de la CAO. Une analyse des connaissances mises en œuvre dans ces outils révèle qu'ils traitent principalement d'informations techniques du produit. Les enjeux pour la CAO résident dans une prise en compte de telles informations techniques, mais de diverses natures (*produit, fabrication, environnement, durée de vie,* …) servant des buts contextuels (*modèles rapide de pré dimensionnement ou de simulation fine,* ...), passant ainsi d'un modèle exhaustif et unique du produit à des modèles divers et évolutifs.

De manière complémentaire à ces informations techniques, la CAO devra de plus en plus considérer (*et non traiter*) une connaissance complexe liée à l'action du concepteur, en reconsidérant leur place à tous deux par rapport à la machine. L'objectif ultime de la CAO étant d'automatiser les tâches compliquées, afin de libérer le concepteur des tâches fastidieuses, lui apportant ainsi une aide nécessaire afin qu'il puisse traiter des tâches complexes telles que celle de l'acte cognitif (*processus créatif, formulation d'hypothèses, prise de décision,* …).

Besoin d'un environnement pour maîtriser l'interaction entre le concepteur et la CAO, durant tout un processus de conception.

Ces différents enjeux se posent pour la CAO en général mais peuvent trouver une solution au niveau des environnements de conception que nous détaillerons au prochain chapitre. Afin de proposer des environnements adaptés à ces enjeux, il est nécessaire de comprendre les causes ayant retardé l'apparition de telles solutions. C'est ainsi que nous allons exposer la nature de la conception sous deux points de vues, celui de la complexité et celui d'une imprévisibilité de la conception, qui nous paraissent mettre en évidence les principaux freins.

PARTIE II : NATURE DE LA CONCEPTION

I. La complexité en conception

I.A. Différence entre la complexité et le compliqué

I.A.1. Les systèmes compliqués

Il existe une différence entre le terme « compliqué » et le terme « complexe » que J.-L. LeMoigne [MOI 90] met en exergue pour argumenter des besoins d'une modélisation systémique (*vision système*) et plus uniquement analytique (*issue de l'analyse par décomposition*).

➥ Un système **compliqué** : on peut le simplifier pour étudier ses parties de manière indépendantes.

Le compliqué peut se caractériser, en conception de dispositifs d'électronique ou de microélectronique par exemple, par l'assemblage de nombreux composants assurant des fonctionnalités élémentaires bien définies, de manière indépendante. Il est ainsi possible d'automatiser le traitement d'assemblages grâce à des formalismes logiques d'association.

En tant que scientifiques, nous gérons instinctivement les systèmes compliqués grâce au précepte cartésien :

> *Diviser chacune des difficultés que j'examinerai en autant de parcelles qu'il se pourrait et qu'il serait requis pour les mieux résoudre.*
>
> *René Descartes (deuxième précepte, Discours de la Méthode - 1637)*

I.A.2. Les systèmes complexes

➥ Un système **complexe** : on peut le modéliser pour construire ses interrelations.

*Un système **compliqué** : on peut le simplifier pour étudier ses parties de manière indépendantes.*
*Un système **complexe** : on peut le modéliser pour construire ses interrelations.*

Le complexe peut se caractériser, en conception d'appareillages électrotechniques par exemple, par le besoin de modéliser les interrelations entre des parties du système. Considérer comme indépendantes des parties magnétiques d'un dispositif électrotechnique, ne permet pas de représenter son comportement global.

La complexité impose au concepteur de faire des hypothèses sur le chemin principal du flux et sur les flux de fuite, selon les positions relatives des constituants, permettant ainsi de modéliser leurs interactions.

> *Toutes choses étant causées et causantes, aidées et aidantes, médiates et immédiates, et toutes s'entretenant par un lien naturel et insensible qui lie les plus éloignées et les plus différentes, je tiens impossible de connaître les parties sans connaître le tout, non plus de connaître le tout sans connaître particulièrement les parties.*
>
> *Blaise Pascal (1623-1662), Pensées.*

La complexité en conception réside dans la difficulté de concevoir indépendamment les parties du système.

Le fait que « toutes choses soient causées et causantes » implique que la complexité, en conception, réside dans le besoin de considérer comme dépendantes les parties du dispositif à concevoir.

Le principe de simplification ne fonctionne plus si le dispositif étudié a un « effet système », c'est-à-dire un comportement qui ne soit pas défini par les parties elles-mêmes, mais par le tissu qu'elles forment.

Il est donc nécessaire de se demander, avant d'appliquer le principe de simplification, si le système étudié est compliqué ou complexe, car en effet :

> *La simplification du compliqué appliqué au complexe a pour conséquence une aggravation de la complexité*
>
> *J.-L. LeMoigne [MOI 90]*

I.B. Complexité des systèmes à concevoir

I.B.1. Complexité par le couplage inter-disciplinaire

Les couplages multi-physiques caractérisent bien la complexité des systèmes actuels. En effet, les applications du génie électrique partagent aujourd'hui un vaste panel de couplages interdisciplinaires, intégrant des compétences telles que la thermique, la mécanique des fluides, la résistance des matériaux, … Il est difficile de concevoir des systèmes en traitant indépendamment chacun de ces domaines disciplinaires. La conception du circuit magnétique d'un stator devra par exemple tenir compte de l'écoulement d'un fluide pour le refroidissement. La conception d'un micro-actionneur magnétique devra nécessairement s'appuyer sur des géométries réalisables en technologie silicium, etc…

I.B.2. Complexité par le couplage système

Les dispositifs à concevoir sont de plus en plus sophistiqués, introduisant toujours plus d'intelligence et d'autonomie dans des dimensions toujours plus faibles. Chaque actionneur possède son convertisseur d'énergie et son système de commande. Des capteurs micrométriques intégrés doivent supporter leur propre alimentation et pouvoir communiquer directement avec l'extérieur, par exemple par radio fréquence. Des contraintes entre les constituants de ces systèmes peuvent, en effet, apparaître telles que celles de l'encombrement total, de la compatibilité électromagnétique, du choix d'un interfaçage commun, du choix de techniques de fabrication communes, etc. Ce faisant, il n'est plus envisageable de définir de manière séparée, les composants constitutifs d'un système.

I.B.3. Complexité par le contexte d'utilisation

Le contexte d'utilisation du dispositif peut aussi contraindre sa conception, par rapport à son utilisation même, ou par rapport à son environnement extérieur. De telles contraintes peuvent correspondre à des normes de pollution (*sonores, électromagnétiques,* …), des conditions de fonctionnement de plus en plus diverses, des durées de vie de plus en plus difficiles à calculer, des matériaux recyclables, l'esthétique, …

I.C. Complexité du processus de conception

Outre le besoin de considérer la complexité des systèmes à concevoir, une autre face de la complexité peut apparaître, celle liée au processus de conception. Nous avons définit dans les enjeux des outils de la CAO, l'objectif d'automatiser le compliqué pour donner au concepteur les moyens de traiter des tâches cognitives telles que la formulation des problèmes ou la prise de décision. Nous détaillons donc, ici, en quoi ces tâches peuvent être considérées comme complexes.

I.C.1. La conception : un processus cognitif

***Concevoir** : spécifier des entités soumises à des objectifs et des contraintes*

La **conception** dont le but est de spécifier des entités soumises à des objectifs et des contraintes est un acte cognitif de la catégorie des « résolutions de problème », qui compte également la prise de décision (*choix d'alternatives*), le diagnostic (*recherche de causes*) ou la prédiction (*recherche de conséquences*).

Un processus cognitif simplifié de la conception est ainsi défini dans [MAL 80] :

- Elaboration des buts (*ex : définition du cahier des charges fonctionnel*)
- Elaboration d'une solution (*ex : conception innovante, routinière ou assemblage*).
- Vérification de la solution (*ex : les performances sont elles atteintes ?, si non que modifier ?*)

Trois tâches cognitives apparaissent alors dans ce processus, la formulation du problème, la créativité, la prise de décision. La complexité se matérialise alors dans l'incapacité de l'informatique à automatiser ce processus par ses seuls moyens de calcul.

H.A. Simon [SIM 87] définissait la résolution de problème comme : une recherche sélective dans un large panel de possibilités guidée par des heuristiques. L'intelligence artificielle a essayée d'apporter quelques solutions pour des problèmes rationnels, formulés explicitement. Mais une particularité de la conception est qu'elle fait partie d'une catégorie de résolution de problèmes dits « mal formulés » (*ill-defined*), pour laquelle le problème est successivement reformulé durant l'exploration des solutions.

Selon Simon, les systèmes experts devaient être capables de simuler ce « processus flexible de résolution de problème » par la connaissance des critères de conception et des moyens permettant de les satisfaire, ainsi que par la faculté de continuellement modifier la définition du problème par l'application de ces critères. Cette démarche ambitieuse est à nuancer aujourd'hui par rapport aux résultats qu'ont donnés ces outils de CAO.

I.C.2. La complexité du processus de conception le rend non automatisable

Les principes mis en œuvre dans les systèmes experts, ou plus généralement les systèmes d'aide à la décision, ne permettent pas de reproduire la complexité des actes cognitifs en jeu durant un processus de conception. Par contre, ils constituent des outils de CAO qui, prenant explicitement en considération les connaissances du concepteur, l'assistent dans le traitement de ces tâches complexes.

> *Le système (SIAD : système interactif d'aide à la décision) implanté en machine est considéré comme n'«apportant» pas «la» solution, mais comme jouant le rôle d'amplificateur cognitif, mettant l'accent sur une ou plusieurs phases du processus de décision.*
>
> *Munier Bertrand, « Décision », Encyclopædia Universalis France, 1999*

Nous pouvons considérer le processus de conception par l'enchaînement de tâches automatisables relevant d'une gestion du compliqué, que la puissance des calculateurs sait gérer grâce au principe de simplification. Le processus de conception fait, par exemple,

intervenir les outils du concepteur qui permettent l'aide au choix de structure, l'aide à la modélisation, la simulation, le dimensionnement, etc. Malgré un certain nombre de tâches bien identifiées, que l'on peut plus ou moins automatiser, la complexité de la conception réside essentiellement dans les transitions cognitives, guidant le cheminement entre ces tâches.

La complexité de la conception réside dans les tâches cognitives qui en guident le processus.

I.D. Conclusions

Nous venons de faire le constat d'une complexité apparente à deux niveaux, celui des systèmes à concevoir et celui du processus de conception. Afin d'offrir des supports efficaces à cette complexité, nous devrons tout d'abord considérer les systèmes par les divers couplages dont ils sont tissés. Nous pourrons ensuite gérer la complexité du processus de conception en prenant en compte, de manière explicite, le rôle du concepteur dans le déroulement du processus de conception.

Nous allons voir maintenant qu'une CAO, cherchant à supporter les différents processus de conception, doit nécessairement faire face à une caractéristique particulière à la conception, celle de l'imprévisibilité.

II. Un processus de conception dynamique et imprévisible

II.A. Tentatives de formalisation d'un processus de conception

Le besoin de supporter le processus de conception dans la CAO a fait émerger le besoin de formaliser les processus de conception dont on peut trouver dans la littérature plusieurs tentatives de description.

II.A.1. Quelques formalisations pour la CAO

Certaines définitions de la conception peuvent insister sur des points particuliers, comme le choix d'une meilleure solution. En effet, Gero [GERO 85] décrit la conception comme « une activité du type résolution de problème, orienté par les buts », dans laquelle l'objectif est de trouver une solution, vérifiant le cahier des charges (*contraintes et objectifs*) et qui soit optimale (*c'est-à-dire la meilleure, selon certains critères, parmi celles trouvées*).

D'autres formalisations de processus de conception peuvent faire apparaître le besoin d'atteindre la solution de conception « chemin faisant », et peuvent intégrer les étapes nécessaires à la production du produit [TOM 85] :

- solution conceptuelle (*conceptual design*) : décide grossièrement des méthodes et des structures d'une solution de conception,
- solution rudimentaire (*basic design*) : décide du cadre et de la structure de la solution de conception,
- solution détaillée (*detailed design*) : spécifie la solution de conception,
- solution pour la production (*production design*) : génère les données nécessaires à la production,
- prototypage et test.

Intégrer dans le processus de conception des étapes de pré-calcul et de calcul fin.

Dans ce processus, les auteurs proposent la recherche d'une solution en étapes de finesse croissante, correspondant à d'autres processus de conception pouvant mettre en œuvre des étapes de pré-calcul et de calcul fin [WUR 97]. Nous utiliserons cette méthodologie dans le *Chapitre IV* sous le nom de métaphore du microscope.

Intégrer dans la conception, le processus de fabrication.

On peut également remarquer une prise en compte du projet de fabrication, mais découplée de la conception détaillée du produit. Il apparaît aujourd'hui que cette séparation peut conduire à des retours intempestifs dans les étapes de conception [LAU 00], caractéristique révélatrice d'une complexité des systèmes à concevoir.

Ces définitions de processus de conception sont issues de contextes bien spécifiques et ne peuvent s'appliquer que pour un nombre de cas bien définis. Cherchant à généraliser la démarche de conception, elles deviennent alors trop abstraites pour servir de formalisation à un support informatique du processus.

II.A.2. Formalisation pour l'entreprise

Des formalisations plus pratiques de la conception ont également été proposées pour l'industrie [DAR 01] :

- Etude de faisabilité (*analyse de la valeur et analyse fonctionnelle*),
- Spécification technique des besoins (*en phase d'avant projet*),
- Développement.

L'ingénierie de conception, par le biais de ces procédures et de normes qui en découlent (*norme BS7000 en Grande-Bretagne, norme DIN ou VDI 2221/2 en Allemagne, AFNOR X50-127 en France*), cherche à améliorer son efficacité et ses performances. Mais ces représentations idéalisées du processus de conception ne reflètent pas sa réelle complexité. Elles ont d'ailleurs été mises en défaut par des études (*Culverhouse, 1995 cité dans [DAR 01]*) qui ont

montré qu'elles ne permettaient pas d'améliorer le processus (*notamment dans sa durée*) avec autant d'efficacité que cela a pu être le cas pour des processus de production.

II.A.3. Pourquoi la formalisation est elle si délicate ?

Les différents travaux de formalisation pour la CAO cherchent à mettre en oeuvre une *méta-représentation* du processus, directement utilisable pour développer un environnement informatique supportant diverses applications de conception. Or, une formalisation adaptée au développement d'un environnement informatique est d'abord statique. En effet, il est plus facile de programmer un logiciel qui prévoit tous les cas envisageables, plutôt qu'un logiciel capable de s'adapter.

En 1931, le mathématicien Gödel démontre que tout système formel consistant (*c'est-à-dire qu'il ne se contredit pas lui même*) est nécessairement incomplet (*théorème d'incomplétude*) [GOD 85]. C'est-à-dire qu'il ne sera pas suffisant pour résoudre certains problèmes, qu'il faudra sortir de ses limites pour parvenir à des solutions, et que ce passage à des niveaux méta est sans fin [HOF 85]. Les diverses tentatives de formalisation ont confirmé cette vision des difficultés et des limites de nos raisonnements sur nos propres raisonnements.

Ainsi, comme nous allons le voir, un processus de conception n'est généralement pas prédéfini et se construit de manière itérative. La *Figure 3* nous illustre par exemple le fait que les buts finaux de conception s'élaborent « chemin faisant » [MOR 99] par l'expression de solutions et buts intermédiaires. Par des ajustements et retours en arrière, il est ainsi possible de converger vers des buts précis, et des solutions y répondant.

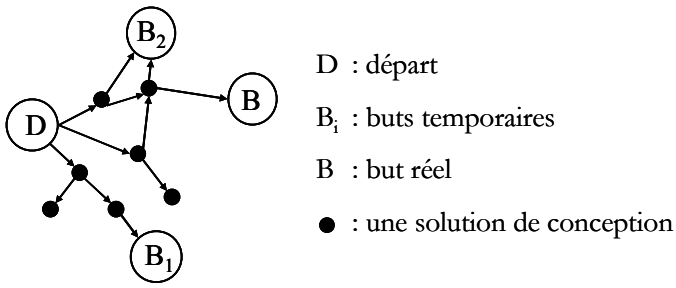


Figure 3 : Les buts de conception se construisent durant la conception (extrait de Tomiyama et Yoshikawa [TOM 85])

La notion de recherche d'une solution en étapes de finesse croissante est essentielle pour prendre en compte la véritable nature de la conception. Nous allons détailler quelques propriétés intrinsèques au processus de conception que nous devrons prendre en considération, tel que le fait qu'il ne soit pas prédéfini, mais itératif et changeant.

II.B. Considérer un processus dynamique et imprévisible

II.B.1. Le processus de conception n'est pas prédéfini

On peut définir le processus comme causal en première approximation. C'est-à-dire que les résultats de chaque étape peuvent influer sur la nature des suivantes. En effet, l'élaboration des buts conditionne le choix de la méthodologie de conception. Un processus de re-conception ne sera par exemple pas le même qu'un processus de conception innovante. De la même façon, la « définition des solutions » va conditionner la manière de choisir et vérifier ces solutions.

Chaque étape du processus influe sur la nature et le déroulement des suivantes.

En plus de cela, on se rend compte qu'il existe implicitement des liens causaux inverses. En effet, l'utilisation d'un outil d'analyse est prescripteur de la modélisation mise en œuvre, ainsi que des types de solutions de conception envisagés.

II.B.2. Le processus de conception est itératif

B. Chandrasekaran [CHA 93] définit le processus de conception de manière classique, comme étant l'exploration d'espaces de solutions, la simulation et la vérification des solutions de conception. Mais en stipulant la possibilité de re-conception et de répétition de ce cycle.

Ces itérations peuvent être dues à plusieurs facteurs. Tout d'abord, celui correspondant à cette notion fondamentale de clarification, de reformulation, évoqué plus haut. Chaque étape va faire apparaître des écueils ou des insuffisances dans les phases amonts, nécessitant potentiellement de reboucler sur tout le processus, ou sur quelques étapes.

Chaque étape du processus remet en cause des hypothèses et des choix faits dans des étapes précédentes.

Ces itérations peuvent également venir du fait que le processus de conception met souvent en avant des phases amonts abstraites (*telles que l'analyse fonctionnelle*) qui sont séparées des phases avales plus concrètes (*comme le choix d'éléments réalisant une fonction particulière*). Or, des études [DAR 01] ont montré que le concepteur entremêle nécessairement ces phases, passant par des représentations concrètes du produit, afin de mieux en cerner les contraintes.

Les divers cycles sont plus généralement dus à une remise en cause des buts à atteindre. Considérons l'exemple du processus suivant, dans lequel sont décrites, à différentes étapes, les besoins de telles itérations.

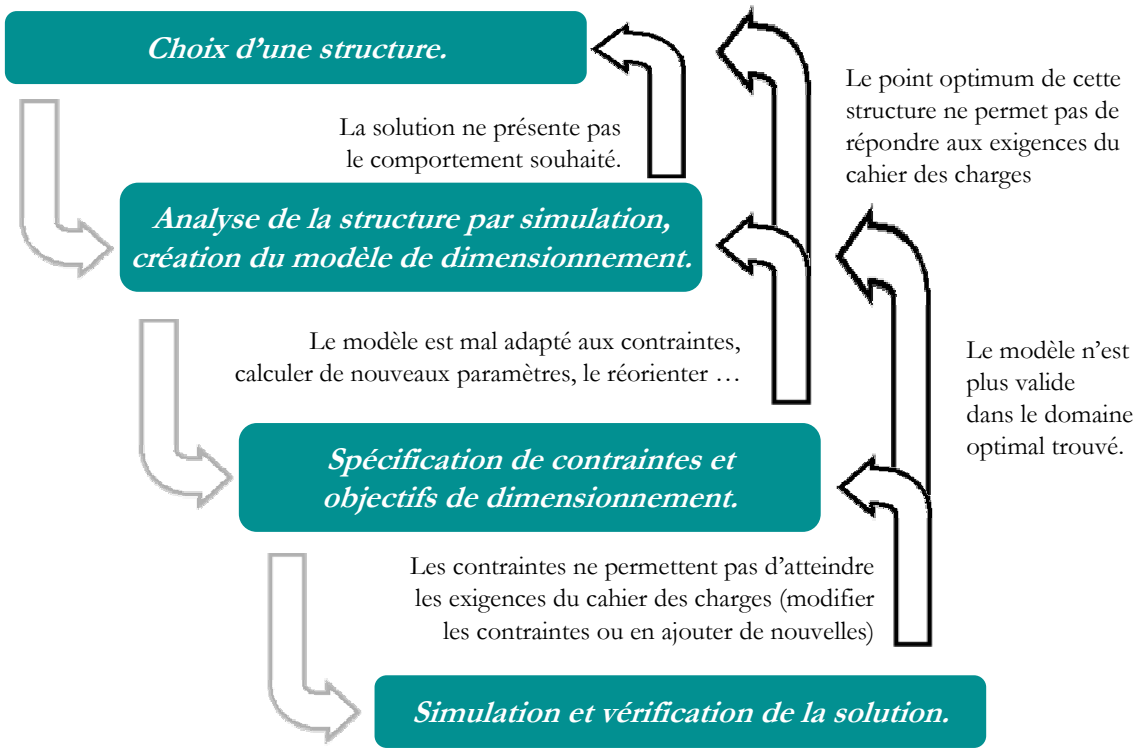


Figure 4 : Un processus de conception nécessitant des itérations à tous les niveaux.

II.B.3. Le processus de conception est contextuel

Les conceptions par similitude ou routinières [BAS 00], utilisant des solutions de conception passées, vont réitérer un processus pour obtenir une solution à un nouveau cahier des charges. Mais un processus de conception est changeant, car dépendant de paramètres qui évoluent dans le temps. Ainsi, des facteurs économiques, industriels, technologiques ou méthodologiques, peuvent faire que de nouveaux objectifs de conception ne seront plus adaptés à un processus dont le contexte de mise en oeuvre aura changé.

Chaque conception est unique par ses objectifs, les acteurs qui y participent, mais également le contexte dans lequel elle se réalise.

Issu de l'interaction entre un concepteur et ses outils, un processus de conception est intimement lié à la connaissance mise en œuvre dans cette interaction. Les recherches menées dans le domaine de la représentation de la connaissance ont évolué depuis quelques années, d'une « matière à extraire et à mettre dans une boite » à un « modèle de connaissance qu'il s'agit de repérer, structurer,

exploiter et maintenir en une spirale sans fin ». Ainsi, pour une CAO cherchant à traiter des savoirs faire de conception (*ce qui n'est pas notre propos*), les méthodes d'intégration et de réutilisation de la connaissance doivent être considérées au sein même de la CAO [EYN 00].

La capitalisation de processus doit pouvoir être utilisée de manière complémentaire à la définition de nouveaux processus de conception.

Un processus de conception est donc soumis à cette même règle d'évolution en spirale, devant être remis en cause à chaque nouvelle utilisation. Bien sûr, cette notion peut paraître radicale et semble condamner toute notion de capitalisation. Il n'en est rien, et les processus existants doivent garder toute légitimité, ils doivent simplement pouvoir être remis en cause. Le moyen de modifier un processus de conception, ne doit pas détruire ceux existants, mais offrir la possibilité de s'en échapper lorsqu'ils ne sont plus adaptés.

III. Conclusions

Les mises en œuvre informatiques de processus de conception tendent à s'appuyer sur un processus de conception formalisé et statique. Or, nous venons de mettre en évidence la nature essentiellement dynamique et imprévisible du processus de conception. Il est en effet difficile de prévoir à l'avance chacune des étapes d'un processus de conception, de même qu'il est difficile de prévoir à l'avance l'enchaînement que va suivre le concepteur entre ces étapes.

De plus, la nature complexe de la conception mise en évidence en début de partie, nous demande de considérer explicitement le rôle du concepteur, qui est le seul à même de définir les multiples couplages qui apparaissent au sein du système à concevoir.

Pour ces raisons, nous estimons qu'un environnement de conception doit offrir un support dynamique aux processus de conception et permettre l'intervention du concepteur pour les définir dynamiquement. C'est sur ces bases que nous allons placer les concepts de notre environnement dans le chapitre qui suit.

Chapitre 2

LES ENVIRONNEMENTS DE CONCEPTION

C h a p i t r e 2

LES ENVIRONNEMENTS DE CONCEPTION

Nous avons présenté une vision de la CAO et de ces enjeux, par ses outils. Nous allons maintenant présenter brièvement les environnements de conception, en nous intéressant particulièrement aux environnements d'intégration, dont nous pensons qu'ils sont la solution la mieux adaptée aux enjeux décrits.

Le contexte de travail de cette thèse sera décrit afin de servir de base aux concepts d'une architecture à base de composant que nous allons proposer. L'utilisation de composants logiciels métiers nous permettra de construire cette architecture qui s'articulera autour de patrons de travail (*générateurs de composants et utilisateurs de composants*) et de différents concepts tels que celui de la composition ou celui de la projection des composants.

PARTIE I :

VERS UN ENVIRONNEMENT D'INTEGRATION

I. Les types d'environnements

> ***Environnement de conception :*** *support d'un certain nombre de tâches de conception dont l'agencement constitue un but de conception.*

La CAO offre des outils qui permettent au concepteur d'accomplir des tâches de conception bien définies. Un environnement constitue, quant à lui, un support aux processus de conception par la mise en œuvre de plusieurs de ces tâches dans un but de conception.

Le génie électrique est, en ce qui concerne les environnements de conception, bien moins équipé que d'autres domaines tels que la mécanique ou l'électronique et la microélectronique. Ces derniers bénéficient de caractéristiques qui leurs permettent, grâce à des compositions logiques de fonctions élémentaires, de formaliser des processus de conception efficaces car compliqués et non complexes. Cette formalisation offre alors la possibilité de développer des supports informatiques (*environnements*), qui fiabilisent et accélérent ces processus de conception.

Nous pouvons différencier, parmi les environnements, les architectures monolithiques réalisant elles mêmes tout un processus de conception, des architectures multi-outils qui chapeautent plusieurs outils de CAO. Nous allons maintenant décrire ces deux typologies d'environnements.

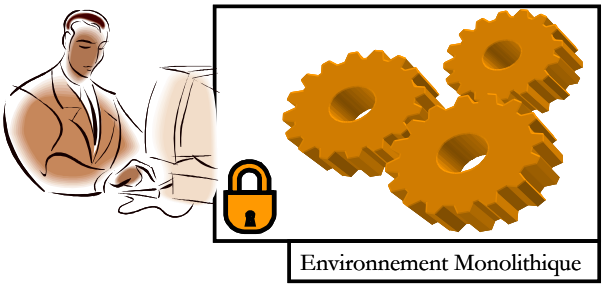
I.A. Des environnements intégrés mais monolithiques

I.A.1. Une vision extrême : l'autosuffisance

L'environnement intégré monolithique a comme caractéristique principale l'autosuffisance. Il permet généralement de traiter un processus de conception bien défini de *A à Z*. Il est capable de décrire le problème dans un formalisme qui lui est propre (*répondant à des besoins bien spécifiés*) et permet de le résoudre (*en plusieurs étapes si nécessaires*).

Les avantages d'un tel environnement peuvent être :

- une vision intégrée du processus pour le concepteur,
- l'autonomie (*palliant toute défaillance d'un outil dont il pourrait dépendre*),
- l'évolutivité (*l'autonomie permet d'évoluer sans contraintes*),
- la performance (*réalisant un but bien spécifié, l'environnement peut bénéficier d'une mise en œuvre spécifiquement adaptée et donc performante*)



Malheureusement, par cette fermeture à l'extérieur, un environnement monolithique devient rapidement spécialisé et prescripteur de la manière de concevoir. Cette solution va à l'encontre des enjeux que nous nous proposons de résoudre.

Figure 5 : L'environnement monolithique est autosuffisant, il est efficace dans les buts qu'il s'est fixés, mais reste fermé au monde extérieur.

I.A.2. Une vision plus modérée : l'ouverture

Les environnements actuels ne sont pas aussi résolument fermés et offrent de plus en plus d'ouvertures qui se justifient à plusieurs titres :

- L'environnement n'intègre pas toutes les fonctionnalités nécessaires à la mise en œuvre du processus qu'il propose, et exploite des capacités externes,
- Les moyens financiers de l'entreprise souhaitant acquérir l'environnement, ainsi que l'inertie psychologique du concepteur utilisant couramment un certain nombre d'outils existants et réalisants des fonctionnalités similaires, font que l'environnement devra exploiter ces capacités externes existantes.

L'ouverture, dans ce genre d'environnement, est d'abord réalisée grâce à des standards d'échange de données (*ex : STEP, IGES, …*). C'est ainsi que l'environnement peut, par exemple, exporter des géométries vers divers outils existants, en s'appuyant sur des standards de description géométrique.

Une intégration spécifique de certains outils du marché est également courante, ne correspondant pas réellement à une ouverture mais à un partenariat entre société développant les outils de la CAO.

Considérons l'exemple de l'environnement de conception de circuit intégrés Cadence[3], qui propose un processus dans lequel plusieurs outils permettent de répondre aux besoins d'une même étape de conception :

- **Capture schématique** : Concept HDL, Capture CIS, OrCAD Capture
- **Simulation Digitale/Analogique/Mixte** :NC-Sim, PCB Analog Expert, PCB Mixed-Signal Expert, PSpice
- **Analyse de circuits imprimés** :SPECCTRAQuest
- **Placement des composants et routage des connections** :SPECCTRA
- **Création des masques**: Allegro, Cadence DFMi, OrCAD Layout

Cet environnement offre une solution adaptée à la conception de circuits intégrés en microélectronique, car il bénéficie d'un processus de conception compliqué dans une large mesure (*mais non complexe*), lui permettant de spécifier les étapes, les outils de manière déterministe.

Considérons un autre exemple, celui d'un environnement de conception de microsystèmes, CoventorWare™[4]. Cet environnement intégré, offre le support d'un processus de conception spécifique aux microsystèmes (*cf. Figure 6*).

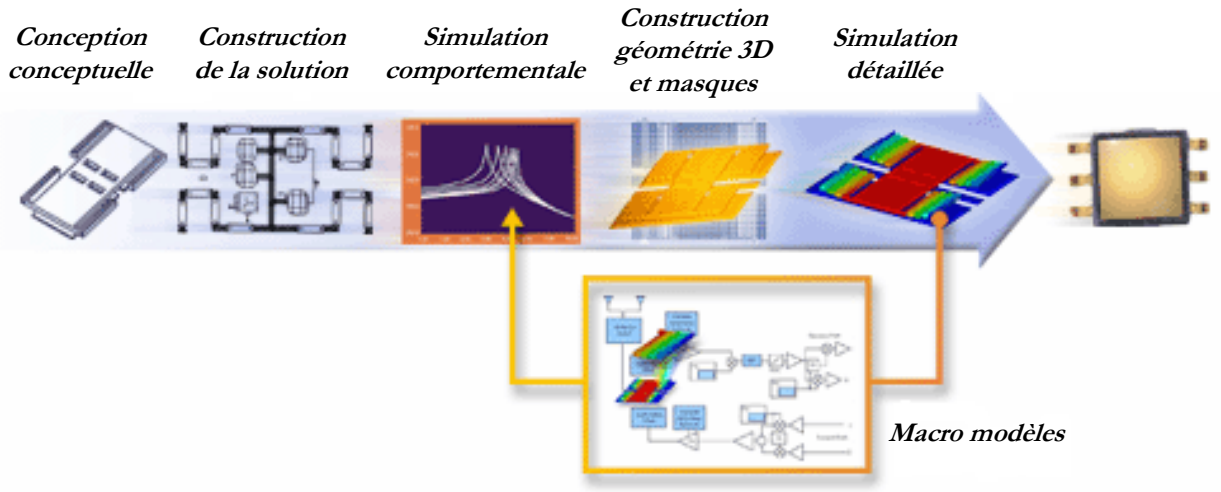


Figure 6 : Description du processus de conception de microsystèmes implanté dans l'environnement CoventorWare™

[3] CDE™: *Cadence Design Environnement*, Cadence Design Systems, Inc, www.cadence.com

[4] CoventorWare™ : Coventor Inc, www.coventor.com

Pour cela, il propose une série de tâches de conception bien identifiées (*cf. Figure 6*):

- **Construction de la solution** : ce module permet de définir un microsystème, à partir de composants prédéfinis dans des bibliothèques (*mécanique, électromécanique, optique, fluidique*).
- **Simulation comportementale**: à partir de la description du système, celui-ci peut être simulé, permettant d'analyser ses performances ou de réaliser des études paramétriques de sensibilité.
- **Création géométrie 3D et masques** : la géométrie peut être générée à partir des informations du système, ou par l'utilisation d'un modeleur géométrique. Les masques de fabrication peuvent alors être définis et le processus de fabrication simulé.
- **Simulation détaillée** : à partir de la géométrie, une analyse FEM / BEM peut être réalisée (*électrostatique, analyse de structure, thermo déformation, piezzo, couplage électro-mécanique avec hystérésis,* ...). Un macro modèle peut alors être généré, pour réintégrer ce dispositif dans les phases amonts du processus.

Cet environnement répond à un certain nombre d'enjeux que nous nous sommes posés. A savoir que le processus proposé commence par une description d'un modèle comportemental permettant de tester rapidement une solution conceptuelle, puis permet de simuler un modèle détaillé du dispositif. Il prend donc en considération le besoin de mettre en œuvre divers modèles du même produit, pour différents enjeux.

L'autre intérêt de cet environnement correspond à la prise en compte explicite de la fabrication dans le processus de conception, en proposant de générer les masques de fabrication mais également de visualiser les étapes du processus de fabrication.

Cet environnement, intégrant les domaines de l'optique, l'électronique et la physique (*cf. Figure 7*), se définit comme une solution complète à la conception de microsystèmes. Mais nous plaçant dans la peau d'un concepteur de microsystèmes magnétiques (*MAGMAS*), ayant à notre disposition des outils de calcul de tels systèmes (*ex : Dipole3D [DELA 93], outil de calcul de forces magnétiques basés sur une méthode des moments*), nous ne pouvons malheureusement pas bénéficier du potentiel offert par cet environnement pour notre conception.

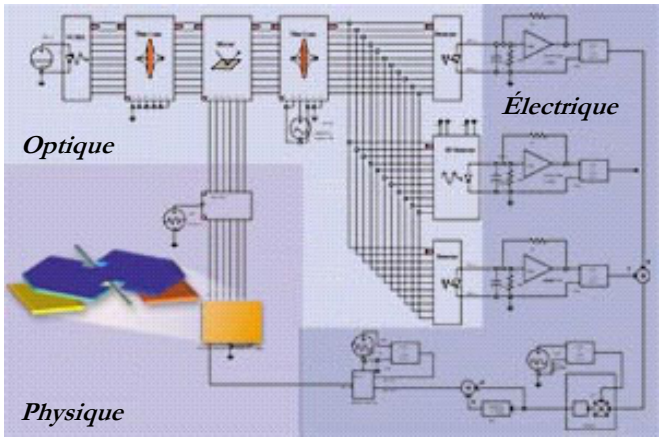


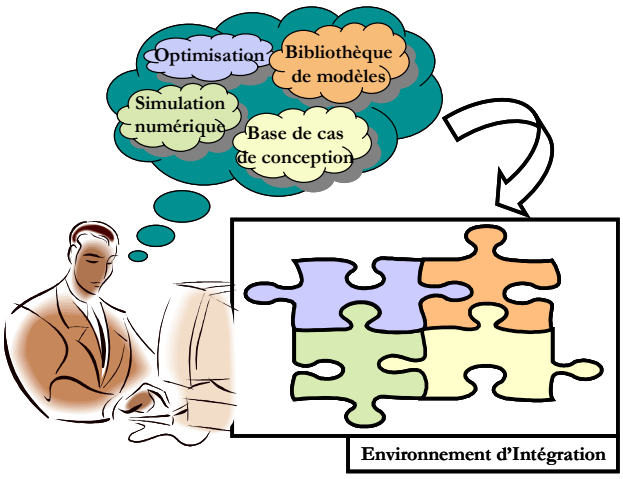
Figure 7 : CoventorWare™, intègre l'optique, la physique et l'électronique pour la conception de microsystèmes.

I.B. Des environnements d'intégration

Comme nous venons de le voir, les environnements spécifiques négligent l'aspect fondamental de l'ouverture. La deuxième typologie d'environnements est, quant à elle, résolument orientée vers l'intégration de capacités externes. La différence majeure avec ceux décrits précédemment, correspond au fait que les outils n'y sont pas prédéfinis mais qu'il existe des moyens techniques permettant de les intégrer.

Un environnement d'intégration est unificateur, il permet d'exploiter des capacités existantes, et permet de s'adapter.

L'idée principale de ces environnements est l'unification, par l'utilisation de services réalisés par d'autres outils. Ainsi, ils produisent une valeur ajoutée faisant que « le tout est plus que la somme des parties ».



La conséquence pratique, pour un tel environnement, est donc qu'il doit s'établir sur une architecture de communication, permettant d'envisager l'intégration d'outils de manière modulaire et donc évolutive (*cf. Figure 8*). Cette caractéristique d'intégration d'outils nous semble être la clé des environnements de conception qui intégrerons les contraintes liées à la complexité de la conception.

Figure 8 : L'environnement d'intégration est ouvert aux outils du concepteur et lui permet de définir son processus

Il n'existe pas à proprement parler d'environnement d'intégration spécifiques à la conception, cependant, nous allons décrire ce qu'ils peut être, en considérant divers exemples d'environnement d'intégration.

Ils peuvent être caractérisés par leur niveau d'ouverture et d'évolutivité. Nous pouvons ainsi définir les environnements spécifiques, intégrant des outils spécifiques dans le cadre d'un processus de conception prédéfini. Par opposition, nous pouvons définir les environnements génériques comme offrant au concepteur de grandes possibilités d'intégration et de mises en œuvre de processus.

I.B.1. Les environnement spécifiques

Un environnement d'intégration spécifique, est destiné à un but bien identifié.

Les environnements spécifiques sont réalisés pour répondre à un besoin bien identifié. Les outils qu'ils permettent d'intégrer correspondent alors à des catégories prédéfinies.

I.B.1.i Les environnements de supervision logiciel

Citons le cas d'environnements à base de connaissance, qui font coexister différents outils de calculs afin de proposer une aide à la mise en œuvre d'un processus de résolution [SAS 94][THO 95]. A partir d'une description du problème à résoudre, l'environnement met en œuvre le processus de résolution le mieux adapté, par l'exécution séquentielle des différents outils intégrés (*cf. Figure 9*).

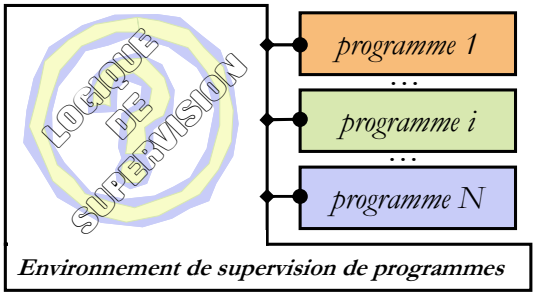


Figure 9 : Environnement de supervision de programmes, ayant spécifié la connaissance de pilotage des programmes

Dans [SAS 94], la connaissance qui définit le processus est catégorisée selon trois axes :

- **Stockage de la connaissance** statique (*conditions initiales, paramètres, valeurs, relations, etc.*) et dynamique (*les buts à suivre, les requêtes de données, les suggestions, etc.*) avec l'information d'évolution de cette connaissance.
- **Architecture de la connaissance** : connaissance sur le produit (*structure physique*), connaissance opérationnelle (*relationnelle, comportementale et procédurale*) des outils de calcul.
- **Control de la connaissance** : permet de raisonner sur le processus de résolution en fonction des spécifications initiales, du but final, et du type de problème. Ainsi, les actions les plus appropriées sont proposées et conduisent à un enchaînement donné des outils de calcul.

I.B.1.ii Les environnements d'optimisation paramétrique

Citons également le cas d'environnements permettant d'intégrer des logiciels de simulation à un algorithme d'optimisation, offrant ainsi la possibilité de dimensionner les dispositifs simulés (*cf. Figure 10*). Un certain nombre d'environnements de cette catégorie existent sur le marché[5].

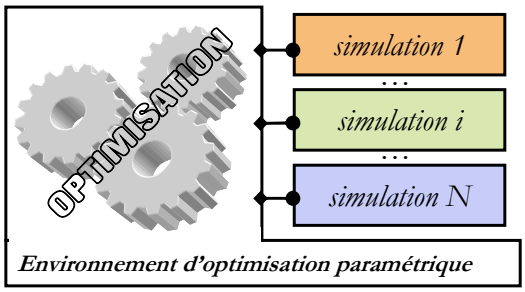


Figure 10 : Environnement d'optimisation paramétrique, ayant spécifié ce qu'est un outil de simulation afin de l'intégrer

[5] LMS International, Optimus : http://www.lmsintl.com/

Synaps, Pointer : http://www.synaps-inc.com/

Engineous Software Inc, iSight : http://www.engineous.com/

I.B.2. Les environnement génériques

D'autres environnements peuvent être qualifiés de génériques, au sens où, contrairement à ceux définis précédemment, ils n'offrent généralement pas un service spécifique. Ils offrent plutôt une structure de communication dans laquelle les outils pourront dialoguer, permettant une utilisation mutuelle de leurs services.

Citons l'environnement MatLab®[6] permettant d'intégrer des codes de calcul programmés en Java ou en C, et de développer des boites à outils « ToolBoxes ». Ces codes viennent compléter les fonctionnalités déjà présentes dans le noyau de MatLab® telles que la programmation d'algorithmes ou les fonctionnalités de post-processing, qui permettent de traiter un grand nombre de problèmes scientifiques. Pourtant, cet environnement ne répond pas à un objectif de gestion dynamique des besoins du concepteur. En effet, ne spécifiant pas une catégorie d'outil spécifique à intégrer, un investissement important en programmation est nécessaire (*cf. Figure 11*).

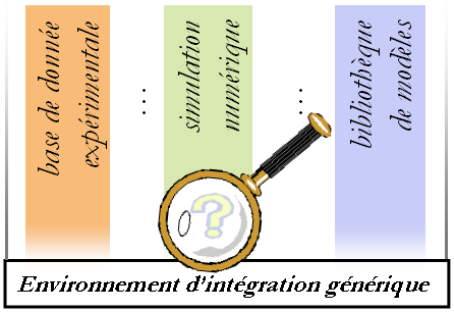


Figure 11 : L'environnement générique ne spécifie pas d'outils particuliers à intégrer, imposant un travail d'autant plus important d'intégration.

I.B.3. Conclusion : le choix de l'ouverture

Les environnements d'intégration, qu'ils soient dédiés à une tâche ou offrant des services plus génériques, autorisent par leur ouverture une plus grande souplesse que les environnements monolithiques. Afin de faire face aux contraintes surgissant durant une activité de conception, un environnement de conception doit s'appuyer sur cette ouverture qu'autorise l'intégration.

Les environnements spécifiques que nous avons décrits ne suffisent pas au support de tout un processus de conception, ne permettant d'intégrer qu'un type prédéfini d'outils.

Les environnements génériques ne spécifiant pas de règles particulières d'intégration, il est nécessaire de réaliser des programmations spécifiques à chaque cas. Ces environnements, demandant un coût d'intégration non négligeable pour chaque nouvelle application, ne permettent pas de gérer la dynamique et l'imprévisibilité du processus de conception.

[6] MatLab® MathWorks, Inc. : www.mathworks.com

II. Enjeux d'un environnement d'intégration

Pour le moment, qu'ils soient spécialisés ou qu'ils soient génériques, les environnements ne parviennent pas à concilier souplesse et accessibilité. Ceux spécifiques restent encore limités dans leur souplesse, ne permettant souvent d'intégrer qu'une étape d'un processus de conception bien défini. Ceux génériques sont difficiles d'accès pour un concepteur sans compétences suffisantes en informatique, et donc non adaptés à la conception.

Les environnements d'intégration nous paraissent, malgré le fait qu'ils ne soient pas encore aboutis, les plus adaptés aux besoins de conception actuels. En effet, ils sont tout d'abord une solution pour automatiser des échanges entre les outils du concepteur. Mais ils sont également une solution à la prise en compte de modèles pour la conception et connaissances de conception liée au concepteur qui les utilise.

II.A. Les enjeux pour le concepteur en génie électrique

Nous évoquons ici quelques uns des enjeux principaux d'un environnement d'intégration, qui selon nous, doit apporter un gain :

- d'efficacité (*rapidité, gestion du compliqué*),
- de fiabilité et qualité (*automatisation, versionnement*),
- et de capitalisation (*modèles et scénarii de conception*),

dans

- la mise en place (*méthodologies de conception et outils intégrés*),
- et l'exécution (*automatisation, gestion du flux d'information*),

des processus de conception (*cf. Figure 12*).

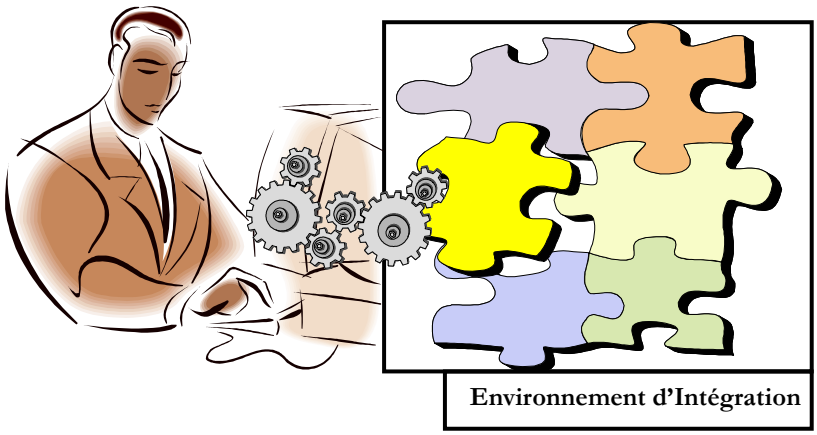


Figure 12 : Le concepteur ajoute à l'environnement de conception de nouvelles pièces à d'autres déjà capitalisées, définissant son processus de conception qui pourra s'exécuter de manière fiable et efficace.

II.B. Un environnement intégrateur des modèles pour la conception

Nous avons présenté dans le premier chapitre l'enjeu d'une CAO qui ne s'appuierait plus sur un modèle unique et exhaustif du produit. En effet, les modèles nécessaires à la conception ne se caractérisent plus uniquement par un modèle physique de la « chose » mais également par les modèles issus du raisonnement du concepteur, de ce qu'il imagine voir de réel ou d'abstrait, l'aidant dans ses tâches d'analyse. En effet, bien que le mot « modèle » tienne son origine de la « maquette », l'objet réduit, les modèles peuvent être définis comme des « figurations ou reproductions qui servent les buts de la connaissance » (*Encyclopædia Universalis*).Une modélisation doit donc se définir par son projet, en assumant explicitement le rôle du modélisateur-concepteur qui est à l'origine de ce projet.

Nous pensons que la conception, mettant en avant ce besoin de posséder plusieurs modèles associés à des buts de conception différents, trouvera un support adapté grâce à un environnement d'intégration et à ses différents outils de CAO.

Les différents outils intégrés dans un environnement de conception pourront représenter le produit à concevoir de façons complémentaires, pour réaliser, par exemple, des calculs de pré-conception et des calculs fins. L'environnement pourra ainsi mettre en œuvre, dans un processus de conception, des outils et leur modèle du dispositif associé selon les buts que le concepteur souhaite atteindre (*cf Figure 13*).

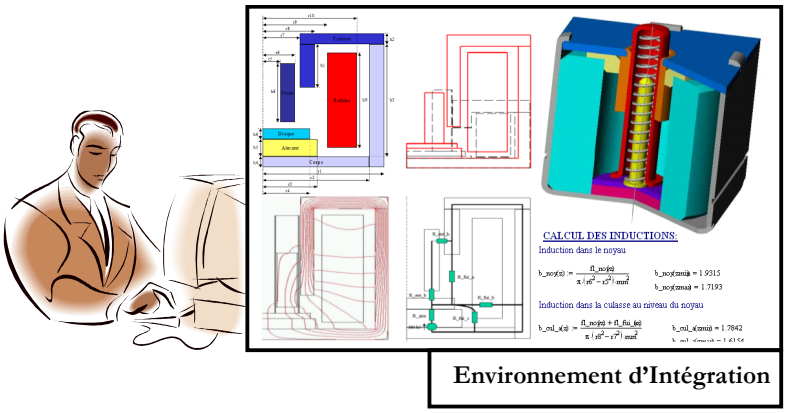


Figure 13 : Pour concevoir un déclencheur électromécanique (cf. Chapitre IV, Partie II), le concepteur pourra mettre en œuvre des modèles analytiques, numériques, mais aussi des géométries paramétrées permettant de visualiser des optimisations, des géométries 3D incluant la conception de la fabrication …

Outre ces différents modèles du produit, il existe des modélisations qui n'ont pas trait au produit. L'objectif d'un modèle de conception n'est donc plus uniquement de représenter un produit, mais aussi de le représenter dans sa durée (*fabrication, utilisation, recyclage*), ou dans son environnement (*conditions d'utilisation, enjeux économiques, …*).

Un environnement d'intégration des modèles permettant de représenter le produit dans sa durée et dans son environnement.

II.C. Un environnement intégrateur de l'acteur concepteur

Un logiciel de CAO n'a pas pour objectif d'être utilisé par un « simple opérateur ». Cette interaction nécessaire entre l'homme et la machine doit conduire à considérer le concepteur comme expert de son domaine et à lui offrir la possibilité d'exploiter ses connaissances. C'est ainsi qu'une bonne CAO, et en particulier un environnement de conception, doit prendre en compte l'interaction entre « acteurs, instruments et savoirs ».

> *Les nombreux propos des concepteurs évoquant la CAO révèlent que ces derniers sont demandeurs d'une souplesse d'utilisation, d'une prise en compte du contexte et de leurs savoirs.*
>
> *Lavoisy [LAV 00]*

Le concepteur doit faire partie du système de CAO.

Les environnements de conception actuels ne considèrent que peu cet objectif, imposant souvent une démarche de conception figée. L'enjeu pour un environnement d'intégration, est de ne pas considérer une simple liste exhaustive d'outils à intégrer, mais d'inclure également le concepteur. Ce dernier doit en effet interagir à la mise en œuvre du processus qu'il aura choisi, mais également durant l'exécution du processus, durant les tâches et entre les tâches de conception.

III. Conclusions

Les différents enjeux que nous avons évoqués dans cette partie cherchent à répondre aux besoins de la CAO de demain. L'environnement d'intégration des outils de CAO nous semble être le meilleur choix pour l'avenir, apportant une dimension supplémentaire aux outils de CAO.

Malgré tout, les environnements actuels n'intègrent pas totalement les enjeux présentés et, en particulier, les contraintes de complexité, de dynamique et d'imprévisibilité. C'est ce que cherche à apporter la solution que nous proposons et que nous allons décrire dans la suite de ce chapitre.

PARTIE II :

UN ENVIRONNEMENT ADAPTATIF A COMPOSANTS

I. Contexte de travail

Ce travail de thèse s'inscrit dans un contexte particulier, puisque depuis plusieurs années, l'équipe « Conception et Dimensionnement Intégrés » (*CDI*) du LEG[7] travaille à la mise en place d'architectures informatiques basées sur le principe des composants logiciels. Nous allons tout d'abord détailler ce principe, puis nous verrons comment il est actuellement utilisé dans notre contexte de travail.

I.A. Les composants logiciels

I.A.1. Définition

Le composant s'est vu utilisé depuis les années 90 dans le domaine de l'ingénierie informatique, dans ce qu'on appelle les **composants logiciels** [SZY 98][MAU 00][BARB 02]. De la même façon que ce paradigme est utilisé pour concevoir des circuits électroniques, un logiciel informatique peut être constitué de composants que l'on assemble.

La réutilisation est le mot clé associé à la notion de composants. En ingénierie de programmation, une réutilisation à base de composants propose une vision différente de celle associée à la notion d'objet [SZY 98][MEI 97]. La principale différence vient d'une définition plus claire et plus sûre de ce que peut faire et de ce qu'il requiert pour fonctionner. Un objet peut utiliser des services externes sans avoir à le spécifier, pouvant conduire à des réutilisations non sécurisées (*cf. Figure 14*).

***Composant logiciel** : défini comme une entité autonome de déploiement [SZY 98], qui encapsule des codes informatiques, et décrit par des interfaces ses interactions qu'il autorise avec d'autres composants.*

[7] LEG : Laboratoire d'Electrotechnique de Grenoble

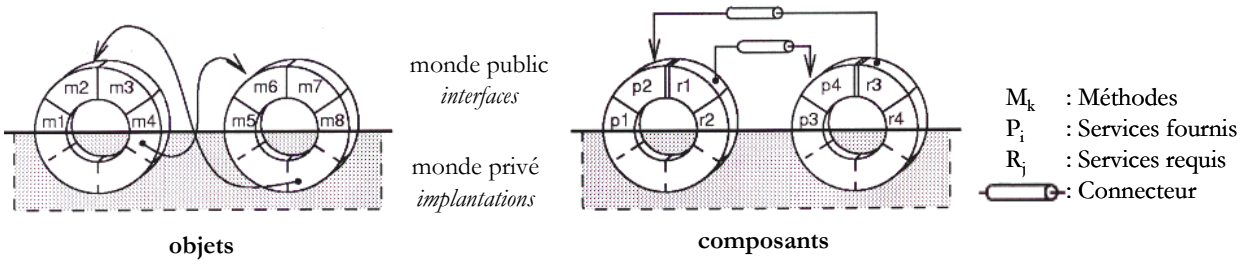
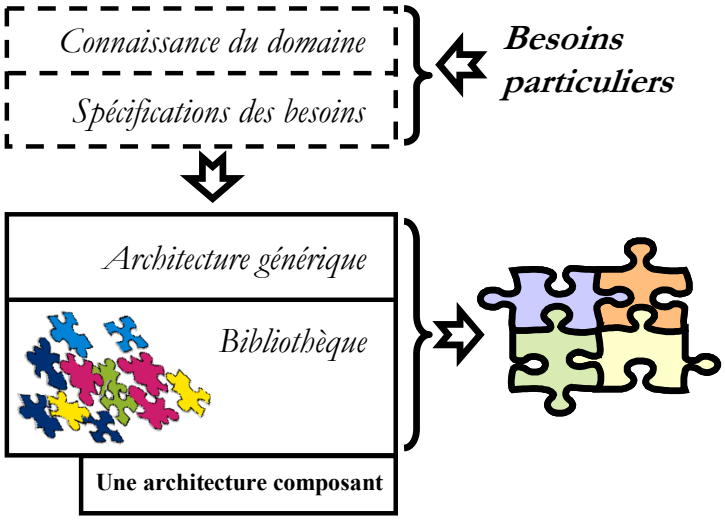


Figure 14 : Une différence entre la programmation objet et la programmation composant : le composant définit explicitement les services qu'il requiert pour fonctionner.

Un composant peut être définit comme une unité de composition [BARB 02]. Il est en effet créé dans un contexte déterminé d'une architecture de composants (*cf. Figure 15*), pour y être connectée à d'autres entités [MEI 97]. Cette **architecture composant** (*component framework*), à laquelle une bibliothèque de composant est associée, permet de créer dynamiquement une application par composition des composants réutilisables.



Architecture composant :
Elle se définit à partir de besoins bien identifiés, appliqués à un domaine particulier. Elle bénéficie alors d'une architecture générique de manipulation de ses composants métiers [MEI 97].

Figure 15 : Définition d'une architecture composant (cf. encart)

I.A.2. Objectifs d'utilisation de composants

L'utilisation de composants informatiques offre un certain nombre d'avantages :

- Avantages attendus en termes de capitalisation / réutilisation :
  - Développements des modèles ou des processus raccourcis grâce à la réutilisation,
  - Réutilisation de codes robustes (*déjà testés, garantissant des résultats, …*).
- Avantages attendus en termes de composition :
  - Construction de systèmes par assemblage de composants disponibles sur *étagère*,
  - Amélioration de la modularité, développement plus abstrait (*fonctionnel*), système plus maintenable (*échange de composants*), extensibilité des applications en temps réel (*composition dynamique*),
  - Système portable et distribuable (*grâce à la gestion des communications inter-composants*).

I.A.3. La réutilisation par des composants métiers

Le problème de réutilisation doit se poser sous un angle « métier » [BARB 02], c'est à dire qu'un certain nombre de connaissances métier doivent pouvoir être mises en œuvre et réutilisées grâce à la notion des composants logiciels. Une des limites de l'approche objet concerne la granularité de ces connaissances à réutiliser qui ne cesse de croître, et qu'une approche composant se propose de gérer.

I.B. Les composants : un nouveau paradigme pour la conception

L'utilisation de composants, dans des domaines tels que l'électronique ou la microélectronique constitue aujourd'hui un **paradigme** apportant des avantages similaires à ceux des composants logiciels. De la même façon, la conception en génie électrique doit pouvoir bénéficier des avantages décrits précédemment. Mais la mise en place d'un tel concept transforme la manière de concevoir, imposant ses propres méthodologies de travail.

> ***Paradigme** : méthodes, connaissances, croyances… admises comme efficaces, ou "vraies", de manière raisonnable, c'est à dire sans que des justifications ne soient plus nécessaires, et gouvernant de manière quasi occulte notre manière de penser et d'agir.*

I.B.1. Un patron de travail

Depuis quelques années, les travaux de l'équipe CDI ont abouti à la mise en place d'un patron de travail (*pattern*) à base de composants (*cf. Figure 16*). Ce patron s'appuie sur la spécification d'un composant métier et d'un certain nombre de générateurs et d'utilisateurs de ce composant. Ce patron de travail peut se décliner selon différents composants métiers correspondants à chaque fois à différents besoins.

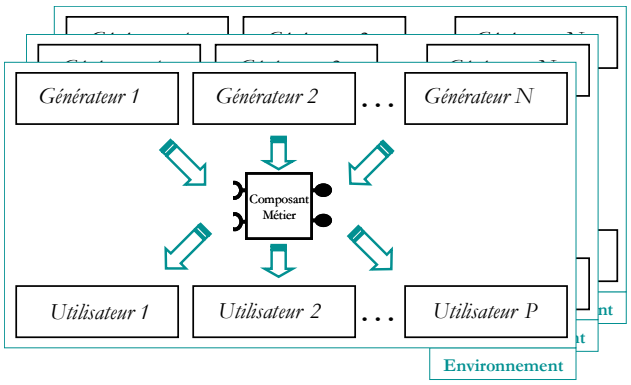


Figure 16 : Patron de travail associé à la méthodologie composant, pouvant se décliner pour différents besoins.

I.B.2. Un composant métier : le modèle de calcul pour le dimensionnement

L'équipe CDI a positionnée un de ses axes de recherche sur la phase de dimensionnement des dispositifs en conception (*actionneurs, micro actionneurs magnétiques, ensembles convertisseur-commande-machine*). Dans ce cadre, une méthodologie a été élaborée lors de la thèse de Frédéric WURTZ [WUR 96a][WUR 96b] que l'on peut résumer ainsi :

- Développement d'un modèle analytique du dispositif à concevoir par le concepteur,
- Création automatique d'un code de calcul associé au modèle, et d'un code de calcul de sensibilité du modèle par dérivation automatique des expressions analytiques.
- Utilisation de ces codes de calcul dans un logiciel de dimensionnement contraint (*optimisation SQP*[8]).

On voit donc ici un générateur (*code de calcul et de sensibilité, cf. pro@DESIGN-generate Figure 18*) et un utilisateur (*logiciel de dimensionnement, cf. pro@DESIGN-optimize Figure 18*). Cette architecture[9] a ensuite bénéficié d'une première spécification de composant de dimensionnement, le COB (*Computational OBject*), durant les travaux de thèse d'Eric ATIENZA [ATI 03]. Ce composant propose les services de calcul et de sensibilité. Il requiert pour cela la valeur des paramètres d'entrée ainsi que leur différentielle, et fournit la valeur des paramètres de sortie et leur différentielle.

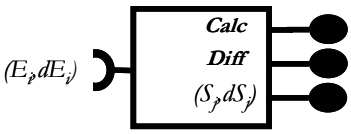
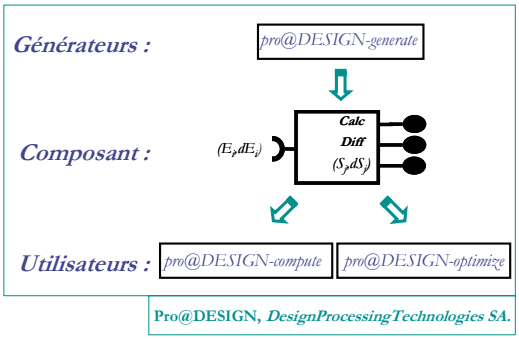


Figure 17 : Données d'entrée et de sortie du composant de calcul pour le dimensionnement COB.



A l'utilisateur de COB permettant de réaliser le dimensionnement peut être ajouté un autre utilisateur permettant de calculer ou tracer les grandeurs de sortie en fonction des grandeurs d'entrée.

Figure 18 : Déclinaison du patron de travail pour l'architecture composant de l'environnement de dimensionnement pro@Design

I.B.3. Evolution des besoins

Afin de répondre à de nouveaux besoins, d'autres générateurs de composants dédiés au dimensionnement ont dus être développés. Un générateur permettant de traiter des simulations temporelles, et plus uniquement des grandeurs paramétrées (*entité→valeur*), a été développé (*Thèse Loig ALLAIN*). De la même manière, de nouveaux utilisateurs de composants ont dus être développés, permettant par exemple de traiter des optimisations à paramètres discrets (*Thèse David MAGOT*).

Ces nouveaux besoins font nécessairement atteindre les limites des spécifications du composant central, montrant que le patron de travail présenté plus haut peut également se décliner pour un même type d'application métier (*ici le dimensionnement*).

[8] SQP : Sequential Quadratic Programming.

[9] Cette architecture a été développée dans un environnement appelé Pro@Design®, qui est aujourd'hui commercialisé par la société Design Processing Technologies SA. : www.designprocessing.com

II. Vers un environnement intégrateur à base de composants

Notre objectif est alors de faire évoluer cette architecture et ce patron de travail pour prendre en compte les difficultés évoquées telles que l'évolution des besoins. Pour cela, nous allons ajouter à l'architecture actuelle quelques concepts nécessaires.

L'environnement, architecturé autour de la notion de composants, met en œuvre deux principaux types d'outils, les *générateurs* et les *utilisateurs* de composants. Dans l'architecture proposée, nous introduirons d'autres *utilisateurs*, c'est pourquoi nous allons renommer les *utilisateurs* actuels en *services (cf. Figure 19)*.

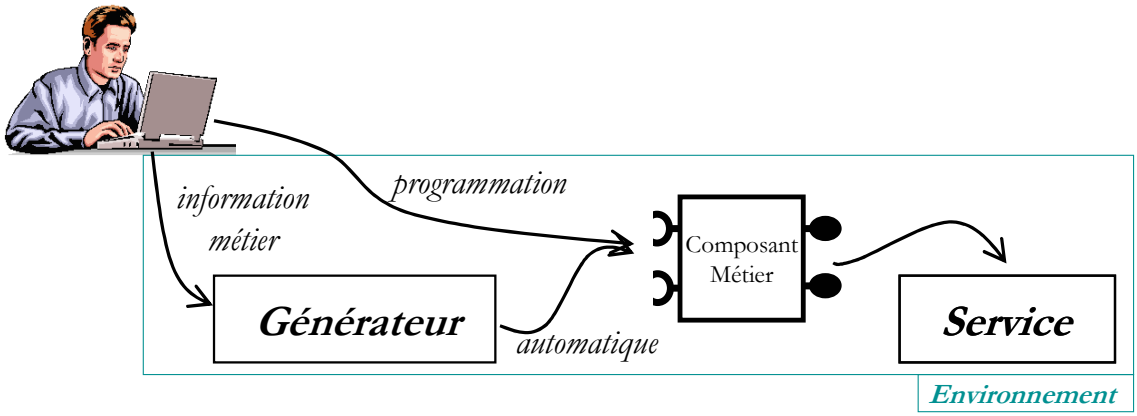


Figure 19 : Architecture composant initiale: un générateur produit un composant utilisé dans un service

II.A. Des concepts nécessaires

Cette architecture est suffisante pour traiter des problèmes de conception bien définis disposants des générateurs et des services associés. Pourtant, elle est insuffisante pour répondre à notre problématique d'un environnement d'intégration des outils du concepteur. Des concepts complémentaires sont donc associés à l'environnement décrit ci-dessus permettant de gérer la diversité potentielle des outils et des besoins du concepteur.

II.A.1. La composition

La composition est la facette fondamentale d'une architecture composant. Elle permettra de gérer la complexité des systèmes en définissant les interactions nécessaires entre les différents composants pour qu'« à eux tous, ils soient plus que la somme de chacun d'eux ». Dans notre architecture, la composition sera récursive *(cf. Figure 20)*, c'est-à-dire qu'elle sera à la fois *utilisatrice* et *génératrice* de composants, elle ne sera pas « terminale » comme le sont les *services*.

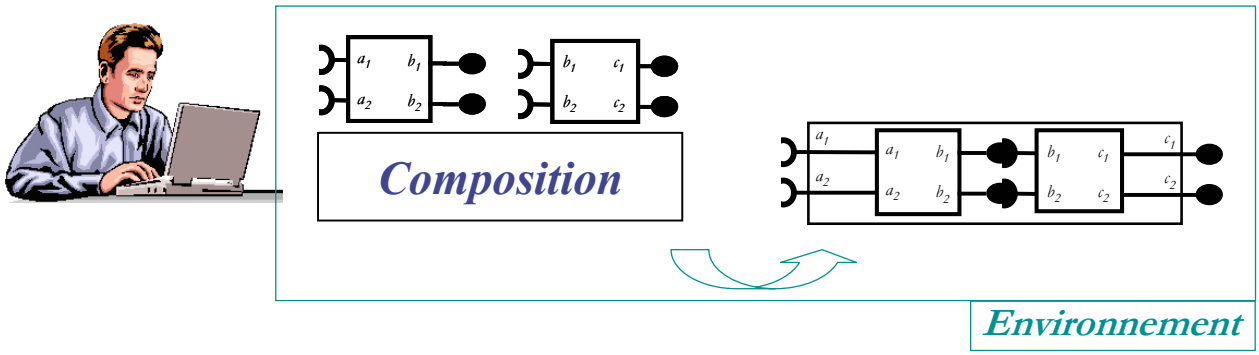


Figure 20 : La composition est à la fois utilisatrice et génératrice de composants autorisant notamment la récursivité.

II.A.2. La Projection

La projection est la notion nécessaire à la réutilisation. En effet, qu'ils s'agissent des services ou des générateurs, ceux-ci utilisent ou génèrent un type de composant particulier. A chaque besoin de conception peut être associée une norme de composant déclinant un patron de travail. Afin d'utiliser un composant dans divers services, il est souvent nécessaire de transformer le composant pour l'adapter à d'autres spécificités. La projection permet en quelque sorte de passer d'une déclinaison du patron de travail (*générateurs, composant, utilisateurs*) à une autre. C'est ainsi que peuvent être gérer les différentes normes de composants au travers des différentes utilisations du patron.

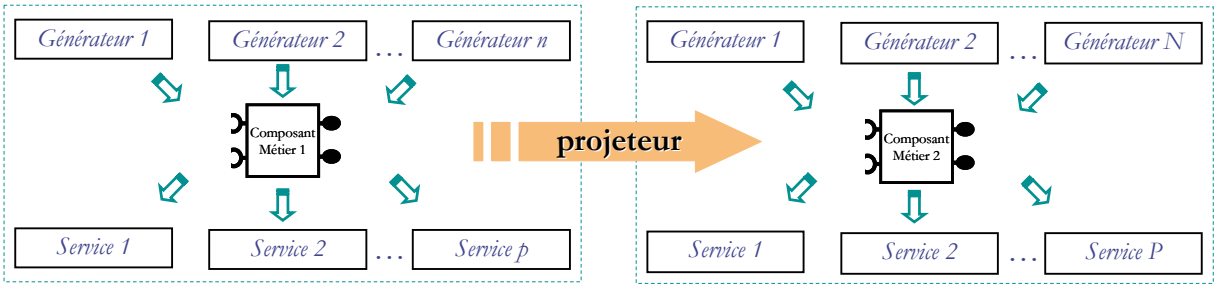


Figure 21 : La projection permet d'exploiter les capacités des composants métiers dans différents patrons

Nous avons montré dans [DEL 00][DEL 02a] que la projection de composants de calcul permettait leur exploitation dans des contextes d'utilisations complémentaires. Nous avons, en particulier, montré comment le concept de projection permettait :

- d'utiliser un modèle de calcul pour développer rapidement une interface homme machine dédiée au pilotage du calcul (*saisie des données, post-processing,* …), sans programmation.
- de déporter un calcul sur différentes machines sans programmation, n'ayant donc pas à gérer les technologies informatiques nécessaires (*cf. Annexe IV : Pilotage de logiciel*).

II.B. Mise en œuvre des concepts pour notre environnement d'intégration

Dans notre travail, nous allons nous intéresser aux différents concepts (*et outils associés*) permettant de mettre en œuvre une telle architecture, afin d'obtenir un environnement d'intégration des outils du concepteur.

II.B.1. Les composants : des boites noires

Notre environnement de conception mettra en œuvre pour cela toutes sortes de composants, encapsulant des outils réalisant diverses fonctionnalités.

Ainsi, plongés dans l'environnement de conception, les outils pourront bénéficier des services disponibles. Et, si l'on considère un dispositif simulé dans un programme informatique utilisé classiquement de manière autonome, celui-ci pourra être intégré à l'environnement et, par exemple, être utilisé par un service de dimensionnement, pour être optimisé.

***Boite noire** : Issu de la cybernétique, ce concept permet d'appréhender un objet sans connaissance nécessaire sur sa constitution interne. Seul les relations entre les entrées et les sorties peuvent être observées, correspondant à son comportement et plus à sa constitution.*

Ces composants seront considérés comme des « **boites noires** », c'est-à-dire que les utilisateurs n'auront *a priori* aucune information sur le contenu de ces composants.

L'environnement n'imposera pas de spécifications particulières pour ses composants, c'est-à-dire que diverses normes pourront être intégrées. Une sémantique minimale sera pourtant privilégiée dans certains cas, celle vérifiant le concept des boites noires, imposant au composant une interface contenant une liste de ports d'entrées et une liste de ports de sortie.

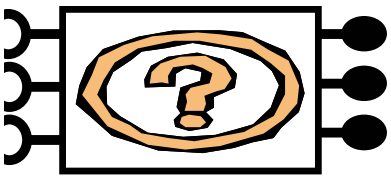


Figure 22 : La boite noire se définit uniquement par une interface réceptacle (entrées nécessaires au fonctionnement) et une interface fournisseur (sorties produites)

II.B.2. Capitalisation des composants

***Boite blanche** : Par opposition à la boite noire, la boite blanche pourra selon le cas être utilisée pour son comportement liant les entrées aux sorties, ou bien être exploitée par la connaissance de sa constitution interne.*

Par opposition à cette notion de boite noire, le concept de « **boite blanche** » peut être défini (*thèse en cours, Loig Allain, équipe CDI*). Elle est une forme dérivée de la notion de composant, dans laquelle celui-ci offre un moyen d'interagir sur la nature de son

contenu. Cette possibilité correspond à un gage de capitalisation et de réutilisation contextualisée, en considérant qu'il peut être nécessaire de refondre un modèle composé afin de réaliser un « couplage fort » entre ses constituants, sous peine de ne pouvoir garantir son fonctionnement.

La capitalisation de boites noires correspond, quant à elle, à la possibilité de réutiliser une mise en œuvre informatique d'un traitement bien spécifié (*à une requête déterminée, le composant fournit un résultat déterminé*). Ainsi, le composant boite noire correspond, par exemple, à une solution pour capitaliser des modèles de calcul de dispositifs électrotechniques, intégrant une implémentation du calcul dans un contexte d'utilisation bien défini.

La boite peut parfois être considérée plus ou moins grise, selon qu'elle possède ou non une information décrivant son fonctionnement interne. Malgré cela, nous n'envisageons pas pouvoir intervenir, comme c'est le cas des boites blanches, sur la constitution d'un composant boite noire. Dans le cadre de la capitalisation, le concepteur doit donc avoir conscience du modèle qu'il met en œuvre par l'utilisation des composants boite noire.

II.B.3. La composition

L'utilisation du paradigme des composants et notamment le principe de composition, doit permettre au concepteur de mettre en relation ses outils. Or, une des gestions possibles de la complexité des systèmes à concevoir, consiste à créer une synergie entre les outils disjoints du concepteur. En effet, la vision composant cherche, à notre avis, à répondre à la pensée complexe développée dans le cadre de l'épistémologie constructiviste :

> *L'ambition de la pensée complexe est de rendre compte des articulations entre domaines disciplinaires qui sont brisés par la pensée disjonctive, […] elle porte en son principe la reconnaissance des liens entre les entités […]*
>
> *Edgar Morin, Introduction à la pensée complexe [MOR 00], avant propos.*

Il s'agit de répondre aux insuffisances de la pensée « simplifiantes », ou paradigme de simplification, et en particulier au principe de disjonction/réduction, par un paradigme de distinction/conjonction permettant de distinguer sans disjoindre, et d'associer sans identifier ou réduire [MOR 00]. Ainsi, le paradigme des composants fuit les extrémismes et ne cherche ni à ne considérer que le tout (*holisme*), ni à simplifier jusqu'à disjoindre (*réductionnisme*).

La composition a en effet pour objectif de créer des liens, de mettre en relation des composants disjoints. Or la nature imprévisible des composants, ainsi que celle des relations définies, nous permettra d'envisager plusieurs applications à la composition. Nous détaillerons deux applications particulières dans le cadre de ces travaux de thèse, celui de la composition de modèles et celui de la description de processus.

Nous considérerons ainsi des **compositions de modèles** de calcul ou de modèles de dimensionnement qui mettent en œuvre un modèle global (*électromagnétique, thermique, économique, contraintes de fabrication,* …) (*cf. Figure 23*). Les relations entre composants permettront alors de définir un couplage entre ces modèles. C'est ainsi qu'une partie de la complexité des systèmes à concevoir pourra être prise en compte.

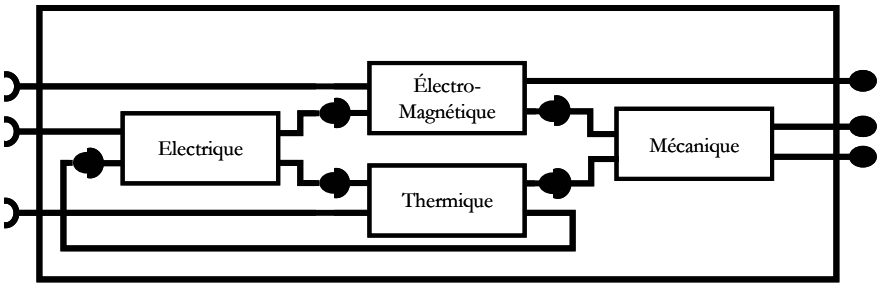


Figure 23 : Une composition de modèle de calcul, intégrant la complexité multiphysique des dispositifs à concevoir

Nous considérerons également des **descriptions de processus** qui mettent en œuvre le séquencement de tâches d'un processus de conception (*workflow [STO 01]*). Les relations entre composants permettront alors d'échanger un « flux de conception » (*des modèles de structure, un dimensionnement optimal,* …), lequel évoluera au travers des diverses tâches de conception. Une telle gestion du processus de conception permettra notamment de gérer sa nature itérative (*cf. Figure 24*).

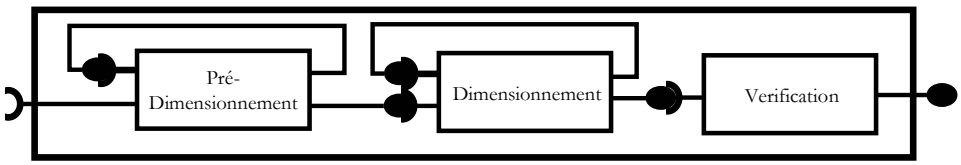


Figure 24 : Une description de processus de conception qui fait intervenir une étape de pré-dimensionnement sur laquelle il est possible d'itérer, puis une étape de dimensionnement final.

III. Vers un environnement non prescripteur et adaptatif

Nous estimons qu'une partie de la complexité des systèmes à concevoir peut être résolue grâce à la capacité d'intégration de notre environnement et son architecture à base de composants. Les concepts que nous avons mis en place pour notre architecture composant ne suffiront pas seuls à considérer la nature imprévisible de la conception.

III.A. Un support non prescripteur du processus de conception

Nous avons évoqué la possibilité, par la composition de composants, de décrire le séquencement des tâches d'un processus de conception. Cette solution de « workflow » adaptatif est indispensable pour gérer la dynamique et l'imprévisibilité du processus de conception. Les workflow standards sont généralement issus de standardisations de procédures qui correspondent à la formalisation du processus. En raison de l'imprévisibilité du processus, ces formalisations sont souvent ou trop générales, pour aider le concepteur dans sa tâche quotidienne, ou trop spécifiques, pour représenter l'ossature d'un environnement de conception.

Qu'il s'agisse de travailler sur le processus ou sur les modèles que le concepteur met en œuvre, une notion fondamentale, que nous retenons dans notre solution d'environnement prenant en compte le caractère imprévisible de la conception, est de savoir s'adapter et réagir. Un support informatique à la conception doit donc :

➥ Réagir (*Système d'information adaptatif*),

Et non :

~~➥ Prévoir (*système d'information exhaustif*)~~

Ne pas prévoir tous les cas possibles mais savoir s'adapter et réagir.

Ne pas imposer un processus particulier mais simplement des méthodes ou plutôt de « bonnes pratiques » associées à des processus modifiables. Ainsi, le concepteur pourra, à sa guise, jouer d'allers et retours pour affiner sa connaissance du problème et préciser sa solution de conception.

Les techniques de composition que nous mettrons en œuvre dans l'environnement permettront au concepteur de modifier l'organisation de sa conception, et ainsi de s'adapter à un processus changeant. Mais ces capacités ne sont pas suffisantes pour répondre aux besoins exprimés, car il est également nécessaire de considérer la dynamique des outils à intégrer dans l'environnement.

III.B. Un support s'adaptant aux outils du concepteur

III.B.1. Nécessité d'un support adaptatif

L'imprévisibilité des outils nécessaires au processus met souvent le concepteur en position inconfortable, soit par rapport à des délais à tenir, soit par rapport à un cahier des charges qu'il n'arrive pas à satisfaire. En effet, le concepteur, étant soumis à l'insuffisance de processus statiques, va devoir mettre en place des « bouts de ficelle », seul moyen pour atteindre ses objectifs de conception. S'il n'est pas à même de mettre au point une alternative aux insuffisances de son processus, le concepteur devra réajuster ses objectifs par rapport aux clients. Ces deux solutions ne sont plus acceptables aujourd'hui, et le concepteur doit être en mesure de mettre au point de meilleures solutions grâce à un environnement capable de s'adapter.

Il est notamment fréquent de disposer de codes « maison » qui permettent de prendre en compte des besoins spécifiques. L'environnement de conception doit donc être ouvert et intégrateur, mais surtout avoir la capacité de gérer l'imprévisibilité quant à l'outil à intégrer, pour lequel on ne peut définir à l'avance les caractéristiques.

III.B.2. Des générateurs de composants : les assistants d'intégration

Nous supposerons *a priori* que les outils du concepteur ne sont pas définis dans un formalisme composant tel que nous l'avons présenté. Il nous sera alors nécessaire de les définir dans de tels formalismes, afin de les intégrer dans l'environnement en tant que composants. Nous pouvons considérer un travail long et fastidieux permettant l'intégration d'un certain nombre d'outils. Mais de telles intégrations, pour justifier leur coût, cherchent à prévoir tous les cas possibles d'utilisation de l'outil pour garantir au concepteur qu'il n'aura pas à coder quoi que ce soit.

Malheureusement, une telle solution est contraire au caractère imprévisible que nous souhaitons gérer en nous adaptant. Notre environnement de conception ne sera pas développé dans cette optique d'intégration exhaustive des outils de CAO existants. Le concepteur devra alors être à même d'intégrer lui même les outils avec lesquels il souhaite mettre en place son processus.

Des assistants d'intégration pour gérer rapidement et simplement les besoins nouveaux.

Considérons l'expérience réalisée à VirginiaTech [CHE 01][WU 02], consistant à mettre en œuvre un environnement de conception de modules intégrés d'électronique de puissance

(*IPEM*). Leur solution se base sur l'intégration de divers outils (*I-DEAS*[10], *Maxwell*[11] et *Saber*[12]) dans l'environnement d'optimisation iSight[13] (*cf. Figure* 25). Le mode d'intégration d'iSight s'appuie sur la capacité des outils à être pilotés par un fichier de commandes. Ce dernier étant propre à une topologie donnée, nécessite sa redéfinition pour chaque nouvelle topologie. Le constat a été fait dans cette expérience que, « *l'effort de re-programmation est clairement non trivial, embarrassant, et inhibe toute exploration rapide d'une conception nouvelle et innovante* ». [WU 02],

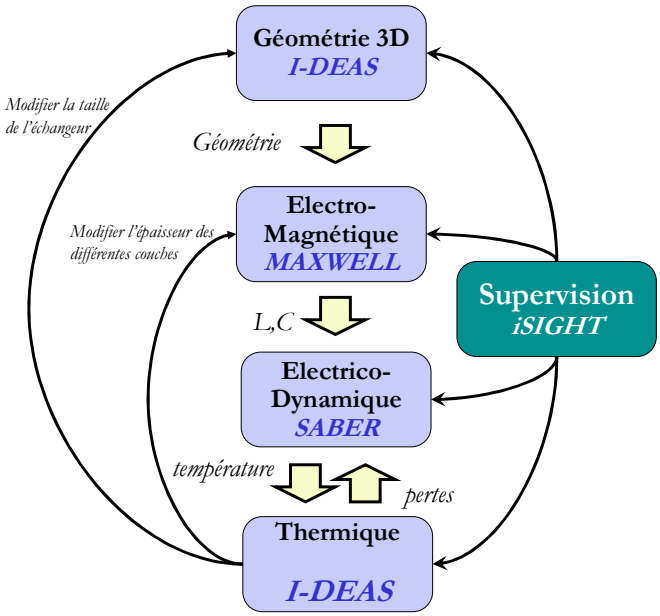


Figure 25 : Processus de dimensionnement d'une structure de modules intégrés d'électronique de puissance (IPEM) mettant en œuvre différents outils et l'environnement d'optimisation iSight.

Nous avons donc choisi de mettre en œuvre une méthodologie basée sur des assistants d'intégration qui devront permettre une intégration rapide, définissant l'outil par un système d'information adapté au concepteur et à l'application de conception (*cf. Figure 26*). Ces assistants d'intégration jouent alors le rôle de générateurs de composants dans le patron proposé avec l'architecture composant. Pouvant être utilisés par le concepteur dès qu'il en ressent le besoin, ces assistants d'intégration nous permettront de gérer l'imprévisibilité des divers contextes d'utilisation des outils, et l'imprévisibilité de la nature même de ces outils.

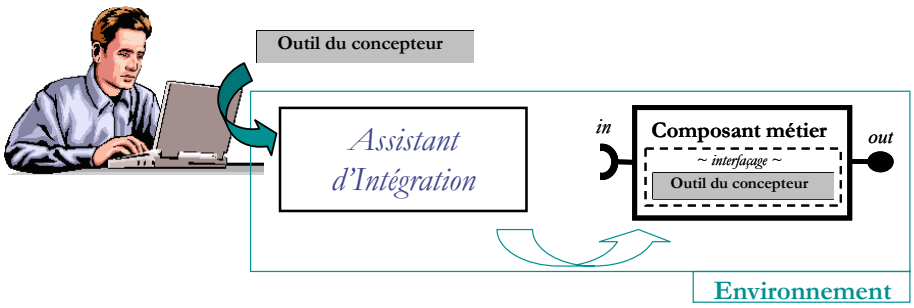


Figure 26 : Un générateur de composants : l'assistant d'intégration, permettant de gérer l'imprévisibilité du contexte d'utilisation d'un outil, et l'imprévisibilité quant à l'outil à intégrer.

[10] I-DEAS, Structural Dynamics Reasearch Corporation, Milford, OH, www.sdrc.com

[11] Maxwell, Ansoft corporation, Pittsburgh, PA, www.ansoft.com

[12] Saber, Analogy Inc. Beaverton, OR, www.analogy.com

[13] iSight, Engineous Software, Cary, NC, www.engineous.com

IV. Les caractéristiques de notre environnement

IV.A. Les critères de gestion de la complexité et de l'imprévisibilité

A partir de l'architecture composant et des capacités d'intégration dynamique proposées, nous pouvons définir les critères que devra remplir notre environnement de conception afin de répondre aux enjeux que nous nous sommes donnés :

- **Intégrabilité** : Supporter l'intégration par l'encapsulation de systèmes existants ou futurs, d'applications commerciales ou propriétaires, en définissant leurs capacités de connexions.
- **Flexibilité** : Offrir la possibilité de remplacer ou modifier des applications de l'environnement avec un minimum d'impact sur les autres applications dejà disponibles.
- **Capitalisation/Réutilisation**: Répondre à de nouveaux besoins par de nouvelles combinaisons d'applications (*ou modèles*) capitalisées dans des composants.
- **Reconfigurabilité** : Des applications existantes doivent pouvoir être reconfigurées très rapidement pour supporter les besoins changeants.
- **Adaptabilité** : De nouvelles applications doivent pouvoir être rapidement intégrées et assemblées pour répondre à de nouveaux besoins.

IV.B. La place des acteurs de la CAO

IV.B.1. Le concepteur est maître de son outil

Un critère supplémentaire nous paraît également important pour la définition de notre environnement, celui de la place de l'acteur dans l'environnement. Comme nous l'avons définit dans la partie précédente, nous pensons que la place du concepteur est primordiale. Il faut que les outils qu'il met en œuvre dans son processus soient le fruit de ses choix et non d'une prescription venant de l'environnement. Il doit également pouvoir définir lui même le processus qu'il souhaite suivre et en réajuster les moindres détails de manière dynamique.

IV.B.2. Accessibilité de l'environnement à plusieurs niveaux de compétences

Il est important de faire apparaître un acteur hybride, à mi chemin entre les deux disciplines du génie électrique et de l'informatique, aujourd'hui intimement liées, et de repositionner son rôle par rapport au concepteur « classique ».

- Concepteur de produits : utilise la CAO pour mettre en place son atelier de conception.
- Concepteur de produits avec compétences informatiques : à partir des limites mises en évidence par l'activité de conception, il doit pouvoir « bricoler » des solutions pour palier ces limites

Ayant défini le concepteur comme un acteur majeur de l'environnement de conception, nous estimons qu'un tel environnement doit pouvoir tirer parti des concepteurs double compétences. C'est ainsi que l'utilisation de l'environnement de conception pourra se décliner selon deux niveaux :

- L'utilisateur (*le concepteur « classique »)* : met en œuvre ses processus en exploitant les capacités d'intégration de l'environnement.
- L'administrateur (*le concepteur à compétences informatiques*) : doit pouvoir offrir au concepteur les facilités pour intégrer de nouvelles catégories d'outils et enrichir l'environnement par de nouveaux services.

V. Conclusions

Ce chapitre nous a permis d'introduire différents types d'environnements pour la CAO. Nous avons présenté les raisons qui nous poussent à penser qu'un environnement d'intégration, de part son ouverture, permet de gérer des aspects de complexité et d'imprévisibilité de la conception.

La deuxième partie de ce chapitre nous a permis de présenter le contexte de travail dans lequel cette thèse s'est déroulée. Nous avons ainsi développé différents concepts basés sur les composants logiciels déclinés en composants métiers. C'est en particulier avec les concepts d'assistants d'intégration et de composition, qui vont être détaillés dans le chapitre qui suit, que nous allons concrètement pouvoir répondre aux critères qui viennent d'être définis.

Chapitre 3

METHODOLOGIES DE L'ENVIRONNEMENT

Chapitre 3

METHODOLOGIES DE L'ENVIRONNEMENT

Après avoir détaillé les concepts d'une architecture composant et les directions que nous avons choisi de prendre pour gérer la complexité et l'imprévisibilité en conception, nous allons détailler les deux concepts nécessaires à notre environnement.

Le premier concerne un générateur de composants particuliers, l'assistant d'intégration, qui permet de générer les composants de l'environnement à partir des outils du concepteur. Nous verrons en particulier une norme de composant dédiée à cette intégration : *le composant abstrait*.

Le deuxième s'intéresse à la composition de composants, permettant de créer une synergie entre ces composants. Nous verrons, en particulier, comment un outil générique qui s'appuie sur le concept de boite noire permet de gérer la composition de différentes normes de composants. Deux applications de cet outil seront détaillées, celle de la composition de composants de calcul pour le dimensionnement et celle de la description de processus de conception comme un enchaînement de tâches (*workflow*).

PARTIE I :

METHODOLOGIE D'INTEGRATION

Nous allons présenter une méthodologie permettant de mettre en œuvre le principe des générateurs de composants à partir des outils du concepteur. L'intégration doit permettre de répondre à la complexité en offrant à ces outils la capacité d'être utilisés dans un environnement de conception, dans lequel peuvent être disponibles un grand nombre de services. L'imprévisibilité doit être gérée par le caractère dynamique de l'intégration, en permettant de reconfigurer ou d'ajouter rapidement des applications à l'environnement, ce qui répond ainsi à l'évolution des besoins durant le processus de conception.

I. Le processus d'intégration

I.A. Deux étapes d'intégration

Lorsque le concepteur souhaite utiliser un logiciel dans un des services disponibles de l'environnement (*service d'optimisation pour des composants de calcul,* …), il lui suffit de suivre la démarche d'intégration proposée (*cf. Figure 27*).

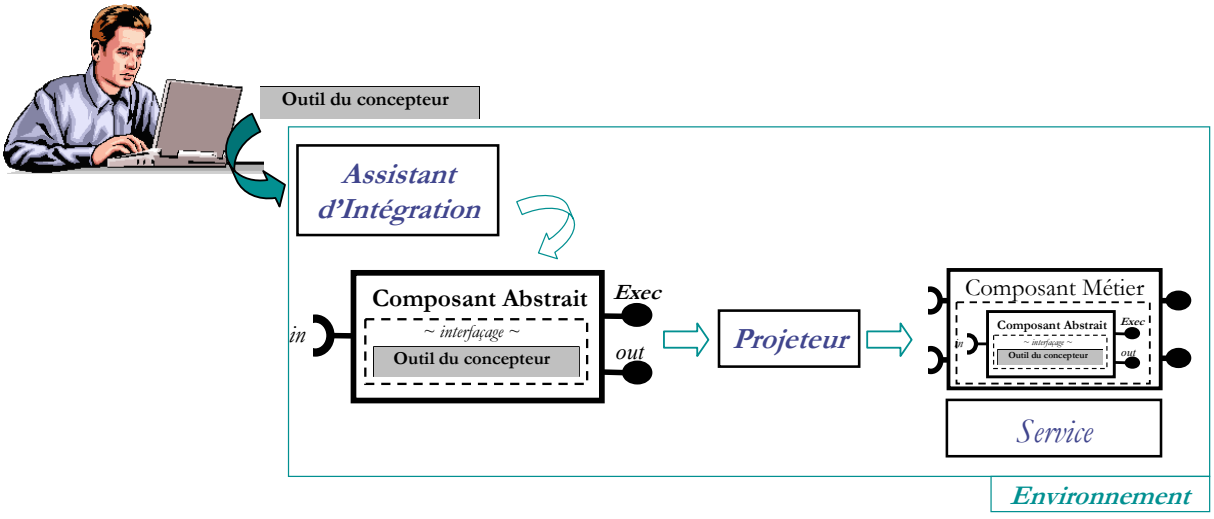


Figure 27 : Méthodologie d'intégration : assistant d'intégration + projection

Cette démarche s'appuie sur deux étapes successives, une première permettant de générer, à partir de l'outil, un composant dédié à l'intégration (*encapsulation de l'outil*) que nous avons baptisé « ABC » pour « ABstract Component » (*composant abstrait*). Ensuite, ce composant

doit être projeté vers la norme de composant spécifique au service que le concepteur souhaite utiliser (*le composant métier*).

Dans la pratique, d'autres étapes peuvent intervenir, telles que des étapes intermédiaires de composition, mais dans tous les cas, deux principes que nous détaillerons plus loin, restent présents. Le principe d'abstraction/spécification encapsule l'outil dans un composant abstrait puis le projette vers une norme de composant spécifique. Le principe suivant définit la dépendance des outils d'encapsulation aux mécanismes de pilotage de l'outil à intégrer, ainsi que la dépendance des outils de projections aux spécifications du composant dans lequel se projeter.

I.B. Itération du processus d'intégration

La capacité d'intégration dynamique définie par ce processus pourra alors être exploitée pour gérer l'imprévisible. Durant le processus de conception, le concepteur devra en effet être à même de redéfinir l'intégration de ses logiciels. Les outils d'encapsulation et de projection devront alors pouvoir être re-exécutés de manière semi-automatique, et ne demander au concepteur que les modifications à apporter aux caractéristiques d'intégration. Ainsi, l'évolution des informations définie dans l'outil pourra être gérée et le processus de conception pourra évoluer.

II. Les briques de bases

Afin de détailler cette méthodologie, nous allons en détailler les étapes en introduisant les concepts suivants :

- Nous allons tout d'abord définir le « composant abstrait ».
- Afin de piloter le logiciel à intégrer, seront alors introduits les outils d'encapsulation basés sur les accès disponibles des logiciels, et une déclaration descriptive de l'information à intégrer.
- Enfin, sera décrite la phase de spécification (*contextualisation*) qui permet de projeter le composant abstrait vers une norme de composant spécifique pour être ensuite utilisé par les services associés à cette norme.

Mais avant cela, nous allons décrire ce que nous voulons intégrer d'un logiciel. Pour cela, nous allons d'abord devoir faire le choix d'une intégration des fonctionnalités du logiciel ou d'une intégration contextuelle, s'appuyant sur le dispositif étudié.

II.A. Intégration d'application vs. Intégration logicielle

II.A.1. L'intégration logicielle

Notre méthodologie s'appuie sur l'intégration des logiciels dans des composants. Or, généralement, la mise en place d'un logiciel dans un composant est longue et fastidieuse [BAS 00]. Elle est réalisée une fois par logiciel, et rend disponible toutes les fonctionnalités du logiciel par l'interface du composant. Cette intégration, que nous appellerons « intégration logicielle », a comme avantage d'être faite une fois pour toute. Par contre, elle a deux inconvénients majeurs : le temps de réalisation est significatif, et le fait qu'elle n'offre qu'une vue fonctionnelle du logiciel qui doit ensuite être contextualisée selon l'utilisation.

Comme nous le voyons sur la *Figure 28*, une telle intégration implique pour l'environnement la nécessité de disposer de divers mécanismes de gestion de ces composants logiciels. Ces mécanismes peuvent être la **configuration** (*pour choisir l'application à traiter,* …), la **persistance** pour sauvegarder l'état du composant une fois spécifié, ainsi qu'un mécanisme de **connexion** aux données et aux services disponibles dans l'environnement. C'est cette vision qui est adoptée par la plupart des environnements logiciels à base de composants génériques[14].

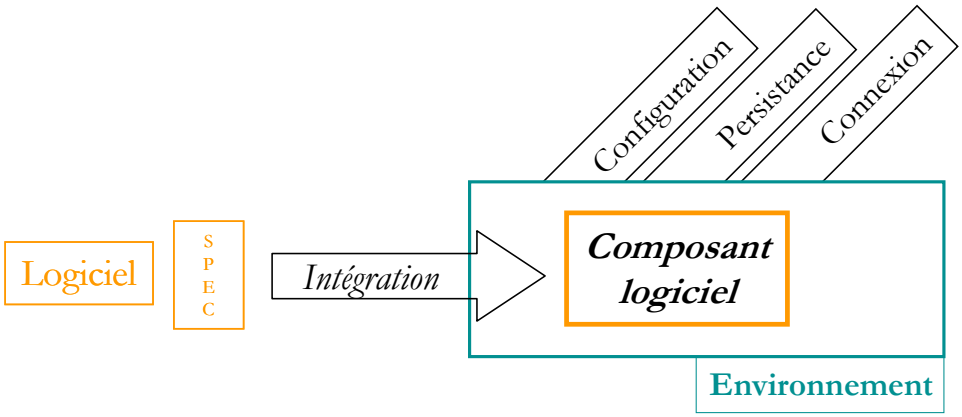


Figure 28 : Principe d'une intégration logicielle qui impose à l'environnement 3 services de gestion des composants : la configuration, la persistance et la connexion

[14] Les composants Enterprise Java Beans (EJB) de Sun Microsystems, doivent par exemple pouvoir compter sur un environnement complexe leur permettant de se configurer et de mémoriser leur état.

II.A.2. L'intégration d'application ou intégration contextuelle

Nous envisagerons un autre type d'intégration, « l'intégration d'application » ou « intégration contextuelle ». Le terme « application » est à prendre au sens du dispositif étudié. Il s'agit, non plus d'intégrer toutes les fonctionnalités du logiciel, mais uniquement celles nécessaires, ainsi que les données spécifiques à l'application traitée. Ainsi, nous disposerons d'un composant facile à utiliser car spécifique à nos besoins. Bien sûr, il est nécessaire de recréer un composant pour chaque application traitée, mais cela correspond bien à notre solution de gestion de l'imprévisible. De plus, cette particularité va nous permettre d'exploiter des moyens automatiques d'encapsulation réduisant considérablement les temps d'intégration et autorisant notre méthodologie d'intégration dynamique. Sur la *Figure 29*, nous pouvons voir que le mécanisme d'intégration de l'application soulage l'environnement de services de configuration et de persistance des composants.

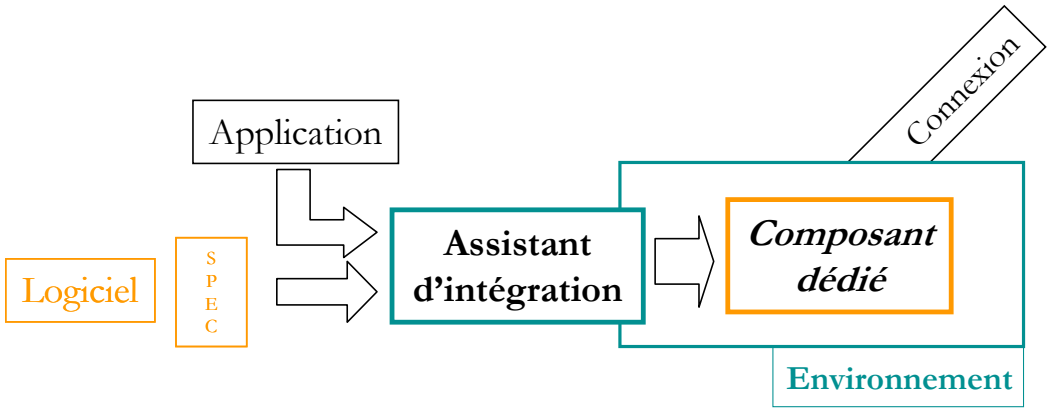


Figure 29 : Principe d'une intégration d'application (ou contextuelle), qui n'impose à l'environnement qu'un service de gestion des composants : la connexion.

Cette intégration contextuelle des fonctionnalités du logiciel, va autoriser le développement de notre méthodologie, ce à quoi nous ne serions pas parvenus dans le cadre d'une intégration logicielle classique.

II.B. Composant abstrait

Cette intégration de l'application se fera, tout d'abord, dans le composant abstrait que nous avons défini pour répondre aux besoins de l'intégration. Comme nous le verrons dans la partie suivante de ce chapitre, ce composant permet de décliner un patron de travail (*générateurs / utilisateurs*) dédié à la description de processus. Pourtant, dans la méthodologie d'intégration, il n'est qu'un intermédiaire pour parvenir jusqu'au composant métier qui sera utilisé dans les services de l'environnement. Il sera utilisé pour décrire ce que le concepteur souhaite utiliser du logiciel (*information et fonctionnalités*). Ne disposant quasiment d'aucune **sémantique** sur l'information qu'il embarque, cette description devra être suffisamment explicite pour que le concepteur puisse la spécifier dans l'étape suivante de projection, et la faire correspondre à une sémantique utilisable par un système informatique. Son niveau d'abstraction devra être suffisant pour transporter des types d'informations divers.

> ***Sémantique de l'information** :*
> *C'est le sens qui est attribué à l'information.*
> *Si un contenu sémantique de l'information est définit, celle-ci peut être interprété, transformé...Si il ne l'est pas, on peut malgré tout, échanger cette information sans en comprendre le sens.*

Pour ce faire, nous avons choisi, comme entité d'échange fondamentale, le fichier informatique (*cf. Figure 30*). Ainsi, le traitement du contenu du fichier peut être particularisé selon sa nature (*un fichier texte fait de caractères ASCII peut être visualisé/édité).*

Figure 30 : Le composant abstrait : requiert une entrée spécifiée par une adresse de fichier (URL), et créé des informations de sortie dans un fichier après qu'une méthode ait été appelée.

Afin de gérer une information évolutive, nous utiliserons dans cette méthodologie d'intégration le standard XML[15], proposé et maintenu par le W3C (*world wide web consortium*). Cette technologie permet de décrire des données conformes à une structure qui peut évoluer facilement. XML est donc une solution à l'adaptation, que nous utilisons largement dans les différents outils développés.

[15] XML (eXtended Markup Language) : www .w3c.org/xml

La *Figure 31* est un exemple de description de quelques données d'un projet de conception définit par le concepteur. Celui-ci devra, à partir de ces données, faire correspondre des informations parmi celles disponibles dans l'outil à intégrer (*phase de déclaration de correspondances*).

```
<Desing name="circuit1" author="benoit">
 <EquationModel>
    <Equation eq="U=R*I"/>
    <Equation eq="R=rho*l/s"/>
    ...
 </EquationModel>
 <DataToIntegratedTool>
    <Parameter name="V" value="230" unit="V" />
    <Parameter name="L" value="0.167" unit="H" />
    <Parameter name="R" value="32.768" unit="Ω" />
    ...
 </DataToIntegratedTool>
 <DataFromIntegratedTool>
    <Parameter name="I" value="1.232" unit="A" />
    ...
 </DataFromIntegratedTool>
 ...
</Desing>
```

Figure 31 : Exemple de description XML de données d'un projet de conception multi-outils.

L'avantage d'avoir à traiter d'un fichier XML est de faire apparaître aux concepteurs une vue hiérarchisée et lisible de ses données, sans qu'il soit nécessaire que l'outil traitant ce fichier connaisse la signification des informations contenues (*cf encart*). Cela permet notamment de modifier la sémantique associée aux données de ce fichier XML sans pour autant modifier l'outil qui traite de ce fichier. C'est ce que nous allons voir dans le paragraphe suivant traitant de l'encapsulation déclarative.

Une description XML du projet de conception permet de plus de bénéficier d'autres outils adaptés à ce standard. Un système de gestion de versions peut par exemple s'avérer pertinent [DEL 02b], offrant au concepteur la sauvegarde des *ajouts / suppression / modification* d'éléments de la structure XML.

Sémantique et norme XML :
3 exemples pour comprendre l'intérêt de XML.

Premier exemple :
"D0CF11E0A1B11AE1…" ceci correspond à l'entête du fichier Microsoft Word de ce rapport de thèse (en hexadécimal). Pour le logiciel, c'est la marque d'un fichier lui appartenant. Pour nous, cela n'a aucun sens et ne représente rien. Malgré l'incompréhension du sens, ce bout de code a pu être écrit dans ce rapport.

Deuxième exemple :
Un logiciel de traitement de texte basique affiche à l'écran le texte que l'on frappe au clavier. Malgré tout, il n'est pas capable d'en comprendre le sens, car il ne connaît pas le sens attaché à chacun des mots, la formation grammaticale d'une phrase…

Troisième exemple :

```
<Balise nom="Balise XML" objectif="donner du sens">
      <Enfant id="1" objectif="structurer">
      <Enfant id="2" objectif="structurer"/>
</Balise>
```

Le format texte de XML fournit une information lisible et compréhensible par l'homme. La structure arborescente et la définition d'éléments et d'attributs offrent une information compréhensible par l'ordinateur.

II.C. Encapsulation déclarative (ou descriptive)

La méthodologie d'intégration que nous mettons en place ici, s'appuie sur cette étape essentielle d'encapsulation déclarative qui demande au concepteur de décrire ce qu'il souhaite utiliser du logiciel. Pour cela, l'assistant d'intégration doit proposer une vision de ce que l'outil est capable de traiter pour l'application en cours de conception.

II.C.1. Introspection de l'outil

Ce que nous appelons mécanisme d'introspection est un moyen permettant d'interroger le logiciel sur les données de l'application en cours et les services disponibles afin de traiter chaque cas de conception en fonction des moyens d'accès identifiés. Ils peuvent aller de la simple mise à disposition de fichier texte par le logiciel jusqu'aux outils autorisant une introspection d'un code informatique pour en ressortir les services et données disponibles.

Le problème technique qui se pose est donc le moyen d'accéder au pilotage du logiciel. Les principes de pilotages de logiciels sont détaillés dans l'*Annexe IV*, mais nous pouvons recenser ici quels peuvent être les assistants en fonctions de quelques-uns des points d'entrée les plus couramment rencontrés :

- Fichier de données au format texte propriétaire ou standardisé.
- Base de donnée (*accessible par requêtes standardisées de type SQL*).
- Script de commande propriétaire ou standardisé (*série d'instructions et de données*).
- Interface de programmation du logiciel (*liste d'objets que l'on peut appeler par un programme informatique*), permettant par exemple de lister les données de l'application en cours.
- Composant avec capacité d'introspection (*possédant des interfaces standardisées d'introspection*)
- Code source (*nécessite d'analyser le code, puis de recompiler le logiciel…*)

II.C.2. Déclaration de correspondances

L'assistant d'intégration a pour principal objectif de réaliser une traduction entre la vision des informations de conception décrites par le concepteur (*fichier XML*), en une forme spécifique que l'outil à intégrer puisse reconnaître et exploiter (*fichier de commande, interface de pilotage,* …). Cette information de traduction est définie par la phase de déclaration de correspondances.

A partir d'une spécification XML de ce que souhaite traiter le concepteur, une traduction peut être réalisée avec les informations disponibles dans l'outil à intégrer (*cf. Figure 32*). Il faut donc, au préalable, que le concepteur définisse clairement dans un fichier XML quelles informations seront nécessaires à l'utilisation du logiciel. Le fait de n'imposer aucun format

aux données de conception permettra à cette description d'être enrichie par d'autres applications sans nuire au bon fonctionnement de l'intégration ; ou à cette intégration de bénéficier de nouvelles données à intégrer sans remettre en cause les anciennes.

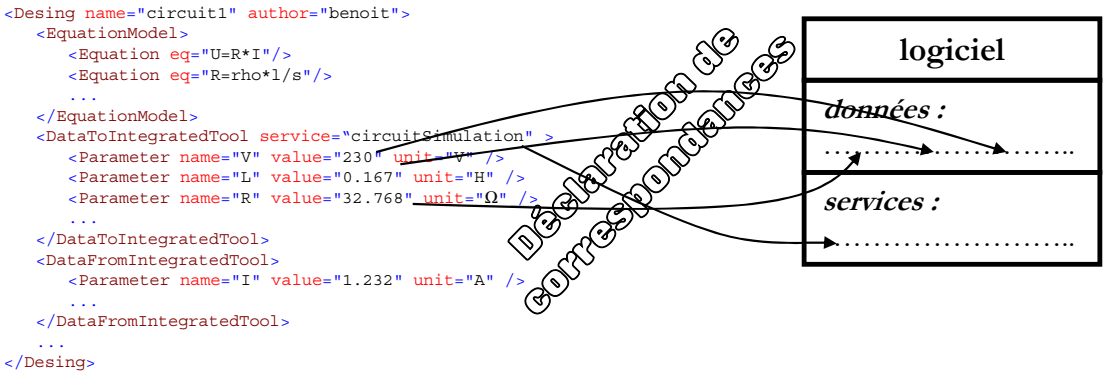


Figure 32 : Exemple d'une déclaration de correspondance entre une description XML de données d'un projet de conception et un logiciel ayant fourni par introspection les informations de données et de services.

Le principe consiste à proposer au concepteur la vision de la description XML et la vision correspondant à ce que propose d'intégrer l'outil grâce aux mécanismes d'introspection. A partir de la connaissance qu'il a des données du logiciel à intégrer et de la structure décrite par son fichier XML, le concepteur va pouvoir déclarer une correspondance.

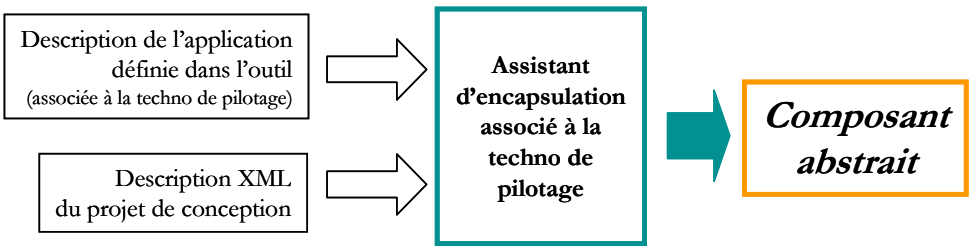


Figure 33 : Principe de l'assistant d'encapsulation, associé à une technologie de pilotage

Cette information de correspondance permettra à l'assistant d'intégration de générer un composant abstrait qui assurera le travail de « glue », transformant les informations du concepteur en informations exécutables par l'outil intégré et réciproquement.

II.C.3. L'éditeur de modèles de fichiers texte : « Template Editor »

Nous avons mis en œuvre cette méthodologie d'encapsulation descriptive dans un outil appelé « TemplateEditor » permettant de générer un composant abstrait à partir d'outils pilotables par fichiers texte. Cet assistant d'intégration fonctionne donc sur le principe décrit, grâce à une déclaration de correspondance intuitive (*glisser-déposer* / *Drag'n Drop*) entre un fichier XML décrit par le concepteur et les fichiers de commande de l'outil à

intégrer. Une fois la correspondance établie, le concepteur décide de créer un composant abstrait « générateur » de fichier de commande (*XML → Fichier texte*) ou un composant abstrait « parser » de fichier de résultat (*Fichier texte → XML*). Cet assistant est détaillé en *Annexe II*.

II.D. Spécification par projection.

II.D.1. Spécifier la sémantique contenue dans le composant abstrait

Cette description XML riche en sémantique pour le concepteur, l'est beaucoup moins pour un logiciel informatique. En effet, la structure des données est ouverte et n'est imposée par aucune spécification. Cela permet d'envisager facilement son enrichissement, mais il est toutefois nécessaire de pouvoir l'utiliser en terme de traitement informatique. De la phase d'encapsulation déclarative résulte un composant abstrait dont les entrées et les sorties, définies par des fichiers, n'ont pas pour vocation à être directement exploitées par les services de l'environnement.

En effet, considérons un fichier XML créé par le concepteur pour décrire son information de conception : aucun service disponible dans l'environnement de conception ne saurait traiter ce fichier sans avoir eu au préalable une description de sa structure. Or, les services disponibles dans l'environnement sont généralement définis pour traiter un type de données bien spécifié. Il peut s'agir de données géométriques pour afficher le dispositif, de courbes de résultats d'analyse, etc.

La phase de spécification consiste donc en une phase de projection durant laquelle le composant abstrait va subir une transformation pour devenir un composant spécifique à un service (*cf. Figure 34*). Les outils de projection sont donc spécifiques aux normes de composant définies par les services.

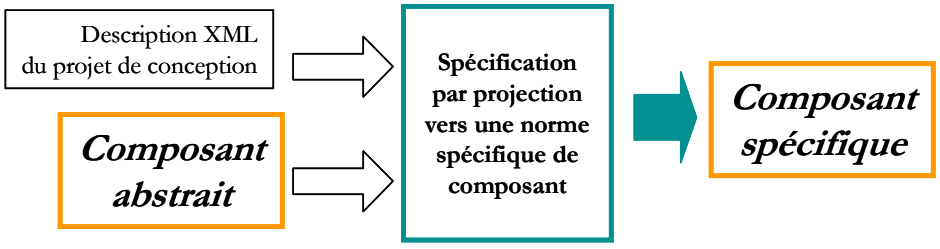


Figure 34 : Principe de la spécification par projection vers une norme de composant

Ainsi, l'outil de projection va permettre au concepteur de définir, grâce à la structure du fichier XML, la sémantique qu'il souhaite affecter à son information de conception. Il va,

par exemple, pouvoir spécifier que telle donnée est une grandeur géométrique dont l'unité est le millimètre et qu'elle peut être modifiée, ou encore que telle autre donnée est une caractéristique physique du système calculée par l'outil encapsulé, etc. Cette sémantique est imposée par les spécifications du composant dans lequel est projeté le composant abstrait.

II.D.2. Projeteur ABC vers COB

Nous avons mis en œuvre cette méthodologie de projection, d'un composant abstrait vers la norme de composant pour le dimensionnement (*COB [ATI 99]*). Ce projeteur s'appuie sur le principe évoqué de spécification par projection. En effet, à partir d'une description XML des informations du composant abstrait, le projeteur définit quels sont les paramètres d'entrée et de sortie du COB grâce à une déclaration de correspondance intuitive (*glisser-déposer / Drag'n Drop*). Il permet également d'ajouter une fonctionnalité de calcul de sensibilité des paramètres de sortie par rapport à ceux d'entrée, fonctionnalité nécessaire pour exploiter le COB dans un service de dimensionnement. Ce projeteur est décrit plus complètement en *Annexe II*.

III. Les principes d'intégration

III.A. Abstraction / Spécification

Le principe utilisé dans notre méthodologie d'intégration s'appuie sur le composant abstrait pour lequel aucune structure de l'information n'est imposée. Les utilisateurs de composants abstraits savent uniquement qu'ils peuvent lui donner un fichier en entrée commandant ainsi la transformation, et récupérer alors un fichier en sortie. Ici, les assistants d'intégration savent également qu'il s'agit d'un fichier XML, donc ils peuvent offrir au concepteur une vision arborescente de son fichier.

Le manque de sémantique manifeste du composant abstrait en fait l'intermédiaire idéal car n'imposant qu'un format XML aux assistants d'encapsulation et aux projeteurs. De ce fait, nous pouvons considérer une ouverture suffisamment large pour que de futures intégrations puissent y faire passer des types de données auxquels nous n'aurions pas pensé s'il avait fallu en faire une liste exhaustive.

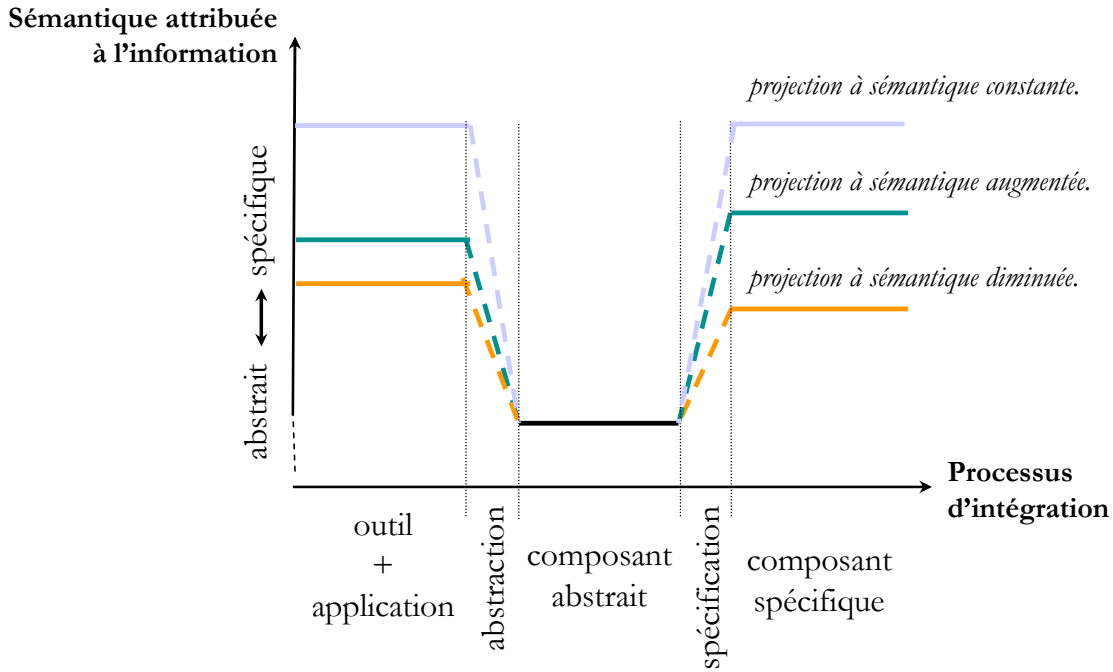


Figure 35 : Principe d'abstraction / spécification associé à la sémantique définie dans les phases d'intégrations

Nous voyons sur le schéma ci-dessus (*cf. Figure 35*) le niveau de sémantique attribué à l'outil dans les différentes phases d'intégration. Nous avons placé dans la dernière phase, celle de projection (*ou de spécification*), trois notions liées au concept de projection que nous pouvons décrire ici à titre d'exemples.

Projection à sémantique constante : le composant résultant de la projection possède les mêmes fonctionnalités que celles de l'outil encapsulé.

Une illustration de projection à sémantique constante d'un composant vers une autre norme de composant, peut correspondre à une projection gérant une hétérogénéité de langages ou de plateforme d'exécution des composants. Considérons une norme de composant de calcul en Java (*les COBs par exemple*), pour laquelle différents services sont disponibles. Considérons maintenant que nous disposons d'un composant répondant aux mêmes spécifications, mais implémenté dans le langage de programmation *C*. Pour avoir accès aux divers services disponibles à l'autre norme de composant, il est possible d'opérer une projection, créant automatiquement la façade du composant Java, et l'interfaçage avec notre composant de calcul en C.

Projection à sémantique dégradée : le composant résultant de la projection n'est pas capable d'exploiter certaines fonctionnalités pourtant présentes dans le composant abstrait,

venant du fait qu'une encapsulation d'outil dans un composant abstrait n'est pas destinée à un composant métier particulier.

Une illustration de projection à sémantique dégradée peut correspondre à la projection d'un composant de dimensionnement dans un composant de calcul. La norme du composant de dimensionnement possède une information supplémentaire (*le calcul de sensibilité*) que l'on ne peut pas projeter dans la norme des composants de calcul. Un tel projeteur n'interfacera alors que les fonctionnalités de calcul.

Projection à sémantique augmentée : le composant résultant de la projection offre des fonctionnalités supplémentaires par rapport aux fonctionnalités encapsulées de l'outil. Dans ce genre de cas, c'est l'outil de projection qui crée les fonctionnalités manquantes pour chacun des composants projetés.

Le projeteur mentionné ci-dessus (*ABC vers COB*) correspond à un tel projeteur à sémantique augmentée, en raison de l'apport de la fonctionnalité du calcul de sensibilité.

III.B. Découplage des outils d'encapsulation et de projection

Le découplage réalisé par le composant abstrait (*cf. Figure 36*) permet notamment de réutiliser plus facilement les différents assistants et projeteurs. En effet, ceux-ci ne dépendent effectivement que de la technologie de pilotage des outils, pour les premiers, et des spécifications des composants métier, pour les seconds. Ainsi, au lieu de réaliser des outils d'intégration spécifiques à chaque logiciel et à chaque service, cette architecture permet de réduire considérablement leur nombre. En effet, elle permet tout d'abord de mutualiser un outil encapsulé dans un composant abstrait pour le projeter ensuite vers différents services selon les besoins. Ainsi l'utilisation de n outils dans k services nécessite $n+k$ outils au lieu de $n \times k$ (*cf. Figure 36*). De plus, les assistants d'encapsulation étant non spécifiques à un outil, mais spécifiques à une technologie de pilotage, ils peuvent être réutilisés par plusieurs outils faisant partis de la même catégorie de pilotage. De même, du coté des services, un certain nombre d'entre eux peuvent utiliser une même norme de composant, permettant ainsi d'exploiter un seul projeteur pour plusieurs services.

IV. Conclusions

Pour résumer la méthodologie d'intégration, rappelons que deux étapes sont nécessaires, articulées autour du composant abstrait :

- Encapsulation (*abstraction*) : utiliser l'assistant d'encapsulation associé aux moyens d'accès définis par l'outil à intégrer, afin de créer un composant abstrait.
- Projection (*spécification*) : ensuite, le composant abstrait doit être spécifié par projection vers la norme du composant associé au service désiré.

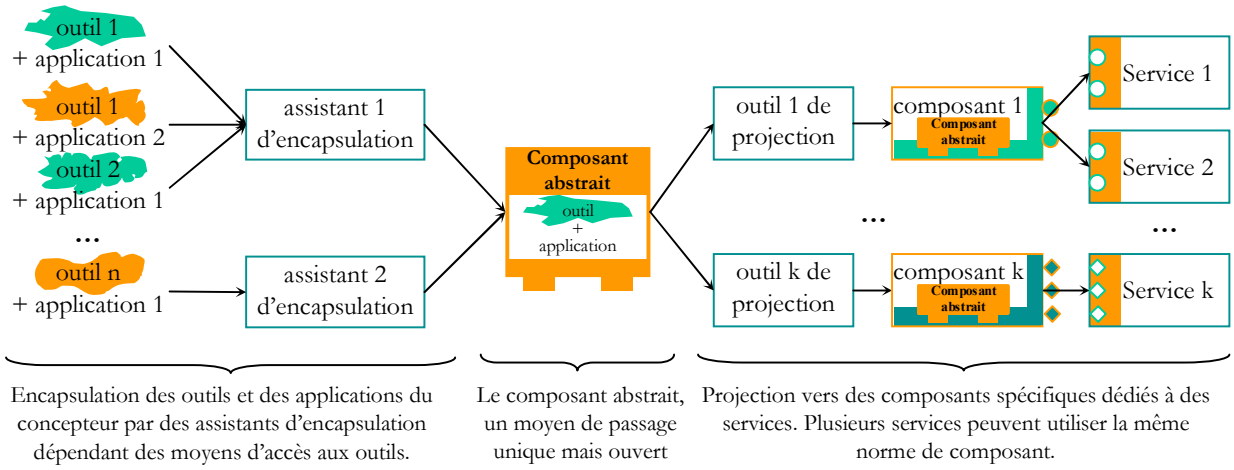


Figure 36 : Détails des étapes du processus d'intégration

Nous avons développé dans cette partie une méthodologie d'intégration, permettant au concepteur d'intégrer ses outils dans l'environnement de conception et ainsi de profiter des services offerts en son sein. Cette méthodologie, s'appuyant sur un composant intermédiaire (*le composant abstrait*), permet d'envisager une mise à disposition rapide d'assistants et de projeteurs qui permettent à un concepteur d'intégrer ses outils sans contraintes de programmation.

Nous détaillerons en conclusion de ce chapitre quels sont les bénéfices de cette méthodologie, mais avant cela, nous allons détailler la méthodologie de composition nécessaire à un environnement à composants.

PARTIE II :

METHODOLOGIE DE COMPOSITION

La méthodologie de composition doit permettre de répondre à la complexité, en créant les relations entre des composants souvent utilisés de manière autonome, et de réutiliser le résultat de cette composition dans les différents services de l'environnement. L'imprévisibilité doit être gérée par le caractère dynamique de la composition, permettant de la reconfigurer rapidement de manière intuitive. Cette méthodologie doit également prendre en compte l'évolution des spécifications des composants au sein de l'environnement de conception.

I. Les besoins d'un outil de composition

I.A. Une composition récursive

La composition, concept capital pour notre environnement, doit être récursive, c'est-à-dire qu'elle doit produire un nouveau composant. Nombre d'environnements mettent en effet en œuvre le principe de composition, mais l'utilisent toutefois de manière non récursive. C'est-à-dire que les informations de composition sont directement exploitées sans leur offrir de possibilités de réutilisation. Notre architecture, résolument orientée composants, doit produire une information de composition qui puisse être re-exploitée par les services existants et futurs de l'environnement.

I.B. Une composition non prescriptive

Comme nous l'avons défini dans le *Chapitre II*, l'architecture composant peut se décliner sur différents patrons de travail selon les besoins spécifiques à chaque norme de composant. Le fait d'autoriser diverses spécifications de composants dans notre environnement impose aux outils, tels qu'un outil de composition, d'être dédié à un type de composant métier, ou alors d'être génériques et de prendre en compte cette diversité. Nous avons choisi de développer ce dernier type d'outil de composition. Pour cela, il devra se baser sur la sémantique minimale de la boite noire n'offrant qu'une liste de ports d'entrée et une liste de ports de sortie (*cf. Figure* 37), à partir desquelles il pourra établir un noyau central générique.

Figure 37 : Sémantique minimale de l'outil de composition : la boite noire se définit par une liste de ports d'entrée et une liste de ports de sortie.

Des compositions peuvent être plus ou moins complexes en fonction du couplage définit. Selon la typologie de connexi on, les composants peuvent échanger des informations de manière unidirectionnelle, bidirectionnelle ou bouclée (*cf. Figure 38*). De plus, la nature des entités échangées peut être plus ou moins complexe, pouvant aller de l'échange de valeurs jusqu'au partage de références d'objets. Enfin, les informations peuvent transiter selon différents modes (*synchrone/asynchrone, continu/discret*). Un outil de composition intégrant les outils du concepteur, sans avoir *a priori* défini leur nature, devra tenir compte de ces différents modes de couplage.

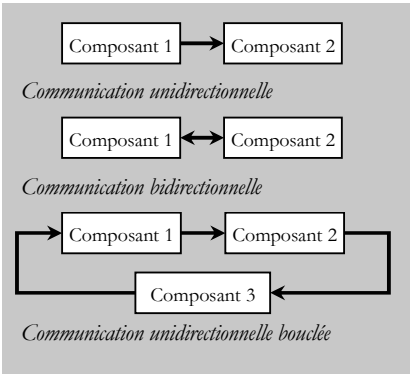


Figure 38 : Les échanges inter composants, typologies de communication

Nous allons donc proposer un tel outil, architecturé autour d'un noyau basé sur une sémantique minimale, celle de la boite noire. Afin de gérer la diversité potentielle de composants et de modes de couplages, des modules spécifiques seront définis et associés à l'outil de composition principal. Ce dernier traitera donc la composition de manière générique et fournira les informations de composition aux modules pour qu'ils gèrent les aspects spécifiques.

I.C. Modifier la composition

L'outil de composition doit être intuitif et donc être basé sur un mécanisme de composition visuelle. Ainsi, son utilisation triviale fera bénéficier le concepteur de la possibilité de gérer rapidement des modifications dans sa composition. Cela lui permettra par exemple de changer un modèle de calcul par un autre pour prendre en compte de nouvelles hypothèses de modélisation, ou d'ajouter des tâches à un processus existant, etc, gérant ainsi l'imprévisibilité liée au processus de conception.

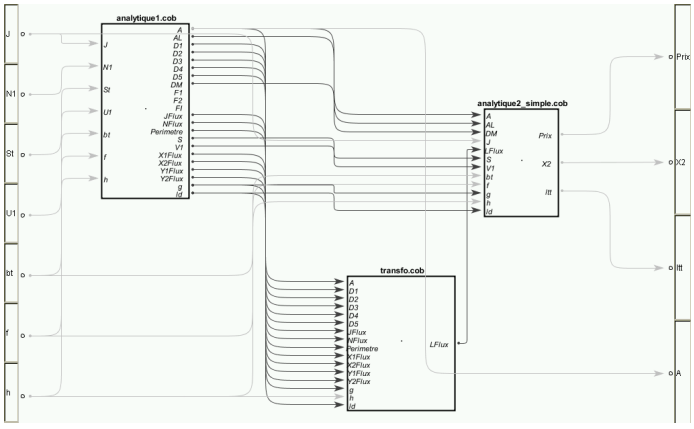


Figure 39 : L'outil de composition permet une composition visuelle

II. L'outil : « Visual Composer »

II.A. Le noyau central

L'architecture que nous avons développée s'articule autour d'un noyau construit sur le concept de boite noire. Autour de ce noyau sont disponibles des bibliothèques de composants à connecter et des modules de composition spécifiques (*cf. Figure 40*).

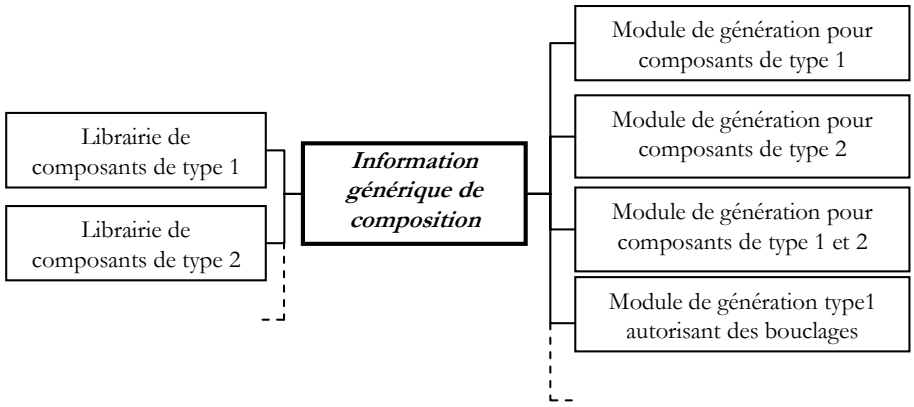


Figure 40 : Architecture modulaire de l'outil de composition

II.A.1. Fonctionnalités graphiques

Le noyau possède un panneau graphique, dans lequel il est possible de positionner différentes boites noires (*cf. Figure 39*). Ces boites noires peuvent être agencées, puis leurs ports connectés automatiquement (*pour des mêmes identifiants, cf* Figure 41) ou manuellement (*pour des identifiants différents*). Ce type de connexion définit une correspondance entre deux ports mais aussi un séquencement entre les composants en raison de l'orientation des connexions (*source, destination*).

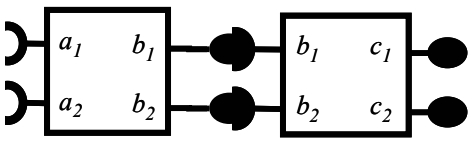
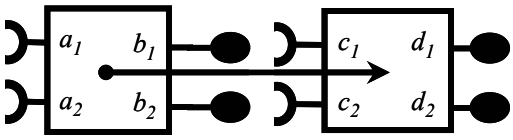


Figure 41 : Les ports nommés permettent de définir des mécanismes de connexion automatique pour des noms identiques.



Un autre type de connexion est donc disponible, cette fois entre les composants eux mêmes, définissant simplement un séquencement, sans connexion de ports (*cf. Figure 42*).

Figure 42 : Les boites sont connectées définissant le séquencement.

Enfin, pour définir la récursivité, il est nécessaire de définir les ports du composant résultant de la composition. Pour cela, différents mécanismes automatiques proposent de

faire ressortir les ports non connectés. L'utilisateur peut également définir manuellement des ports d'entrée ou de sortie globaux puis les connecter à des ports des boites noires de la composition. L'identifiant d'un port global est indépendant de celui des ports auxquels il fait référence dans la composition, ce qui permet de redéfinir complètement les identifiants d'entrée/sortie du composant généré.

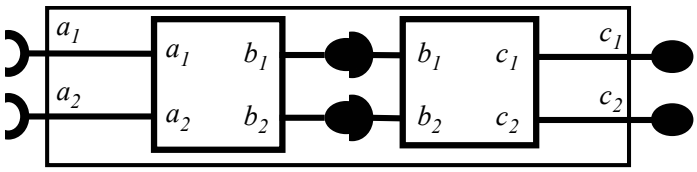
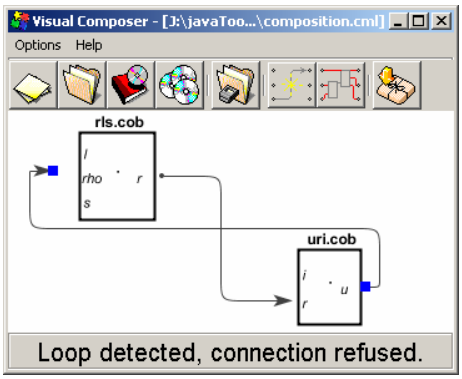


Figure 43 : Les ports résultants de la composition, permettent de définir un nouveau composant (récursivité)

II.A.2. Gestion de la cohérence de composition

Des fonctionnalités génériques de vérification de la composition peuvent utilisées, permettant par exemple d'autoriser ou non des bouclages entre composants (*cf. Figure 44*), ou encore de placer plusieurs connecteurs sur un même port d'entrée.



Un autre mécanisme est disponible, déléguant l'autorisation de connexion aux modules afin qu'ils gèrent eux mêmes les cohérences de composition spécifiques.

Figure 44 : Mécanismes de gestion de cohérence, ici les boucles sont interdites

II.A.3. L'information de composition

Une fois la composition réalisée, c'est le module de composition qui doit être capable de générer le composant composé. Pour cela, l'information de composition est traitée, informant le module :

- des composants présents dans la composition,
- de la liste des connexions entre ports (*source:port, destination:port*),
- de la liste des connexions entre composants (*source, destination*),
- de la liste des ports d'entrée et de sortie du composant résultant,
- de la liste des connexions entre ces ports et les ports des composants internes (*port, destination:port*) et (*source:port, port*),

Des traitements génériques peuvent être réalisés à partir de ces informations, comme la création d'une matrice d'occurrence, permettant ainsi d'alléger les modules afin qu'ils ne traitent que leurs spécificités.

II.B. Les modules de génération

Les modules permettant de générer le composant composé sont dissociés du noyau de l'outil, ils peuvent être ajoutés ou retirés selon les besoins. Chaque module correspond à un fichier placé dans un répertoire « plug-ins » de l'application de composition. Une composition sera nécessairement associée à un module de génération et ne pourra fonctionner si le module n'est plus disponible.

II.B.1. Création d'une boite noire à partir d'un composant

Lorsque l'outil de composition souhaite afficher une boite noire dans l'espace de composition, il demande au module de génération de lui transmettre l'information nécessaire (*listes d'entrée/sortie*). Pour cela, le module introspecte le composant souhaité et définit les entrées et sorties de la nouvelle boite noire qui sera associée à ce composant.

De cette manière, une grande souplesse est laissée aux modules, permettant, par exemple, à certains d'entre eux d'exploiter les capacités de composition pour des composants « boite blanche ». En effet, si le module est à même de définir les entrées/sorties appropriées, le composant boite blanche sera perçu comme un composant boite noire quelconque dans le noyau de l'outil de composition.

II.B.2. Création d'un moteur de résolution

A partir de l'information de composition, les modules doivent produire le code informatique nécessaire à la mise en œuvre de cette composition. Ce code, le moteur de résolution, peut varier selon la nature des composants et de la composition. Il peut par exemple intégrer des capacités de bouclage sur les composants, de gestion de calcul parallèle, etc. Nous pourrons voir, par la suite, deux modules particuliers traitant de manières différentes la composition, selon qu'il s'agit de modèles de calcul ou de tâches d'un processus.

II.B.3. Emballage du composant

***Emballage** : procédure permettant de définir le composant en tant qu'entité autonome (ex : fichier).*

Un dernier point doit encore être traité par les modules, celui de l'**emballage** (*packaging*), nécessaire à tout générateur de composant. L'emballage est la procédure qui permet de définir le composant en tant qu'entité autonome. C'est notamment durant cette étape que

les composants, définissants la composition, sont placés à l'intérieur (*encapsulés*) du composant global. Les modules de génération peuvent alors choisir deux options selon la nature des composants considérés. Soit ils placent les composants tels quels au sein du composant global, soit ils « ouvre l'emballage » des composants pour ne placer que leur contenu. La première solution correspond à une vision boite noires ; la deuxième doit, quant à elle, connaître la manière dont l'emballage est réalisée.

Ces deux méthodes peuvent notamment être utilisées pour de la composition récursive à plusieurs niveaux (*cf. Figure 45*). La deuxième méthode considère les composants comme des « **boites grises** ». En effet, si l'information de composition est accessible au sein des composants déjà composés, il est possible de les ouvrir et de recomposer le tout (*cf. Figure 45 droite*).

***Boite grise** : Composant boite noire dont on connaît une partie de l'implémentation.*

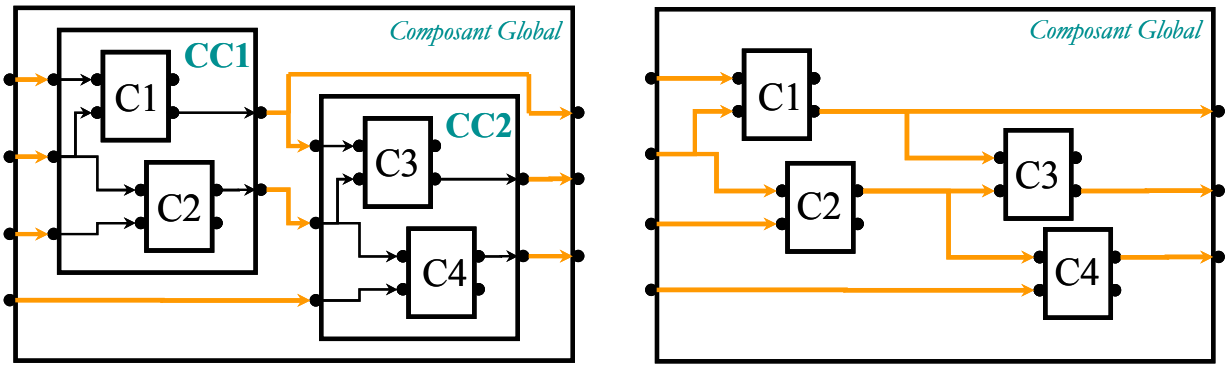


Figure 45 : Deux visions du packaging : boite noire, et boite grise dans laquelle l'information de composition est disponible.

III. Les modules développés

Nous avons développé deux modules de génération particuliers, répondant aux spécifications de l'outil de composition que nous venons de décrire.

- Le premier de ces modules permet la composition simple (*sans boucles*) de composants COB, permettant, par exemple, de dimensionner une structure de dispositif définit par plusieurs modèles distincts (*multi-physiques, multi-disciplinaires*).
- Le deuxième module permet la composition des composants définis pour l'intégration, les composants abstraits (*les ABCs*). Ce genre de composition permet de créer une description de séquencement de tâches (*workflow*) supportant, par exemple, un processus de conception.

III.A. Composition de modèles de calcul pour le dimensionnement

III.A.1. Les composants de calcul pour le dimensionnement

Les COBs, que nous avons décrits dans le *chapitre 2*, correspondent aux composants actuellement les plus disponibles dans notre contexte de travail, car disposant de plusieurs services dont celui du dimensionnement par technique d'optimisation. La particularité de ce composant réside dans la nature des paramètres définis en entrée et en sortie. Ils contiennent en effet deux grandeurs, une valeur et une valeur de différentielle (*ou dérivée directionnelle [GIL 91]*). Ils possèdent également deux méthodes d'appel qui permettent de se calculer ou de se différentier (*cf. Figure 46*).

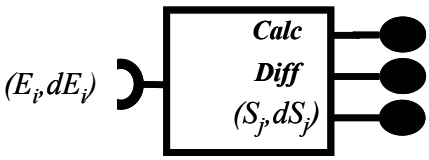


Figure 46 : COB : composant de calcul pour le dimensionnement

III.A.2. Moteur de résolution : séquencement

Les paramètres connectés entre les différents composants définissent l'information de composition, à partir de laquelle le générateur doit définir le séquencement des composants. Pour produire un séquencement qui puisse s'exécuter, chaque composant est encapsulé par une entité plus complète (*Runner*), laquelle lui déléguera les fonctionnalités de calcul.

Ce *Runner* possède notamment des informations qui permettent de savoir s'il est prêt à s'exécuter. Il est capable d'attendre que d'autres *Runners* aient terminé de s'exécuter, ainsi que de signaler aux *Runners* qui se sont enregistrés auprès de lui, qu'il a lui même fini son exécution. Pour définir le séquencement, il est donc nécessaire de :

- définir tous les composants qui doivent être exécutés en premier (*ceux qui ne nécessitent aucune information d'autres composants*).
- enregistrer les autres composants comme attendant la fin de l'exécution des composants le précédant.

Chacun des *Runner* est définis dans un processus indépendant qui permet, par exemple, de distribuer les composants COB, trop lourds, sur différentes machines. Ces COBs distants peuvent être obtenus par deux projections successives à « sémantique constante » (*COB→CORBA*[16]*→COB*) [DEL 02a] (*cf. Figure 47*)

[16] CORBA : norme d'interopérabilité entre langages de programmation et entre systèmes d'exploitation différents. Pour plus d'informations, consulter l'*Annexe IV* décrivant les problématiques de pilotage de logiciels.

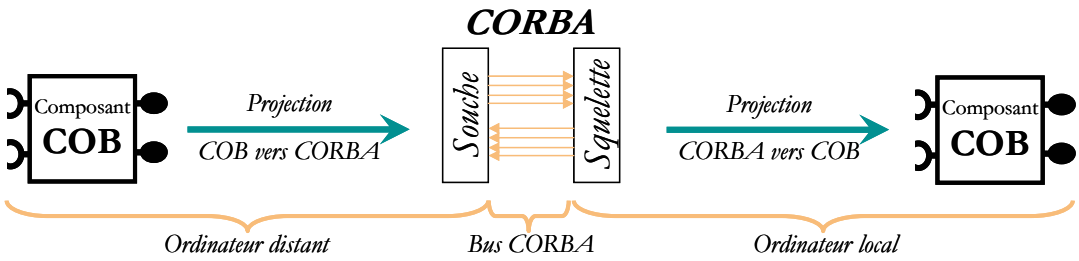


Figure 47 : Mécanisme de double projection permettant de distribuer un composant de calcul sur différentes machines

III.A.3. Moteur de résolution : propagation de valeur

Concernant la propagation des valeurs entre les composants, elle devrait être définie à chaque fin d'exécution d'un *Runner*. Or les spécifications des composants COB, définies dans une optique de composition [ATI 99], permettent une gestion simplifiée de propagation des valeurs. En effet, les COB permettent de gérer cette propagation par un mécanisme de partage d'instance. Les valeurs des paramètres n'existent en fait que dans les interfaces de sortie des composants. Les interfaces d'entrée doivent alors « pointer » vers la valeur définie dans les interfaces de sortie auxquelles elles sont connectées.

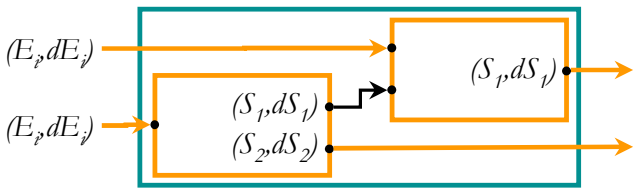
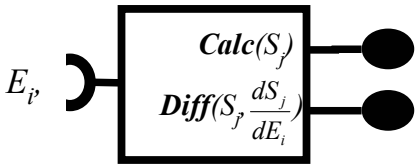


Figure 48 : Mécanisme de partage d'instance permettant la propagation de valeur au sein d'une composition

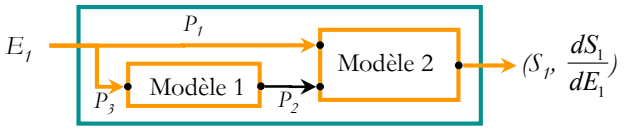
Une autre particularité des COBs peut être soulevée ici, il s'agit de l'utilisation de différentielle. Grâce à ces spécifications, la composition du calcul global de sensibilité est automatique par simple propagation des différentielles.



D'autres normes de composants de dimensionnement ont été introduites dans l'équipe CDI, notamment celle mettant en œuvre des dérivées partielles au lieu des différentielles.

Figure 49 : Norme de composant de dimensionnement utilisant les dérivées partielles au lieu des différentielles

Ce modèle de composant demande un moteur de résolution plus complexe que celui mis en œuvre dans le cas de la composition des COBs. En effet, il nécessite un superviseur remontant les différents composants, à partir de la dérivée partielle de sortie, afin de composer toutes les dérivées partielles intermédiaires. Dans l'exemple de la *Figure 50*, le calcul nécessaire est celui décrit par l'équation *(1)* :



$$\frac{dS_1}{dE_1} = \frac{\partial S_1}{\partial P_1} + \frac{\partial S_1}{\partial P_2} \cdot \frac{dP_2}{dP_3} \qquad (1)$$

Figure 50 : Une composition par dérivées partielles nécessite un superviseur pour propager le calcul de sensibilité

Bien qu'imposant un moteur de résolution plus compliqué à mettre en oeuvre, cette norme de composant offre la possibilité d'alléger les calculs de sensibilité lorsque toutes les dérivées partielles ne sont pas nécessaires. En effet, réalisé dans un contexte de modèles analytiques (*très rapides*), le COB réalisant le calcul des différentielles par propagation automatique, ne permet pas de sélectionner les sorties sur lesquelles on souhaite avoir la sensibilité. L'utilisation du COB pour des modèles de calcul lourds, nécessite alors de gérer de manière très précise le calcul de sensibilité (*cf. Annexe III*) [DEL 03].

Cet exemple nous montre combien le contexte d'utilisation peut imposer des différences fondamentales entre des composants réalisant des fonctionnalités similaires, et *defacto*, la nécessité d'une gestion de l'évolution du besoin en s'adaptant. C'est dans ce cadre d'application que des projeteurs à sémantique constante permettent de réutiliser des composants existants mais différents.

III.A.4. Conclusion

Ce module de génération, composant des modèles de dimensionnement, nous permet d'envisager un grand nombre d'applications afin de venir en aide au concepteur durant son processus de conception, mais aussi de lui offrir de nouvelles méthodologies de travail. Nous détaillerons quelques applications dans le prochain chapitre, dont la mise en œuvre d'un modèle mixte (*analytique* + *numérique*) pour le dimensionnement d'un transformateur de tensions triphasées.

III.B. Description d'un workflow

III.B.1. Une architecture adaptée au support des processus de conception

La difficulté de supporter les processus de conception dans un environnement informatique vient souvent d'outils qui n'offrent qu'une définition statique de ce processus. Le support de processus de conception dynamiques est fortement dépendant de l'architecture informatique choisie.

III.B.1.i Exemple d'une architecture centralisée

Considérons l'environnement intégré EDIP (*Electrical Design Integrated Platform*) qui a été développé dans l'équipe CDI durant la thèse de Basma Bel Habib [BAS 00]. Cet environnement permet, grâce à une architecture centralisée dont la base de données s'appuie sur un modèle paramétrique (*entité→valeur : chaine de caractères→réel*), d'exécuter les outils déjà intégrés et connectés à cette base. La *Figure 51* décrit cette architecture dans un exemple d'utilisation connectant les paramètres de dimensionnement à la géométrie paramétrée du dispositif étudié.

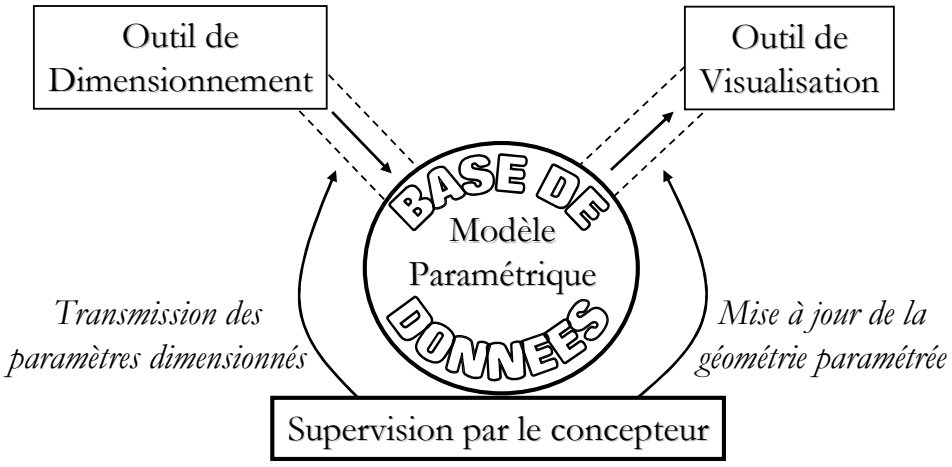
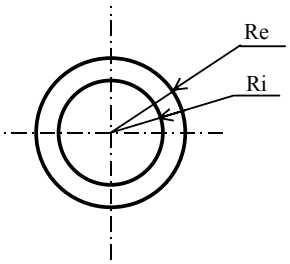


Figure 51 : Architecture d'un environnement de conception centralisée.

Cette solution est très intéressante car le processus est défini dynamiquement par le choix du concepteur (*en temps réel*) des outils à exécuter. Malheureusement, cette architecture possède quelques uns des écueils évoqués dans les outils de CAO classiques, tels que le modèle commun d'échange.

III.B.1.ii Cohérence de la correspondance des données

Dans une telle architecture, les outils n'ont aucune information concernant la provenance des données qu'ils réutilisent, et la destination des données qu'ils modifient. Imposant une transformation « une fois pour toute » de leurs données sur le modèle central, ils ne peuvent pas créer d'échanges spécifiques et contextuels.



Considérons un exemple simple dans lequel deux outils souhaitent échanger le rayon extérieur d'un cylindre. Dans un premier outil, la grandeur « *Re* » est bien définie, dans le deuxième, seuls sont définis les grandeurs « *Ri* » (*rayon intérieur*) et « *ep* » (*l'épaisseur du cylindre*). Ces grandeurs sont reliées par la relation *Re*=*Ri*+*ep*.

Figure 52 : Illustration d'un échange de données entre outils devant faire correspondre des informations différentes

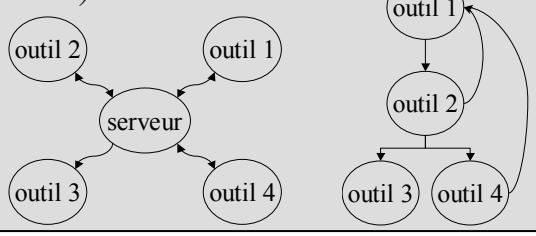
Dans le cas d'une architecture centralisée, se pose la question de ce qui est placé dans la base de données : « *Re* », ou « *Ri* » et « *ep* », ou les 3 ? Dans les deux premiers cas, au moins une des deux connexions doit faire la correspondance : *Re*=*Ri*+*ep*, dans la troisième solution, c'est le serveur de données qui doit gérer la cohérence des données redondantes en propageant les modifications. Une autre architecture permet de résoudre cette difficulté.

III.B.1.iii L'architecture point-à-point

A l'architecture centralisée (*Client/Serveur*) est souvent opposée l'architecture décentralisée « point à point » (peer-to-peer : P2P), où chaque outil communique directement avec les outils auxquels il est connecté. Le principe de composition de composants que nous avons choisi correspond à ce dernier modèle d'architecture.

Un même processus peut être exécuté de deux façons :

- *par une architecture, définissant l'accès à une base commune, pour laquelle il faut définir le séquencement des outils (figure de gauche).*
- *par une architecture définissant les échanges entre outils, et implicitement le séquencement (figure de droite).*



Si l'on considère l'exemple évoqué ci-dessus (*partage de Re*) dans le cas d'une architecture P2P, une unique connexion est définie entre les deux outils (*ou 2 pour des besoins de communication bi-directionnels*). Ainsi, c'est à elle(s) que sera confiée la gestion de la correspondance des données *Re*=*Ri*+*ep* ainsi que la gestion des droits en lecture/écriture (*sens du flux d'information*). Afin de gérer la cohérence des données, elle ne permettra pas la modification dans le sens Ri+ep←Re, ou

bien définira une règle qui spécifiera par exemple que « *Re* » modifie « *Ri* » et garde constant « *ep* ».

La solution point-à-point permet donc, grâce à la spécification de connexions contextuelles, de gérer la cohérence des correspondances de données. Mais la définition des échanges de données entre les différents composants impose implicitement la définition du processus (*cf. encart*). La solution que nous devons mettre en place pour obtenir une définition dynamique des processus de conception, consiste alors à créer dynamiquement les liens entre composants.

III.B.2. Le workflow adaptatif

Le workflow, souvent mis en œuvre pour supporter des procédures administratives, est généralement considéré comme statique. Notre objectif est de mettre en œuvre une solution dynamique d'automatisation de processus (*workflow automation [STO 01]*), grâce à l'outil de composition *VisualComposer*. Pour cela, la procédure est la suivante :

- Modéliser le processus de conception, en définissant les tâches et les séquencements.
- Créer les composants associés aux tâches (*méthodologie d'intégration*),
- Créer les séquencements entre les composants (*méthodologie de composition*).

Nous disposons déjà de la méthodologie d'intégration qui permet de définir des composants abstraits à partir des outils du concepteur. Il reste donc à définir le module de génération qui, à partir de composition de composants abstraits, crée le composant décrivant le processus de conception.

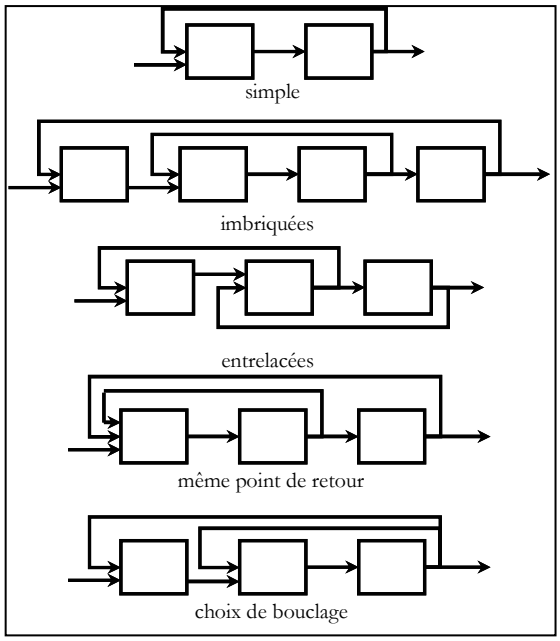
III.B.3. Module de génération de workflow

Le module de génération doit tout d'abord définir la boite noire à partir d'un composant abstrait. Ce dernier ne possède qu'une entrée (*fichier d'entrée*) et qu'une sortie (*fichier de sortie*), la boite noire sera toujours la même :

Figure 53 : Définition de la boite noire associée à un composant abstrait

Ensuite, nous devons considérer le moteur de résolution. Ce moteur pourra fonctionner sur le même principe que le module de génération de composants de dimensionnements. Mais, il devra en plus prendre en compte une caractéristique nécessaire à la définition d'un processus de conception, celle lui permettant d'itérer sur les étapes du processus. Il s'agit

donc de prendre en compte les bouclages entre composants, c'est-à-dire repérer les boucles, les catégoriser, puis définir pour chaque type de boucle la gestion appropriée.



Parmi les différentes boucles présentées dans la *Figure 54*, les trois premières catégories (*simple, imbriquées et entrelacées*) relèvent de la même gestion. En effet, le critère d'arrêt de ces boucles n'est autre que le choix du concepteur à chaque bouclage, fait à partir d'un message s'affichant à l'écran.

Afin de gérer la quatrième catégorie (*boucle avec même point de retour*), il suffit que le mécanisme définissant le flux à donner à la tâche sur laquelle on reboucle (*flux standard, ou flux de bouclage*) puisse gérer plusieurs flux de bouclage.

Figure 54 : Différents bouclages possibles entre les composants

Ces différents bouclages sont pris en compte dans la version actuelle du module de description de processus. Le dernier type n'est, quant à lui, pas géré par cette version car le seul fait de savoir si le concepteur souhaite boucler ou non n'est plus suffisant, celui-ci doit en effet être en mesure de choisir un bouclage parmi plusieurs.

IV. Conclusion

Nous avons développé dans ce chapitre deux méthodologies nécessaires à la mise en œuvre d'un environnement de conception intégrateur des outils du concepteur et gérant la complexité, l'imprévisibilité et la dynamique en conception. C'est notamment grâce à la contextualisation par le concepteur (*intégration de l'application, ou configuration des connections de composants*) que ces méthodologies parviennent à répondre aux critères que nous nous étions imposés.

IV.A.1. Prise en compte de la complexité

La méthodologie d'intégration permet d'**utiliser un outil dans les services existants de l'environnement :** une fois plongé dans l'environnement, l'outil dispose de tous les services du patron de travail. Il dispose également des projeteurs pour changer de patron et utiliser ainsi d'autres services.

La méthodologie de composition permet d'**utiliser un outil, habituellement isolé, dans un contexte synergique :** en assurant la mise en commun des capacités des différents outils du concepteur, il offre de nouvelles possibilités de travail, ou tout au moins, un gain de temps et de robustesse dans les échanges entre outils.

IV.A.2. Prise en compte de l'imprévisibilité et de la dynamique

La méthodologie d'intégration permet d'**intégrer de nouveaux outils :** les assistants d'encapsulation sont réutilisables sans programmation, pour différents outils ayant le même type de pilotage (*accès au logiciel*).

Les méthodologies d'intégration et de composition permettent de **réagir à la dynamique de conception :** en effet, le cycle d'intégration/composition peut être rejoué sans aucune programmation, le concepteur n'ayant qu'à apporter les modifications souhaitées. C'est ainsi qu'il est possible de modifier facilement et rapidement le nombre, la nature et la configuration des informations intégrées et échangées.

IV.A.3. Applications de ces méthodologies

Ces méthodologies nous permettent d'envisager un grand nombre d'applications afin de venir en aide au concepteur durant son processus de conception. Nous allons détailler dans le chapitre qui suit quelques applications qui mettent en évidence les enjeux méthodologiques de notre environnement.

Chapitre 4

APPLICATIONS

C h a p i t r e 4

APPLICATIONS

Après avoir détaillé les méthodologies d'intégration et de composition de notre environnement de conception, nous allons maintenant exprimer de manière applicative leurs potentiels pour parvenir à supporter le travail du concepteur.

Nous allons tout d'abord détailler, dans le cadre du dimensionnement d'un transformateur de tension triphasées, comment le concepteur peut être capable de mettre en œuvre des modèles multidisciplinaires (*électromagnétisme, économique,* …), des modèles multi-outils (*analytique et numérique*), ainsi qu'un processus de dimensionnement multi-niveaux (*modélisation grossière et modélisation fine*).

Nous détaillerons enfin deux applications des méthodologies d'intégration et de composition, dans le cadre de collaborations avec des concepteurs de micro-systèmes magnétiques, et dans le cadre d'un travail de recherche en conception collaborative avec des chercheurs en conception mécanique.

PARTIE I : APPLICATION A L'OPTIMISATION MULTI MODELES, MULTI NIVEAUX

I. Méthodologie de dimensionnement multi modèles

Une méthodologie multi modèles de dimensionnement consiste à dimensionner un dispositif avec un modèle rapide même peu précis, puis vérifier avec une simulation, et enfin affiner si besoin [WUR 97]. Cette méthodologie met donc en œuvre consécutivement deux modélisations des mêmes phénomènes, mais avec des niveaux de finesse différents. Un support informatique à ce processus pourrait être de définir le workflow entre ces deux tâches, en fournissant par exemple à l'outil de simulation les dimensions optimales trouvées par l'optimisation du modèle analytique.

Pourtant un tel processus n'est pas toujours suffisant, nous pouvons en effet imaginer que les hypothèses formulées au niveau du modèle analytique ne sont pas suffisantes pour approcher une solution réellement optimale. L'idée consiste donc à supporter un dimensionnement automatique au niveau d'un modèle de simulation (*éléments finis en électromagnétisme par exemple*) [VAS 94a][VAS 94b][SAR 99][CAL 01].

Il s'avère qu'une telle solution ne permettra pas de prendre en compte la diversité des contraintes de dimensionnement que l'on peut exprimer dans un modèle analytique. Les spécifications de dimensionnement font, en effet, apparaître des contraintes sur des grandeurs telles que celle du coût, que la simulation « monodisciplinaire » ne traite pas.

La solution consiste en un modèle global mixte (*numérique* + *analytique*) qui utilise la souplesse de modélisations analytiques, mais qui prend aussi en compte certaines difficultés de la modélisation analytique, par des méthodes numériques.

Nous allons donc montrer dans cette partie comment, grâce aux méthodologies d'intégration et de composition, nous pouvons parvenir à nos fins de conception sans besoin de programmation.

I.A. Description de l'application à dimensionner

I.A.1. Approximation du modèle analytique

Nous souhaitons dimensionner un transformateur triphasé, dont un modèle analytique est décrit dans [POL 86] et [FAN 99] ainsi qu'en *Annexe I*. L'objectif de conception est de dimensionner la culasse magnétique ainsi que les enroulements par rapport à des contraintes et objectifs du cahier des charges. Pour cela, le modèle analytique calcule les paramètres magnétiques (*induction, densité de courant, pertes Joules et fer* …), les dimensions, le poids et le prix.

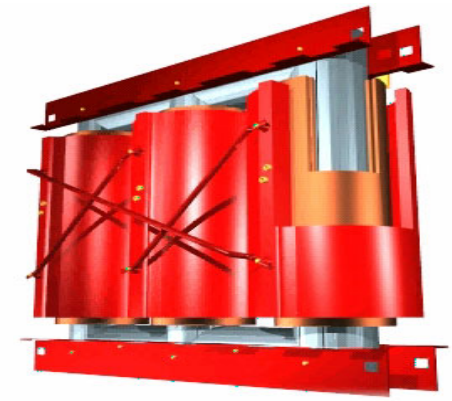


Figure 55 : Image tirée du CD-ROM « Le Transformateur de distribution », www.sfrs.fr, M. Oddon, D. Hilaire, J.-C. Sabonnadière

Le modèle peut être décrit en plusieurs parties plus ou moins indépendantes selon les domaines concernés. Ainsi, 4 modèles ont été définis (*cf. Annexe I*), celui des calculs des paramètres électromagnétiques (*réactance de fuite,* …), celui des calculs géométriques (*coefficient de remplissage des bobinages, volume total de cuivre, de fer*…), celui du calcul des pertes (*Joules et fer*), et enfin un modèle économique permettant de calculer le coût du transformateur (*coût des matières premières, et coût des pertes capitalisées sur 30 années d'utilisation*).

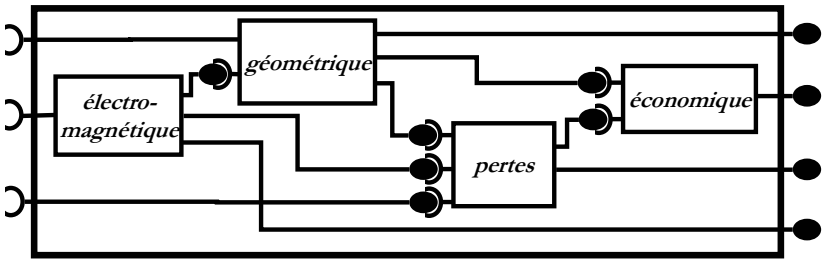


Figure 56 : Composition de 4 modèles pour définir le modèle global de calcul pour le dimensionnement du transformateur

Les 4 composants nécessaires à ce modèle global peuvent être créés par le générateur de composants du logiciel Pro@Design® qui a été présenté dans le *Chapitre II*. Ce générateur, s'appuyant sur la norme de composant de calcul pour le dimensionnement COB, permet par l'analyse des équations du modèle, de créer le code informatique assurant le calcul des équations ainsi que le calcul de sensibilité des différents paramètres du modèle grâce à une dérivation symbolique des équations.

Les composants ainsi créés peuvent alors être composés dans l'outil de composition que nous avons réalisé (*Visual Composer*) afin de créer le modèle global (*cf. Figure 57*)

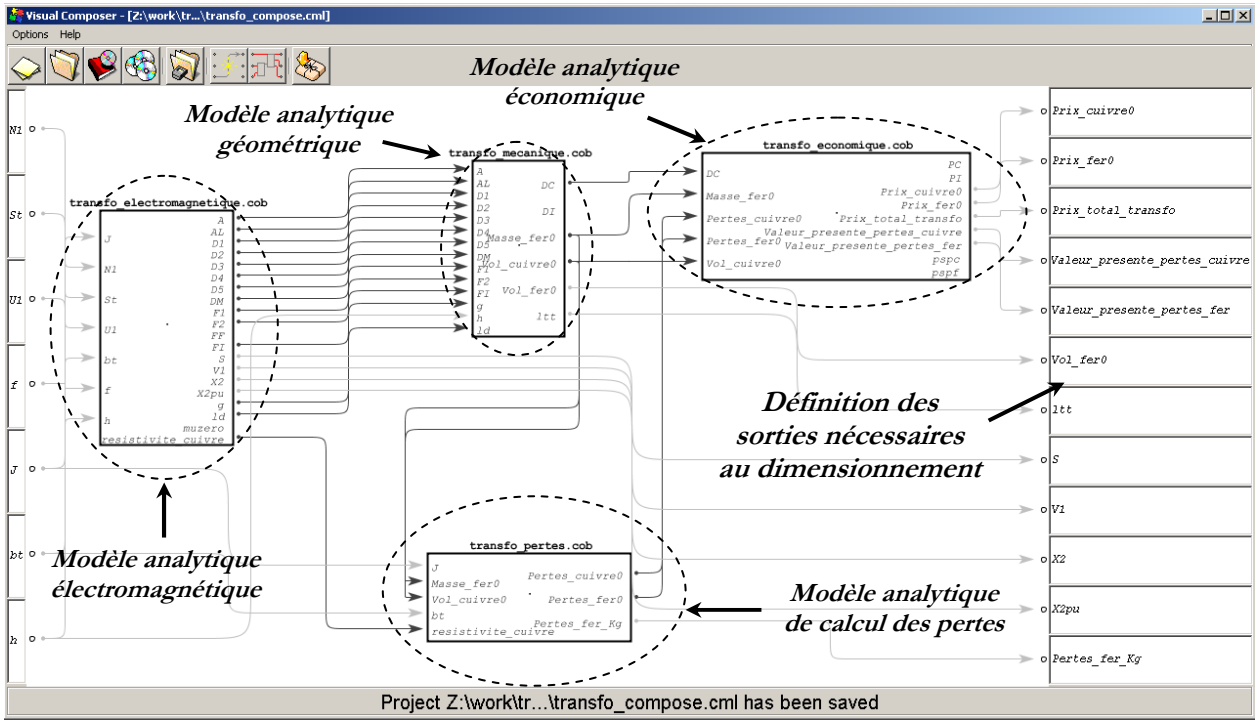


Figure 57 : Composition de 4 modèles analytiques pour définir le modèle de calcul pour le dimensionnement du transformateur

I.A.2. Approximation du modèle analytique

Parmi les paramètres du modèle de calcul électromagnétique est calculée la réactance de fuite vue du primaire. La valeur de cette réactance peut s'exprimer, par l'intermédiaire de l'inductance de fuite, par le calcul de l'énergie dans les enroulements et entre ceux ci durant un court circuit.

$$\frac{1}{2}L \cdot i^2 = \frac{1}{2}\mu_0 \iiint H^2 \cdot dv \qquad (2)$$

Or le flux de fuite augmente de façon linéaire dans les enroulements pour atteindre une valeur maximum et constante entre ceux-ci (*cf. Figure 58*). A partir de cette distribution du flux, la réactance peut dont être exprimée facilement par l'expression suivante (*voir le Tableau 1 pour la description des variables*) :

$$X_1 = \mu_0 . \pi . Dm . N_1^2 . \varpi . \left(D_2 + \frac{A+G}{3} \right) . \frac{1}{h} \qquad (3)$$

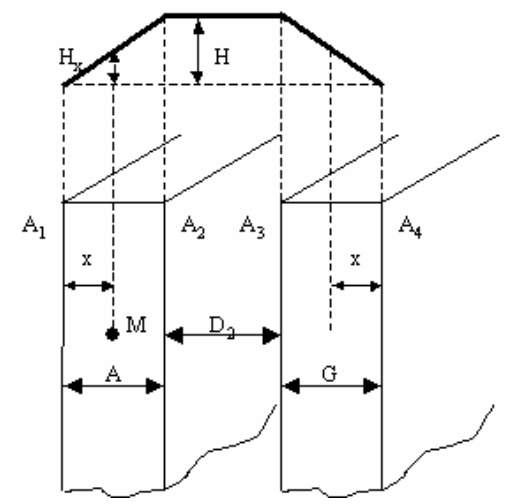


Figure 58 : Distribution du flux magnétique entre et à l'intérieur des enroulements

Tableau 1 : Paramètres du modèle analytique du transformateur

L :	Inductance de fuite	X_1 :	Réactance de fuite vue du primaire
i :	Courant dans les enroulements	N_1 :	Nombre de tours du primaire
μ_0 :	Perméabilité du vide	Dm :	Diamètre moyen de la colonne centrale
dv :	Espace annulaire des enroulements et de l'espace inter enroulements.	D2 :	Distance entre les enroulements du primaire et du secondaire
A :	Epaisseur de l'enroulement primaire	h :	Hauteur des enroulements
G :	Epaisseur de l'enroulement secondaire	H :	Champ magnétique

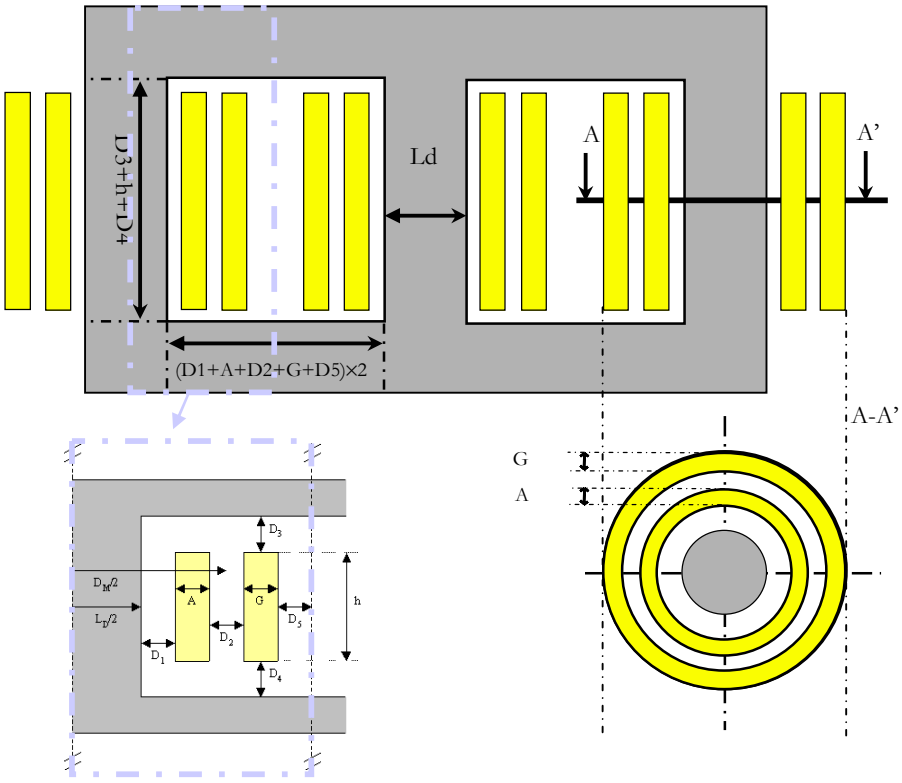
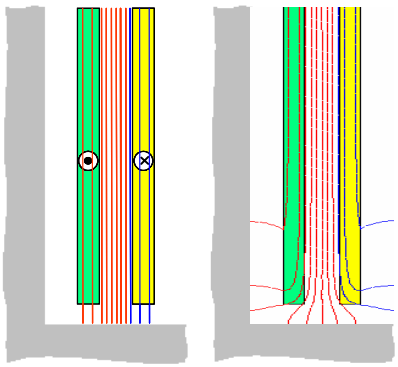


Figure 59 : Géométrie paramétrée du transformateur de tensions triphasées à concevoir

I.A.3. Un modèle semi-numérique



Cette modélisation fait une approximation car elle ne prend pas en compte les effets aux extrémités des enroulements (*cf. Figure 60*). Ainsi, dans ce modèle, le flux est supposé rectiligne sur toute la hauteur des enroulements. En considérant cette approximation comme trop forte pour certains cahiers des charges, nous choisissons de calculer l'inductance de fuite par une méthode numérique à éléments finis.

Figure 60 : Lignes de flux dans les enroulements, sans et avec la prise en compte des effets de bords

Nous allons donc constituer un modèle mixte, utilisant les modèles analytiques précédents mais exploitant les capacités de calcul numérique pour calculer une inductance de fuite moins approximée que dans le modèle analytique.

I.B. Construction du modèle électromagnétique mixte

I.B.1. Composition de trois modèles

Le modèle électromagnétique mixte que nous allons développer afin d'être substitué au modèle électromagnétique analytique utilisé précédemment, est composé de trois modèles (*cf. Figure 61*). Parmi ceux-ci, deux sont définis par un système d'équations et le troisième par un calcul éléments finis.

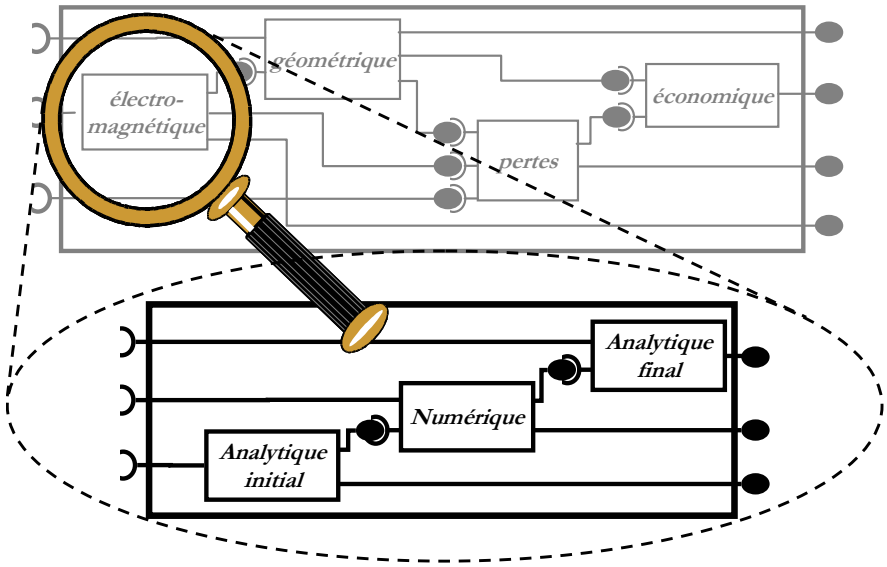


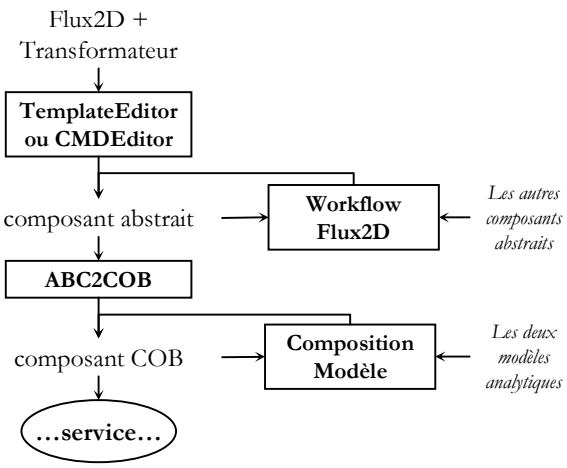
Figure 61 : Le modèle électromagnétique mixte est composé de trois modèles, 2 analytiques et 1 numérique

Le modèle analytique initial calcule les principaux paramètres du modèle électromagnétique, dont ceux nécessaires au calcul de l'inductance de fuite par le logiciel éléments finis. Le modèle numérique permet ensuite de calculer l'inductance de fuite. Enfin, le modèle analytique final assure le calcul des paramètres restants du modèle électromagnétique nécessitant le calcul de l'inductance de fuite, comme le calcul de la réactance de fuite.

Les deux composants associés aux modèles analytiques peuvent être générés comme précédemment grâce au générateur de Pro@Design. Le composant qui effectue le calcul de l'inductance de fuite est généré grâce à un processus d'intégration que nous allons décrire.

I.B.2. Génération du composant de calcul de l'inductance de fuite

Le calcul de l'inductance de fuite est réalisé par le logiciel éléments finis Flux2D®. Le concepteur peut l'intégrer sans programmation, pour cela, il doit suivre un processus dans lequel différents outils mettent en œuvre la méthodologie d'intégration (*encapsulations, compositions, projections*). Ce processus est détaillé en *Annexe II*, mais nous pouvons le résumer par la *Figure 62*:



- TemplateEditor : assistant d'intégration de logiciels pilotés par fichier.
- CMDEditor : assistant d'intégration de logiciels exécutables en ligne de commande.
- Workflow : *VisualComposer* + module de workflow, permettant de décrire l'enchaînement des modules du logiciel Flux2D.
- ABC2COB : projeteur de composants abstraits vers la norme de composants COB pour le dimensionnement,

Figure 62 : Processus d'intégration du calcul d'inductance de fuite en un composant de calcul électromagnétique mixte

Une fois ce composant COB généré, permettant de calculer l'inductance de fuite, les trois modèles (*2 analytiques + 1 numérique*) sont alors composés dans l'outil *VisualComposer* (*cf. Figure 63*), permettant de générer le nouveau modèle de calcul électromagnétique qui sera substitué à celui du modèle purement analytique du transformateur.

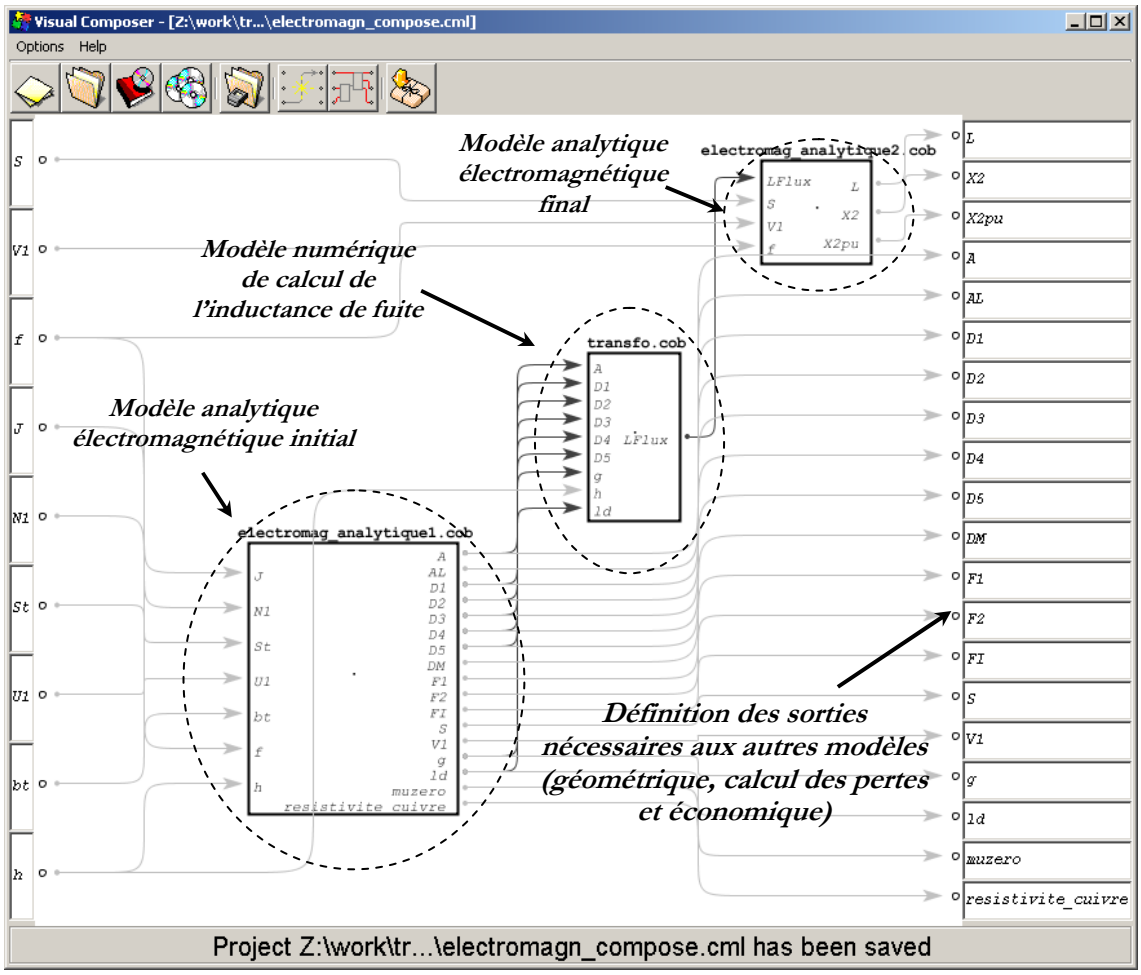


Figure 63 : Modèle électromagnétique composé de deux modèles analytiques et d'un modèle numérique de calcul de l'inductance de fuite.

A partir de là, le composant de calcul résultant peut être utilisé dans les services destinés aux composants COB, comme le service d'optimisation. Mais avant de dimensionner ce modèle, il peut être utilisé par un service de calcul simple dans lequel il peut être comparé au modèle purement analytique dont nous disposons.

I.C. Utilisation du composant dans un service de calcul simple

I.C.1. Utilisation du concept de passerelle pour comparer les modèles de calcul

Disposant des deux modélisations (*purement analytique et mixte*), nous pouvons tester les composants dans un outil de calcul. Les services disponibles dans l'environnement s'appuient sur des normes de composants spécifiques, mais il est également possible d'utiliser des services existants (*en dehors de l'environnement*) qui ne travaillent pas avec le patron de travail composants. Les composants de l'environnement peuvent, en effet, être utilisés dans des « services externes » grâce à la notion de **passerelle** (*cf. Figure 64*). Cette

notion, au même titre que celle des assistants d'intégration qui permet de plonger les outils dans l'environnement, permet d'extraire les composants de l'environnement pour exploiter leurs capacités dans d'autres outils.

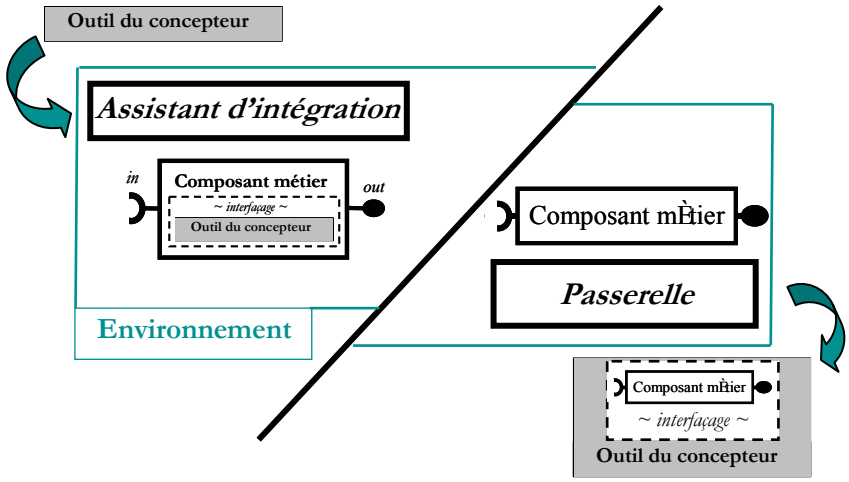
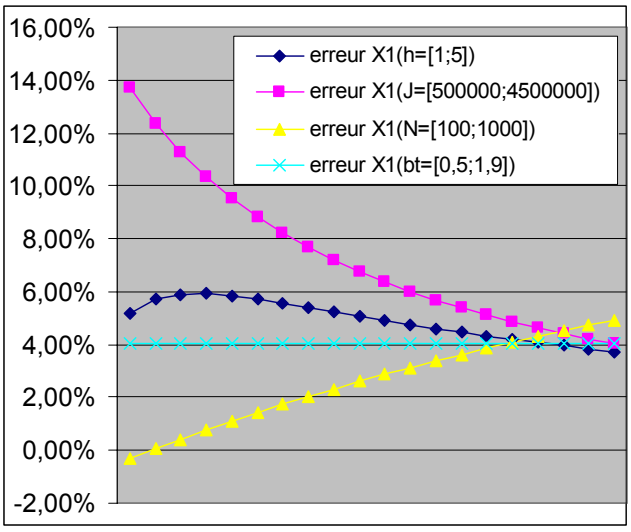


Figure 64 : Notion de passerelle : Symétrique par rapport à la notion d'assistant d'intégration, elle permet d'utiliser un composant dans un outil externe à l'environnement.

Ainsi, nous avons utilisé les capacités de calcul des composants par un simple programme informatique définissant une boucle autour d'une méthode appelant le composant et lui demandant de se calculer. De la même façon, le composant peut être piloté dans un outil tels que MatLab® ou Excel® et tracer les courbes de valeur souhaitées (*cf. Figure 76*).



Dans notre expérience, les résultats du calcul numérique sont proches de ceux du calcul analytique. Pour un point de fonctionnement particulier (*grandeurs optimales du transformateur définies par l'article [POL 86]*), le modèle analytique montre une sous estimation de 4%. Malgré tout, sur des plages de variations relativement grandes, les deux calculs montrent des différences plus importantes (*cf. Figure 65*).

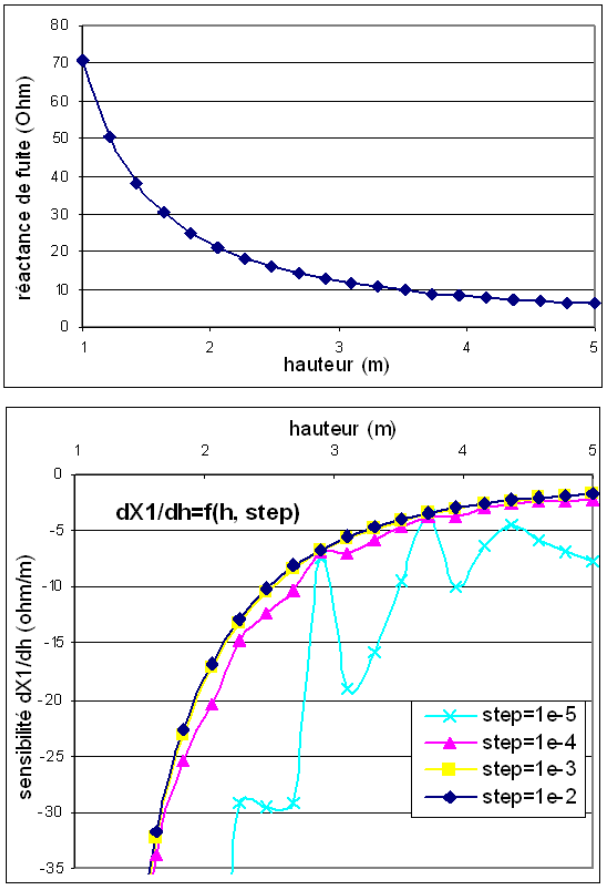
Figure 65 : Erreur (calcul numérique - calcul analytique) en % du calcul de la réactance de fuite X1. Chaque courbe correspond à la variation d'un des 4 paramètres (h∈[1,5] en mètres, J∈[500000,4500000] en A/mm², N∈[100,1000] en tours, Bt∈[0.5,1.9] en Tesla [17]) les autres étant fixés aux valeurs suivantes : h=4.432m, J=4500000A/mm², N1=800, Bt=1.7T

[17] Pour plus d'information sur ces paramètres, se référer à l'*Annexe III* détaillant le dimensionnement du transformateur

Il est important de connaître le domaine de validité d'un modèle que l'on souhaite utiliser pour du dimensionnement car un algorithme d'optimisation peut utiliser des points de fonctionnement en dehors de ce domaine si celui–ci ne lui est pas bien spécifié.

I.C.2. Etude du calcul de sensibilité du modèle mixte

Nous pouvons également vérifier le comportement du modèle de dimensionnement, et notamment le calcul de sensibilité (*c'est à dire la relation quantitative des paramètres de sortie par rapport à ceux d'entrée du modèle*), dont la qualité est un critique important pour notre but de conception. Le calcul de sensibilité est ici effectué par une méthode de type différence finie, défini par l'algorithme choisi lors de la projection à sémantique augmentée (*ABC2COB*) des composants abstraits vers la norme de composant COB. Cet algorithme peut être configuré durant la phase de projection, c'est notamment à ce moment que le pas de la différence finie est choisi.



Si le comportement du modèle n'est pas celui désiré (*ex : le pas de différentiation est trop faible, cf Figure 66*), le concepteur peut « rejouer » l'intégration de manière semi–automatique, n'ayant à intervenir que pour la redéfinition du pas de calcul des différences finies.

Figure 66 : Etude de la qualité du calcul de sensibilité de la réactance de fuite (Ω) en fonction de la hauteur du transformateur (m) pour une méthode par différence finie selon 4 valeurs de pas.

I.D. Utilisation du composant dans un service d'optimisation

Grâce au service d'optimisation disponible pour la norme de composant dans laquelle nous nous sommes intégrés, il est maintenant possible de dimensionner le transformateur. Nous avons réalisé ce dimensionnement, sur la base de différents cahiers des charges plus ou moins contraints (*cf. Annexe III*), conduisant à des différences de solutions à la mesure de la différence du calcul de l'inductance de fuite. Le coût du transformateur optimum (*objectif à*

minimiser) est par exemple légèrement supérieur (*environs 0,5% : 1300€*) car la sous estimation de l'inductance de fuite du modèle purement analytique conduisait à des solutions théoriquement meilleures.

La disponibilité de nos deux modèles, analytique et mixte, va nous permettre de comparer les deux approches en terme de durée d'optimisation et de facilité de convergence (*cf. Tableau 2*). Ainsi, selon les résultats décrits dans ce tableau, le nombre d'itérations entre les deux modèles reste du même ordre de grandeur, malgré une précision meilleure sur le calcul de sensibilité du modèle analytique (*dérivation symbolique des formules*). Les différences minimes peuvent toutefois s'expliquer (*cf.* a*nnexe III*), c'est notamment le cas pour le premier cahier des charges. Il s'explique en effet par une contrainte d'égalité sur la réactance de fuite, devant converger à 7.2000 ± 10^{-4} **Ω**, par rapport à la qualité du calcul de sensibilité du modèle numérique (*différences finies de pas* 10^{-2}).

Tableau 2 : Résultats d'optimisation des 5 CDCs, en terme de temps et d'itérations. Les calculs ont été réalisés sur un PC Pentium IV 1GHz, 512 Mo RAM.

		Modèle analytique	Modèle Mixte
CDC1	Nb d'itérations	11	18
	temps (sec)	0,21	8183(2h16)
CDC2	Nb d'itérations	16	12
	temps (sec)	0,28	5300(1h28)
CDC3	Nb d'itérations	11	11
	temps (sec)	0,21	5000(1h23)
CDC4	Nb d'itérations	7	8
	temps (sec)	0,16	3100(51min)
CDC5	Nb d'itérations	7	7
	temps (sec)	0,16	3100(51min)

Enfin, la différence la plus flagrante concerne le temps de calcul, qui est bien supérieur dans le modèle mixte que dans le modèle analytique. On pouvait supposer cela à cause de la résolution numérique, pourtant, elle n'est pas la plus grosse consommatrice de temps dans ce modèle. En effet, la plus grande partie du temps s'écoule dans le pilotage du logiciel éléments finis, notamment dans l'affichage des différents modules durant la re-paramétrisation du problème (*~70% du temps pour ce modèle*). Un plus gros modèle ne devrait donc pas beaucoup augmenter l'ordre de grandeur du temps de calcul présenté ici.

I.E. Conclusion

La méthodologie de composition nous a tout d'abord permis de définir un modèle composé à partir des différents modèles analytiques (*électromagnétique, géométrique, pertes, économique*). Cette architecture modulaire offre notamment la possibilité de substituer des composants à d'autres, de manière simple et rapide. Elle offre également une vision plus claire du modèle sur lequel le concepteur travail, grâce à la visualisation graphique des entrées et sorties de chaque modèles et de leur interrelations.

Ensuite, les méthodologies d'intégration et de composition nous ont permis de définir un modèle mixte du dispositif, sous la forme d'un composant de dimensionnent. Ce composant a été construit sur la base de modèles de domaines différents (*électromagnétique, économique,* …) mais aussi de nature différente (*analytique, numérique*).

Des résultats de calculs réalisés à partir des deux composants ont été tracé dans Excel®, ré–exploitant des capacités de post–processing externes grâce à la notion de passerelle. Enfin, une fois le composant mis au point par rapport au calcul de sensibilité, il a pu être utilisé dans un service de dimensionnement de l'environnement, afin de trouver une solution optimale selon les contraintes exprimées par un cahier des charges.

Cette gestion de la complexité des dispositifs à concevoir, grâce à une prise en compte de différentes manières de modéliser un dispositif, peut également être poussée plus loin. C'est ce que nous allons maintenant aborder dans la mise en œuvre d'un processus de dimensionnement multi niveaux de modélisation.

II. Méthodologie de dimensionnement multi niveaux

II.A. L'exploration de l'espace des solutions

Le dimensionnement par technique d'optimisation sous contrainte est une méthode cherchant une solution optimale dans un espace de solutions. L'espace des solutions est défini par le cahier des charges du dimensionnement, changeant nécessairement au gré des contraintes. Un modèle de dimensionnement doit donc être valide dans un grand domaine, afin d'être utilisé pour différents cahiers des charges. Certains algorithmes[18] (*cf [VAS 94a][CAL 01] pour un état de l'art*), dont ceux déterministes utilisés dans notre étude, demandent au concepteur de fournir un point initial pour débuter l'exploration. Il peut arriver qu'une distance trop grande de ce point par rapport à la solution optimale, ou simplement par rapport à l'espace des solutions, conduise l'algorithme à des minima locaux, à des solutions sortant du domaine de validité du modèle, voir même à aucune solution.

Afin de pallier ces problèmes d'exploration de l'espace des solutions, des solutions peuvent être envisagées à partir de processus mettant en œuvre plusieurs modèles du dispositif.

Nous pouvons par exemple considérer des modifications du modèle à l'issu de chaque dimensionnement, lorsque la solution modifie les hypothèses de modélisation. Ainsi, l'exploration se fait de manière itérative avec un modèle modifiant son domaine de validité en fonction des solutions successives (ex : *stratégies d'optimisation par plans d'expériences [VIV 02]*).

Dans le même ordre d'idée, nous allons détailler une solution qui exploite une méthode d'exploration à plusieurs niveaux de modélisation.

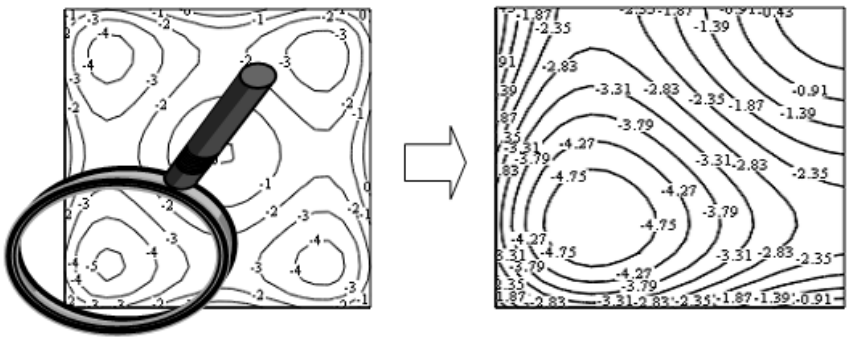
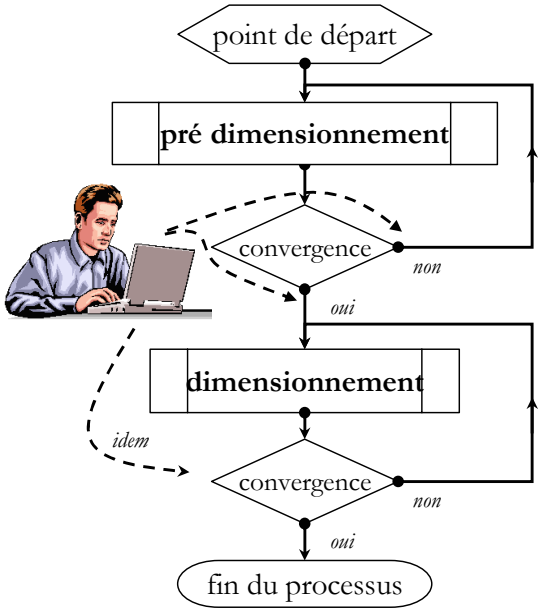


Figure 67 : Métaphore du microscope : on examine grossièrement un large espace de solutions, puis on focalise sur la solution trouvée pour l'affiner

[18] Optimisation Technology Centre: NEOS Guide Optimization Tree. www-fp.mcs.anl.gov/otc/guide/optweb/index.html

II.B. Métaphore du microscope

Grâce aux possibilités offertes par l'architecture de notre environnement, nous pouvons imaginer utiliser différents niveaux de modélisation afin de travailler sur des espaces de solutions différents. L'idée consiste à explorer l'espace à partir d'un modèle « grossier » possédant un domaine de validité très large et des capacités d'exploration rapides. Ensuite, reprenant comme point de recherche initial la solution trouvée précédemment, une exploration peut être réalisée à partir d'un modèle plus fin, présentant des capacités d'exploration beaucoup plus réduite, mais une précision plus importante.



Ainsi, pour un dispositif ayant besoin d'être dimensionné relativement finement à partir de divers cahiers des charges, la méthodologie de la métaphore du microscope doit offrir une solution rapide et efficace. Nous allons maintenant montrer comment une telle solution peut être mise en œuvre dans l'environnement de conception pour le dimensionnement du transformateur, notamment grâce aux méthodologies d'intégration et de composition.

Figure 68 : Processus de dimensionnement : métaphore du microscope

II.C. Définition du processus de dimensionnement associé à la métaphore du microscope

Nous pouvons définir le processus de dimensionnement associé à cette méthodologie en définissant un workflow. Celui-ci débutera par une étape d'optimisation sur le modèle analytique autour de laquelle on pourra réitérer jusqu'à ce qu'un point faisable soit trouvé, puis ce résultat sera fourni comme point initial d'une optimisation sur le modèle mixte (*cf. Figure 69*).

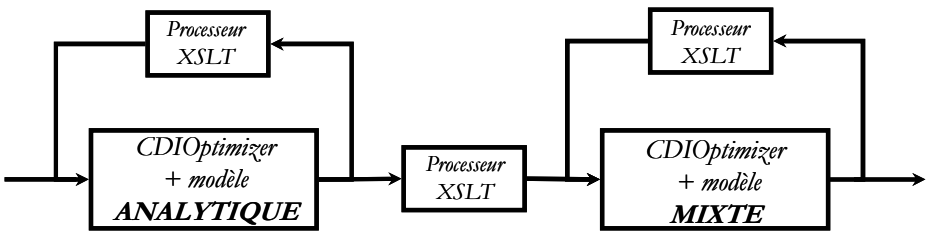


Figure 69 : Description du workflow : métaphore du microscope

Afin de réaliser pratiquement ce workflow, les différentes étapes doivent être intégrées dans des composants abstraits. Pour cela, nous avons utilisé notre assistant d'intégration *CMDEditor (utilisé précédemment pour intégrer le logiciel Flux2D)*, qui intègre des logiciels pilotables par ligne de commande. Deux outils ont été intégrés pour réaliser ce workflow :

- Le service d'optimisation (*CDIOptimizer : développé dans l'équipe CDI par David MAGOT*). Celui-ci est paramétré par un fichier XML et fournit les résultats d'optimisation dans un fichier XML.
- Un traducteur du résultat d'optimisation en point initial : A partir d'une description de correspondance XSLT (*eXtended Stylesheet Language Transformation*), il est possible de traduire une structure XML en une autre. Ainsi, à partir du ficher de résultat, un fichier de configuration, dont le point initial correspond à la solution trouvée, peut être généré.

Une fois les composants abstraits disponibles, le processus est décrit dans l'outil de composition VisualComposer (*cf. Figure 70*) permettant de créer le composant d'exécution du workflow ainsi décrit.

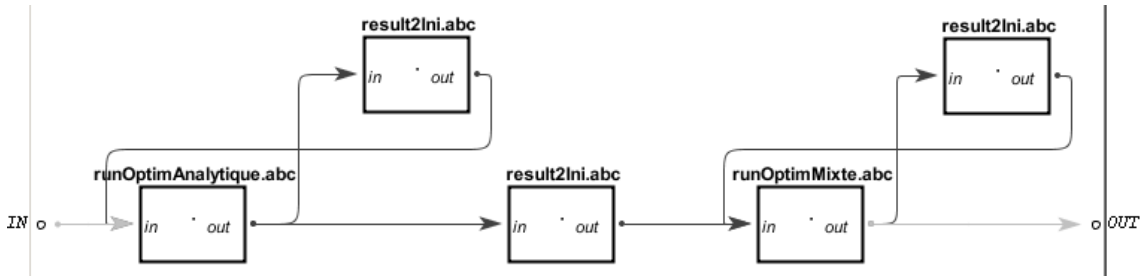


Figure 70 : Mise en œuvre du workflow : métaphore du microscope

II.D. Réduction des temps de dimensionnement

Par rapport à l'optimisation directe par le modèle mixte du transformateur, l'objectif de ce processus de dimensionnent n'est pas de réussir à le faire converger puisque les cahiers des charges ne montrent pas de problèmes de convergence. En revanche, nous pouvons montrer sur cette application, que ce processus est une solution pour réduire les temps de dimensionnement d'un facteur 3 ou 4 (*cf. Tableau 3*) .

Tableau 3 : Optimisations des 5 cahiers des charges, en terme de temps et d'itérations, comparaison avec la métaphore du microscope. Les calculs ont été réalisés sur un PC Pentium IV 1GHz, 512 Mo RAM.

		Modèle Mixte	Métaphore du microscope
CDC 1	Nb d'itérations	18	5
	temps (sec)	8183 (2h16)	2200 (37min)
CDC 2	Nb d'itérations	12	5
	temps (sec)	5300 (1h28)	2200 (37min)
CDC 3	Nb d'itérations	11	3
	temps (sec)	5000 (1h23)	1300 (22min)
CDC 4	Nb d'itérations	8	2
	temps (sec)	3100 (51min)	840 (14min)
CDC 5	Nb d'itérations	7	5
	temps (sec)	3100 (51min)	2150 (36min)

Les quelques itérations du modèle mixte dans l'application de cette méthodologie permettent simplement de caler le calcul de l'inductance de fuite par rapport aux imperfections du modèle analytique (*cf. Figure 71*).

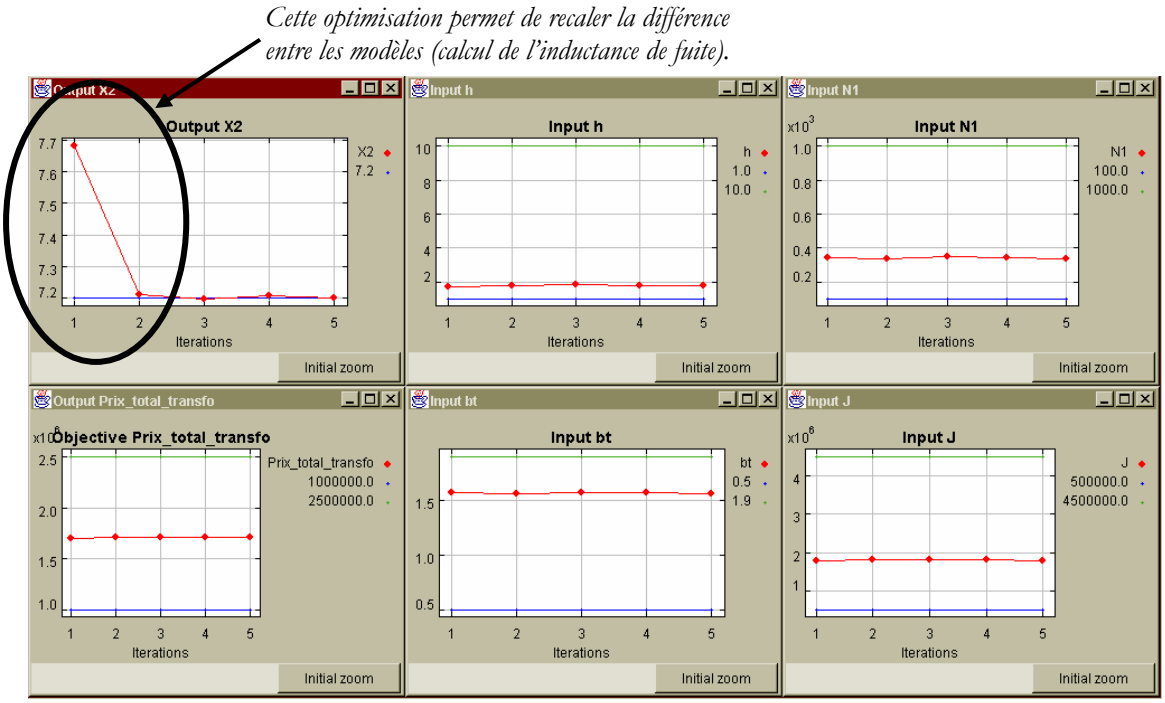


Figure 71 : CDC1 modèle mixte, point initial correspondant à la solution optimale du modèle analytique, précision de fin 10^{-4}, précision des différences finies 10^{-2}, 37 minutes, 5 itérations

III. Conclusions

Dans ce chapitre, nous avons pu mettre en évidence différentes utilisations des méthodologies proposées et des outils associés assurant au concepteur une utilisation sans programmation :

- Composition de composants décrits analytiquement (*électromagnétique, géométrique, pertes, économique*),
- Intégration d'outils de simulation pour réutilisation de ses capacités de calcul (*encapsulation d'un calcul élément finis*),
- Projection de composant de simulation vers un type de composant pour le dimensionnement,
- Composition de modèles mixtes (*analytique/numérique*),
- Utilisation d'un service existant de dimensionnement et d'optimisation.
- Définition d'un processus de dimensionnement multi-niveaux

Les méthodologies de travail mises en œuvre durant le dimensionnement du transformateur (*métaphore du microscope,* …) ne sont que très peu utilisées par les concepteurs, en raison d'un manque manifeste d'environnements tel que celui que nous proposons. Dorénavant, grâce au concept de composant, aux méthodologies d'intégration et de composition permettant de définir des modèles composés et des descriptions de processus, le concepteur pourra mettre en œuvre ses processus de conception de manière rapide et intuitive.

PARTIE II :

AUTRES APPLICATIONS

D'UNE ARCHITECTURE COMPOSANTS

Durant cette thèse, nous avons pu travailler en collaboration avec différentes personnes tout en mettant en œuvre le principe de notre architecture. Nous allons donc regarder brièvement deux exemples de l'utilisation des concepts décrits dans nos travaux, tels que ceux d'une architecture composants, d'une intégration d'outils, et d'une composition de modèles.

I. Conception d'un micro actionneur bistable magnétique

I.A. Contexte : besoin de tester rapidement des solutions

Les chercheurs en conception de micro-systèmes magnétiques de l'équipe Machines du LEG, ont besoin d'outils pour tester et valider les structures qu'ils proposent, sachant que dans ce domaine qui émerge, il est plus fréquent de proposer une nouvelle structure que d'en simuler une.

Considérons un actionneur bistable à concevoir en utilisant les technologies des MAGMAS (*Magnetic Micro Actuator and Systems*). Le cahier des charges imposant une consommation d'énergie nulle au repos, la structure peut être faite d'aimants pour stabiliser les deux positions, et de bobines pour provoquer le mouvement (*cf. Figure 72*)

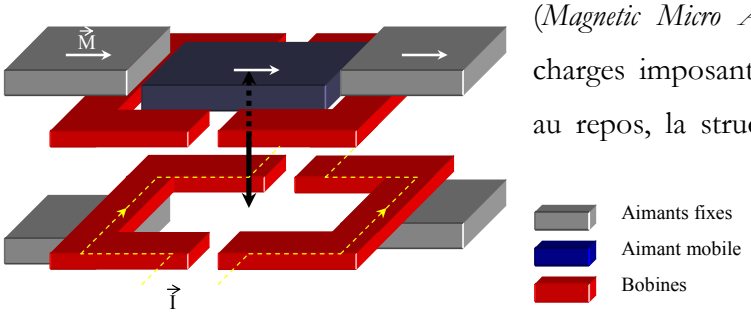


Figure 72 : Structure d'un micro actionneur bistable MAGMAS, thèse Hervé Rostaing

A ce niveau de la conception, il est intéressant de vérifier son fonctionnement théorique et de chercher à dimensionner la structure pour savoir si elle respecte le cahier des charges en terme d'encombrement ou de courant consommé. Si la structure n'est pas suffisamment performante, on doit pouvoir rapidement voir comment elle se comporte si on lui adjoint

des parties en fer, si on double les couches de bobinages, si on place des aimants sur un axe parallèle…

I.B. Modélisation pour le dimensionnement

I.B.1. Intégration d'un outil de calcul de forces magnétiques

Afin de tester cette structure, nous pouvons utiliser un logiciel simulant les efforts exercés sur le mobile par les différentes forces magnétiques en présence. Le logiciel Dipole3D [DELA 93] conçu par Jérôme Delamare du LEG, permet de décrire facilement des objets (*aimants, conducteurs*) et de calculer ces efforts.

Afin d'intégrer dans des composants écrits en langage de programmation Java, ce logiciel écrit dans le langage C, il est nécessaire de gérer cette hétérogénéité (*cf. Annexe IV*). Le code de calcul Dipole3D est tout d'abord interfacé dans le langage C, puis cette interface est automatiquement projetée dans un COB par un outil de projection (*JNI2COB ou CORBA2COB, cf. Annexe IV et [DEL 02a]*).

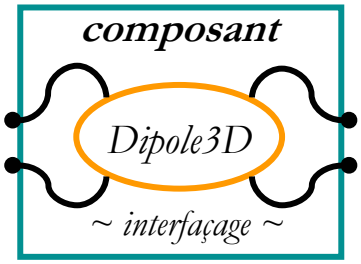


Figure 73 : Encapsulation de dipole3D dans un composant

Cette encapsulation du logiciel Dipole3D dans un composant nous permet de l'utiliser dans divers services tels que ceux de post-processing. Malgré tout, ce que nous souhaitons faire est de mettre en œuvre un modèle de calcul pour le dimensionnement.

Deux solutions s'offrent à nous, la première consiste à réaliser une projection avec une augmentation de sémantique par l'ajout de la fonctionnalité du calcul des sensibilités (*ex : différences finies*). Cette première solution permet de rapidement mettre en œuvre le modèle sans difficultés majeures. Pourtant, dans ce travail, nous avions également accès à un modèle analytique du micro-actionneur s'appuyant sur le calcul d'intégrales volumiques.

La deuxième solution consiste donc à mettre en œuvre ce modèle analytique dans un composant de dimensionnement en définissant les fonctions de calcul du modèle et celles du calcul formel de sensibilité. Ainsi, nous disposions d'un modèle beaucoup plus performant dont l'optimisation reste dans des temps raisonnables (*~20min pour 13 itérations*). Le composant de calcul encapsulant Dipole3D a alors permis de contrôler la validité de ce dernier composant de calcul.

I.B.2. Intégration du calcul des grandeurs contraintes

Le cahier des charges du dispositif cherche à minimiser le courant, tout en respectant une certaine tenue aux chocs (*150 m.s^{-2}*) et en permettant le déplacement du mobile (*la force minimale, sur tout le déplacement du mobile, doit être positive*).

Le courant est un paramètre d'entrée du modèle, quant à la tenue aux chocs, elle est un paramètre de sortie calculé par le composant défini précédemment. Ces deux contraintes peuvent donc être directement gérées dans un service d'optimisation. Par contre une contrainte pose problème, celle devant assurer une force positive sur tout le déplacement du mobile. Il est en effet nécessaire de calculer la valeur minimale de cette force pour la contraindre.

Une solution consisterait à re-développer un algorithme de recherche de maximum que l'on placerait avec les fonctions de calcul du composant. Cette démarche serait longue et donc coûteuse pour une démarche dont l'objectif n'est que mettre rapidement à l'épreuve une structure, grâce au dimensionnement. D'autant plus, que cet algorithme ne doit pas se faire piéger par des optimums locaux (*cf. Figure 74*).

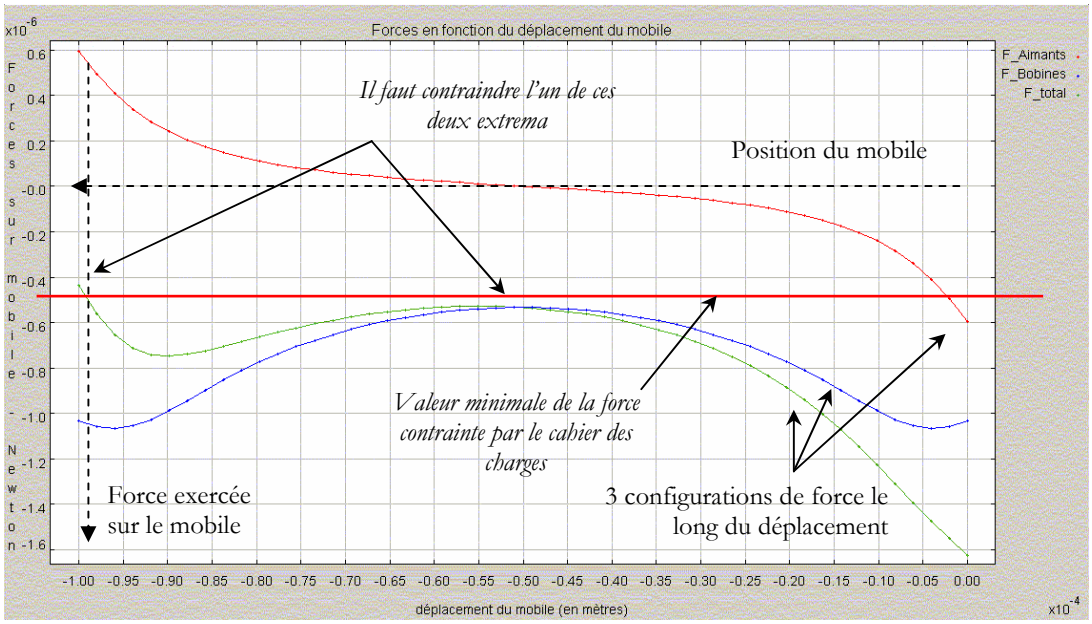


Figure 74 : Besoin de calcul du maximum global de la force sur tout le déplacement du mobile

Grâce à l'architecture composant de l'environnement de conception, il va être possible de traiter rapidement ce cas de conception sans programmation nécessaire. La solution consiste à réutiliser les capacités d'optimisation de l'outil CDIOptimizer (*thèse David Magot*) possédant notamment un algorithme de recherche de maximum global. Pour intégrer

CDIOptimizer dans un composant, un assistant d'intégration direct vers la norme de composant COB (*ne s'appuyant pas sur l'intermédiaire des composants abstraits*) a été développé par David Magot [MAG 03] (*cf. Figure 75*).

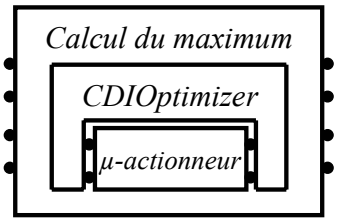
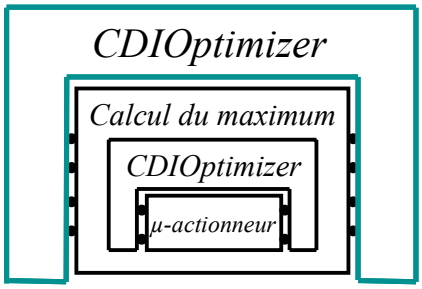


Figure 75 : Encapsulation de CDIOptimizer pour réaliser le calcul du maximum sur le composant de calcul Dipole3D.



Le composant résultant permet alors de calculer le modèle ainsi que les grandeurs contraintes liées au dimensionnement. Il peut ensuite être utilisé dans le service de dimensionnement (*CDIOptimizer*) afin de mettre à l'épreuve la structure proposée pour le micro actionneur à concevoir.

Figure 76 : Utilisation du modèle de dimensionnement (modèle de calcul + calcul des grandeurs à contraindre telles que la valeur maximale de la force) dans le service de dimensionnement

I.C. Optimisation

Le cahier des charges cherchait principalement à contraindre une tenue aux chocs et une force minimale durant le déplacement de la partie mobile, tout en minimisant le courant nécessaire à son fonctionnement. En effet, la miniaturisation et l'autonomie entraînent nécessairement des contraintes très fortes sur les alimentations des microsystèmes.

Partant d'une structure dimensionnée grossièrement, grâce à quelques itérations manuelles dans un outil d'analyse, nous avons utilisé le composant dans le service de dimensionnement. Grâce à un mécanisme de connexion spécifique (« *plug-in », dont le principe est expliqué en Annexe 4*), nous pouvons exploiter les résultats d'optimisation directement dans un autre outil (*MathCAD*™). C'est ainsi que nous pouvons par exemple visualiser une vidéo de l'évolution de la géométrie et des paramètres importants au long des itérations du dimensionnement (*cf. Figure 77*).

On peut ainsi voir évoluer la géométrie, afin d'appréhender rapidement les voies qu'explore l'algorithme d'optimisation, ou simplement, afin de faire coïncider une contrainte violées à la géométrie (*chevauchement de pièces*…).

L'évolution de la fonction objectif (*courant*), de la tenue au choc, ainsi que de la force minimale, peuvent également être visualisées. Une dernière évolution est visualisée, il s'agit de la position du mobile donnant la force minimale précédente, permettant ainsi de vérifier

que l'algorithme trouve tantôt des valeurs autour de la position milieu, tantôt des valeurs proches de l'extrémité gauche de la *Figure 74*.

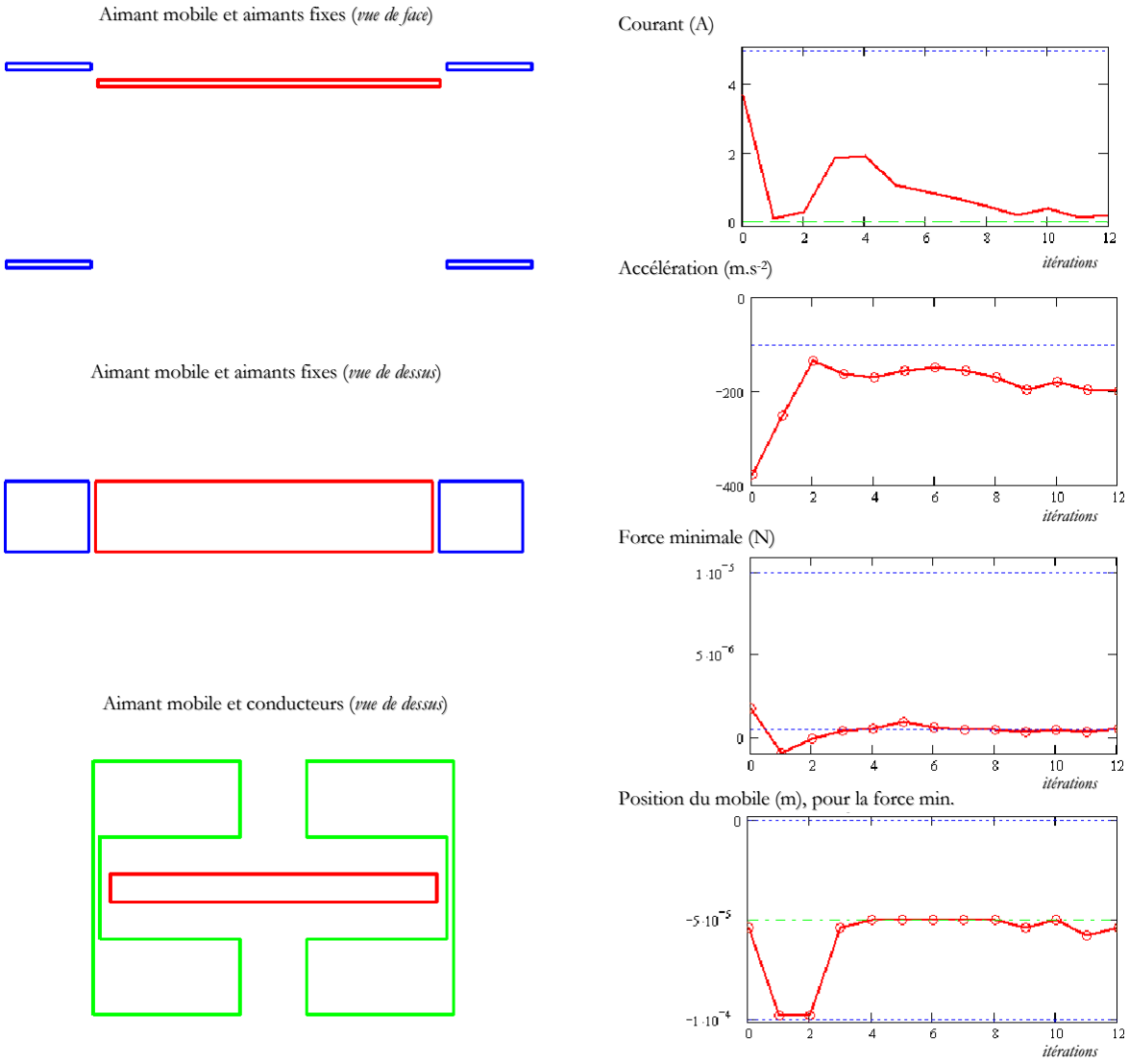


Figure 77 : Visualisation des résultats d'optimisation dans le logiciel MathCAD, permettant d'appréhender le comportement du dimensionnement.

Ce dimensionnement donne donc une solution répondant au mieux aux contraintes que l'on impose dans le cahier des charges. Une première constatation, en examinant les géométries initiales et finales (*cf. Figure 78*), concerne la forme des conducteurs disproportionnée. Cette géométrie optimisée met en évidence les parties fonctionnelles (*courants horizontaux*), et les autres (*verticaux*), uniquement limitées en épaisseur par la densité de courant minimale définie dans le cahier des charges.

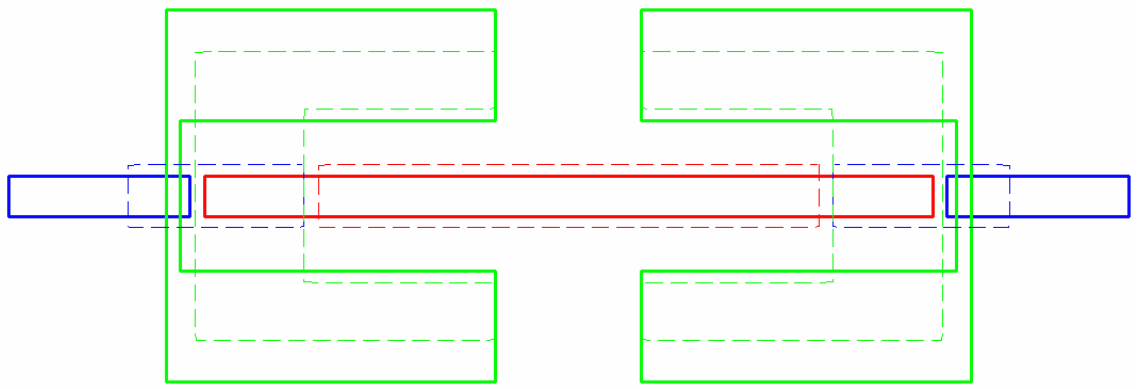


Figure 78 : Comparaison de la géométrie initiale (en pointillées) et de la géométrie optimale (trait continus), représentant l'aimant mobile (rectangle central), les aimants fixes (rectangles à gauche et à droite) et les conducteurs (entourant l'aimant mobile).

I.D. Conclusion

Grâce à la notion de composant et à ses capacités de récursivité, nous avons vu qu'il était possible de créer rapidement des modèles de dimensionnement prenant en compte la complexité liée à l'activité de conception (*calcul d'un maximum* à *contraindre*).

Une connexion (*type « plug-in », cf. Annexe IV*) à l'outil MathCAD a permis de réaliser un post-processing de l'optimisation contenant, notamment, une géométrie simpliste mais importante à la maîtrise de modèle du dimensionnement. Cette connexion, facilement reconfigurable pour un modèle de dimensionnement contenant de nouveaux paramètres, permet de tester rapidement de nouveaux modèles ou cahiers des charges.

Cette dynamique, ainsi que celle offerte par l'environnement à base de composants, nous permet de tester des structures innovantes de manière simple et très rapide, sans soucis de l'échec.

II. Conception collaborative d'un déclencheur électromécanique

II.A. Contexte : besoin de travailler avec d'autres domaines disciplinaires

La conception d'un dispositif électrotechnique est souvent délicate en raison de sa nature multi-physique. En tant que concepteur en génie électrique, nous pouvons maîtriser les difficultés liées à notre domaine. Concernant d'autres domaines, tels que celui la mécanique qui présente des problèmes de résistance des matériaux ou de fabrication, nous travaillons généralement seul en traitant le problème avec nos propres connaissances, ou pire, nous l'esquivons.

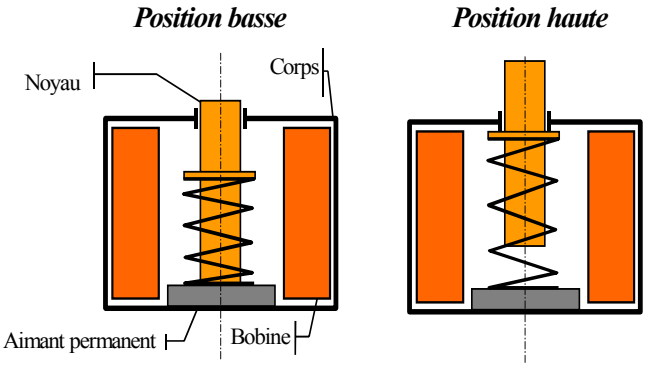
La conception collaborative est une réalité industrielle, par la globalisation des entreprises, et la complexité croissante des systèmes à concevoir. La complexité de la conception liée à la complexité de la coopération donne lieu à des recherches de supports adaptés à ces besoins [BOU 00b][BOU 00a][CAR 97]. Nous avons eu l'occasion de participer à un projet de conception collaborative s'inscrivant dans le cadre de l'Institut de la Production et des organisations Industrielles (IPI). Ce projet nous a permit de collaborer avec des chercheurs en conception mécanique du laboratoire 3S[19] de Grenoble, donnant lieu à une expérience très enrichissante.

II.B. Conception de la structure : problèmes des différences culturelles

Le dispositif à co-concevoir était un déclencheur électromécanique depuis son cahier des charges fonctionnel jusqu'à son dimensionnement optimal. La conception de la structure a été instrumentée par des outils de dialogues formels et informels via Internet. A partir de la solution fonctionnelle (*cf. Figure 79*), des réunions, à distances ou présentielles, synchrones ou asynchrones, ont permis de définir une structure au déclencheur (*cf. Figure 80*).

Le résultat de ce travail a conduit au constat d'une différence de culture entre les métiers d'électrotechniciens et de mécaniciens, contraignant fortement le processus de collaboration [DEL 02c]. Des difficultés peuvent, par exemple, provenir de concepts différents associés à des entités identiques. Cela a notamment été le cas des entrefers (*définition des électromagnéticiens*) et des jeux de fabrication ou de fonctionnement (*définition des mécaniciens*), nécessitant, pour cette même entité, de s'expliquer mutuellement les contraintes spécifiques à chaque métier.

[19] 3S : Sols Solides Structures



Principe de fonctionnement du déclencheur :
Un aimant permet de maintenir le noyau en position basse. Lorsqu'un courant circule dans les bobines, le flux magnétique créé s'oppose à celui de l'aimant, réduisant la force de collage. Dès qu'il y a décollage, c'est le ressort qui assure la percussion en position haute.

Figure 79 : Structure fonctionnelle du déclencheur électromagnétique.

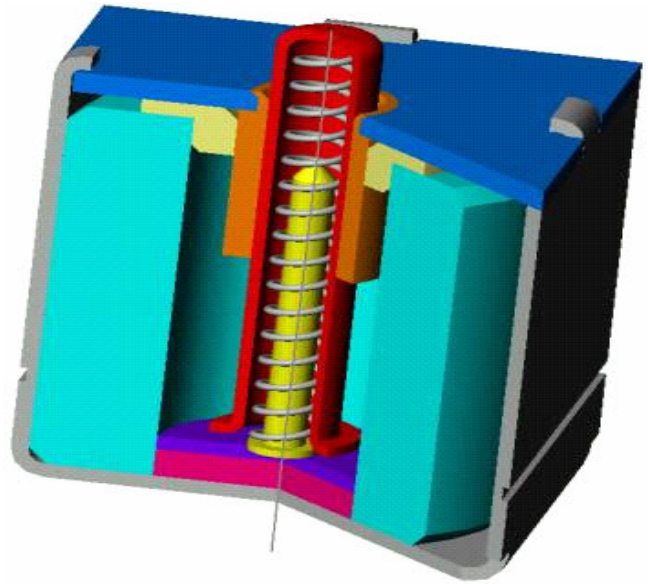


Figure 80 : Structure du déclencheur électromagnétique conçue de manière collaborative entre électromagnéticiens et mécaniciens.

II.C. Modélisation pour le dimensionnement

II.C.1. Dimensionnement dissocié

Une fois la structure déterminée, un modèle du dispositif a été construit. Les électromagnéticiens ont fourni un modèle analytique, s'appuyant sur un réseau de réluctance, qui calcule les forces magnétiques [ATI 00]. De leur côté, les mécaniciens ont fourni les paramètres de l'équation d'un ressort, calculant la force en fonction de la position du noyau mobile. Cette équation, de la forme $F(L)=k(L-L_0)$, a été intégrée au modèle électromagnétique.

A partir du générateur de composants de dimensionnement, ce modèle analytique a pu être utilisé dans le service de dimensionnement avec un cahier des charges intégrant des

contraintes d'encombrement, de tenue aux chocs, de temps de déclenchement, d'énergie de percussion, etc.

Malheureusement, le ressort ayant été dimensionné de manière isolée, il a été impossible de trouver une structure magnétique qui permette de maintenir le noyau en position basse tout en respectant les contraintes d'encombrement du cahier des charges (*cf. Figure 81* [20]). La relaxe des contraintes d'encombrement permet de voir que la structure optimale nécessite un circuit magnétique bien trop important (*cf. Figure 82*), justifiant que la source du problème provient du ressort surdimensionné.

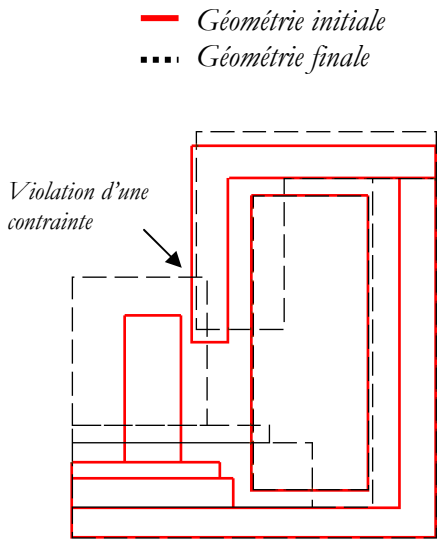


Figure 81 : Dimensionnement ne convergeant pas en raison de violations de contraintes. La cause vient du ressort trop puissant par rapport à la taille de l'aimant et du circuit magnétique.

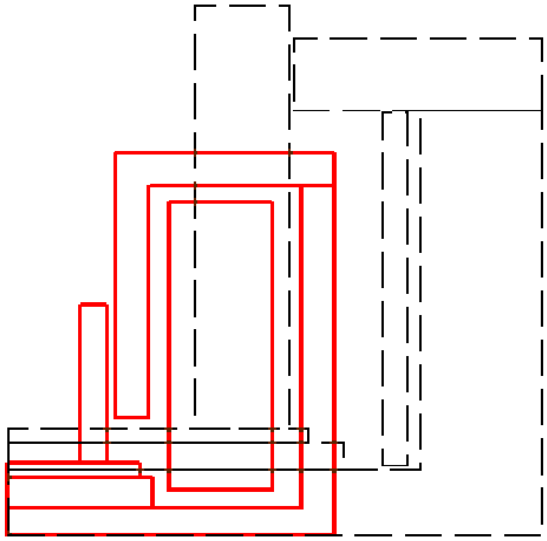


Figure 82 : Dimensionnement convergeant grâce à la relaxe des contraintes d'encombrement, permettant à l'aimant et au circuit magnétique de créer une force suffisante pour maintenir le noyau collé.

Afin de résoudre ce problème, une première solution consiste à redimensionner le ressort en approximant l'ordre de grandeur de la force optimale de collage. Ce processus qui nécessite un certain nombre d'itérations entre les électromagnéticiens et les mécaniciens, ne conduit pas à un dimensionnement optimal.

Afin de pallier cette difficulté, les mécaniciens ont du fournir un modèle de dimensionnement du ressort. Se pose alors le problème de l'intégration de ce modèle plus compliqué, comprenant plusieurs équations de calculs des paramètres de l'équation de la force du ressort, ainsi que des équations de contraintes pour le dimensionnement (*ratio entre grandeurs géométriques,* …). Le problème de coopération fait apparaître d'une façon plus

[20] Ces figures proviennent de vidéos d'optimisation, réalisées dans le logiciel MathCAD grâce à une connexion de type « plug-in » (cf Annexe IV : pilotage informatique de logiciels) aux données d'optimisation du logiciel CDIOptimizer.

générale la notion d'objet intermédiaire [LAU 97][BOU 00b], une entité partagée par les différents acteurs qui ferait office d'interface entre leurs connaissances.

II.C.2. Le composant : entité de collaboration

L'intégration de deux modèles d'équations donne lieu à des contraintes non nécessaires. En effet, l'intégrateur n'étant à priori pas omniscient, il n'est pas censé comprendre les équations des diverses parties. De plus, les modèles développés séparément risquent d'introduire des variables de même nom, mais ne représentant pas la même grandeur.

C'est alors que les deux caractéristiques essentielles du paradigme des composants prennent tout leur sens : « distinguer sans disjoindre, et associer sans identifier ou réduire ».

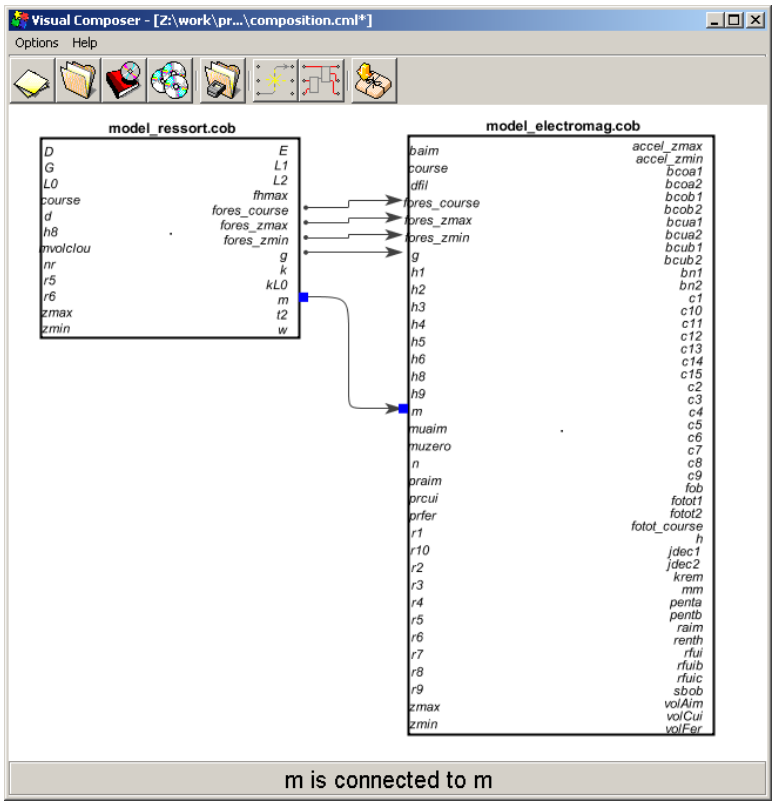


Figure 83 : Composition du modèle global par connexion des paramètres échangés

Nous avons donc choisi de développer le modèle global par composition de composants définis dans chacun des deux domaines (*cf. Figure 83*). Ainsi, l'intégration ne se fait que par les paramètres que « souhaitent échanger » les deux modèles, ne nécessitant de s'accorder que sur cette interface. Les paramètres intermédiaires des deux modèles restent locaux à leur propre composant, ne risquant pas d'interférer les uns avec les autres.

Le choix de travailler avec des composants définit les objets intermédiaires, que partagent les différents acteurs, comme les interfaces de ces composants. Les différents concepteurs n'ont pas à connaître le contenu du modèle de calcul, tant qu'ils se sont entendus sur la définition des interfaces du composant associé à ce modèle.

II.C.3. Dimensionnement intégré

La solution mise en oeuvre a permis de réaliser de manière simple le couplage entre les deux métiers[21]. Ainsi, un premier dimensionnement a pu être réalisé (*cf. Figure 84*), en considérant le ressort comme un composant optimisable associé à des contraintes de fabrication et de viabilité (*cf. encart*).

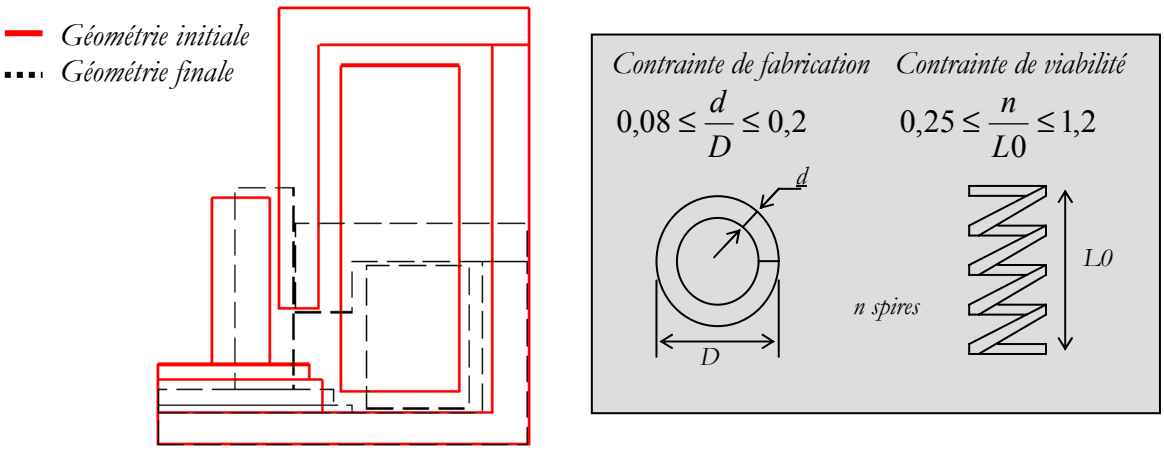


Figure 84 : optimisation de l'ensemble électromagnétique + ressort

Le potentiel offert par les composants a encore pu être exploité par la substitution du composant modélisant le ressort par un autre composant dont les caractéristiques sont définies dans un catalogue constructeur offrant un choix parmi plus de 1000 ressorts (*cf. Tableau 4*).

Tableau 4 : Catalogue constructeur de ressorts Lee Spring Company (www.leespring.com), seuls D, k et L0 ont été utilisé.

	A	B	C	D	E	F	G	H	I
1	Type	D	HD	WD	L0	k	SL	RD	Mat
2	LC-024A-12-M	3,05	3,18	0,61	23,8	4,272	15,72	1,68	MW
3	LC-024A-12-S	3,05	3,18	0,61	23,8	3,554	15,72	1,68	SS
4	LC-024A-13-M	3,05	3,18	0,61	25,4	3,992	16,74	1,68	MW
5	LC-024A-13-S	3,05	3,18	0,61	25,4	3,326	16,74	1,68	SS
1276	LCM-055D-10-M	7,49	8	0,56	24,99	0,525	4,114	6,22	MW
1277	LCM-055D-10-S	7,49	8	0,56	24,99	0,437	4,114	6,22	SS
1278	LCM-055D-11-M	7,49	8	0,56	27,51	0,472	4,419	6,22	MW
1279	LCM-055D-11-S	7,49	8	0,56	27,51	0,385	4,419	6,22	SS

[21] Ce travail a été réalisé durant le stage de Steven GOWERS : « Développement d'un environnement de co-conception distribuée, basé sur une étude entre cultures professionnelles distinctes » Stage de fin d'étude « MEng Innovation and Engineering Design with French », Université de Bath / INP Grenoble.

Cette modification du modèle du dispositif a donc été très simple, puisque seul le modèle du ressort a été modifié, ce qui a permis de générer un nouveau composant qu'il a fallu alors connecter à celui du calcul électromagnétique. Soulignons que ce dernier modèle du ressort nécessite d'être placé dans un service de dimensionnement supportant l'optimisation de paramètres discrets, nous avons pour cela utilisé le service CDIOptimizer.

II.D. Conclusion

Ainsi, grâce à l'architecture composant et à la capacité de composition de modèle de calculs, une partie de la complexité d'un processus collaboratif de conception a pu être gérée. Sa capacité de modularité, tout en privilégiant les associations entre modules, fait du composant une entité d'échange idéale pour la modélisation collaborative.

La mise en commun d'une interface de composant permet de s'accorder sur des bases formalisées. L'adoption de cet objet intermédiaire doit apporter un gain non négligeable en terme d'efficacité et donc de temps, modifiant implicitement les relations de travail entre les acteurs d'une conception collaborative.

CONCLUSIONS & PERSPECTIVES

C o n c l u s i o n s e t P e r s p e c t i v e s

I. Conclusions

I.A. Résumé des solutions apportées

Nous avons développé dans cette thèse un environnement de conception permettant de répondre aux critères de complexité, d'imprévisibilité et de dynamique de la conception.

Pour cela, nous avons mis en œuvre une architecture à composants qui s'appuie sur différents concepts tels que les générateurs et les services, la composition ou la projection.

Deux méthodologies principales ont été développées dans cette thèse :

La première concerne l'intégration des outils du concepteur dans l'environnement. Cela nous a permis d'introduire une norme de composant dédié à l'intégration. Dans ce cadre, nous avons développé différents outils afin de mettre en œuvre cette méthodologie et d'offrir au concepteur des solutions sans qu'il ait à programmer quoi que ce soit :

- *TemplateEditor* : assistant d'intégration s'appuyant sur un pilotage par fichier, générateur de composants abstraits.
- *CMDEditor* : assistant d'intégration s'appuyant sur un pilotage en ligne de commande, générateur de composants abstraits.
- *ABC2COB Projector* : projeteur à sémantique augmentée permettant d'utiliser un composant abstrait à partir d'un composant de calcul pour le dimensionnement (*COB*), ce projeteur permet entre autres d'ajouter la fonctionnalité de calcul de sensibilité.

D'autres projeteurs ont été développés pour des besoins particuliers dont nous n'avons fait que mentionner l'existence dans ce rapport.

La deuxième méthodologie concerne la composition. Elle nous a conduit à développer un outil de composition (*VisualComposer*) permettant, à partir de modules de composition, d'exploiter la mise en commun de composants.

- Un premier module a été développé autorisant la création de modèles de calcul composés, dédiés au dimensionnement.
- Un deuxième module permet au concepteur de décrire et de mettre en œuvre les processus de conception qu'il souhaite réaliser avec les différents composants qu'il a à sa disposition (*outils, modèles,* …).

I.B. Bénéfices de ces travaux de thèse

Nous pouvons résumer brièvement ce que ces deux méthodologies apportent au travail du concepteur :

- **Il peut désormais mettre à l'épreuve sa créativité par la mise en œuvre rapide de ses idées dans l'environnement**. Il peut en effet créer et modifier rapidement et sans programmation, des solutions informatiques (*modèles de calcul, enchaînement de tâches automatisées,* …) pour répondre à ses besoins de conception.
- **Il peut désormais mettre en œuvre de nouvelles méthodologies de conception permettant d'améliorer ses solutions**. En effet, les capacités de l'environnement à fédérer dynamiquement un panel d'outils offrent des moyens, qui n'existaient jusqu'alors pas, pour mettre en œuvre des méthodologies nouvelles.

I.C. Enseignements tirés de ces travaux

La mise en oeuvre de l'environnement que nous avons proposé, nous a tout d'abord permis de mettre en évidence certaines limites à l'approche.

I.C.1. Des normes adaptées à des besoins

La première vient de la spécification de normes de composants adaptés à des besoins spécifiques, pour lesquelles le concepteur n'est pas susceptible d'adhérer totalement. La nécessité de disposer d'un patron de travail (*générateurs, utilisateurs,* …), pour des besoins spécifiques (*une norme de composant*), fait qu'une nouvelle norme de composants impose un certain travail de développement (*projeteurs, module de composition*…).

Le besoin de composants de calcul pour le dimensionnement, qui a été identifié dans l'équipe CDI, a conduit au développement successif d'un certain nombre de générateurs, de services et de passerelles, qui est aujourd'hui bien valorisé mais qui constitue un certain investissement. Pour une nouvelle norme de composants, un premier investissement consiste à voir s'il est possible de réutiliser des instanciations existantes du patron de travail par l'utilisation de projeteurs. Si des services sont développés par rapport à cette nouvelle norme, il est alors important de développer des projeteurs pour que d'autres profitent de ces services et que l'investissement soit ainsi valorisé.

I.C.2. Limites nécessaires à la généricité

Certaines limitations peuvent apparaître dans la méthodologie d'intégration, concernant notamment la norme du composant abstrait qui peut ne pas répondre à des besoins spécifiques du concepteur. En effet, nous pouvons par exemple considérer qu'une interface

d'entrée unique (*un fichier*) est insuffisante et qu'il serait intéressant d'en proposer plusieurs. De plus, parmi les outils que nous avons développé (*Template Editor, ABC2COB Projector*), certains s'appuient sur une norme de fichier spécifique (*XML*), laquelle peut ne pas répondre aux besoins du concepteur.

Certaines limites technologiques viennent évidement réduire les possibilités de notre environnement qui ne peut pas intégrer n'importe quels outils. Nous avons définit les assistants d'intégration comme dépendants des mécanismes de pilotage que proposent l'outil à intégrer. Ces difficultés technologiques liées au pilotage de logiciel (*cf. Annexe IV*), peuvent se caractériser comme un frein à l'utilisation de l'environnement de conception.

II. Perspectives

II.A. Evolution des outils

Afin de promouvoir l'architecture proposée, il est nécessaire de continuer à offrir des outils au concepteur. Il faut notamment augmenter le panel d'assistants d'intégration grâce aux différents moyens d'accès des logiciels à piloter (*cf. Annexe IV*). Il sera également important de développer de nouveaux modules de composition tels que ceux permettant de gérer les boucles dans les modèles de calcul, ou d'autres permettant des modes de propagation différents du calcul de sensibilité, etc.

Il sera également nécessaire de faire évoluer les outils pour chaque nouvelle déclinaison du patron de travail. C'est-à-dire qu'à chaque nouvelle norme de composant, correspondant à un nouveau besoin, il sera nécessaire de programmer des projections vers les patrons appropriés ainsi que le module de composition associé à cette nouvelle norme.

II.B. Capitalisation par bibliothèques de composants

S'appuyant sur le concept des composants logiciels métiers, l'architecture proposée permet de plus grandes possibilités de réutilisation. En effet, le concept de « boite noire » permet la réutilisation de fonctionnalités (*ex : implémentation d'un modèle de calcul*) capitalisées dans les différentes normes de composants. Les différents concepts mis en œuvre dans ce travail de thèse par rapport aux « boites noires » doivent également pouvoir s'appliquer aux « boites blanches » offrant des méthodes de réutilisation complémentaires à notre approche.

Cette architecture doit notamment bénéficier de bibliothèques de composants, lesquels constitueraient des briques de bases afin de mettre en œuvre des systèmes plus complexes. Des travaux visant à étudier de manière systématique les modèles d'actionneurs électromécaniques [JUF 95][HAT 94][PER 94] doivent, par exemple, permettre de réaliser ces bibliothèques offrant à ces modèles une ré–exploitation accrue.

De telles bibliothèques devraient pouvoir être accessibles à tous. Internet constitue un moyen d'accès important pour lequel des normes de composants intégrant nos modèles de calcul doivent pouvoir être définies (*cf. Annexe IV*).

II.C. Vers de nouvelles méthodes de travail

La réutilisation à base de composants métier doit pouvoir modifier les manières de travailler. Elle doit dans un premier temps modifier la façon de travailler du concepteur qui doit de plus en plus réutiliser des briques de bases pour concevoir des dispositifs de plus en plus complexes.

Cette complexité autorisant de moins en moins la conception mono–acteur, des relations d'ingénierie collaborative devront nécessairement se mettre en place telle que nous l'avons décrit dans le *Chapitre IV*.

Dans un deuxième temps, des relations différentes doivent pouvoir s'établir entre clients (*concepteur*) et fournisseurs (*de briques de bases*). En effet, il est envisageable que les fournisseurs puissent proposer à des clients potentiels, les composants permettant d'intégrer les modèles des appareillages qu'ils proposent (*alternateur de voiture, actionneurs pour microsystèmes,* …) au sein du système complet des clients. Cela permettrait, par exemple, de dimensionner le système complet avant de l'assembler, au lieu d'assembler des composants dimensionnés isolément.

ANNEXES

ANNEXE I

ANNEXE I

MODELE ANALYTIQUE DU TRANSFORMATEUR : COMPOSITION DE MODELES EN INTERACTION

I. Besoins d'un modèle composé

Le modèle analytique du transformateur est issu de [POL 86] et modifié pour les besoins du dimensionnement [FAN 99]. Comme nous l'avons décrit dans le *Chapitre IV*, des composants ont été générés, associés à chacune des « disciplines » concernées par le modèle (*économique, électromagnétique, géométrique, calcul de pertes*). Cette composition en modules distincts permet d'obtenir une vision cohérente du modèle d'interaction entre des parties plus ou moins indépendantes (*cf Figure 57, Chapitre IV*). Cette modularité permet en outre d'interchanger un composant par un autre sans nécessairement retoucher le reste du modèle (*cf. Figure 61, Chapitre IV*). Cette séparation des modèles n'est en rien une disjonction puisqu'ils sont toujours en relation grâce au couplage réalisé par leur composition.

II. Processus de génération du composant

Le processus de génération est simple, et consiste à fournir les modèles analytiques (*cf. paragraphe suivant*) au générateur du logiciel Pro@Design, et à indiquer le nom du composant à générer (*nom de fichier*). Le résultat est alors un composant qui peut être utilisé pour vérifier le modèle dans un service de calcul et de post-processing. L'étape suivante consiste à composer les composants générés dans l'outil VisualComposer, puis à indiquer le nom du composant global (*nom de fichier*). Ce dernier composant peut à nouveau être testé dans un service de calcul. Et enfin, le composant global est utilisé dans un service de dimensionnement (*pro@Design ou CDIOptimizer*), dans lequel est définit le cahier des charges et les paramètres d'optimisation.

III. Descriptions des sous modèles

Les modèles sont donnés tels qu'ils sont rentrés dans le générateur de Pro@Design. Un tel modèle peut être composé de commentaires, d'équations et de fonctions. Le générateur détecte automatique les paramètres d'entrée et de sortie, ainsi que les paramètres intermédiaires de calcul qui sont également définis comme paramètres de sortie.

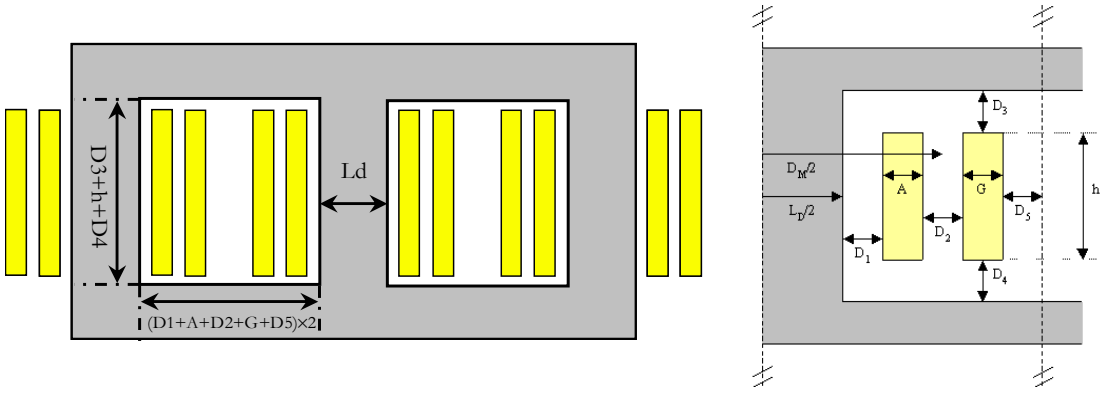


Figure 85 : Géométrie paramétrée du transformateur de tensions triphasées à concevoir

III.A. Modèle géométrique

```
/*
     Modèle de dimensionnement d'un transformateur triphasé 3 colonnes

     Equations issues de:
     M. Poloujadoff, R.D. Findlay," A PROCEDURE FOR ILLUSTRATING THE
     EFFECT OF VARIATION OF PARAMETERS ON OPTIMAL TRANSFORMER DESIGN",
     IEEE Transactions on Power Systems, Vol. PWRS-1, No 4, November
     1986

     Partie du modèle contenant les équations de la géométrie
*/

/* Masse volumique du cuivre */
DC = 8900;
/* Masse volumique du fer */
DI = 7800;

/* Calcul du volume de fer en fonction de la géométrie paramétrée */
Vol_fer0 = AL*FI*(8.0*(D1 + A + D2 + g + D5) + 6.0*ld + 3.0*(h + D4 +
D3));
/* Calcul de la masse de fer */
Masse_fer0 = DI*Vol_fer0;

/* Calcul du volume du cuivre */
Vol_cuivre0 = 3.0*Math.PI*DM*h*(A*F1 + g*F2);

/*
     Calcul de la longueur totale du transformateur en fonction de la
     géométrie paramétrée
*/
ltt = 4*D5 + 3*(ld + 2*D1 + 2*g + 2*D2 + 2*A);
```

III.B. Modèle électromagnétique

```
/*
     Modèle de dimensionnement d'un transformateur triphasé 3 colonnes

     Equations issues de:
     M. Poloujadoff, R.D. Findlay," A PROCEDURE FOR ILLUSTRATING THE
     EFFECT OF VARIATION OF PARAMETERS ON OPTIMAL TRANSFORMER DESIGN",
     IEEE Transactions on Power Systems, Vol. PWRS-1, No 4, November
     1986

     Partie du modèle contenant les équations électromagnétiques
*/

/* Perméabilité du vide */
muzero = 1.257e-006;

/* Résistivité du cuivre */
resistivite_cuivre = 2.6e-008;

/* Coefficient de foisonnement du fer */
FI = 0.8;

/* Coefficient de remplissage du primaire et du secondaire */
F1 = 0.7;
F2 = 0.7;

/* Distance d'isolation entre la culasse et les bobinages */
D1 = 0.05;
D2 = 0.05;
D3 = 0.05;
D4 = 0.05;
D5 = 0.05;

/* Calcul de la puissance par colonne */
S = St/3.0;
/* Calcul de la tension simple par colonne à partir de celle composée
*/
V1 = U1/sqrt(3.0);
/* Calcul de la largeur des bobines primaires et secondaires */
A = (N1*S)/(V1*h*F1*J);
g = (N1*S)/(V1*h*F2*J);
/* Calcul du diamètre moyen des bobines */
DM = ld + 2.0*D1 + 2.0*A + D2;
/* Largeur d'une colonne du transformateur */
ld = sqrt((2.0*sqrt(2.0)*V1)/(pow(Math.PI,2)*f*bt*N1*FI));
/* surface d'une colonne du diamètre*/
AL = (Math.PI/4.0)*pow(ld,2);
/* Calcul de l'inductance de fuite */
FF = (D2 + ((A + g)/3.0))/h;
X2 = muzero*Math.PI*DM*pow(N1,2)*(2.0*Math.PI*f)*FF;
/* Calcul de l'inductance de fuite P.U.*/
X2pu = X2/(pow(V1,2)/S);
```

III.C. Modèle économique

```
/*
     Modèle de dimensionnement d'un transformateur triphasé 3 colonnes

     Equations issues de:
     M. Poloujadoff, R.D. Findlay," A PROCEDURE FOR ILLUSTRATING THE
     EFFECT OF VARIATION OF PARAMETERS ON OPTIMAL TRANSFORMER DESIGN",
     IEEE Transactions on Power Systems, Vol. PWRS-1, No 4, November
     1986

     Partie du modèle contenant les équations de calcul économique
*/
/*
     Introduction de constantes supplémentaires pour pouvoir résoudre
     le problème dans proDesign
*/
/*
     Valeur actualisée du coût de 1 watt de pertes cuivre en charge
     pendant 30 ans avec i=9% et 8760/5 heures et un prix de l'énergie
     de 0.2778E-3 F/(watt*h).
     Donc pspc=0.2778E-3*somme(i=1 à 30, 1/(1+0.09)^i) *3600*8760/5 car
     il y a 8760 heures dans l'année et l'on considère les pertes
     joules moyennes consommées sur une année correspondant a un
     fonctionnement au régime nominal pendant le 1/5ème de l'année
*/
pspc = 5;
/*
     Valeur actualise du coût de 1 watt de pertes fer pendant 30 ans
     avec i=9% et 8760 heures et un prix de l'énergie de 0.2778E-3
     F/(watt*h)      Donc      pspc=0.2778E-3*somme(i=1      à      30,
     1/(1+0.09)^i)*3600*8760 car il y a 8760 heures dans l'année
*/
pspf = 25;

/* Prix au kilo du cuivre */
PC = 25;

/* Prix au kilo du fer */
PI = 12;

/* Calcul du coût du fer */
Prix_fer0 = PI*Masse_fer0;

/* Calcul du coût du cuivre */
Prix_cuivre0 = PC*DC*Vol_cuivre0;

/* Calcul des pertes fers capitalisées */
Valeur_presente_pertes_fer = pspf*Pertes_fer0;

/* Calcul des pertes cuivres capitalisées*/
Valeur_presente_pertes_cuivre = pspc*Pertes_cuivre0;

/* Calcul du prix total du transformateur*/
Prix_total_transfo = Prix_fer0 + Prix_cuivre0 +
Valeur_presente_pertes_fer + Valeur_presente_pertes_cuivre;
```

III.D. Modèle de calcul des pertes

```
/*
     Modèle de dimensionnement d'un transformateur triphasé 3 colonnes

     Equations issues de:
     M. Poloujadoff, R.D. Findlay," A PROCEDURE FOR ILLUSTRATING THE
     EFFECT OF VARIATION OF PARAMETERS ON OPTIMAL TRANSFORMER DESIGN",
     IEEE Transactions on Power Systems, Vol. PWRS-1, No 4, November
     1986

     Partie du modèle contenant les calculs des pertes
*/

/*
     Définition de la loi donnant les pertes en watt/kg en fonction de
     l'induction pour les pertes fer
*/
pfkg(bt) = 1.996 - 8.125*bt + 12.277*bt*bt - 7.502*bt*bt*bt +
1.702*pow(bt,4);

/* Calcul des pertes fer au Kilo: interpolation par moindres carres */
Pertes_fer_Kg = pfkg(bt);

/* Calcul des pertes fer totales*/
Pertes_fer0 = Pertes_fer_Kg*Masse_fer0;

/* Calcul des pertes cuivres totales */
Pertes_cuivre0 = resistivite_cuivre*Vol_cuivre0*pow(J,2);
```

ANNEXE II

ANNEXE II

INTEGRATION D'UN CALCUL REALISE PAR FLUX2D DANS UN COMPOSANT DE DIMENSIONNEMENT

Dans le cadre du dimensionnement d'un transformateur de tensions triphasées (*cf. Chapitre 4*), nous avons mis en œuvre un modèle mixte (*analytique+numérique*), dont la partie numérique doit prendre en compte les difficultés de modélisation de l'inductance de fuite. Cette dernière peut être calculée par un logiciel éléments finis tel que Flux2D®.

Cette annexe détaille la méthodologie d'intégration de ce calcul d'inductance :

- Encapsulation : création d'un composant abstrait à partir du moyen d'accès à l'application Flux2D (*fichier de commandes*) et des informations du projet de conception (*décrites en XML*).
- Projection : création d'un composant COB (*pour le dimensionnement*) à partir du composant abstrait, par la définition des entrées/sorties et du mécanisme de calcul de sensibilité.

I. Encapsulation du calcul d'inductance réalisé par flux2D®

I.A. Un assistant d'encapsulation d'outils pilotés par fichiers : Template Editor

La première étape d'intégration correspond à l'encapsulation du calcul de l'inductance dans un composant abstrait. Flux2D® offre un mécanisme classique de pilotage, correspondant à l'enregistrement de séquences d'actions et la possibilité de « rejouer » ces séquences. Nous avons développé un outil permettant d'encapsuler de tels logiciels, dont le principe de fonctionnement est de faire correspondre à un modèle de fichier de commande, des données se trouvant dans un fichier XML définit par le concepteur.

I.B. Création des informations nécessaires à l'intégration

I.B.1. Création des fichiers modèles

Le fichier généré par flux2D lors de l'enregistrement des séquences d'actions, nommé fichier espion, correspond à ce modèle. Ainsi, il nous suffit d'enregistrer les différentes

phases de modification des paramètres géométriques et physiques ainsi que celles de résolution et d'exploitation des données. La *Figure 86* définit par exemple la séquence permettant de modifier le paramètre H de la géométrie.

```
Q: Nom du parametre
R: ;
R: H
Q: Nouveau commentaire
R:
Q: Nouvelle expression arithmetique
R: 1.773446336E00
```

Figure 86 : Extrait d'un fichier espion flux2D permettant de piloter le module de définition de la géométrie

I.B.2. Création du fichier XML de description du projet de conception

Ensuite, il faut créer le fichier XML correspondant aux données de notre projet de conception (*cf. Figure 87*).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- edited with XML Spy v4.0.1 U (http://www.xmlspy.com) by Laboratoire
D´Electrotechnique de Grenoble (Laboratoire D´Electrotechnique de Grenoble) -->
<ApplicationFlux name="transfo">
   <Configuration>
      <Parameter name="problème" value="TRANSFO"/>
      <Parameter name="système metrique" value="5_metre"/>
      <Parameter name="système de coordonnées" value="1_Cartesien     : X et Y"/>
      <Parameter name="nature de l'espace de travail" value="1_Plan          : section
plane d\'un objet infiniment long"/>
      <Parameter name="nature du problème" value="1_Magnetostatique   : magnetisme,
courant continu"/>
      <Parameter name="region d'inductance" value="PRIMAIRE_DROIT"/>
   </Configuration>
   <Phase name="preflu" batch="geometrie.bat">
      <Inputs>
         <Parameter name="D1" value="5E-02"/>
         <Parameter name="D2" value="5E-02"/>
         <Parameter name="D3" value="5E-02"/>
         <Parameter name="D4" value="5E-02"/>
         <Parameter name="D5" value="5E-02"/>
         <Parameter name="H" value="1.522074301E00"/>
         <Parameter name="LD" value="6.172127838E-04"/>
         <Parameter name="BA" value="2.993745088E-02"/>
         <Parameter name="G" value="2.993745088E-02"/>
      </Inputs>
   </Phase>
   <Phase name="prophy" batch="physique.bat">
      <Inputs>
         <Parameter name="P" value="6.6128E-1"/>
         <Parameter name="DC" value="3.1815E6"/>
         <Parameter name="X1" value="2.10184E-1"/>
         <Parameter name="Y1" value="6.17213E-4"/>
         <Parameter name="X2" value="2.10184E-1"/>
         <Parameter name="Y2" value="8.11654E-1"/>
      </Inputs>
   </Phase>
   <Phase name="expgen" batch="resolution.bat"/>
   <Phase name="resgen" batch="exploitation.bat">
      <Inputs>
         <Parameter name="N" value="1.864588993E02"/>
      </Inputs>
      <Outputs>
         <Parameter name="L" value="0.116894E-01"/>
      </Outputs>
   </Phase>
</ApplicationFlux>
```

Informations nécessaires pour paramétrer la définition de la géométrie dans le module « preflu » de Flux2D

Idem, pour les propriétés physiques du module « prophy »

Aucun paramètre nécessaire pour le module de résolution « expgen »

Le module « expgen » nécessite d'avoir le nombre de spires pour calculer et nous fournir l'inductance de fuite

Figure 87 : Fichier XML détaillant les informations du projet de conception du transformateur à échanger avec le calcul Flux2D de l'inductance de fuite.

I.C. Mise en correspondance des fichiers de commandes et du projet XML

I.C.1. *Association par « glisser-déposer »*

Enfin, en utilisant l'assistant d'encapsulation TemplateEditor, il faut définir la correspondance (*mapping*) entre les éléments du fichier XML et les éléments du fichier de commande au format texte. Pour cela, l'outil propose un moyen intuitif de sélection associé à une technique de glisser-déposer (*drag&drop*), afin de définir l'élément XML en relation avec l'élément de texte.

I.C.2. *Définition d'un « générateur » ou d'un « parser »*

Une fois toutes les associations définies, l'outil propose de créer un composant abstrait qui peut être soit « générateur », soit « parser » (*ou analyseur syntaxique*). Le premier type permet de créer le fichier de commande en fonction des données du fichier XML (*fonction de générateur*). Le deuxième type permet de lire les valeurs dans le fichier de résultat de flux2D et de modifier le fichier XML en conséquence (*fonction de parser*).

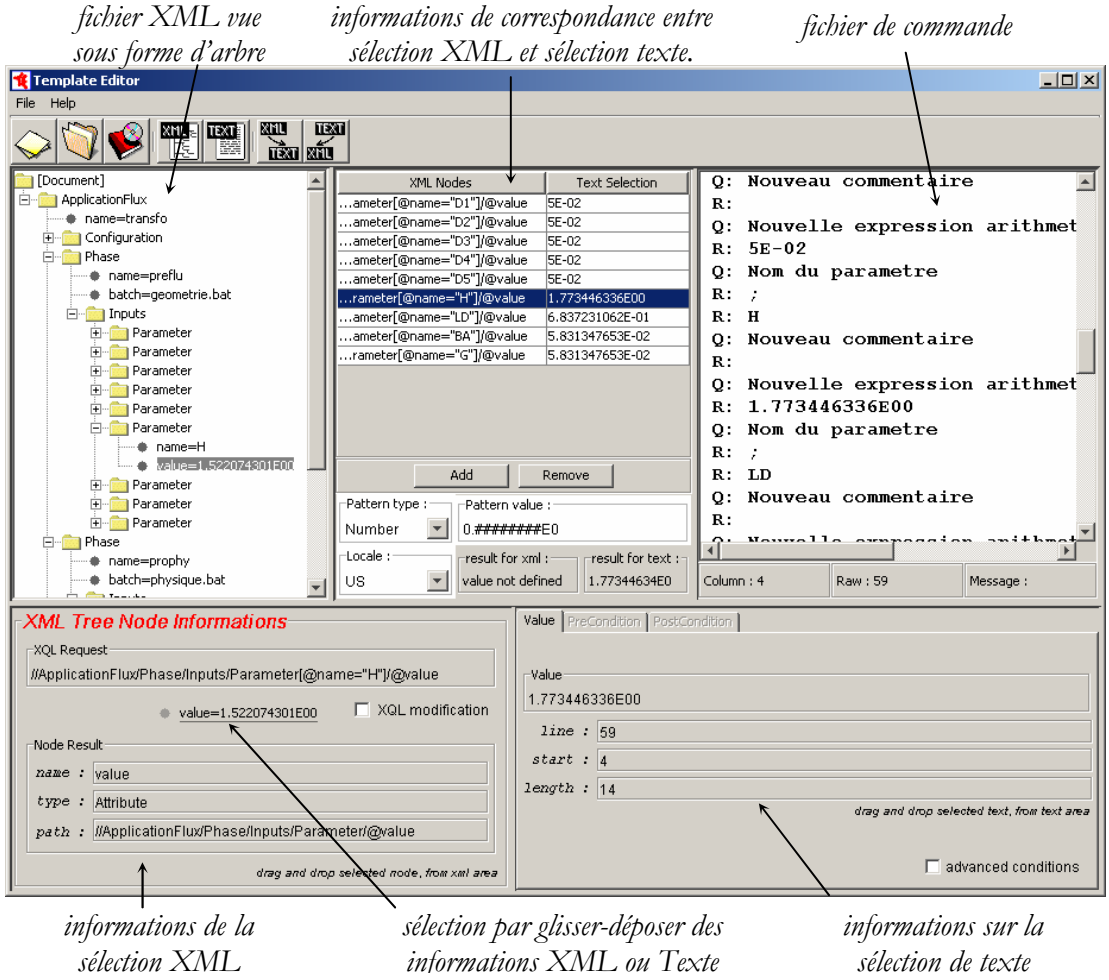


Figure 88 : Template Editor : assistant d'intégration permettant d'encapsuler les outils pilotés par fichier.

I.D. Séquencement des tâches nécessaires au pilotage du logiciel Flux2D

Afin de réaliser le pilotage complet du logiciel que nous souhaitons encapsuler, plusieurs étapes sont nécessaires, correspondantes au pilotage de 4 modules distincts (cf. Figure 89).

Les deux premiers modules (*preflu et prophy*) correspondent à la définition de la géométrie et de la physique du dispositif. Ces deux modules seront paramétrés par les données fournies par le fichier XML, grâce à deux composants abstraits de type générateurs.

Le module suivant est celui de résolution qu'il n'est pas nécessaire de paramétrer et qui sera donc directement exécuté à partir du fichier de commande enregistré.

Enfin, le dernier module correspond à l'exploitation des données, pour lequel il est nécessaire de créer deux composants. Le premier, de type générateur, permet de paramétrer le calcul de l'inductance (*en indiquant le nombre de spires des bobinages*), le deuxième de type parser, permet de lire dans un fichier de résultat la valeur de l'inductance calculée.

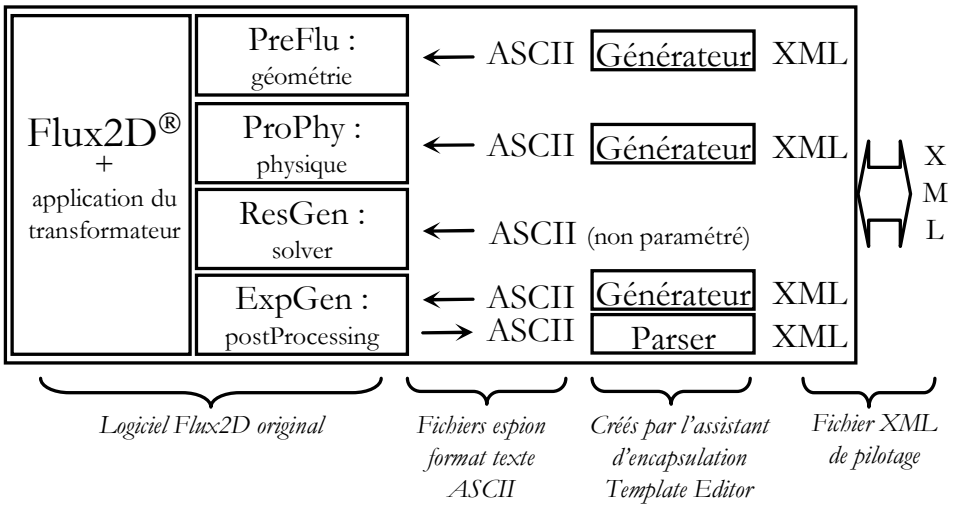
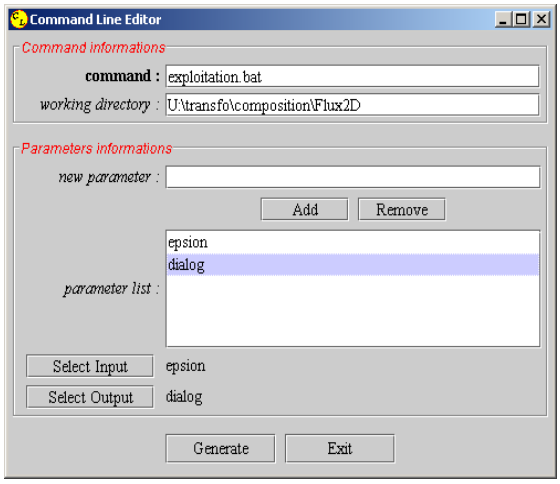


Figure 89 : Modules nécessaires au du logiciel Flux2D

Les composants ainsi générés permettent donc, dans le cas des composants du type générateur, de créer à partir d'informations XML et de modèles de fichiers de commande, les différents fichiers espions nécessaires au pilotage de flux2D. Dans le cas des composants de type parser, ils permettent à partir d'un fichier de résultat de modifier le contenu d'un fichier XML.

I.D.1. *Assistant d'encapsulation de ligne de commande : CMDEditor*



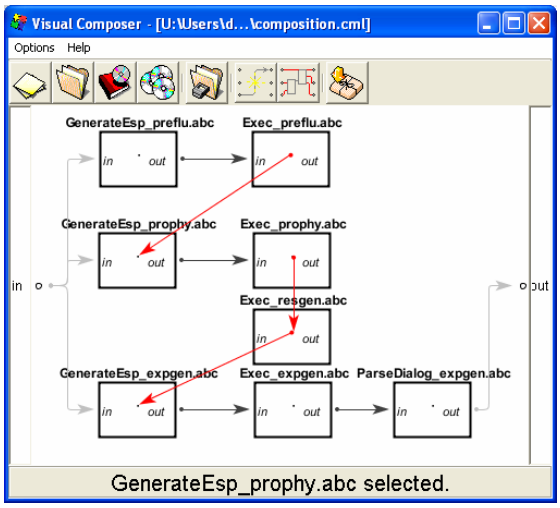
Ces seuls composants ne suffisent pas à exécuter les différents modules de flux2D. Il nous faut en effet quelque chose capable d'exécuter les modules en leur donnant les fichiers espions ainsi générés en paramètres.

Un assistant d'encapsulation (*CMDEditor*) à été créé afin de générer des composants abstraits capables d'exécuter une ligne de commande avec arguments.

Figure 90 : CMDEditor : Assistant d'intégration de ligne de commandes

I.D.2. *Description du séquencement des modules dans l'outil de composition*

Les différents composants résultant de ces assistants d'encapsulation doivent maintenant agir de manière coordonnée pour respecter le processus d'exécution de flux2D.



Pour ce faire, nous utilisons l'outil de composition visuel (*VisualComposer*) afin de définir le workflow correspondant. Ce workflow est composé de diverses tâches successives prenant comme flot d'entrée le fichier XML avec les données de paramétrage de flux, et en flot de sortie le fichier XML contenant les grandeurs calculées par flux, en l'occurrence l'inductance de fuite.

Figure 91 : Description du workflow de pilotage de Flux2D dans l'outil de composition visuel (VisualComposer)

Les tâches à gauche correspondent aux composants générateurs de fichier de commandes, générés par *TemplateEditor*. Les tâches centrales correspondent aux composants exécutant les modules de flux2D, qui viennent d'être générés par *CMDEditor*. Enfin la tâche en bas à droite correspond au composant parser, analysant le fichier de résultat pour inscrire la nouvelle valeur d'inductance dans le fichier XML de sortie, généré par *TemplateEditor*.

Les flots de données sont indiqués par les connexions entre les ports « in » et « out ». Des connections directes entres boites correspondent à un séquencement qui n'est pas explicité par les flots de données. Ces connections sont par exemple nécessaires si les modules partagent implicitement une information (*comme c'est le cas avec les modules de Flux2D*), imposant un séquencement bien précis entre les boites afin qu'il n'y ai pas d'incohérences entre leur exécutions respectives.

I.E. Le composant abstrait résultant de l'encapsulation de Flux2D

Cette composition génère alors un composant abstrait global (*Flux2DProcess.abc pour l'exemple qui suit*), permettant de paramétrer le calcul flux2D de l'inductance de fuite de notre transformateur, à partir du fichier XML décrit plus haut. Les composants abstraits proposent un mécanisme simple permettant de les exécuter, il est ainsi possible de lancer le calcul Flux2D dès à présent grâce à la ligne de commande suivante :

```
java –jar Flux2DProcess.abc /I transfo.xml /O transfo.xml
```

Après avoir successivement vue apparaître à l'écran les différents modules de flux2D, le programme s'arrête, et la nouvelle valeur calculée se trouve dans le fichier *transfo.xml*, à l'emplacement indiqué lors de l'encapsulation.

II. Projection du composant abstrait vers un composant de dimensionnement

II.A. Un outil de projection : ABC vers COB

Ce que nous souhaitons obtenir est un composant de dimensionnement qui puisse se composer avec deux autres composants analytiques représentant le reste du modèle du transformateur (*cf. Figure 63*). Afin de bénéficier des services associés à cette norme de composant (*composition, optimisation,* …), nous devons opérer la projection du composant abstrait vers un composant COB.

Un tel projeteur (*ABC2COB Projector*) a été développé permettant de générer un composant COB à partir d'un composant abstrait (*cf. Figure 92*). Ainsi, le composant généré précédemment, permettant de piloter Flux2D, peut être transformé en composant de type COB pour ensuite pouvoir être composé aux modèles analytiques.

II.B. Définition des paramètres d'E/S

La première étape de cette projection est de définir, à partir du fichier XML de commande du composant abstrait, quels sont les paramètres d'entrée et de sortie du composant à générer. Pour répondre à ce besoin le principe de glisser-déposer a été réutilisé permettant au concepteur de spécifier la sémantique des données abstraites du fichier XML. Ainsi le composant abstrait, n'ayant qu'un fichier en entrée et un en sortie, devient un composant COB ayant une liste de paramètres en entrée et une liste de paramètre en sortie.

II.C. Définition du calcul de sensibilité

Lorsque le logiciel encapsulé ne permet pas de définir le calcul de sensibilité des sorties par rapport aux entrées, il est nécessaire de créer ce calcul. Ce calcul est apporté par le projeteur qualifié pour l'occasion de « projeteur à sémantique augmentée ». Pour apporter cette fonctionnalité, le projeteur offre la possibilité d'utiliser différentes méthodes, en fonction des problèmes numériques éventuels et de la nécessité d'être rapide ou non. Nous avons implémenté pour cet exemple une méthode de calcul par différences finies basée sur la mémorisation du Jacobien au cours des différents appels (*cf Annexe III*).

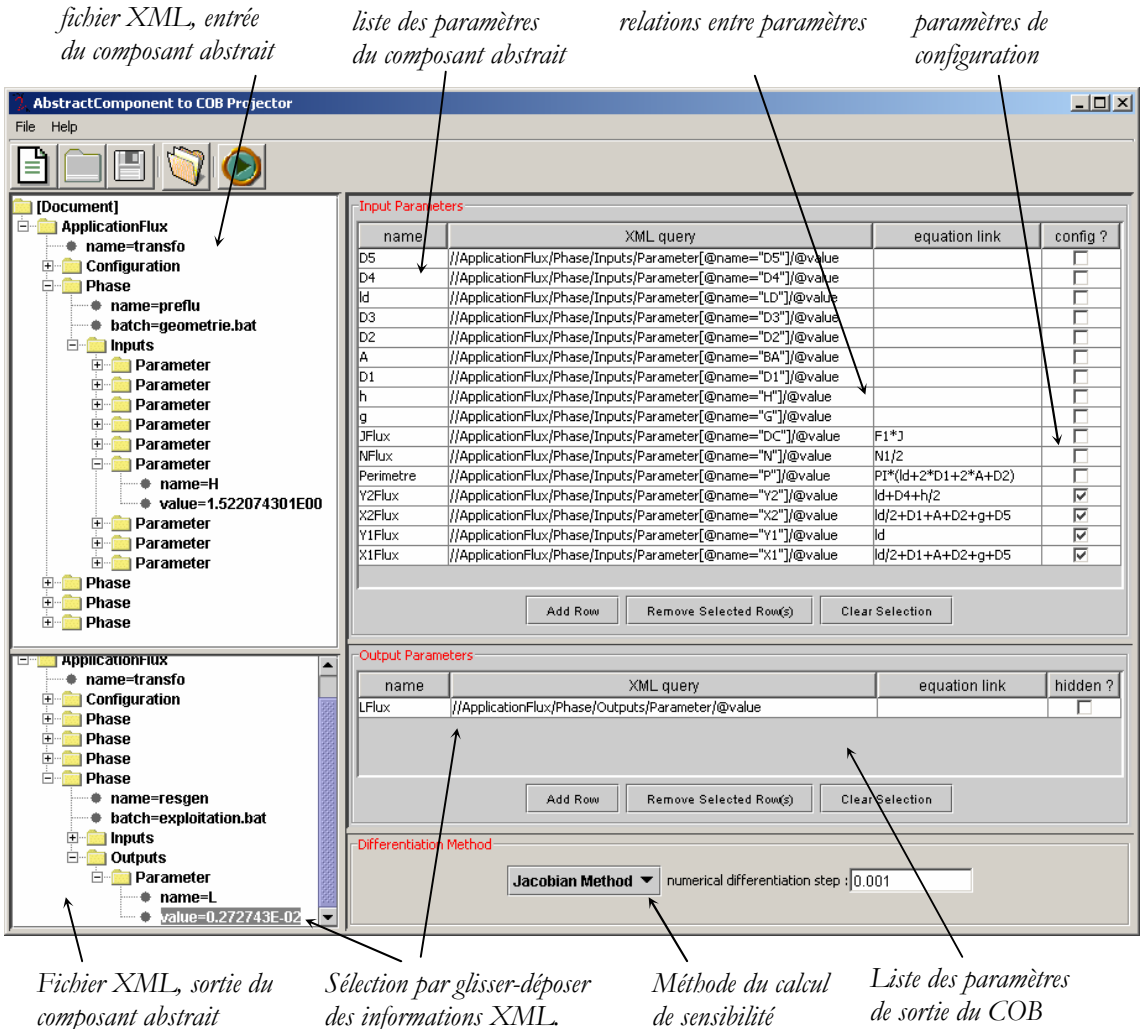


Figure 92 : ABC2COB Projector : outil de projection générant un composant de dimensionnement (COB) à partir d'un composant abstrait.

II.D. Autres fonctionnalités

L'outil gère la possibilité de définir des paramètres de configuration, sémantique qui n'existe pas dans les spécifications des COBs. Une demande de calcul de sensibilité par rapport à ces paramètres ne sera alors pas prise en compte et retournera un résultat nul. Les paramètres de configuration semblent être une nécessité pour de futures spécifications de nos composants de calcul pour le dimensionnement. Ces paramètres, pouvant par exemple configurer un algorithme à l'intérieur du composant, ne nécessitent pas de faire parti du calcul de sensibilité et peuvent surtout induire des erreurs dans le modèle de calcul. Dans l'exemple du transformateur, les conditions limites du modèle éléments finis sont définies par les paramètres de configuration *X1Flux, Y1Flux, X2Flux, Y2Flux* (*cf. Figure 92*), eux même définis par une relation utilisant les paramètres géométriques.

Des relations peuvent en effet être définies entre les entrées du futur composant et les entrées du composant abstrait, et de même pour les sorties. Cela permet de s'adapter à un contexte de connexion en créant la « glue » appropriée. Ainsi, dans l'exemple du transformateur, le périmètre est calculé à partir de la géométrie (*cf. Figure 92*), pour définir la profondeur du problème 2D dans le logiciel Flux2D. Ces équations sont automatiquement traitées pour générer le code de calcul associé, ainsi que le code de calcul de sensibilité par dérivation symbolique. Cette fonctionnalité n'est pas nécessaire dans ce projeteur mais permet de simplifier le modèle global puisqu'il n'est alors pas nécessaire de créer un composant supplémentaire de glue et de réaliser sa composition avec le composant projeté.

Le composant généré par ce projeteur possède alors les entrées déclarées par les paramètres définis dans le composant abstrait, ou si ces derniers sont définis par des relations, les paramètres qui ne sont pas connus sont alors définis comme des entrées du composant à générer. Dans l'exemple du transformateur, le paramètre *NFlux* du composant abstrait est définit comme étant égal à $N_1/2$ (*cf. Figure 92*) car le calcul éléments finis exploite une symétrie horizontale de la géométrie du transformateur. Le paramètre N_1 n'étant pas défini par ailleurs, le projeteur le définira comme paramètre d'entrée du composant COB.

III. Conclusions

III.A. Réintégration du composant dans le modèle global

Ce processus d'intégration terminé, nous disposons d'un composant COB calculant l'inductance de fuite par une méthode numérique réalisée par le logiciel Flux2D. Ce composant peut alors être composé avec les deux modèles analytiques pour créer le composant de dimensionnement électromagnétique, qui à son tour va être réintégré dans la composition globale (*électromagnétique, géométrique, pertes, économique*) (*cf Figure 61, Chapitre IV*).

III.B. Dynamique d'intégration

Ce processus d'intégration peut être « rejoué », notamment pour être redéfinit, offrant ainsi une solution à la dynamique du processus de conception. Les différents cahiers des charges qui peuvent être nécessaires pour parvenir à un dimensionnement peuvent en effet faire apparaître des paramètres qui n'ont pas été intégrés dans une première intégration.

En imaginant un premier cahier des charges ne définissant pas le nombre de spires de bobinages comme un paramètre optimisable (*pas d'algorithme prenant en compte des paramètres discrets,* …), notre intégration n'aurait pas eu la même structure. En effet, le module *Expgen* (*exploitation des données*) n'ayant comme paramètre variable que ce nombre de spires, il n'aurait pas été nécessaire de créer le composant abstrait de transformation XML en fichier de commande. Si par la suite, ce paramètre devait devenir optimisable, la ré-intégration aurait été rapide (*quelques minutes*). En effet, il aurait simplement fallu modifier le projet décrit en XML, ne modifiant en rien les intégrations déjà réalisées à partir de ce fichier. Il aurait ensuite fallu créer le composant de transformation du projet XML en fichier de commande pour le module d'exploitation des résultats, grâce à l'outil *TemplateEditor*. Enfin, ce composant aurait du être intégré dans la description du séquencement des modules Flux2D grâce à l'outil *VisualComposer*.

Diverses modifications peuvent être imaginées remettant en cause l'intégration à différents niveaux, mais la méthodologie d'intégration a été conçue avec l'objectif de s'adapter, devant permettre de gérer l'imprévisible rapidement et sans programmation.

ANNEXE III

ANNEXE III

DETAILS SUR LE DIMENSIONNEMENT DU TRANSFORMATEUR

I. Description des 5 cahiers des charges

Le dimensionnement du transformateur décrit dans le *Chapitre IV* et en *Annexe I*, a été réalisé en suivant un processus de dimensionnement mettant en œuvre successivement plusieurs cahiers des charges (*CDC*) de plus en plus contraints.

Tableau 5 : description des 5 cahiers des charges de dimensionnement appliqués aux modèles du transformateur

		Spec1		Spec2		Spec3		Spec4		Spec5	
		min	max	min	max	min	max	min	max	min	max
Paramètres d'entrée optimisables	H	**1**	**10**	1	10	**1**	**1.4**	1	1.4	1	1.4
	J	**5e5**	**4.5e6**	5e5	4.5e6	5e5	4.5e6	5e5	4.5e6	5e5	4.5e6
	N_1	**100**	**1000**	100	1000	100	1000	100	1000	100	1000
	Bt	**0.5**	**1.9**	0.5	1.9	0.5	1.9	0.5	1.9	0.5	1.9
Paramètres de sortie contraints	cout	**Fonction objectif à minimiser**									
	X_1	**Fixée: 7.2**		**5.76**	**8.64**	5.76	8.64	5.76	8.64	5.76	8.64
	A							**0**	**0.04**	0	0.04
	Ltt									**0**	**3**

Tableau 6 : description des paramètres des modèles du transformateur

Paramètres en entrée		**Paramètres en sortie**	
H :	*Hauteur des enroulements (m)*	cout :	*Coût du transformateur, intégrant les pertes capitalisées pour une période d'utilisation de 30 ans (en Francs)*
J :	*Densité de courant (A/m²)*	X_1 :	*Réactance de fuite du primaire (Ohm)*
N_1 :	*Nombre de tours du primaire*	A :	*Largeur de la jambe (m)*
Bt :	*Densité du flux magnétique (T)*	Ltt :	*Largeur totale du circuit magnétique (m)*

I.A. Sur l'orientation du modèle de dimensionnement

Le CDC1 impose une réactance de fuite fixée à 7.2Ω. Cette valeur fixe a des origines venant du processus même de dimensionnement utilisé par le modèle analytique original. En fait, le modèle original a été orienté de manière à ce que les paramètres à dimensionner soient placés en sortie. Typiquement, la hauteur du circuit magnétique était calculée à partir de paramètres d'entrée tels que la réactance de fuite. Ici, le processus de dimensionnement

utilisé [WUR 96a] permet de résoudre un problème inverse sans devoir réorienter le modèle, pouvant imposer des contraintes à la fois sur les paramètres d'entrée et les paramètres de sortie.

Nous voyons donc qu'à partir du CDC2, la réactance de fuite peut varier dans un intervalle, offrant de plus grandes possibilités d'améliorer la solution. Dans les CDCs suivants, nous imposons successivement des contraintes supplémentaires sur les paramètres du modèle, sans que, en tant que concepteur, aucune difficulté apparaisse.

I.B. Corrélations entre la complexité des cahiers des charges et les temps d'optimisation

Aux premiers regards, ces spécifications nous laissent penser que le dimensionnement est de plus en plus complexe, car on augmente le nombre de contraintes. En réalité, les durées d'optimisation sont moins grandes pour les CDC ayant des sorties contraintes, car elles restreignent la taille de l'espace des solutions qui est définie par les paramètres d'entrée optimisables.

I.C. Détails des cahiers des charges

Les différents CDCs imposent un intervalle de variations sur 4 paramètres d'entrée du modèle (*H, J, N1 et Bl*), les autres paramètres d'entrée étant fixés (*puissance apparente totale : St=40 MVA, tension au primaire : U1=60 kV, fréquence : f=50Hz*).

Le premier CDC impose une valeur de réactance de fuite, contraignant fortement la convergence en guidant l'algorithme. Le second CDC, relaxe cette contrainte, offrant la possibilité à l'algorithme d'explorer un plus grand espace. Le troisième CDC, réduit l'espace des solutions en limitant la hauteur des enroulements à 1,4m au lieu de 10m. Le quatrième CDC impose une nouvelle contrainte en sortie du modèle sur la largeur de la jambe (*A*). Le dernier CDC fait de même pour la largeur du transformateur (*Ltt*).

II. Résultats d'optimisation

II.A. Utilisation du composant dans le service d'optimisation

La disponibilité de nos deux modèles analytique et mixte, va nous permettre de comparer les deux approches, ainsi que l'apport de la métaphore du microscope, sur les 5 cahiers des charges définis précédemment.

Pour utiliser les modèles dans un service d'optimisation, il faut tout d'abord ouvrir dans ce dernier le composant associé au modèle. Le service d'optimisation affiche alors les entrées et les sorties du modèle, pour lesquels il est possible d'associer les spécifications d'un des cahiers des charges. Ainsi, une fonction objectif sera définie sur un paramètre de sortie, les entrées pourront être fixées ou optimisables dans un certain intervalle. Quant aux sorties, elles pourront être libres, contraintes par intervalle ou par valeur fixe.

Les résultats d'optimisation des 5 cahiers des charges appliqués au modèle analytique, au modèle mixte et enfin dans le cadre de la métaphore du microscope, donnent des solutions quasiment identiques mais n'ont pas le même processus d'obtention (*cf. Tableau 7*).

Tableau 7 : Nombre d'itérations et temps pour dimensionner les différents modèles du transformateur pour les 5 CDCs. Les calculs ont été réalisé sur un PC Pentium IV 1GHz, 512 Mo RAM.

		Modèle analytique	Modèle Mixte	Métaphore du microscope
CDC1	Nb d'itérations	11	18	5
	temps (sec)	0,21	8183(2h16)	2200(37min)
CDC2	Nb d'itérations	16	12	5
	temps (sec)	0,28	5300(1h28)	2200(37min)
CDC3	Nb d'itérations	11	11	3
	temps (sec)	0,21	5000(1h23)	1300(22min)
CDC4	Nb d'itérations	7	8	2
	temps (sec)	0,16	3100(51min)	840(14min)
CDC5	Nb d'itérations	7	7	5
	temps (sec)	0,16	3100(51min)	2150(36min)

II.B. Analyse en terme d'itération selon les différents CDC

Comme nous pouvons le voir (*Tableau 7*), le nombre d'itérations pour les 3 approches diminue, sauf exceptions, au cours des différents cahiers des charges. Comme nous l'avons dit dans la description des cahiers des charges, le CDC2 est moins guidé que le CDC1 pour trouver une solution, ce qui explique le plus grand nombre d'itérations.

Concernant le CDC1, le nombre d'itérations plus grand pour le modèle mixte s'explique par une précision d'égalité trop forte sur la contrainte de X1 devant converger à 7.2000 ± 10^{-4}, par rapport à la qualité du calcul de sensibilité du modèle numérique (*différences finies de pas 10^{-2}*). Pour ces spécifications de dimensionnement, nous pouvons voir sur les courbes suivantes (*cf. Figure 93 et Figure 94*) que le comportement des deux modèles est strictement identique, ne différant que sur la condition d'arrêt de l'optimisation.

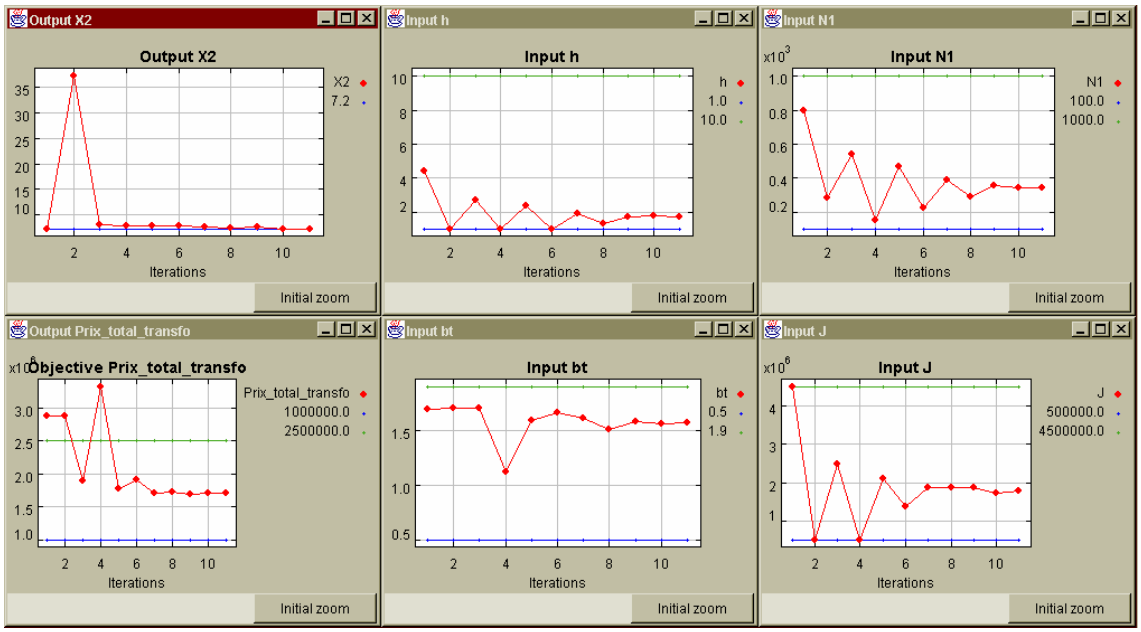


Figure 93 : CDC1, modèle analytique, précision de fin 1e-4, 210ms, 11 itérations

Difficulté de convergence sur une contrainte d'égalité à cause de la précision du calcul de sensibilité.

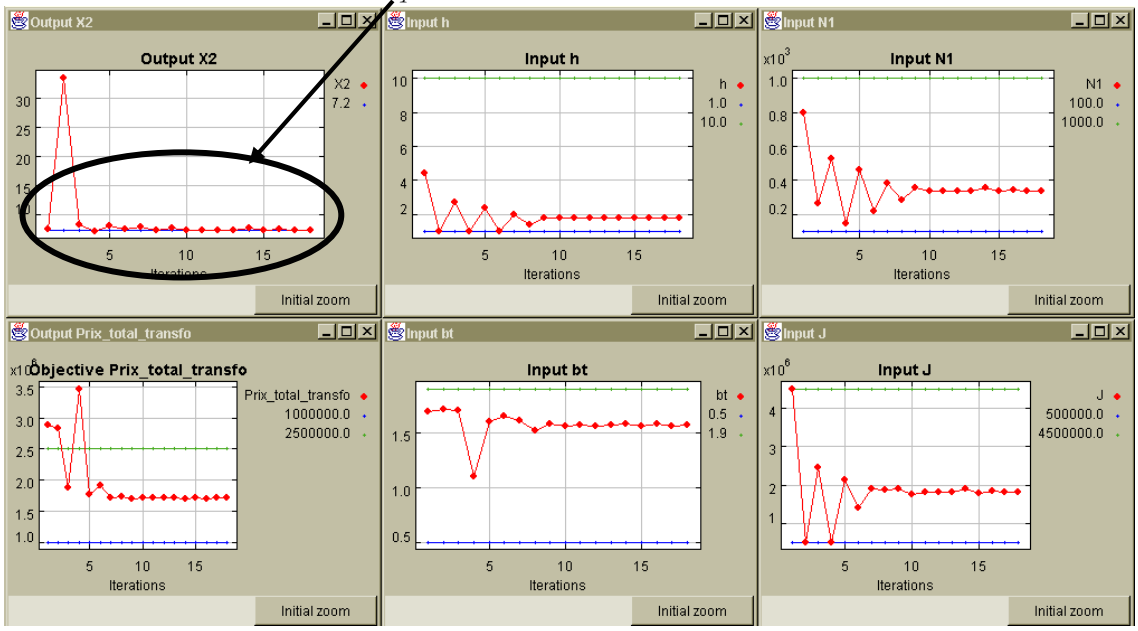


Figure 94 : CDC1, modèle mixte, précision de fin 1e-4, précision des différences finies 1e-2, 2h16, 18 itérations

II.C. Influence de la méthode de calcul de sensibilité sur les temps de dimensionnement

Différentes méthodes de calcul de sensibilité peuvent être codées durant la projection à sémantique augmentée (*ABC2COB*). Nous avons implanté différentes manières de réaliser le calcul des différences finies afin d'optimiser les temps de dimensionnement selon le contexte d'utilisation [DEL 03]. Nous nous sommes principalement attaché à optimiser ce

code pour un calcul réalisé au sein d'une composition, et par rapport aux contraintes qu'impose la norme des composants de calcul utilisée (*COB*).

3 Méthodes ont donc été testées :

La première réalise un calcul de toutes les dérivées partielles nécessaires au calcul de la différentielle de sortie. Ce calcul est réalisé pour chaque appel de calcul d'une différentielle de sortie :

$$dS = \frac{\partial S}{\partial I_1} dI_1 + \cdots + \frac{\partial S}{\partial I_n} dI_n \qquad \frac{\partial S}{\partial I_i} = \frac{S(I_1, \cdots, I_i + \Delta_i, \cdots, I_{n+p}) - S(I_1, \cdots, I_i, \cdots, I_{n+p})}{\Delta_i}$$

La deuxième réalise le même calcul mais le Jacobien est construit au fur et à mesure des appels, et les dérivées partielles déjà calculées pour un même point de fonctionnement sont réutilisées. Les appels redondants à une même dérivée partielle viennent du fait que la propagation du calcul de sensibilité est faite par différentielle, et le calcul de sensibilité à l'intérieur du composant est fait par dérivée partielle.

Enfin, une dernière méthode a été testé, celle calculant la différentielle de sortie (*dérivée directionnelle*) par une différence finie sur l'ensemble des directions de dérivation. Cette méthode ne fonctionne bien que pour une normalisation (*p*) adaptée à la nature des paramètres différentiables à réaliser au cas par cas.

$$dS = \left(S(I_1 + \frac{dI_1}{p}, \ldots, I_n + \frac{dI_n}{p}) - S(I_1, \ldots, I_n) \right) \cdot p$$

Exemples de normalisations : $p = \sqrt{dI_1^{\,2} + \ldots + dI_n^{\,2}}$ ou $p = \sum \left| \frac{dI_i}{I_i} \right|$ ou …

Les résultats suivants (*cf. Tableau 8*) montrent qu'il est important de bien prendre en considération le contexte d'utilisation du code de calcul. Les résultats les plus intéressants dans notre cas d'étude (*composition par propagation de différentielle*) correspondent à la solution par calcul de différentiel, mais cette méthode demande un unique réglage (*la normalisation p*) qui doit être adapté à tous les paramètres. Uns méthode plus robuste et satisfaisante, consiste à travailler avec les dérivées partielles pour lesquelles le calcul des différences finies est adaptable pour chaque paramètre.

Tableau 8 : Temps de dimensionnement des 5 cahiers des charges à partir de modèle mixte, dans lequel le calcul de sensibilité du modèle numérique est optimisé en fonction du contexte d'utilisation

		dérivées partielles	**mémorisation du Jacobien**	**différentielles**
CDC 1	*itérations*	18	18	12
	temps (s)	8183	3835	1875
	temps/iter (s)	454	213	156
CDC 2	*itérations*	12	12	12
	temps (s)	5300	2558	1822
	temps/iter (s)	441	213	151
CDC 3	*itérations*	11	11	10
	temps (s)	5000	2305	1423
	temps/iter (s)	454	209	142
CDC 4	*itérations*	8	13	7
	temps (s)	3100	2716	960
	temps/iter (s)	387	208	137
CDC 5	*itérations*	7	7	7
	temps (s)	3100	1453	1000
	temps/iter (s)	442	207	142

ANNEXE IV

ANNEXE IV

DETAILS SUR LE PILOTAGE INFORMATIQUE DE LOGICIELS

Cette annexe, relevant plus de techniques informatiques que les développements précédents, vise à synthétiser une part du travail qui a été nécessaire dans cette thèse. Partant du constat de contraintes informatiques nécessairement présentes dans un environnement d'intégration comme celui que nous avons proposé, il est nécessaire d'appréhender certains concepts informatiques tels que ceux que nous allons développer sur le pilotage de logiciel.

Cette annexe semble importante quant à la suite de notre environnement de conception. En effet, l'intégration des outils du concepteur nécessite une connaissance des mécanismes de pilotages de logiciels. Ainsi, si nous souhaitons que de nouveaux assistants d'intégration puissent être développés, les concepts de pilotage synchrone ou asynchrone que nous allons détailler sont nécessaires. De même, les techniques permettant de faire intéropérer des codes informatiques hétérogènes peuvent être nécessaires.

I. Un « point d'entrée » des outils

On peut discerner deux points d'entrée différents :

- par l'extérieur du logiciel, on parlera alors de pilotage synchrone,
- par l'intérieur du logiciel, on parlera alors de pilotage asynchrone.

La différence principale vient du sens de la requête initiale (*cf. Figure 95*). Dans le cas du pilotage synchrone, l'initiation de la communication se fait de l'extérieur du logiciel, généralement par un autre outil. Dans le cas du pilotage asynchrone, l'initiation de la communication se fait de l'intérieur du logiciel, généralement par l'utilisateur du logiciel, grâce à un code directement implanté au sein du logiciel appelé « plug-in ».

Ces deux notions sont essentielles à la réalisation d'un environnement de conception multi-outils. Nous allons voir que la première catégorie de pilotage (*synchrone*), est à la base des

assistants d'intégration proposés au *Chapitre III*. Nous verrons ensuite que le pilotage asynchrone est un moyen pour réaliser d'autres connections entre logiciels, dont celle utilisée pour connecter les résultats d'optimisation à une géométrie paramétrée dans l'outil MathCAD® (*Chapitre IV, Partie II*).

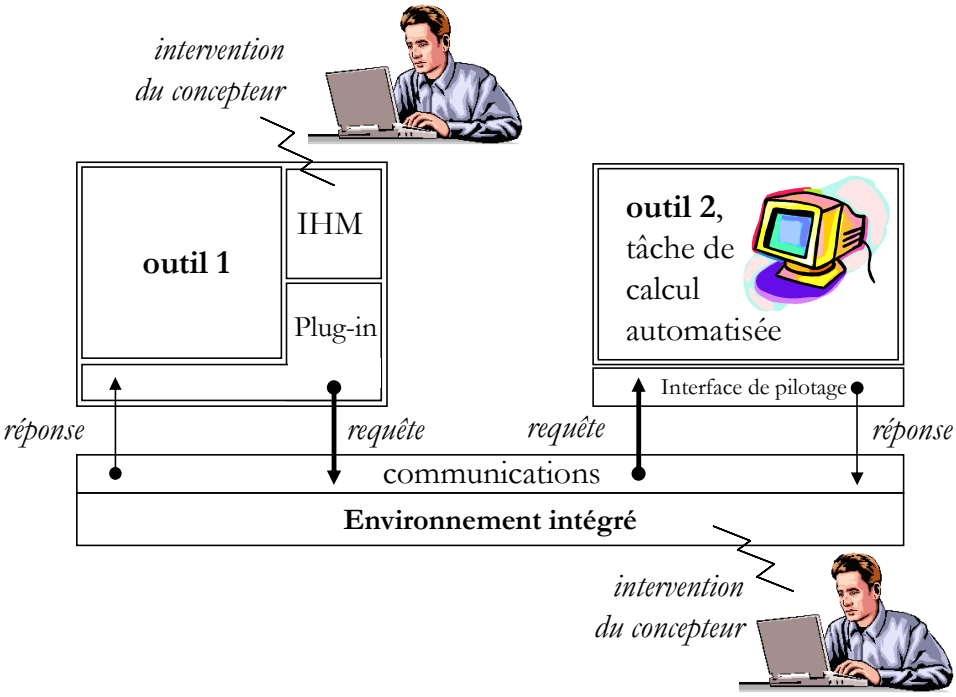


Figure 95 : Utilisation d'outils dont le pilotage est asynchrone (intervention de l'utilisateur dans l'outil) ou synchrone (automatisation possible).

I.A. Interaction inter-logiciels automatisée, ou pilotage synchrone

I.A.1. Le pilotage synchrone pour l'automatisation

La principale application du pilotage synchrone correspond à l'automatisation de tâches. En effet, ce pilotage peut être réalisé sans intervention d'utilisateur et donc être totalement contrôlé par un programme de supervision.

L'illustration dans le cadre du dimensionnement, correspond au pilotage par un algorithme d'optimisation, d'un logiciel simulant le dispositif paramétré à dimensionner. Le simulateur doit par exemple, permettre à l'algorithme :

- de venir lui modifier les valeurs des paramètres du dispositif,
- de lui demander de simuler le dispositif,
- et de récupérer les valeurs des grandeurs simulées.

Si le logiciel offre ces possibilités de pilotage, alors nous sommes en mesure d'automatiser des appels successifs au simulateur.

I.A.2. Les moyens pour entrer dans l'environnement de conception

Le problème technique qui se pose est le moyen d'accéder au pilotage du simulateur ou du logiciel quel qu'il soit. Nous pouvons recenser quelques-uns des ces points d'entrée :

Moyens pour accéder aux services :

- ligne de commande (*exécution paramétrée*),
- API (*interface de programmation du logiciel*), code source,
- script de commande (*langage de programmation simplifié, dédié au logiciel*).

Moyens pour accéder aux données (en lecture ou en écriture) :

- fichiers de données (*généralement en format propriétaire*),
- base de données.

Moyens pour réaliser les deux en même temps :

- scripts ou messages, (*liste de commandes paramétrées*),
- API, code source.

Nous avons pu voir en détail dans l'*Annexe II* comment, grâce aux moyens de pilotage par fichiers de commandes, nous avons pu réaliser le pilotage synchrone d'un logiciel de simulation (*Flux2D*). L'assistant d'encapsulation TemplateEditor est basé sur ce moyen de pilotage et permet d'encapsuler tout logiciels pilotés par fichiers.

I.B. Interaction inter-logiciels contextualisée, ou pilotage asynchrone

I.B.1. Besoins d'interaction avec l'utilisateur

L'autre moyen d'accès (*pilotage asynchrone*) qui a été détaillé dans [FIS 02], s'appuie sur un point d'entrée interne à l'outil dans lequel est placé un code informatique « plug-in ». La principale application du pilotage asynchrone consiste à faire intervenir le concepteur dans un des outils intégrés pour en utiliser les fonctionnalités, puis échanger ses résultats avec l'environnement. Le concepteur peut ainsi bénéficier de l'interface utilisateur du logiciel (IHM), généralement la mieux adaptée pour interagir avec cet outil.

Considérons un exemple que nous avons mis en œuvre pour un logiciel de CAO mécanique définissant la structure du dispositif étudié. Ce logiciel peut être interfacé avec l'environnement de conception pour définir les grandeurs géométriques partagées avec les modèles de simulation ou de dimensionnement. Ainsi, le concepteur peut utiliser sa

géométrie 3D en sélectionnant une côte afin de la connecter à un paramètre du modèle accessible par l'intermédiaire de l'environnement de conception intégré [RIB 02].

Le problème technique qui se pose correspond encore aux moyens qui sont offerts par les logiciels afin d'initier une communication. Les outils ayant comme possibilité d'étendre leur fonctionnalités, offrent généralement des moyens d'accès internes. Ce point d'entrée qui nous est offert pourra alors être utilisé pour réaliser nos objectifs de communication avec l'environnement qui se trouve alors à l'extérieur.

I.B.2. Moyens pour accéder aux services et aux données

Quelques moyens d'accès peuvent être recensés :

- une fonction utilisateur en script propriétaire ou standardisé de type JavaScript ou VisualBasicScript (*interaction essentiellement avec les objets internes du logiciel, limitée avec l'extérieur*),
- une fonction utilisateur dans un langage de programmation de type C, C++, Java, fortran (*permet de se connecter à des codes externes*)
- un code source (*ajout d'un module, interaction avec l'IHM, les objets internes, et l'extérieur*),

Considérons deux exemples. Le premier concerne l'outil MathCad® dans lequel il est possible de définir une fonction utilisateur en C/C++, laquelle permet d'intégrer uniquement un résultat numérique (*ex : $z=f(x,y)$*). C'est notamment ainsi que nous avons pu réaliser des vidéos d'optimisation dont nous avons montré quelques images dans le *Chapitre IV* lors du dimensionnement de l'actionneur magnétique ou du déclencheur électromagnétique.

Considérons aussi l'environnement MatLab® dans lequel il est possible d'exécuter des programmes C/C++ ou Java™ permettant par exemple d'importer ou d'exporter des données numériques [FIS 02].

Dans ces deux cas de pilotage, il n'est pas possible d'initier de communications de l'extérieur, limitant l'utilisation de ces logiciels à un pilotage asynchrone (*intervention de l'utilisateur dans l'outil*).

I.C. Différences entre ces deux modes de pilotage

I.C.1. Communication bi-directionnelle avec acteur humain d'un côté

La différence majeure, comme nous l'avons définit au début de cette partie, vient du fait que la communication est initiée de l'extérieur ou de l'intérieur du logiciel. Une fois cette communication établie, les données peuvent circuler dans les deux sens en tant que paramètres d'une requête ou résultats de cette requête. Dans le cas du pilotage interne, le concepteur a à sa disposition l'interface graphique utilisateur (*IHM*) du logiciel pour gérer au mieux les données à traiter. Dans le cas du pilotage externe, nous avons vue que la particularité correspondait à la possibilité d'exécuter les services du logiciel sans avoir à y intervenir.

	intérieur	extérieur
données en entrée et sortie :	☑	☑
IHM pour gérer les données :	☑	☐
pilotage « boite noire » :	☐	☑

Figure 96 : Comparaison des modes de pilotage synchrone (extérieur) et asynchrone (intérieur)

Du point de vue de leur utilisation, on peut alors caractériser ces deux points d'entrée de la manière suivante :

- le pilotage extérieur permet de répondre à un besoin d'automatisation,
- le pilotage intérieur permet de répondre à un besoin d'intervention de l'utilisateur.

I.C.2. Mise en œuvre d'un processus de conception intégrant les deux modes de communication

Ainsi nous pouvons considérer une utilisation conjointe des deux modes lors d'un processus comprenant des étapes automatisées et des étapes dans lesquelles le concepteur fait des choix et apporte des informations propres au contexte de conception.

Voici un exemple de ce que peut être un tel processus :

- Une base de modèles est consultée, et une liste de solutions de conception est choisie et transférée dans un outil de simulation (*transfert des modèles de simulation de solutions choisies*).
- Le concepteur étudie les dispositifs simulés grâce au post-processing du logiciel, et retient une structure parmi ces différentes solutions (*transfert d'un modèle de dimensionnement de la solution retenue*).
- Un outil d'optimisation est alors lancée dans lequel le concepteur doit définir un cahier des charges correspondant à ses critères de dimensionnement (*transfert des paramètres optimisés et exécution de la simulation*)
- Le concepteur analyse la validité de la solution dimensionnée et reboucle si nécessaire.

I.C.3. Conclusions

Ces deux modes de pilotage sont nécessaires, d'une part parce que les outils n'offrent que rarement le mode désiré et qu'une alternative est alors intéressante, et d'autre part parce qu'un processus de conception peut avoir besoins d'interactions de ces deux genres comme le montre l'exemple précédent.

Comme nous l'avons montré, le mode de pilotage asynchrone permet notamment au concepteur de bénéficier de l'IHM de son outil tout en pouvant dialoguer avec l'environnement. Malheureusement, nous ne proposons pas de solution pour mettre en œuvre simplement de telles connections entre les outils du concepteur. La raison vient d'une forte dépendance du code du « plug-in » à l'outil pour lequel il est destiné, ainsi qu'une dépendance par rapport à ses objectifs d'utilisation. Pour ces raisons, nous ne pouvons proposer des assistants d'intégration basés sur ces points d'entrés.

Par contre, le pilotage synchrone fait partie des solutions que nous proposons pour permettre au concepteur d'intégrer simplement et rapidement ses outils dans l'environnement de conception. Afin d'aller plus loin dans ce sens, et que les bases nécessaires à la mise en œuvre d'assistants d'intégration soient définies dans ce rapport, nous allons maintenant détailler les techniques informatiques d'interopérabilité.

II. Les techniques d'interopérabilité

II.A. Problèmes d'interopérabilité

Des problèmes technologiques peuvent venir perturber les possibilités de pilotage de code. En effet, les codes de calculs peuvent être développés dans divers langages (*fortran, C, C++, Java,* …) et disponibles sous diverses formes (*exécutable Unix, librairie windows, code source, composant java,* …).

D'un point de vue utilisation, ces codes n'offrent encore que très rarement une façon simple d'interagir avec eux, autre que par leur interface graphique. Ces codes utilisent généralement des technologies informatiques répondant à leur besoin d'exécution, voir d'encapsulation quand c'est nécessaire. Mais les solutions mises en place sont souvent propriétaires et n'offrent pas d'ouverture au pilotage simple et efficace.

Pour parvenir à des solutions informatiques performantes et évolutives, il est judicieux de repositionner le problème du développement logiciel par rapport aux techniques

actuellement disponibles [RUB 97]. Quelques-unes de ces techniques peuvent être mises en avant pour aider à la réutilisation de code, telles que les composants logiciels. D'autres technologies peuvent être mises en avant pour la mise en œuvre de ces composants communicants, telles que les techniques d'appel de procédures distantes [FLO].

II.B. Appel de procédure distante à objets répartis

II.B.1. Créer une interopérabilité

Différents types de RPC (*Appel de Procédure Distante : Remote Procedure Call*) existent [FLO], mais les plus courants correspondent aux RPC à objets répartis, permettant de faire correspondre des langages de type C et les langages à objets de type C++/Java.

Java RMI[22] (*Remote Method Invocation*) permet par exemple de faire un appel distant entre méthodes écrites en Java. CORBA[23] de l'OMG (*Object Managment Group*) ou DCOM[24] de Microsoft, permettent quant à eux de faire communiquer des objets hétérogènes.

Les logiciels à intégrer nous imposent généralement à la fois la plate-forme (*Windows, Unix,* …) et le langage de programmation (*Java, C/C++, fortran, cobol,* …). Afin d'assurer l'interopérabilité malgré l'hétérogénéité, différentes technologies peuvent être utilisées :

- JNI[25] (*Java Native Interface*) : permet de gérer l'hétérogénéité de langage entre Java et les codes natifs de la machine tels qu'une DLL Windows (*Dynamic Link Library*). Ainsi un code développé en C/C++ voir même en fortran (*grâce à une passerelle fortran→C*), compilé dans une DLL, pourra être appelé à partir d'un programme en Java.
- RMI (*Remote Method Invocation*) : permet à partir de codes Java, de faire des appels distants (*différentes machines*) et donc de gérer des hétérogénéités de système d'exploitation. Cette technique associée à JNI permet alors de gérer les deux types d'hétérogénéité.
- CORBA : permet d'interfacer deux codes (*Java, C/C++, Cobol*) pouvant être exécutés sur des ordinateurs différents (*Windows, UNIX,* …).
- D'autres technologies telles que DCOM (*Distributed Component Object Model*) de Microsoft permettent de réaliser de l'interopérabilité entre différents langages de manière distribuée, mais privilégie la plate-forme Windows NT.

[22] Java RMI (*Remote Method Invocation*): java.sun.com/products/jdk/rmi

[23] CORBA (Common Object Request Broker Architecture) : www.omg.org/corba

[24] DCOM (Distributed Component Object Model) : www.microsoft.com/com/tech/DCOM.asp

[25] JNI (Java Native Interface) : java.sun.com/products/jdk/1.2/docs/guide/jni

II.B.2. Choisir son RPC

Des études ont montré que l'utilisation de telle ou telle technologie était essentiellement guidée par le contexte d'utilisation [TAL 98]. La nature hétérogène d'un système d'information nécessite par exemple de s'orienter d'avantage vers un standard tel que CORBA plutôt que DCOM ne supportant que le système d'exploitation de Microsoft [MET 98].

Les performances que peuvent apporter une architecture complexe telle que CORBA par rapport à des fonctionnalités similaires mises en œuvre par une technologie telle que RMI, ne sont visibles que pour des scénarii complexes, mettant en œuvre des appels provenant de plus de 5 clients simultanément. L'utilisation sur une même machine, conduit par exemple à des performances 40% moins bonnes par CORBA que par RMI [JUR 00].

II.B.3. Performance d'une interopérabilité par JNI et par CORBA

Il est possible de pousser encore plus loin la dégradation des performances d'une architecture telle que CORBA. En effet, en ne mettant en œuvre que ses capacités à gérer des langages hétérogènes, nous pouvons tester ses performances par rapport à une technologie telle que JNI. Ces deux technologies nous permettent de réaliser le même interfaçage (*Java→C*).

CORBA, au lieu de travailler avec une mémoire partagée, utilise son protocole de communication réseau lorsque le client et le serveur sont sur la même machine. L'interfacage spécifique réalisé en JNI sera alors nécessairement plus performant. Le coût doit évidemment être défini par rapport au temps effectif du service appelé. Il doit également être définit en fonction du contexte d'utilisation, c'est à dire du nombre d'appels effectués sur l'objet. A titre d'exemple, le *Tableau 9* est un comparatif du surcoût pour deux objets de calculs différents, et deux contextes d'utilisation extrêmes :

Tableau 9 : Comparaison d'interopérabilité utilisant JNI ou CORBA

	temps de calcul interfacé par JNI	*temps de calcul interfacé par CORBA*	*ratio CORBA/JNI*
objet de calcul 1	~70 ms	~2300 ms	33
objet de calcul 2	~16000 ms	~15680 ms	0.98

Le premier objet est un calcul rapide appelé 100 fois, faisant donc intervenir beaucoup d'appel à l'interfaçage (*JNI ou CORBA*). Sur cette configuration, le calcul utilisant l'interfaçage spécialisé (*JNI*) est 33 fois plus rapide que par CORBA. Le deuxième objet réalise quant à lui un calcul plus long (*environ 16 sec*), et seulement trois appelées réseaux sont faits (*un premier définissant des grandeurs d'entrée, un deuxième demandant d'effectuer le calcul et un troisième récupérant les grandeurs de sortie*).

On remarque que sur cet exemple, l'objet CORBA est plus rapide. Le fait que l'on soit du même ordre de grandeur est dû au temps de calcul effectif qui est beaucoup plus grand que les temps passé à faire communiquer les interfaces (*JNI ou CORBA*). Le fait que CORBA soit plus rapide est dû à la compilation de l'objet de calcul qui est différente. En effet, l'interfaçage par JNI requiert une DLL (*librairie à chargement dynamique*), alors que l'objet CORBA est un exécutable (*.EXE, qui « tourne » déjà avant qu'on ne l'appelle*).

II.B.4. Performance de CORBA selon la « distance réseau » des codes de calcul.

Dans cet exemple, les capacités de distribution de CORBA n'étaient pas sollicitées. Nous allons maintenant mettre en évidence les performances de CORBA selon la « distance réseau » des codes de calcul (*cf. Tableau 10*).

L'objet 2 est le même qu'étudié précédemment (*environ 16ms de calcul effectif*) appelé dans les même conditions. L'objet 1 correspond à un calcul extrêmement rapide (*de l'ordre de la milliseconde*), appelée 1 fois, puis 100 fois, puis 1000 fois, à partir d'un autre code de calcul, interfacé par CORBA. Nous indiquons le facteur multiplicatif du temps par rapport au temps du calcul effectué sur la même machine. Les temps sont calculés à partir d'une moyenne de 5 valeurs pour prendre en compte les surcharges temporaires du réseau.

Tableau 10 : Comparaison d'appel distant par CORBA selon la « distance » réseau

		sur le même ordinateur	*sur le même réseau local*	*sur deux sous-réseaux différents*
objet de calcul ***1***	*temps (ms) :*	~40 ms	~50 ms	~130 ms
	ratio / local :	1	~1.25	~3.25
objet de calcul ***1*** *appelé 100 fois*	*temps (ms) :*	~278 ms	~370 ms	~904 ms
	ratio / local :	1	~1.33	~3.25
objet de calcul ***1*** *appelé 1000 fois*	*temps (ms) :*	~1637 ms	~2430 ms	~6180 ms
	ratio / local :	1	~1.48	~3.77
objet de calcul ***2***	*temps (ms) :*	~15680 ms	~15850 ms	~16725 ms
	ratio / local :	1	~1.01	~1.066

Le premier résultat que l'on peut déduire de ces tests est que la distribution (*par Internet par exemple*) n'est intéressante que pour des calculs lourds. Il sera préférable de télécharger des codes portables en les rapprochant le plus possible de son réseau local pour les exécuter.

Le deuxième résultat concerne le nombre d'appels à l'objet de calcul. On remarque que le ratio entre le temps local et la distance réseau, n'évolue pas beaucoup en fonction de l'augmentation du nombre d'appels de l'objet de calcul 1. On remarque également pour des nombres de plus en plus élevés d'appels sur le même ordinateur (*colonne 1*), que le temps ramené à 1 appel (*temps total / nombre d'appels*) baisse considérablement (*40/1, 278/100, 1637/1000*), s'approchant du temps réel de calcul de la routine (*~1ms*) (*cf. Figure 97*).

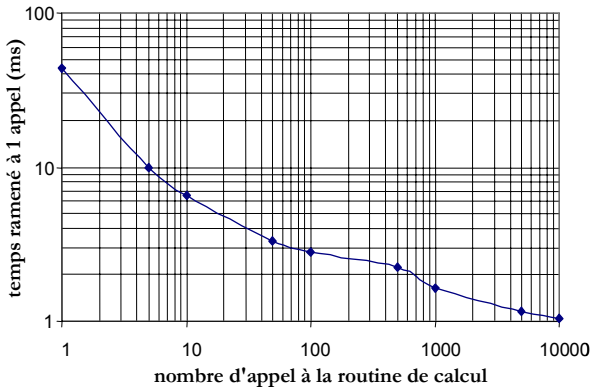


Figure 97 : Réduction du temps d'appel distant, ramené à un appel, en fonction du nombre d'appels effectivement réalisé

Afin de réduire le temps d'un calcul distribué, il est important de vectoriser les communications, afin de réduire le nombre d'appels réseau. Dans l'exemple de l'objet 2, les appels sont réduits à 3, grâce à une affectation de valeur par liste et non de manière individuelle. Grâce à cela, il est possible de réduire les temps lors d'une optimisation distribuée, d'un modèle analytique de machine synchrone à aimants, de 200 fois le temps mis en local, à 40 fois [DEL 02a].

Ces facteurs sont importants à retenir dans le cadre de projets de bibliothèques Internet de composants de calcul pour des dispositifs électromagnétiques. Les modèles de composants dessinés à des applications distribuées doivent en effet être conçus de manières différentes de ceux destinés à des applications locales.

II.B.5. Les nouvelles plate-forme de développement gérant l'hétérogénéité et l'interopérabilité

La problématique d'hétérogénéité que nous avons cherché à décrire ici est très présente en ingénierie informatique. Outre les technologies permettant de faire interopérer des langages et systèmes différents, de nouvelles plate-formes de développement logiciel apparaissent cherchant à masquer les hétérogénéités informatiques et la gestion des communications inter-modules. Java se propose notamment d'être un langage indépendant de la plate-forme d'exécution (*code mobile*), faisant de lui un prétendant puissant pour toute application

distribuée, en particulier par Internet. Microsoft a mit sur le marché, son environnement de développement « .NET », permettant entre autre de faire interopérer les langages qu'il distribue (*C++, C#, Visual Basic et Jscript*) de manière très forte, offrant par exemple l'héritage de classes entre langages.

Les développements futurs seront donc normalement moins sujets à la problématique d'hétérogénéité, bien que les codes de calculs développés, souvent très lourd (*résolution numérique,* …), sont préférablement compilés sur des systèmes informatiques bien précis afin d'en optimiser les performances. Le fait est qu'aujourd'hui, cette problématique est toujours d'actualité et qu'il est nécessaire d'utiliser des solutions technologiques existantes (*CORBA ou autres*) afin d'y répondre, tout en ayant les notions suffisantes pour choisir la technologie la plus appropriée à la situation.

Index

BIBLIOGRAPHIE,
FIGURES ET TABLEAUX

B I B L I O G R A P H I E

[ATI 00] E. Atienza, F. Wurtz, J. Bigeon, M. Perrault, V. Mazauric "a Methodology for the Sizing and the Optimisation of an Electromagnetic release", IEEE Transactions on Magnetics, vol. 36, No. 4, July 2000, pp. 1659-1663

[ATI 99] E. Atienza, J. Bigeon, F. Wurtz, B. Belhabib "Steps to an Electrical Design Environment", IEEE-IECON'99, San Jose, CA, USA, November 29th - December 3rd, 1999.

[ATI 03] E. Atienza, « Méthodologie et Outils pour le Dimensionnement », Thèse de Doctorat spécialité Génie Electrique de l'Institut National Polytechnique de Grenoble, 04 juillet 2003

[BAR 00] A. Bargiela, "Strategic Directions in Simulation and Modelling", CPHC Meeting, Manchester, 6-7 January 2000.

[BARB 02] F. Barbier, C. Cauvet, M. Oussalah, D. Rieu, S. Bennasri, C. Souveyet « Composants dans l'ingénieurie des systèmes d'information : concepts clés et techniques de réutilisation », actes des deuxièmes assises nationales du GdR I3 (Information - Interaction - Intelligence). Nancy, déc. 2002.

[BAS 00] Basma Bel Habib " Méthodologie pour le développement de plateformes intégrées de conception en génie électrique", thèse INPG Génie Electrique, juillet 2000

[BOU 00a] J.-F. Boujut A. Jeantet, J.-C. Sardas, « Enjeux et formes des pratiques coopératives dans la conception », Séminaire PROSPER « Coopération en conception et en exploitation dans les systèmes de production », Jeudi 17 février 2000, Ecole des Mines de Paris

[BOU 00b] J.-F. Boujut, E. Blanco, « Intermediary objects as a means to foster co-operation in engineering design », International Workshop on the Role of Objects in Design CO-Operation, Sophia Antipolis, May 23, 2000

[CAR 97] D. Cardon « Les sciences sociales et les machines à coopérer : une approche bibliographique du CSCW », La coopération dans les situations de travail, Réseaux n°85 CNET – 1997

[CAL 01] M. Caldora Costa "Optimisation de dispositifs électromagnétiques dans un contexte d'analyse par la Méthode des Éléments Finis", thèse de doctorat spécialité génie électrique de l'Institut National Polytechnique de Grenoble, 2001

[CHA 93] B. Chandrasekaran, A.K. Goel, Y. Iwasaki, "Functional representation as design Rationale", IEEE computer, Vol 1, Jan 1993, pp 48-56.

[CHE 01] J.Z. Chen, Y. Wu, C. Gence, D. Borojevich, J.H. Bøhn, "Integrated Electrical and Thermal Analysis of Integrated Power Electronics Modules Using

iSIGHT," Proc., 2001 CPES Power Electronics Seminar, Blacksburg, VA, April 23-25, 2001, pp. 141-145

[COU 85] J.L. Coulomb, J.C. Sabonnadière, « CAO en électrotechnique », Hermes Publishing, 1985.

[COUT 99] C.Coutel, « Contribution méthodologique à la conception sous contraintes de dispositifs électromagnétiques » Thèse de Doctorat spécialité Génie Electrique de l'Institut National Polytechnique de Grenoble, 20 octobre 1999

[DAR 01] F. Darses, F. Détienne, W. Visser « Assister la conception : perspectives pour la psychologie cognitive ergonomique », ÉPIQUE 2001, Actes des Journées d'étude en Psychologie ergonomique, Nantes, IRCCyN, France, 29-30 Octobre 2001

[DEL 00] B. Delinchant « Contribution à l'approche composant pour la conception en génie électrique », DEA INPG, spécialité Génie Electrique, 2000.

[DEL 02a] B. Delinchant, F. Wurtz, E. Atienza, J. Bigeon "Benefits of component methodology applied to electrical software", ELECTRIMACS'02, 7th International Conference on Modeling and Simulation of Electric Machines, Converters and Systems, Montréal, Québec, Aug. 18-21, 2002

[DEL 02b] B. Delinchant, E. Atienza, L. Gerbaud, F. Wurtz, "Concurrent design versioning system, based on XML file", IECON'02, 28th Annual Conference of the IEEE Industrial Electronics Society, Sevilla, Spain, Nov. 5-8 2002. ISBN: 0-7803-7474-6, pp 2485-2490

[DEL 02c] 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. Vol. 45, No. 4, Dec. 02, pp. 231-249.

[DEL 03] B. Delinchant, F. Wurtz, E. Atienza, "Reducing sensitivity analysis time cost of compound model", COMPUMAG'03, 14th Conference on the Computation of Electromagnetics Fields, July 13 - 18, 2003, in Saratoga Springs, New York, USA.

[DELA 93] J. Delamare, E. Rullière, J.P. Yonnet, « 3D calculation of permanent magnet interactions ». IMACS TC'1 Canada Juillet 93, pp 289,292.

[DEW 01] Alita Dewi "Apport des Nouvelles Technologies Interactives pour l'Analyse Intégrée en Génie Électrique : vers un Laboratoire Virtuel d'Expérimentation en Électrotechnique", thèse INPG Génie Electrique, juillet 2001

[EYN 00] B. Eynard, « Intégration de connaissances et capitalisation de savoir-faire : outils d'aide en conception », séminaire Groupement pour la Recherche en Productique, Annecy, les 23 et 24 Mars 2000.

[FAN 99] J. Fandino, F. Wurtz, J. Bigeon, "Nouvelle méthodologie de conception de dispositifs électriques », Revue Internationale de Génie Électrique (RIGE), Volume2-N° 1/1999

[FIS 02] V. Fischer, J. Bigeon, E. Atienza, B. Delinchant "Communication Structure Between Electrical Design Software, ELECTRIMACS'02, 7th International Conference on Modeling and Simulation of Electric Machines, Converters and Systems, Montréal, Québec, Aug. 18-21, 2002

[FLO] Gérard Florin, "Modèles d'interactions pour le client serveur et exemples d'architectures les implantant", Cours du Conservatoire National des Arts et Métiers.

[GEN 91] A. Gentilhomme, "C.O.C.A.S.E Un système expert d'aide à la conception d'appareillages électriques", Thèse spécialité génie électrique de l'Institut National Polytechnique de Grenoble, 3 Mai 1991

[GEN 92] A. Gentilhomme, J. Bigeon, J.L. Coulomb, M. Lauraire (1992),"An expert system for preliminary design of contactors", IEEE Transaction on Mag. VOL-28, No 2, March 1992, pp 1763-1766

[GER 93a] L. Gerbaud "Aide à la conception des ensembles machine – convertisseur – commande : Apport d'une démarche système expert", thèse de doctorat spécialité génie électrique de l'Institut National Polytechnique de Grenoble, 23 avril 1993

[GER 93b] L. Gerbaud, J. Bigeon, G. Champenois, "Expert system bases to automate selection of drive structures", IEEE-IECON'93, International Conference on Industrial Electronics control, and Instrumentation, Lahaina, Maui, Hawaii, USA, November 15-19, 1993, pp360-365.

[GERO 85] Gero, J. S. (ed.) (1985). Design Optimization, Academic Press, New York, 298pp.

[GOD 85] Ernest Nagel, James R. Newman, Kurt Gödel, Jean-Yves Girard, « Le théorème de Gödel » édition du Seuil, ISBN 2.02.032778.3

[GIL 91] J.C. Gilbert, G. Le Vey, J. Masse « La différentiation automatique de fonctions représentées par des programmes » Rapport de recherche n°1557, INRIA, novembre 1991.

[HAT 94] K. Hatefi, « La conception assistée par ordinateur, de moteurs et entraînements électriques à aimants permanents », Diplôme de Doctorat ès sciences techniques, Ecole Polytechnique Fédérale de Lausanne, 1994

[HOF 85] Douglas Hofstadter, « Gödel, Escher, Bach. Les Brins d'une Guirlande Eternelle », InterEditions, 1985

[IRM 96] G. Irmela, « the impact of society on CAD research in the U.S.A., France and Germany, 1955 through 1985 » Perrin et Vinck editions, The role of design in the social shaping of technology, Bruxelles : Cost A4, 1996 (vol.5), 155-183

[JUF 95] M. Jufer, « Electromécanique », Traité d'Electricité, vol IX, Presses Polytechniques et Universitaires Romandes, Lausanne, 3ème édition, 1995

[JUR 00] M.B. Juric, A. Zivkovic, I. Rozman, « Performance Comparision of CORBA and RMI », Information and Software Technology Journal, Elsevier Science, October 2000, vol. 42, no. 13, pg. 915-933

[LAU 97] P. Laureillard, Boujut Jean-Francois, Alain Jeantet, « Conception Intégrée et Entités de Coopération » 01DESIGN, Les objets en conception pp 119-134, Théoule-sur-mer , 1997 B. Trousse, K. Zreik, ISBN 2-909285-09-3 , pp .292. 1997.

[LAU 00] P. Laureillard, « Conception intégrée dans l'usage. Mise en œuvre d'un dispositif d'intégration produit-process dans une filière de conception de pièces forgées », Thèse de Génie Industriel, Grenoble : INPG-UJF, CRISTO-3S, 12 Janvier 2000

[LAV 00] O. Lavoisy, « La matière dans l'Action : Le graphisme technique comme instrument de la coordination industrielle dans le domaine de la mécanique depuis trois siècles », Thèse de Génie Industriel, Grenoble : INPG-UJF, CRISTO-3S, 14 décembre 2000

[LEC 98] Ch. Lechevalier, "Analyse fonctionnelle des convertisseurs statiques en vue de la conception", Thèse de doctorat en génie électrique de l'Institut National Polytechnique de Grenoble, France, 30 novembre 1998

[MA 01] Singva Ma, « Définition d'un Protocole d'Application STEP pour la Simulation en Electromagnétisme », Thèse INPG Génie Electrique, 29 Juin 2001.

[MAL 80] A. Malhotra, J. Thomas, J. Carroll, and L. Miller, "Cognitive Processes in Design", International Journal of Man-Machine Studies (12) 1980, pp. 119-140.

[MAG 03] D. Magot, F. Wurtz, B. Cogitore, B. Delinchant, J.-P. Keradec. "A Methodology and Tools for Worst-Case Tolerance Design", COMPUMAG'03, 14th Conference on the Computation of Electromagnetics Fields, July 13 - 18, 2003, in Saratoga Springs, New York, USA

[MAU 00] P. M. Maurer. "Components: What if they gave a revolution and nobody came" IEEE Software, 33(6):28-34, June 2000

[MEI 97] T.D. Meijler and O. Nierstrasz, "Beyond Objects: Components," Cooperative Information Systems: Current Trends and Directions, M.P. Papazoglou and G. Schlageter (Eds.), pp. 49-78, Academic Press, November 1997

[MET 98] META Group Consulting, « CORBA vs. DCOM: Solutions for the Enterprise ». 20 Mars 1998.

[MOI 90] J.-L. Le Moigne « La modélisation des systèmes complexes » Ed. Dunod, Collection Sciences humaines, ISBN : 210004382X, 192 pages, , 1990

[MOI 95] J.-L. Le Moigne « Les épistémologies constructivistes » Paris, P.U.F., Que Sais-Je ?, N°2969, 1995

[MOR 99] Edgar Morin, Jean-Louis Le Moigne. « L'intelligence de la complexité », Collection cognition et formation, L'Harmattan, Paris, 1999, ISBN : 2-7384-8085-3

[MOR 00] Edgar Morin « Introduction à la pensée complexe », Communication et Complexité, ESF Editeur, Paris, 7ème tirage 2000, ISBN 2-7101-0800-3

[PER 94] Y. Perriard, M. Jufer, "Linear reluctant transducer for vibrator application", 6th Biennal IEEE Conference on Electromagnetic Field computation, CEFC'94, Aix-les-Bains, Juillet 1994, pp 130.

[POL 86] M. Poloujadoff, R.D. Findlay, "A Procedure for illustrating the Effect of Variation of Parameters on Optimal Transformer Design", IEEE Transactions on Power Systems; Vol. PWRS-1, No. 4, November 1986

[RIB 02] V. Riboulet, B. Delinchant, L. Gerbaud, Ph. Marin, F. Noël, "Tools for Dynamic Sharing of Collaborative Design Information" KLUWER Academic Publishers, in the book "Recent Advances in Integrated Design and Manufacturing in Mechanical Engineering", Editors : G. Gogu, D. Coutellier, P. Chedmail and P. Ray.

[RUB 97] Luc Rubiello, « Techniques innovantes en informatique ». Paris, editions Hermes. ISBN : 2-866-01587-8, 1997

[SIM 87] H.A. Simon "Decision Making and Problem Solving", Decision Making and Problem Solving", Interfaces, Sep/Oct 1987; Vol. 17, Iss. 5; pg. 11, 21pgs

[SAR 99] B. Sareni, "Méthodes d'Optimisation Multimodales Associées à la Modélisation Numérique en Electomagnetisme", Thèse de Doctorat, École Centrale de Lyon, 1999

[SAS 94] R.M. Sassine, D.A. Lowther, A Knowledge-Based Environment for Linking Electromagnetic Design Tools, IEEE Transactions on Magnetics, Vol.30 No.5, sept 1994.

[SAU 00] Ch. Sauvey « Contribution méthodologique à la modélisation pour le dimensionnement de moteurs à réluctance variable » Thèse de Doctorat spécialité Génie Electrique de l'Institut National Polytechnique de Grenoble, 8 septembre 2000

[SHA 96] M. Sharples "An Introduction to Human-Computer Interaction", in M. Boden (ed.)Artificial Intelligence, Academic Press, 1996, pp. 293–323.

[SIL 02] M.G. da Silva, R. Giasolli, S. Cunningham, D. DeRoo, "MEMS Design for Manufacturalbility (DFM)", Sensors Expo & Conference 2002. Boston MA, September 25, 2002

[STO 01] E. A. Stohr, J. Leon Zhao, Workflow automation : Overview and Research Issues, Information Systems Frontiers 3:3, 281-296, 2001

[SZY 98] C. Szypersky, "Component Software – Beyond Object-Oriented Programming", Addison-Wesley, 1998

[TAL 98] O. Tallman and J. Bradford Kain, « COM versus CORBA: A Decision Framework », in Distributed Computing, September-December 1998

[THO 95] M. Thonnat, S. Moisan, ``Knowledge-based systems for program supervision", in: First international workshop on Knowledge-Based systems for the (re)Use of Programs libraries KBUP'95, INRIA, Sophia Antipolis, France, 1995

[TOM 85] T. Tomiyama and H. Yoshikawa: "Requirements and Principles for Intelligent CAD Systems," in J.S. Gero (ed.): Knowledge Engineering in Computer-Aided Design, North-Holland, Amsterdam, (1985), pp. 1-23.

[TRI 91] F. Trichon, "Modélisation du processus de conception des machines électriques. Le système expert DAMOCLES ", Thèse spécialité génie électrique de l'Institut National Polytechnique de Grenoble, 12 mars 1991

[VAS 94a] J. A. Vasconcelos, "Optimisation de Forme des Structures Électromagnétiques", Thèse de Doctorat, École Centrale de Lyon, 1994.

[VAS 94b] J.A. Vasconcelos, L. Krähenbühl, L. Nicolas, A. Nicolas: "Design optimization in electrostatic field analysis using the BEM and the augmented Lagrangian method", IEEE Trans. on Magnetics, vol. MAG 30, n°5, September 1994, pp. 3443-3446.

[VIV 02] S. Vivier, « Stratégies d'optimisation par la méthode des plans d'expériences et application aux dispositifs électrotechniques modélisés par éléments finis », Thèse de Doctorat spécialité Génie Electrique, Ecole Centrale de Lille, 11 juillet 2002.

[WU 02] Y. Wu, J. H. Bohn, D. Boroyevich « Software Integration for IPEM Design, Modeling, and Analysis » Virginia Polytechnic Electronics Center (VPEC), Center For Power Electronics Systems (CPES), Seminar Proceedings, pp.559, 2002

[WUR 96a] F. Wurtz, "Une nouvelle approche pour la conception sous contraintes de machines électriques » Thèse de Doctorat spécialité Génie Electrique de l'Institut National Polytechnique de Grenoble, 28 mai 1996.

[WUR 96b] F. Wurtz, J. Bigeon, Poirson C. « A methodologie and a tool for the computer aided design with constraints of electrical devices », IEEE Transactions on Magnetics, vol. 32, Iss. 3, May 1996, p.1429-1432

[WUR 97] Wurtz F., Espanet C., Bigeon J., Kauffmann J-M. "Methodological guidelines for the use of analytical and numerical models in a design process of an electromagnetic device". COMPUMAG'97, Rio de Janeiro, Brasil, 3-6 Novembre 1997

I. Index des figures et illustrations

II. Index des tableaux