Méthodologie pour le Développement de Plates Formes Intégrées dédiées à la Conception en Génie Electrique

Basma Bel Habib

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la Bibliothèque

THESE

Pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : Génie Electrique

Préparée au **laboratoire d'Electrotechnique de Grenoble**

Dans le cadre de **l'Ecole Doctorale** « Electronique, Electrotechnique, Automatique, Télécommunications, Signal »

Présentée et soutenue publiquement

par

BEL HABIB Basma

Le : 13 Juillet 2000

« Méthodologie pour le Développement de Plates Formes Intégrées dédiées à la Conception en Génie Electrique »

Composition du Jury

Rapporteurs :	FELIACHI	Mouloud
	PIRIOU	Francis
Examinateurs :	BIGEON	Jean
	BRISSAUD	Daniel
	PIQUET	Hubert
	WURTZ	Frédéric

A mes parents

A Faouzi

Remerciements

Je tiens à remercier:

Monsieur Piriou Francis, Professeur de l'Université des Sciences et Technologies de Lille (USTL), pour m'avoir fait l'honneur de présider mon jury de thèse et avoir accepté d'être rapporteur de ce travail.

Monsieur M. Feliachi Mouloud, Professeur des Universités à l'I.U.T. de Saint Nazaire, qui a également accepté d'être rapporteur de cette thèse.

Monsieur Brissaud Daniel, Directeur adjoint du laboratoire 3S (Sols, Solides et Structures) et Professeur de l'INPG, pour avoir participer à mon jury de thèse et pour l'intérêt qu'il a porté aux travaux que j'ai menés.

Monsieur Bigeon Jean, Directeur de Recherches au CNRS et chef de l'équipe Conception et Diagnostics Intégrés du LEG (Laboratoire d'Electrotechnique de Grenoble), pour m'avoir proposé cette thèse et m'avoir initié à l'activité de conception. Tout au long de cette thèse, outre son esprit de synthèse, j'ai apprécié particulièrement la confiance qu'il m'a accordée pour mener cette aventure scientifique.

Monsieur Wurtz Frédéric, Chargé de Recherches au CNRS, pour avoir accepté de participer à l'encadrement de ma thèse. Il n'a pas ménagé son temps pour l'aboutissement de mon travail et il m'a été d'une grande aide. Qu'il soit rassuré de mon estime.

Je remercie également le Directeur du Laboratoire d'Electrotechnique de Grenoble M. Rognon ainsi que le Directeur adjoint M. Meunier pour m'avoir accueilli au sein de celui ci.

Durant mon passage au LEG (pour mon DEA et ma thèse) une multitude de personnes m'ont rendu service à un moment ou un autre, je tiens à les saluer tous. En particulier les personnes qui je vais citer dans la suite de ses remerciements. Que les personnes citées en fin de liste se rassurent, il s'agit uniquement d'un classement et non d'une échelle de valeur. Je remercie:

M. Bolopion Alain, maître de conférences à l'INPG, pour avoir accepté de lire en avant première mon manuscrit de thèse et pour ces précieuses aides et conseils techniques.

M. Gerbaud Laurent, Chargé de Recherches au CNRS, pour ses conseils non seulement scientifiques mais surtout en matière de "gestion des gamins": j'essaierai de les suivre en temps voulu. D'autre part: je t'avoue que je déclare forfait car je n'ai pas pu battre ton record d'arrivée le matin, malgré ma bonne volonté

Un grand merci à Atienza Eric, "my private hot line", pour sa disponibilité et pour sa grande contribution dans les travaux de ma thèse. Je n'oublierai pas les discussions d'ordre diverses et variées que nous avons menés tout au long des deux dernières années. Dommage que tu sois unique en ton genre, mais malheureusement tu n'es pas parfait: tu aurais besoin de cours de rattrapage en arabe.

Pour les nouveaux arrivants dans l'équipe CDI, Benoit Delinchant et Alain Loig, je vous souhaite bonne chance. Une pensée particulière à Benoit qui va prendre la relève pour assurer la continuité des travaux entamés dans le cadre de ma thèse.

Mes amitiés à Armando Fonseca, Charlotte Gillot, tous deux que je connais depuis mon arrivée à l'IEG depuis six ans: je suis contente d'avoir pu vous connaître de plus prés pendant ces trois années de thèse. Merci Charlotte (et Laurent) pour m'avoir initié à l'escalade et m'avoir fait découvrir les montagnes de Grenoble. Merci Armando pour ton Amitié.

Jean Michel Guichon: n'est-il pas vrai que ce virus CDI est contagieux? Je te souhaite une bonne guérison et merci également pour ton aide.

Rafika Berrouche, Ammari Sami, tous deux ayant fait le même choix que moi: partir à l'étranger à la quête du savoir, vous étiez amènes de comprendre mes états d'âmes: merci de m'avoir remonté le moral quant j'en avais besoin.

Je tiens à saluer Elise Riado, Monique Boizard, Danielle Collin pour leur bonne humeur.

Je remercie Florence François, dont j'admire le courage, pour sa serviabilité.

Je tiens en particulier à remercier Bruno Mallet pour ses qualités humaines et pour son écoute.

En cette de fin de thèse, je ne peux m'empêcher d'évoquer ma famille: mes parents, ma sœur Sihem, mon frère ainsi que ma Tata: ils ont chacun à sa manière contribuer à l'aboutissement de cette thèse, malgré les kilomètres qui nous séparent. Une pensée particulière à mes parents, cette thèse leur est dédiée en premier.

Malgré qu'il a échappé à la correction du manuscrit, merci Faouzi d'avoir accepté mon choix, et de m'avoir soutenu tout au long de cette expérience.

Je terminerai enfin en remerciant toute la clique de mes compatriotes Lamia, Sami, Rym, Ekbel, Takoua, Wassel, Fatma et Hager (qui a fait du chemin pour assister à ma soutenance) ainsi que Faouzia, Mehdi et Kathia pour leur amitié.

Table des matières

Table des Figures

Introduction

Problématique abordée

La phase de préconception, est la phase de naissance d'un produit. C'est une phase courte dans son cycle de vie mais c'est une phase décisive puisqu'elle conditionne une grande part de la réalisation du produit.

Avec l'évolution de l'informatique, des outils d'aide à la préconception par ordinateur, de plus en plus performants, sont développés pour assister les concepteurs électriques dans le dimensionnement des dispositifs à concevoir.

Toutefois, nous avons constaté que la majorité des outils de préconception existants actuellement ont été conçus pour fonctionner d'une manière autonome. Or de plus en plus, les concepteurs sont amenés à faire converger les informations fournies par les différents outils pour trouver la solution optimale.

Le besoin d'un outil d'aide à la préconception complet qui assumerait la totalité de l'activité de conception, qui prendrait en charge aussi bien la caractérisation du produit, que celle du processus de conception demeure insatisfait.

En réponse à ce constat, ce travail de thèse propose une méthodologie pour le développement de plate forme intégrée dédiée à la préconception en génie électrique. Nous présentons une architecture informatique permettant d'automatiser partiellement l'interaction entre les outils et de réutiliser les processus de conception menés par le concepteur.

Nous avons mis en place une démarche pour réaliser des couplages réutilisables entre les outils hétérogènes de conception permettant le transfert de paramètres et le partage de leurs fonctionnalités. Pour l'implémentation du modèle proposé, nous avons mis en œuvre un modèle Serveur d'Objets en Java en encapsulant les logiciels de conception dans des "Composants".

Plan du document

La thèse comporte cinq chapitres:

(1) Connaître l'activité de conception

(2) Etat de l'art de la conception en génie électrique

(3) Les Plates Formes Intégrées pour la Conception en Génie Electrique

(4) Méthodologie de développement d'une plate forme intégrée

(5) Prototype d'une plate forme de conception intégrée dédiée au génie électrique

Le premier chapitre est un chapitre introductif de l'activité de conception. Nous montrons son importance dans la vie d'un produit. Après un rappel de la définition que nous adoptons, nous présentons un aperçu historique de l'évolution de l'exercice de la conception. Ensuite nous expliquons les différents types ainsi que les différentes phases de la conception. Pour finir ce chapitre, nous situons nos travaux de recherches par rapport à cette activité.

Dans le deuxième chapitre, nous commençons par exposer les problèmes rencontrés pour la modélisation de la conception. Ensuite le modèle produit et le modèle du processus sont analysés. Nous proposons une classification des outils d'aide à la conception selon une approche conceptuelle : c'est à dire en fonction des niveaux d'abstraction des modèles de produit et du processus qu'ils intègrent. Nous dégageons ainsi trois familles d'outils : les outils spécifiques, les outils génériques et les outils méta-génériques.

Dans le troisième chapitre, nous signalons l'absence d'un environnement assisté par ordinateur supportant le concepteur en génie électrique dans la totalité de son exercice. C'est pourquoi nous proposons une architecture informatique permettant l'usage des outils de conception en interaction. Nous introduisons ainsi le concept de plates formes intégrées dédiées à la conception en génie électrique ou EDIP (Electrical Design Integrated Platform). Ensuite, nous passons en revue le cahier des charges d'une EDIP, ses atouts tout en précisant ses pré-requis.

La méthodologie des plates formes intégrées fait l'objet du quatrième chapitre. Nous expliquons la méthodologie pour transformer un outil de conception donné en un composant logiciel facilement intégrable dans une EDIP. Ensuite, nous développons une technique d'assemblage des composants logiciels pour la mise en œuvre d'un processus de conception.

Enfin dans le cinquième chapitre, nous indiquons comment nous avons employé notre approche pour réaliser des composants à partir des logiciels PASCOSMA et SOLID EDGE. A partir de ces composants, nous détaillerons le prototype d'environnement intégré mis en place. Nous illustrons l'exemple du prototype réalisé par un exemple de scénario de processus de conception d'un électroaimant.

En conclusion, nous établissons un bilan de nos travaux et nous finirons par dégager les perspectives de notre approche.

Connaître l'Activité de Conception

Sommaire du premier chapitre :

I "Connaître l'activité de conception"

I.1 Enjeux de l'activité de conception

La conception a un enjeu fondamental pour le positionnement des entreprises sur le marché. Actuellement, nous pouvons affirmer que les corps de métiers ont acquis une bonne maîtrise des systèmes automatiques implantés en industrie (dans les chaînes de production et de fabrication) et des techniques de contrôle de la qualité. De gros progrès ont été réalisés dans les entreprises pour atteindre l'objectif « zéro défaut ».

Mais l'expérience montre que des améliorations restent à accomplir en matière de conception. D'un point de vue stratégique, une maîtrise de la conception est le garant d'une amélioration de la position concurrentielle d'une entreprise. Sa pérennité passe alors forcément par une garantie de transmission du savoir-faire de ses experts et par une grande réactivité à la demande du marché [Bhb98]. C'est pourquoi le développement des moyens de conception est un problème d'actualité.

Les coûts de cette activité sont estimés à 5% du coût global du produit conçu [CE99]. Mais la phase de Conception, à elle seule, représente une grande part de la réalisation du produit puisqu'une fois la spécification du « Comment réaliser l'objet » est faite, les tâches restantes sont a priori plus faciles à mettre en œuvre [Dej95]. Ce qui reste à faire relève d'un ordre pratique. Il s'agit d'exécuter les idées mûries par le ou les experts. « Environ 75% du coût prévisionnel d'un produit sont prédéterminés par les décisions en phases de conception (étude d'ensemble, études détaillées) » [Pol96].

C'est dans la phase de conception que doivent être explorées et exploitées au maximum les éventualités d'optimisation du produit (temps de développement, coût de revient en matériel et en mobilisation de l'outillage de fabrication). C'est pourquoi cette phase de réflexion sur la structure du produit, sur ses fonctions et sur son comportement (durant ses différents états d'utilisation) constitue la tâche la plus coûteuse en temps dans son cycle de production. Afin de mieux comprendre la place accordée à la conception, dans le cycle de vie d'un produit, nous rappelons dans ce chapitre l'évolution de cette activité.

I.2 Définition de l'activité de conception

Rappelons tout d'abord une définition de la tâche de **conception :** « Etant donné un ensemble de composants, un ensemble de relations entre ces composants, une tâche de conception consiste à **spécifier comment réaliser un objet** à l'aide de ces composants, de manière à satisfaire **un ensemble d'exigences**. Bien que dans certains types d'activités l'ensemble des composants ne soit pas toujours fini et que de nouveaux composants doivent être conçus, cela ne pose théoriquement pas de problème particulier car la **conception peut être récursive** » [Har97].

En adoptant cette définition de la conception, nous admettons le principe de décomposition ; « **Il consiste à décomposer un module en cours de conception en modules plus élémentaires** » [Sho95]. Le produit à concevoir est divisé en sous-ensembles définis à partir de règles spécifiques de décomposition. Tout produit peut donc être représenté sous la forme d'un arbre. Chaque embout de ramification correspond à l'un de ses composants et l'aspect système du produit serait introduit par la définition de liens entre les différentes branches.

En effet, cette approche de décomposition est adoptée aussi bien par les électriciens ([Esc97], [Tri], [Gen91]) que les mécaniciens [Els97] ainsi que les informaticiens [Har97]. Il est plus facile de gérer des systèmes compliqués (sièges de phénomènes interdépendants) en les décomposant en sous-systèmes bien définis que les concepteurs maîtrisent, couplés à des éléments qui permettent de gérer cet aspect systémique (liens, règles etc.) Implicitement, ils adoptent la définition fournie par le Dictionnaire Larousse du terme **Compliqué** : «1- composé d'un grand nombre d'éléments/ 2- difficile à comprendre ou à exécuter». Nous avons en principe à résoudre des problèmes difficiles à aborder. Très souvent, pour s'affranchir de ces difficultés de premier degré, nous faisons l'hypothèse que le problème est décomposable en sous-problèmes.

Cette décomposition réfléchie peut être guidée par une approche descendante du produit à concevoir. Ainsi pour chaque produit, les différents niveaux hiérarchiques sont organisés en une structure pyramidale, le produit global est situé en tête de la pyramide alors que les détails de ses sous ensembles se retrouvent au pied de la structure.

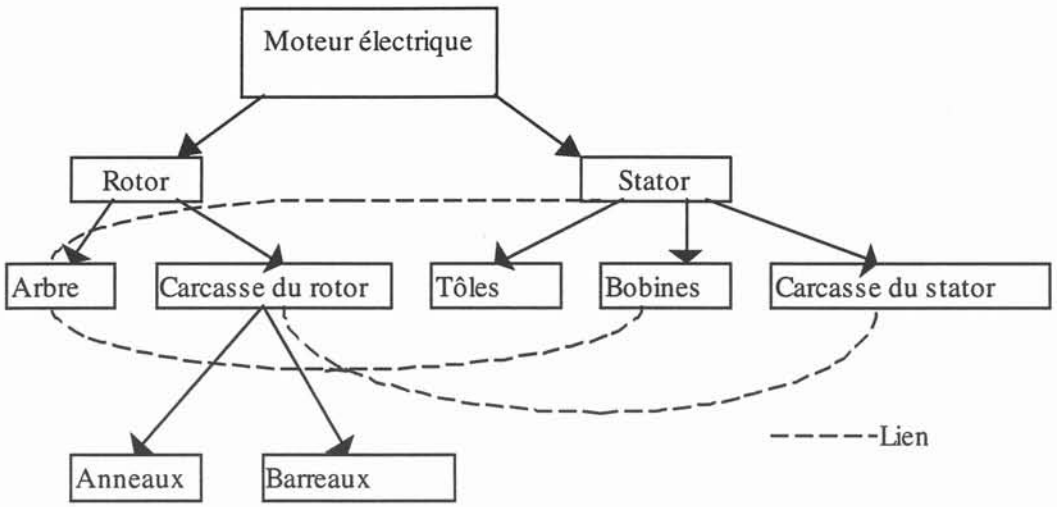


Figure I-1 Décomposition d'un moteur électrique à cage avec quelques liens possibles ([Dej95])

I.3 L'évolution de la Conception

L'évolution de l'activité de conception s'est produite progressivement en fonction de l'intégration du niveau d'abstraction dans cette démarche. Dans ce paragraphe, nous présentons un aperçu historique de cette évolution [Els97]. En effet, quatre périodes dans la vie de la conception se sont particulièrement démarquées :

- 1ère âge :

Dés le début, concevoir un produit quelconque consistait à fournir une réponse à la liste des quatre questions majeures « Quel est le produit? Quelles sont les utilités du produit? A partir de quels matériaux le concevoir? Comment procéder pour le réaliser? ». Toutefois, le travail des concepteurs était dévalorisé, ces derniers se situaient en bas de l'échelle des métiers puisque la conception était considérée comme une activité individuelle qui se réduisait à l'exécution manuelle des tâches.

- 2ème âge :

La Renaissance a donné lieu à la séparabilité dans l'activité de conception entre les tâches intellectuelles et les tâches exécutives ; l'effort intellectuel de réflexion dans la recherche des réponses aux questions a été reconnu et une vision moins superficielle de l'activité s'est détachée ; toutefois la conception reste une « activité intentionnelle et individuelle ».

- 3ème âge :

Il correspond à une conception spécialisée ; les concepteurs maîtrisent une discipline bien spécifique. Leur travail consiste à construire les informations nécessaires à la conception du produit souhaité et à les transmettre à l'exécution (aux compétences responsables de l'étape ultérieure de la chaîne de développement) d'où il apparaît le grand inconvénient de cette époque. En effet, le manque de dialogue entre les différents intervenants dans le cycle de développement d'un produit donné a induit des pertes en temps et en coûts non négligeables ; cette transmission séquentielle de l'information fait que l'incohérence des contraintes exigées par chacune des disciplines n'apparaît qu'a posteriori. Par exemple, un dimensionnement fait par un électromécanicien peut être réfuté par les unités de production faute de disponibilité des matériaux (épaisseur de tôles non conformes aux normes...).

- 4ème âge :

La quatrième ère est née d'un besoin urgent des concepteurs de remédier à ces pertes en temps provenant du manque de communication entre les différentes disciplines. Il semble évident qu'un concepteur de machines électriques doit avoir des acquis de bases en électricité, en mécanique et en thermique. Cependant les connaissances d'un expert aussi compétent qu'il peut l'être restent limitées et sont donc insuffisantes pour mener la totalité d'un projet de conception de moteur. Dans une telle activité, il faut prendre en compte les phénomènes spécifiques à chaque discipline ainsi que leurs interactions ; d'où l'idée de réunir les experts mono-disciplinaires dans une structure appelée plate-forme afin qu'ils puissent intervenir ensemble à chaque stade de la conception depuis le cahier de charges jusqu'à la proposition d'une solution du produit à concevoir. Il ressort de cette dernière évolution de nouveaux concepts de la conception.

Tout au long de ces âges, l'activité de conception a évolué d'une activité mono-acteur (1ère âge et 2ème âge) vers une activité multi-acteurs sur le produit (3ème et 4ème âges). La grande variante entre le 3ème âge et 4ème âge figure dans le mode de collaboration des différents intervenants sur le produit.

Par exemple, au cours du 3ème âge, la collaboration entre les ingénieurs du bureau d'études et l'expert en production se déroulait séquentiellement : ce dernier n'intervenait qu'après réception des plans détaillés du produit à concevoir.

Très souvent, une suite d'allers retours incessants est alors déclenchée faute de moyens de production du produit tel qu'il a été spécifié par le concepteur des bureaux d'études [Pol96]. En établissant la communication dès les premières phases de la conception (4ème âge), l'expert en production informerait le concepteur des limites matérielles permettant ainsi un gain important de temps et par suite une baisse des coûts de revient.

Actuellement, nous vivons dans ce 4ème âge de la conception c'est pourquoi nous consacrons le paragraphe suivant à la définition des concepts émergents.

I.4 Les concepts émergents de la conception

Nous souhaitons mettre l'accent sur trois nouveaux concepts:

- la conception concourante
- la conception intégrée
- l'ingénierie simultanée

I.4.1 La Conception Concourante

Le concept de conception concourante encore appelé co-conception, vise à l'intégration de la diversité imposée par la nature même de l'activité de conception. Les différents acteurs concourent vers une solution : "L'enjeu est de créer un dispositif au sein duquel l'ensemble des points de vue des acteurs doivent être pris en compte pour parcourir au développement du produit" [Lau00].

Ce travail collaboratif peut se dérouler dans un mode synchrone ou asynchrone. En effet, si les différents experts (pouvant être amenés à intervenir à un moment quelconque dans le cycle de vie du produit) travaillent au même moment sur le même produit, ils exercent alors une conception multi-acteurs synchrone. Dans le cas où ils exercent leur activité en décalage temporel, leur activité est qualifiée de conception multi-acteurs asynchrone.

Les acteurs d'une co-conception n'exercent pas forcément leur collaboration dans un même lieu : ils peuvent être répartis géographiquement.

I.4.2 La Conception Intégrée

De nos jours, des outils sont développés pour assister les concepteurs dans l'exercice de leur métier. Il se trouve que pour la conception d'un seul et même produit, les concepteurs font appel très souvent à plusieurs outils.

Le concept de conception intégrée vise à structurer l'usage des outils utilisés dans l'objectif : de concevoir un même produit. Ceci revient en pratique à trouver la réponse à la question «Comment exploiter au mieux les informations fournies par les différents logiciels afin de converger vers le meilleur produit ? »

Prenons l'exemple de la conception d'une machine électrique. Nous pouvons distinguer au minimum trois acteurs : le mécanicien, l'électrotechnicien et l'expert en thermique. Le mécanicien peut utiliser un outil de CAO pour modéliser la géométrie de la machine (tels que Pro Engineer, Solid Edge [Sev4], Catia). L'électrotechnicien peut utiliser un outil de Calcul (tel que Matcad) et un outil de simulation (tel que Flux2D [Flu2]). Quant à l'expert en thermique, il peut utiliser un outil métier (tel que le module thermique de Flux).

Cet exemple montre qu'il est possible de procéder à l'intégration des outils de deux manières. La première consiste à intégrer les outils utilisés par un seul acteur : c'est une conception intégrée mono-acteur. Dans l'exemple cité ci-dessus, ceci revient à organiser les outils utilisés par l'électrotechnicien. Une deuxième façon de procéder, consiste à structurer tous les outils indépendamment des acteurs qui les manipulent : c'est une conception intégrée multi-acteurs.

I.4.3 L'Ingénierie Simultanée

Le concept d'ingénierie simultanée est une mise en œuvre de la co-conception synchrone. C'est une organisation du travail des multi-acteurs dans une même temporalité : les différents intervenants dans la conception du produit sont sollicités simultanément.

Avant l'émergence de la conception simultanée, l'apport des divers experts est pris en compte d'une manière séquentielle : la collaboration était une intégration verticale.

En instaurant la communication en permanence entre les différents agents qui pourront mettre en cause la faisabilité ou la viabilité du produit proposé par le bureau des études, les temps des allers retours sont économisés. En effet, les erreurs éventuels commises par le concepteur seront trouvées par les autres experts depuis le début de la conception [CE99], ce qui lui permettra d'apporter les corrections nécessaires immédiatement.

Ces trois concepts émergents détaillés ci-dessus se distinguent essentiellement en fonction de trois paramètres : le ou les acteurs de la conception, le temps et l'organisation des outils de conception. Dans le cadre de nos travaux, nous allons contribuer à la mise en œuvre de la conception intégrée en génie électrique.

I.5 Les différents types de conception

Après examen des différentes typologies rencontrées dans la littérature ([Esc97], [Har97], [Dej95], [Sho95] et [Bij94]), nous proposons (cf. FigI-2) une matrice bi-dimensionnelle de l'activité de conception. Dans cette matrice, nous avons classé les différents types en fonction de deux critères : le premier étant l'information disponible sur le produit, le second étant l'investissement à faire pour sa réalisation.

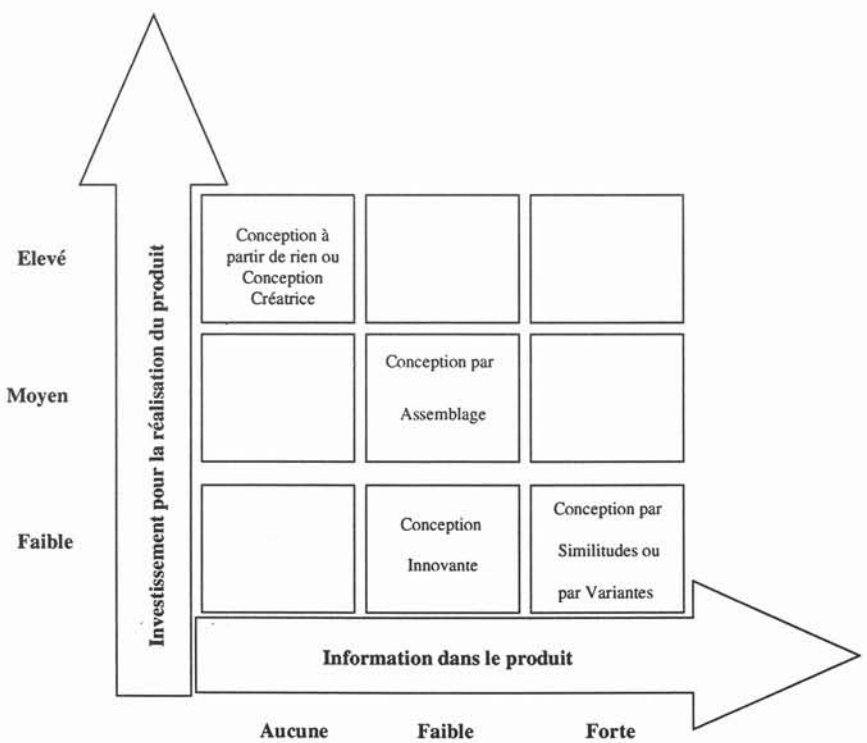


Figure I-2 Matrice de la Typologie de la conception

I.5.1 Axe de l'information disponible sur le produit

Selon la connaissance de l'objet de conception, nous distinguons dans l'ordre décroissant :

- La conception par similitude.
- La conception innovante.
- La conception créatrice.

I.5.1.1 La conception par similitude

Elle est aussi désignée par conception routinière [Esc97]. Les concepteurs s'appuient fortement sur les précédentes expériences de conception et réexploitent leur savoir-faire pour concevoir un produit similaire à un produit antérieur.

Dans ce cas l'ensemble des caractéristiques fonctionnelles, structurelles ainsi que les caractéristiques comportementales du produit sont bien définies au préalable [Har97]. La conception du produit à concevoir consistera :
- soit à reconduire une solution retenue lors d'une précédente réalisation.
- soit à apporter une amélioration ou une modification dans le dimensionnement du produit.

I.5.1.2 La conception innovante

La majorité des caractéristiques du produit sont à définir ; les concepteurs ont à formuler une nouvelle idée de produit à concevoir [Esc97]. Elle se traduit par l'ajout ou la suppression d'une ou plusieurs caractéristiques du produit afin de répondre à un besoin insatisfait par la première version du produit : le produit conçu est donc un nouveau produit à l'échelle de l'entreprise ou du bureau d'études [Har97].

I.5.1.3 La conception créatrice

Elle consiste à la définition de la totalité des caractéristiques du produit c'est à dire que le produit conçu est un produit nouveau aussi bien sur le marché qu'à l'échelle de l'entreprise ou du bureau d'études [Har97].

Lorsqu'il s'agit de poursuivre une lignée de production, la conception est menée en s'inspirant fortement du déjà fait, elle est dite par similitude, routinière ou encore conception prédéfinie. Alors que si le besoin se fait ressentir (suite à une demande spécifique du client ou afin d'assurer une poursuite concurrentielle sur le marché), il peut y avoir une rupture totale avec le produit existant en marquant le début d'un nouveau produit : la conception serait qualifiée de conception créatrice. Dans le cas où cette rupture ne serait que partielle, la conception est dite conception innovante.

I.5.2 Axe de l'investissement pour la réalisation du produit

Selon la méthodologie adoptée pour réaliser l'objet de conception, nous distinguons dans l'ordre croissant de l'importance de l'investissement à faire :

- La conception par variantes.
- La conception par assemblage.
- La conception à partir de rien.

I.5.2.1 La conception par variantes

L'investissement est réduit car elle correspond à modifier un existant [Sho95]. Il s'agit là généralement d'une simple modification de valeurs de paramètres qui a une faible incidence sur le processus de conception du produit. Ainsi le changement de la classe de tôle utilisée pour la fabrication d'un moteur électrique n'a pas une grande incidence sur le « comment réaliser le produit ?» si les marges de tolérance imposée par la 1ère version restent acceptables.

I.5.2.2 La conception par assemblage ou composition

Cette conception s'apparente au jeu LEGO [Big94]. Elle consiste à associer des sous-ensembles bien connus pour la construction d'un ensemble cohérent, telle que l'assemblage d'un rotor, d'un stator et du bobinage pour la construction d'une Machine ASynchrone (MAS).

I.5.2.3 La conception à partir de rien

Elle consiste à proposer un nouveau produit, c'est à dire à entamer un nouveau cycle de vie du produit émanant conception [Dej95]. D'où tout le travail est à faire pour définir le mode de réalisation du produit.

I.5.3 Positionnement de la conception intégrée dans la matrice de typologie

Notre premier objectif est d'apporter une aide aux concepteurs afin de réduire leur temps de réponse aux cahiers de charges. Nous pensons que la réduction du temps de réponse n'est garantie que lorsque la connaissance liée au produit de conception, ainsi que celle du processus de conception sont maîtrisées ; c'est pourquoi nous ne supporterons pas le concepteur dans une activité de conception créatrice.

D'autre part, la réflexion sur l'intégration des outils n'est légitime que si les outils utilisés pour la spécification du produit sont connus. Or dans le cas où le concepteur entame la conception d'un nouveau produit, il est dans une phase de recherches des outils adéquats : cette phase est donc en amont par rapport à la problématique de la conception intégrée. C'est pourquoi la conception intégrée ne couvre pas cette zone de la matrice de typologie (cf. Fig. I-3).

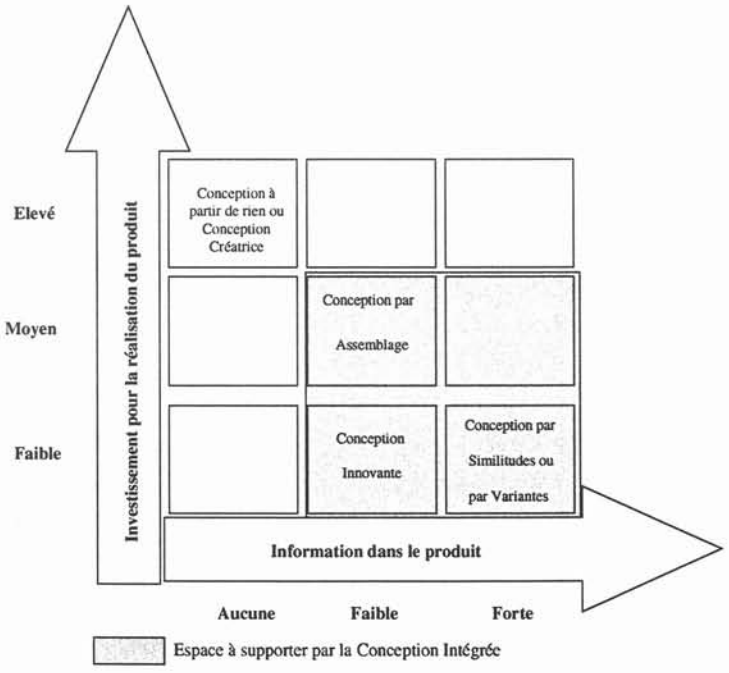


Figure I-3 Positionnement de la conception intégrée dans la matrice de typologie de la Conception

I.6 Les différentes phases de la conception

Avant d'énumérer les différentes étapes de l'activité de conception industrielle, il est nécessaire de rappeler que tout projet de conception est fondé sur trois appuis : les performances à atteindre, les éléments constitutifs du produit à concevoir et les relations qui définissent la sensibilité des performances du système par rapport à ses éléments. Ces relations ne sont pas forcément formulées sous forme matricielle de systèmes d'équations analytiques ; elles peuvent être fournies par des règles de production, des tableaux de données, des abaques ou autres formes de graphes... Il est important de cadrer les problèmes à résoudre lors de l'exercice de la conception.

I.6.1 Les catégories de problèmes à résoudre

Selon les informations disponibles au début de la réalisation d'un produit, trois catégories disjointes de problèmes [Lat72], [Wur96] sont à distinguer :

Les problèmes de la 1ère catégorie appelés ***problèmes inverses :***

Ce sont les problèmes dont les performances à atteindre ont été exprimées par un client X ; les concepteurs des bureaux d'études (industriels ou chercheurs) sont alors amenés à trouver une solution à ce besoin en définissant les éléments constitutifs d'un système potentiel et à vérifier la validité de leurs solutions en analysant selon leur savoir et savoir-faire les éventuels liens entre les constituants et les performances souhaitées.

Les problèmes de la 2ème catégorie appelés les ***problèmes directs :***

Cette catégorie renferme les problèmes dont les éléments constitutifs ont été répertoriés, et tels que toutes les relations qui définissent les interactions entre les performances à atteindre et les caractéristiques des constituants sont maîtrisées. Il s'agit là de définir les limites du système global en précisant les performances qu'il pourra remplir.

Les problèmes de la 3ème catégorie :

Dans ce cas, les constituants du système sont connus, ainsi que les performances atteintes, mais les concepteurs ignorent les liens qui définissent les interactions entre les performances et les caractéristiques des constituants. L'activité chimique est typique de cette catégorie de conception.

Par exemple, en réalisant un mélange de produits chimiques, le scientifique dispose de toutes les informations sur les éléments constitutifs de son produit final ; de même, il a ses notes sur les observations du comportement de son mélange. Toutefois, il lui reste à définir la sensibilité des performances de son mélange par rapport à ses constituants, c'est à dire à formaliser l'ensemble des relations empiriques issues de ses multiples essais.

Dans les deux premières catégories les lois qui lient les caractéristiques du produit à ses performances sont bien définies alors que dans la dernière catégorie elles sont mal définies et restent à expliciter.

Dans le domaine du génie électrique, concevoir des produits électriques consiste essentiellement à résoudre des problèmes inverses. Les connaissances sur le produit sont généralement bien maîtrisées. Une fois que le produit est spécifié, le concepteur cherche à valider la solution envisagée : il reformule ses données afin de résoudre un problème direct.

I.6.2 Les étapes d'un projet de conception en génie électrique :

Avant de voir le produit naître, trois grandes étapes sont à franchir :

- L'expression des besoins.
- La traduction du cahier des charges.
- La conception préliminaire.

Expression des besoins :

Il s'agit là d'établir le cahier de charges que nous qualifierons plutôt de Cahier de Charges Brut tel qu'il est fourni par le client aux bureaux d'études : il est généralement non exploitable directement. Dans cette étape, il s'agit de définir les grandes caractéristiques (tension et courant nominal pour un moteur, puissance nominale...) et les performances recherchées du produit à concevoir (limites de fonctionnement, courant de démarrage admissible...). Il existe une tâche intermédiaire entre la réception de ce Cahier des Charges Brut et la mise en route de la conception préliminaire qui n'est autre que la traduction du cahier des charges.

La traduction du cahier des charges :

Il s'agit d'extraire du Cahier des Charges Brut les données nécessaires au traitement du problème inverse. En effet, le concepteur a besoin de traduire la 1ère version du cahier des charges en un Cahier des Charges Exploitable adapté à son approche de la conception et particulièrement à sa méthodologie. Le concepteur est amené à distinguer les spécifications quantitatives et qualitatives. Le résultat de l'activité de conception peut être défini comme l'évaluation des spécifications quantitatives en prenant en compte les contraintes des spécifications qualitatives.

Par exemple en Génie Electrique, le concepteur est amené à faire le tri de l'information fournie dans le Cahier des Charges brut et à la structurer (choix des paramètres d'entrées dans le cas d'utilisation de solveur, organisation des données dans les différents champs correspondants…). Le concepteur extrait du Cahier des Charges Brut les valeurs d'initialisation de son raisonnement (les performances électriques souhaitées, ses zones de fonctionnement, les conditions d'alimentation etc...). Il s'agit du Cahier des Charges Exploitable.

Une fois le problème de conception explicité, le travail propre à la conception peut commencer ; il s'agit d'une activité de conception préliminaire.

Conception préliminaire :

L'activité de conception préliminaire correspond à la recherche de la ou les solutions optimales d'un problème de conception. Le schéma de la Fig. I-4 est représentatif de cette phase.

Pour chaque discipline connexe au produit, le cheminement du cahier des charges brut à la conception préliminaire passe à travers les étapes citées ci-dessus. Une synthèse des solutions obtenues par les différents acteurs sera effectuée au niveau de la plate forme du projet.

Comme le montre la figure Fig. I-4, cette activité est marquée par les boucles de retour. Durant toute la recherche des spécifications du produit à réaliser, le concepteur doit avoir la possibilité de réviser ses décisions, de modifier une ou plusieurs caractéristiques du produit, d'affiner certaines hypothèses, de relâcher des contraintes…

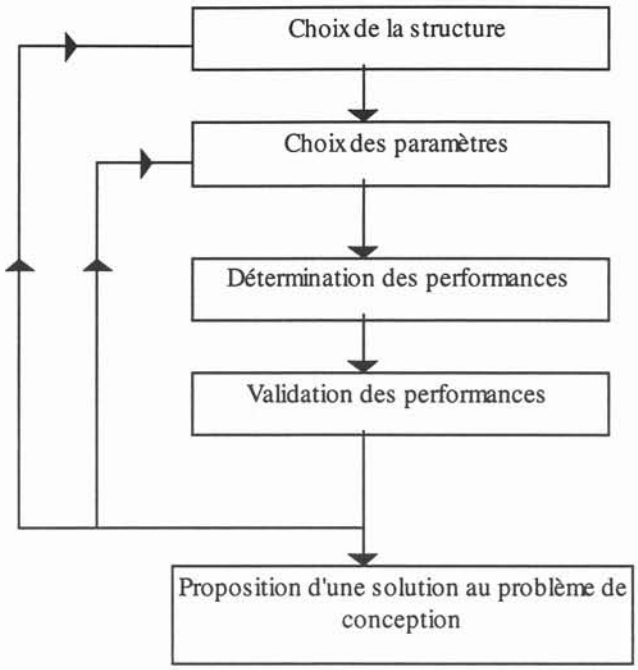


Figure I-4 Les étapes de la conception préliminaire

Dans le schéma présenté ci-dessus, l'étape de choix des paramètres inclut le choix des paramètres techniques et non techniques tels que les paramètres économiques (coût de production), les paramètres stratégiques (politique interne ou externe de l'entreprise ou du bureau de conception) ou encore les paramètres organisationnels (organisation du stockage...). Comme nous l'avons expliqué (cf. le paragraphe sur l'évolution de la conception), un produit est par définition multi-disciplinaire. Il faut donc prendre en compte les paramètres non techniques lors de la conception préliminaire.

Face à un problème de conception d'un moteur électrique, le choix de la structure ne se limite pas au choix de la classe (une Machine ASynchrone, Machine à Courant Continu, Machine Synchrone ou encore une Machine à Réluctance Variable). Mais il s'agit de définir les spécifications de la machine retenue. Pour cela le choix est fortement guidé par les grandes lignes du cahier des charges (domaine d'application, type de branchement, nature du réseau disponible...). Par exemple, une MCC (Machine à Courant Continu) est à exclure des solutions à examiner du moment où le client souhaite réduire au maximum la maintenance de son équipement (à cause des problèmes de collecteur).

Le concepteur de machines a recourt à des outils de conception métiers pour l'aider tout au long de la conception préliminaire. Toutefois, la prise en compte des paramètres non techniques reste difficile dans les outils.

Ainsi, lors de la conception préliminaire d'un produit électrique, un ingénieur électricien peut intégrer les contraintes de coût (choix des matériaux à utiliser, ...). Cependant il est ambitieux de penser qu'il puisse intégrer les paramètres organisationnels dans son outil : ces problèmes sont difficilement modélisables.

Dans le cadre de nos travaux, nous intervenons après l'expression des besoins et la traduction du cahier des charges brut en cahier des charges exploitable. Notre objectif est d'assister le concepteur dans la phase de conception préliminaire, plus exactement nous allons l'assister dans la spécification de son produit une fois que sa structure est choisie.

I.7 Notre pratique de la conception

Nous souhaitons mettre à la disposition des concepteurs en génie électrique un support informatique pour les assister dans l'exercice de leur activité. Nous pensons qu'un tel outil ne peut être efficace que si un minium d'information est disponible pour décrire le produit à réaliser : ses caractéristiques principales doivent être préétablies.

Dans le cadre d'une conception créatrice, non seulement les caractéristiques du produit ne sont pas encore bien définies (c'est à dire le savoir des concepteurs est en cours d'élaboration) mais en plus les aléas du processus de conception restent à maîtriser (c'est à dire le savoir-faire reste à acquérir). L'évaluation technique du nouveau produit est à faire. C'est pourquoi, nous allons focaliser nos efforts autour de la conception routinière et de la conception innovante.

Avant de définir notre contribution au développement d'outils dédiés au génie électrique, nous avons cherché à recenser les besoins en la matière. Dans ce but, nous avons réalisé un état de l'art des outils existants qui fera l'objet du chapitre suivant.

Dans ce deuxième chapitre, nous allons détailler les modèles de produit et de processus de conception implantés dans les outils. Nous allons proposer une classification des outils existants en fonction des niveaux de modélisation. A l'issue de ce bilan, nous allons montrer l'intérêt de mettre en œuvre le concept de conception intégrée.

Etat de l'Art des Outils de Conception en Génie Electrique

Sommaire du deuxième chapitre :

II Etat de l'art de la Conception en génie électrique

Introduction

Après avoir été sensibilisé à la complexité de l'activité de la conception, la deuxième étape dans notre travail consiste à comprendre la formalisation de la Conception. En parlant d'outil de conception, nous sommes amenés à formaliser les connaissances couvrant le savoir et le savoir-faire du domaine d'activité. Ce formalisme se traduit par la construction de modèles. En fonction du niveau d'abstraction recherché et des concepts adaptés, un ou plusieurs modèles sont associés à l'activité de conception automatisée : cette automatisation peut être complète ou partielle.

Un certain nombre des problèmes liés à la conception sont exposés dans le premier paragraphe. L'intérêt du modèle produit et du modèle du processus sont discutés dans le second. Dans le troisième paragraphe, nous avons dégagé une classification des outils de Conception en trois familles d'outils : les outils spécifiques, les outils génériques et les outils méta-génériques. Enfin, nous conclurons en positionnant notre contribution dans cette hiérarchie des outils de Conception.

II.1 Problèmes liés à la conception

Avant de synthétiser les modèles de l'activité de Conception rencontrés dans la littérature, il est utile de soulever les problèmes liés à l'activité de conception à prendre en compte dans la modélisation ; ceux que nous citons ci-dessous sont liés à la conception technique en général et au génie électrique en particulier.

II.1.1 Complexité du produit à concevoir

Le produit industriel et particulièrement les produits du génie électrique sont composites : un produit peut être lui-même un constituant d'un autre composant industriel. Même quand ils ne sont pas des constituants, les produits électriques restent complexes.

En effet, ils sont généralement répertoriés en deux grandes familles, les éléments statiques et les éléments dynamiques mécaniquement : les premiers sont alors le siège de phénomènes électromagnétiques alors que le mouvement des seconds met en jeu des phénomènes électromécaniques.

Pour s'affranchir de cette complexité les concepteurs adoptent le principe de décomposition en définissant des sous-problèmes (principe détaillé dans le chapitre précédent) ou encore en décomposant le sujet de la conception en plusieurs objets [Dej95]....

II.1.2 Extension de l'activité

L'activité de conception est rarement limitée à un seul domaine scientifique. De par la nature du produit à concevoir, le domaine spécifique de son application est associé à plusieurs domaines adjacents répertoriés en fonction des différentes contraintes liées à son utilisation.

Par exemple, une machine électrique tournante doit répondre à des critères électriques et magnétiques et en plus à des critères thermiques et mécaniques. Selon les performances exigées dans le cahier des charges, il faut dimensionner la section des conducteurs pour une bonne évacuation de la chaleur.

D'autre part, la conception industrielle est amenée à s'élargir aux domaines adjacents à la technologie. Ainsi le dimensionnement des paramètres techniques n'est pas fait uniquement en vue des performances technologiques. Il s'agit plutôt de trouver le meilleur compromis entre les paramètres techniques et les paramètres non techniques. Ces derniers sont répertoriés en quatre classes de paramètres : paramètres économiques, sociologiques, organisationnels et stratégiques. Un modèle complet de la conception devrait intégrer une combinaison de tous ces paramètres [Dej95].

D'où la nécessité d'étendre la base de connaissance des outils d'aide à la conception afin d'intégrer tous les éléments décisionnels et réussir à économiser sur le coût global dès les premières phases de la vie du produit.

Lors de l'exercice de la conception, des experts de diverses disciplines auront à collaborer ensemble : il faut structurer une gestion de cette multi-expertise. En effet, les différents agents ou experts ont souvent des méthodes différentes de calculer les paramètres du produit [Tri91] à concevoir, ou encore ils ont différentes approches possibles pour un même produit [Els97] (notion de point de vue détaillé ultérieurement dans le développement du modèle produit).

II.1.3 Contraintes à respecter

Les limites technologiques figurent parmi les freins à la conception. En outre, **l'aspect industriel** de la conception lui impose des contraintes bien particulières. Ces contraintes couvrent à la fois les obligations du fabriquant vis à vis du client industriel ou du bureau d'études (le produit conçu doit répondre le plus possible aux attentes du client en terme de rapport qualité-prix). Les solutions fiables sont à trouver dans les meilleurs délais.

Les contraintes liées à l'environnement sont actuellement en tête des exigences des clients aussi bien à court terme (le respect des seuils de nuisance auditive lors du fonctionnement des machines, ...) qu'à long terme (la nécessité de prévoir le devenir du produit à sa mort en envisageant la possibilité de le démonter, ...). Cette dernière éventualité serait à prendre en compte dès les premières phases de conception. Dans le but de détruire d'une manière intelligente le produit à sa mort, les concepteurs opteraient pour une conception du type Mécano/LEGO (cf. le premier chapitre). Ainsi en définissant des liens bien localisés entre les sous ensembles, le système peut être démonté. Si le concepteur mécanicien a le choix entre un rivetage ou une soudure il pourrait opter pour les rivets en vue de la facilité de rupture du lien.

Très souvent, des closes complémentaires, de natures diverses, sont formulées par le client dans le cahier de charges brut (souci esthétique, ergonomie, ...). En fonction du produit à concevoir, ces données complémentaires peuvent être contraignantes ou complètement négligées lors de la spécification technique. Par exemple, l'esthétique est un élément majeur dans la construction d'automobiles : elle figure parmi les clauses importantes de la spécification du véhicule.

II.1.4 Problèmes liés à la connaissance

II.1.4.1 Représentation

Les concepteurs sont amenés à manipuler des informations «hybrides». Les informations techniques peuvent avoir différents formats : abaques, systèmes d'équations, tableaux de mesures, formules empiriques... Cette formalisation diffère des représentations des données économiques (prix des éventuels concurrents, grilles des salaires, politique d'investissement...) ou encore des données organisationnelles (outils existants au sein de l'entreprise).

D'autre part, la pluridisciplinarité (évoquée plus haut dans ce chapitre) met en jeu le problème du langage de communication. Une fois que la nécessité de dialoguer a été démontrée, les experts ont du confronter le problème des divergences des termes. Bien qu'ils parlent tous la même langue, la définition de certains termes techniques varie du jargon d'un mécanicien, par rapport à celui de l'électricien ou celui du fabricant...

Pour pallier ce problème, il faut prévoir une formalisation des échanges [Sho95], [Els97]. Il est alors nécessaire de structurer le dialogue. Une solution pourrait consister à incorporer un test de langue dans un outil dédié à la conception intégrée [Kle93].

En effet, un produit quel qu'il soit, «n'est pas le fait d'un seul agent mais se fait dans l'interaction (c'est l'entre deux qui compte)» [Pol96]. Le besoin de formaliser cette interaction est réel ; D'où est né l'idée de création de symboles inter-métiers.

Il s'agit de créer des entités qui permettent de supporter les informations utiles à la communication entre les différents acteurs de métiers. "Ces entités concernent donc des points particuliers et locaux qui font l'objet de règles temporaires et locales portées par des acteurs distincts" [Lau00]. Dans un processus de conception, d'une part, elles constituent le canal d'échanges des données et des résultats partagés. D'autre part elles témoignent des compromis entre les différents acteurs.

Suite à un constat de difficultés de dialogue entre les deux acteurs qui interviennent dans la construction d'une fusée (une pièce mécanique utilisée dans la construction d'automobiles), le concept d' *"Entités de Coopération"* fût proposé. Des entités ont été introduites afin de formaliser la coopération entre les acteurs [Lau00].

En pratique, les usineurs ont été confrontés à des difficultés d'obtention de la pièce à partir du brut forgé. L'enrichissement du modèle volumique de la pièce mécanique par le rajout de symboles fut envisagé pour supporter l'échange des données utiles entre les usineurs et les forgerons (comme le montre la figure ci-dessous).

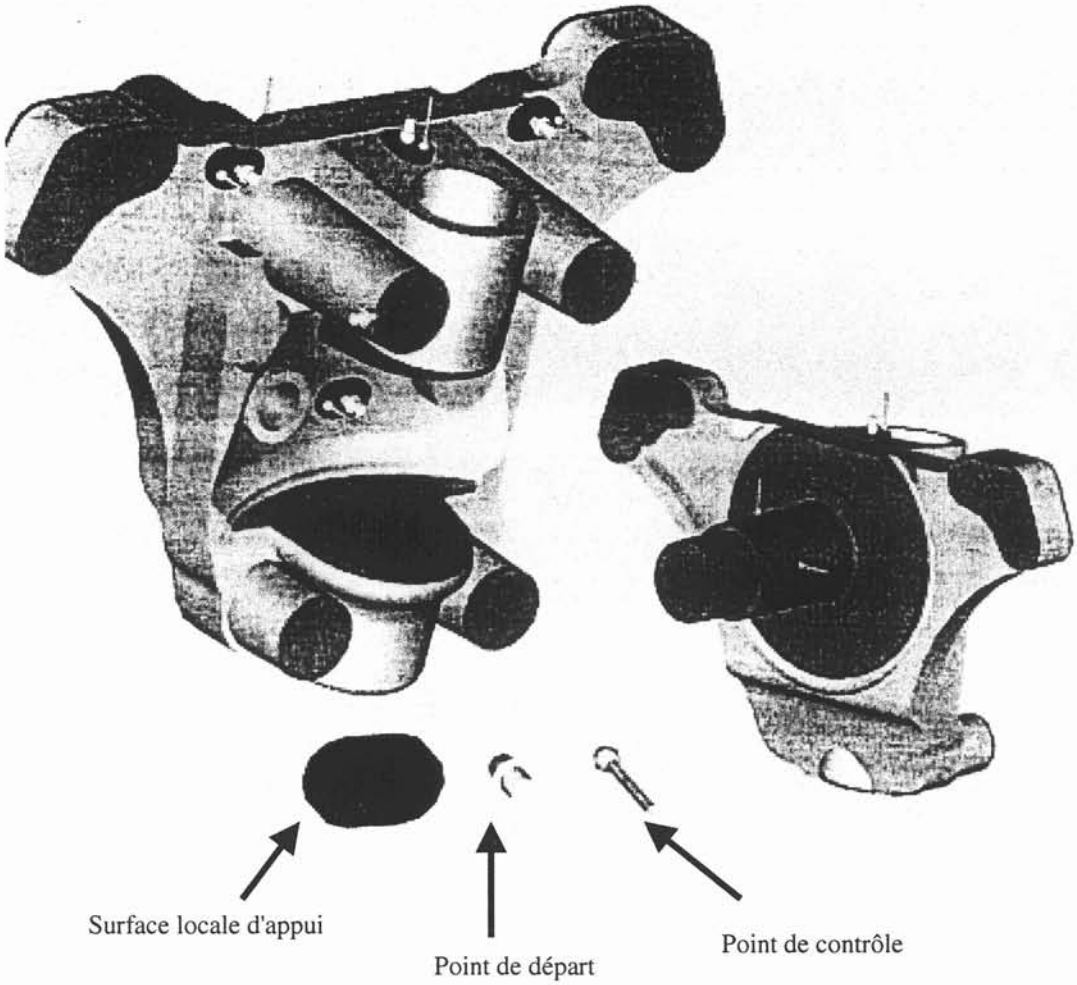


Figure II-1Mise en œuvre des entités de Conception (Fig. extraite de [Lau00])

II.1.4.2 Actualisation

L'outil doit permettre d'intégrer toute évolution technologique (ou autre dans le cas d'un outil intégrant l'aspect non technique). Il ne suffit pas que la mise à jour soit possible, il est surtout essentiel qu'elle ne soit pas trop longue à reprendre [Tri91], [Wur96]. Avec la répétabilité, elle doit offrir la qualité. Certains outils sont performants et évolutifs mais ils reposent sur une méthodologie peu explicite à l'utilisateur non informaticien. Très souvent, la maîtrise de l'outil est nécessaire pour le mettre à jour.

II.1.4.3 Disponibilité du savoir-faire

Afin de développer un outil efficace de Conception d'une catégorie de produit, il faut faire un repérage fin des connaissances de base des concepteurs de ce produit. La connaissance dans une activité de conception n'est pas limitée à la théorie.

En effet, la source la plus valorisante réside plutôt dans la façon d'exploiter les retours d'expérience et l'apprentissage du déjà fait : cet aspect pratique du savoir communément appelé le savoir-faire n'est pratiquement pas protégé. Ce sont généralement les compétences anciennes dans les entreprises ou les bureaux d'études qui maintiennent ce savoir-faire. Leurs départs pour une raison quelconque (retraite, décès, indisponibilité...) induisent des pertes des compétences en la matière. D'où la capitalisation de la connaissance dans le but d'une réutilisation est un souci majeur.

II.1.4.4 Incomplétude des données du cahier de charges

Comme nous l'avons déjà mentionné le cahier des charges brut tel qu'il est fourni par le client n'est pas directement exploitable. Sa traduction en un cahier des charges technique constitue un problème majeur de la conception : les données fournies par le client sont souvent incomplètes et incertaines. Cette situation pourrait être favorable pour le concepteur puisqu'il dispose de degrés de libertés supplémentaires et que les contraintes mises en jeu sont moins rigides. La réalisation du produit global serait forcément plus facile. Cependant, le risque d'échec de la conception grandit d'autant plus qu'un voile d'imprécision et d'incertitude porte sur les spécificités du produit à concevoir.

Notre travail n'inclut pas ce volet ; nous partons du principe que nous disposons pour la conduite de notre résolution des problèmes de conception d'un cahier des charges le plus complet possible et validé par le client.

II.2 Les modèles adaptés à la conception

L'amalgame dans la représentation du produit et celle du processus de conception est courant parmi les concepteurs. Dans la majorité des outils de Conception, l'identité du produit à réaliser est noyée dans le processus suivi pour la résolution du problème.

En effet, le modèle du produit résultat de l'activité n'est pas explicité indépendamment du processus de conception : la description du produit se retrouve implicitement à travers le modèle du processus de conception. Certains concepteurs ont ressenti l'utilité et le besoin de séparer le modèle produit du cheminement de résolution du problème.

Ils ont opté pour deux modèles disjoints pour modéliser l'activité de la conception : un modèle renfermant les informations liées à l'objet de l'activité et le second réservé à formaliser l'exercice intellectuel pour la spécification de produit à réaliser [Har97].

II.2.1 Les modèles du produit

Le modèle de produit c'est la formalisation du produit à concevoir. Afin que le modèle soit qualifié de modèle «Bon », il faut qu'il soit représentatif du comportement réel du produit. En effet, le modèle sert à définir les caractéristiques du produit à concevoir. Selon le degré de finesse souhaité, le modèle renferme des informations plus au moins détaillées sur le produit.

L'utilisation d'un outil lors de la conception a pour objectif même la spécification des éléments du modèle. Les informations apportées par l'outil sont exploitées en vue d'affiner ou non le (ou les) modèle(s).

Ainsi, le modèle du produit à concevoir évolue en cours d'avancement de la conception : (1) au départ le modèle du produit est établi à partir des informations extraites du cahier des charges/ (2) dès que le processus de conception est entamé, la première version du produit peut être remplacée par une ou plusieurs versions intermédiaires.

II.2.1.1 La ou les versions intermédiaires

Le modèle initial du produit à concevoir est incomplet et ne renferme pas toutes les spécifications du produit à l'exception des cas de réutilisation de solutions archétypes déjà réalisées comme point de départ. Tout au long de la conception les spécifications du produit sont estimées, calculées, vérifiées et validées.

Une ou plusieurs modifications peuvent être apportées à ces spécifications à chaque transition dans le processus de conception (activation d'une règle, affectation d'un résultat de calcul,...). Un outil de conception peut présenter selon l'architecture du processus de conception une ou plusieurs versions intermédiaires.

En effet, nous distinguons deux modes d'avancement dans la conception comme le montre le schéma de cheminement ci-dessous :

- Le mode linéaire : l'outil ne traite qu'une seule version du modèle à un instant donné. Le raisonnement est conduit d'une manière séquentielle ; une seule ébauche de solution est menée jusqu'au bout. Des tests de vérification sont implantés à différents stades de l'avancement ; ainsi en cas d'échec, les boucles de retour permettent de revoir les hypothèses de départ.

- Le mode en parallèle : l'outil est capable de gérer le suivi de plusieurs versions distinctes du modèle simultanément. Dans le cadre de la conception parallèle, le concepteur dispose d'un arbre de résolution [Fez97]. Cette méthode de conception peut aboutir à un ensemble de solutions. Le succès de la résolution d'un tel problème est dans la convergence des différents états temporaires vers un espace fini de solutions.

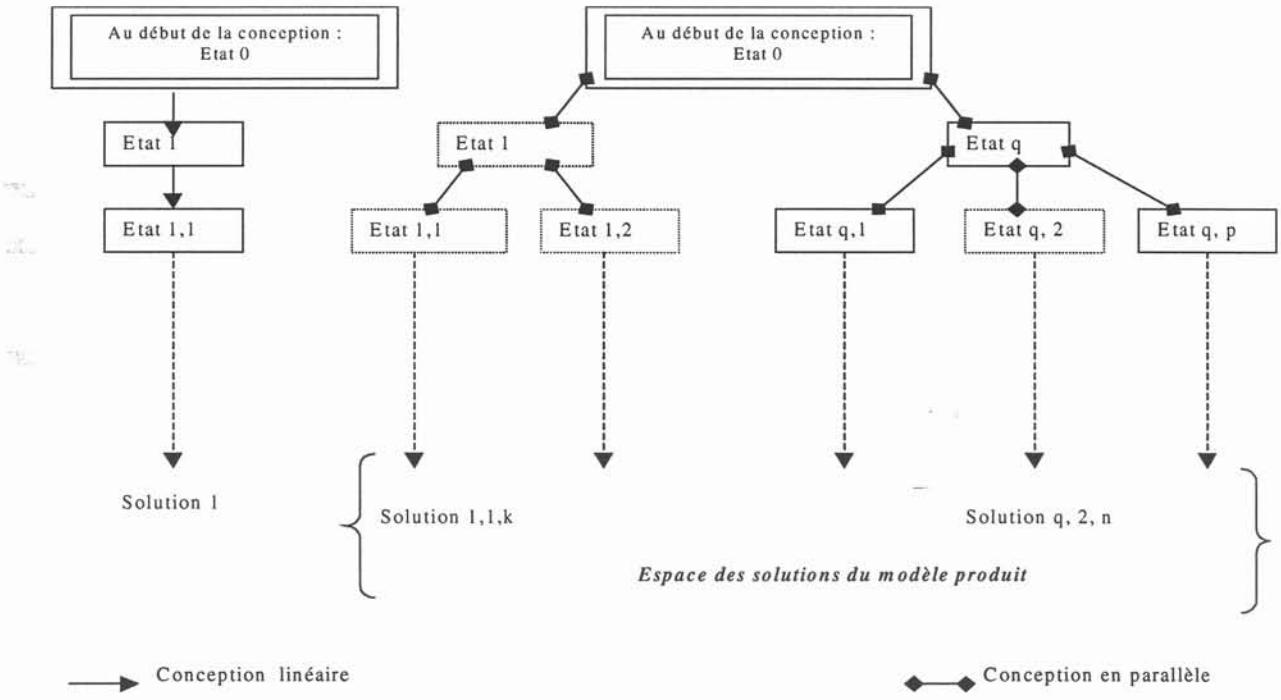


Figure II-2 Schéma du cheminement d'un modèle de produit

II.2.1.2 Les points de vues

Le concept de la vue intrinsèque à l'objet lui-même, dégagé par un même observateur, est réfuté. Le produit était décrit d'un point de vue unique. Généralement la description du produit découlait de sa décomposition fonctionnelle ou structurelle. Ainsi le produit est défini uniquement en fonction des contraintes particulières à un domaine spécifique (à un domaine X) : il s'agit du «**Design for X**» (cf. Glossaire), concept introduit par les mécaniciens [Els97].

Les concepteurs mécaniciens [Els97], [Cao98] ont réfléchi à une structure de la modélisation adéquate pour une conception pluridisciplinaire. C'est alors qu'ils proposent une description du produit selon ses différents angles de vues métiers. L'image du produit vue par chaque acteur intervenant dans sa conception est à modéliser.

Nous illustrons sur le schéma ci dessous le concept point de vue sur l'exemple d'un dispositif électrique simple [Har97] : une MCC (Machine à Courant Continu) alimentée par une source de tension continue. Ainsi, trois points de vues distincts peuvent être dégagés pour une description complète de ce dispositif :

- point de vue structurel : établi pour définir la structure du système électrique.
- point de vue électrique : correspond tout simplement au schéma du circuit électrique.
- point de vue fonctionnel : définit les fonctions assurées par les différents éléments du système.

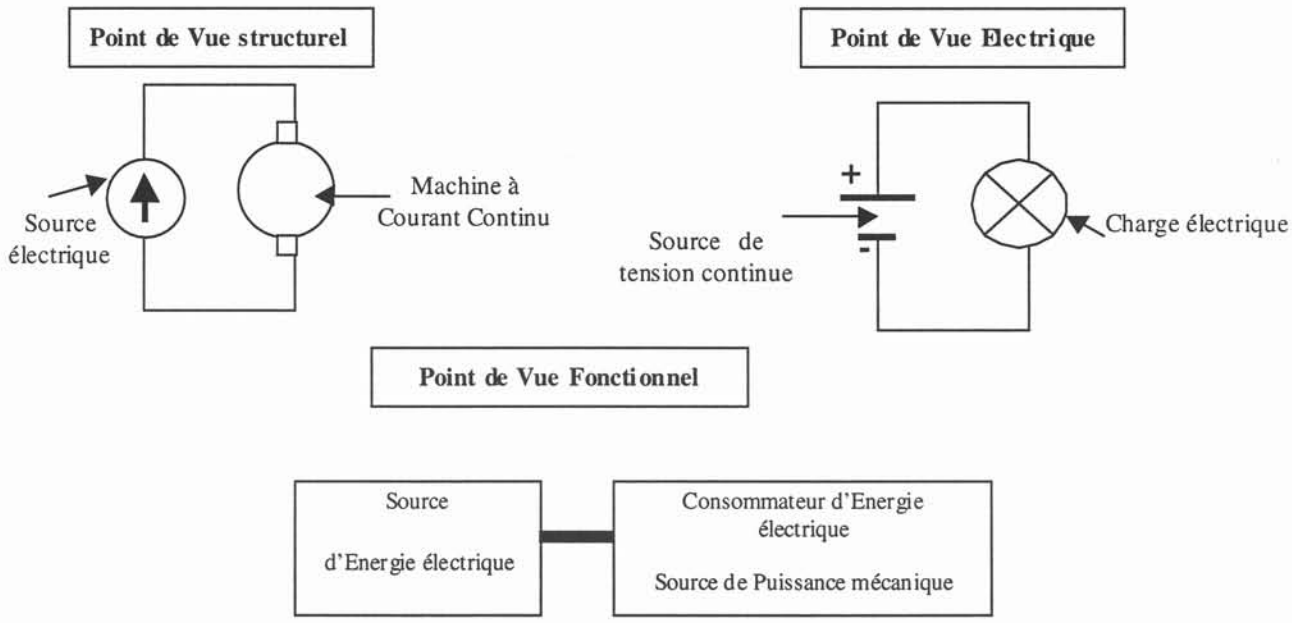


Figure II-3 Illustration du concept de point de Vue

II.2.1.3 Positionnement par rapport au modèle produit

Nous avons constaté ([Tri91], [Wur96], [Fez96]) que la modélisation la plus répandue dans les approches dédiées à la conception en génie électrique est le modèle du "paramètre" du produit à concevoir. Rappelons que dans leurs débuts les outils avaient pour mission principale de se substituer aux calculatrices afin d'effectuer des calculs de plus en plus imbriqués et compliqués à résoudre.

L'approche «paramètre » était le support adéquat à la vision qu'avaient les concepteurs de leur objet de conception : leur problème était limité au dimensionnement des paramètres caractéristiques du produit. Seules les informations techniques étaient intégrées au modèle.

Cependant, comme nous l'avons souligné, les besoins des concepteurs ont évolué. Une demande d'intégration des informations non techniques (organisationnelles, économiques, stratégiques...) s'est fait ressentir. Les experts en génie industriel l'ont présentée comme une condition nécessaire à une conception optimisée en terme de délais et coûts [GI97]. Cette extension de l'identité du produit à concevoir met en cause les limites de la modélisation paramétrée. Le modèle du paramètre est-il adapté à cette projection de l'activité de conception du domaine particulier vers le «multi-domaines» ?

D'autre part en intégrant cette dimension multi-domaines du produit, peut-on encore affirmer l'indépendance du modèle produit vis à vis de son domaine ? Peut-on construire un modèle complet interdisciplinaire et indépendant du domaine direct ou associé de l'activité de conception ?

Nous pensons que l'interdisciplinarité du modèle produit n'engendre pas forcément un modèle particulier à l'activité exercée. Il s'agit de réussir la mise œuvre du compromis entre le générique et le particulier. En effet, la formalisation du modèle paramétré peut supporter la pluridisciplinarité de l'information et décrire les différents aspects (techniques et non techniques) du produit à concevoir.

Du moment que les experts sont capables de formaliser ces connaissances et de quantifier ces paramètres non techniques, il est alors possible de définir une notion de carte d'identité standard (cf. Fig.II-4).

Produit X

Paramètres techniques	*Paramètres économiques*	*Paramètres organisationnels*
Pt1 Pt2 . . Ptn	Pe1 Pe2 . Peq	Po1 Po2 . Pom

Figure II-4 Carte d'identité d'un produit X

Le modèle paramétré reste le plus utilisé par les concepteurs en génie électrique puisqu'il suffit pour supporter la description de la conception dans un problème de dimensionnement. Dés lors, la restriction de la modélisation au modèle paramétré est une simplification de la description du produit afin de faciliter la mise en œuvre d'outils d'aide à sa Conception.

Actuellement, les besoins de modélisation sont orientés vers la modélisation du processus.

II.2.2 Les modèles du processus de conception

Le ou les modèles du processus de conception consistent à décrire l'activité de conception et non l'objet résultat de cette activité. Il s'agit de formaliser toutes les informations liées à l'exercice mental de réflexion pour la spécification du produit. Les détails de cet exercice varient d'un concepteur à un autre.

En effet, pour une même activité il peut y avoir plusieurs versions du processus, autant de versions que de manières de faire le produit. L'activité de conception ne relevant pas de l'ordre des sciences exactes ; nous avons à modéliser une activité mettant en jeu des objets empiriques [Leb97].

Le ou les modèles du processus de conception cherchent à :

1- Expliciter l'organisation des tâches : que doit-on faire ?
2- Exploiter les ressources : comment exploiter ce qui est connu ?
3- Suivre les versions intermédiaires : comment justifier les alternatives de solutions ?
4- Capitaliser le savoir-faire : comment rentabiliser ce qui est déjà fait ?

II.2.2.1 Expliciter l'organisation des tâches

L'activité de conception étant une activité compliquée, les concepteurs adoptent généralement le principe de décomposition du problème à résoudre en sous problèmes. Ils associent à chacun de ces sous-problèmes une ou plusieurs tâches à effectuer.

Les opérations à effectuer au niveau des tâches sont variables ; elles peuvent consister à :

- Adopter une hypothèse : faire une estimation de l'évaluation de certaines spécifications du produit, adopter des valeurs par défaut avant de se lancer dans des calculs plus coûteux,...
- Appliquer une méthode de calcul.

- Vérifier la cohérence des opérations déjà effectuées
- Activer des règles dans le cas d'un processus heuristique.
- ...

Le processus décrit ci-dessus est un processus de conception structuré : il s'agit d'un enchaînement de tâches. Mais ce cas de figure est particulier, car les tâches ne sont pas toujours aussi explicites dans l'esprit des concepteurs. Ces derniers ne détiennent pas toujours la réponse exacte à «que doit-on faire ? ». Ils ne sont pas toujours capables d'établir dès le début de la conception l'organigramme des tâches et de structurer leur enchaînement.

En particulier, trois configurations de processus de conception se distinguent [Har97] :

- *processus structuré* : l'ensemble des tâches à exécuter ainsi que leur enchaînement est établi à l'avance.
- *processus semi-structuré* : l'ensemble des tâches à exécuter est connu mais leur enchaînement n'est pas forcément établi à l'avance.
- *processus non-structuré* : l'ensemble des tâches n'est pas complètement défini et leur enchaînement est à définir au cours de l'avancement de la résolution.

En focalisant notre étude sur une conception routinière, nous ne nous plaçons pas exclusivement dans le cas de processus structuré : la stratégie d'enchaînement du processus n'est pas forcément établie à l'avance. Par contre, nous excluons le processus structuré délibérément et complètement.

II.2.2.2 Exploiter les ressources

L'activité de conception n'étant pas une activité complètement empirique, elle est basée sur des connaissances théoriques ainsi que sur les connaissances pratiques acquises lors de son exercice par les corps de métiers qui y contribuent.

L'archivage des données en document texte (rapport d'activités avec plans détaillés des opérations) n'est pas le meilleur format pour une éventuelle réutilisation. De même, il ne suffit pas de disposer de bases de connaissances mais il s'agit de les exploiter efficacement lors de la conception afin d'optimiser les tâches à exécuter.

Il revient au concepteur ou à l'outil de Conception (dans le cadre d'une conception à facultés automatique cf. Glossaire) de savoir comment exploiter ce qui est connu : en cherchant à y confronter à certains niveaux d'avancement la ou les ébauches de solutions.

II.2.2.3 Suivre les versions intermédiaires

Le contrôle du raisonnement est l'une des composantes d'une démarche constructive de conception. L'examen des étapes franchies est nécessaire pour juger la cohérence de la ou des alternatives de solutions en cours de construction. Ce souci de rigueur est une condition nécessaire à la réussite de l'acte de conception.

Il s'agit donc de garder en permanence l'équivalent «d'un tableau de bord ». En naviguant dans l'espace ouvert qui est celui d'un problème de conception, il faut tenir à jour le tracé justifié du parcours de conception. Ce tracé justifié doit renfermer à la fois les étapes franchies et le justificatif de l'autorisation de franchissement (contraintes respectées ou ignorées, hypothèses faites, règles appliquées...).

II.2.2.4 Capitaliser le savoir-faire

Ce quatrième point à prendre en compte semble être facultatif ; il relève d'ordre organisationnel. Il s'agit de rentabiliser les expériences de l'exercice du métier de conception. Avec les débuts de la Conception, nous avons constaté que la majorité des concepteurs n'ont cherché à archiver que les résultats de la conception.

En effet, les opérations à mener pour une conception mono-disciplinaire et linéaire sont fortement répétitives. Il se trouve que la majorité des concepteurs n'a pas ressenti le besoin de sauvegarder leurs stratégies de conception ; ils ne gardaient que les caractéristiques des produits réalisés. En caricaturant, pour les gens de métiers après quelques années d'expérience ils arrivaient à mémoriser les grandes étapes de leurs activités. Ils pratiquaient intuitivement la conception par analogie en cherchant à appliquer leurs précédentes recettes de conception.

Le savoir-faire en conception était donc détenu par les gens du métier. En absence d'une organisation d'archivage efficace, avec leur départ, une grande partie des compétences des entreprises se perdent [Har97]. Réutiliser les expériences précédentes est parfois plus compliqué que d'entamer une nouvelle démarche de conception.

Avec l'évolution des concepts de l'activité de conception, une capitalisation de ce savoir-faire est devenue nécessaire. Nous tenons particulièrement à rappeler notre définition du mot capitalisation :

Capitalisation : c'est la construction d'un capital (cf. Glossaire). Il s'agit là de consolider le **capital** des concepteurs en matières de connaissances. En effet, les concepteurs cherchent à **accumuler** intelligemment les connaissances liées à la conception sans pourtant les **amasser**. L'objectif de la capitalisation est de dépasser le simple archivage des projets réalisés. En réutilisant les expériences antérieures, les concepteurs **économisent** du temps, améliorent leurs temps de réponses aux cahiers de charges et en conséquence les coûts de conception sont réduits.

II.2.2.5 Positionnement par rapport au modèle du processus de conception

Pour formaliser les quatre points détaillés ci-dessus (l'organisation des tâches, l'exploitation des ressources, le suivi des versions intermédiaires et la capitalisation du savoir-faire), les approches des concepteurs sont multiples. En particulier, nous avons constaté que la modélisation du processus de conception faite par les électrotechniciens s'appuie sur le concept de trilogie [Tri91], [Sho95], [Fez97] : les paramètres, les méthodes et la stratégie.

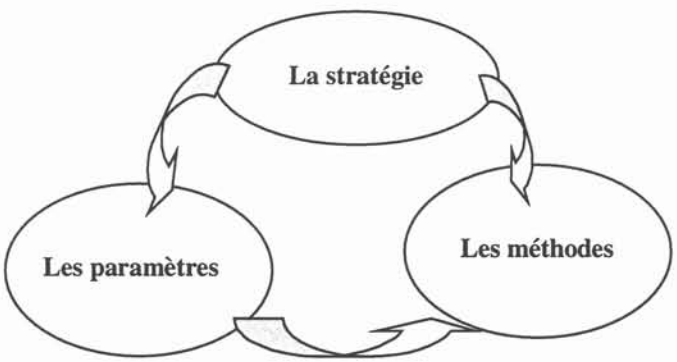


Figure II-5 Un exemple de la Trilogie du processus de conception

Tous les renseignements sur l'objet de la conception sont introduits dans le concept des paramètres. Sous la rubrique méthodes, toutes les sources de connaissances appliquées et exploitées pour la conception sont fournies. Quant au module stratégie, il permet d'organiser la gestion des méthodes, de préciser sur quels critères on s'appuie pour appliquer telle ou telle méthode à un stade donné de l'avancement du cheminement. Il répond ainsi à la problématique "que doit-on faire ? ".

Cette répartition tripartite des connaissances liées à la conception sera mise en place dans l'environnement de conception que nous allons proposer dans nos travaux.

II.3 Classification des outils de Conception

Dans les deux paragraphes précédents, les difficultés rencontrés lors de l'exercice de l'activité de conception ont été soulevées. Les niveaux de modélisation à intégrer dans les outils de Conception ont été développés. Dans ce paragraphe, nous avons établi une classification des outils de conception selon une approche conceptuelle. Nous ne les avons pas répertoriés selon les solutions techniques implantées pour leur réalisation mais en fonction des niveaux d'abstraction des modèles qu'ils intègrent.

Nous avons cherché à nous informer pour savoir ce qui a été fait par nos prédécesseurs en génie électrique et nous avons essayé d'élargir notre champ de recherche pour regarder ce qui s'est fait dans les autres disciplines (mécanique, informatique). Nous avons dégagé trois catégories d'outils dont nous allons développer les caractéristiques dans les paragraphes suivants :

- les outils spécifiques
- les outils génériques
- les outils méta-génériques

Avant tout, nous tenons à rappeler notre définition d'un outil de Conception. Nous appelons outil de Conception tout outil informatique qui apporte une assistance par ordinateur aux concepteurs pendant leurs recherches d'une solution répondant aux cahiers des charges.

II.3.1 Outils spécifiques

II.3.1.1 Définition des outils spécifiques

Ce sont des outils destinés à un usage mono-disciplinaire : ils couvrent généralement une compétence spécifique permettant ainsi de se focaliser sur l'étude d'un phénomène local et d'approfondir l'étude d'un problème déterminé. Un tel outil intègre des modèles fortement marqués par le produit à concevoir : généralement la connaissance à manipuler est incluse dans l'outil.

Ce choix de modélisation de problème spécifique est le plus répandu parmi les outils de Conception existant actuellement. Ainsi un outil spécifique apporte une aide ponctuelle au concepteur dans un domaine précis.

Nous citons l'exemple de l'outil Flux/2D [Flu2], outil de simulation. Le module électromagnétique de cet outil permet d'étudier les phénomènes électromagnétiques présents dans les produits électriques : répartition des lignes de champ, zones de saturation... Nous considérons Flux2/D comme un outil spécifique car l'aide qu'il apporte est restreinte aux applications électriques ou thermiques.

II.3.1.2 Limites des outils spécifiques

L'inconvénient majeur des outils spécifiques réside dans l'amalgame entre les deux modèles. Le modèle du produit à concevoir et celui du processus de la conception ne sont pas dissociés au niveau des outils spécifiques. D'autre part, il ne couvre qu'une tache très spécifique dans l'ensemble de la démarche de conception.

L'approche procédurale des problèmes de conception est un exemple de cette confusion. Cette approche consiste à structurer en des procédures la connaissance de l'activité dans un langage de programmation choisi. L'outil de Conception ainsi conçu présente un modèle unique combiné du processus et du produit de la conception : il permet de mener séquentiellement la construction du produit.

Ces outils ne sont donc adaptés qu'à la conception linéaire. Ils ne peuvent pas gérer plusieurs alternatives de solutions. Leur développement devient très rapidement compliqué dans le cas de produit nécessitant des compétences multiples.

Reprenons l'exemple de Flux2/D, cet outil est basé sur une programmation (en fortran) des équations aux dérivées partielles de l'électromagnétisme [Flu2]. L'outil aide le concepteur à analyser les phénomènes électromagnétiques locaux ayant lieu dans le produit étudié. Tout produit électrique peut être examiné sous Flux ; pour cela il suffit de le définir par ses paramètres géométriques, ses caractéristiques physiques et ses conditions limites. Il en découle que Flux2/D a un certain degré de généricité mais cette généricité est restreinte au domaine électrique c'est pourquoi nous le considérons dans notre approche comme un outil spécifique.

II.3.2 Outils génériques

II.3.2.1 Définition des outils génériques

Un outil générique dépend partiellement de la connaissance qu'il véhicule. Contrairement aux outils spécifiques, les outils génériques ne sont pas destinés à un phénomène bien spécifique. C'est dans le but de s'affranchir de cette restriction que les concepteurs d'outils ont cherché à promouvoir des outils généraux. L'objectif étant de développer un seul outil de Conception et dont la version unique permet de supporter des applications issues de différentes disciplines (mécanique, électrique...).

En contre partie, un outil générique exige une structure de modélisation puisqu'il impose le format du modèle du produit ainsi que celui du modèle de processus.

II.3.2.2 Exemples d'outils génériques utilisés en génie électrique

Nous distinguons particulièrement deux familles d'outils génériques : les outils s'appuyant sur les techniques d'optimisation et les systèmes experts.

II.3.2.2.1 Les outils s'appuyant sur les techniques d'optimisation

Les outils de Conception s'appuyant sur les techniques d'optimisation peuvent traiter des problèmes de natures diverses. L'outil est indépendant de ses applications. Il est d'autant plus performant que les algorithmes de résolution le sont.

Nous pouvons citer en particulier l'exemple de l'outil PASCOSMA (cf. [Wur96]) développé au LEG (Laboratoire d'Electrotechnique de Grenoble). Cet outil permet d'apporter une aide efficace en conception à condition de lui fournir la modélisation analytique du problème. Cet outil sera l'un des composants principaux de notre environnement de conception, c'est pourquoi nous consacrerons la première partie du chapitre V de ce manuscrit à l'exposé de ses limites et de ses apports.

Afin d'être résolu avec PASCOSMA, tout problème inverse (cf. définition introduite dans le premier chapitre) de dimensionnement doit être formulé sur le modèle du schéma ci dessous :

$$\begin{cases} F\ (p_1, p_2, \ldots, p_n) = 0 & ; F \text{ est la Fonction Objectif} \\ G_j\ (p_i) = 0\ \forall j \in \{1,.., m\} & ; (G_j) \text{ contraintes d'égalités des paramètres} \\ H_h(p_i) \leq 0\ \forall h \in \{1.., n\} & ; (H_i) \text{ contraintes d'inégalités des paramètres} \\ p_{imin} \leq p_i \leq p_{imax}\ \forall i \in \{1,..,n\} & \end{cases}$$

pi paramètre de sortie indice i
m : nombre de contraintes d'égalités
n : nombre de contraintes d'inégalités

Le modèle du processus de Conception implanté dans cet outil est l'algorithme d'optimisation. Quant au modèle du produit à concevoir, seul le modèle analytique est introduit.

II.3.2.2.2 Les systèmes experts

Les systèmes experts permettent de résoudre les problèmes de conception dont les connaissances sont heuristiques. En effet, toute la connaissance à traiter aussi bien celle qui se rapporte au produit qu'au processus de la conception doit pouvoir s'écrire sous la forme de règle de production :

SI (Condition(s) vraie(s)) **ALORS** (Action(s))

Les règles sont donc le formalisme adopté pour décrire aussi bien l'objet de la conception que son mécanisme. Le moteur d'inférence du système expert est mis en œuvre afin de gérer la coordination entre les différentes règles et contrôler les changements d'états. Trois types d'enchaînement des règles existent : le chaînage avant, le chaînage arrière et le chaînage mixte (qui n'est autre qu'une combinaison du chaînage avant et du chaînage arrière).

Par exemple dans le cas d'un moteur d'inférence à chaînage avant, tout au long de l'avancement du processus, le moteur tient à jour la liste des règles dont les prémisses (conditions) sont validées et ordonne l'exécution des actions correspondantes et ainsi de suite. Le processus avance à partir des faits connus afin de résoudre le problème.

Des travaux ont été menés pour développer des systèmes experts pour la conception des systèmes électriques ; parmi lesquels, nous citons :

- DAMOCLES [Tri91] : (**D**esign of **A**synchronous **Mo**tors **C**ontroled and **L**ed by **E**xpert **S**ystem) système expert d'aide à la conception de machines asynchrones.

- COCASE [Gen91] : système expert d'aide à la conception d'appareils électriques.
- OPUS [Esc96] : système expert pour l'aide à la conception en génie électrique.
- [Fez97] : Système expert pour la conception en Electronique de Puissance

Dans le système Expert développé par [Fez97], le processus de conception progresse selon le mode de chaînage avant. La connaissance a été répartie sur différents modules. A chaque composant du dispositif électrique correspond un module de connaissance où est stockée toute l'information heuristique le décrivant. Un module principal nommé "Module Expert" est chargé de :

- centraliser les informations issues des différents modules.
- répartir les données vers les modules pour faire avancer le processus de conception.
- notifier l'ordre d'exécution des modules.

La trilogie "Paramètre - Méthodes –Stratégie" (évoquée dans le paragraphe II-2-2-5) a été mise en œuvre dans la réalisation du système expert de [Fez96] puisque dans ce processus de conception seules les valeurs de paramètres sont échangées entre un module donné et le module expert. D'autre part, le choix d'une stratégie centrée autour d'un module maître est explicite dans cette réalisation : le module Expert gère les échanges. En ce qui concerne les méthodes, des fonctions ont été codées pour calculer les valeurs prises par les différents paramètres.

II.3.2.3 Apports des outils génériques

Le processus de conception peut être fortement indépendant du domaine du produit à réaliser. Il existe donc des invariants dans le processus de conception [Kle93] (mécanisme d'amorçage, trilogie de gestion des connaissances, enchaînement du raisonnement...). En dégageant ces invariants de la démarche de conception, les concepteurs ont cherché à disposer d'outils de Conception génériques. Le coût d'investissement global en de tels outils devrait être moins lourd qu'un outil très spécifique dont l'exploitation serait restreinte à une activité limitée en temps.

Par exemple, en se plaçant dans le cadre d'une conception routinière, la gamme des produits ne varie pas à court terme. Dans ce cas, l'inertie de la dynamique des produits à concevoir est élevée.

En conséquence, l'utilisation d'un outil de Conception spécifique semble être la mieux adaptée. Mais dans une politique à long terme, à l'arrivée d'une nouvelle gamme particulière de produits, les concepteurs seront rapidement amenés à renouveler l'outil alors que s'il disposait d'un outil générique il suffirait de lui apporter quelques modifications. Un compromis s'impose alors pour le choix de l'outil de Conception.

D'autre part, les modèles de produit et de processus intégrés dans un outil générique, ne sont pas rattachés à un problème spécifique. Ainsi l'outil n'est pas réduit à apporter une aide au concepteur pour la conception d'un produit électrique bien spécifique : par exemple, il peut supporter aussi bien la conception des convertisseurs statiques que celle de machines.

II.3.3 Outils méta-génériques

II.3.3.1 Définitions des outils méta-génériques

Certains développeurs d'outils proposent des outils d'un niveau d'abstraction au-dessus des outils génériques, d'où la convention de les qualifier d'outils méta-génériques. Il est important d'insister sur l'acte de méta-modélisation qui n'est autre que la modélisation de modèles.

En effet les modèles intégrés dans les outils méta-génériques «ne sont autres que des modèles de niveau de généricité plus élevé » [Har97] : l'outil n'intègre pas alors un modèle de produit mais un modèle du modèle de produit ; de même, son mécanisme de résolution des problèmes de conception n'est autre qu'un méta-modèle du processus de conception.

Il s'agit là de développer un contenant complètement indépendant du contenu. Les concepteurs d'outil méta-génériques mettent à la disposition des utilisateurs un contenant vide mais potentiellement prêt à supporter leurs démarches de conception et à les guider dans leurs recherches de solutions à leurs problèmes. Ils prennent à leur charge la mise en œuvre d'un outil efficace pour la gestion des connaissances pluridisciplinaires tout en veillant à ce que l'outil reste indépendant de toute application.

La figure ci-dessous extraite de [Har97] montre l'architecture de l'outil méta-générique qui y est implanté. Sur ce schéma, nous distinguons les deux modèles de conception le méta-modèle du Processus de Conception et celui du Produit ainsi que les liens établis entre ces deux modèles.

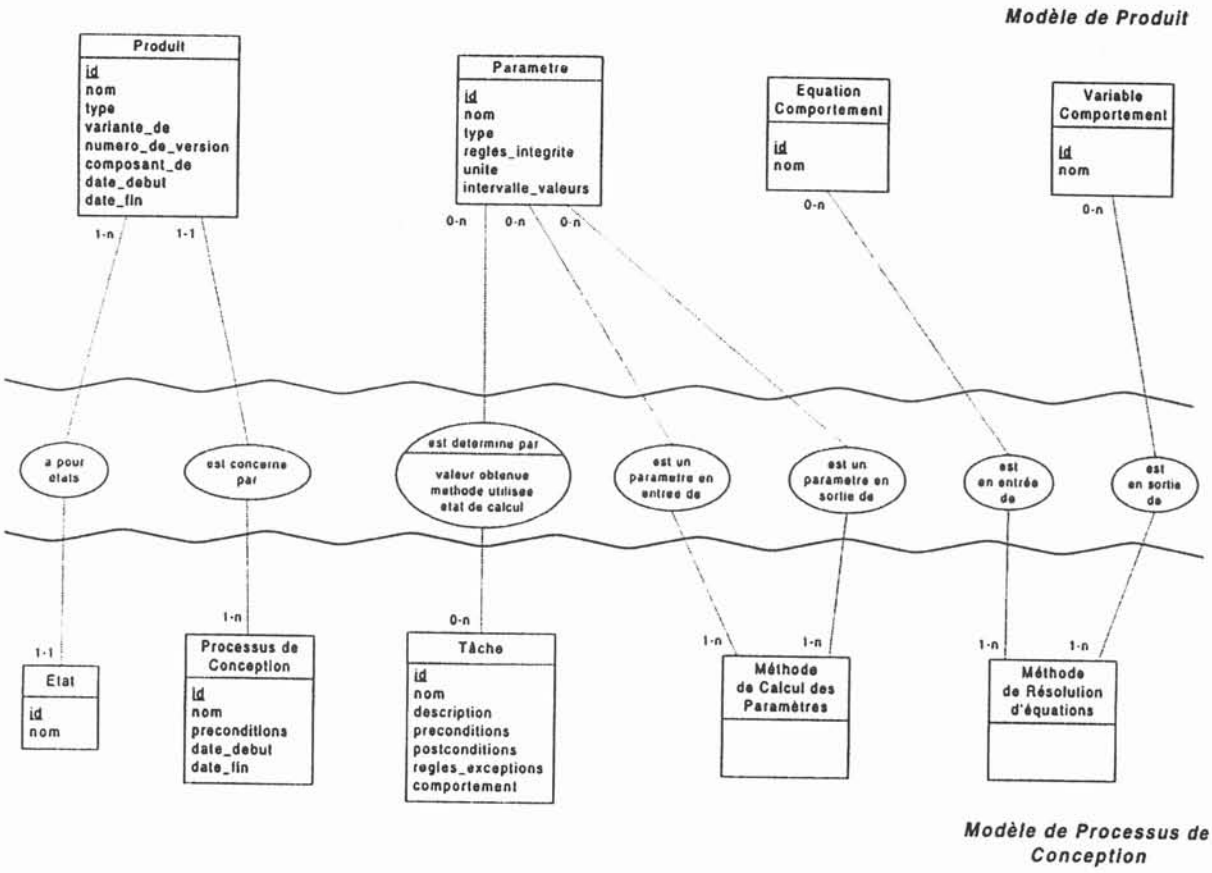


Figure II-6 Architecture des modèles implantés dans un méta outil

L'outil méta-générique de Conception à concevoir doit être complètement indépendant de la connaissance qu'il véhicule (gestion de multiples sources de connaissances) ainsi que du produit à réaliser (issus de domaines complètement disjoints tels que la mécanique, le génie électrique, l'informatique…).

Dans cet objectif, il faut arriver non seulement à extraire les invariants des mécanismes de conception mais aussi à construire un moule unique capable de reconduire et de relancer chacun de ces processus de conception. Les concepts de « Tâche » à réaliser et le concept de « Paramètre » se distinguent parmi ces invariants comme le montre la figure ci-dessus.

II.3.3.2 Positionnement des outils méta-génériques

En manipulant un outil méta-générique l'utilisateur doit manipuler des concepts méta-génériques. Par exemple au lieu de traiter des paramètres, il agit sur des méta-paramètres du produit à concevoir. Il n'a pas à se soucier à ce niveau de la nature de l'objet à concevoir.

En exprimant son souhait de spécifier son domaine d'application X (génie mécanique, génie électrique, génie logiciel), l'utilisateur traitera de la spécification du méta-modèle produit au domaine X.

Un outil générique peut donc être conçu par instanciation des méta-modèles de l'outil méta-générique. De même un outil spécifique peut être mis en œuvre en instanciant les modèles de l'outil générique. A l'instanciation des modèles d'un outil correspond une chute du degré d'abstraction. Rappelons que la définition de l'instanciation est «l'affectation des attributs des concepts » : c'est à dire qu'en instanciant, nous nous situons dans un cadre plus spécifique.

Sur le schéma ci-dessous (Fig. II-7) nous illustrons l'exemple de l'instanciation de certains concepts des modèles génériques pour la réalisation d'un outil de Conception dédié à la conception de Moteurs Asynchrones [Har97].

Les outils de Conception méta-génériques vont-ils révolutionner la méthodologie de conception ? Quel est l'intérêt de concevoir les produits à un niveau conceptuel plus élevé que celui proposé par les outils existants ? Peut-on vraiment obtenir une aide efficace à la conception en utilisant ces outils ? Les outils spécifiques n'offrent-ils pas une meilleure qualité de résolution des problèmes ? La performance de l'aide apportée par l'outil de Conception est-elle déjà garantie pour chercher à créer des outils méta-génériques ?

Nous pensons qu'une étude comparative des performances des outils méta-génériques et les autres outils (outils génériques et outils spécifiques) n'est pas à faire ; car nous comparons là deux outils destinés à deux communautés différentes. Les premiers sont développés pour générer les seconds. Alors que les premiers sont destinés aux concepteurs d'outils de Conception, les seconds sont destinés aux «simples » utilisateurs des outils de Conception.

Cette méthodologie de la méta-conception est une innovation dans le monde de la conception d'outil de Conception. Les premiers travaux d'une telle approche sont aux stades de validation.

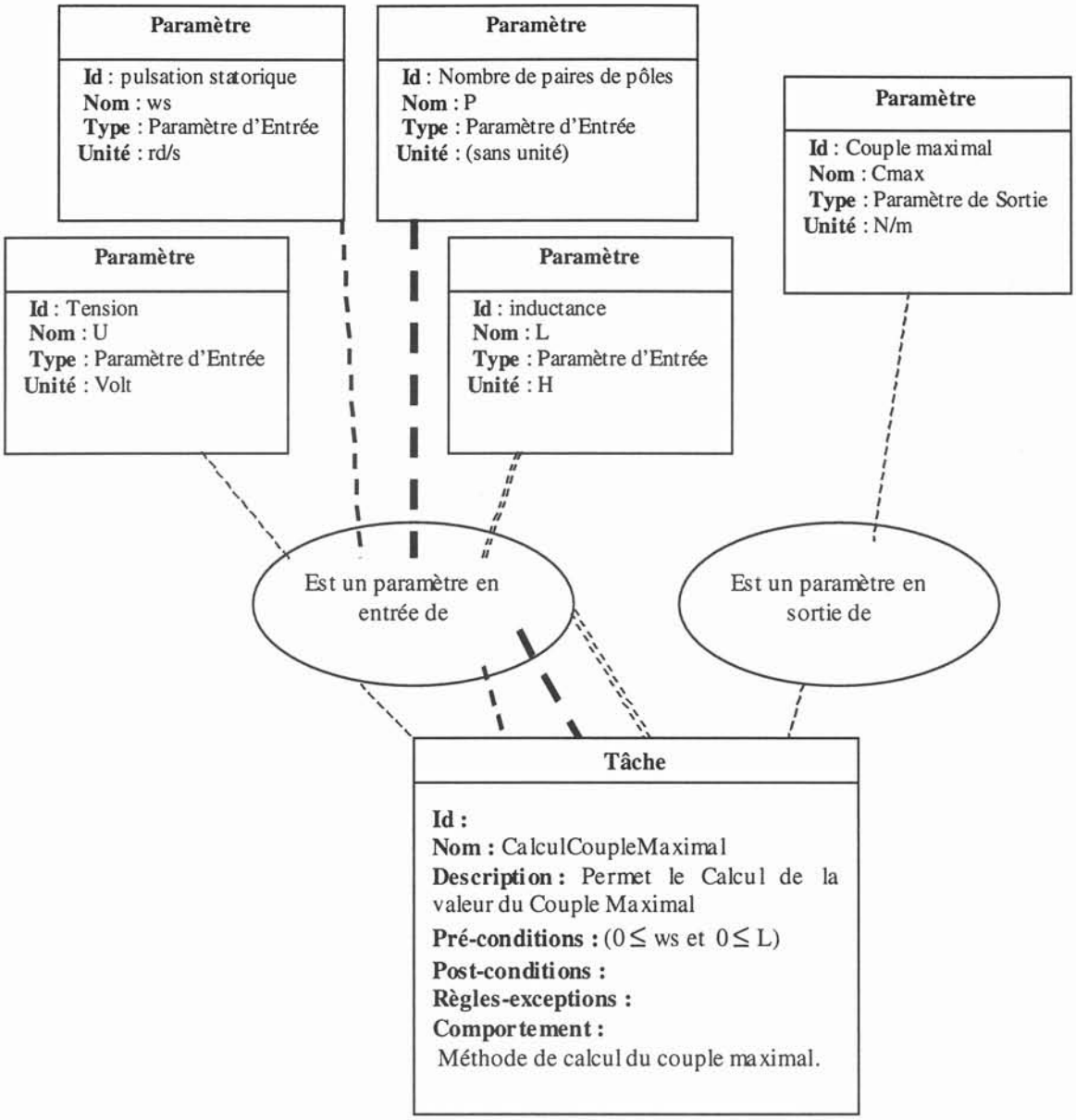


Figure II-7 Instanciation du méta-modèle de l'outil de conception pour la réalisation d'un outil de conception de moteurs Asynchrones

Toutefois, nous restons optimistes quant aux gains de temps inestimables apportés par les outils méta-génériques. En matière de conception d'outil de Conception, nous remarquons une avancée considérable puisque qu'en validant un outil de Conception méta-générique, la démarche pour le développement d'outil de Conception spécifique n'est plus une conception innovante en soi mais serait plutôt de l'ordre de la conception routinière.

Pour créer un outil d'aide à la conception, les développeurs n'ont plus à partir de rien pour le concevoir. Leur travail consisterait à instancier les méta-modèles : pour générer l'outil souhaité, ils n'auront qu'à remplir le contenant vide.

II.4 Conclusion

L'objectif de notre travail est de proposer une méthodologie d'intégration des outils d'aide à la Conception. Pour cela, il a été nécessaire de comprendre la formalisation de la Conception, détaillée tout au long de ce chapitre.

Ce deuxième chapitre, nous a permis de montrer que la conception d'outils de Conception est fortement contrainte par la connaissance du produit et celle du cheminement de la résolution. Dans notre environnement de conception, nous opterons pour une séparation entre le modèle du produit et le modèle du processus, nous justifierons ce choix dans le chapitre IV.

La classification des outils proposée ici établit une hiérarchie des outils de Conception selon leurs niveaux de généricité. A ce jour, à notre connaissance, aucun outil méta-générique ni générique n'est disponible pour supporter la totalité de la conception en génie électrique.

Dans l'exercice de son activité, le concepteur en génie électrique s'appuie sur un ensemble d'outils parmi lesquels on trouve des outils spécifiques et des outils génériques. Cette complémentarité est une condition nécessaire pour apporter une assistance par ordinateur efficace à la conception des produits électriques.

Le besoin d'un outil d'aide à la conception complet qui assisterait le concepteur dans la totalité de l'activité, qui prendrait en charge aussi bien la caractérisation du produit ainsi que celle du processus demeure insatisfait.

Notre contribution n'a pas pour objectif de mettre en œuvre un outil méta-générique mais de proposer une démarche pour coupler les outils indépendamment de leurs niveaux de généricité dans un environnement de Conception Intégrée.

C'est pourquoi nous avons imaginé une nouvelle approche pour supporter une automatisation partielle de l'utilisation d'outils de Conception complémentaires. L'objet du prochain chapitre est d'exposer en détails le principe et les mécanismes de cette approche d'intégration des outils de Conception en génie électrique.

Les Plates Formes Intégrées pour la Conception en Génie Electrique

Sommaire du troisième chapitre :

III Les Plates Formes Intégrées pour la Conception en Génie Electrique

Introduction

Il découle de notre Etat de l'art des outils de Conception en génie électrique que les concepteurs des produits électriques ont à leurs dispositions une panoplie d'outils. Nous nous sommes penchés sur l'étude de l'usage de ces outils. Dans le cadre d'un processus de conception classique en génie électrique, le concepteur fait appel très souvent à un nombre fini d'outils de Conception. Nous convenons de désigner un jeux d'outils par le terme « package » de logiciels.

Comme développé au chapitre précédent, l'outil de Conception complet multi-tâches n'existe pas encore à notre connaissance pour supporter le concepteur dans la totalité de son activité. Nous voulons combler cette lacune en proposant une architecture permettant l'usage d'outils de Conception en interaction.

Dans le premier paragraphe de ce chapitre, nous allons passer en revue les inconvénients de l'intégration manuelle des outils de Conception. Ensuite, nous allons expliciter le concept des Plates Formes Intégrées dédiées à la conception en génie électrique. Dans la suite du rapport nous convenons de désigner une Plate Forme Intégrée dédiée à la conception en génie électrique par l'acronyme EDIP (Electrical Design Integrated Platform). Les caractéristiques d'une EDIP sont détaillées dans le troisième paragraphe. Les hypothèses adoptées pour le développement d'une plate forme intégrée dédiée à la conception en génie électrique feront l'objet du quatrième paragraphe. Pour conclure, nous mettrons l'accent sur l'intérêt du concept de plate forme intégrée en exposant ses atouts.

III.1 Les inconvénients d'une intégration manuelle

III.1.1 La définition d'une intégration manuelle

Aujourd'hui, la conception assistée par ordinateur n'est plus une approche innovante en soi. Les outils de Conception existants dans le commerce apportent une aide à la résolution des problèmes électriques, physiques, mécaniques ou thermiques... Le concepteur ne peut que se réjouir de l'amélioration continue des performances de ces outils.

Avec la disponiblité des outils de Conception, le concepteur s'appuie sur plusieurs outils indépendants pour la recherche d'une solution à son problème. Il lui revient alors de choisir l'outil ou les outils les plus adaptés à sa démarche et à définir sa stratégie de résolution.

Lors d'une activité de conception préliminaire (cf. chapitre I), le concepteur s'appuie généralement sur des outils mono-fonction. L'exécution de chaque outil contribue à l'élaboration du produit. En pratique, les entrées d'un outil de Conception donné peuvent figurer parmi les sorties d'un autre outil de Conception et vice versa.

Définir le produit complet revient à faire converger les informations partagées entre plusieurs outils. Afin d'illustrer l'exercice de l'intégration manuelle, nous citons le cas d'un concepteur qui s'appuie pour la conception d'un contacteur électrique sur trois outils de Conception:

- un outil de calcul et d'optimisation
- un outil de simulation (Eléments Finis)
- un outil de visualisation

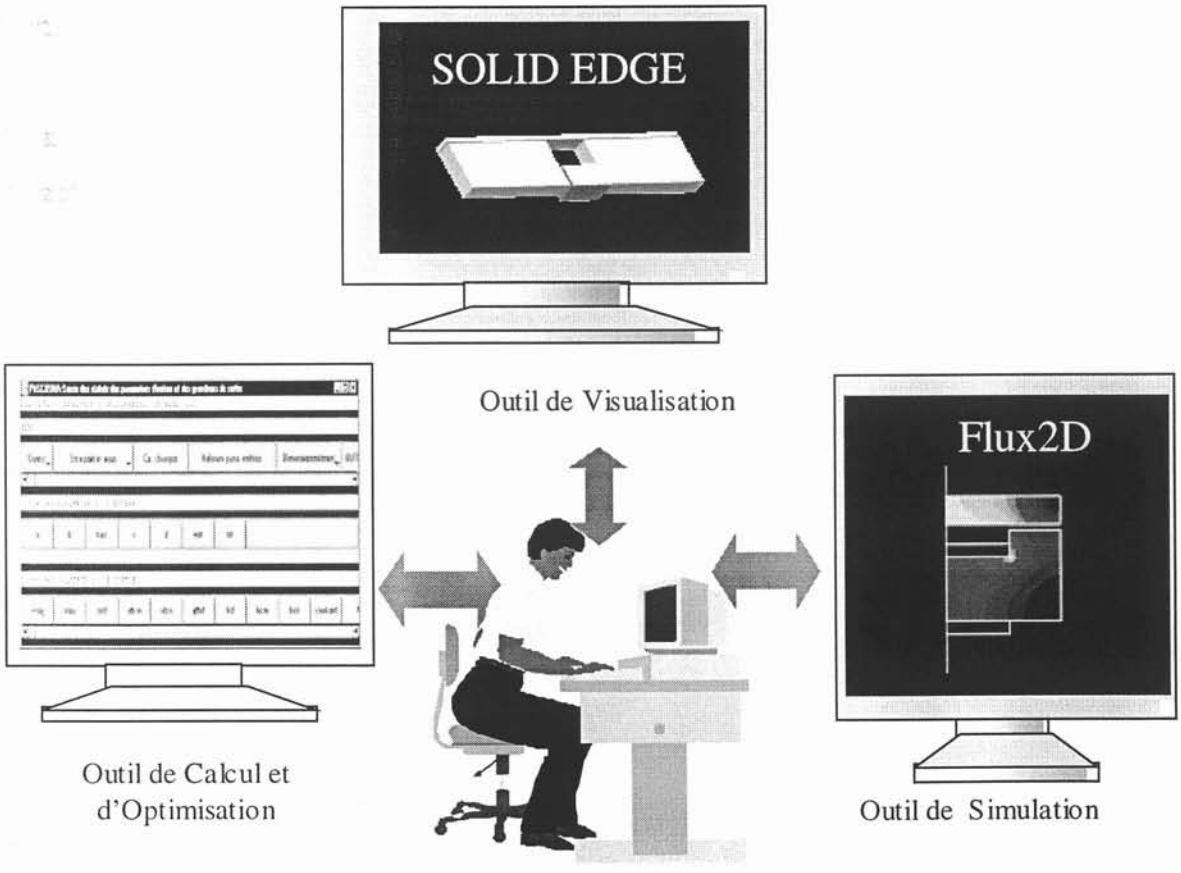


Figure III-1 Une intégration manuelle des outils gérée par le concepteur

Les paramètres géométriques du dispositif étudié sont partagés entre les logiciels. Ainsi à chaque fois que le concepteur apporte des modifications de la géométrie du contacteur au niveau d'un outil, il doit actualiser les deux autres outils afin de garantir la cohérence de l'application étudiée à travers ses outils. Il ne s'agit pas d'introduire seulement les paramètres modifiés mais aussi tous ceux qui y sont liés et varient en conséquence, tel que la valeur de l'induction au niveau de l'entrefer. Sur la Fig. III-1 ci-dessus nous donnons une image de cette interaction manuelle.

A chaque fois qu'il se réfere à un outil de Conception, il doit confronter les informations fournies par l'outil en question avec celles obtenues à partir des autres outils déjà utilisés dans son processus de conception. C'est alors qu'il peut être confronté au problème de transfert d'informations.

III.1.2 Le problème de transfert des informations

La navigation (les allers-retours) entre les outils de Conception est une opération répétitive. Lors de la conception de son produit, le concepteur n'a pas recours qu'une seule fois aux outils de Conception. Il est amené à tester plusieurs configurations de son produit. A chaque fois qu'un changement significatif a eu lieu, il revient au concepteur de le propager au niveau des différents outils qui le mettent en jeu. Cette opération est donc itérative et le transfert d'informations consomme du temps de travail du concepteur. Ce temps est d'autant plus important que les informations partagées entre les différents logiciels sont nombreuses et que les logiciels utilisés sont divers ou compliqués.

D'autre part, lors du transfert des données, le risque d'erreurs et d'omissions est loin d'être nul : l'oubli est un facteur humain qui peut avoir des conséquences lourdes dans l'activité de conception.

En effet, le concepteur pourrait apporter une modification au niveau d'un outil donné et la notifier à tous les outils concernés ; tout comme il pourrait omettre d'actualiser cette modification au niveau d'un outil donné. Si cette modification porte sur une donnée sans grande importance dans la globalité du produit, le produit résultant de l'interaction entre outils peut être une solution acceptable.

Cependant si cette donnée est conséquente, le produit résultant sera donc incohérent. Le concepteur buterait alors sur un problème dont il ne saurait pas identifier l'origine. Le risque d'erreurs et d'omission est amplifié par la complexité du problème traité.

Le transfert de données se traduit en pratique par une série d'allers retours entre les différents outils. Généralement, ces allers retours ne sont pas structurés ; le concepteur ne garde pas une trace de son cheminement. Pour reproduire un enchaînement particulier des outils, le concepteur doit relancer les différentes étapes déjà réalisées et retransmettre les informations manuellement d'un outil à l'autre et les liens entre les outils sont à rétablir.

Chaque concepteur a sa propre façon de faire interagir les outils. En intégrant manuellement les outils, aucune capitalisation (cf. chapitre II) de l'exercice de création de processus n'est faite : le savoir-faire n'est pas sauvegardé pour une éventuelle réutilisation.

III.2 Le concept de plate forme intégrée dédiée au génie électrique

III.2.1 Définition du concept de plate forme intégrée dédiée au génie électrique

Nous tenons à préciser que le but de notre approche n'est pas de mettre en œuvre l'outil de Conception intelligent prenant en charge la totalité du processus de conception assistée. Mais plutôt d'aider les concepteurs dans l'exercice de leur métier, en structurant l'aide apportée par les différents logiciels et en implantant une architecture informatique permettant d'automatiser partiellement cette interaction entre les outils et de réutiliser les processus de conception.

Notre objectif est de définir un support informatique qui permettrait de gérer l'exploitation de n outils (de Conception de produits électriques) ensemble. Ce support informatique n'est autre qu'un contenant d'outils de Conception qui serait doté d'un mécanisme pour la communication entre les outils qu'il renferme. Il s'agit donc d'une plate forme dédiée à l'intégration d'outils et qui permettra de supporter un processus de conception.

A notre connaissance, l'intégration des outils n'etait pas la priorité des développeurs de logiciels de Conception. Cette problématique est émergente ; certains travaux de recherches se sont penchés la dessus.

Dans cet ordre d'idée, Sassine [Sas94] a entrepris de développer une architecture de tableaux noirs (BlackBoard) en vue de supporter la conception intégrée en génie électrique. Il propose un support informatique permettant de coupler et de coordonner les outils de Conception dans un processus de conception de produits électromagnétiques.

Notre ambition est de formaliser l'intégration des outils de Conception en proposant une méthologie pour le développement de plates formes intégrées dédiées à la conception en génie électrique.

III.2.2 La multiplicité des plates formes intégrées dédiées au génie électrique

Un usage des outils de Conception sélectionnés représente un processus de conception assistée par ordinateur. Il existe plusieurs outils pouvant offrir une éventuelle aide pour une même problématique : le choix s'offre à l'utilisateur. Une fois le « package » d'outils choisi, il existe plusieurs façons possibles de les utiliser ensemble.

A partir d'un même package plusieurs scénarios d'utilisation peuvent d'être envisagés par le concepteur. Nous considérons que chaque scénario émane d'une stratégie d'utilisation des outils et représente un processus de conception assistée par ordinateur.

Nous admettons donc que, «pour un problème de conception donné, il n'existe pas un mais plusieurs processus de conception possibles » [Har97].

Nous pensons que de tels scénarios doivent être capitalisés. A travers notre travail, nous allons proposer de structurer la création des scénarios en vue de leurs réutilisation.

A l'issue de nos travaux nous n'allons pas proposer la plate forme intégrée dédiée à la conception en génie électrique mais nous allons proposer une méthodologie d'intégration afin qu'à partir d'un package de logiciels de Conception distincts le concepteur puisse mettre en œuvre une plate forme intégrée adéquate à son processus. Cette plate forme devant être réutilisable, extensible et fiable [Bbh99].

Une EDIP correspond à un usage spécifique d'un package d'outils de Conception. Ainsi le développeur à partir d'un même package d'outils peut réaliser non pas une EDIP mais plusieurs EDIP.

Toutes les EDIP seront construites selon une méthodologie que nous allons formaliser.

Dans les deux paragraphes suivants, nous définissions les différents aspects inhérents au concept d'EDIP : ses caractéristiques et ses hypothèses.

III.3 Les caractéristiques d'une plate forme intégrée

Afin d'aller vers une conception intégrée, nous pensons que chaque EDIP implantée soit :

- hybride
- modulaire
- ouverte
- générique
- distribuée

III.3.1 Définition d'une plate forme hybride

Une plate forme est dite hybride si elle est composée par des éléments disparates. Cette disparité peut porter sur plusieurs critères comme:

- Les fonctions des outils.
- Les langages de programmation.
- Les systèmes d'exploitation.

La première contrainte permettra à la plate forme de supporter des outils de Conception distincts. En effet, pour concevoir un produit électrique le concepteur s'appuie sur un ensemble d'outils de Conception spécifiques et génériques (cf. deuxième chapitre de ce rapport). La plate forme à implanter doit lui permettre d'intégrer l'ensemble des outils dont il a besoin pour l'exercice de son métier moyennant une adaptation de chaque outil (cette adaptation sera explicitée dans le quatrième chapitre).

Ainsi, nous pouvons imaginer dans ce cadre une EDIP pour la simulation, une EDIP dédiée à l'optimisation tout comme nous pouvons imaginer une EDIP combinant des outils de simulation et des outils d'optimisation.

Ces outils de Conception diffèrent les uns des autres de part le langage de programmation utilisé pour leurs codages : par exemple, sur une même plate forme de conception, des outils codés en Fortran peuvent être amenés à cohabiter avec des outils programmés en C/C++, en Java, etc... Une EDIP doit être conçue pour gérer l'hétérogénéité.

La majorité des outils de Conception est mono-système : l'outil ne peut être exploité que sur un seul système d'exploitation. Ainsi, certains logiciels de Conception sont compatibles avec la plate forme Windows alors que d'autres sont compatibles avec Unix. Dans certains cas, le fournisseur de l'outil peut proposer une version adéquate à chaque environnement ; tel est le cas pour le logiciel Flux [Flu2] (il existe une version Flux PC et une autre version pour la plate forme Unix).

A défaut de disponibilité de différentes versions, le concepteur doit travailler sur différents environnements. Nous n'imposons pas de contrainte de choix d'un environnement de travail : la structure informatique d'EDIP doit offrir au concepteur la possibilité d'intégrer des outils de conception d'environnements hétérogènes. Dans le langage des informaticiens, il s'agit de gérer l'intéropérabilité. La solution à cette problématique consiste à mettre en œuvre le concept d'EDIP distribuée (détaillé à la fin de ce paragraphe).

A l'issue de la transformation que nous nous proposons de faire subir à chaque outil candidat à l'intégration, l'outil en question ne sera pas en mesure de migrer vers un autre environnement d'exploitation. Mais il sera capable de communiquer avec des logiciels situés sur un autre système d'exploitation (cf. Fig.III-2).

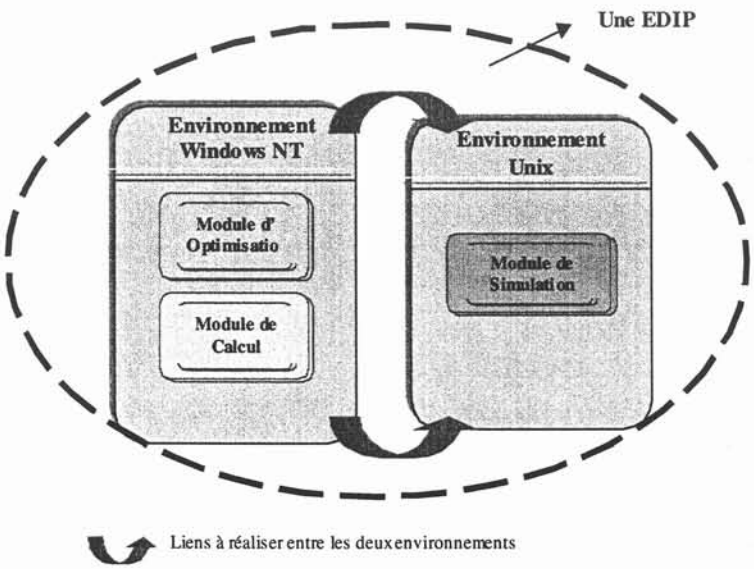


Figure III-2 Un exemple de plate forme distribuée

III.3.2 Définition d'une plate forme à composants

Les éléments de la plate forme ne doivent pas être imbriqués les uns dans les autres. Chaque élément mis en œuvre dans une plate forme donnée doit être réutilisable indépendamment de cette plate forme X. Il doit être facilement adaptable pour la réalisation de nouveaux processus de conception intégrée.

Nous désignons par élément à ce niveau tout logiciel ainsi que le développement informatique mis en œuvre pour son adaptation à l'intégration. A travers cette adaptation, chaque logiciel est transformé en un composant logiciel (cf. schéma ci-dessous).

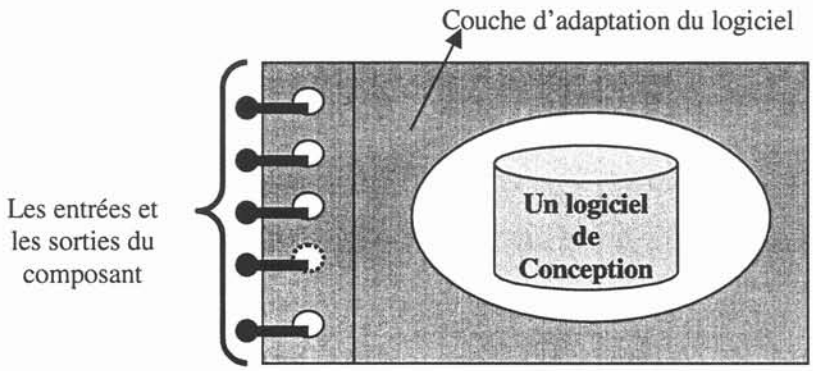


Figure III-3 Structure d'un composant logiciel

En effet, chaque EDIP est bâtie selon le modèle de composants. C'est à dire que la plate forme est constituée de plusieurs éléments (composants). Chaque élément (composant) de la plate forme peut fonctionner indépendamment des autres éléments. Il peut communiquer avec les autres composants. Des règles sont à expliciter pour définir le standard choisi pour la communication entre les différents composants logiciels intégrés dans une EDIP.

Nous venons d'introduire dans ce paragraphe le concept Composant logiciel. Les composants logiciels que nous allons développer correspondent à une implémentation partielle du concept de Composant standardisé. Nous reviendrons en détail sur la mise en œuvre d'un composant de Conception à partir d'un logiciel de Conception dans le quatrième chapitre.

III.3.3 Définition d'une plate forme ouverte

Chaque plate forme intégrée doit être extensible. Le concepteur doit avoir la possibilité de la remanier facilement. Il doit être capable de lui ajouter un nouvel outil ou d'en supprimer un ou plusieurs sans grandes difficultés. La plate forme doit se présenter comme un support informatique flexible à l'intégration.

III.3.4 Définition d'une plate forme générique

Chaque plate forme doit être indépendante des outils qu'elle renferme. Les règles de communication entre les différents composants logiciels doivent êtres génériques et complètement indépendantes des spécificités de chaque outil.

Le mécanisme à mettre en œuvre pour la gestion des interactions entres les différents logiciels s'appuie sur les invariants des composants logiciels.

III.3.5 Définition d'une plate forme distribuée

Les logiciels à intégrer dans une plate forme peuvent avoir le même système d'exploitation mais ils peuvent être installés sur différents environnements physiques (sur des machines différentes, dans des lieux géographiques distants).

Nous souhaitons offrir aux différents acteurs répartis géographiquement (et intervenants pour la conception d'un même produit électrique) la possibilité de contribuer à un même processus de Conception ; c'est pourquoi nous parlons d'une plate forme distribuée.

La mise en œuvre des liens de communication entre ces composants réparties est possible grâce aux apports des nouvelles technologies (Corba [Corba], Java [Jav1.2], OLE [Pro1]).

III.4 Hypothèses du concept de la plate forme intégrée proposée

III.4.1 Orientation des plates formes de Conception vers le Dimensionnement

La conception des produits électriques est encore fortement axée sur le dimensionnement des systèmes [Fez97]. Le dimensionnement constitue le plus grand volet de l'exercice du concepteur.

Rappelons que dimensionner un produit électrique consiste à résoudre un problème inverse. La conception d'un produit électrique revient alors à la détermination de ses paramètres en prenant en compte les contraintes de conception (pouvant être imposées par le cahier des charges ou par les limitations techniques).

Dans le cadre de nos travaux, nous allons contribuer à la définition et à la mise en place d'environnements intégrés pour le dimensionnement. Notre ambition est de mettre en œuvre une méthologie pour le développement de plates formes , entièrement paramétrées, dédiées à la conception intégrée en génie électrique.

Dans un package d'outils de dimensionnement, nous notons le croisement de paramètres entre les différents logiciels. En effet, chaque outil a sa propre base de connaissances. En vue du dimensionnement, nous pouvons assimiler (au premier degré) cette base à la liste des paramètres manipulés. Avec cette hypothèse, nous pouvons schématiser un package d'outils de Conception comme suit (Fig. III-4).

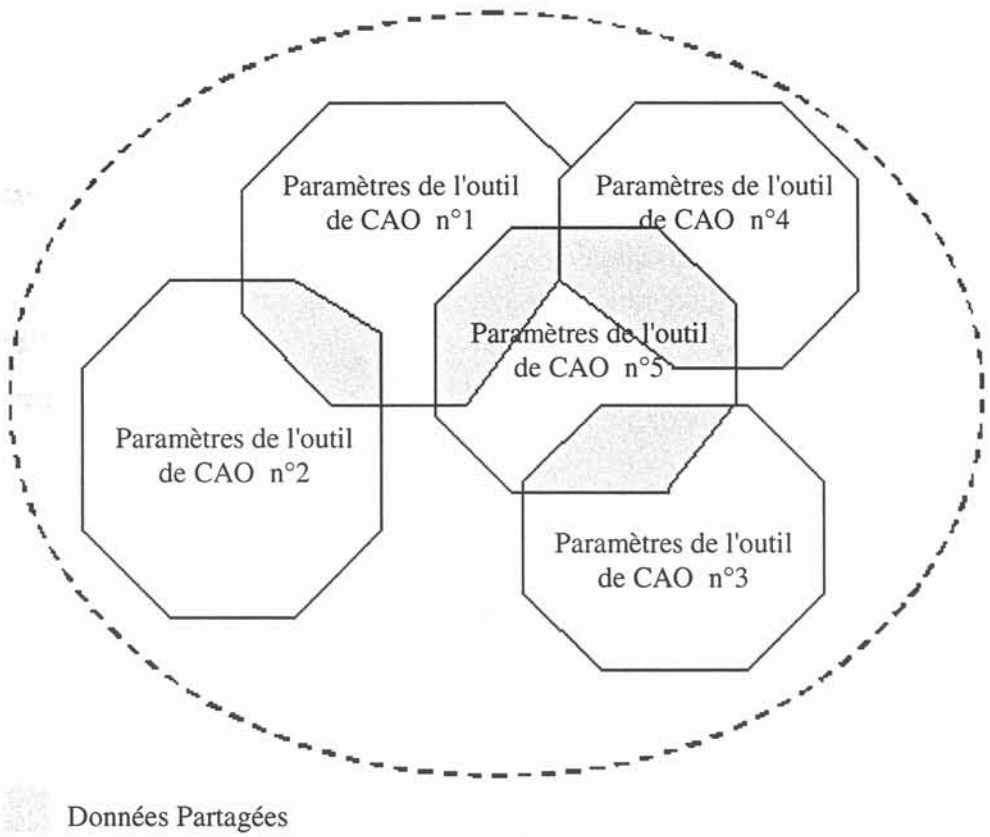


Figure III-4 Schéma d'un package d'outils de Conception

A priori au niveau d'une EDIP, toute la communication entre les outils passe par le transfert des données des paramètres. Ainsi les données à partager entre les outils se confondent avec les paramètres à la croisée des outils. Nous proposons une méthodologie pour gérer les liens entre les paramètres issus de différents outils de Conception.

La principale tâche d'une EDIP est d'automatiser la propagation des données entre ses différents éléments

III.4.2 Définition d'un paramètre en vue du dimensionnement

Les spécifications du produit sont représentées par des paramètres. Les paramètres permettent de formaliser toutes les données utiles pour la description du produit à concevoir. Ainsi, toute la connaissance du produit est traduite au niveau du paramètre.

Rappelons que les produits électriques sont le siège de plusieurs phénomènes physiques (cf. le premier chapitre). En conséquence, les paramètres du produit sont de natures différentes : géométriques, électriques, mécaniques, thermiques, économiques...

Dimensionner un dispositif électrique consiste à évaluer les grandeurs incomplètes à partir de la connaissance acquise du produit (tels que les paramètres imposés par le cahier des charges). L'usage des outils de Conception permettrait au concepteur d'élargir rapidement l'espace de cette connaissance ; à condition d'avoir établi au préalable les relations entre les données connues et celles qui restent à connaître.

Comment alors définir un paramètre de dimensionnement ? Au concept paramètre est associé toute une sémantique. A chaque paramètre du produit nous associons un ensemble d'attributs pour le décrire. Selon le niveau de détails requis, la sémantique du produit peut être étendue ou réduite aux informations minimales (Fig. III-5).

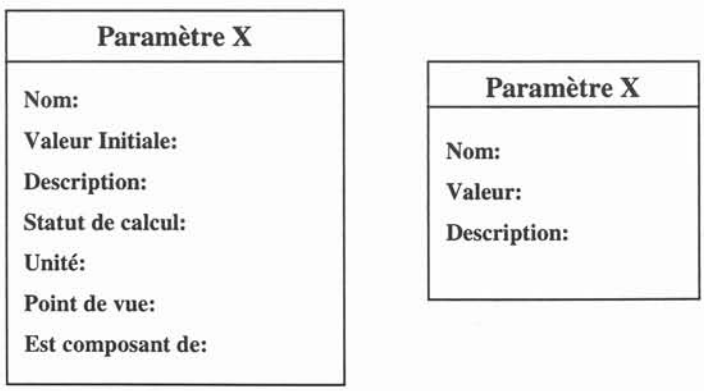


Figure III-5 Sémantique associée au paramètre

A travers chaque EDIP de dimensionnement, à un même paramètre, peuvent correspondre une ou plusieurs sémantiques. En effet, nous pouvons parler de sémantique outil pour chaque paramètre ; cette sémantique correspondrait à la définition du paramètre selon le point de vue de l'outil où il est introduit. Chaque paramètre est alors décrit selon les besoins de modélisation de l'outil qui le manipule.

Prenons par exemple le cas de la grandeur hauteur de la culasse magnétique d'un contacteur électrique. Pour dimensionner le contacteur, le concepteur est amené à utiliser un outil de calcul et un outil d'optimisation : il en découle qu'au moins deux sémantiques seront associées à la hauteur (cf. Fig. III-6).

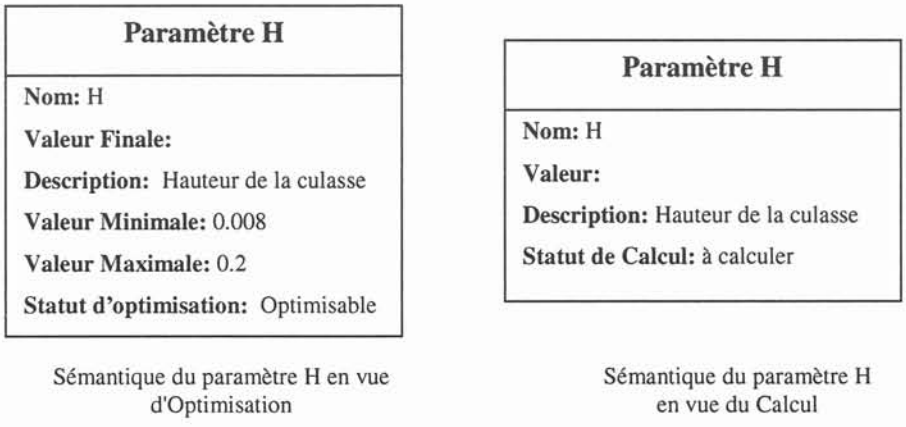


Figure III-6 Sémantique du paramètre H selon l'outil de Conception

Nous pensons qu'il suffit de transmettre à travers la plate forme intégrée une sémantique minimale du produit. Entre outils, le concepteur n'a besoin de transmettre que les informations partagées réellement : il ne communique généralement que le Nom de la grandeur et sa Valeur.

Tous les détails spécifiques sont déjà introduits au niveau de chaque outil. En cas de besoin, le concepteur pourra aller les consulter en se positionnant au niveau du composant correspondant. Nous évitons par ce choix la redondance de l'information et surtout nous réduisons au minimum utile la taille des données d'intégration à transmettre entre composants.

Toutefois la structure d'EDIP étant générique, il est possible d'étendre la sémantique par l'intermédiaire d'une couche de codes supplémentaire afin qu'au niveau d'EDIP non seulement les paramètres à la croisée des outils sont transférés mais également les paramètres combinés : par exemple, l'équation " $a = b + c$ " où a est un paramètre utilisé dans un premier outil et b et c deux autres paramètres fournis par un deuxième outil.

III.4.3 Modèle paramétré du produit

Comme nous l'avons exposé, dans le deuxième chapitre, la connaissance liée au produit électrique peut être représentée sous différents formats (courbes, abaques, tableaux, équations). Il est difficile de gérer cette hétérogénéité du format. Nous avons convenu de s'affranchir de cette difficulté en se limitant au niveau du concept d'EDIP à l'exploitation de la connaissance axée sur les paramètres.

Cette restriction peut dégrader la qualité de la solution conçue mais elle permet d'aider le concepteur à se forger une idée de son dispositif. Toutefois si des données importantes sont négligées (car elles ne peuvent pas être traduites au format paramétré) le concepteur devra se passer du concept d'EDIP. Il devra alors gérer manuellement l'intégration des outils adaptés au format de ses connaissances.

Dans une EDIP, le dispositif est décrit par un jeux de paramètres que nous appelons le modèle paramétré. La nature des liens entre ses différents paramètres diffère d'un outil à l'autre. Au niveau de chaque outil de Conception, le concepteur projette un modèle paramétré de son dispositif [Ati98] (cf. Fig. III-7).

En pratique au niveau d'un logiciel de calcul analytique, le concepteur décrit son produit par un jeu d'équations. Alors que dans l'outil de Visualisation, seuls les paramètres géométriques sont pris en compte pour la modélisation de la géométrie.

Ainsi dans une EDIP, il y a autant de modèles paramétrés que de logiciels intégrés. Chaque outil renferme un bout de la connaissance du produit à concevoir. En intégrant les outils dans une EDIP, la description paramétrée complète du produit est obtenue en assemblant tous les jeux des paramètres. Nous disposons d'un jeu de paramètres global.

A travers une EDIP à « n » logiciels, le concepteur possède « n » modèles paramétrés. Nous convenons pour l'instant que la cohérence de l'application à travers les composants est à la charge du concepteur.

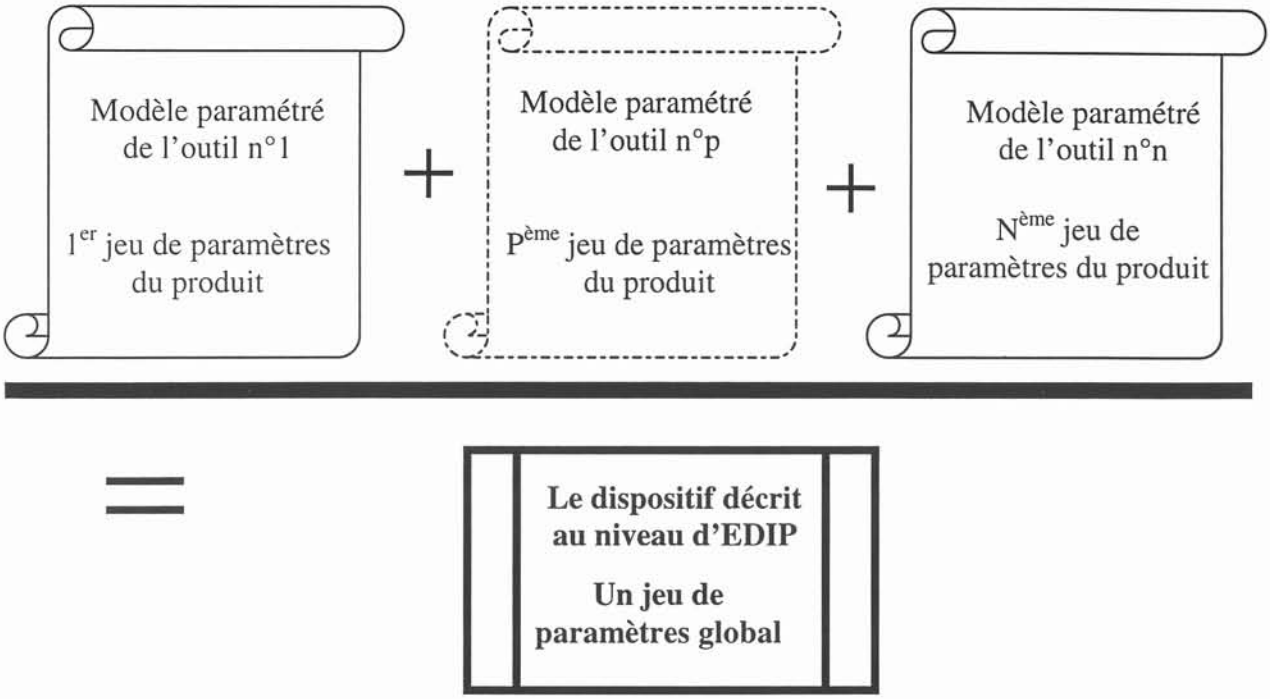


Figure III-7 Le dispositif paramétré à travers EDIP

En effet, la première étape dans l'utilisation d'une EDIP consiste à indiquer pour chaque composant l'application qui sera l'objet de l'intégration. Il revient au concepteur de veiller à éviter d'intégrer à travers une EDIP des applications incohérentes.

Pour illustrer cette situation, reprenons le cas d'une EDIP formée d'un outil de Calcul et d'un outil de visualisation. Si au départ le concepteur charge le modèle paramétré d'un contacteur au niveau du composant de Calcul et le modèle paramétré d'une machine au niveau du composant de visualisation, le jeu de paramètres global n'a plus aucune signification physique. Il n'y a plus lieu d'intégrer parce que les sujets de la conception sont différents.

III.4.4 Objet de conception fixe

Nous avons constaté qu'à priori l'ensemble des outils de Conception fonctionne selon le schéma suivant (Fig. III -8):

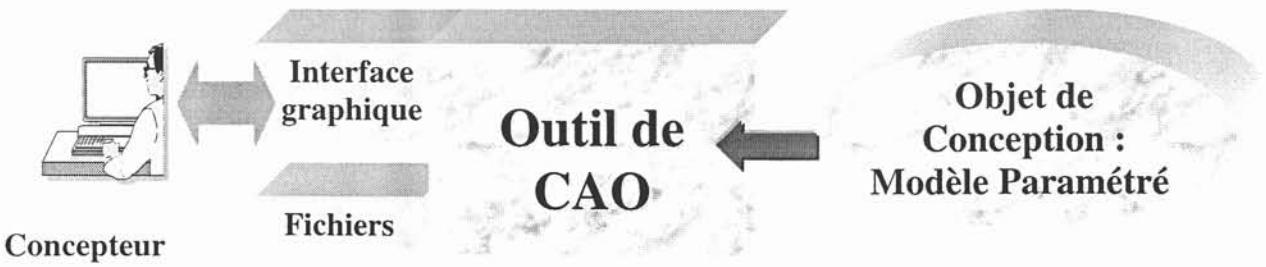


Figure III-8 Usage d'un outil de Conception

Le concepteur ne peut être assisté par une EDIP qu'une fois que son objet de Conception (modèle paramétré) est choisi pour chaque outil à intégrer. Du moment que les relations entre les paramètres ont été définies au niveau de chaque outil ; la topologie de chaque jeu de paramètres est fixée.

Au cours des allers retours entre les outils intégrés, le jeu de paramètres global n'est pas modifié. Ici, l'objectif du concepteur en faisant appel aux outils de Conception n'est plus de trouver une solution mais de trouver une solution optimale.

Par exemple, une fois que le concepteur a choisi le modèle de la machine électrique à concevoir, l'étude qui reste à effectuer consiste souvent à:

- Faire une analyse comparative des résultats issus des différents logiciels.
- Relancer un outil à partir d'un nouveau cahier des charges:
    - Lâcher ou renforcer certaines contraintes sur un ou plusieurs paramètres.
    - Prendre en compte une nouvelle liste de valeurs initiales (dont certaines pouvant être fournies par un autre outil de Conception).

Pour utiliser EDIP, il faut que le concepteur ait par ailleurs construit pour chaque outil à intégrer l'objet de Conception selon la vue de l'outil. Il peut disposer de plusieurs versions de son Objet de conception (à différentes topologies) cf. Fig. III-9.

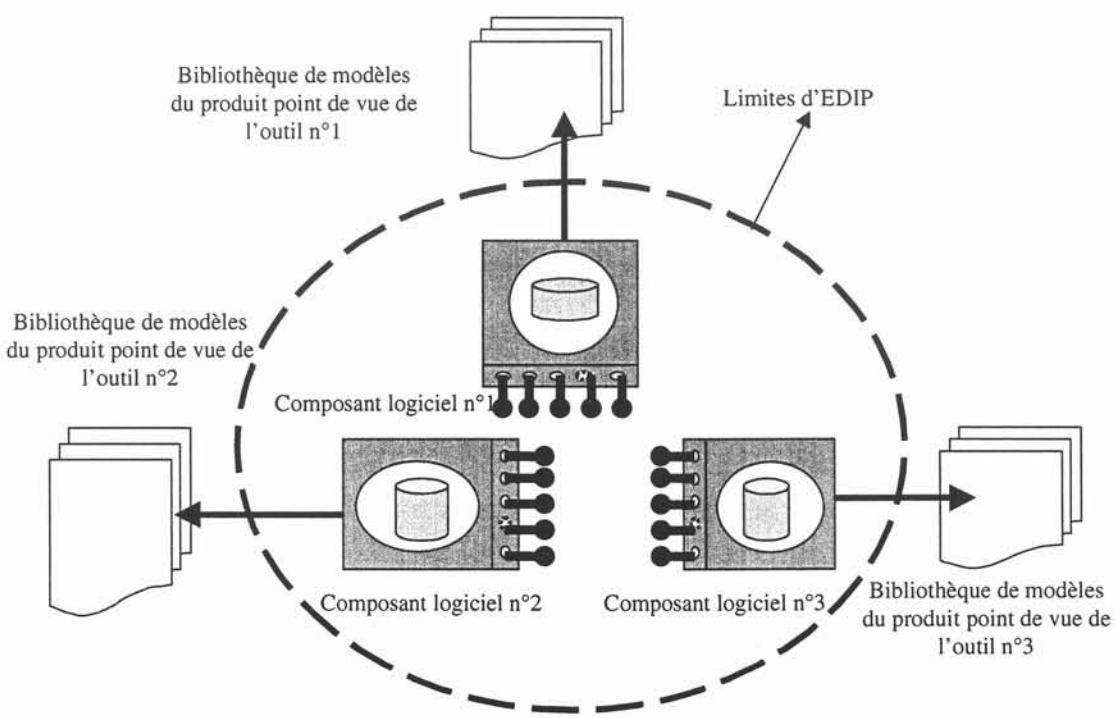


Figure III-9 Bibliothèque de modèles outils au-delà des frontières de EDIP

A travers l'EDIP, il ne peut pas construire un nouvel objet de conception mais il a la possibilité de charger dynamiquement un objet de Conception différent : au niveau de chaque outil l'ancien modèle paramétré sera remplacé par le nouveau.

C'est pourquoi nous précisons que l'intégration des outils n'affecte pas les objets de Conception. L'objet de Conception reste fixe sauf si le concepteur le change explicitement.

III.5 Les atouts d'une plate forme intégrée

Chaque EDIP doit présenter les caractéristiques explicitées ci-dessus. En outre, elle doit remplir les contraintes suivantes :

- Un investissement réduit pour le développement
- La facilité d'utilisation
- La maintenabilité
- Le temps de réponse

III.5.1 Un investissement réduit pour le développement

Nous distinguons deux grandes tâches pour le développement d'une EDIP : la première consiste à fournir les composants logiciels alors que la seconde consiste à générer à partir de ces composants un processus de conception réutilisable. C'est pourquoi, nous scindons les développeurs des plates formes intégrées en deux groupes : les développeurs de composants et les assembleurs.

Ce sont les développeurs de composants qui se chargent de transformer chaque logiciel en un composant logiciel intégrable dans une plate forme de conception. La mise en œuvre de cette adaptation sera d'autant plus ardue que l'outil de Conception n'est pas pré-adapté pour l'intégration (tous les détails de cette adaptation seront fournis au niveau du chapitre suivant).

Une fois l'adaptation du logiciel effectuée, l'assembleur interviendra pour définir son processus de conception à partir des composants logiciels sélectionnés. Il doit être capable de construire la plate forme intégrée correspondant au processus choisi. Il lui revient de mettre en place les liens entre les différents composants tout en respectant le cahier des charges commun à toutes les EDIP.

Il est donc évident qu'avant de bénéficier des atouts d'une EDIP, un investissement en développement est à faire. Jusqu'à l'heure actuelle, à défaut d'outils compatibles avec son environnement de travail, le concepteur optait souvent pour la mise en œuvre d'un outil équivalent. Il s'agit là d'un travail fastidieux qui n'est pas du ressort du concepteur en génie électrique, qui n'est pas forcément très qualifié pour le réaliser.

En mettant en place le concepts d'EDIP, nous souhaitons minimiser les codages et non les annuler. Les coûts d'investissement pour l'adaptation d'un outil sont réduits en comparaison des coûts qui seront dispensés pour le recoder.

III.5.2 La facilité d'utilisation

Une fois développée, chaque EDIP doit être utilisable par tout concepteur en génie électrique. Cette condition n'est remplie qu'en mettant en œuvre une interface Homme-Machine performante et en faisant en sorte que le processus de conception y soit structuré d'une manière claire et transparente pour l'utilisateur.

III.5.3 La maintenabilité

Chaque EDIP doit être facilement maintenable. En cas de besoin, le concepteur peut s'y retrouver sans grandes difficultés afin d'apporter une amélioration ou d'intégrer un nouvel outil. Pour ce faire, il faut que le mécanisme de la plate forme soit explicite et que sa mise en œuvre ne soit pas trop lourde.

III.5.4 Le temps de réponse

Chaque EDIP développée doit justifier son existence. Le gain de temps dans la résolution des problèmes de conception est l'un des objectifs d'une EDIP. Son utilisation doit être plus avantageuse qu'une conduite manuelle du processus de conception assistée par les différents outils.

A priori, en gérant la propagation des informations automatiquement, le concepteur économise toutes les périodes de temps qui étaient nécessaires à l'actualisation des données au niveau des différents outils.

Il faut veiller à ce que l'exécution de la plate forme intégrée soit rapide et donc économique pour la communication des informations lors d'un dimensionnement.

Conclusion

Tout au long de ce chapitre, nous avons décrit notre vision du concept de plate forme intégrée pour la conception en génie électrique. Nous avons passé en revue le cahier des charges d'une EDIP, ses atouts, tout en précisant ses pré-requis.

Nous sommes convaincus qu'une telle approche est une bonne solution pour pallier la non-faisabilité de l'Outil de Conception idéal.

Dans ce chapitre, nous sommes volontairement restés au stade descriptif, c'est à dire que nous n'avons pas explicité comment concrètement réaliser cette approche. Nous allons répondre à cette attente dans le chapitre suivant en dévoilant toute la méthodologie qui permet, à partir d'un package de logiciels, de construire une plate forme intégrée dédiée à la conception de produits électriques.

Méthodologie de Développement d'une Plate Forme Intégrée

Sommaire du quatrième chapitre :

IV Méthodologie de Développement d'une Plate Forme Intégrée

Introduction

Conclusion

Introduction

Ce chapitre s'articule en deux grandes parties. Dans la première partie nous allons expliciter pas à pas comment à partir d'un outil de Conception donné obtenir un composant logiciel facilement intégrable dans une EDIP (plate forme Intégrée pour la Conception). Dans la deuxième partie, nous proposons une technique d'assemblage de n composants logiciels pour la mise en œuvre d'un processus de Conception gérant la propagation d'informations.

IV.A La transformation de l'outil de Conception en Composant Logiciel

IV.A.1 Le concept Composant

« Un composant est un morceau de logiciel assez petit pour qu'on puisse le créer et le maintenir et assez grand pour qu'on puisse l'installer et en assurer le support. De plus il est doté d'interfaces standard pour pouvoir interopérer ». Cette définition du composant fournie dans la référence [CS98] est celle que nous avons adoptée pour le développement d'un composant à partir d'un logiciel.

Un composant logiciel de Conception tel que nous le concevons possède les propriétés suivantes :

C'est une entité autonome :

Un composant logiciel est prêt à l'emploi. Les fonctionnalités proposées par le logiciel à transformer seront maintenues en tant que services au niveau du composant équivalent.

C'est une entité extensible :

Les possibilités de chaque composant sont prédéfinies mais elles peuvent être étendues en créant un nouveau composant en associant le premier composant avec un composant complémentaire.

C'est une entité « prête à brancher » (« plug-in ») :

Un composant logiciel de Conception représente une brique de base pour la construction d'un processus de Conception plus ou moins complexe. Le concepteur ne connaît pas à l'avance tous les agencements possibles du composant.

L'adaptation d'un logiciel existant en vue de sa transformation en un composant doit se faire méthodiquement [Inria]. Ainsi, pour réaliser un composant logiciel à partir d'un outil de Conception, il faut procéder successivement aux tâches suivantes :

(1) Identification des éléments (parties de codes, fichiers exécutables) à réutiliser dans le composant,
(2) Spécification de la structure Objets et de l'architecture du composant à implanter,
(3) Réalisation (codage) et validation du composant.

Nous allons réaliser les composants en utilisant la programmation orientée Objets puisque les composants découlent de la même philosophie.

Rappelons qu'en programmation orientée Objets, tout programme est formé par l'association de plusieurs éléments appelés Objets. Nous reprenons la définition du terme Objet fournie dans [Jav1.2] : « Un Objet est un composant complet d'un programme informatique représentant un groupe de fonctions apparentées et destinées à accomplir des tâches spécifiques. »

Le plus grand apport de la programmation orientée Objets est la possibilité de grouper des objets qui ont des caractéristiques communes sous un même modèle informatique qui n'est autre que le concept de Classe.

Toute Classe constitue un moule informatique permettant de créer plusieurs objets clones, répondants aux même caractéristiques fondamentales. Un cas particulier d'une classe est désigné par le terme instance.

Un composant logiciel manipulé à travers une EDIP est constitué par du code écrit pour encapsuler (cf. glossaire) du code existant ou pour piloter d'autres programmes. Il peut communiquer avec une IHM (Interface Homme Machine) ou avec tout autre composant de Conception.

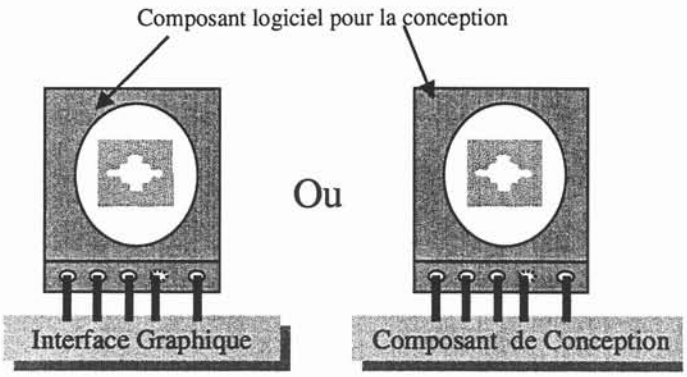


Figure IV-1 A partir d'un composant de Conception

IV.A.2 Evaluation des outils en vue de l'intégration

Nous avons examiné un certain nombre d'outils de Conception (Flux [Flu2], Pascosma [Wur96], Solid Edge [Sev4]) en vue de l'intégration. A l'issue de nos investigations, nous les classons en deux groupes : les outils pré-adaptés à l'intégration et les autres.

IV.A.2.1 Les outils pré-adaptés à l'intégration

Ces outils sont préconçus pour permettre l'échange de données avec d'autres logiciels. Ils sont dotés de mécanismes permettant leurs intégrations avec des programmes de Bureautique (gestion de la documentation technique), mais aussi aux systèmes de conception et d'ingénierie.

Actuellement, deux technologies sont mises en œuvre au niveau des outils de Conception : la technologie OLE et la technologie CORBA.

"CORBA" est l'acronyme de «Common Object Request Broker Architecture", établi par le consortium "Object Management Group". C'est un standard informatique permettant de lier des Objets hétérogènes et répartis. Un Langage de Définition d'Interface (IDL) est associé à tout composant CORBA. L'IDL permet de définir l'interface du composant (lister ses fonctions et sa structure) de l'extérieur facilitant ainsi aux clients son utilisation, [Corba].

"OLE" est l'acronyme de «Object Linking and Embedding » désignant la technologie informatique permettant de lier les Objets. Avec la nouvelle technologie OLE 2, OLE englobe l'ensemble des technologies Composant Objet mis en œuvre par Microsoft, [Pro1].

La mise en œuvre d'une telle technologie dans le développement d'un outil de Conception permet de le piloter par l'intermédiaire de programmes informatiques. En effet, grâce à cette technologie, le logiciel expose ces fonctions aux langages de programmations. L'ensemble de ses fonctions est alors accessible à travers l'architecture Objet.

Dans ce cas, il nous semblait judicieux de profiter des ouvertures existantes dans l'architecture de ces outils pour la transformation du logiciel en un composant logiciel. Il suffit de bâtir au dessus la couche informatique pour le rendre dédié à notre métier (c'est à dire à l'exercice de la conception intégrée en vue du dimensionnement).

Dans le prototype d'EDIP que nous avons réalisé, nous avons utilisé la technologie OLE pour développer un composant logiciel de visualisation à partir de l'outil de Dessin Solid Edge [Pro1]. Dans le cinquième chapitre, nous allons fournir les détails techniques de cette application.

IV.A.2.2 Les outils non adaptés à l'intégration

Nous avons remarqué que l'architecture informatique de certains outils était inadaptée à la communication de l'outil vers d'autres logiciels : l'outil était conçu pour un fonctionnement autonome sans échanges avec l'extérieur.

Ces outils ne sont pas dotés de mécanismes permettant leurs interactions avec d'autres outils de Conception. Dans ce cas, il faut tout mettre en œuvre.

En particulier, comme nous avons choisi d'implémenter nos composants à bases d'Objets, dans le cas des outils dont la programmation est non orientée Objets, il faut commencer par la mise en place d'une structure Objet. Une fois cette transformation réalisée, il est possible de rendre l'outil apte à l'intégration selon le concept EDIP.

D'autre part, nous avons noté dans certains de ces outils (Pascosma, Flux) l'imbrication des modules fonctionnels et ceux d'exploitation. En effet, afin de rendre l'utilisation du logiciel de Conception la plus conviviale possible, l'utilisateur accède facilement à toutes les fonctionnalités de l'outil à travers l'interface, aucune compétence ou connaissance particulière en informatique n'est requise.

Le couplage entre l'interface graphique et le code programmé pour réaliser les fonctions de l'outil est un couplage fort : il n'est pas possible d'accéder aux fonctions du système sans passer par l'intermédiaire de l'interface. Ce choix de réalisation concourrait avec les objectifs de convivialité mais dans notre approche d'intégration, il représente plutôt une restriction. L'accès aux modules fonctionnels du logiciel de Conception est verrouillé.

En cas de couplage fort entre les fonctionnalités de l'outil et son interface graphique, la première tâche pour réaliser le composant logiciel consiste à les découpler (cf. Fig. IV-2).

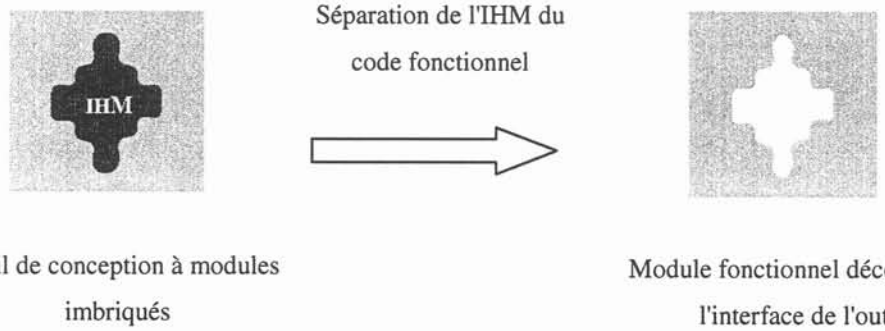


Figure IV-2 Séparation de l'IHM du module fonctionnel

Il est clair que le développement à faire pour transformer un logiciel en un composant logiciel est plus conséquent dans le cas des outils non adaptés à l'intégration.

IV.A.3 Le concept Client/Serveur

IV.A.3.1 La définition de l'architecture Client/Serveur

Il n'y pas une définition unique du concept Client/Serveur : elle varie en fonction des utilisateurs. Dans ce paragraphe, nous allons préciser notre point de vue de cette dualité. La nomination « Client/Serveur » nous laisse deviner l'existence de deux entités distinctes : le Client et le Serveur, et d'une relation qui les lie.

Le modèle « Client/Serveur » est né pour répondre aux évolutions des besoins des utilisateurs d'applications informatiques [CS98]. Ces dernières étaient généralement développées pour être employées par tous les utilisateurs de la même manière. Or, de plus en plus, les utilisateurs ont des besoins spécifiques : l'adaptation d'une application à des besoins particuliers est très coûteuse.

La solution trouvée par les développeurs à cette problématique consiste à bâtir les systèmes informatiques avec une architecture à deux volets. Le premier volet d'un tel système informatique regroupe alors les programmes destinés à la majorité des utilisateurs. Plus exactement, il s'agit d'un module fournisseur de services aux utilisateurs : les informaticiens ont convenu de le nommer le Serveur.

Le deuxième volet de l'architecture Client/Serveur comporte les programmes implantés pour consommer les services. C'est à ce niveau que seront ajoutés (en cas de besoin) les programmes d'adaptations de ces services à chaque utilisateur.

Un Serveur peut être sollicité par plusieurs Clients. La communication entre le module Serveur et le Client peut être établie par un mécanisme de messages ou d'invocations de méthodes (détaillé dans le paragraphe IV.A.4).

Il existe différentes applications du modèle Client/Serveur telles que l'architecture de Client/Serveur pour le partage de fichiers, l'architecture de Client/Serveur pour les transactions bancaires etc.... Nous focalisons nos travaux autour de l'architecture de Client/Serveur à base d'Objets Java.

IV.A.3.2 L'application du modèle Client/Serveur à la Conception Intégrée

IV.A.3.2.1 Le composant Serveur d'Objets

Dans le cas d'un Client/Serveur orienté Objets, l'ensemble des services pourvus par le Serveur sont accessibles à travers des Objets. Le Client, en utilisant le Serveur, manipule des références à des Objets.

Dans cet ordre d'idée, chaque composant logiciel d'EDIP est vu comme un distributeur de services de Conception. La communication entre le module Serveur et le Client, mise en œuvre dans le concept EDIP, est établie par un mécanisme d'invocations de méthodes (détaillé dans le paragraphe IV.A.4). Le Client adresse au serveur une requête (une demande d'utilisation d'un service). En réponse, le Serveur lui renvoie le résultat de l'exécution de la méthode et ainsi de suite (cf. Fig. IV-3).

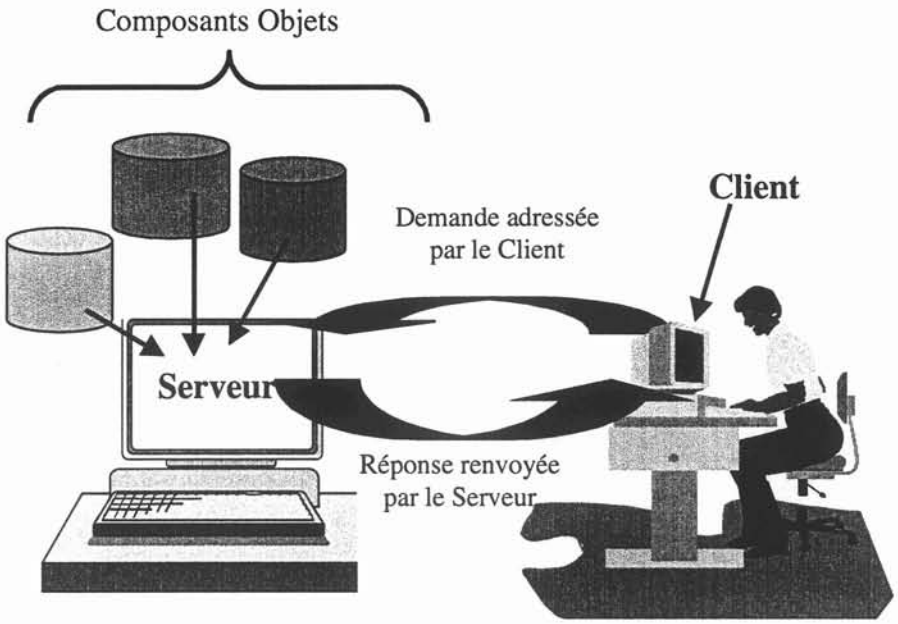


Figure IV-3 Serveur de Composants Objets

Il en découle que chaque EDIP présente trois niveaux hiérarchiques comme le montre la Fig. IV-4. Nous positionnons au premier niveau le ou les éléments récupérés de l'outil à intégrer dans l'EDIP. Le deuxième niveau supporte les Objets encapsulant les services du logiciel ; plus exactement, c'est le composant Serveur à base d'Objets. Enfin au niveau supérieur, le client se place pour consommer les services proposés par le composant logiciel.

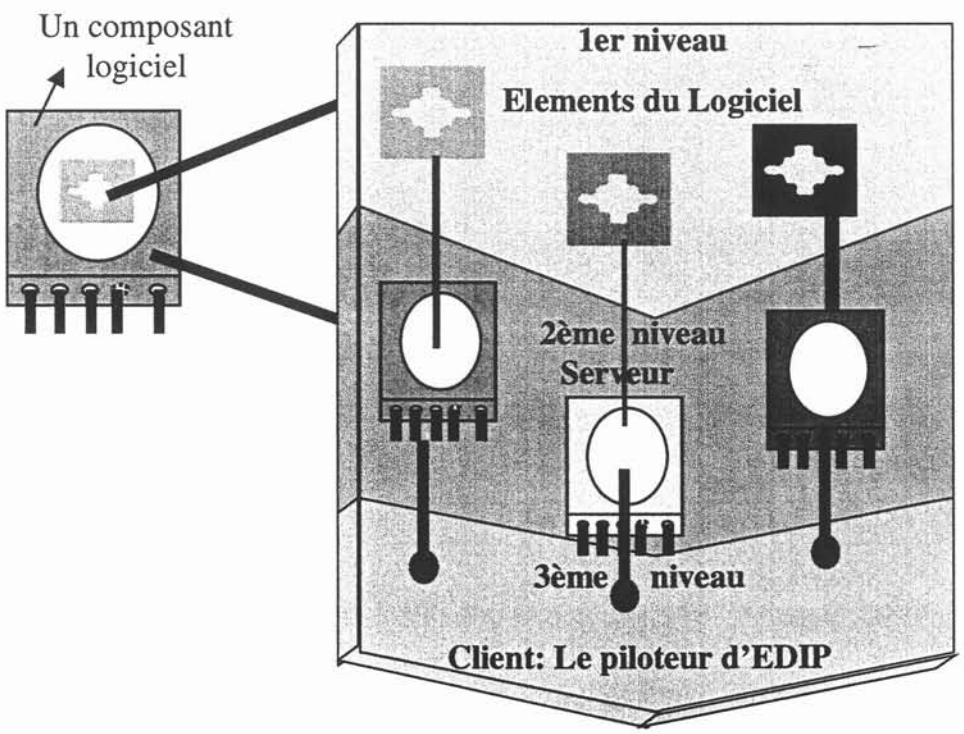


Figure IV-4 Les différents niveaux hiérarchiques d'une EDIP

IV.A.3.2.2 Le choix de JAVA

JAVA est un langage de programmation Objets multi-plate-forme développé par l'entreprise SUN Microsystems en 1991. Des objets JAVA peuvent être conçus sur une plate forme donnée, ils fonctionneront sur toutes les plates formes de la même manière. Pour les réutiliser, pas besoin d'écrire des lignes de codes pour les adapter à la plate forme d'accueil. Le code JAVA est un code mobile. « Il n'y a pas d'incompatibilité due aux types de données entre architecture matérielles et logicielles différentes» [CS98].

Ce langage présente plusieurs atouts : destruction automatique des instances à la fin de leurs utilisations, de nouvelles possibilités pour la gestion de programmes simultanés (multi-thread), etc... Toutefois la portabilité de JAVA le contraint à être plus lent que les autres langages de programmation.

Nous n'allons pas nous étendre sur les détails de JAVA, nous vous renvoyons aux références ([Jav1.2], [JavB], [CS98], [Corba]) pour tout savoir sur JAVA.

Pour notre implémentation, nous avons utilisé la version Java 2.0 du JDK (Java Development Kit) correspondant au langage principal de Java fourni par SUN.

Le langage JAVA est la solution technologique qui nous permettra de mettre en œuvre des composants de Conception portables même si au départ le logiciel de conception est mono-plate-forme. Le composant logiciel pouvant tourner sur toutes les plates formes, il sera installé sur la plate forme choisie par le concepteur et communiquera avec le logiciel logé dans sa plate forme d'origine : nous pouvons avoir à manipuler des Objets répartis.

Nous convenons de structurer nos Objets selon la philosophie de JAVA. Ainsi toutes les Classes apparentées et qui réalisent des fonctionnalités complémentaires d'un même projet sont groupées au niveau d'un même ensemble nommé « package » en Java (désigné en français par paquetage).

Pour chaque composant réalisé, nous développons un paquetage serveur encapsulant ses services.

IV.A.3.2.3 L'organisation des objets du serveur

Nous avons mis en œuvre deux architectures pour structurer les objets d'un même package afin de dégager les services du logiciel encapsulé:

- Une organisation ascendante.
- Une organisation éclatée.

IV.A.3.2.3.1 Une organisation ascendante

Le schéma de la figure suivante montre l'organisation ascendante des classes d'un package.

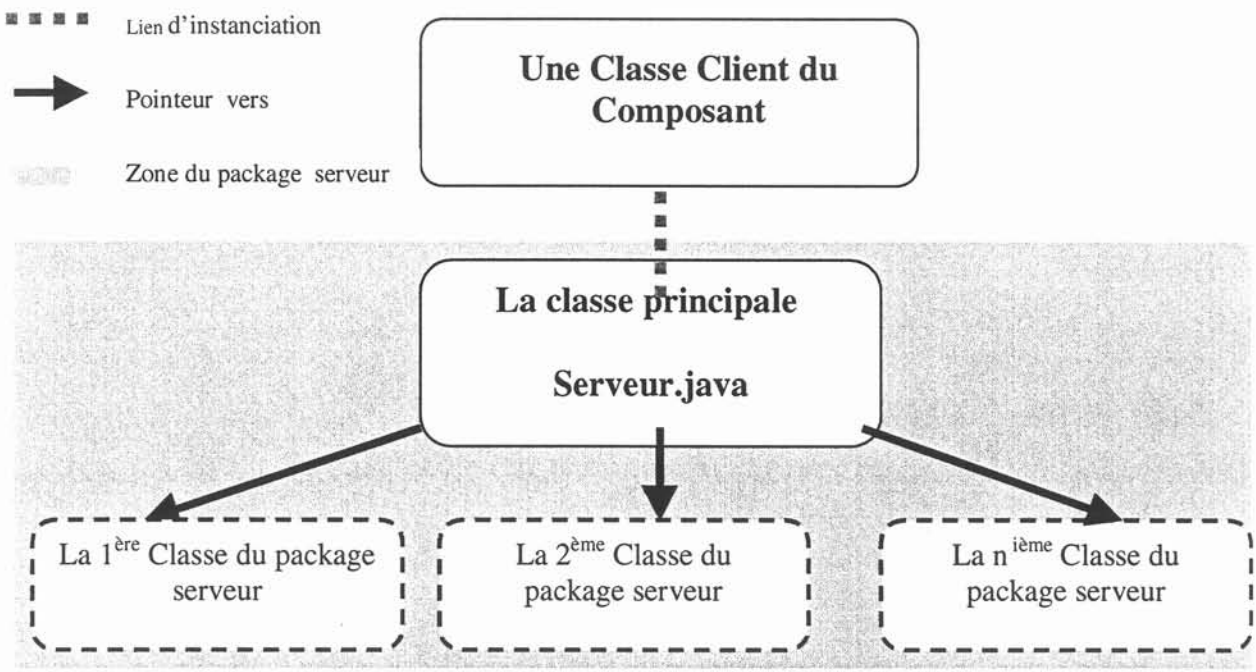


Figure IV-5 Organisation ascendante des Objets du serveur

Avec cette organisation, un ensemble minimal des services du composant est accessible directement à partir de la classe principale Serveur. Comme le montre ce schéma, la classe principale pointe vers l'ensemble des classes du package.

Le concepteur ne manipule de l'extérieur (à partir de la classe Client) qu'une instance de la classe principale. En pratique, si une instance de la classe Serveur est créé, toutes les instances des classes utiles à la définition de l'objet de Conception sont générées automatiquement mais non accessibles au Client de l'extérieur (elles correspondent à des classes protégées).

IV.A.3.2.3.2 *Une organisation* éclatée

Le schéma de la figure suivante montre un exemple d'organisation éclatée des classes d'un package serveur.

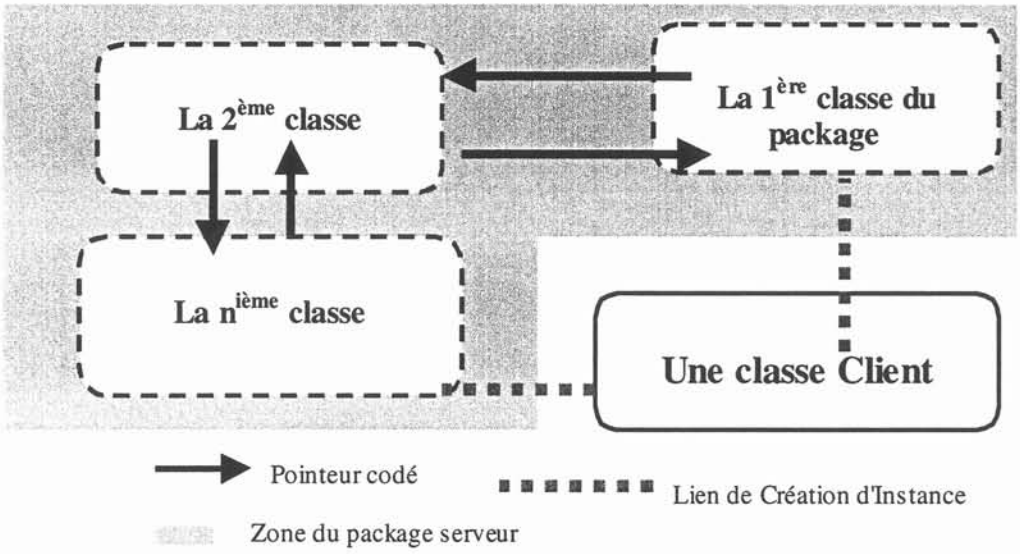


Figure IV-6 Une Organisation éclatée des classes du package serveur

Nous nous sommes inspirés pour cette nouvelle organisation du modèle présenté dans [Ati99]. Les services du composant ne sont plus concentrés au niveau d'une classe principale, c'est pourquoi nous parlons d'organisation éclatée. En les répartissant sur les différents Objets, nous garantissons plus de transparence de l'architecture Objet que nous avons implantée. Et donc nous donnons plus de flexibilité au développeur s'il souhaite accéder aux Objets élémentaires du package.

IV.A.3.2.3.3 Comparaison de l'organisation éclatée et de l'organisation ascendante des objets

La première organisation était la plus adaptée pour centraliser les services du composant alors que la deuxième accorde plus de flexibilité à l'utilisateur afin de manipuler les différents Objets du composant de l'outil.

En verrouillant l'accès aux Objets du package autres que la classe principale dans une organisation ascendante, nous garantissons la cohérence de l'application étudiée. Pour étudier un dispositif donné le concepteur (qui joue le rôle de client) n'a à communiquer qu'avec une seule instance.

Inversement, dans une organisation éclatée, il lui revient de gérer et de créer éventuellement lui-même les références vers les différentes instances nécessaires et utiles pour l'exploitation du composant sur son application. D'où le risque d'incohérence de références.

En effet, l'organisation éclatée augmente les chances de créer des confusions de références. Le schéma suivant illustre un cas de figure où le Concepteur manipule des références incohérentes : à partir de la première instance de sa classe Client, il utilise la deuxième instance de sa n^{ième} classe.

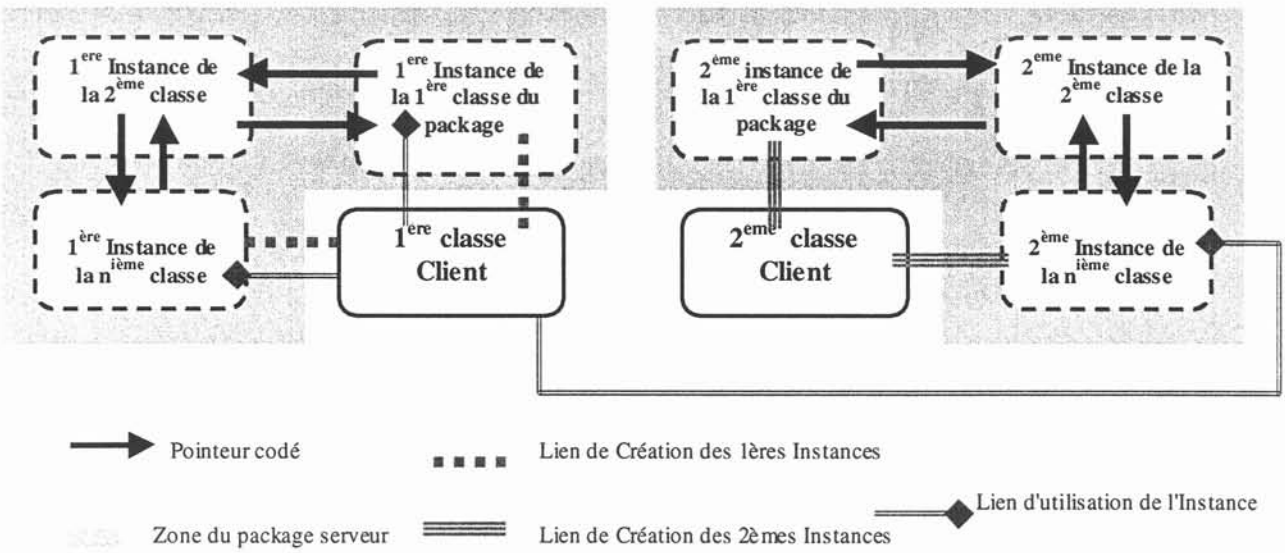


Figure IV-7 Incohérences des références aux classes du package serveur

Par ailleurs, l'inconvénient de l'organisation ascendante est la taille du code de la Classe principale puisque tous les services sont codés à son niveau.

Les deux organisations émanent donc de deux stratégies différentes. Chacune présente des spécificités : nous pensons qu'il n'y pas lieu de trancher entre les deux. C'est pourquoi dans la réalisation des composants de notre prototype de plate forme intégrée, nous avons mis en œuvre les deux variantes (cf. le chapitre V).

IV.A.4 Choix des canaux de communication

IV.A.4.1 Les invocations des méthodes

Le Client communique avec le Composant logiciel à travers les méthodes : il ne peut qu'invoquer les méthodes du composant. Nous voulons prohiber tout affichage au niveau des Objets du Serveur. Toute la communication entre le Composant et son Client passe par le biais des méthodes. Les Objets du package Serveur offrent des méthodes publiques accessibles au Client à partir de la machine supportant le Composant Serveur ou d'une machine différente.

Les méthodes se substituent aux fonctions des logiciels. Pour chaque méthode accessible au Client, le Composant fournit le nom de la méthode ainsi que sa signature (c'est à dire le type et nombre de paramètres d'entrées à fournir par la classe Client, ainsi que le type et nombre de paramètres de sorties).

Les paramètres du dispositif à concevoir sont transmis lors de l'invocation des méthodes. Ainsi le Client fournit aux Objets du Serveur l'information nécessaire à l'exécution du logiciel à travers les entrées des méthodes.

Chaque méthode est exécutée à la demande du Client. Certaines sont programmées pour renvoyer la variable (résultat de l'exécution du code qu'elles renferment), d'autres exécutent des tâches sans renvoi de variable. Sur la Fig. IV-8, nous illustrons un exemple de code écrit pour supporter le dialogue entre un objet Client et un Objet Serveur.

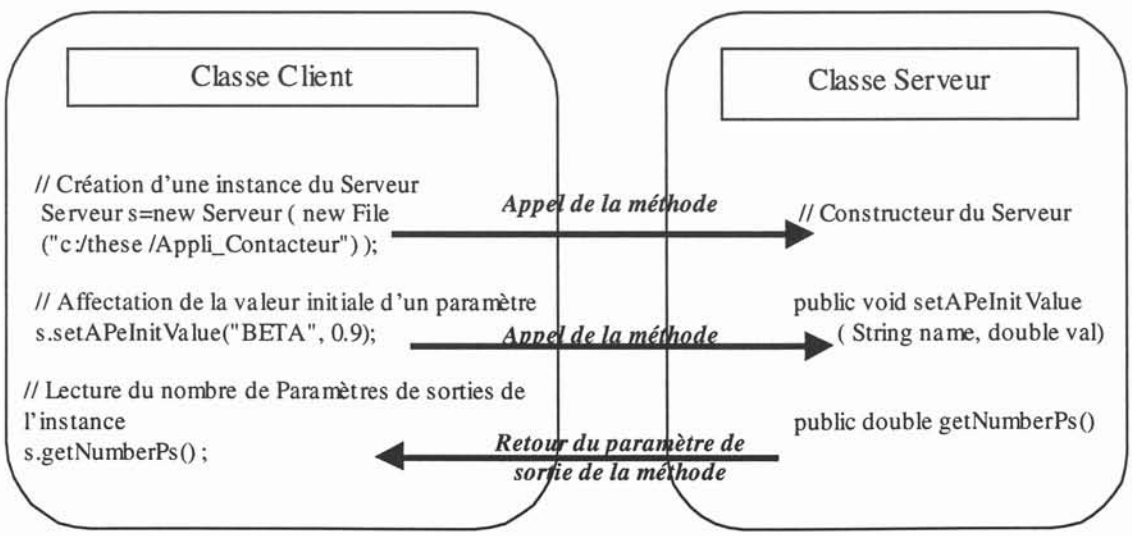


Figure IV-8 Exemple de Dialogue entre un Objet Client et un Composant Serveur

En cas de problème à l'exécution de la méthode où d'invocation incongrue, le Composant renvoie un message d'erreur au Client, selon le mécanisme des Exceptions détaillé dans le paragraphe suivant.

IV.A.4.2 Le mécanisme des Exceptions

Nous avons opté pour le mécanisme de gestion d'erreurs proposé par la programmation en Java. Il s'agit du mécanisme de gestion des Exceptions. En effet, Java offre la possibilité de gérer des évènements exceptionnels. A chaque fois que le programme est bloqué une Exception est générée.

Une Exception est un objet Java à part entière. C'est une classe écrite en Java qui permet de renvoyer un message d'erreur à l'utilisateur pour l'avertir qu'il se trouve dans un cas d'erreur. Java définit un ensemble de règles et de mécanismes qui permettent de manipuler le concept d'Exception [Jav1.2].

Nous avons doté nos classes lors de leur conception de ce mécanisme en respectant dans le code du package des Serveurs les règles de programmation correspondantes.

Toutefois, il nous semble utile de préciser que dans la gestion des Exceptions, deux cas de figures se distinguent : l'erreur à gérer est maîtrisée ou elle ne l'est pas.

Dans le premier cas, l'erreur est prévisible par le programmeur (mauvaises données entrées par l'utilisateur, cas d'utilisation non implanté, etc....) : dans ce cas on va créer le mécanisme qui va gérer cette erreur prévisible et qui donnera l'information complète sur la nature de l'erreur commise et sur les attentes du programme.

Quant au deuxième type d'Exception, le programmeur n'a pas anticipé le cas de figure ou l'erreur n'est pas spécifique à son projet (fichiers corrompus, division par zéro, racine carrée d'un nombre négatif...) auquel cas le message d'erreur renvoyé donnera une orientation de la source d'erreurs tel que l'ensemble des erreurs provenant du système d'exploitation. Nous citons par exemple le cas de NullPointerException, un message renvoyé au Client s'il invoque une méthode d'un Objet du Serveur qu'il a oublié d'instancier (cf. la figure suivante).

Figure IV-9 Renvoi d'une Exception Système

IV.B. La génération de Processus de Conception

Maintenant que nous disposons de composants logiciels pour la conception, nous pouvons réaliser des processus de conception rapidement avec un minimum de programmation. Le concepteur de processus peut aisément agencer les composants de son package (jeux d'outils) et les assembler selon la stratégie choisie.

La principale contrainte à respecter pour l'intégration des composants est d'établir une méthodologie générique indépendante des spécificités des services offerts par les composants. A tout moment, le concepteur peut modifier un composant logiciel, le supprimer ou en rajouter un nouveau sans que cela n'affecte le reste des composants logiciels utilisés dans la conception et la façon dont ils communiquent.

Au début de cette partie, nous tenons à nous positionner par rapport à STEP et à XML. Par la suite, nous allons développer le concept du connecteur ainsi que le mode d'agencement des composants dans une EDIP.

IV.B.1 Positionnement par rapport à STEP et XML

Tout d'abord rappelons les définitions de ces deux concepts.

STEP (STandard for the Exchange of Product model data) est un standard pour l'échange des modèles de produit entre les outils de Conception. La norme STEP a été établie dans le but de structurer la communication entre les différents acteurs intervenant dans le cycle de vie d'un produit. Il s'agit de définir pour chaque discipline une vue propre dans un langage commun à tous les experts.

La méthodologie de STEP consiste à définir des normes d'échanges des données de description d'un produit indépendamment de son évolution et des particularités des utilisateurs potentiels de ces informations [STEP].

"XML (eXtensible Markup Language) est une syntaxe standardisée de description et de structuration de l'information à l'aide de balises de délimitation" [XML]. XML correspond à un ensemble de règles pour organiser des données en vue d'échanges multi-domaine et multi-systèmes à travers le Web.
Nous rappelons que notre objectif de l'association des composants est double : nous souhaitons à travers les plates formes intégrées de Conception:
(1) Assurer le transfert des données entre les composants,
(2) Garantir le partage de leurs services.

Or ces deux méthodologies offrent essentiellement des formalismes d'échanges des informations ; en conséquence, elles ne remplissent pas la totalité du contrat des EDIP. En effet, STEP et XML ne sont pas adaptés aux partages des services entre les composants et ne sont pas prévues pour structurer un scénario d'interactions entre les outils de conception.

IV.B.2 Le concept de connecteur

Le code développé pour la communication entre deux outils est généralement intégré aux outils de Conception ; donc, il n'est pas possible de le réutiliser pour un autre protocole ou une autre stratégie de conception.

Le résultat de ce travail est un nouvel outil à part entière. On ne peut plus décomposer ce nouvel outil pour retrouver séparément les fonctions de chaque outil de base ; ce mode de couplage des outils est désigné par couplage fort.

Avec la mise en place des composants un nouveau mode de couplage est introduit par opposition au couplage fort : le couplage faible. Le code développé pour la communication n'est pas imbriqué dans les outils : les liens entre les logiciels sont définis et codés au niveau d'un intermédiaire informatique appelé connecteur.

Par définition, un connecteur représente le lien fonctionnel entre les composants [Inria].

Cette connexion de liens est transparente. A tout moment le développeur peut rompre les liens entre le composant et le connecteur pour retrouver des composants opérationnels puisque à la base chaque composant est indépendant. Le couplage réalisé peut ne pas porter sur toutes les entrées et sorties du composant (cf. Fig. IV-10).

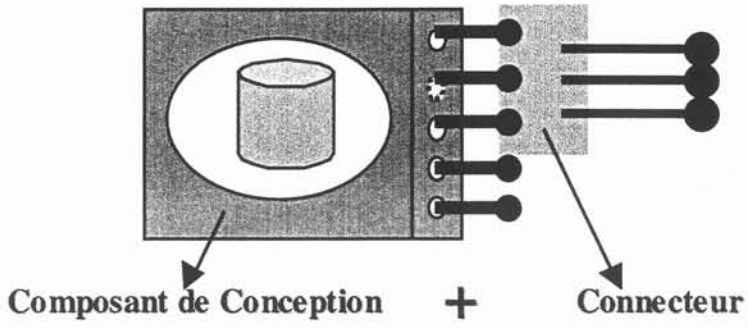


Figure IV-10 Connexion entre le composant et un connecteur

Pour chaque composant de conception, le développeur peut réaliser autant de connecteurs qu'il y a de communications à réaliser. Dans le paragraphe suivant, nous allons étudier les possibilités d'agrégation des composants.

IV.B.3 Choix d'une gestion centralisée

Nous avons réfléchi à deux architectures possibles pour lier les composants logiciels:

- Une architecture de communication composant à composant.
- Une architecture centralisée.

IV.B.3.1 Une architecture de communication composant à composant

Dans cette configuration, les composants communiquent directement. Un canal de communication est mis en œuvre entre chaque couple de composants. Ce canal sera bidirectionnel ; c'est à dire qu'en cas de besoin, il permettra le transfert de données dans les deux sens, du composant n°1 vers le composant n°2 et vice versa. De même, le code nécessaire pour le couplage des services des outils liés sera implémenté au niveau du canal.

En pratique, un canal entre deux composants est formé par deux connecteurs relatifs aux deux composants à interagir et du bout de programme gérant le processus de leur communication, schématisé comme suit :

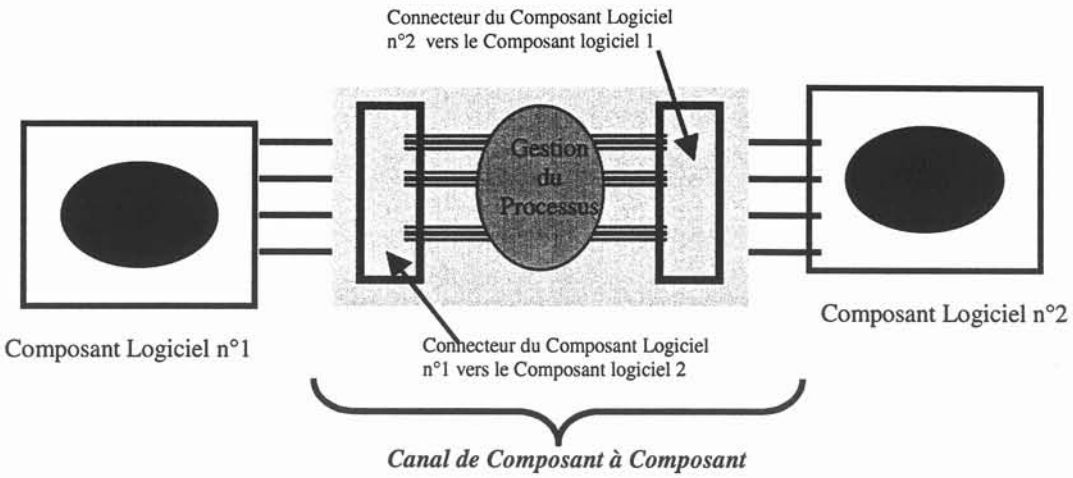


Figure IV-11 Un canal de communication composant à composant

En admettant que le Client ne connaît pas à l'avance les éventuels composants à connecter, nous avons dénombré les canaux à réaliser dans le cas le plus contraignant : chaque composant doit communiquer avec tous les autres composants de la plate forme.

L'architecture à implanter a été schématisée dans le cas de 4 composants à lier sur la Fig. IV-12. Dans ce cas, il faut mettre en œuvre 6 canaux de communication.

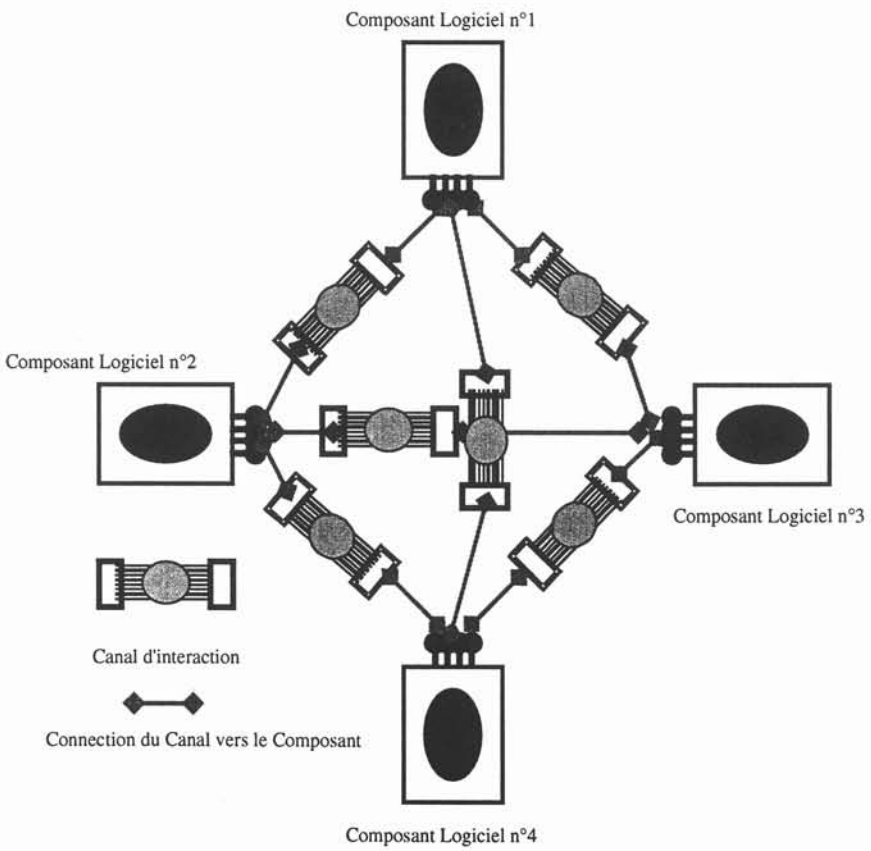


Figure IV-12 Architecture d'une communication composant à composant

Plus généralement, dans le cas où nous cherchons à réaliser une EDIP à partir de n composants, nous avons dénombré $\frac{n*(n-1)}{2}$ canaux de communication. Ceci implique qu'il faut prévoir la mise en œuvre de $n*(n-1)$ connecteurs.

Nous ouvrons une petite parenthèse pour démontrer ce dénombrement. En effet, pour assembler dans une plate forme intégrée n composants selon l'architecture composant à composant :

A partir du 1er composant, il faut implanter (n-1) canaux de communication

A partir du 2ème composant, il faut implanter (n-2) canaux de communication

A partir du 3ème composant, il faut implanter (n-3) canaux de communication

. .

. .

A partir du (n-1)^{ème} composant, il faut implanter 1 canal de communication

En ce qui concerne le n^{ème} composant, il communique déjà avec tous les autres composants.

Au total il faut mettre en œuvre : $[(n-1)+(n-2)+(n-3)+...+1]$ canaux.

C'est la somme des (n-1) premiers termes de la suite arithmétique de raison 1 et dont le premier terme est 1 : elle vaut $\frac{n*(n-1)}{2}$.

IV.B.3.2 Une architecture centralisée

Dans ce cas la communication entre composants logiciels est indirecte : tout passe à travers un noyau central de la plate forme intégrée. Chaque composant communique avec ce noyau central. Ainsi le composant fournit au noyau de l'EDIP les informations utiles aux autres outils et il y puise les informations nécessaires à son propre fonctionnement.

A partir de chaque composant un seul canal de communication est à prévoir (et non plus (n-1) canaux dans le cas d'une plate forme à n composants). De même nous avons représenté l'architecture centralisée à implanter dans le cas de 4 composants sur la Fig. IV-13.

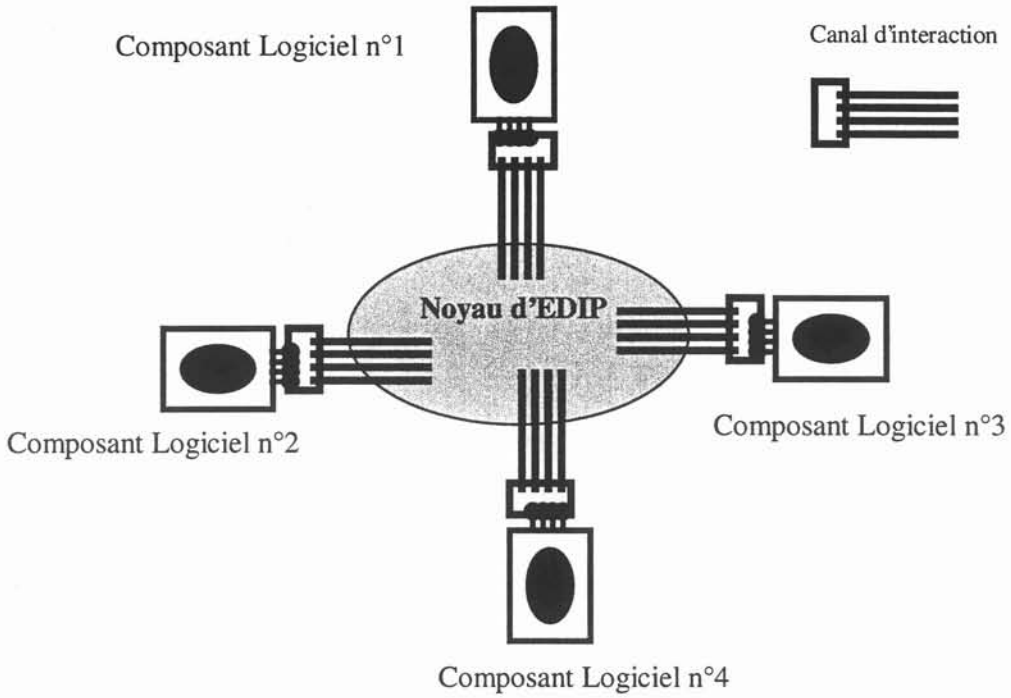


Figure IV-13 Architecture d'une communication centralisée

Dans le cas de l'architecture centralisée, le canal ne renferme que le connecteur du composant vers le noyau de la plate forme comme le montre la Fig. suivante. Contrairement à une connexion composant à composant, le code nécessaire pour le couplage des services des outils qui interagissent sera implémenté au niveau du noyau de l'EDIP et non pas déporté au niveau du canal.

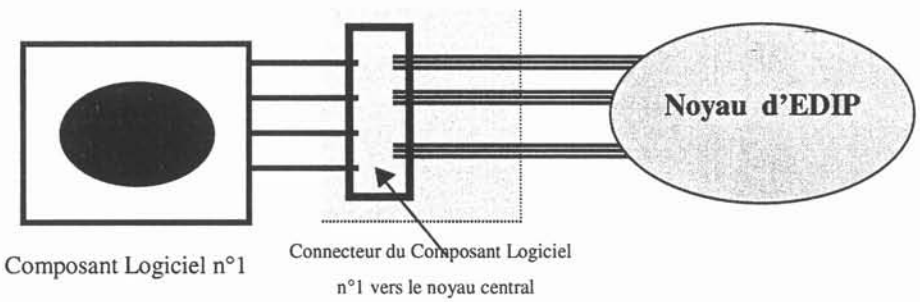


Figure IV-14 Un canal de communication d'un composant vers le noyau d'EDIP

IV.B.3.3 Comparaison des deux architectures

La mise en œuvre d'une architecture centralisée est moins coûteuse que celle d'une architecture composant à composant : il suffit de réaliser n canaux au lieu de $\frac{n*(n-1)}{2}$. Elle est aussi plus adaptée à la génération de processus de Dimensionnement.

Si le Client opte pour une architecture composant à composant, à chaque fois qu'il souhaite brancher un composant dans son EDIP, il est amené à éditer les liens entre le composant en question et tous les autres composants avec lesquels il est susceptible d'échanger des données et des services. Inversement, ces liens sont à supprimer dans le cas où il souhaiterait retirer un composant de conception de son EDIP. En pratique, il doit préciser les paramètres à transférer du composant vers les autres.

Par ailleurs comme le concepteur est entrain de spécifier son produit, il est probable qu'il change la topologie de l'objet de conception à l'extérieur de l'EDIP (cf. le chapitre III), et que certains paramètres du dispositif soient supprimés et d'autres rajoutés. En conséquence, le concepteur doit refaire l'édition de liens entre tous les composants qui les mettent en jeu.

Par contre si le Client opte pour une architecture centralisée, chaque composant établit un dialogue avec la base centrale pour lui fournir ou y puiser des données et des services. Chaque composant est indépendant des autres. A chaque modification de la topologie, les données du noyau sont actualisées automatiquement sans aucune intervention du concepteur.

Nous pensons que l'architecture composant à composant n'offre pas une grande flexibilité pour la génération de processus et que l'architecture centralisée est plus adéquate pour remplir le cahier des charges des EDIP.

Conclusion

Avec l'approche Composant et Connecteur, développer une EDIP revient à faire de l'agrégation (l'association) des composants logiciels de Conception sélectionnés par le concepteur. Dans le cadre d'une architecture centralisée, l'assembleur de composant développera pour chaque outil le connecteur permettant de le lier au noyau de l'EDIP.

Tout au long de ce chapitre, nous avons développé la méthodologie retenue pour la réalisation de plates-formes de Conception Intégrée en génie électrique. Maintenant que nous avons acquis la méthodologie de l'intégration, nous allons la mettre en œuvre en réalisant un prototype d'une EDIP qui fera l'objet du chapitre suivant.

Prototype d'une Plate Forme de Conception Intégrée dédiée au Génie Electrique

Sommaire du cinquième chapitre:

V Prototype d'une plate forme de Conception Intégrée dédiée au génie électrique

Introduction

Ce chapitre s'articule en quatre grands paragraphes. Dans le premier, nous commençons par introduire le dispositif électrique choisi comme application support tout au long de l'EDIP. Ensuite nous présenterons les logiciels à intégrer dans notre prototype. Dans le troisième paragraphe, nous fournirons tous les détails sur la mise en œuvre informatique pour la réalisation de composants logiciels de Conception. Enfin, nous exposerons le processus implémenté à partir des composants de Calcul, d'Optimisation et de Visualisation réalisés.

V.1 Le dispositif électrique étudié

Tout au long de l'étude pratique, nous avons retenu un exemple d'électroaimant comme application test de notre prototype. Dans l'annexe B, nous fournissons les spécifications techniques de ce dispositif électrique.

Cette application a été utilisée dans la thèse [Wur96] pour valider la méthodologie de Pascosma. A priori le dimensionnement d'un tel dispositif électrique ne risque pas de poser des problèmes d'ordre physique.

L'objectif du dimensionnement de cet électroaimant est de réduire sa masse afin qu'il soit le plus léger possible tout en fournissant une force d'appel de 1 N. La valeur de l'entrefer entre la culasse fixe et la culasse mobile est imposée à 0,0005 m.

Nous avons choisi cet exemple simple en terme de dimensionnement afin de disposer d'un dispositif électrique avec un nombre moyen de paramètres à manipuler. Le modèle analytique du contacteur présente 17 paramètres.

Notre priorité dans la réalisation du prototype est la validation de la faisabilité d'environnements de Conception Intégrée à base de composants contenants des logiciels hétérogènes et non pas l'amélioration des performances du dimensionnement du dispositif.

C'est pourquoi nous nous contentons, dans un premier temps, de l'application de l'électroaimant élémentaire pour tester les performances du couplage des logiciels à travers le prototype d'EDIP.

Le dictionnaire des paramètres est fourni dans l'annexe B. Le cahier des charges à remplir est le suivant :

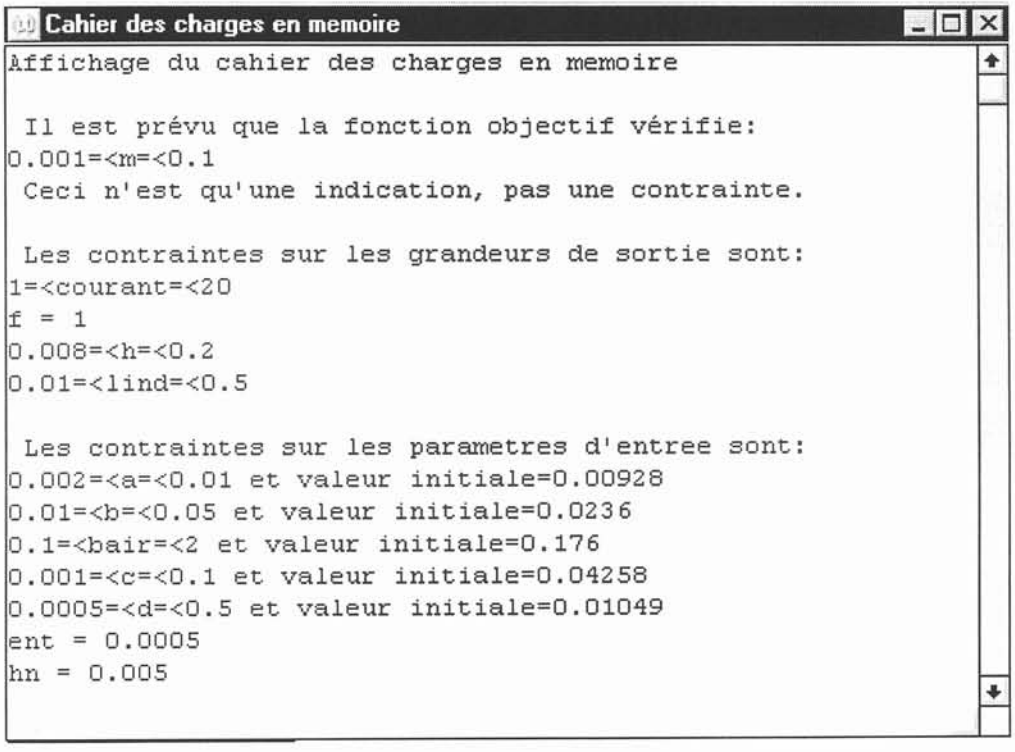


Figure V- 1Cahier des charges pour le dimensionnement

V.2 Les outils à intégrer dans le prototype

Lors de la phase de conception préliminaire, le concepteur en génie électrique peut utiliser quatre types d'outils de Conception :

- Les outils de Calcul
- Les outils d'optimisation
- Les Outils de Calcul par Eléments Finis
- Les outils de Visualisation

Dans notre prototype d'EDIP, nous avons procédé à l'intégration de deux logiciels : PASCOSMA et SOLID EDGE. Nous allons commencer par présenter ces deux logiciels à intégrer.

V.2.1 L'outil de Calcul et d'Optimisation : PASCOSMA

V.2.1.1 Présentation de PASCOSMA

PASCOSMA est un outil développé par Frédéric Wurtz dans le cadre de sa thèse (cf. [Wur96]) effectuée au Laboratoire d'Electrotechnique de Grenoble au sein de l'équipe CDI (Conception Diagnostic Intégrés). PASCOSMA est l'acronyme de **P**rogramme d'**A**nalyse, de **S**ynthèse, de **C**onception et d'**O**ptimisation de **S**ystèmes **M**odélisables **A**nalytiquement.

Deux langages de programmation ont été utilisés pour le développement de PASCOSMA. Le Fortran pour le code «en dur » et le Lisp pour le développement de l'interface graphique. PASCOSMA est uni-plate forme. Dans sa version V2.0, l'outil peut être utilisé sous la plate forme Windows 95 ou 98 (et éventuellement Windows 2000) et sous Windows NT.

PASCOSMA est un outil innovant et unique à notre connaissance. Il permet à partir d'un modèle analytique décrivant le dispositif (représenté par ses équations de fonctionnement) de générer automatiquement un logiciel qui dimensionne et optimise automatiquement ce dispositif en gérant les contraintes de conception. En effet, PASCOSMA offre la possibilité de mener :

- Un Calcul d'Analyse
- Un Calcul de Sensibilité
- Une Optimisation

Nous allons expliciter chaque type de calcul en illustrant le cas de la conception d'un contacteur avec PASCOSMA. Nous admettons que les paramètres d'entrées permettent de décrire le dispositif étudié : parmi ces éléments nous distinguons les spécifications de l'électroaimant (dimensions) et ses conditions de fonctionnement (courant alimentant la bobine). D'autre part, les paramètres de sorties du modèle reflètent les performances du dispositif (force de rappel).

V.2.1.1.1 Calcul d'Analyse

Un calcul d'analyse est mené par l'utilisateur s'il souhaite seulement évaluer les paramètres de son modèle analytique ; ce qui revient à calculer les paramètres de sorties du modèle pour un jeu de paramètres d'entrée.

Une fois le dispositif spécifié, afin de disposer du minimum de données nécessaires à l'analyse de l'électroaimant qui en résulte, le concepteur a tout intérêt à mener un calcul d'analyse. Ce calcul, effectué pour des spécifications et des conditions de fonctionnement données permettra d'évaluer ces performances dans cette configuration.

V.2.1.1.2 Calcul de Sensibilité

Si l'utilisateur souhaite non seulement évaluer les paramètres de sorties de son modèle analytique mais aussi disposer d'informations sur la sensibilité des paramètres les uns par rapport aux autres, un calcul de sensibilité est prévu pour répondre à ce besoin.

Toute l'information sur l'évolution des paramètres est fournie : elle inclut des indications qualitatives aussi bien que quantitatives. Dans la méthodologie PASCOSMA, le support mathématique permettant d'accéder à ces informations passe par les dérivées partielles qui sont calculées symboliquement et programmées automatiquement.

Ce calcul de sensibilité permet d'évaluer la dépendance d'un paramètre de sortie par rapport à une spécification donnée. Le concepteur peut en déduire des informations utiles telles que "De combien vont augmenter ou diminuer les différents paramètres de sorties" ou "A combien est estimée la perte (ou inversement le gain) de performances lorsqu'une de ses spécifications évolue".

V.2.1.1.3 Optimisation

L'utilisateur souhaite obtenir des valeurs optimisées de ses paramètres de sorties en fonction des contraintes introduites et des conditions initiales fournies. Dans la méthodologie PASCOSMA, le support mathématique permettant d'optimiser le modèle consiste à appliquer un algorithme d'optimisation avec gradients VF13AD de la bibliothèque HARWELL [Wur96].

En fournissant un cahier des charges complet et en affectant des contraintes sur les paramètres choisis (intervalles de variations possibles...) le résultat du dimensionnement fournit au concepteur la solution optimale dans cette configuration. Ainsi pour un dispositif donné le concepteur aura les performances optimales pour le fonctionnement prédéfini.

V.2.1.2 *Utilisation de la version 2.0 de PASCOSMA*

Le schéma de la figure ci-dessous montre la méthodologie mise en œuvre dans PASCOSMA.

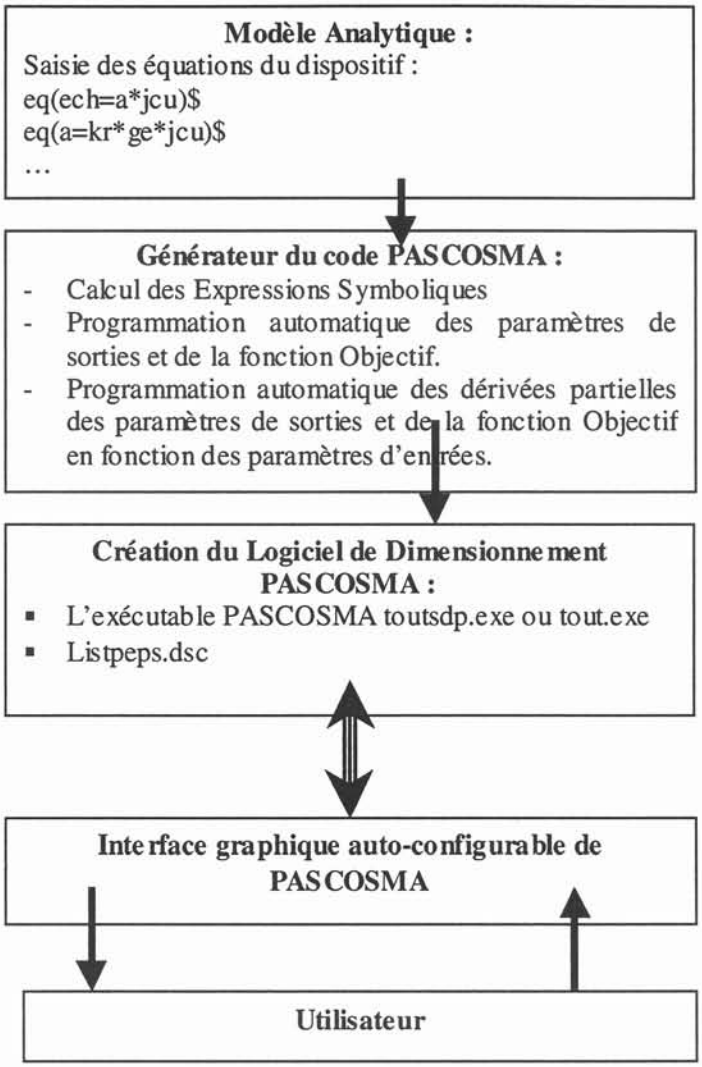


Figure V- 2 La méthodologie de PASCOSMA

Un bref rappel du mode d'emploi de Pascosma s'impose. Afin d'étudier un dispositif sous Pascosma, le concepteur doit générer son application dans l'environnement Pascosma. Pour ce faire il doit :

- Créer l'arborescence de son application sous Pascosma
- Remplir les fichiers de données nécessaires au générateur de code.
- Générer automatiquement le code correspondant à l'application.

Deux configurations du code généré sont possibles. Si le concepteur veut simplement vérifier le modèle de son dispositif (en réalisant un simple calcul des paramètres de sorties en fonction des paramètres d'entrées) il lui suffit de générer le code minimal sans les dérivées partielles. Le fichier exécutable généré s'appelle alors « toutsdp.exe ».

La deuxième situation se présente si le concepteur est sûr de son modèle et qu'il désire utiliser l'application Pascosma pour faire du dimensionnement et de l'optimisation. Dans ce cas, le code complet doit être généré avec les dérivées partielles. Le fichier exécutable généré s'appelle alors « tout.exe ».

Dans les deux cas, un fichier « listpeps.dsc » est généré par Pascosma pour décrire l'application étudiée. Ce fichier listpeps.dsc correspond à un fichier renfermant les informations minimales sur les paramètres du dispositif. Dans la figure Fig. V-3, l'exemple du fichier listpeps.dsc de notre application contacteur est fourni.

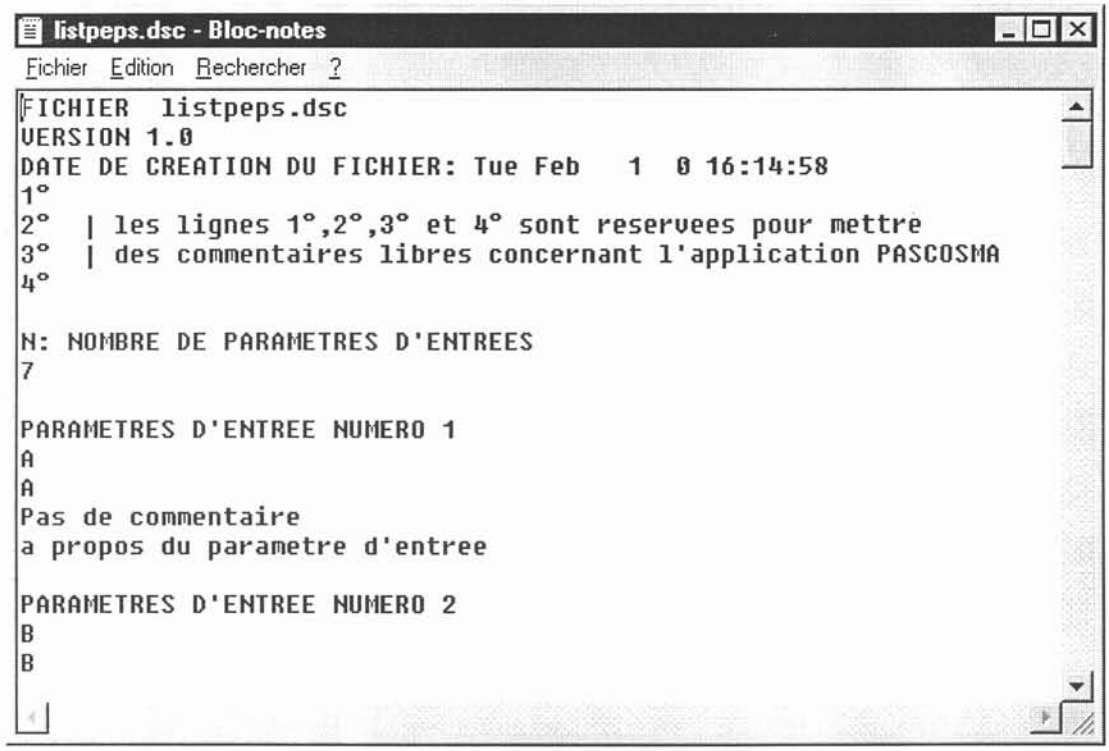


Figure V- 3 Exemple du Fichier listpeps.dsc

Après la génération du code, l'exploitation de l'application se fait à travers l'interface graphique de Pascosma. Cette interface est auto configurable : elle s'adapte automatiquement à l'application PASCOSMA générée grâce au générateur de code.

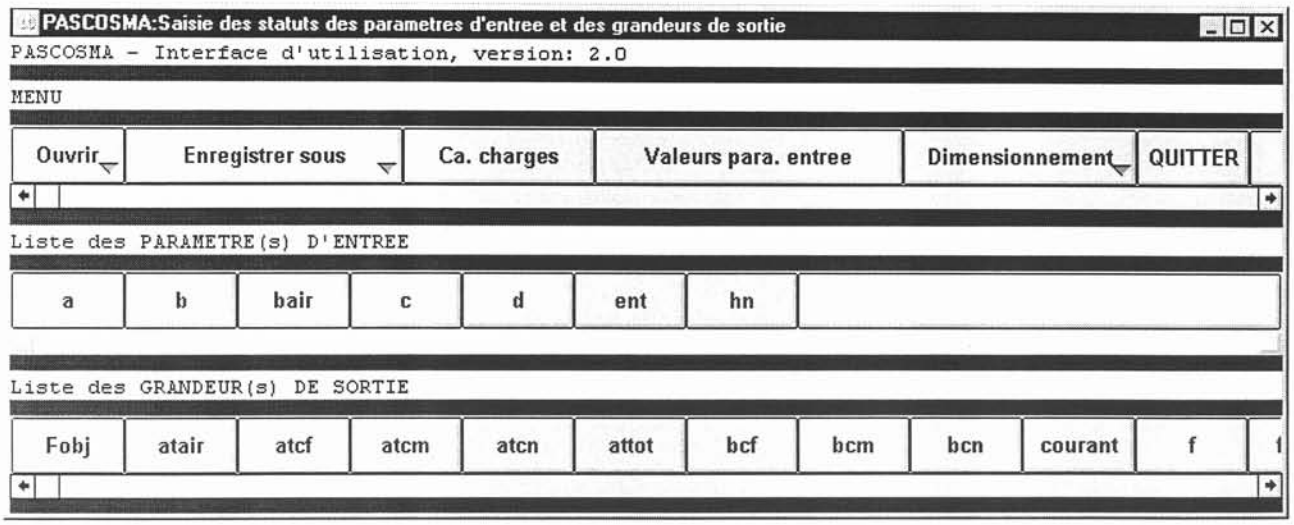


Figure V- 4 Interface graphique de Pascosma sous l'application contacteur

V.2.1.3 Limites de la version 2.0 de *PASCOSMA*

PASCOSMA, en tant que logiciel d'optimisation, présente certaines limites de fonctionnements (influence des valeurs limites des contraintes d'intervalles, problème d'optimums locaux). Ces limites ont été soulignées dans [Wur96].

Dans ce qui suit nous ne nous intéressons pas à ces limites de calcul. Ce sont les limites de l'architecture informatique de l'outil Pascosma qui nous interpellent, en particulier l'imbrication des modules et la non portabilité des applications.

Parmi les contraintes établies pour le développement de Pascosma, [Wur96], figurait la notion d'utilisation conviviale du logiciel de dimensionnement. En conséquence, l'utilisateur accède facilement à toutes les fonctionnalités de l'outil à travers l'interface, aucune compétence ou connaissance particulière en informatique n'est requise. Un protocole de communication entre l'interface graphique et le logiciel de dimensionnement a été défini et est réalisé aux moyens de fichiers. L'ensemble de ces mécanismes a été caché à l'utilisateur.

Ce choix de convivialité mis en œuvre concourrait avec les objectifs de PASCOSMA mais il représente plutôt une restriction à l'intégration de Pascosma dans une EDIP. Vu de l'extérieur, les fonctionnalités de l'outil sont imbriquées avec l'interface.

D'autre part l'exécutable généré par le générateur de code de PASCOSMA n'est pas portable. Les lignes de code des différents programmes automatiques de calcul (calcul d'analyse, calcul de sensibilité et calcul d'optimisation) sont écrites en Fortran et compilées avec un compilateur Fortran compatible uniquement avec l'environnement Windows sous la plate forme PC (Personnal Computer) [Wur96]. En conséquence, les exécutables "toutsdp.exe" ou "tout.exe" ne peuvent être exécuté sous un autre environnement (UNIX ou autre).

V.2.2 L'outil de Visualisation : SOLID EDGE

V.2.2.1 Présentation de SOLID EDGE

Solid Edge est un modeleur 3D qui permet d'effectuer la modélisation de pièces en trois dimensions et la production de plans. C'est un logiciel commercialisé par l'entreprise Unigraphics Solution Incorporation. Nous disposons de la version 4 de Solid Edge.

Solid Edge est spécialement conçu pour l'environnement Windows. Il est doté d'une interface graphique qui assiste le concepteur dans la représentation de ses pièces mécaniques dans son environnement.

V.2.2.2 Les différents environnements de SOLID EDGE

Solid Edge présente trois environnements : un premier environnement pour la modélisation de pièces (Part), un deuxième pour la construction d'assemblages (Assembly), et un troisième pour la production de plans (Draft). Chaque environnement est autonome et il est possible de passer d'un environnement à l'autre.

Afin de représenter une pièce mécanique dans Solid Edge, l'opérateur est amené très souvent à naviguer au moins dans deux des environnements du logiciel. En particulier, dans notre approche, nous allons utiliser les environnements Part et Assembly. Nous allons illustrer leurs utilisations sur l'exemple du contacteur.

V.2.2.2.1 L'environnement Part

L'environnement Part est l'environnement de modélisation de Solid Edge. Il permet au concepteur de construire des modèles solides en 3D par utilisation de fonctions technologiques. Ces fonctions technologiques désignent l'ensemble des opérations à réaliser sur une pièce mécanique. Parmi ces opérations, nous citons les ajouts et les enlèvements de matière, des perçages, etc...

Le processus de modélisation d'une pièce démarre à partir de la création de sa géométrie grâce à une fonction technologique de base, comme l'ajout d'un bloc ou d'un cylindre. La suite de la construction de la pièce sera effectuée à partir de cette base par empilement d'autres fonctions technologiques.

Pour introduire notre application contacteur, nous avons choisi de la décomposer en deux pièces (deux objets Part) : la première étant le circuit magnétique du contacteur alors que la deuxième pièce n'est autre que la bobine du contacteur. A chacune des Part correspond un document dont le nom comporte l'extension .par : nous avons donc le fichier BobineCarre.par et fichier Contacteur.par. La figure suivante correspond à la visualisation de la bobine dans l'environnement Part.

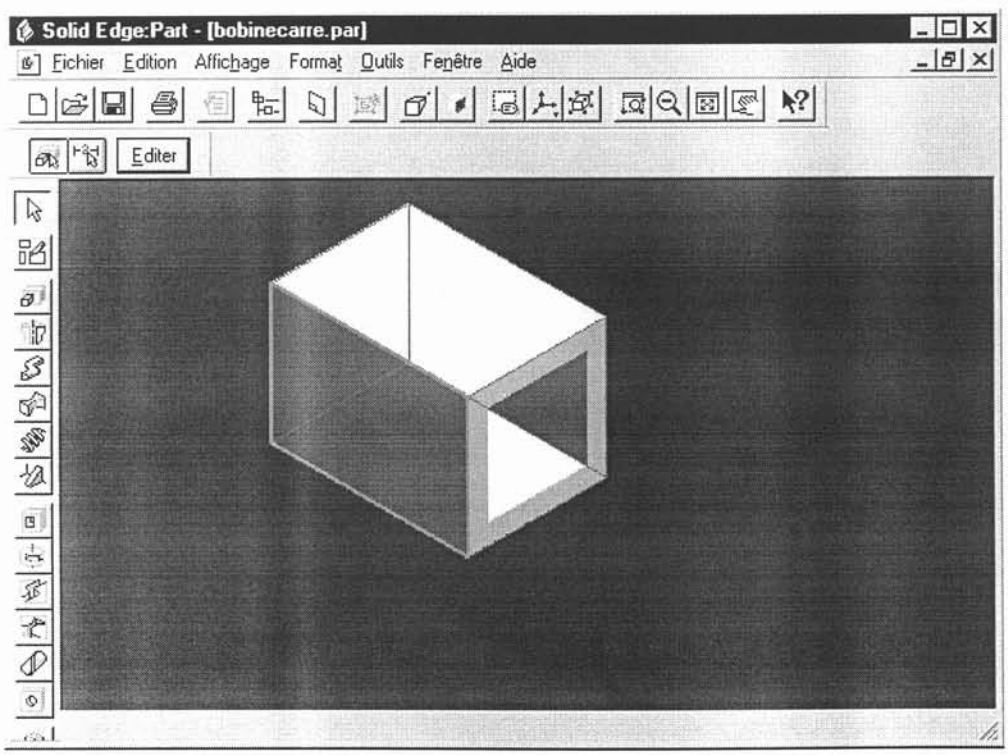


Figure V- 5 La bobine du contacteur dans l'environnement part de Solid Edge

V.2.2.2.2 L'environnement Assembly

En général, les pièces mécaniques de bases (vis, cylindre...) sont conçues en vue d'assemblage. L'environnement Assembly permet de créer des relations associatives entre les différentes pièces grâce à un mécanisme d'assemblage. Il contient des commandes pour lier les pièces entre elles par des techniques d'assemblage naturelles telles que le raccordement et l'alignement. Ces relations sont automatiquement maintenues tout au long de l'exécution du dessin afin de conserver le projet initial. A la géométrie assemblée correspond un document dont le nom porte l'extension .asm.

Par exemple, en assemblant les deux pièces du contacteur, nous avons explicité leurs relations : nous avons précisé que la bobine s'emboîte au niveau de la culasse magnétique en indiquant que hn (la hauteur du noyau de la culasse fixe) n'est autre que le rayon interne de la bobine. Solid Edge maintient automatiquement cette relation entre les deux pièces tout au long du développement du dessin ainsi si hn change la structure de la bobine sera elle aussi modifiée par le logiciel.

La figure suivante montre le contacteur dans l'environnement Assembly. La géométrie illustrée présente l'état initial du contacteur avant tout calcul de dimensionnement : elle ne correspond pas à une solution acceptable du point de vue électromagnétique.

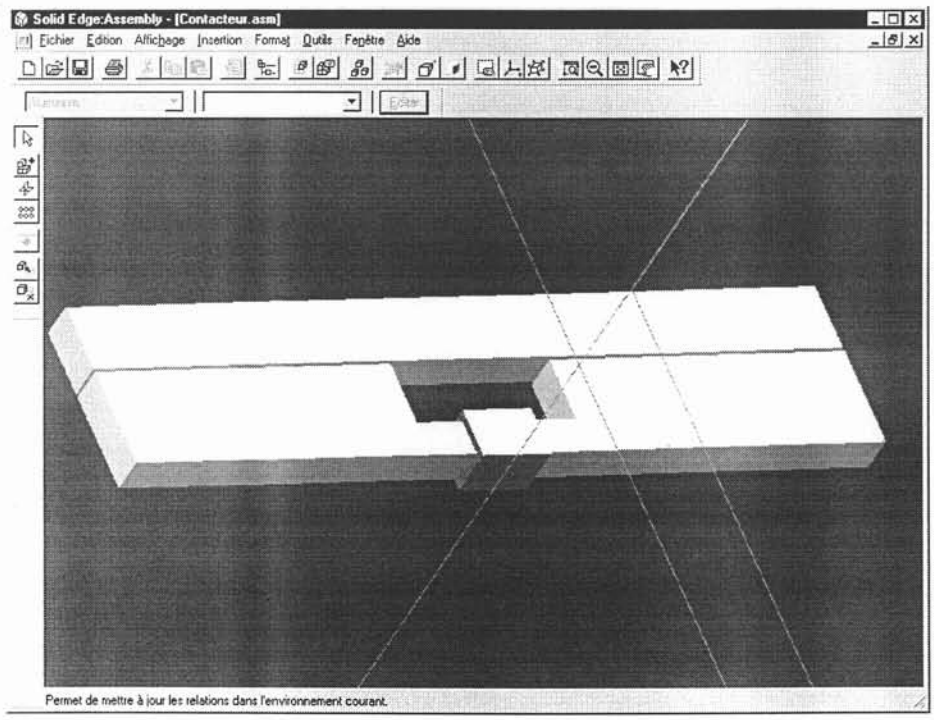


Figure V- 6 Le contacteur dans l'environnement Assembly de Solid Edge

V.2.2.3 Notre utilisation de SOLID EDGE

Dans notre plate forme de conception EDIP, Solid Edge représente le logiciel de visualisation de la géométrie du dispositif à concevoir. C'est un outil de Dessin Assisté par Ordinateur (DAO).

Dans ce paragraphe, nous faisons un bref exposé de la génération d'une application sous Solid Edge. Afin d'étudier un dispositif sous Solid Edge, le concepteur doit créer la géométrie de son application dans l'environnement spécifique à Solid Edge. Nous appliquons ici le principe de décomposition présenté dans le premier chapitre.

En effet, le dispositif est décomposé en plusieurs parties élémentaires. Chacune de ces parties sera créée et paramétrée dans l'environnement Part. Pour ce faire, dans chaque partie, toutes les dimensions seront cotées en choisissant le menu « cote non fixée ». Puis, nous exploitons donc la notion de variables de parties sous Solid Edge en exprimant les cotes en fonction des variables par des formules plus au moins complexes (cf. Fig. V-7 « h_ext = a+hn »).

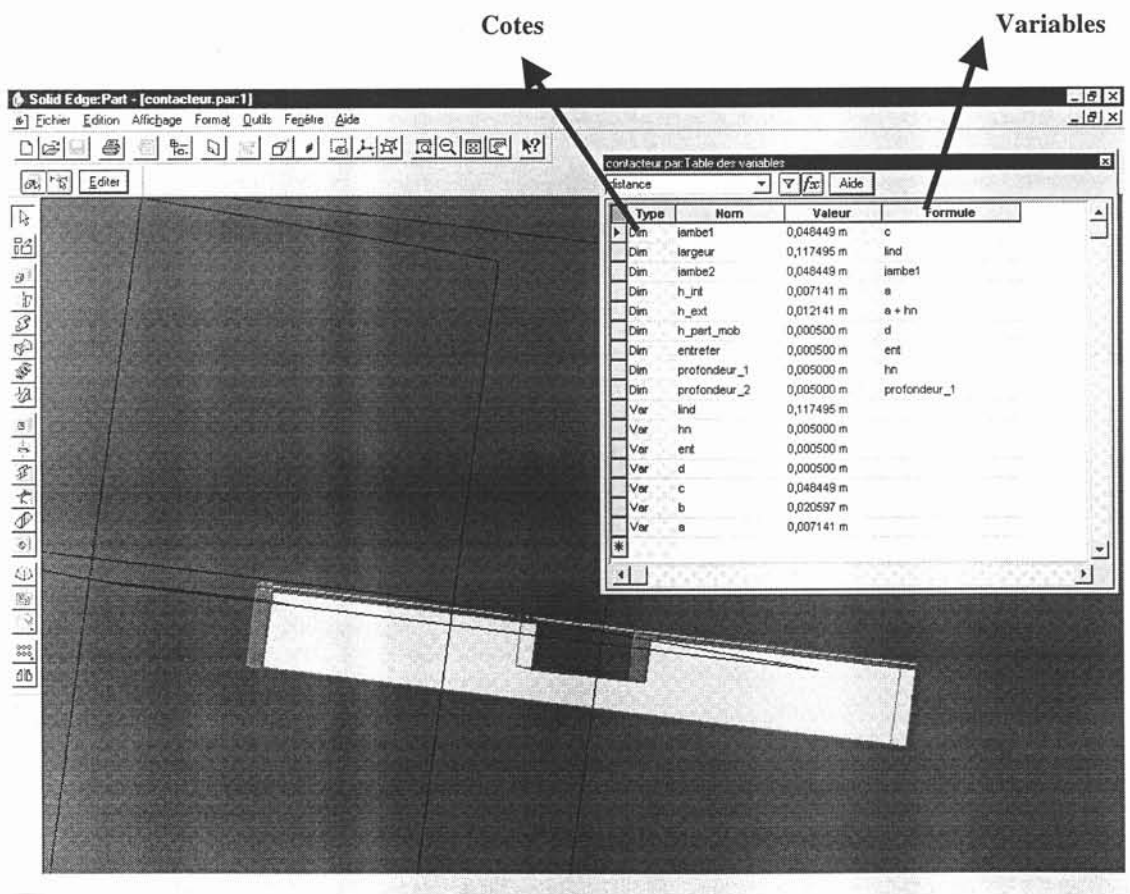


Figure V- 7 Table des variables de la culasse du contacteur dans l'environnement Part de Solid Edge

Cette affectation des variables est la clé de l'interactivité de la géométrie du dispositif dans notre environnement EDIP puisque la modification de la géométrie passe par la modification des valeurs prises par ces variables.

Le Dessin de la géométrie complète est réalisé dans l'environnement Assembly où seront donc assemblées les différentes pièces élémentaires du dispositif en prenant soin de coter les relations d'assemblages. Ainsi des variables d'assemblages seront définies.

V.3 Création des composants

V.3.1 Création du composant PASCOSMA

Pour l'encapsulation de Pascosma, nous pilotons le fichier exécutable de l'application et le fichier de données « listpeps.dsc ». La Fig. V-8 montre la nouvelle architecture de PASCOSMA que nous avons mise en œuvre pour implémenter ses services.

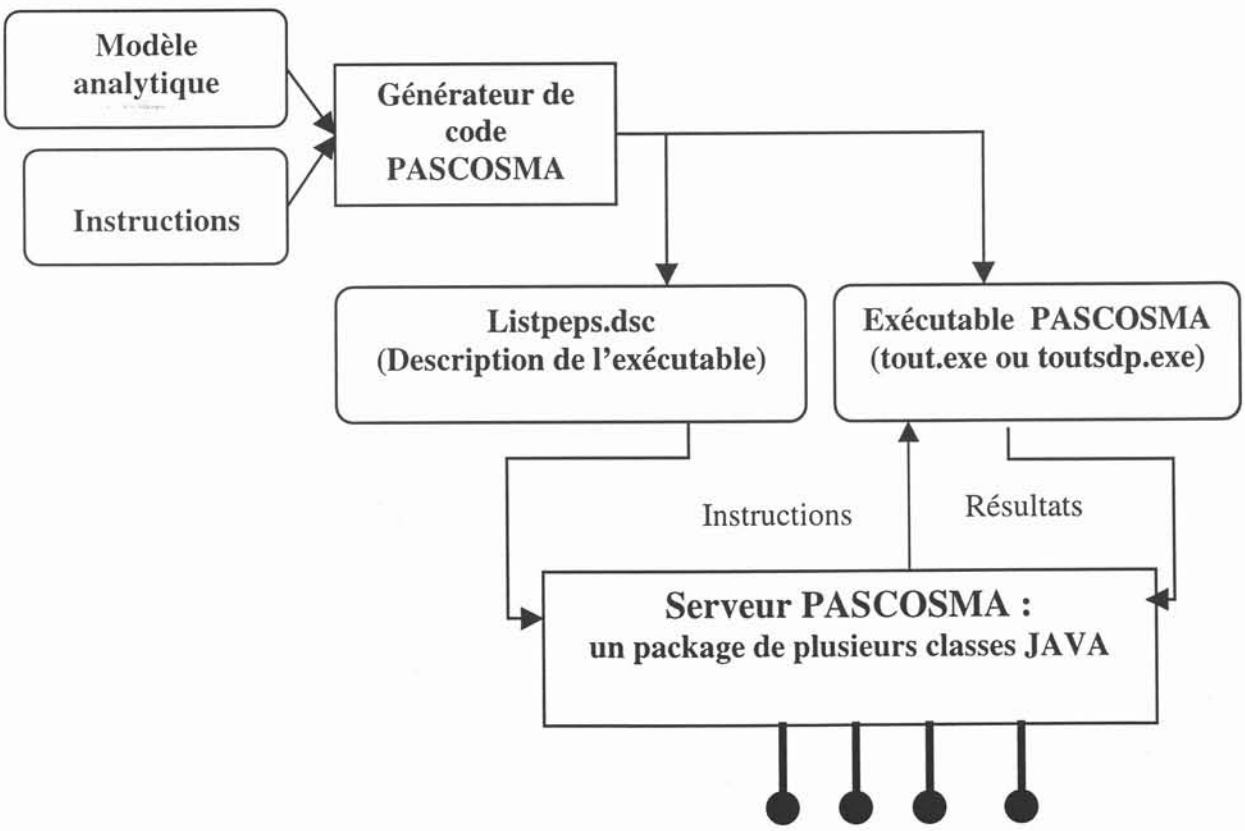


Figure V- 8 La nouvelle architecture de PASCOSMA

V.3.1.1 *Le package serveur Pascosma*

Les classes Java permettant d'encapsuler les exécutables de Pascsosma ont été groupés sous le package serverPascosma. Dans la figure ci-dessus nous avons montré sa hiérarchie complète.

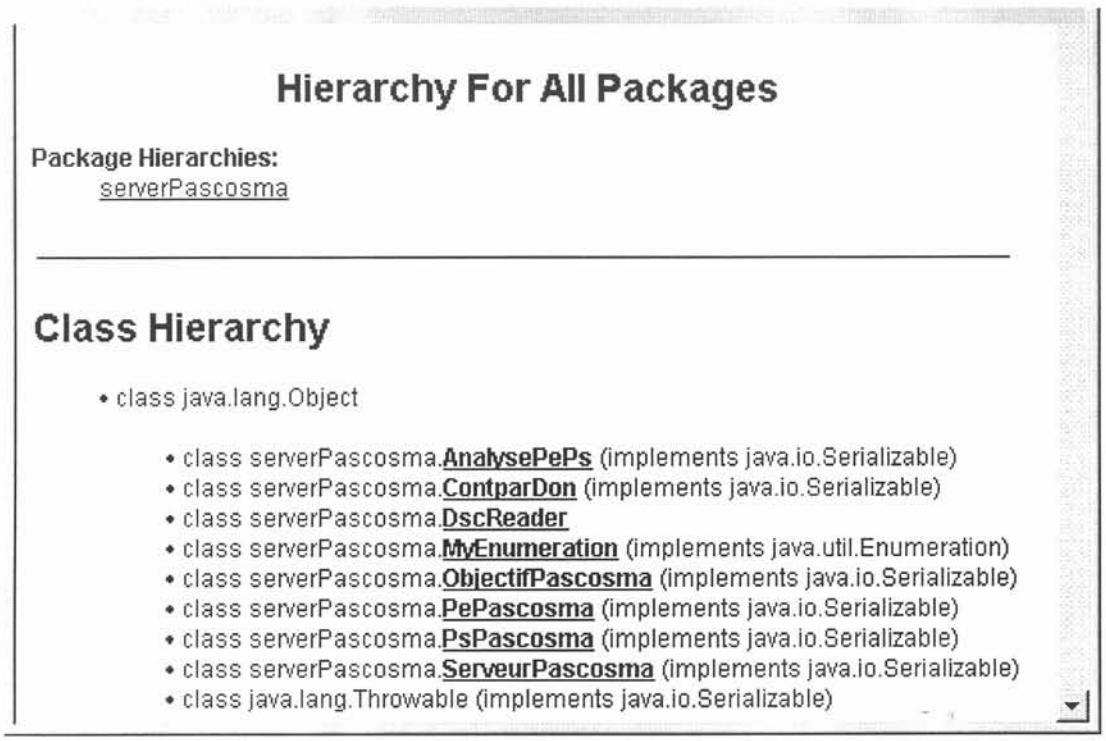


Figure V- 9 La hiérarchie des classes du package serverPASCOSMA

Ces classes ont été organisées selon le modèle de l'organisation ascendante développé dans le chapitre méthodologie comme le montre la Fig. V-10.

Ainsi certaines classes du package serverPascosma ne sont accessibles que par les autres classes du même package ; elles servent uniquement pour la mise en œuvre des services du composant Serveur de Pascosma, donc elles n'ont pas à être utilisées par l'utilisateur.

Par exemple, nous avons développé la classe AnalysePePs.java pour traiter les informations contenues dans le fichier listpeps.dsc. L'une de ces méthodes permettra de créer automatiquement les instances des différents objets paramètres d'entrées et de sorties du dispositif étudié.

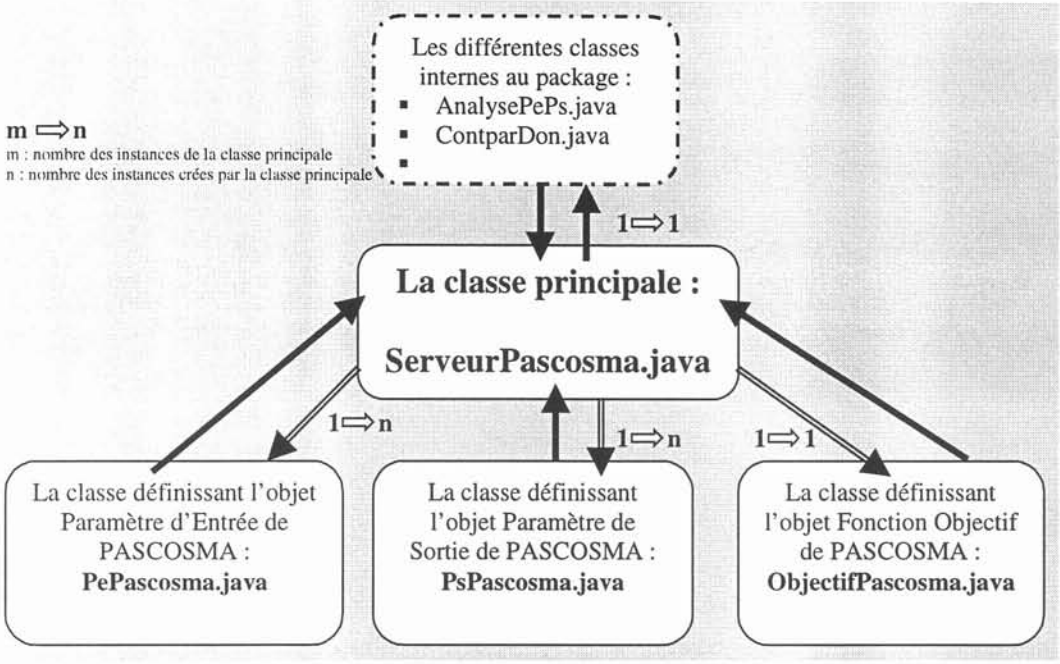


Figure V- 10 Organisation des classes du package serverPascosma

Nous avons donc choisi de rendre accessible l'ensemble minimal des services de Pascosma à partir de la classe principale ServeurPascosma. La figure ci-contre liste quelques-unes unes des méthodes de cette classe. Comme nous l'avons souligné auparavant, le plus grand inconvénient de cette classe est son grand nombre de lignes de codes (elle renferme vers les 3700 lignes de codes).

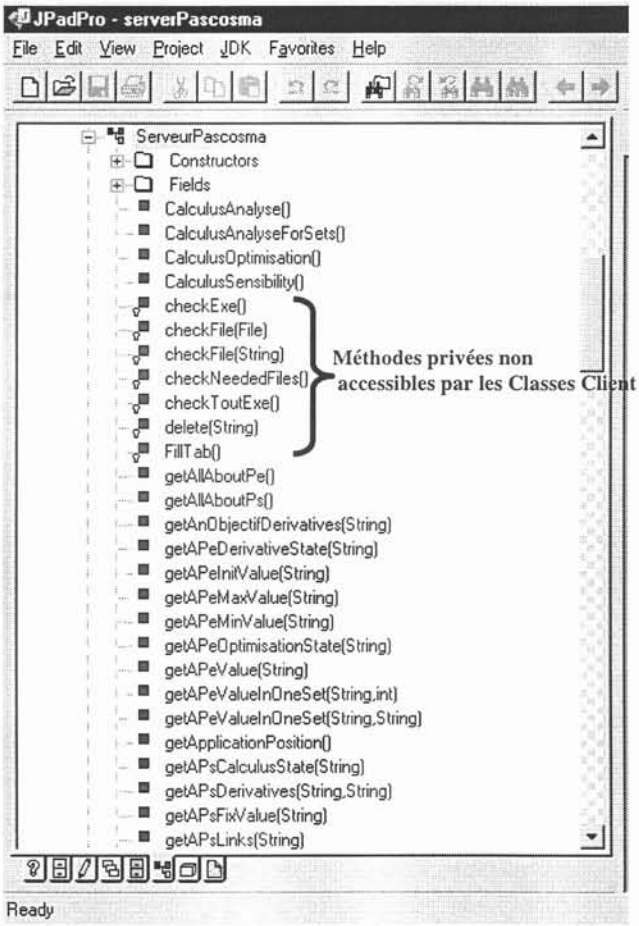


Figure V- 11 Un aperçu des méthodes de la classe principale ServeurPascosma

A partir de la classe ServeurPascosma, nous pouvons accéder à tous les services de Pascosma du calcul simple, du calcul de sensibilité et de l'optimisation.

En ce qui concerne les Exceptions, nous avons recensé les éventuels problèmes à l'utilisation des Objets du package serverPascosma. Pour chaque problème, nous avons implanté la classe Exception correspondante (cf. Annexe A pour la hiérarchie des Exceptions).

V.3.1.2 *Eléments du composant PASCOSMA à distribuer*

En mettant en œuvre l'architecture Objet du package serverPascosma, les services de Pascosma sont accessibles à partir de la classe ServeurPascosma. Pour distribuer le composant logiciel Pascosma il suffit de fournir aux Clients le fichier ThePascosmaServer.zip

Ce fichier zip « ThePascosmaServer.zip » renferme le code compilé de toutes les classes du package serverPascosma. Il groupe l'ensemble des fichiers *.class. Ce fichier est invariant d'une application Pascosma à une autre.

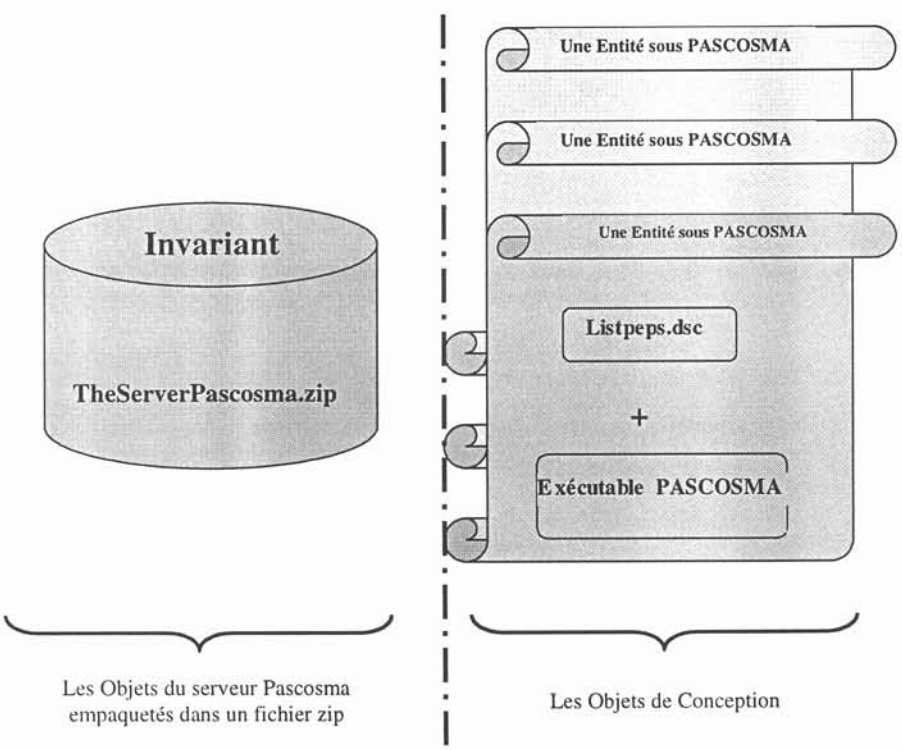


Figure V- 12 Eléments à distribuer pour l'intégration du composant PASCOSMA

En pratique, afin d'étudier un dispositif avec ce composant, il faut que nous disposions du fichier « ThePascosmaServer.zip » et du modèle paramètré sous Pascosma (cf. Fig. V-12). Rappelons que créer ce modèle sous Pascosma revient à générer son entité informatique, qui n'est autre que son exécutable Pascosma (le toutsdp.exe ou le tout.exe) et le fichier listpeps.dsc correspondant.

Le concepteur à partir de son Objet Client instancie la Classe ServeurPascosma en indiquant au niveau de son constructeur le chemin du répertoire qui renferme l'application (Objet de la Conception en cours). La figure suivante illustre cette initialisation par une Interface graphique qui joue le rôle de Client.

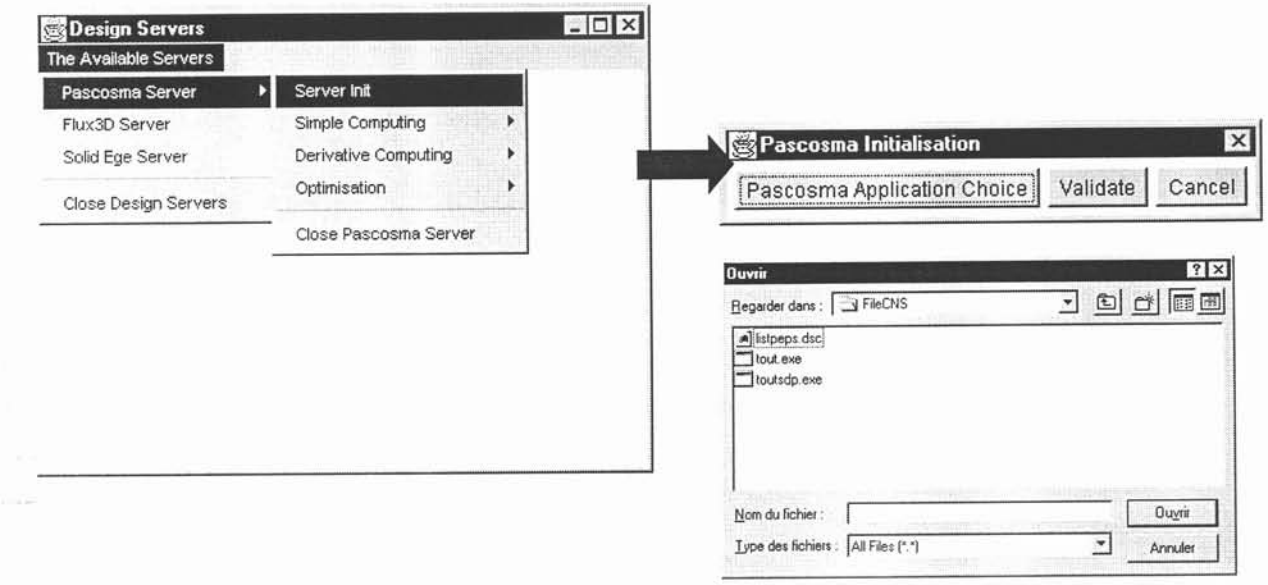


Figure V- 13 Initialisation du ServeurPascosma du composant PASCOSMA

V.3.2 Création du composant Solid Edge

La mise en œuvre de la technologie OLE Automation dans le développement de l'outil Solid Edge, permet de le piloter par l'intermédiaire de langages de programmation tel que le Visual Basic (cf. [Sev4]). C'est pourquoi nous entamons ce paragraphe en exposant l'architecture Objet de l'outil Solid Edge.

V.3.2.1 *L'architecture Objet de Solid Edge*

Afin de manipuler les Objets implantés dans Solid Edge, il faut connaître leur modèle Objet. Le modèle Objet n'est autre que la façon de les organiser et de définir leurs relations de liens. Chaque environnement (Assembly, Draft, Part) de Solid Edge admet sa propre organisation des Objets. La figure suivante montre la hiérarchie des objets implantés dans l'environnement Assembly.

Dans la philosophie OLE, ce sont les objets qui exécutent la plupart des fonctions. Chaque Objet OLE admet non seulement des propriétés qui permettent de le définir mais aussi des méthodes qui permettent d'accéder à ses propriétés et d'agir sur l'Objet.

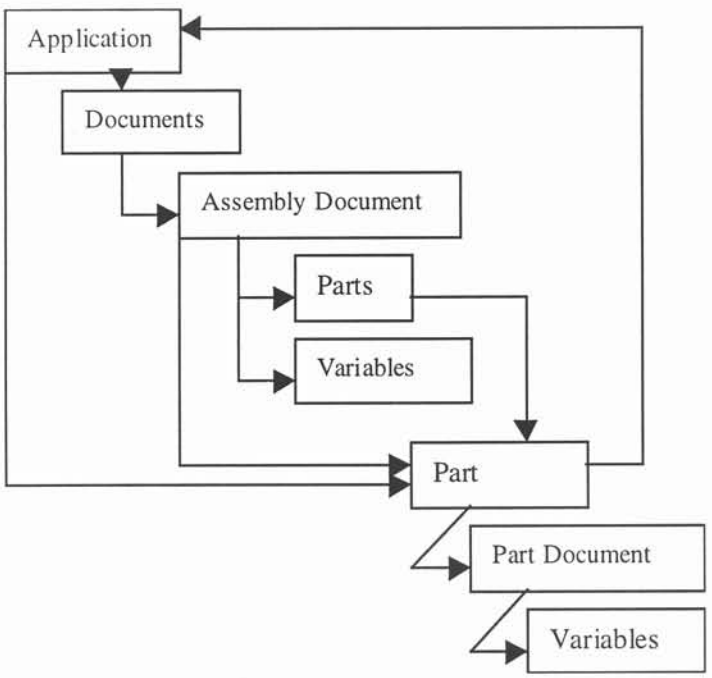


Figure V- 14 La hiérarchie des objets dans l'environnement Assembly de Solid Edge

Par exemple, en dessinant un cercle sous Solid Edge, nous construisons une instance de l'Objet graphique «Circle». Un cercle est une entité géométrique définie par son rayon et sa position dans l'espace. L'ensemble des caractéristiques permettant d'identifier le cercle constituent ses propriétés alors que l'ensemble des actions pouvants s'appliquer sur le cercle constituent ses méthodes.

Une aide en ligne est disponible dans Solid Edge afin d'aider le développeur à accéder à toute l'information sur les méthodes et les propriétés disponibles. Sur le schéma ci-dessous, nous montrons ceux de l'Objet Circle.

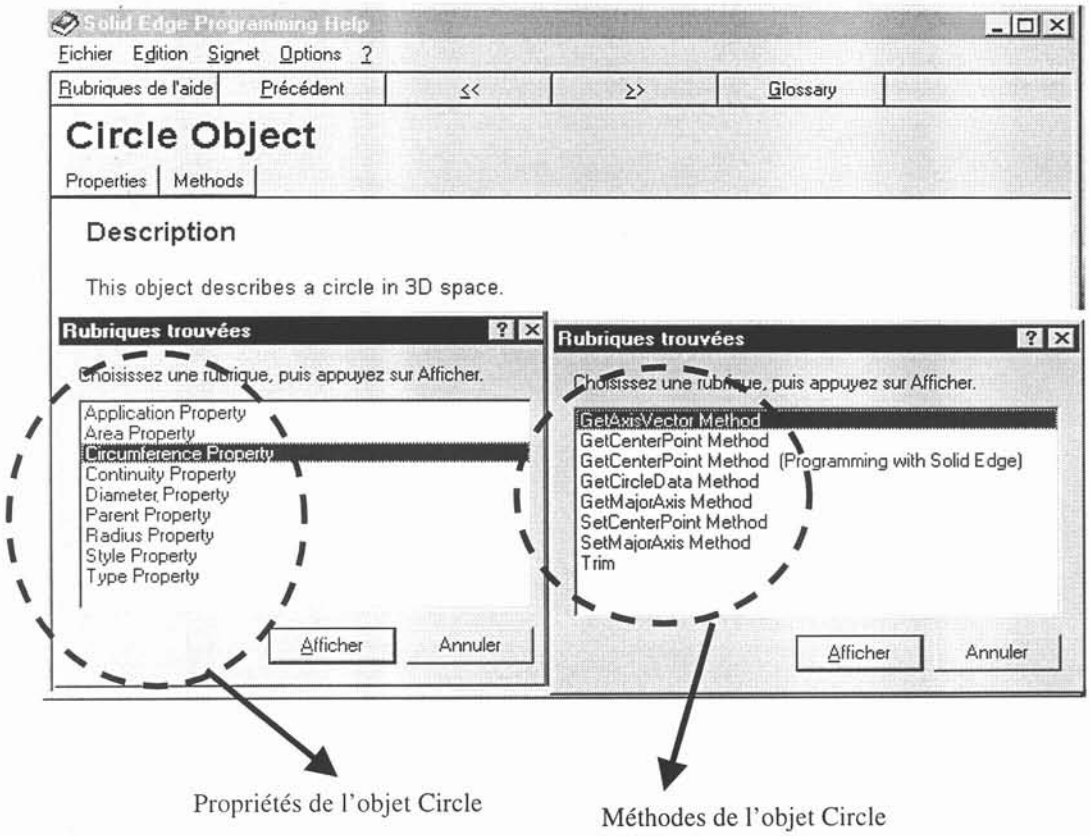


Figure V- 15 Les propriétés et les méthodes de l'Objet Cercle dans l'architecture Objet de Solid Edge

Solid Edge autorise la modification d'objets OLE, par activation de l'application qui a servi à créer les données liées ou incorporées dans le dessin. Il faut donc accéder à l'Objet Application correspondant au dispositif.

La description de chaque Application Solid Edge peut être supportée par quatre différents types de documents : Part Document, Assembly Document, Draft Document et Sheetmetal Document.

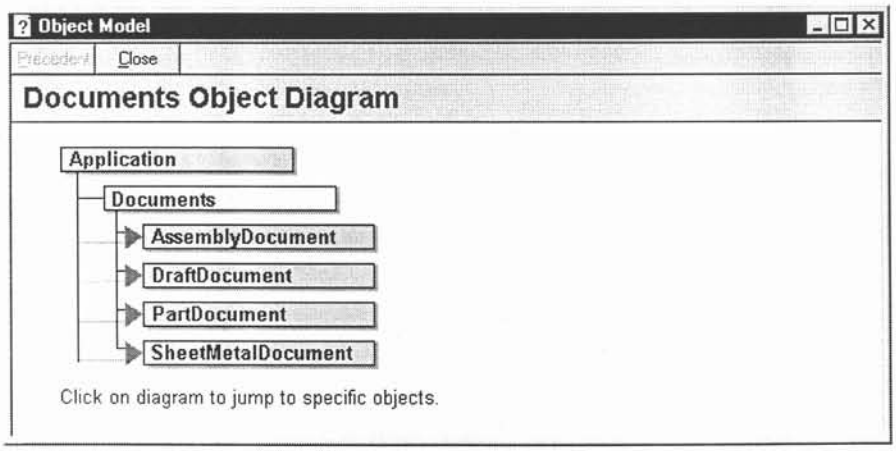


Figure V- 16 La hiérarchie des Documents de Solid Edge

Dans notre plate forme de conception, nous n'avons pas besoin de consulter les plans des dispositifs à concevoir donc nous ne nous intéressons pas au Draft Document ; de même nous ne nous occupons pas de la carrosserie donc nous n'allons pas manipuler les SheetMetal Document. Nous allons décrire nos dispositifs à travers les Part et Assembly documents.

Modifier un dispositif sous Solid Edge revient à modifier les données incorporées dans les documents correspondants à son Application. Avant tout, il faut maîtriser l'accès à l'application générée sous Solid Edge par une application externe. La clé de l'interactivité de Solid Edge avec les autres outils de Conception passe par la réponse à la question suivante : « Comment appeler et piloter l'exécutable Solid Edge de l'extérieur ? ». En particulier dans EDIP, il s'agit d'établir la communication entre Solid Edge avec les autres composants de notre plate forme de Conception écrits par ailleurs selon le standard Java.

Une des solutions consisterait à manipuler l'application Solid Edge à partir d'un programme écrit en Visual Basic. Dans l'Annexe D, nous exposons un programme réalisé en Visual Basic pour assurer le couplage de Solid Edge avec PASCOSMA

Ce programme présente donc un grand inconvénient : son développement est très spécifique puisqu'il ne permet de supporter que le couplage de Solid Edge avec un outil donné, en l'occurrence dans ce cas PASCOSMA.

L'objectif de notre approche est justement de pallier cette restriction en réalisant un environnement de couplage de Solid Edge non plus spécifique à un logiciel, mais générique. Cet environnement permettrait de refaire le plus rapidement possible la même manipulation avec n'importe quel composant de Conception.

V.3.2.2 *Réalisation du composant SOLID EDGE*

V.3.2.2.1 Manipulation des objets OLE en JAVA

Jusqu'à ce stade dans notre rapport, nous avons exposé les possibilités d'intégration de Solid Edge en programmant avec Visual Basic. C'était l'approche exposée dans le «Guide du programmeur avec Solid Edge » [Pro1] et validée par la première expérience de couplage.

Dans nos investigations, nous avons découvert que l'entreprise IBM a mis en œuvre un OLE Bridge. Il s'agit d'une passerelle entre les objets OLE Microsoft et les Java API.

OLE Bridge est un composant logiciel qui met en place un mécanisme permettant d'établir la communication avec les Objets OLE selon le standard Java. Ces objets pouvants être disponibles sur la machine de travail du développeur (qui est en train d'utiliser le OLE Bridge) ou à travers le réseau grâce aux techniques RMI (cf. Glossaire).

OLE Bridge est compatible avec OLE 1.0 et OLE 2.0. Donc, il est compatible avec la version 4 de Solid Edge. Il est donc possible de manipuler les Objets OLE de Solid Edge, selon le modèle Objet Java, à partir de programmes codés en Java en passant par la passerelle OLE Bridge.

A ce niveau deux variantes d'implémentation du code sont possibles comme nous le montrons sur la Fig. V-17.

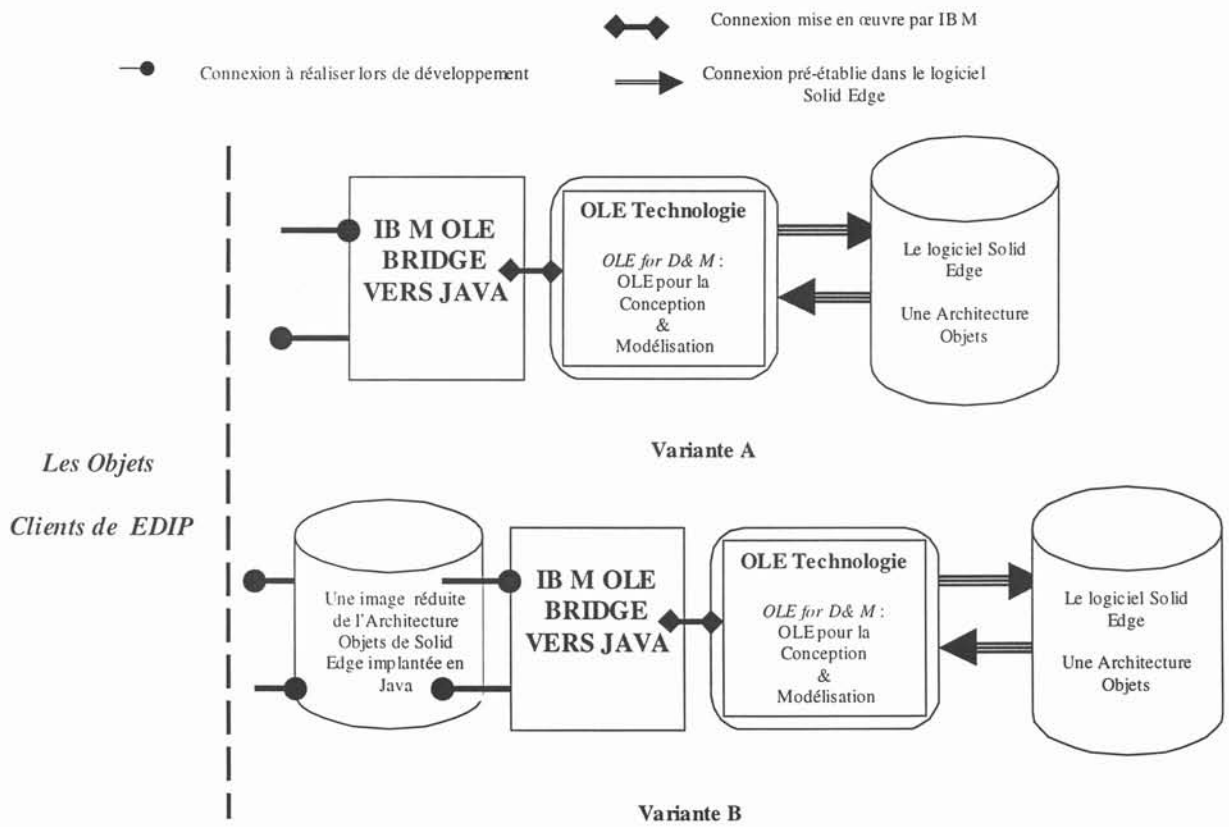


Figure V- 17 Les deux variantes pour la réalisation du composant Objet Java de Solid Edge

Dans la variante A, les classes du Pont OLE-JAVA sont fournies telles qu'elles sans aucune programmation spécifique en vue du projet d'encapsulation de Solid Edge. Ainsi, le développeur du composant Solid Edge aura à manipuler à la fois les objets du Pont OLE-JAVA et les Objets du Serveur OLE de Solid Edge.

Dans la variante B, des classes ont été codées en Java pour assurer la communication entre Solid Edge et les objets du Pont OLE-JAVA. Nous les avons groupés dans un package solidEdgeInJava. Ces classes ne sont pas spécifiques au projet d'intégration de Solid Edge dans notre prototype d'EDIP puisqu'elles peuvent être réutilisées dans le cadre de toute autre communication de Solid Edge avec d'autres composants de Conception ou Interface selon le standard Java ; c'est pourquoi nous avons choisi d'implémenter la variante B.

Dans le package solidEdgeInJava, nous avons transposé l'architecture des Objets telle qu'elle est établie dans le logiciel Solid Edge. Sa hiérarchie est présentée sur la Figure suivante.

Figure V- 18 La hiérarchie du package solidEdgeInJava

Le package solidEdgeInJava joue le rôle d'un connecteur en Java. Vu de l'extérieur le développeur n'a plus à gérer la couche OLE : il peut programmer tous les liens en un seul langage de programmation Java pour communiquer avec Solid Edge. Pour le développement des API du programme de conception interactive, nous n'avons qu'à utiliser les classes de ce connecteur.

Nous nous sommes restreints à dupliquer les objets de Solid Edge nécessaires au transfert de paramètres d'une application développée sous Solid Edge avec son équivalent générée sous un autre outil de Conception (cf. Fig. V-19). C'est pourquoi nous parlons d'image réduite de l'architecture Objet de Solid Edge. Nous avons fourni en Annexe A la documentation des classes java du package solidEdgeInJva.

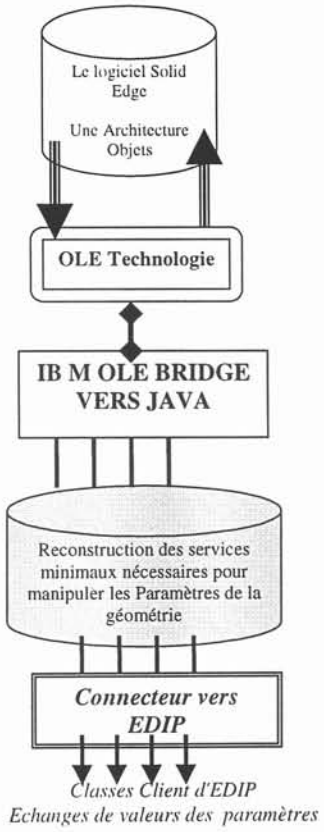


Figure V- 19 Les différentes couches logicielles développées pour le nouveau composant Solid Edge

V.3.2.2.2 Présentation du package serveur Solid Edge

Les Objets Java permettant d'assurer les services du Serveur Solid Edge sont groupés dans le package serverSolidEdge dont nous avons fourni en Annexe A la documentation.

Pour la réalisation du Serveur de Solid Edge, nous avons adopté l'organisation éclatée des classes (cf. Fig. V-20). En pratique, au niveau de la classe Client nommée ici "TestServerSE.", le concepteur instancie trois classes. La première classe ServerSolidEdge doit être instancié explicitement et elle permet de définir l' application Solid Edge à étudier.

Implicitement, elle crée une instance de la classe GeometricIdentity définissant la géométrie de travail. D'autre part, pour visualiser une configuration de la géométrie, le concepteur doit créer une instance de la classe GeometricState. S'il souhaite en manipuler plusieurs simultanément il n'a qu'à créer différentes instances de la classe GeometricState.

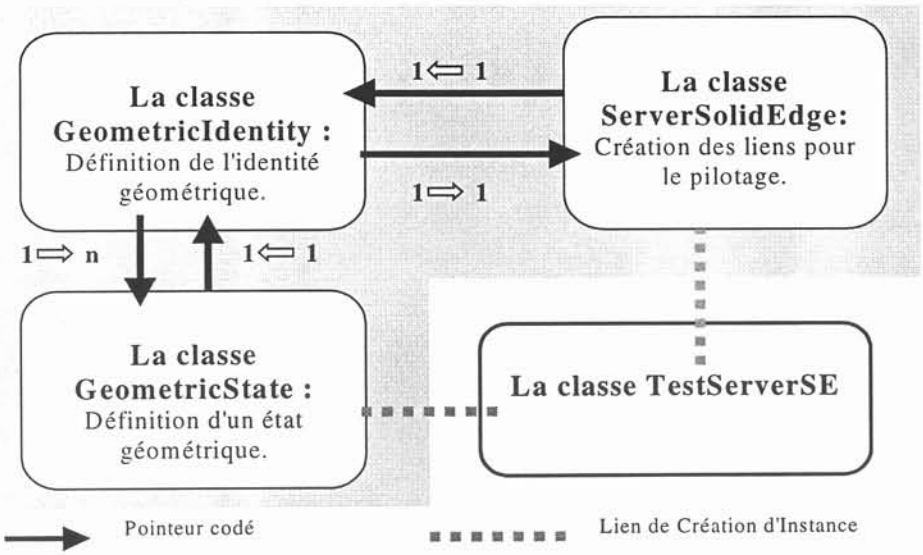


Figure V- 20 Organisation éclatée des classes du package serverSolidEdge

La communication avec SolidEdge est établie à partir de la classe ServerSolidEdge en faisant appel aux Objets du package solidEdgeInJava comme le montre le schéma suivant:

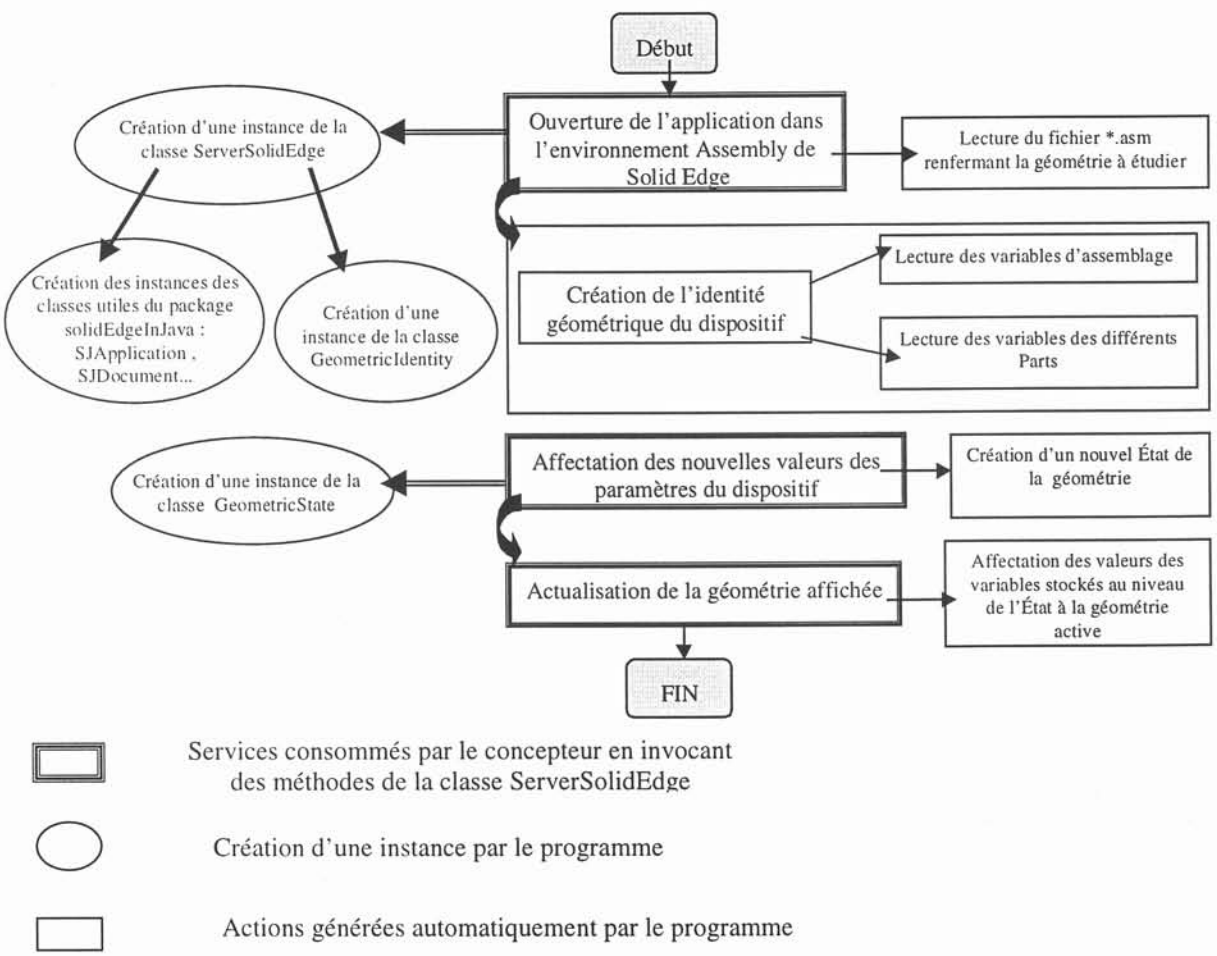


Figure V- 21 Mécanisme de la communication avec Solid Edge

V.3.2.3 *Eléments du composant Solid Edge à distribuer*

Pour distribuer le composant logiciel Solid Edge, il suffit de fournir aux Clients les fichiers :

- "TheSolidEdgeServer.zip" empaquetant les classes du package serveurSolid Edge
- "SolidEdgeInJava.zip" empaquetant les classes du package solidEdgeInJava
- du pont IBM OLE Bridge (fournis par IBM).

Ces éléments sont invariants d'une application Solid Edge à une autre. D'autre part, il faut qu'indépendamment de ce composant, le logiciel Solid Edge soit installé sur la plate forme de travail.

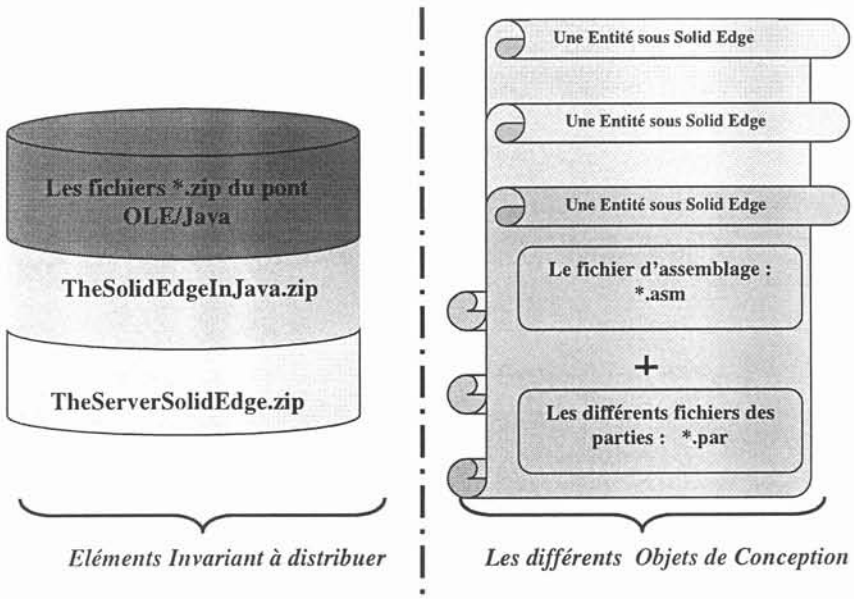


Figure V- 22 Eléments à distribuer pour l'intégration du composant Solid Edge

Rappelons que la génération d'une application sous Solid Edge correspond à la création de son entité informatique définie par son fichier *.asm (l'Assembly Document correspondant) auquel s'ajoutent les différents Documents Parts des pièces élémentaires.

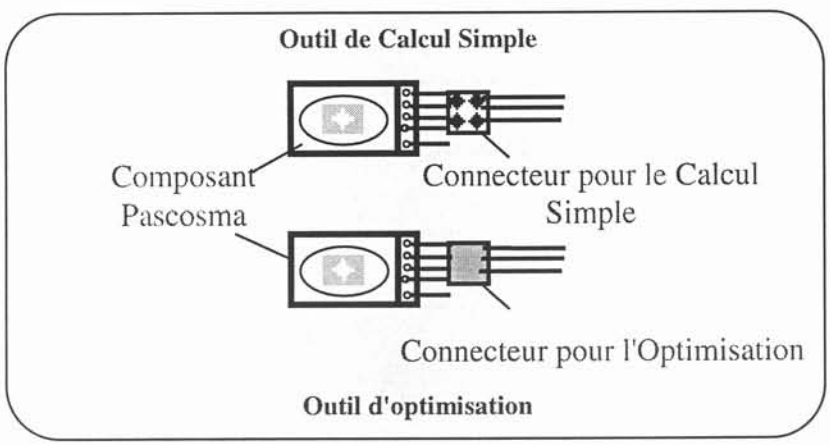
Le concepteur à partir de son Objet Client instancie la Classe ServeurSolidEdge en indiquant dans son constructeur le chemin complet du fichier qui renferme la géométrie assemblée.

V.4 Réalisation d'un processus de conception

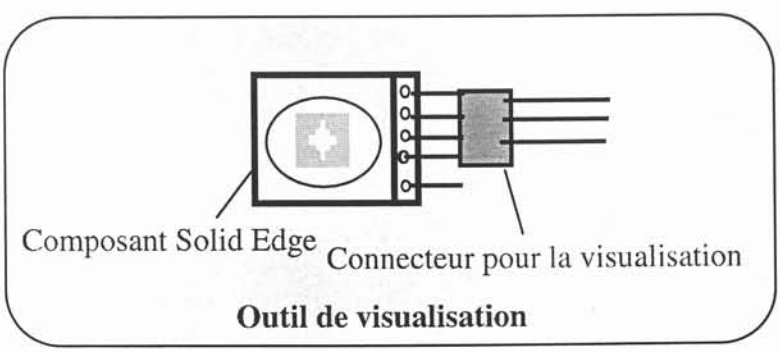
V.4.1 Le concept d'outil

A ce stade nous avons réalisé l'encapsulation de Pascosma et de Solid Edge. Nous disposons actuellement de deux composants de Conception. Pour la réalisation de leur agrégation, nous avons implanté l'architecture centralisée des composants que nous avons détaillée dans le chapitre précédent ; c'est à dire que les composants de Conception vont communiquer à travers le noyau centrale de l'EDIP (cf. Fig. IV- 13).

Dans le prototype que nous allons implémenter nous souhaitons exploiter la fonctionnalité de calcul simple de Pascosma et celle d'optimisation. En conséquence, nous avons réalisé deux connecteurs du composant Pascosma vers notre environnement comme le montre le schéma suivant :



Dans ce même ordre d'idée, nous souhaitons actualiser la géométrie dans l'environnement Assembly de Solid Edge en lui affectant les valeurs évaluées par les composants de calcul (calcul simple ou optimisation) ou fournies par le concepteur. Pour cela, nous avons réalisé un connecteur du composant Solid Edge vers notre environnement comme le montre le schéma suivant :



Chaque composant par l'intermédiaire de son connecteur sera vu dans notre environnement comme un distributeur d'un service principal en vue de la Conception. C'est pourquoi, nous considérons que tout se passe comme si l'association d'un composant à un connecteur spécifique revient à la mise en œuvre d'un outil d'aide à la Conception spécifique (cf. chapitre II). Par la suite, nous utiliserons le terme outil (Tool) pour désigner un couple (composant, connecteur).

V.4.2 Description globale de l'EDIP réalisée

L'image suivante donne une vue globale de la plate forme développée:

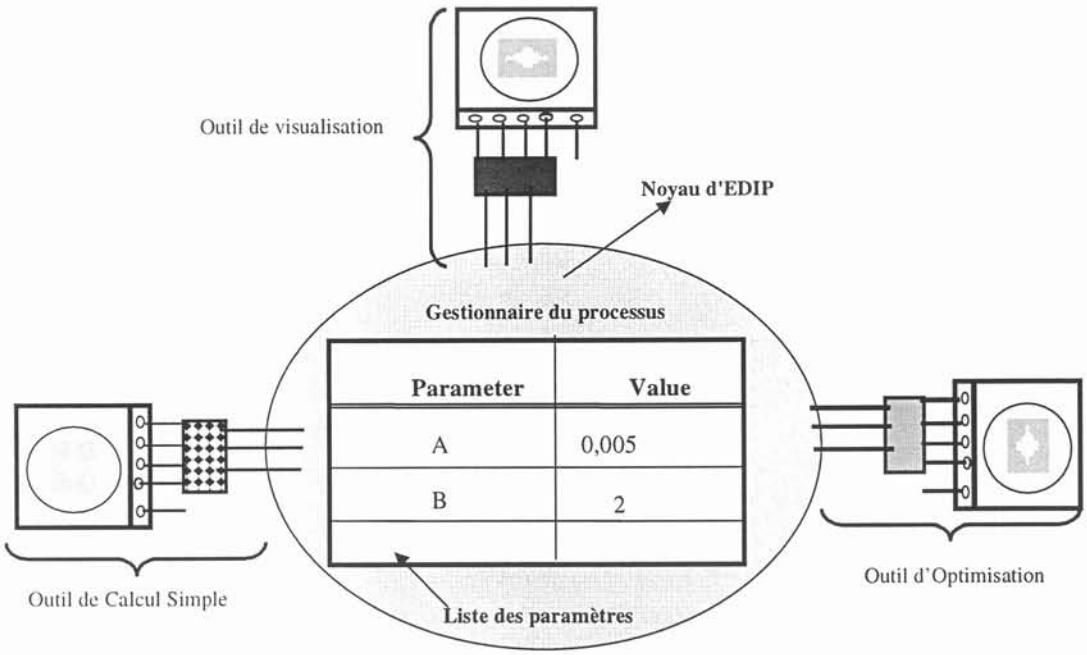


Figure V- 23 Image globale de l'environnement

Nous avons choisi de centraliser les informations sur le produit à concevoir au niveau du noyau. En pratique, pour chaque paramètre du produit, nous stockons dans une base de données centrale son nom et sa valeur prise en cours du processus de conception. Tant que la valeur n'a pas été évaluée par un composant ou fournie pas le concepteur, elle est initialisée à la chaîne de caractères "NotANumber".

En branchant un outil au prototype, l'ensemble des variables de son modèle produit est ajouté à la base de données centrale. Si une variable du même nom existe déjà dans la base, la variable en question ne sera pas dédoublée mais sera ajoutée à la liste des variables partagées en indiquant les noms des composants qui la partagent.

Au moment de l'intégration d'un outil donné, ses différents paramètres manipulés dans son modèle produit sont répertoriés en trois types : Inputs, Outputs ou InOutputs. Cette typologie définie les règles d'échange des valeurs des paramètres entre l'outil et la base de données centrale, qui sont les suivantes:

1	**Si**	*(L'outil lit des valeurs de paramètres à partir de la base centrale)*
Alors		(Ces paramètres sont qualifiés en tant que Inputs)

2	**Si**	*(L'outil modifie des valeurs de paramètres dans la base centrale)*
Alors		(Ces paramètres sont qualifiés en tant que Outputs)

3	**Si**	*(L'outil lit des valeurs de paramètres à partir de la base centrale et les modifie après son exécution)*
Alors		(Ces paramètres sont qualifiés en tant que InOutputs)

Nous avons prévu de visualiser cette classification au niveau de l'interface de l'EDIP réalisée, comme le montre la Fig. suivante (Fig. V-24).

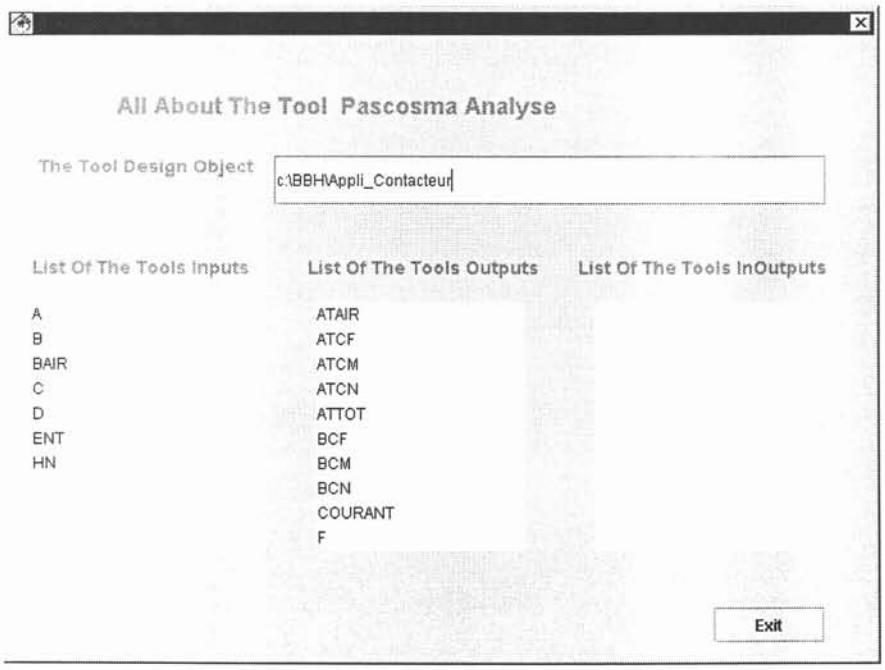


Figure V- 24 Les paramètres du contacteur au niveau de l'outil de Calcul

Toutefois un même paramètre peut être vu différemment par les outils intégrés. Par exemple, certains paramètres géométriques sont à calculer par l'outil de Calcul simple, donc ils sont considérés en tant que Outputs au niveau de Pascosma Analyse alors que ce sont des In/out au point de vue de l'outil de visualisation.

Afin d'établir la communication d'une manière générique entre les outils, nous avons mis en place un programme générique pour:

(1) Coordonner les liens entre les outils et l'environnement (ajout, suppression d'un outil),

(2) Exploiter les services des composants en utilisant les données de la base centrale,

(3) Gérer l'actualisation de la base centrale en cas d'intervention du concepteur (introduction de nouvelles valeurs), ou suite à l'exécution d'un outil.

Le tableau suivant est un récapitulatif de la correspondance entre les concepts définis et les Classes codées pour les concrétiser. L'ensemble de ces classes a été groupé dans un package que nous avons nommé designProcess.

Concept	***Classe***
Connecteur générique d'un composant vers l'environnement	Abstraite "MetaTool"
Connecteur de Pascosma Analyse	"PComputingTool" dérivée de "MetaTool"
Connecteur de Pascosma Optimisation	"POptimisationTool" dérivée de "MetaTool"
Connecteur de Solid Edge	"SEViewTool" dérivée de "MetaTool"
Base de données Centrale	"Device"
Gestionnaire du processus	"ProcessManager"

Pour illustrer l'utilisation de notre EDIP, nous avons décrit dans l'annexe C les différentes étapes d'un scénario d'utilisation des outils pour le dimensionnement de l'application contacteur. Dans ce qui suit nous allons fournir quelques détails sur la mise en œuvre pratique du prototype d'environnement gérant le processus de préconception.

V.4.3 Connecteur générique du composant vers le noyau de la plate forme

Pour mettre en œuvre la connexion entre les composants et l'environnement d'EDIP, nous avons essayé de profiter du polymorphisme de la programmation orientée Objet.

Pour cela, nous avons défini un connecteur générique entre chaque composant et le noyau de l'EDIP. En pratique, ceci revient à spécifier une classe mère définissant les services à mettre en place pour chaque outil comme le montre le schéma ci-dessous.

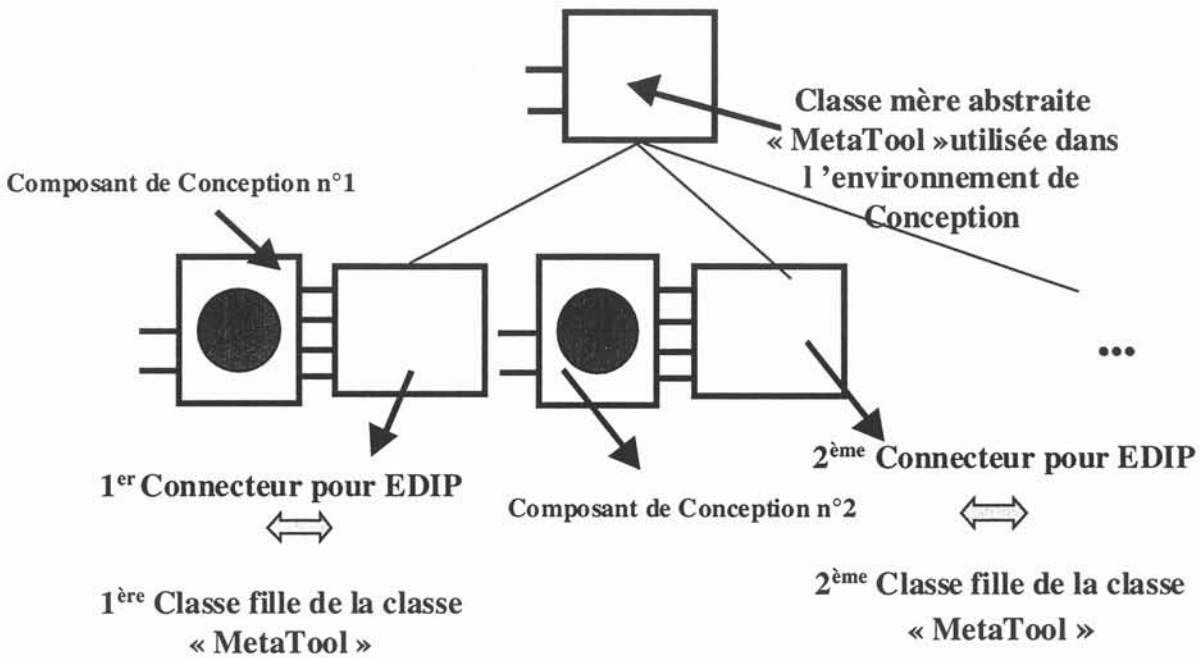


Figure V- 25 Le connecteur générique d'une EDIP

Au niveau de la classe abstraite MetaTool, nous avons indiqué les méthodes que chaque outil est contraint d'offrir afin de permettre son couplage dans notre prototype d'EDIP. La Fig. V-26 les liste.

En effet, chaque outil offre au Client une méthode pour:

- Ecrire et lire son nom : setName et getName
- Ecrire et lire ses Inputs : setInputs et getInputs
- Ecrire et lire ses Outputs : setOutputs et getOutputs
- Ecrire et lire ses InOutputs : setInOutputs et getInOutputs
- Exécuter son service principal : operate

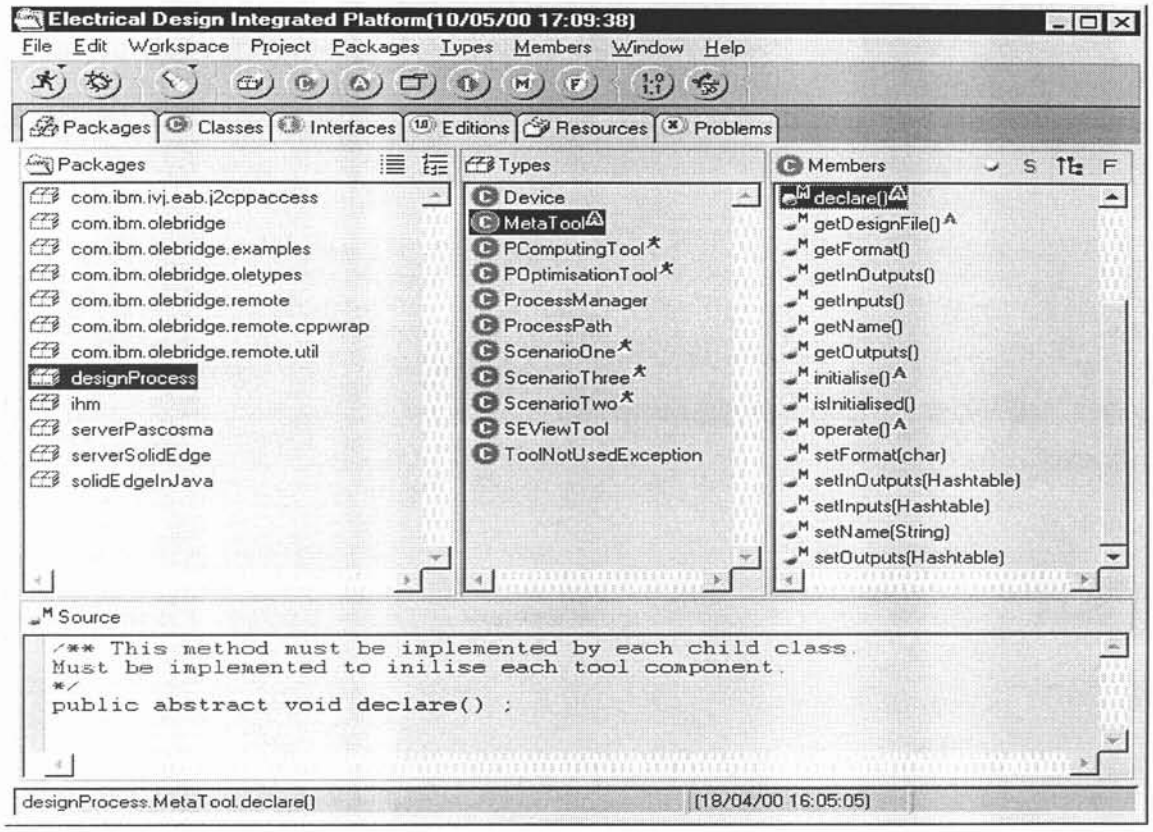


Figure V- 26 Les classes du package designProcess et les méthodes de la classe MetaTool

La méthode "operate" définie la tâche spécifique de l'outil. Chaque composant doit offrir un service principal en vue de la conception. Le contrat de ce service est déclaré en créant la méthode abstraite "operate" de la classe MetaTool. Le code spécifique pour remplir ce contrat est défini au niveau de chaque classe dérivée.

V.4.4 Structure informatique pour le pilotage des composants

V.4.4.1 *La classe ProcessManager*

Pour mettre en œuvre la coordination des liens entre les composants et l'environnement d'EDIP, nous avons recensé les services minimaux pour la manipulation d'outils dans un même processus de Conception et nous les avons imposé en tant que méthodes de la classe ProcessManager.

A partir de cette classe, nous pouvons piloter l'ensemble des composants indépendamment de leurs spécificités. Parmi ces méthodes, figurent l'intégration d'un outil à EDIP (plugATool), la suppression d'un outil d'EDIP (removeATool), l'exécution de la tâche principale de l'outil (operateATool).

Du moment que tous les connecteurs des composants vers EDIP offrent les mêmes services (tous des classes filles de "MetaTool"), il est possible d'implémenter le pilotage à partir des connecteurs d'une manière générique indépendamment des particularités des composants que nous avons choisis d'intégrer dans notre prototype. Ainsi, nous avons pu développer des méthodes génériques au niveau de la classe "ProcessManager".

La Figure suivante schématise la structure informatique mise en place pour implémenter le mécanisme de l'EDIP réalisée.

En effet, lorsque le concepteur entame un processus de conception, il crée une instance de la classe "ProcessManager" qui instancie à son tour la classe "Device" support de la base de données centrale contenant les paramètres et leurs valeurs. Pour intégrer un composant à utiliser dans son processus, il lui suffit de créer une instance de la classe du connecteur correspondant.

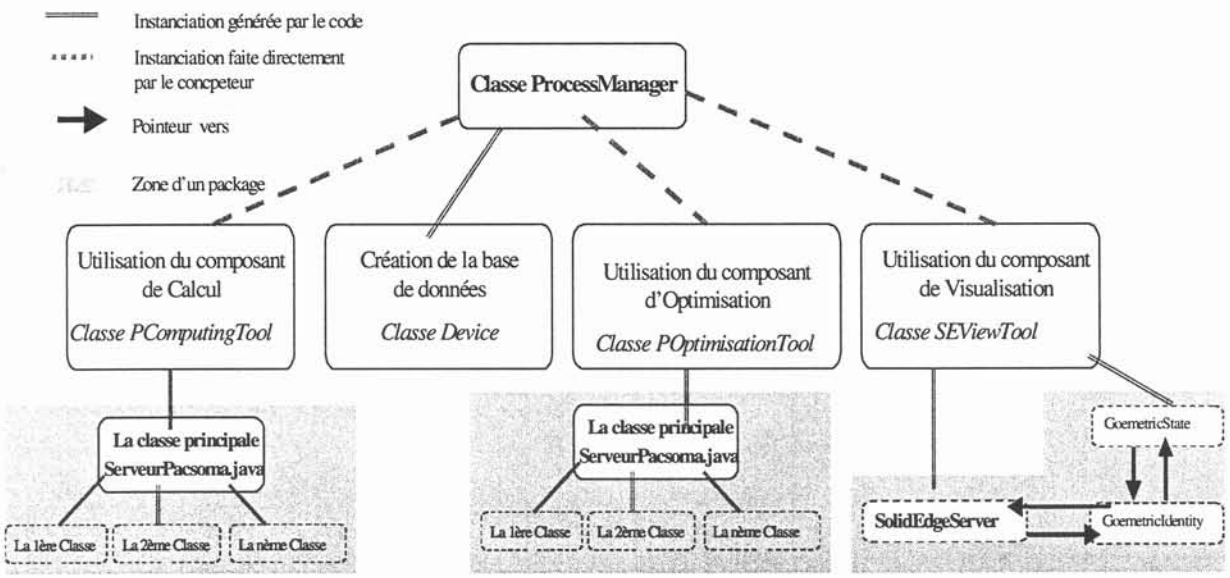


Figure V- 27 La structure informatique du mécanisme de l'EDIP réalisée

Nous avons développé une interface graphique pour assister le concepteur dans l'utilisation de notre prototype d'EDIP. L'ensemble de l'interface a été réalisé donc en Java en liant graphiquement les Objets Java dédiés à l'IHM avec nos Objets métiers. Nous tenons à préciser que l'IHM a été réalisée à l'aide du module Visual Builder de IBM VisualAge. La Figure suivante montre une image de la fenêtre principale de l'IHM.

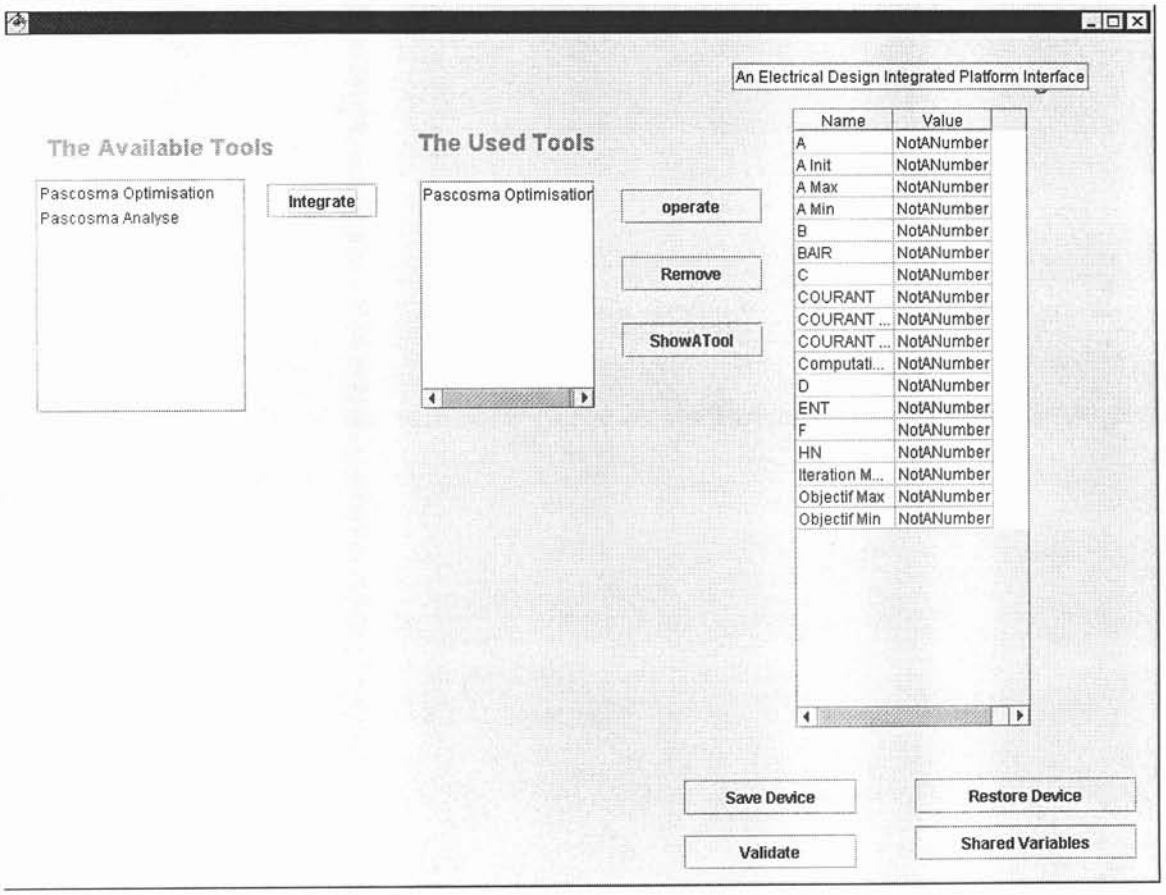


Figure V- 28 La fenêtre principale de l'interface réalisée : l'application contacteur chargée

V.4.4.2 *La structure informatique de la base de données centrale*

Dans le chapitre III, nous avons précisé que les paramètres du dispositif sont groupés au niveau d'une base de données centrale : un modèle produit minimal en vue du dimensionnement. En pratique, nous capturons l'information utile au niveau de l'Objet "Device" du package designProcess. Tous les services pour manipuler les paramètres du dispositif correspondent à des méthodes de cette classe comme le montre la Fig. V-28.

Toutefois, nous rappelons que les changements de structure ne sont pas autorisés à travers le prototype puisque nous manipulons des Objets de Conception à topologie fixe (cf. chapitre III).

Les valeurs affichées sont automatiquement tenues à jour. A chaque fois que le concepteur exécute un outil, les résultats de cette action sont propagés à l'instance du "Device" en cours et les paramètres correspondants seront rafraîchis au niveau de la fenêtre principale.

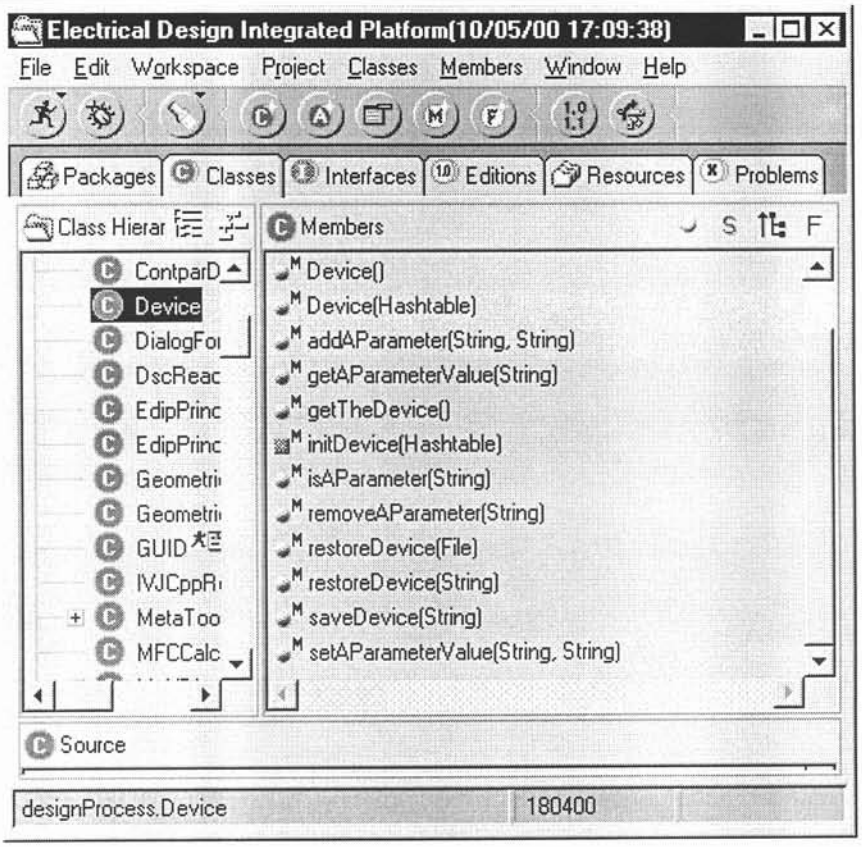


Figure V- 29 Les méthodes de la classe "Device"

V.4.4.3 *Le suivi du processus de Conception*

Comme nous l'avons expliqué dans le deuxième chapitre (au niveau du modèle de processus de conception), il est important de garder une trace du processus mené par le concepteur. Nous souhaitons sauvegarder sur un support les étapes du processus : nous pensons à un exemple «d'un tableau de bord » qui permet de suivre les étapes franchies pour la conception du produit à savoir les outils utilisés pour la conception, l'ordre de leurs exécutions, l'évolution du dispositif en cours de la conception…

Nous avons mis en œuvre la classe "ProcessPath" pour créer un historique du processus de conception dans un fichier "*.html".

Pour cela à chaque fois que le concepteur exécute une étape décisive dans le processus, nous enregistrons son action dans une pile des actions. Pour créer l'historique du processus de conception fait par le concepteur, il lui suffit de cliquer sur le bouton "Historic" au niveau de l'interface principale et d'indiquer au niveau de la fenêtre de dialogue qui s'ouvre le répertoire où il souhaite enregistrer son fichier.

Conclusion : Bilan du prototype

Grâce à la notion d'outil ("Tool") et de ProcessManager, nous avons aboutit à un exemple d'environnement de conception intégré générique et ouvert pour le dimensionnement des dispositifs électriques paramétrés.

Le mécanisme de liens des outils est générique ; le couplage des composants est indépendant de leurs spécificités.

Le processus implémenté nous a permis d'assurer le transfert automatique des données entre Pascosma et Solid Edge. Nous avons pu valider le prototype réalisé pour le dimensionnement de l'application "électroaimant" (cf. scénario en annexe C).

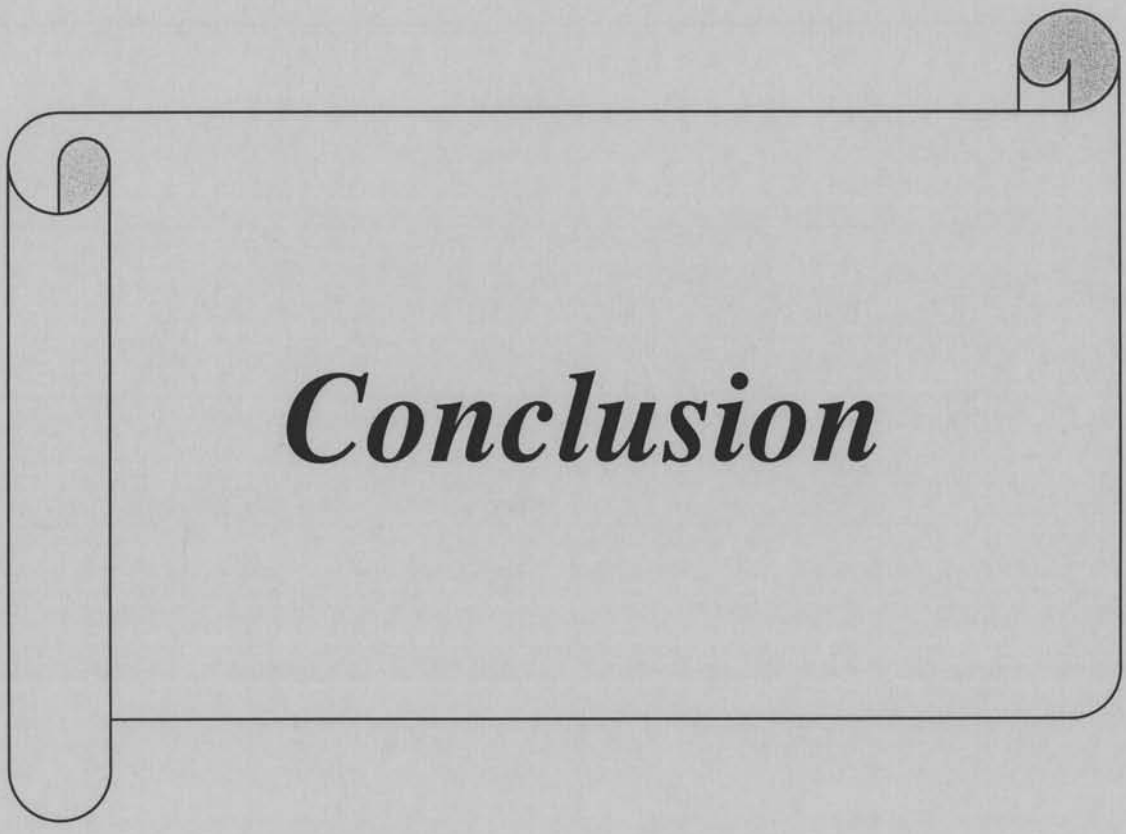
L'environnement réalisé supporte la trilogie : Paramètres, Méthodes et Stratégie, introduite dans le deuxième chapitre. Les paramètres constituent le support de l'information des produits à concevoir. Les méthodes se retrouvent au niveau des services offerts par les composants.

Quant à la stratégie, avec cet exemple nous avons mis en place un environnement à processus semi-structuré puisque l'ensemble des tâches à exécuter est connu mais leur enchaînement n'est pas forcément établi à l'avance.

C'est pourquoi nous pouvons parler d'un générateur de processus de Conception. Nous pouvons créer autant de scénarios d'utilisations des outils que nous le souhaitons.

D'autre part, l'extensibilité du prototype ne fait pas de doute. Néanmoins un investissement est à faire pour intégrer un nouvel outil au prototype de l'EDIP : il faut réaliser le composant logiciel à partir de l'outil en question ainsi que son connecteur.

En ce qui concerne la tracabilité du processus, le prototype offre un historique des actions ordonnées par le concepteur.



Conclusion

Conclusion et Perspectives

Le travail présenté dans ce mémoire porte sur la mise en place d'une méthodologie de développement d'environnements intégrés d'aide à la Conception en génie électrique. Nous avions comme objectif de fournir aux concepteurs une architecture pour organiser et optimiser l'exercice de la Conception Intégrée.

Pour atteindre cet objectif, il nous était nécessaire de connaître de près les spécificités de l'activité de la Conception et le formalisme de modélisation implémenté dans les outils de Conception.

La plus grande contrainte et ambition de l'approche proposée est de définir une démarche générique de couplage de logiciels hétérogènes.

Nous avons ainsi développé un modèle de plate forme intégrée à bases de composants logiciels. Nous avons mis en œuvre le modèle Serveur d'Objets pour la réalisation de ce modèle.

Grâce à cette démarche, nous pouvons désormais encapsuler un outil d'aide à la conception pour en faire un "composant" autonome prêt à être branché dans un environnement de Conception et être associé avec d'autres composants d'une manière générique.

Une architecture centralisée a été dégagée pour l'agrégation des composants, à base d'objets codés en JAVA. Elle permet de gérer à la fois la propagation des paramètres des dispositifs électriques à dimensionner et le partage des services offerts par les composants.

Outre sa modularité, l'organisation des outils d'aide à la conception en composants permet de fournir au concepteur des éléments réutilisables pour mettre au point des processus de Conception Assistée par Ordinateur.

A partir des composants de Conception réalisés le concepteur peut construire autant d'environnements de préconception qu'il le souhaite moyennant un faible investissement: développement du connecteur du composant vers l'environnement.

Le prototype réalisé permet de coupler deux outils métiers PASCOSMA (utilisé essentiellement par les électriciens) et SOLID EDGE (utilisé par les mécaniciens) sans aucune imbrication.

Nous avons opté pour un processus quasi-automatique. Ce choix émane d'une volonté de laisser la décision à la charge de l'acteur humain. Toutefois, il est possible de réorganiser facilement l'environnement en demandant au concepteur de spécifier l'enchaînement souhaité de ses outils dés le départ. Une fois le processus figé l'activité de Conception peut se dérouler automatiquement sans aucune intervention.

A court terme, nous envisageons de rendre réparti le prototype réalisé et de valider notre approche pour le travail collaboratif distribué. Dans ce sens, notre travail servira de référence à la réalisation d'un projet de "Co-conception via Internet" qui a débuté en janvier 2000 en collaboration entre l'équipe Conception Intégrée du laboratoire 3S (Sols, Solides et Structures) et l'équipe CDI du LEG, et dans le cadre de l'IPI (Institut de la Production et des organisations Industrielles).

L'information propagée dans l'état actuel du prototype correspond à une sémantique minimale (correspondance des noms de variables des différents outils). L'enrichissement de cette sémantique est à développer dans le cadre des travaux futurs de l'équipe CDI.

Nous avons mis en place des « composants » logiciels de Conception, sans nous conformer complètement jusqu'à présent à un standard de composants. Il serait intéressant d'envisager de s'aligner pour la suite sur un standard existant, nous citons en particulier celui des JavaBeans.

Annexes

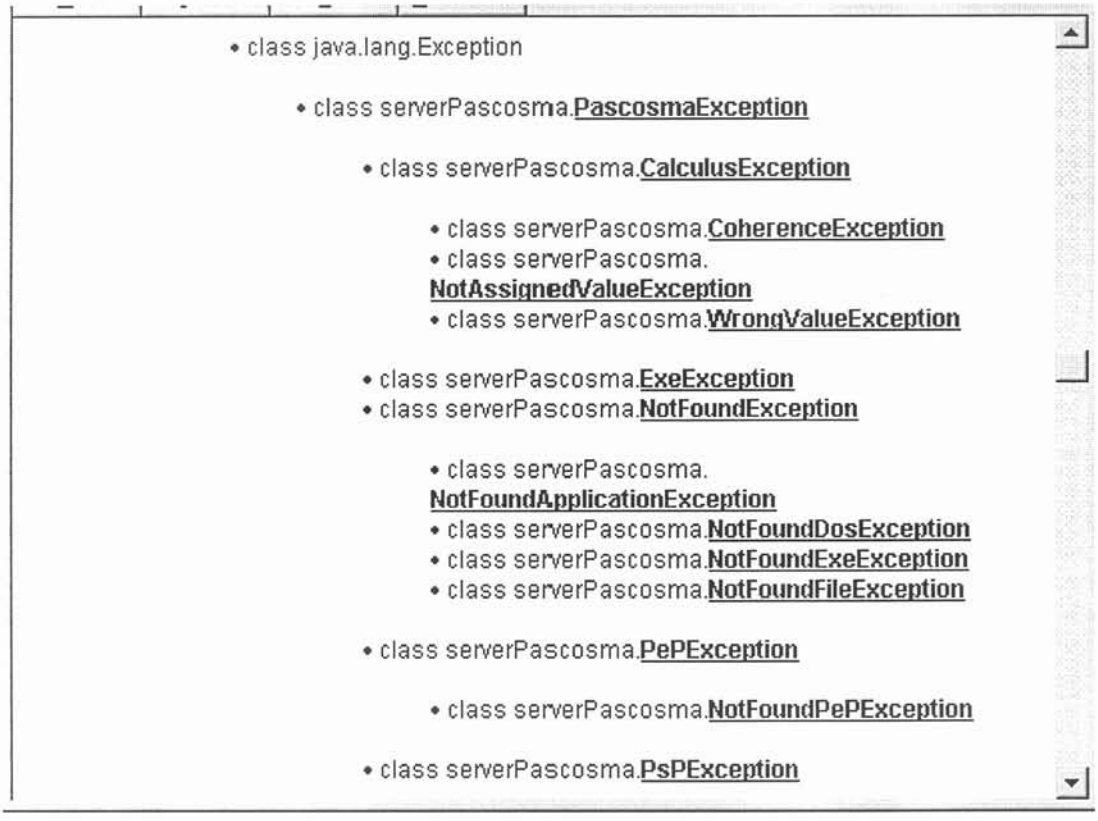
Annexe A

Documentation Java

Dans cette annexe nous fournissons la documentation sur les packages mis en place et les principales classes développés.

1. Le package severPascosma:

- *La hiérarchie des Exceptions du package serverPascosma est :*



- *Un apercu de la classe principale ServeurPascosma:*

serverPascosma
Class ServeurPascosma

java.lang.Object
|
+--serverPascosma.ServeurPascosma

public class ServeurPascosma
extends java.lang.Object
implements java.io.Serializable

La classe ServeurPascosma est écrite afin de rendre transparentes toutes les fonctionnalités de l'outil Pascosma à tout Client de Pascosma (Interface, composant Logiciel, Classe java...). Ainsi chaque fonctionnalité de l'outil Pascosma est capturée dans une méthode de la classe ServeurPascosma et est ainsi assimilée à un service générique proposé à l'opérateur.

Cette classe implémente l'interface java.io.Serializable afin de pouvoir sauvegarder et récupérer des instances de cet objet d'une exécution à l'autre.

Les variables de cette classe doivent être visibles de l'extérieur du package serverPascosma donc elles doivent être accessibles pour toutes les classes à l'extérieur du package.idem pour les méthodes du serveurPascosma.

Pour certaines variables du serveur, une affectation d'une valeur par défaut n'a aucune signification tel que la valeur initiale d'un Pe il faut que l'opérateur affecte une valeur significative pour l'application étudiée. Des que l'affectation de la valeur est obligatoire, je renvoie une erreur NotAssignedValueException lorsque aucune affectation de la valeur n'a eu lieu.

Version:
1.0
Author:
Basma BEL HABIB
See Also:
Serialized Form

Field Summary

java.util.Vector
peList
peList: la liste des noms des Pe
java.util.Vector
psList
psList: la liste des noms des Ps

Constructor Summary

ServeurPascosma()
Constructeur sans argument de la classe ServeurPascosma
ServeurPascosma(java.io.File theposition)
Constructeur à argument de la classe ServeurPascosma

Method Summary

void **CalculusAnalyse()**
Méthode CalculusAnalyse permettant de mener un calcul d'analyse.

void **CalculusAnalyseForSets()**
Méthode CalculusAnalyseForSets permettant de mener un calcul d'analyse lorsqu'il y a N jeux de paramètres à traiter.

void **CalculusOptimisation()**
Méthode CalculusOptimisation permettant de mener un calcul optimisation.

Void **CalculusSensibility()**
Méthode CalculusSensibility permettant de mener un calcul de Sensibilité.

java.lang.String **getAllAboutPe()**
Méthode getAllAboutPe permet d'accéder à toute l'information sur les Pe en fournissant la liste des noms des PE, les valeurs de départ des Pe et en spécifiant leurs types Pe Optimisable ou fixe.

java.lang.String **getAllAboutPs()**
Méthode getAllAboutPs permet d'accéder à toute l'information sur les Ps en fournissant la liste des noms des PS, les valeurs de départ des PS et en spécifiant leurs types Ps Optimisable ou fixe.

double **getAnObjectifDerivatives(java.lang.String thePeName)**
Méthode get pour la lecture d'une dérivée partielle de la fonction Objectif par rapport à un pe spécifique.

int **getNumberOfSets()**
Méthode get pour la lecture du Nombre de jeux de données à traiter Par défaut le nombre de jeu de paramètre vaut 1.

int **getNumberPe()**
Méthode get pour la lecture du Nombre de PE

int **getNumberPs()**
Méthode get pour la lecture du Nombre de PS

ObjectifPascosma **getObjectif()**
Cette méthode getObjectif permet d'accéder à la Fonction Objectif de l'application Pascosma en cours.

double **getObjectifMax()**
Méthode get pour la lecture de la borne sup de la fonction objectif.

La liste des méthodes est trop longue; nous contentons de cet aperçu des méthodes offertes.

,ect...

2. Le package solidEdge

- *La hiérarchie du package serverSolidEdge*

- *Un apercu de la classe principale ServerSolidEdge:*

---

serverSolidEdge
Class ServerSolidEdge

java.lang.Object
|
+--serverSolidEdge.ServerSolidEdge

---

public class ServerSolidEdge
extends java.lang.Object

The serverSolidEdge Class allow the designer to create a GeometricIdentity and to manipulate it.

Version:
1.0
Author:
Basma BEL HABIB

Constructor Summary
ServerSolidEdge(java.io.File f)
constructor with argument
ServerSolidEdge(java.lang.String fname)

Method Summary

java.io.File	**getDevice()** Method get to read the studied application repository
GeometricIdentity	**getGeometricIdentity()** Method getGeometricIdentity() to create the indentity of the studied application.
void	**quit()** close the Solid Edge application
void	**showAGeometricState(GeometricState thestate)** Method showAGeometricState() to visualize the geometric state .

Annexe B

L'électroaimant choisi est un dispositif électrique élémentaire comme le montre le schéma de la Fig. V-1. Il est formé par une bobine électrique entourant un circuit magnétique.

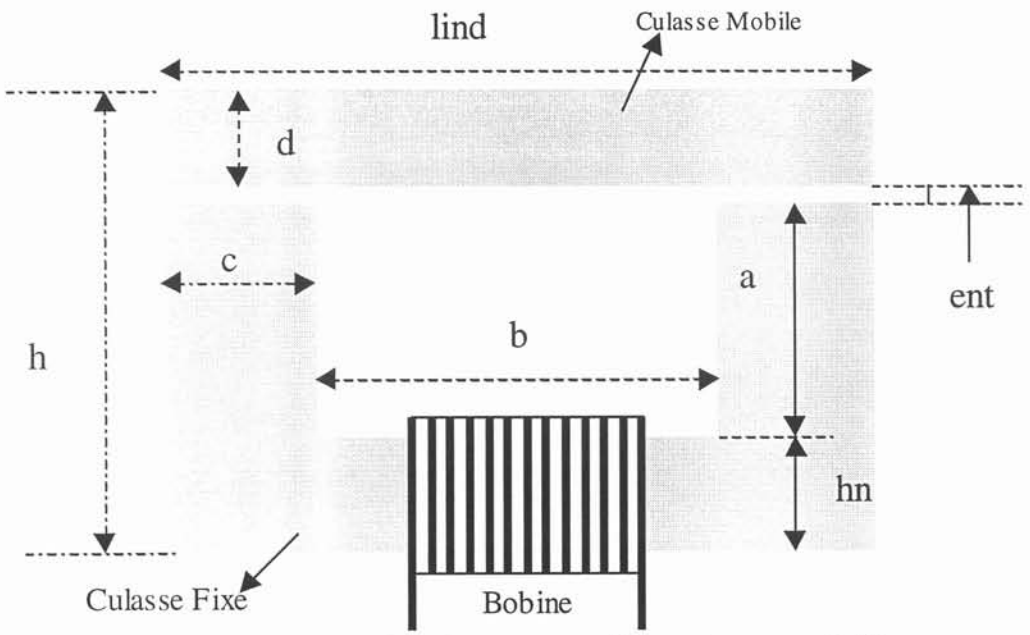


Fig. Annexe-B-1 Schéma du contacteur élémentaire avec sa bobine

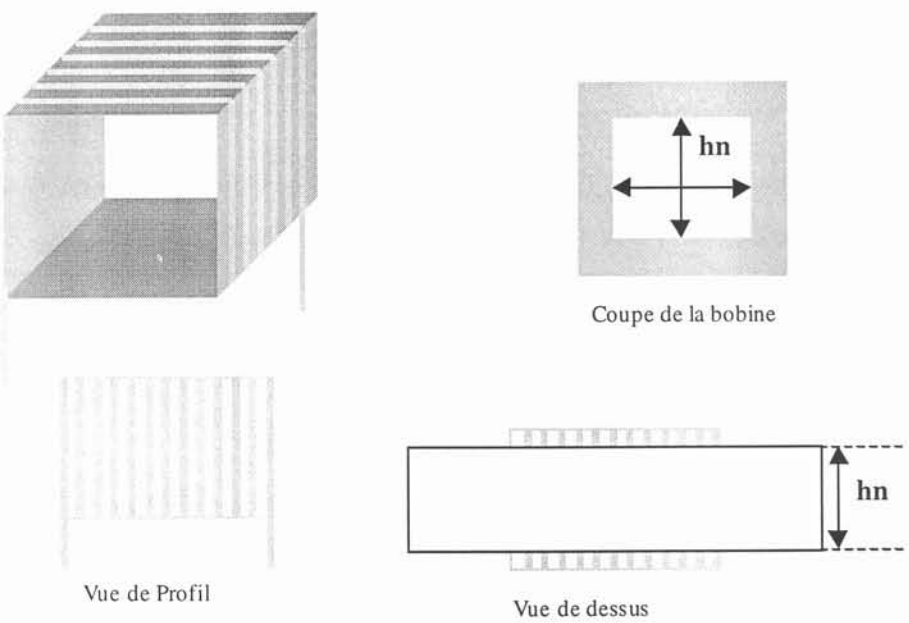


Fig. Annexe-B-2 Les différentes vues de la bobine

Les variables utilisées dans le modèle sont définies comme suit:

- a: Hauteur de la dent de la culasse fixe
- atair: Ampères tours consommés par la circulation du flux dans l'air
- atcf: Ampères tours consommés dans les dents de la culasse fixe

- atcm: Ampères tours consommés dans la culasse mobile
- atcn: Ampères tours consommés dans le noyau de la culasse fixe
- attot: Ampères tours totaux consommés sur la ligne de circulation de flux moyenne
- b: Largeur de l'entraxe de la culasse fixe
- bair: Valeur de l'induction dans l'air
- bcf: Valeur de l'induction dans la dent de la culasse fixe
- bcm: Valeur de l'induction dans la culasse mobile
- bcn: Valeur de l'induction dans le noyau de la culasse fixe
- c: Largueur de la dent de la culasse fixe
- d: Hauteur de la culasse mobile
- ent: Largeur de l'entrefer
- f: Force d'appel entre la culasse fixe et la culasse mobile
- fluair: Valeur du flux total dans l'entrefer
- h: Hauteur totale du circuit magnétique
- hn: Hauteur du noyau de la culasse fixe et profondeur du circuit magnétique
- lind: Largeur totale du circuit magnétique
- m: Masse totale de l'électroaimant

Cette application a été utilisée dans la thèse [Wur96] pour valider la méthodologie de Pascosma. A priori le dimensionnement d'un tel dispositif électrique ne risque pas de poser des problèmes d'ordre physique. Pour ce faire, nous négligeons les fuites et nous calculons les inductions maximales dans les différentes parties du circuit magnétique par les lois de conservation du flux.

Nous disposons donc d'un modèle analytique paramétré du contacteur étudié dont les équations sont fournies sur la Fig. ci-dessous.

equation.mac - Bloc-notes

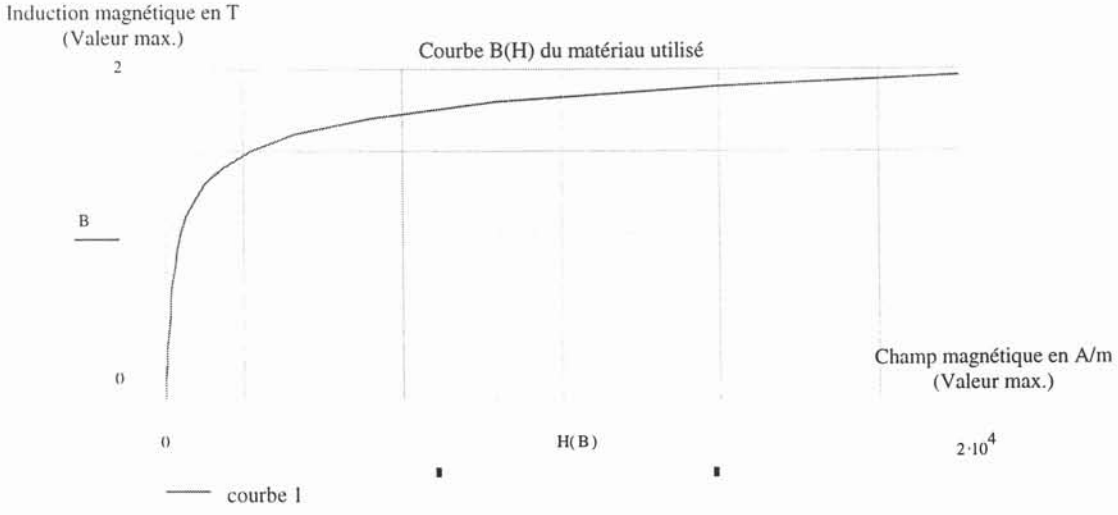
Fichier Edition Rechercher ?

```
   Fichier equation.mac

   Fichier devant contenir les equations du modele
*/

/*
   Date de creation: Wed Jan  27 99 15:21:31

   Repertoire racine de creation de l'application:
   Z:\pascosma\contacteur\
*/
eq(lind=2*c+b);                          eq(atair=2*bair*ent/(4*%pi*10^-7));

eq(fluair=bair*c*hn);                    eq(atcf=(ka*exp(kb*bcf^2)+kc)*bcf*(2*a+hn));

eq(attot=atair+atcm+atcf+atcn);          eq(courant=attot/10);

eq(bcf=bair);                            eq(h=d+ent+a+hn);

eq(bcm=fluair/(hn*d));                   eq(atcm=(ka*exp(kb*bcm^2)+kc)*bcm*(b+c));

eq(bcn=fluair/hn^2);                     eq(atcn=(ka*exp(kb*bcn^2)+kc)*bcn*(lind-c));

eq(m=(d*lind+2*a*c+lind*hn)*hn*7800+0.003);

eq(f=bair^2*c*hn*2/(4*%pi*10^-7*2));


/*
        Constantes du probleme
*/

eq(ka=76.1); eq(kb=1.26); eq(kc=129.5);
```

Fig. Annexe-B-3 Les équations du modèle paramétré de l'électroaimant

Par ailleurs, le comportement magnétique du circuit est fourni par la fonction H(B) suivante:

$$H(B) = [ka * \exp(kb * B\hat{}2) + kc] * B \quad \text{où} \quad \begin{cases} ka = 76.1 \\ kb = 1.26 \\ kc = 129.5 \end{cases}$$



Annexe C

Scénario de processus de conception sous EDIP

V.1 Objectif du Scénario:

Le concepteur souhaite:

1- Brancher le composant Calcul de Pascosma.

2- Initialiser les valeurs de son dispositif.

3- Réaliser un calcul simple avec le composant Calcul de Pascosma.

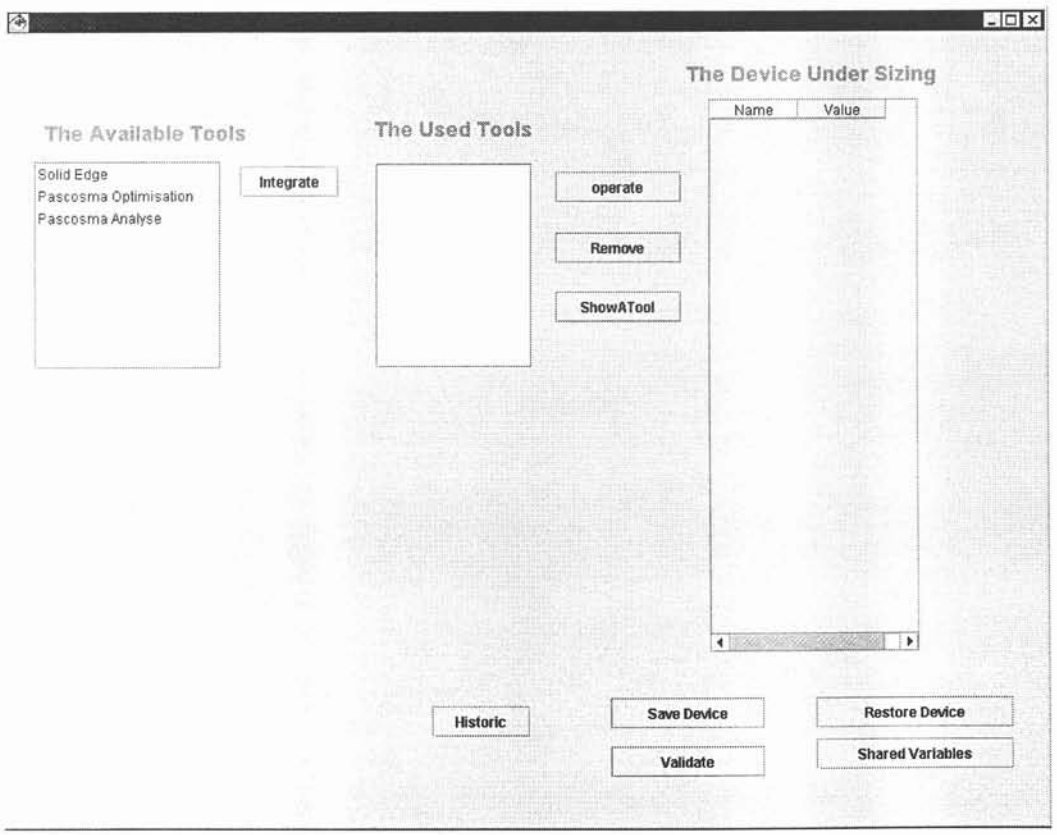
4- Brancher Solid Edge.

5- Visualiser la géométrie du dispositif sous Solid Edge.

V.2 Déroulement normal du scénario:

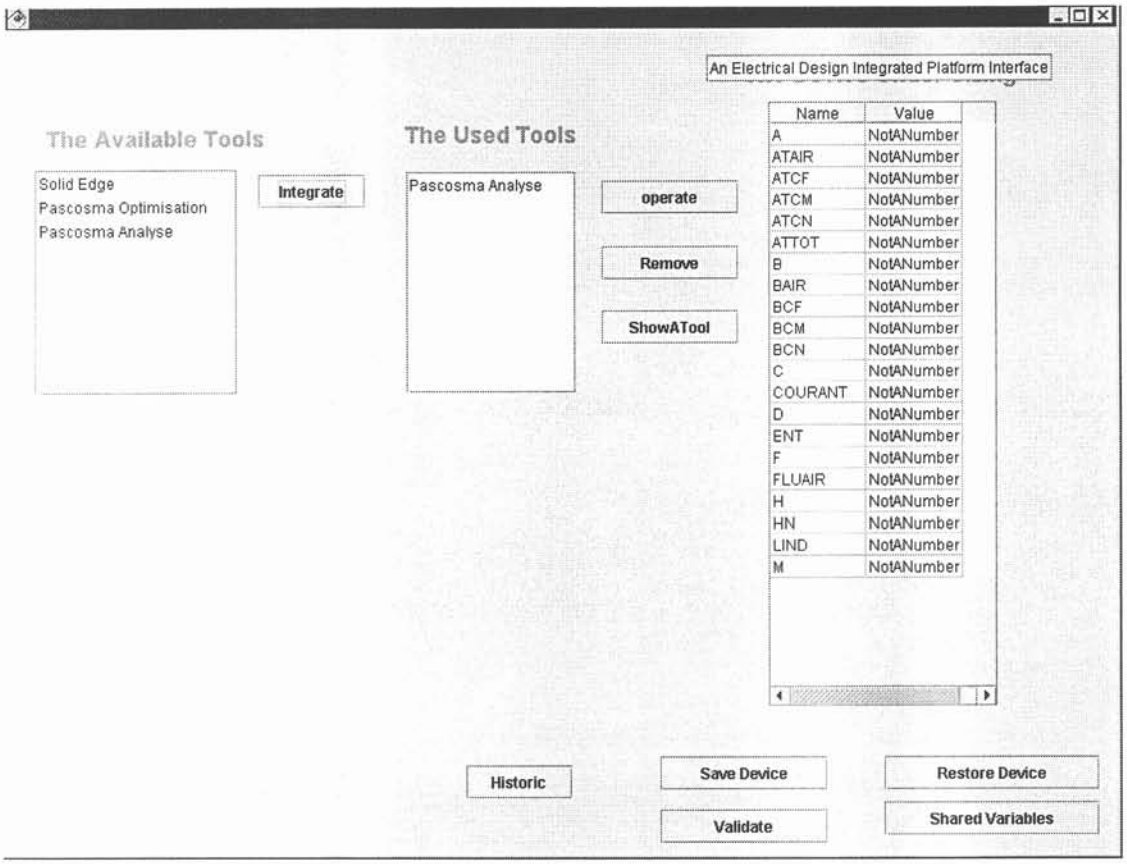
Nous allons décrire ces différentes étapes à travers l'évolution de l'interface:

(1) La fenêtre principale s'affiche; La barre des outils disponibles montre les composants de Conception disponibles. Aucun dispositif n'a été encore chargé.



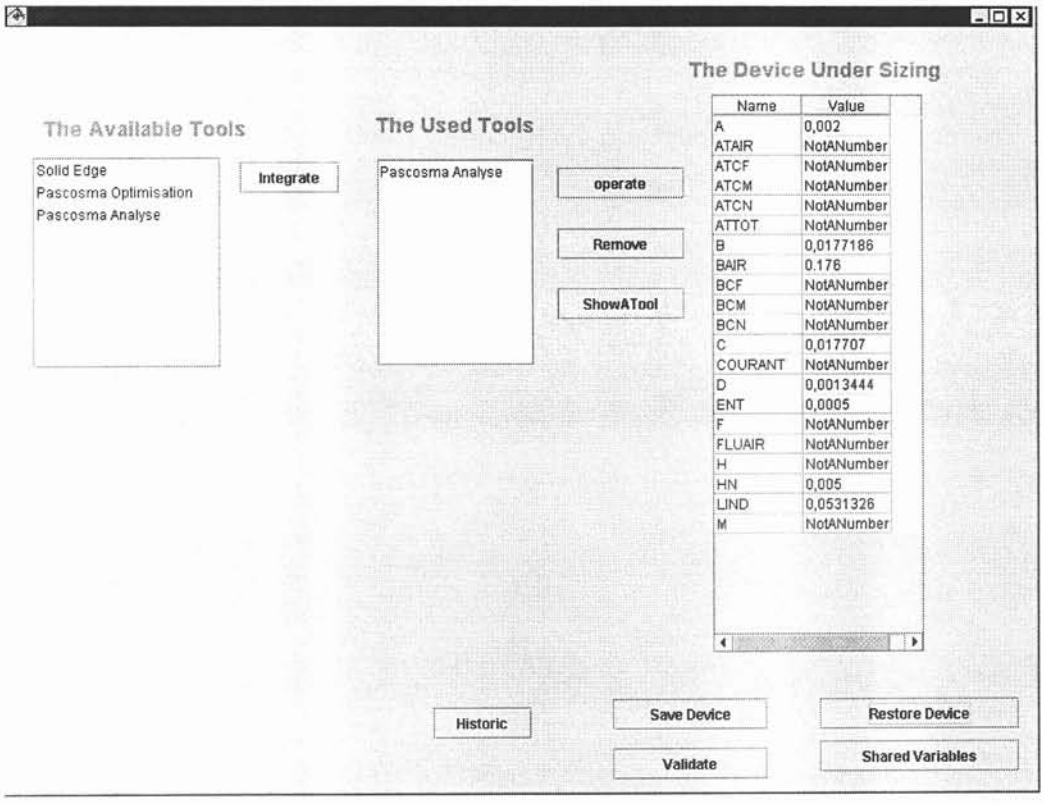
(2) Si le concepteur souhaite intégrer le composant de Calcul de Pascosma dans son processus, il lui suffit de le sélectionner dans la barre d'outils disponibles et de confirmer son choix en cliquant sur le bouton "Integrate".

Le nom du composant choisi va alors s'ajouter à la liste des outils utilisés comme le montre la figure suivante. Une fenêtre de Dialogue s'ouvre pour lui demander d'indiquer le fichier où est placé son objet de Conception. Il procèdera de la même manière s'il souhaite ajouter un autre composant.

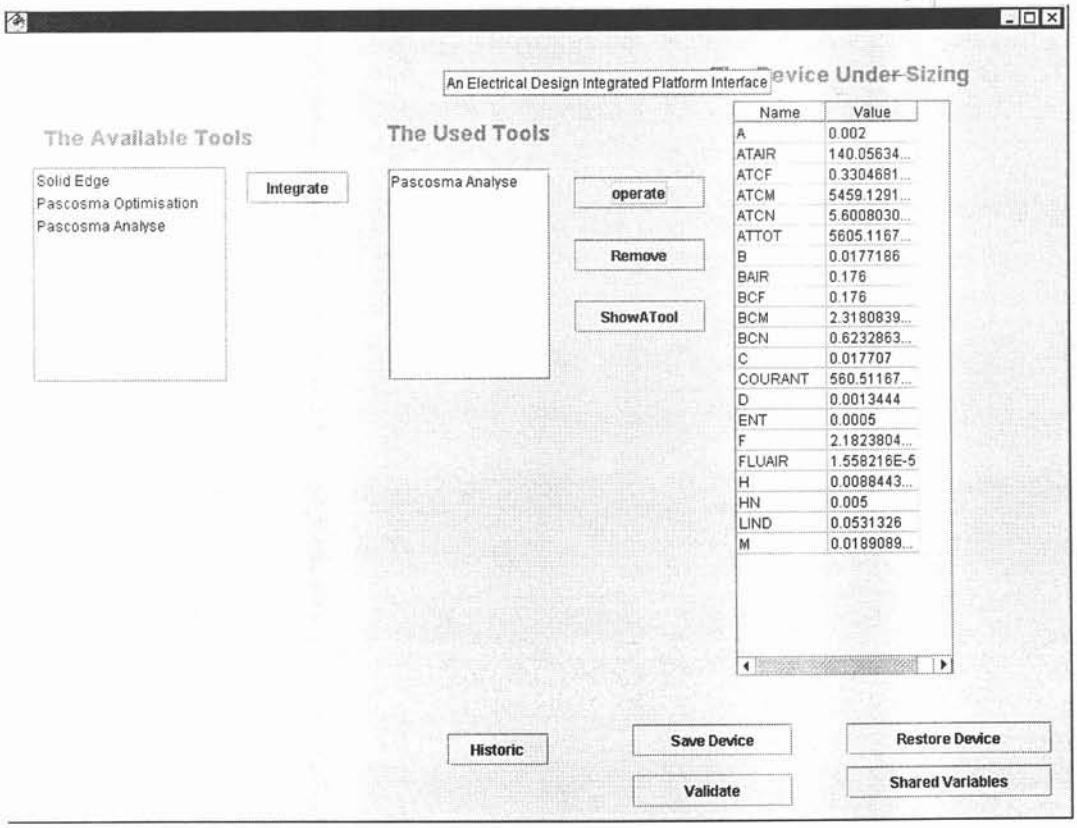


(3) Si le concepteur souhaite un calcul simple avec le composant de Calcul, il peut introduire directement les valeurs des Inputs au niveau de la fenêtre, comme il peut récupérer des valeurs qu'il a déjà introduites une fois et sauvegardées dans un fichier. Pour cela, il lui suffit de valider le bouton "RestoreDevice".

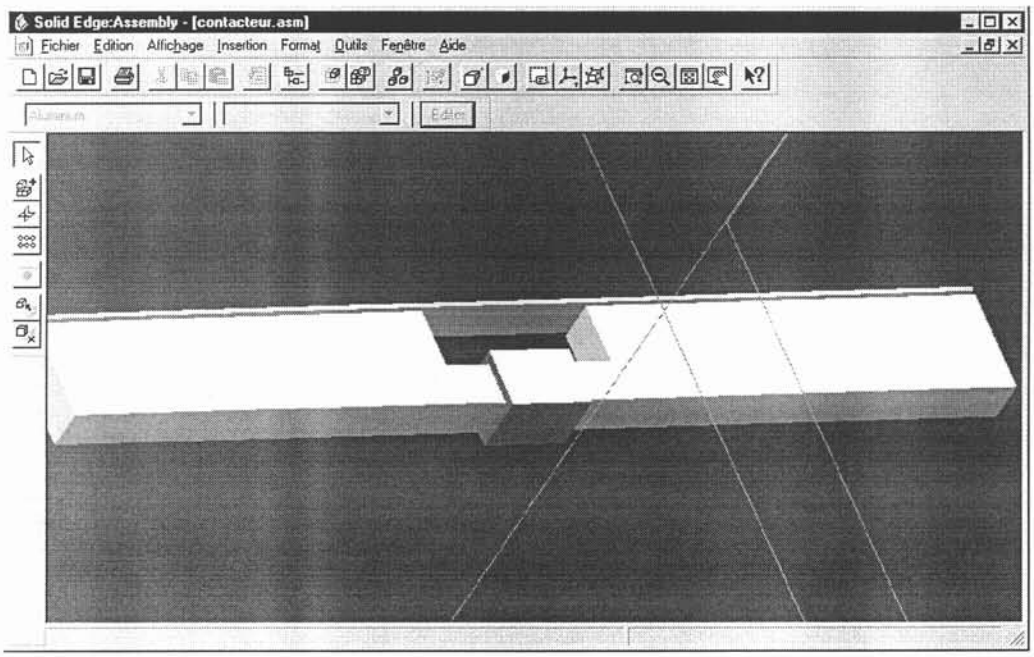
Une fenêtre de Dialogue s'ouvre pour lui demander d'indiquer le fichier où est sauvegardé le "Device" qu'il souhaite récupérer. Les valeurs des variables vont être actualisées au niveau de l'interface comme le montre la Figure suivante:



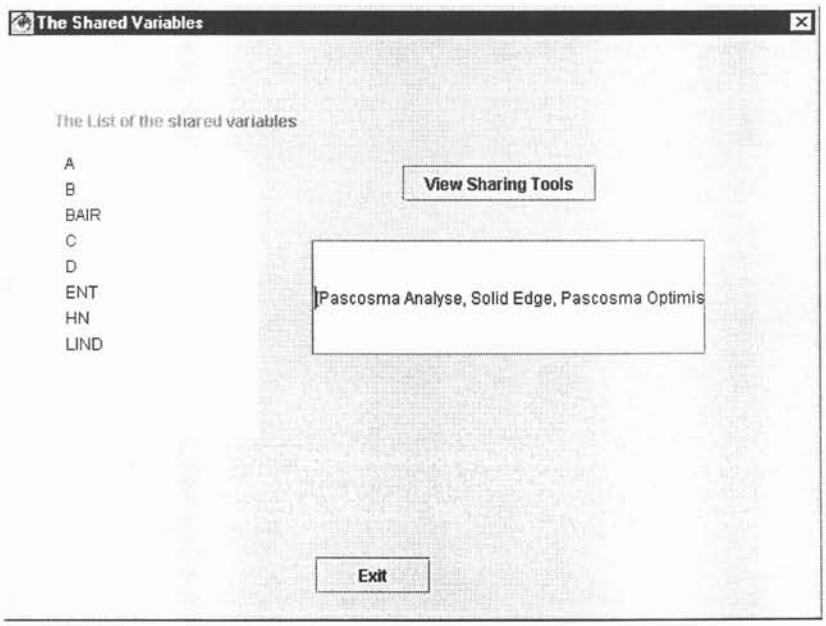
(4) Si le concepteur souhaite calculer les autres paramètres (les Outputs), il sélectionne le composant Calcul parmi la liste des outils intégrés et valide son choix en cliquant sur le bouton "operate". Après exécution du calcul, les valeurs sont actualisées comme le montre la figure suivante:



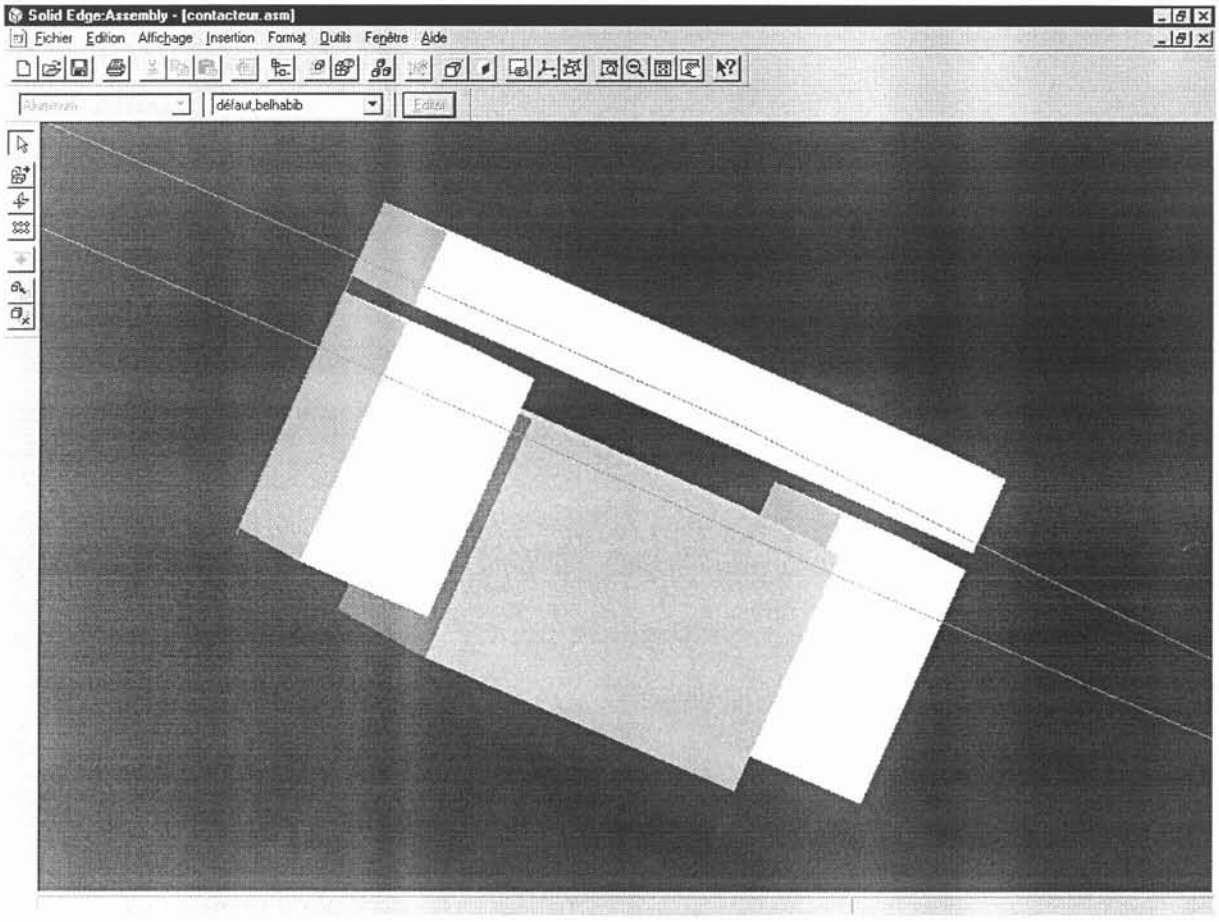
(5) Le concepteur intègre le composant de visualisation en procédant de la même manière que nous l'avons expliqué en (2). La géométrie du contacteur qui s'affiche au moment de la connexion de Solid Edge à EDIP est celle qui est enregistrée dans le fichier d'assemblage. C'est une géométrie aberrante sans aucun dimensionnement.



Les variables géométriques du modèle analytique de Pascosma qui sont paramétrés à travers Solid Edge vont être notifiées par le programme en tant que variables partagées comme le montre l'image suivante:



(6) Pour actualiser la géométrie, le concepteur n'a qu'à sélectionner Solid Edge dans la liste des outils utilisés et cliquer sur le bouton "operate". La nouvelle géométrie qui correspond aux valeurs du dispositif en cours est actualisée:



Annexe D

Il est possible de manipuler les applications Solid Edge à partir d'un programme écrit en Visual Basic.

A partir de ce programme, le développeur peut utiliser les méthodes getObject et createObject sur un objet OLE. En appelant la méthode createObject, il lance une nouvelle instance de Solid Edge. Un exemple de code du programme à écrire :

```
'Creation d'une nouvelle instance de Solid Edge.
 Public objApp As Object
 Set objApp = CreateObject("SolidEdge.Application")
```

Il suffirait par la suite de reconduire le même raisonnement pour le développement du code en se référant à la hiérarchie des objets dans l'environnent souhaité. Le cheminement est schématisé sur la Fig. suivante.

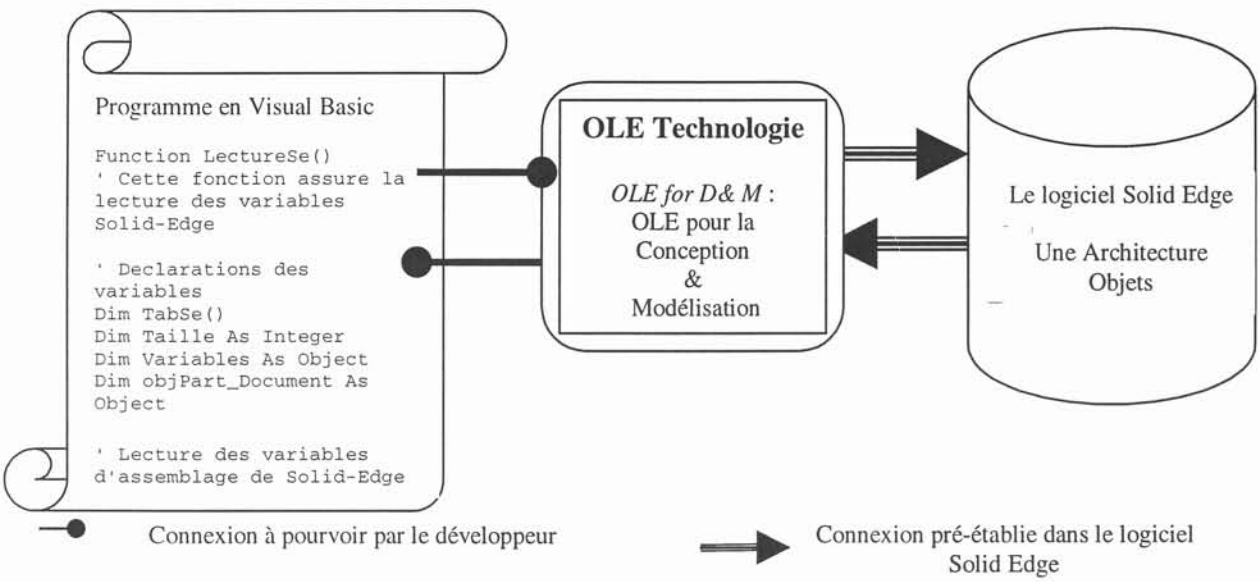


Fig. Annexe-D-1 Une solution pour programmer avec Solid Edge (version 4)

Une première expérience d'intégration a été menée dans l'équipe CDI (Conception et Diagnostic Intégrés) dans le but de coupler Solid Edge avec PASCOSMA. L'objectif était de visualiser la géométrie du dispositif dont le calcul a été réalisé avec PASCOSMA.

Un programme de couplage a été écrit en Visual Basic permettant d'affecter aux variables de l'application Solid Edge les valeurs résultats du calcul du logiciel de Dimensionnement PASCOSMA à une itération donnée.

Le mécanisme mis en place est schématisé sur la Fig. ci-dessous (figure issue de [MCS]). *** Il consistait à écrire dans un fichier les différentes valeurs issues de PASCOSMA. A partir de ces données, le programme de couplage vérifie la cohérence de l'application traitée dans les deux outils de CAO et charge le bon fichier de dessin en rafraîchissant les valeurs des paramètres.

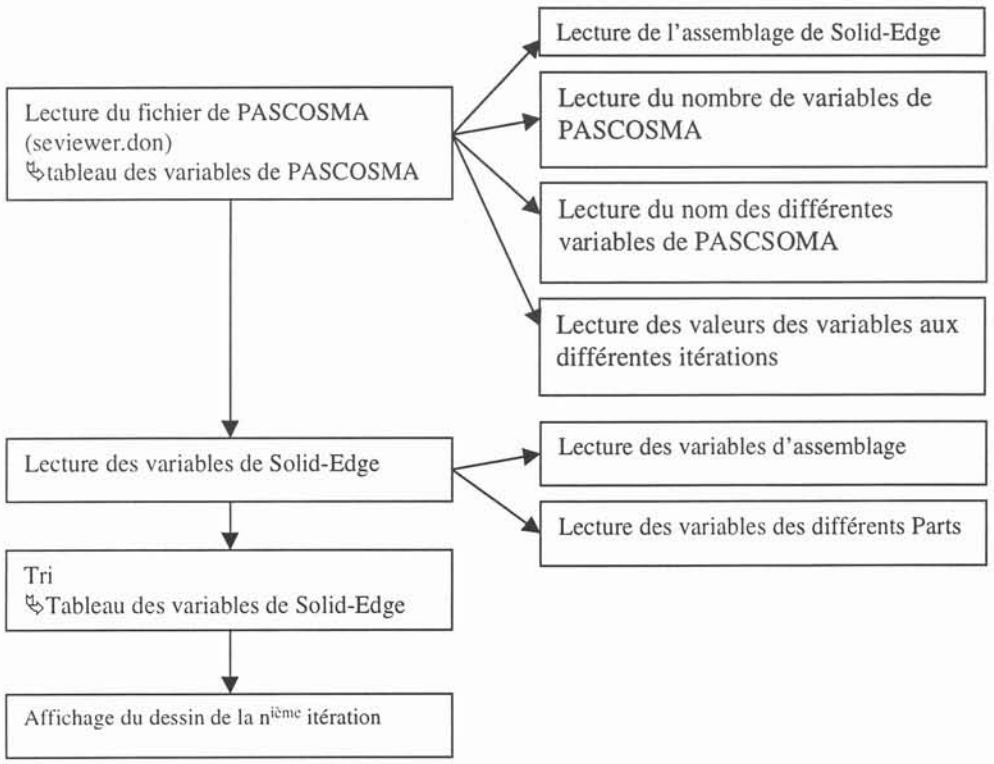
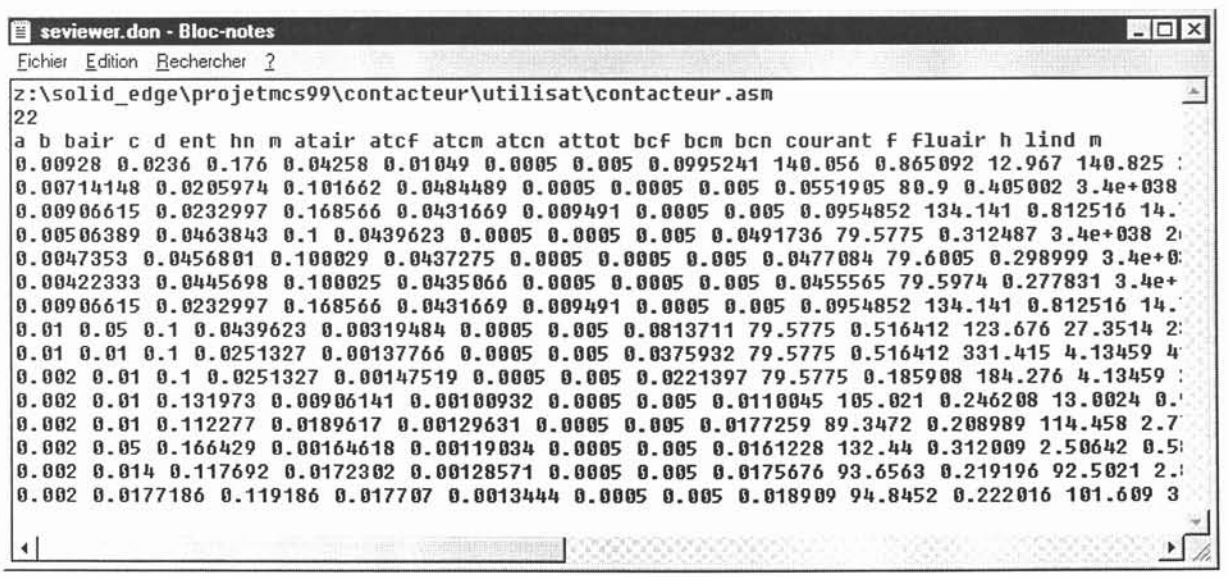


Fig. Annexe-D-2 Le mécanisme de couplage de Pascosma et Solid Edge

La Fig. ci-dessous donne un exemple du fichier de donnés de l'application contacteur.


seviewer.don - Bloc-notes

Fichier Edition Rechercher ?

```
z:\solid_edge\projetmcs99\contacteur\utilisat\contacteur.asm
22
a b bair c d ent hn m atair atcf atcm atcn attot bcf bcm bcn courant f fluair h lind m
0.00928 0.0236 0.176 0.04258 0.01049 0.0005 0.005 0.0995241 140.056 0.865092 12.967 140.825 :
0.00714148 0.0205974 0.101662 0.0484489 0.0005 0.0005 0.005 0.0551905 80.9 0.405002 3.4e+038
0.00906615 0.0232997 0.168566 0.0431669 0.009491 0.0005 0.005 0.0954852 134.141 0.812516 14.
0.00506389 0.0463843 0.1 0.0439623 0.0005 0.0005 0.005 0.0491736 79.5775 0.312487 3.4e+038 2
0.0047353 0.0456801 0.100029 0.0437275 0.0005 0.0005 0.005 0.0477084 79.6005 0.298999 3.4e+0
0.00422333 0.0445698 0.100025 0.0435066 0.0005 0.0005 0.005 0.0455565 79.5974 0.277831 3.4e+
0.00906615 0.0232997 0.168566 0.0431669 0.009491 0.0005 0.005 0.0954852 134.141 0.812516 14.
0.01 0.05 0.1 0.0439623 0.00319484 0.0005 0.005 0.0813711 79.5775 0.516412 123.676 27.3514 2
0.01 0.01 0.1 0.0251327 0.00137766 0.0005 0.005 0.0375932 79.5775 0.516412 331.415 4.13459 4
0.002 0.01 0.1 0.0251327 0.00147519 0.0005 0.005 0.0221397 79.5775 0.185908 184.276 4.13459 :
0.002 0.01 0.131973 0.00906141 0.00100932 0.0005 0.005 0.0110045 105.021 0.246208 13.0024 0.
0.002 0.01 0.112277 0.0189617 0.00129631 0.0005 0.005 0.0177259 89.3472 0.208989 114.458 2.7
0.002 0.05 0.166429 0.00164618 0.00119034 0.0005 0.005 0.0161228 132.44 0.312009 2.50642 0.5
0.002 0.014 0.117692 0.0172302 0.00128571 0.0005 0.005 0.0175676 93.6563 0.219196 92.5021 2.
0.002 0.0177186 0.119186 0.017707 0.0013444 0.0005 0.005 0.018909 94.8452 0.222016 101.609 3
```

Fig. Annexe-D-3 Le fichier de données crées par Pascosma pour le programme en Visual Basic pilotant Solid Edge

Cette fonctionnalité a été ajoutée à l'interface de Pascosma comme montré ci-dessous.

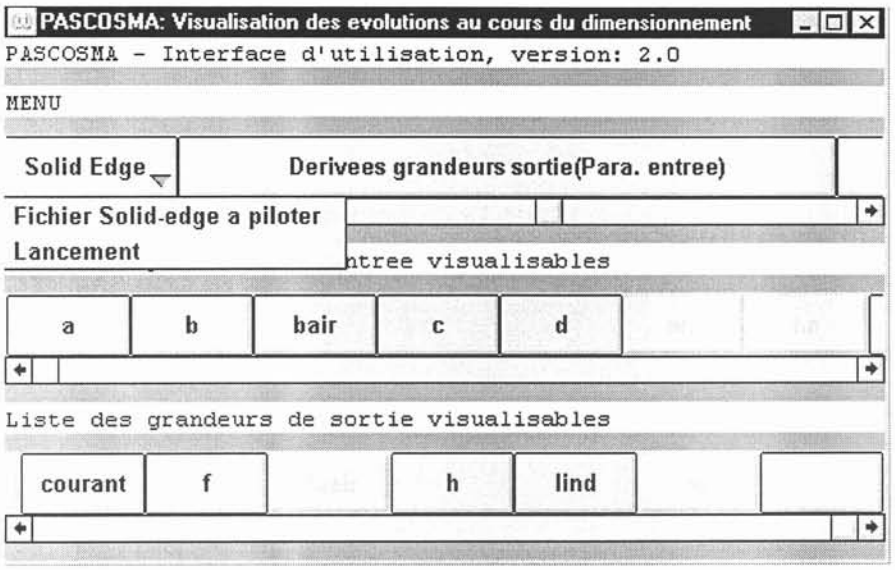


Fig. Annexe-D- 4 Actualisation de la géométrie sous Solid Edge à partir de l'interface graphique de Pascosma

Le programme Visual Basic écrit permet de créer une fenêtre visualisant les différentes solutions obtenues au cours de l'optimisation du dispositif sous Pascosma. Il suffit de faire varier le curseur pour afficher une itération donnée, comme le montre la figure suivante.

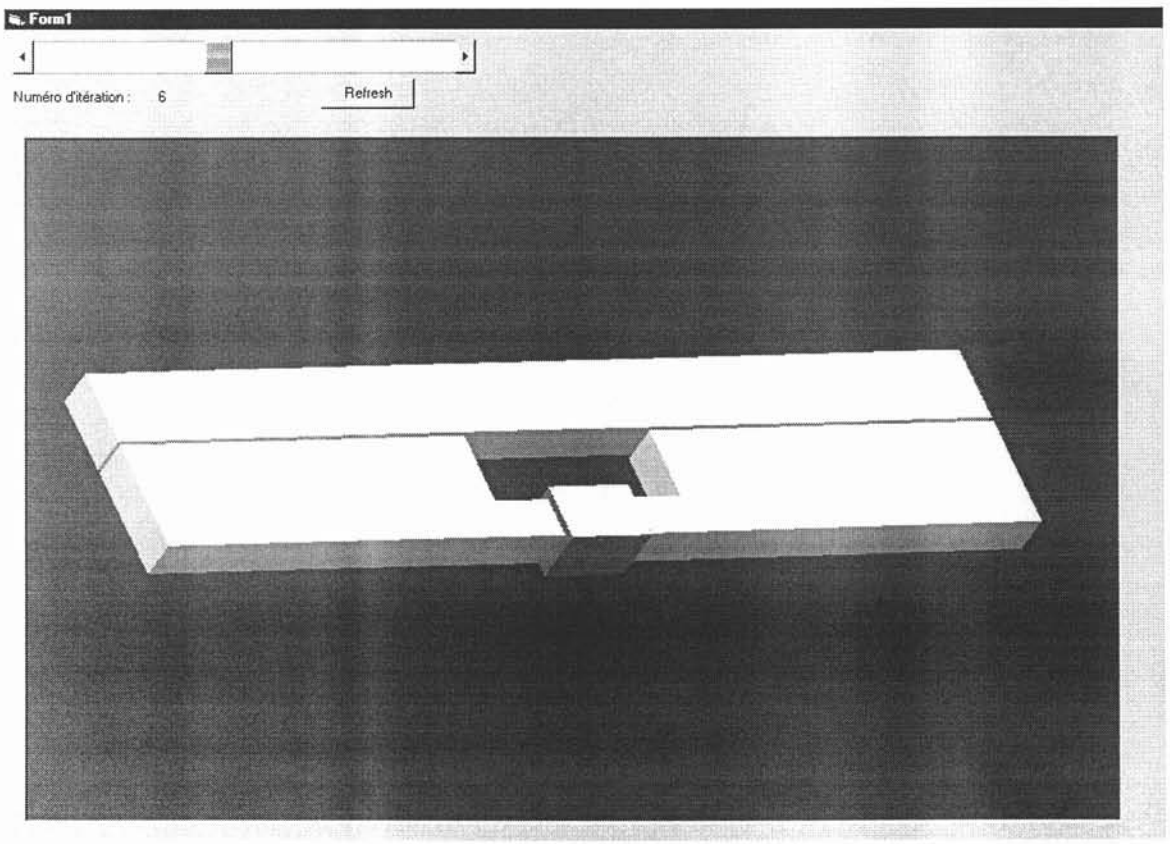
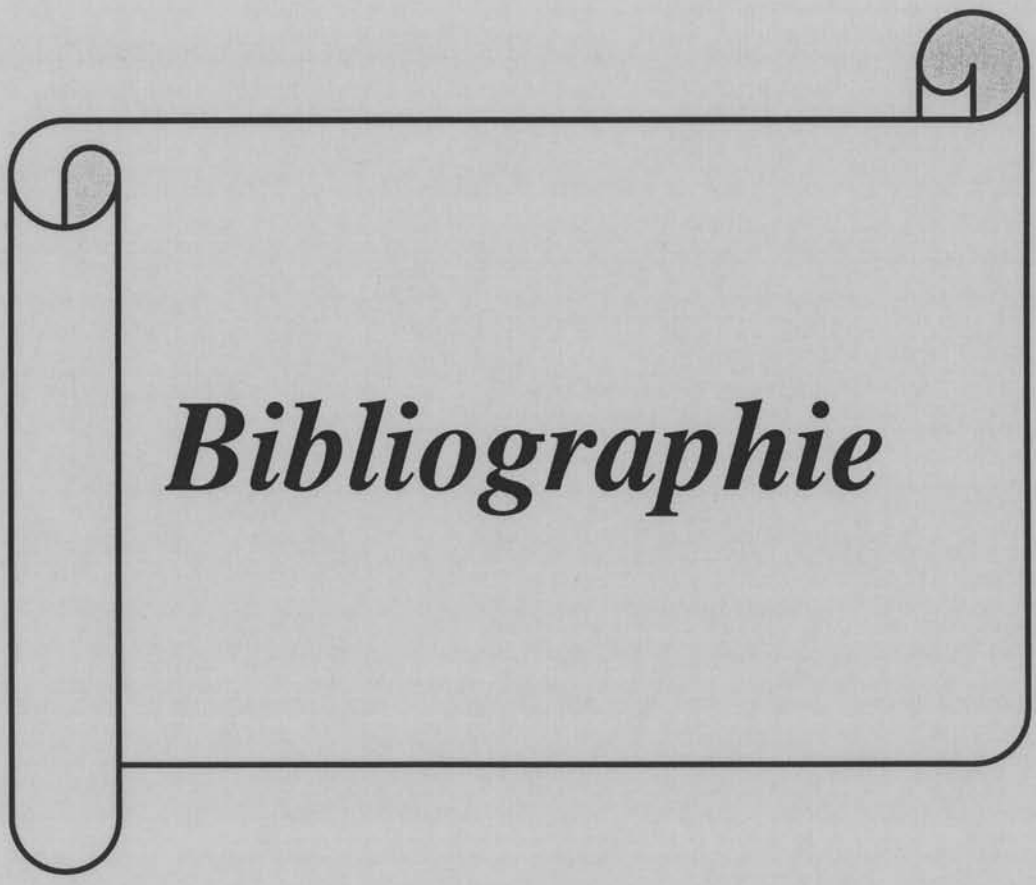


Fig. Annexe-D-5 Fenêtre d'évolution du dispositif sous Solid Edge



Bibliographie

- *A* ----------

[Ati98] « Le modèle produit: une méthodologie de modélisation pour les dispositifs électrotechniques », Atienza Eric, Rapport de DEA préparé à l'INPG spécialité génie industriel, LEG (Laboratoire d'Electrotechnique de Grenoble), ST Martin d'Hères, France, Septembre 1999.

[Ati99] « Steps to an Electrical Design Environment », Atienza Eric, Bigeon Jean, Wurtz Frederic, Bel Habib Basma, IECon'99 Proceedings, Volume II, pages 815-820, Décembre 1999.

- *B* ----------

[Bhb99] «An Integrated Platform for Electrical Design », Bel Habib Basma, Wurtz Frederic, Bigeon Jean, Atienza Eric, Article issu de IECon'99 Proceedings, Volume II , pages 809-814, Décembre 1999.

[Bhb98] « Connaître l'activité de conception », BEL HABIB Basma, Rapport interne: Synthèse de la bibliographie autour de la conception, Diffusion interne 1998.

[Big94] « Processus de conception de produit et dynamique des relations dans l'entreprise et entre entreprises (une approche multi-acteurs) », J. BIGEON J-Ch MONATERI, Rapport du projet de recherches IPI Juin 94,

[Black] « Blackboard Systems », Robert ENGELMORE, Tony MORGAN, ISBN 0 201 17 431 6.

- *C* ----------

[Cao98] « CAO : êtes-vous efficace ? », Philippe BEAUFILS, Jean-françois PREVERAUD , Mierl SHERER et Hélène ANTOINE, Magazine Industries Techniques, pages 58-90, Mars 1998 numéro 791, ISBN 01506617.

[Corba] « Client/ Server Programming with JAVA and CORBA », Robert Orfali & Dan Harkey, ISBN 0-47L-24578-X.

[CS98] «Client/ Serveur », 2ème édition , Robert Orfali, Dan Harkey et Jeri Edwards. Traduction de François Leroy et Jean-Pierre Gout, ISBN

[CE99] «WWW-Based Collaborative System for Integrated Design and Manufacturing », Hsin-Chi Chang, Wen F. Lu, Xiaoqing Frank Liu, CE Concurrent Engineering : Research and Applications, Volume 7, n°4, Décembre 1999 pages 319-334.

- D ----------

[Dej95] « Processus de conception de produit et dynamique des relations dans l'entreprise et entre entreprises. Une approche multi-acteurs », Dejuan Rapport de DEA préparé à l'INPG spécialité génie industriel, LEG, Juin 1995.

- E ----------

[Els97] « Outils et structure pour la coopération formelle et informelle dans un contexte de conception holonique », Elsie Cristina CHAPA KASUSKY, Thèse préparée à l'INPG spécialité génie mécanique, LSSS (Laboratoire des Sols, Solides, Structures de Grenoble), ST Martin d'Hères, France, 1er Octobre 1997.

[Esc96] «Modélisation objet du processus de conception dans le domaine du génie électrique: application au cas de la machine asynchrone », ESCANDE Eric, Thèse préparée à l'INPG spécialité génie électrique, LEG (Laboratoire d'Electrotechnique de Grenoble), ST Martin d'Hères, France, Décembre1996.

[EtJ99] « L'entreprise JAVA: le point sur l'offensive en Europe », Informatiques Magazine hors série n°2, Avril- Mai 1999.

- F ----------

[Fau96] « Moteur asynchrone polyphasé. Conception et calcul », FAURE Alain et LARONZE Joseph, Techniques de l'Ingénieur, Traité Génie Electrique, D 568, 1, 1996.

[Fez96] « Système Expert pour la conception en électronique de puissance », FEZZANI Djamel, PIQUET Hubert, FOCH Henri, Journal de Physique III France 6, Mars 1996 pages 349-361.

[Fez97] « Expert System for the CAD in Power Electronics- application to UPS », FEZZANI Djamel, PIQUET Hubert, FOCH Henri, IEE Transaction on Power Electronics, Vol 12, n°3, Mai 1997 pages 578-586.

[Flu2] « *FLUX2D Tutorial Version 7.2x and PREFLU2D Tutorial Version 2.2* ». août 1997, CEDRAT Electrical Engineering cedrat@cedrat-grenoble.fr.

[Fra92] « La modélisation du processus », Marcel FRANCKSON, Article, Génie Logiciel & Systèmes Experts, N° 27, Juin 1992.

- G ----------

[Gen91] « COCASE. Un Système Expert d'aide à la conception d'appareillages électriques », Gentilhomme Alain, Thèse préparée à l'INPG spécialité génie électrique, LEG (Laboratoire d'Electrotechnique de Grenoble), ST Martin d'Hères, France, Mai 1991

[GI97] « Propositions pour une prospective en Productique- Génie Industriel (2ème congrès du club)», Pierre LADET, François ROUBELLAT, Extrait d'un Rapport adressé au Ministère à la date du 4Juillet 1997.

- H ----------

[Haton] « Le raisonnement en Intelligence Artificielle », Jean-Paul Haton, Najet Bouzid, François Charpillet, Livre, InterEditions, ISBN, 2-7296N-0335-2; 61-2293-1.

[Har97] « Une Approche Multi-Modèles pour la capitalisation des connaissances dans le domaine de la conception », Harani Yasmina, Thèse préparée à l'INPG spécialité génie industriel, Laboratoire industriel de Génie Industriel et de Production Mécanique - Université de Metz, France, Novembre 1997.

- I ----------

[Inria] « Connectors : a Key for Building Distributed Component-Based Architecture », Marangozov Vladimir, Bellissard Luc, Vion-Dury Jean-Yves, Riveill Michel, Article, Projet SIRAC, INRIA Rhônes Alpes, Montbonnot.

- J ----------

[Jav1.2] « JAVA 1.2 », Laura Lemay et Rogers Cadenhard, Simon & Schuster Macmillan (France), ISBN, 2-7440-0519-3.

[JavB] « JAVA Beans: Guide du programmeur », Rob Englander, O'REILLY 1997 Traduction de Bonnie Cupac, ISBN, 2-84177-038-9.

- K ----------

[Kle93] «Capturing Design Rational in Concurrent Engineering Teams», Mark Klein, Boeing computer services, IEEE Computer, Janvier 1993.

- L ----------

[Lat72] « Elaboration d'un Système Pédagogique d'Assistance à la Conception: ESPACE », Jean-Claude LATOMBE, Thèse préparée à l'INPG spécialité génie électrique, LEG (Laboratoire d'Electrotechnique de Grenoble), ST Martin d'Hères, France, Novembre 1972.

[Lau00] « Conception intégrée dans l'usage : Mise en œuvre d'un dispositif d'intégration produit-process dans une filière de conception de pièces forgées », LAUREILLARD Pascal, Thèse préparée à l'INPG spécialité génie mécanique, LSSS (Laboratoire des Sols, Solides, Structures de Grenoble), ST Martin d'Hères, France et le Centre de Recherche Innovation Socio-Technique et « Organisation industrielle et systèmes de production », 12 janvier 2000.

[Leb97] « Les Objets de la Conception », Jean-Charles LEBAHAR, Revue Sciences et techniques de la conception vol6 – n° 1, 1997.

- M ----------

[Mag1] « Using Visaul Basic for Applications Programming with MagNet », David A Lowther, Magnetics Update Winter 2000, 2000.

[Mag2] « Scripting, Optimization and MagNet », Rashid Kash, Magnetics Update Winter 2000, 2000.

[MCS] Rapport 99 de projet d'Option " Couplage du logiciel de conception PASCOSMA et du logiciel de DAO Solid-Edge via Visual Basic", Diffusion interne 99.

- P ----------

[Pra91] « Système d'Aide au Choix et à la Conception des électro-broches et des moteurs d'axes a très grande vitesse », Gilbert Pradel, Erafael Vives-Fos, Sylvain Allano, Conférence Sectorielle Intelligence Artificielle et Génie Electrique ; Onzièmes Journées Internationales Les Systèmes Experts et leurs Applications, 27-31 mai 1991.

[Pol96] «Maîtriser l'activité d'assemblage pour gagner en productivité et offrir plus de valeur ajoutée», Michel JUBIN, Charles SCHEPACZ, Charles SASSO, Journal POLE infos96 édité par le Pôle Productique Rhône-Alpes, n° 42, Mars 1996.

[Pro1] «Solid Edge Programming User's Guide. Version 1.0», Intergraph Corporation Huntsville, Alabama ISBN, 35894-0001.

- *R* ----------

[Ras99] «Optimization of Electromagnetic Devices using Computraional Intelligence Techniques», K. Rashid, J.A. Ramirez, E.M. Freeman, IEEE Transactions on Magnetics, VOL 35, N° 5, Part II of three parts; pages 3727-3729, Septembre 1999.

- *S* ----------

[Sab96] « Conception Assistée par Ordinateur en Génie Electrique: Méthodes et techniques », SABONNADIERE Jean-Claude, Techniques de l'Ingénieur, Traité Génie Electrique, D 3 585, 1996.

[Sas94] «A Knowledge-Based Environment for Linking Electromagnetic Design Tools», R. M. Sassine and D. A. Lowther, IEEE Transactions on Magnetics, VOL 30, NO5, Septembre 1994.

[Sev4] «Solid Edge Guide de l'utilisateur. Version 4 », Intergraph Corporation Huntsville, Alabama ISBN, 35894-0001

[Sho95] « Modélisation objet pour la CFAO: modèle de conception », TOLLENAERE Michel, Rapport de fin de contrat du projet SHOOD, Les Laboratoires d'Electrotechnique de Grenoble (LEG), de l'INRIA Rhônes Alpes, du Sols, Solides, et Structures, le Lméca Annecy, France, Août 1995.

[STEP] « Evaluation de la Norme de Description de Données STEP dans le cadre de la modélisation des phénomènes Electromagnétiques», MA, Rapport de DEA préparé à l'INPG spécialité génie électrique, LEG, Septembre 1998.

- *T* ----------

[Tri88] « Systèmes Experts pour la conception de moteurs asynchrones », TRICHON François, Rapport de DEA préparé à l'INPG spécialité génie électrique, LEG (Laboratoire d'Electrotechnique de Grenoble), ST Martin d'Hères, France, Septembre 1988.

[Tri91] «Modélisation du processus de conception des machines électriques», TRICHON François, Thèse préparée à l'INPG spécialité génie électrique, LEG (Laboratoire d'Electrotechnique de Grenoble), ST Martin d'Hères, France, Mars 1991.

- *W* ----------

[Wur96] «Une nouvelle approche pour la conception sous contraintes de machines électriques», WURTZ Frédéric, Thèse préparée à l'INPG spécialité génie

électrique, LEG (Laboratoire d'Electrotechnique de Grenoble), ST Martin d'Hères, France, Mai1996

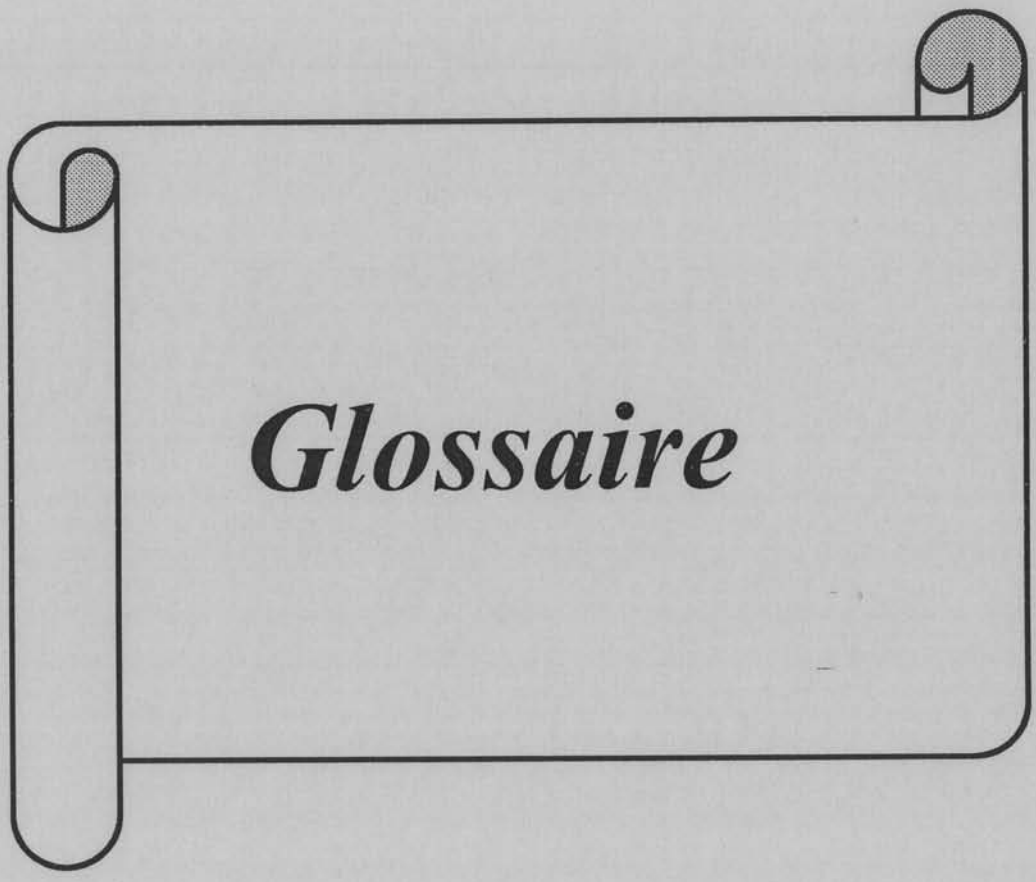
- *X* ----------

[XML] « Mise en cohérence des outils associés à la conception d'une installation électrique à l'aide d'un format unique d'échanges de données: le format XML», ROSNET, Rapport de DEA préparé à l'INPG spécialité génie électrique, LEG, Septembre 1999.

Liens intéressants

Nous fournissons la dessus l'adresse de quelques sites WEB où nous avons puisé de l'information à un moment ou un autre de la thèse. Par contre, nous ne garantissons pas la pérennité de ces adresses.

- Dictionnaire des termes JAVA: http://www.jbusiness.org/articles/dico_java/
- Les acronymes informatiques: http://www.c4i.fr/weldj/acronyme.html



Glossaire

Glossaire

- A

API :
API est l'abréviation d'Applications Programming Interface

Archétype :
Une Solution de conception définie par un jeu de paramètres fonctionnels et géométriques.

- C

Capitalisation:
Action de capitaliser; Accroître un capital par l'addition des intérêts qu'il procure (Grand Dictionnaire Hachette 1992).

Capitaliser:
Ensemble des moyens (financiers et techniques) dont dispose une entreprise industrielle ou commerciale (Grand Dictionnaire Hachette 1992).

Complexe :
1- Ce qui est complexe composé d'éléments différents 2- Chimie : Composé formé d'un ou plusieurs atomes ou d'un ion central généralement métallique, lié à un certain nombre d'ions ou de molécules.

Complexité :
Caractère de ce qui est complexe, difficile.

Complication :
Etat de ce qui est compliqué ; ensemble compliqué. La complication d'une machine.

Compliqué :
1- Composé d'un grand nombre d'éléments/ 2- Difficile à comprendre ou à exécuter.

Concept :
1\ Représentation intellectuelle d'un objet conçu par l'esprit. 2\ Définition des caractères spécifiques d'un projet, d'un produit par rapport à l'objectif ciblé.

Conception :
Action d'élaborer quelque chose dans son esprit ; résultat de cette action [Définition du petit Larousse]

Conception à faculté automatique :
Action d'élaborer un produit en s'appuyant non seulement sur les facultés intellectuelles propres au concepteur mais en s'aidant d'outils informatiques pour la recherche de la solution.

- D

DAO :
Dessein Assistée par Ordinateur (DAO) englobe l'ensemble des outils informatiques permettant de représenter une géométrie en 2D (espace à deux dimensions) ou 3D (espace à trois dimensions).

Design for X :
Modèle prenant en compte une discipline particulière qui consiste:

- soit à définir les entités du produit en prenant en compte les contraintes du domaine X dès les premières phases de conception.
- soit à laisser le concepteur faire et valider ou non les entités du produit selon les règles de X: il s'agit là « **d'Extraction For X** ».

- E

Encapsuler :
Enfermer un processus à l'intérieur d'un objet informatique afin d'en rendre le corps invisible à l'utilisateur, tout en fournissant les moyens d'accéder à ses fonctionnalités.

- F

FAO :
Fabrication Assistée par Ordinateur couvrant l'ensemble de techniques informatiques apportant une aide à la fabrication.

- I

Instancier : Créer une ou plusieurs instances d'une classe

- J

Javadoc :
Outil qui permet la génération de la documentation d'APIs Java à partir de commentaires spéciaux situés dans le code source. Javadoc génère un ensemble de pages HTML, qui contiennent entre autres la documentation des classes, des interfaces, des champs, des méthodes, des constructeurs, des héritages et des implémentations d'interfaces.

- O

Outil de Conception :
Outil de Conception couvre l'ensemble de techniques informatiques utilisées pour assister les concepteurs lors de l'activité de Conception.

- P

Portabilité :
Un programme informatique est dit portable s'il est indépendant de la plate forme: il peut tourner sur toutes les plates formes sans qu'il y ait besoin de lui apporter aucune modification ou adaptation.

Prototype d'une méthode:
la liste des paramètres d'entrées d'une méthode et de sa valeur de retour

- R

RMI (Remote Method Invocation)
Appel de méthode à distance
Technologie permettant à un objet s'exécutant sur une machine virtuelle Java ™ de faire appel aux méthodes d'un objet s'exécutant sur une autre machine virtuelle. Ces appels de méthodes peuvent se faire entre des machines virtuelles s'exécutant sur des machines différentes reliées par le réseau. RMI permet de transférer, d'une machine virtuelle à l'autre, des objets par valeur et des implémentations d'interface.

- S

Signature d'une méthode selon la définition de java:
Sa liste de paramètres d'entrées, sa valeur de retour ainsi que sa visibilité dans l'architecture (public, privé...).

Sérialisation:
Mécanisme intégré à Java qui permet le codage d'un objet (et de tous les objets qu'il référence directement ou indirectement) sous forme d'un flux d'octets. Ce mécanisme offre également la possibilité de reconstituer l'objet par décodage d'un flux d'octets généré par ce biais. La sérialisation est utilisée pour la persistance à court terme et dans les communications via les sockets ou RMI.

" Méthodologie pour le Développement de Plates Formes Intégrées dédiées à la Conception en Génie Electrique "

Résumé:

Ce travail de thèse concerne la création d'environnements de conception intégrée en génie électrique. Cette problématique est abordée d'un point de vue méthodologique. Nous avons commencé par faire une synthèse de l'activité de conception et des outils aidant le concepteur en génie électrique. A partir de cette étude, nous avons montré le besoin d'un support informatique permettant d'automatiser partiellement l'interaction entre les outils et de réutiliser les processus de conception menés par le concepteur. Pour cela, nous avons développé le concept de Plates Formes Intégrées dédiées à la conception en génie électrique. Nous avons mis en place une démarche pour réaliser des couplages réutilisables entre les outils hétérogènes de conception permettant le transfert de paramètres. Pour l'implémentation du modèle proposé, nous avons mis en œuvre un modèle Serveur d'Objets en Java en encapsulant les logiciels de Conception dans des Composants. En dernière partie, nous proposons un prototype de plate forme pour la Conception Intégrée.

Mots Clés:

Conception Intégrée	Génie Electrique	Plate Forme Intégrée
Composants Logiciels	Serveurs d'Objet JAVA	Logiciels hétérogènes

"Methodology for Developing Electrical Design Integrated Platforms"

Summary:

This work deals with the development of environments dedicated to electrical design. The first target of our project is to define a methodology to support interaction between several individual design software. First a synthesis of the design activity and a state of the art of available tools dedicated to electrical products design are made. It has been shown that there is a real need of a computer environment providing software links and reusable design process. To address this problem, we introduce the concept of Electrical Design Integrated Platform. We propose a way of coupling heterogeneous tools and transferring parameters. To implement this architecture, the JAVA_Object Server model has been carried thanks to the encapsulation of the design tools in Components. Finally, the developed prototype of Electrical Design Integrated Platform is detailed.

Key Words:

Integrated Design	Electrical Products	Integrated Platforms
Software Components	JAVA_Object Server	Heterogeneous Software

Laboratoire d'Electrotechnique de Grenoble, INPG / UJF UMR 5529 CNRS
ENSIEG, Domaine Universitaire, BP 46, F-38402 Saint Martin d'Hères Cedex, France