Une nouvelle approche fonctionnelle pour une assistance géométrique pendant les premières phases de conception de produits

Denis Pallez

THÈSE
Présentée à

L'UNIVERSITÉ DE METZ

Pour l'obtention du grade de :
DOCTEUR DE L'UNIVERSITÉ DE METZ

Spécialité : INFORMATIQUE

Denis PALLEZ

Une nouvelle approche fonctionnelle pour une assistance géométrique pendant les premières phases de conception de produits

À Metz, le 12 janvier 2000

Composition du jury :

Directeur de thèse :	*Yvon GARDAN*	*Professeur à l'université de Reims*
Rapporteurs :	*Yves BERTRAND*	*Professeur à l'université de Poitiers*
	Marie-Christine HATON	*Professeur à l'université de Nancy*
Examinateurs :	*Jean-Pierre JUNG*	*Professeur à l'université de Metz*
	Christian MINICH	*Maître de conférences à l'Université de Metz*
	F. J. A. M. VANHOUTEN	*Professeur à l'université de Twente (Pays-Bas)*

LABORATOIRE DE RECHERCHE EN INFORMATIQUE DE METZ

À ma famille,
passée, présente
et à venir.

Remerciements

Je ne pourrais commencer ce document sans remercier sincèrement les personnes qui m'ont permis d'accomplir un de mes vœux les plus chers.

Mes premiers remerciements reviennent à mon directeur de thèse, Monsieur le professeur Yvon Gardan de l'université de Reims qui a bien voulu m'accueillir au sein de son laboratoire et de son équipe de recherche. Il m'est difficile de trouver les mots justes pour le remercier aussi lyriquement que je le voudrais. Toutefois, j'aimerais au moins le remercier pour sa grande disponibilité malgré les kilomètres qui nous séparent, pour la confiance qu'il m'a accordée tout au long de ces années de recherche, pour tous les conseils avisés qu'il a pu me communiquer, pour m'avoir montré la voie de la recherche et m'avoir donné la possibilité de m'exprimer…

Ce manuscrit est l'occasion, pour moi, de lui exprimer ma profonde gratitude. Je ne saurais comment lui rendre l'égal de ce qu'il m'a donné.

Je tiens également à remercier Madame le professeur Marie-Christine Haton de l'université de Nancy et Monsieur le professeur Yves Bertrand de l'université de Poitiers, pour l'intérêt dont ils ont fait preuve à l'égard de mon travail en acceptant d'être rapporteurs de cette thèse. Je tiens à remercier particulièrement Yves Bertrand pour ses conseils clairvoyants sur le modèle des G–Cartes qui m'ont permis de le faire connaître à d'autres.

Par ailleurs, je tiens à remercier Messieurs les professeurs F.J.A.M. VanHouten de l'université de Twente (Pays–Bas) et Jean-Pierre Jung de l'université de Metz pour avoir accepté de juger mon travail.

Ensuite, il est tout naturel de remercier Christian Minich qui a dû subir mes questions et mes angoisses au quotidien. Merci aussi à sa « foultitude » de remarques pertinentes qui ont permis d'améliorer la qualité de mes documents, en particulier celui-ci. Merci encore pour sa patience, sa grande disponibilité, ses conseils, ses critiques, son soutien, son équité …

Il m'est inconcevable de ne pas remercier Estelle Perrin qui était toujours là pour me soutenir dans les

moments les plus difficiles. Elle a le don de vous montrer le chemin qu'il faut suivre pour éviter un maximum d'embûches. Je la remercie aussi pour son amitié et l'ambiance qu'elle a su faire régner dans le bureau.

Je remercie Yann Lanuel pour ses précieux conseils sur les langages à objets qui m'ont permis de développer la maquette informatique plus rapidement.

Je tiens à souligner que j'ai bénéficié d'un encadrement que je qualifierai d'hors du commun et d'exceptionnel, que tout doctorant rêve d'avoir. Pour cela, je tiens à remercier à nouveau les personnes concernées : Yvon Gardan, Christian Minich, Yann Lanuel et Estelle Perrin.

Je remercie aussi les autres personnes du laboratoire qui savent faire régner une ambiance joviale et conviviale qui représente un cadre de travail très agréable : Ahmed, Benoît, Catherine, Dominique, Frédéric, Gaby, Isabelle, Jocelyne, Martine, Myriam, Robin, Stéphane, Vincent …

Je remercie la société Auchan de Semécourt qui a su m'aider financièrement lorsque j'en avais le plus besoin.

Enfin, je remercie mes parents qui m'ont toujours soutenu et m'ont fait bénéficier des meilleures conditions pour la réussite de mes études.

Je finis cette page de remerciements en ayant une pensée pour Christel qui m'a toujours soutenu et aidé dans la phase de rédaction de ce document quoi qu'elle en dise, mais aussi et surtout, pour l'amour qu'elle m'apporte.

Table des matières

Présentation du sujet

L'avènement de l'informatique a révolutionné le monde industriel. Quel que soit le métier concerné, nombre de tâches, effectuées jusqu'alors de manière manuelle, ont été transcrites en un programme informatique. En l'occurrence, le domaine de la conception de produits manufacturiers (vélo, voiture, table, moule de fonderie, micro-ondes …) est pourvu de logiciels de CAO[1] permettant de modéliser, tester, visualiser (…) le produit en cours de conception de manière virtuelle. Plus les logiciels sont munis de fonctionnalités, plus les personnes des bureaux d'études en demandent de nouvelles. Ainsi, le logiciel de CAO consistait principalement à modéliser la géométrie du produit alors qu'actuellement, même si la géométrie reste encore primordiale, ces mêmes logiciels tendent vers une modélisation plus « intelligente ». Cela s'explique notamment par une compétition industrielle qui oblige chaque entreprise à construire des produits de meilleure qualité, à un coût réduit, et dans les meilleurs délais. Malgré le grand nombre de fonctionnalités que proposent ces logiciels, il est vrai que certaines tâches doivent être encore automatisées. En effet, même s'ils procurent aux concepteurs des bureaux d'études le moyen de modéliser le produit en trois dimensions (définition des différents composants du produit, définition de l'assemblage de ces composants …), ils ne permettent pas de modéliser les informations qui ont amené à choisir une telle géométrie ; ils ne fournissent pas non plus ou très peu d'assistances à la conception de la géométrie. En effet, dès lors qu'un concepteur connaît la forme de l'objet qu'il conçoit, il peut sans aucun doute utiliser ces logiciels informatiques ; cependant, à partir du moment où il n'arrive pas à s'imaginer la forme de son produit, il ne peut en aucun cas utiliser de tels logiciels.

Notre sujet a consisté à réaliser pour le laboratoire une première étude dans le domaine de la synthèse automatisée ou assistée de la forme d'un produit à partir de ses informations fonctionnelles. Nous travaux ont mené à l'élaboration d'une méthode d'assistance pendant les phases amonts de création d'un produit. Par exemple, à long terme, on souhaite qu'un système informatique propose au concepteur plusieurs solutions géométriques à partir des informations suivantes : « je désire concevoir un véhicule, beau, économique, pouvant transporter au moins cinq personnes. La beauté du véhicule devra être considérée comme une notion importante par rapport aux autres besoins. ».

[1] Conception Assistée par Ordinateur

Ainsi, comme nos travaux visent à transcrire les fonctions d'un produit en formes, ce manuscrit est organisé de la façon suivante : étude des notions existantes liées aux fonctions, étude des notions existantes liées aux formes, proposition d'une méthode de conception visant à passer des fonctions à des formes et validation de cette approche par un développement informatique et une application industrielle. Dans ce qui suit, nous décrivons de manière plus précise le contenu de chaque partie.

En premier lieu, nous étudions les outils méthodologiques et informatiques disponibles par les ingénieurs des bureaux d'études pour la conception de produits manufacturiers en orientant notre recherche sur la notion de fonction (Chapitre 1). En particulier, nous présentons quelques méthodes de conception (§2.2) comme l'analyse de la valeur, Structured Analysis and Design Technique ou Function Analysis System Technique et nous étudions, pour chacune une éventuelle automatisation. Néanmoins, comme les informations manipulées par ces méthodes sont majoritairement exprimées en langage naturel, le problème du passage fonction/forme apparaît d'autant plus difficile. Quelques outils informatiques d'assistance existent et certain d'entre eux sont présentés (§2.3), mais leur application n'en est que très parcellaire (éditeur de formalisme, simulation du comportement du produit …). Notre objectif étant de concevoir une forme à partir des spécifications fonctionnelles du produit, il nous importe, dans un premier temps, d'étudier la nécessité de modéliser un produit principalement par ses fonctions (§3). Dans un deuxième temps, nous étudions les différentes propriétés des fonctions (§4.2) ainsi qu'une typologie de ces dernières (§4.3) pour tendre vers une structure de données informatique : la hiérarchie fonctionnelle (§5). Nous en donnons un exemple avec le modèle FBS[2] (§5.2).

Le passage fonction/forme implique indubitablement l'étude d'une modélisation géométrique : c'est l'objectif de la deuxième partie (Chapitre 2). Nous décrivons, avant tout, les modèles informatiques les plus communs permettant de représenter la géométrie d'un objet (§2.1). Ensuite, nous détaillons les méthodes de construction permettant de fournir les informations au modèle choisi pour représenter la forme (§2.2). Elles seront présentées suivant une complexité croissante du niveau sémantique des informations manipulées. Plus la méthode est automatique et plus le niveau sémantique des informations est élevé, plus les formes solutions seront nombreuses. Il est alors nécessaire de sélectionner celles qui conviennent le mieux au concepteur, c'est-à-dire celles qui satisfont le mieux les spécifications du produit (§3). Pour cela, nous introduisons la notion d'espace de solutions (§3.1) qui permet de modéliser l'ensemble des solutions fournies par la méthode de construction. Ensuite, l'utilisation d'un parcours de l'espace des solutions associé à une estimation des solutions (§3.2) permet d'extraire de cet espace les solutions les plus prometteuses. Comme le passage des fonctions à une forme se révèle complexe, la synthèse automatique d'une forme satisfaisant idéalement toutes les spécifications du produit dès la première génération est illusoire. C'est pourquoi, les solutions, même les plus prometteuses, doivent être modifiées, en l'occurrence par le concepteur, pour améliorer leur niveau de satisfaction (§3.3).

Dans la troisième partie (Chapitre 3), nous présentons succinctement plusieurs méthodes de passage des fonctions du produit à des formes (§2) avant de n'en développer qu'une de manière détaillée (§3). Pour cette dernière, nous définissons différentes notions relatives aux spécifications (§3.1), entre autres les paramètres intermédiaires (§3.2), utilisés pour définir ces spécifications. À partir d'un certain nombre

[2] Function – Behaviour – State en anglais

d'hypothèses de travail (§3.3), nous nous proposons de construire l'espace des formes solutions à partir des spécifications définies au préalable et de formes primitives contenues dans une bibliothèque (§3.4). Comme nous l'aurons justifié dans le chapitre précédent, il est nécessaire de définir l'estimation des formes solutions : cela est fait pour les formes primitives provenant de la bibliothèque (§4.1) et est étudié pour des formes plus élaborées que nous appelons combinées (§4.2). Enfin, comme les formes solutions peuvent être comparées les unes aux autres, nous étudions la possibilité d'utiliser deux types d'algorithmes pour parcourir l'espace des solutions et donc déterminer les solutions les plus prometteuses : les algorithmes stochastiques (§5.1) ou par échantillonnage (§5.2). Enfin, en fonction de la méthode proposée, nous présentons les différentes possibilités d'intervention du concepteur lors de cette synthèse de forme (§6).

La méthodologie de synthèse automatique de formes à partir de spécifications est étayée (Chapitre 4), dans un premier temps, par une maquette informatique (§2), développée dans un langage orienté objets, et dans un deuxième temps par une application industrielle (§3). Dans le premier cas, nous présentons l'analyse orientée objets (§2.1) à l'aide du formalisme Unified Modeling Language (diagrammes des paquets et de classes) ainsi qu'une rapide présentation de la maquette (§2.2). Dans le deuxième cas, nous expliquons brièvement le problème en fonderie qui consiste à concevoir un moule de fonderie (§3.1). Ensuite, nous discutons sur la manière d'appliquer notre méthodologie dans ce domaine de conception. Cela nous permet d'améliorer notre approche dans la mesure où nous utilisons des règles métiers propres au domaine.

Nous terminons ce manuscrit en présentant un bilan du passage fonction/forme et en proposant un certain nombre de perspectives faisant appel à d'autres domaines de recherche que la conception assistée par ordinateur.

Chapitre 1 Outils méthodologiques ou informatiques d'assistance à la conception fonctionnelle

« La critique est aisée, mais l'art est difficile »
Destouches

1 Introduction

La volonté de capitaliser le savoir-faire de l'entreprise et de réduire les temps de production font que l'on aborde maintenant véritablement les systèmes de conception dans l'optique d'une modélisation fonctionnelle, avec l'objectif d'assister les différents concepteurs intervenant sur le produit dès les toutes premières phases de sa réalisation, à savoir l'élaboration du cahier des charges et la décomposition fonctionnelle du cahier des charges.

Le but de ce chapitre est d'établir un panorama des techniques déjà utilisées dans ces deux phases pour en étudier l'éventuelle automatisation et de faire un point sur les recherches déjà entreprises. Ce chapitre a fait l'objet d'une publication [Minich et al. 99].

Dans la bibliographie, les termes de processus, méthode, méthodologie, démarche sont souvent employés pour décrire sensiblement les mêmes choses. Pour éviter d'étudier les différences subtiles entre chaque terme comme le fait [Evbuomwan et al. 96], nous convenons du vocabulaire suivant : une *méthode* est la donnée d'une démarche et d'un formalisme ; la *démarche* dit comment mener la conception, le *formalisme* permet au concepteur de décrire d'une manière ou d'une autre le fruit de ses réflexions. Par ailleurs, nous nommerons *historique de conception* la séquence d'opérations réalisées par le concepteur.

On peut imaginer que l'ordinateur fournira dans les phases amont de la conception plusieurs niveaux d'assistance. On peut d'abord envisager que les systèmes de CAO offrent à leurs utilisateurs le moyen d'enregistrer leur méthode de conception. Ceci apparaît comme un outil incontournable de la pérennisation du savoir-faire. Nous présentons donc dans la deuxième partie les notions liées aux méthodes de conception, à savoir ce que l'on entend classiquement par cahier des charges, les principales

méthodes utilisées dans les bureaux d'études et les possibilités de les automatiser.

Une autre avancée consistera à conserver un historique détaillé du cycle de vie du produit, depuis la rédaction du cahier des charges jusqu'à la fabrication, la maintenance et le recyclage. Or, la mémorisation des activités menées dans les premières phases de conception implique la capacité, pour le système, d'enregistrer ce pour quoi le produit est prévu, ce que nous appelons les *fonctions* du produit. La troisième partie présente les avantages d'une modélisation des fonctions du produit et la suivante recense les différentes classifications de fonctions proposées dans la littérature, étant donné que ces classifications sont supposées déboucher sur une structure informatique.

Parmi les éléments qui composent l'historique de conception, la hiérarchie fonctionnelle tient une place importante [Gardan et al. 97]. Elle décrit la manière dont un ensemble de fonctions initiales (le cahier des charges) a été décomposé en des sous-ensembles de fonctions de plus en plus simples et la manière dont est apparue la forme, d'abord vague puis de plus en plus précise.

La hiérarchie fonctionnelle fait l'objet de nombreuses réflexions. Elle apparaît comme un moyen de produire des variantes du produit et prend alors une importance particulière dans le domaine de la conception routinière. Des études visent également à réaliser de manière automatique cette hiérarchie ou à fournir une assistance lors de son établissement. On peut par exemple citer l'étude du modèle « FBS » réalisée par un groupe de chercheurs japonais, composé principalement des professeurs Umeda, Tomiyama, Yoshikawa, Takeda et Shimomura [Tomiyama et al. 93]. La cinquième partie est consacrée à la hiérarchie fonctionnelle et en particulier au modèle FBS.

2 La conception de produits

[Afav 89] définit le terme produit par « tout ce qui est l'objet (matériel ou immatériel) d'une activité (financière, commerciale, technique, industrielle, de service, familiale) et répondant aux besoins des utilisateurs ».

La conception de produits est un domaine très vaste dans la mesure où le terme « produit » rassemble énormément de notions. Par exemple, un vêtement est considéré comme un produit dans les entreprises de confection ; la confection d'un wagon bar est considérée comme une conception de produit pour la SNCF ; un pétrolier, un four industriel à résistances, un siège avant de voiture, un véhicule, un logiciel sont autant d'autres produits. Cette variété rend très difficile la réalisation d'un outil de conception automatique utilisable dans tous les secteurs. Il faudrait pour cela emmagasiner les connaissances de tous les métiers, trouver des invariants dans les méthodes de conception … Il est plus réaliste de fournir aux concepteurs d'un secteur et d'une société donnés le moyen d'enregistrer leur méthode de conception. Cela a deux avantages : on standardise les conceptions menées par la société et on assure la transmission du savoir-faire à la prochaine génération.

Dans ce qui suit, nous rappelons pourquoi on ne peut se contenter du cahier des charges comme d'un modèle du produit, quelles sont les méthodes utilisées par les concepteurs et quels outils d'assistance sont disponibles ou spécifiés.

2.1 Le cahier des charges client

Le cahier des charges client est le point de départ d'une conception. Il établit la liste des fonctions du

produit. Il représente certes un modèle du produit mais on ne peut l'exploiter comme support de traitements du type visualisation ou calcul de résistance à l'effort. En effet, il est principalement exprimé en langage naturel. Or on ne sait pas encore réaliser une analyse sémantique d'un texte. On pourrait se contenter d'une analyse syntaxique comme on le fait pour vérifier que les prémisses d'une règle apparaissent dans la base de faits d'un système expert, comme on le fait aussi en conception orientée objets [Abbott 83] en associant un nom à une classe, un verbe à une méthode, un adjectif à une donnée membre, etc. Cependant, les traitements restreints à un niveau syntaxique sont vite limités. Par exemple, dans un contexte automobile, un volant possède plusieurs significations et même en se référant au contexte il est difficile de faire un choix parmi les interprétations possibles (volant de direction, volant d'inertie, volant magnétique, volant de sécurité …).

De plus, les informations que le cahier des charges renferme sont presque toujours incomplètes, ce qui en fait un énoncé sous contraint et ce qui oblige les concepteurs à fournir de nouvelles informations en cours de conception.

En conséquence, de manière générale, on ne peut directement tirer parti du cahier des charges ; il reste nécessaire de le raffiner sous forme d'une décomposition fonctionnelle. Nous pouvons citer deux exemples qui illustrent le besoin de raffiner les spécifications purement fonctionnelles pour envisager une quelconque automatisation : la conception de circuits VLSI[3] et la génération de compilateurs. Dans le premier cas, les spécifications peuvent être traduites dans un formalisme accessible à la fois au concepteur et au système informatique et sont relativement peu nombreuses, ce qui diminue la probabilité d'en oublier. Il en va sensiblement de même dans l'autre cas puisqu'il suffit de fournir la grammaire du langage et les règles de production du code objet. Dans ces deux cas, plus le formalisme offre de possibilités de description, plus le concepteur peut fournir de précisions et donc plus la génération de la solution de conception est automatique.

2.2 Méthodes de conception

Les systèmes de CFAO[4] actuels fournissent peu d'assistance pendant les premières phases de conception, qui sont pourtant primordiales pour la qualité du produit. Pour systématiser la tâche des concepteurs, des méthodes manuelles sont disponibles. Nous résumons celles que nous pensons être les plus significatives car une liste exhaustive des méthodes existantes sortirait du cadre de ce document ; nous décrivons brièvement la démarche qu'elles conseillent, le formalisme qui leur est associé et évoquons la faisabilité d'une automatisation.

2.2.1 Méthode de l'analyse de la valeur

L'analyse de la valeur [Afav 89, Afnor 84] est une méthode bien connue de l'ensemble des concepteurs. Elle permet de déterminer les fonctions du produit à concevoir en caractérisant les liens qui existent entre le produit lui-même et son environnement. Le principe de la méthode est de faire un inventaire aussi exhaustif que possible des éléments environnant le produit, pour ensuite en déduire les fonctions que le produit doit satisfaire. Cette technique identifie plusieurs types de fonctions : les fonctions de services, « exprimant l'action attendue du produit sur un élément du milieu extérieur au bénéfice d'un autre élément

[3] Very Large Scale Integration en anglais

[4] Conception et Fabrication Assistées par Ordinateur

de ce milieu » [Afav 89], les fonctions contraintes, « traduisant des réactions, des résistances ou des adaptations à des éléments du milieu extérieur » [Afnor 85a] et les fonctions techniques, « propres au produit et attachées à un principe ou à une solution technique » [Afav 89].

La norme AFNOR préconise d'utiliser comme formalisme le diagramme « Pieuvre » [Afnor 85b] pour représenter ces informations. Il se présente sous la forme d'un graphique permettant d'identifier et de représenter les relations entre le produit et son environnement (cf. Figure 1-1).

Cette technique est satisfaisante pour la première étape de l'analyse fonctionnelle, c'est à dire l'expression fonctionnelle du besoin, mais elle reste insuffisante pour décrire de manière plus précise le produit ou les relations entre les fonctions du produit. Par exemple, il est impossible de préciser qu'une fonction est conséquence d'une autre.

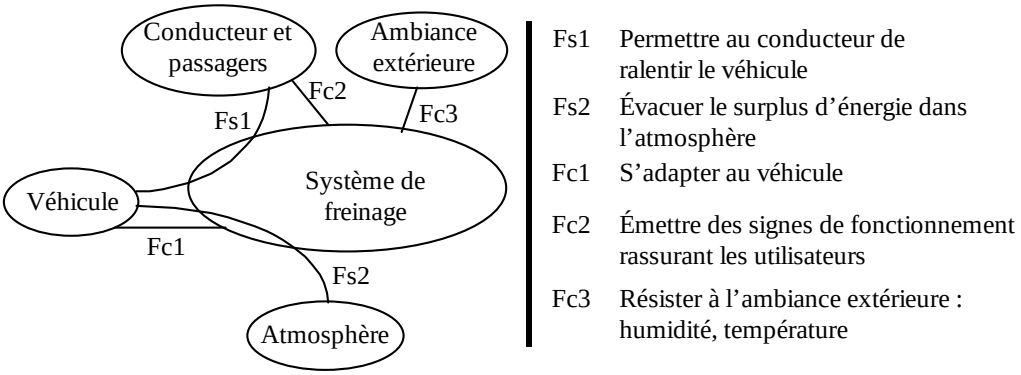


Figure 1-1. *Diagramme Pieuvre d'un système de freinage [Constant 96]*

Les fonctions sont représentées par un verbe et des compléments. C'est un premier facteur qui rend difficile une automatisation de l'analyse de la valeur étant donné qu'elle nécessiterait une interprétation sémantique des fonctions, difficilement réalisable à ce jour. En effet, des correcteurs orthographiques et grammaticaux sont déjà disponibles. Néanmoins, ils connaissent de nombreux problèmes car ils ignorent la signification des termes à corriger. Par ailleurs, on voit difficilement comment un programme pourrait s'acquitter des différentes phases de la démarche (inventaire des éléments de l'environnement, identification des fonctions et répartition en services, contraintes …).

D'autres méthodes mettant mieux en évidence la décomposition et les liens entre fonctions sont présentées dans les deux prochains paragraphes.

2.2.2 Méthode SADT

La méthode SADT[5] est munie d'un formalisme graphique permettant de représenter une suite logique de diagrammes d'activité (actigrammes) et de données (datagrammes). Le datagramme décrit les systèmes de traitement de l'information ; il est plus utilisé en génie logiciel qu'en génie mécanique. L'actigramme est obtenu en décomposant récursivement le produit en modules d'activités qui représentent le produit du point de vue de son fonctionnement. Un module est caractérisé par des données en entrée, des données en sortie, une fonction du type « faire quelque chose », qui transforme les données d'entrée en données de sortie, et des données de contrôle agissant sur le module (cf. Figure 1-2). Ces fonctions sont classiquement appelées « fonctions de transformation ».

Cette méthode est bien adaptée pour une conception de systèmes de manière hiérarchique ; elle est très

[5] Structured Analysis and Design Technique en anglais

proche de la notion de décomposition fonctionnelle et de ce point de vue elle améliore le principe du modèle pieuvre. Elle a en outre le mérite de matérialiser les échanges de flux d'énergie entre les différents composants.

L'automatisation de la décomposition qu'elle implique apparaît difficile pour les mêmes raisons que celles de l'analyse de la valeur.

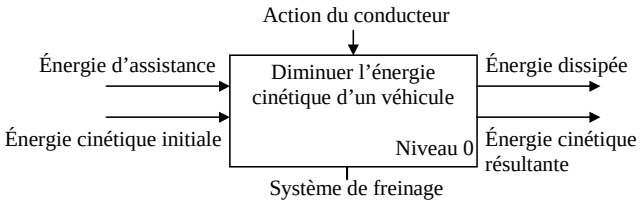


Figure 1-2. *Exemple d'actigramme d'un système de freinage [Constant 96]*

2.2.3 Méthode FAST

La démarche associée à la méthode FAST[6] consiste à hiérarchiser les fonctions du produit en répondant aux questions pourquoi, comment et quand [Afnor 85b]. Les questions « pourquoi » permettent de remonter au besoin fonctionnel et d'articuler entre elles les fonctions de service et techniques au sens de l'analyse de la valeur. Les questions « comment » permettent de mettre en avant des solutions technologiques. Dans le cas d'une reconception, elles permettent une remise en cause des principes et des solutions adoptées [Afav 89]. Les questions « quand » aident aux choix lorsqu'une fonction a plusieurs réalisations. Le formalisme FAST se présente sous la forme d'un graphe et/ou (cf. Figure 1-3). Le connecteur « et » précise que toutes les fonctions connectées doivent être réalisées ; par contre le connecteur « ou » permet de modéliser le fait qu'il y ait différentes façons de réaliser une même fonction.

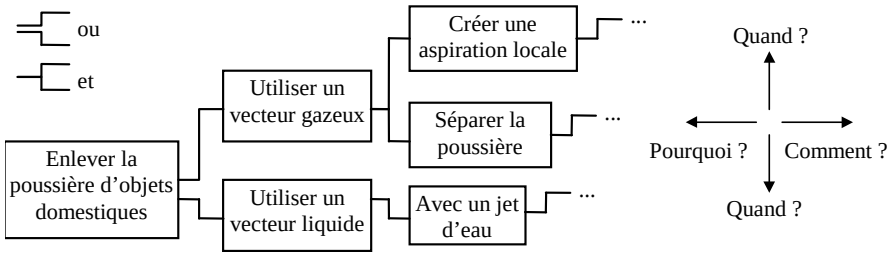


Figure 1-3. *Diagramme FAST d'un aspirateur[Afav 89]*

Cette méthode offre une organisation hiérarchique des fonctions du produit ; l'assistance informatique existante consiste simplement à modéliser le graphe et/ou et aucunement à aider le concepteur à répondre aux questions pourquoi, quand et comment dans la mesure où l'information manipulée est textuelle.

2.2.4 Autres méthodes

[Shah et al. 95] propose une vision de la conception structurée en cinq étapes nommées conceptions fonctionnelle, conceptuelle, intégrée, détaillée et analyse d'ingénierie. [BenAmara et al. 96] décompose une conception de manière légèrement différente puisqu'il ne distingue que trois étapes. En revanche, le fait qu'il cible son étude sur les mécanismes lui permet de préciser davantage chacune d'elles en mettant

[6] Function Analysis System Technique en anglais

l'accent sur leur aspect cyclique. La première étape, dite de spécification, a pour objectif de valider les besoins du client et de proposer un principe de fonctionnement du produit. Elle consiste donc à définir les fonctions de service et les fonctions contraintes, selon la méthode de l'analyse de la valeur (cf. 2.2.1). L'étape suivante, la conception préliminaire, consiste à réaliser une énumération des chaînes cinématiques possibles, à les classer et à valider les solutions proposées par le calcul cinématique. La dernière étape, la conception détaillée, consiste à réaliser le mécanisme à l'aide de composants standards en « habillant » la chaîne cinématique. Les auteurs soulignent que les choix effectués à une étape peuvent être validés à cette même étape ou alors réfutés dans une étape ultérieure (cf. Figure 1-4). Ils appellent cette méthode, la conception par fonctionnalités.

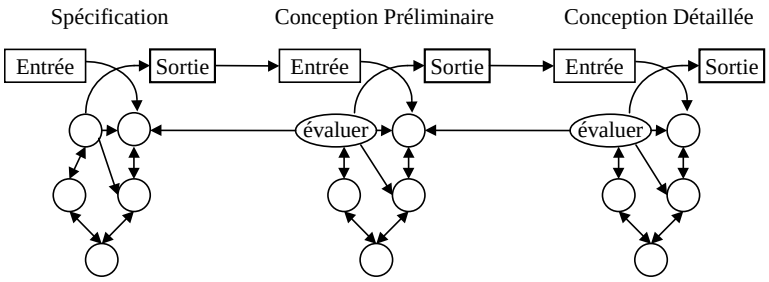


Figure 1-4. *Méthode de conception de [BenAmara et al. 96]*

La méthode de [Shimomura et al. 95, Takeda et al. 96], appelée FEP (Functional Evolution Process), met moins l'accent sur des phases de conception que sur l'aspect récursif de la conception, qui consiste alors essentiellement à exprimer récursivement une fonction comme une combinaison de sous-fonctions. La méthode s'appuie par ailleurs sur un modèle de produit, développé par la même équipe et nommé FBS, pour Function – Behaviour – State. Un produit y est décrit à trois niveaux : le premier recense ses fonctions. Le deuxième spécifie comment réaliser ces fonctions par des comportements, ce qu'en première approximation, on pourrait appeler le mode de fonctionnement du produit. Le dernier niveau décrit chaque comportement comme une séquence d'états des constituants du produit, les transitions d'états étant justifiées par des lois physiques stockées de manière explicite. Ce modèle sera décrit en détail en 5.2.

La méthode FEP fait évoluer la description FBS du produit en itérant sur trois étapes :

- la description fonctionnelle (Functional Description) est l'étape au cours de laquelle le concepteur choisit une fonction et la lie à d'autres fonctions. Par exemple, il décompose une fonction en plusieurs sous-fonctions, comme on pourrait le faire dans les méthodes FAST ou SADT ;

- le concepteur s'étant fait une idée précise de la fonction sur laquelle il travaille, il peut alors lui associer un comportement et décrire les lois physiques et les composants qui lui correspondent (Functional Actualization) ;

- on évalue enfin avec quel niveau de qualité le comportement réalise la fonction (Functional Evaluation). Cette tâche peut être accomplie de manière automatique si l'on dispose de puissants simulateurs mais dans l'état actuel du projet FEP, elle n'est que partiellement assistée (cf. 5.2.2). Si le comportement est validé, on répète ces trois étapes pour une autre fonction, sinon on les répète pour la même fonction.

Le modèle FBS final représente la décomposition fonctionnelle du produit. Si l'on conservait en outre toutes les tentatives infructueuses, on obtiendrait un historique complet de la conception. Une étude est

menée en ce sens par les membres de l'équipe travaillant sur les modèles FBS/FEP[7].

2.3 Outils informatiques d'assistance actuels et à venir

Nous exposons dans ce qui suit les outils informatiques d'assistance à la conception que nous avons recensés ou qui font l'objet de recherches actuelles.

2.3.1 Éditeurs de formalismes

Plusieurs méthodes sont munies d'un formalisme et l'on trouve des éditeurs de schémas exprimés dans ces formalismes. On peut se référer par exemple au logiciel CdCF – Produit, pour Cahier des Charges Fonctionnel du Produit, édité par la société TDC [Tdc], permettant, entre autres, de conserver de manière structurée les informations obtenues par le concepteur lors de l'élaboration du modèle Pieuvre.

Dans le contexte de la conception de logiciels, on dispose par exemple de l'atelier Rational Rose [Quatrani 98] qui, en plus d'assister la réalisation des diagrammes (de classes, des catégories, d'interaction …), construit automatiquement le squelette de l'application avec un niveau de détail qui est fonction de la quantité d'informations fournies par le concepteur au travers des diagrammes.

2.3.2 Simulation du comportement des produits

Les méthodes de conception permettent de faire évoluer la description du produit vers des entités de plus en plus précises. Or, il arrive que l'être humain se trompe ou ne sache pas trouver la meilleure solution sans en essayer plusieurs, surtout pour un problème aussi complexe que la conception de produits. Il faut donc lui fournir des outils de simulation afin qu'il ait la possibilité d'évaluer au plus tôt les choix qu'il a faits pour les confirmer ou les infirmer.

Or, de manière quasi systématique, une conception passe par une description du fonctionnement du produit selon différents points de vue (cinématique, hydraulique …). Des outils de modélisation et de simulation de systèmes dynamiques (chaîne cinématique, chaîne hydraulique, circuit électrique …) sont donc indispensables. À ce niveau, les possibilités d'automatisation sont bien plus importantes qu'à d'autres parce que les entités manipulées (formules, paramètres, équations, lois physiques …) se prêtent bien à une informatisation ; cela justifie l'existence de nombreux outils de simulation dont certains sont évoqués ci-dessous.

La simulation du fonctionnement d'un mécanisme exige sa description dans un certain formalisme. Parmi tous les formalismes disponibles, les Bond-Graphs sont fréquemment cités comme un excellent moyen de représenter les systèmes de puissance, c'est-à-dire les flux d'énergie entre systèmes ou composants d'un système [Umeda et al. 97, Rosenberg 93]. Ils ont néanmoins l'inconvénient de se limiter aux seules fonctions de transformation, au même titre que la méthode SADT. Les Bond-Graphs ne gèrent pas non plus la géométrie des composants comme l'encombrement [Constant 96]. Ils restent toutefois très utilisés.

L'université de Twente travaille à un logiciel permettant à un concepteur de décrire l'architecture d'un mécanisme sous la forme d'un schéma physique idéal, composé d'entités abstraites détachées de tout domaine ; on inclut par exemple un organe chargé de réaliser une attraction [Salomons 98]. Ce schéma est ensuite automatiquement traduit en des Bond-Graphs équivalents, cette fois dépendants de domaines tels

[7] information fournie par M. Yoshioka lors de son passage au laboratoire au mois d'août 1999.

que l'hydraulique, l'électricité etc. L'attraction peut par exemple être réalisée par un vérin hydraulique, par un ressort mécanique ou par un champ magnétique. Les différents Bond-Graphs sont soumis au simulateur développé dans cette même université (20-Sim [Twente], cf. Figure 1-5) pour être simulés, évalués et acceptés ou rejetés.

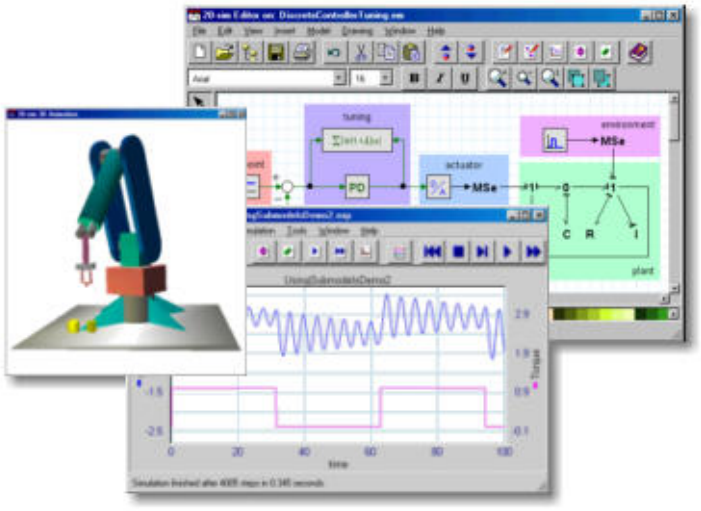


Figure 1-5. *Logiciel 20Sim de l'université de Twente [Twente]*

D'une manière comparable, un modèle FBS (Function – Behaviour – State) conserve des phénomènes physiques et les lois qui les régissent [Kief, Tomiyama et al. 96]. D'un point de vue théorique, il doit donc autoriser les mêmes simulations qu'un Bond-Graph. Une autre importante potentialité du concept est d'offrir la possibilité de mener un raisonnement qualitatif [Kiriyama et al. 91, Ishii et al. 93], ce qui ne semble pas à la portée des simulateurs classiques. Le raisonnement qualitatif[8] [Forbus 93, Veerkamp et al. 93, Forbus 84] offre des techniques de raisonnement à un niveau conceptuel sans références à des valeurs numériques précises. On peut ainsi prédire qu'en plaçant sur une flamme un récipient contenant un liquide, on élèvera la température du liquide, qui finira par s'évaporer. On n'exploite pas pour cela de formules, on se contente de principes physiques.

Cette démarche présente plusieurs avantages par rapport aux techniques conventionnelles d'analyse quantitative [Iwasaki 97] :

- elle n'utilise pas de valeurs numériques exactes ni mêmes, dans la plupart des cas, de formules ;
- elle permet de déduire plusieurs scénarios, éventuellement incompatibles, d'une configuration initiale. On engendre ainsi un arbre ou un graphe de scénarii possibles. Dans l'exemple du récipient qui chauffe, si la température d'ébullition du liquide est supérieure à la température de fusion du récipient, un scénario alternatif à l'évaporation du liquide est la détérioration du récipient et l'extinction de la flamme ou la combustion du liquide s'il est inflammable. Il faut savoir que les scénarios déduits appartiennent toujours à des scénarios modélisés ;
- enfin, l'interprétation des résultats d'une simulation qualitative est plus aisée que ceux d'une simulation quantitative car ils sont plus proches du langage naturel.

[8] Qualitative Process Theory (QPT)

[Pham et al. 97] confirme l'intérêt d'un raisonnement qualitatif sauf dans les phases finales de la conception, lorsque des valeurs précises sont requises.

2.3.3 Outils de conception par analogie

La conception par analogie a pour objectif de concevoir un produit en essayant de voir si ses spécifications ou des spécifications proches n'ont pas déjà trouvé de solutions lors de conceptions précédentes.

[Goel 97] caractérise la conception par analogie à l'aide de quatre questions dans le cas où le produit est décrit à l'aide d'un formalisme FBS :

- le « Pourquoi » concerne la tâche ou le but de conception pour lesquels l'analogie est utilisée. Par exemple, l'analogie peut servir à trouver une solution candidate au problème de conception, à modifier une conception initiale, à compléter une conception partielle, à interpréter et élaborer le problème de conception, à décomposer le problème …

- le « Quoi » se rapporte au contenu des connaissances transférées, c'est-à-dire les informations d'un ancien problème de conception utilisées pour satisfaire un nouveau problème. La réponse à cette question dépend principalement de la première question. Ainsi, la connaissance transférée peut être de la connaissance sur le problème de conception, la solution de conception, le domaine impliqué, les stratégies utilisées pour résoudre le problème …

- le « Comment » recouvre les techniques de comparaison du problème à résoudre avec les problèmes déjà résolus, et de transfert de l'information. L'auteur présente les abstractions génériques comme un moyen de comparer les problèmes et de transférer de l'information. Ces abstractions expriment la structure des relations entre les objets de conception (relations géométriques, topologiques, temporelles, causales ou fonctionnelles) ou entre les méthodes de conception ;

- le « Quand » concerne les contrôles stratégiques de fonctionnement, c'est-à-dire quand acquérir une connaissance extérieure.

Remarquons que cette technique est similaire à la méthode FAST dans le sens où les mêmes questions sont posées. Toutefois, l'ambition de la conception par analogie est d'être plus générale : elle peut être appliquée à plusieurs phases de la méthode de conception, elle peut aussi avoir plusieurs finalités comme l'extraction de méthodes, d'informations concernant le produit …

Ses fondements théoriques sont ceux de l'intelligence artificielle et de l'apprentissage. Ces disciplines n'ont pas atteint un niveau tel que l'on puisse réellement simuler le raisonnement d'analogie humain, si bien que les réalisations citées par l'auteur sont encore très parcellaires (Logiciels Syn [Börner et al.], Dssua [Qian et al. 92], Kritik [Goel et al.] et Ideal [Bhatta et al. 96]).

2.3.4 Vers une automatisation de la conception

Quelle que soit la méthode envisagée, une automatisation de la conception se heurte à la nécessité d'une compréhension d'un énoncé rédigé en langage naturel, ce qui reste encore hors de portée du monde informatique. Étudier d'autres méthodes ne ferait que confirmer cette impression (maison de la qualité – QFD[9] –, modèle en V de l'AFSIQ). L'esprit est toujours de dire « quoi » faire mais pas « comment » le

[9] Quality Function Deployment (cf. Chapitre 2, §3.2.2)

faire. La nuance est parfois subtile : dire « quoi » faire peut consister à énoncer une séquence de tâches. Si l'on pouvait détailler récursivement le « quoi » faire de chacune de ces tâches, on aboutirait à une description si fine de la tâche principale que l'on saurait « comment » la réaliser. On ne peut cependant pas faire l'hypothèse qu'il est toujours possible de décrire le « quoi » faire des sous-tâches.

Malgré la difficulté que représente une automatisation de la conception, on enregistre certaines tentatives dans cette voie.

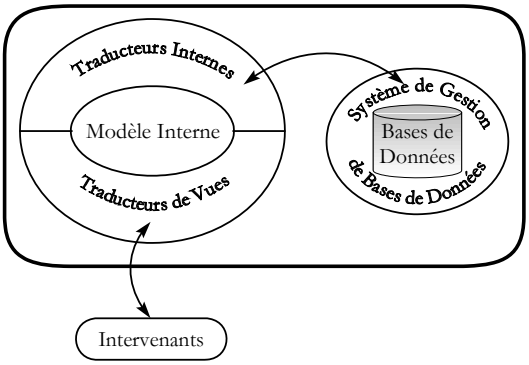


Figure 1-6. *Outils informatiques proposés par [Gardan et al. 97]*

[Gardan et al. 97] propose une vision abstraite de l'ensemble des outils nécessaires à la réalisation d'un système de Conception Fonctionnelle Assistée par Ordinateur (cf. Figure 1-6). Les auteurs distinguent :

- le modèle interne, contenant la description complète du produit, dont son historique de conception. Le modèle FBS en est une réalisation possible ;
- les traducteurs internes, dont la fonction est de faire progresser le modèle interne de manière automatique. Un traducteur interne est composé d'un analyseur qui étudie la possibilité de faire progresser le modèle interne et d'un solveur qui réalise la progression si l'analyseur a conclu positivement. Un superviseur maintient la cohérence globale du modèle interne. On peut considérer que les simulateurs vus précédemment sont des analyseurs, qu'un moteur d'inférence est un solveur et que la méthode de conception est un superviseur ;
- les bibliothèques, qui recensent les modèles internes des produits conçus au préalable et les méthodes de conception, qu'elles soient pré-enregistrées ou issues d'analyses statistiques des précédentes conceptions ;
- les traducteurs de vues, traduisant les informations fournies dans un langage métier en informations compréhensibles par le modèle interne et vice versa. On favorise ainsi une conception simultanée.

Dans le domaine de la conception routinière assistée par ordinateur, [Veerkamp et al. 93] exprime un produit sous une forme évaluée et sous une forme déclarative. Cette dernière consiste en un ensemble de faits, qui sont les mots clés du cahier des charges (par exemple : *voiture* et *rapide*), et en un ensemble de règles qui expriment, sous forme d'inférences, la manière de décomposer un ou plusieurs mots clés en un groupe d'autres mots clés, correspondant à des composants de plus bas niveau (par exemple : *voiture* $\Rightarrow$ *carrosserie* et *roue* et *moteur*). Partant des mots clés initiaux, on arrive ainsi à déduire automatiquement la structure complète du produit ; dans une certaine mesure, on a donc pu automatiser la conception en raisonnant à un niveau syntaxique plutôt que sémantique. Ceci est possible parce que l'on se limite à des

familles de produits dont la structure mécanique est parfaitement maîtrisée et vraisemblablement aussi parce que l'on fait partiellement abstraction de la géométrie.

[Takeda 94, Takeda et al. 94], faisant partie de l'équipe qui travaille sur le modèle FBS, étendent cette approche en se libérant de tout secteur d'activité. Ils utilisent deux niveaux de règles : des règles qui traduisent la manière dont ont été satisfaites les spécifications des produits conçus antérieurement (du type « pour réaliser telle fonction, utiliser tel ou tel comportement ») et des méta-règles précisant dans quel ordre et à quels moments utiliser la déduction (chaînage avant), l'abduction (chaînage arrière) et la restriction[10]. Nous illustrons ceci sur l'exemple suivant (cf. Figure 1-7) : les précédentes conceptions ont établi que le comportement C_0 réalise les fonctions F_0 et F_1 (règle 1), que C_1 réalise F_0 et F_2 (règle 2), que C_2 réalise F_2 et F_3 (règle 3), que F_1 et F_3 (règle 4) sont incompatibles. On souhaite satisfaire F_0, F_2 et F_3.

Commençons par essayer de satisfaire F_0. Une première abduction sur la règle 1 permet d'obtenir C_0. Rien ne garantit que C_0 soit un choix légitime mais on tente notre chance (si A implique B et si B alors il y a une certaine probabilité d'avoir A). Une déduction amène F_1 (règle 1) et une autre amène une contradiction avec F_3 (règle 4). La première abduction est annulée. On en tente une autre avec la règle 2 : on obtient cette fois C_1 (règle 2) puis F_2 (déduction) puis, avec une troisième abduction, F_3. La restriction a été introduite avant Takeda [Lifschitz 85, McCarthy 80] et est supposée gérer les limites d'applications des règles, lorsque ces dernières n'ont pas pu être décrites au préalable. En effet, d'après Takeda, il arrive particulièrement dans le domaine de la conception, que l'on ne puisse établir ces limites à cause de leur nombre, de leur imprévisibilité, du manque de temps, etc. Lorsque survient une inconsistance dans le raisonnement, l'auteur fait l'hypothèse que la cause est ce manque de spécifications et suppose, mais sans donner de précision, que la restriction peut formaliser la cause de l'inconsistance et ainsi enrichir la base de connaissances d'une exception à une règle.

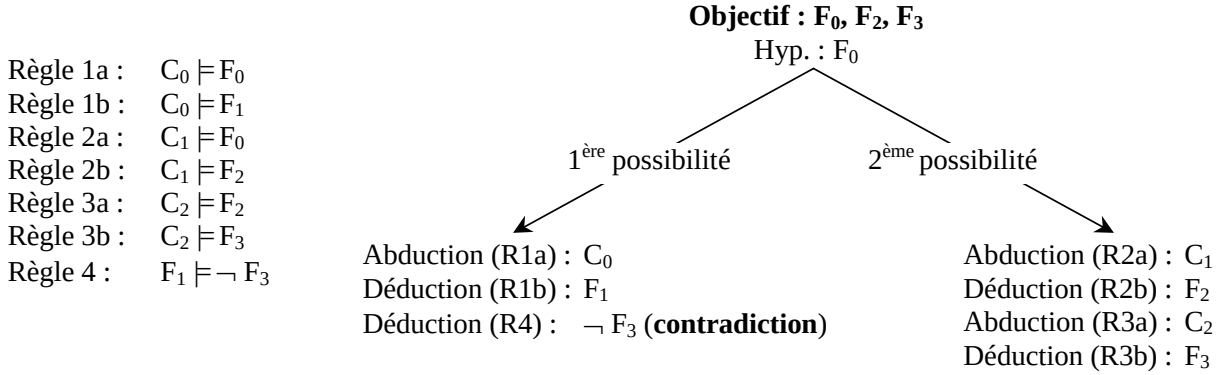


Figure 1-7. *Application de déductions et d'abductions*

Le raisonnement qui a mené à la satisfaction de toutes les fonctions du cahier des charges constitue un historique de conception et les méta-règles constituent la méthode de conception.

Cette technique suscite les commentaires suivants : la manière de traduire l'expérience acquise au cours des précédentes conceptions n'est absolument pas abordée. Cela constitue pourtant une difficulté majeure. Il est probable que la rédaction d'une nouvelle règle implique une part de programmation ne serait ce que pour obtenir les données qui interviennent dans les prémisses de la règle. Par exemple, pour intégrer dans la base de connaissances la règle « la distance entre un câble et une source de chaleur doit être supérieure à

[10] Circumscription en anglais

une quantité fonction de l'intensité de la source », il faut décrire la manière de calculer la distance du câble à toutes les sources de chaleur, donc un algorithme.

Par ailleurs, très peu d'indications sont données sur la manière de décrire les méta-règles qui déclenchent les déductions, abductions et restrictions, ni sur les critères de choix des règles sur lesquelles elles s'appliquent. En contrepartie, si l'on suppose ces difficultés résolues, la conception est menée de manière automatique.

On peut noter que les propositions relèvent essentiellement de la conception routinière, donc de domaines maîtrisés. Ceci ne diminue aucunement leur intérêt car les conceptions sont en grande majorité routinières et toute avancée dans ce domaine aura de très considérables retombées économiques.

On peut également remarquer que, dans ces deux références comme dans [Shah et al. 96], on suggère que des méthodes de conception pourraient être déduites de manière automatique par analyse des conceptions antérieures. Ceci impliquerait d'abstraire l'ensemble des informations contenues dans les différentes conceptions de tous les concepteurs, ce qui rejoint le problème de l'abstraction de l'information dans le cadre de la conception par analogie.

3 Intérêts d'une modélisation des fonctions d'un produit

À ce jour, la géométrie reste indispensable, en particulier pour assurer la cohérence des visions qu'ont les différents concepteurs d'un même produit [Tichkiewitch et al. 95, Tichkiewitch et al. 97] mais, à terme, son rôle devrait aller en décroissant. [Brun 97] constate que les éditeurs de logiciels de CFAO donnent une part trop importante au modèle géométrique en considérant ce dernier comme le modèle central et en lui ajoutant des « couches » contenant des informations propres à divers domaines d'activités. Selon l'auteur, c'est aller à « contre-courant d'une tendance de la recherche qui tend à évincer la géométrie de sa position centrale ». Toutefois, le modèle géométrique reste nécessaire pour vérifier la cohérence globale du produit à concevoir. Pour résoudre la contradiction fonction–géométrie, l'auteur propose un modèle où les fonctions évoluent conjointement avec la géométrie. Nous présentons dans ce qui suit des arguments à l'appui du modèle fonctionnel.

Les premières étapes de conception sont les plus déterminantes pour le coût du produit, sa longévité et son adaptation aux exigences fonctionnelles [Ullman 97]. Or, les informations manipulées à ce stade sont essentiellement fonctionnelles. En conséquence, la capacité à traiter des fonctions de manière informatique ouvre la voie à une automatisation des phases amont de la conception.

Une facette de cette automatisation sera la possibilité de déduire, des fonctions du produit, la forme de ses composants ainsi que leur position et les relations qui les lient.

Inversement, si le cheminement qui a mené le concepteur de la fonction à la forme est correctement conservé, on peut retrouver la fonction d'un composant à partir de sa forme. Ceci prend toute sa signification lors d'une reconception, étant donné que cela dispense le nouveau concepteur d'interpréter la structure du produit pour en extraire la fonction. Ceci ne résout cependant pas le problème complexe plus général de déduire la fonction d'un produit de sa seule forme.

Lors de conceptions routinières, on travaille peu à un niveau fonctionnel puisque l'on agit essentiellement sur la structure du produit pour le faire légèrement évoluer. Une modélisation des fonctions est donc moins primordiale. En revanche, une conception innovante implique un raisonnement

fonctionnel. Par conséquent, la capacité de traiter les fonctions de manière informatique autorise l'intervention de l'ordinateur dans les conceptions innovantes [Goel 97].

Les fonctions d'un produit peuvent se révéler utiles dans d'autres contextes : par exemple, [Eustache 99] prévoit d'utiliser des fonctions particulières (l'appartenance, l'inclusion, la séparation) pour l'amélioration des algorithmes de visualisation.

Un corollaire à la faculté de modéliser les fonctions est la possibilité de modéliser une hiérarchie fonctionnelle, dont les applications sont exposées en 5.1.

4 Les diverses classifications des fonctions

Nous avons souvent employé le terme « fonction » dans ce qui précède. La seule définition que nous en ayons donné est : « ce pour quoi le produit est prévu ». Cependant, si l'on vise un traitement au moins partiellement automatisé des fonctions, il importe de les caractériser de manière plus précise. Nous revenons donc sur la signification du terme et sur les diverses classifications proposées dans la littérature. Ces dernières ont leur importance car elles mettent en évidence des aspects communs à des sous-ensembles de fonctions. En cela, elles sont un premier pas vers un type de donnée abstrait « Fonction », donc vers une modélisation informatique.

4.1 Définition d'une fonction

Comme le précise [Umeda et al. 97], les fonctions, considérées comme un concept dérivant des intentions du concepteur, n'ont pas de définition claire, précise, et surtout objective. Toutefois, il est facile d'admettre que les fonctions constituent le concept fondamental dans le choix des caractéristiques du produit à concevoir. Le dictionnaire définit une fonction comme « l'utilité d'un élément dans un ensemble ». En appliquant cette définition dans le cadre de la conception fonctionnelle, nous pouvons associer « l'ensemble » au produit à concevoir, « l'élément » à une pièce ou une partie d'une pièce du produit et enfin, « l'utilité » à l'intérêt que peut représenter ce constituant dans l'ensemble du produit.

4.2 Propriétés des fonctions

Dans ce qui suit, nous recensons les différentes propriétés des fonctions d'un produit. Cet inventaire servira à une meilleure modélisation et offrira au concepteur une plus grande exactitude dans la spécification de ses intentions. Nous identifions dans la littérature deux propriétés principales des fonctions et en proposons une troisième (bénéfique ou néfaste) [Gardan et al. 98].

4.2.1 Fonctions objectives et fonctions subjectives

Lors des spécifications fonctionnelles, il apparaît que des fonctions se distinguent par leur aspect objectif ou subjectif [Gardan et al. 97, Shimomura et al. 96, Takeda et al. 96, Afav 89]. La fonction « propulsion » d'un vélo, que l'on peut réaliser avec un système de pédalier, est un exemple de fonction objective. Le fait de plaire aux personnes de 15 à 20 ans est un critère subjectif. Selon [Hearst 97], la beauté est la qualité attribuée à quelque chose qui fait plaisir ou qui satisfait dans sa forme. Le problème est que le degré de satisfaction peut changer radicalement avec le temps, l'endroit ou la communauté de

référence, en d'autres termes la population destinée à recevoir et utiliser le produit. De plus, il ne faudrait pas associer le terme « forme » de la définition de l'auteur à la géométrie, mais plutôt à une apparence globale comprenant par exemple l'aspect des matériaux utilisés, l'agencement spatial des différentes parties du produit.

Le terme objectif est utilisé dans le sens où aucun élément personnel n'intervient dans un jugement et dans la mesure où le caractère scientifique ne peut pas être contesté. Cela suggère que les fonctions objectives sont plus facilement représentables, parce qu'elles peuvent être ramenées à (et donc être représentées par) des lois physiques et des domaines de paramétrage. C'est pour cela que [Umeda et al. 90] s'intéresse dans un premier temps à la modélisation des fonctions objectives même si, d'un point de vue conceptuel, les auteurs identifient les deux types de fonctions.

[Shimomura et al. 96] propose une solution pour gérer aussi bien les fonctions objectives que subjectives. La technique utilisée « objective » les fonctions subjectives en associant les paramètres des fonctions objectives à un caractère subjectif. Par exemple, la beauté d'une feuille photocopiée est exprimée comme une combinaison linéaire des trois paramètres suivants : l'épaisseur des traits, la netteté et la brillance du support. Le poids de chaque paramètre est déduit par une étude statistique des opinions exprimées par des échantillons de la population. Toutefois, avant de consulter la population sur l'importance de certains paramètres physiques dans les fonctions qualitatives, il faut connaître ces paramètres, ce qui implique une conception préalable. La prise en compte des fonctions qualitatives ne peut donc se faire qu'après la réalisation des fonctions objectives.

On peut imaginer certaines heuristiques pour une détermination automatique des paramètres physiques intervenant dans la satisfaction d'une fonction subjective. Par exemple, pour réaliser la beauté d'une voiture, seuls les paramètres des organes visibles par l'observateur interviennent ; si ce dernier se situe à l'extérieur de la voiture, il est inutile de prendre en compte les paramètres du moteur. Dans ce cas précis, la fonction objective « visibilité » joue le rôle d'heuristique.

4.2.2 Fonctions intentionnelles et fonctions déduites

Dans le paragraphe 2.3.2, nous avons montré comment le raisonnement qualitatif pouvait déduire des comportements que le concepteur pouvait ne pas avoir envisagés. L'échauffement d'un récipient pouvait par exemple soit provoquer l'évaporation du contenu, soit l'extinction de la flamme. Ces comportements correspondent l'un à une fonction intentionnelle (chauffer le liquide), les deux autres à des fonctions déduites (faire s'évaporer un liquide, éteindre une flamme). On peut donc distinguer les fonctions intentionnelles, correspondant à une intention de conception, des fonctions déduites. Grâce à cette distinction, dans les conceptions futures, on saura déterminer un comportement réalisant la fonction déduite, ce sera le comportement de la fonction intentionnelle.

4.2.3 Fonctions bénéfiques et fonctions néfastes

Il faut donner la possibilité aux concepteurs de préciser si les fonctions déduites du raisonnement qualitatif, par conséquent non intentionnelles, sont bénéfiques au produit en cours de conception, ou si elles sont néfastes. Il faut préciser que ce genre de point de vue est dépendant d'un domaine d'activité. Par exemple, la simulation du comportement d'un moteur à essence fait apparaître l'émission de gaz toxiques. Par conséquent, le système informatique pourrait introduire une nouvelle fonction « émission de gaz toxiques » considérée comme néfaste à l'environnement. D'un point de vue militaire, l'émission de gaz

toxiques (fonction) pourrait être considérée souhaitable dans le cas d'une arme chimique (produit à concevoir).

Toutefois, les fonctions néfastes d'un produit ne doivent pas forcément être supprimées de la décomposition fonctionnelle parce qu'elles peuvent servir à une évaluation du produit en cours de description. En effet, si le concepteur ou le système hésite entre plusieurs solutions pour une partie du produit, le fait de compter le nombre de fonctions néfastes induites par cette partie l'aide à faire un choix.

4.3 Typologie des fonctions

Une propriété répartit les fonctions en deux groupes, les fonctions qui ont la propriété et celles qui ne l'ont pas ; une typologie propose une partition avec davantage de catégories. Elle offre au concepteur un vocabulaire plus précis pour l'expression des spécifications du produit. Dans un premier temps, nous étudions les typologies qui ne sont pas liées à un secteur d'activité, et tentons de discerner leurs points communs. Par la suite, nous nous intéressons plus particulièrement au domaine de la mécanique, un des domaines d'application privilégiés de la CFAO.

4.3.1 Typologies générales

[Keuneke 91] distingue quatre types de fonctions :

- de réalisation (To Make). Ce type permet de préciser que la fonction d'un composant est de réaliser un état spécifique de l'objet ou de réaliser une valeur ou un état d'un paramètre que l'objet manipule ;
- de maintien (To Maintain). Ces fonctions permettent d'entretenir dans le temps l'état d'un composant ou d'un paramètre. En effet, il existe une différence, par exemple, entre décrire une fonction qui spécifie un appareil apportant un liquide à une température de 40°C (Réaliser(température du liquide à 40°C)) et une fonction indiquant qu'un produit maintient une température de 40°C du liquide (Maintenir (température du liquide à 40°C)) ;
- d'interdiction (To Prevent). Celles-ci permettent d'exprimer un état indésirable sur le produit, contrairement aux fonctions de réalisation, auxquelles elles ressemblent, qui expriment un état désirable. On pourrait penser qu'interdire un état et maintenir son contraire sont deux choses équivalentes ; [Keuneke 91] fonde la nuance sur la durée : l'interdiction porte sur un état ponctuel, le maintien sur un état qui perdure. L'auteur illustre son propos avec l'éclairage d'un immeuble : Maintenir(Éclairage) équivaut à Maintenir(Non Obscurité) mais n'est pas équivalent à Interdire(Obscurité) ;
- et enfin de commande (To Control). Ces fonctions régulent les changements d'états.

Les nuances entre ces quatre types de fonctions sont subtiles et, malheureusement, l'auteur donne peu d'exemples pour comprendre leurs différences. Cette classification est reprise, selon [Umeda et al. 97], dans les travaux de [MacDowell et al. 96], dérivant de ceux de [Iwasaki et al. 93] mais on y adjoint une catégorie : les fonctions de permission (To Allow) qui s'opposent aux fonctions d'interdiction.

Selon [Josephson 97], les fonctions de réalisation décrites par [Keuneke 91] correspondent aux fonctions les plus connues et les mieux représentées : les fonctions de transformation (cf. 2.2.2). Elles sont décrites et utilisées dans l'ensemble de la littérature [Constant 96, Goel et al., Umeda et al. 97, Takeda et al. 96, Rosenman et al. 96] dans la mesure où elles sont associées à des états physiques impliquant la

modélisation du comportement, ce qui semble être le plus facilement représentable en informatique.

[Constant 96] affirme, en se basant sur l'analyse de la valeur représentée graphiquement par le modèle « Pieuvre » [Afnor 84], qu'il existe une seule fonction principale du produit à concevoir : selon lui, elle représente « la raison d'être du produit ». Cette fonction peut être de deux types : soit de participation à l'action, qui ressemble fortement à la fonction de permission ou de réalisation, soit d'opposition à l'action, qui équivaut à la fonction d'interdiction. Nous ne sommes pas vraiment en accord sur le point de l'unicité d'une fonction principale, dans la mesure où un produit ou un objet peut avoir plusieurs utilités. Un mur, par exemple, a pour fonctions de porter, de séparer et d'isoler et, sauf cas particuliers, aucune d'elles n'est moins importante que les autres.

En plus de la fonction principale, [Constant 96] utilise des fonctions auxiliaires et des fonctions contraintes. Les premières permettent le maintien, le contrôle ou la modification des caractéristiques de la fonction principale. Ainsi, elles regroupent les fonctions de commande, de maintien et dans une moindre mesure les fonctions de réalisation de [Keuneke 91]. Les fonctions de contraintes sont des relations s'établissant entre le produit lui-même et un ou plusieurs éléments de son environnement et représentent le choix de solutions technologiques. Il existe deux types de fonctions contraintes :

- une contrainte du produit sur l'environnement ;
- une contrainte de l'environnement sur le produit.

Par exemple, un radiateur à eau chaude (le produit à concevoir) possède comme élément de son environnement : l'eau chaude et l'air à réchauffer. Le but de la fonction principale est de participer à la transmission de chaleur entre l'eau contenue dans le radiateur et l'air extérieur, d'où la notion de *flux*. Selon le formalisme de l'auteur, l'eau fournit de la chaleur au produit (fonction A) via les surfaces fonctionnelles 1 (pour l'eau) et 2 (pour le radiateur). Le radiateur va alors transmettre ce flux (association interne B) à l'air extérieur (fonction C) via les surfaces fonctionnelles 3 (pour le radiateur) et 4 (pour l'air) (cf. Figure 1-8).

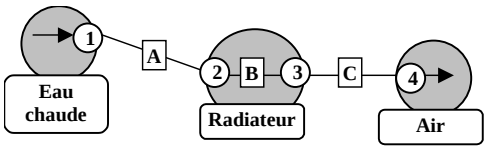


Figure 1-8. *Représentation d'une fonction d'un produit participant à la transmission d'un flux [Constant 96]*

Dans un cadre plus général, [Afav 89] identifie les fonctions de service, les fonctions contraintes, et les fonctions techniques.

Les fonctions de service « expriment l'action attendue du produit sur un élément du milieu extérieur (…) ». Il en existe de deux sortes : les fonctions d'usage qui correspondent à des fonctions objectives et les fonctions d'estime qui correspondent à des fonctions subjectives. Les fonctions contraintes sont définies comme une « limitation de la liberté du concepteur » et correspondent aux mêmes fonctions que les fonctions contraintes de [Constant 96]. Les fonctions techniques sont considérées comme des fonctions propres à chaque domaine d'activités. Il semblerait que les fonctions de service et les fonctions de contrainte servent à la définition du cahier des charges, et soient donc indépendantes du domaine d'activité. Ces fonctions sont alors traduites en fonctions propres à chaque métier, les fonctions techniques. Aucune information n'est donnée sur la manière de traduire ces différentes fonctions. Par

conséquent, la traduction des fonctions de service et contraintes reste à la charge des experts des différents domaines d'activités qui doivent interpréter les informations du cahier des charges en fonctions techniques.

On retrouve dans [Rosenman et al. 96] la nuance entre les fonctions du cahier des charges, détachées de tout domaine, et les fonctions techniques, propres à un métier. Les premières sont appelées « buts », les autres portent le nom de « fonctions ». Comme il est possible de trouver des réalisations (des fonctions) d'un but dans plusieurs métiers, les auteurs proposent de les lier par une liaison explicite du type « même que ». Ces relations permettront, par exemple, de dire que la hauteur d'une cloison, appartenant à la vue de l'architecte, est en relation avec les dimensions de la poutre, appartenant au modèle de l'ingénieur.

On facilite ainsi la gestion de la cohérence des fonctions associées à un même but. La difficulté réside dans la détection de ce genre de relation. C'est ce problème qui nous a conduit à proposer un modèle unique, que nous avons appelé modèle interne [Gardan et al. 97]. C'est ce que fait également [Veerkamp et al. 93] en ne considérant qu'un seul modèle de données (méta-modèle). Toutefois, les auteurs ne considèrent aussi qu'un seul langage de description (Artifact and Design Description Language), ce qui impose à tous les concepteurs de pratiquer un même langage alors que leurs habitudes et leurs formations les poussent certainement à la tendance inverse. En revanche [Rosenman et al. 96, Tichkiewitch et al. 95] défendent le point de vue d'une duplication des informations en autant d'exemplaires que de métiers ou de points de vue intervenant sur le produit.

Il est difficile de comparer plus avant la typologie des différents auteurs dans la mesure où la notion de fonction est elle-même très subjective. Toutefois, nous avons essayé de représenter les correspondances entre les diverses classifications dans la Figure 1-9.

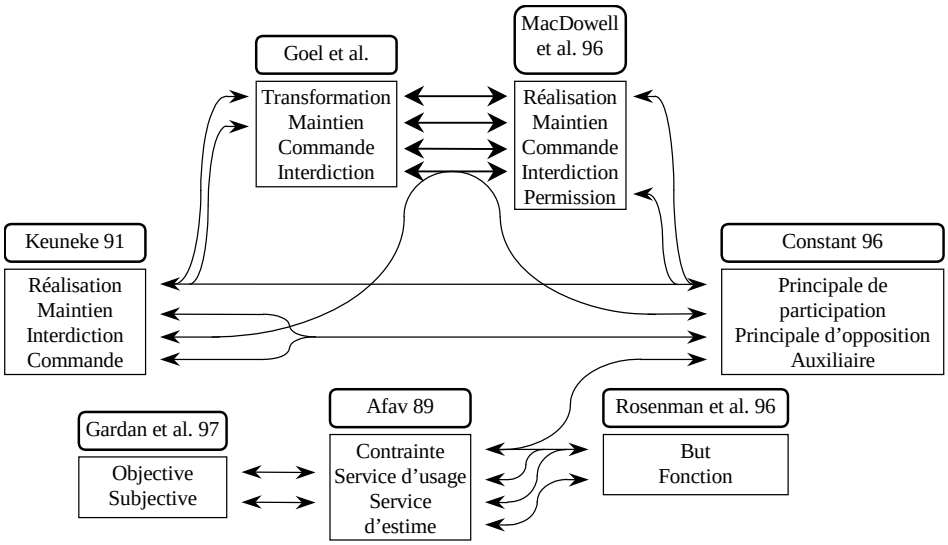


Figure 1-9. *Correspondances entre les types de fonctions des différents auteurs*

4.3.2 Typologie d'un point de vue mécanique

[Kirschman et al. 96] propose une typologie de fonctions restreinte à la mécanique (cf. Figure 1-10). Par exemple, la fonction puissance est utilisée dans un sens mécanique pour décrire les sources d'énergie ; la fonction de mouvement est possible de manière rotative, linéaire ou oscillatoire ; le mouvement peut être créé, converti, modifié, dissipé ou transmis.

Mouvement	– Rotation, linéaire, oscillatoire, autres – Créer, convertir, modifier, dissiper, transmettre – Flexible, rigide
Contrôle	– Puissance, mouvement, information – Continu, discret – Modification, indication – De l'utilisateur
Puissance / Matière	– Emmagasiner, prendre, expulser, modifier, transmettre, dissiper – Électrique, mécanique, autres
Assemblage	– Enveloppe, vue, protection – Permanent, temporaire – Supporte, attache, connecte, guide, limite

Figure 1-10. *Typologie de fonctions mécaniques [Kirschman et al. 96]*

Les auteurs proposent à l'aide de ces fonctions et de leurs propriétés des phrases types permettant de décrire une intention fonctionnelle mécanique. Ils recensent 150 combinaisons possibles. Toutefois, selon les auteurs, ces fonctions sont utiles à un niveau relativement concret de la décomposition fonctionnelle, donc plutôt vers les feuilles de la hiérarchie.

En contrepartie, cet aspect concret permet d'envisager une réelle automatisation : il suffirait, dans la phase d'apprentissage du logiciel, de collecter diverses réalisations de chaque fonction, leurs conditions d'utilisation et, dans la phase d'exploitation, de parcourir toutes les combinaisons correspondant à un mécanisme en les évaluant pour obtenir une meilleure solution. Le principe en lui-même n'est pas nouveau mais il est rarement exploitable du fait de l'énorme combinatoire qui lui est associée. Dans ce contexte par contre, il est probable qu'un fort élagage de l'arbre de recherche pourrait être réalisé si chaque composant était paramétré par tout l'arsenal des grandeurs mécaniques possibles (conditions d'utilisations, cadences, longévité, consommation, dimensions, intensité des efforts …). Des tables de ce type sont d'ailleurs proposées dans les livres de dessin industriel et de conception mécanique.

5 La hiérarchie fonctionnelle

En dépit de la multiplicité des types de fonctions, on trouve peu de structures de données dans la bibliographie. Les difficultés majeures soulevées par l'élaboration d'une structure de données sont l'absence d'une définition exploitable d'un point de vue informatique et le côté subjectif de certaines fonctions.

Dans ce qui suit, nous abordons la notion de hiérarchie fonctionnelle pour ensuite nous attarder sur la réalisation particulière qu'est le formalisme FBS.

5.1 Intérêts d'une modélisation de la hiérarchie fonctionnelle

Comme le précisent la plupart des auteurs, il est légitime de considérer l'ensemble des spécifications fonctionnelles du produit à concevoir comme une hiérarchie dans le sens où une fonction d'un très haut niveau d'abstraction est décomposée par un ensemble d'autres fonctions d'un niveau d'abstraction plus faible jusqu'à obtenir des spécifications très concrètes qui représentent les feuilles de la hiérarchie. Le terme de hiérarchie est plus adapté que celui d'arborescence étant donné que des branches de la

décomposition se rejoignent parfois, par exemple lorsqu'un organe participe à plusieurs fonctions. Cette hiérarchie est communément appelée *décomposition* ou *hiérarchie fonctionnelle*. Elle est riche de potentialités.

Elle retrace d'abord l'historique complet de la conception du produit. En cela, elle constitue une procédure acceptant, en entrée, les spécifications du produit et fournissant, en retour, le produit lui-même. À terme, on peut donc imaginer exploiter cette procédure pour générer des variantes du produit, en modifiant certaines spécifications si l'on souhaite une remise en cause fondamentale ou en changeant des paramètres apparus en cours de conception, pour des modifications mineures.

On peut constater que le cahier des charges, la hiérarchie fonctionnelle et le modèle produit (géométrie + caractéristiques de formes[11] + matériaux + tolérances …) sont des modèles de plus en plus détaillés du produit. On peut considérer que la hiérarchie fonctionnelle est un modèle implicite du produit dont le modèle produit est une version explicite ou évaluée. Or, en modélisation géométrique, une situation similaire se présente avec le schéma CSG, qui est une description implicite de l'objet, et la représentation par les bords (B-Rep) qui est une représentation évaluée. On sait effectuer des traitements sur des modèles CSG alors qu'initialement, on ne pouvait les exécuter que sur des modèles B-Rep (voir [Fischer et al. 91] pour un exemple dans le domaine de l'extraction de caractéristiques de forme). On sait également traduire de manière automatique une description CSG en une description par les bords [Gardan et al. 95a]. Par analogie, on peut espérer que l'évolution ira dans le même sens pour la hiérarchie fonctionnelle et le modèle produit. À terme, on espère donc deux choses : pouvoir appuyer des traitements sur la hiérarchie fonctionnelle alors qu'à l'heure actuelle ils se fondent sur le modèle produit, en particulier sur sa géométrie ; et traduire automatiquement la hiérarchie fonctionnelle en un modèle produit, ou tout au moins générer sa géométrie (donc sa forme).

Une dernière potentialité ambitieuse des hiérarchies fonctionnelles est la possibilité de déduire, de l'examen d'un grand nombre de hiérarchies fonctionnelles, des habitudes de conception, ce que nous appelons plus haut des méthodes de conception. Par exemple, si l'on constate que dans un contexte donné, le concepteur procède de la même manière neuf fois sur dix, on peut en déduire une fraction de la méthode de conception, la traduire en une règle et exploiter cette dernière pour faire des suggestions au concepteur [Gardan et al. 97, Shah et al. 96]. Cette pratique est typiquement une technique d'apprentissage. Cependant, aux difficultés intrinsèques de sa réalisation, s'ajoutent celles de la modélisation des données impliquées, les fonctions.

5.2 Le formalisme Function – Behaviour – State (FBS)

5.2.1 Présentation générale

Nous avons déjà brièvement décrit le modèle FBS en 2.2.4. Rappelons qu'il modélise un objet à trois niveaux : un niveau fonctionnel, un niveau comportemental et un niveau d'états. Au premier niveau, on décrit une fonction sous la forme d'un verbe, de plusieurs adverbes, de relations avec d'autres fonctions et du comportement qui la réalise (ou des comportements si l'on veut modéliser des alternatives, des niveaux de détails ou des redondances pour plus de fiabilité) [Tomiyama et al. 93, Umeda et al. 90, Umeda et al. 97] :

[11] features en anglais (cf. Chapitre 2, §2.2.1)

- le verbe correspond à l'action ;
- les adverbes sont des modificateurs de fonctions ; ils en complètent l'énoncé. Par exemple, pour l'action « déplacer », les modificateurs pourraient être « rapidement », « silencieusement » et « économiquement ». Sur ces deux premiers points, on peut noter la ressemblance avec la vision des fonctions dans le schéma Pieuvre, les formalismes FAST et SADT (cf. 2.2) ;
- les relations sont de quatre natures [Shimomura et al. 96, Shimomura et al. 95] :
  - la relation « se décompose en » indique que la fonction se décompose en d'autres fonctions ;
  - la relation « conditionnée par » permet d'exprimer une relation causale entre deux fonctions. Par exemple, si on considère qu'une fonction f_A est associée à un comportement c_A et que ce comportement est insuffisant pour réaliser f_A, alors on peut souhaiter ajouter un comportement c_B à la décomposition, et par conséquent la fonction f_B qui lui est associée. On dit que : « f_A est conditionnée par f_B » ;
  - la relation « renforcée par » spécifie qu'une fonction f_B est nécessaire pour satisfaire un modificateur m_A d'une fonction f_A. f_B est donc un terme de la décomposition fonctionnelle de m_A ;
  - la relation « décrit par » décompose un modificateur en d'autres modificateurs plus concrets, et donc en fonctions plus concrètes que la précédente.

Trois des quatre relations expriment donc une décomposition. Les auteurs ne justifient pas qu'elles soient distinguées. Ceci permet vraisemblablement d'exprimer d'avantage de nuances.

Le deuxième niveau de FBS contient les comportements associés aux fonctions. Un comportement est une succession d'états ; les transitions d'états sont justifiées par des lois physiques, explicitement modélisées sous forme de formules et de conditions de déclenchement [Kief, Ishii et al. 93, Tomiyama et al. 96]. Ces informations sont fondamentales pour une analyse qualitative du comportement [Kiriyama et al. 91].

Le niveau états décrit les entités qui composent le produit, les relations qui les lient et leurs paramètres. Par exemple, une structure peut être composée de deux engrenages paramétrés par leurs diamètres, positions, nombre de dents, liés par des relations géométriques comme le parallélisme des axes et par une relation d'entraînement. La géométrie des entités du produit n'est pas modélisée dans ce niveau. Une extension de FBS à un méta–modèle intégrant les aspects fonctionnels, géométriques et, de manière générale, tous les modèles applicatifs nécessaires est en cours d'étude dans le projet KIEF[12] [Kief, Yoshioka et al. 98, Yoshioka et al. 97].

5.2.2 Les apports effectifs et potentiels de FBS dans la conception fonctionnelle

L'objectif du formalisme FBS est d'apporter une assistance lors de la conception conceptuelle. Nous faisons donc un point des diverses exploitations qui en sont faites dans la bibliographie. Nous pensons que ces outils feront partie de la prochaine génération de logiciels industriels.

La première assistance envisageable, liée à l'existence d'une bibliothèque des produits conçus

[12] Knowledge Intensive Engineering Framework

préalablement, consiste à proposer au concepteur les solutions apportées par le passé à un problème donné lorsque celui-ci se pose à nouveau. C'est ce que propose [Umeda et al. 96] pour faciliter la réalisation d'une fonction par un comportement. Il ne s'agit donc pas directement d'une assistance au développement de la hiérarchie fonctionnelle (premier niveau de FBS) mais plutôt à celui des couches comportements et structures (deuxième et troisième niveau de FBS). La recherche d'une solution adaptée dans la bibliothèque de produits se fait, semble-t-il, sur une base syntaxique avec les inconvénients que cela suppose, par exemple au niveau des synonymies.

Une recherche du même type est menée dans un contexte proche : il arrive que le concepteur oublie certains organes nécessaires dans une structure ou certaines étapes dans un comportement. Comme les conditions de déclenchement des lois physiques régissant le comportement sont stockées, il est possible de constater de manière automatique que certaines d'entre elles ne sont pas satisfaites et de prévenir le concepteur d'un oubli [Umeda et al. 96]. De plus, le système parcourt la bibliothèque de produits pour proposer les comportements dans lesquels interviennent un organe dont les effets correspondent aux prémisses non réalisées. À supposer que le concepteur accepte l'un des comportements proposés par le système, la fonction associée au comportement est automatiquement ajoutée à la hiérarchie fonctionnelle, si bien que le système intervient cette fois au niveau fonctionnel de FBS.

L'analyse qualitative est un autre moyen pour le système d'agir au niveau fonctionnel. Elle permet d'étudier un comportement donné c_A, de constater qu'il a pour effet de créer les conditions de déclenchement d'un autre comportement c_B et donc, que la fonction f_B associée à cet autre comportement est réalisée. Une relation du type « conditionnée par » est donc introduite dans la hiérarchie. On peut noter que les deux actions menées par FBS an niveau fonctionnel permettent de compléter la hiérarchie mais de manière purement locale. On n'est pas encore à un stade où la hiérarchie fonctionnelle progresse automatiquement.

D'après [Umeda et al. 96, Tomiyama et al. 93] l'analyse des composants sert également à introduire une redondance fonctionnelle dans le produit pour en augmenter la fiabilité. On procède ainsi : le concepteur choisit la fonction à fiabiliser. Le système parcourt alors les comportements des autres fonctions pour trouver des entités de substitution aux organes intervenant dans le comportement de la fonction à fiabiliser. En fin d'analyse, on obtient un bilan du coût de la redondance fonctionnelle : on sait quels organes peuvent être relayés en cas de panne, lesquels ne le peuvent pas et impliquent l'ajout d'un composant.

[Shimomura et al. 96] fait remarquer que l'analyse du comportement ne permet qu'une évaluation binaire : la fonction est réalisée ou ne l'est pas. Lorsque plusieurs composants candidats réalisent une fonction, il faut donc pouvoir les évaluer plus finement, en estimant avec quel niveau de qualité ils réalisent la fonction. Selon les auteurs, cette estimation revient à calculer la manière dont sont réalisés les modificateurs de la fonction dans chaque comportement, à affecter un poids à chaque modificateur et, par combinaison, à en déduire la qualité de chaque comportement. Le choix des poids des modificateurs est toutefois subjectif et il varie d'un concepteur à l'autre. Deux difficultés se présentent donc : évaluer la qualité de la réalisation des modificateurs et diminuer la part de subjectivité dans l'affectation de poids aux modificateurs.

La première semble résolue dans l'article en faisant l'hypothèse que les modificateurs sont parfaitement réalisés (qualité = 100%). L'hypothèse ne semble pas exagérément restrictive : dans le cas où un modificateur est lié à un paramètre de la structure (par exemple, « vite » est lié à une durée exprimée en

secondes), il est possible d'évaluer le modificateur (par exemple, si durée max. = 3 secondes alors pourcentage qualité = *Min*(100%, 300/*x* %)) ; dans le cas où le modificateur est décomposé en sous fonctions, on lui applique la méthode récursivement. Dans les deux cas, on peut donc traduire en un pourcentage la qualité de la réalisation du modificateur.

Pour le choix des poids des modificateurs, les auteurs proposent une méthode qui s'applique également à la prise en compte des fonctions subjectives, à condition de pouvoir préciser quels critères objectifs interviennent dans la satisfaction de la fonction subjective : il s'agit en effet à nouveau d'attribuer un poids à chaque critère objectif. La méthode est illustrée par le critère « beauté » d'une photocopie, que l'on sait être lié, par expérience, à la brillance du fond, l'épaisseur des traits et à la netteté. Elle consiste à consulter un échantillon de la population (des utilisateurs du produit dans ce cas précis), à leur faire attribuer un poids à chaque critère objectif en fonction de leurs goûts et à en faire la synthèse via une formule de la théorie de la distribution (théorème de Shannon).

Dans le cadre du projet international IMS, le modeleur FBS de l'université de Tokyo (SYSFUND) [Ranta et al. 96] a été interfacé avec le modeleur produit de l'université de Helsinki (MCOES). L'intention était d'établir la maquette d'un système couvrant toute la conception, de la partie conceptuelle (SYSFUND) jusqu'à la conception détaillée (MCOES). L'expérimentation menée sur la conception d'un contacteur a amené les constats suivants : le résultat de la conception fonctionnelle était proche d'une description grossière de l'assemblage. Cela suggère que dans un domaine bien maîtrisé, les entités fonctionnelles sont très proches des composants du produit. Plus généralement, les auteurs estiment qu'il y a une sérieuse possibilité de coder la connaissance de divers domaines communs à de nombreuses branches de l'ingénierie.

6 Conclusion

La recherche en conception fonctionnelle assistée par ordinateur a deux principales facettes : la modélisation des méthodes de conception et la modélisation des produits. Nous avons essayé de donner un aperçu des travaux menés dans ces deux domaines.

En ce qui concerne la méthode de conception, nous avons d'abord rappelé les principales méthodes utilisées par les concepteurs dans les premières phases de la conception, afin d'y déceler des possibilités d'automatisation. Le bilan est plutôt négatif, étant donné que les informations y sont traitées de manière sémantique, ce qui reste hors de portée des techniques automatiques actuelles. Nous retenons cependant que les fonctions y sont principalement décrites par un verbe et des compléments. Nous retenons également l'étude de [Takeda et al. 94] qui spécifie un système capable de mener une conception sans intervention du concepteur à la condition très forte que les produits puissent être décrits sous forme de règles et que la méthode de conception ait pu être modélisée sous forme de méta-règles. Ceci suppose un très volumineux travail de formalisation qui, à notre connaissance, n'a pu être réalisé que dans des secteurs particuliers. Nous notons encore que l'analogie fait l'objet d'une étude approfondie, fondée en particulier sur des techniques d'intelligence artificielle. Elle n'est pas utilisée seulement pour adapter d'anciennes solutions à la conception d'un nouveau produit mais également pour adapter des méthodes dédiées à certains produits à la conception d'autres produits. L'analogie se heurte cependant à une triple difficulté : la

représentation du produit, la détection des similitudes entre deux produits ou deux spécifications et l'adaptation au nouveau produit des solutions inspirées des anciennes conceptions.

Les efforts en modélisation du produit concernent soit la structure de données, soit les traitements qui s'y appuient. Pour qu'une structure soit complète, elle doit renfermer l'historique du produit, sa forme et son fonctionnement. Les classifications relevées dans la bibliographie ne débouchent généralement pas sur une telle structure, à l'exception du schéma FBS qui s'en approche réellement et, dans une moindre mesure, d'autres outils qui travaillent essentiellement à un niveau comportemental.

Il en résulte que les traitements possibles se situent essentiellement au niveau du comportement. On recense des outils de simulation d'ores et déjà opérationnels et d'autres qui sont encore du domaine de la recherche. Il s'agit d'analyser le comportement de manière qualitative, de le fiabiliser par introduction de redondances fonctionnelles, de l'évaluer d'une façon autre que binaire et d'en détecter les incohérences. La tentative pour une prise en compte de fonctions subjectives est également liée au comportement puisqu'elle consiste à exprimer le critère subjectif comme une combinaison de critères objectifs, ce qui implique une disponibilité préalable des éléments objectifs. Ce n'est pas forcément un inconvénient ; nous avons le sentiment que cela reflète l'attitude de l'être humain face aux aspects subjectifs, que ce soit de manière consciente ou inconsciente.

Signalons enfin que les chercheurs reconnaissent unanimement la nécessité d'une bibliothèque de produits avec leurs historiques de conception.

On peut s'attendre à ce que les progrès en conception fonctionnelle assistée par ordinateur prennent deux voies complémentaires : la première consiste à appuyer certains traitements sur la hiérarchie fonctionnelle du produit plutôt que, comme c'est le cas aujourd'hui, sur un modèle évalué. En effet, il serait plus intuitif pour le concepteur d'intervenir sur les fonctions du produit que sur les paramètres déduits de ces fonctions. Par exemple, si une pièce déjà conçue se révèle trop fragile, il serait plus facile d'accentuer le poids de la fonction « solidité » que d'identifier les paramètres du modèle produit qui influent sur la solidité (épaisseurs, matériaux, positionnements …) et de les modifier.

Néanmoins, pour certains traitements, un modèle évalué restera indispensable. La forme du produit est par exemple nécessaire à la conception de son moule. On peut donc prévoir que des efforts seront menés pour soulager la tâche du concepteur en assistant ou en automatisant la transformation des fonctions en des modèles évalués. Cela n'apparaît possible que si l'intelligence artificielle progresse de son côté dans la prise en compte de la sémantique des textes et de l'analogie.

Chapitre 2 Synthèse des méthodes de construction d'une forme

« Pour expliquer un brin de paille,
il faut démontrer tout l'univers »
Remy de Gourmont

1 Introduction

L'informatique dans le domaine de la CAO a d'abord été appliquée à la représentation d'un objet physique de manière à ce que l'on puisse réaliser des opérations (telles que visualisation, calculs …) sur cet objet. Naissait alors la notion de géométrie et de représentation bi ou tri-dimensionnelle. L'engouement pour une représentation par une machine venait du fait que les implications (tant industrielles qu'économiques) étaient innombrables. En effet, une grande majorité des secteurs d'activités utilisent – nécessitent– la vision d'un ou plusieurs objets (assemblages). Dans le cas de la conception de produits manufacturiers par exemple (faisant intervenir simultanément des métiers d'horizons bien différents), la géométrie du produit et son affichage (impliquant le besoin de développer des algorithmes de plus en plus performants) permettent aux concepteurs de valider et/ou de vérifier des concepts mis en œuvre lors des premières phases de conception. En ce sens, nous pouvons préciser que la géométrie était et reste encore un langage commun à un grand nombre de métiers intervenant dans la conception de produits [Tichkiewitch et al. 95].

L'historique fait que les systèmes de CFAO actuels ont abordé la notion de forme comme une donnée primordiale et donc centrale à la modélisation produit. L'évolution de la CFAO a conduit à différents modèles pour représenter cette géométrie (B–Rep / G–Cartes, CSG …), chacun possédant des atouts et des inconvénients. Toutefois, même si la modélisation géométrique est dorénavant bien contrôlée, elle devient de plus en plus insuffisante pour minimiser le coût (de temps et d'argent) de production du produit. En effet, lors de la conception d'un nouveau produit, un des objectifs de la conception assistée par ordinateur est de construire la géométrie du produit. Or, les logiciels actuels ne conservent que les informations relatives à la géométrie et négligent celles qui ont amené à considérer telle géométrie pour le produit plutôt qu'une autre. En les conservant, une reconception du même produit (modification des

besoins du client par une modification du cahier des charges) aurait été plus rapide dans la mesure où des traitements auraient pu être appliqués sur ces informations pour générer plus facilement la nouvelle géométrie du produit. Ceci explique pourquoi les systèmes actuels ajoutent des informations nécessaires aux traitements : nous les appelons informations de niveau sémantique plus élevé que la géométrie. Néanmoins, l'ajout d'informations alourdit soit les traitements concernant directement le modèle géométrique soit les manipulations de ces informations (ajout, suppression, mise en relation entre entités …). De ce fait, afin d'améliorer les temps de calcul et la validité des modèles, certains auteurs ne considèrent plus le modèle géométrique comme le noyau central du système de CFAO mais comme une vue du modèle plus fondamental qu'est le modèle produit [Brun 97].

Même si nous avons pu voir dans le précédent chapitre que la conception de produits n'adoptait pas des règles –des procédés, des méthodes …– pour la génération du produit et en particulier pour la génération de sa forme, il en ressort tout de même un cycle de travail très général :

- génération d'un espace de solutions correspondant à des solutions de conception a priori valides satisfaisant le cahier des charges du produit exprimé par le client ;
- sélection des solutions les plus prometteuses en estimant l'adéquation entre les solutions et le cahier des charges client ;
- modification des précédentes solutions afin d'améliorer leur adéquation avec le cahier des charges. Ceci peut être fait soit de manière automatique en utilisant des techniques d'intelligence artificielle (utilisation d'un système expert et d'une base de règles ou de méthodes stochastiques tels que les algorithmes génétiques) soit de manière manuelle en faisant intervenir le concepteur d'une quelconque façon.

Avant d'aborder plus particulièrement la génération, l'estimation et la modification de solutions géométriques, nous relatons dans le paragraphe suivant les moyens actuels dont nous disposons pour décrire et construire une forme.

2 Description de la forme

De manière générale, une forme se définit comme l'aspect extérieur d'un objet physique. En conception assistée par ordinateur, il est couramment admis de distinguer deux notions : la topologie et la géométrie. La géométrie s'intéresse à la définition d'éléments de l'espace euclidien tels que des points, des droites, des segments, des plans, des surfaces (cf. Figure 2-1b) … La topologie permet de représenter des relations entre ces différentes entités indépendamment de leur grandeur ou de leur position (cf. Figure 2-1a). Par exemple, elle permet de préciser qu'un sommet est incident à une arête ou que deux surfaces sont adjacentes. Pour distinguer le point topologique du point géométrique, le premier est appelé sommet ou cellule de dimension 0 et le deuxième simplement point. Par extension, une droite est appelée arête ou cellule de dimension 1, la surface est appelée face ou cellule de dimension 2 et l'objet un volume ou cellule de dimension 3. Comme la représentation topologique ne considère pas les dimensions et les positions, certaines cellules sont associées à une entité géométrique. Dans la plupart des cas, il est extrêmement complexe, voire impossible, de déterminer la topologie seulement à partir d'une géométrie d'un objet. Cela

peut s'expliquer par le fait que des problèmes d'imprécision peuvent contribuer à considérer une mauvaise représentation topologique par rapport à la géométrie considérée. Pour éviter ces problèmes, il est donc nécessaire de conserver simultanément la topologie et la géométrie de l'objet. Le modèle qui conserve les représentations topologique et géométrique est appelé par abus de langage le modèle géométrique de l'objet. Le fait de construire et/ou de décrire le modèle géométrique est appelé modélisation géométrique.

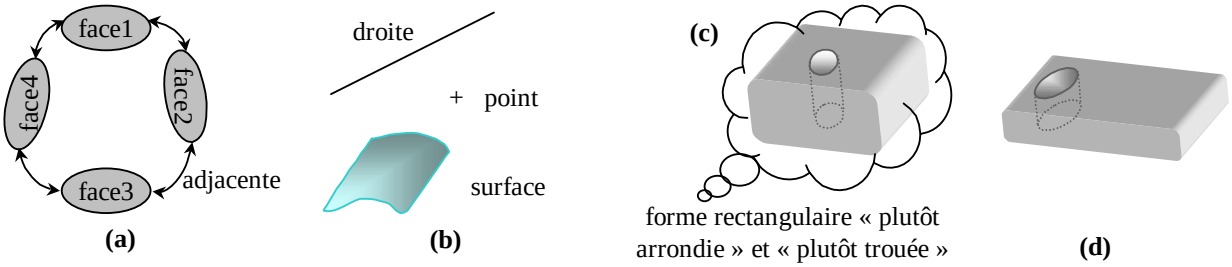


Figure 2-1. *Notions de topologie (a), géométrie (b), morphologie (c) et modèle géométrique (d)*

Le modèle géométrique, une fois ses informations fournies par le concepteur, ne permet de définir qu'un seul objet (cf. Figure 2-1d). Certes, certains modèles incluent des paramètres qui facilitent alors la définition d'une famille d'objets. Cependant, ces paramètres modifient principalement les dimensions de l'objet et ne permettent pas facilement de définir des objets à géométrie, voire à topologie, variable. Par exemple, un modèle paramétrique polyédrique permet de représenter une boîte avec des dimensions que l'on peut modifier aisément. Mais, il ne permet pas de représenter le « squelette » de l'objet sur lequel le concepteur applique toute une panoplie de géométries pour les entités topologiques du modèle. Il semble donc nécessaire de définir un autre modèle, que nous appelons modèle morphologique, permettant de définir un objet de manière imprécise, tout en gardant la possibilité d'appliquer des traitements informatiques. Notre objectif n'est pas de spécifier un modèle morphologique mais il nous paraît néanmoins naturel qu'un tel modèle contienne à la fois une topologie et une géométrie « floues », dans le but de modéliser plus de familles d'objets que ne le permettent les modèles géométriques actuels (cf. Figure 2-1c). Toutefois, cet aspect conduit à des développements théoriques qui sortent du cadre de notre travail.

Nous présentons au paragraphe 2.1 l'intégration de ces notions dans les modèles géométriques les plus communs (B–Rep / G–Cartes, CSG, modèles variationnel et paramétrique).

Modéliser une forme consiste à définir les informations manipulées dans l'un des trois niveaux d'abstraction présentés ci-dessus (topologie, morphologie, géométrie). Le choix du modèle de représentation dépend des traitements que l'on souhaite réaliser sur l'objet une fois construit. Il faut savoir que le modèle topologique et le modèle morphologique permettent de représenter beaucoup plus d'objets que le modèle géométrique.

Plus l'objet est complexe, plus la tâche de modélisation est fastidieuse. C'est pourquoi des opérations de construction aussi automatiques que possible sont nécessaires afin d'alléger considérablement la charge du concepteur. Nous présentons au paragraphe 2.2 des méthodes de construction d'objets géométriques à partir d'informations de niveau sémantique élevé (caractéristiques de forme, esquisses, fonctions …).

2.1 Les modèles géométriques

Dans ce paragraphe, nous présentons trois types de modèles géométriques qui sont les modèles par les

frontières, les modèles implicites et les modèles hybrides, représentant une combinaison des deux premiers. Ces derniers sont principalement utilisés par les logiciels de CFAO car ils fournissent plus de fonctionnalités au concepteur. Nous verrons en dernier lieu la validité des modèles.

2.1.1 La modélisation par les frontières

Les deux modèles présentés dans ce paragraphe consistent à modéliser la « peau » de l'objet, c'est-à-dire la surface séparant les points de l'espace qui appartiennent au solide de ceux qui n'appartiennent pas. C'est une modélisation par les frontières[13].

2.1.1.1 B–Rep

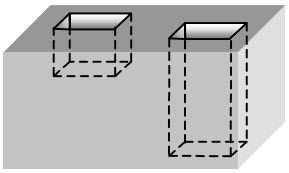
Les entités géométriques sont au nombre de trois : point, courbe, surface. Elles sont reliées entre elles à différents niveaux pour former progressivement le solide à modéliser. Les relations entre entités géométriques caractérisent les notions topologiques discutées précédemment et se représentent à l'aide d'un graphe dont les nœuds sont les entités topologiques et les arcs les relations d'adjacence, d'incidence ou de composition.

Plusieurs catégories de B–Rep existent, pouvant être distinguées par exemple suivant :

- la géométrie : certains B–Rep (dénommés polyédriques) n'utilisent que des surfaces planes alors que d'autres permettent de modéliser des surfaces gauches (Bézier, NURBS …). Ces derniers sont nommés modèles à faces gauches. Leur intérêt repose dans leur modélisation exacte de la géométrie. Les modèles polyédriques approchent les surfaces gauches par un nombre fini de surfaces planes : l'approximation polyédrique. L'utilisation d'un B–Rep polyédrique permet l'implémentation d'algorithmes qu'on ne sait pas définir dans le cas d'un B–Rep à faces gauches : par exemple, le résultat de l'intersection de deux surfaces gauches est approché par des surfaces planes ;

- la topologie : certains B–Rep font que les solides qu'ils modélisent respectent la règle d'Euler–Poincaré (cf. Figure 2-2 : S est le nombre de sommets, A le nombre d'arêtes, F le nombre de faces, CI le nombre de contours intérieurs aux faces –i.e. le nombre de trous dans les faces–, CC le nombre de composantes connexes et G le nombre de trous traversant l'objet de part en part –aussi connu sous le nom de *genre*). L'utilisation de modèles vérifiant la règle d'Euler-Poincaré permet de construire des objets topologiquement valides. Il faudra, par la suite, vérifier que le plongement de la topologie dans un espace réel définisse un objet géométriquement valide car il est possible de définir un objet topologiquement valide et incorrect d'un point de vue physique. Pour le concepteur, il est possible d'utiliser des primitives ou procédures lui permettant de construire le solide tout en respectant la règle d'Euler-Poincaré : ces procédures sont nommées opérateurs d'Euler. Par exemple, lors de l'ajout d'un sommet, il est nécessaire d'adjoindre une arête ou un contour intérieur pour maintenir la validité du solide. D'autres opérateurs peuvent être utilisés pour construire plus intuitivement le solide à concevoir : par exemple les opérations booléennes [Perrin 95] ont le mérite de pouvoir être basées sur les opérateurs d'Euler ; nous en détaillons le principe ultérieurement (cf. 2.1.2).

[13] Boundary Representation en anglais (B–Rep)



$$\begin{array}{cccccccc} S & - & A & +F & - & CI & = & 2\times(CC & - & G) \\ 24 & & 36 & 15 & & 3 & & 1 & & 1 \end{array}$$

Figure 2-2. *Vérification de la formule d'Euler–Poincaré [Foley et al. 95]*

En réalité, le choix d'un modèle en particulier dépend essentiellement des utilisations envisagées. Remarquons que les B–Reps dits « classiques » ne possèdent pas de relations d'adjacence entre solides ; c'est-à-dire qu'il n'existe pas de relations topologiques entre volumes. Cela peut être gênant par exemple dans le cas où le concepteur veut modéliser la structure interne d'un ski ou représenter des couches géologiques. Dans ces deux cas, chaque volume représentera une matière différente. Mais, il est important de spécifier que les couches sont collées les unes aux autres. Avec un B–Rep classique, c'est seulement la géométrie (les points, les lignes) qui permettront au concepteur de coller les différentes couches. L'enregistrement des informations topologiques au niveau volumique permettrait d'éviter des problèmes d'imprécision numérique lors de traitements géométriques.

2.1.1.2 G–Cartes

Le modèle des cartes généralisées étend les relations topologiques entre faces, arêtes et sommets à une dimension n quelconque. En l'occurrence, il permet entre autres de modéliser les relations d'adjacence entre volumes. Une carte généralisée[14] de dimension n [Lienhardt 89, Lienhardt 91] ou n–G–Carte est définie par une algèbre $G = \{B, \alpha_0, \ldots, \alpha_i, \ldots, \alpha_n\}$ où :

i. B est un ensemble fini de brins,

ii. $\forall\ 0 \le i \le n$, α_i est une involution sur B (i.e. $\forall\ b \in B$, $\alpha_i(\alpha_i(b)) = b$),

iii. $\forall\ 0 \le i \le n$, $\forall\ i+2 \le j \le n$, $\alpha_i\alpha_j$ est une involution.

Figure 2-3. *Modélisation d'une bande de Moëbius (a, b) ou d'une bouteille de Klein (c) à l'aide d'une 2–G–Carte*

L'unique élément de base utilisé est le *brin* (i). Il se représente schématiquement par un segment correspondant à une demi-arête. Le fait de diviser une arête en deux éléments identiques permet de modéliser des solides non orientables (ruban de Moëbius ou bouteille de Klein, Figure 2-3[15]). Pour former une arête, il suffit de « coudre » deux brins par la relation α_0, caractérisant simplement la relation d'adjacence entre deux sommets (Figure 2-4a[15]). De manière similaire, les relations α_1, α_2, α_3 sont utilisées

[14] Generalised Maps en anglais

[15] images fournies par Yves Bertrand de l'université de Poitiers

pour coudre respectivement des arêtes (Figure 2-4b[15]), des faces (Figure 2-4c[15]) et des volumes entre eux (Figure 2-4d[15]).

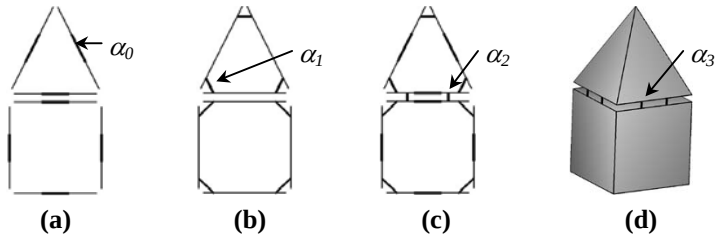


Figure 2-4. *Modélisation d'une maison à l'aide d'une 3–G–Carte utilisant les relations α_0 (a), α_1 (b), α_2 (c), α_3 (d)*

De manière générale, la contrainte (*ii*) exprime simplement le fait que les relations d'adjacence étendues aux brins sont des relations binaires symétriques (si un brin b_1 est cousu avec b_2 par α_0, alors b_2 est cousu avec b_1 par α_0).

D'autre part, $\alpha_0\alpha_2$ est une involution (*iii*) signifie simplement que les faces sont « cousues » le long de leur arêtes. Au niveau volumique, cela se traduit par $\alpha_0\alpha_3$ et $\alpha_1\alpha_3$ sont des involutions (les volumes sont cousus le long de leur faces).

Même si le modèle des cartes généralisées n'utilise qu'un unique élément (le brin), il faut pouvoir identifier, pour toutes les dimensions, les cellules du modèle topologique plus communément appelées sommets, arêtes, faces, volumes. Pour cela, il suffit de supprimer la relation α_i dans G pour obtenir l'ensemble des cellules de dimension *i* ; chaque composante connexe de G étant alors une cellule (par exemple, les sommets sont obtenus en supprimant α_0).

Comme il est intuitif de représenter une G–Carte par une structure de données de type graphe dont les nœuds sont les brins et les arcs les involutions entre brins, l'accès aux brins se fait simplement en utilisant un parcours en profondeur utilisant des marquages booléens (cf. Figure 2-5). En pratique, les involutions ne sont pas supprimées, mais plutôt ignorées lors du parcours.

```
Algorithme Parcours (brin b, ensemble d'involutions E_I)
    Marquer (b)
    Pour toute involution i de E_I
        Si non (marqué(i(b)) alors Parcours (i(b), E_I)
        Fin si
    Fin pour

Fin Algorithme
```

Figure 2-5. *Algorithme de parcours de cellules de dimension quelconque*

Cette modélisation par les frontières a été spécifiée algébriquement par [Bertrand et al. 98] et implantée par [Bertrand 98] pour former le modeleur Topofil[16] ; les opérations booléennes ont aussi été spécifiées pour ce modèle géométrique [Cazier 97]. Par exemple, il est possible de considérer la dimension 4 comme une relation temporelle simplifiant alors l'animation de solides modélisés [Bechmann 95]. Par ailleurs, les cartes généralisées ont permis en visualisation de simplifier et d'accélérer un algorithme de radiosité

[16] déjà utilisé par le Centre d'Étude Atomique pour modéliser les couches géologiques

[Pallez 96].

Les intérêts d'une modélisation par les cartes généralisées sont :

- une généralisation des relations d'adjacence et d'incidence à une dimension quelconque ; en particulier au niveau volumique ;
- l'unicité d'une structure informatique « brin ».

2.1.2 La modélisation implicite

Un modèle *implicite* précise comment obtenir le solide alors qu'un modèle *explicite* (B–Rep, G–Carte …) décrit comment est ce solide. Cela signifie que la visualisation d'un solide modélisé à l'aide d'un modèle implicite nécessite des pré–traitements pour déterminer ce qu'est l'objet.

L'un des modèles implicites les plus populaires est l'arbre de construction (Constructive Solid Geometry – CSG). Le principe consiste à combiner des volumes primitifs (tels que boîtes, sphères, cônes …) à l'aide d'opérations géométriques ensemblistes (union ∪, intersection ∩, différence –) et à l'aide de transformations géométriques (translation, rotation, changement d'échelle … suivant les trois dimensions de l'espace) (cf. Figure 2-6).

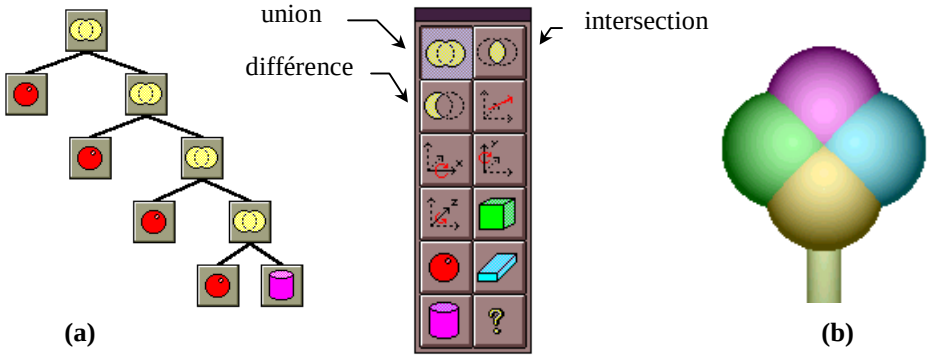


Figure 2-6. *Exemple d'arbre CSG (a) et de son résultat par lancé de rayon (b)*[17] *[Glassner 89]*

Les primitives volumiques sont paramétrées, ce qui permet de définir des contraintes sur ces paramètres. De cette façon, un paramètre d'une primitive peut être fonction d'un paramètre d'une autre primitive. Nous considérons qu'un arbre CSG est un exemple de modèle morphologique dans le sens où il est possible d'obtenir plusieurs pièces morphologiquement similaires à partir du même arbre, simplement en changeant les valeurs des paramètres de cet arbre.

La combinaison d'objets primitifs se fait principalement à l'aide d'opérations booléennes. Toutefois, l'application ordinaire de telles opérations sur des volumes n'aboutit pas forcément à un volume (cf. Figure 2-7). De ce fait, de nouveaux opérateurs booléens dénommés *régularisés* et notés ∪*, ∩*, –* [Requicha et al. 78] ont été introduits pour pallier ce problème. Ainsi, l'utilisation de ces opérateurs booléens régularisés assure que l'objet résultant est une forme comme nous l'avons définie auparavant.

[17] images obtenues à partir du logiciel créé à l'occasion d'un projet de maîtrise par B. Marchal, O. Mathieu et D. Pallez

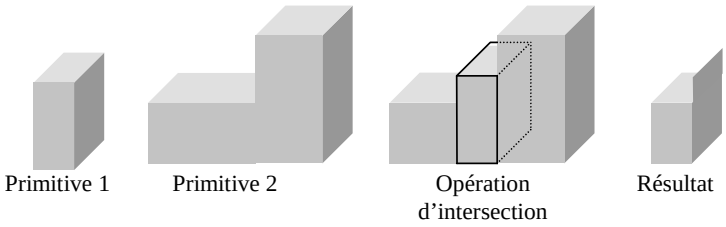


Figure 2-7. *Opération booléenne ne donnant pas d'objet résultat valide*

Dans le monde académique, les exemples de volumes construits suivant le formalisme du modèle CSG sont très souvent présentés comme des arbres binaires correctement équilibrés. Toutefois, il apparaît difficile que le concepteur ou l'ingénieur arrive à imaginer de tels arbres lors de la construction de pièces complexes. De ce fait, il s'avère en pratique que l'historique de construction d'un objet mécanique ou autre se résume en une liste d'actions (opérations diverses appliquées à la géométrie ou à la morphologie) (cf. Figure 2-8a).

En outre, il est possible d'enrichir la panoplie de primitives ou d'opérations (notion d'extrusion, de congé, d'esquisse couramment utilisé dans les systèmes de CFAO actuels) à condition que les critères de validité du modèle soient vérifiés.

Pour plus de précisions sur les modèles implicites ou explicites, le lecteur pourra se référer à [Foley et al. 95, Gardan 91, Rossignac et al. 99, Tollenaere 98 …].

2.1.3 Modeleurs sous-jacents aux systèmes de CFAO actuels

Les deux modèles de représentation volumique précédents possèdent chacun des avantages et des inconvénients. Le fait que les modèles de représentation par les frontières soient évalués constitue à la fois un avantage et un inconvénient. En effet, des traitements informatiques peuvent être effectués beaucoup plus rapidement sur un modèle évalué. Toutefois, l'évaluation ne permet pas de conserver la façon dont l'objet a été construit. Il se trouve que les modèles implicites comme les arbres CSG contiennent cet historique. De ce fait, les systèmes de CFAO, visant à fournir aux concepteurs de multiples fonctionnalités, ont combiné ces deux représentations en un modèle *hybride*. Ceci permet d'appliquer des traitements beaucoup plus rapidement lorsque cela est nécessaire. Mais le fait que l'historique de construction soit conservé permet, lors d'une modification, de reconstruire l'objet de manière quasi automatique. Cela évite au concepteur de détruire manuellement la forme pour la reconstruire suivant les modifications désirées. Néanmoins, l'historique utilisé dans ces systèmes ne correspond pas directement à un arbre CSG comme nous avons pu le définir dans le paragraphe 2.1.2. Il correspond plutôt à un formalisme de construction conservant une liste des objets primitifs utilisés et les différentes opérations effectuées (cf. Figure 2-8a).

Nous avons pu remarquer dans le chapitre 1 que la conception de produits consiste à faire des choix et à valider ou à réfuter ces choix : le produit se construit par tâtonnements. Comme la forme dépend des choix de conception, la forme se construit aussi par tâtonnements. Dans l'esprit d'offrir toujours plus de fonctionnalités, les systèmes de CFAO proposent une paramétrisation des primitives et des opérations lorsque cela est possible. Par exemple, dans la Figure 2-8, l'arrondi d'un des bords de la pièce peut être paramétré par un rayon de raccordement. Dans le cas où la pièce n'est pas fabricable, il suffit peut être au concepteur de modifier la valeur du rayon pour la voir se modifier automatiquement.

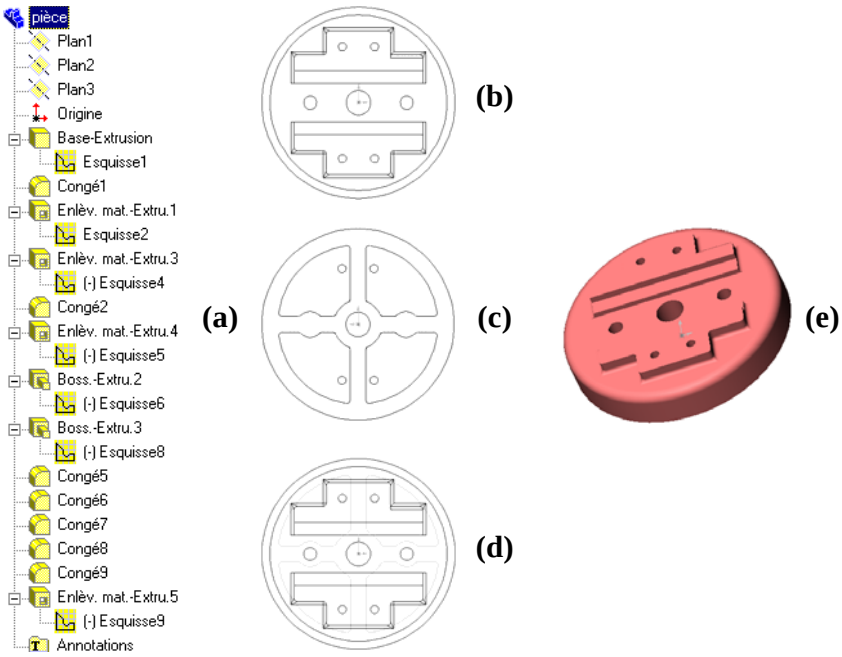


Figure 2-8. *Exemple de construction d'une pièce mécanique dans Solidworks© ; (a) historique, (b) vue avant, (c) vue arrière, (d) vue combinée, (e) vue en 3D*[18]

Par ailleurs, la notion de paramètres s'accompagne dans la majorité des cas de contraintes sur ces paramètres [Anderl et al. 96]. En effet, il est possible qu'une modification de l'arrondi de la pièce entraîne la modification d'un autre paramètre (par exemple, l'épaisseur de la pièce). Ceci est simplement traduit en contraintes, liant ces deux paramètres de manière mathématique. Ainsi, la modification d'une valeur entraîne une modification automatique de l'autre valeur. Ces contraintes sont souvent appelées contraintes de conception. Mais, il existe, en outre, des contraintes purement géométriques telles que le parallélisme ou la perpendicularité entre deux entités géométriques (une face doit être parallèle à une autre face, qui doit être elle-même perpendiculaire à un plan donné par exemple).

Dans la mesure où un nombre très important de contraintes peut être défini et appliqué sur l'objet augmentant indéniablement la complexité de la construction, il est nécessaire de fournir un outil informatique permettant de valuer l'ensemble des paramètres contraints. Ces outils sont appelés méthodes de résolution de problèmes contraints. Il en existe en réalité deux familles bien maîtrisées : les méthodes paramétrique et variationnelle [Leinen 97, Anderl et al. 95, Shapiro et al. 95, Chung et al. 89].

Les contraintes géométriques et de conception sont censées représenter les intentions du concepteur. Néanmoins, au cours des premières phases de conception, ces intentions s'expriment principalement en terme de fonctions ; ceci oblige alors le concepteur à traduire lui-même des informations sémantiques de niveau très élevé en informations de très faible niveau telles que le parallélisme ou la perpendicularité d'entités géométriques. Les contraintes d'ingénieries correspondent principalement à des paramètres des entités géométriques se rapportant souvent à des dimensions ou à des angles.

Enfin, même si les logiciels de CFAO (Catia©, ProEngineer©, SolidWorks©, TopSolid©, Unigraphics© …) proposent beaucoup de fonctionnalités au concepteur, ces dernières restent tout de même de faible niveau dans la mesure où il doit imaginer, proposer et construire la forme qui le satisfait.

[18] images fournies par S. Leinen et H. Bonnefoy de l'IFTS–Charleville.

Nous abordons donc dans le prochain paragraphe les méthodes actuelles et futures d'assistance à la construction d'objets géométriques pouvant s'appliquer sur les différents modèles géométriques.

2.1.4 Validité des modèles géométriques

Quel que soit le modèle géométrique utilisé, il doit vérifier les trois critères suivants [Perrin 95] :

- rigidité : l'objet modélisé doit avoir une forme invariante, quelles que soient sa position et son orientation ;
- finitude : le volume de l'objet modélisé est fini ;
- homogénéité : un point de l'espace ne peut appartenir qu'à un seul objet modélisé.

Les deux premières propriétés sont automatiquement vérifiées lors de la création de l'objet à concevoir avec l'utilisation des opérateurs que nous avons présentés (opérateur de base, opérateurs d'Euler, involutions, opérations booléennes). Néanmoins, le dernier critère reste à la charge du concepteur car il peut très bien définir des objets s'interpénétrant. Il existe néanmoins des algorithmes permettant d'assister le concepteur dans cette tâche de vérification.

2.2 Les méthodes de construction à sémantique élevée

Dans les précédents paragraphes, nous avons brièvement expliqué plusieurs méthodes de construction de faible niveau sémantique : les opérateurs d'Euler, les involutions, les opérations booléennes et de manière générale les méthodes variationnelle et paramétrique. Nous les appelons opérateurs de faible niveau dans la mesure où le concepteur est obligé de manipuler des informations basiques telles que les arêtes, faces, sommets, brins, cubes, sphères … Mais, dans le but de libérer le concepteur de cette lourde tâche, la tendance actuelle est d'automatiser le plus possible la construction de la forme d'un objet. Toutefois, il ne faudrait pas pour autant omettre totalement la participation du concepteur dans cette génération dans la mesure où, à l'heure actuelle, il est impossible pour un système informatique de modéliser d'une part la créativité et d'autre part l'esprit très synthétique du concepteur.

Plusieurs méthodes de construction d'un objet existent et sont assimilées à des méthodes d'assistance. Elles sont présentées dans les paragraphes suivants, par complexité croissante du niveau sémantique des informations manipulées par le système informatique.

2.2.1 Méthode de construction par caractéristiques de forme

Les entités basiques présentées dans les paragraphes précédents et manipulées par le concepteur lui permettant de construire un solide n'avaient pour lui que peu de significations. Afin d'améliorer la sémantique des informations à manipuler, les caractéristiques[19] ont été ajoutées aux systèmes de CFAO et représentent une première avancée dans cette voie. Elles sont classifiées en plusieurs catégories [Poinsignon 97] :

- les caractéristiques de forme, participant à un aspect fonctionnel généralement local à un solide. Par exemple, on peut considérer qu'une rainure permet un guidage en translation. Cette caractéristique modifie localement la forme d'un objet ;

[19] features en anglais

- les caractéristiques d'assemblage, telles que vis, écrou permettent comme leur nom l'indique d'assembler plusieurs pièces entre elles. Ces caractéristiques ne sont pas uniquement matérialisées par des objets volumiques mais aussi par des informations telles que la chronologie de l'assemblage ou encore la trajectoire empruntée par une pièce …
- les caractéristiques cinématiques expliquent le comportement de l'objet et peuvent être représentées par des chaînes cinématiques, des vitesses, des couples …
- les caractéristiques de précision matérialisant les tolérances de fabrication par exemple ;
- les caractéristiques relatives aux matériaux du solide, permettant par exemple d'appliquer des algorithmes de rendu plus réaliste ;
- les caractéristiques administratives comme la référence de la pièce, la quantité en stock, le coût ;
- …

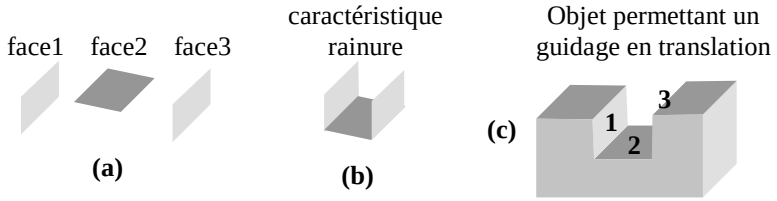


Figure 2-9. *Exemple de caractéristique de forme*

Les caractéristiques permettent en réalité d'associer des entités géométriques à d'autres types d'informations de niveau sémantique plus élevé. En ce sens, les caractéristiques de forme, en regroupant des entités géométriques de bas niveau (faces, arêtes, sommets …) (cf. Figure 2-9a), permettent de définir des entités géométriques de niveau sémantique plus élevé qui peuvent être éventuellement associées à des fonctions. Par exemple, la caractéristique « rainure » (cf. Figure 2-9b) peut être associée à un guidage en translation et peut être appliquée facilement sur un objet préexistant (cf. Figure 2-9c). Cette association, dépendant du domaine considéré, simplifie la tâche du concepteur pour la construction du volume. D'un point de vue dialogue, ceci permet en outre d'améliorer considérablement les interactions entre le système de CFAO et le concepteur dans la mesure où il a la possibilité de modifier rapidement et localement la pièce.

Toutefois, la modélisation par les caractéristiques connaît quelques inconvénients : une application successive de caractéristiques de forme sur le même objet peut correspondre à une autre caractéristique de forme (cf. Figure 2-10a) et donc à une autre fonction, éventuellement désirée ou non. Il en va de même lorsque l'utilisateur modifie simplement un seul des paramètres de la caractéristique (cf. Figure 2-10b). De ce fait, il est nécessaire d'utiliser des techniques reconnaissant automatiquement les caractéristiques de l'objet en cours de construction : elles sont appelées techniques d'extraction [Minich 96, Shah et al. 95, Salomons 94, Weber et al. 92]. Elles restent cependant difficiles à appliquer dans la mesure où elles dépendent du domaine d'application.

La modélisation par les caractéristiques est devenue incontestablement une méthode d'assistance à la construction d'objets complexes. Mais la manipulation de ces entités reste encore une fois à la charge du concepteur, ce qui en fait une méthode de construction manuelle.

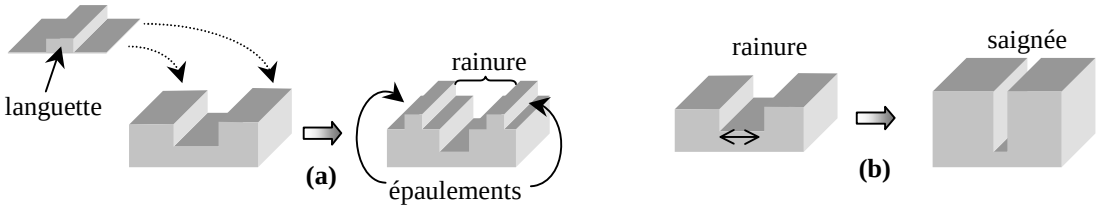


Figure 2-10. *Problème de reconnaissance de caractéristiques de forme après modifications*

2.2.2 Méthode de construction par esquisses

L'esquisse est attachée à la notion de spontanéité et de rapidité ; elle ne répond donc pas à un souci de modélisation précise ni d'achèvement [Danesi et al. 99, Liang et al. 94]. Elle permet par exemple d'exprimer rapidement une forme que le concepteur a à l'esprit sans qu'il lui soit nécessaire de prendre en considération les dimensions, les positions, les orientations … de la forme comme c'est encore partiellement le cas dans les méthodes de construction par les caractéristiques. [Danesi et al. 99] distingue deux types d'esquisses : pour le dialogue et pour la modélisation.

L'esquisse pour le dialogue est une technique approximative d'interaction entre l'homme et la machine. Dans ce sens, elle a pour objectif à la fois de minimiser le nombre d'opérations nécessaires pour obtenir quasiment le même résultat qu'habituellement et à la fois de proposer des techniques d'interactions plus naturelles telle que la désignation à l'aide des mains plutôt qu'un outil électro-mécanique comme la souris. L'esquisse consiste à interpréter les désirs du concepteur exprimés sous différentes formes comportant éventuellement des « maladresses » (cf. Figure 2-11). Plusieurs systèmes utilisent déjà un tel principe soit en 2D (QuickSketch [Eggli et al. 97]) soit en 3D (3Dsketch [Han et al. 97]).

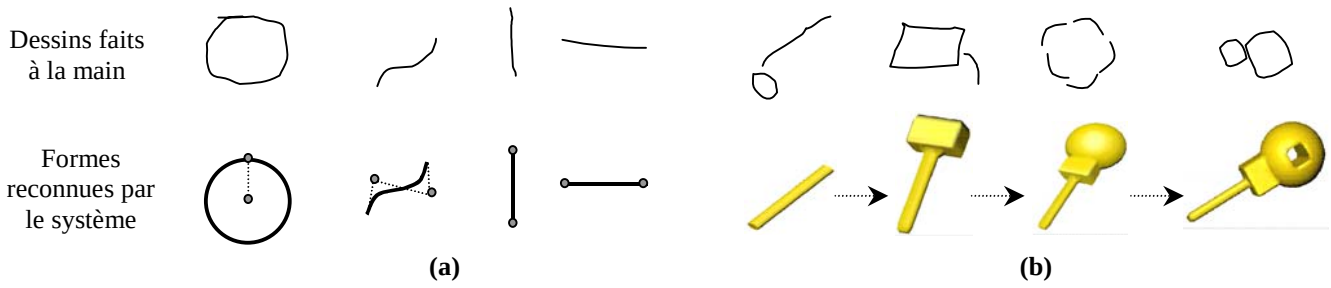


Figure 2-11. *Interprétation de l'esquisse humaine par un système informatique en 2D (a) [Eggli et al. 97] ou 3D (b) [Han et al. 97]*

Par ailleurs, en proposant des outils d'interaction naturels, l'esquisse permet à l'utilisateur de construire rapidement une forme sans qu'il se soucie d'informations fondamentales comme la notion de contraintes géométriques, la valuation de certains paramètres … dans la mesure où le système est capable d'anticiper ces informations. De nombreux systèmes détectent automatiquement des contraintes (parallélisme, perpendicularité …) sur des entités géométriques (faces, arêtes, sommets …) [Eggli et al. 97, Zeleznik et al. 96, Gardan et al. 95b]. Ces contraintes sont souvent nommées contraintes implicites contrairement à celles explicitement données par l'utilisateur. Cette distinction est importante pour que les contraintes dites explicites aient plus de poids lors de la construction automatique de la forme car il n'est pas certain qu'une contrainte détectée automatiquement (implicite) soit réellement désirée par le concepteur.

L'esquisse pour la modélisation [Liang et al. 94] permet d'obtenir rapidement un modèle imprécis (flou,

simplifié) mais représentatif de l'objet conçu ou à concevoir, par analogie avec l'esquisse pour le dialogue. Plusieurs notions peuvent être approchées : la forme de l'objet, ses couleurs, ses dimensions … Par exemple, si l'opérateur souhaite construire une bouteille d'eau, il peut la modéliser de manière approximative en esquissant un cercle et une extrusion à l'aide d'une flèche (esquisse pour le dialogue, cf. Figure 2-12a) pour construire un cylindre. Cette forme cylindrique sera une esquisse pour la modélisation (cf. Figure 2-12b) de l'objet à concevoir (cf. Figure 2-12c). Il est possible ensuite de se servir de la forme esquissée pour construire par raffinement le modèle final de l'objet. L'esquisse pour la modélisation doit être vue comme une simplification du modèle final ; le modèle simplifié est destiné à une utilisation spécifique. Par exemple, le modèle VRML offre la possibilité de modéliser un objet différemment en fonction de la distance par rapport à l'observateur dans le but d'accélérer la visualisation d'une scène tridimensionnelle. Plus l'observateur est éloigné, moins les détails sont importants et VRML choisit automatiquement le modèle géométrique correspondant. On peut donc considérer chaque forme représentée dans le modèle VRML comme une esquisse pour la modélisation. Il existe d'autres types de simplification : par exemple, [Jahami 91] propose de changer un modèle géométrique en une simple face plane sur laquelle est appliquée une texture. Par extension, on peut envisager que les traducteurs de vue (cf. Chapitre 1 §2.3.4) d'un système de conception fonctionnelle assistée par ordinateur sont des outils gérant des esquisses pour la modélisation puisqu'ils extraient du modèle interne une vue plus simplifiée pour un métier donné.

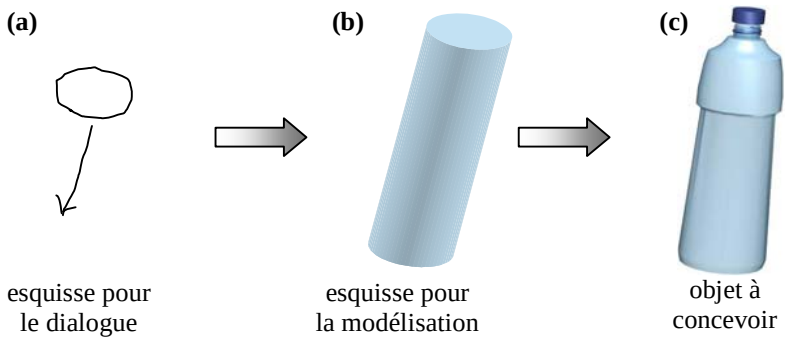


Figure 2-12. *L'esquisse pour le dialogue (a) et l'esquisse pour la modélisation (b) d'un objet réel (c)*

D'un point de vue géométrique, nous constatons que le niveau sémantique des informations manipulées par le système est très peu élevé par rapport à ce que l'on a pu voir précédemment (caractéristiques de forme). Plusieurs solutions sont possibles pour améliorer le niveau sémantique des informations. Une possibilité est d'appliquer une technique d'extraction de caractéristiques de forme sur le modèle esquissé. Comme chaque caractéristique peut être associée à des informations plus conceptuelles, il en résulte que l'esquisse contiendra des informations de niveau sémantique plus élevé. Par ailleurs, une autre alternative est de laisser la possibilité au concepteur d'esquisser ces informations conceptuelles. Mais, cette étude sort du cadre de notre travail.

Néanmoins, le fait que le concepteur décrive et manipule des informations plus naturellement représente un avantage notable par rapport aux méthodes de construction par les caractéristiques de formes. Un autre intérêt est l'anticipation des actions du concepteur qui peut se traduire d'un point de vue géométrique en une détection automatique des contraintes, ce qui assure une certaine automatisation de la méthode de construction. Là encore, le concepteur est obligé d'imaginer la forme du produit à concevoir.

2.2.3 Méthodes fonctionnelles

Les deux méthodes précédentes permettent de décrire, aussi naturellement que possible, la forme du produit que le concepteur souhaite concevoir. Pour utiliser de telles méthodes, il est nécessaire pour le concepteur de connaître a priori la forme du produit. En réalité, il aura fallu étudier au préalable les différentes solutions de réalisation possibles pour n'en retenir qu'une seule. Ensuite, c'est à la charge du concepteur d'imaginer la forme de l'objet qui correspond à la solution de réalisation choisie et qui satisfait au mieux le cahier des charges. Les deux méthodes ne représentent donc qu'une assistance lors de la construction de la forme de l'objet et négligent ainsi les informations qui ont amené à considérer une telle forme.

Les méthodes fonctionnelles ont un objectif plus ambitieux qui est de déduire le plus automatiquement possible la forme de l'objet à partir de spécifications (le cahier des charges) exprimées dans un langage le plus naturel possible. Nous appelons cette déduction la *génération* ou la *synthèse* de la forme de l'objet. Même si la synthèse automatique de la forme semble très complexe et donc inaccessible, les tentatives vont dans ce sens. Certaines de ces tentatives, peu automatiques, ont au moins l'avantage par rapport aux deux méthodes précédentes de conserver (ou d'associer à la forme) les informations qui ont amené à la forme synthétisée. Nous considérons que les informations conservées sont d'un niveau sémantique plus élevé que celles manipulées dans les précédentes méthodes.

Les travaux que nous relatons peuvent s'organiser selon quatre axes :

- l'approche associative consiste à lier des informations fonctionnelles à des informations plus géométriques telles que les caractéristiques de forme ;
- l'approche déclarative permet à partir de propriétés imprécises de l'objet de conception, décrites par le concepteur, de caractériser l'ensemble des solutions ;
- l'approche déductive ne se limite pas à associer simplement une fonction à une forme prédéfinie ; elle tente d'extraire des fonctions suffisamment d'informations pour construire progressivement une décomposition fonctionnelle du produit. Ensuite, une forme est déduite à partir de cette décomposition. La différence avec l'approche déclarative se situe au niveau des informations initiales gérées par l'approche : l'approche déductive prend en considération une description très générale du problème de conception et ne connaît donc pas forcément une idée, même imprécise, de la forme alors que l'approche déclarative connaît a priori la morphologie de la solution ;
- l'approche constructive procède tout autrement en utilisant des entités géométriques de faible niveau pour construire par exemple par combinaison une forme plus complexe ; elle obtient alors énormément de formes qui doivent être examinées a posteriori.

2.2.3.1 L'approche associative

[Feng et al. 96] tentent de représenter les relations entre les caractéristiques de forme et les fonctions pour assister le concepteur lors de la conception détaillée. Les auteurs identifient plusieurs types de relations entre les fonctions : explicite (dépendance ou interdépendance entre deux fonctions) et implicite (relative à la géométrie ou à la précision). De cette façon, les auteurs construisent un graphe pour associer de manière valide les fonctions aux caractéristiques de forme. Cela permet alors, après avoir détaillé la décomposition fonctionnelle, d'associer des fonctions de bas niveau (les feuilles de la hiérarchie fonctionnelle) à des caractéristiques de formes.

L'inconvénient de cette approche est que l'association fonction–forme est à la charge du concepteur et

de ce fait l'assistance du système est très réduite. Il n'y a donc pas à proprement parler de synthèse de la forme du produit.

[Kirschman et al. 96] proposent de décrire les intentions fonctionnelles du concepteur à l'aide d'un ensemble de mots clés (cf. Chapitre 1 §4.3.2) qui, combinés, permettent d'obtenir cent cinquante phrases types. Par exemple, une combinaison des mots clés proposés pourrait donner les phrases suivantes : « créer un mouvement de rotation » ou « convertir un mouvement linéaire en mouvement oscillatoire ». À chacune est associée une ou plusieurs solutions techniques, qui représentent dans une certaine mesure une forme. Cette démarche est sensiblement la même que précédemment. Dès lors que les associations entre les phrases et les solutions techniques sont spécifiées par un expert du domaine, le système peut proposer automatiquement des solutions au concepteur à partir du moment où il aura défini son problème de conception en utilisant le langage proposé. En revanche, le système n'est pas très général dans la mesure où l'ensemble de mots clés est dédié au domaine de la mécanique. De plus, les cent cinquante phrases ne reflètent pas toutes les solutions possibles dans ce domaine.

[BenAmara et al. 96] considèrent qu'en mécanique la chaîne cinématique est une notion incontournable pour la représentation du produit. Le système présenté consiste à déterminer une forme à partir d'une chaîne cinématique. Pour ce faire, aux liaisons inter-pièces sont associées des solutions technologiques. Ensuite, pour chaque solution technologique est proposée une ou plusieurs solutions techniques ; par exemple, un encastrement par obstacle (solution technologique) peut être réalisé par une clavette, une goupille, des cannelures (solutions techniques). Enfin, comme les solutions techniques sont associées à des caractéristiques de forme, les formes fonctionnelles sont déduites. Il reste cependant à les dimensionner en utilisant la notion de torseur.

Là encore, l'intervention du concepteur reste importante dans les divers choix du passage de la fonction à la forme dans la mesure où c'est à lui de choisir les solutions techniques et technologiques parmi une liste proposée par le système. De plus, la forme obtenue du produit est incomplète puisque seule la forme des liaisons inter-pièces a été définie.

L'approche de [Mukherjee et al. 95] consiste à proposer un niveau intermédiaire entre la représentation purement fonctionnelle et le modèle géométrique. Ce niveau est constitué de ce que les auteurs appellent des caractéristiques fonctionnelles qui correspondent assez précisément à la notion de caractéristiques de forme et pour lesquelles une morphologie est décrite. Les caractéristiques fonctionnelles sont reliées par des liens abstraits (fort ou faible) qui matérialisent une morphologie non connue (cf. Figure 2-13a). Des quasi-nœuds peuvent être insérés dans des quasi-liens pour exprimer une déformation à caractère morphologique de ce lien. Cette représentation intermédiaire est manuellement décrite par le concepteur. Ensuite, il associe les fonctions du produit, exprimées à l'aide d'un verbe et d'un complément, aux caractéristiques fonctionnelles par le biais d'une matrice binaire ; la valeur 1 indique que la caractéristique fonctionnelle joue un rôle dans la satisfaction de la fonction. L'avantage de cette représentation intermédiaire est qu'une fois établie, elle autorise un raisonnement abstrait sur la forme de l'objet et permet de générer des variantes de formes (cf. Figure 2-13b). De plus, cette approche repose sur l'utilisation d'une bibliothèque de solutions techniques prédéfinies qui permet de réduire la combinatoire. L'inconvénient majeur est que les propositions des auteurs se limitent à un formalisme : il n'y a pas d'assistance dans la mesure où toutes les tâches sont manuelles. D'autre part, le fait que la matrice soit binaire ne permet pas de connaître le degré avec lequel la caractéristique fonctionnelle satisfait la fonction. De plus, comme dans les approches précédentes, il manque toujours des portions de la forme finale.

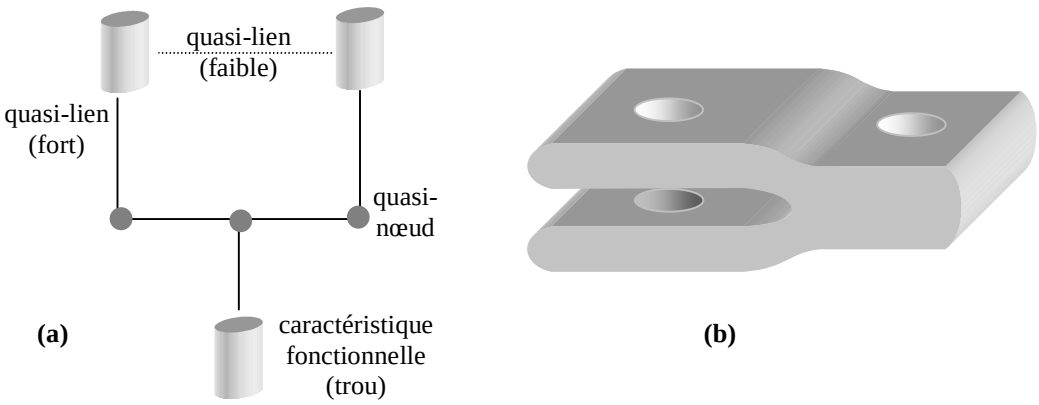


Figure 2-13. *Exemple d'un modèle intermédiaire (a) et d'une forme correspondante (b) [Mukherjee et al. 95]*

On constate qu'aucune de ces méthodes ne génère toute la forme du produit mais uniquement les surfaces fonctionnelles selon un point de vue donné. Par ailleurs, elles ont tendance à utiliser une représentation intermédiaire (chaîne cinématique, quasi-liens et quasi-nœuds) pour associer les fonctions aux formes. L'utilisation d'un modèle intermédiaire est compréhensible dans la mesure où il est ardu d'interpréter directement le cahier des charges pour en déduire une forme.

2.2.3.2 L'approche déclarative

Le principe d'une méthode de construction déclarative consiste à fournir à un utilisateur (même à un non spécialiste) des outils informatiques lui donnant la possibilité d'exprimer une idée de conception sous différents aspects (schémas, esquisses, contraintes, propriétés, langage naturel …) et d'une manière la plus naturelle possible. Les spécifications étant établies par le concepteur et le système vérifiant la consistance des informations fournies, elles sont automatiquement traduites en solutions de conception (dans notre cas en des formes solutions) (cf. Figure 2-14) [Colin et al. 97, Lucas et al. 95]. Dans la mesure où les spécifications sont souvent insuffisantes pour caractériser une seule solution, le système génère un ensemble de solutions (aussi appelé espace de solutions, cf. 3.1). Une fois cet espace caractérisé, il faut présenter les solutions obtenues automatiquement à l'utilisateur : c'est la prise de connaissance. Les auteurs précisent qu'il existe plusieurs moyens de réaliser cette présentation (nous les détaillerons en 3.1).

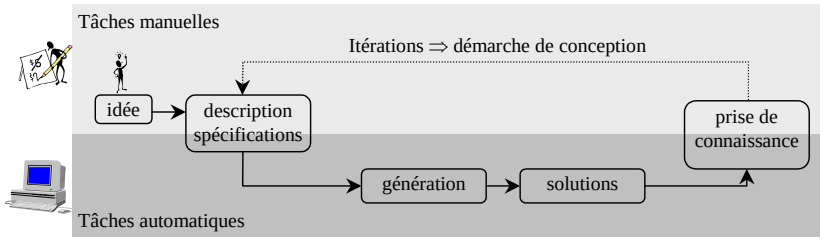


Figure 2-14. *Méthode de construction déclarative [Colin et al. 97]*

Il est certain, comme nous l'avons déjà précisé plus haut (conception par tâtonnements), que les solutions générées automatiquement ne satisferont pas pleinement le concepteur lors de la première analyse. C'est pourquoi, les méthodes de construction déclaratives proposent au concepteur d'améliorer les propriétés du produit en cours de construction, si bien que la solution retenue par le concepteur n'est obtenue qu'après plusieurs itérations (cf. Figure 2-14). De ce fait, les auteurs présentent la démarche de

conception comme une spirale partant du premier cahier des charges, passant par un ensemble de spécifications et d'espaces de solutions intermédiaires, et terminant par la solution satisfaisant au mieux le concepteur.

Ce principe a déjà fait l'objet de nombreuses réalisations et en particulier la modélisation de courbes en Conception Assistée par Ordinateur [Daniel et al. 97]. Les spécifications sont exprimées dans un langage très proche du langage naturel utilisant des termes très représentatifs de courbes planes (aplati, arrondi, élancé, arc, boucle …) ainsi que des quantificateurs flous (trop peu, beaucoup, plus, moins, très …) (cf. Figure 2-15). Le modèle de représentation utilisé pour modéliser les courbes est une association continue de Segments Élémentaires de Courbe (SEC) exprimé à l'aide d'une B–Spline. La génération automatique consiste à déterminer le nombre de points de contrôles, leur position …

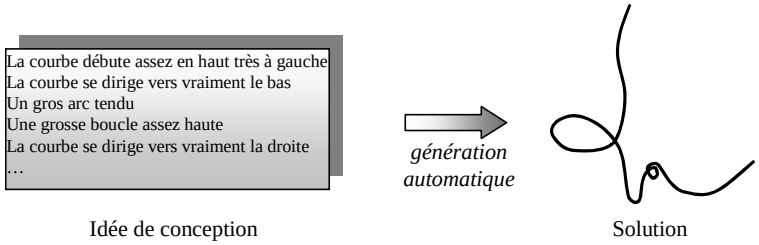


Figure 2-15. *Construction déclarative de courbes [Daniel et al. 97]*

En théorie, l'intérêt d'une approche déclarative réside dans la simplicité de la conception de produits. Cette technique décharge le concepteur d'un grand nombre d'opérations de construction et de manipulation de bas niveau comme c'est le cas dans les méthodes de construction habituelles [Colin et al. 97]. D'autre part, le fait que le système soit capable d'analyser les spécifications permet de construire des cahiers des charges correctement contraints pouvant alors être utilisés pour d'autres applications éventuellement plus dédiées.

On constate que les informations manipulées sont de haut niveau sémantique puisqu'elles sont exprimées dans un langage proche de celui utilisé habituellement par le concepteur (schémas, esquisses, contraintes, propriétés …). Toutefois, ces méthodes relativement attrayantes comportent encore des problèmes prédominants tels que la génération automatique de solutions, l'interprétation des informations fournies ou encore la comparaison des solutions obtenues. Par conséquent, elles ne sont appliquées qu'à des domaines très particuliers. Ainsi, la génération automatique consiste véritablement à interpréter les propriétés imprécises ou floues de l'objet à concevoir décrites dans les spécifications. Ceci implique d'une certaine façon que le système connaît la morphologie de l'objet à concevoir puisqu'il est censé comprendre les propriétés utilisées dans les spécifications. Le véritable intérêt de cette approche consiste à pouvoir générer des solutions à partir de données floues : les valeurs des propriétés.

Au final, ce que les auteurs appellent modélisation déclarative se rapproche de la conception fonctionnelle utilisée par d'autres auteurs. Dans les deux cas, le problème de conception est énoncé sous formes de fonctions, le but étant d'obtenir une forme qui satisfait ces fonctions.

2.2.3.3 L'approche déductive

Ces méthodes interprètent le cahier des charges pour en déduire d'autres informations que la forme. C'est seulement à partir de ces informations qu'une forme peut être obtenue.

En étudiant les interactions existant entre le produit en cours de conception et son milieu environnant,

[Constant 96] identifie les fonctions principales, auxiliaires et contraintes du produit. Ces fonctions sont alors caractérisées par la nature et l'intensité du flux auxquelles elles correspondent (cf. Chapitre 1 §4.3.1). Ensuite, grâce à des opérateurs de décomposition, une représentation de plus en plus précise du produit est obtenue. C'est au concepteur de choisir l'opérateur qu'il veut appliquer ; de ce fait, la décomposition n'est pas automatique. Comme un flux traverse deux surfaces fonctionnelles (cf. Figure 1-8), le raffinement de chacun des flux identifiés permet d'ajouter de nouvelles Surfaces qui sont Associées Technologiquement et Topologiquement (SATT) aux précédentes. Ainsi, la structure topologique du produit est obtenue. D'un point de vue informatique, un ensemble de SATT est représentée à l'aide d'un arbre binaire : les feuilles de l'arbre sont des surfaces fonctionnelles et les nœuds de l'arbre sont des SATT. La racine de l'arbre représente alors l'ensemble des surfaces fonctionnelles d'une pièce simple appartenant à un mécanisme [Desrochers 91] (cf. Figure 2-16).

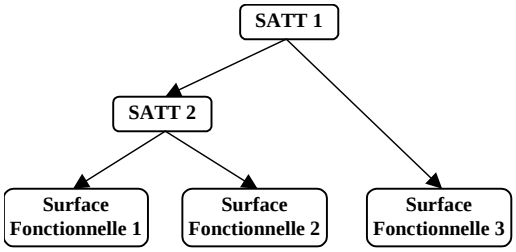


Figure 2-16. *Représentation d'un SATT [Desrochers 91]*

Le raffinement des informations fonctionnelles contenues dans le cahier des charges à l'aide d'opérateurs de décomposition permet de raffiner progressivement la forme du produit et donc obtenir, une fois la décomposition terminée, un graphe de surfaces fonctionnelles. C'est ce principe qui distingue cette méthode des approches associatives. L'intérêt de ces travaux est d'offrir un formalisme et des outils de raffinement qui permettent d'obtenir une topologie du produit. Ce formalisme permet de caractériser les surfaces fonctionnelles suivant leur type (sphérique, cylindrique …). Néanmoins, aucune dimension ou position ne sont données à ces surfaces. Par ailleurs, il restera à déterminer les surfaces complémentaires, reliant les surfaces fonctionnelles, pour déterminer complètement la forme de l'objet.

2.2.3.4 L'approche constructive

Dans les approches précédentes, les solutions obtenues sont celles imaginées par le concepteur ; elles ne sont donc pas construites automatiquement par le système. De plus, seule une géométrie incomplète est obtenue (surfaces fonctionnelles, représentation intermédiaire …) ; ce qui rend toute visualisation plus complexe.

Les algorithmes génétiques sont une des voies de recherche pour produire automatiquement des solutions. L'idée générale consiste à simuler le modèle d'évolution des organismes biologiques. La forme d'un organisme biologique (le phénotype) est représentée par des informations génétiques (le génotype). Un génotype contient à la fois les règles de développement de l'organisme et le processus de développement lui-même. Un algorithme génétique est caractérisé par une population initiale sur laquelle est appliquée différents mécanismes d'évolution tels que : le croisement (mélange de deux génotypes), la mutation (modification aléatoire d'un génotype) …

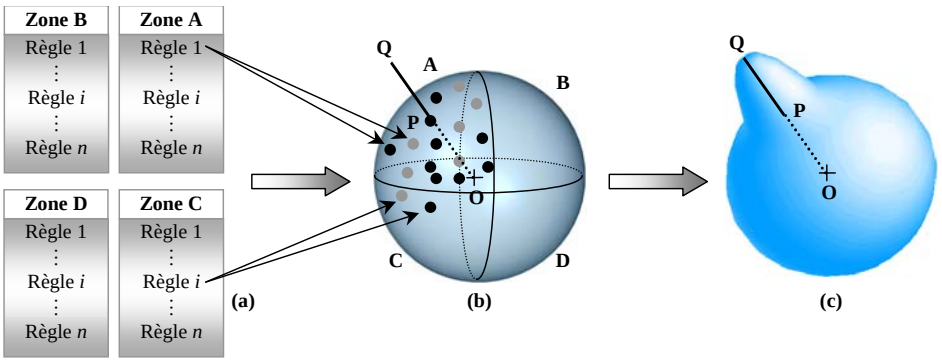


Figure 2-17. *Un exemple de phénotype considéré par [Taura et al. 98]*

L'objectif de [Taura et al. 98] consiste à générer un génotype à partir d'un phénotype cible. Le phénotype est une sphère découpée en régions auxquelles des règles de production sont associées (cf. Figure 2-17a). Chaque règle produit deux points de $\mathbb{S}^3$ situés à la surface de la sphère et appelés cellules (cf. Figure 2-17b). Pour un point P de la surface de la sphère, la forme de l'objet est obtenue en (cf. Figure 2-17c) :

- calculant la densité δ de cellules autour du point P ;
- convertissant la densité δ en une distance d. Cette distance représente la longueur du segment d'extrémité O (centre de la sphère) passant par P. L'autre extrémité Q du segment appartient à la surface de l'objet en cours de construction.

La stratégie utilisée par les auteurs consiste à croiser l'ensemble des règles de production (génotypes) de deux individus. L'évaluation d'un individu se fait en comparant les distances des segments, qui vont du centre de la sphère à un point de la surface, de l'individu et de la forme cible.

[Rosenman 96] utilise les algorithmes génétiques pour générer automatiquement des plans 2D de bâtiments. Le phénotype est une surface composée de carrés unitaires. Il est représenté par une séquence de vecteurs unitaires horizontaux ou verticaux. Le génotype est un ensemble de règles constituées de deux phénotypes de la génération précédente, unis par un coté tel que les vecteurs correspondant soient opposés. En définitive, le génotype peut se représenter par une simple liste d'association (cf. Figure 2-18).

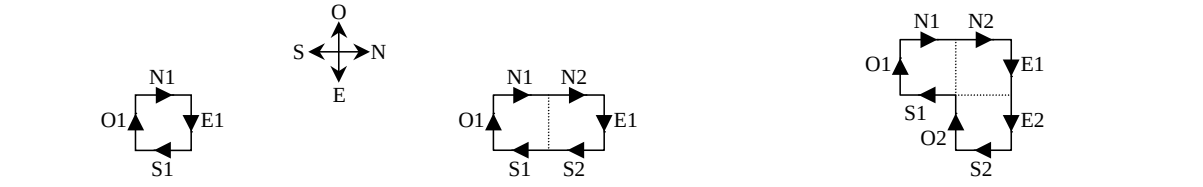


Figure 2-18. *Méthode de construction d'une forme par [Rosenman 96]*

L'opération de croisement consiste à découper au même endroit les listes de deux génotypes. Les nouveaux génotypes sont obtenus en croisant les morceaux de listes pour conserver la longueur de deux phénotypes. L'auteur utilise cette technique à la fois pour générer la forme des pièces du bâtiment et pour agencer les pièces entre elles.

L'avantage de l'approche productive est son caractère aléatoire qui permet de générer des formes originales donc innovantes. Son principal inconvénient réside dans le trop grand nombre de formes

générées. Cette explosion combinatoire rend cette technique très peu performante. De plus, les formes ainsi obtenues ne sont pas forcément des solutions au problème posé. Il convient encore de vérifier si les fonctions du produit sont satisfaites par la forme générée ; c'est-à-dire si c'est une solution ou non. Par ailleurs, les opérations d'évolution (croisement, mutation …) sont le plus souvent empiriques. Les stratégies internes à ces opérations sont souvent basées sur l'intuition ou l'expérience de l'auteur de l'algorithme génétique.

2.3 Synthèse

Nous avons présenté des modèles pour stocker une forme du produit à concevoir ainsi que des méthodes pour fournir la forme à ces modèles. Certaines de ces méthodes sont considérées comme de bas niveau sémantique (opérateur d'Euler, involutions, opérations booléennes …). D'autres, par contre, tentent d'assister le concepteur lors de la construction de l'objet dans la mesure où elles proposent des entités géométriques de plus haut niveau. Là encore, la construction de la forme ne se fait que manuellement, et, de ce fait, la charge du concepteur reste importante. Les dernières méthodes (approches productive et déclarative) ont pour objectif de construire le plus automatiquement possible la forme à partir d'un ensemble de spécifications. Comme le cahier des charges d'un problème de conception est trop souvent sous–contraint, l'automatisme des méthodes fait qu'elles ne peuvent générer qu'un ensemble de solutions. Nous étudions dans le paragraphe suivant les moyens permettant de comparer les solutions et ainsi de proposer les plus prometteuses. Par ailleurs, il faudrait étendre davantage nos investigations en étudiant les différentes techniques de modélisation des connaissances dans la mesure où l'automatisme implique un comportement intelligent. Toutefois, c'est au delà du cadre de notre travail car cela représente un domaine à part entière [Harjani 87].

3 Choix des meilleures solutions de conception

Dans le paragraphe précédent, nous avons présenté différents moyens de construire des formes en utilisant différents modèles. Dans ce qui suit, nous nous intéressons plus précisément aux multiples solutions synthétisées par les méthodes les plus automatiques manipulant des informations de haut niveau sémantique (méthodes fonctionnelles). D'une manière ou d'une autre, des formes auront été générées, mais les limitations des méthodes ne permettent pas de synthétiser les meilleures solutions dès les premières itérations. Ce n'est qu'en synthétisant par tâtonnements que le système, aidé par le concepteur, propose un ensemble de solutions très satisfaisantes. Le principe est le suivant : le système synthétise un ensemble de formes qu'il suppose être satisfaisantes d'après le cahier des charges fourni par le concepteur. Les meilleures solutions sont présentées au concepteur. Celui-ci doit confirmer les solutions obtenues. Il peut par ailleurs orienter le système vers un ensemble de solutions plus restreint que celui obtenu par le système.

C'est pourquoi, dans un premier temps, nous présentons la notion d'espace de solutions qui permet de caractériser toutes les solutions de conception synthétisées par le système. Ensuite, nous étudions les moyens permettant une estimation des solutions. Une fois que l'on peut comparer des solutions entre elles, les meilleures solutions synthétisées par le système peuvent être présentées au concepteur. Nous étudierons enfin comment le concepteur peut orienter le système.

3.1 Espace des solutions

Comme le souhait initial est de fournir une assistance lors des toutes premières phases de conception, il est normal de considérer un ensemble sous–contraint de spécifications. Nous considérons que la décomposition fonctionnelle permet d'obtenir une ou plusieurs morphologies de la forme de l'objet. L'optimisation permet ensuite d'obtenir les meilleures valeurs des paramètres de la morphologie. De ce fait, le nombre de solutions étant vaste, il faut pouvoir parcourir l'ensemble de ces solutions. Il faut donc, dans un premier temps, étudier la continuité de cet espace qui influe sur le choix du parcours à utiliser. Ensuite, nous étudions les diverses méthodes d'optimisation permettant le choix des meilleures solutions parmi un ensemble relativement vaste.

3.1.1 Continuité de l'espace des solutions

Soient Ω l'univers des formes à trois dimensions et Σ l'ensemble des formes solutions satisfaisant partiellement ou totalement les spécifications du concepteur. Comme une forme est un sous–ensemble de $\mathbb{R}^3$, Ω est représenté par les parties $\mathbb{P}$ de $\mathbb{R}^3$ aussi notées $\mathbb{P}(\mathbb{R}^3)$. Le cardinal de $\mathbb{P}$ est infini car le nombre de formes à trois dimensions est infini. Comme une forme est une partie de Ω, Σ est un ensemble de parties de Ω. Si on considère qu'une forme est une instance d'une morphologie –les paramètres de la morphologie sont valués–, il est possible que plusieurs formes solutions aient la même morphologie. Dans ce cas, plutôt que de stocker toutes les instances de formes solutions, nous préférons stocker les morphologies solutions et stocker les valeurs pour lesquelles la forme est solution. De ce fait, Σ peut être considéré comme un ensemble de morphologies. Chaque morphologie définit une classe de solutions que nous appelons Γ_m (cf. Figure 2-19). Comme les valeurs des paramètres sont réelles, une méthode simple, mais pas nécessairement la meilleure, consiste à conserver les valeurs minimales et maximales pour les différents paramètres de chaque classe. Cela définit un intervalle réel pour chaque paramètre : de ce fait, on peut préciser qu'une classe est continue par morceaux. Par contre, la continuité est perdue entre les classes de forme puisqu'elles sont morphologiquement différentes.

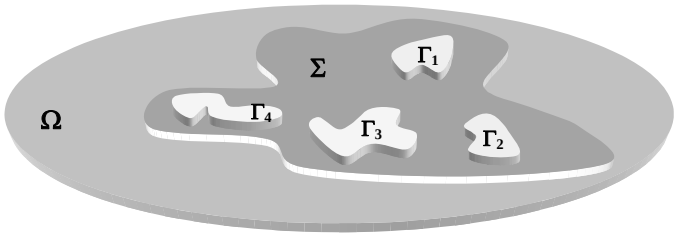


Figure 2-19. *Les différents espaces de formes*

Nous supposons que les bornes des intervalles ne sont pas infinies puisqu'elles représentent des paramètres réalistes de l'objet (il est inconcevable d'imaginer une voiture mesurant 1000km ou 5mm). Les bornes minimales et maximales d'un paramètre sont définis par un expert du métier concerné. Ensuite, c'est l'interprétation des spécifications qui permettra de réduire chaque intervalle.

Les meilleures solutions doivent être présentées au concepteur. Nous verrons dans le paragraphe suivant que le choix des meilleures solutions implique l'étude d'un grand nombre de solutions. C'est-à-dire qu'à partir d'une solution donnée, il faut déterminer les solutions voisines –très proches de la précédente d'un point de vue géométrique. Au sein d'une classe de solutions Γ_m, une méthode, pas nécessairement la

mieux adaptée, consiste à faire varier faiblement un des paramètres de la classe à condition que la variation effectuée appartienne à l'intervalle de variation du paramètre considéré. Cette restriction est du au fait qu'une classe de solutions est continue par morceaux.

Entre classes de formes, il est effectivement possible de découvrir des formes permettant de passer d'une morphologie à une autre ; une technique réalisant cette transformation est appelée morphing [Cohen-Or et al. 98, Decaudin 96]. Toutefois, l'utilisation de cette technique ne permet pas forcément d'obtenir des formes solutions. Prenons comme illustration de cet inconvénient la conception de tuyauterie métallique. Une contrainte de conception très souvent intégrée dans le cahier des charges est le prix de revient du produit. Et, supposons que deux classes de solutions soient obtenues : les tuyaux à section circulaire et à section carrée. Le morphing peut éventuellement proposer une section carrée arrondie (cf. Figure 2-20). Néanmoins, ce type de tuyaux coûte plus cher que les autres et peut ne pas être considéré comme une solution.

Figure 2-20. *Exemple de forme générée par morphing*

En considérant que chaque paramètre d'une classe Γ_i est défini sur un intervalle réel, il est informatiquement irréalisable de parcourir un espace même continu dans son intégralité. Le moyen le plus commun de résoudre ce problème est de parcourir de manière discrète l'ensemble des intervalles des paramètres d'une classe. Cela engendre un autre problème qui est de négliger éventuellement une très bonne solution contenue dans l'espace (cf. Figure 2-21).

Un parcours exhaustif ne peut être utilisé idéalement que lorsque les domaines de définition des paramètres sont déjà discrets comme c'est le cas par exemple dans la nomenclature de pièces mécaniques (le diamètre d'une vis varie dans {5 cm, 6cm, 8cm …}). Il ne faudrait pas non plus que le nombre de valeurs possibles pour un paramètre discret soit trop important ; si tel est le cas, on rejoint le problème précédent.

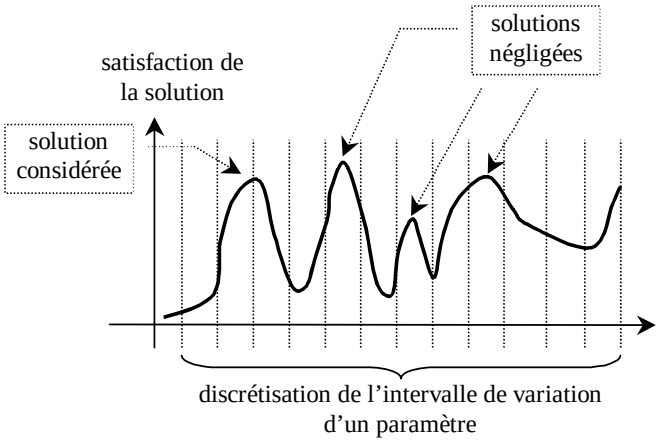


Figure 2-21. *Problème de discrétisation d'un espace continu*

3.1.2 Méthodes d'optimisation

L'optimisation consiste en une opération permettant d'obtenir le meilleur élément d'un ensemble dans des conditions bien définies [Pun 72]. L'importance de cette opération est capitale car elle permet d'automatiser la recherche de solutions. Toutefois, sa résolution pratique demeure encore difficile pour certaines raisons :

- il est nécessaire de définir d'une façon précise les éléments d'un problème d'optimisation ;
- il est important d'organiser ces éléments de telle sorte que le problème paraisse soluble ;
- il faut choisir la méthode de résolution adaptée au problème à résoudre.

Ces difficultés peuvent paraître encore plus importantes en réalité dans la mesure où elles sont inter–dépendantes. En effet, le choix d'une méthode diminue considérablement la liberté de définition et de structuration du problème. Réciproquement, il est possible que les éléments du problème soient définis et organisés de telle façon qu'on ne puisse pas trouver de méthodes appropriées.

En fonction de la formulation du problème de manière mathématique (problème non contraint, contraint, linéaire ou non linéaire, dépendant ou non du temps …), [Pun 72] répertorie les méthodes adaptées pour chacun des problèmes (théorie des minimums et maximums, multiplicateurs de Lagrange, méthode du simplexe …).

Toutes ces méthodes mathématiques sont dites *exactes* dans la mesure où elles sont capables de trouver la meilleure solution si on leur en laisse le temps et si on leur donne les moyens matériels nécessaires. Toutefois, l'utilisation de telles méthodes implique qu'il est possible de caractériser le problème à résoudre sous forme mathématique ; or ce n'est pas toujours le cas. Pour résoudre des problèmes difficilement exprimables sous forme mathématique, [Prins 97] présente brièvement des heuristiques (trouver une solution réalisable en tenant compte d'une fonction de coût sans garantir l'optimalité) en les classant en trois catégories :

- les méthodes construisant une seule solution en réalisant à chaque étape de la construction des choix partiels et définitifs. Si chaque option choisie est la plus avantageuse localement, la méthode est alors appelée *gloutonne* ;
- les méthodes locales partent d'une solution (en réalité d'un état initial) et construisent par transformation son voisinage. Elles recherchent dans ce voisinage une autre solution améliorant la précédente (ce n'est pas forcément la meilleure solution du voisinage qui est recherchée). Le processus s'arrête quand on ne peut plus améliorer la solution courante. L'inconvénient de cette approche est que la solution définitive peut correspondre à un maximum local et non global. Un exemple d'heuristique locale est la méthode du gradient ;
- les méthodes globales, appelées aussi méta–heuristiques évitent le piège du maximum local en considérant aussi des solutions qui peuvent temporairement aggraver la solution précédente (diminuer le résultat de la fonction de coût). Le recuit simulé, les méthodes taboues et les algorithmes génétiques sont des méthodes globales bien connues.

De bons résultats peuvent être obtenus en combinant une approche locale à une approche globale, comme le confirme l'expérience de [Hertz].

Toutes ces méthodes utilisent un concept commun qui est la fonction de coût. Cette fonction permet

d'estimer une solution selon un certain nombre de critères, impliquant alors la possibilité de comparer des solutions entre elles. Nous étudions dans le paragraphe suivant les différentes façon d'estimer.

3.2 Estimation d'une solution

Nous avons vu dans les paragraphes précédents qu'une génération automatique fournit un ensemble vaste de solutions. De ce fait, il est indispensable d'utiliser des techniques d'optimisation recherchant automatiquement les meilleures solutions car il est impossible de présenter toutes les solutions au concepteur. La recherche des meilleures solutions implique donc la nécessité de les comparer. Il faut savoir qu'une comparaison implique deux notions interdépendantes [Ullman 97] : l'estimation d'une solution de conception et la prise de décision concernant la solution à retenir. En fait, c'est l'estimation des solutions de conception par rapport aux spécifications qui fournit les informations nécessaires à la prise de décision. Même si le système est capable d'estimer automatiquement les solutions, il peut s'avérer que la meilleure solution calculée automatiquement par le système ne soit pas en fin de compte la solution de conception retenue par le concepteur. Le système ne peut donc que suggérer les meilleures solutions et c'est au concepteur de choisir la solution finale parmi les solutions proposées par le système –c'est-à-dire prendre une décision. Nous verrons que l'estimation peut se faire manuellement ou automatiquement en fonction du niveau d'abstraction considéré, donc en fonction de la phase de conception en cours d'étude.

3.2.1 Jugement subjectif basé sur la faisabilité

Bien avant la génération de solutions de conception, les démarches de conception conseillent d'envisager le plus grand nombre de concepts (techniques ou technologiques) car cela favorise l'innovation. Une fois le concept choisi, il peut être traduit en solutions et donc être adapté au cas de conception considéré. Le jugement de faisabilité s'applique au choix d'un concept. Il est généralement nuancé de trois façons, exprimant les réactions des concepteurs face à un concept à appliquer [Ullman 97] :

- « ce n'est pas faisable, ça ne marchera jamais » ;
- « ça peut marcher mais il faut autre chose » ;
- « ça vaut la peine d'être considéré ».

Si un concept semble inapplicable sur le produit en cours de conception, il doit tout de même être brièvement étudié selon différents points de vue avant de le rejeter définitivement. Il faut alors se demander pour quelles raisons il n'est pas utilisable. L'étude de la non faisabilité pourra faire jaillir éventuellement d'autre concepts, d'autres idées. Un rejet immédiat peut s'expliquer par plusieurs raisons :

- le concept est technologiquement irréalisable à l'heure actuelle ;
- il ne répond pas aux spécifications demandées ;
- l'être humain possède une tendance naturelle à préférer la tradition aux changements. De ce fait, il préférera utiliser un concept qui a déjà fait ses preuves plutôt qu'un nouveau concept sur lequel il faudra investir en temps et en argent.

Ce jugement est une comparaison basée sur une expérience de conception emmagasinée lors des nombreux cas de conceptions précédents lors desquels des concepts ont aussi été choisis et appliqués.

Ainsi, plus le nombre de cas étudiés est important, plus le choix sera juste. C'est pourquoi, à l'inverse de l'être humain possédant une grande capacité à synthétiser, la machine est actuellement incapable d'appliquer une telle évaluation.

3.2.2 Maison de la qualité (QFD)

Dès les toutes premières phases de conception, l'estimation est déjà utilisée. Les premières données utilisées lors d'une conception sont les spécifications du client. Celles-ci sont traduites en spécifications plus techniques appelées spécifications d'ingénierie. C'est une étape primordiale dans le sens où une mauvaise caractérisation des spécifications d'ingénierie entraîne la conception d'un produit inadapté pour le client. Ainsi, des techniques sont utilisées pour générer des spécifications d'ingénierie adaptées au problème de conception et une des plus populaires est la maison de la qualité[20] (cf. Figure 2-22).

Le besoin du client est représenté par l'entrée ou le « quoi », le quoi est résolu par le « comment » qui correspond aux différentes fonctions du produit à réaliser ou aux spécifications d'ingénierie. Ces spécifications sont une reformulation du besoin du client en termes de paramètres qui peuvent être mesurés à l'aide de valeurs. Il est possible de comparer (donc d'estimer dans un certain sens) les spécifications du client avec les produits déjà existants, la comparaison est caractérisée par le garage de la maison. L'estimation se fait par le client de manière tout à fait subjective. Le corps de la maison représente les dépendances existantes entre les paramètres des spécifications et les besoins du client ; chacune d'elles étant affectée d'un poids. Les dépendances au sein même des spécifications sont établies dans le toit de la maison ; par exemple, il est possible qu'un paramètre de conception dépende d'un autre paramètre. Ensuite, chacune des fonctions du comment est estimée et le résultat de l'estimation est représenté par le « combien », c'est-à-dire les fondations de la maison.

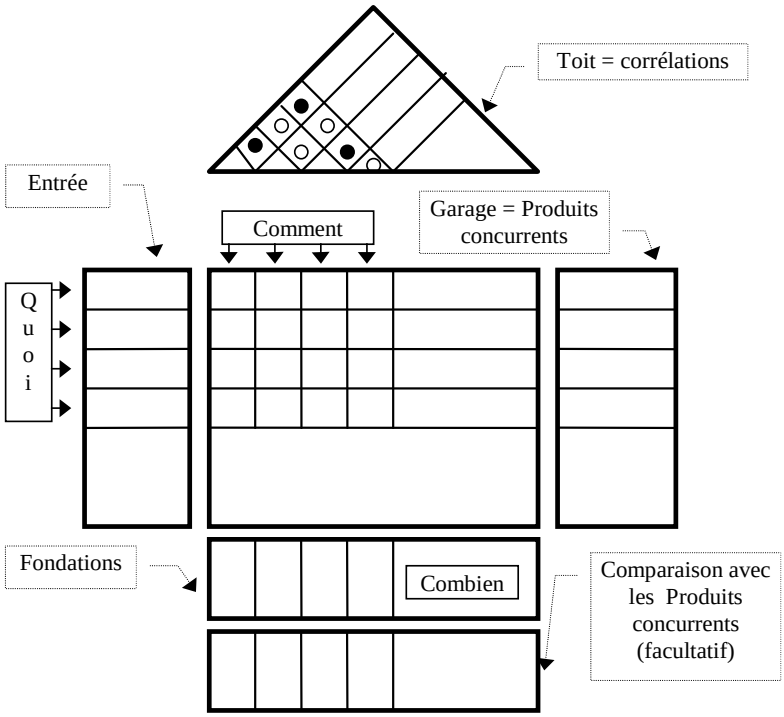


Figure 2-22. *Le principe de la « maison de la qualité »*

[20] Quality Function Deployment en anglais (QFD)

Une fois que chaque cellule de la maison de la qualité est renseignée, il est possible d'appliquer exactement la même démarche en construisant une nouvelle maison où le quoi est déterminé par le comment de la maison précédente. De ce fait, c'est une démarche itérative qui peut être appliquée à tous les niveaux de la conception (fonctionnelle, détaillée …). L'inconvénient réside dans le temps pris pour la constitution des différentes maisons ; en outre, il est difficile d'extraire une vue synthétique de toutes les maisons obtenues. Enfin, l'estimation et le renseignement de chaque cellule se fait en utilisant une technique de réflexion en groupe tel que le brainstorming. Ceci en fait donc une démarche complètement manuelle qui paraît très difficile à automatiser.

3.2.3 Utilisation de statistiques

Comme nous avons pu le voir dans le chapitre 1, [Shimomura et al. 96, Shimomura et al. 98] propose une démarche de conception qui consiste en premier lieu à décrire une fonction du produit à concevoir suivant le formalisme de FBS (utilisant la notion de modificateur). Ensuite, le concepteur est censé décrire le comportement qui réalise la fonction en cours de description. Enfin, on évalue la qualité du comportement obtenu. Ceci est fait en deux étapes :

- premièrement, si le comportement réalise la fonction alors il est conservé. A l'inverse, si le comportement n'est pas satisfaisant, il est rejeté. C'est une estimation binaire ;
- dès qu'un comportement satisfait la fonction, on étudie la façon dont le comportement réalise les modificateurs (appartenant à la fonction) ; c'est à dire que le comportement est estimé dans le détail. L'intérêt d'une estimation plus fine est que le concepteur, lorsqu'il a proposé plusieurs comportements pour la même fonction, peut conserver celui qui réalise la fonction au mieux.

Comme un modificateur est lié à des paramètres de conception du produit, on peut déterminer les valeurs de ces paramètres pour lesquels le modificateur est le mieux réalisé ; et par conséquent la fonction. Une hypothèse des auteurs est de considérer une fonction subjective comme une combinaison linéaire de fonctions objectives dépendant chacune d'un seul paramètre. Par exemple, la beauté d'une photocopie est décomposée en « clairement », « finement » et « un support clair ». De cette façon, on peut déterminer la satisfaction d'une fonction subjective en calculant la satisfaction des fonctions objectives. L'objectif est de déterminer :

- les courbes exprimant la satisfaction des fonctions objectives ;
- le poids de chaque fonction objective pour ainsi calculer la satisfaction de la fonction subjective ;
- la valeur maximum de satisfaction de la fonction subjective.

Pour déterminer la courbe de satisfaction d'une fonction objective, il faut tout d'abord établir la liste des valeurs possibles du paramètre associé à cette fonction. Pour chacune de ces valeurs, un prototype est réalisé. Par chance, dans le cas du photocopieur, les prototypes se limitent simplement à la réalisation de photocopies qui peuvent être réalisées simplement en changeant les paramètres du photocopieur. Mais de manière générale, comme un produit est constitué d'un nombre très important de paramètres, il est difficile d'envisager une telle méthode d'investigation.

Ensuite, on fait réagir chaque personne d'une population donnée en lui demandant de choisir la valeur du paramètre qui lui plaît. Pour chaque valeur du paramètre, on calcule le pourcentage de personnes qui ont préféré cette valeur. On peut donc interpoler les statistiques obtenues (cf. Figure 2-23).

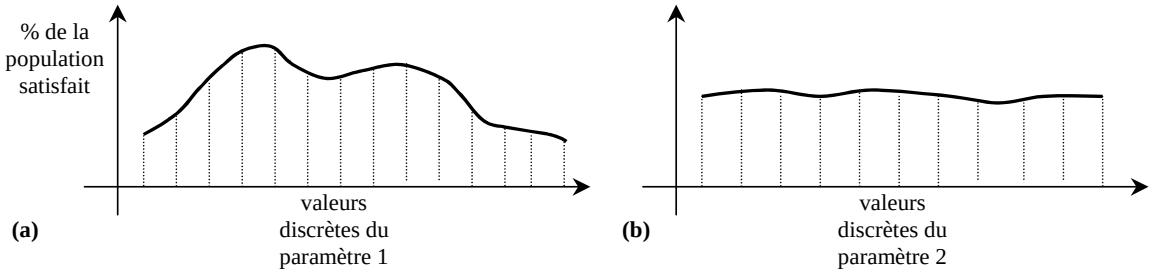


Figure 2-23. *Interpolation des réactions d'une population [Shimomura et al. 98]*

Les courbes interpolées déterminent la satisfaction de la fonction objective. Pour calculer le degré de satisfaction de la fonction subjective, il ne reste plus qu'à déterminer le poids de chacune des fonctions objectives. Ce poids est déterminé à l'aide de l'équation de la Figure 2-24 de telle façon que si la courbe est uniforme (la satisfaction est quasiment la même quelle que soit la valeur du paramètre, cf. Figure 2-23b), le poids de la fonction correspondante doit être très faible. À l'inverse, si la valeur d'un paramètre fait l'unanimité (cf. Figure 2-23a), le poids de la fonction associée à ce paramètre doit être important.

À l'inverse des deux méthodes précédentes, l'application d'une telle méthode ne peut se faire que très tardivement lors de la conception. En effet, le concepteur est obligé de faire réagir un échantillon de personnes sur le produit en cours de conception. Cela implique impérativement de valuer les paramètres physiques qui semblent être prédominants.

$$poids = 1 + \frac{\sum_{i=1}^{n} p_i \log_2 p_i}{\log_2 n}$$

n est le nombre de valeurs possibles du paramètre
p_i est la satisfaction de la $i^{ème}$ valeur du paramètre

Figure 2-24. *Formule permettant de calculer le poids d'une fonction objective [Shimomura et al. 98]*

3.2.4 Formalisation par la logique floue

[Desmontils 96] propose une formalisation du cahier des charges du client supposé être exprimé dans un langage proche du langage naturel. Cette formalisation, réalisée à l'aide de la logique floue, sert à une meilleure interprétation de la part d'un système informatique visant aussi bien à vérifier les solutions de conception obtenues qu'à générer des solutions par tirage aléatoire.

Selon le vocabulaire de l'auteur, le cahier des charges est une *description* constituée d'un ensemble de *propriétés* exprimées dans le langage naturel telles que : « le cube est grand », « le cube est de 2m », « le cube est beaucoup plus grand que la pyramide », « le cube n'est pas petit » … Le *domaine de description* d'une notion ou d'un paramètre consiste à recenser les valeurs possibles prises par cette notion ou ce paramètre. Or, habituellement, la taille d'un objet est définie par une valeur bien précise. La logique floue consiste à associer une propriété (grand par exemple) à un ensemble de valeurs du domaine de description par ce qu'on appelle une *fonction d'appartenance* $\mu_{\text{propriété}}$. Par exemple, plutôt que d'associer une valeur unique (1m70 pour un être humain) à la notion de grandeur, la logique floue se contente d'associer un ensemble de valeurs à cette même notion. Ainsi, plus une personne est proche de 1m70, plus cette personne est grande ; inversement, plus elle dépasse 1m70, moins elle est grande (en réalité, elle est *très* grande) : ceci explique l'allure de la courbe de la Figure 2-25. La fonction μ est définie sur l'intervalle [0, 1] et possède

plusieurs représentations possibles pouvant être choisies en fonction du domaine de conception.

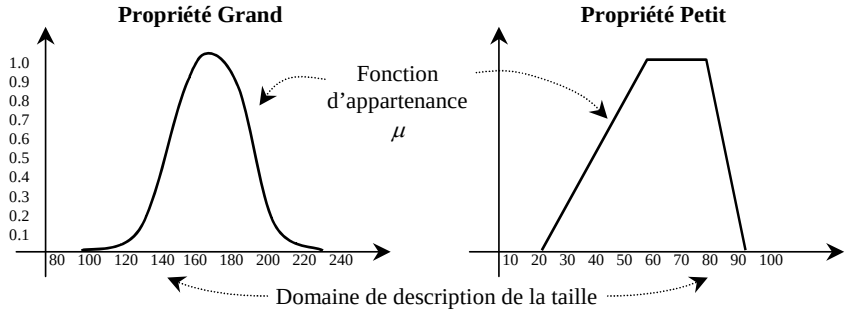


Figure 2-25. *Principe d'association de la logique floue*

De cette façon, l'auteur caractérise la majorité des termes du langage en lui associant à chaque fois une fonction d'appartenance. Il distingue toutefois quatre types de descriptions qui peuvent être formalisées par la logique floue :

- les descriptions simples : « X est Propriété_Simple » où X est un objet que l'on définit par une propriété simple, par exemple « petit », « grand » …
- les descriptions paramétrées : « X est Propriété_Paramétrée » par exemple « le cube est entre 1m70 et 1m90 » ou « le cube est de 1m80 ». La propriété paramétrée ne fait que modifier le domaine de description du domaine sur lequel s'applique la fonction d'appartenance ;
- les descriptions relatives : « X est Propriété_Relative Y » par exemple « le cube est assez proche de 1m80 » ou « le cube est très loin de la pyramide » ;
- les descriptions de comparaison : « X est Propriété_Relation Y » par exemple « le cube est plus grand que la pyramide » ou « le cube est un peu moins grand que la sphère ».

Des modificateurs peuvent être appliqués sur chacune de ces quatre descriptions. Un modificateur correspond par exemple à « extrêmement », « très très », « très », « assez », « normalement », « assez peu » … Le cas du modificateur « normalement » est particulier car il est considéré comme un modificateur vide ; c'est-à-dire qu'il ne change rien à la description. L'application d'un modificateur influe simplement sur la fonction d'appartenance en la déplaçant positivement ou négativement (vers la droite ou vers la gauche) par rapport au domaine de description (cf. Figure 2-26).

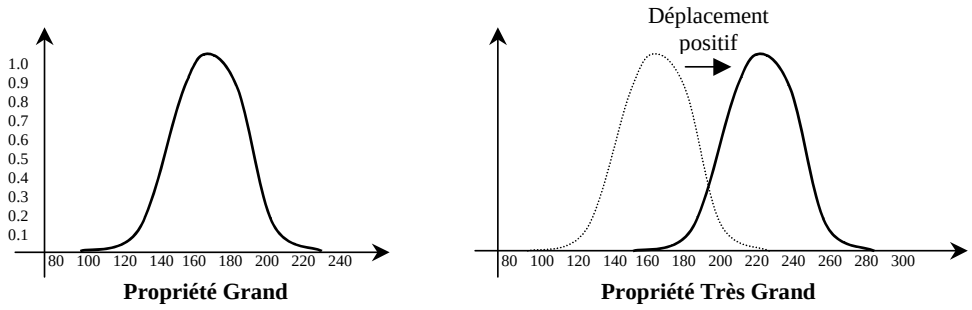


Figure 2-26. *Application d'un modificateur sur une propriété*

L'auteur précise, dans le but d'être plus proche de la réalité, que les descriptions peuvent être représentées comme une disjonction de conjonctions de propriétés : (P_{11} et P_{12} et …) ou (P_{21} et P_{22} et …)

ou … Chaque conjonction est une description possible du produit de conception. L'interprétation d'une conjonction consiste à appliquer un produit cartésien redéfini pour le cas spécifique de la logique floue : c'est la notion de t-norme (l'opérateur *min* en est un exemple) (cf. Annexe A). L'auteur précise toutefois qu'il serait intéressant d'étudier la possibilité d'utiliser une moyenne pondérée à la place d'une t-norme (cf. Annexe A). Par analogie, une disjonction de propriétés consiste à appliquer une union de n ensembles flous ou plus précisément une t-conorme (operateur *max* par exemple).

Enfin, un objet de conception est dit solution si toutes les propriétés (au sens de [Desmontils 96]) d'une des disjonctions de la description sont vérifiées. Pour cela l'auteur définit la notion de Seuil d'Acceptation (SdA) qui est une valeur de degré d'appartenance à partir de laquelle la propriété est considérée comme vérifiée. Il en donne une définition mathématique : le SdA est défini par une fonction S et une valeur m telle que :

$$
\begin{array}{lcll}
S_m : [0,1] & \mapsto & [0,1] & \\
\tau & \rightarrow & \tau & \text{si } \tau \geq m \\
 & & 0 & \text{sinon}
\end{array}
$$

Simplement, le SdA a la même valeur que la fonction d'appartenance sauf à partir d'un seuil particulier m où le SdA vaut 0. En logique floue, le seuil d'acceptation est défini par la notion de α–coupe avec $\alpha = m$ (cf. Annexe A). Il est précisé que le SdA est déterminé en fonction des opérateurs flous de la propriété de la description. Toutefois, l'auteur ne précise pas explicitement comment calculer ce SdA par rapport à une forme. Nous verrons dans le chapitre suivant comment nous proposons de réaliser cette opération d'estimation.

Par ailleurs, il serait possible non pas de vérifier (d'estimer) la forme obtenue a posteriori mais plutôt a priori en utilisant une technique de tirage aléatoire dans un intervalle et en considérant la fonction d'appartenance comme une fonction de probabilité. Mais cette approche n'est que très peu détaillée par l'auteur.

3.3 Modification des solutions / Intervention de l'opérateur

Nous nous plaçons dans le cas où la forme est obtenue quasi–automatiquement par le système informatique. Quelle que soit la méthode utilisée et les informations fournies par le concepteur, il est important d'admettre que même si le système est capable d'estimer une solution, la solution générée et considérée comme la plus prometteuse par le système informatique ne satisfera pas pleinement le concepteur dès la première itération. C'est ce que nous présentions précédemment comme la conception par tâtonnements. De ce fait, une prise de décision de la part du concepteur doit suivre chaque itération et/ou sélection automatique. Les intérêts de faire intervenir le concepteur sont :

- il est possible que la forme générée par le système et considérée comme une bonne solution soit totalement différente de ce que le concepteur attendait. Deux possibilités sont envisageables : soit la forme satisfait réellement le concepteur et dans ce cas, il ne lui reste plus qu'à raffiner les spécifications pour faire en sorte que le produit correspondant à la forme soit fabricable. Dans de tels cas, on peut dire que le système est innovant. Soit la forme générée ne correspond pas réellement aux désirs du concepteur ; ce qui signifie que le cahier des charges doit être redéfini ou complété. La prise de décision concerne la validation du cahier des charges fourni par le concepteur ;

- une intervention humaine peut confirmer ou infirmer la voie empruntée par le système pour la création des formes ; en ce sens, le concepteur peut élaguer considérablement l'espace des solutions considéré par le système.

L'intervention du concepteur peut se matérialiser de plusieurs manières :

- des règles expertes : c'est ce qui est le plus couramment utilisé car le moins difficile à mettre en œuvre. Ces règles ont pour objectif d'exprimer l'expérience acquise par le concepteur. C'est ce qui permet au système informatique d'élaguer l'espace des solutions pour qu'il propose plus rapidement les solutions les plus prometteuses. Toutefois, l'utilisation de règles entrave dans une certaine mesure la part de créativité que pourrait avoir un système dans le sens où il est « obligé » de suivre la démarche de construction du concepteur. Le tout est de construire un système à base de règles qui soit à la fois ni trop restrictif et ni trop évasif pour qu'il puisse proposer un nombre convenable de solutions (ni trop, ni trop peu). Un tel système est difficile à mettre en œuvre puisqu'il peut dépendre du problème en cours de résolution ;

- l'utilisation de paramètres modifiant le comportement du système : comme nous avons pu le voir pour les ensembles flous (cf. 3.2.4), le seuil d'acceptation est paramétré par une valeur en dessous de laquelle il vaut zéro. Cette valeur correspond donc à un paramètre du système sur lequel le concepteur peut agir. Dans ce cas précis, plus la valeur est proche de 1, plus le système proposera des solutions satisfaisantes mais avec le risque qu'il n'en propose aucune. D'autres paramètres peuvent être utilisés tels que l'importance d'une contrainte de conception par rapport aux autres contraintes du cahier des charges. Le problème réside dans la détermination de ces paramètres ; il faut distinguer ceux que le concepteur a le droit de modifier de ceux qui doivent être masqués par le système ;

- des outils de dialogue adaptés à des interactions entre l'homme et la machine : ces outils sont les plus délicats à mettre en œuvre. En effet, en supposant que le résultat de la génération et de la sélection automatique soit une forme affichée à l'écran, l'interaction homme–machine consisterait pour le concepteur à désigner les parties de la forme qui lui plaît. Une autre possibilité est de l'assister pendant les modifications apportées sur les parties de la forme qui ne le satisfont pas, c'est-à-dire de ne le laisser générer que des modifications améliorant globalement la forme. Ceci signifie que le système est capable à chaque instant d'interpréter et d'estimer la forme en cours de construction par le concepteur. Il est aussi possible que la nouvelle forme dessinée par le concepteur réponde à d'autres fonctions non précisées dans le cahier des charges qu'il faudrait alors mettre à jour. En outre, il est encore plus délicat de déterminer un outil de dialogue qui permette au concepteur de décrire les incohérences globales de la forme obtenue automatiquement. Par exemple, comment interpréter le fait que la forme soit trop disproportionnée, ou encore qu'elle soit trop allongée si ces notions ne sont pas compréhensibles par le système informatique ?

Après chaque génération et sélection des solutions les plus prometteuses, il est important de considérer le concepteur et le système informatique comme une équipe devant construire la forme du produit de conception. La collaboration ne peut donc se faire qu'en appliquant l'ensemble des interventions présentées.

4 Conclusion

Dans un premier temps, nous avons étudié les différentes méthodes d'assistance que le concepteur avait à sa disposition pour construire une forme. Nous avons brièvement présenté les modèles géométriques pour constater que les entités manipulées étaient de très bas niveau. D'autres méthodes plus avancées sont proposées dans la bibliographie mais ne sont pas encore toutes incorporées dans les systèmes de CFAO actuels. Toutefois, certaines, telles que les méthodes de construction à base de caractéristiques de formes, sont déjà implantées dans ces systèmes mais ne manipulent pas vraiment d'informations fonctionnelles comme le concepteur le désirerait. En réalité, ces entités répondent à différentes fonctionnalités mais c'est le concepteur, expert d'un domaine particulier, qui réalise une association entre les informations fonctionnelles et les entités géométriques (une rainure permet par exemple un guidage en translation). Nous avons donc étudié le fondement des méthodes fonctionnelles et nous en avons caractérisé de quatre types : les méthodes associatives qui relient directement des fonctions de bas niveau avec des caractéristiques de forme, les méthodes déductives qui proposent de générer plus intelligemment la forme du produit en gérant des opérateurs de décomposition de fonctions construisant ainsi simultanément la décomposition fonctionnelle et la topologie de l'objet à construire, les méthodes productives qui génèrent une forme plutôt qu'une topologie mais l'espace des solutions engendré est trop vaste, et enfin les méthodes déclaratives qui proposent de générer une forme de manière automatique. Toutefois, dans cette dernière approche, seul le principe de construction est décrit et aucune méthode valable n'est proposée pour construire automatiquement une forme tridimensionnelle satisfaisante. En fait, ces méthodes déclaratives connaissent des problèmes prédominants qui ne peuvent être résolus que pour des applications très particulières. Ces problèmes ont été présentés et concernent l'immensité de l'espace des solutions, le moyen d'obtenir les solutions les plus prometteuses en utilisant des techniques d'optimisation. Ces techniques nécessitent l'estimation de solutions et une prise décision de la part du concepteur. Plusieurs estimations sont proposées dans la littérature mais il nous semble que celle basée sur la logique floue est la plus intéressante puisqu'elle consiste à représenter chaque paramètre de conception sur un intervalle flou [Desmontils 96]. Enfin, comme une conception de produits ne se fait que par tâtonnements, il est important que le concepteur puisse modifier la forme obtenue de manière automatique ; pour cela nous avons présenté plusieurs possibilités : l'utilisation de règles traduisant les connaissances expertes du métier concerné par la conception, l'utilisation de paramètres modifiant le comportement du système informatique et la définition d'outils de dialogue assistant considérablement le concepteur lors de la construction de la forme.

Nous proposons dans le chapitre suivant d'utiliser ces différentes approches, en les combinant, pour pouvoir proposer au concepteur un ensemble satisfaisant de formes, élémentaires dans un premier temps, puis plus complexes. Nous supposerons que l'énoncé du problème de conception ne peut être exprimé d'une manière mathématique ; de ce fait, nous proposerons une méthode appropriée de parcours de l'espace des solutions qui permet d'obtenir un ensemble raisonnable représentant les solutions les plus prometteuses. Comme toutes les méthodes d'optimisation nécessitent de comparer deux solutions, une méthode d'estimation sera élaborée pour étudier l'adéquation des formes générées aux cahier des charges.

Chapitre 3 Vers une nouvelle méthode pour la synthèse automatique de formes

« Faîtes simple : aussi simple que possible, mais pas simpliste »
« L'imagination est plus forte que le savoir »
Albert Einstein

1 Introduction

Comme nous avons pu le remarquer dans les deux précédents chapitres, la génération de la forme d'un produit manufacturier répondant à un ensemble de fonctionnalités apparaît très complexe. Nous tentons, dans ce qui suit, de proposer un ensemble de formes en utilisant des intuitions. Celles-ci peuvent se matérialiser de différentes façons et en particulier les règles ou heuristiques en sont une première approche couramment utilisée. Dans un premier temps, nous esquissons une étude que nous avons menée pour générer des formes à partir d'informations fonctionnelles. Cette étude va mettre en évidence l'importance de considérer un modèle intermédiaire entre les fonctions et les formes pour simplifier leurs associations. Ceci nous permet alors de présenter, dans un deuxième temps, la notion de cahier des charges intermédiaire contenant des contraintes obtenues à partir des fonctions du produit. Ce cahier des charges nous permet de construire un espace de solutions contenant une vaste panoplie de solutions. Comme elle ne peuvent pas toutes êtres proposées au concepteur, il est nécessaire de les comparer ; nous proposons pour cela une méthode d'estimation. Cette dernière méthode permet enfin de choisir les meilleures solutions : celles qui vont être présentées au concepteur. Nous terminons ce chapitre par la présentation des différentes possibilités d'intervention liées à nos méthodes de synthèse et d'estimation de formes.

2 Génération de la forme

Dans ce paragraphe, nous discutons des différentes possibilités envisageables pour générer une forme à partir d'informations fonctionnelles. Cette génération consiste à construire aussi automatiquement que possible la géométrie d'un produit à partir d'entités conceptuelles telles que les fonctions. Nous classons les objets à concevoir en deux catégories : les objets isolés et assemblés. Les premiers sont des objets

relativement simples composés d'une seule pièce : un vase, une gomme, un porte-crayon … Par contre, les objets assemblés sont définis comme un assemblage d'objets isolés : une voiture, une bouteille, un téléphone, un ordinateur … Avant de construire la géométrie d'un objet assemblé, il est nécessaire de construire la géométrie de ces objets isolés en considérant par exemple les contraintes d'assemblage. C'est pourquoi nous ne considérons dans la suite que la génération de la géométrie d'objets isolés.

Nous envisageons trois possibilités : associer des surfaces fonctionnelles entre elles pour construire un solide correspondant à l'objet isolé, déformer un solide primitif pour qu'il satisfasse les fonctions demandées, et enfin, combiner des objets primitifs qui satisfont chacun une fonction. Une étude prospective est présentée pour chacune des approches.

2.1 Association de surfaces fonctionnelles

La première approche consiste à associer des surfaces entre elles, à condition que chaque surface satisfasse au moins une fonction. Pour cela, on suppose que le cahier des charges exprimé par le client aura été transcrit en informations compréhensibles par le système. On suppose, par ailleurs, qu'une bibliothèque de fonctions auxquelles sont associées des surfaces est disponible. Les informations compréhensibles par le système correspondent aux fonctions contenues dans la bibliothèque. Il est certain que la bibliothèque devra être définie par un expert de chaque métier concerné par la conception du produit en cours d'étude. Par exemple, pour la conception d'une bouteille, la stabilité pourra être représentée par un ensemble de surfaces alors que le transport aisé par l'être humain de cette bouteille sera défini par un autre ensemble de surfaces. L'ensemble de surfaces associées à une fonction peut être assemblé pour former une caractéristique de forme. Chaque caractéristique de forme peut satisfaire une ou plusieurs fonctions. Par conséquent, la géométrie du produit sera définie comme un raccordement de caractéristiques de forme. La grande difficulté est de savoir comment raccorder ces surfaces même si l'on sait quoi raccorder.

La différence de cette approche avec celles de [Constant 96, Mukherjee et al. 95] se situe au niveau de l'automatisation. Pour ces auteurs, c'est au concepteur de décrire les assemblages de surfaces fonctionnelles. À partir du moment où une bibliothèque d'associations est connue, il est possible de tenter toutes les combinaisons possibles d'assemblages de surfaces. Ensuite, il suffit, en quelque sorte, de choisir les meilleures solutions. Cela nécessite une fonction qui permet de calculer avec quel degré la solution satisfait les fonctions.

Une automatisation peut simplement se réaliser à l'aide d'algorithmes stochastiques comme les algorithmes génétiques [Rosenman 96]. Au lieu de considérer uniquement des éléments de surfaces unitaires (carreau de surface égale à 1), on considère plusieurs éléments de bases que sont les caractéristiques de formes. Le génotype d'un individu (cf. Chapitre 2 §2.2.3.4) peut être considéré comme un solide primitif (une pièce brute comme une sphère, un cylindre, une boîte …) et une liste de caractéristiques de forme à appliquer au solide primitif. Pour obtenir le phénotype de l'individu, il est nécessaire de conserver les positions et les paramètres de chaque caractéristique (cf. Figure 3-1). Comme les paramètres d'une caractéristique dépendent de la caractéristique, le croisement de deux individus parents ne peut se faire entre la caractéristique et les paramètres de cette caractéristique de forme à moins que les caractéristiques utilisées dans le génotype des deux parents soient les mêmes. La mutation peut consister à modifier aléatoirement soit les valeurs de certains paramètres soit la position de caractéristiques

ou encore de remplacer une caractéristique de forme par une autre.

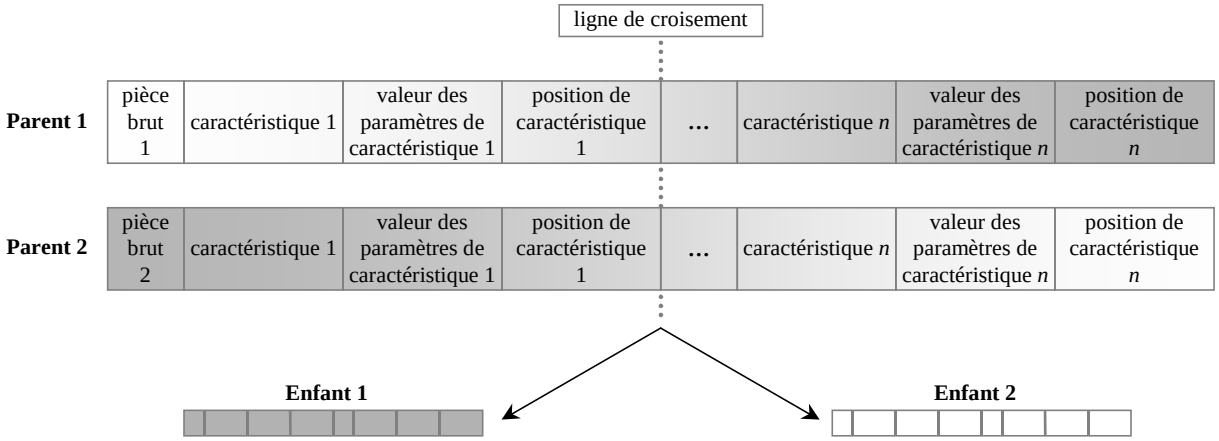


Figure 3-1. *Croisement de génotypes représentant une forme*

Ce principe permet d'automatiser l'évolution des individus d'une population initiale. Il faut néanmoins construire cette population. Pour cela, il est possible d'utiliser la notion de catégorie d'objets : chaque catégorie satisfait un ensemble de fonctions ; et à une catégorie est associée une morphologie, c'est-à-dire un ensemble d'entités géométriques. Par exemple, on peut construire la catégorie « récipient » qui doit satisfaire les fonctions « remplissable », « vidable », « maniement aisé pour l'être humain », « contenir une fluide » … Pour chacune de ces fonctions, on définit une ou plusieurs géométries représentées par les caractéristiques de formes. De cette façon, la construction de la population initiale peut consister à représenter toutes les combinaisons possibles d'assemblages de caractéristique de forme. Il reste à positionner les caractéristiques et à valuer leurs paramètres : ceci peut être fait de manière aléatoire.

Lors d'une conception d'un nouveau produit, le principe consiste à réutiliser autant que possible les catégories d'objets déjà définies en dérivant les propriétés communes comme le préconise la conception de produits logiciels avec les patrons de conception [Gamma 95]. Cette construction de la population initiale n'est pas automatique mais représente une assistance certaine.

À terme, l'objectif est de pouvoir construire des objets aussi innovants que possible à partir d'objets connus déjà réalisés en assemblant les meilleures parties de chaque objet pour construire un objet assemblés [Gardan et al. 97] : par exemple, un vélo comportant un guidon de triathlète et un cadre classique est croisé avec un vélo comportant un guidon classique et un cadre dont la flexibilité assure la suspension (cf. Figure 3-2). Le croisement permet d'obtenir un vélo combinant les deux avantages et un autre sans avantage particulier.

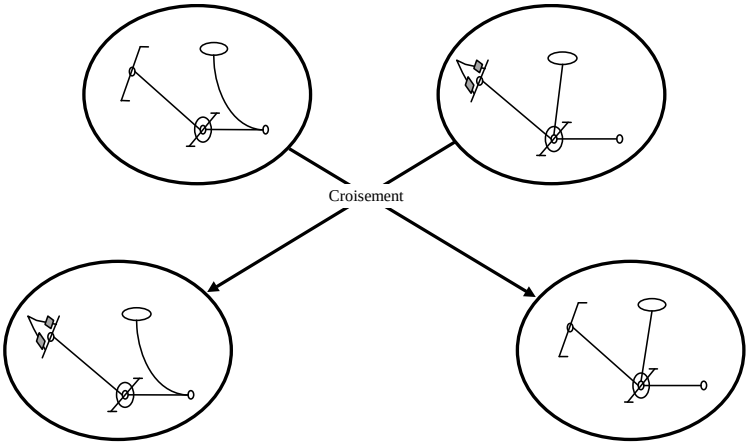


Figure 3-2. *Utilisation d'algorithmes génétiques [Gardan et al. 97]*

2.2 Déformation de solides

Cette approche consiste à déformer un objet de base connu par le système de telle façon que la déformation effectuée améliore la satisfaction des contraintes. Le seul moyen possible pour construire le plus automatiquement possible une forme à partir d'un objet primitif est de considérer au moins une règle experte par fonction. De ce fait, il y aurait au moins autant de règles que de fonctions. Pour illustrer cette démarche, on peut donner l'exemple suivant : à supposer que l'objet primitif est un cube et qu'une des fonctions à satisfaire est « rouler », on peut supposer qu'une règle métier associée à cette fonction soit d'arrondir les coins du cube. On peut par ailleurs imaginer une règle plus générale qui ne dépende pas de la forme à déformer.

Comme le produit à concevoir est défini par plusieurs fonctions, il suffirait d'appliquer plusieurs déformations correspondant aux diverses fonctions. Toutefois, cette approche possède un inconvénient majeur : il est possible que les modifications proposées par une règle associée à une fonction aggrave la satisfaction d'une autre fonction. Par conséquent, appliquer une succession de déformations pour améliorer l'ensemble des fonctions n'est pas forcément très judicieux. Pour améliorer cette approche, il faudrait détecter les incohérences ou les contradictions entre modifications avant d'appliquer les déformations. Pour résoudre cela, une quelconque méthode d'optimisation devrait être nécessaire pour déterminer l'ordre des déformations à appliquer sur la forme primitive.

2.3 Combinaison de solides primitifs

La première approche consiste à combiner des surfaces fonctionnelles (une surface satisfait une ou plusieurs fonctions) alors que la deuxième approche propose de déformer des objets primitifs ; cette dernière perspective s'approprie le principe des deux premières approches dans le sens où elle a pour objectif de combiner des solides primitifs tels que la sphère, la boîte, la pyramide …

Lorsque la notion de combinaison de formes est mise en avant, la première idée de modèle informatique qui vient à l'esprit est la notion d'arbre CSG. Si on conserve cette première idée, il faut donc définir un moyen de traduire les fonctions du produit en opérations booléennes ($\cup$, $\cap$, —) sur des objets. De ce fait, il faut à la fois déterminer la structure de l'arbre (les opérations et les primitives utilisées) et à la

fois valuer les différents paramètres de cet arbre : les valeurs des paramètres des primitives ainsi que la position et l'orientation de ces dernières. Encore une fois, une éventuelle automatisation ne peut se faire qu'avec l'aide d'heuristiques définies par un expert du domaine de conception. Pour la construction de l'arbre, on peut distinguer deux types de règles qui associent les fonctions aux entités utilisées par l'arbre CSG :

- des règles qui listent les propriétés fonctionnelles des opérations booléennes pour chaque fonction utilisée dans le cahier des charges du client. Par exemple, si une des fonctions utilise un paramètre tel que le volume, une des propriétés serait de préciser que le volume augmente inévitablement lorsqu'on applique une union et à tendance à diminuer lorsqu'on souhaite appliquer une différence ;
- des règles qui listent les propriétés fonctionnelles d'opérations booléennes appliquées à des objet précis : ceci correspond à lister les propriétés de sous-arbre CSG. Par exemple, un guidage en translation peut s'effectuer simplement en ajoutant une rainure ; on associe donc la fonction « guidage en translation » au sous-arbre « Union–Rainure ».

2.4 Comparaison des approches

On peut considérer que la première approche qui consiste à construire un solide à partir de surfaces peut servir d'étape préliminaire aux deux autres approches qui utilisent des solides comme entités de bases. Par ailleurs, si on utilisait la notion de déformation pour modifier localement une forme déjà considérée comme bonne solution, on pourrait appliquer alternativement les trois approches de la façon suivante :

- construction d'un solide en combinant des surfaces fonctionnelles ;
- possibilité de combiner plusieurs solides solutions créés auparavant ;
- application de modifications locales à l'aide de déformation sur les solides obtenus précédemment.

Quelle que soit l'approche considérée, il est nécessaire d'associer les fonctions, informations trop conceptuelles pour être interprétées de manière automatique par un système informatique, aux entités du modèle considéré : caractéristiques de forme, objets primitifs, opérations booléennes. En réalité, cette association est inévitable dans la mesure où il est difficile de déterminer une déduction logique entre les fonctions et les formes. Toutefois, l'association reste complexe dans la mesure où il faut faire correspondre des entités très conceptuelles à des entités plus physiques telle que la géométrie. Or, comme le propose [Mukherjee et al. 95], l'utilisation d'un modèle intermédiaire est plus judicieux puisqu'il permet de faire des associations moins directes entre les fonctions et les formes ; surtout que l'expérience montre que des fonctions peuvent relativement facilement être « objectivées » : elles sont ramenées à des contraintes sur des entités objectives. Intuitivement, il est possible de traduire la beauté d'une voiture un ensemble de propriétés telles que le rapport entre le largeur et la longueur, le coefficient de pénétration dans l'air …

Dans ce qui suit, nous présentons une méthode simplifiée qui utilise un modèle intermédiaire représenté par un cahier des charges intermédiaire. L'approche consiste à générer des formes satisfaisantes dont la morphologie est connue. Le cahier des charges intermédiaire permet de valuer les paramètres de ces formes. Une méthode d'estimation permet ensuite de calculer les meilleures solutions.

3 Approche proposée et définitions

Dans cette partie, nous détaillons l'approche décrite ci-dessus qui consiste à construire un espace de solutions à partir de formes primitives. Pour cela, nous considérons que les seules informations dont nous disposons sont décrites dans le langage naturel et forment le cahier des charges du client (cf. Figure 3-3a). À l'heure actuelle, ces informations ne peuvent être traduites que manuellement en une décomposition fonctionnelle (cf. Figure 3-3b) en utilisant par exemple les fonctions de réalisation, de maintien, d'interdiction, de contrôle [Keuneke 91] et de permission [MacDowell et al. 96]. Certes, il est possible d'assister le concepteur dans cette démarche en utilisant des outils appropriés. Par exemple, l'analyse de la valeur peut aider le concepteur à identifier les fonctions du produit et des logiciels comme [Tdc] ou FBS [Ranta et al. 96] peuvent proposer des formalismes facilitant l'enregistrement des informations obtenues. Par ailleurs, des outils informatiques tels que les analyseurs, les solveurs et les superviseurs [Gardan et al. 97] ou des principes de décomposition de fonctions [Shimomura et al. 96] permettent d'enrichir le modèle fonctionnel soit en introduisant de nouvelles fonctions, soit en faisant apparaître petit à petit les composants du produit. Cela fait surgir la notion de mécanismes (cf. Figure 3-3c) en raison des liens qui existent entre les différents composants du produit. Il est évident, qu'à ce stade de la décomposition fonctionnelle, le concepteur n'a pas encore d'idée très précise de la géométrie du produit. Du fait qu'un composant soit lié avec d'autres composants, il en résulte un certain nombre de contraintes propres au composant considéré. L'ensemble de ces contraintes forment le *cahier des charges intermédiaire* (cf. Figure 3-3d). Nous considérons ces spécifications comme des informations nécessaires pour générer automatiquement la forme du composant. À partir de ces informations, interprétables par un système informatique, l'espace des formes solutions est généré en utilisant des formes primitives contenues dans une bibliothèque (cf. Figure 3-3e). Ce traitement correspond à trouver les valeurs des paramètres des formes de la bibliothèque qui satisfont le cahier des charges intermédiaire. Il est possible que cet espace soit vaste ; de ce fait, il est inconcevable de présenter l'ensemble des solutions au concepteur. La définition et la spécification de l'estimation d'une solution par rapport au cahier des charges intermédiaire permet d'utiliser une méthode d'optimisation qui trouvera automatiquement les solutions les plus prometteuses (cf. Figure 3-3f).

Le problème réside dans l'expression du cahier des charges intermédiaire de telle sorte qu'il puisse être interprété de manière informatique pour générer un ensemble d'informations géométriques qui serviront à construire la forme du produit considéré (cf. 3.1). Nous introduisons pour cela la notion de paramètres intermédiaires dans le paragraphe suivant (cf. 3.2). Ces paramètres vont nous permettre d'exprimer des contraintes sur le composant pour lequel on souhaite obtenir le plus automatiquement possible une forme. Une fois le cahier des charges défini, il faut ensuite construire l'espace des solutions (cf. 3.4).

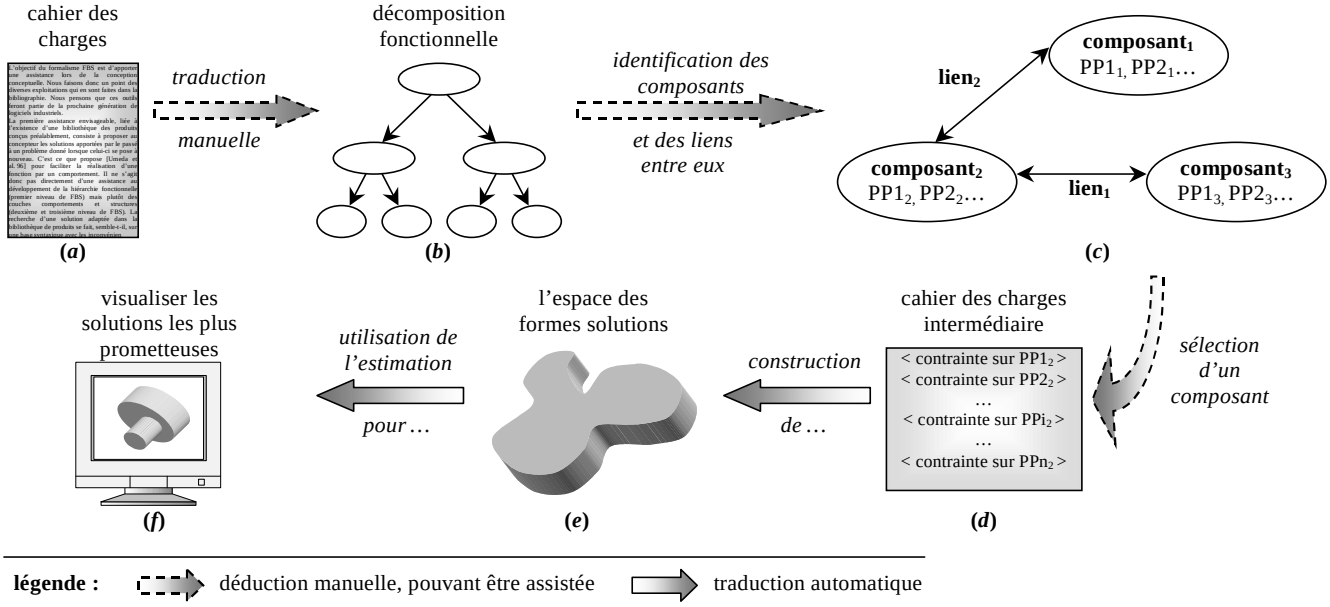


Figure 3-3. *Principe général de la génération de formes [Gardan et al. 99b]*

3.1 Le cahier des charges intermédiaire

À l'heure actuelle, il apparaît difficile d'appliquer des traitements automatiques sur les informations fonctionnelles fournies par le client dans la mesure où elles sont souvent exprimées dans le langage naturel. Toutefois, quelques tentatives visent à représenter les fonctions d'un produit manufacturier sous la forme d'entités modifiant des données d'entrée en données de sortie (fonction de transformation, cf. Chapitre 1 §4.3). Cependant, ces approches ignorent certains types de fonctions telles que l'esthétisme … Dans le but de prendre en considération des fonctions de natures différentes, il est important de définir un formalisme permettant une automatisation. Ce formalisme est appelé *cahier des charges intermédiaire* dans le sens où il est construit d'après le cahier des charges du client et dans la mesure où il sert de point de départ pour la génération de la géométrie du produit. On peut considérer le cahier des charges intermédiaire comme une autre modélisation du cahier des charges du client. De plus, nous verrons que ce cahier des charges sera représenté par des courbes : nous pouvons donc parler de multi-modélisation. Le choix d'un des modèles dépendra des traitements à appliquer sur le cahier des charges.

Intuitivement, les fonctions du produit peuvent être traduites en contraintes sur les entités composant le produit (cf. Figure 3-4). C'est l'approche utilisée par la méthode de la maison de la qualité (cf. Chapitre 2 §3.2.2) qui préconise aux concepteurs de reformuler les besoins du client (exprimés en langage naturel) en spécifications d'ingénierie basées sur des paramètres. Ces paramètres doivent pouvoir être valués. Ce qui signifie, d'une certaine façon, que des notions relativement subjectives (les fonctions) sont transcrites en informations beaucoup plus objectives (les paramètres). Cette intuition ne peut se montrer qu'à l'aide d'un exemple. Pour cela, considérons la conception d'une bouteille d'eau. Nous nous plaçons dans le cas d'une nouvelle conception sans considérer les éventuelles conceptions précédentes. De ce fait, la conception de la bouteille ne doit pas être vue comme une conception routinière dans la mesure où on suppose qu'aucune bouteille d'eau n'a déjà fait l'objet d'une réalisation. En réalité, on suppose que le concepteur n'a aucune idée de la forme du produit ; il ne connaît que ses spécifications.

Voici une liste non–exhaustive des fonctions que doit satisfaire le produit :

- contenir une certaine quantité de liquide (1l ; 1,25l ; 1,5l …) donc respecter une certaine étanchéité pour un liquide précis ;
- pouvoir tenir dans la main d'un être humain ;
- être compressible pour utiliser un minimum de place une fois l'utilisation terminée (recyclage) ;
- être léger ;
- être stable ;
- être beau ;
- être résistant à des chocs d'une certaine puissance …

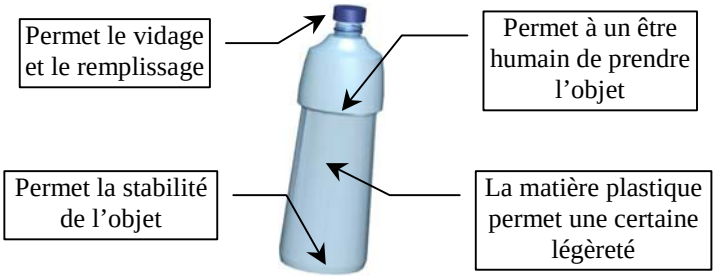


Figure 3-4. *Une solution de conception pour la bouteille d'eau*

Une étude approfondie de chacune de ces fonctions aboutit au cahier des charges intermédiaire pouvant s'exprimer de la façon suivante :

- la masse totale de l'objet ne doit pas dépasser 1,6 kg pour une contenance de 1,5 litres ;
- le volume doit être inférieur à 1,5l pour satisfaire la possibilité à l'être humain de le manipuler d'une seule main sans anse ;
- certaines proportions dimensionnelles doivent être respectées, c'est-à-dire que la hauteur est proportionnelle à la largeur, elle même proportionnelle à la profondeur. Ainsi, il est possible de déterminer un volume englobant le produit. De ce fait, on peut obtenir les contraintes suivantes : la hauteur est inférieure à 35cm, la largeur est supérieure à 5cm et inférieure à la moitié de la hauteur …

On peut donc en déduire un certain nombre de contraintes sur les paramètres d'ingénierie du produit que nous appelons paramètres intermédiaires (cf. 3.2). Comme le concepteur n'a pas d'idées précises de la géométrie du produit, nous supposons que l'ensemble des contraintes sur les paramètres va permettre, au fur et à mesure du raffinement du cahier des charges intermédiaire et de l'identification de nouveaux paramètres, de restreindre l'espace des solutions. Il est certain que tous les paramètres intermédiaires ne serviront pas à restreindre la géométrie de l'objet mais un bon nombre d'entre eux, du fait qu'ils sont liés à la géométrie, permettront de faire des choix. Par exemple, certains paramètres peuvent se référer au matériau de l'objet ; et le choix du matériau peut agir sur la géométrie du produit dans le sens où une distance minimale ou maximale est requise pour l'utilisation de ce matériau … Néanmoins, nous supposons que le nombre de paramètres et de contraintes suffiront à caractériser un ensemble de formes qu'il faut déterminer le plus automatiquement possible.

Par ailleurs, remarquons que les fonctions du produit n'ont pas toutes la même importance : on ne portera peut être pas le même intérêt pour la compressibilité ou le recyclage que pour l'esthétisme. Ou encore, il faut absolument que la bouteille soit étanche pour contenir un liquide avant d'être esthétique. Il faut donc distinguer les fonctions principales des fonctions auxiliaires. C'est au concepteur de classer chacune des fonctions du cahier des charges suivant leur importance pour le client. Toutefois, une gestion directe des fonctions du produit apparaît très complexe. Or, nous savons que les fonctions du produit se transcrivent en contrainte sur des paramètres. De ce fait, le poids d'une fonction doit être répercuté sur les contraintes. Ainsi, chaque contrainte sera affectée d'un poids. Ce poids peut être représenté qualitativement comme le fait [Desmontils 96]. Ceci permet, par exemple, d'utiliser des termes du langage naturel tels que « important », « très important », « peu important » pour qualifier l'importance des fonctions. Comme nous l'avons vu (Chapitre 2, §3.2.4), à chaque terme du langage est associé une courbe de satisfaction. Mais, nous préférons représenter le poids d'une contrainte à l'aide d'un nombre réel. Ce nombre, appelé *poids* permet d'obtenir directement la ou les fonctions principales du produit. Il permettra par la suite d'opérer des calculs qui serviront à générer une forme adaptée aux spécifications, c'est-à-dire de savoir avec quel degré une forme satisfait les contraintes ou indirectement les fonctions. Rien ne s'oppose à ce que, par la suite, une représentation qualitative soit utilisée.

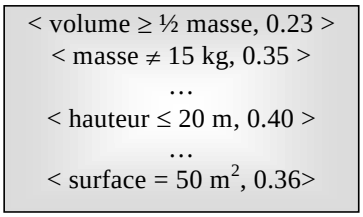


Figure 3-5. *Représentation du cahier des charges intermédiaire*

Le cahier des charges intermédiaire représente le cahier des charges du client exprimé dans un format compréhensible par un système informatique. Nous le définissons comme un ensemble de contraintes (cf. Figure 3-5). Chaque contrainte est exprimée formellement de la façon suivante : <*PI*, *Relation*, *Expression*, *Poids*> où :

- *PI* est un paramètre intermédiaire correspondant à un paramètre de haut niveau tels que la masse, le volume, le coefficient de pénétration dans l'air, la compressibilité du matériau … Nous expliquons la notion de paramètre intermédiaire dans le paragraphe suivant (cf. 3.2) ;
- *Relation* est une relation mathématique contraignant les valeurs pouvant être prises par le paramètre. Nous gérons les relations suivantes : supérieure ou égal (≥), inférieure ou égal (≤), égal (=) et différent (≠). Nous expliquerons ultérieurement pourquoi les relations supérieure et inférieure ne sont pas gérées même si elles peuvent l'être ;
- *Expression* est une expression arithmétique pouvant contenir d'autres paramètres intermédiaires. Par exemple, une contrainte peut définir le fait que le volume doit être supérieur ou égal à la moitié de la masse (volume ≥ ½ masse) ;
- *Poids* est un nombre réel qui représente l'importance de la contrainte.

Comme d'importants progrès ont été réalisés ces dernières années dans le domaine de la résolution de problèmes contraints, nous espérons utiliser cette avancée pour générer un ensemble de formes solutions satisfaisant ces contraintes plus facilement qu'avec une autre méthode. La différence entre notre approche et les approches étudiées pour la résolution de contraintes se situe au niveau des informations manipulées : elles sont d'un niveau sémantique plus élevé que des contraintes purement géométriques telles que le parallélisme ou la perpendicularité, ce qui accroît la complexité du problème à traiter. Les algorithmes de résolution de problèmes contraints connaissent encore quelques difficultés lorsque les contraintes ne sont pas linéaires, c'est-à-dire lorsque le problème ne s'exprime pas comme un système d'équations linéaires.

3.2 La notion de paramètres intermédiaires

Dans le précédent paragraphe, nous avons précisé que le cahier des charges intermédiaire utilisait des paramètres pour contraindre le produit à concevoir. Ces paramètres sont nommés *intermédiaires* car ils interviennent dans le cahier des charges intermédiaire. Nous avons donné comme exemple le volume, les dimensions, la masse. On peut qualifier ces paramètres comme des paramètres physiques dans le sens où ils font référence à des propriétés physiques du produit. Toutefois, d'autres paramètres peuvent être utilisés pour spécifier le produit en cours de conception. En l'occurrence, le coût de revient du produit est souvent un moyen pour les concepteurs de réduire l'espace des solutions. Nous considérons que le coût est aussi un paramètre intermédiaire mais de nature différente des précédents. Par ailleurs, certain produits ne sont construits que pour une certaine catégorie de personnes (soit pour une tranche d'âges bien précise, soit pour des personnes démunies …). De ce fait, il apparaît que plusieurs types de paramètres existent : ils sont classés en *familles*. Dans cet état d'esprit, on peut imaginer plusieurs familles telles que : physique, économique, esthétique, sociale … (cf. Figure 3-6a).

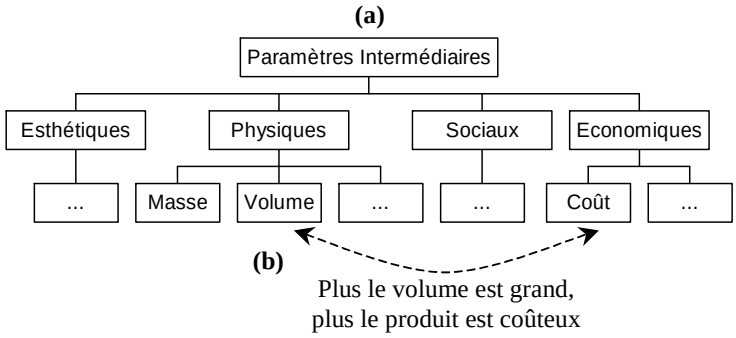


Figure 3-6. *Hiérarchie des familles de paramètres intermédiaires (a) et relations dans cette hiérarchie (b)*

Des relations entre paramètres de familles différentes peuvent exister et montrer les dépendances entre plusieurs domaines (cf. Figure 3-6b). Par exemple, plus le produit est volumineux, plus il sera coûteux ; cette relation fait intervenir le paramètre « volume » de la famille « Physique » et le paramètre « coût » de la famille « Économique ». Remarquons que ce genre de relations ressemble fortement aux relations utilisées dans QPT (cf. Chapitre 1, §2.3.2).

Dans la suite, nous nous intéressons seulement aux paramètres physiques. Il sont définis comme des entités quantifiables se référant au monde physique. Au sein de cette famille, il est possible de distinguer deux sous-familles de paramètres : les paramètres de bases et les paramètres de plus haut niveau définis

comme une combinaison mathématique des premiers. L'avantage de cette distinction réside dans le nombre de paramètres de bas niveau : il en existe sept en suivant le Système International :

- la longueur exprimée en mètre ;
- la masse exprimée en kilogramme ;
- la température exprimée en kelvin ;
- le temps exprimé en seconde ;
- la quantité électrique exprimée en ampère ;
- la quantité de matière exprimée en mole ;
- la quantité de lumière, exprimée en candela.

De ce fait, une force, considérée comme un paramètre de haut niveau dont l'unité est le newton, peut s'exprimer suivant les sept paramètres de base comme suit : $Newton = \frac{masse \times longueur}{temps^2}$.

3.3 Hypothèses de travail

Les produits manufacturiers peuvent être distingués en deux catégories :

- les produits assemblés comportant plusieurs pièces. Ces pièces sont assemblées selon des mécanismes. Par exemple, nous considérons qu'un engrenage, un carter de voiture, une télévision sont des exemples de produits assemblés. La génération automatique de la géométrie d'un tel produit est complexe dans la mesure où il faut prendre en considération les contraintes propres à chaque pièce du produit ainsi que les contraintes liant les pièces entre elles ;
- les produits isolés ne comportant qu'une seule pièce. Par exemple, on peut considérer qu'un porte-crayon, un vase, une gomme sont des produits isolés.

Avant de s'intéresser à la synthèse de produits assemblés, il est nécessaire de résoudre la synthèse de produits isolés. C'est pourquoi, nous nous intéressons, dans la suite, à la deuxième catégorie de produits. Les produits assemblés pourront être générés avec la même méthode en générant itérativement la forme de chaque pièce intervenant dans l'assemblage de ce produit.

Par ailleurs, si on suppose que le problème du passage des contraintes du cahier des charges intermédiaire à une forme peut être résolu de manière purement mathématique, les solutions de conception devraient être représentées par des expressions mathématiques qui devraient contenir un ensemble de morphologies de forme. Or, il est difficile d'envisager une expression mathématique à partir de laquelle on peut extraire plusieurs morphologies différentes. De ce fait, nous considérons que le problème du passage d'un ensemble de contraintes sur des paramètres de haut niveau à des formes n'est pas un problème purement mathématique mais qu'il nécessite la description d'un algorithme approprié.

De plus, nous avons déjà précisé que le cahier des charges était au début de la conception considéré comme sous-contraint. Ce n'est qu'au fur et à mesure qu'il devient de plus en plus contraint pour n'obtenir qu'une seule solution de conception. Ainsi, si le problème de conception n'est étudié que d'un point de

vue mathématique, il sera difficile pour le système informatique de proposer des solutions puisque le problème sera sous-contraint.

3.4 Création de l'espace des solutions

Dans ce paragraphe, nous présentons une méthode qui permet de construire l'espace des formes solutions satisfaisant les contraintes intermédiaires. Pour cela, nous définissons la notion de classe de forme. Ensuite, nous présentons certains problèmes liés au passage des contraintes intermédiaires aux classes de forme. Une fois qu'une solution est proposée pour pallier ces différents problèmes, nous formalisons cette traduction.

3.4.1 Les classes de forme

Comme nous avons pu le voir dans le paragraphe 3, notre approche consiste à construire des formes solutions à partir des contraintes sur les paramètres intermédiaires et des formes de bases contenues dans une bibliothèque (cf. Figure 3-7). Chaque forme est définie à l'aide de paramètres purement géométriques tels que le rayon, les longueurs des cotés, la hauteur, des angles … Pour distinguer ces paramètres des paramètres utilisés dans le cahier des charges, ils sont nommés paramètres *terminaux* dans la mesure où ils interviennent directement sur la géométrie de l'objet.

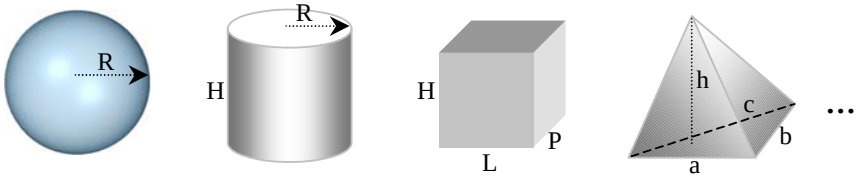


Figure 3-7. *Différentes formes de base avec leurs paramètres terminaux associés*

Par ailleurs, deux ensembles différents de paramètres terminaux peuvent définir la même forme : par exemple, un tétraèdre peut être caractérisé soit par la longueur de chaque coté de la base et la hauteur (cf. Figure 3-8a), soit par deux angles, une longueur et la hauteur (cf. Figure 3-8b).

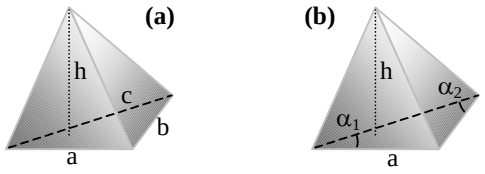


Figure 3-8. *Deux manières différentes de représenter un tétraèdre : avec des dimensions (a) ou avec des angles (b)*

L'ensemble des paramètres définissant une forme est appelé *classe* de forme. Une classe peut aussi avoir des méthodes permettant de passer d'un ensemble de paramètres à un autre, définissant la même forme. De cette façon, une classe devrait être capable de passer de trois longueurs et une hauteur à une longueur, une hauteur et deux angles (cas du tétraèdre).

L'espace des solutions satisfaisant les contraintes du cahier des charges intermédiaire sera représenté par un ensemble de classes de forme. Le choix d'une valeur réelle pour chaque paramètre terminal d'une

classe correspond au plongement de cette classe dans l'espace réel de dimension *n* – *n* étant le nombre de paramètres terminaux de la classe (cf. Figure 3-9).

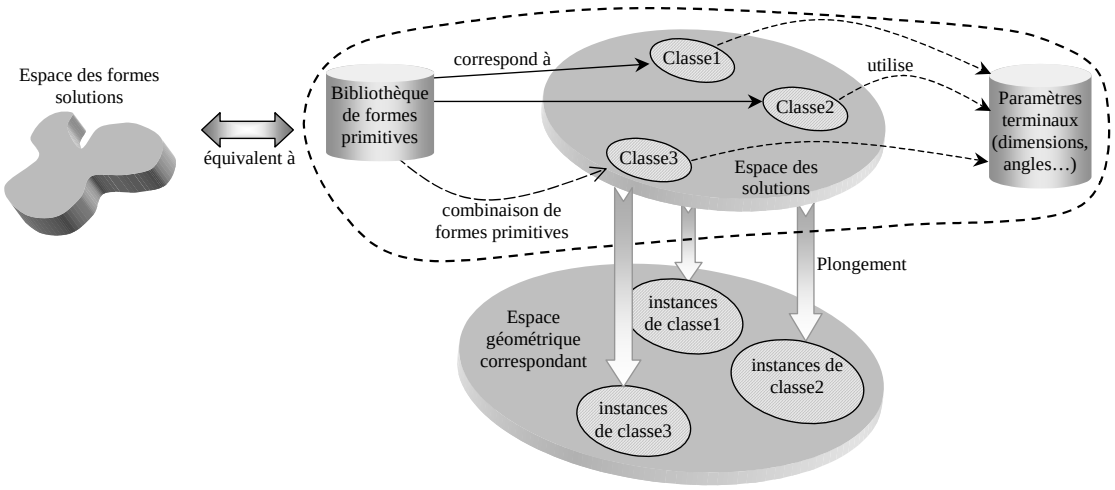


Figure 3-9. *Représentation de l'espace des solutions contenant les classes de forme [Gardan et al. 99b]*

3.4.2 Intervalles de variation des paramètres terminaux pour chaque classe

L'intérêt de s'appuyer sur une bibliothèque de formes prédéfinies est d'éviter de nous soucier de la morphologie de la forme à générer. Seul le paramétrage nous importe. Il suffit, en réalité, de calculer pour chaque forme les valeurs de ses paramètres terminaux qui satisfont le cahier des charges intermédiaire. Ainsi, le principe est de traduire les contraintes définies dans le cahier des charges intermédiaire (les contraintes *intermédiaires*) en contraintes sur les paramètres terminaux (contraintes *terminales*). Nous obtenons donc un nouveau cahier des charges où les contraintes sont appliquées sur les paramètres terminaux de chacune des formes de la bibliothèque. De cette façon, il y aura autant de cahiers des charges terminaux qu'il y aura de formes prédéfinies (cf. Figure 3-10a). Pour réaliser cette étape, les paramètres intermédiaires doivent être exprimés en fonction des paramètres terminaux. Ceci peut être réalisé soit mathématiquement soit de façon procédurale. En effet, si une partie des contraintes du cahier des charges intermédiaires est définie par « volume ≥ 5 » et « volume ≤ 10 » et si la forme primitive sphère est considérée, alors on peut obtenir mathématiquement les contraintes terminales suivantes : « $R \geq 5/(2\pi)$ et « $R \leq 5/\pi$ ». Par contre, la traduction d'une contrainte intermédiaire plus complexe telle que la compressibilité de la bouteille ne peut se faire qu'avec l'aide d'une procédure devant être définie par le concepteur.

La traduction des contraintes intermédiaires en contraintes terminales est difficile parce que, dans la plupart des cas, un paramètre intermédiaire est fonction de plusieurs paramètres terminaux. Au cours des premières conceptions, ces algorithmes ou ces formules mathématiques doivent être spécifiés par le concepteur. Toutefois, au fur et à mesure des conceptions, la connaissance du système augmentera et il sera plus facile de réutiliser des formules existantes ou de donner la possibilité au concepteur de combiner ces formules pour en construire d'autres plus complexes. Par exemple, supposons que le paramètre intermédiaire « volume » soit utilisé dans le cahier des charges intermédiaire et que le système informatique considère la classe de forme « sphère ». Il existe trois possibilités :

– le système sait déjà traduire la contrainte sur le volume en contraintes sur le rayon de la sphère. Dans

ce cas, aucune action particulière n'est nécessaire ;

- le système sait seulement traduire une contrainte sur le paramètre intermédiaire « surface » en contrainte sur le rayon de la sphère : dans ce cas, il suffit au concepteur d'exprimer le lien mathématique qui existe entre le volume et la surface d'une sphère ;
- aucune formule faisant intervenir la sphère n'est connue de la part du système ; dans ce cas, le concepteur est obligé de définir la traduction d'une telle contrainte intermédiaire en contraintes sur le rayon de la sphère.

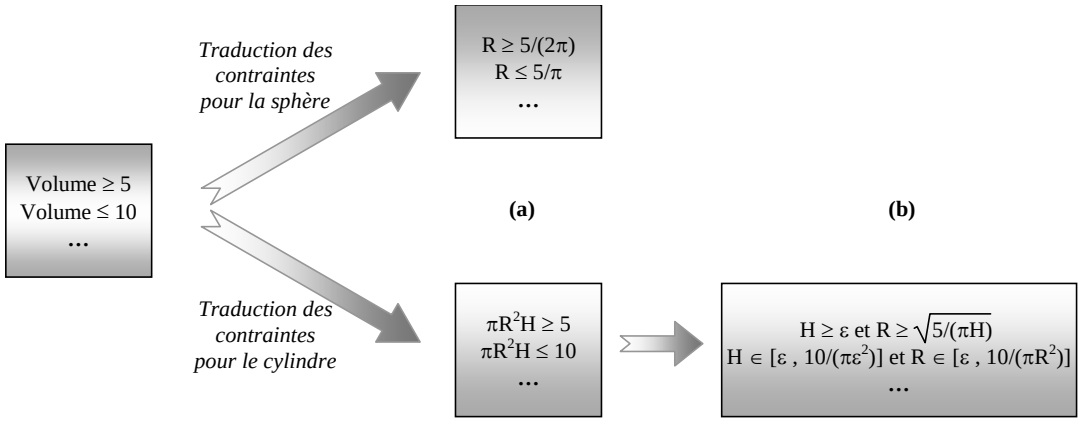


Figure 3-10. *Traduction des contraintes intermédiaires en contraintes terminales(a), puis en intervalles sur les paramètres terminaux(b)*

À partir des contraintes terminales, un ensemble d'intervalles peut être défini pour chaque paramètre terminal de la forme primitive considérée (cf. Figure 3-10b). Il est toutefois difficile à partir d'une seule contrainte intermédiaire (donc faisant référence à un seul paramètre intermédiaire) de déterminer les intervalles pour chaque paramètre terminaux de la forme primitive, surtout que ces derniers interviennent tous pour le calcul du paramètre intermédiaire. Pour illustrer le problème, considérons un cas en deux dimensions : supposons que la contrainte intermédiaire soit « Périmètre = 10 » ; considérons par ailleurs que la forme primitive utilisée soit le rectangle. Supposons ensuite que la largeur du rectangle soit égale à x (x appartenant à l'intervalle [0,5]), la hauteur doit être égale à $5–x$ pour vérifier la contrainte intermédiaire. Or, il existe une autre solution qui est $5–x$ pour la largeur et x pour la hauteur (cf. Figure 3-11).

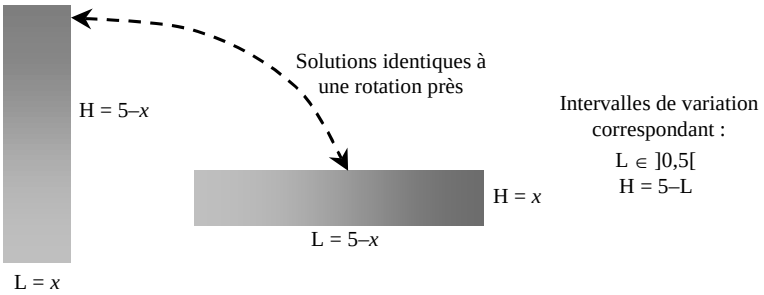


Figure 3-11. *Solutions identiques dans l'espace des solutions*

Dans ce qui suit, nous considérons que les deux solutions sont identiques (à une rotation près). Il est possible, dans certains cas, que le concepteur souhaite les considérer comme deux solutions totalement différentes. Pour éviter de retrouver les mêmes solutions à une rotation près, nous réduisons les intervalles

de variations des paramètres terminaux. Par ailleurs, comme le rectangle correspond à une solution de conception, il est nécessaire qu'il possède une épaisseur minimale, que l'on note ε. Ainsi, la borne inférieure de l'intervalle de variation de la largeur est ε ; la borne supérieure pour la largeur correspond, dans le cas du rectangle, au moment où la hauteur à égale à ε. Les intervalles de variation pour les paramètres terminaux du rectangle sont donc : $L \in [\varepsilon, 5-\varepsilon]$ et $H = 5-L$. Nous pouvons appliquer le même raisonnement pour les autres relations mathématiques de la contrainte intermédiaire (cf. Figure 3-12).

Contraintes intermédiaires	**Intervalles de variation des paramètres terminaux du rectangle**
Périmètre = 10	$L \in [\varepsilon, 5-\varepsilon]$ $H = 5-L$
Périmètre ≥ 10	$L \in [\varepsilon, 5-\varepsilon]$ $H \geq 5-L$
Périmètre ≤ 10	$L \in [\varepsilon, 5-\varepsilon]$ $H \in [\varepsilon, 5-L]$
Périmètre ≠ 10	$L \in [\varepsilon, 5-\varepsilon]$ $H \neq 5-L$

Figure 3-12. *Intervalles de variation des paramètres terminaux du rectangle pour toutes les relations mathématiques*

Par extension, on peut réduire les intervalles de variation des paramètres terminaux de la boîte (définie par trois paramètres) ou de la pyramide (définie par quatre paramètres).

Une fois que chaque contrainte intermédiaire a été traduite en contraintes terminales, il est nécessaire d'appliquer un produit cartésien sur tous les intervalles construits pour chaque paramètre terminal. Ceci permet d'obtenir l'intervalle de variation de chaque paramètre terminal. On est sûr que les valeurs contenues dans cet intervalle satisfont les contraintes du cahier des charges intermédiaires. Toutefois, le résultat du produit cartésien peut donner un ensemble discontinu. [Yao et al. 97] propose un algorithme pour déterminer les intervalles de variation minimaux à partir de contraintes (exprimées de manière similaire) ; toutefois les auteurs ne gèrent pas le problème de discontinuité des intervalles.

3.4.3 Formalisation du passage entre contraintes intermédiaires et terminales

Dans ce paragraphe, nous formalisons la traduction de contraintes intermédiaires en contraintes terminales qui débouchent sur des intervalles des paramètres terminaux.

Soient :

- T une fonction qui traduit le cahier des charges intermédiaire (ensemble de contraintes intermédiaires) en intervalles sur les paramètres terminaux (T : Cahier des charges intermédiaire ↦ Intervalles) ;
- P une fonction qui traduit chaque contrainte intermédiaire en un ensemble de contraintes sur les paramètres terminaux, tels que PT_i est un paramètre terminal de la classe de forme, Rel_i et Exp_i sont respectivement la relation et l'expression mathématique associées à la contrainte terminale. La fonction P est définie de la façon suivante :

P : Contrainte intermédiaire ↦ Intervalles

$$< PI, Relation, Expression, Poids > \rightarrow \bigcup_{i=1}^{n} < PT_i, \mathrm{Rel}_i, Exp_i(\bigcup_{j=1}^{i-1} PT_j) >$$

$\bigcup_{j=1}^{i-1} PT_j$ exprime le fait que les bornes de l'intervalle d'un paramètre terminal peut dépendre des paramètres terminaux dont les intervalles ont été définis auparavant. En réalité, le but est d'obtenir au moins un intervalle dont les bornes sont des réels et non des fonctions dépendant d'autres paramètres terminaux comme nous avons pu le voir dans l'exemple du rectangle. Les classes de forme contenues dans l'espace des solutions sont construites de la façon suivante : soit *CdCI* le cahier des charges intermédiaire, *C* une classe de forme définie par p paramètres terminaux (correspondant à une forme primitive). On cherche l'image de *CdCI* par la fonction *T* pour la classe *C* :

$$T(CdCI) = T\left(\bigcup_{i=1}^{n} C_i\right) \text{ où } C_i \text{ est une contrainte intermédiaire} \quad (1)$$

$$= \bigcup_{i=1}^{n} P(C_i) \quad (2)$$

$$= \bigcup_{i=1}^{n} \bigcup_{j=1}^{p} < PT_j, Op_j, Exp_j(\bigcup_{r=1}^{j-1} PT_r) > \text{ grâce à la définition de la fonction } P \quad (3)$$

$$= \bigcup_{i=1}^{n} \bigcup_{j=1}^{p} < PT_j, IT_j > \text{ avec } IT_j = \left[b_1^j, b_2^j\right] \text{ où } b_{k=1,2}^j \text{ varie en fonction de } \bigcup_{r=1}^{j-1} PT_r \quad (4)$$

$$= \bigcup_{j=1}^{p} < PT_j, \bigcap_{k=1}^{n} IT_k > \quad (5)$$

L'équation (1) exprime le f ait que toutes les contraintes intermédiaires sont prises en compte pour déterminer les intervalles des paramètres terminaux. Chacune de ces contraintes est traduite en un ensemble de contraintes sur les paramètres terminaux (cf. équation 2 et 3). Ensuite, les expressions arithmétiques des contraintes sont transformées en intervalles en considérant la relation de la contrainte $\{\leq, \geq, =, \neq\}$. Cependant, comme chaque contrainte est traduite pour chaque paramètre terminal, l'équation (4) précise qu'il existe plusieurs intervalles pour chaque paramètre terminal. Comme le souhait est de satisfaire toutes les contraintes intermédiaires, il est nécessaire de calculer l'intersection de tous ces intervalles (équation 5). Ainsi, cette dernière équation spécifie toutes les instances de la classe de forme *C* qui peuvent être considérées comme solution.

Il est possible que l'intersection de tous les intervalles soit vide. Ce résultat peut être interprété de trois façons différentes :

- il est possible que le problème de conception défini par le concepteur soit sur–contraint. Dans ce cas, le système ne peut que le signaler au concepteur qui devra relâcher un certain nombre de contraintes intermédiaires et par conséquent modifier le cahier des charges du client. C'est une façon de faire intervenir le concepteur lors du passage de la fonction à la forme en lui demandant une prise de décision(cf. Chapitre 2 §3.3) ;

- si le souhait n'est pas de modifier le cahier des charges intermédiaire (donc de manière détournée, le cahier des charges du client), il est simplement possible de considérer ce résultat nul comme le fait que la classe de forme ne soit pas candidate quelles que soient les instances possibles de cette classe. Si ce choix d'interprétation est fait et si l'intersection est vide quelle que soit la classe de forme considérée, alors on peut considérer que le cahier des charges intermédiaire est sur–contraint ; ce qui nous ramène à la première interprétation ;

– si le concepteur considère que le cahier des charges est correct et qu'aucune solution n'est trouvée et si la volonté du concepteur est d'obtenir dans tous les cas un certain nombre de solutions, il est possible de ne pas appliquer le produit cartésien en sachant que les valeurs attribuées aux paramètres terminaux ne satisferont pas toutes les contraintes intermédiaires. Il faudra néanmoins pénaliser la forme considérée d'autant plus que la valeur attribuée au paramètre intermédiaire ne vérifie pas la contrainte intermédiaire.

3.4.4 Synthèse de formes combinées

Auparavant, nous avons précisé qu'une classe de forme correspondait à une morphologie définie par un ensemble de paramètres terminaux. Les classes de forme qui ont été données en exemple sont principalement des formes primitives telles que la sphère, le cylindre, le cube, la pyramide … Néanmoins, ces classes ne suffiront pas à générer des solutions satisfaisant le concepteur pour des cas de conception réels (voiture, four à micro-ondes …). En effet, une voiture ne peut pas être représentée simplement par une boîte. Il faut donc donner la possibilité au système de proposer des formes plus complexes. Ces formes peuvent être vues comme une combinaison de formes primitives dans la mesure où on peut réutiliser la définition des classes de forme primitive. Par exemple, on peut les combiner par des opérations booléennes (cf. Figure 3-13).

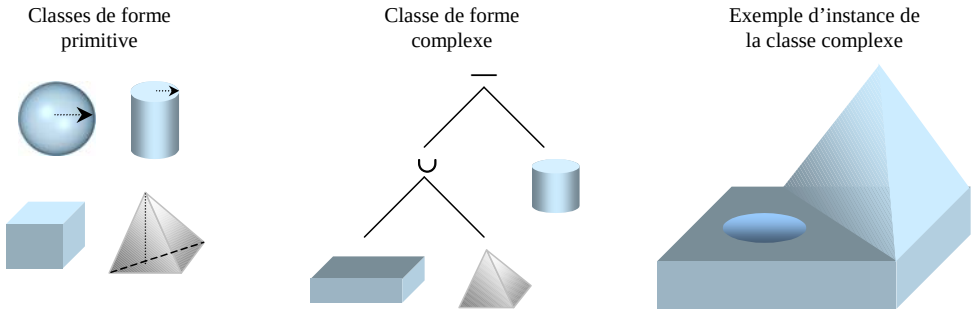


Figure 3-13. *Exemple de définition d'une classe « complexe »*

Intuitivement, comme on sait générer des formes primitives satisfaisant le cahier des charges intermédiaire, on peut imaginer qu'on est capable de faire de même avec une combinaison de formes primitives. Pour cela, la méthode présentée dans les précédents paragraphes n'empêche pas la définition d'une classe de forme correspondant à une combinaison de formes primitives. La différence par rapport aux autres classes représentant des formes primitives sera le nombre de paramètres nécessaires pour la définir. En effet, le nombre de paramètres terminaux de la classe combinée sera supérieur à la somme des paramètres utilisés pour définir chaque classe de forme primitive. Si la classe combinée est effectivement représentée par un arbre CSG, des paramètres de position, d'orientation et d'échelle seront nécessaires.

La méthode proposée permet donc de générer des formes combinées satisfaisant le cahier des charges intermédiaire à partir du moment où la morphologie de la forme (contenue dans la classe correspondante) est décrite par le concepteur dans une bibliothèque de classes. Ce principe est un bon compromis industriel entre l'approche simplifiée qui consiste à générer des formes basiques (sphère, cylindre, tétraèdre …) et une approche qui consisterait à générer des formes très complexes pour lesquels il faudrait déterminer le plus automatiquement possible la morphologie ainsi que les valeurs des paramètres terminaux. En effet, d'un point de vue industriel, il arrive fréquemment que le concepteur participe à une

conception routinière et non à une conception innovante. De ce fait, si le concepteur définit des classes de formes combinées qu'il utilise fréquemment, il pourra utiliser ce principe pour générer automatiquement des formes qui satisfont le cahier des charges qu'il aura fourni.

4 Estimation des solutions générées

Dans la partie précédente, nous avons réussi à caractériser l'espace des solutions comme un ensemble de classes de forme ; chacune d'elles contient un ensemble de paramètres définis sur des intervalles. Le fait de choisir une valeur pour chaque paramètre dans son intervalle correspondant signifie forcément que la forme résultante sera solution. Toutefois, comme il est possible de générer de cette façon une grande quantité de solutions, il est important de les comparer. Pour cela, nous proposons dans ce paragraphe un moyen d'estimer une forme par rapport au cahier des charges intermédiaire, et donc indirectement par rapport au cahier des charges du client. Une fois l'estimation définie pour une forme, on sait avec quel degré la forme satisfait le cahier des charges. Comme on peut estimer l'ensemble des formes de l'espace des solutions, on peut donc les comparer. Nous distinguons deux cas : lorsque la forme est une primitive (sphère, cylindre, boîte …) et lorsque la forme est complexe.

4.1 Formes simples

4.1.1 Les degrés de satisfaction

On définit le degré de satisfaction $DS_{ci}(f)$ d'une contrainte intermédiaire *ci* pour une forme *f* comme un nombre réel appartenant à l'intervalle [0,1]. Ce nombre exprime la qualité avec laquelle la forme *f* satisfait la contrainte intermédiaire *ci*. Plus le nombre est proche de 0, moins la contrainte est satisfaite, et plus le nombre est proche de 1, plus la contrainte est satisfaite.

On définit le degré de satisfaction $DS_{CdCI}(f)$ du cahier des charges intermédiaire pour une forme *f* comme un nombre réel appartenant à l'intervalle [0,1]. Ce nombre exprime la qualité avec laquelle la forme *f* satisfait le cahier des charges intermédiaire, c'est-à-dire l'ensemble des contraintes intermédiaires. Comme chaque contrainte intermédiaire est affectée d'un poids, le degré de satisfaction du cahier des charges peut être calculé mathématiquement à l'aide de la formule définie dans la Figure 3-14. Celle-ci représente simplement une moyenne pondérée de l'ensemble des degrés de satisfaction des contraintes intermédiaires (*n* est le nombre de contraintes).

$$DS_{CdCI}(f) = \frac{\sum_{i=1}^{n} DS_{ci_i}(f) \cdot P_{ci_i}}{\sum_{i=1}^{n} P_{ci_i}}$$

Figure 3-14. *Calcul du degré de satisfaction du cahier des charges intermédiaires pour une forme*

On remarquera que la somme des poids définis pour chaque contrainte intermédiaire n'est pas forcément égale à 1. De plus, le calcul de ce degré de satisfaction ne dépend pas de la nature de *f*. De ce

fait, le degré de satisfaction du cahier des charges intermédiaire peut être effectué dès lors que le degré de satisfaction de chaque contrainte intermédiaire est connu. Nous appelons *estimation* le calcul d'un degré de satisfaction : on peut donc estimer une unique contrainte intermédiaire ou le cahier des charges intermédiaire.

Il reste donc à définir le calcul du degré de satisfaction d'une contrainte intermédiaire pour une forme *f* donnée. La première étape de ce calcul consiste à calculer les valeurs des paramètres intermédiaires pour la forme *f*. Le fait que la forme *f* soit connue signifie que les valeurs de ses paramètres terminaux sont également connues. À partir de ces valeurs, on doit déduire la valeur du paramètre intermédiaire utilisé dans la contrainte intermédiaire. Par exemple, supposons que l'on étudie le degré de satisfaction de la contrainte intermédiaire « Volume ≥ 10 » dans le cas d'une boîte dont les valeurs de ses paramètres terminaux sont « Hauteur = 2 », « Largeur = 2 » et « Profondeur = 3 ». À ce stade, le système nécessite une formule ou un algorithme qui lui spécifie le calcul du volume de la boîte dont les paramètres sont (2,2,3). Pour cet exemple, le calcul peut simplement être défini par la formule « largeur × hauteur × profondeur ». Toutefois, dans d'autres cas, il sera éventuellement nécessaire de définir des algorithmes. Par exemple, le calcul de la compressibilité de la bouteille ou du coefficient de pénétration dans l'air d'une voiture ne pourra se faire que par un algorithme. Durant les premières conceptions, ces formules ou algorithmes devront être fournis par le concepteur. Mais, au fur et à mesure des conceptions et de la définition de formules et d'algorithmes, la base de connaissances du système augmentera et il sera possible de réutiliser des définitions de formules ou d'algorithmes pour les combiner. Comme pour la traduction des contraintes intermédiaires en contraintes terminales (cf. 3.4.2), il existe trois possibilités pour calculer la valeur d'un paramètre intermédiaire à partir d'une instance d'une classe de forme : le système sait déjà faire le calcul, le système sait déjà faire le calcul d'un autre paramètre dont dépend mathématiquement celui qu'on cherche et enfin le système ne sait pas calculer ; dans ce dernier cas, c'est au concepteur de spécifier le calcul. En réalité, le calcul de la valeur du paramètre intermédiaire peut être vue comme la fonction mathématique ou la procédure inverse à la traduction d'une contrainte intermédiaire en intervalles sur les paramètres terminaux (cf. Figure 3-15).

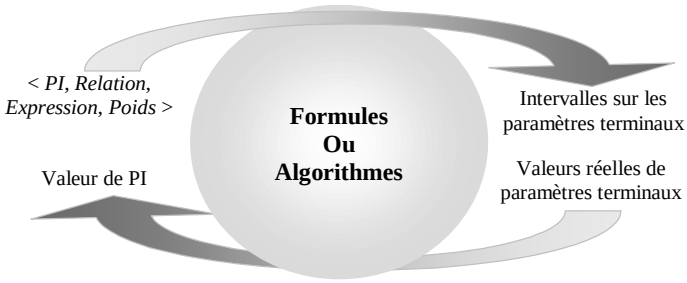


Figure 3-15. *Différentes utilisations de formules ou d'algorithmes*

La valeur du paramètre intermédiaire de l'instance de la classe de forme obtenue est appelée *valeur réelle* par opposition à la valeur de l'expression contenue dans la contrainte intermédiaire qui est nommée *valeur désirée*. La seconde étape du calcul du degré de satisfaction est de déterminer la valeur de l'expression contenue dans la contrainte intermédiaire. Il est possible que cette expression contienne d'autres paramètres intermédiaires ; dans ce cas, il est nécessaire de calculer la valeur réelle de ce paramètre sur la forme en cours d'estimation.

Enfin, une fois que la valeur réelle du paramètre intermédiaire est connue, le calcul du degré de

satisfaction de la contrainte intermédiaire *ci* est déduit d'une part de l'écart entre la valeur réelle et la valeur désirée et d'autre part d'une courbe définie de 6 dans l'intervalle [0,1] (cf. Figure 3-16). Cette courbe dépend de la relation mathématique (≤, ≥, =, ≠) utilisée dans la contrainte intermédiaire *ci*.

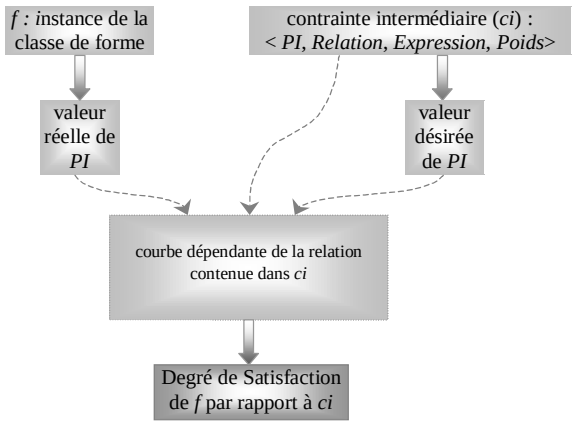


Figure 3-16. *Principe pour l'estimation d'une forme*

4.1.2 Courbes associées aux relations mathématiques

Du point de vue de la fabrication, il est difficile de respecter des valeurs numériques précises. En effet, les machines outils utilisées pour réaliser physiquement le produit –conçu virtuellement par le concepteur à l'aide d'un système informatique– ne permettent pas, aussi précises soient elles, de respecter des valeurs numériques précisées dans la maquette informatique. Il existe toujours une différence entre le produit décrit virtuellement par le concepteur et le produit réel : ces différences se situent en partie au niveau des valeurs des paramètres du produit. Par exemple, même si la valeur informatique de la largeur d'une rainure est de 5 cm, il est possible que la valeur réelle de la largeur de la rainure sur le produit construit soit égale à 5,05 cm ou 4,95 cm. Ces différences sont déjà gérées d'un point de vue informatique et sont appelées problèmes de *tolérance*. Nous amplifions ces problèmes au niveau de la génération de formes à partir de contraintes dans le sens où même si une contrainte précise que la largeur de la rainure doit être égale à 5 cm, nous supposons que si la valeur réelle de cette largeur est 4 cm, le produit ne doit pas être forcément considéré comme n'étant pas une solution au problème de conception. Nous préférons la considérer comme une mauvaise solution, mais tout de même comme une solution. C'est pourquoi, nous construisons la courbe de satisfaction associée à la relation d'égalité comme une gaussienne (cf. Figure 3-17) centrée sur la valeur désirée correspondant à la valeur de l'expression mathématique contenue dans la contrainte intermédiaire (cf. 4.1.1). Cette courbe signifie que plus la valeur réelle du paramètre intermédiaire (la largeur de la rainure dans l'exemple) est proche de la valeur désirée, plus la contrainte intermédiaire « largeur = 5 cm » sera satisfaite ; inversement, plus la valeur réelle du paramètre s'en éloigne, moins la contrainte sera satisfaite.

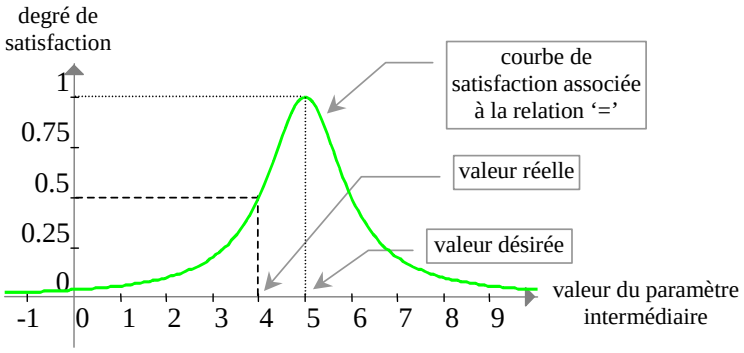


Figure 3-17. *Courbe de satisfaction associée à la relation '=' pour une contrainte du type < PI, =, 5>*

À l'aide de la courbe, il est aisé de calculer le degré de satisfaction $DS_{ci}(f)$ d'une contrainte intermédiaire *ci* (utilisant une relation d'égalité) par rapport à une forme *f* : il suffit de calculer l'image de la valeur réelle par la courbe de la Figure 3-17. Par exemple, si la valeur réelle de la largeur de la rainure est égale à 4 cm, le degré de satisfaction de la forme pour cette contrainte sera égale à 0,5. La courbe associée à la relation d'égalité peut être définie de la façon suivante :

$$\text{Égalité}_{\text{valeur désirée}} : \quad \mathbb{R} \mapsto [0,1]$$

$$\text{valeur réelle} \rightarrow \frac{1}{1 + (\text{valeur réelle} - \text{valeur désirée})^2}$$

De manière analogue, on construit les courbes pour les relations de supériorité (cf. Figure 3-18b), d'infériorité (cf. Figure 3-18d) et d'inégalité (cf. Figure 3-18c). Par exemple, la courbe associée à la relation de supériorité est construite de telle façon que plus la valeur réelle du paramètre intermédiaire est grande, plus le degré de satisfaction est grand ; sans pour autant être proportionnel. Il est fort probable que toutes les contraintes du cahier des charges ne pourront pas être idéalement satisfaites. C'est pourquoi, les courbes ont une allure telle que dès l'instant que la contrainte n'est pas satisfaite (au sens de l'algèbre de Boole), le degré de satisfaction n'est pas nécessairement égal à zéro. Cela justifie que, dans le cas d'une relation de supériorité ou d'infériorité, la valeur désirée corresponde au point d'inflexion de la courbe.

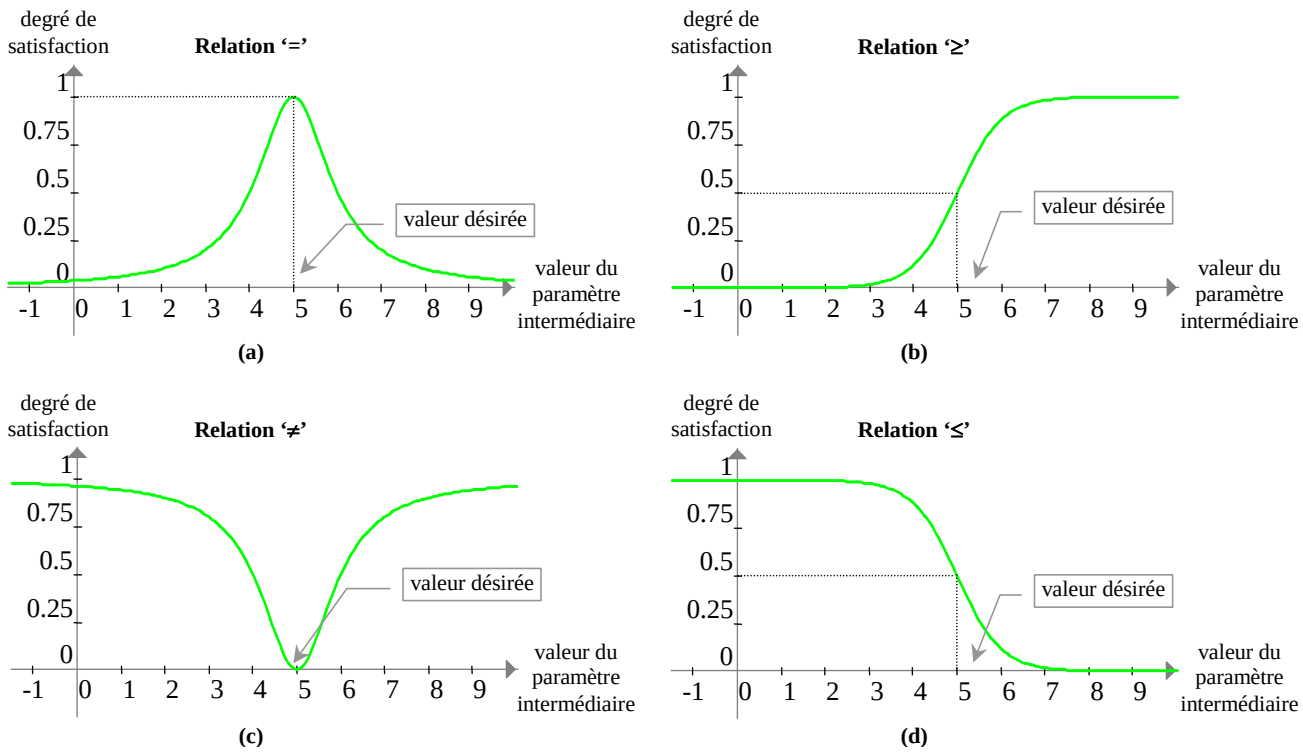


Figure 3-18. *Courbes associées à chaque relation mathématique utilisée dans les contraintes intermédiaires*

Les équations des courbes associées aux relations mathématiques sont définies dans la Figure 3-19.

$$\textit{Égalité}_{\textit{valeur désirée}} : \quad \mathbb{R} \mapsto [0,1], \quad \textit{valeur réelle} \rightarrow \frac{1}{1 + (\textit{valeur réelle} - \textit{valeur désirée})^2} \quad \textbf{(a)}$$

$$\textit{Supériorité}_{\textit{valeur désirée}} : \quad \mathbb{R} \mapsto [0,1], \quad \textit{valeur réelle} \rightarrow \frac{1 + \tanh(\textit{valeur réelle} - \textit{valeur désirée})}{2} \quad \textbf{(b)}$$

$$\textit{Inégalité}_{\textit{valeur désirée}} : \quad \mathbb{R} \mapsto [0,1], \quad \textit{valeur réelle} \rightarrow 1 - \frac{1}{1 + (\textit{valeur réelle} - \textit{valeur désirée})^2} \quad \textbf{(c)}$$

$$\textit{Infériorité}_{\textit{valeur désirée}} : \quad \mathbb{R} \mapsto [0,1], \quad \textit{valeur réelle} \rightarrow \frac{1 + \tanh(\textit{valeur désirée} - \textit{valeur réelle})}{2} \quad \textbf{(d)}$$

Figure 3-19. *Équations des courbes de la Figure 3-18*

4.1.3 Commentaires à propos des courbes représentant les degrés de satisfaction

Dans la plupart des systèmes de résolution de contraintes, une contrainte ne peut posséder que deux états : satisfait ou non. Les courbes associées aux relations mathématiques utilisées dans les contraintes intermédiaires permettent de passer progressivement d'un état satisfait à un état non satisfait. Cette progression se fait à l'aide de valeurs réelles comprises entre 0 et 1. Comme notre objectif est d'assister le

concepteur lors des premières phases de conception, nous faisons l'hypothèse qu'il ne connaît pas précisément les valeurs des paramètres de conception. Par exemple, lorsqu'il stipule que la largeur d'une rainure doit être égal à 4cm, nous l'interprétons comme *environ* égal à 4cm. Par conséquent, le fait qu'un objet de conception proposée automatiquement par le système ait une rainure de largeur égale à 5cm ne doit pas forcément être considéré comme non solution au problème posé. Il est certain que cet objet de conception répond moins bien aux spécifications qu'un objet ayant une rainure égal à 5cm. En réalité, les contraintes exprimées dans le cahier des charges intermédiaire sont considérées comme *floues* [Tong-Tong 95, Gacôgne 97, Mantel].

4.1.3.1 Autres courbes de satisfaction

Les courbes utilisées pour le calcul du degré de satisfaction d'une contrainte intermédiaire ne sont pas uniques. En effet, les courbes présentées dans la Figure 3-20 ne bouleverseraient pas pour autant la méthode utilisée. Toutefois, les premières sont définies à l'aide d'une seule équation mathématique alors que les dernières doivent être définies par morceaux. Ceci permet donc une représentation informatique plus simple. Néanmoins, le temps de calcul de la fonction « tangente^{-1} » est relativement coûteux comparée aux droites utilisées dans les nouvelles courbes. De ce fait, le choix d'une ou l'autre représentation dépend de l'optimisation que l'on souhaite réaliser.

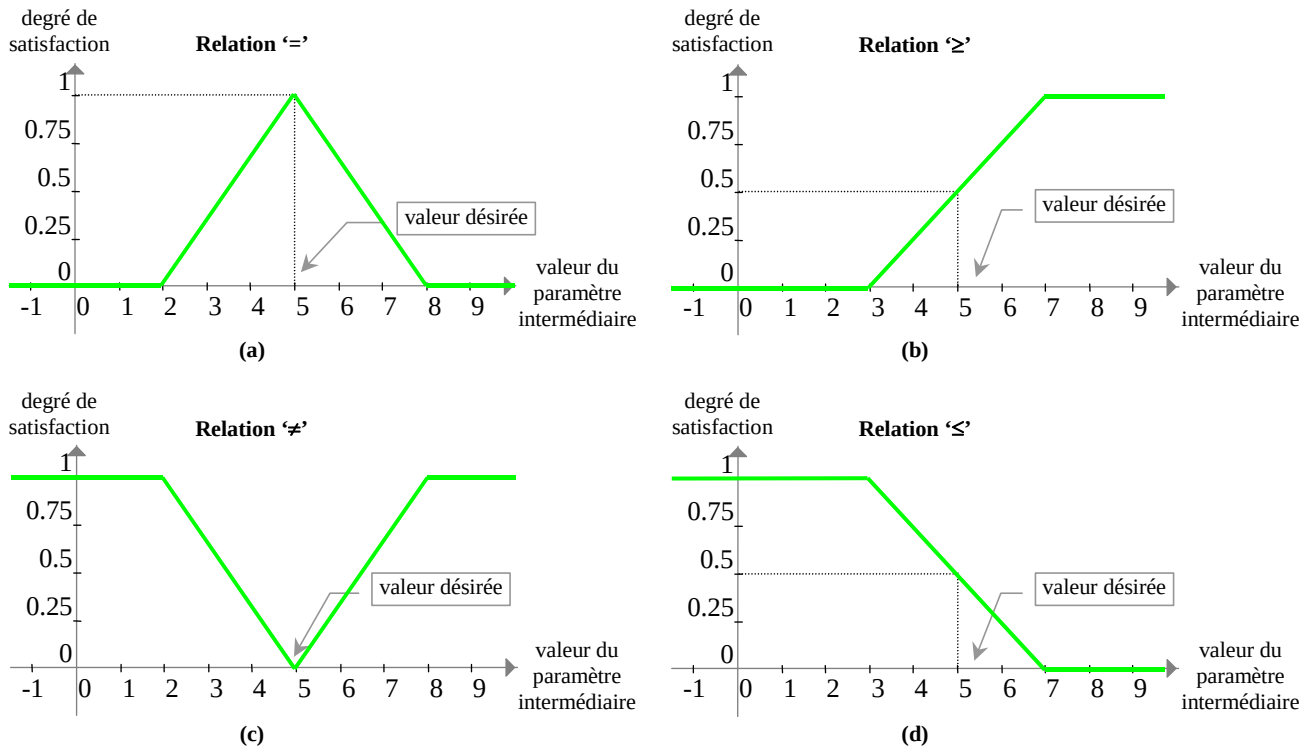


Figure 3-20. *Courbes de satisfaction définies par morceaux*

De plus, on peut remarquer que les courbes associées aux relations « supérieure à » et « inférieure à » sont en réalité définies sur l'intervalle ouvert]0,1[. Toutefois, d'un point de vue informatique, nous nous servons des problèmes d'arrondis pour considérer qu'un nombre très proche de 1 sera égal à 1 ; il en va de même pour 0. De ce fait, on peut considérer que les courbes sont définies informatiquement sur l'intervalle fermé [0,1]. Néanmoins, les courbes définies par morceaux ne connaissent pas ce problème.

Par ailleurs, comme les courbes de satisfaction doivent représenter, d'une certaine façon, l'expertise

acquise par le concepteur, il est libre de construire les courbes qui lui semble les plus appropriées à son domaine de conception. Toutefois, le choix de nouvelles courbes doit respecter les principes énoncés.

4.1.3.2 Les relations supérieure et inférieure stricte

Il est possible de construire de nouvelles relations. En l'occurrence, nous pouvons définir les relations supérieure et inférieure avec les mêmes principes énoncés précédemment. La relation supérieure stricte peut être considérée comme une combinaison des relations « supérieure à » et « différent de » (cf. Figure 3-21a). Encore une fois, il sera plus long de traiter la relation supérieure stricte comme une combinaison de deux autres relations. Il est plus simple d'appliquer une translation sur la courbe de la relation « supérieure à » (cf. Figure 3-21b).

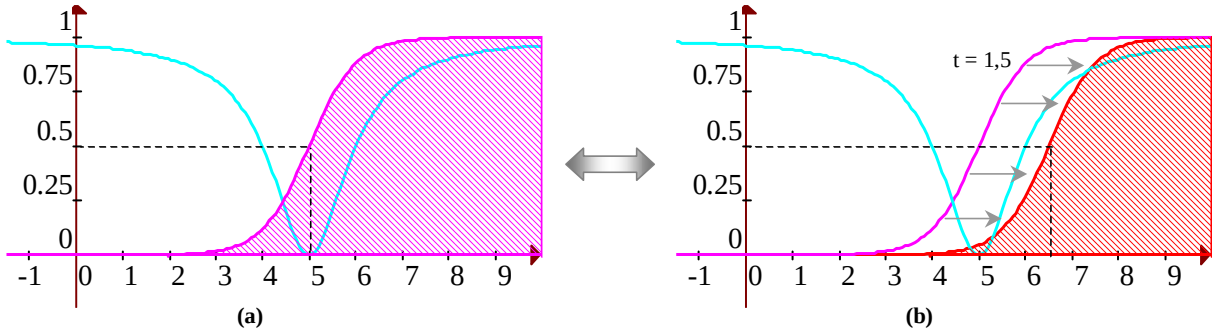


Figure 3-21. *Comparaison des relations '≥' et '>' avec la relation '≠'*

Mathématiquement, cela se fait simplement en ajoutant une constante t dans la définition de la courbe de la relation supérieure :

$$Supérieure\_Stricte_{valeur\ désirée} : \quad \mathbb{R} \mapsto [0,1]$$

$$valeur\ réelle \rightarrow \frac{1 + tanh\,[valeur\ réelle - (valeur\ désirée + t)]}{2}$$

4.1.3.3 Adaptation des courbes de satisfaction à un intervalle de variation d'un paramètre intermédiaire

Tel que nous avons défini les courbes associées aux relations mathématiques utilisées dans les contraintes intermédiaires, le domaine de variation des paramètres intermédiaires est égal à $\mathbb{R}$. Or, il est bien évident que cela ne reflète pas la réalité : par exemple, le poids ou le volume d'un objet n'est jamais négatif, ou la vitesse n'est jamais infinie. Ceci nous permet alors de supposer qu'il existe un intervalle de variation borné par deux valeurs réelles pour chaque paramètre intermédiaire comme le considère [Yao et al. 97].

Si l'on conserve la définition des courbes proposée au paragraphe 4.1.2 et que l'on suppose qu'un paramètre intermédiaire ne peut prendre de valeurs que dans son intervalle de variation, le degré de satisfaction d'une contrainte définie à l'aide de ce paramètre ne variera plus dans l'intervalle [0,1] mais éventuellement entre deux valeurs proches (entre 0,3 et 0,6 par exemple) si l'intervalle de variation du paramètre est « petit » (cf. Figure 3-22a).

Notre objectif, dans ce paragraphe, est alors de faire correspondre une courbe associée à une relation mathématique définie sur $\mathbb{R}$ à une courbe, ayant la même « allure », mais définie sur l'intervalle de variation d'un paramètre ; ceci peut se faire à l'aide d'un changement d'échelle (cf. Figure 3-22b). De cette façon, les courbes représentant les degrés de satisfaction seront dépendantes de la relation mathématique utilisée dans la contrainte intermédiaire, comme auparavant, mais aussi du domaine de variation du paramètre

intermédiaire utilisé dans cette contrainte.

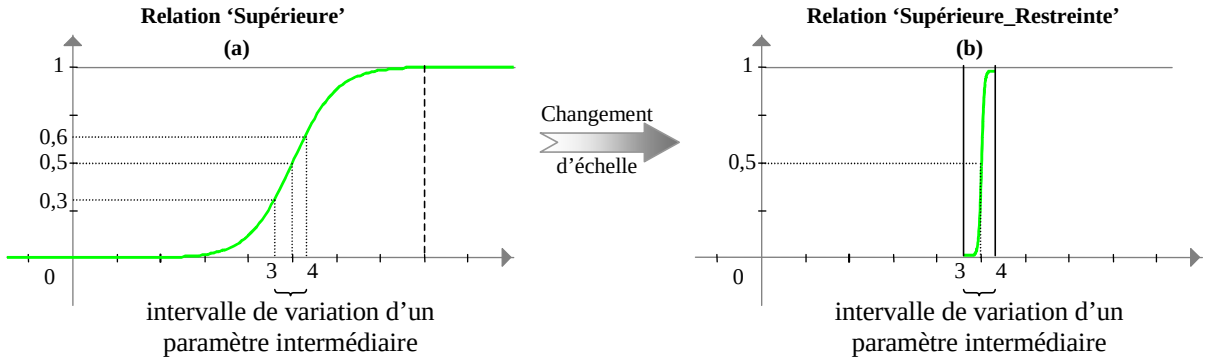


Figure 3-22. *Modification de l'intervalle de variation d'un paramètre intermédiaire*

Pour notre étude, considérons la courbe associée à la relation ≥, l'intervalle de variation $[a,b]$ pour le paramètre intermédiaire PI_1 et la contrainte intermédiaire $< PI_1 \geq vd >$ (où vd est une valeur désirée). Déterminons la nouvelle courbe associée à la relation ≥, restreinte à l'intervalle $[a,b]$ et appelée Supérieure_Restreinte. Pour garder la même « allure » que la courbe associée à la relation « Supérieure », nous considérons l'équation mathématique de la relation « Supérieure » en y ajoutant un réel k :

$Supérieure\_Restreinte_{vd}$: $[a,b] \mapsto [0,1]$

$vr \rightarrow \dfrac{1 + tanh\,[k \times (vr - vd)]}{2}$ où vr représente la valeur réelle et vd la valeur désirée.

Ensuite, toujours pour garder la même « allure », nous devons respecter le fait que le degré de satisfaction de la valeur désirée est égal à 0,5 comme nous l'avons fait dans le cas de la courbe associée à la relation « Supérieure ».

Pour appliquer un changement d'échelle sur la courbe « Supérieure », nous devons considérer un intervalle borné. Pour cela, calculons les valeurs du paramètre intermédiaire pour lesquels la machine arrondie le degré de satisfaction à 0 et à 1 pour l'équation $\dfrac{1 + tanh(vr - vd)}{2}$; notons ces valeurs a_0 et a_1 (cf. Figure 3-23a).

Comme la courbe est symétrique par rapport à la droite d'équation « $x = vd$ », on peut supposer que a_0 et a_1 seront à égale distance d de la valeur désirée vd. Le changement d'échelle consiste à faire correspondre la portion de la courbe « Supérieure » définie sur l'intervalle $[a_0,a_1]$ à la portion de la courbe « Supérieure_Restreinte » sur l'intervalle $[a,b]$. Il faut donc déterminer la valeur de k pour laquelle $Supérieure\_Restreinte_{vd}(a) = Supérieure_{vd}(a_0)$ et $Supérieure\_Restreinte_{vd}(b) = Supérieure_{vd}(a_1)$:

$$Supérieure\_Restreinte_{vd}(a) = Supérieure_{vd}(a_0) \Leftrightarrow \frac{1+tanh\,[k \times (a - vd)]}{2} = \frac{1+tanh\,(a_0 - vd)}{2}$$

$$\Leftrightarrow k \times (a - vd) = a_0 - vd = (vd - d) - vd = -d$$

$$\Leftrightarrow \boxed{k = \frac{d}{vd - a}}$$

$$Supérieure\_Restreinte_{vd}(b) = Supérieure_{vd}(a_1) \Leftrightarrow \frac{1+tanh\,[k \times (b - vd)]}{2} = \frac{1+tanh\,(a_1 - vd)}{2}$$

$$\Leftrightarrow k \times (b - vd) = a_1 - vd = (vd + d) - vd = d$$

$$\Leftrightarrow \qquad \boxed{k = \frac{d}{b - vd}}$$

On obtient deux valeurs différentes pour la même courbe ; cela signifie qu'elle doit être définie par morceaux de la façon suivante (cf. Figure 3-23b) :

$$\begin{array}{lll} Supérieure\_Restreinte_{vd} : & [a,b] \mapsto & [0,1] \\ & vr \rightarrow & \dfrac{1 + tanh\,[k_g \times (vr - vd)]}{2} \text{ avec } k_g = \dfrac{d}{vd - a} \textbf{ si } vr \leq vd \\ & & \dfrac{1 + tanh\,[k_d \times (vr - vd)]}{2} \text{ avec } k_d = \dfrac{d}{b - vd} \textbf{ sinon} \end{array}$$

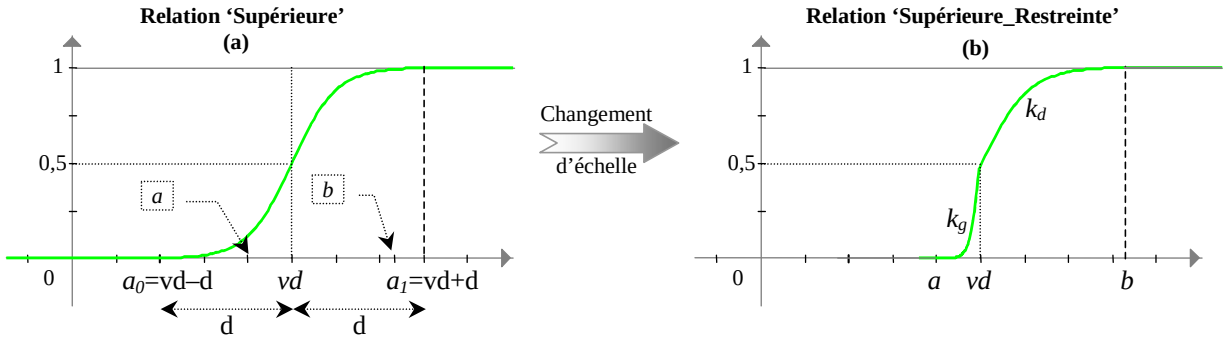


Figure 3-23. *Application d'un changement d'échelle sur la courbe associée à la relation supérieure*

On montre facilement que $k_g = k_d$ si et seulement si la valeur désirée est au milieu de l'intervalle $[a,b]$. C'est seulement dans ce cas ($k_g = k_d$) où l'allure de la courbe est conservée. Dans le cas contraire, on perd l'aspect symétrique des courbes de degré de satisfaction. Cette perte implique que l'écart existant entre deux valeurs n'est pas constant. Par exemple, si la valeur désirée vaut 3 et si les bornes de l'intervalle de variation du paramètre sont 2 et 10, alors l'écart entre les valeurs 1,2 et 1,3 ne sera pas le même que l'écart entre 8,1 et 8,2. Une première solution serait de supprimer la contrainte qui stipule que le degré de satisfaction de la valeur désirée est égale à 0,5. Toutefois, cela implique que les solutions estimées sont difficilement comparables puisqu'elles n'ont pas la même valeur de référence qui est représentée par un degré de satisfaction égal à 0,5 pour la valeur désirée (cf. Figure 3-24a).

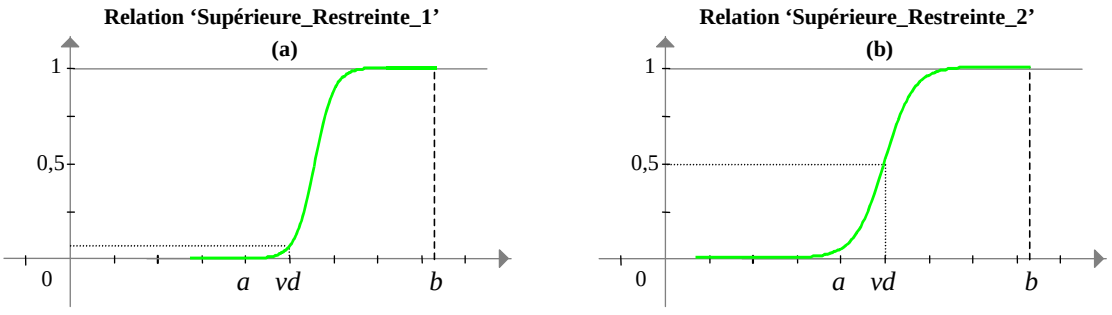


Figure 3-24. *Amélioration de la courbe associée à la relation « Supérieure_Restreinte »*

Une deuxième solution serait de supprimer la contrainte qui stipule que le degré de satisfaction de la borne inférieure de l'intervalle de variation doit être égal à 0. De ce fait, le degré de satisfaction ne variera plus entre 0 et 1. On revient donc au problème initial.

4.1.3.4 Multi-modélisation

Comme nous le présentions dans le paragraphe 3.1, on peut considérer les courbes représentant les degrés de satisfaction comme un autre modèle du cahier des charges intermédiaire. En réalité, ce cahier des charges permet de traduire les contraintes intermédiaires en contraintes terminales alors que les courbes de satisfaction, représentant les mêmes informations que le cahier des charges intermédiaires, permettent de calculer l'estimation des résultats obtenus. Ainsi, on utilise le modèle qui apparaît le plus adapté pour appliquer les traitements que l'on souhaite réaliser.

4.2 Formes combinées

Dans les paragraphes précédents, nous avons supposé que le concepteur avait fourni au système au moins une fois le moyen de calculer la valeur d'un paramètre intermédiaire pour une solution donnée ; c'est-à-dire calculer la valeur d'un paramètre intermédiaire à partir d'un ensemble de paramètres terminaux valués. Chacun de ces algorithmes de calcul doit être fourni pour chaque paramètre intermédiaire et pour chaque classe contenue dans la bibliothèque. Par exemple, si cette bibliothèque contient les classes sphère, cylindre et boîte et si le cahier des charges intermédiaire utilise les paramètres volume et poids, cela signifie que le concepteur doit fournir les algorithmes ou les formules qui permettent de calculer :

- le poids pour la sphère, pour le cylindre et pour la boîte ;
- le volume pour la sphère, pour le cylindre et pour la boîte.

Ce principe s'applique aussi bien pour les classes de forme définissant des formes simples ou combinées (cf. 3.4.4). Néanmoins, le nombre d'algorithmes ou de formules à définir risque de surcharger la tâche du concepteur plutôt que de la simplifier ; quoique l'algorithme une fois défini reste réutilisable (cf. 3.4.2). Nous étudions donc, dans ce paragraphe, la possibilité de supprimer les algorithmes qui définissent le calcul des valeurs des paramètres intermédiaires pour les classes de formes combinées. Cette étude porte uniquement sur des classes définies par des arbres CSG comme nous l'avons présenté dans le paragraphe 3.4.4.

De ce fait, nous envisageons deux possibilités pour le calcul du degré de satisfaction d'une forme définie par une classe de formes combinées : en utilisant ou non la transformation de l'arbre CSG de la forme en un modèle B–Rep. Dans la suite, cette transformation est appelée *évaluation* de l'arbre CSG.

4.2.1 Estimation sans évaluation

Dans le but d'estimer une forme complexe décrite à l'aide d'un arbre CSG sans évaluer cette forme, il est nécessaire de déduire récursivement les degrés de satisfaction des contraintes intermédiaires à partir de ceux des feuilles de l'arbre. Comme les degrés de satisfaction des contraintes intermédiaires sont connus pour les formes « simples » utilisées au niveau des feuilles de l'arbre, une première option est de déduire, pour un nœud donné, le degré de satisfaction de toutes les contraintes intermédiaires à partir des degrés de satisfaction de ses fils (cf. Figure 3-26a). Nous avons brièvement envisagé cette possibilité, mais il apparaît rapidement que c'est irréaliste. Montrons le à l'aide d'un exemple. Pour simplifier le problème, supposons que le cahier des charges spécifie la création d'un objet stable. Une boîte posée sur le sol est un objet très stable : quand on le pousse, il ne roule pas facilement. De ce fait, le degré de satisfaction de cette contrainte particulière est très élevé, proche de 1. Ceci nous amène à penser que l'on peut donc combiner

deux boîtes (objets stables) pour construire un objet plus complexe (cf. Figure 3-25a). Cependant, selon les positions et les dimensions des objets utilisés au niveau des feuilles, l'objet résultant par l'opération booléenne (une union dans notre cas) peut aussi bien être stable (cf. Figure 3-25b) qu'instable (cf. Figure 3-25c).

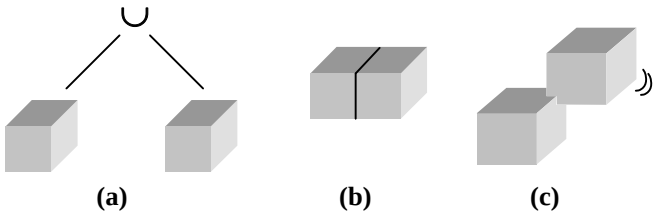


Figure 3-25. *La combinaison de deux objets stables (a) peut donner un objet stable (b) ou instable (c) [Gardan et al. 99a]*

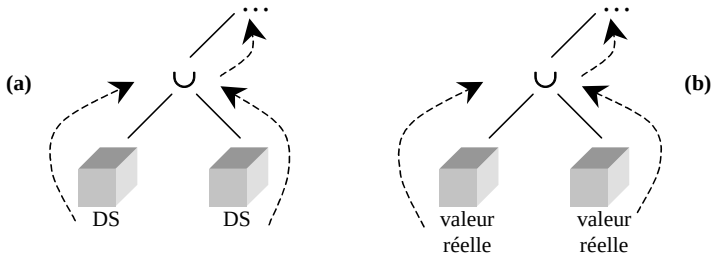


Figure 3-26. *Estimation de la forme sans évaluation*

Nous pouvons donc dire que la connaissance des degrés de satisfaction des fils d'un nœud de l'arbre est insuffisante pour calculer celui du père.

Comme les degrés de satisfaction sont calculés à partir des valeurs des paramètres intermédiaires et comme les valeurs des paramètres intermédiaires sont calculables pour les feuilles de l'arbre, une autre option serait d'obtenir récursivement les valeurs des paramètres intermédiaires de la racine de l'arbre à partir de celles calculées aux niveaux inférieurs (les valeurs des paramètres intermédiaires des nœuds fils) (cf. Figure 3-26b). Pour cela, supposons que nous disposons d'un algorithme ou d'une fonction qui permettent de calculer le périmètre d'un objet à deux dimensions, par souci de simplification. Le périmètre représente un des paramètres intermédiaires qui peut être utilisé dans le cahier des charges intermédiaire. Soit P_A le périmètre d'un opérande A et P_B le périmètre d'un autre opérande d'un arbre CSG. Tentons de calculer le périmètre ou de déterminer un intervalle qui encadre ce périmètre dans le cas où l'objet est construit à l'aide d'une union, d'une intersection ou d'une différence des deux opérandes :

- *union* : Si A et B sont deux objets disjoints alors le périmètre de $A \cup B$ est égal à la somme des périmètres des deux opérandes (Périmètre $(A \cup B) = P_A + P_B$). Si les deux objets ne sont pas disjoints, le périmètre peut être réellement plus petit que la somme des périmètres des deux opérandes (cf. Figure 3-27a). Par conséquent, $\boxed{\textit{Périmètre}(A \cup B) \in]0, P_A + P_B]}$;

- *intersection* : Si A et B sont deux objets disjoints alors le périmètre de $A \cap B$ est nul. Cependant, si les deux objets ne sont pas disjoints, le périmètre peut devenir aussi grand que la somme des périmètres

des deux opérandes (cf. Figure 3-27b). Par conséquent, $\boxed{Périmètre(A \cap B) \in [0, P_A + P_B]}$;

– *différence* : Si A est totalement inclus dans B alors le périmètre est nul. Inversement, si B est totalement inclus dans A sans qu'il y ait d'intersection entre le bord des deux objets, le périmètre est égal à la somme des périmètres des opérandes (cf. Figure 3-27c). Par conséquent, $\boxed{Périmètre(A — B) \in [0, P_A + P_B]}$.

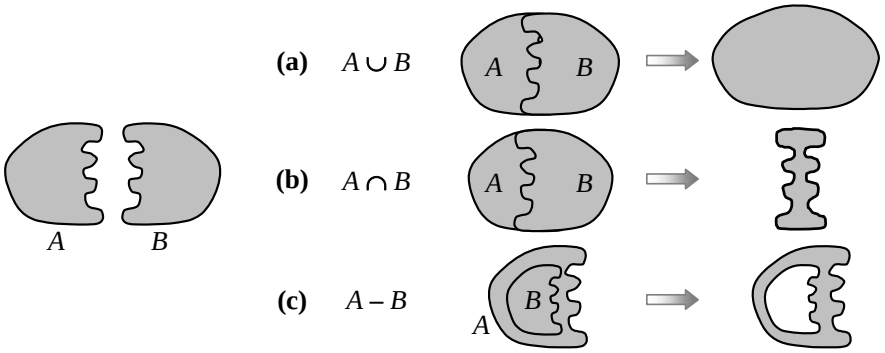


Figure 3-27. *Application d'opérations booléennes sur deux opérandes A et B*

Nous venons de montrer que quelle que soit l'opération utilisée, on ne peut encadrer le périmètre qu'avec le même intervalle qui est $[0, P_A + P_B]$. Cet intervalle n'est obtenu que pour l'application d'une seule opération booléenne, ce qui veut dire que plus l'arbre sera complexe (i.e. un nombre de nœuds très importants), plus la borne supérieure de l'intervalle sera grande ; elle tend vers l'infini. Comme on ne peut pas déterminer une valeur précise ou même un intervalle plus fin pour le périmètre, il est difficile de calculer de manière précise le degré de satisfaction de la formes complexe dans le cas du paramètre intermédiaire « périmètre ».

Ainsi, nous venons de montrer que le calcul du degré de satisfaction d'une classe de forme complexe apparaît difficile en utilisant seulement soit les valeurs des paramètres intermédiaires soit les degrés de satisfaction des primitives de l'arbre CSG. La seule possibilité envisageable est d'évaluer l'arbre CSG et de calculer le degré de satisfaction des contraintes sur ce modèle évalué. C'est ce que nous présentons dans le paragraphe suivant.

4.2.2 Estimation avec évaluation

L'évaluation d'un arbre CSG s'effectue en calculant récursivement la géométrie des sous-arbres de chaque nœud de l'arbre. On obtient alors un modèle B–Rep de la racine de l'arbre, c'est-à-dire de l'objet complexe. Plutôt que de calculer les valeurs des paramètres intermédiaires ou d'estimer le degré de satisfaction des feuilles de l'arbre, l'estimation consiste, dans un premier temps, à calculer les valeurs des paramètres intermédiaires de l'objet complexe. Ensuite, le degré de satisfaction est obtenu en appliquant la même méthode que pour les objets simples en utilisant les courbes de satisfaction (cf. 4.1). Pour le moment, la seule solution que nous envisageons pour acquérir les algorithmes de calculs des paramètres intermédiaires est de demander au concepteur de les programmer. Là encore, il sera de moins en moins nécessaire de solliciter le concepteur dans la mesure où la connaissance du système devrait accroître au fur et à mesure des conceptions et au fur et à mesure de l'ajout de nouvelles procédures. D'ailleurs, les systèmes de CAO proposent déjà de nombreuses fonctions de calcul spécialisée (calcul de la masse, de la

surface …) qui peuvent être utilisées. Finalement, notons qu'aucune meilleure solution n'est proposée même dans les systèmes à base de connaissance les plus récents : lorsque le concepteur ajoute une nouvelle règle dans la base, les méthodes utilisées dans cette règle doivent être disponibles. Dans le cas contraire, les méthodes utilisées doivent être programmées.

Par ailleurs, le besoin d'une représentation par les bords (B–Rep) n'est pas forcément un inconvénient. En effet, comme nous l'avons vu au Chapitre 2 §2.1.3, les systèmes de CAO les plus courants maintiennent deux représentations simultanées : un historique de construction (ressemblant à un arbre CSG et utilisant dans la plupart des cas de récents formalismes telles que les caractéristiques de formes) et un modèle évalué, une représentation par les bords. Ainsi, le modèle B–Rep est disponible et son utilisation pour le calcul de l'estimation ne cause aucun calcul supplémentaire. Toutefois, la tendance actuelle de la recherche est d'évincer le rôle de la géométrie en faveur de modèles de plus haut niveau sémantique [Brun 97]. De cette façon, appliquer des traitements informatiques basés sur un modèle évalué est aller contre cette tendance.

Par ailleurs, par souci de simplification, on pourrait envisager d'utiliser cette technique d'estimation à la fois pour les formes complexes et les formes simples : ceci simplifierait la tâche du concepteur qui définirait qu'une seule procédure quelle que soit la morphologie de la forme à estimer. Cela pourrait représenter une phase de transition pour l'estimation de formes jusqu'au moment où la recherche aura définit un modèle informatique du produit ne contenant qu'implicitement la géométrie de ce produit.

5 Exploration de l'espace des solutions

Dans le paragraphe précédent, nous avons présenté une méthode pour estimer une instance géométrique d'une classe de forme. Ce calcul nécessite la définition d'algorithmes qui à partir d'une instance de classe de forme déduit les valeurs des paramètres intermédiaires. Comme ces paramètres peuvent correspondre à des notions de niveau sémantique élevé telle que la compressibilité de l'objet, son coefficient de pénétration dans l'air, ou encore sa stabilité, le calcul de la valeur d'un seul paramètre peut prendre un temps non négligeable. Par ailleurs, rappelons que l'espace des solutions est construit automatiquement à partir d'autres algorithmes définis par le concepteur et à partir du cahier des charges intermédiaire. Cet espace est défini par un ensemble d'intervalles sur des paramètres terminaux de la classe de forme. De ce fait, le nombre de solutions contenues dans cet espace est infini puisque chaque paramètre est supposé être réel. Par conséquent, il apparaît impossible de montrer toutes les solutions considérées à l'utilisateur. Nous présentons dans ce paragraphe plusieurs méthodes permettant d'extraire les meilleures solutions de l'espace des solutions ainsi construit. Toutefois, nous ne considérons pas de méthodes mathématiques car, comme nous l'avons supposé (cf. Chapitre 2, §3.1.2), le problème d'optimisation ne peut pas s'exprimer sous forme mathématique puisque la fonction de coût nécessite l'application d'algorithmes pour calculer la valeur réelle des paramètres intermédiaires. De ce fait, il apparaît impossible d'utiliser des méthodes telle que les multiplicateurs de Lagrange ou la méthode du Simplex dans la mesure où elles nécessitent de dériver la fonction de coût.

5.1 Algorithmes stochastiques

À l'inverse des méthodes exactes, les méthodes stochastiques permettent d'obtenir de très bonnes

solutions dans un temps raisonnable. Elles sont utilisées dès l'instant que les méthodes mathématiques choisies sont impuissantes ou dès l'instant que le problème à résoudre est NP–difficile. Elles sont basées sur des principes aléatoires qui de manière statistique donnent de bons résultats, sans pour autant garantir l'optimalité de la solution obtenue.

Dans la suite, nous présentons deux algorithmes stochastiques et nous illustrons le fait qu'ils peuvent être utilisés.

5.1.1 Le recuit simulé

Historiquement, le principe du recuit simulé[21] s'inspire du recuit des matériaux en métallurgie : un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un minimum local. S'il est refroidi plus lentement, les atomes du matériaux se réarrangent, les défauts disparaissent et le métal possède alors une structure interne mieux ordonnée ; c'est l'équivalent du minimum global [Prins 97].

Le principe général de l'algorithme (cf. Figure 3-28) consiste à :

- choisir une solution initiale quelconque ; la température représente l'énergie de cette solution ;
- ensuite, on fait varier la température de la solution jusqu'à la température finale ou jusqu'à la stabilité de la solution ; c'est-à-dire qu'aucune meilleure solution n'a pu être générée. Le nombre maximum de solutions générées par itération est limité par Nb_Max_Modif_Tentées. La modification de la température consiste à étudier le voisinage proche de la solution courante. On étudie donc Nb_Max_Modif_Tentées solutions voisines à la solution courante. Le choix des solutions voisines se fait aléatoirement ce qui justifie le caractère stochastique de la méthode ;
- classiquement, le recuit simulé cherche à diminuer une fonction de coût. Comme nous cherchons la solution qui possède le meilleure degré de satisfaction, nous remplaçons la fonction de coût par une fonction de qualité pour rester cohérent. On calcule alors la différence de qualité qu'il y a entre la solution courante et la solution voisine considérée temporairement ; cette différence est représentée par ΔE ;
- si ΔE est positif, cela signifie que la qualité augmente et par conséquent la solution voisine est meilleure. Elle est donc conservée ;
- si ΔE est négatif, cela signifie que la qualité diminue. Il faut pénaliser cette solution d'autant plus que la température est basse et que la variation ΔE est forte. La fonction exponentielle possède les propriétés désirées. On décide de manière aléatoire l'acceptation de cette solution qui diminue la qualité à l'aide de la fonction Nb_Aléatoire(0,1) ;
- pour assurer la convergence de l'algorithme, la température est diminuée lentement à chaque itération à l'aide de la fonction Évolution(Température).

Les problèmes classiques du recuit simulé sont de déterminer les modifications aléatoires de la solution courante, le calcul de la qualité, et les valeurs initiale et finale de la température. La qualité de la solution est simplement représenté par son degré de satisfaction.

[21] simulated annealing en anglais

```
Algorithme Recuit_Simulé
    Solution_Courante ← Choix_Solution_Initiale()
    Meilleure_Solution ← Solution_Courante
    Température ← Température_Initiale
    Solution_Stable ← Faux
    Tant que Température ≥ Température_Finale et non Solution_Stable faire
        Nb_Modif_Tentées ← 0          Nb_Modif_Acceptées ← 0
        Tant que (Nb_Modif_Tentées ≤ Nb_Max_Modif_Tentées) et
                 (Nb_Modif_Acceptées ≤ Nb_Max_Modif_Acceptées) faire
            ΔE = Calcul_Variation_Qualité()
            Si ΔE > 0 alors Modif_Acceptée ← Vrai
            sinon si Nb_Aléatoire (0,1) ≤ e^(ΔE / Température) alors Modif_Acceptée ← Vrai
            sinon Modif_Acceptée ← Faux
            Fin si
            Nb_Modif_Tentées ← Nb_Modif_ Tentées + 1
            Si Modif_Acceptée alors
                Nb_Modif_Acceptées ← Nb_Modif_Acceptées + 1
                Si Qualité(Solution_Courante) > Qualité(Meilleure_Solution) alors
                    Meilleure_Solution ← Solution_Courante
                Fin si
            Fin si
        Fin Tant
        Température ← Évolution(Température)
        Solution_Stable ← (Nb_Modif_Acceptées = 0)
    Fin Tant
Fin Algorithme
```

Figure 3-28. *Algorithme général du recuit simulé*

Une solution peut être représentée par l'ensemble de ses paramètres terminaux valués. Ainsi, une modification aléatoire pour déterminer une solution voisine serait en premier lieu de sélectionner aléatoirement le paramètre terminal à modifier. Ensuite, il faudrait choisir aléatoirement la quantité à ajouter ou à retirer au paramètre choisi. Il faudra néanmoins vérifier que la solution voisine ainsi construite appartient toujours à l'espace des solutions. Enfin, la température initiale doit garantir que toutes les modifications sont acceptées pendant les premières itérations ; cela signifie que $e^{\Delta E / \text{ Température initiale}}$ est proche de 1. Comme ΔE est au moins égal à 1, il suffit de choisir une valeur nettement supérieure à 1 pour la température initiale.

5.1.2 Les algorithmes génétiques

Au début de ce chapitre, nous avons déjà présenté le principe de ces algorithmes pour générer des formes complexes. Nous proposons, dans ce paragraphe, d'utiliser ce concept informatique pour obtenir les meilleures solutions de l'espace des solutions. Toutefois, nous retrouvons, ici, les mêmes problèmes liés à ce concept qui sont la modélisation des individus, les méthodes de croisement et de mutation.

Comme pour le recuit simulé, il est possible de modéliser les individus simplement en conservant la liste des valeurs des paramètres terminaux. De ce fait, le croisement se fait en prenant une partie des valeurs d'un individu et une partie des valeurs d'un autre individu. Toutefois, cela présuppose que tous les individus manipulés par l'algorithme génétique appartiennent à la même classe. Il faudra donc appliquer la méthode stochastique pour chaque classe de forme contenue dans la bibliothèque. Enfin, la mutation peut s'effectuer de la même façon que le calcul du voisinage dans l'algorithme du recuit simulé, c'est-à-dire

modifier aléatoirement la valeur d'un des paramètres terminaux de l'individu.

On rencontre le même problème que précédemment à savoir vérifier que l'individu obtenu par croisement ou mutation appartient bien à l'espace des solutions construit. Cette opération de vérification soulève quelques difficultés à cause de l'aspect dynamique des intervalles terminaux : lorsqu'une valeur d'un paramètre terminal est choisie, il est presque inévitable que les intervalles de variation des autres paramètres terminaux changent dans la mesure où les bornes des intervalles sont fonction d'autres paramètres terminaux (cf. 3.4.3).

5.2 Algorithmes d'échantillonnage

Comme il est irréalisable de parcourir de manière exhaustive chaque classe de forme dans la mesure où elles sont définies par des intervalles réels, nous proposons d'échantillonner chaque intervalle d'une classe de forme. Nous identifions deux méthodes : l'échantillonnage simple et l'échantillonnage par interpolation qui améliore la première. Une méthode exacte permettrait de calculer les meilleures solutions pour chaque classe si on lui en laisse le temps et si on lui en donne les moyens, matériels par exemple. Comme nous ne pouvons appliquer de méthodes exactes, nous ne pourrons pas être absolument sûr que les solutions obtenues sont les meilleures dans l'absolu. Comme nous utilisons des méthodes heuristiques, seules les meilleures solutions de celles échantillonnées peuvent être calculées ; pour les différencier des précédentes, nous les appelons solutions *prometteuses*. Il se peut néanmoins que les solutions prometteuses obtenues coïncident avec les meilleures solutions. Par ailleurs, pour stimuler la créativité du concepteur, nous ne pensons pas considérer uniquement une seule solution prometteuse par classe de forme. En effet, même si une solution prometteuse n'est pas la meilleure solution prometteuse, elle peut le devenir seulement en changeant une partie de sa forme que le système informatique est incapable de deviner et de réaliser automatiquement. Le fait de la présenter au concepteur peut le stimuler à choisir cette solution en la modifiant selon ses désirs (cette étude fait l'objet du paragraphe 6) pour obtenir une très bonne solution de conception, ou à étudier la morphologie de la solution prometteuse pour répercuter dans la mesure du possible les parties qui lui plaisent le plus sur la solution la plus prometteuse pour construire une solution encore plus prometteuse.

5.2.1 Algorithme d'échantillonnage simple

L'échantillonnage consiste à choisir des valeurs arbitraires pour chaque paramètre terminal d'une classe de forme. Une instance géométrique peut alors être construite puisque la morphologie de la forme est connue par la classe elle-même et la géométrie est obtenue par la valuation de ses paramètres terminaux. Il est nécessaire d'estimer chaque instance choisie. L'estimation permet ensuite de comparer toutes les instances pour n'en retenir que les meilleures. Le problème de l'échantillonnage réside dans le choix des valeurs arbitraires données aux paramètres intermédiaires. Comme on connaît les bornes des intervalles de variation de ces paramètres, il suffit à chaque itération, d'ajouter un nombre ε fixé. Néanmoins, cette possibilité n'est pas très judicieuse dans la mesure où les intervalles n'ont pas tous la même longueur. En effet, nous n'obtiendrons pas le même nombre d'échantillons pour tous les intervalles. De ce fait, il est plus juste de préciser le nombre d'échantillons à estimer et d'en déduire pour chaque intervalle le nombre ε.

Néanmoins, plus le nombre d'échantillons à estimer est grand, plus la méthode sera coûteuse en temps

du fait de la complexité éventuelle de l'algorithme qui permet de calculer la valeur réelle d'un paramètre intermédiaire. À l'inverse, plus le nombre d'échantillons est petit, moins on a de chance de faire coïncider les solutions prometteuses avec les meilleures solutions. C'est pourquoi, nous proposons une méthode qui permet d'augmenter le nombre d'échantillons à estimer plus on s'approche d'une solution prometteuse. Cette méthode n'assure pas pour autant le calcul des meilleures solutions comme le font les méthodes exactes.

Au début, le principe consiste à estimer un nombre d'échantillons faible pour chaque intervalle. Comme chaque paramètre de la classe de forme est valué, il est possible d'estimer l'instance de forme correspondante. On conserve ensuite les instances qui possèdent le meilleur degré de satisfaction. Enfin, on échantillonne à nouveau avec le même nombre d'échantillons mais dans un intervalle beaucoup plus restreint : celui délimité par les échantillons voisins de la solution la plus prometteuse (cf. Figure 3-29).

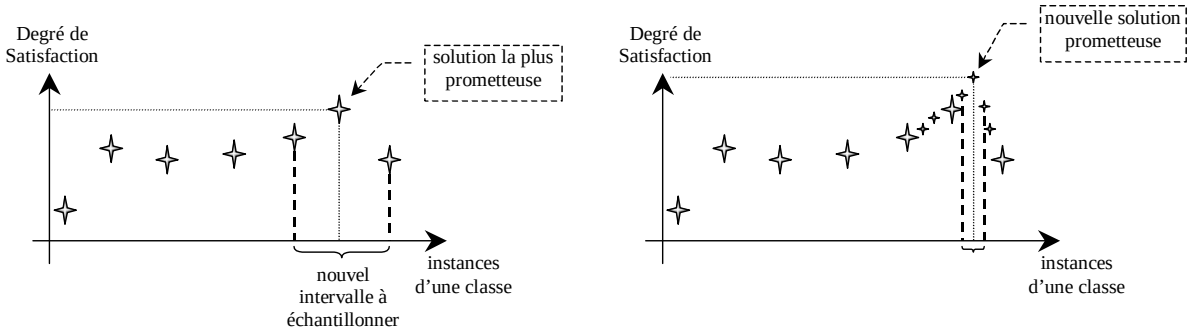


Figure 3-29. *Méthode d'échantillonnage simple*

Il faut toutefois définir un test d'arrêt pour éviter que la méthode étudie trop d'échantillons. Le plus simple est de préciser un nombre maximum d'itérations à appliquer.

Par ailleurs, dans ce qui précède, nous avons présenté le cas d'une seule solution prometteuse. Il est fort possible qu'il en existe plusieurs. Si les solutions prometteuses sont voisines les unes des autres, il suffit de considérer la borne inférieure de l'intervalle à échantillonner comme la première solution prometteuse et la borne supérieure comme la dernière solution prometteuse (cf. Figure 3-30a). Si les solutions les plus prometteuses ne sont pas voisines, il suffit de considérer plusieurs intervalles à échantillonner (cf. Figure 3-30b).

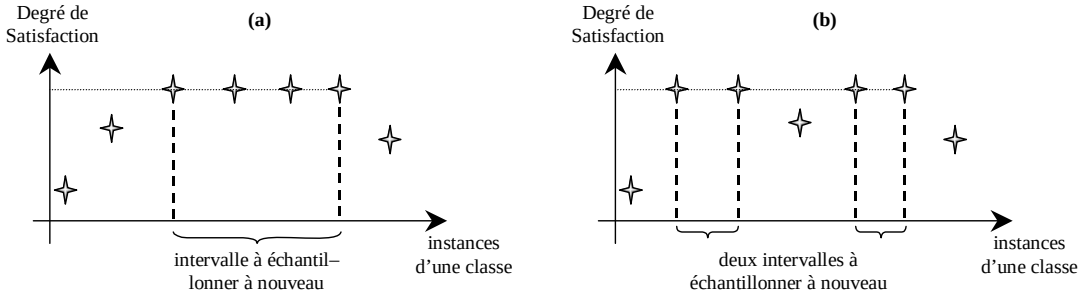


Figure 3-30. *Choix des intervalles à échantillonner à nouveau*

5.2.2 Combinaison de l'échantillonnage et de l'interpolation

Dans ce paragraphe, nous présentons une autre méthode basée sur le principe de la précédente. L'idée est toujours d'échantillonner plus finement aux abords d'une éventuelle solution prometteuse en se basant

sur un nombre d'échantillons fixe pour chaque intervalle d'une classe de forme (cf. Figure 3-31a). Toutefois, cette méthode consiste à interpoler les points des degrés de satisfaction obtenus (cf. Figure 3-31b) par une méthode d'interpolation quelconque. On peut par exemple choisir une méthode d'interpolation par morceaux qui consisterait à construire uniquement des équations de degré 3. Les méthodes classiques construisent automatiquement au mieux une courbe de degré $p-1$ avec p le nombre de points à interpoler. L'utilisation d'une interpolation par morceaux simplifie le calcul des zéros de l'équation.

Les courbes sont à considérer uniquement dans le cas où le nombre de paramètres de la classe de forme est unique. Pour deux paramètres, on devra interpoler une surface ; pour trois paramètres, un volume et pour un nombre de paramètres supérieur à trois, une hypersurface. Dans ce qui suit, nous présentons la recherche de solutions prometteuses dans le cas de l'interpolation d'une courbe. La même méthode pourra être appliquée pour un nombre de paramètres supérieur à un.

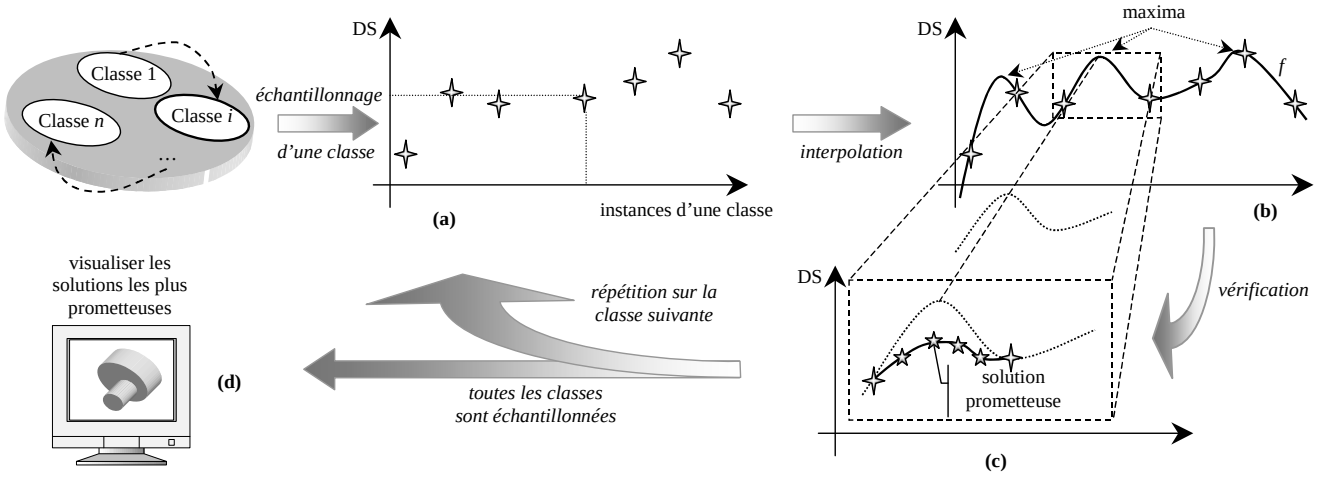


Figure 3-31. *Méthode d'optimisation par échantillonnage et interpolation [Gardan et al 99b]*

Une fois l'interpolation effectuée, on peut calculer de manière mathématique les maxima, correspondants éventuellement à des solutions prometteuses. Il reste à vérifier que ces maxima correspondent à des solutions prometteuses puisqu'on n'est pas sûr que l'interpolation reflète la réalité. Pour cela, nous envisageons deux méthodes : une méthode mathématique et une méthode moins formelle. Dans le premier cas, la matrice des dérivées partielles doit être calculée et éventuellement un développement de Taylor pour obtenir les valeurs des paramètres terminaux qui correspondent au maximum. Cela implique de nombreux calculs numériques. C'est pourquoi, nous optons, plutôt, pour la deuxième méthode qui consiste à échantillonner à nouveau aux abords des maxima. La méthode la plus rapide consiste à échantillonner l'intervalle délimité par les échantillons voisins du maximum, déjà calculés à l'itération précédente (cf. Figure 3-31c). Une fois les solutions prometteuses calculées pour chaque classe de forme, elles peuvent être présentées par ordre décroissant au concepteur (cf. Figure 3-31d).

L'avantage de cette méthode est qu'elle tient compte du résultat de toutes les instances échantillonnées : c'est-à-dire qu'elle tient compte du comportement de l'estimation. En effet, comme le montre la Figure 3-32, l'échantillon possédant le meilleur degré de satisfaction ne converge pas forcément vers une meilleure solution. Il se peut effectivement qu'une très bonne solution se situe entre deux échantillons ayant un degré de satisfaction inférieur au meilleur échantillon. Par conséquent, l'intervalle à échantillonner pour la même itération n'est pas le même suivant la méthode utilisée. La nouvelle méthode interprète

mieux l'estimation effectuée sur l'ensemble des échantillons. Par ailleurs, si un maximum se situe entre deux échantillons, l'intervalle considéré par l'interpolation est deux fois plus petit que l'intervalle considéré par la méthode par échantillonnage simple.

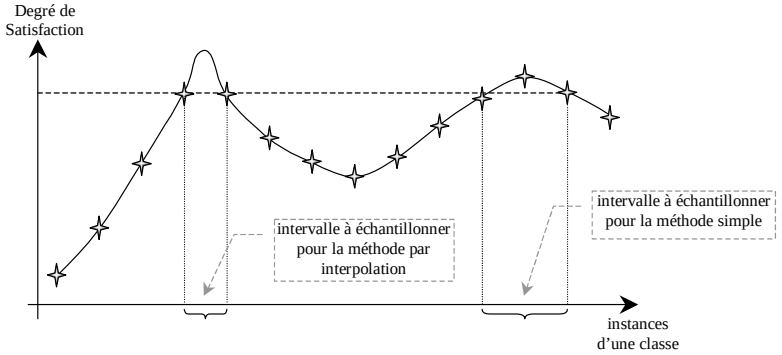


Figure 3-32. *Différence de l'intervalle choisi par chaque méthode*

6 Intervention de l'opérateur

Quelles que soient les méthodes de synthèse de forme ou les méthodes d'optimisation utilisées, il apparaît très difficile pour un système informatique de proposer automatiquement des solutions qui satisfont le concepteur dès les toutes premières itérations (cf. Chapitre 2 §3.3). Ceci est dû partiellement à la complexité des produits manufacturiers à synthétiser et à la masse de connaissances liée au problème de conception : un grand nombre de métiers sont concernés pour tout produit à concevoir. L'utilisation d'un système informatique pour mener à bien une conception de produits dans son intégralité nécessite, d'une part une automatisation des méthodes de conception et d'autre part une modélisation adéquate du savoir-faire de l'entreprise : c'est la notion de gestion des connaissances. Le problème d'intégration de connaissances dans la modélisation produit représente un axe de recherche à part entière et sort du cadre de ce manuscrit. Néanmoins, les méthodes proposées dans les paragraphes précédents sont une avancée notoire pour une éventuelle future automatisation. Comme nous ne revendiquons pas une complète automatisation de la synthèse d'une forme à partir de contraintes fonctionnelles, il apparaît donc nécessaire de faire intervenir le concepteur lors de cette automatisation partielle. Comme nous l'avons déjà proposé au Chapitre 2 (§3.3), plusieurs possibilités peuvent être envisagées :

- le choix des valeurs de certains paramètres liés au système informatique représente une partie de la connaissance qui n'est sans doute pas formalisable par un expert du domaine. Ces valeurs dépendent fortement du métier concerné et ne peuvent être choisies que par expérimentation ;
- l'utilisation de règles métiers représente une autre partie de la connaissance des métiers concernés par le problème de conception ;
- enfin, l'utilisation d'outils de dialogue adaptés permettrait au concepteur de modifier directement les formes proposées automatiquement par le système. Cette partie n'est pas détaillée dans la suite dans la mesure où elle représente, elle aussi, un axe de recherche à part entière.

Les paramètres que le concepteur peut manipuler sont au moins au nombre de trois :

- les poids des contraintes intermédiaires : à supposer que l'on sache générer une forme complexe automatiquement, le fait de modifier les poids des contraintes du cahier des charges peut amener le système à proposer des formes différentes. Mais, le choix idéal du poids de chaque contrainte ne peut se faire que par tâtonnement ; c'est au concepteur de comparer les solutions obtenues par rapport aux poids attribués aux contraintes ;
- le nombre d'échantillons à considérer par intervalle : ce nombre doit être choisi judicieusement en fonction de l'importance que le concepteur accorde aux solutions et en fonction du temps de calcul des différents paramètres intermédiaires. Pour estimer une solution, chaque paramètre intermédiaire utilisé dans le cahier des charges doit être calculé pour chaque échantillon. C'est pourquoi, plus le calcul de la valeur réelle du paramètre intermédiaire est long, plus l'estimation sera longue. De ce fait, plus le nombre d'échantillons sera grand, plus le calcul des solutions prometteuses sera long. Et comme c'est au concepteur de fournir les algorithmes qui calculent les valeurs réelles des paramètres intermédiaires, le choix du nombre d'échantillons par intervalle lui revient ;
- l'allure des courbes de satisfaction : nous avons proposé aux paragraphes 4.1.2 et 4.1.3.1 deux ensembles de courbes permettant d'estimer les solutions générées automatiquement par le système informatique. Néanmoins, il est possible que ces courbes ne soient pas adaptées au domaine de conception du concepteur. Il peut les modifier comme bon lui semble. Par exemple, tout en gardant l'allure des courbes proposées (cf. 4.1.2), il peut simplement modifier la valeur de la constante *k* contenue dans l'équation de la courbe (cf. Figure 3-33).

Nous remarquons que l'ensemble des paramètres du système sur lesquels le concepteur peut agir est lié à l'estimation d'une forme par rapport au cahier des charges intermédiaire. Mais, l'intervention du concepteur au niveau de la synthèse d'une forme se fait par les différents algorithmes qui permettent de passer des contraintes intermédiaires aux contraintes terminales et inversement.

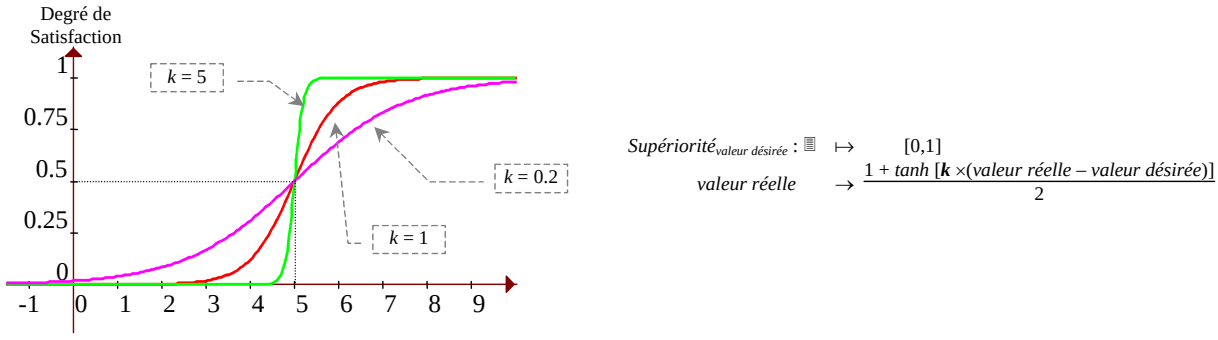


Figure 3-33. *Différentes courbes pour différentes valeurs de la constante k – courbe associée à la relation « Supérieure »*

7 Conclusion

Au début de ce chapitre, nous avons présenté plusieurs pistes pour générer aussi automatiquement que possible une forme à partir d'informations fonctionnelles : une première approche qui consiste à assembler un ensemble de caractéristiques de forme répondant chacune à au moins une fonction du cahier des

charges du client ; la deuxième, considérant plutôt des solides que des surfaces fonctionnelles, propose de déformer des objets primitifs pour mieux satisfaire au fur et à mesure des déformations les fonctions spécifiées par le client ; enfin, la troisième qui peut aussi être vue comme une combinaison des deux premières approches propose de combiner des solides primitifs tels que sphère, cylindre, boîte … Cette étude nous a permis de mettre en avant la nécessité de proposer un modèle intermédiaire qui permet un passage moins brutal des fonctions à des formes. Ce modèle intermédiaire a été caractérisé par le cahier des charges intermédiaire modélisé comme un ensemble de contraintes sur des paramètres aussi appelés intermédiaires. L'automatisation apparaît plus simple puisque le modèle intermédiaire n'est plus totalement exprimé dans le langage naturel. Les contraintes intermédiaires peuvent alors être traduites en contraintes sur des paramètres de niveau sémantique moins élevé : ce sont les paramètres terminaux utilisés dans chaque classe de forme. L'association directe des fonctions à des formes, utilisée dans les approches présentées au début du chapitre, s'est transformée en méthodes de traduction entre les paramètres intermédiaires et les paramètres terminaux ainsi qu'en méthodes calculant la valeur d'un paramètre intermédiaire à partir des valeurs des paramètres terminaux. L'intérêt de telles méthodes permettant une traduction automatique réside dans leurs réutilisation pour définir d'autres méthodes de traduction éventuellement plus complexe : c'est-à-dire permettant d'utiliser des paramètres intermédiaires de niveau sémantique plus élevé.

Ces méthodes favorisant la traduction de contraintes intermédiaires en contraintes sur des paramètres terminaux permettent de définir les intervalles de variation de ces paramètres terminaux en appliquant un simple produit cartésien. Le résultat de ce produit constitue l'espace des solutions. Comme les paramètres sont considérés comme étant des réels, nous obtenons une vaste panoplie de solutions qu'il faut comparer pour n'en présenter que les meilleures au concepteur. Pour cela, nous avons proposé une méthode qui permet de calculer le degré de satisfaction d'une forme par rapport aux contraintes intermédiaires, et indirectement par rapport aux fonctions exprimées par le client. Cette méthode d'estimation est basée sur le principe de la logique floue dans le sens où une forme qui ne vérifie pas une contrainte en particulier ne doit pas forcément être rejetée de l'espace des solutions. Ceci n'est évidemment possible que si l'on se place dans le cas d'une conception innovante et dans les premières phases de conception.

Enfin, la synthèse automatique de formes ne peut générer, dès les premières itérations, une solution qui satisfait entièrement le concepteur ; surtout vu la complexité du problème. Nous avons donc proposé plusieurs moyen de paramétrer le système de génération suivant le domaine de conception du concepteur. Des outils de dialogues adaptés à une interaction plus conviviale entre le système de génération et d'estimation de formes et le concepteur seraient les bienvenus. Toutefois, ils représentent à eux seuls à un axe de recherche à part entière.

Dans le chapitre suivant, nous proposons une mise en œuvre des méthodes de synthèse et d'estimation pour valider notre démarche.

Chapitre 4 Réalisation informatique et application industrielle

« L'itération est humaine, la récursion divine »
L. Peter Deutsch

1 Introduction

L'objectif de ce chapitre est de démontrer que les propositions faites précédemment sont tangibles. Comme le but principal est de fournir une assistance informatique lors des premières phases de conception d'un produit manufacturier, il est important de montrer que les méthodes avancées ne sont pas uniquement valides d'un point de vue théorique. Pour cela, nous présentons tout d'abord une maquette informatique qui applique directement certaines méthodes développées dans le précédent chapitre. Cette maquette a pour objectif de prouver la faisabilité des différentes méthodes. Elle se place dans un cas de conception assez simplifié puisqu'elle propose au concepteur différentes solutions de plateaux de tables. Ensuite, la deuxième partie de ce chapitre montre qu'une telle méthodologie peut être appliquée au domaine de la fonderie afin de proposer au concepteur des solutions qu'il n'avait pas envisagées auparavant. Nous expliquerons que la méthodologie annoncée peut être perfectionnée dès l'instant qu'un domaine de prédilection est choisi. L'application de cette méthode au domaine de la fonderie permet alors d'évincer certains inconvénients.

2 Mise en œuvre

Dans cette partie, nous nous proposons d'étudier la réalisation d'une maquette informatique utilisant les méthodes présentées (cf. Chapitre 3, §3 à §5). Pour mener à bien cette réalisation, il a été important de prendre des décisions simplificatrices. En l'occurrence, le contexte de développement de la maquette informatique est le suivant : l'objectif est de concevoir des plateaux de tables. De ce fait, la modélisation d'objets en trois dimensions s'avère inutile. Il faut savoir que l'ajout d'une troisième dimension ne modifie en rien notre méthodologie de conception. La gestion d'autres paramètres intermédiaires n'apporterait rien de significatif sauf la possibilité d'enrichir le cahier des charges intermédiaire. Par ailleurs, seul le paramètre

intermédiaire « périmètre » est utilisé pour définir les informations du cahier des charges intermédiaires. Par contre, pour pouvoir définir un cahier des charges assez significatif, nous avons implémenté les quatre relations mathématiques $\{\leq, \geq, \neq, =\}$ utilisées lors du calcul du degré de satisfaction.

Une première partie est consacrée à l'analyse orientée objets des méthodes de synthèse et d'estimation d'une forme à partir de contraintes pour la réalisation de notre maquette informatique. Cette étude sera représentée suivant le formalisme UML[22]. Le principe et le formalisme de la méthode UML [Muller 97] sont succinctement décrits dans l'Annexe B. Pour terminer, nous présentons brièvement l'interface et l'utilisation de notre maquette.

2.1 Le modèle orienté objets

Avant d'examiner plus en détail le logiciel et la manière dont il a été construit, nous présentons d'abord brièvement les caractéristiques du modèle orienté objets. Ensuite, nous justifions le choix de cette approche par la description des contraintes de développement et des qualités recherchées. Enfin, à travers un ensemble de diagrammes UML, nous détaillons l'organisation de cette application, aussi bien du point de vue statique (organisation des objets) que du point de vue dynamique (comportement des objets).

2.1.1 Concepts fondamentaux des modèles et des méthodes orientés objets

[Booch 94] a donné une définition du modèle orienté objets qui repose sur quatre propriétés fondamentales :

- l'*abstraction* : « Une abstraction fait ressortir les caractéristiques essentielles d'un objet (qui le distingue de tous les autres) et donc procure des frontières conceptuelles rigoureusement définies, relativement au point de vue de l'observateur ». Une abstraction est plus communément appelée classe. Un objet est donc une instance de classe. Les caractéristiques essentielles se traduisent généralement par un ensemble d'opérations (ou méthodes). La représentation interne de l'objet n'en fait pas partie ;

- l'*encapsulation* : « L'encapsulation est le procédé qui cache tous les détails d'un objet qui ne font pas partie de ses caractéristiques essentielles ». En d'autres termes, le principe d'encapsulation interdit l'accès direct aux données liées à la représentation de l'objet. Un objet ne peut être manipulé qu'à travers des requêtes sur ses opérations ;

- la *modularité* : « La modularité est la propriété d'un système qui a été décomposé en un ensemble de modules cohérents et faiblement couplés ». Un module peut être vu comme un ensemble d'entités (classes, objets, etc.) liées entre elles. Deux modules sont couplés lorsqu'il existe des liens entre les entités qui les composent ;

- la *hiérarchie* : « La hiérarchie est un rangement ou un ordonnancement des abstractions ». Ce concept est plus communément appelé héritage.

Un langage de programmation qui supporte ces quatre propriétés peut être appelé langage de programmation orienté objets. Toutefois, si l'on compare ce modèle orienté objets au modèle structuré traditionnel, on constate que la seule différence réside dans le concept d'héritage. En effet, les types structurés, les fonctions d'accès sur ces structures et la compilation séparée sont des moyens, certes

[22] Unified Modeling Language en anglais

primaires, de supporter l'abstraction, l'encapsulation et la modularité. Ces principes ont été mis en œuvre par les programmeurs rigoureux dès l'apparition des langages impératifs. En fait, l'apport essentiel de l'approche objets est l'héritage et ses mécanismes associés. Définir une hiérarchie ne consiste pas simplement à ordonner les classes entre elles. Une bonne hiérarchie permet de mettre en commun dans les classes de base, des attributs et des opérations, qui seront automatiquement hérités dans les classes dérivées. Mais l'héritage prend sa véritable dimension dans le principe du polymorphisme. Le polymorphisme est la propriété d'un objet d'une hiérarchie a être considéré indifféremment comme une instance de sa propre classe où d'une instance d'une de ses quelconques classes de base (mais pas de ses classes dérivées). De plus, il possible, pour ces objets polymorphes, de spécialiser les opérations héritées des classes de base. En d'autres termes, il est possible de définir de manière générale le comportement des objets, mais que ces derniers réalisent ce comportement de manière spécifique.

Développer une application en s'appuyant sur le modèle orienté objets ne consiste pas simplement à utiliser un langage de programmation orienté objets. Réussir un développement orienté objets nécessite la mise en œuvre d'un véritable « mode de pensée » orienté objets. On parle alors de méthode orientée objets. Une méthode orientée objets repose sur un principe simple : représenter informatiquement un problème de la même manière que le cerveau humain le perçoit. Ainsi, il est couramment admis que physiologiquement [Encyclopædia], le cerveau humain perçoit les objets de quatre manières différentes :

- objet/attributs : une Ferrari se distingue par sa couleur, son modèle …
- objet/constituants : une Ferrari est composée de quatre roues, d'un moteur …
- objet/classe : une Ferrari est une instance de la classe « Voiture » ;
- classifications : la classe voiture est une sorte de véhicule, au même titre que la classe « Camion ».

La plupart des méthodes orientées objets tiennent compte de ces considérations et proposent, entre autre, des notations adaptées à chacune de ces perceptions (cf. Annexe B).

2.1.2 Qualités logicielles d'une approche orientée objets

Parmi l'ensemble des qualités potentielles d'un logiciel, cinq sont communément admises comme étant critiques : la validité, la robustesse, la compatibilité, la réutilisabilité et l'extensibilité. La validité d'un logiciel peut être définie comme étant sa capacité à satisfaire ses spécifications correctement. La robustesse est sa capacité à résister à des conditions anormales de fonctionnement (comme l'introduction de données erronées par exemple). La compatibilité est la capacité du logiciel à communiquer avec d'autres logiciels. La réutilisabilité correspond à la capacité du logiciel à être réutilisé (entièrement ou en partie) dans de nouvelles applications. Enfin, l'extensibilité est la capacité du logiciel à s'adapter à des changements de spécifications.

La validité et la robustesse sont deux qualités évidemment recherchées coûte que coûte, quelle que soit l'approche de développement ou le langage utilisé. L'utilisation de standard d'échange est un moyen de rendre une application compatible. La réutilisabilté est essentiellement conditionnée par une bonne modularité, c'est-à-dire que les modules sont les plus indépendants possibles les uns des autres. Enfin, l'extensibilité est une qualité relativement facile à obtenir en s'appuyant sur les propriété de l'héritage et du polymorphisme. L'approche orientée objets offre donc un cadre propice au développement de logiciels

respectant ces qualités.

Dans notre cas, les avantages d'utiliser une méthode de développement orientée objets se situent à plusieurs niveaux. Tout d'abord, nous avons précisé dans le chapitre précédent que notre méthode de synthèse d'une forme nécessitait une bibliothèque de formes primitives. Au fur et à mesure des conceptions, il est possible que ces formes évoluent ; par conséquent, il devient nécessaire au concepteur de pouvoir ajouter des formes primitives à la bibliothèque, donc pouvoir en hériter. Par ailleurs, nous avons proposé au début du Chapitre 3 (cf. §2) plusieurs méthodes utilisant des formes en deux ou trois dimensions. Enfin, nous avons proposé plusieurs types de courbes de satisfaction en sachant que leur utilisation dépend principalement du contexte de conception. Pour toutes ces raisons, l'utilisation d'une hiérarchie de classes s'avère nécessaire (utilisation de l'héritage) pour faciliter l'ajout de nouvelles composantes (de formes ou de courbes de satisfaction).

La réutilisation peut se justifier par la nécessité de construire des formes dites combinées à partir de formes primitives. Ceci permet d'éviter de redéfinir les propriétés et les comportements basiques de la nouvelle forme mais simplement d'ajouter les propriétés et les comportements de la combinaison. Par exemple, lors de la définition d'une classe définissant une forme complexe combinant un cercle et un rectangle, la réutilisation évite de spécifier à nouveau les différents paramètres de la nouvelle classe.

Enfin, nous savons que notre application ne représente qu'une maquette informatique ; de ce fait, il sera nécessaire de la faire évoluer vers un logiciel plus conséquent. Ainsi, pour toutes ces raisons, il est nécessaire de développer une maquette informatique à l'aide des principes orientés objets.

2.1.3 Le diagramme des paquets

D'un point de vue général, l'application informatique doit traiter les problèmes principaux suivants :

- la représentation informatique des spécifications, c'est-à-dire modéliser le cahier des charges intermédiaires et la notion de contrainte qui est directement liée. Ceci est réalisé par le paquet « Spécifications » ;
- la représentation de l'espace des solutions ;
- l'estimation des solutions construites et contenues dans l'espace des solutions ;
- la représentation informatique d'une bibliothèque de formes ;
- la représentation d'expressions arithmétiques.

Les dépendances entre les paquets sont représentées dans la Figure 4-1. On remarquera que l'estimation et la construction de l'espace des solutions se font totalement indépendamment l'une de l'autre. Cette organisation permet de réutiliser l'estimation sans avoir recours au paquet « Espace des solutions ». Dans la mesure où le paquet « Spécifications » assure la responsabilité de proposer les solutions prometteuses, il est naturel que ce paquet dépende des paquets « Espace des solutions » et « Estimation ». Rechercher les solutions consiste à transcrire les contraintes intermédiaires du cahier des charges en contraintes terminales sur les familles de formes, d'où la dépendance entre les paquets « Espace de solutions » et « Bibliothèque de formes ». Enfin, les expressions arithmétiques sont des outils de base et sont utilisées par les autres paquets.

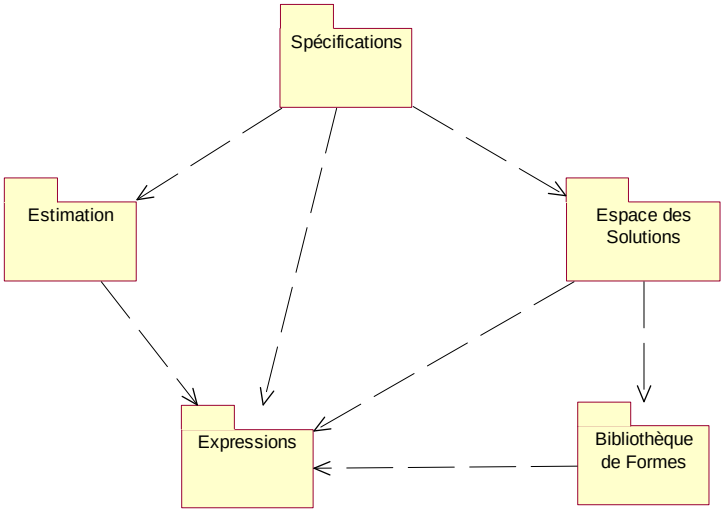


Figure 4-1. *Diagramme des paquets*

2.1.4 Le diagramme des classes

Dans cette partie, nous détaillons les paquets identifiés dans le paragraphe précédent en expliquant les méthodes, les attributs et les classes relatifs à chacun d'eux. Pour chaque paquet, nous proposons un diagramme de classes dans le formalisme UML ; le diagramme de classes complet est disponible en Annexe C.

2.1.4.1 Les classes du paquet « Spécifications »

Comme nous l'avons présenté dans le chapitre précédent, les premières informations nécessaires à la synthèse de formes sont contenues dans le cahier des charges intermédiaires. Ce dernier, représenté par la classe **`TCdC`** (cf. Figure 4-2), est le noyau de l'application. Ses principales méthodes sont :

- **`Construire_Sol`** qui construit automatiquement l'espace des solutions à partir d'un ensemble de contraintes (attribut **`L_Cont`**) et d'une bibliothèque de formes. Pour chaque cahier des charges, un espace de solutions est créé ; c'est pourquoi une liste des intervalles de variation des paramètres terminaux de chaque famille de forme est conservé (attribut **`l_defam`**) ;

- **`Evalue_DS`** estime une forme solution *f* par rapport à l'ensemble des contraintes du cahier des charges intermédiaire : c'est le calcul de $DS_{CdCI}(f)$ (cf. Chapitre 3, §4.1.1). Ce calcul se fait au niveau de la classe **`TCdC`** puisqu'elle est la seule à connaître l'ensemble des contraintes intermédiaires. Toutefois, elle ne permet pas de calculer directement le degré de satisfaction d'une seule contrainte. Cette tâche est réalisée par la méthode **`Evalue`** relative à la classe **`TContrainte`** ;

- enfin, cette classe est aussi la seule à connaître l'espace des solutions, de ce fait, elle est la seule à pouvoir présenter de manière conviviale les solutions les plus prometteuses au concepteur : ceci est réalisé par la méthode **`Morpher`**. Néanmoins, par souci de simplification et de gain de temps, le parcours de l'espace des solutions implémenté se résume à parcourir à l'aide d'un pas régulier les intervalles de variation de chaque paramètre terminal. Il est bien évident que ce parcours tient compte du produit cartésien de ces intervalles ;

- **`Charge`** permet le chargement d'un cahier des charges décrits de manière textuelle dans un fichier.

Nous présentons ultérieurement le format du fichier à utiliser. Dans le futur, cette méthode pourrait interpréter un cahier des charges écrit en langage naturel pour en extraire les contraintes intermédiaires.

La classe ***TCdC*** représente la seule communication entre notre méthodologie de synthèse de formes satisfaisantes et l'interface de dialogue de notre application.

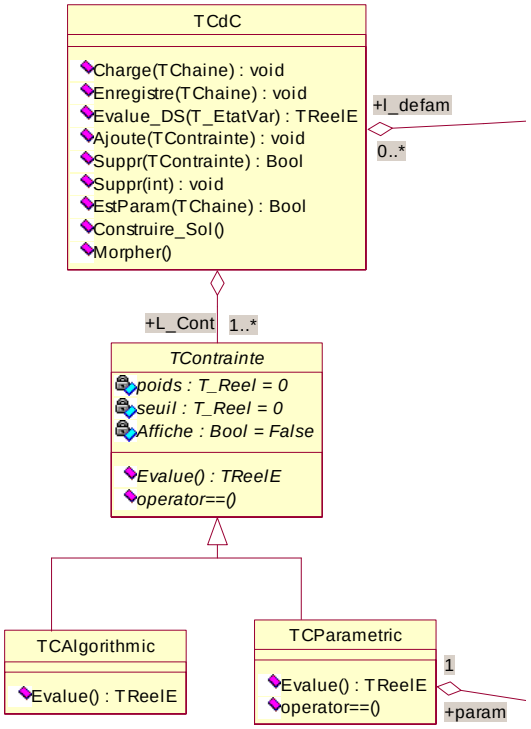


Figure 4-2. *Classes du paquet contraintes*

Dans le chapitre précédent, nous avons supposé que nous traduisions les fonctions en contraintes sur des paramètres intermédiaires. Ceci nous a permis de proposer une modélisation pour l'objet contrainte qui est : <Paramètre, Relation, Expression, Poids> en sachant que les trois premières composantes sont des classes des paquets « Expressions » et « Estimation ». Mais, il est possible que d'autres types de contraintes existent : en l'occurrence, des contraintes de types conditionnelles que nous pourrions appeler « algorithmiques » si la complexité de la contrainte conditionnelle vient à croître considérablement. De ce fait, le diagramme de classes comporte deux sous-classes ***TCParametric*** et ***TCAlgorithmic***. La classe ***TContrainte***, considérée comme la classe de base des deux précédentes, permet de regrouper les propriétés et les méthodes communes à ses sous-classes (ou classes dérivées) qui sont :

- l'importance de la contrainte par rapport aux autres contraintes du cahier des charges, représentée par l'attribut ***Poids*** ;
- la possibilité d'estimer le degré de satisfaction de cette contrainte par rapport à une forme solution considérée. Ceci est modélisé par la méthode virtuelle ***Evalue***.

2.1.4.2 Les classes du paquet « Espace des solutions »

Toujours dans le chapitre précédent, nous avons montré comment les contraintes intermédiaires devaient être traduites (aussi automatiquement que possible) en contraintes terminales sur les paramètres

terminaux de chaque classe de forme. Par ailleurs, nous avons précisé que les contraintes terminales devaient s'exprimer sous la forme d'intervalles. Comme les paramètres terminaux d'une classe de forme sont interdépendants, les bornes des intervalles de variation doivent se représenter à l'aide de deux expressions mathématiques. De ce fait, la classe ***TIntervalle*** (cf. Figure 4-3) modélise un intervalle de variation d'un seul paramètre terminal. Comme dans la grande majorité des cas, le cahier des charges intermédiaire est constitué de plusieurs contraintes, il s'ensuit une multiplicité d'intervalles de variation pour un seul paramètre terminal. De plus, le produit cartésien de plusieurs intervalles peut donner lieu à une discontinuité du domaine de variation d'un seul paramètre. Ainsi, la classe ***TDomDefParam*** permet de modéliser le domaine de variation d'un paramètre (attribut ***param***) en conservant une liste d'intervalles (représenté par l'attribut ***L_Int***).

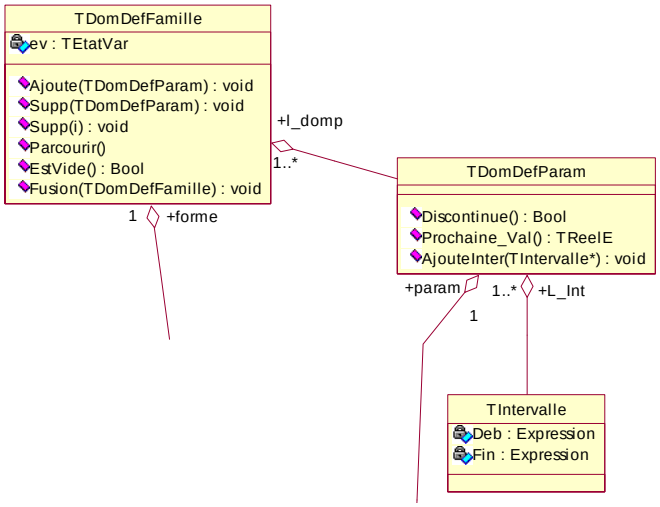


Figure 4-3. *Classes du paquet espace des solutions*

Enfin, la classe ***TdomDefFamille*** permet de conserver le domaine de variation de tous les paramètres terminaux d'une classe de forme par l'attribut ***l_domp***. La principale méthode de cette dernière classe correspond au parcours de l'espace des solutions. En effet, les domaines de variation des paramètres d'une classe de forme correspond à l'espace des solutions relatifs à cette classe de formes.

2.1.4.3 Les classes du paquet « Estimation »

Une fois l'espace des solutions construit, nous avons présenté dans le chapitre précédent une méthode permettant d'estimer une forme par rapport à l'ensemble des contraintes du cahier des charges intermédiaire. Cette méthode consiste à calculer le degré de satisfaction d'une contrainte à partir des valeurs réelles et désirées du paramètre intermédiaire utilisé dans la contrainte. Le calcul se fait en utilisant une courbe dépendante de la relation utilisée dans la contrainte ($\geq$, $\leq$, $\neq$, $=$). La classe ***TRelation*** (cf. Figure 4-4) permet de modéliser les différentes relations mathématiques gérées par l'application. Le seul attribut associé à cette classe est le nom de la relation (attribut ***label***). Par ailleurs, nous avons vu que plusieurs courbes pouvaient être associées à la même relation mathématique : les courbes normales et les courbes améliorées. La classe ***TRelContinue*** permet de représenter différentes méthodes d'estimation pour la même relation mathématique. Par souci de simplification et de gain de temps, les courbes utilisées pour le calcul de l'estimation sont celles définies au paragraphe 4.1.2 du Chapitre 3. De ce fait, les quatre relations mathématiques sont définies par une seule expression mathématique représentée par l'attribut ***exp***. La principale méthode de la classe est représentée par ***Evalue*** qui permet de calculer le degré de

satisfaction d'une contrainte à partir des valeurs réelle et désirée du paramètre intermédiaire.

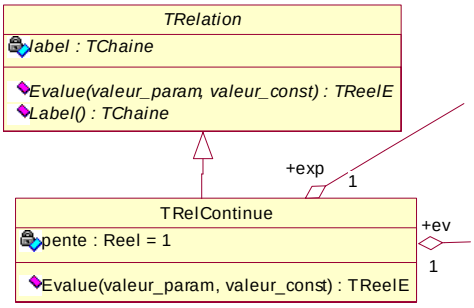


Figure 4-4. *Classes du paquet estimation*

2.1.4.4 Les classes du paquet « Bibliothèque de formes »

La méthode de synthèse de forme que nous avons présenté dans le chapitre précédent consiste à utiliser des formes primitives prédéfinies. Comme toutes ces formes sont supposées utiliser des paramètres terminaux (attribut ***L_Param***), nous définissons une classe virtuelle (ne pouvant pas être instanciée) qui exprime les particularités de toutes les formes primitives : c'est la classe ***TFamilleForme*** (cf. Figure 4-5). De cette classe est dérivée un certain nombre de primitives pouvant être instanciées : ***TRectangle***, ***TCercle***, ***TTriangle***. Nous avons aussi ajouter une classe ***TCombine2D*** qui permet de combiner des formes primitives pour constituer une forme complexe (Chapitre 3, §3.4.4). Néanmoins, cette possibilité n'a pas été plus développée.

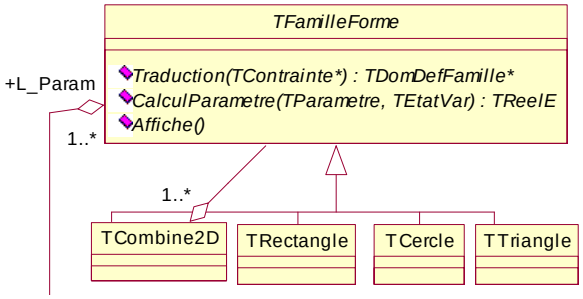


Figure 4-5. *Classes du paquet bibliothèque de formes*

L'autre grande particularité de la classe ***TFamilleForme*** se situe au niveau des méthodes ***Traduction*** et ***CalculParametre***, répercutée aux différentes sous-classes, qui traduisent les règles et les algorithmes de la Figure 3-15 : traduire une contrainte intermédiaire en intervalles de variation des paramètres terminaux propre à la famille de forme et calculer la valeur réelle du paramètre intermédiaire utilisée dans la contrainte à partir des valeurs des paramètres terminaux.

2.1.4.5 Classes du paquet « Expressions »

La classe ***Expression*** de la Figure 4-6 permet de modéliser une expression mathématique ; en réalité, elle se décompose en une hiérarchie de classes plus complexe, mais l'étude de cette hiérarchie sort du cadre de ce manuscrit. Le fait que la classe ***TParametre*** hérite de la classe ***NVariable*** qui elle-même hérite de la classe ***Expression*** permet de définir des expressions mathématiques utilisant des paramètres intermédiaires. L'intérêt se situe au niveau de la définition des contraintes intermédiaires : un paramètre intermédiaire peut être contraint par un autre paramètre intermédiaire défini préalablement dans le cahier

des charges. Enfin, la classe ***TEtatVar*** contient la liste de toutes les variables utilisées dans les expressions mathématiques ainsi que leur valeur numérique associée si elle a été calculée (classe ***TReelE***).

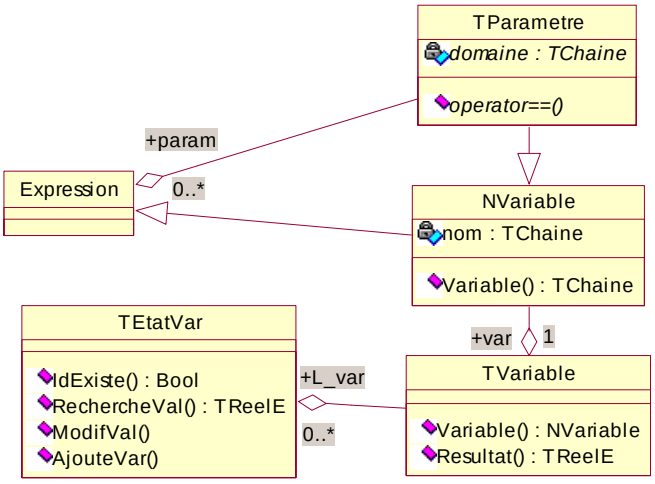


Figure 4-6. *Classes du paquet expressions*

Les algorithmes et équations mathématiques utilisés d'une part pour traduire des contraintes intermédiaires en contraintes terminales et, d'autre part, pour calculer la valeur réelle d'un paramètre intermédiaire à partir de paramètres terminaux ne constituent pas une classe en soi dans la mesure où ils sont directement inclus dans le code source de notre application. Pour bien faire, une classe ***TRègle_Métier*** devrait être définie. De cette façon, elle pourrait se dériver en deux sous-classes ***TTraduit_Contrainte*** et ***Calcul_VR***. Toutefois, cette modélisation aurait nécessité l'interprétation de règles comme cela est fait dans les systèmes experts à base de règles. Or, notre souci est de démontrer la faisabilité de notre approche et non de développer un logiciel complet permettant de passer de contraintes à des formes satisfaisantes. L'intégration des règles dans le code source nous a permis un gain de temps notable.

2.2 Utilisation de la maquette

Dans les parties précédentes, nous avons détaillé la conception orientée objets de la maquette. Dans ce paragraphe, nous présentons son utilisation ainsi que son interface. Le développement a été réalisé en langage C++ en utilisant un outil intégré de développement rapide d'applications fourni par BuilderC++© de la société Borland™. L'intérêt d'un tel outil de développement réside dans la facilité de description de l'interface. Il suffit, en réalité, de sélectionner un composant graphique (bouton, fenêtre de dialogue, image, intitulé …), de le placer dans l'application en cours de construction, et de déterminer ses propriétés. Tout cela se fait interactivement d'une manière très conviviale. L'utilisation de cet outil RAD[23] nous a permis de nous concentrer principalement sur le développement de la méthode plutôt que sur son interface.

Avant d'utiliser notre projet, il est nécessaire de définir le cahier des charges intermédiaire du problème de conception. Comme la maquette ne gère que le paramètre intermédiaire « Périmètre », le cahier des charges ne pourra contenir que le paramètre « Périmètre ». Supposons qu'il soit défini par trois contraintes

[23] Rapid Application Development (RAD) en anglais

intermédiaires qui sont :

- Périmètre > 2 avec un poids de 0,5 ;
- Périmètre < 10 avec un poids de 0,3 ;
- Périmètre ≠ 5 avec un poids de 0,6.

Le fichier texte associé à ce cahier des charges sera construit de la façon suivante (cf. Figure 4-7) : définition des équations des quatre relations mathématiques utilisées, précision du nombre de contraintes suivi de la définition de chacune des contraintes.

```
[Config]
Pente de la relation Egal = 1
Pente de la relation Sup = 1
Pente de la relation Inf = 1
Pente de la relation Dif = 1
Expression de la relation Egal = 1/(1+Pente*((VReel-VDesire)^2))
Expression de la relation Sup = (1+TANH(Pente*(VReel-VDesire)))/2
Expression de la relation Inf = (1+TANH(Pente*(VDesire-VReel)))/2
Expression de la relation Dif = 1-(1/(1+Pente*((VReel-VDesire)^2)))

Nb Contraintes = 3

[Contrainte1]                          [Contrainte2]
Type = Parametrique                    Type = Parametrique
Parametre = Perimetre                  Parametre = Perimetre
Domaine = Physique                     Domaine = Physique
Relation = >                           Relation = <
Expression = 2                         Expression = 10
Poids = 0,5                            Poids = 0,3

[Contrainte3]
Type = Parametrique
Parametre = Perimetre
Domaine = Physique
Relation = <>
Expression = 5
Poids = 0,6
```

Figure 4-7. *Fichier contenant la définition du cahier des charges intermédiaire*

Après avoir défini le cahier des charges, il est nécessaire de le charger dans l'application pour ensuite le traduire en contraintes terminales pour chaque famille de forme et ainsi obtenir les classes de solutions. Ceci se fait simplement en sélectionnant les menus adéquats. Une fois les classes de solutions créées, il est possible de visualiser les différentes solutions (cf. Figure 4-8) en sachant que le système calcule automatiquement les degrés de satisfaction de chaque contrainte intermédiaire et du cahier des charges intermédiaire (degré de satisfaction de la forme).

Enfin, on peut construire les courbes représentant les degrés de satisfaction en fonction de la valeur réelle du paramètre intermédiaire « Périmètre » pour les deux familles de formes cercle et rectangle (cf. Figure 4-9). Ceci peut être fait dans la mesure où il n'existe qu'un seul paramètre intermédiaire. La courbe serait devenue une surface si le cahier des charges avait contenu deux paramètres et une hypersurface de dimension *n* si le cahier des charges était constitué de *n* paramètres intermédiaires différents.

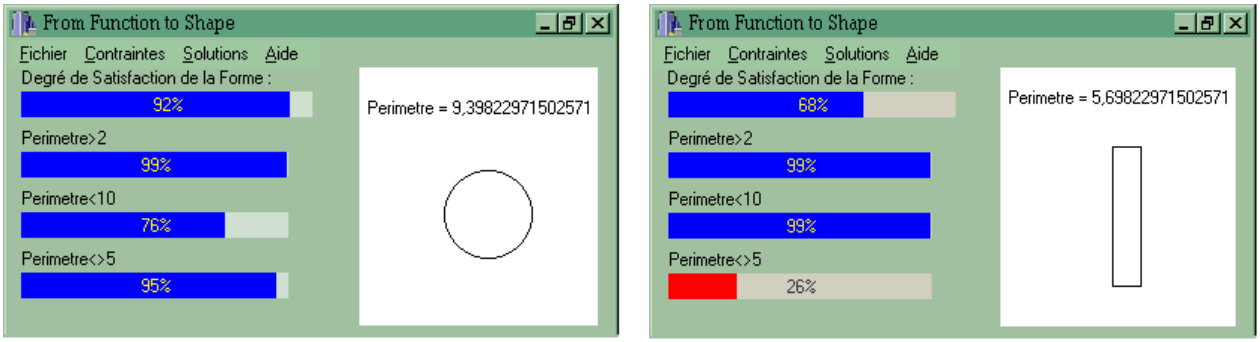


Figure 4-8. *Représentation du degré de satisfaction des différentes contraintes pour une solution donnée*

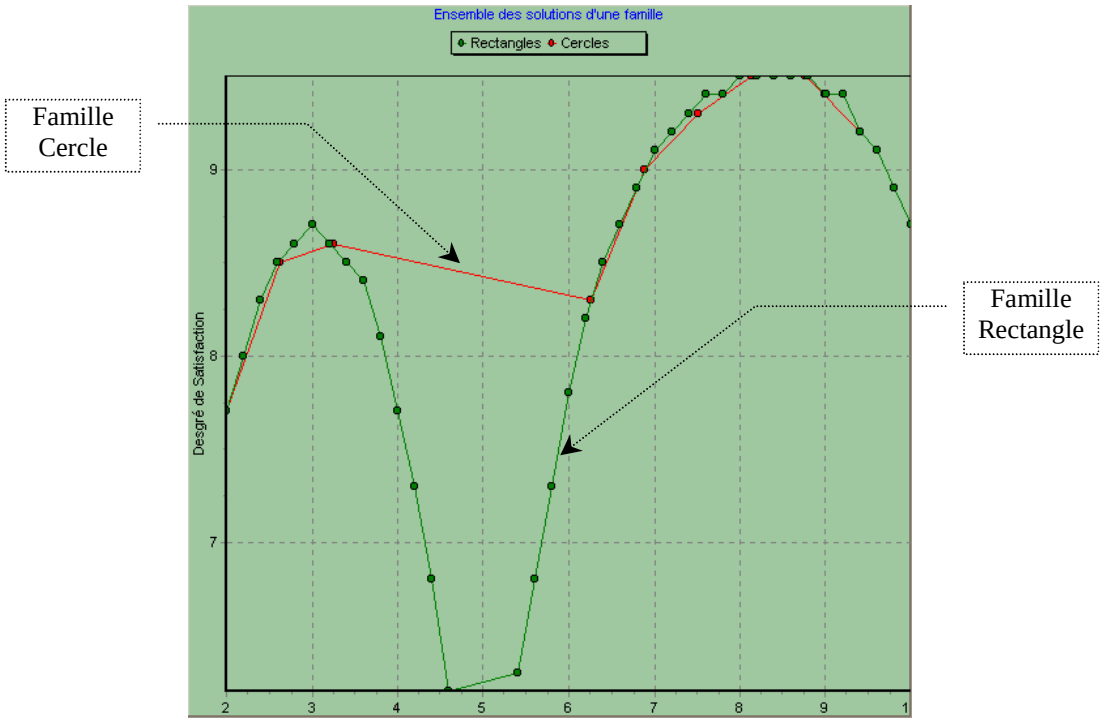


Figure 4-9. *Courbe réelle des degrés de satisfaction des contraintes pouvant servir au calcul des solutions les plus prometteuses*

2.3 Synthèse

La maquette fut réalisée dès l'instant où notre méthode de synthèse de formes avait été définie. L'intérêt de cette maquette a été de démontrer la nécessité de proposer une méthode permettant de calculer le plus automatiquement possible les solutions les plus prometteuses. C'est pourquoi nous avons présenté dans le chapitre précédent une méthode d'optimisation qui calcule à l'aide du degré de satisfaction les meilleures solutions et les proposent au concepteur. Cela lui évite de faire défiler toutes les solutions pour n'en retenir que quelques unes.

Il est vrai que cette maquette n'est pas très représentative d'un problème de conception réel tel que ceux que les concepteurs de voiture, d'avion ou de pièces mécaniques peuvent connaître. Cela est en partie dû au fait que la liste des primitives de formes n'est pas conséquente et n'est pas du tout représentative des différents domaines de conception que l'on vient d'énoncer. Il faut savoir aussi que l'objectif initial n'était

pas de créer un système de conception assisté par ordinateur donnant la possibilité de construire automatiquement une solution de conception à partir de contraintes d'assez haut niveau sémantique. Néanmoins, à partir du moment où les primitives contenues dans la bibliothèque de formes seront plus élaborées, on peut supposer que le système pourra proposer des solutions que le concepteur n'aurait pas imaginées en appliquant une méthode de conception classique (celle qu'il a l'habitude d'utiliser).

Par ailleurs, on peut remarquer qu'aucune règle métier du domaine de conception concerné n'est utilisée. Ceci permettrait de réduire le nombre de formes de l'espace des solutions. Nous étudions cet aspect dans le paragraphe suivant en l'appliquant au domaine de la fonderie.

3 Application industrielle

Dans cette partie, nous présentons une application de notre méthode dans le domaine particulier de la fonderie. Après avoir décrit le problème de conception d'un moule, nous exposons le fait que les problèmes liés à la fonderie peuvent être résolus en utilisant notre méthode de conception. Pour cela, nous expliquons le problème de la fonderie en utilisant notre vocabulaire. Par la suite, nous verrons que l'application de la synthèse d'une forme dans un domaine de conception précis permet de faire évoluer notre méthodologie. Cette partie fait l'objet d'une publication proposée à une revue [Gardan et al.] correspondant à un travail en commun réalisé dans le cadre d'une application industrielle au laboratoire.

3.1 Description du domaine de la fonderie

Connaissant la pièce à fabriquer, le problème du fondeur est de concevoir un moule de fonderie qui permette de fabriquer un maximum de pièces simultanément en respectant scrupuleusement des contraintes de fabrication. Un moule est constitué de plusieurs composants qui sont (cf. Figure 4-10) :

- le bol correspondant au point d'entrée du système de coulé pour le métal en fusion ;
- la pièce à fabriquer ;
- le système de remplissage permettant d'acheminer le métal en fusion du bol jusqu'aux différentes pièces ;
- l'attaque qui est le point de contact entre le système de remplissage et une pièce. Il est possible qu'une pièce soit « attaquée » en plusieurs endroits.

Les contraintes de fabrication devant être respectées sont, par exemple, les distances minimales entre les différents composants. En réalité, le problème du fondeur consiste à placer le maximum de pièces sur le moule de telle sorte que la perte de métal soit minimale et que les pièces fabriquées soient correctes : toutes les cavités doivent être remplies, elles doivent répondre à un certain nombre de propriétés physiques … La perte de métal s'exprime par la « mise au mille » qui correspond au rapport entre le métal utilisé pour les pièces et le métal total fourni au système de coulée du moule : le métal restant dans le système de remplissage ne peut pas être réutilisé dans la mesure où il a refroidi ; il est donc perdu. Par ailleurs, ce problème de conception est complexe dans la mesure où le placement des pièces induit le système de remplissage et vice–versa. Il apparaît donc difficile pour le fondeur de concevoir automatiquement le moule idéal à cause du grand nombre de paramètres intervenant dans la conception.

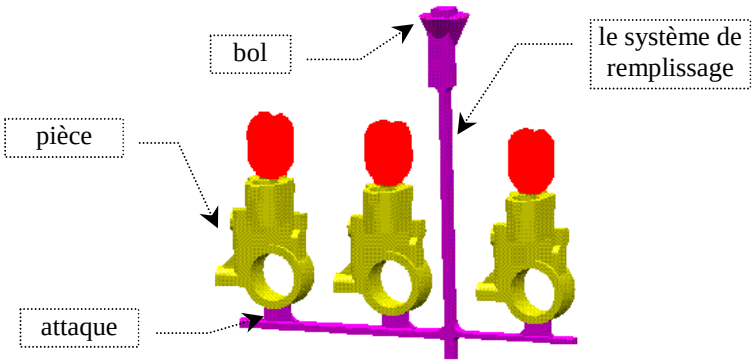


Figure 4-10. *Composition d'un moule en fonderie*[24]

Toutefois, le fondeur ne peut pas se permettre d'étudier (manuellement) les différentes solutions qui s'offrent à lui dans la mesure où il doit tester, pour chaque solution, la vérification des contraintes de fabrication.

De ce fait, dans le but d'obtenir une solution réalisable très rapidement, le fondeur s'oblige à utiliser des règles métier. Il en existe de deux types :

- les règles mécaniques, ne pouvant pas être modifiées, permettent de maintenir les propriétés physiques que les pièces doivent posséder ;
- les règles de savoir faire qui précisent principalement deux choses : les empreintes doivent être remplies simultanément, le métal doit remplir les empreintes avec une vitesse appropriée qui reste à déterminer à l'aide d'équations mathématiques.

Pour satisfaire ces règles très générales, les experts en fonderie conseillent de construire un moule symétrique (cf. Figure 4-11). Une telle pratique facilite les différents concepteurs au dimensionnement des différents composants du moule. Néanmoins, la pérennisation de telles règles de conception va à l'encontre d'une certaine évolution des méthodes de conception. En effet, on constate que l'ensemble des sociétés de fonderie construit des moules similaires. Par ailleurs, les règles métier n'envisagent que très peu de solutions. Or, il est possible que d'autres solutions permettent de construire des moules avec un meilleur rendement tout en conservant les propriétés physiques des pièces à fabriquer. Par conséquent, les règles métier apparaissent trop contraignantes ; il faut donc relâcher certaine d'entre elles pour pouvoir étudier plusieurs solutions de conception. Ceci justifie l'application de notre approche à ce domaine de conception.

[24] image fournie par F. Vexo de l'IFTS–Charleville

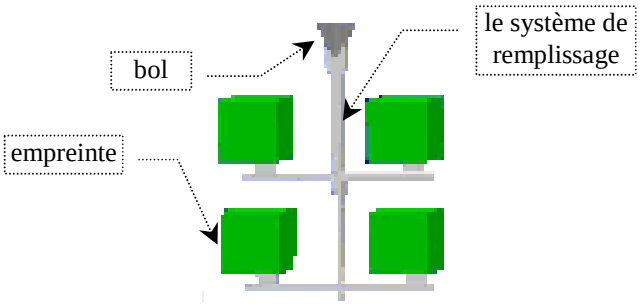


Figure 4-11. *Un moule symétrique*[24]

3.2 Adaptation de notre approche

La créativité d'un outil d'aide à la conception se mesure à sa capacité à proposer des solutions satisfaisantes qui ne sont pas envisagées a priori par le concepteur faute de temps ou de moyens. Notre approche est une méthodologie qui permet de déterminer l'espace des solutions satisfaisantes. Grâce à des heuristiques appropriées, cet espace peut être parcouru pour proposer des solutions prometteuses au concepteur. Nous venons de voir, dans le paragraphe précédent, que l'approche utilisée par les fondeurs conduit à une seule solution : ce qui représente un inconvénient majeur de cette technique. À l'inverse, on note, pour ce domaine de conception précis qu'est la fonderie, deux faiblesses de notre approche :

- la première est la continuité de l'espace des solutions. En effet, la définition des heuristiques permettant d'isoler les solutions optimales parmi une infinité de solutions est un problème compliqué (cf. Chapitre 3, §5.2) ;
- la seconde faiblesse vient de son caractère général. Ne pas tenir compte des connaissances métier d'un domaine particulier permet en théorie de résoudre tous les passages fonction/forme ; mais il est bien évident, qu'en pratique, cette approche n'est pas réaliste.

Cela signifie que l'on peut appliquer notre approche en ajoutant des connaissances métiers. L'ajout de ces connaissances va permettre de réduire l'espace des solutions et de le rendre discret lorsque cela est possible. En effet, ces deux améliorations permettront de rendre réaliste le parcours de l'espace de solutions, certes restreint mais encore suffisamment vaste pour contenir des solutions inattendues pour le concepteur et probablement meilleures.

3.2.1 Correspondance de vocabulaire entre les deux approches

Le système de fonderie est constitué de formes intangibles telles que les empreintes des pièces à fabriquer et de formes non déterminées telles que le placement des empreintes et le système de remplissage. Ce dernier peut être calculé à partir des paramètres terminaux suivants :

- position du bol ;
- nombre de moules à placer ;
- position relative des moules.

On fait le choix de ne pas remettre en cause la détermination du système de remplissage, basée sur des connaissances d'un expert. On pourrait bien entendu le faire en suivant la même démarche, mais cela n'est

pas utile dans ce contexte. Il est cependant indispensable d'en tenir compte pour le placement des empreintes.

En considérant le vocabulaire que nous avons introduit dans les précédents chapitres, les fonctions du cahier des charges à satisfaire sont de construire un moule qui minimise le temps de fabrication d'une série de pièces et minimise le coût de chaque pièce. Les données du problème sont la géométrie de la pièce et la quantité de pièces à fabriquer. L'étude de ce cahier des charges client, dans le contexte de la fonderie, permet d'obtenir un cahier des charges intermédiaire où les contraintes intermédiaires décrivant les fonctions initiales font intervenir la « mise au mille », le temps de cycle … Par exemple, on peut obtenir :

- mise au mille = 1 ;
- temps cycle ≥ 5 secondes ;
- temps cycle ≤ 12 secondes.

Un examen des règles métiers qui permettent de réduire ou de discrétiser l'espace des solutions permet d'appliquer la démarche théorique avec :

- position du bol : son positionnement est limité par les paramètres des moyens de fabrication. Le nombre de positions à examiner peut être restreint. En effet, seules quelques positions clés sont significatives pour la topologie du système de remplissage et un déplacement faible autour de ces positions n'influence pas celui-ci ;
- le nombre de pièces : la borne minimum est bien entendu égale à 1. Un poids maximum de métal est dans tous les cas défini pour un système de production donné. On obtient donc en fonction du volume de chaque pièce un nombre maximum N_1 de pièces. En tenant compte de contraintes (zones libres autour des pièces …) un calcul simple fonction de la surface utile de la plaque et de la surface d'une pièce donne un nombre maximum N_2 de pièces. La borne supérieure est le minimum de N_1 et N_2. Cet espace de solutions est naturellement discret ;
- la position des pièces : on peut considérer l'orientation de la pièce par rapport à un repère absolu de la plaque et sa position dans ce repère. Théoriquement, l'orientation peut être choisie entre 0 et 360 degrés. Suivant la géométrie de la pièce, un expert est capable de déterminer quelques orientations a priori intéressantes, ce qui rend l'espace discret (cf. Figure 4-12). Le placement de l'espace des pièces se fonde sur quelques règles métiers, en particulier la contrainte « en croix » (moule symétrique), qui réduit l'espace des solutions. En effet, pour des raisons d'homogénéité du remplissage, des symétries simplifient le problème.

Malgré ces simplifications, il reste une infinité de solutions pour le placement des pièces. L'espace de solutions, tout en ayant été réduit, reste continu. Le cas le plus difficile de parcours des solutions est indiscutablement celui du placement des pièces (espace continu). On va donc chercher à déterminer une stratégie, fondée sur des connaissances métiers, qui le rende possible et qui converge rapidement vers des solutions prometteuses (c'est-à-dire pas obligatoirement la meilleure, mais parmi les meilleures).

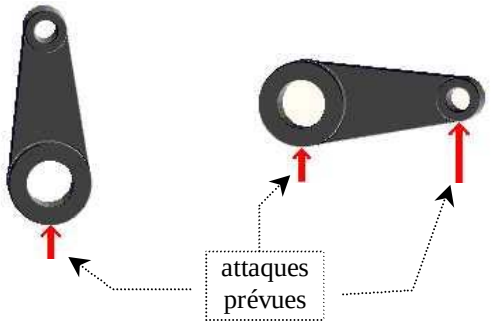


Figure 4-12. *Positionnement de pièces par un expert*[24]

On est capable automatiquement de déterminer, étant donné un nombre de pièces, des familles (topologies) de solutions. La Figure 4-13 présente des exemples de topologies obtenues pour le placement de six pièces. Ainsi, les classes de solutions sont identifiées. Sans valuer tous les paramètres (ou éventuellement en prenant des valeurs moyennes), il est possible de caractériser les classes de solutions les plus prometteuses (que l'on peut classer). On a alors franchi une première étape qui permet de proposer des solutions, éventuellement inattendues pour l'expert. On peut lui laisser le choix ou aller plus loin dans la détermination des formes. Dans la classe de solutions la plus prometteuse, il est possible de présenter le placement le plus prometteur en minimisant les longueurs des canaux. Cette règle donne normalement la meilleure solution.

Figure 4-13. *Exemple de trois topologies différentes pour six pièces*[24]

À partir du moment où des solutions peuvent être proposées, il est important, comme nous l'avons vu dans le chapitre précédent, d'estimer ces solutions pour pouvoir les comparer. Pour cela, il est indispensable de calculer les valeurs réelles des paramètres intermédiaires. Le calcul de la « mise au mille » est immédiat à partir des volumes du système de remplissage. Par ailleurs, la simulation du système de fonderie permet de calculer un certain nombre de paramètres intermédiaires : solidification, remplissage … Ayant les valeurs des paramètres intermédiaires, on peut calculer automatiquement le degré de satisfaction de la solution en se basant sur le cahier des charges intermédiaires et les courbes de satisfaction associées aux relations mathématiques. Toutefois, la simulation du moule prend beaucoup plus de temps que le calcul de la mise au mille. Ainsi, pour estimer une solution, on donnera priorité au calcul de la mise au mille puisqu'il est simple et immédiat mais aussi parce que le poids affecté à la contrainte intermédiaire correspondante est élevé. Cette utilisation joue le rôle d'une heuristique de parcours de l'espace des solutions.

3.2.2 Déroulement de la méthode dans le cas de la fonderie

Connaissant la géométrie de la pièce à fabriquer, on fait appel à un expert du domaine de la fonderie

pour obtenir les quelques orientations et les attaques possibles pour le remplissage de la pièce. Ces données doivent permettre à l'outil de proposer les solutions les plus prometteuses. Les étapes générales du traitement, appliquant notre méthode adaptée à la fonderie, sont les suivantes :

- détermination du nombre maximum de pièces à placer sur la plaque du moule ;
- recherche des topologies (classes de solutions ou classes de formes) pour chaque nombre de pièces possible. Un algorithme permet de trouver toutes les topologies admissibles ;
- pour chaque classe (ou topologie), construction d'un système de remplissage complet en fonction des quelques positions significatives du bol ;
- grâce à l'heuristique de parcours correspondant à la mise au mille, on ordonne les classes de formes de la plus prometteuse à la moins prometteuse ;
- il est théoriquement facile d'estimer par simulation chaque classe de formes, en suivant l'ordre obtenu par le calcul de la mise au mille. Cependant, dans la mesure où la simulation est coûteuse (au moins en temps), il est préférable de laisser au concepteur le soin de choisir la solution qui devra être simulée. Néanmoins, la simulation a besoin d'un positionnement précis (d'une instanciation). Pour effectuer cette instanciation, on utilise le positionnement qui donne la meilleure estimation de mise au mille.

On peut voir à la Figure 4-14 plusieurs résultats obtenus avec l'application de notre méthode.

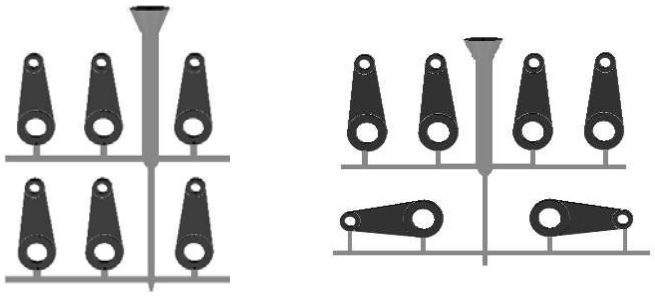


Figure 4-14. *Résultats obtenus en utilisant notre méthode*[24]

3.3 Synthèse

Dans la plupart des métiers, on constate que par formation ou par habitude (pérennisation des méthodes de conception), les concepteurs aboutissent en général à certaines familles de solutions qui deviennent peu ou prou des standards. Par extraction des connaissances utilisées et leur modélisation, il est relativement facile de créer des systèmes de CAO qui implémentent les méthodes utilisées. Ce système aboutit donc naturellement à des solutions du même type. Nous avons appliqué une telle approche dans le cadre du système de fonderie. Cela nous a permis de valider notre approche, que l'on peut qualifier de théorique, en tenant compte des spécificités du domaine de la fonderie. Nous avons montré que cette approche était applicable à ce cas et que l'on obtenait rapidement les mêmes résultats que les experts, voire de meilleurs puisque l'on étudie plusieurs alternatives que le fondeur n'a pas le temps ou les moyens d'approfondir.

4 Conclusion

L'objectif de ce chapitre était de démontrer la faisabilité de l'approche exposée au Chapitre 3. Pour cela, nous avons présenté une maquette informatique implémentant directement notre approche et une application plus industrielle dédiée au domaine de la fonderie.

Dans le cadre de l'application informatique, nous avons détaillé la conception orientée objets de notre logiciel, c'est-à-dire la description des différents diagrammes : de paquets, de classes et de séquences pour quelques algorithmes essentiels au passage fonction/forme. L'intérêt de ce développement fut de montrer la nécessité de définir une méthode de recherche des solutions les plus prometteuses. Il faut bien comprendre que la détermination de la meilleure solution n'est pas adaptée puisque l'on se situe aux phases amonts de la conception de produits. Cela signifie que des solutions moins prometteuses (pour le système informatique) peuvent, moyennant quelques modifications par le concepteur, devenir plus prometteuses que les précédentes ; d'où l'intérêt de les signaler au concepteur à l'aide d'un outil de dialogue adapté qu'il reste à définir.

Suite à ce développement informatique, nous avons montré que notre approche, aussi théorique soit elle, pouvait s'appliquer à un domaine de conception privilégié, la conception de moule de fonderie. Du fait que l'approche théorique possédait un caractère très général, donc difficilement utilisable en pratique, il a fallu l'adapter au domaine de conception concerné. Cette adaptation s'est matérialisée, entre autres, par l'utilisation de règles métiers permettant ainsi de réduire l'espace des solutions à considérer sans pour autant contraindre le système informatique de telle sorte qu'il ne fournisse qu'une seule solution (méthode manuelle utilisée par les fondeurs).

Enfin, la réalisation pratique de l'approche théorique dans un domaine bien précis nous a ouvert de nouvelles voies de recherche : par exemple, estimer une classe de solutions plutôt qu'une instance est un grand pas en avant. De plus, l'utilisation d'un algorithme de placement des empreintes permet de construire automatiquement les classes de solutions.

Bilan et perspectives

Pour conclure, nous établissons un bilan des avancées dans le domaine de l'assistance aux premières phases de conception, qu'elles soient manuelles ou automatiques. Nous en déduisons les points sur lesquels nous nous sommes concentrés et poursuivons en résumant nos propositions qui consistent essentiellement en une proposition d'une méthode de passage des fonctions à des formes. Nous terminons par une présentation de quelques perspectives pour montrer que le passage des fonctions à des solutions géométriques reste encore un problème ouvert. En effet, on est loin encore du logiciel de conception qui construira automatiquement une voiture directement à partir de ses spécifications fonctionnelles. Nos travaux représentent une étude exploratoire pour la synthèse de forme dont on peut penser qu'elle est une première réflexion pour de nombreux autres travaux du laboratoire.

L'étude de l'existant nous a montré qu'il existait effectivement des outils ou des méthodes permettant de traiter des informations purement fonctionnelles. Toutefois, les méthodes restent à un niveau sémantique élevé, proposant au concepteur une démarche « manuelle » pour le développement du niveau fonctionnel mais ne l'assistant pas dans la synthèse des formes associées. L'automatisation de ces méthodes, manipulant essentiellement du langage naturel, ne deviendra envisageable que lorsque l'interprétation sémantique automatique du langage naturel sera fiabilisée. En conséquence, les outils fournissent trop peu de tâches automatisées au concepteur : ils se limitent à des éditeurs de formalismes permettant de manipuler de manière conviviale les informations que le concepteur aura obtenues de manière manuelle.

La qualité des assistances aux premières phases de conception est également liée à la richesse du modèle produit. Idéalement, il faudrait conserver toutes les informations fonctionnelles, les informations plus « physiques » telles que la géométrie de ses composants puisque cette dernière reste un moyen universel de communication entre les divers domaines de conception et les liens qui les unissent. La modélisation des informations géométriques étant relativement maîtrisée, on se concentre actuellement sur la modélisation des informations fonctionnelles. Nous avons recensé les classifications proposées dans la bibliographie et avons proposé notre propre typologie ; ce type d'étude devrait favoriser une meilleure représentation informatique des fonctions. Comme l'objectif est de déduire le plus naturellement possible les

informations structurelles du produit à partir de ces notions conceptuelles que sont les fonctions, nous insistons sur un exemple de hiérarchie fonctionnelle que nous pensons prometteuse et qui pourrait, dans le futur, devenir le noyau des logiciels de conception assistée par ordinateur : le modèle FBS. Malheureusement, trop peu de tâches y sont automatisées au niveau fonctionnel.

Les remarques précédentes concernent essentiellement les conceptions au cours desquelles la part dévolue à l'imagination et à l'innovation est importante. Dans un contexte routinier, dans lequel la hiérarchie fonctionnelle et la structure du produit n'évoluent que très peu, les possibilités d'automatisation sont bien plus importantes puisque le produit a déjà été conçu (étudié, testé …) préalablement.

Puisque l'objectif de notre travail est de synthétiser la forme du produit, nous nous intéressons dans le deuxième chapitre aux méthodes de construction géométrique existantes. Nous rappelons les modèles géométriques les plus communs (B–Rep, G–Cartes, CSG, hybride) pour constater que les informations manipulées sont d'un faible niveau sémantique. D'autres modes de construction existent : construction à l'aide de caractéristiques de formes, qui ne font que regrouper des entités de bas niveau, construction basée sur l'esquisse qui tente d'interpréter une vision floue de la géométrie imaginée par le concepteur, ou encore ce que nous avons appelé les méthodes fonctionnelles, qui ont le mérite de conserver les informations ayant motivé la forme de l'objet. Ces dernières méthodes ont du mal à automatiser la tâche de conception et lorsqu'elles aboutissent partiellement à leur objectif, elles sont limitées à un domaine précis et génèrent beaucoup de solutions. Cet état de fait oblige à modéliser l'ensemble des solutions, que l'on appelle espace des solutions, pour y rechercher les solutions les plus prometteuses. En effet, il est inconcevable de présenter toutes les formes obtenues au concepteur ; celui-ci n'en attend que quelques unes. Le choix de quelques solutions parmi plusieurs implique deux choses : d'une part, parcourir l'espace des solutions et d'autre part comparer une solution avec les autres, d'où la nécessité de définir l'estimation d'une solution par rapport aux spécifications fournies par le concepteur. Enfin, au vu de la complexité d'automatiser la tâche qui permet de passer des fonctions à des formes solutions, au vu des limitations des processus automatiques et les inévitables insuffisances des bases de connaissances, on propose parfois de n'automatiser que partiellement le processus et de faire intervenir le concepteur pour lui demander de modifier, localement ou globalement, les formes proposées par le système afin d'en synthétiser de meilleures.

Nous concluons de cela qu'un passage automatique et direct du niveau fonctionnel au niveau géométrique représente pour l'instant une trop grande difficulté. Nous proposons donc dans le Chapitre 3 d'établir un niveau de difficulté médian qui se traduit par un cahier des charges intermédiaire. Nous faisons l'hypothèse que ce dernier est obtenu manuellement par traduction des spécifications du produit et nous nous concentrons sur sa traduction automatique en une ou plusieurs formes. Il est constitué d'un ensemble de contraintes dites intermédiaires faisant intervenir des paramètres dits intermédiaires. Ces paramètres sont des grandeurs quantifiables mais restent à un niveau sémantique élevé. Notre approche fait abstraction du domaine de conception même si certaines informations, indispensables au bon fonctionnement du passage fonction/forme, doivent être définies par le concepteur. Toutefois, la notion de « périmètre », par exemple, reste la même quel que soit le domaine de conception considéré. Il y a donc capitalisation des connaissances et, au fur et à mesure des conceptions, il devient de moins en moins nécessaire de consulter le concepteur dans cette phase. Les informations fournies par le concepteur

permettent, entre autres, la traduction des contraintes du cahier des charges en intervalles de variation des paramètres géométriques de formes provenant d'une bibliothèque (paramètres terminaux). Le produit cartésien des intervalles définit l'espace des solutions. Dans un premier temps, nous avons supposé que les formes contenues dans la bibliothèque étaient des formes primitives. On montre que l'ajout de formes plus élaborées, c'est-à-dire combinées, ne modifie pas l'approche proposée ; toutefois, le nombre de paramètres associés aux formes combinées s'en trouve considérablement augmenté.

L'espace des solutions, étant un produit cartésien d'intervalles réels, est infini. Comme nous l'avons précisé précédemment, il est nécessaire de choisir les solutions les plus prometteuses parmi cette infinité. Pour cela, nous proposons une méthode d'estimation des formes solutions par rapport aux contraintes du cahier des charges intermédiaire. Cette estimation permet d'obtenir dans un premier temps le degré de satisfaction d'une contrainte intermédiaire par une forme solution et dans un deuxième celui du cahier des charges intermédiaire en calculant une moyenne pondérée. Cette moyenne se fait grâce à l'affectation d'un poids à chaque contrainte qui correspond à sa prépondérance sur les autres contraintes du cahier des charges. Comme une solution peut être estimée, il nous importe, désormais, de choisir les solutions qui conviennent le mieux au concepteur. Pour cela, nous étudions l'application de deux algorithmes de parcours et nous en proposons deux autres plus adaptés à notre modélisation de l'espace des solutions. Enfin, nous établissons différentes possibilités d'intervention du concepteur adaptées à notre méthodologie.

Afin d'étayer notre approche, nous détaillons quelque peu, dans le dernier chapitre, une maquette informatique qui illustre la faisabilité du passage des contraintes de haut niveau sémantique à des formes simples. Enfin, parce qu'à ce jour nous sommes plus attachés à l'aspect méthodologique qu'à l'aspect programmatoire, nous appliquons notre méthode à la conception de moules de fonderie. Cette expérience a conduit à certaines adaptations et a ouvert deux principales perspectives de recherches. Nous les exposons dans ce qui suit et terminons par des projets à plus longue échéance.

Premièrement, nous avons vu que l'utilisation d'un algorithme de placement automatique, dans le cas de la conception d'un moule de fonderie, nous a permis de construire les classes de solutions à partir desquelles notre approche construit les formes solutions. Une évolution de notre méthodologie consisterait donc non pas à conserver les classes de formes dans une bibliothèque mais à construire automatiquement ces classes en fonction du domaine de conception concerné. L'idée serait d'élaborer un deuxième modèle intermédiaire, entre le cahier des charges intermédiaire et l'espace des solutions et, de ce fait, permettrait un passage moins brusque encore entre les fonctions et les formes.

Suivant notre méthodologie, l'estimation nécessite l'instanciation d'une solution, c'est-à-dire l'attribution d'une valeur réelle à chaque paramètre terminal de la solution. La deuxième amélioration consisterait à estimer une classe de solutions plutôt qu'une solution. Cette possibilité permettrait, par exemple, de construire en premier lieu les classes satisfaisantes, desquelles les meilleures seraient tirées. Ensuite, dans les classes les plus prometteuses, on choisirait les solutions les plus prometteuses. On pourrait ainsi examiner bien plus de solutions. Une réalisation de cette amélioration, sans pour autant être la meilleure, reviendrait (cf. Chapitre 4) à calculer les valeurs moyennes de chaque paramètre terminal de la classe de solution. Toutefois, cela implique à nouveau un parcours d'une partie de l'espace des solutions.

Les perspectives dans un cadre plus général sont nombreuses et font bien souvent appel à d'autres

domaines de recherche. En particulier, les travaux effectués en intelligence artificielle pourraient servir de base pour une meilleure compréhension de la sémantique des informations fonctionnelles fournies par le concepteur. Cela permettrait au système informatique de fournir une assistance plus précoce pour le passage fonction/forme ; par exemple, un passage plus automatique du cahier des charges du client au cahier des charges intermédiaire. Un traitement automatique à un niveau fonctionnel ne peut se faire que si l'ensemble des fonctions utilisées lors de la définition des spécifications a une représentation informatique. Même si nous avons identifié une bonne partie des fonctions, il n'en demeure pas moins qu'une modélisation adéquate reste complexe. De plus, cette modélisation seule ne suffit pas : il faut y associer des méthodes de construction, de transformation, d'identification …

Dans le manuscrit, on se consacre essentiellement à la synthèse de formes simples. Les quelques propositions faites en début du troisième chapitre pour la synthèse de formes combinées sont autant de perspectives. Comme le laboratoire dans lequel s'est déroulé ce doctorat mène une étude sur la conception par les caractéristiques de forme, nous aurons l'opportunité d'explorer plus particulièrement celle qui consiste à conjuguer un algorithme génétique avec un modèle de conception par les caractéristiques de formes. Rappelons que l'idée générale consiste à associer à chaque fonction un ensemble de surfaces fonctionnelles (caractéristiques de forme) susceptibles de la réaliser et à engendrer un grand nombre de solutions que l'on fait se croiser avec l'espoir de faire progresser la qualité des solutions au fil des générations (cf. Chapitre 3, §2.1).

La qualité de l'interface homme–machine est une notion essentielle à un logiciel de conception apprécié de ses utilisateurs. Or, nous avons déjà dit que le mécanisme de synthèse de forme devra très certainement être assisté par le concepteur. Nous envisageons de faire défiler une sélection de solutions à l'écran et de demander au concepteur d'intervenir quand un aspect de la forme lui convient ou lui déplaît.

Si les solutions se succèdent dans un ordre quelconque, l'opération sera particulièrement longue et pénible. Pour la rendre conviviale, nous prévoyons de présenter les solutions de sorte qu'elles présentent une continuité géométrique et que leur défilement apparaisse comme une animation. Ceci présente l'avantage de faire évoluer les solutions de manière progressive et de laisser le temps au concepteur d'intervenir lorsqu'une zone intéressante ou mauvaise se dessine. Pour assurer cette continuité géométrique, il faut soit mettre au point un algorithme de parcours de l'espace des solutions capable de trouver une solution ressemblant à une autre solution, soit avoir recours à un algorithme de morphing géométrique pour passer progressivement d'une solution à la suivante lorsqu'elles sont dissemblables ou lorsque l'algorithme précédent a échoué. Ceci fait cependant encourir le risque de générer des formes extérieures à l'espace des solutions.

Quand sera résolu le problème du défilement fluide des solutions, il restera à prendre en compte les interventions de l'utilisateur. Ceci soulève d'importantes difficultés de désignation des zones concernées, de leur interprétation d'un point de vue fonctionnel et de leur prise en compte dans le mécanisme de parcours de l'espace des solutions (s'il a été entièrement calculé au préalable) ou de génération (si l'espace est construit au fur et à mesure du parcours).

Ces propositions devraient permettre de converger plus rapidement vers la solution idéale.

Annexe A Introduction aux ensembles flous

1 Introduction

La logique floue, ou plus généralement le traitement des incertitudes, a pour objet d'étude la représentation des connaissances imprécises et le raisonnement approché [Gacôgne 97]. On peut donc la situer à côté des heuristiques de résolutions de problèmes, des systèmes experts, de l'apprentissage, de l'intelligence artificielle distribuée et même du traitement de la langue naturelle [Desmontils 96], domaines qui sont des techniques d'intelligence artificielle au sein des sciences cognitives.

Dans les problèmes de prise de décision, d'aide au diagnostic et plus généralement dans tous les systèmes à base de connaissances, on souhaite, à partir d'observations, parvenir à une conclusion qui peut être la détermination d'un objet ou une action à entreprendre.

Tous les problèmes concrets sont, en fait, confrontés aux notions d'incertitude et d'imprécision. Ces deux notions sont habituellement mêlées et c'est essentiellement l'observation statistique qui induisait, jusqu'à présent dans la pratique, la mesure probabiliste des incertitudes.

2 Quelques concepts

Dans ce qui suit, nous présentons les diverses notions liées aux ensembles flous (cf. [Gacôgne 97]) :

Imprécision :

C'est ce qui est relatif au contenu d'une proposition « mesurer environ 1,75m » (considéré comme flou), « avoir entre 20 et 25 ans » (considéré comme non flou), on ne donne pas de valeur précise mais un intervalle, une « fourchette ». Ainsi, ce qui était couramment appelé incertitude probabiliste ou intervalle de confiance (cas par exemple d'un poids donné à 10g près) est une imprécision.

Incertitude :

C'est un coefficient apporté au fait qu'une proposition soit vraie ou fausse. Dans « il est possible qu'il soit grand », le prédicat « grand » est considéré comme vrai ou faux sans aucune nuance, mais c'est la

proposition qui est douteuse. Par ailleurs, donner un degré de vérité situé entre 0 pour le faux et 1 pour le vrai constitue une incertitude probabiliste.

Ensembles flous :
Il se peut que les ensembles où les variables peuvent prendre leurs valeurs aient des frontières mal définies : c'est bien évidemment le cas de l'exemple « grand ». La notion d'ensembles flous permet de prendre en compte cette nouvelle notion. Nous détaillons un peu plus ce concept dans la suite.

Prédicats vagues :
Dans « être jeune », « il a environ 15 ans », le prédicat est quantitatif mais mal défini et dans « l'ambiance est bonne », il est qualitatif ou difficilement quantifiable. En fait, l'opposition classique entre qualitatif et quantitatif n'est pas elle-même bien définie.

Comparaisons floues :
Lorsqu'on exprime « x est bien plus petit que y », on peut se ramener à des prédicats vagues unaires par des définitions telles que x, y voisins si et seulement si le rapport $\frac{x}{y}$ est environ égal à 1.

Modificateur de prédicat :
Dans le cas « il est presque grand », il s'agit d'une proposition certaine dans laquelle « presque » peut être regardé comme un modificateur du prédicat « grand ». « pas très grand » ou « très grand » sont aussi d'autres prédicats obtenus à partir de « grand ».

Connecteur flou :
Dans les propositions « il est presque grand » ou « il est grand est presque vrai », nous avons deux concepts complètements distincts. Le premier modifie un prédicat ; le second modifie la valeur de vérité de la proposition, ce pourrait être un connecteur au même titre que la négation.

3 Les ensembles flous

La théorie des ensembles flous est une théorie mathématique dont l'objectif est de modéliser les notions vagues du langage naturel pour pallier l'inadéquation de la théorie des ensembles classiques dans ce domaine [Tong-Tong 95]. La caractéristique fondamentale d'un ensemble classique est la frontière abrupte entre les éléments qui appartiennent à l'ensemble et ceux qui ne lui appartiennent pas. Le concept d'appartenance est modélisé par une fonction à valeur dans l'ensemble à deux éléments {0,1} dans le cas de ces ensembles classiques. Pour ce qui est des ensembles flous, la fonction d'appartenance a ses valeurs dans l'intervalle [0,1].

3.1 Définitions

Soit un ensemble de référence U (l'univers), on définit un ensemble flou A dans U par la donnée d'une fonction d'appartenance μ_A de U dans l'intervalle réel [0,1]. À tout élément $x \in U$, on associe une valeur

$\mu_A(x)$ telle que :

$$\begin{array}{lccl} \mu_A & : & U & \mapsto & [0,1] \\ & & x & \rightarrow & \mu_A(x) \end{array}$$

À tout élément x de U la valeur associée n'est pas nécessairement égale à 0 ou à 1, elle est a priori quelconque et désigne le degré d'appartenance de x à l'ensemble A.

Pour tout ensemble flou A d'un univers U, on donne les définitions suivantes :

Noyau	$N(A) = \{x \in A \ / \ \mu_A(x) = 1\}$	Les éléments qui sont « vraiment » dans A.
Support	$S(A) = \{x \in A \ / \ \mu_A(x) \neq 0\}$	Ceux qui appartiennent à A à des degrés divers.
Hauteur	$h(A) = sup_{(x \in U)}\{\mu_A(x)\}$	C'est le plus grand degré d'appartenance à A.
α–coupe	$A_\alpha = \{x \in U, \alpha \in [0,1] \ / \ \mu_A(x) \geq \alpha\}$	C'est l'ensemble des valeurs qui ont un degré d'appartenance au moins égal à α.
α–niveau	$A^\alpha = \{x \in U, \alpha \in [0,1] \ / \ \mu_A(x) = \alpha\}$	C'est l'ensemble des valeurs qui ont un degré d'appartenance égal à α.

3.2 Remarques

1. $N(A) \subseteq S(A) \subseteq U$
2. Il est possible que la hauteur de A ne soit jamais atteinte. Par exemple, si la fonction d'appartenance est définie par la fonction tanh(x) sur $\mathbb{R}^+$, la limite quand x tend l'infini de *tanh* est égale à 1, donc la hauteur sera égale à 1 ; mais aucune valeur de x n'aura de degré d'appartenance égal à 1.
3. $A_0 = S(A)$ si on considère une α–coupe stricte (i.e. $\mu_A(x) > \alpha$).
4. $A' = N(A)$

3.3 Opérations sur les ensembles flous

Les opérations sur les ensembles flous sont, en général, des extensions des opérations bien connues sur les ensembles classiques (égalité, intersection, union, complément …). Cependant, d'autres définitions, en l'occurrence pour l'union et l'intersection, ont été proposées par divers auteurs. Elles sont parfois préférables aux opérations d'origine dans certains développements. L'intersection, quelle que soit sa définition, est généralement appelée une *norme triangulaire* ou *t-norme*, alors que l'union est appelée *conorme triangulaire* ou *t-conorme*.

On dit que deux ensembles flous définis sur le même univers U sont égaux si et seulement si leurs fonctions d'appartenance sont égales en tout point de U.

L'union (resp. l'intersection) de deux ensembles flous A et B de même référentiel U est un ensemble flou noté $A \cup B$ (resp. $A \cap B$) dont la fonction d'appartenance peut se définir par (selon [Zadeh 65]) :

$$\begin{array}{lccl} \mu_{A \cup B} \text{ (resp. } \mu_{A \cap B}) \ : & U & \mapsto & [0,1] \\ & x & \rightarrow & max\{\mu_A(x), \mu_B(x)\} \text{ (resp. } min\{\mu_A(x), \mu_B(x)\}) \end{array}$$

4 Conclusion

L'objectif de cette annexe n'est pas de fournir un cours sur les ensembles flous mais de proposer au lecteur les bases théoriques liées à cette notion datant des années 60 et pourtant relativement mal connue. La logique floue connaît depuis une dizaine d'année seulement des applications pratiques dans divers domaines liés à une automatisation de la prise de décision. À l'heure actuelle, des circuits intégrés contiennent des microprocesseurs dédiés à la logique floue.

La présentation de cette notion se justifie surtout par le fait que la logique floue peut être considérée comme un modèle mathématique de notre approche d'estimation de formes par rapport aux contraintes. En effet, la courbe de satisfaction correspond bien évidemment à la fonction d'appartenance μ.

Annexe B Le formalisme UML

1 Introduction

Un système peut être décrit selon trois points de vue. Premièrement, le point de vue statique (encore appelé point de vue des données) montre les classes et leurs relations. Les notations utilisées dans cette vue sont souvent dérivées des schémas entités–associations utilisés en base de données. Cette vue constitue le noyau central de la représentation orientée objets d'un problème. Ensuite, la vue dynamique montre le comportement du système au cours du temps. Généralement, deux types de comportements sont distingués. D'une part, le comportement collectif correspond aux communications entre les objets. Par exemple, le fonctionnement d'un véhicule est le résultat de la collaboration entre les roues, le moteur, la direction … D'autre part, le comportement individuel correspond à l'évolution de l'état d'un objet, c'est-à-dire de l'évolution de son comportement. Par exemple, l'alarme d'une voiture réagit différemment selon qu'elle est neutralisée, activée, désactivée … Enfin la vue architecturale montre l'organisation physique du système (organisation des fichiers, répartition des processus …). La plupart des formalismes proposent des notations pour décrire le système selon ces trois points de vue. Dans ce qui suit, nous présentons la liste des diagrammes UML en fonction de ces trois points de vue.

2 Bref historique

Le domaine des méthodes orientées objets est un domaine très prolifique où de très nombreuses méthodes ont été proposées. Après cette phase dite de fragmentation, l'évolution des différentes méthodes proposées a mis en évidence de nombreux points de convergence. En particulier, les méthodes proposées dans [Booch 94, Rumbaugh et al. 95] ont été unifiées pour former un langage de modélisation orienté objets appelé UML [Rumbaugh et al. 99].

Plusieurs objectifs ont été recherchés par les auteurs de UML. Il se réduit à un formalisme totalement indépendant, des langages de programmation orientés objets, et des méthodes orientées objets. Il est vu

comme un moyen de communication entre les informaticiens et les experts du domaine du problème à informatiser.

3 Les diagrammes de la vue statique

3.1 Diagramme des cas d'utilisation

Ce diagramme (cf. Figure B-1) est l'outil de communication principal entre informaticiens et utilisateurs lors de la phase d'analyse. Sa vocation essentielle est de faire l'inventaire des fonctionnalités que doit satisfaire le système.

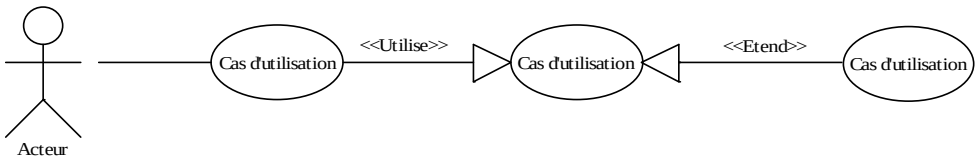


Figure B-1. *Diagramme des cas d'utilisation*

3.2 Diagramme des paquets

Le diagramme des paquets (cf. Figure B-2) permet un regroupement cohérent des différentes classes identifiées dans le diagramme précédent. Il n'existe qu'un seul type de relation entre les paquets : la dépendance. Il existe une dépendance si au moins une classe d'un paquet dépend d'une classe d'un autre paquet. Cette relation est transitive.

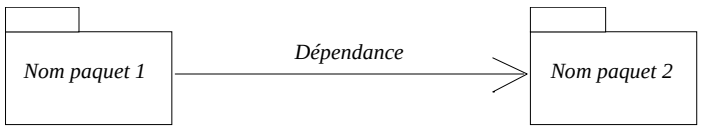


Figure B-2. *Diagramme des paquets*

3.3 Diagramme des classes

Le diagramme des classes (cf. Figure B-3) montre l'organisation statique des classes. Il est dérivé du modèle entités–associations. On y retrouve les informations telles que les attributs, les relations, leurs cardinalités …

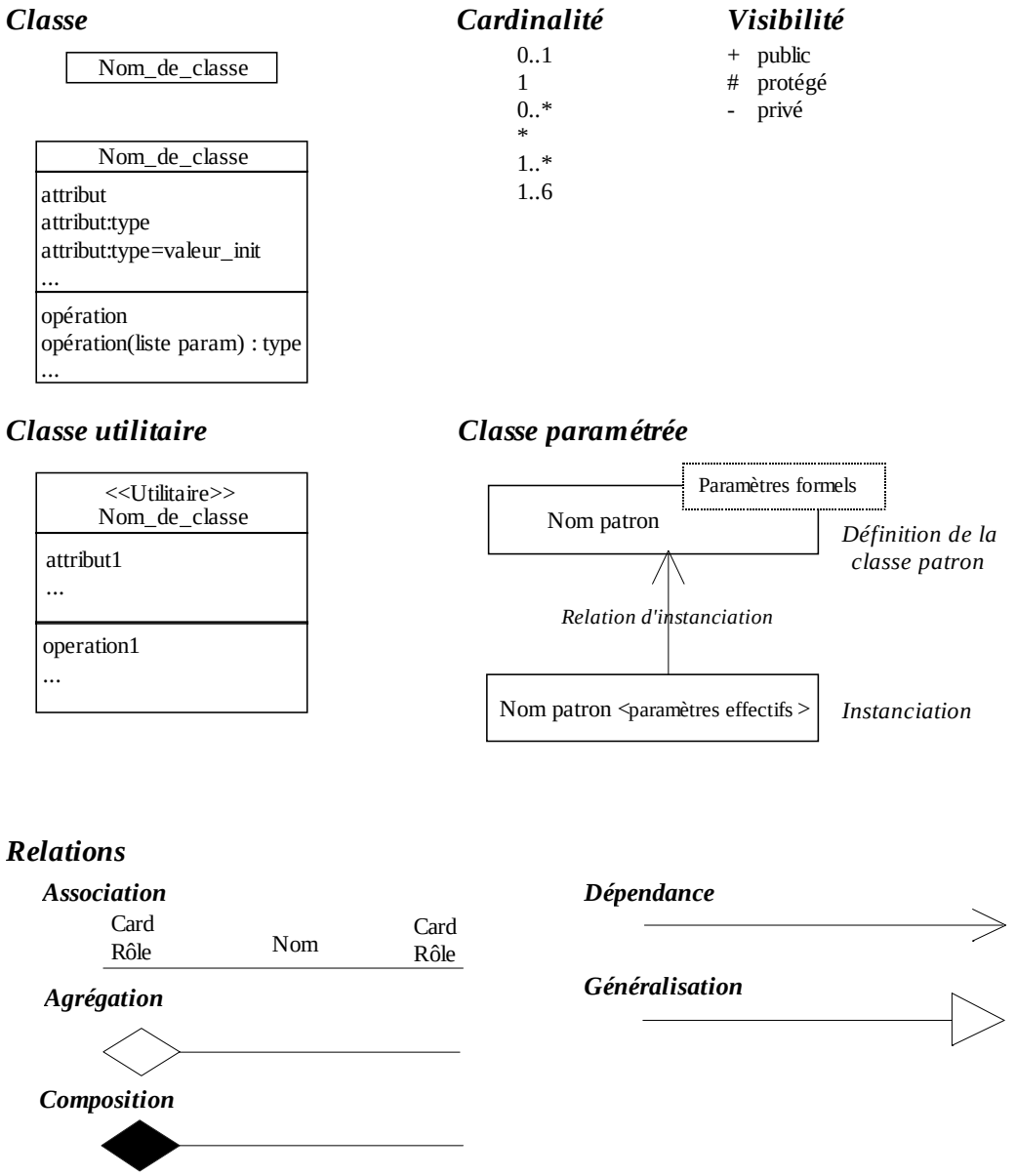


Figure B-3. *Diagramme des classes*

4 Les diagrammes de la vue dynamique

4.1 Diagrammes des interactions

Les diagrammes des interactions permettent de représenter le comportement collectif du système. Chaque phase particulière du comportement du système fait l'objet d'un diagramme d'interaction. Il existe deux types de diagramme d'interactions : le diagramme de séquence et le diagramme de collaboration. Ils sont identiques d'un point de vue sémantique, mais ils sont utilisés à des phases différentes du processus de conception du logiciel. Le diagramme de séquence (cf. Figure B-4) montre simplement, sous formes d'une chronologie de messages, le déroulement d'une phase caractéristique du système. Il est généralement

utilisé dans les phases d'analyse. Le diagramme de collaboration (cf. Figure B-5), plus complet du point de vue des relations entre objets, est plutôt utilisé dans les phases de conception.

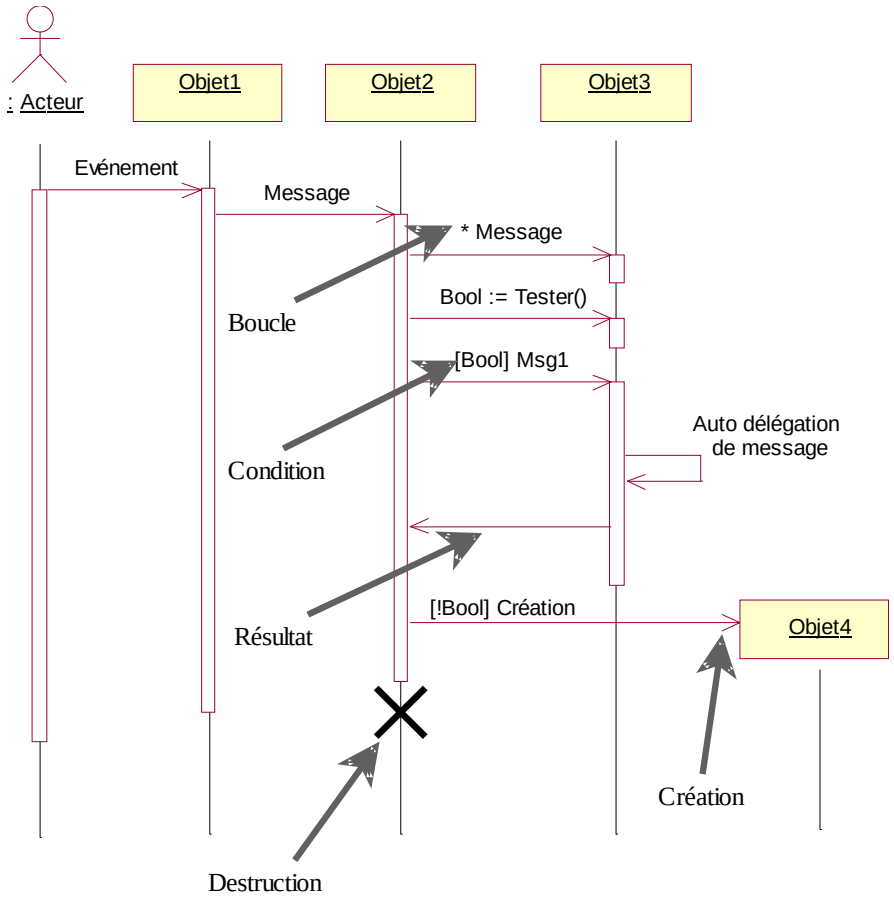


Figure B-4. *Diagramme de séquence*

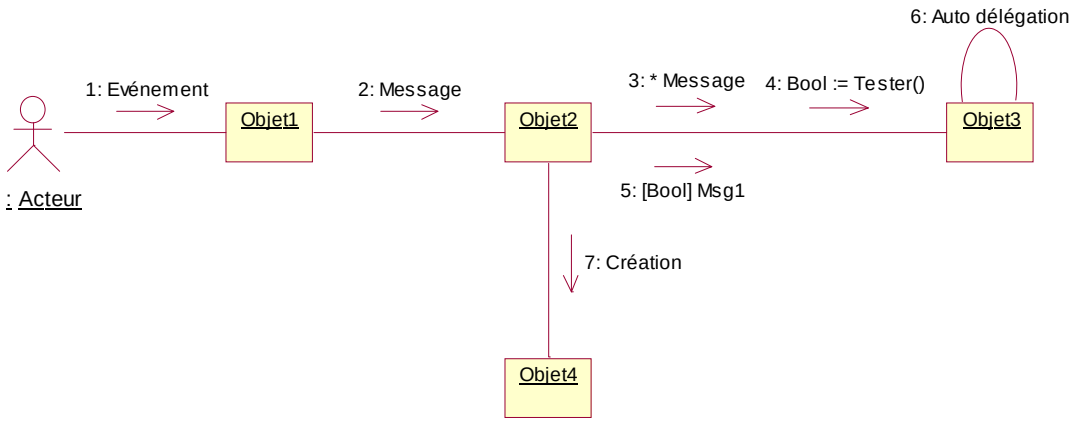


Figure B-5. *Diagramme de collaboration*

4.2 Diagramme de transitions d'états

Le diagramme de transitions d'états (cf. Figure B-6) montre le comportement individuel du comportement d'une classe. Il montre les différents états d'un objet et les transitions qui permettent de passer d'un état à un autre. À chaque état correspond un comportement particulier de l'objet.

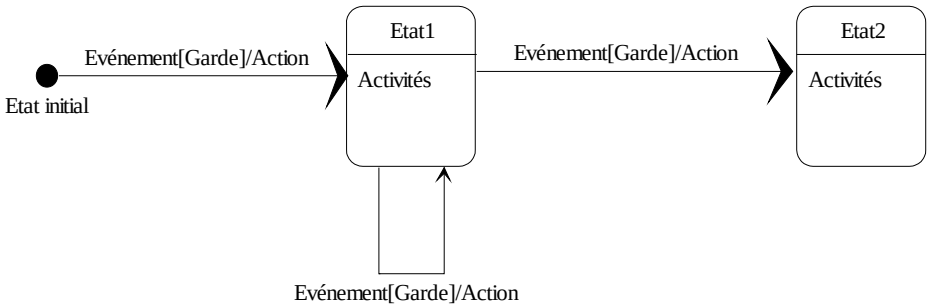


Figure B-6. *Diagramme de transitions d'états*

5 Les diagrammes de la vue architecturale

Le diagramme des composants (cf. Figure B-7) montre la répartition physique des classes dans les modules. Il n'existe qu'un seul type de relation entre modules : la dépendance. Une dépendance entre deux modules exprime le fait que la modification d'un module entraîne la recompilation de l'autre module.

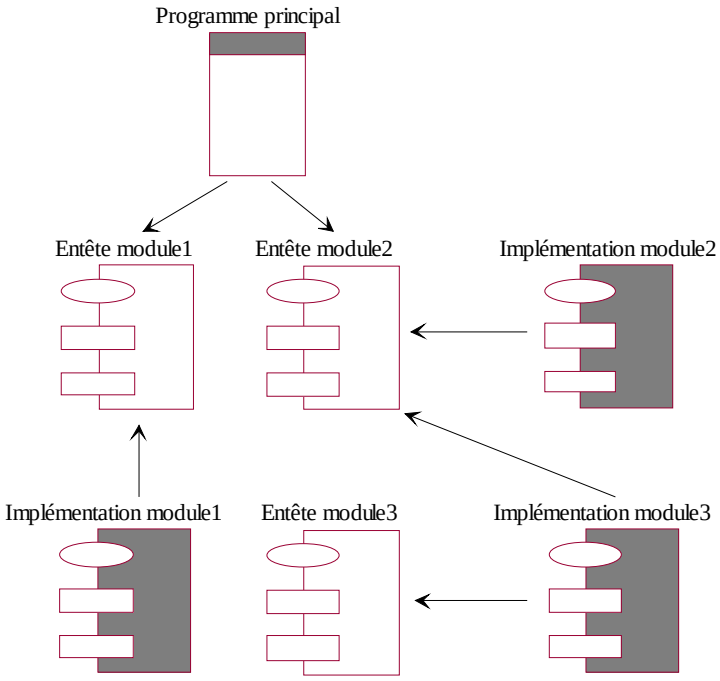
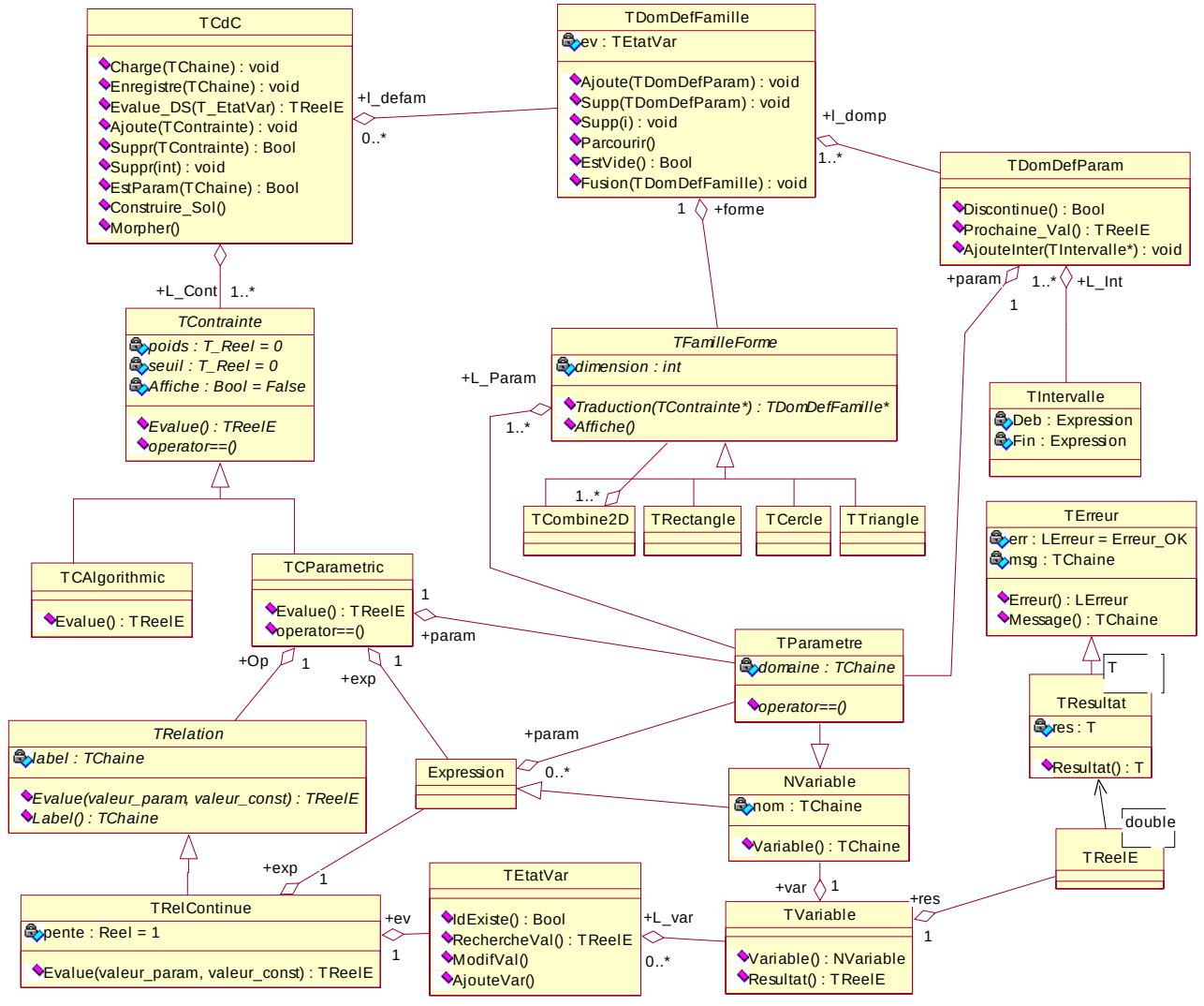


Figure B-7. *Diagramme des composants*

Annexe C Le diagramme des classes complet de notre application

En regroupant l'ensemble des classes décrites dans chaque paquet définis dans le paragraphe 2.1.4 du Chapitre 4, on obtient le diagramme des classes suivant :



Références bibliographiques

[Abbott 83] R. J. ABBOTT. « Program design by informal English descriptions ». *Comm. ACM*, 26(11), pages 882-894, 1983.

[Afav 89] ASSOCIATION FRANÇAISE POUR L'ANALYSE DE LA VALEUR. « Exprimer le besoin. Applications de la démarche fonctionnelle ». AFNOR Gestion, 1989.

[Afnor 84] AFNOR, « Analyse de la valeur. Guide pour l'élaboration d'un cahier des charges fonctionnel. Expression fonctionnelle du besoin ». Norme expérimentale X 50-151, juin 1984.

[Afnor 85a] AFNOR, « Analyse de la valeur. Vocabulaire ». Norme expérimentale X 50-150, mai 1985.

[Afnor 85b] AFNOR, « Analyse de la valeur. Recommandations pour sa mise en œuvre ». Fascicule de documentation X 50-153, mai 1985.

[Anderl et al. 95] R. ANDERL ET R. MENDGEN. « Parametric design and its impact on solid modeling applications ». *Solid Modeling, Salt Lake City, Utah USA, ACM,* 1995.

[Anderl et al. 96] R. ANDERL ET R. MENDGEN. « Modelling with constraints : theoretical foundation and application ». *Computer Aided Design, 28(3) :155-168*, 1996.

[Bechmann 95] D. BECHMANN. « *Modèles de déformation pour la modélisation géométrique et l'animation* », Habilitation à diriger des recherches, Université Louis Pasteur, Strasbourg, 1995.

[BenAmara et al. 96] A. BEN AMARA, D. DENEUX, R. SOËNEN, ET A. DOGUI. «Pré-dimensionnement en conception fonctionnelle». *Revue internationale de CFAO et d'Informatique graphique*, 11(1-2): 133-147, 1996.

[Bertrand 98] Y. BERTRAND. « Topofil : un modeleur interactif d'objets tridimensionnels à base topologique ». *Technique et science informatiques*, 17(4) :443-483, 1998.

[Bertrand et al. 98] Y. BERTRAND ET J. F. DUFOURD. « Du modèle géométrique à la programmation par les spécifications algébriques ». *Technique et science informatiques*, 17(4) :485-516, 1998.

[Bhatta et al. 96] S. BHATTA ET A. GOEL. « From design experiences to generic mechanisms : model-based learning in Analogical Design ». *AI in Engineering Design, Analysis and manufacturing,* special issue on machine learning in design, Vol. 10, pp. 131-136, 1996.

[Booch 94] G. BOOCH. *Object oriented design with applications*, Second edition, Addison Wesley, 1994.The Benjamin/Cummings publishing company, Inc., 1994

[Börner et al.] K. BÖRNER ET AL. « Structural similarity and adaptation ». *Proceedings Third European Workshop Case-Based Reasoning, Springer-Verlag, New-York*, pages 58-75.

[Brun 97] J. M. BRUN. « Modèle produit : les conditions de cohérence et leur évolution au cours du processus de conception ». *Revue internationale de CFAO et d'Informatique graphique*, 12(5) :513-529, 1997.

[Cazier 97] D. CAZIER. « *Construction de systèmes de réécriture pour les opérations booléennes en modélisation géométrique* ». Thèse de doctorat en informatique, Université Louis Pasteur de Strasbourg, 1997.

[Chung et al. 89] J. C. H. CHUNG ET M. D. SCHUSSEL. « Comparison of variational and parametric design ». *Revue de CFAO et d'Infographie*, 4(4) :81-101, 1989.

[Cohen-Or et al. 98] D. COHEN-OR, D. LEVIN ET A. SOLOMOVICI. « Three-dimensional distance field metamorphosis ». *ACM Transaction on Graphics, 17(2) :116-141*, 1998.

[Colin et al. 97] C. COLIN, E. DESMONTILS, J-Y. MARTIN ET J-P. MOUNIER. «Modèle utilisateur d'un modeleur déclaratif ». *Journées « Modeleurs géométriques », 17-19 Septembre, http ://www.emn.fr/dept_info/GEODE/La-MD/Qu-est-ce-la-MD/MD.html, Grenoble,* 1997.

[Constant 96] D. CONSTANT. « *Contribution à la spécification d'un modèle fonctionnel de produits pour la conception intégrée de systèmes mécaniques* ». Thèse de doctorat, Université de Grenoble I, France, 1996.

[Danesi et al. 99] F. DANESI, Y. GARDAN, B. MARTIN, I. PECCI. « La conception 3D par esquisses ». *Les 12ème Journées de l'AFIG, Reims,* pp. 372-381, 24-26 Novembre 1999.

[Daniel et al. 97] M. DANIEL ET V. ROSSIGNOL. « L'approche déclarative de la modélisation de courbes pour la CAO ». *Revue de CFAO et d'Informatique graphique*, 12(5) :497-512, 1997.

[Decaudin 96] P. DECAUDIN. « *Modélisation par fusion de formes 3D pour la synthèse d'images* ». Thèse de doctorat, Université de Technologie de Compiègne, France, 1996.

[Desmontils 96] E. DESMONTILS. « Formalisation des propriétés en modélisation déclarative à l'aide des ensembles flous ». *3IA*, Limoges (France), pages 87-105, avril 1996. http ://www.sciences.univ-nantes.fr/info/perso/permanents/desmontils/pages_perso/Publications.html

[Desrochers 91] A. DESROCHERS. « *Modèle conceptuel du dimensionnement et du tolérancement des mécanismes. Représentation dans les systèmes de CFAO* ». Thèse de doctorat en automatique, Laboratoire de Mécatronique, École Centrale de Paris, 1991.

[Eggli et al. 97] L.. EGGLI, C. HSU, B. D. BRÜDERLIN ET G. ELBER. « Inferring 3D models from freehand sketches and constraints ». *Computer Aided Design,* 29(2) :101-112, 1997.

[Encyclopædia] ENCYCLOPÆDIA BRITANNICA. http://search.eb.com/, « Classification Theory »

[Eustache 99] J. EUSTACHE, Y. LANUEL ET R. VIVIAN. « Utilisation de la fonctionnalité d'un objet pour optimiser le calcul d'une image de synthèse », 12ème Journée de l'AFIG, Reims–L.E.R.I., 24-26 Novembre 1999.

[Evbuomwan et al. 96] N. F. O. EVBUOMWAN, S. SIVALOGANATHAN ET A. JEBB. « A survey of design philosophies, models, methods and systems ». *Journal of enginneering manufacture*, 210(4) :301-320, 1996.

[Feng et al. 96] C. X. FENG, C. C. HUANG, A. KUSIAK ET P. G. LI. « Representation of functions and features in detail design ». *Computer Aided Design, 28(12) :961-971*, 1996.

[Fischer et al. 91] A. FISCHER ET M. SHPITALNI. « Accelerating the evaluation of volumetric modelers by manipulating CSG trees and DAGs ». *Computer Aided Design, March*, 1991.

[Foley et al. 95] J. D. FOLEY, A. VAN DAM, S. K. FEINER, J. F. HUGHES ET R. L. PHILLIPS. *Introduction à l'infographie*, Addison-Wesley France, 1995.

[Forbus 84] K. FORBUS. «Qualitative Process Theory». *Artificial Intelligence*, 24(3) :85-168, 1984.

[Forbus 93] K. FORBUS. «Qualitative Reasoning about Function : a ProgressReport». *Qualitative reasoning about function : a progress report*, *AAAI Reasoning About Function Workshop*, 1993.

[Gacôgne 97] L. GACÔGNE. *Éléments de logique floue*, Hermes, 1997.

[Gamma 95] E. GAMMA. *Design patterns : elements of reusable object-oriented software*, Addison-Wesley, 1995.

[Gardan 91] Y. GARDAN. *La CFAO : introduction, techniques et mise en œuvre, 3ème édition*, Hermes, 1991.

[Gardan et al.] Y. GARDAN, Y. LANUEL, D. PALLEZ, F. VEXO. « A methodology for a function to shape translation tool in foundry ». *Computers in Industry*. En soumission.

[Gardan et al. 95a] Y. GARDAN, E. PERRIN. « An algorithm reducing 3D boolean operations to a 2D problem : concepts and results ». *Computer Aided Design*, 28(4), pages 277-287, 1995.

[Gardan et al. 95b] Y. GARDAN, J.P. JUNG, S. LEINEN, B. MARTIN, C. MINICH, C. POINSIGNON, I. STEMART. « Conception et réalisation d'une maquette illustrant une approche du dialogue, de la gestion des contraintes et de la modélisation en CAO ». *Rapport de recherche N° 95-01, LRIM, Université de Metz*, 1995.

[Gardan et al. 97] Y. GARDAN, Y. LANUEL, C. MINICH, D. PALLEZ, E. PERRIN, C. POINSIGNON. « Typologie des outils informatique pour la conception fonctionnelle des produits ». *Rapport de recherche N° 97-05, LRIM, Université de Metz*, 1997.

[Gardan et al. 98] Y. GARDAN, C. MINICH, D. PALLEZ. « Synthèse des outils et des méthodes d'assistances à la conception fonctionnelle de produits ». *Rapport de recherche N°98-02, LRIM, Université de Metz*, 1998.

[Gardan et al. 99a] Y. GARDAN, C. MINICH, D. PALLEZ. « On shape to specifications adequacy ». *Proceedings of the 1999 IEEE International Conference on Information Visualisation*, London, 14-16 July 1999.

[Gardan et al. 99b] Y. GARDAN, C. MINICH, D. PALLEZ ET E. PERRIN. « From functions to shapes ». *Third International Conference on Engineering and Automation*, Vancouver, August 1999.

[Glassner 89] A. GLASSNER. *An introduction to Ray-Tracing*, Academic Press, London, 1989.

[Goel 97] A. K. GOEL. «Design, Analogy and Creativity». *IEEE Expert Intelligent Systems & Their Applications*, 12(2), pages 42-48, May/June 1997.

[Goel et al.] A. GOEL, S. BHATTA ET E. STROULIA. «Kritik : an early case based system». *E. Stroulia, M. L. Maher, Pearl Pu (eds), Issues and Applications of Case-Based Reasoning to design, L. Erlbaum associates.* http ://www.cc.gatech.edu/aimosaic/ faculty/goel/ABSTRACTS.html.

[Han et al. 97] S. HAN ET G. MEDIONI. « 3Dsketch : Modeling by digitizing with a Smart 3D Pen ». Proceedings of Multimedia, pp. 41-49, 1997.

[Harjani 87] D. HARJANI. « Panorama des utilisations de l'intelligence artificielle en conception assistée par ordinateur ». *Rapport interne N° 6291187, laboratoire d'Informatique, Université de Lyon I*, 1987.

[Hearst 97] M. A. HEARST. «The role of aesthetics in intelligent systems». *IEEE Expert Intelligent Systems & Their Applications*, 12(3), pages 6-12, May - June 1997.

[Hertz] A. HERTZ. http://dmawww.epfl.ch/rose.mosaic/ah/hertz.html, Alain.Hertz@epfl.ch

[Ishii et al. 93] M. ISHII, T. TOMIYAMA, ET H. YOSHIKAWA. «A synthetic reasoning method for conceptual design». WOZNY ET OLLING, éd., *Proceedings of Towards World Class Manufacturing*, IFIP 94, North-Holland, pages 3-16, 1993.

[Iwasaki 97] Y. IWASAKI. «Real-World Applications of Qualitative Reasoning». *IEEE Expert Intelligent Systems & Their Applications*, 12(2), pages 42-48, March - April 1997.

[Iwasaki et al. 93] Y. IWASAKI, R. FIKES, M. VESCOVI ET B. CHANDRASEKARAN. «How things are intended to work :

capturing functionnal knowledge in device design». *Proceedings International Joint Conference of Artificial Intelligence, AAAI Press, Menlo Park, California*, pages 1516-1522, 1993.

[Jahami 91] G. JAHAMI. « *Pour un système de synthese d'images flexible et évolutif : quelques propositions* ». Thèse de doctorat, École nationale supérieure des mines de Saint-Etienne, Informatique, France, 1991.

[Josephson 97] J. R. JOSEPHSON. «Technical note on formalizing functionnal representation». *Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, Ohio State University*, 4 pages, 1997.

[Keuneke 91] A. M. KEUNEKE. «Device Representation. The significance of functional knowledge». *IEEE Expert Intelligent Systems & Their Applications*, 6(2), pages 22-25, April 1991.

[Kief] Knowledge Intensive Engineering Framework (KIEF). Http://www.zzz.pe.u-tokyo.ac.jp/KIEF/contents-e.html.

[Kiriyama et al. 91] T. KIRIYAMA, T. TOMIYAMA, ET H. YOSHIKAWA. «The use of qualitative physics for integrated design object modeling». L. A. STAUFFER, éd., *Proceedings of Design Theory and Methodology, ASME Press*, pages 53-60, 1991.

[Kirschman et al. 96] C. F. KIRSCHMAN, G. M. FADEL ET C. C. JARA-ALMONTE. «Classifying functions for mechanical design». *Proceedings of the ASME Design Engineering Technical Conference and Computers in Engineering Conference, Irvine, California*, 18-22 Août, 1996.

[Leinen 97] S. LEINEN. « *Une nouvelle approche pour la modélisation et la gestion des contraintes en CAO* ». Thèse de doctorat, Université de Metz, France, 1997.

[Liang et al. 94] J. LIANG ET M. GREEN, « JDCAD : A highly interactive 3D modeling system ». Computer and Graphics, 18(4), pp. 499-506, 1994.

[Lienhardt 89] P. LIENHARDT. « Subdivisions of surfaces and generalised maps ». *Proceedings of Eurographics'89, Hamburg, Germany,* september, pages 439-452, 1989.

[Lienhardt 91] P. LIENHARDT. « Topological models for boundary representation : a comparison with *n* dimensional generalised maps ». Computer Aided Design, 23(1), pages 59-82, 1991.

[Lifschitz 85] V. LIFSCHITZ. « Computing circumscription ». *Proceedings IJCAI-85, Los Angeles, CA*, pages 121-127, 1985.

[Lucas et al. 95] M. LUCAS ET E. DESMONTILS. « Les modeleurs déclaratifs ». *Revue de CFAO et d'Informatique graphique*, 10(6):559-585, 1995.

[MacDowell et al. 96] J. MCDOWELL, T. LENZ, J. STICKLEN, ET M. HAWLEY. «Conceptual design for polymer composite assemblies». *AI System Support for Conceptual Design, J. Sharpe, ed., Springer-Verlag* , pages 377-389, 1996.

[McCarthy 80] J. MCCARTHY. «Circumscription – a form of non-monotonic reasoning». *Artificial Intelligence*, 13:27-39, 1980.

[Mantel] B. MANTEL. « Pour y voir plus clair…dans la logique floue ! », http://perso.club-internet .fr/bmantel/pages/logfloue.html.

[Minich 96] C. MINICH. « Un bilan des techniques d'extraction de caractéristiques de forme ». *Revue internationale de CFAO et d'Informatique graphique*, 11(6):591-615, 1996.

[Minich et al. 99] C. MINICH, D. PALLEZ. « Vers des outils informatiques d'assistance aux phases amont de la conception – État de l'art ». Revue internationale de CFAO et d'informatique graphique, 30 pages, 1999. À paraître.

[Mukherjee et al. 95] A. MUKHERJEE ET C. R. LIU. « Representation of function-form relationship for the conceptual design of stamped metal parts ». *Research in Engineering Design*, 7:253-269, 1995.

[Muller 97] P. A. MULLER. *Modélisation objet avec UML*, Eyrolles, Paris, 1997.

[Pallez 96] D. PALLEZ. « Conception d'un algorithme de radiosité basé sur les cartes généralisées ». *Mémoire de stage de DEA, LSIIT, Université de Strasbourg*, Juin 1996.

[Perrin 95] E. PERRIN. « *Une nouvelle approche pour les opérations booléennes : formalisation et mise en œuvre* ». Thèse de doctorat, Université de Metz, France, 1995.

[Pham et al. 97] D. T. PHAM ET E. TACGIN. «Techniques for Intelligent Computer Aided Design». *Artificiel intelligence in design, 2nd edition, P.T Pham ed., London, Springer-Verlag*, Chapitre 1, 1997.

[Poinsignon 97] C. POINSIGNON. « *Contribution à l'analyse et à la réalisation d'un système de CAO à base de caractéristiques de forme* ». Thèse de doctorat, Université de Metz, France, 1997.

[Prins 97] C. PRINS. *Algorithmes de graphes*, Eyrolles, 1997.

[Pun 72] L. PUN. *Introduction à la pratique de l'optimisation*, Dunod Paris, 1972.

[Qian et al. 92] L. QIAN ET J. S. GERO. «A Design Support System Using Analogy». *Proc. Second Int'l Conf. AI in Design, Kluwer Academic publishers, Dordrecht, The Netherlands*, pp. 795-813, 1992.

[Quatrani 98] T. QUATRANI. *Visual modeling with Rational Rose and UML*, Addison-Weysley, Object Technologie Series, 1998.

[Ranta et al. 96] M. RANTA, M. MÄNTYLÄ, Y. UMEDA, ET T.TOMIYAMA. «Integration of Functionnal and Feature-based product modeling – the IMS/GNOSIS experience». *Computer Aided Design*, 28(5):371-381, 1996.

[Requicha et al. 78] A.A.G. REQUICHA ET R.B. TILOVE. « Mathematical foundations of constructive solid geometry : general topology of regular closed sets ». Technical memo 27, Production Automation Project, University of Rochester, N.Y., Mars 1978.

[Rosenberg 93] M. C. ROSENBERG. «Reflections on engineering systems and Bond Graphs». *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, 115(1-2): 242-251, 1993.

[Rosenman 96] M. A. ROSENMAN. « The generation of form using an evolutionary approach ». *Artificial Intelligence*, Kluwer Academic, in J. S. Gero and F. Sudweeks (eds), Dordrecht, The Netherlands, pp. 643-662, 1996.

[Rosenman et al. 96] M. A. ROSENMAN ET J. S. GERO. « Modelling multiple views of design objects in a collaborative CAD environnement ». *Computer Aided Design*, 28(3):193-205, 1996.

[Rossignac et al. 99] J. R. ROSSIGNAC AND A. A. G. REQUICHA. «Solid Modelling ». *Encyclopedia of Electrical and Electronics Engineering*, 1999. A paraître.

[Rumbaugh et al. 95] J. RUMBAUGH, M. BLAHA, F. EDDY, W. PREMERLANI, W. LORENSEN. *OMT – Modélisation et conception orientées objet*, Masson, Paris, Prentice Hall, London, 1995.

[Rumbaugh et al. 99] J. RUMBAUGH, I. JACOBSON AND G. BOOCH. *The Unified Modeling Language Reference Manual,* Addison Wesley, Object Technology Series, 1999.

[Salomons 94] O. SALOMONS. « *Computer support in the design of mechanical products. Constraint specification and satisfaction in feature based design for manufacturing* ». Thèse de doctorat, Université de Twente, Nederland, 1994.

[Salomons 98] O. SALOMONS. « Dynamic tolerance analysis using bondgraphs ». *Proceedings DETC'98*, Atlanta, Septembre 1998.

[Shah et al. 95] J. J. SHAH ET M. MÄNTYLÄ. *Parametric and Feature based CAD / CAM*. John Wiley & Sons, Inc., New York, 1995.

[Shah et al. 96] J. J. SHAH, D. K. JEON, S. D. URBAN, P. BLIZNAKOV ET M. ROGERS. « Database infrastructure for supporting engineering design histories ». *Computer Aided Design*, 28(5):347-360, 1996.

[Shapiro et al. 95] V. ANDERL ET D. L. VOSSLER. « What is a parametric family of solids ? ». *Solid Modeling, Salt Lake City, Utah USA, ACM,* 1995.

[Shimomura et al. 95] Y. SHIMOMURA, H. TAKEDA, Y. UMEDA, ET T. TOMIYAMA. «Representation of design object based on the functional evolution process model». *Proceedings of Design Theory and Methodology, ASME*, pages 351-360, 1995.

[Shimomura et al. 96] Y. SHIMOMURA S. TANIGAWA, H. TAKEDA, Y. UMEDA, ET T. TOMIYAMA. «Functional evaluation based on function content». *Proceedings of ASME Design Engineering Technical Conference and Computers in Engineering Conference, Irvine, California*, 18-22 Août 1996.

[Shimomura et al. 98] Y. SHIMOMURA Y. YOSHIOKA, H. TAKEDA, Y. UMEDA, ET T. TOMIYAMA. « Representation of design object based on the functionnal evolution process model ». Journal of Mechanical Design, (120):221-229, June 1998.

[Takeda 94] H. TAKEDA. «Towards multi-aspect design support systems». Technical Report NAIST-IS-TR94006, Nara Institute of Science and Technology, Nara, Japan, February 1994.

[Takeda et al. 94] H. TAKEDA ET T. NISHIDA. «Integration of aspects in design processes». *J. S. Gero and F. Sudweeks editors, Artificial Intelligence in Design, Kluwer Academic Publishers*, pages 309-326, 1994.

[Takeda et al. 96] H. TAKEDA, M. YOSHIOKA, T. TOMIYAMA, ET Y. SHIMOMURA. «Analysis of design processes by Function, Behavior, Structure». *N. Cross, H. Christiaans, and K. Dorst editors, Analysing Design Activity, J. Wiley & Sons, Chichester*, pages 187-209, 1996.

[Taura et al. 98] T. TAURA, I. NAGASAKA ET A. YAMAGISHI. « Application of evolutionary programming to shape design ». *Computer Aided Design*, 30(1):29-35, 1998.

[Tdc] TDC - Les logiciels pour la Qualité dès la Conception. BP2. 25520 Goux les Usiers. Tél : 0 381 382 950.

[Tichkiewitch et al. 95] S. TICHKIEWITCH, E. CHAPA, ET P. BELLOY. «Un modèle produit multi-vues pour la conception intégrée». *Congrés international de génie industriel*, Montréal, pages 122-131, 1995.

[Tichkiewitch et al. 97] S. TICHKIEWITCH, E. C. KASUSKY. « Méthodes et outils pour l'intégration et la conception holonique ». *Revue internationale de CFAO et d'Informatique graphique*, 12(6), 1997.

[Tollenaere 98] M. TOLLENAERE. *Conception de produits mécaniques – méthodes, modèles et outils*, Hermes, 1998.

[Tomiyama et al. 93] T. TOMIYAMA, Y. UMEDA, H. YOSHIKAWA. «A CAD for functionnal Design». *Annals of CIRP'93*, pages 143-146, 1993.

[Tomiyama et al. 96] T. TOMIYAMA, Y. UMEDA, M. ISHII, M. YOSHIOKA, T. KIRIYAMA. « Knowledge systematization for a knowledge intensive engineering framework ». *Tomiyama, Mäntylä and Finger (eds) : Knowledge Intensive CAD-1, preprints of the first IFIP WG 5.2 Workshop on knowledge intensive CAD 1, pp 33-52, Chapman & Hall, 1996*.

[Tong-Tong 95] J-R. TONG-TONG. *La logique floue*, Hermes, 1995.

[Twente] Université de Twente. Hollande. http://www.rt.el.utwente.nl/clp/home/index.htm.

[Ullman 97] D. G. ULLMAN. *The mechanical design process, 2nd edition*, McGraw-Hill, 1997.

[Umeda et al. 90] Y. UMEDA, H. TAKEDA, T. TOMIYAMA ET H. YOSHIKAWA. «Function, Behaviour and Structure». Applications of Artificial Intelligence in Engineering, Computational Mechanics Publications and Springer-Verlag, pages 177-193, 1990.

[Umeda et al. 96] Y. UMEDA, M. ISHII, M. YOSHIOKA, Y. SHIMOMURA ET T. TOMIYAMA. «Supporting conceptual design based on the function-behavior-state modeler». Artificial Intelligence for Enginnering Design, Analysis and Manufacturing, vol. 10, pages 275-288, 1996.

[Umeda et al. 97] Y. UMEDA ET T. TOMIYAMA. «Functional reasoning in design». *IEEE Expert Intelligent Systems & Their Applications*, 12(2), pages 42-48, March - April 1997.

[Veerkamp et al. 93] P. VEERKAMP ET P. T. HAGEN. «Qualitative Reasoning about Design Objects». *Robotics and Computer Integrated Manufacturing*, 10(1/2), pages 33-39, 1993.

[Weber et al. 92] C. WEBER, M.SCHULTE, ET R. STARK. « Functionnal features for design in mechanical engineering ». *CARs & FOF, 8th International Conference on CAD/CAM, Robotics and Factories of the future, Tome 1,* Metz, France, pages 179-192, 1992.

[Yao et al. 97] Z. YAO ET A. L. JOHNSON. « On estimating the feasible solution space of design ». *Computer Aided Design*, 29(9):649-655, 1997.

[Yoshioka et al. 97] M. YOSHIOKA ET T. TOMIYAMA. « Pluggable metamodel mechanism : a framework of an integrated design object modeling environment ». *Proceedings of the Lancaster International Workshop on Engineering Design,* A. Bradshaw & J. Counsell Eds., pp. 17-26, 14-15 Avril 1997.

[Yoshioka et al. 98] M. YOSHIOKA, T. SEKIYA ET T. TOMIYAMA. « Design knowledge collection by modeling ». *Proceedings of the Tenth International IFIP WG 5.2/5.3 Conference, Prolamat 98, Trento–Italy*, 9-12 Septembre 1998.

[Zadeh 65] L.A. ZADEH. « Fuzzy sets », *Information and control*, 8 :338-353, 1965.

[Zeleznik et al. 96] R. C. ZELEZNIK, K. P. HERNDON, J. F. HUGHES. « Sketch : an interface for sketching 3D scenes ». Computer Graphics, 30(4) : 163-170, 1996.

Index des références bibliographiques

Index alphabétique

Une nouvelle approche fonctionnelle pour une assistance géométrique pendant les premières phases de conception de produits

Denis PALLEZ

Université de Metz
Laboratoire de Recherche en Informatique de Metz (LRIM)

Résumé : Ce mémoire est relatif à la conception assistée par ordinateur et plus particulièrement à la conception fonctionnelle de produits manufacturiers. Notre objectif est de proposer une *méthode* informatique permettant de construire le plus automatiquement possible la forme d'un produit à partir de ses spécifications (ses fonctions).

Le premier chapitre s'intéresse aux outils et aux méthodes, informatiques ou manuelles, d'assistance à la conception fonctionnelle (SADT, FAST …). Il montre que les spécifications d'un produit s'expriment principalement dans le langage naturel, ce qui rend difficile une quelconque automatisation de la conception à ce niveau. On ressent la nécessité de définir un modèle intermédiaire facilitant un passage moins brusque entre les fonctions et la forme du produit.

Le second chapitre traite de la modélisation géométrique du produit, puisque l'objectif principal est de construire sa forme. Il montre que les systèmes informatiques actuels de CAO ne fournissent pas d'assistance à la synthèse de formes. Nous identifions ensuite différentes approches de synthèse de la forme, des approches qui gèrent des informations d'un plus haut niveau sémantique que les habituelles méthodes associées aux modèles géométriques courants (B–Rep, CSG …).

Le troisième chapitre propose une méthode permettant de construire automatiquement un ensemble de formes satisfaisant les spécifications du produit à la condition que ces dernières aient été traduites en un modèle dit intermédiaire. La méthode proposant plusieurs solutions, il est nécessaire de définir la notion d'estimation permettant de comparer les solutions entre elles. La comparaison des solutions permet d'appliquer des méthodes d'optimisation pour choisir les meilleures formes.

Le quatrième chapitre étaye l'approche en proposant d'une part une maquette informatique et d'autre part une application dans un cadre industriel : la conception assistée de moules de fonderie à partir d'une définition fonctionnelle du moule.

Mots clés : CAO, conception fonctionnelle assistée, génération automatique de la forme, synthèse de la forme