MASTER EN SCIENCES INFORMATIQUES

Développement d'un S.I.A.D. dans le domaine de la vente automobile

Di Donato, Sergio

Award date:
1996

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année Académique 1995-1996

DEVELOPPEMENT D'UN S.I.A.D. DANS LE DOMAINE DE LA VENTE AUTOMOBILE

Di Donato Sergio

Promoteurs : Jean-Paul LECLERCQ
Jean VANDERDONCKT

Mémoire présenté en vue de l'obtention du titre de
Licencié et Maître en Informatique

ABSTRACT

The goal of this document is to provide the demonstration that even in a simple field as cars selling it is possible to lead the system user in order to define criterions about cars and to give, as a result, a set of cars matcing them. We call this system an Interactive Decision Support System for many reasons. First because it interacts with the user. At any stage during the use of the program, the system gives answers at the user's questions. Second because we consider that buying or selling a car is taking a decision.Third, the way imagined to answer some questions is quite original. In fact, the user is given a way of asking as many different questions he wants to ask about the cars. It is like making graphical SQL requests on the database, with the particularity that it is made without any knowledge on the semantic of the database.
Another difficulty encountred raised because it is impossible to define with some exactitude what is the best car for anyone. In fact, if we consider the price, the best car means surely the cheapest car. But if we consider the power, everyone would choose the higest power possible. That is, if we try to consider simultaneousely the price, the power and the consumption, and if we add the possibility to ask for some equipements the user desires in his car, we have to deal with a great problem because we are not sure there is a car matching entirely the combination of all the criterions. One way used to solve this problem is to guide the choice of the possible values for each criterion among all the set of possible values. Each criterion is given a rank an thus each criterion is depending of others. The first criterion, the most important, determines the values for the others criterions. The user can, of course, choose the categories, the models, the brands, the number of doors, if he wants petrol or diesel engine, etc. This are also criterions that have to be matched if choosed.

RESUME

Le but de ce document est de fournir la démonstration que même dans un domaine simple comme la vente de voitures il est possible de diriger l'utilisateur afin de définir des critères sur les véhicules et de donner, comme résultat, un ensemble de voitures qui y correspondent. Nous appelons ce système un Système Interactif d'Aide à la Décision pour plusieurs raisons. Premièrement parce qu'il interagit avec l'utilisateur. A n'impoprte quelle étape du déroulement du programme, le système donne des réponses aux questions de l'utilisateur. Deuxièmement, parce que nous considérons qu'acheter ou vendre une voiture c'est prendre une décision. Troisièmement, la manière de répondre à certaines questions est assez originale. En effet, l'utilisateur peut poser autant de questions qu'il veut à propos des voitures. C'est comme s'il faisait des requêtes SQL sur la base de données, avec la particularité qu'il le fait sans aucune connaissance de la sémantique, ni de la structure de la base de données.
Une autre difficulté est survenue, car il n'est pas possible de définir avec exactitude quelle est la meilleure voiture pour tout le monde. En effet, si nous considérons le prix, la meilleure voiture sera certainement la moins chère. Mais si nous considérons la puissance, tout le monde choisira la plus puissante.
Cela étant dit, si nous considérons simultanément le prix, la puissance et la consommation, et si nous ajoutons la possibilité de pouvoir demander un équipement particulier, nous aurons un grand problème, car nous ne sommes pas sûrs qu'il y a une voiture correspondant entièrement à la combinaison de tous les critères. Une manière de résoudre le problème est de guider le choix des valeurs possibles des critères, parmi toutes les valeurs possibles. Chaque critère reçoit une pondération. Le premier détermine les valeurs permises des autres critères. L'utilisateur peut bien entendu choisir la catégorie, le modèle, le nombre de portes,... Ce sont également des critères auxquels doivent répondre les véhicules sélectionnés.

Tout d'abord je remercie Monsieur Jean-Paul Leclercq pour la confiance qu'il m'a témoigné en acceptant d'être le promoteur de ce mémoire. Ces remerciements sont également valables pour Monsieur Jean Vanderdonckt.

Je tiens à remercier toutes les personnes qui ont permis la réalisation de ce travail, en particulier mon ami David Gallone sur qui j'ai toujours pu compter lorsqu'un problème technique s'est présenté. Son expérience et sa patience m'ont permis de terminer ce travail dans les temps.

Enfin, j'adresse mes remerciements les plus sincères à tous ceux, parents ou amis, dont le soutien m'a permis de mener le développement du logiciel à son terme.

1. INTRODUCTION

Depuis une dizaine d'années déjà, avec l'arrivée des ordinateurs personnels, la chute des prix du hardware, la standardisation toujours croissante des environnements de développements et des plates-formes de travail, nous assistons à une croissance sans cesse du nombre d'applications proposées au grand public. La complexité de celles-ci n'a de limite que les possibilités des processeurs et autres périphériques de stockage de données. De plus, les environnements mécanisés sont de plus en plus vastes. En effet, il est difficile de pénétrer dans un lieu quelconque sans remarquer un ordinateur ou autre outil informatique. Prenons par exemple les pharmacies, les supermarchés, les hôtels, sans parler de certains endroits où depuis des années l'ordinateur occupe une place dominante, à savoir, les banques et les assurances. C'est aujourd'hui une certitude, l'informatique représente dans certains domaines une réalité vitale. Il suffirait d'une panne d'un jour pour que la faillite soit prononcée.

Cette constatation étant faite, il s'agit de ne pas tomber dans un autre extrême, celui de l'informatisation obligée. L'effet serait celui de la non utilisation de l'infrastructure proposée ou d'une utilisation à contrecoeur, donc forcément mauvaise.
Dans le cas qui nous occupe, le but est de proposer un logiciel dans un domaine assez particulier qui est celui de la vente automobile. Ce domaine est particulier, car il suffit pour s'en rendre compte de voir comment un vendeur procède afin de convaincre un client potentiel, ou de voir comment s'effectue la vente proprement dite avec un client décidé à acheter. Les deux démarches sont différentes, car dans le premier cas le vendeur essaie par toutes sortes d'effets psychologiques ou autres d'intéresser une personne à franchir le pas décisif. Dans le second cas, le vendeur tente de rencontrer au mieux l'attente du client.

Nous nous proposons donc de développer deux logiciels ayant pour objectif commun de donner un maximum d'informations concernant la vente de véhicules, mais destinés à deux publics différents. En effet, le premier logiciel s'adresse exclusivement aux professionnels de la vente automobile (les concessionnaires), tandis que le second s'adresse à un public plus vaste, c'est-à-dire tout un chacun susceptible de vouloir acheter une automobile.

Ceci dit, nous allons démontrer comment un logiciel bien construit peut porter une aide efficace tant aux vendeurs qu'aux clients. Effectivement, l'idée se base sur la constatation que le nombre de véhicules proposés par chaque marque, moyennant le nombre d'options grandissant, est en hausse constante. Un vendeur, qui ne l'oublions pas est un professionnel de la vente, doit non seulement emmagasiner un grand nombre d'informations, mais encore être capable de faire des comparatifs rapides entre véhicules et en même temps donner des conseils pertinents aux clients. Il y a donc lieu de se demander s'il ne serait pas possible d'alléger la charge de travail du vendeur, afin qu'il puisse mieux se dédicacer à son objectif premier: aider le client. C'est la question qui a constamment été gardée à l'esprit durant le développement de ce logiciel destiné aux vendeurs.

Les principales fonctionnalités du logiciel destiné aux vendeurs seront les suivantes:

- choix d'un modèle;
- choix d'une catégorie (segment) de véhicule;
- choix de séries spéciales, y compris les occasions, véhicules de stock,...;
- choix en fonction de critères: prix, puissance, consommation, équipement, nombre de portes, alimentation,...;

- comparaison de véhicules;
- calcul du financement;
- proposition d'offre à un client;
- impression de catalogues personnalisés;

...

En ce qui concerne le client, avant de focaliser son choix sur un véhicule, il est confronté à plusieurs marques (plus de soixante) et pour chaque marque, il rencontre le problème que nous avons évoqué au paragraphe précédent, à savoir le nombre de véhicules proposés. Pour le client, la question est simple: comment être sûr que son choix sera le meilleur? En effet, il est peut-être possible qu'une autre marque propose un véhicule offrant les mêmes caractéristiques, mais coûtant moins cher ou étant mieux équipé (cela est du domaine du possible, car actuellement les constructeurs développent des études en R&D avec des concurrents afin d'en limiter les coûts et l'unique chose qui change quant aux véhicules proposés sont les noms, les prix et les équipements. Par exemple, le Fiat Ulysse, le Citroën Evasion et le Peugeot 806 ne sont qu'un seul et même véhicule à quelques détails près,... dont le prix!). Quand on sait qu'en cette période de crise économique que nous traversons, ce qui conditionne avant tout un client est le budget (source: vendeur Renault), on est à même de constater qu'un logiciel offrant une aide efficace est le bienvenu. Donc la question peut se résumer de la manière suivante: comment en avoir pour son argent? N'y aurait-il pas une alternative au choix effectué? C'est à ce genre de questions que se propose de répondre le logiciel destiné aux clients.

Les principales fonctionnalités du logiciel destiné aux clients seront les suivantes:
- choix d'un véhicule par nationalité;
- choix par marque;
- choix par catégorie;
- choix par modèles;
- choix par critères et classement des véhicules choisis;
- comparaison de véhicules;
- cotes des véhicules d'occasion;
- interrogation de la base de données de manière simple;

...

Les points forts des deux logiciels sont les choix par critères et l'interrogation de la base de données. En effet, d'une part le vendeur et/ou le client aura la possibilité d'obtenir un classement de véhicules en fonction des critères fournis (le logiciel est fait de telle sorte qu'il y ait toujours un véhicule à proposer, nous verrons par la suite comment cela est possible), et d'autre part, dans le logiciel destiné aux clients, l'utilisateur disposera d'un moyen à la fois très puissant et très simple d'interroger selon ses désirs la base de données. Il pourra poser autant de questions qu'il voudra et selon le mode qui lui convient le mieux. Anticipons sur ce qui va suivre. Le client construira des requêtes SQL d'une manière tout à fait graphique, inconsciente et guidée par l'interface qui lui évite des erreurs de syntaxes, de types,... Cette dernière se charge de cacher les détails.

Comme nous pouvons le constater, la création d'un logiciel dans ce domaine n'est pas dénuée de sens, bien au contraire. Un autre argument en faveur d'un tel logiciel est la constatation suivante: dans le domaine automobile tout a évolué. Depuis la conception du véhicule jusqu'à la production, la composante informatique-robotique occupe une place très importante. Même lorsqu'il s'agit de l'après-vente, l'ordinateur est très présent. Il suffit pour

cela de se rendre à un entretien et voir les logiciels de facturation, gestion du stock des pièces de rechange, gestion des clients, et autres.
L'unique domaine où pratiquement rien n'a été prévu est la vente du véhicule, mis à part l'un ou l'autre logiciel de financement. En gros, on peut affirmer qu'on continue à vendre les voitures comme on le faisait il y a trente ans. Or, si à l'époque il suffisait de parler avec une personne pour la convaincre, il faut aussi se rappeler qu'on n'avait pas non plus le choix immense qu'on a aujourd'hui.

Si de plus nous calculons les moyens économiques investis dans les coups de marketing, on se rend compte que tous les moyens sont bons pour attirer un client. A propos des journées portes ouvertes organisées désormais par toutes les marques, un vendeur (Fiat) me faisait la remarque suivante: *"Même si les journées portes ouvertes coûtent quelques millions, on est pratiquement sûr de regagner la somme investie, tout en ayant vendu plus de voitures durant cette période. Et une voiture vendue représente une publicité sur quatre roues dans la rue en plus. Donc on gagne en part de marché. C'est mathématique..."*
Or, j'ai fait remarquer à ce vendeur que son raisonnement était faux pour le motif suivant: *"Etant donné que toutes les marques procèdent à des journées portes ouvertes, chacun vendra plus, mais proportionnellement aux ventes habituelles. Ce qui veut dire qu'on retrouvera toujours la même proportion de véhicules de telle marque sur la route. Donc on ne gagne ni ne perd de parts de marché.*
On a juste vendu un peu plus pendant la période et moins après, comme tout le monde, mais ce n'est pas cela qui va changer les parts de marché. Ce qui influence réellement les parts de marché c'est de proposer un vrai service au client, de lui conseiller un véhicule dont il a vraiment besoin ou qui sera vraiment équipé comme il le désire, tout en restant dans les limites de son budget, donc de lui personnaliser le service. Ça c'est mathématique..."
Le vendeur fut entièrement convaincu de ce point de vue. Sensitive Car faisait son chemin, du moins dans l'acceptation du principe.

Les chapitres suivants vont donc être divisés de la manière suivante:

- analyse détaillée du domaine à informatiser;
- développement de la base de données du système informatique envisagé;
- développement de l'interface lié aux différentes fonctionnalités des logiciels;
- développement des modules spéciaux (choix par critères et interrogation de la base de données). On s'y attardera plus longtemps, car s'il peut être évident de choisir un véhicule selon la marque, catégorie,..., il l'est moins en ce qui concerne les critères et surtout la manière dont se fera la recherche du véhicule désiré. En ce qui concerne l'interrogation de la base de données, il s'agit d'un outil communément appelé *de deuxième génération*, c'est-à-dire obtenir ce que l'on veut sans programmation; cela vaut la peine de s'y attarder;
- preuve que les logiciels développés sont des systèmes d'aide à la décision.

Remarque importante: Au moment où l'idée de développer ce type de logiciel est née, il n'y avait sur le marché aucun autre logiciel ayant les mêmes objectifs. On était alors en septembre 1994. Actuellement, il est possible via le réseau Internet d'accéder à plusieurs adresses dans le monde entier susceptible d'apporter des réponses à des questions relatives aux voitures. Un bon exemple est "Le Moniteur de l'Automobile" qui donne les prix, les options et autres informations concernant toutes les voitures commercialisées en Belgique. Il propose également des choix par marque, modèle, critères,...
Citons enfin l'équivalent italien du "Moniteur de l'Automobile", le "Quattroruote" qui a édité un CD-ROM ayant pour mêmes objectifs ceux du *Sensitive Car*. Un dernier logiciel (avril 1996) est Cargus, développé avec la collaboration des importateurs en Belgique. Nous constatons donc un intérêt de plus en plus prononcé pour ce type de logiciel. Nous n'entrerons

pas dans un comparatif détaillé de ce logiciel avec *Sensitive Car*. Citons simplement que par rapport à ceux-ci, *Sensitive Car* ne suit pas forcément une voie standard afin d'avoir les informations retenues importantes par l'utilisateur. Ceci est possible grâce au module d'interrogation da la base de données. De plus, *Sensitive Car* s'intéresse de près à la problématique du choix de véhicules en fonction de critères, de l'importance liée à ceux-ci, ainsi que de l'équipement disponible en fonction de critères souhaités. Bien que toutes les solutions aux problèmes de choix et rangement souffrent de défauts inhérents à la méthode employée, nous avons le privilège d'être les premiers et seuls à essayer, malgré ces problèmes, d'apporter une aide. La décision, quant à elle, reste toujours du ressort de l'utilisateur. Pour pallier aux défauts des méthodes, il serait possible de proposer plusieurs méthodes de rangement, afin de donner plusieurs solutions à l'utilisateur. Nous en reparlerons largement dans le chapitre consacré à cette problématique.

2. HISTORIQUE DE SENSITIVE CAR

Les différentes étapes qui se sont succédées dans le développement du logiciel vont être succinctement abordées.

- Comme tout logiciel, il fallait lui donner un nom. L'origine du terme "Sensitive" est due au fait que le logiciel a été conçu en vue d'une utilisation sans clavier, car il peut également être utilisé, selon les versions, comme borne d'information dans un showroom, à l'aide d'un écran tactile. *Sensitive Car*, en anglais, car ce logiciel fonctionne en plusieurs langues. Cette étape n'a pas été des plus simples, car il s'agissait de trouver un nom parlant, attirant, simple et sans tomber dans des termes informatiques. Quelques propositions furent PC Car, Sys Car, Auto Car (Auto pour automatique et non automobile!), mais ce dernier semblait plutôt une blague. Automatic Car ne fut pas retenu, car il faisait allusion à Automatic System®, société fabriquant des systèmes automatisés d'ouvertures.

- Une fois baptisé, la décision de développer un logiciel exclusivement à l'usage des concessionnaires fut prise. La version relative aux clients était reportée, car plus difficile à réaliser.

- Développement d'un prototype expérimental *Sensitive Car Professional* et démonstration chez des concessionnaires. Les résultats furent assez encourageants. Des améliorations furent proposées, en vue de rendre le logiciel plus utile au niveau de la gestion des clients, personnalisation des clients et suivi de ceux-ci.

- Finalement, la version "client", baptisée *Sensitive Car* sans aucune extension, est née. Afin de la distinguer de la version *Professional*, dans ce document, nous l'appellerons *Sensitive Car User Guide*. Il s'agit d'une version pluri-marques, avec des caractéristiques importantes: une interface simplifiée à l'extrême et configuration de celle-ci à souhait, choix d'un véhicule selon des critères variables et le point le plus important étant la création d'un module supplémentaire par rapport à la version Professional: le *Sensitive Query*. Il s'agit d'un SQL graphique qui permettra à l'utilisateur d'avoir n'importe quelle information, même complexe, concernant les voitures reprises dans la base de données. En fait le logiciel travaille de deux manières: une façon *interface guided* et une façon *query guided.* La première guide le choix de l'utilisateur en proposant des critères de choix via une interface conviviale, tandis que la seconde demande à l'utilisateur d'avoir ses critères en tête.

3. DEVELOPPEMENT DE *SENSITIVE CAR*

3.1 ANALYSE CONCEPTUELLE

Ce paragraphe a pour but de donner l'analyse qui a conduit au développement du modèle entité-association concernant le logiciel. Cette analyse sera reprise sous forme de phrases simples, c'est-à-dire la plupart du temps sous la forme sujet-verbe-complément. Ceci aide fortement à discerner les différentes entités, leurs attributs et associations du modèle [BODART89]. Ces phrases ont été construites à l'aide de l'expérience personnelle dans le domaine automobile, ainsi qu'avec beaucoup de dépliants et publicités de véhicules, provenant de la plupart des importateurs, ainsi qu'avec l'aide de certains vendeurs contactés à certains moments. Le vocabulaire utilisé est bien entendu commun à toutes les marques.
Nous allons faire une analyse top-down, en ce sens que nous allons du plus général au plus particulier. En partant du fait que ce qui nous intéresse ce sont les voitures, il faut les regrouper en marques, modèles, versions, catégories,...
Bien entendu, le résultat présenté ici est un peu faussé puisque le modèle initial a parfois (rarement) été revu et amélioré afin qu'il représente le plus possible la conception réelle d'un véhicule. Le modèle présenté ci-après est le modèle final et l'analyse présentée est celle qui correspond à la version User Guide de *Sensitive Car*. Les informations relatives particulièrement à l'une ou l'autre version sont signalées.

Nous allons reprendre ci-après les entités du modèle avec les relations entre elles, ainsi que les cardinalités. Un exemple illustratif est également fourni afin d'éclaircir l'utilité de certaines entités. Les attributs seront donnés par la suite.

Une marque n'a qu'une et une seule nationalité.
"La marque Renault est française"
Une nation peut avoir plusieurs constructeurs, éventuellement aucun.
"La Belgique n'a pas de constructeur, tandis que la France en a trois"
Une marque est composée d'au moins un modèle.
"La marque Renault est composée des modèles Twingo, Clio, Laguna,..."
Un modèle ne peut appartenir qu'à une seule marque.
"Le modèle Renault Laguna ne peut appartenir qu'à Renault"
Un modèle possède au moins une version.
"Le modèle Renault Laguna possède les versions RN, RT, RXE"
Une version ne peut appartenir qu'à un seul modèle.
"La version Laguna RT ne peut appartenir qu'au modèle Renault Laguna"
Une version apparrtient à au moins une catégorie.
"La version Renault Safrane RXE appartient à la catégorie berlines familiales, mais aussi à la catégorie grandes routières"
Une catégorie comprend plusieurs versions, éventuellement aucune.
"Mercedes ne possède pas de catégorie petite citadine"
"La catégorie petite citadine de Renault comprend toutes les versions Twingo et Clio"
Une version possède au moins une cote d'occasion, une par année, échelonnées sur cinq ans (*Sensitive Car User Guide*).

"La cote de la Renault 19 RN de 1993 est de 150.000 Fb"

Une cote d'occasion n'appartient qu'à une seule version

Une version est disponible dans au moins une couleur.

"La Renault Laguna RT est disponible en blanc neige, vert émeraude, ..."

Une version est disponible dans au moins un habillage intérieur.

"La Renault Laguna RT est disponible en cuir, alcantara, ..."

Une couleur détermine au moins un habillage intérieur.

"L'intérieur cuir n'est pas disponible avec une couleur extérieure blanc neige"

Une couleur est proposable sur au moins une version, éventuellement plusieurs.

"La couleur blanc neige est proposable sur la Renault Lagune RXE et sur la Renault Safrane RT"

Un habillage intérieur détermine au moins une couleur.

"La couleur extérieure blanc neige n'est pas disponible avec un intérieur cuir"

Une version est équipée d'au moins un élément (communément appelés les options, cependant il peut également s'agir d'accessoires, c'est pour cela que le terme élément a été choisi).

"la Renault Laguna RXE possède L'ABS de série"

Un élément équipe au moins une version.

"L'ABS est disponible sur les Renault Laguna RT et RXE"

Un élément est caractérisé par au moins une option.

"L'ABS de la Renault Laguna RXE coûte plus cher que l'ABS de la Renault Clio RXE"

Une option peut être exclusive avec une ou plusieurs autres options, éventuellement aucune.

"Si l'autoradio x est monté sur le véhicule, on ne peut pas monter l'autoradio y en même temps"

Une version peut disposer éventuellement de kits.

"La Renault Laguna RT est équipable du pack électrique"

Un kit est à la disposition d'au moins une version.

"Le pack électrique est disponible sur toutes les versions RT"

Un kit est un ensemble d'au moins deux éléments.

"Le pack électrique est composé des vitres avant électriques, du toit ouvrant et du verrouillage central"

Un élément peut faire partie de plusieurs kits, éventuellement aucun.

"Les appuis-tête à l'arrière font partie à la fois du kit confort et du kit sécurité"

Un élément peut appartenir à plusieurs critères.

"Les appuis-tête à l'arrière appartiennent à la fois au critère confort et au critère sécurité"

Un critère peut comprendre plusieurs éléments, il en comprend au moins un.

"Le critère sécurité comprend l'Airbag et l'ABS"

Maintenant que nous avons identifié les entités et les associations, nous pouvons donner plus de détails concernant celles-ci. Il s'agit de donner les attributs avec leur définition, leur type de valeurs permises, ainsi que le nom que nous retrouverons dans le modèle et dans la base de données qui en découlera.

Entité NATION

Définition	Nom	Type
Nom identifiant	Nom_Pays	Texte
Continent d'appartenance	Continent	Texte
Photo drapeau national	Photo_Pays	Image

Entité MARQUE

Définition	Nom	Type
Nom identifiant	Nom_Marque	Texte
Sigle représentatif (logo)	Photo_Marque	Image
description (prestigieuse,...)	Descr_Marque	Texte

Entité MODELE

Définition	Nom	Type
Nom identifiant	Nom_Pays	Texte
description (modèle destiné aux jeunes,...)	Descr_Modèle	Texte
Photo du modèle	Photo_Modèle	Image
longueur	Longueur	Numérique décimal
largeur	Largeur	Numérique décimal
empattement	Empattement	Numérique décimal
largeur des coudes à l'avant	Larg_coude_av	Numérique décimal
largeur des coudes à l'arrière	Larg_coude_ar	Numérique décimal
capacité du réservoir	Réservoir	Numérique entier
le nombre de places	Place	Numérique entier
année de sortie officielle	Année	Date

Entité VERSION

Définition	Nom	Type
Nom Identifiant	Nom_Version	Texte
description	Descr_Version	Texte
Photo du modèle	Photo_Version	Image
Photo de grande taille	Photo_Max	Image
puissance fiscale	Puis_fiscale	Numérique entier
nombre de cylindres	Nb_Cylindre	Numérique entier
nombre de soupapes	Nb_soup	Numérique entier
type de distribution	Typ_Distr	Texte
alésage	Alésage	Numérique décimal
course	Course	Numérique décimal
puissance maximale	Puis_Maxi	Numérique entier
couple	Couple	Numérique décimal
type de carburateur	Typ_Carbu	Texte
type de catalyseur	Type_Catal	Texte
dimension du coffre	Coffre	Numérique décimal
prix	Prix	Numérique entier

hauteur	Hauteur	Numérique décimal
type de boîte de vitesse	Typ_Boîte	Texte
type de transmission	Typ_Trans	Texte
nombre de portes	Portes	Numérique entier
type d'alimentation	Alimentation	Texte
type de freins à l'avant	Frein_Av	Texte
type de freins à l'arrière	Frein_AR	Texte
Break (oui = Break, non = Autre)	Break	Texte
dimension des voies avant	Dim_Voies_av	Numérique décimal
dimension des voies arrière	Dim_Voies_ar	Numérique décimal
consommation en ville	Cons_ville	Numérique décimal
consommation à 120 Km/h	Cons_120	Numérique décimal
consommation à 90 Km/h	Cons_90	Numérique décimal
consommation moyenne	Cons_moyenne	Numérique décimal
type de pneus	Typ_Pneu	Texte
temps requis pour parcourir 1000 mètres départ arrêté	1000M_DA	Numérique décimal
temps requis pour passer de 0 à 100 Km/h	0-100	Numérique décimal
Reprise pour passer de 70 Km/h à 120 Km/h	Reprise_70_120	Numérique décimal
vitesse maximale	Vitesse	Numérique décimal
coefficient de pénétration dans l'air	Cx	Numérique décimal
type de carter	Carter	Texte
poids en ordre de marche	Poids_Ordre	Numérique décimal
poids de la charge utile	Poids_Charge	Numérique décimal
poids maximum remorquable	Charge_Remorq	Numérique décimal
poids total autorisé	Total_autor	Numérique décimal
rayon de braquage	Rayon_braq	Numérique décimal
nombre de tours du volant	Nb_Tour_Volant	Numérique décimal
type de suspension	Typ_Susp	Texte
cylindrée	Cylindrée	Numérique entier
remise maximale accordée lors de la vente (Sensitive Car Professional)	Remise_Max	Numérique décimal
année de construction	Année	Date

Rem: la remise maximale est relative à chaque version. Comme c'est une donnée pour la version Professional, cela veut dire qu'elle est connue du vendeur. Dans la version User Guide, cette donnée n'est bien entendu pas exploitée, puisqu'elle n'a pas de sens. Ce n'est pas une donnée communiquée par un importateur au même titre qu'une consommation, une cylindrée,...

Entité SERIE (spéciale)

Définition	Nom	Type
Nom identifiant	Nom_Série	Texte
Description de la série	Descr_Série	Texte
Bénéfice par rapport à la version normale	Bénéfice	Numérique entier

Rem: Il s'agit d'un cas particulier de version.

Entité OCCASION (Sensitive Car Professional)

Définition	Nom	Type
Identifiant interne	ID_Occas	Compteur
garantie	Garantie	Numérique entier
Kilométrage	Kilométrage	Numérique entier
Couleur	Couleur_Occas	Texte

Entité COTE

Définition	Nom	Type
Identifiant interne	Num_Cote	Compteur
Année de cotation	Année	Date
Cotation	Cote	Numérique entier

Entité CATEGORIE

Définition	Nom	Type
Nom identifiant	Nom_Catégorie	Texte
Description	Descr_Catégorie	Texte
Photo représentative de la catégorie	Photo_Catégorie	Image

Entité OPTION

Définition	Nom	Type
Identifiant interne	ID_Option	Compteur
prix	Prix	Numérique entier
Type d'option	Type_Option	Texte
importance	Importance	Numérique entier

- *type d'option* : permet de savoir si l'option est de série sur la version ou pas
- *importance* : utilisé dans une version future du logiciel, permet de donner une importance à une option. Dans la version actuelle de Sensitive Car, elles ont toutes la même importance

Entité KIT

Définition	Nom	Type
Nom identifiant	Nom_Kit	Texte
description	Descr_Kit	Texte
prix	Prix	Numérique entier

Rem: le prix du kit n'est pas la somme des prix des éléments qui le composent. Un kit étant un assemblage d'éléments (au moins deux), coûte moins cher que si le client devait acheter les éléments séparément.

Entité ELEMENT

Définition	Nom	Type
Nom identifiant	Nom_Elément	Texte
description	Descr_Elément	Texte
code (TO pour toit ouvrant, VC pour verrouillage central,...)	Code	Texre
obligatoire à la commande (oui = option, non = accessoire)	Obligatoire	Texte

Entité CRITERE-OPTION

Définition	Nom	Type
Identifiant interne	ID_Crit_Opt	Compteur
Nom du critère	Critère	Texte

Rem: l'entité Critère-Option sert à dire à quel groupe de critère appartient une option. Par exemple, l'airbag est une option de groupe Sécurité, alors que l'option intérieur cuir appartient au groupe Confort.

Entité COULEUR

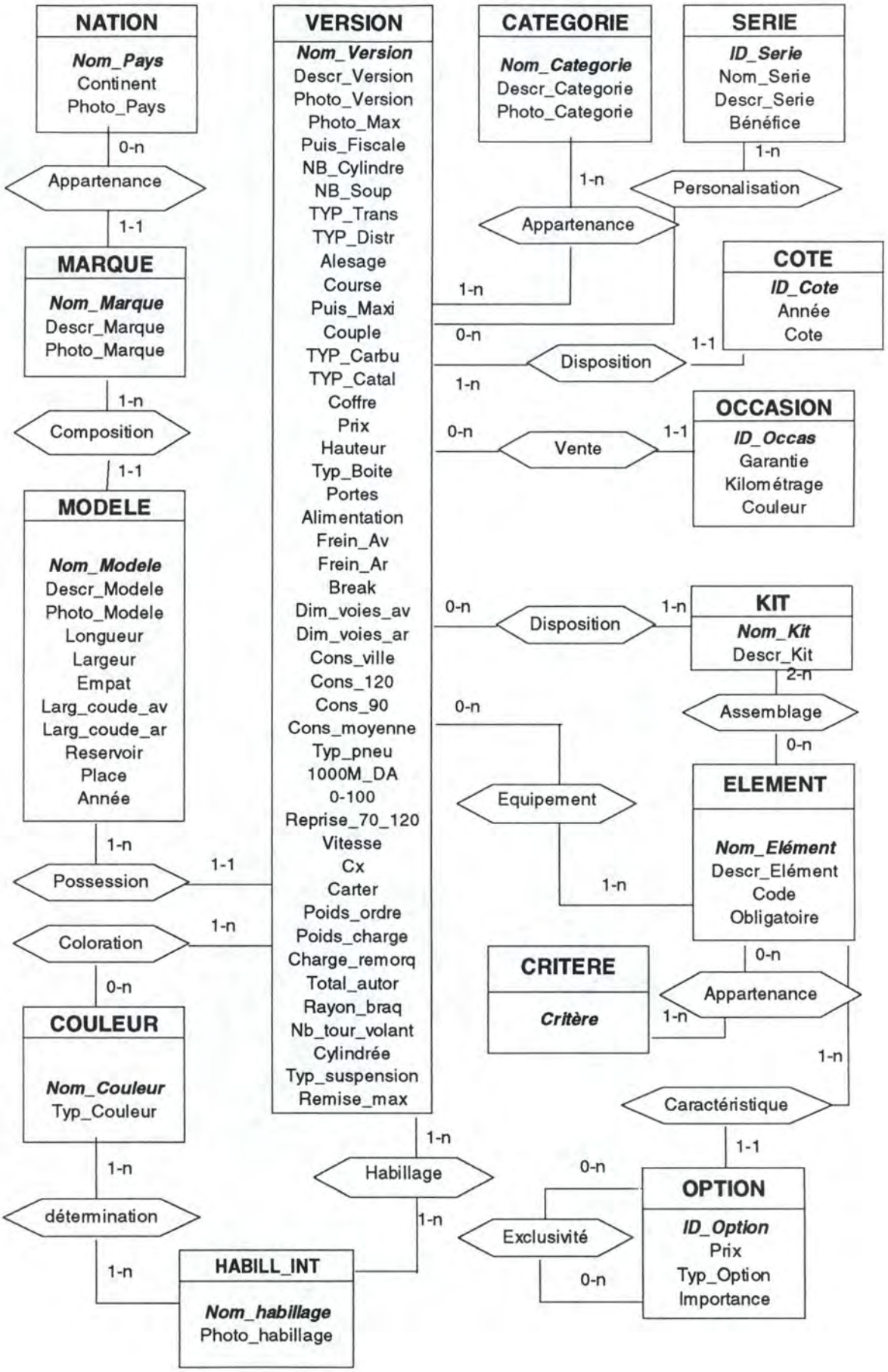
Définition	Nom	Type
Nom identifiant	Nom_Couleur	Texte
Type de couleur (métallisée,...)	Typ_Couleur	Texte
échantillon	Photo_Couleur	Image

Entité HABILLAGE

Définition	Nom	Type
Nom identifiant	Nom_Habillage	Texte
échantillon	Photo_Habillage	Photo

3.2 MODELE ENTITE -ASSOCIATION

Ci-après nous retrouvons le modèle résultant de l'analyse conceptuelle du paragraphe précédent. Il est à remarquer que certains choix de conception peuvent paraître peu clairs de prime abord. Nous verrons qu'ils sont totalement justifiés.


NATION
Nom_Pays
Continent
Photo_Pays
0-n
Appartenance
1-1
MARQUE
Nom_Marque
Descr_Marque
Photo_Marque
1-n
Composition
1-1
MODELE
Nom_Modele
Descr_Modele
Photo_Modele
Longueur
Largeur
Empat
Larg_coude_av
Larg_coude_ar
Reservoir
Place
Année
1-n
Possession
1-1
Coloration
1-n
0-n
COULEUR
Nom_Couleur
Typ_Couleur
1-n
détermination
1-n
VERSION
Nom_Version
Descr_Version
Photo_Version
Photo_Max
Puis_Fiscale
NB_Cylindre
NB_Soup
TYP_Trans
TYP_Distr
Alesage
Course
Puis_Maxi
Couple
TYP_Carbu
TYP_Catal
Coffre
Prix
Hauteur
Typ_Boite
Portes
Alimentation
Frein_Av
Frein_Ar
Break
Dim_voies_av
Dim_voies_ar
Cons_ville
Cons_120
Cons_90
Cons_moyenne
Typ_pneu
1000M_DA
0-100
Reprise_70_120
Vitesse
Cx
Carter
Poids_ordre
Poids_charge
Charge_remorq
Total_autor
Rayon_braq
Nb_tour_volant
Cylindrée
Typ_suspension
Remise_max
1-n
Habillage
1-n
HABILL_INT
Nom_habillage
Photo_habillage
CATEGORIE
Nom_Categorie
Descr_Categorie
Photo_Categorie
1-n
Appartenance
SERIE
ID_Serie
Nom_Serie
Descr_Serie
Bénéfice
1-n
Personalisation
1-n
0-n
COTE
ID_Cote
Année
Cote
1-1
Disposition
1-n
0-n
OCCASION
ID_Occas
Garantie
Kilométrage
Couleur
1-1
Vente
0-n
0-n
KIT
Nom_Kit
Descr_Kit
1-n
Disposition
2-n
Assemblage
0-n
0-n
ELEMENT
Nom_Elément
Descr_Elément
Code
Obligatoire
Equipement
1-n
0-n
CRITERE
Critère
Appartenance
1-n
1-n
Caractéristique
1-1
0-n
OPTION
ID_Option
Prix
Typ_Option
Importance
Exclusivité
0-n

3.2.1Remarques

Comme nous l'avons déjà dit, il n'est pas tout à fait clair pourquoi dans l'entité Version on retrouve un attribut break qui fait office de "flag" au sens programmation, alors que nous nous trouvons ici à un niveau de conception. Le motif est que tous les attributs de Version se retrouvent dans une version break. En fait l'entité Version est un sur-type et Break (en tant qu'entité et non attribut) un sous-type de Version. Cependant, bien qu'étant une manière élégante de présenter les choses, nous aurions retrouvé le désavantage de devoir exprimer le modèle contenant les sur-types avec un modèle équivalent n'en contenant pas lorsque dans le paragraphe suivant nous dériverons la base de données à partir du modèle. D'où un passage obligé de cette contrainte de sur-typage par une association entre une entité Break et l'entité Version. Or, le but de cette étape n'étant pas de présenter un modèle dans une quelconque des formes normales, j'ai abrégé l'étape de conception en donnant le modèle définitif où finalement l'association aurait tout de même été éliminée en introduisant un attribut break dans Version. Ceci se justifie également par le fait qu'on ne désire pas gérer les breaks séparément des autres véhicules, mais comme les 3 portes ou 4 portes. D'ailleurs une version *break* est une 5 portes un peu spéciale, chez certains constructeurs c'est une *familiale*, d'autres une *station wagon.* Ce n'est finalement qu'une question de dénomination.

Le cas des entités Série, Catégorie et Occasion est identique: elles héritent des caractéristiques de Version, mais vu qu'elles ont des caractéristiques supplémentaires, et qu'en plus elles sont gérées séparément des versions dans le programme, il a été utile de les considérer à part entière.

Un raisonnement analogue peut être tenu lorsque nous considérons l'attribut Obligatoire de l'entité Elément. Ici nous aurions dû diviser les éléments en deux classes: ceux obligatoirement commandés lors de l'achat du véhicule (les options) et les autres (les accessoires). Cela aurait donné une entité et une association de type many-to-many en plus à gérer (le prix à payer lorsqu'on aurait dérivé la base de données, en relationnel, aurait été de deux tables supplémentaires à gérer). Le choix de remplacer cette association par un attribut m'a semblé plus judicieux.

Identiquement, lorsque l'on considère l'attribut Typ_Option, qui subdivise les options en deux classes, les options de séries et celles dont on peut disposer moyennant paiement.

L'entité Habillage pourrait sembler superflue, mais il n'en est rien, car elle sert à connaître tous les habillages intérieurs concernant une version.
On connaîtra donc toutes les couleurs associées à un habillage intérieur, ainsi que tous les habillages intérieurs associés à une couleur, et ce pour chaque version.

3.2.2 Les cardinalités

Dans ce paragraphe, nous justifierons certains choix de cardinalités minimales qui ont été imposés. Lorsqu'un doute subsistait, il a été délibérément choisi de prendre le cas le plus général, quitte à paraître parfois dénué de sens. Nous verrons un exemple ci-après.

Une nation peut avoir plusieurs constructeurs, éventuellement aucun. On pourrait se demander pourquoi attacher de l'importance aux pays qui n'ont pas de constructeurs. Cela semble dénué de sens. Cependant, nous avons dit dans l'introduction que l'utilisateur pouvait demander n'importe quelle question relative aux véhicules. Une de celle-ci pourrait par exemple être : *"Quels sont les pays qui n'ont pas de constructeur?"* Avec une cardinalité minimale à 1, on ne peut pas répondre à cette question puisqu'on ne considère pas les pays qui n'ont pas de constructeurs. Le *Sensitive Car* ne répondrait pas à ses objectifs dans ce cas-ci. C'est pour cela que nous avons choisi le cas général, c'est-à-dire une cardinalité minimale à 0.

Il en est de même pour la cardinalité minimale du rôle est-équipée-de entre Version et Equipement, lorsqu'on désire connaître les éléments qui équipent une version. On pourrait par exemple se demander s'il existe des versions qui n'ont aucun élément en option. Or, en imposant une cardinalité minimale à 1, on suppose que sur toutes les versions on peut toujours ajouter quelque chose. En pratique c'est vrai, mais ici aussi, afin de rester général, la cardinalité minimale a été imposée à 0.

3.3 DERIVATION DE LA BASE DE DONNEES

Ce chapitre a pour but d'expliquer les motivations qui ont guidé certains choix plutôt que d'autres en ce qui concerne la construction des différentes tables qui constitueront la base de données de l'application.

Etant donné que nous utiliserons une base de données relationnelle, nous allons tout d'abord revoir le modèle entité-association afin qu'il soit conforme au modèle relationnel et ensuite dériver la construction des différentes tables de la base de données relationnelle. Cela par application de règles réversibles [HAINAUT86] et [HAINAUT93]. Nous obtiendrons donc un schéma naïf. Une fois cette base de données obtenue, nous procéderons à différentes optimisations afin d'obtenir la base de données telle qu'elle a été implémentée physiquement.

3.3 .1 SCHEMA E-A CONFORME AU MODELE RELATIONNEL

Afin d'obtenir ce schéma, nous appliquerons quelques règles sur le modèle entité-association que nous avons développé ci-avant. Ces règles sont réversibles, c'est-à-dire que nous pouvons toujours revenir, si nous le désirons, au modèle original. Cette étape est obligée parce que le modèle entité-association que nous avons obtenu n'est pas directement traduisible en tables. Cela est dû au fait que le modèle relationnel ne fonctionne qu'avec des associations one-to-many et non many-to-many. Par exemple, l'association Habillage entre Version et Habill_Int. De plus, il nous est interdit d'avoir des associations récursives. C'est le cas pour l'association Exclusivité qui est récursive sur l'entité Option. D'autre part, on ne peut pas non plus avoir d'attributs multivalués.
L'utilisation d'attributs multivalués a été évitée dès le départ, en sachant qu'elle était interdite dans une optique de construction d'une base de données ne supportant pas ce type d'attributs.

Propriétés d'un schéma E-A conforme au modèle relationnel

- Il n'existe pas de types d'associations.
- Tout type d'entité possède au moins un attribut.
- Tout attribut est monovalué.
- Tout attribut est élémentaire (non décomposable).
- Les contraintes d'intégrité reconnues sont les identifiants et les contraintes référentielles.

La procédure à suivre afin de rendre le schéma conforme au modèle relationnel est la suivante (je ne donne que les étapes valables pour ce modèle qui est en partie normalisé):

- Eliminer les types d'associations non fonctionnels: l'association many-to-many devient une entité et on ajoute à gauche et à droite deux associations one-to-many.
- Eliminer les attributs décomposables monovalués.

- Eliminer les attributs multivalués dépendant directement d'un type d'entités, c-à-d non composant d'un attribut décomposable: ce n'est pas le cas ici puisqu'il n'y en a pas.

- Répéter les deux opérations précédentes jusqu'à ce qu'il n'y ait plus d'attributs décomposables ou multivalués.

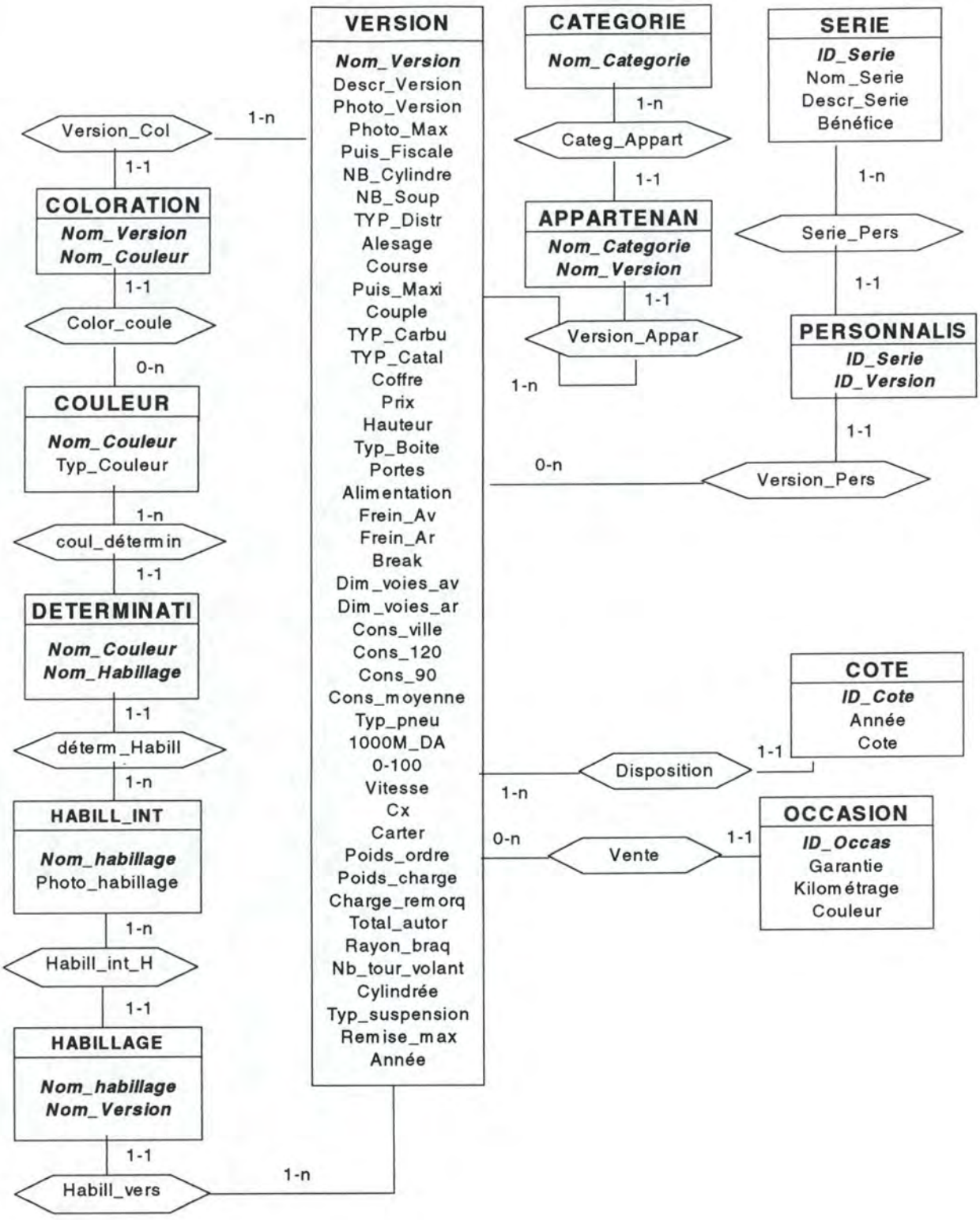
- Eliminer les types d'associations fonctionnels.

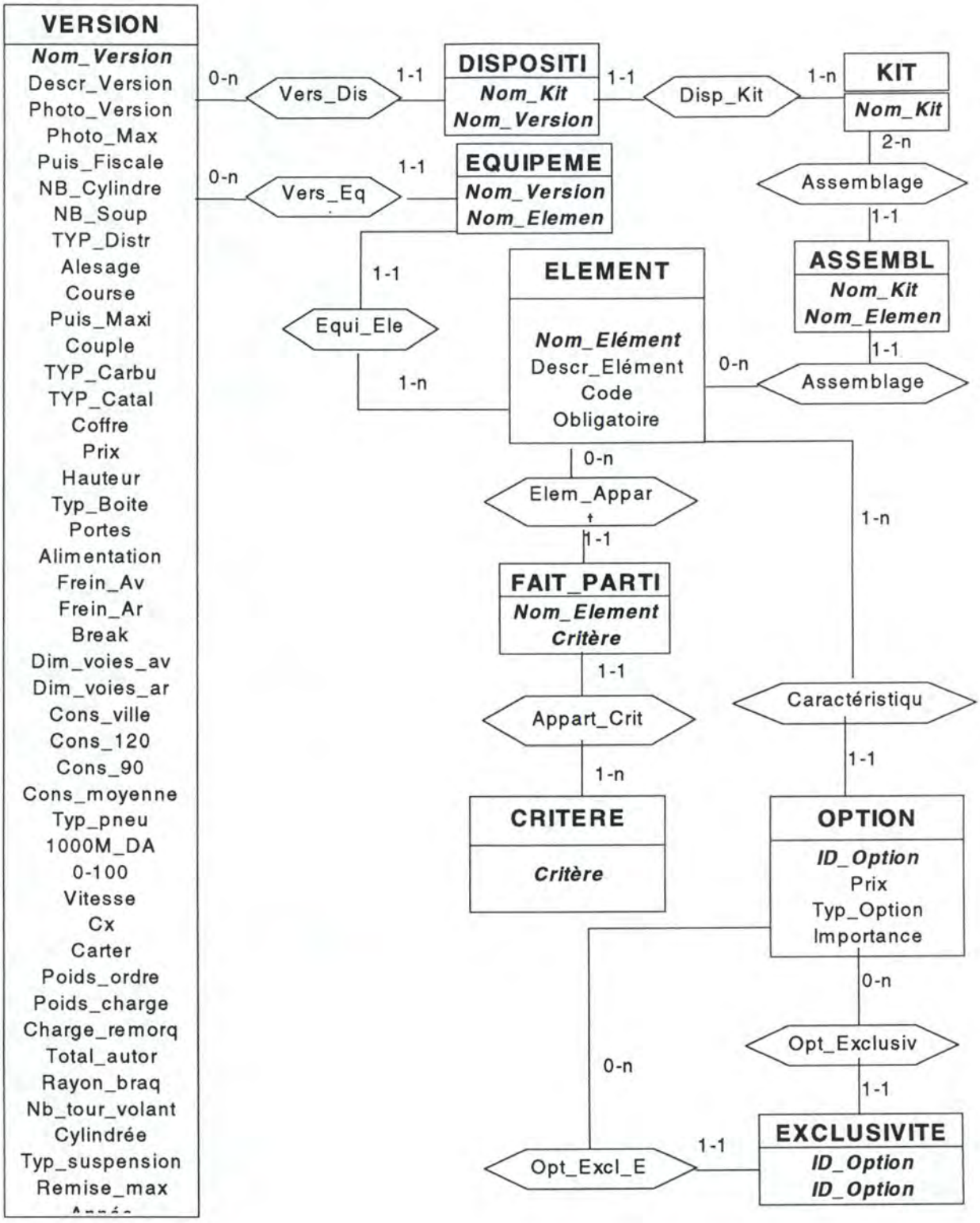
- Là où cette opération a échoué, ajouter un identifiant et reprendre la transformation.

Les différents schémas qui vont suivre concernent les éliminations des associations many-to-many. Nous le feront en plusieurs schémas, par souci de clarté.

Il est à remarquer que les entités n'ont pas toujours été reprises avec tous leurs attributs, autrement le schéma devenait assez confus. Les entités qui étaient déjà one-to-many ne figurent pas non plus sur les schémas qui suivent.

Les associations many-to-many ont donné lieu à des entités qui portent le même nom et qui sont identifiées par les identifiant des entités auxquelles se rapportait l'association many-to-many. Comme les cardinalités de cette nouvelle entité est 1-1 des deux côtés, nous avons donc bien un remplacement d'une association par une entité, et ce sans aucune ambiguïté.
Les noms donnés aux associations intermédiaires sont simplement choisis de telle manière qu'ils fassent référence aux entités auxquelles elles se rapportent. De toute manière ce sont des associations qui sont purement inutiles au moment de concevoir la base de données relationnelle. Elles font uniquement partie du processus de transformation du schéma entité-association en schéma conforme au modèle relationnel. Leur nom n'a donc aucune espèce d'importance.


VERSION
Nom_Version
Descr_Version
Photo_Version
Photo_Max
Puis_Fiscale
NB_Cylindre
NB_Soup
TYP_Distr
Alesage
Course
Puis_Maxi
Couple
TYP_Carbu
TYP_Catal
Coffre
Prix
Hauteur
Typ_Boite
Portes
Alimentation
Frein_Av
Frein_Ar
Break
Dim_voies_av
Dim_voies_ar
Cons_ville
Cons_120
Cons_90
Cons_moyenne
Typ_pneu
1000M_DA
0-100
Vitesse
Cx
Carter
Poids_ordre
Poids_charge
Charge_remorq
Total_autor
Rayon_braq
Nb_tour_volant
Cylindrée
Typ_suspension
Remise_max
Année
CATEGORIE
Nom_Categorie
SERIE
ID_Serie
Nom_Serie
Descr_Serie
Bénéfice
Version_Col
1-n
1-1
COLORATION
Nom_Version
Nom_Couleur
Color_coule
0-n
COULEUR
Nom_Couleur
Typ_Couleur
coul_détermin
DETERMINATI
Nom_Couleur
Nom_Habillage
déterm_Habill
HABILL_INT
Nom_habillage
Photo_habillage
Habill_int_H
HABILLAGE
Nom_habillage
Nom_Version
Habill_vers
Categ_Appart
APPARTENAN
Nom_Categorie
Nom_Version
Version_Appar
Serie_Pers
PERSONNALIS
ID_Serie
ID_Version
Version_Pers
COTE
ID_Cote
Année
Cote
Disposition
OCCASION
ID_Occas
Garantie
Kilométrage
Couleur
Vente


VERSION
Nom_Version
Descr_Version
Photo_Version
Photo_Max
Puis_Fiscale
NB_Cylindre
NB_Soup
TYP_Distr
Alesage
Course
Puis_Maxi
Couple
TYP_Carbu
TYP_Catal
Coffre
Prix
Hauteur
Typ_Boite
Portes
Alimentation
Frein_Av
Frein_Ar
Break
Dim_voies_av
Dim_voies_ar
Cons_ville
Cons_120
Cons_90
Cons_moyenne
Typ_pneu
1000M_DA
0-100
Vitesse
Cx
Carter
Poids_ordre
Poids_charge
Charge_remorq
Total_autor
Rayon_braq
Nb_tour_volant
Cylindrée
Typ_suspension
Remise_max
0-n
Vers_Dis
1-1
DISPOSITI
Nom_Kit
Nom_Version
1-1
Disp_Kit
1-n
KIT
Nom_Kit
2-n
Assemblage
1-1
0-n
Vers_Eq
1-1
EQUIPEME
Nom_Version
Nom_Elemen
1-1
Equi_Ele
1-n
ELEMENT
Nom_Elément
Descr_Elément
Code
Obligatoire
0-n
Assemblage
1-1
ASSEMBL
Nom_Kit
Nom_Elemen
0-n
Elem_Appar
1-1
1-n
FAIT_PARTI
Nom_Element
Critère
1-1
Appart_Crit
1-n
Caractéristiqu
1-1
CRITERE
Critère
OPTION
ID_Option
Prix
Typ_Option
Importance
0-n
Opt_Exclusiv
1-1
0-n
1-1
Opt_Excl_E
EXCLUSIVITE
ID_Option
ID_Option

3.3 .2 BASE DE DONNEES: SCHEMA NAIF

Ce paragraphe a pour but de procéder aux dernières transformations du schéma entité-association afin d'obtenir la base de données de l'application. Cependant, ce ne sera pas encore la version définitive, car nous obtenons un schéma naïf, résultant toujours de transformations réversibles. Nous verrons par la suite que certaines optimisations (transformations basées sur des considérations sémantiques des données contenues dans les différentes tables) nous permettrons d'éliminer certaines tables jugées superflues.

Les transformations auxquelles nous allons procéder sont les suivantes:

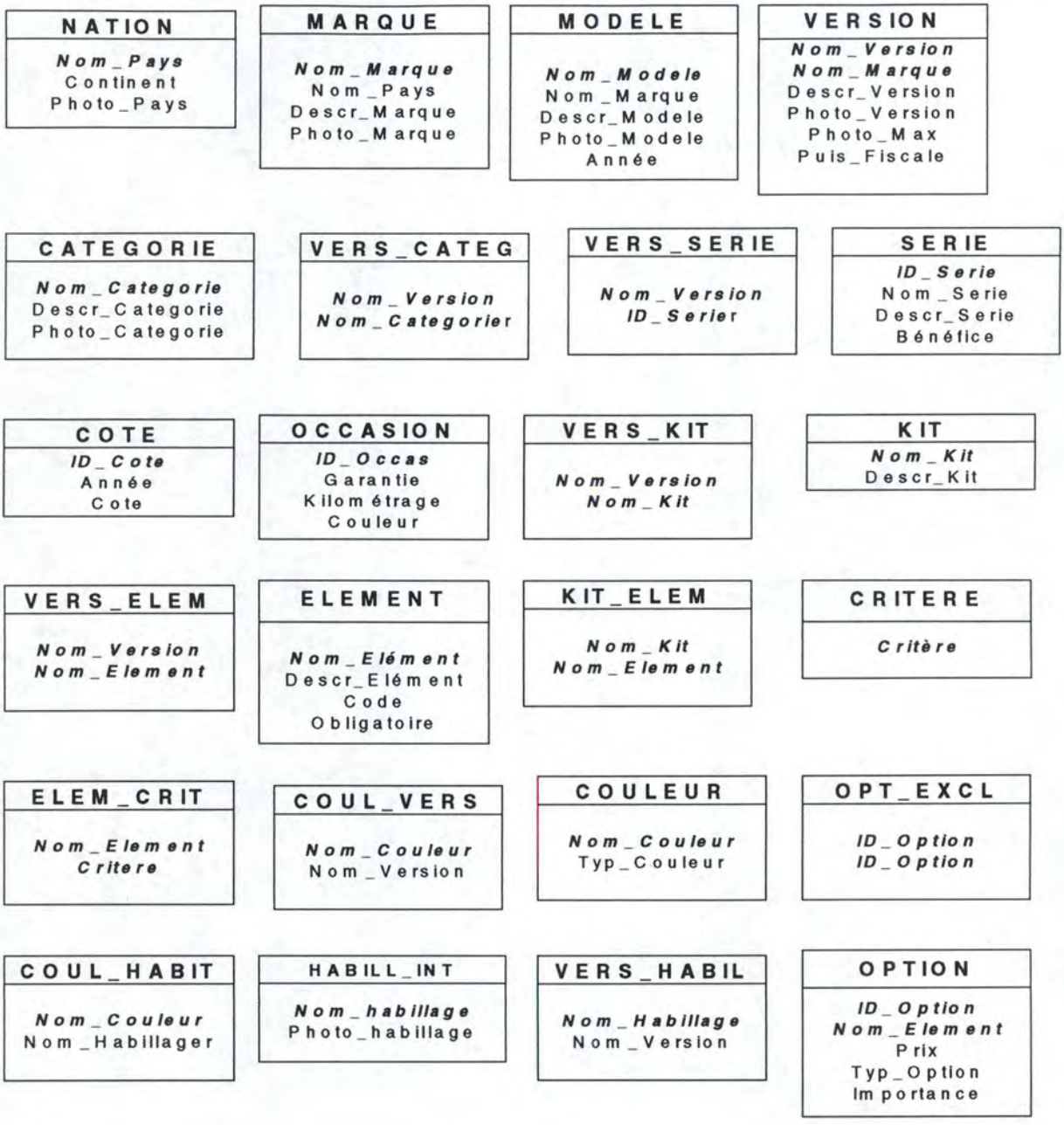
- Chaque entité devient une table dont l'identifiant est l'identifiant de l'entité.

- Toutes les associations (qui sont toutes one-to-many) disparaissent et sont remplacées par un attribut de référence dans une des entités auxquelles elles font référence (dans celle qui a la cardinalité 1-1 ou 0-1). L'attribut est l'identifiant de l'autre entité (celle qui a la cardinalité 1-n ou 0-n).

Ces transformations nous garantissent que la base de données obtenue est bien dérivée du modèle conceptuel des données, puisque toutes les transformations effectuées sont réversibles et que, de plus, le fait d'avoir transformé une association many-to-many par une entité et deux associations one-to-many, non seulement avait un sens, mais gardait bien la sémantique du modèle conceptuel original, comme nous l'avons vu au paragraphe précédent. Pour nous en convaincre, il suffirait de procéder aux transformations inverses, ce que nous ne ferons pas.

Cela étant dit, il se peut, et nous le verrons au paragraphe suivant, que certaines tables ne sont pas vraiment utiles, puisque, rappelons-nous, une table est la transformation d'une entité et une entité est la conceptualisation d'un objet appartenant au réel vécu, que nous considérons comme un tout et à propos duquel nous désirons avoir de l'information. Cependant, avec cette manière de voir les choses, il se peut que certaines informations n'ont pas vraiment besoin d'être gérées comme un tout (c'est-à-dire être identifiées et gérées comme des objets à part entière), mais plutôt comme des caractéristiques d'objets plus importants qui eux seront vus comme un tout et à propos desquels on veut avoir des informations et les gérer. Dans ce cas, ce qui était vu comme une entité et donc comme une table au sens modèle relationnel, ne représentera qu'un attribut d'une autre entité et donc sera un champ d'une table. Nous y reviendrons au paragraphe suivant.

Remarquons pour terminer que les noms des tables représentant les associations many-to-many dans le modèle conceptuel, et qui ont été transformées en entités, ont été changés de manière à représenter logiquement le lien entre deux entités. Par exemple, la table Vers_Categ ci-dessous représente l'entité Appartenance dans le schéma conforme au modèle relationnel, et représente bien l'association Appartenance du modèle conceptuel initial. Comme cela était peu parlant, il a été préférable de changer le nom de manière à faire ressortir l'origine de cette table.


NATION
Nom_Pays
Continent
Photo_Pays
MARQUE
Nom_Marque
Nom_Pays
Descr_Marque
Photo_Marque
MODELE
Nom_Modele
Nom_Marque
Descr_Modele
Photo_Modele
Année
VERSION
Nom_Version
Nom_Marque
Descr_Version
Photo_Version
Photo_Max
Puis_Fiscale
CATEGORIE
Nom_Categorie
Descr_Categorie
Photo_Categorie
VERS_CATEG
Nom_Version
Nom_Categorier
VERS_SERIE
Nom_Version
ID_Serier
SERIE
ID_Serie
Nom_Serie
Descr_Serie
Bénéfice
COTE
ID_Cote
Année
Cote
OCCASION
ID_Occas
Garantie
Kilométrage
Couleur
VERS_KIT
Nom_Version
Nom_Kit
KIT
Nom_Kit
Descr_Kit
VERS_ELEM
Nom_Version
Nom_Element
ELEMENT
Nom_Elément
Descr_Elément
Code
Obligatoire
KIT_ELEM
Nom_Kit
Nom_Element
CRITERE
Critère
ELEM_CRIT
Nom_Element
Critere
COUL_VERS
Nom_Couleur
Nom_Version
COULEUR
Nom_Couleur
Typ_Couleur
OPT_EXCL
ID_Option
ID_Option
COUL_HABIT
Nom_Couleur
Nom_Habillager
HABILL_INT
Nom_habillage
Photo_habillage
VERS_HABIL
Nom_Habillage
Nom_Version
OPTION
ID_Option
Nom_Element
Prix
Typ_Option
Importance

Contraintes d'identifiant:

Id(Nation) = Nation.Nom_Pays
Id(Marque) = Marque.Nom_Marque
Id(Modèle) = Modèle.Nom_Modèle
Id(Version) = Version.Nom_Version + Version.Nom_Modèle
Id(Catégorie) = Catégorie.Nom_Catégorie
Id(Série) = Série.ID_Série
Id(Cote) = Cote.ID_Cote
Id(Occasion) = Occasion.ID_ Occasion
Id(Kit) = Kit.Nom_Kit
Id(Elément) = Elément. Nom_Elément
Id(Critère) = Critère.Critère
Id(Couleur) = Couleur.Nom_Couleur
Id(Habill_Int) = Habill_Int.Nom_Habillage
Id(Option) = Option.ID_Option
Id(Vers_Categ) = Vers_Categ.Nom_Version + Vers_Categ.Nom_Catégorie
Id(Vers_Série) = Vers_Série.Nom_Version + Vers_Série.ID_Série
Id(Vers_Kit) = Vers_Kit.Nom_Version + Vers_Kit.Nom_Kit
Id(Vers_Elem) = Vers_Elem.Nom_Version + Vers_Elem.Nom_Elément
Id(Vers_Coul) = Vers_Coul.Nom_Version + Vers_Coul.Nom_Couleur
Id(Vers_Habil) = Vers_Habil.Nom_Version + Vers_Habil.Nom_Habillage
Id(Kit_Elem) = Kit_Elem.Nom_Kit + Kit_Elem. Nom_Elément
Id(Elem_Crit) = Elem_Crit.Nom_Elément + Elem_Crit.Critère
Id(Opt_Excl) = Opt_Excl.ID_Option + Opt_Excl.ID_Option_e
Id(Coul_Habit) = Coul_Habit.Nom_Couleur + Coul_Habit.Nom_Habillage
Id(Elem_Opt) = Elem_Opt.Nom_Elément + Elem_Opt.ID_Option

Contraintes référentielles:

Tout Nom_Modèle de Version est un Nom_Modèle de Modèle
Tout ID_Option_e de Opt_Excl_E est un ID_Option de Option
Tout Nom_Version de Vers_Categ est un Nom_Version de Version
Tout Nom_Catégorie de Vers_Categ est un Nom_Catégorie de Catégorie
Tout Nom_Version de Vers_Série est un Nom_Version de Version
Tout ID_Série de Vers_Série est un ID_Série de Série
Tout Nom_Version de Vers_Kit est un Nom_Version de Version
Tout Nom_Kit de Vers_Kit est un Nom_Kit de Kit
Tout Nom_Version de Vers_Elem est un Nom_Version de Version
Tout Nom_Elément de Vers_Elem est un Nom_Elément de Elément
Tout Nom_Version de Vers_Coul est un Nom_Version de Version
Tout Nom_Couleur de Vers_Coul est un Nom_Couleur de Couleur
Tout Nom_Version de Vers_Habil est un Nom_Version de Version
Tout Nom_Habillage de Vers_Habil est un Nom_Habillage de Habillage
Tout Nom_Kit de Kit_Elem est un Nom_Kit de Kit
Tout Nom_Elément de Kit_Elem est un Nom_Elément de Elément
Tout Nom_Elément de Elem_Crit est un Nom_Elément de Elément
Tout Critère de Elem_Crit est un Critère de Critère
Tout ID_Option de Opt_Excl est un ID_Option de Option

Tout ID_Option_e de Opt_Excl est un ID_Option_e de Opt_Excl_E
Tout Nom_Couleur de Coul_Habit est un Nom_Couleur de Couleur
Tout Nom_Habillage de Coul_Habit est un Nom_Habillage de Habillage
Tout Nom_Elément de Elem_Opt est un Nom_Elément de Elément
Tout ID_Option de Elem_Opt est un ID_Option de Option

3.3 .3 BASE DE DONNEES: OPTIMISATIONS

Comme nous l'avons déjà précisé au paragraphe précédent, certaines des tables que nous avons obtenues ne sont pas vraiment utiles. Nous allons voir lesquelles sont superflues, pourquoi elles le sont et comment obtenir une base de données équivalente, c'est-à-dire sans perte d'informations.

- ajout d'un identifiant artificiel à chaque table.

Tout d'abord, pour des raisons de facilité d'implémentation, il a été choisi d'ajouter un identifiant artificiel à chaque table. La raison qui a motivé ce choix peut paraître surprenante, mais tout à fait justifiable: lorsque par exemple nous traitons les versions, l'identifiant complet est composé normalement du nom de la marque, suivi du nom du modèle, lui-même suivi du nom de la version. Cela fait trois chaînes de caractères à traîner derrière soi chaque fois que l'on veut désigner une voiture donnée. Pour éviter cela, il est préférable d'avoir un identifiant interne, même si cela semble artificiel. Cet identifiant a le préfixe "ID_".

- disparition des tables Série et Occasion

Les tables Série et Occasion ont disparu, car elles ont été remplacées par des attributs supplémentaires dans la table version. Pour savoir si on a à faire à une série spéciale, il suffit de voir la valeur de l'attribut série_spec. Si sa valeur est "O" alors c'est une série spéciale. Dans le cas où on a "N", c'est un véhicule normal.
On aurait pu également tester la valeur de l'attribut Bénéfice. Si sa valeur est différente de 0, on a une série spéciale.

Pour Occasion, on a les attributs supplémentaires Garantie, Kilométrage et Couleur_Occas. Cela est entièrement justifiable par la considération sémantique suivante: si on compare avec la table Catégorie, qui a été gardée, on se rend compte qu'on gère des catégories en ce sens qu'on peut ajouter des catégories de voitures qui pourraient exister un jour, mais qui ne sont qu'au stade de prototype et ne sont donc pas encore reprises dans la base de données. Donc il nous faut garder une table Catégorie. Il est à noter que la cardinalité de catégorie est 0-n et non 1-n. Ce qui veut dire que ce que je viens de dire a du sens, c'est-à-dire qu'il peut y avoir des catégories qui ne correspondent à aucune version.

Cela est tout à fait différent en ce qui concerne les séries spéciales et les occasions. On n'invente pas un type de série spéciale et ensuite on vérifie quelle est la version qui s'y rapporte. Dans ce cas, on aurait dû garder la table Série. Bien au contraire, la cardinalité de Série est 1-n, c'est-à-dire qu'à coup sûr on a une version qui se rapporte à une série spéciale puisque celle-ci est généralement équipée à partir d'une version normale. Bien qu'on aurait pu considérer qu'on pourrait demander toutes les séries spéciales comprises dans la base de données, il a été choisi d'éliminer la table Série. Un raisonnement analogue a été fait en ce qui concerne la table Occasion. Ici il faut également s'attacher au sens de cette table. Le fait de la garder ne nous apporte pas grand-chose de plus, car cela voudrait dire qu'on gèrerait toutes les occasions avec tous les kilométrages et toutes les garanties qu'on pourrait imaginer. On aurait une explosion du nombre d'enregistrements dans la table Occasion. Je me demande qu'est ce qu'on pourrait bien en faire et quelles requêtes (au sens SQL) on pourrait faire à propos de cette table, dans quel objectif? La décision de l'éliminer a donc été prise.

Cependant, revenant un instant sur le cas de Série, il est évident qu'on aurait tout aussi bien pu la garder. La décision contraire a été prise, je m'y tiens donc.

La table Vers_Série a donc également disparu puisqu'elle représentait l'association entre Version et Catégorie.

- disparition de la table Vers_Coul.

La table Vers_Coul a disparu et a été remplacée par l'attribut ID_Version dans la table couleur. En fait on peut s'étonner de cette manière de faire, car cette table ne traduisait pas une association one-to-many dans le shéma conceptuel de départ. Alors on est en droit de se demander pourquoi avoir agit de la sorte. Tout simplement pour essayer de concentrer les données dans un minimum de tables, plutôt que de les disperser dans une multitude de tables dans un souci de clarté conceptuelle. Tant qu'on respecte les conventions, qu'on évite des pertes d'informations et qu'on garde la cohérence des données, on peut agir de la sorte.

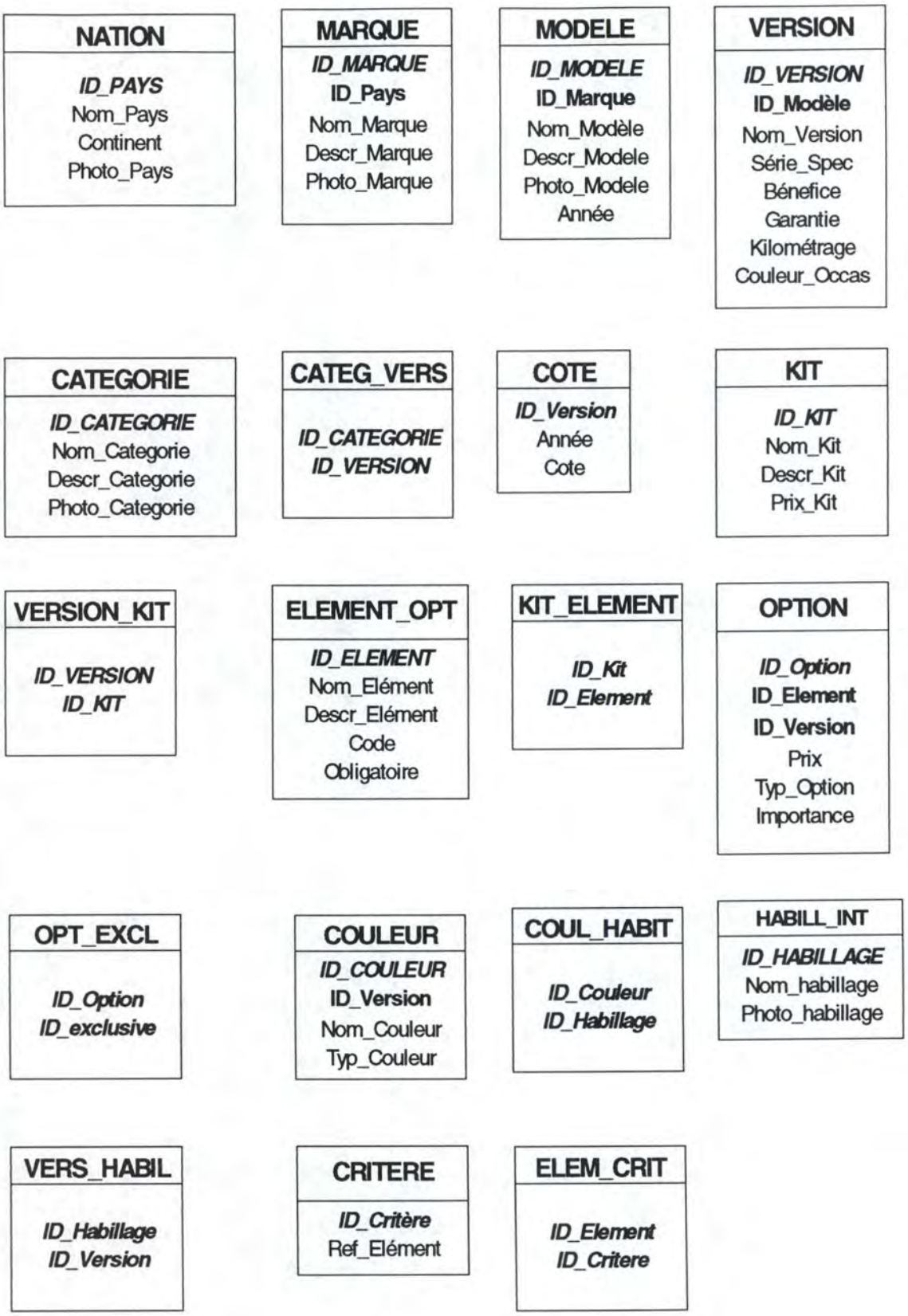
- disparition de la table Vers_Elem

Un raisonnement analogue a été tenu lorsqu'il a été décidé de supprimer la table Vers_Elem qui représente, rappelon-nous, l'association Equipement (une version est équipée de plusieurs éléments et un élément peut équiper plusieurs versions). Dans ce cas, plutôt que de passer par l'association Vers_Elem, on accède à la table Option, qui contient également les attributs de référence ID_Version et ID_Elément. Ceci s'avèrera judicieux, nous le verrons plus loin, au niveau des traitements, lorsque nous devrons considérer les critères de choix de véhicules basés sur l'équipement du véhicule et pour lequel le prix est bien entendu de première importance. Si nous n'avions pas les prix des options et les identifiants des versions dans la même table, il aurait fallu naviguer entre plusieurs tables, d'où une perte de temps considérée comme non négligeable.

C'est ce qui a également motivé en partie le choix global d'élimination de tables. Au delà de considérations purement conceptuelles, il faut également faire des choix au niveau de contraintes non fonctionnelles comme le temps d'exécution du programme, la rapidité d'accès aux informations, quitte à semer un peu de trouble dans l'organisation physique des données.

Quoiqu'il en soit, le tout est d'avoir procédé à une analyse conceptuelle convenable et d'avoir remarqué certains liens entre les données, même si par après celles-ci sont organisées différement pour des questions d'ordre fonctionnel. Cette stratégie permet d'une part de travailler avec des données cohérentes, puisqu'au départ on a bien remarqué l'existence d'autres entités et d'autres associations, et d'autre part de garder en tête que le logiciel va être écrit avec tel langage, avec accès à telle base de données, somme toute des considérations purement physiques dont il faut malgré tout tenir compte en parallèle lors de l'étape de conception, même si dans ces pages tout paraît avoir été fait de manière tout à fait séquentielle.

Ceci reste un problème, non seulement pour le développement de ce logiciel, mais dans la conception de n'importe quel logiciel, puisqu'il faut dès le départ faire des choix décisifs relatifs à l'implémentation alors qu'on essaie encore de comprendre sur quelles données on travaille et le sens de celles-ci. Néanmoins, les choix qui ont été faits se sont avérés être les bons puisque, anticipons légèrement sur la suite, les temps d'exécution sont remarquables et le logiciel est fonctionnellement parlant une réussite. Jusqu'ici, l'aventure ne fait que commencer, voyons les étapes suivantes...


NATION
ID_PAYS
Nom_Pays
Continent
Photo_Pays
MARQUE
ID_MARQUE
ID_Pays
Nom_Marque
Descr_Marque
Photo_Marque
MODELE
ID_MODELE
ID_Marque
Nom_Modèle
Descr_Modele
Photo_Modele
Année
VERSION
ID_VERSION
ID_Modèle
Nom_Version
Série_Spec
Bénefice
Garantie
Kilométrage
Couleur_Occas
CATEGORIE
ID_CATEGORIE
Nom_Categorie
Descr_Categorie
Photo_Categorie
CATEG_VERS
ID_CATEGORIE
ID_VERSION
COTE
ID_Version
Année
Cote
KIT
ID_KIT
Nom_Kit
Descr_Kit
Prix_Kit
VERSION_KIT
ID_VERSION
ID_KIT
ELEMENT_OPT
ID_ELEMENT
Nom_Elément
Descr_Elément
Code
Obligatoire
KIT_ELEMENT
ID_Kit
ID_Element
OPTION
ID_Option
ID_Element
ID_Version
Prix
Typ_Option
Importance
OPT_EXCL
ID_Option
ID_exclusive
COULEUR
ID_COULEUR
ID_Version
Nom_Couleur
Typ_Couleur
COUL_HABIT
ID_Couleur
ID_Habillage
HABILL_INT
ID_HABILLAGE
Nom_habillage
Photo_habillage
VERS_HABIL
ID_Habillage
ID_Version
CRITERE
ID_Critère
Ref_Elément
ELEM_CRIT
ID_Element
ID_Critere

Contraintes d'identifiant:

Id(Nation) = Nation.ID_Pays
Id(Marque) = Marque. ID_Marque
Id(Modèe) = Modèle. ID_Modèle
Id(Version) = Version. ID_Version
Id(Catégorie) = Catégorie. ID_Catégorie
Id(Cote) = Cote.ID_Cote
Id(Kit) = Kit.Nom_Kit
Id(ElémentOpt) = Elément. ID_Elément
Id(Critère_Option) = Critère. ID_Critère
Id(Couleur) = Couleur. ID_Couleur
Id(Habill_Int) = Habill_Int. ID_Habillage
Id(Option) = Option.ID_Option
Id(Vers_Categ) = Vers_Categ. ID_Version + Vers_Categ. ID_Catégorie
Id(Vers_Kit) = Vers_Kit. ID_Version + Vers_Kit. ID_Kit
Id(Vers_Habil) = Vers_Habil. ID_Version + Vers_Habil. ID_Habillage
Id(Kit_Elem) = Kit_Elem. ID_Kit + Kit_Elem. ID_Elément
Id(Elem_Crit) = Elem_Crit. ID_Elément + Elem_Crit. ID_Critère
Id(Opt_Excl) = Opt_Excl.ID_Option + Opt_Excl.ID_Option
Id(Coul_Habit) = Coul_Habit. ID_Couleur + Coul_Habit. ID_Habillage

Contraintes référentielles:

Tout ID_Modèle de Version est un ID_Modèle de Modèle
Tout ID_Option de Opt_Excl est un ID_Option de Option
Tout ID_Version de Vers_Categ est un ID_Version de Version
Tout ID_Catégorie de Vers_Categ est un ID_Catégorie de Catégorie
Tout ID_Version de Vers_Kit est un ID_Version de Version
Tout ID_Kit de Vers_Kit est un ID_Kit de Kit
Tout ID_Version de Vers_Elem est un ID_Version de Version
Tout ID_Elément de Vers_Elem est un ID_Elément de Elément
Tout ID_Version de Vers_Coul est un ID_Version de Version
Tout ID_Couleur de Vers_Coul est un ID_Couleur de Couleur
Tout ID_Version de Vers_Habil est un ID_Version de Version
Tout ID_Habillage de Vers_Habil est un ID_Habillage de Habillage
Tout ID_Kit de Kit_Elem est un ID_Kit de Kit
Tout ID_Elément de Kit_Elem est un ID_Elément de Elément
Tout ID_Elément de Elem_Crit est un ID_Elément de Elément
Tout ID_Critère de Elem_Crit est un ID_Critère de Critère
Tout ID_Option de Opt_Excl est un ID_Option de Option
Tout ID_Couleur de Coul_Habit est un ID_Couleur de Couleur
Tout ID_Habillage de Coul_Habit est un ID_Habillage de Habillage

3.4 ANALYSE DES TRAITEMENTS

Maintenant que nous avons une idée plus claire des données sur lesquelles nous allons travailler, il est utile de s'intéresser à ce que nous allons devoir faire, en d'autres mots quels sont les traitements qu'on va devoir opérer sur les données. Quelles sont les fonctionnalités attendues du logiciel, à quoi va-t-il servir, à quelles questions ou groupe de questions va-t-il devoir répondre, quel type de réponse faudra-t-il prévoir? Voilà le but de ce chapitre.

Il est important, dès le départ, de préciser que ce qui est présenté comme une succession de traitements n'a pas été vue comme cela durant le développement et ce pour une raison assez normale: étant donné que ce logiciel a été inventé de toute pièce (il n'a été demandé ni par un concessionnaire, ni par un distributeur), il y a eu des retours en arrière et des ajustements afin de mieux cerner le problème relatif à un traitement particulier. De plus, il a fallu également penser à l'interface dans le même moment où le traitement en question était imaginé. Il n'était pas question de reconstruire une nouvelle interface chaque fois que surgissait un nouveau traitement, et ce pour une question évidente de temps. Comme nous le verrons ci-après, l'approche de la version User Guide est complètement différente par rapport à la version Professional. Dans le cas de la première version, non seulement elle est destinée à un public particulier (rappelons que la version Professional est une version destinée aux concessionnaires et est une version mono-marque, tandis que la User Guide est destinée à n'importe qui, elle est pluri-marques), mais de plus, au moment de son développement, elle ne connaissait pas encore de logiciel identique. En effet, on pourra le remarquer, la version User Guide, non seulement est plus performante, mais est également d'un niveau professionnel supérieur. Pour mettre en évidence cela, nous allons faire, du moins essayer lorsque c'est possible, un comparatif entre les deux logiciels.

Le but du logiciel, ne l'oublions pas, est d'être un support à la décision dans le choix d'un véhicule. Il est donc tout à fait normal de s'interroger sur la manière de choisir un véhicule. Soyons un peu psychologues et lançons-nous dans les méandres du cerveau humain afin de comprendre les mécanismes internes qui s'opèrent lorsque le problème ou la joie (cela dépend des points de vue et nous verrons que cela est très important) de choisir un nouveau véhicule se présente à nous.

3.4 .1 Sensitive Car Professional

Tout d'abord, intéressons-nous à la première version. Cela suppose donc que l'on a déjà choisi la marque du véhicule (peu importe en ce moment ce qui a poussé le client à choisir telle marque plutôt qu'une autre, cela intéressera la deuxième version).

- Nous devons donc présenter les modèles disponibles au sein de la marque au client, car non seulement cela représente une manière de choisir un véhicule, mais de plus il n'est pas dit que le client ait connaissance de tous les modèles disponibles (ce serait par exemple le cas d'un client travaillant pour une société et qui serait obligé d'acheter une voiture de société dans telle marque, parce que des accords subsisteraient par exemple entre sa société et le distributeur).

- On peut également s'intéresser aux catégories de véhicules proposées par la marque. Ici aussi, il est utile d'avoir cette information, car chaque marque essaie d'être présente dans à peu près tous les segments de voitures. Une voiture pouvant appartenir à plusieurs segments (cfr. la table Catégorie), on peut choisir cette voie afin de cerner le véhicule que l'on veut acheter.

- Supposons que le client n'ait ni idée du modèle ni de la catégorie, mais que par exemple ce qui motive son choix serait plutôt un intérêt économique portant sur toutes les voitures de la marque, il voudrait peut-être une liste de toutes les voitures avec un choix qui s'affinerait en fonction du prix. En d'autres termes, que peut-on avoir pour telle somme d'argent? C'est une question qui est loin d'être dénuée de sens, car ce qui motive les gens en particulier durant cette période de crise économique généralisée est bien une question d'argent.

- Une autre manière de choisir un véhicule est de faire des comparatifs avec d'autres véhicules de la même marque. En fait, cela suppose que le client ait déjà une idée assez fine des véhicules qu'il désire comparer, soit parce qu'il s'est déjà renseigné auparavant, soit parcequ'il hésite entre deux ou plusieurs versions d'un même modèle, par exemple.
  Les renseignements qu'il peut vouloir sont d'ordre général: puissance, consommation, équipement de série, équipement optionnel, équipement de confort, équipement de sécurité,... C'est parfois un détail qui fait décider un client à l'achat d'une voiture.

- On peut également s'intéresser aux bonnes affaires proposées par une marque. Il s'agit principalement des séries spéciales, des séries limitées, mais aussi des voitures de direction (ce sont des voitures qui ont en général servi à des essais, ou tout simplement des voitures qui étaient exposées dans le salon. Quoiqu'il en soit, elles ont très peu roulé et bénéficient d'une bonne garantie.) ou des véhicules en stock (pas encore disponible dans la version Professional).

- La dernière approche lorsque nous voulons acheter un véhicule est lorsqu'on n'a aucune idée en ce qui concerne la forme, mais qu'on a un certain nombre de considérations fonctionnelles relatives au véhicule souhaité. On parle alors de critères.
  On peut se demander quelle différence il y a avec le troisième point ci-dessus.
  En fait, comme on essaie de choisir un véhicule par raffinements successifs, il est inutile de proposer toute la panoplie de critères selon lesquels on peut choisir un véhicule. En général, les gens se contentent d'une liste où ils peuvent choisir le prix, le nombre de portes et le type de boîte de vitesse. S'ils veulent plus, on leur propose d'autres critères, autrement,

autant rester simples, car ceci aussi est un des objectifs du logiciel, ne l'oublions pas, la simplicité dans l'utilisation et dans la présentation des résultats et des possibilités de choix.

Ceci fera l'objet du chapitre consacré à l'étude de l'interface, nous y reviendrons donc. Les critères que l'on considère ici sont le prix, la puissance, la consommation, le nombre de portes, le type de boîte da vitesse, la catégorie de véhicule (il est inutile de proposer une petite citadine à un client qui s'intéresse à une voiture de segment supérieur), l'espace de l'habitacle, l'espace du coffre, les options qui peuvent être de type confort, sécurité, audio, téléphonie et autres. Une fois que le client a décidé de donner une valeur à ces critères (quand je dis donner une valeur, il peut, à ce stade, s'agir de n'importe quelle manière et sous n'importe quelle forme), et une importance à ces critères, le logiciel passe en revue la base de données afin de trouver le véhicule qui *"répond le mieux"* aux besoins du client. En aucun cas le client ne peut partir avec une réponse négative qui dirait par exemple: *"Désolé, il n'y a pas ce que vous cherchez"*.

Il faut donner le moyen de choisir parmi ce qui est disponible, autrement les gens veulent l'impossible, c'est bien connu. D'ailleurs, arrivé à un certain point il faut bien se contenter, car c'est inutile de demander une petite voiture qui puisse transporter trois tonnes, avec de la place pour huit passagers, une vitesse de pointe de 250 Km/H, une consommation de 4L/100Km, un moteur diesel et une boîte de vitesse automatique, pour un prix de 400.000Fb. J'oubliais l'ABS et le climatiseur. Je crois qu'on peut chercher partout, ça m'étonnerait que ça existe. Si c'est le cas, *Sensitive Car* est inutile, tout le monde achèterait cette voiture. Donc l'approche par les critères doit être pensée de manière à éviter les excès et à ne demander que ce qui est permis.

Ceci nous a permis de dépoussiérer grossièrement les différentes fonctionnalités attachées à *Sensitive Car* Professional. Nous allons maintenant devoir formaliser un peu plus ces fonctions. Il est utile de dire que les fonctions citées ici ne se retrouveront pas forcément dans le programme. Les noms donnés dans ce chapitre sont des noms formels et parlant, alors que dans le programme une fonction a parfois été utilisée plusieurs fois avec des arguments différents, mais avait le même type de comportement. Par exemple, si on veut la liste de tous les modèles ou de toutes les catégories on aura la même fonction qui fait une recherche dans une table de la base de données, mais avec un argument différent. Les fonctions sont définies en termes d'input et d'output. Les inputs représentent les arguments de la fonction, tandis que l'output représente le résultat de l'application de la fonction sur ces arguments. Nous ne donnons pas les détails sur la manière de procéder de ces fonctions afin de ne pas encore entrer dans un monde algorithmique. Nous y reviendrons plus loin, mais uniquement pour certaines fonctions plus difficiles à implémenter. Certaines sont vraiment trop évidentes pour qu'on utilise un langage de description d'algorithme afin de les décrire.

3.4 .1.1 Liste des fonctionnalités

Recherche_Tous_Modèles (Modèle)
    input: la table dans laquelle s'effectue la recherche
    output: tous les modèles proposés

Recherche_Toutes_Catégories(Catégorie)
    input: la table dans laquelle s'effectue la recherche

output: toutes les catégories proposées

Recherche_Un_Modèle(Modèle)
input: la table dans laquelle s'effectue la recherche
output: Un modèle donné

Recherche_Une_Catégorie(Catégorie)
input: la table dans laquelle s'effectue la recherche
output: Une catégorie donnée

Recherche_Toutes_Versions(Modèle, Version)
input: les tables dans lesquelles s'effectue la recherche
output: toutes les versions proposées accompagnées du modèle auquel elles se réfèrent

Recherche_Une_Version(Modèle, Version)
input: les tables dans lesquelles s'effectue la recherche
output: Une version donnée

Recherche_Séries_Spéciales(Modèle, Version)
input: les tables dans lesquelles s'effectue la recherche
output: **toutes** les séries spéciales proposées accompagnées du modèle auquel elles se réfèrent

Recherche_Séries_Spéciales(Modèle, Version)
input: les tables dans lesquelles s'effectue la recherche
output: **Une** série spéciale donnée

Recherche_Occasions(Modèle, Version)
input: les tables dans lesquelles s'effectue la recherche
output: toutes les occasions proposées accompagnées du modèle auquel elles se réfèrent

Recherche_Véhicules_De_Direction(Modèle, Version)
input: les tables dans lesquelles s'effectue la recherche
output: tous les véhicules de direction proposés accompagnés du modèle auquel ils se réfèrent

Recherche_Véhicules_De_Stock(Modèle, Version) {non implémenté}
input: les tables dans lesquelles s'effectue la recherche
output: tous les véhicules de stock proposés accompagnés du modèle auquel ils se réfèrent

Rechercher_Liste_Critères_Impératifs(Modèle, Liste_de_critères_Impératifs)
input: la table dans laquelle s'effectue la recherche
une liste de critères auxquels doit répondre impérativement le modèle
output: tous les modèles qui répondent entièrement aux critères

Rechercher_Liste_Critères_Impératifs (Version, Liste_de_critères_Impératifs)
input: la table dans laquelle s'effectue la recherche
une liste de critères auxquels doit répondre impérativement le modèle

output: toutes les versions qui répondent entièrement aux critères

Les deux fonctionnalités qui vont suivre se distinguent des deux précédentes par le fait qu'elles ne se content pas de trouver un modèle ou une version qui répond exactement à une ou plusieurs conditions (par exemple le nombre de portes ou le type d'alimentation), mais essaient de trouver des modèles ou des versions qui répondent *plus ou moins* aux conditions exigées (c'est par exemple la cas lorsqu'on veut équiper un véhicule avec des options d'un certain type - confort ou sécurité par exemple - mais que le prix ne peut pas dépasser une certaine borne supérieure). Il est fort probable que le véhicule recherché n'existe pas exactement dans la base de données, il faut donc développer un algorithme de recherche spécifique que nous étudierons par la suite.

Rechercher_Liste_Critères_Approximatifs(Modèle, Liste_de_critères_ Approximatifs)
input: la table dans laquelle s'effectue la recherche
une liste de critères auxquels doit répondre approximativement le modèle
output: tous les modèles qui répondent entièrement aux critères

Rechercher_Liste_Critères_ Approximatifs(Version, Liste_de_critères_ Approximatifs)
input: la table dans laquelle s'effectue la recherche
une liste de critères auxquels doit répondre approximativement le modèle
output: toutes les versions qui répondent entièrement aux critères

Calcul_Remise(Version)
input: le véhicule sur lequel on calcule la remise
output: la remise est calculée et le prix réel du véhicule qu'on désire acheter est mis à jour

Calcul_Reprise(Version)
input: le véhicule sur lequel on calcule la reprise
output: la reprise est calculée et le prix réel du véhicule qu'on désire acheter est mis à jour

Calcul_financement(Somme, Mensualités)
input: la somme pour laquelle on désire un financement
le nombre de mensualités désirées pour effectuer le remboursement
output: un tableau donnant les possibilités pour la somme et les mensualités désirées

Impression_Catalogue_Personnalisé(Véhicule, Client, Vendeur)
input: le véhicule auquel le client s'intéresse
le client qui s'intéresse au véhicule
le vendeur qui s'est chargé du client
output: un catalogue reprenant le véhicule auquel s'intéresse le client avec différentes caractéristiques, ainsi que l'équipement supplémentaire dsiré par le client et la liste des options supplémentaires et accessoires disponibles sur le véhicule

Modification_Couleur_Habillage(Version)
input: le véhicule auquel on s'intéresse
output: la liste des couleurs avec ses habillages intérieurs disponibles et vice versa

Recherche_Caractéristiques_Techniques(Version)
input: le véhicule auquel on s'intéresse
output: la liste des caractéristiques techniques plus importantes de ce véhicule relatifves au poids, la dimension, performances,...)

Les fonctionnalités qui suivent ne sont pas encore complètement implémentées dans la version Professional, mais l'interface a déjà en partie été prévue. En fait le modèle conceptuel des données que nous avons développé jusqu'à présent ne prévoyait pas les données manipulées par ces traitements. Nous les citons juste par pure formalité, et seront peut-être développées dans une version future du logiciel. Cela demandera d'étendre le modèle conceptuel des données, ainsi que la base de données. La version User Guide ne les prévoit pas non plus, car, rappelons-le, elle ne s'adresse pas aux concessionnaires, mais à tout le monde. Il n'est donc pas utile de prévoir une gestion de clients, des ventes, de statistiques et autres fonctionnalités qui sont spécifiques des concessionnaires et garages.
Le motif pour lequel ces fonctions n'ont pas été développées a été la décision d'arrêter temporairement le développement d'un logiciel spécifique des concessionnaires pour s'attaquer à une version qui pourrait être plus diffuse que celle-ci. Nous y reviendrons plus loin lorsque nous parlerons des fonctionnalités attachées à la version User Guide.

Impression_Bon_de_Commande(Version, Liste_Options, Liste_Accessoires, Client)
input: le véhicule que le client désire acheter
la liste des options qui équiperont le véhicule
la liste des accessoires qui équiperont le véhicule
les coordonnées du client qui achète le véhicule
output: un bordereau de commande en double exemplaire
une nouvelle commande
les statistiques mises à jour (nous ne spécifierons pas plus ces outputs)

Encoder_Nouveau_Client(Client, Coordonnées_Client)
input: la table dans laquelle s'effectue l'encodage
les coordonnées du client que l'on veut introduire (pas déjà encodé auparavant)
output: une nouvelle occurrence de client dans la table des clients Client

Rechercher_Client(Client, Coordonnées_Client)
input: la table dans laquelle s'effectue la recherche
les coordonnées du client que l'on recherche (déjà encodé auparavant)
output: la fiche signalétique du client

Effacer_Client(Client, Coordonnées_Client)
input: la table dans laquelle s'effectue la mise à jour
les coordonnées du client que l'on veut effacer (il doit exister)
output: une occurrence de client en moins dans la table des clients Client

Mise_à_jour_Client(Client, Coordonnées_Client)
input: la table dans laquelle s'effectue la mise à jour
les coordonnées du client que l'on veut mettre à jour (déjà encodé auparavant)
output: la table Client mise à jour

Rédaction_Mailing(Client, Mail)
input:
output: une lettre écrite

Envoi_Mailing(Client, Mail)
input: les coordonnées du client auquel on envoie un mail ou une lettre
la lettre qu'on envoie
output: une lettre envoyée

Réception_Mailing(Client, Mail)
input: les coordonnées du client qui envoie un mail ou une lettre
la lettre qu'on reçoit
output: une lettre reçue

Recherche_Mailing(Client, Mail)
input: une lettre qu'on a envoyé à un client ou qu'on a reçu d'un client
les coordonnées du client
output: une lettre envoyée

Statistiques_Client(Client, Liste_Objets_de_statistique)
input: les coordonnées du client sur lequel on veut faire une statistique (éventuellement tous)
une liste d'objets sur lequels porte la statistique (ex: nombre de voitures diesel vendues au courant du mois, réparties par clients privés et sociétés)
output: un graphique et une explication de la statistique (formule,...)

Archivage_Statistique(Liste_Statistiques)
input: une liste de statistiques
output: les archives de statistiques mises à jour

Recherche_Statistique(Archives_Statistiques, Objet_de_statistique)
input: les archives des statistiques
l'objet sur lequel porte la recherche
output: un tableau ou une explication de la statistique si elle existe

Effacer_Statistique(Archives_Statistiques, Objet_de_statistique)
input: les archives des statistiques
l'objet sur lequel porte la statistique que l'on veut effacer
output: une occurrence de statistique de moins dans les archives si cette statistique existait

Calcul_Automatique_des_Statistiques(Archives, Liste_Objets_de_statistiques)
input: les archives des statistiques
une liste d'objets sur lesquels portent les statistiques que l'on veut calculer
output: les archives concernant la liste d'objets de statistiques mise à jour

3.4 .1.2 Commentaires

Ces fonctionnalités recouvrent la quasi totalité des opérations qui sont effectuées tous les jours chez un concessionnaire. Le logiciel étant un système d'aide à la décision dans le domaine de la **vente** automobile et non pas une gestion de concessionnaires de voitures, il n'a pas semblé utile d'insister sur ces fonctionnalités, mais plutôt sur celles qui intéressaient directement la manière de choisir un véhicule. Ces autres fonctions gravitent un peu autour de l'activité principale qui est choisir une voiture. Le reste est directement et facilement automatisable puisqu'il ne s'agit que de petites applications d'impressions ou d'encodage de données stockées dans une base de données. Il existe déjà une multitude de logiciels s'occupant très bien de ce type d'application, nous n'insisterons donc pas ici

3.4 .2 Sensitive Car User Guide

Nous allons à présent nous intéresser à la version User Guide du logiciel qui, comme cela a déjà été annoncé précédement, voit une autre approche du problème puisqu'ici il s'agit également de choisir la marque du véhicule auquel on s'intéresse. De plus, nous ne nous occupons plus de problèmes relatifs à des garages, puisque ce logiciel est destiné à un public plus vaste. En fait, il faut rappeler pourquoi il a été décidé de passer à une approche différente. Le fait d'avoir découvert d'autres logiciels ayant les mêmes objectifs montre qu'il y a un intérêt certain pour ce type de logiciel. Il faut donc faire au moins aussi bien que ces logiciels dans le cadre de ce travail.

L'accent sera mis encore plus sur la capacité du logiciel à proposer un véhicule sur base des désirs de l'utilisateur et à répondre à toutes les questions qu'on peut se poser à propos des voitures reprises dans la base de données, pour autant que la réponse s'y trouve, bien entendu. L'interface sera encore plus convivale, plus facile d'utilisation et d'apprentissage en un minimum de temps.

L'utilisateur aura le choix entre deux méthodes afin de choisir un véhicule: la méthode par guidage d'interface et la méthode par requêtes directes à la base de données. Il faudra donc développer une méthode particulière afin d'accéder facilement aux informations contenues dans la base de données.

Nous n'aurons qu'à revoir les traitements puisque la base de données avait déjà été prévue pour cette nouvelle application. Ceux-ci seront un peu plus complexes puisqu'il faut également envisager de ne considérer qu'une nationalité de constructeur ou qu'une marque. Dans la version Professional, le problème ne se posait pas puisque d'office la nationalité et la marque étaient imposées.
Les **principales nouveautés** intéressantes sont donc l'**interrogation de la base de données et le classement de véhicules choisis d'après des critères par une méthode multicritère.**

3.4 .2.1 Commentaires

Comme nous le voyons, cette version du logiciel est uniquement axée sur le choix d'un véhicule et élimine un tas de fonctions accessoires qui font la particularité de la version Professional. L'appellation système d'aide à la décision est donc bien d'application puisque tout a été pensé de telle sorte que l'utilisateur ne reste jamais sans réponse à une question. Nous verrons par la suite comment développer ces fonctionnalités en les intégrant directement dans l'interface, car ne l'oublions pas, celle-ci doit être pensée en parallèle, durant le développement même du logiciel et non par après. Il n'est donc pas question de spécifier de trop *comment* ces fonctions seront implémentées dans les détails. Contentons-nous en ce moment, en anticipant légèrement sur la suite, de savoir que toutes ces fonctions se traduiront en requêtes SQL sur la base de données, mais que certaines d'entre elles feront l'objet d'un chapitre entier parce que ce sont elles qui représentent le point crucial du logiciel, à savoir:

Rechercher_Liste_Critères_Approximatifs(Marque, Modèle,
Liste_de_critères_ Approximatifs)

Rechercher_Liste_Critères_Approximatifs(Marque, Modèle, Version,
Liste_de_critères_ Approximatifs)

Rechercher_Requête(Base_de_données, Requête)

Cette dernière fonction représente une grosse partie du logiciel. Nous en reparlerons en détail plus loin, car il va falloir recréer un modèle conceptuel des données, non pas sur les véhicules, mais à propos du modèle conceptuel des données relatif à l'application *Sensitive Car*. En fait nous pouvons déjà dire que nous ferons un méta-modèle du modèle, pour des raisons que nous expliquerons, bien entendu.

Les deux premières fonctions quant à elles travailleront avec une méthode d'analyse multi-critères que nous développerons spécifiquement pour cette application. C'est pour cela que les critères sont appelés approximatifs, car il faudra que le ou les véhicules trouvés répondent *au mieux* à la liste de critères envisagée. Nous verrons ce que signifie ce mot "au mieux".

3.4 .2.2 Liste des fonctionnalités

Recherche_Toutes_Nations(Nations)

Recherche_Nation(Nations, Continent)

Recherche_Toutes_Marques(Marque)

Recherche_Marques(Marque, Critère)

Recherche_Tous_Modèle(Marque, Modèle)

Recherche_Tous_Modèle(Marque, Modèle, Critère)

Recherche_Toutes_Catégorie(Catégorie, Marque)

Recherche_Toutes_Catégorie(Catégorie, Marque, Critère)

Recherche_Toutes_Versions(Marque, Modèle, Version)

Recherche_Séries_Spéciales(Marque, Modèle, Version)

Rechercher_Liste_Critères_Impératifs(Modèle, Liste_de_critères_Impératifs)

Rechercher_Liste_Critères_Impératifs (Version, Liste_de_critères_Impératifs)

Les deux fonctionnalités qui vont suivre se distinguent des deux précédentes par le fait qu'elles ne se contentent pas de trouver un modèle ou une version qui répond exactement à une ou plusieurs conditions (par exemple le nombre de portes ou le type d'alimentation), mais essaient de trouver des modèles ou des versions qui répondent *plus ou moins* aux conditions exigées (c'est par exemple la cas lorsqu'on veut équiper un véhicule avec des options d'un certain type - confort ou sécurité par exemple - mais que le prix ne peut pas dépasser une certaine borne supérieure). Il est fort probable que le véhicule recherché n'existe pas exactement dans la base de données, il faut donc développer un algorithme de recherche spécifique que nous étudierons par la suite.

Rechercher_Liste_Critères_Approximatifs(Marque, Modèle,
Liste_de_critères_ Approximatifs)

Rechercher_Liste_Critères_ Approximatifs(Marque, Modèle, Version,
Liste_de_critères_ Approximatifs)

Rechercher_Liste_Critères_ Approximatifs(Version, Liste_de_critères_ Approximatifs)

Rechercher_Requête(Base_de_données, Requête)
{Propre à la version User Guide}

Calcul_financement(Somme, Mensualités, Taux_d_intérêt)
{Cette fonction reste afin qu'un utilisateur puisse se faire une idée des mensualités lors d'un financement quelconque}

Impression_Catalogue_Personnalisé(Véhicule)
{ Cette fonction reste pour un simple motif: lorsque par exemple un utilisateur a choisi un véhicule avec un certain équipement, il imprime un catalogue du véhicule désiré et peut ainsi se rendre chez un concessionnaire en lui facilitant le travail}

Modification_Couleur_Habillage(Version)

Recherche_Caractéristiques_Techniques(Version)

4. CONCEPTION DE L'INTERFACE

Le but de ce chapitre est de concevoir l'interface qui supportera les fonctionnalités énoncées plus haut. Cette partie du développement, comme déjà annoncé, n'est pas une partie séparée de la conception, mais vient en parallèle.
Ceci étant dit, il faut préciser que le processus de développement d'une interface n'étant pas une science exacte, mais un ensemble de règles basées sur une grande partie de considérations cognitives, il n'a pas toujours semblé utile de respecter aveuglément ces règles, car ici aussi il y avait des contraintes non spécifiables de manière formelle. Nous les reprenons ci-après.

4.1 CONTRAINTES RELATIVES A L'INTERFACE

Celles-ci sont reprises sous une forme de règles devant absolument être respectées par l'interface et nous ferons une vérification a posteriori, lorsque nous aurons obtenu les écrans qui formeront l'interface entre les fonctionnalités et l'utilisateur, afin de prouver que ces écrans que nous aurons développé respectent bien toutes ces règles. Outre celles-ci, il faut bien entendu que l'interface repecte également les règles ergonomiques usuelles. Pour cela, nous renvoyons le lecteur à [BODART94], [SACRE92] et [VANDERDONCKT92] dans le cadre de la conception ergonomique d'une interface.

Il est intéressant de reprendre les critères ergonomiques que doit respecter une interface, ainsi nous pourrons vérifier que l'interface obtenue est ergonomiquement correcte. Pour résumer ces règles, ramenons-nous aux règles d'or de conception d'une Interface Homme-Machine [IHM95] sans les décomposer dans les détails.

1) La compatibilité

Définition: Le (re)codage d'informations et de tâches du monde réel en données et actions du système est réduit. En effet, une interface est d'autant meilleure qu'elle est compatible, car le transfert d'informations est d'autant plus rapide que le (re)codage en données est réduit, l'accomplissement de la tâche est d'autant plus rapide que son interprétation en actions est courte. La compatibilité est ici interprétée comme une cohérence avec l'environnement extérieur de l'application.
Objectif: Le but est de réduire le besoin de traduire, de transposer, d'interpréter l'information en données du système, de raccourcir l'interprétation de la tâche en actions du système, de minimiser les références à la documentation lors de l'évaluation.
Décomposition:

1.1 compatibilité comportementale : compatibilité avec d'autres attentes, les habitudes et les méthodes de travail de l'utilisateur;

Il semble évident de choisir un véhicule en s'intéressant aux catégories, modèles, marque,... Voir le menu en annexes A - page 1

1.2 compatibilité sémantique : compatibilité avec la sémantique du monde réel;

Les termes utilisés sont les termes habituellement utilisés. Voir annexes A - B1 - B2 - B3

1.3 compatibilité syntaxique :

1.3.1 compatibilité opérationnelle : compatibilité avec l'ordre des opérations, les procédures en place;

1.4 compatibilité lexicale

1.4.1 compatibilité de support : compatibilité entre les saisies/affichages et les supports utilisés pour le transfert d'informations;
1.4.2 compatibilité grammaticale : compatibilité avec la terminologie de l'utilisateur
1.4.3 compatibilité linguistique : compatibilité avec la langue naturelle de l'utilisateur;

Il est possible de choisir la langue désirée. Voir annexes A - page 1, B1 - page 7, B1 - page 9

2) La cohérence

Définition: Une IHM est cohérente si et seulement si les données et les actions sont facilement identifiables, reconnaissables et utilisables. En effet, les données sont d'autant mieux perçues et les actions d'autant mieux accomplies qu'elles sont présentées de manière stable et uniformisée. Ce critère concerne la cohérence d'une application avec d'autres applications ou au sein d'une application.

Objectif: Le but est de recourir aux mêmes moyens pour arriver aux mêmes résultats dans des contextes similaires. En standardisant l'interface et les procédures, on favorise l'instauration d'une interface prédictible dans laquelle l'utilisateur connaît à l'avance le résultat.

Décomposition:

2.1 cohérence inter-application : cohérence à tous niveaux entre plusieurs applications, par exemple au sein d'un projet;
2.2 cohérence intra-application : cohérence à tous niveaux au sein d'une même application;
2.2.1 cohérence pragmatique : cohérence entre la métaphore, le modèle véhiculé par l'interface et la tâche utilisateur;
2.2.2 cohérence sémantique : cohérence au sens attribué aux objets, à leurs propriétés, leurs relations;
2.2.3 cohérence syntaxique : cohérence de la syntaxe et de l'ordre des procédures, des saisies, des menus;
2.2.3.1 cohérence opérationnelle : cohérence des actions;
2.2.3.2 homogénéité : cohérence des séquences d'actions similaires pour parvenir à un résultat identique;
2.2.4 cohérence lexicale : cohérence du choix et de l'utilisation des libellés, des dénominations des objets interactifs concrets (OIC), des items de menus;
2.2.4.1 cohérence spatiale : cohérence des OIC;
2.2.4.2 cohérence grammaticale : cohérence de la structure grammaticale des libellés, des dénominations;
2.2.4.3 cohérence linguistique : cohérence de la langue de dialogue avec l'utilisateur;
2.2.5 cohérence alphabétique : cohérence des symboles alphabétiques, graphiques;
2.2.6 cohérence physique : cohérence de la lumière (contraste, clignotement), du son (fréquences), du mouvement (vitesse de déplacement du curseur).

Nous pouvons aisément vérifier que les objets utilisés le sont toujours de la même manière. Exemple: (dé)sélection d'objets: les objets sont toujours présentés dans une liste et le résultat de la (dé)sélection est une autre liste. L'opération de (dé)sélection se fait de manière standard à l'aide de boutons indiquant la direction des deux listes. Un objet courant au sein des listes est toujours sélectionné (sauf si elles sont vides).
Voir annexes A - page 9, A - page 10, B1 - page 3, B1 - page 4 - B1 - page 5 et autres.

3) La charge de travail

Définition: Une IHM est efficace en charge de travail si et seulement si le volume de données à manipuler et d'actions à accomplir par unité de tâche est réduit. En effet, l'interaction est d'autant plus rapide que les actions de l'utilisateur portant sur un nombre limité de données sont courtes: l'utilisateur est d'autant plus efficace lors de l'accomplissement de sa tâche qu'il est moins distrait par des informations étrangères à sa tâche.

Objectif: Les buts sont de garantir la charge de travail dans les limites de capacité des facultés humaines et de garantir une performance.

Décomposition:

3.1 performance : objectif de réaliser une performance optimale dans l'accomplissement de la tâche;

3.2 charge mentale : respect de la capacité de la mémoire à court terme, des facultés de vision;

3.3 brièveté

- 3.3.1 concision : charge informationnelle véhiculée par les saisies/affichages de données;
- 3.3.2 actions minimales : charge véhiculée par le nombre minimal d'actions à effectuer pour accomplir une tâche;

3.4 charge symbolique : limitation du nombre de symboles employés;

3.5 respect cognitif : respect des limites cognitives de l'utilisateur humain;

3.6 respect physique

- 3.6.1 respect physiologique : respect des processus neurophysiologiques de la vue, de l'ouïe et du toucher de l'utilisateur humain, des habiletés sensori-motrices;
- 3.6.2 respect perceptuel : compte tenu des capacités de vision et de différentiation visuelles de l'être humain.

Nous avons minimisé le nombre d'écrans afin d'arriver à un véhicule (minimum 3). Chaque écran possède un titre afin d'indiquer la tâche en cours. Les informations présentées sont limitées à la tâche en cours. Certains icônes sont utilisés afin de respecter la règle 3.6.2. Voir annexes B1 - page 3, 4, 5

4) L'adaptabilité

Définition: Une IHM est adaptable si et seulement si elle possède la faculté de mimétisme comportemental vis-à-vis de son utilisateur. En effet, l'utilisateur est d'autant moins dérouté et acquerra d'autant plus d'expérience que l'interface peut s'adapter aux différents contextes de travail

Objectif: Le but est de fournir à l'utilisateur différentes voies pour accoplir sa tâche qui peuvent varier en fonction de différents paramètres.

Décomposition:

4.3 flexibilité syntaxique : possibilité de faire varier la présentation, la conversation lors de saisies/affichages;

- 4.3.1 flexibilité opérationnelle : possibilité de faire varier les actions de l'utilisateur (saisie, affichage, contrôle) ainsi que leur ordre;
  - 4.3.1.1 réversibilité des actions : capacité du système à revenir à un état antérieur connu suffisamment stable;
- 4.3.2 structuration des actions : aptitude à organiser la conversation, grouper les actions de l'utilisateur en niveaux de complexité différente suivant le fond;

4.4 flexibilité lexicale

4.4.1 flexibilité de présentation : possibilité de faire varier la présentation;

4.4.1.1 structuration de la présentation : aptitude à organiser la présentation, grouper les données en niveaux de complexité différente suivant la forme;

4.4.2 flexibilité linguistique : possibilité de faire varier la langue du système en fonction de la langue de l'utilisateur.

Il est toujours possible de revenir à un état antérieur ou annuler les données fournies au système, afin de revenir à un état cohérent. De plus, l'utilisateur peut choisir la langue qui lui convient. (Dans la version User Guide, il pourra également configurer la couleur des écrans et des différents objets, ainsi qu'utiliser d'autres polices de caractères. cfr. panneau de configuration de Windows®. Voir annexes B1 - pages 1, 7, 8

5) Le contrôle de dialogue

Définition: Une IHM est à contrôle explicite si et seulement si elle peut fournir à l'utilisateur l'apparence, l'illusion d'être placée sous contrôle de l'utilisateur en exécutant des actions suite aux demandes explicites de ce dernier.

Objectif: Le but est de laisser l'utilisateur contrôler le déroulement du dialogue autant que possible, de compléter une action uniquement lorsque l'utilisateur en spécifie le but.

Décomposition:

5.1 contrôle de la conversation

5.1.1 actions explicites : le déclenchement des actions doit incomber à l'utilisateur, non au système;

5.2 contrôle de la présentation

5.2.2 présentation automatique : le système ajuste automatiquement la présentation sans intervention de l'utilisateur;

Les actions sont explicites: le logiciel attend un "ordre" de la part de l'utilisateur et ne déclenche qu'une action à la fois, celle-ci étant attendue de l'utilisateur. Voir annexes A, B1, B2, B3 en général.

6) La représentativité

Définition: Une IHM est représentative si et seulement si les codes utilisés, les items de menu, les libellés facilitent l'encodage, la rétention.

Objectif: Le but est de répandre l'usage de dénomination significatives au sein du dialogue.

Décomposition:

6.1 représentativité dans la conversation

6.1.1 raccourci : les raccourcis-clavier ou -icône doivent être facilement associables avec la procédure attachée;

6.2 représentativité dans la présentation

6.2.1 abréviation : les abréviations doivent rappeler le mot suffisamment facilement;

6.2.2 codification : les codes choisis pour représenter les états, les données doivent être représentatifs.

Voir annexes A - page 1 (les icônes), page 3 (les items) et autres écrans en général

7) Le guidage

Définition: Une IHM est efficace en guidage si et seulement si elle informe de manière constante l'utilisateur sur l'issue de ses actions et sur sa position dans l'accomplissement de sa tâche. En effet, l'utilisateur réalise d'autant mieux sa tâche qu'il est guidé à travers toutes les étapes pour la mener à bien.

Objectif: Le but est de fournir à l'utilisateur une aide sur ce qu'il peut entreprendre, sur la situation dans laquelle il se trouve et sur les résultats des actions effectuées; on demande en outre de faire attention à la lisibilité.

Décomposition:

7.1 guidage dans la conversation
- 7.1.1 invitation : inviter l'utilisateur sur l'état courant et sur l'état qu'il souhaite atteindre;
- 7.1.2 progression : informer l'utilisateur sur l'avancement d'une action;
- 7.1.3 feed-back immédiat : informer l'utilisateur du résultat d'une action entreprise;

7.2 guidage dans la présentation
- 7.2.1 groupement/distinction entre objets
  - 7.2.1.1 groupement/distinction par le placement : guidage résultant du placement des objets interactifs de l'interface;
  - 7.2.1.2 groupement/distinction par le format : guidage résultant d'un format d'écran prédéterminé;
- 7.2.2 guidage visuel, clarté : informer l'utilisateur sur les relations de similitude,de différence ou de sémantique reliant les objets interactifs affichés;

A chaque écran est associé une aide qui explique à l'utilisateur ce qu'il peut faire et ce qu'il doit faire, ainsi que le résultat de ses choix. Dans la version User Guide, l'aide sera un help Windows®, permettant de rechercher un mot, de se référer à l'aide d'un autre écran,... Voir annexes B1 - page 12

8) L'efficacité en gestion des erreurs

Définition: Une IHM est efficace en gestion des erreurs si et seulement si elle s'avère robuste aux erreurs commises par l'utilisateur et se montre conviviale dans la manière de les corriger. En outre, la performance de réalisation d'une tâche est d'autant meilleure que les occasions d'erreurs sont réduites.

Objectif: Le but est d'éviter les erreurs autant que possible.

Décomposition:

8.1 protection vis-à-vis des erreurs : résister à la propension d'introduire une erreur, protéger les zones ne contenant pas de saisies et valider les saisies autant que possible;

Il n'est pas possible d'encoder de valeurs à la main, mais uniquement à l'aide de la manière d'encoder fournie par le logiciel. Ceci afin d'éviter les erreurs d'encodage, mais également afin de ne pas sortir des bornes autorisées par le système.
Voir annexes A - pages 3, 14, B2 - page 2, B3 - page 1

8.2 identification des erreurs
- 8.2.1 cause de l'erreur : identifier le(s) motif(s) de l'erreur survenue
- 8.2.2 lieu des erreurs : déterminer de manière suffisamment explicite l'endroit de l'erreur;

Une erreur peut survenir exclusivement après avoir appuyé sur un bouton de commande. Le lieu est donc toujours l'endroit où on était. Voir annexes...

8.3 explicitation des erreurs : expliquer de manière complète l'erreur et le moyen de la corriger, le cas échéant;

Un message est prévu pour chaque type d'erreur possible (le logiciel a été fait de telle sorte que celles-ci soient limitées au maximum. Voir règle 8.1). Voir annexes...

8.4 correction des erreurs : prévoir le moyen de corriger toute erreur survenue.

Il suffit de recommencer à l'endroit où on était. Voir annexes B2 - page 2 (Bouton Annuler) B3 - page 1 (Bouton Annuler les critères).

Nous distinguons, comme précédement, la version Professional de la User Guide, car la conception de l'interface répondait à des objectifs distincts puisque les fonctionnalités étaient différentes.

4.2 SENSITIVE CAR PROFESSIONAL

Afin de développer l'interface, il faut garder à l'esprit quelques objectifs importants de ce logiciel. Tout d'abord, il s'adresse à des professionnels de la vente dans le domaine automobile. Le but est de leur faciliter le travail, en les déchargeant d'une série de tâches fastidieuses et entre autre de devoir se rappeler les prix, les options disponibles sur un véhicule donné, les détails techniques d'une version, faire des offres, comparer des véhicules, diriger un client vers un type de véhicule, contenter au mieux un client. Pour ce dernier objectif, il faut déjà remarquer qu'il n'est pas formalisable, en ce sens qu'il n'est pas possible de concevoir un système formel basé sur le contentement d'une personne, car le contentement est quelque chose de subjectif sujet à réflexion et à changement. Or, on ne peut pas retenir ces informations dans une base de données. C'est pourtant un objectif important du logiciel et il faut que l'interface supporte cette "fonctionnalité".

D'autre part, *Sensitive Car* peut être utilisé comme borne d'information dans un show-room et sera donc utilisé non pas par un vendeur, mais par un client qui n'aura donc aucune formation particulière, ni en informatique, ni avec le logiciel. Pour cela, certaines fonctionnalités ne devront pas être disponibles au client (Calcul de la remise et de la reprise, gestion des clients,...).

Afin que tout cela soit possible, l'interface a été conçue avec les impératifs suivants:

- Facilité d'utilisation: disponibilité d'une aide efficace, l'utilisation est réduite à sa plus simple expression: appuyer sur des boutons. Clarté des écrans. Voir annexes A - pages 1, 2 et suivantes.
- convivialité de l'interface. Le logiciel communique avec l'utilisateur soit via des résultats à un ordre, soit via un message. Voir annexes B2 - pages 2 et 3: la page 3 est le résultat de la requête exprimée à la page 2, après avoir appuyé sur le bouton Résultats.

• Nombre minimum d'écrans afin de ne pas se noyer dans un paysage fait d'écrans. En effet, lorsque cela a été possible, il a été utilisé plusieurs fois le même écran afin d'effectuer des tâches similaires. Ceci est surtout vrai pour la version User Guide (le choix des modèles, catégories, nations et marques se fait avec le même écran). Voir annexes B1 - pages 3, 4 et 5.
• Eliminer l'utilisation du clavier ou la rendre aussi minime que possible (utilisation d'un écran tactile ou d'une souris). Le clavier est inutile. Tout encodage se fait à l'aide de potentiomètres. Voir annexes B3 - page 1
• Ressemblance des écrans, afin que l'utilisateur ait l'impression de se trouver toujours au même endroit. Voir annexes B1 - pages 3, 4 et 5 (cfr. nombre minimum d'écrans).
• Clarté des commandes. Un mot résume souvent une commande Voir annexes B1 - pages 1 et 2.
• Utilisation maximum des métaphores graphiques. Utilisation d'icônes pour mimétiser certaines commandes (impression, information techniques, (dé)sélectionner,...).Voir annexes A - page 4.
• Redondance des fonctionnalités: Possibilité d'arriver de plusieurs manières au même résultat
• Concision, efficacité et disponibilité de l'aide pour chaque écran. A aucun moment il ne faut que l'utilisateur se demande ce qu'il peut ou ce qu'il doit faire sans avoir de réponse. Voir tous les écrans en annexe.
• Rapidité des résultats fournis: les procédures de recherche doivent rendre des résultats aussi vite que possible, on accepte un délai de cinq secondes entre le moment où la recherche est lancée et le moment où les résultats sont affichés. Voir annexes A - page 14, B3 - page 1 pour la recherche d'un véhicule sur base de critères.
• Communication du programme avec l'utilisateur soit via des résultats, soit via des messages clairs et concis qui résument la situation et ce qui va se passer.

L'interface étant le moyen de communication entre les données et l'utilisateur, le développement de celle-ci représente un point crucial du logiciel, car si jusqu'à présent nous avons suivi une démarche habituelle de développement de logiciel (analyse conceptuelle, développement de la base de donnée, spécification - de manière tout à fait informelle ici - des traitements), le produit qui sera présenté le sera au travers de l'interface; ce qui apparaîtra du logiciel ce ne seront ni le modèle entité-association, ni la base de données ni les différentes spécifications que l'on peut faire, mais ce seront les écrans. Si ceux-ci sont bien conçus, le logiciel sera accepté. Autrement ce sera un échec complet, même si le développement a été fait de manière parfaite d'un point de vue informatique. N'oublions pas que ce ne seront pas des informaticiens à utiliser le logiciel, mais des vendeurs au contact avec des clients. Il faut que le produit soit convainquant pour le vendeur qui l'utilise et pour le client qui voit les résultats de l'utilisation sous forme d'un service rendu ou une information cohérente fournie. Le logiciel **doit** être perçu comme une **Aide** et non comme un fardeau, autrement quel système d'aide serait-il.

Nous allons suivre la méthodologie proposée dans *"Dimensions clé pour une méthodologie de développement d'applications interactives"* afin de converger vers une interface conviviale et correcte, avec toutefois certaines restrictions dues au domaine d'activité qui nous intéresse, les systèmes d'aides à la décision.

Dimension 1: Formalisation des spécifications de l'IHM à partir de l'analyse de la tâche

1.1 *Analyse de la tâche*

Une *tâche* est définie comme une activité dont l'accomplissement par un opérateur (utilisateur) produit un changement d'état significatif d'un domaine d'activité dans un domaine donné.

Dans notre contexte spécifique, la tâche est la recherche d'un véhicule qui répond à certains critères, peu importe lesquels, objectifs ou non. Or, d'après la définition que nous venons de donner, on pourrait croire que le choix d'un véhicule se limite à quelque chose de très objectif comme identifier un client ou passer une commande d'un produit. Ceci limiterait fortement la capacité de l'être humain, c'est pourquoi nous ne pouvons nous limiter à suivre la démarche proposée, car choisir un véhicule est un type de tâche faiblement structuré. Au fond, jusqu'à présent, nous n'avons pas encore défini ce que signifiait choisir un véhicule. C'est même quasi impossible, le phénomène est très difficile à appréhender. C'est pourquoi, malheureusement, nous nous limiterons volontairement à considérer que choisir un véhicule se résume à considérer les fonctionnalités que nous avons envisagées dans le chapitre qui s'y rapportait.

Bien que cela simplifie l'activité du cerveau humain et quoique nous en soyons conscients, faute de posséder une méthodologie qui puisse supporter le développement d'un système d'aide à la décision, nous poursuivons dans cette voie. Ceci est dû au fait que nous ne pouvons humainement pas envisager toutes les considérations que le cerveau humain peut avoir concernant un domaine donné, ici les voitures. Ce problème a été résolu en partie dans la version Professional par un module consacré aux critères et à l'importance accordée à ceux-ci, en supposant, de manière restrictive, que choisir un véhicule pouvait se limiter à considérer des critères. On a l'impression que le choix se limite à quelque chose de très rationnel, fonctionnel. Nous ne pouvons pas envisager des critères comme la beauté, la ligne, la fluidité, le coup de foudre, ces critères n'étant pas formalisables, même par une approche floue, car comment définir la beauté d'une voiture, par rapport à quel critère objectif et dans quelle proportion faut-il le considérer pour estimer que ce critère est respecté? En d'autres termes comment le pondérer et le quantifier? Cela résulte être impossible.

En conclusion, nous nous contenterons donc de considérer les fonctionnalités qui, *grosso modo*, n'ayant choqué personne lors des diverses démonstrations du logiciel et n'ayant reçu aucune autre proposition considérant une autre approche du problème, nous estimerons que le but est atteint.

Ceci étant dit, nous anticipons légèrement sur la suite en annonçant que le problème qui vient d'être cité a été résolu différement dans la version User Guide, en laissant l'utilisateur poser d'autres questions aux logiciel, celles-ci n'ayant pas été prévues à l'avance, proposer ses propres choix et se guider tout seul, selon ses envies et considérations, par rapport aux réponses reçues à ses questions. Nous en reparlerons lorsque le moment sera venu.

1.1.1. Identification des buts et des sous-buts

Dans l'approche que nous considérons, le changement d'état significatif du domaine d'activité est associé à un but principal. Un *but* constitue un état particulier du domaine à atteindre. Un but explicite ce pourquoi une tâche est accomplie. Un but est décomposable récursivement en unités plus petites appelées *sous-buts*. Chaque sous-but traduit un état intermédiaire du but considéré par lequel il faut passer pour atteindre un but.

Nos buts seront les fonctionnalités énoncées plus haut; reprenons-les pour rappel, mais uniquement celles qui concernent le choix du véhicule et non la gestion du concessionnaire (gestion client, statistiques, mailing...). Cette fois, nous allons en parler plus longuement puisque l'interface devra être construite relativement à la fonctionnalité, à ses inputs et ses outputs. Il faudra donc spécifier un peu plus ces deux éléments.
Remarquons toutefois que nous nous limitons à considérer un but qui est le choix d'un véhicule. Pour y arriver, nous décomposons ce but en sous-buts qui seront les fonctionnalités rattachées au choix d'un véhicule (il n'est toutefois pas nécéssaire de passer par tous les sous-buts pour atteindre le but. Ex: on n'est pas obligé de choisir la catégorie de véhicules, auquel cas, la recherche se fera toutes catégories confondues).

Recherche_Tous_Modèles (Modèle)
Recherche_Toutes_Catégories(Catégorie)
Recherche_Un_Modèle(Modèle)
Recherche_Une_Catégorie(Catégorie)
Recherche_Toutes_Versions(Modèle, Version)
Recherche_Une_Version(Modèle, Version)
Recherche_Séries_Spéciales(Modèle, Version)
Recherche_Séries_Spéciales(Modèle, Version)
Recherche_Occasions(Modèle, Version)
Recherche_Véhicules_De_Direction(Modèle, Version)
Recherche_Véhicules_De_Stock(Modèle, Version) {non implémenté}
Rechercher_Liste_Critères_Impératifs(Modèle, Liste_de_critères_Impératifs)
Rechercher_Liste_Critères_Impératifs (Version, Liste_de_critères_Impératifs)
Rechercher_Liste_Critères_Approximatifs(Modèle, Liste_de_critères_ Approximatifs)
Rechercher_Liste_Critères_ Approximatifs(Version, Liste_de_critères_ Approximatifs)
Modification_Couleur_Habillage(Version)
Recherche_Caractéristiques_Techniques(Version)

1.1.2. Identification des procédures

Comme nous l'avons déjà précisé ci-avant, nous ne pourrons malheureusement pas suivre à la lettre la méthodologie proposée, puisque nous ne savons pas expliciter de manière séquentielle les fonctionnalités à appliquer afin de choisir un véhicule. Dans certains cas, il suffira de se limiter au choix d'une version de base d'un modèle donné, tandis que dans d'autres il faudra faire intervenir une série de critères plus ou moins formels. Cela dépendra du type de client auquel on à faire, à son intention réelle d'acheter un véhicule, à sa connaissance dans le domaine automobile, à son attirance a priori pour une catégorie donnée ou un véhicule donné, et autres intérêts peu formalisables. De ce fait, nous allons, parmi les fonctions citées, en sélectionner certaines qui influenceront vraiment - ce sont des hypothèses simplificatrices de travail - le choix du véhicule. Nous verrons que considérer d'autres alternatives n'influencera pas vraiment le type d'interface que nous obtiendrons, car nous procéderons par enrichissement successifs de l'interface de départ. Nous expliquerons comment.

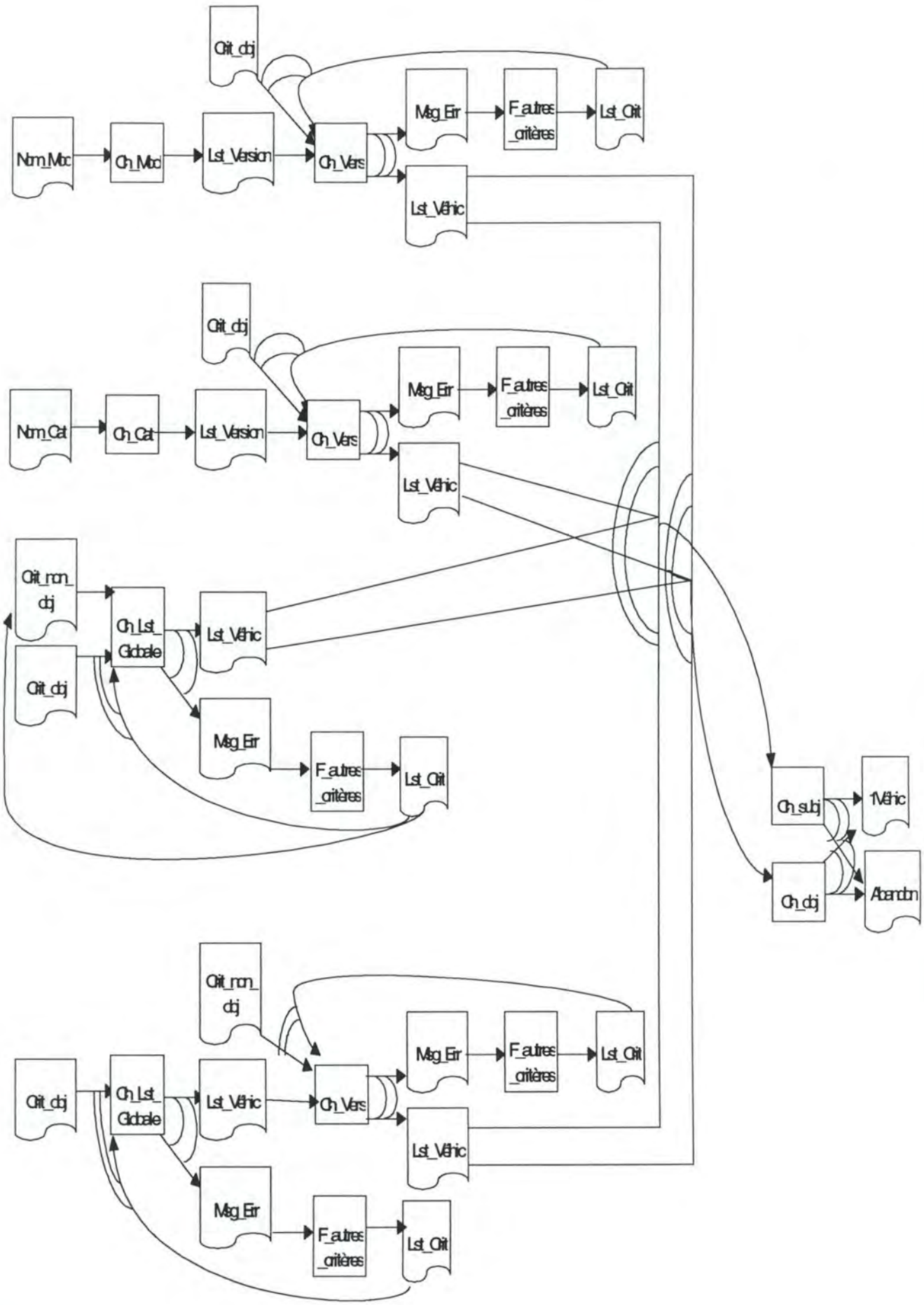
1.2 *Expression du produit de l'analyse de la tâche*

2. Identification des fonctions sémantiques de l'application

Voir les fonctions retenues utiles plus haut.

3. Rédaction d'un graphe d'enchaînement des fonctions

Voir ci-dessous.


Crit_obj
Nom_Moc
Ch_Moc
Lst_Version
Ch_Vers
Msg_Err
F_autres_critères
Lst_Crit
Lst_Véhic
Crit_obj
Nom_Cat
Ch_Cat
Lst_Version
Ch_Vers
Msg_Err
F_autres_critères
Lst_Crit
Lst_Véhic
Crit_rech_obj
Ch_Lst_Globale
Lst_Véhic
Crit_obj
Msg_Err
F_autres_critères
Lst_Crit
Ch_subj
1Véhic
Ch_obj
Abandon
Crit_rech_obj
Crit_obj
Ch_Lst_Globale
Lst_Véhic
Ch_Vers
Msg_Err
F_autres_critères
Lst_Crit
Lst_Véhic
Msg_Err
F_autres_critères
Lst_Crit

Interprétation du graphe d'enchaînement des fonctions

Comme nous n'avons pas la possibilité de décrire de manière séquentielle les fonctions qui s'enchaînent afin de conduire à un véhicule, nous prenons les conventions suivantes:

Les trois enchaînements considérés sont à interpréter comme des OU, c'est-à-dire qu'en fonction des désirs de l'utilisateur, le choix d'un véhicule se limitera parfois à un seul enchaînement, auquel cas il faut considérer que le lien entre les trois graphes est un OU et parfois ce peut être d'abord le choix d'un modèle, ensuite comme cet enchaînement n'a rien donné on devra passer à l'enchaînement successif (choix par catégorie) et si on n'a pas de résultat, on passe au troisième.

La liste Lst_Version obtenue est l'ensemble des versions correspondant au modèle donné. C'est une liste non vide (cardinalité 1-n entre modèle et version).

Crit_Obj est l'ensemble des critères objectifs que l'on peut demander à propos d'une version. Il s'agit du type d'alimentation, type de boîte de vitesse, nombre de portes, type de transmission (non prévu dans les versions actuelles du logiciel) et bien entendu le prix. Il s'agit des caractéristiques principales ("habituelles") qu'un client demande lors de l'achat d'un véhicule. Ce sont des conditions que le véhicule désiré doit impérativement remplir pour être sélectionné comme étant "bon".

La liste Lst_Véhic est une liste résultante de l'application des critères objectifs sur la liste des versions correspondant au modèle désiré.

Msg_Err est un message d'erreur au cas où tous les critères n'ont pû être satisfaits.
Il est à remarquer qu'étant donné que nous construisons un logiciel qui se veut efficace, il faut que l'interface soit prévue pour limiter les erreurs des utilisateurs. Nous pouvons *checker* jusqu'à un certain point les inputs des utilisateurs, il faut tout de même prévoir les cas impossible, c'est-à-dire des critères incompatibles. Il se peut également qu'il y ait de fortes chances que le prix qu'on donne ne corresponde à aucune version, mais qu'il existe une version, répondant aux critères, dont le prix est soit supérieur, soit inférieur (environ le prix demandé. Nous verrons plus loin ce que veut dire environ).

F_Autres_Critères est une fonction qui recalcule d'autres critères. Dans le cas où on choisit un véhicule à partir d'un modèle, on annule tous les critères choisis et on recommence si les critères donnés sont incompatibles.

Choix_Subj est une fonction ***non implémentable*** sensée dépendre de facteurs externes au logiciel et au monde automobile. Elle s'applique en tant que processus mental de l'utilisateur. C'est ici que se fera un choix final d'une seule version parmi la liste de véhicules répondant aux critères ou un abandon, car l'utilisateur reste indécis. On peut faire certaines suppositions quant aux facteurs qui influencent le choix. Est-ce le prix qui correspond exactement à celui demandé ou encore la photo proposée qui a beaucoup plu à l'utilisateur? Nous n'en savons rien.Tout dépend de facteurs inquantifiables. Nous supposerons donc pour notre modélisation des enchaînement des fonctions que cette fonction existe et qu'elle s'applique à un certain moment, mais nous ne nous intéressons pas au comment cette fonction est construite.

Choix_Obj est une fonction qui permet à l'utilisateur de choisir de manière objective (nous ne définissons pas ce que veut dire cela signifie, car cela aussi est impossible). Pour certains, il s'agira de comparer deux véhicules proposés en fonction de l'équipement de série, tandis que

pour d'autres, il s'agit d'avoir toute une série de tableaux de valeurs se rapportant à certaines vatiables jugées importantes dans le choix d'un véhicule.

En fait nous faisons comme si l'utilisateur comparait implicitement les véhicules entre eux et *quelque chose*, à la fin le faisait décider en faveur d'une seule version ou à l'abandon. A ce stade nous pouvons donner toutes sortes de variables pouvant intervenir lors des comparatifs. Ces variables sont la puissance, la consommation moyenne, le prix, l'équipement de série (version Professional), ainsi que d'autres possibilités de comparaisons (version User Guide). Il s'agira des options disponibles et de leur prix et dans une version ultérieure du logiciel, nous donnerons des tableaux reprenant toutes sortes de mesures regroupées du point de vue consommation à différentes vitesses, distances de freinage, accélérations, reprises, liste des concurrentes potentielles d'un véhicule et autres données. Ces informations, pour la plupart, ne sont pas encore reprises dans la base de données et nous n'en avons même pas tenu compte du point de vue conceptuel, afin de limiter le cadre de ce travail. Les deux manières de choisir ne sont pas forcément indispensables, c'est-à-dire que l'utilisateur peut très bien ne pas vouloir opérer un choix objectif. Dans ce cas, uniquement la fonction Choix_Subj interviendra. C'est pour cela que nous avons un lien OU entre ces fonctions.

Il faut remarquer qu'étant donné que nous ne connaissons pas exactement la manière de choisir un véhicule (quitte à se répéter, il est préfèrable d'insister sur ce point très important), on peut aussi considérer que Choix_Obj constitue également à elle toute seule une manière afin d'arriver à un véhicule. C'est le cas par exemple lorsque l'utilisateur sait déjà quelle voiture il risque d'acheter, mais hésite pour un motif quelconque (par exemple s'il veut la plus équipée de série). Il désirera donc uniquement comparer ces véhicules et le choix définitif s'en suivra. Il n'était donc pas utile de refaire un graphe d'enchaînement.

Abandon signifie que l'utilisateur n'a pas trouvé de véhicule à son goût (c'est rare).

Les enchaînements concernant le choix à partir d'une catégorie de véhicules sont similaires à ceux démarrant d'un modèle. L'unique différence est que la liste de versions obtenue est composée de versions se référant à un ou plusieurs modèles et appartenant à la catégorie souhaitée.

L'avant-dernier enchaînement de fonctions est le plus intéressant et le plus difficile à implémenter. Il s'agit de faire un choix à partir de critères objectifs et non objectifs. Les critères objectifs priment sur les autres. En fait lorsque nous parlons de critères non objectifs, nous parlons de critères moins importants et qui se réfèrent entre autre à l'équipement dont doit disposer le véhicule qu'on désire.

Ch_Lst_Globale est une fonction qui tient compte des critères en input et essaye de répondre en donnant une liste de véhicules (Lst_Véhic) qui correspond le plus possible aux critères donnés. Ceci est très difficile à implémenter, car on veut essayer d'équiper un véhicule qu'on ne connaît pas encore (ou on connaît uniquement le modèle ou la catégorie) avec certaines options qui sont de type confort ou sécurité. Outre à répondre aux critères impératifs, il faudra se débrouiller pour équiper au mieux un véhicule avec les options demandées et si cela n'est pas possible, la fonction renvoie un message (Msg_Err) afin de prévenir l'utilisateur.

4. Dérivation des attributs de dialogue

4.1 *Table des paramètres relatifs à la tâche interactive*

Lors des choix qui sont faits, nous prendrons les valeurs suivantes pour les paramètres:

prérequis : minimaux à modérés (connaissance du vocabulaire du monde automobile).
productivité : faible à moyenne (comme nous envisageons ici la version Professional, il est un fait qu'il s'agit d'un outil de travail. Le vendeur sera amené à l'utiliser peut-être plusieurs fois par jour.
environnement objectif de la tâche : inexistant.
reproductibilité de l'environnement : non praticable.
structuration de la tâche : faible (on ne sait même pas dire avec exactitude ce que signifie choisir un véhicule).
importance de la tâche : modérée à élevée. (La satisfaction du client est d'autant plus grande que le véhicule choisi correspond mieux à ses exigences).
complexité de la tâche : modéré à élevé.

De ces valeurs, on peut dériver les styles d'interaction, tout en estimant que certains paramètres sont moins importants que d'autres. Ceux qui nous concernent le moins sont la productivité, l'environnement objectif et la reproductivité de l'environnement.
Nous pouvons donc utiliser l'interaction iconique, la manipulation directe, le multi-fenêtrage, le remplissage de formes sous réserve, la sélection de menus et les touches de fonction. Tout dépend bien entendu du cadre dans lequel le vendeur effectue le travail (si par exemple il désire avoir un rappel concernant le prix d'une option pour information personnelle, ce n'est pas aussi important que de choisir une option pour un client. L'importance de la tâche et la complexité seront donc faible. Or, nous avons considéré modéré à élevé. Donc le cas faible doit être inclus aussi.).

4.2 *Table des paramètres relatifs à un stéréotype d'utilisateur*

expérience de la tâche : riche, cependant le logiciel est utilisé par le vendeur avec le client; il faut donc estimer que l'expérience du client est élémentaire. De plus, le logiciel peut être utilisé comme borne d'information dans un show-room par un client.
expérience du système : élémentaire (Nous estimons que le vendeur n'a jamais utilisé d'autre moyen de vente que ceux habituels).
motivation : élevée (Le vendeur bénéficie d'un pourcentage par voiture vendue).
expérience d'un moyen d'interaction complexe : élémentaire (mais cela dépend d'un vendeur à l'autre. On pourrait avoir un vendeur branché par l'informatique et serait un manieur de souris hors pair après quoi son expérience d'un moyen d'interaction serait élevé).

Nous pouvons donc garder les mêmes styles d'interaction que ci-dessus.

4.3 *Table des paramètres relatifs au poste de travail de l'utilisateur final*

type de traitement : multi-traitement. Le vendeur peut essayer de vendre, mais également fournir un renseignement téléphonique ou donner un dépliant à un client.

capacité de traitement : disons moyenne à élévée, car pendant que le vendeur choisit une voiture, il peut également répondre à une question concernant autre chose et dont la réponse est fournie par le logiciel (ex: on s'intéresse à une voiture essence, mais on voudrait également savoir le prix de la même version en diesel et s'apercevoir que c'est plus intéressant).

Nous nous apercevons que les mêmes styles d'interaction reviennent.

4.4 *Table des paramètres relatifs aux attributs de dialogue*

contrôle du dialogue : mixte.
mode de dialogue : asynchrone.
mode de déclenchement d'une fonction : manuel explicite affiché.
conversation : mini-monde.

Les mêmes styles d'interaction, sauf le remplissage de formes sont utilisables.

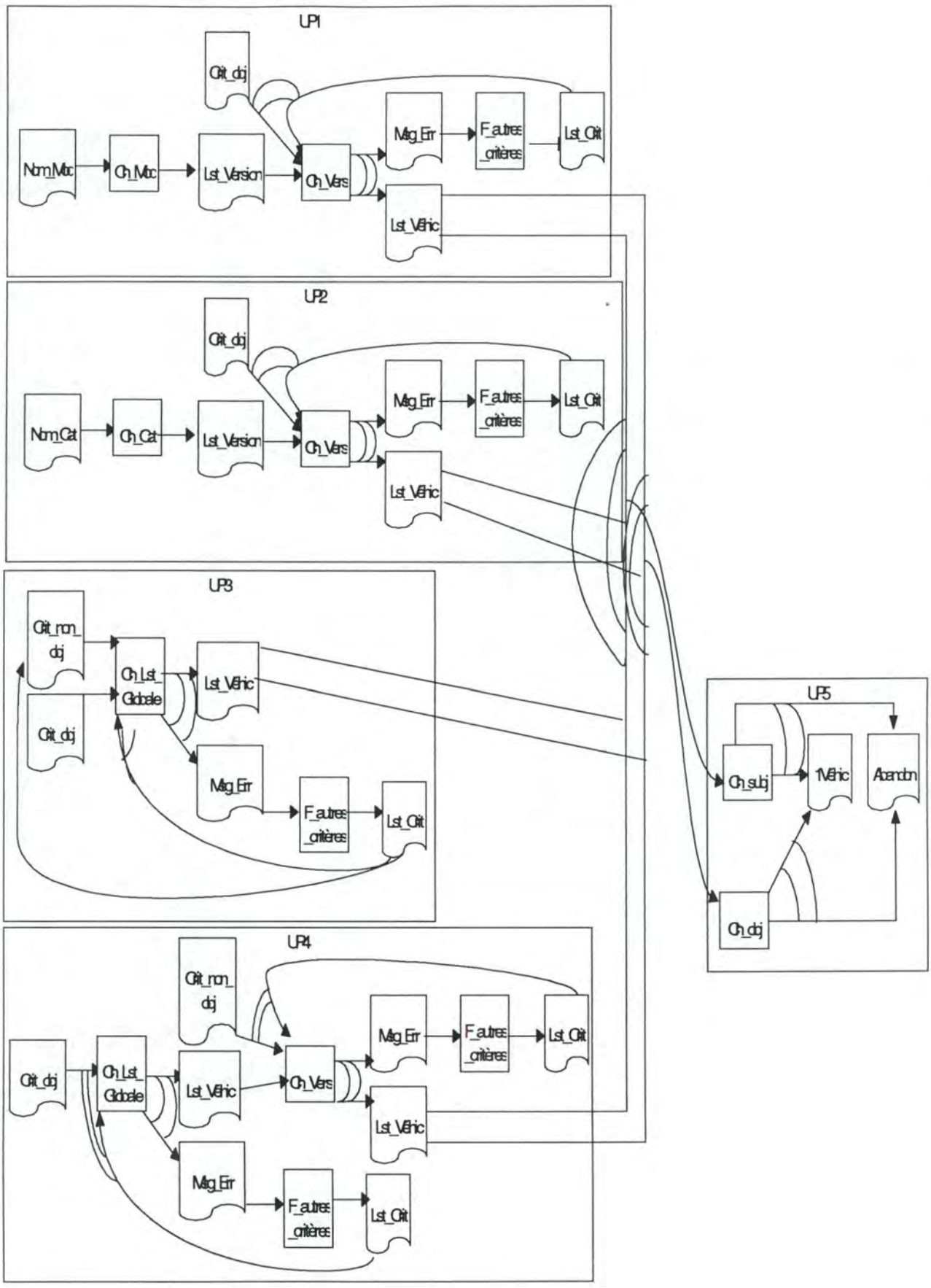
Eu égard aux paramètres identifiés, on constate en parcourant les tableaux de correspondance du document "Dérivation des styles d'interaction", que nous pouvons utiliser:

1. l'interaction iconique
2. la manipulation directe
3. le multi-fenêtrage
4. la sélection de menus
5. les touches de fonction.

Dimension 2: Pilotage actif de la présentation à partir de règles ergonomiques

2.1 *Définition de la présentation*

2.1.1 *Identification des unités de présentation (UP)*


UP1
Crit_obj
Nom_Mdc
Ch_Mdc
Lst_Version
Ch_Vers
Msg_Err
F_autres_critères
Lst_Crit
Lst_Véhic
UP2
Crit_obj
Nom_Cat
Ch_Cat
Lst_Version
Ch_Vers
Msg_Err
F_autres_critères
Lst_Crit
Lst_Véhic
UP3
Crit_non_obj
Crit_obj
Ch_Lst_Gdvale
Lst_Véhic
Msg_Err
F_autres_critères
Lst_Crit
UP4
Crit_non_obj
Crit_obj
Ch_Lst_Gdvale
Lst_Véhic
Ch_Vers
Msg_Err
F_autres_critères
Lst_Crit
Lst_Véhic
Msg_Err
F_autres_critères
Lst_Crit
UP5
Ch_subj
1Véhic
Abandon
Ch_obj

Pour l'identification, nous retiendrons une UP par sous-tâche (la tâche principale étant de choisir un véhicu une sous-tâche correspond à un ensemble de fonctionnalités menant à la sous-tâche "choix d'une version".
UP1: Choix à partir des modèles
UP2: Choix à partir des catégories
UP3: Choix des versions dans la liste des versions, à partir de critères
UP4: Choix des versions correpondant aux critères
UP5: Choix objectif et subjectif d'une version

2.1.2 *Identification des fenêtres*

A partir du GE dessiné et décomposé en UP, on fait correspondre une partition du sous-graphe de chaque UP en sous-sous-graphes, qui matérialisent les fenêtres.
Il est donc important de souligner que le GE lui-même induit déjà d'une certaine manière l'identification des fenêtres.

Critères d'identification des fenêtres

Il existe différents critères d'identification de fenêtres, chacun déterminant une décomposition plus ou moins fine de l'UP. Plus la décomposition est fine -et donc le nombre de fenêtres par UP est élevé-, plus c'est généralement adéquat pour un utilisateur novice, car cela structure de plus en plus finement l'interface. En revanche, un utilisateur expert ne tient pas à être encombré par de multiples fenêtres. Le niveau de décomposition d'une UP en fenêtres doit traduire le niveau d'expérience d'un utilisateur. Les différentes identifications sont:

- *l'identification minimale* : une information par fenêtre. A chaque information externe peut ainsi correspondre une fenêtre. Cette solution offre généralement le guidage maximal.

- *l'identification maximale* : une fenêtre par UP, soit toutes les informations externes une et une seule fois dans une seule et même fenêtre. A réserver aux utilisateurs experts.

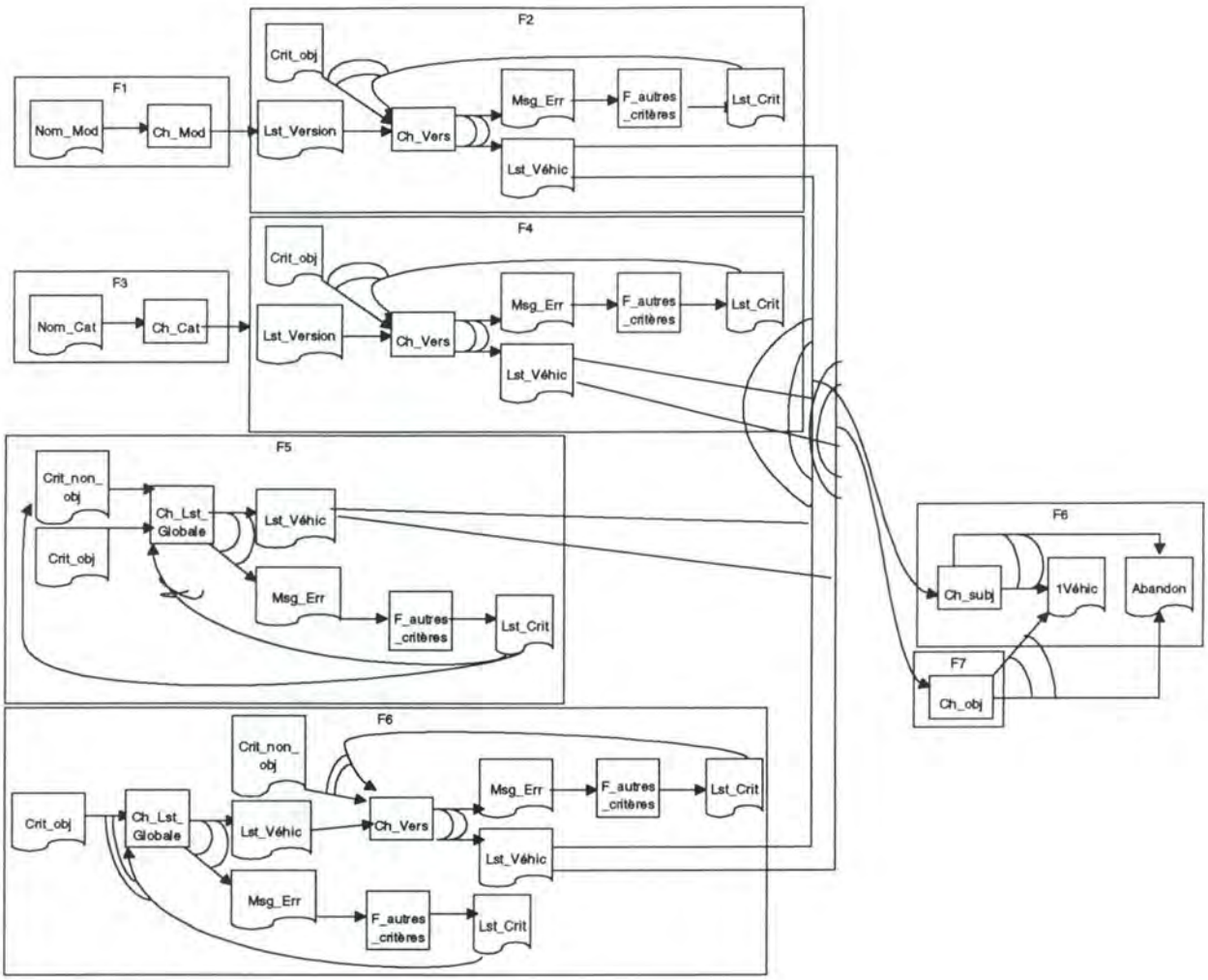
- *l'identification entrée/sortie* : toutes les informations externes en entrée sont groupées dans une fenêtre, et toutes les informations externes en sortie dans une autre fenêtre.

- *l'identification fonctionnelle* : toutes les informations en E/S sont regroupées dans une seule et même fenêtre. En d'autres termes, il y a une fenêtre regroupant toutes les informations externes de chaque fonction. Cela correspond à un niveau d'utilisateur moyen.

- *l'identification libre* : on définit chaque fenêtre comme on veut. Ce n'est donc pas une identification véritablement guidée par la sémantique ou par les règles ergonomiques.

- *l'identification groupée* : à partire des identifications précédentes, on peut décider de regrouper plusieurs fenêtres ensemble pour des raisons d'ergonomie, pour autant que la charge de travail n'excède pas les limites cognitives de l'utilisateur.

Un autre facteur important à prendre en compte est le fait que pour un style d'interaction donné, une ou plusieurs identifications déterminées peuvent convenir.



Justification de l'identification des fenêtres

F1 - on retient l'identification entrées/sorties

F2 - on retient l'identification maximale

F3 - on retient l'identification entrées/sorties

F4 - on retient l'identification maximale

F5 - on retient l'identification maximale

F6 - on retient l'identification maximale

F7 - on retient l'identification entrées/sortie

F8 - non implémenté

2.1.3 *Sélection des Objets Interactifs Abstraits (OIA)*

Les fenêtres qui viennent d'être définies constituent des fenêtres logiques. Du style d'interaction retenu pour ces fenêtres, nous pouvons en déduire que les OIA seront les suivants (se référer aux annexes A afin de voir les écrans):

F1 : annexe A - page 2
- un panneau contenant le titre de la fenêtre faisant référence à la tâche en cours,
- un panneau contenant des boutons de commande diverses (help, retour écran précédent),
- des boutons de commande iconiques (symbolisant une flèche à droite et/ou gauche afin de mimétiser le défilement de l'écran, dans le cas où il y aurait plus de modèles que l'écran ne peut contenir,
- menu plein écran mixte, en ce sens que nous pouvons choisir le modèle en voyant une photo de ce modèle, ainsi que son nom et en cliquant dessus (sur la photo).

F2 : annexe A - page 3
- un panneau contenant le titre de la fenêtre faisant référence à la tâche en cours,
- une list box textuelle afin de visualiser les différentes versions du modèle choisi,
- deux flèches de défilement pour indiquer la direction dans laquelle on veut se déplacer dans la liste (à chaque clic sur une flèche, un nouvel élément, le successif ou leprimitif, est sélectionné),
- une barre de défilement,
- un panneau indiquant le prix de base de la version sélectionnée dans la liste
- un panneau contenant:
  des boutons de commandes diverses (qu'on retrouve dans chaque fenêtre: accès à un autre écran, c'est-à-dire retour à l'écran précédent, au menu général, bouton d'aide, et autre fonctionnalité),
- un panneau constitué de:
  boîtes à cocher pour les critères objectifs (essence - diesel - 2 portes - 3 portes - 4 portes - 5 portes - station wagon - boîte automatique - boîte manuelle), un potentiomètre permettant d'encoder le prix, un bouton de commande pour accepter le prix encodé (le rendre effectif), un bouton de commande pour annuler tous les critères et recommencer.

F3 : idem F1 - annexe A - page 10

F4 : idem F2 - annexe A - page 3

F5 : idem F2 - annexe A - page 3

F6 : annexe A - page 13
- un panneau contenant le titre de la fenêtre faisant référence à la tâche en cours,
- un panneau (critères) contenant quatre boutons de commandes: les trois premiers portant les noms des critères (prix, puissance et consommation) et le quatrième portant le nom "Modification", permettant de modifier l'ordre des critères.
- un panneau (équipement) contenant deux boutons de commandes: un pour l'équipement au niveau confort et l'autre pour l'équipement au niveau sécurité
- un potentiomètre permettant d'encoder le prix,
- un potentiomètre permettant d'encoder la puissance,
- un potentiomètre permettant d'encoder la consommation,
- un bouton de commande permettant de lancer la recherche lorsque les valeurs des potentiomètres ont été fixées,

- une list box textuelle afin de visualiser les différentes versions sélectionnées lors de la recherche,
- un panneau indiquant le prix de base de la version sélectionnée dans la liste
- un panneau indiquant le prix équipemnt compris de la version sélectionnée dans la liste
- un bouton de commande permettant d'obtenir un détail des résultats
- un panneau contenant:
  des boutons de commandes diverses (qu'on retrouve dans chaque fenêtre: accès à un autre écran, c'est-à-dire retour à l'écran précédent, au menu général, bouton d'aide, et autre fonctionnalité),

F7 : annexe A - page 8
selon que la décision de comparer des versions se fait au départ d'une fenêtre ou un choix de départ (dans un menu général qui correspond aux différentes tâches, voir plus loin), nous aurons:
un panneau contenant:
- deux list box textuelles (une de celles-ci est celle de F1),
- quatre flèches de défilement, deux pour chaque liste pour indiquer la direction dans laquelle on veut se déplacer dans la liste (à chaque clic sur une flèche, un nouvel élément, le successif ou le primitif, est sélectionné),
- deux flèches iconiques (symbolisées par des doigts indiquant les directions gauche et droite) afin de sélectionner les versions qu'on veut comparer,
- deux boutons de commande pour comparer et annuler,
- un bouton de commande d'aide

ou annexe A - page 9
un panneau constitué de:
- une list box textuelle,
- deux flèches de défilement,
- une flèche iconique (symbolisée par un doigt indiquant la direction gauche) afin de sélectionner les versions qu'on veut comparer,
- deux boutons de commande pour comparer et annuler,

un panneau constitué de:
- un bouton de commande d'aide,
- des boutons de commande diverses (retour écran précédent, affichage de la liste de versions)

F8 : non implémentée

Afin que le logiciel reflète réellement une manière de décider, il faut encore que toutes les tâches soient activables. Pour cela, nous avons une fenêtre supplémentaire qui donne la possibilité à l'utilisateur de choisir réellement de quelle manière il va s'intéresser à un véhicule. Cette fenêtre sera constituée de:

- un panneau contenant le titre de la fenêtre faisant référence à la tâche en cours, dans ce cas le nom du logiciel et le nom du constructeur,
- menu plein écran textuel,permettant de choisir une des tâches,
- un panneau constitué de:
  boutons de commandes relatifs au choix des langues, informations (about box),

aide et un bouton de commande afin de quitter l'application.

2.1.4 *Transformation des OIA en Objets Interactifs Concrets (OIC)*

Cette section n'est pas utile, parce que tous les OIA que nous avons considéré dans la section précédente sont directement utilisable (ils existent tels quels) dans le langage de programmation utilisé (Visual Basic 3.0®). Nous obtenons donc les fenêtres que l'on peut consulter dans les annexes.

En ce qui concerne les pricipales manières de choisir un véhicule, nous avons obtrnu les fenêtres de l'application. Il reste à présent à terminer le raffinement que nous avons commencé, à savoir partir de général (menu général) et arriver à une seule version, sur base de laquelle il est possible de discuter avec le vendeur afin de concorder le prix et les accessoires disponibles sur le véhicule. Nous obtenons donc la fenêtre finale de l'application (consulter les annexes).

4.3 SENSITIVE CAR USER GUIDE

Dans cette brève section, nous nous proposons de passer à la version User Guide. Nous avons déjà annoncé à plusieurs reprises que les objectifs de cette version étaient à la fois les mêmes que ceux de la version Professional, mais différents également puisque le public visé, ainsi que les moyens mis en oeuvre sont différents. Reprenons succintement ces objectifs:

- Logiciel plurimarques
- Logiciel visant un public de non-professionnels de l'automobile
- But de répondre à n'importe quelle question relative à un véhicule (Module Sensitive Query)
- Classement avec ex aequo des véhicules choisis à partir de critères (Module Classement)

Pour le premier point, nous n'avons pas à nous préoccuper, car il s'agit d'exploiter les acquis de la version Professional en ce qui concerne l'interface. Le fait d'ajouter un "niveau" dans la possibilité de choisir un véhicule ne nous empêche pas de garder ce qui a déjà été développé auparavant.
Les points qui nous intéressent le plus et qui représentent réellement une nouveauté de La version User Guide sont les deux derniers et font partie des deux prochains chapitres.

4.4 EVALUATION CRITIQUE DE LA METHODE UTILISEE

Comme nous l'avons déjà dit au début de ce chapitre, un système d'aide à la décision est un logiciel devant fournir un certain nombre d'outils dans le but d'aider l'utilisateur à accomplir une tâche. Celle-ci, dans le domaine de la vente automobile.
Nous avons essayé de respecter la méthodologie proposée, afin de dériver l'interface devant supporter les fonctionnalités. Ceci n'a pas toujours été possible, car, comme nous l'avons déjà précisé auparavant, la méthodologie s'applique très bien pour des tâches structurées. Afin de pouvoir l'utiliser, nous avons dû l'adapter en simulant une fonctionnalité supplémentaire, à savoir, choix_critères_non_objectf. Nous avons dit qu'elle n'était pas implémentable réellement et que nous considérions qu'elle se limitait à offrir un certain nombre d'informations à l'utlisateur. Celui-ci choisirait son véhicule après en avoir pris connaissance.

Les avantages de la méthode sont par exemple un ensemble de règles à appliquer presque aveuglément afin d'obtenir une interface de qualité (nous entedons par là qui respecte les règles énoncées au début du chapitre). Ceci étant dit, le développement d'une interface étant soumis à des règles cognitives, ce qui est communément appelé *le bon sens* peut parfois suffire. En effet, le but de l'interface est de fournir le lien entre les fonctionnalités et l'utilisateur, c'est-à-dire que ce dernier accède à des services via l'interface. Il est donc souhaitable de proposer les choses clairement. C'était un objectif majeur au début du chapitre et il a été atteint: le logiciel a été mis dans les mains de non-informaticiens et non-utilisateurs de la micro-informatique afin de tester leur réaction. Le résultat a été convainquant, le temps d'apprentissage ayant été de quelques minutes seulement. Leur problème majeur ayant été l'utilisation de la souris, mais au bout de quelques heures, le problème ne se posait plus.

5. LE MODULE SENSITIVE QUERY

5.1 INTRODUCTION

Afin de répondre entièrement à l'objectif que nous nous sommes posés en envisageant de développer la version User Guide, il faut admettre que l'utilisateur peut vouloir questionner la base de données d'une manière que nous n'avons pas prévue. En effet, via l'interface développée, nous accédons aux données selon des voies prédéfinies et statiques, qui semblaient évidentes.
En effet, il semble logique de demander les modèles d'une certaine nationalité roulant à l'essence et coûtant environ 700.000 francs. On a également envisagé d'équiper les véhicules selon les désirs de l'utilisateur et dans les limites du faisable, le logiciel essaie de satisfaire l'utilisateur.

Cependant, il fallait aller plus loin, car bien qu'on réponde à 85% des questions qu'on peut se poser relatives à des voitures, il faut admettre qu'il y a peut-être des utilisateurs qui veulent aller encore plus loin. Il y avait deux solutions pour pallier à cet inconvénient.

La première était de fournir une liste de mots-clés et chaque fois que l'utilisateur sélectionnait un de ceux-ci, il obtenait une série de renseignements connectés à ce mot-clé. Par exemple, si je choisis Airbag, je peux en donner une description, la liste des voitures sur lequel il est disponible de série, ainsi que celles où il est disponible en option, ou encore la liste des véhicules sur lesquels il n'est pas disponibles du tout, toutes ces listes étant triées par marques ou par nations ou par continent. On peut continuer longtemps. Cette manière de faire est encore trop statique et nous ramène de nouveau au problème de devoir prévoir les questions que pourraient poser les utilisateurs, donc nous ne résolvons pas réellement le problème. C'est la solution qui a été adoptée par certains logiciels sur le marché.

La deuxième solution, celle qui a été adoptée, est le *Sensitive Query*. Il s'agit de donner la possibilité à l'utilisateur de questionner la base de données comme il l'entend, selon ses schémas mentaux et non selon les miens. En effet, considérons l'exemple suivant:
"Existe-t-il un véhicule (donner la marque, modèle et version) tel que la cylindrée n'excède pas 1800 cc, la puissance inférieure à 100 CV, ayant le toit ouvrant en option pour un prix inférieur à 15.000 francs et 3 portes, ou si ce prix est supérieur à 15.000 francs, on acceptera une cylindrée de 2.300 cc ou un Break, la liste éventuelle triée sur la marque?"

On pouvait difficilement penser à cette question, mais cela reste du domaine du possible. Pourquoi pas. Nous avons assez répété que nous ne pouvions pas donner une définition rigoureurse de ce que signifiait choisir un véhicule, alors il faut admettre également que les questions que se posent certaines personnes soient assez inattendues.

Quoiqu'il en soit, la solution envisagée est celle de donner la possibilité à l'utilisateur de poser des questions à la base de données, en simulant des requêtes SQL. Il fallait toutefois, et c'est là que réside toute la difficulté, faire de telle sorte que l'utilisateur ne soit pas amené à apprendre un langage, mais plutôt le guider via une interface conviviale et simple d'utilisation. Il ne fallait pas non plus créer un SGBD comme Microsoft Acess®, même en plus réduit, car cela avait plusieurs inconvénients majeurs: cela supposait que l'utilisateur avait une connaissance de la structure de la base de données utilisée pour *Sensitive Car*, mais pouvait également attacher une sémantique aux données contenues dans les tables, c'est-à-dire connaître le modèle

conceptuel des données que nous avons développé. A partir de cela, il devait également connaître le langage SQL afin de créer ses propores requêtes et interroger la base de données. Cela est pratiquement impossible pour un non-informaticien. Il faudrait qu'il fasse d'abord de la rétro-ingéniérie des bases de données afin de comprendre la sémantique de la base de données et ensuite étudier le langage SQL.

Pour reprendre l'exemple que nous avons donné, on obtiendrait la requête suivante:
SELECT Nom_Marque, Nom_Modele, Nom_Version
FROM Marque, Modele, Version, Option, ElementOpt
WHERE Modele.ID_Marque = Marque.ID_Marque AND Version.ID_Modele = Modele.ID_Modele AND (Version.Cylindrée ≤ 1800 AND Version.Puissance ≤ 100 AND ((ElementOpt.Nom_Element = "Toit ouvrant" AND Option.Typ_Option = "O" AND Option.Prix ≤ 15000 AND Version.Nb_Portes = 3) OR ((Option.Prix ≥ 15000 AND Version.Cylindrée = 2300) OR Version.Break = "O")) AND Option.ID_Element = ElementOpt.ID_Element
ORDER BY Nom_Marque

C'est peut-être cette requête que l'utilisateur voulait créer (ou une autre en fonction de l'ordre des parenthèses!). Peut-on demander à un non-informaticien de créer une telle requête? Cela est bien trop difficile. C'est pourquoi nous avons développé une autre méthode.

5.2 SIMULATION DE REQUETES SQL

Pour remédier à ces difficultés, il a fallu faire de telle sorte que l'utilisateur n'ait pas à se soucier de la provenance des champs qu'il sélectionnait via l'instruction SELECT. Rappelons brièvement la syntaxe de sélection des données de SQL:

SELECT [*prédicat*]{[*table.*]*champ1*[,[*table.*]*champ2*[,...]]]}
FROM *liste_de_tables*
WHERE *critère*
GROUP BY *liste_de_champs_groupe*
HAVING *condition*
ORDER BY *champ1* [ASC|DESC] [, *champ2*[ASC|DESC] [,...]]]

prédicat = ALL, DISTINCT, DISTINCTROW, TOP
liste_de_tables = la liste des tables d'où proviennent les champs sélectionnés
critère = une expression à laquelle doivent répondre les champs afin qu'ils soient valides
liste_de_champs_groupe = les champs (10 maximum) sur lesquels on veut grouper les records répondant au critère
condition = expression qui détermine quels records groupés seront affichés

A partir de ces quelques lignes, nous devons essayer de développer une interface donnant la possibilité à l'utilisateur de:

- choisir les champs qui lui intéressent (clause SELECT),
- donner les tables d'appartenance des champs (clause FROM),

- donner le critère auquel doivent répondre ces champs (clause WHERE),
- pouvoir grouper les données (clause GROUP BY),
- donner la condition à laquelle doivent répondre les records affichés (clause HAVING),
- pouvoir trier les résultats (clause ORDER BY).
- pouvoir utiliser certaines fonctions statistiques (MIN, MAX, AVERAGE, COUNT)

Or, étant donné que l'utilisateur n'a pas connaissance de la structure de la base de données, et même s'il l'avait, il ne comprend pas forcément la signification de tous les champs et de leur contenu, il fallait faire de telle sorte qu'il choisisse les champs sans dire d'où ils proviennent. On élimine ainsi la clause FROM, car si il peut encore être évident de sélectionner des données qui proviennent de Modèle ou de Version, cela devient moins évident de choisir des champs provenant de tables représentant une association dans le modèle E-A. La clause SELECT doit être également simulée, car il faudra que l'utilisateur choisisse un "mot-clé" et non le nom physique d'un champ d'une table. Il faudra également que l'utilisateur ait un moyen de construire une expression syntaxiquement exacte et qu'il ne fasse pas d'erreur de type (en effet, ce ne serait pas agréable pour un non-informaticien de se retrouver devant un message tel que *"type checking error in query expression"*). Il faudra donc guider l'utilisateur et ne lui permettre que ce qui est possible, en évitant de lui donner la possibilité de faire des erreurs. Les trois dernières clauses semblent moins difficiles à implémenter. Nous verrons par la suite ce qu'il en sera.

Afin de rendre cela possible, il a fallu créer une base de données de la base de données (son catalogue). Cette base de données reprendra:

pour chaque champ non identifiant, ni champ de référence de chaque table ne représentant pas une association many to many:

- son nom physique -identifiant, car nous avons pris la précaution de donner un nom distinct à chaque attribut de la base de données,
- un alias français étant un nom parlant (mot-clé),
- une description ou signification (ex: Airbag: coussin se gonflant lors d'un impact frontal du véhicule...),
- un ensemble de synonymes éventuel: un champ peut avoir plusieurs dénominations communément acceptées,
- une table d'appartenance: la table à laquelle il appartient (afin d'éliminer la clause FROM),
- un type: le type de la donnée qu'il contient (numérique, string, photo, ce dernier étant considéré un type particulier)
- une valeur,
- un intervalle de valeurs éventuel: cela aidera afin de proposer une valeur pour un type numérique,
- une référence éventuelle à d'autres champs.

pour chaque champ:

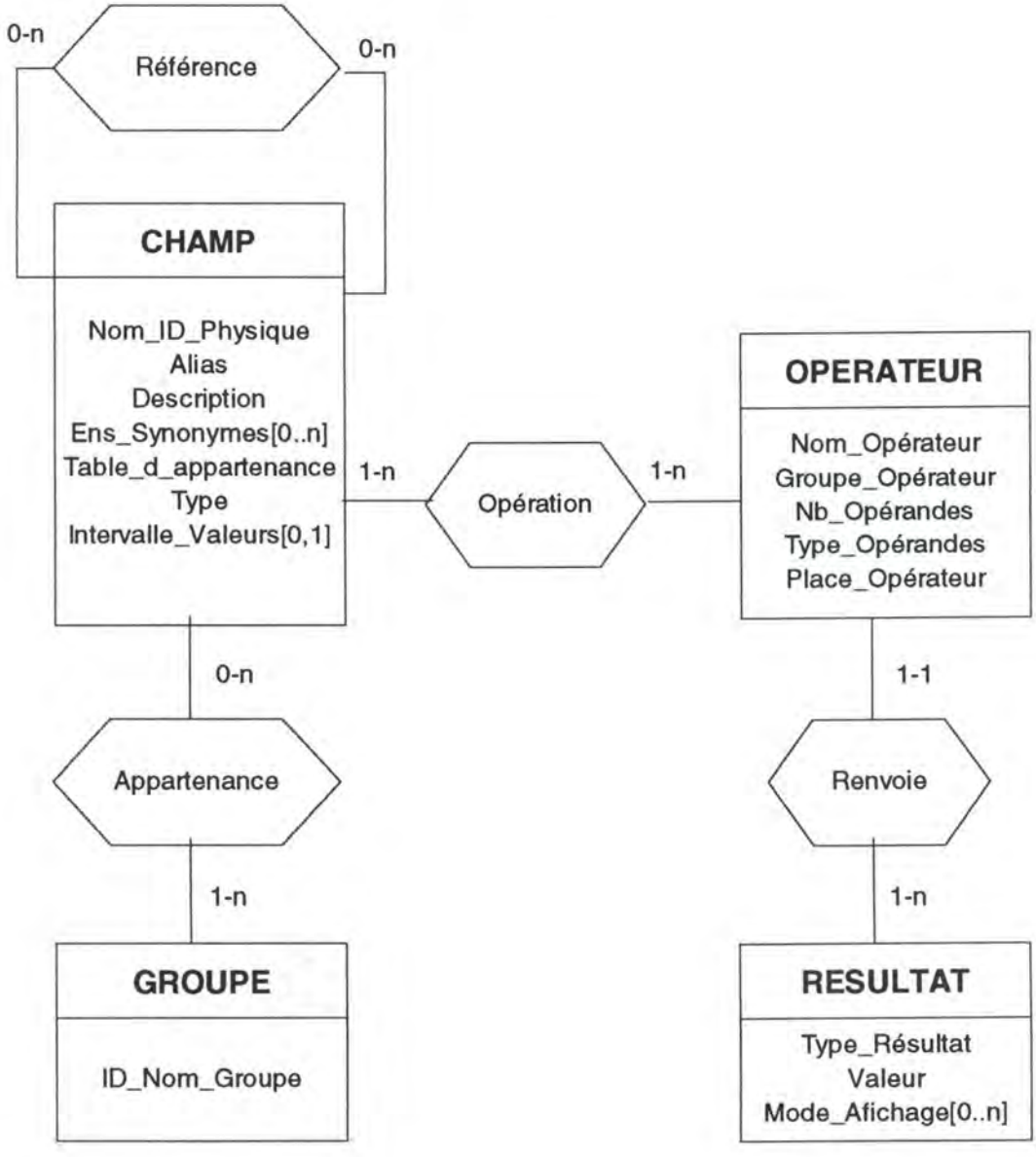
- on connaîtra le ou les groupes éventuels auxquels il appartient : ex : le groupe performances (champs 0-100, 1000DA,...), le groupe consommation (champs cons_90, cons_ville,...).

En outre, sur chaque champ on pourra appliquer un certain nombre d'opérateurs.
Pour chaque opérateur:

- un nom: +, *, et, ou,...,
- un groupe: arithmétique, comparaison, logique,
- le nombre d'opérandes,
- le type des opérandes,
- la place de l'opérateur (préfixé, infixé, postfixé): en effet des opérateurs tels que Min ou Max sont préfixés alors que + ou * sont infixés.

L'application d'un opérateur renvoie un certain résultat.
Pour chaque résultat:
- un type: booléen, entier, réel, string, photo.
- une valeur,
- un ou plusieurs modes d'affichage: si c'est une photo, on peut l'afficher et éventuellement l'agrandir. Si c'est un string, on pourra peut-être en faire autre chose que l'afficher.

Construisons donc le schéma E-A de cette base de données:

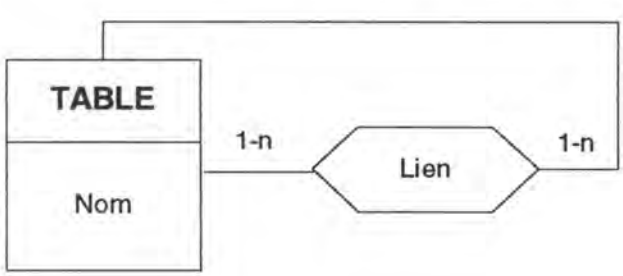


Ceci nous permettra de construire des requêtes assez facilement, mais uniquement des requêtes simples ne portant que sur une seule table. En effet, imaginons que l'utilisateur veuille avoir la liste de tous les modèles, ainsi que leur nationalité. Nous avons le problème de devoir naviguer à travers la base de données, en donnant des conditions sur des champs de plusieurs tables, afin de construire la requête exacte. Ici nous aurions:

```
SELECT nom_modèle, nom_pays
FROM Modele, Nation, Marque
WHERE Modele.Id_Marque = Marque.Id_Marque
        and Marque.Id_Pays = Nation.Id_Pays
```

Il faut pouvoir construire le lien entre les tables, ces liens étant les champs externes (foreign keys - UNION JOIN). Comme nous ne savons pas a priori quelle question va poser l'utilisateur, nous devrions avoir, d'un point de vue des graphes [FICHEFET92], un chemin ayant comme origine une table et comme extrémité n'importe quelle table. Dans notre modèle E-A de *Sensitive Car*, l'entité Version se trouve sur plusieurs chemins. Mais alors se pose le problème du parcours de ces chemins lorsque nous devrions naviguer au travers des tables. En effet, il va falloir considérer des aspects d'optimisation, car certains chemins sont plus courts que d'autres afin d'arriver au résultat escompté. Le temps de parcours au sens des graphes est ramené au temps de construction de la requête complète lorsque plusieurs tables sont concernées. Afin d'éliminer cette difficulté, nous avons deux solutions: la première consiste à construire ce que nous considérons être la **base de données complète**, en référence à un graphe complet.

En fait, à partir de n'importe quelle entité, il sera possible d'atteindre une autre entité, en ajoutant une association entre toutes les entités du schéma, deux à deux. Ce moyen, coûteux en espace il est vrai, nous garantit d'avoir accès à toutes les informations possibles et imaginables de la base de données. On obtient un pseudo-schéma E-A qui relève même des associations qui peuvent ne pas avoir de sens. C'est le prix à payer afin de couvrir le maximum de requêtes, mais n'oublions pas que cela suit tout de même une certaine logique, celle de prévoir le maximum de schémas mentaux des utilisateurs, et pas seulement le mien, reflété - lui - dans le schéma E-A que nous avons construit au chapitre 3.1 dans l'analyse conceptuelle qui était la mienne. Ici j'essaie d'être le plus général possible, malgré une perte de sémantique du schéma E-A que nous ne construirons pas, car il serait énorme, ni la base de données dérivée dudit schéma. La base de données complète est construite selon le shéma suivant:



La seconde solution consiste à relever tous les chemins qui existent dans la base de données. Pour ce faire, il faut passer par les tables ayant une *foreign key* en commun. Par exemple, pour aller de Nation à Modèles (au cas où on voudrait connaître tous les modèles et leur nationalité), il faut une requête comme

"SELECT Nom_Modèle
FROM Modèle, Nation, **Marque**
WHERE Marque.Id_Marque = Modèle.Id_Modèle and Nation.Id_Pays = Marque.Id_Pays"

Nous voyons qu'il a fallu ajouter la table Marque, puisqu'elle sert de "lien" entre Nation et Modèle. Si nous voulons un équivalent en termes de graphes, nous disons que le chemin ayant pour origine le sommet Nation et pour extrémité le sommet Version passe par les tronçons Nation-Marque et Marque-Version (on remarquera que cela a un sens, puisque l'extrémité d'un tronçon est bien l'origine du tronçon suivant (dans notre exemple: Marque). C'est un graphe non-orienté puisque le chemin Version-Nation est le même que le chemin Nation-Version (en ce sens que les mêmes tronçons sont empruntés et donc les mêmes conditions sont retrouvées dans la clause WHERE).
La modélisation de ces conceptes est la suivante:

Cette solution, bien que plus compliquée à implémenter que la première solution, nous permet de rester avec la base de données que nous avons développé au chapitre 3.3.3 (optimisation de la BD). C'est la solution que nous allons implémenter.

Avec ce stratagème, nous pouvons donc commencer à penser à comment construire une requête, tout en restant le plus général possible. Nous avons déjà éliminé le problème des clauses SELECT et FROM. Le problème majeur reste bien entendu la clause WHERE puisqu'il faut donner le moyen à l'utilisateur non informaticien d'énoncer une condition plus ou moins complexe auxquels doivent répondre les champs sélectionnés. Il serait donc utile ici d'analyser la condition d'un peu plus près et voir comment fournir des outils adéquats de construction.

Nous avons déjà vu que la condition ou critère était une expression. Pour nous, une expression sera une combinaison d'opérateurs, constantes, litéraux, valeurs et noms de champs (les alias pour l'utilisateur) qui donne une valeur simple. Les opérateurs sont soit arithmétiques, auquel cas ils opèrent sur des opérandes numériques, soit des opérateurs de comparaison, opérant sur des numériques ou chaînes de caractères, soit des opérateurs logiques servant de lien entre plusieurs expressions, dans le but de construire une expression plus complexe. A propos de ces derniers, nous ne retiendrons que les opérateurs ET, OU et XOU; le NON n'étant pas vraiment nécéssaire.

D'un point de vue plus formel, une expression sera vue de la manière suivante, selon la notation BNF (nous donnons également un exemple afin de clarifier la notation):

5.3 NOTATION BNF

Conventions de notation :

[xxx] signifie que ce qui est entre crochets est optionnel
{ xxx } signifie une suite de 1 à n fois ce qui est entre accolades
| signifie une alternative
--> xxx signifie renvoie la valeur de type xxx (donné à titre tout à fait indicatif)
constante, littéral, valeur sont pris dans leur sens usuel (on ne les spécifie pas plus)
[{ xxx }] signifie une suite de xxx optionnelle

<expression> =
(*<expression_simple>*)
| (*<expression_composée>*)
| [(...] (*<expression>*) [<opérateur_logique> (*<expression>*)...)]
Cette dernière étant par exemple: (((a<b) ET (c>d)) OU (e = f)). D'après notre convention, nous enfermons la première expression simple dans des parenthèses, chaque fois que nous ajoutons un opérateur_logique et une expression entre parenthèses.

<expression_composée> =
(*<expression_simple>* *<opérateur_logique>* *<expression simple>*) --> booléen
Exemple: ((a<b) ET (c>d))

<expression_simple> =
| (*<expression_arithmétique_1>*) *<opérateur_comparaison>* (*<expression_arithmétique_2>*)
--> booléen
Exemples: (a) > (b)
(a)> (b+c)
(a+b) > (c+d)
(a+b+c) > (d+e)
(a+b+c+d) > (c)
(a+b+c+d) > (e+f+g+h+i)

|(*<opérande_string_1>* *<opérateur_comparaison>* *<opérande_string_2>)* --> booléen

<expression_arithmétique> =
| *<opérande_numérique>*[*<opérateur_arithmétique>*<*expression_arithmétique*>]

<opérande_string> = constante | littéral | valeur | nom_de_champ (strings)

<opérande_numérique> = constante | littéral | valeur | nom_de_champ (numériques)

<opérateur_logique> = "ET" | "OU" | "XOU"

<opérateur_comparaison> = < | = | > | <= | >= | <>

<opérateur_arithmétique> = + | - | * | /

Nous avons donc défini ce que signifiait une expression, mais nous avons dû introduire des parenthèses afin de construire des expressions plus complexes. Cela implique plusieurs conséquences: d'une part, il faudra résoudre les conflits des parenthèses par rapport à la priorité des opérateurs et d'autre part, telle que nous l'avons définie, une expression n'est qu'une conjonction ou disjonction d'expressions simples ou composées enfermées dans des parenthèses.

Prenons un exemple afin de clarifier ce dernier point:
((a ET b) OU (c ET d)) ET (e OU f) sera une expression correcte. En fait, d'après la notation, cette expression sera (((a ET b) OU (c ET d)) ET (e OU f)), car une expression sera toujours enfermée entre parenthèses.

((a ET b) OU (c ET d)) ET e OU f ne sera pas correcte dans notre notation, car les parenthèses ont l'avantage de régler de manière rigoureuse la priorité des opérateurs. En effet, afin de faciliter le travail de l'utilisateur, on interdira qu'il puisse travailler sans parenthèses. Celles-ci seront automatiquement placées au bon endroit. La deuxième expression serait en effet évaluée (((a ET b) OU (c ET d)) ET e) OU f, auquel cas, le résultat ne serait pas tout à fait celui escompté. Dans l'implémentation, nous donnerons la possibilité à l'utilisateur de modifier le placement des parenthèses, mais alors il faudra avoir un analyseur d'expression afin de détecter des erreurs de type et fournir une aide efficace contre ces erreurs.

La clause suivante est GROUP BY. Cela ne représente pas un problème, car il suffit de choisir un champ qui avait été sélectionné.
La clause HAVING est équivalente à la clause WHERE, nous n'en parlerons pas plus.
La clause ORDER BY sert à trier les résultats sur un des champs sélectionnés, il suffira de dire lequel et si l'on veut l'ordre ascendant ou descendant. Cela est une chose aisée.

Un dernier problème qui nous reste à résoudre est celui d'une série d'opérateurs servant à effectuer une recherche dans la liste de résultats obtenus. En effet, nous considérons que chaque requête SQL renvoie une liste de résultats. Pour cela, formalisons ce que signifie une liste de résultats:

Liste = **SET OF** élément
élément = {type}
type = string | integer | photo

Les opérateurs que nous considérons sont Minimum, Maximum, Moyenne, Nombre (ce dernier étant l'opérateur Count de SQL).
Dès lors, formalisons l'emploi de ces opérateurs:
Soient Liste_résultat = Liste
Liste_entiers, Sous_liste_entier = Liste(integer)

NOMBRE: Liste_résultat --> nombre = #(Liste_résultat)

MINIMUM: Liste_entier --> Sous_liste_entier:
$(\#(\text{Liste\_entier}) \neq 0 \Rightarrow \#(\text{Sous\_liste\_entier}) \neq 0$
$\wedge (\forall e_i \in \text{Sous\_liste\_entier}, \neg\exists e_j \in \text{Liste\_entier}: i \neq j \wedge e_j \geq e_i)$
$\vee(\#(\text{Liste\_entier}) = 0 \Rightarrow \#(\text{Sous\_liste\_entier}) = 0)$

MAXIMUM: Liste_entier --> Sous_liste_entier:
$(\#(\text{Liste\_entier}) \neq 0 \Rightarrow \#(\text{Sous\_liste\_entier}) \neq 0$
$\wedge (\forall e_i \in \text{Sous\_liste\_entier}, \neg\exists e_j \in \text{Liste\_entier}: i \neq j \wedge e_j \leq e_i)$
$\vee(\#(\text{Liste\_entier}) = 0 \Rightarrow \#(\text{Sous\_liste\_entier}) = 0)$

MOYENNE: Liste_entier --> moyenne = #(Liste_résultat)
$(\#(\text{Liste\_entier}) \neq 0 \Rightarrow \text{moyenne} = (\sum e_i)/\#(\text{Liste\_entier}))$
avec e_i $e_i \in$ Liste_entier $\forall i$
$\vee$ (#(Liste_entier) = 0 $\Rightarrow \neg\exists$ moyenne (la moyenne n'est pas définie)

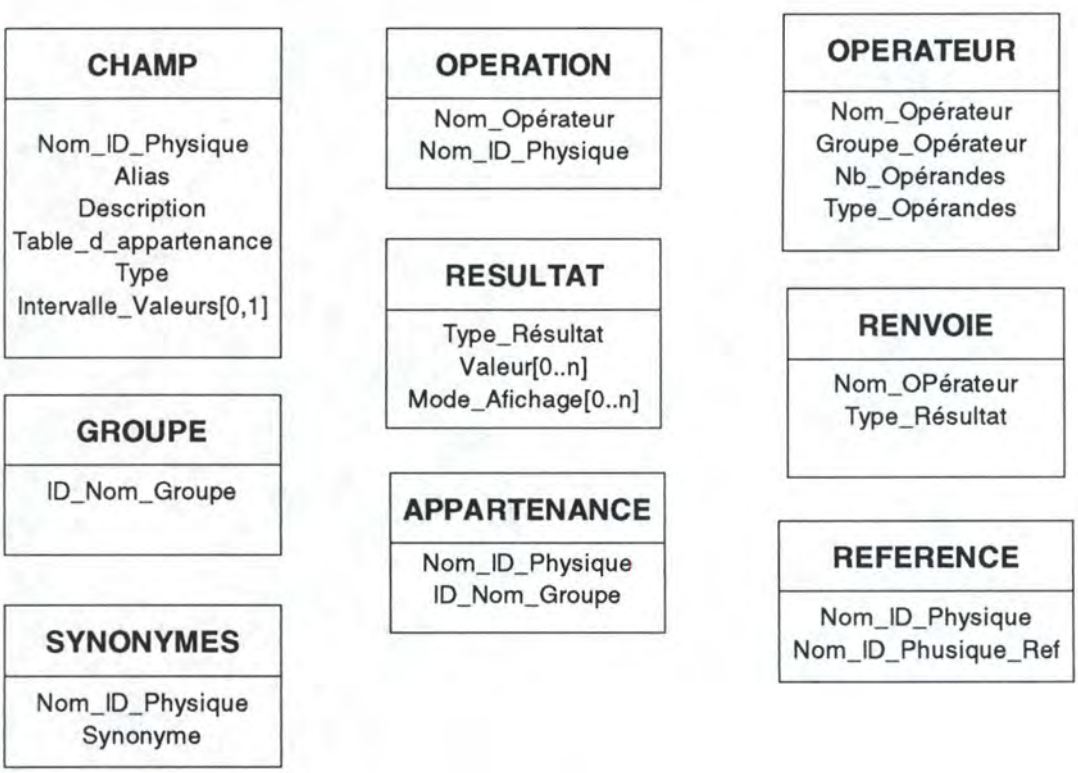
5.4 CONCLUSION

Nous avons étudié la manière de simuler les requêtes du langage SQL, de telle sorte que l'utilisateur ait un moyen simple et graphique afin de se construire ses propres requêtes. Il va falloir penser à construire cet outil, à partir des spécifications que nous en avons donné. C'est ce que nous allons faire dans la partie suivante.

5.5 DEVELOPPEMENT DU MODULE SENSITIVE QUERY

5.5.1 Le catalogue

Il nous faut tout d'abord développer le catalogue de la base de données de *Sensitive Car*. Pour cela, il suffit de le dériver du schéma conceptuel que nous avons obtenu dans le chapitre précédent.



5.5.2 Les chemins

Les tables que nous obtenons sont les suivantes:

5.5.3 L' interface

A partir d'ici, nous pouvons déjà imaginer à quoi ressemblera la simulation de la clause SELECT...FROM... Il s'agira de pouvoir choisir des informations relatives à des véhicules. Or, selon le catalogue, nous pouvons nous intéresser à plusieurs types d'informations: celles provenant de la table champ (l'attribut Alias), les inforamtions en référence à un champ, les synonymes et les groupes d'informations. Nous aurons donc une liste pour chaque provenance d'information et une liste résultante qui représente les informations sélectionnées par l'utilisateur.

La clause FROM est implicite puisque lorsqu'un alias est sélectionné, il suffit de vérifier le champ Table_d_appartenance afin de savoir la provenance du champ sélectionné. Nous avons donc déjà une partie de la requête:
SELECT *champ1, champ2, ..., champn*
FROM *tab_appart_champ1, tab_appart_champ2, ..., tab_appart_champn*
Une vérification sera bien entendu faite, au cas où plusieurs champs appartiendraient à la même table.

Nous avons déjà une requête, car les clauses successives sont optionnelles.
En ce qui concerne la clause WHERE, la condition la plus simple est de na pas avoir de condition, auquel cas la clause WHERE est une clause vide et donc inexistente.
La deuxième possibilité est d'avoir une expression simple. Il faut envisager les cas prévus par notre notation BNF. Remarquons au passage que chaque expression simple est enfermée automatiquement dans des parenthèses.

Il faut donc prévoir une liste de champs (uniquement ceux sélectionnés par la clause SELECT). Lorsque l'utilisateur sélectionne un de ces champs (considérés ici comme des opérandes), on sait directement quels opérateurs proposer, puisqu'on connaît le type du champ. Le type de la seconde opérande est lui aussi connu et on propose de nouveau la liste de champs (du type adéquat uniquement). Cependant il faut aussi donner la possibilité à l'utilisateur de donner une valeur (par exemple, l'utilisateur veut les options dont le prix est inérieur à 15.000 francs. Il devra pouvoir donner la valeur 15.000 et la requête ressemblera à

"SELECT nom_Element, prix_option
FROM Element WHERE (prix_option < 15.000)",
les parenthèses étant ajoutées directement par l'analyseur d'expression.
Après cela, l'utilisateur pourra soit lancer la requête, soit construire une expression plus complexe, par l'ajout d'un opérateur logique. Dans ce cas, l'expression construite jusque là est entourée de parenthèses et la liste des champs sélectionnés est reproposée.

Il faudra bien entendu que le type de l'expression qu'on fournit soit un type compatible avec l'opérateur logique (donc que cette expression fournie retourne un type booléen). Il est à noter que l'expression générale devra impérativement faire intervenir au moins un des champs sélectionnés, car une clause WHERE (2+2 = 4) n'aurait pas de sens, celle-ci étant soit toujours vérifiée(comme dans l'exemple) soit jamais vérifiée (2+2=5), mais n'implique en rien les champs de la base de données.

Afin d'ajouter la clause GROUP BY, on donne la possibilité à l'utilisateur d'obtenir la liste des champs qu'il avait sélectionné par la clause SELECT, afin qu'il choisisse ceux par rapport auxquels il veut grouper ses données.
La clause HAVING est similaire à la clause WHERE.
Afin de trier les données obtenues - la clause ORDER BY - (pour autant que cela ait un sens - il n'y en a pas si on obtient de données du type photo), il faut que l'utilisateur ait la liste des champs qu'il avait sélectionné par la clause SELECT et qu'il puisse décider l'ordre du tri (ascendant ou descendant).

Une fois que cela est fait (rappelons qu'uniquement les clauses SELECT et FROM sont nécéssaires), l'utilisateur peut lancer la requête et obtenir le résultat de celle-ci. Ce résultat sera un sous-ensemble des données contenues dans la base de données de *Sensitive Car*. Ce sous-ensemble sera soit une liste de champs, soit la liste vide, le sous-ensemble vide, soit un élément, un singleton. Cela va bien si on obtient des données numériques ou string, on peut manipuler cette liste et fournir éventuellement des fonctions de recherche dans cette liste (on appelle cela un *SubQuery*). Dans le cas des photos, on aura uniquement l'affichage et l'agrandissement éventuel.
Afin de clarifier ce que nous venons de dire, il est préférable de consulter les annexes B1, qui contiennent les copies des écrans successifs.

5.6 LIMITES DU MODULE SENSITIVE QUERY

Le problème que nous avons avec *Sensitive Query* tel que développé jusqu'à présent, est l'utilisation des parenthèses. En effet, si nous analysons l'expression suivante telle que générée par le module, (((a < b) ET (c > d)) OU (e = f)), celle-ci a été construite de la manière suivante:
construction d'une expression simple: (a < b),
construction d'une expression composée: ((a < b) ET (c > d))
construction d'une expression: (((a < b) ET (c > d)) OU (e = f))
Or, il se peut que l'utilisateur voulait ((a < b) ET ((c > d)) OU (e = f))), ce qui n'est pas tout à fait la même chose. Il y a donc un problème quant à l'affectation des parenthèses dans la génération automatique des expressions.

Une autre limite concerne un aspect des requêtes que nous avons volontairement laissé de côté, dans la perspective d'un développement futur. Il s'agit des jointures sur une seule table. Ce cas, bien qu'existant, ne représente pas un intérêt majeur dans le cadre du *Sensitive Car*.

Un dernier point représentant une limite au *Sensitive Query* est l'utilisation des quantificateurs. En effet, si nous voulons par exemple connaître tous les pays qui ont un

constructeur, nous devrions écrire une requête où intervindrait un quantificateur **exist** dans la clause WHERE. Cette requête serait la suivante:

```
SELECT nom_pays
FROM Nation
WHERE EXIST (SELECT nom_marque
             FROM marque
             WHERE marque.id_pays = Nation.id_pays)
```

Comme nous le voyons, cette requête un peu particulière ne sélectionnera que les noms de pays qui interviennent dans la relation avec marque (pour lesquels il **existe** une marque). Ce type de requête ne sera pas disponible dans la version actuellement développée.

5.7 SOLUTIONS

a) Les parenthèses

La première solution est de laisser les choses telles qu'elles sont. C'est-à-dire que l'utilisateur peut se rendre compte lui-même que l'expression
((a < b) ET ((c > d)) OU (e = f))) est équivalente à (((c > d)) OU (e = f) ET (a < b)).
Cela ne semble pas évident pour un non-informaticien.

La deuxième solution qui vient à l'esprit est de donner le choix à l'utilisateur de placer ses parenthèses. Cela implique que celui-ci soit conscient de la différence de signification entre les deux expressions présentées ci-dessus. Cela semble très peu probable.

La troisième solution est de permettre la construction d'une expression de manière progressive. C'est-à-dire que l'utilisateur doit être au courant de l'existence d'expressions simples, composées et d'expressions qui sont une combinaison, moyennnant des opérateurs logiques, de ces deux types d'expressions. Cela semble peu évident aussi, mais on a néanmoins l'avantage d'éviter le désagrément lié au problème posé. D'autre part, si on regarde la définition que nous avons donné d'une expression, on peut considérer qu'on peut toujours se ramener à une combinaison d'expressions simples. Cela veut dire qu'on peut donner la possibilité à l'utilisateur de construire d'abord toutes ses expressions simples, ensuite de les combiner progressivement en expressions composées et enfin de les recombiner afin d'obtenir une expression.

Cela semble assez difficile, mais il faut également estimer qu'un utilisateur ne construira pas non plus une expression faisant intervenir dix expressions composées. A ce stade de difficulté, même un informaticien aura des difficultés à énoncer une expression qui traduise réellement ses désirs. Il ne faut donc pas non plus s'attendre à des miracles de la part d'un utilisateur normal. Enfin, une dernière remarque s'impose: si l'expression n'est composée que de conjonctions (resp. disjonctions) d'expressions composées, elles-mêmes composées de conjonctions (resp. disjonctions) d'expressions simples, le problème ne se pose pas. En effet, les ET (resp. OU) sont associatifs. Ce n'est que
lorsque nous avons une combinaison de ET et de OU qu'il y a ambiguïté possible.

b) La jointure sur une seule table

Laissé à un dévelopement futur.

c) Utilisation de quantificateurs

Laissé à un dévelopement futur.

5.8 IMPLEMENTATION

- Dans la version développée jusqu'à présent, nous ne prévoyons pas de pouvoir utiliser les opérateurs Minimum, Maximum, Nombre et Moyenne.
- Il ne sera pas non plus possible de choisir la valeur d'un champ dans une liste de valeurs.
- Une autre simplification concerne la construction d'une expression simple de type numérique.
  Celle-ci se limitera à une expression d'un des quatre types suivants:
  ((a+b) < (c+d))
  (a < (b+c))
  ((a+b) < c)
  (a<b).
- Il ne sera pas possible de construire des requêtes avec des jointures sur une seule table.
- Il ne sera pas possible de construire des requêtes faisant intervenir des quantificateurs.

6. LE MODULE CLASSEMENT

6.1 INTRODUCTION

Dans ce paragraphe, nous nous proposons de fournir une aide en ce qui concerne le choix de véhicules basé sur des critères. Nous entendons par là que l'utilisateur donne des critères de choix au logiciel et ce dernier choisit un ensemble de véhicules *correspondant* aux critères. Cela représente une difficulté énorme, car s'il peut être évident de vérifier si un véhicule répond à certaines caractéristiques ou critères, il en est tout à fait autrement pour d'autres critères. En effet, on peut aisément vérifier qu'une voiture roule à l'essence ou au diesel, si elle a trois, quatre ou cinq portes, mais il n'est pas évident de décider si une voiture répond au critère prix de 500.000 francs alors que son prix réel est de 520.000 francs. Faut-il la considérer bonne ou non? De plus, si nous avons à considérer également l'équipement, peut-être sera-t-elle meilleure qu'une autre voiture qui elle coûtera disons 495.000 francs, mais moins bien équipée.

Un autre problème qui sera résolu, une fois que les véhicules répondant aux critères auront été choisis, sera celui du classement afin de déterminer quel est le *meilleur* véhicule qu'on puisse proposer, vu les critères fournis. Cela représente également une grande difficulté. C'est ce genre de problème que nous nous proposons de résoudre avec le module *Classement*.
Le lecteur pourra trouver une approche plus formelle dans [FICHEFET95] pour une approche théorique du problème et de ses solutions et dans [VINCK89] pour une approche formelle du problème et des différentes méthodes développées pour apporter une solution.

6.2 FORMALISATION DU PROBLEME DE RECHERCHE DE VEHICULES

Nous allons à présent donner une définition plus rigoureuse du problème, de la notion de critères que nous considèrerons et de certains choix qui ont été faits dans le cadre de cette étude.
Tout d'abord, nous considèrerons plusieurs types de critères auxquels doivent "matcher" les versions sélectionnées:

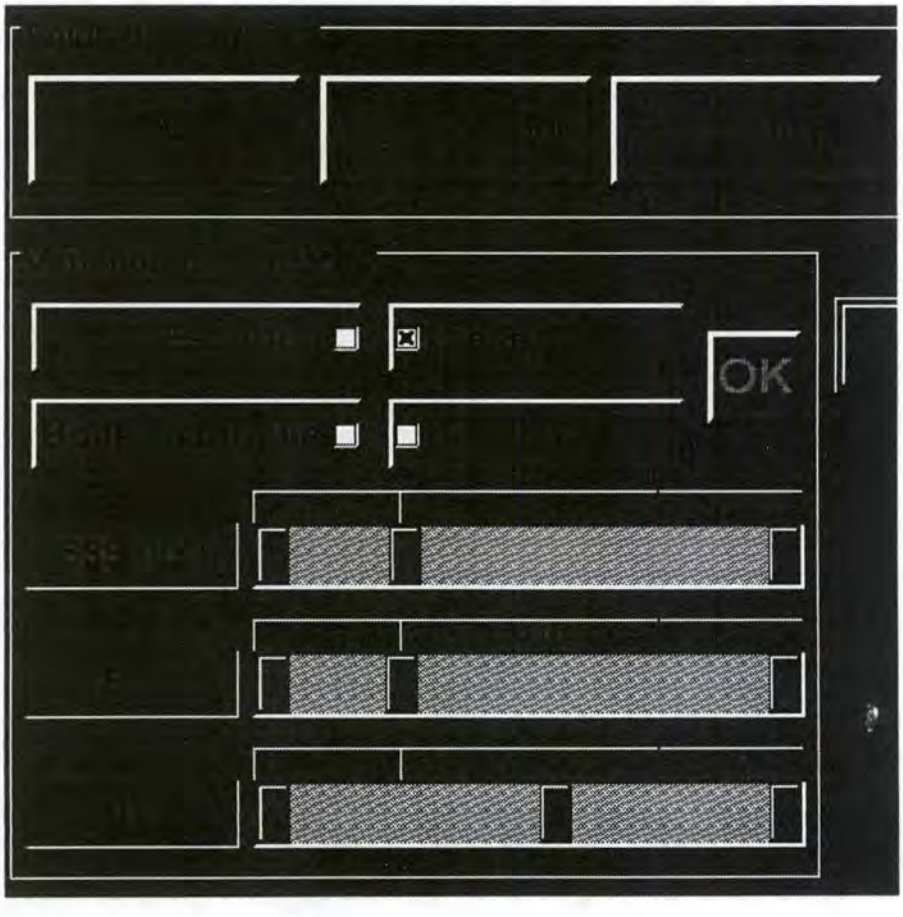
- Les critères **impératifs**: il s'agit de la marque, catégorie, modèle, nation et séries spéciales. Ces critères se rapportent aux modèles. En effet, il semble évident d'appeler ce type de critère impératif, car si on désire les véhicules de telle nationalité ou de telle marque, il faut impérativement que les véhicules sélectionnés par la méthode utilisée respectent ces critères. Le résultat est une liste éventuellement vide, car les nations par exemple n'ont pas forcément de marque ou alors il n'y a pas toujours de séries spéciales. Toutefois, nous répétons que si une version est choisie, elle doit obligatoirement répondre à ces critères à 100%.

- Les critères **variants**: il s'agit du nombre de portes, type d'alimentation, traction, type de boîte de vitesse, nombre de cylindres,... Ce sont des critères se rapportant aux versions. Le résultat est une liste éventuellement vide. Cela dépend de l'ordre dans lequel on considère ces critères, car, par exemple, si je désire les voitures diesel, alors il ne sera pas possible de sélectionner les voitures à boîte automatique (on suppose que ces deux critères sont incompatibles). Par contre, si je sélectionne d'abord la boîte automatique, ce sera le diesel qui sera indisponible.
  De plus, ces critères variants ne seront disponibles que pour les critères impératifs.

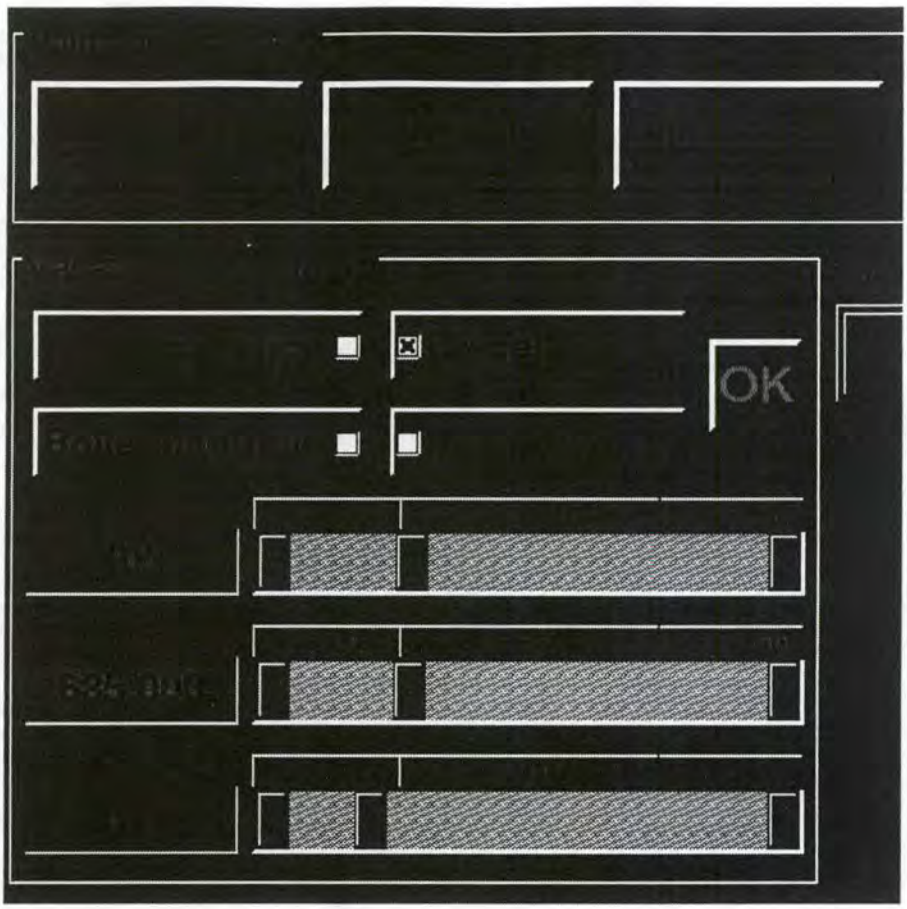
ex: si dans la catégorie "prestigieuse" sélectionnée dans les critères impératifs il n'y a pas de diesel, on ne pourra pas sélectionner diesel dans les critères variants.
Donc, si on a un passé (critères impératifs), on hérite de ce passé.
Les versions sélectionnées doivent absolument répondre à ces critères à 100%.

- Les critères **flous**: Il s'agit du prix, de la puissance et de la consommation. Leur ordre a de l'importance. Donc, les bornes correspondantes (limites inférieures et supérieures du prix, de la puissance et de la consommation) sont initialisées selon l'ordre d'apparition de ces critères. Donc, si on estime que le prix est plus important que la puissance, c'est pour la borne minimale (resp. maximale) du prix que l'on donnera la borne minimale (resp. maximale) de la puissance. Il faut, en outre, pouvoir travailler avec des intervalles explicites donnés par l'utilisateur.
  Le résultat est une liste éventuellement vide. Une version sélectionnée doit répondre *approximativement* à ces critères. Si aucune version ne correspond à ces critères flous, on propose à l'utilisateur d'agrandir les intervalles s'il travaille avec des intervalles ou de donner un seuil de tolérance plus grand afin de considérer un nouvel intervalle implicite (au cas où l'utilisateur ne travaille pas avec des intervalles explicites, mais avec un budget si on parle du prix). Par exemple, si on cherche des voitures de 700.000 francs avec un seuil de 5%, on aura un intervalle implicite allant de 665.000 francs à 735.000 francs et on est ramené au cas des intervalles explicites. A ce stade, les bornes sont fixées.

Il est certainement utile de montrer une partie de l'écran concernant les critères flous.



En modifiant l'ordre des critères, nous obtenons l'écran suivant:



Il faut remarquer que les critères sont toujours conjonctifs, mais si on les considère au sein d'un type de critère, ils sont disjonctifs. Par exemple, si je sélectionne les marques m1 et m2 et les catégries c1 et c2, il faudra comprendre:
"(marque = m1 OU marque = m2) ET (catégorie = c1 OU catégorie = c2)"

6.3 ALGORITHME GENERAL DE RECHERCHE

Nous pouvons donc considérer deux manières de procéder, selon que l'on désire demander un certain équipement que devra avoir le véhicule sélectionné ou non.
L'algorithme pourrait ressembler à ce qui suit:

SI On choisit de s'intéresser à l'équipement
ALORS {
SI on a un passé
(c-à-d qu'on a choisit des critères impératifs et/ou variants et/ou flous)
ALORS {
- on n'a que les groupes d'équipements disponibles pour l'héritage (tenir compte des critères impératifs et/ou variants et/ou flous)
(groupe = sécurité, confort, ...)
}
SINON {
- on a tous les groupes d'équipements de la BD

}
- Sélectionner les équipements désirés au sein des groupes
- Fixer l'ordre d'importance éventuel des **groupes d'équipements**
- Lancer la recherche en tenant compte de l'équipement
}

SINON Lancer la recherche sans tenir compte de l'équipement.

Une variante pourrait être la suivante: on ne s'intéresse pas aux groupes, mais uniquement aux équipements. On fixe donc l'ordre d'importance des équipements choisis et non des groupes. Il faut peut-être donner le choix de la méthode à l'utilisateur.

SI On choisit de s'intéresser à l'équipement
ALORS {
SI on a un passé
(c-à-d qu'on a choisit des critères impératifs et/ou variants)
ALORS {
- on n'a que les groupes d'équipements disponibles pour l'héritage (tenir compte des critères impératifs et/ou variants) (groupe = sécurité, confort, ...)
}
SINON {
- on a tous les groupes d'équipements de la BD
}
- Sélectionner les équipements désirés au sein des groupes
- Fixer l'ordre d'importance éventuel des **équipements**
- Lancer la recherche en tenant compte de l'équipement (voir plus loin ce que cela signifie)
}

SINON Lancer la recherche sans tenir compte de l'équipement.(voir plus loin ce que cela signifie)

Le résultat est une liste éventuellement vide. Les versions éventuellement sélectionnées doivent répondre "approximativement" aux critères, tout en tenant compte du meilleur équipement possible.

6.4 VERIFICATION DU RESULTAT

Le résultat final est donc une liste de versions éventuellement vide (remarquons que la liste peut contenir deux foix la même version, mais avec un équipement différent et donc probablement un prix différent. Ces véhicules seront considérés comme deux versions différentes. Il faudra faire de telle sorte qu'on réduise le risque d'obtenir une liste vide. Pour cela, à chaque étape (c-à-d pour chaque type de critère envisagé) il faudra minimiser le risque:

Pour les **critères impératifs**: dès qu'on envisagera de choisir un véhicule selon les critères, si on a une liste vide, on considère que l'utilisateur s'intéresse à tous les véhicules. Donc s'il ne choisit pas de critères impératifs (marque, nation,...), on fait comme s'il choisissait toutes les marques, tous les pays,... On est obligé de faire ainsi, car autrement, à partir d'une liste vide, on ne sait pas obtenir une sous-liste non vide.

Pour les **critères variants**: Il se peut qu'en faisant certains choix de critères on obtienne une liste vide (ex: pas de diesel à boîte automatique). Au départ (après l'étape ci-dessus), on a une liste non vide. Lorsqu'on a choisi les critères variants, on doit obtenir également une liste non vide. Si ce n'est pas le cas, c'est qu'on a sûrement choisi des critères variants incompatibles. Il faut donc en aviser l'utilisateur et annuler ces critères (on recommence pour les critères variants). On ne va plus loin que si on a une liste non vide qui est donc une sous-liste de la liste obtenue par le choix des critères fixes, car on ne recherchera que les véhicules répondant aux critères variants, au sein de la liste de véhicules répondant aux critères fixes.

Pour les **critères flous**:

SI on a un passé (critères impératifs et/ou variants)
ALORS (les intervalles sont ceux correspondant aux critères choisis)
SINON (les intervalles sont ceux correspondant à tout véhicule de la BD)
SI on travaille avec des intervalles (sur le critère le plus important uniquement)
ALORS la recherche se fait sur l'intervalle

Pour éviter que le résultat ne soit une liste vide, au cas où il n'y aurait aucune version, on avise l'utilisateur et on agrandit l'intervalle de recherche d'une quantité correspondant au seuil de tolérance si l'utilisateur en a donné ou par agrandissement successif de l'intervalle, jusqu'au moment où une version est trouvée (on peut éventuellement aider l'utilisateur en lui proposant un seuil de tolérance afin d'être sûr de trouver un véhicule).
On finira par trouver une version au moins, car le pire des cas pourrait nous amener à considérer tout l'intervalle de valeur de la base de données (si le critère le plus important est le prix, on considèrera donc l'intervalle dont la borne inférieure est la voiture la moins chère (pour les critères éventuellement choisis précédemment) et la borne supérieure la voiture la plus chère (pour les critères éventuellement choisis précédemment)). On est donc sûr de trouver.

SINON (on travaille sans intervalle)

rem: cela signifie qu'on a demandé une valeur. Exemple: prix = 500.000. Donc min = 500.000 et max = 500.000.

Ce n'est qu'un cas particulier du cas avec les intervalles, il suffit de considérer que l'on a un intervalle dont la borne inférieure = la borne supérieure. Donc on agrandit l'"intervalle" avec une des deux méthodes proposées au cas précédent. On continue jusqu'à ce qu'on trouve une version au moins.

Conclusion: La liste qu'on obtiendra ne sera jamais vide. Une version au moins est toujours trouvée.

Pour l'**équipement**:

SI on n'a pas de passé
ALORS on propose toutes les options disponibles dans la BD. L'utilisateur les choisit par ordre d'importance (par groupe et/ou équipemnt).

rem: - importance par groupe:
ex: supposons que l'utilisateur choisisse l'ABS, le climatiseur et le cuir, ces derniers équipements appartenant au groupe Confort, tandis que le premier appartient au groupe Sécurité; importance(climatiseur) > importance(cuir) et importance(Sécurité) > importance(Confort)
⇒ importance(ABS) > importance(climatiseur et cuir)

et importance(ABS) > (importance(climatiseur) > importance(cuir))
le groupe prime donc sur les équipements.

- importance par équipement:

ex: supposons que l'utilisateur ait choisi les mêmes options que ci-dessus, mais que l'on ne considère pas l'importance du groupe. L'utilisateur classe directement l'importance d'une option.

⇒ importance(climatiseur) > importance(ABS) > importance(cuir)
l'équipement prime sur le groupe.

Le choix de la méthode peut être laissée au soin de l'utilisateur, tout en remarquant que la deuxième n'est qu'une généralisation de la première. En effet, l'utilisateur peut opérer lui-même le classement des options par groupe. Dans la version que nous allons implémenter, nous procèderons au classement des équipements et non des groupes.
Le résultat donne-t-il une liste non vide?
Ce n'est pas dit, car il se peut très bien que l'utilisateur sélectionne tous les équipements et veuille savoir quels sont les véhicules équipés de tous les équipements proposés. Posé de cette manière, le problème admettra difficilement une solution. C'est pourquoi nous avons introduit la notion d'importance d'un équipement par rapport à un autre. En effet, cela sert à "écarter certains équipements" s'ils sont moins importants. Tout le problème réside à savoir quels équipements écarter.
Peu importe la méthode, nous obtenons une liste non vide de versions, quitte à ce que l'équipement ne soit pas exactement celui demandé. Mais le problème dans cet exemple est résolu par le fait que l'utilisateur n'a pas d'autres critères (pas de passé), le prix du véhicule n'a pas d'importance non plus. Les véhicules plus chers (en général mieux équipés) seront donc plus souvent sélectionnés.

SINON (on a un passé) on propose uniquement les équipements des véhicules qui répondent aux critères demandés.

Cela semble une chose aisée, mais il n'en est rien, car si on peut exactement faire une vérification pour les critères impératifs et variants, il en est tout autrement pour les critères flous. En effet, il faudra se ramener à une des méthodes considérées ci-dessus (choix des intervalles explicites ou implicites) et lorsqu'on ajoute le prix des options, il faut vérifier qu'on reste bien dans les intervalles. Si on en sort de manière acceptable, il faut encore définier ce que cela signifie. Faut-il ajouter un seuil au seuil de tolérance? Nous verrons plus loin.
Un autre problème à résoudre est le cas de véhicules ne répondant pas aux critères flous (lorsque le plus important est le prix), mais sont tels que si on ajoute le prix d'un ou plusieurs équipements, il répondent aux critères. Afin de résoudre ce problème, nous allons devoir envisager les deux méthodes d'équiper le véhicule:

1) L'équipement est fourni si et seulement si cela est possible:
Supposons que l'on ait n options sélectionnées. Par la manière de classer les options, on donne un poids à chaque option, afin d'exprimer son importance. La plus importante aura un poids n, la moins importante aura un poids 1.
On peut également donner le même poids à plusieurs équipements et au cas où l'utilisateur donnerait un poids maximal ou minimal à tous les équipements, on estime que les équipements ont tous la même importance.
On sélectionne tous les véhicules qui ont leur prix de base en-dessous de la borne inférieure de l'intervalle. Pour chacun de ceux-ci, on cherche à l'équiper avec des équipements sélectionnés (en commençant par le plus important) afin d'arriver dans

l'intervalle considéré. Le véhicule ainsi équipé sera lui aussi gardé afin de procéder au classement de tous les véhicules trouvés.

2) On veut absolument l'équipement et, dans ce cas, il faut faire des concessions quant aux bornes.
SI le critère flou le plus important est le prix
ALORS
Nous avons le même problème que dans le cas envisagé précédemment, mais en plus, il faut estimer de combien l'utilisateur est prêt à sortir de l'intervalle (puisque c'est une question d'argent).
Pour cela, on demande à l'utilisateur, pour chaque véhicule trouvé, s'il est d'accord avec le prix envisagé. Il faut absolument agir de cette façon, car nous ne pouvons pas juger à la place de l'utilisateur. La question à laquelle le logiciel répond est alors la suivante: quels sont les véhicules qui possèdent les équipements jugés importants par l'utilisateur et qui entrent dans les prix de cet utilisateur, tout en respectant les autres critères?
SINON le problème n'est pas une question d'argent
Il faut rechercher des véhicules qui sont équipés comme le veut l'utilisateur, tout en demandant à l'utilisateur s'il est d'accord avec la puissance ou la consommation proposée.

Le résultat donne-t-il une liste non vide?
Pour être sûr, il faut considérer deux cas:
1) L'équipement est fourni si et seulement si cela est possible (on reste dans les bornes). Autrement on dit à l'utilisateur que cela n'est pas possible et on garde les versions telles qu'elles avaient été sélectionnées avec les critères impératifs, variants et flous.
2) On veut absolument l'équipement et dans ce cas, il faut faire des concessions quant aux bornes. Cela veut dire qu'on réussira à avoir des versions équipées selon les désirs de l'utilisateur, mais sans respecter les bornes. Cette manière de faire peut sembler inutile a priori, mais elle se justifie tout de même. En effet, un utilisateur peut vouloir connaître le prix qu'il devra payer pour obtenir un véhicule équipé comme il l'entend, tout en restant *proche* des limites qu'il s'était fixé. Peut-être que la différence ne lui semblera pas très grande et qu'il se décidera à payer cette différence s'il estime que cela en vaut la peine (c'est entre autre pour cela que ce logiciel est un SIAD, mais n'anticipons pas sur le châpitre qui leur est consacré).
Au cas où l'utilisateur ne veut pas faire de concessions, on se ramène au cas précédent. On avertit toutefois l'utilisateur qu'il n'obtient pas tout à fait ce qu'il désire.

Finalement, nous obtenons bien une liste non vide de versions. Pour le premier cas considéré, ce sera au pire exactement la liste des versions fournie par les critères impératifs, variants et flous. Au mieux, on aura une sous-liste (non vide) de versions de la liste obtenue, auxquels on a ajouté des équipements. Une sous-liste d'une liste non vide est une liste et telle qu'elle a été construite, elle sera non vide.
Pour le deuxième cas, si on reste dans les intervalles fixés par les critères impératifs, variants et flous, on se ramène au premier cas. La liste qu'on obtient est donc non vide. Si on sort des bornes, on agrandit l'intervalle et on procède à la recherche. Donc on aura plus de versions qu'avec les critères flous (les critères impératifs et variants doivent de toute façon être respectés, car si l'utilisateur voulait une grande routière, essence à cinq portes à boîte automatique et bien équipée pour un prix de x milliers de francs, ça ne sert à rien de lui proposer une petite polyvalente sous prétexte qu'elle est très bien équipée), mais en sachant

que les véhicules correspondant aux critères flous ne répondent pas aux attentes relatives à l'équipement. La liste obtenue sera donc non vide. Dans le pire des cas, on aura comme intervalle celui formé par la borne inférieure et la borne supérieure de la BD (les valeurs dépendent du critère flou le plus important: si c'est le prix, ce sera la voiture la moins chère (resp. plus chère) pour la borne inférieure (resp. supérieure), si c'est la puissance, on aura la voiture la moins (resp. plus) puissante). De par la méthode esposée, on obtiendra une liste non vide de versions qui correspondent le mieux possible à ce qu'on cherche.

Conclusion: Lorsque la procédure de recherche est lancée, on finit par obtenir, au vu de tout ce que nous avons dit, une liste non vide de versions qui répondent "au mieux" aux critères fournis. En effet, à chaque étape (chaque type de critère), nous avons démontré qu'on obtenait une liste non vide de versions. En agrégeant toutes les étapes, on obtient une liste non vide de versions, peut-être même un seul véhicule.

6.5 FORMALISATION DU PROBLEME DE CLASSEMENT DE VEHICULES

Dans ce qui va suivre, nous nous proposons de fournir un classement des véhicules sélectionnés dans l'étape précédente.
Nous nous baserons essentiellement sur la méthode ELECTRE II améliorée, afin de proposer un classement des véhicules sélectionnés. Se référer à [FICHEFET95] et [VINCK89] pour des justifications théoriques. Cette méthode a été choisie à cause de sa popularité dans le domaine de classements d'actions avec ex aequo.

6.5.1 Rappel de la méthode ELECTRE II améliorée:

Soit A = {a, b,c,...z} un ensemble fini d'actions. Dans ce cas, il s'agit de la liste de voitures que l'on désire classer.

Soient $J^+(a,b) = \{j \in C \mid a\ P_j\ b\}$ = nombre de critères pour lesquels a est préféré à b
$J^=(a,b) = \{j \in C \mid a\ I_j\ b\}$ = nombre de critères pour lesquels a et b sont indifférents
$J^-(a,b) = \{j \in C \mid a\ P_j\ b\}$ = nombre de critères pour lesquels b est préféré à a

De même, nous définissons les poids relatifs comme suit:
$W^+(a,b) = \sum_{j \in J^+(a,b)} p_j$ = somme des poids des critères pour lesquels a est préféré à b
$W^=(a,b) = \sum_{j \in J^=(a,b)} p_j$ = somme des poids des critères pour lesquels a et b sont indifférents
$W^+(a,b) = \sum_{j \in J^-(a,b)} p_j$ = somme des poids des critères pour lesquels b est préféré à a
Les trois indices de concordance sont les valeurs standard:
$c_1 = 3/4$, $c_2 = 2/3$, $c_3 = 3/5$, de telle sorte que $c_1 > c_2 > c_3$

Les seuils de discordance sont, deux par critère: $d_{2j} > d_{1j} > 0$

Afin de départager les ex aequo éventuels, la méthode ELECTRE II améliorée, de même que la méthode ELECTRE II officielle, introduit une notion supplémentaire, par rapport à la méthode ELECTRE I:

a est assurément au moins aussi bonne que b
a est vraisemblablement au moins aussi bonne que b

Cela induit à définir deux relations de surclassement: une relation de surclassement forte (S_F) et une relation de surclassement faible (S_f), définies comme suit:

$a\ S_F\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_1$
et $\forall j \in J^-(a,b)$: $g_j(a) - g_j(b) < d_{2j}$

ou

$a\ S_F\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_2$
et $\forall j \in J^-(a,b)$: $g_j(a) - g_j(b) < d_{1j}$

$a\ S_f\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_3$
et $\forall j \in J^-(a,b)$: $g_j(a) - g_j(b) < d_{2j}$

Avec ces deux fonctions, on construit le graphe correspondant au surclassement fort (resp. faible). Si deux actions seront classées ex aequo, on utilisera le surclassement faible pour les départager.

Afin de déterminer le classement, on effectue d'abord une découpe par niveaux du graphe [FICHEFET92], si c'est possible (existence éventuelle de circuits), en considérant uniquement la relation S_F.
La deuxième étape consiste à effectuer une découpe du graphe par plateaux, si cela est possible, pour le même motif [FICHEFET92], en considérant S_f.
Ensuite, pour tout sommet (action - véhicule), on somme le numéro qu'il a obtenu dans le classement avec le tri topologique. On trie de manière ascendante tous les classements obtenus (les sommes qu'on vient d'effectuer correspondent au classement du sommet) et on obtient ainsi le préordre total final. Ainsi, si le sommet n° i est classé en j° position, cela signifie que le véhicule n°i (ou toute autre identification du véhicule) est le j° dans le classement final (avec des ex aequo possibles).

Si on obtient *plus ou moins* (laissé à l'appréciation de l'utilisateur) le même classement, cela signifie que le classement obtenu est pertinent. Autrement, il y a quelque chose qui ne va pas. Il faudra que l'utilisateur recommence la procédure de recherche en affectant d'autres poids aux critères, en donnant un maximum d'informations au logiciel afin que la recherche non seulement soit pertinente, mais qu'on n'essaie pas de comparer l'incomparable. Ceci pourrait effectivement arriver si le logiciel ne dispose pas d'assez d'informations (cela peut être dû au nombre de critères envisagés par l'utilisateur insuffisant, bornes trop grandes,...) afin de rechercher des véhicules correspondant aux critères. Ce problème peut être une limite dans l'emploi d'une méthode mathématique quant au choix et au classement d'un véhicule. Cela n'est nullement dû à la méthode envisagée, mais exclusivement au manque de connaissances

spécifiques de la part du logiciel. En effet, l'espace de recherche est d'autant plus grand que les critères sont peu nombreux ou que l'intervalle des critères flous est grand. Cela provient de ce que l'utilisateur ne sait pas bien ce qu'il veut. Le logiciel doit (peut) donc ratisser large lors de la recherche. D'autre part, nous estimons que si un utilisateur joue le jeu comme il le faut, en ce sens qu'il désire effectivement donner un sens à sa recherche, il n'hésitera pas à formuler ses critères assez finement. Dans ce cas, la méthode donne de bons résultats.

6.5.2 Simplification de la méthode ELECTRE II améliorée :

Choix des poids:

afin de simplifier la méthode pour l'utilisateur, nous fixons des poids de3 pour les équipements estimés très importants, un poids de 2 pour ceux estimés importants et de 1 pour ceux dont on pourrait le plus facilement se passer au cas où il faudrait choisir. Cela semble artificiel et effectivement il en est ainsi, car rien ne pourrait empêcher l'utilisateur de donner des poids 10 - 8 - 2 au lieu de ceux donnés. Il fallait bien choisir, nous nous tiendrons à notre choix.

Construction du graphe:

Une autre version du classement peut se faire comme suit:

On définit deux classements:
Classement direct: a avant b $\Leftrightarrow$ le nombre de flèches partant de a > nombre de flèches partant de b

Classement indirect: a avant b $\Leftrightarrow$ le nombre d'actions surclassant a < nombre d'actions surclassant b
On obtient un classement indirect fort: les sommets qui ont leur nombre plus petit que les autres sont classés en tête. On départage en considérant le surclassement faible.

On procède de même en considérant le classement direct fort.

Seuils de concordance

Nous les avons fixé à des valeurs empiriques, soit 3/4, 2/3 et 3/5. Ici aussi on peut remarquer que cela est fait dans un souci de simplification par rapport à l'utilisateur. Mais cela est malheureusement aussi source d'approximation. Pour éviter cela, il aurait fallu que l'utilisateur fixe lui-même ces seuils. Cela semble une chose assez compliquée, car il faudrait qu'il puisse comprendre exactement à quoi ils servent.

Indices de discordance

Nous n'allons pas utiliser d'indices de discordance. En effet, ceux-ci servent à éviter le *droit de veto*. Lorsque par exemple on cherche à classer les voitures bien équipées, il y a de fortes chances que les voitures classées en premier soient les plus chères (par exemple une Rolls Royce). Comme nous considérons des intervalles de prix, nous n'aurons jamais une voiture très chère qui surclasse toutes les autres. Nous n'avons donc pas besoin des indices de discordance. Voyons donc d'un peu plus près les différents cas qui peuvent se proposer au logiciel.

Les lignes suivantes vont donc être modifiées:

$a\ S_F\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_1$
et $\forall j \in J^-(a,b)$: $g_j(a) - g_j(b) < d_{2j}$
ou
$a\ S_F\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_2$
et $\forall j \in J^-(a,b)$: $g_j(a) - g_j(b) < d_{1j}$
$a\ S_f\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_3$
et $\forall j \in J^-(a,b)$: $g_j(a) - g_j(b) < d_{2j}$

Voici ce que nous devons considérer:

$a\ S_F\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_1$
ou
$a\ S_F\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_2$
$a\ S_f\ b \Leftrightarrow W^+(a,b) \geq W^-(a,b)$
et $C(a,b) \geq c_3$

Avant cela, il faut considérer deux cas, selon que l'on avait demandé un équipement ou non.

1) On ne s'occupe pas de l'équipement:
$\Rightarrow$ on considère uniquement les critères impératifs, variants et flous. Les deux premiers doivent être repsectés à 100%.

SI on trouve des versions dans les intervalles considérés (implicites ou explicites)
ALORS les versions sont considérées comme répondant à 100% aux critères, même les flous.

6.5.3 Pratiquement (ce qui sera implémenté) :

Il faut demander le seuil de tolérance à l'utilisateur (qu'on peut appeler marge d'erreur tolérée), c-à-d de combien on peut sortir des bornes par rapport au premier critère flou. Cet indice s'exprime en pourcentage.
Exemple: si l'utilisateur a choisi de travailler avec des intervalles, on diminie à gauche de la borne inférieure et à droite de la borne sup. (toujours si c'est possible, car il se peut que l'utilisateur ait déjà considéré l'intervalle tout entier. Il n'y aura donc pas de variation par rapport à la borne inférieure, mais bien pour la borne supérieure et ce uniquement si l'utilisateur s'intéresse à l'équipement. Clarifions ce point: si on ne s'intéresse pas à l'équipement, le prix de la voirure recherchée est celui qui est dans la BD, donc on ne sait pas varier plus que la borne supérieure, puisque celle-ci est déjà la valeur maximale trouvée dans la BD. Tandis que si on s'intéresse à l'équipement, on peut dépasser la borne supérieure, cela signifiant que l'on

ajoute à un véhicule des équipements en option et le prix dépasse la borne supérieure que l'on avait pour le prix de base dans la BD. Cela ne fonctionne que si le critère le plus important est le prix. En effet, supposons que ce soit la puissance, le fait d'ajouter le prix d'une option ne fait pas varier la puissance. Il se pourrait juste que si le premier critère est la puissance ou la consommation et le deuxième le prix, en prenant la puissance maximale, on ait le prix maximal (puisqu'il dépend de la puissance maximale) et en ajoutant le prix de l'option on sorte quand-même de la borne supérieure du prix. Il faudra donc vérifier qu'on ne sorte pas de l'intervalle du deuxième critère, car celui-ci est également important (il a un poids de deux).

Récapitulons: l'utilisateur choisit une marge d'erreur. Le pas est de 5%. La marge est affichée. Ainsi on a les bornes définitives (sans compter la marge, voir ci-dessous).
Il faut à présent choisir tous les véhicules tels que les critères flous soient respectés. Si on trouve au moins un véhicule qui est dans les intervalles, alors on le prend (il est bon). Si on ne trouve rien, on agrandit l'intervalle avec la marge d'erreur. Ce n'est donc qu'ici qu'elle intervient et pas directement pour fixer les intervalles.
Si l'utilisateur n'a pas donné de marge, alors on agrandit progressivement: d'abord 5%, puis 10% et normalement on devrait trouver. Dans le pire des cas, on agrandit jusqu'à ce qu'on trouve une version au moins. Rappelons que les bornes des intervalles sont fixées par rapport aux critères impératifs et variants. Ca signifie qu'il y a des véhicules répondant aux critères. Si à chaque recherche, on ne trouve pas, on affiche un messga et on demande si on peut agrandir l'intervalle, avec la marge si le user en a donné une, avec notre méthode autrement. Donc on finira par obtenir un véhicule au moins, à moins que le user arrête la procédure en répondant non au message qu'on lui a envoyé (à propos d'agrandir l'intervalle).

Il faut penser à retenir quelques informations sur les voitures qu'on sélectionne.
Tout d'abord, il faut donner une pondération aux critères. Pour ce faire, donnons trois au premier, deux au deuxième et un au troisième. Il faut garnir un tableau dont les colonnes sont les voitures sélectionnées et chaque ligne un critère. Pour chaque voiture sélectionnée, on remplit le tableau. Comme on considère des valeurs numériques pour les trois critères, on peut construire une matrice de trois lignes et autant de colonnes que de voitures sélectionnées. Les valeurs qu'on mettra dans le tableau, ce seront les valeurs des prix, puissance, consommation des voitures sélectionnées. A partir de ce tableau, on pourra construire le classement des voitures. Nous expliquerons plus loin comment.

2) On désire un équipement:

a) l'équipement est moins important que les critères flous,
on équipe donc le véhicule **si on peut.**

Il faut donc revoir la pondération des critères flous. Il faut que le poids soit plus important. Donc on prend comme poids du critère flou le plus important n+3 (n=nombre d'équipements sélectionnés), le deuxième critère = n+2 et le troisième n+1. En fait le cas traité précédemment est un cas particulier de ceui-ci: n=0.
La matrice qu'on va construire est différente, car elle contient plus de lignes.
Pour la recherche de véhicules, c'est ici que ça change par rapport à au cas précédent.
Plusieurs cas peuvent se présenter:

<u>- le prix est le critère le plus important</u>

dans ce cas, il faut chercher même des véhicules moins chers tels qu'équipés avec les équipements demandés, ils arrivent dans l'intervalle considéré. Comme on a une pondération

pour les équipements, on commence par ajouter au prix de la voiture le prix de l'équipement considéré. Et ainsi de suite. Il ne faut pas oublier de prendre uniquement les véhicules correspondant aux critères impératifs et variants, s'il ont été choisis par l'utilisatur. Autrement, on scanne toute la BD.
Ensuite, il faut garnir le tableau. Pour le prix de la voiture, on donne celui avec l'équipement et non le prix de base. Les lignes supplémentaires correspondent aux équipements considérés. Plusieurs valeurs sont possibles :
pas de valeurs: l'équipement n'est pas disponible.
un nombre: c'est le prix de l'option
0 (zéro) : l'équipement est de série.
On obtient donc un nouveau tableau avec des nombres (plus des blancs). Si en Visual Basic (c'est le langage que no!s utilisons pour l'implémentation) on ne donne pas de valeur, peut-être y aura-t-il un problème, on ne sait pas ce que vaudront les valeurs du tableau où on ne voulait rien mettre (équipement pas disponible). Dans ce cas, on prend comme convention de mettre -1 et il faudra gérer ce cas.
On a le tableau et on peut relancer la procédure de classement, en estimant les choses suivantes:
a est meilleure que b pour l'équipement x si le prix de l'équipement de x pour a est inférieur au prix de l'équipement de x pour b et en même temps supérieur à -1 (il faut qu'il soit au moins disponible pour être meilleur).

- le prix n'est pas le critère le plus important, mais est deuxième.

Ici il faudra repenser à ce que nous avons dit pour le cas où on ne s'occupait pas de l'équipement, car les bornes sont plus larges, mais le deuxième critère est encore un critère imoortant.

- le prix est le critère le moins important

Dans ce cas, on s'en occupe pas pour la recherche, le classement n'est pas aussi compliqué que ce que nous avons expliqué précédemment. En effet, le critère le plus important est soit la puissance soit la consommation. Nous pourrons donc vérifier uniquement que l'équipement demandé soit possible pour les véhicules correspondant aux critères impératifs, variants et flous. Etant donné que le prix ne pose pas de problème (il existe des personnes pour qui le prix d'une voiture n'est pas un élément pouvant poser un problème, eh oui), le fait d'ajouter le prix d'un équipement ne change en rien l'intervalle pour le premier critère flou considéré.

b) l'équipement est plus important que les critères flous
on équipe le véhicule **malgré les bornes fixées, tout en essayant de rester dans celles-ci.**

Il va falloir changer la pondération:
les équipements reçoivent comme pondération leur pondération + 3. Les critères flous gardent leur pondération. Ainsi les équipements sont plus importants.
On doit ici aussi considérer quelques cas:

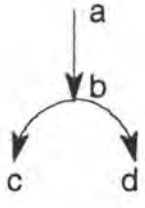
- le prix est le critère le plus important

Ici aussi le prix qu'on met dans le tableau est celui du véhicule avec l'équipement.
Comme l'équipement est plus important, il faut accepter de sortir des bornes et on s'arrête lorsqu'on a réussi à équiper un véhicule.

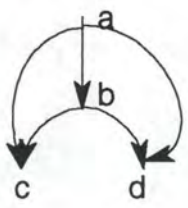
Concrètement, pour chaque version qu'on réussit à équiper comme l'utilisateur veut, on lui demande s'il est d'accord avec le prix. Ce prix on le retiens. Chaque voiture qu'on trouvera par la suite et qui a un prix supérieur, on ne la proposes même pas. Par contre pour les voitures qui ont un prix inférieur on les propose et de nouveau si l'utilisateur refuse, on retient ce nouveau prix comme référence. Donnons un exemple pour clarifier: supposes que l'on proposes une voiture équipée à 700.000 francs et que l'utilisateur refuse parce que c'est trop cher. Dans ce cas, ça ne sert à rien de proposer des autres voitures dont le prix est supérieur à 700.000 francs. La prochaine voiture qu'on proposera aura forcément un prix inférieur à 700.000 fr. Supposons 650.000 fr. Si l'utilisateur est d'accord on la retient, autrement ça devient la nouvelle valeur de référence. On ne proposera que les voitures inférieures à 650.000. Par contre dans la proposition, on pourra également proposer la même voiture, mais en retirant un équipement afin de faire diminuer le prix. Cet équipement doit être un ayant sa pondération la plus basse.
On peut lancer le classement avec les véhicules obtenus par la recherche.

6.6 CONCLUSION

Comme nous l'avons dit dans l'introduction, le but de ce chapitre était de fournir une aide afin de pouvoir choisir des véhicules en fonction de critères, ainsi qu'un classement. Nous avons construit une procédure permettant d'y arriver. Cependant, sommes-nous sûrs que le résultat soit bon? Nous ne pouvons malheureusement pas le garantir, et cela n'est pas dû à une erreur dans le raisonnement suivi, mais étant donné que nous utilisons une méthode mathématique, Electre II, qui modélise donc un problème, il y a des approximations inhérentes à cette méthode, comme à toute autre méthode d'ailleurs. Nous n'allons pas entrer dans le détail des approximations, le lecteur intéressé pourra se référer à [ROY]. Nous allons simplement donner une petite indication à l'aide d'un exemple. Les seuils de concordance que nous avons choisi correspondent à des valeurs empiriques, nous l'avons déjà dit. Cependant, le graphe obtenu pourrait être légèrement faussé et ne pas exprimer réellement les préférences. Prenons l'exemple suivant:



Il indique que a est préféré à b et b est préféré à c et d. Donc globalement, vu que le nombre de flèches partant de b est supérieur au nombre de flèches partant de a, le véhicule b est préféré à a (classé avant a). Or, le graphe indique bien que a est classé avant b. Il y a donc un problème quelque part. Ceci est peut-être tout simplement dû aux indices de concordance. Si on choisissait d'autres valeurs (cela revient à faire une analyse de sensibilité), il serait fort probable qu'on aurait découvert d'autres préférences et donc d'autres flèches dans le graphe. Nous aurions peut-être obtenu le graphe suivant:



Le véhicule a aurait donc été classé en première position, le b en deuxième et le c et d ex aequo en troisième position.
Pour éviter ce genre de problèmes, il faudrait que l'utilisateur choisisse lui-même les seuils de concordance. Or, cela semble peu évident pour un non-informaticien de devoir choisir des valeurs correspondant à des concepts rattachés à une méthode mathématique. Cette approximation pose déjà le problème de toutes les méthodes de classement utilisées. D'autres méthodes possèdent d'autres approximations.
Le résultat final que nous obtenons est donc soumis à l'utilisateur, afin qu'il juge lui-même si le classement lui convient. Dans le pire des cas, même si le classement n'est pas le meilleur (il faut également ajouter que "meilleur" est une notion assez vague), on obtient tout de même une liste de véhicules qui correspondent aux critères et qu'on aurait eu du mal à rechercher sans le logiciel, et ce au-delà du classement.

7. Les systèmes d'information d'aide à la décision

7. 1 GENERALITES

Ce chapitre apporte une vue très générale sur les Systèmes d'Information d'Aide à la Décision. Il prétend y donner non pas une définition stricte, car ce n'est pas l'objet de ce document, mais une explication visant à montrer l'intérêt de tels systèmes qui se développent de plus en plus et la manière d'apporter une aide dans de nombreux domaines où la décision fait partie de la vie de l'organisation. Le lecteur intéressé se réfèrera à [BODART-cours] et [PIGNEUR].

A la fin du chapitre, nous montrerons comment ***Sensitive Car*** répond à cet objectif dans un domaine particulier, somme toute assez restreint, qu'est la vente d'un véhicule et comment il pourrait être étendu en vue de s'intégrer dans une vue plus générale de gestion d'une organisation (ici l'organisation visée est un constructeur de voitures) où le **processus décisionnel** s'insère dans des **procédures de gestion** telles que les procédures de gestion budgétaire, de choix d'investissement, de politiques financières, de gestion de la production, d'examen mensuel des situations de gestion.

Comme exemple d'insertion, citons la réaction à une baisse de prix des produits concurrents avec l'examen mensuel de la situation commerciale.

Un Système d'Information d'Aide à la Décision (S.I.A.D.) est un Système d'Information (S.I.) construit à l'aide de technologies d'information et destiné à supporter les activités décisionnelles des gestionnaires.

Un S.I.A.D. est un S.I.:

Il vérifie donc la définition générale d'un S.I.: construction formée d'ensembles d'informations, de traitements, de règles d'organisation et de ressources humaines et techniques. Il possède cependant des propriétés spécifiques liées aux particularités du système cible auquel il se rapporte.

Les activités décisionnelles des gestionnaires:

Les activités des gestionnaires liées directement aux problèmes de décision sont de natures diverses sans qu'il soit possible a priori de définir une importance relative parmi celles-ci. Parmi ces natures, citons la perception de problèmes, le diagnostic, la recherche d'informations, leur analyse, les échanges d'information, la prévision, la négociation, la construction d'une solution, le choix d'une alternative, etc.

Un S.I.A.D. est un support:

Un S.I.A.D. contribue à la réalisation d'activités de gestion, mais n'a pas pour but d'automatiser ces activités. Il offre des fonctions de support ou d'aide qui seront exploitées interactivement à l'initiative, en général, du gestionnaire (dans le cas du .
Sensitive Car le vendeur).

Un S.I.A.D. est construit à l'aide de technologies d'information:

Parmi les technologies, citons les Systèmes de Gestion de Bases de Données (S.G.B.D.), les S.G.B.D. multimédia, les hypermédia, les systèmes documentaires, les technologies de la communication, les tableurs, les 4GL, les logiciels graphiques, les logiciels de recherche opérationnelle et de statistique.

7. 2 A PROPOS DE Sensitive Car

Il est aisé de remarquer les similitudes de ***Sensitive Car*** avec les différentes définitions que nous venons de donner. Passons-les en revue une par une.

Sensitive Car *est un S.I.:*

C'est bien une construction formée d'ensembles d'informations, de traitements, de règles d'organisation et de ressources humaines et techniques. Il possède également des propriétés spécifiques liées aux particularités du système cible auquel il se rapporte, à savoir la vente de véhicules.

Activités décisionnelles des gestionnaires:

En ce qui concerne ***Sensitive Car,*** le gestionnaire est le vendeur pour la version Professional et l'utilisateur en général pour la version User Guide. En effet, nous pouvons considérer que le problème qui se pose au vendeur est non seulement la vente d'un véhicule, mais encore faut-il que ce véhicule réponde aux exigences du client potentiel.
La recherche d'informations est constituée par la recherche de la 'meilleure' voiture qui correspond 'le mieux possible' à ces exigences.
Tout ce qui a été dit dans le paragraphe précédent à propos des activités décisionnelles du gestionnaires est donc facilement transposable à notre cas particulier.

Sensitive Car *est un support:*

Sensitive Car contribue à la réalisation d'activités de gestion, dans notre cas l'activité est la vente, mais n'a pas pour but d'automatiser ces activités. En effet, le vendeur joue le rôle de médiateur entre les exigences d'un client et les propositions faites par notre logiciel. Un exemple est le cas des remises dont notre logiciel ne se charge pas, mais qu'il prévoit néanmoins lors des propositions. Il offre des fonctions de support ou d'aide qui seront exploitées interactivement à l'initiative, en général, du gestionnaire (dans le cas du ***Sensitive Car Professional*** le vendeur). Ces fonctions d'aide sont les propositions des différents critères que le vendeur peut imposer au logiciel.

Sensitive Car *est construit à l'aide de technologies d'information:*

Nous retrouvons bien un S.G.B.D. multimédia, les technologies de la communication, les logiciels graphiques, les logiciels de recherche opérationnelle
(analyse multicritères) et de statistique.

7. 3 Mais pourquoi aurait-on besoin de Sensitive Car ?

Vous vous serez certainement posé la question. Et si vous ne l'avez pas fait, moi, concepteur du logiciel, j'ai dû me la poser avant de construire ce logiciel, car comme un constructeur automobile ne se lance pas dans la conception et la construction d'un nouveau modèle sans une analyse profonde du bien fondé de cette réalisation, nous aussi, informaticiens, avons pareille exigence.

Je suis conscient que ***Sensitive Car*** est un **défi**, car à ce jour, il n'existe aucun autre logiciel égal à celui-ci. Le jour où cela arrivera, on n'achètera plus ***Sensitive Car*** afin d'innover en matière de vente, mais on l'achètera parcequ'il faudra faire comme tout le monde, dans la but avoué de ne pas perdre des parts de marché si difficilement acquises. En effet, il ne viendrait à l'idée d'aucune banque de ne pas proposer de facilités de paiement telles que les cartes magnétiques ou les chèques.
Une banque qui voudrait émerger sans ces facilités serait en faillite avant même de commencer son activité. Son seul moyen d'exister face à la concurrence serait non seulement de proposer ces formules (donc sans innover), mais pour prendre le dessus, elle devrait également proposer quelque chose d'innovatif, ce que j'appelle un défi. Dans ce cas, elle profiterait d'un avantage pendant un certain temps et ensuite les autres banques devraient se mettre au pas afin de combler le retard. Cependant, le premier a avoir osé, reste dans la mémoire de tous comme étant le premier. Cela peut être un avantage non négligeable dans certains cas. C'est une loi des marchés concurrentiels que nous observons tous les jours. Et si l'exemple de la banque paraît appartenir à un autre univers, prenons un exemple dans le domaine automobile.
Pour cela, référons-nous aux barres de protection latérale. Le premier à en avoir proposé a été le constructeur suédois Volvo. Pourrait-on citer une seule marque, un seul modèle, à ce jour et dans le monde entier qui ne soit pas équipé d'un tel système?
La réponse est non bien entendu, mais malgré cela,Volvo reste le premier aux yeux de tous. De plus, si nous considérons les stratégies actuelles de vente de véhicules, nous constatons que celles-ci se limitent à des 'journées portes ouvertes', à des concours pouvant faire gagner des voyages de rêve ou encore à des distributions de cadeaux ou autres avantages financiers lors de l'achat réel d'une voiture.

Cependant, hormis le dernier point cité, il n'y a aucune stratégie qui vise la **réelle satisfaction du client.** Ceci est un paradoxe, car de tous les sondages, il ressort clairement que l'accent est mis sur ce point crucial.
Ceci est dû au fait qu'***un client content et satisfait est un client qui revient.***
Toutes les personnes impliquées dans le marketing le savent. Et pourtant on continue à vendre comme on le faisait il y a trente ans.
De plus, ayant effectué un recensement des personnes présentes lors des dites journées portes ouvertes, j'ai remarqué qu'il y a à peine plus de monde que lors d'une journée normale, sans compter que la plupart des personnes présentes étaient là uniquement pour emporter le cadeau proposé. Alors est-il réellement besoin d'investir des sommes importantes pour un retour de bénéfice qui n'est pas du tout assuré, alors qu'avec ***Sensitive Car*** on ne paie qu'une fois pour un résultat qui fera ses preuves?
Il est à noter qu'un vendeur nous a fait remarquer la chose suivante afin de justifier ces journées portes ouvertes: "Même si ces journées ne rapportent pas, il n'en reste pas moins qu'un véhicule vendu est une publicité de plus pour la marque sur la route. C'est

mathématique". Or, ce brave vendeur oublie une chose importante: étant donné que toutes les marques font des journées portes ouvertes, elles vendent toutes un pourcentage (supérieur?) proportionnel aux ventes opérées durant les autres jours. Donc ce qui se passe c'est que le pourcentage de ventes de chaque marque est consolidé et non augmenté. Les journées portes ouvertes ne servent plus à augmenter considérablement le nombre de ventes, mais uniquement à faire comme toutes les autres marques, afin justement de ne pas diminuer ce nombre. Cela a déjà été expliqué dans l'introduction.
Un autre point qui consolide ce qui a été dit ci-avant est le besoin exprimé par les clients d'avoir **"quelque chose"** qui puisse leur donner un maximum pour une dépense minimale. En effet, au même titre que l'argent est le nerf de la guerre, pour le client il s'agit également d'un problème de budget. Nous l'avons déjà précisé dans un autre chapitre, on n'achète plus une voiture pour sa ligne ou sa beauté, mais pour ses réelles fonctionnalités, son prix et ce qu'elle offre comme agrément. Ceci ressort directement d'un sondage effectué auprès de vendeurs.

Pour terminer sur la question que nous avons posé, considérons un instant le cycle de fabrication d'un produit 'high-tech', comme une chaîne hi-fi ou une voiture.
Qu'en est-il?

Nous remarquons qu'à chaque stade du cycle, la composante informatique-robotique est énorme, et nul doute quant à son extension future. Le motif en est simple. Au début de l'informatique, on a très vite remarqué que l'on pouvait se décharger de toute tâche fastidieuse et routinière pour optimiser certains départements (cfr. les chaînes de montage - l'ère du Fordisme). Cet exemple a été suivi par tous (une fois de plus, le cycle innovation - prolifération cité plus haut est de mise).

Personne ne contesterait actuellement la place de l'informatique au niveau du **développement** (C.A.O. : conception assistée par ordinateur, aucun ingénieur ne travaille sans cet outil devenu indispensable), au niveau de la **production** (il s'agit essentiellement de robotique ou informatique industrielle), au niveau de la **distribution** (un exemple simple pour clarifier ce point est celui de la gestion automatisée des clients ou des pièces de rechange) et enfin de la **maintenance** (notons que chez BMW, les voitures sont équipées d'un ordinateur de bord qui enregistre toutes les anomalies remarquées lorsque la voiture roule. Lors d'une visite à l'officine pour un entretien, le mécanicien connecte un ordinateur au véhicule et les résultats sont imprimés. Le mécanicien sait directement ce qu'il doit effectuer comme réparation ou vérification. D'où un gain de temps considérable pour le client, qui est totalement satisfait (du moins on l'espère) et ne regrettera pas son choix. Nous sommes persuadés que ce système sera banalisé d'ici quelques années, mais BMW reste le premier.

Vous pensez que j'ai oublié de parler du niveau **vente?** Rassurez-vous, il n'en est rien. Le problème c'est qu'il n'y a rien de prévu à ce stade de la chaîne. Est-ce pour cela un problème? Rassurez-vous encore. ***Sensitive Car*** a été créé dans le but de combler cette lacune. Aujourd'hui ce logiciel semble être un défi audacieux. Demain, il s'intégrera dans une nouvelle manière de gérer les ventes, les processus de conception, de fabrication, de livraison, de maintenance et de gestion globalisée de cette chaîne commerciale, car comme vous le pensez certainement, ***Sensitive Car*** ne représente qu'un module, en d'autres mots qu'une application particulière de ce qui pourrait être un nouveau mode automatisé et complètement intégré de services rendus aux 'gestionnaires', à quelque niveau que ce soit.

7. 4 La place des SIAD dans les organisations

Tel qu'il a été défini, un SIAD non seulement justifie sa place dans une organisation, mais de plus il permet de modifier l'environnement de celle-ci. En effet, ces systèmes ne se développent pas en fonction de l'intérêt purement théorique de certains informaticiens, mais sont le désir de beaucoup de gestionnaires qui voient en ces logiciels de vrais outils leur permettant d'augmenter l'efficacité de leur gestion. Cela résulte du fait que ces SIAD intègrent des outils performants de communication, (souvent appelés mail électroniques), de gestion des données, et même plus que cela, car certains logiciels permettent de comparer des données entre elles, de simuler des situations, d'anticiper des résultats, de rassembler et de récolter nombre de données automatiquement, alors qu'auparavant il était impossible de les obtenir sans accroître la charge de travail des employés. Ces SIAD peuvent également organiser des meetings réunissant des personnes en des lieux différents ou à des moments différents et permettre par la même occasion de travailler sur un même document à distance, ou encore faire fonction de secrétaire infatigable. Un autre intérêt de ces logiciels est de permettre un stockage des documents par la voie informatique. Il n'est plus besoin de garder le document physique. De plus, des systèmes de classement automatique existent et permettent d'indexer et retrouver ces documents, d'avoir une copie, de travailler sur un document en y ajoutant des objets d'un autre type que du texte (photo, son, film,...).

Comme on peut le voir dans ce paragraphe, les possibilités des SIAD sont réelles et leur avenir est prometteur. En y réfléchissant d'un peu plus près, il est aisé de constater que les possibilités d'une organisation sont décuplées. En effet, les distances sont raccourcies et actuellement il est possible, grâce à ces systèmes, de considérer une société ayant ses sièges en des endroits distants de plusieurs milliers de kilomètres comme une seule société, puisque le transfert de documents est immédiat, via des réseaux de communication intégrés à ces systèmes. De plus, une réunion peut se faire à distance et si un membre ne peut se déplacer, il n'y a pas de problème pour le faire participer à la réunion. Il en résulte également un gain d'argent, puisque on évite ainsi de payer un voyage en avion pour quelques heures de réunion. Il s'en suivra également un gain de productivité. Quoiqu'il en soit, on ne peut plus ignorer l'impact des technologies de l'information sur les modes organisationnels d'une entreprise. Il s'en suit une ré-ingéniérie des processus décisionnels, de contrôle, de coordination et de flux des informations au sein de l'entreprise. Une chose est certaine: les limites des organisations sont déplacées au-delà des limites physiques fixées par les bâtiments de l'entreprise. ***Sensitive Car*** fait partie de ces systèmes, mais ne représente qu'un module et il pourrait devenir le noyau du SIAD d'un constructeur. *Sensitive Car Professional* a été développé dans le but de construire un SIAD d'une nouvelle génération, afin que l'informatique ne semble plus un fardeau, mais un outil efficace et utile dans le métier de la vente automobile, en garantissant une transparence totale des moyens techniques mis en oeuvre pour arriver à un résultat efficace. Le désir le plus grand est de relever le défi de proposer un service résolument tourné vers la clientèle!
Sensitive Car User Guide quant à lui a été construit dans le but de proposer ce service à la clientèle, malgré le constructeur. Etant donné que ce logiciel pourra être acquis par n'importe qui, on ne pourra plus vendre n'importe quoi, puisque le client pourra faire lui-même une comparaison avec la concurrence.

8. CONCLUSION

Nous sommes partis d'une idée très vague qu'était l'achat d'un véhicule et le moyen d'aider un acheteur potentiel à effectuer le "meilleur" achat. Au fur et à mesure que nous avancions dans le développement du logiciel, nous avons remarqué plusieurs problèmes liés au type de logiciel que nous développions, un système d'aide à la décision. De la difficulté que nous avons eu à définir rationnellement ce qu'était une décision, limitée ici au domaine automobile, nous avons pris certaines décisions qui ont eu des répercussions sur le logiciel et la manière de le concevoir.

Cela a comme conséquences que les deux versions développées considèrent la décision d'acheter un véhicule comme étant un mélange de définitions de critères et de comparaisons de véhicules suivant d'autres critères. Cela est dû à l'impossibilité de cerner avec exactitude l'activité de décision. Nous en avons déjà parlé auparavant dans d'autres chapitres, mais il est bon de le rappeler dans la conclusion. En effet, à chaque véhicule proposé, nous pouvons considérer qu'il y a une marge d'erreur dans la décision qui a été prise. Est-ce pour cela que la décision prise est mauvaise? Non, car de toute manière nous ne savons pas définir ce qu'est la bonne décision. Si nous le savions, nous aurions automatisé la manière d'y arriver. De plus, le logiciel ne fournit pas de résultats hors attente (j'entends par là que les choix offerts ne sortent pas d'un intervalle de valeurs plus ou moins attendues. On ne se voit pas proposer une voiture à 300.000 francs lorsque le budget du client est de 1.000.000 de francs, cela est déjà un bon signe). Il faut également considérer que lorsque le logiciel est utilisé par quelqu'un qui veut réellement acheter une voiture, il finit toujours par proposer une solution acceptable (finalement, à moins qu'on ait déjà le véhicule en tête et l'argent nécéssaire pour l'acheter, on finit toujours par faire des compromis. Tout le monde rêve de rouler avec une Rolls Royce et finit par acheter une voiture moins chère. Il faut rester rationnel).

Si le logiciel est utilisé dans l'espoir de se voir proposer l'impossible, la déception sera grande. A ce propos, un vendeur Fiat m'a raconté qu'un client était amoureux de la Fiat Croma Turbo Diesel, mais qu'en envisageant toutes les possibilités possibles et imaginables, ce client n'avait pas l'argent nécéssaire afin de concrétiser son rêve, pas même avec un financement à long terme. Il est inutile de croire que le logiciel lui aurait trouvé une solution. La froideur de l'ordinateur aurait probablement envoyé un message comme *"Il existe une ou des version(s) correspondant aux critères, mais dont le prix est supérieur"*. Cela veut dire qu'il faut utiliser le logiciel dans le but d'obtenir une aide, pas un faux espoir. Ceci fixe en quelques sortes les limites de *Sensitive Car*, car il ne permet pas, face à un vendeur d'essayer d'obtenir un véhicule d'une catégorie supérieure, moyennant une petite hausse de la somme qu'on est disposé à payer. Cela est en partie résolu par l'intervalle pris afin de réaliser la recherche pour un budget donné. D'autre part la composante rêve est difficilement informatisable.

Finalement, nous pouvons considérer que nous avons développé deux versions d'un même logiciel et qui, pour résumer le travail, a les caractéristiques suivantes.

- **Logiciel flexible, intégrable à d'autres applications, et extensible.**
  Flexible, car il permett de rester vague dans les choix qui sont faits par le client, tout en restreignant les choix possibles, en évitant au maximum l'erreur humaine (dans l'encodage de valeurs, par exemple).
  Intégrable: *Sensitive Car* n'impose aucun choix de matériel ou logiciel particuliers qui influenceraient l'avenir informatique d'un concessionnaire (version Professional). Il s'adapte à un environnement existant.

Extensible: Il supportera facilement de nouvelles fonctionnalités, sans pour autant recommencer tout le développement, du moins quant à l'interface.

- **Simple d'utilisation.** Il a été développé en tenant compte de critères ergonomiques réels, dans la plupart des cas négligés par les informaticiens.

- **Polyglote**: français, néerlandais, anglais, ou autres.

- **Outil qui formalise les processus décisionnels mis en oeuvre lors de l'achat d'un véhicule par un client**, en ce sens qu'il aide celui-ci à restreindre le nombre de choix possibles qui s'offrent à lui, en proposant des éventualités considérées comme des critères.

- **Il permet de diriger un acheteur potentiel, mais qui n'aurait qu'une idée assez vague de ce qu'il désire acheter**. *Sensitive Car* tente, avec l'aide du client, de formaliser ses désirs ou ses espoirs en montrant ce qui est possible, en fonction des critères fournis. On peut donner un ordre d'importance aux critères et demander que certaines options fassent partie de l'équipement du véhicule, pour le prix demandé.

- **Obtention d'un classement avec ex aequo des véhicules choisis par le logiciel, ainsi qu'une explication du classement.**

- **Il aide le vendeur**, car, le travail à fournir par chacun des vendeurs devient vite difficile, le cerveau humain ne pouvant gérer un grand nombre d'informations en quelques minutes passées avec un client. Actuellement, le nombre de combinaisons de véhicules disponibles croît de manière exponentielle, ce qui implique une simplification des procédures de ventes.

- **Il calcule le financement** (version Professional).

Les développements futurs du logiciel sont prévus. En effet, ce qui a été implémenté jusquà présent ne représente qu'un prototype et demande donc d'être raffiné. Ce sera le but des versions successives. Ces améliorations regardent notamment certaines fonctionnalités, ainsi qu'un raffinement du module *Classement* et du module *Sensitive Query*, les problèmes liés à ces modules ont déjà été discuté. Il sera envisagé d'y apporter une solution

A l'avenir, il sera entre autre possible de **comparer un véhicule** donné (une version) **par rapport à ses concurrents directs**. Par exemple, si on choisit la Renault Laguna 1.8 RT équipé avec double-airbag, climatiseur, intérieur cuir, peinture métallisée et auto-radio, elle coûtera disons 950.000 francs. Il sera dès lors possible de comparer cette voiture avec ses concurrentes directes (Peugeot 406 1.8, Citroën Xantia 1.8i, Alfa 155 1.7 et 1.8, Toyota Carina 1.6 et bien d'autres). Cela demandera bien sûr de prévoir une association "concurrence" qui est récursive sur l'entité Version. Une version a au moins une concurrente (la voiture parfaite n'existe pas, du moins le suppose-t-on) et une version est concurrente d'au moins une autre version (pour le même motif).

Comparatifs basés sur des rapports de professionnels de l'automobile. Il s'agirait en quelque sorte d'articles hypertextes qu'il serait possible de consulter dans le logiciel et ayant été rédigés pas des journalistes spécialistes de l'automobile (ex: Marcel Pirotte du journal Vlan).

La cote de l'occasion. Nous n'en avons pas vraiment parlé, mais il sera possible d'ajouter comme critère que la dévaluation du véhicule après x années ne soit pas supérieure à y% (pour autant que ce véhicule existe depuis x années évidemment). Cela peut également être un critère d'achat.

Proposer des graphiques (statistiques). Il sera également possible de fournir des graphiques avec différentes données comparant des véhicules (courbes de consommation, de bruit, vitesses maximum, minimum,...). Nous pourrons comparer le véhicule par rapport à ses concurrentes (voir ci-dessus, mais de manière graphique), ainsi que par rapport au maximum et minimum sur le marché ou encore par rapport à la catégorie envisagée...

L'**estimation du calcul du financement** dans la version User Guide sera ajoutée.

Le service à domicile. Même si le client ne peut se déplacer durant la journée pour des raisons professionnelles, il aura sans doute la possibilité d'accéder à *Sensitive Car* via un PC. Exemple : **Home-Bank®** et accès au réseau **Internet®**. A ce titre, notons que le logiciel du Moniteur de l'Automobile® semble aller dans cette voie.

Commandes d'un véhicule depuis le domicile. Cela se fait déjà dans d'autres domaines et surtout dans d'autres pays.

Show-room: une version *light* (simplifiée) sera certainement développée, afin qu'elle puisse servir de borne d'information dans les show-rooms.

Le suivi des clients. Il sera possible de garder la trace des clients qui sont passés au show-room et qui reviendraient pour rediscuter du véhicule qui avait précédemment été choisi. Celui-ci, avec toutes ses options sera mémorisé et il suffira de demander le nom du client. D'où un gain de temps et un service personnalisé (version Professional).

Statistiques sans charge de travail supplémentaire de la part du vendeur, car celui-ci n'aura pas à s'occuper de certainess statistiques et autres renseignements qu'il sera possible de demander à *Sensitive Car*. En effet, si actuellement on désirait avoir ces informations, il faudrait imposer au vendeur de remplir des fiches. Cela devrait se faire au moment où le vendeur parle avec le client. Par conséquent, ce dernier devrait être également impliqué dans ce processus. Cela deviendrait presque un questionnaire. D'un autre côté, on pourrait demander au vendeur de remplir ces fiches en fin de journée, mais je pense que personne n'est disposé à travailler quelques heures de plus tous les jours, sans compensation. Or, avec notre logiciel, il n'y a pas besoin de compensation. Tout se fera automatiquement. On voit l'impossibilité de disposer de ces statistiques, dans l'état actuel des choses, mais grâce à *Sensitive Car,* cela sera rendu possible.

Possibilité d'ajouter d'autres langues. En effet, le logiciel a été développé de telle sorte que l'ajout d'une langue ne pose aucun problème (même si nous n'en avons pas parlé, cet aspect semblait moins intéressant).

Personalisation des écrans. Il sera effectivement possible de changer les couleurs, les polices de caractères, définir des couleurs personalisées, des polices personalisées,... Ce travail a été entamé pour la version User Guide, mais n'a pas encore été entièrement complété, c'est pour cela que nous n'en avons pas parlé. Nous avons choisi de traiter d'un noyau représentant bien le problème dans le cas d'un S.I.A.D. Ces aspects cités appartiennent plutôt à des problèmes ergonomiques.

Ceci termine donc le travail que nous avons entamé. Le but poursuivi semble atteint et nous pouvons à présent dire un petit mot des autres logiciels présents sur le marché. Il est un fait certain qu'aucun autre logiciel ne présente les caractéristiques de *Sensitive Car*. Le module *Sensitive Query*, bien qu'incomplet, est original. En effet, aucun autre logiciel ne permet d'interroger la base de données d'une manière aussi flexible et simple. D'autre part, le module *Classement*, représente une première tentative de vouloir approcher le problème du *meilleur* véhicule par rapport à ce que l'on désire. Loin d'être parfait, il permet néanmoins de laisser les autre logiciels sur place, puisqu'à nouveau, ils ne prévoient que la recherche en fonction de critères et pas toujours avec des résultats...

Comme on peut le voir, les développements futurs sont certainement plus nombreux que ceux que nous avons fait jusqu'à présent, sans toutefois sous-estimer ceux-ci. Les fondations de ce qui peut devenir une référence en matière de S.I.A.D. dans le domaine de la vente automobile sont là, il suffit d'assembler d'autres modules indépendants (comme le sont *Sensitive Query* et *Classement*) et de les ajouter au logiciel. Le tout est de travailler comme nous l'avons fait, c'est-à-dire ne rien laisser au hasard, mais prévoir de manière rigoureuse l'interface qui supportera la fonctionnalité souhaitée et l'intégration dans le logiciel, en suivant les standards que nous nous sommes fixés, se fera naturellement.

BIBLIOGRAPHIE

[BODART89]
"Conception assistée des systèmes d'information Méthode - Modèles - Outils "
François Bodart et Yves Pigneur

[BODART94]
"Dimensions clé pour une méthodologie de développement d'applications interactives" rédigé par François BODART, Anne-Marie HENNEBERT, Jean-Marie LEHEUREUX, Isabelle PROVOT, Jean VANDERDONCKT, Giovanni ZUCHINETTI.
Papier présenté lors du International EurographicsWorkshop on Design, Specification and Verification of Interactive Systems, Bocca di Magra (La Spezia), 8-10 juin 1994.

[BODART- cours]
"Cours Systèmes d'aide à la décision de 2° licence informatique FUNDP"
François Bodart

[FICHEFET92]
"Cours Théorie des graphes et compléments de mathématique FUNDP 1°licence"
Jean Fichefet

[FICHEFET95]
"Cours d'Analyse multicitère FUNDP 2°licence"
Jean Fichefet

[HAINAUT86]
"Cours de Conception des bases de données - 1° licence et maîtrise informatique - FUNDP"
Jean-Luc Hainaut

[HAINAUT93]
"Cours de Conception des bases de données matière approfondie - 2° licence et maîtrise informatique - FUNDP"
Jean-Luc Hainaut

[IHM95]
"Cours Interface Homme-Machine de 2° licence informatique FUNDP"
François Bodart

[PYGNEUR] *"Cours Systèmes d'aide à la décision de 2° licence informatique FUNDP"*
Yves Pigneur

[ROY] *"Aide multicritère à la décision: méthodes et cas"* Economica - Paris - 1993
B. Roy et D. Bouyssou

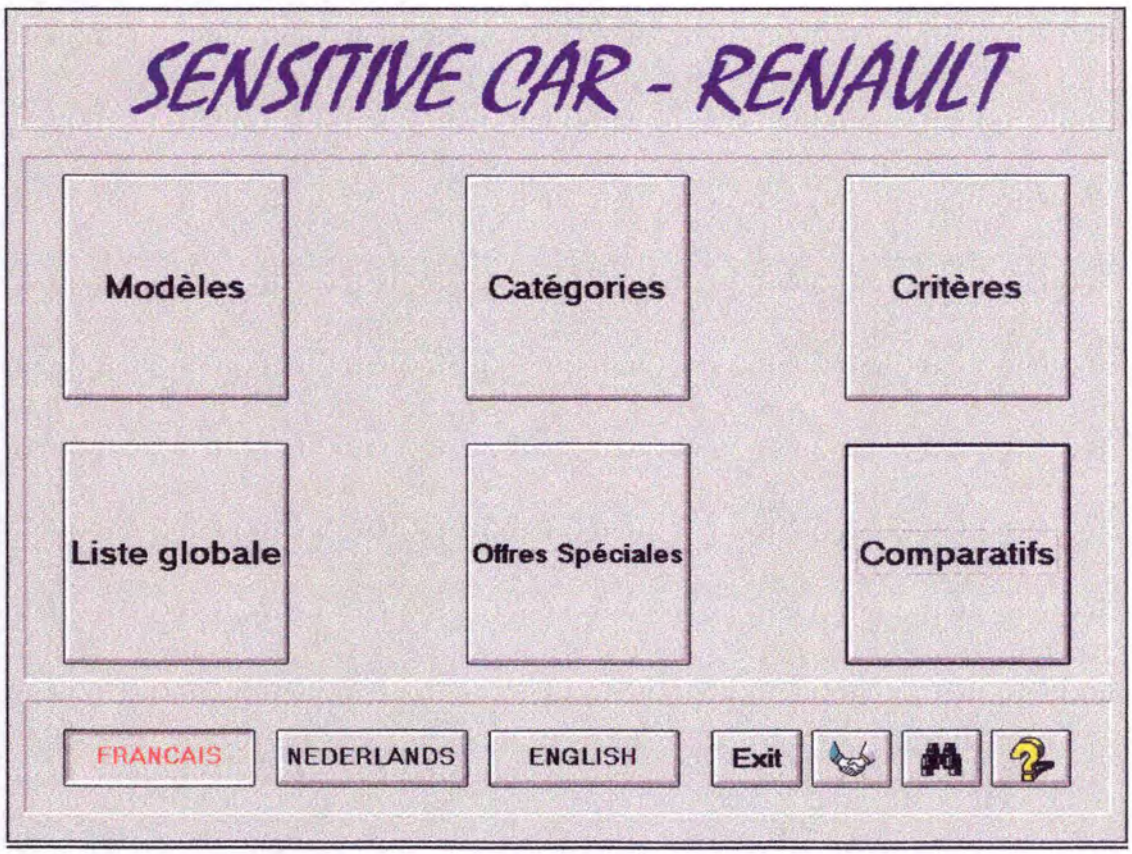
[SACRE92]
"Une description orientée objet des objets interactifs abstraits utilisés en Interface Homme-Machine" rédigé par Benoît SACRE, Isabelle PROVOT-SACRE, Jean VANDERDONCKT. Projet TRIDENT, FUNDP, 7 octobre 1992, rapport de

recherche IHM/Ergo/10

[VANDERDONCKT92]
"Corpus ergonomique minimal des applications de gestion" rédigé par Jean VANDERDONCKT. Projet TRIDENT, FUNDP, octobre 1992, rapport de recherche.

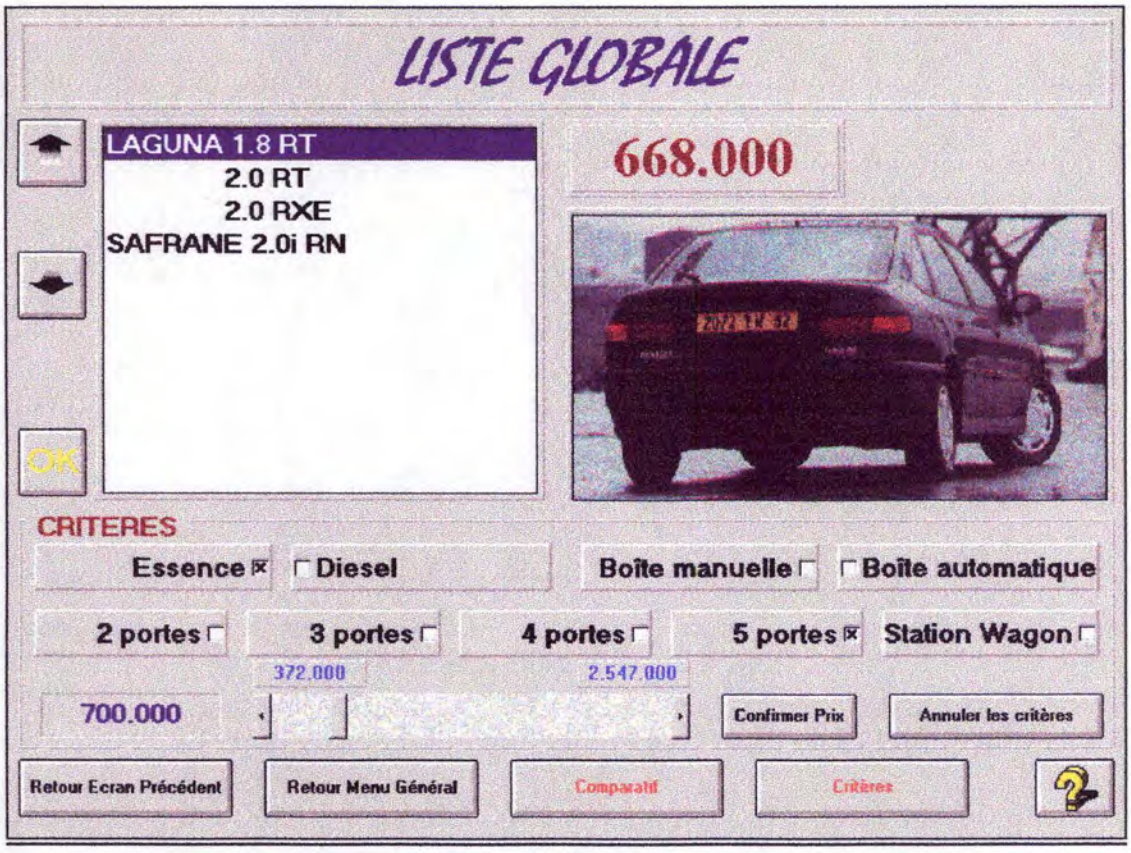
[VINCK89]
L'aide Multicritère à la décision"
Philippe Vincke

1) *Menu général*

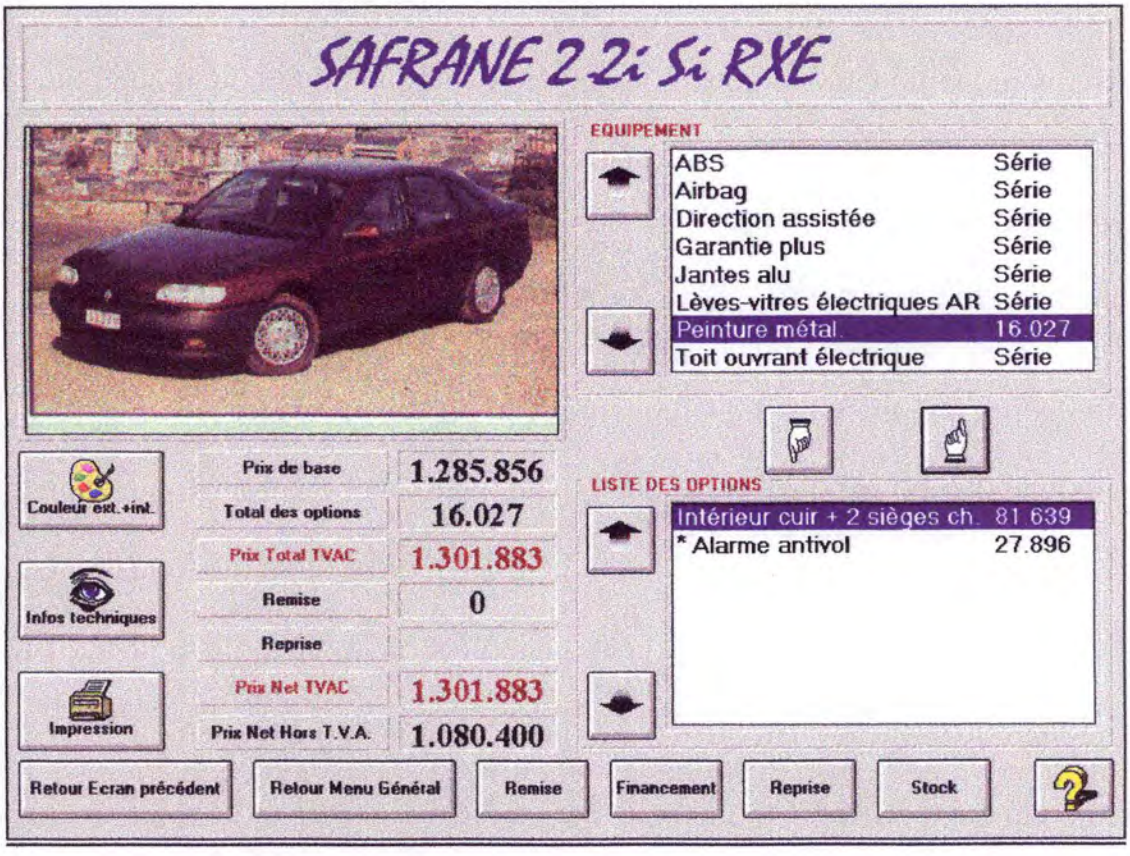


2) *Choix par modèles*

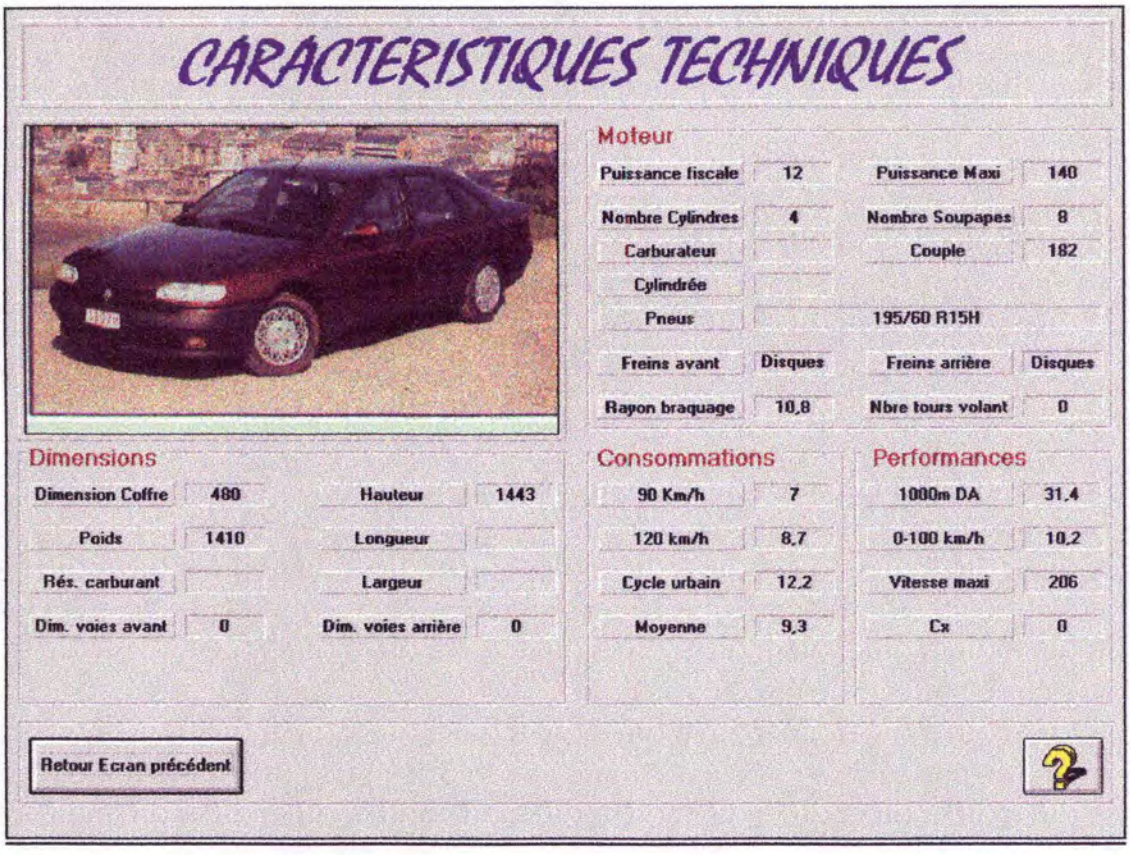
3) *Spécification de la version désirée*



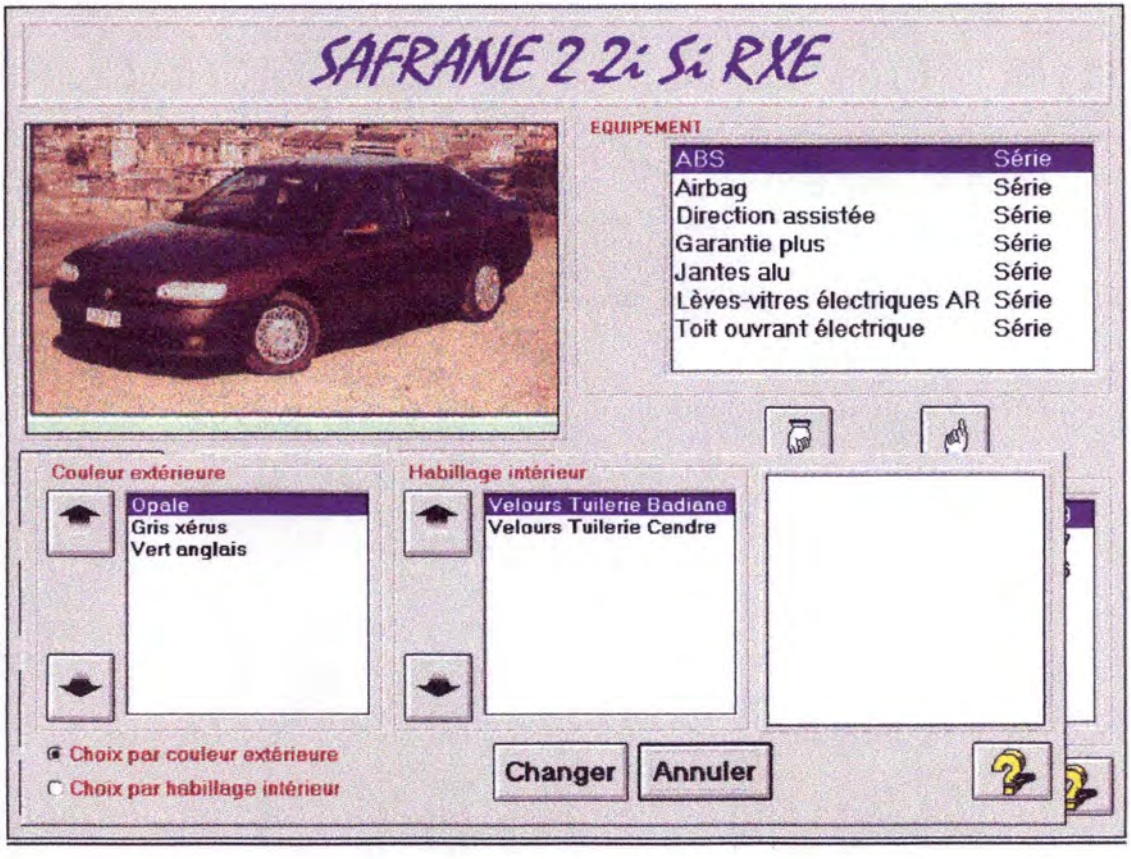
4) *Choix de l'équipement*



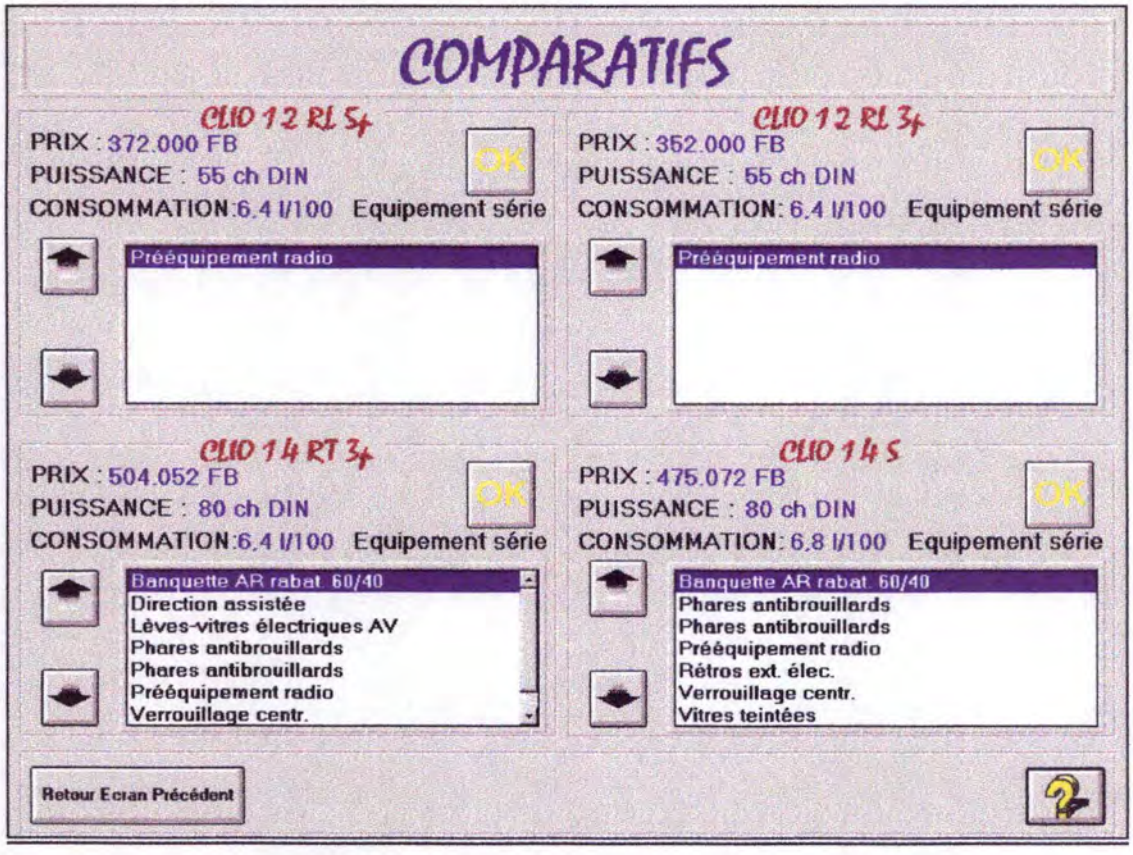
5) *Caractéristiques techniques de la version désirée*



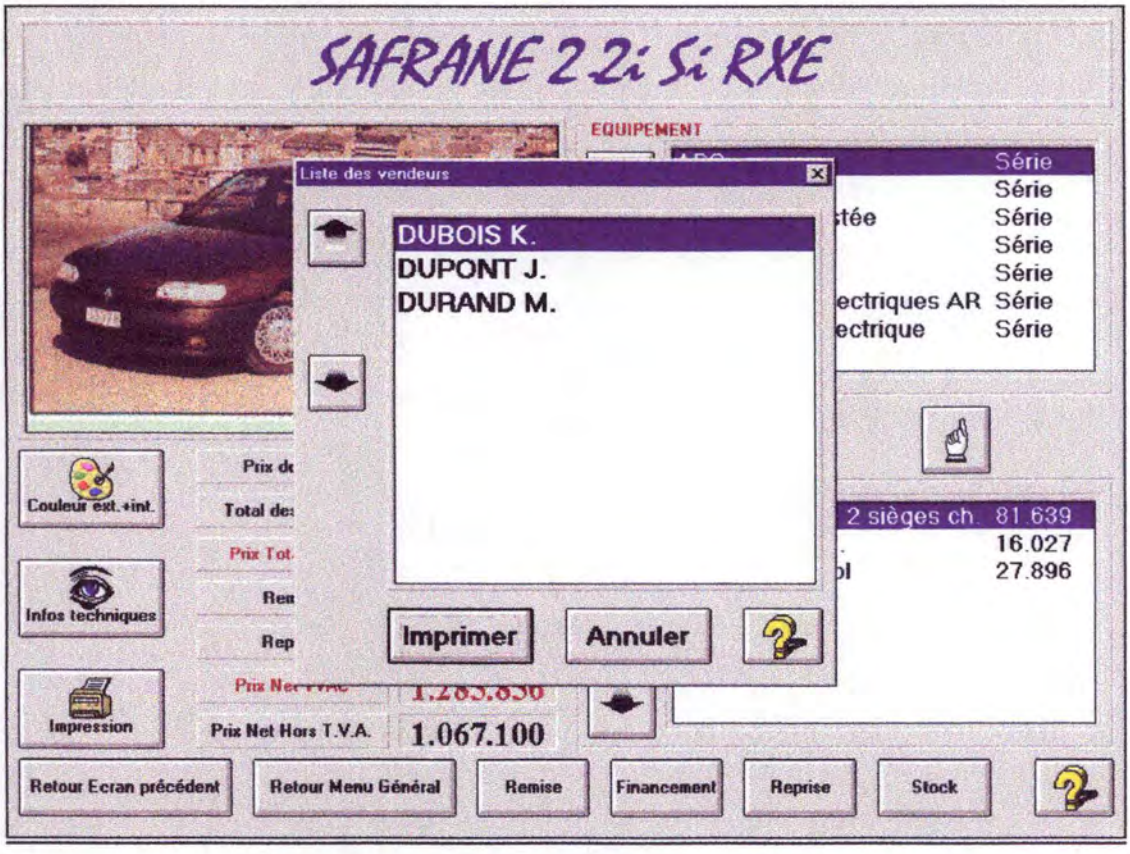
6) *Choix de l'habillage intérieur et de la couleur extérieure*



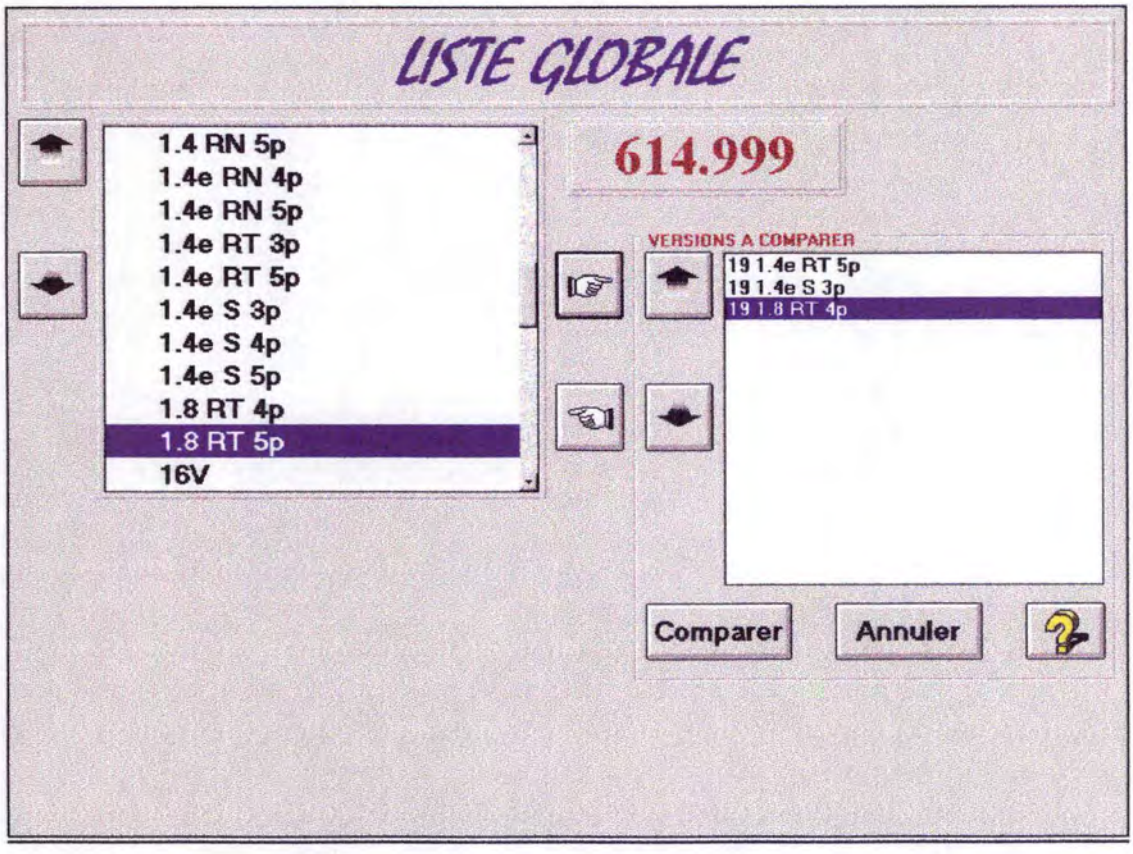
7) *Comparatif de véhicules*



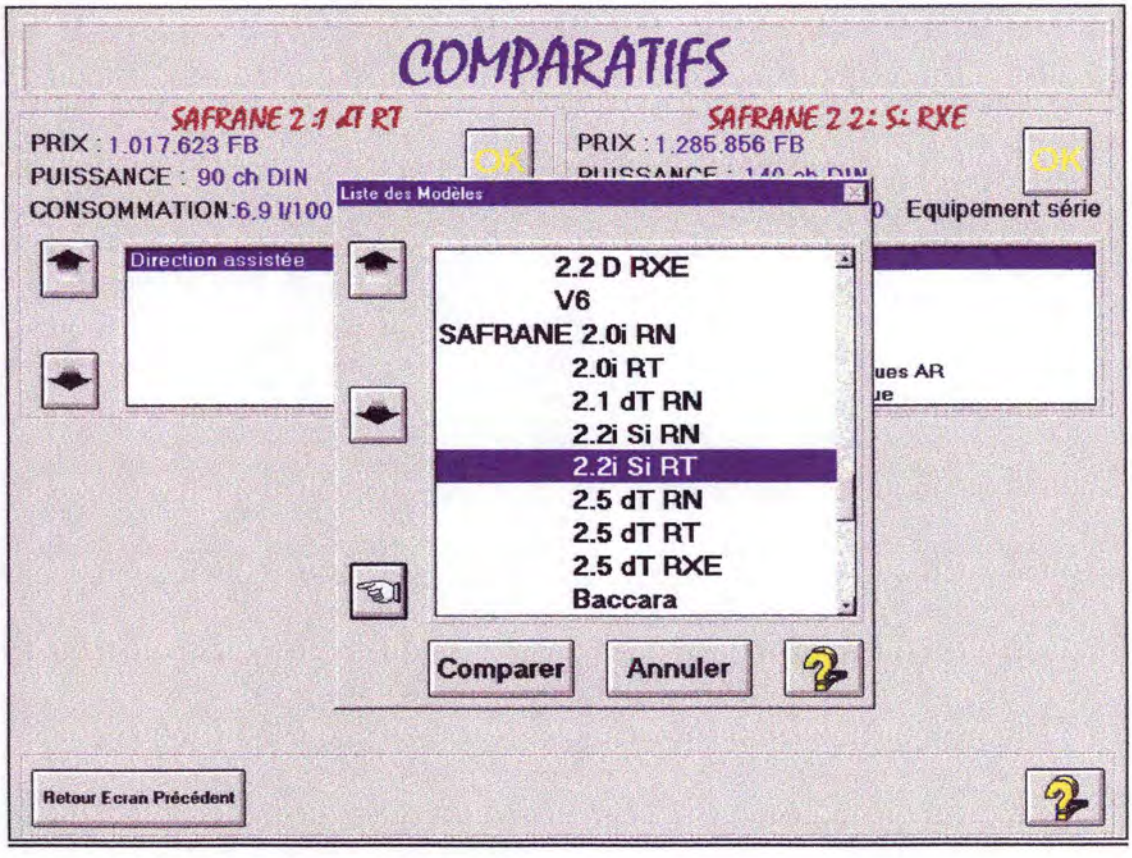
8) *Choix du vendeur effectuant la vente*



9) *Choix des véhicules à comparer*



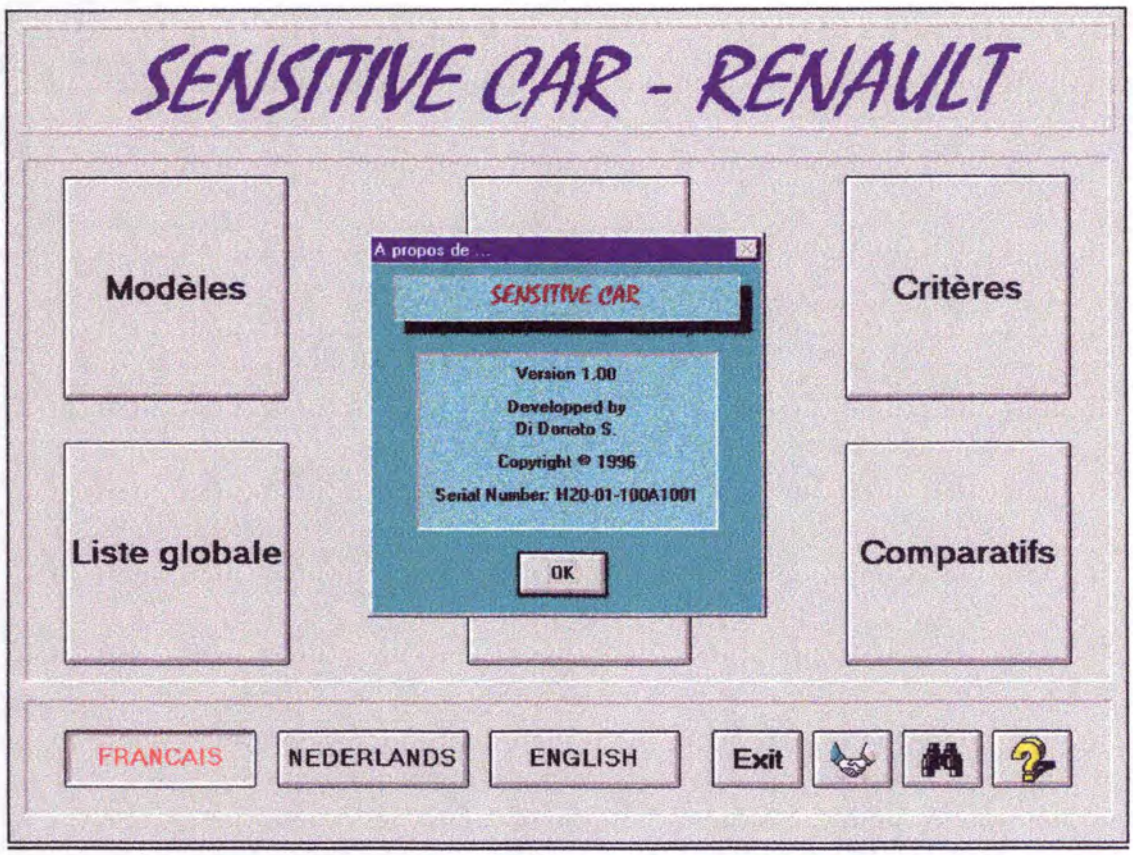
10) *Choix de véhicules à comparer (autre méthode)*



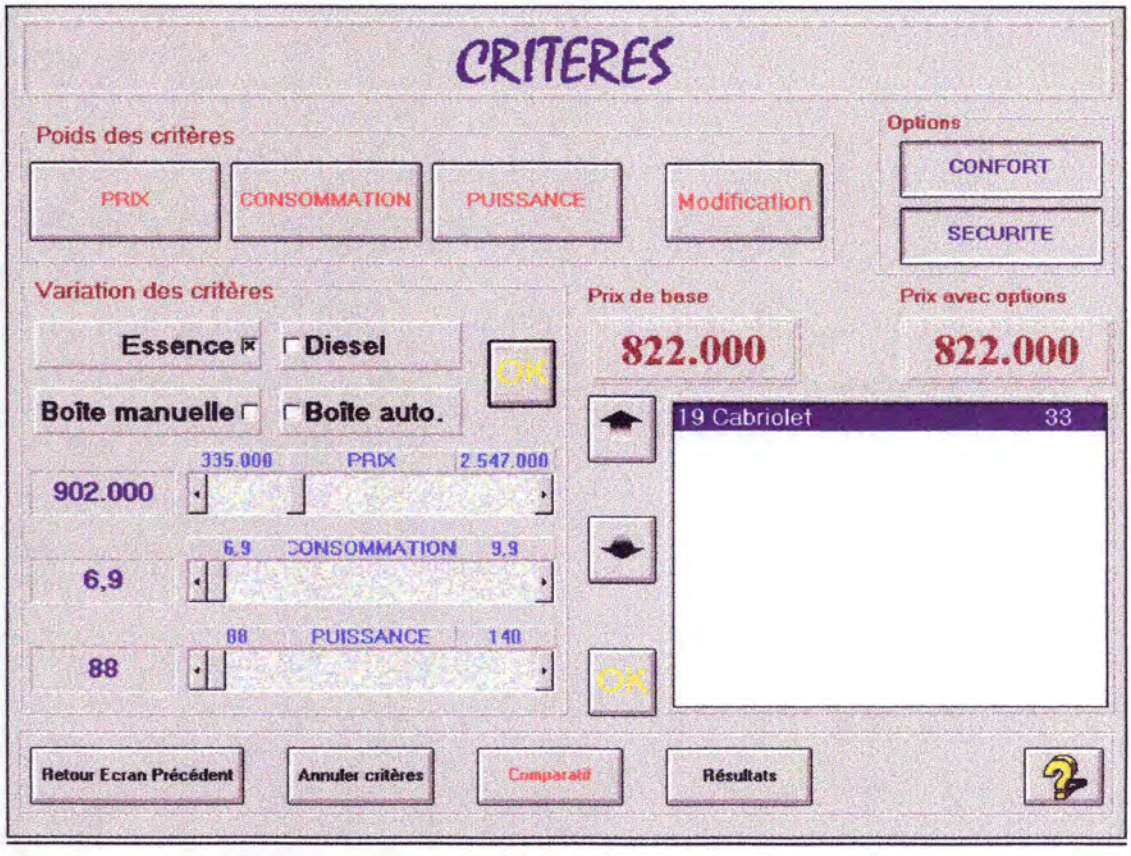
11) *Choix de la catégorie de véhicules*

12) *Ecran d'aide à l'utilisateur*

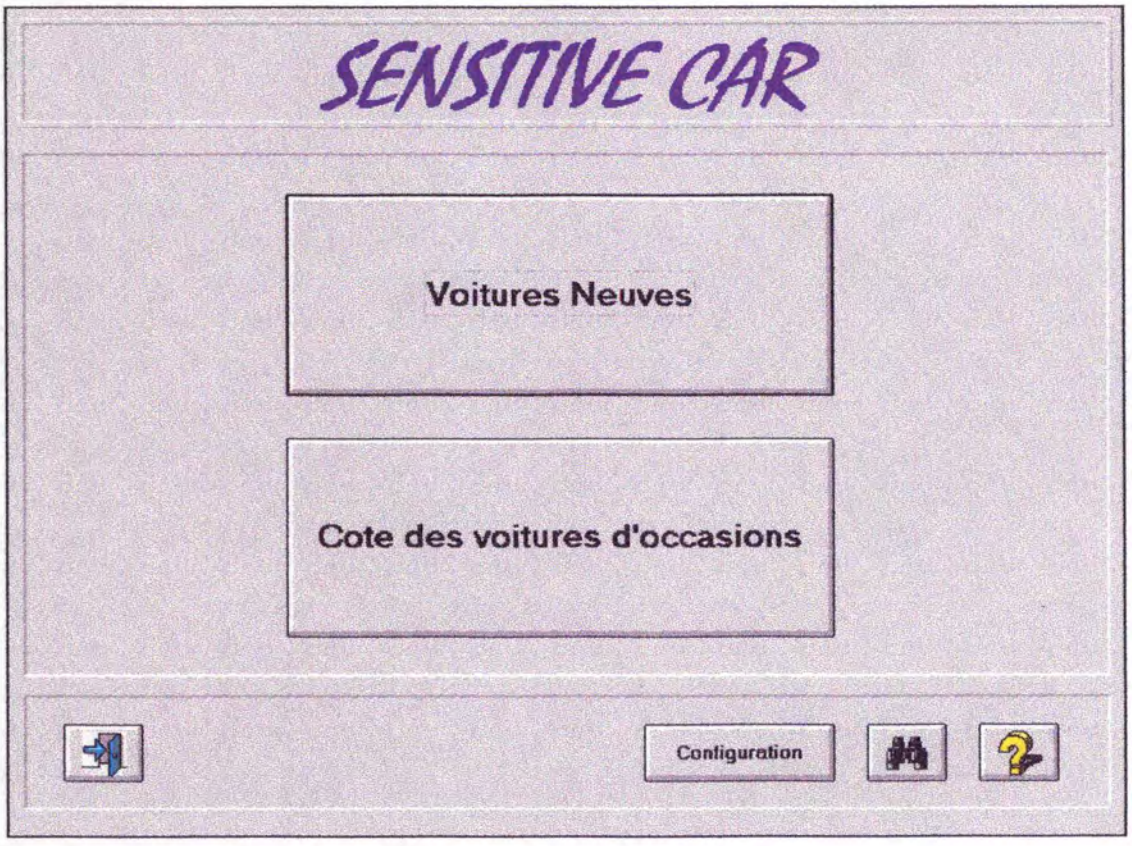
13) *About box*


SENSITIVE CAR - RENAULT
Modèles
Critères
Liste globale
Comparatifs
A propos de ...
SENSITIVE CAR
Version 1.00
Developped by
Di Donato S.
Copyright © 1996
Serial Number: H20-01-100A1001
OK
FRANCAIS
NEDERLANDS
ENGLISH
Exit

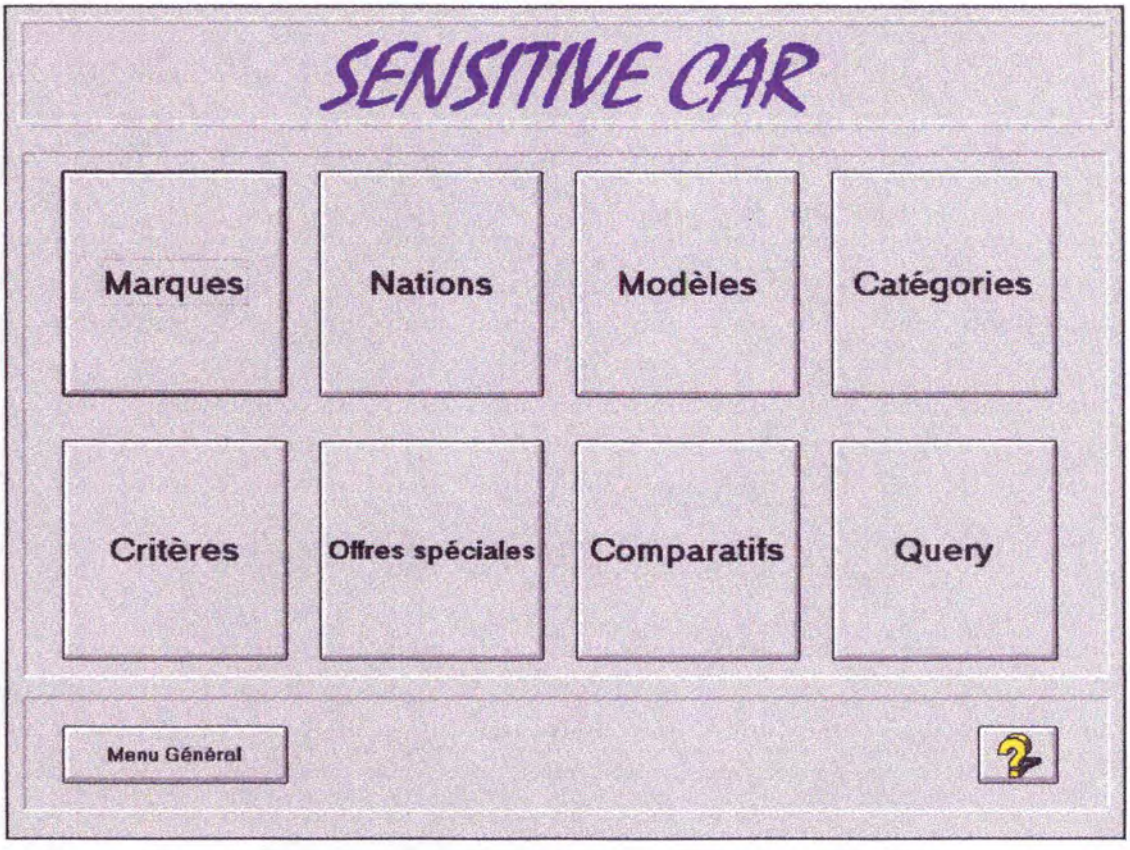
14) *Critères*


CRITERES
Poids des critères
PRIX
CONSOMMATION
PUISSANCE
Modification
Options
CONFORT
SECURITE
Variation des critères
Essence
Diesel
Boîte manuelle
Boîte auto.
OK
Prix de base
822.000
Prix avec options
822.000
19 Cabriolet 33
335.000 PRIX 2.547.000
902.000
6,9 CONSOMMATION 9,9
6,9
88 PUISSANCE 140
88
OK
Retour Ecran Précédent
Annuler critères
Comparatif
Résultats

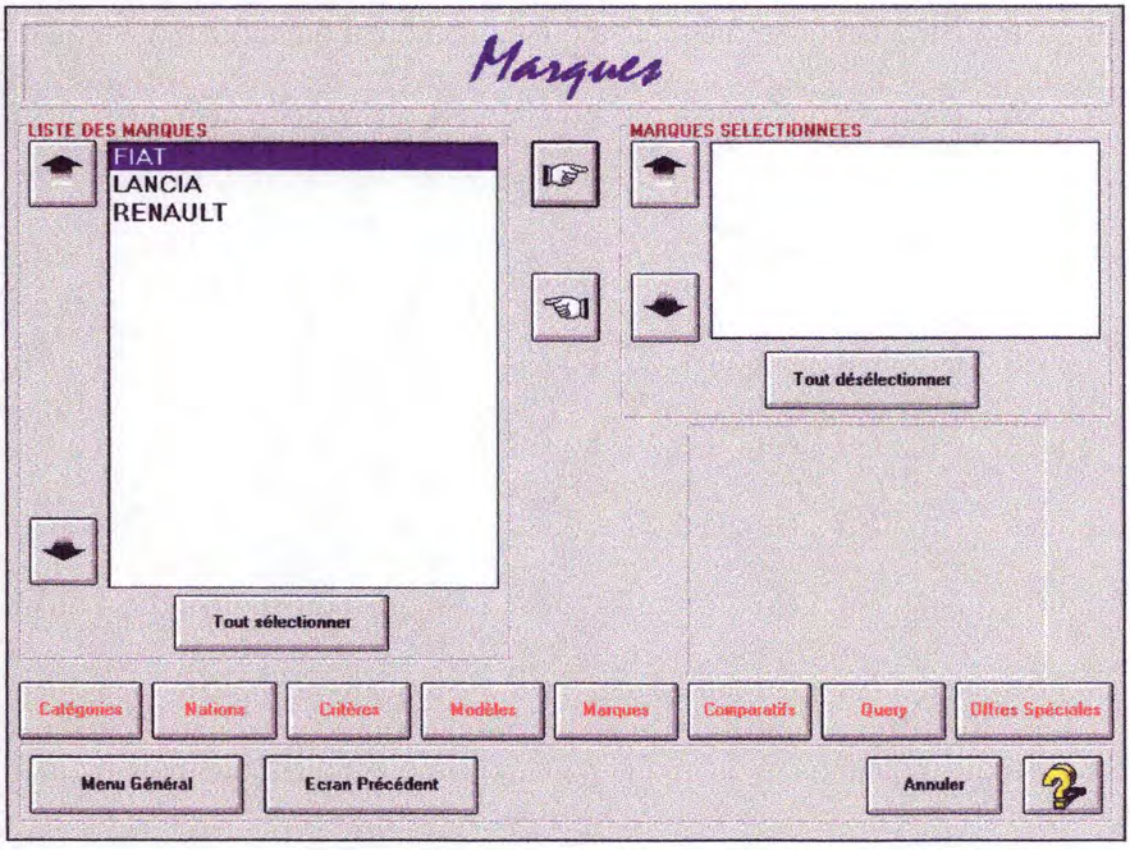
1) *Menu général*



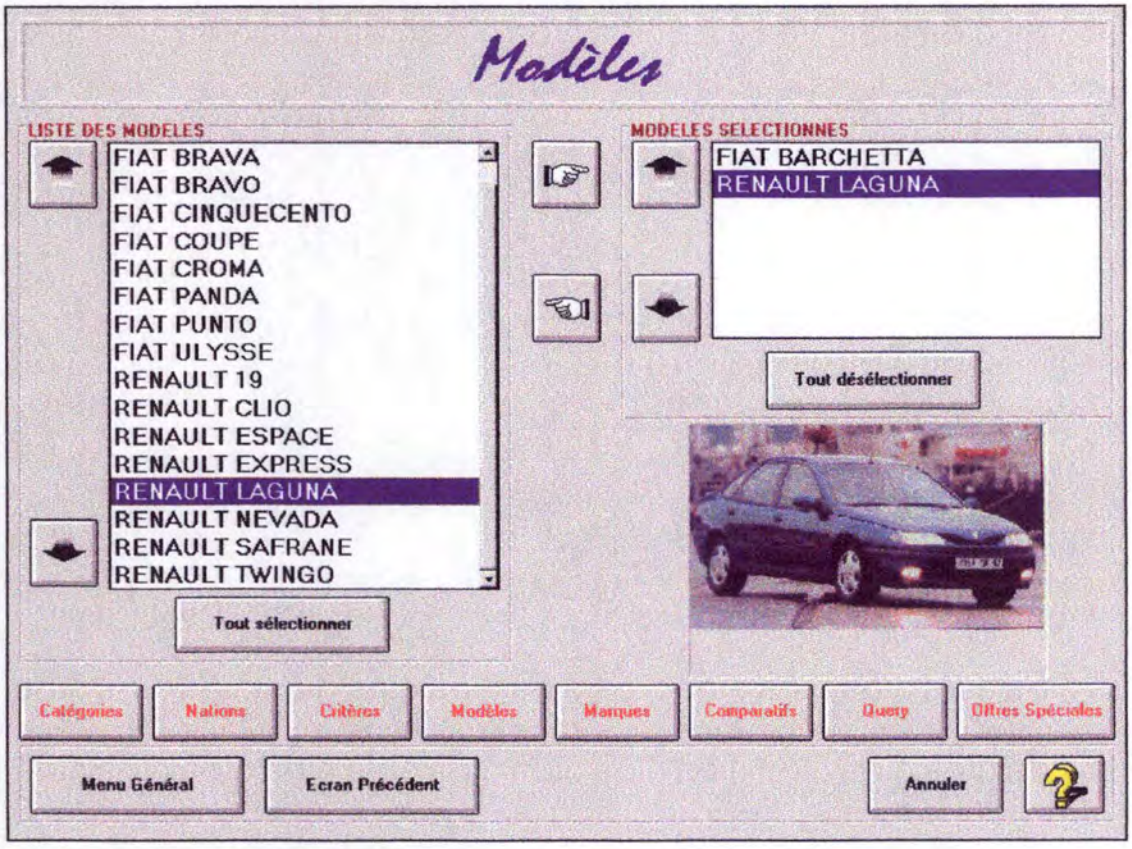
2) *Sous-menu*


SENSITIVE CAR
Marques
Nations
Modèles
Catégories
Critères
Offres spéciales
Comparatifs
Query
Menu Général

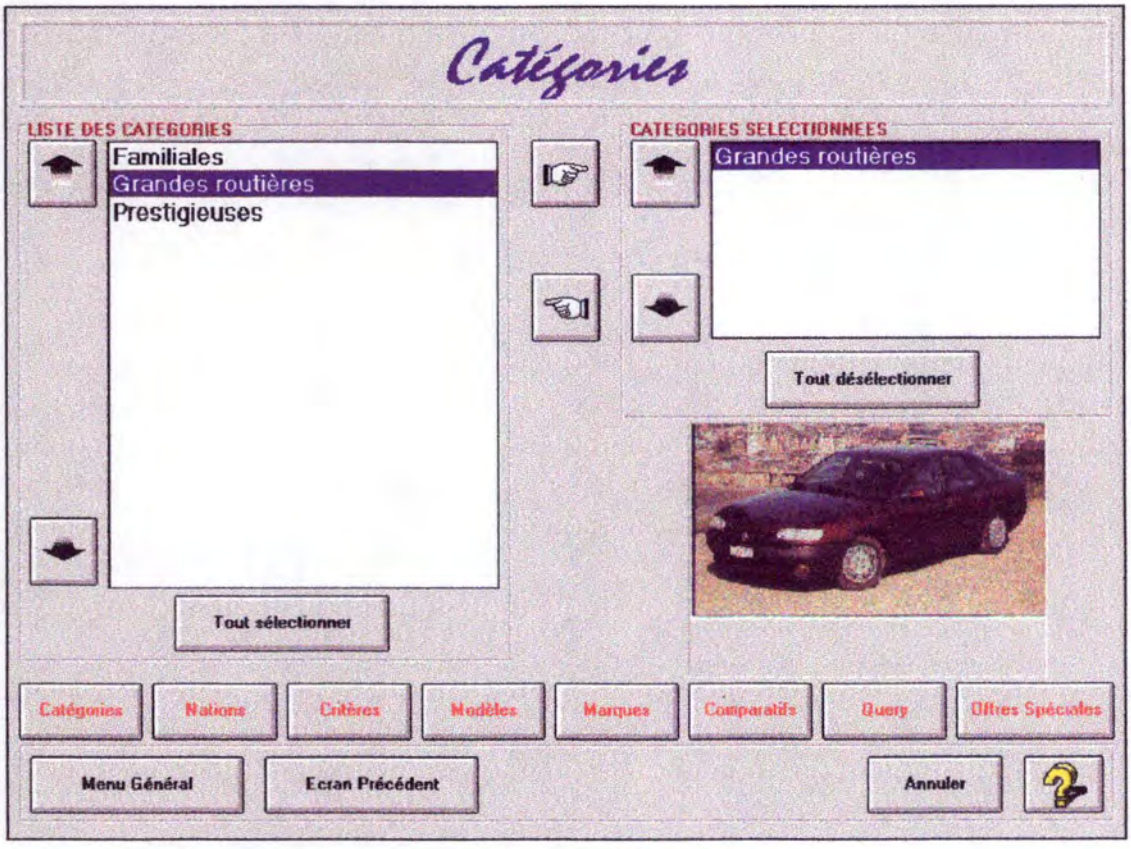
3) *Choix des critères fixes: marques*



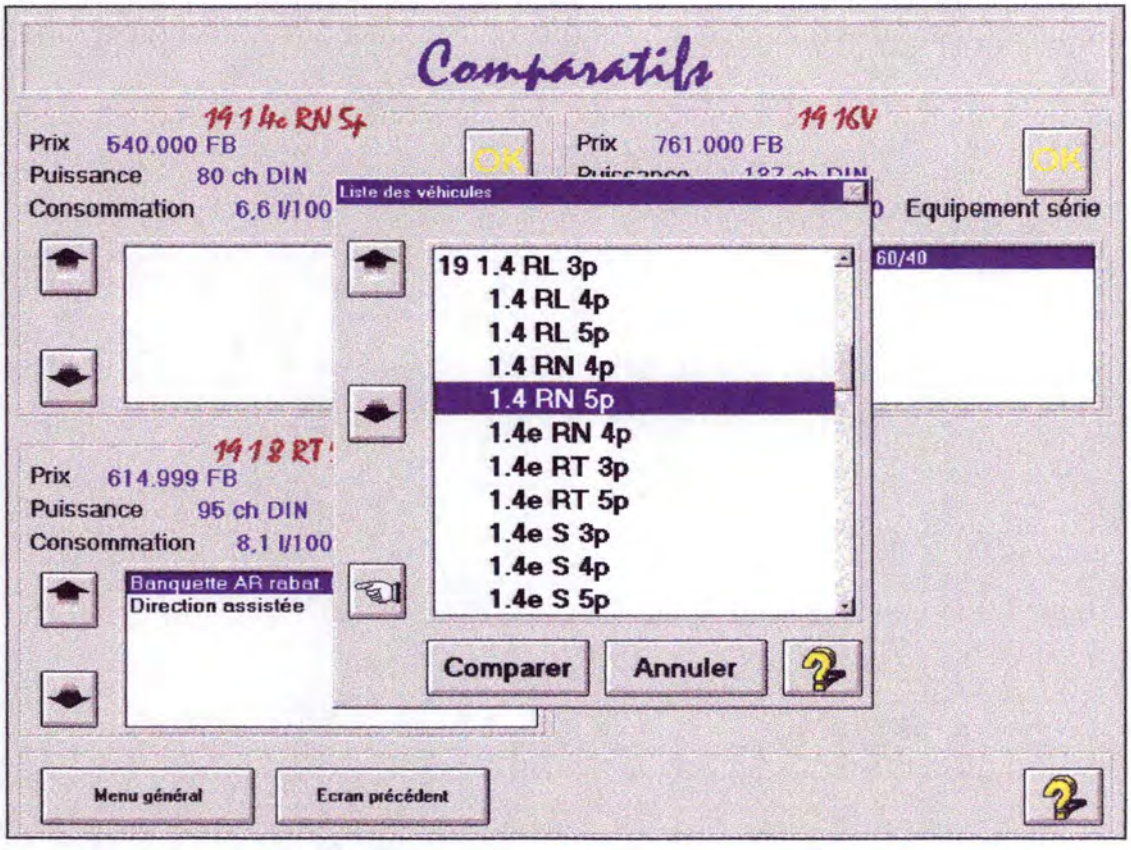
4) *Choix des critères fixes: modèles*



5) *Choix des critères fixes: catégories*

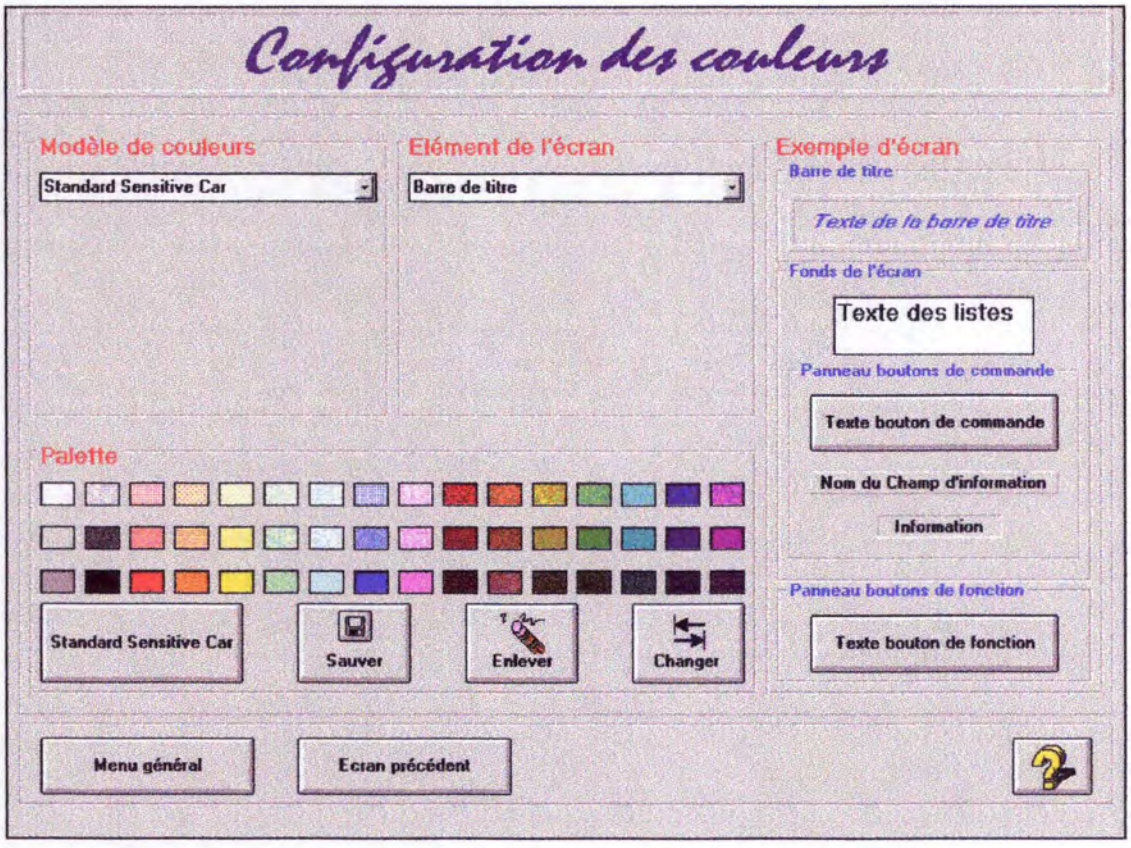


6) *Choix des véhicules à comparer*



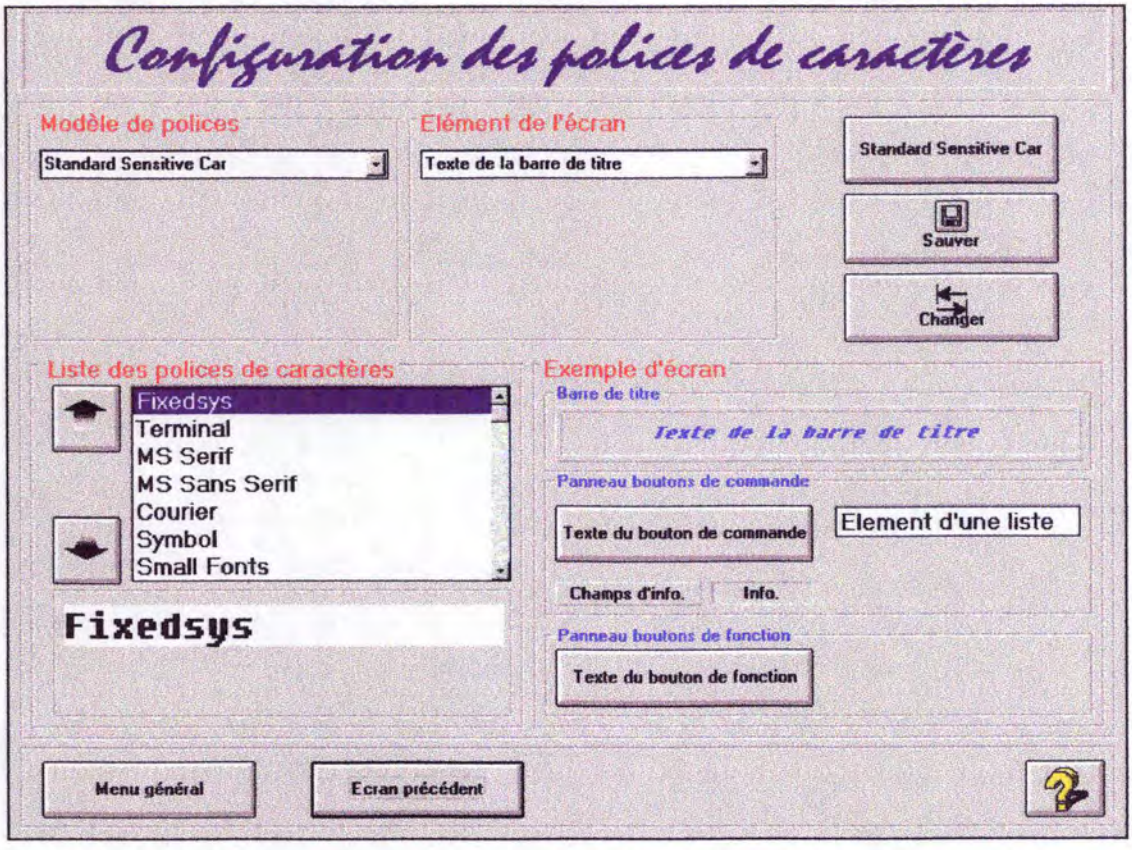
7) *Options de configuration du logiciel*

8) *Configuration des couleurs des écrans*

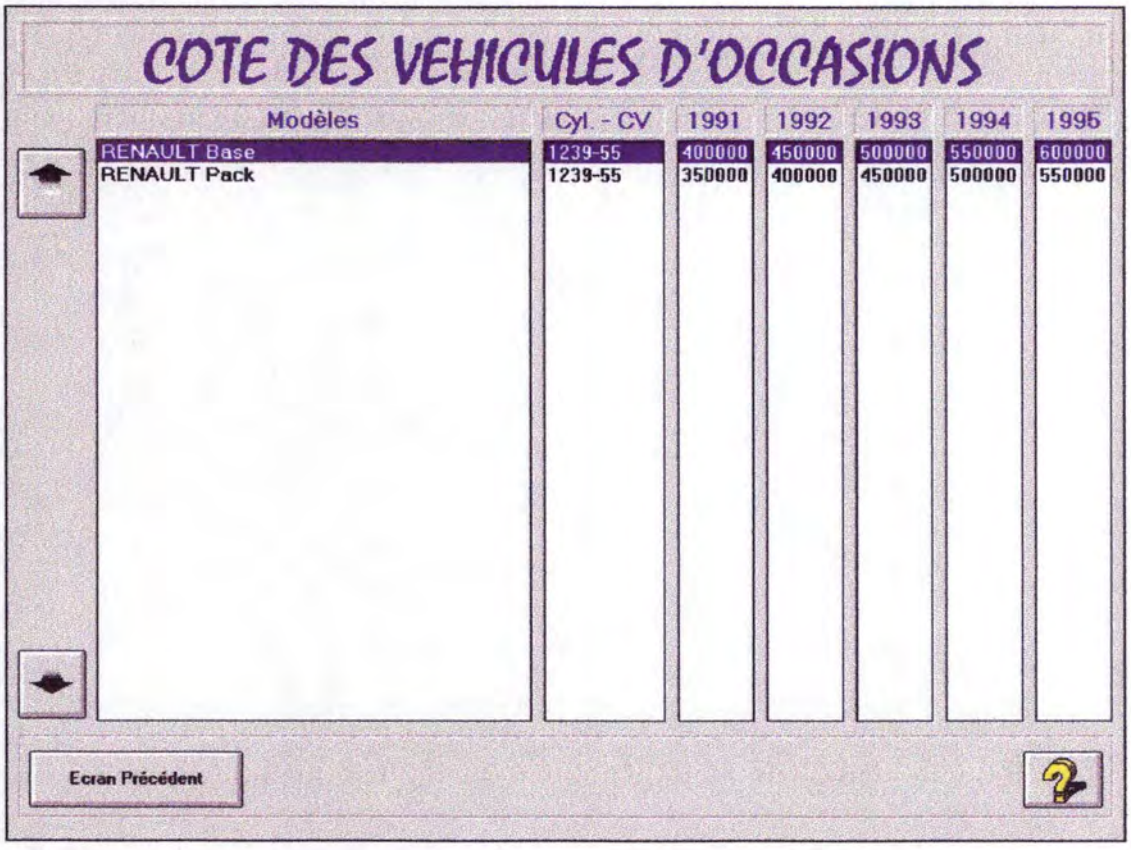


9) *Configuration de la langue utilisée par le logiciel*

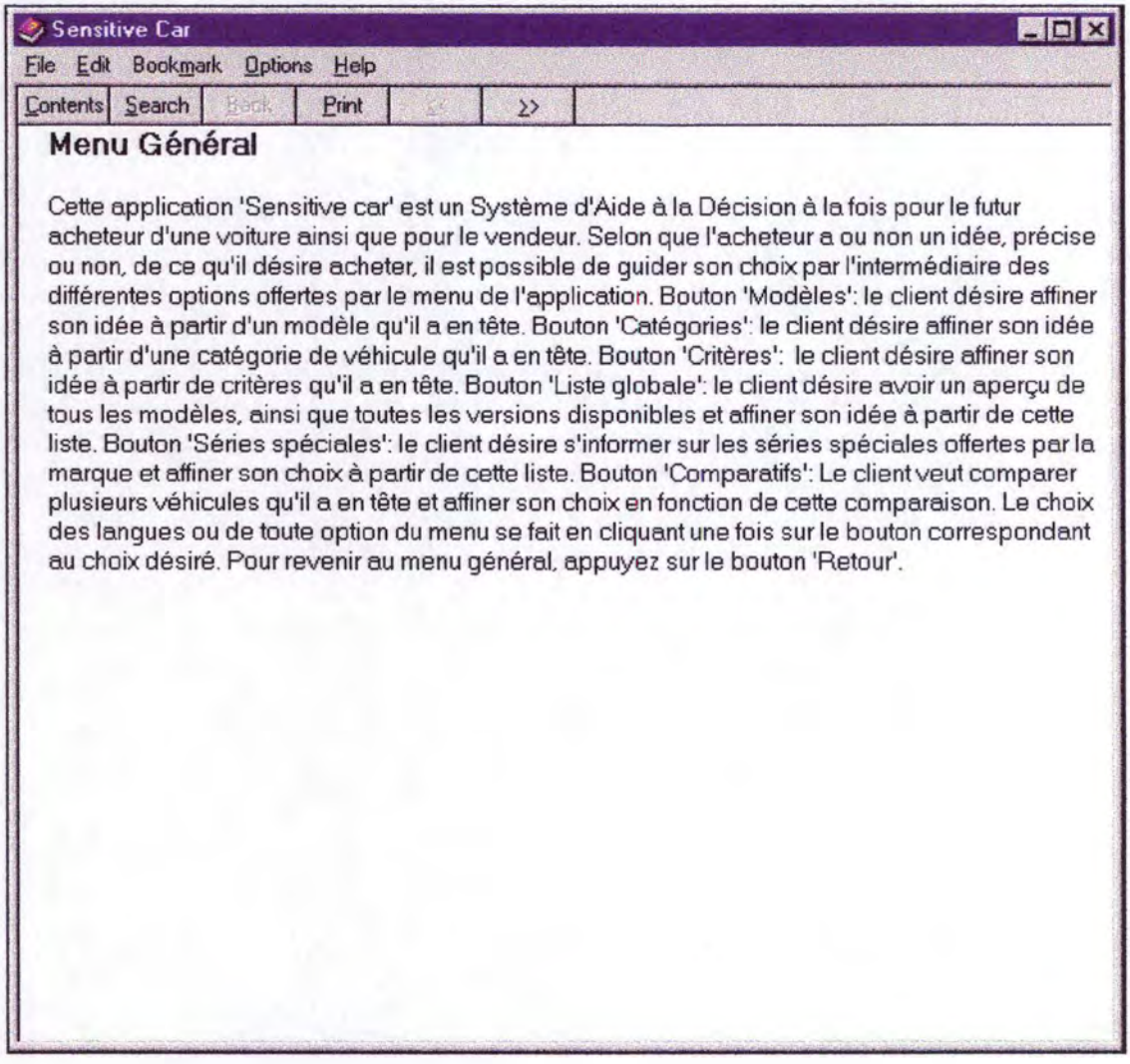
10) *Configuration des polices de caractères utilisées par le logiciel*



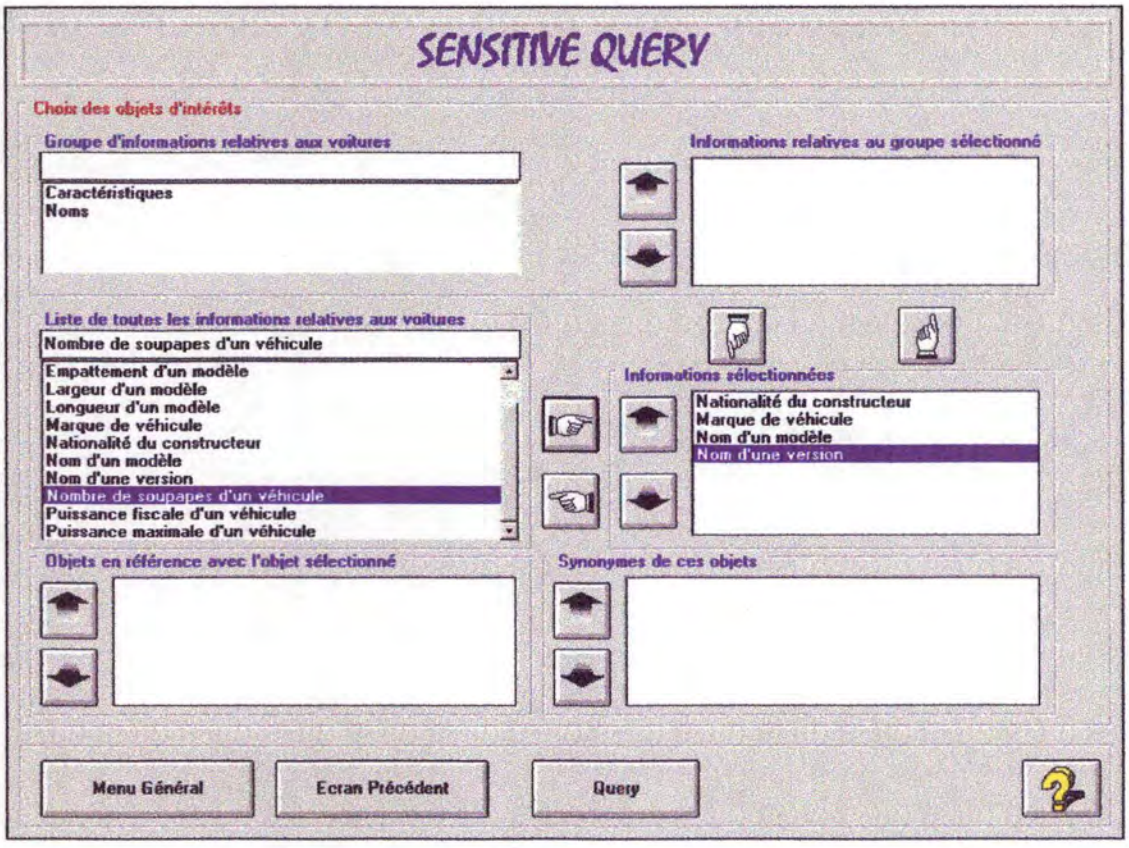
11) *Cote des véhicules d'occasion*



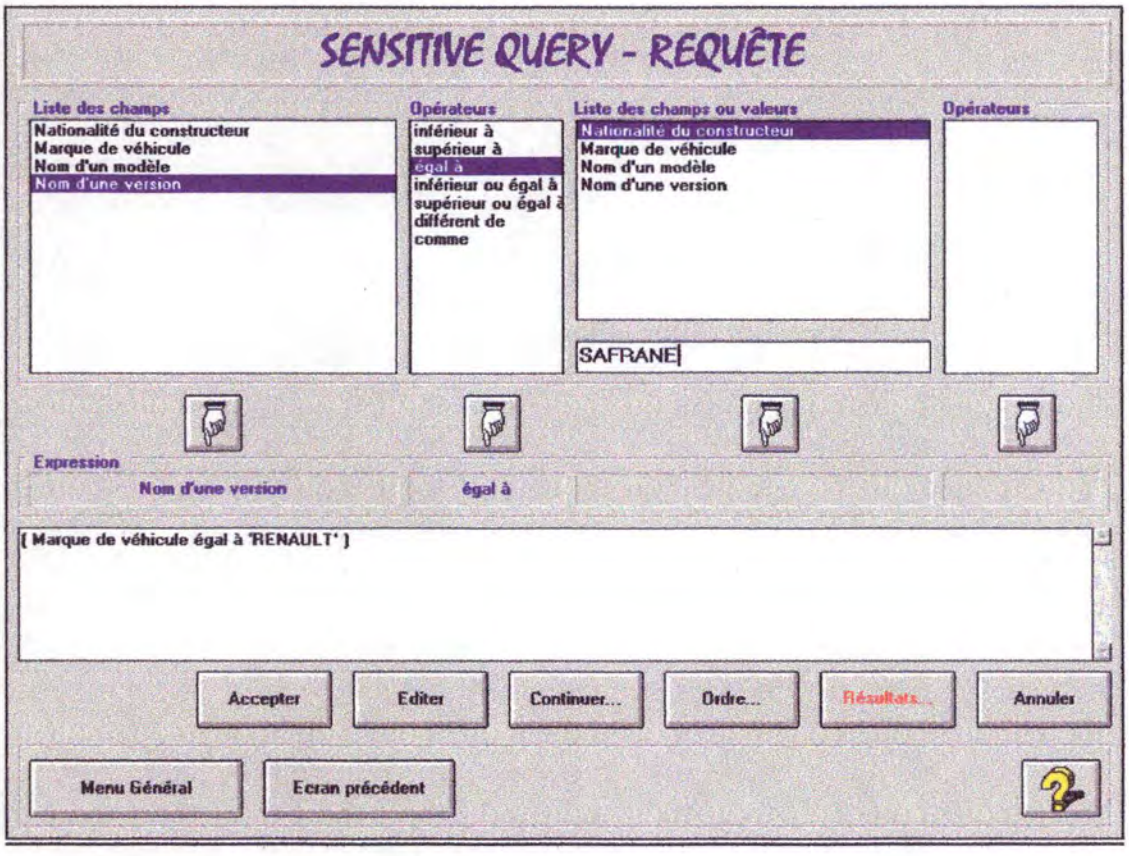
12) *Ecran d'aide*



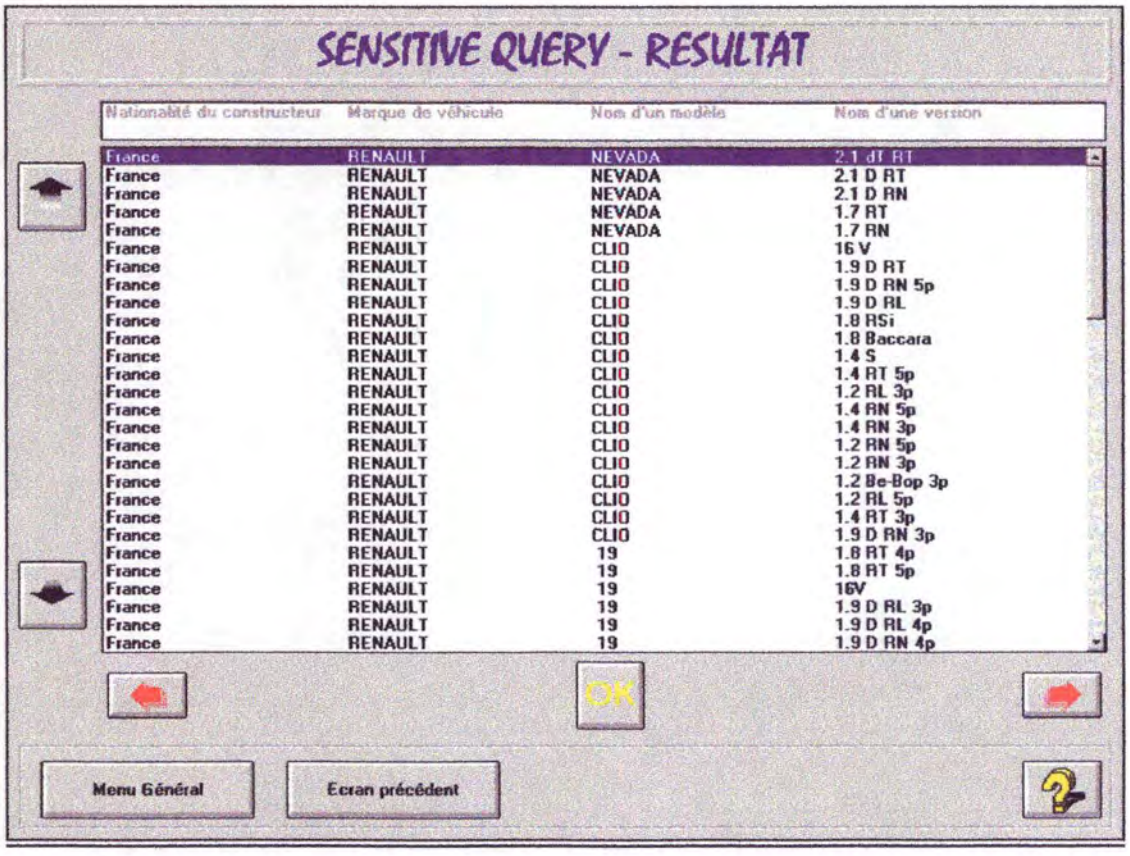
1) *écran permettant le choix des informations désirées (SELECT... FROM...)*



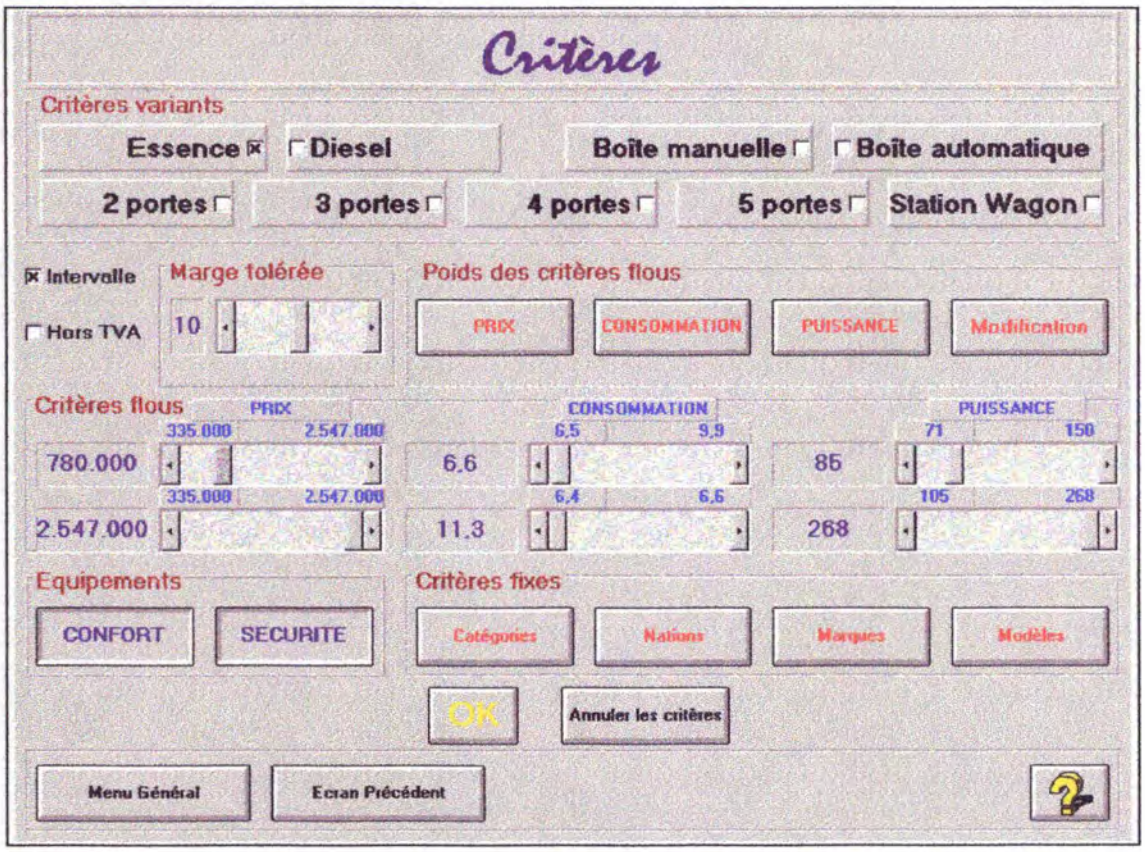
2) *écran permettant d'exprimer la clause WHERE*



3) *écran donnant la liste des résultats relatifs à la requête exprimée*



1) *Choix des critères variants et flous*



2) *Choix des équipements désirés dans le véhicule recherché*