Système de gestion de connaissances fondé sur le Principe de Coopération - Application à l'aide à la création de variétés de blés hybrides

Joël Deslandes, Marinette Revenu

Système de gestion de connaissances fondé sur le Principe de Coopération

Application à l'aide à la création de variétés de blés hybrides

DESLANDES Joël
Hybritech Europe SNC
Centre de recherche de MORANVAL
60350 Cuise-La-Motte
Tél: (33) 44.42.90.43

REVENU Marinette
Laboratoire d'Algorithmique et d'Intelligence Artificielle de CAEN (LAIAC-ISMRA)
6, Bd du Maréchal Juin
14050 CAEN Cedex
Tél: (33) 31.45.27.01
email: mrevenu@l2i.ismra.fr

Résumé

L'utilisation efficace de l'outil informatique reste un problème dans beaucoup d'entreprises: Comment gérer la connaissance accumulée depuis leur création et la rendre accessible à tous ? Nous présentons une méthode de conception des systèmes de gestion de connaissances fondée sur le principe d'une coopération entre utilisateurs, concepteurs et système. Cette coopération s'établit d'abord par la définition de concepts, appelés "normes d'interaction", communs à ces trois types d'acteurs pour représenter les connaissances et la façon de les utiliser : ce sont les tâches, les méthodes et les outils. Elle inclut également la mise à disposition de mécanismes pour organiser les connaissances, que ce soient des connaissances opératoires ou des connaissances déclaratives, portant sur le domaine de l'application ou sur son contrôle. Enfin, le système est construit sur une architecture réflexive ce qui lui permet d'adapter dynamiquement son comportement. Cette méthologie a été mise en oeuvre et le système réalisé, HybrIA, gère les données et la connaissance de la société HybriTech Europe SNC.

Mots clés:

Coopération, gestion des connaissances, système réflexif

Abstract

The efficient use of computers remains difficult in many firms: How can they manage the knowledge gathered since their creation and make it accessible to everyone ? In this paper, a method to design Knowledge Based Systems ensuring a great cooperation between users, programmers and the system is discussed. First, this cooperation is defined through concepts, named "norms of interaction", common to these three types of actors and representing the knowledge and how to use it : we distinguish tasks, methods and tools. It also includes mechanisms that can be used to organize this knowledge, whether it is operative or declarative and dealing with the application domain as well as its control. Finally, the system is built around a reflexive architecture, that allows to dynamically change its behaviour. This method has been implemented to manage the data and knowledge of the firm HybriTech Europe SNC.

Keywords:

Cooperation, knowledge management, reflexive system

1 Introduction

La société HybriTech Europe SNC est spécialisée dans l'élaboration de nouvelles variétés de blés par hybridation au moyen d'un agent chimique. En parallèle, elle poursuit son programme de sélection classique utilisant l'hybridation manuelle. Cette activité regroupe un ensemble de partenaires européens dans un projet EUREKA. L'objectif visé est la commercialisation, dès 1995, de variétés créées par hybridation chimique.

Les chercheurs expérimentent chaque année environ 80 000 individus se situant à différents stades du schéma de sélection qui dure de 6 à 10 ans. Afin de trouver les variétés les meilleures, ils ont besoin d'une gestion informatique efficace des données expérimentales et de moyens souples et variés pour les mettre en relation, en correspondance avec leur expertise et conformément à leurs méthodes de travail. En particulier, doivent être intégrés un SGBD, des bibliothèques de calculs statistiques et des logiciels de présentation et d'impression de résultats. Une contrainte supplémentaire est l'incrémentalité de la solution informatique proposée : l'intégration des connaissances doit pouvoir se faire progressivement sans remise en cause de l'architecture du système pendant son cycle de vie.

En réponse à ce problème, nous proposons une contribution pour la mise en place de systèmes industriels permettant de gérer la connaissance. Tout d'abord, de façon à rendre compte de la démarche sous-jacente au travail décrit ici, nous présenterons les acquis en représentation des connaissances issus des recherches menées en Intelligence Artificielle. Nous introduirons ensuite le Principe de Coopération destiné à être un guide pour acquérir la connaissance. Les spécifications du système seront alors explicitées selon le point de vue de l'utilisateur. Enfin, nous décrirons l'implantation du système HybrIA réalisé dans le cadre d'une thèse CIFRE.

2 Acquis en représentation des connaissances

L'utilisation efficace de l'outil informatique reste un problème dans beaucoup d'entreprises : "Comment gérer la connaissance accumulée depuis leur création et la rendre accessible à tous ?". Une des hypothèses fondatrices de l'Intelligence Artificielle repose sur la possiblité de construire des systèmes symboliques physiques capables de réaliser toute action intelligente [Newell & Simon 1976]. Ces systèmes doivent être composés de symboles organisés en expressions ou structures et disposer de processus opérant sur ces expressions pour en produire de nouvelles.

L'application de cette hypothèse au problème de la gestion de connaissances suppose que la pensée créatrice peut se ramener à une manipulation syntaxiquement rigoureuse de symboles, eux-mêmes physiquement instanciés, à l'aide d'expressions énoncées dans un langage formel, et que l'action est isomorphe à une pensée structurée qui la contrôle et la détermine [Visetti 1991]. Ce postulat induit par contrecoup une réduction de la connaissance à sa représentation [Bachimont 1992]. Or, la connaissance n'est pas une notion scientifique formelle et elle ne se laisse pas capturer aisément par une représentation symbolique. Les systèmes experts font partie des exemples paradigmatiques de l'approche symbolique [Laurière 1987]. Ils s'appuient sur des logiques qui assurent leur cohérence mais qui entraînent aussi leurs limites. Le formalisme imposé aux experts ainsi que les mécanismes d'inférence sous-jacents bloquent souvent le processus d'explicitation des connaissances.

Pour échapper à une focalisation portant sur la représentation des connaissances lors de la phase de conception d'un Système à Base de Connaissances, ou SBC, A. Newell a introduit, dans sa description en couches d'un système informatique, un "Niveau Connaissance", indépendant de l'implantation que l'on peut en faire [Newell 1982]. De cette étude, émerge une directive qui consiste à privilégier d'abord l'aspect fonctionnel du système, c'est-à-dire ce qu'il doit faire, pour décrire l'ensemble des perceptions/actions, l'ensemble des buts et le corpus de connaissances descriptives qui caractérisent le système. Cette prise en compte de la séparation entre connaissance et représentation s'est traduite par deux axes de recherche dans l'activité d'ingénierie de la connaissance : *proposer une structuration du niveau connaissance* ou *aider à son opérationnalisation* .

Dans le premier cas, il s'agit alors de construire un modèle conceptuel situé entre connaissance et représentation. Le projet KADS [Schreiber et al 1988] propose une structuration des connaissances selon 4 niveaux: *stratégies* (plans et métarègles), *tâches* (buts et décomposition d'une tâche en sous-tâches à l'aide de termes de contrôle), *inférences* (opérateurs primitifs agissant sur des données décrites par un rôle et organisés dans des structures d'inférence) et *domaine* (concepts, propriétés, relations et structures de données).

Dans le second cas, de nouvelles architectures ont été conçues pour construire un pont entre l'expression naturelle des connaissances et la représentation opérationnelle finale. Elles s'appuient sur les *modèles*, analogues aux entités du niveau domaine de KADS, sur les *tâches*, entités structurées décrivant ce qu'il faut faire pour atteindre un but et sur les *méthodes*, spécifiant comment une tâche peut être accomplie, soit directement, soit en la décomposant [Pierret-

Goldreich et Delouis 1990]. Dans de tels systèmes, à un but est associée une tâche qui peut elle-même se décomposer en sous-tâches. C'est cette décomposition et la description d'une ou plusieurs méthodes pour effectuer une tâche qui constituent la représentation de la connaissance. Les systèmes COMMET [Steels 1990] [Steels 1992] et COPILOTE [Delouis & Krivine 1992] rendent opérationnelles cette représentation en lui associant des codes informatiques et en assurant le contrôle de l'enchaînement des traitements. En particulier, les connaissances portant sur le contrôle ont le même statut dans le système que les connaissances spécifiques au domaine. Il peut donc exister plusieurs méthodes de résolution pour effectuer les choix des méthodes et l'appel à l'utilisateur figure parmi celles-ci. C'est pourquoi plusieurs modes de coopération entre utilisateurs, concepteurs et machine, peuvent cohabiter dans le système.

Nous nous sommes placés dans le courant "opérationnalisation du niveau connaissance". Dans un premier temps, nous avons tenté d'appliquer une démarche analogue à celles présentées ci-dessus et il est apparu un point d'achoppement lié au problème de la formulation de la connaissance. Les approches précédentes supposent que la connaissance soit formulable pour être représentée. Ce n'est pas le cas dans les domaines où la connaissance n'est pas totalement formalisée, en particulier dans le domaine de l'expertise en sélection de blés, et où, malgré tout, un outil informatique favorise une démarche experte. Même en se plaçant au "niveau connaissance", l'expert est obligé d'utiliser un formalisme qui ne lui est pas naturel pour exprimer sa connaissance. La solution que nous préconisons consiste à ajouter un niveau de description, les *outils* considérés comme briques de base de la connaissance, sur lesquelles un consensus est plus aisément établi, ainsi qu'un mécanisme pour les combiner, puis de placer l'expert utilisateur en situation d'expérimentation. Les outils sont les procédures informatiques qu'il utilise couramment et pour lesquelles il sait comment déterminer les paramètres d'entrée et comment utiliser les résultats. Il peut alors construire des réseaux de dépendances entre ces briques ainsi que le présente la figure 1 et la connaissance introduite dans le système résultera de l'interprétation qui en sera faite. Dans cette optique, toute tâche que l'on souhaite voir réaliser par le système peut avoir une représentation propre à chaque utilisateur.

C'est par la participation active des utilisateurs que s'effectuera l'acquisition des connaissances. Le Principe de Coopération est à la base de l'élaboration de SBC intégrés dans les systèmes d'information des entreprises car ils seront bien acceptés par les experts. L'ordinateur sera alors considéré comme une technologie intellectuelle au service des utilisateurs.

3 Principe de Coopération

L'interaction entre l'homme et la machine devrait être à la base des SBC. Parmi les agents informatiques et les agents humains, chacun réaliserait alors les tâches qu'il fait le mieux, et l'intelligence du système serait une émergence de cette collaboration [Lenat et Feigenbaum 1991]. Pour qu'un système soit "intelligent", il est nécessaire qu'il se construise et s'utilise en coopération avec l'utilisateur qui est introduit directement dans la boucle de conception [Boy 1988].

Pour introduire le plus naturellement possible l'utilisateur dans la boucle de conception, nous proposons une méthode fondée sur un "Principe de Coopération" pour établir le modèle conceptuel du système. Coopérer signifie mettre en commun des moyens pour atteindre un but. L'objectif est d'obtenir un système permettant de gérer des connaissances, c'est-à-dire de les représenter et de les rendre opérationnelles. Trois agents coopèrent à la résolution de ce problème : le système, les concepteurs et les utilisateurs. Le système offre un cadre pour représenter la connaissance et utiliser cette représentation : il s'agit des normes d'interaction. Le concepteur crée les briques de base. L'utilisateur, en menant des expérimentations à l'aide de ces briques de base, construit une représentation opérationnelle de sa connaissance qu'il peut ensuite formaliser et éventuellement généraliser.

3.1 Les normes d'interaction

Une norme d'interaction entre un utilisateur et le système définit un mode d'expression de la connaissance et l'utilisation que l'on peut en faire. En plus des concepts de tâches et de méthodes, nous avons introduit les outils, les groupes d'outils et les groupes de tâches.

Les concepts

- Un outil est une représentation d'un code informatique incluant les types des paramètres d'entrée et du résultat produit.
- Une méthode est une représentation d'une utilisation d'un outil destiné à accomplir une tâche.
- Une tâche matérialise un but dans le système. Une même tâche peut être effectuée par plusieurs méthodes selon le contexte, les contraintes et les préférences de l'utilisateur.
- Un groupe de tâches permet de structurer logiquement les tâches à l'aide de hiérarchies. Une même tâche peut faire partie de plusieurs groupes selon le point de vue adopté.
- Un groupe d'outils, de la même façon, organise logiquement les outils.

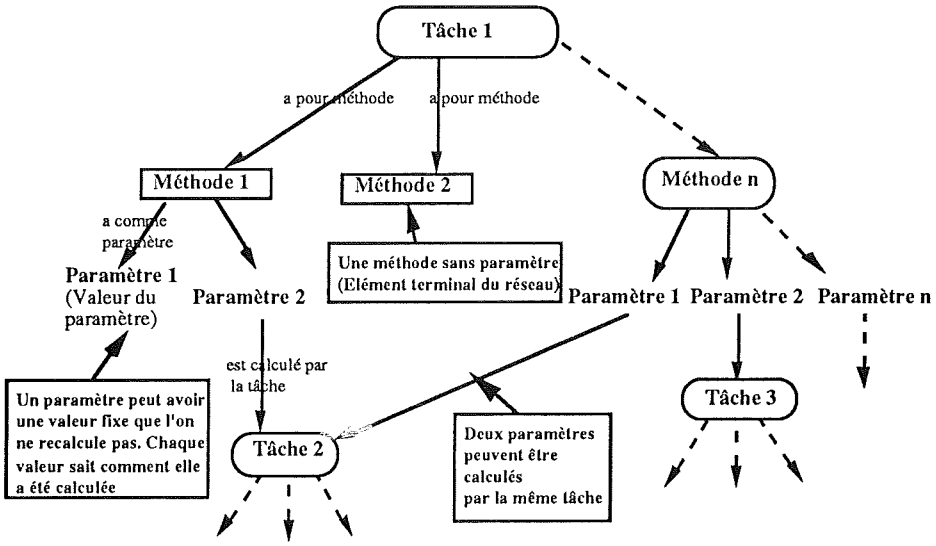

figure 1 : Réseau de dépendances entre tâches et méthodes

Utilisation des concepts

Pour appréhender les briques de base que l'utilisateur a à sa disposition, plusieurs vues sont proposées: une vue structurée des groupes d'outils et des groupes de tâches suivant la relation groupe - sous-groupes et une vue des relations entre tâche, méthodes qui peuvent être utilisées pour l'effectuer et sous-tâches permettant de calculer les valeurs des paramètres des méthodes. Le réseau de dépendances de la figure 1 montre le principe de décomposition d'une tâche en sous-tâches.

A chaque vue correspond une interface graphique à l'aide de laquelle l'utilisateur peut agir: visualisation partielle des arborescences, édition des entités présentées, modification de l'organisation.

3.2 Création des briques de base : les outils

Le travail de l'informaticien consiste à créer des outils en tant que représentation de codes informatiques. Pour cela, il décrit la nature du code, les entrées et la sortie produite, ainsi que les caractéristiques opératoires. L'activation d'un outil déclenchera automatiquement les actions prévues dans le code. Toute représentation des connaissances utilisant un de ces outils sera donc directement opérationnelle.

Les outils appartiennent à deux catégories : ceux qui agissent sur les données du domaine et ceux qui opèrent sur les entités décrivant le système.

3.3 Utilisation des briques de base : rôles des acteurs et modes de coopération

L'utilisation des briques de base se fait en deux étapes : création de méthodes et de tâches, puis création de réseaux de dépendances. Nous allons décrire le rôle de chacun des intervenants et les modes de coopérations qui s'établissent alors.

Rôle de l'utilisateur

L'utilisateur a à sa disposition un ensemble d'outils qu'il peut utiliser pour créer des méthodes et des tâches. La création d'une tâche n'implique pas la création immédiate des sous-tâches nécessaires à son accomplissement. La création du réseau de dépendances se fait au fur et à mesure lors de l'exécution d'une tâche. Selon la nature du contrôle mis en place, c'est l'utilisateur ou le système qui peuvent choisir une méthode adaptée à la tâche courante. A cette méthode sont associés des paramètres d'entrée typés. Dans une approche "dirigée par l'utilisateur", c'est l'utilisateur qui détermine la tâche la plus appropriée, en *coopération avec le système* qui propose des tâches susceptibles de calculer des données de ces types. Ce processus est récursif et c'est par la représentation de ces liens entre tâches et méthodes que l'utilisateur crée un réseau de dépendances.

S'il n'existe pas de tâche susceptible de calculer une donnée, l'utilisateur en crée une et la relie à un outil renvoyant une valeur du type souhaité, par l'intermédiaire d'une méthode. Dans ce cas, il y a création d'une branche "tâche, méthode, outil". S'il n'existe aucun outil calculant une valeur du type souhaité, l'utilisateur peut créer une représentation d'un outil, d'une méthode et d'une tâche pour calculer ce paramètre. Cette représentation n'est pas opérationnelle, seuls les champs descriptifs des entités créées sont remplis par l'utilisateur, mais elle est destinée à être utilisée par le programmeur comme spécification d'un outil à créer. Ainsi s'établit une *coopération utilisateur-concepteur.*

Rôle du concepteur

Pour que les actions de contrôle sur le système agissent de la même façon que les actions de

contrôle sur les données de l'application, une architecture réflexive est nécessaire. Le concepteur peut alors procéder comme l'utilisateur final pour la mise en place des tâches de contrôle. Par exemple, il doit créer les tâches "lancer un outil" et "exécuter une tâche". En particulier, il lui est possible de faire cohabiter plusieurs types de contrôle puisqu'à une tâche peuvent être associées plusieurs méthodes. Chaque type de contrôle, que nous avons appelé "approche", fixe la méthode à appliquer pour chaque tâche de contrôle sur le contrôle, évitant ainsi le problème de la récursion à l'infini. Dans la première étape de conception d'un système, trois approches nous ont semblé indispensables : une *approche dirigée par l'utilisateur* où il est fait appel à l'utilisateur à chaque choix, une *approche de routine* où le choix de la première méthode est systématique et une *approche expliquée* qui permet de faire l'apprentissage du système.

Rôle du système

Nous souhaitons que le comportement du système s'adapte en fonction de la description de l'utilisateur qui peut être expert du domaine ou concepteur. Une représentation des connaissances peut alors être propre à chaque utilisateur. Pour cela, chacun d'eux dispose d'un espace de travail dans lequel il peut créer des tâches et des méthodes et les structurer comme il le souhaite. Pour faciliter la perception des connaissances disponibles dans le système, seules sont présentées celles qui appartiennent au niveau requis. Les utilisateurs sont placés dans des groupes caractérisés par une compétence sur le domaine et une compétence sur le système. Chaque tâche et chaque méthode disposent de deux attributs descriptifs spécifiant les compétences nécessaires pour les utiliser.

4 Implantation du système HybrIA

Le système HybrIA est le résultat de l'implantation du modèle conceptuel décrit ci-dessus. N. Aussenac souligne que la définition d'un environnement pour l'acquisition de connaissances est nécessairement complexe [Aussenac et al 1992] : en plus du choix des processus et de la spécification de leur enchaînement, elle nécessite un environnement dans lequel ils pourront s'agencer. HybrIA est réalisé en CLOS (Common Lisp Object System) avec l'environnement de développement orienté objet GENERA de Symbolics. La programmation par objets offre un mode de codage informatique en correspondance avec l'analyse au "niveau connaissance" et elle contribue à structurer notre compréhension du monde et de la connaissance [Nicolle 1987] [Nicolle 1990]. Les classes intervenant dans la représentation des connaissances et leurs liens sont montrés sur la figure 2. Tous les objets sont maintenus dans une base de données créée par le SGBD Orienté Objets STATICE.

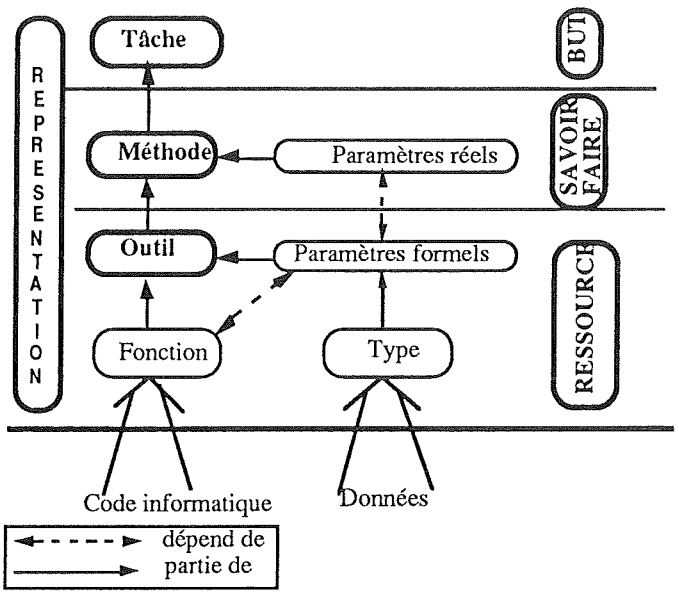

Figure 2 : les concepts de tâche, méthode et outil

Les tâches, méthodes et outils sont décrits dans HybrIA par des attributs dont les principaux sont énumérés ci-dessous. D'autres attributs peuvent être ajoutés dynamiquement par le système, si nécessaire.

Les attributs en **gras** sont obligatoires, les attributs normaux sont facultatifs, les attributs en *italique* sont remplis automatiquement par le système.

Tâche
- **Nom**
- **Méthodes**
- NiveauSystème
- NiveauDomaine
- Documentation
- Granularité
- *Date De Création*
- *Créateur*

Méthode
- **Nom**
- **Outil**
- **Paramètres**
- NiveauSystème
- NiveauDomaine
- Documentation

Outil
- **Nom**
- **Groupe**
- **Type**
- **Fonction**
- **Paramètres**
- NiveauSystème
- NiveauDomaine
- Documentation
- Granularité
- *Date De Création*
- *Créateur*

Les types sont représentés également par des objets et ils sont regroupés dans une hiérarchie à laquelle l'utilisateur peut accéder à l'aide d'une interface pour la visualiser et la modifier.

L'attribut **NiveauSystème** peut prendre les valeurs suivantes : Concepteur, Administrateur,

Stagiaire, Chercheur-Créateur, Chercheur-Utilisateur, et l'attribut **NiveauDomaine** : Expert, Efficace, Compétent, DébutantAvancé, Débutant. Leur rôle est de définir par quels types d'utilisateurs les entités décrites peuvent être utilisées. L'attribut **Granularité** spécifie des propriétés telles que le temps d'exécution, les ressources mémoire utilisées et la complexité au moyen d'une échelle de valeurs discrètes.

Pour éviter, lors de l'exécution, la récursion à l'infini qui pourrait découler, par exemple, de l'exécution de la tâche "Exécuter une tâche" pour la tâche "Exécuter une tâche" ... , un mécanisme spécifique a été mis en place. Pour chaque tâche de contrôle telle que "Exécuter une tâche", le choix de la méthode à exécuter est déterminé directement par la valeur de la variable "Approche". Cette variable qui fait partie de la description du contexte d'une session fige, en quelque sorte, le contrôle sur le contrôle. Les attributs descriptifs d'une session sont:

Session	Contexte
Date Ouverture	**Utilisateur**
Contexte	**Tâche-courante**
Fichier de Trace	**Méthode-courante**
	Outil-courant
	Approche

Le comportement du système est donc fonction de l'utilisateur et de l'approche choisie. Le choix d'une méthode ne se fait pas actuellement en fonction d'autres critères. C'est par l'ajout d'autres approches que le système pourra acquérir des capacités de résolution automatique de problèmes.

5 Conclusion

La conception de ce système a été à l'origine d'une réorganisation de l'information gérée par l'entreprise. Le système est opérationnel depuis un an et demi pour assurer la gestion classique des données. La base de données contient 300 000 objets du domaine ainsi que 175 outils, 130 tâches et 200 méthodes. Le système est bien accepté et pendant les absences du concepteur, il a déjà été modifié par des utilisateurs non informaticiens.

Le système n'intègre pas de connaissances pour résoudre des problèmes selon des méthodes générales. Il a été développé selon un principe pragmatique et aucune tentative n'a été faite pour pousser l'utilisateur à introduire des tâches plus conceptuelles, telles que la généralisation ou la spécialisation, tâches conduisant à la mise en place d'un apprentissage automatique de connaissances. Ce pourra être fait dans une étape ultérieure sans révision totale de l'architecture du système, puisque celle-ci est réflexive.

6 Bibliographie

[AUSSENAC et al 1992] Nathalie Aussenac-Gilles, Jean-Paul Krivine, Jean Sallantin, "L'acquisition des connaissances pour les systèmes à base de connaissances", Revue d'Intelligence Artificielle, Vol. 6 - n°1-2/1992, pp7-15.

[BACHIMONT 1992] Bruno Bachimont, "Le contrôle dans les systèmes à base de connaissances", Ed Hermès, 1992.

[BOY 1988] Guy Boy, "Assistance à l'opérateur : Une approche de l'intelligence Artificielle", Ed Teknea, 1988.

[DELOUIS et KRIVINE 1992] Isabelle Delouis et Jean-Paul Krivine, "Opérationnalisation du modèle conceptuel : Vers une architecture permettant une meilleure coopération système-utilisateur", Avignon'92, pp 165-176, 1992.

[LAURIERE 1987] J-L Laurière, "Résolution de problèmes par l'homme et la machine", Eyrolles, 1987.

[LENAT et FEIGENBAUM 1991] D.B. Lenat et E.A. Feigenbaum, "On the thresholds of knowledge"; Artificial Intelligence, 47, pp 185-250, 1991.

[NEWELL et SIMON 1976] A. Newell et H.A.Simon, "Computer Science as Empirical Inquiry : Symbols and Search", The tenth Turing lecture, Communications of the Association for Computing Machinery, 1976.

[NEWELL 1982] A. NEWELL, "The Knowledge Level", Artificial Intelligence, 18, pp 87-127, 1982.

[NICOLLE 1987] A. Nicolle-Adam, B. Victorri, C. Porquet, M. Revenu, "Métaclasses et autoréférence en Airelle pour la description et la coopération de langages", 6ème Congrès RFIA, tome 1, pp 523-528, 1987.

[NICOLLE 1990] A.Nicolle, P. Volle, J-P Barthès, P-Y Gloess, J Ferber, "Objets et Intelligence Artificielle", Actes des 3èmes journées nationales du PRC IA, Hermès, 1990.

[PIERRET-GOLDREICH et DELOUIS 1990] Christine Pierret-Goldreich et Isabelle Delouis. "TASK : Task Architecture for the Structuration of Knowledge", Avignon'90.

[SCHREIBER et al 1988] Guss Schreiber, Joost Breuker, Bert Bredeweg, Bob Wielenga, "Modelling in KBS Development", EKAW'88.

[STEELS 1990] Luc Steels, "Components of expertise", AI magazine, Summer 1990, pp 28-49.

[STEELS 1992] Luc Steels, "Reusability and configuration of applications by non-programmers". VUB AI Memo 92-4.

[VISETTI 1991] Yves-Marie Visseti, "Des systèmes experts aux systèmes à base de connaissances : A la recherche d'un nouveau schéma régulateur", Intellectica, 1991/2, 12, pp 221-279.